Numerical solutions for PDEs in Option pricing

Undergraduate Thesis

Submitted in partial fulfillment of the requirements of
BITS F421T Thesis

By

Jaskamal kainth
ID No. 2013B4A70586P

Under the supervision of:

Dr. Devendra Kumar

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI, PILANI CAMPUS
December 2017

Declaration of Authorship

I, Jaskamal KAINTH, declare that this Undergraduate Thesis titled, 'Numerical solutions for PDEs in Option pricing' and the work presented in it are my own. I confirm that:

- This work was exclusively done during candidature for a research degree at this University.
- Where any part of this thesis has formerly been submitted for a degree or this has been clearly stated that any part of this thesis has formerly been submitted.
- Where I have referred to the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always provided. exclusive of these quotations, this thesis is entirely my own work.
- I have acknowledged all the major sources of help.
- Where the thesis is the result of work done by myself, I have made clear exactly what was done by others and what I have done myself.

Signed: ________________________________

Date: ________________________________

Certificate

This is to certify that the thesis entitled, "*Numerical solutions for PDEs in Option pricing*" and submitted by Jaskamal KAINTH ID No. 2013B4A70586P in partial fulfillment of the requirements of BITS F421T Thesis embodies the work done by him under my supervision.

Supervisor
Dr. Devendra KUMAR
Asst. Professor,
BITS-Pilani Pilani Campus
Date:

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI, PILANI CAMPUS

Abstract

M.Sc. (Hons.) Mathematics

Numerical solutions for PDEs in Option pricing

by Jaskamal KAINTH

In this report, we start with the basic introduction to options and related terms which are used alongwith option pricing. After the basic introduction part, first numerical method "Cubic B-spline" is discussed. The aim of this thesis is to apply the B-spline collocation method to solve Black scholes equation used in option pricing. Firstly, this method is used to solve various boundary value problems. After the description of B-splines, various examples are solved using this method which shows the accuracy of this method. After working out with ODEs, we shift our focus towards solving general PDEs with cubic B-splines After applying Cubic b-spline to ODEs and PDEs , we shift towards our main goal to solve Black scholes equation. The methods accuracy and error analysis is done for the Asian options.

The second part of the thesis is the description of the Radial basis functions and the method to solve various PDE numerically. We have use Inverse quadratic as our radial basis function. After this ,the method is explained to solve various PDEs (linear and Non linear PDE). Various examples of Linear PDEs like Heat equation and Diffusion equation are numerically solved and results are shown. After this, Nonlinear Burger equation is solved for various shape parameters. In the end, we conclude with the discussion on the convergence and influence of the shape parameter on the accuracy of this RBF method.

Acknowledgements

Firstly, I would like to express my sincere gratitude to my advisor Prof. Devendra Kumar for his regular support of my thesis study and related research, for his time, guidance and immense knowledge in this area. I could not have imagined having a better advisor and mentor for my thesis study. Beside my advisor, i would also thank my M.Sc(Hons) Mathematics batch mates who helped me in correcting, implementing my code. Finally, I would like to thank Mathematics department at BITS Pilani, Pilani campus for allowing me to do this thesis study and evaluating this project report.

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

Options in layman terms is a contract that gives the right but not an obligation to either buy or sell an assest(stocks/shares of a company/ foreign currencies/ commodities) on or before a certain date.

In general,there are two types of options named **Call** and **Put**. When we buy a call option we wish that the option price goes up and when we buy a put option we wish that the option price goes down. So, In general we can buy an option and then sell it in future (hopefully with higher price) Or we can sell an option as an opening trade and hopefully buy it back at a lower price as a closing trade.

A **Call** Option is basically a well defined contract b/w two parties to agree to exchange the known stock at predefined ”strike” price by a predetermined date. The buyer of the ”call” (the first party) , has the right, but not an obligation, to buy the stock at the strike price by the predetermined future date, while the seller of the call(the second party), has the obligation to sell the stock to the buyer at the strike price if the buyer decides to exercise the option.

A **Put** Option is basically a well defined contract b/w two parties to exchange a stock at a ”strike” price, by a predetermine date. The buyer of the ”put” has the right , but not an obligation, to sell the stock at the strike price by the future date, while the seller of the put, has the obligation to buy the stock from the buyer at the strike price if the buyer decides to exercise the option.

Option prices are based on 3 elements:

- TIME TO EXPIRATION: - Options have expiration dates. Just like in insurance we have to pay periodically a fixed amount , similar case happens with options. More the time , more the price you have to pay. Time decay happens as we go closer to expiration. This graph is exponential i.e as we go near expiration , time decay speeds up and option time value reaches 0 at the expiration date. Most stocks have options with weekly, monthly, quaterly exprations. We can choose any one of these to trade in.

- UNDERLYING STOCK PRICE: for each stock we have multiple options at different price increments.These are called the option **strike price**. Strike price is the Pre-determined price at which the shares of the stock will be exchanged (buy/ sell) if the option is exercised. Options are worthless at the expiration. Also, we do not have to hold our position until expiration. Ex: If we buy a call option today and the stock price goes up overnight then we can sell it and make profit rather than waiting for 30days which is the expiration date.
- VOLATILITY: Magnitude of a Stock's price swings. Higher volatility means more risk for the stock owner. Since more risk, it will cost us more money.

  The combination of these three factors determines the option prices.

Basic shorting Borrow the stock from someone and sell it, then hope that its price falls you buy it again and give it back.

Call option as leverage Call option gives us financial leverage as Percent gain with call option is more than normal stock price. So with call option we multiply our potential gains/losses.

Payoff Diagrams Value - Underlying stock price.

- **Call payoff Diagram**

  $Value(S,T)$ is the value of Call option , where S is the current value of the stock with expiration date T is given by, When $StockPrice < StrikePrice$, then there is no need to exercise it since during this time we have negative profit which is our premium, else $Value = StockPrice - StrikePrice$ (linear line) , here profit is positive after StockPrice + Premium.

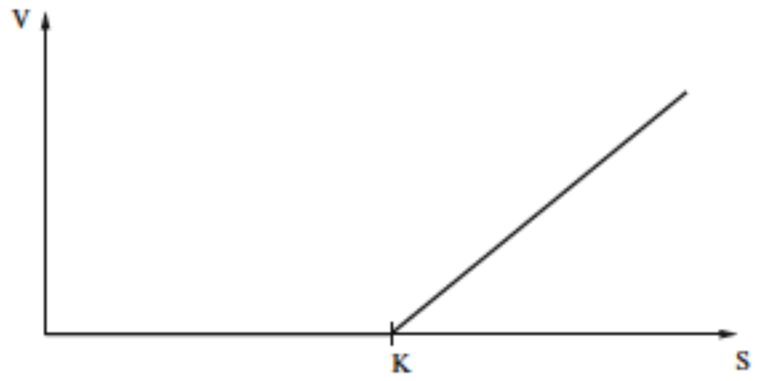

Figure 1.1: Value of a Call option with Strike Price (K)

- **Put payoff Diagram** [similar to call with minor changes].

  if underlying stock price is 0 then we will definitely exercise our put option and make profit of our strike price [y-intercept is strike price].Now above strike price we have right to not exercise our option therefore value is 0 and profit is negative premium.

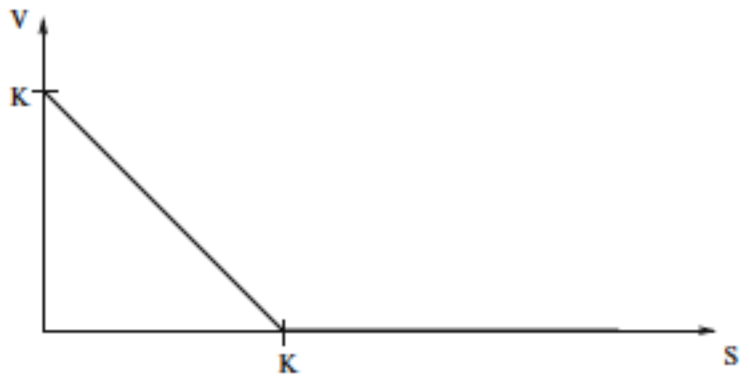

FIGURE 1.2: Value of a Put option with Strike Price (K)

The Popular **Black-Scholes model** is used to calculate the value the value (price) of an option. In this thesis, a highly accurate numerical method which solves this equation is explained and implemented.There are many types of options, like *European options*, *American options*, *Asian options* etc. The exact solution for the European option is known, but we will solve it numerically using our numerical methods. This is done to create a general Numerical method which solves this *Black-Scholes equation* for all type of options.

Chapter 2

Cubic B-Splines

2.1 Introduction

Spline is a function define by piecewise polynomials. Splines are used in Interpolating problems. The most common spline is the cubic spline of **degree 3**. Any spline function can be represented as a linear combination of B-Splines of that degree, where B-Spline is a shortform of Basis spline.

B-splines method can be used to solve one-dimensional heat or wave equations using discretization in the time space while the cubic B-spline is applied in the space dimension as an interpolation function.

Let $I = [a, b] \subset R$ be an interval. Let $p, q, r : [a, b] \to$ be continuous functions.
Throughout this report, we will consider the Linear 2_{nd} order Differential equation given by

$$y'' + p(x)y' + q(x)y = r(x), a \leq x \leq b \tag{2.1}$$

Where the Boundary conditions are, Dirichlet(First kind) : $y(a) = \eta_1, y(b) = \eta_2$.

The cubic B-Splines are used to approximate the exact solution of the equation 2.1.

Lets define the Cubic splines, Consider the Infinite Linear space $S_3(\pi)$, which contains all the functions $s(t)$ that can be reduced to cubic polynomials on sub-intervals (t_i, t_{i+1}) where $0 \leq i \leq n-1$ on the interval $[a, b]$.

There exists a unique function $s(t)$ in the space $S_3(\pi)$ such that it satisfies the following constraints:

$s'(t_0) = f'(t_0)$
$s(t_i) = f(t_i)$ where $0 \leq i \leq n-1$
$s'(t_n) = f'(t_n)$ with these, the piecewise spline function $s(t)$ interpolates to $f(t)$.

Now, To find this $s(t)$, we define the partition on the interval $[a, b]$ given by $x_i = x_0 + i * h$ where $0 \leq i \leq n-1$ where $h = (b-a)/n$ is the step difference.

This partition forms the collocation points on the interval. Along with this, we defined the piecewise functions by,

$$B_i(x) = \frac{1}{6h^3} \begin{cases} (x - x_{i-2})^3, \, x_{i-2} \leq x \leq x_{i-1}, \\ h^3 + 3h^2(x - x_{i-1}) + 3h(x - x_{i-1})^2 - 3(x - x_{i-1})^3, \, x_{i-1} \leq x \leq x_i, \\ h^3 + 3h^2(x_{i+1} - x) + 3h(x_{i+1} - x)^2 - 3(x_{i+1} - x)^3, \, x_i \leq x \leq x_{i+1}, \\ (x_{i+2} - x)^3, \, x_{i+1} \leq x \leq x_{i+2}, \\ 0, \text{otherwise}. \end{cases} \tag{2.2}$$

This $B_i(x)$ is piecewise cubic functions with the same collocation points and also it fulfills the property that they are twice continuously differentiable $\forall x \in R$

Thus, We can compute values of $B_i(x_i)$, $B_i'(x)$ and $B_i''(x)$ using above $B_i(x)$ and the tabulated values are shown in **table** 2.1.

	x_{i-2}	x_{i-1}	x_i	x_{i+1}	x_{i+2}
$B_i(x)$	0	$1/6$	$4/6$	$1/6$	0
$B_i'(x)$	0	$1/2h$	0	$-1/2h$	0
$B_i''(x)$	0	$1/h^2$	$-2/h^2$	$1/h^2$	0

Table 2.1: Table showing Spline function and its derivatives evaluated at collocation points

2.2 Application of Cubic B-Splines to solve BVP

Now, lets approximate the exact solution of the below equation using the theory discussed in the previous section.

$$y'' + p(x)y' + q(x)y = r(x), \qquad a \leq x \leq b \tag{2.3}$$

Where the Boundary conditions are,
Dirichlet(First kind) : $y(a) = \eta_1, y(b) = \eta_2$.

Over any sub-interval $[x_i, x_{i+1}]$, there exist 4 cubic B-splines(refer in 2.1). $S(x)$ gives the approximated value of $y(x)$ in this sub-interval. Where $S(x)$ is some linear combination of 4 cubic B-splines over this sub-interval.

Therefore,

$$S(x) = \sum \alpha_k B_k(x) \qquad \forall \quad i-1 \leq k \leq i+2 \tag{2.4}$$

where α_k are the corresponding coefficients of the collocation points that must be evaluated. Similarly,

$$S'(x) = \sum \alpha_k B'_k(x) \qquad \forall \quad i-1 \leq k \leq i+2 \tag{2.5}$$

$$S''(x) = \sum \alpha_k B''_k(x) \qquad \forall \quad i-1 \leq k \leq i+2 \tag{2.6}$$

Now , after substituting the former approximate solution into the given equation 2.3 and using the the table 2.1 ,we get the corresponding coefficients as,

$$\alpha_{i-1} := 1/h^2 - a(x_i)/2h + b(x_i)/6 \tag{2.7}$$

$$\alpha_i := -2/h^2 + 2b(x_i)/3 \tag{2.8}$$

$$\alpha_{i+1} := -1/h^2 + a(x_i)/2h + b(x_i)/6 \tag{2.9}$$

With this, we obtain $n+1$ equations since $\forall 0 \leq i \leq n$ and $n+3$ unknowns since α_{-1} and α_{n+1} are also unknowns. We can obtain 2 more equations using the given Boundary conditions. $y(a) = \eta_1, y(b) = \eta_2$ For simplicity, assume a = 0, b = 1. So we get,

$$\alpha_{-1} + 4\alpha_0 + \alpha_1 = 6\eta_1 \tag{2.10}$$

$$\alpha_{n-1} + 4\alpha_n + \alpha_{n+1} = 6\eta_2 \tag{2.11}$$

Hence, using these, we can solve the system of equations $AX = F$ (where A is the coefficient matrix , X are alpha values and F are function values matrix) to obtain the approximate solution.

This can be done by finding inverse of A in MATLAB.

After finding the X (column matrix), we can get the approximated value of $y(x_i)$ using

$$s(x_i) = (\alpha_{i-1} + 4\alpha_i + \alpha_{i+1})/6 \tag{2.12}$$

and hence solving the given BVP numerically. In the next section we will see some solved examples using the same theory.

2.2.1 Examples of BVP

$$y'' + 4y' + 4y = 0; \quad y(0) = 1, y(1) = 3; \tag{2.13}$$

The exact solution is given by $Y_{exact} = (3e^2 - 1)xe^{-2x} + e^{-2x}$. Using B-spline method explained in previous section to approximate the solution for the equation 2.13, we get the error as tabulated in the table.

No.	N (number of collocation points)	Error
1	64	6.01E-4
2	128	1.52E-4
3	256	3.79E-5
4	512	9.46E-6
5	1024	2.37E-6
6	2048	5.92E-7
7	4096	1.47E-7

TABLE 2.2: Table showing Number of collocation points vs error for given BVP

The plot of Numerical solution and Exact solution is shown..

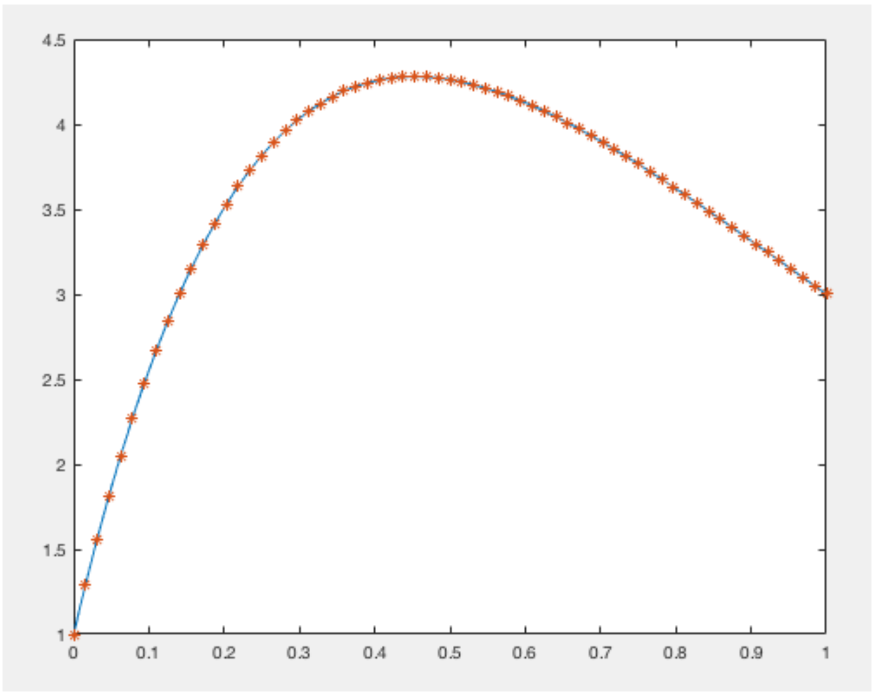

FIGURE 2.1: Numerical solution of BVP, N = 64

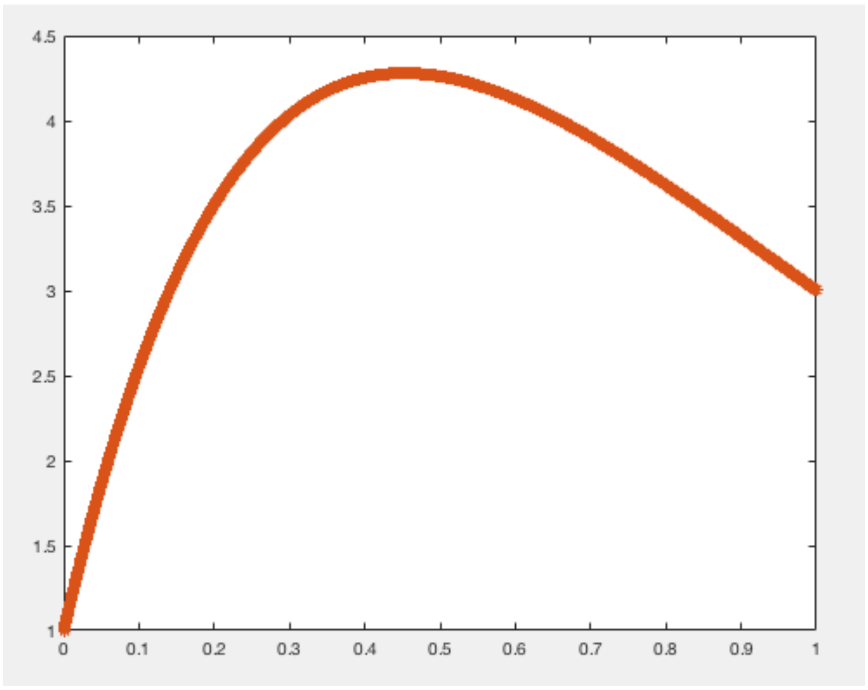

FIGURE 2.2: Numerical solution of BVP, N = 1024

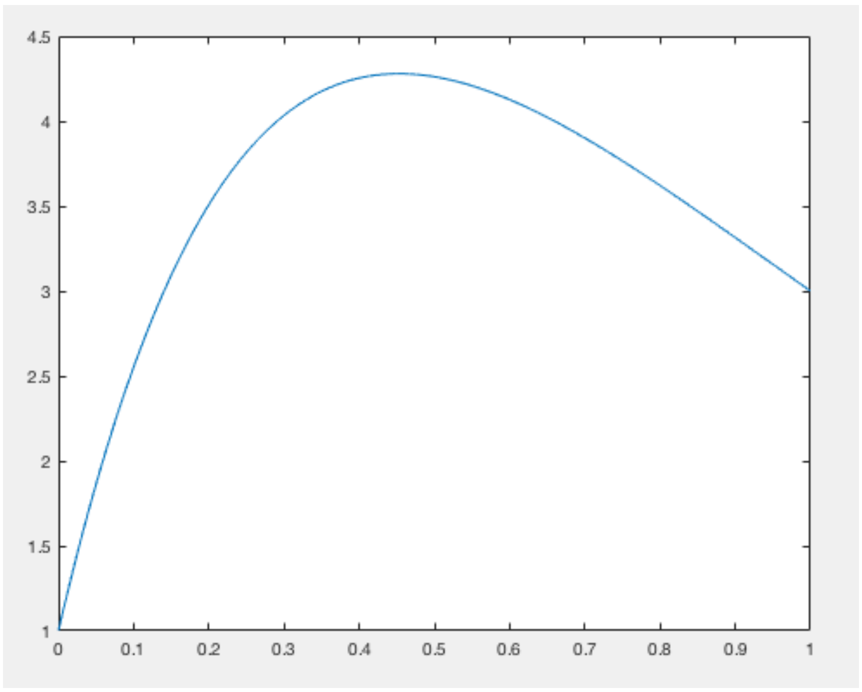

FIGURE 2.3: Exact solution of BVP

2.3 Application of Cubic B-Splines to solve PDE's

A common finite difference scheme is applied to discretise the time space while the cubic B-spline is used to interpolate the approximated solutions at time t. The Partial differential equation governing the heat equation is given by

$$u_t = \alpha u_{xx}, \qquad a \leq x \leq b, t > 0 \tag{2.14}$$

with the initial and boundary conditions given by,

$$u(x, t_in) = f(x) \tag{2.15}$$

$$u(x_in, t) = g_1(t) \tag{2.16}$$

$$u(x_f, t) = g_2(t) \tag{2.17}$$

where α denotes the thermal diffussion constant of the rod. After discretising with respect to time space we get,

$$u(x, t_i) = u(x, t_{i-1}) + \delta t \alpha u'' \tag{2.18}$$

Now, we approximate $u(x, t_i)$ using the B-spline method discussed in the previous section.

2.3.1 Solution of Heat equation

Now, lets apply the method discussed in previous section to find the approximate solution of the linear Heat PDE.

$$u_t = u_{xx}, \qquad 0 \leq x \leq 1, t > 0 \tag{2.19}$$

with initial and boundary conditions

$$u(x,0) = \frac{\sin(\pi x)}{2}$$
$$u(0,t) = 0$$
$$u(1,t) = 0$$

The exact solution is given by $Y_{exact} = \frac{\sin(\pi x)e^{-\pi^2 t}}{2}$. Using B-spline method to approximate the solution for the equation 3.27, we get the error as tabulated in the table below.

No.	N (number of collocation points)	Error
2	8	7.30E-2
3	16	3.79E-2
4	32	1.61E-2
5	64	5.20E-3
6	128	1.70E-3

TABLE 2.3: Table showing Number of collocation points vs error for give Heat equation 3.27

As we can see the order of convergence for this problem is nearly which can be seen by the error results we got. for every double points of collocation, the error gets reduced to half.

The mesh plot for this heat equation and also the plot of numerical solution for heat equation is shown below.

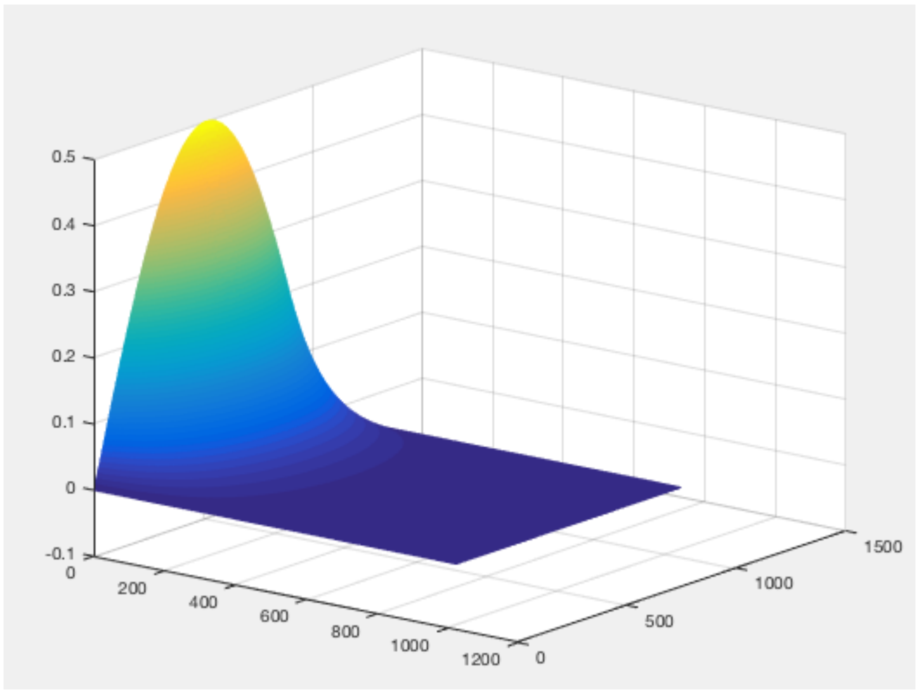

FIGURE 2.4: mesh plot for Heat Equation 3.27

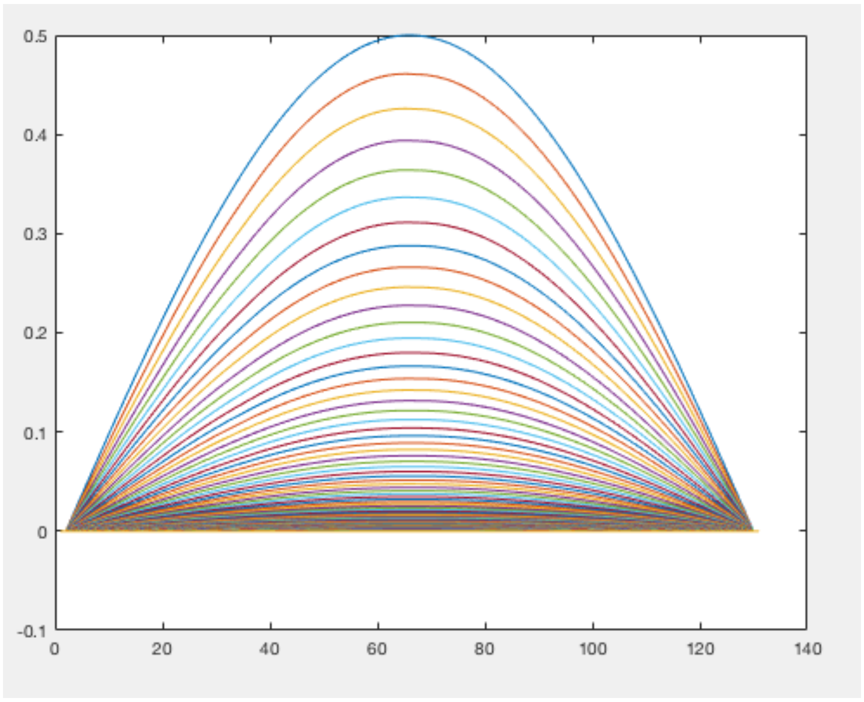

FIGURE 2.5: Plot of Numerical solution for Heat Equation 3.27

2.4 Solution of Black Scholes equation

$$\frac{\partial u}{\partial t} = \frac{1}{2}\sigma^2 x^2 (\ln x)^2 \frac{\partial^2 u}{\partial x^2} + [\,(\,\frac{1}{T} - r\ln x)\,x + \frac{\sigma^2}{2}x(\ln x)^2\,]\frac{\partial u}{\partial x} \tag{2.20}$$

$$(x,t) \in (0,1) \times (0,T]$$

with initial and boundary conditions

$$\begin{aligned} u(x,0) &= 0, & x &\in (0,1) \\ u(0,t) &= 0, & t &\in [0,T] \\ u(1,t) &= \frac{(1-e^{-rt})}{Tr}, & t &\in [0,T] \end{aligned}$$

In 2.20 σ is the volatility, T is the expiration time, r is the risk free interest rate which is constant throughout the time period of interest.

We solve this Initial boundary value problem using Cubic B-spline methods for PDE as discussed in previous sections.

Discretize the 2.20 in time space , we get

$$u(x,t_i) = u(x,t_{i-1}) + \delta t(\alpha_1 u'' + \alpha_2 u') \tag{2.21}$$

where α_1 and α_2 are coefficients of $\frac{\partial^2 u}{\partial x^2}$ and $\frac{\partial u}{\partial x}$ respectively in 2.20 .

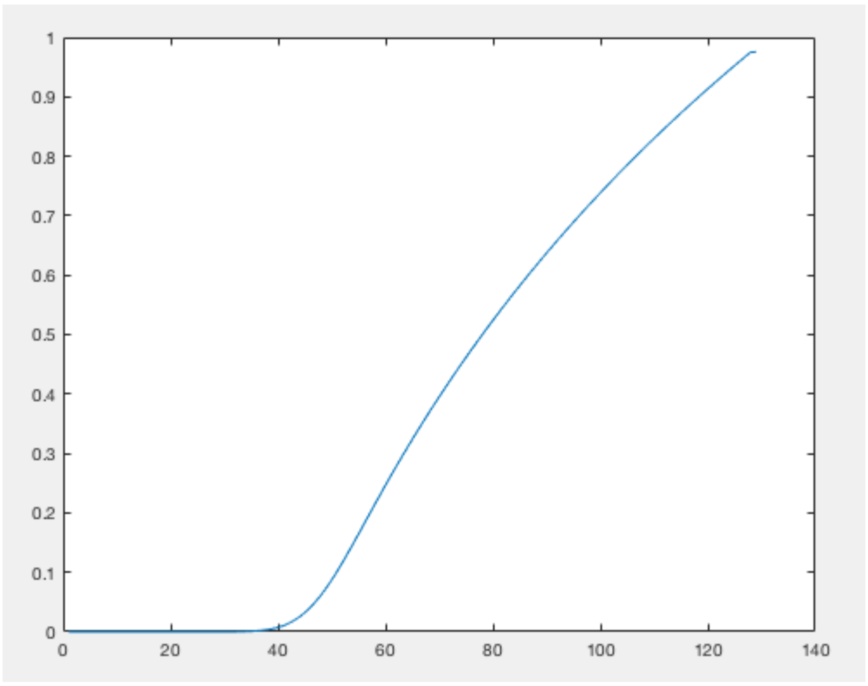

FIGURE 2.6: Exact solution for Black Scholes equation 2.20

No.	N (number of collocation points)	Error (Double Mesh)
1	4	1.15E-1
2	8	3.15E-2
3	16	2.33E-2
4	32	1.56E-2
5	64	9.70E-3
6	128	5.81E-3
7	256	3.50E-3

TABLE 2.4: Table showing Number of collocation points vs double mesh error for Black scholes equation

Chapter 3

Radial Basis functions

3.1 Radial basis function(RBF) Definition

A Radial basis function (RBF) is a real-valued function whose value depends on the distance from the origin, i.e $\phi(x) = \phi(||x||)$ or in general from any other point say x_i so that $\phi(x) = y\phi(||x-x_i||)$. A function is said to be radial function if it satisfies $\phi(x) = \phi(||x||)$. There are large number of Radial basis function that can be used in RBF approximation. For example, MultiQuadratic RBF , Gaussian RBF , Inverse Quadratic RBF , inverse multiquadric RBF. The Inverse Quadratic RBF is the focus of this thesis due to its recent discoveries and its good approximation properties.

$$\phi(r, s) = \frac{1}{1 + (sr)^2} \tag{3.1}$$

where $r = \phi(||x - x_i||)$.

The Inverse Quadratic RBF is a function of the shape parameter s and the 3.1 shows how the function becomes flat as s increases.The MATLAB code which plots this is given in listing .

Some other Radial basis functions are ,

$$\phi(r, s) = e^{-sr^2} \tag{3.2}$$

$$\phi(r, s) = \frac{1}{\sqrt{1 + (sr)^2}} \tag{3.3}$$

3.2 is Gaussian RBF and 3.3 is Inverse Multi-quadratic RBF.

```
x = linspace(0,1,6);
s = 4;
v = 1./(1 + (s.*x).^2 );
plot(x, v, 'b-', 'LineWidth', 3);
hold on;
s = 2;
v = 1./(1 + (s.*x).^2 );
plot(x, v, 'g-', 'LineWidth', 3) ;
hold on;
s = 0.5;
v = 1./(1 + (s.*x).^2 );
plot(x, v, 'r-', 'LineWidth', 3) ;
legend('s = 4','s = 2','s = 0.5');
```

LISTING 3.1: IQX plot code with different shape parameters

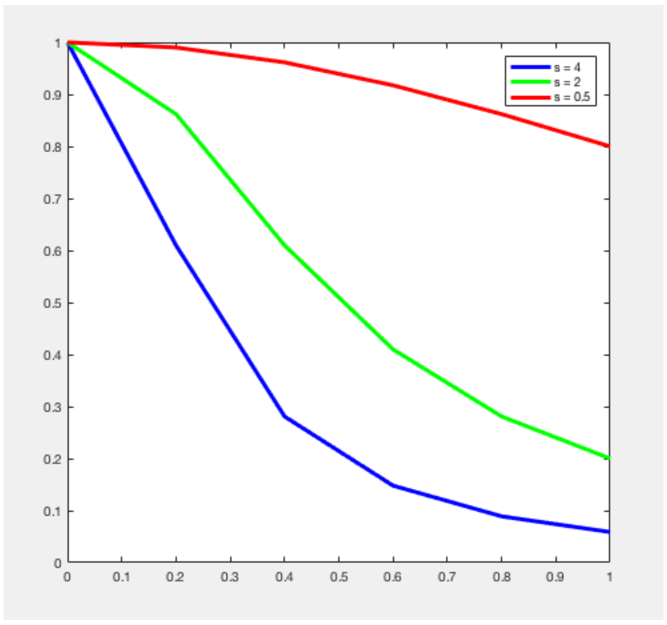

FIGURE 3.1: Plot of IQX with 3 different shape parameters s

3.2 RBF Interpolation Method

In RBF Interpolation method , we take linear combination of radial basis function $\phi(r)$. Given a set of centers $[x_i, \forall i \in [1, N] \in R^d]$, the RBF interpolant function takes the form

$$P(x) = \sum \alpha_k \phi(||x - x_k||, s) \qquad 1 \leq l \leq N \tag{3.4}$$

where r $= ||x||_2 = \sqrt{x_1^2 + ... + x_d^2}$ The coefficients α_k are computed by using the given condition,

$$P(x_i) = f(x_i) \tag{3.5}$$

at a set of nodes which coincides with the pre-defined centers. This results in a NXN linear system of equation which can be represented in Matrix form as ,

$$B\alpha = f \tag{3.6}$$

which can be solved to compute IQX expansion coefficients α. The matrix B has the entries,

$$b_{ij} = \phi(||x_i - x_j||_2), \qquad 1 \leq i \leq N \quad 1 \leq j \leq N \tag{3.7}$$

and is called the *Interpolation matrix.* This matrix serves as the basis of the approximation we will be doing using IQX RBF method. To compute the function interpolant at M points x_i using 3.4 , the MXN *Evaluation matrix* H is formed with entries,

$$h_{ij} = \phi(||x_i - x_j||_2), \qquad 1 \leq i \leq M \quad 1 \leq j \leq N \tag{3.8}$$

Now, this can be represented in matrix form as,

$$f_\alpha = H\alpha \tag{3.9}$$

which can be evaluated at the M centers in MATLAB.

3.2.1 Aprroximating Derivatives

By linearity of the equation, the RBF equation 3.4 can be used to approximate the derivatives of the function $f(x)$ as

$$\frac{\partial}{\partial x_i}P(x) = \sum \alpha_k \frac{\partial}{\partial x_i}\phi(||x - x_k||, s) \qquad 1 \leq j \leq N \tag{3.10}$$

Other higher order derivatives , mixed partial derivatives can be handled in the same way.

Evaluating at the N centers , 3.10 can be written in matrix notation as,

$$\frac{\partial}{\partial x_i}f_\alpha = \frac{\partial}{\partial x_i}H\alpha \tag{3.11}$$

where the *Evaluation matrix* is the $N \times N$ matrix $\frac{\partial}{\partial x_i}H$ has the entries ,

$$h_{ij} = \frac{\partial}{\partial x_i}\phi(||x_i - x_j||_2), \qquad 1 \leq i, j \leq N \tag{3.12}$$

By substituting $\alpha = B^{-1}f$ in the 3.11 , we get another matrix named *Differentiation matrix* D,

$$D = \frac{\partial}{\partial x_i}HB^{-1} \tag{3.13}$$

Therefore, we can get the approximate derivatives of the function $f(x)$ at the N centers. These can be represented as,

$$\frac{\partial}{\partial x_i}P(x) = Df \tag{3.14}$$

The differentiation matrix is well defined since we know that the matrix B is invertible.

```matlab
function v = rbf(obj,r,s), v = 1./(1 + (s.*r).^2 ); end
% iqx definition

function d = D1(obj,r,s,x), d = -(2*x.*s.^2)./(1.0 + (s.*r).^2 ).^2; end
%first derivative

function d = D2(obj, r, s, x)
        d = 2*s.^2.*(-r.^2.*s.^2 + 4*s.^2*x.^2 - 1)./(r.^2*s.^2 + 1).^3;
%second derivative
end
```

Listing 3.2: First and second derivative of Inverse quadratic RBF

The first derivative and second derivative of the Inverse quadratic function is shown via MATLAB code in above listing.

3.3 Solving PDEs by RBF method

We will use Inverse quadratic function as our Radial basis function. Let L be a linear differential operator and B be the boudnary operator. Let the steady problem be,

$$Lu = f \quad \text{in} \quad \omega \tag{3.15}$$

and the time dependent problem be,

$$\frac{\partial u}{\partial t} = Lu \quad \text{in} \quad \omega \tag{3.16}$$

Both the PDEs have there respective boundary conditions defined on the boundary $\partial\omega$ to make the PDE well-posed.

Let the N distinct centers be divided into two subsets. First subset contains N_1 centers where the original PDE will be enforced upon and the second subset contains N_2 centers where the boundary condition is enforced.

So after applying the linear differential operator l to the equation at the interior center points 3.4 we get,

$$Lu(x) = \sum \alpha_k L\phi(||x - x_k||, s) \quad 1 \leq k \leq N_1 \tag{3.17}$$

Now apply Boundary operator B to the equation 3.4 at the boundary center points we get,

$$Bu(x) = \sum \alpha_k B\phi(||x - x_k||, s) \quad \forall N_1 + 1 \leq k \leq N \tag{3.18}$$

These can be expressed in matrix terms as,

$$H = \begin{bmatrix} L\phi \\ B\phi \end{bmatrix} \tag{3.19}$$

where

$$L\phi_{ij} = L\phi(||x_i - x_j||_2), \quad 1 \leq i \leq N_1 \quad 1 \leq j \leq N \tag{3.20}$$

$$B\phi_{ij} = B\phi(||x_i - x_j||_2), \quad N_1 + 1 \leq i \leq N \quad 1 \leq j \leq N \tag{3.21}$$

So 3.19 forms our H matrix and will be used to compute the approximation values.

From the problem discussed in 3.4, we have that $\alpha = B^{-1}u$ where B is the matrix with elements given by equation 3.7 and the matrix

$$D = HB^{-1} \tag{3.22}$$

Now the steady problem is discretized as

$$Du = f \tag{3.23}$$

and the solution to this is given by,

$$u = D^{-1}f = BH^{-1}f \tag{3.24}$$

Now for the **Time dependent problems**

$$\frac{\partial u}{\partial t} = Lu \quad \in \quad \omega \tag{3.25}$$

These are discretized in space as,

$$\frac{\partial u}{\partial t} = Lu \approx Du \tag{3.26}$$

Now this can be solved using method of lines approach and since B matrix is always non-singular therfore matrix D is always non-singular and this approach is well-posed for time dependent PDEs.

3.4 Applications of Radial basis functions

3.4.1 Solving Heat Equation

Now, lets apply the method discussed in previous section to find the approximate solution of the linear Heat PDE.

$$u_t = ku_{xx}, \qquad x \in [0,1], t > 0 \tag{3.27}$$

with initial and boundary conditions

$$u(x,0) = 6\sin(\pi x)$$
$$u(0,t) = 0$$
$$u(1,t) = 0$$

The exact solution is given by $Y_{exact} = 6\sin(\pi x)e^{-k\pi^2 t}$. Using RBF method to approximate the solution for the equation 3.27, we get the error as tabulated in the table below.

No.	N (number of collocation points)	Error (s = 10)
1	16	1.33E-1
2	32	2.72E-2
3	64	2.01E-3
4	80	5.56E-4
5	96	1.55E-4
6	128	5.37E-5

Table 3.1: Table showing Number of collocation points vs error for give Heat equation 3.27

The approximated numerical solution alongwith the exact solution of the given heat equation and point wise error graph is shown in the below plots.

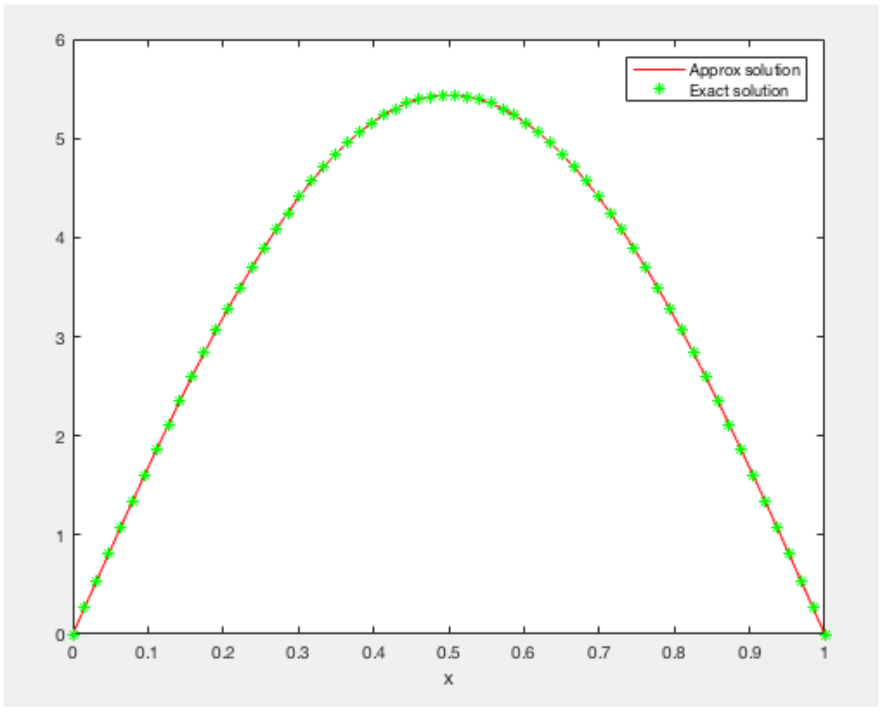

FIGURE 3.2: Numerical solution (N = 64) and Exact solution for Heat Equation 3.27 using RBF

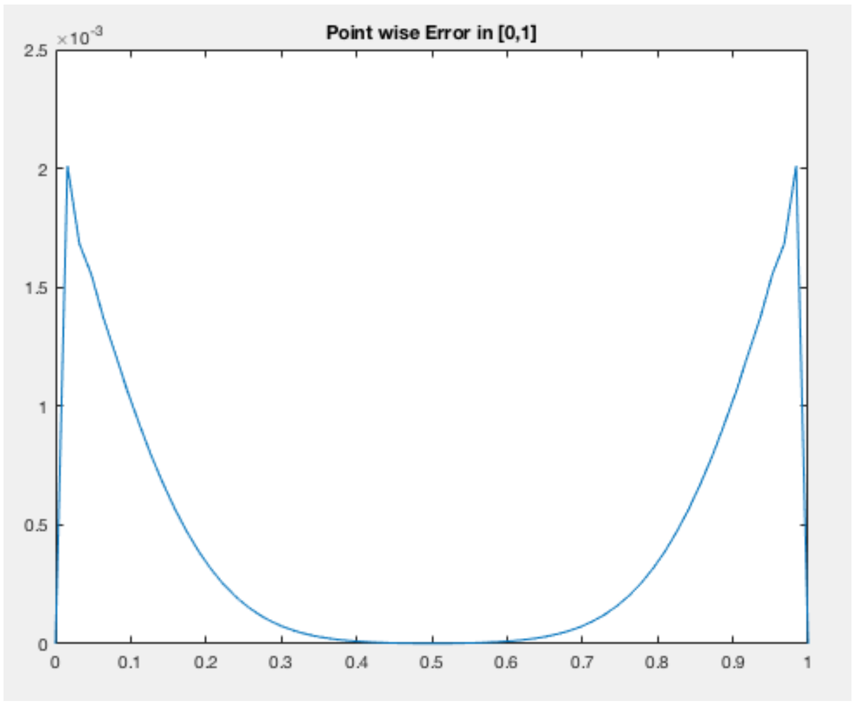

FIGURE 3.3: Error Plot of Numerical solution for Heat Equation 3.27

```
function heateq(N)
phi = iqx();            % inverse quadratic RBF
s = 10;                 % shape parameter
N = 256;                 % number of centers
dt = 0.0001;             % time step size
finalTime = 1;
x = linspace(0,1,N)';
k = 0.01;
r = rbfx.distanceMatrix1d(x);
B = phi.rbf(r,s);
H2 = phi.D2(r,s,r);
D2 = phi.dm(B,H2,0,true);
U = exactSol(x,0);
t = 0;
while t < finalTime
    u = rbfMisc.rk4(U,t,dt,@F);
    t = t + dt;
    u(1) = 0;
    u(N) = 0;
    U = u;
end
maxerr = max(abs(U-exact))
% ----------------------------------------------------------------------
    function fp = F(u,t)
        u(1) = 0; u(end) = 0;
        fp =  k*D2*u;
    end
    function ex = exactSol(x,t)
        ex = 6*sin(pi*x)*exp(-k*(pi)^2*t);
    end
% ---------------------------------------------------------------------
end
\label{heateqn}
```

Listing 3.3: Heat Equation MATLAB code using RBF

3.4.2 Solving 1-D Diffusion Equation

Now, lets apply the method discussed in previous section to find the approximate solution of the linear Diffusion PDE.

$$u_t + au_x = vu_{xx}, \qquad x \in [0,1], t > 0, v > 0 \tag{3.28}$$

with initial and boundary conditions

$$u(x,0) = 0$$
$$u(0,t) = 1$$

And u(1, t) is computed using the exact solution. The exact solution is given by ,

$$U_{exact} = \frac{1}{2}\text{erfc}\Big(\frac{x-t}{2\sqrt{v}\sqrt{t}}\Big) + \frac{1}{2}e^{\frac{x}{v}}\text{erfc}\Big(\frac{x+t}{2\sqrt{v}\sqrt{t}}\Big) \tag{3.29}$$

where erfc is the complementary error function. More about it can be found at http://mathworld.wolfram.com/Erfc.html Using RBF method to approximate the solution for the equation 3.28, we get the error as tabulated in the table below.

No.	N (number of collocation points)	Error (s = 10)
1	8	4.82E-1
2	16	1.33E-1
3	32	3.97E-2
4	48	9.81E-3
4	64	2.52E-3
5	96	1.66E-4
6	112	4.35E-5

Table 3.2: Table showing Number of collocation points vs error for give 1-D diffusion equation 3.28

The approximated numerical solution alongwith the exact solution of the given heat equation and point wise error graph is shown in the below plots.

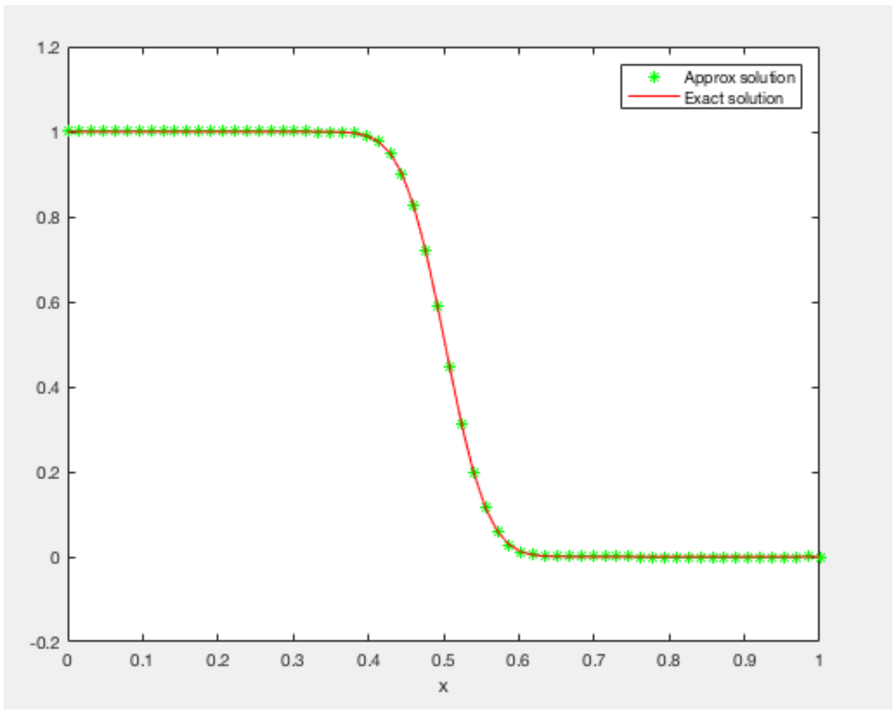

FIGURE 3.4: Numerical solution (N = 64) and Exact solution forr 1-D Diffusion 3.28 using RBF

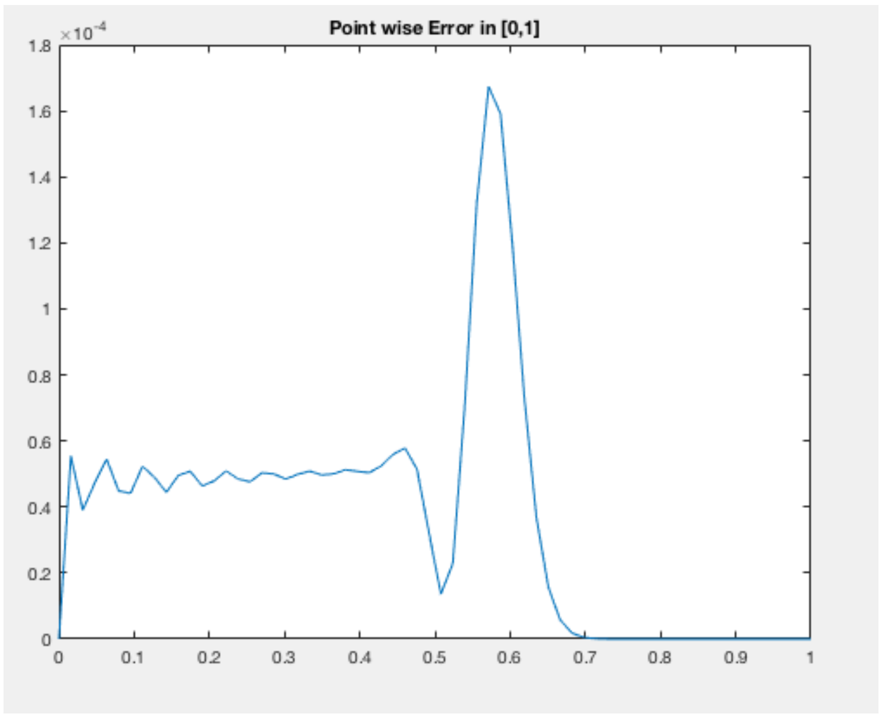

FIGURE 3.5: Error Plot of Numerical solution forr 1-D Diffusion Equation 3.28

```
function DiffusionIQ()
phi = iqx();              % inverse quadratic RBF
s = 6;                  % shape parameter
N = 64;                  % number of centers
dt = 0.001;              % time step size
finalTime = 0.5;
x = linspace(0,1,N)';
r = rbfx.distanceMatrix1d(x);
B = phi.rbf(r,s);
H1 = phi.D1(r,s,r);
D1 = phi.dm(B,H1,0,true);
H2 = phi.D2(r,s,r);
D2 = phi.dm(B,H2,0,true);
U = exactSol(x,0);
t = 0;
while t < finalTime
    u = rbfMisc.rk4(U,t,dt,@F);
    t = t + dt;
    u(1) = 1;
    u(N) = exactSol(1,t);
    U = u;
end
maxerr = max(abs(U-exact))
% ------------------------------------------------------------------------
    function fp = F(u,t)
        u(1) = 1; u(end) = exactSol(1,t);
        fp = 0.002*D2*u -D1*u;
    end
    function sol = exactSol(x,t)
      if t < dt
          if length(x) > 1
              sol(1) = 1;
              sol(2:length(x)) = 0;
              sol = sol(:);
          else
              sol = 0;
          end
      else
          cc = 2.0*sqrt(0.002*t);
          w11 =(x-t)/cc;
          w22 =(x+t)/cc;
          sol = 0.5*(erfc(w11) + exp(x/0.002).*erfc(w22));
      end
    end
% -----------------------------------------------------------------------
end
```

LISTING 3.4: 1-D Linear Diffusion Equation MATLAB code using RBF

3.4.3 Solving Nonlinear Burger Equation

Now, lets apply the method discussed in previous section to find the approximate solution of the linear Diffusion PDE.

$$u_t + \frac{1}{2}u_x^2 = vu_{xx}, \qquad x \in [0,1], t > 0, v > 0 \tag{3.30}$$

with initial and boundary conditions

$$u(x,0) = 0$$
$$u(0,t) = 1$$

And u(1, t) is computed using the exact solution. The exact solution is given by ,

$$U_{exact} = \frac{A+B+C}{10A+2B+C} \tag{3.31}$$

where, $A = 0.1e^a$, $B = 0.5e^b$, $C = e^c$ and, $a = (-x - 0.5 - 4.95t)/2v$, $b = (-x - 0.5)/4v$ and $a = (-x - 0.625 - 0.75t)/2$ Using RBF method to approximate the solution for the equation 3.30, we get the error as tabulated in the table below.

No.	N (number of collocation points)	Error (s = 10)
1	64	2.77E-2
2	96	8.20E-3
3	112	4.00E-3
4	128	1.30E-3
5	154	5.54E-4
6	196	6.74E-5

Table 3.3: Table showing Number of collocation points vs error (s = 10) for give nonlinear Burger equation 3.30

If we change the shape parameter $s = 6$, then we get errors which are relatively better than the errors at $s = 10$.

No.	N (number of collocation points)	Error (s = 6)
1	8	8.03E-1
2	16	1.70E-1
3	32	8.81E-2
4	64	2.60E-2
5	96	7.50E-3
6	112	3.90E-3

Table 3.4: Table showing Number of collocation points vs error (s = 6)for give nonlinear Burger equation 3.30

For smaller values of N for $s = 10$ and for larger values of N for $s = 6$, the system becomes ill-conditioned.

Table 3.5 shows that a smaller s shape parameter is desired for a better accuracy.But small shape parameter can cause conditioning problems which is to be taken care of by using algorithms like **Contour-Pade algorithm**. This algorithm avoids working with the ill conditioned linear system of equation.

No.	s (Shape paramter)	Error (N = 100)
1	4	5.40E-3
2	8	5.90E-3
3	16	7.00E-3
4	32	3.58E-2
5	128	9.99E-1

TABLE 3.5: Table showing variation of Shape paramter vs error for given Burger equation 3.30

Also, With large N, these direct RBF methods becomes too slow as it takes $O(N^3)$ time due to Gaussian elimination. So there are methods like **Domain Decomposition** which can implement this RBF method.

The approximated numerical solution along-with the exact solution of the given heat equation and point wise error graph is shown in the below plots.

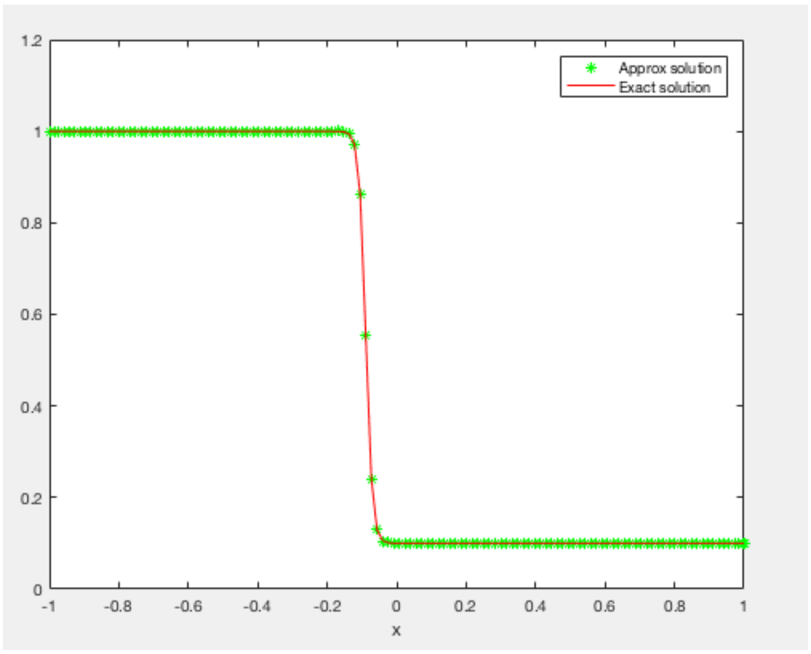

FIGURE 3.6: Numerical solution (N = 128, s = 10) and Exact solution for Burger Equation 3.30 using RBF

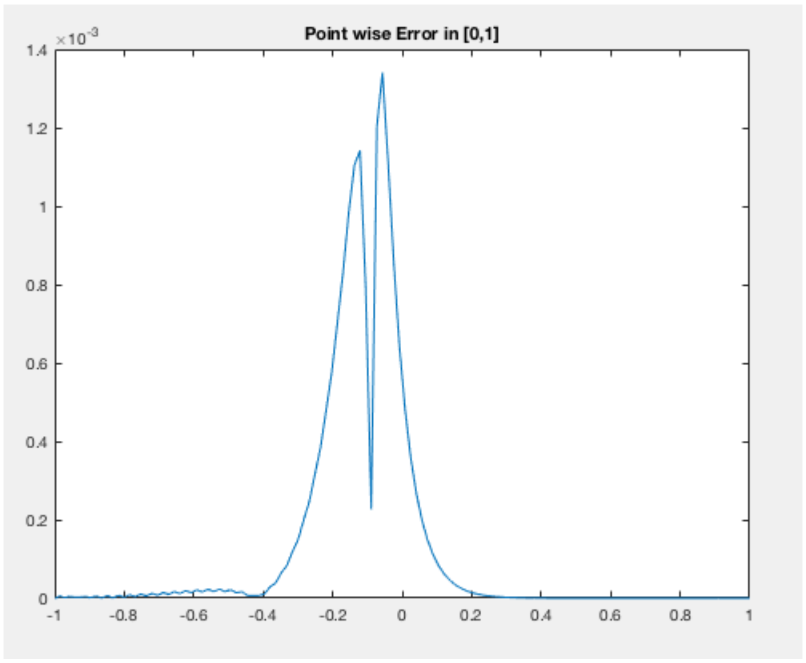

FIGURE 3.7: Error Plot of Numerical solution for Burger Equation 3.30

```
function burger()
nu = 0.004275;
phi = iqx();                % inverse quadratic RBF
s = 10;                    % shape parameter
N = 128;                    % number of centers
dt = 0.001;                % time step size
finalTime = 1;
x = rbfCenters.xKT(N,-1,1,0.999);  % boudary clustered centers
r = rbfx.distanceMatrix1d(x);
B = phi.rbf(r,s);
H1 = phi.D1(r,s,r);
D1 = phi.dm(B,H1,0,true);

H2 = phi.D2(r,s,r);
D2 = phi.dm(B,H2,0,true);

U = exactSol(x,0);
t = 0;
while t < finalTime
    u = rbfMisc.rk4(U,t,dt,@F);
    t = t + dt;
    u(1) = exactSol(-1,t);
    u(N) = exactSol(1,t);
    U = u;
end
maxerr = max(abs(U-exact))
% ------------------------------------------------------------------------
    function fp = F(u,t)
        u(1) = exactSol(-1,t); u(end) = exactSol(1,t);
        fp =  -0.5*D1*u.^2 + nu*D2*u;
    end
    function ex = exactSol(x,t)

       a = 0.05*(x+0.5+4.95*t)/nu;
       b = 0.25*(x+0.5+0.75*t)/nu;
       c = 0.5*(x+0.625)/nu;
       ex = (0.1*exp(-a) + 0.5*exp(-b) + exp(-c))./(exp(-a)+exp(-b)+exp(-c));
    end
% -----------------------------------------------------------------------
end
```

Listing 3.5: Non linear Burger Equation MATLAB code using RBF

Chapter 4

Conclusions

The numerical methods like , finite element methods (FEM), finite volume methods (FVM) and finite difference methods (FDM) requires a mesh to compute the approximated values. But this mesh construction is a non-trivial problem in 3 or more dimensions.
The order of convergence of a method is calculated by the rate at which the error reaches zero when the grid spacing reaches zero. For smooth RBFs, the method has spectral convergence. This means that the method has an error which reduces at the rate of $O(\eta^N)$ where $0 < \eta < 1$. This is different from the above mentioned methods like FDM as they have algebraic convergence rates where the error reduced by $O(N^{-c})$, where c is some constant.
According to the paper " *Benft Fornberg and Natasha flyer: Accuracy of radial basis function interpolation and derivative approximations on 1-D infinite grids.*", the approximation error depends on the *smoothness of the Radial basis function.* For example: If $\phi(r) = r^3$, the convergence rate was found to be $O(h^4)$ whereas when $\phi(r) = r^2 \ln r$, the convergence rate was $O(h^3)$.
For both of these Radial functions the convergence is *algebraic.* Now, for other RBFs like Inverse Quadratic (IQ), MultiQuadratic (MQ), it is seen that they show *Spectral Convergence* when $h- > 0$ and $s- > 0$. It is shown in this paper that MQ converges faster than IQ. In the case Gaussian RBF, it is seen that it has super spectral convergence as shown in the paper.

So, Them major strength of using RBF methods are the flexibility and the the accuracy obtained when applied to scattered multi-dimensional data sets. It has been seen that the error decreases when the Radial basis functions are made flat.The different type of convergence in which various Radial basis functions lies are algebraic, spectral and super-spectral which is computed using the decay rates of the fourier transformation of the radial function.

Huge amount of research has been done to find the **most optimal value of the shape parameter** s. The most basic solution to find this value is to brute force all the parameter values and then find the one which has least maximum absolute error. But this strategy only works if the function being approximated is already known and is also very time costly.

According to this paper "*R. Hardy. Multiquadratic equations of topography and other irregular surfaces. Journal of Geophysical Research.*" the optimal value of s is given by,

$$s = 0.815d \quad \text{where} \quad d = \frac{1}{N}\sum d_i \quad ,1 \leq i \leq N \tag{4.1}$$

where, d_i is the distance from the i^{th} center to its nearest neighbor.
Another paper " *R. Franke. A critical comparison of some methods for the interpolation of scattered data. Technical Report NPS-53-79-03, Naval Postgraduate School, 1979*" gave the formula for the shape parameter as,

$$s = \frac{(1.25D)}{\sqrt{N}} \tag{4.2}$$

where D is the diameter of the smallest cirlces which enclose all the chosen centers.

There are other strategies to improve the accuracy of the approximated solution. **Variable shape parameter** is one of them. In this we choose a different value of s for each center. The benefit of this is that the RBF matrices which is formed (after using variable shape parameters) have more distinct values which leads to a lower condition number, which means having more accuracy.

Bibliography

[1] J. Biazar ,H. Aminikhahb. Exact and numerical solutions for non-linear Burger's equation by VIM

[2] Mohan K. Kadalbajoo, Lok Pati Tripathi , Alpesh Kumar.A cubic B-spline collocation method for a numerical solution of the generalized Black–Scholes equation

[3] Seydel R.U. Tools for Computational Finance (4ed., Springer, 2009)(ISBN 3540929282)

[4] Mohan K. Kadalbajoo, Lok Pati Tripathi. A robust nonuniform B-spline collocation method for solving the generalized Black–Scholes equation

[5] Scott A. Sarra Marshall University and Edward J. Kansa University of California, Davis. Multiquadric Radial Basis Function Approximation Methods for the Numerical Solution of Partial Differential Equations

[6] Sarra, S A 2017 The Matlab Radial Basis Function Toolbox. Journal of Open Research Software, 5: 8, DOI: https://doi.org/10.5334/jors.131

[7] Kansa, E J 1990 Multiquadrics – a scattered data approximation scheme with applications to computational fluid dynamics II: Solutions to parabolic, hyperbolic, and elliptic partial differential equations. Computers and Mathematics with Applications, 19(8/9): 147–161. DOI: https://doi. org/10.1016/0898-1221(90)90271-K