NUMERICAL ANALYSIS AND MULTI-PRECISION COMPUTATIONAL METHODS APPLIED TO THE EXTANT PROBLEMS OF ASIAN OPTION PRICING AND SIMULATING STABLE DISTRIBUTIONS AND UNIT ROOT DENSITIES

Liang Cao

A Thesis Submitted for the Degree of PhD at the University of St Andrews

2014

Full metadata for this item is available in Research@StAndrews:FullText at:
http://research-repository.st-andrews.ac.uk/

Please use this identifier to cite or link to this item: http://hdl.handle.net/10023/6539

This item is protected by original copyright

This item is licensed under a Creative Commons Licence
Numerical Analysis and Multi-Precision Computational Methods Applied to the Extant Problems of Asian Option Pricing and Simulating Stable Distributions and Unit Root Densities

Liang Cao

This thesis is submitted in partial fulfilment for the degree of PhD Economics and Finance at the University of St Andrews

9th July 2014
Numerical Analysis and Multi-Precision
Computational Methods Applied to the Extant
Problems of Asian Option Pricing and Simulating
Stable Distributions and Unit Root Densities

This thesis is submitted for the degree of
PhD Economics and Finance

Liang Cao
University of St Andrews
July 2014

Abstract

This thesis considers new methods that exploit recent developments in computer technology to address three extant problems in the area of Finance and Econometrics. The problem of Asian option pricing has endured for the last two decades in spite of many attempts to find a robust solution across all parameter values. All recently proposed methods are shown to fail when computations are conducted using standard machine precision because as more and more accuracy is forced upon the problem, round-off error begins to propagate. Using recent methods from numerical analysis based on multi-precision arithmetic, we show using the Mathematica platform that all extant methods have efficacy when computations use sufficient arithmetic precision. This creates the proper framework to compare and contrast the methods based on criteria such as computational speed for a given accuracy. Numerical methods based on a deformation of the Bromwich contour in the Geman-Yor Laplace transform are found to perform best provided the normalized strike price is above a given threshold; otherwise
methods based on Euler approximation are preferred.

The same methods are applied in two other contexts: the simulation of stable distributions and the computation of unit root densities in Econometrics. The stable densities are all nested in a general function called a Fox $H$ function. The same computational difficulties as above apply when using only double-precision arithmetic but are again solved using higher arithmetic precision. We also consider simulating the densities of infinitely divisible distributions associated with hyperbolic functions. Finally, our methods are applied to unit root densities. Focusing on the two fundamental densities, we show our methods perform favorably against the extant methods of Monte Carlo simulation, the Imhof algorithm and some analytical expressions derived principally by Abadir. Using Mathematica, the main two-dimensional Laplace transform in this context is reduced to a one-dimensional problem.
1. Candidate’s declarations:

I, Liang Cao, hereby certify that this thesis, which is approximately 78000 words in length, has been written by me, that it is the record of work carried out by me and that it has not been submitted in any previous application for a higher degree.

I was admitted as a research student in March 2009 and as a candidate for the degree of PhD Economics and Finance in March 2010; the higher study for which this is a record was carried out in the University of St Andrews between 2009 and 2014.

Date                                   signature of candidate

2. Supervisor’s declaration:

I hereby certify that the candidate has fulfilled the conditions of the Resolution and Regulations appropriate for the degree of PhD Economics and Finance in the University of St Andrews and that the candidate is qualified to submit this thesis in application for that degree.

Date                                   signature of supervisor

3. Permission for electronic publication: (to be signed by both candidate and supervisor)

In submitting this thesis to the University of St Andrews I understand that I am giving permission for it to be made available for use in accordance with the regulations of the University Library for the time being in force, subject to any copyright vested in the work not being affected thereby. I also understand that the title and the abstract will be published, and that a copy of the work may be made and supplied to any bona fide library or research worker, that my thesis will be electronically accessible for personal or research use unless exempt by award of an embargo as requested below, and that the library has the right to migrate my thesis into new electronic forms as required to ensure continued access to the thesis. I have obtained any third-party copyright permissions that may be required in order to allow such access and migration, or have requested the appropriate embargo below.

The following is an agreed request by candidate and supervisor regarding the electronic publication of this thesis:

Add one of the following options:

(i) Access to printed copy and electronic publication of thesis through the University of St Andrews.

Date                                   signature of candidate       signature of supervisor

A supporting statement for a request for an embargo must be included with the submission of the draft copy of the thesis. Where part of a thesis is to be embargoed, please specify the part and the reasons.
TABLE OF CONTENTS

Chapter 1: Introduction ................................................................................................... 1
  1. Motivation ................................................................................................................... 1
  2. Overview .................................................................................................................. 6

Chapter 2: Comparing New and Extant Numerical and Analytical Methods of Asian Option Pricing ........................................................................................................ 9
  1. Introduction ................................................................................................................. 9
     1.1. Literature review .................................................................................................. 9
     1.2. Motivation .......................................................................................................... 13
  2. Preliminary Material on Asian Options ..................................................................... 15
     2.1. The stock price ................................................................................................... 15
     2.2. European options ............................................................................................... 15
     2.3. Asian options ..................................................................................................... 17
     2.4. Pricing formulae for geometric Asian options ................................................... 18
     2.5. Black-Scholes equation ...................................................................................... 19
  3. Numerical and Analytical Methods .......................................................................... 21
     3.1. Preliminaries ...................................................................................................... 21
     3.2. German and Yor's Laplace transform ................................................................. 25
     3.3. Numerical inversion algorithms .......................................................................... 30
     3.4. Asymptotic method ............................................................................................ 46
     3.5. PDE method ....................................................................................................... 48
     3.6. Spectral series expansion ................................................................................... 52
     3.7. Constructive complex analysis ........................................................................... 55
Chapter 3: A Comparison of Various Methods for Computing Stable Distributions and Infinitely Divisible Distributions Associated with Hyperbolic Functions

1. Introduction

2. Simulation of Brownian Motion, Bridge and Excursion
2.1. Brownian motion ................................................................. 97
2.2. Brownian bridge ................................................................. 98
2.3. Brownian meander and excursion ......................................... 99
3. Infinitely Divisible Distributions Associated with Hyperbolic Functions .... 101
   3.1. Definition of distributions ............................................... 101
   3.2. Properties of $C_t$, $S_t$ and $T_t$ ..................................... 102
   3.2. Laws of $\tilde{C}_t$, $\tilde{S}_t$ and $\tilde{T}_t$ ......................... 107
4. Numerical Experiments with $C_t$, $S_t$ and $T_t$ ..................... 112
   4.1. Computing the density of $C_t$ ....................................... 112
   4.2. Computing the density of $S_t$ ....................................... 117
   4.3. Computing the density of $T_t$ ....................................... 122
5. Stable Distributions .............................................................. 126
   5.1. The Definition of $S(\alpha, \beta, \mu, \sigma)$ ............................ 126
   5.2. The Definition of $S_1(\alpha, \beta_1, \mu, \sigma_1)$ ..................... 127
   5.3. Generation of stable random variables $S_1(\alpha, \beta_1)$ ........ 129
   5.4. Special functions related to stable distributions ................ 130
   5.5. Densities of various stable distributions .......................... 131
6. Numerical Experiments with Stable Distributions .................. 141
   6.1. Computing the density $g_{m,\alpha}$ .................................. 141
   6.2. Computing the density $f_{m,\alpha}$ of generalized one-sided stable distribution $F_{m,\alpha}$ 142
   6.3. Computing the density $f_\alpha$ of one-sided stable distribution $F_\alpha$ ............. 146
   6.4. Computing the density $f_{\alpha,\beta}$ of two-sided stable distribution $F_{\alpha,\beta}$ ....... 150
7. Conclusion .............................................................................. 153

Chapter 4: Numerical Computation of Unit Root Distributions ............ 155
1. Introduction and Motivation ................................................................................... 155

2. Unit Root Problem in AR(1) Model ....................................................................... 160
   2.1. Stationary case ................................................................................................. 160
   2.2. Nonstationary case ........................................................................................... 161
   2.3. Limiting distributions....................................................................................... 163
   2.4. Test unit root hypothesis .................................................................................. 165

3. Simulate the Distributions of $S_{IT}$, $S_{2T}$, $S_{3T}$ and $S_{4T}$ ................................. 167

4. Computing the Distribution of $S_3$ from the Characteristic Function .............. 172
   4.1. Imhof’s formula for $S_3$ .................................................................................. 172
   4.2. Evaluating the characteristic function associated with $S_3$ ......................... 173
   4.3. Computing the distribution function of $S_3$ .................................................. 174
   4.4. Computing the probability density of $S_3$ ......................................................... 177

5. Joint Density of $U$ and $V$ .................................................................................... 185
   5.1. The Laplace transform of the joint distribution $(U, V)$ ................................. 186
   5.2. Computation of the density $f_V(v)$ ................................................................. 187
   5.3. Computation of the density $f_U(u)$ ................................................................ 188
   5.4. Computation of the joint density $f_{U,V}(u, v)$ ................................................. 193

6. Joint Density of $R$ and $S$ ..................................................................................... 199
   6.1. Standardized quadratic forms .......................................................................... 199
   6.2. Formulae for joint density $f_{R,S}(r, s)$ ........................................................... 200
   6.3. Formulae for joint distribution function $F_{R,S}(r, s)$ ...................................... 204

7. Generating Densities of Unit Root Statistics ...................................................... 206
   7.1. Transformation between $f_{U,V}(u, v)$ and $f_{R,S}(r, s)$ .................................... 206
   7.2. Generating the density of $S_3$ from $f_{U,V}(u, v)$ ............................................ 208
   7.3. Generating the density of $S_3$ from $f_{R,S}(r, s)$ ............................................. 213
7.4. Generating the density of $S_4$ from $f_{U,Y}(u,v)$ ........................................ 217
7.5. Generating the density of $S_4$ from $f_{R,S}(r,s)$ ........................................ 222
8. Conclusion ............................................................................................................. 227

Chapter 5: Conclusion .............................................................................................. 229
1. Contributions of the Thesis .................................................................................... 229
   1.1. Chapter 2 ........................................................................................................ 229
   1.2. Chapter 3 ........................................................................................................ 230
   1.3. Chapter 4 ........................................................................................................ 231
   1.4. Mathematica code ......................................................................................... 232
2. Suggestions for Further Work ............................................................................... 233

Appendices .................................................................................................................. 235
Bibliography ............................................................................................................... 237

Mathematica Code ..................................................................................................... 241
1. Code in Chapter 2 ................................................................................................. 241
2. Code in Chapter 3 ................................................................................................. 286
3. Code in Chapter 4 ................................................................................................. 308
Chapter 1

Introduction

1. Motivation

The Laplace transform is an integral transform which is widely used in finance and econometrics to, for example, characterize the Asian option price, define probability distributions and represent the joint density related to unit root distributions. This thesis seeks to apply state-of-the-art methods of Laplace transform inversion and other numerical methods in an explicit multi-precision computing environment to examine three problems that have endured in various fields: the problem of pricing an Asian option; the problem of computing and simulating stable distributions and densities (we also consider other infinitely divisible distributions associated with hyperbolic functions); and the problem of computing and simulating unit root densities in Econometrics.

Let \( f(t) \) be a real-valued function of a real variable \( t > 0 \). The Laplace transform of \( f(t) \) is defined as

\[
\hat{f}(s) = \int_0^\infty e^{-st} f(t) \, dt
\]

where \( s \) is a complex variable. It transforms the function \( f(t) \) in time domain to the function \( \hat{f}(s) \) in complex domain by integration with the kernel \( e^{-st} \).

Assume the above Laplace transform is well-defined and analytic for \( \text{Re}(s) > 0 \). This ensures the region of convergence of \( \hat{f}(s) \) covers the right half plane. This assumption is natural for a large class of applications, but can be generalized for others by making a change of variables. Then, the inverse Laplace transform also known as Bromwich integral is given by

\[
f(t) = \frac{1}{2\pi i} \int_{\sigma-i\infty}^{\sigma+i\infty} e^{st} \hat{f}(s) \, ds
\]
where the integration is done along a vertical line \( s = a \) such that all singularities of \( \hat{f}(s) \) are to the left of the contour path. This keeps the contour path in the region of convergence of \( \hat{f}(s) \).

The Laplace transform can be applied in many areas. In finance, it is difficult to evaluate an Asian option because the payoff of the Asian option depends on the average price of the underlying stock over a prespecified period. The distribution of the average is too complicated to be characterized analytically. Turnbull and Wakeman (1991) approximate the distribution of the average analytically by matching the first two moments to the lognormal distribution, while Milevsky and Posner (1998) match them to the reciprocal gamma distribution. The problem with analytical approximations is that there is no reliable error estimates (Linetsky, 2004).

Geman and Yor (1993) have derived a closed-form expression for the Laplace transform of the normalized Asian call price:

\[
\hat{C}(\lambda, q) = \int_{0}^{\infty} e^{-\lambda h} C_v(h, q) \, dh = \frac{\Gamma(1/2) \Gamma(\nu/2-2) \Gamma(\mu/2-1)}{\Gamma(\nu/2)} \int_{0}^{\infty} e^{-x} x^{\nu-2} (1-2q) x^2 \, dx
\]

where \( \mu = \sqrt{2\lambda + \nu^2} \) and the normalized interest rate \( \nu \), the normalized maturity \( h \), and the normalized strike price \( q \) are given by

\[
\nu = \frac{2(r-\delta)}{\sigma^2} - 1, \quad h = \frac{\sigma^2}{4} (T-t), \quad q = \frac{\sigma^2}{4S(0)} [K T - \int_{0}^{t} S(u) \, du]
\]

where \( r \) is the constant risk-free interest rate, \( \delta \) is the constant dividend yield, \( \sigma \) is the constant volatility, \( K \) is the strike price, \( T \) is the time to maturity, and \( S(t) \) is the stock price at present time \( t \). \( \int_{0}^{t} S(u) \, du \) divided by \( t \) stands for the realized average price over the time interval \([0, t]\). The process of numerically inverting the Geman and Yor (1993) Laplace transform will lead to the normalized Asian call price from which the Asian call price can be computed with ease. As we shall explain in Chapter 2, the Asian put option price is related to the call price by the notion of “put-call parity”. Fu, Madan and Wang (1999) and Craddock, Heath and Platen (2000) discuss and compare various inversion algorithms for the problem of Asian option pricing. They show inversion of the Laplace transform encounters numerical difficulties for low volatility and short maturity, and therefore do not recommend it as a method of
pricing an Asian option. But, they have not considered the factor of computation precision which determines the round-off errors. Also, given that computers has now become much more powerful than a decade ago, the conclusion would be different.

The infinitely divisible distributions of non-negative random variables $C_t$, $S_t$ and $T_t$ can be characterized by Laplace transforms (Pitman and Yor, 2003): for $t \geq 0$,

$$E[e^{-\lambda C_t}] = \int_0^\infty e^{-\lambda x} f_{C_t}(x) \, dx = \left( \frac{1}{\cosh \sqrt{2} \lambda} \right)^t$$

$$E[e^{-\lambda S_t}] = \int_0^\infty e^{-\lambda x} f_{S_t}(x) \, dx = \left( \frac{\sqrt{2} \lambda}{\sinh \sqrt{2} \lambda} \right)^t$$

$$E[e^{-\lambda T_t}] = \int_0^\infty e^{-\lambda x} f_{T_t}(x) \, dx = \left( \frac{\tanh \sqrt{2} \lambda}{\sqrt{2} \lambda} \right)^t$$

(1.5)

The laws of $C_t$ and $S_t$ occur naturally in the study of Brownian motion and Bessel processes (Yor, 1997, §18.6 cited in Pitman and Yor, 2003). The distribution of $C_{1/2}$ arises when studying the Dickey-Fuller distributions. Analytical formulae have been derived for the density of $C_t$ for general $t$ (Biane, Pitman and Yor, 2001), the density of $S_t$ for general $t$ (Biane and Yor, 1987 cited in Biane, Pitman and Yor, 2001), the densities of $C_1$ and $S_1$ (Devroye, 2009a), and the density of $S_{1/2}$ (Tolmatz, 2002). Note the formula for the density of $S_t$ is intractable other than $t = 1$. Also, the formula for the density of $T_t$ is not available. If the Laplace transforms of $C_t$, $S_t$ and $T_t$ can be inverted numerically, the densities of $C_t$, $S_t$ and $T_t$ for general $t$ will be obtained.

The stable distributions are a class of distributions such that a linear combination of two i.i.d. stable random variables has the same distribution up to location and scale parameters. There are many types of stable distributions including $S(\alpha, \beta, \mu, \sigma)$, $S_1(\alpha, \beta_1, \mu, \sigma_1)$, the generalized one-sided stable distribution $F_{m,a}$, the one-sided stable distribution $F_a$ and the two-sided stable distribution $F_{a,\beta}$. The first two stable distributions can be conveniently described by their characteristic functions. While, $F_{m,a}$, $F_a$ and $F_{a,\beta}$ discussed in the paper by Schneider (1987) are defined by some probability density $g_{m,a}$, the Laplace transform and the Fourier transform respectively. Let $f_a$ be the density of $F_a$. The Laplace transform of $f_a$ is given by

$$\int_0^\infty e^{-\lambda x} f_a(x) \, dx = e^{-\lambda a}$$

(1.6)

Let $f_{m,a}$ be the density of $F_{m,a}$. Schneider (1987) obtains the Laplace transform of $f_{m,a}$ which can be
expressed in terms of the Fox function. In the special case of \( m = 1, 2 \), the Laplace transform of \( f_{m,a} \) has a closed-form expression.

For \( m = 1 \),

\[
\int_0^\infty e^{-\lambda x} f_{1,a}(x) \, dx = e^{-\left(\frac{\lambda}{b}\right)^{1/\alpha}}
\]

(1.7)

where \( b = \left( \frac{\alpha}{\Gamma(1-\alpha)} \right)^{1/\alpha} \).

For \( m = 2 \),

\[
\int_0^\infty e^{-\lambda x} f_{2,a}(x) \, dx = \frac{2}{\Gamma(\beta)} \left( \frac{\lambda}{b}\right)^{1/2} K_\beta\left(2 \left( \frac{\lambda}{b}\right)^{1/2}\right)
\]

(1.8)

where \( \beta = \frac{1}{1+\alpha} \) and \( K_\beta(z) \) is the modified Bessel function of the second kind.

The distribution \( S(\alpha, \beta, \mu, \sigma) \) can be computed or simulated by *Mathematica 9.0* built-in function *StableDistribution*[]. The distribution \( S_1(\alpha, \beta_1, 0, 1) \) can be simulated by a recipe proposed by Chambers, Mallows and Stuck (1976). Schneider (1987) derives Fox function representations and series expansions for the densities \( f_{m,a}, f_a \) and \( f_{a,\beta} \) where \( f_{a,\beta} \) is the density of \( F_{a,\beta} \). Alternative series expansions for \( f_{a,\beta} \) can be found in the Feller’s (1970, p.583) text. Penson and Gorska (2011) show the density \( f_a \) for rational \( \alpha = l/k \) can be written as a finite sum of generalized hypergeometric functions. Schneider (1986) and Garoni and Frankel (2002) give special function representations for the special cases of \( f_{a,\beta} \), which correct \( f_{2/3,0} \) and recover \( f_{1,2,0} \) discussed by Zolotarev (1954). The density \( f_{a,\beta} \) can also be computed by numerical inversion of the Fourier transform. By numerical inversion of the associated Laplace transforms, we can obtain the densities \( f_{m,a} \) and \( f_a \). But, the complete relations between \( S(\alpha, \beta, \mu, \sigma) \), \( S_1(\alpha, \beta_1, \mu, \sigma_1) \), \( F_{m,a} \), \( F_a \) and \( F_{a,\beta} \) has been lacking, and is therefore worth studying.

When studying the first order autoregressive time series under a unit root, the so-called Dickey-Fuller distributions of random variables \( S_3 \) and \( S_4 \) are of interest to many researchers. As the limiting cases of unit root statistics \( S_3 T \) and \( S_4 T \) with the notation used in Tanaka (1996), \( S_3 \) and \( S_4 \) can be expressed in terms of functionals of Brownian motion:
\[ S_1 = \frac{U}{V} \]
\[ S_4 = \frac{U}{\sqrt{V}} \]

with

\[ U = \int_0^1 \omega(t) \, d\omega(t) = \frac{1}{2} \left[ \omega^2(1) - 1 \right] \]
\[ V = \int_0^1 \omega^2(t) \, dt \]

(1.10)

where \( \omega(t) \) is a standard Brownian motion with \( t \in [0, 1] \). The distributions of \( S_3 \) and \( S_4 \) were first approximated by Monte Carlo simulation with finite samples (Fuller, 1976 cited in Tanaka, 1996, p.17), but the approximations are usually poor especially on the tails. The distribution of \( S_3 \) was computed numerically from the associated characteristic function (Tanaka, 1996). However, the problem of computing the Dickey-Fuller distributions properly remains unsolved. We notice that the densities of \( S_3 \) and \( S_4 \) can be constructed from the joint density \( f_{U,V}(u, v) \) of \( (U, V) \) provided that the joint density is readily available.

White (1958, p.1193) have derived the Laplace transform of \( (U, V) \)

\[ \tilde{f}(\alpha, \beta) = \mathbb{E} \left[ \exp(-\alpha U - \beta V) \right] \]
\[ = \int_0^\infty \int_{-1/2}^{1/2} e^{-\alpha u - \beta v} f_{U,V}(u, v) \, du \, dv \]
\[ = e^{-\alpha^2/2} \left( \cosh \sqrt{2 \beta} + \frac{\alpha}{\sqrt{2 \beta}} \sinh \sqrt{2 \beta} \right)^{-1/2} \]  

(1.11)

As we can see, the joint density \( f_{U,V}(u, v) \) is embedded in the above Laplace transform which is two-dimensional. Theoretically, numerical inversion of two-dimensional Laplace transforms will result in the joint density \( f_{U,V}(u, v) \). But this needs to be verified by numerical experiments.

Nonetheless, numerical inversion of Laplace transforms is non-trivial. It involves a lot of effort, very careful selection for inversion parameters, and specification for computation precision. In general, the difficulty of the inversion mainly depends on the Laplace transform to be inverted and inversion algorithm used. The inversion algorithms to be considered in this thesis include the Euler method (Euler) and the Post-Widder method (PW) proposed by Abate and Whitt (1995); the Laguerre method (Laguerre) suggested by Abate, Choudhury and Whitt (1996); the Bromwich integral (Bromwich) applied by Shaw (1998); the fixed Talbot method (FT) and the Gaver-Wynn-Rho algorithm (GWR) presented by Abate and Valkó (2004); and three inversion routines in the unified framework (Abate and
Whitt, 2006): the unified Gaver-Stehfest algorithm (UniG), the unified Euler algorithm (UniE), and the unified Talbot algorithm (UniT). Among them, UniG, UniE and UniT can be combined to form nine different two-dimensional inversion algorithms (Abate and Whitt, 2006): UniTG, UniTT, UniEG, UniET, UniTE, UniGT, UniGG, UniEE and UniGE with first operator, say $T$, applying to the outer loop and the second operator, say $G$, applying to the inner loop.

2. Overview

This thesis investigates a series of methods for numerically inverting Laplace transforms and applies them to various problems. The inversion technique is also compared with other methods such as Monte Carlo simulation, PDE method, analytical formulae and so on. We write Mathematica code for each method. The numerical experiments are conducted in Mathematica 9.0 on a HP ProBook 4520s laptop equipped with 2.4GHz Intel Core i3-370M processor and 3GB DDR3 RAM.

Chapter 2 deals with the problem of Asian option pricing. Inversion algorithms such as Euler, PW, Laguerre, Bromwich, FT, GWR, UniG, UniE, and UniT are used to invert the Geman and Yor (1993) Laplace transform. All computations are done with arbitrary-precision arithmetic rather than machine-precision arithmetic where the latter is used by Fu, Madan and Wang (1999) and Craddock, Heath and Platen (2000). Hence, the round-off errors can be controlled properly: round-off errors do not propagate when one increases parameter values of the inversion algorithms. Eventually, by specifying appropriate parameter values for each method we find all inversion algorithms achieve the same result. The matched result can be computed to have high accuracy so that we can use it as the reference price to check the accuracy of each method with its parameter settings. By manipulating the parameter settings for a method, we can also see how the accuracy changes with different parameter values. It is found that the truth of difficulties with numerical inversion for Asian option pricing is that the accuracy of the algorithm drops and the computation time increases when the volatility is low and the maturity is short. But we can regain the accuracy by increasing both the parameter values of the algorithm and the computation precision at the cost of extra computation time.

In addition to numerical inversion, we consider many other methods for pricing an Asian option. These methods include Monte Carlo simulation, analytical approximations (Turnbull and Wakeman, 1992;
Milevsky and Posner, 1998), PDE method (Rogers and Shi, 1995; Večeř, 2001; 2002), asymptotic expansions (Shaw, 2002), spectral expansions (Linetsky, 2004) and constructive complex analysis (Schröder, 2008). With the reference prices, the accuracy of all methods can be verified.

Chapter 3 investigates various methods for computing infinitely divisible distributions and stable distributions. Following Biane, Pitman and Yor (2001), we derive the formula for the density of $S_t$ for general $t$ using Mathematica. The exact relations between different types of stable distributions are established. Inversion algorithms UniG, UniE and UniT are applied to numerically inverting the Laplace transforms associated with the infinitely divisible distributions and stable distributions with the yielded results compared with those of analytical formulae. We show that UniG is a universal method for the densities of $S_t, C_t$ and $T_t$ for general $t > 0$, while UniE and UniT only work for the densities of $S_t$ and $C_t$ for integer $t > 0$ though they can compute the density of $T_t$ for general $t > 0$. When only the density of $T_t$ is concerned, UniE and UniT become superior to UniG in terms of the speed. With regard to the computations of $f_{m,p}$ and $f_o$, UniE, UniT and UniG are all able to invert their Laplace transforms with UniT faster than the other two.

Chapter 4 studies the unit root distributions $S_3$ and $S_4$ arising in AR(1) model with $S_3 = U/V$ and $S_4 = U/\sqrt{V}$, and seeks to compute the densities of $S_3$ and $S_4$ in a numerical way rather than based on the approximations of Monte Carlo simulation. Following Tanaka (1996), we compute the density of $S_3$ from the associated characteristic function using Imhof’s (1961 cited in Tanaka, 1996, p.196) formula. However this method cannot be applied to the computation of the density of $S_4$. Then, we attempt to construct the densities of $S_3$ and $S_4$ from the joint density $f_{U,V}(u, v)$. Given the original two-dimensional Laplace transform of $(U, V)$, we reduce it to a one-dimensional Laplace transform of $f_{U,V}(u, v)$ with respect to $v$. It is shown that the reduced Laplace transform can be inverted by UniG. With the numerical results of $f_{U,V}(u, v)$, we are able to generate not only the densities of $S_3$ and $S_4$ but also the density of almost any unit root distribution which can be expressed in terms of $U$ and $V$ only. In addition, Abadir (1995) proposes analytical formulae for the joint density $f_{R,S}(r, s)$ of $(R, S)$ with $R = \sqrt{2}U$ and $S = 2V$. We establish a relation between $f_{U,V}(u, v)$ and $f_{R,S}(r, s)$, and demonstrate $f_{R,S}(r, s)$ can also be used for the generation of unit root distributions. By comparing results computed from $f_{U,V}(u, v)$ and
those computed from $f_{RS}(r, s)$, we find that using $f_{UV}(u, v)$ is better than using $f_{RS}(u, v)$.

Chapter 5 summarizes the contributions made in this thesis and offers some suggestions for future work.
Chapter 2
Comparing New and Extant Numerical and Analytical Methods of Asian Option Pricing

1. Introduction
   - 1.1. Literature review

Asian options are financial derivatives whose payoffs are based on the average price of the underlying asset over some prescribed period of time. Owing to the averaging feature, Asian options are widely used in the stock market, the foreign exchange market and the commodity market. There are several reasons why they are popular. First, the companies with sales in foreign currency are faced with the unanticipated changes in the exchange rate between the foreign currency and the home currency. Asian options provide them with an option of using the average foreign exchange rate to avoid the foreign exchange risk. Second, it may be possible for large market participants to manipulate the prices in thinly traded markets such as the crude oil market and the gold market. But it is much harder for them to manipulate the average price with the existence of Asian options. Thirdly, Asian options are usually cheaper than the ordinary options because the volatility of the average price is lower than the volatility of the price itself.

Asian options receive their name Asian because they were first priced in Tokyo by David Spaughton and Mark Standish of Bankers Trust who were there in 1987 on business developing ‘the first commercially...
used pricing formula for options linked to the average of crude oil’ (Wilmott, 2006, p.427). There are many types of Asian options depending on whether the option is a call or a put, where the average price is used, what type the average is, and how to sample the underlying price over a time period. The average price can be used either in place of the underlying price (an average price option) or in place of the strike price (an average strike option). The average can be arithmetic or geometric. The sampling can also be continuous or discrete. In this chapter, we mainly focus on the valuation of the continuously sampled arithmetic average price call option (or continuous arithmetic Asian call in short) because it is the case of greatest importance in terms of being type of Asian option that is traded most. It has also been the type of Asian option most focused on in the academic literature.

The problem of pricing arithmetic Asian options has interested many researchers for more than two decades as there are no closed-form formulae for them. The arithmetic average is not lognormally distributed (McDonald, 2006, p.593) when the stock price follows geometric Brownian motion as in the traditional framework in finance, the Black-Scholes framework, following from assumptions laid out by Black and Scholes (1973). See, e.g. McDonald (2006, pp.649-650) or some other similar textbook. In contrast to arithmetic average, the geometric average is lognormally distributed since the product of lognormal random variables is also lognormal (McDonald, 2006, p.593). Therefore, closed-form formulae exist for geometric Asian options.

The continuous arithmetic Asian call options were in the past valued using Monte Carlo simulation with the control variate method, which is a variance reduction technique, by Kemna and Vorst (1990), where they notice the continuous geometric Asian call serves as a quality control variate. Boyle, Broadie and Glasserman (1997) further improve the equation for the control variate estimate of which the variance can be minimized. Although Monte Carlo method is straightforward for path-dependent options such as Asian options, it is often computationally expensive. Also, discretely sampling a continuous time process results in the discretization bias (Broadie, Glasserman and Kou, 1999). Then, analytical approximations are used in the valuation of Asian options. Turnbull and Wakeman (1991) approximate the distribution of the arithmetic average by matching the first two moments to the lognormal distribution. Later, Milevsky and Posner (1998) match them to the reciprocal gamma distribution. Analytical approximations are fast but have no reliable error estimates (Linetsky, 2004).
An important contribution is made by Geman and Yor (1993) who derive a closed-form expression for the Laplace transform of the continuous arithmetic Asian call price. The call price \( C_t, r(K) \) can be expressed as

\[
C_t, r(K) = \frac{e^{(r-\delta)T}}{T} \left( \frac{4S_0}{\sigma^2} \right) C^{(\nu)}(h, q)
\]

with \( S(t) \) the stock price at time \( t \), \( K \) the strike price, \( T \) the time to maturity, \( r \) the constant risk-free interest rate, and \( \sigma \) the constant volatility. The normalized interest rate \( \nu \), the normalized maturity \( h \), and the normalized strike price \( q \) are given by

\[
\nu = \frac{2(r-\delta)}{\sigma^2} - 1
\]

\[
h = \frac{\sigma^2}{4}(T-t)
\]

\[
q = \frac{\sigma^2}{4S_0} \left[ K T - \int_0^T S(u) \, du \right]
\]

where \( \delta \) is the constant dividend yield, and \( \int_0^T S(u) \, du \) divided by \( t \) stands for the realized average price over the time period \([0, t]\). The Laplace transform of the normalized Asian call price \( C^{(\nu)}(h, q) \) with respect to \( h \) is given by

\[
\hat{C}(\lambda, q) = \int_0^\infty e^{-\lambda h} C^{(\nu)}(h, q) \, dh
\]

\[
= \frac{1}{\lambda^{-\nu/2}} \Gamma\left(\frac{\nu}{2}\right) \Gamma\left(\frac{\mu-\nu}{2}\right) \Gamma\left(\frac{\mu}{2}\right)
\]

where \( \mu = \sqrt{2 \lambda + \nu^2} \).

The Geman-Yor Laplace transform \( \hat{C}(\lambda, q) \) makes it possible to price Asian options by numerical inversion of the Laplace transform. Geman and Eydeland (1995) first invert the Geman-Yor Laplace transform using the fast Fourier transform (FFT). Shaw (1998, p.208) evaluates the Geman-Yor Laplace transform in Mathematica revealing that \( \hat{C}(\lambda, q) \) can be expressed in terms of the Kummer confluent hypergeometric function as discussed below. He further exploits built-in Mathematica function NIntegrate[] and inverts \( \hat{C}(\lambda, q) \) based on the Bromwich integral (which is the contour integral involved in Laplace transform inversion as discussed below). Fu, Madan and Wang (1999) price Asian options using the Euler method and the Post-Widder method, which are two inversion algorithms proposed by Abate and Whitt (1995). They compare and contrast numerical inversion methods with Monte Carlo simulation, and find that numerical inversion techniques encounter numerical difficulties for low
volatility and short maturity, or specifically for $\sigma^2(T-t) < 0.01$. Craddock, Heath and Platen (2000) investigate and compare different approaches to the numerical inversion of the Geman-Yor Laplace transform, and draw a similar conclusion that numerical inversion methods can be extremely time-consuming and unreliable when the normalized strike price $q$ is small, i.e. $q < 0.03$. There are a number of important inversion algorithms in the numerical analysis literature that have not yet been applied to Asian option pricing. These methods include the Laguerre method proposed by Abate, Choudhury and Whitt (1996); the Gaver-Wynn-Rho algorithm and the fixed Talbot algorithm by Abate and Valkó (2004); and the unified Gaver-Stehfest algorithm, the unified Euler algorithm, and the unified Talbot algorithm by Abate and Whitt (2006). One of the contributions of this thesis is to apply these algorithms to Asian option pricing. Given they are currently the most effective algorithms for Laplace transform inversion, it is not surprising that we are able successfully to apply them to Asian option pricing, yielding computational methods that are competitive with, and sometimes supersede, the currently best extant methods in the literature.

The value of the continuous arithmetic Asian option can be characterized by a partial differential equation (PDE) (see Wilmott, 2006, p.431). Hence, the price of an Asian option can be found by solving the partial differential equation. Rogers and Shi (1995) derive a one-dimensional PDE for both fixed and floating strike Asian options, but this PDE is difficult to solve numerically (Večeř, 2002). Večeř (2001; 2002) provides an alternative one-dimensional PDE for both continuous and discrete arithmetic Asian options. Večeř claims that his PDE can be easily implemented to give very fast and accurate results.

Other methods include asymptotic method for low volatility proposed by Shaw (2002), spectral expansion by Linetsky (2004), and constructive complex analysis by Schröder (2008). Among them, Shaw (2002) approximates the Mellin transform of the Laplace transform by exploiting an asymptotic expansion of the quotient of Gamma functions. Linetsky (2004) attacks the pricing problem directly by using an identity in law between the integral of geometric Brownian motion and the state of a one-dimensional diffusion process, and developing spectral expansion for the Asian put price. The Asian call price can be recovered by using the put-call parity. Schröder (2008) develops a two-stage approach to the valuation of Asian options which first expresses the normalized Asian call price as a single contour integral in terms of Hermite functions, and then represents the integral by series and asymptotic
expansions with error estimates.

1.2. Motivation

The papers by Fu, Madan and Wang (1999) and Craddock, Heath and Platen (2000) are of central importance in the literature because they have become seen as the benchmark in terms of an experimental design that any method of Asian option pricing can be evaluated against. That said, their own experiments were conducted in a fixed machine-precision computing environment and, as we shall see, this can be severely limiting. As we demonstrate below, this is because there comes a point where the round-off errors in calculations begin to propagate when more and more accuracy is forced upon the pricing problem. Another two types of errors which are not properly handled are truncation errors and discretization errors. The truncation errors occur when an infinite series is truncated, while the discretization errors arise when an integral is approximated by the trapezoidal rule. A fact which both papers fail to notice is that the round-off errors may rise correspondingly in the machine-precision computing environment when one tries to increase algorithm parameters to reduce truncation errors and discretization errors. In other words, the accuracy of the results is improved consistently with the truncation size only when the round-off errors and the discretization errors are controlled.

Regarding the computation time, Fu, Madan and Wang (1999) and Craddock, Heath and Platen (2000) conclude that numerical inversion of the Geman-Yor Laplace transform is unreliable and time-consuming, especially for low volatilities and short maturities. This is true when machine-precision arithmetic is used and the computations are done with outdated computing software and a fifteen-year old computer. But, the results may be different when we use the more powerful CPU and the ultimate application for computations, Mathematica.

This chapter draws a comparison of various numerical and analytical methods for Asian option pricing. These methods include numerical inversion, the PDE method by Večer (2002), asymptotic method by Shaw (2002), spectral expansion by Linetsky (2004), constructive complex analysis by Schröder (2008), the Turnbull and Wakeman (1991) approximation, the Milevsky and Posner (1998) approximation, and Monte Carlo simulation. The algorithms considered for numerical inversion cover the Euler method, the Post-Widder method, Bromwich integration, the Gaver-Wynn-Rho algorithm, the fixed Talbot algorithm, the unified Gaver-Stehfest algorithm, the unified Euler algorithm, the unified Talbot
algorithm, and the Laguerre method. We use Mathematica 9.0 to perform numerical experiments, and write Mathematica codes for these methods (with the exception of asymptotic method). All computations are done with arbitrary-precision arithmetic on a laptop equipped with 2.4GHz Intel Core i3-370M processor and 3GB DDR3 RAM.

The rest of this chapter is organized as follows. Section 2 defines Asian options in more detail and states the basic problem of Asian option pricing we consider (in the Black-Scholes framework). Section 3 considers various approaches to Asian option pricing both methods that already exist in the literature and new methods we propose that apply recent state-of-the-art work relevant to our problem. In Sections 2 and 3, there are many methods to consider and so, by necessity, these sections are long. But they have to be if we want our treatment to be taxonomic and encompassing. One contribution of this thesis is in collecting and in some cases recognizing, and then organizing in one place, relevant methods for Asian option pricing from the Finance and Computational Statistics literature. A second contribution is to provide dedicated Mathematica code to implement each and every method that is discussed. The first key conclusion is that in a fixed machine-precision environment, all methods have their drawbacks in some regions of the parameter space. This result is widely known (and indeed this is the reason why the Asian option pricing problem has been so enduring) but here we provide the reason why this is so. Turning the problem on its head, we then show in a multi-precision environment, where the computing precision is allowed to vary in accordance with the method in hand, that all methods have efficacy provided the computing precision is large enough. This is our second key conclusion. Comparing and contrasting the various methods then becomes, properly, an exercise in assessing them on other criteria, such as the speed of computation to achieve a given accuracy. Some results are reported in Section 4. Section 5 concludes.
2. Preliminary Material on Asian Options

2.1. The Stock price

In the Black-Scholes option-pricing model, the stock price is assumed to follow the geometric Brownian motion (Black and Scholes, 1973 cited in McDonald, 2006, pp.649)

\[
\frac{dS(t)}{S(t)} = \alpha dt + \sigma dW(t)
\]

(2.1)

where \(S(t)\) is the stock price, \(dS(t)\) the instantaneous change in the stock price, \(\alpha\) the expected return on the stock, \(\sigma\) the volatility of the stock, and \(W(t)\) the value of a Brownian motion at time \(t\).

Brownian motion is a continuous stochastic process which has the following properties (McDonald, 2006, p.650):

- \(W(0) = 0\). Brownian motion always starts at the origin.
- \(W(t+s) - W(t)\) is normally distributed with mean zero and variance \(s\). Mathematically, \(W(t+s) - W(t) \sim \sqrt{s} N(0, 1)\) where \(N(0, 1)\) is a standard normal distribution.
- \(W(t+s_1) - W(t)\) and \(W(t) - W(t-s_2)\) are independently distributed for \(s_1, s_2 > 0\).
- \(W(t)\) is continuous everywhere.

Brownian motion \(W(t)\) is a martingale which means \(E[W(t+s) \mid W(t)] = W(t)\). The process \(W(t)\) is also called a diffusion process.

In the risk-neutral pricing, (2.1) is written as (see McDonald, 2006, p.661):

\[
\frac{dS(t)}{S(t)} = r dt + \sigma d\tilde{W}(t)
\]

(2.2)

where \(r\) is the risk-free interest rate, and \(\tilde{W}(t)\) is another Brownian motion. If the stock pays dividends at the continuous rate \(\delta\), we should replace \(r\) with \(r - \delta\).

The geometric Brownian motion is a stochastic differential equation (SDE). The solution to it is given by (see McDonald, 2006, p.596)

\[
S_t = S_0 e^{\left(r-\frac{1}{2} \sigma^2\right) t + \sigma W(t)}
\]

\[
= S_0 e^{\left(r-\frac{1}{2} \sigma^2\right) t + \sigma \sqrt{T} Z}
\]

(2.3)

where \(Z\) is a standard normal random variable, i.e. \(Z \sim N(0, 1)\). Equation (2.3) implies the stock price follows a lognormal distribution.

2.2. European options

European options can only be exercised at expiration. There are two types of European options: call options and put options. A call option gives the buyer the right but not the obligation to buy the
underlying stock for the strike price. While, a put option gives the owner the right but not the obligation to sell the underlying stock for the strike price.

Let $S(t)$ be the stock price at time $t$, $K$ be the strike price, and $T$ be the time to maturity. The payoff of a call option is

$$\text{Call payoff} = (S(T) - K, 0)^+$$ (2.4)

where $(x, 0)^+$ denotes the maximum of $0$ and $x$.

The payoff of a put option is

$$\text{Put payoff} = [K - S(T), 0]^+$$ (2.5)

Then, the price of the option can be obtained by discounting the payoff of the option to the current time.

The famous Black-Scholes formula is derived by Black and Scholes (1973) and Merton (1973) for pricing European options. McDonald (2006, Ch.12) shows the Black-Scholes formula is a limiting case of the binomial formula for the price of a European option. The price of a European call option on a stock is given by the Black-Scholes formula

$$C(S, K, \sigma, r, T, \delta) = S e^{-\delta T} N(d_1) - K e^{-r T} N(d_2)$$ (2.6)

where

$$d_1 = \frac{\ln(S/K) + (r - \delta + \frac{1}{2} \sigma^2) T}{\sigma \sqrt{T}}$$

$$d_2 = d_1 - \sigma \sqrt{T}$$ (2.7)

The Black-Scholes formula has six input parameters: $S$ the current price of the stock, $K$ the strike price of the option, $r$ the continuously compounded risk-free interest rate, $\delta$ the continuously compounded dividend yield on the stock, $\sigma$ the volatility of the stock, and $T$ the time to maturity of the option. $N(x)$ is the cumulative normal distribution function of the standard normal distribution.

The Black-Scholes formula for a European put option is

$$P(S, K, \sigma, r, T, \delta) = K e^{-r T} N(-d_2) - S e^{-\delta T} N(-d_1)$$ (2.8)

where $d_1$ and $d_2$ are the same as those in the Black-Scholes formula for a European call option.

The Black-Scholes call price is related to the Black-Scholes put price by the put-call parity (see
McDonald, 2006, p.378)

\[ P(S, K, \sigma, r, T, \delta) = C(S, K, \sigma, r, T, \delta) + K e^{-rT} - S e^{-\delta T} \]

(2.9)

2.3. Asian options

An Asian option is a path-dependent option of which the payoff depends on the average price of the underlying asset over some period of time. There are many types of Asian options depending on whether the option is a call or a put, whether the average is arithmetic or geometric, whether the average is continuously sampled or discretely sampled, and whether the average is used in place of the stock price or the strike price.

Assume the stock price \( S_t \) is evenly sampled \( N + 1 \) times over the time interval \([0, T]\). The time period is then \( h = T / N \). McDonald (2006, p.446) defines the discretely sampled arithmetic average as

\[ A_d(T) = \frac{1}{N+1} \sum_{i=0}^{N} S_{ih} \]

(2.10)

and the discretely sampled geometric average as

\[ G_d(T) = (S_0 \times S_h \times \cdots \times S_{Nh})^{\frac{1}{N+1}} \]

(2.11)

Suppose \( S(t) \) is the stock price at time \( t \). If the average is continuously sampled, Wilmott (2006, p.431) shows the continuously sampled arithmetic average over the period \([0, t]\) is

\[ A_c(t) = \frac{1}{t} \int_0^t S(\tau) \ d\tau \]

(2.12)

and continuously sampled geometric average over the period \([0, t]\) is

\[ G_c(t) = \exp\left(\frac{1}{t} \int_0^t \log S(\tau) \ d\tau\right) \]

(2.13)

The arithmetic average is commonly used in practice. Since the sum of lognormal variables is not lognormally distributed given that the stock price follows a lognormal distribution, options based on the arithmetic average have no closed-form solutions. Although the geometric average is less common in derivatives markets, it has computational convenience. There are closed-form formulae for options based on the geometric average.

When the average is used in place of the stock price, the option is called average price option. When the average is used in place of the strike price, the option is called average strike option. Let \( A(T) \) be the arithmetic average, \( G(T) \) the geometric average, \( S_T \) the stock price at maturity, and \( K \) the strike price.

Here are eight basic types of Asian option
Arithmetic average price call payoff = \((A(T) - K)^+\)
Arithmetic average price put payoff = \((K - A(T))^+\)
Arithmetic average strike call payoff = \((S_T - A(T))^+\)
Arithmetic average strike put payoff = \((A(T) - S_T)^+\)
Geometric average price call payoff = \((G(T) - K)^+\)
Geometric average price put payoff = \((K - G(T))^+\)
Geometric average strike call payoff = \((S_T - G(T))^+\)
Geometric average strike put payoff = \((G(T) - S_T)^+\)

(2.14)

Following Geman and Yor (1993) and others, this chapter mainly focuses on comparing numerical and analytical methods for pricing continuous arithmetic Asian options (average price) of which the average is defined as

\[ A(T) = \frac{1}{T} \int_0^T S(\tau) \, d\tau \]  

(2.15)

where \(S(t)\) is the stock price at time \(t\).

- **2.4. Pricing formulae for geometric Asian options**

Geometric Asian options can be easily priced due to the fact that the product of lognormal variables is still lognormally distributed. The pricing formulae are available for both average price options and average strike options (see McDonald, 2006, pp.466-467).

Assume the interest rate is \(r\) and the underlying stock has a dividend yield \(\delta\) and volatility \(\sigma\). The average is computed using \(N\) equally spaced stock prices from time 0 to \(T\), with the first observation at time \(T/N\). A discrete geometric average price option can then be priced by substituting \(\delta^*\) and \(\sigma^*\) for \(\delta\) and \(\sigma\) in the Black-Scholes formula with

\[ \delta^* = \frac{1}{2} \left[ r \frac{N-1}{N} + \left( \frac{\delta + \frac{1}{2} \sigma^2}{\sqrt{N}} \right) \frac{N+1}{N} - \frac{\sigma^2}{N} \frac{(N+1)(2N+1)}{6} \right] \]  

(2.16)

and

\[ \sigma^* = \frac{\sigma}{\sqrt{N}} \sqrt{\frac{(N+1)(2N+1)}{6}} \]  

(2.17)

where \(\delta^*\) is the dividend yield for the average and \(\sigma^*\) is the volatility of the average.

For a continuous geometric average price option, the formula reduces to, by letting \(N \to \infty\),

\[ \delta^* = \frac{1}{2} \left[ r + \delta + \frac{1}{6} \sigma^2 \right] \]  

(2.18)
and

\[ \sigma^* = \frac{1}{\sqrt{3}} \sigma \]  \hspace{1cm} (2.19)

Let \( S_T \) be the stock price at time \( T \) and \( A(T) \) the discrete arithmetic average. To price the discrete geometric average strike option, it requires to know the volatility \( \sigma^{**} \) of the difference between \( \ln(S_T) \) and \( \ln(A(T)) \), which is given by

\[ \sigma^{**} = \sigma \sqrt{T} \sqrt{1 + \frac{(N+1)(2N+1)}{6N^2} - 2\rho \sqrt{\frac{(N+1)(2N+1)}{6N^2}}} \]  \hspace{1cm} (2.20)

where the correlation coefficient \( \rho \) is

\[ \rho = \frac{1}{2} \sqrt{\frac{6(N+1)}{2N+1}} \]  \hspace{1cm} (2.21)

A discrete geometric average strike option can then be valued using the Black-Scholes formula by replacing \( K \), \( r \), and \( \sigma \) with \( S_0 \), \( \delta^* \), and \( \sigma^{**} \) where \( \delta^* \) is given by (2.16). The dividend yield \( \delta \) on the underlying stock in the Black-Scholes formula remains the same.

- **2.5. Black-Scholes equation**

The Black-Scholes partial differential equation (PDE), or Black-Scholes equation, can be used to describe the behavior of the option price for virtually all derivatives. It is important because the correct option pricing formula must satisfy the Black-Scholes equation (assuming the validity of the Black-Scholes framework).

Let \( V(S(t), t) \) be the value of an option depending on the stock price \( S(t) \) and the time \( t \). The Black-Scholes equation is (e.g. McDonald, 2006, p.682)

\[ \frac{\partial V}{\partial t} + \frac{1}{2} \sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + (r - \delta) S \frac{\partial V}{\partial S} - r V = 0 \]  \hspace{1cm} (2.22)

where \( r \) is the risk-free interest rate, \( \delta \) the dividend yield, and \( \sigma \) the volatility. The Black-Scholes equation describes the change in value of almost any option including European options. The equation (2.22) above must be solved with appropriate boundary conditions depending on the payoff of the option. Thus, the solution obtained is the correct pricing formula for the option.

The Black-Scholes equation can be extended to characterizing options contingent on the average such as
Asian options. Let $V(S(t), I(t), t)$ be the value of an option contingent on the average. A new state variable $I(t)$ is introduced to describe the average.

For an option contingent on a continuous arithmetic average defined as

$$\frac{1}{t} \int_0^t S(\tau) \, d\tau$$

we define

$$I(t) = \int_0^t S(\tau) \, d\tau$$

The partial differential equation for this option is (e.g. Wilmott, 2006, p.431)

$$\frac{\partial V}{\partial t} + S \frac{\partial V}{\partial I} + \frac{1}{2} \sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + (r - \delta) S \frac{\partial V}{\partial S} - r V = 0$$

If the option is, for example, a continuous arithmetic Asian call, the boundary condition for the PDE above is

$$V(S(T), I(T), T) = \left( \frac{1}{T} I(T) - K \right)^+$$

where $K$ is the strike price of the Asian option.

To value an option contingent on a continuous geometric average defined as

$$\exp\left( \frac{1}{t} \int_0^t \log S(\tau) \, d\tau \right)$$

we define

$$I(t) = \int_0^t \log S(\tau) \, d\tau$$

and the corresponding partial differential equation is (e.g. Wilmott, 2006, p.432)

$$\frac{\partial V}{\partial t} + \log S \frac{\partial V}{\partial I} + \frac{1}{2} \sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + (r - \delta) S \frac{\partial V}{\partial S} - r V = 0$$
3. Numerical and Analytical Methods

- 3.1. Preliminaries
  - 3.1.1. Cumulative distribution function

Let $X$ be a real-valued random variable. The cumulative distribution function (CDF) of $X$ is defined by

$$F(x) = P(X \leq x)$$  \hspace{1cm} (3.1)

The pdf of $X$ is the derivative of the CDF when it exists

$$f(x) = F'(x)$$  \hspace{1cm} (3.2)

For a continuous random variable $X$, the CDF can be defined in terms of its pdf

$$F(x) = \int_{-\infty}^{x} f(t) \, dt$$  \hspace{1cm} (3.3)

The complementary CDF or simply the tail distribution of $X$ is defined by

$$F^c(x) = 1 - F(x)$$  \hspace{1cm} (3.4)

- 3.1.2. Characteristic function

Let $X$ be a real-valued random variable. The characteristic function of $X$ is the complex-valued function of a real variable

$$\varphi_X(t) = E[e^{itX}] = \int_{-\infty}^{\infty} e^{itx} \, dF_X(x) = \int_{-\infty}^{\infty} e^{itx} \, f_X(x) \, dx$$  \hspace{1cm} (3.5)

where $i$ is the imaginary unit defined such that $i^2 = -1$.

Using the Euler's formula $e^{ix} = \cos x + i \sin x$, we rewrite the characteristic function as

$$\varphi_X(t) = \int_{-\infty}^{\infty} \cos tx + i \sin tx \, dF_X(x)$$
$$= \int_{-\infty}^{\infty} \cos tx \, dF_X(x) + i \int_{-\infty}^{\infty} \sin tx \, dF_X(x)$$  \hspace{1cm} (3.6)

The conjugate of $\varphi_X(t)$

$$\overline{\varphi}_X(t) = \int_{-\infty}^{\infty} \cos tx \, dF_X(x) - i \int_{-\infty}^{\infty} \sin tx \, dF_X(x)$$
$$= \int_{-\infty}^{\infty} \cos (tx - i \sin tx) \, dF_X(x)$$
$$= \int_{-\infty}^{\infty} e^{-itx} \, dF_X(x)$$
$$= E[e^{-itX}]$$
$$= \varphi_X(-t)$$  \hspace{1cm} (3.7)

Recall that $\text{Re}(z) = (z + \bar{z}) / 2$ and $\text{Im}(z) = (z - \bar{z}) / 2i$ where $z$ is a complex number. Thus, the real part and the imaginary part of $\varphi_X(t)$ are represented by
\[
\text{Re}(\varphi_X(t)) = \frac{\varphi_X(t) + \varphi_X(-t)}{2} = \int_{-\infty}^{\infty} \cos tx \, dF_X(x) \\
\text{Im}(\varphi_X(t)) = \frac{\varphi_X(t) - \varphi_X(-t)}{2i} = \int_{-\infty}^{\infty} \sin tx \, dF_X(x)
\]

Since \(\cos tx\) is even function, \(\text{Re}(\varphi_X(t))\) is thus even function. Since \(\sin tx\) is odd function, \(\text{Im}(\varphi_X(t))\) is thus odd function.

Given the characteristic function \(\varphi_X(t)\), the inversion formula allows us to find the corresponding \(f_X(x)\).

If \(\varphi_X(t)\) is integrable, then we have
\[
f_X(x) = F'_X(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{-ix} \varphi_X(t) \, dt
\]
(3.9)

Since \(f_X(x)\) is real, we can write (3.9) as the integral of a real-valued function of a real variable.

\[
f_X(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \text{Re}[e^{-ix} \varphi_X(t)] \, dt
\]
\[
= \frac{1}{2\pi} \int_{-\infty}^{\infty} \text{Re}[\cos tx - i \sin tx] \{\text{Re}(\varphi_X(t)) + i \text{Im}(\varphi_X(t))\} \, dt
\]
\[
= \frac{1}{2\pi} \int_{-\infty}^{\infty} \text{Re}[\cos tx \varphi_X(t) + \sin tx \text{Im}(\varphi_X(t))] \, dt
\]
\[
= \frac{1}{2\pi} \left[ \int_{-\infty}^{0} \text{Re}[\cos tx \varphi_X(t) + \sin tx \text{Im}(\varphi_X(t))] \, dt + \int_{0}^{\infty} \text{Re}[\cos tx \varphi_X(t) + \sin tx \text{Im}(\varphi_X(t))] \, dt \right]
\]

Since \(\cos tx\) and \(\text{Re}(\varphi_X(t))\) are both even, while \(\sin tx\) and \(\text{Im}(\varphi_X(t))\) are both odd, we have

\[
\int_{-\infty}^{0} \text{Re}[\cos tx \varphi_X(t) + \sin tx \text{Im}(\varphi_X(t))] \, dt = \int_{0}^{\infty} \text{Re}[\cos tx \varphi_X(t) + \sin tx \text{Im}(\varphi_X(t))] \, dt
\]
(3.10)

Hence

\[
f_X(x) = \frac{1}{\pi} \int_{0}^{\infty} \text{Re}[\cos tx \varphi_X(t) + \sin tx \text{Im}(\varphi_X(t))] \, dt
\]
(3.11)

### 3.1.3. Fourier transform

Let \(f(t)\) be a real-valued function of a real variable \(t\). If \(\int_{-\infty}^{\infty} |f(t)| \, dt\) is finite, then the Fourier transform of \(f(t)\) is defined by

\[
\tilde{f}(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(t) e^{-i\omega t} \, dt
\]
(3.12)

The inverse Fourier transform is given by the Fourier's inversion theorem.

\[
f(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \tilde{f}(\omega) e^{i\omega t} \, d\omega
\]
(3.13)

The constant \(1/\sqrt{2\pi}\) in the definition of the Fourier transform is arbitrary as long as that the product of the constants in the Fourier transform and the inverse Fourier transform is equal to \(1/(2\pi)\).
If \( f(t) \) is odd, i.e. \( f(-t) = -f(t) \), then the Fourier transform takes another form.

\[
\tilde{f}(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(t) e^{-i\omega t} dt
\]

\[
= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(t) (\cos \omega t - i \sin \omega t) dt
\]

\[
= \frac{1}{\sqrt{2\pi}} \left[ \int_{-\infty}^{0} f(t) (\cos \omega t - i \sin \omega t) dt + \int_{0}^{\infty} f(t) (\cos \omega t - i \sin \omega t) dt \right]
\]

\[
= \frac{1}{\sqrt{2\pi}} \left[ \int_{0}^{\infty} f(t) (\cos \omega t - i \sin \omega t) dt + \int_{0}^{\infty} f(t) (\cos \omega t - i \sin \omega t) dt \right]
\]

\[
= \frac{-2i}{\sqrt{2\pi}} \int_{0}^{\infty} f(t) \sin \omega t dt
\]

It is clear that \( \tilde{f}(\omega) \) is an odd function of \( \omega \).

If \( f(t) \) is even, i.e. \( f(-t) = f(t) \), then the Fourier transform takes the following form.

\[
\tilde{f}(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(t) e^{-i\omega t} dt
\]

\[
= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(t) (\cos \omega t - i \sin \omega t) dt
\]

\[
= \frac{1}{\sqrt{2\pi}} \left[ \int_{-\infty}^{0} f(t) (\cos \omega t - i \sin \omega t) dt + \int_{0}^{\infty} f(t) (\cos \omega t - i \sin \omega t) dt \right]
\]

\[
= \frac{1}{\sqrt{2\pi}} \left[ \int_{-\infty}^{\infty} f(t) (\cos \omega t - i \sin \omega t) dt \right]
\]

\[
= \frac{1}{\sqrt{2\pi}} \int_{0}^{\infty} f(t) \cos \omega t dt
\]

Thus, \( \tilde{f}(\omega) \) is an even function of \( \omega \).

### 3.1.4. Laplace transform

Let \( f(t) \) be a real-valued function of a real variable \( t > 0 \). The Laplace transform of \( f(t) \) is defined as

\[
\hat{f}(s) = \mathcal{L}[f(t)] = \int_{0}^{\infty} e^{-st} f(t) dt
\]

(3.17)

where the symbol \( \mathcal{L} \) denotes the Laplace transform. The transform variable \( s \) is complex with its real part \( \text{Re}(s) \geq 0 \).

The inverse Laplace transform is known as the Bromwich integral defined by

\[
f(t) = \mathcal{L}^{-1} \left[ \hat{f}(s) \right] = \frac{1}{2\pi i} \int_{a-i\infty}^{a+i\infty} e^{st} \hat{f}(s) ds
\]

where the symbol \( \mathcal{L}^{-1} \) denotes the inverse Laplace transform. The integration is done along a contour which is any vertical line \( s = a \) such that all singularities of \( \hat{f}(s) \) are to the left of it.

Since
\[ \mathcal{L}[e^{-at} f(t)] = \int_0^\infty f(t) e^{-st} \, dt \]
\[ = \int_0^\infty f(t) e^{-t(s+a)} \, dt \]
\[ = \hat{f}(s+a) \]

(3.19)

the value of \( f(t) \) can be obtained by implementing inverse Laplace transform on \( \hat{f}(s+a) \)

\[ \mathcal{L}^{-1}[\hat{f}(s+a)] = e^{-at} f(t) \]
\[ f(t) = e^{at} \mathcal{L}^{-1}[\hat{f}(s+a)] \]

(3.20)

We can invert \( \hat{f}(s+a) \) instead of \( \hat{f}(s) \) to ensure that all the singularities are to the left of the contour \( \text{Re}(s+a) \). In particular, Laplace transform \( \hat{C}(\lambda, q) \) has two simple poles at \( \lambda = 0 \) and \( \lambda = 2 + 2 \nu \). We add real \( a \) to \( \lambda \) where \( a > \max(0, 2 + 2 \nu) \) such that \( \hat{C}(\lambda + a, q) \) is regular in the right half plane. Then, we invert \( \hat{C}(\lambda + a, q) \) and obtain \( C^{\nu}(h, q) \) by using

\[ C^{\nu}(h, q) = e^{ah} \mathcal{L}^{-1}[\hat{C}(\lambda + a, q)] \]

(3.21)

Furthermore, Making the change of \( s = a + i u \) in the Bromwich integral (3.0), we obtain

\[ f(t) = \frac{1}{2\pi i} \int_{a-i\infty}^{a+i\infty} e^{(a+i u)t} \hat{f}(a+i u) \, du \]
\[ = \frac{e^{at}}{2\pi} \int_{-\infty}^{\infty} e^{iut} \hat{f}(a+i u) \, du \]

(3.22)

Introducing \( g(t) = e^{-at} f(t) \) and \( \hat{g}(u) = \hat{f}(a+i u) \), we have

\[ f(t) e^{-at} = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{iut} \hat{f}(a+i u) \, du \]
\[ g(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{iut} \hat{g}(u) \, du \]

(3.23)

The above integral is exactly the inverse Fourier transform of \( \hat{g}(u) \). Hence, the Laplace transform is basically a variant of Fourier transform. If we use \( \mathcal{F}^{-1} \) to denote the inverse Fourier transform, we have

\[ f(t) = e^{at} \, g(t) \]
\[ = e^{at} \mathcal{F}^{-1}[\hat{g}(u)] \]
\[ = e^{at} \mathcal{F}^{-1}[\hat{f}(a+i u)] \]

(3.24)

If \( f(t) \) is defined on nonnegative real line, in other words, \( f(t) = 0 \) for \( t < 0 \), the Laplace transform is linked with the characteristic function in the following way

\[ \hat{f}(s) = \int_0^\infty e^{-st} f(t) \, dt \]
\[ = \int_0^\infty e^{i(s+i)s} f(t) \, dt \]
\[ = \varphi(is) \]

(3.25)
3.1.5. The trapezoidal rule

The trapezoidal rule approximates the integral of a function \( g(u) \) over an interval \([a, b]\) of the real line by a series.

\[
\int_a^b g(u) \, du \approx h \left[ \frac{g(a)+g(b)}{2} + \sum_{k=1}^{n-1} g(a + k h) \right]
\]  
(3.26)

where \( h = (b-a)/n \). The above formula also applies when \( a = -\infty \) or \( b = \infty \) with modifications.

3.2. German and Yor’s Laplace transform

3.2.1. The mathematical setting

The problem of Asian option pricing involves the integral of an exponential of Brownian motion. German and Yor (1993) used Bessel processes to derive the Laplace transform of the price of an Asian option which is out-of-the-money. They also obtained an analytical expression of the Asian option price when the option is in-the-money. An out-of-the-money option would have a negative payoff if exercised immediately. By contrast, an in-the-money option would have a positive payoff. German and Yor’s work makes it possible to price an Asian option by numerically inverting the Laplace transform.

Given a probability space \((\Omega, \mathcal{F}, P)\) with a filtration \((\mathcal{F}_t)_{0\leq t<\infty}\), under the risk-neutral probability measure \(Q\) (equivalent to \(P\)), the underlying stock price is driven by

\[
dS(t) = rS(t) \, dt + \sigma S(t) \, dW_t
\]  
(3.27)

where \(r\) is the constant instantaneous risk-free interest rate. \(\sigma\) is the constant volatility. \(W_t\) is a standard one-dimensional Brownian motion. The above price dynamics is called the stochastic differential equation (SDE). The solution to it is given by

\[
S_t = S_0 \exp\{(r - \frac{1}{2} \sigma^2) t + \sigma W_t\}, \quad t \in [0, \infty)
\]  
(3.28)

Let the average value of the stock is calculated over the time interval \([t_0, T]\), where \(T\) is the maturity of the option. Define an average process \((A(x), x > 0)\)

\[
A(x) = \frac{1}{x-t_0} \int_{t_0}^T S(u) \, du
\]  
(3.29)

Then, the payoff of an Asian call option at maturity \(T\) is given by

\[
\text{max}[(A(T) - k), 0] = (A(T) - k)^+
\]

where \(k\) is the strike price of the option. The value of the Asian call at time \(t\) is
where

\[ C_{i,T}(k) = e^{-r(T-t)} E_Q[A(T) - k]^+ | \mathcal{F}_t] \]

German and Yor proceeded with three steps.

**Step 1.**

\[ C_{i,T}(k) = e^{-r(T-t)} E_Q[A(T) - k]^+ | \mathcal{F}_t] = \frac{e^{-r(T-t)}}{T-t_0} \tilde{C}_{i,T}(k') \]

where

\[
\tilde{C}_{i,T}(k') = E_Q[(\tilde{A}(T) - k')^+ | \mathcal{F}_t]
\]

\[
\tilde{A}(T) = \int_{t_0}^T S(u) \, du
\]

\[ k' = k(T-t_0) \]

**Step 2.** Reduce the computation of the conditional expectation to that of the deterministic function

Define

\[ \xi^{a,b}(s, \gamma) \equiv E[\left( \int_0^s \exp(a W(u) + b u) \, du - \gamma \right)^+] \]

where \( b \in \mathbb{R}, a, s, \gamma \in \mathbb{R} \). The case \( t_0 \leq t < T \) is considered here while the case \( t_0 > t \) is called the forward-start Asian option. The average value \( \tilde{A}(T) \) at maturity \( T \) can be decomposed into the sum of the realised average \( \tilde{A}(t) \) and the unrealized average.

\[ \tilde{A}(T) = \tilde{A}(t) + S(t) \int_0^{T-t} \exp(\tilde{\gamma}(s)) \, ds \]

with

\[ \tilde{\gamma}(s) \equiv \sigma \tilde{W}(s) + \left( r - \frac{1}{2} \sigma^2 \right) s \]

where \( \tilde{W}(s) = W(s + t) - W(t), s \geq 0 \) is another Brownian motion independent of \( \mathcal{F}_t \). Hence

\[
E_Q[(\tilde{A}(T) - k')^+ | \mathcal{F}_t] = E_Q[S(t) \int_0^{T-t} \exp(\tilde{\gamma}(s)) \, ds - (k' - \tilde{A}(t))^+] | \mathcal{F}_t]
\]

\[ = S(t) E_Q[(\int_0^{T-t} \exp(\tilde{\gamma}(s)) \, ds - k'')^+] | \mathcal{F}_t] \]

where \( k'' = (k' - \tilde{A}(t))/S(t) \). As a result,

\[ \tilde{C}_{i,T}(k') = S(t) \xi^{a,b}(T-t, k'') \]

where \( b = r - \frac{1}{2} \sigma^2 \)
Step 3. Use the scaling property of Brownian motion to replace \( \sigma \) by 2 in the expression of \( \zeta^{\sigma,b}(s, \gamma) \)

Making the change of variable \( u = \frac{4y}{\sigma^2} \) and recalling the scaling property of Brownian motion

\[
\exp(a W(t) + \nu t) = \exp(\hat{W}(t a^2) + \nu t),
\]

we have

\[
\begin{align*}
\zeta^{\sigma,b}(s, \gamma) &= \mathbb{E}\left[ \int_0^s \exp\left( \sigma W(u) + b u \right) du - \gamma \right]^+ \\
&= \mathbb{E}\left[ \int_0^s \exp\left( \sigma W\left( \frac{4y}{\sigma^2} \right) + \frac{4b}{\sigma^2} u \right) du - \gamma \right]^+ \\
&= \mathbb{E}\left[ \frac{4}{\sigma^2} \int_0^{\frac{s}{4}} \exp\left( \frac{\sigma^2}{2} \right) W\left( \frac{4y}{\sigma^2} \right) + \frac{4b}{\sigma^2} u \right] du - \frac{\gamma \sigma^2}{4} \right]^+ \\
&= \frac{4}{\sigma^2} \mathbb{E}\left[ \int_0^{\frac{s}{4}} \exp\left( \frac{\sigma^2}{2} \right) W\left( \frac{4y}{\sigma^2} \right) + \frac{4b}{\sigma^2} u \right] du - \frac{\gamma \sigma^2}{4} \\
&= \frac{4}{\sigma^2} 2^{2b/\sigma^2} \left( \frac{s \sigma^2}{4}, \frac{\gamma \sigma^2}{4} \right)
\end{align*}
\]

Define

\[
C^{(s)}(x, q) \equiv \mathbb{E}\left[ \int_0^s \exp\left( 2 \left( W(u) + \nu u \right) \right) du - q \right]^+
\]

\[
= \mathbb{E}\left[ \int_0^s \exp\left( 2 \left( W(u) + 2 \nu u \right) \right) du - q \right]^+
\]

\[
= \zeta^{2,2\nu}(x, q)
\]

Then

\[
\zeta^{\sigma,b}(s, \gamma) = \frac{4}{\sigma^2} 2^{2b/\sigma^2} \left( \frac{s \sigma^2}{4}, \frac{\gamma \sigma^2}{4} \right)
\]

\[
= \frac{4}{\sigma^2} C^{(2,2\nu)}(x, q)
\]

In summary

The Asian option price is expressed as

\[
C_{i,t}(k) = \frac{e^{c(t-\sigma)/\sigma^2}}{T-t_0} \mathbb{E}_{i,T}(k')
\]

\[
= \frac{e^{c(T-t)/\sigma^2}}{T-t_0} \mathbb{E}(T) \zeta^{\sigma,b}(T-t, k''')
\]

\[
= \frac{e^{c(T-t)/\sigma^2}}{T-t_0} \mathbb{E}(T) \frac{4}{\sigma^2} C^{(2, b/\sigma^2)}\left( \frac{(T-t) \sigma^2}{4}, \frac{k'''}{4} \right)
\]

\[
= \frac{e^{c(T-t)/\sigma^2}}{T-t_0} \left( \frac{4 \mathbb{E}(T)}{\sigma^2} \right) C^{(s)}(q, k)
\]

where
\[ v = \frac{2r}{\sigma^2} - 1 \]
\[ h = \frac{\sigma^2}{4} (T - t) \]
\[ q = \frac{\sigma^2}{4S_0} \left[ k (T - t_0) - \int_{t_0}^{t} ds S(u) \right] \]

recalling that \( b = r - \frac{1}{2} \sigma^2, \ k^n = (k' - \bar{A}(t))/S(t), \ k' = k(T - t_0) \) and \( \bar{A}(t) = \int_{t_0}^{t} ds S(u) \)

Geman and Yor observed that \( q \) may be negative if the realised average \( \bar{A}(t) \) over the time interval \([t_0, t] \) is large enough, which means the Asian option is already known at time \( t \) to be certainly in-the-money at maturity \( T \) for the call and worthless for the put.

- **3.2.2. Laplace transform of \( C^{(v)}(h, q) \)**

Set
\[
A^{(v)}_h = \int_{0}^{h} \exp[2 (W_s + v s)] \, ds
\]

The first two moments of \( A^{(v)}_h \) can be computed easily by integration. In particular, the first-order moment is given by
\[
E[A^{(v)}_h] = \frac{1}{2(v+1)} \left[ \exp(2 (v + 1) h) - 1 \right]
\]

Define
\[
C^{(v)}(h, q) = E\left[(A^{(v)}_h - q)^+ \right]
\]

where
\[
A^{(v)}_h = \int_{0}^{h} \exp[2 (W_s + v s)] \, ds
\]

**When \( q \leq 0 \).** Using the first-order moment of \( A^{(v)}_h \), the calculation can be reduced to
\[
C^{(v)}(h, q) = E(A^{(v)}_h) - q
= \left\{ \frac{1}{2(v+1)} \left[ \exp(2 (v + 1) h) - 1 \right] \right\} - q
\]

Substituting this quantity into (3.38), we obtain a closed-form formula of the Asian option price
\[
C_{AT}(k) = \frac{e^{\gamma(T-t)}}{T-t_0} \left( \frac{4S_0}{\sigma^2} \right) \left\{ \frac{1}{2(v+1)} \left[ \exp(2 (v + 1) h) - 1 \right] \right\} - q
\]

It is interesting that (3.45) has some resemblance to the Black and Scholes (1973) formula. The volatility
\(\sigma\) does not appear explicitly in the formula, but it is embedded implicitly in \(S(t)\) and in the integral \(\int_0^s S(u) \, ds\).

**When \(q > 0\).** There is no simple reduction of \(C^{(\nu)}(h, q)\) for \(q > 0\), Geman and Yor, however, have derived its Laplace transform with respect to the variable \(h\) by exploiting the properties of the Bessel process. We skip the involved derivation and show you their final result. The Laplace transform of \(C^{(\nu)}(h, q)\) with respect to \(h\) is given by

\[
\hat{C}(\lambda, q) = \int_0^\infty e^{-\lambda h} C^{(\nu)}(h, q) \, dh = \frac{\Gamma(2 \nu - 1)}{\Gamma(\nu - 1) \Gamma(\nu + 1)} \left( \frac{h}{\lambda^2 - 2 \nu} \right)^{\nu - 1} \Gamma(\nu) \, dh
\]

where \(\mu = \sqrt{2 \lambda + \nu^2}\) and \(\Gamma\) denotes the gamma function. The Laplace transform \(\hat{C}(\lambda, q)\) has two simple poles at \(\lambda = 0\) and \(\lambda = 2 + 2 \nu\). We add real \(\alpha\) to \(\lambda\) where \(\alpha > \max(0, 2 + 2 \nu)\) such that \(\hat{C}(\lambda + \alpha, q)\) is regular in the right half plane. Then we can obtain \(C^{(\nu)}(h, q)\) by inverse Laplace transform of \(\hat{C}(\lambda + \alpha, q)\)

\[
C^{(\nu)}(h, q) = e^{\alpha h} \mathcal{L}^{-1}\left[\hat{C}(\lambda + \alpha, q)\right]
\]

Using the symbolic integration of *Mathematica*, the Laplace transform of \(C^{(\nu)}(h, q)\) can take another form in terms of the regularized confluent hypergeometric function \(_1\Phi_1(a; b; z)\) and gamma function

\[
\hat{C}(\lambda, q) = \frac{(2 q)^{1 - \nu(\nu - 1)}}{\Gamma(\nu - 1) \Gamma(\nu + 1)} \left( \frac{h}{\lambda^2 - 2 \nu} \right)^{-\nu - 1} \, dh
\]

with

\[
_1\Phi_1(a; b; z) = \frac{\Gamma(b)}{\Gamma(b)}
\]

where \(_1\Phi_1(a; b; z)\) is the Kummer confluent hypergeometric function as discussed below.

Calculating the Laplace transform of \(\hat{C}(\lambda, q)\) one more time with respect to \(q\) in Wolfram Mathematica yields the double Laplace transform of \(C^{(\nu)}(h, q)\) in terms of Bessel function, hypergeometric function and gamma function.
\[
\frac{1}{\lambda(\lambda-2(1+y))} 2^{-1+y} \sqrt{2\lambda+y^2} + \frac{1}{2} \left\{4 + y + \sqrt{2\lambda+y^2}\right\} \Gamma \left(\frac{3}{2}, \left[4 + y + \sqrt{2\lambda+y^2}\right]\right)
\]
\[
\left\{\begin{align*}
2\pi &Bessel\left(\sqrt{2\lambda+y^2}, \sqrt{\nu}ight) C\left[\frac{\nu}{2}\right] \left[\frac{y}{2}\right] \left[\sqrt{2\lambda+y^2}\right] \\
\Gamma &\left(\frac{3}{2}, \left[2\lambda+y^2\right]\right) + \frac{1}{2} \left\{4 + y + \sqrt{2\lambda+y^2}\right\} \Gamma \left(\frac{3}{2}, \left[4 + y + \sqrt{2\lambda+y^2}\right]\right)
\end{align*}\right.
\]
(3.49)

Two-dimensional Laplace transform algorithms are able to invert the double Laplace transform.

- **3.2.3. The put-call parity**

Linetsky (2004) gives the put-call parity relationship for newly written Asian options, i.e., \( t = t_0 = 0 \) (also see Geman and Yor 1993, Dufresne 2000)

\[
C_0 = P_0 + \begin{cases} 
\frac{e^{rT} - e^{-rT}}{r-\delta} S_0 - e^{-rT} K, & r \neq \delta \\
e^{-rT} (S_0 - K), & r = \delta 
\end{cases}
\]
(3.50)

(3.50) enables us to price Asian call options by evaluating Asian put options, and vice versa. However, (3.50) solely holds when Asian options are newly written. If we are interested in pricing them at time \( 0 < t < T \) other than zero (so-called seasoned Asian option), Equation (3.50) can not help. We have deduced the put-call parity relationship for Asian options at any time \( t \) in \([t_0, T]\) as follows

\[
C_t = P_t + \begin{cases} 
\frac{e^{r(T-t)} - e^{-r(T-t)}}{(r-\delta)(T-t)} S_t - e^{-r(T-t)} \left(K - \frac{1}{T-t} \int_{t_0}^{t} S_u \, du\right), & r \neq \delta \\
e^{-r(T-t)} \left(\frac{T-t}{T-t_0} S_t - K + \frac{1}{T-t_0} \int_{t_0}^{t} S_u \, du\right), & r = \delta 
\end{cases}
\]
(3.51)

- **3.3 Numerical inversion algorithms**
- **3.3.1. Euler method**

Abate and Whitt (1995) presents a simple algorithm called Euler for numerically inverting Laplace transforms. The Euler method is effective, but it lacks explicit error bound. Hence, Abate and Whitt (1995) proposed using two different methods to cross-check the accuracy in a way that two methods agree to within desired precision. They used the Post-Widder method to confirm the accuracy. Although the Euler method and the Post-Widder method are both variants of the Fourier series method, they are very different.

The Euler method is based on the Bromwich integral (3.18). Making the change of variable \( s = a + iu \), we obtain
\[ f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{it} \hat{f}(s) \, ds \]
\[ = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{it} \hat{f}(a + i u) \, d(a + i u) \]
\[ = \frac{e^{iu}}{2\pi} \int_{-\infty}^{\infty} e^{iut} \hat{f}(a + i u) \, du \]
\[ = \frac{e^{iu}}{2\pi} \int_{-\infty}^{\infty} (\cos ut + i \sin ut) \hat{f}(a + i u) \, du \]
\[ = \frac{e^{iu}}{2\pi} \int_{-\infty}^{\infty} \left[ \text{Re} \left( \hat{f}(a + i u) \right) \cos ut - \text{Im} \left( \hat{f}(a + i u) \right) \sin ut \right] \, du + \int_{0}^{\infty} \left[ \text{Re} \left( \hat{f}(a + i u) \right) \cos ut - \text{Im} \left( \hat{f}(a + i u) \right) \sin ut \right] \, du \]
\[ = \frac{e^{iu}}{2\pi} \left( \text{Re} \left( \hat{f}(a - i u) \right) \cos ut + \text{Im} \left( \hat{f}(a - i u) \right) \sin ut \right) \, du + \int_{0}^{\infty} \left[ \text{Re} \left( \hat{f}(a + i u) \right) \cos ut - \text{Im} \left( \hat{f}(a + i u) \right) \sin ut \right] \, du \]
\[ = \frac{e^{iu}}{2\pi} \int_{0}^{\infty} \left[ \text{Re} \left( \hat{f}(a - i u) \right) \cos ut + \text{Re} \left( \hat{f}(a + i u) \right) \cos ut + \right. \]
\[ \left. \text{Im} \left( \hat{f}(a - i u) \right) \sin ut + \text{Im} \left( \hat{f}(a + i u) \right) \sin ut \right] \, du \]

Since
\[ \text{Re} \left( \hat{f}(a - i u) \right) = \text{Re} \left( \int_{0}^{\infty} f(t) e^{-(a-iu)t} \, dt \right) \]
\[ = \int_{0}^{\infty} f(t) e^{-a t} \text{Re} \left( e^{iu t} \right) \, dt \]
\[ = \int_{0}^{\infty} f(t) e^{-a t} \cos ut \, dt \]
\[ \text{Re} \left( \hat{f}(a + i u) \right) = \text{Re} \left( \int_{0}^{\infty} f(t) e^{(a+iu)t} \, dt \right) \]
\[ = \int_{0}^{\infty} f(t) e^{-a t} \text{Re} \left( e^{-iu t} \right) \, dt \]
\[ = \int_{0}^{\infty} f(t) e^{-a t} \cos ut \, dt \]
\[ \text{Im} \left( \hat{f}(a - i u) \right) = \text{Im} \left( \int_{0}^{\infty} f(t) e^{-(a-iu)t} \, dt \right) \]
\[ = \int_{0}^{\infty} f(t) e^{-a t} \text{Im} \left( e^{iu t} \right) \, dt \]
\[ = \int_{0}^{\infty} f(t) e^{-a t} \sin ut \, dt \]
\[ \text{Im} \left( \hat{f}(a + i u) \right) = \text{Im} \left( \int_{0}^{\infty} f(t) e^{(a+iu)t} \, dt \right) \]
\[ = \int_{0}^{\infty} f(t) e^{-a t} \text{Im} \left( e^{-iu t} \right) \, dt \]
\[ = -\int_{0}^{\infty} f(t) e^{-a t} \sin ut \, dt \]

we have
\[ \text{Re} \left( \hat{f}(a - i u) \right) = \text{Re} \left( \hat{f}(a + i u) \right) = \int_{0}^{\infty} f(t) e^{-a t} \cos ut \, dt \]
\[ \text{Im} \left( \hat{f}(a - i u) \right) = -\text{Im} \left( \hat{f}(a + i u) \right) = \int_{0}^{\infty} f(t) e^{-a t} \sin ut \, dt \]

Thus
\[ f(t) = \frac{e^{iu}}{2\pi} \int_{0}^{\infty} \left[ 2 \text{Re} \left( \hat{f}(a + i u) \right) \cos ut - 2 \text{Im} \left( \hat{f}(a + i u) \right) \sin ut \right] \, du \]
\[ = \frac{e^{iu}}{\pi} \int_{0}^{\infty} \left[ \text{Re} \left( \hat{f}(a + i u) \right) \cos ut - \text{Im} \left( \hat{f}(a + i u) \right) \sin ut \right] \, du \]
Abate and Whitt (1995) showed

$$ f(t) = \frac{2e^{\alpha t}}{\pi} \int_0^\infty \text{Re} \left( \hat{f}(a + i u) \right) \cos u t \, du $$

(3.59)

The integral is replaced by an alternating series using the trapezoidal rule (3.26) with the step size

$$ h = \pi / 2 t $$

$$ f_h(t) = \frac{2e^{\alpha t}}{\pi} h \left[ \frac{\text{Re} \left( \hat{f}(a) \right)}{2} + \sum_{k=1}^\infty \text{Re} \left( \hat{f}(a + i k h) \right) \cos k h t \right] $$

$$ = \frac{h e^{\alpha t}}{\pi} \text{Re} \left( \hat{f}(a) \right) + \frac{2k e^{\alpha t}}{\pi} \sum_{k=1}^\infty \text{Re} \left( \hat{f}(a + i k h) \right) \cos k h t $$

$$ = \frac{e^{\alpha t}}{2t} \text{Re} \left( \hat{f}(a) \right) + \frac{e^{\alpha t}}{t} \sum_{k=1}^\infty (-1)^k \text{Re} \left( \hat{f}(a + \frac{2k\pi i}{2t}) \right) $$

(3.60)

Letting $a = A / 2 t$, we obtain

$$ f_h(t) = \frac{e^{\alpha t}}{2t} \text{Re} \left( \hat{f} \left( \frac{A}{2t} \right) \right) + \frac{e^{\alpha t}}{t} \sum_{k=1}^\infty (-1)^k \text{Re} \left( \hat{f} \left( \frac{A+2k\pi i}{2t} \right) \right) $$

(3.61)

Thus, $f(t)$ is approximated by $f_h(t)$ with the discretization error defined by $e_d = |f(t) - f_h(t)|$. Using the Poisson summation formula, the discretization error $e_d$ associated with the trapezoidal rule in (3.61) is given by

$$ e_d = \sum_{k=1}^\infty e^{-k A} f((2k + 1) t) $$

(3.62)

In probability applications, e.g. $|f(t)| \leq 1$ for all $t$, the discretization error is bounded by

$$ |e_d| \leq \frac{e^{-A}}{1 - e^{-A}} $$

(3.63)

To achieve a discretization error of at most $10^{-7}$, we require

$$ 10^{-7} = \frac{e^{-A}}{1 - e^{-A}} $$

$$ e^{-A} = \frac{10^{-7}}{1 + 10^{-7}} $$

(3.64)

We choose $\gamma$ large, which is equivalent to choosing $A$ large, in order to make the discretization error $e_d$ small. As Abate and Whitt (1992) noted, increasing $\gamma$ can make the computation (3.61) more difficult, e.g. by setting decimals to have higher precision to reduce round-off error. They remarked that at least $1.5 \gamma$-digit precision was required to achieve an error of $10^{-7}$.

Abate and Whitt used Euler summation as the acceleration technique to calculate (3.61). By truncating the infinite series, we obtain
\[ s_n(t) = \frac{e^{it}}{2} \text{Re} \left( \hat{f} \left( \frac{n+2k+1}{2} \right) \right) + \frac{e^{it}}{2} \sum_{k=1}^{\infty} (-1)^k a_k(t) \]  

(3.65)

where

\[ a_k(t) = \text{Re} \left( \hat{f} \left( \frac{n+2k+1}{2} \right) \right) \]

Using Euler summation, we approximate \( f(t) \) by

\[ f(t) \approx E(m, n, t) = \sum_{k=0}^{m} \binom{m}{k} 2^{-m} s_{n+k}(t) \]  

(3.66)

where \( \binom{m}{k} \) is the binomial coefficient defined as \( \frac{m!}{k!(m-k)!} \). Abate and Whitt (1995) suggested using the difference of successive terms, i.e. \( E(m, n+1, t) - E(m, n, t) \), to estimate the error associated with Euler summation, and remarked that ‘this usually is a good error estimate, but not always so’.

Fu, Madan and Wang (1999) used the setting of \( m = 11, n = 15 \) and \( A = 18.4 \) (\( A = 18.4 \) is equivalent to \( \gamma = 8 \)) recommended in Abate and Whitt (1995) to calculate Asian option prices and remarked that with these values, the Euler method in general provides accuracy of at least three significant figures.

### 3.3.2. Post-Widder method

The Post-Widder method proposed by Abate and Whitt (1995) to serve as cross-checks with the Euler method is based on the Post-Widder theorem. Feller (1971, p.233) showed that \( f(t) \) is the pointwise limit of \( f_n(t) \) as \( n \to \infty \)

\[ f_n(t) = \frac{(-1)^n}{n!} \binom{n+1}{r} f^{(n)} \left( \frac{n+1}{r} \right) \]  

(3.67)

where \( f^{(n)}(s) \) denotes the \( n^{th} \) derivative of \( f(s) \).

Abate and Whitt represented \( f_n(t) \) as an integral using the associated generating function and the Cauchy contour integral

\[ f_n(t) = \frac{n+1}{r} \frac{1}{2\pi i} \int_{C} 2\pi e^{iu} \left( 1 - re^{iu} \right) e^{-inu} du \]  

(3.68)

Applying the Fourier-series method, Abate and Whitt obtained the trapezoidal-rule approximation of (3.68)

\[ f_n(t) \approx \frac{n+1}{2rn} \left[ \hat{f} \left( \frac{(n+1)(1-r)}{r} \right) - \hat{f} \left( \frac{(n+1)(1+r)}{r} \right) + 2 \sum_{k=1}^{n-1} (-1)^k \text{Re} \hat{f} \left( \frac{n+1}{r} \left( 1 - r \text{Exp} \left( \frac{\pi ik}{n} \right) \right) \right) \right] \]  

(3.69)
with the discretization error \( e_d \) bounded by, assuming \( |f_n(t)| \leq 1 \),

\[
|e_d| = \frac{p^2n}{1-p^2n}
\]

(3.70)

To achieve a discretization error of at most \( 10^{-\gamma} \), we require

\[
10^{-\gamma} = \frac{p^2n}{1-p^2n}
\]

\[
p^2n = \frac{10^{-\gamma}}{1+10^{-\gamma}}
\]

\[
r = \left( \frac{10^{-\gamma}}{1+10^{-\gamma}} \right)^{1/2}
\]

(3.71)

Like the parameter \( A \) in the Euler method, too small value of \( r \) causes round-off problem in computations. Abate and Whitt stated that roughly \( 1.5 \gamma \)-digit precision was required to achieve an error of \( 10^{-\gamma} \).

As Abate and Whitt indicated, the approximating function \( f_n(t) \) converges to \( f(t) \) quite slowly. They used a linear combination of terms developed by Stehfest (1970) to improve the convergence.

\[
f_{j,m}(t) = \sum_{k=1}^{m} \omega(k, m) f_{j,k}(t)
\]

(3.72)

where the weights \( \omega(k, m) \) are

\[
\omega(k, m) = (-1)^{m-k} \frac{k^n}{k!(m-k)!}
\]

(3.73)

Abate and Whitt suggested that start with \( j = 10 \) and \( m = 6 \), and increase them if necessary. In the application of pricing Asian options, Fu, Madan and Wang (1999) stated that \( j = 10, m = 16 \) and \( \gamma = 8 \) did not always provide the desired accuracy. Craddock, Heath and Platen (2000) claimed the linear combination (3.72) did not improve the convergence noticeably.

- **3.3.3. Fast Fourier transform**

Geman and Eydeland (1995) modified the Bromwich integral (3.18) so the inverse Laplace transform becomes inverse Fourier transform (3.23). They then used fast Fourier transform (FFT) to evaluate the resulting inverse Fourier transform. FFT reduces the number of calculations required for \( N \) points from \( 2N^2 \) to \( 2N \log_2 N \).

Beginning with \( \hat{C}(\lambda + \alpha, q) \) and the expression of the Laplace transform of an Asian option (3.46), we have
\[ \hat{C}(\lambda + \alpha, q) = \frac{\int_0^1 e^{y \frac{w}{2}} (1-y)^{\frac{\mu+2}{2}} \, dy}{(2)^{\frac{\mu+2}{2}} \Gamma(\frac{\mu+2}{2})} \]  

(3.74)

where

\[ \mu = \sqrt{2(\lambda + \alpha) + y^2} \]  

(3.75)

We choose \( \alpha > \max(0, 2 + 2\nu) \) so \( \hat{C}(\lambda + \alpha, q) \) is regular for all \( \lambda \geq 0 \). Defining \( y = 2q x \), equation (3.74) changes to

\[ \hat{C}(\lambda + \alpha, q) = \frac{\int_0^1 e^{y \frac{w}{2}} (1-y)^{\frac{\mu+2}{2}} \, dy}{(2)^{\frac{\mu+2}{2}} \Gamma(\frac{\mu+2}{2})} \]  

(3.76)

The term \( e^{y \frac{w}{2}} \) can be approximated by a polynomial

\[ e^{y \frac{w}{2}} = \sum_{k=0}^{M} \left(-\frac{x}{2q}\right)^k / k! \]  

(3.77)

Substituting (3.77) into (3.76) gives

\[ \hat{C}(\lambda + \alpha, q) = \sum_{k=0}^{M} \frac{1}{k!} \left(-\frac{x}{2q}\right)^k \frac{\Gamma(\frac{\mu+2}{2})}{\Gamma(\frac{\mu+2}{2})} \]  

(3.78)

Applying the beta function \( \int_0^1 t^{\nu-1} (1-t)^{\nu-1} \, dt = B(x, y) \) and the identity \( B(x, y) = \frac{\Gamma(x) \Gamma(y)}{\Gamma(x+y)} \), we have

\[ \hat{C}(\lambda + \alpha, q) = \frac{\sum_{k=0}^{M} \frac{1}{k!} \left(-\frac{x}{2q}\right)^k \frac{\Gamma(\frac{\mu+2}{2})}{\Gamma(\frac{\mu+2}{2})} \frac{\Gamma(\frac{\mu+2}{2}+k)}{\Gamma(\frac{\mu+2}{2}+k)}}{(2)^{\frac{\mu+2}{2}} \Gamma(\frac{\mu+2}{2})} \frac{1}{\Gamma(\frac{\mu+2}{2}+k)} \]  

(3.79)

Since \( \Gamma(x+1) = x \Gamma(x) \), (3.79) is written as

\[ \hat{C}(\lambda + \alpha, q) = \left(\frac{1}{(2)^{\frac{\mu+2}{2}} \Gamma(\frac{\mu+2}{2})} \right) \frac{\Gamma(\frac{\mu+2}{2}+k)}{\Gamma(\frac{\mu+2}{2}+k)} \sum_{k=0}^{M} \rho_k \]  

(3.80)

where \( \rho_k \) is defined recursively by

\[ \rho_0 = 1 \]

\[ \rho_{k+1} = \frac{1}{2q(1+k)} \left(\frac{\mu+1+k}{\mu+1+k}\right) \rho_k \]  

(3.81)

The discrete inverse Fourier transform is given by

\[ x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k \left(\frac{2\pi}{N} \right)^k n, \quad n = 0, ..., N-1 \]  

(3.82)
Geman and Eydeland (1995) used Mathematica to conduct fast Fourier transform.

- **3.3.4. Bromwich integration**

Shaw (1998) exploited the ability of numerical integration in Wolfram Mathematica. He used the syntax: NIntegrate to invert the Laplace transform of an Asian option along the truncated Bromwich contour. Shaw claimed that pricing by numerical integration was effective and highly accurate but one needed to experiment and increase the truncation size for some Asian option with peculiar parameter values. Fu, Madan and Wang (1999) obtained an entirely wrong price of an Asian option with \( \sigma = 0.1 \) simply because the truncation size chosen was insufficient.

Using Bromwich integral (3.18) to invert the Laplace transform of an Asian option, we have

\[
C^{(q)}(h, q) = \frac{1}{2\pi i} \int_{a-i\infty}^{a+i\infty} e^{\lambda t} \hat{C}(\lambda, q) d\lambda
\]  

(3.83)

Let \( \lambda = a + i u \) where \( a > \max(0, 2 + 2n) \). Thus, all singularities of \( \hat{C}(\lambda, q) \) are to the left of the contour.

Moreover, optimal choice of \( a \) is just greater than \( \max(0, 2 + 2n) \) as too large \( a \) deteriorates the accuracy. Then

\[
C^{(q)}(h, q) = \frac{1}{2\pi i} \int_{a-i\infty}^{a+i\infty} e^{(a+i u)t} \hat{C}(a + i u, q) d(a + i u)
\]

\[
= \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{a+iu} e^{iu} \hat{C}(a + i u, q) du
\]

\[
= \frac{1}{\pi} \int_{0}^{\infty} e^{a+iu} \hat{C}(a + i u, q) du
\]  

(3.84)

Finally, \( C^{(q)}(h, q) \) is approximated by truncating the integration range. Larger truncation size gives more accurate result but also demands more computation time.

- **3.3.5. Gaver-Wynn-Rho algorithm**

In the numerical analysis literature, Abate and Valkó (2004) presented two procedures for the numerical inversion of Laplace transforms in a multi-precision computing environment, showing the use of such an environment to be essential for accurate Laplace transform inversion. For some inversion algorithms, round-off errors increase dramatically as one pursues high accuracy in a fixed-precision computing environment. Hence, one needs to increase the computing precision when round-off errors cause a problem. Abate and Valkó (2004) suggested letting the algorithm determine the level of precision. We claim that the algorithm does not always generate correct level of precision for all areas of application. The number of required precision needs to be confirmed by experiment in the application of pricing Asian options.
The Gaver-Wynn-Rho (GWR) algorithm presented by Abate and Valkó (2004) is based on the Gaver functionals accelerated by the Wynn rho scheme. It requires evaluation of the transform only on the real axis. Starting with the Post-Widder Theorem (3.67), Gaver (1966) presented a discrete analogy which avoids computing high-order derivatives.

\[ f_k(t) = \frac{(-1)^{\alpha k}}{t} \binom{2k}{k} \Delta^k \hat{f}(\frac{k \alpha}{t}) \]  

(3.85)

where \( \alpha = \log(2) \) and \( \Delta \) denotes the difference operator, i.e., \( \Delta \hat{f}(n x) = \hat{f}((n + 1) x) - \hat{f}(n x) \). Expanding the difference operator yields

\[ f_k(t) = \frac{\alpha k}{t} \binom{2k}{k} \sum_{j=0}^{k} (-1)^j \binom{k}{j} \hat{f}(\frac{(k+j) \alpha}{t}) \]  

(3.86)

\( f_k(t) \) can be computed by the Gaver functionals

\[ G_0^{(n)} = \frac{\alpha n}{t} \hat{f}(\frac{n \alpha}{t}), \quad 1 \leq n \leq 2M \]

\[ G_k^{(n)} = \left(1 + \frac{n}{k}\right) G_{k-1}^{(n)} - \left(\frac{n}{k}\right) G_{k-1}^{(n+1)}, \quad k \geq 1, \quad n \geq k \]  

(3.87)

Unfortunately, the sequence \( f_k(t) \) approximates \( f(t) \) very poorly due to low convergence. Abate and Valkó (2004) used the Wynn rho algorithm to accelerate the Gaver functionals. This acceleration scheme is given by

\[ \rho_{k-1}^{(n)} = 0, \quad n \geq 0 \]

\[ \rho_0^{(n)} = f_n(t), \quad n \geq 0 \]

\[ \rho_k^{(n)} = \rho_{k-2}^{(n+1)} + \frac{k}{\rho_{k-1}^{(n+2)} - \rho_{k-1}^{(n+1)}}, \quad k \geq 1 \]  

(3.88)

\[ f(t, M) = \rho_M^{(0)} \]

The value of \( f(t) \) is approximated by \( f(t, M) \) where \( M \) is an even integer.

The computation of the GWR algorithm is unstable due to the round-off error. Given a fixed computing precision, the accuracy of \( f(t, M) \) increases to a point and thereafter decreases quickly as one increases \( M \). To solve this problem, the computation should be done in a multi-precision environment. Abate and Valkó made the computing precision larger as \( M \) increases as

\[ \text{Number of digits of precision} = 2.1M \]  

(3.89)
Abate and Valkó (2004) found the relative error estimate of the GWR algorithm is

\[
\text{relative error} = \left| \frac{f(t_0, t, M)}{f(t_0)} \right| \approx 10^{-0.8M}
\]  

(3.90)

That is, the accuracy of the result is about 0.8\(M\) significant digits.

### 3.3.6. Fixed Talbot algorithm

The second procedure in Abate and Valkó (2004) is the fixed Talbot (FT) algorithm based on deforming the Bromwich contour in (3.18). The convergence of the Bromwich integral improves greatly if the contour takes on large negative value for its real part. Thus, it is reasonable to deform the contour into any open path wrapping around the negative real axis as long as no singularity of \(\hat{f}(s)\) is crossed by the path. Talbot (1979) carefully chose the path in the form

\[
s(\theta) = \gamma \theta (\cot \theta + i), \quad -\pi < \theta < +\pi
\]  

(3.91)

There are two parameters in Talbot's path, i.e. \(\gamma\) and \(\theta\). Abate and Valkó (2004) replaced the Bromwich contour with this path, then

\[
f(t) = \frac{1}{2\pi i} \int_{-\pi}^{\pi} e^{i\lambda(\theta)} \hat{f}(s(\theta)) s'(\theta) d\theta
\]

\[
= \frac{\gamma}{\pi} \int_{0}^{\pi} \text{Re}\left[ e^{i\lambda(\theta)} \hat{f}(s(\theta)) \right. (1 + i \zeta(\theta)) d\theta
\]  

(3.92)

where

\[
\sigma(\theta) = \theta + (\theta \cot \theta - 1) \cot \theta
\]  

(3.93)

Use the trapezoidal rule with step size \(\pi / M\) and \(\theta_k = k \pi / M\) to approximate the integral

\[
f(t, M) = \frac{r}{M} \left\{ \frac{1}{2} \hat{f}(\gamma) e^{\epsilon t} + \sum_{q=1}^{M-1} \text{Re}\left[ e^{i\lambda(\theta_k)} \hat{f}(s(\theta_k)) \right. (1 + i \sigma(\theta_k)) \right\}
\]  

(3.94)

Abate and Valkó (2004) fixed the value of the parameter \(r\) according to numerical experiments

\[
r = \frac{2 M}{5t}
\]  

(3.95)

Thus, \(f(t)\) is approximated by \(f(t, M)\). The computing precision is specified by

\[
\text{Number of digits of precision} = M
\]  

(3.96)

The relative error estimate of the FT algorithm is

\[
\left| \frac{f(t_0, t, M)}{f(t_0)} \right| \approx 10^{-0.6M}
\]  

(3.97)

Therefore, the accuracy produced is about 0.6\(M\) significant digits.
### 3.3.7. Laguerre method

Abate, Choudhury and Whitt (1996) presented a new variant of the Laguerre method for numerically inverting the Laplace transform of a real-valued function \( f(t) \) on the nonnegative real line. The Laplace transform \( \hat{f}(s) \) is defined by

\[
\hat{f}(s) = \int_0^\infty e^{-st} f(t) \, dt
\]

(3.98)

Assume the Laplace transform is well defined, that is, analytic for \( \text{Re}(s) > 0 \). Following Abate, Choudhury and Whitt (1996), the Laguerre method is derived from the Laguerre-series representation

\[
f(t) = \sum_{n=0}^\infty q_n I_n(t), \quad t \geq 0
\]

(3.99)

where \( I_n(t) \) are the associated Laguerre functions. They can be calculated from the Laguerre polynomials \( L_n \),

\[
I_n(t) = e^{-t/2} L_n(t), \quad t \geq 0
\]

\[
L_n(t) = \sum_{k=0}^n \frac{n^k t^k}{k!}, \quad t \geq 0
\]

(3.100)

and \( q_n \) are the Laguerre coefficients which are embedded in the Laguerre generating function \( Q(z) \)

\[
Q(z) = \sum_{n=0}^\infty q_n z^n = \frac{1}{1-z} \hat{f}\left(\frac{1+z}{2(1-z)}\right)
\]

(3.101)

\( q_0 \) can be calculated from the above Laguerre generating function by letting \( z = 0 \). So, we have

\[
q_0 = Q(0)
\]

(3.102)

The expression for the coefficients \( q_n \) with \( n \geq 1 \) can be obtained via Cauchy contour integrals

\[
q_n = \frac{1}{2\pi i} \oint_{C_r} Q(z) z^{-(n+1)} \, dz
\]

(3.103)

where \( C_r \) is a circle centered at the origin with the radius \( r \), \( 0 < r < 1 \), such that \( Q(z) \) is analytic in \( \{z : |z| < r\} \). By making the change of variable \( z = re^{iu} \), \( 0 \leq u \leq 2\pi \), we have

\[
q_n = \frac{1}{2\pi i} \int_0^{2\pi} Q(re^{iu}) (re^{iu})^{-(n+1)} \, d(re^{iu})
\]

\[
= \frac{1}{2\pi i} \int_0^{2\pi} Q(re^{iu}) (re^{iu})^{-(n+1)} e^{iu} \, du
\]

\[
= \frac{1}{2\pi i} \int_0^{2\pi} Q(re^{iu}) e^{-iu} \, du
\]

\[
= \frac{1}{2\pi i} \int_0^{2\pi} Q(re^{iu}) e^{-i\pi u} \, du
\]

(3.104)

Then, we approximate the integral by the trapezoidal rule defined as
\[ I = \int_{a \Delta}^{b \Delta} g(u) \, du \]
\[ = h \left[ \frac{g(a) + g(b)}{2} + \sum_{j=1}^{m-1} g(a + j \, h) \right] \]  

(3.105)

with the step size

\[ h = \frac{b-a}{m} \]  

(3.106)

In this case, we set \( a = 0, \ b = 2\pi, \ h = 2\pi / m \) and

\[ g(u) = Q(r \, e^{iu}) \, e^{-iu} \]  

(3.107)

Then,

\[ g(a) = g(0) = Q(r) \]
\[ g(b) = g(2\pi) = Q(r) \]
\[ g(a + j \, h) = g\left(\frac{2\pi j}{m}\right) = \text{Re}\left(Q\left(r \, e^{\frac{2\pi j}{m}}\right) e^{-i\frac{2\pi j}{m}}\right) \]  

(3.108)

Hence,

\[ \int_0^{2\pi} Q(r \, e^{iu}) \, e^{-iu} \, du = \frac{2\pi}{m} \left\{ \frac{Q(r) + Q(-r)}{2} + \sum_{j=1}^{m-1} \text{Re}\left(Q\left(r \, e^{\frac{2\pi j}{m}}\right) e^{-i\frac{2\pi j}{m}}\right) \right\} \]
\[ = \frac{2\pi}{m} \left\{ Q(r) + \sum_{j=1}^{m-1} \text{Re}\left(Q\left(r \, e^{\frac{2\pi j}{m}}\right) e^{-i\frac{2\pi j}{m}}\right) \right\} \]  

(3.109)

As a result,

\[ q_n = \frac{1}{2\pi} \int_0^{2\pi} Q(r \, e^{iu}) \, e^{-iu} \, du \]
\[ = \frac{1}{m \, r^2} \left\{ Q(r) + \sum_{j=1}^{m-1} \text{Re}\left(Q\left(r \, e^{\frac{2\pi j}{m}}\right) e^{-i\frac{2\pi j}{m}}\right) \right\} \]  

(3.110)

This expression for \( q_n \) is quite different from that given by Abate, Choudhury and Whitt (1996) where the formula is

\[ \tilde{q}_n = \frac{1}{2\pi} \left\{ Q(r) + (-1)^n \, Q(-r) + 2 \sum_{j=1}^{n} (-1)^j \, \text{Re}\left(Q\left(r \, e^{\frac{2\pi j}{n}}\right)\right) \right\} \]  

(3.111)

We remark that the above formula for \( \tilde{q}_n \) is wrong. Using it will lead to an incorrect result. Our expression for \( q_n \) has one more parameter \( m \) which comes from the step size \( h = 2\pi / m \).

We also notice that, to make the Laguerre method work, the truncation size in the Laguerre-series representation cannot be arbitrary. The Laguerre-series representation, then, becomes

\[ f(t) = \sum_{n=0}^{n_0} q_n \, b_n(t), \quad t \geq 0 \]  

(3.112)

The truncation size \( n_0 \) must be less than or equal to \( m - 1 \) which is exactly the upper limit of the summation in the expression for \( q_n \). Otherwise, the result degenerates. Further experiments show using
\[ n_0 = m - 1 \] yields the maximum accuracy. Considering that it is very easy to calculate \( l_n(t) \) compared with computing \( q_n \), we suggest setting \( n_0 = m - 1 \).

If \( |q_n| \leq C \), the discretization error \( e_d \) in \( q_n \) associated with the trapezoidal rule is bounded by, as shown in Abate, Choudhury and Whitt (1996),

\[ |e_d| \leq C \frac{r^2}{1 - r^2} \approx C r^2 \quad (3.113) \]

Assume \( C = 1 \). To have discretization error of less than \( 10^{-7} \), \( r \) is set to be

\[ r = 10^{-7/n} \quad (3.114) \]

However the above setting for \( r \) is not justified. The discretization error should be a function of the parameter \( m \) of the trapezoidal rule rather than the index \( n \) in the Laguerre-series representation. Experiment results imply the correct setting for \( r \) could be

\[ r = 10^{-7/n} \quad (3.115) \]

though it has not been mathematically proved yet.

**Application of Wynn’s \( \epsilon \)-algorithm**

It is usually the case that the convergence rate of \( l_n(t) \) as \( n \to \infty \) is very slow. Abate, Choudhury and Whitt (1996) proposed using Wynn’s \( \epsilon \)-algorithm to accelerate the convergence of the Laguerre series. The Wynn’s \( \epsilon \)-algorithm is defined by the following recursion

\[
\begin{align*}
\epsilon_{n+1}^0 &= 0 \\
\epsilon_n^0 &= S_n \\
\epsilon_k^l &= \frac{\epsilon_k^{l+1} + (\epsilon_k^{l+1} - \epsilon_k^{l-1})^{-1}}{2k+1}
\end{align*}
\]  

(3.116)

where \( S_n \) is the \( n^{\text{th}} \) partial sum of the Laguerre series. Finally, the function \( f(t) \) is approximated by \( \epsilon_{2k}^{m-1-2k} \). Note this setting is different from that in Abate, Choudhury and Whitt (1996) which uses \( \epsilon_{2k}^n \) with \( n \) an arbitrary integer. Here \( \epsilon_{2k}^{m-1-2k} \) sets a restriction so that the largest partial sum of the Laguerre series used in the recursion is \( S_{m-1} \). This ensures that the truncation size in the Laguerre series is less than or equal to the upper limit of the summation in \( q_n \).

**Scaling**

Using Wynn’s \( \epsilon \)-algorithm alone sometimes can not provide good acceleration, especially in the problem of pricing an Asian option. Combing Wynn’s \( \epsilon \)-algorithm with suitable scaling can improve the accuracy
of the Lapierre method substantially. Following Abate, Choudhury and Whitt (1996), the function \( f(t) \) is scaled to \( f_{\sigma, b}(t) \) with two parameters non-negative \( \sigma \) and positive \( b \).

\[
f_{\sigma, b}(t) = e^{-\sigma t} f(t / b), \quad t \geq 0
\]  

(3.117)

To recover \( f(t) \) from \( f_{\sigma, b}(t) \), we use

\[
f(t) = e^{\sigma b t} f_{\sigma, b}(b t)
\]

(3.118)

Note \( f(t) = f_{0, 1}(t) \). The Laguerre generating function associated with \( f_{\sigma, b}(t) \) is given by

\[
Q_{\sigma, b}(z) = \frac{b}{1-z} \hat{f} \left( \frac{b(1+z)}{2(1-z)} + b \sigma \right)
\]

(3.119)

The parameter \( \sigma \) is usually set to zero.

- 3.3.8. Unified Gaver-Stehfest algorithm

Abate and Whitt (2006) introduced a unified framework that can encompass a wide range of algorithms to invert Laplace transforms numerically. Given a Laplace transform \( \hat{f}(s) \) of a real-valued function \( f(t), \quad t \geq 0 \), the function \( f(t) \) is approximated by a finite linear combination of the transform values.

\[
f(t) = f_n(t) \equiv \frac{1}{\pi} \sum_{k=0}^{n} \omega_k \hat{f} \left( \frac{\alpha_k}{\pi} \right), \quad 0 < t < \infty
\]

(3.120)

where the weights \( \omega_k \) and nodes \( \alpha_k \) only depend on \( n \). In other words, they do not depend on either the transform \( \hat{f}(s) \) or the time \( t \). Both \( \omega_k \) and \( \alpha_k \) are complex. The inversion algorithms need to fit in the framework (3.120), which means the algorithms have to be linear. Furthermore, the acceleration schemes also have to be linear if the acceleration techniques are used.

In the application of pricing an Asian option, \( f(t) \) is real-valued, then the price is obtained by taking the real part

\[
\text{Re}\{f(t)\} \equiv \text{Re}\{f_n(t)\} = \frac{1}{\pi} \sum_{k=0}^{n} \omega_k \hat{f} \left( \frac{\alpha_k}{\pi} \right)
\]

(3.121)

As in Abate and Valkó (2004), Abate and Whitt suggested using multi-precision computing environment to implement the unified framework. The required precision depends on the specific procedure used for inversion. Abate and Whitt investigated three widely used procedures in the framework: (i) the Euler algorithm using Fourier series expansions with Euler summation, (ii) the Gaver-Stehfest algorithm using the Salzer summation to accelerate the Gaver functionals, and (iii) the Talbot’s method based on
deforming the Bromwich contour. Note that the Euler algorithm is the same as the Euler method discussed in Section 3.3.1. The Gaver-Stehfest algorithm is analogous to the Gaver-Wynn-Rho algorithm in Section 3.3.5 of which the acceleration scheme is the Wynn rho algorithm.

In comparison to the Gaver-Wynn-Rho algorithm, the unified Gaver-Stehfest algorithm uses the linear Salzer acceleration scheme proposed by Stehfest (1970) to tackle the poor convergence of the Gaver functionals. Valkó and Abate (2004) reviewed six sequence accelerators for the Gaver functionals and found that the Salzer scheme performed surprisingly well being outperformed only by the nonlinear Wynn rho algorithm. The Salzer scheme is used here simply because the accelerator fits the framework.

Start with the sequence of Gaver approximations

\[ f_n(t) = \frac{n \log(2)}{t} \left( \sum_{k=0}^{n} (-1)^k \binom{n}{k} \hat{f}(\frac{n+k \log(2)}{t}) \right) \]  

(3.122)

Apply the Salzer summation scheme to accelerate the convergence \( f_n(t) \to f(t) \) as \( n \to \infty \)

\[ f_{\xi}(t, M) = \sum_{n=M}^{n=M} (-1)^{n+M} \left( \frac{n!}{M!} \right) \frac{M}{n} f_n(t) \]  

(3.123)

Substituting (3.122) into (3.123) and rearranging the double summation yields the unified Gaver-Stehfest algorithm

\[ f_{\xi}(t, M) = \frac{\log(2)}{t} \sum_{k=1}^{k=M} \xi_k \hat{f}(\frac{1 \log(2)}{t}) \]  

(3.124)

where

\[ \xi_k = (-1)^{M+k} \sum_{j=\lceil k+1/2 \rceil}^{M} \frac{M}{j} \binom{M}{j} \binom{j}{k-j} \]  

(3.125)

where \( \lfloor x \rfloor \) denotes the greatest integer less than or equal to \( x \) and \( k \wedge M \) stands for \( \min[k, M] \).

The unified Gaver-Stehfest algorithm (3.124) is in the unified framework (3.120) with \( n = 2M \), \( \omega_k = \log(2) \xi_k \) and \( \alpha_k = k \log(2) \). The evaluation of the unified Gaver-Stehfest algorithm requires high working precision. Based on the experiments in Abate and Valkó (2004), Abate and Whitt (2006) concluded that if the inversion parameter was \( M \), the system precision was required to be about \( 2.2M \), and the result was accurate to about \( 0.9M \) significant digits. Hence, given the inversion parameter \( M \), the relative error of the unified Gaver-Stehfest algorithm is
relative error \[ = \left| \frac{f(\theta) - f_{\theta}(\theta, M)}{f(\theta)} \right| \approx 10^{-0.9M} \] \hspace{1cm} (3.126)

The efficiency is measured by the ratio of the significant digits of the accuracy to that of required system precision

\[ \text{eff}(\mathcal{G}) = \frac{\text{accuracy produced}}{\text{system precision required}} \approx \frac{0.9M}{2.2M} \approx 0.4 \] \hspace{1cm} (3.127)

### 3.3.9. Unified Euler algorithm

Using the Euler algorithm summarized in Abate, Choudhury and Whitt (1999), Abate and Whitt (2006) obtained the unified Euler algorithm by fixing the Euler parameter vector \((l, m, n, A)\) to be \((1, M, M, 2 \log(2) M / 3)\). The unified Euler algorithm is

\[ f_{\theta}(t, M) = \frac{10^{12.5}}{t} \sum_{k=0}^{2M} \eta_k \Re \left[ \hat{f} \left( \frac{\beta_k}{t} \right) \right] \] \hspace{1cm} (3.128)

where

\[ \beta_k = \frac{M \log 10}{3} + \pi i k \]

\[ \eta_k \equiv (-1)^k \xi_k \] \hspace{1cm} (3.129)

with

\[ \xi_0 = \frac{1}{2} \]

\[ \xi_k = 1, \quad 1 \leq k \leq M \]

\[ \xi_{2M} = \frac{1}{2} \]

\[ \xi_{2M-k} = \xi_{2M-k+1} + 2^{-M} \binom{M}{k}, \quad 0 < k < M \] \hspace{1cm} (3.130)

The unified Euler algorithm (3.128) is in the unified framework (3.120) with \(n = 2M\), \( \omega_k = 10^{M/3} \eta_k \) and \( \alpha_k = \beta_k \). The unified Euler algorithm is relatively more efficient. Given the inversion parameter \(M\), the required system precision is about \(M\), and accuracy of the result is about \(0.6M\) significant digits. Therefore, the relative error of the unified Euler algorithm is

\[ \text{relative error} = \left| \frac{f(\theta) - f_{\theta}(\theta, M)}{f(\theta)} \right| \approx 10^{-0.6M} \] \hspace{1cm} (3.131)

The efficiency is

\[ \text{eff}(\mathcal{E}) = \frac{\text{accuracy produced}}{\text{system precision required}} \approx \frac{0.6M}{M} \approx 0.6 \] \hspace{1cm} (3.132)

### 3.3.10. Unified Talbot algorithm

Abate and Whitt (2006) acquired the unified Talbot algorithm by fixing the parameters of the Talbot’s
method in Abate and Valkó (2004). The unified Talbot algorithm in the framework is given by

\[ f_b(t, M) = \frac{2}{\pi} \sum_{k=0}^{M-1} \text{Re}\left[ \gamma_k \hat{f}\left( \frac{\theta_k}{\pi} \right) \right] \]  \hspace{1cm} (3.133)

where

\[ \begin{align*}
\theta_0 &= \frac{2M}{\pi} \\
\theta_k &= \frac{2k\pi}{\pi} \left( \cot \left( \frac{k\pi}{M} \right) + i \right), \quad 0 < k < M \\
\gamma_0 &= \frac{1}{2} e^{\theta_0} \\
\gamma_k &= \left\{ 1 + i \left( \frac{k\pi}{M} \right) \left[ 1 + \cot^2 \left( \frac{k\pi}{M} \right) \right] - i \cot \left( \frac{k\pi}{M} \right) \right\} e^{\theta_k}, \quad 0 < k < M
\end{align*} \]  \hspace{1cm} (3.134)

The unified Talbot algorithm (3.133) is in the unified framework (3.120) with \( n = M, \omega_k = (2/5) \gamma_k \) and \( \alpha_k = \theta_k \). Given the inversion parameter \( M \), the required system precision is about \( M \), and accuracy of the result is about \( 0.6M \) significant digits. Thus, the relative error of the unified Talbot algorithm is

\[ \text{relative error} = \left| \frac{f(t; f(t; M))}{f(t)} \right| \approx 10^{-0.6M} \]  \hspace{1cm} (3.135)

The efficiency is

\[ \text{eff}(T) = \frac{\text{accuracy produced}}{\text{system precision required}} \approx \frac{0.6M}{M} = 0.6 \]  \hspace{1cm} (3.136)

### 3.3.11. Two-dimensional inversion algorithms

Abate and Whitt (2006) showed that the simple framework (3.120) can easily extend to the case of multidimensional Laplace transforms. For a two-dimensional Laplace transform \( \hat{f}(s_1, s_2) \), \( s_1, s_2 \in \mathbb{C} \) of a real-valued function \( f(t_1, t_2), t_1, t_2 \geq 0 \), the function \( f(t_1, t_2) \) is calculated approximately by

\[ f(t_1, t_2) \approx f_{n_1, n_2}(t_1, t_2) = \frac{1}{t_1 t_2} \sum_{k_1=0}^{n_1} \omega_{k_1} \sum_{k_2=0}^{n_2} \omega_{k_2}' \hat{f}\left( \frac{n_1}{t_1}, \frac{n_2}{t_2} \right) \]  \hspace{1cm} (3.137)

where \( n_1 \) and \( n_2 \) are the inversion parameters, the weights \( \omega_{k_1} \) and nodes \( \alpha_{k_1} \) are for the outer loop, and the weight \( \omega_{k_2}' \) and nodes \( \alpha_{k_2}' \) are for the inner loop. As in one-dimensional case, the real part is used when \( f(t_1, t_2) \) is real-valued.

Given the three one-dimensional Laplace transform inversion algorithms, namely Gaver (G), Euler (E) and Talbot (T), Abate and Whitt (2006) combined them in every possible permutation to get nine two-dimensional inversion algorithms. They found that combining two different one-dimensional inversion procedures can be beneficial.

The three one-dimensional Laplace transform inversion algorithms denoted by \( G, E \) and \( T \) respectively can be combined to form nine two-dimensional inversion algorithms with the notation \( GE, EGE, GT, ET, EGT, GE, GTE, TGE, TEG \).
and so forth. The notation $\mathcal{G}E$, for example, stands for the operation where the first operator $\mathcal{G}$ applies to the outer loop while the second operator $E$ applies to the inner loop. Abate and Whitt (2006) let the inversion parameter for the outer loop be $M$ and that of the inner loop be $cM$. So, the two-dimensional algorithms still have only one parameter $M$. Table 3.1 summarizes both one-dimensional inversion algorithms and two-dimensional inversion algorithms where $\xi_k$ are the Gaver weights, $\eta_k$ and $\beta_k$ are the Euler weights and nodes, and $\gamma_k$ and $\theta_k$ are the Talbot weights and nodes. We need to keep in mind that the weights and nodes in the outer loop depend on $M$ while the weights and nodes in the inner loop depend on $cM$. Thus, the weights and nodes with a prime in the inner loop indicate that these parameters depend on $cM$ not $M$. It occurs in only three cases, i.e., $\mathcal{G}(M)T(3M)$, $\mathcal{G}(M)\mathcal{G}(2M)$ and $\mathcal{G}(M)\mathcal{E}(3M)$. The weights and nodes with a bar represents complex conjugates.

<table>
<thead>
<tr>
<th>$\mathcal{G}(M)$</th>
<th>$\frac{1}{e} \sum_{k=1}^{M} \xi_k \hat{F} \left( \frac{1}{e} \xi_k \right)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{E}(M)$</td>
<td>$\frac{10^{\gamma/2}}{e} \sum_{k=1}^{M} \eta_k \text{Re} \left[ \hat{F} \left( \frac{\eta_k}{e} \right) \right]$</td>
</tr>
<tr>
<td>$T(M)$</td>
<td>$\frac{2}{5} \sum_{k=1}^{M} \eta_k \text{Re} \left[ \hat{F} \left( \frac{\eta_k}{e} \right) \right]$</td>
</tr>
<tr>
<td>$T(M) \mathcal{G}(M)$</td>
<td>$\frac{2 \ln^2}{5 \varepsilon_1 \varepsilon_2} \sum_{k=1}^{M} \eta_k \text{Re} \left[ \hat{F} \left( \frac{\eta_k}{e} \right) \right]$</td>
</tr>
<tr>
<td>$T(M) \mathcal{E}(M)$</td>
<td>$\frac{10^{\gamma/2} \ln^2}{5 \varepsilon_1 \varepsilon_2} \sum_{k=1}^{M} \eta_k \text{Re} \left[ \hat{F} \left( \frac{\eta_k}{e} \right) \right]$</td>
</tr>
<tr>
<td>$T(M) \mathcal{E}(M) \mathcal{G}(M)$</td>
<td>$\frac{10^{\gamma/2} \ln^2}{5 \varepsilon_1 \varepsilon_2} \sum_{k=1}^{M} \eta_k \text{Re} \left[ \hat{F} \left( \frac{\eta_k}{e} \right) \right]$</td>
</tr>
<tr>
<td>$T(M) \mathcal{E}(M) \mathcal{G}(M) \mathcal{E}(3M)$</td>
<td>$\frac{10^{\gamma/2} \ln^2}{5 \varepsilon_1 \varepsilon_2} \sum_{k=1}^{M} \eta_k \text{Re} \left[ \hat{F} \left( \frac{\eta_k}{e} \right) \right]$</td>
</tr>
</tbody>
</table>

### 3.4. Asymptotic method

Shaw realised that it was very difficult to numerically invert the Laplace transform of an Asian option when volatility is low. The computing time grows exponentially as $\sigma \to 0$. Shaw (2002) used the asymptotic method based on Mellin transform to obtain an asymptotic series, which improves the computation time dramatically with good accuracy retained.
Exploiting the ability of symbolic computation of Mathematica, Shaw (2002) expressed the Laplace transform of an Asian option as

\[ U = \frac{2^{\frac{1}{2}(\mu+y+\lambda)} q^{\frac{1}{2}(\nu+\rho+\lambda)}}{\lambda(\lambda-2y-2)\Gamma(\mu+1)} \]

(3.138)

where \( \mu = \sqrt{2\lambda + y^2} \) and \( _1F_1(a; b; z) \) denotes the Kummer confluent hypergeometric function.

Equation (3.138) is equivalent to (3.48) because

\[ _1F_1(a; b; z) = \Gamma(b) \ _1\tilde{F}_1(a; b; z) \]  

(3.139)

Let \( U = \frac{U_N}{\lambda(\lambda-2y-2)} \) and \( t = 2q \), then

\[ U_N = \frac{\Gamma\left(\frac{1}{2}(\nu+\lambda+4)\right)}{\Gamma(\mu+1)} \Gamma\left(\frac{1}{2}\right) \ _1F_1\left(\frac{1}{2}, \mu + 1; \mu + 1; -\frac{1}{t}\right) \]  

(3.140)

We are interested in the limit of \( t \mu \) as \( t \to 0 \) and \( \mu \to \infty \). Using the Dirichlet series, \( U_N \) can be written in terms of its Mellin transform

\[ U_N = \frac{1}{2\pi i} \frac{\Gamma(c)}{\Gamma(a)} \int_{-i\infty}^{i\infty} t^{-z} \frac{\Gamma(\sigma) \Gamma(a-\sigma)}{\Gamma(c+\sigma)} \, dz \]  

(3.141)

where

\[ a = \frac{\mu-y}{2} - 1 \]
\[ c = \frac{\mu+y}{2} + 2 \]  

(3.142)

With this parameter setting, (3.140) can be represented compactly as

\[ U_N = \frac{\Gamma(c)}{\Gamma(a+c)} t^{-a} \ _1F_1\left(a; a+c; -\frac{1}{t}\right) \]  

(3.143)

Knowing the Mellin transform identity

\[ (1+t)^{-a} = \frac{1}{2\pi i} \int_{-i\infty}^{i\infty} t^{-z} \frac{\Gamma(z) \Gamma(a-z)}{\Gamma(a)} \, dz \]  

(3.144)

we rewrite \( U_N \) in (3.141) as

\[ U_N = \frac{1}{2\pi i} \int_{-i\infty}^{i\infty} t^{-z} \frac{\Gamma(z) \Gamma(a-z)}{\Gamma(a)} \frac{\Gamma(c)}{\Gamma(c+z)} \, dz \]  

(3.145)

Note \( c \) approaches infinity as \( \sigma \to 0 \). Applying the gamma quotient identity given by

\[ \frac{\Gamma(z+a)}{\Gamma(z+b)} \sim s^{z-B} \text{ as } s \to \infty \]  

(3.146)

we have
\[
\frac{\Gamma(c)}{\Gamma(c+2)} \sim c^{-2} \text{ as } c \to \infty
\]  
(3.147)

Therefore
\[
U_N \sim \frac{1}{2\pi i} \int_{-i\infty}^{i\infty} (c t)^{-2} \frac{\Gamma(c+2)}{\Gamma(a)} \, dz = (1 + c t)^{-a}
\]  
(3.148)

(3.148) shows a geometric limit for \( U_N \) as \( \sigma \to 0 \). Note \( c \sim \nu + 2 \) as well. The gamma quotient identity has some flexibility. \( c^{-2} \) can be replaced with \( s^{-2} \) where \( s \) is any number providing \( A - B = z \) and \( A + B + 2 s = 2 c + z \). Hence, we have
\[
U_N \sim \frac{1}{2\pi i} \int_{-i\infty}^{i\infty} (s t)^{-2} \frac{\Gamma(a+2)}{\Gamma(a)} \, dz = (1 + s t)^{-a}
\]  
(3.149)

\( (1 + s t)^{-a} \) can be used as the leading term in an asymptotic series. Shaw found it was optimal to choose \( s = (\mu + \nu) / 2 + 1 \) because \( s \sim \nu + 1 \). To expand \( U_N \) as an asymptotic series, we apply a relevant series proposed by Paris and Kaminski (2001)
\[
\frac{\Gamma(s+A)}{\Gamma(s+B)} \sim s^{A-B} \sum_{j=0}^{\infty} C_j(A, B) s^{-j}
\]  
(3.150)

Shaw used Wolfram Mathematica to obtain the asymptotic series of the Mellin transform and then invert it numerically. He stated that a 13-term asymptotic series is a good compromise between the accuracy and the efficiency in practice.

### 3.5. PDE method

The problem of pricing an Asian option can be solved by a PDE in two space dimensions (see Ingersoll 1987). However, the solution of a two-dimensional PDE tends to be oscillatory. Rogers and Shi (1995) reduced the two-dimensional PDE to a one-dimensional PDE which can model both the fixed and the floating strike Asian options. Unfortunately, it is difficult to solve their one-dimensional PDE numerically. Večer (2001, 2002) derived an alternative one-dimensional PDE by treating the Asian option as a traded account. The resulting one-dimensional PDE can be easily implemented and the results are fast and accurate even for low volatility and short maturity cases.

An option on a traded account is a contract which allows the holder of the option to buy or sell the underlying asset according to the trading strategy \( q_t \) during the life of the option. At maturity of the option, the holder receives the wealth in his final account which is equivalent to the payoff of the option with the strike \( K = 0 \). Options on a traded account generalize many options, i.e., not only Asian options with both fixed and floating strike but also European, American, passport and vacation options. We here
concentrate on the arithmetic average Asian option with the fixed strike. Večer (2002) first replicated the Asian forward as an option on a traded account.

Under the risk-neutral probability measure, the underlying stock price is driven by

$$dS_t = S_t (r - \delta) dt + \sigma S_t dW_t$$

(3.151)

where $r$ is the interest rate, $\delta$ is the continuous dividend yield, and $\sigma$ is the volatility of the underlying stock. Večer (2002) carefully chose the trading strategy $q_t$ to be

$$q_t = \frac{1}{(r-\delta)T} \left( e^{-\delta(T-t)} - e^{-r(T-t)} \right)$$

(3.152)

The value of the option holder's account according to the trading strategy $q_t$ satisfies

$$dX_t = q_t dS_t + r(X_t - q_t S_t) dt + \delta q_t S_t dt$$

$$= r X_t dt + q_t (dS_t - r S_t dt + \delta S_t dt)$$

(3.153)

with the initial wealth

$$X_0 = q_0 S_0 = \frac{1}{(r-\delta)T} \left( e^{-\delta T} - e^{-r T} \right) S_0$$

(3.154)

Since

$$d\left( e^{(T-t)} X_t \right) = e^{(T-t)} dX_t - r e^{(T-t)} X_t dt$$

(3.155)

and

$$d\left( e^{(T-t)} q_t S_t \right) = e^{(T-t)} q_t dS_t - r e^{(T-t)} q_t S_t dt + e^{(T-t)} S_t d q_t + e^{(T-t)} q_t S_t d q_t$$

(3.156)

we have

$$dX_t = r X_t dt + q_t (dS_t - r S_t dt + \delta S_t dt)$$

$$d\left( e^{(T-t)} X_t \right) = e^{(T-t)} X_t dt + e^{(T-t)} dX_t + e^{(T-t)} q_t (dS_t - r S_t dt + \delta S_t dt)$$

$$e^{(T-t)} dX_t = r e^{(T-t)} X_t dt = e^{(T-t)} q_t dS_t - r e^{(T-t)} q_t S_t dt + \delta e^{(T-t)} q_t S_t dt$$

$$e^{(T-t)} dX_t = r e^{(T-t)} X_t dt = e^{(T-t)} q_t dS_t - r e^{(T-t)} q_t S_t dt + e^{(T-t)} S_t d q_t + \delta e^{(T-t)} q_t S_t dt - e^{(T-t)} S_t d q_t$$

$$d\left( e^{(T-t)} q_t S_t \right) = d\left( e^{(T-t)} q_t S_t \right) + e^{(T-t)} q_t S_t d q_t - d q_t$$

(3.157)

$$\int_0^T d\left( e^{(T-t)} X_t \right) = \int_0^T d\left( e^{(T-t)} q_t S_t \right) + \int_0^T e^{(T-t)} S_t (\delta q_t dt - d q_t)$$

$$X_T = e^{rT} X_0 + q_T S_T - e^{rT} q_0 S_0 + \int_0^T e^{(T-t)} S_t (\delta q_t dt - d q_t)$$

$$X_T = e^{rT} X_0 + q_T S_T - e^{rT} q_0 S_0 + \int_0^T e^{(T-t)} S_t (\delta q_t dt - d q_t) dt$$

Also
\[ X_0 = \frac{1}{(r-\delta)T} \left( e^{-\delta T} - e^{-\gamma T} \right) S_0 \]
\[ q_t = \frac{1}{(r-\delta)T} \left( e^{-\delta T} - e^{-\gamma T} \right) \]
\[ q_0 = \frac{1}{(r-\delta)T} \left( e^{-\delta T} - e^{-\gamma T} \right) \]
\[ q_T = 0 \]
\[ q_t' = \frac{1}{(r-\delta)T} \left( \delta e^{-\delta(T-t)} - r e^{-\gamma(T-t)} \right) \]

Substituting (3.158) into the expression of \( X_T \) gives

\[ X_T = e^{rT} X_0 + q T S_T - e^{rT} q_0 S_0 + \int_0^T e^{r(T-t)} S_t \left( \delta q_t - q_t' \right) dt \]
\[ = e^{rT} \frac{1}{(r-\delta)T} \left( e^{-\delta T} - e^{-\gamma T} \right) S_0 - e^{rT} \frac{1}{(r-\delta)T} \left( e^{-\delta T} - e^{-\gamma T} \right) S_0 + \int_0^T e^{r(T-t)} S_t \left( \delta q_t - q_t' \right) dt \]
\[ = \int_0^T e^{r(T-t)} S_t \left( \frac{1}{(r-\delta)T} \left( \delta e^{-\delta(T-t)} - \delta e^{-\gamma(T-t)} \right) - \frac{1}{(r-\delta)T} \left( \delta e^{-\delta(T-t)} - r e^{-\gamma(T-t)} \right) \right) dt \]
\[ = \int_0^T e^{r(T-t)} S_t \left( \frac{1}{(r-\delta)T} \left( r - \delta \right) e^{-\gamma(T-t)} \right) dt \]
\[ = \frac{1}{T} \int_0^T S_t \, dt \]
\[ = S_T \]

Thus, the final account value is exactly the payoff of the Asian forward at maturity. It implies that the Asian forward price at time 0 must have the same value as the initial wealth, that is

\[ e^{-rT} E[S_T] = e^{-rT} E[X_T] \]
\[ P_{\text{Asian forward}} = X_0 \]
\[ P_{\text{Asian forward}} = q_0 S_0 \]

The above analysis of the Asian forward is model independent because it does not require the specification of the stock dynamics. The analysis will be the same for alternative dynamics such as stochastic volatility or jumps.

Večer then replicated the Asian option by only adjusting the initial wealth in the analysis of the Asian forward. The payoff of an Asian call option is \((S_T - K)^+\) while the payoff of an Asian put option is \((K - S_T)^+\). According to the Asian put-call parity, we have

\[ C(t) + e^{-rT} K = P(t) + e^{-rT} E[S_T] \]
\[ e^{-rT} E[S_T - K] + e^{-rT} K = e^{-rT} E[(S_T - K)^+] + e^{-rT} E[S_T] \]
\[ e^{-rT} E[(S_T - K)^+] - e^{-rT} E[(K - S_T)^+] = e^{-rT} E[S_T] - e^{-rT} K \]
\[ e^{-rT} E[S_T - K] = q_0 S_0 = e^{-rT} K \]

In order to replicate an Asian call option as an option on a traded account, we hold \( q_t \) shares of the stock
at time $t$.

$$ q_t = \frac{1}{(r-\delta)^T} \left( e^{-\delta(T-t)} - e^{-r(T-t)} \right) \tag{3.162} $$

Start with the initial wealth

$$ X_0 = q_0 S_0 - e^{-rT} K $$

$$ = \frac{1}{(r-\delta)^T} \left( e^{-\delta T} - e^{-rT} \right) S_0 - e^{-rT} K \tag{3.163} $$

and let the wealth evolves according to the self-financing strategy

$$ dX_t = q_t dS_t + r(X_t - q_tS_t)dt + \delta q_t S_t dt \tag{3.164} $$

Eventually, the final wealth is

$$ X_T = \frac{1}{T} \int_{0}^{T} S_t dt - K $$

$$ = S_T - K \tag{3.165} $$

Then, the payoff of an Asian call option is $(X_T, 0)^+$. In order to reduce the dimensionality of the problem, Večeř made a change of the variable

$$ Z_t = g(t, S_t, X_t) = \frac{X_t}{e^{rS_t}} \tag{3.166} $$

Applying the Ito's formula, we have

$$ dZ_t = $$

$$ g_t dt + (g_{S_t} dS_t + g_{X_t} dX_t) + \frac{1}{2} \left[ g_{S_t S_t} (dS_t)^2 + g_{S_t X_t} dS_t dX_t + g_{X_t X_t} dX_t dX_t + g_{X_t} (dX_t)^2 \right] \tag{3.167} $$

Calculate each term in (3.167) respectively

$$ g_t = -\delta e^{-\delta t} S_t^{-1} X_t = -\delta Z_t $$

$$ g_{S_t} = -e^{-\delta t} S_t^{-2} X_t = -Z_t S_t^{-1} $$

$$ g_{X_t} = e^{-\delta t} S_t^{-1} = Z_t X_t^{-1} $$

$$ g_{S_t S_t} = 2 e^{-\delta t} S_t^{-3} X_t = 2 Z_t S_t^{-2} $$

$$ g_{S_t X_t} = g_{X_t S_t} = -e^{-\delta t} S_t^{-2} $$

$$ g_{X_t X_t} = 0 $$

$$ (dS_t)^2 = \sigma^2 S_t^2 dt $$

$$ dS_t dX_t = dX_t dS_t = q_t (dS_t)^2 + r(X_t - q_t S_t) dt dS_t + \delta q_t S_t dt dS_t = q_t \sigma^2 S_t^2 dt $$

Then
$$dZ_i = -\delta Z_i \, dt + \left( -Z_i, S_i^{-1} \, dS_i + Z_i, X_i^{-1} \, dX_i \right) + \frac{1}{2} \left[ 2 Z_i \sigma^2 \, dt - 2 e^{-\delta t} q_i \sigma^2 \, dt \right]$$

$$= \left( Z_i - e^{-\delta t} q_i \right) \sigma^2 \, dt - \left( \delta Z_i \, dt + Z_i \frac{\partial u}{\partial X_i} - Z_i \frac{\partial u}{\partial X_i} \right)$$

$$= \left( Z_i - e^{-\delta t} q_i \right) \sigma^2 \, dt - \left( \delta Z_i \, dt + (r - \delta) Z_i \, dt + \sigma Z_i \, dW_i - e^{-\delta t} \frac{\partial X_i}{\partial X_i} \right)$$

$$= \left( Z_i - e^{-\delta t} q_i \right) \sigma^2 \, dt - \left( r Z_i \, dt + \sigma Z_i \, dW_i - e^{-\delta t} \left( q_i, (r - \delta) \, dt + q_i \sigma \, dW_i + r \frac{X_i}{S_i} \, dt - q_i \sigma \, dW_i + q_i \, dt \right) \right)$$

$$= \left( Z_i - e^{-\delta t} q_i \right) \sigma^2 \, dt - \left( \delta Z_i \, dt + \sigma Z_i \, dW_i - e^{-\delta t} \left( q_i, (r - \delta) \, dt + q_i \sigma \, dW_i + r \frac{X_i}{S_i} \, dt - q_i \sigma \, dW_i + q_i \, dt \right) \right)$$

$$= \left( Z_i - e^{-\delta t} q_i \right) \sigma^2 \, dt - \left( \sigma Z_i \, dW_i - e^{-\delta t} q_i \sigma \, dW_i \right)$$

$$= \left( Z_i - e^{-\delta t} q_i \right) \sigma^2 \, dt - \left( Z_i - e^{-\delta t} q_i \right) \sigma \, dW_i$$

$$= -Z_i \, e^{-\delta t} q_i \sigma \, dW_i$$

where $\tilde{W}_i = -\sigma t + W_i$ is a Brownian motion independent of $W_i$. Thus, the price of an Asian call option can be written as

$$V(0, S_0, K) = e^{-rT} E[(X_T)^+]$$

$$= e^{-rT} \tilde{E}\left[ (\delta T S_T Z_T)^+ \right]$$

$$= e^{-\left(\delta T - r T\right) S_T} \tilde{E}[(Z_T)^+]$$

(3.170)

Introduce a new function

$$u(0, Z_0) = \tilde{E}[(Z_T)^+]$$

(3.171)

where

$$Z_0 = \frac{X_0}{e^\delta S_0} = \frac{1}{\left( r - \delta \right) T} \left( e^{-\delta T} - e^{-r T}\right) - e^{-r T} \frac{K}{S_0}$$

(3.172)

Then the price of the Asian call option is

$$V(0, S_0, K) = S_0 \, u(0, Z_0)$$

(3.173)

The function $u$ satisfies the following PDE

$$u_t + \frac{1}{2} \left( z - e^{-\delta t} q_i \right)^2 \sigma^2 u_{zz} = 0$$

$$u(T, z) = \left( z \right)^+$$

(3.174)

This PDE can be solved numerically in Wolfram Mathematica.

### 3.6. Spectral series expansion

Linetsky (2004, p.857) exploits “an identity in law between the integral of geometric Brownian motion over a finite time interval $[0, t]$ and the state at time $t$ of a one-dimensional diffusion process with affine drift and linear diffusion” and develops spectral expansions for the values of continuous arithmetic Asian puts. Asian call prices can be computed from the put-call parity.
Under the risk-neutral probability measure, the stock price \( S_t \) at time \( t \) is assumed to follow a geometric Brownian motion and has the following expression: for \( t \geq 0 \)

\[
S_t = S_0 e^{\left(r - \frac{1}{2} \sigma^2\right)t + \sigma W_t}
\]  
(3.175)

where \( r \) is the constant risk-free interest rate, \( \delta \) is the constant dividend yield, \( \sigma \) is the constant volatility, and \( W_t \) is a standard Brownian motion. The continuous arithmetic average of the stock prices over the time interval \([0, t]\) is defined by

\[
A_t = \frac{1}{t} \int_0^t S_u \, du
\]  
(3.176)

For a newly written Asian option with the strike price \( K \) and time to maturity \( T \), the payoff of an Asian call at maturity is \((A_T - K)^+\) and the price of the Asian call at \( t = 0 \) is \( e^{-rT} E[(A_T - K)^+] \), whereas the payoff of an Asian put at maturity is \((K - A_T)^+\) and the price of the Asian put at \( t = 0 \) is \( e^{-rT} E[(K - A_T)^+] \). The prices of the Asian call and put with the same underlying stock, strike price and time to maturity are related by the put-call parity (Geman and Yor, 1993; Dufresne, 2000 cited in Linetsky, 2004)

\[
e^{-rT} E[(A_T - K)^+] = e^{-rT} E[(K - A_T)^+] + \begin{cases} 
\frac{e^{rT} - e^{-rT}}{\sigma^2 T} S_0 - e^{-rT} K, & r \neq \delta \\
e^{-rT} (S_0 - K), & r = \delta
\end{cases}
\]  
(3.177)

Linetsky (2004) follows Geman and Yor (1993) and expresses the value of an Asian put as

\[
e^{-rT} E[(K - A_T)^+] = e^{-rT} \left( \frac{h}{\sigma^2 T} \right) P^{(y)}(h, q)
\]  
(3.178)

where the normalized interest rate \( v \), the normalized time to maturity \( h \), and the normalized strike price \( q \) are

\[
v = \frac{2(r - \delta)}{\sigma^2} - 1
\]

\[
h = \frac{\sigma^2 T}{4}
\]

\[
q = \frac{hK}{S_0}
\]  
(3.179)

\( P^{(y)}(h, q) \) is the normalized value of an Asian put:

\[
P^{(y)}(h, q) = E\left[\left(q - A_h^{(y)}\right)^+\right]
\]  
(3.180)

where \( A_h^{(y)} \) is called the exponential functional of Brownian motion (Yor, 2001 cited in Linetsky, 2004):

\[
A_h^{(y)} = \int_0^h \exp\left[2(W_s + v s)\right] \, ds
\]  
(3.181)
Using an identity in law between $A_t^{(q)}$ and the state at time $t$ of a one-dimensional diffusion process, Linetsky develops spectral expansions for Asian puts without resorting to Laplace transforms. Given Asian put prices, Asian call prices can be obtained via the put-call parity.

Let $P_{b}^{(q)}(q, h)$ be the spectral series expansion for $P_{b}^{(q)}(q, h)$ which is the Asian put price mentioned above. The expression for $P_{b}^{(q)}(q, h)$ involves several special functions: the first Whittaker function $M_{k,n}(z)$ and the second Whittaker function.

The first Whittaker function $M_{k,n}(z)$ is defined as

$$M_{k,n}(z) = e^{-z/2} z^{m+1/2} \, _1F_1 \left(m-k + \frac{1}{2}; 1 + 2m; z \right)$$

(3.182)

where $\, _1F_1 \left(a; b; z \right)$ is the Kummer confluent hypergeometric function defined as

$$\, _1F_1 \left(a; b; z \right) = \sum_{k=0}^{\infty} \frac{(a)_k}{(b)_k} \frac{z^k}{k!}$$

(3.183)

where $(a)_k$ is the Pochhammer symbol with the definition $(a)_0 = \Gamma(a + 1) / \Gamma(a)$.

The second Whittaker function $W_{k,n}(z)$ is defined as

$$W_{k,n}(z) = e^{-z/2} z^{m+1/2} U(m-k + \frac{1}{2}; 1 + 2m; z)$$

(3.184)

where $U(a, b, z)$ is the confluent hypergeometric function defined as

$$U(a, b, z) = \frac{1}{\Gamma(a)} \int_0^\infty e^{-zt} t^{a-1}(1+t)^{b-a-1} \, dt$$

(3.185)

Let $\nu \in \mathbb{R}$, $h > 0$, $q > 0$ and $b > q$. The function $P_{b}^{(q)}(q, h)$ is given by the following series:

$$P_{b}^{(q)}(q, h) = \sum_{n=0}^{\infty} e^{-z/2} z^{n+1/2} \frac{p, h}{4} \left(q^{n+1}; \frac{1}{4}, \frac{1}{4} \right) \left(q^{-n}; \frac{1}{4}, \frac{1}{4} \right) \left(q^{-n+1}; \frac{1}{4}, \frac{1}{4} \right) \left(q^{-n+1}; \frac{1}{4}, \frac{1}{4} \right) \left(q^{-n+1}; \frac{1}{4}, \frac{1}{4} \right) \left(q^{-n+1}; \frac{1}{4}, \frac{1}{4} \right) \left(q^{-n+1}; \frac{1}{4}, \frac{1}{4} \right) \left(q^{-n+1}; \frac{1}{4}, \frac{1}{4} \right) \left(q^{-n+1}; \frac{1}{4}, \frac{1}{4} \right) \left(q^{-n+1}; \frac{1}{4}, \frac{1}{4} \right) \left(q^{-n+1}; \frac{1}{4}, \frac{1}{4} \right)$$

(3.186)

where $0 < p_{1,h} < p_{2,h} < \cdots < p_{n,h}$ with $p_{n,h} \to \infty$ as $n \to \infty$ are the positive simple zeros of the equation

$$W_{\frac{1}{2}} ^{\nu} \left(1, -\frac{1}{2}, -\frac{1}{2} \right) p_{n,h} = 0$$

(3.187)

$\xi_{n,h}^{(q)}$ is defined as

$$\xi_{n,h}^{(q)} = \left. \frac{d}{dp} W_{\frac{1}{2}} ^{\nu} \left(1, -\frac{1}{2}, -\frac{1}{2} \right) p_{n,h} \right|_{p_{n,h}}$$

(3.188)
For \( \nu \geq 0 \), \( N_\nu(b) = 0 \) and we have \( \sum_{n=1}^{\nu} = 0 \). For \( \nu < 0 \), \( N_\nu(b) \leq \lfloor |\nu|/2 \rfloor + 1 \) where \([x]\) denotes the integer part of \( x \), is the total number of roots of the equation

\[
W_{\frac{1}{2}}\left( 1 + \nu \right)(\frac{1}{2}) = 0
\]  
(3.189)

in the interval \( s \in [0, |\nu|] \). Let \( 0 \leq s_{N_\nu(b),b} < \cdots < s_{1,b} < |\nu| \) be the roots of the above equation ordered in descending order.

\( \eta_{n,b}^{(v)} \) is defined as

\[
\eta_{n,b}^{(v)} = -\frac{\partial}{\partial s} W_{\frac{1}{2}}\left( 1 + \nu \right)(\frac{1}{2}) \bigg|_{s=s_{n,b}}
\]  
(3.190)

For the purpose of numerical computations, we need to find the estimates of the roots \( p_{n,b} \) and \( s_{n,b} \) before we solve the equations (3.187) and (3.189) for the precise numerical values.

Let \( \tilde{p}_{n,b} \) and \( \tilde{s}_{n,b} \) be the estimates of \( p_{n,b} \) and \( s_{n,b} \) respectively. The estimates \( \tilde{p}_{n,b} \) can be obtained by solving the equation

\[
\tilde{p}_{n,b} \left( \ln(4b \tilde{p}_{n,b}) - 1 \right) = 2\pi \left( n + \frac{\nu}{4} - \frac{1}{2} \right)
\]  
(3.191)

For \( \nu < 0 \) and large enough \( b \), \( N_\nu(b) = \lfloor |\nu|/2 \rfloor + 1 \) and \( \tilde{s}_{n,b} = (|\nu| - 2n + 2) \).

### 3.7. Constructive complex analysis

German and Yor (1993) shows

\[
C_t = \frac{e^{-r T}}{T - t} \frac{4 S_0}{a^2} C^{(v)}(h, q)
\]  
(3.192)

where

\[
\nu = \frac{2(r - \delta)}{a^2} - 1
\]
\[
h = \frac{a^2}{4} (T - t)
\]
\[
q = \frac{a^2}{4 S_0} \left[ K(T - t_0) - \int_{t_0}^{T} ds \, S(u) \right]
\]  
(3.193)

The valuation of an Asian option is then reduced to compute the normalized value \( C^{(v)}(h, q) \).

Schröder (2008) develops a new approach to value Asian options based on the constructive complex analysis. He first represents \( C^{(v)}(h, q) \), \( q > 0 \) by integrals. As a second step, he computes these integrals by series and asymptotic expansions. He also obtains explicit error estimates for these expansions. Thus,
by combining the approximation errors, we can get the error estimates of \( C^{\nu}(h, q) \) and \( C_r \).

### 3.7.1. Reductions

The normalized value \( C^{\nu}(h, q) \) is represented as the weighted sum of the total real part contribution \( R \) and the total imaginary part contribution \( I \) by reductions

\[
C^{\nu}(h, q) = \gamma_{\nu, q} (R + I)
\]

where

\[
\gamma_{\nu, q} = \frac{1}{2\pi} \frac{\Gamma(\nu+4)}{\Gamma(\nu+1)} \exp\left(-\frac{1}{2q}\right) (2q)^{\frac{\nu}{2}+1} \tag{3.195}
\]

### 3.7.2. Series and Asymptotic Expansions

We set the following constants

\[
\mu = \nu + 4 \\
\alpha = \frac{1}{\sqrt{2h}} \\
\beta = \frac{1}{\sqrt{2q}} \\
A = 0
\]

**Computing \( R \).** The total real part contribution \( R \) is given by

\[
R = \frac{1}{2} \left\{ \exp\left[ 2h(\nu + 2) \bar{R}(\nu + 2) - \bar{R}(\nu) \right] \right. \\
\bar{R}(a) = 2\pi \beta^2 \frac{\Gamma(a+\mu/2)}{\Gamma(a)} \Phi\left( \frac{\mu}{2} , a + 1 ; \beta^2 \right) - 4 K(a) \tag{3.197}
\]

where \( \Phi \) is the Kummer confluent hypergeometric function. \( K(a) \) is the error term and bounded by

\[
|K(a)| \leq \frac{\Gamma(\mu/2)}{\Gamma(\mu)} \frac{1-\exp(-a)}{a} + D(a) \tag{3.198}
\]

where

\[
D(a) = \left( \frac{\sqrt{2}/\beta}{a+\mu} \right)^2 \sum_{n=0}^{M} \frac{(-M)_k \exp[-i(\mu+\beta+2n)B]}{(\beta)^{n+1}} \tag{3.199}
\]

for any \( B > 0, M = (\mu - a) / 2 - 1, \) and \( P = (\mu + a) / 2 + 1 \). While, \( (\lambda)_k \) is the Pochhammer symbol defined by, for any complex \( \lambda \) and any integer \( k \geq 0, \)

\[
(\lambda)_k = \frac{\Gamma(\lambda+k)}{\Gamma(\lambda)} \tag{3.200}
\]

**Computing \( I \).** The total imaginary part contribution \( I \) is given by

\[
I = \exp[2h(\nu + 1)] \left[ I_{\nu+2, \nu+2}(0, \infty) + I_{-\nu-2, -\nu-2}(0, \infty) \right] - \left[ I_{\nu, \nu}(0, \infty) + I_{-\nu, -\nu}(0, \infty) \right] \tag{3.201}
\]
where \( I_{a,c}(0, \infty) \) is decomposed as, for any \( B > 0 \),

\[
I_{a,c}(0, \infty) = I_{a,c}(0, B) + I_{a,c}(B, \infty) \tag{3.202}
\]

By varying \( B \), \( I_{a,c}(0, B) \) can be computed up to an error of \( \varepsilon > 0 \) by an asymptotic expansion, while \( I_{a,c}(B, \infty) \) can be computed up to an error of \( \varepsilon \) by a series. Thus, \( I_{a,c}(0, \infty) \) can be computed up to an error of 2 \( \varepsilon \).

**Computing \( I_{a,c}(B, \infty) \).** For any integer \( N_a \geq 0 \), \( I_{a,c}(B, \infty) \) has the following asymptotic expansion

\[
I_{a,c}(B, \infty) = \sum_{n=0}^{N_a-1} (-1)^n \frac{b_n(\mu)}{\beta^{n+\nu}} \text{Im}\{\exp(i \frac{\pi}{2} (\mu + a)) A_{1,2+n+\nu}(a)\} + \rho_{N_a} \tag{3.203}
\]

where \( \gamma = \frac{c}{2a} + i \frac{\pi}{2} a \) and

\[
b_n(\mu) = \frac{(-1)^{n} (\mu)^{2n}}{n!} \tag{3.204}
\]

The terms \( A_{\delta, \lambda}(a) \), for any complex \( \delta \) and \( \lambda \), are series in the iterated weighted error functions \( \text{IEc} \).

\[
A_{\delta, \lambda}(a) = \sum_{k=0}^{K_0} (-1)^k \left( \frac{\lambda}{k} \right) \text{IEc}_2(\lambda - a + 2k) + R_{\delta, \lambda, K_0+1} \tag{3.205}
\]

where \( R_{\delta, \lambda, K_0+1} \) are the remainder terms. With regard to \( \text{IEc} \), for \( z = 0 \)

\[
\text{IEc}_2(0) = \frac{1}{\sqrt{\pi}} \exp\left(-\left(\frac{\lambda}{2}\right)^2\right) - \exp\left(\left(\frac{\lambda}{2}\right)^2\right) \text{Erfc}(\lambda) \tag{3.206}
\]

and for any complex \( z \) different from zero

\[
\text{IEc}_2(z) = \frac{1}{z} \left[ \exp(-z) \text{Erfc}(\alpha B + \delta) - \exp\left(\left(\frac{z}{\alpha} + \delta\right)^2 - \delta^2\right) \text{Erfc}(\alpha B + \delta + \frac{z}{\alpha}) \right] \tag{3.207}
\]

The error estimate \( \rho_{N_a} \) of \( I_{a,c}(B, \infty) \) is

\[
\rho_{N_a} \leq \exp\left[\frac{(\alpha \pi)^2}{4} \frac{\Gamma(\text{Re}(\mu))}{\Gamma(\mu)} |b_{N_a}(\text{Re}(\mu))| \left(\frac{\sqrt{\pi}}{\beta}\right)^{2N_a+\text{Re}(\mu)} \text{IEc}_2(2N_a + \text{Re}(\mu) - \text{Re}(a)) \right] \tag{3.208}
\]

**Computing \( I_{a,c}(0, B) \).** For any integer \( N_a \geq 0 \), \( I_{a,c}(A, B) \) has an absolutely convergent series.

\[
I_{a,c}(A, B) = \sum_{n=0}^{N_a-1} (-1)^n \frac{b_n(\mu)}{\beta^{n+\nu}} \sum_{k=0}^{[n/2]} b_{n,k} \left( \omega_{a,n-k-2}^{+} + \omega_{a,n-k}^{+} \right) + \rho_{N_a} \tag{3.209}
\]

where \( \gamma = \frac{c}{2a} + i \frac{\pi}{2} a \), \([x]\) denotes the integer part of \( x \), and for any complex \( \mu \)

\[
a_n(\mu) = 2^{-\mu} \frac{\Gamma(1/2)}{\Gamma(1+\nu/2)} \left( \frac{(\mu)^2}{(2\beta)^2} \right)^{n/2} \frac{1}{(n/2)!}, \quad \text{n is even} \tag{3.210}
\]

\[
a_n(\mu) = 2^{-\mu} \frac{\Gamma(-1/2)}{\Gamma(\mu/2)} \left( \frac{(\mu)^2}{(2\beta)^2} \right)^{n/2} \frac{1}{(n/2)!}, \quad \text{n is odd} \tag{3.210}
\]
If $\mu$ is a positive integer, then

$$a_{n}(\mu) = \frac{1}{2} \frac{\Gamma(n+\mu/2)}{\Gamma(\mu)} \frac{(-2)^{\mu}}{n!} \tag{3.211}$$

The coefficients $\beta_{n,k}$ for integers $n \geq k \geq 0$ are given by

$$\beta_{n,k} = \frac{1}{2^k} \binom{n}{k} \tag{3.212}$$

If $n$ is even and $k = n/2$, then

$$\beta_{n,n/2} = \frac{1}{2^{n+1}} \binom{n}{n/2} \tag{3.213}$$

The terms $\omega_{r,\eta}$ for any complex $\eta$ are given by

$$\omega_{r,\eta} = \text{Im}(\exp(i \frac{r}{2} \eta) \text{IE}_z(-\eta)) \tag{3.214}$$

where the iterated weighted error functions IE are given by, for $z = 0$,

$$\text{IE}_d(0) = \frac{1}{d} \left[ (\alpha B + \delta) \text{Erfc}(\alpha B + \delta) - (\alpha A + \delta) \text{Erfc}(\alpha A + \delta) 
+ \frac{1}{\sqrt{2\pi}} \left( \exp(-(\alpha A + \delta)^2) - \exp(-(\alpha B + \delta)^2) \right) \right] \tag{3.215}$$

and for any complex $z$ different from zero,

$$\text{IE}_d(z) = \frac{1}{z} \left[ \exp(-z A) \text{Erfc}(\alpha A + \delta) - \exp(-z B) \text{Erfc}(\alpha B + \delta) 
+ \exp((\delta + \frac{z}{2a})^2 - \delta^2) \left( \text{Erfc}(\frac{z}{2a} + \delta + \alpha B) - \text{Erfc}(\frac{z}{2a} + \delta + \alpha A) \right) \right] \tag{3.216}$$

The error estimate $\rho_{\text{NS}}$ of $I_{a,c}(A, B)$ is

$$\rho_{\text{NS}} \leq \exp \left( \frac{(a \eta)^2}{4} \right) |a_{N}(\text{Re}(\mu))| \left( \frac{\beta}{a} \right)^{N} \text{IE}_z\left(\frac{-\text{Re}(\mu) - N_i}{2a}\right) \tag{3.217}$$

### 3.8. Turnbull and Wakeman’s approximation

Turnbull and Wakeman (1991) approximate the distribution of the arithmetic average by matching the first two moments to the lognormal distribution. The Turnbull and Wakeman approximation is an analytical formula which allows to price continuous arithmetic Asian options using the generalized Black-Scholes formula with adjusted mean and adjusted variance. Let $S$ be the stock price at time zero, $K$ the strike price, $r$ the risk-free interest rate, and $\delta$ the dividend yield. The Turnbull and Wakeman approximation is (see Haug, 2006, pp.186-187)

$$P_{\text{Asian call}} \approx S e^{(b-c)r} T N(d_1) - K e^{-r T} N(d_2) \tag{3.218}$$

$$P_{\text{Asian put}} \approx K e^{-r T} N(d_2) - S e^{(b-c)r} T N(d_1) \tag{3.219}$$
with

\[ d_1 = \frac{\ln(S(0)/(K + \frac{1}{2} \sigma^2) T)}{\sigma \sqrt{T}}, \quad d_2 = d_1 - \sigma \sqrt{T} \]  

(3.220)

where the cost-of-carry of the average \( b_A \) and the volatility of the average \( \sigma_A \) are given by

\[ b_A = \frac{\ln(M)}{T} \]  

(3.221)

\[ \sigma_A = \sqrt{\frac{\ln(M^2)}{T} - 2b_A} \]  

(3.222)

The exact first two moments of the continuous arithmetic average are:

for \( r \neq \delta \),

\[ M_1 = \frac{e^{r-h}T-1}{(r-h)T} \]  

(3.223)

\[ M_2 = \frac{2e^{2(r-h)\sigma^2}T}{(r-h)\sigma^2 T} + \frac{e^{r-h}T}{(r-h)T^2} - \frac{e^{r-h}T}{(r-h)\sigma^2 T} \]  

(3.224)

and for \( r = \delta \),

\[ M_1 = 1 \]  

(3.225)

\[ M_2 = \frac{2e^{2\sigma^2} - 2(1 + \sigma^2 T)}{\sigma^2 T} \]  

(3.226)

- **3.9. Milevsky and Posner’s reciprocal gamma approximation**

Milevsky and Posner (1998) approximate the value of an arithmetic Asian option by matching the first two risk-neutral moments of the average to the reciprocal gamma distribution.

Let \( X \) be the gamma distributed random variable. Then the random variable \( Y = 1 / X \) is the reciprocal gamma distributed. The gamma distribution with shape parameter \( \alpha \) and scale parameter \( \beta \) has the probability density function (pdf), for \( x > 0 \),

\[ g(x \mid \alpha, \beta) = \frac{x^{\alpha-1} e^{-x/\beta}}{\Gamma(\alpha)} \]  

(3.227)

The cumulative distribution function \( G_R(x \mid \alpha, \beta) \) and probability density function \( g_R(x \mid \alpha, \beta) \) of \( Y \) are related to the cumulative distribution function \( G(x \mid \alpha, \beta) \) and probability density function \( g(x \mid \alpha, \beta) \) of \( X \)

\[ G_R(y \mid \alpha, \beta) = 1 - G(1 / y \mid \alpha, \beta) \]  

(3.228)

\[ g_R(y \mid \alpha, \beta) = \frac{g(1/y \mid \alpha, \beta)}{y^2} \]  

(3.229)

Therefore, \( G(x \mid \alpha, \beta) \) can be used in the pricing formula instead of \( G_R(x \mid \alpha, \beta) \). Let \( S \) be the stock price
at time zero, $K$ the strike price, $r$ the risk-free interest rate, and $\delta$ the dividend yield. The value of a continuous arithmetic Asian option is approximated by

$$P_{\text{Asian call}} \approx \frac{e^{-\delta T} - e^{-\gamma T}}{(r-\delta)T} S G(\frac{1}{K} | \alpha - 1, \beta) - e^{-\gamma T} K G(\frac{1}{K} | \alpha, \beta)$$

(3.230)

where

$$\alpha = \frac{2 M_2 - M_1^2}{M_2 - M_1^2}, \quad \beta = \frac{M_2 - M_1^2}{M_2 M_1}$$

(3.231)

The first two moments of the continuous arithmetic average in Milevsky and Posner approximation are

for $r \neq \delta$,

$$M_1 = S \frac{e^{(r-\delta)T} - 1}{(r-\delta)T}$$

(3.232)

$$M_2 = \frac{2 S^2 e^{[2(r-\delta)\sigma^2]T}}{(r-\delta+\sigma^2)(2(r-\delta+\sigma^2)T^2)} + \frac{2 S^2}{(r-\delta)T^2} \left( \frac{1}{2(r-\delta+\sigma^2)} - \frac{e^{(r-\delta)T}}{r-\delta+\sigma^2} \right)$$

(3.233)

and for $r = \delta$,

$$M_1 = S$$

(3.234)

$$M_2 = \frac{2 S^2 e^{[2\sigma^2]T} - 2 S^2 [1+\sigma^2] T}{\sigma^4 T^2}$$

(3.235)
4. Numerical Results

4.1. Experiment design

We conduct a series of numerical experiments to investigate the efficiency and accuracy of various methods used to price an Asian option. Numerical inversion algorithms include Abate and Whitt’s (1995) Euler method (Euler); Abate and Whitt’s (1995) Post-Widder method (PW); Shaw’s (1998) Bromwich integration; Abate and Valkó’s (2004) Gaver-Wynn-Rho algorithm (GWR); Abate and Valkó’s (2004) fixed Talbot method (FT); Abate and Whitt’s (2006) unified Gaver-Stehfest algorithm (UniG); Abate and Whitt’s (2006) unified Euler algorithm (UniE); Abate and Whitt’s (2006) unified Talbot algorithm (UniT); Abate, Choudhury and Whitt’s (1996) Laguerre method. Other methods cover Linetsky’s (2004) spectral series expansion, Schröder’s (2008) constructive complex analysis, Večef’s (2002) PDE method, Shaw’s (2002) asymptotic expansion, Turnbull and Wakeman’s (1991) approximation, Milevsky and Posner’s (1998) reciprocal gamma approximation, and Monte Carlo simulation with control variate. We compute the prices of nineteen cases of Asian options in Table 4.1 with every method and draw comparisons between different methods. The results can also be compared with the literature. For example, all cases are used in Craddock, Heath and Platen (2000) which has become the benchmark in Asian option pricing experimental design, and first seven cases are used in Fu, Madan and Wang (1999) and Linetsky (2004). We write Mathematica code for all methods except Shaw’s asymptotic expansion (where we use Shaw’s own code).
By experiment, there are three types of errors that could affect the accuracy of the algorithm, i.e., truncation error, discretization error, and round-off error. In practical implementation, in order to achieve desired accuracy we have to control these errors. The truncation error and discretization error are associated with the parameter settings of the algorithm. While, the round-off error is determined by the computing precision, also called working precision (wp).

As we know, *Mathematica* has a arbitrary-precision, or multi-precision, computing environment.

However, an approximate real number you enter is treated by *Mathematica* as either a machine-precision number (also called machine number) or an arbitrary-precision number depending on the number of the digits. For example, A number such as 1.2 will be treated as a machine number since the digits entered are less than the typical value of machine precision, i.e. 16. Machine number always contain the same number of digits, and has no information on their precision. While, arbitrary-precision number contains any number of digits, and maintain information on their precision. Notice that precision is defined as the effective number of digits. Given the fact that an approximate real number always has some uncertainty in its value, the precision of a number can indicate the number of digits that has no uncertainty.

To implement in the multi-precision computing environment, we must change machine numbers to arbitrary-precision numbers. This can be done easily by using built-in Mathematica function `SetPrecision[expr, p]`. For example, `SetPrecision[1.2, 20]` sets number 1.2 to have precision of 20 digits.

---

**Table 4.1. Nineteen cases of Asian options with normalized parameters**

<table>
<thead>
<tr>
<th>Case</th>
<th>S</th>
<th>K</th>
<th>r</th>
<th>σ</th>
<th>T</th>
<th>ν</th>
<th>h</th>
<th>q</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.9</td>
<td>2</td>
<td>0.05</td>
<td>0.5</td>
<td>1</td>
<td>-0.6</td>
<td>0.0625</td>
<td>0.05979</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0.05</td>
<td>0.5</td>
<td>1</td>
<td>-0.6</td>
<td>0.0625</td>
<td>0.05925</td>
</tr>
<tr>
<td>3</td>
<td>2.1</td>
<td>2</td>
<td>0.05</td>
<td>0.5</td>
<td>1</td>
<td>-0.6</td>
<td>0.0625</td>
<td>0.05925</td>
</tr>
<tr>
<td>4</td>
<td>2.2</td>
<td>2</td>
<td>0.02</td>
<td>0.1</td>
<td>1</td>
<td>-3</td>
<td>0.0025</td>
<td>0.0025</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>2</td>
<td>0.18</td>
<td>0.3</td>
<td>1</td>
<td>-3</td>
<td>0.0225</td>
<td>0.0225</td>
</tr>
<tr>
<td>6</td>
<td>2.2</td>
<td>0.0125</td>
<td>0.25</td>
<td>2</td>
<td>-0.6</td>
<td>0.03125</td>
<td>0.03125</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>2</td>
<td>0.05</td>
<td>0.5</td>
<td>2</td>
<td>-0.6</td>
<td>0.125</td>
<td>0.125</td>
</tr>
<tr>
<td>8</td>
<td>17</td>
<td>16</td>
<td>0.06</td>
<td>0.3</td>
<td>2.5</td>
<td>0.3333</td>
<td>0.05625</td>
<td>0.05294</td>
</tr>
<tr>
<td>9</td>
<td>17</td>
<td>17</td>
<td>0.06</td>
<td>0.3</td>
<td>2.5</td>
<td>0.3333</td>
<td>0.05625</td>
<td>0.05625</td>
</tr>
<tr>
<td>10</td>
<td>17</td>
<td>18</td>
<td>0.06</td>
<td>0.3</td>
<td>2.5</td>
<td>0.3333</td>
<td>0.05625</td>
<td>0.05956</td>
</tr>
<tr>
<td>11</td>
<td>53</td>
<td>51</td>
<td>0.07</td>
<td>0.4</td>
<td>1.5</td>
<td>-0.125</td>
<td>0.06</td>
<td>0.05794</td>
</tr>
<tr>
<td>12</td>
<td>53</td>
<td>53</td>
<td>0.07</td>
<td>0.4</td>
<td>1.5</td>
<td>-0.125</td>
<td>0.06</td>
<td>0.06</td>
</tr>
<tr>
<td>13</td>
<td>53</td>
<td>55</td>
<td>0.07</td>
<td>0.4</td>
<td>1.5</td>
<td>-0.125</td>
<td>0.06</td>
<td>0.06226</td>
</tr>
<tr>
<td>14</td>
<td>29</td>
<td>27</td>
<td>0.11</td>
<td>0.15</td>
<td>0.5</td>
<td>8.7778</td>
<td>0.00281</td>
<td>0.00262</td>
</tr>
<tr>
<td>15</td>
<td>29</td>
<td>29</td>
<td>0.11</td>
<td>0.15</td>
<td>0.5</td>
<td>8.7778</td>
<td>0.00281</td>
<td>0.00281</td>
</tr>
<tr>
<td>16</td>
<td>29</td>
<td>31</td>
<td>0.11</td>
<td>0.15</td>
<td>0.5</td>
<td>8.7778</td>
<td>0.00281</td>
<td>0.00301</td>
</tr>
<tr>
<td>17</td>
<td>100</td>
<td>90</td>
<td>0.09</td>
<td>0.3</td>
<td>1</td>
<td>1</td>
<td>0.0225</td>
<td>0.02025</td>
</tr>
<tr>
<td>18</td>
<td>100</td>
<td>100</td>
<td>0.09</td>
<td>0.3</td>
<td>1</td>
<td>1</td>
<td>0.0225</td>
<td>0.0225</td>
</tr>
<tr>
<td>19</td>
<td>100</td>
<td>110</td>
<td>0.09</td>
<td>0.3</td>
<td>1</td>
<td>1</td>
<td>0.0225</td>
<td>0.02475</td>
</tr>
</tbody>
</table>

ν is normalized interest rate, h is normalized maturity and q is normalized strike price.
The secret of making multi-precision computing successful is to ensure all input arguments have been set to be arbitrary-precision numbers or combination of exact numbers and arbitrary-precision numbers. Thus, the final result yielded will have information on its precision. We can then be confident that there is no round-off error within the precision. But, If we input even one machine-precision argument, the result will lose information on its precision, and we will have no idea about the size of round-off error.

By increasing the truncation size in numerical inversion, the accuracy improves consistently provided that the discretization error and round-off error are smaller than the truncation error. Fu, Madan and Wang (1999) find that numerical inversion encounters difficulties for low volatility and short maturity. Furthermore, Craddock, Heath and Platen (2000) point out that numerical difficulties arise when the value of \( q \) is small. Hence we regard cases of \( q > 0.02 \) as normal cases and cases of \( 0.0025 \leq q < 0.02 \) as difficult cases. For normal cases, we seek an accuracy of 5, 10, and 15 significant digits respectively. For difficult cases, we aim for an accuracy of 5 significant digits. All of the tolerances we use should be acceptable to those who actually trade Asian options.

- **4.2. Reference prices of nineteen Asian option cases**

Although we do not know the true value of an actual Asian option, the theoretical value can be obtained for the continuous arithmetic Asian option. We find that all Laplace transform inversion algorithms can compute the price to any desired accuracy using multi-precision arithmetic. Therefore, we can compute the reference price for each Asian option case by comparing the results yielded from two different inversion methods. The number of significant digits to which two methods agree is used to measure accuracy.
Table 4.2. Reference prices of nineteen cases of Asian options

<table>
<thead>
<tr>
<th>Case</th>
<th>Q</th>
<th>Euler ED</th>
<th>Euler CPU</th>
<th>FT ED</th>
<th>FT CPU</th>
<th>Accu</th>
<th>Reference Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.06579</td>
<td>0.16</td>
<td>0.34</td>
<td>0.38</td>
<td>0.34</td>
<td>0.38</td>
<td>0.0193</td>
</tr>
<tr>
<td>2</td>
<td>0.0625</td>
<td>0.16</td>
<td>0.34</td>
<td>0.40</td>
<td>0.37</td>
<td>0.37</td>
<td>2.56415694933704182315275662885990</td>
</tr>
<tr>
<td>3</td>
<td>0.05952</td>
<td>0.11</td>
<td>0.84</td>
<td>0.41</td>
<td>0.37</td>
<td>0.37</td>
<td>0.3062203647943653310436982583932800762</td>
</tr>
<tr>
<td>4</td>
<td>0.0025</td>
<td>0.16</td>
<td>0.49</td>
<td>0.17</td>
<td>0.06</td>
<td>0.06</td>
<td>0.05596064154020690</td>
</tr>
<tr>
<td>5</td>
<td>0.0225</td>
<td>0.34</td>
<td>0.53</td>
<td>0.48</td>
<td>0.37</td>
<td>0.37</td>
<td>0.218387546595568024772054027584764359</td>
</tr>
<tr>
<td>6</td>
<td>0.03125</td>
<td>0.32</td>
<td>0.84</td>
<td>0.41</td>
<td>0.39</td>
<td>0.39</td>
<td>0.17268741018016633570749057861043067601</td>
</tr>
<tr>
<td>7</td>
<td>0.125</td>
<td>0.49</td>
<td>0.04</td>
<td>0.20</td>
<td>0.21</td>
<td>0.21</td>
<td>0.350095281896540234492743916798</td>
</tr>
<tr>
<td>8</td>
<td>0.05294</td>
<td>0.11</td>
<td>0.84</td>
<td>0.40</td>
<td>0.36</td>
<td>0.36</td>
<td>0.281586201560703126759064957764793782</td>
</tr>
<tr>
<td>9</td>
<td>0.05625</td>
<td>0.16</td>
<td>0.47</td>
<td>0.38</td>
<td>0.28</td>
<td>0.28</td>
<td>2.3108788872322412823172820323811490655</td>
</tr>
<tr>
<td>10</td>
<td>0.0556</td>
<td>0.16</td>
<td>0.86</td>
<td>0.42</td>
<td>0.38</td>
<td>0.38</td>
<td>1.879023661294478200965956223467068841</td>
</tr>
<tr>
<td>11</td>
<td>0.05774</td>
<td>0.30</td>
<td>0.87</td>
<td>0.42</td>
<td>0.36</td>
<td>0.36</td>
<td>7.8957951994470821343341099805591440</td>
</tr>
<tr>
<td>12</td>
<td>0.08</td>
<td>0.81</td>
<td>0.41</td>
<td>0.37</td>
<td>0.37</td>
<td>0.37</td>
<td>6.9354226319959048250853728922032140</td>
</tr>
<tr>
<td>13</td>
<td>0.0622</td>
<td>0.34</td>
<td>0.83</td>
<td>0.41</td>
<td>0.38</td>
<td>0.38</td>
<td>6.07098179038842034033621863430648646</td>
</tr>
<tr>
<td>14</td>
<td>0.00262</td>
<td>0.45</td>
<td>0.56</td>
<td>0.46</td>
<td>0.22</td>
<td>0.22</td>
<td>2.697871537555891478875</td>
</tr>
<tr>
<td>15</td>
<td>0.00281</td>
<td>0.46</td>
<td>0.63</td>
<td>0.43</td>
<td>0.20</td>
<td>0.20</td>
<td>1.1347414322206809100</td>
</tr>
<tr>
<td>16</td>
<td>0.00301</td>
<td>0.46</td>
<td>0.64</td>
<td>0.42</td>
<td>0.17</td>
<td>0.17</td>
<td>0.283532493868894175</td>
</tr>
<tr>
<td>17</td>
<td>0.02025</td>
<td>0.46</td>
<td>0.43</td>
<td>0.20</td>
<td>0.38</td>
<td>0.38</td>
<td>14.93958333573850641383863550353691471</td>
</tr>
<tr>
<td>18</td>
<td>0.0225</td>
<td>0.34</td>
<td>5.27</td>
<td>4.85</td>
<td>1.87</td>
<td>1.87</td>
<td>8.828758522806605767802336378020297</td>
</tr>
<tr>
<td>19</td>
<td>0.02475</td>
<td>0.49</td>
<td>4.76</td>
<td>4.91</td>
<td>1.72</td>
<td>1.72</td>
<td>4.69670913213636275823388491137724143</td>
</tr>
</tbody>
</table>

wp: working precision; ED: effective number of significant digits; Accu: accuracy measured by the number of significant digits; CPU: computing time in seconds.

Euler: the Euler method. FT: the fixed Talbot algorithm.
Parameter settings: Euler: \( m = 15 \), \( n = 200 \), \( \gamma = 50 \), wp = 50 vs FT: \( M = 100 \), wp = 50.

Table 4.2 shows the reference prices for nineteen Asian option cases using Euler method with the setting \( m = 15 \), \( n = 200 \), \( \gamma = 50 \), wp = 50 and FT method with the setting \( M = 100 \), wp = 50. The accuracy of each reference price is indicated in the column labeled "Accu". We use "ED" to denote the effective number of significant digits in the yielding result. The effective digits are always less than the working precision due to the round-off error. It also indicates there is no round-off error within the effective digits. The computing time in seconds is reported in column labeled "CPU". The reference prices can be used to check the accuracy of various methods for pricing an Asian option.

### 4.3. Numerical results of the Euler method

The Euler method involves the settings of three parameters: \( m \), \( n \), and \( \gamma \). The sum of \( m \) and \( n \) controls the truncation error, and \( \gamma \) controls the discretization error. We set \( \gamma = 18 \) (equivalent to \( A = 41.4 \)) so that there is no discretization error within 16-digit precision. The value of \( m \) is fixed at \( m = 15 \), and the value of \( n \) is allowed to vary. It is shown in Table 4.3 that the accuracy increases consistently with the truncation size \( n \). Using machine precision, the accuracy increases with the value of \( n \) until it reaches a limit, for example 11 digits in case 3 of Table 4.3. After that, the accuracy does not improve anymore when we continue to increase \( n \). Figure 4.1 shows the price of Asian option Case 1 converges and remain stable.
Table 4.3. Results of the Euler method using machine precision

<table>
<thead>
<tr>
<th>Case</th>
<th>PM</th>
<th>CHP</th>
<th>Reference</th>
<th>n = 14</th>
<th>n = 31</th>
<th>n = 45</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accu</td>
<td>Price</td>
<td></td>
<td>CPU</td>
<td>Price</td>
<td>CPU</td>
</tr>
<tr>
<td>1</td>
<td>0.194</td>
<td>0.194</td>
<td>38 0.1933737903</td>
<td>0.19317 0.03</td>
<td>12 0.19117 0.03</td>
<td>14 0.19317 0.03</td>
</tr>
<tr>
<td>2</td>
<td>0.247</td>
<td>0.248</td>
<td>38 0.2464156095</td>
<td>0.24642 0.03</td>
<td>12 0.24642 0.03</td>
<td>13 0.24642 0.05</td>
</tr>
<tr>
<td>3</td>
<td>0.350</td>
<td>0.348</td>
<td>37 0.3501497548</td>
<td>0.35022 0.03</td>
<td>11 0.35064 0.05</td>
<td>11 0.35064 0.03</td>
</tr>
<tr>
<td>4</td>
<td>0.0560</td>
<td>0.055</td>
<td>17 0.05538604154</td>
<td>0.05539 0.03</td>
<td>8 0.055986 0.03</td>
<td>13 0.055986 0.11</td>
</tr>
<tr>
<td>5</td>
<td>0.219</td>
<td>0.222</td>
<td>38 0.2183754666</td>
<td>0.21839 0.03</td>
<td>12 0.21839 0.50</td>
<td>12 0.21839 0.75</td>
</tr>
<tr>
<td>6</td>
<td>0.172</td>
<td>0.172</td>
<td>39 0.1722887410</td>
<td>0.17223 0.14</td>
<td>12 0.17227 0.24</td>
<td>14 0.17227 0.33</td>
</tr>
<tr>
<td>7</td>
<td>0.352</td>
<td>0.340</td>
<td>30 0.350925190</td>
<td>0.35010 0.03</td>
<td>11 0.35010 0.02</td>
<td>13 0.35010 0.03</td>
</tr>
<tr>
<td>8</td>
<td>2.808</td>
<td>36 2.815862016</td>
<td>2.8159 0.08</td>
<td>11 2.8159 0.12</td>
<td>12 2.8159 0.16</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>2.265</td>
<td>38 2.21097888</td>
<td>2.3109 0.56</td>
<td>11 2.3109 0.06</td>
<td>14 2.3109 0.06</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1.875</td>
<td>39 1.78702366</td>
<td>1.7970 0.03</td>
<td>11 1.7970 0.02</td>
<td>12 1.7970 0.05</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>7.903</td>
<td>36 7.897959199</td>
<td>7.8958 0.03</td>
<td>11 7.8958 0.05</td>
<td>12 7.8958 0.05</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>6.942</td>
<td>37 6.915422632</td>
<td>6.9254 0.03</td>
<td>11 6.9254 0.05</td>
<td>13 6.9254 0.05</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>6.077</td>
<td>38 6.070987190</td>
<td>6.0710 0.03</td>
<td>11 6.0710 0.03</td>
<td>13 6.0710 0.05</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>2.808</td>
<td>22 2.67971538</td>
<td>2.6982 0.13</td>
<td>6 2.6979 19.3</td>
<td>12 2.6979 28.5</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>1.129</td>
<td>20 1.12149462</td>
<td>1.1147 0.55</td>
<td>9 1.1147 17.8</td>
<td>12 1.1147 25.0</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>0.378</td>
<td>17 0.285324938</td>
<td>0.28532 0.81</td>
<td>10 0.28532 16.0</td>
<td>13 0.28532 24.3</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>15.058</td>
<td>38 14.88395813</td>
<td>14.884 0.31</td>
<td>10 14.884 0.58</td>
<td>12 14.884 0.97</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>8.964</td>
<td>37 8.823758224</td>
<td>8.8288 0.27</td>
<td>13 8.8288 0.48</td>
<td>12 8.8288 0.73</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>4.700</td>
<td>38 4.69709112</td>
<td>4.6967 0.22</td>
<td>13 4.6967 0.42</td>
<td>13 4.6967 0.62</td>
<td></td>
</tr>
</tbody>
</table>

wp: working precision; ED: effective number of significant digits; Accu: accuracy measured by the number of significant digits; CPU: computing time in seconds.

Euler: the Euler method.

In multi-precision, the accuracy constantly improves with \( n \) given that discretization error and round-off error are controlled. For normal cases in Table 4.4, the settings of \( n = 15 \) and wp = 20 computes results to accuracy of at least 5 digits within 0.4 CPU second, while the settings of \( n = 31 \) and wp = 20 gives at least 10 digits accuracy within 0.7 CPU second. To achieve higher accuracy of 15 digits, the working precision is set to 25 digits to eliminate the impact of the round-off error on the accuracy. The Euler method takes less than 1 second for normal cases to achieve 15-digit accuracy with \( n = 45 \). For difficult cases, the method yields less accurate result but consumes considerably more CPU time with the same settings. Table 4.5 shows the Euler method takes between 9 to 12 seconds to have 5-digit accuracy for Case 4, 14, 15 and 16 with the settings of \( n = 22 \) and wp = 20.

![Figure 4.1. The price of Asian option Case 1 computed by Euler method with machine precision](image-url)
We suggest the working precision should be set such that the result has several more effective digits than its accuracy, say 5 more digits. If the difference between the accuracy and the effective digits is less than one digit, it will imply that the round-off error may dominate. In this case, increasing the working precision usually improves the accuracy.

Table 4.4. Results of the Euler method with arbitrary precision

<table>
<thead>
<tr>
<th>Case</th>
<th>Reference</th>
<th>n = 14; wp = 20</th>
<th>n = 31; wp = 26</th>
<th>n = 45; wp = 25</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accu</td>
<td>Price</td>
<td>CPU</td>
<td>Accu</td>
</tr>
<tr>
<td>1</td>
<td>0.19317</td>
<td>0.04137</td>
<td>0.08</td>
<td>0.19317</td>
</tr>
<tr>
<td>2</td>
<td>0.24642</td>
<td>0.06224</td>
<td>0.09</td>
<td>0.24642</td>
</tr>
<tr>
<td>3</td>
<td>0.30622</td>
<td>0.09648</td>
<td>0.09</td>
<td>0.30622</td>
</tr>
<tr>
<td>4</td>
<td>0.05598</td>
<td>0.04137</td>
<td>0.08</td>
<td>0.05598</td>
</tr>
<tr>
<td>5</td>
<td>0.21899</td>
<td>0.06224</td>
<td>0.09</td>
<td>0.21899</td>
</tr>
<tr>
<td>6</td>
<td>0.17226</td>
<td>0.09648</td>
<td>0.09</td>
<td>0.17226</td>
</tr>
<tr>
<td>7</td>
<td>0.35095</td>
<td>0.04137</td>
<td>0.08</td>
<td>0.35095</td>
</tr>
<tr>
<td>8</td>
<td>2.81586</td>
<td>0.06224</td>
<td>0.09</td>
<td>2.81586</td>
</tr>
<tr>
<td>9</td>
<td>2.31087</td>
<td>0.09648</td>
<td>0.09</td>
<td>2.31087</td>
</tr>
<tr>
<td>10</td>
<td>1.89702</td>
<td>0.04137</td>
<td>0.08</td>
<td>1.89702</td>
</tr>
<tr>
<td>11</td>
<td>2.89857</td>
<td>0.06224</td>
<td>0.09</td>
<td>2.89857</td>
</tr>
<tr>
<td>12</td>
<td>6.95422</td>
<td>0.09648</td>
<td>0.09</td>
<td>6.95422</td>
</tr>
<tr>
<td>13</td>
<td>6.07098</td>
<td>0.04137</td>
<td>0.08</td>
<td>6.07098</td>
</tr>
<tr>
<td>14</td>
<td>2.69781</td>
<td>0.06224</td>
<td>0.09</td>
<td>2.69781</td>
</tr>
<tr>
<td>15</td>
<td>1.13474</td>
<td>0.09648</td>
<td>0.09</td>
<td>1.13474</td>
</tr>
<tr>
<td>16</td>
<td>0.28532</td>
<td>0.04137</td>
<td>0.08</td>
<td>0.28532</td>
</tr>
<tr>
<td>17</td>
<td>14.998</td>
<td>0.06224</td>
<td>0.09</td>
<td>14.998</td>
</tr>
<tr>
<td>18</td>
<td>8.82870</td>
<td>0.09648</td>
<td>0.09</td>
<td>8.82870</td>
</tr>
<tr>
<td>19</td>
<td>4.69670</td>
<td>0.04137</td>
<td>0.08</td>
<td>4.69670</td>
</tr>
</tbody>
</table>

wp: working precision; ED: effective number of significant digits; Accu: accuracy measured by the number of significant digits; CPU: computing time in seconds.

Euler: the Euler method.

Table 4.5. Results of the Euler method on the difficult cases with enhanced parameter settings

<table>
<thead>
<tr>
<th>Case</th>
<th>Reference</th>
<th>n = 22; wp = 20</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accu</td>
<td>Price</td>
</tr>
<tr>
<td>4</td>
<td>0.05598604154</td>
<td>0.05598604154</td>
</tr>
<tr>
<td>14</td>
<td>2.69781538462</td>
<td>2.69781538462</td>
</tr>
<tr>
<td>15</td>
<td>1.134741432</td>
<td>1.134741432</td>
</tr>
<tr>
<td>16</td>
<td>0.2853249387</td>
<td>0.2853249387</td>
</tr>
</tbody>
</table>

wp: working precision; ED: effective number of significant digits; Accu: accuracy measured by the number of significant digits; CPU: computing time in seconds.

Euler: the Euler method.

4.4. Numerical results of the Post-Widder method

The Post-Widder method involves the settings of three parameters: $j$, $m$, and $\gamma$. The product of $j$ and $m$ controls the truncation error, while $\gamma$ controls the discretization error. As in Euler method, we set $\gamma = 18$ so that no discretization error arises within 16-digit precision. Parameter $j$ is fixed at $j = 15$, and allow the value of $m$ to change. In machine precision, the accuracy increases first with $j$ and then deteriorates rapidly as in Table 4.6. This is caused by propagation by round-off error. Figure 4.2 depicts this situation
clearly where the price of the first case converges initially and keeps stable before it diverges again.

Table 4.6. Results of the Post-Widder method with machine precision

<table>
<thead>
<tr>
<th>Case</th>
<th>Reference</th>
<th>m = 12</th>
<th>m = 20</th>
<th>m = 27</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>38</td>
<td>0.193173903</td>
<td>0.193173903</td>
<td>0.193173903</td>
</tr>
<tr>
<td>2</td>
<td>37</td>
<td>0.2464156905</td>
<td>0.2464156905</td>
<td>0.2464156905</td>
</tr>
<tr>
<td>3</td>
<td>37</td>
<td>0.3062203648</td>
<td>0.3062203648</td>
<td>0.3062203648</td>
</tr>
<tr>
<td>4</td>
<td>37</td>
<td>0.05598604154</td>
<td>0.05598604154</td>
<td>0.05598604154</td>
</tr>
<tr>
<td>5</td>
<td>38</td>
<td>0.2183875466</td>
<td>0.2183875466</td>
<td>0.2183875466</td>
</tr>
<tr>
<td>6</td>
<td>39</td>
<td>0.1722687410</td>
<td>0.1722687410</td>
<td>0.1722687410</td>
</tr>
<tr>
<td>7</td>
<td>30</td>
<td>0.3500952190</td>
<td>0.3500952190</td>
<td>0.3500952190</td>
</tr>
<tr>
<td>8</td>
<td>36</td>
<td>2.815862016</td>
<td>2.815862016</td>
<td>2.815862016</td>
</tr>
<tr>
<td>9</td>
<td>38</td>
<td>2.310878887</td>
<td>2.310878887</td>
<td>2.310878887</td>
</tr>
<tr>
<td>10</td>
<td>38</td>
<td>1.879023661</td>
<td>1.879023661</td>
<td>1.879023661</td>
</tr>
<tr>
<td>11</td>
<td>38</td>
<td>1.879023661</td>
<td>1.879023661</td>
<td>1.879023661</td>
</tr>
<tr>
<td>12</td>
<td>37</td>
<td>6.935422632</td>
<td>6.935422632</td>
<td>6.935422632</td>
</tr>
<tr>
<td>13</td>
<td>38</td>
<td>6.070987190</td>
<td>6.070987190</td>
<td>6.070987190</td>
</tr>
<tr>
<td>14</td>
<td>22</td>
<td>2.697871538</td>
<td>2.697871538</td>
<td>2.697871538</td>
</tr>
<tr>
<td>15</td>
<td>20</td>
<td>1.134741432</td>
<td>1.134741432</td>
<td>1.134741432</td>
</tr>
<tr>
<td>16</td>
<td>17</td>
<td>0.2853249387</td>
<td>0.2853249387</td>
<td>0.2853249387</td>
</tr>
<tr>
<td>17</td>
<td>38</td>
<td>14.983958333</td>
<td>14.983958333</td>
<td>14.983958333</td>
</tr>
<tr>
<td>18</td>
<td>37</td>
<td>8.828758224</td>
<td>8.828758224</td>
<td>8.828758224</td>
</tr>
<tr>
<td>19</td>
<td>38</td>
<td>4.664079132</td>
<td>4.664079132</td>
<td>4.664079132</td>
</tr>
</tbody>
</table>

wp: working precision; ED: effective number of significant digits; Accu: accuracy measured by the number of significant digits; CPU: computing time in seconds.

Figure 4.2. The price of Asian option Case 1 computed by Post-Widder method with machine precision

To combat this problem, we have to use multi-precision arithmetic to reduce the round-off error. Table 4.7 shows the accuracy of the Post-Widder method increases consistently with m in suitable working precision. In fact, with the settings of m = 27 and wp = 40, the accuracy of some cases could still be improved by increasing γ as existing value of γ causes the discretization error to dominate. Table 4.8 gives the settings and CPU time for computing difficult cases to at least 5-digit accuracy. In general, the Post-Widder method requires two order of magnitude more computing time than the Euler method to get
the same accuracy.

Table 4.7. the Post-Widder method with arbitrary precision

<table>
<thead>
<tr>
<th>Case</th>
<th>Reference</th>
<th>m = 12; wp = 25</th>
<th>m = 20; wp = 35</th>
<th>m = 27; wp = 40</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accu</td>
<td>Price</td>
<td>Accu</td>
<td>Price</td>
</tr>
<tr>
<td>1</td>
<td>38</td>
<td>0.1931737903</td>
<td>15.4</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>37</td>
<td>0.2445156905</td>
<td>15.4</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>37</td>
<td>0.3062306468</td>
<td>15.4</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>37</td>
<td>0.35598604154</td>
<td>15.4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>38</td>
<td>0.2138714565</td>
<td>15.4</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>39</td>
<td>0.1726874101</td>
<td>15.4</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>39</td>
<td>0.3500952190</td>
<td>15.4</td>
<td>9</td>
</tr>
<tr>
<td>8</td>
<td>36</td>
<td>2.8156201614</td>
<td>15.4</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>38</td>
<td>2.3108788875</td>
<td>15.4</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>38</td>
<td>1.870023661</td>
<td>15.4</td>
<td>8</td>
</tr>
<tr>
<td>11</td>
<td>38</td>
<td>1.6977951189</td>
<td>15.4</td>
<td>8</td>
</tr>
<tr>
<td>12</td>
<td>38</td>
<td>1.5356265322</td>
<td>15.4</td>
<td>8</td>
</tr>
<tr>
<td>13</td>
<td>38</td>
<td>6.0709871970</td>
<td>15.4</td>
<td>8</td>
</tr>
<tr>
<td>14</td>
<td>22</td>
<td>2.6987871558</td>
<td>15.4</td>
<td>2</td>
</tr>
<tr>
<td>15</td>
<td>20</td>
<td>1.1847414372</td>
<td>15.4</td>
<td>2</td>
</tr>
<tr>
<td>16</td>
<td>17</td>
<td>0.2853249387</td>
<td>15.4</td>
<td>2</td>
</tr>
<tr>
<td>17</td>
<td>38</td>
<td>14.98395834</td>
<td>15.4</td>
<td>6</td>
</tr>
<tr>
<td>18</td>
<td>38</td>
<td>5.6087012906</td>
<td>15.4</td>
<td>5</td>
</tr>
<tr>
<td>19</td>
<td>38</td>
<td>4.6967091212</td>
<td>15.4</td>
<td>5</td>
</tr>
</tbody>
</table>

wp: working precision; ED: effective number of significant digits; Accu: accuracy measured by the number of significant digits; CPU: computing time in seconds.

PW: the Post-Widder method.

Table 4.8. Results of the Post-Widder method on the difficult cases with enhanced parameter settings

<table>
<thead>
<tr>
<th>Case</th>
<th>Reference</th>
<th>m = 26; wp = 25</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accu</td>
<td>Price</td>
</tr>
<tr>
<td>4</td>
<td>17</td>
<td>0.0598604154</td>
</tr>
<tr>
<td>14</td>
<td>22</td>
<td>2.697871538</td>
</tr>
<tr>
<td>15</td>
<td>20</td>
<td>1.134741432</td>
</tr>
<tr>
<td>16</td>
<td>17</td>
<td>0.2853249387</td>
</tr>
</tbody>
</table>

wp: working precision; ED: effective number of significant digits; Accu: accuracy measured by the number of significant digits; CPU: computing time in seconds.

PW: the Post-Widder method.

4.5. Numerical results of Bromwich integration

Bromwich integral involves the settings of two parameters \( a \) and \( x \). Parameter \( a \) is set to the smallest integer greater than \( \text{Max}[2 \gamma + 2, 0] \) so that no singularity is to the right of the contour. Parameter \( x \) controls the truncation size of the integration. The Bromwich integral uses built-in Mathematica function "NIntegrate" to invert the Laplace transform. The precision used in internal computations of "NIntegrate" is specified by the "WorkingPrecision" option. With machine precision, Bromwich integral can produce up to 11-digit accuracy as shown in Table 4.9, and the accuracy does not decay as depicted in Figure 4.3.
Table 4.9. Results of Bromwich integration with machine precision

<table>
<thead>
<tr>
<th>Case</th>
<th>CHP</th>
<th>Reference</th>
<th>Bromwich : wp = Machine Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>x = 1500</td>
<td>x = 3500</td>
</tr>
<tr>
<td></td>
<td>Accu</td>
<td>Price</td>
<td>CPU</td>
</tr>
<tr>
<td>1</td>
<td>0.194</td>
<td>38</td>
<td>0.1931737903</td>
</tr>
<tr>
<td>2</td>
<td>0.247</td>
<td>37</td>
<td>0.2464156905</td>
</tr>
<tr>
<td>3</td>
<td>0.307</td>
<td>37</td>
<td>0.3062203648</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>17</td>
<td>0.05598604154</td>
</tr>
<tr>
<td>5</td>
<td>0.219</td>
<td>38</td>
<td>0.2183875466</td>
</tr>
<tr>
<td>6</td>
<td>0.172</td>
<td>39</td>
<td>0.1722687410</td>
</tr>
<tr>
<td>7</td>
<td>0.352</td>
<td>39</td>
<td>0.3500952190</td>
</tr>
<tr>
<td>8</td>
<td>2.767</td>
<td>36</td>
<td>2.815862016</td>
</tr>
<tr>
<td>9</td>
<td>2.273</td>
<td>38</td>
<td>2.310878887</td>
</tr>
<tr>
<td>10</td>
<td>1.849</td>
<td>38</td>
<td>1.879023661</td>
</tr>
<tr>
<td>11</td>
<td>7.779</td>
<td>36</td>
<td>7.895795199</td>
</tr>
<tr>
<td>12</td>
<td>6.837</td>
<td>37</td>
<td>6.935622632</td>
</tr>
<tr>
<td>13</td>
<td>5.987</td>
<td>38</td>
<td>6.070987190</td>
</tr>
<tr>
<td>14</td>
<td>-</td>
<td>22</td>
<td>2.697871538</td>
</tr>
<tr>
<td>15</td>
<td>-</td>
<td>20</td>
<td>1.137414432</td>
</tr>
<tr>
<td>16</td>
<td>-</td>
<td>17</td>
<td>0.263329337</td>
</tr>
<tr>
<td>18</td>
<td>-</td>
<td>37</td>
<td>0.828758224</td>
</tr>
<tr>
<td>19</td>
<td>-</td>
<td>38</td>
<td>4.096709132</td>
</tr>
</tbody>
</table>

wp: working precision; ED: effective number of significant digits; Accu: accuracy measured by the number of significant digits; CPU: computing time in seconds.

Bromwich: Bromwich integration

![Figure 4.3. The price of Asian option Case 1 computed by Bromwich integral with machine precision](image)

Using multi-precision, we can continue to increase the accuracy. However, we find the number of effective digits in the result computed by Bromwich integral can not accurately reflect the size of round-off error. Take the price of Case 1 with \( x = 5600 \) and \( \text{wp} = 20 \) in Table 4.10 as an example. The number of effective digits in the result is 8 digits more than the accuracy. We could think the result is not constrained by the round-off error. But, this is not the case. If we increase the working precision to 40 digits with other factors fixed as in Table 4.11, it is surprising to discover that the accuracy increases from 12 significant digits to 31 significant digits. The cause of it could be associated with internal computations of built-in function “NIntegrate”. Therefore, we need to be careful when we set the
digits with other factors fixed as in Table 4.11, it is surprising to discover that the accuracy increases
70 | Chapter 2. Comparing New and Extant Numerical and Analytical Methods of Asian Option Pricing

from 12 significant digits to 31 significant digits. The cause of it could be associated with internal
computations of built-in function “NIntegrate”. Therefore, we need to be careful when we set the
working precision for Bromwich integral.
Table 4.10. Results of Bromwich integration with 20-digit precision
Bromwich

Case
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19

Reference
Accu
Price
38
0.1931737903
37
0.2464156905
37
0.3062203648
17
0.05598604154
38
0.2183875466
39
0.1722687410
30
0.3500952190
36
2.815862016
38
2.310878887
38
1.879023661
36
7.895795199
37
6.935422632
38
6.070987190
22
2.697871538
20
1.134741432
17
0.2853249387
38
14.98395833
37
8.828758224
38
4.696709132

x = 1900; wp = 20

x = 3500; wp = 20

x = 5600; wp = 20

ED
Accu
Price
CPU
ED
Accu
Price
CPU
ED
Accu
Price
CPU
20.0
15
0.19317 4.73 20.0
14
0.19317 4.65 20.0
12
0.19317 4.32
20.0
13
0.24642 3.67 20.0
13
0.24642 5.85 20.0
11
0.24642 5.58
20.0
13
0.30622 3.99 20.0
13
0.30622 6.24 20.0
12
0.30622 6.07
20.0
0
0.14883 58.8 20.0
0
0.091635 65.4 20.0
0
0.069335 136
20.0
6
0.21839 7.13 20.0
10
0.21839 11.8 20.0
16
0.21839 19.0
20.0
9
0.17227 9.91 20.0
13
0.17227 9.89 20.0
13
0.17227 9.64
20.0
12
0.35010 1.72 20.0
12
0.35010 2.70 20.0
10
0.35010 2.57
20.0
12
2.8159 5.26 20.0
11
2.8159 8.44 20.0
15
2.8159 8.39
20.0
12
2.3109 4.88 20.0
13
2.3109 7.99 20.0
11
2.3109 7.72
20.0
14
1.8790 4.48 20.0
12
1.8790 6.24 20.0
15
1.8790 6.18
20.0
12
7.8958 4.87 20.0
12
7.8958 7.91 20.0
14
7.8958 6.36
20.0
13
6.9354 3.81 20.0
13
6.9354 6.15 20.0
11
6.9354 6.04
20.0
14
6.0710 3.06 20.0
13
6.0710 4.85 20.0
11
6.0710 4.73
20.0
1
3.3204 43.5 20.0
2
2.7461 71.9 20.0
1
2.6300 77.5
20.0
0
2.2082 44.4 20.0
1
1.4996 68.7 20.0
1
1.2439 71.9
20.0
0
1.3044 61.5 20.0
0
0.61838 66.6 20.0
0
0.38463 71.9
20.0
6
14.984 7.35 20.0
10
14.984 12.4 20.0
15
14.984 12.7
20.0
6
8.8288 11.5 20.0
11
8.8288 11.7 20.0
15
8.8288 11.9
20.0
6
4.6967 11.2 20.0
10
4.6967 11.3 20.0
15
4.6967 11.5

wp: working precision; ED: effective number of significant digits; Accu: accuracy measured by the number of significant digits; CPU:
computing time in seconds.
Bromwich: Bromwich integration.

Table 4.11. Results of Bromwich integration with 40-digit precision
Bromwich
Case
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19

Reference
Accu
Price
38
0.1931737903
37
0.2464156905
37
0.3062203648
17
0.05598604154
38
0.2183875466
39
0.1722687410
30
0.3500952190
36
2.815862016
38
2.310878887
38
1.879023661
36
7.895795199
37
6.935422632
38
6.070987190
22
2.697871538
20
1.134741432
17
0.2853249387
38
14.98395833
37
8.828758224
38
4.696709132

x = 1900; wp = 40

x = 3500; wp = 40

x = 5600; wp = 40

ED
Accu
Price
CPU
ED
Accu
Price
CPU
ED
Accu
Price
CPU
40.0
15
0.19317 16.2 40.0
22
0.19317 16.8 40.0
31
0.19317
17.5
40.0
13
0.24642 19.7 40.0
22
0.24642 20.7 40.0
31
0.24642
21.8
40.0
13
0.30622 21.1 40.0
21
0.30622 21.7 40.0
29
0.30622
23.5
40.0
0
0.14883 178 40.0
0
0.091635 197 40.0
0
0.069335 11 082
40.0
6
0.21839 37.1 40.0
10
0.21839 41.6 40.0
16
0.21839
2329
40.0
9
0.17227 31.9 40.0
13
0.17227 37.6 40.0
20
0.17227
37.6
40.0
20
0.35010 11.1 40.0
30
0.35010 13.4 40.0
30
0.35010
12.3
40.0
12
2.8159 29.8 40.0
20
2.8159 35.5 40.0
28
2.8159
34.8
40.0
12
2.3109 26.6 40.0
20
2.3109 27.8 40.0
27
2.3109
27.9
40.0
14
1.8790 24.4 40.0
20
1.8790 21.9 40.0
29
1.8790
22.7
40.0
12
7.8958 25.8 40.0
20
7.8958 26.8 40.0
28
7.8958
24.3
40.0
13
6.9354 21.0 40.0
20
6.9354 22.3 40.0
29
6.9354
22.6
40.0
14
6.0710 17.1 40.0
21
6.0710 18.4 40.0
29
6.0710
18.9
40.0
1
3.3204
204 40.0
2
2.7461
230 40.0
1
2.6300
245
40.0
0
2.2082
188 40.0
1
1.4996
209 40.0
1
1.2439
225
40.0
0
1.3044
177 40.0
0
0.61838 195 40.0
0
0.38463
211
40.0
6
14.984 42.0 40.0
10
14.984 39.5 40.0
15
14.984
44.0
40.0
6
8.8288 36.5 40.0
11
8.8288 38.0 40.0
15
8.8288
42.2
40.0
6
4.6967 35.0 40.0
10
4.6967 36.5 40.0
15
4.6967
39.4

wp: working precision; ED: effective number of significant digits; Accu: accuracy measured by the number of significant digits; CPU:
computing time in seconds.
Bromwich: Bromwich integration.

For difficult cases, Bromwich integral requires much bigger truncation size. To achieve an accuracy of at
least 5 digits, the truncation size is set to 29000 as in Table 4.12. Our code rounds all approximate real
numbers to exact number before put them into "NIntegrate", and use the "WorkingPrecision" option to
control the output precision. This gets rid of the annoying warning message when the internal precision
is less than the working precision. However, the computing time increases dramatically as a result. The
Bromwich integral takes an order of magnitude more computational time than the Euler method to get a
similar accuracy.


is less than the working precision. However, the computing time increases dramatically as a result. The Bromwich integral takes an order of magnitude more computational time than the Euler method to get a similar accuracy.

Table 4.12. Results of Bromwich integration on the difficult cases with enhanced parameter settings

<table>
<thead>
<tr>
<th>Case</th>
<th>Reference</th>
<th>$x = 29000$, wp = 20</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accu</td>
<td>Price</td>
</tr>
<tr>
<td>4</td>
<td>17</td>
<td>0.05598604154</td>
</tr>
<tr>
<td>14</td>
<td>22</td>
<td>2.697871338</td>
</tr>
<tr>
<td>15</td>
<td>20</td>
<td>1.134741432</td>
</tr>
<tr>
<td>16</td>
<td>17</td>
<td>0.2853249387</td>
</tr>
</tbody>
</table>

wp: working precision; ED: effective number of significant digits; Accu: accuracy measured by the number of significant digits; CPU: computing time in seconds.

Bromwich: Bromwich integration


The Gaver-Wynn-Rho algorithm involves the setting of one parameter $M$. Parameter $M$ controls the truncation size and must be even. The round-off error increases dramatically with the increase in $M$. So it is necessary to use multi-precision to obtain correct results. To illustrate how fixed precision harms the accuracy, we show what the results look like in machine precision in Table 4.13 where the results become worse and worse with the increase in $M$. Figure 4.4 reveals a similar trend as Post-Widder method has. The accuracy first increases, and then reduces.
Table 4.13. Results of the Gaver-Wynn-Rho algorithm with machine precision

<table>
<thead>
<tr>
<th>Case</th>
<th>Reference</th>
<th>M = 20</th>
<th>M = 36</th>
<th>M = 52</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accu</td>
<td>Price</td>
<td>CPU</td>
<td>Accu</td>
</tr>
<tr>
<td>1</td>
<td>0.1931737903</td>
<td>0.19310</td>
<td>0.2032</td>
<td>0.2074</td>
</tr>
<tr>
<td>2</td>
<td>0.2464156905</td>
<td>0.24624</td>
<td>0.1932</td>
<td>0.2765</td>
</tr>
<tr>
<td>3</td>
<td>0.3062203648</td>
<td>0.30610</td>
<td>0.2733</td>
<td>0.4233</td>
</tr>
<tr>
<td>4</td>
<td>0.05598604154</td>
<td>0.07076</td>
<td>0.3560</td>
<td>0.6381</td>
</tr>
<tr>
<td>5</td>
<td>0.218375466</td>
<td>0.21841</td>
<td>0.3156</td>
<td>0.1815</td>
</tr>
<tr>
<td>6</td>
<td>0.1722687410</td>
<td>0.17203</td>
<td>0.83</td>
<td>0.1821</td>
</tr>
<tr>
<td>7</td>
<td>0.3500952190</td>
<td>0.34961</td>
<td>0.12</td>
<td>0.3949</td>
</tr>
<tr>
<td>8</td>
<td>2.815862016</td>
<td>2.81311</td>
<td>0.58</td>
<td>-2.0258</td>
</tr>
<tr>
<td>9</td>
<td>2.310878887</td>
<td>2.3090</td>
<td>0.59</td>
<td>-2.5278</td>
</tr>
<tr>
<td>10</td>
<td>1.879023661</td>
<td>1.8781</td>
<td>0.42</td>
<td>1.9417</td>
</tr>
<tr>
<td>11</td>
<td>7.89579199</td>
<td>7.8884</td>
<td>0.37</td>
<td>9.2124</td>
</tr>
<tr>
<td>12</td>
<td>6.935422632</td>
<td>6.9304</td>
<td>0.22</td>
<td>9.0363</td>
</tr>
<tr>
<td>13</td>
<td>6.070987190</td>
<td>6.0613</td>
<td>0.22</td>
<td>6.6976</td>
</tr>
<tr>
<td>14</td>
<td>2.697871538</td>
<td>2.6945</td>
<td>0.47</td>
<td>9.6542</td>
</tr>
<tr>
<td>15</td>
<td>1.134741432</td>
<td>1.1265</td>
<td>0.34</td>
<td>1.5684</td>
</tr>
<tr>
<td>16</td>
<td>0.2853249387</td>
<td>0.35002</td>
<td>0.32</td>
<td>1.4534</td>
</tr>
<tr>
<td>17</td>
<td>14.98395833</td>
<td>14.935</td>
<td>0.44</td>
<td>21.536</td>
</tr>
<tr>
<td>18</td>
<td>8.828758224</td>
<td>8.8332</td>
<td>0.26</td>
<td>15.716</td>
</tr>
<tr>
<td>19</td>
<td>4.696709132</td>
<td>4.7117</td>
<td>0.20</td>
<td>1.8968</td>
</tr>
</tbody>
</table>

wp: working precision; ED: effective number of significant digits; Accu: accuracy measured by the number of significant digits; CPU: computing time in seconds.

Figure 4.4. The price of Asian option Case 1 computed by GWR with machine precision

In Table 4.14, the working precision is set to $2M$. Then, the yielded accuracy increases with $M$. When $M$ gets large, the working precision becomes large proportionally. As $M = 52$ and working precision is set to 104 digits, the number of effective digits in the result is much higher than the yielded accuracy.

The excess effective digits are actually wasted in the computations. We can reduce working precision to save computing time without harming the accuracy. More intuitively, If we lower the working precision by 10 digits, the effective digits in the result will decrease by 10 digits accordingly.
### Table 4.14. Results of the Gaver-Wynn-Rho algorithm with arbitrary precision

<table>
<thead>
<tr>
<th>Case</th>
<th>Reference</th>
<th>M = 20; wp = 2 M</th>
<th>M = 38; wp = 2 M</th>
<th>M = 52; wp = 2 M</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accu</td>
<td>Price</td>
<td>Accu</td>
<td>Price</td>
</tr>
<tr>
<td>1</td>
<td>38</td>
<td>0.1931773903</td>
<td>16.4</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>37</td>
<td>0.2464153905</td>
<td>14.8</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>37</td>
<td>0.3062203868</td>
<td>13.7</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>36</td>
<td>0.05598604154</td>
<td>12.7</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>38</td>
<td>0.2183975466</td>
<td>11.1</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>39</td>
<td>0.1726807410</td>
<td>10.5</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>30</td>
<td>0.3500952190</td>
<td>15.5</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>36</td>
<td>2.815862016</td>
<td>13.4</td>
<td>6</td>
</tr>
<tr>
<td>9</td>
<td>38</td>
<td>2.510788887</td>
<td>12.2</td>
<td>6</td>
</tr>
<tr>
<td>10</td>
<td>38</td>
<td>1.879926661</td>
<td>10.8</td>
<td>6</td>
</tr>
<tr>
<td>11</td>
<td>36</td>
<td>7.895791499</td>
<td>13.6</td>
<td>6</td>
</tr>
<tr>
<td>12</td>
<td>37</td>
<td>6.935422662</td>
<td>11.7</td>
<td>5</td>
</tr>
<tr>
<td>13</td>
<td>38</td>
<td>6.070871990</td>
<td>15.5</td>
<td>5</td>
</tr>
<tr>
<td>14</td>
<td>22</td>
<td>2.697871538</td>
<td>12.9</td>
<td>3</td>
</tr>
<tr>
<td>15</td>
<td>20</td>
<td>1.134741432</td>
<td>14.1</td>
<td>3</td>
</tr>
<tr>
<td>16</td>
<td>18</td>
<td>0.2852249387</td>
<td>12.3</td>
<td>2</td>
</tr>
<tr>
<td>17</td>
<td>38</td>
<td>14.013959133</td>
<td>11.1</td>
<td>5</td>
</tr>
<tr>
<td>18</td>
<td>36</td>
<td>6.828756244</td>
<td>14.8</td>
<td>5</td>
</tr>
<tr>
<td>19</td>
<td>38</td>
<td>4.696709132</td>
<td>13.3</td>
<td>5</td>
</tr>
</tbody>
</table>

wp: working precision; ED: effective number of significant digits; Accu: accuracy measured by the number of significant digits; CPU: computing time in seconds.  

### Table 4.15. Results of the Gaver-Wynn-Rho algorithm on the difficult cases with enhanced parameter settings

<table>
<thead>
<tr>
<th>Case</th>
<th>Reference</th>
<th>M = 34; wp = 65</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accu</td>
<td>Price</td>
</tr>
<tr>
<td>4</td>
<td>17</td>
<td>0.05598604154</td>
</tr>
<tr>
<td>14</td>
<td>22</td>
<td>2.697871538</td>
</tr>
<tr>
<td>15</td>
<td>20</td>
<td>1.134741432</td>
</tr>
<tr>
<td>16</td>
<td>17</td>
<td>0.2853249387</td>
</tr>
</tbody>
</table>

wp: working precision; ED: effective number of significant digits; Accu: accuracy measured by the number of significant digits; CPU: computing time in seconds.  

### 4.7. Numerical results for the fixed Talbot method

The fixed Talbot method (FT) has one free parameter $M$. It works very well for normal cases with small value of $M$ (see Table 4.16 and 4.17) although larger $M$ is required for difficult cases (see Table 4.18).

In machine precision, the fixed Talbot method can achieve about 14-digit accuracy which is adequate for derivatives traders. But if we want to have higher accuracy, using multi-precision is necessary.
Table 4.16. Results of the fixed Talbot method with machine precision

<table>
<thead>
<tr>
<th>Case</th>
<th>Reference</th>
<th>FT : wp = MachinePrecision</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accu</td>
<td>Price</td>
</tr>
<tr>
<td>1</td>
<td>38</td>
<td>0.1931737903</td>
</tr>
<tr>
<td>2</td>
<td>37</td>
<td>0.2464156905</td>
</tr>
<tr>
<td>3</td>
<td>37</td>
<td>0.3062203648</td>
</tr>
<tr>
<td>4</td>
<td>17</td>
<td>0.05508604154</td>
</tr>
<tr>
<td>5</td>
<td>38</td>
<td>0.2183875466</td>
</tr>
<tr>
<td>6</td>
<td>39</td>
<td>0.1726687410</td>
</tr>
<tr>
<td>7</td>
<td>30</td>
<td>0.3500952190</td>
</tr>
<tr>
<td>8</td>
<td>36</td>
<td>2.815862016</td>
</tr>
<tr>
<td>9</td>
<td>38</td>
<td>2.310878887</td>
</tr>
<tr>
<td>10</td>
<td>38</td>
<td>1.879023661</td>
</tr>
<tr>
<td>11</td>
<td>36</td>
<td>7.895795199</td>
</tr>
<tr>
<td>12</td>
<td>37</td>
<td>6.935422632</td>
</tr>
<tr>
<td>13</td>
<td>38</td>
<td>6.070987190</td>
</tr>
<tr>
<td>14</td>
<td>22</td>
<td>2.697871538</td>
</tr>
<tr>
<td>15</td>
<td>10</td>
<td>1.134741432</td>
</tr>
<tr>
<td>16</td>
<td>17</td>
<td>0.2853249387</td>
</tr>
<tr>
<td>17</td>
<td>38</td>
<td>12.983985833</td>
</tr>
<tr>
<td>18</td>
<td>37</td>
<td>8.282785224</td>
</tr>
<tr>
<td>19</td>
<td>38</td>
<td>4.696709132</td>
</tr>
</tbody>
</table>

wp: working precision; ED: effective number of significant digits; Accu: accuracy measured by the number of significant digits; CPU: computing time in seconds.

FT: the fixed Talbot method.

Figure 4.5. The price of Asian option Case 1 computed by FT with machine precision
Table 4.17. Results of the fixed Talbot method with arbitrary precision

<table>
<thead>
<tr>
<th>Case</th>
<th>Reference</th>
<th>M = 19; wp = 20</th>
<th>M = 29; wp = 25</th>
<th>M = 39; wp = 30</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accu</td>
<td>Price</td>
<td>FT</td>
<td>Accu</td>
</tr>
<tr>
<td></td>
<td>ED</td>
<td>Accu</td>
<td>CPU</td>
<td>ED</td>
</tr>
<tr>
<td>1</td>
<td>38</td>
<td>0.1931773903</td>
<td>17.8</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>37</td>
<td>0.2464429905</td>
<td>17.8</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>33</td>
<td>0.2357223843</td>
<td>17.8</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>35</td>
<td>0.2183755466</td>
<td>17.8</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>39</td>
<td>0.1722687410</td>
<td>17.8</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>30</td>
<td>0.3500952190</td>
<td>17.8</td>
<td>11</td>
</tr>
<tr>
<td>7</td>
<td>36</td>
<td>2.81582016</td>
<td>17.8</td>
<td>9</td>
</tr>
<tr>
<td>8</td>
<td>39</td>
<td>2.310878887</td>
<td>17.8</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>37</td>
<td>1.879023661</td>
<td>17.8</td>
<td>7</td>
</tr>
<tr>
<td>10</td>
<td>33</td>
<td>1.37002661</td>
<td>17.8</td>
<td>6</td>
</tr>
<tr>
<td>11</td>
<td>36</td>
<td>0.895782159</td>
<td>17.8</td>
<td>5</td>
</tr>
<tr>
<td>12</td>
<td>37</td>
<td>0.935422662</td>
<td>17.8</td>
<td>4</td>
</tr>
<tr>
<td>13</td>
<td>38</td>
<td>0.607087190</td>
<td>17.8</td>
<td>3</td>
</tr>
<tr>
<td>14</td>
<td>22</td>
<td>2.697871538</td>
<td>17.8</td>
<td>2</td>
</tr>
<tr>
<td>15</td>
<td>20</td>
<td>1.134744352</td>
<td>17.8</td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>18</td>
<td>0.285329387</td>
<td>17.8</td>
<td>1</td>
</tr>
</tbody>
</table>

wp: working precision; ED: effective number of significant digits; Accu: accuracy measured by the number of significant digits; CPU: computing time in seconds.
FT: the fixed Talbot method.

Table 4.18. Results of the fixed Talbot method on the difficult cases with enhanced parameter settings

<table>
<thead>
<tr>
<th>Case</th>
<th>Reference</th>
<th>M = 60; wp = 20</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accu</td>
<td>Price</td>
</tr>
<tr>
<td></td>
<td>ED</td>
<td>Accu</td>
</tr>
<tr>
<td>4</td>
<td>17</td>
<td>0.05598604154</td>
</tr>
<tr>
<td>14</td>
<td>22</td>
<td>2.697871538</td>
</tr>
<tr>
<td>18</td>
<td>20</td>
<td>1.134744352</td>
</tr>
<tr>
<td>16</td>
<td>17</td>
<td>0.285329387</td>
</tr>
</tbody>
</table>

wp: working precision; ED: effective number of significant digits; Accu: accuracy measured by the number of significant digits; CPU: computing time in seconds.
FT: the fixed Talbot method.

4.8. Numerical results of the unified Gaver-Stehfest algorithm

The unified Gaver-Stehfest algorithm (UniG) has one free parameter $M$. The working precision required for UniG method is about 1.5 $M$. In machine precision, the UniG method can give ridiculous results with careless selection of $M$ (see Table 4.19). Even we choose most suitable $M$, the accuracy obtained is not enough for traders, about 3 digits. The UniG method diverges quickly when $M$ get large as shown in Figure 4.6.
Table 4.19. Results of the unified Gaver-Stehfest algorithm with machine precision

<table>
<thead>
<tr>
<th>Case</th>
<th>Reference</th>
<th>M = 31</th>
<th>N = 62</th>
<th>M = 78</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accu</td>
<td>Price</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Accu</td>
<td>Price</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Accu</td>
<td>Price</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>38</td>
<td>0.1931737503</td>
<td>1.6947×10^{-10}</td>
<td>0.03</td>
</tr>
<tr>
<td>2</td>
<td>37</td>
<td>0.2464156905</td>
<td>4.3646×10^{-10}</td>
<td>0.01</td>
</tr>
<tr>
<td>3</td>
<td>37</td>
<td>0.3562203649</td>
<td>3.0868×10^{-10}</td>
<td>0.03</td>
</tr>
<tr>
<td>4</td>
<td>37</td>
<td>0.055986041254</td>
<td>2.6020×10^{-10}</td>
<td>0.17</td>
</tr>
<tr>
<td>5</td>
<td>38</td>
<td>0.2183875466</td>
<td>1.7863×10^{-10}</td>
<td>0.11</td>
</tr>
<tr>
<td>6</td>
<td>39</td>
<td>0.1722688414</td>
<td>1.8529×10^{-10}</td>
<td>0.09</td>
</tr>
<tr>
<td>7</td>
<td>38</td>
<td>0.3500952190</td>
<td>1.7849×10^{-10}</td>
<td>0.09</td>
</tr>
<tr>
<td>8</td>
<td>38</td>
<td>2.815862015</td>
<td>7.4455×10^{-10}</td>
<td>0.05</td>
</tr>
<tr>
<td>9</td>
<td>38</td>
<td>2.310978987</td>
<td>3.9276×10^{-10}</td>
<td>0.03</td>
</tr>
<tr>
<td>10</td>
<td>38</td>
<td>1.87923661</td>
<td>1.7407×10^{-10}</td>
<td>0.03</td>
</tr>
<tr>
<td>11</td>
<td>36</td>
<td>7.895795199</td>
<td>1.6812×10^{-10}</td>
<td>0.03</td>
</tr>
<tr>
<td>12</td>
<td>37</td>
<td>6.935422632</td>
<td>1.0022×10^{-10}</td>
<td>0.03</td>
</tr>
<tr>
<td>13</td>
<td>38</td>
<td>6.070987190</td>
<td>6.2503×10^{-10}</td>
<td>0.03</td>
</tr>
<tr>
<td>14</td>
<td>22</td>
<td>2.699871538</td>
<td>-2.8865×10^{-10}</td>
<td>0.03</td>
</tr>
<tr>
<td>15</td>
<td>20</td>
<td>1.134743432</td>
<td>5.7882×10^{-10}</td>
<td>2.93</td>
</tr>
<tr>
<td>16</td>
<td>17</td>
<td>0.285249367</td>
<td>-4.0757×10^{-10}</td>
<td>2.61</td>
</tr>
<tr>
<td>17</td>
<td>38</td>
<td>14.93395833</td>
<td>5.5456×10^{-10}</td>
<td>0.12</td>
</tr>
<tr>
<td>18</td>
<td>37</td>
<td>8.826758224</td>
<td>-1.6858×10^{-10}</td>
<td>0.11</td>
</tr>
<tr>
<td>19</td>
<td>38</td>
<td>4.96709132</td>
<td>-8.2007×10^{-10}</td>
<td>0.09</td>
</tr>
</tbody>
</table>

wp: working precision; ED: effective number of significant digits; Accu: accuracy measured by the number of significant digits; CPU: computing time in seconds.

UniG: the unified Gaver-Stehfest algorithm.

Figure 4.6. The price of Asian option Case 1 computed by UniG with machine precision

After setting the working precision to 1.5 $M$, the UniG method yields increasing accuracy as $M$ in Table 4.20. It also performs well for difficult cases as shown in Table 4.21.
Table 4.20. Results of the unified Gaver-Stehfest algorithm with arbitrary precision

<table>
<thead>
<tr>
<th>Case</th>
<th>Reference</th>
<th>( M = 31; ) ( wp = 1.5 )</th>
<th>( M = 52; ) ( wp = 1.5 )</th>
<th>( M = 78; ) ( wp = 1.5 )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accu</td>
<td>Price</td>
<td>CPU</td>
<td>Accu</td>
</tr>
<tr>
<td>1</td>
<td>38</td>
<td>0.193177903</td>
<td>14.0</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>37</td>
<td>0.246413605</td>
<td>14.0</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>35</td>
<td>0.306220341</td>
<td>14.0</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>36</td>
<td>0.353457845</td>
<td>14.0</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>38</td>
<td>0.218375466</td>
<td>14.0</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>39</td>
<td>0.1722687410</td>
<td>14.0</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>37</td>
<td>0.3500952190</td>
<td>14.0</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>36</td>
<td>2.81562016</td>
<td>14.0</td>
<td>7</td>
</tr>
<tr>
<td>9</td>
<td>35</td>
<td>2.51688887</td>
<td>14.0</td>
<td>7</td>
</tr>
<tr>
<td>10</td>
<td>38</td>
<td>1.875025661</td>
<td>14.0</td>
<td>7</td>
</tr>
<tr>
<td>11</td>
<td>36</td>
<td>7.89578159</td>
<td>14.0</td>
<td>7</td>
</tr>
<tr>
<td>12</td>
<td>37</td>
<td>4.935422632</td>
<td>14.0</td>
<td>7</td>
</tr>
<tr>
<td>13</td>
<td>38</td>
<td>6.070987190</td>
<td>14.0</td>
<td>7</td>
</tr>
<tr>
<td>14</td>
<td>22</td>
<td>2.69787153</td>
<td>14.0</td>
<td>2</td>
</tr>
<tr>
<td>15</td>
<td>20</td>
<td>1.13474132</td>
<td>14.0</td>
<td>2</td>
</tr>
<tr>
<td>16</td>
<td>18</td>
<td>0.2853249387</td>
<td>14.0</td>
<td>2</td>
</tr>
<tr>
<td>17</td>
<td>38</td>
<td>14.99398913</td>
<td>14.0</td>
<td>6</td>
</tr>
<tr>
<td>18</td>
<td>37</td>
<td>8.828754224</td>
<td>14.0</td>
<td>6</td>
</tr>
<tr>
<td>19</td>
<td>38</td>
<td>4.696709132</td>
<td>14.0</td>
<td>5</td>
</tr>
</tbody>
</table>

wp: working precision; ED: effective number of significant digits; Accu: accuracy measured by the number of significant digits; CPU: computing time in seconds.

UniG: the unified Gaver-Stehfest algorithm.

Table 4.21. Results of the unified Gaver-Stehfest algorithm on the difficult cases with enhanced parameter settings

<table>
<thead>
<tr>
<th>Case</th>
<th>Reference</th>
<th>( M = 78; ) ( wp = 90 )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accu</td>
<td>Price</td>
</tr>
<tr>
<td>4</td>
<td>17</td>
<td>0.05598604154</td>
</tr>
<tr>
<td>14</td>
<td>22</td>
<td>2.697871538</td>
</tr>
<tr>
<td>15</td>
<td>20</td>
<td>1.134741352</td>
</tr>
<tr>
<td>16</td>
<td>17</td>
<td>0.2853249387</td>
</tr>
</tbody>
</table>

wp: working precision; ED: effective number of significant digits; Accu: accuracy measured by the number of significant digits; CPU: computing time in seconds.

UniG: the unified Gaver-Stehfest algorithm.

### 4.9. Numerical results of the unified Euler algorithm

The unified Euler algorithm (UniE) has one free parameter \( M \). From Table 4.22, machine precision is sufficient for traders to value an Asian option. The maximum accuracy obtained is about 13 digits. The propagation of the round-off error of the UniE method is not sensitive to \( M \) as shown in Figure 4.7.
Table 4.22. Results of the unified Euler algorithm with machine precision

<table>
<thead>
<tr>
<th>Case</th>
<th>Reference</th>
<th>UnifE: wp = MachinePrecision</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accu</td>
<td>M = 11</td>
</tr>
<tr>
<td></td>
<td>Price</td>
<td>CPU</td>
</tr>
<tr>
<td>1</td>
<td>38</td>
<td>0.19317</td>
</tr>
<tr>
<td>2</td>
<td>37</td>
<td>0.24642</td>
</tr>
<tr>
<td>3</td>
<td>37</td>
<td>0.30622</td>
</tr>
<tr>
<td>4</td>
<td>38</td>
<td>0.055986</td>
</tr>
<tr>
<td>5</td>
<td>39</td>
<td>0.21839</td>
</tr>
<tr>
<td>6</td>
<td>39</td>
<td>0.17227</td>
</tr>
<tr>
<td>7</td>
<td>30</td>
<td>0.35010</td>
</tr>
<tr>
<td>8</td>
<td>36</td>
<td>2.81586</td>
</tr>
<tr>
<td>9</td>
<td>38</td>
<td>2.31088</td>
</tr>
<tr>
<td>10</td>
<td>38</td>
<td>1.87902</td>
</tr>
<tr>
<td>11</td>
<td>36</td>
<td>7.895795</td>
</tr>
<tr>
<td>12</td>
<td>37</td>
<td>6.93542</td>
</tr>
<tr>
<td>13</td>
<td>38</td>
<td>6.0709#</td>
</tr>
<tr>
<td>14</td>
<td>37</td>
<td>2.69787</td>
</tr>
<tr>
<td>15</td>
<td>38</td>
<td>1.34741</td>
</tr>
<tr>
<td>16</td>
<td>38</td>
<td>0.28532</td>
</tr>
<tr>
<td>17</td>
<td>38</td>
<td>14.98395</td>
</tr>
<tr>
<td>18</td>
<td>37</td>
<td>8.82878</td>
</tr>
<tr>
<td>19</td>
<td>38</td>
<td>4.69670</td>
</tr>
</tbody>
</table>

wp: working precision; ED: effective number of significant digits; Accu: accuracy measured by the number of significant digits; CPU: computing time in seconds.

UniE: the unified Euler algorithm.

Figure 4.7. The price of Asian option Case 1 computed by UniE with machine precision

Table 4.23 and Table 4.24 shows the performance of the UniE method in multi-precision environment.
Table 4.23. Results of the unified Euler algorithm with arbitrary precision

<table>
<thead>
<tr>
<th>Case</th>
<th>Reference</th>
<th>M = 11; wp = 20</th>
<th>M = 21; wp = 20</th>
<th>M = 31; wp = 25</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accu</td>
<td>Price</td>
<td>Accu</td>
<td>Price</td>
</tr>
<tr>
<td>1</td>
<td>38</td>
<td>0.1911778903</td>
<td>17.8</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>37</td>
<td>0.2464129805</td>
<td>17.8</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>35</td>
<td>0.3032203846</td>
<td>17.9</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>33</td>
<td>0.3553914154</td>
<td>17.9</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>38</td>
<td>0.4180954664</td>
<td>17.9</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>39</td>
<td>0.4726878104</td>
<td>17.9</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>30</td>
<td>0.5300952190</td>
<td>17.9</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>36</td>
<td>2.81562016</td>
<td>17.9</td>
<td>6</td>
</tr>
<tr>
<td>9</td>
<td>38</td>
<td>2.910778888</td>
<td>17.9</td>
<td>6</td>
</tr>
<tr>
<td>10</td>
<td>39</td>
<td>3.067022661</td>
<td>17.9</td>
<td>6</td>
</tr>
<tr>
<td>11</td>
<td>36</td>
<td>7.85792159</td>
<td>17.9</td>
<td>6</td>
</tr>
<tr>
<td>12</td>
<td>37</td>
<td>4.935422664</td>
<td>17.9</td>
<td>6</td>
</tr>
<tr>
<td>13</td>
<td>38</td>
<td>6.070987190</td>
<td>17.9</td>
<td>6</td>
</tr>
<tr>
<td>14</td>
<td>22</td>
<td>2.69781538</td>
<td>17.8</td>
<td>3</td>
</tr>
<tr>
<td>15</td>
<td>20</td>
<td>1.13471432</td>
<td>17.8</td>
<td>3</td>
</tr>
<tr>
<td>16</td>
<td>18</td>
<td>2.983249383</td>
<td>17.9</td>
<td>3</td>
</tr>
<tr>
<td>17</td>
<td>18</td>
<td>14.93935913</td>
<td>17.8</td>
<td>7</td>
</tr>
<tr>
<td>18</td>
<td>38</td>
<td>8.828758224</td>
<td>17.9</td>
<td>5</td>
</tr>
<tr>
<td>19</td>
<td>38</td>
<td>4.696709132</td>
<td>17.9</td>
<td>5</td>
</tr>
</tbody>
</table>

wp: working precision; ED: effective number of significant digits; Accu: accuracy measured by the number of significant digits; CPU: computing time in seconds.

UniE: the unified Euler algorithm.

Table 4.24. Results of the unified Euler algorithm on the difficult cases with enhanced parameter settings

<table>
<thead>
<tr>
<th>Case</th>
<th>Reference</th>
<th>M = 18; wp = 20</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accu</td>
<td>Price</td>
</tr>
<tr>
<td>4</td>
<td>17</td>
<td>0.05598604154</td>
</tr>
<tr>
<td>14</td>
<td>22</td>
<td>2.697871558</td>
</tr>
<tr>
<td>15</td>
<td>20</td>
<td>1.134714352</td>
</tr>
<tr>
<td>16</td>
<td>17</td>
<td>0.2853249387</td>
</tr>
</tbody>
</table>

wp: working precision; ED: effective number of significant digits; Accu: accuracy measured by the number of significant digits; CPU: computing time in seconds.

UniE: the unified Euler algorithm.

■ 4.10. Numerical results for the unified Talbot algorithm

The unified Talbot algorithm (UniT) has one free parameter $M$. It performs very similar as the FT method. Table 4.25 shows the performance in machine precision, while Table 4.26 and 4.27 demonstrate the performance in multi-precision. Figure 4.8 reveals the oscillation in price with the choice of $M$ in machine precision.
Table 4.25. Results of the unified Talbot algorithm with machine precision

<table>
<thead>
<tr>
<th>Case</th>
<th>Reference</th>
<th>M = 19</th>
<th>M = 29</th>
<th>M = 39</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accu</td>
<td>Price</td>
<td>Accu</td>
<td>Price</td>
</tr>
<tr>
<td>1</td>
<td>0.1931737903</td>
<td>9</td>
<td>0.193170.02</td>
<td>14</td>
</tr>
<tr>
<td>2</td>
<td>0.2464156905</td>
<td>9</td>
<td>0.246420.02</td>
<td>14</td>
</tr>
<tr>
<td>3</td>
<td>0.3062203648</td>
<td>9</td>
<td>0.306220.02</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td>0.05598604154</td>
<td>0</td>
<td>0.0569448.49</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>0.2183875466</td>
<td>7</td>
<td>0.218390.02</td>
<td>10</td>
</tr>
<tr>
<td>6</td>
<td>0.1722687419</td>
<td>6</td>
<td>0.172270.08</td>
<td>12</td>
</tr>
<tr>
<td>7</td>
<td>0.050952190</td>
<td>11</td>
<td>0.050950.02</td>
<td>15</td>
</tr>
<tr>
<td>8</td>
<td>2.815862016</td>
<td>9</td>
<td>2.81590.05</td>
<td>14</td>
</tr>
<tr>
<td>9</td>
<td>2.310878887</td>
<td>9</td>
<td>2.31090.03</td>
<td>15</td>
</tr>
<tr>
<td>10</td>
<td>1.879023661</td>
<td>8</td>
<td>1.87900.02</td>
<td>14</td>
</tr>
<tr>
<td>11</td>
<td>7.895795199</td>
<td>8</td>
<td>7.89580.03</td>
<td>13</td>
</tr>
<tr>
<td>12</td>
<td>6.935422632</td>
<td>8</td>
<td>6.93540.02</td>
<td>14</td>
</tr>
<tr>
<td>13</td>
<td>6.070987190</td>
<td>8</td>
<td>6.07100.02</td>
<td>14</td>
</tr>
<tr>
<td>14</td>
<td>2.697871538</td>
<td>1</td>
<td>2.69804.12</td>
<td>4</td>
</tr>
<tr>
<td>15</td>
<td>1.134741432</td>
<td>0</td>
<td>1.134743.88</td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>0.2853249387</td>
<td>0</td>
<td>0.285323.57</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>8.828785224</td>
<td>5</td>
<td>8.82880.20</td>
<td>10</td>
</tr>
<tr>
<td>19</td>
<td>4.696709132</td>
<td>5</td>
<td>4.69670.09</td>
<td>10</td>
</tr>
</tbody>
</table>

wp: working precision; ED: effective number of significant digits; Accu: accuracy measured by the number of significant digits; CPU: computing time in seconds.

UniT: the unified Talbot algorithm.

Figure 4.8. The price of Asian option Case 1 computed by UniT with machine precision
Table 4.26. Results of the unified Talbot algorithm with arbitrary precision

<table>
<thead>
<tr>
<th>Case</th>
<th>Reference</th>
<th>M = 19; wp = 20</th>
<th>M = 29; wp = 25</th>
<th>M = 39; wp = 30</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accu</td>
<td>Price</td>
<td>ED</td>
<td>Accu</td>
</tr>
<tr>
<td>1</td>
<td>36</td>
<td>0.1931737903</td>
<td>17.8</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>37</td>
<td>0.2464293905</td>
<td>17.8</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>38</td>
<td>0.2984208333</td>
<td>17.8</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>39</td>
<td>0.3559644149</td>
<td>17.8</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>40</td>
<td>0.2181959466</td>
<td>17.8</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>41</td>
<td>0.1726887410</td>
<td>17.8</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>42</td>
<td>0.350952190</td>
<td>17.8</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>43</td>
<td>2.815620161</td>
<td>17.8</td>
<td>9</td>
</tr>
<tr>
<td>9</td>
<td>44</td>
<td>2.516788871</td>
<td>17.8</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>45</td>
<td>1.679026661</td>
<td>17.8</td>
<td>5</td>
</tr>
<tr>
<td>11</td>
<td>46</td>
<td>7.895791199</td>
<td>17.8</td>
<td>8</td>
</tr>
<tr>
<td>12</td>
<td>47</td>
<td>6.935422642</td>
<td>17.8</td>
<td>8</td>
</tr>
<tr>
<td>13</td>
<td>48</td>
<td>6.070987190</td>
<td>17.8</td>
<td>8</td>
</tr>
<tr>
<td>14</td>
<td>49</td>
<td>2.697815328</td>
<td>17.8</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>50</td>
<td>1.134714342</td>
<td>17.8</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>51</td>
<td>0.821329328</td>
<td>17.8</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>52</td>
<td>14.983953513</td>
<td>17.8</td>
<td>5</td>
</tr>
<tr>
<td>18</td>
<td>53</td>
<td>8.828754224</td>
<td>17.8</td>
<td>9</td>
</tr>
<tr>
<td>19</td>
<td>54</td>
<td>4.696701302</td>
<td>17.8</td>
<td>9</td>
</tr>
</tbody>
</table>

wp: working precision; ED: effective number of significant digits; Accu: accuracy measured by the number of significant digits; CPU: computing time in seconds.

UniT: the unified Talbot algorithm.

Table 4.27. Results of the unified Talbot algorithm on the difficult cases with enhanced parameter settings

<table>
<thead>
<tr>
<th>Case</th>
<th>Reference</th>
<th>M = 60; wp = 20</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accu</td>
<td>Price</td>
</tr>
<tr>
<td>4</td>
<td>17</td>
<td>0.0559864154</td>
</tr>
<tr>
<td>14</td>
<td>22</td>
<td>2.697815358</td>
</tr>
<tr>
<td>15</td>
<td>20</td>
<td>1.134714352</td>
</tr>
<tr>
<td>16</td>
<td>17</td>
<td>0.2853249387</td>
</tr>
</tbody>
</table>

wp: working precision; ED: effective number of significant digits; Accu: accuracy measured by the number of significant digits; CPU: computing time in seconds.

UniT: the unified Talbot algorithm.

### 4.11. Numerical results of the Laguerre method

As Weideman (1999) pointed out, the major advantage of the Laguerre method over other Laplace transform inversion algorithms is that it provides one with function expansions of the Laplace transform, namely the Laguerre coefficients $q_n$. They can be evaluated without the information on the $t$ value. This means we may compute $q_n$ once for any given $\hat{f}(s)$, and use the values to evaluate $f(t)$ for any $t$ value. In this way, we can make computations very quick, even immediately since evaluation of the associated Laguerre functions $L_n(t)$ has no difficulty at all. However, this great advantage cannot be applied to the problem of pricing Asian options. This is because the Laplace transform of an Asian option is a function of the complex variable $\lambda$, the normalized interest rate $\nu$ and the normalized strike price $q$. The normalized time to maturity $h$ served as the $t$ value in $f(t)$ is, unfortunately, embedded in the Laplace transform. Thus, the values of $q_n$ changes with $h$. Hence, the Laguerre method cannot gain the benefits
from function expansions in the application of pricing Asian options.

The Laguerre method requires selection of a number of parameters. First of all, the Laplace transform of the Asian option price needs to be shifted so that no singularities are to the right of the contour. This can be done by setting the shifting parameter $\alpha = 1$. Then, we can choose suitable values for other parameters to have desired accuracy. The parameter $m$ determines the truncation size. Increasing $m$ gains more accuracy. The parameter $\gamma$ is used to control the discretization error in $q_n$. We set $\gamma = 25$ so that the maximum accuracy of the final result is about 23 significant digits. The parameter $k$ is associated with Wynn’s $\epsilon$-algorithm. The choice of $k = 18$ is recommended for the problem of pricing Asian options. When $k = 0$, $f(t)$ is approximated by $\epsilon^{-1}$, namely $S_{m-1}$ as if Wynn’s $\epsilon$-algorithm is not used. The scaling parameters $\sigma$ and $b$ are used together with the parameter $k$ to improve the accuracy. For best performance, we set $\sigma = 0$. The value of $b$ is critical to the accuracy of the Laguerre method. Weideman (1999) proposed two algorithms for computing the optimal parameters $\sigma$ and $b$. Here, we choose the optimal $b$ by simply checking the numerical results. Different case may have different optimal $b$. The working precision is very important in the computations. When an infinite expression is encountered during the calculation, it implies the current working precision is not sufficient. We should increase the working precision to solve this problem.

Table 4.28 shows the numerical results for the prices of Asian options using Laguerre method with working precision equal to 55 digits. The optimal $b$ is chosen based on trials and it is a rough estimate. We notice that the optimal $b$ increases as $q$ becomes small. The Laguerre method yields good accuracy for normal cases and requires more truncation size for difficult cases. We carry out the same experiments in machine precision to check the effect of multi-precision. Numerical results in Table 4.29 show moderate size of $m$ gives about 6-digit accuracy and larger $m$ in fact reduces the accuracy. To obtain a accuracy of more than 5 digits for difficult cases, we need to use $m = 130$ to have desired accuracy as shown in Table 4.30.
Table 4.28. Results of the Laguerre method with arbitrary precision

<table>
<thead>
<tr>
<th>Case</th>
<th>Reference</th>
<th>b</th>
<th>m = 60, wp = 55</th>
<th>m = 60, wp = 55</th>
<th>m = 100, wp = 55</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>38</td>
<td>0.19317</td>
<td>0.19317</td>
<td>0.19317</td>
<td>0.19317</td>
</tr>
<tr>
<td>2</td>
<td>37</td>
<td>0.24641</td>
<td>0.24642</td>
<td>0.24642</td>
<td>0.24642</td>
</tr>
<tr>
<td>3</td>
<td>37</td>
<td>0.24641</td>
<td>0.24642</td>
<td>0.24642</td>
<td>0.24642</td>
</tr>
<tr>
<td>4</td>
<td>37</td>
<td>0.24641</td>
<td>0.24642</td>
<td>0.24642</td>
<td>0.24642</td>
</tr>
<tr>
<td>5</td>
<td>37</td>
<td>0.24641</td>
<td>0.24642</td>
<td>0.24642</td>
<td>0.24642</td>
</tr>
</tbody>
</table>

Laguerre: the Laguerre method.

Table 4.29. Results of the Laguerre method with machine precision

<table>
<thead>
<tr>
<th>Case</th>
<th>Reference</th>
<th>b</th>
<th>m = 60</th>
<th>m = 80</th>
<th>m = 100</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>38</td>
<td>0.19317</td>
<td>0.19317</td>
<td>0.19317</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>37</td>
<td>0.24641</td>
<td>0.24642</td>
<td>0.24642</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>37</td>
<td>0.24641</td>
<td>0.24642</td>
<td>0.24642</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>37</td>
<td>0.24641</td>
<td>0.24642</td>
<td>0.24642</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>37</td>
<td>0.24641</td>
<td>0.24642</td>
<td>0.24642</td>
<td></td>
</tr>
</tbody>
</table>

Laguerre: the Laguerre method.
Table 4.30. Results of the Laguerre method on the difficult cases with enhanced parameter settings

<table>
<thead>
<tr>
<th>Case</th>
<th>Reference</th>
<th>Accu</th>
<th>Price</th>
<th>b</th>
<th>ED</th>
<th>Accu</th>
<th>Price</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>17</td>
<td>0.05598604154</td>
<td>12000</td>
<td>19.2</td>
<td>5</td>
<td>0.055986</td>
<td>30.8</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>22</td>
<td>2.697871558</td>
<td>12000</td>
<td>19.2</td>
<td>6</td>
<td>2.6979</td>
<td>31.1</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>20</td>
<td>1.134741432</td>
<td>12000</td>
<td>19.2</td>
<td>6</td>
<td>1.1397</td>
<td>28.6</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>17</td>
<td>0.2853249387</td>
<td>12000</td>
<td>19.2</td>
<td>7</td>
<td>0.28532</td>
<td>26.9</td>
<td></td>
</tr>
</tbody>
</table>

wp: working precision; ED: effective number of significant digits; Accu: accuracy measured by the number of significant digits; CPU: computing time in seconds.

Laguerre: the Laguerre method.

### 4.12. Numerical results of the spectral series expansion

The spectral series expansion (SpectralS) involves selection of two parameters $b$ and $M$. We let $b = 1$ so that $b$ is much greater than $q$ for all 19 cases. Parameter $M$ controls the truncation size. The SpectralS method does not require high working precision. It works well in machine precision where an accuracy of 15 digits can be achieved as shown in Table 4.31. This is the limit of accuracy that machine precision can provide. Figure 4.9 displays the accuracy of the SpectralS method does not degenerate with $M$ using machine precision.

Table 4.31. Results of the spectral series expansion with machine precision

<table>
<thead>
<tr>
<th>Case</th>
<th>Reference</th>
<th>M = 44</th>
<th>M = 59</th>
<th>M = 69</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accu Price</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.1931737903</td>
<td>15</td>
<td>6.63</td>
<td>15</td>
</tr>
<tr>
<td>2</td>
<td>0.2464156905</td>
<td>14</td>
<td>6.65</td>
<td>14</td>
</tr>
<tr>
<td>3</td>
<td>0.306203648</td>
<td>15</td>
<td>6.71</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td>0.05598604154</td>
<td>15</td>
<td>6.74</td>
<td>15</td>
</tr>
<tr>
<td>5</td>
<td>0.213875466</td>
<td>11</td>
<td>7.13</td>
<td>11</td>
</tr>
<tr>
<td>6</td>
<td>0.1722687410</td>
<td>11</td>
<td>7.22</td>
<td>11</td>
</tr>
<tr>
<td>7</td>
<td>0.350095190</td>
<td>14</td>
<td>6.55</td>
<td>14</td>
</tr>
<tr>
<td>8</td>
<td>2.815920161</td>
<td>15</td>
<td>8.81</td>
<td>15</td>
</tr>
<tr>
<td>9</td>
<td>2.310878887</td>
<td>15</td>
<td>8.78</td>
<td>15</td>
</tr>
<tr>
<td>10</td>
<td>1.879236614</td>
<td>14</td>
<td>8.53</td>
<td>14</td>
</tr>
<tr>
<td>11</td>
<td>7.895795699</td>
<td>14</td>
<td>8.66</td>
<td>14</td>
</tr>
<tr>
<td>12</td>
<td>6.935422632</td>
<td>14</td>
<td>8.58</td>
<td>14</td>
</tr>
<tr>
<td>13</td>
<td>6.070987190</td>
<td>15</td>
<td>8.55</td>
<td>15</td>
</tr>
<tr>
<td>14</td>
<td>2.697871558</td>
<td>2</td>
<td>27.6</td>
<td>2</td>
</tr>
<tr>
<td>15</td>
<td>1.134741432</td>
<td>15</td>
<td>26.3</td>
<td>15</td>
</tr>
<tr>
<td>16</td>
<td>0.2853249387</td>
<td>0</td>
<td>25.1</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>8.828758224</td>
<td>11</td>
<td>8.95</td>
<td>11</td>
</tr>
<tr>
<td>19</td>
<td>4.496709132</td>
<td>5</td>
<td>8.81</td>
<td>10</td>
</tr>
</tbody>
</table>

wp: working precision; ED: effective number of significant digits; Accu: accuracy measured by the number of significant digits; CPU: computing time in seconds.

SpectralS: the spectral series expansion.
When we set working precision to 30 digits, the accuracy for the cases of \( q > 0.05 \) increases substantially to 26 digits (see Table 4.32). Since the effective digits in the results are 26.1 digits, we can increase the accuracy further by using more working precision. For difficult cases in Table 4.33, we need to set \( M \) to 480 for at least 5-digit accuracy. Note the SpectralS method does not consume many effective digits in the result. This can be seen from that the effective digits in the results of Case 14–16 decreases by about 4 digits when \( M \) increases from 44 to 480 with the same working precision, 30 digits.

It is clear that cases with small value of \( q \) requires larger \( M \) to have the same accuracy.

### Table 4.32. Results of the spectral series expansion with arbitrary precision

<table>
<thead>
<tr>
<th>Case</th>
<th>Reference</th>
<th>( M = 44 ); ( wp = 30 )</th>
<th>( M = 59 ); ( wp = 30 )</th>
<th>( M = 69 ); ( wp = 30 )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accu</td>
<td>Price</td>
<td>CPU</td>
<td>Accu</td>
</tr>
<tr>
<td>1</td>
<td>27</td>
<td>0.3912777093</td>
<td>26.1</td>
<td>26</td>
</tr>
<tr>
<td>2</td>
<td>37</td>
<td>0.248156905</td>
<td>26.1</td>
<td>26</td>
</tr>
<tr>
<td>3</td>
<td>37</td>
<td>0.3062203648</td>
<td>26.1</td>
<td>26</td>
</tr>
<tr>
<td>17</td>
<td>40</td>
<td>0.05598604154</td>
<td>26.1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>38</td>
<td>0.2183754566</td>
<td>26.1</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>39</td>
<td>0.122867410</td>
<td>26.1</td>
<td>11</td>
</tr>
<tr>
<td>33</td>
<td>5.2569522190</td>
<td>26.1</td>
<td>11</td>
<td>0.30919</td>
</tr>
<tr>
<td>26</td>
<td>2.855932016</td>
<td>26.1</td>
<td>26</td>
<td>2.8559</td>
</tr>
<tr>
<td>38</td>
<td>2.310278887</td>
<td>26.1</td>
<td>26</td>
<td>2.3109</td>
</tr>
<tr>
<td>10</td>
<td>38</td>
<td>1.87023661</td>
<td>26.1</td>
<td>26</td>
</tr>
<tr>
<td>20</td>
<td>38</td>
<td>1.875795199</td>
<td>26.1</td>
<td>26</td>
</tr>
<tr>
<td>12</td>
<td>37</td>
<td>0.935426326</td>
<td>26.1</td>
<td>26</td>
</tr>
<tr>
<td>13</td>
<td>38</td>
<td>0.679081920</td>
<td>26.1</td>
<td>26</td>
</tr>
<tr>
<td>14</td>
<td>22</td>
<td>2.697971205</td>
<td>26.1</td>
<td>2</td>
</tr>
<tr>
<td>25</td>
<td>1.314741432</td>
<td>26.1</td>
<td>0</td>
<td>0.7686</td>
</tr>
<tr>
<td>17</td>
<td>5.2823249387</td>
<td>26.1</td>
<td>0</td>
<td>1.1241</td>
</tr>
<tr>
<td>38</td>
<td>8.828758224</td>
<td>26.1</td>
<td>5</td>
<td>8.8288</td>
</tr>
<tr>
<td>19</td>
<td>38</td>
<td>4.690791322</td>
<td>26.1</td>
<td>5</td>
</tr>
</tbody>
</table>

wp: working precision; ED: effective number of significant digits; Accu: accuracy measured by the number of significant digits; CPU: computing time in seconds.

SpectralS: the spectral series expansion.
Table 4.33. Results of the spectral series expansion on difficult cases with enhanced parameter settings

<table>
<thead>
<tr>
<th>Case</th>
<th>Reference</th>
<th>Price</th>
<th>ED</th>
<th>Accu</th>
<th>Price</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>17</td>
<td>0.05988604154</td>
<td>16.8</td>
<td>5</td>
<td>0.059864478</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>12</td>
<td>2.1007871538</td>
<td>21.6</td>
<td>10</td>
<td>2.0979470</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>20</td>
<td>1.159741432</td>
<td>21.6</td>
<td>16</td>
<td>1.159744333</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>17</td>
<td>0.20852249387</td>
<td>21.6</td>
<td>12</td>
<td>0.208532412</td>
<td></td>
</tr>
</tbody>
</table>

wp: working precision; ED: effective number of significant digits; Accu: accuracy measured by the number of significant digits; CPU: computing time in seconds.
SpectralS: the spectral series expansion.

- 4.13. Numerical results of constructive complex analysis

Before implementing the constructive complex analysis, we need to choose the break point $B$ for decomposition of $I_{a,c}(0, \infty)$, the number of terms $N_a$ of the asymptotic expansion, and the number of terms $N_s$ of the series expansion. This is done by minimizing the error estimate of $\gamma_{v,q} I$ with respect to $B$ given fixed $N_a$ and $N_s$ with the working precision equal to 20 decimal digits. First of all, $N_a$ is chosen so that the local error estimate is minimum given some initial $N_s$. Next, $N_s$ is selected so that the local error estimate is below the desired accuracy, i.e., 17 decimal places, given $N_a$ in the first step. After that, we may refine our choice of $N_a$ and $N_s$ by repeating the first two steps until the global minimum of error estimate is achieved. Finally, $B$ is determined by minimizing the error estimate with respect to $B$ given refined $N_a$ and $N_s$. The minimized error estimate is simply the maximal error of $\gamma_{v,q} I$.

Given the desired accuracy for $\gamma_{v,q} I$ is 17 decimal places, we first determine $B$, $N_a$ and $N_s$ for each case. Then, with the determined settings, we obtain the maximal error of $\gamma_{v,q} I$, the maximal error of $\gamma_{v,q} R$, the maximal error of $C^0(h,q)$, and maximal error of $C_I$. From Table 4.34, the maximal error of $\gamma_{v,q} I$ is indeed below 17 decimal places. The maximal error of $\gamma_{v,q} R$ affected by $q$, $v$ and $B$ where $q$ plays a leading role. The accuracy of $\gamma_{v,q} R$ soars to 95 decimal places as $q$ reduces to 0.0025. However, the accuracy is poor when $q$ is large. In seventh case, the accuracy for $\gamma_{v,q} R$ is only one decimal place. The final maximal error of $C_I$ is determined by the largest one between the maximal errors of $\gamma_{v,q} I$ and the maximal errors of $\gamma_{v,q} R$. For example, the maximal error of $\gamma_{v,q} I$ which is chosen to be just below $5 \times 10^{-18}$ exceeds the maximal error of $\gamma_{v,q} R$ for difficult cases. Since the maximal errors of $\gamma_{v,q} R$ is fixed given specific $B$, $N_a$, and $N_s$, constructive complex analysis cannot achieve any level of accuracy.

To use constructive complex analysis for pricing, we need to specify one more parameter $K_0$. The parameter $K_0$ is the number of terms of the summation in the term $A_0, a(a)$ which appears in the
asymptotic expansion of $I_{n,e}(B, \infty)$. Required value of $K_0$ increases when $q$ decreases. With any other parameters fixed, rising $K_0$ does not increase CPU time considerably. Hence, we use $K_0 = 40$ for normal cases, while use $K_0 = 100$ for difficult cases.

To summarize, the maximal error of $\gamma_{v,q} I$ can be controlled, while the maximal error of $\gamma_{v,q} R$ is fixed once $B$, $N_n$, and $N_r$ are determined. Therefore, the accuracy of constructive complex analysis is limited by the maximal error of $\gamma_{v,q} R$. Regarding the accuracy rather than the efficiency, this procedure can obtain good accuracy as $q$ is low, but works poorly as $q$ is large, i.e. $q > 0.05$.

Table 4.34. Results of constructive complex analysis with arbitrary precision

<table>
<thead>
<tr>
<th>Case</th>
<th>$B$</th>
<th>$N_n$</th>
<th>$N_r$</th>
<th>$\text{ME of } Y_{v, q}$</th>
<th>$\text{ME of } Y_{v, q} R$</th>
<th>$\text{ME of } C_1$</th>
<th>$\text{Eff}$</th>
<th>$\text{Acc}$</th>
<th>$\text{Price}$</th>
<th>$\text{CPU}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.5431</td>
<td>19</td>
<td>267</td>
<td>$1.73 \times 10^{-4}$</td>
<td>$3.01 \times 10^{-4}$</td>
<td>$1.00 \times 10^{-4}$</td>
<td>40</td>
<td>30</td>
<td>3.42</td>
<td>3.02</td>
</tr>
<tr>
<td>2</td>
<td>1.5203</td>
<td>19</td>
<td>269</td>
<td>$1.59 \times 10^{-4}$</td>
<td>$3.01 \times 10^{-4}$</td>
<td>$1.00 \times 10^{-4}$</td>
<td>40</td>
<td>30</td>
<td>3.20</td>
<td>3.02</td>
</tr>
<tr>
<td>3</td>
<td>1.5042</td>
<td>19</td>
<td>271</td>
<td>$1.50 \times 10^{-4}$</td>
<td>$3.01 \times 10^{-4}$</td>
<td>$1.00 \times 10^{-4}$</td>
<td>40</td>
<td>30</td>
<td>3.02</td>
<td>3.02</td>
</tr>
<tr>
<td>4</td>
<td>0.8973</td>
<td>145</td>
<td>1921</td>
<td>$7.05 \times 10^{-4}$</td>
<td>$3.66 \times 10^{-4}$</td>
<td>$3.66 \times 10^{-4}$</td>
<td>100</td>
<td>260</td>
<td>20.7</td>
<td>0.055986</td>
</tr>
<tr>
<td>5</td>
<td>1.1586</td>
<td>22</td>
<td>166</td>
<td>$1.14 \times 10^{-4}$</td>
<td>$2.14 \times 10^{-4}$</td>
<td>$2.14 \times 10^{-4}$</td>
<td>40</td>
<td>60</td>
<td>25.8</td>
<td>0.21839</td>
</tr>
<tr>
<td>6</td>
<td>1.2501</td>
<td>23</td>
<td>113</td>
<td>$1.72 \times 10^{-4}$</td>
<td>$1.48 \times 10^{-4}$</td>
<td>$9.20 \times 10^{-4}$</td>
<td>40</td>
<td>50</td>
<td>21.5</td>
<td>0.17272</td>
</tr>
<tr>
<td>7</td>
<td>1.8758</td>
<td>20</td>
<td>272</td>
<td>$1.99 \times 10^{-4}$</td>
<td>$0.0137$</td>
<td>$0.0137$</td>
<td>40</td>
<td>25</td>
<td>5.56</td>
<td>0.23717</td>
</tr>
<tr>
<td>8</td>
<td>1.4542</td>
<td>19</td>
<td>278</td>
<td>$4.90 \times 10^{-4}$</td>
<td>$1.00 \times 10^{-4}$</td>
<td>$0.00260$</td>
<td>40</td>
<td>30</td>
<td>5.56</td>
<td>2.8158</td>
</tr>
<tr>
<td>9</td>
<td>1.4827</td>
<td>19</td>
<td>277</td>
<td>$1.44 \times 10^{-4}$</td>
<td>$1.92 \times 10^{-4}$</td>
<td>$1.92 \times 10^{-4}$</td>
<td>40</td>
<td>30</td>
<td>5.76</td>
<td>3.5044</td>
</tr>
<tr>
<td>10</td>
<td>1.5076</td>
<td>19</td>
<td>275</td>
<td>$1.46 \times 10^{-4}$</td>
<td>$3.42 \times 10^{-4}$</td>
<td>$3.42 \times 10^{-4}$</td>
<td>40</td>
<td>30</td>
<td>5.76</td>
<td>4.8789</td>
</tr>
<tr>
<td>11</td>
<td>1.4975</td>
<td>19</td>
<td>274</td>
<td>$1.56 \times 10^{-4}$</td>
<td>$8.81 \times 10^{-4}$</td>
<td>$8.81 \times 10^{-4}$</td>
<td>40</td>
<td>30</td>
<td>6.19</td>
<td>2.8986</td>
</tr>
<tr>
<td>12</td>
<td>1.5091</td>
<td>19</td>
<td>273</td>
<td>$1.43 \times 10^{-4}$</td>
<td>$0.000129$</td>
<td>$0.000129$</td>
<td>40</td>
<td>30</td>
<td>6.22</td>
<td>6.9339</td>
</tr>
<tr>
<td>13</td>
<td>1.5239</td>
<td>19</td>
<td>271</td>
<td>$1.85 \times 10^{-4}$</td>
<td>$0.000184$</td>
<td>$0.000184$</td>
<td>40</td>
<td>30</td>
<td>6.45</td>
<td>6.0695</td>
</tr>
<tr>
<td>14</td>
<td>0.8359</td>
<td>420</td>
<td>312</td>
<td>$1.87 \times 10^{-11}$</td>
<td>$7.36 \times 10^{-11}$</td>
<td>$7.36 \times 10^{-11}$</td>
<td>100</td>
<td>235</td>
<td>19.4</td>
<td>2.6979</td>
</tr>
<tr>
<td>15</td>
<td>0.8979</td>
<td>123</td>
<td>1726</td>
<td>$1.33 \times 10^{-11}$</td>
<td>$5.67 \times 10^{-11}$</td>
<td>$5.33 \times 10^{-11}$</td>
<td>100</td>
<td>235</td>
<td>20.4</td>
<td>1.1347</td>
</tr>
<tr>
<td>16</td>
<td>0.9390</td>
<td>123</td>
<td>1731</td>
<td>$1.23 \times 10^{-11}$</td>
<td>$7.78 \times 10^{-11}$</td>
<td>$4.23 \times 10^{-11}$</td>
<td>100</td>
<td>235</td>
<td>22.2</td>
<td>0.28532</td>
</tr>
<tr>
<td>17</td>
<td>1.1081</td>
<td>25</td>
<td>364</td>
<td>$4.30 \times 10^{-11}$</td>
<td>$1.04 \times 10^{-11}$</td>
<td>$4.22 \times 10^{-11}$</td>
<td>40</td>
<td>60</td>
<td>25.1</td>
<td>14.884</td>
</tr>
<tr>
<td>18</td>
<td>1.1594</td>
<td>25</td>
<td>363</td>
<td>$4.30 \times 10^{-11}$</td>
<td>$1.50 \times 10^{-11}$</td>
<td>$6.10 \times 10^{-11}$</td>
<td>40</td>
<td>60</td>
<td>25.8</td>
<td>8.8288</td>
</tr>
<tr>
<td>19</td>
<td>1.2014</td>
<td>24</td>
<td>359</td>
<td>$5.00 \times 10^{-11}$</td>
<td>$1.35 \times 10^{-11}$</td>
<td>$5.49 \times 10^{-11}$</td>
<td>40</td>
<td>60</td>
<td>26.3</td>
<td>4.6967</td>
</tr>
</tbody>
</table>

wp: working precision; ED: effective number of significant digits; Accu: accuracy measured by the number of significant digits; CPU: computing time in seconds.

It is essential to use multi-precision when implement constructive complex analysis. If we choose machine precision, this approach fails completely as shown in Table 4.35.
Table 4.35. Results of constructive complex analysis with machine precision

<table>
<thead>
<tr>
<th>Case</th>
<th>Reference</th>
<th>Accu</th>
<th>Price</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>38</td>
<td>0.1931737903</td>
<td>1.5654</td>
<td>19</td>
</tr>
<tr>
<td>2</td>
<td>37</td>
<td>0.4264156905</td>
<td>1.5235</td>
<td>19</td>
</tr>
<tr>
<td>3</td>
<td>37</td>
<td>0.3062203648</td>
<td>1.5028</td>
<td>19</td>
</tr>
<tr>
<td>4</td>
<td>17</td>
<td>0.05598604154</td>
<td>0.8973</td>
<td>149</td>
</tr>
<tr>
<td>5</td>
<td>38</td>
<td>0.2183875466</td>
<td>1.1586</td>
<td>22</td>
</tr>
<tr>
<td>6</td>
<td>39</td>
<td>0.1722687419</td>
<td>1.2531</td>
<td>23</td>
</tr>
<tr>
<td>7</td>
<td>30</td>
<td>0.3500952190</td>
<td>1.8758</td>
<td>20</td>
</tr>
<tr>
<td>8</td>
<td>36</td>
<td>2.815862016</td>
<td>1.4542</td>
<td>19</td>
</tr>
<tr>
<td>9</td>
<td>38</td>
<td>2.310878887</td>
<td>1.4827</td>
<td>19</td>
</tr>
<tr>
<td>10</td>
<td>38</td>
<td>1.879023661</td>
<td>1.5076</td>
<td>19</td>
</tr>
<tr>
<td>11</td>
<td>36</td>
<td>7.895795199</td>
<td>1.4917</td>
<td>19</td>
</tr>
<tr>
<td>12</td>
<td>37</td>
<td>6.935422632</td>
<td>1.5091</td>
<td>19</td>
</tr>
<tr>
<td>13</td>
<td>36</td>
<td>6.070987190</td>
<td>1.5239</td>
<td>19</td>
</tr>
<tr>
<td>14</td>
<td>22</td>
<td>2.697871938</td>
<td>0.8598</td>
<td>122</td>
</tr>
<tr>
<td>15</td>
<td>20</td>
<td>1.134741432</td>
<td>0.8997</td>
<td>123</td>
</tr>
<tr>
<td>16</td>
<td>17</td>
<td>0.2853249387</td>
<td>0.9345</td>
<td>123</td>
</tr>
<tr>
<td>17</td>
<td>38</td>
<td>14.98395833</td>
<td>1.1081</td>
<td>25</td>
</tr>
<tr>
<td>18</td>
<td>37</td>
<td>8.828758224</td>
<td>1.1594</td>
<td>25</td>
</tr>
<tr>
<td>19</td>
<td>38</td>
<td>4.696709132</td>
<td>1.2014</td>
<td>24</td>
</tr>
</tbody>
</table>

wp: working precision; ED: effective number of significant digits; Accu: accuracy measured by the number of significant digits; CPU: computing time in seconds.


Turnbull and Wakeman (1991) method (TW), Milevsky and Posner (1998) reciprocal gamma approximation (RG), Večř (2002) PDE method (PDE), and Shaw (2002) asymptotic method (a 13-term asymptotic series is used by Shaw) are compared together in Table 4.36 because they yield a result of fixed accuracy. RG method and TW method are the quickest procedures. But they perform poorly: get about 2-digit accuracy. PDE method performs better and have around 3-digit accuracy within 0.1 CPU second. With the setting of truncation size \( x = 50000 \) and the "MaxRecursion" option equal to 18 in built-in Mathematica function "NIntegrate", asymptotic method in general gives accuracy of 3 digits and takes about 2 CPU seconds.
<table>
<thead>
<tr>
<th>Case</th>
<th>Reference</th>
<th>Reference</th>
<th>TW</th>
<th>TW</th>
<th>TW</th>
<th>PDE</th>
<th>PDE</th>
<th>Asymptotic Method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accu</td>
<td>Price</td>
<td>CPU</td>
<td>Accu</td>
<td>Price</td>
<td>CPU</td>
<td>Accu</td>
<td>Price</td>
</tr>
<tr>
<td>1</td>
<td>38</td>
<td>0.1931737903</td>
<td></td>
<td>2</td>
<td>0.19558</td>
<td>0</td>
<td>2</td>
<td>0.19598</td>
</tr>
<tr>
<td>2</td>
<td>37</td>
<td>0.2464156905</td>
<td></td>
<td>2</td>
<td>0.24974</td>
<td>0</td>
<td>1</td>
<td>0.24343</td>
</tr>
<tr>
<td>3</td>
<td>27</td>
<td>0.5062203648</td>
<td></td>
<td>2</td>
<td>0.31063</td>
<td>0</td>
<td>1</td>
<td>0.32259</td>
</tr>
<tr>
<td>4</td>
<td>17</td>
<td>0.0058654134</td>
<td></td>
<td>2</td>
<td>0.00586</td>
<td>0</td>
<td>2</td>
<td>0.00586</td>
</tr>
<tr>
<td>5</td>
<td>38</td>
<td>0.2138560449</td>
<td></td>
<td>2</td>
<td>0.41393</td>
<td>0</td>
<td>2</td>
<td>0.41422</td>
</tr>
<tr>
<td>6</td>
<td>29</td>
<td>0.1722687410</td>
<td></td>
<td>2</td>
<td>0.17349</td>
<td>0.02</td>
<td>2</td>
<td>0.17121</td>
</tr>
<tr>
<td>7</td>
<td>30</td>
<td>0.3500952190</td>
<td></td>
<td>1</td>
<td>0.35920</td>
<td>0</td>
<td>0</td>
<td>0.34179</td>
</tr>
<tr>
<td>8</td>
<td>34</td>
<td>2.8158600104</td>
<td></td>
<td>1</td>
<td>2.8846</td>
<td>0</td>
<td>2</td>
<td>2.8647</td>
</tr>
<tr>
<td>9</td>
<td>38</td>
<td>2.310878887</td>
<td></td>
<td>2</td>
<td>2.3413</td>
<td>0</td>
<td>2</td>
<td>2.2843</td>
</tr>
<tr>
<td>10</td>
<td>38</td>
<td>1.890236684</td>
<td></td>
<td>2</td>
<td>1.9411</td>
<td>0</td>
<td>2</td>
<td>1.9534</td>
</tr>
<tr>
<td>11</td>
<td>38</td>
<td>2.895795193</td>
<td></td>
<td>1</td>
<td>2.8950</td>
<td>0</td>
<td>1</td>
<td>2.8042</td>
</tr>
<tr>
<td>12</td>
<td>37</td>
<td>6.9354226312</td>
<td></td>
<td>1</td>
<td>7.0307</td>
<td>0</td>
<td>2</td>
<td>6.8522</td>
</tr>
<tr>
<td>13</td>
<td>38</td>
<td>6.070987190</td>
<td></td>
<td>2</td>
<td>6.1491</td>
<td>0</td>
<td>1</td>
<td>6.998</td>
</tr>
<tr>
<td>14</td>
<td>22</td>
<td>2.697871538</td>
<td></td>
<td>3</td>
<td>2.6997</td>
<td>0</td>
<td>3</td>
<td>2.696</td>
</tr>
<tr>
<td>15</td>
<td>20</td>
<td>1.134741432</td>
<td></td>
<td>2</td>
<td>1.1368</td>
<td>0</td>
<td>3</td>
<td>1.1333</td>
</tr>
<tr>
<td>16</td>
<td>17</td>
<td>0.285244938</td>
<td></td>
<td>1</td>
<td>0.3244</td>
<td>0</td>
<td>2</td>
<td>0.265</td>
</tr>
<tr>
<td>17</td>
<td>38</td>
<td>14.98301903</td>
<td></td>
<td>2</td>
<td>15.057</td>
<td>0</td>
<td>2</td>
<td>14.954</td>
</tr>
<tr>
<td>18</td>
<td>37</td>
<td>8.828758224</td>
<td></td>
<td>1</td>
<td>8.8858</td>
<td>0</td>
<td>2</td>
<td>8.192</td>
</tr>
<tr>
<td>19</td>
<td>38</td>
<td>4.696709132</td>
<td></td>
<td>3</td>
<td>4.6951</td>
<td>0.02</td>
<td>2</td>
<td>4.6908</td>
</tr>
</tbody>
</table>

**Table 4.36. Results and comparison of TW, RG, PDE and asymptotic method with machine precision**

**wp**: working precision; **ED**: effective number of significant digits; **Accu**: accuracy measured by the number of significant digits; **CPU**: computing time in seconds. **TW**: the Turnbull and Wakeman (1991) approximation. **RG**: the Milevsky and Posner (1998) approximation. **PDE**: the PDE method.

### 4.15. Numerical results of Monte Carlo simulation

Monte Carlo simulation involves the settings of two parameters: daily reading and the replication. Setting daily reading equal to 1 means sample once each day. Since we compute the price for continuous Asian options, it causes discretization bias when simulate discretely. But using the price of an geometric Asian option as the control variate, it can effectively offset the discretization bias. Increasing the replication size helps reduce the standard error of the simulated result. Meanwhile, the simulation time increases proportionally with the replication size. As shown in Table 4.37, using 100 000 replications takes an order of magnitude more time than using 10 000 replications. However the accuracy increases by one digit in general. With the same replications, long maturity takes more computing time than short maturity. The value of \( q \) does not matter the accuracy and efficiency of the simulation. The price obtained by Monte Carlo simulation is just one realization. The value changes when we repeat the simulation every time.
Table 4.37. Results of Monte Carlo simulation

<table>
<thead>
<tr>
<th>Case</th>
<th>T</th>
<th>q</th>
<th>Reference Price</th>
<th>Replication = 10,000 Price</th>
<th>SE</th>
<th>CPU</th>
<th>Replication = 100,000 Price</th>
<th>SE</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1.0</td>
<td>0.6679</td>
<td>0.19317</td>
<td>0.192877</td>
<td>0.000271</td>
<td>14.4</td>
<td>3</td>
<td>0.193844</td>
<td>0.000859</td>
</tr>
<tr>
<td>2</td>
<td>1.0</td>
<td>0.6679</td>
<td>0.19317</td>
<td>0.192877</td>
<td>0.000271</td>
<td>14.4</td>
<td>3</td>
<td>0.193844</td>
<td>0.000859</td>
</tr>
<tr>
<td>3</td>
<td>1.0</td>
<td>0.6679</td>
<td>0.19317</td>
<td>0.192877</td>
<td>0.000271</td>
<td>14.4</td>
<td>3</td>
<td>0.193844</td>
<td>0.000859</td>
</tr>
<tr>
<td>4</td>
<td>1.0</td>
<td>0.6679</td>
<td>0.19317</td>
<td>0.192877</td>
<td>0.000271</td>
<td>14.4</td>
<td>3</td>
<td>0.193844</td>
<td>0.000859</td>
</tr>
<tr>
<td>5</td>
<td>1.0</td>
<td>0.6679</td>
<td>0.19317</td>
<td>0.192877</td>
<td>0.000271</td>
<td>14.4</td>
<td>3</td>
<td>0.193844</td>
<td>0.000859</td>
</tr>
<tr>
<td>6</td>
<td>1.0</td>
<td>0.6679</td>
<td>0.19317</td>
<td>0.192877</td>
<td>0.000271</td>
<td>14.4</td>
<td>3</td>
<td>0.193844</td>
<td>0.000859</td>
</tr>
<tr>
<td>7</td>
<td>1.0</td>
<td>0.6679</td>
<td>0.19317</td>
<td>0.192877</td>
<td>0.000271</td>
<td>14.4</td>
<td>3</td>
<td>0.193844</td>
<td>0.000859</td>
</tr>
<tr>
<td>8</td>
<td>1.0</td>
<td>0.6679</td>
<td>0.19317</td>
<td>0.192877</td>
<td>0.000271</td>
<td>14.4</td>
<td>3</td>
<td>0.193844</td>
<td>0.000859</td>
</tr>
<tr>
<td>9</td>
<td>1.0</td>
<td>0.6679</td>
<td>0.19317</td>
<td>0.192877</td>
<td>0.000271</td>
<td>14.4</td>
<td>3</td>
<td>0.193844</td>
<td>0.000859</td>
</tr>
<tr>
<td>10</td>
<td>1.0</td>
<td>0.6679</td>
<td>0.19317</td>
<td>0.192877</td>
<td>0.000271</td>
<td>14.4</td>
<td>3</td>
<td>0.193844</td>
<td>0.000859</td>
</tr>
<tr>
<td>11</td>
<td>1.0</td>
<td>0.6679</td>
<td>0.19317</td>
<td>0.192877</td>
<td>0.000271</td>
<td>14.4</td>
<td>3</td>
<td>0.193844</td>
<td>0.000859</td>
</tr>
<tr>
<td>12</td>
<td>1.0</td>
<td>0.6679</td>
<td>0.19317</td>
<td>0.192877</td>
<td>0.000271</td>
<td>14.4</td>
<td>3</td>
<td>0.193844</td>
<td>0.000859</td>
</tr>
<tr>
<td>13</td>
<td>1.0</td>
<td>0.6679</td>
<td>0.19317</td>
<td>0.192877</td>
<td>0.000271</td>
<td>14.4</td>
<td>3</td>
<td>0.193844</td>
<td>0.000859</td>
</tr>
<tr>
<td>14</td>
<td>1.0</td>
<td>0.6679</td>
<td>0.19317</td>
<td>0.192877</td>
<td>0.000271</td>
<td>14.4</td>
<td>3</td>
<td>0.193844</td>
<td>0.000859</td>
</tr>
<tr>
<td>15</td>
<td>1.0</td>
<td>0.6679</td>
<td>0.19317</td>
<td>0.192877</td>
<td>0.000271</td>
<td>14.4</td>
<td>3</td>
<td>0.193844</td>
<td>0.000859</td>
</tr>
<tr>
<td>16</td>
<td>1.0</td>
<td>0.6679</td>
<td>0.19317</td>
<td>0.192877</td>
<td>0.000271</td>
<td>14.4</td>
<td>3</td>
<td>0.193844</td>
<td>0.000859</td>
</tr>
<tr>
<td>17</td>
<td>1.0</td>
<td>0.6679</td>
<td>0.19317</td>
<td>0.192877</td>
<td>0.000271</td>
<td>14.4</td>
<td>3</td>
<td>0.193844</td>
<td>0.000859</td>
</tr>
<tr>
<td>18</td>
<td>1.0</td>
<td>0.6679</td>
<td>0.19317</td>
<td>0.192877</td>
<td>0.000271</td>
<td>14.4</td>
<td>3</td>
<td>0.193844</td>
<td>0.000859</td>
</tr>
<tr>
<td>19</td>
<td>1.0</td>
<td>0.6679</td>
<td>0.19317</td>
<td>0.192877</td>
<td>0.000271</td>
<td>14.4</td>
<td>3</td>
<td>0.193844</td>
<td>0.000859</td>
</tr>
</tbody>
</table>

wp: working precision; ED: effective number of significant digits; Accu: accuracy measured by the number of significant digits; SE: standard error; CPU: computing time in seconds.
5. Conclusion

In this chapter, we have solved the problem of computing the price of a continuous arithmetic Asian call option - the case of greatest relevance to both traders and the academic literature - through the explicit use of multi-precision arithmetic supported by the Mathematica platform. The problem has endured because, until now, every method proposed has been beset by some problems in some region of the parameter space. We have shown that the source of the problem in the literature, in every case, is the use of fixed machine-precision, with the consequence that, as more and more accuracy is sought, there comes a point where round-off error propagates to destroy the accuracy of computations in at least some regions of the parameter space. Here, using a multi-precision environment, we have shown that all Laplace transform inversion algorithms and the algorithm based on spectral series expansion can achieve any desired accuracy provided we choose the right parameter values and use appropriate precision.

The performance of the methods is summarized in Table 5.1 with CPU time collected from above tables. Nineteen Asian option cases are divided into two groups: normal cases for $0.02025 \leq q \leq 0.125$ and difficult cases for $0.0025 \leq q \leq 0.00301$. Table 5.1 gives computing time spent for achieving accuracy of at least 5, 10 and 15 significant digits respectively. For difficult cases, CPU time for at least 5-digit accuracy is reported.

For normal cases, the best methods for pricing an Asian option are the UniT method and the FT method which use the least CPU time and have only one free parameter. In fact, the UniT method and the FT method are very much alike since both of them are based on the Talbot algorithm. The third best and fourth best methods are the UniE method and Euler method. The UniE method performs slightly better than the Euler method although the fundamental algorithms used are the same, i.e. the Euler algorithm. In addition, the UniE method is superior to the Euler method because the former uses only one free parameter while the latter uses three free parameters. Considering the CPU time, Laguerre method in general outperforms GWR method. However it is much easier to apply GWR method as Laguerre method requires selection of many parameters. The slowest methods are Bromwich integral, the SpectralS method and the PW method. It is surprising to see that the PW method outperforms the other two although the PW method spends the most time in computing the difficult cases.

Regarding the difficult cases, the situation changes. The best methods are the UniE method and Euler method using around 10 seconds to have 5-digit accuracy. The third best is the UniG method consuming
about 13 seconds. The UniT method and FT method use slightly more time, between 17 and 22 seconds.

The PW method, the SpectralS method and Bromwich integral are impractical when they are used in cases with low $q$.

<table>
<thead>
<tr>
<th>Method</th>
<th>0.02025 ≤ $q$ ≤ 0.125</th>
<th>0.0025 ≤ $q$ ≤ 0.00301</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accu</td>
<td>CPU</td>
</tr>
<tr>
<td>UniT</td>
<td>5</td>
<td>0.03–0.17</td>
</tr>
<tr>
<td>FT</td>
<td>5</td>
<td>0.03–0.19</td>
</tr>
<tr>
<td>UniE</td>
<td>5</td>
<td>0.03–0.25</td>
</tr>
<tr>
<td>Euler</td>
<td>5</td>
<td>0.06–0.37</td>
</tr>
<tr>
<td>UniG</td>
<td>5</td>
<td>0.06–0.23</td>
</tr>
<tr>
<td>Laguerre</td>
<td>5</td>
<td>0.41–1.44</td>
</tr>
<tr>
<td>GWR</td>
<td>5</td>
<td>0.19–0.78</td>
</tr>
<tr>
<td>PW</td>
<td>5</td>
<td>1.09–11.8</td>
</tr>
<tr>
<td>SpectralS</td>
<td>5</td>
<td>6.83–7.43</td>
</tr>
<tr>
<td>Bromwich</td>
<td>5</td>
<td>11.1–42.0</td>
</tr>
</tbody>
</table>

wp: working precision; ED: effective number of significant digits; Accu: accuracy measured by the number of significant digits; SE: standard error; CPU: computing time in seconds.


To conclude, the UniT method and the FT method work best for $q ≥ 0.02$ while the UniE method and Euler method perform well when $q$ is low. The computing time used by the SpectralS method is not sensitive to $q$ because the reported time range is marginal. The TW method and the RG method work poorly in pricing an Asian option. PDE method and asymptotic method performs slightly better. But the yielded accuracy is still unsatisfactory and can not meet the traders' need. The constructive complex analysis yields accurate price when $q$ is low, but get inaccurate result when $q$ is large. Monte Carlo simulation can be implemented with ease but it may be computationally expensive without enhanced variance reduction techniques.
Chapter 3

A Comparison of Various Methods for Computing Stable Distributions and Infinitely Divisible Distributions Associated with Hyperbolic Functions

1. Introduction

In Chapter 2, we were able to solve the computational problem of computing the continuous arithmetic-average Asian option price through the explicit use of multi-precision arithmetic supported by the Mathematica platform. This mostly involved either the robust numerical inversion of a Laplace transform (the Geman-Yor Laplace transform) or careful use of a series expansion. In this chapter, we apply the same techniques to other sets of problems - this time in the Statistics literature rather than the Finance literature - that are challenging from a computational point of view. Specifically, we shall look at the computation and simulation of stable densities and distributions, and of infinitely divisible distributions associated with hyperbolic functions. The former problem has been historically challenging and some computational issues remain unresolved to this day, while the latter problem has been of
interest more recently and involves densities and distributions that can be related to the unit root densities and distributions that we consider in Chapter 4. We are again able to find contribution and success in computing and simulating the densities in both types of problems by explicitly operating in a multi-precision computing environment.

Pitman and Yor (2003) define the infinitely divisible distributions of non-negative random variables $C_t$, $S_t$ and $T_t$ via Laplace transforms, for $t > 0$

\[
E[e^{-\lambda C_t}] = \left(\frac{1}{\cosh \sqrt{2} \lambda}\right)^t
\]

\[
E[e^{-\lambda S_t}] = \left(\frac{\sqrt{2} \lambda}{\sinh \sqrt{2} \lambda}\right)^t
\]

\[
E[e^{-\lambda T_t}] = \left(\frac{\tanh \sqrt{2} \lambda}{\sqrt{2} \lambda}\right)^t
\]

Series expansions for the densities $f_S(x)$ and $f_{C_t}(x)$ were derived by Biane and Yor (1987, BY) and Biane, Pitman and Yor (2001, BPY) respectively. Other analytical formulae exist in the literature for certain special cases. As shown by Devroye (2009a), formulae for the densities of $C_1$ and $S_1$ relate to Jacobi theta functions. Tolmatz (2002) obtains a formula in terms of the parabolic cylinder function for the density of $S_{1/2}$. There is currently no available analytical expression for the density of $T_t$ other than for $t = 1$. Here, we wish to address a current gap in the literature: so far, numerical inversion of the Laplace transforms in (1.1), (1.2) and (1.3) has not been studied and evaluated as a method of computing or approximating the relevant densities.

The stable distributions which are first introduced by Levy (1925 cited in Borak, Härdle and Weron, 2005) have the property that a sum of two independent stable random variables with index $\alpha$ is again stable with the same index $\alpha$. The stable distributions have attracted increasing attentions since they can capture the fat tails and asymmetry and give a better fit to empirical data especially the data sets with extreme events such as economic crisis and natural disasters. Zolotarev (1986) wrote a monograph that discusses the basic properties of one-dimensional stable distributions.

The stable distribution has no closed-form formula for its density in general and is most conveniently described by its characteristic function. There are notable exceptions we shall discuss below. Owing to different parameterizations of the characteristic function, there are various types of stable distributions.
Two of them are denoted by $S(\alpha, \beta, \mu, \sigma)$ with its characteristic function (Weron, 1996)

$$\log \phi(t) = \begin{cases} 
-\sigma^n |t|^\alpha \left(1 - i \beta \text{sign}(t) \tan \frac{\pi \alpha}{2}\right) + i \mu t, & \alpha \neq 1, \\
-\sigma |t| \left(1 + i \beta \text{sign}(t) \frac{\pi}{2} \ln |t|\right) + i \mu t, & \alpha = 1, 
\end{cases}$$

(1.4)

and $S_1(\alpha, \beta_1, \mu, \sigma_1)$ with its characteristic function (Weron, 1996)

$$\log \phi(t) = \begin{cases} 
-\sigma_1^n |t|^\alpha \exp \left(-i \beta_1 \text{sign}(t) \frac{\pi}{2} K(\alpha)\right) + i \mu t, & \alpha \neq 1, \\
-\sigma_1 |t| \left(\frac{\pi}{2} + i \beta_1 \text{sign}(t) \ln |t|\right) + i \mu t, & \alpha = 1, 
\end{cases}$$

(1.5)

where $\alpha \in (0, 2], \beta \in [-1, 1], \beta_1 \in [-1, 1], \mu \in \mathbb{R}, \sigma > 0, \sigma_1 > 0$ and

$$K(\alpha) = 1 - |1 - \alpha| = \begin{cases} 
\alpha, & 0 < \alpha < 1 \\
2 - \alpha, & 1 < \alpha \leq 2.
\end{cases}$$

(1.6)

Schneider (1987) discusses alternative stable distributions in types of the generalized one-sided stable distribution $F_{m,\alpha}$ with $m$ a positive integer, one-sided stable distribution $F_{\alpha}$ and two-sided stable distribution $F_{\alpha,\beta}$ which can be represented by the Mellin transform, the Laplace transform and the Fourier transform respectively. In particular, the Laplace transform of $F_{\alpha}$ with $f_{\alpha}$ its density is given by

$$F_{\alpha}(x) = \int_0^\infty e^{-x t} f_{\alpha}(t) \, dt = e^{-\lambda^n}$$

(1.7)

One of the sources of the computational difficulty in computing or simulating $f_{\alpha}(x)$ in (1.7) is that its form (and the numerical properties of the Laplace transform inversion) is very sensitive to changes in $\alpha$. Chambers, Mallows and Stuck (1976) propose a recipe for simulating the stable random variable $S_1(\alpha, \beta_1, 0, 1)$ from a combination of the exponential distribution and the uniform distribution.

Schneider (1987) suggests analytical formulae for the densities of $F_{m,\alpha}, F_{\alpha}$ and $F_{\alpha,\beta}$ in terms of the Fox functions or their series expansions. In addition, Schneider converts the Mellin transform of $F_{m,\alpha}$ to the Laplace transform of $F_{m,\alpha}$ which has the Fox function representation for general integer $m$ and the closed-form expression in the case of $m = 1, 2$. Penson and Gorska (2011) obtain exact and explicit expressions for the density of $F_{\alpha}(x)$ for all $\alpha = l/k < 1$ with $k$ and $l$ positive integers. Feller (1970, p.583) derives a series expansion for the density of $F_{\alpha,\beta}$. Besides, Mathematica 9.0 has a built-in function called

`In this chapter, we investigate and compare various methods for computing the densities of infinitely divisible distributions associated with hyperbolic functions, and stable distributions. These methods...`
include Monte Carlo simulation, the careful use of analytical formulae, and numerical inversion of the Laplace transform. All methods are made as robust as they can be through the careful use of multi-precision arithmetic. In addition, this chapter attempts to collect the currently scattered results in the literature on the closed-form representation of stable densities in certain special cases. And, using Mathematica 9.0, we provide a new closed-form expression for the density of $S_t$ as given implicitly in (1.2) above. One of the contributions of this chapter is in providing ready-to-use Mathematica code that implements the methods and algorithms we consider.

The rest of this chapter is organized as follows. Section 2 simulates the Brownian motion, Brownian bridge and Brownian excursion in which the laws of $C_t$ and $S_t$ arise naturally. Section 3 reviews the infinitely divisible distributions associated with hyperbolic functions, and collects all analytical formulae for the densities of $C_t$, $S_t$ and $T_t$. Section 4 shows numerical experiments with the infinitely divisible distributions of $C_t$, $S_t$, and $T_t$. Section 5 identifies different types of stable distributions and gathers the methods for their densities. Then, numerical experiments with stable distributions are conducted in Section 6. Finally, the conclusion of this chapter is drawn in Section 7.
2. Simulation of Brownian Motion, Bridge and Excursion

2.1. Brownian motion

Standard Brownian motion $B(t)$ over $[0, T]$ is also called a Wiener process which is a continuous-time stochastic process. It is characterized by the following properties (e.g. Higham, 2001)

1. $B(0) = 0$ with probability 1.
2. $B(t)$ is continuous.
3. $B(t + s) - B(t)$ is normally distributed with mean zero and variance $s$. For example, $B(t + s) - B(t) \sim \sqrt{s} \ N(0, 1)$ where $N(0, 1)$ is a random variable from a standard normal distribution.
4. For $0 \leq s < t \leq u < v \leq T$, the increments $B(t) - B(s)$ and $B(v) - B(u)$ are independent.

To simulate a continuous Brownian motion $B(t)$ over time $t \in [0, T]$, Higham (2001) discretized the time $t$ by setting $t_j = j \ dt$ with $dt = T / n$ where $n$ is the number of sampling. According to the properties of Brownian motion, we have

$$
B(0) = 0 \\
B(j) = B(j - 1) + dB(j)
$$

(2.1)

where $dB(j)$, $j = 1, 2, ..., n$ are i.i.d. $N(0, dt)$ random variables. The realization of $B(t)$, thus, can be computed recursively with $n$ sampling points. This method is intuitive and straightforward. But, the computation time soars as sampling size increases. We can make the computation more elegant and efficient by using vectorization. First, generate a list of $n$ independent $N(0, dt)$ random variates, and add zero to the list such that the list becomes $[0, dB(1), ..., dB(n)]$. Then, compute the cumulative sum of each element in the list so that we obtain a new list of $[0, dB(1), dB(1) + dB(2), ..., \sum_{j=1}^{n} dB(j)]$. Thus, the new list is a realization of Brownian motion over $[0, T]$ with $n$ sampling points. The time corresponding to the discretized Brownian motion is $[0, t_1, ..., t_{n-1}, T]$.

Drawing a line through points $B(0)$, $B(t_1)$, ..., $B(t_{n-1})$ and $B(T)$ generates a one-dimensional Brownian motion path. Repeating this process $m$ times over the same time interval yields $m$ independent Brownian motion paths. The one-dimensional Brownian motion path can be easily extended to two-dimensional space and three-dimensional space by generating independent Brownian motion in each plane. More specifically, in a three-dimensional space, each point has three coordinates. Thus, we need three independent Brownian motions to construct the coordinates of each point. Figure 2.1 shows Brownian motion paths over $[0, 1]$ with $10^4$ sampling points in different dimensions.
Figure 2.1. Left plots: a single realization of Brownian motion over $[0, T]$ in different dimensions; Right plots: three realizations of Brownian motion over $[0, T]$ in different dimensions. Sampling size: $10^4$.

2.2. Brownian bridge

Brownian bridge $B^{br,x}$ is defined as a Brownian motion $B(t)$ over time $t \in [0, 1]$ conditioned on $B(1) = x$. Pitman (1999) wrote the definition simply as

$$B^{br,x} \overset{d}{=} (B \mid B(1) = r)$$

where $\overset{d}{=}$ represents equality in distribution. When $B(1) = 0$, the Brownian bridge is defined as the standard Brownian motion.

$$B^{br} := B^{br,0}$$

It is well-known that the Brownian bridge $B^{br,x}$ can be constructed from the Brownian motion $B$.

$$B^{br,x}(t) := B(t) + t (x - B(1)), \quad 0 \leq t \leq 1$$

Figure 2.2 plots the standard Brownian bridge paths over $[0, 1]$ with $10^4$ sampling points in different dimensions.
Figure 2.2. Left plots: a single realization of standard Brownian bridge over [0, 1] in different dimensions; Right plots: three realizations of standard Brownian bridge over [0, 1] in different dimensions. Sampling size: $10^4$.

- 2.3. Brownian meander and excursion

Standard Brownian meander $B_{me}$ is defined on time $t \in [0, 1]$ as

$$B_{me} \overset{d}{=} (B \mid B(t) > 0 \text{ for all } 0 < t < 1)$$

While, restricted Brownian meander $B_{me,r}$ is the standard Brownian meander conditioned on the endpoint $r \geq 0$.

$$B_{me,r} \overset{d}{=} (B \mid B(t) > 0 \text{ for all } 0 < t < 1, B(1) = r)$$

A restricted Brownian meander $B_{me,r}$ can be generated by three independent standard Brownian bridges $B_{i,t}^{br}$, $i = 1, 2, 3$ (see Pitman, 2009).

$$B_{me,r}(t) \overset{d}{=} \sqrt{r^2 t + B_{1,t}^{br}(t)^2 + (B_{2,t}^{br}(t))^2 + (B_{3,t}^{br}(t))^2}$$
The standard Brownian meander \( B^{mc} \) can then be obtained from the restricted Brownian meander \( B^{mc,r} \) by setting \( r = \sqrt{2E} \) where \( E \) is a standard exponential random variable (see Devroye, 2009b).

Standard Brownian excursion \( B^{ex} \) is defined as a restricted Brownian meander \( B^{mc,r} \) with \( r \) equal to 0.

\[
B^{ex} := B^{mc,0}
\]

Figure 2.3 draws the standard Brownian excursion paths over \([0, 1]\) with \(10^4\) sampling points in different dimensions.

While the concepts of Brownian motion, Brownian bridge, Brownian meander, and Brownian excursion can often underlie the distributions we now focus on, we are more interested in the computation of their distributions and density functions, rather than seeing where they came from in the setting of a stochastic process. They were introduced here to allow us to refer to them from time to time, if need be.
3. Infinitely Divisible Distributions Associated with Hyperbolic Functions

- 3.1. Definition of distributions

Following Pitman and Yor (2003), non-negative random variables \( C_t, S_t \) and \( T_t \) which have infinitely divisible distributions are characterized by their Laplace transforms. For \( t > 0 \),

\[
E[e^{-\lambda C_t}] = \int_0^\infty e^{-\lambda x} f_{C_t}(x) \, dx = \left( \frac{1}{\cosh \sqrt{2\lambda}} \right)^t \\
E[e^{-\lambda S_t}] = \int_0^\infty e^{-\lambda x} f_{S_t}(x) \, dx = \left( \frac{\sqrt{2\lambda}}{\sinh \sqrt{2\lambda}} \right)^t \\
E[e^{-\lambda T_t}] = \int_0^\infty e^{-\lambda x} f_{T_t}(x) \, dx = \left( \frac{\tanh \sqrt{2\lambda}}{\sqrt{2\lambda}} \right)^t 
\]

(3.1)  (3.2)  (3.3)

We also consider the processes \( \hat{C}_t, \hat{S}_t, \) and \( \hat{T}_t \), characterized by related characteristic functions. For \( t \geq 0 \) and \( \theta \in \mathbb{R} 
\]

\[
E[\exp(i \theta \hat{C}_t)] = E[\exp(-\frac{1}{2} \theta^2 C_t)] = \left( \frac{1}{\cosh \theta} \right)^t \\
E[\exp(i \theta \hat{S}_t)] = E[\exp(-\frac{1}{2} \theta^2 S_t)] = \left( \frac{\theta}{\sinh \theta} \right)^t \\
E[\exp(i \theta \hat{T}_t)] = E[\exp(-\frac{1}{2} \theta^2 T_t)] = \left( \frac{\tanh \theta}{\theta} \right)^t 
\]

(3.4)  (3.5)  (3.6)

By setting \( \theta = \sqrt{2\lambda} \), we can see that the characteristic functions of \( \hat{C}_t, \hat{S}_t, \) and \( \hat{T}_t \), become the Laplace transforms of \( C_t, S_t, \) and \( T_t \). Let \( \hat{X}_t \) denote \( \hat{C}_t, \hat{S}_t, \) or \( \hat{T}_t \); and \( X_t \) denote \( C_t, S_t, \) or \( T_t \). Pitman and Yor (2003) construct \( \hat{X}_t \) from \( X_t \) by Brownian subordination

\[ \hat{X}_t = \beta_{\hat{X}_t} \]  

(3.7)

where \( \beta_{\hat{X}_t}, u \geq 0 \) is a standard Brownian motion and is independent of the subordinator \( u \) that is the increasing Lévy process \( X_t \). This means that \( \hat{X}_t = \beta_1 \sqrt{X_t} \), where \( \beta_1 \) is a standard normal random variable (whose characteristic function is \( \exp(-\theta^2 / 2) \)), and we have following moments relation

\[ E(|\hat{X}_t|^2) = E(|\beta_1|^2) E(X_t^s), \quad \text{Re}(s) > -\frac{1}{2} \]  

(3.8)

where

\[ E(|\beta_1|^\alpha) = 2^{1-\alpha} \frac{\Gamma(\alpha)}{\Gamma(\alpha/2)} \]  

(3.9)

**Identity in Distribution**  Pitman and Yor (2003) show some distributional identities of those processes. For example, \( C_1 \) has a distribution of the hitting time of \( \pm 1 \) by a one-dimensional Brownian motion. \( S_1 \) has a distribution of the hitting time of the unit sphere by a three-dimensional Brownian
motion started at the origin. \((\pi/2) \sqrt{S_t}\) has a distribution of the maximum of a standard Brownian excursion. The following identities hold as well

\[
4S_t \overset{d}{=} S_t + C_t
\]  

(3.10)

where \(S_t\) and \(C_t\) are independent.

\[
C_t \overset{d}{=} S_t + T_t
\]  

(3.11)

where \(S_t\) and \(T_t\) are independent.

### 3.2. Properties of \(C_t\), \(S_t\) and \(T_t\)

#### Moment Recurrence

Pitman and Yor (2003) showed that each of the processes \(C_t\), \(S_t\) and \(T_t\) exhibit moment recurrences.

For \(t \geq 0\) and \(s = 1, 2, \ldots\), the process \(C_t\) has

\[
(t^2 + t) E[C_{t+s}^2] = t^2 E[C_t^2] + (2s + 1) E[C_t^{s+1}]
\]  

(3.12)

For \(t \geq 0\) and \(s = 1, 2, \ldots\), the process \(S_t\) has

\[
(t^2 + t) E[S_{t+s}^2] = (t - 2s)(t - 2s + 1) E[S_t^2] + 2st^2 E[S_t^{s-1}]
\]  

(3.13)

For \(t \geq 1\) and \(s = 1, 2, \ldots\), the process \(T_t\) has

\[
(2s + t) E[T_{t+s}^2] = t E[T_{t-1}^2] + 2st E[S_{t+s}^{s-1}]
\]  

(3.14)

A special moment of \(C_t\) is

\[
E[C_t^{-1/2}] = \frac{\Gamma(2)}{\sqrt{\pi} \Gamma((t+1)/2)}
\]  

(3.15)

In particular, when \(t = 1/2\),

\[
E[C_{1/2}^{-1/2}] = E\left[\frac{1}{\sqrt{C_{1/2}}}\right] = \frac{\Gamma(1/4)}{\sqrt{\pi} \Gamma(3/4)}
\]  

(3.16)

Knowing that

\[
\Gamma(x) \Gamma(1-x) = \frac{\pi}{\sin(\pi x)}
\]  

(3.17)

we have

\[
\Gamma\left(\frac{1}{4}\right) \Gamma\left(\frac{3}{4}\right) = \frac{\pi}{\sin(\pi/4)} = \sqrt{2}\pi
\]  

(3.18)
Therefore, $E[C_{1/2}^{-1/2}]$ can be simplified as

$$E[C_{1/2}^{-1/2}] = \frac{\Gamma(1/4)}{\sqrt{2} \Gamma(3/4)} \frac{\Gamma(1/4)^2}{\sqrt{2} \Gamma(1/4) \Gamma(3/4)} = \frac{\Gamma(1/4)^2}{2\pi}$$

(3.19)

Let $L$ be the Lemniscate constant defined as

$$L = \frac{1}{2\sqrt{2\pi}} \Gamma\left(\frac{1}{4}\right)^2$$

(3.20)

$E[C_{1/2}^{-1/2}]$ can also be expressed in terms of the Lemniscate constant

$$E[C_{1/2}^{-1/2}] = \sqrt{\frac{2}{\pi}} L$$

(3.21)

This special moment, $E[C_{1/2}^{-1/2}]$, indeed has relevance for this thesis because it is the expectation of the reciprocal of the square root of the denominator of the Dickey-Fuller distribution as expressed as a ratio of Brownian functionals.

**Connections with the Gamma Process**

Following Pitman and Yor (2003), The characterizations of the processes $C_t$ and $S_t$ have connections with that of the gamma process $\Gamma_t$. Recall that $\Gamma_t$ can be characterized by the density, for $t > 0$ and $x > 0$,

$$f(\Gamma_t; x, \alpha) = \frac{\Gamma(t+\alpha)}{\Gamma(t)} x^{t-1} e^{-x}$$

(3.22)

by the moments, for $\text{Re}(s) > -t$,

$$E(\Gamma_t^s) = \frac{\Gamma(t+s)}{\Gamma(t)}$$

(3.23)

or by the Laplace transform

$$E(e^{-\lambda \Gamma_t}) = \left(\frac{1}{1+\lambda}\right)^t$$

(3.24)

Let $\Gamma_{\alpha,t}$, $t \geq 0$ be a sequence of independent gamma process, and define, for $\alpha > 0$ and $t \geq 0$,

$$\Sigma_{\alpha,t} := \frac{2}{\pi t} \sum_{n=0}^{\infty} \frac{\Gamma_{\alpha,n}}{(\alpha+n)^t}$$

(3.25)

Then, applying the Laplace transform of the gamma process we have
\[ E \left[ e^{-\frac{1}{2} \theta^2 \Sigma_{ij}} \right] = E \left[ e^{-\frac{1}{2} \theta^2 \sum_{n=0}^{\infty} \frac{\Gamma_n}{(a+n)^2}} \right] \]
\[ = E \left[ \prod_{n=0}^{\infty} e^{-\frac{\theta^2}{\pi^2(a+n)^2} \Gamma_n} \right] \]
\[ = \prod_{n=0}^{\infty} E \left[ e^{-\frac{\theta^2}{\pi^2(a+n)^2} \Gamma_n} \right] \]
\[ = \prod_{n=0}^{\infty} \left( 1 + \frac{\theta^2}{\pi^2(a+n)^2} \right)^{-1} \]
\[ = \left[ \prod_{n=0}^{\infty} \left( 1 + \frac{\theta^2}{\pi^2(a+n)^2} \right)^{-1} \right] \]  
(3.26)

Using the Euler’s infinite products for \( \sinh \theta \) and \( \cosh \theta \)

\[ \sinh \theta = \theta \prod_{n=1}^{\infty} \left( 1 + \frac{\theta^2}{\pi^2 n^2} \right) \]
\[ = \theta \prod_{n=0}^{\infty} \left( 1 + \frac{\theta^2}{\pi^2(1+n)^2} \right) \]
\[ \cosh \theta = \prod_{n=1}^{\infty} \left( 1 + \frac{4 \theta^2}{\pi^2(2n-1)^2} \right) \]
\[ = \prod_{n=0}^{\infty} \left( 1 + \frac{\theta^2}{\pi^2 \left( \frac{1}{2} + n \right)^2} \right) \]  
(3.27)

we obtain

\[ E \left[ e^{-\frac{1}{2} \theta^2 \Sigma_{ij}} \right] = \left[ \prod_{n=0}^{\infty} \left( 1 + \frac{\theta^2}{\pi^2(1+n)^2} \right)^{-1} \right] \]
\[ = \left( \frac{\theta}{\sinh \theta} \right)^\gamma \]  
(3.28)

and

\[ E \left[ e^{-\frac{1}{2} \theta^2 \Sigma_{ij}} \right] = \left[ \prod_{n=0}^{\infty} \left( 1 + \frac{\theta^2}{\pi^2 \left( \frac{1}{2} + n \right)^2} \right)^{-1} \right] \]
\[ = \left( \frac{1}{\cosh \theta} \right)^\gamma \]  
(3.29)

Hence

\[ C_t = \sum_{j=1}^{d} \frac{d_j}{\pi^2} \sum_{n=0}^{\infty} \frac{\Gamma_n}{(1+n)^2} \]
\[ S_t = \sum_{i=1}^{d} \frac{d_i}{\pi^2} \sum_{n=0}^{\infty} \frac{\Gamma_n}{(1+n)^2} \]  
(3.30)

Expression (3.30) tells us that the random variables \( C_t \) and \( S_t \) can be decomposed, for given \( t \), up to a multiplicative constant as a weighted sum of independent gamma distributed random variables. For \( C_{1/2} \) and \( S_{1/2} \), i.e. for the case \( t = 1/2 \), this weighted sum will be of chi-squared random variables given the way the gamma distribution specializes in this case.

**Densities** Using the negative binomial expansion, for \( t > 0 \) and \(|x| < 1\),
\[
\frac{1}{(1-x)^\theta} = \frac{1}{\Gamma(\theta)} \sum_{n=0}^{\infty} \frac{\Gamma(\theta+n)}{\Gamma(n+1)} x^n 
\]  

(B3.1)

Biane, Pitman and Yor (2001) obtained the following series expansions

\[
\left( \frac{1}{\cosh \sqrt{2} \lambda} \right)^\theta = \frac{2^\theta}{\Gamma(\theta)} \sum_{n=0}^{\infty} (-1)^n \frac{\Gamma(\theta+n)}{\Gamma(n+1)} e^{-(2n+1) \sqrt{2} \lambda} 
\]

(B3.2)

\[
\left( \frac{\sqrt{2} \lambda}{\sinh \sqrt{2} \lambda} \right)^{\theta-1} = \frac{2^\theta}{\Gamma(\theta)} \sum_{n=0}^{\infty} \frac{\Gamma(\theta+n)}{\Gamma(n+1)} e^{-(2n+1) \sqrt{2} \lambda} 
\]

(B3.3)

Replacing \( \theta \) with \( \sqrt{2} \lambda \), we represent the Laplace transforms of \( C_t \) and \( S_t \) as series expansions

\[
\left( \frac{1}{\cosh \sqrt{2} \lambda} \right)^\theta = \frac{2^\theta}{\Gamma(\theta)} \sum_{n=0}^{\infty} (-1)^n \frac{\Gamma(\theta+n)}{\Gamma(n+1)} e^{-(2n+1) \sqrt{2} \lambda} 
\]

(B3.4)

\[
\left( \frac{\sqrt{2} \lambda}{\sinh \sqrt{2} \lambda} \right)^{\theta-1} = \frac{2^\theta}{\Gamma(\theta)} \sum_{n=0}^{\infty} \frac{\Gamma(\theta+n)}{\Gamma(n+1)} e^{-(2n+1) \sqrt{2} \lambda} 
\]

(B3.5)

Since

\[
\left( \frac{\tanh \sqrt{2} \lambda}{\sqrt{2} \lambda} \right)^\theta = \frac{1}{\cosh \sqrt{2} \lambda} \left( \frac{\sqrt{2} \lambda}{\sinh \sqrt{2} \lambda} \right)^{\theta-1} 
\]

we can write \( \frac{\tanh \sqrt{2} \lambda}{\sqrt{2} \lambda} \) as the quotient of two series expansions

\[
\left( \frac{\tanh \sqrt{2} \lambda}{\sqrt{2} \lambda} \right)^\theta = -\frac{1}{(2\lambda)^{\theta+2}} \sum_{n=0}^{\infty} \frac{\Gamma(\theta+n)}{\Gamma(n+1)} e^{-(2n+2) \sqrt{2} \lambda} 
\]

(B3.6)

(B3.7)

Biane, Pitman and Yor (2001) suggested the series expansions for the Laplace transforms of \( C_t \) and \( S_t \) can be inverted term by term using the Lévy’s formula

\[
\int_0^{\infty} \frac{a}{\sqrt{2\pi x^3}} \exp\left( -\frac{a^2}{2x} \right) e^{-\lambda x} \, dx = e^{-a \sqrt{2 \lambda}} 
\]

(B3.8)

From the Lévy’s formula, we can see the original function of the Laplace transform \( e^{-a \sqrt{2 \lambda}} \) is \( \frac{a}{\sqrt{2\pi x^3}} \exp\left( -\frac{a^2}{2x} \right) \). In the series expansion of the Laplace transform of \( C_t \), the term involving the complex variable \( \lambda \) is \( e^{-(2n+1) \sqrt{2 \lambda}} \). Thus, the original function of the term is \( \frac{2n+1}{\sqrt{2\pi x^3}} \exp\left( -\frac{(2n+1)^2}{2x} \right) \).

Hence, as shown by Biane, Pitman and Yor (2001), the density function of \( C_t \) is given by

\[
f_{C_t}(x) = \frac{d(\text{Laplace}(dx))}{dx} = \frac{2^\theta}{\Gamma(\theta)} \sum_{n=0}^{\infty} (-1)^n \frac{\Gamma(\theta+n)}{\Gamma(n+1)} \frac{2n+1}{\sqrt{2\pi x^3}} \exp\left( -\frac{(2n+1)^2}{2x} \right) 
\]

(B3.9)

While, in the series expansion for the Laplace transform of \( S_t \), the term involving the complex variable \( \lambda \) is \( (2\lambda)^{\theta+2} e^{-(2n+1) \sqrt{2 \lambda}} \). Its inverse Laplace transform can be computed by exploiting symbolic calculations of Mathematica.
\begin{equation}
L^{-1}\left\{ (2\lambda)^{t/2} e^{-t/2} \right\} = 2^t (2 n + t) x^{t+3/2} \times \\
\left( \frac{\sqrt{x}}{(2 n+\theta) \Gamma(-\frac{t+1}{2})} \right) \text{Fr}_{\frac{t+2}{2} \cdot \frac{1}{2} \cdot \frac{(2 n+\theta)}{2 x}} - \frac{\sqrt{x}}{\Gamma(-\frac{t+1}{2})} \text{Fr}_{\frac{t+3}{2} \cdot \frac{3}{2} \cdot \frac{(2 n+\theta)}{2 x}} \right)
\end{equation}

Therefore, the density function of \( S_t \) is expressed in terms of the Kummer confluent hypergeometric function \( \text{Fr}_{a; b; z} \)

\begin{equation}
\text{fr}(x) = \frac{\text{fr}_{a}(x)}{dx} = \frac{2^{t/2}}{\Gamma(x)} \sum_{n=0}^{\infty} \frac{\Gamma(n+\theta)}{\Gamma(n+1)} \left( 2 n + t \right) x^{t+3/2} \times \\
\left( \frac{\sqrt{x}}{(2 n+\theta) \Gamma(-\frac{t+1}{2})} \text{Fr}_{\frac{t+2}{2} \cdot \frac{1}{2} \cdot \frac{(2 n+\theta)}{2 x}} - \frac{\sqrt{x}}{\Gamma(-\frac{t+1}{2})} \text{Fr}_{\frac{t+3}{2} \cdot \frac{3}{2} \cdot \frac{(2 n+\theta)}{2 x}} \right)
\end{equation}

An alternative formula for the density of \( S_t \) can be found in Biane and Yor’s (1987) paper.

**Special Densities** Abadir and Parulo (1996) considered the inversion of the Laplace transform of \( S_t \) in (3.35) when \( t = 1/2 \), and yielded the density function of \( S_{1/2} \)

\begin{equation}
f_{S_{1/2}}(x) = \frac{1}{\sqrt{\pi}} x^{-3/4} \sum_{n=0}^{\infty} \left( \frac{1/2}{n} \right) (-1)^n e^{-n(1/4)/x} \text{Fr}_{2/3}(\frac{2 n+1/2}{\sqrt{x}})
\end{equation}

with \( D_n(x) \) the parabolic cylinder function (see Appendices A2)

\begin{equation}
D_n(x) = 2^{-n/2} e^{-x^2/4} H_n(x / \sqrt{2})
\end{equation}

where \( H_n(x) \) is a Hermite polynomial. They also noted a reference to the classic paper by Anderson and Darling (1952) for a derivation of the distribution function of \( S_{1/2} \).

The Laplace transform of \( S_t \) in (3.2) can be written as a series expansion involving the parabolic cylinder function \( D_n(x) \) when \( t = 1/2 \) (see Kac, 1951, Section 6 cited in Tolmatz, 2002)

\begin{equation}
\left( \frac{\sqrt{2 \lambda}}{\sinh \sqrt{2 \lambda}} \right)^{1/2} = 2^{1/4} \lambda^{1/4} \sum_{n=0}^{\infty} \frac{D_n(0)}{n!} e^{-n(1/2)/2 \lambda} \right)
\end{equation}

Tolmatz (2002) inverted this series expansion and obtained the density of \( S_{1/2} \).

\begin{equation}
f_{S_{1/2}}(x) = x^{-5/4} \sum_{n=0}^{\infty} \left( \frac{-1}{n!} \right) e^{-n(1/4)/x} \left( \frac{1}{\Gamma(1/2-n)} \right) D_n(2n+1/2) \sqrt{2 \lambda}
\end{equation}

\begin{equation}
\text{Fr}(x) = 2 x^{-1/4} \sum_{n=0}^{\infty} \left( \frac{-1}{n!} \right) e^{-n(1/4)/x} \left( \frac{1}{\Gamma(1/2-n)} \right) D_{-1/2}(2n+1/2) \sqrt{2 \lambda}
\end{equation}

Devroye (2009a) studied two non-negative random variables denoted by \( J \) and \( J' \) which are
characterized by

\[
E[e^{-\lambda J}] = \frac{\sqrt{2\lambda}}{\sinh \sqrt{2\lambda}} \\
E[e^{-\lambda J^*}] = \frac{1}{\cosh \sqrt{2\lambda}}
\]  

(3.47)

Comparing them with the definitions of \(C_t\) and \(S_t\), we find

\[
J \overset{\mathcal{L}}{=} S_1, \quad J^* \overset{\mathcal{L}}{=} C_1
\]

(3.48)

Using the Jacobi theta function \(\theta(x) = \sum_{n=-\infty}^{\infty} \exp(-n^2 \pi x), \ x > 0\) and its property \(\sqrt{x} \theta(x) = \theta(1/x)\), Devroye (2009a) obtained the densities of \(J\) and \(J^*\) (\(J\) for Jacobi)

\[
f_{C_t}(x) = f_{J^*}(x) = \pi \sum_{n=0}^{\infty} (-1)^n \left(n + \frac{1}{2}\right) \exp \left(\frac{-(n+1/2)^2 \pi x}{2}\right)
\]

\[
f_{S_t}(x) = f_{J_t}(x) = \sum_{n=1}^{\infty} (-1)^n \left(n + \frac{1}{2}\right) \exp \left(\frac{-(n+1/2)^2 \pi x}{2}\right)
\]

(3.49)

Pitman and Yor (2003) obtain \(f_{J_t}(x)\) by using the relation between the density of \(T_t\) and the Kolmogorov measure \(K_C\) of \(C_t\)

\[
f_{T_t}(x) = \frac{K_C(dx)}{x^2} = \sum_{m=1}^{\infty} e^{-\pi \left(m^2 - \frac{1}{2}\right)^2} x^2
\]

(3.50)

The Kolmogorov measure \(K_X\) of a Lévy process \((X_t)\) is defined as a scalar multiple of the distribution of \(X_u\) for some \(u \geq 0\)

\[
\frac{K_X(dx)}{K_X(R)} = P(X_u \in dx)
\]

(3.51)

\subsection*{3.3. Laws of \(\hat{C}_t, \hat{S}_t\) and \(\hat{T}_t\)}

\textbf{Density of} \(\hat{C}_t\) \quad The density of \(\hat{C}_t\) defined by the inverse characteristic function

\[
\psi_t(x) := \frac{f_{\hat{C}_t}(dx)}{dx} = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{-i \theta x} \left(\frac{1}{\cosh \theta}\right)^t d\theta
\]

satisfies the recurrence (see Pitman and Yor, 2003, Lemma 2)

\[
t(t+1) \psi_{t+2}(x) = (t^2 + x^2) \psi_t(x)
\]

(3.53)

Pitman and Yor (2003) showed

\[
\psi_t(x) = \frac{2^{t-2}}{\pi} B \left(\frac{t+i}{2}, \frac{t-i}{2}\right) = \frac{2^{t-2}}{\pi \Gamma(t)} \left|\Gamma\left(\frac{t+i}{2}\right)\right|^2
\]

(3.54)

where \(B(a, b) := \frac{\Gamma(a) \Gamma(b)}{\Gamma(a+b)}\) is the beta function.
Inverting the characteristic function of $\hat{C}$ in Mathematica yields the expression

$$
\psi(x) = \frac{2^{-i}}{\pi} \left( \frac{\text{$_2F_1$}[\frac{1}{2}(t-i)x;\frac{3}{2}(t-i)x+2]_{-1}}{t-i} + \frac{\text{$_2F_1$}[\frac{1}{2}(t+i)x;\frac{3}{2}(t+i)x+2]_{-1}}{t+i} \right)
$$

(3.55)

where $\text{$_2F_1$}(a, b; c; z) := \sum_{k=0}^{\infty} \frac{(a)_k(b)_k}{(c)_k k!}$ is the hypergeometric function. Here is an example in which fails to resolve the expression obtained into its most concise or recognizable form.

**Density of $\hat{S}_t$** The density of $\hat{S}_t$ defined by the inverse characteristic function

$$
\phi(x) := \frac{\mathbb{P}(\hat{S}_t \leq x)}{dx} = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{-i\theta x} \left( \frac{\theta}{\sinh \theta} \right) d\theta
$$

(3.56)

satisfies the recurrence (see Pitman and Yor, 2003, Lemma 2)

$$
t(t + 1) \phi_{t+2}(x) = (t^2 + x^2) \phi''_t(x) + (2t + 4) x \phi'_t(x) + (1 + t)(2 + t) \phi_t(x)
$$

(3.57)

Inverting the characteristic function of $\hat{S}_t$ with $t$ equal to 1, 2, 3 and 4 respectively in Mathematica yields

$$
\phi_1(x) = \frac{\pi}{2(\cosh(\pi x) + 1)} = \frac{\pi}{4 \cosh(\frac{x}{2})}
$$

(3.58)

$$
\phi_2(x) = \frac{i\pi x \cosh(\frac{x}{2})}{2(\cosh(\pi x) + 1)} = \frac{\pi}{2} \frac{(\frac{x}{2}) \cosh(\frac{x}{2})}{\sinh(\frac{x}{2})} - 1
$$

(3.59)

$$
\phi_3(x) = \frac{1}{16} \pi \sech^4(\frac{x}{2}) \left( \left[ \pi^2 (x^2 + 1) + 6 \cosh(\pi x) - 2 \left( \pi^2 (x^2 + 1) + 3 \pi x \sinh(\pi x) - 3 \right) \right] \right)
$$

(3.60)

$$
\phi_4(x) = \frac{1}{96} \pi \cosh^5(\frac{x}{2}) \left[ \pi x \left[ 11 \pi^2 (x^2 + 4) - 36 \cosh(\frac{x}{2}) + \pi \left( \pi^2 (x^2 + 4) + 36 \cosh(\frac{3\pi}{2}) x \right) \right] - \sinh(\frac{x}{2}) \left[ \pi^2 (6 x^2 + 8) + \left( \pi^2 (3 x^2 + 4) + 6 \cosh(\pi x) - 6 \right) \right] \right]
$$

(3.61)

Note the four equations above are the products of the facility of Mathematica for symbolic calculations.

In contrast to the evaluation of (3.55), expressions (3.58) ~ (3.61) offers examples in which Mathematica facilitates progress on evaluating the densities.

The second equality in the respective expressions of $\phi_1(x)$ and $\phi_2(x)$ holds by applying the half-argument formulas

$$
\sinh(\frac{x}{2}) = \sqrt{\frac{1}{2} (\cosh x - 1)}
$$

(3.62)

$$
cosh(\frac{x}{2}) = \sqrt{\frac{1}{2} (\cosh x + 1)}
$$

we can see that $\phi_1(x)$ and $\phi_2(x)$ match the densities given by Pitman and Yor (2003, Table 6). It is easy to verify that $\phi_3$ and $\phi_4$ can also be derived from $\phi_1$ and $\phi_2$ respectively using the recurrence. Hence,
with the recurrence we can easily derive the formula for the density \( \phi_t(x) \) of \( \hat{S}_t \) for \( t = 1, 2, \ldots \). While, it is difficult to obtain the explicit formula for the density \( \phi_t(x) \) of \( \hat{S}_t \) for \( t = \frac{1}{2}, \frac{3}{2}, \ldots \) by inverting the characteristic function. However, numerical inversion can obtain the approximate density for general \( t > 0 \).

The density \( \eta_t(x) \) of \( \hat{T}_t \) defined by the inverse characteristic function

\[
\eta_t(x) := \frac{\rho_{\hat{T}_t}(dx)}{dx} = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{-i\theta x} \left( \frac{\tanh \theta}{\theta} \right)^t d\theta
\]

(3.63)

satisfies the recurrence (see Pitman and Yor, 2003, Lemma 2)

\[-x \eta_t'(x) + (t - 1) \eta_t(x) = t \eta_{t-1}(x) + t \eta'_{t+1}(x)\]

(3.64)

**Density of \( \hat{T}_t \)** Pitman and Yor (2003) gave the formula of the density \( \eta_1(x) \) of \( \hat{T}_1 \) as follows

\[
\eta_1(x) = \frac{1}{\pi} \log \coth \left( \frac{x}{4} \right)
\]

(3.65)

and derived the density \( \eta_2(x) \) of \( \hat{T}_2 \) from \( \eta_1(x) \) using the recurrence for the density. With \( t = 1 \) we find

\[-x \eta'_1(x) = \eta_2''(x)\]

(3.66)

Then

\[
\eta_2''(x) = -x \left( \frac{1}{\pi} \log \coth \left( \frac{x}{4} \right) \right)'
\]

\[
= -x \left( \frac{1}{\pi} \log \coth \left( \frac{x}{4} \right) \right)
\]

\[
\left. \left. \right. \right. = -x \left( \frac{1}{\pi} \log \coth \left( \frac{x}{4} \right) \right)
\]

\[
\left. \left. \right. \right. = -x \left( \frac{1}{\pi} \log \coth \left( \frac{x}{4} \right) \right)
\]

(3.67)

The expression of \( \eta_2''(x) \) is identical to the density \( \psi_2(x) \) of \( \hat{C}_2 \). Thus, Pitman and Yor (2003) showed

\[
\eta_2(x) = E \left[ \hat{C}_2 - |x| \right]
\]

\[
= \int_{|s|=\infty} \left( \frac{1}{2\sinh(x/2)} \right) (s - |x|) d\phi
\]

(3.68)

Here \( \eta_2(x) \) is expressed as an integral. Evaluating the integral for \( \eta_2(x) \) in *Mathematica* gives

\[
\eta_2(x) = \frac{2}{\pi} |x| \left( \text{Li}_2(e^{-\frac{1}{2} \pi |x|}) - \text{Li}_2(-e^{-\frac{1}{2} \pi |x|}) \right) + \frac{\pi}{2} \left( \text{Li}_3(e^{-\frac{1}{2} \pi |x|}) - \text{Li}_3(-e^{-\frac{1}{2} \pi |x|}) \right)
\]

(3.69)

where \( \text{Li}_n(z) := \sum_{k=1}^{\infty} \frac{z^k}{k^n} / k^n \) is the polylogarithm function.

When \( x = 0 \), the formula of \( \eta_2(x) \) reduces to
\[ \eta_2(0) = \frac{14 \zeta(3)}{\pi^2} \]  

(3.70)

where for \( \text{Re}(s) > 1 \), \( \zeta(s) := \sum_{k=1}^{\infty} k^{-s} \) is the Riemann zeta function.

**Related Distributions**  The random variable \( H^* \) with the hyperbolic secant distribution, the random variable \( H \) with the hyperbolic cosecant distribution and the random variable \( H' \) with the hyperbolic tangent distribution are defined by their characteristic functions (see Devroye, 2009a)

\[
\begin{align*}
\varphi_{H^*}(t) &= \frac{1}{\cosh t} \left( e^{i t} + e^{-i t} \right) = \frac{2}{e^{t} - e^{-t}} \nonumber \\
\varphi_H(t) &= \frac{t}{\sinh t} \left( e^{i t} - e^{-i t} \right) = \frac{2 i t}{e^{t} - e^{-t}} \\
\varphi_{H'}(t) &= \frac{\tanh t}{t} \left( e^{i t} - e^{-i t} \right) = \frac{1}{i} \left( e^{i t} - 1 \right) \\
\end{align*}
\]  

(3.71)

These show that

\[
H^* \overset{\text{d}}{=} \hat{C}_1, \quad H \overset{\text{d}}{=} \hat{S}_1, \quad H' \overset{\text{d}}{=} \hat{T}_1
\]  

(3.72)

Since

\[
\begin{align*}
\varphi_{H+H'}(t) &= \mathbb{E}\left[ e^{i t (H+H')} \right] \\
&= \mathbb{E}\left[ e^{i t H} e^{i t H'} \right] \\
&= \mathbb{E}\left[ e^{i t H} \right] \mathbb{E}\left[ e^{i t H'} \right] \\
&= \frac{1}{\cosh t}
\end{align*}
\]  

(3.73)

we have

\[ H^* = H + H' \]  

(3.74)

or equivalently

\[ \hat{C}_1 = \hat{S}_1 + \hat{T}_1 \]  

(3.75)

The GHS (generalized hyperbolic secant) distribution has the characteristic function (see Devroye, 2009a)

\[ \varphi(t) = \left( \frac{1}{\cosh t} \right)^\alpha = \left( \frac{2}{e^{t} - e^{-t}} \right)^\alpha \]  

(3.76)

where \( \alpha > 0 \). The GHS random variable \( G_\alpha \) has the following identities

\[ G_\alpha \overset{\text{d}}{=} \hat{C}_\alpha \]  

(3.77)

and, for integer \( \alpha \),

\[ G_\alpha \overset{\text{d}}{=} H^*(1) + \ldots + H^*(\alpha) \]  

(3.78)
where $H^*(i)$ are i.i.d. copies of $H^*$. 
4. Numerical Experiments with $C_t$, $S_t$ and $T_t$

In this section, the densities of the infinitely divisible distributions associated with hyperbolic functions proposed in Pitman and Yor (2003) are computed by various methods including analytical formulae and numerical inversion of the Laplace transform. The results of different methods are compared with each other in terms of the accuracy and the computation time. We find the most suitable method for a problem and give the recommended settings of parameters.

The algorithms for numerical Laplace transform inversion considered here are the unified Gaver-Stehfest algorithm (UniG), the unified Euler algorithm (UniE), and the unified Talbot algorithm (UniT) which are proposed in Abate and Whitt’s (2006). These three algorithms are based on different methods and have respective merits. Among them, one may be more suitable for some problem than the others.

- 4.1. Computing the density of $C_t$

The density of $C_t$ can be computed using a series expansion (3.39) derived by Biane, Pitman and Yor (2001) who express the Laplace transform of $C_t$ as an infinite series and then invert the series term by term based on the Lévy’s formula. When $t = 1$, $C_1$ is identical to $J^*$ whose density is represented by Devroye (2009a) as an infinite series (see eq. 3.49) using the properties of the Jacobi function. For clarification, we denote the formula (3.39) by the BPY method and the formula (3.49) by the Devroye method.

By using numerical inversion, we find that UniG is the most universal method which can invert the Laplace transform of $C_t$ in (3.1) for any $t > 0$, while UniT and UniE can invert the Laplace transform for integer $t > 0$. When the Laplace transform of $C_t$ is written as a series expansion in (3.34), it turns out that UniT and UniE are able to invert the series expansion for any $t > 0$. In this case, one more parameter $n0$ will appear for UniT and UniE to indicate the number of terms used in the expansion.

- 4.1.1 The density of $C_{1/2}$

We first calculate the reference density of $C_{1/2}$ by matching the results of two different methods, and then use the reference density to check the accuracy of each method. By setting the truncation parameter $n0 = 100$ and working precision $wp = 80$ for the BPY method and setting the truncation parameter $M = 100$ and working precision $wp = 200$ for UniG, the BPY method agrees with UniG to more than 40 significant digits. The number of matching digits is determined by the least accurate one among the
methods used for comparison. The reference density of other random variables can be computed in a similar way.

Next, we treat the matching digits as correct density and use it check the accuracy of the BPY method with \( n_0 = 20 \), \( wp = 60 \) and the accuracy of UniG with \( M = 35 \), \( wp = 70 \). As shown in Table 4.1, the BPY method is much more accurate than UniG. With only 20 terms in the truncated series the BPY method has the same accuracy as the reference density. Most likely, the accuracy of the BPY method is limited by the working precision, and will increase with it. The BPY method is extremely fast: it yields a result instantly. While, UniG with the current setting is accurate to about 15 digits but still performs very well as the computation costs merely 0.03 seconds. Note the accuracy of numerical inversion such as UniG can always be improved by increasing \( M \) and \( wp \).

Table 4.1. Accuracy of the BPY method and UniG when computing the density of \( C_{12} \)

<table>
<thead>
<tr>
<th>x</th>
<th>Reference Accu</th>
<th>Reference Density</th>
<th>BPY n0 = 20, wp = 60</th>
<th>UniG M = 35, wp = 70</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accu</td>
<td>Density</td>
<td>Accu</td>
<td>Density</td>
</tr>
<tr>
<td>0.01</td>
<td>41</td>
<td>0.001051269451</td>
<td>58.9</td>
<td>41</td>
</tr>
<tr>
<td>0.02</td>
<td>43</td>
<td>0.1925349438</td>
<td>59.0</td>
<td>43</td>
</tr>
<tr>
<td>0.03</td>
<td>46</td>
<td>2.07113296</td>
<td>59.3</td>
<td>46</td>
</tr>
<tr>
<td>0.04</td>
<td>48</td>
<td>2.555800587</td>
<td>59.5</td>
<td>48</td>
</tr>
<tr>
<td>0.05</td>
<td>44</td>
<td>1.688168107</td>
<td>59.7</td>
<td>44</td>
</tr>
<tr>
<td>0.06</td>
<td>44</td>
<td>1.131638871</td>
<td>59.7</td>
<td>44</td>
</tr>
<tr>
<td>0.07</td>
<td>45</td>
<td>0.8146801528</td>
<td>59.7</td>
<td>44</td>
</tr>
<tr>
<td>0.08</td>
<td>45</td>
<td>0.6175424262</td>
<td>59.7</td>
<td>45</td>
</tr>
<tr>
<td>0.09</td>
<td>45</td>
<td>0.4845206735</td>
<td>59.7</td>
<td>45</td>
</tr>
<tr>
<td>1</td>
<td>44</td>
<td>0.3890356224</td>
<td>59.7</td>
<td>45</td>
</tr>
<tr>
<td>2</td>
<td>45</td>
<td>0.3173890190</td>
<td>59.7</td>
<td>45</td>
</tr>
<tr>
<td>3</td>
<td>44</td>
<td>0.2619189121</td>
<td>59.6</td>
<td>45</td>
</tr>
<tr>
<td>4</td>
<td>44</td>
<td>0.2179999738</td>
<td>59.6</td>
<td>44</td>
</tr>
<tr>
<td>5</td>
<td>45</td>
<td>0.1925349438</td>
<td>59.6</td>
<td>45</td>
</tr>
</tbody>
</table>


wp: computing precision. Accu: accuracy measured by the number of significant digits. ED: effective number of digits of precision in the result. CPU: computation time in seconds.

The reference density is obtained by digits matching with the following option specifications for our code DensityC[ ]:
(Method -> ("Biane", "n0" -> 100), WorkingPrecision -> 80) vs
(Method -> ("UniG", "M" -> 100), WorkingPrecision -> 200).

UniT and UniE cannot invert the Laplace transform of \( C_i \) in (3.1) directly, but they can invert the series expansion of the Laplace transform in (3.34). Using the first 20 terms of the series expansion (set \( n0 = 20 \), UniT and UniE with parameter settings indicated in Table 4.2 compute the density of \( C_{12} \) to about 15-digit accuracy within 0.05 seconds and 0.11 seconds respectively. UniT and UniE consume more CPU time than UniG. However, the numerical results in Chapter 2 on computing Asian option
prices suggest UniT is faster than UniE and UniG. This is because in the computation of an Asian option price, the same expression of the Laplace transform is used in the inversion. But as computing the density of \( C \), UniG is applied to the Laplace transform in a simple form. While, UniT and UniE have to be applied to the series expansion which has 20 terms. This difference causes UniG to work faster than UniT and UniE.

Table 4.2. Accuracy of UniT and UniE when computing the density of \( C \)

<table>
<thead>
<tr>
<th>( x )</th>
<th>( )</th>
<th>Reference</th>
<th>Accu</th>
<th>Density</th>
<th>UniT</th>
<th>UniE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>( n_0 = 20, \ M = 25, \ wp = 25 )</td>
<td>( n_0 = 20, \ M = 25, \ wp = 30 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.01</td>
<td>41</td>
<td>0.001051269451</td>
<td>23.6</td>
<td>14</td>
<td>0.0010513</td>
<td>0.05</td>
</tr>
<tr>
<td>0.02</td>
<td>43</td>
<td>0.1925349438</td>
<td>23.6</td>
<td>17</td>
<td>0.19253</td>
<td>0.05</td>
</tr>
<tr>
<td>0.05</td>
<td>46</td>
<td>2.071113296</td>
<td>22.8</td>
<td>17</td>
<td>2.0711</td>
<td>0.05</td>
</tr>
<tr>
<td>0.1</td>
<td>48</td>
<td>2.555800587</td>
<td>22.0</td>
<td>19</td>
<td>2.5558</td>
<td>0.03</td>
</tr>
<tr>
<td>0.2</td>
<td>44</td>
<td>1.688168107</td>
<td>21.4</td>
<td>17</td>
<td>1.6882</td>
<td>0.05</td>
</tr>
<tr>
<td>0.3</td>
<td>44</td>
<td>1.131638871</td>
<td>21.1</td>
<td>16</td>
<td>1.1316</td>
<td>0.05</td>
</tr>
<tr>
<td>0.4</td>
<td>44</td>
<td>0.8146801528</td>
<td>20.9</td>
<td>15</td>
<td>0.81468</td>
<td>0.05</td>
</tr>
<tr>
<td>0.5</td>
<td>45</td>
<td>0.6175424262</td>
<td>20.8</td>
<td>15</td>
<td>0.61754</td>
<td>0.05</td>
</tr>
<tr>
<td>0.6</td>
<td>45</td>
<td>0.4845206735</td>
<td>20.7</td>
<td>15</td>
<td>0.48452</td>
<td>0.05</td>
</tr>
<tr>
<td>0.7</td>
<td>45</td>
<td>0.3890356224</td>
<td>20.6</td>
<td>15</td>
<td>0.38904</td>
<td>0.05</td>
</tr>
<tr>
<td>0.8</td>
<td>45</td>
<td>0.3173890190</td>
<td>20.4</td>
<td>16</td>
<td>0.31739</td>
<td>0.05</td>
</tr>
<tr>
<td>0.9</td>
<td>45</td>
<td>0.2619189121</td>
<td>20.4</td>
<td>15</td>
<td>0.26192</td>
<td>0.05</td>
</tr>
<tr>
<td>1.0</td>
<td>45</td>
<td>0.2179999738</td>
<td>20.3</td>
<td>15</td>
<td>0.21800</td>
<td>0.05</td>
</tr>
<tr>
<td>2.0</td>
<td>45</td>
<td>0.04354877489</td>
<td>19.9</td>
<td>14</td>
<td>0.043549</td>
<td>0.05</td>
</tr>
<tr>
<td>3.0</td>
<td>44</td>
<td>0.01025980622</td>
<td>19.3</td>
<td>13</td>
<td>0.010260</td>
<td>0.05</td>
</tr>
</tbody>
</table>


wp: computing precision. Accu: accuracy measured by the number of significant digits. ED: effective number of digits of precision in the result. CPU: computation time in seconds.

The reference density is obtained by digits matching with the following option specifications for our code DensityC[]:

\{(Method -> "Biane", "n0" -> 100), WorkingPrecision -> 80\} \&
\{(Method -> "UniG", "M" -> 100), WorkingPrecision -> 200\}.

### 4.1.2 The density of \( C \)

When \( t = 1 \), the density of \( C \) can also be obtained by the Devroye method. Table 4.3 shows the Devroye method is as good as the Biane method. The density is computed to very high accuracy immediately by both methods. It should be noted that the Devroye method only works for \( t = 1 \), whereas the BPY method works for any \( t > 0 \).
Table 4.3. Accuracy of the BPY method and the Devroye method when computing the density of $C_1$

<table>
<thead>
<tr>
<th>x</th>
<th>Reference</th>
<th>BPY n=20, wp=60</th>
<th></th>
<th>Devroye n=20, wp=60</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accu</td>
<td>Density</td>
<td>ED</td>
<td>Accu</td>
<td>Result</td>
</tr>
<tr>
<td>0.05</td>
<td>42</td>
<td>0.03239964382</td>
<td>58.9</td>
<td>42</td>
<td>0.0032400</td>
</tr>
<tr>
<td>0.1</td>
<td>44</td>
<td>0.1700073232</td>
<td>59.2</td>
<td>44</td>
<td>0.17001</td>
</tr>
<tr>
<td>0.2</td>
<td>45</td>
<td>0.322491236</td>
<td>59.4</td>
<td>45</td>
<td>0.32225</td>
</tr>
<tr>
<td>0.3</td>
<td>44</td>
<td>0.327238771</td>
<td>59.6</td>
<td>44</td>
<td>0.32723</td>
</tr>
<tr>
<td>0.4</td>
<td>44</td>
<td>0.434488906</td>
<td>59.8</td>
<td>44</td>
<td>0.43449</td>
</tr>
<tr>
<td>0.5</td>
<td>44</td>
<td>0.5624979476</td>
<td>59.6</td>
<td>44</td>
<td>0.56293</td>
</tr>
<tr>
<td>0.6</td>
<td>44</td>
<td>0.7432584500</td>
<td>59.6</td>
<td>45</td>
<td>0.74326</td>
</tr>
<tr>
<td>0.7</td>
<td>44</td>
<td>0.6603341573</td>
<td>59.6</td>
<td>46</td>
<td>0.66033</td>
</tr>
<tr>
<td>0.8</td>
<td>45</td>
<td>0.5847941345</td>
<td>59.6</td>
<td>45</td>
<td>0.58479</td>
</tr>
<tr>
<td>0.9</td>
<td>44</td>
<td>0.5172834959</td>
<td>59.6</td>
<td>44</td>
<td>0.51728</td>
</tr>
<tr>
<td>1.0</td>
<td>44</td>
<td>0.4573652256</td>
<td>59.5</td>
<td>44</td>
<td>0.45737</td>
</tr>
<tr>
<td>1.5</td>
<td>45</td>
<td>0.2468512783</td>
<td>59.4</td>
<td>45</td>
<td>0.24685</td>
</tr>
<tr>
<td>2.0</td>
<td>44</td>
<td>0.1332133192</td>
<td>59.2</td>
<td>44</td>
<td>0.13321</td>
</tr>
<tr>
<td>2.5</td>
<td>44</td>
<td>0.07188636504</td>
<td>59.0</td>
<td>44</td>
<td>0.07188</td>
</tr>
<tr>
<td>3.0</td>
<td>44</td>
<td>0.0387926484</td>
<td>58.8</td>
<td>44</td>
<td>0.03879</td>
</tr>
</tbody>
</table>


The reference density is obtained by digits matching with the following option specifications for our code DensityC[] :

- Method -> ("Biane", "n0"=100), WorkingPrecision -> 80 vs (Method -> ("UniG", "M"=100), WorkingPrecision -> 200).

When $t$ is an integer, all three inversion algorithms can invert the Laplace transform of $C_t$ in (3.1) successfully. As illustrated in Table 4.4, UniT and UniE become faster than UniG, and UniT seems to be slightly better than UniE.

Table 4.4. Accuracy of UniG, UniT and UniE when computing the density of $C_t$

<table>
<thead>
<tr>
<th>x</th>
<th>Reference</th>
<th>UniG M=25, wp=70</th>
<th>UniT M=25, wp=25</th>
<th>UniE M=25, wp=30</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accu</td>
<td>Density</td>
<td>ED Accu CPU ED</td>
<td>Accu CPU ED Accu CPU</td>
</tr>
<tr>
<td>0.05</td>
<td>42</td>
<td>0.003239964382</td>
<td>33.8  13 0.02</td>
<td>23.3  14 0.02</td>
</tr>
<tr>
<td>0.1</td>
<td>44</td>
<td>0.1700073232</td>
<td>31.4  15 0.03</td>
<td>23.0  17 0.03</td>
</tr>
<tr>
<td>0.2</td>
<td>45</td>
<td>0.322491236</td>
<td>29.2  14 0.02</td>
<td>22.3  16 0.02</td>
</tr>
<tr>
<td>0.3</td>
<td>44</td>
<td>0.327238771</td>
<td>28.1  15 0.03</td>
<td>21.9  17 0.03</td>
</tr>
<tr>
<td>0.4</td>
<td>44</td>
<td>0.434488906</td>
<td>27.4  14 0.03</td>
<td>21.6  17 0.03</td>
</tr>
<tr>
<td>0.5</td>
<td>44</td>
<td>0.5624979476</td>
<td>26.9  15 0.03</td>
<td>21.4  16 0.03</td>
</tr>
<tr>
<td>0.6</td>
<td>45</td>
<td>0.7432584500</td>
<td>26.6  15 0.03</td>
<td>21.2  16 0.03</td>
</tr>
<tr>
<td>0.7</td>
<td>46</td>
<td>0.6603341573</td>
<td>26.3  15 0.03</td>
<td>21.1  17 0.03</td>
</tr>
<tr>
<td>0.8</td>
<td>45</td>
<td>0.5847941345</td>
<td>26.0  15 0.03</td>
<td>21.0  17 0.03</td>
</tr>
<tr>
<td>0.9</td>
<td>44</td>
<td>0.5172834959</td>
<td>25.8  15 0.03</td>
<td>20.8  17 0.03</td>
</tr>
<tr>
<td>1.0</td>
<td>44</td>
<td>0.4573652256</td>
<td>25.7  15 0.03</td>
<td>20.8  17 0.03</td>
</tr>
<tr>
<td>1.5</td>
<td>45</td>
<td>0.2468512783</td>
<td>25.0  16 0.03</td>
<td>20.5  16 0.03</td>
</tr>
<tr>
<td>2.0</td>
<td>44</td>
<td>0.1332133192</td>
<td>24.5  16 0.03</td>
<td>20.2  16 0.03</td>
</tr>
<tr>
<td>2.5</td>
<td>44</td>
<td>0.07188636504</td>
<td>24.1  15 0.03</td>
<td>19.8  16 0.03</td>
</tr>
<tr>
<td>3.0</td>
<td>44</td>
<td>0.0387926484</td>
<td>23.7  15 0.03</td>
<td>19.5  16 0.03</td>
</tr>
</tbody>
</table>


wp: computing precision, Accu: accuracy measured by the number of significant digits. ED: effective number of digits of precision in the result. CPU: computation time in seconds.

The reference density is obtained by digits matching with the following option specifications for our code DensityC[] :

- Method -> ("Biane", "n0"=100), WorkingPrecision -> 80 vs (Method -> ("UniG", "M"=100), WorkingPrecision -> 200).

The reference density is obtained by digits matching with the following option specifications for our code DensityC[] :

- Method -> ("Biane", "n0"=100), WorkingPrecision -> 80 vs (Method -> ("UniG", "M"=100), WorkingPrecision -> 200).
## 4.1.3 The density of \( C_2 \)

The density of \( C_2 \) can be computed by the BPY method and numerical inversion with UniG, UniT and UniE. Table 4.5 gives the density with its accuracy yielded by the BPY method, while Table 4.6 shows the results obtained by inverting the Laplace transform with UniG, UniT and UniE respectively. The performance of each method is similar to that observed in the computation of the density of \( C_1 \).

### Table 4.5. Accuracy of the BPY method when computing the density of \( C_2 \)

<table>
<thead>
<tr>
<th>( x )</th>
<th>Reference</th>
<th>BPY ( n_0 = 20, , wp = 60 )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accu</td>
<td>Density</td>
</tr>
<tr>
<td>0.2</td>
<td>43</td>
<td>0.001619982191</td>
</tr>
<tr>
<td>0.3</td>
<td>42</td>
<td>0.02471847202</td>
</tr>
<tr>
<td>0.4</td>
<td>43</td>
<td>0.08500356201</td>
</tr>
<tr>
<td>0.5</td>
<td>44</td>
<td>0.1653318194</td>
</tr>
<tr>
<td>0.6</td>
<td>44</td>
<td>0.2449322823</td>
</tr>
<tr>
<td>0.7</td>
<td>44</td>
<td>0.3127392552</td>
</tr>
<tr>
<td>0.8</td>
<td>44</td>
<td>0.3653145797</td>
</tr>
<tr>
<td>0.9</td>
<td>44</td>
<td>0.4030148761</td>
</tr>
<tr>
<td>1.0</td>
<td>44</td>
<td>0.4276456023</td>
</tr>
<tr>
<td>1.5</td>
<td>43</td>
<td>0.4244815797</td>
</tr>
<tr>
<td>2.0</td>
<td>43</td>
<td>0.3336908021</td>
</tr>
<tr>
<td>2.5</td>
<td>43</td>
<td>0.2365328141</td>
</tr>
<tr>
<td>3.0</td>
<td>44</td>
<td>0.1581107640</td>
</tr>
<tr>
<td>4.0</td>
<td>43</td>
<td>0.06378916026</td>
</tr>
</tbody>
</table>

BPY: series expansion (3.39) derived by Biane, Pitman and Yor (2001).

wp: computing precision. Accu: accuracy measured by the number of significant digits. ED: effective number of digits of precision in the result. CPU: computation time in seconds.

The reference density is obtained by digits matching with the following option specifications for our code `DensityC[ ]`:

\`
(Method → "Biane", "n0" → 100), WorkingPrecision → 80) vs
(Method → "UniG", "M" → 100), WorkingPrecision → 200).
\`
Table 4.6. Accuracy of UniG, UniT and UniE when computing the density of $C_2$

<table>
<thead>
<tr>
<th>$x$</th>
<th>Reference Density</th>
<th>UniG $M = 45$, wp = 70</th>
<th>UniT $M = 25$, wp = 25</th>
<th>UniE $M = 25$, wp = 30</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accu</td>
<td>ED</td>
<td>Accu</td>
<td>CPU</td>
</tr>
<tr>
<td>0.2</td>
<td>43</td>
<td>0.001611992191</td>
<td>33.8</td>
<td>13</td>
</tr>
<tr>
<td>0.3</td>
<td>42</td>
<td>0.02471847202</td>
<td>32.4</td>
<td>14</td>
</tr>
<tr>
<td>0.4</td>
<td>43</td>
<td>0.08500356201</td>
<td>31.4</td>
<td>14</td>
</tr>
<tr>
<td>0.5</td>
<td>44</td>
<td>0.165318194</td>
<td>30.7</td>
<td>15</td>
</tr>
<tr>
<td>0.6</td>
<td>44</td>
<td>0.244932283</td>
<td>30.1</td>
<td>15</td>
</tr>
<tr>
<td>0.7</td>
<td>44</td>
<td>0.312739255</td>
<td>29.6</td>
<td>15</td>
</tr>
<tr>
<td>0.8</td>
<td>44</td>
<td>0.365314579</td>
<td>29.2</td>
<td>14</td>
</tr>
<tr>
<td>0.9</td>
<td>44</td>
<td>0.4030148761</td>
<td>28.9</td>
<td>16</td>
</tr>
<tr>
<td>1</td>
<td>44</td>
<td>0.4276456023</td>
<td>28.6</td>
<td>15</td>
</tr>
<tr>
<td>1.5</td>
<td>43</td>
<td>0.424481979</td>
<td>27.5</td>
<td>15</td>
</tr>
<tr>
<td>2</td>
<td>43</td>
<td>0.3336908021</td>
<td>26.8</td>
<td>15</td>
</tr>
<tr>
<td>2.5</td>
<td>43</td>
<td>0.2365328141</td>
<td>26.3</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>44</td>
<td>0.1581107640</td>
<td>25.8</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td>43</td>
<td>0.06378916026</td>
<td>25.1</td>
<td>14</td>
</tr>
</tbody>
</table>


The reference density is obtained by digits matching with the following option specifications for our code DensityC[]:

```
(Method -> "Biane", "Mn" -> 100), WorkingPrecision -> 80) vs
(Method -> "UniG", "M" -> 100), WorkingPrecision -> 200).
```

The densities of $C_{1/2}$, $C_1$ and $C_2$ are plotted in Figure 4.1. As $t$ increases, the right tail of the distribution of $C_t$ becomes bigger and longer.

![Figure 4.1. Densities of $C_{1/2}$, $C_1$ and $C_2$](image)

### 4.2. Computing the density of $S_t$

Biane, Pitman and Yor (2001) claim a complicated formula for the density $f_{S_t}(x)$ is obtained in Biane and Yor (1987). Instead of quoting the formula for $f_{S_t}(x)$ from Biane and Yor’s (1987) paper, we follow the same method used for deriving the formula for $f_{C_t}(x)$ to find the formula for $f_{S_t}(x)$. By inverting the series expansion of the Laplace transform of $S_t$ using Mathematica, we obtain expression (3.41) for the density $f_{S_t}(x)$ in terms of the Kummer confluent hypergeometric function. We call this the Cao method.
In addition, Tolmatz (2002) proposes the formula (3.45) in terms of the parabolic cylinder function for the density of \( S_{1/2} \), and Devroye (2009a) presents the formula in (3.49) for the density of \( S_1 \). To refer to them clearly, the formulae (3.45) and (3.49) are denoted by the Tolmatz method and the Devroye method respectively.

Apart from the analytical formulae in the form of an infinite series, numerical inversion techniques such as UniG, UniT and UniE can be used to invert the Laplace transform of \( S_t \) in (3.2). We find UniG can compute the density \( f_{S_t}(x) \) for any \( t > 0 \), while UniT and UniE can compute the density \( f_{S_t}(x) \) for integer \( t > 0 \). If the Laplace transform is written as a series expansion in (3.35), all three algorithms are able to invert it for any \( t > 0 \).

\[ \textbf{4.2.1 The density of } S_{1/2} \]

Table 4.7 shows the accuracy of the Cao method, the Tolmatz method and UniG in the computation of the density \( f_{S_{1/2}}(x) \). The working precision for the analytical formulae is set to \( wp = 20 \) so as to reduce the computation time. Hence, the accuracy of them decreases to about 18 significant digits. UniG appears to be competitive. The computation time is on average 0.03 seconds making UniG faster than the Cao method and equal to the Tolmatz method. UniT and UniE take a little bit longer to invert the series expansion of the Laplace transform with 0.05 seconds and 0.09 seconds respectively as shown in Table 4.8.
Table 4.7. Accuracy of the Cao method, the Tolmatz method and UniG when computing the density of $S_{1/2}$

<table>
<thead>
<tr>
<th>$x$</th>
<th>Reference</th>
<th>Cao</th>
<th>Tolmatz</th>
<th>UniG</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Density</td>
<td>Ed</td>
<td>Accu</td>
<td>CPU</td>
</tr>
<tr>
<td>0.01</td>
<td>0.007322886205</td>
<td>18.8</td>
<td>15</td>
<td>0.05</td>
</tr>
<tr>
<td>0.02</td>
<td>0.9341863782</td>
<td>19.0</td>
<td>17</td>
<td>0.05</td>
</tr>
<tr>
<td>0.03</td>
<td>3.284581237</td>
<td>19.5</td>
<td>18</td>
<td>0.03</td>
</tr>
<tr>
<td>0.04</td>
<td>5.157178444</td>
<td>18.7</td>
<td>18</td>
<td>0.05</td>
</tr>
<tr>
<td>0.05</td>
<td>6.073097982</td>
<td>18.8</td>
<td>18</td>
<td>0.03</td>
</tr>
<tr>
<td>0.1</td>
<td>4.002705945</td>
<td>18.9</td>
<td>18</td>
<td>0.05</td>
</tr>
<tr>
<td>0.15</td>
<td>3.047084035</td>
<td>18.8</td>
<td>18</td>
<td>0.03</td>
</tr>
<tr>
<td>0.2</td>
<td>1.937112130</td>
<td>18.7</td>
<td>18</td>
<td>0.05</td>
</tr>
<tr>
<td>0.3</td>
<td>0.8678372001</td>
<td>18.6</td>
<td>18</td>
<td>0.05</td>
</tr>
<tr>
<td>0.4</td>
<td>0.4393770287</td>
<td>18.5</td>
<td>18</td>
<td>0.05</td>
</tr>
<tr>
<td>0.5</td>
<td>0.2328932331</td>
<td>18.3</td>
<td>18</td>
<td>0.05</td>
</tr>
<tr>
<td>0.6</td>
<td>0.127466364</td>
<td>18.2</td>
<td>18</td>
<td>0.05</td>
</tr>
<tr>
<td>0.7</td>
<td>0.0711618895</td>
<td>18.0</td>
<td>17</td>
<td>0.05</td>
</tr>
<tr>
<td>0.8</td>
<td>0.0402923236</td>
<td>17.8</td>
<td>17</td>
<td>0.05</td>
</tr>
<tr>
<td>0.9</td>
<td>0.0230434549</td>
<td>17.6</td>
<td>17</td>
<td>0.05</td>
</tr>
<tr>
<td>1.0</td>
<td>0.01327961503</td>
<td>17.5</td>
<td>17</td>
<td>0.06</td>
</tr>
</tbody>
</table>


wp: computing precision. Accu: accuracy measured by the number of significant digits. ED: effective number of digits of precision in the result. CPU: computation time in seconds.

The reference density is obtained by digits matching with the following option specifications for our code DensityS[ ]:

- (Method → "Cao", "n0" → 100), WorkingPrecision → 80
- (Method → "UniG", "M" → 100), WorkingPrecision → 200.

Table 4.8. Accuracy of UniT and UniE when computing the density of $S_{1/2}$

<table>
<thead>
<tr>
<th>$x$</th>
<th>Reference</th>
<th>UniT</th>
<th>UniE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Density</td>
<td>Ed</td>
<td>Accu</td>
</tr>
<tr>
<td>0.01</td>
<td>0.007322886205</td>
<td>23.6</td>
<td>13</td>
</tr>
<tr>
<td>0.02</td>
<td>0.9341863782</td>
<td>23.5</td>
<td>15</td>
</tr>
<tr>
<td>0.03</td>
<td>3.284581237</td>
<td>23.1</td>
<td>16</td>
</tr>
<tr>
<td>0.04</td>
<td>5.157178444</td>
<td>22.8</td>
<td>16</td>
</tr>
<tr>
<td>0.05</td>
<td>6.073097982</td>
<td>22.5</td>
<td>18</td>
</tr>
<tr>
<td>0.1</td>
<td>4.002705945</td>
<td>21.7</td>
<td>18</td>
</tr>
<tr>
<td>0.15</td>
<td>3.047084035</td>
<td>21.2</td>
<td>17</td>
</tr>
<tr>
<td>0.2</td>
<td>1.937112130</td>
<td>20.9</td>
<td>16</td>
</tr>
<tr>
<td>0.3</td>
<td>0.8678372001</td>
<td>20.5</td>
<td>15</td>
</tr>
<tr>
<td>0.4</td>
<td>0.4393770287</td>
<td>20.1</td>
<td>15</td>
</tr>
<tr>
<td>0.5</td>
<td>0.2328932331</td>
<td>19.8</td>
<td>15</td>
</tr>
<tr>
<td>0.6</td>
<td>0.127466364</td>
<td>19.6</td>
<td>14</td>
</tr>
<tr>
<td>0.7</td>
<td>0.0711618895</td>
<td>19.4</td>
<td>15</td>
</tr>
<tr>
<td>0.8</td>
<td>0.0402923236</td>
<td>19.0</td>
<td>14</td>
</tr>
<tr>
<td>0.9</td>
<td>0.0230434549</td>
<td>18.9</td>
<td>14</td>
</tr>
<tr>
<td>1.0</td>
<td>0.01327961503</td>
<td>18.7</td>
<td>14</td>
</tr>
</tbody>
</table>


wp: computing precision. Accu: accuracy measured by the number of significant digits. ED: effective number of digits of precision in the result. CPU: computation time in seconds.

The reference density is obtained by digits matching with the following option specifications for our code DensityS[ ]:

- (Method → "Cao", "n0" → 100), WorkingPrecision → 80
- (Method → "UniG", "M" → 100), WorkingPrecision → 200.

#### 4.2.2 The density of $S_1$

For the density $f_{S_1}(x)$, the accuracy of the Cao method, the Devroye method, UniG, UniT and UniE is tabulated in Table 4.9 and Table 4.10. The Devroye method becomes the fastest method which computes
the density almost instantly. UniT and UniE are the second fast methods taking less than 0.02 seconds.

The Cao method and UniG spend 0.02–0.03 seconds in the computation.

<table>
<thead>
<tr>
<th>x2</th>
<th>Cao Accu</th>
<th>Cao Density</th>
<th>Devroye CPU</th>
<th>Devroye Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.05</td>
<td>40</td>
<td>0.06155932327</td>
<td>8.8</td>
<td>0.0615590.02</td>
</tr>
<tr>
<td>0.1</td>
<td>42</td>
<td>1.530059698</td>
<td>18.9</td>
<td>1.53010.02</td>
</tr>
<tr>
<td>0.15</td>
<td>45</td>
<td>2.77463394</td>
<td>18.9</td>
<td>2.77640.02</td>
</tr>
<tr>
<td>0.2</td>
<td>45</td>
<td>2.928996579</td>
<td>19.0</td>
<td>2.92900.02</td>
</tr>
<tr>
<td>0.25</td>
<td>44</td>
<td>2.591569795</td>
<td>19.0</td>
<td>2.59160.02</td>
</tr>
<tr>
<td>0.3</td>
<td>44</td>
<td>2.140028564</td>
<td>19.0</td>
<td>2.14000.02</td>
</tr>
<tr>
<td>0.35</td>
<td>44</td>
<td>1.715242368</td>
<td>19.0</td>
<td>1.71520.02</td>
</tr>
<tr>
<td>0.4</td>
<td>44</td>
<td>1.350289959</td>
<td>19.0</td>
<td>1.35030.02</td>
</tr>
<tr>
<td>0.45</td>
<td>43</td>
<td>1.06574213</td>
<td>19.0</td>
<td>1.06570.02</td>
</tr>
<tr>
<td>0.5</td>
<td>43</td>
<td>0.834949601</td>
<td>18.9</td>
<td>0.834950.02</td>
</tr>
<tr>
<td>0.6</td>
<td>43</td>
<td>0.510690267</td>
<td>18.9</td>
<td>0.510700.02</td>
</tr>
<tr>
<td>0.7</td>
<td>43</td>
<td>0.3119139020</td>
<td>18.9</td>
<td>0.311910.02</td>
</tr>
<tr>
<td>0.8</td>
<td>43</td>
<td>0.1900441027</td>
<td>18.2</td>
<td>0.190040.02</td>
</tr>
<tr>
<td>0.9</td>
<td>43</td>
<td>0.1162666815</td>
<td>18.6</td>
<td>0.116270.02</td>
</tr>
<tr>
<td>1</td>
<td>41</td>
<td>0.0709809300</td>
<td>18.4</td>
<td>0.0709810.02</td>
</tr>
</tbody>
</table>

Cao: the formula (3.41). Devroye: the formula (3.49).

The reference method is obtained by digits matching with the following option specifications for our code DensityS[]:

(Method → "Cao", "n0" → 100), WorkingPrecision → 80 vs (Method → "UniG", "M" → 100), WorkingPrecision → 200).

<table>
<thead>
<tr>
<th>x</th>
<th>Reference</th>
<th>UniG M = 35, wp = 70</th>
<th>UniT M = 25, wp = 25</th>
<th>UniE M = 25, wp = 30</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.05</td>
<td>40</td>
<td>0.06155932327</td>
<td>33.5</td>
<td>0.03</td>
</tr>
<tr>
<td>0.1</td>
<td>42</td>
<td>1.530059698</td>
<td>30.9</td>
<td>0.03</td>
</tr>
<tr>
<td>0.15</td>
<td>44</td>
<td>2.77463394</td>
<td>29.5</td>
<td>0.03</td>
</tr>
<tr>
<td>0.2</td>
<td>45</td>
<td>2.928996579</td>
<td>28.5</td>
<td>0.03</td>
</tr>
<tr>
<td>0.25</td>
<td>44</td>
<td>2.591569795</td>
<td>27.8</td>
<td>0.03</td>
</tr>
<tr>
<td>0.3</td>
<td>44</td>
<td>2.140028564</td>
<td>27.3</td>
<td>0.03</td>
</tr>
<tr>
<td>0.35</td>
<td>44</td>
<td>1.715242368</td>
<td>26.9</td>
<td>0.03</td>
</tr>
<tr>
<td>0.4</td>
<td>44</td>
<td>1.350289959</td>
<td>26.5</td>
<td>0.03</td>
</tr>
<tr>
<td>0.45</td>
<td>43</td>
<td>1.06574213</td>
<td>26.1</td>
<td>0.03</td>
</tr>
<tr>
<td>0.5</td>
<td>43</td>
<td>0.834949601</td>
<td>25.8</td>
<td>0.03</td>
</tr>
<tr>
<td>0.6</td>
<td>43</td>
<td>0.510690267</td>
<td>25.3</td>
<td>0.03</td>
</tr>
<tr>
<td>0.7</td>
<td>43</td>
<td>0.3119139020</td>
<td>24.9</td>
<td>0.03</td>
</tr>
<tr>
<td>0.8</td>
<td>44</td>
<td>0.1900441027</td>
<td>24.6</td>
<td>0.03</td>
</tr>
<tr>
<td>0.9</td>
<td>43</td>
<td>0.1162666815</td>
<td>24.2</td>
<td>0.03</td>
</tr>
<tr>
<td>1</td>
<td>41</td>
<td>0.0709809300</td>
<td>23.9</td>
<td>0.03</td>
</tr>
</tbody>
</table>


The reference density is obtained by digits matching with the following option specifications for our code DensityC[]:

(Method → "Cao", "n0" → 100), WorkingPrecision → 80 vs (Method → "UniG", "M" → 100), WorkingPrecision → 200).

4.2.3 The density of $S_2$
Methods which are suitable to compute the density \( f_{S_2}(x) \) include the Cao method, UniG, UniT and UniE with their results tabulated in Table 4.11 and Table 4.12. UniT and UniE turn faster than the Cao method and UniG. The densities of \( S_{1/2}, S_1 \) and \( S_2 \) are plotted in Figure 4.2.

Table 4.11. Accuracy of the Cao method when computing the density of \( S_2 \)

<table>
<thead>
<tr>
<th>x</th>
<th>Reference</th>
<th>Cao n=20, wp=20</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accu</td>
<td>Density</td>
</tr>
<tr>
<td>0.15</td>
<td>39</td>
<td>0.01403834847</td>
</tr>
<tr>
<td>0.2</td>
<td>40</td>
<td>0.1376984863</td>
</tr>
<tr>
<td>0.25</td>
<td>40</td>
<td>0.4453869917</td>
</tr>
<tr>
<td>0.3</td>
<td>40</td>
<td>0.8514140773</td>
</tr>
<tr>
<td>0.35</td>
<td>41</td>
<td>1.224338414</td>
</tr>
<tr>
<td>0.4</td>
<td>42</td>
<td>1.487573776</td>
</tr>
<tr>
<td>0.5</td>
<td>42</td>
<td>1.653594508</td>
</tr>
<tr>
<td>0.6</td>
<td>42</td>
<td>1.498835055</td>
</tr>
<tr>
<td>0.7</td>
<td>42</td>
<td>1.220309746</td>
</tr>
<tr>
<td>0.8</td>
<td>43</td>
<td>0.9325240775</td>
</tr>
<tr>
<td>0.9</td>
<td>41</td>
<td>0.6839846978</td>
</tr>
<tr>
<td>1.</td>
<td>41</td>
<td>0.4876155424</td>
</tr>
<tr>
<td>1.2</td>
<td>42</td>
<td>0.233952441</td>
</tr>
<tr>
<td>1.5</td>
<td>40</td>
<td>0.07105716236</td>
</tr>
<tr>
<td>2.</td>
<td>39</td>
<td>0.0085451544948</td>
</tr>
</tbody>
</table>

Cao: the formula (3.41).
wp: computing precision. Accu: accuracy measured by the number of significant digits. ED: effective number of digits of precision in the result. CPU: computation time in seconds.
The reference density is obtained by digits matching with the following option specifications for our code DensityS[ ]:
(Method -> ("Cao", "n0" -> 100), WorkingPrecision -> 80) vs (Method -> ("UniG", "M" -> 100), WorkingPrecision -> 200).

Table 4.12. Accuracy of UniG, UniT and UniE when computing the density of \( S_2 \)

<table>
<thead>
<tr>
<th>x</th>
<th>Reference</th>
<th>UniG M=35, wp=70</th>
<th>UniT M=25, wp=25</th>
<th>UniE M=25, wp=30</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accu</td>
<td>Density</td>
<td>ED Accu CPU</td>
<td>ED Accu CPU</td>
</tr>
<tr>
<td>0.15</td>
<td>39</td>
<td>0.01403834847</td>
<td>34.3 12 0.03</td>
<td>23.0 13 0</td>
</tr>
<tr>
<td>0.2</td>
<td>40</td>
<td>0.1376984863</td>
<td>32.2 14 0.03</td>
<td>23.0 14 0.02</td>
</tr>
<tr>
<td>0.25</td>
<td>40</td>
<td>0.4453869917</td>
<td>32.3 12 0.02</td>
<td>22.8 14 0.02</td>
</tr>
<tr>
<td>0.3</td>
<td>40</td>
<td>0.8514140773</td>
<td>31.6 12 0.03</td>
<td>22.7 15 0</td>
</tr>
<tr>
<td>0.35</td>
<td>41</td>
<td>1.224338414</td>
<td>31.0 13 0.03</td>
<td>22.4 16 0</td>
</tr>
<tr>
<td>0.4</td>
<td>42</td>
<td>1.487573776</td>
<td>30.4 14 0.03</td>
<td>22.3 15 0</td>
</tr>
<tr>
<td>0.5</td>
<td>42</td>
<td>1.653594508</td>
<td>29.5 14 0.03</td>
<td>21.9 15 0.02</td>
</tr>
<tr>
<td>0.6</td>
<td>42</td>
<td>1.498835055</td>
<td>28.8 13 0.03</td>
<td>21.5 15 0</td>
</tr>
<tr>
<td>0.7</td>
<td>42</td>
<td>1.220309746</td>
<td>28.2 14 0.02</td>
<td>21.3 16 0.02</td>
</tr>
<tr>
<td>0.8</td>
<td>43</td>
<td>0.9325240775</td>
<td>27.7 12 0.02</td>
<td>21.0 15 0.02</td>
</tr>
<tr>
<td>0.9</td>
<td>41</td>
<td>0.6839846978</td>
<td>27.2 13 0.02</td>
<td>20.7 14 0.02</td>
</tr>
<tr>
<td>1.</td>
<td>41</td>
<td>0.4876155424</td>
<td>26.8 13 0.02</td>
<td>20.5 15 0.02</td>
</tr>
<tr>
<td>1.2</td>
<td>42</td>
<td>0.233952441</td>
<td>26.0 14 0.02</td>
<td>20.1 16 0.02</td>
</tr>
<tr>
<td>1.5</td>
<td>40</td>
<td>0.07105716236</td>
<td>25.0 13 0.03</td>
<td>19.3 15 0</td>
</tr>
<tr>
<td>2.</td>
<td>39</td>
<td>0.0085451544948</td>
<td>23.6 12 0.02</td>
<td>18.4 13 0.02</td>
</tr>
</tbody>
</table>

wp: computing precision. Accu: accuracy measured by the number of significant digits. ED: effective number of digits of precision in the result. CPU: computation time in seconds.
The reference density is obtained by digits matching with the following option specifications for our code DensityS[ ]:
(Method -> ("Cao", "n0" -> 100), WorkingPrecision -> 80) vs (Method -> ("UniG", "M" -> 100), WorkingPrecision -> 200).
4.3. Computing the density of $T_I$

There is no available analytical expression proposed for the density of $T_I$ other than for $t = 1$. Numerical inversion is adopted to compute the density $f_{T_I}(x)$. Unlike the Laplace transforms of $C_t$ and $S_t$, all three inversion algorithms UniG, UniT and UniE can invert the Laplace transform of $T_I$ in (3.3) for any $t > 0$.

4.3.1 The density of $T_{1/2}$

As shown in Table 4.13, UniT and UniE takes 0–0.02 seconds for computing the density $f_{T_{1/2}}(x)$ with UniT marginally faster than UniE. Although UniG is slower than UniT and UniE, it is still very fast taking about 0.03 seconds.

![Figure 4.2: Densities of $S_{1/2}$, $S_1$ and $S_2$](image)

Table 4.13. Accuracy of UniG, UniT and UniE when computing the density of $T_{1/2}$

<table>
<thead>
<tr>
<th>$x$</th>
<th>Reference</th>
<th>UniG $M = 35$, wp = 70</th>
<th>UniT $M = 25$, wp = 25</th>
<th>UniE $M = 25$, wp = 30</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accu</td>
<td>ED Accu CPU</td>
<td>ED Accu CPU</td>
<td>ED Accu CPU</td>
</tr>
<tr>
<td>0.001</td>
<td>57</td>
<td>23.8 23 0.03</td>
<td>20.7 14 0.02</td>
<td>20.6 15 0.02</td>
</tr>
<tr>
<td>0.005</td>
<td>58</td>
<td>23.8 23 0.03</td>
<td>20.7 14 0.02</td>
<td>20.6 15 0.02</td>
</tr>
<tr>
<td>0.01</td>
<td>56</td>
<td>23.8 23 0.03</td>
<td>20.7 13 0.02</td>
<td>20.6 14 0.02</td>
</tr>
<tr>
<td>0.05</td>
<td>49</td>
<td>23.8 22 0.03</td>
<td>20.7 14 0.02</td>
<td>20.6 15 0.02</td>
</tr>
<tr>
<td>0.1</td>
<td>46</td>
<td>23.8 18 0.03</td>
<td>20.7 15 0.02</td>
<td>20.6 15 0.02</td>
</tr>
<tr>
<td>0.2</td>
<td>45</td>
<td>23.8 16 0.03</td>
<td>20.7 14 0.02</td>
<td>20.6 14 0.02</td>
</tr>
<tr>
<td>0.3</td>
<td>46</td>
<td>23.8 16 0.03</td>
<td>20.7 13 0.02</td>
<td>20.6 15 0.02</td>
</tr>
<tr>
<td>0.4</td>
<td>46</td>
<td>23.8 15 0.02</td>
<td>20.7 13 0.02</td>
<td>20.6 15 0.02</td>
</tr>
<tr>
<td>0.5</td>
<td>46</td>
<td>23.8 17 0.03</td>
<td>20.7 14 0.02</td>
<td>20.6 15 0.02</td>
</tr>
<tr>
<td>0.6</td>
<td>46</td>
<td>23.7 17 0.03</td>
<td>20.6 14 0.02</td>
<td>20.6 15 0.02</td>
</tr>
<tr>
<td>0.7</td>
<td>46</td>
<td>23.7 16 0.03</td>
<td>20.6 14 0.02</td>
<td>20.5 15 0.02</td>
</tr>
<tr>
<td>0.8</td>
<td>46</td>
<td>23.7 17 0.03</td>
<td>20.6 13 0.02</td>
<td>20.5 15 0.02</td>
</tr>
<tr>
<td>0.9</td>
<td>46</td>
<td>23.6 17 0.03</td>
<td>20.5 14 0.02</td>
<td>20.5 14 0.02</td>
</tr>
<tr>
<td>1.0</td>
<td>46</td>
<td>23.6 15 0.02</td>
<td>20.5 13 0.02</td>
<td>20.4 16 0.02</td>
</tr>
<tr>
<td>2.0</td>
<td>45</td>
<td>23.2 16 0.03</td>
<td>20.0 13 0.02</td>
<td>20.0 14 0.02</td>
</tr>
</tbody>
</table>


wp: computing precision. Accu: accuracy measured by the number of significant digits. ED: effective number of digits of precision in the result. CPU: computation time in seconds.

The reference density is obtained by digits matching with the following option specifications for our code DensityT1/2:

```
(Method -> ("UniG", "M" -> 100), WorkingPrecision -> 200) \&
(Method -> ("UniT", "M" -> 100), WorkingPrecision -> 100).
```
### 4.3.2 The density of \( T_1 \)

The density \( f_{T_1}(x) \) is tabulated in Table 4.14 showing that UniT and UniE are better than UniG.

<table>
<thead>
<tr>
<th>( x )</th>
<th>Reference</th>
<th>UniG ( M = 35, \ wp = 70 )</th>
<th>UniT ( M = 25, \ wp = 25 )</th>
<th>UniE ( M = 25, \ wp = 30 )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accu</td>
<td>Density</td>
<td>ED Accu CPU</td>
<td>ED Accu CPU</td>
</tr>
<tr>
<td>0.001</td>
<td>58</td>
<td>12.61566261</td>
<td>24.2 24 0.03</td>
<td>21.0 15 0.02</td>
</tr>
<tr>
<td>0.005</td>
<td>58</td>
<td>5.641895835</td>
<td>24.2 24 0.03</td>
<td>21.0 14 0</td>
</tr>
<tr>
<td>0.01</td>
<td>59</td>
<td>3.989422804</td>
<td>24.2 24 0.03</td>
<td>21.0 15 0.02</td>
</tr>
<tr>
<td>0.05</td>
<td>50</td>
<td>1.784124116</td>
<td>24.2 22 0.03</td>
<td>21.0 15 0</td>
</tr>
<tr>
<td>0.1</td>
<td>47</td>
<td>1.261566256</td>
<td>24.2 20 0.03</td>
<td>21.0 15 0</td>
</tr>
<tr>
<td>0.2</td>
<td>45</td>
<td>0.8919810592</td>
<td>24.2 16 0.02</td>
<td>21.0 15 0.02</td>
</tr>
<tr>
<td>0.3</td>
<td>46</td>
<td>0.7265117350</td>
<td>24.2 17 0.03</td>
<td>21.0 14 0</td>
</tr>
<tr>
<td>0.4</td>
<td>46</td>
<td>0.6222827665</td>
<td>24.2 16 0.03</td>
<td>21.0 15 0</td>
</tr>
<tr>
<td>0.5</td>
<td>46</td>
<td>0.543527252</td>
<td>24.2 16 0.03</td>
<td>21.0 15 0.02</td>
</tr>
<tr>
<td>0.6</td>
<td>46</td>
<td>0.4782874222</td>
<td>24.1 16 0.03</td>
<td>21.0 15 0.02</td>
</tr>
<tr>
<td>0.7</td>
<td>45</td>
<td>0.4220667297</td>
<td>24.1 17 0.03</td>
<td>20.9 15 0</td>
</tr>
<tr>
<td>0.8</td>
<td>46</td>
<td>0.3728466156</td>
<td>24.1 17 0.03</td>
<td>20.9 15 0</td>
</tr>
<tr>
<td>0.9</td>
<td>46</td>
<td>0.3294957815</td>
<td>24.1 16 0.03</td>
<td>20.9 15 0.02</td>
</tr>
<tr>
<td>1.0</td>
<td>46</td>
<td>0.2912297957</td>
<td>24.0 18 0.03</td>
<td>20.8 15 0.02</td>
</tr>
<tr>
<td>2.0</td>
<td>45</td>
<td>0.08480497270</td>
<td>23.7 16 0.03</td>
<td>20.4 14 0</td>
</tr>
</tbody>
</table>


The reference density is obtained by digits matching with the following option specifications for our code DensityT1: (Method → ("UniG", "M" → 100), WorkingPrecision → 200) vs (Method → ("UniT", "M" → 100), WorkingPrecision → 100).

### 4.3.3 The density of \( T_2 \)

Table 4.15 shows UniT and UniE are still superior to UniG when computing the density \( f_{T_2}(x) \).

<table>
<thead>
<tr>
<th>( x )</th>
<th>Reference</th>
<th>UniG ( M = 35, \ wp = 70 )</th>
<th>UniT ( M = 25, \ wp = 25 )</th>
<th>UniE ( M = 25, \ wp = 30 )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accu</td>
<td>Density</td>
<td>ED Accu CPU</td>
<td>ED Accu CPU</td>
</tr>
<tr>
<td>0.02</td>
<td>50</td>
<td>0.5000000000</td>
<td>24.2 21 0.03</td>
<td>21.1 16 0</td>
</tr>
<tr>
<td>0.2</td>
<td>48</td>
<td>0.4999999999</td>
<td>24.2 17 0.03</td>
<td>21.1 16 0</td>
</tr>
<tr>
<td>0.6</td>
<td>47</td>
<td>0.4999999999</td>
<td>24.2 17 0.03</td>
<td>21.1 16 0</td>
</tr>
<tr>
<td>0.8</td>
<td>48</td>
<td>0.4999999999</td>
<td>24.2 17 0.03</td>
<td>21.1 16 0</td>
</tr>
<tr>
<td>1.2</td>
<td>47</td>
<td>0.386264496</td>
<td>24.1 17 0.03</td>
<td>20.9 15 0</td>
</tr>
<tr>
<td>1.4</td>
<td>47</td>
<td>0.309526597</td>
<td>24.1 16 0.03</td>
<td>20.9 16 0</td>
</tr>
<tr>
<td>1.6</td>
<td>46</td>
<td>0.278554916</td>
<td>24.1 17 0.03</td>
<td>20.9 16 0</td>
</tr>
<tr>
<td>1.8</td>
<td>47</td>
<td>0.239557495</td>
<td>24.1 17 0.03</td>
<td>20.9 16 0</td>
</tr>
<tr>
<td>2.0</td>
<td>46</td>
<td>0.203960162</td>
<td>24.1 17 0.03</td>
<td>20.9 16 0</td>
</tr>
<tr>
<td>2.2</td>
<td>47</td>
<td>0.160538249</td>
<td>24.1 17 0.03</td>
<td>20.9 16 0</td>
</tr>
<tr>
<td>2.4</td>
<td>47</td>
<td>0.084097498</td>
<td>24.1 17 0.03</td>
<td>20.9 16 0</td>
</tr>
<tr>
<td>2.6</td>
<td>47</td>
<td>0.05204631782</td>
<td>23.9 16 0.03</td>
<td>20.8 14 0</td>
</tr>
<tr>
<td>2.8</td>
<td>47</td>
<td>0.03168229396</td>
<td>23.9 16 0.03</td>
<td>20.8 14 0</td>
</tr>
</tbody>
</table>


The reference density is obtained by digits matching with the following option specifications for our code DensityT2: (Method → ("UniG", "M" → 100), WorkingPrecision → 200) vs (Method → ("UniT", "M" → 100), WorkingPrecision → 100).
The densities \( f_{T_1}^{(1)}(x) \), \( f_{T_1}(x) \) and \( f_{T_2}(x) \) are plotted in Figure 4.3. When \( x \to 0 \), the densities \( f_{T_1}^{(1)}(x) \) and \( f_{T_2}(x) \) approach infinity, while the density \( f_{T_2}(x) \) approaches the value of 0.5.

![Figure 4.3. Densities of \( T_{1/2}, T_1 \) and \( T_2 \)](image)

- **4.3.4 Plots of the densities of \( C_t, S_t \) and \( T_t \)**

To compare the infinitely divisible distributions for the same value of \( t \), we draw the curves of the densities of \( C_t, S_t \) and \( T_t \) for \( t = 1/2 \), \( t = 1 \) and \( t = 2 \) respectively in Figure 4.4, Figure 4.5 and Figure 4.6.

![Figure 4.4. Densities of \( C_{1/2}, S_{1/2} \) and \( T_{1/2} \)](image)
Chapter 3. A Comparison of Various Methods for Computing Stable Distributions and Distributions of $C_t$, $S_t$ and $T_t$ | 125

Figure 4.5. Densities of $C_1$, $S_1$ and $T_1$

Figure 4.6. Densities of $C_2$, $S_2$ and $T_2$
5. Stable Distributions

- 5.1. The Definition of $S(\alpha, \beta, \mu, \sigma)$

The stable distributions, also called $\alpha$-stable distributions, are introduced by the French mathematician Paul Lévy in a monograph in 1925, as cited in the text by Borak, Härdle and Weron (2005). They have the property that a linear combination of two i.i.d. stable random variables is also stable. The stable distribution denoted by $S(\alpha, \beta, \mu, \sigma)$ can be represented by its characteristic function (Weron, 1996):

$$
\log \phi(t) = \begin{cases} 
-\sigma^\alpha |t|^{\alpha} \left(1 - i \beta \text{sign}(t) \tan \frac{\pi \alpha}{2}\right) + i \mu t, & \alpha \neq 1, \\
-\sigma |t| \left(1 + i \beta \text{sign}(t) \frac{2}{\pi} \ln |t|\right) + i \mu t, & \alpha = 1,
\end{cases}
$$

(5.1)

where $\alpha \in (0, 2]$, $\beta \in [-1, 1]$, $\mu \in \mathbb{R}$ and $\sigma > 0$.

As we can see, the description of a stable distribution requires four parameters: the index of stability $\alpha$, the skewness parameter $\beta$, the location parameter $\mu$ and the scale parameter $\sigma$. The index of stability $\alpha$ is also called the tail exponent which determines the decrease rate of the tails of the distribution. The location parameter $\mu$ indicates the shift of the mode (the peak) of the density. When $\mu = 0$ and $\sigma = 1$, $S(\alpha, \beta, 0, 1)$ is called a standard stable distribution and the notation can be abbreviated to $S(\alpha, \beta)$.

Further, $S(\alpha, 0, 0, 1)$ can be written as $S(\alpha)$.

Borak, Härdle and Weron (2005) give a brief description of the stable distribution for various values of the parameters. When $\alpha = 2$, the stable distribution is a normal distribution with $\beta$ irrelevant. When $\alpha < 2$, the variance of the distribution is infinite. When $\alpha > 1$, the mean of the distribution exists and is equal to $\mu$ regardless of $\beta$ and $\sigma$. The distribution is right-skewed for a positive $\beta$, i.e. the right tail is longer, whereas it is left-skewed for a negative $\beta$. When $\beta = 0$, the distribution is symmetric about $\mu$.

There are closed-form expressions, which have a finite number of elementary functions, for the densities of stable distributions in only three cases (see Weron, 1996): $S(2, 0, \mu, \sigma)$, $S(1, 0, \mu, \sigma)$ and $S(1/2, 1, \mu, \sigma)$. Mathematica 9.0 has built-in function StableDistribution[] to represent the stable distribution $S(\alpha, \beta, \mu, \sigma)$ which allows us to compute its densities although the formulae adopted is concealed.

Using Mathematica 9.0 to verify the three special cases above, we find the distribution $S(2, 0, \mu, \sigma)$ is equivalent to the normal distribution $N(\mu, \sqrt{2}\sigma)$ with the density function given by
\[ f_{S(2,0,\mu,\sigma)}(x) = \frac{1}{2\sqrt{\pi}} e^{-\frac{(x-\mu)^2}{4\sigma^2}} \]  

(5.2)

The distribution \( S(1, 0, \mu, \sigma) \) is identical to the Cauchy distribution with location parameter \( \mu \) and scale parameter \( \sigma \). The density function of \( S(1, 0, \mu, \sigma) \) is therefore

\[ f_{S(1,0,\mu,\sigma)}(x) = \frac{1}{\pi(1+(\frac{x-\mu}{\sigma})^2)} \sigma \]  

(5.3)

While, the distribution \( S(1/2, 1, \mu, \sigma) \) is exactly the Lévy distribution with location parameter \( \mu \) and dispersion parameter \( \sigma \). The density function of \( S(1/2, 1, \mu, \sigma) \) is given by

\[
\begin{align*}
 f_{S(1/2,1,\mu,\sigma)}(x) &= \begin{cases} 
 e^{-\frac{(x-\mu)^2}{2\sigma^2}} & x > \mu \\
 0 & \text{otherwise}
\end{cases}
\end{align*}
\]  

(5.4)

### 5.2. The Definition of \( S_1(\alpha, \beta_1, \mu, \sigma_1) \)

An alternative parameterization of the characteristic function of a \( \alpha \)-stable random variable is given by (Weron, 1996)

\[ \log \phi(t) = \begin{cases} 
 -\sigma_1^\alpha |t|^{\alpha} \exp\{-i \beta_1 \text{sign}(t) \left( \frac{\pi \sigma}{2} K(\alpha) \right) + i \mu t, & \alpha \neq 1, \\
 -\sigma_1^\alpha |t|^{\alpha} \text{sign}(t) \ln |t| + i \mu t, & \alpha = 1,
\end{cases} \]  

(5.5)

where

\[ K(\alpha) = 1 - |1 - \alpha| = \begin{cases} 
 \alpha, & 0 < \alpha < 1 \\
 2 - \alpha, & 1 < \alpha \leq 2.
\end{cases} \]  

(5.6)

for \( \alpha \in (0, 2], \beta_1 \in [-1, 1], \mu \in \mathbb{R} \) and \( \sigma_1 > 0 \). Note we make a correction to the expression for \( K(\alpha) \).

Denote the stable distributions of this parameterization by \( S_1(\alpha, \beta_1, \mu, \sigma_1) \) which is related to \( S(\alpha, \beta, \mu, \sigma) \) by the following equations (Weron, 1996):

For \( \alpha \neq 1 \),

\[ \tan\left( \beta_1 \frac{\pi K(\alpha)}{2} \right) = \beta \tan\left( \frac{\pi \alpha}{2} \right) \]

\[ \sigma_1 = \sigma \left( 1 + \beta^2 \tan^2\left( \frac{\pi \sigma}{2} \right) \right)^{1/2} \]  

(5.7)

For \( \alpha = 1 \),

\[ \beta_1 = \beta \]

\[ \sigma_1 = \frac{2}{\pi} \sigma \]  

(5.8)
However we notice \( S_1(\alpha, 0, \mu, \sigma) = S(\alpha, 0, \mu, \sigma) \) for all \( \alpha \). This is justified in this way: when \( \beta_1 = 0 \), the c.f. of \( S_1(\alpha, \beta_1, \mu, \sigma_1) \) reduces to

\[
\log \phi(t) = -\sigma_1^\alpha |t|^\alpha + i \mu t
\] (5.9)

While, when \( \beta = 0 \), the c.f. of \( S(\alpha, \beta, \mu, \sigma) \) reduces to

\[
\log \phi(t) = -\sigma^\alpha |t|^\alpha + i \mu t
\] (5.10)

Therefore, \( S_1(\alpha, \beta_1, \mu, \sigma_1) \) is equivalent to \( S(\alpha, \beta, \mu, \sigma) \) when \( \beta_1 = \beta = 0 \) and \( \sigma_1 = \sigma \). Let \( S_1(\alpha, \beta_1) \) be the abbreviation of \( S_1(\alpha, \beta_1, 0, 1) \) and \( S_1(\alpha) \) be the abbreviation of \( S_1(\alpha, 0, 0, 1) \). In particular, we have \( S_1(\alpha) = S(\alpha) \).

Also, when \( \alpha = 2 \), the c.f. of \( S(\alpha, \beta, \mu, \sigma) \) becomes

\[
\log \phi(t) = -\sigma^\alpha |t|^\alpha + i \mu t
\] (5.11)

while, the c.f. of \( S_1(\alpha, \beta_1, \mu, \sigma_1) \) reduces to

\[
\log \phi(t) = -\sigma_1^\alpha |t|^\alpha \exp \left( -i \beta_1 \text{sign}(t) \frac{\pi}{2} (2 - \alpha) \right) + i \mu t
= -\sigma_1^\alpha |t|^\alpha + i \mu t
\] (5.12)

The values of \( \beta \) and \( \beta_1 \) become irrelevant for the relation between \( S(\alpha, \beta, \mu, \sigma) \) and \( S_1(\alpha, \beta_1, \mu, \sigma_1) \).

Thus, we have \( S_1(2, \beta_1, \mu, \sigma_1) = S(2, \beta, \mu, \sigma) \) when \( \sigma_1 = \sigma \).

To summarize, \( S_1(\alpha, \beta_1, \mu, \sigma_1) \) is equivalent to \( S(\alpha, \beta, \mu, \sigma) \) when the following equations hold

For \( \alpha \neq 1, \alpha \neq 2 \) and \( \beta \neq 0 \)

\[
\tan \left( \beta_1 \frac{\pi K(\alpha)}{2} \right) = \beta \tan \left( \frac{\pi \alpha}{2} \right)
\]

\[
\sigma_1 = \sigma \left( 1 + \beta^2 \tan^2 \frac{\pi \alpha}{2} \right)^{1/(2 \alpha)}
\] (5.13)

For \( \beta = 0 \)

\[
\beta_1 = 0
\]

\[
\sigma_1 = \sigma
\] (5.14)

For \( \alpha = 1 \) and \( \beta \neq 0 \)

\[
\beta_1 = \beta
\]

\[
\sigma_1 = \frac{2}{\pi} \sigma
\] (5.15)

For \( \alpha = 2 \)
\[ \sigma_1 = \sigma \]  

(5.16)

In the case of \( \alpha = 2 \), the values of \( \beta \) and \( \beta_1 \) are irrelevant. For computational purpose, we may set \( \beta_1 = \beta = 0 \).

- **5.3. Generation of stable random variables** \( S_1(\alpha, \beta_1) \)

Kanter (1975) was the first to give a method for simulating \( S_1(\alpha, \beta_1) \) random variables for \( \alpha < 1 \) and \( \beta_1 = 1 \). Chambers, Mallows and Stuck (1976) generalized the Kanter method and provided a recipe for simulating \( S_1(\alpha, \beta_1) \) random variables for \( \alpha \in (0, 2] \) and \( \beta_1 \in [-1, 1] \). The recipe is the following:

For \( \alpha \neq 1 \),

\[ S_1(\alpha, \beta_1) = \frac{\sin(\Phi - \Phi_0)}{(\cos \Phi)^{1/\alpha}} \left( \frac{\cos(\Phi - \Phi_0)}{W} \right)^{(1-\alpha)/\alpha} \]

(5.17)

For \( \alpha = 1 \),

\[ S_1(1, \beta_1) = \frac{2}{\pi} \left( \frac{1}{2} \pi + \beta_1 \Phi \right) \tan \Phi - \beta_1 \ln \left( \frac{1}{2} \pi W \cos \Phi \right) \]

(5.18)

where \( W \) is a standard exponential random variable, \( \Phi \) is a uniform\((-\frac{1}{2} \pi, \frac{1}{2} \pi)\) random variable, and

\[ \Phi_0 = -\frac{1}{2} \pi \beta_1 \frac{k(\alpha)}{\alpha} \]

(5.19)

where \( k(\alpha) = 1 - |1 - \alpha| \).

Using Chambers-Mallows-Stuck method, we generate \( 10^6 \) random variates of each stable distribution \( S_1(2, 0), S_1(1, 0), S_1(1/2, 1), S_1(3/4, 1/2), S_1(4/3, 1/3) \) and \( S_1(3/2, -1) \) among which the first two are equivalent to the normal distribution and the Cauchy distribution respectively. The *Mathematica* built-in function `Histogram[ ]` permits us to plot histograms from the simulated data with respective distribution overlay curves in Figure 5.1. The distribution curves are plotted from \( S(\alpha, \beta, \mu, \sigma) \) with \( \beta \) and \( \sigma \) be the corresponding values such that \( S_1(\alpha, \beta_1, \mu, \sigma_1) = S(\alpha, \beta, \mu, \sigma) \). Recall that the variance of a stable distribution is infinite as \( \alpha < 2 \). Those distributions except \( S_1(2, 0) \) are heavy-tailed. Hence, the simulated data for them contains extremely large values which will distort the automatic bin specification for `Histogram[ ]`. To draw the correct histograms for such distributions, the bins are specified explicitly to gather data in the heavy tail into a single bin. For example, we set the bin delimiters for \( S_1(1, 0) \) as \([-10^{10}, -4, ..., 4, 10^{10}]\) with an interval of 0.05 between -4 and 4. The bins for \( S(1/2, 1) \) is specified as \([0, ..., 3, 10^{10}]\) with an interval of 0.01 between 0 and 3. The bins for other
heavy-tailed distributions are set in a similar manner. Thus, the distribution curves fit the simulated distributions nicely as shown in Figure 5.1.

![Figure 5.1. Histograms of various simulated stable distributions based on Chambers-Mallows-Stuck method with respective distribution overlay curves](image)

5.4. Special functions related to stable distributions

The class of representable densities for the stable distributions can be increased from the three in terms of elementary functions discussed above by considering special functions, in particular generalized hypergeometric series, Meijer $G$ function and Fox $H$ function.

The Fox $H$ function is defined as a complex integral which contains products of gamma functions in its integrand

\[
H_{p,q}^{m,n} \left( \begin{array}{c} (a_1, A_1), \ldots, (a_m, A_m), (a_{m+1}, A_{m+1}), \ldots, (a_p, A_p) \\ (b_1, B_1), \ldots, (b_q, B_q) \\
\end{array} ; z \right) = \\
\frac{1}{2\pi i} \int_L \frac{\prod_{\alpha=1}^{p} \Gamma(b_j + B_j s) \prod_{\beta=1}^{q} \Gamma(1-a_j + A_j s) z^{-s}}{\prod_{\gamma=1}^{m} \Gamma(1-b_j - B_j s) \prod_{\delta=1}^{n} \Gamma(a_j + A_j s)} z^{-s} ds
\]

(5.20)

where all the $A_j$ and $B_j$ are positive, and the $a_j$ and $b_j$ may be complex. The contour $L$ runs from $c - i\infty$ to $c + i\infty$ such that the poles of $\Gamma(b_j + B_j s), j = 1, \ldots, m$ lie to the left of $L$ and the poles of $\Gamma(1-a_j + A_j s), j = 1, \ldots, n$ lie to the right of $L$. $1/\Gamma(0)$ is interpreted as zero. See, e.g. Luke (1969) or Mathai, Saxena and Haubold (2009) for textbook treatments of the $H$ function.

The Meijer $G$ function is the special case of (5.20) when $A_j = 1, j = 1, \ldots, p$ and $B_j = 1, j = 1, \ldots, q$:
\[ G_{p,q}^{m,n}(z \mid a_1, \ldots, a_p, b_1, \ldots, b_q) := H_{p,q}^{m,n}(z \mid (a_1, 1), \ldots, (a_p, 1), (b_1, 1), \ldots, (b_q, 1)) \]  

(5.21)

(see, e.g. Luke, 1969)

The generalized hypergeometric function is defined for positive integers \( p \) and \( q \) as

\[ _pF_q(a_1, \ldots, a_p; b_1, \ldots, b_q; z) := \sum_{k=0}^{\infty} \frac{(a_1)_k \cdots (a_p)_k}{(b_1)_k \cdots (b_q)_k} \frac{z^k}{k!} \]  

(5.22)

with

\[ b_j \in \mathbb{Z} := \{0, -1, -2, \ldots\}, \quad j = 1, \ldots, q \]  

(5.23)

where \((a)_k\) is the Pochhammer symbol defined by

\[ (a)_n := a(a+1) \cdots (a+n-1) = \frac{\Gamma(a+n)}{\Gamma(a)} \]  

(5.24)

The generalized hypergeometric function is actually a special case of Meijer \( G \) function, owing to the relations

\[ _pF_q(a_1, \ldots, a_p; b_1, \ldots, b_q; z) = \frac{\Gamma(b_1) \cdots \Gamma(b_q)}{\Gamma(a_1) \cdots \Gamma(a_p) \Gamma(a_q)} G_{p,q}^{1,1}(1 - a_1, \ldots, 1 - a_p; 0, 1 - b_1, \ldots, 1 - b_q; -z) \]  

(5.25)


### 5.5. Densities of various stable distributions

Schneider (1987) uses the Fox \( H \) function to conduct a thorough investigation into stable distributions including one-sided stable distributions, generalized one-sided stable distributions and two-sided stable distributions. Schneider establishes connections between the densities of different stable distributions, and provides many ways to compute the densities of stable distributions. However, numerical inversion of Laplace transforms has not been used to obtain the densities. Also, numerical examples and a comparison of various methods are missing from the literature. We will summarize all the findings on stable distributions obtained by Schneider (1986; 1987) using the Fox \( H \) function - the most general special function discussed above - and list some important special cases in an attempt to collect the expressions for stable densities that have appeared in various literature. One of the challenges of the literature is that, as we shall see, the mathematical form of the densities of stable random variables is very sensitive to the underlying parameters and this makes the problem of finding a computational
method that is uniformly successful across all of the stable densities very difficult. We shall, nevertheless, address this problem through focusing on Schneider’s Fox H function representations. Through their generality, they encompass the simpler functions like generalized hypergeometric series that can arise in special cases. The strength of our approach is the width of its applicability which can be useful in contexts where some of the parameters are not known; the weakness of our approach is that if all the parameters are known, it could be that we could design a method tailored to this case that will outperform our general method.

5.5.1 Probability density \( g_{m,a} \)

To define the density \( f_{m,a} \) of the generalized one-sided stable distribution \( F_{m,a} \), Schneider (1987) introduces the density \( g_{m,a} \) and derives the Fox function representation and the series expansion for the density respectively as follows.

The density \( g_{m,a} \) on \( \mathbb{R}_+ \) is defined by the integral equation

\[
g_{m,a}(x) = x^{m+a-2} \int_x^\infty y^a (y-x)^{-a} g(y) \, dy
\]

where \( 0 < \alpha < 1 \) and \( m \) is positive integer.

The Mellin transform \( \hat{g}_{m,a} \) of \( g_{m,a} \) exist for \( 0 < \text{Re}(s) < \infty \) and can be obtained by solving a difference equation. The expression for \( \hat{g}_{m,a} \) is

\[
\hat{g}_{m,a}(s) = \int_0^\infty x^{s-1} g_{m,a} \, dx = A b^s \frac{1}{\Gamma(s-1)} \prod_{k=1}^m \Gamma \left( \frac{s+k-2}{\alpha} \right)
\]

with

\[
a = m + \alpha - 1 \\
b = \left( \frac{a^n}{\Gamma(1-\alpha)} \right)^{1/a} \\
A = \frac{1}{ab \prod_{k=1}^m \Gamma \left( \frac{s+k-2}{\alpha} \right)}
\]

Note the arbitrary prefactor \( A \) is computed by solving \( \hat{g}_{m,a}(1) = 1 \). Computation suggests \( a \) does not exist in the denominator of the expression for \( A \). However, further experiment shows the result can not be correct without \( a \) added.
By inverting its Mellin transform, the density $g_{m,\alpha}$ can be expressed in terms of the Fox function $H_{p,q}^{m,a}$

$$g_{m,\alpha}(x) = A \sum_{a=1}^{m} \sum_{n=1}^{\infty} \frac{c_{k,a}}{\Gamma(1-k-n)} \left( \frac{z^a}{n!} \right)^{-1} \left( \frac{x}{b} \right)^{k+2+n,a}$$

The Fox function is an analytical function under certain conditions. We refer readers to Schneider (1987) for details about the evaluation of the Fox function. The density $g_{m,\alpha}$ in the above form has a series expansion which converges absolutely.

$$g_{m,\alpha}(x) = A \sum_{a=1}^{m} \sum_{n=1}^{\infty} \frac{c_{k,a}}{\Gamma(1-k-n,a)} \left( \frac{1}{n!} \right)^{k+2+n,a}$$

with

$$c_{k,a} = \prod_{j=1}^{m} \Gamma \left( \frac{j-k+a}{a} - n \right)$$

where the prime denotes the omission of $j = k$.

### 5.5.2 Generalized one-sided stable distribution $F_{m,\alpha}$

Let $f_{m,\alpha}$ on $R_+$ be the density of generalized one-sided stable distribution $F_{m,\alpha}$ with $0 < \alpha < 1$ and $m$ a positive integer. Schneider (1987) obtains the Laplace transform of $f_{m,\alpha}$ in the forms of the Fox function representation and the series expansion respectively. When $m = 1$, the Laplace transform of $f_{1,\alpha}$ has a closed-form expression. We recap Schneider’s work as follows.

The density $f_{m,\alpha}$ is defined from $g_{m,\alpha}$ by

$$f_{m,\alpha}(x) = x^{-2} g_{m,\alpha}(x^{-1})$$

To obtain the Laplace transform $\phi_{m,\alpha}$ of $f_{m,\alpha}$, the Mellin transform $\hat{\phi}_{m,\alpha}$ of the Laplace transform $\phi_{m,\alpha}$ is introduced. Since $\hat{\phi}_{m,\alpha}$ is related to the Mellin transform $\hat{f}_{m,\alpha}$ of $f_{m,\alpha}$ by

$$\hat{\phi}_{m,\alpha}(s) = \Gamma(s) \hat{f}_{m,\alpha}(1-s)$$

we need to know the expression for $\hat{f}_{m,\alpha}$ which can be computed using the relation $\hat{f}_{m,\alpha}(s) = \hat{g}_{m,\alpha}(2-s)$. Thus,

$$\hat{f}_{m,\alpha}(s) = A \prod_{k=1}^{m} \Gamma \left( \frac{k-s}{a} \right)$$

and this yields
\[
\hat{\phi}_{m,\alpha}(s) = A \ b^{i \pi s} \prod_{k=1}^{m} \Gamma\left(\frac{s-k+1}{\alpha}\right)
\]  
(5.35)

Hence, the Laplace transform \(\phi_{m,\alpha}\) is given by inverting its Mellin transform \(\hat{\phi}_{m,\alpha}\), and the solution involves the Fox function

\[
\phi_{m,\alpha}(\lambda) = 2 \ A \ b \ H_0^m \left\{ \frac{\lambda}{b} \left| \begin{array}{c}
\left( \frac{k-1}{\alpha}, \frac{1}{\alpha} \right)_{k=1, \ldots, m}
\end{array} \right. \right\}^{-1}
\]  
(5.36)

Numerical experiments suggest Schneider (1987) misses out the factor 2.

The series expansion for \(\phi_{m,\alpha}\) is

\[
\phi_{m,\alpha}(\lambda) = A \ a \ b \ \sum_{k=1}^{m} \sum_{n=0}^{\infty} c_{k,n} \ (-1)^n \ \lambda^{k-1+n} \ a
\]  
(5.37)

Note that \(\phi_{m,\alpha}(0) = A \ a \ b \ c_{1,0} = 1\). The arbitrary prefactor \(A\), thus, coincides with the one in \(\hat{\phi}_{m,\alpha}\).

When \(m = 1\), \(\phi_{1,\alpha}\) reduces to

\[
\phi_{1,\alpha}(\lambda) = e^{\left(\frac{\lambda}{b}\right)^\alpha}
\]  
(5.38)

with

\[
b = \left( \frac{a}{\Gamma(1-\alpha)} \right)^{1/\alpha}
\]  
(5.39)

When \(m = 2\), \(\phi_{2,\alpha}\) can be expressed in terms of the modified Bessel function of the second kind \(K_\alpha(z)\)

\[
\phi_{2,\alpha}(\lambda) = \frac{2}{\Gamma(\beta)} \left( \frac{\lambda}{b} \right)^{1/2} \ K_\beta \left( 2 \ \left( \frac{\lambda}{b} \right)^{1/2} \right)
\]  
(5.40)

with

\[
\beta = \frac{1}{1+\alpha}
\]  
(5.41)

**5.5.3 One-sided stable distribution \(F_\alpha\)**

The one-sided stable distribution \(F_\alpha\) on \(R^+\) is related to the generalized one-sided stable distribution \(F_{m,\alpha}\) when \(m = 1\). Let \(f_\alpha(x)\), \(0 < \alpha < 1\), be the density of \(F_\alpha\). The one-sided stable distribution is defined by the Laplace transform

\[
\int_{0}^{\infty} e^{-\lambda x} f_\alpha(x) \ dx = e^{-\lambda^\alpha}
\]  
(5.42)

Schneider (1987) shows the density \(f_\alpha(x)\) is connected with \(f_{m,\alpha}(x)\) by the following relation

\[
f_\alpha(x) = b^{-1} f_{1,\alpha}(b^{-1} x)
\]  
(5.43)
where
\[
b = \left( \frac{a}{\Gamma(1-a)} \right)^{1/\alpha}
\] (5.44)

and also gives the Fox function representation for \( f_a(x) \)
\[
f_a(x) = \alpha^{-1} x^{-2} H_{11}^{10}\left( x^{-1} \left| \begin{array}{c}
(-1, 1) \\
(-\alpha^{-1}, \alpha^{-1})
\end{array} \right. \right)
\] (5.45)

He shows the series expansion for the above Fox function representation is
\[
f_a(x) = \sum_{n=1}^{\infty} \frac{(-1)^n}{\Gamma(-\alpha)} \frac{(-x)^n}{n!} x^{1-n \alpha}
\] (5.46)

By using \( \Gamma(z) \Gamma(1-z) \sin(\pi z) = \pi \), the series expansion has another form, which can also be found in the Demni (2011) paper,
\[
f_a(x) = \frac{1}{\pi x} \sum_{n=1}^{\infty} \frac{\Gamma(n \alpha + 1)}{n!} (-x^{-\alpha})^n \sin(\pi n \alpha)
\] (5.47)

By numerical experiments, we find one-sided stable distribution \( F_a \) is in fact equivalent to \( S_l(\alpha, 1, 0, 1) \), or \( S_l(\alpha, 1) \) for short. Note \( F_a \) should not be confused with \( S_l(\alpha) \) which is the short form of \( S_l(\alpha, 0) \).

In addition to the infinite series formulae, Penson and Gorska (2011) obtain exact and explicit expressions for \( f_a(x) \) for all \( \alpha = l/k < 1 \) with \( k \) and \( l \) positive integers. They show the Laplace transform of one-sided stable distribution (5.42) for \( \alpha = l/k \) can be inverted yielding an expression involving the Meijer G function
\[
G_{m,n}^{p,q}\left( \begin{array}{c}
a_1, \ldots, a_p \\
b_1, \ldots, b_q
\end{array} \right)
\]
\[
f_{l/k}(x) = \frac{\sqrt{k/l}}{(2\pi)^{n-k/2}} \frac{1}{x} G_{1,1}^{0,0}\left( \begin{array}{c}
\Delta(l, 0) \\
\Delta(k, 0)
\end{array} \right)
\] (5.48)

where \( \Delta(m, n) = \frac{n}{m}, \frac{n+1}{m}, \ldots, \frac{n+m-1}{m} \) is a list of \( m \) elements. Mathematica 9.0 has built-in function MeijerG[] to evaluate the Meijer G function.

Penson and Gorska (2011) further write the right-hand side of (5.48) as a finite sum of \( k - 1 \) generalized hypergeometric functions \( _pF_q(a; b; z) \).
\[
f_{l/k}(x) = \sum_{j=0}^{k-1} \frac{b(j,l)}{\Delta(j,l)} \Gamma^{-1} F_k\left( 1, \Delta(l, 1 + l/k); \Delta(k, j + 1); (-1)^{k-l} \frac{x}{\Delta(j,l)} \right)
\] (5.49)

where
The two-sided stable distribution $F_{\alpha,\beta}$ is defined by the Fourier transform (Schneider, 1987; Feller, 1970, pp.581-582)

\[ \int_{-\infty}^{\infty} e^{i k x} f_{\alpha,\beta}(x) \, dx = e^{i\phi_{\alpha,\beta}(k)} \]  

(5.56)

where $f_{\alpha,\beta}(x)$ is the density of $F_{\alpha,\beta}$ and

\[ \phi_{\alpha,\beta}(k) = \begin{cases} 
-|k|^{\alpha} e^{i\frac{k^2}{2} \beta}, & k > 0 \\
-|k|^{\alpha} e^{-i\frac{k^2}{2} \beta}, & k < 0 
\end{cases} \]  

(5.57)

with the parameter $\beta$ satisfying

\[ |\beta| \leq \left\{ \begin{array}{ll}
\alpha, & \text{if } 0 < \alpha < 1 \\
2 - \alpha, & \text{if } 1 < \alpha \leq 2 
\end{array} \right. \]  

(5.58)

The case of $\alpha = 1$ shall not be considered for the two-sided stable distribution $F_{\alpha,\beta}$. We now consider the stable distribution $S_{\alpha}(\alpha, \beta_1, \mu, \sigma_1)$ for $\alpha \neq 1$. When $\mu = 0$ and $\sigma_1 = 1$, the c.f. of $S_{\alpha}(\alpha, \beta_1, 0, 1)$ reduces to
\[
\log \phi(t) = -|t|^\alpha \exp \left\{ -i \beta_1 \text{sign}(t) \frac{\pi}{2} K(\alpha) \right\} \\
= \begin{cases} 
-|t|^\alpha \exp \left\{ i \frac{\pi}{2} [-\text{sign}(t) \beta_1 \alpha] \right\}, & 0 < \alpha < 1 \\
-|t|^\alpha \exp \left\{ i \frac{\pi}{2} [-\text{sign}(t) \beta_1 (2 - \alpha)] \right\}, & 1 < \alpha < 2 \\
-|t|^2, & \alpha = 2
\end{cases}
(5.59)
\]

In the case of \( \alpha = 2 \), the value of \( \beta_1 \) becomes irrelevant. By comparison, \( \psi_{\alpha,\beta}(k) \) can be written neatly as
\[
\psi_{\alpha,\beta}(k) = -|k|^\alpha \exp \left\{ i \frac{\pi}{2} \text{sign}(k) \beta \right\}
(5.60)
\]

Since both \( t \) and \( k \) are transform variables, they can be regarded as the same thing. Thus, \( F_{\alpha,\beta} \) is related to \( S_1(\alpha, \beta_1, 0, 1) \) when
\[
\beta = \begin{cases} 
-\beta_1 \alpha, & 0 < \alpha < 1 \\
-\beta_1 (2 - \alpha), & 1 < \alpha < 2 \\
0, & \alpha = 2
\end{cases}
(5.61)
\]

Therefore, we have
\[
F_{\alpha,\beta} = \begin{cases} 
S_1(\alpha, -\frac{\beta}{\alpha}, 0, 1), & 0 < \alpha < 1 \\
S_1(\alpha, -\frac{\beta}{2-\alpha}, 0, 1), & 1 < \alpha < 2
\end{cases}
(5.62)
\]

and when \( \alpha = 2 \)
\[
F_{2,0} = S_1(2, \beta_1, 0, 1) \quad \text{for all} \beta_1
(5.63)
\]

Assuming \( \beta_1 = 0 \), we have \( F_{2,0} = S(2, 0) \) since \( S_1(\alpha, 0) \) is equivalent to \( S(\alpha, 0) \). Note the conditions
\[
(5.58)
\]
ensures \( \beta_1 \) of \( S_1(\alpha, \beta_1, 0, 1) \) falls in \([-1, 1]\).

Using the correspondence between \( S_1(\alpha, \beta_1, \mu, \sigma_1) \) and \( S(\alpha, \beta, \mu, \sigma) \), the density of \( F_{\alpha,\beta} \) can then be computed from the density of \( S(\alpha, \beta, \mu, \sigma) \).

Given the Fourier transform (5.56), Schneider (1987) shows the density \( f_{\alpha,\beta} \) can be obtained by the inverse Fourier transform
\[
f_{\alpha,\beta}(x) = \frac{1}{\pi} \text{Re} \int_0^\infty e^{-i \xi x} \phi_{\alpha,\beta}(\xi) \, d\xi
(5.64)
\]

Because
\[
f_{\alpha,\beta}(-x) = f_{\alpha,-\beta}(x),
(5.65)
\]
it suffices to consider \( f_{\alpha,\beta}(x) \) for \( x \geq 0 \).
Let \( \hat{f}_{\alpha,\beta} \) be the Mellin transform of \( f_{\alpha,\beta} \). By inverting the Mellin transform

\[
\hat{f}_{\alpha,\beta}(s) = \frac{\Gamma(s-1) \Gamma(1+\alpha-\epsilon s)}{\Gamma(1+\gamma-\gamma s) \Gamma(-\gamma+\gamma s)}
\]

(5.66)

where

\[
\epsilon = \alpha^{-1} \\
\gamma = \alpha^{-1} \frac{a-\beta}{2a}
\]

(5.67)

Schneider (1987) expresses \( f_{\alpha,\beta} \) in terms of the Fox function as follows:

For \( 0 < \alpha < 1 \) and \( x > 0 \)

\[
f_{\alpha,\beta}(x) = x^{-2} H_{11}^1 \left( x^{-1} \mid (0, 1), (1 - \gamma, \gamma) \right) \]

\[
(1 - \epsilon, \epsilon), (1 - \gamma, \gamma)
\]

(5.68)

For \( 1 < \alpha \leq 2 \) and \( x > 0 \)

\[
f_{\alpha,\beta}(x) = H_{12}^1 \left( x \mid (-\epsilon, \epsilon), (-\gamma, \gamma) \right)
\]

\[
(-1, 1), (-\gamma, \gamma)
\]

(5.69)

In particular, when \( 0 < \alpha < 1 \) and \( \beta = -\alpha \),

\[
f_{a,-a}(x) = \epsilon x^{-2} H_{11}^{10} \left( x^{-1} \mid (-1, 1) \right)
\]

(5.70)

which implies

\[
f_{a,-a}(x) = f_a(x)
\]

(5.71)

Feller (1970, p.583) derives a series expansion for the inverse Fourier transform (5.56):

For \( 0 < \alpha < 1 \) and \( x > 0 \)

\[
f_{\alpha,\beta}(x) = \frac{1}{\pi x} \sum_{k=1}^{\infty} \frac{\Gamma(k(\alpha+1))}{k!} (-x^{-\alpha})^k \sin \left[ \frac{k\pi}{2} (\beta - \alpha) \right]
\]

(5.72)

which provides an asymptotic estimate for \( x \to \infty \).

For \( 1 < \alpha \leq 2 \) and \( x > 0 \)

\[
f_{\alpha,\beta}(x) = \frac{1}{\pi x} \sum_{k=1}^{\infty} \frac{\Gamma(1+k(\alpha))}{k!} (-x)^k \sin \left[ \frac{k\pi}{2} (\beta - \alpha) \right]
\]

(5.73)

When \( x < 0 \), the value of \( f_{\alpha,\beta}(x) \) is given by (5.65).

Schneider (1986) shows the following five special cases of \( f_{\alpha,\beta}(x) \) which can be expressed in terms of
special functions.

(1) When \( \alpha = 1/2 \) and \( \beta = -1/2 \), stable density \( f_{a,b}(x) \) is one-sided

\[
f_{1/2,-1/2}(x) = \frac{1}{2\sqrt{\pi}} x^{-1/2} e^{-x/4}, \quad x > 0
\]

(5.74)

Notice that the expression for \( f_{1/2,-1/2}(x) \) is the same as the expression for \( f_{1/2}(x) \) in (5.51)

(2) When \( \alpha = 2/3 \) and \( \beta = -2/3 \), stable density \( f_{a,b}(x) \) is one-sided

\[
f_{2/3,-2/3}(x) = \frac{3}{\pi} x^{-1} e^{-z^2/4} W_{1/2,1/6}(z), \quad x > 0
\]

(5.75)

with \( z = 4/(27x^2) \) and \( W_{k,m}(z) \) the Whittaker function given by

\[
W_{k,m}(z) = e^{-z^2} z^{m+1/2} U(m-k+1/2; 1+2m; z)
\]

(5.76)

where \( U(a, b, z) \) is the confluent hypergeometric function defined as

\[
U(a, b, z) = \frac{1}{\Gamma(a)} \int_0^\infty e^{-zt} t^{a-1}(1+t)^{b-a-1} \, dt
\]

(5.77)

By numerical experiments, we note \( f_{2/3,-2/3}(x) \) is equal to \( f_{2/3}(x) \) which can be expressed in terms of the Kummer confluent hypergeometric function \( _1F_1(\alpha; b; z) \) as in (5.54).

(3) When \( \alpha = 2/3 \) and \( \beta = 0 \), stable density \( f_{a,b}(x) \) is two-sided

\[
f_{2/3,0}(x) = \frac{1}{2\sqrt{3\pi}} |x|^{-1/2} e^{-z^2/4} W_{-1/2,1/6}(z)
\]

(5.78)

with \( z = 4/(27x^2) \) and \( W_{k,m}(z) \) the Whittaker function.

(4) When \( \alpha = 3/2 \) and \( \beta = 1/2 \), stable density \( f_{a,b}(x) \) is two-sided

\[
f_{3/2,1/2}(x) = \frac{1}{2\sqrt{3\pi}} x^{-1/2} e^{-z^2/4} W_{-1/2,1/6}(z), \quad x > 0
\]

(5.79)

\[
f_{3/2,1/2}(x) = -\frac{3}{\pi} x^{-1} e^{-z^2} W_{1/2,1/6}(z), \quad x < 0
\]

(5.80)

with \( z = 4/(27x^2) \) and \( W_{k,m}(z) \) the Whittaker function.

(5) When \( \alpha = 1/3 \) and \( \beta = -1/3 \), stable density \( f_{a,b}(x) \) is one-sided

\[
f_{2/3,-2/3}(x) = \frac{1}{4\sqrt{3\pi}} x^{1/2} K_{1/3}\left(\frac{2}{\sqrt{27x}}\right), \quad x > 0
\]

(5.81)
where $K_\alpha(z)$ is the modified Bessel function of the second kind which satisfies the differential equation

$$z^2 \frac{d^2 y}{dz^2} + z \frac{dy}{dz} - (z^2 + n^2) y = 0 \quad (5.82)$$

Numerical experiments show $f_{1/3, -1/3}(x)$ is equal to $f_{1/3}(x)$ which can be expressed in terms of the modified Bessel function of the first kind $I_\alpha(z)$ as in (5.53).

The stable density $f_{\alpha, \beta}(x)$ is symmetric when $\beta = 0$. Garoni and Frankel (2002) discuss the stable density $f_{\alpha, \beta}(x)$, called Lévy stable density in their paper, with $\alpha$ a rational number, and give special function representations of $f_{\alpha, \beta}(x)$ for $\alpha = 3/2, 2/3, 1/2$. When $\alpha = 3/2$, the density $f_{3/2, \beta}(x)$ is called the Holtsmark density expressed in terms of the Bessel function of the first kind $I_\alpha(z)$ and the hypergeometric function $\text{}_2F_2(a_1, a_2; b_1, b_2; z)$.

$$f_{3/2, \beta}(x) = \frac{4x^3}{27\sqrt{3}} \cos\left(\frac{2\sqrt{3}x}{27}\right) I_{-2/3}\left(\frac{2\sqrt{3}x}{27}\right) - \frac{4x^3}{27\sqrt{3}} \cos\left(\frac{2\sqrt{3}x}{27}\right) I_{2/3}\left(\frac{2\sqrt{3}x}{27}\right) - \frac{4x^3}{27\sqrt{3}} \sin\left(\frac{2\sqrt{3}x}{27}\right) I_{-1/3}\left(\frac{2\sqrt{3}x}{27}\right) + \frac{x^2}{6\pi} \left(\text{}_2F_2\left(1, \frac{3}{2}; \frac{5}{3}, \frac{5}{3} + \frac{i4\sqrt{3}x}{27}\right) + \text{}_2F_2\left(1, \frac{3}{2}; \frac{5}{3}, \frac{5}{3} - \frac{i4\sqrt{3}x}{27}\right)\right) \quad (5.83)$$

Garoni and Frankel (2002) correct the representation of $f_{3/2, \beta}(x)$ and recover the representation of $f_{1/2, \beta}(x)$ which were previously given by Zolotarev (1954). Recalling (5.78), Schneider (1986) also shows the correction for $f_{2/3, \beta}(x)$. Garoni and Frankel (2002) states

$$f_{2/3, \beta}(x) = \frac{3}{4\sqrt{\pi}} \text{}_2F_0\left(\frac{5}{6}, \frac{7}{6}; -\frac{27x^2}{4}\right) \quad (5.84)$$

$$f_{2/3, \beta}(x) = \frac{x^{-5/3}}{2^{1/3}3^{1/2}\sqrt{\pi}} U\left(\frac{5}{6}, \frac{2}{3}, \frac{4}{27x^2}\right) \quad (5.85)$$

$$f_{1/2, \beta}(x) = \frac{1}{\pi} \left(\text{}_2F_0\left(1, \frac{3}{2}; i4x\right) + \text{}_2F_0\left(1, \frac{3}{2}; -i4x\right)\right) \quad (5.86)$$

However, numerical experiments suggest the representations (5.83) and (5.84) yield incorrect results which shall be further checked.
6. Numerical Experiments with Stable Distributions

The densities of various stable distributions discussed in Schneider (1987) are computed by various methods including analytical formulae, numerical inversion of the Laplace transform, inverse Fourier transform, and conversion from known densities. The inversion algorithms for the Laplace transforms include the unified Gaver-Stehfest algorithm (UniG), the unified Euler algorithm (UniE), and the unified Talbot algorithm (UniT). The results of different methods are compared and contrasted to find out the superior approach for each type of stable distribution.

- 6.1. Computing the density $g_{m,\alpha}$

The density $g_{m,\alpha}$ can be computed by the G1 method and the G2 method which are tabulated in Table 6.1. The G1 method exploits the Fox function, while the G2 method which involves the gamma function is a series expansion for the Fox representation G1. A Mathematica code DensityG[] is written by us to implement the formulae. “FoxFunction” and “SeriesGamma” are two possible settings for the Method option for DensityG[] with M the specification of the truncation size.

<table>
<thead>
<tr>
<th>Method</th>
<th>Setting</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1</td>
<td>FoxFunction</td>
<td>$A H_n^0\left[ \frac{x}{\alpha} \right] = \sum_{k=0}^{\infty} \left[ \frac{k-2}{a} \right]^{\frac{1}{a}} k! \cdot \frac{1}{n!} \left( \frac{x}{\alpha} \right)^{k-1}$</td>
</tr>
<tr>
<td>G2</td>
<td>SeriesGamma</td>
<td>$A a \sum_{k=1}^{\infty} \frac{\Delta_{m,\alpha}(-1)}{n!} \left( \frac{x}{\alpha} \right)^{k-1} \cdot \frac{1}{n!} \left( \frac{x}{\alpha} \right)^{k-1}$</td>
</tr>
</tbody>
</table>

$g_{m,\alpha}$ is a density on $R_+$ with $m$ an integer and $0 < \alpha < 1$.

Table 6.2 computes the density $g_{m,\alpha}$ for $m = 1, 2, 3$ and $\alpha = 1/3, 2/3$. It shows the reference densities for $g_{m,\alpha}$ and the accuracy of the G1 method and the G2 method both with $M = 35$ and wp = 35. The reference densities are obtained by comparing the G1 method (with $M = 70$ and wp = 100) with the G2 method (with $M = 100$ and wp = 100) in order to find the matching digits of the results. The G1 method and the G2 method have exactly the same accuracy. The accuracy drops quickly as $x$ increases and even faster for large value of $\alpha$. To maintain the accuracy, we suggest using $M = 100$ for the truncation size. The machine precision is sufficient for the computations. Thus, the CPU time is merely 0.02–0.12 seconds.
### Table 6.2. Accuracy of the G1 method and the G2 method for $g_{m,a}$

<table>
<thead>
<tr>
<th>m</th>
<th>$\alpha$</th>
<th>$x$</th>
<th>Reference</th>
<th>G1</th>
<th>G2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Accu</td>
<td></td>
</tr>
<tr>
<td>1/2</td>
<td>0.1</td>
<td>59</td>
<td>1.2104977943</td>
<td>34.3</td>
<td>24</td>
</tr>
<tr>
<td>1/2</td>
<td>0.2</td>
<td>51</td>
<td>0.4821406842</td>
<td>33.9</td>
<td>19</td>
</tr>
<tr>
<td>1/2</td>
<td>0.4</td>
<td>44</td>
<td>0.1614473427</td>
<td>33.4</td>
<td>16</td>
</tr>
<tr>
<td>1/2</td>
<td>0.6</td>
<td>39</td>
<td>0.0764615122</td>
<td>32.9</td>
<td>13</td>
</tr>
<tr>
<td>1/2</td>
<td>0.1</td>
<td>48</td>
<td>0.3038077068</td>
<td>35.2</td>
<td>21</td>
</tr>
<tr>
<td>1/2</td>
<td>0.2</td>
<td>32</td>
<td>2.1514515350</td>
<td>34.6</td>
<td>13</td>
</tr>
<tr>
<td>1/2</td>
<td>0.4</td>
<td>17</td>
<td>0.6602797777</td>
<td>33.1</td>
<td>5</td>
</tr>
<tr>
<td>1/2</td>
<td>0.6</td>
<td>8</td>
<td>0.958447467</td>
<td>31.0</td>
<td>0</td>
</tr>
<tr>
<td>1/2</td>
<td>0.1</td>
<td>100</td>
<td>0.4901268882</td>
<td>35.3</td>
<td>35</td>
</tr>
<tr>
<td>1/2</td>
<td>0.5</td>
<td>84</td>
<td>0.7879548266</td>
<td>34.5</td>
<td>34</td>
</tr>
<tr>
<td>1/2</td>
<td>1.2</td>
<td>55</td>
<td>0.4661920321</td>
<td>33.4</td>
<td>22</td>
</tr>
<tr>
<td>1/2</td>
<td>1.8</td>
<td>39</td>
<td>0.1374671297</td>
<td>31.8</td>
<td>13</td>
</tr>
<tr>
<td>2/3</td>
<td>0.1</td>
<td>100</td>
<td>0.2615530193</td>
<td>35.1</td>
<td>35</td>
</tr>
<tr>
<td>2/3</td>
<td>0.5</td>
<td>34</td>
<td>1.854144178</td>
<td>34.2</td>
<td>15</td>
</tr>
<tr>
<td>2/3</td>
<td>0.8</td>
<td>20</td>
<td>1.2085459555</td>
<td>32.9</td>
<td>7</td>
</tr>
<tr>
<td>2/3</td>
<td>0.9</td>
<td>13</td>
<td>0.5529020186</td>
<td>31.8</td>
<td>3</td>
</tr>
<tr>
<td>3/4</td>
<td>0.4</td>
<td>99</td>
<td>0.3505897211</td>
<td>34.7</td>
<td>34</td>
</tr>
<tr>
<td>3/4</td>
<td>0.8</td>
<td>90</td>
<td>0.8709123277</td>
<td>34.2</td>
<td>34</td>
</tr>
<tr>
<td>3/4</td>
<td>1.2</td>
<td>61</td>
<td>0.9000879451</td>
<td>33.4</td>
<td>24</td>
</tr>
<tr>
<td>3/4</td>
<td>1.6</td>
<td>40</td>
<td>0.3354379642</td>
<td>31.8</td>
<td>14</td>
</tr>
<tr>
<td>3/4</td>
<td>2.0</td>
<td>69</td>
<td>0.7570151825</td>
<td>34.5</td>
<td>32</td>
</tr>
<tr>
<td>3/4</td>
<td>0.8</td>
<td>32</td>
<td>2.4228004286</td>
<td>33.8</td>
<td>13</td>
</tr>
<tr>
<td>3/4</td>
<td>0.9</td>
<td>22</td>
<td>2.008545210</td>
<td>32.9</td>
<td>8</td>
</tr>
<tr>
<td>3/4</td>
<td>1.2</td>
<td>12</td>
<td>0.7685104401</td>
<td>31.5</td>
<td>2</td>
</tr>
</tbody>
</table>

wp: computing precision. Accu: accuracy measured by the number of significant digits. ED: effective number of digits of precision in the result. CPU: computation time in seconds.

The reference density is obtained by digits matching with the following option specifications for our code DensityG[]:

(Method -> ("SeriesGamma", "M" -> 70), WorkingPrecision -> 100) vs (Method -> ("FoxFunction", "M" -> 100), WorkingPrecision -> 100).

### 6.2. Computing the density $f_{m,a}$ of generalized one-sided stable distribution $F_{m,a}$

The methods for computing the density $f_{m,a}$ are summarized in Table 6.3. The GS1 method computes $f_{m,a}$ from the density $g_{m,a}$ using the relation (5.32). Two possible settings of “FoxFunction” and “SeriesGamma” can be chosen to decide the formula used for $g_{m,a}$. The GS2 method calculates the density by numerically inverting the Laplace transform which takes the corresponding form when the subsetting is specified. When $m = 1, 2$, the Laplace transform has one term. While in other cases, the Laplace transform has infinite terms, and the truncation is required. UniE, UniT and UniG are three different inversion algorithms which can be called by the GS2 method. The Mathematica code written for the density $f_{m,a}$ is DensityGeneralizedOneSidedStable[].
Table 6.3. Methods for computing the density \( f_{m,a} \)

<table>
<thead>
<tr>
<th>Method</th>
<th>Setting</th>
<th>Subsetting</th>
<th>Formula / Expression</th>
<th>Tr.</th>
<th>Eq.</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS1</td>
<td>FoxFunction</td>
<td>-</td>
<td>( x^{-2} g_{m,a}(x^{-1}) )</td>
<td>M</td>
<td>(5.32)</td>
</tr>
<tr>
<td></td>
<td>SeriesGamma</td>
<td></td>
<td>( \frac{2 A b (\frac{\lambda}{b})^{\frac{1}{2}}} {\Gamma(\frac{1}{2})} K_{\frac{1}{2}} \left( \frac{\lambda}{b} \right)^{\frac{1}{2}} ) for ( m = 1 ), ( e^{-\frac{1}{2}x^2} ) for ( m = 2 ), ( e^{-\frac{1}{2}x^2} ) for ( m &gt; 2 ), the same as (4.26)</td>
<td>M1</td>
<td>(5.36)</td>
</tr>
<tr>
<td></td>
<td>UniE</td>
<td>SeriesGamma</td>
<td>( A b \sum_{k=0}^{\infty} c_k, a \left( \frac{\lambda}{b} \right)^{\frac{1}{2}} K_{\frac{1}{2}} \left( \frac{\lambda}{b} \right)^{\frac{1}{2}} ) for ( m = 1 ), ( \frac{2 A b (\frac{\lambda}{b})^{\frac{1}{2}}} {\Gamma(\frac{1}{2})} K_{\frac{1}{2}} \left( \frac{\lambda}{b} \right)^{\frac{1}{2}} ) for ( m = 2 ), the same as (4.26)</td>
<td>M2</td>
<td>(5.37)</td>
</tr>
<tr>
<td></td>
<td>UniT</td>
<td></td>
<td>( \frac{2 A b (\frac{\lambda}{b})^{\frac{1}{2}}} {\Gamma(\frac{1}{2})} K_{\frac{1}{2}} \left( \frac{\lambda}{b} \right)^{\frac{1}{2}} ) for ( m = 1 ), ( e^{-\frac{1}{2}x^2} ) for ( m = 2 ), ( e^{-\frac{1}{2}x^2} ) for ( m &gt; 2 ), the same as (4.26)</td>
<td></td>
<td>(5.38)</td>
</tr>
<tr>
<td></td>
<td>UniG</td>
<td></td>
<td>( \frac{2 A b (\frac{\lambda}{b})^{\frac{1}{2}}} {\Gamma(\frac{1}{2})} K_{\frac{1}{2}} \left( \frac{\lambda}{b} \right)^{\frac{1}{2}} ) for ( m = 1 ), ( e^{-\frac{1}{2}x^2} ) for ( m = 2 ), ( e^{-\frac{1}{2}x^2} ) for ( m &gt; 2 ), the same as (4.26)</td>
<td></td>
<td>(5.40)</td>
</tr>
</tbody>
</table>

M1 is the truncation size of the inversion algorithm. M2 is the truncation size of the Laplace transform. \( f_{m,a} \) is the density of generalized one-sided stable distribution on \( R_+ \) with \( m \) an integer and \( 0 < a < 1 \).

We first use the GS1 method with different rules to compute the density \( f_{m,a} \) for \( m = 1, 2 \) and \( \alpha = 1/3, 1/2, 2/3 \). The results are tabulated in Table 6.4. The GS1 method provides an asymptotic estimate of \( f_{m,a} \) for \( x \to \infty \). When \( x \) becomes very small, the accuracy would drop quickly and the error would be larger than the estimate. To obtain the correct result, it is necessary to increase the size of \( M \) and the computation precision. Alternatively, we may choose not to compute the density when \( x \) is below some small value since \( f_{m,a} \to 0 \) as \( x \to 0 \). We recommend the settings of \( M = 100 \) and \( wp = 30 \) for the GS1 method and increase the size of \( M \) if necessary.
Next, we compute $f_{m,a}$ using the GS2 method with the inversion algorithms of UniE, UniT and UniG respectively and demonstrate the results in Table 6.5. The accuracy of the GS2 method is less sensitive to the value of $x$ than that of the GS1 method. But the truncation error may still become a problem when $x$ is small. Therefore, it is better to rise the truncation size to see whether the result changes significantly.

UniT takes the minimal computation time when compared with UniE and UniG. It is also more accurate than the other two for the given parameter settings. For all three algorithms, the computation time for $m = 2$ is one order of magnitude more than that of $m = 1$ as Laplace transform is much simpler in the latter case.

<table>
<thead>
<tr>
<th>m</th>
<th>$\alpha$</th>
<th>$x$</th>
<th>Reference</th>
<th>GS1: SeriesGamma</th>
<th>GS1: FoxFunction</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Accu</td>
<td>Density</td>
</tr>
<tr>
<td>1</td>
<td>1/3</td>
<td>0.1</td>
<td>129</td>
<td>0.0005089704021</td>
<td>191</td>
</tr>
<tr>
<td>1</td>
<td>1/3</td>
<td>1</td>
<td>215</td>
<td>0.0257147760</td>
<td>197</td>
</tr>
<tr>
<td>1</td>
<td>1/3</td>
<td>2</td>
<td>240</td>
<td>0.0270517045</td>
<td>198</td>
</tr>
<tr>
<td>1</td>
<td>1/3</td>
<td>3</td>
<td>274</td>
<td>0.0192862730</td>
<td>199</td>
</tr>
<tr>
<td>1</td>
<td>1/3</td>
<td>15</td>
<td>299</td>
<td>0.008682158461</td>
<td>200</td>
</tr>
<tr>
<td>1</td>
<td>1/2</td>
<td>1</td>
<td>145</td>
<td>0.0432191826</td>
<td>197</td>
</tr>
<tr>
<td>1</td>
<td>1/2</td>
<td>2</td>
<td>183</td>
<td>0.0734952905</td>
<td>199</td>
</tr>
<tr>
<td>1</td>
<td>1/2</td>
<td>5</td>
<td>234</td>
<td>0.0477162547</td>
<td>200</td>
</tr>
<tr>
<td>1</td>
<td>1/2</td>
<td>10</td>
<td>272</td>
<td>0.02309736113</td>
<td>200</td>
</tr>
<tr>
<td>1</td>
<td>1/2</td>
<td>15</td>
<td>293</td>
<td>0.0139601942</td>
<td>200</td>
</tr>
<tr>
<td>2</td>
<td>1/2</td>
<td>2</td>
<td>92</td>
<td>0.05617288973</td>
<td>197</td>
</tr>
<tr>
<td>2</td>
<td>1/2</td>
<td>4</td>
<td>144</td>
<td>0.0070654894</td>
<td>199</td>
</tr>
<tr>
<td>2</td>
<td>1/2</td>
<td>6</td>
<td>172</td>
<td>0.0460079243</td>
<td>200</td>
</tr>
<tr>
<td>2</td>
<td>1/2</td>
<td>8</td>
<td>193</td>
<td>0.0440052389</td>
<td>200</td>
</tr>
<tr>
<td>2</td>
<td>1/2</td>
<td>10</td>
<td>209</td>
<td>0.0303907076</td>
<td>200</td>
</tr>
<tr>
<td>2</td>
<td>1/3</td>
<td>0.5</td>
<td>215</td>
<td>0.3181670635</td>
<td>196</td>
</tr>
<tr>
<td>2</td>
<td>1/3</td>
<td>1</td>
<td>298</td>
<td>0.6042454577</td>
<td>199</td>
</tr>
<tr>
<td>2</td>
<td>1/3</td>
<td>1.5</td>
<td>299</td>
<td>0.3439152052</td>
<td>199</td>
</tr>
<tr>
<td>2</td>
<td>1/3</td>
<td>2</td>
<td>299</td>
<td>0.1977543099</td>
<td>200</td>
</tr>
<tr>
<td>2</td>
<td>1/3</td>
<td>2.5</td>
<td>299</td>
<td>0.1230115656</td>
<td>200</td>
</tr>
<tr>
<td>2</td>
<td>1/2</td>
<td>0.7</td>
<td>147</td>
<td>0.2905364159</td>
<td>196</td>
</tr>
<tr>
<td>2</td>
<td>1/2</td>
<td>1</td>
<td>206</td>
<td>0.749512133</td>
<td>196</td>
</tr>
<tr>
<td>2</td>
<td>1/2</td>
<td>1.5</td>
<td>271</td>
<td>0.4776511117</td>
<td>199</td>
</tr>
<tr>
<td>2</td>
<td>1/2</td>
<td>2</td>
<td>299</td>
<td>0.2496397118</td>
<td>200</td>
</tr>
<tr>
<td>2</td>
<td>1/2</td>
<td>3</td>
<td>299</td>
<td>0.0872601882</td>
<td>200</td>
</tr>
<tr>
<td>2</td>
<td>1/2</td>
<td>4</td>
<td>200</td>
<td>0.1404166540</td>
<td>195</td>
</tr>
<tr>
<td>2</td>
<td>1/3</td>
<td>1.2</td>
<td>107</td>
<td>0.6822829569</td>
<td>198</td>
</tr>
<tr>
<td>2</td>
<td>1/3</td>
<td>1.6</td>
<td>160</td>
<td>0.7283310383</td>
<td>199</td>
</tr>
<tr>
<td>2</td>
<td>1/3</td>
<td>2</td>
<td>201</td>
<td>0.4042798883</td>
<td>200</td>
</tr>
<tr>
<td>2</td>
<td>1/3</td>
<td>3</td>
<td>275</td>
<td>0.1068501478</td>
<td>200</td>
</tr>
</tbody>
</table>
Table 6.5. Accuracy of the GS2 method for $f_{m,a}$ with the rules of UniE, UniT and UniG respectively

<table>
<thead>
<tr>
<th>m</th>
<th>$\alpha$</th>
<th>$\alpha$</th>
<th>Reference</th>
<th>GS2 : UniE</th>
<th>GS2 : UniT</th>
<th>GS2 : UniG</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>M = 30, wp = 30</td>
<td>M = 30, wp = 30</td>
<td>M = 30, wp = 60</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Accu</td>
<td>Density</td>
<td>Accu</td>
</tr>
<tr>
<td>1/3</td>
<td>0.1</td>
<td>129</td>
<td>0.00508970421</td>
<td>24.9</td>
<td>18</td>
<td>0.02</td>
</tr>
<tr>
<td>1/3</td>
<td>1</td>
<td>215</td>
<td>0.02571454760</td>
<td>22.0</td>
<td>19</td>
<td>0.02</td>
</tr>
<tr>
<td>1/3</td>
<td>2</td>
<td>240</td>
<td>0.02705517045</td>
<td>21.3</td>
<td>19</td>
<td>0.02</td>
</tr>
<tr>
<td>1/3</td>
<td>5</td>
<td>274</td>
<td>0.01928562730</td>
<td>20.5</td>
<td>18</td>
<td>0.02</td>
</tr>
<tr>
<td>1/3</td>
<td>15</td>
<td>299</td>
<td>0.08662185461</td>
<td>19.7</td>
<td>18</td>
<td>0.02</td>
</tr>
<tr>
<td>1/2</td>
<td>1</td>
<td>145</td>
<td>0.0432391826</td>
<td>24.3</td>
<td>18</td>
<td>0.02</td>
</tr>
<tr>
<td>1/2</td>
<td>2</td>
<td>183</td>
<td>0.0734952905</td>
<td>22.7</td>
<td>18</td>
<td>0.02</td>
</tr>
<tr>
<td>1/2</td>
<td>5</td>
<td>234</td>
<td>0.04771662547</td>
<td>21.0</td>
<td>18</td>
<td>0.02</td>
</tr>
<tr>
<td>1/2</td>
<td>10</td>
<td>272</td>
<td>0.02309736113</td>
<td>20.0</td>
<td>17</td>
<td>0.02</td>
</tr>
<tr>
<td>1/2</td>
<td>15</td>
<td>293</td>
<td>0.0139601942</td>
<td>19.5</td>
<td>19</td>
<td>0.02</td>
</tr>
<tr>
<td>1/2</td>
<td>2</td>
<td>92</td>
<td>0.06917289973</td>
<td>26.0</td>
<td>15</td>
<td>0.02</td>
</tr>
<tr>
<td>1/2</td>
<td>3</td>
<td>144</td>
<td>0.1070654894</td>
<td>23.3</td>
<td>15</td>
<td>0.02</td>
</tr>
<tr>
<td>1/2</td>
<td>4</td>
<td>172</td>
<td>0.0680872434</td>
<td>22.0</td>
<td>15</td>
<td>0.02</td>
</tr>
<tr>
<td>1/2</td>
<td>8</td>
<td>193</td>
<td>0.0440552389</td>
<td>21.2</td>
<td>15</td>
<td>0.02</td>
</tr>
<tr>
<td>1/2</td>
<td>10</td>
<td>209</td>
<td>0.0303907706</td>
<td>20.6</td>
<td>17</td>
<td>0.02</td>
</tr>
<tr>
<td>1/2</td>
<td>0.5</td>
<td>215</td>
<td>0.3181670635</td>
<td>26.7</td>
<td>16</td>
<td>0.12</td>
</tr>
<tr>
<td>1/2</td>
<td>1</td>
<td>298</td>
<td>0.6042454577</td>
<td>24.1</td>
<td>14</td>
<td>0.23</td>
</tr>
<tr>
<td>1/2</td>
<td>1.5</td>
<td>299</td>
<td>0.3439312502</td>
<td>22.7</td>
<td>15</td>
<td>0.33</td>
</tr>
<tr>
<td>1/2</td>
<td>2</td>
<td>299</td>
<td>0.1975430299</td>
<td>21.9</td>
<td>15</td>
<td>0.42</td>
</tr>
<tr>
<td>1/2</td>
<td>2.5</td>
<td>299</td>
<td>0.1230115656</td>
<td>21.3</td>
<td>17</td>
<td>0.50</td>
</tr>
<tr>
<td>1/2</td>
<td>0.7</td>
<td>147</td>
<td>0.2950634159</td>
<td>26.6</td>
<td>14</td>
<td>0.09</td>
</tr>
<tr>
<td>1/2</td>
<td>1</td>
<td>206</td>
<td>0.4795412133</td>
<td>24.8</td>
<td>11</td>
<td>0.16</td>
</tr>
<tr>
<td>1/2</td>
<td>1.5</td>
<td>271</td>
<td>0.4776516117</td>
<td>23.8</td>
<td>10</td>
<td>0.22</td>
</tr>
<tr>
<td>1/2</td>
<td>2</td>
<td>299</td>
<td>0.2496397118</td>
<td>22.7</td>
<td>12</td>
<td>0.28</td>
</tr>
<tr>
<td>1/2</td>
<td>3</td>
<td>299</td>
<td>0.0872601882</td>
<td>21.1</td>
<td>12</td>
<td>0.39</td>
</tr>
<tr>
<td>2/3</td>
<td>1</td>
<td>75</td>
<td>0.1404096410</td>
<td>27.7</td>
<td>10</td>
<td>0.08</td>
</tr>
<tr>
<td>2/3</td>
<td>1.2</td>
<td>107</td>
<td>0.6822829560</td>
<td>27.2</td>
<td>9</td>
<td>0.09</td>
</tr>
<tr>
<td>2/3</td>
<td>1.6</td>
<td>160</td>
<td>0.283343183</td>
<td>25.2</td>
<td>8</td>
<td>0.14</td>
</tr>
<tr>
<td>2/3</td>
<td>2</td>
<td>201</td>
<td>0.4042709883</td>
<td>24.1</td>
<td>8</td>
<td>0.19</td>
</tr>
<tr>
<td>2/3</td>
<td>3</td>
<td>275</td>
<td>0.1068501478</td>
<td>21.6</td>
<td>9</td>
<td>0.25</td>
</tr>
</tbody>
</table>

wp: computing precision. Accu: accuracy measured by the number of significant digits. ED: effective number of digits of precision in the result. CPU: computation time in seconds.

The reference density is obtained by digits matching with the following option specifications for our code

```
DensityGeneralizedOneSidedStable[] : (Method -> "SeriesGamma", "M" -> 250), WorkingPrecision -> 300) vs
(Method -> "FoxFunction", "M" -> 350), WorkingPrecision -> 300).
```

The curves of $f_{m,a}$ for $m = 1, 2$ and $\alpha = 1/3, 1/2, 2/3$ are drawn in Figure 6.1 and 6.2.
6.3. Computing the density \( f_a \) of one-sided stable distribution \( F_a \)

There are eight different methods for computing the density \( f_a \) which are summarized in Table 6.6. The IS1 method calculates \( f_a \) via the density \( f_{m,a} \). The IS2 method finds the density by numerical inversion of the Laplace transform. The IS3 method takes the form of the Fox function, while the IS4 method and the IS5 method are two different forms of the series expansion for the Fox function. The IS6 method and the IS7 method are two exact and explicit formulae for \( f_a \). Given that \( f_a \) is the density of \( S_1(\alpha, 1, 0, 1) \), the IS8 method utilizes the relation between \( S_1(\alpha, \beta_1, \mu, \sigma_1) \) and \( S(\alpha, \beta, \mu, \sigma) \) to compute \( f_a \) using the Mathematica built-in function StableDistribution[ ] which is designed for \( S(\alpha, \beta, \mu, \sigma) \). The IS8 method also yields an exact result. The Mathematica code written for computing \( f_a \) is named DensityOneSidedStable[ ].
By computing $f_a$ for $\alpha = 1/3, 1/2, 2/3$, the IS1 method, the IS3 method, the IS4 method and the IS5 method exhibit the similarity of the performance. As shown in Table 6.7, the results obtained by different methods have the same number of digits of both precision and accuracy for all cases which implies these methods are essentially identical. We may set $M = 100$ and wp = 30 when using these methods and increase $M$ as $x$ is very close to zero.
Table 6.7. Accuracy of the IS1 method, the IS3 method, the IS4 method and the IS5 method for $f_a$

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>$x$</th>
<th>Reference</th>
<th>IS1</th>
<th>IS2</th>
<th>IS3</th>
<th>IS4</th>
<th>IS5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Accu</td>
<td>Density</td>
<td>ED</td>
<td>Accu</td>
<td>CPU</td>
<td>ED</td>
</tr>
<tr>
<td>1/3</td>
<td>0.005</td>
<td>99</td>
<td>0.6891204216</td>
<td>104.41</td>
<td>0.02</td>
<td>105.41</td>
<td>0.02</td>
</tr>
<tr>
<td>1/3</td>
<td>0.007</td>
<td>99</td>
<td>1.049140075</td>
<td>105.48</td>
<td>0.02</td>
<td>106.48</td>
<td>0.02</td>
</tr>
<tr>
<td>1/3</td>
<td>0.01</td>
<td>99</td>
<td>1.420687926</td>
<td>106.53</td>
<td>0.02</td>
<td>107.53</td>
<td>0.02</td>
</tr>
<tr>
<td>1/3</td>
<td>0.02</td>
<td>99</td>
<td>1.833669199</td>
<td>107.84</td>
<td>0.02</td>
<td>108.84</td>
<td>0.02</td>
</tr>
<tr>
<td>1/3</td>
<td>0.04</td>
<td>99</td>
<td>1.697714701</td>
<td>108.74</td>
<td>0.02</td>
<td>109.74</td>
<td>0.02</td>
</tr>
<tr>
<td>1/3</td>
<td>0.08</td>
<td>99</td>
<td>1.241965661</td>
<td>108.85</td>
<td>0.02</td>
<td>109.85</td>
<td>0.02</td>
</tr>
<tr>
<td>1/3</td>
<td>0.1</td>
<td>99</td>
<td>1.080842851</td>
<td>109.88</td>
<td>0.02</td>
<td>109.88</td>
<td>0.02</td>
</tr>
<tr>
<td>1/3</td>
<td>0.2</td>
<td>99</td>
<td>0.8405189129</td>
<td>109.97</td>
<td>0.02</td>
<td>109.97</td>
<td>0.02</td>
</tr>
<tr>
<td>1/3</td>
<td>0.07</td>
<td>99</td>
<td>0.4282489303</td>
<td>106.35</td>
<td>0.02</td>
<td>106.35</td>
<td>0.02</td>
</tr>
<tr>
<td>1/3</td>
<td>0.08</td>
<td>99</td>
<td>0.5477593904</td>
<td>107.38</td>
<td>0.34</td>
<td>107.38</td>
<td>0.34</td>
</tr>
<tr>
<td>1/3</td>
<td>0.1</td>
<td>99</td>
<td>0.7322491281</td>
<td>107.43</td>
<td>0.27</td>
<td>107.43</td>
<td>0.27</td>
</tr>
<tr>
<td>1/3</td>
<td>0.2</td>
<td>99</td>
<td>0.9036193633</td>
<td>109.59</td>
<td>0.02</td>
<td>109.59</td>
<td>0.02</td>
</tr>
<tr>
<td>1/3</td>
<td>0.3</td>
<td>99</td>
<td>1.7461070053</td>
<td>109.68</td>
<td>0.02</td>
<td>109.68</td>
<td>0.02</td>
</tr>
<tr>
<td>1/3</td>
<td>0.4</td>
<td>99</td>
<td>1.5968880444</td>
<td>109.74</td>
<td>0.02</td>
<td>109.74</td>
<td>0.02</td>
</tr>
<tr>
<td>1/3</td>
<td>0.5</td>
<td>99</td>
<td>0.6891204216</td>
<td>110.79</td>
<td>0.02</td>
<td>110.79</td>
<td>0.02</td>
</tr>
<tr>
<td>1/3</td>
<td>0.6</td>
<td>99</td>
<td>0.4001601755</td>
<td>110.83</td>
<td>0.02</td>
<td>110.83</td>
<td>0.02</td>
</tr>
<tr>
<td>1/3</td>
<td>0.25</td>
<td>98</td>
<td>0.2331824095</td>
<td>106.17</td>
<td>0.02</td>
<td>106.17</td>
<td>0.02</td>
</tr>
<tr>
<td>1/3</td>
<td>0.3</td>
<td>98</td>
<td>0.8168792541</td>
<td>108.29</td>
<td>0.02</td>
<td>108.29</td>
<td>0.02</td>
</tr>
<tr>
<td>1/3</td>
<td>0.4</td>
<td>98</td>
<td>0.3519702146</td>
<td>108.37</td>
<td>0.02</td>
<td>108.37</td>
<td>0.02</td>
</tr>
<tr>
<td>1/3</td>
<td>0.5</td>
<td>98</td>
<td>0.8579353313</td>
<td>109.44</td>
<td>0.02</td>
<td>109.44</td>
<td>0.02</td>
</tr>
<tr>
<td>1/3</td>
<td>0.6</td>
<td>98</td>
<td>0.7207307337</td>
<td>109.50</td>
<td>0.02</td>
<td>109.50</td>
<td>0.02</td>
</tr>
<tr>
<td>1/3</td>
<td>0.7</td>
<td>98</td>
<td>0.5962134742</td>
<td>109.54</td>
<td>0.02</td>
<td>109.54</td>
<td>0.02</td>
</tr>
<tr>
<td>1/3</td>
<td>0.8</td>
<td>98</td>
<td>0.4946557018</td>
<td>110.58</td>
<td>0.02</td>
<td>110.58</td>
<td>0.02</td>
</tr>
</tbody>
</table>

wp: computing precision. Accu: accuracy measured by the number of significant digits. ED: effective number of digits of precision in the result. CPU: computation time in seconds.

The reference density is computed using built-in function StableDistribution[] by setting the options for our code DensityOneSidedStable[] to {Method -> "DensityStableSt1", WorkingPrecision -> 100}.

With appropriate parameter settings, the IS2 method is as fast as the analytical formulae such as the IS4 method. As observed in Table 6.8, computation of $f_a$ using UniE, UniT and UniG takes 0~0.03 seconds.

Among three algorithms, UniT is the most accurate routine.
Table 6.8. Accuracy of the IS2 method for $f_\alpha$ with the rules of UniE, UniT and UniG respectively

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>$\alpha$</th>
<th>Reference</th>
<th>UniE</th>
<th>UniT</th>
<th>UniG</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$x$</td>
<td>M = 30, wp = 30</td>
<td>M = 30, wp = 30</td>
<td>M = 30, wp = 30</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Accu</td>
<td>Density</td>
<td>ED</td>
<td>Accu</td>
</tr>
<tr>
<td>1/3</td>
<td>0.005</td>
<td>99</td>
<td>0.6891204216</td>
<td>23.3</td>
<td>19</td>
</tr>
<tr>
<td>1/3</td>
<td>0.007</td>
<td>99</td>
<td>1.049140070</td>
<td>22.8</td>
<td>19</td>
</tr>
<tr>
<td>1/3</td>
<td>0.01</td>
<td>99</td>
<td>1.420687728</td>
<td>22.4</td>
<td>19</td>
</tr>
<tr>
<td>1/3</td>
<td>0.02</td>
<td>99</td>
<td>1.833461999</td>
<td>21.6</td>
<td>21</td>
</tr>
<tr>
<td>1/3</td>
<td>0.04</td>
<td>99</td>
<td>1.69714701</td>
<td>21.0</td>
<td>19</td>
</tr>
<tr>
<td>1/3</td>
<td>0.08</td>
<td>99</td>
<td>1.241964561</td>
<td>20.4</td>
<td>18</td>
</tr>
<tr>
<td>1/3</td>
<td>0.1</td>
<td>99</td>
<td>1.00894251</td>
<td>20.3</td>
<td>19</td>
</tr>
<tr>
<td>1/3</td>
<td>0.2</td>
<td>99</td>
<td>0.6405189129</td>
<td>19.8</td>
<td>18</td>
</tr>
<tr>
<td>1/3</td>
<td>0.07</td>
<td>99</td>
<td>0.4282489303</td>
<td>24.6</td>
<td>19</td>
</tr>
<tr>
<td>1/3</td>
<td>0.08</td>
<td>99</td>
<td>0.5477693904</td>
<td>24.3</td>
<td>18</td>
</tr>
<tr>
<td>1/3</td>
<td>0.1</td>
<td>99</td>
<td>0.7322491281</td>
<td>23.7</td>
<td>18</td>
</tr>
<tr>
<td>1/3</td>
<td>0.2</td>
<td>99</td>
<td>0.903619635</td>
<td>22.2</td>
<td>17</td>
</tr>
<tr>
<td>1/3</td>
<td>0.3</td>
<td>99</td>
<td>0.7461070053</td>
<td>21.5</td>
<td>17</td>
</tr>
<tr>
<td>1/3</td>
<td>0.4</td>
<td>99</td>
<td>0.5968580144</td>
<td>21.0</td>
<td>17</td>
</tr>
<tr>
<td>1/3</td>
<td>0.5</td>
<td>99</td>
<td>0.4839414490</td>
<td>20.7</td>
<td>17</td>
</tr>
<tr>
<td>1/2</td>
<td>0.6</td>
<td>99</td>
<td>0.4001401755</td>
<td>20.4</td>
<td>17</td>
</tr>
<tr>
<td>2/3</td>
<td>0.2</td>
<td>98</td>
<td>0.2331824095</td>
<td>26.9</td>
<td>17</td>
</tr>
<tr>
<td>2/3</td>
<td>0.25</td>
<td>98</td>
<td>0.5680216464</td>
<td>26.0</td>
<td>15</td>
</tr>
<tr>
<td>2/3</td>
<td>0.3</td>
<td>98</td>
<td>0.8168792541</td>
<td>25.3</td>
<td>15</td>
</tr>
<tr>
<td>2/3</td>
<td>0.4</td>
<td>98</td>
<td>0.9519702146</td>
<td>24.1</td>
<td>16</td>
</tr>
<tr>
<td>2/3</td>
<td>0.5</td>
<td>98</td>
<td>0.8579353313</td>
<td>23.3</td>
<td>14</td>
</tr>
<tr>
<td>2/3</td>
<td>0.6</td>
<td>98</td>
<td>0.7207307337</td>
<td>22.6</td>
<td>15</td>
</tr>
<tr>
<td>2/3</td>
<td>0.7</td>
<td>98</td>
<td>0.5962134742</td>
<td>22.1</td>
<td>15</td>
</tr>
<tr>
<td>2/3</td>
<td>0.8</td>
<td>98</td>
<td>0.4946557018</td>
<td>21.7</td>
<td>15</td>
</tr>
</tbody>
</table>

wp: computing precision. Accu: accuracy measured by the number of significant digits. ED: effective number of digits of precision in the result. CPU: computation time in seconds.

The reference density is computed using built-in function StableDistribution[ ] by setting the options for our code DensityOneSidedStable[ ] to {Method -> "DensityStableSt1", WorkingPrecision -> 100].

The IS6 method, the IS7 method and the IS8 method yield the exact results in Table 6.9. The accuracy of the results is only a few digits below the computation precision. Write any rational number $0 < \alpha < 1$ as a simple fraction $l/k$ with $k$ and $l$ integers. These methods compute the density almost instantly for $\alpha$ with a relatively small denominator $k$ such as $\alpha = 1/3$, 1/2, 2/3. When $\alpha$ has many decimal digits, the difficulty of the computation may exponentially increase. For example, the computation time for $\alpha = 0.11$ requires from 3 seconds to 16 seconds for different methods since the simple fraction of 0.11 is $11/100$ with $k = 100$. 
Table 6.9. Accuracy of the IS6 method, the IS7 method and the IS8 method for $f_a$

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>$x$</th>
<th>Reference</th>
<th>IS6 $\text{wp = 100}$</th>
<th>IS7 $\text{wp = 100}$</th>
<th>IS8 $\text{wp = 100}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Accu</td>
<td>Density</td>
<td>ED</td>
<td>Accu</td>
</tr>
<tr>
<td>1/3</td>
<td>0.005</td>
<td>99</td>
<td>0.6891204216</td>
<td>99.7</td>
<td>99</td>
</tr>
<tr>
<td>1/3</td>
<td>0.05</td>
<td>99</td>
<td>1.049140075</td>
<td>99.7</td>
<td>99</td>
</tr>
<tr>
<td>1/3</td>
<td>0.01</td>
<td>99</td>
<td>1.420687726</td>
<td>99.7</td>
<td>99</td>
</tr>
<tr>
<td>1/3</td>
<td>0.02</td>
<td>99</td>
<td>1.83469199</td>
<td>99.7</td>
<td>99</td>
</tr>
<tr>
<td>1/3</td>
<td>0.04</td>
<td>99</td>
<td>1.69714701</td>
<td>98.0</td>
<td>98</td>
</tr>
<tr>
<td>1/3</td>
<td>0.08</td>
<td>99</td>
<td>1.241964561</td>
<td>98.5</td>
<td>98</td>
</tr>
<tr>
<td>1/3</td>
<td>1</td>
<td>99</td>
<td>1.080842851</td>
<td>98.6</td>
<td>98</td>
</tr>
<tr>
<td>1/2</td>
<td>0.2</td>
<td>99</td>
<td>0.6405189129</td>
<td>98.9</td>
<td>98</td>
</tr>
<tr>
<td>1/2</td>
<td>0.07</td>
<td>99</td>
<td>0.4282489303</td>
<td>99.3</td>
<td>99</td>
</tr>
<tr>
<td>1/2</td>
<td>0.08</td>
<td>99</td>
<td>0.5477591904</td>
<td>99.3</td>
<td>99</td>
</tr>
<tr>
<td>1/2</td>
<td>0.1</td>
<td>99</td>
<td>0.7322491281</td>
<td>99.3</td>
<td>99</td>
</tr>
<tr>
<td>1/2</td>
<td>0.2</td>
<td>99</td>
<td>0.903619633</td>
<td>99.5</td>
<td>99</td>
</tr>
<tr>
<td>1/2</td>
<td>0.3</td>
<td>99</td>
<td>0.7461070053</td>
<td>99.6</td>
<td>99</td>
</tr>
<tr>
<td>1/2</td>
<td>0.4</td>
<td>99</td>
<td>0.5968580144</td>
<td>99.7</td>
<td>99</td>
</tr>
<tr>
<td>1/2</td>
<td>0.5</td>
<td>99</td>
<td>0.4839414490</td>
<td>99.6</td>
<td>99</td>
</tr>
<tr>
<td>1/2</td>
<td>0.6</td>
<td>99</td>
<td>0.4001401755</td>
<td>99.7</td>
<td>99</td>
</tr>
<tr>
<td>2/3</td>
<td>0.2</td>
<td>98</td>
<td>0.2331824095</td>
<td>97.9</td>
<td>97</td>
</tr>
<tr>
<td>2/3</td>
<td>0.25</td>
<td>98</td>
<td>0.5680216464</td>
<td>98.3</td>
<td>98</td>
</tr>
<tr>
<td>2/3</td>
<td>0.3</td>
<td>98</td>
<td>0.816872541</td>
<td>98.4</td>
<td>98</td>
</tr>
<tr>
<td>2/3</td>
<td>0.4</td>
<td>98</td>
<td>0.9519702164</td>
<td>98.5</td>
<td>98</td>
</tr>
<tr>
<td>2/3</td>
<td>0.5</td>
<td>98</td>
<td>0.8579535313</td>
<td>98.5</td>
<td>98</td>
</tr>
<tr>
<td>2/3</td>
<td>0.6</td>
<td>98</td>
<td>0.7207307337</td>
<td>98.6</td>
<td>98</td>
</tr>
<tr>
<td>2/3</td>
<td>0.7</td>
<td>98</td>
<td>0.5962134742</td>
<td>98.6</td>
<td>98</td>
</tr>
<tr>
<td>2/3</td>
<td>0.8</td>
<td>98</td>
<td>0.4946557018</td>
<td>98.6</td>
<td>98</td>
</tr>
<tr>
<td>0.11</td>
<td>0.2</td>
<td>100</td>
<td>0.2032818378</td>
<td>99.5</td>
<td>93</td>
</tr>
<tr>
<td>0.25</td>
<td>0.2</td>
<td>100</td>
<td>0.4328421415</td>
<td>99.3</td>
<td>94</td>
</tr>
</tbody>
</table>

wp: computing precision. Accu: accuracy measured by the number of significant digits. ED: effective number of digits of precision in the result. CPU: computation time in seconds.

The reference density is computed using built-in function StableDistribution[ ] by setting the options for our code DensityOneSidedStable[ ] to {Method -> "DensityStableStable1", WorkingPrecision -> 100}.

The densities of $f_a$ for $\alpha = 1/3, 1/2, 2/3$ are plotted in Figure 6.3.

![Figure 6.3. Plots of densities $f_{1,3}$, $f_{1/2}$ and $f_{2/3}$](image)

6.4. Computing the density $f_{a,\beta}$ of two-sided stable distribution $F_{a,\beta}$

As shown in Table 6.10, there are four methods for computing the density $f_{a,\beta}$ of two-sided stable distribution $F_{a,\beta}$. The TS1 method evaluates the inverse Fourier transform numerically with
Mathematica built-in function NIntegrate[]. The TS2 method and the TS3 method are analytical formulae for the density with truncation size $M$. The TS4 method calculates $f_{\alpha,\beta}$ using Mathematica built-in function StableDistribution[] given that there is a relation between $F_{\alpha,\beta}$, $S_1(\alpha, \beta_1, \mu, \sigma_1)$ and $S(\alpha, \beta, \mu, \sigma)$. The result yielded by the TS4 method is exact.

<table>
<thead>
<tr>
<th>Method</th>
<th>Setting</th>
<th>Formula / Expression</th>
<th>Tr.</th>
<th>Eq.</th>
</tr>
</thead>
<tbody>
<tr>
<td>TS1</td>
<td>InverseFourierTransform</td>
<td>$\frac{1}{\pi} \Re \left[ e^{-\frac{1}{2} \cdot \gamma^2} \int_{0}^{\gamma} e^{-\frac{1}{2} \cdot \epsilon^2} , d\epsilon \right]$</td>
<td></td>
<td>(5.64)</td>
</tr>
<tr>
<td>TS2</td>
<td>FoxFunction</td>
<td>$X^2 H_{\frac{3}{2}} \left[ \frac{1}{2} \left( \begin{array}{c} (0, 1), \quad (1 - \gamma, \gamma) \ (1 - \epsilon, \epsilon), \quad (1 - \gamma, \gamma) \end{array} \right) \ \right.$ $\left. \text{for } 0 &lt; \alpha &lt; 1 \text{ and } x &gt; 0 \right]$ $H_{\frac{3}{2}} \left[ x \left( (\epsilon, \epsilon), \quad (-\gamma, \gamma) \ (-1, 1), \quad (-\gamma, \gamma) \right) \right.$ $\left. \right.$ $\left. \text{for } 1 &lt; \alpha \leq 2 \text{ and } x &gt; 0 \right]$ $f_{\alpha,\beta}(-x) = f_{\alpha,\beta}(x)$ $\text{for } x &lt; 0$</td>
<td>M</td>
<td>(5.68)</td>
</tr>
<tr>
<td>TS3</td>
<td>SeriesExpansion</td>
<td>$\frac{1}{\pi} \sum_{k=1}^{\infty} \frac{(\alpha+1)^k}{k!} (-x^3)^k \sin \left[ \frac{k \pi}{2} (\beta - \alpha) \right]$ $\text{for } 0 &lt; \alpha &lt; 1 \text{ and } x &gt; 0$ $\frac{1}{\pi} \sum_{k=1}^{\infty} \frac{(1-k)^{\alpha}}{k!} (-x)^k \sin \left[ \frac{k \pi}{2} (\beta - \alpha) \right]$ $\text{for } 1 &lt; \alpha \leq 2 \text{ and } x &gt; 0$ $f_{\alpha,\beta}(-x) = f_{\alpha,\beta}(x)$ $\text{for } x &lt; 0$</td>
<td>M</td>
<td>(5.72)</td>
</tr>
<tr>
<td>TS4</td>
<td>DensityStableS1</td>
<td>Use StableDistribution[] to compute $f_{\alpha,\beta}$</td>
<td></td>
<td>(5.62)</td>
</tr>
</tbody>
</table>

$f_{\alpha,\beta}$ is the density of two-sided stable distribution on $R$ with $|\beta| \leq \alpha$ for $0 < \alpha < 1$ and $|\beta| \leq 2 - \alpha$ for $1 < \alpha \leq 2$.

In Table 6.11, these four methods are compared with each other when computing the densities $f_{1/2,1/3}$ and $f_{5/3,-1/4}$ where $f_{1/2,1/3}$ is representative of $0 < \alpha < 1$ and $f_{5/3,-1/4}$ is representative of $1 < \alpha \leq 2$. It is noticeable that both $\alpha$ and $x$ have effects on the performance of each method and influence the accuracy and computation time in various ways. The TS1 method requires about 0.25 seconds for $1 < \alpha \leq 2$ but varying time from 0.5 seconds to 1.5 seconds for $0 < \alpha < 1$ depending on the value of $x$. Specifically, it takes less effort to compute the density in the vicinity of the origin for $0 < \alpha < 1$. By comparison, the TS4 method has a similar trend in computation time. The effort required near $x = 0$ is less than that of the tails especially for $1 < \alpha \leq 2$. Regarding the TS2 method and the TS3 method, computation time is very fast as they are analytical formulae. Their formulae for $0 < \alpha < 1$ are asymptotic estimates for $f_{\alpha,\beta}$ for $x \to \infty$. The accuracy of them drops as $x \to 0$ and the required truncation size $M$ increases at the same time. For $1 < \alpha \leq 2$, the decrease in accuracy is not observed. But when $x = 0$, both methods encounter division by zero. To summarize, no method works equally well for all $\alpha$ and $x$. In general, the TS2 method and the TS3 method are the best choices for computing the density $f_{\alpha,\beta}$ due to the high
accuracy and fast speed. When the evaluation at small $x$ is needed, we recommend using the TS4 method which performs well for all $x$.

Table 6.11. Accuracy of the TS1 method, the TS2 method, the TS3 method and the TS4 method for $f_{a,b}$

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>$\beta$</th>
<th>$x$</th>
<th>Reference</th>
<th>TS1 $\rho = 30$</th>
<th>$M = 100$, $\rho = 30$</th>
<th>$M = 100$, $\rho = 50$</th>
<th>$M = 100$, $\rho = 50$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ACCU Density</td>
<td>ED Accu CPU</td>
<td>ED Accu CPU</td>
<td>ED Accu CPU</td>
</tr>
<tr>
<td>1/2</td>
<td>1/3</td>
<td>-3</td>
<td>49</td>
<td>0.01426715823</td>
<td>29.8 23 0.02</td>
<td>29.5 23 0.02</td>
<td>29.6 23 0.02</td>
</tr>
<tr>
<td>1/2</td>
<td>1/3</td>
<td>-2</td>
<td>49</td>
<td>0.07054747889</td>
<td>29.5 23 0.03</td>
<td>29.4 23 0.03</td>
<td>29.5 23 0.03</td>
</tr>
<tr>
<td>1/2</td>
<td>1/3</td>
<td>-1</td>
<td>50</td>
<td>0.163570799</td>
<td>29.3 23 0.03</td>
<td>29.3 23 0.03</td>
<td>29.3 23 0.03</td>
</tr>
<tr>
<td>1/2</td>
<td>1/3</td>
<td>-0.5</td>
<td>50</td>
<td>0.3457243190</td>
<td>29.3 23 0.03</td>
<td>29.2 23 0.03</td>
<td>29.3 23 0.03</td>
</tr>
<tr>
<td>1/2</td>
<td>1/3</td>
<td>-0.1</td>
<td>50</td>
<td>0.7082245978</td>
<td>29.4 23 0.03</td>
<td>29.5 23 0.03</td>
<td>29.5 23 0.03</td>
</tr>
<tr>
<td>1/2</td>
<td>1/3</td>
<td>0</td>
<td>49</td>
<td>0.1803098662</td>
<td>29.2 23 0.03</td>
<td>29.3 23 0.03</td>
<td>29.3 23 0.03</td>
</tr>
<tr>
<td>1/2</td>
<td>1/3</td>
<td>0.01</td>
<td>50</td>
<td>0.288631439</td>
<td>29.5 23 0.03</td>
<td>29.5 23 0.03</td>
<td>29.5 23 0.03</td>
</tr>
<tr>
<td>1/2</td>
<td>1/3</td>
<td>0.02</td>
<td>50</td>
<td>0.2644173632</td>
<td>29.8 27 0.03</td>
<td>29.8 27 0.03</td>
<td>29.8 27 0.03</td>
</tr>
<tr>
<td>1/2</td>
<td>1/3</td>
<td>0.03</td>
<td>50</td>
<td>0.241967426</td>
<td>29.3 27 0.03</td>
<td>29.3 27 0.03</td>
<td>29.3 27 0.03</td>
</tr>
<tr>
<td>1/2</td>
<td>1/3</td>
<td>0.04</td>
<td>50</td>
<td>0.22840101219</td>
<td>29.2 27 0.03</td>
<td>29.2 27 0.03</td>
<td>29.2 27 0.03</td>
</tr>
<tr>
<td>1/2</td>
<td>1/3</td>
<td>0.05</td>
<td>50</td>
<td>0.2166556492</td>
<td>29.2 27 0.03</td>
<td>29.2 27 0.03</td>
<td>29.2 27 0.03</td>
</tr>
<tr>
<td>1/2</td>
<td>1/3</td>
<td>0.06</td>
<td>50</td>
<td>0.2059776059</td>
<td>29.2 27 0.03</td>
<td>29.2 27 0.03</td>
<td>29.2 27 0.03</td>
</tr>
<tr>
<td>1/2</td>
<td>1/3</td>
<td>0.07</td>
<td>50</td>
<td>0.201029075073</td>
<td>29.2 27 0.03</td>
<td>29.2 27 0.03</td>
<td>29.2 27 0.03</td>
</tr>
<tr>
<td>1/2</td>
<td>1/3</td>
<td>0.08</td>
<td>50</td>
<td>0.2056245698</td>
<td>29.2 27 0.03</td>
<td>29.2 27 0.03</td>
<td>29.2 27 0.03</td>
</tr>
<tr>
<td>1/2</td>
<td>1/3</td>
<td>0.09</td>
<td>50</td>
<td>0.2056245698</td>
<td>29.2 27 0.03</td>
<td>29.2 27 0.03</td>
<td>29.2 27 0.03</td>
</tr>
<tr>
<td>1/2</td>
<td>1/3</td>
<td>0.1</td>
<td>50</td>
<td>0.2056245698</td>
<td>29.2 27 0.03</td>
<td>29.2 27 0.03</td>
<td>29.2 27 0.03</td>
</tr>
<tr>
<td>1/2</td>
<td>1/3</td>
<td>0.5</td>
<td>50</td>
<td>0.2056245698</td>
<td>29.2 27 0.03</td>
<td>29.2 27 0.03</td>
<td>29.2 27 0.03</td>
</tr>
<tr>
<td>1/2</td>
<td>1/3</td>
<td>1.0</td>
<td>50</td>
<td>0.2056245698</td>
<td>29.2 27 0.03</td>
<td>29.2 27 0.03</td>
<td>29.2 27 0.03</td>
</tr>
</tbody>
</table>

$\rho$: computing precision. Accu: accuracy measured by the number of significant digits. ED: effective number of digits of precision in the result. CPU: computation time in seconds.

The reference density is computed using built-in function StableDistribution[] by setting the options for our code DensityTwoSidedStable[ ] to {Method -> "DensityStable51", WorkingPrecision -> 50].

The curves of $f_{1/2,1/3}$ and $f_{5/3,-1/4}$ are drawn in Figure 6.4.

![Figure 6.4. Plots of densities $f_{1/2,1/3}$ and $f_{5/3,-1/4}$](image-url)
7. Conclusion

In this chapter, numerical Laplace transform inversion algorithms such as UniG, UniT and UniE are employed to compute the densities of $C_t$, $S_t$ and $T_t$, the density of generalized one-sided stable distribution $F_{m,a}$ and the density of one-sided stable distribution $F_a$. The results calculated by inversion algorithms are compared and contrasted with those computed by analytical formulae or series expansions. Analytical formulae are usually fast but not always available for all cases. Inversion algorithms provide alternative methods for computing those densities, and often have better applicability.

The analytical formulae associated with infinitely divisible distributions include the BPY method applied to $C_t$ for any $t > 0$, the Cao method applied to $S_t$ for any $t > 0$, the Devroye method applied to $C_t$ and $S_t$, and the Tolmatz method applied to $S_{1/2}$. Note there is no analytical formula derived for the density of $T_t$ other than for $t = 1$. By contrast, UniG can invert the Laplace transforms of all three random variables $C_t$, $S_t$ and $T_t$ for any $t > 0$ to obtain densities, while UniT and UniE can invert the Laplace transform of $T_t$ for any $t > 0$ and the Laplace transforms of $C_t$ and $S_t$ for integer $t > 0$. When the Laplace transforms of $C_t$ and $S_t$ are expressed as series expansions, they can be inverted by UniT and UniE for any $t > 0$.

We recommend UniG for computing the densities of $C_t$, $S_t$ and $T_t$ because of its universal applications and fast speed, i.e., about 0.03 seconds. If only the density of $C_t$ is concerned, the BPY method is recommended as it gives very accurate result almost instantly. For the density of $T_t$, UniT is the best choice since it is faster than UniG and UniE. The computation time is 0–0.02 seconds.

Due to the variety of the stable distributions, no method works for all types except Monte Carlo simulation. The Chambers-Mallows-Stuck (1976) method simulates stable random variable $S_t(\alpha, \beta_1, 0, 1)$, while Mathematica built-in function StableDistribution[] represents stable random variable $S(\alpha, \beta, \mu, \sigma)$. With the relations between different stable distributions we have derived in this chapter, StableDistribution[], nevertheless, is able to simulate other stable random variables, i.e. $S_t(\alpha, \beta_1, \mu, \sigma_1)$, $F_a$ and $F_{a,b}$, or compute their densities exactly. Fox function representations and series expansions for the densities are available for the generalized one-sided stable $F_{m,a}$, the one-sided stable $F_a$ and the two-sided stable $F_{a,b}$. They yield the densities without any delay but the results become highly unreliable for sufficiently small $x$.

The density $f_a(x)$ for rational $\alpha = l/k$ can be expressed as a finite sum of generalized hypergeometric functions. This method is fast for small $k$ but become exponentially slower as $k$ increases. The density
The density \( f_{a,b}(x) \) also has many special cases which can be represented in terms of special functions such as the Whittaker function \( W_{k,m}(z) \), the modified Bessel function of the second kind, and the hypergeometric functions.

The numerical inversion of the Laplace transform can be applied to \( F_{m,a} \) and \( F_a \). Among three inversion algorithms, UniT is faster and more accurate than UniE and UniG. The computation time of using numerical inversion is about 0.02 seconds for \( F_{1,a} \) and \( F_a \), but longer for \( F_{2,a} \), which has a more complicated expression for the Laplace transform. Mathematica may evaluate the inverse Fourier transform for the density of \( F_{a,b} \) using the numerical integration function NIntegrate[]. This method requires varying computation time depending on the value of \( x \). When a warning message of slow convergence appears in Mathematica, we need to increase the value of the MaxRecursion option for NIntegrate[].

We have studied the applications of UniE, UniT and UniG in the computation of the infinitely divisible distributions and the stable distributions. Alternative inversion algorithms such as the Euler method, the Post-Widder method, the Gaver-Wynn-Rho algorithm, the Laguerre method and so on can be employed for comparison in the future study. The Laplace transform of \( F_{m,a} \) with \( m \geq 3 \) takes the form of a series expansion. The numerical inversion of the series expansion may be conducted to inspect the accuracy.
Chapter 4

Numerical Computation of Unit Root Distributions

1. Introduction and Motivation

Consider the first order autoregressive model (which we consider to be the basic AR(1) model)

\[ y_j = \rho y_{j-1} + \varepsilon_j, \quad j = 1, 2, \ldots, T \]
\[ y_0 = 0 \]  

(1.1)

where \( \rho \), the autoregressive parameter, is the main parameter of interest. Expression (1.1) says that the value the time series takes today is some proportion \( \rho \) of the value it took yesterday plus a noise term that is not fully explained. \( \{\varepsilon_j\} \) is a sequence of normally and independently distributed random variables with mean zero and variance \( \sigma^2 \). It can be abbreviated as \( \{\varepsilon_j\} \sim \text{NID}(0, \sigma^2) \).

We may write \( y_T \) as

\[ y_T = \varepsilon_T + \rho \varepsilon_{T-1} + \cdots + \rho^{T-1} \varepsilon_1 \]  

(1.2)

Since \( \{\varepsilon_j\} \sim \text{NID}(0, \sigma^2) \), we have

\[ \text{Var}(y_T) = \sigma^2 \left( 1 + \rho^2 + \cdots + \rho^{2(T-1)} \right) \]

\[ = \begin{cases} 
\frac{1 - \rho^T}{1 - \rho^2} \sigma^2, & |\rho| \neq 1 \\
T \sigma^2, & |\rho| = 1 
\end{cases} \]  

(1.3)

If \( |\rho| < 1 \), then \( \text{Var}(y_T) \to \frac{\sigma^2}{1 - \rho^2} \) as \( T \to \infty \) and the time series \( y_T \) converges to a covariance stationary time series. If \( |\rho| = 1 \), the time series is not stationary because \( \text{Var}(y_T) = T \sigma^2 \) is unbounded as \( T \to \infty \).
The time series with $\rho = 1$ is also called a random walk. If $|\rho| > 1$, the variance grows exponentially as $T$ increases and the time series is not stationary. In the case, the time series is said to be “explosive” (White, 1958).

Given $T$ observations $y_1, y_2, \ldots, y_T$, the least squares estimators (LSE) of $\rho$ and $\sigma^2$ are given by (e.g. Tanaka, 1996, p.73)

$$\hat{\rho} = \frac{\sum_{j=2}^{T} y_j y_{j-1}}{\sum_{j=2}^{T} y_j^2}$$

$$\hat{\sigma}^2 = \frac{1}{T-1} \sum_{j=2}^{T} (y_j - \hat{\rho} y_{j-1})^2$$

(1.4)

where $\hat{\rho}$ is derived explicitly from a minimization problem and $\hat{\sigma}^2$ is taken to be the usual unbiased estimator of $\sigma^2$. $\hat{\rho}$ is the same as the (Gaussian) maximum likelihood estimator (MLE) of $\rho$, and $\hat{\sigma}^2$ differs from the MLE of $\sigma^2$ by a factor $T/(T-1)$. Rao (1961) shows $\hat{\rho}$ is a consistent estimator for $\rho$, while Tanaka (1996, p.76) shows $\hat{\sigma}^2$ is a consistent estimator for $\sigma^2$.

Following Abadir (1995b), in this chapter we consider the model (1.1) as an artificial model that generates unit root statistics for a more realistic model such as

$$(1 - \rho L) \gamma(L) y_j = \epsilon_j$$

(1.5)

where $\{\epsilon_j\}$ is a zero-mean process satisfying conditions discussed, say, in Phillips (1987) and subsequent literature, $L$ is the lag operator, and $\gamma(L)$ is a lag polynomial all of whose roots lie outside the unit circle.

This model has been used as a generic model to describe univariate time in econometrics, with the root $\rho$ separated so to allow for testing for a unit root, and Phillips (1987) shows that all major statistics related to the unit-root process in the context of (1.5) can be written in terms of two functionals generated from (1.1). Two important practical models extending (1.1) are the AR(1) model with drift

$$y_j = \alpha + \rho y_{j-1} + \epsilon_j, \ j = 1, 2, \ldots, T$$

$$y_0 = 0$$

(1.6)

and the AR(1) model with drift and a time trend

$$y_j = \alpha + \beta j + \rho y_{j-1} + \epsilon_j, \ j = 1, 2, \ldots, T$$

$$y_0 = 0$$

(1.7)

See Hamilton (1994, Ch.17). The basic model (1.6) and (1.7) can be generalized to include a non-zero initial condition. While the limiting distributions for both models are well known in terms of functionals
of Brownian motion, expressions for the densities and distribution of the estimator of the autoregressive parameter are not yet available in the published literature. Accordingly, we will focus exclusively on statistics generated by (1.1).

We are interested in the statistics \( S_3T \) and \( S_4T \) given in Tanaka (1996, p.72) for testing the null hypothesis \( H_0: \rho = 1 \).

\[
S_3T = T (\hat{\rho} - 1) \\
S_4T = \frac{\hat{\rho}^{-1}}{\hat{\sigma} / \sqrt{\sum_{j=2}^{T} y_j^2}}
\]

(1.8)

The statistic \( S_4T \) is called the Studentized \( t \) ratio in Abadir (1995a). The expressions of \( S_3T \) and \( S_4T \) may be written as (Tanaka, 1996, p.74-76)

\[
S_3T = \frac{U_T}{V_T} \\
S_4T = \frac{U_T}{\hat{\sigma} \sqrt{V_T}}
\]

(1.9)

where

\[
U_T = \frac{1}{T} \sum_{j=2}^{T} y_{j-1} (y_j - y_{j-1}) \\
= \frac{1}{T} \sum_{j=2}^{T} y_{j-1} e_j \\
V_T = \frac{1}{T} \sum_{j=2}^{T} y_j^2
\]

(1.10)

The limiting distributions of \( S_3T \) and \( S_4T \), which are known as Dickey-Fuller distributions are of interest. Tanaka (1996, p.74-76) shows they can be expressed in terms of functionals of Brownian motion.

\[
S_3T \xrightarrow{L} S_3 = \frac{U}{T} \\
S_4T \xrightarrow{L} S_4 = \frac{U}{\sqrt{V}}
\]

(1.11)

where \( U \) and \( V \) are the limiting cases of \( U_T \) and \( V_T \)

\[
U = \mathcal{L}(U_T) = \int_0^1 w(t) \, dw(t) = \frac{1}{2} [w^2(1) - 1] \\
V = \mathcal{L}(V_T) = \int_0^1 w^2(t) \, dt
\]

(1.12)

where \( w(t) \) is Brownian motion on \([0, 1] \). Abadir (1995b) and others prefer a different parametrization, considering instead the statistics \( R = \sqrt{2} \, U \) and \( S = 2 \, V \).

There have been three main approaches in the literature towards computing the functionals in (1.10): simulation; inversion of the limiting characteristic function or Laplace transform of the relevant
densities; and, occasionally, the explicit construction of series for the relevant densities and distributions themselves. Based on the criteria of computational efficiency and numerical robustness, none has been entirely satisfactory.

The limiting distribution of $S_{3T}$ is first simulated in Fuller (1976) by Monte Carlo method. The limiting distributions of other statistics such as $S_{4T}$ can be easily simulated in a similar manner. But the accuracy obtained by Monte Carlo simulation is usually not good enough especially when one approximates the tail behavior of the distributions, and it is computationally expensive to increase the accuracy. Another disadvantage of Monte Carlo method is that only finite sample distributions can be simulated. In practice, we can only set $T$ equal to a finite number. By increasing $T$, the simulated distribution gets closer and closer to the limiting distribution but never be the limiting case. Tanaka (1996) follows White’s (1958) approach and gives a limiting characteristic function (c.f.) associated with $S_{3T}$. He then numerically inverts the c.f. using Imhof’s (1961) formula for the ratio of quadratic forms in normal variables. Abadir (1995b) proposes integral-free analytical formulae for the joint density and the joint distribution function of $(R, S)$ where $R = \sqrt{2} U$ and $S = 2 V$, and shows that the probability densities of statistics expressed in terms of $U$ and $V$ only can be generated from the joint density $f_{R,S}(r, s)$. However how to apply $f_{R,S}(r, s)$ in a specific example of generating the density still remains unclear. White (1958) derives the Laplace transform of $(U, V)$. Then, two-dimensional (2-D) numerical inversion appears to lead to the joint density $f_{U,V}(u, v)$. But inversion of the original Laplace transform is difficult and problematic. An appropriate 2-D inversion algorithm takes quite a lot of effort and only works in a small part of the region.

The aim in this chapter is to take a fresh look at the problem of computing the densities and distribution functions of the fundamental unit root statistics, drawing on some of the methods and techniques that were shown to be useful in previous chapters. With a few exceptions, the literature has contented itself with expressions such as (1.10) without ever getting to grips with expressions for the densities and distributions themselves. Our contributions are as follows: 1. We outline two new approaches to the problem: computing the relevant densities first from the joint density $f_{U,V}(u, v)$ and then from the joint density $f_{R,S}(r, s)$. 2. For the first time in the literature, various extant methods are compared and contrasted alongside each other, and alongside our own proposed methods. 3. We offer results that
illustrate contexts in which each method performs well and in which each method does not perform well. To do this, we allow the computing precision to vary to achieve optimum results. This is especially relevant when using analytical formulae to compute the relevant densities and distribution functions. Using Mathematica 9.0 to explore various facets of the unit root problem, we provide a new expression that reduces the inversion of a two-dimensional Laplace transform of the joint density \( f_{U,V}(u, v) \) to a one-dimensional problem. This is potentially a major advance on the extant literature (although the result is not established in a formal proof). We provide complete computer code that allows all the methods described in the chapter to be shared and implemented in a numerically robust way.

The rest of this chapter is organized as follows. Section 2 describes the unit root problem in detail. Section 3 presents the simulated distributions of various statistics including \( S_1 T \) and \( S_4 T \), and reports the quantiles of the distributions based on Monte Carlo simulation. In Section 4, we replicate and extend the work done by Tanaka (1996), and compute the density of \( S_3 \) by numerically inverting the relevant c.f.

Section 5 makes an important contribution. The original 2-D Laplace transform of the joint density \( f_{U,V}(u, v) \) with respect to both \( u \) and \( v \) is reduced to an one-dimensional Laplace transform of the joint density \( f_{U,V}(u, v) \) with respect to \( v \) only. The reduced Laplace transform is then inverted easily by an appropriate one-dimensional numerical inversion algorithm. A series of numerical Laplace transform inversion algorithms are fully investigated and compared in Chapter 2 on computational methods for Asian option pricing. Readers may refer to that chapter for the accuracy and efficiency of each algorithm. Section 6 studies the analytical formulae for the joint density \( f_{R,S}(r, s) \) to check their accuracy and to suggest suitable parameter settings for the formulae. In Section 7, we generate numerically the densities of \( S_3 \) and \( S_4 \) from the joint density \( f_{U,V}(u, v) \) and the joint density \( f_{R,S}(r, s) \) respectively.

Section 8 draws a conclusion from our results.
2. Unit Root Problem in AR(1) Model

Consider the Gaussian AR(1) model

\[ y_j = \rho y_{j-1} + \varepsilon_j, \quad j = 1, 2, \ldots, T \]
\[ y_0 = 0 \]  \hspace{1cm} (2.1)

where \( \{\varepsilon_j\} \sim \text{NID}(0, \sigma^2) \). Given \( T \) observations \( y_1, y_2, \ldots, y_T \), the LSE of \( \rho \) is the same as the MLE of \( \rho \), and is given by (Hamilton, 1994, p.475; Tanaka, 1996, p.14)

\[ \hat{\rho} = \frac{\sum_{t=2}^{T} y_{j-1} y_j}{\sum_{t=2}^{T} y_{j-1}^2} \]  \hspace{1cm} (2.2)

The LSE of \( \sigma^2 \) is given by (Tanaka, 1996, p.73)

\[ \hat{\sigma}^2 = \frac{1}{T-1} \sum_{t=2}^{T} (y_j - \hat{\rho} y_{j-1})^2 \]  \hspace{1cm} (2.3)

\[ \text{2.1. Stationary case} \]

When the true value \( |\rho| < 1 \), the model (2.1) is stationary. Hamilton (1994, Ch.8) shows \( \sqrt{T} \left( \hat{\rho} - \rho \right) \) has a limiting distribution as \( T \to \infty \):

\[ \sqrt{T} \left( \hat{\rho} - \rho \right) \overset{L}{\to} N(0, 1 - \rho^2) \]  \hspace{1cm} (2.4)

Substitute (2.1) into (2.2)

\[ \hat{\rho} = \frac{\sum_{t=2}^{T} y_{j-1} (\rho y_{j-1} + \varepsilon_j)}{\sum_{t=2}^{T} y_{j-1}^2} = \frac{\rho \sum_{t=2}^{T} y_{j-1}^2 + \sum_{t=2}^{T} y_{j-1} \varepsilon_j}{\sum_{t=2}^{T} y_{j-1}^2} = \rho + \frac{\sum_{t=2}^{T} y_{j-1} \varepsilon_j}{\sum_{t=2}^{T} y_{j-1}^2} \]  \hspace{1cm} (2.5)

Rearrange (2.5) and multiply both sides by \( \sqrt{T} \), and then we have

\[ \sqrt{T} \left( \hat{\rho} - \rho \right) = \frac{1}{\sqrt{T}} \frac{\sum_{t=2}^{T} y_{j-1} \varepsilon_j}{\sum_{t=2}^{T} y_{j-1}^2} \]  \hspace{1cm} (2.6)

By backward substitution, it can be induced that

\[ y_1 = \varepsilon_1 \]
\[ y_2 = \rho \varepsilon_1 + \varepsilon_2 \]
\[ y_3 = \rho^2 \varepsilon_1 + \rho \varepsilon_2 + \varepsilon_3 \]
\[ \vdots \]
\[ y_T = \varepsilon_T + \rho \varepsilon_{T-1} + \cdots + \rho^{T-1} \varepsilon_1 \]  \hspace{1cm} (2.7)

Since \( \{\varepsilon_j\} \sim \text{NID}(0, \sigma^2) \), we have
\[ y_T \sim N\left(0, \sigma^2 \left(1 + \rho^2 + \cdots + \rho^{2(T-1)}\right)\right) \] (2.8)

When \(|\rho| < 1\), we can write it as
\[ y_T \sim N\left(0, \frac{1 - \rho^{2T}}{1 - \rho^2} \sigma^2\right) \] (2.9)

As \(T \to \infty\), \(y_T\) converges in distribution
\[ y_T \overset{L}{\to} N\left(0, \frac{\sigma^2}{1 - \rho^2}\right) \] (2.10)

and the variance of \(y_{T-1}\) converges in probability
\[
\text{Var}[y_{T-1}] = E[y_{T-1}^2] - (E[y_{T-1}])^2
= E[y_{T-1}^2]
= \frac{\sigma^2}{1 - \rho^2}
\]

Note \(E[y_{T-1}^2] = \frac{1}{T} \sum_{j=2}^{T} y_{j-1}^2\). Hence, the denominator of (2.6) converges in probability
\[ \frac{1}{T} \sum_{j=2}^{T} y_{j-1}^2 \overset{P}{\to} \frac{\sigma^2}{1 - \rho^2} \] (2.12)

Hamilton (1994, p.210, eq. 8.2.7) shows the numerator of (2.6) converges in distribution
\[ \frac{1}{\sqrt{T}} \sum_{j=2}^{T} y_{j-1} e_j \overset{L}{\to} N\left(0, \frac{\sigma^2}{1 - \rho^2}\right) \] (2.13)

Therefore,
\[
\frac{1}{\sqrt{T}} \sum_{j=2}^{T} y_{j-1} e_j \overset{L}{\to} N\left(0, \frac{\sigma^2}{1 - \rho^2}\right)
\]
\[ \sqrt{T} \left(\hat{\rho} - \rho\right) \overset{L}{\to} N\left(0, 1 - \rho^2\right) \] (2.14)

When \(|\rho| < 1\), statistic \(\sqrt{T} \left(\hat{\rho} - \rho\right)\) converges in distribution at the rate \(\sqrt{T}\) to a normal distribution with zero mean and variance \(1 - \rho^2\). Hamilton (1994, p.476) pointed out that the limiting distribution is also valid when \(\rho = 1\) although it is not useful for hypothesis tests because of the implied degeneracy.

\section*{2.2. Nonstationary case}

When the true value of \(\rho\) is unity, to obtain a nondegenerate limiting distribution, \(\hat{\rho} - 1\) needs to be multiplied by \(T\) instead of \(\sqrt{T}\) (Hamilton, 1994, p.476). Then, we have
\[
(\hat{\rho} - 1) = \frac{\sum_{j=1}^{T} y_{j-1} \epsilon_j}{\sum_{j=1}^{T} y_{j-1}^2}
\]

\[
T (\hat{\rho} - 1) = \frac{\frac{1}{T} \sum_{j=1}^{T} y_{j-1} \epsilon_j}{\frac{1}{T} \sum_{j=1}^{T} y_{j-1}^2}
\]

(2.15)

When \( \rho = 1 \), the model (2.1) becomes \( y_j = y_{j-1} + \epsilon_j \) which is in fact a random walk as \( y_0 = 0 \)

\[
y_j = \epsilon_1 + \epsilon_2 + \cdots + \epsilon_j
\]

(2.16)

Since \( \{\epsilon_j\} \sim \text{NID}(0, \sigma^2) \), we have

\[
y_j \sim N(0, \sigma^2 j)
\]

(2.17)

It can be shown (Hamilton, p.475) that

\[
\frac{1}{\sigma^2 T} \sum_{j=1}^{T} y_{j-1} \epsilon_j = \frac{1}{2} \left( \frac{1}{\sigma \sqrt{T}} y_T \right)^2 - \frac{1}{2 \sigma^2} \frac{1}{T} \sum_{j=1}^{T} \epsilon_j^2
\]

(2.18)

\( y_j \sim N(0, \sigma^2 j) \) implies \( \frac{1}{\sigma \sqrt{T}} y_T \sim N(0, 1) \), so that

\[
\left( \frac{1}{\sigma \sqrt{T}} y_T \right)^2 \sim \chi^2(1)
\]

(2.19)

which is a chi-squared distribution with one degree of freedom.

Also, by the law of large numbers (LLN),

\[
\frac{1}{T} \sum_{j=1}^{T} \epsilon_j^2 \xrightarrow{p} E[\epsilon_j^2] = \text{Var}[\epsilon_j] = \sigma^2
\]

(2.20)

Therefore, (2.18) has a limiting distribution

\[
\frac{1}{\sigma^2 T} \sum_{j=1}^{T} y_{j-1} \epsilon_j \xrightarrow{L} \frac{1}{2} \chi^2(1) - \frac{1}{2}
\]

(2.21)

Given that a \( \chi^2(1) \) distribution is a gamma distribution with both its parameters equal to \( 1/2 \), (2.21) implies that the numerator is essentially a shifted gamma distribution.

Consider the denominator of (2.15)

\[
\sum_{j=1}^{T} y_{j-1}^2
\]

(2.22)

Recall that \( y_{j-1} \sim N(0, \sigma^2 (j - 1)) \), so \( E[y_{j-1}^2] = \sigma^2 (j - 1) \) and further

\[
E[\sum_{j=1}^{T} y_{j-1}^2] = \sigma^2 \sum_{j=1}^{T} (j - 1)
\]

\[
= \sigma^2 \left[ \frac{1}{2} T(T + 1) - T \right]
\]

\[
= \frac{1}{2} \sigma^2 (T^2 - T)
\]

(2.23)
To construct a statistic that could have a convergent distribution, we have to divide (2.23) by $T^2$

$$E\left[ \frac{1}{T^2} \sum_{j=1}^{T} y_{j-1}^2 \right] = \frac{1}{2} \sigma^2 \left( 1 - \frac{1}{T} \right)$$  \hspace{1cm} (2.24)

Thus the statistic converges to its limiting distribution at rate $T$, i.e. at a rate faster than in the stationary case. Furthermore, the limiting distribution is not a normal distribution but a distribution that involves a ratio of a shifted gamma distribution and a non-standard distribution in the denominator (Hamilton, 1994, p.477). The distribution as a whole is called the Dickey-Fuller distribution which is skewed to the left.

The implication of (2.14) and (2.24) together is that when we conduct least squares or Gaussian maximum likelihood estimation in the context of as fundamental a time series model as an AR(1) model, the standard estimator will converge at different rates and to different distributions in different regions of the parameter space. The problem of statistical inference to discern the region of the parameter space we are in is therefore of fundamental importance. The skewness of the Dickey-Fuller distribution makes this problem difficult because random sampling from it throws up values below one around twice as often as values above one in a context where we seek evidence of a realized statistic below one to reject a hypothesis that the autoregressive parameter is unity. This problem has been dealt with at length in the literature. Our focus, instead, is on the computation of the underlying distributions themselves, a topic that has perhaps been neglected.

2.3. Limiting distributions

For a AR(1) model

$$y_j = \rho y_{j-1} + \varepsilon_j, \ j = 1, 2, ..., T$$

$$y_0 = 0$$  \hspace{1cm} (2.25)

where $\{\varepsilon_j\} \sim \text{NID}(0, \sigma^2)$. When the true value of rho is unity, the time series becomes a random walk, i.e. $y_j = \varepsilon_1 + \cdots + \varepsilon_j$.

Tanaka (1996, p.72) considers the following four statistics
\[ S_1 r = \frac{1}{t^2} \sum_{j=1}^{T} y_j^2 \]
\[ S_2 r = \frac{1}{t^2} \sum_{j=1}^{T} (y_j - \bar{y})^2 \]
\[ S_3 r = T(\hat{\rho} - 1) \]
\[ S_4 r = \frac{\hat{\rho}^{1/2}}{\sqrt{\sum_{j=2}^{T} y_j^2}} \] (2.26)

where
\[ \hat{\rho} = \frac{\sum_{j=2}^{T} y_{j-1} y_j}{\sum_{j=2}^{T} y_{j-1}^2} \]
\[ \hat{\sigma}^2 = \frac{1}{T-1} \sum_{j=2}^{T} (y_j - \hat{\rho} y_{j-1})^2 \] (2.27)

\( S_3 r \) and \( S_4 r \) can be expressed in terms of \( U_T \) and \( V_T \) (Tanaka, 1996, p.74-76)
\[ S_3 r = \frac{U_T}{V_T} \]
\[ S_4 r = \frac{U_T}{\hat{\sigma} \sqrt{V_T}} \] (2.28)

where
\[ U_T = \frac{1}{t} \sum_{j=2}^{T} y_{j-1} (y_j - y_{j-1}) \]
\[ = \frac{1}{t} \sum_{j=2}^{T} y_{j-1} e_j \] (2.29)
\[ V_T = \frac{1}{t^2} \sum_{j=2}^{T} y_{j-1}^2 \]

\( S_4 r \) is sometimes called Dickey-Fuller \( t \)-statistic. \( S_3 r \) and \( S_4 r \) can be used to test the unit root hypothesis \( H_0: \rho = 1 \). Both the finite sample distributions and the limiting distributions are therefore of interest.

Determining the former seems even today to be an insuperable problem: there are almost no results in the literature even in the Gaussian case. A possible approach might be to use Mathematica to facilitate computing the very complicated distributions that would arise from ratios of quadratic forms in normal random variables and would involve Meijer's G function, as considered in Chapter 3. We leave this to future work. Here, we focus exclusively on the limiting distributions. One reason to justify this position is that if we really were operating in the context of unit root non-stationarity, we know from (2.15) that convergence to the limiting distribution, at rate \( T \), is relatively fast.

Using functional central limit theorem and continuous mapping theorem, we can express the limiting distributions of statistics as functionals of Brownian motion. Tanaka (1996, Ch.3.4) and Hamilton (1994, Ch.17.4) show that
\[
S_1 T \rightarrow S_1 = \sigma^2 \int_0^T w^2(t) \, dt \\
S_2 T \rightarrow S_2 = \sigma^2 \int_0^T |w(t) - t \tau w(t)|^2 \, dt \\
S_3 T \rightarrow S_3 = \left( \frac{\int_0^T w(t) \, dw(t)}{\sqrt{\int_0^T w^2(t) \, dt}} \right) = \frac{\frac{1}{2}[w^2(1) - 1]}{\sqrt{\int_0^T w^2(t) \, dt}} \\
S_4 T \rightarrow S_4 = \left( \frac{\int_0^T w(t) \, dw(t)}{\sqrt{\int_0^T w^2(t) \, dt}} \right)
\]

where \( w(t) \) is the standard Brownian motion on \([0, 1]\). \( S_3 \) and \( S_4 \) have such expressions because

\[
U_T \xrightarrow{L} U = \sigma^2 \int_0^T w(t) \, dw(t) = \frac{1}{2} \sigma^2 [w^2(1) - 1] \\
V_T \xrightarrow{L} V = \sigma^2 \int_0^T w^2(t) \, dt
\]

and \( \hat{\sigma} \) is a consistent estimator of \( \sigma \), i.e. \( \hat{\sigma} \rightarrow \sigma \). Thus, \( \sigma^2 \) appearing in both the numerators and denominators of \( S_3 \) and \( S_4 \) cancels out. Their distributions were first attacked by simulating data from the given autoregressive specification. However, it is difficult to accurately approximate the tail behavior of the distributions using simulation.

\section*{2.4. Test unit root hypothesis}

Consider a AR(1) model

\[
y_j = \rho y_{j-1} + \epsilon_j, \quad j = 1, 2, \ldots, T \\
y_0 = 0
\]

where \( \{\epsilon_j\} \sim \text{NID}(0, 1) \) and assume the true value of \( \rho \) is unity.

As an example, we want to test the unit root hypothesis for the model, that is,

\[
H_0 : \rho = 1 \\
H_1 : \rho < 1
\]

This test is one-sided. Using the statistic \( S_3 T \), we need a significantly negative value to reject \( H_0 \). Table B.5 (Case 1) of Hamilton (1994) gives the critical values for various sample sizes and significance levels. For example, at 5\% level, the critical value for sample size \( T = 50 \) is \(-7.7\), while it is \(-8.1\) for \( T \rightarrow \infty \).

Using the statistic \( S_4 T \), we also need a significantly negative value to reject \( H_0 \). Critical values can be found in Table B.6 (Case 1) of Hamilton (1994). For example, at 5\% level, the critical value for \( T = 50 \) is \(-1.95\), while it is also \(-1.95\) for \( T \rightarrow \infty \).

Note the test based on \( S_3 \) must be one-sided. This is because the test statistic has another (i.e. a third) distribution to the right of unity, which is a Cauchy distribution if the initial value is zero. If the test is
based on $S_4$, the test can either be one-sided or two-sided because the non-standard distribution in this case only occurs when the (absolute value of the) autoregressive parameter is unity.
3. Simulate the Distributions of $S_{1T}$, $S_{2T}$, $S_{3T}$ and $S_{4T}$

Using Monte Carlo method to simulate the distributions of test statistics such as $S_{1T}$, $S_{2T}$, $S_{3T}$ and $S_{4T}$ is straightforward. It does not require advanced mathematics. The simulation procedure based on arithmetic operations is illustrated as follows. First, define $e_j$ as independent pseudorandom numbers from a standard normal distribution $N(0, 1)$. Second, define $y_j$ by the AR(1) model which is in fact an iteration formula. Thirdly, construct one pseudorandom variate using the expression of the statistic. Fourthly, repeat step 1 through step 3 many times to obtain a large number of pseudorandom variates, say $10^5$ samples. Lastly, use the simulated pseudorandom variates as the data to generate a smooth kernel distribution (see Appendices A1). Then, various analysis can be performed on the resulting distribution such as visualizing the density function and the distribution function, computing the quantile and so on.

It is clear that Monte Carlo simulation is simple to implement. However, this method is time-consuming. For example, to produce $10^5$ pseudorandom variates of $S_{1T}$ with $T = 50$, we need to generate $5 \times 10^6$ independent pseudorandom numbers from the $N(0, 1)$ distribution. The computer spends time not only in producing pseudorandom numbers but also in handling this huge amount of information. Follow the procedure based on arithmetic operations and implement it in Mathematica 9.0. Our laptop takes 117 seconds to produce $10^5$ pseudorandom variates of $S_{1T}$ with $T = 50$.

We may reduce the computation time by using matrix operations. The simulation procedure based on matrix operations is quite different from that based on arithmetic operations. To produce $n$ pseudorandom variates of a statistic, say $S_{3T}$, using matrix operations, we first generate a $T \times n$ matrix $A$ of independent pseudorandom numbers from the $N(0, 1)$ distribution. Thus, each column of the matrix $A$ represents an independent sequence $\{e_j\}$ and the number of the rows is equal to $T$. Then, form a $T \times n$ matrix $B$ by accumulating the matrix $A$ within columns such that each column of the matrix $B$ is an independent $\{y_j\}$. Next, manipulate the matrix $A$ and the matrix $B$ by dropping the first row of $A$ and dropping the last row of $B$ such that each successive entry in each column of the matrix is consistent with each successive term in the relevant series in the expression of $S_{3T}$. Finally, use Mathematica’s matrix operations to obtain a vector of $n$ pseudorandom variates of $S_{3T}$. The Mathematica code we write have the ability to output multidimensional result. In other words, replacing the input value $n$ with $\{n_1, n_2, \ldots\}$ will yield an $n_1 \times n_2 \times \ldots$ array of pseudorandom variates.
Table 3.1 presents computation time for producing $10^5$ pseudorandom variates for various statistics and various $T$ using the simulation procedure based on matrix operations. All of our results are based on using the state-of-the-art package *Mathematica* 9.0. It can be seen that the computation time is reduced enormously. For the generation of $10^5$ pseudorandom variates of $S_{3,T}$ with $T = 50$, the procedure based on matrix operations takes only 1.7 seconds, which is merely one hundredth of the computation time of the procedure based on arithmetic operations. However, random variate generation in practice is not merely a matter of time. The computer memory available to the computing applications such as *Mathematica* is another important factor in the generation. *Mathematica* may abort the computation if there is no sufficient memory even if the computation time is estimated to be short. For example, considering that it takes 2.8 seconds to generate $10^5$ random variates of $S_{1,T}$ with $T = 100$, generation of $10^6$ random variates of the same statistic is supposed to take around half a minute. The time consumed is moderate but the computation fails due to insufficient memory. To overcome this problem, we may generate less random variates at a time, and repeat the computation several times to obtain the required number of random variates.

<table>
<thead>
<tr>
<th>$T$</th>
<th>$T = 10$</th>
<th>$T = 20$</th>
<th>$T = 50$</th>
<th>$T = 100$</th>
<th>$T = 200$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\bar{A}$</td>
<td>0.4</td>
<td>0.8</td>
<td>1.8</td>
<td>3.2</td>
<td>5.2</td>
</tr>
<tr>
<td>$\bar{a}$</td>
<td>0.5</td>
<td>1.6</td>
<td>2.2</td>
<td>3.9</td>
<td>6.6</td>
</tr>
<tr>
<td>$S_{1,T}$</td>
<td>0.4</td>
<td>0.7</td>
<td>1.5</td>
<td>2.8</td>
<td>4.7</td>
</tr>
<tr>
<td>$S_{2,T}$</td>
<td>0.4</td>
<td>0.8</td>
<td>1.8</td>
<td>3.2</td>
<td>5.3</td>
</tr>
<tr>
<td>$S_{3,T}$</td>
<td>0.4</td>
<td>0.7</td>
<td>1.7</td>
<td>3.0</td>
<td>5.2</td>
</tr>
<tr>
<td>$S_{4,T}$</td>
<td>0.5</td>
<td>1.1</td>
<td>2.3</td>
<td>3.9</td>
<td>6.7</td>
</tr>
</tbody>
</table>

The simulation procedure based on matrix operations is applied.

For each value of $T = 10, 20, 50,$ and 100, we simulate $2 \times 10^5$ random variates of $S_{3,T}$ in *Mathematica*, and use the built-in function `SmoothKernelDistribution[ ]` to form a smooth kernel distribution based on the random variates generated. Figure 3.1 visualizes the probability density functions of $S_{3,T}$ by applying the built-in function `PDF[ ]` to the smooth kernel distributions generated earlier. The simulated distributions of $S_{3,T}$ are left-skewed that the left tails are longer. Since the distributions are obtained by simulation, we can find that the left tails wobble while the right tails tend to be smooth. As $T$ increases, the finite sample distribution of $S_{3,T}$ converges to the limiting distribution. When $T \geq 50$, there is little difference between the finite sample distribution and the limiting distribution. This can be seen from that in Figure 3.1 the density for $T = 50$ and the density for $T = 100$ are much the same. Therefore, the finite
sample distribution is a very good approximation of the limiting distribution when $T$ is reasonably large, i.e. $T = 50$.

![Figure 3.1. Simulated distributions of $S_{3T}$ based on $2 \times 10^5$ random variates for $T = 10, 20, 50, \text{and} 100$](image)

Figure 3.2 illustrates the simulated distributions of $S_{4T}$ for $T = 10, 20, 50, \text{and} 100$ based on $2 \times 10^5$ random variates. The distributions of $S_{4T}$ are right-skewed that the right tails are longer. A unique feature of the distributions of $S_{4T}$ is that the curve of each density function has a bump on the right tail. The convergence rate of the finite sample distribution of $S_{4T}$ is slower than that of $S_{3T}$ since the density for $T = 50$ is not very close to that for $T = 100$, while it is very close in Figure 3.1. This implies that the finite sample distribution of $S_{4T}$ with at least $T = 100$ is a better estimate of the limiting distribution.

![Figure 3.2. Simulated distributions of $S_{4T}$ based on $2 \times 10^5$ random variates for $T = 10, 20, 50, \text{and} 100$](image)

By observing the definitions of the statistics $S_{1T}$ and $S_{2T}$, it is obvious that $S_{1T}$ and $S_{2T}$ are non-negative random variables. But the simulated distribution of $S_{1T}$ goes beyond the positive domain on the left as
shown in Figure 3.3 (a) where a dashed line indicates \( x = 0 \). This is caused by the built-in function SmoothKernelDistribution[ ] which can extend the estimate beyond the data supplied to make the distribution smooth based on the default setting. The problem can be corrected by using smaller bandwidth in SmoothKernelDistribution[ ]. Meanwhile, the estimated distribution becomes rougher. The simulated distributions of \( S_{2,T} \), \( S_{3,T} \) and \( S_{4,T} \) are also presented in Figure 3.3 for a cross-comparison.

![Figure 3.3. Simulated distributions of \( S_{1,T} \), \( S_{2,T} \), \( S_{3,T} \) and \( S_{4,T} \) based on \( 2 \times 10^5 \) random variates for \( T = 100 \)](image)

Using the built-in function Quantile[ ] with the smooth kernel distribution generated, we can compute the quantile of a simulated distribution. For example, the \( q^{th} \) quantile of a distribution is the inverse of the cumulative distribution function \( F(x) \) at \( q \). More specifically, it is the value \( x \) such that \( F(x) = q \).

Table 3.2 presents various quantiles of the simulated distributions of \( S_{1,T} \), \( S_{2,T} \), \( S_{3,T} \) and \( S_{4,T} \) for \( T = 100 \) together with the mean and the standard deviation of each simulated distribution. Note the smooth kernel distribution of a statistic is slightly different every time it is simulated. Hence, the \( q^{th} \) quantile in a new simulation is a realization of a random variable. A problem may arise when \( q \) is very close to zero or to one. For a statistic \( X \) and large values \( x >> 0 \), the probabilities \( P(X < -x) \) or \( P(X > x) \) become very small. The sample size in those regions can be too small to yield a quantile with good accuracy. Therefore, Monte Carlo method usually estimates the tails of a distribution poorly. However, no problem arises when calculating quantiles with \( q \) between 1% and 99%.
Table 3.2. $q$th quantile of the simulated distributions for $T = 100$ based on $2 \times 10^5$ random variates

<table>
<thead>
<tr>
<th>S</th>
<th>0.01</th>
<th>0.05</th>
<th>0.1</th>
<th>0.5</th>
<th>0.9</th>
<th>0.95</th>
<th>0.99</th>
<th>Mean</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>0.005</td>
<td>0.047</td>
<td>0.074</td>
<td>0.296</td>
<td>1.215</td>
<td>1.685</td>
<td>2.818</td>
<td>0.506</td>
<td>0.585</td>
</tr>
<tr>
<td>S2</td>
<td>0.019</td>
<td>0.035</td>
<td>0.045</td>
<td>0.120</td>
<td>0.347</td>
<td>0.460</td>
<td>0.742</td>
<td>0.167</td>
<td>0.149</td>
</tr>
<tr>
<td>S3</td>
<td>13.22</td>
<td>7.889</td>
<td>5.620</td>
<td>0.843</td>
<td>0.978</td>
<td>1.350</td>
<td>2.140</td>
<td>1.732</td>
<td>2.110</td>
</tr>
<tr>
<td>S4</td>
<td>-2.744</td>
<td>2.017</td>
<td>1.661</td>
<td>-0.493</td>
<td>0.918</td>
<td>1.326</td>
<td>2.122</td>
<td>-0.425</td>
<td>1.022</td>
</tr>
</tbody>
</table>

Monte Carlo simulation is particularly good for sketching the distribution functions and density functions of unit root statistics. But the results are usually correct to a few significant digits, and it is computationally expensive to improve the accuracy as it requires larger sample size. Hence, it is necessary to investigate other methods for accurately computing the relevant distributions.
4. Computing the Distribution of $S_3$ from the Characteristic Function

4.1. Imhof's formula for $S_3$

Recall that $S_3$ follows the limiting distribution of $S_{3,T}$ and takes the form

$$S_3 = \frac{U}{V} = \frac{\int [w^2(t)^{-1}] \int_0^t w^2(t) \, dt}{\int_0^t w^2(t) \, dt}$$  \hspace{1cm} (4.1)

The characteristic function (c.f.) of $x V - U$ is given in Tanaka (1996, p.197)

$$\phi_3(\theta; x) = E[\exp(i \theta (x V - U))] = e^{i \theta^2} \left[ \cos \sqrt{2 i \theta} x + i \theta \sin \sqrt{2 i \theta} x \right]^{-1/2}$$  \hspace{1cm} (4.2)

As $\theta \to 0$, the limit of $\phi_3(\theta, x)$ is one. As $x \to 0$, $\phi_3(\theta, x)$ reduces to $e^{i \theta^2} / \sqrt{1 + i \theta}$. We have

$$\phi_3(0, x) = 1$$

$$\phi_3(\theta, 0) = \frac{e^{i \theta^2}}{\sqrt{1 + i \theta}}$$  \hspace{1cm} (4.3)

$S_3$ is a statistic which takes the form $S = U / V$ where $P(V > 0) = 1$. Then, Imhof's (1961) formula (cited in Tanaka, 1996, p.196) for $S_3$ gives us the distribution function

$$F_3(x) = P(S_3 \leq x) = P(x V - U \geq 0) = \frac{1}{2} + \frac{1}{\pi} \int_0^{\infty} \frac{1}{\theta} \cdot \text{Im}[\phi_3(\theta; x)] \, d\theta$$  \hspace{1cm} (4.4)

Tanaka (1996, p.197) added that the variable $\theta$ should be transformed into another variable to make the integrand vanish at the origin.

If $F(x)$ is differentiable, the probability density of $S_3$ is given by

$$f_3(x) = \frac{dF_3(x)}{dx} = \frac{1}{\pi} \int_0^{\infty} \frac{1}{\theta} \cdot \text{Im}[\partial_x \phi_3(\theta; x)] \, d\theta$$  \hspace{1cm} (4.5)

where $\partial_x \phi_3(\theta; x)$ is the partial derivative of $\phi_3(\theta; x)$ with respect to $x$. Tanaka (1996, p.197) points out that the computation of $\partial \phi(\theta; x) / \partial x$ is either tedious, or the integration for computing density functions is more difficult than for distribution functions. Tanaka suggests computing $f_3(x)$ by the numerical derivative of $F_3(x)$ as

$$f_3(x) \approx \frac{F_3(x + \Delta x) - F_3(x)}{\Delta x}$$  \hspace{1cm} (4.6)

where $\Delta x$ is a small number, say $10^{-6}$. $F_3(x + \Delta x)$ and $F_3(x)$ are computed by (4.4). Computing $f_3(x)$ in
this way, Tanaka avoids examining the behavior of the integrand in (4.5) and concentrates on the computation of $F_3(x)$.

### 4.2. Evaluating the characteristic function associated with $S_3$

The c.f. in (4.2) involves the square root of a complex-valued function. In this case, the c.f. cannot be evaluated properly by a computer. The computer-generated c.f. has discontinuity points although it should be continuous for all $\theta$. Plot in Figure 4.1 the real part and the imaginary part of the computer-generated $\hat{\phi}_3(\theta; x)$ of $\phi_3(\theta; x)$ for $x = -8.039$ which is supposed to be the 5th quantile of the distribution of $S_3$.

![Figure 4.1. Graph of computer-generated $\hat{\phi}_3(\theta; x)$ for $x = -8.039$](image)

The computer-generated $\hat{\phi}_3(\theta; x)$ has a discontinuity point near $\theta = 1.1$. In fact, the graph has more than one discontinuity points on the positive real line. The number of them is possibly infinite. The greatest difficulty is that one cannot locate the exact positions of the discontinuity points.

To solve this problem, Nabeya and Tanaka (1988) proposed an algorithm as follows to obtain the correct c.f. Let $\phi(\theta)$ be the correct c.f. and $\hat{\phi}(\theta)$ be the computer-generated c.f. Start with $\theta_0 = 0$ at which $\phi(\theta_0) = \hat{\phi}(\theta_0) = 1$. Then, for $\theta_1 > \theta_0$ where $\theta_1$ is close to $\theta_0$, if

$$\left| \phi(\theta_0) + \hat{\phi}(\theta_1) \right| \leq \left| \phi(\theta_0) - \hat{\phi}(\theta_1) \right|$$

we have

$$\phi(\theta_1) = -\hat{\phi}(\theta_1)$$

(4.7)

otherwise we have

$$\phi(\theta_1) = \hat{\phi}(\theta_1)$$

(4.8)
\[
\phi(\theta_1) = \tilde{\phi}(\theta_1)
\]

Next for \(\theta_2 > \theta_1\) where \(\theta_2\) is close to \(\theta_1\), if

\[
|\phi(\theta_1) + \tilde{\phi}(\theta_2)| \leq |\phi(\theta_1) - \tilde{\phi}(\theta_2)|
\]

we have \(\phi(\theta_2) = -\tilde{\phi}(\theta_2)\), otherwise we have \(\phi(\theta_2) = \tilde{\phi}(\theta_2)\). Proceed in this way for \(\theta_i\), \(i \geq 3\) until \(\theta_i\) reach the point \(\theta\) at which we would like to evaluate.

In practice, the computation with above iteration can easily go beyond the limit of a computer without special treatment. We design the following procedure to compute the correct c.f. but avoid the redundant computations of intermediate values. First, set up a sequence of intermediate points \(\theta_i\), \(i = 1, 2, \ldots, N\) with user-specified interval \(\delta\) before the point \(\theta\) at which we would like to evaluate. The last intermediate point \(\theta_N\) is close to \(\theta\) such that \(\theta - \theta_N \leq \delta\). Then, compute correct \(\phi(\theta_i)\) at intermediate point one by one using the iteration. All intermediate values are saved globally so that they can be used when evaluate at a different point. Last, the correct value of \(\phi(\theta)\) is obtained by substituting the last intermediate value \(\phi(\theta_N)\) into the iteration.

The interval \(\delta\) between intermediate points can be chosen as large as possible to reduce the iteration providing that the first intermediate point is to the left of the first discontinuity point and there is only one discontinuity point between any two successive intermediate points. Draw the graph of correct \(\phi_3(\theta)\) in Figure 4.2. The computation time with \(\delta = 1\) in the iteration is only 0.08 seconds.

![Figure 4.2. Graph of correct \(\phi_3(\theta; x)\) for \(x = 8.039\)](image)

- **4.3. Computing the distribution function of \(S_3\)**

Once the c.f. is evaluated properly, we can compute the distribution function \(F_3(x)\) of \(S_3\) from the c.f.
\(\phi_3(\theta; x)\). Recall the Imhof’s formula and transform \(\theta\) into \(\theta = u^2\) to avoid computing the value of the integrand at the origin (Tanaka, 1996, p.198).

\[
F_3(x) = \frac{1}{2} + \frac{1}{\pi} \int_0^\infty \frac{1}{u} \text{Im}\left[\phi_3(u^2; x)\right] d\,u^2
\]

\[
= \frac{1}{2} + \frac{1}{\pi} \int_0^\infty \frac{2}{u} \text{Im}\left[\phi_3(u^2; x)\right] d\,u
\]  

(4.10)

Let \(g_3(u; x)\) denote the integrand in (4.10)

\[
g_3(u; x) = \frac{2}{u} \text{Im}\left[\phi_3(u^2; x)\right]
\]  

(4.11)

where \(g_3(u; x) \to 0\) as \(u \to 0\) and \(g_3(u; x)\) is simplified when \(x = 0\). Hence, we have

\[
g_3(0; x) = 0
\]

\[
g_3(u; 0) = \frac{2}{u} \text{Im}\left[\frac{e^{u^2/2}}{\sqrt{1+u^2}}\right]
\]  

(4.12)

Plot the integrand \(g_3(u; x)\) for various values of \(x\) in Figure 4.3 including \(x = 0\). The values of \(x = -8.039, x = -0.853\) and \(x = 1.285\) are supposed to be 5th, 50th, and 95th quantiles of the distribution of \(S_3\) respectively. The convergence of \(g_3(u; x)\) is fast when \(|x|\) is large but gets slow as \(|x| \to 0\).

![Figure 4.3. Graphs of \(g_3(u; x)\) for various values of \(x\)](image)

We can see how the integrand \(g_3(u; x)\) decreases with \(u\) at different value of \(x\) in Table 4.1. At the extreme point \(x = 0\), the convergence of \(g_3(u; x)\) becomes very slow. The integrand is still greater than 0.001 at \(u = 100\). The slow convergence at \(x = 0\) could affect the accuracy when compute the distribution function of \(S_3\).

\[
I = \int_a^b f(u) \, du = \frac{h}{3} \left[ 4 \sum_{i=1}^{n} f(u_{2i-1}) + 2 \sum_{i=1}^{n} f(u_{2i}) + f(a) + f(b) \right]
\]

(4.13)

where

\[
h = \frac{b-a}{2n}, \quad u_i = a + i \cdot h
\]

(4.14)

An alternative method is the trapezoidal rule (cited in Abate and Whitt, 1992, p.18) which approximates the integral of a function \(f(u)\) over an interval \([a, b]\) of the real line by a series

\[
I = \int_a^b f(u) \, du = h \left[ \frac{f(a)+f(b)}{2} + \sum_{k=1}^{n-1} f(a + k \cdot h) \right]
\]

(4.15)

where

\[
h = \frac{b-a}{n}
\]

(4.16)

Abate and Whitt (1992) remark that the trapezoidal rule turns out to be surprisingly better than Simpson’s rule for inversion integrals. In this chapter, we use neither Simpson’s rule nor the trapezoidal rule: the built-in Mathematica function NIntegrate[] supersedes both owing to the control over computations given on WorkingPrecision, MaxRecursion, Exclusions and so on. In Table 4.2, results computed with the same intervals of integration are but different precision are compared with the reference density. The reference density is obtained by digits matching, and accurate to the number of matching digits. It is noted that the closer to zero the point is, the more the computation time is. This is caused by

<table>
<thead>
<tr>
<th>(u)</th>
<th>(x = 0)</th>
<th>(x = -0.01)</th>
<th>(x = 0.01)</th>
<th>(x = -0.1)</th>
<th>(x = 0.1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>-0.017307</td>
<td>-0.018454</td>
<td>-0.015299</td>
<td>-0.0096279</td>
<td>0.0019193</td>
</tr>
<tr>
<td>20</td>
<td>-0.0084177</td>
<td>-0.0072818</td>
<td>-0.0041236</td>
<td>-0.00088338</td>
<td>0.00072342</td>
</tr>
<tr>
<td>30</td>
<td>0.000072414</td>
<td>-0.0013108</td>
<td>-0.0012744</td>
<td>-0.000094220</td>
<td>0.000093231</td>
</tr>
<tr>
<td>40</td>
<td>0.00011864</td>
<td>-0.00019815</td>
<td>0.00015898</td>
<td>0.000013346</td>
<td>0.000010558</td>
</tr>
<tr>
<td>50</td>
<td>-0.00072660</td>
<td>0.000025604</td>
<td>0.00020377</td>
<td>-1.3765 \times 10^{-6}</td>
<td>1.8514 \times 10^{-7}</td>
</tr>
<tr>
<td>60</td>
<td>0.00044128</td>
<td>-0.000042532</td>
<td>-0.00011323</td>
<td>-1.6993 \times 10^{-7}</td>
<td>-2.9330 \times 10^{-7}</td>
</tr>
<tr>
<td>70</td>
<td>-0.00038436</td>
<td>0.000050933</td>
<td>0.000052274</td>
<td>3.1054 \times 10^{-8}</td>
<td>-1.2675 \times 10^{-7}</td>
</tr>
<tr>
<td>80</td>
<td>0.00027463</td>
<td>-0.000015547</td>
<td>-0.000027230</td>
<td>8.3243 \times 10^{-9}</td>
<td>5.8907 \times 10^{-9}</td>
</tr>
<tr>
<td>90</td>
<td>0.000072559</td>
<td>-0.000013184</td>
<td>8.5764 \times 10^{-8}</td>
<td>1.4607 \times 10^{-7}</td>
<td>-1.2130 \times 10^{-7}</td>
</tr>
<tr>
<td>100</td>
<td>-0.00016160</td>
<td>-3.5874 \times 10^{-6}</td>
<td>4.8068 \times 10^{-6}</td>
<td>2.4201 \times 10^{-10}</td>
<td>1.5642 \times 10^{-10}</td>
</tr>
</tbody>
</table>
oscillations of the integrand \( g_3(u; x) \). The computation at \( x = 0 \) takes surprisingly a little time though oscillations at the origin are largest. This is because the integrand \( g_3(u; x) \) has a much simpler form at \( x = 0 \). Accuracy decreases as \(|x|\) moves towards zero due to the increase of oscillations.

Table 4.2. Distribution function \( F_3(x) \) of \( S_3 \)

<table>
<thead>
<tr>
<th>x</th>
<th>Reference</th>
<th>NIntegrate</th>
<th>NIntegrate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accu</td>
<td>ED</td>
<td>Result</td>
</tr>
<tr>
<td>-5</td>
<td>15</td>
<td>0.124479679</td>
<td>19.5</td>
</tr>
<tr>
<td>-4</td>
<td>16</td>
<td>0.1703431006</td>
<td>19.7</td>
</tr>
<tr>
<td>-3</td>
<td>15</td>
<td>0.235560644</td>
<td>19.9</td>
</tr>
<tr>
<td>-2</td>
<td>15</td>
<td>0.3306001429</td>
<td>20.3</td>
</tr>
<tr>
<td>-1</td>
<td>12</td>
<td>0.4734701398</td>
<td>21.3</td>
</tr>
<tr>
<td>-0.5</td>
<td>10</td>
<td>0.5698317975</td>
<td>20.9</td>
</tr>
<tr>
<td>-0.1</td>
<td>7</td>
<td>0.6588028</td>
<td>20.6</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>0.6827</td>
<td>20.6</td>
</tr>
<tr>
<td>0.1</td>
<td>6</td>
<td>0.707163</td>
<td>20.5</td>
</tr>
<tr>
<td>0.5</td>
<td>10</td>
<td>0.807214584</td>
<td>20.4</td>
</tr>
<tr>
<td>1</td>
<td>13</td>
<td>0.9122868642</td>
<td>20.3</td>
</tr>
<tr>
<td>2</td>
<td>15</td>
<td>0.9892637609</td>
<td>20.3</td>
</tr>
</tbody>
</table>

NIntegrate: Compute the density using the formula (4.10) and evaluate the integral with Mathematica built-in function NIntegrate[].

wp: computing precision. MP: MachinePrecision. Accu: accuracy measured by the number of significant digits. ED: effective number of digits of precision in the result. CPU: computing time in seconds.

The reference density is obtained by digits matching with the following option specifications for our code DistributionFunctionS3[]:

- (Method -> "NIntegrate", "s" -> 48), WorkingPrecision -> 25
- (Method -> "NIntegrate", "s" -> 43), WorkingPrecision -> 25)

The graph of the distribution function of \( S_3 \) is plotted in Figure 4.4 using Mathematica built-in function Plot[].

![Figure 4.4. Curve of distribution function \( F_3(x) \) of \( S_3 \)](image)

4.4. Computing the probability density of \( S_3 \)

Tanaka (1996, p.199) computes the density \( f_3(x) \) of \( S_3 \) using the approximation (4.6) so that he can use the established function \( F_3(x) \) without resorting to the formula (4.5) which involves the derivative of the
Recall the approximation to the density of $S_3$ given by
\[
f_3(x) \approx \frac{F_{S_3}(x + \Delta x) - F_{S_3}(x)}{\Delta x}
\]  
(4.17)

By assigning an appropriate value to $\Delta x$, the approximation gives the density $f_3(x)$. We can also use the formula (4.5) to compute $f_3(x)$ as
\[
f_3(x) = F''(x) = \frac{1}{\pi} \int_0^\infty \Im \{\partial_x \phi_3(\theta; x)\} d\theta
\]  
(4.18)

The formula (4.5) will be investigated and compared with the approximation (4.6) to find the merits of these two methods. This aspect was not studied by Tanaka (1996).

The partial derivative of $\phi_3(\theta; x)$ with respect to $x$ can be easily computed using Mathematica built-in function $D[\ ]$ although the resulting expression looks complicated.
\[
\partial_x \phi_3(\theta; x) = \frac{e^{i\theta} \theta^2 \left[2 \sqrt{1 + \theta x} \cos(\sqrt{2 + x} \theta) - \sqrt{2 + x} \sin(\sqrt{2 + x} \theta)\right]}{8 \left(\theta + i \sqrt{2 + x}\right)^{\frac{3}{2}} \left[\cos(\sqrt{2 + x} \theta) + \frac{\sqrt{2 + x} - \sqrt{2}}{\sqrt{2}}\right]}
\]  
(4.19)

where the limits of $\partial_x \phi_3(\theta; x)$ as $\theta \to 0$ and $x \to 0$ respectively are obtained using Mathematica built-in function Limit[ ]. Then, we have
\[
\partial_x \phi_3(0; x) = 0,
\]
\[
\partial_x \phi_3(\theta; 0) = \frac{e^{i\theta} \theta (3 + i \theta)}{6 \sqrt{1 + i \theta}}
\]  
(4.20)

The computer-generated $\partial_x \phi_3(\theta; x)$ of $\partial_x \phi_3(\theta; x)$ for $x = -8.039$ has discontinuity points as shown in Figure 4.5 due to the square root of the complex-valued function.

![Graph of computer-generated $\partial_x \phi_3(\theta; x)$ for $x = -8.039$](image)

**Figure 4.5.** Graph of computer-generated $\partial_x \phi_3(\theta; x)$ for $x = -8.039$

Using Nabeya and Tanaka’s (1988) algorithm discussed in Section 4.2, we can obtain correct $\partial_x \phi_3(\theta; x)$
as shown in Figure 4.6.

Figure 4.6. Graph of correct $\delta_x \phi_3(\theta; x)$ for $x = -8.039$

By applying the transformation of $\theta = u^2$ to (4.5), the formula for $f_3(x)$ becomes

$$f_3(x) = \frac{1}{\pi} \int_0^\infty \frac{1}{u^2} \text{Im} \left[ \partial_x \phi_3(u^2; x) \right] d u^2$$

$$= \frac{1}{\pi} \int_0^\infty \frac{2}{u} \text{Im} \left[ \partial_x \phi_3(u^2; x) \right] d u \quad (4.21)$$

Let $h_3(u; x)$ denote the integrand in (4.21)

$$h_3(u; x) = \frac{2}{u} \text{Im} \left[ \partial_x \phi_3(u^2; x) \right] \quad (4.22)$$

where the limit of $h_3(u; x)$ is zero as $u \to 0$, and $h_3(u; x)$ at $x = 0$ has a reduced form. Then, we have

$$h_3(0; x) = 0$$

$$h_3(u; 0) = \frac{1}{3u} \text{Im} \left[ \frac{e^{x^2/u^2} u^3}{\sqrt{1+iu^2(u^2-1)}} \right] \quad (4.23)$$

Note that $h_3(0; x) = 0$ allows us to dispense with computing the value of the integrand at the origin.

Figure 4.7 gives the graphs of the integrand $h_3(u; x)$ for various values of $x$ where $x = -8.039$, $x = -0.853$ and $x = 1.285$ are supposed to be $5^{th}$, $50^{th}$, and $95^{th}$ quantiles of the distribution of $S_3$ respectively. The integrand converges fast as $|x|$ is moderately large but slowly as $|x|$ is small. When $x = 0$, the integrand is oscillating and seems it never converges. This may cause a problem when computing $f_3(x)$ at $x = 0$. 
With the default setting $\delta = 1$ for the intervals between intermediate points, we notice $h_3(\nu; x)$ has discontinuity points on $0.454585 < x < 0.571429$ as shown in Figure 4.8. Then, we reduce the interval to $\delta = 0.5$. Discontinuity points still occur on $0.542599 < x < 0.571429$, but the interval for $x$ has shrunk and the upper endpoint is the same. Continuing to reduce the interval, we find $\delta = 0.1$ causes discontinuity points on $0.570280 < x < 0.571429$, $\delta = 0.02$ causes discontinuity points on $0.571382 < x < 0.571429$, and $\delta = 0.01$ causes discontinuity points on $0.571417 < x < 0.571429$. It appears that the closer to $0.571429$ from the left the parameter $x$ is, the smaller $\delta$ is required to eliminate discontinuity points. We recommend setting the value of $\delta$ as

$$
\begin{align*}
\delta &= 0.5, & \text{for } 0.454585 < x \leq 0.542599 \\
\delta &= 0.1, & \text{for } 0.542599 < x \leq 0.570280 \\
\delta &= 0.02, & \text{for } 0.570280 < x \leq 0.571382 \\
\delta &= 0.01, & \text{for } 0.571382 < x \leq 0.571417 \\
\delta &= 1, & \text{otherwise}
\end{align*}
$$

(4.24)

Note the above settings does not root out all discontinuity points. They still exist on $0.571417 < x < 0.571429$ but the interval of $x$ is now very narrow. Our code hS3[ ] automatically use the above settings if the value of $\delta$ is not specified.
Further experiments for computing the density $f_3(x)$ using $h_3(u; x)$ suggest for $x \geq 0.571429$ the density is negative but the absolute value is correct. Then we have the following conjecture. The graph of $h_3(u; x)$ for $x \geq 0.571429$ is incorrect just like the incorrect graphs of $h_3(u; x)$ for $x = 0.571429$ in Figure 4.8. The correct graph of the integrand in (4.21) for $x \geq 0.571429$ should be plotted with $-h_3(u; x)$.

When there is an discontinuity point close to the origin, Nabeya and Tanaka’s (1988) algorithm corrects the graph such that the first biggest wave is above the $x$-axis. When $x \geq 0.571429$, we can think in a way that Nabeya and Tanaka’s algorithm fails to correct the graph because there is no discontinuity point close to the origin. Therefore, we should take the integrand in (4.21) as $-h_3(u; x)$ when $x \geq 0.571417$. 

Figure 4.8. Graphs of $h_3(u; x)$ for various $x$ with different $\delta$. 

Here are the graphs for various $x$ and $\delta$: 

- $x = 0.454585, \delta = 1$ 
- $x = 0.454586, \delta = 1$ 
- $x = 0.571428, \delta = 1$ 
- $x = 0.571429, \delta = 1$ 
- $x = 0.542599, \delta = 0.5$ 
- $x = 0.542600, \delta = 0.5$ 
- $x = 0.571428, \delta = 0.5$ 
- $x = 0.571429, \delta = 0.5$ 
- $x = 0.570280, \delta = 0.1$ 
- $x = 0.570281, \delta = 0.1$ 
- $x = 0.571428, \delta = 0.1$ 
- $x = 0.571429, \delta = 0.1$ 
- $x = 0.571382, \delta = 0.02$ 
- $x = 0.571383, \delta = 0.02$ 
- $x = 0.571428, \delta = 0.02$ 
- $x = 0.571429, \delta = 0.02$ 
- $x = 0.571417, \delta = 0.01$ 
- $x = 0.571418, \delta = 0.01$ 
- $x = 0.571428, \delta = 0.01$ 
- $x = 0.571429, \delta = 0.01$
So we rewrite the formula (4.21) as

\[ f_3(x) = \frac{1}{\pi} \int_0^\infty h_3(u; x) \, du \]  

(4.25)

where

\[ \tilde{h}_3(u; x) = h_3(u; x), \quad x < 0.571417 \]
\[ \tilde{h}_3(u; x) = -h_3(u; x), \quad x \geq 0.571417 \]  

(4.26)

The interval \( 0.571417 \leq x < 0.571429 \) is included because the required \( \delta \) is too small to correct the graph. Thus a small error will be introduced to the density \( f_3(x) \) on the interval \( 0.571417 \leq x < 0.571429 \). Figure 4.9 plots graphs of the integrand \( h_3(u; x) \) in (4.21) and the integrand \( \tilde{h}_3(u; x) \) in (4.25) for various values of \( x \).

![Graphs of \( h_3(u; x) \) and \( \tilde{h}_3(u; x) \) for various \( x \)](image)

The probability density of \( S_3 \) can then be computed by either the approximation (4.6) or the formula (4.25). Table 4.3 gives the results computed by both methods and the accuracy related to each result. The integrals involved in the approximation (4.6) are evaluated by Mathematica built-in function NIntegrate[] by default. Our code ProbabilityDensityS3[] also allows us to change the evaluation method to the Simpson’s rule or the trapezoidal rule. The reference density is obtained by digits matching using two different parameter settings. The accuracy of the result decreases as expected when \(|x|\) approaches zero. The accuracy at \( x = 0 \) is only two digits even for the reference density.

The formula (4.25) is better than the approximation (4.6) considering the fact that it yields slightly better results and uses less computation time. Both methods encounter difficulties in the vicinity of \( x = 0 \) due
to the nature of the integrand: slow convergence.

<table>
<thead>
<tr>
<th>x</th>
<th>Reference</th>
<th>Approximation</th>
<th>NIntegrate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accu</td>
<td>Density</td>
<td>ED</td>
</tr>
<tr>
<td>-3</td>
<td>14</td>
<td>0.07789546841</td>
<td>13.2</td>
</tr>
<tr>
<td>-2</td>
<td>14</td>
<td>0.1151198597</td>
<td>13.5</td>
</tr>
<tr>
<td>-1</td>
<td>11</td>
<td>0.1749471596</td>
<td>14.5</td>
</tr>
<tr>
<td>-0.5</td>
<td>9</td>
<td>0.209746421</td>
<td>14.2</td>
</tr>
<tr>
<td>-0.1</td>
<td>5</td>
<td>0.23564</td>
<td>13.9</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>0.24</td>
<td>13.8</td>
</tr>
<tr>
<td>0.1</td>
<td>5</td>
<td>0.24727</td>
<td>13.8</td>
</tr>
<tr>
<td>0.5</td>
<td>10</td>
<td>0.2448722555</td>
<td>13.6</td>
</tr>
<tr>
<td>1</td>
<td>13</td>
<td>0.1627628744</td>
<td>13.3</td>
</tr>
<tr>
<td>2</td>
<td>14</td>
<td>0.02346194296</td>
<td>12.4</td>
</tr>
</tbody>
</table>

Approximation: compute the density using the approximation (4.6). NIntegrate: compute the density using the formula (4.25) and evaluate the integral with Mathematica built-in function NIntegrate[].

dx: Δx in the approximation (4.6). wp: computing precision. Accu: accuracy measured by the number of significant digits. ED: effective number of digits of precision in the result. CPU: computation time in seconds.

The reference density is obtained by digits matching with the following option specifications for our code ProbabilityDensityS3[]:

Method -> ["Approximation", "NIntegrate", "s" -> 48, "dx" -> 10^-7], WorkingPrecision -> 25] vs

Method -> ["Approximation", "NIntegrate", "s" -> 43, "dx" -> 10^-7], WorkingPrecision -> 25].

Figure 4.10 reveals the curve of the probability density \( f_3(x) \) of \( S_3 \) where \( f_3(x) \) is left-skewed. When we draw the graph of \( f_3(x) \), machine precision will suffice for the computations.

It should be noted that there is a distortion of \( f_3(x) \) on the interval \(-0.02 < x < 0.02\) as depicted in Figure 4.11. This distortion is caused by slow convergence of the integrand. If increasing the convergence is impossible or very difficult, interpolation around \( x = 0 \) could be a reasonable solution.
Figure 4.11. Distortion of the curve of $f_3(x)$ in the vicinity of $x = 0$
5. Joint Density of $U$ and $V$

White (1958) derives an explicit expression for the Laplace transform of the two random variables $U$ and $V$. This Laplace transform denoted by $\tilde{f}(\alpha, \beta)$ is two-dimensional, and the joint density $f_{U,V}(u, v)$ of $U$ and $V$ is embedded in it. Specifically, $\tilde{f}(\alpha, \beta)$ is the two-dimensional Laplace transform of $f_{U,V}(u, v)$ with respect to $u$ and $v$. By setting $\alpha = 0$ or $\beta = 0$, we can obtain the Laplace transform of $V$ and the Laplace transform of $U$ respectively. The density $f_{U}(v)$ and the density $f_{U}(u)$ are then available by numerical inversion. An important finding in this section is that, we are able to deduce a reduced Laplace transform $\hat{f}(u, \beta)$ of $f_{U,V}(u, v)$ with respect to $v$ only. The joint density $f_{U,V}(u, v)$ can then be computed from $\hat{f}(u, \beta)$ by an one-dimensional inversion algorithm instead of from $\tilde{f}(\alpha, \beta)$ by a two-dimensional inversion algorithm. Using $\hat{f}(u, \beta)$ also avoids inversion of a two-sided Laplace transform.

In Chapter Two, we investigated various inversion algorithms and use them to numerically invert the Laplace transform of an Asian option price. In this chapter, we will use them to obtain the densities $f_{V}(v)$, $f_{U}(u)$ and $f_{U,V}(u, v)$ from the relevant Laplace transforms. The inversion algorithms considered here include Abate and Whitt’s (1995) Euler method (Euler) and Post-Widder method (PW); Shaw’s (1998) Bromwich integration (Bromwich); Abate and Valkó’s (2004) Gaver-Wynn-Rho algorithm (GWR) and fixed Talbot algorithm (FT); Abate and Whitt’s (2006) Gaver-Stehfest algorithm in the unified framework (UniG), Euler algorithm in the unified framework (UniE) and Talbot algorithm in the unified framework (UniT). Among the one-dimensional inversion algorithms listed above, UniG, UniE and UniT can be combined to form nine two-dimensional inversion algorithms (Abate and Whitt, 2006): UniTG, UniTT, UniEG, UniET, UniTE, UniGT, UniGG, UniEE and UniGE where the operator $G$ denotes Gaver-Stehfest, the operator $E$ denotes Euler, and the operator $T$ denotes Talbot. When using UniTG for the inversion, the first operator $T$ applies to the outer loop in the unified framework, while the second operator $G$ applies to the inner loop.

The joint density $f_{U,V}(u, v)$ of $(U, V)$ can be used to generate the densities of almost all unit root statistics expressed in terms of $U$ and $V$ only such as $S_3$ and $S_4$. 
5.1. The Laplace transform of the joint distribution \((U, V)\)

Recall the representations of the limiting distributions \(S_3\) and \(S_4\)

\[
S_3 = \frac{U}{V} = \frac{\frac{1}{2}[w^2(1)-1]}{\int_0^1 w^2(t) \, dt} = \frac{\int_0^1 w(t) \, dw(t)}{\int_0^1 w^2(t) \, dt}
\]

\[
S_4 = \frac{U}{\sqrt{V}} = \frac{\int_0^1 w(t) \, dw(t)}{\sqrt{\int_0^1 w^2(t) \, dt}}
\]

where \(w(t)\) is the standard Brownian motion. From the above expression, we can deduce that

\[
U = \int_0^1 w(t) \, dw(t) = \frac{1}{2}[w^2(1)-1]
\]

\[
V = \int_0^1 w^2(t) \, dt
\]

We know \(w^2(t) \geq 0\), so \((w^2(t)-1)/2 \geq -1/2\). This implies random variable \(U\) takes on both negative values and positive values on the set \(\{u : -1/2 \leq u < \infty\}\). Since \(w^2(t) \geq 0\) at any time \(t\), the integral \(\int_0^1 w^2(t) \, dt\) is always positive. Thus, the random variable \(V\) is positive on the set \(\{v : 0 < v < \infty\}\). Subsequently, the domain of \((U, V)\) is \(\{(u, v) : -1/2 \leq u < \infty, 0 < v < \infty\}\). For points outside the domain, we set the joint density \(f_{U,V}(u, v)\) equal to zero.

White (1958) gives the Laplace transform of the limiting distribution \((U, V)\)

\[
\tilde{f}(\alpha, \beta) = E[\exp(-\alpha U - \beta V)]
\]

\[
= \int_0^\infty \int_{-1/2}^0 e^{-u \beta - v \alpha} f_{U,V}(u, v) \, du \, dv
\]

\[
= e^{\alpha/2} \left( \cosh \sqrt{2 \beta} + \frac{\alpha}{\sqrt{2 \beta}} \sinh \sqrt{2 \beta} \right)^{-1/2}
\]

(5.3)

where the joint density \(f_{U,V}(u, v)\) is embedded in the above two-dimensional Laplace transform, and may be obtained by applying a suitable two-dimensional Laplace transform inversion algorithm. If we let \(\alpha = i \theta\) and \(\beta = -i \theta x\), Laplace transform (5.3) becomes the characteristic function (4.2). Before we deal with \(f_{U,V}(u, v)\), it is worth taking a further look at the expression.

When \(\alpha = 0\), (5.3) reduces to the Laplace transform of \(V\)

\[
E[e^{-\beta V}] = \int_0^\infty e^{-\beta v} f_V(v) \, dv = \left( \cosh \sqrt{2 \beta} \right)^{-1/2}
\]

(4.5)

When \(\beta \to 0\), \(\frac{\sinh \sqrt{2 \beta}}{\sqrt{2 \beta}} \to 1\) and (5.3) reduces to the Laplace transform of \(U\)

\[
E[e^{-\alpha U}] = \int_{-1/2}^\infty e^{-\alpha u} f_U(u) \, du = \frac{e^{\alpha/2}}{\sqrt{\alpha+1}}
\]

(5.5)
Note that $U$ takes on values in the range of $u \geq -1/2$ and is zero otherwise. The expression (5.5) is therefore a bilateral Laplace transform or a two-sided Laplace transform. By comparison, the expression (5.4) is the usual one-sided Laplace transform.

5.2. Computation of the density $f_V(v)$

Recall the non-negative random variable $C_t$ which is studied by Pitman and Yor (2003) and called to have an infinitely divisible distribution. The definition of $C_t$ is given by the following Laplace transform

$$E[e^{-\lambda C_t}] = \int_0^\infty e^{-\lambda x} f_C(x) \, dx = \left( \frac{1}{\cosh \sqrt{2\lambda}} \right)^t$$  \hspace{1cm} (5.6)

By comparing the Laplace transform of $V$ with the Laplace transform of $C_{1/2}$

$$E[e^{-\beta V}] = \left( \cosh \sqrt{2 \beta} \right)^{-1/2} = \left( \frac{1}{\cosh \sqrt{2 \beta}} \right)^{1/2}$$

$$E[e^{-\lambda C_{1/2}}] = \left( \frac{1}{\cosh \sqrt{2 \lambda}} \right)^{1/2}$$

we have

$$V = C_{1/2}$$  \hspace{1cm} (5.7)

The methods for computing the density of $C_t$, for $t > 0$, is well discussed in Chapter Three. For example, the formula for the density of $C_t$, referred to as BPY, is given by Biane, Pitman and Yor (2001) in the form of a series

$$f_C(x) = \frac{P(C_t \leq x)}{dx} = \frac{2^\nu}{\Gamma(\nu)} \sum_{n=0}^{\infty} (-1)^n \frac{\Gamma(n+\nu)}{\Gamma(n+1)} \sqrt{\frac{2\pi x}{2\nu}} \exp\left( -\frac{(2\nu\nu+\nu^2)^2}{2x} \right)$$  \hspace{1cm} (5.8)

The density $f_V(v)$ is then computed by calculating $f_{C_{1/2}}$.

Alternatively, the density $f_V(v)$ can be computed by applying a proper numerical inversion algorithm to its Laplace transform. It turns out that only UniG and GWR works for the inversion of (5.4) but any other methods fail. We notice that UniG and GWR use the same inversion technique but with different acceleration algorithms. Table 5.1 reveals the densities of $V$ at various points as well as the accuracy of different methods. We recommend BPY method with parameters setting of $M = 15$ and $w_p = 20$ for computing $f_V(v)$. UniG and GWR are also accurate and fast although they are slightly inferior to BPY. This conclusion made here agrees with the results in Chapter Three.
Table 5.1. Density \( f_v(v) \)

<table>
<thead>
<tr>
<th>( v )</th>
<th>Reference Density</th>
<th>( \text{BPY} )</th>
<th>( \text{UnIG} )</th>
<th>( \text{GWR} )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( \text{Accu} )</td>
<td>Density</td>
<td>( \text{CPU} )</td>
<td>( \text{Accu} )</td>
</tr>
<tr>
<td>1.0</td>
<td>98</td>
<td>0.0001526951</td>
<td>0.00020513</td>
<td>0.0001526951</td>
</tr>
<tr>
<td>2.0</td>
<td>99</td>
<td>2.071137264</td>
<td>2.67111</td>
<td>2.071137264</td>
</tr>
<tr>
<td>3.2</td>
<td>99</td>
<td>1.688168107</td>
<td>1.991</td>
<td>1.688168107</td>
</tr>
<tr>
<td>5.3</td>
<td>99</td>
<td>1.131668871</td>
<td>1.393</td>
<td>1.131668871</td>
</tr>
<tr>
<td>8.4</td>
<td>99</td>
<td>0.8148080252</td>
<td>0.937</td>
<td>0.8148080252</td>
</tr>
<tr>
<td>12.7</td>
<td>99</td>
<td>0.6175442462</td>
<td>0.673</td>
<td>0.6175442462</td>
</tr>
<tr>
<td>20.7</td>
<td>99</td>
<td>0.4840288324</td>
<td>0.515</td>
<td>0.4840288324</td>
</tr>
<tr>
<td>37.7</td>
<td>99</td>
<td>0.3850315024</td>
<td>0.348</td>
<td>0.3850315024</td>
</tr>
<tr>
<td>68.8</td>
<td>99</td>
<td>0.3173890190</td>
<td>0.284</td>
<td>0.3173890190</td>
</tr>
<tr>
<td>99.9</td>
<td>99</td>
<td>0.261918012</td>
<td>0.228</td>
<td>0.261918012</td>
</tr>
<tr>
<td>1.9</td>
<td>99</td>
<td>0.217997738</td>
<td>0.194</td>
<td>0.217997738</td>
</tr>
<tr>
<td>2.9</td>
<td>99</td>
<td>0.1906621030</td>
<td>0.167</td>
<td>0.1906621030</td>
</tr>
<tr>
<td>15.0</td>
<td>99</td>
<td>0.088592527</td>
<td>0.075</td>
<td>0.088592527</td>
</tr>
</tbody>
</table>


wp: computing precision. Accu: accuracy measured by the number of significant digits. ED: effective number of digits of precision in the result. CPU: computation time in seconds.

The reference densities are obtained by BYP with parameters settings: \( M = 60 \), wp = 100 versus \( M = 100 \), wp = 100.

Figure 5.1 plots the density \( f_v(v) \) where the distribution of \( V \) is right-skewed.

![Figure 5.1. Curve of the density \( f_v(v) \)](image)

5.3. Computation of the density \( f_v(u) \)

Recall the Laplace transform (5.5)

\[
E[e^{-uU}] = \int_{0}^{\infty} e^{-uU} f_v(u) \, du = \frac{e^{uv}}{\sqrt{v^2 + 1}}
\]

which is the two-sided Laplace transform.

5.3.1. Obtaining the density \( f_v(u) \) from the density of \( \chi^2(1) \)

We know from (2.21) that (5.5) is the Laplace transform of a \( \chi^2(1) - 1 \)/2 distributed random variable.

In other words, we have
\[ U = \frac{1}{2} \left( \chi^2(1) - 1 \right) \] (5.10)

The p.d.f. of \( \chi^2(1) \) is readily available
\[
f_{\chi^2}(x) = \frac{e^{-\frac{x}{2}}}{\sqrt{2\pi x}}, \quad x > 0
= 0 \quad \text{elsewhere}
\] (5.11)

We can find the p.d.f. of \( U \) from the p.d.f. of \( \chi^2(1) \) by following Hogg and Craig (1995, pp.168-170).

Let random variable \( X \) be \( \chi^2(1) \). Then, we have \( U = (X - 1) / 2 \) where \( u = h(x) = (x - 1) / 2 \) defines a one-to-one transformation that maps the set \( A = \{ x : x \geq 0 \} \) onto the set \( B = \{ u : u \geq -1 / 2 \} \). The inverse of \( u = h(x) \) is \( x = g(u) = 2u + 1 \), so that the derivative of \( g(u) \) is \( g'(u) = 2 \). Then, the p.d.f. of \( U \) is given by
\[
f_U(u) = f_X(g(u)) \cdot g'(u)
= \frac{e^{-\frac{u^2}{2}}}{\sqrt{2\pi(2u+1)}} \cdot 2
= 2 \frac{e^{-\frac{u^2}{2}}}{\sqrt{2\pi(2u+1)}}, \quad u > -1 / 2
= 0 \quad \text{elsewhere}
\] (5.12)

Now, we can plot the density \( f_U(u) \) as in Figure 5.2.

![Figure 5.2. Curve of the density \( f_U(u) \)](image)

- **5.3.2. Deducing the density \( f_U(u) \) by applying Bromwich integral to the two-sided Laplace transform**

Next, we deduce the density function of \( U \) by inverting its Laplace transform with Bromwich integral, and undertake the calculations to illustrate the techniques involved. Although the eventual result is well know, the very same method will be employed later in the context of real interest, reducing White’s (1958) Laplace transform, where the calculations are non-standard.
The Bromwich integral for the inverse Laplace transform is also valid for the two-sided Laplace transform. The Bromwich integral is defined by

\[
\mathcal{L}^{-1}\{\hat{f}(\lambda)\}(t) = \frac{1}{2\pi i} \int_{a-i\infty}^{a+i\infty} e^{\lambda t} \hat{f}(\lambda) d\lambda
\]  

(5.13)

where the integration is done along the vertical line Re(\(\lambda\)) = a such that all singularities are to the left of it. Applying the Bromwich integral and setting \(\alpha = a + is\), we have

\[
\mathcal{L}^{-1}\{E[e^{-\alpha U}]\} = f_U(u) = \frac{1}{2\pi i} \int_{a-i\infty}^{a+i\infty} e^{\alpha u} \frac{e^{\alpha^2}}{\sqrt{\alpha+i}} d\alpha
\]

\[
= \frac{1}{2\pi i} \int_{a-i\infty}^{a+i\infty} e^{\alpha(u+1/2)} \frac{1}{\sqrt{\alpha+1}} d\alpha
\]

\[
= \frac{1}{2\pi i} \int_{a-i\infty}^{a+i\infty} e^{\alpha+is} (u+1/2) \frac{1}{\sqrt{\alpha+i+1}} d(a + is)
\]

\[
= \frac{e^{(u+1/2)}}{2\pi} \int_{-\infty}^{+\infty} e^{is(u+1/2)} \frac{1}{\sqrt{\alpha+i+1}} ds
\]

\[
\approx \frac{e^{(u+1/2)}}{2\pi} \int_{-b}^{+b} e^{is(u+1/2)} \frac{1}{\sqrt{\alpha+i+1}} ds
\]

(5.14)

where the last expression has a truncated integral over the interval \([-b, +b]\). By setting numerical values for \(a\) and \(b\), we can evaluate the above truncated integral using Mathematica built-in function Integrate.

Let \(a = 1\) so that all singularities are to the left of the contour. Then, the results are as follows

When \(b = 10\), the result is

\[
\frac{e^{-a/2}}{\sqrt{2\pi(2a+1)}} \times i \left\{ \text{erfi}\left(\sqrt{1-5i}\sqrt{1+2u}\right) - \text{erfi}\left(\sqrt{1+5i}\sqrt{1+2u}\right) \right\}
\]

(5.15)

When \(b = 100\), the result is

\[
\frac{e^{-a/2}}{\sqrt{2\pi(2a+1)}} \times i \left\{ \text{erfi}\left(\sqrt{1-50i}\sqrt{1+2u}\right) - \text{erfi}\left(\sqrt{1+50i}\sqrt{1+2u}\right) \right\}
\]

(5.16)

When \(b = 1000\), the result is

\[
\frac{e^{-a/2}}{\sqrt{2\pi(2a+1)}} \times i \left\{ \text{erfi}\left(\sqrt{1-500i}\sqrt{1+2u}\right) - \text{erfi}\left(\sqrt{1+500i}\sqrt{1+2u}\right) \right\}
\]

(5.17)

where \(\text{erfi}(z)\) is the imaginary error function defined by \(\text{erf}(iz)/i\), where \(\text{erf}(x)\) is the error function.

By observing the trend in the result, we can deduce that the p.d.f. of \(U\) is

\[
f_U(u) = \frac{e^{-a/2}}{\sqrt{2\pi(2a+1)}} \times i \left\{ \text{erfi}\left(\sqrt{1-bi/2}\sqrt{1+2u}\right) - \text{erfi}\left(\sqrt{1+bi/2}\sqrt{1+2u}\right) \right\}
\]

(5.18)

Next, assume

\[
A = i \left\{ \text{erfi}\left(\sqrt{1-bi/2}\sqrt{1+2u}\right) - \text{erfi}\left(\sqrt{1+bi/2}\sqrt{1+2u}\right) \right\}
\]

(5.19)
Let \( u \) take on a specific number as long as \( u \geq -1/2 \), say \( u = 0.3 \). Take the limit of \( A \) as \( b \to \infty \) by using Mathematica built-in function Limit. After numerous experiments of setting different values of \( u \), we find that for any value of \( u > -1/2 \), the limit is equal to two. For \( u = -1/2 \), the limit is just zero.

\[
\lim_{b \to \infty} A = 2, \quad \text{for } u > -1/2 \\
= 0, \quad \text{for } u = -1/2
\]  

(5.20)

Therefore, the p.d.f. of \( U \) can be written as

\[
f_U(u) = \begin{cases} 
2 \frac{e^{-u^{1/2}}}{\sqrt{2\pi(2u+1)}}, & u > -1/2 \\
0, & \text{elsewhere}
\end{cases}
\]  

(5.21)

which is exactly the same as the expression derived from the p.d.f. of \( \chi^2(1) \).

- **5.3.3. Computing the density \( f_U(u) \) by numerically inverting the two-sided Laplace transform**

The numerical Laplace transform inversion algorithms we have encountered so far are designed to invert one-sided Laplace transforms. Now we test them to find out whether they can be used to invert the two-sided Laplace transform (5.5). Experiments reveal that UniT and FT work only for part of the domain as shown in Table 5.2. The accuracy deteriorates as \( u \) decreases to 0.34 and numerical inversion produces ridiculous result for \( u < 0.34 \). When \( u = 0 \), numerical inversion encounters infinite expression 1/0 and the result is hence indeterminate. With the same parameters setting, UniT and FT compute the density to the same accuracy though there is a marginal difference in the precision of the results. This suggests UniT and FT are essentially the same.
Table 5.2. Density $f_U(u)$

<table>
<thead>
<tr>
<th>u</th>
<th>Reference</th>
<th>UnitT</th>
<th>FT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>m = 60, wp = 60</td>
<td>ED Accu Density</td>
<td>CPU</td>
</tr>
<tr>
<td>-0.49</td>
<td>98</td>
<td>5.585758034</td>
<td>56.6</td>
</tr>
<tr>
<td>-0.2</td>
<td>100</td>
<td>0.7630905788</td>
<td>56.3</td>
</tr>
<tr>
<td>0</td>
<td>100</td>
<td>0.4839414490</td>
<td>-</td>
</tr>
<tr>
<td>0.2</td>
<td>100</td>
<td>0.3348681147</td>
<td>53.1</td>
</tr>
<tr>
<td>0.31</td>
<td>100</td>
<td>0.278814301</td>
<td>38.0</td>
</tr>
<tr>
<td>0.32</td>
<td>100</td>
<td>0.2744079329</td>
<td>36.2</td>
</tr>
<tr>
<td>0.33</td>
<td>100</td>
<td>0.2700359573</td>
<td>34.0</td>
</tr>
<tr>
<td>0.34</td>
<td>100</td>
<td>0.2657529266</td>
<td>32.3</td>
</tr>
<tr>
<td>0.35</td>
<td>100</td>
<td>0.2615563638</td>
<td>32.8</td>
</tr>
<tr>
<td>0.36</td>
<td>100</td>
<td>0.2574438668</td>
<td>33.2</td>
</tr>
<tr>
<td>0.37</td>
<td>100</td>
<td>0.2534132029</td>
<td>33.6</td>
</tr>
<tr>
<td>0.38</td>
<td>100</td>
<td>0.2494621055</td>
<td>33.9</td>
</tr>
<tr>
<td>0.39</td>
<td>100</td>
<td>0.2455846888</td>
<td>34.3</td>
</tr>
<tr>
<td>0.5</td>
<td>100</td>
<td>0.2075537487</td>
<td>37.3</td>
</tr>
<tr>
<td>1.</td>
<td>99</td>
<td>0.1027868665</td>
<td>42.6</td>
</tr>
<tr>
<td>5.</td>
<td>99</td>
<td>0.0009831597001</td>
<td>45.7</td>
</tr>
<tr>
<td>10</td>
<td>98</td>
<td>4.79445146 \times 10^{-4}</td>
<td>44.4</td>
</tr>
</tbody>
</table>


wp: computing precision. Accu: accuracy measured by the number of significant digits. ED: effective number of digits of precision in the result. CPU: computation time in seconds.

The reference densities are computed by the p.d.f. of $U$ with wp = 100.

Just for curiosity, we shift the integration line in numerical inversion to the left although it is deemed unjustified. Interestingly, the interval of accuracy moves towards the left of the domain with negative shifting in Table 5.3. But frequency shifting cannot solve all problems. It is still unable to compute correct densities for points close to $-1/2$. Moreover, negative shifting increases accuracy at points of small values, say $u = 0.31$, but simultaneously decreases accuracy at points on the right, say $u = 5$. 
Table 5.3. Density $f_u(a)$ with frequency shifting

<table>
<thead>
<tr>
<th>$u$</th>
<th>Reference</th>
<th>Density</th>
<th>$M = 60$, wp = 60, Shifting = -20</th>
<th>$M = 60$, wp = 60, Shifting = -60</th>
<th>FT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>ED Accu Density CPU</td>
<td>ED Accu Density CPU</td>
<td></td>
</tr>
<tr>
<td>-0.49</td>
<td>98</td>
<td>5.585758034</td>
<td>56.6 0 $-4.9 \times 10^{-7}$ 0.02</td>
<td>56.9 0 $-3.3 \times 10^{-7}$ 0.03</td>
<td></td>
</tr>
<tr>
<td>-0.2</td>
<td>100</td>
<td>0.7630805878</td>
<td>56.3 0 $-2.5 \times 10^{-7}$ 0.02</td>
<td>56.4 0 $1.6 \times 10^{-7}$ 0.02</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>100</td>
<td>0.483944490</td>
<td>- - Indeterminate -</td>
<td>- - Indeterminate -</td>
<td></td>
</tr>
<tr>
<td>0.2</td>
<td>100</td>
<td>0.3348651147</td>
<td>53.1 0 $1.1877 \times 10^{-7}$ 0.02</td>
<td>52.8 0 $9.1642 \times 10^{-7}$ 0.02</td>
<td></td>
</tr>
<tr>
<td>0.31</td>
<td>100</td>
<td>0.288714301</td>
<td>51.1 0 $9.7065 \times 10^{-7}$ 0.02</td>
<td>51.2 12 $0.2780 \times 10^{-7}$ 0.02</td>
<td></td>
</tr>
<tr>
<td>0.32</td>
<td>100</td>
<td>0.244078329</td>
<td>50.4 12 $2.2596 \times 10^{-7}$ 0.02</td>
<td>50.9 12 $0.2744 \times 10^{-7}$ 0.02</td>
<td></td>
</tr>
<tr>
<td>0.33</td>
<td>100</td>
<td>0.2700359573</td>
<td>50.0 12 $2.7000 \times 10^{-7}$ 0.02</td>
<td>50.7 11 $0.2700 \times 10^{-7}$ 0.02</td>
<td></td>
</tr>
<tr>
<td>0.34</td>
<td>100</td>
<td>0.2655729266</td>
<td>51.5 9 $2.6572 \times 10^{-7}$ 0.02</td>
<td>53.2 9 $0.2675 \times 10^{-7}$ 0.02</td>
<td></td>
</tr>
<tr>
<td>0.35</td>
<td>100</td>
<td>0.2615563638</td>
<td>50.1 11 $2.6156 \times 10^{-7}$ 0.02</td>
<td>54.1 7 $0.2615 \times 10^{-7}$ 0.02</td>
<td></td>
</tr>
<tr>
<td>0.36</td>
<td>100</td>
<td>0.2574438668</td>
<td>50.6 13 $2.5744 \times 10^{-7}$ 0.02</td>
<td>54.8 6 $0.2574 \times 10^{-7}$ 0.02</td>
<td></td>
</tr>
<tr>
<td>0.37</td>
<td>100</td>
<td>0.2534123029</td>
<td>51.1 15 $2.5341 \times 10^{-7}$ 0.02</td>
<td>55.4 4 $0.2534 \times 10^{-7}$ 0.02</td>
<td></td>
</tr>
<tr>
<td>0.38</td>
<td>100</td>
<td>0.2494818355</td>
<td>51.8 11 $2.4948 \times 10^{-7}$ 0.02</td>
<td>55.9 4 $0.2495 \times 10^{-7}$ 0.02</td>
<td></td>
</tr>
<tr>
<td>0.39</td>
<td>100</td>
<td>0.2455884688</td>
<td>51.3 12 $2.4559 \times 10^{-7}$ 0.02</td>
<td>56.5 3 $0.2464 \times 10^{-7}$ 0.02</td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>100</td>
<td>0.207557487</td>
<td>55.8 30 $2.0755 \times 10^{-7}$ 0.02</td>
<td>57.2 0 $1.9 \times 10^{-7}$ $-7.4 \times 10^{-7}$ 0.03</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>99</td>
<td>0.1027886865</td>
<td>55.2 12 $0.1027 \times 10^{-7}$ 0.02</td>
<td>57.2 0 $8.4 \times 10^{-24}$ $-1.2 \times 10^{-23}$ 0.02</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>99</td>
<td>0.000983159001</td>
<td>57.9 0 $6.2 \times 10^{-7}$ 0.03</td>
<td>57.2 0 $8.2 \times 10^{-25}$ $-7.6 \times 10^{-25}$ 0.02</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>98</td>
<td>4.794445416×10^{-7}</td>
<td>57.7 0 $3.5 \times 10^{-7}$ 0.03</td>
<td>57.9 0 $7.4 \times 10^{-26}$ $5.8 \times 10^{-26}$ 0.02</td>
<td></td>
</tr>
</tbody>
</table>


wp: computing precision. Accu: accuracy measured by the number of significant digits. ED: effective number of digits of precision in the result. CPU: computation time in seconds.

The reference densities are computed by the p.d.f. of $U$ with wp = 100.

### 5.4. Computation of the joint density $f_{U,V}(u,v)$

Recall the Laplace transform (5.3) of $(U,V)$

$$\tilde{f}(\alpha, \beta) = E[\exp(-\alpha U - \beta V)] = \int_0^\infty \int_0^\infty e^{-\alpha u} e^{-\beta v} f_{U,V}(u,v) \, du \, dv$$

$$= e^{\alpha/2} \left(\cosh \sqrt{2 \beta} + \frac{\alpha}{\sqrt{2 \beta}} \sinh \sqrt{2 \beta} \right)^{-1/2}$$

(5.22)

### 5.4.1. Reducing the two-dimensional Laplace transform by applying Bromwich integral

The above Laplace transform is two-dimensional and have double integrals. The inner integral is a two-sided Laplace transform of $f_{U,V}(u,v)$ with respect to $u$, regarding $v$ as constant, and yields $\tilde{f}(\alpha,v)$. The outer integral is one-sided Laplace transform of $\tilde{f}(\alpha,v)$ with respect to $v$, regarding $\alpha$ as constant, and finally yields $\tilde{f}(\alpha,\beta)$. Since we have symbolically inverted the two-sided Laplace transform (5.5) of $U$ in *Mathematica*, it inspires us to invert the Laplace transform $\tilde{f}(\alpha,\beta)$ with respect to $\alpha$, regarding $\beta$ as constant, to get $f(u,\beta)$. In this way, we can reduce the dimension of the Laplace transform from two to one. Then, the numerical inversion algorithm of one-dimensional Laplace transform can be applied, and it makes computation much easier in comparison with numerical inversion of two-dimensional Laplace transform.
Applying the Bromwich integral to $\tilde{f}(\alpha, \beta)$ and setting $\alpha = a + i s$, we have

\[
\mathcal{L}^{-1}\{\tilde{f}(\alpha, \beta)\} = \hat{f}(u, \beta) = \frac{1}{2\pi i} \int_{a-i\infty}^{a+i\infty} e^{\alpha u} \ e^{\alpha/2} \left( \cosh \sqrt{2 \beta} + \frac{a}{\sqrt{2 \beta}} \sinh \sqrt{2 \beta} \right)^{-1/2} \, d\alpha
\]

\[
= \frac{1}{2\pi i} \int_{a-i\infty}^{a+i\infty} e^{(u+1/2) \alpha} \left( \cosh \sqrt{2 \beta} + \frac{a}{\sqrt{2 \beta}} \sinh \sqrt{2 \beta} \right)^{-1/2} \, d\alpha
\]

\[
= \frac{e^{(u+1/2)} e^{-i \pi u}}{2\pi} \int_{-\infty}^{+\infty} e^{i \pi u(s+1/2)} \left( \cosh \sqrt{2 \beta} + \frac{a+is}{\sqrt{2 \beta}} \sinh \sqrt{2 \beta} \right)^{-1/2} \, ds
\]

\[
\approx \frac{e^{(u+1/2)} e^{-i \pi u}}{2\pi} \int_{-b}^{+b} e^{i \pi u(s+1/2)} \left( \cosh \sqrt{2 \beta} + \frac{a+is}{\sqrt{2 \beta}} \sinh \sqrt{2 \beta} \right)^{-1/2} \, ds
\]

where the last expression has a truncated integral over the interval $[-b, +b]$. By setting numerical values for $a$ and $b$, we can evaluate the above truncated integral using Mathematica built-in function

Integrate[]. Let $a = 1$ so that all singularities are to the left of the contour. Then, the results are as follows

When $b = 10$, the result is

\[
\left( \frac{\beta}{2} \right)^{1/4} \sqrt{\frac{\cosh(\sqrt{2 \beta})}{(2u+1)\pi}} \exp \left( -\frac{1}{\sqrt{2}} \left( \sqrt{2u+1} \sqrt{\beta \cosh(\sqrt{2 \beta})} \right) \right) 
\]

\[
i \left\{ \text{erfi} \left( \frac{\beta}{2} \right)^{1/4} \sqrt{1+2u} \sqrt{2\text{Coth}(\sqrt{2\beta}) + (1 - 10i) \sqrt{2/\beta}} \right\} - 
\]

\[
erfi \left( \frac{\beta}{2} \right)^{1/4} \sqrt{1+2u} \sqrt{2\text{Coth}(\sqrt{2\beta}) + (1 + 10i) \sqrt{2/\beta}} \right\}
\]

When $b = 100$, the result is

\[
\left( \frac{\beta}{2} \right)^{1/4} \sqrt{\frac{\cosh(\sqrt{2 \beta})}{(2u+1)\pi}} \exp \left( -\frac{1}{\sqrt{2}} \left( \sqrt{2u+1} \sqrt{\beta \cosh(\sqrt{2 \beta})} \right) \right) 
\]

\[
i \left\{ \text{erfi} \left( \frac{\beta}{2} \right)^{1/4} \sqrt{1+2u} \sqrt{2\text{Coth}(\sqrt{2\beta}) + (1 - 100i) \sqrt{2/\beta}} \right\} - 
\]

\[
erfi \left( \frac{\beta}{2} \right)^{1/4} \sqrt{1+2u} \sqrt{2\text{Coth}(\sqrt{2\beta}) + (1 + 100i) \sqrt{2/\beta}} \right\}
\]

When $b = 1000$, the result is

\[
\left( \frac{\beta}{2} \right)^{1/4} \sqrt{\frac{\cosh(\sqrt{2 \beta})}{(2u+1)\pi}} \exp \left( -\frac{1}{\sqrt{2}} \left( \sqrt{2u+1} \sqrt{\beta \cosh(\sqrt{2 \beta})} \right) \right) 
\]

\[
i \left\{ \text{erfi} \left( \frac{\beta}{2} \right)^{1/4} \sqrt{1+2u} \sqrt{2\text{Coth}(\sqrt{2\beta}) + (1 - 1000i) \sqrt{2/\beta}} \right\} - 
\]

\[
erfi \left( \frac{\beta}{2} \right)^{1/4} \sqrt{1+2u} \sqrt{2\text{Coth}(\sqrt{2\beta}) + (1 + 1000i) \sqrt{2/\beta}} \right\}
\]
\[
\left(\frac{\beta}{2}\right)^{1/4} \sqrt{\frac{\cosh(\sqrt{2} \beta)}{(2u + 1) \pi}} \exp\left(-\frac{1}{\sqrt{2}} (2u + 1) \sqrt{\beta} \coth(\sqrt{2} \beta)\right) \times
\]
\[
i \left\{ \text{erfi}\left[\left(\frac{\beta}{8}\right)^{1/4} \sqrt{1 + 2u} \sqrt{2 \coth(\sqrt{2} \beta) + (1 - 1000) \sqrt{2} / \beta}\right] \right\} - \]
\[
\text{erfi}\left[\left(\frac{\beta}{8}\right)^{1/4} \sqrt{1 + 2u} \sqrt{2 \coth(\sqrt{2} \beta) + (1 + 1000) \sqrt{2} / \beta}\right]\right\}
\]

(5.26)

where \(\text{erfi}(z)\) is the imaginary error function defined by \(\text{erf}(i z) / i\), where \(\text{erf}(x)\) is the error function.

By observing the trend in the result, we can deduce that the function \(\hat{f}(u, \beta)\) is

\[
\hat{f}(u, \beta) = \left(\frac{\beta}{2}\right)^{1/4} \sqrt{-\frac{1}{\sqrt{2}} (2u + 1) \sqrt{\beta} \coth(\sqrt{2} \beta) \times
\]
\[
i \left\{ \text{erfi}\left[\left(\frac{\beta}{8}\right)^{1/4} \sqrt{1 + 2u} \sqrt{2 \coth(\sqrt{2} \beta) + (1 - b i) \sqrt{2} / \beta}\right] \right\} - \]
\[
\text{erfi}\left[\left(\frac{\beta}{8}\right)^{1/4} \sqrt{1 + 2u} \sqrt{2 \coth(\sqrt{2} \beta) + (1 + b i) \sqrt{2} / \beta}\right]\right\}
\]

(5.27)

Next, assume

\[
A = i \left\{ \text{erfi}\left[\left(\frac{\beta}{8}\right)^{1/4} \sqrt{1 + 2u} \sqrt{2 \coth(\sqrt{2} \beta) + (1 - b i) \sqrt{2} / \beta}\right] \right\} - \]
\[
\text{erfi}\left[\left(\frac{\beta}{8}\right)^{1/4} \sqrt{1 + 2u} \sqrt{2 \coth(\sqrt{2} \beta) + (1 + b i) \sqrt{2} / \beta}\right]\right\}
\]

(5.28)

Let \(u\) take on a specific number as long as \(u \geq -1/2\), say \(u = 0.3\), and \(\beta\) take on a specific complex number as long as \(\text{Re}(\beta) > 0\), say \(\beta = 2 + 3i\). Then, take the limit of \(A\) as \(b \to \infty\) by using Mathematica built-in function Limit. After numerous experiments of setting different values of \(u\) and \(\beta\), we find that for any value of \(u > -1/2\) and any value of \(\beta\) with \(\text{Re}(\beta) > 0\), the limit is equal to two. For \(u = -1/2\) and any value of \(\beta\) with \(\text{Re}(\beta) > 0\), the limit is just zero.

\[
\lim_{b \to \infty} A = 2, \quad \text{for } u > -1/2 \text{ and } \text{Re}(\beta) > 0
\]
\[
= 0, \quad \text{for } u = -1/2 \text{ and } \text{Re}(\beta) > 0
\]

(5.29)

Therefore, the function \(\hat{f}(u, \beta)\) can be written as
\[
\hat{f}(u, \beta) = \int_0^\infty e^{-\beta v} f_{U,V}(u, v) \, dv
\]
\[
= \begin{cases} 
2 \left( \frac{\beta}{2} \right)^{1/4} \left( \frac{\text{csch}(\sqrt{2} \beta)}{(2 u + 1) \pi} \right)^{1/2} e^{-\frac{1}{\sqrt{1}} (2 u + 1) \sqrt{\beta} \coth(\sqrt{2} \beta)}, & u > -1/2 \text{ and } \text{Re}(\beta) > 0 \\
0, & u = -1/2 \text{ and } \text{Re}(\beta) > 0
\end{cases}
\] (5.30)

This is a new result in the theory of unit root econometrics, and has been obtained, in the spirit of this thesis, through a judicious use of the Mathematica platform. But we also have to admit that the challenge to analytically prove the result remains. The expression of \(\hat{f}(u, \beta)\) is very useful because it is the Laplace transform of joint density \(f_{U,V}(u, v)\) with respect to \(v\). We can obtain the joint density \(f_{U,V}(u, v)\) by numerically inverting reduced Laplace transform (5.30).

**5.4.2. Computing the joint density \(f_{U,V}(u, v)\) by numerically inverting the reduced Laplace transform \(\hat{f}(u, \beta)\)**

Regarding numerical inversion, we have two choices to obtain the joint density \(f_{U,V}(u, v)\): one is to apply two-dimensional inversion algorithm to Laplace transform \(\tilde{f}(\alpha, \beta)\) in (5.3); another is to apply one-dimensional inversion algorithm to reduced Laplace transform \(\hat{f}(u, \beta)\) in (5.30). It is obvious that inversion of \(\hat{f}(u, \beta)\) takes less effort than inversion of \(\tilde{f}(\alpha, \beta)\). Subsequent experiments will reveal that inversion of \(\hat{f}(\alpha, \beta)\) involves problems in some region.

By testing UniE, UniT and UniG, we find the reduced Laplace transform \(\hat{f}(u, \beta)\) in (5.30) can be inverted by UniG with respect to \(\beta\). Regarding the inversion of \(\hat{f}(\alpha, \beta)\), it turns out that only UniTG inverts \(\tilde{f}(\alpha, \beta)\) successfully with the first operator \(T\) applied to the inner integral in the Laplace transform (5.3) and the second operator \(G\) applied to the outer integral. Note it is UniGT instead of UniTG that will work if we change both the order of input complex variables and the order of input real variables. But, using UniGT is not very useful because it causes computation time to soar.

Table 5.4 demonstrates the joint density \(f_{U,V}(u, v)\) obtained from UniG and UniTG where UniG applies to the reduced Laplace transform \(\hat{f}(u, \beta)\) in (5.30) and UniTG applies to the original Laplace transform \(\tilde{f}(\alpha, \beta)\) in (5.3). It is clear that inversion of the reduced Laplace transform \(\hat{f}(u, \beta)\) is very easy and accurate, taking less than 0.05 seconds and obtaining an accuracy of at least 12 decimal places. Note the accuracy shown in the table is measured by the number of significant digits. For example, the density
1.8278 \times 10^{-10} with 6-digit accuracy is actually accurate to 15 decimal places. In contrast, inversion of the original Laplace transform \( \hat{f}(\alpha, \beta) \) in (5.3) is problematic. UniTG does not work until \( u \) is greater than 0.34. Besides that, it encounters infinite expression 1/0 when \( u = 0 \).

| \( u \) | \( v \) | Reference | \( f_{U,V}(u, v) \) of \((U, V)\) |
|---|---|---|---|---|---|---|
| \( -0.49 \) | 0.01 | 31 | 0.02570053186 | 34.6 | 11 | 0.025701 | 0.03 | 38.1 | 0 | \(-4.3865 \times 10^{10}\) | 2.71 |
| \( -0.49 \) | 0.1 | 38 | 27.32930627 | 27.1 | 15 | 27.329 | 0.03 | 15.7 | 0 | \(-4.4634 \times 10^{10}\) | 2.73 |
| \( -0.49 \) | 5.5 | 34 | 1.409365024 | 23.8 | 16 | 1.4094 | 0.03 | 7.8 | 0 | \(-1.2997 \times 10^{13}\) | 2.68 |
| \( -0.49 \) | 1 | 34 | 0.08778256915 | 22.4 | 14 | 0.087785 | 0.03 | 35.3 | 0 | \(-1.7904 \times 10^{11}\) | 2.71 |
| \( -0.49 \) | 5 | 27 | 1.827764279 \times 10^{-10} | 13.8 | 6 | 1.8278 \times 10^{-10} | 0.03 | 26.5 | 0 | \(-937.38\) | 3.60 |
| \(-0.3\) | 0.01 | 27 | 3.472036737 \times 10^{-8} | 36.1 | 9 | 3.4720 \times 10^{-8} | 0.03 | 54.0 | 0 | 4.8959 \times 10^{10}\) | 2.70 |
| \(-0.3\) | 0.1 | 35 | 3.330349581 | 28.8 | 15 | 3.3303 | 0.03 | 42.2 | 0 | 3.2943 \times 10^{11}\) | 2.65 |
| \(-0.3\) | 5 | 33 | 0.6578639446 | 24.9 | 14 | 0.65786 | 0.05 | 73.8 | 0 | 1.5660 \times 10^{12}\) | 2.68 |
| \(-0.3\) | 5 | 26 | 1.5317167477 \times 10^{-1} | 15.9 | 7 | 1.5317 \times 10^{-1} | 0.03 | 26.4 | 0 | 1.0418 \times 10^{12}\) | 2.62 |
| 0 | 0 | 27 | 1.551263485 \times 10^{-8} | 36.1 | 10 | 1.5513 \times 10^{-8} | 0.03 | | | - | Indeterminate | 2.43 |
| 0 | 0.1 | 34 | 0.2921287672 | 31.2 | 14 | 0.29213 | 0.05 | | | - | Indeterminate | 2.15 |
| 0 | 0.5 | 34 | 0.5402627972 | 26.5 | 13 | 0.54026 | 0.03 | | | - | Indeterminate | 2.12 |
| 0 | 5 | 27 | 1.336754489 \times 10^{-1} | 17.9 | 9 | 1.3367 \times 10^{-1} | 0.03 | | | - | Indeterminate | 2.07 |
| 0.33 | 0.4 | 34 | 0.3167875392 | 28.5 | 15 | 0.316787 | 0.03 | 15.0 | 0 | 17.85 | 2.64 |
| 0.34 | 0.4 | 34 | 0.3278706219 | 28.6 | 15 | 0.32787 | 0.03 | 18.4 | 0 | 3.2978 | 2.78 |
| 1.61 | 29 | 1.501024389 \times 10^{-7} | 35.7 | 10 | 1.5010 \times 10^{-7} | 0.03 | 27.0 | 21 | 1.5010 \times 10^{-7} | 2.71 |
| 1.61 | 29 | 3.6401905304 | 28.2 | 14 | 0.088460 | 0.03 | 34.8 | 21 | 0.088460 | 2.64 |
| 5.67 | 28 | 2.747641087 \times 10^{-11} | 36.1 | 9 | 2.7476 \times 10^{-11} | 0.03 | 19.5 | 16 | 2.7476 \times 10^{-11} | 2.65 |
| 5 | 4 | 29 | 0.0003656503985 | 30.2 | 12 | 0.00036565 | 0.03 | 25.9 | 21 | 0.00036565 | 2.62 |


wp: computing precision. Accu: accuracy measured by the number of significant digits. ED: effective number of digits of precision in the result. CPU: computation time in seconds.

The reference densities are obtained by UniG with parameters settings: \( M = 100 \), wp = 200 versus \( M = 80 \), wp = 160.

Since we are confident of computing correct joint density \( f_{U,V}(u, v) \) using UniG with parameters setting of \( M = 35 \) and wp = 70, we can then plot the surface of \( f_{U,V}(u, v) \) as in Figure 5.3.
Observing the surface of $f_{U, V}(u, v)$ carefully, we notice that the cross-section of the surface has a serrated edge when $u$ is close to $-0.5$ and $v > 0.3$. To investigate what happens there. We draw the curve of the joint density $f_{U, V}(u, v)$ with $v$ fixed at $v = 0.7$ and $u$ varied between $-0.5$ and $-0.3$. Note how the curve declines first and then rises in Figure 5.4. Further experiments show this is not caused by round-off error. We, hence, believe it is the own feature of the joint density $f_{U, V}(u, v)$. 

Figure 5.3. Surface of the joint density $f_{U, V}(u, v)$

Figure 5.4. Curve of the joint density $f_{U, V}(u, v)$ with $v$ is fixed at $v = 0.7$. 

6. Joint Density of $R$ and $S$

Rao (1978) produced an analytical expression for the density of $S_1$ but it was intractable. It was Abadir’s work that opened up the area of finding usable, analytical formulae for the distribution and density functions of unit root statistics. In section 5, we compute the joint density $f_{U,V}(u, v)$ of $U$ and $V$ by inverting a reduced Laplace transform. The computation is proved to be effective and accurate. Abadir (1995b) proposes explicit formulae for the joint density $f_{R,S}(r, s)$ of $R$ and $S$. The formulae are integral-free but in the form of infinite series. The random variables $R$ and $S$ are connected with the random variables $U$ and $V$ via the relations $R = \sqrt{2} U$ and $S = 2 V$. This implies that the joint density $f_{R,S}(r, s)$ can not only be computed from the joint density $f_{U,V}(u, v)$, but also be used in the generation of distributions of unit root statistics written in terms of $U$ and $V$. We will examine those formulae for $f_{R,S}(r, s)$ and use $f_{U,V}(u, v)$ as a reference to check the accuracy and efficiency of $f_{R,S}(r, s)$.

- **6.1. Standardized quadratic forms**

All major unit root statistics have been shown to converge asymptotically to functionals of Brownian motion (Abadir, 1993; Phillips, 1987 cited in Abadir, 1995b). Moreover, Abadir (1995b) notices that almost all known unit root statistics can be represented in terms of the following two functionals:

$$
U = \frac{1}{\sqrt{2}} R = \int_0^1 w(u) \, dw(u) = \frac{1}{2} \left( u(1)^2 - 1 \right)
$$

$$
V = \frac{1}{2} S = \int_0^1 w(u)^2 \, du
$$

(6.1)

Following White (1958, 1959), Abadir (1995b) chooses a different normalization, i.e. $R$ and $S$, for the unit root statistics than in Hamilton’s (1994) and Tanaka’s (1996) textbooks. The first element of Fisher’s information matrix is given by (2.24) above:

$$
I_{11} = E[-H_{11}] = E\left[ \sum_{j=1}^{T} \frac{1}{\sigma^2} y_j^2 \right] = \frac{1}{2} T (T - 1)
$$

(6.2)

where $H$ is the Hessian of the log-likelihood function (i.e. the derivative of the score). This motivated Abadir to define the standardized quadratic forms which all unit root distributions are constructed in the form:

$$
R_T = \frac{\sum_{j=1}^{T} y_j^2 - \sum_{j=1}^{T} \frac{\sum_{l=1}^{j-1} y_l^2}{j^2}}{\frac{1}{\sigma^2}}
$$

$$
S_T = \frac{\sum_{j=1}^{T} \frac{y_j^2}{j^2}}{\left( \frac{1}{\sigma^2} \right)}
$$

(6.3)
where \( g = T / \sqrt{2} \) is the square root of the asymptotically dominant term of \( I_{11} \). As \( T \to \infty \), Abadir (1995b) shows these quadratic forms converge to

\[
R_T \xrightarrow{d} R = \sqrt{2} \, U \\
S_T \xrightarrow{d} S = 2 \, V
\]  

(6.4)

In terms of Tanaka’s \( S_3 \) and \( S_4 \) statistics, tests for a unit root in the basic AR(1) model are therefore based on

\[
\frac{R_T}{S_T} = \frac{S_T}{\sqrt{2}^2} = \frac{T(\hat{\rho}-1)}{\sqrt{2}}
\]  

(6.5)

and

\[
\frac{R_T}{S_T^{1/2}} = S_4T = \frac{T^{1/2}}{\hat{\sigma}} \sqrt{\sum_{j=2}^T \gamma_j^2}
\]  

(6.6)

\( R_T / S_T \) and \( R_T / \sqrt{S_T} \) are called the normalized autocorrelation coefficient and the Studentized \( t \) ratio in Abadir (1995a).

Consider the Laplace transform of \( (U, V) \) given in White (1958)

\[
\tilde{f}(\alpha, \beta) = E[\exp(-\alpha U - \beta V)] \\
= \int_0^\infty \int_0^\infty e^{-u\alpha - v\beta} f_{U,V}(u, v) \\ du \, dv \\
= e^{i\alpha^2/2} \left( \cosh 2\beta + \frac{\alpha}{\sqrt{2}\beta} \sinh 2\beta \right)^{-1/2}
\]  

(6.7)

Let \( U = R / \sqrt{2} \), \( V = S / 2 \), \( \theta_1 = \frac{-\alpha}{\sqrt{2}i} \), and \( \theta_2 = \frac{\beta}{2i} \). Then, we have \( R = \sqrt{2} \, U \), \( S = 2 \, V \), \( \alpha = -\sqrt{2} \, i \theta_1 \), and \( \beta = 2 \, i \theta_2 \). Substituting \( U, V, \alpha \) and \( \beta \) into (6.7)

\[
\tilde{f}(\theta_1, \theta_2) = E\left[ \exp \left( \sqrt{2} \, i \theta_1 \frac{R}{\sqrt{2}} - 2i \theta_2 \frac{S}{2} \right) \right] \\
= E[\exp(i \theta_1 R - i \theta_2 S)] \\
= \int_0^\infty \int_0^\infty e^{i \theta_1 r} e^{-i \theta_2 s} f_{R,S}(r, s) \, dr \, ds \\
= e^{-\sqrt{2} \, i \theta_1 \theta_2 \frac{1}{2}} \left[ \cosh 4i \theta_2 + \frac{\sqrt{2} i \theta_1}{\sqrt{4 i \theta_2}} \sinh 4i \theta_2 \right]^{-1/2}
\]  

(6.8)

(6.8) is the characteristic function of \( (R, S) \), and corrects the expression given in Abadir (1995a).

- 6.2. Formulae for joint density \( f_{R,S}(r, s) \)

Abadir (1995b) then inverts the limiting joint characteristic function of \( R \) and \( S \), which is readily available, to obtain analytical formulae for the joint density \( f_{R,S}(r, s) \) and joint distribution functions
The joint density \( f_{R,z}(r, s) \) has two forms

\[
f_{R,z}(r, s) = g_1(r, s) = g_2(r, s)
\]

with

\[
g_1(r, s) = \left( \frac{2}{\pi^{3/4} s^2} \right)^{1/4} \sum_{j=0}^{\infty} \left( \frac{j - 1/2}{j} \right) e^{-\left(\omega + b/2\right)^2/s} \sum_{k=0}^{\infty} \left( j + k - 1/2 \right) e^{-a^2/s} K\left(j + \frac{3}{2}, a \sqrt{2/s} \right)
\]

\[
g_2(r, s) = \left( \frac{2}{\pi^{3/4} s^2} \right)^{1/4} \sum_{j=0}^{\infty} \left( \frac{-b \sqrt{2/s} \sqrt{j}}{\sqrt{j+1}} \right) \sum_{k=0}^{\infty} \left( j + k - 1/2 \right) e^{-a^2/s} K\left(j + \frac{3}{2}, a \sqrt{2/s} \right)
\]

where \( b = 1 + \sqrt{2r} > 0, a = 2(j + k) + (b + 1)/2, \omega = 2j + 1/2, \binom{n}{m} \) is the binomial coefficient, and \( K(\nu, \zeta) = \exp(\zeta^2/4) D_\nu(\zeta) \) where \( D_\nu(\zeta) \) is the parabolic cylinder function (see Appendices A2).

The domain of \( f_{R,z}(r, s) \) is \([r, s] = -1/\sqrt{2} \leq r < \infty, 0 \leq s < \infty\). This is because \( R = \sqrt{2} U \) where \( U \) takes on any value of \( u \geq -1/2 \) and \( S = 2V \) where \( V \) is a non-negative random variable. Abadir (1995b) does not mention this domain at all. But we should pay particular attention to it. If we inadvertently assign a point outside the domain to \( f_{R,z}(r, s) \), the formulae will give a misleading result instead of zero. This is particularly true when we use \( f_{R,z}(r, s) \) to generate densities of unit root statistics where points outside the domain can be assigned to \( f_{R,z}(r, s) \) in the internal computations. In practice, we set \( f_{R,z}(r, s) \) to zero as \( r \leq -1/\sqrt{2}/2 \) or \( s \leq 0 \). The boundary points are included because the densities at this points go to infinity and setting them to zero does not make any difference.

The joint density \( f_{R,z}(r, s) \) can also be represented by the following expressions which involve operation of inverse Laplace transform. These expressions are intermediate steps in the derivation of the formulae (6.10) in Abadir (1995b). Using numerical inversion technique, we can then compute the joint density \( f_{R,z}(r, s) \) from these expressions. The only exception is that formula 13 cannot yields correct densities based on our numerical experiments. The expression inside the inverse Laplace transform operator appears to be incorrect.
\[ I_1 = f_{R,S}(r, s) = \sqrt{\frac{2}{\pi c}} \sum_{j=0}^{\infty} \left( j - \frac{1}{2} \right)_j \Gamma^{-1} \left( 2V^{1/4} e^{-\sqrt{V} \left( 2u + c \sqrt{V} \right)} I_1 \left( -j; 1/2; 2c \sqrt{2V} \right) \right) \]

\[ I_2 = f_{R,S}(r, s) = \sqrt{\frac{2}{\pi c}} \sum_{j=0}^{\infty} \left( j - \frac{1}{2} \right)_j \sum_{e=0}^{\infty} \left( -2c \sqrt{2V} \right)_j \Gamma^{-1} \left( e^{-\sqrt{V} \left( 2u + c \sqrt{V} \right)} (2V)^{1/4} \right) \]

\[ I_3 = f_{R,S}(r, s) = \frac{1}{4 \sqrt{\pi b}} \Gamma^{-1} \left( e^{-\frac{1}{2} b \sqrt{V} \mathrm{coth}(\sqrt{V})} \left[ \sqrt{V} \csch(\sqrt{V}) \right]^{1/2} \right) \]

Recall the reduced Laplace transform \( \hat{f}(u, \beta) \) of \((U, V)\) in (5.30), for \( u > -1/2 \) and \( \text{Re}(\beta) > 0 \),

\[ \hat{f}(u, \beta) = 2 \left( \frac{\beta}{2} \right)^{1/4} \sqrt{\frac{\cosh \left( \frac{\sqrt{2}\beta}{2} \right)}{(2u+1)\pi}} \exp \left( -\frac{1}{\sqrt{2}} (2u+1) \sqrt{\beta} \csch(\sqrt{2\beta}) \right) \]

\[ = \sqrt{2} (2\beta)^{1/4} \frac{1}{\sqrt{(2u+1)\pi}} e^{-\frac{1}{\sqrt{2}} (2u+1) \sqrt{\beta} \csch(\sqrt{2\beta})} \left[ \csch(\sqrt{2\beta}) \right]^{1/2} \]

\[ = \sqrt{2} \frac{1}{\sqrt{(2u+1)\pi}} e^{-\frac{1}{\sqrt{2}} (2u+1) \sqrt{\beta} \csch(\sqrt{2\beta})} \left[ \sqrt{2\beta} \csch(\sqrt{2\beta}) \right]^{1/2} \]  

(6.12)

We surprisingly find the Laplace transform in I3 and the Laplace transform of \( f_{U,V}(u, v) \) with respect to \( v \) are alike. Both expressions have similar structures and the same functions such as the hyperbolic cosecant and the hyperbolic cotangent. The fact that the Laplace transform I3 is, up to a constant and transformation, the same as the Laplace transform of \( f_{R,S}(r, s) \) offers very strong support for the validity of our expression.

The joint density \( f_{R,S}(r, s) \) and the joint density \( f_{U,V}(u, v) \) introduced earlier are related with each other. In fact, \( f_{R,S}(r, s) \) can be computed from \( f_{U,V}(u, v) \) with a transformation rule, and vice versa. The transformation formulae will be discussed later in Section 7.1. By now, we use \( f_{U,V}(u, v) \) to form the reference densities for \( f_{R,S}(r, s) \). Note the actual accuracy of a result can exceed the accuracy of the reference density. But when determine the accuracy of a result, the highest accuracy is limited by that of reference density.

The points to be used for \( f_{R,S}(r, s) \) are mapped from those used for \( f_{U,V}(u, v) \) in Table 5.4. So, a comparison can be made between the performance of \( f_{R,S}(r, s) \) and that of \( f_{U,V}(u, v) \) without the bias of the selection of points. Table 6.1 shows the accuracy and the speed of Abadir's (1995b) formulae \( g_1 \) and \( g_2 \). With the same parameters setting, the accuracy yielded is affected by the location of the point. More seriously, the computation time also varies according to the coordinates of the point. In some region, \( g_1 \)
gives almost exact result with small amount of time, about 0.05 seconds. But in some other region, the accuracy drops significantly to around 15 decimal places. The computation time can take as long as 0.5 seconds. The formula \( g_2 \) performs even worse than the formula \( g_1 \), taking 1.05 seconds at most.

In comparison, numerical inversion UniG for \( f_{U,V}(u, v) \) in Section 5.4 yields stable accuracy and takes similar amount of time, 0.03 seconds across all points. The joint density \( f_{U,V}(u, v) \) is far superior to the joint density \( f_{R,S}(r, s) \), not only in terms of the computation of itself, but also when we generate densities of unit root statistics where the joint density needs to be integrated.

Table 6.1. Joint density \( f_{R,S}(r, s) \) of \((R, S)\)

| \( x_1 \) | \( x_2 \) | Reference | g1 \( 10 = 8, \text{wp} = 40 \) | g2 \( 10 = 8, x_2 = 8, \text{wp} = 40 \) \\
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Accu</td>
<td>Density</td>
<td>CPU</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Accu</td>
<td>Density</td>
<td>CPU</td>
</tr>
<tr>
<td>-0.699</td>
<td>0.02</td>
<td>30</td>
<td>0.009045610119</td>
<td>0.0090865</td>
</tr>
<tr>
<td>-0.699</td>
<td>0.2</td>
<td>31</td>
<td>9.83945141</td>
<td>9.8374</td>
</tr>
<tr>
<td>-0.697</td>
<td>1.</td>
<td>35</td>
<td>0.492657829</td>
<td>0.498292</td>
</tr>
<tr>
<td>-0.920</td>
<td>0.2</td>
<td>34</td>
<td>0.0010367755</td>
<td>0.001037</td>
</tr>
<tr>
<td>-0.699</td>
<td>10.2</td>
<td>26</td>
<td>6.462122580 \times 10^{-2}</td>
<td>6.4620 \times 10^{-2}</td>
</tr>
<tr>
<td>-0.424</td>
<td>0.2</td>
<td>26</td>
<td>3.3489707 \times 10^{-4}</td>
<td>3.344999 \times 10^{-4}</td>
</tr>
<tr>
<td>-0.424</td>
<td>0.2</td>
<td>36</td>
<td>1.174261386</td>
<td>1.1777</td>
</tr>
<tr>
<td>-0.424</td>
<td>1.</td>
<td>34</td>
<td>0.323688149</td>
<td>0.32577</td>
</tr>
<tr>
<td>-0.424</td>
<td>10.2</td>
<td>27</td>
<td>1.248650335 \times 10^{-1}</td>
<td>1.2487 \times 10^{-1}</td>
</tr>
<tr>
<td>0.04</td>
<td>27</td>
<td>5.48464467 \times 10^{-10}</td>
<td>5.4845 \times 10^{-10}</td>
<td>0.05</td>
</tr>
<tr>
<td>0.04</td>
<td>10.2</td>
<td>36</td>
<td>0.1032831161</td>
<td>0.10322</td>
</tr>
<tr>
<td>0.1</td>
<td>35</td>
<td>0.190117438</td>
<td>0.19101</td>
<td>0.19101</td>
</tr>
<tr>
<td>0.2</td>
<td>10.2</td>
<td>1.674688670 \times 10^{-8}</td>
<td>1.6748 \times 10^{-8}</td>
<td>0.20</td>
</tr>
<tr>
<td>0.4660</td>
<td>0.2</td>
<td>34</td>
<td>0.449905829</td>
<td>0.45507</td>
</tr>
<tr>
<td>0.4807</td>
<td>0.2</td>
<td>34</td>
<td>0.1153197700</td>
<td>0.11592</td>
</tr>
<tr>
<td>1.4142</td>
<td>0.2</td>
<td>25</td>
<td>5.36092620 \times 10^{-10}</td>
<td>5.3609 \times 10^{-10}</td>
</tr>
<tr>
<td>1.4142</td>
<td>2.</td>
<td>33</td>
<td>0.02950762019</td>
<td>0.029508</td>
</tr>
<tr>
<td>0.0711</td>
<td>1.4</td>
<td>27</td>
<td>9.3147826 \times 10^{-8}</td>
<td>9.3144 \times 10^{-8}</td>
</tr>
<tr>
<td>0.0711</td>
<td>8.</td>
<td>36</td>
<td>0.001269368</td>
<td>0.001292</td>
</tr>
</tbody>
</table>

\( g_1 \): the first formula \( g_1(r, s) \) in (6.10). \( g_2 \): the second formula \( g_2(r, s) \) in (6.10).

wp: computing precision. Accu: accuracy measured by the number of significant digits. ED: effective number of digits of precision in the result. CPU: computation time in seconds.

The reference densities are obtained by transformed \( f_{U,V}(u, v) \) using UniG with parameters settings: \( M = 100, \text{wp} = 200 \) versus \( M = 80, \text{wp} = 160 \).

Since \( f_{R,S}(r, s) \) can be generated from \( f_{U,V}(u, v) \) and the efficiency of \( f_{U,V}(u, v) \) is better than that of \( f_{R,S}(r, s) \), we draw in Figure 6.1 the surface of \( f_{R,S}(r, s) \) using the transformed \( f_{U,V}(u, v) \) with \( M = 35 \) and \( \text{wp} = 70 \).
Recall that the cross-section of the surface of \( f_{U,V}(u, v) \) in Figure 5.3 has a serrated edge. We find a similar serrated edge in the cross-section of the surface of \( f_{R,S}(r, s) \). We purposely draw a graph of the joint density \( f_{R,S}(r, s) \) with \( s \) fixed at \( s = 1.4 \) but with \( r \) varied between \(-0.7\) and \(-0.4\). Then we see a familiar \( U \) shape just as \( f_{U,V}(u, v) \) with \( v \) fixed at \( v = 0.7 \).

**Figure 6.2.** Curve of the joint density \( f_{R,S}(r, s) \) with \( s \) is fixed at \( s = 1.4 \).

- **6.3. Formulae for joint distribution function \( F_{R,S}(r, s) \)**

Abadir (1995b) also gives formulae for the joint distribution function \( F_{R,S}(r, s) \) in five different forms

\[
F_{R,S}(r, s) = G_1(r, s) = G_2(r, s) = G_3(r, s) = G_4(r, s) = G_5(r, s)
\]  

(6.13)

with
\(G_i(r, s) = \left(\frac{s}{\pi h}\right)^{1/4} \sum_{j=0}^{\infty} \left(\frac{j - 1/2}{j}\right) e^{-s^{1/2} \frac{j}{1}} \sum_{\ell=0}^{\infty} \left(\frac{j - \frac{1}{2}}{\ell}\right) \sum_{k=0}^{\infty} \left(\frac{\xi}{\ell}(\ell + 1/2)\right) K\left(k + \frac{1}{2}, \sqrt{2s/\ell}\right)\)

\(G_2(r, s) = \left(\frac{s}{\pi h}\right)^{1/4} \sum_{j=0}^{\infty} \left(\frac{j - 1/2}{j}\right) \sum_{\ell=0}^{s^{1/2} \frac{j}{1}} \sum_{k=0}^{\infty} \left(\frac{\xi}{\ell}(\ell + 1/2)\right) K\left(k + \frac{1}{2}, \sqrt{2s/\ell}\right)\)

\(G_3(r, s) = \left(\frac{s}{\pi h}\right)^{1/4} \sum_{j=0}^{\infty} \left(\frac{j - 1/2}{j}\right) \sum_{\ell=0}^{s^{1/2} \frac{j}{1}} \sum_{k=0}^{\infty} \left(\frac{\xi}{\ell}(\ell + 1/2)\right) K\left(k + \frac{1}{2}, \sqrt{2s/\ell}\right)\)

\(G_4(r, s) = \left(\frac{s}{\pi h}\right)^{1/4} \sum_{j=0}^{\infty} \left(\frac{j - 1/2}{j}\right) \sum_{\ell=0}^{s^{1/2} \frac{j}{1}} \sum_{k=0}^{\infty} \left(\frac{\xi}{\ell}(\ell + 1/2)\right) K\left(k + \frac{1}{2}, \sqrt{2s/\ell}\right)\)

where \(G_a\) is an asymptotic series as \(b / \sqrt{s} \to \infty\). \(F(s)\) is the marginal CDF of \(S\) given by

\[F(s) = \sqrt{s} \sum_{j=0}^{\infty} \left(\frac{-1/2}{j}\right) \Phi\left(-\omega \sqrt{2/s}\right)\]

where \(\Phi(x)\) is the CDF of the standard normal distribution.
7. Generating Densities of Unit Root Statistics

So far, the joint density \( f_{U,V}(u,v) \) and the joint density \( f_{R,S}(r,s) \) are readily available. They are useful because we can use them to generate densities of unit root statistics such as \( S_3 \) and \( S_4 \). In this section, formulae are proposed for computing such densities, and numerical experiments are conducted as well.

- 7.1. Transformation between \( f_{U,V}(u,v) \) and \( f_{R,S}(r,s) \)

As we mentioned in Section 6.2, there is a link between \( f_{U,V}(u,v) \) and \( f_{R,S}(r,s) \). They can be transformed from each other. Recall that

\[
\begin{align*}
R &= \sqrt{2} \left( V \right) \\
S &= 2 \left( V \right)
\end{align*}
\]

where

\[
\begin{align*}
U &= \int_0^1 w(t) \, dw(t) = \frac{1}{2} \left[ w^2(1) - 1 \right] \\
V &= \int_0^1 w^2(t) \, dt
\end{align*}
\]

Following Hogg and Craig (1995, pp.170-173), the joint p.d.f. of \( (R, S) \) can be found from the joint p.d.f. of \( (U, V) \).

Let

\[
\begin{align*}
r &= h_1(u,v) = \sqrt{2} \, u \\
s &= h_2(u,v) = 2 \, v
\end{align*}
\]

define a one-to-one transformation that maps the set \( \mathcal{A} = \{(u,v): -1/2 \leq u < \infty, 0 < v < \infty\} \) onto the set \( \mathcal{B} = \{(r,s): -\sqrt{2} / 2 \leq r < \infty, 0 < s < \infty\} \). The inverse of \( r = h_1(u,v) \) and \( s = h_2(u,v) \) can be written as

\[
\begin{align*}
u &= g_1(r,s) = r / \sqrt{2} \\
v &= g_2(r,s) = s / 2
\end{align*}
\]

The Jacobian of the transformation denoted by \( J \) is given by

\[
J = \begin{bmatrix} \frac{\partial r}{\partial u} & \frac{\partial r}{\partial v} \\ \frac{\partial s}{\partial u} & \frac{\partial s}{\partial v} \end{bmatrix} = \begin{bmatrix} 1 / \sqrt{2} & 0 \\ 0 & 1 / 2 \end{bmatrix}
\]

\[
= \frac{1}{2 \sqrt{2}}
\]
Finally, the joint p.d.f. of \((R, S)\) can be computed from the joint p.d.f. of \((U, V)\) via

\[
f_{R,S}(r, s) = f_{U,V}(g_1(r, s), g_2(r, s)) |J|
\]

\[
= \frac{1}{2\sqrt{2}} f_{U,V}\left(\frac{r}{\sqrt{2}}, \frac{s}{\sqrt{2}}\right)
\]

(7.6)

Similarly, the joint p.d.f. of \((U, V)\) can be computed from the joint p.d.f. of \((R, S)\) via

\[
f_{U,V}(u, v) = 2\sqrt{2} f_{R,S}(\sqrt{2} u, 2 v)
\]

(7.7)

Now, we can compute the joint density \(f_{U,V}(u, v)\) from Abadir’s (1995b) formulae proposed for the joint density \(f_{R,S}(r, s)\). Correspondingly, the joint density \(f_{R,S}(r, s)\) can also be obtained from numerical inversion of the reduced Laplace transform (5.30). Table 7.1 compares the numerical inversion and Abadir’s formulae in terms of accuracy and efficiency by using them to compute the joint density \(f_{U,V}(u, v)\). Numerical inversion produces a steady accuracy of about 13 decimal places within 0.03 seconds for all cases. However, Abadir’s formula \(g_1\) in (6.10) takes varying computation time for different points although the accuracy is very high at some points. Since Abadir’s formulae are not always accurate and generally slow, they are inferior to numerical inversion.

**Table 7.1. Joint density \(f_{U,V}(u, v)\)**

<table>
<thead>
<tr>
<th>(u)</th>
<th>(v)</th>
<th>Reference</th>
<th>Accu Density</th>
<th>Numerical Inversion UniG, R = 35, wp = 70</th>
<th>Accu Density</th>
<th>Abadir’s Formulae (g_1), J0 = 8, wp = 40</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.47</td>
<td>0.01</td>
<td>31</td>
<td>0.0257003196</td>
<td>36.4 (s)</td>
<td>0.025701 (s)</td>
<td>37.8</td>
</tr>
<tr>
<td>0.495</td>
<td>0.1</td>
<td>38</td>
<td>27.25850627</td>
<td>27.1 (s)</td>
<td>0.250408 (s)</td>
<td>0.39 (s)</td>
</tr>
<tr>
<td>0.495</td>
<td>0.35</td>
<td>35</td>
<td>1.4093650424</td>
<td>23.9 (s)</td>
<td>1.40944 (s)</td>
<td>0.381</td>
</tr>
<tr>
<td>0.495</td>
<td>5</td>
<td>32</td>
<td>0.0077854905</td>
<td>32.1 (s)</td>
<td>0.007785 (s)</td>
<td>0.030</td>
</tr>
<tr>
<td>0.495</td>
<td>5</td>
<td>26</td>
<td>1.82776279 \times 10^{-5}</td>
<td>13.6 (s)</td>
<td>1.84278 (s)</td>
<td>0.031</td>
</tr>
<tr>
<td>0.3</td>
<td>0.01</td>
<td>27</td>
<td>8.47203673 \times 10^{-5}</td>
<td>16.1 (s)</td>
<td>8.47200 (s)</td>
<td>0.031</td>
</tr>
<tr>
<td>0.3</td>
<td>0.1</td>
<td>36</td>
<td>3.33034981</td>
<td>28.8 (s)</td>
<td>3.33039 (s)</td>
<td>0.032</td>
</tr>
<tr>
<td>0.3</td>
<td>0.5</td>
<td>33</td>
<td>0.6578039446</td>
<td>24.9 (s)</td>
<td>0.65780 (s)</td>
<td>0.033</td>
</tr>
<tr>
<td>0.3</td>
<td>5</td>
<td>27</td>
<td>3.52176472 \times 10^{-3}</td>
<td>35.9 (s)</td>
<td>3.52176 (s)</td>
<td>0.034</td>
</tr>
</tbody>
</table>

wp: computing precision. Accu: accuracy measured by the number of significant digits. ED: effective number of digits of precision in the result. CPU: computing time in seconds.

The reference density is obtained by digits matching with the following option specifications for our code JointDensityUV[ ]:

- (Method -> "UniG", "M" = 100), WorkingPrecision = 200) vs
- (Method -> "UniG", "M" = 80), WorkingPrecision = 160).
7.2. Generating the density of $S_3$ from $f_{U,V}(u, v)$

Recall

$$S_3 = \frac{U}{V} \quad (7.8)$$

Following Hogg and Craig (1995, pp. 170-175), we can find the density of $S_3$ from the joint density $f_{U,V}(u, v)$.

Set

$$X = \frac{U}{V}$$
$$Y = V \quad (7.9)$$

and let

$$x = u / v$$
$$y = v \quad (7.10)$$

define a one-to-one transformation that maps the set $\mathcal{A} = \{(u, v) : -1/2 \leq u < \infty, 0 < v < \infty\}$ onto the set $\mathcal{B} = \{(x, y) : -\infty < x < \infty, 0 < y < \infty\}$. The inverse of $x$ and $y$ can be written as

$$u = xy$$
$$v = y \quad (7.11)$$

The Jacobian of the transformation $J$ defined by the determinant of order 2 is

$$J = \begin{vmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} \end{vmatrix}$$
$$= \begin{vmatrix} y & x \\ 0 & 1 \end{vmatrix}$$
$$= y \neq 0 \quad (7.12)$$

Then, the p.d.f. of $(X, Y)$ is given by, for $(x, y) \in \mathcal{B},$

$$f_{X,Y}(x, y) = f_{U,V}(u, v) |J|$$
$$= f_{U,V}(xy, y) y$$
$$= y f_{U,V}(xy, y) \quad (7.13)$$

Finally, the p.d.f. of $S_3$ can be obtained from $f_{X,Y}(x, y)$ by integrating $y$ out.

$$f_{S_3}(x) = \int_{0}^{\infty} f_{X,Y}(x, y) \, dy$$
$$= \int_{0}^{\infty} y f_{U,V}(xy, y) \, dy \quad (7.14)$$

where the integrand is then equal to $y f_{U,V}(xy, y)$. Note the domain of $f_{U,V}(xy, y)$ is
[(x, y) : \(-1/2 \leq xy < \infty\), \(0 < y < \infty\)].

When \(x > 0\), the point \((x, y)\) is always inside the domain. Given a fixed \(x > 0\), the integrand \(y f_{U, V}(x, y)\) in (7.14) diminishes as \(y \to \infty\).

When \(x < 0\), the value of \(xy\) can be smaller than \(-1/2\) as \(y\) goes from zero to infinity. Thus, the first argument may be outside the domain. In this case, we should set \(f_{U, V}(x, y)\) equal to zero

\[
 f_{U, V}(x, y) = 0 \text{ for } y \geq -\frac{1}{2x} \tag{7.15}
\]

where \(x < 0\) and \(f_{U, V}(x, y) = 0\) at \(y = -1/(2x)\) due to the reduced Laplace transform \(\hat{f}(u, \beta) = 0\) at \(u = -1/2\) in (5.30). This setting is vital for the integral (7.14) to yield the correct result since \(f_{U, V}(x, y)\) is not automatically equal to zero for arguments outside the domain by numerical inversion in Section 5.4.2. Moreover, it helps reduce the computation time.

The integral (7.14) can be evaluated numerically by built-in Mathematica function NIntegrate[]. Two alternative methods for numerical integration are Simpson’s rule and the trapezoidal rule discussed in Section 4.3. But NIntegrate[] turns out to be more effective and adaptable. In practice, we need to truncate the integration range to make the computation easier and more efficient as the integrand becomes negligible before or after some certain point. Denote the lower and upper limit of integration by \(s_1\) and \(s_2\) respectively. The integral (7.14) is then truncated as follows

\[
 f_{S_1}(x) = \int_{S_1}^{S_2} y f_{U, V}(x, y) \, dy \tag{7.16}
\]

where the truncation size is \(S_2 - S_1\). The settings for \(S_1\) and \(S_2\) are obtained through trail and error to make the values of the integrand \(y f_{U, V}(x, y)\) at \(y = S_1\) and \(y = S_2\) sufficiently small but not excessively small. Thus, we can find a suitable truncation size for the numerical integration.

The following setting is suggested for \(S_1\) such that the integrand \(y f_{U, V}(x, y)\) at \(y = S_1\) is between \(10^{-29}\) and \(10^{-22}\) for \(x \in [-8, 5]\).

\[
\begin{align*}
\text{As } x \leq -8, & \quad S_1 = 0 \\
\text{Otherwise, } & \quad S_1 = 0.008
\end{align*} \tag{7.17}
\]

The upper limit \(S_2\) is set such that \(S_2\) is the boundary point of the domain if \(x \leq -1/14\) or the integrand \(y f_{U, V}(x, y)\) at \(y = S_2\) is between \(10^{-17}\) and \(5 \times 10^{-10}\) for \(x \in (-1/14, 5]\).
As \( x \leq -\frac{1}{14}, \) set \( s_2 = -\frac{1}{2x} \)
As \( x \leq 0, \) set \( s_2 = 7 \)
As \( x \leq 0.2, \) set \( s_2 = 10 \)
As \( x \leq 2, \) set \( s_2 = 17 \)
As \( x \leq 2.5, \) set \( s_2 = 10 \)
As \( x \leq 3, \) set \( s_2 = 7 \)
As \( x \leq 4, \) set \( s_2 = 4.5 \)
Otherwise, set \( s_2 = 2.5 \)

Figure 7.1 depicts the curves of the integrand \( y f_{U,V}(x,y) \) for different values of \( x. \) From the graphs, we can see it is reasonable to truncate the integration range because the integrand becomes marginal beyond a certain point.

Table 7.2 shows the density of \( S_J \) computed from the joint density \( f_{U,V}(u,v) \) using the formula (7.14). Mathematica built-in function NIntegrate[ ] is used to evaluate the integral numerically with option specifications: WorkingPrecision \( \to 20 \) and MaxRecursion \( \to 12. \) At each point \( x, \) two results are computed using two different parameter settings for numerical inversion for \( f_{U,V}(u,v) \). These two results are compared by digits matching to find the accuracy and the density composed of matching digits. Note the resulting accuracy is determined by the most inaccurate result. In Table 7.2, the figures in the column labeled ‘Accu’ reflect the accuracy of using UniG with \( M = 35 \) and 70-digit working precision. The resulting density can be used as a reference to check the accuracy of other methods.
Table 7.2. Accuracy of using $f_{i,j}(u, v)$ to compute the density of $S_3$

<table>
<thead>
<tr>
<th>x</th>
<th>NIntegrate, wp = 20</th>
<th>NIntegrate, wp = 20</th>
<th>Digits Matching</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>UniG, $M = 40$, wp = 80</td>
<td>UniG, $M = 35$, wp = 70</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ED Result CPU</td>
<td>ED Result CPU</td>
<td>Accu</td>
</tr>
<tr>
<td>-8</td>
<td>20.0 0.014814 13.6</td>
<td>20.0 0.014814 10.5</td>
<td>15</td>
</tr>
<tr>
<td>-7</td>
<td>20.0 0.020191 13.7</td>
<td>20.0 0.020191 10.9</td>
<td>15</td>
</tr>
<tr>
<td>-6</td>
<td>20.0 0.027741 13.8</td>
<td>20.0 0.027741 10.9</td>
<td>16</td>
</tr>
<tr>
<td>-5</td>
<td>20.0 0.038515 13.9</td>
<td>20.0 0.038515 10.7</td>
<td>16</td>
</tr>
<tr>
<td>-4</td>
<td>20.0 0.054233 14.0</td>
<td>20.0 0.054233 10.7</td>
<td>16</td>
</tr>
<tr>
<td>-3</td>
<td>20.0 0.077895 14.1</td>
<td>20.0 0.077895 10.6</td>
<td>16</td>
</tr>
<tr>
<td>-2</td>
<td>20.0 0.11512 14.2</td>
<td>20.0 0.11512 10.5</td>
<td>18</td>
</tr>
<tr>
<td>-1</td>
<td>20.0 0.17495 14.3</td>
<td>20.0 0.17495 12.5</td>
<td>17</td>
</tr>
<tr>
<td>0</td>
<td>20.0 0.20975 14.4</td>
<td>20.0 0.20975 23.6</td>
<td>16</td>
</tr>
<tr>
<td>0.1</td>
<td>20.0 0.23364 14.5</td>
<td>20.0 0.23364 13.2</td>
<td>16</td>
</tr>
<tr>
<td>0.2</td>
<td>20.0 0.24197 14.6</td>
<td>20.0 0.24197 17.9</td>
<td>14</td>
</tr>
<tr>
<td>0.3</td>
<td>20.0 0.24727 14.7</td>
<td>20.0 0.24727 13.2</td>
<td>13</td>
</tr>
<tr>
<td>0.5</td>
<td>20.0 0.25217 14.9</td>
<td>20.0 0.25217 17.6</td>
<td>10</td>
</tr>
<tr>
<td>0.7</td>
<td>20.0 0.24487 15.1</td>
<td>20.0 0.24487 17.4</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>20.0 0.16276 15.3</td>
<td>20.0 0.16276 18.2</td>
<td>11</td>
</tr>
<tr>
<td>1.3</td>
<td>20.0 0.10037 15.5</td>
<td>20.0 0.10037 17.7</td>
<td>12</td>
</tr>
<tr>
<td>1.6</td>
<td>20.0 0.055462 15.7</td>
<td>20.0 0.055462 18.3</td>
<td>11</td>
</tr>
<tr>
<td>2</td>
<td>20.0 0.023462 15.9</td>
<td>20.0 0.023462 17.3</td>
<td>12</td>
</tr>
<tr>
<td>2.5</td>
<td>20.0 0.0078333 16.0</td>
<td>20.0 0.0078333 19.8</td>
<td>13</td>
</tr>
</tbody>
</table>

NIntegrate: Compute the density using the formula (7.14) and evaluate the integral with Mathematica built-in function NIntegrate[].

UniG: Compute the joint density $f_{i,j}(u, v)$ using the numerical inversion algorithm UniG.

wp: computing precision. Accu: accuracy measured by the number of significant digits. ED: effective number of digits of precision in the result. CPU: computing time in seconds.

The accuracy and density is obtained by digits matching with the following option specifications for our code DensityS3[]:

(Method -> {"NIntegrate", WorkingPrecision -> 20}, 
Method -> {"NIntegrate", WorkingPrecision -> 20, 

The computation of the density at a single point takes more than 10 seconds. Thus, it is computationally expensive to plot the graph of the density directly from the formula (7.14). We use those densities listed in Table 7.2 as data points and join them by applying quadratic spline interpolation. Then, Figure 7.2 plots the graph of the density of $S_3$, which looks the same as the graph in Figure 7.3 plotted using Imhof’s formula (4.25).
Since we have two different methods to compute the density of $S_3$, we can check the accuracy of Imhof’s formula by comparing the result with the one generated from $f_{U,T}(u, v)$. From Table 7.3, it is observed that Imhof’s formula is accurate on the tails but the accuracy drops significantly in the vicinity of $x = 0$ due to slow convergence.
Table 7.3. Accuracy of using Imhof’s formula to compute the density of $S_3$

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>NIntegrate, wp = 20</th>
<th>Imhof’s formula, s = 30, wp = 25</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ED</td>
<td>Result</td>
</tr>
<tr>
<td>-5</td>
<td>20.0</td>
<td>0.014814</td>
</tr>
<tr>
<td>-7</td>
<td>20.0</td>
<td>0.021911</td>
</tr>
<tr>
<td>-6</td>
<td>20.0</td>
<td>0.027741</td>
</tr>
<tr>
<td>-5</td>
<td>20.0</td>
<td>0.038515</td>
</tr>
<tr>
<td>-4</td>
<td>20.0</td>
<td>0.054233</td>
</tr>
<tr>
<td>-3</td>
<td>20.0</td>
<td>0.078895</td>
</tr>
<tr>
<td>-2</td>
<td>20.0</td>
<td>0.115122</td>
</tr>
<tr>
<td>-1</td>
<td>20.0</td>
<td>0.174955</td>
</tr>
<tr>
<td>-0.5</td>
<td>20.0</td>
<td>0.209758</td>
</tr>
<tr>
<td>0.1</td>
<td>20.0</td>
<td>0.23664</td>
</tr>
<tr>
<td>0.2</td>
<td>20.0</td>
<td>0.24917</td>
</tr>
<tr>
<td>0.3</td>
<td>20.0</td>
<td>0.25217</td>
</tr>
<tr>
<td>0.5</td>
<td>20.0</td>
<td>0.24487</td>
</tr>
<tr>
<td>0.7</td>
<td>20.0</td>
<td>0.22166</td>
</tr>
<tr>
<td>1.0</td>
<td>20.0</td>
<td>0.16276</td>
</tr>
<tr>
<td>1.3</td>
<td>20.0</td>
<td>0.10037</td>
</tr>
<tr>
<td>1.5</td>
<td>20.0</td>
<td>0.055462</td>
</tr>
<tr>
<td>2.0</td>
<td>20.0</td>
<td>0.023462</td>
</tr>
<tr>
<td>2.5</td>
<td>20.0</td>
<td>0.0078333</td>
</tr>
</tbody>
</table>

NIntegrate (in the first row): compute the density using the formula (7.14) and evaluate the integral with Mathematica built-in function NIntegrate[]. UniG: Compute the joint density $f_{U_3}(x, s)$ using the numerical inversion algorithm UniG. NIntegrate (in the second row): evaluate Imhof’s formula (4.25) by Mathematica built-in function NIntegrate[] with truncation size $s$.

wp: computing precision. Accu: accuracy measured by the number of significant digits. ED: effective number of digits of precision in the result. CPU: computing time in seconds.

The accuracy and the density are obtained by digits matching with the following option specifications for our code DensityS3[]:

(Method -> ("NIntegrate", WorkingPrecision -> 20, MaxRecursion -> 12, "UniG", "M" -> 35), vs WorkingPrecision -> 70)


### 7.3. Generating the density of $S_3$ from $f_{R, S}(r, s)$

$S_3$ can also be expressed in terms of $R$ and $S$

$$S_3 = \frac{U}{V} = \frac{R \sqrt{S}}{S^{2/2}} = \sqrt{2} \frac{R}{S} \quad (7.19)$$

Following Hogg and Craig (1995, pp.170-175), the density of $S_3$ can be computed from the joint density $f_{R, S}(r, s)$.

Set

$$X = \sqrt{2} \frac{R}{S} \quad (7.20)$$

and let

$$x = \sqrt{2} \frac{r}{s} \quad y = s$$
define a one-to-one transformation that maps the set \( \mathcal{A} = \{(r, s) : -\sqrt{2} < r < \infty, 0 < s < \infty\} \) onto the set \( \mathcal{B} = \{(x, y) : -\infty < x < \infty, 0 < y < \infty\} \).

The inverse of \( x \) and \( y \) can be written as
\[
\begin{align*}
r &= \frac{1}{\sqrt{2}} x y \\
s &= y
\end{align*}
\]  
(7.21)

The Jacobian of the transformation \( J \) is calculated as
\[
J = \begin{vmatrix}
\frac{\partial x}{\partial r} & \frac{\partial x}{\partial s} \\
\frac{\partial y}{\partial r} & \frac{\partial y}{\partial s}
\end{vmatrix}
= \begin{vmatrix}
\frac{y}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\
0 & 1
\end{vmatrix}
= \frac{y}{\sqrt{2}} \neq 0
\]  
(7.22)

Then, the p.d.f. of \( (X, Y) \) is given by, for \( (x, y) \in \mathcal{B} \),
\[
\begin{align*}
f_{X,Y}(x, y) &= f_{R,S}(r, s) |J| \\
&= f_{R,S}(\frac{1}{\sqrt{2}} x y, y) \frac{y}{\sqrt{2}} \\
&= \frac{1}{\sqrt{2}} y f_{R,S}(\frac{1}{\sqrt{2}} x y, y)
\end{align*}
\]  
(7.23)

Finally, the p.d.f. of \( S_3 \) can be obtained from \( f_{X,Y}(x, y) \) by integrating \( y \) out.
\[
\begin{align*}
f_{S_3}(x) &= \int_0^\infty f_{X,Y}(x, y) \, dy \\
&= \int_0^\infty \frac{1}{\sqrt{2}} y f_{R,S}(\frac{1}{\sqrt{2}} x y, y) \, dy
\end{align*}
\]  
(7.24)

where the domain of \( f_{R,S}(\frac{1}{\sqrt{2}} x y, y) \) is \( ((x, y) : -1 \leq x y < \infty, 0 < y < \infty) \).

The integral (7.24) needs to be truncated in the actual evaluation given that Adadir’s formula \( g_1 \) is used to compute \( f_{R,S}(r, s) \).
\[
\begin{align*}
f_{S_3}(x) &= \int_{g_1}^{\infty} \frac{1}{\sqrt{2}} y f_{R,S}(\frac{1}{\sqrt{2}} x y, y) \, dy
\end{align*}
\]  
(7.25)

where
As \( x \leq -8 \), \( s_1 = 0 \)
Otherwise, \( s_1 = 0.016 \) \( \text{(7.26)} \)

so that the integrand \( \frac{1}{\sqrt{2}} y f_{R,S} \left( \frac{1}{\sqrt{2}} x y, y \right) \) at \( y = s_1 \) is between \( 10^{-28} \) and \( 5 \times 10^{-23} \) for \( x \in [-8, 5] \), and

As \( x \leq -\frac{1}{13} \), set \( s_2 = -\frac{1}{x} \)
As \( x \leq 0.2 \), set \( s_2 = 13 \)
As \( x \leq 0.45 \), set \( s_2 = 17 \)
As \( x \leq 0.7 \), set \( s_2 = 20 \)
As \( x \leq 2 \), set \( s_2 = 25 \)
As \( x \leq 3 \), set \( s_2 = 17 \)
As \( x \leq 4 \), set \( s_2 = 8 \)
Otherwise, set \( s_2 = 4.5 \) \( \text{(7.27)} \)

so that \( s_2 \) is the boundary point of the domain when \( x \leq -\frac{1}{13} \), or the integrand \( \frac{1}{\sqrt{2}} y f_{R,S} \left( \frac{1}{\sqrt{2}} x y, y \right) \) at \( y = s_2 \) is between \( 10^{-19} \) and \( 5 \times 10^{-7} \) for \( x \in (-1/13, 5] \).

Note the integrand is not monotonic decreasing as \( y \) increases. The value of the integrand decreases to a point, and then increases slightly before it drops below zero as shown in Figure 7.4. This implies that Abadir’s formula \( g_1 \) is of poor quality in some region of the domain. Therefore, truncation of the integration range is necessary for obtaining the correct density.

![Figure 7.4. Distortion of the integrand \( \frac{1}{\sqrt{2}} y f_{R,S} \left( \frac{1}{\sqrt{2}} x y, y \right) \) using Adadir’s formula \( g_1 \)](image)

Figure 7.4 draws the integrand \( \frac{1}{\sqrt{2}} y f_{R,S} \left( \frac{1}{\sqrt{2}} x y, y \right) \) at the same \( x \)'s as in Figure 7.1. By comparison with the graphs of the integrand \( y f_{U,V}(x y, y) \) in Figure 7.1, the integrand \( \frac{1}{\sqrt{2}} y f_{R,S} \left( \frac{1}{\sqrt{2}} x y, y \right) \) has similar curves but with different magnitude and spread.
The densities of $S_1$ computed from the joint density $f_{R,S}(r, s)$ are shown in Table 7.4. The accuracy is obtained by comparing the results with that of using the joint density $f_{U,V}(u, v)$. It is noticeable that using $f_{R,S}(r, s)$ produces reduced accuracy, i.e. especially around $x = 0.6$ with only 6 digits. The computation time is varying at different $x$ ranging from 11 seconds to 122 seconds. In general, it is more difficult to compute the density at the right tail than at the left tail.
Table 7.4. Accuracy of using \( f_{R,S}(r, s) \) to compute the density of \( S_1 \)

<table>
<thead>
<tr>
<th>( x_1 )</th>
<th>From ( f_{U,V} (u, v) ) (U, V)</th>
<th>From ( f_{R,S} (r, s) )</th>
<th>Digits Matching</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ED Result CPU</td>
<td>ED Result CPU</td>
<td>Accu Density</td>
</tr>
<tr>
<td>( -8 )</td>
<td>20.0 0.014814 11.0</td>
<td>20.0 0.014814 11.5</td>
<td>13 0.0148136805</td>
</tr>
<tr>
<td>( -7 )</td>
<td>20.0 0.020191 11.1</td>
<td>20.0 0.020191 11.8</td>
<td>14 0.02019086991</td>
</tr>
<tr>
<td>( -6 )</td>
<td>20.0 0.027741 11.2</td>
<td>20.0 0.027741 11.7</td>
<td>14 0.02774130633</td>
</tr>
<tr>
<td>( -5 )</td>
<td>20.0 0.038515 10.7</td>
<td>20.0 0.038515 11.4</td>
<td>14 0.03851513655</td>
</tr>
<tr>
<td>( -4 )</td>
<td>20.0 0.054233 10.8</td>
<td>20.0 0.054233 11.7</td>
<td>13 0.05423263604</td>
</tr>
<tr>
<td>( -3 )</td>
<td>20.0 0.077895 10.6</td>
<td>20.0 0.077895 16.4</td>
<td>14 0.07789546695</td>
</tr>
<tr>
<td>( -2 )</td>
<td>20.0 0.11512 11.0</td>
<td>20.0 0.11512 17.3</td>
<td>15 0.1151198573</td>
</tr>
<tr>
<td>( -1 )</td>
<td>20.0 0.17495 13.0</td>
<td>20.0 0.17495 31.8</td>
<td>15 0.1749471560</td>
</tr>
<tr>
<td>( -0.5 )</td>
<td>20.0 0.20975 24.5</td>
<td>20.0 0.20975 123</td>
<td>9 0.209746145</td>
</tr>
<tr>
<td>( -0.1 )</td>
<td>20.0 0.23564 13.6</td>
<td>20.0 0.23564 60.6</td>
<td>14 0.2356398151</td>
</tr>
<tr>
<td>0</td>
<td>20.0 0.24197 17.8</td>
<td>20.0 0.24197 75.1</td>
<td>9 0.2419707274</td>
</tr>
<tr>
<td>0.1</td>
<td>20.0 0.24727 13.6</td>
<td>20.0 0.24727 66.5</td>
<td>7 0.24772072</td>
</tr>
<tr>
<td>0.3</td>
<td>20.0 0.25217 17.6</td>
<td>20.0 0.25217 73.6</td>
<td>7 0.25211111</td>
</tr>
<tr>
<td>0.5</td>
<td>20.0 0.24487 17.7</td>
<td>20.0 0.24487 75.4</td>
<td>6 0.244872</td>
</tr>
<tr>
<td>0.7</td>
<td>20.0 0.22166 17.5</td>
<td>20.0 0.22166 67.1</td>
<td>6 0.221660</td>
</tr>
<tr>
<td>1.0</td>
<td>20.0 0.16276 18.6</td>
<td>20.0 0.16276 94.8</td>
<td>7 0.1627629</td>
</tr>
<tr>
<td>1.3</td>
<td>20.0 0.10037 18.9</td>
<td>20.0 0.10037 111</td>
<td>7 0.1003674</td>
</tr>
<tr>
<td>1.6</td>
<td>20.0 0.05466 20.5</td>
<td>20.0 0.05466 122</td>
<td>7 0.05546210</td>
</tr>
<tr>
<td>2.0</td>
<td>20.0 0.02346 19.0</td>
<td>20.0 0.02346 114</td>
<td>11 0.02346194552</td>
</tr>
<tr>
<td>2.5</td>
<td>20.0 0.007833 18.5</td>
<td>20.0 0.007833 90.1</td>
<td>10 0.007833468689</td>
</tr>
</tbody>
</table>

NIntegrate (first): compute the density using the formula (7.14) and evaluate the integral with Mathematica built-in function NIntegrate\. UniG: Compute the joint density \( f_{U,V}(u, v) \) using the numerical inversion algorithm UniG. NIntegrate (second): compute the density using the formula (7.24) and evaluate the integral with Mathematica built-in function NIntegrate\. \( g_1 \): Abadir’s formula \( g_1 \).

wp: computing precision. Accu: accuracy measured by the number of significant digits. ED: effective number of digits of precision in the result. CPU: computing time in seconds.

The accuracy and the density are obtained by digits matching with the following option specifications for our code DensityS[ ]:

\[
\text{Method} \rightarrow ("NIntegrate", \text{WorkingPrecision} \rightarrow 20), \quad \text{vs} \\
\text{MaxRecursion} \rightarrow 12, "\text{JointDensityUVR}", "\text{UniG}", "\text{M}" \rightarrow 35, \text{WorkingPrecision} \rightarrow 70)
\]

\[
\text{Method} \rightarrow ("NIntegrate", \text{WorkingPrecision} \rightarrow 20), \\
\text{MaxRecursion} \rightarrow 12, "\text{JointDensityURS}", "g1", "j0" \rightarrow 8, \text{WorkingPrecision} \rightarrow 30)
\]

Remember that Abadir’s method is geared towards generating results for the statistics \( R \times S \) and \( R / \sqrt{S} \), not the statistics \( S_3 \) and \( S_4 \). Using it for the purpose of computing \( S_3 \) and \( S_4 \) might not be using this particular joint density \( f_{R,S}(r, s) \), to its best advantage. Nevertheless, the formulae should work either way. Considering the integrand \( \frac{1}{\sqrt{\gamma^2}} y f_{R,S} \left( \frac{1}{\sqrt{\gamma^2}} x, y \right) \) is not monotonic decreasing in some parameter space, we conjecture that the joint density \( f_{U,V}(u, v) \) will still be preferred if we were instead to compute the densities of \( R \times S \) and \( R / \sqrt{S} \). This leads to a prima facie argument for preferring the normalization used by Tanaka (1996) rather than the normalization used by Abadir (1995b), following White (1958).

- 7.4. Generating the density of \( S_4 \) from \( f_{U,V}(u, v) \)

Recall that \( S_4 \) can be expressed by
Following Hogg and Craig (1995, pp. 170-175), the density of $S_4$ can be computed from the joint density $f_{U, V}(u, v)$.

Set

$$X = \frac{U}{\sqrt{V}}$$
$$Y = \sqrt{V}$$

and let

$$x = u / \sqrt{v}$$
$$y = \sqrt{v}$$

define a one-to-one transformation that maps the set $\mathcal{A} = \{(u, v) : -1/2 \leq u < \infty, 0 < v < \infty\}$ onto the set $\mathcal{B} = \{(x, y) : -\infty < x < \infty, 0 < y < \infty\}$.

The inverse of $x$ and $y$ can be written as

$$u = xy$$
$$v = y^2$$

The Jacobian of the transformation $J$ is calculated as

$$J = \begin{vmatrix}
\frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\
\frac{\partial v}{\partial x} & \frac{\partial v}{\partial y}
\end{vmatrix}
= \begin{vmatrix}
y & x \\
0 & 2y
\end{vmatrix}
= 2y^2 \neq 0$$

Then, the p.d.f. of $(X, Y)$ is given by, for $(x, y) \in \mathcal{B}$,

$$f_{X,Y}(x, y) = f_{U,V}(u, v) |J|
= f_{U,V}(xy, y^2) 2y^2
= 2y^2 f_{U,V}(xy, y^2)$$

Finally, the p.d.f. of $S_4$ can be obtained from $f_{X,Y}(x, y)$ by integrating $y$ out.

$$f_{S_4}(x) = \int_0^\infty f_{X,Y}(x, y) \, dy
= \int_0^\infty 2y^2 f_{U,V}(xy, y^2) \, dy$$

where the domain of $f_{U,V}(xy, y^2)$ is $\{(x, y) : -1/2 \leq xy < \infty, 0 < y < \infty\}$.
We truncate the integral (7.34) to make the computation efficient with the lower limit of integration $s_1$ and the upper limit of integration $s_2$.

$$f_{S_4}(x) = \int_{s_1}^{s_2} y^2 f_{U,V}(x, y^2) \, dy$$

where

As $x \leq -3.5$, set $s_1 = 0$
As $x \leq -2.4$, set $s_1 = 0.08$
As $x \leq -1.3$, set $s_1 = 0.09$
Otherwise, set $s_1 = 0.1$  

(7.35)

so that the integrand $2 y^2 f_{U,V}(x, y^2)$ at $y = s_1$ is between $10^{-32}$ and $5 \times 10^{-15}$ for $x \in [-3, 2.5]$, and

As $x \leq -1/6$, set $s_2 = -\frac{1}{2x}$
As $x \leq 0.4$, set $s_2 = 3$
As $x \leq 2.5$, set $s_2 = 4$
Otherwise, set $s_2 = 5$  

(7.36)

so that $s_2$ is the boundary point of the domain when $x \leq -1/6$ or the integrand $2 y^2 f_{U,V}(x, y^2)$ at $y = s_2$ is between $10^{-20}$ and $5 \times 10^{-10}$ for $x \in (-1/6, 2.5]$.

The graphs of the integrand $2 y^2 f_{U,V}(x, y^2)$ at different $x$ are drawn in Figure 7.6.

![Figure 7.6. Graphs of the integrand $2 y^2 f_{U,V}(x, y^2)$ at different $x$](image)

Imhof's formula cannot compute the density of $S_4$ as the characteristic function associated with $S_4$ is unknown. But the joint density $f_{U,V}(u, v)$ can be used to generate the density of almost all unit root statistics including $S_4$. Table 7.5 shows the densities of $S_4$ and the corresponding accuracy. The
computation time for each density varies from 10 seconds to 80 seconds. The accuracy is obtained by comparing two results computed with different parameter settings.
Table 7.5: Accuracy of using $f_{U_{1,T}}(u, v)$ to compute the density of $S_T$

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>NIntegrate,(wp=$20$)</th>
<th>NIntegrate,(wp=$80$)</th>
<th>Digits Matching</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ED</td>
<td>Result</td>
<td>CPU</td>
</tr>
<tr>
<td>-3</td>
<td>20.0</td>
<td>0.0086343</td>
<td>13.5</td>
</tr>
<tr>
<td>-2.7</td>
<td>20.0</td>
<td>0.020185</td>
<td>103</td>
</tr>
<tr>
<td>-2.5</td>
<td>20.0</td>
<td>0.033783</td>
<td>74.5</td>
</tr>
<tr>
<td>-2.3</td>
<td>20.0</td>
<td>0.054257</td>
<td>42.3</td>
</tr>
<tr>
<td>-2.2</td>
<td>20.0</td>
<td>0.067696</td>
<td>47.8</td>
</tr>
<tr>
<td>-2.1</td>
<td>20.0</td>
<td>0.083580</td>
<td>43.1</td>
</tr>
<tr>
<td>-2</td>
<td>20.0</td>
<td>0.102211</td>
<td>14.9</td>
</tr>
<tr>
<td>-1.9</td>
<td>20.0</td>
<td>0.12541</td>
<td>46.6</td>
</tr>
<tr>
<td>-1.8</td>
<td>20.0</td>
<td>0.14755</td>
<td>67.8</td>
</tr>
<tr>
<td>-1.6</td>
<td>20.0</td>
<td>0.20400</td>
<td>51.0</td>
</tr>
<tr>
<td>-1.4</td>
<td>20.0</td>
<td>0.24926</td>
<td>37.1</td>
</tr>
<tr>
<td>-1.2</td>
<td>20.0</td>
<td>0.33791</td>
<td>44.0</td>
</tr>
<tr>
<td>-1</td>
<td>20.0</td>
<td>0.40002</td>
<td>28.1</td>
</tr>
<tr>
<td>-0.9</td>
<td>20.0</td>
<td>0.42377</td>
<td>33.1</td>
</tr>
<tr>
<td>-0.8</td>
<td>20.0</td>
<td>0.43899</td>
<td>38.2</td>
</tr>
<tr>
<td>-0.7</td>
<td>20.0</td>
<td>0.44246</td>
<td>33.9</td>
</tr>
<tr>
<td>-0.6</td>
<td>20.0</td>
<td>0.43224</td>
<td>33.2</td>
</tr>
<tr>
<td>-0.5</td>
<td>20.0</td>
<td>0.41135</td>
<td>30.4</td>
</tr>
<tr>
<td>-0.4</td>
<td>20.0</td>
<td>0.38913</td>
<td>24.2</td>
</tr>
<tr>
<td>-0.3</td>
<td>20.0</td>
<td>0.37165</td>
<td>20.7</td>
</tr>
<tr>
<td>-0.2</td>
<td>20.0</td>
<td>0.35642</td>
<td>19.0</td>
</tr>
<tr>
<td>-0.1</td>
<td>20.0</td>
<td>0.34129</td>
<td>20.9</td>
</tr>
<tr>
<td>0</td>
<td>20.0</td>
<td>0.32557</td>
<td>19.5</td>
</tr>
<tr>
<td>0.1</td>
<td>20.0</td>
<td>0.30899</td>
<td>20.2</td>
</tr>
<tr>
<td>0.2</td>
<td>20.0</td>
<td>0.29152</td>
<td>20.3</td>
</tr>
<tr>
<td>0.3</td>
<td>20.0</td>
<td>0.27324</td>
<td>19.0</td>
</tr>
<tr>
<td>0.5</td>
<td>20.0</td>
<td>0.23494</td>
<td>18.4</td>
</tr>
<tr>
<td>0.7</td>
<td>20.0</td>
<td>0.19588</td>
<td>18.9</td>
</tr>
<tr>
<td>1</td>
<td>20.0</td>
<td>0.14020</td>
<td>18.9</td>
</tr>
<tr>
<td>1.3</td>
<td>20.0</td>
<td>0.09237</td>
<td>18.8</td>
</tr>
<tr>
<td>1.6</td>
<td>20.0</td>
<td>0.05671</td>
<td>16.6</td>
</tr>
<tr>
<td>2</td>
<td>20.0</td>
<td>0.025847</td>
<td>17.7</td>
</tr>
<tr>
<td>2.5</td>
<td>20.0</td>
<td>0.0078343</td>
<td>15.4</td>
</tr>
</tbody>
</table>

NIntegrate: compute the density using the formula (7.34) and evaluate the integral with Mathematica built-in function NIntegrate[]. UniG: Compute the joint density $f_{U_{1,T}}(u, v)$ using the numerical inversion algorithm UniG.

wp: computing precision. Accu: accuracy measured by the number of significant digits. ED: effective number of digits of precision in the result. CPU: computing time in seconds.

The accuracy and the density are obtained by digits matching with the following option specifications for our code DensityS4[]:


Using the densities in Table 7.5 as data points, we can draw the graph of the density of $S_T$ with quadratic spline interpolation as illustrated in Figure 7.7. By comparison, Figure 7.8 shows the simulated density of $S_{1:T}$ with $T$ finite.
7.5. Generating the density of $S_4$ from $f_{R,S}(r, s)$

$S_4$ can be expressed in terms of $R$ and $S$ as well

$$S_4 = \frac{U}{\sqrt{V}} = \frac{R/\sqrt{s}}{\sqrt{N/2}} = \frac{R}{\sqrt{s}}$$

(7.37)

Following Hogg and Craig (1995, pp.170-175), the density of $S_4$ can be computed from the joint density $f_{R,S}(r, s)$

Set

$$X = \frac{R}{\sqrt{s}}$$
$$Y = \sqrt{s}$$

(7.38)

and let

$$x = r/\sqrt{s}$$
$$y = \sqrt{s}$$

(7.39)
define a one-to-one transformation that maps the set \( \mathcal{A} = \{(r, s) : -\sqrt{2} / 2 \leq r < \infty, 0 < s < \infty \} \) onto the set \( \{(x, y) : -\infty < x < \infty, 0 < y < \infty \} \).

The inverse of \( x \) and \( y \) can be written as

\[
\begin{align*}
    r &= x y \\
    s &= y^2
\end{align*}
\]  

(7.40)

The Jacobian of the transformation \( J \) is calculated as

\[
J = \begin{vmatrix}
    \frac{\partial r}{\partial x} & \frac{\partial r}{\partial y} \\
    \frac{\partial s}{\partial x} & \frac{\partial s}{\partial y}
\end{vmatrix}
= \begin{vmatrix}
    y & x \\
    0 & 2y
\end{vmatrix}
= 2y^2 \neq 0
\]  

(7.41)

Then, the p.d.f. of \( (X, Y) \) is given by, for \( (x, y) \in \mathcal{B} \),

\[
f_{X,Y}(x, y) = f_{R,S}(r, s) |J|
= f_{R,S}(x y, y^2) 2y^2
= 2y^2 f_{R,S}(x y, y^2)
\]  

(7.42)

Finally, the p.d.f. of \( S_4 \) can be obtained from \( f_{X,Y}(x, y) \) by integrating \( y \) out.

\[
f_{S_4}(x) = \int_{-\infty}^{\infty} f_{X,Y}(x, y) \, dy 
= \int_{-\infty}^{\infty} 2y^2 f_{R,S}(x y, y^2) \, dy
\]  

(7.43)

where the domain of \( f_{R,S}(x y, y^2) \) is \( \{(x, y) : -\sqrt{2}/2 \leq x y < \infty, 0 < y < \infty \} \). The joint density \( f_{R,S}(r, s) \) is evaluated by Abadir’s formula \( g_1 \).

The integral (7.43) is truncated with parameters \( s_1 \) and \( s_2 \).

\[
f_{S_4}(x) = \int_{s_1}^{s_2} 2y^2 f_{R,S}(x y, y^2) \, dy
\]

where

\[
\begin{align*}
    &\text{As } x \leq -4, \ s_1 = 0 \\
    &\text{As } x \leq -2.3, \ s_1 = 0.11 \\
    &\text{As } x \leq -0.9, \ s_1 = 0.13 \\
    &\text{As } x \leq 0, \ s_1 = 0.15 \\
    &\text{Otherwise, } s_1 = 0.14
\end{align*}
\]  

(7.44)
so that the integrand $2y^2 f_{R,S}(x, y, y^2)$ at $y = s_1$ is between $10^{-33}$ and $6 \times 10^{-15}$ for $x \in [-3, 2.5]$, and

\[
\begin{align*}
\text{As } x &\leq -\frac{\sqrt{2}}{7}, \quad \text{set } s_2 = -\frac{\sqrt{2}}{2x} \\
\text{As } x &\leq 0, \quad \text{set } s_2 = 3.5 \\
\text{As } x &\leq 0.3, \quad \text{set } s_2 = 3.8 \\
\text{As } x &\leq 0.9, \quad \text{set } s_2 = 4 \\
\text{As } x &\leq 1.5, \quad \text{set } s_2 = 4.3 \\
\text{Otherwise,} &\quad \text{set } s_2 = 4.5
\end{align*}
\]

so that $s_2$ is the boundary point of the domain when $x \leq -\frac{\sqrt{2}}{7}$, or the integrand $2y^2 f_{R,S}(x, y, y^2)$ at $y = s_2$ is between $10^{-10}$ and $2 \times 10^{-6}$ for $x \in [-\sqrt{2}/7, 2.5]$.

Figure 7.9 reveals that the integrand diminishes before it degenerates as $y$ increases. The truncation point $s_2$ in (7.45) is carefully chosen so that the integration range only covers the non-degenerate interval.

![Figure 7.9. Distortion of the integrand $2y^2 f_{R,S}(x, y, y^2)$ using Adadir’s formula $g_1$](image)

The graphs of the integrand $2y^2 f_{R,S}(x, y, y^2)$ at different $x$ are sketched in Figure 7.10.
Table 7.6 shows the accuracy of using $f_{R,S}(r, s)$ to compute the density of $S_t$. The accuracy is poor in some range since Abadir’s formulae approximate the joint density poorly. The computation time consumed is significantly longer than that of using $f_{U,V}(u, v)$, i.e. as long as 161 seconds. Therefore, it is better to use $f_{U,V}(u, v)$ to generate the density of $S_t$ than $f_{R,S}(u, v)$.
Table 7.6. Accuracy of using $f_{u,v}(u, v)$ to compute the density of $S_t$

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>From $f_{u,v}(u, v)$</th>
<th>From $f_{u,x}(u, s)$</th>
<th>Digits Matching</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ED</td>
<td>Result</td>
<td>CPU</td>
</tr>
<tr>
<td>-3</td>
<td>20.0</td>
<td>0.0086343</td>
<td>10.1</td>
</tr>
<tr>
<td>-2.7</td>
<td>20.0</td>
<td>0.020185</td>
<td>78.6</td>
</tr>
<tr>
<td>-2.5</td>
<td>20.0</td>
<td>0.033783</td>
<td>55.4</td>
</tr>
<tr>
<td>-2.3</td>
<td>20.0</td>
<td>0.054257</td>
<td>31.4</td>
</tr>
<tr>
<td>-2.2</td>
<td>20.0</td>
<td>0.067696</td>
<td>35.4</td>
</tr>
<tr>
<td>-2.1</td>
<td>20.0</td>
<td>0.083580</td>
<td>33.3</td>
</tr>
<tr>
<td>-2</td>
<td>20.0</td>
<td>0.10211</td>
<td>10.6</td>
</tr>
<tr>
<td>-1.9</td>
<td>20.0</td>
<td>0.12341</td>
<td>34.5</td>
</tr>
<tr>
<td>-1.8</td>
<td>20.0</td>
<td>0.14755</td>
<td>52.2</td>
</tr>
<tr>
<td>-1.6</td>
<td>20.0</td>
<td>0.20400</td>
<td>39.1</td>
</tr>
<tr>
<td>-1.4</td>
<td>20.0</td>
<td>0.26926</td>
<td>28.2</td>
</tr>
<tr>
<td>-1.2</td>
<td>20.0</td>
<td>0.33791</td>
<td>34.0</td>
</tr>
<tr>
<td>-1</td>
<td>20.0</td>
<td>0.40002</td>
<td>28.3</td>
</tr>
<tr>
<td>-0.9</td>
<td>20.0</td>
<td>0.42377</td>
<td>24.8</td>
</tr>
<tr>
<td>-0.8</td>
<td>20.0</td>
<td>0.43899</td>
<td>28.5</td>
</tr>
<tr>
<td>-0.7</td>
<td>20.0</td>
<td>0.44246</td>
<td>25.0</td>
</tr>
<tr>
<td>-0.6</td>
<td>20.0</td>
<td>0.43224</td>
<td>24.9</td>
</tr>
<tr>
<td>-0.5</td>
<td>20.0</td>
<td>0.41185</td>
<td>25.1</td>
</tr>
<tr>
<td>-0.4</td>
<td>20.0</td>
<td>0.38913</td>
<td>18.2</td>
</tr>
<tr>
<td>-0.3</td>
<td>20.0</td>
<td>0.37165</td>
<td>16.7</td>
</tr>
<tr>
<td>-0.2</td>
<td>20.0</td>
<td>0.35642</td>
<td>14.2</td>
</tr>
<tr>
<td>-0.1</td>
<td>20.0</td>
<td>0.34129</td>
<td>15.3</td>
</tr>
<tr>
<td>0</td>
<td>20.0</td>
<td>0.32557</td>
<td>14.6</td>
</tr>
<tr>
<td>0.1</td>
<td>20.0</td>
<td>0.30899</td>
<td>15.6</td>
</tr>
<tr>
<td>0.2</td>
<td>20.0</td>
<td>0.29152</td>
<td>15.2</td>
</tr>
<tr>
<td>0.3</td>
<td>20.0</td>
<td>0.27324</td>
<td>14.2</td>
</tr>
<tr>
<td>0.5</td>
<td>20.0</td>
<td>0.23494</td>
<td>14.5</td>
</tr>
<tr>
<td>0.7</td>
<td>20.0</td>
<td>0.19588</td>
<td>15.7</td>
</tr>
<tr>
<td>1</td>
<td>20.0</td>
<td>0.14020</td>
<td>14.3</td>
</tr>
<tr>
<td>1.3</td>
<td>20.0</td>
<td>0.092837</td>
<td>14.7</td>
</tr>
<tr>
<td>1.6</td>
<td>20.0</td>
<td>0.056715</td>
<td>12.4</td>
</tr>
<tr>
<td>2</td>
<td>20.0</td>
<td>0.025847</td>
<td>13.6</td>
</tr>
<tr>
<td>2.5</td>
<td>20.0</td>
<td>0.0078343</td>
<td>11.5</td>
</tr>
</tbody>
</table>

NIntegrate (first): compute the density using the formula (7.34) and evaluate the integral with Mathematica built-in function NIntegrate[].

UniG: Compute the joint density $f_{u,v}(u, v)$ using the numerical inversion algorithm UniG. NIntegrate (second): compute the density using the formula (7.43) and evaluate the integral with Mathematica built-in function NIntegrate[].

wp: computing precision. Accu: accuracy measured by the number of significant digits. ED: effective number of digits of precision in the result. CPU: computing time in seconds.

The accuracy and the density are obtained by digits matching with the following option specifications for our code DensitySample[]:

- (Method -> "NIntegrate", WorkingPrecision -> 20, MaxRecursion -> 36, "JointDensityRS", "j0" -> 8), WorkingPrecision -> 30)
8. Conclusion

This chapter investigates various approaches to calculating the densities of the unit root statistics $S_3$ and $S_4$. Their densities can be estimated by Monte Carlo simulation, computed from the relevant characteristic function by Imhof’s formula, generated from the joint density $f_{U,V}(u, v)$ which is obtained by numerically inverting a reduced Laplace transform $\hat{f}(u, \beta)$ with UniG, and generated from the joint density $f_{R,S}(r, s)$ for which the analytical formulae are given by Abadir (1995b). All methods have their own merits and limitations.

The Monte Carlo method is the most straightforward to use for all unit root statistics. It can estimate the whole distribution of a statistic within seconds if matrix operation is used in the simulation. However its accuracy is questionable especially when computing the tails of the distribution, the very areas that are relevant for hypothesis testing. Furthermore, the distributions simulated is a finite sample distribution. Imhof’s formula can compute both the distribution function and the density function numerically. The computation time is reasonable. For the distribution of $S_3$ at a single point, it takes half a second to compute the distribution function and a few seconds to compute the density function. Compared with Monte Carlo method, Imhof’s formula has much better accuracy especially for computing the tails: the density can be correct to seven significant digits. But the accuracy close to the origin is poor. The density at the origin is correct to only two significant digits. Another disadvantage of Imhof’s formula is the restriction on its applicability. Since the formula is applied to the c.f. associated with the statistic, it requires that the relevant c.f. is readily available. Therefore, we cannot use Imhof’s formula to calculate the distribution of $S_4$ as the c.f. related to $S_4$ is unknown.

The joint densities $f_{U,V}(u, v)$ and $f_{R,S}(r, s)$ are extremely useful because they can be used to generate the density of the distribution of almost any unit root statistic in the context of a model as general as (1.5) above. We make joint density $f_{U,V}(u, v)$ available through our result (albeit obtained using Mathematica alone) on the one-sided Laplace transform. This thesis contains fully worked-out code for obtaining $f_{U,V}(u, v)$. The joint density $f_{U,V}(u, v)$ is far superior to the joint density $f_{R,S}(r, s)$ for density generation of unit root statistics because computing $f_{U,V}(u, v)$ takes less time and yields more steady result. More importantly, the integrand involving $f_{U,V}(u, v)$ is monotonic decreasing. For the distribution of $S_3$, 


computing the density at a single point from \( f_{U,V}(u, v) \) costs 10–23 seconds and has accuracy of 10–18 significant digits. While, the computation time ranges from 11 seconds to 123 seconds when computing the density from \( f_{R,S}(r, s) \). The accuracy yielded is as low as six significant digits. For the distribution of \( S_4 \), the computation becomes more difficult. But it is still better to use \( f_{U,V}(u, v) \) than to use \( f_{R,S}(r, s) \).

To conclude, the joint density \( f_{U,V}(u, v) \) is recommended when one wants to compute the distribution of a statistic accurately and is less concerned about the computation time. Monte Carlo simulation will be a good choice if one would like to sketch the distribution quickly. Imhof’s formula is only useful when the relevant c.f. is available. It is very accurate for the tails of the distribution and faster than using \( f_{U,V}(u, v) \).

The natural first step for further work would be to consider the AR(1) model with drift and the AR(1) model with drift and a time trend as were treated in Chapter 17 in Hamilton (1994). This work shows that, while the unit root test statistics can be written entirely in terms of \( S_1, S_2, S_3 \) and \( S_4 \), the formulae are highly specific to each model. And indeed, analytical formulae of the form derived by Abadir (1995b) are not available for these models in the published literature. Another aspect of the approach that was left untreated was the role of the initial condition. Provided the AR(1) model exhibits either stationarity or unit-root non-stationarity, the initial condition is asymptotically negligible (see Abadir, 1993) but obviously finite-sample properties may well be affected by it. One ambition would be to derive analytical formulae for the finite-sample distributions of the test statistics and, in such research program, using a computing platform like Mathematica as is done here would seem to be an essential tool, at least from today’s standpoint.
Chapter 5

Conclusion

1. Contributions of the Thesis

1.1. Chapter 2

This thesis has applied numerical analytic methods to some difficult and as yet unsolved problems in Finance, Econometrics and Statistics. It has offered a modern, computational approach to these problems that has been based on using multi-precision arithmetic using the Mathematica 9.0 platform. In Chapter 2, we considered the problem of pricing the continuous arithmetic Asian options. Even in the standard Black-Scholes framework adopted in this thesis, which entails the asset evolving according to geometric Brownian motion, the problem is non-trivial and unresolved in the literature. The approach here in a sense turned the problem on its head: it attacked the pricing problem purely as a computational one, firstly demonstrating that with almost all proposed methods, trying to force more and more accuracy on the problem in a context of fixed machine precision eventually led to round-off error propagating back to degrade the efficacy in many methods. The solution was to employ multi-precision arithmetic and it was shown that the computational problem for almost all the methods was solved when doing so. Many methods were based on inverting the Geman-Yor Laplace transform. This gave us, we believe, the proper basis on which to compare and contrast the various methods, computational time becoming the key diagnostic.

There are various possibilities for further work on the Asian option pricing problem. On the computational problem itself, the experimental design of Fu, Madan and Wang (1999) augmented by Craddock, Heath and Platen (2000) is established as the standard, and the results in all situations can be computed with multi-precision arithmetic to be so accurate, that we were able to adopt reference prices.
The next stage from the point of view of numerical analysis would be to construct a formal error analysis for each method (even if such a task would be another thesis in itself). The focus here was on the computational aspects of the pricing problem and little consideration was given to the trader who actually buys and sells Asian options. One question would be whether the trader’s optimum method would be different from the numerical analyst’s: perhaps the trader would be prepared to put up with less robustness across possible parameter values to gain increased computational speed, even if it meant on a few occasions facing losses owing to the option not being priced very accurately. More fundamentally, the trader has to operate in the framework of the actual process that generates the asset’s path, to which geometric Brownian motion is only an approximation. There are other possibilities (that each would probably require a thesis chapter in their own right). For example, Dassios and Nagaradjasarma (2006) price Asian options in the context of the square root process. There does not appear, however, to be an approach to the pricing problem as yet that seeks to select a process generating the asset’s path among plausible alternatives on the basis of the data themselves.

1.2. Chapter 3

We considered the problem of computing the distribution and density functions for two types of infinitely divisible distributions: stable distributions and certain such distributions associated with hyperbolic functions. Computing the former distributions and densities is a classical and still unresolved problem in the Statistics literature; computing the latter has become of interest only recently owing to the work of Biane, Pitman and Yor (2001) and Pitman and Yor (2003). As we discuss below, this latter type of distribution is potentially useful in the study of unit root distributions.

With respect to the stable distributions, our contribution was similar to that in Chapter 2: we used multi-precision arithmetic to get around the problems of computing the distributions and density functions whose series representations are notoriously sensitive to the value of the exponent in the Laplace transform representation. We worked with the most general type of function describing the stable distributions—the Fox \( H \) function—in conjunction with variable computing precision to seek a method that would apply robustly across all values of the exponent. A weakness of our work here is that, like in Chapter 2, we used reference densities, even if they were robustly computed under various favorable scenarios given the underlying parameters. While such an approach is strongly justifiable in the context of the established experimental design by Craddock, Heath and Platen (2000), it is less justifiable here.
Also, in searching for a robust, uniform method, it should be recognized that we could get significant computational improvements in some subclass of distributions, for example those described by the Meijer $G$ function. This would be worthy of investigation in further work. For the infinitely divisible distributions associated with hyperbolic functions, we were able, using the Mathematica platform to extend results in the literature and compute appropriately smooth distributions and density functions in the most important cases.

1.3. Chapter 4

Here, we looked at an unresolved problem in the econometric literature: computing and simulating unit root distributions and densities. In fact, following Abadir (1995), we concentrated only on the two fundamental distributions for unit root statistics: in principle the densities of all standard unit root statistics derived in the autoregressive framework are related to these two densities (see Hamilton, 1994, Chapter 17 for some examples) by means of the joint density $f_{U,V}(u, v)$ or the joint density $f_{R,S}(r, s)$. Analytical representations for the densities and distributions (as opposed to expressions involving Brownian functionals) are, however, only available in this basic case, which limited our discussion. Nevertheless, we were still able to make a contribution against the literature. A key result, using the Mathematica 9.0 platform, reduced the two-dimensional Laplace transform for the relevant joint density of the unit root statistics to a one-dimensional Laplace transform. Specifically, the joint density $f_{U,V}(u, v)$ is embedded in the two-dimensional Laplace transform of $(U, V)$

$$
\tilde{f}(\alpha, \beta) = \mathbb{E} \left[ \exp(-\alpha U - \beta V) \right] 
= \int_0^\infty \int_{-1/2}^1 e^{-u-\beta v} f_{U,V}(u, v) \, du \, dv
= e^{\alpha/2 \left( \cosh \sqrt{2 \beta} + \frac{\alpha}{\sqrt{2 \beta}} \sinh \sqrt{2 \beta} \right)^{1/2}}
$$

We reduce it to an one-dimensional Laplace transform

$$
\hat{f}(u, \beta) = \int_0^\infty e^{-\beta v} f_{U,V}(u, v) \, dv
= \begin{cases} 
2 \sqrt{\frac{\beta}{\pi}} \left( \frac{\cosh \sqrt{2 \beta}}{(2u+1)} \right)^{1/2} e^{-\frac{i}{\sqrt{2}} (2u+1) \sqrt{\beta} \ \coth \left( \sqrt{2 \beta} \right)}, & u > -1/2 \ \text{and} \ \text{Re}(\beta) > 0 \\
0, & u = -1/2 \ \text{and} \ \text{Re}(\beta) > 0
\end{cases}
$$

This reduction not just lowers the dimensions to one but also eliminates the two-sided Laplace transform of $f_{U,V}(u, v)$ with respect to $u$. Therefore, one-dimensional inversion algorithms we studied throughout
this thesis can be used to invert the reduced Laplace transform \( \hat{f}(u, \beta) \). We further show \( \hat{f}(u, \beta) \) can be inverted by the UniG algorithm or the GWR algorithm to obtain the joint density \( f_{U,V}(u, v) \), whereas the inversion fails with all of the other algorithms. By comparing \( f_{U,V}(u, v) \) with \( f_{R,S}(r, s) \) in terms of computing densities of unit root distributions, we find \( f_{U,V}(u, v) \) is superior to \( f_{R,S}(r, s) \) although the latter can be obtained by analytical formulae proposed by Abadir (1995).

1.4. Mathematica code

The Mathematica code for numerical experiments in this thesis also make a contribution. The code is written by us with great skills so that the functions defined can be manipulated with various options such as the Method option and the WorkingPrecision option. We create various functions for different kinds of tasks, from computing the density of a distribution to finding the number of matching digits between two results. For example, the function DensityComparison[ ] can be used to draw an entire table for comparison of different methods with full details of the results such as the accuracy and computation time, providing that we specify the points to be evaluated at, what methods to be used to find the reference results, and what methods to be compared. The code is saved in a Mathematica notebook. To use the functions, we should first evaluate all initialization cells in the notebook since many functions need calling other functions to perform the computation.
2. Suggestions for Further Work

In Chapter 2, we have suggested that the approach be extended to encompass different models for the asset price, beyond that standard assumption of geometric Brownian motion and the standard Black-Scholes framework itself. It could also be extended robustly to compute various quantities, called Greeks, that would relate to how Asian options could be used as an asset in the wider context of an investment portfolio. In Chapter 3, our search was for a computational method that could apply uniformly across the stable distributions. This could be supported by further work that characterizes the numerical methods which best support computing a given type of stable distribution and then assesses the advantages in using a specific method over our general method when the parameters governing the form of the stable density or distribution are known. Perhaps the most interesting avenue for future research, however, is in the use of the infinitely divisible distributions based on hyperbolic functions in the context of the unit root densities. For example, equation (1.1) above can be decomposed into the product

\[ \exp\left(\frac{\alpha}{2}\right) \left( \cosh \sqrt{2\beta} \right)^{-1/2} \left( 1 + \alpha \frac{\tanh \sqrt{2\beta}}{\sqrt{2\beta}} \right)^{-1/2} \]  

(2.1)

which translates into the convolution of \( C_{1/2} \) with a density that is like \( T_{1/2} \) (or with terms in a binomial expansion in various \( T \)). Series expressions for the densities of unit root statistics pertaining to the AR(1) model with drift, and with a time trend, are likely to be even more complicated but it is possible that using the distributions considered by Pitman and Yor (2003), the expressions that would be involved could be tamed. The characteristic function (1.1) and (2.1) becomes more complicated if the initial condition is not zero (see Tanaka, 1996). Finally, it might be possible to take a different approach based on the factorization of the relevant joint density into the product of a conditional density and a marginal density. The approach by Malham and Wiese (2014) and its antecedents (e.g. Rydén and Wiktorsson, 2001) might then prove to be fruitful. While this thesis has contributed problems such as the above, it is clear that there are many more problems to solve.
Appendices

- **A1. Smooth Kernel Distribution**

`Mathematica` 9.0 can draw a smooth kernel distribution based on the input data values. Let \( \{x_i\} \) be a sequence of independent and identically distributed (i.i.d.) samples drawn from some distribution with unknown probability density function \( f(x) \). To estimate the shape of the density function, \( f(x) \) for a value \( x \) is given by its kernel density estimator \( \hat{f}(x) \):

\[
\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} k\left( \frac{x-x_i}{h} \right)
\]

where \( k(x) \) is a smoothing kernel which can be given different specifications, and \( h > 0 \) is the bandwidth parameter.

- **A2. Parabolic Cylinder Function**

A parabolic cylinder function \( D_n(z) \) satisfies the Weber differential equation

\[
y'' + \left( v + \frac{1}{4} - \frac{1}{4} z^2 \right)y = 0
\]

When \( v \) is a nonnegative integer \( n \), \( D_n \) is

\[
D_n(x) = 2^{-n/2} e^{-x^2/4} H_n\left( x / \sqrt{2} \right)
\]

where \( H_n(x) \) is a Hermite polynomial.

Letting \( K(v, z) = e^{z^2/4} D_n(z) \) the series expansion for \( K(v, z) \) is given by

\[
K(v, z) = 2^{v/2} \sqrt{\pi} \sum_{j=0}^{\infty} \left( \frac{1}{j!} \frac{(\sqrt{2})^j}{(1-v+j/2)} \right) e^{z^2/2} \sum_{j=0}^{\infty} \frac{(-z/\sqrt{2})^j}{j!} e^{-(1-v-j/2)/2}
\]


Boyle, Broadie and Glasserman 1997


1. Code in Chapter 2

1.1. Check accuracy

1.1.1. MatchedDP

MatchedDP[{x_1, x_2, ...}] compares numbers \( x_1, x_2, \ldots \) and gives the effective number of digits of precision in each number, the number of decimal places to which they agree, and a result which consists of the matched decimal places. The result is referred to as a reference number.

MatchedDP[{x_1, x_2, ...}, n] gives the same information as MatchedDP[{x_1, x_2, ...}] and, in addition, shows a truncated result which consists of the first \( n \) matched decimal places if the number of matched decimal places is greater than or equal to \( n \). Otherwise, the truncated result will consist of only the matched decimal places.

MatchedDP[{x_1, x_2, ...}, {y_1, y_2, ...}] compares each \( y_j \) with the reference number obtained by comparing \( x_1, x_2, \ldots \) and gives the effective number of digits of precision in each \( y_j \), the number of decimal places to which each \( y_j \) and the reference number agree, and the corresponding result which consists of the matched decimal places.

MatchedDP[{x_1, x_2, ...}, {y_1, y_2, ...}, n] gives the same information as MatchedDP[{x_1, x_2, ...}, {y_1, y_2, ...}] and, in addition, shows each truncated result which consists of the first \( n \) matched decimal places if the number of matched decimal places is greater than or equal to \( n \). Otherwise, the truncated result will consist of only the matched decimal places.

MatchedDP[x, {y_1, y_2, ...}] uses \( x \) as the reference number and compares each \( y_j \) with \( x \). In this case, the extra decimal places to the right of the reference number is padded with zero within its precision.

- The tie-breaking rule for MatchedDP is round half to even.
- MatchedDP rounds decimal numbers to each decimal place within their precision from left to right successively until the function finds the number of the most matched decimal places.

Examples

Source code

MatchedDP[xr_, ylist__List, n__Integer] :=
Module[{xlist, xlen, x, xld, xp, xpl, xrd, xlimit, mdpl, ref1, mdp2, j, ref2, mdp, ref, xspecify, xrow1, xcolheadings, xrow2, ylen, y, yld, yp, ypl, yrd, ylimit, accur, k, ycorrect, yspecify, yrow1, yrow2, colheadings, xcolumn, ycolumnheadings, ytable, ycolheadings, ytable, rowheadings, subtable, whoetable},
xlist = If[VectorQ[xr], xr, {xr}];
xlen = xlen = Length[xlist];
x[i_] := x[i] = xlist[[i]]; xld[i_] := xld[i] = RealDigits[x[i]][[2]]; xp[i_] := xp[i] = Precision[x[i]]; xpl[i_] := xpl[i] = If[xp[i] = MachinePrecision, "MP", N[Round[xp[i], 10^(-1)], 3]]; xrd[i_] := xrd[i] = xp[i] - xld[i];
xlimit = Floor[Min[Table[xrd[i], {i, xlen}]]]; (*******
mdpl := mdpl = Floor[xrd[1]]; ref1 := ref1 = If[mdpl = \infty, x[1], N[Round[x[1], 10^(-mdpl)], xld[1] + mdpl]]; mdp2 := mdp2 = Which[xlimit = \infty, If[Apply[Equal, Table[x[i], {i, xlen}]], \infty, 0], Apply[Unequal, Sign[Table[x[i], {i, xlen}]]] &&
Total[Abs[Sign[Table[x[i], {i, xlen}]]]] = xlen, 0,
True, For[j = 1, Or[Apply[Equal, Round[xlist, 10^(-1)]],
Apply[Equal, Floor[xlist, 10^(-1)]]] \&\& j \le xlimit, j++,
Which[j = 1, 0, Apply[Equal, Round[xlist, 10^(-1)]]], j - 1, True, j - 2]]; ref2 := ref2 = Which[mdp2 = 0, "N/A", mdp2 = \infty, x[1], xld[1] + mdp2 < 0, 0, True, N[Round[x[1], 10^(-mdp2)], xld[1] + mdp2]];
If[xlen == 1, mdp = mdp1; ref = ref1, mdp = mdp2; ref = ref2];
**************
xspecify := xspecify = 
If[n <= mdp, N[Round[x[1], 10^-n], x[1] + n], N[Round[x[1], 10^-N[ref]], x[1] + mdp]]; 
xrow1[i_] := xrow1[i] = If[i == 1, {xpl[i], mdp, ref}, 
{xpl[i], SpanFromAbove, SpanFromAbove}]; 
xrow2[i_] := xrow2[i] = If[i == 1, {xpl[i], mdp, ref, xspecify}, 
{xpl[i], SpanFromAbove, SpanFromAbove, SpanFromAbove}];
**************
ylen := ylen = Length[ylist]; 
y[i_] := y[i] = ylist[[i]]; 
yld[i_] := yld[i] = RealDigits[y[i]][[2]]; 
yp[i_] := yp[i] = Precision[y[i]]; 
yp1[i_] := yp1[i] = If[yp[i] = MachinePrecision, "MP", N[Round[yp[i], 10^-i], 3]]; 
yrd[i_] := yrd[i] = yp[i] - yld[i]; 
ylimit[i_] := ylimit[i] = Floor[Min[yrd[i], mdp]]; 
accu[i_] := accu[i] = 
Which[mdp = 0, 0, ylimit[i] = \infty, If[y[i] = ref, \infty, 0], 
!Sign[y[i]] = Sign[ref] \&\& And[!Sign[y[i]] = 0, !Sign[ref] = 0, 0, 
True, For[k = 1, 0 \infty[Round[y[i], 10^-k], 
Floor[y[i], 10^-k] = Floor[ref, 10^-k] \&\& k \leq ylimit[i], k++]]]; 
Which[k = 1, 0, Round[y[i], 10^k-1] = Round[ref, 10^k-1], k - 1, True, k - 2]]; 
ycorrect[i_] := ycorrect[i] = Which[ycorrect[i] = 0, "N/A", accu[i] = \infty, y[i], 
yld[i] + accu[i] < -0, 0, True, N[Round[y[i], 10^-accu[i]], yld[i] + accu[i]]]; 
yspecify[i_] := yspecify[i] = If[n <= accu[i], N[Round[y[i], 10^n], yld[i] + n], 
N[Round[y[i], 10^-accu[i]], yld[i] + accu[i]]]; 
yrow1[i_] := yrow1[i] = {yp1[i], accu[i], ycorrect[i]}; 
yrow2[i_] := yrow2[i] = {yp1[i], accu[i], ycorrect[i], yspecify[i]};
**************
If[n == Null, 
colheadings := 
colheadings = Map[Style[H, Bold] \&, {"Precision", "AccuracyDP", "Result"}]; 
xcolheadings := xcolheadings = {"Reference", SpanFromLeft, SpanFromLeft}; 
xtable := xtable = Table[xrow1[i], {i, xlen}]; 
ycolheadings := ycolheadings = {"Candidate", SpanFromLeft, SpanFromLeft}; 
ytable := ytable = Table[yrow1[i], {i, ylen}], 
(*otherwise*) 
colheadings := colheadings = Map[Style[H, Bold] \&, 
{"Precision", "AccuracyDP", "Result"}, Row[{"Show ", n, " DPs"}]]]; 
xcolheadings := xcolheadings = {"Reference", SpanFromLeft, 
SpanFromLeft, SpanFromLeft}; 
xtable := xtable = Table[xrow2[i], {i, xlen}]; 
ycolheadings := 
{"Candidate", SpanFromLeft, SpanFromLeft, SpanFromLeft}; 
ytable := ytable = Table[yrow2[i], {i, ylen}]; ]; 
If[ylist == Null, 
rowheadings := 
rowheadings = Join[{"", SpanFromAbove}, Table[Row[{"x", i}], {i, xlen}]]; 
subtable := subtable = Join[{colheadings, xcolheadings}, xtable], 
(*otherwise*) 
rowheadings := rowheadings = Join[{"", SpanFromAbove}, Table[Row[{"x", i}], 
{i, xlen}], {""}, Table[Row[{"y", i}], {i, ylen}]]; subtable := 
subtable = Join[{colheadings, xcolheadings}, xtable, {ycolheadings}, ytable]; 
wholetable := wholetable = MapThread[Prepend, {subtable, rowheadings}]; 
Grid[wholetable, Alignment -> (Center, Center), Frame -> All, Background -> 
None, None, 
{{{2, 2}, {2, 4}} \rightarrow LightGray, {{xlen + 3, xlen + 3}, {2, 4}} \rightarrow LightGray}]
]
1.1.2. MatchedSD

MatchedSD[\{x_1, x_2, ...\}] compares numbers \(x_1, x_2, ...\) and gives the effective number of digits of precision in each number, the number of significant digits to which they agree, and a result which consists of the matched significant digits. The result is referred to as a reference number.

MatchedSD[\{x_1, x_2, ..., n\}]
gives the same information as MatchedSD[\{x_1, x_2, ...\}] and, in addition, show a truncated result which consists of the first \(n\) matched significant digits if the number of matched significant digits is greater than or equal to \(n\). Otherwise, the truncated result will consist of only the matched significant digits.

MatchedSD[\{x_1, x_2, ..., \\{y_1, y_2, ...\}\}] compares each \(y_i\) with the reference number obtained by comparing \(x_1, x_2, ...\) and gives the effective number of digits of precision in each \(y_i\), the number of significant digits to which each \(y_i\) and the reference number agree, and the corresponding result which consists of the matched significant digits.

MatchedSD[\{x_1, x_2, ..., \\{y_1, y_2, ..., n\}\}]
gives the same information as MatchedSD[\{x_1, x_2, ...\}], \{y_1, y_2, ...\} and, in addition, show each truncated result which consists of the first \(n\) matched significant digits if the number of matched significant digits is greater than or equal to \(n\). Otherwise, the truncated result will consist of only the matched significant digits.

MatchedSD[x, \{y_1, y_2, ...\}]
use \(x\) as the reference number and compare each \(y_i\) with \(x\). In this case, the extra significant digits to the right of the reference number is padded with zero within its precision.

- The tie-breaking rule for MatchedSD is round half to even.
- MatchedSD rounds decimal numbers to each significant digit within their precision from left to right successively until the function finds the number of the most matched significant digits.
- The matched significant digits can be used as an indicator of accuracy for numerical methods.

Examples

Source code

Clear[MatchedSD];
MatchedSD[\{x_, yList__List, n__Integer\}] :=
  Module[{xList, xLen, x, xLD, xWindow, xLimit, RoundSD, FloorSD, msd1, ref1, msd2, j, ref2, msd, ref, xSpecify, xRow1, xRow2, yLen, y, yLD, yp, yPL, yLimit, accu, k, yCorrect, xSpecify, yRow1, yRow2, colHeadings, xColHeadings, xTable, yColHeadings, yTable, rowHeadings, subtable, wholetable},
    xList := xList = If[VectorQ[x], x, (x,)];
    xLen := xLen = Length[xList];
    xList[i_] := x[i] = xList[[i]];
    xLD[i_] := xLD[i] = RealDigits[x[i]][[2]];
    xWindow[i_] := xWindow[i] = Precision[x[i]];
    xPL[i_] := xPL[i] = If[xP[i] = MachinePrecision, "MP", N[Round[xP[i], 10^-1], 3]];
    xLimit = If[Apply[And, Map[IntegerQ, xList]],
        Max[Table[xLD[i], (i, xLen)]], Floor[Min[Table[xP[i], (i, xLen)]]]]
    RoundSD[x_, i_] := Round[x, 10^(-i + Floor[RealExponent[x] + 1])];
    FloorSD[x_, i_] := Floor[x, 10^(-i + Floor[RealExponent[x] + 1])];
    (******)
    msd1 := msd1 = Floor[xP[1]]; 
    ref1 := ref1 = If[msd1 = Infinity, x[1], N[RoundSD[x[1], msd1], msd1]]; 
    msd2 := msd2 = For[j = 1, Or[Apply[Equal, RoundSD[xList, j]], ] && j <= xLimit, j++]; 
    Which[j = 1, 0, Apply[Equal, RoundSD[xList, j - 1]], j - 1, True, j - 2]; 
    ref2 := ref2 = Which[msd2 = 0, "N/A", msd2 <= xLD[1], RoundSD[x[1], msd2], True, N[RoundSD[x[1], msd2, msd2]]]; 
    If[xLen = 1, msd = msd1; ref = ref1, msd = msd2; ref = ref2]; 
    (******)
    xSpecify := xSpecify = Which[n <= msd && n <= xLD[1], RoundSD[x[1], n], 
        n <= msd, N[RoundSD[x[1], n], n], True, N[RoundSD[x[1], msd], msd]];
    xRow1[i_] := xRow1[i] = If[i = 1, (xP[i], msd, ref), 
        (xP[i], SpanFromAbove, SpanFromAbove)];
AsianOptionPrice

Examples

Source code

Options[AsianOptionPrice] =

{Method -> Automatic, WorkingPrecision -> MachinePrecision, AttemptAccuracy -> 15,
  "Shifting" -> 0, "Scaling" -> {0, 1}, Contour -> Automatic, NDSolveMethod -> Automatic,
FindRootMethod -> Automatic, NIntegrateMethod -> Automatic, MaxRecursion -> 18];
AsianOptionPrice[ds_, dk_, dr_, dδ_, do_, dt_, dt0_: 0, dt0: 0, dSbar_: 0, 
  opts : OptionsPattern[]] := Module[
  {S, K, r, δ, σ, T, t, t0, Sbar, ν, h, q, a, 
   Cv1, C1, μ, fhat, Cv2a, Cv2b, Cv2, Ct2, algorithm, arguments, Pv3, Pt3, Ct3, 
   Ct4, Ct5, Cv6, Ct6, Cv7, Ct7, vectorise, method, parameters, call, rules}, 
  S = Rationalize[ds]; (*price*)
  K = Rationalize[dk]; (*strike price*)
  r = Rationalize[dr]; (*interest rate*)
  δ = Rationalize[dδ]; (*dividend*)
  σ = Rationalize[do]; (*volatility*)
  T = Rationalize[dt]; (*time to maturity*)
  t = Rationalize[dt0]; (*time to average*)
  Sbar = Rationalize[dSbar]; (*realised price*)
  ν = (r - δ) / σ^2; (*normalized interest rate*)
  h = (σ/2) S^2 (T - t); (*normalized time to maturity*)
  q = T - t0 / (2 σ^2) (K - Sbar / T - t0); (*normalized strike price*)
  a := Ceiling[Max[0, 2 + 2 ν]] + 1;
  (**)(*When q is non-positive*)
  Cv1 := Exp[2 h (ν + 1)] - 1 - q;
  Ct1 := Exp[-r (T - t)] S / (σ/2)^2 Cv1;
  (**)
  (**)(*Invert Laplace transform*)
  μ = Sqrt[2 λ + ν^2];
  fhat := 1 / (λ^2 - 2 λ (ν + 1)) Power[2 q, -1/2 (μ - ν - 2)]
  Gamma[1/2 (μ + ν + 4)] Hypergeometric1F1Regularized[1/2 (μ - ν - 2), μ + 1, -1/2 q];
  Cv2a := NInverseLaplaceTransform[fhat, λ, h, 
    FilterRules[{opts}, Options[NInverseLaplaceTransform]]];
  Cv2b := NInverseLaplaceTransform[fhat, λ, h, 
    FilterRules[{opts, Contour -> a}, Options[NInverseLaplaceTransform]]];
  Cv2 := If[algorithm == "Bromwich" & OptionValue[Contour] == Automatic, 
    Cv2b, Cv2a];
  Ct2 := Exp[-r (T - t)] S / (σ/2)^2 Cv2;
  (**)
  (**)(*Use spectral representation*)
  Pv3 := Apply[ToExpression, arguments, FilterRules[{opts}, Options[ToExpression]]];
  Pt3 := Exp[-r (T - t)] S / (σ/2)^2 Pv3;
  Ct3 := Pt3 + 
    Which[
      r == δ, 
      Exp[-r (T - t)] / (T - t0) * Sbar,
      True, 
      Exp[-r (T - t)] / (T - t0) * Sbar,
      r == δ, 
      Exp[-r (T - t)] / (T - t0) * Sbar];
  (**)
/** Use PDF and ShawCallMellin */
Ct4 := Apply[ToExpression[algorithm],
Join[{S, K, r, δ, σ, T}, FilterRules[{opts}, Options[ToExpression[algorithm]]]]];
/**
/** use TW and RG *
Ct5 := Apply[ToExpression[algorithm], Join[{S, K, r, δ, σ, T, t, t0, Sbar},
FilterRules[{opts}, Options[ToExpression[algorithm]]]]];
/**
/** called by Schroder *
Cv6 := Apply[ToExpression[algorithm], Join[{v, h, q},
arguments, FilterRules[{opts}, Options[ToExpression[algorithm]]]]];
Ct6 := \[\frac{\exp[-r (T-t)] S}{T \cdot t0 \cdot (\sigma/2)^2}\] Cv6;
/**
/** called by FFT *
Cv7 := Apply[ToExpression[algorithm], Join[{\μ, v, h, q},
arguments, FilterRules[{opts}, Options[ToExpression[algorithm]]]]];
Ct7 := \[\frac{\exp[-r (T-t)] S}{T \cdot t0 \cdot (\sigma/2)^2}\] Cv7;
/**
vectorise[x_] := If[VectorQ[x], x, (x)];
method = vectorise[OptionValue[Method]];
algorithm = First[method];
Which[qi, call = Ct1,
Or[algorithm === "Euler", algorithm === "FW", algorithm === "Bromwich",
algorithm === "FT", algorithm === "GWR", algorithm === "UniG",
algorithm === "UniE", algorithm === "UniT", algorithm === "Laguerre"], call = Ct2,
(**)
algorithm === "Spectral3", parameters = {"M", "b"}; rules = {"M" \to 50, "b" \to 1};
arguments = parameters /. Join[Rest[method], rules]; call = Ct3,
algorithm === "Spectral1I", parameters = {"x"}; rules = {"x" \to 40};
arguments = parameters /. Join[Rest[method], rules]; call = Ct3,
algorithm === "PDE", call = Ct4,
algorithm === "ShawCallMellin", parameters = {"x"}; rules = {"x" \to 1000};
arguments = parameters /. Join[Rest[method], rules], call = Ct4,
Or[algorithm === "TW", algorithm = "RG"], call = Ct5,
algorithm === "Schroder", parameters = {"B", "K0", "Na", "Ns"};
rules = {"B" \to 1.454, "K0" \to 40, "Na" \to 16, "Ns" \to 235};
arguments = parameters /. Join[Rest[method], rules]; call = Ct6,
algorithm === "FFT", call = Ct7
];
call];
AsianOptionPrice[dS_, dK_, dr_, dδ_, dσ_, dT_, opts : OptionsPattern[]] :=
AsianOptionPrice[dS, dK, dr, dδ, dσ, dT, 0, 0, 0, opts];

1.2.2. AsianOptionCase
1.2.3. AsianOptionReferencePrice

Examples

Source code

AsianOptionReferencePrice[n_] := Module[{refprice},
  refprice[1] := 0.1931737902858918441110357107533227281.38;,
  refprice[2] := 0.246415690493387041872315275662885991.37;,
  refprice[3] := 0.30622036479436533301439682583932800762.37;,
  refprice[4] := 0.055986041544402069.17;,
  refprice[5] := 0.21838754655555802147720540275847646359.38;,
  refprice[6] := 0.172268741018016633570749057861043067601.39;,
  refprice[7] := 0.350095218965402034492743916779.31;,
  refprice[8] := 2.81586201560703126759064957764793782.36;,
  refprice[9] := 2.3108788872324218128318720323811490655.38;,
  refprice[10] := 1.879023661294478200965965122347068841.38;,
  refprice[14] := 2.697871537555891478875.22;,
  refprice[15] := 1.13474143222068091.20;,
  refprice[16] := 0.28532493868894175.17;,
  refprice[17] := 14.98395833573850641383863550353691471.38;,
  refprice[18] := 8.828758223806608576780233633780202979.37;,
  refprice[19] := 4.6967091321376386275825388491137724143.38;,
  refprice[n]]

1.2.4. AsianOptionComparison

Examples
Source code

AsianOptionComparison[case_, opts_List] :=
Module[{caselist, options, len, method, algorithm, settings, wp, finewp, candidate, 
    rounding, RoundSD, data, price, cpu, fineprice, finecpu, refprice, finereferenceprice, 
    comparison, refaccu, accu, prec, r1cell, r2cell, rowcell, row1, row2, row},
  caselist = Which[VectorQ[case], case, IntegerQ[case], (case), 
    True, "Incorrect format for the case number"];
  options = Which[ArrayDepth[opts] == 2, opts, VectorQ[opts], {opts}, True, opts];
  len = Length[options];
  method[i_] := method[i] = options[[i]];
  algorithm[i_] := algorithm[i] = Part[Method / method[i], 1];
  settings[i_] := settings[i] = Rest[Method / method[i]]; 
  wp[i_] := wp[i] = WorkingPrecision / method[i];
  finewp[i_] := 
    finewp[i] = If[wp[i] == WorkingPrecision \ wp[i] == MachinePrecision, "MP", wp[i]]; 
  candidate[i_] := candidate[i] = Join[{algorithm[i]}, settings[i], (finewp[i])];
(*---------*)
  rounding[x_] := Which[x < 0.1, N[Round[x, 10^-2], 1], x < 1, N[Round[x, 10^-2], 2], 
    x < 10, N[Round[x, 10^-2], 3], x < 100, N[Round[x, 10^-1], 3], True, Round[x];
  RoundSD[x_, i_] := Round[x, 10^(-i + Floor[RealExponent[x] + 1])];
  data[i_, j_] :=
    data[i, j] = Timing[Apply[AsianOptionPrice, Join[AsianOptionCase[j], method[i]])];
  price[i_, j_] := price[i, j] = data[i, j][[2]];
  cpu[i_, j_] := cpu[i, j] = data[i, j][[1]];
  fineprice[i_, j_] := fineprice[i, j] = N[Round[price[i, j], 5], 5];
  finecpu[i_, j_] := finecpu[i, j] = rounding[cpu[i, j]];
  refprice[i_, j_] := refprice[i] = AsianOptionReferencePrice[i];
  finereferenceprice[i_, j_] := finereferenceprice[i] = N[Round[refprice[i], 10], 10];
  comparison[i_, j_] := comparison[i, j] = MatchedSD[{refprice[i]}, {price[i, j]}];
  refaccu[i_, j_] := refaccu[i] = comparison[i, j][[1, 3, 3]];
  accu[i_, j_] := accu[i, j] = comparison[i, j][[1, 5, 3]];
  prec[i_] := prec[i] = comparison[i, caselist[[1]]][[1, 5, 2]];
  r1cell[i_] :=
    r1cell[i] = If[prec[i] === "MP", {candidate[i], SpanFromLeft, SpanFromLeft}, 
      {candidate[i], SpanFromLeft, SpanFromLeft, SpanFromLeft}];
  r2cell[i_] := r2cell[i] = If[prec[i] === "MP", ({"Accu", "Price", "CPU"}, 
      {"ED", "Accu", "Price", "CPU"}), 
    {rowcell[i, j] := rowcell[i, j] = If[prec[i] === "MP", {accu[i, j], fineprice[i, j], 
      finecpu[i, j]}, {prec[i], accu[i, j], fineprice[i, j], finecpu[i, j]}]; 
    row1 = Join["Case", "Reference", SpanFromLeft], 
    Flatten[Table[r1cell[i], {i, len}]]];
  row2 = Join[SpanFromAbove, "Accu", "Price"], Flatten[Table[r2cell[i], {i, len}]]];
  row[j_] :=
    Join[j, refaccu[j], finereferenceprice[j]], Flatten[Table[rowcell[i, j], {i, len}]]];
Grid[Join[{{row1}, {row2}, Table[row[j], {j, caselist}]}], 
Frame -> All, Alignment -> {Center, Center}]
]

\[ 1.2.5. \textbf{AsianOptionNormalisedParameters} \]

Examples
Source code

AsianOptionNormalisedParameters[S_, K_,
   ρ_, δ_, σ_, T_, t_: 0, t0_: 0, Sbar_: 0] := Module[{v, h, q},
   v = 2 (x - δ)/σ^2 - 1;
   h = (σ/2)^2 (T - t);
   q = (T - t0) (K - t - t0)/Sbar /
   (2/σ)^2 (K - t - t0/T - t0/Sbar);
   Grid[{{"v", "h", "q"}, {v, h, q}}, Frame -> All, Alignment -> {Center, Center}]
]
1.2.6. AsianOptionLaplaceTransform

Examples

Source code

Options[AsianOptionLaplaceTransform] = {WorkingPrecision -> Automatic};
AsianOptionLaplaceTransform[dS_, dK_, dr_, dδ_, dω_, dT_, dt_, dt0_: 0, dSbar_: 0,
  opts : OptionsPattern[]] := Module[{wp, S, K, r, δ, σ, T, t, t0, Sbar, n, h, q, μ},
  wp = OptionValue[WorkingPrecision];
  Which[
    wp === Automatic,
    S = Rationalize[dS]; (*price*)
    K = Rationalize[dK]; (*strike price*)
    r = Rationalize[dr]; (*interest rate*)
    δ = Rationalize[dδ]; (*dividend*)
    σ = Rationalize[dω]; (*volatility*)
    T = Rationalize[dT]; (*time to maturity*)
    t = Rationalize[dt]; (*current time*)
    t0 = Rationalize[dt0]; (*time to average*)
    Sbar = Rationalize[dSbar], (*realised price*)
    NumberQ[wp],
    S = SetPrecision[dS, wp];
    K = SetPrecision[dK, wp];
    r = SetPrecision[dr, wp];
    δ = SetPrecision[dδ, wp];
    σ = SetPrecision[dω, wp];
    T = SetPrecision[dT, wp];
    t = SetPrecision[dt, wp];
    t0 = SetPrecision[dt0, wp];
    Sbar = SetPrecision[dSbar, wp];
  ];
  ν = 2 (r - δ)/σ^2;
  h = (T - t); (*normalized time to maturity*)
  q = σ^2 T - t0/(4 S) (K - t - t0)/(T - t0); (*normalized strike price*)
  μ = Sqrt[2 λ ν^2];
  1/
  λ^2 - 2 λ (ν + 1) Power[2 q, -1/2] (μ - ν - 2]
  Gamma[1/2 (μ + ν + 4)] Hypergeometric1F1Regularized[1/2 (μ - ν - 2), μ + 1, -1/2 q]
];
AsianOptionLaplaceTransform[dS_, dK_, dr_, dδ_, dω_, dT_, dt_, dt0_: 0, dSbar_: 0,
  opts : OptionsPattern[]] :=
  AsianOptionLaplaceTransform[dS, dK, dr, dδ, dω, dT, 0, 0, 0, opts];
AsianOptionLaplaceTransform[dS_, dK_, dr_, dδ_, dω_, dT_, dt_, dt0_: 0, dSbar_: 0,
  opts : OptionsPattern[]] :=
  AsianOptionLaplaceTransform[dS, dK, dr, dδ, dω, dT, dt, 0, 0, opts];
AsianOptionLaplaceTransform[dS_, dK_, dr_, dδ_, dω_, dT_, dt_, dt0_: 0, dSbar_: 0,
  opts : OptionsPattern[]] :=
  AsianOptionLaplaceTransform[dS, dK, dr, dδ, dω, dT, dt, dt0, 0, opts];
Apply[AsianOptionLaplaceTransform, AsianOptionCase[1]]

\[
\frac{1}{-\frac{4\lambda}{5} + \lambda^2} \left[ \frac{1}{5} \left( \frac{1}{4} \sqrt{-2\lambda} \right)^3 \right] \left[ \frac{1}{38} \left( \frac{1}{\pi} \sqrt{2\lambda} \right)^3 \right] \Gamma \left( \frac{1}{2} \left( \frac{17}{5} + \frac{9}{25} + 2\lambda \right) \right)
\]

\[
\text{Hypergeometric1F1Regularized}\left[ \frac{1}{2} \left( \frac{7}{5} + \frac{9}{25} + 2\lambda \right), \frac{1}{2} + \frac{9}{25} + 2\lambda, \frac{38}{5} \right]
\]

Apply[AsianOptionLaplaceTransform, 
Join[AsianOptionCase[1], (WorkingPrecision \to Automatic)]]

\[
\frac{1}{-\frac{4\lambda}{5} + \lambda^2} \left[ \frac{1}{5} \left( \frac{1}{4} \sqrt{-2\lambda} \right)^3 \right] \left[ \frac{1}{38} \left( \frac{1}{\pi} \sqrt{2\lambda} \right)^3 \right] \Gamma \left( \frac{1}{2} \left( \frac{17}{5} + \frac{9}{25} + 2\lambda \right) \right)
\]

\[
\text{Hypergeometric1F1Regularized}\left[ \frac{1}{2} \left( \frac{7}{5} + \frac{9}{25} + 2\lambda \right), \frac{1}{2} + \frac{9}{25} + 2\lambda, \frac{38}{5} \right]
\]

### 1.3. Numerically Inverting Laplace Transforms

#### 1.3.1. Numerical Inversion of Laplace Transform Including 2D's

Examples

Source code

ClearAll[NInvertLaplaceTransform];
Options[NInvertLaplaceTransform] =

{Method \to Automatic, WorkingPrecision \to MachinePrecision};

NInvertLaplaceTransform[hat_, s_, t_, opts : OptionsPattern[]] :=
Module[{inioptions},
inrules, inoptions, pars, optionsR, inioptions = Cases[Options[Flatten[opts]], Method, String, (2, 4, 1)];
choice = Cases[Append[inioptions, "UniE"], "UniE" | "UniT" | "UniT" | "UniT" | "UniG" | "FT" | "GWR" | "Euler" | "FW" | "Bromwich" | "Laguerre"][1]];
Which[
choice == "UniE",
inoptions = Flatten[opts, Method \to ("M" \to 15), WorkingPrecision \to MachinePrecision];
pars = ("M"),
choice == "UniT",
inoptions = Flatten[opts, Method \to ("M" \to 15), WorkingPrecision \to MachinePrecision];
pars = ("M"),
choice == "UniG",
inoptions = Flatten[opts, Method \to ("M" \to 15), WorkingPrecision \to MachinePrecision];
pars = ("M"),
choice == "FW",
inoptions = Flatten[opts, Method \to ("M" \to 15), WorkingPrecision \to MachinePrecision];
pars = ("M"),
choice == "GWR",
inoptions = Flatten[opts, Method \to ("M" \to 15), WorkingPrecision \to MachinePrecision];
pars = ("M"),
choice == "Euler",
inoptions = Flatten[opts, Method \to ("m" \to 15, "n" \to 15, "y" \to 18), WorkingPrecision \to MachinePrecision];]
pars = {"m", "n", "γ"},
choice == "FW",
inioptions = Flatten[{opts,
    Method -> {"j" -> 10, "m" -> 10, "γ" -> 18}, WorkingPrecision -> MachinePrecision}];
pars = {"j", "m", "γ"},
choice == "Bromwich",
inioptions = Flatten[{opts, Method -> {"x" -> 1000}, WorkingPrecision -> MachinePrecision}];
pars = {"x"},
choice == "Laguerre",
inioptions = Flatten[{opts, Method -> {"m" -> 100, "k" -> 18, "γ" -> 25, "Shifting" -> 1},
    WorkingPrecision -> MachinePrecision}];
pars = {"m", "k", "γ"});
(*--------*)
optionsM := optionsM = FilterRules[inioptions, Method];
optionsR := optionsR = DeleteDuplicates[
    FilterRules[inioptions, Except[Method]], First[#!1] === First[#!2] &];
inirules := inirules = DeleteDuplicates[Cases[optionsM, _Rule, 3],
    First[#!1] === First[#!2] &];
inistrings := inistrings = DeleteDuplicates[Cases[optionsM, _String, {2, 3}]];
rules := rules = FilterRules[inirules, pars];
(*--------*)
args := args = pars /. rules;
subopts1 := subopts1 = FilterRules[inirules, Options[ToExpression[choice]]];
subopts2 := subopts2 = FilterRules[optionsR, WorkingPrecision];
subopts := subopts = Join[subopts1, subopts2];
ToExpression[choice] @@@ Join[{fhat, s, t}, args, subopts]
]
Options[NInverseLaplaceTransform] =
{Method -> Automatic, WorkingPrecision -> MachinePrecision};
NInverseLaplaceTransform[fhat2_, s_List, t_List, opts : OptionsPattern[]] :=
Module[{inichoice, choice, inioptions, pars, optionsM, optionsR,
inirules, inistrings, rules, args, subopts1, subopts2, subopts},
inichoice = Cases[FilterRules[Flatten[{opts}], Method], _String, {2, 3}, 1];
choice = Cases[Append[inichoice, "UniTG"], "UniTG" | "UniTT" | 
    "UniEG" | "UniET" | "UniTE" | "UniGT" | "UniGG" | "UniGE" | "UniGE"][[1]];
Which[
    choice == "UniTG",
    inioptions =
    Flatten[{opts, Method -> {"M" -> 15}, WorkingPrecision -> MachinePrecision}];
pars = {"M"},
choice == "UniTT",
inioptions =
    Flatten[{opts, Method -> {"M" -> 15}, WorkingPrecision -> MachinePrecision}];
pars = {"M"},
choice == "UniEG",
inioptions =
    Flatten[{opts, Method -> {"M" -> 15}, WorkingPrecision -> MachinePrecision}];
pars = {"M"},
choice == "UniET",
inioptions =
    Flatten[{opts, Method -> {"M" -> 15}, WorkingPrecision -> MachinePrecision}];
pars = {"M"},
choice == "UniTe",
inioptions =
    Flatten[{opts, Method -> {"M" -> 15}, WorkingPrecision -> MachinePrecision}];
pars = {"M"},
choice == "UniGT",
inioptions =
Flatten[{opts, Method -> {"M" -> 15}, WorkingPrecision -> MachinePrecision}];
pars = {"M"},
choice == "UniGG",
inoptions =
Flatten[{opts, Method -> {"M" -> 15}, WorkingPrecision -> MachinePrecision}];
pars = {"M"},
choice == "UniEE",
inoptions =
Flatten[{opts, Method -> {"M" -> 15}, WorkingPrecision -> MachinePrecision}];
pars = {"M"},
choice == "UniGE",
inoptions =
Flatten[{opts, Method -> {"M" -> 15}, WorkingPrecision -> MachinePrecision}];
pars = {"M"}];
(*-----*)
optionsM := optionsM = FilterRules[inoptions, Method];
optionsR := optionsR = DeleteDuplicates[
  FilterRules[{inoptions, Except[Method]], First[#1] === First[#2] &};
inirules := inirules = DeleteDuplicates[Cases[optionsM, _Rule, {3}],
  First[#1] === First[#2] &];
inistrings := inistrings = DeleteDuplicates[Cases[optionsM, _String, {2, 3}]];
rules := rules = FilterRules[inirules, pars];
(*-----*)
args := args = pars /. rules;
subopts1 := subopts1 = FilterRules[inirules, Options[ToExpression[choice]]];
subopts2 := subopts2 = FilterRules[optionsR, WorkingPrecision];
subopts := subopts = Join[subopts1, subopts2];
ToExpression[choice] @ Join[{fhat2, s, t}, args, subopts]

Alternative code

1.3.2. Euler: The Euler method (Abate and Whitt, 1995)

Examples

Source code

ClearAll[Euler]
Options[Euler] = {WorkingPrecision -> MachinePrecision, "Shifting" -> 0};
Euler[fhat_, s_, t0_, m_Integer: 15, n_Integer: 15, _Integer: 18, OptionsPattern[]] :=
Module[{wp, t, a, A, fn1, fn2, sn},
  wp = OptionValue[WorkingPrecision];
  If[wp == Automatic, 
    t = t0,
    t = SetPrecision[t0, wp];
  a = Rationalize[OptionValue["Shifting", 0];
  A = Log[1 + 10^a];
  fn1[k_] := fn1[k] = Re[fhat] /. s -> A + 2 k π ð t + a;
  fn2 := fn2 = Re[fhat] /. s -> A + 2 t + a;
  sn[k0_] := sn[k0] = 
  Exp[A/2] 2 t 
  (fn2 + 2 Sum[(-1)^k fn1[k], {k, 1, k0}]);
  Exp[a t] Sum[Binomial[m, k] 2^m sn[n + k], {k, 0, m}]
];

1.3.3. PW: The Post-Widder method (Abate and Whitt, 1995)

Examples
Source code

ClearAll[PW]
Options[PW] = {WorkingPrecision -> MachinePrecision, "Shifting" -> 0};
PW[fhat_, s_, t0_, j_Integer: 10, m_Integer: 10, γ_Integer: 18, OptionsPattern[]] :=
Module[{wp, t, α, r, fn1, fn2, fn3, fn4, sn, w},
  wp = OptionValue[WorkingPrecision];
  If[wp === Automatic,
    t = t0,
    t = SetPrecision[t0, wp];
  α = Rationalize[OptionValue["Shifting"], 0];
  r[n_] := r[n] = \left(\frac{10^{-\gamma}}{1 + 10^{-\gamma}}\right)^{\frac{n}{2}};
  fn1[n_] := fn1[n] = fhat /. \( s \rightarrow \frac{(n+1) (1-r[n])}{t} + \alpha; \)
  fn2[n_] := fn2[n] = (-1)^n fhat /. \( s \rightarrow \frac{(n+1) (1+r[n])}{t} + \alpha; \)
  fn3[n_, k_] := fn3[n, k] = (-1)^k Re[fhat] /. \( s \rightarrow \frac{n+1}{t} (1-r[n] \exp\left[\frac{\pi i k}{n}\right]) + \alpha; \)
  fn4[n_] := fn4[n] = Sum[fn3[n, k], {k, n - 1}];
  sn[n_] := sn[n] = \frac{n+1}{2 t n r[n]^n} (fn1[n] + fn2[n] + 2 fn4[n]);
  w[k_] := \omega[k] = (-1)^{n-k} \frac{k^n}{k! (m-k)!};
  Exp[α t] Sum[ω[k] sn[j k], {k, m}] ];

● 1.3.4. Laguerre: The Laguerre method (Abate Choudhury and Whitt, 1996)

Examples
Source code

ClearAll[Laguerre]
Options[Laguerre] =
  {WorkingPrecision -> MachinePrecision, "Shifting" -> 1, "Scaling" -> {0, 1}};

Laguerre[fhat_, s_, t0_, m_Integer, k_Integer: 18, 
  γ_?IntegerQ : 25, opts : OptionsPattern[]] := 
  Module[{a, scaling, σ, b0, wp, t, subwp, b, Q, r, qLaguerre, qLaguerrel, l, sfn, e},
    a = Rationalize[OptionValue["Shifting"], 0];
    scaling = Rationalize[OptionValue["Scaling"], 0];
    σ = scaling[[1]];
    b0 = scaling[[2]];
    wp = OptionValue[WorkingPrecision];
    If[wp == Automatic,
      t = t0; subwp = Precision[t]; b = If[subwp == Infinity, b0, SetPrecision[b0, subwp]],
      t = SetPrecision[t0, wp]; b = SetPrecision[b0, wp];
    ]
    (*Laguerre generating function Q(z)*)
    Q[z_] := Q[z] == b/(1 - z) fhat /. s -> (b (1 + z) + a b σ);
    r = Power[10, -Y/m];
    (*Laguerre coefficients q_n*)
    qLaguerre[fhat, 0, m, γ, opts] := Q[0];
    qLaguerre[fhat, n_?NonNegative, m, γ, opts] := qLaguerre[fhat, n, m, γ, opts] =
      1/m^n Q[r] + Sum[Re[Q[r Exp[I 2 π j/m]] Exp[-I n 2 π j/m]], (j, 1, m - 1)];
    (*associated Laguerre functions l_n(t)*)
    l[n_] := l[n] := Exp[-b t/2] Laguerrel[n, b t];
    (*Wynn's ε-algorithm*)
    Exp[a t] Sum[q[n] l[n], (n, 0, m - 1)]
  ];

Alternative code

1.3.5. Bromwich: The Bromwich integral (Shaw, 1998)

Examples
Source code

ClearAll[Bromwich]
Options[Bromwich] = Join["Contour" -> 5, MaxRecursion -> 18],
    FilterRules[Options[NIntegrate], Except[MaxRecursion, Method]]];
Bromwich[fhat_, s_, t0_, x_Integer: 1000, opts : OptionsPattern[]] :=
Module[{wp, t, inisubwp, subwp, inioptions, options, subopts, a, fn, u},
    wp = OptionValue[WorkingPrecision];
    If[wp === Automatic,
        t = t0;
        inisubwp = Precision[Exp[s t] fhat];
        subwp = If[inisubwp = Infinity,
            Print[Text["Style["Working Precision of 20 digits is used", Larger, Italic, Red]]]; 20,
            inisubwp];
        inioptions = Flatten[{WorkingPrecision -> subwp, opts, "Contour" -> 5, MaxRecursion -> 18}],
        t = Rationalize[t0, 0];
        inioptions = Flatten[
            {opts, "Contour" -> 5, MaxRecursion -> 18, WorkingPrecision -> MachinePrecision}]];
    options = DeleteDuplicates[inioptions, First[#1] === First[#2] &];
    a = Rationalize["Contour"/. options, 0];
    subopts = FilterRules[options, Except["Contour"]];
    fn[u_] := Re[Exp[s t] fhat] /. s -> a + I u;
    1/
        NIntegrate @@ Flatten[{fn[u], {{u, 0, x}}, subopts}, 1]
];

Alternative code

d 1.3.6. FT: The fixed Talbot method (Abate and Valko, 2004)

Details

Examples

Source code

ClearAll[FT]
Options[FT] = {WorkingPrecision -> MachinePrecision, "Shifting" -> 0];
FT[fhat_, s_, t0_, M_Integer: 16, OptionsPattern[]] :=
Module[{wp, t, a, r, s1, σ, t0, fn1, fn2, f},
    wp = OptionValue[WorkingPrecision];
    If[wp === Automatic,
        t = t0;
        t = SetPrecision[t0, wp]];
    a = Rationalize[OptionValue["Shifting"], 0];
    2 M
    r = ;
    5 t
    s1[θ_] /.; -π < θ < π := s1[θ] = r θ (Cot[θ] + i);
    σ[θ_] := σ[θ] = θ + (θ Cot[θ] - 1) Cot[θ];
    k π
    θ[k_] := θ[k] = ;
    M
    f = fhat /. s -> r + a;
    fn2[k_] := fn2[k] = fhat /. s -> s1[θ[k]] + a;
    1
    f = \[Integral] \[Integral] fn1[Exp[r t]] + Sum[Re[Exp[t. s1[θ[k]]]] fn2[k] (1 + i σ[θ[k]])], {k, 1, M - 1}];
    Exp[a t] f
];

Examples

Source code

```
ClearAll[GWR]
GWR[fhat_, s_, t0_, M_Integer: 16, OptionsPattern[]] := Module[{wp, t, α, fn, ρ},
If[OddQ[M],
  Print[Text[Style["M must be even", Larger, Italic, Red]]]; 0,
  wp = OptionValue[WorkingPrecision];
If[wp === Automatic,
  t = t0,
  t = SetPrecision[t0, wp]];
α = Rationalize[OptionValue["Shifting"], 0];
fn[k_] := fn[k] = Log[2] k
  Sum[(-1)^j Binomial[k, j] fhat /. s -> ((k + j) Log[2] + α), {j, 0, k}];
ρ[{-1, n_}] := ρ[{-1, n}] = 0;
ρ[0, n_] := ρ[0, n] = fn[n];
ρ[k_ /; k ≥ 1, n_] := ρ[k, n] = ρ[k - 2, n + 1] + k/(ρ[k - 1, n + 1] - ρ[k - 1, n]);
Exp[α t] ρ[M, 0]
];
```


Examples

Source code

```
ClearAll[UniG]
Options[UniG] = {WorkingPrecision -> MachinePrecision, "Shifting" -> 0};
UniG[fhat_, s_, t0_, M_Integer: 15, OptionsPattern[]] := Module[{wp, t, α, ζ, fn},
  wp = OptionValue[WorkingPrecision];
If[wp === Automatic,
  t = t0,
  t = SetPrecision[t0, wp]];
α = Rationalize[OptionValue["Shifting"], 0];
(*α=If[ExactNumberQ[α0], α0, SetPrecision[α0, wp]];*)
(*ζₖ are the Gaver-Stehfest weights*)
ζ[k_] := ζ[k] = (-1)^k ∑ (-1)^[j+1] Binomial[M, j] Binomial[2 j, j] Binomial[j, k - j],
  (j, Floor[(k + 1) / 2], Min[k, M])
Exp[α t] Log[2] / t ∑ ζ[k] fn[k], {k, 1, 2 M}
];
```
1.3.9. UniE: The unified Euler algorithm (Abate and Whitt, 2006)

Examples

Source code

ClearAll[UniE]
Options[UniE] = {WorkingPrecision -> MachinePrecision, "Shifting" -> 0};
UniE[fhat_, s_, t0_, M_Integer: 15, OptionsPattern[]] :=
Module[{wp, t, a, ξ, β, η, fn},
  wp = OptionValue[WorkingPrecision];
  If[wp === Automatic,
    t = t0,
    t = SetPrecision[t0, wp];
  ∂ηk are the Euler weights and βk are the Euler nodes∂
  a = Rationalize[OptionValue["Shifting"], 0];
  ξ[0] = 1/2;
  ξ[k_] /; 1 ≤ k ≤ M := ξ[k] = 1;
  ξ[k_] /; M + 1 ≤ k ≤ 2M - 1 := ξ[k] = ξ[k + 1] + 2^-M Binomial[M, 2M - k];
  ξ[2M] = 2M;
  β[k_] := β[k] = \frac{M \log[10]}{3} + \pi i k;
  η[k_] := η[k] = \frac{(-1)^k ξ[k]}{\sqrt{2}};
  fn[k_] := fn[k] = Re[fhat] /. s -> \frac{β[k]}{t} + a;
  Exp[α t] \frac{10^{M/3}}{t} \sum[η[k] fn[k], {k, 0, 2M}]
];
1.3.10. UniT: The unified Talbot algorithm (Abate and Whitt, 2006)

Examples

Source code

ClearAll[UniT]

Options[UniT] = {WorkingPrecision -> MachinePrecision, "Shifting" -> 0};

UniT[fhat_, s_, t0_, M_Integer: 15, OptionsPattern[]] := Module[{wp, t, a, δ, γ, fn},
    wp = OptionValue[WorkingPrecision];
    If[wp === Automatic,
        t = t0,
        t = SetPrecision[t0, wp];
    ]
a = Rationalize[OptionValue["Shifting"], 0];

(* γₖ are the Talbot weights and δₖ are the Talbot nodes *)

δ[0] = 2 M/5;

δ[k_] /; 1 ≤ k ≤ M - 1 := δ[k] = 2 k π/5 (Cot[k π/M] + δ);

γ[0] = 1/2 Exp[δ[0]];

γ[k_] /; 1 ≤ k ≤ M - 1 := γ[k] = (1 + k π M/4 (1 + Cot[k π/M])) - δ Cot[k π/M] Exp[δ[k]];

fn[k_] := fn[k] = fhat /. s → Exp[α t] t δ[k] + a;

Exp[a t] 2 Sum[Re[γ[k] fn[k]], {k, 0, M - 1}]

];

1.3.11. UniTG: The unified 2-D TG algorithm (Abate and Whitt, 2006)

Examples
Source code

ClearAll[UniTG]
Options[UniTG] = {WorkingPrecision -> MachinePrecision};
UniTG[fhat2_, s_List, t_List, M_Integer: 15, OptionsPattern[]] :=
Module[{s1, s2, t10, t20, wp, t1, t2, δ, γ, ξ, fn},
   s1 = s[[1]]; s2 = s[[2]]; 
   t10 = t[[1]]; t20 = t[[2]]; 
   wp = OptionValue[WorkingPrecision];
   If[wp === Automatic, 
     t1 = t10; t2 = t20, 
     t1 = SetPrecision[t10, wp]; t2 = SetPrecision[t20, wp]];

(*γ k are the Talbot weights and δ k are the Talbot nodes*)
δ[k_ /; 0 < k < M] := δ[k] = \(\frac{2 k \pi}{5} \) (Cot[k π / M] + 1);
γ[0] = \(\frac{1}{2} \) \(\text{Exp}[\delta[0]]\);
γ[k_ /; 0 < k < M] := γ[k] = \((1 + i (k π / M) (1 + Cot[k π / M]^2) - i Cot[k π / M]) \) \(\text{Exp}[\delta[k]]\);

(*ξ k are the Gaver-Stehfest weights*)
ξ[k_] := ξ[k] = (-1)^M k \[Sum\] \(\frac{\text{Binomial}[M, j] \text{Binomial}[2 j, j] \text{Binomial}[j, k - j]}{M!}\)
   \((j, \text{Floor}[\frac{(k + 1) / 2, \text{Min}[k, M]]\));

fn[k1_, k2_] := fn[k1, k2] = fhat2 /.
   \{s1 \[Rule] \\\frac{\delta[k1]}{t1}, s2 \[Rule] \\\frac{k2 \text{Log}[2]}{t2}\};

2 Log[2]
   \[Sum\]\[Re]γ[k1] ξ[k2] fn[k1, k2], \(\{k1, 0, M - 1\}, \{k2, 1, 2 M\}\)
   \[Sum\]\[Re]γ[k1] ξ[k2] fn[k1, k2], \(\{k1, 0, M - 1\}, \{k2, 1, 2 M\}\)
]

Alternative code


Examples
Source code

ClearAll[UniTT]
Options[UniTT] = {WorkingPrecision -> MachinePrecision};
UniTT[fhat2_, s_List, t_List, M_Integer : 15, OptionsPattern[]] :=
Module[{s1, s2, t10, t20, wp, t1, t2, δ, γ, afn, bfn},
  s1 = s[[1]]; s2 = s[[2]];
  t10 = t[[1]]; t20 = t[[2]];
  wp = OptionValue[WorkingPrecision];
  If[wp === Automatic,
    t1 = t10; t2 = t20,
    t1 = SetPrecision[t10, wp]; t2 = SetPrecision[t20, wp]];
  (*γk are the Talbot weights and δk are the Talbot nodes*)
  δ[0] = 2 M/5;
  δ[k_] /; 0 < k < M := δ[k] = 2 k π / 5 (Cot[k π / M] + δ);
  γ[0] = 1/2 Exp[δ[0]];
  γ[k_] /; 0 < k < M := γ[k] = (1 + i (k π / M) (1 + Cot[k π / M]^2) - i Cot[k π / M]) Exp[δ[k]];
  afn[k1_, k2_] := afn[k1, k2] = fhat2 /.
    {s1 -> δ[k1] / t1, s2 -> δ[k2] / t2};
  bfn[k1_, k2_] := bfn[k1, k2] = fhat2 /.
    {s1 -> δ[k1] / t1, s2 -> Conjugate[δ[k2]] / t2};
  2

Examples
Source code

ClearAll[UniEG]
Options[UniEG] = (WorkingPrecision -> MachinePrecision);
UniEG[fhat2_, s_List, t_List, M_Integer : 15, OptionsPattern[]] :=
  Module[{s1, s2, t10, t20, wp, t1, t2, ξ, β, η, ξ, fn},
    s1 = s[[1]]; s2 = s[[2]]; t10 = t[[1]]; t20 = t[[2]]; wp = OptionValue[WorkingPrecision];
    If[wp === Automatic, t1 = t10; t2 = t20, t1 = SetPrecision[t10, wp]; t2 = SetPrecision[t20, wp]];
    (η, are the Euler weights and β, are the Euler nodes)
    ξ[0] = 1/2;
    ξ[k_] /; 1 ≤ k ≤ M := ξ[k] = 1;
    ξ[k_] /; M + 1 ≤ k ≤ 2 M - 1 :=
      ξ[k] = ξ[k + 1] + 2^M Binomial[M, 2 M - k];
    ξ[2 M] = 1/2^n;
    β[k_] := β[k] = M Log[10]/3 + π * k;
    η[k_] := η[k] = (-1)^k ξ[k];
    (*ξ, are the Gaver-Stehfest weights*)
       (j, Floor[(k + 1)/2], Min[k, M])] ;
    fn[k1_, k2_] := fn[k1, k2] = fhat2 /. 
      {s1 -> ξ[k1]/t1, s2 -> ξ[k2]/t2};
    10^(M^3 Log[2]) Sum[η[k1] ζ[k2] Re[fn[k1, k2]], {k1, 0, 2 M}, {k2, 1, 2 M}]
  ];


Examples
Source code

ClearAll[UniET]
Options[UniET] = {WorkingPrecision -> MachinePrecision};
UniET[fhat2_, s_List, t_List, M_Integer: 15, OptionsPattern[]] :=
Module[{s1, s2, t10, t20, wp, t1, t2, ξ, β, η, γ, afn, bfn},
  s1 = s[[1]]; s2 = s[[2]]; 
  t10 = t[[1]]; t20 = t[[2]]; 
  wp = OptionValue[WorkingPrecision];
  If[wp === Automatic,
    t1 = t10; t2 = t20,
    t1 = SetPrecision[t10, wp]; t2 = SetPrecision[t20, wp]];

(*ηh are the Euler weights and βh are the Euler nodes*)

ξ[0] = 1/2;
ξ[k_/; 1 <= k <= M] := ξ[k] = 1;
ξ[k_/; M < k < 2 M] := ξ[k] = ξ[k + 1] + 2^-M Binomial[M, 2 M - k];
ξ[2 M] = 1/2^M;

β[k_] := β[k] = M Log[10]/3 + π I k;
η[k_] := η[k] = (-1)^k ξ[k];

(*γh are the Talbot weights and δh are the Talbot nodes*)

δ[0] = 2 M/5;
δ[k_/; 0 < k < M] := δ[k] = 2 k π/5 Cot[π k/M] + 1;

γ[0] = 1/2 Exp[δ[0]];
γ[k_/; 0 < k < M] := γ[k] = 1/2 (1 + I Cot[π k/M] + 1 - I Cot[π k/M]) Exp[δ[k]];

afn[k1_, k2_] := afn[k1, k2] = fhat2 . {s1 -> β[k1]/t1, s2 -> δ[k2]/t2};
bfn[k1_, k2_] := bfn[k1, k2] = fhat2 . {s1 -> β[k1]/t1, s2 -> Conjugate[δ[k2]/t2]};

10^M/3 Sum[η[k1] Re[γ[k2] afn[k1, k2] + Conjugate[γ[k2]] bfn[k1, k2]], {k1, 0, 2 M}, {k2, 0, M - 1}]

1.3.15. UniTE: The unified 2-D TE algorithm (Abate and Whitt, 2006)

Examples
Source code

ClearAll[UniTE]
Options[UniTE] = (WorkingPrecision -> MachinePrecision);
UniTE[fhat2__, s_List, t_List, M_Integer: 15, OptionsPattern[]] :=
  Module[{s1, s2, t10, t20, wp, t1, t2, δ, γ, ξ, β, η, afn, bfn},
    s1 = s[[1]]; s2 = s[[2]];
    t10 = t[[1]]; t20 = t[[2]];
    wp = OptionValue[WorkingPrecision];
    If[wp === Automatic,
      t1 = t10; t2 = t20,
      t1 = SetPrecision[t10, wp]; t2 = SetPrecision[t20, wp];
    (*γk are the Talbot weights and δk are the Talbot nodes*)
    δ[0] = 2 M / 5;
    δ[k_ /; 0 < k < M] := δ[k] = 2 k π / 5 (Cot[k π / M] + δ);

    γ[0] = - Exp[δ[0]] / 2
    γ[k_ /; 0 < k < M] := γ[k] = (1 + i (k π / M) (1 + Cot[k π / M]^2) - i Cot[k π / M]) Exp[δ[k]];
    (*ηk are the Euler weights and βk are the Euler nodes*)
    ξ[0] = 1 / 2
    ξ[k_ /; 1 ≤ k ≤ M] := ξ[k] = 1;
    ξ[k_ /; M + 1 ≤ k ≤ 2 M - 1] := ξ[k] = ξ[k + 1] + 2^-k Binomial[M, 2 M - k];
    ξ[2 M] = 1 / 2^M;
    β[k_] := β[k] = M Log[10] / 3 + π i k;
    η[k_] := η[k] = (-1)^k ξ[k];
    afn[k1_, k2_] := afn[k1, k2] = fhat2 /. {s1 -> δ[k1] / t1, s2 -> β[k2] / t2};
    bfn[k1_, k2_] := bfn[k1, k2] = fhat2 /. {s1 -> δ[k1] / t1, s2 -> Conjugate[β[k2]] / t2};
    10^M / 5 Sum[Re[γ[k1] η[k2] (afn[k1, k2] + bfn[k1, k2])], (k1, 0, M - 1), (k2, 0, 2 M)]
  ];

1.3.16. UniGT: The unified 2-D GT algorithm (Abate and Whitt, 2006)

Examples
Source code

ClearAll[UniGT]
Options[UniGT] = 

UniGT[fhat2_ , s_List, t_List, M_Integer: 15, OptionsPattern[]] :=
Module[{s1, s2, t10, t20, wp, t1, t2, G, D, fn},
    s1 = s[[1]]; s2 = s[[2]]; 
    t10 = t[[1]]; t20 = t[[2]]; 
    wp = OptionValue[WorkingPrecision]; 
    If[wp === Automatic, 
        t1 = t10; t2 = t20, 
        t1 = SetPrecision[t10, wp]; t2 = SetPrecision[t20, wp]];
    (*G are the Gaver-Stehfest weights*)
    G[k_] := G[k] = (-1)^k M Sum[j^k \text{Binomial}[M, j] Binomial[2 j, j] Binomial[j, k - j], j = 0, Floor[(k + 1)/2], Min[k, M]];
    (*D are the Talbot weights and \delta are the Talbot nodes*)
    D[k_, m_] := D[k, m] = Which[k == 0, 2 m / 5, 0 < k < m, 2 k \pi / 5 (\text{Cot}[k \pi / m] + i)]; 
    Y[k_, m_] := Y[k, m] = Which[k == 0, 1 / 2 Exp[d[0, m]], 0 < k < m, 
    \{1 + i (k \pi / m) \{1 + \text{Cot}[k \pi / m]^2\} - i \text{Cot}[k \pi / m] \text{Exp}[\delta[k, m]\}, \}]; 
    fn[k1_, k2_] := fn[k1, k2] = fhat2 /. \{s1 \to \frac{k1 \text{Log}[2]}{t1}, s2 \to \frac{\delta[k2, 3 M]}{t2}\}; 
    2 Log[2] / 5 \text{Sum}[G[k1] \Re[Y[k2, 3 M] fn[k1, k2]], \{k1, \{1, 2 M\}, \{k2, 0, 3 M - 1\}] 
    ]
    
\[ \square \] 1.3.17. UniGG: The unified 2-D GG algorithm (Abate and Whitt, 2006)

Examples
Source code

ClearAll[UniGG]
Options[UniGG] = (WorkingPrecision -> MachinePrecision);
UniGG[fhat2_, s_List, t_List, M_Integer: 15, OptionsPattern[]] :=
Module[{s1, s2, t10, t20, wp, t1, t2, $\zeta$, fn},
   s1 = s[[1]]; s2 = s[[2]]; 
   t10 = t[[1]]; t20 = t[[2]]; 
   wp = OptionValue[WorkingPrecision];
   If[wp === Automatic, 
      t1 = t10; t2 = t20,
      t1 = SetPrecision[t10, wp]; t2 = SetPrecision[t20, wp]];
   (*$\zeta_k$ are the Gaver-Stehfest weights*)
   $\zeta[k_, m_] :=$
   (-1)^m k Sum[Binomial[m, j] Binomial[2 j, j] Binomial[j, k - j],
      {j, Floor[(k + 1)/2], Min[k, m]}];
   fn[k1_, k2_] := fn[k1, k2] = fhat2 /. {s1 -> $\frac{k1 \text{Log}[2]}{t1}$, 
   s2 -> $\frac{k2 \text{Log}[2]}{t2}$};
   Log[2]^2 Sum[$\zeta[k1, M] \zeta[k2, 2 M]$ fn[k1, k2], 
   {k1, 1, 2 M}, {k2, 1, 4 M}]
]

1.3.18. UniEE: The unified 2-D EE algorithm (Abate and Whitt, 2006)

Examples
Source code

ClearAll[UniEE]
Options[UniEE] = {WorkingPrecision -> MachinePrecision};
UniEE[fhat2_, s_List, t_List, M_Integer: 15, OptionsPattern[]] :=
  Module[{s1, s2, t10, t20, wp, t1, t2, ϵ, β, η, afn, bfn},
    s1 = s[[1]]; s2 = s[[2]];
    t10 = t[[1]]; t20 = t[[2]];
    wp = OptionValue[WorkingPrecision];
    If[wp === Automatic,
      t1 = t10; t2 = t20,
      t1 = SetPrecision[t10, wp]; t2 = SetPrecision[t20, wp]];
    (*ηk are the Euler weights and βk are the Euler nodes*)
    ϵ[0] = ϵ[1] = 1;
    ϵ[2] := ϵ[k_] /; 1 ≤ k ≤ M := ϵ[k] = 1;
    ϵ[2 M] = 1/(2^M);
    β[k_] := β[k] = M Log[10]/3 + π ÷ k;
    η[k_] := η[k] = (-1)^k ϵ[k];
    afn[k1_, k2_] := afn[k1, k2] = fhat2 /. {s1 -> β[k1], s2 -> β[k2], t1 -> t1, t2 -> t2};
    bfn[k1_, k2_] := bfn[k1, k2] = fhat2 /. {s1 -> β[k1], s2 -> Conjugate[β[k2]], t1 -> t1, t2 -> t2};
    10^(2 M/3) 
    2 t1 t2


Examples
Source code

ClearAll[UniGE]
Options[UniGE] = (WorkingPrecision -> MachinePrecision);
UniGE[fhat2_ , s_List, t_List, M_Integer: 15, OptionsPattern[]] :=
Module[{s1, s2, t10, t20, wp, t1, t2, ξ, ζ, β, η, fn},
  s1 = s[[1]]; s2 = s[[2]];
  t10 = t[[1]]; t20 = t[[2]];
  wp = OptionValue[WorkingPrecision];
  If[wp === Automatic,
    t1 = t10; t2 = t20,
    t1 = SetPrecision[t10, wp]; t2 = SetPrecision[t20, wp];
  ]
  ((*ξk are the Gaver-Stehfest weights*)
    {j, Floor[(k + 1) / 2], Min[k, M]}];
  (/*ηk are the Euler weights and βk are the Euler nodes*/
   η[k_, m_] := η[k, m] = Which[k == 0, 1, 1 ≤ k ≤ m, 1,
    m + 1 ≤ k ≤ 2 m - 1, ξ[k + 1, m] + 2^m Binomial[m, 2 m - k], k == 2 m, 1 / 2^m];
   β[k_, m_] := β[k, m] = m Log[10] / 3 + π i k;
   η[k_, m_] := η[k, m] = (-1)^k ξ[k, m];

   fn[k1_, k2_] := fn[k1, k2] = fhat2 /. {s1 -> k1 Log[2], s2 -> β[k2, 3 M]};
   10^wp Log[2] / t1 t2
   Sum[ξ[k1] η[k2, 3 M] Re[fn[k1, k2]], {k1, 1, 2 M}, {k2, 0, 6 M}] ]
]

1.4. The Spectral Series Representation

Examples
Source code

Options[SpectralS] = Join[(FindRootMethod → Automatic),
FilterRules[Options[FindRoot], Except[{Method}]]];
SpectralS[v_, h_, k_, m_, b0_, opts : OptionsPattern[]] :=
Module[{b, opt1, opt2, Nv, x, ptildex, p, d, qtilde, q, n, Pb},
   b = Rationalize[b0];
   opt1 = {Method → OptionValue[FindRootMethod]};
   opt2 = FilterRules[{opts}, Except[{FindRootMethod}]];
   Nv = If[
      v ≥ 0 ∨ FindMinimum[
         {WhittakerW[1 - v, x, 1/2, 2 b, x], 0 ≤ x < Abs[v]}, x][[1]] > 0,
      0, IntegerPart[Abs[v]/2] + 1];
   ptildex[n_] := ptildex[n] = x /. Solve[
      x (Log[4 N[b] x] - 1) = 2 π (n + 1/4),
      x, InverseFunctions → True][[1]];
   p[n_] := p[n] = Re[x] /. Apply[FindRoot, Join[
      {WhittakerW[1 - v, x, 1/2, 2 b, x], x, ptildex[n]}, opt1, opt2]];
   q[n] := q[n] = x /. Apply[FindRoot, Join[
      {WhittakerW[1 - v, x, 1/2, 2 b, x], x, ptildex[n]}, opt1, opt2]];
   n = n /. Apply[FindRoot, Join[
      {WhittakerW[1 - v, x, 1/2, 2 b, x], x, ptildex[n]}, opt1, opt2]];
   Pb = Sum[
       WhittakerW[-v + 3/2, i p[n]/2, 1/2, 2 k] WhittakerW[1 - v, i p[n]/2, 1/2, 2 b], \(n, 1, m\)] +
      Sum[
         Exp[-v^2 - q[n]^2/2] q[n] Gamma[(v + q[n])/2] (2 k)^(-v/2) Exp[-1/2 k 
         WhittakerW[-v + 3/2, q[n]/2, 1/2, 2 k] WhittakerW[1 - v, q[n]/2, 1/2, 2 b], \(n, 1, Nv\)];
   Re[Pb]
];

1.5. The Spectral Integral Representation

Examples
Source code

Options[SpectralI] = Join[{NIntegrateMethod -> Automatic},
  FilterRules[Options[NIntegrate], Except[{Method}]]];
SpectralI[v_, h_, k_, x_, opts : OptionsPattern[]] := Module[{opt1, opt2, fn, p, Pv},
  opt1 = (Method -> OptionValue[NIntegrateMethod]);
  opt2 = FilterRules[{opts}, Except[{NIntegrateMethod}]];
  fn = Apply[NIntegrate,
    Join[{Exp[-(v^2 + p^2)/2] (2 k)^v k^2 (1 + p)/2 Exp[-(1/4) k^2 Re[WhittakerW[-v/2, 1/2 k]]
      Abs[Gamma[v + p/2]^2 Sinh[p] p, (p, 0, x), opt1, opt2]]},
    If[v < 0, 1, 0] - 1/(2 Gamma[Abs[v]])
    (2 k Gamma[Abs[v], 1/2] - Gamma[Abs[v] - 1, 1/2 k]) +
    If[v < -4, 1, 0] Sum[Exp[-2 n (Abs[v] - n) h] (-1)^n (Abs[v] - 2 n)/2 n (n - 1) Gamma[1 + Abs[v] - n]
      (2 k)^v n]
    LaguerreL[n - 2, Abs[v] - 2 n, 1/2 k], {n, 2, IntegerPart[Abs[v]/2]}];
  Re[Pv]]

■ 1.6. Constructive Complex Analysis (Schroder 2008)

Error estimates

Options[SchroderErr] = (pB -> 1.22, pK -> 40, pNa -> 19, pNs -> 275, WorkingPrecision -> 20);
SchroderErr[price_, strike_, interest_, dividend_, volatility_, maturity_,
  currenttime_: 0, timetoaverage_: 0, realisedprice_: 0, OptionsPattern[]] :=
Module[{wp, S, K, r, o, t, t0, Sbar, v, h, q, r, μ, β, a, A, B, Na, Ns, K0, IE, IEC,
  Ha, Hb, CB, Rb, totalR, teD, teK, errRb, errtotalR, γRes, Δ, teA, asI, errasI, ws,
  teB, seI, erreI, totalI, errI1, errI2, errI3, errI4, errtotalI, errCv, errCt},
  wp = OptionValue[WorkingPrecision];
  S = If[MachineNumberQ[price], SetPrecision[price, wp], price];
  K = If[MachineNumberQ[strike], SetPrecision[strike, wp], strike];
  r = If[MachineNumberQ[interest], SetPrecision[interest, wp], interest];
  σ = If[MachineNumberQ[dividend], SetPrecision[dividend, wp], dividend];
  o = If[MachineNumberQ[volatility], SetPrecision[volatility, wp], volatility];
  T = If[MachineNumberQ[maturity], SetPrecision[maturity, wp], maturity];
  t = If[MachineNumberQ[currenttime], SetPrecision[currenttime, wp], currenttime];
  t0 = If[MachineNumberQ[timetoaverage], SetPrecision[timetoaverage, wp], timetoaverage];
  Sbar = If[MachineNumberQ[realisedprice],
    SetPrecision[realisedprice, wp], realisedprice];
  v = 2 (r - o)/o^2 - 1;
  h = (σ^2/2) (T - t);
  q = (T - t0)/(2/σ)^2 (K - t - t0)/(T - t0) Sbar;
  γ = 1/(2 π) Gamma[v + 4] Exp[-1/2 q] (2 q)^(v/2)
\[ \mu = \nu + 4; \quad \beta = \frac{1}{\sqrt{2q}}; \quad \alpha = \frac{1}{\sqrt{2h}}; \]

\{*
\xi 2a = \nu + 2; \xi 3a = -\nu - 2; \xi 4a = \nu + 2;
\xi 2b = \nu; \xi 3b = -\nu; \xi 4b = \nu;
(a, c) = (\xi 2, \xi 4) \text{ or } (\xi 3, -\xi 4);
*
\}

\{*
B > A \geq 0; \alpha > 0
*
\}

\text{A} = 0; \ B = \text{SetPrecision[OptionValue[pB], wp];}
\text{Na} = \text{OptionValue[pNa]}; \ \text{Ns} = \text{OptionValue[pNs]}; \ \text{K0} = \text{OptionValue[pK];}
\{*
\text{6.1.1 Iterated weighted error functions IE;}
\text{IE}[z, \alpha, \delta, A, B] \text{ depends on } J = (a, \delta, A, B); 
\}

\text{IE}[z, \delta] :=
\text{IE}[z, \delta] = \text{If} [z = 0, \frac{1}{\alpha} \left( (a B + \delta) \ Erfc[a B + \delta] - (a A + \delta) \ Erfc[a A + \delta] + \left( \frac{1}{\sqrt{\pi}} \right) \right]
\left( \text{Exp} \left[ -(a A + \delta)^2 \right] - \text{Exp} \left[ -(a B + \delta)^2 \right] \right) \right), \frac{1}{z} \left( \text{Exp} \left[ -z A \right] \ Erfc[a A + \delta] - \text{Exp} \left[ -z B \right] \ Erfc[a B + \delta] + \text{Exp} \left[ \left( \delta + \frac{z}{2a} \right)^2 - \delta^2 \right] \left( \text{Erfc} \left[ \frac{z}{2a} + \delta + a B \right] - \text{Erfc} \left[ \frac{z}{2a} + \delta + a A \right] \right) \right];
\{*
\text{6.1.2 Iterated weighted error functions IEC;}
\text{IEc}[z, \alpha, \delta, A, B] \text{ depends on } J_0 = (\alpha, \delta, B); 
\}

\text{IEc}[z, \delta] := \text{IEc}[z, \delta] = \text{If} [z = 0, \frac{1}{\alpha} \left( \text{Exp} \left[ -\left( a B + \delta \right)^2 \right] - (a B + \delta) \ Erfc[a B + \delta] \right), \frac{1}{z} \left( \text{Exp} \left[ -z B \right] \ Erfc[a B + \delta] - \text{Exp} \left[ \left( \delta + \frac{z}{2a} \right)^2 - \delta^2 \right] \text{Erfc} \left[ a B + \delta + \frac{z}{2a} \right] \right); 
\{*
\text{6.1.3 Hermite function coefficients a and b;}
a[a, \mu] \text{ and } b[\mu, k] \text{ for } H_{\mu} \text{ with large arguments in the right half-plane;}
\}

\text{Ha}[\mu_0, k_0] := \text{Ha}[\mu, k] = \frac{1}{2 \Gamma(k + \mu)} \left( \frac{k + \mu}{2} \right) (-2)^k ;
\text{Ha}[\mu, k] = \Gamma[(1 + \mu) / 2] \ Pochhammer[\mu / 2, k / 2] \frac{1}{\Gamma[(1 + \mu) / 2] \ Pochhammer[1 / 2, k / 2] \ (k / 2)!} ;

\text{Ha}[\mu, k_0] := \text{Ha}[\mu, k] = 2^\mu \Gamma[1 / 2] \ Pochhammer[\mu / 2, k / 2] \frac{1}{\Gamma[1 / 2] \ Pochhammer[1 / 2, k / 2] \ (k / 2)!} ;

\text{Hb}[\mu, k_0] := \text{Hb}[\mu, k] = (-1)^k \frac{\Gamma[(1 + \mu) / 2, (k - 1) / 2]}{\Gamma[(1 + \mu) / 2, (k - 1) / 2] \ (k - 1)!} ;

\text{Hb}[\mu, k] = (-1)^k \frac{\Gamma[(1 + \mu) / 2, (k - 1) / 2]}{\Gamma[(1 + \mu) / 2, (k - 1) / 2] \ (k - 1)!} ;

\{*
\text{6.1.4 The coefficients } \beta; 
\beta[n, k] 
\}

\text{C8}[n_0, k_0] := \text{C8}[n, k] = \left( \frac{1}{\text{Binomial}[n, k]} \right) \text{ for } n \geq k;
\[ R_b[a] := 2 \pi^a \frac{\Gamma((a + \mu) / 2)}{\Gamma(\mu)} \text{Hypergeometric1F1}[(a + \mu) / 2, a + 1, \beta^2] \];
\[ \text{totalR} := \frac{1}{2} (\exp[2 h (v + 1)] R_b[v + 2] - R_b[v]) ; \]
\[ \text{teD}[a] := \text{teD}[a] = \left(\frac{\sqrt{2}}{\beta}\right)^{a + \mu} \text{Sum}\left[ \frac{\text{Pochhammer}[-(\mu - a) / 2 + 1, n]}{\text{Pochhammer}[(\mu + a) / 2 + 1, n]} \frac{\exp[-(\mu + a + 2 n) B]}{(1 - \exp[-2 B] B^{-1})}, (n, 0, (\mu - a) / 2 - 1) \right] ; \]
\[ \text{teK}[a] := \text{teK}[a] = \frac{\Gamma(\mu / 2)}{\Gamma(\mu)} \frac{1 - \exp[-B\alpha]}{\text{teD}[a]} + \text{teD}[a] ; \]
\[ \text{errRb}[a] := \text{errRb}[a] = -4 \text{teK}[a] ; \]
\[ \text{errtotalR} := \frac{1}{2} (\exp[2 h (v + 1)] \text{errRb}[v + 2] - \text{errRb}[v]) ; \]
\[ \gamma\text{Res} := \gamma\text{Res} = \frac{1}{2 (v + 1)} (\exp[2 h (v + 1)] - 2 q (v + 1) - 1) ; \]
\[ \Delta := \Delta = -2 (\exp[2 h (v + 1)] \text{teK}[v + 2] - \text{teK}[v]) ; \]
\[ \gamma\text{xtotalR} = \gamma\text{totalR} ; \]
\[ \gamma\text{xerrtotalR} = \gamma\text{errtotalR} ; \]
\[ \gamma\text{Res} = \gamma\text{Res} ; \]
\[ \gamma\text{xDelta} = \gamma\Delta ; \]
\[ \text{asI}[a, c] := \text{asI}[a, c] = \text{Sum}\left[ \frac{(-1)^n H_b[\mu, n]}{\beta^{2 n + 1}} \text{Im}\left[ \exp\left[ \frac{i}{2} (\mu + a) \right] \text{teA}\left[ \frac{c}{2 a} + \frac{i}{2} - \mu, 2 n + \mu, a \right] \right], (n, 0, Na - 1) \right] ; \]
\[ \text{errasI}[a, c] := \text{errasI}[a, c] = \exp[(\alpha \pi)^2 / 4] \frac{\Gamma(\mu / 2)}{\Gamma(\mu)} \text{Re}\frac{\text{Res}[\mu]}{\text{Res}[\mu]} \frac{\epsilon}{2 a} ; \]
\[ \text{teB}[a, c, n] := \text{teB}[a, c, n] = \text{Sum}\left[ C[n, k] \left( \omega \left[ \frac{c}{2 a} + \frac{i}{2} - \mu, 2 n + \mu, a \right] + \omega \left[ \frac{c}{2 a} + \frac{i}{2} - \mu, 2 n + \mu, a - n + 2 k \right] \right) \right] ; \]
\[
\{k, 0, \text{IntegerPart}[n/2]\};
\]

\[
\text{seI}[a_, c_] := \text{seI}[a, c] = \text{Sum}[(\beta)^n \text{Ha}[\mu, n] \text{teB}[a, c, n], \{n, 0, \text{Ns} - 1\}];
\]

\[
\text{errseI}[a_, c_] :=
\text{errseI}[a, c] = \text{Exp}\left[(\alpha n)^2/4 \right] \text{Abs}[\text{Ha}[\text{Re}[\mu], \text{Ns}]] (\beta/2)^{\text{Ns}} \text{IE}\left[-\text{Re}[a] - \text{Ns}, c, 2 a\right];
\]

\[
(*)
\xi2a=\xi+2; \xi3a=-\xi-2; \xi4a=\xi+2;
\xi2b=\xi; \xi3b=-\xi; \xi4b=\xi;
\]

\[
\text{totalI} := \text{totalI} = \text{Exp}[2 h (\xi + 1)]
\]

\[
(\alpha I[\xi + 2, \xi + 2] + \text{seI}[\xi + 2, \xi + 2] + \text{seI}[-\xi - 2, -\xi - 2] + \text{seI}[-\xi - 2, -\xi - 2] -
\text{seI}[\xi, \xi] + \text{seI}[\xi, \xi] + \text{seI}[\xi - \xi, -\xi] + \text{seI}[\xi - \xi, -\xi])
\]

\[
\text{errI1} := \text{errI1} = \text{errseI}[\xi + 2, \xi + 2] + \text{errseI}[\xi + 2, \xi + 2];
\]

\[
\text{errI2} := \text{errI2} = \text{errseI}[-\xi - 2, -\xi - 2] + \text{errseI}[-\xi - 2, -\xi - 2];
\]

\[
\text{errI3} := \text{errI3} = \text{errseI}[\xi, \xi] + \text{errseI}[\xi, \xi];
\]

\[
\text{errI4} := \text{errI4} = \text{errseI}[-\xi - \xi, -\xi];
\]

\[
\text{errtotalI} := \text{errtotalI} = \text{Exp}[2 h (\xi + 1)] (\text{errI1} + \text{errI2}) - (\text{errI3} + \text{errI4});
\]

\[
(*
\text{Print}["y*errI1"," y errI1];
\text{Print}["y*errI2"," y errI2];
\text{Print}["y*errI3"," y errI3];
\text{Print}["y*errI4"," y errI4];
\text{Print}["y*errtotalI"," y errtotalI];
*)
\]

\[
\text{errCv} := \text{errCv} = \text{If}[q > 0, y (\text{errtotalR} + \text{errtotalI}), 0];
\]

\[
\text{errCt} := \text{errCt} = \frac{e^{-\xi (T-t)}}{T-t0} \frac{S}{(\sigma/2)^2} \text{errCv};
\]

\[
(*)
\text{Print}["y*errtotalR"," N[y errtotalR, 3]];\]
\text{Print}["y", N[y]];\]
\text{Print}["y*h10^4", N[y h10^4]];\]
\text{Print}["y*q10^4", N[y q10^4]];\]
\text{Print}["y", N[y, h, q]];\]
\text{Print}["y*errtotalI", N[y errtotalI, 3]];\]
\text{Print}["y*errtotalR", N[y errtotalR, 3]];\]
\text{Print}["y*errCv", N[y errCv, 3]];\]
\text{Print}["y*errCt", N[y errCt, 3]];\]
\]

\text{SchrodrrErr[price, strike, interest, dividend, volatility, maturity, OptionsPattern[]]} := \text{SchrodrrErr[price, strike, interest, dividend, volatility, maturity, 0, 0, 0, 0, pB \rightarrow \text{OptionValue}[pB], pK \rightarrow \text{OptionValue}[pK], pNa \rightarrow \text{OptionValue}[pNa], pNs \rightarrow \text{OptionValue}[pNs], WorkingPrecision \rightarrow \text{OptionValue}[WorkingPrecision]]};

\text{Options[SchrodrrB]} = \{\text{pB} \rightarrow 1.22, \text{pK} \rightarrow 40, \text{pNa} \rightarrow 19, \text{pNs} \rightarrow 275, \text{WorkingPrecision} \rightarrow 50\};

\text{SchrodrrB[price, strike, interest, dividend, volatility, maturity, \text{currenttime:} 0, \text{timeaverage:} 0, \text{realisedprice:} 0, \text{OptionsPattern[]}} :=
\text{Module}\left[\{\text{wp, S, K, r, } \delta, \sigma, T, t, 0, \text{Sbar, v, h, q, } \gamma, \mu, \beta, \alpha, A, B, Na, Ns, K0, IE, IEC, Ha, Hb, C\beta, Rb, totalR, teD, teK, errRb, errtotalR, yRes, A, teA, asI, errseI, ws, \right\]
teB, seI, errseI, totalI, errI1, errI2, errI3, errI4, errtotalI, errCv, errCt),
wp = OptionValue[WorkingPrecision];
S = If[MachineNumberQ[price], SetPrecision[price, wp], price];
K = If[MachineNumberQ[strike], SetPrecision[strike, wp], strike];
r = If[MachineNumberQ[interest], SetPrecision[interest, wp], interest];
δ = If[MachineNumberQ[dividend], SetPrecision[dividend, wp], dividend];
σ = If[MachineNumberQ[volatility], SetPrecision[volatility, wp], volatility];
T = If[MachineNumberQ[maturity], SetPrecision[maturity, wp], maturity];
t = If[MachineNumberQ[currenttime], SetPrecision[currenttime, wp], currenttime];
t0 =
If[MachineNumberQ[timetoaverage], SetPrecision[timetoaverage, wp], timetoaverage];
Sbar = If[MachineNumberQ[realisedprice],
SetPrecision[realisedprice, wp], realisedprice];
ν = 2 (r - δ) / σ^2 - 1;
h = (σ / 2)^2 (T - t);
q = T - t0 / (2 / σ)^2 S (K - t - t0 Sbar);
y = 1 / 2π ν + 1 / (ν + 1) (2 q)^1 / 2;
μ = ν + 4; β = 1 / √2 q; α = 1 / √2 h;
(*
ξ2a=ν+2;ξ3a=ν-2;ξ4a=ν+2;
ξ2b=ν;ξ3b=ν;ξ4b=ν;
(a,c)=(ξ2,ξ4) or (ξ3,ξ4);
*)
(*
B>A≥0;a>0
*)
A = 0; B = SetPrecision[OptionValue[pB], wp];
Na = OptionValue[pNa]; Ns = OptionValue[pNs]; K0 = OptionValue[pK];
(*
6.1.1 Iterated weighted error functions \text{IE}:
\text{IE}(z,a,δ,A,B) depends on J=(a,δ,A,B);
*)
\text{IE}(z_, δ_, _) :=
\text{IE}(z, δ) = If\left[z = 0, \frac{1}{α} \left((-a A + δ) \text{Erfc}[a A + δ] - (a A + δ) \text{Erfc}[a A + δ] + \left(1 / \sqrt{π}\right) \left(\text{Exp}[\left((-a A + δ)^2\right)] - \text{Exp}[\left((-a B + δ)^2\right)]\right)\right), \frac{1}{z} \left(\text{Exp}[\left(-z A\right)] \text{Erfc}[a A + δ] - \text{Exp}[\left(-z B\right)] \text{Erfc}[a B + δ] + \text{Exp}[\left(\frac{z}{2 α} + δ\right)^2 - δ^2] \left(\text{Erfc}\left[\frac{z}{2 α} + δ + a A\right] - \text{Erfc}\left[\frac{z}{2 α} + δ + a A\right]\right)\right]\right];
(*
6.1.2 Iterated weighted error functions \text{IEc};
\text{IEc}(z,a,δ,B) depends on J_{0}=(a,δ,B);
*)
\text{IEc}(z_, δ_, _) := \text{IEc}(z, δ) = If\left[z = 0, \frac{1}{α} \left(\text{Exp}[\left((-a B + δ)^2\right)] - (a B + δ) \text{Erfc}[a B + δ]\right), \frac{1}{z} \left(\text{Exp}[\left(-z B\right)] \text{Erfc}[a B + δ] - \text{Exp}[\left(\frac{z}{2 α} + δ\right)^2 - δ^2] \text{Erfc}\left[\frac{z}{2 α} + δ + \frac{z}{2 α}\right]\right]\right];
(*
6.1.3 Hermite function coefficients \text{a} and \text{b};
a[m,k] and b[m,k] for H_{m} with large arguments in the right half-plane;
*)
Ha[μ_, k_?IntegerQ /; μ > 0, k_?IntegerQ /; k ≥ 0] :=
Ha[μ, k] = 1/(2 Gamma[μ] Gamma[k/2]) Pochhammer[(μ + 1)/2, k/2] (1 - (-2)^k/k!);

Ha[μ_, k_?EvenQ /; k ≥ 0] := Ha[μ, k] =
2^μ Gamma[1/2] Pochhammer[μ/2, k/2] Gamma[μ] Pochhammer[1/2, k/2] (k/2)!;

Ha[μ_, k_?OddQ /; k ≥ 0] := Ha[μ, k] =
Pochhammer[(1 + μ)/2, (k - 1)/2] Gamma[μ] Pochhammer[1/2, k/2] (k/2)!;

Ha[μ_, k_?IntegerQ /; k ≥ 0] := Ha[μ, k] = (-1)^k Gamma[μ, 2 k/k!];

(*
6.1.4 The coefficients β;
β[n,k]
*)

Cβ[n_, k_?IntegerQ /; n ≥ 0, k_?IntegerQ /; k ≥ 0] :=
Cβ[n, k] = If[EvenQ[n] && k = n/2 Binomial[n, k/2], 1/2n Binomial[n, k]] /; n ≥ k;

(* Theorem 6.1;
Theorem 6.2; *)

Rb[a_] := 2 π β^a Gamma[(a + μ) / 2] Hypergeometric1F1[(a + μ) / 2, a + 1, β^2];

γRes := γRes = 1/(2 (v + 1)) (Exp[2 h (v + 1)] - 2 q (v + 1) - 1);

Δ := Δ = -2 (Exp[2 h (v + 1)] teK[v + 2] - teK[v]);

(* Theorem 6.4 An asymptotic expansion for I_s(B,∞); *)
teA[\delta, \lambda, a_] :=
teA[\delta, \lambda, a] = \text{Sum}([-1]^k \text{Binomial}[-\lambda, k] \text{IEC}[-\lambda - a + 2 k, \delta], (k, 0, 0, 0));

\text{asl}[a, c_] := \text{asl}[a, c] =
\text{Sum}\left(\frac{(-1)^n \text{Hb}[\mu, n]}{\beta^{2n+\mu}} \text{Im}\left[\text{Exp}\left(\frac{\pi i}{2} (\mu + a)\right) \text{teA}\left(\frac{c}{2} + \frac{\pi i}{2}, 2n+\mu, a\right)\right], (n, 0, \text{Na}-1)\right);

\text{errasI}[a, c_] := \text{errasI}[a, c] = \text{Exp}\left(\frac{(\alpha \pi)^2}{4}\right) \frac{\Gamma[\text{Re}[\mu]]}{\Gamma[\text{Re}[\mu] + \text{Im}[\beta]]} \text{IEC}\left[2 \text{Na} + \text{Re}[\mu] - \text{Re}[a], \frac{c}{2} \right];

(*
Theorem 6.6 A series for I_b(A,B);
*)
\text{ws}[\gamma, \eta_] := \text{ws}[\gamma, \eta] = \text{Im}\left[\text{Exp}\left(\frac{\pi i}{2}\right) \text{IE}[-\eta, \gamma]\right];

\text{teB}[a, c, n_] :=
\text{teB}[a, c, n] = \text{Sum}\left(\text{C30}[n, k] \left(\text{ws}\left[\frac{c}{2} + \frac{\pi i}{2}, a + n - 2 k\right] + \text{ws}\left[\frac{c}{2} - \frac{\pi i}{2}, a - n + 2 k\right]\right),
\right)
\text{(k, 0, \text{IntegerPart}[n/2])};

\text{sei}[a, c_] := \text{sei}[a, c] = \text{Sum}(\beta^n \text{Ha}[\mu, n] \text{teB}[a, c, n], (n, 0, \text{Ns}-1));

\text{errasI}[a, c_] :=
\text{errasI}[a, c] = \text{Exp}\left(\frac{(\alpha \pi)^2}{4}\right) \text{Abs}[\text{Ha}[\text{Re}[\mu]], \text{Ns}] \text{IE}\left[-\text{Re}[a] - \text{Ns}, \frac{c}{2} \right];

(*
\xi2a\varepsilon v+2; \xi3a\varepsilon -v-2; \xi4a\varepsilon v+2;
\xi2b\varepsilon v; \xi3b\varepsilon -v; \xi4b\varepsilon v;
*)
\text{totalI} := \text{totalI} = \text{Exp}[2 h (v+1)]
\left(\text{asl}[v+2, v+2] + \text{si}[v+2, v+2] + \text{asl}[-v-2, -v-2] + \text{si}[-v-2, -v-2] -
\text{asl}[v, v] + \text{si}[v, v] + \text{si}[-v, -v] + \text{si}[-v, -v]\right);

\text{errI1} := \text{errI1} = \text{errasI}[v+2, v+2] + \text{errasI}[v+2, v+2];
\text{errI2} := \text{errI2} = \text{errasI}[-v-2, -v-2] + \text{errasI}[-v-2, -v-2];
\text{errI3} := \text{errI3} = \text{errasI}[v, v] + \text{errasI}[v, v];
\text{errI4} := \text{errI4} = \text{errasI}[-v, -v] + \text{errasI}[-v, -v];
\text{errtotalI} := \text{errtotalI} = \text{Exp}[2 h (v+1)] (\text{errI1} + \text{errI2}) - (\text{errI3} + \text{errI4});

(*
Print[" \gammaeratorI1"", \gamma \text{errI1}];
Print[" \gammaeratorI2"", \gamma \text{errI2}];
Print[" \gammaeratorI3"", \gamma \text{errI3}];
Print[" \gammaeratorI4"", \gamma \text{errI4}];
Print["\gammaeratortotalI"", \gamma \text{errtotalI}];
*)
(*
Print[" \gamma totalI"", \gamma \text{totalI}];
*)
\text{errCv} := \text{errCv} = \text{If}[q > 0, \gamma (\text{errtotalR} + \text{errtotalI}), 0];
\text{errCt} := \text{errCt} = \frac{\text{exp}(-x (T-t)) \cdot S}{(T-t0)^2} \text{errCv};

(*
Print["\gammaeratortotalR"", \gamma \text{errtotalR}, 3];
Print["v\gamma", \text{N}[\gamma]];
)
Print["x10^4z",N[x 10^4]];  
Print["y10^4z",N[y 10^4]];  
*

d SchroderB[price_, strike_, interest_, dividend_, volatility_, maturity_,  
OptionsPattern[]] := SchroderB[price, strike, interest, dividend, volatility,  
maturity, 0, 0, 0, pb -> OptionValue[pB], pK -> OptionValue[pK], pNa -> OptionValue[pNa],  
pNs -> OptionValue[pNs], WorkingPrecision -> OptionValue[WorkingPrecision]];  

\textbf{Pricing}  

Options[SchroderSlow] = {WorkingPrecision -> 25};  
SchroderSlow[v_, h_, q_, BB_, K0_, Na_, Ns_, OptionsPattern[]] :=  
Module[{wp, \gamma, \mu, \beta, \alpha, A, B, IE, IEc, Ha, Hb, C\beta, Rb, totalR,  
\teD, \teK, errRb, errtotalR, \gammaRes, \Delta, \teA, asi, errasI, \omega, \teB,  
\seI, errseI, totalI, errI1, errI2, errI3, errtotalI, Cv},  
wp = OptionValue[WorkingPrecision];  
\gamma = \frac{1}{2 \pi} \frac{\Gamma[v+4]}{\nu+1} \exp\left(-\frac{1}{2 q} (2 q)^{v+1}\right);  
\mu = v+4; \beta = \frac{1}{\sqrt{2 q}}; \alpha = \frac{1}{\sqrt{2 h}};  
(*  
\xi_2 = v+2; \xi_3 = v-2; \xi_4 = v+2;  
\xi_2 = v; \xi_3 = v; \xi_4 = v;  
(a, c) = (\xi_2, \xi_4) or (\xi_3, -\xi_4);  
*)  
(*  
B > A >= 0; a > 0  
*)  
A = 0;  
B = SetPrecision[BB, wp];  
(*  
6.1.1 Iterated weighted error functions IE;  
IE[z,\alpha,\delta,A,B] depends on J=(\alpha,\delta,A,B);  
*)  
IE[z_, \delta_] :=  
IE[z, \delta] = If[z == 0, \frac{1}{\alpha} \left((a B + \delta) \text{erfc}[a B + \delta] - (a A + \delta) \text{erfc}[a A + \delta] + \left(\frac{1}{\sqrt{\pi}}\right) \exp[-(a A + \delta)^2] - \exp[-(a B + \delta)^2]\right),  
\frac{1}{z} \left(\text{erfc}[\frac{z}{2 \alpha} + \delta + a B] - \text{erfc}[\frac{z}{2 \alpha} + \delta + a A]\right)];  
(*  
6.1.2 Iterated weighted error functions IEc;  
IEc[z,\alpha,\delta,B] depends on J_0=(\alpha,\delta,B);  
*)  
IEc[z_, \delta_] :=  
IEc[z, \delta] = If[z == 0, \frac{1}{\alpha} \left(\frac{1}{\sqrt{\pi}} \exp[-(a B + \delta)^2] - (a B + \delta) \text{erfc}[a B + \delta]\right),  
\frac{1}{z} \left(\text{erfc}[\frac{z}{2 \alpha} + \delta + a B] - \text{erfc}[\frac{z}{2 \alpha} + \delta + a B]\right)];  
(*  
6.1.3 Hermite function coefficients a and b;  
a[\mu, k] and b[\mu, k] for \mu_\mu with large arguments in the right half-plane;  
*)
\[\text{Ha}[\mu, k] = \frac{1}{2 \Gamma[\mu] \Gamma[k+\mu]} \int_{0}^{1} x^{\mu-1} (1-x)^{k-1} \, dx;\]

\[\text{Ha}[\mu, k] = 2^{-\mu} \Gamma[\mu/2] \Gamma[k/2] \Gamma[1/2] \Gamma[(k+\mu)/2]/\Gamma[(k+\mu)/2];\]

\[\text{Ha}[\mu, k] = 2^{-\mu} \Gamma[\mu/2] \Gamma[k/2] \Gamma[1/2] \Gamma[(k+\mu)/2]/\Gamma[(k+\mu)/2];\]

\[\text{Hb}[\mu, k] = (-1)^k \frac{\Gamma[\mu, 2k]}{\Gamma[k]};\]

\[\text{C}^* (= \text{C}^*; n, k) = \begin{cases} 1 & \text{if } \text{Even}[n] \& k = n \geq k; \\ \frac{1}{2^{n-1}} \text{Binomial}[n, n/2] & \text{Binomial}[n, k] \end{cases};\]

\[\text{Rb}[a] = 2 \pi \beta^a \frac{\text{Hypergeometric1F1}[a + \mu/2]}{\Gamma[\mu] \Gamma[a + 1]};\]

\[\text{totalR} = \frac{1}{2} (\text{Exp}[2h(v+1)] \text{Rb}[v+2] - \text{Rb}[v]);\]

\[\text{teD}[a] = \left(\frac{\sqrt{2}}{\beta}\right)^a \sum \frac{\text{Pochhammer}[-(\mu-a)/2+1, n]}{\text{Pochhammer}[\mu+a/2+1, n]} \left(1 - \text{Exp}[\text{B}^{\mu-1}]ight) / (n, 0, (\mu-a)/2-1);\]

\[\text{teK}[a] = \frac{\Gamma[\mu/2]}{\Gamma[\mu]} \frac{1 - \text{Exp}[\text{B}a]}{a} + \text{teD}[a];\]

\[\text{errRb}[a] = -4 \text{teK}[a];\]

\[\text{errtotalR} = \frac{1}{2} (\text{Exp}[2h(v+1)] \text{errRb}[v+2] - \text{errRb}[v]);\]

\[\gamma_{\text{Res}} = \frac{1}{2} (\text{Exp}[2h(v+1)] - 2q(v+1) - 1);\]

\[\Delta = -2 (\text{Exp}[2h(v+1)] \text{teK}[v+2] - \text{teK}[v]);\]

\[\text{asI}[a, c] = \sum (-1)^k \text{Binomial}[-\lambda, k] \text{IEc}[\lambda-a+2k, \delta], \{k, 0, 0\};\]

\[\text{asI}[a, c] = \sum \frac{(-1)^k \text{Hb}[\mu, n]}{\beta^{2n+\mu}} \text{Im}[\left\{\frac{\pi}{2} (\mu+a) \right\} \text{teA} \left\{ \frac{c}{2a} + \frac{\pi}{2} (\mu+a, 2n+\mu), a \right\}, \{n, 0, 0\}, \{n, 0, \lambda-a-1\}];\]
\texttt{errasi[a_, c_] := errasi[a, c] = Exp[(\alpha \pi)^2 / 4] \frac{\text{Gamma}[\text{Re}[\mu]]}{\text{Abs}[\text{Gamma}[\mu]]} \Abs[\text{Hb}[\text{Re}[\mu], \text{Na}]] (\sqrt{2} / \beta)^{2 \text{Na-Re}[\mu]} \text{IEC}[2 \text{Na} + \text{Re}[\mu] - \text{Re}[a], \frac{c}{2 a}] ;}

\texttt{(* Theorem 6.6 A series for } I_n(A,B) ; *)

\texttt{\text{ws}[\gamma_n, \eta_n] := \text{ws}[\gamma, \eta] = \text{Im}[\text{Exp}[i \frac{\pi}{2}] \text{IE}[-\eta, \gamma]] ;}

\texttt{\text{teB}[a_, c_, n_] :=}

\texttt{\text{teB}[a, c, n] = \text{Sum}[C[n, k] \left( \text{ws}\left[ \frac{c}{2 a} + i \frac{\pi}{2}, a + n - 2 k \right] + \text{ws}\left[ \frac{c}{2 a} + i \frac{\pi}{2}, a - n + 2 k \right] \right),}

\texttt{(k, 0, \text{IntegerPart}[n / 2])] ;}

\texttt{\text{sel}[a_, c_] := sel[a, c] = \text{Sum}[(-\beta)^n \text{Ha}[\mu, n] \text{teB}[a, c, n], \{n, 0, \text{Ns} - 1\}] ;}

\texttt{\text{errsel}[a_, c_] :=}

\texttt{\text{errsel}[a, c] = \text{Exp}[\frac{(\alpha \pi)^2}{4}] \Abs[\text{Hb}[\text{Re}[\mu], \text{Ns}]] (\beta / 2)^{\text{Ns}} \text{IE}[-\text{Re}[a] - \text{Ns}, \frac{c}{2 a}] ;}

\texttt{(* \xi2a=v+2; \xi3a=-v-2; \xi4a=v+2; \xi2b=v; \xi3b=-v; \xi4b=v; *)}

\texttt{\text{totalI} := \text{totalI} = \text{Exp}[2 h (\nu+1)]}

\texttt{(\text{asI}[\nu+2, \nu+2] + \text{sel}[\nu+2, \nu+2] + \text{sel}[-\nu-2, -\nu-2] + \text{sel}[-\nu-2, -\nu-2] -}

\texttt{(\text{asI}[\nu, \nu] + \text{sel}[\nu, \nu] + \text{sel}[-\nu, -\nu] + \text{sel}[-\nu, -\nu]) ;}

\texttt{\text{errI1} := errI1 = errasi[\nu+2, \nu+2] + errsel[\nu+2, \nu+2] ;}


\texttt{errI3 := errI3 = errasi[\nu, \nu] + errsel[\nu, \nu] ;}

\texttt{errI4 := errI4 = errasi[-\nu, -\nu] + errsel[-\nu, -\nu] ;}

\texttt{errtotalI := errtotalI = \text{Exp}[2 h (\nu+1)] (\text{errI1 + errI2}) - (\text{errI3 + errI4}) ;}

\texttt{\text{CV} = \gamma (\text{totalR + totalI}) ;}

\texttt{Options[\text{Schroder}] = \{\text{WorkingPrecision} -> 25\} ; \text{Schroder}[\nu0_, \text{h0}_-, \text{q0}_-, \text{B0}_-, \text{K0}_-, \text{Na}_-, \text{Ns}_-, \text{OptionsPattern[]} :=}

\texttt{\text{Module}[(wp, v, h, q, \gamma, \mu, \beta, \alpha, A, B, IE, IEC, Ha, Hb, C[n], Rb,}

\texttt{\text{totalR, teB, teK, errRb, errtotalR, } \gamma \text{Res}, \Delta, \text{teA, asI, errasi, ws, teB,}

\texttt{selI, errselI, totalI, errI1, errI2, errI3, errI4, errtotalI, CV),}

\texttt{wp = \text{OptionValue}[\text{WorkingPrecision}] ;}

\texttt{v = \text{SetPrecision}[\nu0, wp] ;}

\texttt{h = \text{SetPrecision}[\text{h0}, wp] ;}

\texttt{q = \text{SetPrecision}[\text{q0}, wp] ;}

\texttt{\gamma = \frac{1}{2 \pi} \frac{\text{Gamma}[\nu+4]}{\nu+1} \text{Exp}\left[-\frac{1}{2 q}\right] (2 q)^{-1} ;}

\texttt{\mu = \nu + 4 ; \beta = \frac{1}{2 \sqrt{q}} ; \alpha = \frac{1}{\sqrt{2 h}} ;}

\texttt{(* \xi2a=v+2; \xi3a=-v-2; \xi4a=v+2; \xi2b=v; \xi3b=-v; \xi4b=v;}

\texttt{(a, c) = (\xi2, \xi4) or (\xi3, -\xi4) ;*)}

\texttt{(*}
B>A≥0; α>0
*)
A = 0;
B = SetPrecision[B0, wp];
(* 6.1.1 Iterated weighted error functions IE;  
IE[\(z, \alpha, \delta, A, B\)] depends on \(J=(\alpha, \delta, A, B)\);  
*)
IE[\(_z, \delta\)] :=
IE[\(z, \delta\)] = \[z = 0, \(1/(\alpha B + \delta) \text{ Erfc}[\alpha B + \delta] - (\alpha A + \delta) \text{ Erfc}[\alpha A + \delta] + \left(1/\sqrt{\pi}\right)\]
\left(\frac{\text{Exp}[-(\alpha A + \delta)^2] - \text{Exp}[-(\alpha B + \delta)^2]}{\text{Exp}[-\alpha B] \text{ Erfc}[\alpha A + \delta] - \text{Exp}[-\alpha B]}\right), \frac{1}{z} \left(\frac{\text{Exp}[-\alpha B] \text{ Erfc}[\alpha B + \delta] - \text{Exp}[\alpha B + \delta]}{\text{Exp}[\frac{\delta^2}{2\alpha} - \delta^2] \text{ Erfc}[\alpha B + \delta + \frac{\delta}{2\alpha}] - \text{Exp}[\frac{\delta^2}{2\alpha} + \alpha B] - \text{Exp}[\frac{\delta^2}{2\alpha} + \alpha A]}\right)\];
(* 6.1.2 Iterated weighted error functions IEC;  
IEC[\(_z, \delta\)] := IEC[\(z, \delta\)] = \[z = 0, \(1/(\alpha B + \delta) \text{ Erfc}[\alpha B + \delta]\]
\left(\frac{\text{Exp}[-\alpha B] \text{ Erfc}[\alpha B + \delta] - \text{Exp}[\alpha B + \delta]}{\text{Exp}[\frac{\delta^2}{2\alpha} - \delta^2] \text{ Erfc}[\alpha B + \delta + \frac{\delta}{2\alpha}] - \text{Exp}[\frac{\delta^2}{2\alpha} + \alpha B] - \text{Exp}[\frac{\delta^2}{2\alpha} + \alpha A]}\right)\];
(* 6.1.3 Hermite function coefficients a and b;  
a[\(\mu, k\)] and b[\(\mu, k\)] for \(H_{\mu, k}\) with large arguments in the right half-plane;  
*)
Ha[\(_\mu, k\) ? IntegerQ /; \(\mu > 0, k \_ ? IntegerQ /; k \geq 0\) :=
Ha[\(\mu, k\)] = \frac{1}{\text{2 Gamma}[\mu]} \text{ Gamma}
\left[\left(\frac{\mu + k}{2}\right)\right] (-2)^k ;\]
Ha[\(\mu, k\) ? EvenQ /; \(k \geq 0\) := Ha[\(\mu, k\)] = \frac{2^{-\mu} \text{ Gamma}[1/2]}{\text{Pochhammer}[\mu/2, k/2]} \frac{1}{\text{Pochhammer}[1 + \mu/2, k/2]} \left(\frac{k/2}{\text{Pochhammer}[1/2, k/2]}\right);\]
Ha[\(\mu, k\) ? OddQ /; \(k \geq 0\) := Ha[\(\mu, k\)] = \frac{2^{-\mu} \text{ Gamma}[1/2]}{\text{Gamma}[\mu/2]} \frac{1}{\text{Pochhammer}[3/2, (k-1)/2]} \left(\frac{(k-1)/2}{\text{Pochhammer}[3/2, (k-1)/2]}\right);\]
Hb[\(_\mu, k\) ? IntegerQ /; \(k \geq 0\) := Hb[\(\mu, k\)] = (-1)^k \frac{\text{Pochhammer}[\mu, 2 k]}{k!} ;\]
(* 6.1.4 The coefficients \(\beta;\)  
\(\beta[n, k]\)  
*)
C\(\beta[n, k] ? IntegerQ /; \(n \geq 0, k \_ ? IntegerQ /; k \geq 0\) :=
\(C\beta[n, k] = \text{If}[\text{EvenQ}[n] \& \& k = n, \frac{1}{2^{n-1}} \text{ Binomial}[n, \frac{n}{2}], \frac{1}{2^{n}} \text{ Binomial}[n, k]]\) /; n ≥ k;
(* Theorem 6.1;
Theorem 6.2;  
*)
Rb[\(_a\)] := Rb[a] = 2 \pi \beta^a \frac{\text{Gamma}[(a + \mu)/2]}{\text{Gamma}[\mu] \text{ Gamma}[a + 1]} \text{ Hypergeometric1F1}[(a + \mu)/2, a + 1, \beta^2];\]
totalR := totalR = \frac{1}{2} (\text{Exp}[2 h (v + 1)] \text{ Rb}[v + 2] - \text{Rb}[v]);
\[ \text{teD}[a_] := \frac{(\sqrt{2}/\beta)^\mu}{a + \mu} \]
\[
\text{Sum}\left[\frac{\text{Pochhammer}[-(\mu - a)/2 + 1, n] \text{Exp}[-(\mu + a + 2n) B]}{\text{Pochhammer}[(\mu + a)/2 + 1, n] (1 - \text{Exp}[-2B]^{-\mu-1})}, (n, 0, (\mu - a)/2 - 1)\right];
\]
\[ \text{teK}[a_] := \frac{\text{Gamma}[\mu/2] (1 - \text{Exp}[-B a])}{\text{Gamma}[\mu]} + \text{teD}[a]; \]
\[ \text{errRb}[a_] := \text{errRb}[a] = -4 \text{teK}[a]; \]
\[ \text{errtotalR} := \text{errtotalR} = \frac{1}{2} (\text{Exp}[2h(\nu + 1)] \text{errRb}[\nu + 2] - \text{errRb}[\nu]); \]
\]
\[ (\star) \]
\[ \text{Corollary 6.3 } \gamma_{\text{Res}}; \]
\[ (* ) \]
\[ \gamma_{\text{Res}} := \frac{1}{2 (\nu + 1)} (\text{Exp}[2h(\nu + 1)] - 2q(\nu + 1) - 1); \]
\[ \Delta := \Delta = -2 (\text{Exp}[2h(\nu + 1)] \text{teK}[\nu + 2] - \text{teK}[\nu]); \]
\[ (\star) \]
\[ \text{Theorem 6.4 An asymptotic expansion for } I_k(B, \infty); \]
\[ (* ) \]
\[ \text{teA}[\delta, \lambda, a_] := \text{teA}[\delta, \lambda, a] = \text{Sum}\left[(\nu)^k \text{Binomial}[\nu - \lambda, k] \text{IEC}[\nu - a + 2k, \delta], (k, 0, \text{K0})\right]; \]
\[ \text{asi[a_, c_] := asi[a, c] = } \text{Sum}\left[\frac{(-1)^k \text{Hb}[\mu, n]}{\beta^{2n+\mu}} \text{Im}\left[\text{Exp}\left[i \frac{\pi}{2}(\mu + a)\right] \text{teA}\left[\frac{c}{2\alpha} + i \frac{\pi}{2} - a, 2n+\mu, a\right]\right], (n, 0, \text{Na} - 1)\right]; \]
\[ \text{errasi[a_, c_] := errasi[a, c] = } \text{Exp}\left[(\alpha n)^2/4\right] \frac{\text{Gamma}[\text{Re}[\mu]]}{\text{Abs}[\text{Gamma}[\mu]]} \text{Abs}[\text{Hb}[\text{Re}[, \text{Na}]]] \left(\sqrt{2}/\beta\right)^{\nu a + \text{Re}[\mu]} \text{IEC}[2\nu + \text{Re}[\mu] - \text{Re}[a], \frac{c}{2\alpha}]; \]
\[ (\star) \]
\[ \text{Theorem 6.6 A series for } I_k(A, B); \]
\[ (\star) \]
\[ \text{ws}[\gamma_{e}, \eta_{e}] := \text{ws}[\gamma, \eta] = \text{Im}\left[\text{Exp}\left[i \frac{\pi}{2}\right] \text{IEC}[\nu e, \gamma]\right]; \]
\[ \text{teB[a_, c_, n_] := } \text{teB}[a, c, n] = \text{Sum}\left[\text{C}[\nu, k] \left(\text{ws}\left[\frac{c}{2\alpha} + i \frac{\pi}{2} - a, a + n - 2k\right] + \text{ws}\left[\frac{c}{2\alpha} + i \frac{\pi}{2} - a, a - n + 2k\right]\right), (k, 0, \text{IntegerPart}[n/2])\right]; \]
\[ \text{sei[a_, c_] := sei[a, c] = } \text{Sum}\left[(-\beta)^n \text{Ha}[\mu, n] \text{teB}[a, c, n], (n, 0, \text{Na} - 1)\right]; \]
\[ \text{errsei[a_, c_] := } \text{errsei}[a, c] = \text{Exp}\left[(\alpha n)^2/4\right] \text{Abs}[\text{Ha}[\text{Re}[\mu], \text{Na}]] \left(\beta/2\right)^\nu \text{IEC}[\nu a - \text{Na} - \text{Re}[a], \frac{c}{2\alpha}]; \]
\[ (\star) \]
\[ \text{totalI := totalI = } \text{Exp}[2h(\nu + 1)] \]
\[ (\text{asi}[\nu + 2, \nu + 2] + \text{sei}[\nu + 2, \nu + 2] + \text{asi}[\nu - 2, \nu - 2] + \text{sei}[\nu - 2, \nu - 2]) - (\text{asi}[\nu, \nu] + \text{sei}[\nu, \nu] + \text{asi}[\nu - \nu, \nu] + \text{sei}[\nu - \nu, \nu]); \]
\[ \text{errII := errII = } \text{errasi}[\nu + 2, \nu + 2] + \text{errsei}[\nu + 2, \nu + 2]; \]
\begin{verbatim}
errI2 := errI2 = errasI[-v_2, -v_2] + errseI[-v_2, -v_2];
errI3 := errI3 = errasI[v, v] + errseI[v, v];
errI4 := errI4 = errasI[-v, -v] + errseI[-v, -v];
errtotalI := errtotalI = Exp[2 h (v + 1)] (errI1 + errI2) - (errI3 + errI4);

Cv = y (totalR + totalI);
Cv]
\end{verbatim}

\begin{itemize}
  \item \textbf{Examples}

1.7. PDE Method (Vecer 2002)

\begin{verbatim}
Options[PDE] = Join[Options[NDSolve], {"CutoffPoints" -> (-1, 3/2)}];
PDE[dS_, dK_, dr_, do_, dt_, opts : OptionsPattern[]] :=
Module[{iniopts, fnopts, cutoff, a, b, S, K, r, \delta, \sigma, T, q, z0, sol, u},
iniopts = Flatten[{opts}];
fnopts = FilterRules[iniopts, Except["CutoffPoints"]];
cutoff = Rationalize[OptionValue["CutoffPoints"], 0];
a = cutoff[[1]]; b = cutoff[[2]]; S = Rationalize[dS, 0]; (\*price\)
K = Rationalize[dK, 0]; (\*strike\ price\)
r = Rationalize[dr, 0]; (\*interest\ rate\)
\delta = Rationalize[do, 0]; (\*dividend\)
\sigma = Rationalize[do, 0]; (\*volatility\)
T = Rationalize[dt, 0]; (\*time to maturity\)
q[t_] := \frac{1}{(r - \delta) T} \left( \text{Exp}\left[-\delta \left( T - t \right)\right] - \text{Exp}\left[-r \left( T - t \right)\right] \right);
z0 = q[0] - \text{Exp}\left[-r T \right] \frac{K}{S}; (\*Z_t=X_t/S_t;\*)
sol = NDSolve[
\left\{ D[u[t, z], t] + \frac{1}{2} \left( z - \text{Exp}\left[-\delta t\right] q[t]\right)^2 \sigma^2 D[u[t, z], (z, 2)] = 0, u[T, z] = \text{Max}[0, z],
\right\}
u[t, a \sigma T] = 0, u[t, b \sigma T] = b \sigma T}, {u, {t, 0, T}, (z, a \sigma T, b \sigma T)}, fnopts];
Su[0, z0] /. First[sol]
\end{verbatim}

Alternative code

Clear[μ, rules, A, B, f, NMax, funcarr, operator, newrules,

GeometricHypergeometric, stuff, UMellinRaw, UMellin, ShawCallMellin];

\[ \mu[\_\_\_, \_\_] := \sqrt{\nu^2 + 2\lambda}; \]

rules = \{A \to 1, B \to z + 1\};

f[t\_, s\_, a\_] := \((1 + z s)^{-a}\);

NMax = 30;

funcarr = NestList[Factor[(-t D[H, t])] & f[t, s, a], NMax - 1];

Operator[list_List] := Sum[list[[k]] * funcarr[[k]], \{k, 1, Length[list]\}];

newrules = \{t \to 2\alpha, s \to \frac{\mu + \nu}{2} + 1, a \to \frac{\mu - \nu - 1}{2}\};

GeometricHypergeometric[n\_] := Module[\{firstpart, secondpart, 

nonexp, ser, coeffs, invpowers, seriesdata, rawfunc, rez, reza\},

firstpart = Series[\[-A + B + \frac{1}{x} + A - \frac{1}{2}\] Log[1 + x A] - \(\frac{1}{x} + B - \frac{1}{2}\) Log[1 + x B], \{x, 0, n + 2\}];

secondpart = Series[

\[\text{Sum}\left[\text{BernoulliB}[2k, \frac{1}{2k(2k-1)}] x^{2k}\{1 + x A\^{1-2k} - (1 + x B)^{1-2k}\}, \{k, 1, 2n\}\right], \{x, 0, n + 2\}\];

nonexp = Normal[firstpart + secondpart]; Remove[firstpart, secondpart];

ser = Normal[Series[Exp[nonexp], \{x, 0, 0\}]]; Remove[nonexp];

coeffs = Map[Factor, CoefficientList[ser, x]]; Remove[ser];

invpowers = Table[s^\((1-k)\), \{k, 1, n + 1\}];

(*Print[coeffs*invpowers]*);

seriesdata = Map[CoefficientList[\#, z] & coeffsxinvpowers];*

Remove[coeffs];

rawfunc = Map[Factor, Map[Operator[\#] & seriesdata]]; Remove[\#];

rez = Apply[Plus, invpowers \times rawfunc]; Remove[rawfunc];

reza = rez / newrules; Remove[\#];

]

stuff = GeometricHypergeometric[12];

UMellinRaw[λ\_, μ\_, \_\_, \_\_] := stuff /\ x = \(\lambda - 2 (\nu + 1)\);

UMellin[λ\_, μ\_, \_\_, \_\_] := UMellinRaw[λ, μ, ν, α];

Options[ShawCallMellin] = \{WorkingPrecision \to MachinePrecision\};

ShawCallMellin[S\_, K\_, T\_, δ\_, \_\_, \_\_, \_\_, \_\_: 1000, OptionsPattern[\]] :=

Module[\{wp, t, t0, Sbar, h, ν, q, contour, res, int, Ct\},

wp = OptionValue[WorkingPrecision];

t = 0; t0 = 0; Sbar = 0;

h = \((\sigma / 2)\)^2 (T - t);

ν = \frac{2 (r - \delta)}{\sigma^2} - \frac{1}{2};

q = \frac{T - t0}{(2 / \sigma)^2 S} \left[ K - \frac{t - t0}{T - t0} Sbar \right];

contour = If[\nu > -0.9, Min[0.5, \nu + 1], 0.5];

res = If[\nu > -0.9, \(\frac{e^{(2r+2)h}}{2 \nu + 2}\), 0];

int = \frac{1}{\pi} \sum_{p \in \mathbb{Z}} \text{Im} \left[ \text{UMellin}[\text{contour} + \text{\mu}[\nu, \text{contour} + \text{\mu}[p, \nu, q] e^{(\text{contour} + \text{\mu}[p, h]},

(p, 0, x), MaxRecursion \to 18, WorkingPrecision \to wp];

\text{e}^{-\frac{x}{T-T0}} \times S

\text{Ct} = \frac{1}{T \sigma^2} \left[ \text{res} + \text{Re}[\text{int}]\right];

\text{Ct} \]
1.9. Approximation by the lognormal distribution (Turnbull and Wakeman 1992)

Options[TW] = (WorkingPrecision -> MachinePrecision);
TW[dS_, dK_, dr_, dδ_, do_, dt_, dt0_: 0, dSbar_: 0, OptionsPattern[]] :=
Module[{wp, S, K, x, δ, σ, T, t, t0, Sbar, τ, m1, m2, δA, oA, d1, d2, KA, Ct},
  wp = OptionValue[WorkingPrecision];
  S = Rationalize[dS]; (*price*)
  K = Rationalize[dK]; (*strike price*)
  x = Rationalize[dr]; (*interest rate*)
  δ = Rationalize[dδ]; (*volatility*)
  σ = Rationalize[do]; (*volatility*)
  T = SetPrecision[dt, wp]; (*time to maturity*)
  t = Rationalize[dt]; (*current time*)
  t0 = Rationalize[dt0]; (*time to average*)
  Sbar = Rationalize[dSbar]; (*realised price*)
  τ = Abs[t - t0];
  m1 = \(e^{(x - δ)} (T-t) - e^{(x - δ) t}\)
  \( (x - δ) (T - t - t)\)
  \(m2 = \frac{2 e^{(2 (x - δ + σ^2) (T-t))} + 2 e^{(2 (x - δ) + σ^2) (T-t - t) + 2 (x - δ + σ^2) (T-t) - T - t}}{T - t + 2 (x - δ + 2) (T-t) + oA} \)
  \(\delta A = x - \frac{\log[m1]}{T - t}\);
  \(oA = \sqrt{\frac{\log[m2]}{T - t} + \frac{(x - δ + \sigma^2) (T-t)}{2 (x - δ + \sigma^2) (T-t)}}\);
  d1[KK_] := \(\log[S/KK] + \frac{(x - δ + \frac{1}{2} oA^2) (T-t)}{oA \sqrt{T - t}}\);
  d2[KK_] := d1[KK] - oA \sqrt{T - t};
  KA = \(\frac{T - t}{T - t}\) Sbar;
  Ct = If[t ≥ t0, \(\exp[-δA (T-t)] CDF[NormalDistribution[], d1[K]] - \exp[-(x (T-t)) CDF[NormalDistribution[], d2[K]]]\),
  S Exp[-δA (T-t)] CDF[NormalDistribution[], d1[K]] - \(\exp[-x (T-t)] CDF[NormalDistribution[], d2[K]]\)]
1.10. Approximation by the Reciprocal Gamma Distribution (Milevsky and Posner 1998)

Remove[RG]
Options[RG] = {WorkingPrecision -> MachinePrecision};
RG[ds_, dr_, dt_, dδ_, dm_, dT_, dt_, t0_, Sbar_, 0, OptionsPattern[]] :=
Module[{wp, S, K, r, σ, T, t, t0, Sbar, m1, m2, α, β, Ct},
wp = OptionValue[WorkingPrecision];
S = Rationalize[ds]; (*price*)
K = Rationalize[dr]; (*strike price*)
r = Rationalize[dt]; (*interest rate*)
δ = Rationalize[dδ]; (*dividend*)
σ = Rationalize[dm]; (*volatility*)
T = SetPrecision[dT, wp]; (*time to maturity*)
t = Rationalize[dt]; (*current time*)
t0 = Rationalize[dt0]; (*time to average*)
Sbar = Rationalize[Sbar]; (*realised price*)
m1 = If[r ≠ δ, S (e^{(r - δ) (T - t)} - 1) / ((r - δ) (T - t)), S];
m2 = If[r ≠ δ, 2 S^2 / (T - t)^2,
2 S^2 (e^{α (T - t)} - 1 - α^2 (T - t)) / α^4];
α = m2 - m1;
β = m2 - m1;
Ct = e^{-2 (T - t)} - e^{-r (T - t)} / (r - δ) T 
CDF[GammaDistribution[α - 1, β], T - t / (T - t) Sbar - T / (T - t) Sbar];
]

1.11. Monte Carlo Simulation with Control Variate

Options[MonteCarloControlCall] =
(DailyReading -> 1, Replication -> 10000, ReferencePrice -> False);
MonteCarloControlCall[S_, K_, r_, δ_, σ_, T_, OptionsPattern[]] :=
Module[{dr, rp, samples, dt, tpath, dWtrial, Wtrial, Spathtrial, 
Sariavetrial, Sgeoavetrial, Varitrial, Vgeotrial, θ, δEff, σEff, d1, d2, geocall, dw, W, Spath, Sariave, Sgeoave, Vari, Vgeo, V, Ct, hM, 
error, interval, RefPrice, SigDigit, SigDigitQ, DisplayResult],
dr = OptionValue[DailyReading];
rp = OptionValue[Replication];
samples = OptionValue[ReferencePrice];
samples = Floor[dr x 365 x T];
T =
samples
tpath = Table[i, {i, dt, T, dt}];
dWtrial = \sqrt{dt} RandomReal[NormalDistribution[], {samples, 5000}];
Wtrial = Accumulate[dWtrial]; Clear[dWtrial];
Spathtrial = S e^{(r - δ - \sigma^2 / 2) tpath + \sigma Wtrial}; Clear[Wtrial];
Sariavetrial = Mean[Spathtrial];
Sgeoavetrial = GeometricMean[Spathtrial]; Clear[Spathtrial];
Varitrial = e^{-T} Map[Max[H - K, 0] &, Sariavetrial]; Clear[Sariavetrial];
Vgeotrial = e^{-T} Map[Max[H - K, 0] &, Sgeoavetrial]; Clear[Sgeoavetrial];
\[
\theta = \frac{\text{Covariance}[\text{Vartrial}, \text{Vgeotrial}]}{\text{Variance}[\text{Vgeotrial}]}; \text{Clear}[\text{Vartrial}, \text{Vgeotrial}];
\]

\[
\delta_{\text{Eff}} = \frac{1}{2} \left( x + \delta + \frac{1}{\sigma^2} \right);
\]

\[
\sigma_{\text{Eff}} = \frac{\sigma}{\sqrt{3}};
\]

\[
\log[S/K] + \left( x - \delta_{\text{Eff}} + \frac{1}{2} \sigma_{\text{Eff}}^2 \right) \cdot T
\]

\[
d_1 = \frac{\sigma_{\text{Eff}} \sqrt{T}}{\sigma_{\text{Eff}} \sqrt{T}};
\]

\[
d_2 = d_1 - \sigma_{\text{Eff}} \sqrt{T};
\]

ggeoall = 
S \cdot e^{-\delta_{\text{Eff}}^T} \cdot \text{CDF}[\text{NormalDistribution}[], d_1] - K \cdot e^{-\delta_{\text{Eff}}^T} \cdot \text{CDF}[\text{NormalDistribution}[], d_2];
\]

dw = \sqrt{dt} \cdot \text{RandomReal}[\text{NormalDistribution}[], \{\text{samples}, \text{rp}\}];
\]

\[
W = \text{Accumulate}[dw]; \text{Clear}[dw];
\]

\[
\text{Spath} = S \cdot e^{\left( x-\delta_{\text{Eff}}^T \cdot \text{path} \right) \cdot \sigma^2 \cdot w}; \text{Clear}[\text{path}, W];
\]

\[
\text{Sariave} = \text{Mean}[	ext{Spath}];
\]

\[
\text{Sgeoave} = \text{GeometricMean}[	ext{Spath}]; \text{Clear}[	ext{Spath}];
\]

\[
\text{Vari} = e^{-\delta_{\text{Eff}}^T} \cdot \text{Map}[	ext{Max}[\text{H} - K, 0] \&, \text{Sariave}]; \text{Clear}[	ext{Sariave}];
\]

\[
\text{Vgeo} = e^{-\delta_{\text{Eff}}^T} \cdot \text{Map}[	ext{Max}[\text{H} - K, 0] \&, \text{Sgeoave}]; \text{Clear}[	ext{Sgeoave}];
\]

\[
V = \text{Vari} + \theta \left( \text{ggeoall} - \text{Vgeo} \right); \text{Clear}[	ext{Vari}, \text{Vgeo}];
\]

\[
\text{Ct} = \text{Mean}[V];
\]

\[
\text{bM} = \text{StandardDeviation}[V]; \text{Clear}[V];
\]

\[
\text{error} = \text{bM} \cdot \sqrt{\text{rp}};
\]

\[
\text{interval} = \{\text{Ct} - 1.96 \cdot \text{error}, \text{Ct} + 1.96 \cdot \text{error}\};
\]

\[
\text{TableForm}[[\text{Ct}, \text{error}, \text{interval}]],
\]

\[
\text{TableHeadings} \rightarrow \{\text{None}, \{\text{"Price"}, \text{"Standard Error"}, \text{"Confidence Interval"}\}\}
\]

\[
\{\ast \text{TableForm}[[\text{Ct}, \text{error}, \text{interval}]],
\]

\[
\text{TableHeadings} \rightarrow \{\text{"Price"}, \text{"Standard Error"}, \text{"Confidence Interval"}\}, \text{None}\};
\]

\[
\text{RefPrice} = \text{OptionValue}[	ext{ReferencePrice}];
\]

\[
\text{SigDigit} = -\text{Log}[10, \text{Abs}[\text{RefPrice} - \text{N}[\text{Ct}]]];
\]

\[
\text{SigDigitQ} = \text{If}[\text{SigDigit} == \text{Indeterminate}, \text{\_}, \text{SigDigit}];
\]

\[
\text{DisplayResult} = \text{If}[\text{RefPrice} == \text{False}, \text{N}[\text{Ct}],
\]

\[
\{\text{N}[\text{Ct}], \text{SigDigitQ}\}, \text{TableHeadings} \rightarrow \{\text{None}, \{\text{"Price"}, \text{"SigDigit"}\}\}
\]

\[
\}
\]

2. Code in Chapter 3

2.1. Some functions

2.1.1. Characteristic function

\[
\text{Characteristic}[\text{expr}, x, t]
\]

gives the characteristic function of \text{expr}.

- The characteristic function of a function \( f(x) \) is defined to be \( \int_{-\infty}^{\infty} e^{itx} f(x) \, dx \).
- Assumptions and other options to \text{Integrate} can also be given in \text{Characteristic}.

\[
\text{Characteristic}[\text{expr}_1, x_, t_-, \text{opts} : \text{OptionsPattern}[]] := \text{Module}[[\text{fn1}, \text{fn2}],
\]

\[
\text{fn1} := \text{NIntegrate}[\text{Exp}[\text{I} \cdot t \cdot x] \cdot \text{expr}, \{x, -\infty, \infty\}, \text{opts}];
\]

\[
\text{fn2} := \text{Integrate}[\text{Exp}[\text{I} \cdot t \cdot x] \cdot \text{expr}, \{x, -\infty, \infty\}, \text{opts}];
\]

\[
\text{If}[	ext{\text{NumericQ}[x]}, \text{fn1}, \text{fn2}]
\];

Examples
2.1.2. Inverse characteristic function

InverseCharacteristic[expr, t, x]
gives the inverse characteristic function of expr.

- The inverse characteristic function of a function \( \varphi(t) \) is defined to be \( \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{itx} \varphi(t) \, dt \).
- Assumptions and other options to Integrate can also be given in InverseCharacteristic.

\[
\text{InverseCharacteristic}[\text{expr\_}, \text{t\_}, \text{x\_}, \text{opts\_} : \text{OptionsPattern[]} ] := \text{Module}[\{\text{fn1}, \text{fn2}\},
\begin{align*}
\text{fn1} & := \frac{1}{2\pi} \text{NIntegrate}[\text{Re}[\text{Exp}[-\text{I} \text{t} \text{x}] \text{expr}], \{\text{t}, -\infty, \infty\}, \text{opts}] ; \\
\text{fn2} & := \frac{1}{2\pi} \text{Integrate}[\text{Exp}[-\text{I} \text{t} \text{x}] \text{expr}, \{\text{t}, -\infty, \infty\}, \text{opts}] ; \\
\text{If}[\text{NumericQ}[\text{x}], \text{fn1}, \text{fn2}] 
\}
\]

Examples

2.1.3. Mellin transform

MellinTransform[expr, x, s]
gives the Mellin transform of expr.

- The Mellin transform of a function \( f(x) \) is defined to be \( \int_{0}^{\infty} x^{s-1} f(x) \, dx \).
- Assumptions and other options to Integrate can also be given in MellinTransform.

\[
\text{MellinTransform}[\text{expr\_}, \text{x\_}, \text{s\_} \text{opts\_} : \text{OptionsPattern[]} ] := \text{Module}[\{\text{fn1}, \text{fn2}\},
\begin{align*}
\text{fn1} & := \frac{1}{2\pi} \text{NIntegrate}[x^{s-1} \text{expr}, \{x, 0, \infty\}, \text{opts}] ; \\
\text{fn2} & := \text{Integrate}[x^{s-1} \text{expr}, \{x, 0, \infty\}, \text{opts}] ; \\
\text{If}[\text{NumericQ}[s], \text{fn1}, \text{fn2}] 
\}
\]

Examples

2.1.4. Entire functions

Entire[s]
gives the entire function \( \xi(s) \).

- The entire function \( \xi(s) \) is defined to be \( \frac{1}{2} s(s-1) \left( \frac{1}{2} \right)^{1/2} \Gamma\left( \frac{s}{2} \right) \zeta(s) \), where \( \zeta(s) \) is the Riemann zeta function.

\[
\text{Entire}[s] := \frac{1}{2} s(s-1) \text{Power}\left[ \frac{1}{\pi}, \frac{s}{2}, \Gamma\left[ \frac{s}{2} \right] \zeta[s] \right] ;
\]

Entire[s]
gives the entire function \( \xi(s) \).

- The entire function \( \xi(s) \) is defined to be \( \left( \frac{3}{2} \right)^{(s+1)/2} \Gamma\left( \frac{s+1}{2} \right) L_4(s) \), where \( L_4(s) \) is the Dirichlet L-function \( L(\chi, s) \) for the Dirichlet character \( \chi(n) \) with modulus 4 and index 2.

\[
\text{Entire}[s] := \text{Power}\left[ \frac{4}{\pi}, \frac{s+1}{2}, \Gamma\left[ \frac{s+1}{2} \right] \text{DirichletL}[4, 2, s] \right] ;
\]

2.1.5. Euler zigzag numbers \( A_n \)

EulerZigzagA[n]
gives the Euler zigzag number \( A_n \).

- The exponential generating function of the sequence \( A_n \) is \( \sec \theta + \tan \theta = \tan \left( \frac{\theta}{2} + \frac{\pi}{2} \right) = \sum_{n=0}^{\infty} A_n \frac{\theta^n}{n!} \).
- The explicit formula for the sequence \( A_n \) is \( A_n = \sum_{k=0}^{n} \sum_{l=0}^{k} \frac{k!}{2^l l!} \binom{k}{l} (-1)^{l} (k-2)^{-1} \binom{k}{l} \).
The numbers $A_{z,n}$ are called secant numbers owing to the expansion

$$\sec \theta = \sum_{n=0}^\infty A_{z,n} \frac{\theta^n}{(2n)!}.$$ 

The numbers $A_{z,n+1}$ are called tangent numbers owing to the expansion

$$\tan \theta = \sum_{n=0}^\infty A_{z,n+1} \frac{\theta^n}{(2n+1)!}.$$ 

**EulerZigzagA[n_] := I^{n-1} \text{Sum}\left[\text{Binomial}[k, j] \frac{(-1)^j (k - 2 j)^{n-1}}{2^k 1^k}, (k, 1, n + 1), (j, 0, k)\right];**

**Examples**

#### 2.2. Simulation of Brownian Motion, Meander and Excursion

**2.2.1. Simulation of Brownian motion**

Simulating Brownian motion

BrownianMotion[\(T, n\)]

generates an one-dimensional Brownian motion over \([0, T]\) with \(n\) sampling points.

BrownianMotion[\(T, \{n_1, n_2\}\)]

generates an \(n_2\)-dimensional Brownian motion over \([0, T]\) with \(n_1\) sampling points.

**BrownianMotion[\(T\_, \{n\_\}] := \text{Module}[\{\text{dim, n1, n2, dt, dB1, dB2, B1, B2, B}\},
\begin{align*}
\text{dim} &= \text{If}[\text{VectorQ}[n], n, \{n, 1\}]; \\
n1 &= \text{dim}[[1]]; \\
n2 &= \text{dim}[[2]]; \\
dt &= \frac{T}{n1}; \\
dB1 &= dB1 = \text{RandomVariate}[\text{NormalDistribution}[0, \sqrt{dt}], n1]; \\
dB2 &= dB2 = \text{RandomVariate}[\text{NormalDistribution}[0, \sqrt{dt}], \text{dim}]; \\
B1 &= B1 = \text{Prepend}[\text{Accumulate}[dB1], 0]; \\
B2 &= B2 = \text{Prepend}[\text{Accumulate}[dB2], \text{ConstantArray}[0, n2]]; \\
B &= \text{If}[n2 = 1, \text{B1}, \text{B2}];
\end{align*}**

Simulating Brownian motion path

BrownianMotionPath[\(T, n\)]

plots a path of one-dimensional Brownian motion over \([0, T]\) with \(n\) sampling points.

BrownianMotionPath[\(T, n, m\)]

plots paths of \(m\) independent one-dimensional Brownian motions over \([0, T]\) with \(n\) sampling points.

BrownianMotionPath[\(T, \{n_1, n_2\}\)]

plots a path of \(n_2\)-dimensional (up to 3) Brownian motion over \([0, T]\) with \(n_1\) sampling points.

BrownianMotionPath[\(T, \{n_1, n_2\}, m\)]

plots paths of \(m\) independent \(n_2\)-dimensional (up to 3) Brownian motions over \([0, T]\) with \(n_1\) sampling points.

**BrownianMotionPath[\(T\_, \{n\_, m\_: 1\}] := \text{Module}[\{\text{dim, n2, plot1, plot2, plot3}\},
\begin{align*}
\text{dim} &= \text{If}[\text{VectorQ}[n], n, \{n, 1\}]; \\
n2 &= \text{dim}[[2]]; \\
plot1 &= \text{plot1} = \text{ListLinePlot}[\text{Table}[\text{BrownianMotion}[T, n], \{i, m\}],
\text{DataRange} -> \{0, T\}, \text{AxesLabel} -> \{"x", "B(t)"\}]; \\
plot2 &= \text{plot2} = \text{ListLinePlot}[\text{Table}[\text{BrownianMotion}[T, n], \{i, m\}],
\text{AxesLabel} -> \{"x", "y"\}]; \\
plot3 &= \text{plot3} = \text{Graphics3D}[\text{Table}[\{\text{ColorData}[1, i], \text{Line}[\text{BrownianMotion}[T, n]]\},
\{i, 1, m\}], \text{Axes} -> \text{True}, \text{AxesLabel} -> \{"x", "y", "z"\}]; \\
\text{Which}[n2 = 1, \text{plot1}, n2 = 2, \text{plot2}, n2 = 3, \text{plot3}]
\end{align*}**
2.2.2. Simulation of Brownian bridge

Simulating Brownian bridge

BrownianBridge[n, x]

gives an one-dimensional Brownian bridge over [0, 1] with endpoint x using n sampling points.

BrownianBridge[n1, n2, x]

gives an n2-dimensional Brownian bridge over [0, 1] with each coordinate of the endpoint equal to x using n1 sampling points.

BrownianBridge[n1, n2, {x1, ..., xn}]

gives an n2-dimensional Brownian bridge over [0, 1] with endpoint (x1, ..., xn) using n1 sampling points.

BrownianBridge[n__, x__] :=
Module[{dim, n1, n2, dt, t1, t2, dB1, dB2, B1, B2, Bbr1, Bbr2, Bbr},
  dim = If[VectorQ[n, n, {n, 1}];
  n1 = dim[[1]];
  n2 = dim[[2]];
  dt = \[Fraction]\[1, n1];
  t1 := t1 = Range[0, 1, dt];
  t2 := t2 = Map[ConstantArray[n, n2] & , Range[0, 1, dt]];
  dB1 := dB1 = RandomVariate[NormalDistribution[0, \[Sqrt][dt]], n1];
  dB2 := dB2 = RandomVariate[NormalDistribution[0, \[Sqrt][dt]], dim];
  B1 := B1 = Prepend[Accumulate[dB1], 0];
  B2 := B2 = Prepend[Accumulate[dB2], ConstantArray[0, n2]];
  Bbr1 := Bbr1 = B1 + t1 (x - Last[B1]);
  Bbr2 := Bbr2 = B2 + t2 (ConstantArray[x - Last[B2], n1 + 1]);
  Bbr = If[n2 = 1, Bbr1, Bbr2]
]

Simulating Brownian bridge path

BrownianBridgePath[n, x]

plots a path of an one-dimensional Brownian bridge over [0, 1] with endpoint x using n sampling points.

BrownianBridgePath[n, x, m]

plots paths of m independent one-dimensional Brownian bridges over [0, 1] with endpoint x using n sampling points.

BrownianBridgePath[n1, n2, x]

plots a path of an n2-dimensional (up to 3) Brownian bridge over [0, 1] with each coordinate of the endpoint equal to x using n1 sampling points.

BrownianBridgePath[n1, n2, x, m]

plots paths of m independent n2-dimensional (up to 3) Brownian bridges over [0, 1] with each coordinate of the endpoint equal to x using n1 sampling points.

BrownianBridge[n1, n2, {x1, ..., xn}]

plots a path of an n2-dimensional Brownian bridge over [0,1] with endpoint (x1, ..., xn) using n1 sampling points.

BrownianBridge[n1, n2, {x1, ..., xn}, m]

plots paths of m independent n2-dimensional Brownian bridges over [0, 1] with endpoint (x1, ..., xn) using n1 sampling points.
BrownianBridgePath[n__, x__, m__:1] := Module[{dim, n2, plot1, plot2, plot3},
  dim = If[VectorQ[n], n, {n, 1}];
  n2 = dim[[2]]; 
  plot1 := plot1 = ListLinePlot[Table[BrownianBridge[n, x], {i, 1, m}], 
  DataRange -> {0, 1}, AxesLabel -> {"x", "\[ScriptScript]x"[t]"[t]"}];
  plot2 := plot2 = ListLinePlot[Table[BrownianBridge[n, x], {i, 1, m}], 
  AxesLabel -> {"x", "y"}];
  plot3 := plot3 = Graphics3D[Table[{ColorData[1, i], Line[BrownianBridge[n, x]]}, 
  {i, 1, m}], Axes -> True, AxesLabel -> {"x", "y", "z"}];
  Which[n2 == 1, plot1, n2 == 2, plot2, n2 == 3, plot3] 
]

Examples

2.2.3. Simulation of Brownian meander

Simulating Brownian meander

BrownianMeander[n]

gives an one-dimensional standard Brownian meander over [0, 1] using n sampling points.

BrownianMeander[n1, n2]

gives an n2-dimensional standard Brownian meander over [0, 1] using n1 sampling points.

BrownianBridge[n, r]

gives an one-dimensional restricted Brownian meander over [0, 1] with endpoint r using n sampling points.

BrownianBridge[n1, n2, r]

gives an n2-dimensional restricted Brownian meander over [0, 1] with each coordinate of the endpoint equal to r using n1 sampling points.

BrownianMeander[n1, n2, {r1, ..., r[n_]}]

gives an n2-dimensional restricted Brownian meander over [0, 1] with endpoint (r1, ..., r[n]) using n1 sampling points.

BrownianMeander[n__, r__] :=
Module[{dim, n1, n2, r0, dt, t1, t2, temp1, temp2, r1, r2, Bme1, Bme2, Bme},
  dim = If[VectorQ[n], n, {n, 1}];
  n1 = dim[[1]]; 
  n2 = dim[[2]]; 
  dt = 1/n1;
  t1 := t1 = Range[0, 1, dt];
  t2 := t2 = Map[ConstantArray[m, n2] &, Range[0, 1, dt]];
  temp1 := If[r === Null, Sqrt[2 RandomVariate[ExponentialDistribution[1]], r];
  temp2 := If[r === Null, Sqrt[2 RandomVariate[ExponentialDistribution[1], n2]], r];
  r1 := r1 = temp1;
  r2 := r2 = ConstantArray[temp2, n1 + 1];
  Bme1 := Bme1 = Power[
  (r1 t1 + BrownianBridge[n, 0])^2 + BrownianBridge[n, 0]^2 + BrownianBridge[n, 0]^2, 1/2];
  Bme2 := Bme2 = Power[(r2 t2 + BrownianBridge[n, 0])^2, 
  BrownianBridge[n, 0]^2 + BrownianBridge[n, 0]^2, 1/2];
  Bme = If[n2 == 1, Bme1, Bme2] ]

Simulating Brownian meander path

BrownianMeanderPath[n]
plots a path of an one-dimensional standard Brownian meander over [0, 1] using \( n \) sampling points.

\[
\text{BrownianMeanderPath}[n, m]
\]

plots paths of \( m \) independent one-dimensional standard Brownian meanders over [0, 1] using \( n \) sampling points.

\[
\text{BrownianMeanderPath}[[n_1, n_2]]
\]

plots a path of an \( n_2 \)-dimensional (up to 3) standard Brownian meander over [0, 1] with using \( n_1 \) sampling points.

\[
\text{BrownianMeanderPath}[[n_1, n_2], m]
\]

plots paths of \( m \) independent \( n_2 \)-dimensional (up to 3) standard Brownian meanders over [0, 1] with using \( n_1 \) sampling points.

\[
\text{BrownianMeanderPath}[n, r, m]
\]

plots paths of \( m \) independent one-dimensional restricted Brownian meanders over [0, 1] with endpoint \( r \) using \( n \) sampling points.

\[
\text{BrownianBridgePath}[[n_1, n_2], r, m]
\]

plots paths of \( m \) independent \( n_2 \)-dimensional (up to 3) restricted Brownian meanders over [0, 1] with each coordinate of the endpoint equal to \( r \) using \( n_1 \) sampling points.

\[
\text{BrownianBridge}[[n_1, n_2], [r_1, ..., r_n], m]
\]

plots paths of \( m \) independent \( n_2 \)-dimensional (up to 3) restricted Brownian meanders over [0, 1] with endpoint \((r_1, ..., r_n)\) using \( n_1 \) sampling points.

\[
\text{BrownianMeanderPath}[n_\_, r_\_, m_\_]:=\text{Module}[\{\text{dim}, n_2, \text{plot1}, \text{plot2}, \text{plot3}\},
\begin{array}{l}
\text{dim}=\text{If}[\text{VectorQ}[n],n,\{n,1\}];
\text{n2}=\text{dim}[2];
\text{plot1}:\text{plot1}=\text{ListLinePlot}[\{\text{BrownianMeander}[n,r],\{i,1,m\}\},
\quad \text{DataRange}\to\{0,1\},\text{AxesLabel}\to\{"x","E_0^m(t)"\}];
\text{plot2}:\text{plot2}=\text{ListLinePlot}[\{\text{BrownianMeander}[n,r],\{i,1,m\}\},
\quad \text{AxesLabel}\to\{"x","y"\}];
\text{plot3}:\text{plot3}=\text{Graphics3D}[\{\text{ColorData}[1,i],\text{Line}[\text{BrownianMeander}[n,r]],
\quad \{i,1,m\}\},\text{Axes}\to\text{True},\text{AxesLabel}\to\{"x","y","z"\}];
\end{array}
\]

\text{Which}[n_2=1,\text{plot1},n_2=2,\text{plot2},n_2=3,\text{plot3}]
]

Examples

\subsection{2.3. Infinitely Divisible Distributions}

\subsubsection{2.3.1. Density of \( C_t \)}

Examples
Source code

ClearAll[DensityC]
Options[DensityC] = {Method -> "Biane", WorkingPrecision -> MachinePrecision};
DensityC[t0_, x0_, opts0 : OptionsPattern[]] := Module[
    {method, choice, wp, rule1, t1, t, x, n0, opts, fn1, fn2, λ, fn3, fn4expr, fn4},
    method = Flatten[{OptionValue[Method]}];
    choice = Cases[method, "Biane"] 
    "Devroye" | "UniE" | "UniT" | "UniG"][[1]]; 
    wp = OptionValue[WorkingPrecision];
    rule1 = Cases[method, HoldPattern["n0" -> _]]; 
    t1 = Rationalize[t0, 0];
    x1 = Rationalize[x0, 0];
    Which[
        choice === "Biane",
        t = t1; x = SetPrecision[x1, wp]; n0 = "n0" /.
        Append[rule1, "n0" -> 20],
        choice === "Devroye",
        t = t1; x = SetPrecision[x1, wp]; n0 = "n0" /.
        Append[rule1, "n0" -> 20],
        MemberQ[{"UniE", "UniT", "UniG"}, choice],
        t = t1; x = x1; n0 = "n0" /.
        Append[rule1, "n0" -> Null];
        opts = DeleteCases[FilterRules[Flatten[{opts0}],
            Options[NInverseLaplaceTransform]], "n0" -> _, {3}];
        
        fn1 := 2^t Sum[(-1)^n Gamma[n + t] (2 n + t) Gamma[n + 1]√2 π x^2 
            Exp[-((2 n + t)^2)/2 x], {(n, 0, n0)}];
        fn2 := π Sum[(-1)^n (n + 1/2) π^2 x 
            Exp[-((n + 1/2)^2 π^2 x)/2], {(n, 0, n0)}];
        fn3 := NInverseLaplaceTransform[
            1/Cosh[√2 λ]]^t, λ, x, opts];
        fn4expr := 2^t Sum[(-1)^n Gamma[n + t] Gamma[n + 1] 
            Exp[-(2 n + t) √2 λ], {(n, 0, n0)}];
        fn4 := NInverseLaplaceTransform[fn4expr, λ, x, opts];
        Which[
            choice === "Biane", fn1,
            (**)
            choice === "Devroye",
            If[t == 1, fn2, Text[Style["The Devroye formula only works for t = 1", Red]]],
            (**)
            MemberQ[{"UniE", "UniT", "UniG"}, choice],
            Which[n0 === Null, fn3, IntegerQ[n0], fn4] 
            ];
    ];

Alternative code

2.3.2. Density of $S_t$

Examples
Source code
Options[DensityS] = {Method -> "Cao", WorkingPrecision -> MachinePrecision};
DensityS[t_0__, x0_, opts0 : OptionsPattern[]] :=
  Module[{method, choice, wp, rule1, t1, x1, t, x, n0, opts, fn1, fn2, fn3, fn4, fn5expr, fn5},
    method = Flatten[{{OptionValue[Method]}}, choice = Cases[method, "Cao" | "Tolmatz" | "Devroye" | "UniE" | "UniT" | "UniG"][[1]]; wp = OptionValue[WorkingPrecision];
    rule1 = Cases[method, HoldPattern["n0" -> _]];
    t1 = Rationalize[t0, 0];
    x1 = Rationalize[x0, 0];
    Which[
      choice === "Cao",
      t = t1; x = SetPrecision[x1, wp]; n0 = "n0" /. Append[rule1, "n0" -> 20],
      choice === "Tolmatz",
      t = t1; x = SetPrecision[x1, wp]; n0 = "n0" /. Append[rule1, "n0" -> 20],
      choice === "Devroye",
      t = t1; x = SetPrecision[x1, wp]; n0 = "n0" /. Append[rule1, "n0" -> 20],
      MemberQ[{"UniE", "UniT", "UniG"}, choice],
      t = t1; x = x1; n0 = "n0" /. Append[rule1, "n0" -> Null];
      opts = DeleteCases[FilterRules[Flatten[{opts0}], {NInvertLaplaceTransform}, "n0" -> _, (3)]];
      (*--------*)
      fn1 := 2^(3 t/2)
          Sum[Gamma[n + t] Gamma[n + 1]
              Hypergeometric1F1[t + 2, 1/2, -(2 n + t)^2/2 x] -
              Hypergeometric1F1[t + 3, 3/2, -(2 n + t)^2/2 x], {n, 0, n0}];
      fn2 := x^5/4 Sum[(-1)^n
              Exp[-(n + 1/4)^2]
              1
              Gamma[1/2 - n], {n, 0, n0}];
      fn3 := Sum[(-1)^n x^2
              Exp[-n^2 x^2/2], {n, 1, n0}];
      fn4 := NInvertLaplaceTransform[
              Hypergeometric1F1[t, 2 x/2 Gamma[1/2 - n], {n, 0, n0}];
      fn5expr := 2^(3 (2 x/2))
          Sum[Gamma[n + t] Gamma[n + 1]
              Exp[-(2 n + t) sqrt(2 x) Sinh(sqrt(2 x)), {n, 0, n0}];
      fn5 := NInvertLaplaceTransform[fn5expr, \lambda, x, opts];
      Which[
        choice === "Cao", fn1,
        (**)
        choice === "Tolmatz",
        If[t = 1/2, fn2, Text[Style["The Tolmatz formula only works for t = 1/2", Red]]],
        (**)
        choice === "Devroye",
        If[t = 1, fn3, Text[Style["The Devroye formula only works for t = 1", Red]]],
        (**)
        MemberQ[{"UniE", "UniT", "UniG"}, choice],
        Which[n0 === Null, fn4, IntegerQ[n0], fn5]
      ];
2.3.3. Density of $T_1$

Examples

Source code

```
ClearAll[DensityT]
Options[DensityT] = (Method -> {"UniT", "M" -> 25}, WorkingPrecision -> 25);
DensityT[t0_, x0_, opts0 : OptionsPattern[]] :=
Module[{method, choice, rule, n0, t, x, opts1, opts, \lambda, fn1, fn2expr, fn2},
  method = Flatten[{OptionValue[Method]}];
  choice = Cases[method, "UniG" | "UniT" | "UniE"][[1]];
  rule = Cases[method, HoldPattern["n0" -> _]];  
  n0 = "n0" /. Append[rule, "n0" -> Null];
  t = Rationalize[t0, 0];
  x = Rationalize[x0, 0];
  opts1 : = opts1 = DeleteCases[Flatten[{opts0}], "n0" -> _, (3)];
  opts = If[opts0 == Null, (Method -> {"UniT", "M" -> 25}, WorkingPrecision -> 25),
    FilterRules[opts1, Options[NInvertLaplaceTransform]]];

  (*--------*)

  fn1 := NInvertLaplaceTransform[
    Tanh[Sqrt[2 \lambda]/2]^t, \lambda, x, opts];

  fn2expr := Divide[1/(2 \lambda)^(t/2) Sum[(-1)^n Gamma[n + t] Exp[-(2 n + t) \sqrt{2 \lambda}], {n, 0, n0}],
    Sum[Gamma[n + t]/Gamma[n + 1] Exp[-(2 n + t) \sqrt{2 \lambda}], {n, 0, n0}]];

  fn2 := NInvertLaplaceTransform[fn2expr, \lambda, x, opts];

  Which[n0 == Null, fn1, IntegerQ[n0], fn2]
];
```

2.3.4. Density of $\hat{C}_1$

Examples

Source code

```
DensityChat[\tau, x]
gives the density $\psi(\tau)$ of random variable $\hat{C}_1$.

- The characteristic function of $\hat{C}_1$ is $\left(\frac{1}{\cosh(\tau)}\right)^{\nu}$.

- DensityChat by default uses the formula involving the beta function.

- The Method option can be given in DensityChat with the following possible settings

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Beta&quot;</td>
<td>using the formula in terms of the beta function</td>
</tr>
<tr>
<td>&quot;Gamma&quot;</td>
<td>using the formula in terms of the gamma function</td>
</tr>
<tr>
<td>&quot;Hypergeometric2F1&quot;</td>
<td>using the formula in terms of the hypergeometric function</td>
</tr>
<tr>
<td>&quot;NumericalInversion&quot;</td>
<td>numerically inverting the characteristic function</td>
</tr>
</tbody>
</table>
```
Options[DensityChat] = (Method -> "Beta");
DensityChat[t_, x_, OptionsPattern[]] := Module[{choice, fn1, fn2, fn3, fn4, θ},
     choice = OptionValue[Method];
     fn1 := 2^(-2) x Re[Beta[2 (-I x + I x), 2]]; 
     fn2 := 2^(-2) x Abs[Gamma[2 (-I x + I x)]^2];
     fn3 := 2^(-1) x Re[Hypergeometric2F1[2 (-I x + I x), 1 2 (t - I x) + 1][(t - I x + 2), -1] +
                       Hypergeometric2F1[2 (-I x + I x), 1 2 (t + I x) + 1][(t + I x + 2), -1]]; 
     fn4 := InverseCharacteristic[
         Abs[1 - Cosh[θ]]^t, θ, x];
     Which[choice === "Beta", fn1, choice === "Gamma", fn2,
           choice === "Hypergeometric2F1", fn3, choice === "NumericalInversion", fn4]
]

\subsection*{2.3.5. Density of $\hat{S}_t$}

\textbf{Examples}

\textbf{Source code}

DensitySha[t, x]

gives the density $\phi(t,x)$ of random variable $\hat{S}_t$.

- The characteristic function of $\hat{S}_t$ is $\left(\frac{\theta}{\sinh \theta}\right)^t$.
- DensitySha by default calculates the density by numerically inverting the characteristic function.
- The Method option can be given in DensitySha with the following possible settings

\begin{tabular}{|c|l|}
\hline
"NumericalInversion" & numerically inverting the characteristic function \\
"Formula" & using the explicit formula for integer $t$ \\
\hline
\end{tabular}

- The explicit formula of the density is available for $t = 1, 2, \ldots$. The formulas of the densities for $t = 1, 2, 3$ and 4 are written into the code. Further formulas need to be written manually.
Options[DensityShat] = (Method -> "NumericalInversion");
DensityShat[t_, x_, OptionsPattern[]] :=
  Module[
    {choice, fn1, θ, fn2, φ1, φ2, φ3, φ4},
    choice = OptionValue[Method];
    fn1 := InverseCharacteristic[θ, θ, x];
    fn2 := Which[t == 1, φ1, t == 2, φ2, t == 3, φ3, t == 4, φ4];
    φ1 := \[\frac{π}{2 (\cosh(π x) + 1)}\] ;
    φ2 := \[\frac{π (\cosh(\frac{π x}{2}) - 2)}{2 (\cosh(π x) - 1)}\] ;
    φ3 := \[\frac{1}{16} π \text{Sech}\left(\frac{π x}{2}\right)^4 \left((6 + π^2 (1 + x^2)) \cosh(π x) - 2 (-3 + π^2 (1 + x^2) + 3 π x \sinh(π x))\right)\] ;
    φ4 := \[\frac{1}{96} π \text{csch}\left(\frac{π x}{2}\right)^5 \left(π x (-36 + 11 π^2 (4 + x^2)) \cosh\left(\frac{π x}{2}\right) + π x (36 + π^2 (4 + x^2)) \cosh\left(\frac{3 π x}{2}\right) - 8 (-6 + π^2 (8 + 6 x^2) + 6 π^2 (4 + 3 x^2)) \cosh(π x) \sinh\left(\frac{π x}{2}\right)\right)\] ;
    Which[choice === "NumericalInversion", fn1, choice === "Formula", fn2] ];

2.3.6. Density of \( \hat{T}_t \)

Examples

Source code

DensityThat[t, x]

gives the density \( \eta(x) \) of random variable \( \hat{T}_t \).

- The characteristic function of \( \hat{T}_t \) is \( \left(\frac{\tanh(θ)}{θ}\right)^t \).

Options[DensityThat] = (Method -> "NumericalInversion");
DensityThat[t_, x_, OptionsPattern[]] := Module[
  {choice, fn1, θ, fn2, η1, η2},
  choice = OptionValue[Method];
  fn1 := InverseCharacteristic[\(\frac{Tanh[θ]}{θ}\), θ, x, MaxRecursion -> 15];
  fn2 := Which[t == 1, η1, t == 2, η2];
  η1 := \[\frac{1}{π} \log(\coth(\frac{π}{4} \text{Abs}[x]))\] ;
  η2 := \[\frac{2}{π^2} \text{Abs}[x] \left(\text{PolyLog}[2, \text{Exp}[\frac{-1}{2} π \text{Abs}[x]]] - \text{PolyLog}[2, -\text{Exp}[\frac{-1}{2} π \text{Abs}[x]]]\right) + \frac{8}{π^3} \left(\text{PolyLog}[3, \text{Exp}[\frac{-1}{2} π \text{Abs}[x]]] - \text{PolyLog}[3, -\text{Exp}[\frac{-1}{2} π \text{Abs}[x]]]\right)\] ;
  Which[choice === "NumericalInversion", fn1, choice === "Formula", fn2] ];

2.4. Stable Distributions

2.4.1. Density of \( S_t(\alpha, \beta_t) \)

Examples
Source code

DensityStableS1[rawa_ /; 0 < rawa ≤ 2, rawb1_ /; -1 ≤ rawb1 ≤ 1, rawμ_,
    rawo1_ /; rawo1 > 0, x_] := Module[{α, β1, μ, ω1, K, solβ, β, ωσ, σ, ωσ},
    α = Rationalize[rawa, 0];
    β1 = Rationalize[rawb1, 0];
    μ = Rationalize[rawμ, 0];
    ω1 = Rationalize[rawo1, 0];
    K := K = Which[α < 1, α, α > 1, 2 - α];
    solβ := solβ = Solve[Tan[β1 π/2] = xβ Tan[π/2 α/2], {xβ}];
    β = Which[β1 = 0, 0, α = 1, β1, α = 2, 0, True, xβ /. solβ[[1]]];
    solω := solω = Solve[ω1 = xσ \(1 + \beta^2 Tan[\frac{\pi}{2} \left(\frac{\alpha}{2}\right)^2]\), {xσ}];
    σ = Which[β1 = 0 || α = 2, ω1, α = 1, \(\frac{\pi}{2}\) - ω1, True, xσ /. solω[[1]]];
    PDF[StableDistribution[α, β, μ, σ], x]
    ];
DensityStableS1[rawa_, rawb1_, x_] := DensityStableS1[rawa, rawb1, 0, 1, x];
DensityStableS1[rawa_, x_] := DensityStableS1[rawa, 0, 0, 1, x];
StableVariate generates \( \xi \) from the stable distribution with index \( \alpha \) \((0, 2)\) and skewness parameter \( \beta \) \([-1, 1]\).

\[
\text{StableVariate}[\alpha, \beta] \quad \text{generates a pseudorandom variate from a stable distribution with index } \alpha \in (0, 2) \text{ and skewness parameter } \beta \in [-1, 1].
\]

\[
\text{StableVariate}[\alpha, \beta, n] \quad \text{generates a list of } n \text{ pseudorandom variates from a stable distribution with index } \alpha \in (0, 2) \text{ and skewness parameter } \beta \in [-1, 1].
\]

\[
\text{StableVariate}[\alpha, \beta, \{n_1, n_2, \ldots\}] \quad \text{generates an } n_1 \times n_2 \times \ldots \text{ array of } n \text{ pseudorandom variates from a stable distribution with index } \alpha \in (0, 2) \text{ and skewness parameter } \beta \in [-1, 1].
\]

**Examples**
Source code

\text{StableS1}[\alpha_/; 0 < \alpha \leq 2, \beta1_/; -1 \leq \beta1 \leq 1, n_/]:=} \text{Module}\{(k, \beta0, \beta, W, S1, S2),
\kern-3.5em
\begin{align*}
\text{k} &= 1 - \text{Abs}[1 - \alpha]; \\
\beta0 &= -\pi \beta1 \text{k} / \alpha; \\
\beta &= \text{RandomVariate[UniformDistribution[\{-1/2, 1/2\}], n]}; \\
W &= \text{RandomVariate[ExponentialDistribution[1], n]}; \\
S1 &= \frac{\text{Sin}[\alpha (\beta - \beta0)]}{\text{Power}[\text{Cos}[\beta], 1/\alpha]} \times \frac{\text{Power}[\text{Cos}[\beta - \alpha (\beta - \beta0)], 1 - \alpha]}{W}; \\
S2 &= \frac{\pi}{2} \left(\frac{1}{2} + \beta1 \beta\right) \text{Tan}[\beta] - \beta1 \text{Log}\left[\frac{\pi \text{W Cos}[\beta]/2}{\pi/2 + \beta1 \beta}\right]; \\
\text{If}[\alpha = 1, \text{S2, S1}]\}
\end{align*}
}
2.4.3. Probability density \( g_{m,a} \)

Source code

```mathematica
(* \( g_{m,a} \) is a probability density *)
Options[DensityG] = (Method -> "SeriesGamma", WorkingPrecision -> MachinePrecision);
DensityG[m_Integer, a0_, x0_, opts : OptionsPattern[]] :=
Module[{a1, x1, method, choice, iniopts, inioptions, optionsM, optionsR, strings, rules, wp, a, x, M, a, b, A, bjβj, indexj, c, f1, f2},
a1 = Rationalize[a0, 0];
x1 = Rationalize[x0, 0];
method = Flatten[{OptionValue[Method]}];
choice = Cases[Append[method, "SeriesGamma"], "FoxFunction" | "SeriesGamma"][[1]]; iniopts = Flatten[{opts}];
Which[
choice == "FoxFunction", inioptions = Join[iniopts, {Method -> "SeriesGamma", WorkingPrecision -> MachinePrecision}],
choice == "SeriesGamma", inioptions = Join[iniopts, {Method -> {"M" -> 100}, WorkingPrecision -> MachinePrecision}]
];
optionsM := optionsM = FilterRules[inioptions, Method];
optionsR := optionsR = DeleteDuplicates[
FilterRules[inioptions, Except[Method]], First[H1] === First[H2] &];
strings := strings = DeleteDuplicates[Cases[optionsM, _String, {2, 3}]]; rules :=
rules = DeleteDuplicates[Cases[optionsM, _Rule, {3}], First[H1] === First[H2] &]; (*---*)
wp := wp = WorkingPrecision /. FilterRules[optionsR, WorkingPrecision];
a := a = a1;
x := x = SetPrecision[x1, wp];
M := M = "M" /. FilterRules[rules, "M"];
a = m + a - 1;
b = Power[1, \(\frac{a^n}{\Gamma[1 - a]}\), \(\frac{1}{a}\)];
A = Divide[1, a b Product[Gamma[\(\frac{k - 1}{a}\)], \(\{k, 2, m\}\)]];
bjβj := Table[\(\frac{k - 2}{a}, \frac{1}{a}\), \(\{k, m\}\)];
indexj[k_] := indexj[k] = Delete[Table[i, \(\{i, m\}\), k];
c[k_, n_] := c[k, n] = Product[Gamma[\(\frac{j - k}{a}\) - n], \(\{j, indexj[k]\}\)];
f1 := A FoxH[m, 0, (-1, 1), bjβj, \(\frac{x}{b}\), M];
f2 := A a Sum[\(\frac{c[k, n]}{\Gamma[1 - k - n a]}\) \(\frac{(-1)^n}{n!}\) Power[\(\frac{x}{b}\), k - 2 + n a], \(\{k, m\}, \{n, M\}\)];
Which[
choice == "FoxFunction", f1,
choice == "SeriesGamma", f2]
];
```

Alternative code

2.4.4. Density \( f_{m,a} \) of generalized one-sided stable distribution \( F_{m,a} \)

Source code

```mathematica
Options[DensityGeneralizedOneSidedStable] =
(Method -> "SeriesGamma", WorkingPrecision -> MachinePrecision);
DensityGeneralizedOneSidedStable[m_Integer, a0_, x0_, opts : OptionsPattern[]] :=
```
Module[{a1, xl, method, choice, iniopts, inisubopts, inioptions, inisuboptions,
   optionsM, optionsR, strings, rules, f1strings, f1rules, f1opts1, f1opts2,
   f1optsFull, f2strings, f2rules, f2opts1, f2opts2, f2optsFull, suboptions,
   subchoice, subrules, subM, a, x, a, b, \beta, A, index}, c, bj\beta, \phi, \lambda, f1, f2],
a1 = \text{Rationalize}[a0, 0];
x1 = \text{Rationalize}[x0, 0];
method = \text{Flatten}[[\text{OptionValue}[\text{Method}]]];
choice = \text{Cases}[\text{Append}[\text{method}, \text{"SeriesGamma"}],
   \text{"FoxFunction"} | \text{"SeriesGamma"} | \"UniE" | \"UniT" | \"UniG"][[1]];
iniopts := iniopts = \text{DeleteCases}[\text{Flatten}[\{\text{opts}\}],
   \text{HoldPattern}[\text{"LaplaceTransform"} \to _], \{3\}];
inisubopts := inisubopts = \text{Cases}[\text{Flatten}[\{\text{opts}\}],
   \text{HoldPattern}[\text{"LaplaceTransform"} \to _], \{3\}];
Which[
choice == \"FoxFunction",
iniopts = \text{Join}[iniopts, \{\text{Method} \to \{\"M" \to 100}, \text{WorkingPrecision} \to 30\}]];
choice == \"SeriesGamma",
iniopts = \text{Join}[iniopts, \{\text{Method} \to \{\"M" \to 100}, \text{WorkingPrecision} \to 30\}]];
choice == \"UniE",
iniopts = \text{Join}[iniopts, \{\text{Method} \to \{\"M" \to 30}, \text{WorkingPrecision} \to 30\}];
inisuboptions =
\text{Join}[inisubopts, \{\text{"LaplaceTransform"} \to \{\"Automatic"}, \"M" \to 30\}]];
choice == \"UniT",
iniopts = \text{Join}[iniopts, \{\text{Method} \to \{\"M" \to 30}, \text{WorkingPrecision} \to 30\}];
inisuboptions =
\text{Join}[inisubopts, \{\text{"LaplaceTransform"} \to \{\"Automatic"}, \"M" \to 30\}]];
choice == \"UniG",
iniopts = \text{Join}[iniopts, \{\text{Method} \to \{\"M" \to 30}, \text{WorkingPrecision} \to 60\}];
inisuboptions = \text{Join}[inisubopts, \{\text{"LaplaceTransform"} \to \{\"Automatic"}, \"M" \to 30\}]]];
optionsM := optionsM = \text{FilterRules}[inioptions, \text{Method}];
optionsR := optionsR = \text{DeleteDuplicates}[
   \text{FilterRules}[\{\text{inioptions, Except[\text{Method}]}, \text{First}[\#1] \text{\&} \text{First}[\#2] \&\}],
   \{2, 3\}];
strings := strings = \text{DeleteDuplicates}[[\text{Cases}[\text{optionsM, _String}, \{2, 3\}]]];
rules :=
\text{rules} = \text{DeleteDuplicates}[[\text{Cases}[\text{optionsM, \_Rule, \{3\}}, \text{First}[\#1] \text{\&} \text{First}[\#2] \&\]],
   \{2, 3\}];
f1strings := f1strings = \{\text{choice}\};
f1rules := f1rules = \text{FilterRules}[\text{rules}, \{\"M"\}];
f1opts1 := f1opts1 = \{\text{Method} \to \text{Join}[\text{f1strings}, \text{f1rules}]\};
f1opts2 := f1opts2 = \text{FilterRules}[\text{optionsR, WorkingPrecision}];
f1optsFull := f1optsFull = \text{Join}[f1opts1, f1opts2];
f2strings := f2strings = \{\text{choice}\};
f2rules := f2rules = \text{FilterRules}[\text{rules}, \{\"M", \"Shifting"\}];
f2opts1 := f2opts1 = \{\text{Method} \to \text{Join}[\text{f2strings}, \text{f2rules}]\};
f2opts2 := f2opts2 = \text{FilterRules}[\text{optionsR, WorkingPrecision}];
f2optsFull := f2optsFull = \text{Join}[f2opts1, f2opts2];
(*---*)
suboptions := suboptions = \text{FilterRules}[\text{inisuboptions, \"LaplaceTransform"}];
subchoice := subchoice =
\text{Cases}[\text{suboptions, \"Automatic" | \"FoxFunction" | \"SeriesGamma"}, \{2, 3\}][[1]]; subrules := subrules = \text{DeleteDuplicates}[[\text{Cases}[\text{suboptions, \_Rule, \{3\}}],
   \text{First}[\#1] \text{\&} \text{First}[\#2] \&\]]; subM := subM = \"M" /. \text{FilterRules}[\text{subrules, \{"M"\}]};
(*---*)
a := a = a1;
x := x = x1;
a := a = a + \alpha - 1;
\[ b := b = \text{Power}\left[\frac{a^n}{\Gamma[1 - \alpha]}, \frac{1}{a}\right]; \]
\[ \beta := \beta = \frac{1}{1 + \alpha}; \]
\[ A := A = \text{Divide}\left[1, a \text{Product}\left[\Gamma\left[\frac{k - 1}{a}\right], (k, 2, m)\right]\right]; \]
\[ \text{indexj}[k_] := \text{indexj}[k] = \text{Delete}\left[\text{Table}\left[i, (i, m), k\right]\right]; \]
\[ c[k_, n_] := c[k, n] = \text{Product}\left[\Gamma\left[\frac{j - k}{a} - n\right], (j, \text{indexj}[k])\right]; \]
\[ \text{bj}\beta j := \text{bj}\beta j = \text{Table}\left[\left\{\frac{k - 1}{a}, \frac{1}{a}\right\}, (k, m)\right]; \]
\[ \phi := \phi = \text{Which}\left[ \begin{array}{l}
\text{subchoice} === \text{"Automatic"}, \\
\text{Which}\left[ \\
\text{m} = 1, \text{Exp}\left[-(\lambda / b)\right], \\
\text{m} = 2, \frac{2}{\Gamma[\beta]} \text{Power}\left[\frac{\lambda}{b}, 2\right] \text{BesselK}\left[\beta, 2 \text{Power}\left[\frac{\lambda}{b}, 2\beta\right]\right], \\
\text{m} > 2, A \text{Product}\left[c[k, n], (k, m), (n, 0, \text{subM})\right]\right], \\
\text{subchoice} === \text{"FoxFunction"}, \\
2 A b \text{FoxH}[m, 0, \{\}, \text{bj}\beta j, \frac{\lambda}{b}, \text{subM}], \\
\text{subchoice} === \text{"SeriesGamma"}, \\
A \text{Product}\left[c[k, n], (k, m), (n, 0, \text{subM})\right]\right]; \\
\text{f1} := x^2 \text{DensityG}[m, a, x^{-1}, \text{f1optsFull}]; \\
\text{f2} := \text{NInvertLaplaceTransform}[\phi, \lambda, x, \text{f2optsFull}]; \]
\[ \text{Which}\left[ \\
\text{choice} === \text{"FoxFunction"} \lor \text{choice} === \text{"SeriesGamma"}, \text{f1}, \\
\text{choice} === \text{"UniE"} \lor \text{choice} === \text{"UniT"} \lor \text{choice} === \text{"UniG"}, \text{f2} \right]; \]

Alternative code

\[ 2.4.5. \text{Density } f_a \text{ of one-sided stable distribution } F_a \]

Source code

\[ \text{Options}[\text{DensityOneSidedStable}] = \]
\[ \{\text{Method} \rightarrow \text{"DensityStableSI"}, \text{WorkingPrecision} \rightarrow \text{MachinePrecision}\}; \]
\[ \text{DensityOneSidedStable}[a_0_, x_0_, \text{opts} : \text{OptionsPattern}[]} := \]
\[ \text{Module}[\{a1, x1, \text{method}, \text{choice}, \text{iniops}, \text{inioptions}, \text{optionsSM}, \text{optionsR}, \text{strings}, \text{rules}, \text{f1strings}, \text{f1rules}, \text{f1opts1}, \text{f1opts2}, \text{f1optsFull}, \text{f2strings}, \text{f2rules}, \text{f2opts1}, \text{f2opts2}, \text{f2optsFull}, \wp, a, x, m, b, k, l, bc, \delta, \lambda, f1, f2, f3, f4, f5, f6, f7, f8\}, \]
\[ a1 = \text{Rationalize}[a0, 0]; \]
\[ x1 = \text{Rationalize}[x0, 0]; \]
\[ \text{method} = \text{Flatten}[\{\text{OptionValue}[\text{Method}]\}]; \]
\[ \text{choice} = \]
\[ \text{Cases}[\text{Append}[\text{method}, \text{"DensityStableSI"}], \text{\"DensityGeneralizedOneSidedStable\" | \"UniE\" | \"UniT\" | \"UniG\" | \"FoxFunction\" | \"SeriesGamma\" | \"SeriesSine\" | \"MeijerG\" | \"HypergeometricPFQ\" | \"DensityStableSI\"][[1]]; \]
iniopts = Flatten[{opts}];
Which[
choice === "DensityGeneralizedOneSidedStable", inioptions = 
Join[iniopts, {Method -> {"SeriesGamma", "M" -> 100}, WorkingPrecision -> 30}],
choice === "UniE", inioptions = Join[iniopts,
{Method -> {"M" -> 30}, WorkingPrecision -> 30}],
choice === "UniT", inioptions = Join[iniopts,
{Method -> {"M" -> 30}, WorkingPrecision -> 30}],
choice === "UniG", inioptions = Join[iniopts,
{Method -> {"M" -> 30}, WorkingPrecision -> 60}],
choice === "FoxFunction", inioptions = Join[iniopts,
{Method -> {"M" -> 100}, WorkingPrecision -> 30}],
choice === "SeriesGamma", inioptions = Join[iniopts,
{Method -> {"M" -> 100}, WorkingPrecision -> 30}],
choice === "SeriesSine", inioptions = Join[iniopts,
{Method -> {"M" -> 100}, WorkingPrecision -> 30}],
choice === "MeijerG", inioptions = Join[iniopts,
{WorkingPrecision -> MachinePrecision}],
choice === "HypergeometricPFQ", inioptions = 
Join[iniopts, {WorkingPrecision -> MachinePrecision}],
choice === "DensityStableS1", inioptions = 
Join[iniopts, {WorkingPrecision -> MachinePrecision}]
];
optionsM := optionsM = FilterRules[inioptions, Method];
optionsR := optionsR = DeleteDuplicates[
strings := strings = DeleteDuplicates[Cases[optionsM, _String, {2, 3}]]; 
rules := 
rules = DeleteDuplicates[Cases[optionsM, _Rule, {3}, First[HH1] === First[HH2] &];
f1strings := f1strings = Cases[strings,
"FoxFunction" | "SeriesGamma" | "UniE" | "UniT" | "UniG"];
f1rules := f1rules = FilterRules[rules, {"M", "LaplaceTransform"}];
f1opts1 := f1opts1 = {Method -> Join[f1strings, f1rules]};
f1opts2 := f1opts2 = FilterRules[optionsR, WorkingPrecision];
f1optsFull := f1optsFull = Join[f1opts1, f1opts2];
f2strings := f2strings = {choice};
f2rules := f2rules = FilterRules[rules, {"M", "Shifting"}];
f2opts1 := f2opts1 = {Method -> Join[f2strings, f2rules]};
f2opts2 := f2opts2 = FilterRules[optionsR, WorkingPrecision];
f2optsFull := f2optsFull = Join[f2opts1, f2opts2];
(*-------*)
wp := wp = WorkingPrecision /. FilterRules[optionsR, WorkingPrecision];
a := a = al; 
x := x = SetPrecision[x1, wp];
M := M = "M" /. FilterRules[rules, "M"];
b := b = Power[1 - a, 1];
k := k = Denominator[a];
l := l = Numerator[a];
bc[j_] := bc[j] = 1/[n - 1/2] 
Product[Gamma[1 - i]/k, {i, j - 1}] Product[Gamma[1 - i]/k, {i, j + 1, k - 1}]/ 
Product[Gamma[1 - i]/k, {i, l - 1}];
delta[m_, n_] := delta[m, n] = Table[n + i - 1, {i, m}];
Alternative code

**2.4.6. Density \( f_{\alpha, \beta} \) of two-sided stable distribution \( F_{\alpha, \beta} \)**

Source code

ClearAll[DensityTwoSidedStable]

Options[DensityTwoSidedStable] = 

DensityTwoSidedStable[a0_, b0_, x0_, opts : OptionsPattern[]] :=
Module[{a1, b1, x1, wp, \[Alpha], \[Beta], x, M, \[Psi], \[Epsilon], \[Gamma], f1, f2, f3, f4},

a1 = Rationalize[a0, 0];
b1 = Rationalize[b0, 0];
x1 = Rationalize[x0, 0];

method = Flatten[{OptionValue[Method]}];
choice = Cases[{Append[method, "DensityStableS1"], "InverseFourierTransform" | 
"FoxFunction" | "SeriesExpansion" | "DensityStableS1"}, {1}];

iniopts = Flatten[{opts}];

Which[
choice == "InverseFourierTransform",
iniopts = Join[iniopts, {MaxRecursion \[Rule] 18}];

choice == "FoxFunction",

choice == "SeriesExpansion",

choice == "DensityStableS1",
iniopts = Join[iniopts, {WorkingPrecision \[Rule] MachinePrecision}];
]

optionsM := optionsM = FilterRules[iniopts, Method];

optionsR := optionsR = DeleteDuplicates[

strings = DeleteDuplicates[Cases[optionsM, _String, (2, 3)]]

rules :=
  rules = DeleteDuplicates[Cases[optionsM, _Rule, (3)], First[#1] === First[#2] &]
  f1opts1 := f1opts1 = Cases[optionsM, HoldPattern[Method -> ___], (3), 1]
  f1opts2 := f1opts2 = FilterRules[optionsR, Options[NIntegrate]]
  f1optsFull := f1optsFull = Join[f1opts1, f1opts2]

(*-------*)
α := α = α1;
β := β = β1;
x := x = SetPrecision[x1, wp];
M := M = "M" /. FilterRules[rules, "M"];
ψ[k_] := Which[k > 0, -Abs[k]α Exp[I π/2 β], k < 0, -Abs[k]α Exp[-I π/2 β]];
ε := ε = α^(-1);
γ[β_] := γ[β] = α - β;

f1 := -Re[NIntegrate[Exp[-I k x1] Exp[ψ[k]], {k, 0, ∞}, Evaluate[f1optsFull]]];
f2[β_, x_] := Which[
  0 < α < 1,
  x^2 FoxH[1, 1, ((0, 1), (1 - γ[β], γ[β])), ((1 - ε, ε), (1 - γ[β], γ[β])), x, M],
  1 < α ≤ 2,
  FoxH[1, 1, {(-ε, ε), (-γ[β], γ[β])}, {{-1, 1}, {-γ[β], γ[β]}}, x, M]
];
f3[β_, x_] := Which[
  0 < α < 1,
  1
  π x
  Sum[Gamma[k α + 1] (-x^-k) Sin[k π 2 (β - α)], {k, 1, M}],
  1 < α ≤ 2,
  1
  π x
  Sum[Gamma[1 + k / α] (-x)^k Sin[k π 2 α (β - α)], {k, 1, M}]
];
f4 := Which[
  0 < α < 1, DensityStable[1 - β, α],
  1 < α ≤ 2, DensityStable[1 - β, 2 - α],
  α = 2, DensityStable[2, 0, x] ];

Which[
  choice == "InverseFourierTransform", f1,
  choice == "FoxFunction", If[x ≥ 0, f2[β, x], f2[-β, -x]],
  choice == "SeriesExpansion", If[x ≥ 0, f3[β, x], f3[-β, -x]],
  choice == "DensityStable1", If[x1 == 0, SetPrecision[f4, wp], f4]
] /; Or[0 < α < 1 && Abs[β] ≤ α, 1 < α ≤ 2 && Abs[β] ≤ 2 - α]
2.4.7. The Fox function

Source code

FoxH[m_Integer, n_Integer, a[j0_List, b[j0_List, z0_, k0_Integer: 20] :=
  Module[{wp, a[j1, b[j1, a[j, b[j, p, q, z, a, b, β, A, B, C, D, s, c},
    a[j1] = Rationalize[a[j0, 0];
    b[j1] = Rationalize[b[j0, 0];
    a[j] = Which[ VectorQ[a[j1], (a[j1), MatrixQ[a[j1], a[j1];
    b[j] = Which[ VectorQ[b[j1], (b[j1), MatrixQ[b[j1], b[j1];
    p = If[a[j] == (()), 0, Length[a[j]];]
    q = If[b[j] == (()), 0, Length[b[j]];]
    z = z0;
    a[j_] := a[j] = Part[a[j1, j, 1];
    a[j_] := a[j] = Part[a[j1, j, 2];
    b[j_] := b[j] = Part[b[j1, j, 1];
    β[j_] := β[j] = Part[b[j1, j, 2];
    B[s_] := B[s] = If[n == 0, 1, Product[Gamma[1 - a[j] + a[j] s], (j, n)];]
    C[s_] := C[s] = If[m == q, 1, Product[Gamma[1 - b[j] + β[j] s], (j, m + 1, q)];]
    D[s_] := D[s] = If[n == p, 1, Product[Gamma[a[j] - a[j] s], (j, n + 1, p)];]
    s[j_, k_] := s[j, k] = b[j] + k;
    c[j_, k_] := c[j, k] = A[j, s[j, k]] B[s[j, k];
    C[s[j, k]] D[s[j, k]]
    Sum[c[j, k] (-1)^k k! β[j] Re[Power[z, b[j] + k]], (j, m), (k, 0, k0)];
    0 ≤ n ≤ p & 1 ≤ m ≤ q;]

Alternative code

- 2.5. Generation of a random variable with a given characteristic function

Source code

CFVariate[c, k, expr, x]

generates a pseudorandom variate from the distribution with a given characteristic function φ(t) with

c = \frac{1}{2\pi} \int_{-\infty}^{\infty} \phi(t) dt

k = \frac{1}{2\pi} \int_{-\infty}^{\infty} \phi''(t) dt

expr(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{ixt} \phi(t) dt
CFVariate[c_, k_, expr_, x_] := Module[{f, V1, V2, U, X, begin, other, end},
    f[u_] := expr /. x -> u;
    Label[begin];
    V1 = RandomReal[UniformDistribution[{-1, 1}]];
    V2 = RandomReal[UniformDistribution[{-1, 1}]];
    U = RandomReal[UniformDistribution[{0, 1}]];
    X = \[K浔o \sqrt{k/c \[K浔o \[K浔o V1 \[K浔o V2 \[K浔o \[K浔o If[Abs[V1] < Abs[V2], Goto[other]]; If[k U < f[X] X^2, Goto[end], Goto[begin]]; Label[other]; If[c U < f[X], Goto[end], Goto[begin]]; Label[end];
    X 
    ]

3. Code in Chapter 4

3.1. Numerical Integration

3.1.1. Simpson's rule

Details

Examples

Source code

ClearAll[SimpsonRule]
Options[SimpsonRule] =
   
   {WorkingPrecision -> MachinePrecision, "SamplePoints" -> 150, "StepSize" -> 0.05}

SimpsonRule[expr_, {x_Symbol, a_, b_}, opts : OptionsPattern[]] :=
   Module[{inioptions, options, rules, wp, a, b, n, h0, h, u, f},
      inioptions = Flatten[{opts, WorkingPrecision -> MachinePrecision,
         "SamplePoints" -> 150, "StepSize" -> 0.05}];
      options = DeleteDuplicates[inioptions, First[#1] === First[#2] &];
      rules = FilterRules[options, {"SamplePoints", "StepSize"}];
      wp = WorkingPrecision /. options;
      If[wp === Automatic,
         a = a0; b = b0,
         a = SetPrecision[a0, wp]; b = SetPrecision[b0, wp]];
      If[Order @@ rules === 1,
         n = "SamplePoints" /. rules,
         h0 = "StepSize" /. rules; n = Ceiling[\[K浔o (b0 - a0) / h0 \[K浔o ]];
         h = \[K浔o b - a \[K浔o ] / 2 n;
         u[i_] := u[i] = a + i h;
         f[u_] := f[u] = expr /. x -> u;
         h = (4 Sum[f[u[2 i - 1]], {i, n}] + 2 Sum[f[u[2 i]], {i, n - 1}] + f[a] + f[b]) / 3;

Alternative code

3.1.2. Trapezoidal rule

Details

Examples
Source code

ClearAll[TrapezoidalRule]
Options[TrapezoidalRule] =
  (WorkingPrecision -> MachinePrecision, "SamplePoints" -> 200, "StepSize" -> 0.05);
TrapezoidalRule[expr_, (x_Symbol, a0_, b0_), opts : OptionsPattern[]] :=
  Module[{inioptions, options, rules, wp, a, b, n, h0, h, u, f},
    inioptions = Flatten[{opts, WorkingPrecision -> MachinePrecision,
      "SamplePoints" -> 200, "StepSize" -> 0.05}];
    options = DeleteDuplicates[inioptions, First[#1] == First[#2] &];
    rules = FilterRules[options, {"SamplePoints", "StepSize"}];
    wp = WorkingPrecision /. options;
    If[wp == Automatic,
      a = a0; b = b0,
      a = SetPrecision[a0, wp]; b = SetPrecision[b0, wp]];
    If[Order @@@ rules == 1,
      n = "SamplePoints" /. rules,
      h0 = "StepSize" /. rules; n = Ceiling[(b0 - a0)/h0];
      b - a
      h = ———;
      n
      f[u_] := f[u] = expr /. x -> u;
      h (f[a] + f[b])/2 + Sum[f[a + kh], {k, n - 1}])];

Alternative code

- 3.2. Simulation of Statistics

- 3.2.1. Simulation of \( \hat{\rho} \)

Examples

Source code

\[ \hat{\rho}_{\text{trv}}(T_, n_: 1) := \text{Module}[\{\epsilon, \epsilonR, y, yM, yR, rv\}, \]
\[ \epsilon = \text{Which}[\text{VectorQ}[n], \text{RandomVariate}[\text{NormalDistribution[]}, \text{Prepend}[n, T]], \]
\[ n = 1, \text{RandomVariate}[\text{NormalDistribution[]}, T], \]
\[ n \geq 2, \text{RandomVariate}[\text{NormalDistribution[]}, \{T, n\}]\];
\[ y = \text{Accumulate}[\epsilon]; \]
\[ yM = \text{Most}[y]; (*y_{j-1} \text{ for } j \text{ from 2 to } T*) \]
\[ yR = \text{Rest}[y]; (*y_j \text{ for } j \text{ from 2 to } T*) \]
\[ \text{Clear}[y]; \]
\[ \text{rv} = \frac{\text{Total}[yM yR]}{\text{Total}[yM^2]}; \]
\[ \text{Clear}[yM, yR]; \]
\[ \text{rv} \];

- 3.2.2. Simulation of \( \hat{\sigma} \)

Examples
Source code

\[ \text{chatr}[T\_\_, n\_\;:\! 1] := \text{Module}[\{(\epsilon, \varepsilon R, y, yM, yR, \rho \hat{y}, \text{rv}), \]
\[ \epsilon = \text{Which}[\text{VectorQ}[n], \text{RandomVariate}[\text{NormalDistribution[]}, \text{Prepend}[n, T]], \]
\[ n = 1, \text{RandomVariate}[\text{NormalDistribution[]}, T], \]
\[ n \geq 2, \text{RandomVariate}[\text{NormalDistribution[]}, \{T, n\}]]; \]
\[ y = \text{Accumulate}[\epsilon]; \]
\[ yM = \text{Most}[y]; (*y_{j-1} for j from 2 to T*) \]
\[ yR = \text{Rest}[y]; (*y_j for j from 2 to T*) \]
\[ \text{Clear}[y]; \]
\[ \rho \hat{y} = \frac{\text{Total}[yM yR]}{\text{Total}[yM^2]}; \]
\[ \text{rv} = \frac{1}{T-1} \text{Total}[T - 1] \text{y} M^2; \]
\[ \text{Clear}[yM, yR, \rho \hat{y}]; \]
\[ \text{rv} \];
\]

\[ \text{S1rv}[T\_\_, n\_\;:\! 1] := \text{Module}[\{(\epsilon, y, \text{ysq}, \text{rv}), \]
\[ \epsilon = \text{Which}[\text{VectorQ}[n], \text{RandomVariate}[\text{NormalDistribution[]}, \text{Prepend}[n, T]], \]
\[ n = 1, \text{RandomVariate}[\text{NormalDistribution[]}, T], \]
\[ n \geq 2, \text{RandomVariate}[\text{NormalDistribution[]}, \{T, n\}]]; \]
\[ y = \text{Accumulate}[\epsilon]; (*y_j for j from 1 to T*) \]
\[ \text{Clear}[\epsilon]; \]
\[ \text{rv} = \frac{1}{T^2} \text{Total}[y^2]; \]
\[ \text{Clear}[y]; \]
\[ \text{rv} \];
\]

\[ \text{S2rv}[T\_\_, n\_\;:\! 1] := \text{Module}[\{(\epsilon, y, \text{ybar}, \text{rv}), \]
\[ \epsilon = \text{Which}[\text{VectorQ}[n], \text{RandomVariate}[\text{NormalDistribution[]}, \text{Prepend}[n, T]], \]
\[ n = 1, \text{RandomVariate}[\text{NormalDistribution[]}, T], \]
\[ n \geq 2, \text{RandomVariate}[\text{NormalDistribution[]}, \{T, n\}]]; \]
\[ y = \text{Accumulate}[\epsilon]; (*y_j for j from 1 to T*) \]
\[ \text{Clear}[\epsilon]; \]
\[ \text{ybar} = \text{Mean}[y]; \]
\[ \text{rv} = \frac{1}{T^2} \text{Total}[(y - \text{ConstantArray}[ybar, T])^2]; \]
\[ \text{Clear}[y, \text{ybar}]; \]
\[ \text{rv} \];
\]

\[ \text{3.2.5. Simulation of } S_{3T} \]

Examples

Source code

\[ \text{S3rv}[T\_\_, n\_\;:\! 1] := \text{Module}[\{(\epsilon, y, \text{ysq}, \text{rv}), \]
\[ \epsilon = \text{Which}[\text{VectorQ}[n], \text{RandomVariate}[\text{NormalDistribution[]}, \text{Prepend}[n, T]], \]
\[ n = 1, \text{RandomVariate}[\text{NormalDistribution[]}, T], \]
\[ n \geq 2, \text{RandomVariate}[\text{NormalDistribution[]}, \{T, n\}]]; \]
\[ y = \text{Accumulate}[\epsilon]; (*y_j for j from 1 to T*) \]
\[ \text{Clear}[\epsilon]; \]
\[ \text{rv} = \frac{1}{T^2} \text{Total}[(y - \text{ConstantArray}[ybar, T])^2]; \]
\[ \text{Clear}[y, \text{ybar}]; \]
\[ \text{rv} \];
\]

\[ \text{3.2.6. Simulation of } S_{4T} \]

Examples
Source code

S3rv[T_, n_: 1] := Module[{
ε, εR, yM, yMsq, U, V, rv},
  ε = Which[VectorQ[n], RandomVariate[NormalDistribution[], Prepend[n, T]],
         n = 1, RandomVariate[NormalDistribution[], T],
         n ≥ 2, RandomVariate[NormalDistribution[], {T, n}]];
  εR = Rest[ε]; (* εj for j from 2 to T*)
  yM = Most[Accumulate[ε]]; (* yj-1 for j from 2 to T*)
  Clear[ε];
  rv = Total[yM εR];
  rv = Total[yM^2];
  Clear[εR, yM];
  rv]

S3rvArithmetic[T_] := Module[(ε, y),
  ε[j_] := ε[j] = RandomVariate[NormalDistribution[]];
  y[0] = 0;
  y[j_] := y[j] = y[j - 1] + ε[j];
  T
  Sum[y[j - 1] ε[j], (j, 2, T)]
  Sum[y[j - 1]^2, (j, 2, T)]
]

3.2.6. Simulation of S4T

Examples
Source code

S4rv[T_, n_: 1] := Module[{
ε, εR, y, yM, yMsq, yR, ρhat, ohat, rv},
  ε = Which[VectorQ[n], RandomVariate[NormalDistribution[], Prepend[n, T]],
         n = 1, RandomVariate[NormalDistribution[], T],
         n ≥ 2, RandomVariate[NormalDistribution[], {T, n}]];
  y = Accumulate[ε];
  yM = Most[y]; (* yj-1 for j from 2 to T*)
  yMsq = yM^2;
  yR = Rest[y]; (* yj for j from 2 to T*)
  Clear[y];
  ρhat = Total[yM yR];
  ρhat = Total[yM yR];
  ohat = Total[yM yR];
  ohat = Total[yM yR];
  ρhat = Total[yM yR];
  ohat = Total[yM yR];
  Clear[yM, yR];
  rv = Divide[ρhat - 1, ohat];
  rv = Divide[ρhat - 1, ohat];
  Clear[ρhat, ohat, yMsq];
  rv]

3.3. Density Comparison

3.3.1. Compare Densities

Examples
Source code

Options[DensityComparison] = {"Index" -> False, "DisplayResult" -> True};
DensityComparison[points_, reffun_,
refopts_List, fun_, opts_List, OptionsPattern[] :=
Module[{pointlist, coordi, colen, newopts, options, newrefopts, reoptions,
leni, lenj, reflenj, method, settings1, wp, shortwp, settings2,
settings, rounding, RoundSD, data, density, cpu, shortdensity, shortcpu,
refdata, refrawdensity, refcpu, shortrefrawdensity, shortrefcpu,
comparison, refprec, refaccu, refdensity, shortrefdensity, prec, accu,
ricell, row1, row2, row3, rowcell, row, table, col0],
pointlist = If[ArrayDepth[points] = 2, points, Flatten[{points}]];
coordi[i_] :=
  coordi[i] = If[ArrayDepth[points] = 2, pointslist[[i]], {pointslist[[i]]}];
colen = Length[coordi[1]];
newopts = If[And @@ Map[VectorQ, opts], opts, {Flatten[{opts}]}];
options[j_] := options[j] = newopts[[j]];
newrefopts = If[And @@ Map[VectorQ, refopts], refopts, {Flatten[{refopts}]}];
reoptions[j_] := reoptions[j] = newrefopts[[j]];
leni = Length[pointlist];
lenj = Length[newopts];
reflenj = Length[newrefopts];
(*********)
settings1[j_] := settings1[j] = Cases[Method /. options[j], _Rule];
wp[j_] := wp[j] = WorkingPrecision /. options[j];
settings2[j_] := settings2[j] = Cases[options[j],
  Except[{Method -> WorkingPrecision}]];
rounding[x_] := Which[x < 0.1, N[Round[x, 10^-2], 1], x < 1, N[Round[x, 10^-2], 2],
x < 10, N[Round[x, 10^-2], 3], x < 100, N[Round[x, 10^-2], 5], True, Round[x];
RoundSD[x_, i_] := Round[x, 10^(-i + Floor[RealExponent[x]] + 1)];
(****compute densities to be checked****)
data[i_, j_] := data[i, j] = Timing[fun @ Join[coordi[i], options[j]]];
density[i_, j_] := density[i, j] = data[i, j][[2]]; (*density to be checked*)
cpu[i_, j_] := cpu[i, j] = data[i, j][[1]]; (*computing time*)
shortdensity[i_, j_] := shortdensity[i, j] = N[RoundSD[density[i, j], 5], 5];
shortcpu[i_, j_] := shortcpu[i, j] = rounding[cpu[i, j]]; (*compute densities to find the reference density****)
refdata[i_, j_] :=
  refdata[i, j] = Timing[reffun @ Join[coordi[i], refopts[j]]];
refrawdensity[i_, j_] := refrawdensity[i, j] = refdata[i, j][[2]]; (*density for finding the reference density*)
refcpu[i_, j_] := refcpu[i, j] = refdata[i, j][[1]]; (*computing time for each result for finding the reference density*)
shortrefrawdensity[i_, j_] := shortrefrawdensity[i, j] = N[RoundSD[refrawdensity[i, j], 5], 5];
shortrefcpu[i_, j_] := shortrefcpu[i, j] = rounding[refcpu[i, j]]; (*make a comparison****)
comparison[i_] := comparison[i] = MatchedSD[
  Table[refrawdensity[i, j], {j, reflenj}], Table[density[i, j], {j, lenj}]], 10];
refprec[i_, j_] := refprec[i, j] = comparison[i][[j + 2, 2]]; (*precision of each result for finding the reference density*)
refaccu[i_] := refaccu[i] = comparison[i][[1, 3, 3]]; (*accuracy of the reference density*)
refdensity[i_] := refdensity[i] = comparison[i][[1, 3, 4]]; (*reference density*)
shortrefdensity[i_] := shortrefdensity[i] = comparison[i][[1, 3, 5]]; (*reference density in the short form*)
prec[i_, j_] := prec[i, j] = comparison[i][[1, j + reflenj, 3, 2]];
DensityComparison

In[1, j_] := accu[i, j] = comparison[i][[1, j + reflenj + 3, 3]]

(*accuracy of each density*)

(*draw a table----*)

If[OptionValue["DisplayResult"] == True,
rlcell[j_] := rlcell[j] = If[pre[1, j] == "MP", (method[j], SpanFromLeft, 
SpanFromLeft, (method[j], SpanFromLeft, SpanFromLeft, SpanFromLeft));
rlcell[j_] := rlcell[j] = If[pre[1, j] == "MP", (settings[j], SpanFromLeft, 
SpanFromLeft, (settings[j], SpanFromLeft, SpanFromLeft, SpanFromLeft));
rlcell[j_] := rlcell[j] = If[pre[1, j] == "MP", ("Accu", "Result", "CPU"), 
("ED", "Accu", "CPU"));
rowcell[i_, j_] := rowcell[i, j] = If[pre[1, j] == "MP", 
(accu[i, j], shortdensity[i, j], shortcpu[i, j]),
{pre[i, j], accu[i, j], shortdensity[i, j], shortcpu[i, j]}],

rlcell[j_] := rlcell[j] = If[pre[1, j] == "MP", 
(method[j], SpanFromLeft), (method[j], SpanFromLeft, SpanFromLeft);
rlcell[j_] := rlcell[j] = If[pre[1, j] == "MP", (settings[j], SpanFromLeft), 
(SpanFromLeft, SpanFromLeft, SpanFromLeft); 
rowcell[i_, j_] := rowcell[i, j] = If[pre[1, j] == "MP", 
{accu[i, j], shortcpu[i, j]}, {pre[i, j], accu[i, j], shortcpu[i, j]}]];

row1 = Join[Table[Row["x", i], {i, colen}], 
{"Reference", SpanFromLeft}, Flatten[Table[rlcell[j], {j, lenj}], 1]]; 
row2 = Join[Table[SpanFromAbove, {i, colen}], {SpanFromAbove, SpanFromBoth}, 
Flatten[Table[rlcell[j], {i, lenj}], 1]]; 
row3 = Join[Table[SpanFromAbove, {i, colen}], {"Accu", "Density"}, 
Flatten[Table[rlcell[j], {j, lenj}]]]; 
row[i_] := row[i] = Join[Map[InputForm, coordi[i]], 
{refaccu[i], shortrefdensity[i]}, Flatten[Table[rowcell[i, j], {j, lenj}]]]; 
table = Join[Table[roww, (roww, (roww, Table[roww, {i, leni}]), 
If[OptionValue["Index"] == True, 
Grid[MapThread[Prepend, (table, co10)]]], Frame -> All, Alignment -> {Center, Center}], 
Grid[table, Frame -> All, Alignment -> {Center, Center}]]; }

DensityComparison[points_, fun_, opts_, OptionsPattern[]] :=
Module[{pointslist, coordi, colen, newopts, options, leni, lenj, method, settings1, wp, 
shortwp, settings2, settings, rounding, RoundSD, data, density, cpu, shortdensity, 
shortcpu, comparison, prec, accu, densitymatching, shortdensitymatching, 
rlcell, row1, r2cell, row2, r3cell, row3, rowcell, row, table, co10], 
pointslist = If[ArrayDepth[points] == 2, points, Flatten[points]]; 
coordi[i_] := 
coordi[i] = If[ArrayDepth[points] == 2, pointslist[[i]], {pointslist[[i]]}]; 
colen = Length[coordi[i]]; 
newopts = If[And@Map[VectorQ, opts], (Flatten[opts])]; 
newopts[i_] := options[j] = newopts[i]; 
leni = Length[pointslist]; 
lenj = Length[newopts]; 
(*---*)
method[i_] := method[i] = Cases[Flatten[Method /. options[i]]], Except[\_ Rule]; 
method[i_] := method[i] = Cases[Flatten[Method /. options[i], _Rule]; 
wp[j_] := wp[j] = WorkingPrecision /. options[j]; 
(), wp[j] == MachinePrecision, ("MP"), True, {wp[j]}]; 
settings2[j_] := settings2[j] = Cases[options[j],

Except[Method -> WorkingPrecision]];
settings[j_] := settings[j] = Join[settings1[j], shortwp[j], settings2[j]];
{-------}
rounding[x_] := Which[x < 0.1, N[Round[x, 10^-2], 1], x < 1, N[Round[x, 10^-2], 2],
   x < 10, N[Round[x, 10^-3], 3], x < 100, N[Round[x, 10^-1], 3], True, Round[x];
RoundSD[x_, i_] := Round[x, 10^-1 (1 + Floor[RealExponent[x] + 1])];
{-------}
data[i_, j_] := data[i, j] = Timing[Apply[fun, Join[coordi[i], options[j]]]];
density[i_, j_] := density[i, j] = data[i, j][[2]];
cpu[i_, j_] := cpu[i, j] = data[i, j][[1]];
shortdensity[i_, j_] := shortdensity[i, j] = N[RoundSD[density[i, j], 5], 5];
shortcpu[i_, j_] := shortcpu[i, j] = rounding[cpu[i, j]];
{-------}
comparison[i_] := comparison[i] = MatchedSD[Table[density[i, j], {j, lenj}], 10];
prec[i_, j_] := prec[i, j] = comparison[i][[1, j + 2, 2]];
accu[i_] := accu[i] = comparison[i][[1, 3, 3]]; 
densitymatching[i_] := densitymatching[i] = comparison[i][[1, 3, 4]];
shortdensitymatching[i_] := shortdensitymatching[i] = comparison[i][[1, 3, 5]]; 
{-------}
If[OptionValue["DisplayResult"] == True,
r1cell[j_] = If[prec[1, j] === "MP", 
   (method[j], SpanFromLeft), (method[j], SpanFromLeft, SpanFromLeft)];
  r2cell[j_] = If[prec[1, j] === "MP", 
   (settings[j], SpanFromLeft), 
   (settings[j], SpanFromLeft, SpanFromLeft)];
  r3cell[j_] = If[prec[1, j] === "MP", 
   ("Result", "CPU"), ("ED", "Result", "CPU")];
rowcell[i_, j_] = rowcell[i, j] = If[prec[1, j] === "MP", 
   (shortdensity[i, j], 
   shortcpu[i, j], 
   (prec[i, j], shortdensity[i, j], shortcpu[i, j]))],
{-------}
r1cell[j_] := 
   r1cell[j] = If[prec[1, j] === "MP", 
   (method[j], SpanFromLeft), (method[j], SpanFromLeft)];
  r2cell[j_] := r2cell[j] = If[prec[1, j] === "MP", 
   (settings[j], SpanFromLeft), 
   (settings[j], SpanFromLeft, SpanFromLeft)];
  r3cell[j_] := r3cell[j] = If[prec[1, j] === "MP", 
   ("CPU"), ("ED", "CPU")];
rowcell[i_, j_] := rowcell[i, j] = 
   If[prec[1, j] === "MP", 
   (shortcpu[i, j]), 
   (prec[i, j], shortcpu[i, j])];
row1 = Join[Table[Row["{x, i}"], {i, colen}], 
   Flatten[Table[r1cell[j], {j, lenj}], 1], 
   ("Digits Matching", SpanFromLeft)];
row2 = Join[Table[SpanFromAbove, {i, colen}], 
   Flatten[Table[r2cell[j], {j, lenj}], 1], 
   (SpanFromAbove, SpanFromBoth)];
row3 = Join[Table[SpanFromAbove, {i, colen}], 
   Flatten[Table[r3cell[j], {j, lenj}], 1], 
   ("Accu", 
   ("Density"));
row[i_] := row[i] = Join[Map[InputForm, coordi[i]], 
   Flatten[Table[rowcell[i, j], {j, lenj}], 1], 
   (accu[i], shortdensitymatching[i])];
table = Join[row1], (row2), (row3), Table[row[i], {i, lenj}];
co01 = Join[{Null, SpanFromAbove, SpanFromAbove}, Table[i, {i, lenj}];
If[OptionValue["Index"] == True,
Grid[MapThread[Prepend, 
   (table, co10)], Frame -> All, Alignment -> (Center, Center)], 
Grid[table, Frame -> All, Alignment -> (Center, Center)]
];
Options[DensityTable] = {"Index" -> False};
DensityTable[points___, fun___, opts___, OptionsPattern[]] :=
Module[{pointslist, funlist, options, options, leni, lenj, 
   method, settings1, wp, shortwp, settings2, settings, rounding, 
   RoundSD, data, density, cpu, longprec, prec, shortdensity, shortcpu, 
   r1cell, row1, r2cell, row2, r3cell, row3, rowcell, row, table, co10],

pointslist = If[ArrayDepth[points] == 2, points, Flatten[{{points}}]]
coordi[1_] :=
  coordi[i] = If[ArrayDepth[points] == 2, pointslist[[i]], {pointslist[[i]]}];
colen = Length[coordi[1]]
newopts = If[And @@ Map[VectorQ, opts, opts, {Flatten[{opts}]}]];
options[j_] := options[j] = newopts[j];
leni = Length[pointslist];
lenj = Length[newopts];

(----------)
settings1[j_] := settings1[j] = Cases[Method /. options[j], _Rule];
wp[j_] := wp[j] = WorkingPrecision /. options[j];
settings2[j_] := settings2[j] = Cases[options[j], _Rule];

(----------)
r0 = Timing[Apply[fun, Join[coordi[i], options[j]]]]
density[i_, j_] := density[i, j] = data[i, j][[2]]
cpu[i_, j_] := cpu[i, j] = data[i, j][[1]]
longprec[i_, j_] := longprec[i, j] = Precision[density[i, j]]
prec[i_, j_] := prec[i, j] = If[longprec[i, j] = MachinePrecision, "MP", N[Round[longprec[i, j], 10^-1], 3]]
shortdensity[i_, j_] := shortdensity[i, j] = N[RoundSD[density[i, j], 5], 5]
shortcpu[i_, j_] := shortcpu[i, j] = rounding[cpu[i, j]]

(----------)
r1cell[j_] := r1cell[j] = If[prec[i, j] === "MP",
  (method[j], spanfromleft, (method[j], spanfromleft, spanfromleft))
  Flatten[Table[r1cell[j], {{i, colen}}, {1}]]
r2cell[j_] := r2cell[j] = If[prec[i, j] === "MP", {settings[j], spanfromleft},
  Flatten[Table[r2cell[j], {{i, lenj}}, {1}]]
r2row2 = Join[Table[spanfromabove, {{i, colen}},
  Flatten[Table[r2cell[j], {{i, lenj}}, {1}]]]
r3cell[j_] := r3cell[j] = If[prec[i, j] === "MP",
  {"Density", "CPU", "ED", "Density", "CPU"}
  Flatten[Table[r3cell[j], {{i, lenj}}, {1}]]
r3row3 = Join[Table[spanfromabove, {{i, colen}},
  Flatten[Table[r3cell[j], {{i, lenj}}, {1}]]]
rrow[i_] := rrow[i] = Join[Map[InputForm, coordi[i]],
  Flatten[Table[rrowcell[i, j], {{i, lenj}}, {1}]]]
table = Join[rrow1, rrow2, rrow3, Table[rrow[i], {i, leni}]]
col0 = Join[Null, spanfromabove, spanfromabove, Table[i, {i, leni}]]

If[OptionValue["Index"] === True,
  Grid[MapThread[Prepend, {table, col0}], Frame -> All, Alignment -> {Center, Center}],
  Grid[Table, Frame -> All, Alignment -> {Center, Center}]
];

3.4. Computation of $S_3$ from its Characteristic Function

3.4.1. Evaluate the characteristic function $\phi_{S3}$

Source code
ClearAll[\$tilde, \$correct]
(* computer-generated \phi_3(\theta;x) *)
\$S3temp[\theta_, \_x_] := Which[\theta = 0, 1, \_x_ = 0, 
\frac{\text{Exp}[\text{I} \theta/2]}{\sqrt{1+\text{I} \theta}},
\text{True}, \frac{\cos\left(\sqrt{2} \text{I} \theta \_x_ \right) + \text{I} \theta \frac{\sin\left(\sqrt{2} \text{I} \theta \_x_ \right)}{\sqrt{2} \text{I} \theta \_x_}}{\sqrt{2} \text{I} \theta \_x_}^{\text{1/2}} ];

(* correct \phi_3(\theta;x) *)
Options[\$S3] = ("Interval" \to 1);
\$S3[\theta_, \_x_, OptionsPattern[]] := If[\theta = 0 \vee \_x_ = 0, \$S3temp[\theta, \_x_],
Block[{$RecursionLimit = Infinity},
Module[{int, name},
int = \text{Rationalize}[\text{ OPT}\text{Value}["Interval"], 0];
name = "$\phi3$";
\$tilde[\theta0_, \_x0_, name] := \$tilde[\theta0, \_x0, name] = \$S3temp[\theta0, \_x0];
\$correct[i_?\text{ExactNumberQ}, \_x0_, name, int] := \$correct[i, \_x0, name, int] =
If[\text{Abs}[\$correct[i - int, \_x0, name, int] + \$tilde[i, \_x0, name]] \leq
\text{Abs}[\$correct[i - int, \_x0, name, int] - \$tilde[i, \_x0, name]],
-\$tilde[i, \_x0, name], \$tilde[i, \_x0, name]];  
\text{If}[\text{IntegerQ}\left[\frac{\theta}{\text{Int}} \right], \text{\$correct}[\theta, \_x, name, int],
\text{If}[\text{Abs}[\$\text{correct}[\text{Int Quotient}[\theta, \text{int}], \_x, name, int] + \$\text{tilde}[\theta, \_x, name]] \leq
\text{Abs}[\$\text{correct}[\text{Int Quotient}[\theta, \text{int}], \_x, name, int] - \$\text{tilde}[\theta, \_x, name]],
-\$\text{tilde}[\theta, \_x, name], \$\text{tilde}[\theta, \_x, name]]]);

3.4.2. Compute the distribution function of S_3

Examples

Source code

(* integrand g_3(u;x) *)
ClearAll[gS3];
Options[gS3] = ("Interval" \to 1, WorkingPrecision \to MachinePrecision);
gS3[u_?\text{NumericQ}, \_x_?\text{NumericQ}, opts : OptionsPattern[]] :=
Module[{wp, int, u, \_x_},
wp = \text{ OPT}\text{Value}[\text{WorkingPrecision}];
int = \text{ OPT}\text{Value}["Interval"];  
\text{If}[wp == \text{Automatic},
\_x_ = \_x0; u = \_x0,
\_x_ = \text{SetPrecision}[\_x0, wp]; u = \text{SetPrecision}[\_x0, wp]];  
\text{Which}[
\_x_ = 0, 0,
u = 0, 0,
u = \frac{2}{\text{Im}[\text{Divide}[\text{Exp}[\text{I} u^2/2], \sqrt{1+\text{I} u^2}]]},
\text{True}, \frac{2}{\text{u}} \text{Im}[gS3[u^2, x, "Interval" \to int]]
]

(* distribution function F_3(x) *)
ClearAll[DistributionFunctionsS3];
Options[DistributionFunctionsS3] = (Method \to \text{Automatic}, WorkingPrecision \to \text{Automatic});
DistributionFunctionsS3[x_?\text{NumericQ}, opts : OptionsPattern[]] :=
Module[{inichoice, choice, inioptions, optionsM, optionsR, inistrings,
\text{Which}[
\_x_ = 0, 0,
u = 0, 0,
u = \frac{2}{\text{Im}[\text{Divide}[\text{Exp}[\text{I} u^2/2], \sqrt{1+\text{I} u^2}]]},
\text{True}, \frac{2}{\text{u}} \text{Im}[gS3[u^2, x, "Interval" \to int]]
]
inirules, rules, g3opts1, g3opts2, g3opts, f1patt, f1opts1, f1opts2, f2opts, f2opts1, f2opts2, f2opts, f3opts, wp, x, s, dx, g3, dummyu, f1, f2, f3),
inichoice = Cases[FilterRules[Flatten[{opts}], Method], _String, {2, 3}, 1];
choice = Cases[Append[inichoice, "NIntegrate"],
   "NIntegrate" | "SimpsonRule" | "TrapezoidalRule"][[1]];
inioptions = Which[
   choice == "NIntegrate",
   Flatten[{opts, Method -> {"s" -> 30}, WorkingPrecision -> MachinePrecision}],
   choice == "SimpsonRule",
   Flatten[{opts,
    Method -> {"s" -> 30, "StepSize" -> 0.1}, WorkingPrecision -> MachinePrecision}],
   choice == "TrapezoidalRule",
   Flatten[{opts, Method -> {"s" -> 30, "StepSize" -> 0.1},
    WorkingPrecision -> MachinePrecision}]];
(*-----*)
optionsM := optionsM = FilterRules[inioptions, Method];
optionsR := optionsR = DeleteDuplicates[
   FilterRules[inioptions, Except[Method]], First[#[1] == First[#[2]] &];
inistrings := inistrings = DeleteDuplicates[Cases[optionsM, _String, {2, 3}]];
inirules := inirules =
   DeleteDuplicates[Cases[optionsM, _Rule, {3}], First[#[1] == First[#[2]] &];
rules := rules = FilterRules[inirules, {"s"}];
(*-----*)
g3opts1 := g3opts1 = FilterRules[inirules, "Interval"];
g3opts2 :=
g3opts2 = If[x0 == 0, (WorkingPrecision -> wp), (WorkingPrecision -> Automatic)];
g3opts = Join[g3opts1, g3opts2];
f1patt := f1patt = FilterRules[Options[NIntegrate], Except[WorkingPrecision]];
f1opts1 := f1opts1 = FilterRules[inirules, f1patt];
f1opts2 := f1opts2 = FilterRules[optionsR, WorkingPrecision];
f1opts := f1opts = Join[f1opts1, f1opts2];
f2opts1 := f2opts1 = FilterRules[inirules, {"StepSize", "SamplePoints"}];
f2opts2 := f2opts2 = (WorkingPrecision -> Automatic);
f2opts := f2opts = Join[f2opts1, f2opts2];
f3opts := f3opts = f2opts;
(*-----*)
wp := wp = WorkingPrecision /. optionsR;
x = x = SetPrecision[x0, wp];
s := s = "s" /. rules;
dx := dx = Rationalize["dx" /. rules, 0];
g3[u_] := g3[u, x, g3opts];
(*-----*)
f1 := - + NIntegrate @@ Join[{g3[dummyu], {dummyu, 0, s}}, f1opts];
2 π
f2 := - + SimpsonRule @@ Join[{g3[dummyu], {dummyu, 0, s}}, f2opts];
2 π
f3 := - + TrapezoidalRule @@ Join[{g3[dummyu], {dummyu, 0, s}}, f3opts];
2 π
Which[
   choice == "NIntegrate", f1,
   choice == "SimpsonRule", f2,
   choice == "TrapezoidalRule", f3]
3.4.3. Evaluate the partial derivative of the characteristic function $\phi_{S_3}$ w.r.t. $x$

Source code:

Clear[ϕtilde, ϕcorrect]
dϕS3temp[θ_, x_] := Which[
  θ == 0, 0,  
  x == 0, \(Exp[\text{I} \theta / 2] (3 + \text{I} \theta) \theta / (6 \sqrt{1 + \text{I} \theta})\),
  True, \(\text{Divide}[\text{Exp}[\text{I} \theta / 2] \theta^2 \left(2 \sqrt{\text{I} \theta x} \cos\left[\sqrt{2 \text{I} \theta x}\right] - \sqrt{2} (2 x + 1) \sin\left[\sqrt{2 \text{I} \theta x}\right]\right), 8 (\text{I} \theta x)^{3/2} \left(\cos\left[\sqrt{2 \text{I} \theta x}\right] + \frac{\sqrt{\text{I} \theta x} \sin\left[\sqrt{2 \text{I} \theta x}\right]}{\sqrt{2} x}\right)^{3/2}\)]\)
];

Options[dϕS3] = {"Interval" -> 1};
dϕS3[θ_, x_, OptionsPattern[]] := If[θ == 0 \[Or] x == 0, dϕS3temp[θ, x],
  Block[{$RecursionLimit = Infinity},
    Module[\{int, name\},
      int = \text{Rationalize}[\text{OptionValue}\\"Interval\\", 0];
      name = "dϕS3";
      \(ϕtilde[θ0, x0_, name] := ϕtilde[θ0, x0, name] = dϕS3temp[θ0, x0]\);
      \(ϕcorrect[i_?\text{ExactNumberQ}, x0_, name, int] := ϕcorrect[i, x0, name, int] = \)
      If[Abs[ϕcorrect[i - int, x0, name, int] + ϕtilde[i, x0, name]] ≤
        Abs[ϕcorrect[i - int, x0, name, int] - ϕtilde[i, x0, name]],
        -ϕtilde[i, x0, name], ϕtilde[i, x0, name]]];
      If[Abs[ϕcorrect[\text{IntQuotient}[θ, int], x, name, int] + ϕtilde[θ, x, name]] ≤
        Abs[ϕcorrect[\text{IntQuotient}[θ, int], x, name, int] - ϕtilde[θ, x, name]],
        -ϕtilde[θ, x, name], ϕtilde[θ, x, name]]]]];

3.4.4. Compute the probability density of $S_3$

Examples
Source code

(* integrand h3(u;x) *)
ClearAll[h3];
Options[h3] = {"Interval" -> Automatic, WorkingPrecision -> MachinePrecision};
h3[u0_?NumericQ, x0_?NumericQ, opts : OptionsPattern[]] :=
Module[{inioptions, options, wp, u, x, interval, int},
inioptions = Flatten[{opts, "Interval" -> Automatic, WorkingPrecision -> MachinePrecision}] ;
options = DeleteDuplicates[inioptions, First[#1] === First[#2] & ];
wp = WorkingPrecision /. options;
If[wp === Automatic, 
 u = u0; x = x0, 
 u = SetPrecision[u0, wp]; x = SetPrecision[x0, wp];
interval = "Interval" /. options;
int := int = If[interval === Automatic,
 Which[
  0.454585 < x0 < 0.542599, 0.5,
  0.542599 < x0 < 0.570280, 0.1, 
  0.570280 < x0 < 0.571382, 0.02, 
  0.571382 < x0 < 0.571417, 0.01, 
  True, 1], interval];
Which[
 u0 = 0, 0, 
 x0 = 0, 1/3
 Im[Exp[I u^2 / 2] u^2 (3 + I u^2)] / \sqrt{1 + I u^2} (u^2 - 1), 
 x0 <= 0.571417, -2/n[0, x, "Interval" -> int]], 
 True, -2/n[0, x, "Interval" -> int]] ];
(* probability density f3(x) *)
ClearAll[ProbabilityDensity3];
ProbabilityDensity3[x0_?NumericQ, opts : OptionsPattern[]] :=
Module[{inirules, rules, h3opts1, h3opts2, h3opts, f1patt, f1opts1, f1opts2, 
 f2opts, f2opts2, f2opts, f4opts, f4opts, f4strings, f4rules, 
 f4opts1, f4opts2, f4opts, wp, x, s, dx, dummyu, f1, f2, f3, f4},
inirules = Cases[FilterRules[Flatten[{opts}], Method], _String, {2, 3}, 1];
inirules = Cases[Append[inirules, "NIntegrate"], "NIntegrate" | "SimpsonRule" | "TrapezoidalRule" | "Approximation"][[1]]; 
inioptions = Which[
 choice === "NIntegrate", 
 Flatten[{opts, Method -> {"s" -> 30}, WorkingPrecision -> MachinePrecision} ],
 choice === "SimpsonRule", 
 Flatten[{opts, Method -> {"s" -> 30, "StepSize" -> 0.1}, WorkingPrecision -> MachinePrecision} ],
 choice === "TrapezoidalRule", 
 Flatten[{opts, Method -> {"s" -> 30, "StepSize" -> 0.1}, WorkingPrecision -> MachinePrecision} ],
 choice === "Approximation", 
 Flatten[{opts, Method -> {"s" -> 30, "dx" -> 10^-6}, WorkingPrecision -> MachinePrecision} ] ];
(*---------*)
optionsM := optionsM = FilterRules[inioptions, Method];
optionsR := optionsR = DeleteDuplicates[
  FilterRules[inioptions, Except[Method]], First[#1] === First[#2] &];
instrings := inistrings = DeleteDuplicates[Cases[optionsM, _String, {2, 3}]];
inrules := inirules =
  DeleteDuplicates[Cases[optionsM, _Rule, {3}], First[#1] === First[#2] &];
rules := rules = FilterRules[inirules, {"s", "dx"}];
(*------*)
h3opts1 := h3opts1 = FilterRules[inirules, "Interval"];
h3opts2 :=
h3opts2 = If[x0 == 0, {WorkingPrecision → wp}, {WorkingPrecision → Automatic}];
h3opts = Join[h3opts1, h3opts2];
f1patt := f1patt = FilterRules[Options[NIntegrate], Except[WorkingPrecision]];
f1opts1 := f1opts1 = FilterRules[inirules, f1patt];
f1opts2 := f1opts2 = FilterRules[optionsR, WorkingPrecision];
f1opts := f1opts = Join[f1opts1, f1opts2];
f2opts1 := f2opts1 = FilterRules[inirules, {"StepSize", "SamplePoints"}];
f2opts2 := f2opts2 = (WorkingPrecision → Automatic);
f2opts := f2opts = Join[f2opts1, f2opts2];
f3opts := f3opts = f2opts;
f4patt := f4patt = Join["Interval", "s", "StepSize", "SamplePoints"], f1patt];
f4strings :=
f4strings = Cases[inistrings, "NIntegrate" | "SimpsonRule" | "TrapezoidalRule"];
f4rules := f4rules = FilterRules[inirules, f4patt];
f4opts1 := f4opts1 = (Method → Join[f4strings, f4rules]);
f4opts2 := f4opts2 = FilterRules[optionsR, WorkingPrecision];
f4opts := f4opts = Join[f4opts1, f4opts2];
(*------*)
wp := wp = WorkingPrecision /. optionsR;
x := x = SetPrecision[x0, wp];
s := s = "s" /. rules;
dx := dx = Rationalize[dx /. rules, 0];
h3[u_] := h3[u, x, h3opts];
(*------*)
f1 := NIntegrate[Join[1, h3[dummyu], dummyu, 0, s)), f1opts];
f2 := SimpsonRule[Join[1, h3[dummyu], dummyu, 0, s)), f2opts];
f3 := TrapezoidalRule[Join[1, h3[dummyu], dummyu, 0, s)), f3opts];
f4 :=
(DistributionFunctionS3[x0 + dx, f4opts] - DistributionFunctionS3[x0, f4opts]) / dx;
Which[
  choice = "NIntegrate", f1,
  choice = "SimpsonRule", f2,
  choice = "TrapezoidalRule", f3,
  choice = "Approximation", f4]
];

Alternative code

- 3.4.5. Draw a table indicating the accuracy of CDF of S3

Examples
Remove[TableDistributionFunction3];
TableDistributionFunction3[xlist_, specifications_, goal_Integer: 15] :=
Module[{length, spec, u0, n0, opts, wp, int, method, data, cellr1, row1,
cellr2, row2, cellr3, row3, cellr4, row4, row5, uppertable, lowertable},
length = Length[specifications];
spec[n_] :=
spec[n] = Join[specifications[[n]], {WorkingPrecision -> MachinePrecision,
CheckpointInterval -> 1, Method -> "NIntegrate"]];
u0[n_] := u0[n] = spec[n][[1]];
n0[n_] := n0[n] = spec[n][[2]];
opts[n_] := opts[n] = Cases[spec[n], _Rule];
wp[n_] := wp[n] = Block[{WorkingPrecision, WorkingPrecision /. opts[n]}];
int[n_] := int[n] = Block[{CheckpointInterval, CheckpointInterval /. opts[n]}];
method[n_] := method[n] = Block[{Method, Method /. opts[n]}];
data[input_] := Module[{fn, time, result, match},

fn[n_] := fn[n] = Timing[DistributionFunction3[input, u0[n], n0[n], opts[n]]];
time[n_] := time[n] = N[Round[fn[n][[1]], 10^-2]];
result[n_] := result[n] = fn[n][[2]];
match = MatchQ[Table[result[n], {n, length}], goal];
Riffle[Table[match[[1, n, 1]], {n, length}], Table[time[n], {n, length}]],
match[[1, 1, 2]], match[[1, 1, 4]]] /. MachinePrecision -> "MP"];

cellr1[n_] := cellr1[n] = {method[n], SpanFromLeft};
row1 := Flatten["x", Table[cellr1[n], {n, length}], "MDP", "Density"];
cellr2[n_] := cellr2[n] = {Row["wp=", wp[n]], SpanFromLeft};
row2 := Flatten[{SpanFromAbove, Table[cellr2[n], {n, length}]},
SpanFromAbove, SpanFromAbove] /. MachinePrecision -> "MP";
cellr3[n_] := cellr3[n] = {Row["u0=", u0[n]], SpanFromLeft};
row3 := Flatten[{SpanFromAbove, Table[cellr3[n], {n, length}]},
SpanFromAbove, SpanFromAbove];
cellr4[n_] := cellr4[n] = If[method[n] == "NIntegrate",
(SpanFromAbove, SpanFromBoth), {Row["n=", n0[n]], SpanFromLeft}];
row4 := Flatten[{SpanFromAbove, Table[cellr4[n], {n, length}]},
SpanFromAbove, SpanFromAbove];
row5 := Flatten[{SpanFromAbove,
Table["ED", "CPU"], {n, length}], SpanFromAbove, SpanFromAbove}];
uppertable = If[Apply[And, Table[method[n] == "NIntegrate", {n, length}]],
{row1, row2, row3, row5}, {row1, row2, row3, row4, row5}];
lowertable = Table[data[input], {input, If[VectorQ[xlist], xlist, (xlist)]}];
Grid[FlattenAt[Prepend[lowertable, uppertable], 1], Frame -> All,
Alignment -> {Center, Center, {({Length[uppertable] + 1, -1}, (1, 2)) \[LeftRightArrow] ".",
({Length[uppertable] + 1, -1}, (-1, -1)) \[LeftRightArrow] Left}}];

- 3.5. Laplace Transform of Dickey-Fuller Distributions
- 3.5.1. Density \(f_\theta(v)\) of the Random Variable \(V\)

Examples
Source code
Options[DensityV] =
Join[{Method -> "BPY", "M" -> 15}, WorkingPrecision -> 20], FilterRules[
Options[NInvertLaplaceTransform], Except[{Method, WorkingPrecision}]];
DensityV[v0_, opts : OptionsPattern[]] :=
Module[{vectorize, method, choice, wp, v, n0, f1, f2, β},
  vectorize[x_] := If[VectorQ[x], x, (x)];
  method = vectorize[OptionValue[Method]];
  choice = First[method];
  wp = OptionValue[WorkingPrecision];
  v = SetPrecision[v0, wp]; (*Rationalize[v0, 0];*)
  n0 := "M" /. Rest[method];
  f1 := With[{{t = 1/2, x = v},
    \[2^t \frac{\text{Gamma}[n + t]}{\text{Gamma}[n + 1]} \frac{(2 n + t)}{\sqrt{2 \pi x^3}} \text{Exp}\left[-\frac{(2 n + t)^2}{2 x}\right], (n, 0, n0)]};
  (* Use the inversion algorithm UniG *)
  f2 := NInvertLaplaceTransform[Cosh[\[2 \beta \]^{-1/2},
    β, v, FilterRules[{opts}, Options[NInvertLaplaceTransform]]];
Which[
  v0 ≤ 0, 0,
  choice == "BPY", f1,
  True, f2]
];

\[3.5.2. \text{Density} f_d(u) \text{ of the Random Variable} U\]

Examples

Source code
Options[DensityU] = Join[{Method -> "PDF"},
  FilterRules[Options[NInvertLaplaceTransform], Except[{Method}]]];
DensityU[u0_, opts : OptionsPattern[]] :=
Module[{vectorize, method, choice, wp, u, f1, f2, α},
  vectorize[x_] := If[VectorQ[x], x, (x)];
  method = vectorize[OptionValue[Method]];
  choice = First[method];
  wp = OptionValue[WorkingPrecision];
  u = SetPrecision[u0, wp]; (*Rationalize[u0, 0];*)
  f1 := \[2 \text{Exp}\left[-\frac{u - 1/2}{\sqrt{2 \pi (2 u + 1)}}\right];
  f2 := NInvertLaplaceTransform[\text{Exp}\left[\frac{\alpha}{2}\right], α,
    \sqrt{\frac{\alpha + 1}{\alpha + 1}} u, FilterRules[{opts}, Options[NInvertLaplaceTransform]]];
Which[
  u0 ≤ -1/2, 0,
  choice == "PDF", If[u == 0, N[f1, wp], f1],
  True, f2]
];

\[3.5.3. \text{Joint Density} f(u, v) \text{ of} (U, V)\]

Examples

Source code
ClearAll[JointDensityUV];
JointDensityUV[u0_?NumericQ, v0_?NumericQ, opts : OptionsPattern[]] := Module[
    {inhchoice, choice, inioptions, optionsM, optionsR, inistrings, firules, flopts1, flopts2, flopts, f2rules, f2opts1, f2opts2, f2opts, f3opts, histrings, h1rules, h1opts1, h1opts2, h1opts, u, v, a, β, f1, f2, f3, h1},
inchoice = Cases[FilterRules[Flatten[{opts}], Method], _String, {2, 3}, 1];
choice = Cases[Append[inicheck, "UniG"], "UniG" | "UniE" | "UniT" | "FT" | "Euler" | "FW"][[1]]; 
inioptions = Which[
    choice == "UniG",
    Flatten[{opts, Method -> "$M$" -> 35, WorkingPrecision -> 70}],
    choice == "UniE",
    Flatten[{opts, Method -> "$M$" -> 34, WorkingPrecision -> 70}],
    choice == "UniE",
    Flatten[{opts, Method -> "$M$" -> 40, WorkingPrecision -> 85}],
    choice == "UniT",
    Flatten[{opts, Method -> "$M$" -> 40, WorkingPrecision -> 125}],
    choice == "Transformation",
    Flatten[{opts, Method -> {"g1", "j0" -> 8}]}],
    True,
    Flatten[{opts, WorkingPrecision -> 20}]];
(-----)
optionsM := optionsM = FilterRules[inioptions, Method];
optionsR := optionsR = DeleteDuplicates[
    FilterRules[inioptions, Except[Method]], First[#1] == First[#2] &];
inistrings := inioptions = DeleteDuplicates[Cases[optionsM, _String, {2, 3}]];
inirules := inirules = DeleteDuplicates[
    Cases[optionsM, Rule, {3}], First[#1] == First[#2] &];
(-----)
f1strings := f1strings = {choice};
firules := firules = FilterRules[inirules, {"M", "Shifting"}];
flopts1 := flopts1 = {Method -> Join[f1strings, firules]};
flopts2 := flopts2 = FilterRules[optionsR, WorkingPrecision];
flopts := flopts = Join[flopts1, flopts2];
f2strings := f2strings = {choice};
f2rules := f2rules = FilterRules[inirules, "M"];
f2opts1 := f2opts1 = {Method -> Join[f2strings, f2rules]};
f2opts2 := f2opts2 = FilterRules[optionsR, WorkingPrecision];
f2opts := f2opts = Join[f2opts1, f2opts2];
f3opts := f3opts = f2opts;
histrings := h1strings = Cases[inirules, {"g1" | "g2" | "I1" | "I2" | "I3" | "UniE" | "UniT" | "UniG"}];
h1rules := h1rules = FilterRules[inirules, {"j0", "k0", "M", "Shifting"}];
h1opts1 := h1opts1 = {Method -> Join[histrings, h1rules]};
h1opts2 := h1opts2 = FilterRules[optionsR, WorkingPrecision];
h1opts := h1opts = Join[h1opts1, h1opts2];
(-----)
u = Rationalize[u0, 0];
v = Rationalize[v0, 0];
(-----)
f1 := NInvertLaplaceTransform[
    2 (β/2)^1/4 \text{Csch}\left[\sqrt{2} \beta\right] \left[\frac{1}{\sqrt{2}} \left(2 u + 1\right) \pi\right] \text{Exp}\left[-\frac{1}{\sqrt{2}} \left(2 u + 1\right) \sqrt{\beta} \text{Coth}\left[\sqrt{2} \beta\right]\right], \beta, v, flopts];
\[ f_2 := \text{NInverseLaplaceTransform}\left[ \frac{\alpha}{2} \left( \cosh\left(\sqrt{2}\beta\right) + \frac{\alpha}{\sqrt{2}\beta} \sinh\left(\sqrt{2}\beta\right) \right) \right], \]

\( (\alpha, \beta), \{u, v\}, f2\text{opts} \); 

\[ f_3 := \text{NInverseLaplaceTransform}\left[ \frac{\alpha}{2} \left( \cosh\left(\sqrt{2}\beta\right) + \frac{\alpha}{\sqrt{2}\beta} \sinh\left(\sqrt{2}\beta\right) \right) \right], \]

\( \{\beta, \alpha\}, \{v, u\}, f3\text{opts} \);

(*Comment f(u,v) from f(x,s) using the transformation rule*)

\[ h_1 := 2\sqrt{2} \text{JointDensityRS}\left[ \sqrt{2} u, 2 v, h1\text{opts} \right]; \]

Which[
    Or[u0 \leq -1/2, v0 \leq 0], 0,
    Or[choice === "UniG", choice === "GWR", choice === "UniE",
       choice === "UniT", choice === "FT", choice === "Euler", choice === "FW"], f1,
    choice === "UniTG", f2,
    choice === "UniGT", f3,
    choice === "Transformation", h1]
];

Alternative code

  - 3.6.1. Joint Density \( f(r, s) \) of \( (R, S) \)

Examples

Source code

ClearAll[JointDensityRS]

Options[JointDensityRS] := (Method -> Automatic, WorkingPrecision -> Automatic);

JointDensityRS[r_?NumericQ, s_?NumericQ, opts : OptionsPattern[]] :=

Module[{inichoice, choice, inioptions, optionsM, optionsR, inistrings,
    irules, rules, fistrings, firules, fiopts1, fiopts2, fiopts,
    f2opts, f3opts, hilstrings, hirules, hiopts1, hiopts2, hiopts, j0,
    k0, wp, x, r, s, r1, s1, b, c, w, a, K, g1fn1, g1fn2, g1, galt, g2fn1,
    g2, g2alt, v, fhat1, f1, fhat2, f2fn1, f2fn2, f2, fhat3, f3, h1],
    inioptions = Cases[FilterRules[Flatten[{opts}], Method], _String, {2, 3}, 1];
    choice = Cases[Append[inioptions, "g1"],
        "g1" | "g1alt" | "g2" | "g2alt" | "I1" | "I2" | "I3" | "Transformation"]][[1]]; 
    inioptions = Which[
        Or[choice === "g1", choice === "g1alt"],
        Flatten[{opts, Method -> {"j0" -> 8}, WorkingPrecision -> MachinePrecision}],
        Or[choice === "g2", choice === "g2alt"],
        Flatten[{opts, Method -> {"j0" -> 8}, WorkingPrecision -> MachinePrecision}],
        Or[choice === "I1", choice === "I2"],
        Flatten[{opts, Method -> {"j0" -> 8, "k0" -> 10}, WorkingPrecision -> MachinePrecision}],
        Or[choice === "I3"],
        Flatten[{opts, Method -> {"j0" -> 8, "UniT", "M" -> 25}, WorkingPrecision -> 25}],
        choice === "I3",
    ];
    inioptions = Flatten[{opts}];

    optionsM := optionsM = FilterRules[inioptions, Method];
    optionsR := optionsR = DeleteDuplicates[
        FilterRules[inioptions, Except[Method]], First[H1] === First[H2] &];
    inistrings := inistrings = DeleteDuplicates[Cases[optionsM, _String, {2, 3}]];
    inirules := inirules =
DeleteDuplicates[Cases[optionsM, _Rule, {3}], First[#1] =!= First[#2] &];
rules := rules = FilterRules[inirules, ("j0", "k0")];
(*----*)
fstrings := fstrings = Cases[inistrings, "UniE" | "UniT" | "UniG"];
frules := frules = FilterRules[inirules, ("M", "Shifting")];
fopts1 := fopts1 = (Method -> Join[fstrings, frules]);
fopts2 := fopts2 = (WorkingPrecision -> Automatic);
fopts := fopts = Join[fopts1, fopts2];
f2opts := f2opts = fopts;
fopts := fopts = fopts;
hstrings :=
  hstrings = Cases[inistrings, "UniG" | "GWR" | "UniTG" | "UniGT", {1}, 1];
hirules := hirules = frules;
hopts1 := hopts1 = (Method -> Join[hstrings, hirules]);
hopts2 := hopts2 := FilterRules[optionsR, WorkingPrecision];
hopts := hopts = Join[hopts1, hopts2];
(*----*)
j0 := j0 = "j0" /. rules;
k0 := k0 = "k0" /. rules;
wp := wp = WorkingPrecision /. optionsR;
r := r = SetPrecision[r0, wp];
s := s = SetPrecision[s0, wp];
r1 := r1 = Rationalize[r0, 0];
s1 := s1 = Rationalize[s0, 0];
(*----*)
b := b = 1 + \sqrt{2} \ r;
c := c = b / \sqrt{2} ;
\omega[j_] := \omega[j] = 2 j + 1 / 2;
a[j_, k_] := a[j, k] = 2 (j + k) + (b + 1) / 2;
K[v_, \xi_] := K[v, \xi] = \text{Exp}[\xi^2 / 4] \text{ParabolicCylinderD}[v, \xi];
(*Formula 1*)
g1fn1[j_] := g1fn1[j] = \text{Binomial}[j - 1 / 2, j] \text{Exp}\left[\frac{-(\omega[j] + b / 2)^2}{s}\right];
g1fn2[j_] := g1fn2[j] = \left(\frac{-b \sqrt{2 / s}}{\Gamma[j + 1 / 2]}\right)^{j};
g1 := \frac{2}{\pi^2 b^2 s^5} \left(\frac{2}{\pi^2 b^2 s^5}\right)^{1/4} \sum_k \text{g1fn1}[j] \text{Binomial}[j, k]
g1fn2[k] := g1fn2[k] = \left(\frac{-b \sqrt{2 / s}}{\Gamma[j + 1 / 2]}\right)^{j};
g1alt := \frac{2}{\pi^2 b^2 s^5} \left(\frac{2}{\pi^2 b^2 s^5}\right)^{1/4} \sum_k \text{Binomial}[j - 1 / 2, j] \text{Exp}\left[\frac{-(\omega[j] + b / 2)^2}{s}\right] \text{Binomial}[j, k]
g1alt := \frac{-b \sqrt{2 / s}}{\Gamma[j + 1 / 2]} \left(\frac{-b \sqrt{2 / s}}{\Gamma[j + 1 / 2]}\right)^{j};
(*Formula 2*)
g2fn1[j_] := g2fn1[j] = \left(\frac{-b \sqrt{2 / s}}{j!}\right)^{j};
g2 := \frac{2}{\pi^2 b^2 s^5} \left(\frac{2}{\pi^2 b^2 s^5}\right)^{1/4} \sum_{j+k} \text{g2fn1}[j] \text{Binomial}[j + k - 1 / 2, k]
\text{Exp}\left[-a[j, k]^2 / s\right] K[j + 3 / 2, a[j, k] \sqrt{2 / s}], \{j, 0, 0\}, \{k, 0, 0\}];
g2alt := \frac{2}{\pi^2 b^2 s^5} \left(\frac{2}{\pi^2 b^2 s^5}\right)^{1/4} \sum_{j+k} \left(\frac{-b \sqrt{2 / s}}{j!}\right)^{j} \text{Binomial}[j + k - 1 / 2, k]
\[ \exp[-a[j, k]^2 / s] K[j + 3/2, a[j, k] \sqrt{2} / s], \{j, 0, j0\}, \{k, 0, k0\}; \]

(*Inversion 1 related to g1*)
\[ \text{fhat1}[j_] := (2 \nu)^{1/4} \exp[-\sqrt{v} \left(2 \omega[j] + c \sqrt{2}\right)] \text{Hypergeometric1F1}[1 - j/2, 2 c \sqrt{2} v]; \]
\[ f1 := \frac{2}{\pi c} \text{Sum}[\text{Binomial}[j - 1/2, j] \text{NInvertLaplaceTransform}[\text{fhat1}[j], v, s, \text{f1opts}], \{j, 0, j0\}; \]

(*Inversion 2 related to g1*)
\[ \text{fhat2}[j_, \{\}] := \text{fhat2}[j, \{\} = \exp[-\sqrt{v} \left(2 \omega[j] + c \sqrt{2}\right)] \text{Power}[2 v, \{2 + 1/4\}]; \]
\[ f2fn1[j_] := f2fn1[j] = \text{Binomial}[j - 1/2, j]; \]
\[ f2fn2[\{\}] := f2fn2[\{\} = \frac{\text{Power}[-2 c, \{]} \text{Pochhammer}[1/2, \{]} + \text{Sum}[\text{f2fn1}[j] \text{Binomial}[j, \{], \{j, 0, j0\}] \text{NInvertLaplaceTransform}[\text{fhat2}[j, \{\}, v, s, \text{f2opts}], \{j, 0, j0\}, \{\{0, j\}); \]

(*Inversion 3 related to g2*)
\[ f3 := \frac{1}{4 \pi b} \text{Exp}[-\frac{1}{2} b \sqrt{v} \text{Coth} \left(\sqrt{v}\right)] \text{Power}[\sqrt{v} \text{Csch} \left(\sqrt{v}\right), 1/2]; \]
\[ f3 := \text{NInvertLaplaceTransform}[f3[v, s, f3opts]]; \]

(*Compute f(r,s) from f(u,v) using the transformation rule*)
\[ h1 := \frac{1}{2 \sqrt{2}} \text{JointDensityUV}[\frac{r1}{\sqrt{2}}, \frac{s1}{2}, \text{h1opts}]; \]

Which
\[ r0 \leq \frac{1}{2} \sqrt{2}, s0 \leq 0, 0, \]

choice = "g1", g1,
choice = "galt", g1alt,
choice = "g2", g2,
choice = "g2alt", g2alt,
choice = "I1", f1,
choice = "I2", f2,
choice = "I3", f3,
choice = "Transformation", h1]

Alternative code

\[ \text{3.6.2. Joint Distribution Function } F(r, s) \text{ of } (R, S) \]

Source code

JointDistributionRS[r0_?NumericQ, s0_?NumericQ, opts : OptionsPattern[]] := 
Module[{inichoice, choice, injunctions, optionsN, optionsN, optionsR, optionsR, inirules, rules, j0, k0, m0, wp, r, s, b, c, \omega, a, k, fn1a, fnlb, fn1c, G1, G1alt, fn2a, fn2b, fn2c, G2, G2alt, fn3a, fn3b, fn3c, G3, G3alt, fn4a, fn4b, fn4c, G4, G4alt, F, fn5a, fn5b, fn5c, fn5d, G5, G5alt], inichoice = Cases[FilterRules[{opts}, Method], _String, {2, 3}, 1]; choice = Cases[Append[inichoice, "G1"],}
G1 | G2 | G3 | G4 | G5 | G1alt | G2alt | G3alt | G4alt | G5alt

inioptions = Which[
  Or[choice == "G1", choice == "G1alt"],
  Flatten[
    (opts, Method -> {"j0" -> 8, "k0" -> 10}, WorkingPrecision -> MachinePrecision)],
  Or[choice == "G2", choice == "G2alt"],
  Flatten[
    (opts, Method -> {"j0" -> 8, "k0" -> 10}, WorkingPrecision -> MachinePrecision)],
  Or[choice == "G3", choice == "G3alt"],
  Flatten[(opts, Method -> {"j0" -> 8, "k0" -> 10, "m0" -> 10},
    WorkingPrecision -> MachinePrecision)],
  Or[choice == "G4", choice == "G4alt"],
  Flatten[(opts, Method -> {"j0" -> 8, "k0" -> 10, "m0" -> 10},
    WorkingPrecision -> MachinePrecision)],
  Or[choice == "G5", choice == "G5alt"],
  Flatten[
    (opts, Method -> {"j0" -> 8, "k0" -> 10}, WorkingPrecision -> MachinePrecision)]];

(*---*)
optionsM := optionsM = FilterRules[inioptions, Method];
optionsR := optionsR = DeleteDuplicates[
  FilterRules[inioptions, Except[Method]], First[#1] == First[#2] &];
inistings := inistings = DeleteDuplicates[Cases[optionsM, _String, (2, 3)]];
inrules := inirules = DeleteDuplicates[Cases[optionsM, _Rule, {3}], First[#1] == First[#2] &];
rules := rules = FilterRules[inirules, ("j0", "k0", "m0")];

(*---*)
j0 := j0 = "j0" /. rules;
k0 := k0 = "k0" /. rules;
m0 := m0 = "m0" /. rules;
wp := wp = WorkingPrecision /. optionsR;
r := r = SetPrecision[r0, wp];
s := s = SetPrecision[s0, wp];

(*---*)
b = 1 + \sqrt{2} \; r;
c = b / \sqrt{2};
\omega[j_] := \omega[j] = 2 j + 1 / 2;
a[j_, k_] := a[j, k] = 2 (j + k) + (b + 1) / 2;
K[v_, \xi_] := K[v, \xi] = \text{Exp}[\xi^2 / 4] \text{ParabolicCylinderD}[v, \xi];

(*Formula 1*)
fn1a[j_] := fn1a[j] = \text{Binomial}[j - \frac{1}{2}, j] \text{Exp}\left[-\frac{\omega[j]^2}{s}\right];

fn1b[\ell_] := fn1b[\ell] = \frac{(-b \sqrt{2 / s})^\ell}{\text{Gamma}[\ell + 1 / 2]};

fn1c[\ell_, k_] := fn1c[\ell, k] = \frac{(-b / \sqrt{2 s})^k}{k! \{\ell + k + 1 / 2\}};

G1 := \left(\frac{8 b^2}{\pi^2 s}\right)^{1/4} \text{Sum} \text{ fn1a[j] Binomial[j, \ell] fn1b[\ell] fn1c[\ell, k] K[\ell + k - \frac{1}{2}, j - \frac{1}{2}, \omega[j] \sqrt{2 / s}]};

\{j, 0, j0\}, \{\ell, 0, j\}, \{k, 0, k0\};

G1alt := \left(\frac{8 b^2}{\pi^2 s}\right)^{1/4} \text{Sum} \text{ Binomial[j - \frac{1}{2}, j] Exp[-\frac{\omega[j]^2}{s}] Binomial[j, \ell] \frac{(-b \sqrt{2 / s})^\ell}{\text{Gamma}[\ell + 1 / 2]}} \frac{(-b / \sqrt{2 s})^k}{k! \{\ell + k + 1 / 2\}} K[\ell + k - \frac{1}{2}, \omega[j] \sqrt{2 / s}];

\{j, 0, j0\}, \{\ell, 0, j\}, \{k, 0, k0\};
(*Formula 2*)

\[
fn2a[j_] := fn2a[j] = Binomial[j - \frac{1}{2}, j] \exp\left[-\frac{(\omega[j] + b/2)^2}{s}\right];
\]

\[
fn2b[\{\} := fn2b[\{\} = \left(-b \sqrt{2/s}\right)^{T};
\]

\[
fn2c[\{\}, k_] := fn2c[\{\}, k] = \frac{\left(b/\sqrt{2/s}\right)^k}{\Gamma(\{k + 3/2\})};
\]

\[
G2 := \left(\frac{8b^2}{\pi^2 s}\right)^{1/4} \sum \left[fn2a[j] \text{ Binomial}[j, \emptyset] \text{ fn2b[\{\}] fn2c[\{\}, k]\right] \\
K[j + k - \frac{1}{2}, (\omega[j] + b/2) \sqrt{2/s}], (j, 0, j0), \{\{0, j\}, \{0, k, 00\}\};
\]

\[
G2alt := \left(\frac{8b^2}{\pi^2 s}\right)^{1/4} \sum \left[\text{ Binomial}[j - \frac{1}{2}, j] \exp\left[-\frac{(\omega[j] + b/2)^2}{s}\right]\right] \\
\text{ Binomial}[j, \emptyset] \left(-b \sqrt{2/s}\right)^{T} \frac{(b/\sqrt{2/s})^k}{\Gamma(\{k + 3/2\})} \\
K[j + k - \frac{1}{2}, (\omega[j] + b/2) \sqrt{2/s}], (j, 0, j0), \{\{0, j\}, \{0, k, 00\}\};
\]

(*Formula 3*)

\[
fn3a[j_] := fn3a[j] = \frac{(-b \sqrt{2/s})^3}{j!};
\]

\[
fn3b[\{\}, k_] := fn3b[\{\}, k] = \text{ Binomial}[j + k - \frac{1}{2}, k] \exp\left[-\frac{(\omega[j] + 2k)^2}{s}\right];
\]

\[
fn3c[\{\}, m_] := fn3c[\{\}, m] = \frac{(-b \sqrt{2/s})^m}{(j + m + 1/2) m!};
\]

\[
G3 := \left(\frac{8b^2}{\pi^2 s}\right)^{1/4} \sum \left[fn3a[j] \text{ fn3b[\{\}, k]} fn3c[\{\}, m]\right] \text{ K[j + m - \frac{1}{2}, (\omega[j] + 2k) \sqrt{2/s}],} \\
(j, 0, j0), \{\{k, 0, k0\}, \{m, 0, m0\}\};
\]

\[
G3alt := \left(\frac{8b^2}{\pi^2 s}\right)^{1/4} \sum \left[\frac{(-b \sqrt{2/s})^3}{j!} \text{ Binomial}[j + k - \frac{1}{2}, k]\right] \\
\exp\left[-\frac{(\omega[j] + 2k)^2}{s}\right] \frac{(-b/\sqrt{2/s})^m}{(j + m + 1/2) m!} K[j + m - \frac{1}{2}, (\omega[j] + 2k) \sqrt{2/s}], \\
(j, 0, j0), \{\{k, 0, k0\}, \{m, 0, m0\}\};
\]

(*Formula 4*)

\[
fn4a[j_] := fn4a[j] = \text{ Binomial}[j - \frac{1}{2}, j] \left(-b \sqrt{2/s}\right)^3;
\]

\[
fn4b[\{\}, k_] := fn4b[\{\}, k] = \text{ Binomial}[j + k - \frac{1}{2}, k] \exp\left[-\frac{a[j, k]^2}{s}\right];
\]

\[
fn4c[\{\}, m_] := fn4c[\{\}, m] = \frac{(-b/\sqrt{2/s})^m}{\Gamma(j + m + 3/2)};
\]

\[
G4 := \left(\frac{8b^2}{\pi^2 s}\right)^{1/4} \sum \left[fn4a[j] \text{ fn4b[\{\}, k]} fn4c[\{\}, m]\right] \text{ K[j + m - \frac{1}{2}, a[j, k] \sqrt{2/s}],} \\
(j, 0, j0), \{\{k, 0, k0\}, \{m, 0, m0\}\};
\]

\[
G4alt := \left(\frac{8b^2}{\pi^2 s}\right)^{1/4} \sum \left[\text{ Binomial}[j - \frac{1}{2}, j] \left(-b \sqrt{2/s}\right)^3 \text{ Binomial}[j + k - \frac{1}{2}, k]\right]
\]
\[
\begin{aligned}
\text{Exp}\left[-\frac{a[j,k]^2}{s}\right] \frac{\left(b/\sqrt{2s}\right)^n}{\Gamma[j+m+3/2]} K\left[j+m-\frac{1}{2}, a[j,k] \sqrt{2/s}\right],
\end{aligned}
\]

\{(j, 0, j0), (k, 0, k0), (m, 0, m0)\};

(*Formula 5*)

\[
F :=
\sqrt{8} \sum \text{Binomial}[j - \frac{1}{2}, j] \text{CDF}[\text{NormalDistribution}[\omega[j], -\sqrt{2/s}], (j, 0, j0)];
\]

\[
\text{fn5a}[j_] := \text{fn5a}[j] = \text{Binomial}[j - \frac{1}{2}, j] \exp[-(\omega[j] + b/2)^2/2];
\]

\[
\text{fn5b}[\{\}] := \text{fn5b}\{\} = \{-b/\sqrt{2/s}\};
\]

\[
\text{fn5c}[k_] := \text{fn5c}[k] = \left(-\sqrt{2/s}/b\right)^k;
\]

\[
\text{fn5d}[k_, \{\}] := \text{fn5d}[k, \{\}] = \text{Gamma}[k - \{\frac{1}{2}\}];
\]

\[
G5 := F - \left(\frac{32}{\pi^3 b^2}\right)^{1/4} \sum \text{fn5a}[j] \text{Binomial}[j, \{\}] \text{fn5b}[\{\}] \text{fn5c}[k] \text{fn5d}[k, \{\}]
\]

\[
K\left[-k - \frac{3}{2}, \left(\omega[j] + \frac{b}{2}\right) \sqrt{2/s}\right], (j, 0, j0), (\{\}, 0, j), (k, 0, k0)];
\]

\[
G5\text{alt} := F - \left(\frac{32}{\pi^3 b^2}\right)^{1/4} \sum \text{Binomial}[j - \frac{1}{2}, j] \exp[-(\omega[j] + b/2)^2/2]
\]

\[
\text{Binomial}[j, \{\}] \left(-b/\sqrt{2/s}\right)^\left(\{-b/\sqrt{2/s}\}\right)^k \text{Gamma}[k - \{\frac{1}{2}\}]
\]

\[
K\left[-k - \frac{3}{2}, \left(\omega[j] + \frac{b}{2}\right) \sqrt{2/s}\right], (j, 0, j0), (\{\}, 0, j), (k, 0, k0)];
\]

Which

\[
r0 \leq \frac{\sqrt{2}}{2} \sqrt{s0} \leq 0, 0,
\]

choice == "G1", G1,
choice == "G1a", G1alt,
choice == "G2", G2,
choice == "G2a", G2alt,
choice == "G3", G3,
choice == "G3a", G3alt,
choice == "G4", G4,
choice == "G4a", G4alt,
choice == "G5", G5,
choice == "G5a", G5alt

};

Alternative code

Examples

- 3.7. Generate Densities of Unit Root Statistics
- 3.7.1. Generate $S_3$ from $f(u, \nu)$ or $f(r, s)$

Examples

Source code

ClearAll[DensityS3]
Options[DensityS3] = {Method -> Automatic, WorkingPrecision -> 20};
DensityS3[x_, ?NumericQ, u___?NumericQ, opts : OptionsPattern[]] :=
Module[{ininchoice, choice, inioptions, optionsM, optionsR, inistrings, inirules,
rules, subchoice, glstrings, glrules, glopts1, glopts2, glopts, g2strings, g2rules, g2opts1, g2opts2, g2opts, f1opts, f2opts1, f2opts2, f2opts, f3opts, f4opts, u, s0, s, smin, smax, s1UV, s2UV, s1RS, s2RS, s1, s2, g, dummyu, f1, f2, f3, f4,
inchoic = Cases[FilterRules[Flatten[{opts}], Method], _String, {2, 3}, 1];
choice = Cases[Append[inchoic, "NNIntegrate"],
           "NNIntegrate" | "SimpsonRule" | "TrapezoidalRule" | "ProbabilityDensityS3"][[1]];
inioptions = Which[
    choice == "NNIntegrate",
    Flatten[{opts, Method -> {"s" -> Automatic, WorkingPrecision -> 20, 
          "JointDensityUV", "UniG" -> 35}, WorkingPrecision -> 70}],
    choice == "SimpsonRule",
    Flatten[{opts, Method -> {"s" -> Automatic, "StepSize" -> 0.1, 
          "JointDensityUV", "UniG" -> 35}, WorkingPrecision -> 70}],
    choice == "TrapezoidalRule",
    Flatten[{opts, Method -> {"s" -> Automatic, "StepSize" -> 0.1, 
          "JointDensityUV", "UniG" -> 35}, WorkingPrecision -> 70}],
    choice == "ProbabilityDensityS3",
    Flatten[{opts}]];
(*-------*)
optionsM := optionsM = FilterRules[inioptions, Method];
optionsR := optionsR = DeleteDuplicates[
    FilterRules[inioptions, Except[{Method}]], First[H1] === First[H2] &];
inistrings := inistrings = DeleteDuplicates[Cases[optionsM, _String, {2, 3}]];
inirules := inirules = DeleteDuplicates[Cases[optionsM, Rule, {3}], First[H1] === First[H2] &];
rules := rules = FilterRules[inirules, {"s"}];
subchoice := subchoice = Cases[Append[inistrings, "JointDensityUV"],
          "JointDensityUV" | "JointDensityRS"][[1]];
(*-------*)
g1strings = Cases[inistrings, "UniG" | "GWR" | "UniTG" | "UniGT" | "Transformation"];
g2rules := glrules = FilterRules[inirules, {"M", "Shifting"}];
glopts1 := glopts1 = Method -> Join[g1strings, glrules];
glopts2 := glopts2 = FilterRules[optionsR, WorkingPrecision];
glopts := glopts = Join[glopts1, glopts2];
g2strings := Cases[inistrings, "g1" | "g2" | "I1" | "I2" | "I3" | "UniE" | "UniT" | "UniG"];
g2rules := g2rules = FilterRules[inirules, {"M", "Shifting"}];
g2opts1 := g2opts1 = Method -> Join[g2strings, g2rules];
g2opts2 := g2opts2 = FilterRules[optionsR, WorkingPrecision];
g2opts := g2opts = Join[g2opts1, g2opts2];
f1opts := f1opts = FilterRules[inirules, Options[NNIntegrate]]; f2opts1 := f2opts1 = FilterRules[inirules, {"StepSize", "SamplePoints"}]; f2opts2 := f2opts2 = (WorkingPrecision -> Automatic); f2opts := f2opts = Join[f2opts1, f2opts2]; f3opts := f3opts = f2opts; f4opts := f4opts = DeleteCases[inioptions, "ProbabilityDensityS3", {3}];
(*-------*)
u = u = Rationalize[u0]; s0 := s0 = "s" /. rules;
s := s = Which[VectorQ[s0], s0, 
    NumericQ[s0], {0, s0}, s0 === Automatic, {Automatic, Automatic}];
smin := smin = s[[1]];
smax := smax = s[[2]];
s1UV := s1UV = If[smin === Automatic, Which[x ≤ -8, 0, True, 0.008], smin];
s2UV := s2UV = If[smax === Automatic, Which[x ≤ -1, -1, x ≤ 0, 7, x ≤ 0.2, 14/2x ^ 10, x ≤ 2, 17, x ≤ 2.5, 10, x ≤ 3, 7, x ≤ 4, 4.5, True, 2.5], smax];
\[s1RS := s1RS = \text{If}[\text{If}[s1 \leq -8, 0, \text{True}], 0.016, s1];\]
\[s2RS := s2RS = \text{If}[\text{If}[smax > 0.45, 17, x \leq 0.7, 20, x \leq 2, 25, x \leq 3, 17, x \leq 4, 8, \text{True}], 4.5], smax];\]
\[g1[u_] := u \text{JointDensityUV}[x, u, g1opts];\]
\[g2[u_] := \frac{1}{\sqrt{2}} u \text{JointDensityRS}[\frac{1}{\sqrt{2}} x, u, g2opts];\]

Which[
  subchoice === "\text{JointDensityUV}"_, s1 := s1 = s1UV; s2 := s2 = s2UV; g[u_] := g1[u],
  subchoice === "\text{JointDensityRS}"_, s1 := s1 = s1RS; s2 := s2 = s2RS; g[u_] := g2[u];
]

ClearAll[\text{DensityS3Integrand}]
Options[\text{DensityS3Integrand}] = Options[\text{DensityS3}];
\text{DensityS3Integrand}[x_?\text{NumericQ}, u_?\text{NumericQ}, opts : OptionsPattern[]] :=
  \text{DensityS3}[x, u, opts];

\textbf{Alternative code}

\subsection*{3.7.2. Generate \(S_q\) from \(f(u, v)\) or \(f(r, s)\)}

\textbf{Examples}

\textbf{Source code}

ClearAll[\text{DensityS4}]
Options[\text{DensityS4}] = (\text{Method} -> \text{Automatic}, \text{WorkingPrecision} -> 20);

\text{DensityS4}[x_?\text{NumericQ}, u_?\text{NumericQ}, opts : OptionsPattern[]] :=
  Module[{inichoice, choice, inioptions, optionsM, optionsR, inistings, inirules,
    rules, subchoice, g1strings, girules, g1opts1, g1opts2, g2opts, g2strings,
    g2rules, g2opts1, g2opts2, g2opts, f1opts, f2opts1, f2opts2, f2opts, f3opts, u,
    s, smin, smax, s1UV, s2UV, s1RS, s2RS, g1, g2, s1, s2, g, dummyu, f1, f2, f3},
    inichoice = \text{Cases}[\text{FilterRules}[	ext{Flatten}[[\text{opts}]], \text{Method}], _\text{String}, \{2, 3\}, 1];
    choice = \text{Cases}[\text{Append}[\text{inichoice}, "\text{NIntegrate}"],
      "\text{NIntegrate}" | "\text{SimpsonRule}" | "\text{TrapezoidalRule}"][1];
    inioptions = \text{Which[}
      choice === "\text{NIntegrate}",
      \text{Flatten}[[\text{opts}, \text{Method} -> \{"s" -> \text{Automatic}, \text{WorkingPrecision} -> \text{MachinePrecision},
        "\text{JointDensityUV}"}, \text{Uniq"} -> 35), \text{WorkingPrecision} -> 70]],
      choice === "\text{SimpsonRule}",
      \text{Flatten}[[\text{opts}, \text{Method} -> \{"s" -> \text{Automatic}, \text{StepSize} -> 0.1, "\text{JointDensityUV}"}),
        \text{WorkingPrecision} -> \text{MachinePrecision}],
      choice === "\text{TrapezoidalRule}",
      \text{Flatten}[[\text{opts}, \text{Method} -> \{"s" -> \text{Automatic}, \text{StepSize} -> 0.1, "\text{JointDensityUV}"},
        \text{WorkingPrecision} -> \text{MachinePrecision}]]];
    ]};
inistrings := inistrings = DeleteDuplicates[Cases[optionsM, _String, {2, 3}]];
inrules := inirules = 
DeleteDuplicates[Cases[optionsM, _Rule, {3}]], First[#1] === First[#2] &];
rules := rules = FilterRules[inirules, {"s"}];
subchoice := subchoice = Cases[Append[inistrings, "JointDensityUV"], 
"JointDensityUV" | "JointDensityRS" | {1}];

(*-----*)
gstrings :=

g1strings = Cases[inistrings, "UniG" | "GWR" | "UniTG" | "UniGT" | "Transformation"];
g1rules := g1rules = FilterRules[inirules, {"M", "Shifting"}];
g1opts1 := g1opts1 = {Method -> Join[g1strings, g1rules]};
g1opts2 := g1opts2 = FilterRules[optionsR, WorkingPrecision];
g1opts := g1opts = Join[g1opts1, g1opts2];
g2strings := g2strings =
Cases[inistrings, "g1" | "g2" | "I1" | "I2" | "I3" | "UniE" | "UniT" | "UniG"];
g2rules := g2rules = FilterRules[inirules, {"M", "Shifting"}];
g2opts1 := g2opts1 = {Method -> Join[g2strings, g2rules]};
g2opts2 := g2opts2 = FilterRules[optionsR, WorkingPrecision];
g2opts := g2opts = Join[g2opts1, g2opts2];
f1opts := f1opts = FilterRules[inirules, Options[NIntegrate]];  
f2opts1 := f2opts1 = FilterRules[inirules, {"StepSize", "SamplePoints"}];
f2opts2 := f2opts2 = Join[WorkingPrecision -> Automatic];
f2opts := f2opts = Join[f2opts1, f2opts2];
f3opts := f3opts = Join[f2opts];

(*-----*)
u := u = Rationalize[u0];
s0 := s0 = "s" /. rules;
s := s = Which[VectorQ[s0], s0,
NumericQ[s0], {0, s0}, s0 === Automatic, {Automatic, Automatic}];

smin := smin = s[[1]];

smax := smax = s[[2]];
s1UV := s1UV = If[smin === Automatic,
Which[x ≤ -3.5, 0, x ≤ -2.4, 0.08, x ≤ -1.3, 0.09, True, 0.1], smin];

s2UV := s2UV = If[smax === Automatic, 
Which[x ≤ -\frac{1}{6}, -\frac{1}{2} x,

x ≤ 0.4, 3, x ≤ 2.5, 4, True, 5], smax];

s1RS := s1RS = If[smin === Automatic, 
Which[x ≤ -4, 0, x ≤ -2.3, 0.11, x ≤ -0.9, 0.13, x ≤ 0, 0.15, True, 0.14], smin];

s2RS := s2RS = If[smax === Automatic, 
Which[x ≤ -\frac{\sqrt{2}}{7}, -\frac{\sqrt{2}}{2} x, x ≤ 0,

3.5, x ≤ 0.3, 3.8, x ≤ 0.9, 4, x ≤ 1.5, 4.3, True, 4.5], smax];

g1[u_] := 2 u^2 JointDensityUV[x u, u^2, g1opts];
g2[u_] := 2 u^2 JointDensityRS[x u, u^2, g2opts];

Which[
subchoice === "JointDensityUV", s1 := s1 = s1UV; s2 := s2 = s2UV; g[u_] := g1[u],
subchoice === "JointDensityRS", s1 := s1 = s1RS; s2 := s2 = s2RS; g[u_] := g2[u]]

(*-----*)
f1 := NIntegrate @@@ Join[{g[dummyu], {dummyu, s1, s2}}], f1opts];
f2 := SimpsonRule@@Join[{g[dummyu], {dummyu, s1, s2}}, f2opts];
f3 := TrapezoidalRule@@Join[{g[dummyu], {dummyu, s1, s2}}, f3opts];

Which[
  u0 === Null, g[u],
  choice === "NIntegrate", f1,
  choice === "SimpsonRule", f2,
  choice === "TrapezoidalRule", f3]
ClearAll[DensityS4Integrand]
Options[DensityS4Integrand] = Options[DensityS3];
DensityS4Integrand[x_, u_, opts : OptionsPattern[]] := DensityS4[x, u, opts];