UPPSALA
UNIVERSITET

IT 12 051
Examensarbete 30 hp
Oktober 2012

Evaluation of a least-squares radial basis function approximation method for solving the Black-Scholes equation for option pricing

Cong Wang

Institutionen för informationsteknologi
Department of Information Technology

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

Evaluation of a least-squares radial basis function approximation method for solving the Black-Scholes equation for option pricing

Cong Wang

Radial basis function (RBF) approximation, is a new extremely powerful tool that is promising for high-dimensional problems, such as those arising from pricing of basket options using the Black-Scholes partial differential equation. The main problem for RBF methods have been ill-conditioning as the RBF shape parameter becomes small, corresponding to flat RBFs. This thesis employs a recently developed method called the RBF-QR method to reduce computational cost by improving the conditioning, thereby allowing for the use of a wider range of shape parameter values. Numerical experiments for the one-dimensional case are presented and a MATLAB implementation is provided. In our thesis, the RBF-QR method performs better than the RBF-Direct method for small shape parameters. Using Chebyshev points, instead of a standard uniform distribution, can increase the accuracy through clustering of the nodes towards the boundary. The least squares formulation for RBF methods is preferable to the collocation approach because it can result in smaller errors for the same number of basis functions.

Keywords: RBF, radial basis function, ill-conditioning, shape parameter, Black-Scholes equation, option pricing

Handledare: Elisabeth Larsson
Ämnesgranskare: Lina von Sydow
Examinator: Jarmo Rantakokko
IT 12 051
Tryckt av: Reprocentralen ITC

Contents

1 Introduction

As the financial markets are becoming more and more complex, people nowadays trade not only stocks, but also numerous types of financial derivatives. The market requires updated information about the values of these derivatives continuously. Thus, the market is in great need of more accurate and faster computer simulations.

In this thesis we consider the problem of pricing financial contracts on several underlying assets. As the demand of complex derivatives from the customers and the speed of computers have increased over the years, these contracts have become more and more popular. Here we have chosen to use a European basket call option as an example which is simple but working well as an indicator of the usefulness of our method.

One potentially effective way to price financial contracts is to solve the Black-Scholes equation [2], a partial differential equation (PDE) in which the number of spatial dimensions is determined by the number of underlying assets. When the number of dimensions grows, solving this PDE becomes computationally very demanding. That is why we really need to use fast and memory efficient algorithms.

There are different modern numerical techniques for computational option pricing in mathematical finance. Monte Carlo methods have the advantage of scaling linearly with the number of dimensions, but have the drawback of converging very slowly [14]. Several other recent deterministic approaches have also been considered, such as sparse tensor [29] or sparse grid [6] approximations. Finite difference methods are generally well known. These have better convergence properties but also suffer from the curse of dimensionality [32]. Adaptive finite difference methods have also been successfully used for pricing European options in [25], [28].

Here we consider Radial basis function (RBF) approximation [5], [9], [33] as a potentially effective approach for solving the multi-dimensional Black-Scholes equation. Option pricing using RBFs has been explored in one dimension for European and American options by Hon et al. [17], [34] and in both one and two dimensions by Fasshauer et al. [8] and Marcozzi et al. [27] with promising results. Hon has also applied a quasi-radial basis function method to option pricing in one dimension [18]. Larsson et al. used RBFs for pricing options in one and two dimensions [30] and in two and three dimensions [23] for European options. Strategies involving least squares approximation and a multi-level approach for radial basis functions for European options are also discussed by Larsson and Gomes [21]. Recently, Belova et al. described the penalty method for pricing American options [1] and Golbabai et al. successfully used the RBF method for the American put

option under jump diffusion [15]. Credit default swap (CDS) contracts were priced using the meshfree RBF interpolation by Guarin et al. [16] and spline approximations were used to solve a Black-Scholes partial differential equation modelling a European option pricing problem by Khabir et al. [19]. A multilevel kernel-based interpolation method using anisotropic RBFs were presented by Georgoulis et al. [13].

A typical RBF approximation has the form

$$s(\underline{x}, \varepsilon) = \sum_{j=1}^{N} \lambda_j \phi(\varepsilon ||\underline{x} - \underline{x}_j||) \tag{1}$$

where $\phi(r)$ is the RBF and $\|\cdot\|$ denotes the Euclidian norm, $\underline{x}_j$, $j = 1, ..., N$ are centre points, and ε is a shape parameter. The coefficients λ_j are typically determined by collocation with input data values at the center points. In our thesis, we also consider another approximating scheme, the least square (LS) formulation. Using LS, which can have advantages over interpolation in RBF approximations, is presented in [5]. Numerous forms of useful radial functions are available, such as the Gaussian RBF (GA) and the Multiquadric RBF (MQ). A small value of the shape parameter ε leads to flatter RBFs. The shape parameter is an important method parameter, with a significant effect on the accuracy of the method. Since the method only needs pairwise distances between points, it is meshfree. Therefore, it is easy to use in higher dimensions and it also allows for problem adapted node placement.

RBF approximations are sensitive to the treatment of the boundary conditions. Possible solutions can be boundary clustering of nodes, or variable shape parameter, as revealed by the studies in [12]. In this thesis, we make sure to fulfil the boundary conditions exactly also when using the LS scheme.

The contribution of this thesis is a numerical study of the effects of the method parameters on the accuracy and performance of the method, providing some insights regarding the limitations and possibilities of RBF methods. We look at sample problems in one dimension and we also compare the results of the different RBF methods. Furthermore, we discuss how the optimal shape parameter changes with time for the 1D solution of the Black-Scholes problem and using multiquadric RBF in a collocation scheme to show that the best shape parameter may even depend nonlinearly on the number of center points N. The RBF-QR method [10] which has been presented to be able to compute stably even in the $\varepsilon \to 0$ basis function limit, is compared with the RBF-Direct method which is associated with numerical ill-conditioning. The RBF-QR method is proved to have better results for both interpolation and derivatives. We also make a series of numerical experiments to find that when ε is large, uniform node points perform better than Chebyshev node

points, otherwise, when ε is small, Chebyshev node points get better results.

The outline of the thesis is as follows. In Section 2, we briefly review the RBF methods and especially the ill-conditioning of the RBF-Direct approach. Section 3 describes the RBF-QR approach for the 1D case, including expressions for the first and second derivatives. Then, in Section 4, we present the sample problems and boundary conditions and derive the space approximation and time discretization of the Black-Scholes model problem. The LS formulation is also explained here. A series of numerical results for several 1D tests concerning efficiency of the simulations and the quality of the results in terms of the different parameters involved. Finally, Section 5 gives some conclusions.

2 RBF methods

A typical radial basis function usually has the form $\phi(r) = \phi(\varepsilon\|\underline{x} - \underline{x}_j\|)$, where ε means the shape parameter of the radial basis function. Some of the most popular used RBFs are showed in Table 1. In this thesis, the Gaussian RBF (GA) and the Multiquadric RBF (MQ) are used in the numerical experiments. When infinitely smooth RBFs are used, the approximations feature spectral convergence as the points get more dense, which has been proven in [4], [26].

Table 1: The definitions of some commonly used radial basis functions

Name	**Abbreviation**	**Definition**
Gaussian	GA	$\phi(r)=e^{-(\varepsilon r)^2}$
Multiquadric	MQ	$\phi(r)=\sqrt{1+(\varepsilon r)^2}$
Inverse multiquadric	IMQ	$\phi(r)=1/\sqrt{1+(\varepsilon r)^2}$
Inverse quadratic	IQ	$\phi(r)=1/(1+(\varepsilon r)^2)$
Polyharmonic spline	PS	$\phi(r)=r^k, k=1,3,5...; r^k\ln(r), k=2,4,6...$

Another key feature of the RBF method is that it is mesh-free, which means that it does not require a grid. It only depends on distances to center points $\underline{x}_j$ in the approximation. As pairwise distances are very easy to compute in any number of space dimensions, it also works well for high-dimensional problem.

A standard RBF interpolant is a linear combination of RBFs $\phi(r)$ centered at the scattered points $\underline{x}_j$, $j = 1, ..., N$ which has the following form

$$s(\underline{x}, \varepsilon) = \sum_{j=1}^{N} \lambda_j \phi(\varepsilon\|\underline{x} - \underline{x}_j\|) \equiv \sum_{j=1}^{N} \lambda_j \phi_j(\underline{x}), \tag{2}$$

the unknown coefficients λ_j are determined through the interpolation condition $s(\underline{x}_j, \varepsilon) = f(\underline{x}_j)$ and can be computed as the solution of the following linear system

$$A_\phi \underline{\lambda} = \underline{f}, \tag{3}$$

where the symmetric matrix A has elements $a_{ij} = \phi_j(\underline{x}_i) = \phi(\varepsilon \|\underline{x} - \underline{x}_j\|)$, and the definitions of column vectors are $\underline{\lambda} = (\lambda_1, ..., \lambda_N)^T$ and $\underline{f} = (f_1, ..., f_N)^T$, respectively.

Furthermore, the implementation of an RBF method is also straightforward. However, there are still some issues left such as stability for the time-dependent problems and computational efficiency.

With the advantages mentioned above of achieving spectral accuracy using infinitely smooth basis functions, the geometrical flexibility with arbitrary choice of node locations and the ease of implementation, radial basis function (RBF) approximation is rising as an important method for interpolation, approximation, and solution of partial differential equations (PDEs).

2.1 The ill-conditioning of the RBF-Direct method

With the RBF-Direct approach to find the interpolant, we simply compute the coefficients λ_j as the solution of the linear system (3) mentioned above. However, in practical cases, convergence is often negatively affected by ill-conditioning of the matrix A as the shape of the basis functions become flatter.

The accuracy of the solution is typically highest for small values of the shape parameter ε for smooth functions. However the coefficients λ_j become extremely large in magnitude when $\varepsilon \to 0$ and a huge amount of numerical cancellation will occur as the quantity $s(\underline{x}, \varepsilon)$ obtained in (2) through the combination of these large quantities. Therefore, in the flat basis function regime, (2) and (3) form two successive ill-conditioned numerical steps in obtaining a quantity $s(\underline{x}, \varepsilon)$ which we know in general depends in a well conditioned way on the nodes $\underline{x}_j$ and data f_j [3], [7], [11], [24]. Moving to larger shape parameter values (less flat values) will get conditioning better, but make accuracy worse.

2.2 The RBF-QR method

With the RBF-QR method [10], we compute exactly the same quantity $s(\underline{x}, \varepsilon)$ as in the RBF-Direct method, but instead the results remain stable for all the values of ε, even when $\varepsilon \to 0$. In this thesis, the RBF-QR method is mostly used to solve the Black-Scholes model in finance approximation in section 4.

As mentioned above, if the data f_j is sampled from an infinitely smooth function, highly accurate interpolation results are typically achieved for small values of ε. The main idea is to recognize that the RBFs themselves constitute an ill-conditioned basis in a good approximation space [10]. In order to make a change of basis, first we expand the RBFs in terms of the expansion functions T_k, $k = 1, ...$ Then we truncate the expansions at $k = Tru \geq N$ based on the size of the contributions and QR-factorize the coefficient matrix, please see more details in [10]. The new basis functions are then obtained as

$$\begin{pmatrix} \psi_1(\underline{x}) \\ \vdots \\ \psi_N(\underline{x}) \end{pmatrix} = D_1^{-1} R_1^{-1} Q^T \begin{pmatrix} \phi_1(\underline{x}) \\ \vdots \\ \phi_N(\underline{x}) \end{pmatrix} \approx \begin{pmatrix} I_N & D_1^{-1} R_1^{-1} R_2 D_2 \end{pmatrix} \begin{pmatrix} T_1(\underline{x}) \\ \vdots \\ T_{Tru}(\underline{x}) \end{pmatrix}, \tag{4}$$

where I_N is the unit matrix of size $(N \times N)$ and the correction matrix $\widetilde{R} = D_1^{-1} R_1^{-1} R_2 D_2$ is small when ε is small. R_1 and D_1 is upper triangular and both of them are $N \times N$. Using the expression (3) for the new basis functions $\psi_j(\underline{x})$, we can calculate the action of a linear differential operator L on the basis functions through

$$\begin{pmatrix} L\psi_1(\underline{x}) \\ \vdots \\ L\psi_N(\underline{x}) \end{pmatrix} = \begin{pmatrix} I_N & \tilde{R} \end{pmatrix} \begin{pmatrix} LT_1(\underline{x}) \\ \vdots \\ LT_{Tru}(\underline{x}) \end{pmatrix}, \tag{5}$$

We can use the transpose of relation (4) above at each evaluation point $\underline{x}_i$, $i = 1, ...N$ to compute the new interpolation matrix A_ψ with elements $a_{ij} = \psi_j(\underline{x}_i)$ in the following way

$$A_\psi = T \begin{pmatrix} I_N \\ \widetilde{R}^T \end{pmatrix}, \tag{6}$$

where the matrix T has elements $t_{ij} = T_j(\underline{x}_i)$,$i = 1, ...N, j = 1, ...Tru$.

2.3 Numerical comparison between the RBF-Direct method and the RBF-QR method

First we choose a number of smooth functions as test examples to clearly present the stability and accuracy of the respective methods. Even though the methods also work for less smooth functions; these have been excluded since for these it is rarely advantageous to use small ε, which is the shape parameter range we are addressing here. The amount of variation is gradually increased which means the functions are more and more difficult. All function

values lie within the range $[-1, 1]$. The functions used in the numerical experiments are as follows

$$\begin{aligned} f_2(x) &= \frac{165}{165 + (x - 0.2)^3} \\ f_4(x) &= \sin(x^2) - \sin(2x^2) \\ f_5(x) &= \sin(2\pi x) \\ f_6(x) &= \sin(2\pi x^2) - \sin(4\pi x^2) \end{aligned}$$

Then the evaluation points and the center points are defined and their formats are showed in Figure 1.

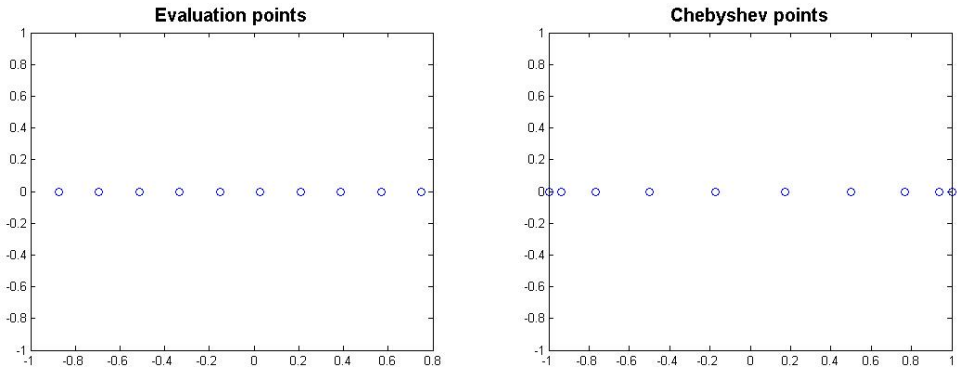

Figure 1: Ten uniform evaluation points and ten center points for each type within $[-1, 1]$ are shown in the figures. The center points used here are Chebyshev points.

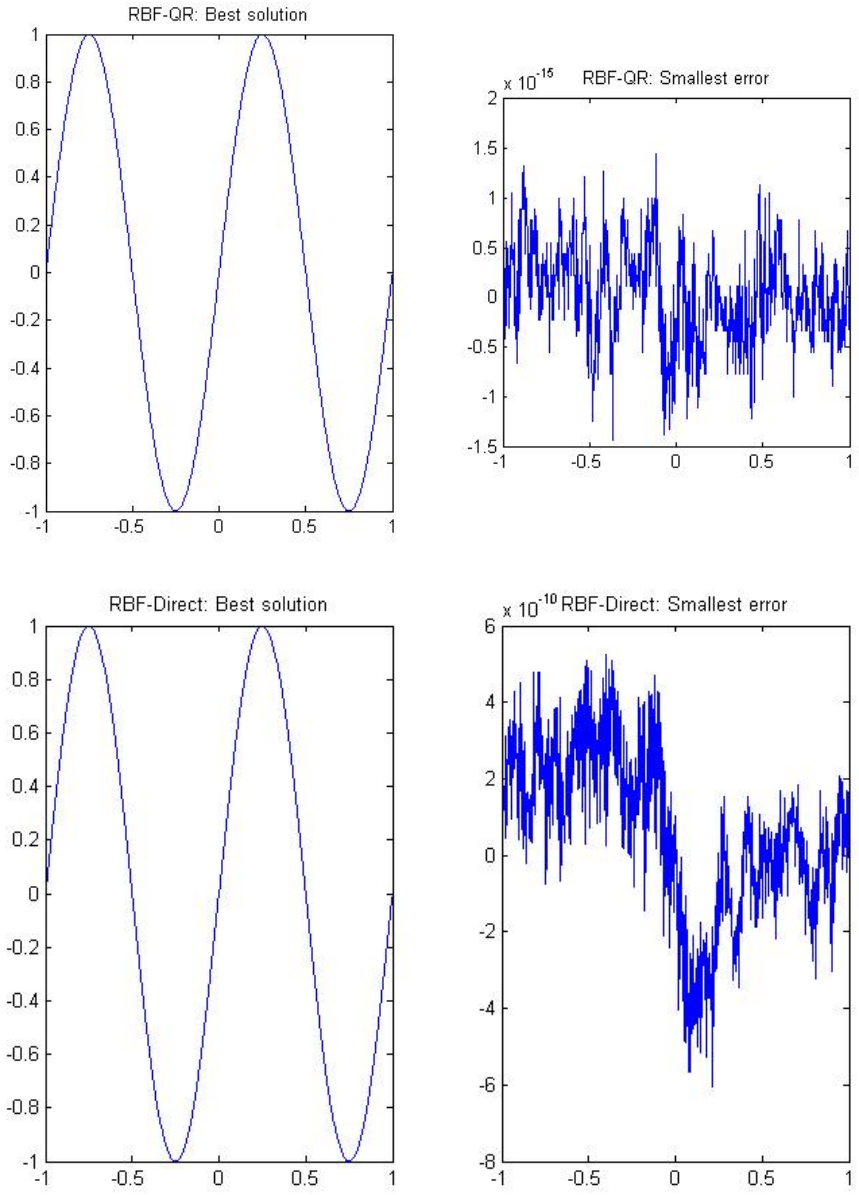

Figure 2: The best result with optimal ε (left subplot) and smallest errors compared with the exact solution (right subplot) in the RBF-QR method and the RBF-Direct method for $N = 100$ Chebyshev points.

In the above numerical comparison, $N = 100$ Chebyshev center points are chosen as an example and $N_e = 700$ evaluation points are used. An example function $f_5 = \sin(2\pi x)$ is interpolated over the unit interval using both the RBF-QR method and the RBF-Direct method and the best results

and smallest errors compared with the exact solution with each method are showed in Figure 2. After comparing with the errors in the two right subplots, we can easily find that the RBF-QR method performs much better than the RBF-Direct method to interpolate the example function.

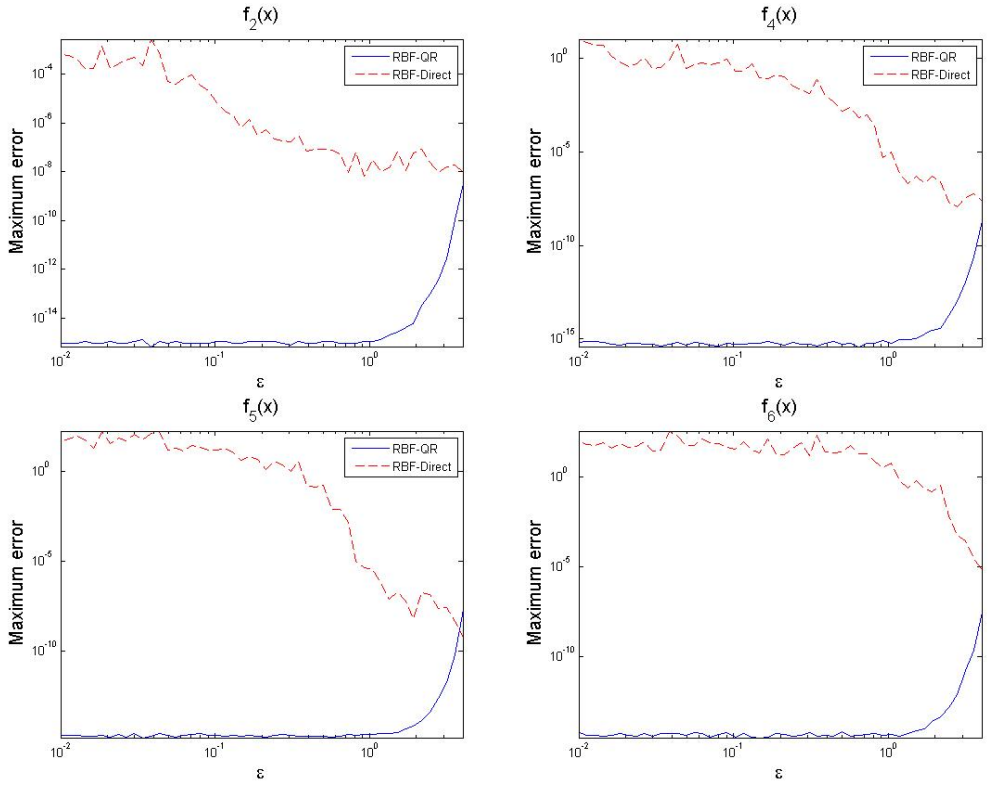

Figure 3: Interpolation max errors as a function of the shape parameter ε using RBF-QR (solid lines) and RBF-Direct (dashed lines) for functions f_2, f_4, f_5 and f_6. In all cases $N = 100$ $N_e = 700$ was used. All figures were shown using Chebyshev nodes.

Figure 3 shows results for a fixed N and a range of ε values. For small ε, the RBF-QR method produces more accurate results than the RBF-Direct method, whereas, for large ε, they give the same results.

3 The RBF-QR method in one dimension

3.1 Expansion of the GA radial function

The RBF-QR expansion in 1-D is stated in [10] without derivation. Here, we include the derivation of the interpolation case and also derive the expressions

for the first and second derivatives.

In the one-dimensional case, we expand the Gaussian RBFs only through exchanging powers for Chebyshev polynomials. From Table 1, the GA radial function has the form $\phi(r) = e^{-(\varepsilon r)^2}$. For an RBF centered at the point $\underline{x}_k$, we have

$$\phi(\underline{x}, \underline{x}_k) = e^{-\varepsilon^2 \|\underline{x}-\underline{x}_k\|} = e^{-\varepsilon^2 (\underline{x}-\underline{x}_k)\cdot(\underline{x}-\underline{x}_k)} = e^{-\varepsilon^2 (\underline{x}\cdot\underline{x})} e^{-\varepsilon^2 (\underline{x}_k\cdot\underline{x}_k)} e^{2\varepsilon^2 (\underline{x}\cdot\underline{x}_k)}. \quad (7)$$

Since only the last factor above mixes $\underline{x}$ and $\underline{x}_k$ values, we get the Taylor expansion of it

$$e^{2\varepsilon^2 (\underline{x}\cdot\underline{x}_k)} = 1 + 2\varepsilon^2 (\underline{x}\cdot\underline{x}_k) + \frac{(2\varepsilon^2)^2}{2!}(\underline{x}\cdot\underline{x}_k)^2 + \cdots = \sum_{j=0}^{\infty} \frac{(2\varepsilon^2)^j}{j!}(\underline{x}\cdot\underline{x}_k)^j. \quad (8)$$

We expand (8) using the following formula

$$x^j = \frac{1}{2^{j-1}} \sum_{l=0}^{\frac{j-p}{2}} \left(\begin{array}{c} j \\ l \end{array} \right) t_{j-2l} T_{j-2l}(x), \quad (9)$$

where $T_j(x)$ are the Chebyshev polynomials, $t_j = \frac{1}{2}$ if $j = 0$ and $t_j = 1$ otherwise. See more details in [10].

We can get

$$\begin{aligned} e^{2\varepsilon^2 x x_k} &= \sum_{j=0}^{\infty} \frac{(2\varepsilon^2 x x_k)^j}{j!} \\ &= 2\sum_{j=0}^{\infty} \frac{\varepsilon^{2j} x_k^j}{j!} \sum_{l=0}^{\frac{j-p}{2}} \left(\begin{array}{c} j \\ l \end{array} \right) t_{j-2l} T_{j-2l} \\ &= 2\sum_{j=0}^{\infty} \frac{\varepsilon^{2j} x_k^j}{j!} \sum_{l=0}^{\infty} \varepsilon^{4l} x_k^{2l} \frac{j!}{(j+2l)!} \frac{(j+2l)!}{l!(j+l)!} t_j T_j \\ &= 2\sum_{j=0}^{\infty} \frac{\varepsilon^{2j} x_k^j}{j!} {}_0F_1([], j+1, \varepsilon^4 x_k^2) t_j T_j \end{aligned}$$

In order to keep the Chebyshev evaluation positive, we instead let $x = sr$,

where $s = -1, 1$. Some delicacy is required when $r = 0$, we then let $s = 1$.

$$\begin{aligned} e^{2\varepsilon^2 x x_k} &= \sum_{j=0}^{\infty} \frac{(2\varepsilon^2 s s_k r r_k)^j}{j!} \\ &= 2\sum_{j=0}^{\infty} \frac{\varepsilon^{2j} s^j s_k^j r_k^j}{j!} \sum_{l=0}^{\frac{j-p}{2}} \left(\begin{array}{c} j \\ l \end{array} \right) t_{j-2l} T_{j-2l} \\ &= 2\sum_{j=0}^{\infty} \frac{\varepsilon^{2j} r_k^j}{j!} \sum_{l=0}^{\infty} \varepsilon^{4l} x_k^{2l} \frac{j!}{(j+2l)!} \frac{(j+2l)!}{l!(j+l)!} t_j T_j \\ &= 2\sum_{j=0}^{\infty} \frac{(s s_k)^p \varepsilon^{2j} r_k^j}{j!} {}_0F_1([], j+1, \varepsilon^4 r_k^2) t_j T_j \end{aligned}$$

Then we get the final form of expansion

$$\phi(\underline{x}, \underline{x}_k) = \sum_{j=0}^{\infty} d_j c_j(x_k) \widetilde{T}_j(x), \tag{10}$$

with expansion functions

$$\widetilde{T}_j(x) = e^{-\varepsilon^2 x^2} T_j(x). \tag{11}$$

where

$$T_j(x) = \cos(\arccos(x_j)). \tag{12}$$

and with the scale factors and coefficients in our case are

$$d_j = \frac{2(s s_k)^p \varepsilon^{2j}}{j!}, c_j(x_k) = t_j e^{-\varepsilon^2 r_k^2} r_k^j \, {}_0F_1([], j+1, \varepsilon^4 r_k^2), \tag{13}$$

where $p = 0$ if j is even and $p = 1$ if j is odd.

3.2 The computation of the first derivative and the second derivative

Now we can compute the first derivative of $\widetilde{T}_j(x)$ to get

$$\frac{d}{dx}\widetilde{T}_j(x) = (-2\varepsilon^2 x e^{-\varepsilon^2 x^2})T_j(x) + (e^{-\varepsilon^2 x^2})\frac{d}{dx}T_j(x), \tag{14}$$

And the second derivative of $\widetilde{T}_j(x)$ is

$$\begin{aligned}
\frac{d^2}{dx^2}\widetilde{T}_j(x) &= \frac{d}{dx}(-2\varepsilon^2 x e^{-\varepsilon^2 x^2})T_j(x) - 2\varepsilon^2 x e^{-\varepsilon^2 x^2}\frac{d}{dx}T_j(x) \\
&\quad -2\varepsilon^2 x e^{-\varepsilon^2 x^2}\frac{d}{dx}T_j(x) + e^{-\varepsilon^2 x^2}\frac{d}{dx^2}T_j(x) \\
&= (4\varepsilon^4 x^2 e^{-\varepsilon^2 x^2} - 2\varepsilon^2 e^{-\varepsilon^2 x^2})T_j(x) \\
&\quad -4\varepsilon^2 x e^{-\varepsilon^2 x^2}\frac{d}{dx}T_j(x) + e^{-\varepsilon^2 x^2}\frac{d}{dx^2}T_j(x)
\end{aligned}$$

Then the derivatives of $\widetilde{T}_j(x)$ is used in the equation (5) in Section 2 to approximate the derivatives of example test functions.

3.3 Numerical experiments for the first and second derivatives

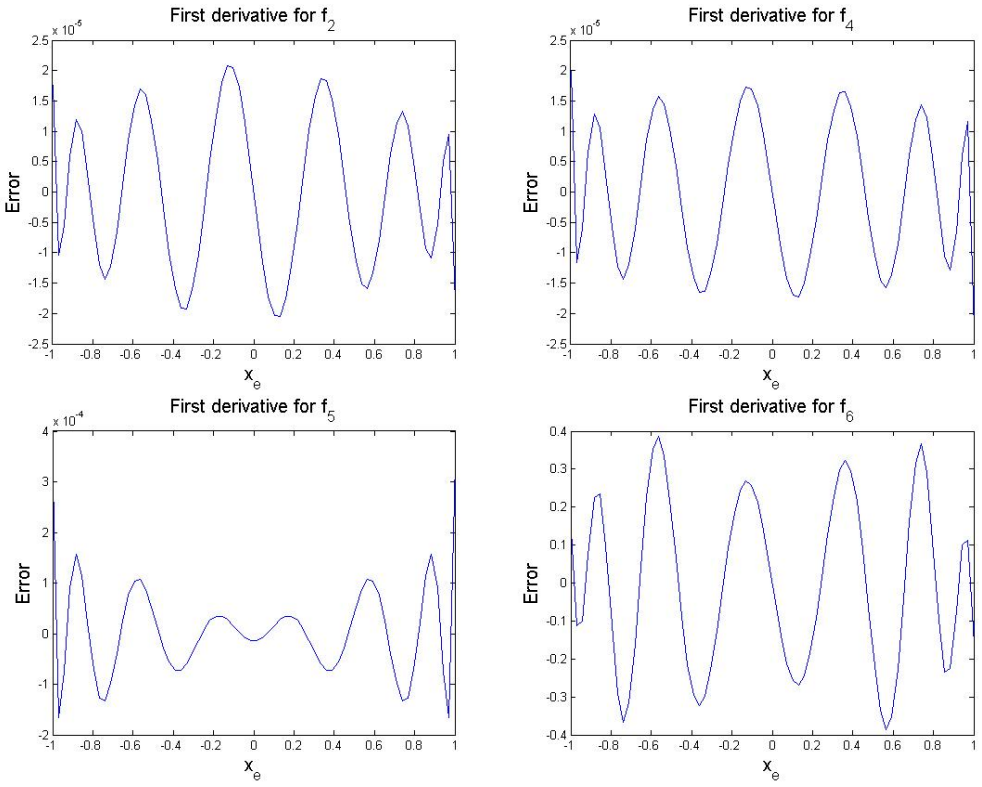

Figure 4: Derivative errors compared with the exact solution as a function of the evaluation points x_e for first derivatives of functions f_2, f_4, f_5 and f_6. In all cases $N = 14$ $\varepsilon = 0.1$ was used. All figures were shown using Chebyshev nodes.

The errors between the RBF approximation of the first and second derivative of four different test functions and the exact solution are shown in Figure 4 and Figure 5, repectively. The RBF-QR method can really get accurate results in the derivative approximation and the errors grow as the test functions have more variation or grow more complicated.

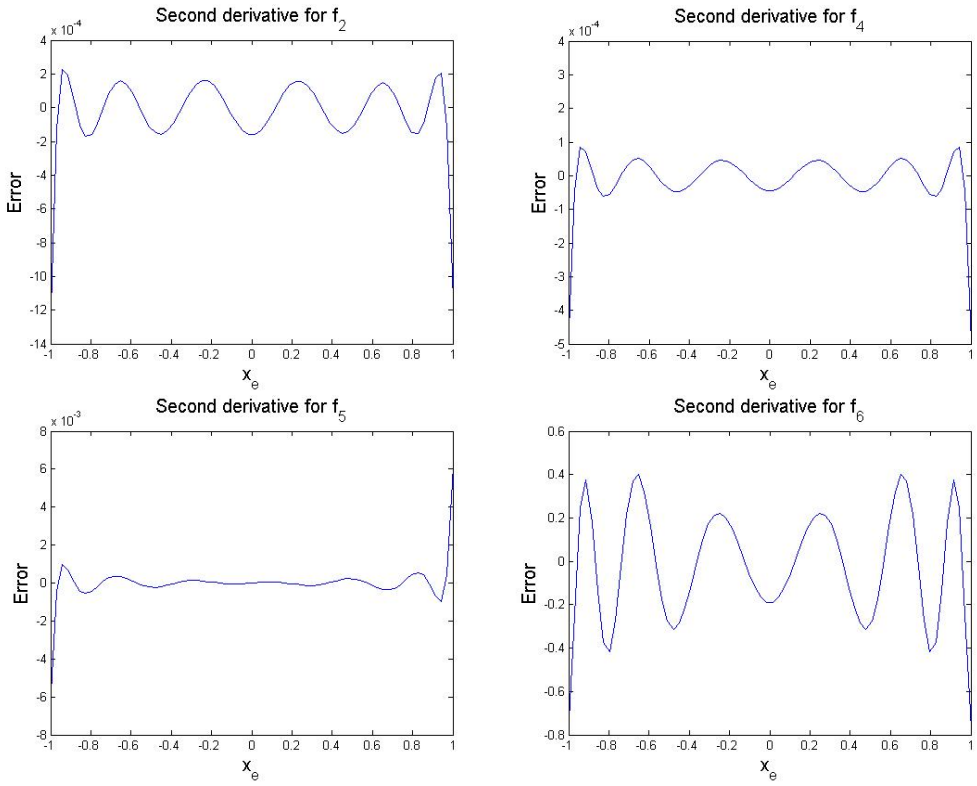

Figure 5: Derivative errors compared with the exact solution as a function of the evaluation points x_e for second derivatives of functions f_2, f_4, f_5 and f_6. In all cases $N = 14$ $\varepsilon = 0.1$ was used. All figures were shown using Chebyshev nodes.

Figure 6 shows maximum errors in the derivatives as a function of N with a fixed ε value and figure 7 shows maximum errors in the derivatives as a function of ε with a fixed N value. From both figure 6 and figure 7, we can clearly see that the interpolation approximations for the first and second derivatives of smooth test function f_5 can get a really accurate result with different number of node points and a range of ε values. Errors for uniform nodes grow with N. The most accurate result is achieved for a small non-zero value of ε in the region where RBF-QR is needed.

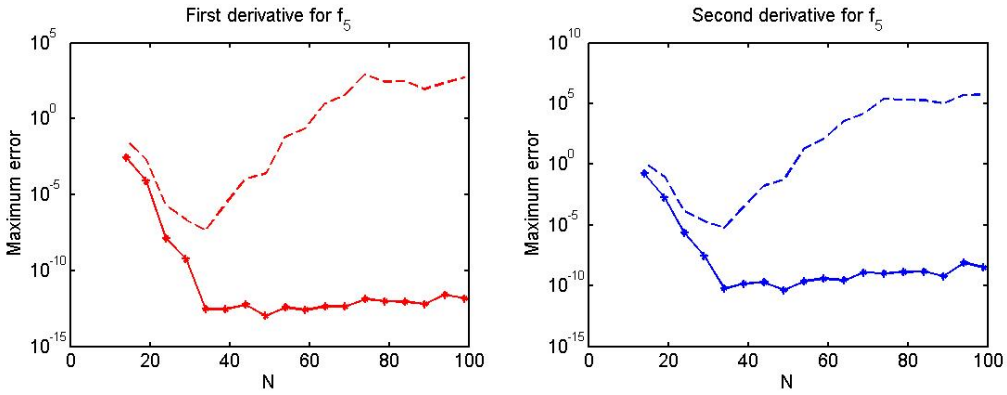

Figure 6: Derivative errors compared with the exact solution as a function of the evaluation points x_e for the first and second derivative of function f_5. Different number of points $N = [14 : 5 : 100]$ were used. All figures were shown using Chebyshev nodes (star lines) and uniform nodes (dashed lines) and with $\varepsilon = 2$.

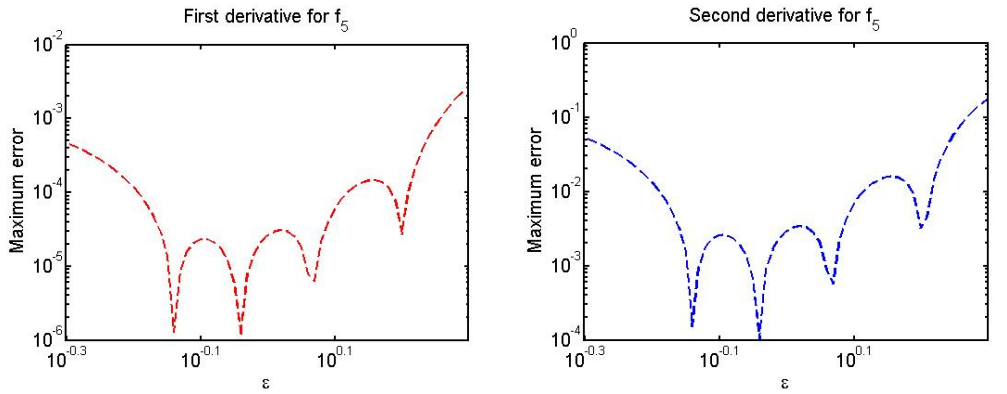

Figure 7: Derivative errors compared with the exact solution as a function of ε for the first and second derivative of function f_5. Different values of $\varepsilon = 10^{[log10(0.5):0.01:log10(2)]}$ were used. All figures were shown using Chebyshev nodes and with $N = 14$.

4 Modeling Black-Scholes equation

4.1 The Black-Scholes model

The multi-dimensional Black-Scholes partial differential equation (PDE) which is linear, time-dependent, and parabolic can be used for pricing of options based on several underlying assets. Here we use the transformed version of

the PDE and a European basket option as an example. Time is reversed to make standard texts on time-integration for PDEs applicable, and all variables have been scaled to be dimensionless. The details of the transformation can be found in [23], [28]. The transformed problem can be written

$$u_t(\underline{x},t) = Lu(\underline{x},t), \qquad \underline{x} \in \Omega, \qquad t > 0, \tag{15}$$
$$u(\underline{x},t) = g(\underline{x},t), \qquad \underline{x} \in \Gamma, \qquad t > 0, \tag{16}$$
$$u(\underline{x},0) = \Phi(\underline{x}), \qquad \underline{x} \in \Omega, \tag{17}$$

where, $\underline{x} \in R_+^d$ contains the scaled values of the d assets, t is the time left to the exercise time T of the option, Ω is some sub-region of R_+^d, and $u(\underline{x},t)$ is the value of the European option. The spatial operator in equation (15) has the form

$$Lu(\underline{x},t) = r\sum_{i=1}^{d} x_i \frac{\partial u}{\partial x_i} + \frac{1}{2}\sum_{i,j=1}^{d} [\sigma\sigma^T]_{ij} x_i x_j \frac{\partial^2 u}{\partial x_i \partial x_j} - ru, \tag{18}$$

where σ is the volatility matrix and r is the risk free interest rate. The contract function $\Phi(\underline{x})$ depends on the type of option we are solving for. In our numerical examples, we use the European basket call option with the following contract function

$$\Phi(\underline{x}) = \max(0, \frac{1}{d}\sum_{i=1}^{d} x_i - K), \tag{19}$$

where the exercise price K is always equal to 1 due to scaling. The boundary conditions are linked to the contract function [30]. At the near-field boundary, which defined as $\underline{x} = \underline{0}$ we use

$$u(\underline{x},t) = 0, \tag{20}$$

then at the far-yield boundary, here defined as the part of the boundary where $\frac{1}{d}\sum_{i=1}^{d} x_i \geq 4K$, then we impose

$$u(\underline{x},t) = \frac{1}{d}\sum_{i=1}^{d} x_i - K\exp(-rt), \tag{21}$$

Here we need to note that $\Gamma \subset \partial\Omega$, we do not impose boundary conditions on the whole boundary.

4.2 Notational conventions

The RBF solution is computed at discrete times and the approximation is expressed in terms of the coefficients of the basis functions. Here we use a systematic way to index variables and give the following conventions, see more details in [21]:

- Affiliation with a named set is denoted with a superscript, such as $\underline{x}_j^c$, where c stands for the set of center points.
- Indices related to space are presented as subscripts.
- Time is indicated by a superscript such as in $v^n(\underline{x})$, which is the approximate solution at the time t^n.
- The row index descriptors and column index descriptors of matrices are subscripts separated by a semicolon. For example, $A_{b;\lambda}$ has rows corresponding to the boundary points (b) and columns corresponding to the first partition (λ) of center points.

We also follow the notations above to make the MATLAB code easy to understand.

4.3 Approximation in space and discretization in time

Here we use the scheme which is spectrally accurate in space and second-order accurate in time for constant shape parameter. For other non-optimal choices of shape parameter values, the resulting convergence rate is algebraic, which has been proved by Pettersson et al. [30].

After using a time-dependent linear combination of N radial basis functions, centered at the points $\underline{x}_j^c$, $j = 1, ..., N$. we get the approximate solution

$$u(\underline{x}, t) = \sum_{j=1}^{N} \lambda_j(t)\phi(\|\underline{x} - \underline{x}_j^c\|) \equiv \sum_{j=1}^{n} \lambda_j(t)\phi_j(x). \tag{22}$$

Then we divide the time interval $[0, T]$ into M steps of length $k^n = t^n - t^{n-1}$, $n = 1, ..., M$, and denote the approximate solution at the discrete times t^n as

$$v^n(\underline{x}) = \sum_{j=1}^{n} \lambda_j^n \phi_j(\underline{x}) \approx u(\underline{x}, t^n). \tag{23}$$

Next our PDE problem is discretized in time using the unconditionally stable, second-order accurate, implicit BDF-2 method [3] and we get

$$v^1(\underline{x}) - k^1 L v^1(\underline{x}) = v^0(x), \quad x \in \Omega, \tag{24}$$

$$v^n(\underline{x}) - \beta_0^n L v^n(\underline{x}) = \beta_1^n v^{n-1}(\underline{x}) - \beta_2^n v^{n-2}(\underline{x}) \equiv f^n(x), \quad \underline{x} \in \Omega, \quad n = 2, ..., M, \tag{25}$$

$$v^n(\underline{x}) = g(\underline{x}, t^n) \equiv g^n(\underline{x}), \quad x \in \Gamma, \quad n = 1, ..., M, \tag{26}$$

$$v^0(\underline{x}) = \Phi(\underline{x}), \quad x \in \Omega, \tag{27}$$

where

$$\beta_0^n = k^n \frac{1 + \omega_n}{1 + 2\omega_n}, \quad \beta_1^n = \frac{(1 + \omega_n)^2}{1 + 2\omega_n}, \quad \beta_2^n = \frac{(\omega_n)^2}{1 + 2\omega_n}, \tag{28}$$

and $\omega_n = \frac{k^n}{k^{n-1}}$. After choosing ω_n such that $\beta_0^n = k_1 \equiv \beta_0$. we will have the same operator in the left hand sides of equations (24) and (25) for all time steps. And a recursion formula for this purpose was derived in [23].

4.4 The least squares formulation

Instead of collocation, which is a common formulation in the RBF context, we adopt a least squares scheme, which turns out to be more efficient to reduce the error in the region of most interest, where we mean stock prices close to the strike price.

The least squares scheme also fulfils the boundary conditions exactly and is used only for the discrete differential operator formulation. Let $\underline{x}_i^b$, $i = 1, ..., N_b$, be the points where we enforce boundary conditions and let $\underline{x}_i^{ls}$, $i = 1, ..., N_{ls}$, where $N_{ls} + N_b >= N$ be the equation points we use for the least squares fit of the Black-Scholes equation. Then the unknown coefficients at each time level are the two vectors $\underline{\lambda}^n = (\lambda_1^n, ..., \lambda_{N-N_b}^n)^T$ and $\underline{\mu}^n = (\lambda_{N-N_b+1}^n, ..., \lambda_N^n)^T$. Then we define the vectors $\underline{v}_{ls}^n = (v^n(\underline{x}_1^{ls}), ..., v^n(\underline{x}_{N_{ls}}^{ls}))^T$ and $\underline{v}_b^n = (v^n(\underline{x}_1^b), ..., v^n(\underline{x}_{N_b}^b))^T$. Then we can get the following relations

$$\underline{v}_\xi^n = A_{\xi;\lambda}\underline{\lambda}^n + A_{\xi;\mu}\underline{\mu}^n, \xi = b, ls \tag{29}$$

$$L\underline{v}_{ls}^n = B_{ls;\lambda}\underline{\lambda}^n + B_{ls;\mu}\underline{\mu}^n, \tag{30}$$

where

$$[A_{\xi;\lambda}]_{ij} = \psi_j(\underline{x}_i^\xi), i = 1, ..., N_\xi, j = 1, ..., N - N_b \tag{31}$$

$$[A_{\xi;\mu}]_{ij} = \psi_{j+N-N_b}(\underline{x}_i^\xi), i = 1, ..., N_\xi, j = 1, ..., N_b \tag{32}$$

$$[B_{ls;\lambda}]_{ij} = L\psi_j(\underline{x}_i^{ls}), i = 1, ..., N_{ls}, j = 1, ..., N - N_b \tag{33}$$

$$[B_{ls;\mu}]_{ij} = L\psi_{j+N-N_b}(\underline{x}_i^{ls}), i = 1, ..., N_{ls}, j = 1, ..., N_b \tag{34}$$

Then we substituting (29) and (30) into (24) − (27) and get the system of equations to solve for each time step as follows

$$\begin{pmatrix} A_{ls;\lambda} - \beta_0 B_{ls;\lambda} & A_{ls;\mu} - \beta_0 B_{ls;\mu} \\ A_{b;\lambda} & A_{b;\mu} \end{pmatrix} \begin{pmatrix} \underline{\lambda}^n \\ \underline{\mu}^n \end{pmatrix} = \begin{pmatrix} \underline{f}^n \\ \underline{g}^n \end{pmatrix}, \tag{35}$$

where we have

$$\begin{aligned}
\underline{f}^1 &= \underline{\upsilon}^0_{ls} = (\Phi(\underline{x}^{ls}_1), ..., \Phi(\underline{x}^{ls}_{N_{ls}}))^T, \\
\underline{f}^2 &= \beta^2_1 \begin{pmatrix} A_{ls;\lambda} & A_{ls;\mu} \end{pmatrix} \begin{pmatrix} \underline{\lambda}^1 \\ \underline{\mu}^1 \end{pmatrix} - \beta^2_2 \underline{\upsilon}^0_{ls}, \\
\underline{f}^n &= \begin{pmatrix} A_{ls;\lambda} & A_{ls;\mu} \end{pmatrix} \begin{pmatrix} \underline{\beta}^n_1 \begin{pmatrix} \underline{\lambda}^{n-1} \\ \underline{\mu}^{n-1} \end{pmatrix} - \underline{\beta}^n_2 \begin{pmatrix} \underline{\lambda}^{n-2} \\ \underline{\mu}^{n-2} \end{pmatrix} \end{pmatrix}, n = 2, ...M.
\end{aligned} \tag{36}$$

and $\underline{g}^n = (g^n(\underline{x}^b_1), ..., g^n(\underline{x}^b_{N_b}))^T$. Then we enforce the boundary conditions and eliminate μ^n from the first block row in the system of equations by block Gaussian elimination, resulting in the system

$$\begin{pmatrix} S_{ls;\lambda} & 0 \\ A_{b;\lambda} & A_{b;\mu} \end{pmatrix} \begin{pmatrix} \underline{\lambda}^n \\ \underline{\mu}^n \end{pmatrix} = \begin{pmatrix} \underline{f}^n - (A_{ls;\mu} - \beta_0 B_{ls;\mu}) A^{-1}_{b;\mu} \underline{g}^n \\ \underline{g}^n \end{pmatrix}, \tag{37}$$

where $S_{ls;\lambda} = (A_{ls;\lambda} - \beta_0 B_{ls;\lambda}) - (A_{ls;\mu} - \beta_0 B_{ls;\mu}) A^{-1}_{b;\mu} A_{b;\lambda}$, see more details in [21].

In this thesis, a least squares solution of the system (23) is implemented in MATLAB. The basic steps for the algorithm are presented as follows:

1. Solve $A_{b;\mu}\underline{\omega} = \underline{g}^n$.
2. Solve $S_{ls;\lambda}\underline{\lambda}^n = \underline{f}^n - (A_{ls;\mu} - \beta_0 B_{ls;\mu})\underline{\omega}$ in the least squares sense.
3. Solve $A_{b;\mu}\underline{\upsilon} = A_{b;\lambda}\underline{\lambda}^n$.
4. Compute $\underline{\mu}^n = \underline{\omega} - \underline{\upsilon}$.

Because we use the special choice of time step, see more details in [7], the coefficient matrices in Step 1-3 above are all constant and can be factorized once prior to the time stepping. Then we have to do the following computations first

1. Factorize $A_{b;\mu} = LU$.
2. Form $S_{ls;\lambda}$ using the factorization of $A_{b;\mu}$.
3. Factorize $S_{ls;\lambda} = QR$.

These factorizations are then applied in the usual way in our time stepping algorithm.

4.5 Numerical experiments

All the following numerical experiments are implemented in the one-dimensional case, where the number of assets $d = 1$.

Figure 8 shows the node points x_c and x_b, least squares points x_{ls} and evaluation points x_e which we use to evaluate the solution and errors in our Black-Scholes model problem. Here we always use five times the number of the center points as least squares points to make sure to have a fine enough grid.

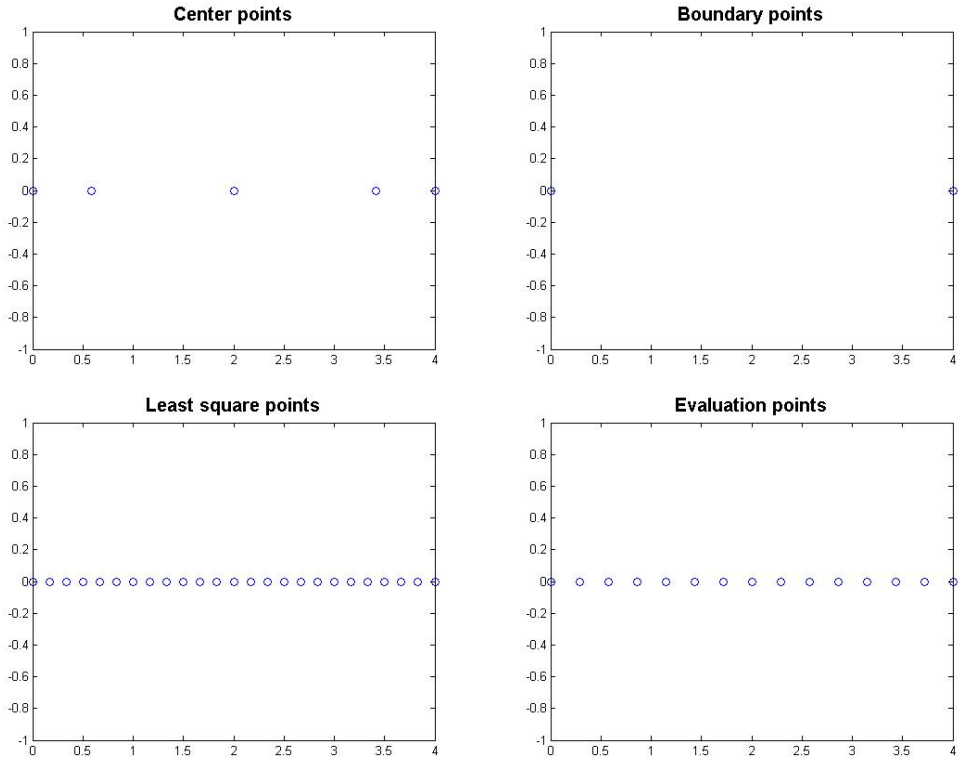

Figure 8: The center points with $N = 5$ with two boundary points, the evaluation points with $N_e = 15$ and the least squares points $N_{ls} = 5 \times N$. All the points are scaled in the interval [0,4]. All figures were shown using uniform nodes.

Maxnorm errors with different values of ε are shown in Figure 9 for both the RBF-Direct method and the RBF-QR method. Here we use the test values for ε as $\varepsilon = 10^{[log10(0.5):0.01:log10(2)]}$. From the figure, we can see apparently that when ε is small, it is better use the RBF-QR method, otherwise, when ε is large, it is better use the RBF-Direct method. Chebyshev points generate

better results than uniform points in all the three cases.

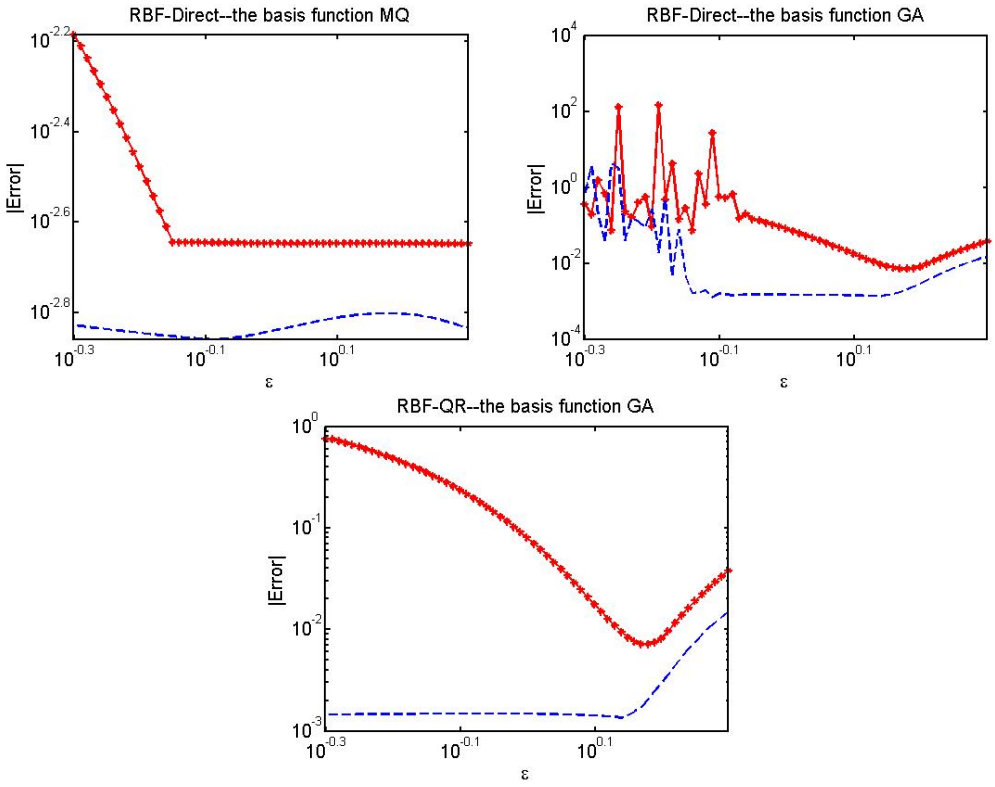

Figure 9: Maxnorm errors compared with the exact solution as a function of ε with different basis function MQ and GA. As the RBF-QR method just exists for GA basis function, so we just show one kind of basis funcion here. Here $N = 20$ $M = 120$ were used. The Black-Scholes model is solved with both the RBF-Direct method and the RBF-QR method here. All figures were shown using uniform nodes (star lines) and Chebyshev points (dashed lines).

In Figure 10 the errors are pretty small for different values of N, which means our RBF-QR method in the collocation approach and the least squares approach both work well for the Black-Scholes model. We can also find that the least squares approach performs better result than the collocation approach.

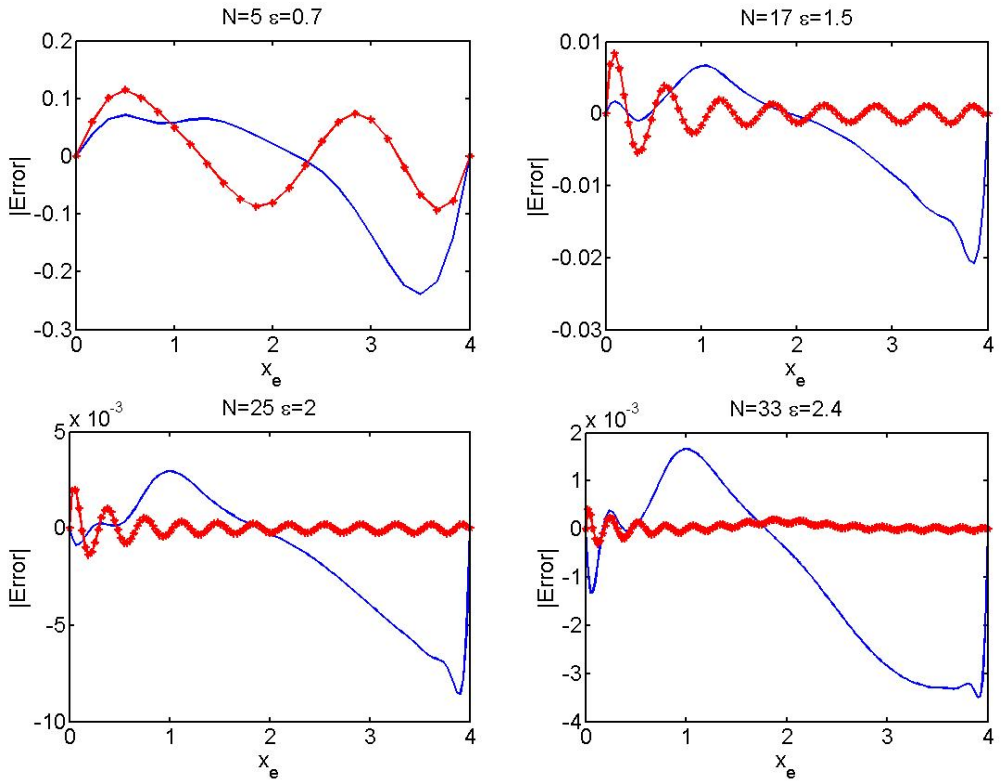

Figure 10: Errors compared with the exact solution as a function of the evaluation points x_e. Different values of N: $N = 5$, $N = 17$, $N = 25$, $N = 33$, with optimal ε: $\varepsilon = 0.7$, $\varepsilon = 1.5$, $\varepsilon = 2$, $\varepsilon = 2.4$, were used. $M = 120$ $N_{ls} = 10 \times N$ in all cases. All figures were shown using uniform nodes and the basis function GA in the collocation approach (solid lines) and the least squares approach (star lines).

Figure 11 compares the time errors using the RBF-QR method to solve the Black-Scholes equation in the collocation approach and the least square approach. The least squares approach performs better result than the collocation approach.

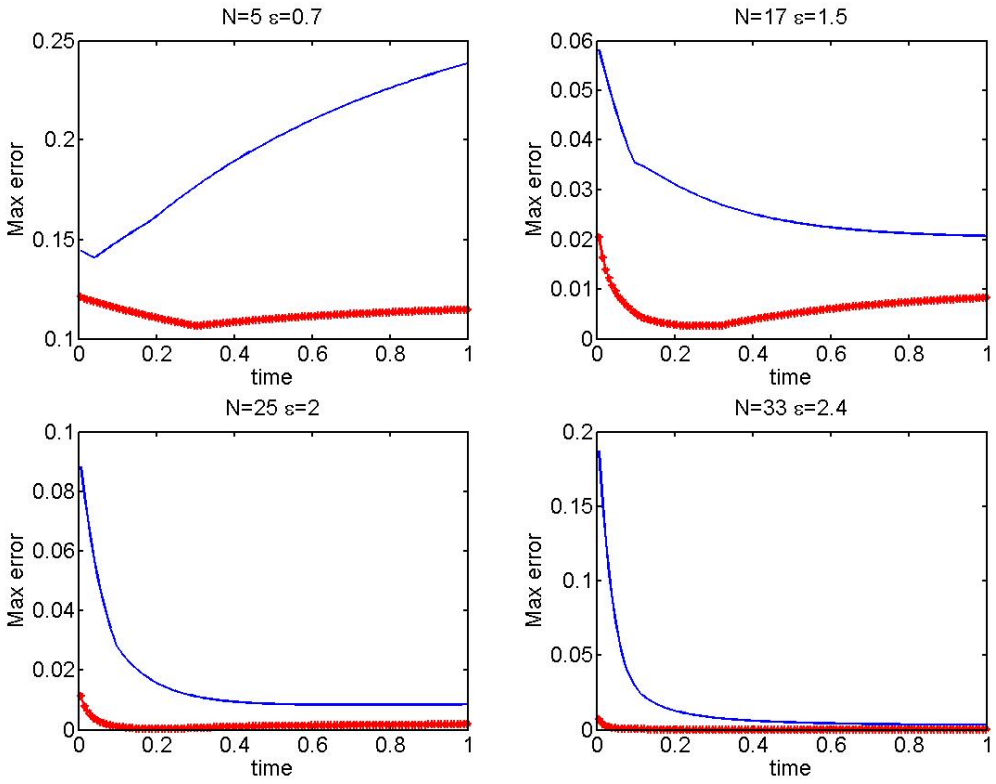

Figure 11: Time errors as a function of the time. Different values of N: $N = 5$, $N = 17$, $N = 25$, $N = 33$, with optimal ε: $\varepsilon = 0.7$, $\varepsilon = 1.5$, $\varepsilon = 2$, $\varepsilon = 2.4$, were used. $M = 120$ $N_{ls} = 10 \times N$ in all cases. All figures were shown using uniform nodes and the basis function GA in the collocation approach (solid lines) and the least squares approach (star lines).

In Figure 12 the errors are pretty small for different values of N, which means our RBF-QR method in the collocation approach works well for the Black-Scholes model and gets accurate result. After comparing two kinds of points used here, we can find that the Chebyshev points performs better result than the uniform points and the Chebyshev points gets better result as N becomes larger.

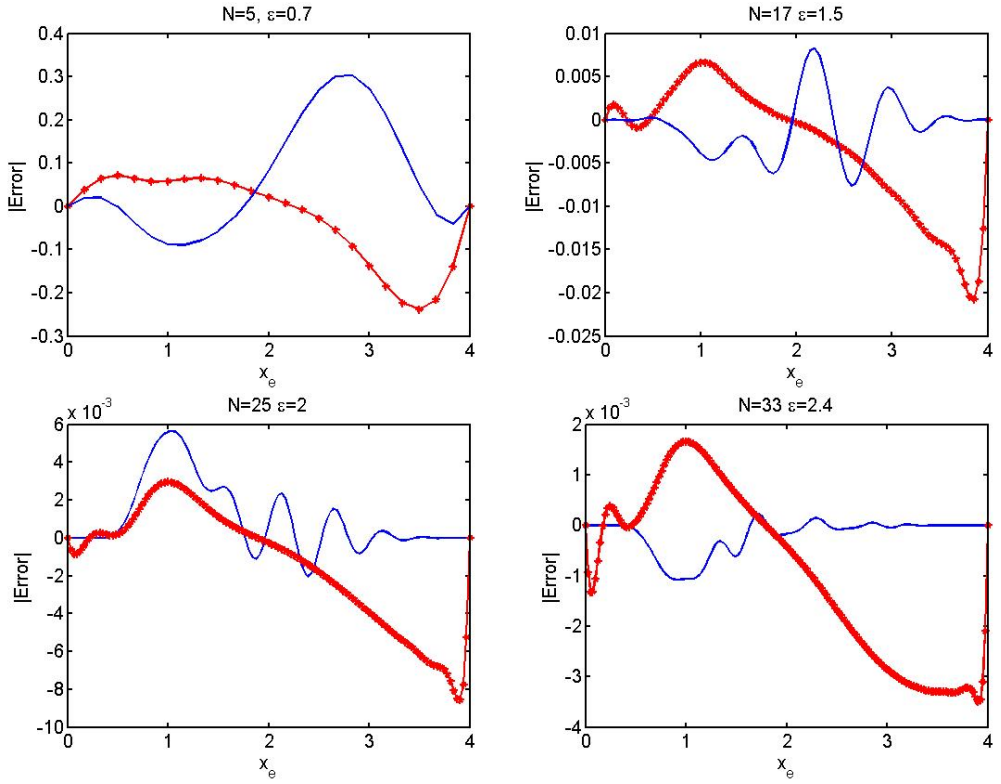

Figure 12: Errors compared with the exact solution as a function of the evaluation points x_e. Different values of N: $N = 5$, $N = 17$, $N = 25$, $N = 33$, with optimal ε: $\varepsilon = 0.7$, $\varepsilon = 1.5$, $\varepsilon = 2$, $\varepsilon = 2.4$, were used. $M = 120$ in all cases. All figures were shown using uniform nodes (star lines) and Chebyshev nodes (solid lines) with the basis function GA in the collocation approach.

Figure 13 shows the errors using the RBF-QR method to solve the Black-Scholes equation in the least squares approach with the Chebyshev points and the uniform points. With growing N, the time errors decrease. After comparing, we can find that the Chebyshev points performs better results than the uniform points.

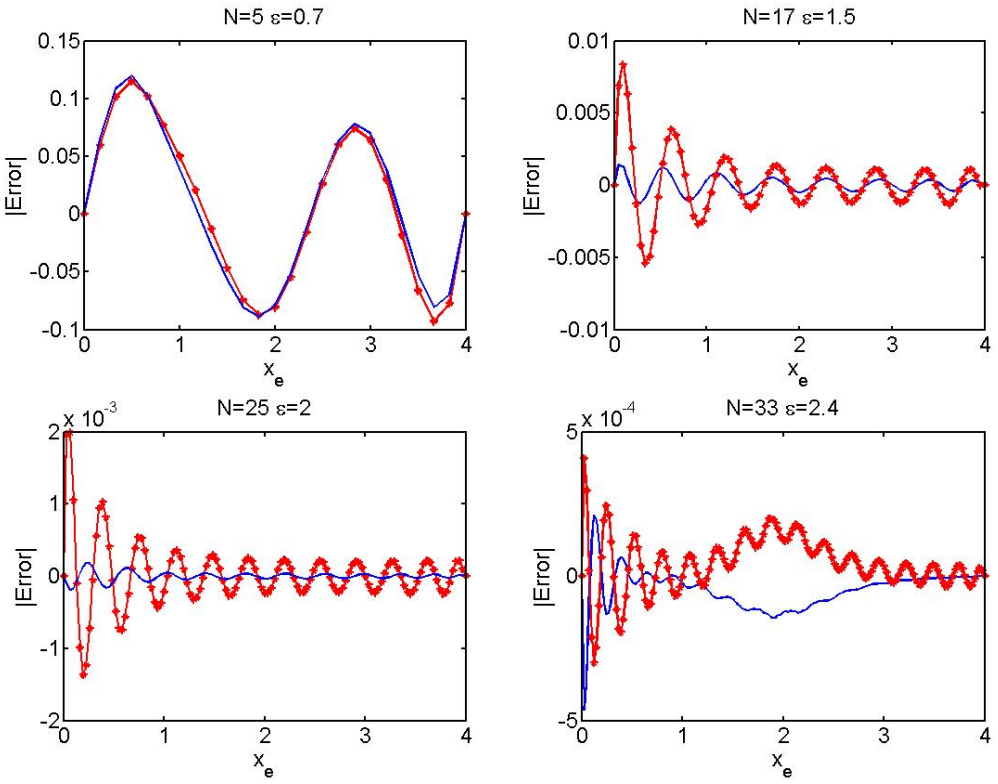

Figure 13: Errors compared with the exact solution as a function of the evaluation points x_e. Different values of N: $N = 5$, $N = 17$, $N = 25$, $N = 33$, with optimal ε: $\varepsilon = 0.7$, $\varepsilon = 1.5$, $\varepsilon = 2$, $\varepsilon = 2.4$, were used. $M = 120$ $N_{ls} = 10 \times N$ in all cases. All figures were shown using uniform nodes (star lines) and Chebyshev points (solid lines) with the basis function GA in the least squares approach.

In Figure 14 the errors become smaller when the number of the least squares points N_{ls} increases and our RBF-QR method works well for the Black-Scholes model.

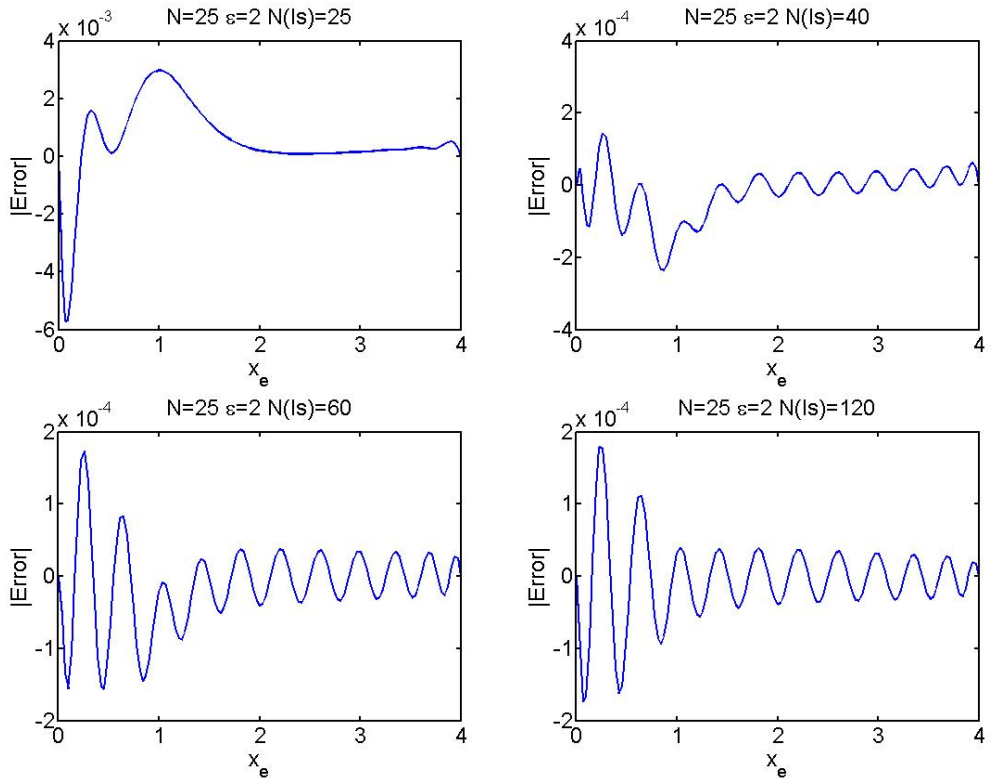

Figure 14: Errors compared with the exact solution as a function of the evaluation points x_e. Different values of N_{ls}: $N_{ls} = 25$ $N_{ls} = 40$ $N_{ls} = 60$ $N_{ls} = 120$ were used. $M = 120$ $N = 25$ $\varepsilon = 2$ were used. All figures were shown using uniform nodes and the basis function GA in the least squares approach.

Figure 15 compare the maximum errors with the uniform points and the Chebyshev points. The Chebyshev points improves the accuracy compared with a uniform distribution for small ε. For large ε, it is less obvious and it may work with both points, possibly the uniform points are better. We can also find there is a gap in the middle where none of the methods work.

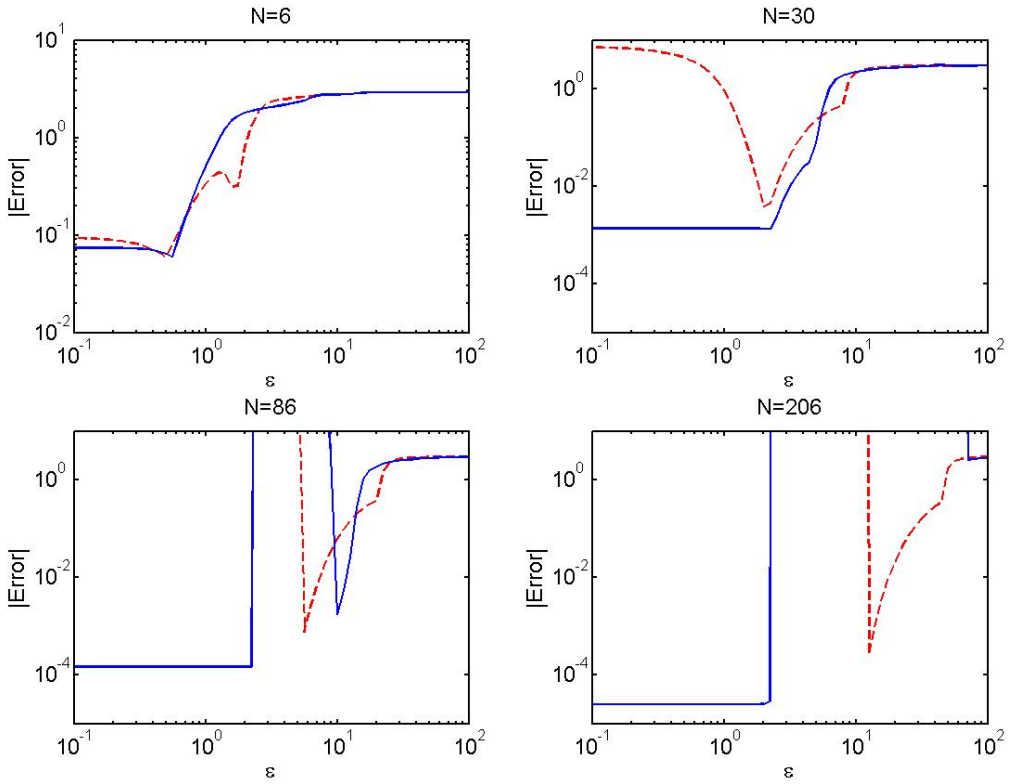

Figure 15: Max-norm errors compared with the exact solution as a funtion of ε. Different values of N $N = 6$ $N = 56$ $N = 182$ $N = 800$ were used. Here $\varepsilon = 10^{[log10(0.1):0.05:log10(20)]}$ was used for each value of N. All figures were shown using uniform nodes (dashed lines) and Chebyshev points (solid lines). The basis function GA was used in the collocation approach here.

5 Conclusions

In this thesis we have derived an RBF-QR method for option pricing with infinitely smooth Gaussian RBFs in one dimension, including boundary conditions. We have shown that it can be difficult to take full advantage of the spectral property due to the ill-conditioning of the RBF-Direct matrices for small shape parameter values. So we have to strike a favorable balance between unavoidable accuracy losses for large ε and avoidable RBF-Direct accuracy losses for low values of ε.

The RBF-QR method is numerically stable for small shape parameters, even when $\varepsilon \to 0$. It is the only numerical algorithm that can deliver an accurate result for small ε with the range of intermediate to large N.

Furthermore, we have shown how an adapted placement of the node points such as Chebyshev points, instead of a standard uniform distribution, can increase the accuracy by up to an order of magnitude by performing the clustering towards the boundary.

For the type of PDE application that we have studied here, the LS formulation for RBF methods is preferable to the collocation approach. This is because typically the LS approximation makes the error small in the interesting region.

We conclude that overall, the RBF-QR method performs significantly better than the RBF-Direct method. There are further improvements to be made such as evaluating the efficiency of the methods more closely and trying to find a rule for choosing optimal parameters, and we expect that the RBF-QR method for option pricing will be competitive in higher dimensions also.

Acknowledgements

First of all, I would like to thank my advisor, Elisabeth Larsson for all her encouragement and support, and more importantly for generously sharing her invaluable expertise. I deeply believe that my career will benefit a lot from the eight-month's study under your supervision.

I would like to thank my reviewer, Lina von Sydow for her constructive criticism which led to improvements both in the contents and the presentation of the thesis.

Many thanks to my thesis coordinator, Olle Eriksson for his help to settle down my presentation time and all the other problems I met.

References

[1] A. Belova, M. Ehrhardt, and T. Shmidt, *Meshfree methods in option pricing*, Mediterranean Conference on Embedded Computing, **21** (2012), 243–246.

[2] F. Black, and M. Scholes, *The pricing of options and corporate liabilities*, J. Polit. Econ., **81** (1973), 637–654.

[3] M. D. Buhmann, S. Dinew, and E. Larsson, *A note on radial basis function interpolant limits*, IMA J. Numer. Anal., **30** (2010), 543–554.

[4] M. D. Buhmann, and N. Dyn, *Spectral convergence of multiquadric interpolation*, Proc. Edinburgh Math. Soc., **36** (1993), 319–333.

[5] M. D. Buhmann, *Radial basis functions: theory and implementations*, Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press, Cambridge, UK, **12** (2003).

[6] H. J. Bungartz, and M. Griebel, *Sparse grids*, Acta Numerica, **13** (2004), 147-269.

[7] T. A. Driscoll, B. Fornberg, *Interpolation in the limit of increasingly flat radial basis functions*, Comput. Math. Appl., **43** (2002), 413–422.

[8] G. E. Fasshauer, and A. Q. M. Khaliq, *Using mesh-free approxiamtion for multi-asset american option problems*, Journal of Chinese Institute Engineers, **27** (2004), 563–571.

[9] G. E. Fasshauer, *Meshfree approximation methods with MATLAB*, Interdisciplinary Mathematical Sciences, World Scientific Publishing Co. Pte. Ltd., **6** (2007).

[10] B. Fornberg, E. Larsson, and N. Flyer, *Stable computations with Gaussian radial basis functions*, SIAM J. Sci. Comput., **33** (2010), 869–892.

[11] B. Fornberg, G. Wright, and E. Larsson, *Some observations regarding interpolatants in the limit of flat radial basis functions*, Comput. Math. Appl., **47** (2004), 37–55.

[12] B. Fornberg, T. A. Driscoll, G. Wright, and R. Charles, *Observations on the behavior of radial basis function approximations near boundaries*, Comput. Math. Appl., **43** (2002), 473–490.

[13] E. H. Georgoulis, J. Levesley, and F. Subhan, *Multilevel sparse kernel-based interpolation*, Thesis for the degree of Doctor of Philosophy, Department of Mathematics, University of Leicester, England, United Kingdom, (2011).

[14] P. Glasserman, *Monte Carlo Methods in Financial Engineering*, Springer-Verlag, (2004).

[15] A. Golbabai, D. Ahmadian, and M. Milev, *Radial basis functions with application to finance: American put option under jump diffusion*, Mathematical and Computer Modelling, **55** (2012), 1354–1362.

[16] A. Guarin, X. Q. Liu, and W. L. Ng, *Enhancing credit default swap valuation with meshfree methods*, European Journal of Operational Research, **214** (2011), 805-813.

[17] Y. C. Hon, and X. Z. Mao, *A radial basis function method for solving options pricing model*, Journal of Financial Engineering, **8** (1999), 1–24.

[18] Y. C. Hon, *A quasi-radial basis functions method for American options pricing*, Appl. Math. Comput., **43** (2002), 513–524.

[19] M. H.M. Khabir, K. C. Patidar, *Spline approximation method to solve an option pricing problem*, Journal of Difference Equations and Applications, **06** (2012).

[20] E. Larsson, E. Lehto, A. Heryudono, and B. Fornberg, *Stable computation of differentiation matrices and scattered node stencils based on gaussian radial basis functions*, Uppsala University, Technical Report 2012–020, 1-3, https://www.it.uu.se/research/publications/reports/2012-020/.

[21] E. Larsson, S. Gomes, *A least squares multi-level RBF method with applications in finance*, manuscript in preparation.

[22] E. Larsson, B. Fornberg, *A Numerical study of Some Radial Basis Function Based Solution Methods for Elliptic PDEs*, Computers and Mathematics with Applications , **46** (2003), 891–902.

[23] E. Larsson, K. Åhlander, and A. Hall, *Multi-dimensional option pricing using radial basis functions and the generalized Fourier transform*, J. Comput. Appl. Math., **222** (2008), 175-192.

[24] E. Larsson, and B. Fornberg, *Theoretical and computational aspects of multivariate interpolation with increasingly flat radial basis functions*, Comput. Math. Appl., **49** (2005), 103–130.

[25] P. Lötstedt, J. Persson, L. von Sydow, and J. Tysk, *Space-time adaptive finite difference method for European multi-asset options*, Comput. Math. Appl., **53** (2007), 1159–1180.

[26] W. R. Madych, and S. A. Nelson, *Bounds on multivariate polynomials and exponential error estimates for multiquadric interpolation*, J. Approx. Theory., **70** (1992), 94–114.

[27] M. D. Marcozzi, S. Choi and C. S. Chen, *On the use of boundary conditions for variational formulations arising in financial mathematics*, Appl. Math. Comput., **124** (2001), 197–214.

[28] J. Persson, L. Von Sydow, *Pricing European multi-asset options using a space-time adaptive FD-method*, Comput. Vis. Sci., **10** (2007), 173–183.

[29] V. Petersdorf, and C. Schwab, *Numerical solutions of parabolic equations in high dimensions*, M2AN Math. Model. Numer. Anal., **38** (2004), 93–128.

[30] U. Pettersson, E. Larsson, G. Marcusson, and J. Persson, *Improved radial basis function methods for multi-dimensional option pricing*, J. Comput. Appl. Math., **222** (2008), 82–93.

[31] R. U. Seydel, *Tools for Computational Finance*, Third Edition.

[32] D. Tavella, and C. Randall, *Pricing Financial Instruments: The Finite Difference Method*, John Wiley Sons, Inc., New York, (2000).

[33] H. Wendland, *Scattered data approximation*, Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press, Cambridge, **17** (2005).

[34] Z. Wu, and Y. C. Hon, *Convergence error estimate in solving free boundary diffusion problem by radial basis functions method*, Engrg. Anal. Bound. Elem., **27** (2003), 73–79.