UPPSALA
UNIVERSITET

Evaluating the Performance of Radial Basis Function Methods for Option Pricing

Carl Joneus and Andreas Abrahamsson

Project in Computational Science: Report

January 2017

PROJECT REPORT

Department of Information Technology

Abstract

Trading of different financial derivatives has increased drastically during the last couple of decades. Option contracts that give the holder the right but not the obligation to buy or sell, depending on the option type, an underlying asset at a predetermined price is perhaps the most commonly used. Traders and hedgers use them to speculate or risk minimize the future outcome of the financial market. Since analytical formulas exist only for pricing the most basic option types, there is demand for accurate and efficient numerical models to evaluate option contracts with more complex features. A method for pricing options based on several different underlying assets, referred to as basket options, proposed by Elisabeth Larsson and Victor Shcherbakov is by using radial basis functions (RBF) with a partition of unity framework. The idea with RBF based methods is to approximate the solution as a weighted sum of real-valued radial basis functions centered at a set of scattered nodes. These methods have been proven to show high order error convergence but due to the occurring dense linear system they are computationally expensive to use. In order to solve this numerical issue Larsson and Shcherbakov introduce a partition of unity framework that divides the computational domain into overlapping subdomains, which generates sparsification in the linear system. In this paper we investigated this method further by evaluating it on non-uniform nodal and partition grids. We also investigated how the shape of the applied RBFs should be set in order to reach stable and accurate results. It was shown that a non-uniform nodal grid should be preferred over a uniformly distribution. The computational time can be reduced heavily in order to reach a desired error tolerance by rearranging the evaluation nodes. For the case with non-uniform partition grid, it was shown that by modifying the size and location of the applied patches the method can meet the same accuracy while reducing the computational time, compared to a uniform framework. Further it was shown that the shape of the applied RBFs should be proportional to the fill distance in order to prevent instabilities.

Contents

1 Introduction

1.1 Background

An option contract is a financial tool widely used by traders and hedgers to speculate or risk minimize the outcome of the future financial market. The basic concept of an option is that it gives the holder the right but not the obligation to buy or sell, depending on the option type, an asset at a predetermined price called exercise price. Aside from the buy- or sell type an option can have many different features. The holder can be allowed to exercise the option only at the specified maturity time, which is the case for a so called European option, or be allowed to exercise in prior of and including the maturity time, which applies for the case with an American option. These two types are often referred to as Vanilla options and are the most commonly used option types in the financial market [1]. However, there are many other and more complex option structures, often referred to as Exotic options. Some examples of these types are Chooser options, for which the holder, up until a certain point, can determine whether the option is of type put or call. Another example is the so called Compound option which is an option written on another option [2].

Trading of options increased drastically in 1973 when Fischer Black and Myron Scholes published their Nobel prize awarded Black-Scholes model. [3]. Their model determines the price of a European put or call option in one dimension, meaning that the option is written on one single underlying asset. However, because of the increasing amount of option trading and the introduction of more complex and multi-dimensional option types, which lacks analytical pricing models, there is a demand of accurate and efficient numerical methods to approximate the price. The most commonly used methods in this area are Monte Carlo and finite difference methods. When computing the option value using Monte Carlo methods one solves the stochastic differential equation which describes the price process of the asset under the Black-Scholes model [4]. Monte Carlo methods are efficient for multi-dimensional problems, although the error is converging slowly. For the finite difference case one approximates the option value by solving the Black-Scholes partial differential equation [3]. Finite difference methods are computationally costly for multi-dimensional problems but show high order error convergence [5].

Another method for pricing options, proposed by Victor Shcherbakov and Elisabeth Larsson in [5] is by using radial basis function partition of unity methods (RBF-PUM). It has been shown that RBF methods can reach high order error convergence, especially for options written on several different assets. RBF methods are meshfree, meaning that there is no requirement of any connections between the nodes. This property makes the method applicable not only on trivial computational domains such as squares, rectangles, cubes etc. but also on more complex geometries. One disadvantage for RBF methods when pricing options is that the linear system that needs to be solved for each time step becomes dense. In order to solve this numerical issue Shcherbakov and Larsson introduce a partition of unity framework which is applied onto the computational domain. The global linear system obtained by

collecting the local systems from all partitions will allow for more sparsification which leads to higher efficiency in terms of CPU time.

The results in [5] showed that RBF-PUM is competitive with finite differences when pricing multi-asset options. The method has high order error convergence for smooth problems. However, even for non-smooth problems, which is the case of option pricing, it reaches a certain error tolerance using fewer nodal points. As a results from this RBF-PUM will be more efficient in terms of computational time. In this paper we aim to investigate the method further in order to optimize it in terms of error convergence and efficiency. Due to a discontinuity in the first derivative of the initial value that arises at the exercise price we expect it is in this specific region where the error will be dominant. Because of this we investigate the outcome of RBF-PUM when applied on different types of clustered evaluation grids focused on this region. Also, it was shown in [5] that an increasing amount of partitions leads to lower accuracy but higher efficiency in terms of computational time. Therefore, we evaluate the method when using a non-uniform framework with fewer partitions in the region near the exercise price. The non-uniform nodal and partition grids are implemented and compared with completely uniform grids as was used in [5]. Another parameter that has to be taken into consideration in order to reach stable and accurate results is the shape of the applied radial basis functions. The optimal shape parameter is computed and the resulting value are used for all further experiments. All simulations are made on a European two-dimensional put option that pays no dividend. As there is no closed formula for pricing this type of option a finite difference solution on a fine grid is used as reference.

The outline of the paper is as follows. Firstly we go through the theory behind RBF-based methods, the additional partition of unity framework and the Black-Scholes equation. Secondly, we explain how RBF-PUM is used for option pricing in theory, the different types of non-uniform nodal and partition grids is presented and the choice of shape parameter is motivated. Thirdly, the numerical results from our computations are presented. Lastly, conclusions and discussions regarding RBF-PUM and our obtained results.

1.2 Theory

In this section we explain the general theory behind methods based on radial basis functions and the partition of unity framework. We also give a brief introduction of the Black-Scholes equation for European basket options.

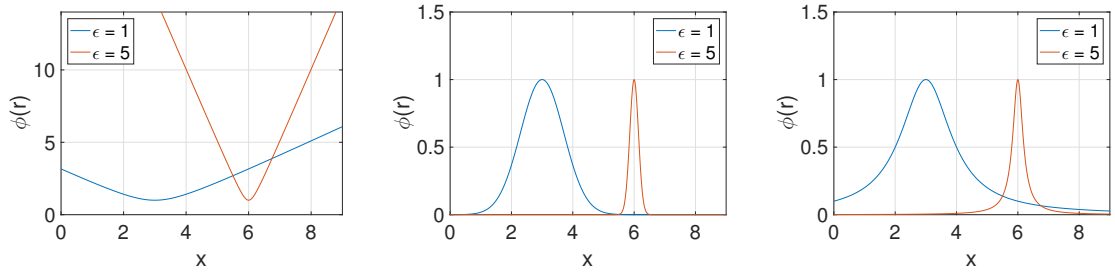
1.2.1 Radial basis function methods

RBF-approximations are based on real-valued functions applied on a set of scattered nodes $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$ where $\mathbf{x}_i \in \mathbb{R}^d$. Due to the fact that they are applicable to complex geometries of the computational domain and have been proven to reach high order convergence [5] [6] they are a preferred choice to discretize and solve partial differential equations (PDE) numerically. The idea is to compute an interpolant, $\tilde{u}$, based on a chosen RBF evaluated at

the nodes with some coefficients λ,

$$\tilde{u}(\mathbf{x}) = \sum_{i=1}^{N} \lambda_i \phi(||\mathbf{x} - \mathbf{x}_i||), \qquad \mathbf{x} \in \Omega. \tag{1.2.1}$$

The RBF, ϕ, can be of different types such as multi-quadratic $\phi(r) = \sqrt{1 + (\varepsilon r)^2}$, Gaussian $\phi(r) = e^{-(\varepsilon r)^2}$ or inverse-quadratic $\phi(r) = \frac{1}{1+(\varepsilon r)^2}$. Figure 1.2.1 illustrates the three mentioned RBFs computed in one dimension and their dependency of the shape parameter, ε, which determines the flatness of the curve. The choice of basis function and shape parameter are crucial in order to achieve good accuracy and stable results. These choices are motivated and discussed in Sections 1.3.2 and 2.1.



(a): Multi-quadratic (b): Gaussian (c): Inverse quadratic

Figure 1.2.1 (a-c). Visualization of multi-quadratic, Gaussian and inverse quadratic RBF, respectively, computed with two different values of the shape parameter ε. Here r denotes the Euclidean distance between all nodal values X and the nodal value x_i corresponding to basis function ϕ_i. That is, $r = ||X - x_i||$.

By forcing our interpolant to be our approximated solution, $\tilde{u}(\mathbf{x}) = u(\mathbf{x})$ one can write the RBF approximation described in equation (1.2.1) as

$$u(\mathbf{x}) = \phi(\mathbf{x}, \mathbf{X})\lambda, \tag{1.2.2}$$

where $\lambda = (\lambda_1, ..., \lambda_N)^T$ and $\phi(\mathbf{x}, \mathbf{X}) = (\phi(||\mathbf{x} - \mathbf{x}_1||), ..., \phi(||\mathbf{x} - \mathbf{x}_N||))$.

Equation (1.2.2) is used to form a linear system which describes the RBF approximation in terms of all nodal values,

$$u(\mathbf{X}) = \phi(\mathbf{X}, \mathbf{X})\lambda, \tag{1.2.3}$$

where

$$\phi(\mathbf{X}, \mathbf{X}) = \begin{bmatrix} \phi(||\mathbf{x}_1 - \mathbf{x}_1||) & \phi(||\mathbf{x}_1 - \mathbf{x}_2||) & ... & \phi(||\mathbf{x}_1 - \mathbf{x}_N||) \\ . & . & & . \\ . & . & & . \\ . & . & & . \\ \phi(||\mathbf{x}_N - \mathbf{x}_1||) & \phi(||\mathbf{x}_N - \mathbf{x}_2||) & ... & \phi(||\mathbf{x}_N - \mathbf{x}_N||) \end{bmatrix}$$

and

$$u(\mathbf{X}) = (u(\mathbf{x}_1), ..., u(\mathbf{x}_N))^T.$$

By solving equation (1.2.3) for λ, equation (1.2.2) can be formulated as

$$u(\mathbf{x}) = \phi(\mathbf{x}, \mathbf{X})\phi(\mathbf{X}, \mathbf{X})^{-1}u(\mathbf{X}). \tag{1.2.4}$$

1.2.2 Partition of unity

A drawback with RBF methods is that the linear system described in equation (1.2.3) becomes dense and computationally costly to solve numerically. In order to resolve this computational issue a partition of unity extension can be applied to the method. The framework divides the computational domain into a set of overlapping subdomains, Ω_j, $j = 1, ..., P$, such that

$$\bigcup_{j=1}^{P} \Omega_j \supseteq \Omega. \tag{1.2.5}$$

The RBF interpolants are constructed locally and then combined into a global solution. The partitioning generates local sparsification in our global RBF approximation and also enables for solving in parallel [5]. Figure 1.2.2 shows an example of a partition framework applied onto a domain.

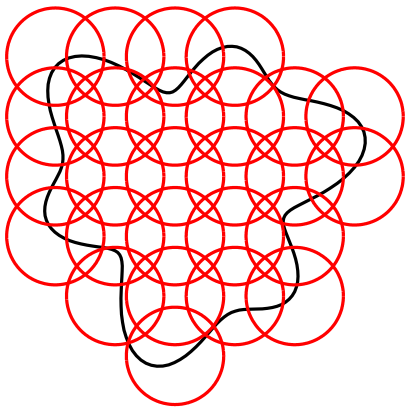


Figure 1.2.2. A set of partitions (red circles) covering the computational domain drawn in black.

For each partition a weight function, $\omega(\mathbf{x})$, is constructed. The weight functions should satisfy the partition of unity property,

$$\sum_{j=1}^{P} \omega_j(\mathbf{x}) = 1, \qquad \forall\ \mathbf{x} \in\ \Omega. \tag{1.2.6}$$

In order to fulfill this requirement the weight functions $\omega_j(\mathbf{x})$ are constructed using Shepard's method [7],

$$\omega_j(\mathbf{x}) = \frac{\varphi_j(\mathbf{x})}{\sum_{j=1}^{P} \varphi_j(\mathbf{x})}, \qquad j = 1, ..., P. \tag{1.2.7}$$

The weight function should only be nonzero-valued inside of its corresponding partition, therefore $\varphi_j(\mathbf{x})$ is chosen to be a compactly supported C^2 Wendland function [8],

$$\varphi(r) = \begin{cases} (4r+1)(1-r)^4 & \text{if } 0 \leq r \leq 1 \\ 0 & \text{if } r > 1. \end{cases} \tag{1.2.8}$$

The geometry of the applied partitions is optional. For implementation ease they are often chosen to be circular or elliptic. In this paper all computations are made using circular partitions, as was the case in [5]. Therefore, the Wendland functions needs to be shifted and scaled such that

$$\varphi_j(\mathbf{x}) = \varphi\left(\frac{||\mathbf{x} - \mathbf{c}_j||}{R_j}\right), \tag{1.2.9}$$

where R_j and $\mathbf{c}_j$ is the radius and centre point respectively, corresponding to partition Ω_j.

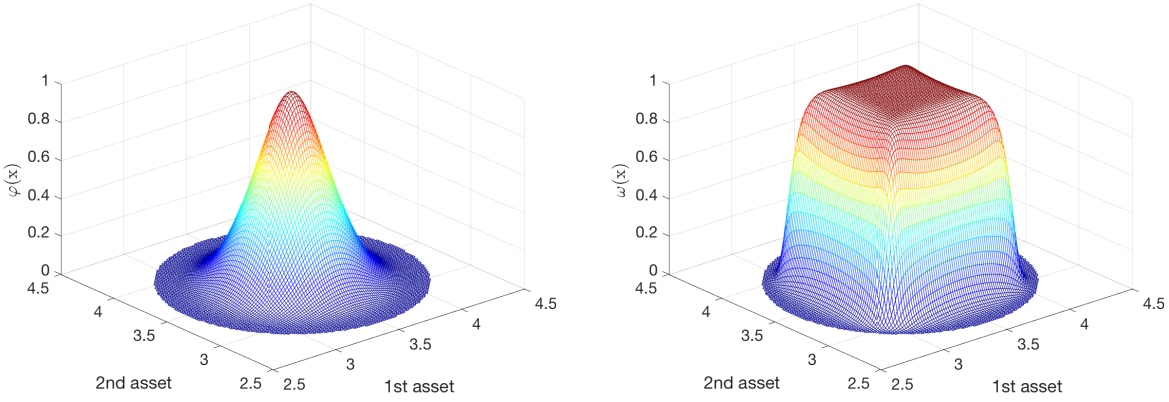


Figure 1.2.3 To the left, an example of the Wendland function from equation (1.2.9). To the right, an example of a weight function described in equation (1.2.7).

Using the constructed weight functions, the global RBF approximation is computed as a weighted sum of the local approximations,

$$u^{glob}(\mathbf{x}) = \sum_{j=1}^{P} \omega_j(\mathbf{x}) u_j^{loc}(\mathbf{x}). \tag{1.2.10}$$

1.2.3 Black-Scholes equation

The following PDE is the Black-Scholes equation describing the value of a d-dimensional European basket option [3],

$$\frac{\partial V(\mathbf{s},t)}{\partial t} + \frac{1}{2}\sum_{i,j=1}^{d} \sigma_{i,j} s_i s_j \frac{\partial^2 V(\mathbf{s},t)}{\partial s_i \partial s_j} + \sum_{i=1}^{d} (r - D_i) s_i \frac{\partial V(\mathbf{s},t)}{\partial s_i} - rV(\mathbf{s},t) = 0, \tag{1.2.11}$$
$$t \in (0, T].$$

V is the option value, $\mathbf{s} = (s_1, s_2, ..., s_d)$ is the price at time t of the d underlying assets, r is the interest rate, D_i is the dividend payed out by the i-th asset, $\sigma_{i,j}$ is the volatility matrix describing the correlation between the different assets and T is the time of maturity of the option.

By defining the linear operator

$$\mathcal{L} = \frac{1}{2}\sum_{i,j=1}^{d} \sigma_{i,j} s_i s_j \frac{\partial^2}{\partial s_i \partial s_j} + \sum_{i=1}^{d} (r - D_i) s_i \frac{\partial}{\partial s_i} - r, \tag{1.2.12}$$

equation (1.2.11) takes the form

$$-\frac{\partial V(\mathbf{s}, t)}{\partial t} = \mathcal{L}V(\mathbf{s}, t). \tag{1.2.13}$$

The value of an option is known at its maturity time from the so called payoff function. Equation (1.2.14) shows the payoff function for the case with a d-dimensional European put option [3],

$$V(\mathbf{s}, T) = \Phi(\mathbf{s}), \tag{1.2.14}$$

where

$$\Phi(\mathbf{s}) = (K - \sum_{i=1}^{d} \beta_i s_i)_+. \tag{1.2.15}$$

Here $(\cdot)_+$ denotes $\max(\cdot, 0)$ and β_i is the weight of the i-th asset, e.g. for a two-dimensional option consisting of 100 shares of s_1 and 200 shares of s_2 then $\beta = (\frac{1}{3}, \frac{2}{3})$. As we are interested of the option value today, the PDE needs to be integrated backwards in time with the payoff described in equation (1.2.15) used as initial condition.

For a put option the contract becomes worthless if the price of the underlying asset tends to infinity, which is trivial because the holder would then sell to the market price instead of a lower exercise price. In theory, the domain of the asset price is $0 \leq s < \infty$ but to be able to price financial derivatives for practical use one need to truncate the infinite interval to a finite computational domain. The truncation has to be made in such way that desired boundary conditions can be set without losing accuracy. For the case with a d-dimensional European put option the following boundary conditions need to hold,

$$V(\mathbf{0}, t) = Ke^{-r(T-t)} \quad \text{for } t \in [0, T] \tag{1.2.16}$$

At the far-field boundary we require

$$V(\mathbf{s}, t) = 0 \quad \text{for } t \in [0, T] \text{ and } \mathbf{s} \in \Gamma^F, \tag{1.2.17}$$

where $\Gamma^F = \bigcup_{i=1}^{d} \Gamma_i^F$ and Γ_i^F is the boundary of where $s_i = s_{max}$ in the truncated domain.

1.3 Method

This section explains how the RBF-PUM method is used for option pricing. It is also shown how the non-uniform nodal and partition grids, on which the RBF-PUM will be evaluated, are constructed as well as motivate the choice of shape parameter.

1.3.1 RBF-PUM for option pricing

In order to determine the price of a d-dimensional European basket option we apply the Black-Scholes linear differential operator described in equation (1.2.12) on our RBF-PUM approximation defined in equation (1.2.10),

$$\mathcal{L}u^{glob}(\mathbf{x}) = \sum_{j=1}^{P} \mathcal{L}(\omega_j(\mathbf{x})u_j^{loc}(\mathbf{x})) = \sum_{j=1}^{P} \mathcal{L}[\omega_j(\mathbf{x})\phi(\mathbf{x}, \mathbf{X}_j)]\phi(\mathbf{X}_j, \mathbf{X}_j)^{-1}u_j^{loc}(\mathbf{X}_j), \tag{1.3.1}$$

where $\mathbf{X}_j$ is the set of nodes located in the j-th partition. We define the local, discrete, spatial differential matrix as

$$D_j^{\mathcal{L}}(\mathbf{X}_j, \mathbf{X}_j) = \mathcal{L}[W_j(\mathbf{X}_j)\phi(\mathbf{X}_j, \mathbf{X}_j)]\phi(\mathbf{X}_j, \mathbf{X}_j)^{-1}, \tag{1.3.2}$$

where $W_j(\mathbf{X}_j) = \text{diag}(\omega_j(\mathbf{X}_j))$. We collocate the contribution from all partitions to define our global differential matrix

$$D^{\mathcal{L}}(\mathbf{X}, \mathbf{X}) = \sum_{j=1}^{P} D_j^{\mathcal{L}}(\mathbf{X}_j, \mathbf{X}_j). \tag{1.3.3}$$

Equation (1.3.1) now takes the form

$$\mathcal{L}u(\mathbf{X}) = D^{\mathcal{L}}(\mathbf{X}, \mathbf{X})u(\mathbf{X}). \tag{1.3.4}$$

From the previous section we remember the Black-Scholes equation,

$$-\frac{\partial V(\mathbf{S}, t)}{\partial t} = \mathcal{L}V(\mathbf{S}, t), \tag{1.3.5}$$

where $\mathbf{S} = \{\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_N\}$ and $\mathbf{s}_i \in \mathbb{R}^d$ is the spot prices of the d assets. Using the equality in equation (1.3.4) leads to the following expression:

$$-\frac{\partial V(\mathbf{S}, t)}{\partial t} = D^{\mathcal{L}}(\mathbf{S}, \mathbf{S})V(\mathbf{S}, t). \tag{1.3.6}$$

By introducing the transformation $\tau = T - t$, which allows us to solve forward in time, equation (1.3.6) can then be formulated as

$$\frac{\partial V(\mathbf{S}, \tau)}{\partial \tau} = D^{\mathcal{L}}(\mathbf{S}, \mathbf{S})V(\mathbf{S}, \tau), \tag{1.3.7}$$

where $-\frac{\partial V}{\partial t} = \frac{\partial V}{\partial \tau}$ follows from the chain rule.

The time marching in equation (1.3.7) is performed using finite differences. In this paper, all computations are made using second order backward differentiation formula (BDF-2) [3],

$$\frac{V(\mathbf{S}, \tau^{n+2}) - \frac{4}{3}V(\mathbf{S}, \tau^{n+1}) + \frac{1}{3}V(\mathbf{S}, \tau^{n})}{\frac{2}{3}\Delta\tau} = D^{\mathcal{L}}(\mathbf{S}, \mathbf{S})V(\mathbf{S}, \tau^{n+2}), \tag{1.3.8}$$

where $n = 1, 2, ..., Nt$ and Nt is the total number of time steps. $\tau^n = T - t^n$ and t is the backwards time, $t^1 = T, t^2 = T - \Delta t, ..., t^{Nt} = 0$. By rearranging the terms in equation (1.3.8) and solving for $V(\mathbf{S}, \tau^{n+2})$ we obtain

$$V(\mathbf{S}, \tau^{n+2}) = \left(I - \frac{2}{3}\Delta\tau D^{\mathcal{L}}(\mathbf{S}, \mathbf{S})\right)^{-1} \left(\frac{4}{3}V(\mathbf{S}, \tau^{n+1}) - \frac{1}{3}V(\mathbf{S}, \tau^{n})\right), \tag{1.3.9}$$

where I is the identity matrix. BDF-2 is an implicit discretization scheme, meaning we have to solve the linear system described in equation (1.3.9) for each iteration. Implicit methods are unconditionally stable but we have to pay in computational time since the linear system to solve for each iteration is an expensive operation [3].

1.3.2 Choice of the shape parameter

The choice of the shape parameter ε is crucial to obtain stable and accurate results. The shape parameter determines the flatness of the applied shape function, illustrated in Figure 1.2.1. Consider the two-dimensional case, where $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$ and $\mathbf{x} \in \mathbb{R}^2$ is the nodal set that defines our computational domain. The obtained matrix $\phi(\mathbf{X}, \mathbf{X})$ then contains the basis function valued for the Euclidean distance between all nodal combinations. More specific, the matrix $\phi(\mathbf{X}, \mathbf{X})$ is a $N \times N$ matrix, where the i-th row consists of the basis function valued for the distances between the i-th node $\mathbf{x}_i$ and $\mathbf{X}$. When the basis function is too flat, that is when ε is small, the function values are approximately equal $\phi(\mathbf{x}_i, \mathbf{x}_1) \approx \phi(\mathbf{x}_i, \mathbf{x}_2) \approx ... \approx \phi(\mathbf{x}_i, \mathbf{x}_N)$ for $i = 1, ..., N$. This results in a close to singular matrix $\phi(\mathbf{X}, \mathbf{X})$, which cause problems when solving linear systems and leads to unstable outcomes. In order to handle this difficulty, the shape parameter has to be chosen with respect to the problem size. As a rule of thumb, ε can be computed as

$$\varepsilon = \frac{C}{h}, \tag{1.3.10}$$

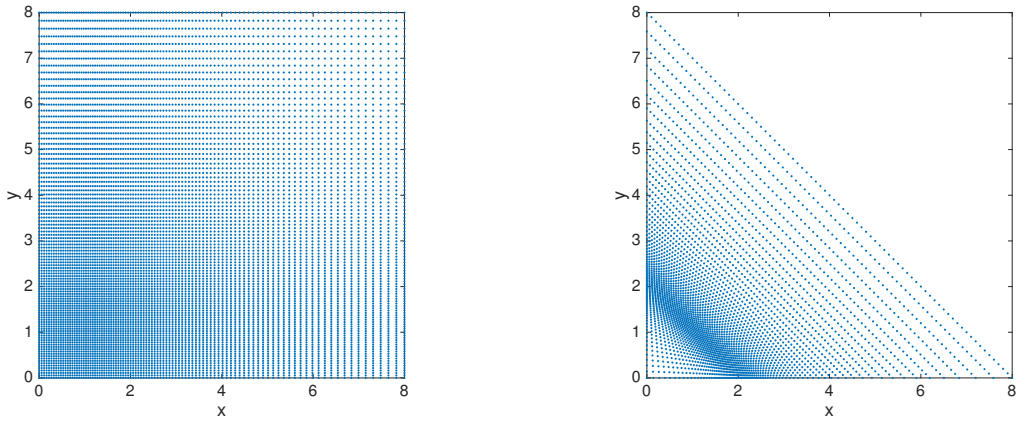
where h is the fill distance and C is a constant which has to be determined. Focusing only on reaching stable results, the relation described in equation (1.3.10) is rather a minimum demand $\varepsilon \geq \frac{C}{h}$. That is, when N is increased then ε must also be increased but the upper limit on ε can be set arbitrarily.

1.3.3 Construction of non-uniform nodal grids

It is shown in [5] that the accuracy for the RBF-PUM is increasing with the problem size, however, an increasing number of evaluation nodes also lead to a higher computational cost. We aim to improve the accuracy but maintain the efficiency by implementing two non-uniform nodal grids. The idea by using a non-uniform nodal grid is to place the majority of the nodes in a specific region where we expect the error to be large and therefore receive higher accuracy of the approximation. The number of nodal points remain constant but the layout is changed. For the case with a multi-dimensional, European put option the payoff function defined in equation (1.2.15) is used as initial condition. The payoff function has a discontinuity in the first derivative at the exercise price, K, which is why we are expecting the error to be dominant in that region. To achieve a higher node point density around the exercise price region we construct two non-uniform grids on which we will evaluate the RBF-PUM, the cross-clustered grid [6] and the diagonal-clustered grid [9].

Consider again the two-dimensional case. The cross-clustered grid is similar to a uniformly distributed grid, the nodes are structured placed along the x^1- and x^2-direction such that $\mathbf{X} = \left\{(x_1^1, x_1^2), (x_1^1, x_2^2), ..., (x_1^1, x_{N_2}^2), (x_2^1, x_1^2), ..., (x_{N_1}^1, x_{N_2}^2)\right\}$, where $\mathbf{X}$ defines our nodal set and N_1 and N_2 denote the number of space discretization steps in direction x^1 and x^2 respectively. However, the density is increased in a specified region, as a result from this the density has to be decreased in another part of the domain, this since the number of nodes is kept constant. The cross-clustered nodal distribution is demonstrated in Figure 1.3.1 (a).

For the diagonal case, the nodes placed on the axes are distributed the same way as for the cross-clustered grid, that is more dense at a specified region. However, the distribution differs for the interior nodes. The aim is to create a grid where the interior nodes are placed along the diagonal between the boundary points $\mathbf{x} = (x_1^1, x_i^2)$ and $\mathbf{x} = (x_i^1, x_1^2)$ where $i = 1, .., N_1$. Because of this structure a diagonal clustered grid can only be constructed when the number of discretization points are the same in both dimensions, $N_1 = N_2$. Further, the grid is constructed such that the i-th diagonal consists of i number of nodes. The diagonal-clustered nodal distribution is demonstrated in Figure 1.3.1 (b).



(a): Cross-clustered grid (b): Diagonal-clustered grid

Figure 1.3.1 (a-b). Two non-uniformly distributed nodal grids. To the left: Cross-clustered grid. Right: Diagonal-clustered grid.

A non-uniform nodal grid is constructed by defining the boundary coordinates in the x^i direction, where $i = 1, ..., d$, such that $X^i = \left\{x_1^i, ..., x_j^i\right\}$ where x_j^i is defined by the transformation

$$x_j^i = K + l\sinh(\xi_j^i), \qquad j = 1, ..., N_i. \tag{1.3.11}$$

N_i is the number of nodes in the i-th direction, l determines the amount of clustering and $\xi_j^i \in [\xi_1^i, \xi_{N_i}^i]$ are equidistant points given by

$$\xi_j^i = \sinh^{-1}\left(\frac{x_1^i - K}{l}\right) + j \cdot \Delta\xi^i \tag{1.3.12}$$

with

$$\Delta\xi^i = \frac{(\xi_{N_i}^i - \xi_1^i)}{N_i}. \tag{1.3.13}$$

The end and start points of our equidistant vectors ξ^i in equation (1.3.13) are given by $\xi_{N_i}^i = \sinh^{-1}\left(\frac{x_{N_i}^i - K}{l}\right)$ and $\xi_1^i = \sinh^{-1}\left(\frac{x_1^i - K}{l}\right)$.

Whenever $x_j^i \approx K$ the parameter l ensures that a major part of the evaluation nodes are located around the exercise price as can be seen in Figure 1.3.1 where $K = 1$. In this region, the distances between our created evaluation nodes follows from $\Delta x_j^i \approx l\Delta\xi^i$.

As mentioned earlier in the section, the boundary points are created in the same way for both the cross-clustered and the diagonal-clustered grid. However, the interior points are distributed differently.

1.3.4 Construction of non-uniform partition grids

It was shown in [5] that the accuracy is increased when fewer larger partitions are applied onto the nodal grid, because a larger partition will contain more nodes. However, by the same reasons a grid with fewer larger partitions will lead to more expensive computations. As we expect the dominant error to appear in the region near the exercise price, K, we aim to construct a non-uniform partition grid where large patches are applied in the area where the error is expected to be large, but in order to lower the computational time we apply smaller patches onto the remaining part of the domain. Below follows an illustration of a non-uniform partition framework applied onto a uniformly distributed nodal grid.

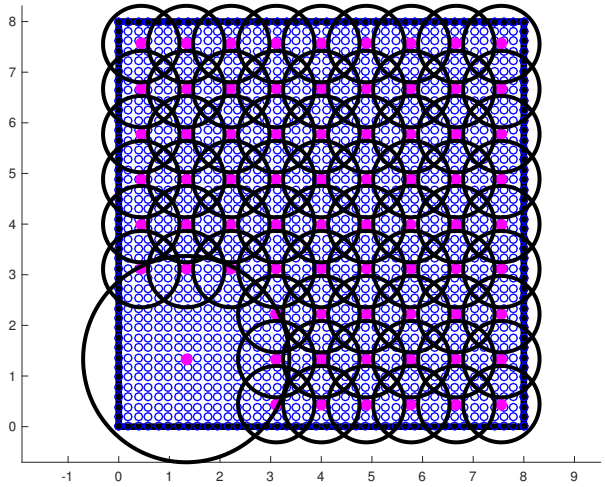


Figure 1.3.2 An example of a non-uniform partition grid. Here circular partitions (black) have been applied on a quadratic domain containing uniformly distributed nodes, represented as blue circles. The pink dots represent the centre point of each partition. The specific distribution depicted here would be suitable to use when pricing a two-dimensional European option with exercise price $K \in (0, 1.5]$, since we in those cases expect a large error at the lower left part of the domain.

2 Numerical results

In this section we present the obtained numerical results. First we motivate our choice of the shape parameter ε, then we show the method's performance in terms of accuracy and efficiency. All computations are made for a two-dimensional European put option which pays no dividend. The exercise price $K = 1$, time of maturity $T = 1$ and the interest rate $r = 0.05$. The volatilities $\sigma_1 = \sigma_2 = 0.2$ have a correlation of $\rho = 0.5$. The multiquadric basis function is used since it is less sensitive to the choice of the shape parameter [5]. In order for our results to reach high accuracy, the computational domain is truncated such that $\Omega = [0 \;\; cK]^2$ where $c = 8$. For the time marching we used the BDF-2 scheme which we know is second order accurate [3] and all computations are made with $N_t = 100$ discretization steps in time. The computed price for the described option type with the mentioned parameters is demonstrated in Figure 2.1. To measure the error, we use a 256×256 finite difference solution as a reference. All computations are made in MATLAB on a laptop with 2.4 GHz Intel Core i5 processor and 8 GB RAM.

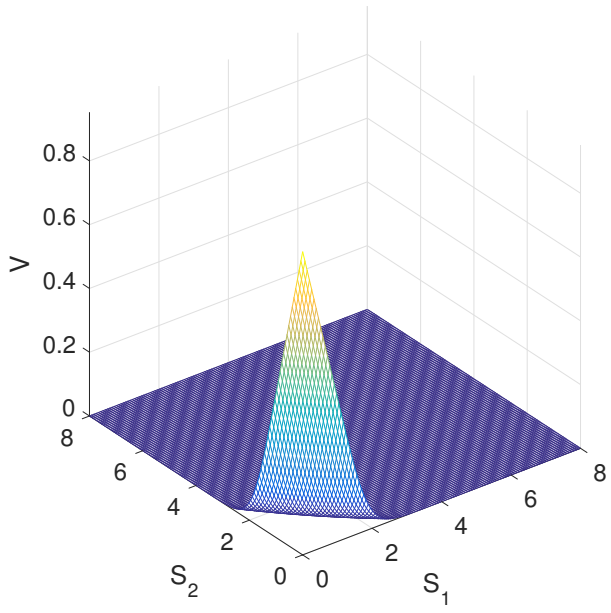


Figure 2.1 The option price today for a European two-dimensional put option. The x- and y-axis represent the underlying assets and the z-axis the option price. For this specific case $\sqrt{N} = 124$ nodes and $P^{1/2} = 9$ partitions, both uniformly distributed, was used.

As we can see in Figure 2.1, the option value is decreasing with increasing asset values. From the exercise line the option value is converging towards $V = 0$ which is the applied boundary condition at the far field boundaries. Since we expect the error to be dominant in the region

near the exercise price we redefine our computational domain so that we are only computing the option value in the lower triangular part of Ω, that is
$\Omega_C = \{\mathbf{s} \in \Omega |\ s_1 + s_2 \leq 8K\}$. This is possible because of the flexibility of RBF methods to domain geometries and since the solution on the upper corner of the square tends to zero it will not have any significant effect on the accuracy. We also define the subspace $\Omega_e = \left\{\mathbf{s} \in \Omega_C |\ \mathbf{s} \in [\frac{K}{3}, \frac{11K}{3}]\right\}$ in which the error, relative to our reference solution, is computed. This is the region which is of most interest from a financial point of view [5]. We denote the number of partitions per dimension as p. Note that $P \neq p^2$ since we use a simplex domain.

2.1 Stability

As mentioned earlier, the choice of the shape parameter ε is crucial to maintain the stability of the method. We have stated the rule of thumb in equation (1.3.10) and therefore we need to determine the constant C. By iterating over a set of constant values the following results was obtained.

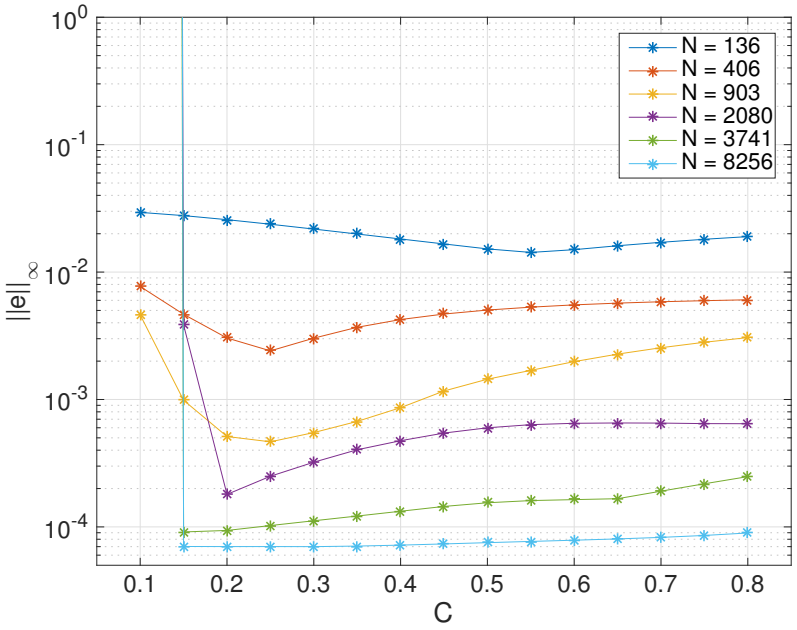


Figure 2.1.1 The error on the y-axis with varying constant values on the x-axis for different problem sizes where N is the total number of nodal points in our simplex domain. The error is computed as the infinite norm of the absolute value between our numerical- and reference solution. The shape parameter is computed according to equation (1.3.10).

The results show that instability occurs for small values of C, which implies a small value of ε. By taking the mean value of the constants generating the lowest errors we obtain $C \approx 0.25$, which we will use for further computations. For the case with uniform nodal grid h is constant and for the case with non-uniform nodal grids we use $h = h_{min}$.

2.2 Accuracy and efficiency

Here we investigate the convergence rates of our method evaluated on both non-uniform nodal and partition grids. Keeping the number of partitions fixed and refine the nodal grid leads to higher accuracy but the problem is more computationally costly to solve. Same thing follows from keeping the number of nodes fixed and increasing the partition radius, which means that each patch contains more nodes. We start by refining the nodal grid while applying a uniform partition framework consisting of $p = 9$ patches.

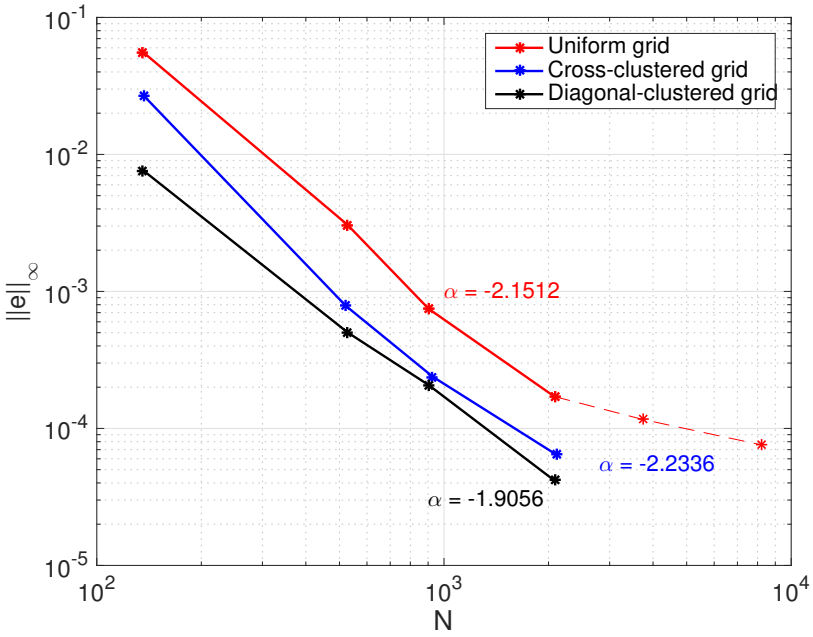


Figure 2.2.1 Infinity norm of the error on the y-axis against a varying number of nodal points on the x-axis for our three different nodal distributions. The figure also presents the slope of the linear approximations to the three curves.

The results show that the error is decreasing when refining the nodal grid. The convergence rates are approximately the same for the three cases until the flattening that occurs for the uniform case for large problem sizes (dashed line). In general for RBF methods we expect a higher convergence rate but the discontinuity in the initial condition limits it to $\alpha \approx 2$ [5]. The mentioned flattening is due to ill-conditioning, which is challenging to overcome completely even with the strategy that we used to select the value of the shape parameter. The diagonal cluster reaches a higher accuracy than the other type of grids and seem to be the best choice in terms of problem size to accuracy.

Further we investigate the computational time for the three different types of grids. In Figure 2.2.2 we present the CPU time while varying the problem size using the same applied partition framework as above.

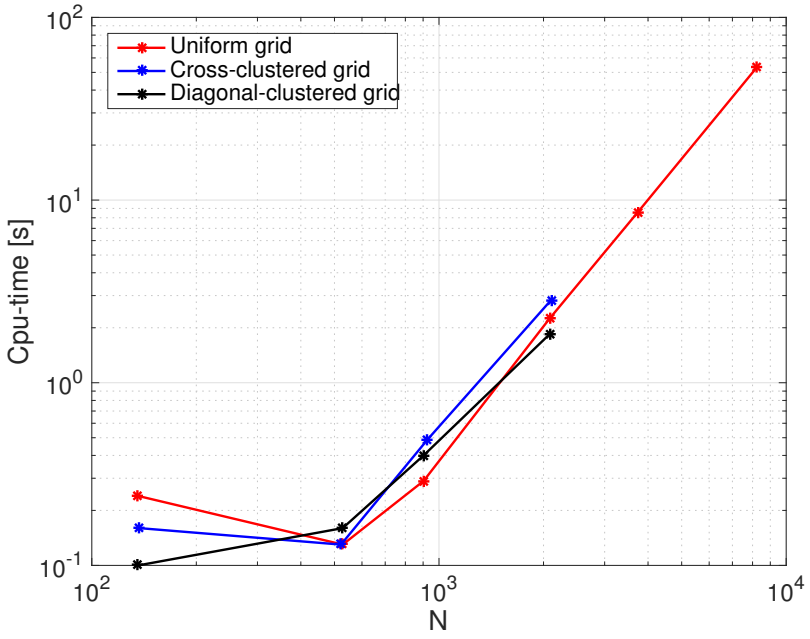


Figure 2.2.2 CPU time on the y-axis against the problem size on the x-axis for our three different nodal distributions.

The results show that the CPU time for the computations on the three nodal grids is not differing substantially. We know that an increasing amount of nodes in a partition leads to more computations because of the linear system that becomes more dense [5]. For the case with non-uniform grids the partitions located in the clustered region contains more nodes than the corresponding partitions for the uniformly distributed grid. On the other hand the remaining partitions contain fewer nodes and therefore the solution is computationally cheaper to compute in these regions.

We are interested to investigate the relation between computational time and accuracy and, therefore, we combine our results from Figures 2.2.1 and 2.2.2.

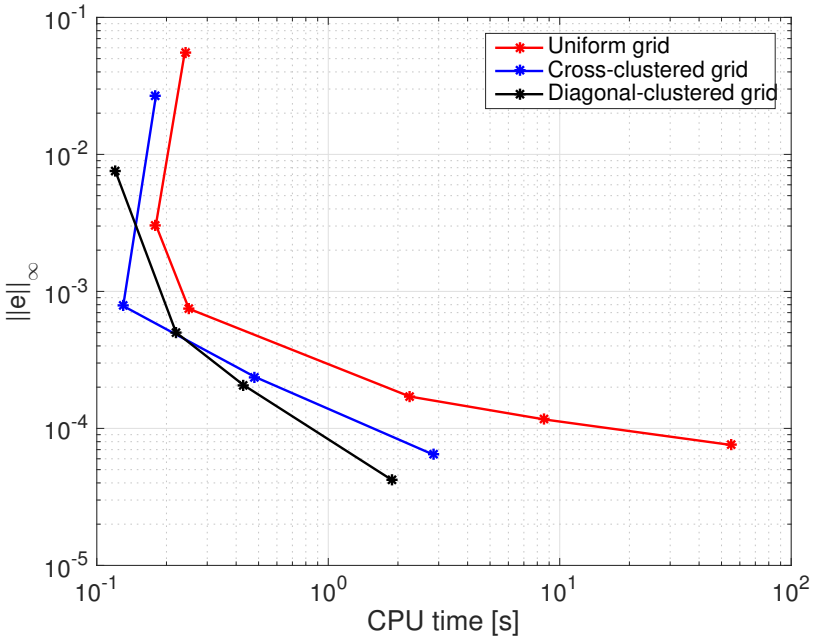


Figure 2.2.3 Infinity norm of the error on the y-axis against the CPU time on the x-axis. The results was obtained by varying the problem size.

The results show that a non-uniform grid seems preferable, especially when requiring highly accurate results. In order to reach results accurate to $||e||_\infty \leq 10^{-4}$, which is considered a good accuracy from a financial point of view, a significant amount of computational cost can be reduced by using non-uniformly distributed grids.

Table 1: CPU time and problem size required for RBF-PUM to reach a specified accuracy when evaluated on the three different types of nodal grids.

$\|\|e\|\|_\infty$	Uniform		Cross-clustered		Diagonal-clustered	
	CPU (sec)	N	CPU (sec)	N	CPU (sec)	N
1e-02	0.22	300	0.24	232	0.19	136
1e-03	0.34	820	0.25	644	0.21	300
5e-04	0.36	990	0.86	1153	0.29	528
1e-04	10.48	4095	3.02	2105	0.73	1176
7e-05	78.17	9591	3.02	2105	1.15	1596

In order to reach an error of $||e||_\infty \leq 10^{-4}$ the CPU time is reduced by 71.2% by using a cross-clustered grid and 93.0% by using a diagonal-clustered grid, compared to the uniform case. The required number of evaluation nodes in the system is reduced by 48.6% and 71.3%. In order to reach an error $||e||_\infty \leq 7 \times 10^{-5}$, which is the highest accuracy obtained for this

case, the CPU time and problem size are reduced by 96.1% and 78.1% respectively for the cross-clustered case. For the diagonal case the CPU time is reduced by 98.5% and the problem size with 83.4%.

Further we investigate the performance of the RBF-PUM when a non-uniform partition grid is applied. The simulations are made with a uniformly distributed nodal grid which is refined. The applied partition grid is demonstrated in Figure 1.2.3 where nine smaller patches, located in the region where we expect the error to be dominant have, been replaced with one larger patch. Figures 2.2.4 - 2.2.6 show the outcome from RBF-PUM using this framework. The outcome is compared to the uniform case with $p = 9$ and $p = 3$.

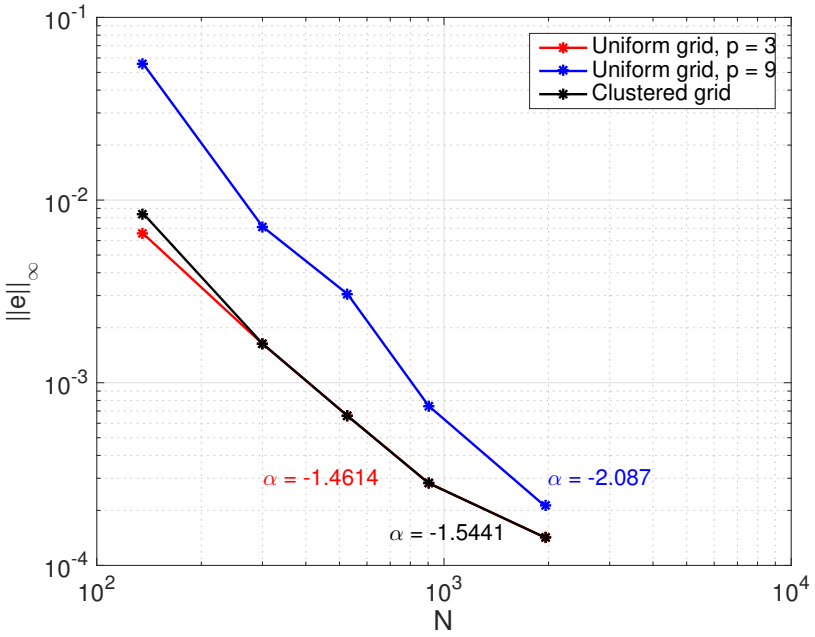


Figure 2.2.4 Infinity norm of the error on the y-axis against the problem on the x-axis. The blue curve is the outcome from a uniform partition grid with $p = 9$ patches, the red curve for $p = 3$ patches and the black curve for the non-uniform case described above.

The results show that the error generated using a clustered grid is approximately equal to that when using a uniform grid with $p = 3$. As expected, the framework with the highest amount of partitions generates the lowest accuracy. Larger problem sizes than demonstrated were not operable for the clustered- and $p = 3$ case due to computational limitations. Figure 2.2.5 shows the computational time when refining the nodal grid.

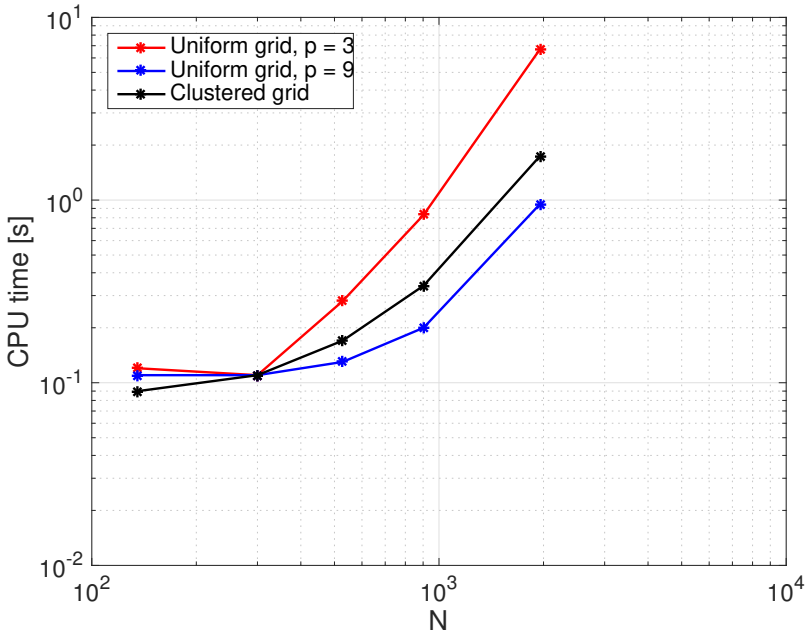


Figure 2.2.5 CPU time on the y-axis against the problem size on the x-axis.

As expected, using a grid with fewer and larger partitions is most expensive while the uniform grid with small partitions is the cheapest. It is notable that the clustered grid does not differ much from the case with $p = 9$ in terms of CPU time. The solution on the larger patch in the clustered case is more expensive to compute since it contain more nodes. However, substituting nine patches by one leads to fewer local approximations to compute. In Figure 2.2.6 we combine the results from Figures 2.2.4 and 2.2.5.

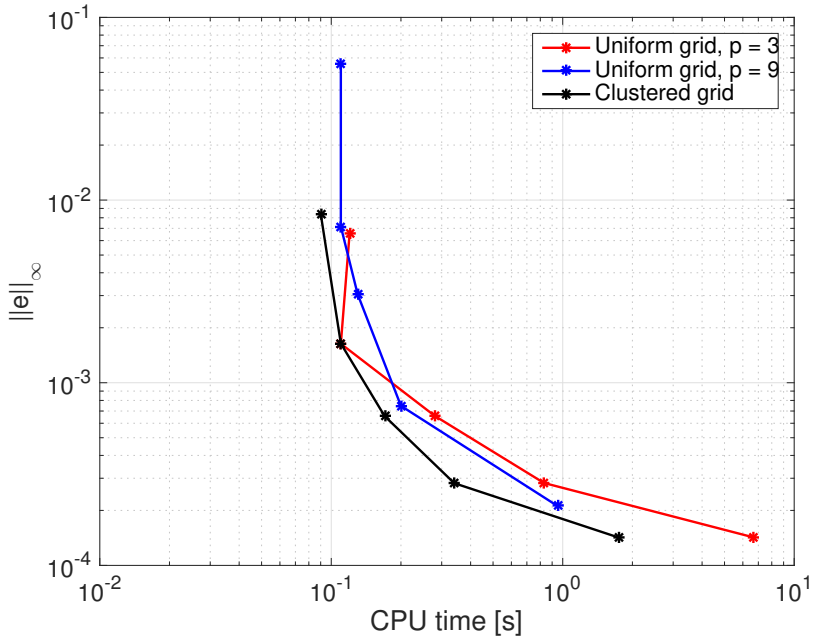


Figure 2.2.6 Infinity error norm on the y-axis against the CPU time on the x-axis. The results was obtained by varying the problem size.

Keeping a large patch in the exercise price region and substitute those located in the remaining part of the domain to smaller patches does not affect the accuracy of the RBF-PUM markedly. It does however lower the computational cost for large problem sizes. In contrast, comparing the clustered case with the uniform containing $p = 9$ partitions we observe a higher accuracy, especially for small problem sizes, but the computations are more expensive.

Table 2: CPU time and problem size required for the RBF-PUM to reach a specified accuracy when evaluated on the three different type of partition grids.

$\|\|e\|\|_\infty$	Uniform, $p = 3$		Uniform, $p = 9$		Clustered	
	CPU (sec)	N	CPU (sec)	N	CPU (sec)	N
1e-2	0.10	105	0.20	300	0.20	136
5e-3	0.12	171	0.28	465	0.20	253
1e-3	0.21	325	0.30	820	0.27	406
5e-4	0.47	595	0.38	990	0.28	595
2e-4	6.91	1953	10.05	4095	1.92	1596

In order to reach an accuracy of $||e||_\infty \leq 2 \times 10^{-4}$, which was the highest reachable accuracy due to computational limitations, the clustered grid reduces the CPU time and the problem

size by 80.9% and 61.0% compared to the case with $p = 9$. For the case with $p = 3$ the CPU time and the problem size is reduced by 72.2% and 18.3% respectively by using a clustered framework.

3 Conclusion

In order to avoid instabilities, the shape of the applied basis functions has to be set with respect to the number of evaluation nodes in the system. The value of the shape parameter follows from $\epsilon = \frac{C}{h}$ where the constant, C, is computed as $C = 0.25$ and h is the fill distance. In theory instabilities occur when ε is too small, this is also confirmed by our obtained results. Hence, h is set to h_{min} for the case with non-uniformed distributed nodal grids.

RBF-PUM was evaluated on two different types of clustered nodal grids; cross-clustered and diagonal-clustered. Due to a discontinuity in the first derivative of the initial condition at the exercise price, which leads to a dominant error, the non-uniformed grids was focused in that region to resolve the issue. Our results showed that using a non-uniform grid is to prefer over a uniform distribution when focusing on CPU time to accuracy. A higher accuracy can be reached by rearranging the location of the evaluation nodes and the CPU time is lowered as a result of this. In order to reach an accuracy of $||e||_\infty \leq 10^{-4}$ the CPU time is reduced by 71.2% and 93.0% for the cross-clustered and diagonal-clustered respectively, compared to a uniform distribution.

Further, the performance is evaluated when a non-uniform partition grid is applied onto the computational domain. An increasing amount of evaluation points in a partition leads to higher accuracy but more computational expense. Therefore, we constructed a partition grid with a larger sized patch covering the region near the exercise price and smaller sized patches elsewhere. Our obtained results show that in order to reach an accuracy of $||e||_\infty \leq 2 \times 10^{-4}$ the CPU time is reduced by 80.9% by using this type of clustered grid compared to a uniform grid of the smaller sized partition. Compared to a uniform grid of the larger sized partition the CPU time is reduced by 72.2%.

4 Discussion

Since both a non-uniform partition and nodal grid improved the performance of RBF-PUM in terms of CPU time to accuracy it would probably be preferable to combine the two approaches. However, due to computational limitations on the available computer resources when too many nodes are contained in one single partition this was not possible to simulate. Although, for future work one could construct the partition grid such that each patch contains approximately the same number of nodes and combine this with a non-uniform nodal grid. This would generate a partition grid with smaller patches in the region where the non-uniform nodal grid is focused. The dense nodal grid would improve the accuracy while

the smaller sized partitions would decrease the accuracy but also the computational time.

In Figure 2.2.1 we notice a flattening in the error convergence for large sized uniform nodal distributions. We believe this is due to ill-conditioning in the basis function matrix meaning that the shape parameter is not optimal. The constant from which we determine the value of the shape parameter is computed as a mean value of the constant values that gave the highest accuracy for different problem sizes. Perhaps this mean value can be weighted or the constant value can be varying with the problem size in order to prevent this ill-conditioning completely.

References

[1] Investopedia, "Vanilla Option", (Retrieved 9 December 2016) [Link]

[2] Investopedia, "Exotic Option", (Retrieved 9 December 2016) [Link]

[3] R. Seydel, Tools for Computational Finance, Fifth Edition, London: Springer, (2012) [Link]

[4] T. Björk, Arbitrage Theory in Continuous Time, Third Edition, Oxford University Press, (2009) [Link]

[5] V. Shcherbakov and E. Larsson, Radial Basis Function Partition of Unity Methods for Pricing Vanilla Basket Options, Computers and Mathematics with Applications, Vol. 71, (2016), pp. 185–200 [Link]

[6] A. Safdari-Vaighani, A. Heryudono, and E. Larsson, A Radial Basis Function Partition of Unity Collocation Method for Convection-Diffusion Equations Arising in Financial Applications, Journal of Scientific Computing, (2015), pp. 1-27 [Link]

[7] D. Shepard, A two-dimensional interpolation function for irregularly-spaced data, ACM '68 Proceeding of the 1968rd ACM national conference, (1968), pp. 517-524 [Link]

[8] H. Wendland, Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree, Advances in Computational Mathematics, Vol. 4, (1995), pp.389-396 [Link]

[9] K. J. In 'T Hout, and S. Foulon, ADI Finite Difference schemes for Option Pricing in the Heston Model with Correlation, International Journal of Numerical Analysis and Modeling, Vol. 7, (2009), pp. 303-320 [Link]