A Computational Study of the Bivariate Normal Probability Function

by

Jing Gai

A project submitted to the
Department of Mathematics and Statistics
in conformity with the requirements for
the degree of
Master of Science (Eng.)

Queen's University
Kingston, Ontario, Canada
December, 2001

Acknowledgments

I would like to express my deep thankfulness to my supervisors, Dr. Fady Alajaji and Dr. Glen Takahara. Their excellent guidance, valuable suggestions, and careful attention to the details were essential during the whole process of development of this project.

I would especially like to thank Dr. Leo Jonker who provided me with a great opportunity to join the Communication theory program, from which I have gained so much.

I sincerely appreciate Mrs Jennifer Read, Dr. Hwashin Hyun Shin, Guangchong Zhu, Libo Zhong and all friends in the Department of Mathematics and Statistics, for their help during my graduate studies.

I am very grateful to Dr. Fady Alajaji, Dr. Glen Takahara, and Graduate School at Queen's University for providing financial support over the past year.

Finally, I wish to thank my parents, my sisters, my husband and son, for their constant encouragement, endless love, support and understanding.

Abstract

Calculation of the bivariate normal integral has been of interest to researchers for a long time. Various algorithms have been proposed for approximate evaluation of the bivariate normal distribution function $\phi(h, k; \rho)$ and the bivariate normal probability function $L(h, k; \rho)$. In this project, we focus on the implementation and comparison of some typical and recently developed algorithms for computing the bivariate normal probability, in particular the algorithms due to Owen-Donnelly, Divgi, Drezner, Drezner-Wesolowsky, Simon-Divsalar and Vasicek.

The algorithm computations are conducted in two procedures. First we implement the above six algorithms by directly coding the original algorithms. We see that the Divgi and Vasicek algorithms do not perform consistently well over the whole argument range and are therefore eliminated from further study. Next, we concentrate on the other four algorithms, and an improved method is employed for the further computation. The four algorithms achieve identical results to at least 8 decimal digits for each point over the whole variable range.

Accuracy and efficiency are two criterion for our comparisons and performance analysis. We selected the well known Owen-Donnelly algorithm as the benchmark. After applying the improved method, the Drezner-Wesolowsky algorithm performed best in accuracy for most values of the correlation coefficient ρ. The Owen-Donnelly algorithm emerges as the fastest algorithm to calculate the desired results. The most efficient gain in performance in going from the originally coded algorithm to the improved method is realized by the Simon-Divsalar algorithm.

All the computations and comparisons in this project are done over a wide argument range. The limits of integration h and k range from -8 to 8, and ρ from -0.9 to 0.9.

Contents

Acknowledgments **i**

Abstract **ii**

List of Figures **v**

List of Tables **viii**

1 Introduction **1**

1.1 Bivariate Normal Distribution 1

1.1.1 Bivariate Normal Probability Density 1

1.1.2 The Function $\phi(h, k; \rho)$ and $L(h, k; \rho)$ 3

1.1.3 Important Relationships 4

1.2 Literature Overview 6

1.3 Contribution 8

1.4 Project Overview 9

2 Overview of Bivariate Normal Algorithms **10**

2.1 Owen-Donnelly Algorithm 10

2.2 Divgi Algorithm 11
2.3 Drezner Algorithm 12
2.4 Drezner-Wesolowsky Algorithm 13
2.5 Simon-Divsalar Algorithm 14
2.6 Vasicek Algorithm 16

3 Implementation and Comparison of the Bivariate Normal Algorithms 18
3.1 Preparations for the Implementation of the Algorithms 18
3.2 Procedure Introduction 20
3.2.1 Procedure 1 20
3.2.2 Procedure 2 21
3.3 Numerical Results and Analysis 24
3.3.1 Accuracy Analysis 25
3.3.2 Efficiency Analysis 41

4 Summary 43

A The error distribution for D, D-W and S-D algorithms in Procedure 1 45

B Tables of $L(h, k; \rho)$ 54

Bibliography 85

List of Figures

1.1 The surface of the standard bivariate normal density with $\rho = 0.5$ 2

3.1 The values of $D_i(\rho)$ in different quadrants vs. ρ in procedure 1 23

3.2 The mean of error vs. correlation in procedure 1 30

3.3 The mean of error vs correlation in procedure 2 30

3.4 The error distribution for Drezner algorithm with $\rho = -0.9$ 31

3.5 The error distribution for Drezner-Wesolowsky algorithm with $\rho = -0.9$. 31

3.6 The error distribution for Simon-Divsalar algorithm with $\rho = -0.9$ 31

3.7 The error distribution for Drezner algorithm with $\rho = -0.7$ 32

3.8 The error distribution for Drezner-Wesolowsky algorithm with $\rho = -0.7$. 32

3.9 The error distribution for Simon-Divsalar algorithm with $\rho = -0.7$ 32

3.10 The error distribution for Drezner algorithm with $\rho = -0.5$ 33

3.11 The error distribution for Drezner-Wesolowsky algorithm with $\rho = -0.5$. 33

3.12 The error distribution for Simon-Divsalar algorithm with $\rho = -0.5$ 33

3.13 The error distribution for Drezner algorithm with $\rho = -0.3$ 34

3.14 The error distribution for Drezner-Wesolowsky algorithm with $\rho = -0.3$. 34

3.15 The error distribution for Simon-Divsalar algorithm with $\rho = -0.3$ 34

3.16 The error distribution for Drezner algorithm with $\rho = -0.1$ 35

3.17 The error distribution for Drezner-Wesolowsky algorithm with $\rho = -0.1$. 35

3.18 The error distribution for Simon-Divsalar algorithm with $\rho = -0.1$ 35

3.19 The error distribution for Drezner algorithm with $\rho = 0.1$ 36

3.20 The error distribution for Drezner-Wesolowsky algorithm with $\rho = 0.1$ . . 36

3.21 The error distribution for Simon-Divsalar algorithm with $\rho = 0.1$ 36

3.22 The error distribution for Drezner algorithm with $\rho = 0.3$ 37

3.23 The error distribution for Drezner-Wesolowsky algorithm with $\rho = 0.3$ . . 37

3.24 The error distribution for Simon-Divsalar algorithm with $\rho = 0.3$ 37

3.25 The error distribution for Drezner algorithm with $\rho = 0.5$ 38

3.26 The error distribution for Drezner-Wesolowsky algorithm with $\rho = 0.5$ . . 38

3.27 The error distribution for Simon-Divsalar algorithm with $\rho = 0.5$ 38

3.28 The error distribution for Drezner algorithm with $\rho = 0.7$ 39

3.29 The error distribution for Drezner-Wesolowsky algorithm with $\rho = 0.7$ . . 39

3.30 The error distribution for Simon-Divsalar algorithm with $\rho = 0.7$ 39

3.31 The error distribution for Drezner algorithm with $\rho = 0.9$ 40

3.32 The error distribution for Drezner-Wesolowsky algorithm with $\rho = 0.9$ . . 40

3.33 The error distribution for Simon-Divsalar algorithm with $\rho = 0.9$ 40

A.1 The error distribution for D algorithm with $\rho = -0.9$ in Procedure 1 . . 46

A.2 The error distribution for D-W algorithm with $\rho = -0.9$ in Procedure 1 . 46

A.3 The error distribution for S-D algorithm with $\rho = -0.9$ in Procedure 1 . 46

A.4 The error distribution for D algorithm with $\rho = -0.7$ in Procedure 1 . . 47

A.5 The error distribution for D-W algorithm with $\rho = -0.7$ in Procedure 1 . 47

A.6 The error distribution for S-D algorithm with $\rho = -0.7$ in Procedure 1 . 47

A.7 The error distribution for D algorithm with $\rho = -0.5$ in Procedure 1 . . 48

A.8 The error distribution for D-W algorithm with $\rho = -0.5$ in Procedure 1 . 48

A.9 The error distribution for S-D algorithm with $\rho = -0.5$ in Procedure 1 . 48

A.10 The error distribution for D algorithm with $\rho = -0.3$ in Procedure 1 . . 49

A.11 The error distribution for D-W algorithm with $\rho = -0.3$ in Procedure 1 . 49

A.12 The error distribution for S-D algorithm with $\rho = -0.3$ in Procedure 1 . 49

A.13 The error distribution for D algorithm with $\rho = 0.3$ in Procedure 1 50

A.14 The error distribution for D-W algorithm with $\rho = 0.3$ in Procedure 1 . . 50

A.15 The error distribution for S-D algorithm with $\rho = 0.3$ in Procedure 1 . . 50

A.16 The error distribution for D algorithm with $\rho = 0.5$ in Procedure 1 51

A.17 The error distribution for D-W algorithm with $\rho = 0.5$ in Procedure 1 . . 51

A.18 The error distribution for S-D algorithm with $\rho = 0.5$ in Procedure 1 . . 51

A.19 The error distribution for D algorithm with $\rho = 0.7$ in Procedure 1 52

A.20 The error distribution for D-W algorithm with $\rho = 0.7$ in Procedure 1 . . 52

A.21 The error distribution for S-D algorithm with $\rho = 0.7$ in Procedure 1 . . 52

A.22 The error distribution for D algorithm with $\rho = 0.9$ in Procedure 1 53

A.23 The error distribution for D-W algorithm with $\rho = 0.9$ in Procedure 1 . . 53

A.24 The error distribution for S-D algorithm with $\rho = 0.9$ in Procedure 1 . . 53

List of Tables

3.1 The mean errors in different quadrants for the D, D-W and S-D algorithms in procedure 1 . . . 22

3.2 Accuracy results for the D, D-W and S-D algorithms in procedure 1 . . . 28

3.3 Accuracy results for the D, D-W and S-D algorithms in procedure 2 . . . 29

3.4 Average system CPU time (in seconds) $\bar{t}_A(\rho)$ . . . 42

Chapter 1

Introduction

The normal distribution occupies a central position in statistical theory, and the evaluation of univariate and multivariate normal integrals has been of interest to researchers for a long time. In this project, we concentrate our attention on the implementation and comparison of the algorithms for evaluating the bivariate normal probability function $L(h, k; \rho)$.

1.1 Bivariate Normal Distribution

1.1.1 Bivariate Normal Probability Density

Consider random variables X and Y that are jointly distributed according to the bivariate normal distribution. The mean and standard deviation of X are denoted by μ_1 and σ_1, respectively, and for Y by μ_2 and σ_2; $-1 < \rho < 1$ is the correlation coefficient between X and Y. The bivariate normal probability density is given by

$$f(x, y) = \frac{e^{-\frac{1}{2(1-\rho^2)}\left[\left(\frac{x-\mu_1}{\sigma_1}\right)^2 - 2\rho\left(\frac{x-\mu_1}{\sigma_1}\right)\left(\frac{y-\mu_2}{\sigma_2}\right) + \left(\frac{y-\mu_2}{\sigma_2}\right)^2\right]}}{2\pi\sigma_1\sigma_2\sqrt{(1-\rho^2)}}. \tag{1.1}$$

The density(1.1) can be normalized into a standard form by changing the variables as follows:

$$z_1 = \frac{x - \mu_1}{\sigma_1}, \qquad z_2 = \frac{x - \mu_2}{\sigma_2}.$$

Without loss of generality, we only consider the standard normal distribution in this project, in which $\mu_1 = \mu_2 = 0$, and $\sigma_1 = \sigma_2 = 1$.

Figure 1.1 illustrates the standard bivariate normal density with $\rho = 0.5$ in three-dimensional space. Many interesting properties of the bivariate normal density are obtained by studying the bivariate normal surface.

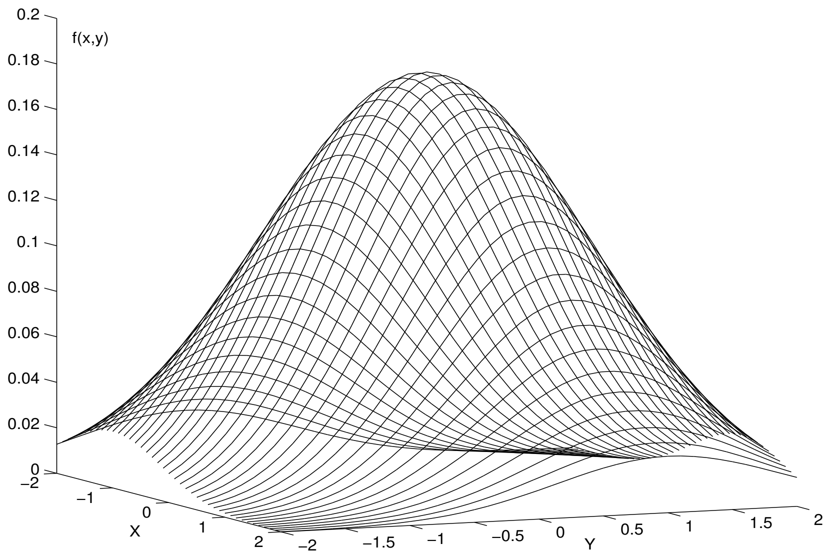

Figure 1.1: The surface of the standard bivariate normal density with $\rho = 0.5$

The standard bivariate normal surface is bell-shaped and has its maximum point at $(0, 0)$. Any plane which is perpendicular to the xy-plane intersects the surface in a curve having the shape of a normal distribution, and any plane parallel to the xy-plane intersects the surface in an ellipse called a contour of constant probability density.

When $\rho = 0$, the contour of constant probability density are circles. If $\rho = 0$, the random variables are uncorrelated and independent.

1.1.2 The Function $\phi(h, k; \rho)$ and $L(h, k; \rho)$

The standard bivariate normal distribution function (cdf) is given as:

$$\begin{aligned} \phi(h, k; \rho) &= \Pr\{(X \leq h) \cap (Y \leq k)\} \\ &= \frac{1}{2\pi\sqrt{1-\rho^2}} \int_{-\infty}^{h} \int_{-\infty}^{k} exp\left\{-\frac{x^2 - 2\rho xy + y^2}{2(1-\rho^2)}\right\} dxdy \end{aligned} \tag{1.2}$$

where h denotes the upper integration limit of the first variate, and k denotes the upper integration limit of the second variate. Furthermore, the standard bivariate normal probability function is defined by:

$$\begin{aligned} L(h, k; \rho) &= \Pr\{(X > h) \cap (Y > k)\} \\ &= \frac{1}{2\pi\sqrt{1-\rho^2}} \int_{h}^{\infty} \int_{k}^{\infty} exp\left\{-\frac{x^2 - 2\rho xy + y^2}{2(1-\rho^2)}\right\} dxdy. \end{aligned} \tag{1.3}$$

In the past 100 years, there have been many algorithms for calculating the bivariate normal integral based on both $\phi(h, k; \rho)$ and $L(h, k; \rho)$. In this project, we focus on $L(h, k; \rho)$, because its evaluation is more commonly considered in the literature [11], and evaluation of $L(h, k; \rho)$ is essentially equivalent to evaluation of $\phi(h, k; \rho)$ from the relationship (1.4) in the next subsection.

1.1.3 Important Relationships

The relationships below are very useful when dealing with the bivariate normal probability function; they will be used in later chapters.

(1) The relationship between $L(h, k; \rho)$ and $\phi(h, k; \rho)$ is:

$$\phi(h, k; \rho) = L(-h, -k; \rho). \tag{1.4}$$

(2) In different quadrants of the hk-plane, $L(h, k; \rho)$ satisfies the following relationships:

$$L(-h, -k; \rho) = L(h, k; \rho) + Q(-h) + Q(-k) - 1, \tag{1.5}$$

$$L(h, -k; \rho) = Q(h) - L(h, k; -\rho), \tag{1.6}$$

$$L(-h, k; \rho) = Q(k) - L(h, k; -\rho), \tag{1.7}$$

where the Q-function $Q(\cdot)$ is the univariate standard normal probability:

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty exp(-\frac{z^2}{2})dz.$$

(3) Since x and y can be exchanged in (1.2) and (1.3), $\phi(h, k; \rho)$ and $L(h, k; \rho)$ are symmetric about the plane $h = k$.

(4) Abramowitz and Stegun (1970) [1] provided the following relationship between the bivariate normal distribution in the general case and in the special case when one of the variables is zero:

$$\phi(x,y;\rho) = \phi\left(x,0;\frac{(\rho x - y)\ sign(x)}{\sqrt{x^2-2\rho xy+y^2}}\right) + \phi\left(y,0;\frac{(\rho y - x)\ sign(y)}{\sqrt{x^2-2\rho xy+y^2}}\right) - C \quad (1.8)$$

for $xy \neq 0$, where

$$sign(x) = \begin{cases} +1, & \text{if } x \geq 0 \\ -1, & \text{if } x < 0, \end{cases}$$

and $C = C(x,y)$ is defined as follows:

$$C = \begin{cases} 0, & \text{if } xy > 0 \\ 1/2, & \text{if } xy < 0. \end{cases}$$

1.2 Literature Overview

The bivariate normal integrals arise in many applications. There are now many algorithms available for computation of the function $L(h, k; \rho)$ or $\phi(h, k; \rho)$.

W. F. Sheppard was perhaps the first statistician who concerned himself with the evaluation of the bivariate normal integral. In his paper [18] published in 1900 he reduced (1.3) to a single integral with finite limits.

In 1901, Pearson published a method for evaluating the integral as a power series in ρ involving tetrachoric functions. Based on this method Pearson produced a table for determining the "volume of any quadrant or of any cell of a bivariate normal distribution" [16] in 1931. These were subsequently extended and corrected to form the National Bureau of Standards' table of the Bivariate Normal Distribution Function [25] (1959).

Charlier (1931) [5] suggested to use of a series expansion in terms of the correlation coefficient and Plackett (1954) [17] showed how known values of the integral may be used to calculate further values, by integrating over the range of the correlation coefficient. Both of these methods are, however, slow and laborious.

Gupta (1963) [12] presented a survey of the work on multivariate probability integrals and related functions starting with the bivariate case. Two tables of the probability integrals of the equicorrelated multivariate normal are given in this paper.

A method more suited to computer computations was developed by Owen (1956) [14]. He first introduced the T-function as a key term in expressing the bivariate normal CDF (see Section 2.1). Donnelly (1973) [9] directly coded Owen's method and formed the Owen-Donnelly algorithm, which has been the most widely implemented of all bivariate normal algorithms [22], and gives the values of the bivariate normal CDF to 15 digit accuracy [9] [11]. Much of the early research on computation of the bivariate normal integrals is based on it. The work on which Sowden and Ashford (1969) [20], Borth

(1973) [4], Daley (1974) [7], Young and Minder (1974) [24] concentrated was to find more efficient methods for the calculation of Owen's T-function.

Divgi (1979) [8] estimated the complementary probability $L(h, k; \rho)$ by transforming the two correlated random variables to orthogonal random variables and finding the equivalent probability in terms of polar coordinates.

Drezner (1978) [10] provided different formulas that apply to various combinations of h, k, ρ based on direct computation of the double integral by the Gauss quadrature method. Drezner and Wesolowsky (1990) [11] presented a simple method that achieves single precision accuracy with less computation time than Divgi's algorithm.

Terza and Welland (1991) [22] compared eight algorithms (Owen, Young, Daley, Drezner, Divgi, Bouver and Bargmann (1979) [3], Parrish and Bargmann (1981) [15], Terza and Welland) for evaluating the bivariate normal CDF. Terza and Welland selected variables h and k range from -4.1 to 4.1, ρ from -0.9 to 0.9; accuracy and efficiency were the criterion on which the comparisons were made. All algorithms were coded in the FORTRAN language provided by algorithm developers or authors. The algorithm by Divgi was judged the best for each criterion.

In 1998, Simon and Divsalar [19], by shifting the coordinate system, derived a simple and efficient expression for the bivariate normal probability function $L(h, k; \rho)$. This alternate form consisted of two single integrals with finite limits.

Vasicek (1998) [23] gave an infinite series expansion for the bivariate normal distribution function. It is an alternating series that converges as a geometric series with quotient $(1 - \rho^2)$. Hence the method represents a good performance when ρ is large in absolute value.

1.3 Contribution

The major contributions of this work are:

(1) We implement the following typical algorithms for the bivariate normal probability function $L(h, k; \rho)$ by using the original algorithms:

- Owen-Donnelly.
- Drezner.
- Divgi.
- Drezner-Wesolowsky.
- Simon-Divsalar.
- Vasicek.

(2) We implement the Owen-Donnelly, Drezner, Drezner-Wesolowsky and Simon-Divsalar algorithms by using an improved method.

(3) We present performance analysis and comparison of the above algorithms in terms of accuracy and efficiency, for a wide argument range ($h, k \sim [-8, 8], \rho \sim [-0.9, 0.9]$). This is the widest range of arguments that has been considered to date.

(4) We provide the software written in C implementing all of the above calculations from which the bivariate normal probability $L(h, k; \rho)$ can be computed.

1.4 Project Overview

The project is organized into four chapters including Chapter 1.

In Chapter 2, we briefly describe the six approximation algorithms for computing the bivariate normal probability function. They are: Owen-Donnelly, Drezner, Divgi, Drezner-Wesolowsky, Simon-Divsalar and Vasicek.

In Chapter 3, we implement all the algorithms described in the previous chapter following two procedures. At first, we use the original methods to get the desired results; then we eliminate the Divgi and Vasicek algorithms which do not perform consistently well over for the whole range of the arguments h, k, and ρ. Next, we implement the other four algorithms in the positive quadrant in the hk plane, and use these computations to evaluate $L(h, k; \rho)$ in the whole plane by using the relationships introduced in Section 1.1.3. We also present the comparisons and the performance analysis in terms of accuracy and efficiency for the above algorithms.

Chapter 4 summarizes our work, and it is followed by an appendix. The appendix provides tables of $L(h, k; \rho)$ calculated by the Owen-Donnelly, Drezner, Drezner-Wesolowsky and Simon-Divsalar algorithms for a wide range of arguments ($h, k \sim [-8, 8]$ with increment 1, $\rho \sim [-0.9, 0.9]$ with increment 0.2).

A floppy is attached on the back cover, which contains the codes written in C for each algorithm implemented under both procedures.

Chapter 2

Overview of Bivariate Normal Algorithms

In this chapter we present the formulas and brief descriptions of typical algorithms developed in the past 40 years for calculation of the bivariate normal integral $L(h, k; \rho)$ or $\phi(h, k; \rho)$ function that will be implemented in Chapter 3.

2.1 Owen-Donnelly Algorithm

Owen [14] expressed the bivariate normal CDF in terms of the univariate distribution function and the T-function:

$$\phi(h, k; \rho) = \frac{1}{2}\phi(h) + \frac{1}{2}\phi(k) - T(h, a_h) - T(k, a_k) - C, \tag{2.1}$$

where $C = 0$ if $hk > 0$ or if $hk = 0$ but $h + k \geq 0$, otherwise $C = 1/2$, $\phi(\cdot)$ denotes the standard univariate normal CDF,

$$a_h = \frac{k}{h\sqrt{1-\rho^2}} - \frac{\rho}{\sqrt{1-\rho^2}}, \qquad a_k = \frac{h}{k\sqrt{1-\rho^2}} - \frac{\rho}{\sqrt{1-\rho^2}},$$

and the T-function is defined as:

$$T(h,a) = \frac{1}{2\pi}\int_0^a \frac{exp\left\{-\frac{1}{2}h^2(1+x^2)\right\}}{1+x^2}dx. \tag{2.2}$$

A series expression for $T(h,a)$ may be obtained by expanding the numerator of the integrand of (2.2) in the usual exponential series, dividing by the denominator, and integrating term by term. Rearrangement of the terms of this series gives:

$$T(h,a) = \frac{arctan\ \ a}{2\pi} - \frac{1}{2\pi}\sum_{j=0}^{\infty} c_j a^{2j+1}, \tag{2.3}$$

where

$$c_j = \frac{(-1)^j}{2j+1}\left\{1 - \exp(-\frac{h^2}{2})\sum_{i=0}^{j}\frac{h^{2i}}{2^i i!}\right\}.$$

Donnelly [9] implemented the algorithm by using the FORTRAN language to code Owen's method and achieved a high degree of accuracy.

2.2 Divgi Algorithm

Divgi [8] estimated the L-function by transforming the two correlated random variables to orthogonal random variables and finding the equivalent probability in terms of polar coordinates. He showed the following:

$$L(h,k;\rho) = W(R,\pi/2-\theta) + W(R,\theta-\varphi) + C, \tag{2.4}$$

where

$$R^2 = \frac{h^2 - 2\rho hk + k^2}{1-\rho^2},$$

θ, φ are special angles which are given by

$$cos(\pi/2 - \theta) = (k - \rho h)/R(1 - \rho^2)^{\frac{1}{2}},$$

$$cos(\theta - \varphi) = (h - \rho k)/R(1 - \rho^2)^{\frac{1}{2}},$$

where $C = 1$ if $h < 0$ and $k < 0$ and $C = 0$ otherwise.

As for the W-function which was first introduced by Ruben (1961), Divgi suggested to approximate it as follows:

$$W(R, \psi) \cong (2\pi)^{-1} e^{-R^2/2} \left\{ \psi - \sum_{k=0}^{10} d_k R^{k+1} \int_0^{\psi} cos(z)^{k+1} dz \right\}, \qquad |\psi| \leq \pi/2, \tag{2.5}$$

where the d_k's are coefficients given in [8]. We can reduce the error by using more terms in the sum, but in this project we only use the terms in (2.5) as suggested by Divgi. Equation (2.5) is only suitable for $|\psi| \leq \pi/2$, but we can use the following relationship to calculate the W-function for all $\psi \in$ [-π, π]. For $\psi > 0$,

$$W(R, -\psi) = -W(R, \psi), \tag{2.6}$$

$$W(R, \psi) + W(R, \pi - \psi) = Q(R\ sin\psi). \tag{2.7}$$

2.3 Drezner Algorithm

Drezner [10] transformed the bivariate normal integral (1.2) by using the following change of variables:

$$u_1 = \frac{h - x}{\{2(1 - \rho^2)\}^{1/2}}, \qquad u_2 = \frac{k - y}{\{2(1 - \rho^2)\}^{1/2}}.$$

Then $\phi(h, k; \rho)$ can be expressed as:

$$\phi(h, k; \rho) = \frac{(1 - \rho^2)^{1/2}}{\pi} \int_0^{\infty} \int_0^{\infty} exp\{-u_1^2\} exp\{-u_2^2\} f(u_1, u_2) du_1 du_2 \tag{2.8}$$

where

$$f(u_1, u_2) = exp\{h_1(2u_1 - h_1) + k_1(2u_2 - k_1) + 2\rho(u_1 - h_1)(u_2 - k_1)\},$$

and

$$h_1 = \frac{h}{\{2(1-\rho^2)\}^{1/2}}, \qquad k_1 = \frac{k}{\{2(1-\rho^2)\}^{1/2}}.$$

The expression for $\phi(h, k; \rho)$ in (2.8) is evaluated by using the Gaussian quadrature method developed by Steen (1969) [21]:

$$\phi(h, k; \rho) = \frac{(1-\rho^2)^{1/2}}{\pi} \sum_{i,j=1}^{K} A_i A_j f(x_i, x_j), \tag{2.9}$$

where K is the degree of the approximation, and the values of A_i, x_i for K=15 can be found in [21]. If $h, k, \rho < 0$, then $0 < f(u_1, u_2) \leq 1$, and the error in (2.8) is relatively small. By using the relationships introduced in Section 1.1.3, it is easy to get the desired result for all values of h, k and ρ considered in this project. The advantage of this method is that for every ρ (even close to 1) there is no convergence problem.

2.4 Drezner-Wesolowsky Algorithm

The Drezner-Wesolowsky algorithm [11] is based on a formula for the partial derivative of the bivariate probability function $L(h, k; \rho)$ with respect to the correlation coefficient.

Sheppard [18] reduced (1.3) to a one dimensional integral:

$$L(h, k; \rho) = \frac{1}{2\pi} \int_{cos^{-1}\rho}^{\pi} exp\left\{-\frac{h^2 - 2hkcosz + k^2}{2sin^2 z}\right\} dz. \tag{2.10}$$

By taking the derivative with respect to ρ, we get the interesting relationship:

$$\partial L/\partial\rho(h,k;\rho) = \frac{1}{2\pi\sqrt{1-\rho^2}} exp\left\{-\frac{h^2-2hk\rho+k^2}{2(1-\rho^2)}\right\}. \tag{2.11}$$

The bivariate normal probability function $L(h,k;\rho)$ for uncorrelated random variables is

$$L(h,k;0) = Q(h)Q(k),$$

hence the desired $L(h,k;\rho)$ is obtained as

$$L(h,k;\rho) = \int_0^\rho \partial L/\partial\rho(h,k;\rho)d\rho + Q(h)Q(k). \tag{2.12}$$

Substituting (2.11) into (2.12), we obtain:

$$L(h,k;\rho) = \frac{1}{2\pi}\int_0^\rho \frac{1}{\sqrt{1-r^2}} exp\left\{-\frac{h^2-2hkr+k^2}{2(1-r^2)}\right\} dr + Q(h)Q(k). \tag{2.13}$$

When $h=k=0$, the L-function (2.13) results in the well-known Johnson and Kotz formula (1972) [13]:

$$L(0,0;\rho) = 1/4 + (sin^{-1}\rho)/2\pi.$$

2.5 Simon-Divsalar Algorithm

Simon and Divsalar [19] rewrote (1.3) by changing the variables $x = x_1 + h$, $y = y_1 + k$, and obtained:

$$L(h,k;\rho) = \int_0^\infty\int_0^\infty \frac{exp\left\{-\frac{(x_1+h)^2-2\rho(x_1+h)(y_1+k)+(y_1+k)^2}{2(1-\rho^2)}\right\}}{2\pi\sqrt{1-\rho^2}} dx_1dy_1. \tag{2.14}$$

Simon and Divsalar interpreted the above integrals as the probability that a signal vector $\vec{S} = (-h, -k)$ received in correlated unit variance Gaussian noise falls in the upper right quadrant of the $x_1 y_1$-plane.

Define $\bar{S} = \sqrt{h^2 + k^2}$, $\varphi_s = tan^{-1}\frac{k}{h}$. The evaluation of the average probability of error in a two-dimensional additive white Gaussian noise (AWGN) channel is considerably simplified by choosing the origin of coordinates for each decision region as that defined by the signal vector (Craig, 1991) [6]. After switching to polar coordinates (see details in [19]), the error probability that the signal $\vec{S} = (-h, -k)$ falls in the right upper quadrant can be expressed as the sum of the two integrals:

$$\begin{aligned} L(h,k;\rho) &= \frac{1}{2\pi}\int_0^{\pi/2-\varphi_s} \frac{\sqrt{1-\rho^2}}{1-\rho sin2\theta}\cdot exp\left\{-\frac{\bar{S}^2}{2}\frac{1-\rho sin2\theta}{1-\rho^2}\frac{cos^2\varphi_s}{sin^2\theta}\right\}d\theta \\ &+ \frac{1}{2\pi}\int_0^{\varphi_s} \frac{\sqrt{1-\rho^2}}{1-\rho sin2\theta}\cdot exp\left\{-\frac{\bar{S}^2}{2}\frac{1-\rho sin2\theta}{1-\rho^2}\frac{sin^2\varphi_s}{sin^2\theta}\right\}d\theta, \end{aligned} \tag{2.15}$$

where Eq. (2.15) is valid only for $h \geq 0$.

Plugging $\bar{S}$ and φ_s in (2.15) yields

$$\begin{aligned} L(h,k;\rho) &= \frac{1}{2\pi}\int_0^{\pi/2-tan^{-1}(k/h)} \frac{\sqrt{1-\rho^2}}{1-\rho sin2\theta}\cdot exp\left\{-\frac{h^2}{2}\frac{1-\rho sin2\theta}{(1-\rho^2)sin^2\theta}\right\}d\theta \\ &+ \frac{1}{2\pi}\int_0^{tan^{-1}(k/h)} \frac{\sqrt{1-\rho^2}}{1-\rho sin2\theta}\cdot exp\left\{-\frac{k^2}{2}\frac{1-\rho sin2\theta}{(1-\rho^2)sin^2\theta}\right\}d\theta. \end{aligned} \tag{2.16}$$

For our calculation results in Chapter 3, this method can only be used for $h \geq 0$. In order to compare all the algorithms in the same variable ranges, we will calculate $L(h, k; \rho)$ over the entire range of h and k by using the relationships introduced in Chapter 1.

2.6 Vasicek Algorithm

The Vasicek algorithm [23] concentrated on calculating $\phi(x, 0; \rho)$, then using the Abramowitz and Stegun relationship (1.8) to obtain the values of $\phi(h, k; \rho)$.

For $\rho^2 \leq 0.5$, Vasicek recommends using the tetrachoric series

$$\phi(x, 0; \rho) = \frac{1}{2}\phi(x) + \frac{1}{\sqrt{2\pi}} f(x) \sum_{j=0}^{\infty} \frac{1}{j!(2j+1)} (-1)^j 2^{-j} He_{2j}(x) \rho^{2j+1}, \tag{2.17}$$

where $He_{2j}(\cdot)$ is the Hermite polynomial, given by

$$He_{2j}(x) = \sum_{i=0}^{j} \frac{(2j)!}{i!(2j-2i)!} (-1)^i 2^{-i} x^{2(j-i)},$$

and $f(x)$ is the standard normal pdf.

The tetrachoric series (2.17) converges approximately as $\sum \rho^{2j}/j$, and is not very practical to use when ρ is large in absolute value. Vasicek suggested, when $\rho^2 > 0.5$, the use of the following formulas (2.18) and (2.19) to calculate $\phi(x, 0; \rho)$. Since the series (2.20) converges approximately as $\sum (1-\rho^2)^j/j$, it represents a good performance when ρ is large in absolute value.

For $\rho^2 > 0.5$ and $\rho > 0$, use:

$$\phi(x, 0; \rho) = min\left\{\phi(x), \frac{1}{2}\right\} - S(x); \tag{2.18}$$

and for $\rho^2 > 0.5$ and $\rho < 0$, use:

$$\phi(x, 0; \rho) = max\left\{\phi(x) - \frac{1}{2}, 0\right\} - S(x); \tag{2.19}$$

where

$$S(x) = \sum_{j=0}^{\infty} A_j(x), \tag{2.20}$$

$$A_j(x) = -\frac{2j-1}{2j(2j+1)} x^2 A_{j-1}(x) + B_j(x),$$

$$B_j(x) = \frac{(2j-1)^2}{2j(2j+1)}(1-\rho^2)B_{j-1}(x)$$

with

$$B_0(x) = (2\pi)^{-1}(1-\rho^2)^{1/2} exp\left\{-\frac{x^2}{2(1-\rho^2)}\right\},$$

$$A_0(x) = -(2\pi)^{-1/2}|x|\phi\left(-\frac{|x|}{\sqrt{1-\rho^2}}\right) + B_0(x).$$

Chapter 3

Implementation and Comparison of the Bivariate Normal Algorithms

3.1 Preparations for the Implementation of the Algorithms

In order to fairly make our comparison of the algorithms, we select the same implementation conditions for each algorithm.

1. Benchmark Selection

The standard table of the bivariate normal probability $L(h, k; \rho)$ with a wide variable range (e.g., $h \in [-8, 8]$) and high accuracy (e.g., 15 significant digits), is not available in the literature. Hence a benchmark selection is necessary when we evaluate the performance of an algorithm. The Owen-Donnelly algorithm is well known for computation of the bivariate normal integral and has been considered to be the most widely implemented [22] with high accuracy (15 significant digits [9] [11]). There is mature code written in the FORTRAN language for this algorithm [9]. Therefore, it is selected as the benchmark in this project. We note that in this project the error is defined as the difference between $L(h,k;\rho)$ obtained by the Owen-Donnelly algorithm and by another algorithm. Another possible measure is the relative error, we however do not choose it since our main interest is for absolute error of the order of 10^{-6}.

2. Integration Method Selection

The Divgi, Drezner-Wesolowsky and Simon-Divsalar algorithms require the one-dimensional integration computation in this project. We employ the classical integration method, Simpson's rule, for the computation needed. The accuracy in Simpson's rule depends on the number of subintervals into which the range of the single integral is divided. More subintervals give greater accuracy, but more excessive computing time is consumed. Therefore, there is a tradeoff between the accuracy and efficiency of computation results. In this project, we choose $n = 200$ subintervals, since when $n > 200$, the integral results change more slowly compared with when $n < 200$.

3. Q-function Algorithm Selection

There are many Q-function algorithms which give very good approximations to $Q(x)$. We adopt the algorithm developed by Beaulieu [2], which has been found to be very efficient and accurate with a simple code. $Q(x)$ is expressed in Beaulieu's paper as:

$$Q(x) \doteq \frac{1}{2} - \frac{2}{\pi} \sum_{\substack{n=1 \\ n \text{ odd}}}^{\infty} \frac{exp(-n^2 w^2/2) sin(nwx)}{n},$$

where $w = 2\pi/T$. T=28 is empirically determined given in [2], and as paper suggested, we use 17 terms in the expansion.

4. The Computer Environment for the Implementation of Algorithms

All the algorithms are implemented on a Sun Solaris 7 UNIX operating system, Ultra SPARC 60 model with two 400MHz CPUs. Owen-Donnelly's algorithm was written by Donnelly [9] in the FORTRAN language. We used the macro f2c to convert the Donnelly FORTRAN code into C code, and we coded all the other algorithms in the C language.

3.2 Procedure Introduction

We used a 2890-point grid in which ρ ranges from -0.9 to 0.9 in increments of 0.2 and the variables h and k range from -8 to 8 in increment of 1. The algorithm implementations are described in the following subsections.

3.2.1 Procedure 1

First, we code the Drezner (D), Divgi (Di), Drezner-Wesolowsky (D-W), Simon-Divsalar (S-D) and Vasicek (V). algorithms in C. The Owen-Donnelly (O-D) algorithm is selected as the benchmark.

For each of the above algorithms, $L(h, k; \rho)$ is evaluated at every point on the grid by directly using the original algorithm, except the one by Simon-Divsalar, as mentioned in Chapter 2, which only can be used when $h \geq 0$; in this case, we use relationships (1.5) and (1.7) to evaluate $L(h, k; \rho)$ for $h < 0$.

We found that Divgi's algorithm performs poorly when $|h|$ or $|k| > 4$ and Vasicek's algorithm has lower accuracy compared with the other algorithms. The other four algorithms, O-D, D, W-D and S-D, yielded identical results to at least 7 decimal digits at each point on the grid (see Table 3.2). So we eliminate the Di and V algorithms, and focus on the other four algorithms. (see Fig. 3.1)

3.2.2 Procedure 2

When analyzing the results of procedure 1, we notice that the D algorithm achieves a mean error of less than 10^{-15} in the upper right quadrant for ρ=-0.9, -0.7, -0.5 and -0.3; the D-W algorithm performs best in this positive quadrant except when $\rho = 0.9$; while the S-D algorithm performs best in this positive quadrant except when ρ=0.9, 0.7 (see Table 3.1). Also, these four algorithms yielded results closer to one another in the upper right quadrant when $\rho \leq 0.5$, especially when $\rho \leq -0.3$. For $\rho \leq -0.3$, the value of $D_i(\rho)$ (explained below) in the upper right quadrant ($i = 1$) is between 1500 to 9000 times smaller than $D_i(\rho)$ in the other quadrants ($i = 2, 3, 4$), but is larger than $D_i(\rho)$ in the upper left and lower right quadrants when $\rho \geq 0.7$, though not more than 2 times larger. The $D_i(\rho)$ measure the closeness of the L-function results in the 4 quadrants by the four algorithms. It is defined as:

$$D_i(\rho) = \sum_{(h,k)\in Q(i)} \sum_{m\neq n} |L_{A_m}(h,k;\rho) - L_{A_n}(h,k;\rho)|/|Q(i)|,$$

where i=1 denotes the upper right quadrant, i=2 denotes the upper left quadrant, i=3 denotes the lower left quadrant, i=4 denotes the lower right quadrant, $Q(i)$ denotes the set of points (h,k) in the ith quadrant considered in project (not including points on the axes), $|Q(i)|$=64 denotes the number of points in $Q(i)$, and $L_{A_m}(\cdot)$ and $L_{A_n}(\cdot)$ denote the L-function calculated by the four algorithms A_m, A_m=O-D, D, D-W and S-D. The values of $D_i(\rho)$ in different quadrants for each ρ in Procedure 1 are plotted in Figure 3.1.

From the above analysis, overall the upper right quadrant is the best one in which all algorithms perform relatively better than in the other quadrants of the hk-plane, and it seems more possible that the values of $L(h,k;\rho)$ in this quadrant are closer to their true values. Based on this idea, we modify the code for each algorithm (including the benchmark), restricting h, k to nonnegative values, while ρ is still chosen from -0.9 to 0.9. Then we compute $L(h,k;\rho)$ for each algorithm in this restricted variable range by directly using the algorithms. Next we use the above computations to evaluate $L(h, k; \rho)$ in the other quadrants by using the relationships (1.5), (1.6), and (1.7) discussed in Chapter 1. Therefore, in procedure 2, the value of $L(h, k; \rho)$ for any point in the hk-plane is computed using the L-function $L(|h|, |k|; \rho)$ or $L(|h|, |k|; -\rho)$, and possibly the Q-function values $Q(|h|)$ and/or $Q(|k|)$.

ρ	abs.	D			D-W			S-D		
	err.	U-R	U-L L-R	L-L	U-R	U-L L-R	L-L	U-R	U-L L-R	L-L
-0.9	Avg.	lessE-15	1.55E-11	6.73E-15	2.17E-13	2.44E-11	2.42E-10	lessE-15	6.50E-10	2.47E-10
	Max.	lessE-15	9.87E-10	5.00E-14	1.33E-11	9.18E-10	1.97E-9	2.00E-14	1.53E-8	1.97E-9
-0.7	Avg.	lessE-15	4.63E-11	6.79E-15	5.37E-14	3.07E-12	2.41E-10	8.38E-15	2.12E-10	2.47E-10
	Max.	1.00E-15	9.87E-10	5.00E-14	2.64E-12	1.57E-10	1.97E-9	2.41E-13	6.53E-9	1.97E-9
-0.5	Avg.	lessE-15	6.18E-11	6.57E-15	3.23E-15	2.83E-12	2.41E-10	3.52E-14	1.72E-10	2.47E-10
	Max.	lessE-15	9.87E-10	5.00E-14	1.18E-13	1.57E-10	1.97E-9	5.99E-13	4.40E-9	1.97E-9
-0.3	Avg.	lessE-15	9.26E-11	6.78E-15	lessE-15	2.82E-12	2.41E-10	3.62E-14	1.18E-10	2.47E-10
	Max.	lessE-15	9.87E-10	5.00E-14	4.00E-15	1.57E-10	1.97E-9	4.58E-13	2.44E-9	1.97E-9
0.3	Avg.	6.17E-11	1.23E-10	5.25E-15	lessE-15	2.82E-12	2.41E-10	1.48E-13	1.83E-11	2.47E-10
	Max.	9.87E-10	9.87E-10	5.00E-14	8.00E-15	1.57E-10	1.97E-9	5.17E-12	6.52E-10	1.97E-9
0.5	Avg.	1.23E-10	1.23E-10	4.50E-15	9.27E-15	2.83E-12	2.41E-10	6.00E-13	4.48E-12	2.48E-10
	Max.	9.87E-10	9.87E-10	5.00E-14	2.71E-13	1.57E-10	1.97E-9	1.28E-11	1.84E-10	1.97E-9
0.7	Avg.	1.54E-10	1.23E-10	3.77E-15	2.41E-13	2.88E-12	2.42E-10	1.69E-12	7.88E-13	2.49E-10
	Max.	9.87E-10	9.87E-10	5.00E-14	7.03E-12	1.57E-10	1.97E-9	3.59E-11	4.25E-11	1.97E-9
0.9	Avg.	2.16E-10	1.23E-10	2.02E-15	2.16E-11	3.04E-12	2.63E-10	8.04E-12	1.33E-14	2.55E-10
	Max.	9.87E-10	9.87E-10	5.00E-14	9.18E-10	1.57E-10	1.97E-9	2.26E-10	8.50E-13	1.97E-9

Table 3.1: The mean errors in different quadrants for the D, D-W and S-D algorithms in procedure 1

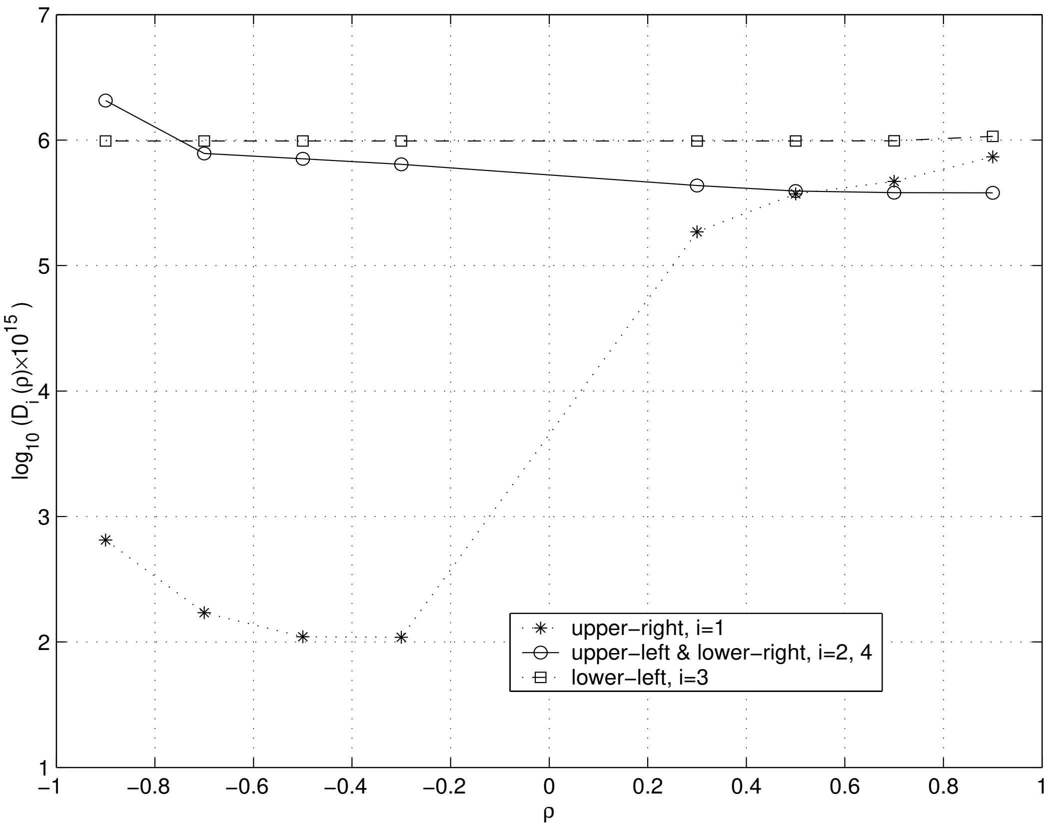

Figure 3.1: The values of $D_i(\rho)$ in different quadrants vs. ρ in procedure 1

If we neglect the error from the Q-function, for $h \geq 0$, $k \geq 0$, the error of $L(-h, -k; \rho)$ is the same as the error of $L(h, k, \rho)$, and both the errors of $L(h, -k; \rho)$ and $L(-h, k; \rho)$ are the same as the error from $L(h, k; -\rho)$. Thus, for a given ρ, the errors in procedure 2 distribute symmetrically about the lines $h = k$ and $h = -k$.

For those points on the coordinate axes, they can be considered as the points of their adjacent quadrants, hence the error of any point on the axes can be obtained by both the formulas for calculating the error in adjacent quadrants, one with the error of $L(|h|, |k|; \rho)$, and another with the error of $L(|h|, |k|; -\rho)$. When we design the code for the algorithms, we consider all the points located in the positive h axis and positive k axis (including the origin (0, 0)) as belonging to the upper right quadrant.

Therefore, assuming negligible error from the Q-function computation, for a given ρ, the total error over the whole hk-plane can be expressed as:

$$\begin{aligned} M(\rho) &= 2\sum_{h>0,k>0} |L_A(h,k;\rho) - L_{O-D}(h,k;\rho)| \\ &+ 2\sum_{h>0,k>0} |L_A(h,k;-\rho) - L_{O-D}(h,k;-\rho)| \\ &+ 4\sum_{h>0} |L_A(h,0;\rho) - L_{O-D}(h,0;\rho)| \\ &+ |L_A(0,0;\rho) - L_{O-D}(0,0;\rho)|, \end{aligned} \tag{3.1}$$

where $L_A(\cdot)$ denotes the L-function obtained by one of the algorithms A=D, D-W or S-D, and $L_{O-D}(\cdot)$ denotes the L-function obtained by the benchmark O-D algorithm. Clearly, the sum of the first two terms in (3.1) are the same for ρ and $-\rho$.

In our project, the D-W and S-D algorithms satisfy:

$$|L_A(h,0;\rho) - L_B(h,0;\rho)| = |L_A(h,0;-\rho) - L_B(h,0;-\rho)|,$$

where $h \geq 0$. However, the above equation does not hold for the D algorithm for some points on the axes. Hence, $M(\rho)$=$M(-\rho)$ is true for the D-W and S-D algorithms, and the means of the errors over the whole range of the variables h, k are the same for ρ and $-\rho$, but this is not true for the D algorithm.

3.3 Numerical Results and Analysis

In the last section, we introduced the procedures for implementing the bivariate normal probability function $L(h,k;\rho)$. In procedure 1, we calculated $L(h,k;\rho)$ by directly using the original methods, then we eliminated the Di and V algorithms and kept the other four algorithms for further computations. Next, by using the relationships (1.5) to (1.7) in Chapter 1, we spread the computation results of $L(h,k;\rho)$ in the upper right quadrant to the whole range of the hk-plane. In this section, we present performance analysis and comparison for the O-D, D, D-W and S-D algorithms with regard to accuracy and efficiency.

3.3.1 Accuracy Analysis

Table 3.2 provides the mean of absolute errors in procedure 1 for each algorithm when ρ is -0.9, -0.7, -0.5 -0.3, 0.3, 0.5, 0.7, and 0.9, respectively. The maximum error and standard deviation also are given in the same table. As mentioned, the error refers to the relative error, since it is not the error compared with the true value, but with the benchmark.

In procedure 1, over the whole range of h, k and ρ, the maximum error for the D, D-W and S-D algorithms is 9.8659×10^{-10}, 1.9732×10^{-9}, 1.5160×10^{-8}, respectively. Therefore, all these four algorithms achieve identical results to at least 7 decimal digits for each point on the grid. When considering the mean of the absolute errors for each ρ, the D algorithm is not worse than 1.0938×10^{-10}; the D-W algorithm is not worse than 7.7226×10^{-11}; and the S-D algorithm is not worse than 3.5001×10^{-10}. All of these maximum mean absolute errors happen when $|\rho|$ is large (i.e., ρ =0.9 and -0.9).

Based on the maximum mean absolute error, in procedure 1, the rank order from most accurate to least accurate is D-W, D, S-D. The performance of the D-W algorithm is stable in the sense that it is not affected too much by the correlation ρ. The S-D algorithm performs worse when $\rho < 0$, while the D algorithm is worse when $\rho > 0$ (see Fig. 3.2).

Table 3.3 presents analogous results for procedure 2 as the results in Table 3.2 for procedure 1. As mentioned before, the means of errors over the whole range of the variables h, k are the same for ρ and -ρ for the D-W and S-D algorithms, but are different for the D algorithm. We provide exactly the same values for both positive and negative $\rho's$ for the D-W and S-D algorithms, and different values for the D algorithm in Table 3.3.

Comparing the three algorithms with the benchmark, the maximum absolute error in the D-W algorithm is 1.5070×10^{-9}. From Figures 3.5 and 3.32, it can be seen that these large errors occur only at two points: $(0, 0; 0.9)$ and $(0, 0; -0.9)$. Except at these two points, all the algorithms at the other points on the grid achieve identical results to at least 9 decimal digits. Among these three algorithms, in procedure 2, the D-W algorithm performs the best when $|\rho| \leq 0.7$, the S-D algorithm performs the best for $|\rho|$ >0.7, and the D algorithm performs the worst over the whole range of ρ, except when ρ=-0.1, where it performs just as well as the D-W algorithm (see Table 3.3 and Figure 3.3).

The mean of the absolute errors for each algorithm and each ρ for both procedures 1 and 2 are plotted separately in Figures 3.2 and 3.3. Comparing these two figures, the curves for the D-W and S-D algorithms are much closer to the benchmark in procedure 2 than they are in procedure 1; therefore, the performance is much improved for these two algorithms after applying procedure 2. The performance for the D algorithm is slightly improved when $\rho \geq -0.3$, but degraded for $\rho \leq -0.5$.

From Figure 3.3 we may see the performance of all algorithms in procedure 2 becomes better when $|\rho|$ becomes smaller. The D-W and S-D algorithms have an abrupt increase from $|\rho| \geq 0.7$. The most efficient gain of the performance from procedure 2 goes to the S-D algorithm.

Figures 3.4 to 3.18 illustrate the error distribution over the whole range of variables h, k for each ρ in procedure 2. The analogous figures for procedure 1 are given in Appendix A. In these figures the different symbols denote the different error levels. Each plot in procedure 2 is symmetric about the lines $h = k$ and $h = -k$; while in procedure 1 is symmetric about the line $h = k$ only. In procedure 2, the plots of the error distribution are similar for ρ and $-\rho$; basically, one can be obtained by rotating the other by $\pi/2$

radians clockwise or counterclockwise; namely, the error at a point $(h, k; \rho)$ on the grid is essentially the same as the error at the point $(-k, h; -\rho)$, $(h, -k; -\rho)$, $(k, -h; -\rho)$ or $(-h, k; -\rho)$. However, as mentioned, some points on the axes for the D algorithm do not satisfy this property because this algorithm yields different results for $L(h, k, \rho)$ and $L(h, k, -\rho)$ at these points on the axes.

The errors in procedure 2 for the D algorithm are congregated in the areas where $|h|$ or $|k|$ is large (i.e., 7 and 8). If we restrict the range of both h and k to [-6,6], then the results of the D algorithm match the benchmark (see Figures 3.4, 3.7, 3.10, 3.13, 3.16, 3.19, 3.22, 3.25, 3.28, 3.31) to within 10^{-14}.

So far, we have discussed the performance in accuracy for each algorithm in procedure 1 and procedure 2. From the above analysis of the overall performance of the D, D-W and S-D algorithms in both procedures, the computation method of procedure 2 is a simple and efficient method to improve the accuracy for evaluating the L-function over the entire range of h, k considered. In the next section, we will discuss the performance in efficiency for each algorithm in procedure 2.

ρ	Algorithm	Mean of errors	Max of errors	Std. Dev.
ρ=-0.9	D	6.8480×10^{-12}	9.8659×10^{-10}	8.1787×10^{-11}
	D-W	7.7226×10^{-11}	1.9732×10^{-9}	2.6388×10^{-10}
	S-D	3.5001×10^{-10}	1.5160×10^{-8}	1.6787×10^{-9}
ρ=-0.7	D	2.0521×10^{-11}	9.8659×10^{-10}	1.4067×10^{-10}
	D-W	5.8400×10^{-11}	1.9732×10^{-9}	2.3833×10^{-10}
	S-D	1.5595×10^{-10}	6.5282×10^{-9}	6.3264×10^{-10}
ρ=-0.5	D	2.7357×10^{-11}	9.8659×10^{-10}	1.6185×10^{-10}
	D-W	5.8123×10^{-11}	1.9732×10^{-9}	2.3839×10^{-10}
	S-D	1.3783×10^{-10}	4.4022×10^{-9}	5.2150×10^{-10}
ρ=-0.3	D	4.1021×10^{-11}	9.8659×10^{-10}	1.9681×10^{-10}
	D-W	5.8112×10^{-11}	1.9732×10^{-9}	2.3840×10^{-10}
	S-D	1.1391×10^{-10}	2.4418×10^{-9}	3.9864×10^{-10}
ρ=0.3	D	7.5202×10^{-11}	9.8659×10^{-10}	2.6161×10^{-10}
	D-W	5.8112×10^{-11}	1.9732×10^{-9}	2.3840×10^{-10}
	S-D	6.9865×10^{-11}	1.9732×10^{-9}	2.5721×10^{-10}
ρ=0.5	D	8.8866×10^{-11}	9.8659×10^{-10}	2.8226×10^{-10}
	D-W	5.8123×10^{-11}	1.9732×10^{-9}	2.3839×10^{-10}
	S-D	6.3893×10^{-11}	1.9732×10^{-9}	2.5211×10^{-10}
ρ=0.7	D	9.5702×10^{-11}	9.8659×10^{-10}	2.9180×10^{-10}
	D-W	5.8400×10^{-11}	1.9732×10^{-9}	2.3833×10^{-10}
	S-D	6.3247×10^{-11}	1.9732×10^{-9}	2.5184×10^{-10}
ρ=0.9	D	1.0938×10^{-10}	9.8659×10^{-10}	3.0954×10^{-10}
	D-W	7.7225×10^{-11}	1.9732×10^{-9}	2.6388×10^{-10}
	S-D	6.5751×10^{-11}	1.9732×10^{-9}	2.5209×10^{-10}

Table 3.2: Accuracy results for the D, D-W and S-D algorithms in procedure 1

ρ	Algorithm	Mean of errors	Max of errors	Std. Dev.
ρ=0.9 or -0.9	D (0.9)	1.0937×10^{-10}	9.8659×10^{-10}	3.1008×10^{-10}
	D (-0.9)	9.5693×10^{-11}	9.8659×10^{-10}	2.9231×10^{-10}
	D-W	1.9114×10^{-11}	1.5070×10^{-9}	1.2277×10^{-10}
	S-D	4.2170×10^{-12}	2.2597×10^{-10}	2.1896×10^{-11}
ρ=0.7 or -0.7	D (0.7)	8.2020×10^{-11}	9.8659×10^{-10}	2.7270×10^{-11}
	D (-0.7)	6.8347×10^{-11}	9.8659×10^{-10}	2.5081×10^{-10}
	D-W	2.8900×10^{-13}	1.1704×10^{-11}	1.2640×10^{-12}
	S-D	1.3740×10^{-12}	5.2204×10^{-10}	5.5430×10^{-12}
ρ=0.5 or -0.5	D (0.5)	6.8347×10^{-11}	9.8659×10^{-10}	2.5081×10^{-10}
	D (-0.5)	4.1019×10^{-11}	9.8659×10^{-10}	1.9715×10^{-10}
	D-W	1.2000×10^{-14}	4.9700×10^{-13}	5.3000×10^{-14}
	S-D	3.9600×10^{-13}	1.2839×10^{-11}	1.6300×10^{-12}
ρ=0.3 or -0.3	D (0.3)	4.1019×10^{-11}	9.8659×10^{-10}	1.9715×10^{-10}
	D (-0.3)	2.7346×10^{-11}	9.8659×10^{-10}	1.6213×10^{-10}
	D-W	1.0000×10^{-15}	1.8000×10^{-14}	2.0000×10^{-15}
	S-D	2.6600×10^{-13}	7.0850×10^{-12}	1.0350×10^{-12}
ρ=0.1 or -0.1	D (0.1)	1.3673×10^{-12}	9.8659×10^{-10}	1.1546×10^{-10}
	D (-0.1)	$< 10^{-15}$	1.0000×10^{-15}	$< 10^{-15}$
	D-W	$< 10^{-15}$	1.2000×10^{-14}	2.0000×10^{-15}
	S-D	2.0400×10^{-13}	5.0350×10^{-12}	6.8100×10^{-13}

Table 3.3: Accuracy results for the D, D-W and S-D algorithms in procedure 2

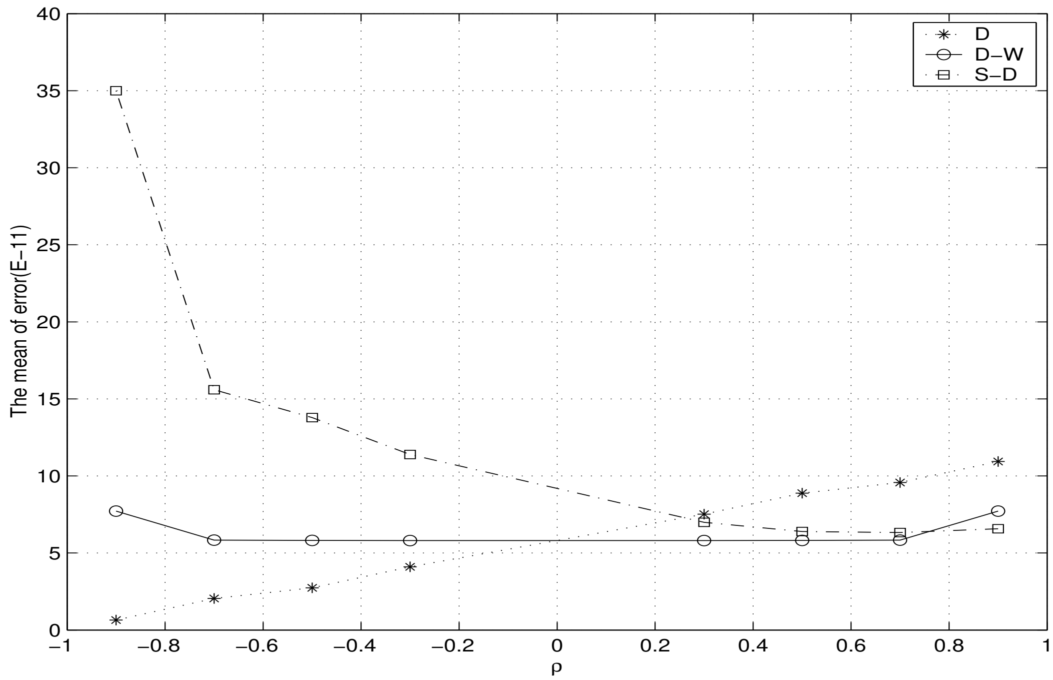

Figure 3.2: The mean of error vs. correlation in procedure 1

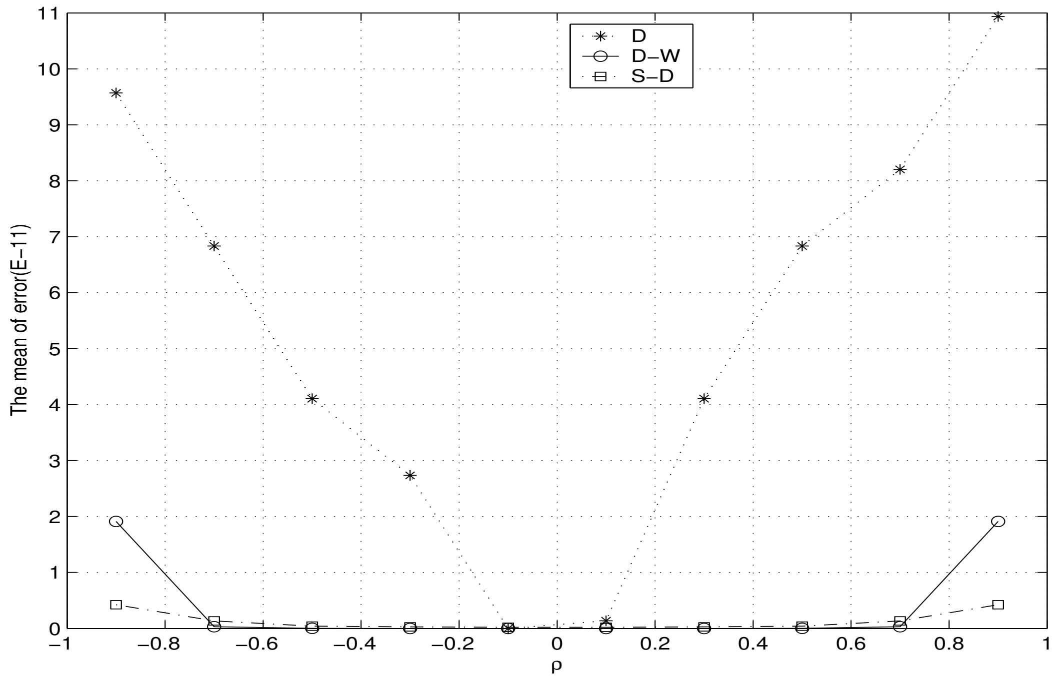

Figure 3.3: The mean of error vs correlation in procedure 2

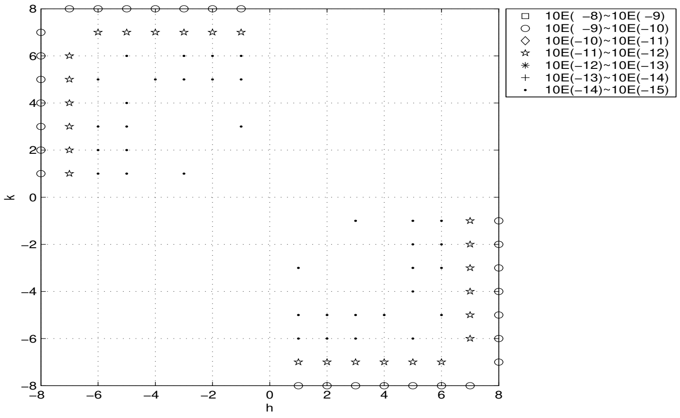

Figure 3.4: The error distribution for Drezner algorithm with $\rho = -0.9$

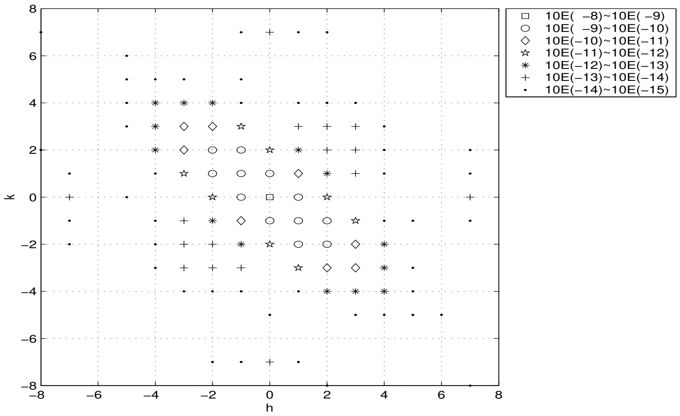

Figure 3.5: The error distribution for Drezner-Wesolowsky algorithm with $\rho = -0.9$

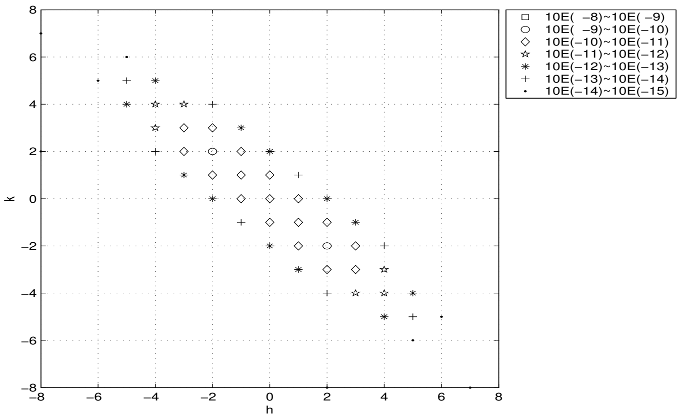

Figure 3.6: The error distribution for Simon-Divsalar algorithm with $\rho = -0.9$

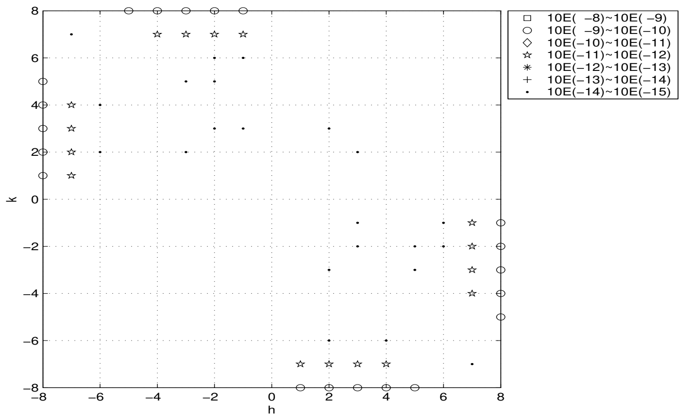

Figure 3.7: The error distribution for Drezner algorithm with $\rho = -0.7$

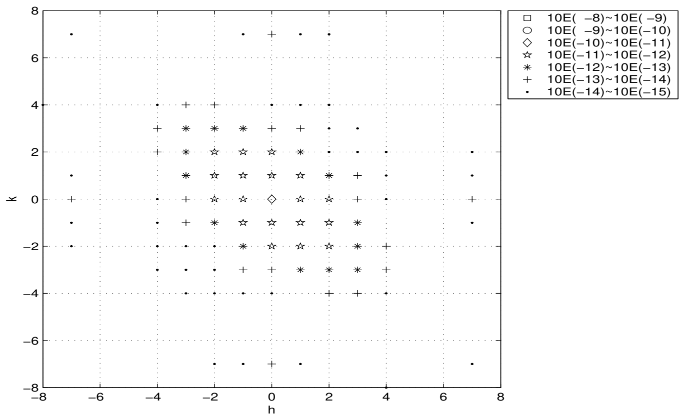

Figure 3.8: The error distribution for Drezner-Wesolowsky algorithm with $\rho = -0.7$

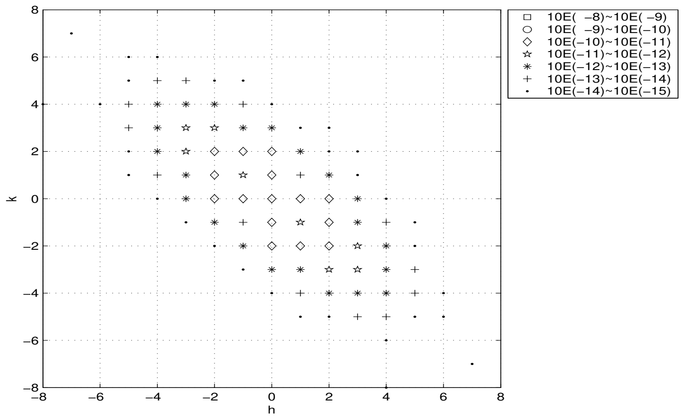

Figure 3.9: The error distribution for Simon-Divsalar algorithm with $\rho = -0.7$

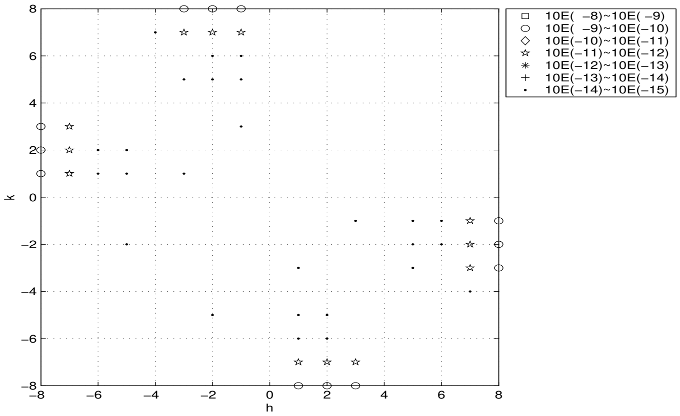

Figure 3.10: The error distribution for Drezner algorithm with $\rho = -0.5$

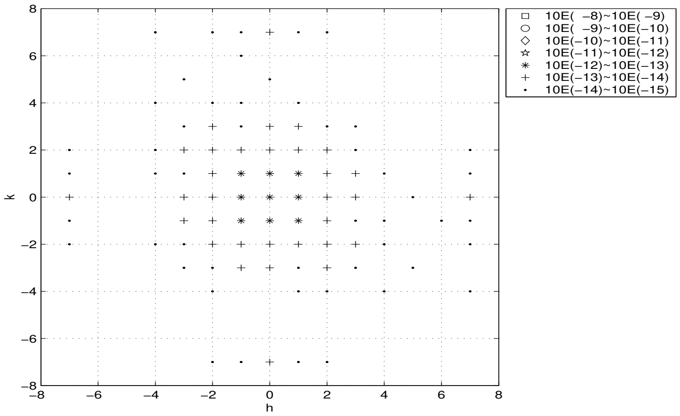

Figure 3.11: The error distribution for Drezner-Wesolowsky algorithm with $\rho = -0.5$

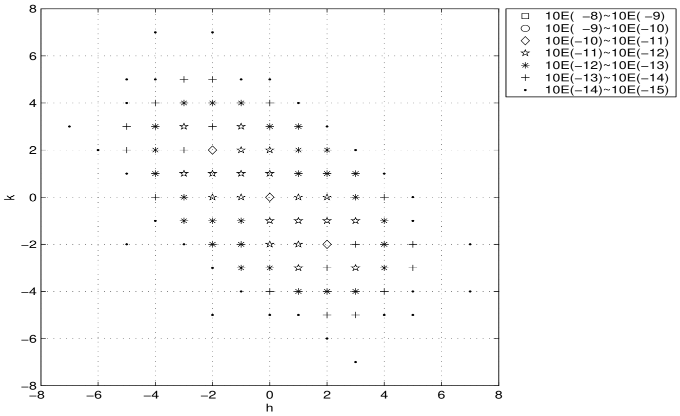

Figure 3.12: The error distribution for Simon-Divsalar algorithm with $\rho = -0.5$

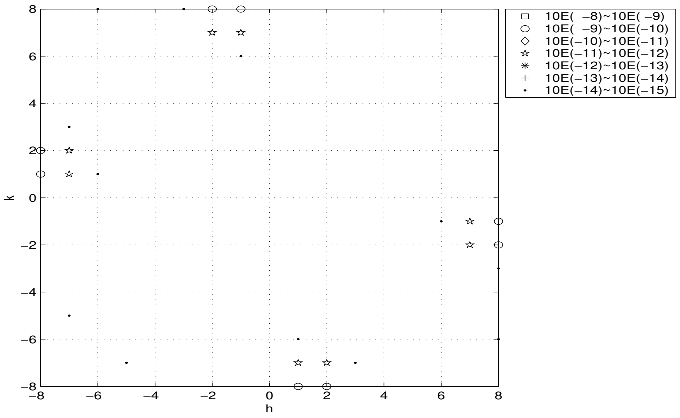

Figure 3.13: The error distribution for Drezner algorithm with $\rho = -0.3$

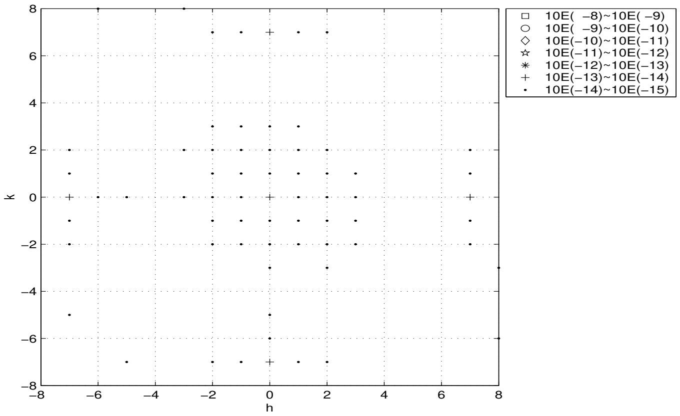

Figure 3.14: The error distribution for Drezner-Wesolowsky algorithm with $\rho = -0.3$

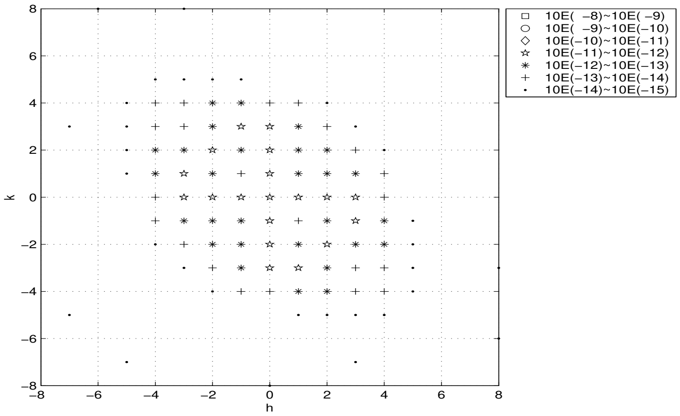

Figure 3.15: The error distribution for Simon-Divsalar algorithm with $\rho = -0.3$

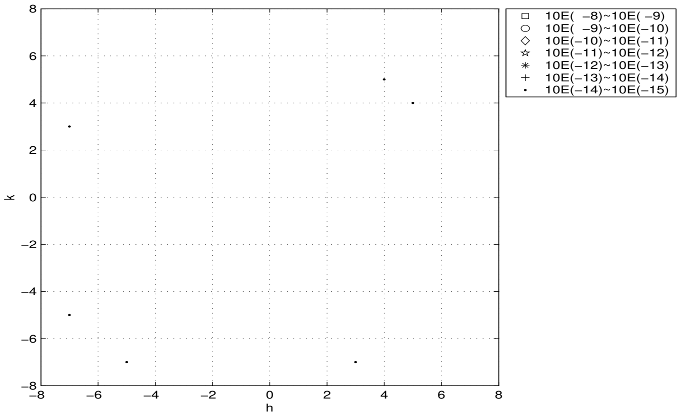

Figure 3.16: The error distribution for Drezner algorithm with $\rho = -0.1$

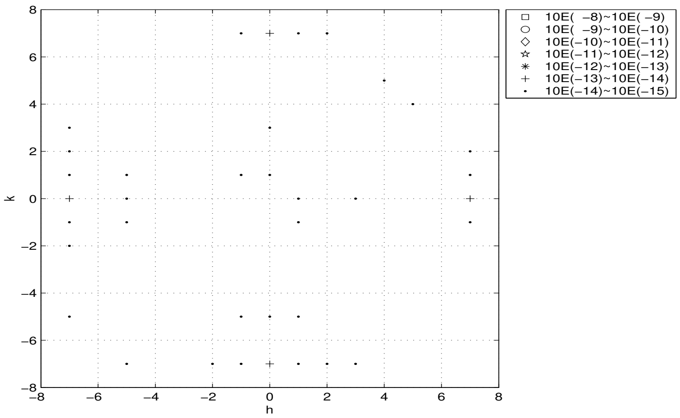

Figure 3.17: The error distribution for Drezner-Wesolowsky algorithm with $\rho = -0.1$

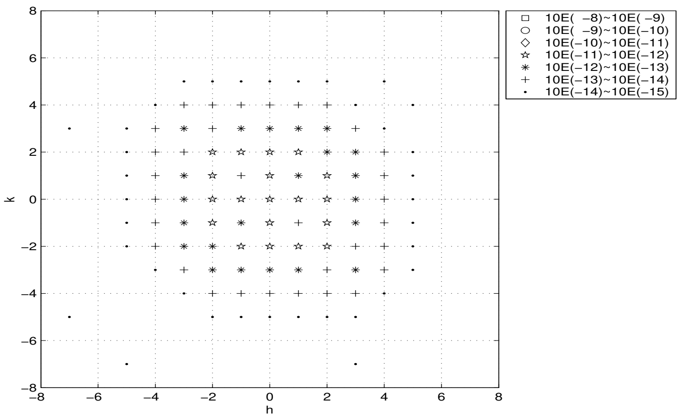

Figure 3.18: The error distribution for Simon-Divsalar algorithm with $\rho = -0.1$

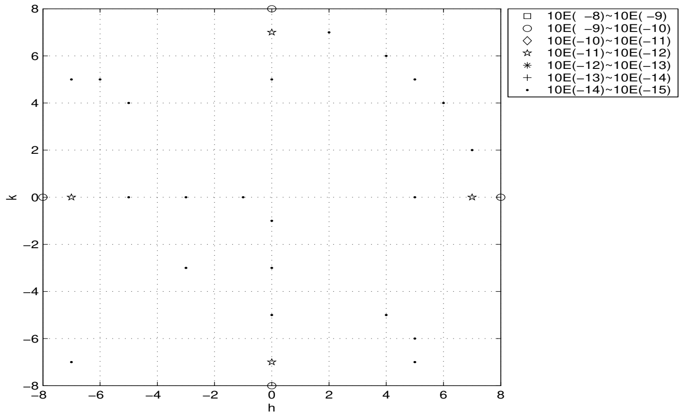

Figure 3.19: The error distribution for Drezner algorithm with $\rho = 0.1$

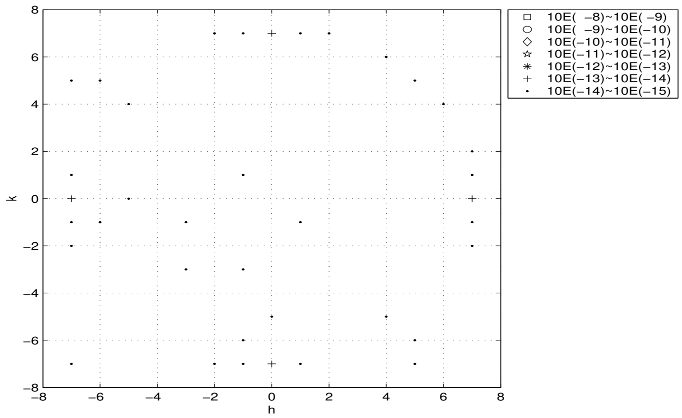

Figure 3.20: The error distribution for Drezner-Wesolowsky algorithm with $\rho = 0.1$

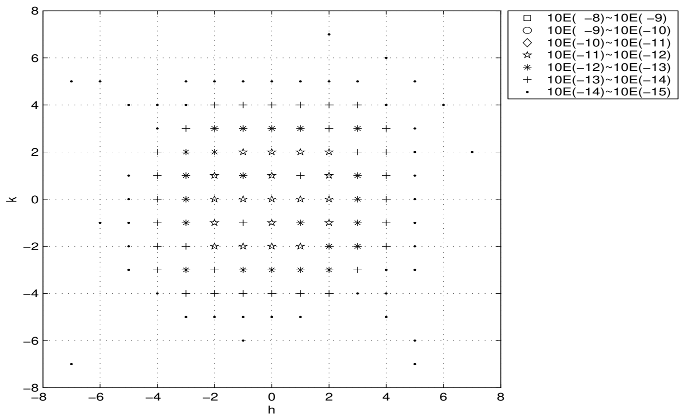

Figure 3.21: The error distribution for Simon-Divsalar algorithm with $\rho = 0.1$

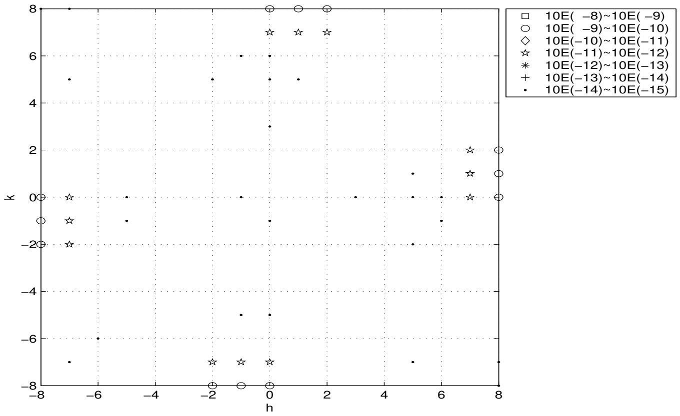

Figure 3.22: The error distribution for Drezner algorithm with $\rho = 0.3$

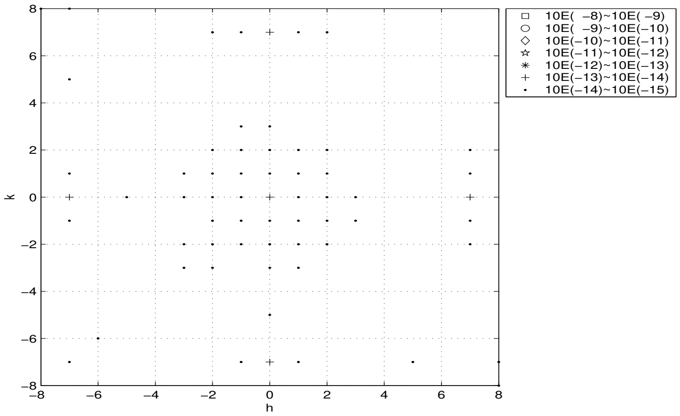

Figure 3.23: The error distribution for Drezner-Wesolowsky algorithm with $\rho = 0.3$

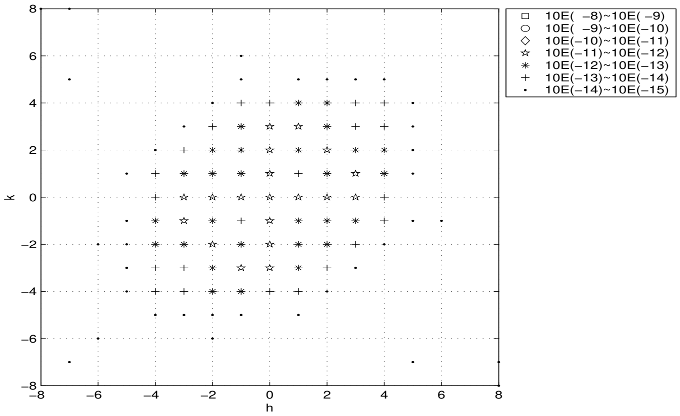

Figure 3.24: The error distribution for Simon-Divsalar algorithm with $\rho = 0.3$

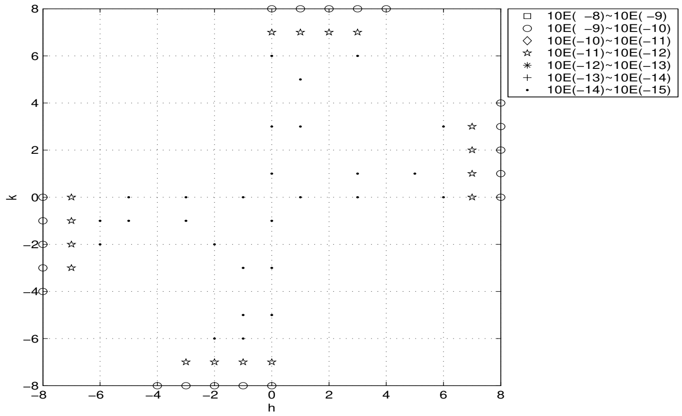

Figure 3.25: The error distribution for Drezner algorithm with $\rho = 0.5$

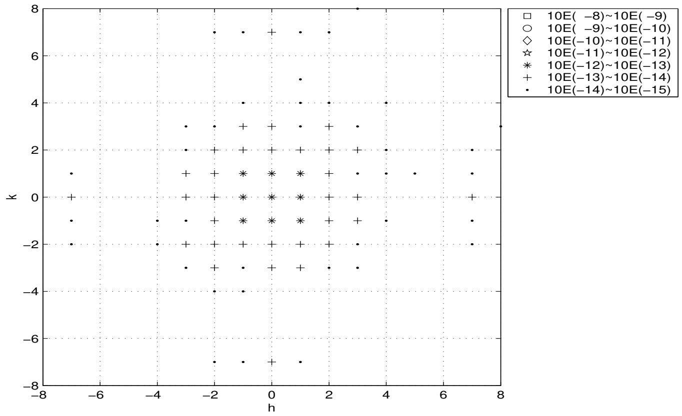

Figure 3.26: The error distribution for Drezner-Wesolowsky algorithm with $\rho = 0.5$

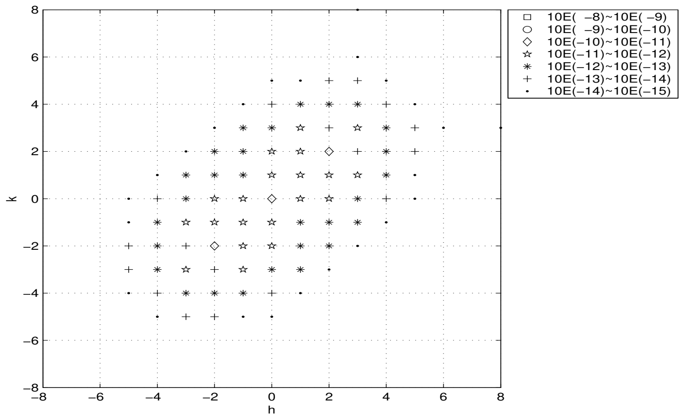

Figure 3.27: The error distribution for Simon-Divsalar algorithm with $\rho = 0.5$

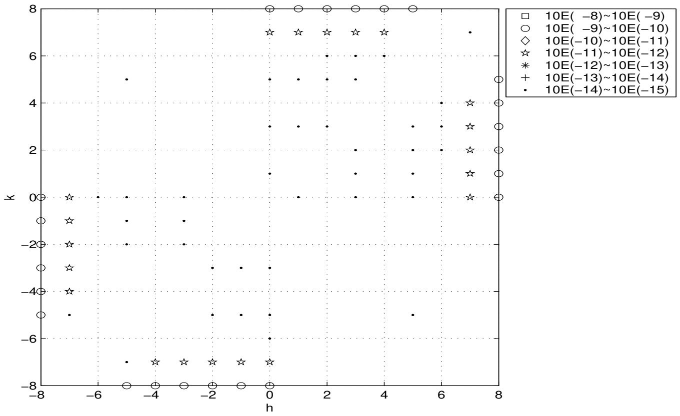

Figure 3.28: The error distribution for Drezner algorithm with $\rho = 0.7$

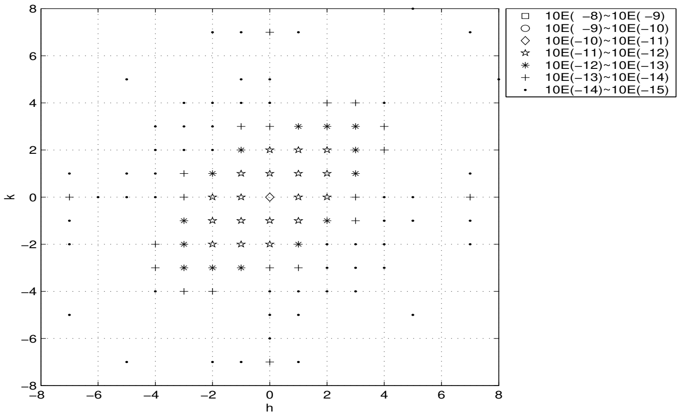

Figure 3.29: The error distribution for Drezner-Wesolowsky algorithm with $\rho = 0.7$

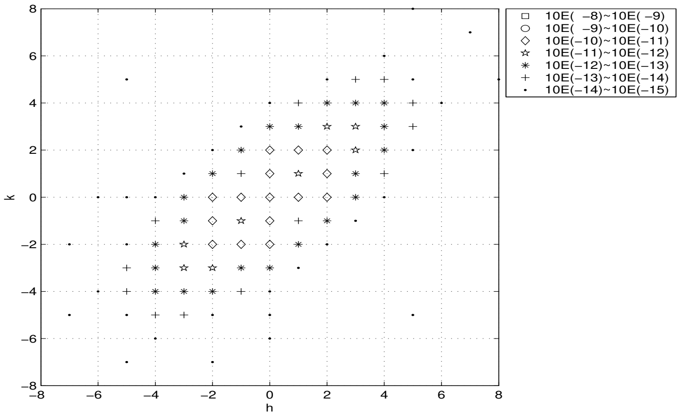

Figure 3.30: The error distribution for Simon-Divsalar algorithm with $\rho = 0.7$

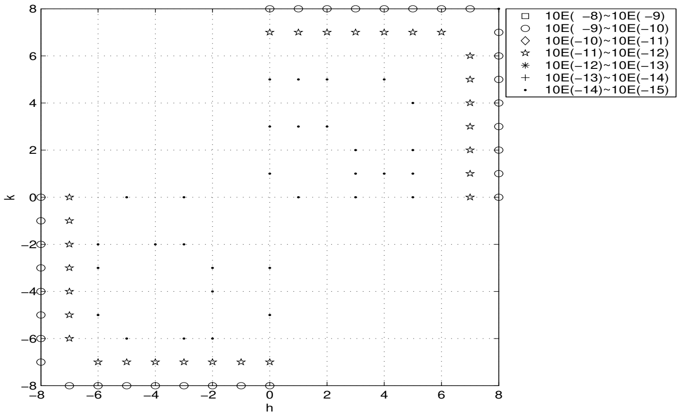

Figure 3.31: The error distribution for Drezner algorithm with $\rho = 0.9$

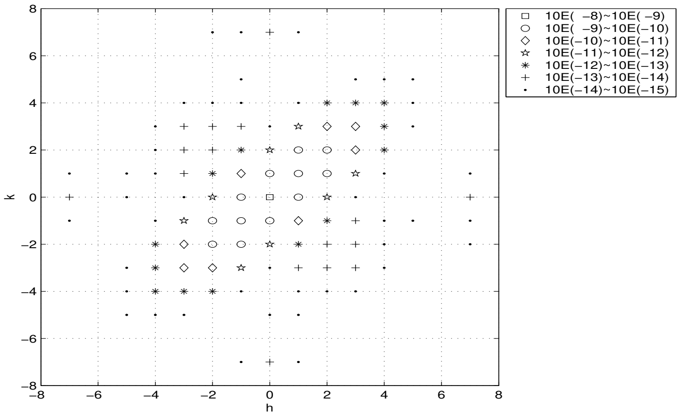

Figure 3.32: The error distribution for Drezner-Wesolowsky algorithm with $\rho = 0.9$

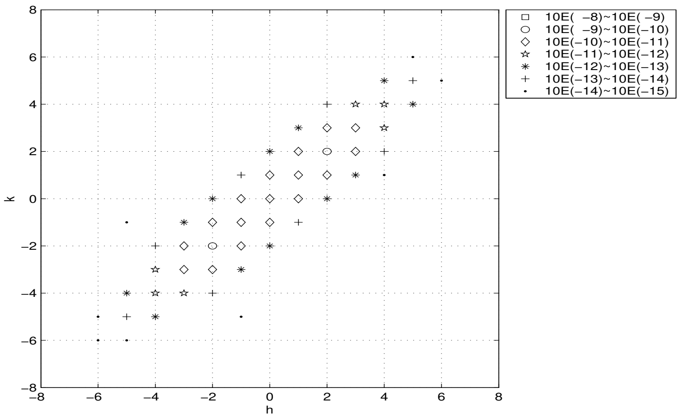

Figure 3.33: The error distribution for Simon-Divsalar algorithm with $\rho = 0.9$

3.3.2 Efficiency Analysis

The function $L(h, k; \rho)$ in the right upper quadrant of the hk-plane plays an important role in our procedure 2. Based on it, we implement the computations of $L(h, k; \rho)$ over the whole variable range in the hk-plane. In this section, we concentrate on analyzing the performance in efficiency for each algorithm in this nonnegative quadrant (we include the axis points in this quadrant).

We construct another 65610-point grid on which h, k ranges from 0 to 8 in increments of 0.1, and ρ ranges from -0.9 to 0.9 in increments of 0.2. For each of the O-D, D, D-W and S-D algorithms, for a given ρ, we calculate $L(h, k; \rho)$ at each point on the grid corresponding to this ρ over the nonnegative quadrant, then record the total system CPU time (in units of 10ms) consumed for computing $L(h, k; \rho)$ at these 6561 points. We repeat the same calculation for each algorithm and each ρ ten times, and then take the average system CPU time for each point. According to the above procedure, for a given ρ, the average system CPU time at a point on the grid for an algorithm A (A=O-D, D, D-W, S-D) can be expressed:

$$\bar{t}_A(\rho) = \frac{1}{10} \sum_{n=1}^{10} \sum_{(h,k)} t_n(h, k; \rho)/6561,$$

where $t_n(h, k; \rho)$ refers to the system CPU time to calculate the L-function at the point $(h, k; \rho)$ during the n^{th} experiment.

Table 3.4 provides the values of the average system CPU time $\bar{t}_A(\rho)$ for each algorithm and each ρ. They are computed on a Sun Solaris 7 UNIX operating system, Ultra SPARC 60 model with two 400MHz CPUs. The average system CPU time for each algorithm over all ρ also is given in the last line of Table 3.4. The fastest algorithm is the O-D algorithm, about 8 times faster than the D-W algorithm, 39 times faster than the S-D algorithm, 26 times faster than the D algorithm when $\rho > 0$, and 48 times faster than the D algorithm when $\rho < 0$. All algorithms we coded are slower than the benchmark,

ρ	O-D	D	D-W	S-D
-0.9	2.13×10^{-5}	6.43×10^{-4}	1.75×10^{-4}	1.01×10^{-3}
-0.7	2.13×10^{-5}	5.09×10^{-4}	1.75×10^{-4}	8.51×10^{-4}
-0.5	2.29×10^{-5}	5.26×10^{-4}	1.75×10^{-4}	8.09×10^{-4}
-0.3	2.29×10^{-5}	5.26×10^{-4}	1.75×10^{-4}	7.90×10^{-4}
-0.1	2.29×10^{-5}	5.26×10^{-4}	1.75×10^{-4}	7.79×10^{-4}
0.1	1.98×10^{-5}	1.03×10^{-3}	1.75×10^{-4}	7.74×10^{-4}
0.3	1.98×10^{-5}	1.04×10^{-3}	1.75×10^{-4}	7.74×10^{-4}
0.5	1.83×10^{-5}	1.05×10^{-3}	1.75×10^{-4}	7.80×10^{-4}
0.7	1.98×10^{-5}	1.05×10^{-3}	1.75×10^{-4}	7.96×10^{-4}
0.9	1.98×10^{-5}	1.04×10^{-3}	1.75×10^{-4}	8.43×10^{-4}
Avg.	2.09×10^{-5}	7.91×10^{-4}	1.75×10^{-4}	8.21×10^{-4}

Table 3.4: Average system CPU time (in seconds) $\bar{t}_A(\rho)$

one of the possible reasons is that the C codes for these algorithms are not optimized as the benchmark's.

It is interesting to note that the Owen-Donnelly (O-D) algorithm was developed long before the other algorithms, but its performance in terms of both accuracy and efficiency holds up quite well.

Chapter 4

Summary

The major goal of this work is to implement various algorithms for evaluating the bivariate normal probability function $L(h,k;\rho)$, and to compare those algorithms in terms of accuracy and efficiency.

We began with a studying typical $L(h,k;\rho)$ algorithms which have been developed in the past 40 years, such as the Owen-Donnelly(O-D), Divgi(Di), Drezner(D), Drezner-Wesolowsky(D-W), Simon-Divsalar(S-D) and Vasicek(V) algorithms.

We then implemented the above algorithms by directly coding the original algorithms in C (except for the O-D algorithm, which we downloaded as Fortran code then converted to C code). The O-D, D, D-W and S-D algorithms yielded identical results to at least 7 decimal digits over whole range of the variables h, k and ρ. The Di algorithm performed poorly when the limit h or k of integration was more than 4; and the V algorithm had lower accuracy compared with the other algorithms over the whole variable range. Based on the maximum mean absolute error, the rank order from the most accurate to the least accurate algorithms is D-W, D, S-D, using the O-D algorithm as the benchmark.

Next we focused on the O-D, D, D-W and S-D algorithms. Since overall the upper right quadrant in the hk-plane is the best one in which the algorithms perform relatively better, we spread the computation results of $L(h, k; \rho)$ in this positive quadrant to the other quadrants of the hk-plane by using relationships from which we can convert the L-function from one quadrant to the another. The performance is much improved after the above computation for the S-D and D-W algorithms over the whole argument range. For the D algorithm the performance is improved when $\rho \geq -0.3$. Compared with the benchmark O-D algorithm, the D algorithm performed the worst in general, the D-W algorithm performed the best among the three algorithms when $|\rho| \leq 0.7$, and the S-D algorithm most efficiently gained from this further computation.

The O-D algorithm performed the best in efficiency, consuming an average of 2.09×10^{-5} seconds of system CPU time to finish $L(h, k; \rho)$ at one point, which is 8 times faster than the nearest competitor, the D-W algorithm, and about 39 times faster than the slowest algorithm, the S-D algorithm. Although the O-D algorithm was published as the earliest one among all the algorithms discussed in this project, it performed remarkably well in both accuracy and efficiency.

Software to support all the computations in this project was written in the C programming language. It can be used to compute the L-function at any point $(h, k; \rho)$ over a wide variable range by using any of the above algorithms.

Appendix A

The error distribution for D, D-W and S-D algorithms in Procedure 1

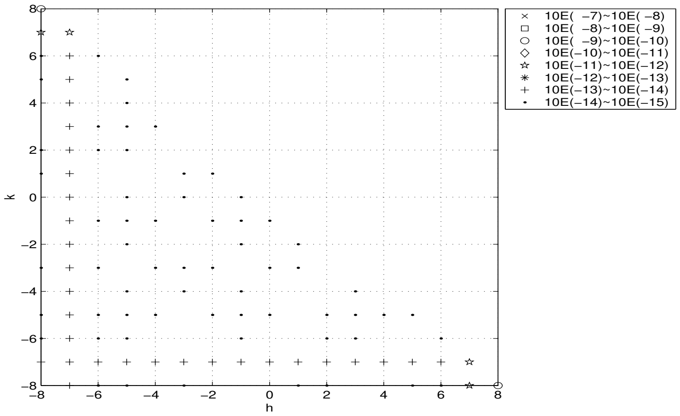

Figure A.1: The error distribution for D algorithm with $\rho = -0.9$ in Procedure 1

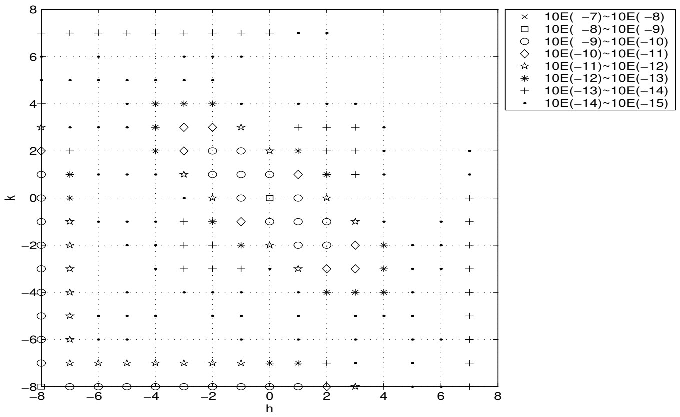

Figure A.2: The error distribution for D-W algorithm with $\rho = -0.9$ in Procedure 1

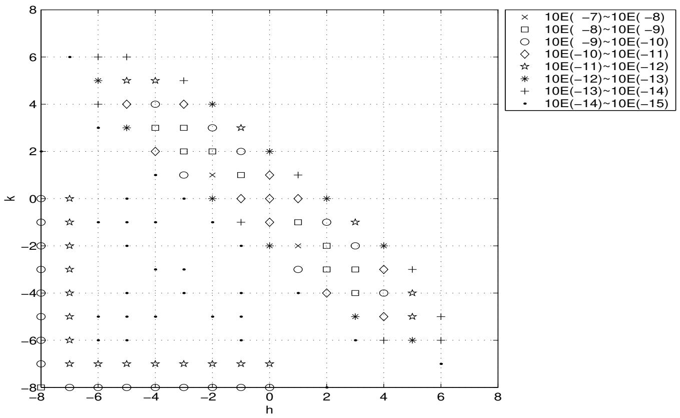

Figure A.3: The error distribution for S-D algorithm with $\rho = -0.9$ in Procedure 1

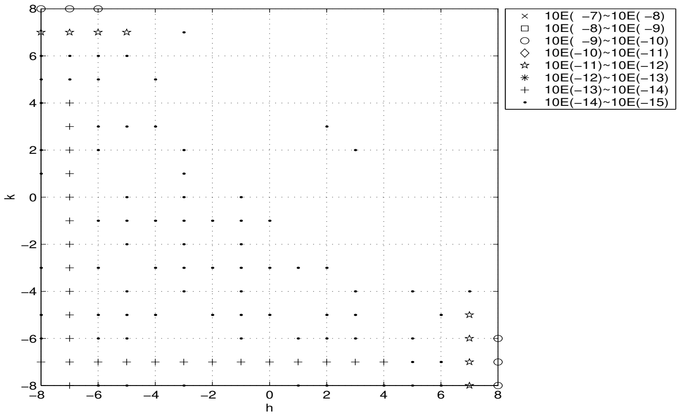

Figure A.4: The error distribution for D algorithm with $\rho = -0.7$ in Procedure 1

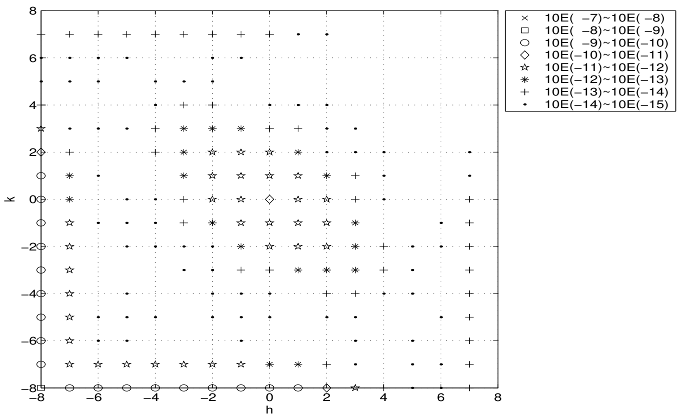

Figure A.5: The error distribution for D-W algorithm with $\rho = -0.7$ in Procedure 1

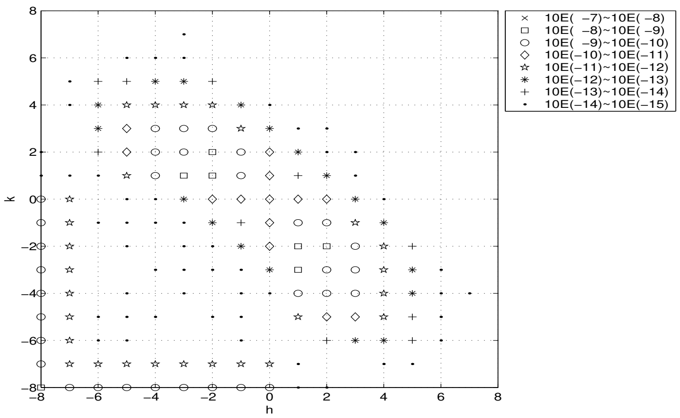

Figure A.6: The error distribution for S-D algorithm with $\rho = -0.7$ in Procedure 1

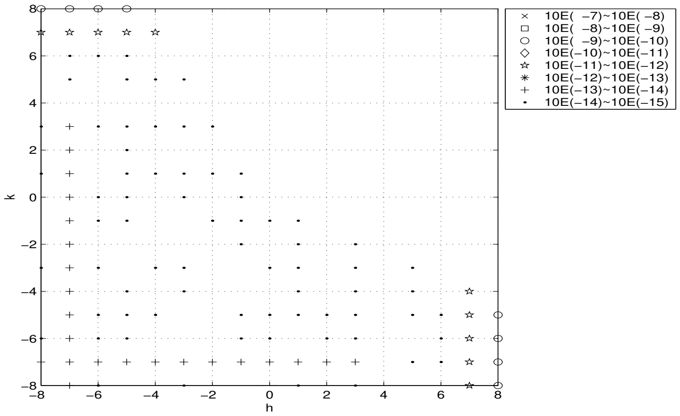

Figure A.7: The error distribution for D algorithm with $\rho = -0.5$ in Procedure 1

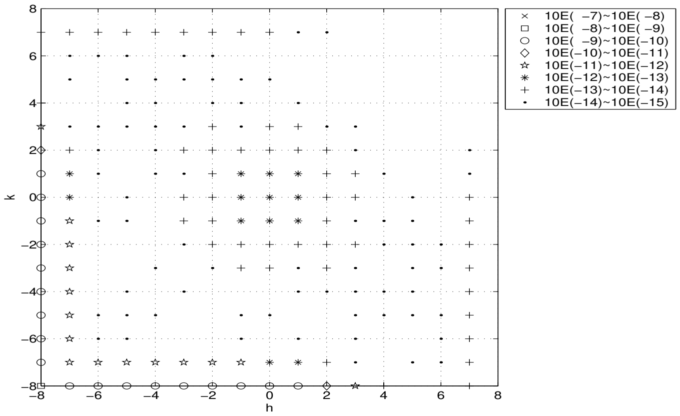

Figure A.8: The error distribution for D-W algorithm with $\rho = -0.5$ in Procedure 1

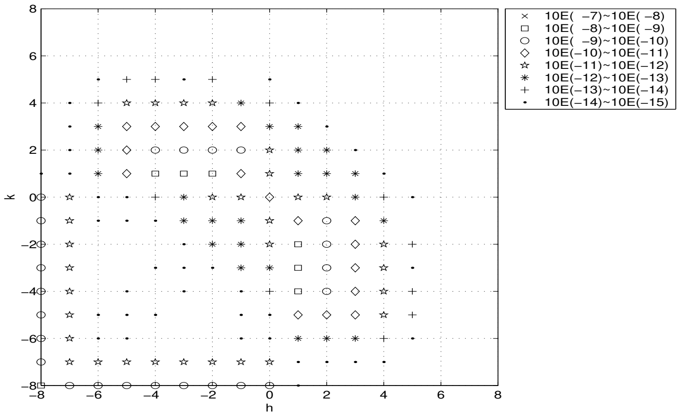

Figure A.9: The error distribution for S-D algorithm with $\rho = -0.5$ in Procedure 1

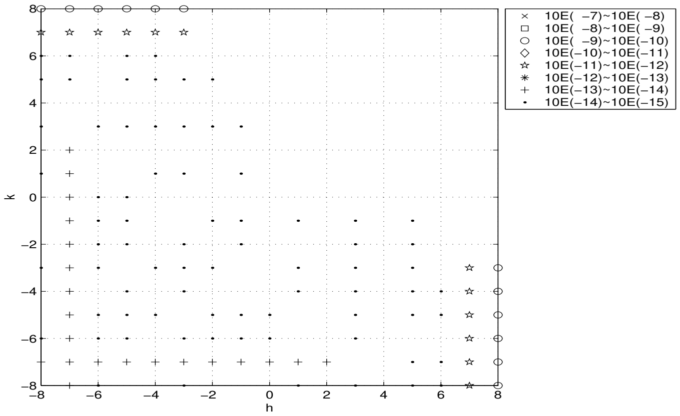

Figure A.10: The error distribution for D algorithm with $\rho = -0.3$ in Procedure 1

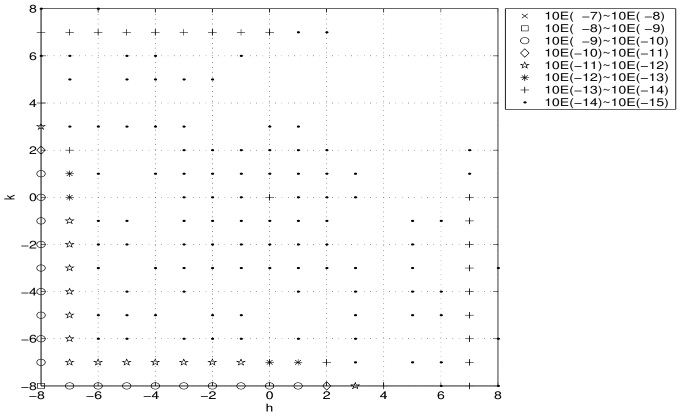

Figure A.11: The error distribution for D-W algorithm with $\rho = -0.3$ in Procedure 1

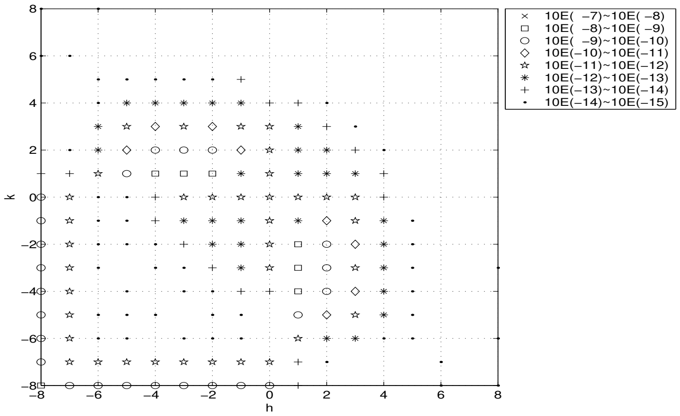

Figure A.12: The error distribution for S-D algorithm with $\rho = -0.3$ in Procedure 1

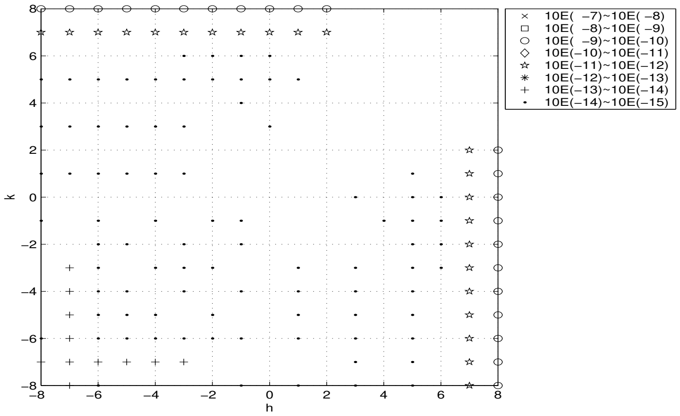

Figure A.13: The error distribution for D algorithm with $\rho = 0.3$ in Procedure 1

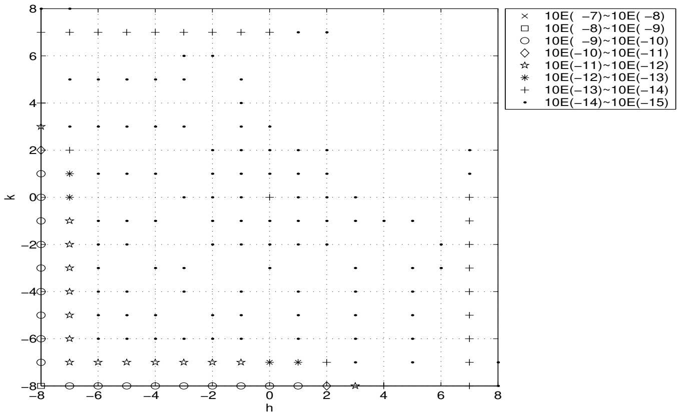

Figure A.14: The error distribution for D-W algorithm with $\rho = 0.3$ in Procedure 1

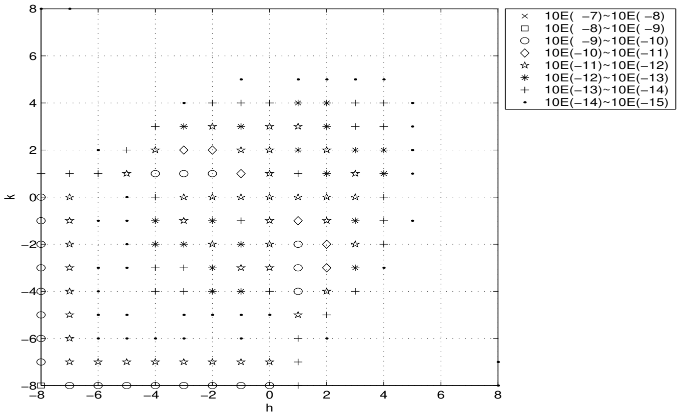

Figure A.15: The error distribution for S-D algorithm with $\rho = 0.3$ in Procedure 1

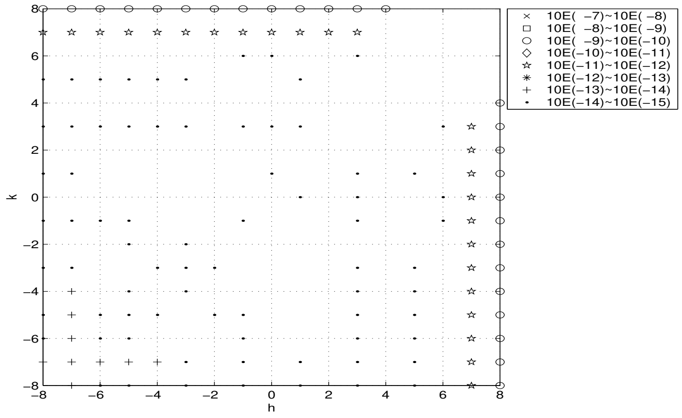

Figure A.16: The error distribution for D algorithm with $\rho = 0.5$ in Procedure 1

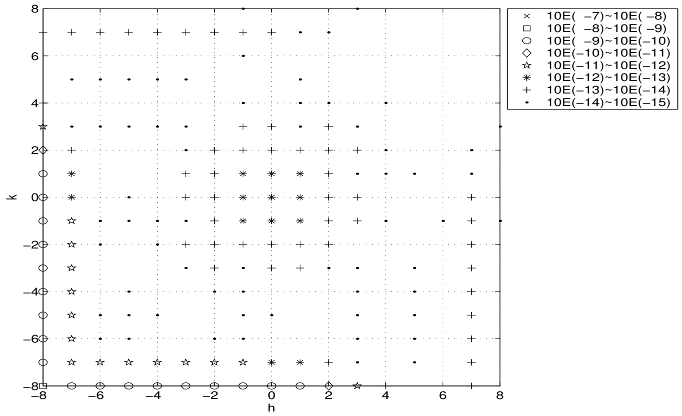

Figure A.17: The error distribution for D-W algorithm with $\rho = 0.5$ in Procedure 1

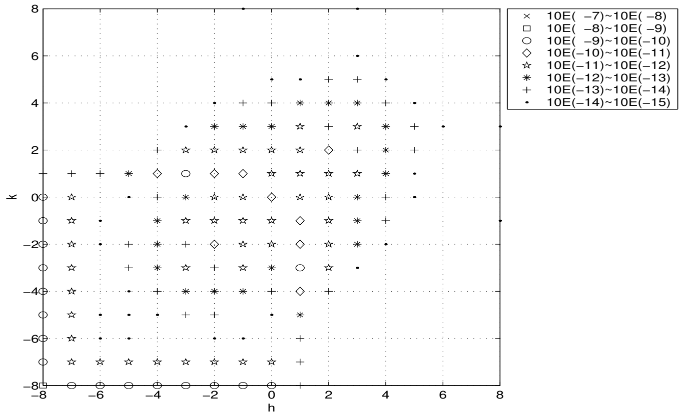

Figure A.18: The error distribution for S-D algorithm with $\rho = 0.5$ in Procedure 1

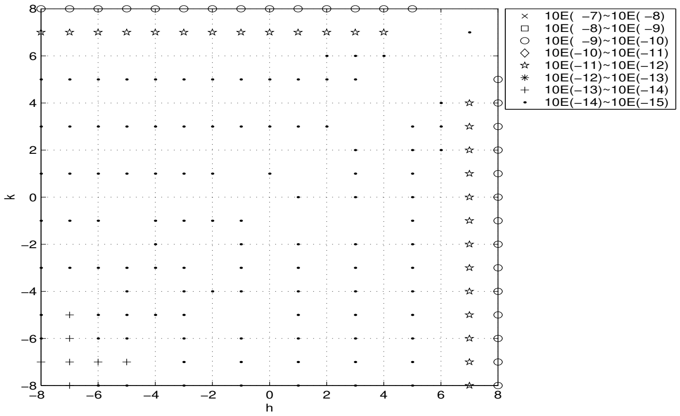

Figure A.19: The error distribution for D algorithm with $\rho = 0.7$ in Procedure 1

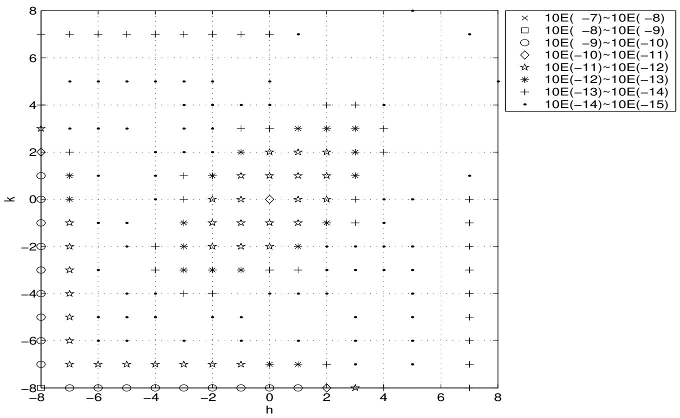

Figure A.20: The error distribution for D-W algorithm with $\rho = 0.7$ in Procedure 1

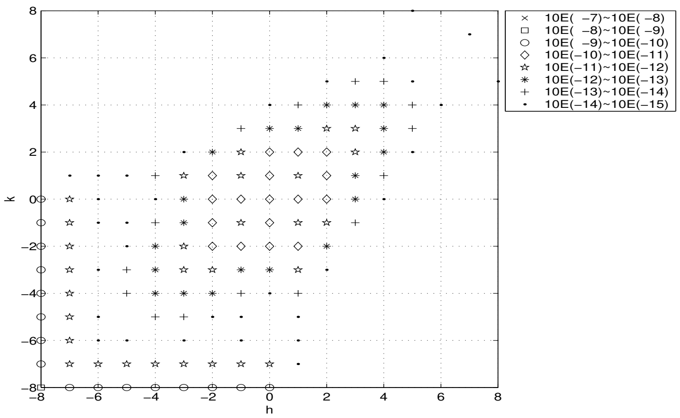

Figure A.21: The error distribution for S-D algorithm with $\rho = 0.7$ in Procedure 1

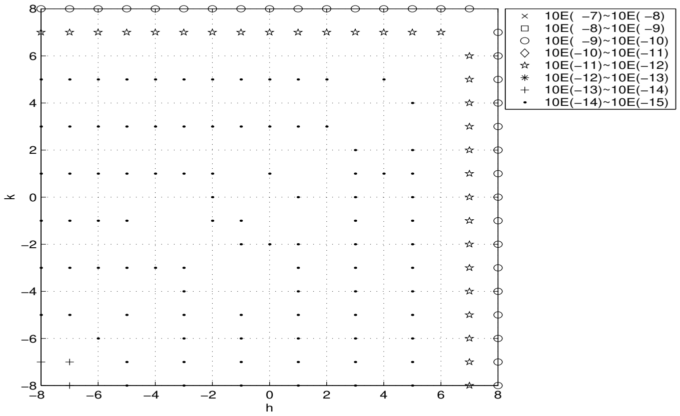

Figure A.22: The error distribution for D algorithm with $\rho = 0.9$ in Procedure 1

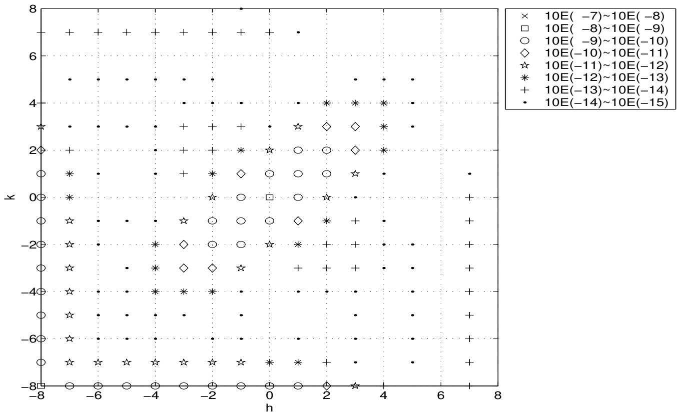

Figure A.23: The error distribution for D-W algorithm with $\rho = 0.9$ in Procedure 1

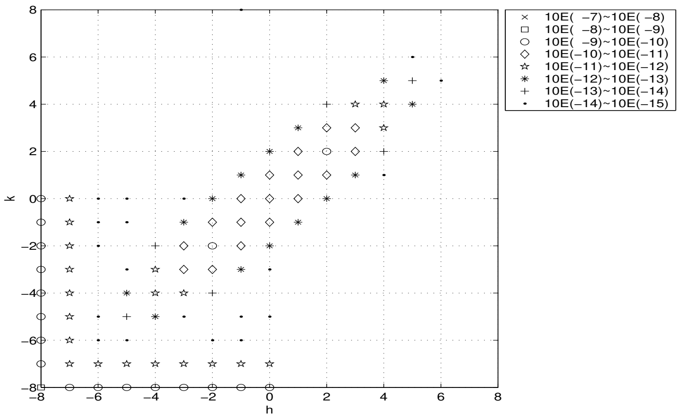

Figure A.24: The error distribution for S-D algorithm with $\rho = 0.9$ in Procedure 1

Appendix B

Tables of $L(h, k; \rho)$

This appendix provides the tables of the bivariate standard normal probability function $L(h, k; \rho)$. They are computed by the method from the Procedure 2 introduced in Chapter 3.

For a particular value of the correlation ρ, the value of $L(h, k; \rho)$ for each point (h, k), can be obtained by using any one of the four algorithms–Owen-Donnelly, Drezner, Drezner-Wesolowsky and Simon-Divsalar, where the correlation ρ ranges from -0.9 to 0.9 in increments of 0.2, and the integral limits h and k range from -8 to 8 in increments of 1.

Bibliography

[1] M. Abramowitz and I. A. Stegun, "Handbook of mathematical functions with formulas, graphs and mathematical tables," *U.S. Department of Commerce*, 1970.

[2] N.C. Beaulieu, "A simple series for personal computer computation of the error function $Q(\cdot)$," *IEEE Transactions on Communications*, Vol. 37, pp. 989-991, Sep. 1998.

[3] H. Bouver and R. E. Bargmann, "Comparison of computational algorithm for the evaluation of the univariate and bivariate normal distribution," *Proceedings of the Computer Science and Statistics: 12th Annual Symposium on the Interface*, pp. 344-348, 1979.

[4] D. Borth, "A modification of Owen's methods for computing the bi-variate normal integral," *Applied Statistics*, Vol. 22, pp. 82-85, 1973.

[5] C. V. L. Charlier, "Applications de la Théorie des Probabilités à l' Astronomie. Traité du Calcul des Probabilités et de ses Applications," *Paris: Gauthier-Villais et Cie.*, Vol. 2, Part 4, 1931.

[6] J. W. Craig, "A new simple and exact result for calculating the probability of error for two-dimensional signal constellations," *IEEE MILCOM'91 Conf. Rec., Boston, MA*, pp. 25.5.1-25.5.5, 1991.

[7] D. J. Daley, "Computation of bi- and tri-variate normal integrals," *Applied Statistics*, Vol. 23, pp. 435-438, 1974.

[8] D. R. Divgi, "Calculation of univariate and bivariate normal probability functions," *Annals of Statistics*, Vol. 7, pp. 903-910, 1979.

[9] T. G. Donnelly, "Algorithm 462: Bivariate normal distribution," *Communications of the Association for Computing Machinery*, Vol. 16, pp. 636, 1973.

[10] Z. Drezner, "Computation of the bivariate normal integral," *Mathematics of Computation*, Vol. 32, pp. 277-279, 1978.

[11] Z. Drezner and G. O. Wesolowsky, "The computation of the bivariate normal integral," *Journal of Statistical Computation and Simulation*, Vol. 35, pp. 101-107, 1990.

[12] S. S. Gupta, "Probability integral of multivariate normal and multivariate t," *Annals of Mathematical Statistics*, Vol. 34, pp. 792-828, 1963.

[13] N. L. Johnso and S. Kotz, "Distributions in statistics: Continuous multivariate distribution," *John Wiley and Sons, Inc.*, 1972.

[14] D. B. Owen, "Tables for computing bivariate normal probabilities," *Annals of Mathematical Statistics* , Vol. 27, pp. 1075-1090, 1956.

[15] R. S. Parrish and R. E. Bargmann, "A method for the evaluation of cumulative probability of bivariate distribution using the Pearson family," *Statistical Distributions in Scientific Work* , 5 , pp. 241-257, 1981.

[16] K. Pearson, "Tables for Statisticians and Biometricians," *Part 2. Cambridge: University Press*, 1931.

[17] R. L. Plackett, "A reduction for normal multivariate integrals," *Biometrika*, Vol. 41, pp. 351-360, 1954.

[18] J. W. F. Sheppard, "On the calculation of the double integral expressing normal correlation," *Transactions of the Cambridge Philosophical Society*, 19, pp. 23–69, 1900.

[19] M. K. Simon and D. Divsalar, "Some new twists to problems involving the Gaussian probability integral," *IEEE Transactions on Communications*, Vol. 46, pp. 200-210, Feb. 1998.

[20] R. R. Sowden and J.R. Ashford, "Computation of the bi-variate normal integral," *Applied Statistics*, Vol. 18, pp. 160-180, 1969.

[21] N. M. Steen, G. O. Byrne and E. M. Gelband, "Gaussian quadratures," *Mathematics of Computation*, Vol. 23, pp. 661-671, 1969.

[22] J. Terza and U. Welland, "A comparison of bivariate normal algorithms," *Journal of Statistical Computation and Simulation*, Vol. 39, pp. 115-127, 1991.

[23] O. A. Vasicek, "A series expansion for the bivariate normal integral," *The Journal of Computational Finance*, Vol. 1, pp. 5-10, 1998.

[24] J. C. Young and C. E. Minder, "Algorithm AS 76: An integral useful in calculating non-central t and bi-variate normal probability," *Applied Statistics*, Vol. 23, pp. 455-457, 1974.

[25] National Bureau of Standards, "Tables of the Bi-variate Normal Distribution Function and Related Functions," *Washington: U.S. Government Printing Office.*, 1959.