MESHFREE METHODS FOR THE THE BLACK-SCHOLES PARTIAL DIFFERENTIAL EQUATION

A Thesis

Presented to the Faculty of the Department of Mathematical Sciences

Middle Tennessee State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Mathematical Sciences

---

by

Nana Akwasi Abayie Boateng

June 2012

APPROVAL

This is to certify that the Graduate Committee of

Nana Akwasi Abayie Boateng

met on the

14th day of June, 2012.

The committee read and examined his/her thesis, supervised his/her defense of it in an oral examination, and decided to recommend that his/her study should be submitted to the Graduate Council, in partial fulfillment of the requirements for the degree of Master of Science in Mathematics.

Dr. A.Q.M.Khaliq
Chair, Graduate Committee

Dr. Zachariah Sinkala

Dr. Yuri Melnikov

Dr. James Hart
Graduate Coordinator,
Department of Mathematical Sciences

the Graduate Council

Dr. Michael Allen
Dean,
School of Graduate Studies

ABSTRACT

Meshfree radial basis functions (RBF) is an interpolation technique for constructing an unknown function from scattered data. We apply the RBF method in evaluating the price of standard American options. The analytical solution of the European option exists and can be obtained by the Black-Scholes formula. There is no exact solution of the American option problem due to the existence of an early exercise constraint which leads to a free boundary condition. We evaluate the American Option by adding a small continuous nonlinear penalty term to the Black-Scholes model to remove the free boundary condition. The application of RBFs leads to a system ordinary differential equations which are solved by a time integration scheme known as the θ-method. The option price is approximated with RBF with unknown parameters at each time step. We compare the accuracy, efficiency and computational cost of three RBFs Gaussian, Multiquadric and the Inverse-multiquadric. Finally a comparison is made between the three RBFs and the solution obtained by finite difference approximations.

DEDICATION

This thesis is dedicated to my parents. I am forever grateful to them for their unconditional love,prayers,material and emotional support throughout my whole life. I pray the good Lord reward them for all their sacrifices in my life.

ACKNOWLEDGMENTS

I am thankful to my Lord Jesus Christ for granting me the strength to make this work a possibility. I cannot forget about the enormous help I have received from my advisor Dr. Khaliq throughout the period of writing this thesis. I thank him so much for all his advice,suggestions and direction. I am particularly grateful to the graduate coordinator Dr. Hart for all his concern and guidance for the time I have spent here as a student in the Mathematics department. I also would like to say a special thank you to my thesis committee members Dr.Sinkala and Dr. Melnikov for their role in making this whole work a success.

Contents

List of Tables

List of Figures

CHAPTER 1

INTRODUCTION

An option is a financial contract which gives the holder of the option the right to purchase or sell a prescribed asset at a prescribed time in the future known as the expiry date at a prescribed amount which is the exercise or strike price.

In 1973,Fisher Black and Myron Scholes showed that the option value of the European call option can be modeled by a lognormal diffusion partial differential equation.There are two categories of options namely, standard options(European and American options) and non standard options.

Hon and Mao[12] introduced a numerical scheme in which by applying global radial basis function as a spatial approximation for the numerical solution of the value of the option and it's derivative in the Black-Scholes equation. From their numerical results,they showed that the use of RBFs does not require the generation of a rectangular grid and also the computational domain is composed of scattered data points.

Khaliq *et al*[2] investigated meshfree RBF approximation to options with non-smooth payouts. By taking advantage of parallel architecture, they developed a strongly stable time stepping fourth order method which was a linear combination of four Backward Euler-like solver on four concurrent processors. Khaliq *et al*[1] considered a penalty method approach to solving American options. They observed that by introducing a carefully chosen continuous penalty term to the Black-Scholes equation, the free and moving boundary condition can be removed and allow the problem to be solved on a fixed domain. They introduced a linearly implicit scheme with superior accuracy and stability by solving the nonlinear term explicitly.

The remaining chapters of this thesis are organized as follows. We introduce op-

tion pricing and discuss two standard options, the European and American options in chapter 2. In chapter 3 we implement the θ-method ($\theta = 0$), the backward Euler method to evaluate the price of the option using finite difference approximation. The theory and development of RBF Meshfree methods is also presented here. In chapter 4, we elaborate on the discretization , algorithms of the RBFs methods and the stability analysis of the numerical scheme. Finally all numerical results from the experiments are presented in chapter 5 and interpreted in chapter 6.

CHAPTER 2

OPTION PRICING

An option is a financial contract which gives the holder of the option the right to purchase or sell a prescribed asset at a prescribed time in the future known as the expiry date at a prescribed amount which the exercise or strike price[10]. The most common kinds of prescribed assets which are traded on financial markets are stocks, bonds, currency and commodities. An option is a derivative product because it is traded on an underlying asset. The holder of a call option makes profit if the price of the underlying asset rises on the market whereas the holder of a put option does so when the price of the underlying asset falls on the financial market. The two primary uses of option are for hedging and speculation[10]. There are numerous kinds of options which are traded on financial markets. Vanilla options are options which do not possess any special features or characteristics. Examples are the European and American options. Exotic options possess special features. Examples include Asian options,Barrier options,Basket options. In this thesis, we consider the pricing of American option using a penalty method approach for Black-Scholes partial differential equation.

2.1 The European Option

The European option is an option which can only be exercised at its maturity time. The exact or analytical formula for estimating a fair price for the European options exist. In 1973 Fisher Black and Myron Scholes by making a set of explicit assumptions including the risk-neutrality of the underlying asset price showed that the value of the European call option satisfies a backward -in-time lognormal partial differ-

ential equation of diffusion type. This has come to be known as the Black-Scholes equation[7]. Let the $V(S,t)$ be the price of an option which is function of both asset price and time. This option satisfies the following Black-Scholes equation.

$$\frac{\partial P}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 P}{\partial S^2} + rS\frac{\partial P}{\partial S} - rP = 0, \quad 0 \leq t < T. \tag{2.1.1}$$

where r is the risk-free interest interest rate, σ is the volatility of the asset price, S is the asset price. The Final condition is given by[10].

$$V(S,t) = \begin{cases} max\{E-S,0\} & \text{for a put option} \\ max\{S-E,0\} & \text{for a call option} \end{cases} \tag{2.1.2}$$

where E is the strike price.

The Boundary condition of the European call option is given as follows:

$$C(S,t) \sim S \quad asS \to \infty, \quad C(0,t) = 0. \tag{2.1.3}$$

where $C(S,t)$ is the value of the European call option satisfying (2.1.1). The Boundary condition at time t of the European put option is given as follows:

$$P(S,t) \to 0 \quad as \quad S \to \infty, \quad P(0,t) = E\exp^{-\int_t^T \tau(\tau)d\tau} \tag{2.1.4}$$

where $P(S,t)$ is the value of the European put option satisfying equation (2.1.1), for a time dependent interest rate.

Equation (2.1.1) can be transformed exponentially by making the substitution $S = e^y$ to

$$\frac{\partial U}{\partial t} + \frac{1}{2}\sigma^2 \frac{\partial^2 U}{\partial y^2} + (r - \frac{1}{2}\sigma^2 \frac{\partial U}{\partial y}) - rU = 0 \tag{2.1.5}$$

$$U(y,T) = \begin{cases} max\{E-e^y,0\} & \text{for a put option} \\ max\{e^y-E,0\} & \text{for a call option} \end{cases}$$

The analytical solution of the Black- Scholes partial differential equation (2.1.1) with corresponding final and initial conditions (2.1.2) and (2.1.4) with a constant volatility and interest rate for the European call option is given as[10]

$$C(S,t) = SN(d_1) - E\exp^{-r(T-t)}N(d_2) \tag{2.1.6}$$

where $N(.)$ is the cumulative distribution function for the standardized normal random variable given by

$$N(x) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x}\exp^{-\frac{1}{2}y^2 dy} \tag{2.1.7}$$

The corresponding analytical solution of the European put option is given by

$$P(S,t) = E\exp^{-r(T-t)}N(-d_2) - SN(-d_1) \tag{2.1.8}$$

where

$$d_1 = \frac{\log(\frac{S}{E} + (r + \frac{1}{2}\sigma^2)(T-t))}{\sigma\sqrt{T-t}}$$
$$d_2 = \frac{\log(\frac{S}{E} + (r - \frac{1}{2}\sigma^2)(T-t))}{\sigma\sqrt{T-t}}$$

2.2 The American Option

The American option can be exercised at any time prior to expiry. The American option is complicated because at at each time t not only is one interested in the value of the option but also for each asset price S,whether it should be exercised or not. This creates a free boundary problem[10]. At each time t there is a particular value of S which lies in the boundary between two regions: one where early exercise is optimal to the other where one should hold on to the option. The optimal exercise price $S_f(t)$ which in general depends on time is not known *priori* unlike the case of

European options. The American option valuation can be uniquely specified by a set of constraints among which are that the option value must be greater than or equal to the payoff function, the option value must be a continuous function of S, replacing the Black-Scholes equation by an inequality and lastly making the derivative of the option with respect to the asset price(option delta) continuous. The value $V(S,t)$ of the American option satisfies the following inequality

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV \leq 0 \tag{2.2.1}$$

The Final condition is at expiry time T given by[10]

$$V(S,t) = \begin{cases} max\{E-S,0\} & \text{for a put option} \\ max\{S-E,0\} & \text{for a call option} \end{cases}$$

where E is the strike price.

In the region $0 \leq S \leq S_f(t)$ where early exercise is optimal, the value of the American put option satisfies the following inequality

$$\frac{\partial P}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 P}{\partial S^2} + rS\frac{\partial P}{\partial S} - rP < 0 \tag{2.2.2}$$

and

$$P = E - S \tag{2.2.3}$$

In the other region, $S_f(t) < S < \infty$, early exercise is not optimal and the value of the American put option satisfies the Black-Scholes equation

$$\frac{\partial P}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 P}{\partial S^2} + rS\frac{\partial P}{\partial S} - rP = 0 \tag{2.2.4}$$

and

$$P > E - S \tag{2.2.5}$$

The boundary condition at $S_f(t) = S$ are that P and its slope(delta) are continuous.

$$P(S_f(t), t) = max(E - S_f(t), 0) \tag{2.2.6}$$

$$\frac{\partial P}{\partial S}(S_f(t), t) = -1 \tag{2.2.7}$$

The boundary condition(2.2.6) determines the option value at the free boundary, whereas (2.2.7)known as the *smooth pasting condition* determines the location of the free boundary and simultaneously maximizes the benefit to the holder whiles avoiding arbitrage. The value $C(S, t)$ of the American Call option satisfies the corresponding equality in the holding region $0 \leq S \leq S_f(t)$

$$\frac{\partial C}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 C}{\partial S^2} + rS\frac{\partial C}{\partial S} - rC = 0 \tag{2.2.8}$$

in the other region where early exercise is optimal $S_f(t) < S < \infty$,the value $C(S, t)$ of the American call option satisfies

$$\frac{\partial C}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 C}{\partial S^2} + rS\frac{\partial C}{\partial S} - rC < 0 \tag{2.2.9}$$

and

$$P = E - S \tag{2.2.10}$$

2.2.1 The Penalty Method

We introduce a penalty term into the Black-Scholes equation to obtain a parabolic nonlinear partial differential equation[5]. The introduction of the penalty term changes the problem from that of a constrained optimization problem to that of a series of unconstrained optimization problem. The solution of the unconstrained optimization

problems converges to the original constrained optimization problem.[5]

$$\frac{\partial P_\epsilon}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 P_\epsilon}{\partial S^2} + rS\frac{\partial P_\epsilon}{\partial S} - rP_\epsilon + \frac{\epsilon C}{P_\epsilon + \epsilon - q(S)} = 0, \quad 0 \le S \le S_\infty,\ 0 \le t < T. \tag{2.2.11}$$

The addition of a penalty term allows the problem to be solved on a fixed domain.Here $0 < \epsilon \ll 1$ is a small regularization parameter, $C \ge rE$ is a positive constant, and $q(S) = E - S$ is the barrier function. The value S_∞ is a (relatively very large) price for which the option is worthless. The following are terminal and boundary conditions which accompany the nonlinear partial differential equation[5].

$$P_\epsilon(S, T) = \max(E - S, 0) \tag{2.2.12}$$

$$P_\epsilon(0, t) = E, \tag{2.2.13}$$

$$P_\epsilon(S_\infty, t) = 0. \tag{2.2.14}$$

CHAPTER 3

FINITE DIFFERENCE AND RBF APPROXIMATIONS

The method of Finite Difference Approximation which is based on Taylor series expansion of functions near the point of interest is be used to discretize the Black-Scholes partial differential equation[10].

3.1 Finite Difference Approximations

$$\frac{\partial P}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 P}{\partial S^2} + rS\frac{\partial P}{\partial S} - rP + \frac{\epsilon C}{P + \epsilon - q(s)} = 0 \tag{3.1.1}$$

The spatial derivatives are approximated by central differencing with spatial step size $h=\frac{S_f - S_0}{N}$ and time step size $k=\frac{T_f - T_0}{M}$ as follows. We apply the θ-method($\theta = 0$) [5] by performing a Backward-Euler approximation on $\frac{\partial P}{\partial t}$ and treat the penalty term Q explicitly.The discretization of the derivatives of the asset price S, is given as follows:

$$\frac{\partial P^2}{\partial S^2} = \frac{P_{m+1,n} - 2P_{m,n} + P_{m-1,n}}{h^2},$$

$$\frac{\partial P}{\partial S} = \frac{P_{m+1,n} - P_{m-1,n}}{2h}.$$

Using the approximations above , the Black-Scholes partial differential equation is then discretized at mesh points (mh, nk) ,$m = 1, 2...N$,at the time level $t = nk$,$n = 1, 2, ..., M$.At each n we obtain

$$\frac{P_{m,n} - P_{m,n-1}}{k} = \frac{1}{2}\sigma^2 S^2 \left[\frac{P_{m+1,n} - 2P_{m,n} + P_{m-1,n}}{h^2}\right] + rs\left[\frac{P_{m+1,n} - P_{m-1,n}}{2h}\right] - rP_{m,n} + \frac{\epsilon C}{P + \epsilon - q(s)}, \tag{3.1.2}$$

$$P_{m,n} - P_{m,n-1} = \frac{1}{2}\frac{\sigma^2 S^2 k}{h^2}\left[P_{m+1,n} - 2P_{m,n} + P_{m-1,n}\right] + \frac{krS}{2h}\left[P_{m+1,n} - P_{m-1,n}\right] - rkP_{m,n} + \frac{\epsilon C k}{P + \epsilon - q(s)}. \tag{3.1.3}$$

Let $\beta = \frac{1}{2}\frac{\sigma^2 S^2 k}{h^2}$, $\alpha = \frac{krS}{2h}$ and $Q = \frac{\epsilon C k}{P+\epsilon-q(s)}$

By substituting α and β in equation (3.1.3), we obtain,

$$P_{m,n} - P_{m,n-1} = \beta P_{m+1,n} - 2\beta P_{m,n} + \beta P_{m-1,n} + \alpha P_{m+1,n} - \alpha P_{m-1,n} - rkp_{m,n} + Q$$

$$P_{m,n} + 2\beta P_{m,n} + rkP_{m,n} - \beta P_{m+1,n} - \alpha P_{m-1,n} - \beta P_{m-1,n} = P_{m,n-1} + Q$$

$$(1 + 2\beta + rk)P_{m,n} - (\alpha + \beta)P_{m+1,n} + (\alpha - \beta)P_{m-1,n} = P_{m,n-1} + Q \tag{3.1.4}$$

This leads to the following tridiagonal system

$$A = \begin{pmatrix} 1+2\beta+rk & -(\alpha+\beta) & \dots & 0 \\ \alpha-\beta & 1+2\beta+rk & & \\ & \ddots & \ddots & \\ & & & \\ 0 & \dots\dots & \alpha-\beta & 1+2\beta+rk-(\alpha+\beta) \\ 0 & \dots & \alpha-\beta & 1+2\beta+rk \end{pmatrix}.$$

The boundary condition vector $\mathbf{b}_n$ resulting from writing (3.1.4) as a tridiagonal matrix system is given by

$$\mathbf{b}_n = [(\alpha-\beta)P_1, 0, \dots\dots 0, -(\alpha+\beta)P_{N,}]^T$$

$$AV_{m,n} + b_n = V_{m,n-1} + Q$$

The eigenvalues of the matrix A can be shown to be

$$\lambda_s = 1 + 2\beta + rk + 2\sqrt{-(\alpha+\beta)(\alpha-\beta)}\cos\left(\frac{S\pi}{N+1}\right) \quad S = 1, \cdots, N$$

$$\lambda_s = 1 + 2\beta + rk + 2\sqrt{\beta^2 - \alpha^2}\cos\left(\frac{S\pi}{N+1}\right)$$

If $0 < \alpha \leq \beta$ then the eigenvalues are real numbers satisfying

$$-(1 + 2\beta + rk + 2\sqrt{\beta^2 - \alpha^2}) \leq \lambda_s \leq 1 + 2\beta + rk + 2\sqrt{\beta^2 - \alpha^2}$$

If $\beta \leq \alpha$,then the eigenvalues are complex numbers satisfying

$$\begin{aligned} \lambda_s &= 1 + 2\beta + rk + 2\sqrt{-(-\beta^2 - \alpha^2)} \\ &= 1 + 2\beta + rk + 2\sqrt{\alpha^2 - \beta^2}i \\ &= 1 + 2\beta + rk + 2i\gamma_j \end{aligned}$$

where

$$-\sqrt{\alpha^2 - \beta^2} \leq \gamma_j \leq \sqrt{\alpha^2 - \beta^2}.$$

The eigenvalues of $\mathbf{A}$ are essential in determining the stability of the numerical scheme above. The system is stable if $max|\lambda_s| \leq 1$ for $S = 1, ..., N$.

3.2 RBF Approximations

The application of radial basis functions(RBF'S) in solving partial differential equations begun in the early 1990's[9]. Meshfree RBF methods have considerable advantages over traditional grid based methods such as finite difference, finite elements or finite volumes by being highly accurate, free from mesh generation and geometrically flexible since it's technique is based on only a set of independent points. Meshfree radial basis function approximation approach to solving the Black-Scholes equations for both the European and American options has been proposed by several authors (see, e.g., [13, 14, 15, 8]).

The basic assumption underlying this technique is that the value P of the option at asset price S and time t can be expanded in the form

$$P(S,t) = \sum_{j=1}^{N} c_j(t)\phi(\|S - x_j\|), \tag{3.2.1}$$

where time and "space" have been separated. The approximation space which is made up of the span of the functions $\phi(\|\cdot - x_1\|), \ldots, \phi(\|\cdot - x_N\|)$ is determined by the radial basis function $\phi(\|.\|)$. The *centers* x_j form a "discretization" of the domain $0 \leq S \leq S_\infty$. Numerous radial functions which have been studied which include the Gaussians, multiquadrics, thin plate splines, compactly supported RBFs which. We consider Gaussians, multiquadrics and Inverse- Multiquadrics in our numerical experiments because of the existence of exponential convergence rates for both Gaussian and Multiquadric RBFs.

- Gaussian-RBF $\phi(\|S - x_j\|) = \exp^{\left(-\frac{\|S-x_j\|^2}{c^2}\right)}$
- Multiquadric-RBF $\phi(\|S - x_j\|) = \sqrt{c^2 + \|S - x_j\|}$
- Inverse-Multiquadric $\phi(\|S - x_j\|) = \frac{1}{\sqrt{c^2+\|S-x_j\|}}$

After choosing a particular type of radial function ϕ , an approximate solution to the PDE 3.1.1 is given after the time-dependent coefficients c_j have been evaluated. The solution is given in the entire domain, and its derivatives (the Greeks) can be found similarly. Similarly to the method of lines, this approach produces a system of ordinary differential equations for the coefficients c_j. The time -dependent coefficients can be found by inverting the matrix ϕ. Detailed description of our approach is given below. The two asset case of the American option problem is a natural extension in the radial basis function formulation approach and is given below

$$P(\boldsymbol{S}, t) = \sum_{j=1}^{N} c_j(t)\phi(\|\boldsymbol{S} - \boldsymbol{x_j}\|), \tag{3.2.2}$$

with centers $x_j \in \Omega$. The problem still acts like a one-dimensional problem because of the location of the norm inside ϕ.

The only theoretical restriction on the location of the centers is that they be distinct. This motivates our adoption of regular discretization in our numerical experiments

CHAPTER 4

DISCRETIZATION WITH RADIAL BASIS FUNCTIONS

4.1 Gaussian-RBF

We implement a numerical solution of the partial differential equation (3.1.1)

$$\frac{\partial P_\epsilon}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 P_\epsilon}{\partial S^2} + rS\frac{\partial P_\epsilon}{\partial S} - rP_\epsilon + \frac{\epsilon C}{P_\epsilon + \epsilon - q(S)} = 0.$$

Since we use a collocation approach we not only require an expression for the value of the option

$$P_\epsilon(S,t) = \sum_{j=1}^{N} c_j(t)\phi(\|S - x_j\|), \tag{4.1.1}$$

but also for the partial derivatives present in (3.1.1). Thus, by differentiating (3.2.2),

$$\begin{aligned}
\frac{\partial P_\epsilon}{\partial t} &= \sum_{j=1}^{N} \dot{c}_j(t)\phi(\|S - x_j\|), \\
\frac{\partial P_\epsilon}{\partial S} &= \sum_{j=1}^{N} c_j(t)\phi'(\|S - x_j\|), \\
\frac{\partial^2 P_\epsilon}{\partial S^2} &= \sum_{j=1}^{N} c_j(t)\phi''(\|S - x_j\|),
\end{aligned}$$

where $\dot{}$ denotes a derivative with respect to t, and primes $'$ denote the derivatives with respect to S. In the specific case of Gaussian radial basis functions we have

$$\begin{aligned}
\phi'(\|S - x_j\|) &= \frac{-2(S - x_j)}{c^2}\exp\left(\frac{-\|S - x_j\|^2}{c^2}\right), \\
\phi'(\|S - x_j\|) &= \frac{-4(S - x_j)}{c^4}\exp\left(\frac{-\|S - x_j\|^2}{c^2}\right).
\end{aligned}$$

Inserting these expansions into (3.1.1) yields

$$\sum_{j=1}^{N} \dot{c}_j(t)\phi(\|S - x_j\|) + \frac{1}{2}\sigma^2 S^2 \sum_{j=1}^{N} c_j(t)\phi''(\|S - x_j\|) + rS\sum_{j=1}^{N} c_j(t)\phi'(\|S - x_j\|)$$

$$-r\sum_{j=1}^{N} c_j(t)\phi(\|S-x_j\|) + \frac{\epsilon C}{\sum_{j=1}^{N} c_j(t)\phi(\|S-x_j\|+\epsilon - q(S)} = 0.$$

At this point, we collocate at the points x_i, $i = 1, \ldots, N$, producing a discretization of the spatial part of the partial differential equation. This leads to a system of (nonlinear) ODEs for the coefficients c_j (collected in the vector $\mathbf{c}$)

$$\Phi\dot{\mathbf{c}} + R\mathbf{c} + Q_{\mathbf{c}} = 0.$$

Here

$$R = \frac{1}{2}\sigma^2\Phi_S'' + r\Phi_S' - r\Phi,$$

the matrices Φ, Φ_S' and Φ_S'' are given by

$$\Phi_{ij} = \phi(\|x_i - x_j\|), \quad \Phi'_{S,ij} = x_i\phi'(\|x_i - x_j\|), \quad \Phi''_{S,ij} = x_i\phi''(\|x_i - x_j\|),$$

and the vector $Q_{\mathbf{c}}$ has components

$$\frac{\epsilon C}{\Phi_i\mathbf{c} + \epsilon - q(x_i)}, \quad i = 1, \ldots, N,$$

4.2 Multiquadratic -RBF

We implement a numerical solution of the partial differential equation (3.1.1)

$$\frac{\partial P_\epsilon}{\partial t} + \frac{1}{2}\sigma^2 S^2\frac{\partial^2 P_\epsilon}{\partial S^2} + rS\frac{\partial P_\epsilon}{\partial S} - rP_\epsilon + \frac{{}_\epsilon C}{P_\epsilon + \epsilon - q(S)} = 0.$$

Since we use a collocation approach we not only require an expression for the value of the option

$$P_\epsilon(S,t) = \sum_{j=1}^{N} c_j(t)\phi(\|S-x_j\|), \tag{4.2.1}$$

but also for the partial derivatives present in (3.1.1). Thus, by differentiating (3.1.1),

$$\begin{aligned}\frac{\partial P_\epsilon}{\partial t} &= \sum_{j=1}^{N} \dot{c}_j(t)\phi(\|S - x_j\|),\\ \frac{\partial P_\epsilon}{\partial S} &= \sum_{j=1}^{N} c_j(t)\phi'(\|S - x_j\|),\\ \frac{\partial^2 P_\epsilon}{\partial S^2} &= \sum_{j=1}^{N} c_j(t)\phi''(\|S - x_j\|),\end{aligned}$$

where $\dot{}$ denotes a derivative with respect to t, and primes , $'$ denote derivatives with respect to S but also for the partial derivatives present in (3.1.1). Thus, by differentiating (4.2.1),

$$\begin{aligned}\phi(\|S - x_j\|) &= \sqrt{(s - x_j) + c^2}\\ \phi'(\|S - x_j\|) &= \frac{s - x_j}{\sqrt{(s - x_j) + c^2}}\\ \phi''(\|S - x_j\|) &= \frac{1}{\sqrt{(s - x_j) + c^2}} - \frac{(s - x_j)^2}{\sqrt{((s - x_j)^2 + c^2)^{\frac{3}{2}}}}\end{aligned}$$

The Multiquadratic RBF with shape parameter $c = 1 * d_{min}$,where d_{min} is the minimum distance between any collocation points s_j and $\dot{}$ denotes a derivative with respect to t, and primes denote derivatives with respect to S.

Inserting these expansions into (3.1.1) results in

$$\sum_{j=1}^{N} \dot{c}_j(t)\phi(\|S - x_j\|) + \frac{1}{2}\sigma^2 S^2 \sum_{j=1}^{N} c_j(t)\phi''(\|S - x_j\|) + rS\sum_{j=1}^{N} c_j(t)\phi'(\|S - x_j\|)$$
$$-r\sum_{j=1}^{N} c_j(t)\phi(\|S - x_j\|) + \frac{\epsilon C}{\sum_{j=1}^{N} c_j(t)\phi(\|S - x_j\|) + \epsilon - q(S)}$$

At this point, we collocate at the points x_i, $i = 1, \ldots, N$, forming a discretization of the spatial part of the partial differential equation. This leads to a system of (nonlinear) ODEs for the coefficients c_j (collected in the vector $\mathbf{c}$)

$$\Phi\dot{\mathbf{c}} + R\mathbf{c} + Q_{\mathbf{c}} = 0.$$

Here

$$R = \frac{1}{2}\sigma^2\Phi''_S + r\Phi'_S - r\Phi,$$

the matrices Φ, Φ'_S and Φ''_S are given by

$$\Phi_{ij} = \phi(\|x_i - x_j\|), \quad \Phi'_{S,ij} = x_i\phi'(\|x_i - x_j\|), \quad \Phi''_{S,ij} = x_i\phi''(\|x_i - x_j\|),$$

and the vector $Q_{\mathbf{c}}$ has components

$$Q_{\mathbf{c},i} = \frac{\epsilon C}{\Phi_i\mathbf{c} + \epsilon - q(x_i)} \quad i = 1, \ldots, N,$$

with Φ_i denoting the i-th row of the matrix Φ.

4.3 Inverse-Multiquadratic -RBF

We implement a numerical solution of the partial differential equation (3.1.1)

$$\frac{\partial P_\epsilon}{\partial t} + \frac{1}{2}\sigma^2 S^2\frac{\partial^2 P_\epsilon}{\partial S^2} + rS\frac{\partial P_\epsilon}{\partial S} - rP_\epsilon + \frac{{}_\epsilon C}{P_\epsilon + \epsilon - q(S)} = 0$$

Since we use a collocation approach we not only require an expression for the value of the option

$$P_\epsilon(S,t) = \sum_{j=1}^{N} c_j(t)\phi(\|S - x_j\|), \tag{4.3.1}$$

but also for the partial derivatives present in (4.3.1). Thus, by differentiating (4.3.1),

$$\begin{aligned}
\frac{\partial P_\epsilon}{\partial t} &= \sum_{j=1}^{N} \dot{c}_j(t)\phi(\|S - x_j\|), \\
\frac{\partial P_\epsilon}{\partial S} &= \sum_{j=1}^{N} c_j(t)\phi'(\|S - x_j\|), \\
\frac{\partial^2 P_\epsilon}{\partial S^2} &= \sum_{j=1}^{N} c_j(t)\phi''(\|S - x_j\|),
\end{aligned}$$

where $\dot{}$ denotes a derivative with respect to t, and primes denote derivatives with respect to S. In the specific case of the Inverse Multiquadric radial basis functions we have[8]

$$\phi(\|S - x_j\|) = \frac{1}{\sqrt{c^2 + \|S - x_j\|}}$$
$$\phi'(\|S - x_j\|) = \frac{S - x_j}{(c^2 + \|S - x_j\|^2)^{\frac{3}{2}}}$$
$$\phi''(\|S - x_j\|) = \frac{-(2S^2 + 4Sx_j - 2x_j^2 + c^2)}{(c^2 + \|S - x_j\|^2)^{\frac{5}{2}}}$$

Inserting these expansions into (3.1.1) yields

$$\sum_{j=1}^{N} \dot{c}_j(t)\phi(\|S - x_j\|) + \frac{1}{2}\sigma^2 S^2 \sum_{j=1}^{N} c_j(t)\phi''(\|S - x_j\|) + rS\sum_{j=1}^{N} c_j(t)\phi'(\|S - x_j\|)$$
$$-r\sum_{j=1}^{N} c_j(t)\phi(\|S - x_j\|) + \frac{\epsilon C}{\sum_{j=1}^{N} c_j(t)\phi(\|S - x_j\|) + \epsilon - q(S)} = 0.$$

At this point, we collocate at the points x_i, $i = 1, \ldots, N$, forming a discretization of the spatial part of the partial differential equation. This leads to a system of (nonlinear) ODEs for the coefficients c_j (collected in the vector $\mathbf{c}$)

$$\Phi\dot{\mathbf{c}} + R\mathbf{c} + Q_{\mathbf{c}} = 0. \tag{4.3.2}$$

Here

$$R = \frac{1}{2}\sigma^2\Phi''_S + r\Phi'_S - r\Phi,$$

the matrices Φ, Φ'_S and Φ''_S are given by

$$\Phi_{ij} = \phi(\|x_i - x_j\|), \quad \Phi'_{S,ij} = x_i\phi'(\|x_i - x_j\|), \quad \Phi''_{S,ij} = x_i\phi''(\|x_i - x_j\|),$$

and the vector $Q_{\mathbf{c}}$ has components

$$Q_{\mathbf{c},i} = \frac{\epsilon C}{\Phi_i\mathbf{c} + \epsilon - q(x_i)}, \quad i = 1, \ldots, N,$$

with Φ_i denoting the i-th row of the matrix Φ.

4.4 Time Stepping Procedure

Fasshauer *et al* [8] applied the θ-method for time stepping given as follows:

$$\Phi\frac{c^{n+1}-c^n}{k}+\theta Rc^{n+1}+(1-\theta)Rc^n+\theta Q(c^{n+1})+(1-\theta)Q(c^n)=0$$

where $c^n = c(nk)$. Fasshauer *et al* [8] developed a linearly implicit method approach to avoid solving nonlinear system of equations at each time step. They replaced c^n in the penalty term by c^{n+1} which is given as follows

$$\Phi\frac{c^{n+1}-c^n}{k}+\theta Rc^{n+1}+(1-\theta)Rc^n+Q(c^n)=0$$

or

$$[\Phi-(1-\theta)kR]c^n=[\Phi+\theta kR]c^{n+1}+kQ(c^{n+1}).$$

4.4.1 The Algorithm

We use the θ-method($\theta = 0$) in the above equation which leads to the linearly implicit Backward Euler method given as follows.

$$[\Phi-kR]c^n=[\Phi+kR]c^{n+1}+kQ^{n+1}. \tag{4.4.1}$$

The terminal condition serves as an initial condition for the ODE system. After collocation at the points x_i, $i = 1, \ldots, N$, the time dependant coefficients $c_j(T)$ are provided as the solution of the linear system below

$$\Phi c(T) = \mathbf{P},$$

where Φ is as above, and $\mathbf{P} = [P_\epsilon(x_1, T), \ldots, P_\epsilon(x_N, T)]^T$.

The boundary conditions are enforced at each time step by the addition of specific equations at each time step. This is necessary because radial basis functions do not satisfy their boundary conditions automatically.

An algorithm for the θ-method ($\theta = 0$) method is as follows[8]:

1. Choose a time step k .
2. Assemble the matrices Φ and R.
3. Compute the matrices $R_1 = \Phi - kR$ and $R_2 = \Phi + kR$.
4. Factor the matrices Φ and R_1.
5. Initialize the solution vector $\boldsymbol{P}$ via $P(x_i, T) = \max(E - x_i, 0)$, $i = 1, ..., N$.
6. For each time step
   (a) Update the coefficients by solving $\Phi c = P$.
   (b) Compute $b = R_2 c$ the vector Q_c
   (c) Find the next coefficients by solving the linear system $R_1 c = b + kQ_c$.
   (d) Update the solution vector $\boldsymbol{P}$ via $P(x_i, t) = \Phi c, i = 2, ..., N - 1$.
   (e) Enforce the boundary conditions $P(x_1, t) = E$ and $P(X_N, t) = 0$.

At each time step,we solve two linear systems.The matrix Φ is known to be invertible for any choice of (distinct) collocation points (=centers) x_i from the theory of radial basis functions interpolation.

Of course, this does not ensure satisfaction of the positivity constraint. However, the plots resulting from our numerical experiments indicate that this constraint is indeed satisfied for our choice of parameters.

4.5 Stability

Let error at the n^{th} time level be defined by

$$e^n = V^n_{exact} - V^n_{app} \tag{4.5.1}$$

Where V^n_{exact} and V^n_{app} is the exact solution and approximate solution obtained by the numerical process (2.1.1). and (2.1.3)

$$e^n = \mathbf{H}e^{n+1} \tag{4.5.2}$$

where $\mathbf{H}$ is the amplification matrix which is given by

$$\mathbf{H} = \Phi^{-1}[\Phi + \theta k\mathbf{R}]^{-1}\Phi \tag{4.5.3}$$

The numerical scheme is stable if the spectral radius of $\mathbf{H}$, $\rho(\mathbf{H}) \leq 1$. We substitute the value of $\mathbf{H}$ into equation (4.5.2) to obtain

$$[\Phi - (1-\theta)kM]\Phi^{-1}e^n = [\Phi + \theta k\mathbf{R}]\Phi^{-1}e^{n+1} \tag{4.5.4}$$

This implies

$$[I - (1-\theta)kM)]e^n = [I + \theta kM]e^{n+1} \tag{4.5.5}$$

where $M = \mathbf{R}\Phi^{-1}$ and I is an $N \times N$ identity matrix. The Numerical scheme is stable if all eigenvalues of the matrix $[\Phi - (1-\theta)kM]^{-1}[I + \theta kM]$ are less than one.

$$\left|\frac{1-(1-\theta)k\lambda_M}{1+\theta k\lambda_M}\right| \leq 1 \tag{4.5.6}$$

where λ_M represents the eigenvalues of the matrix M. Submitting $\theta = 0$ in 4.5.6 it is seen that, the backward Euler scheme is unconditionally stable.

CHAPTER 5

NUMERICAL EXPERIMENTS AND RESULTS

We compare the results obtained from our RBF methods to those obtained earlier by Fasshauer *et al* [8] using very high order finite difference schemes.

The error introduced by the penalty term in the finite difference method is roughly of the order ϵ [8]. The effects of the penalty parameter ϵ were studied more extensively by Fasshauer *et al* [8]. The parameters used in the numerical experiments for our single asset American put problem are given in the table below. We compare the the efficiency and computational accuracy of the three different radial basis functions Gaussian,Multiquadric and the Inverse-Multiquadric that were used in our numerical experiments. We further compare the results obtained by finite difference approximation using the θ-method ($\theta = 0$) to that of our radial basis function approach.

Parameter	Value
Minimum Asset Price	$S_0 = 0$
Maximum Asset Price	$S_\infty = 2$
Number of asset data points	$N = 101$
Number of time steps	$M = 100$
Time-step size	$k = 0.01$
Initial Time	$t = 0$
Expiration Time	$T = 1(year)$
Exercise price	$E = 1$
Risk-free interest rate	$r = 0.1$
Volatility	$\sigma = 0.2$
Shape parameter for Gaussian RBF	$c = 1.5$
Shape parameter Multiquadric RBF	$c = 1.0$
Shape parameter Inverse-Multiquadric RBF	$c = 1.5$
Regularization parameter	$\epsilon = 0.01$

Table 1: The Parameters for The Numerical Experiments

S	FD 1001	RBF 41	RBF 81	RBF 101
0.6	0.4000037	4.0000792842314e-01	4.0001372743884e-01	4.0001401552277e-01
0.7	0.3001161	3.0021903867805e-01	3.0022445841840e-001	3.0022481733287e-01
0.8	0.2020397	2.0260268722895e-01	2.0262172199297e-01	2.0262374715816e-01
0.9	0.1169591	1.1762509746978e-01	1.1769271815686e-01	1.1770056533954e-01
1.0	0.0602833	5.9528498119338e-02	5.9633310317163e-02	5.9645627931200e-02
1.1	0.0293272	2.7391459512703e-02	2.7486157537236e-02	2.7497344648467e-02
1.2	0.0140864	1.2108983580388e-02	1.2169988040340e-02	1.2177229864704e-02
1.3	0.0070408	5.6200254020830e-03	5.6508993770802e-03	5.6546031360821e-03
1.4	0.0038609	3.0344895112096e-03	3.0478633179663e-03	3.0495530284302e-03
RMSE		3.8005310000283e-04	3.6794723221557e-04	3.6654139838875e-04
CPU time		7.8000499999998e-02	3.2760210000000e-01	9.5160609999999e-01
COND number		8.5407111409489e+03	9.3715171581300e+03	9.4771870782724e+03

Table 2: Values of American option at $t = 0$ using Gaussian-RBF with $c = 1.5$.

S	FD 1001	RBF 41	RBF 81	RBF 101
0.6	0.4000037	4.0001360259473e-01	4.0001435446753e-01	4.0001446911887e-01
0.7	0.3001161	3.0022818611416e-01	3.0023280545987e-01	3.0023339283174e-01
0.8	0.2020397	2.0269906984195e-01	2.0271180531493e-01	2.0271322414513e-01
0.9	0.1169591	1.1800540492925e-01	1.1803788738272e-01	1.1804150249713e-01
1.0	0.0602833	6.0033601400926e-02	6.0118393221061e-02	6.0128083973080e-02
1.1	0.0293272	2.7773782298606e-02	2.7878195078764e-02	2.7891299637203e-02
1.2	0.0140864	1.2305166514669e-02	1.2402537546842e-02	1.2418247830917e-02
1.3	0.0070408	5.6826007102331e-03	5.7904633543829e-03	5.8172497012335e-03
1.4	0.0038609	3.0197673096413e-03	3.2041007551050e-03	3.2636931458391e-03
RMSE		3.4681339736604e-04	3.2557964828659e-04	3.2129115307467e-04
CPU time		1.4040090000000e-01	2.6520170000000e-01	5.4600349999999e-01
COND number		1.3502977553524e+04	5.2768316669994e+04	8.2049555126791e+04

Table 3: Values of American option at $t = 0$ using Multiquadric-RBF with $c = 1.0$.

S	FD 1001	RBF 41	RBF 81	RBF 101
0.6	0.4000037	4.0000818467312e-01	3.9999446772512e-01	3.9998485970236e-01
0.7	0.3001161	3.0021943862441e-01	3.0021108336036e-01	3.0020393403501e-01
0.8	0.2020397	2.0261104277165e-01	2.0262543924154e-01	2.0262455307839e-01
0.9	0.1169591	1.1766194094992e-01	1.1774648197568e-01	1.1776509832035e-01
1.0	0.0602833	5.9585357318176e-02	5.9739802425052e-02	5.9781503619760e-02
1.1	0.0293272	2.7451845834621e-02	2.7618763228463e-02	2.7672048190991e-02
1.2	0.0140864	1.2166938810012e-02	1.2317814004681e-02	1.2376673477242e-02
1.3	0.0070408	5.6804323723393e-03	5.8358171908381e-03	5.9090431725781e-03
1.4	0.0038609	3.1102998051219e-03	3.3324656560929e-03	3.4470898426795e-03
RMSE		3.6739869981565e-04	3.3417369206080e-04	3.2138512825344e-04
CPU time		2.8080179999999e-01	1.0608068000000	1.3884088999999
COND number		1.5116714978881e+03	1.8902391120150e+03	2.0114698011374e+03

Table 4: Values of American option at $t = 0$ using Inverse-Multiquadric-RBF with $c = 1.5$.

c	RMSE	CPU Time
0.5	6.180467527014752e-002	1.404009000000031e-001
0.7	5.943695978989005e-002	1.248008000000027e-001
0.9	1.048115147988156e-003	2.028012999999902e-001
1.1	3.763686518227007e-004	9.360060000000203e-002
1.3	3.799290451810216e-004	1.404009000000031e-001
1.5	3.799938827710493e-004	1.092007000000024e-001
1.7	3.799855151911536e-004	9.360060000000203e-002
1.9	3.800289462433457e-004	9.360059999997361e-002
2.1	3.800742178954350e-004	1.716011000000037e-001
2.3	3.801378742389429e-004	9.360060000000203e-002
2.5	3.803864448512753e-004	1.092007000000024e-001
2.7	3.813691028102942e-004	9.360060000000203e-002
2.9	3.849603077913177e-004	1.092007000000024e-001
3.1	3.978068233174443e-004	1.560010000000034e-001
3.3	4.454215456274659e-004	1.092007000000024e-001
3.5	6.300820110166411e-004	1.560010000000034e-001

Table 5: Comparison of different c values for Gaussian-RBF and their RMSE .

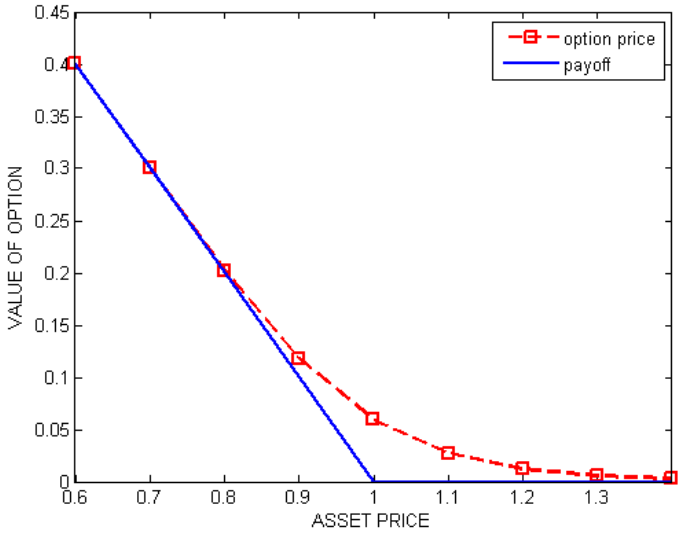

Figure 1: Graph of Gaussian-RBF at $N = 101$ nodes

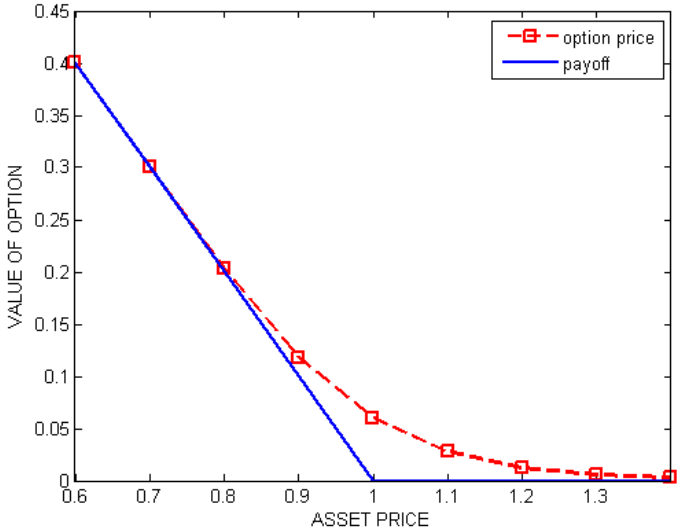

Figure 2: Graph of Multiquadric-RBF at $N = 101$ nodes

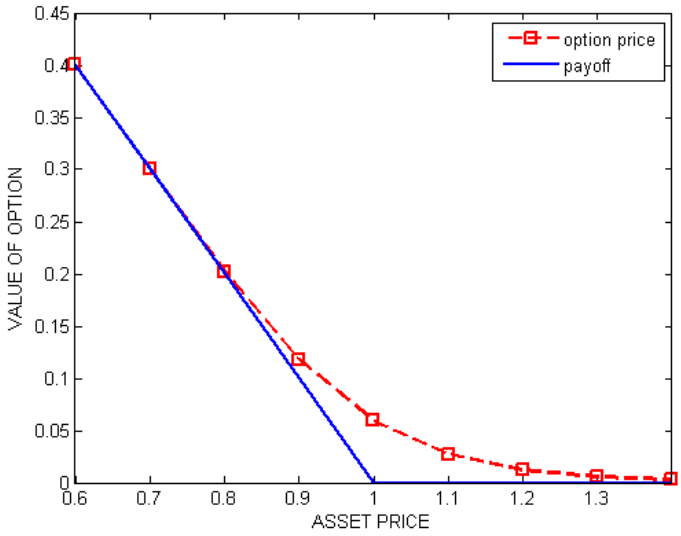

Figure 3: Graph of Inverse Multiquadric-RBF at $N = 101$ nodes $N = 101$

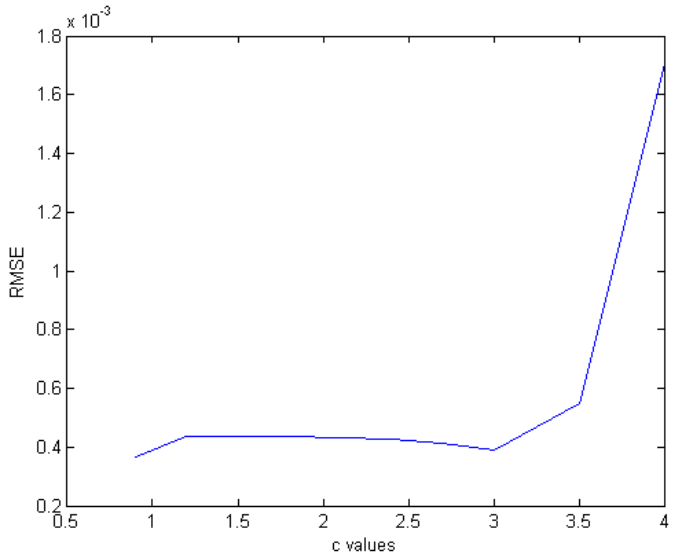

Figure 4: Comparison of RMSE for different c values using MQ-RBF at $N = 101$

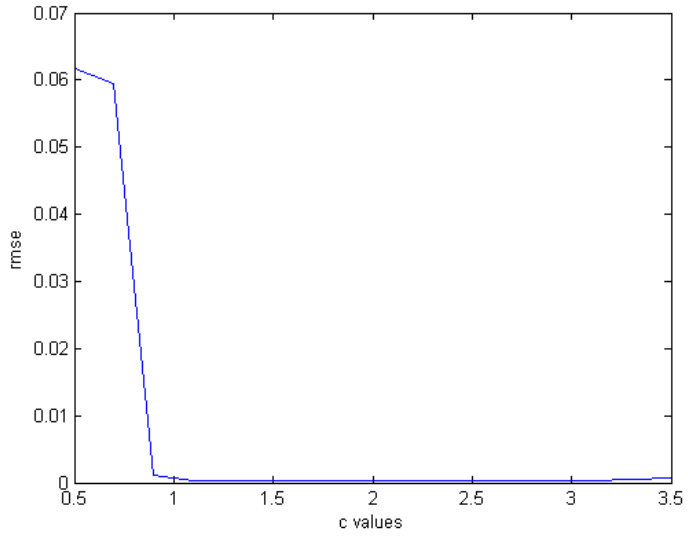

Figure 5: Comparison of RMSE for Different c values for Gaussian-RBF at $N = 101$ nodes

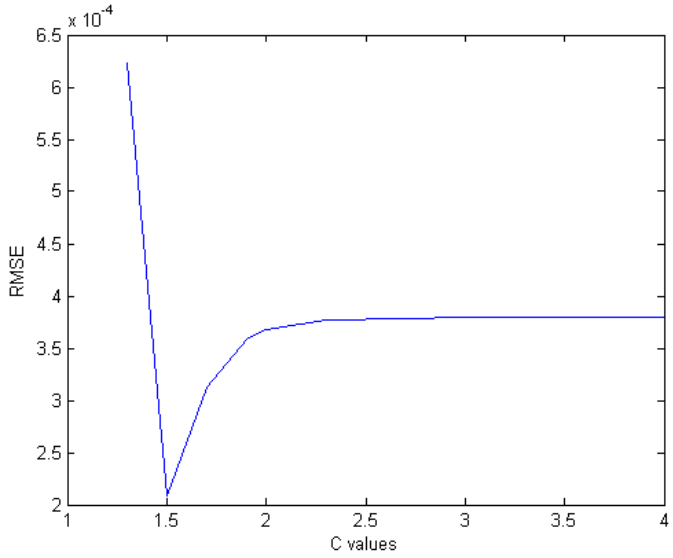

Figure 6: Comparison of RMSE for Different c values for IMQ-RBF at $N = 101$ nodes

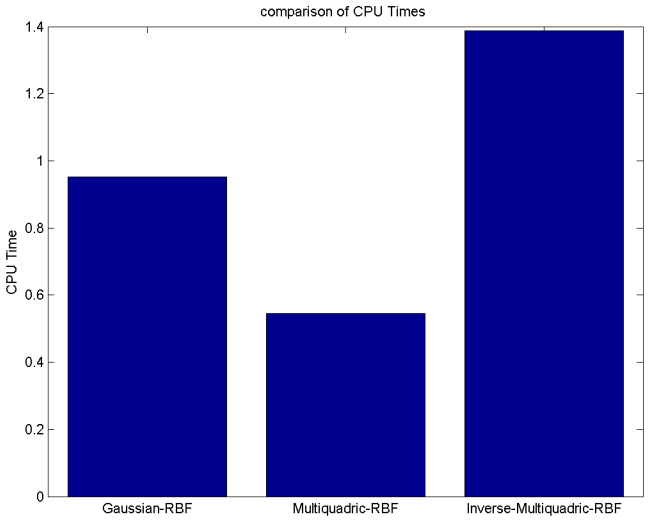

Figure 7: Comparison of CPU times of the RBF's at $N = 101$ nodes

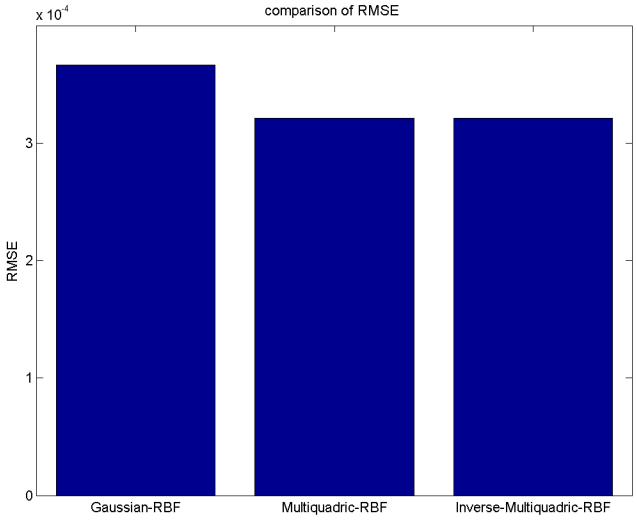

Figure 8: Comparison of RMSE for RBF's at $N = 101$ nodes

c	RMSE	CPU Time
0.2	8.206261648864639e-00	1.232407899999998e+000
0.4	2.148128533777387e-003	1.185607599999997e+000
0.8	3.060974866471791e-004	1.060806799999995e+000
1.0	3.212911530746730e-004	8.736055999999906e-001
1.3	3.466796105517317e-004	8.736056000000190e-001
1.5	3.516081867974326e-004	8.268052999999895e-001
1.8	3.542657007260861e-004	9.984063999999933e-001
2.1	3.554989239753822e-004	1.294808300000000e+000
2.5	3.566160605759570e-004	1.123207199999996e+000
3.0	3.576356069306907e-004	8.892056999999909e-001
3.5	3.583680279546284e-004	9.984063999999933e-001
4.0	3.588897897535030e-004	7.644048999999882e-001

Table 6: Comparison of different c values for MQ-RBF and their RMSE .

c	RMSE	CPU Time
0.5	4.951360842189655e-002	6.708043000000004e-001
0.7	1.532664011401649e-002	4.992032000000002e-001
0.9	7.794887198969099e-003	5.148032999999970e-001
1.1	2.578845380361731e-003	7.020045000000010e-001
1.3	6.232380475217872e-004	3.744024000000010e-001
1.5	2.089107130800181e-004	4.212027000000020e-001
1.7	3.126186527403213e-004	4.836031000000034e-001
1.9	3.582142527202816e-004	3.432022000000004e-001
2.0	3.800742178954350e-004	5.460034999999976e-001
2.3	3.673986998156583e-004	4.992031999999966e-001
2.7	3.768248122380777e-004	6.240039999999993e-001
3.0	3.787147145926920e-004	5.148032999999970e-001
3.5	3.790322985167134e-004	5.148032999999970e-001
4.0	3.793193743515548e-004	2.340015000000051e-001

Table 7: Comparison of different c values for IMQ-RBF and their RMSE .

S	Option Value	FD1001
0.6	0.4001500	0.4000037
0.7	0.3002437	0.3001161
0.8	0.2049304	0.2020397
0.9	0.1288876	0.1169591
1.0	0.0746978	0.0602833
1.1	0.0399911	0.0293272
1.2	0.0199422	0.0140864
1.3	0.0093622	0.0070408
1.4	0.0041843	0.0038609
CPUTIME	7.2540	
RMSE	0.0075	

Table 8: Finite Difference solution at $N = 2001, \epsilon = 10^{-4}, k = 0.001$.

S	Option Value	FD1001
0.6	0.4015118	0.4000037
0.7	0.3028086	0.3001161
0.8	0.2100498	0.2020397
0.9	0.1334426	0.1169591
1.0	0.0778183	0.0602833
1.1	0.0419259	0.0293272
1.2	0.0211123	0.0140864
1.3	0.0100855	0.0070408
1.4	0.0046513	0.0038609
CPUTIME	1.0920	
RMSE	0.0098	

Table 9: Finite Difference solution at $N = 2001, \epsilon = 10^{-3}, k = 0.01$.

S	Option Value	FD1001
0.6	0.4134769	0.4000037
0.7	0.3191524	0.3001161
0.8	0.2295821	0.2020397
0.9	0.1513232	0.1169591
1.0	0.0913208	0.0602833
1.1	0.0513920	0.0293272
1.2	0.0277877	0.0140864
1.3	0.0149302	0.0038609
1.4	0.0082378	0.0038609
CPUTIME	0.312002	
RMSE	0.0216	

Table 10: Finite Difference solution at $N = 2001, \epsilon = 10^{-2}, k = 0.1$.

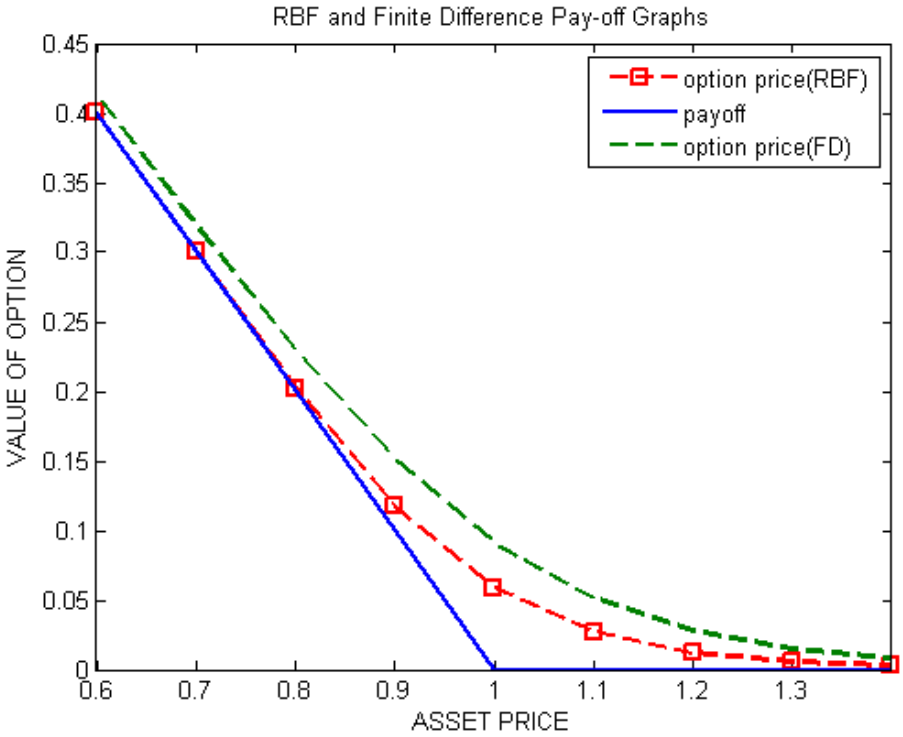

Figure 9: RBF N=101, k=0.01 and FD N=2001,k=0.01

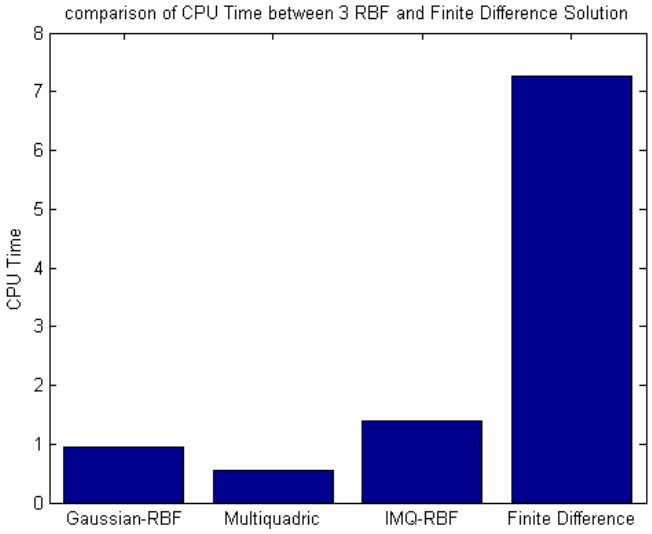

Figure 10: Comparison of CPU Times between the 3 RBFs for $N = 101$ and FD for N=2001

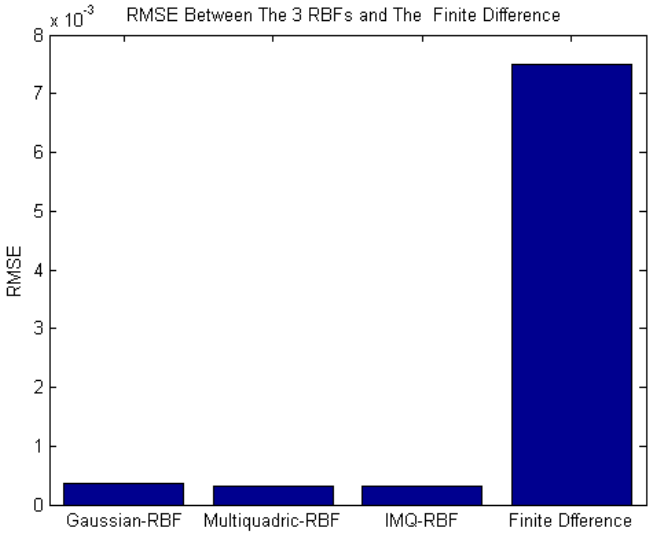

Figure 11: Comparison of RMSE between the 3 RBFs for $N = 101$ and FD for N=2001

CHAPTER 6

DISCUSSION OF NUMERICAL RESULTS AND CONCLUSION

6.1 Discussion of Numerical Results

Meshfree radial basis function interpolation exercise numerous advantages over traditional finite difference approximation schemes. RBFs depend on only the scalar distance between between two nodes at each point,this makes its discretization to be independent of coordinate system,dimension and domain geometry. With meshfree methods,it is possible to obtain the value of the option for different combination of option prices by evaluating the derivative of the basis function(3.2.1) as compared to finite difference which requires the inclusion of an extra interpolation step. RBFS are infinitely differentiable functions and as such produce highly accurate approximations to spatial derivatives whereas finite difference approximations to spatial derivatives are mostly second order accurate.

In Table 2,3 and 4,we evaluate a fair price of the American option using Gaussian,Multiquadric and Inverse-Multiquadric RBFs at different nodes of 41,81 and 101 with shape parameter $c = 1.5$. It was observed that the level of accuracy of the numerical results increased as the number of meshpoints was increased. The root mean squared error was computed as

$$RMSE = \frac{1}{\sqrt{N}}\sqrt{\sum_{i=1}^{N} |V(S_j,t)_{RBF} - V(S_j,t)_{FD1001})|^2}$$

where $V(S_j,t)_{RBF}$ and $V(S_j,t)_{FD1001})$ are the value of the option obtained using RBF meshfree approach and the bench mark solution in Fasshauer *et al* [8] respectively.

The condition number of the matrix Φ was computed as

$$Condition \quad number(\Phi) = \frac{max\lambda_i}{min\lambda_i} = \|\Phi\|\|\Phi^{-1}\|$$

where λ_i is an eigenvalue of the matrix Φ.

The choice of an optimal value for the shape parameter c is critical to the accuracy and stability of the numerical system. There is no precise or definite approach to choosing an optimal value for the shape parameter. Table 5,6 and 7 has several values of the shape parameter c with corresponding root mean squared error and CPU time at 101 nodes with parameters as listed in Table 1 for Gaussian,Multiquadric and Inverse-Multiquadric RBFs respectively. Figures $4, 5, 6, 7$ and 8 are based on these values. From figures 4,5 and 6,the optimal value of the shape parameter c for Gaussian RBF is 1.5,1.0 for for Multiquadric and 1.5 for Inverse-Multiquadric. The Multiquadric -RBF proved to be the most computationally eficient and accurate as depicted by figures 7 and 8.It has the lowest CPU time usage and root mean squared error among the three RBFs.

Figure $1, 2, 3$ represent the values of a fair price of the American put option obtained by Gaussian,Multiquadric and Inverse-Multiquadric RBFs at 101 nodes obtained from Table 1,2,and 3 respectively and their payoff function with parameters as listed in Table 1.

In Tables $8, 9$ and 10,we evaluate the value of the American put option by finite difference approximation approach at 2001 nodes. It is evident from our numerical results that the accuracy of the method greatly improves if the time step k is made smaller and smaller. The efficiency of the numerical scheme however deteriorates by decreasing the regularization parameter ϵ.

In figure 9,we compare the value of a fair price of the American option by Multiquadric at 101 nodes to that of finite difference approximation at 2001 nodes.

Figures 10 and 11 confirm that each one of the three RBF meshfree approach implemented in our numerical analysis was computationally efficient and accurate even at 101 nodes than a finite difference approximation obtained at 2001 nodes using the solution the solution obtained by Fasshauer *et al* [8] as our bench mark solution.

6.2 Conclusion

In this thesis,we evaluated a fair price of standard American options using both radial basis interpolation approach and finite difference approximation scheme. We implemented numerical solutions from three different RBFs namely the Gaussisn,Multiquadric and Inverse-Multiquadric RBFs. The free boundary in the American option problem was removed by the introduction of a nonlinear continuous penalty term. This allowed the problem to be solved on fixed domain. The numerical results obtained from our experiments shows that the Multiquadric-RBF is highly efficient and accurate in computing the value of the American option. This confirms its exponential convergence rate property. Our results suggest that each of the three RBFs implemented was considerably efficient and accurate even at a less number of nodes than the finite difference approximation scheme implemented.

BIBLIOGRAPHY

[1] A.Q.M. Khaliq, D.A. Voss and S.H.K. Kazmi, *A linearly implicit predictor-corrector scheme for pricing American options using a penalty method approach*, Journal of Banking and Finance,30 (2006) 489-502.

[2] A.Q.M. Khaliq, D.A. Voss and G. E.Fasshauer,*A Parallel Time Stepping Approach using Mesh-free Approximations for Pricing Options with Non-Smooth Payouts*,The Journal of Risk 10(2009) 135-142.

[3] B. Fornberg, E. Larsson, and N. Flyer, *Stable computations with Gaussian Radial Basis Functions*, SIAM Journal on Scientific Computing, 33 (2011), 869-892.

[4] B. Nielsen, O. Skavhaug, and A. Tveito, *Penalty and front-fixing methods for the numerical solution of American option problems*, Journal of Computational Finance, 5 (2002) 69-97.

[5] B. Nielsen, O. Skavhaug, and A. Tveito, *Penalty methods for the numerical solution of American multi-asset option problems*, Journal of Computational and Applied Mathematics, 222 (2008) 3-16.

[6] D. J. Higham , *An Introduction to Financial Option Valuation*,Cambridge Univeristy Press(2004).

[7] F. Black and M.S. Scholes, *The pricing of options and corporate liabilities*, Journal of Political Economy 81 (1973) 637-659.

[8] G.E. Fasshauer, A.Q.M. Khaliq and D.A. Voss, *Using meshfree approximation for multi-asset American option problems*, Journal of Chinese Institute of Engineers, 27 (2004) 563-571

[9] N. Flyer and B. Fornberg, *Radial basis function: Development and applications to planetary scale flows, Computers and Fluids* 46(2011) 23-32.

[10] P. Wilmot,S. Howson and J. Dewynne,*The Mathematics of Financial Derivatives*:A Student Introduction.Cambridge University Press(1995).

[11] S. A. Sarra and E. J. Kansa, *Multiquadric Radial Basis Function Approximation Methods for the Numerical Solution of Partial Differential Equations*. Advances in Computational Mechanics, Tech Series Press, 2 (2009).

[12] Y.C. Hon and X.Z. Mao,*A Radial Basis Function Method for Solving Options Pricing Models.*The Journal of Financial Engineering,8(1999) 31-49.

[13] Y. Goto, Z. Fei, S.Kan, and E. Kita, *Options valuation by using radial basis function approximation*, Engineering Analysis with Boundary Elements, 31(2007) 836-843.

[14] Y.C. Hon and Z. Yang, *Meshless collocation method by Delta-shaped basis functions for default barrier model*, Engineering Analysis with Boundary Elements 33(2009) 951-958.

[15] U. Pettersson, E. Larsson, G. Marcusson, and J. Persson, *Improved radial basis function methods for multi-dimensional option pricing*, Journal of Computational and Applied Mathematics, 222 (2008) 82-93.

APPENDICES

APPENDIX A

COMPUTER PROGRAMMING CODES

A.0.1 Tridiagonal solver

```
function vv=tridiagonal(aa,dd,cc,bb)
% Function to solve a Tridiagonal System of Equations:
nn=length(bb);
for i=2:nn
     xmult=aa(i)/dd(i-1);
    dd(i)=dd(i)-xmult*cc(i-1);
     bb(i)=bb(i)-xmult*bb(i-1);
end;
vv(nn)=bb(nn)/dd(nn);
for i=nn-1:-1:1
     vv(i)=(bb(i)-cc(i)*vv(i+1))/dd(i);
end
```

A.0.2 Theta method codes

```
function V=RBasis3GaussianAmerican(theta,~)

%Theta Methods(without matrix iniversion)for solving the ODE's

%clear all
format('longE')
```

```
global c

t=cputime;

r=0.1;sigma=0.20;T=1.;

S_min=0;S_max=2;

E=1;epsilon=0.01;C=r*E;theta=0.0;

N=21;

h=(S_max-S_min)/(N-1);

%c=1.5*h;
c=2.1*h;

Y=linspace(S_min,S_max,N);

M=100;

k=T/M;

% ************************************************

% Matrix P
```

```
L=L1(Y); L_Sy=L1_Sy(Y); L_Syy=L1_Syy(Y);

CND=cond(L)

P=(1/2)*sigma^2*(L_Syy)+r*(L_Sy)-r*L;

LL=L-(1-theta)*k*P;

PP=L+theta*k*P;

[LL1,UU1]=lu(LL);

[LL2,UU2]=lu(L);

% ****************************************

% Initial Condition

U0=ic(Y);

a=UU2\(LL2\U0);

% ****************************************

iplot=1;
```

```
60
for i=1:M

    b=PP*a;

65  for j=1:N

        Q(j)=epsilon*C/(L(j,:)*a+epsilon-(E-Y(j)));

    end

70

    an=UU1\(LL1\(b+k*Q'));

    Un=L*an;

    Un(1)=E;

75

    Un(N)=0;

    a=UU2\(LL2\Un);

80

%     if mod(i,10)==1
%
%         for j=1:N
%
%             W(iplot,j)=U(Y(j),a,Y);
```

```
%
%          end
%
%          iplot = iplot+1;
%
%      end

end

% surf(W);

colormap(hot);

SS=[0.6:0.1:1.4]
FD1001=[0.4000037 0.3001161 0.2020397 ...
    0.1169591 0.0602833 0.0293272 ...
    0.0140864 0.0070408 0.0038609];
for i=1:(length(SS))

   V(i)=U(SS(i),a,Y);

end;

temp=0;

for i=1:length(SS)
```

```
    temp=temp+(max(E-SS(i),0)-V(i))^2;
k(i)=max(E-SS(i),0);
end;

RMSE=sqrt(temp/length(SS))
error2=norm(FD1001-V)/(length(SS))
time=cputime-t

h=plot(SS,V,'--rs',SS,k);
xlabel('ASSET PRICE')
ylabel('VALUE OF OPTION')
legend('option price','payoff')
set(h,'LineWidth',2)
%title('Graph of Gaussian-RBF at 101 Nodes')
% ****************************************

function y=U(x,a,Y)

N=length(Y);

y=0;

for i=1:N

    y=y+a(i)*phi(x,Y(i));
```

```
end;

% ************************************************

function y=phi(x,yj)

global c

%c=0.05;

  %y=sqrt((x-yj)^2+c^2);

  y=exp(-(x-yj)^2/c^2);
%y=(c^2 +(x-yj)^2)^(-1/2);

% ************************************************

function y=phi_y(x,yj)
```

```
165 global c

%c=0.05;

  %y=(x-yj)/(sqrt((x-yj)^2+c^2));

170

  y=(-2*(x-yj)/c^2)*exp(-(x-yj)^2/c^2);
%y=(x-yj)/(c^2 +(x-yj)^2)^(3/2);



175 % ********************************************



function y= phi_yy(x,yj)

180

global c

%c=0.05;

185 %y=1/sqrt((x-yj)^2+c^2)- ((x-yj)^2)/(((x-yj)^2+c^2)^(3/2));

y=-2/c^2*exp(-(x-yj)^2/c^2)+4*(x-yj)^2/c^4*exp(-(x-yj)^2/c^2);
%y=-(-2*x^2 +4*x*yj-2*yj^2 +c^2)/(c^2 +(x-yj)^2)^(5/2);
```

```
% ***************************************************

function y=L1(Y)

N=length(Y);

for i=1:N

    for j=1:N

        y(i,j)=phi(Y(i),Y(j));

    end

end

% ***************************************************

function y=L1_Sy(Y)

```

```
N=length(Y);

for i=1:N

    for j=1:N

        y(i,j)=Y(i)*phi_y(Y(i),Y(j));

    end

end



% ********************************************************



function y=L1_Syy(Y)

N=length(Y);

for i=1:N

    for j=1:N
```

```
        y(i,j)=Y(i)*phi_yy(Y(i),Y(j));

    end

end

% ************************************************

function y=ic(Y)

E=1;

N=length(Y);

y=zeros(N,1);

for i=1:N

    y(i)=max(E-Y(i),0);

end
```

```
% ********************************************************
```

```
function V=RBasis3GaussianAmerican(theta,M)

%Theta Methods(without matrix iniversion)for solving the ODE's

%clear all
format('longE')
global c

t=cputime;

r=0.1;sigma=0.20;T=1.;

S_min=0;S_max=2;

E=1;epsilon=0.01;C=r*E;theta=0.5;

N=101;

h=(S_max-S_min)/(N-1);

%c=4.0*h;
c=1*h;
```

```
Y=linspace(S_min,S_max,N);

M=100;

k=T/M;

% ****************************************

% Matrix P

L=L1(Y); L_Sy=L1_Sy(Y); L_Syy=L1_Syy(Y);

CND=cond(L)

P=(1/2)*sigma^2*(L_Syy)+r*(L_Sy)-r*L;

LL=L-(1-theta)*k*P;

PP=L+theta*k*P;

[LL1,UU1]=lu(LL);

[LL2,UU2]=lu(L);

% ****************************************
```

```
50
% Initial Condition

U0=ic(Y);

55 a=UU2\(LL2\U0);

% ****************************************

iplot=1;

60
for i=1:M

    b=PP*a;

65     for j=1:N

        Q(j)=epsilon*C/(L(j,:)*a+epsilon-(E-Y(j)));

    end

70
    an=UU1\(LL1\(b+k*Q'));

    Un=L*an;

75     Un(1)=E;
```

```
    Un(N)=0;

    a=UU2\(LL2\Un);

%      if mod(i,10)==1
%
%          for j=1:N
%
%              W(iplot,j)=U(Y(j),a,Y);
%
%          end
%
%          iplot = iplot+1;
%
%      end

end

% surf(W);

colormap(hot);

SS=[0.6:0.1:1.4];
FD1001=[0.4000037 0.3001161 0.2020397 ...
    0.1169591 0.0602833 0.0293272 ...
```

```
    0.0140864  0.0070408  0.0038609];
for i=1:(length(SS))

   V(i)=U(SS(i),a,Y);

end;

temp=0;

for i=1:length(SS)

   temp=temp+(max(E-SS(i),0)-V(i))^2;
k(i)=max(E-SS(i),0);
end;

RMSE=sqrt(temp/length(SS))
error2=norm(FD1001-V)/(length(SS))
time=cputime-t

h=plot(SS,V,'--rs',SS,k);
xlabel('ASSET PRICE')
ylabel('VALUE OF OPTION')
legend('option price','payoff')
set(h,'LineWidth',2)
%title('Graph of MQ-RBF at 101 Nodes')
% ************************************
```

```
function y=U(x,a,Y)

N=length(Y);

y=0;

for i=1:N

   y=y+a(i)*phi(x,Y(i));

end;



% ************************************



function y=phi(x,yj)

global c

%c=0.05;

  y=sqrt((x-yj)^2+c^2);
```

```
 % y=exp(-(x-yj)^2/c^2);
%y=(c^2 +(x-yj)^2)^(-1/2);



% ******************************************



function y=phi_y(x,yj)

global c

%c=0.05;

  y=(x-yj)/(sqrt((x-yj)^2+c^2));

 % y=-2*(x-yj)/c^2*exp(-(x-yj)^2/c^2);
%y=(x-yj)/(c^2 +(x-yj)^2)^(3/2);



% ****************************************



function y= phi_yy(x,yj)
```

```
180
global c

%c=0.05;

185 y=1/sqrt((x-yj)^2+c^2)- ((x-yj)^2)/(((x-yj)^2+c^2)^(3/2));

%y=-2/c^2*exp(-(x-yj)^2/c^2)+4*(x-yj)...
%^2/c^4*exp(-(x-yj)^2/c^2);
%y=-(-2*x^2 +4*x*yj-2*yj^2 +c^2)/(c^2 +(x-yj)^2)^(5/2);
190

% ****************************************

195 function y=L1(Y)

N=length(Y);

for i=1:N
200

   for j=1:N

      y(i,j)=phi(Y(i),Y(j));

205    end
```

```
end


% ******************************************


function y=L1_Sy(Y)
N=length(Y);
for i=1:N
    for j=1:N
        y(i,j)=Y(i)*phi_y(Y(i),Y(j));
    end
end


% ******************************************
```

```
%
235
function y=L1_Syy(Y)
N=length(Y);
240 for i=1:N
    for j=1:N
        y(i,j)=Y(i)*phi_yy(Y(i),Y(j));
245 end
end
250
% ****************************************
% **
255
function y=ic(Y)
```

```
E=1;

N=length(Y);

y=zeros(N,1);

for i=1:N

    y(i)=max(E-Y(i),0);

end



% ******************************************
```

```
function V=RBasis3GaussianAmerican(theta,M)

%Theta Methods(without matrix iniversion)for solving the ODE's

%clear all
format('longE')
global c

t=cputime;
```

```
r=0.1;sigma=0.20;T=1.;

S_min=0;S_max=2;

E=1;epsilon=0.01;C=r*E;theta=0.0;

N=21;

h=(S_max-S_min)/(N-1);

%c=1.5*h;
c=2.4*h;


Y=linspace(S_min,S_max,N);

M=100;

k=T/M;

% ************************************************

% Matrix P

L=L1(Y); L_Sy=L1_Sy(Y); L_Syy=L1_Syy(Y);
```

```
CND=cond(L)

P=(1/2)*sigma^2*(L_Syy)+r*(L_Sy)-r*L;

LL=L-(1-theta)*k*P;

PP=L+theta*k*P;

[LL1,UU1]=lu(LL);

[LL2,UU2]=lu(L);

% ****************************************

% Initial Condition

U0=ic(Y);

a=UU2\(LL2\U0);

% ****************************************

iplot=1;

for i=1:M
```

```
    b=PP*a;
65
    for j=1:N
        Q(j)=epsilon*C/(L(j,:)*a+epsilon-(E-Y(j)));
70  end
    an=UU1\(LL1\(b+k*Q'));
    Un=L*an;
75
    Un(1)=E;
    Un(N)=0;
80  a=UU2\(LL2\Un);
%     if mod(i,10)==1
%
%         for j=1:N
85  %
%             W(iplot,j)=U(Y(j),a,Y);
%
%         end
```

```
%
%          iplot = iplot+1;
%
%      end

end

% surf(W);

colormap(hot);

SS=[0.6:0.1:1.4];
FD1001=[0.4000037 0.3001161 0.2020397 ...
    0.1169591 0.0602833 0.0293272 ...
    0.0140864 0.0070408 0.0038609];
for i=1:(length(SS))

   V(i)=U(SS(i),a,Y);

end;

temp=0;

for i=1:length(SS)

   temp=temp+(max(E-SS(i),0)-V(i))^2;
```

```
k(i)=max(E-SS(i),0);
end;

RMSE=sqrt(temp/length(SS))
error2=norm(FD1001-V)/sqrt(length(SS))
%error2=sqrt((FD1001-V)^2)/sqrt(length(SS))
time=cputime-t

h=plot(SS,V,'--rs',SS,k);
xlabel('ASSET PRICE')
ylabel('VALUE OF OPTION')
legend('option price','payoff')
set(h,'LineWidth',2)
%title('Graph of Inverse MQ-RBF at 101 Nodes')
% ****************************************

function y=U(x,a,Y)

N=length(Y);

y=0;

for i=1:N

   y=y+a(i)*phi(x,Y(i));

```

```
end;



% ************************************************

function y=phi(x,yj)

global c

%c=0.05;

  %y=sqrt((x-yj)^2+c^2);

  %y=exp(-(x-yj)^2/c^2);
y=(c^2 +(x-yj)^2)^(-1/2);



% ************************************************

```

```
function y=phi_y(x,yj)

global c

%c=0.05;

  %y=(x-yj)/(sqrt((x-yj)^2+c^2));

  %y=-2*(x-yj)/c^2*exp(-(x-yj)^2/c^2);
%y=(x-yj)/(c^2 +(x-yj)^2)^(3/2);

y=-(1/2)*(2*x-2*yj)/((x-yj)^2+c^2)^(3/2);
% ****************************************


function y= phi_yy(x,yj)

global c

%c=0.05;

%y=1/sqrt((x-yj)^2+c^2)- ((x-yj)^2)/...
%(((x-yj)^2+c^2)^(3/2));
%y=-2/c^2*exp(-(x-yj)^2/c^2)+4*(x-yj)...
%^2/c^4*exp(-(x-yj)^2/c^2);
```

```
% y=-(-2*x^2 +4*x*yj-2*yj^2 +c^2)/...
%(c^2 +(x-yj)^2)^(5/2);
y=(3/4)*(2*x-2*yj)^2/((x-yj)^2+c^2)...
^(5/2)-1/((x-yj)^2+c^2)^(3/2);
% ************************************************



function y=L1(Y)

N=length(Y);

for i=1:N

   for j=1:N

      y(i,j)=phi(Y(i),Y(j));

   end

end



% ************************************************
```

```
220
function y=L1_Sy(Y)

N=length(Y);

225 for i=1:N

    for j=1:N

        y(i,j)=Y(i)*phi_y(Y(i),Y(j));

230
    end

end

235

% ******************************************

240
function y=L1_Syy(Y)

N=length(Y);
```

```
for i=1:N
    for j=1:N
        y(i,j)=Y(i)*phi_yy(Y(i),Y(j));
    end
end


% ******************************************


function y=ic(Y)
E=1;
N=length(Y);
y=zeros(N,1);
for i=1:N
```

```
    y(i)=max(E-Y(i),0);

end


% ****************************************
```