Finite Differences Based on Radial Basis Functions to Price Options

Xi Wang

Examensarbete i matematik, 30 hp
Handledare: Lina von Sydow
Examinator: Anders Öberg
Juni 2014
Abstract
In order to determine prices of pricing financial derivatives such as options, numerical methods must be developed continuously. The method must be very efficient since the option prices are computed in real time as the market condition changes. The assumption of Brownian motion of stock prices leads to a parabolic partial differential equation (PDE). In this project, we have developed finite differences based on radial bases functions, a combination of both radial basis function approximations and finite differences, to price options.
Contents

1 Introduction ............................................................................................................................... 5

2 Background .............................................................................................................................. 7
   2.1 Black Scholes Equation ................................................................................................. 7
   2.2 Finite Difference Method .............................................................................................. 7
   2.3 Radial Basis Function .................................................................................................... 9

3 Radial Basis Functions Based on Finite Differences Techniques (RBF-FD) ................................................. 11

4 Implementation ....................................................................................................................... 13

5 Comparison of Different Radial Basis Functions for RBF-FD ............................................ 15

6 Improvement on Accuracy ...................................................................................................... 18

7 Final Results and Conclusion .............................................................................................. 21

References .................................................................................................................................... 23
1. Introduction

The financial markets are becoming more and more complex, with trading not only of stocks, but also of numerous types of financial derivatives. Options are one kind of derivative financial instrument, which financial investors use for the arbitrage and hedging transactions [13]. And the most important and difficult part of option problems is option valuation, which requires a continuous effort in modeling and development of numerical solution methods in order to determine fair prices. As an example, we can consider the widely used Black-Scholes model. The underlying assumption is Brownian motion of stock prices and this leads to a parabolic partial differential equation (PDE). Since option prices must be computed in real time, over and over again depending on the changing market conditions we need efficient computational methods.

Numerical methods have greatly influenced the development of option valuation. Advances in numerical methods offer the opportunity to examine issues in options with greater details and precision. These methods can explore extremely complicated options in far greater detail than previously. A number of practical numerical methods are used in today’s financial derivative market, e.g. Monte Carlo methods, finite-difference methods, and methods based on approximation with radial basis functions.

Monte Carlo simulations are a broad class of computational algorithms that rely on repeated random sampling. This approach has been proved to be an easy and flexible numerical method in security pricing, even in high dimensional problems [11]. However, the convergence of this method is slow. To achieve reliable results many thousands of simulations are needed, leading a lot computation and time.

Finite-difference methods (FD) are numerical methods to solve partial differential equations (PDE) using finite differences to approximate derivatives. Compared to the Monte Carlo methods, finite difference methods converge very fast but require a structured grid to store local field quantities. Therefore, when increasing the number of underlying asset of option, this make solving option price become a high-dimension problem as spatial dimensions in the PDEs correspond to the prices of the assets let FD methods difficult to use. In [15] and [16], adaptive methods are used to mitigate the curse of dimensionality.
The last method presented here is approximation with Radial Basis Functions (RBF). A radial basis function is a real-valued function whose value depends only on the distance from the origin. This method became popular as a truly mesh-free method for the solution of partial differential equations (PDEs) on irregular domains and was proposed by Kansa [7] that compute a global solution. This method simplifies the approximations in high dimensions, give spectral convergence but result in full, ill-conditioned linear system of equations to solve each time-step. Later, a local version of the method was proposed by several authors [3, 20, 21]. It is to sacrifice the spectral accuracy inherent to the global version, in order to have a sparse better-conditioned linear system capable of solving large multidimensional PDEs. However, in this paper, accuracy is the one that I considered most, so I choose global version to compare and the best accuracy for a given number of node points is typically achieved when the basis functions are scaled to be nearly flat, which means that smaller shape parameter is needed.

In this paper, RBF-FD method, which is a combination of the RBF method and the FD method in order to get higher accuracy and reduce the amount of calculations. In 2007, Chandhini G, Sanyasiraju Y [5] demonstrate that by varying the shape parameter in RBF can get higher accuracy. In comparison with the FD scheme, the solutions can be found without too much calculations [4]. Although the RBF-FD method is fairly new, the high accuracy and few calculations attract plenty of people to research it. Victor Bayona, Miguel Moscoso, Manuel Carretero and Manuel Kindelan declared in [1] that as the RBF-FD is becoming increasingly popular as an alternative to the global version, people are more and more relying on it and RBF-FD method is consistent with the properties of RBF method that can achieve better accuracy when the shape parameter is small. The RBF-FD method can work well for many problems in vary area, in 2012 [8], RBF-FD was impled by Natash Flyer and Erik Lehto et.al for nonlinear systems of purely convective PDEs that model complex dynamics on the spheres in large-scale geoscience modeling. It was demonstrated that RBF-FD, as good as DG and global RBFs, gives an order of higher accuracy while also show relative fast computation rates. In this paper, the aim is to develop and investigate RBF-FD methods for Black-Scholes equation to price European options.
2. Background

2.1 Black Scholes Equation

We are interested in solving the Black-Scholes equation, a PDE describing the option price over time, to solve option valuation problems. Consider a stock option price described as \( P_t = f(t, S_t) \), where \( f \) is a second-order differential operator in stock price \( S \) and time \( t \), and \( K \) is the strike price. Then we have a transformed version with following properties

\[
Kx = s, \quad r = \frac{\bar{r}}{\hat{\sigma}^2}, \quad KP(t, x) = F(\hat{t}, s), \quad (2.1)
\]

\[
\hat{\sigma} = \frac{\hat{\sigma}}{\hat{\sigma}}, \quad \hat{t} = \hat{\sigma}^2 (\hat{T} - \hat{t}), \quad K\psi = \Phi(s),
\]

where \( K \) is the strike price, more details can be found in [15] and the transformed PDE reads

\[
\begin{cases}
\frac{\partial P}{\partial t} = \mathcal{L}P, \\
P(T, x_T) = \Phi(x),
\end{cases}
\]

\[
\mathcal{L}P = rx \frac{\partial P}{\partial x} + \frac{1}{2} \sigma^2 x^2 \frac{\partial^2 P}{\partial x^2} - rP, \quad (2.2)
\]

\[
r \text{ is the risk-free interest rate, } \sigma \text{ is the volatility. In this study, European call options are considered as an example of contract function}
\]

\[
\psi(x) = \max(x - 1, 0). \quad (2.4)
\]

2.2 Finite Difference Method

The FD method is one of the most useful numerical methods for approximating the solution of partial differential equations. Finite differences approximate occurring derivatives, introducing a grid (e.g. in space and time).

To illustrate the finite difference method, first we define the domain discretization in \( x \) and \( t \):

\[
0 \leq x_j \leq x_{\max}, \quad \Delta x = \frac{x_{\max}}{M}, \quad x_j = j \cdot \Delta x, \quad (2.5)
\]

\[
0 \leq t_n \leq T, \quad k = \frac{T}{N}, \quad t_n = i \cdot k, \quad (2.6)
\]
where $x_{\text{max}} = 4$.

Then we have the initial condition boundary conditions of European call option as follows

$$P(0, x) = \max(x - 1, 0),$$

$$P(t, 0) = 0,$$

$$P(t, x_{\text{max}}) = (x_{\text{max}} - 1)e^{-rt}.$$  

(2.7)

(2.8)

The spatial derivatives are approximated by

$$\frac{\partial P}{\partial x} = \frac{P_{j+1} - P_{j-1}}{2\Delta x} + \mathcal{O}(\Delta x^2),$$

$$\frac{\partial^2 P}{\partial x^2} = \frac{P_{j+1} - 2P_j + P_{j-1}}{\Delta x^2} + \mathcal{O}((\Delta x)^2),$$

(2.9)

(2.10)

where $P_j = P(t, S_j(t))$, $j = 1, 2, ..., N - 1$, see [19]

For the time evolution of the problem, we can use the unconditionally stable BDF2 method [12]. Denote $P^n_j \approx P_j(t^n)$, which yields

$$P^n_j + \beta_1 P^{n-1}_j + \beta_2 P^{n-2}_j = k\beta_0 \mathcal{L} P^n_j,$$

(2.11)

where $\beta_0 = 1, \beta_1 = -1$ and $\beta_2 = 0$ for the first time-step and $\beta_0 = \frac{2}{3}, \beta_1 = -\frac{4}{3}$ and $\beta_2 = \frac{1}{3}$ for subsequent steps. For more information about this method to price option, see [17]

![Figure 2.1](image1)

**Figure 2.1.** Pricing European call option price using finite differences method with 30 as strike price

Figure 2.1 shows the error as a function of stock price. The second plot shows the estimate of option price as a function of stock price. We can see that we get a large error located around strike price 30. All plots in this thesis are for the original Black-Scholes equation, not the transformed one.
2.3 Radial Basis Function

A radial basis function is a real-valued function of the form $\Phi(\|x-x_i\|)$ whose value only depends only on the distance from the origin ($x_i = 0$) or alternatively on the distance from some other point $c$, called a center ($x_i = c$).

The reasons for using RBFs for option pricing are that it is easy to use in higher dimensions, it allows adaptive node placement, and can gives spectral accuracy.

Radial Basis Functions are often used in function approximations, [2] , on the form:

$$y(x) = \sum_{j=1}^{N} w_j \phi(\|x-x_j\|) \tag{2.12}$$

Here, $y(x)$ is the sum of $N$ radial basis functions, each function associated with a different center $x_i$, and weighted by an appropriate coefficient $w_i$.

Commonly used are four types of the radical basis functions below (let $r = \|x-x_i\|$) [18]:

<table>
<thead>
<tr>
<th>Radial Basis Function</th>
<th>$\phi(r)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiquadric (mq)</td>
<td>$\sqrt{1+(\varepsilon r)^2}$</td>
</tr>
<tr>
<td>Inverse quadric (iq)</td>
<td>$\frac{1}{1+(\varepsilon r)^2}$</td>
</tr>
<tr>
<td>Gaussian (gs)</td>
<td>$e^{-(\varepsilon r)^2}$</td>
</tr>
<tr>
<td>Polyharmonic spline (r3)</td>
<td>$r^{3k}, k = 3$</td>
</tr>
</tbody>
</table>

where $\varepsilon$ is a shape parameter.

Approximating the solution with a time-dependent linear combination of RBF’s centered at the node points $x_j$, we get

$$P(t, x) = \sum_{j=1}^{N} \lambda_j(t) \phi(\varepsilon \|x-x_j\|) = \sum_{j=1}^{N} \lambda_j(t) \phi_j(x), \tag{2.13}$$

where $\phi(r)$ is the radial basis function and $\lambda_j(t)$ represent the coefficients to be determined. Again, we can use BDF2 for time stepping.
Figure 2.2 shows that the result of pricing European call option using an RBF method with $mq$ as radial basis function and $5$ as shape parameter. The first plot shows the error as a function of stock price. The second plot shows the estimate of option price as a function of stock price. We can see that the error in this case is spread everywhere.
3. Radial Basis Functions Based on Finite Differences Techniques (RBF-FD)

The radial basis functions based finite difference method (RBF-FD) extracts the best features from both radial basis function approximations and finite differences. Rather than having all basis functions coupled to each other leading to the full linear system of equation, a nearest neighbor region is connected to each basis function. This leads to a sparse matrix, and we found that good convergence properties are retained.

Given \( N \) total nodes in the domain, here we have a set of initial stock price \( \{x_j\}_{j=1}^N \). First we should find the inner weights for each \( \{x_j\}_{j=1}^N \). We assume \( n \) neighboring points for each \( x_j \), this means that for each \( x_j \) there are \( 2m \) points that should be used and \( n = 2m + 1 \), where \( m \) is introduced for computation in programming. Then we have the equation

\[
A_{(n \times n)} w_j = B_{(n \times 1)} 
\]

where

\[
A = a_{ij}, a_{ij} = \phi(\epsilon \|x_i - x_j\|), i, j = \{1, 2, \ldots, n\},
\]

\[
B = b_j, b_j = \mathcal{L} \phi(\epsilon \|x - x_i\|)_{x=x_j}, i = \{1, 2, \ldots, n\}, j = \{2, 3, \ldots, N - 1\},
\]

and the vector \( w_j \) contains the weights. In matrix form this is

\[
\begin{pmatrix}
\phi(\epsilon \|x_1 - x_1\|) & \phi(\epsilon \|x_1 - x_2\|) & \cdots & \phi(\epsilon \|x_1 - x_n\|) \\
\phi(\epsilon \|x_2 - x_1\|) & \phi(\epsilon \|x_2 - x_2\|) & \cdots & \phi(\epsilon \|x_2 - x_n\|) \\
\vdots & \vdots & \ddots & \vdots \\
\phi(\epsilon \|x_n - x_1\|) & \phi(\epsilon \|x_n - x_2\|) & \cdots & \phi(\epsilon \|x_n - x_n\|)
\end{pmatrix} \begin{pmatrix}
\begin{pmatrix}
w_{j1} \\
w_{j2} \\
\vdots \\
w_{jn}
\end{pmatrix}
\end{pmatrix} = \begin{pmatrix}
\mathcal{L} \phi(\epsilon \|x - x_1\|)_{x=x_j} \\
\mathcal{L} \phi(\epsilon \|x - x_2\|)_{x=x_j} \\
\vdots \\
\mathcal{L} \phi(\epsilon \|x - x_n\|)_{x=x_j}
\end{pmatrix} \begin{pmatrix}
\begin{pmatrix}
w_{j1} \\
w_{j2} \\
\vdots \\
w_{jn}
\end{pmatrix}
\end{pmatrix}
\]

Solving 3.2 we get \( n \) weights for \( x = x_j \). Do the same procedure for each stencil center \( x_j, j = \{2, 3, \ldots, N - 1\} \) and plug them into a matrix of size \( (N \times N) \) which has \( n \) non-zero values on each row and for the first and last row they only have a value 1 for the boundary conditions. Then we have
Then we use the BDF2 scheme as described above. After combination of similar terms and simplification we get

\[
(I - k \beta_0 W) P^n(x) = -\beta_1 P^{n-1}(x) - \beta_2 P^{n-2}(x),
\]

Inserting boundary conditions gives

\[
(I - k \beta_0 D) P^n(x') - k \beta_0 g(t) = -\beta_1 P^{n-1}(x') - \beta_2 P^{n-2}(x'),
\]

where \( g = [w_{2f}(t,x_1)w_{3f}(t,x_1)\ldots 0\ldots 0\ldots w_{N-2f}(t,x_N)w_{N-1f}(t,x_N)]^T \) of size \((N-2) \times 1, x' = [x_2x_3\ldots x_{N-2}]^T\), and

\[
D = \begin{pmatrix}
1 & 0 & 0 & \ldots & 0 & 0 & 0 & \ldots & 0 & 0 \\
w_2^2 & w_3^2 & w_n^2 & \ldots & w_n^2 & 0 & 0 & \ldots & 0 & 0 \\
w_3^2 & w_3^3 & w_n^3 & \ldots & w_n^3 & 0 & 0 & \ldots & 0 & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
w_{m+1}^2 & w_{m+1}^3 & \ldots & w_{m+1}^n & 0 & 0 & \ldots & 0 & 0 \\
0 & w_{m+2}^1 & w_{m+2}^2 & \ldots & w_{m+2}^n & 0 & \ldots & 0 & 0 \\
0 & 0 & w_{m+3}^1 & w_{m+3}^2 & \ldots & w_{m+3}^n & 0 & \ldots & 0 \\
0 & 0 & 0 & w_{m+4}^1 & w_{m+4}^2 & \ldots & w_{m+4}^n & 0 & \ldots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & 0 & \ldots & 0 & w_{N-m-1}^1 & w_{N-m-1}^2 & w_{N-m-1}^3 & \ldots & w_{N-m}^n \\
0 & 0 & 0 & \ldots & 0 & w_{N-1}^1 & w_{N-1}^2 & w_{N-1}^3 & \ldots & w_{N-1}^n \\
0 & 0 & 0 & \ldots & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix} \quad (N-2) \times (N-2)
\]
4. Implementation

During the implementation of the algorithm, it was obvious that many things were computed several times. By using a clever algorithm, an amount of time and calculation can be saved.

Naively, $N$ square matrices with $n \times n$ are used in formula 3.2 in order to calculate the weight matrix. For example, for the case of $n = 7$ and $N = 20$, 14 distance matrices are shown in Table 4.1. For simplicity, the elements of all the distance matrices are represented as either 0 or 1, where 1 means the distance is nonzero and 0 means the distance is exactly 0. Then we have as follows:

Table 4.1. Old form of distance matrix

|   | 1 | 1 | 1 | 1 | 1 | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* |
|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0 | 1 | 1 | 1 | 1 | 1 | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* |
| 1 | 0 | 1 | 1 | 1 | 1 | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* |
| 1 | 1 | 1 | 0 | 1 | 1 | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* |
| 1 | 1 | 1 | 1 | 0 | 1 | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* |
| 1 | 1 | 1 | 1 | 0 | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* |
| 1* | 1* | 1* | 1* | 1* | 1* | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1
| 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1
| 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1
| 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1
| 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1

Apparently, there are shared computations in two consecutive matrix calculations and also 0 values can be skipped for efficient calculations. Therefore to optimize the computation, Table 4.1 is then transformed into a $20 \times 13$ matrix as shown in Table 4.2, which is the input argument of the RBF.
Table 4.2. *New form of distance matrix*

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

The italic area shows the same values in both tables and ones with star are the values that need to be calculated at each loop.

This means that only need a matrix with size $N \times (2n - 1)$ and compute $N \times (2n - 1)$ distances rather than $N \times N$. Then use distances in Table 4.2 to get $\phi_j(x)$ and this leads to a lot of time reduction. As we still need the matrix in the form of Table 4.1, after all computations, we transform back to the old form.

In this study, all the solutions are implemented in Matlab on Mac OS X 10.7.4 and linear system of equations are solved with ‘\’.

Because we are only interested in option prices near the strike price. We compute norm of the error in the interval $[K, \frac{5K}{3}]$ for comparisons.

The Euclidean norm error is calculated as follows:

$$error_j = estimate\ value_j - exact\ value_j, \ j = 1, 2, ..., N,$$

$$\|error\| = \sqrt{error_1^2 + error_2^2 + ... + error_N^2}.$$
5. Comparison of Different Radial Basis Functions for RBF-FD

In this thesis the goal is to find the optimal parameters \((n, \epsilon, \phi)\) to get high accuracy with low computing time. Figure 5.1 and 5.2 to start with, I try out a reasonable range for \(\epsilon\) and \(n\). Figure 5.1 demonstrates that choosing \(n = 7\) is enough for all \(N\) as there is no much better accuracy when increasing \(n\) larger than 7 and larger stencil size cost more computation time and also needs larger shape parameter as the coefficient matrices in 3.1 ill-conditioning. So I will only use \(n = 3, n = 5\) and \(n = 7\) when compare with FD. For the shape parameter \(\epsilon\), it is known that the best accuracy for a given number of node points is typically achieved when the basis functions are scaled to be nearly flat [14]. I take \(mq\) as radial basis function and \(M = 100, N = 60,\) and \(n = 5\) as an example in 5.2. It shows that the shape parameter \(\epsilon\) has a big effect on the error. By choosing 2 as shape parameter we can avoid singular weight matrix in 3.1 when \(N\) is large. This result also holds for other radial basis function except \(r3\).

![Error function](image)

**Figure 5.1.** The error as a function of \(n\) for the solution using RBF-FD with \(mq\) as radial basis function using 2 as shape parameter (ep) and \(M = 100\) as an example.
Figure 5.2. The error as a function of shape parameter $\varepsilon$ for the solution using FD-RBF with mq as radial basis function using $M = 100$ as an example.

The following figure 5.3 shows four graphs to compare radial basis functions described previously, plotting error against $N$, for different stencil sizes.

Figure 5.3. Error as function of N for the solution of RBF-FD using $M = 100$, $\varepsilon = 2$ as an example.

When increasing the number of nodes in space, we get a smaller error and larger stencil sizes give smaller error.

When using $r3$ as radial basis function, we cannot get reasonable results for $n = 3$, so we only have results with $n = 5$ and 7. We can see that the errors are
too big compare to the other three radial basis functions.

After comparing all four radial basis functions and stencil sizes $n$, we can conclude that only three of four radial basis function, $mq$, $iq$ and $gs$ generate reasonable solutions for pricing of European option. The following experiments are all based on this conclusion. Furthermore, we conclude that larger stencil sizes gives smaller errors but for $n > 7$ we can run into ill-conditioned systems. The shape parameter is for this problem ideally chosen to be 2.
6. Improvement on Accuracy

A possible improvement is adding a constant by [9]

\[
y(x) = \sum_{i=1}^{N} w_i \phi(\|x-x_i\|) + w_{i+1},
\]

\[
\sum_{i=1}^{N} w_i = 0,
\]

When apply this alternate procedure to RBF-FD, the inner matrices 3.2 becomes 6.1.

\[
\begin{pmatrix}
\phi(\varepsilon \|x-x_1\|) & \phi(\varepsilon \|x_1-x_2\|) & \cdots & \phi(\varepsilon \|x_1-x_n\|) \\
\phi(\varepsilon \|x_2-x_1\|) & \phi(\varepsilon \|x_2-x_2\|) & \cdots & \phi(\varepsilon \|x_2-x_n\|) \\
\vdots & \vdots & \ddots & \vdots \\
\phi(\varepsilon \|x_n-x_1\|) & \phi(\varepsilon \|x_n-x_2\|) & \cdots & \phi(\varepsilon \|x_n-x_n\|) \\
1 & 1 & \cdots & 1
\end{pmatrix}
\begin{pmatrix}
w_1^j \\
w_2^j \\
\vdots \\
w_n^j \\
w_{n+1}^j
\end{pmatrix}
= 
\begin{pmatrix}
\mathcal{L} \phi(\varepsilon \|x-x_1\|)_{x=x_j} \\
\mathcal{L} \phi(\varepsilon \|x-x_2\|)_{x=x_j} \\
\vdots \\
\mathcal{L} \phi(\varepsilon \|x-x_n\|)_{x=x_j} \\
-r
\end{pmatrix}
\]

Solving 6.1 we get \(n+1\) weights. We only take the first \(n\) weights into use and do the same procedure as above. Then we get the following plots.

![Figure 6.1. Error as functions of N for adding constant for the solution of RBF-FD using M=100 as an example.](image)
Figure 6.2. Error as functions of N to compare new improvements and choosing mq as RBF

Compared with the one without adding constant we can see that adding constants generates only a slightly better result.

Since the method is concerned about financial concepts, we can adapt node point distribution to reduce financial error norms. We take $p = (N - 2)/3$, where $p$ is an integer and distribute $2p + 1$ points uniformly in the interval $[0, 2K]$ and the remaining $p$ points in the rest of the computational domain [17].

Figure 6.3. non-uniform node distributions in one dimension

Using the same values of parameters we get the following plots.
Figure 6.4. Error of functions of N using non-equidistant node for the solution of RBF-FD using M=100 as an example

Figure 6.5. Error as functions of N to compare new improvements choosing mq as RBF

Figure 6.4 and 6.5 show that non-uniform node distribution make slightly better results for $n = 3$ but there is no big difference for $n = 5$ and 7. Therefore, standard RBF-FD and equidistant node distribution can be chosen as the final version of the method.
7. Final Results and Conclusion

After a series of tests to improve the technique, we compare RBF-FD with optimal values of parameters decided above with RBF and FD.

Figure 7.1. The error as functions for the solution using RBF-FD, RBF and FD with \( mq \) as radial basis function for RBF-FD and \( M=100 \)

Figure 7.1 demonstrates that RBF-FD has smaller errors than FD and larger than RBF for same \( N \). For \( n = 3 \), RBF-FD behaves like FD which is reasonable. In this paper, we have implemented the algorithm of RBF-FD and made a series of experiments to get best accuracy of the method, we compared four types of radial basis functions, various shape parameters and stencil size \( n \). We also tried to add a constant in our approximations and also to use non-equidistant grids. The main conclusion of the work is following:

- Three of the four radial basis functions examined are applicable, but multiquadric (\( mq \)) radial basis function is stable for most cases.
- \( \varepsilon = 2 \) turns out to be the optimal value with respect to accuracy and stability.
• Non-equidistant grids show a slightly better accuracy than equidistant ones. Adding a constant to the approximation does not improve the accuracy.
• Compared to finite differences, the RBF-FD method shows a better accuracy larger than 3.
• For high-dimensional problems we expect this method to be highly efficient since we can have node-points placed arbitrarily where they are needed for accuracy reasons. For FD we are restricted to the usage of Cartesian grids.
References


