Multilevel sparse grid kernels collocation with radial basis functions for elliptic and parabolic problems

Thesis submitted for the degree of

Doctor of Philosophy

at the

University of Leicester

by

Yangzhang Zhao

Department of Mathematics

University of Leicester

August 2016

"Everybody should have a dream, what if that dream comes true."

Jack Ma – Founder and Chairman of Alibaba Group

Abstract

Radial basis functions (RBFs) are well-known for the ease implementation as they are the mesh-free method [31, 37, 71, 72]. In this thesis, we modify the multilevel sparse grid kernel interpolation (MuSIK) algorithm proposed in [48] for use in Kansa's collocation method (referred to as MuSIK-C) to solve elliptic and parabolic problems. The *curse of dimensionality* is a significant challenge in high dimension approximation. A full grid collocation method requires $\mathcal{O}(N^d)$ nodal points to construct an approximation; here N is the number of nodes in one direction and d means the dimension. However, the sparse grid collocation method in this thesis only demand $\mathcal{O}(N \log^{d-1}(N))$ nodes. We save much more memory cost using sparse grids and obtain a good performance as using full grids. Moreover, the combination technique [20, 54] allows the sparse grid collocation method to be parallelised. When solving parabolic problems, we follow Myers et al.'s suggestion in [90] to use the space-time method, considering time as one spatial dimension. If we apply sparse grids in the spatial dimensions and use time-stepping, we still need $\mathcal{O}(N^2 \log^{d-1}(N))$ nodes. However, if we use the space-time method, the total number of nodes is $\mathcal{O}(N \log^d(N))$.

In this thesis, we always compare the performance of multiquadric (MQ) basis function and the Gaussian basis function. In all experiments, we observe that the collocation method using the Gaussian with scaling shape parameters does not converge. Meanwhile, in Chapter 3, there is an experiment to show that the space-time method with MQ has a similar convergence rate as a time-stepping method using MQ in option pricing. From the numerical experiments in Chapter 4, MuSIK-C using MQ and the Gaussian always give more rapid convergence and high accuracy especially in four dimensions ($\mathbb{T} \times \mathbb{R}^3$) for PDEs with smooth conditions. Compared to some recently proposed mesh-based methods, MuSIK-C shows similar performance in low dimension situation and better approximation in high dimension. In Chapter 5, we combine the Method of Lines (MOL) and our MuSIK-C to obtain good convergence in pricing one asset European option and the Margrabe option, that have non-smooth initial conditions.

Acknowledgements

It is an unforgettable memory to live and study in Leicester for five years. I would like to express my sincere gratitude to everyone who gives me help and support during my Ph.D. life. Without you, I could never ever achieve that.

First and foremost, I would like to express utmost gratitude to my supervisor, instructor and mentor, Prof. Jeremy Levesley. Thanks to his support, patience and immense helpful advice, I could persevered in my Ph.D. study and related research. Prof. Jeremy Levesley also provided plenty of wise advices for me in other fields. I could not imagine a person who could be a better supervisor for me than Prof. Jeremy Levesley.

Secondly, I am also grateful to all the staff members of the mathematics department and the administration of the University for their help. I would like to thank Prof. Emmanuil (Manolis) Georgoulis, Dr. Andrea Cangiani, Dr. Bo Wang and Dr. Steven Hales for their guidance, suggestions on my studies. They are great source and give me endless support in my research.

Finally, I would like to thank every kindly friend in MAB; Daniel, Juxi, Masha, Matt, Ruhao, Sam, Yanshan and so on. It's a great pleasure to work with them. I would like to thank especially Dr. Zhaonan Dong, known as Peter Dong, and Dr. Qi Zhang. Peter is always ready to deliver helpful and inspiring bits of advice wholeheartedly at request since we are aware each other. To some extents, Qi is my co-researcher, and he is helpful and motivative for all the time. It's wonderful to have Peter and Qi as my close friends.

Last but not least, my warmest thanks goes to my beloved soul mate Dr. Yun Song for her patience, support, her sense of understanding and help throughout my research. The hard time she lived with me during my research in Leicester means a lot to me and she owes much more than thanks.

Contents

List of Figures

List of Tables

Abbreviations

ACBF = Approximate Cardinal Basis Function
ATPBFs = Anisotropic Tensor Product Basis Functions
BDF = Backward Differential Formula
BUP = Boundary Update Procedure
CSRBFs = Compactly Supported Radial Basis Functions
FEM = Finite Element Method
FDM = Finite Difference Method
FVM = Finite Volume Method
GBM = Geometric Brownian motion
GSRBFs = Globally Supported Radial Basis Functions
IgA = Isogeometric Analysis
IMQ = Inverse Multiquadric
IPDG = interior penalty discontinuous Galerkin
MLRBF-C= Multilevel Full Grid RBF Collocation
MOL = Method of Lines
MQ = Multiquadric
MuSIK = Multilevel Sparse Grid Kernel Interpolation
MuSIK-C= Multilevel Sparse Grid Kernel Collocation
ODEs = Ordinary Differential Equations
PD = Positive Definite
PDEs = Partial Differential Equations
PSM = Pseudospectral Method
PUM = Partition of Unity
RBFs = Radial Basis Functions
RBF-C = RBF collocation
RE = Richardson Extrapolation
RMS = Root Mean Square Error
SIK = Sparse Grid Kernel Interpolation
SIK-C = Sparse Grid Kernel Collocation
TPS = Thin Plate Splines
VSK = Variably Scaled Kernels

To my parents

Chapter 1

Introduction

1.1 Background

Numerical approximations of partial differential equations (PDEs) have been widely utilised in diverse subject areas, such as fluid dynamics, electromagnetism, material science, astrophysics and financial markets. There are a variety of numerical methods for solving PDEs. The most well-known is perhaps the finite difference method (FDM) which utilises finite difference equations to approximate differential equations [52]. The computational domain is usually divided into discretization grids. The discretization results in a system of equations of the variable at nodal points, and once a solution is found, then we have a discrete representation of the solution at each nodal point. The FDM is easy to understand when the domain is a rectangle. However, when the shape of domain is complex, it is difficult to implement the FDM without employing other techniques. The finite element method (FEM) is another well-known numerical technique for solving PDEs and it can cope with geometric variation better than the FDM [66, 104]. The computational domain is divided into smaller domains (finite elements) and the solution in each element is constructed from the basis functions. The number of variables or dimensions might reach hundreds or even thousands in a realistic problem. The above two methods require some pre-decided meshes, however mesh construction is difficult in high dimensional problems. In view of this, it is very

significant to devise efficient numerical methods for high dimensional problems. In this thesis, our purpose is to examine the performance of a new method for solving the linear PDEs, especially in high-dimensional situations.

Mesh-free methods are believed to have a huge advantage when spatial dimension is increasing. They just require data points located in the domain instead of being concerned with the connectivity of the nodes. Radial basis functions (RBFs) are one kind of such mesh-free method. Hardy firstly proposed the multiquadric (MQ) RBF [56] in 1971. Hardy also published a review of the development of the MQ RBF from 1968 to 1988 in [57]. In 1982, Franke examined various methods to solve the scattered data interpolation problem in the plain and concluded that Hardy's MQ was the best in [41]. Later, Micchelli demonstrated the interpolation matrix of many RBFs (including MQ) are invertible [89]. Further, Madych and Nelson [87] stated the spectral convergence rate of MQ interpolation in 1992. In the literature [71, 72], Kansa first utilised the MQ RBF to solve PDEs, and this approach is called Kansa's method which is an unsymmetric collocation method. Afterwards, using the MQ became quite popular in PDE problems such as [18, 35, 63, 100]. Later, for initial and boundary conditions, Hon et al. extended this method to solve the engineering biphasic model with the nonlinear initial and boundary conditions in [63]. Fornberg et al. applied variate boundary treatments (edge enhancement techniques) to RBFs to investigate the problem of RBF approximations at the edge of an interval in both one and two dimensions [17]. Furthermore, examples of option pricing (American and European option) with RBFs (Global RBFs and Quasi-RBFs) have been demonstrated by Hon in [61, 62], where the Boundary Updated Procedure (BUP) technique is applied to capture the free boundary condition problem in the American option. Hon also investigated the theoretical convergence of RBFs for the *Black-Scholes equation* [65]. The attractive factors of MQ are not only its suitability in high dimension and spectral convergence but also its infinite differentiability. Other RBFs such as the Gaussian RBFs have the same properties, see [31, 38, 64]. Infinitely differentiable RBFs are most popular in applications. However, it is not clear that how to choose the type of RBF for one particular problem.

Besides the Kansa method, Fasshauer [27] produced a symmetric matrix from the PDE system by applying the Hermite interpolation property of RBFs. Wu adopted the RBF Hermite-Birkhoff interpolation and proved the convergence of this approach in [122]. Later in 1998, Wu [124] also proved that this method keeps its convergence when solving PDE problems. In the same year, Schaback and Franke [43] also provided their convergence order estimation. In this thesis we use the Kansa collocation method [71, 72] which was mentioned above. Some others use the symmetric RBF method [27, 37].

1.1.1 Radial basis functions and collocation

Let $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \subseteq \mathbb{R}^d$ be a set of scattered data sites, the RBF interpolant $\hat{u}$ which will be described in detail in Chapter 2 takes the following form:

$$\hat{u}(\mathbf{x}) = \sum_{i=1}^{N} \lambda_i \phi(\|\mathbf{x} - \mathbf{x}_i\|), \quad \mathbf{x} \in \mathbb{R}^d, \tag{1.1}$$

where the norm $\|\cdot\| : \mathbb{R}^d \to \mathbb{R}_+$ normally stands for the Euclidean norm, $\boldsymbol{\lambda} = \{\lambda_1, \ldots, \lambda_N\}$ is coefficient vector and $\phi : \mathbb{R}_+ \to \mathbb{R}$ is the RBF. For an interpolation problem to a known or unknown function f, we need the measurements at the data site $\{f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots, f(\mathbf{x}_N)\}$ to make the interpolant satisfy

$$\hat{u}(\mathbf{x}_i) = f(\mathbf{x}_i), \quad i = 1, 2, \ldots, N.$$

For the boundary value problem for partial differential equations:

$$\begin{aligned} \mathcal{L}u &= f \quad in\ \Omega, \\ u &= g \quad on\ \partial\Omega, \end{aligned}$$

where $\mathcal{L}$ is a differential operator, f and g are prescribed functions, Ω is the domain, and $\partial\Omega$ denotes the boundary of domain. The interpolant $\hat{u}$ matches the

system at the given data:

$$\begin{aligned} \mathcal{L}\hat{u}(\mathbf{x}) &= f(\mathbf{x}), \quad \mathbf{x} \in \mathbf{X} \cap \Omega, \\ \hat{u}(\mathbf{x}) &= g(\mathbf{x}), \quad \mathbf{x} \in \mathbf{X} \cap \partial\Omega. \end{aligned}$$

The corresponding matrix system is

$$\begin{bmatrix} \mathcal{L}\phi(\|\mathbf{x}_1 - \mathbf{x}_1\|) & \mathcal{L}\phi(\|\mathbf{x}_1 - \mathbf{x}_2\|) & \cdots & \mathcal{L}\phi(\|\mathbf{x}_1 - \mathbf{x}_N\|) \\ \mathcal{L}\phi(\|\mathbf{x}_2 - \mathbf{x}_1\|) & \mathcal{L}\phi(\|\mathbf{x}_2 - \mathbf{x}_2\|) & \cdots & \mathcal{L}\phi(\|\mathbf{x}_2 - \mathbf{x}_N\|) \\ \vdots & \vdots & \cdots & \vdots \\ \mathcal{L}\phi(\|\mathbf{x}_n - \mathbf{x}_1\|) & \mathcal{L}\phi(\|\mathbf{x}_n - \mathbf{x}_2\|) & \cdots & \mathcal{L}\phi(\|\mathbf{x}_n - \mathbf{x}_N\|) \\ \phi(\|\mathbf{x}_{n+1} - \mathbf{x}_1\|) & \phi(\|\mathbf{x}_{n+1} - \mathbf{x}_2\|) & \cdots & \phi(\|\mathbf{x}_{n+1} - \mathbf{x}_N\|) \\ \vdots & \vdots & \cdots & \vdots \\ \phi(\|\mathbf{x}_N - \mathbf{x}_1\|) & \phi(\|\mathbf{x}_N - \mathbf{x}_2\|) & \cdots & \phi(\|\mathbf{x}_N - \mathbf{x}_N\|) \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_n \\ \lambda_{n+1} \\ \vdots \\ \lambda_N \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \\ g_{n+1} \\ \vdots \\ g_N \end{bmatrix} \tag{1.2}$$

here $\mathbf{x}_i \in \mathbf{X} \cap \Omega$ for $i = 1, 2, \ldots, n$ and $\mathbf{x}_j \in \mathbf{X} \cap \partial\Omega$ for $j = n+1, \ldots, N$.

Since RBFs were proposed almost five decades ago, RBFs have had a fast development and been widely used in many areas. There are lots of publications mentioned the numerous advantages of RBFs, for example, [10, 24, 119]. In [37], Larsson and Fornberg concluded that RBFs are more accurate than the standard second-order finite difference method (FDM) and the Fourier-Chebyshev pseudospectral method (PSM) in elliptic problems. Buhmann et al. and Madych et al. proved that infinitely smooth RBFs have spectral orders of convergence (faster than any polynomial order) in [9, 87]. In [15], Cecil et al. illustrated the example of solving the Hamilton Jacobi equation in two to four dimensions by using RBFs. Alternatively, finite difference (FDM), finite element (FEM), finite volume (FVM) can also be used to solve this problem [66, 107]. Pena's eight step "cooking recipe" [95] showed that RBFs can be easily applied to a range of different option pricing problems. The RBFs method only requires an arbitrary set of nodes which is extremely useful for high-dimensional implementation, such as an American basket option [32, 75]. To sum up, the profits are high-order accuracy, applicability in high dimension, easy to implement and mesh free.

Kansa summarised the achievements and challenges concerning RBFs in Cheng and Brebbia's book [19]. In the article, he classifies RBFs into three classes:

- Compactly supported radial basis functions (CSRBFs), see Wendland [117] and Wu [123]; for example

$$\phi(r) = (1-r)_+^4(4r+1), \quad (2DRBF, \quad \text{Wendland}),$$
$$\phi(r) = (1-r)_+^5(5r^4+25r^3+48r^2+40r+8), \quad (2DRBF, \quad \text{Wu}).$$

  Here the notation $(1-r)_+ = \max\{1-r, 0\}$, hence the above CSRBFs have a support parameter $0 \leq r \leq 1$.

- Polyharmonic splines and thin plate splines (TPS):

$$\begin{aligned} \phi(r) &= r^k, \quad k = 1, 3, 5, \ldots, \quad \text{(polyharmonic)}, \\ \phi(r) &= r^k \log(r), \quad k = 2, 4, 6, \ldots, \quad \text{(TPS)}. \end{aligned}$$

- C^∞ RBF splines can be any transcendental univariate function with a local scale factor (also called shape parameter) c. Some examples as we mentioned:

$$\begin{aligned} \phi(r) &= \sqrt[\beta]{r^2+c^2}, \quad \text{(MQ)}, \\ \phi(r) &= \exp\left(-\frac{r^2}{c^2}\right), \quad \text{(Gaussian)}. \end{aligned}$$

In the above three classifications, only C^∞ RBFs have the shape parameter c. The operator can change the shape from flat to peak by employing different values of c (more details in Chapter 2). According to Schaback's uncertainty principle [101]: "Either one goes for a small error and gets a bad sensitivity, or one wants a stable algorithm and has to take a comparatively large error." There is not a perfect approach and theory to optimise choice of shape parameters currently. However, many researchers have done some remarkable procedures to solve the problem, for instance, [7, 31, 33, 38, 39, 40, 79, 80, 81].

Although infinitely smooth RBFs are widely used to implement for high dimensional problems, the computational cost grows fast when the dimension is increasing.

Moreover, the density of data will raise up at a high speed so that will cause an ill-conditioning problem. Even if we don't take care of the trouble in ill-conditioning, the long running time to solve the system matrix is prohibitive. In 1999, the authors in [4] pointed out it was not appropriate (at that time) to utilise direct method when the size of data is bigger than ten thousand. Current practical applications often require tens of thousands or even millions of data sites. The requirement of large data forces the researchers to develop strategies to overcome the difficulties. Utilising CSRBFs [117, 123] is an easy way to handle computational cost problem. As a result, we will lose spectral convergence. Some of other suggested methods to address the difficulties are domain decomposition method [23, 77], multilevel method [28, 55, 70], preconditioning strategies [4, 8, 34], sparse grid method [46, 51], RBF partition of unity (PUM) method [59, 60, 105]. In the next two subsections, we would like to give a brief introduction to the multilevel method and sparse grid method that are employed in this thesis.

1.1.2 Multilevel method

It is a general approach to utilise the multilevel method to accelerate convergence rate for scattered data, see [28, 48, 52, 69, 70, 91, 93]. The basic idea of the multilevel method is the hierarchical decomposition of the data sites. For instance, given a set of data $\mathbf{X} \subseteq \Omega \subseteq \mathbb{R}^d$, the first requirement is to form a nested sequence of subsets

$$\mathbf{X}_1 \subseteq \mathbf{X}_2 \subseteq \cdots \subseteq \mathbf{X}.$$

Furthermore, Wendland suggested [119] that the separation distance (see Definition 2.5) $q_{\mathbf{X}}$ and fill distance (Definition 2.4) $h_{\mathbf{X},\Omega}$ should follow:

$$\begin{aligned} q_{\mathbf{X}_{j+1}} &= \frac{1}{2} q_{\mathbf{X}_{j+1}}, \\ h_{\mathbf{X}_{j+1},\Omega} &= \frac{1}{2} h_{\mathbf{X}_j,\Omega}. \end{aligned}$$

Regardless of whether it is an interpolation problem or to solve PDE system, the approximations $\hat{u}^l$ at the lth level satisfies the residual equation

$$\hat{u}^l|_{\mathbf{X}_l} = \left(f - \sum_{j=0}^{l-1} \hat{u}^j \right) |_{\mathbf{X}_l},$$

where $\hat{u}^0 \equiv 0$, f is the target function.

It is also demonstrated in [119] that there are two restrictions in utilising the multilevel method. The first is that we must use different basis functions at every level l. This is because the data sites are nested

$$\mathbf{X}_l \subseteq \mathbf{X}_{l+1}.$$

If the basis functions are the same, the approximation spaces will also be nested

$$\text{span}\{\phi_l\left(\mathbf{x}-\mathbf{y}\right), \mathbf{y} \in \mathbf{X}_l\} \subseteq \text{span}\{\phi_{l+1}\left(\mathbf{x}-\mathbf{y}\right), \mathbf{y} \in \mathbf{X}_{l+1}\}.$$

Therefore, using the multilevel method will not improve the approximations. In the Example 1.1, we compare estimation results under utilising the same basis functions and applying different basis functions. The second restriction is that we only apply one cycle of the algorithm and any further cycle would not improve our approximations as a result of that the data sets we used are nested.

Example 1.1. *For a simple one dimension interpolation problem, suppose target function is* $f(x) = \sin(\pi x)$ *on the interval* $[0, 1]$.

The numerical results are obtained with multiquadric RBF. In the following two tables, N stands for the number of uniformly spaced nodes, l_∞ is max absolute error. From Table 1.1 and Table 1.2, we can recognise that using scaling kernels is crucial for multilevel method if we do not change basis function.

Level	N	c	Direct l_∞	Multilevel l_∞
1	3	0.05	1.8e-1	1.8e-1
2	5	0.05	4.3e-2	4.3e-2
3	9	0.05	2.1e-2	2.1e-2
4	17	0.05	8.7e-3	8.7e-3
5	33	0.05	2.1e-3	2.1e-3

TABLE 1.1: The performance of multilevel method and direct method with $c = 0.05$.

Level	N	c	Direct l_∞	Multilevel l_∞
1	3	0.8	2.8e-3	2.8e-3
2	5	0.4	1.1e-2	1.6e-3
3	9	0.2	7.3e-3	6.5e-4
4	17	0.1	4.0e-3	1.8e-4
5	33	0.05	2.1e-3	4.5e-5

TABLE 1.2: The performance of multilevel method and direct method with scaling c.

1.1.3 Sparse grid method

The sparse grid is the core part of this thesis, and there will be detailed explanation in Chapter 4. Here, we just give some common background. The sparse grid method was proposed by Zenger [126] in 1991 in order to deal with PDE problems in high-dimensional situations. It resulted from the Smolyak algorithm [106] for numerical integration. The fundamentals of sparse grid techniques have a close relationship with the hyperbolic cross [1, 112], the Boolean method [20] and the discrete blending method [3]. The sparse grid method relies on a tensor product hierarchical basis construction, which can reduce the degrees of freedom from $\mathcal{O}(N^d)$ to $\mathcal{O}(N \log^{d-1}(N))$ for d-dimensional problems without compromising much on accuracy.

In 1992, Griebel et al. [54] proposed an alternative representation of the sparse grid, which is called the combination technique. This technique separates a sparse grid into several coarser sub-grids that are directionally uniform. Thus, partial solutions can be estimated on every sub-grid and be linearly combined to construct the final approximation. As the sparse grid is divided into much coarser grids, it is beneficial for reducing memory requirement. For example, we can solve two

dimensional elliptic problem with 13,313 nodes or three dimensions with 21,249 centres using the sparse grid on a laptop while it is not possible to handle similar size of data sites on the full grid. On the other hand, the processes to achieve approximations on sub-grids are independent so that it is convenient to apply parallel method on sparse grid combination technique.

Currently, sparse grid techniques are widely used in interpolation and approximation. They have been incorporated in collocation methods for high-dimensional stochastic differential equations [83, 92, 125], Galerkin and finite element methods [11, 12, 13, 45, 104, 116], finite difference methods [52], finite volume methods [58] for high-dimensional PDEs.

1.1.4 Space-time method

When we try to solve parabolic equations (see Definition 3.2), one normal way is to consider the time dimension and spatial dimensions separately, like the Method of Lines (MOL) which is introduced in Chapter 3; see [62, 95, 98]. In 2002, Myers et al. [90] proposed treating time as one spatial dimension which is called the space-time method in RBF field. Recently, this method is becoming a standard approach, see e.g. [2, 53, 74, 94, 111, 114].

Using previous algorithms to solve time dependent PDEs with RBFs, particular discretization in the time direction is essential. Once the discretization nodes in the time direction are determined, we only have approximations at those time nodes. If we would like to have approximations outside the time discretization grid, we have to relocate the time nodes and resolve the problem. Having an efficient time and spatial grid is crucial to the approximation accuracy in previous major algorithms. However, it is not clear which direction has more effect on the estimation. The space-time method can avoid the above disadvantages by applying RBFs on all directions. Because we don't need to balance the influences from different methods in the time direction and spatial directions. In Chapter 3, we present an example to show the performance of the MOL and the space-time method. When utilising a sparse grid, one can reduce the complexity. For example,

solving a $\mathbb{T} \times \mathbb{R}^d$ PDE as usual, we may need $\mathcal{O}(N^2 \log^{d-1}(N))$ operations, where N is nodes number in each direction. However, if consider time as one dimension, the complexity is $\mathcal{O}(N \log^d(N))$. In this thesis, we use the space-time method to solve parabolic problems.

1.1.5 Black-Scholes equation

Options are contracts that offer owners the right, but not the obligation to buy (called call option) or sell (called put option) underlying assets or instruments at a specified strike price. One option has an expiry date, also known as maturity date. The holder can exercise his option before the expiry date or on the date, depending on the form of the option. There are many styles of options such as European option, American option, Spread option, Barrier option and so on. As many options and financial derivatives are concerned about multiple assets, solving high dimensional problems is significant in financial framework. The time when holder would like to trade depends on the option style. For instance, an American option holder can exercise the option at any time before maturity. However, a European call option holder only can exercise at the expiry time T, which is determined when signing the contract. The corresponding payoff function for one asset European call option at time T is

$$P = (S - E)_+ = \max\{S - E, 0\}, \tag{1.3}$$

where E is the exercise price, S is the stock price at time T.

The *Black-Scholes equation* [6] is very famous in financial mathematics to price options. It is widely used in different kinds of options pricing by matching variety of boundary conditions. Because the *Black-Scholes equation* is also one kind of parabolic equations, the normal approach of option pricing is to solve the equation is by time stepping [13, 16, 32, 61, 62, 75, 95]. However, there are also some research on option pricing using the space-time method. For instance, Urschel [114] introduced an adaptive space-time multigrid method for the pricing of barrier options in 2013. Urschel stated one disadvantage of the space-time method that it

generates a large system matrix due to the addition of one dimension. Therefore, we will cost more computation time. However, we can reduce the influence of this by employing the sparse grid technique which reduces the complexity and is parallel. In the following, we will introduce some basic concepts about the *Black-Scholes equation*.

In the *Black-Scholes* model, the underlying asset S follows Geometric Brownian motion (GBM):

$$dS = \mu S dt + \sigma S dW. \tag{1.4}$$

Here μ and σ represent the expected asset return and volatility of asset return respectively. $W(t)$ is a Wiener process with $W(0) = 0$ and the increment $W(t) - W(s)$ is Gaussian with mean 0 and variance $t-s$ for any $0 \leq s < t$, and increments for non-overlapping time intervals are independent. In the *Black-Scholes* framework, σ is constant and μ is set to be risk-free rate r. Based on the arbitrage-free assumption, the multi-asset *Black-Scholes equation* on domain $[0, T] \times \Omega$ holds:

$$\frac{\partial C}{\partial t} + \frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{d}\rho_{ij}\sigma_i\sigma_j S_i S_j \frac{\partial^2 C}{\partial S_i S_j} + \sum_{i=1}^{d}(r - q_i)S_i\frac{\partial C}{\partial S_i} - rC = 0, \tag{1.5}$$

- C stands for the analytical option price which is determined by time t and d assets' prices.
- σ_i is volatility of ith underlying stock.
- $\rho_{i,j}$ is the correlation between stocks i and j.
- q_i is continuous dividend payment for ith asset.
- r is risk-free rate.
- T means maturity time.

1.2 Motivation and objective

In high dimensional approximation, there is a challenge we cannot ignore called the *curse of dimensionality*, a term due to Bellmann [5]. If we demand fast

convergence when dimension d is growing, we need to employ an exponentially increasing number of collocation points at order N^d, where N is the number of points in one direction. More recently, a method called multilevel sparse grid kernel interpolation (MuSIK) [48] (formerly referred to as MLSKI) was successfully applied in high dimension interpolations by Georgoulis, Levesley and Subhan. In Subnhan's thesis [109], he proposed the sparse grid kernels (SIK) algorithm, for the solution of interpolation problem in high dimensions. The scheme uses direction-wise decomposition of structured interpolation data sites in conjunction with the application of kernel-based interpolants with different scaling in each direction. SIK algorithm can be viewed as an extension of the idea of sparse grids to kernel-based functions. To achieve accelerated convergence, SIK is extended to the multilevel version (MuSIK). The experiments in [48, 109] showed that both SIK and MuSIK are stable and MuSIK gives rapid convergence results even in four dimensions interpolation. This phenomenon is the main motivation for us to develop SIK collocation (SIK-C) and MuSIK collocation (MuSIK-C) algorithm in PDEs solving. We not only have an interest in elliptic and parabolic problem with smooth boundaries but also would like to exploit problems with non-smooth conditions like options pricing in the *Black-Scholes* model.

1.3 Main achievements

In this thesis, we do not mean that MuSIK-C is perfect. But MuSIK-C is easily implemented for high dimensional approximations meanwhile it keeps rapid convergence and gives accurate approximations. In Section 4.3, we show that MuSIK-C has similar performance compared to mesh-based methods in low dimension and achieves better approximations in high dimension. During the experiments, we observe that the convergence rates of MuSIK-C do not decrease while the dimension d increases. Moreover, MuSIK-C even shows spectral convergence in many experiments, such as pricing one dimensional European call option in Section 5.1.3. We found our MuSIK-C method can not handle parabolic problem with non-smooth initial conditions, like *Black-Scholes equation*. In Chapter 5, we introduce a method to

take a relatively smooth estimation as the initial condition at an earlier time before the maturity time. Then we make MuSIK-C applicable in the remaining domain.

1.4 Thesis outline

In Chapter 2 we introduce anisotropic tensor product basis functions that will be used in this thesis and some basic definitions, theories and notations that are helpful to have an overview of the RBF.

Chapter 3 focuses on typical methods for partial differential equation solving with RBFs, including the Kansa collocation method, the Method of Lines (MOL) and the space-time method. In Section 3.3, there is an example to show the performance of the Method of Lines and the space-time method when approximating option price. We demonstrate that the space-time method with MQ has a similar convergence rate as the MOL using MQ.

Chapter 4 shows our sparse grid kernel collocation (SIK-C) and multilevel sparse grid kernel collocation (MuSIK-C) algorithms. At the end of this chapter, we show the superiority of MuSIK-C in some collocation experiments with smooth conditions that reach four dimensions (including time dimension for the parabolic problem) and comparisons with some recent mesh-based methods.

In Chapter 5, we observe that only SIK-C using MQ can be used directly to solve *Black-Scholes equation* with a non-smooth initial condition. So we introduce a method to estimate a relatively smooth initial condition at an earlier time. Then we implement MuSIK-C to solve *Black-Scholes equation* with that relatively smooth initial condition for pricing the European option and the Margrabe option. At the end, we apply Richardson Extrapolation to make the solutions more accurate and accelerate the convergence rate.

A summary of the thesis and future work is given in Chapter 6.

Chapter 2

Scattered data approximation

2.1 Scattered data interpolation problem

The interpolation scheme is to construct an estimating function $\hat{u}$ which is good enough to pass through all the given measurements (called the data values) at the corresponding locations (called the data sites). Normally this is a mapping from $\mathbb{R}^d$ to $\mathbb{R}$ (d is the dimension of the data). Depending on the process, we are not only interested in the given data values but also concerned about deducing approximations at locations that are different from those indicated by available measurements. If the region on which the measurements located do lie on a uniform or a regular grid then the process is called grid or mesh data interpolation, otherwise, it's called scattered data interpolation.

Definition 2.1 (Scattered data interpolation problem)**.** Let $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \subseteq \mathbb{R}^d$, for those points, given pairwise data $(\mathbf{x}_i, y_i)$, $y_i \in \mathbb{R}$. The multivariate scattered data interpolation problem is to find a function $\hat{u} : \mathbb{R}^d \to \mathbb{R}$ such that $\hat{u}(\mathbf{x}_i) = y_i$ for $i = 1, ..., N$, where $\hat{u}$ is called the interpolant to the data.

Here $\mathbf{x}_i$ are the measurements locations, and y_i are the corresponding measurements, the data set $(\mathbf{x}_i, y_i)$ can also be called sample set. Normally, the convenient approach for resolving the scattered data interpolation problem is to assume the interpolant $\hat{u}$ is a linear combination of certain number of basis functions, $\phi_k(\mathbf{x})$,

$k = 1, ..., N$, i.e.,

$$\hat{u}(\mathbf{x}) = \sum_{k=1}^{N} \lambda_k \phi_k(\mathbf{x}), \quad \mathbf{x} \in \mathbb{R}^d. \tag{2.1}$$

The interpolant satisfies the conditions

$$\hat{u}(\mathbf{x}_i) = y_i, \quad i = 1, 2, \ldots, N. \tag{2.2}$$

These equations lead to the linear system

$$\mathbf{A}\boldsymbol{\lambda} = \mathbf{y}, \tag{2.3}$$

where the entries of the interpolation matrix $\mathbf{A}$ are given by

$$\mathbf{A}_{j,k} = \phi_k(\mathbf{x}_j), \quad j, k = 1, 2, \ldots, N,$$

and $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_N]^T, \mathbf{y} = [y_1, \ldots, y_N]^T$.

The non-singularity of matrix $\mathbf{A}$ guarantees there is only a unique solution of the problem. In [89], Micchelli demonstrated some restrictions to ensure matrix $\mathbf{A}$ is non-singular by proving that complete monotonicity of one function implies that its conditional positive definiteness.

2.2 Basic concepts

In this section, we will introduce some basic, essential definitions and theorems. These can be found in any elementary textbook on RBFs (see [10, 29, 119]).

Definition 2.2 $(l_p - norm)$**.** Let $A : \mathbb{R}^n \to \mathbb{R}^m$ and $u \in \mathbb{R}^n$. The $p - norm$ or $l_p - norm$, $p \geq 1$ of matrix A induced by the vector norm $\|\cdot\|_p$ is defined as

$$\|A\|_p = \max_{u \in \mathbb{R}^n \setminus \{\mathbf{0}\}} \frac{\|Au\|_p}{\|u\|_p},$$

here $\|u\|_p = (\sum_{i=1}^{n} |u_i|)^{\frac{1}{p}}$.

$l_1 - norm$, $l_2 - norm$ and $l_\infty - norm$ are displayed as follows:

- $(l_1 - norm)$ $\|u\|_1 = \sum_{i=1}^{n} |u_i|$,
  $(l_1 - norm)$ $\|A\|_1 = \max_{1 \le j \le n} \sum_{i=1}^{n} |A_{ij}|$,
- $(l_2 - norm)$ $\|u\|_2 = (\sum_{i=1}^{n} |u_i|^2)^{\frac{1}{2}}$,
  $(l_2 - norm)$ $\|A\|_2 = \sqrt{\rho\,(A^T A)}$, where ρ is the spectral radius,
- $(l_\infty - norm)$ $\|u\|_\infty = \max_{1 \le i \le n} |u_i|$,
  $(l_\infty - norm)$ $\|A\|_\infty = \max_{1 \le i \le n} \sum_{j=1}^{n} |A_{ij}|$.

Definition 2.3 (Multi-index notation)**.** Let $\mathbb{N}_0$ denote the set of non-negative integers. A d-dimensional multi-index is a d-tuple $\alpha = (\alpha_1, \ldots, \alpha_d) \in \mathbb{N}_0^d$. For $\mathbf{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$, we define

$$\begin{aligned} |\alpha| &= \sum_{i=1}^{d} \alpha_i, \\ \mathbf{x}^\alpha &= \prod_{j=1}^{d} x_j^{\alpha_j}, \end{aligned}$$

and

$$D^\alpha = \left(\frac{\partial}{\partial x_1}\right)^{\alpha_1} \cdots \left(\frac{\partial}{\partial x_d}\right)^{\alpha_d} = \frac{\partial^{|\alpha|}}{\partial x_1^{\alpha_1} \cdots \partial x_d^{\alpha_d}}.$$

Definition 2.4 (Fill distance)**.** The fill distance corresponding to the data set $\mathbf{X}$ in domain Ω is defined as:

$$h_{\mathbf{X},\Omega} = \sup_{\mathbf{x} \in \Omega} \min_{\mathbf{x}_j \in \mathbf{X}} ||\mathbf{x} - \mathbf{x}_j||_2.$$

The fill distance is used as a measure of the data distribution is also known as the covering radius.

Definition 2.5 (Separation distance)**.** The separation distance of data site $\mathbf{X}$ is defined as:

$$q_{\mathbf{X}} = \frac{1}{2} \min_{i \ne j} ||\mathbf{x}_i - \mathbf{x}_j||_2.$$

This is also referred to as the packing radius. The separation distance can be understood physically as the maximum radius r that guarantees there are not overlapped open spheres centring at nodes in $\mathbf{X}$, $\{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x} - \mathbf{x}_j\|_2 \le r, \mathbf{x}_j \in \mathbf{X}\}$.

Definition 2.6 (Condition number)**.** The condition number of a matrix A with respect to any matrix norm $||.||$ is

$$\kappa(A) = ||A||||A^{-1}||.$$

Using the $l_2 - norm$, the condition number $\kappa(A)$ is displayed as:

$$\kappa\left(A\right) = ||A||_2||A^{-1}||_2 = \frac{\sigma_{max}}{\sigma_{min}},$$

here σ_{max} and σ_{min} are maximal and minimal singular values of A respectively, and especially, when matrix A is positive definite the statement can be rewritten as

$$\kappa(A) = \frac{\lambda_{max}}{\lambda_{min}},$$

where λ_{max} and λ_{min} are maximal and minimal modulus values of eigenvalues of A respectively.

It is critical to control the size of condition number because conditioning is a measure of the numerical stability of the interpolation process. The condition number is used to quantify the sensitivity to perturbations of a linear system, and to estimate the accuracy of a computed solution. For example, there is a linear system

$$A\mathbf{y} = \mathbf{b}. \tag{2.4}$$

It there is a perturbation in the vector $\mathbf{b}$, such that $\tilde{\mathbf{b}} = \mathbf{b} + \delta\mathbf{b}$, then Equation (2.4) becomes

$$A\tilde{\mathbf{y}} = \tilde{\mathbf{b}}. \tag{2.5}$$

Define $\delta\mathbf{y} = \tilde{\mathbf{y}} - \mathbf{y}$, it can be shown [113] that

$$\frac{\|\delta\mathbf{y}\|}{\|\mathbf{y}\|} \leq \kappa\left(A\right)\frac{\|\delta\mathbf{b}\|}{\|\mathbf{b}\|}. \tag{2.6}$$

Similarly, if $\tilde{A} = A + \delta A$, from equation

$$\tilde{A}\tilde{\mathbf{y}} = \mathbf{b}, \tag{2.7}$$

we obtain [113]

$$\frac{\|\delta \mathbf{y}\|}{\|\mathbf{y}\|} \leq \kappa\left(A\right) \frac{\|\delta A\|}{\|A\|} \tag{2.8}$$

If the condition number of matrix A is $\kappa(A) = \mathcal{O}(1)$, we can demonstrate that the system is not sensitive to small perturbations. Therefore, it is well-conditioned. In contrast, a system which is sensitive to small perturbations is called ill-conditioned, and it suffers from instabilities when solved on a machine, while the matrix has a large condition number.

Definition 2.7 (Positive definite matrix)**.** A real N square symmetric matrix A is called positive semi-definite if its associated quadratic form is non-negative, i.e.,

$$\sum_{j=1}^{N} \sum_{k=1}^{N} \lambda_j \lambda_k A_{jk} \geq 0,$$

for $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_N]^T \in \mathbb{R}^N$. If the only vector $\boldsymbol{\lambda}$ that turns the above quadratic form into an equality is the zero vector, then A is called positive definite.

A positive definite matrix A has an inverse matrix A^{-1}, because its determinant is not zero.

Definition 2.8 (Positive definite (PD) function)**.** A real valued continuous function $\phi : \mathbb{R}^d \to \mathbb{R}$ is positive semi-definite on $\mathbb{R}^d$ if and only if it is even and

$$\sum_{j=1}^{N} \sum_{k=1}^{N} \lambda_j \lambda_k \phi(\mathbf{x}_j - \mathbf{x}_k) \geq 0,$$

for any N pairwise different points $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \subseteq \mathbb{R}^d$, and $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_N]^T \in \mathbb{R}^N$. The function ϕ is strictly positive definite on $\mathbb{R}^d$ if the only vector $\boldsymbol{\lambda}$ that turns the above into the equality is the zero vector.

Positive definite functions have an important role in approximation theory and statistics, and the property is crucial for interpolation.

2.3 Radial basis functions

Firstly, we give the definition of a radial function as:

Definition 2.9 (Radial function)**.** A function $\Phi : \mathbb{R}^d \to \mathbb{R}$ is defined as a radial if there is a univariate function $\phi : [0, \infty) \to \mathbb{R}$, such that $\Phi(\mathbf{x}) = \phi(r)$, where $r = \|\mathbf{x}\|$, and $\|.\|$ is some norm on $\mathbb{R}^d$ (typically the Euclidean norm).

Supposing we have some scattered points located in a domain $\mathbb{R}^d$, which is called centres or nodes and represented by $\mathbf{x}_i = (x_i^1, x_i^2, \ldots, x_i^d)$. From the definition above, a radial function depends on the distance from variable point $\mathbf{x}$ to centre node $\mathbf{x}_i$ as $\Phi(\mathbf{x} - \mathbf{x}_i) = \phi(\|\mathbf{x} - \mathbf{x}_i\|)$, here the norm represents for the Euclidean distance.

From the different kinds of RBFs, we list some widely utilised globally supported radial basis functions (GSRBFs) in Table 2.1. Because the interpolation matrix depending on GSRBFs is full, computation cost grows exponentially as the size of problem increasing. In contrast, compactly supported radial basis functions (CSRBFs) were introduced with the advantage that they produce a sparser linear system for the interpolation problem. Currently, Wendland's functions are popular CSRBFs and have been further developed and modified in [117, 119]. Some examples are listed in Table 2.2. Other CSRBFs such as Wu's functions can be found in [29, 123].

Name of RBFs	Functional Form $\phi(r) =$	Parameters
Gaussians	$e^{-(cr)^2}$	$c > 0$
Polyharmonic Splines	r^ν	$\nu > 0$, $\nu \notin 2\mathbb{N}$
Thin Plate Splines (TPS)	$r^{2k}\log(r)$	$k \in \mathbb{N}$
Multiquadric(MQ)	$(c^2 + r^2)^{\frac{\nu}{2}}$	$\nu > 0,\ \nu \notin 2\mathbb{N},\ c > 0$
Inverse Multiquadric(IMQ)	$(c^2 + r^2)^{\frac{\nu}{2}}$	$v < 0,\ c > 0$

TABLE 2.1: Example of some globally supported radial basis functions

As early as two decades ago, many researchers demonstrated the accuracy, stability and ease of implementation of MQ and Gaussians basis functions; for instance [41, 89, 96, 108]. Since this MQ and Gaussians have comparatively high accuracy and are infinitely differentiable, many authors have a preference to utilise them

Dimension d	Radial Basis Function	smoothness
$d = 1$	$\phi_{1,0}(r) = (1-r)_+$	C^0
	$\phi_{1,1}(r) = (1-r)_+^3(3r+1)$	C^2
	$\phi_{1,2}(r) = (1-r)_+^5(8r^2+5r+1)$	C^4
$d \leq 3$	$\phi_{3,0}(r) = (1-r)_+^2$	C^0
	$\phi_{3,1}(r) = (1-r)_+^4(4r+1)$	C^2
	$\phi_{3,2}(r) = (1-r)_+^6(35r^2+18r+3)$	C^4
	$\phi_{3,3}(r) = (1-r)_+^8(32r^3+25r^2+8r+1)$	C^6
$d \leq 5$	$\phi_{5,0}(r) = (1-r)_+^3$	C^0
	$\phi_{5,1}(r) = (1-r)_+^5(5r+1)$	C^2
	$\phi_{5,2}(r) = (1-r)_+^7(16r^2+7r+1)$	C^4

TABLE 2.2: Example of Wendland's compactly supported radial basis functions.

in the literature, for example, [32, 38, 62, 75, 95]. In this thesis, we also consider these two as our basis functions.

The multiquadric(MQ) and Gaussian functions are displayed as follow:

$$Multiquadric \quad : \quad \phi(\|\mathbf{x}-\mathbf{x}_i\|) = \sqrt{\|\mathbf{x}-\mathbf{x}_i\|^2 + c^2}, \tag{2.9}$$

$$Gaussian \quad : \quad \phi(\|\mathbf{x}-\mathbf{x}_i\|) = e^{-\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{c^2}}, \tag{2.10}$$

where c is called the shape parameter which has a huge influence for RBFs. For the above presentations, different choices of shape parameters can lead to different shapes of RBFs from peak to flat as shown in Figure 2.1.

As mentioned earlier, while the shape of the RBF is becoming flatter the condition number of the system is also growing and the approximation is more accurate. However, once the shape parameter c exceed a limit which is not easy to predict,

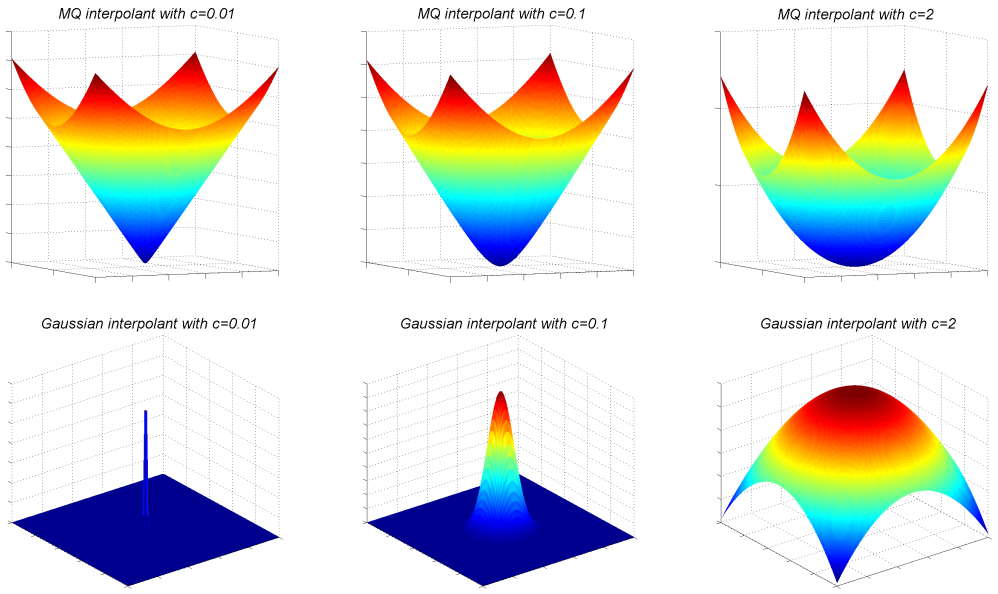

FIGURE 2.1: Different RBF shapes with different shape parameter values as $c = 0.01$, $c = 0.1$ and $c = 2$.

the system becomes too ill-conditioned and unstable, so the results are not credible. Furthermore, in [25] Driscoll and Fornberg stated that even though a small value of c ($c \to 0$) can provide a well-conditioned linear system, an inaccurate solution also comes out. Those observations suggest that different RBFs with shape parameters should have a reliable region for their shape parameters. In 1995, Schaback [101] mentioned that there is a balancing point between accuracy and good condition and we cannot guarantee both. Later, the Contour-Padé algorithm [40] and RBF-QR method [39] were proposed to handle flatter RBFs more stably. Then, Fasshauer and Mccourt [31] introduced a stable method with flat Gaussian kernels and Fornberg, Larsson and Flyer [38] extended the RBF-QR approach to three dimensions. Recently, in [7] variably scaled kernels (VSK) method was proposed to reduce the condition number by treating the shape parameter as an extra space variable. As so far, the optimal shape parameter is still an open problem in RBF research.

Definition 2.10 (Completely monotone)**.** A function φ is completely monotone on $[0, \infty)$ if:

1. $\varphi \in \mathbf{C}[0, \infty)$.
2. $\varphi \in \mathbf{C}^{\infty}(0, \infty)$.

3. $(-1)^l \varphi^{(l)}(r) \geq 0$ where $r > 0$ and $l = 0, 1, \cdots$

Theorem 2.11 (Micchelli)**.** *Let $g \in C^\infty[0, \infty)$ be such that g' is completely monotonic but not constant. Suppose further that $g(r) \geq 0$. Then the interpolation matrix A is nonsingular for $\phi(r) = g(r^2)$.*

Proof. Supposing $A = \{\phi(\|\mathbf{x}_i - \mathbf{x}_j\|)\}_{\mathbf{x}_i, \mathbf{x}_j \in X}$ and X is set of points. Since $g(r) \in C^\infty[0, \infty)$, then we have

$$g(r) = g(0) + \int_0^r g'(x)dx.$$

By replacing $g'(x)$ with the Bernstein-Widder representation

$$g'(x) = \int_0^\infty e^{-\alpha x} d\mu(\alpha),$$

and exchange integrals. From Fubini's theorem, it is allowed to change the order of integration in iterated integrals. We obtain

$$g(r) = g(0) + \int_0^r \int_0^\infty e^{-\alpha x} d\mu(\alpha) dx.$$

Suppose $\lambda \in \mathbb{R}^X$ and $\sum_{i \in X} \lambda_i = 0$, then we have

$$\int_0^r e^{-\alpha x} dx = -\alpha^{-1} e^{-\alpha r} + \alpha^{-1}.$$

and

$$\lambda^T A \lambda = -\int_0^\infty \sum_{i \in X} \sum_{j \in X} \lambda_i \lambda_j \alpha^{-1} e^{-\alpha \|i-j\|^2} d\mu(\alpha),$$

Thus $\lambda^T A \lambda < 0$ for all λ and except $\lambda = 0$, this means there is one negative eigenvalue in A with remainder of positive eigenvalues. □

The above proof is taken from [10]. There is also a nice proof by Powell in [96].

2.4 Anisotropic tensor product basis function

Definition 2.12. (Anisotropic Radial Basis Function) Let $\phi(\|\cdot - \mathbf{x}_i\|)$ be a given RBF centred at $\mathbf{x}_i \in \mathbb{R}^d$ and let $A \in \mathbb{R}^{d\times d}$ be an invertible matrix. The anisotropic radial basis function ϕ_A is defined by

$$\phi_A(\|\cdot - \mathbf{x}_i\|) = \phi(\|A\,(\cdot - \mathbf{x}_i)\,\|).$$

Considering the domains on different directions are not in the same size and sparse grid distribution which will be introduced in Chapter 4, here we choose anisotropic tensor product basis functions (ATPBFs) instead of norm form:

$$Multiquadric \quad : \quad \phi_{A,\mathbf{x}_i}(\mathbf{x}) = \prod_{k=1}^{d} \sqrt{A_k^2(x_k - x_i^k)^2 + c_k^2}, \tag{2.11}$$

$$Gaussian \quad : \quad \phi_{A,\mathbf{x}_i}(\mathbf{x}) = \prod_{k=1}^{d} e^{-\frac{A_k^2(x_k - x_i^k)^2}{c_k^2}}, \tag{2.12}$$

where k is the k^{th} dimension of variable $\mathbf{x}$, coefficient A_k is k^{th} diagonal element of scaling matrix $A \in \mathbb{R}^d \times \mathbb{R}^d$ that make approximations suitable for different grids, see Figure 2.2 and Figure 2.3. From the equations (2.11) and (2.12), we observe that ATPBF for the Gaussian is still belong to RBFs. In particular, it is a scaled RBF. However, ATPBF for MQ is not radial any more.

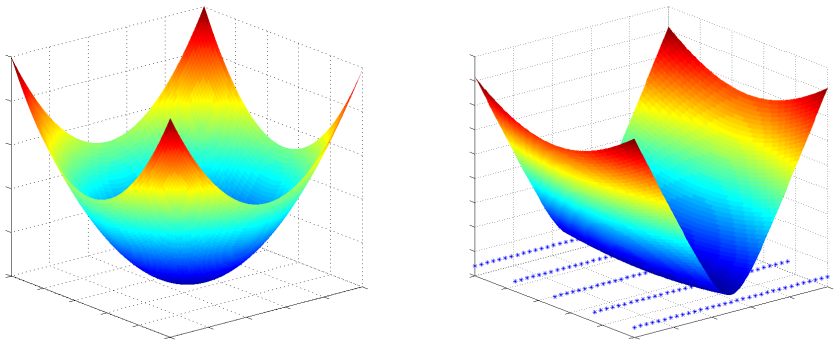

FIGURE 2.2: An example of normal MQ and anisotropic tensor MQ funcions in two dimensions.

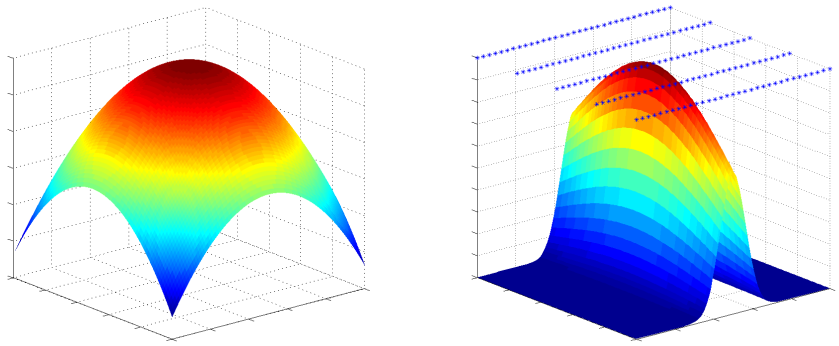

FIGURE 2.3: An example of normal Gaussian and anisotropic tensor Gaussian functions in two dimensions.

It is not necessary to put A_k before $(x_k - x_i^k)^2$. We can draw a similar expression as:

$$Multiquadric \quad : \quad \phi_{A,\mathbf{x}_i}(\mathbf{x}) = \prod_{k=1}^{d} \sqrt{(x_k - x_i^k)^2 + (Ch_k)^2}, \tag{2.13}$$

$$Gaussian \quad : \quad \phi_{A,\mathbf{x}_i}(\mathbf{x}) = \prod_{k=1}^{d} e^{-\frac{(x_k - x_i^k)^2}{(Ch_k)^2}}, \tag{2.14}$$

here we consider $Ch_k = \frac{c_k}{A_k}$, $(k = 1, 2, \ldots, d)$ as "shape parameters" specially chosen for center $\mathbf{x}_i$ according to different spatial dimensions, h_k is nodes distance in k^{th} direction. The values of A_k are always set to be nodes number in corresponding direction minus one. Shape parameters $c_k \in \mathbb{R}$ are always selected as $c_k = C\mathfrak{L}_k$, here C is a connection constant number. Supposing $x_k \in [a_k, b_k]$, then $\mathfrak{L}_k = b_k - a_k$. The ATPBFs are the main basis functions used in this paper. The derivatives from this expression can be derived as:

$$\begin{aligned}
Multiquadric : D_{x_p}(\phi_{A,\mathbf{x}_i}) &= \frac{x_p - x_i^p}{\sqrt{(x_p - x_i^p)^2 + (Ch_p)^2}} \prod_{k \neq p} \sqrt{(x_k - x_i^k)^2 + (Ch_k)^2}, \\
D^2_{x_p}(\phi_{A,\mathbf{x}_i}) &= \frac{(Ch_p)^2}{[(x_p - x_i^p)^2 + (Ch_p)^2]^{\frac{3}{2}}} \prod_{k \neq p} \sqrt{(x_k - x_i^k)^2 + (Ch_k)^2}, \\
Gaussian : D_{x_p}(\phi_{A,\mathbf{x}_i}) &= -\frac{2(x_p - x_i^p)}{(Ch_p)^2} \prod_{k} e^{-\frac{(x_k - x_i^k)^2}{(Ch_k)^2}}, \\
D^2_{x_p}(\phi_{A,\mathbf{x}_i}) &= (-\frac{2}{(Ch_p)^2} + \frac{4(x_p - x_i^p)^2}{(Ch_p)^4}) \prod_{k} e^{-\frac{(x_k - x_i^k)^2}{(Ch_k)^2}}.
\end{aligned}$$

Given a set of scattered data points $\mathbf{X} = \{\mathbf{x_i},\ i = 1, 2, \ldots, N\} \subseteq \mathbb{R}^d$, then the approximation $\hat{u}$ in the rest of this thesis is formed as:

$$\hat{u}(\mathbf{x}) = \sum_{i=1}^{N} \lambda_i \phi_{A,\mathbf{x}_i}(\mathbf{x}). \tag{2.15}$$

For the interpolation problem, firstly we consider one dimensional case. Given a set of scattered data $X = \{\xi_1, \xi_2, \ldots, \xi_N\}$ and corresponding function values $Y = \{\eta_1, \eta_2, \ldots, \eta_N\}$. The interpolant $\hat{u}$ can be written in Lagrange form as

$$\hat{u}(x) = \sum_{j=1}^{n} \eta_j u_j(x) \tag{2.16}$$

where the $u_j(x)$ are the so-called *univariate cardinal basis functions*, defined by the property that $u_j(\xi_i) = \delta_{ij}$, the Kronecker delta.

From Equation (2.15), we obtain the following matrix-vector equation by enforcing $\hat{u}(\mathbf{x}_i) = \mathbf{y}_i$

$$\Phi \cdot \boldsymbol{\lambda} = \mathbf{y}. \tag{2.17}$$

Here, $\Phi_{i,j} = \phi_{A,\mathbf{x}_j}(\mathbf{x}_i)$. As dimension d increases, the size of matrix Φ also grows. The tensor product nature of our radial basis functions give us an alternative approach to solve the above equation.

$$\begin{aligned} \Phi \cdot \boldsymbol{\lambda} &= \mathbf{y} \\ \Rightarrow \boldsymbol{\lambda} &= \Phi^{-1} \cdot \mathbf{y}. \end{aligned}$$

Then the problem is to find the inverse of the matrix Φ without solving it. We can use the property of *Kronecker Product* on the univariate cardinal basis function:

$$\begin{aligned} (\Phi_1 \cdot (\Phi_1)^{-1}) \otimes \cdots \otimes (\Phi_d \cdot (\Phi_d)^{-1}) = \mathbf{I}_1 \otimes \cdots \otimes \mathbf{I}_d &= \mathbf{I} \\ \Rightarrow (\Phi_1 \otimes \cdots \otimes \Phi_d) \cdot ((\Phi_1)^{-1} \otimes \cdots \otimes (\Phi_d)^{-1}) &= \mathbf{I}. \end{aligned}$$

By using the tensor product nature we know that:

$$\Phi = \Phi_1 \otimes \cdots \otimes \Phi_d. \tag{2.18}$$

Combine the above equations together, we can have:

$$\begin{aligned} (\Phi)^{-1} &= (\Phi_1)^{-1} \otimes \cdots \otimes (\Phi_d)^{-1} \\ \Rightarrow \boldsymbol{\lambda} &= (\Phi)^{-1} \cdot \mathbf{y} = ((\Phi_1)^{-1} \otimes \cdots \otimes (\Phi_d)^{-1}) \cdot \mathbf{y}. \end{aligned}$$

This formula shows that the high dimensional interpolation coefficients can be constructed by tensor product of the univariate cardinal basis function, due to the tensor product nature of ATBRFs, but not solving the large size ill-conditioned matrix. For instance, the size of a interpolation matrix Φ on a full grid is N^{2d}, N is nodes number in one direction and d is dimension. The normal cost to invert Φ is $\mathcal{O}(N^{6d})$. However, the cost to invert Φ_i is $\mathcal{O}(N^6)$ and the cost to make *Kronecker Product* is $\mathcal{O}(N^{2d})$. Moreover, we can easily generalize this method. Hence, we not only improve the numerical stability but also reduce the memory requirement for solving the interpolation problem. Therefore, the numerical stability is only related with the condition number of univariate cardinal function interpolation.

2.5 Convergence

For the MQ and Gaussian RBFs used in this thesis, there are two approaches to make the approximation converge. One is to refine the mesh size h. That means we need to utilise more nodes, with the relative we increase the computational cost. Another way is to use larger shape parameter c, which is performed without extra cost. However, as the c becomes larger, the shape of the basis function is flatter and the system matrix becomes more ill-conditioned. This phenomenon is demonstrated by Schaback's uncertainty principle [101]. Our numerical experiments also provided the same performance.

For the accuracy of the RBF interpolation, Madych and Nelson [87] gave the proof of the exponential convergence for a class of RBFs. Furthermore, Wendland [118]

refined the error bound to $\mathcal{O}\left(\lambda^{\sqrt{\frac{1}{h}}}\right)$ for the Gaussian, where $0 < \lambda < 1$. In [84], Madych proposed an error estimate for MQ as $\mathcal{O}\left(e^{ac}\lambda^{\frac{c}{h}}\right)$, where a is a positive constant. In Hermite interpolation, the investigation of the rate of convergence of Hermite interpolation has been done by Luo and Levesley [82] with a modification method of variational approach of Madych and Nelson [85, 86]. For the elliptic PDE problem, Franke and Schaback [43] used symmetric collocation method to find an L_∞ error bound, and in term of L_2 norm, it has an additional convergence factor $h^{d/2}$ for MQ basis function. There is some theoretical work on multilevel RBF collocation method, such as [55, 91, 120]. Afterwards, Farrell and Wendland [26] showed a convergence theory for multilevel collocation using CSRBFs which is also based on symmetric collocation method. Although there is no theoretical proof for the convergence of our method, numerical experiments demonstrate that good convergence results are also observed using MuSIK-C.

2.6 An example

In this section, we present a two dimensional elliptic example on the domain $\Omega = [0, 1]^2$ to investigate on the condition number (Cond) and convergence in function of different kinds of shape parameters. We use Kansa method which is described in Section 3.1 to solve this example. As shown in Equations (2.13) and (2.14) that we utilise in this thesis, the shape parameter at the kth direction is in the form Ch_k. h_k means the nodes distance in the kth direction. In this example, we only use the full grid collocation and there are two dimensions. So at the level n, $h_1^n = h_2^n = \frac{1}{N^n-1}$, here N^n is the number of nodes in one dimension. C is the connection constant number and is equal to 2 in this experiment. This choice is also used in most examples in this thesis. Therefore, shape parameters are scaling with different uniform grids at different levels. e^n means the l_∞ error at the level n:

$$e^n = \max_{\mathbf{x}\in\mathbf{T}} |u(\mathbf{x}) - \hat{u}^n(\mathbf{x})|,$$

here $\mathbf{T}$ is a testing points set that contains 24,000 uniform points in the whole domain. The order is depending on the nodes distance h_1^n at the level n.

$$\text{Order} = \frac{\log(e^{n+1}) - \log(e^n)}{\log(h_1^{n+1}) - \log(h_1^n)}.$$

Example 2.1. *In this example, we consider Poisson's equation*

$$u_{xx} + u_{yy} = \frac{2x(x^2y^2 - 3y^2 + x^4 + 2x^2 + 1)}{(x^2+1)^3}, \quad \Omega = [0,1]^2, \tag{2.19}$$

with boundary conditions

$$\begin{aligned} u(x,0) &= 0, \quad x \in [0,1], \\ u(x,1) &= \frac{x}{x^2+1}, \quad x \in [0,1], \\ u(0,y) &= 0, \quad y \in [0,1], \\ u(1,y) &= \frac{y^2}{2}, \quad y \in [0,1]. \end{aligned}$$

Here, the analytical solution is

$$u(x,y) = \frac{xy^2}{x^2+1}.$$

Level	N	Cond(MQ)	e(MQ)	Order(MQ)
1	3	8e3	2.0e-2	—
2	5	7e5	6.1e-3	1.7
3	9	2e7	2.8e-3	1.1
4	17	4e8	2.0e-3	0.5
5	33	8e9	1.2e-3	0.7
6	65	1e11	6.8e-4	0.8

TABLE 2.3: Results using MQ with the scaling shape parameter $2h_1^n$.

In Table 2.3 and 2.4, it is evident that we control condition number in an acceptable range. It is interesting that convergence rate in Table 2.3 seems to be near 1, while the order in Table 2.4 is closing to 0. This phenomenon about the collocation with the Gaussian basis function is also observed in Section 3.3, Section 4.3 and Chapter 5. However, the numerical results in Section 4.3 also demonstrate that MuSIK-C can overcome the problem to obtain a rapid convergence. In Table 2.5

Level	N	Cond(G)	e(G)	Order(G)
1	3	1e3	5.0e-2	—
2	5	8e4	2.4e-2	1.04
3	9	4e6	1.8e-2	0.46
4	17	6e7	1.6e-2	0.16
5	33	3e8	1.5e-2	0.07
6	65	1e9	1.5e-2	0.03

TABLE 2.4: Results using the Gaussian with the scaling shape parameter $2h_1^n$.

Level	N	Cond(MQ)	e(MQ)	Order(MQ)
1	3	50	4.7e-2	—
2	5	5e3	1.6e-2	1.6
3	9	4e6	4.2e-3	1.9
4	17	2e11	6.4e-4	2.7
5	33	1e20	2.6e-5	4.7
6	65	5e21	3.5e-6	2.9

TABLE 2.5: Results using MQ with a constant shape parameter 0.2.

Level	N	Cond(G)	e(G)	Order(G)
1	3	100	2.9e-1	—
2	5	198	8.8e-2	1.7
3	9	8e4	3.7e-2	1.2
4	17	3e15	1.1e-3	5.1
5	33	2e21	1.9e-5	5.8
6	65	2e22	6.6e-5	-1.8

TABLE 2.6: Results using the Gaussian with a constant shape parameter 0.2.

and Table 2.6, we take a constant distance $h_c^n = 0.1$ instead of h_1^n at different levels. Therefore, shape parameter is fixed as 0.2. We can see that the orders are very fast, and we can achieve very accurate results in most cases. However, the condition number is growing extremely fast so that in the level 6 in Table 2.6, the error is bigger, and the order is a negative number.

Chapter 3

Solving partial differential equations using RBFs

One important application of radial basis functions (RBFs) is to solve partial differential equations (PDEs). The collocation method with RBFs is renowned in resolving elliptic boundary value problems (see Definition 3.1). After Myers et al. [90] proposed the space-time method in RBF field, many researchers have worked on applying collocation into parabolic problems. In this chapter, we firstly review one well-known collocation method called the Kansa method which is utilised in this thesis in Section 3.1. We then introduce two main methods used for the parabolic problem (see Definition 3.2), the Method of Lines (MOL) and the space-time method in Section 3.2. In Section 3.3, we present one option pricing example to show the performance of the space-time method and the MOL when solving a parabolic problem.

3.1 Elliptic PDEs

Definition 3.1 (Elliptic differential operator)**.** A linear operator $\mathcal{L} : C^2(\Omega) \to C(\Omega)$ in the following form is an elliptic differential operator of second order

$$\mathcal{L}u(\mathbf{x}) = \sum_{i,j=1}^{d} a_{ij}(\mathbf{x}) \frac{\partial^2}{\partial x_i \partial x_j} u(\mathbf{x}) + \sum_{i=1}^{d} b_i(\mathbf{x}) \frac{\partial}{\partial x_i} u(\mathbf{x}) + b_0(\mathbf{x}) u(\mathbf{x}), \tag{3.1}$$

where the coefficient matrix $\Big[a_{ij}(\mathbf{x})\Big] \in \mathbb{R}^{d\times d}$ satisfies

$$\exists\ \alpha > 0,\ \sum_{i,j=1}^{d} a_{ij}(\mathbf{x}) c_i c_j \geqslant \alpha \|\mathbf{c}\|_2^2$$

for all $\mathbf{x} \in \Omega$ and $\mathbf{c} \in \mathbb{R}^d$.

We solve the second order elliptic PDE with Dirichlet boundary conditions system:

$$\mathcal{L}u = f \quad in\ \Omega, \tag{3.2}$$

$$u = g \quad on\ \partial\Omega, \tag{3.3}$$

where $\mathcal{L}$ is an elliptic operator, f and g are prescribed functions. We choose a set of uniformly distributed points $\Xi = \Xi_1 \cup \Xi_2$ as centre nodes, here $\Xi_1 = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ are interior points of Ω and $\Xi_2 = \{\mathbf{x}_{n+1}, \mathbf{x}_{n+2}, \ldots, \mathbf{x}_N\}$ are located on the boundary $\partial\Omega$. In this thesis, the PDE system is solved by Kansa method [71, 72]. The method is well-known for its easy programming in high dimension and regardless of geometric complexity of problems, while the well-posedness problem of system is still an open question as described in [42, 43] by Franke and Schaback.

Kansa's method is a spectral method in which a global approximation at the form

$$\hat{u}(\mathbf{x}) = \sum_{i=1}^{N} \lambda_i \phi_{A,\mathbf{x}_i}(\mathbf{x})$$

is used. In order to acquire the coefficients $\boldsymbol{\lambda}$, we substitute the equation into the above elliptic system (3.2) and (3.3):

$$\sum_{i=1}^{N} \lambda_i \mathcal{L}\phi_{A,\mathbf{x}_i}(\mathbf{x}_j) = f(\mathbf{x}_j), \quad j = 1, 2, \ldots, n, \tag{3.4}$$

$$\sum_{i=1}^{N} \lambda_i \phi_{A,\mathbf{x}_i}(\mathbf{x}_j) = g(\mathbf{x}_j), \quad j = n+1, n+2, \ldots, N. \tag{3.5}$$

For visual simplicity, a matrix-vector product equation is obtained as:

$$\begin{bmatrix} \mathcal{L}\phi_{A,\mathbf{x}_1}(\mathbf{x}_1) & \mathcal{L}\phi_{A,\mathbf{x}_2}(\mathbf{x}_1) & \cdots & \mathcal{L}\phi_{A,\mathbf{x}_N}(\mathbf{x}_1) \\ \mathcal{L}\phi_{A,\mathbf{x}_1}(\mathbf{x}_2) & \mathcal{L}\phi_{A,\mathbf{x}_2}(\mathbf{x}_2) & \cdots & \mathcal{L}\phi_{A,\mathbf{x}_N}(\mathbf{x}_2) \\ \vdots & \vdots & \cdots & \vdots \\ \mathcal{L}\phi_{A,\mathbf{x}_1}(\mathbf{x}_n) & \mathcal{L}\phi_{A,\mathbf{x}_2}(\mathbf{x}_n) & \cdots & \mathcal{L}\phi_{A,\mathbf{x}_N}(\mathbf{x}_n) \\ \phi_{A,\mathbf{x}_1}(\mathbf{x}_{n+1}) & \phi_{A,\mathbf{x}_2}(\mathbf{x}_{n+1}) & \cdots & \phi_{A,\mathbf{x}_N}(\mathbf{x}_{n+1}) \\ \vdots & \vdots & \cdots & \vdots \\ \phi_{A,\mathbf{x}_1}(\mathbf{x}_N) & \phi_{A,\mathbf{x}_2}(\mathbf{x}_N) & \cdots & \phi_{A,\mathbf{x}_N}(\mathbf{x}_N) \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_n \\ \lambda_{n+1} \\ \vdots \\ \lambda_N \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \\ g_{n+1} \\ \vdots \\ g_N \end{bmatrix}. \tag{3.6}$$

3.2 Parabolic PDEs

Definition 3.2 (Parabolic differential operator)**.** A linear operator $\mathcal{L}$ on Ω in the form:

$$\mathcal{L}u(t, \mathbf{x}) = u_t - \sum_{i,j=1}^{d} a_{ij}(t, \mathbf{x}) \frac{\partial^2 u}{\partial x_i \partial x_j} - \sum_{i=1}^{d} b_i(t, \mathbf{x}) \frac{\partial u}{\partial x_i} - c(t, \mathbf{x}) u(\mathbf{x}), \tag{3.7}$$

is said to be parabolic if for fixed t, the operator with second order term is an elliptic operator. Here the variable t stands for time.

The heat equation is a typical example of a parabolic PDE. The generalization of heat equation with Dirichlet condition on domain $\mathbf{x} \in \Omega \subseteq \mathbb{R}^d$ and $t \in [a, b]$ is:

$$u_t - \overline{\mathcal{L}}u = 0, \quad [a, b] \times \Omega, \tag{3.8}$$

$$u(a, \mathbf{x}) = f(\mathbf{x}), \quad \mathbf{x} \in \overline{\Omega}, \tag{3.9}$$

$$u(t, \mathbf{x}) = g(t, \mathbf{x}), \quad [a, b] \times \partial\Omega, \tag{3.10}$$

where $\overline{\mathcal{L}}$ is an elliptic operator of second order, $f(\mathbf{x})$ is an initial condition and $g(\mathbf{x})$ is a boundary condition.

To solve the heat equation with radial basis functions, there are two main methods to deal with the time variable. One way is to apply the method of lines (MOL) in the time direction with any finite difference method, such as Runge-Kutta method, Crank-Nicolson method and so on. The other is to consider time t as one spatial dimension [90].

3.2.1 Method of lines

The method of lines (MOL) is a standard numerical method for approximating PDEs. The essence of the method is to convert PDE problems to ordinary differential equations (ODEs) by preserving partial derivatives in specially chosen directions and take the place of partial derivatives on other directions by algebraic approximations [102]. In this subsection, I just present the Crank-Nicolson method which was used in [62, 95, 98] and in the following experiments. In the FDM, the Crank-Nicolson scheme as one implicit method is well-known as an unconditionally stable scheme while the explicit scheme is called conditionally stable. Moreover, Giles and Carter [50] emphasized that the Crank-Nicolson method is unconditionally stable in L_2 norm.

Recall the nodes set Ξ in Section 3.1. Set the approximation $\hat{u}_{\mathrm{MOL}}$ constructed by using the MOL in the form:

$$\hat{u}_{\mathrm{MOL}}(t, \mathbf{x}) = \sum_{i=1}^{N} \lambda_i(t) \phi_{A,\mathbf{x}_i}(\mathbf{x}), \quad [a, b] \times \Omega, \tag{3.11}$$

where coefficients $\boldsymbol{\lambda}(t) = [\lambda_1(t), \lambda_2(t), \ldots, \lambda_N(t)]^T$ are functions based on time variable t.

From the initial condition (3.9), we can have N equations:

$$\sum_{i=1}^{N} \lambda_i(a) \phi_{A,\mathbf{x}_i}(\mathbf{x}_j) = f(\mathbf{x}_j) \tag{3.12}$$

for $j = 1, 2, \ldots, N$. Now, introducing matrix algebra to make algorithm clearer, we define matrix Φ:

$$\Phi = \begin{bmatrix} \phi_{A,\mathbf{x}_1}(\mathbf{x}_1) & \phi_{A,\mathbf{x}_2}(\mathbf{x}_1) & \cdots & \phi_{A,\mathbf{x}_N}(\mathbf{x}_1) \\ \phi_{A,\mathbf{x}_1}(\mathbf{x}_2) & \phi_{A,\mathbf{x}_2}(\mathbf{x}_2) & \cdots & \phi_{A,\mathbf{x}_N}(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{A,\mathbf{x}_1}(\mathbf{x}_N) & \phi_{A,\mathbf{x}_2}(\mathbf{x}_N) & \cdots & \phi_{A,\mathbf{x}_N}(\mathbf{x}_N) \end{bmatrix}. \tag{3.13}$$

Thus Equation (3.12) becomes

$$\Phi \boldsymbol{\lambda}^a = \mathbf{f}. \tag{3.14}$$

Here $\boldsymbol{\lambda}^a$ standing for $\boldsymbol{\lambda}(a)$ is coefficients at time $t = a$, $\mathbf{f} = [f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots, f(\mathbf{x}_N)]^T$. According to Theorem 2.11, matrix Φ (3.13) which is constructed by basis function (2.11) or (2.12) is invertible. Therefore, we can multiply inverse matrix Φ^{-1} on both sides of (3.14) in order to achieve coefficients:

$$\boldsymbol{\lambda}^a = \Phi^{-1}\mathbf{f}. \tag{3.15}$$

Following up with the first series of coefficients at initial time, we need an iteration equation to derive coefficients at subsequent times. Substitute Equation (3.11) into (3.8), we obtain

$$\frac{\partial}{\partial t}\sum_{i=1}^{N} \lambda_i(t)\phi_{A,\mathbf{x}_i}(\mathbf{x}) - \sum_{i=1}^{N} \lambda_i(t)\overline{\mathcal{L}}\phi_{A,\mathbf{x}_i}(\mathbf{x}) = 0. \tag{3.16}$$

Apply this equation at the interior points in the set $\Xi_1 = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ to form the matrix-vector product equation

$$\Phi^I \frac{\partial \boldsymbol{\lambda}}{\partial t} - \overline{\mathcal{L}}\Phi^I \boldsymbol{\lambda} = 0, \tag{3.17}$$

where

$$\Phi^I = \begin{bmatrix} \phi_{A,\mathbf{x}_1}(\mathbf{x}_1) & \phi_{A,\mathbf{x}_2}(\mathbf{x}_1) & \cdots & \phi_{A,\mathbf{x}_N}(\mathbf{x}_1) \\ \phi_{A,\mathbf{x}_1}(\mathbf{x}_2) & \phi_{A,\mathbf{x}_2}(\mathbf{x}_2) & \cdots & \phi_{A,\mathbf{x}_N}(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{A,\mathbf{x}_1}(\mathbf{x}_n) & \phi_{A,\mathbf{x}_2}(\mathbf{x}_n) & \cdots & \phi_{A,\mathbf{x}_N}(\mathbf{x}_n) \end{bmatrix}. \tag{3.18}$$

$$\overline{\mathcal{L}}\Phi^I = \begin{bmatrix} \overline{\mathcal{L}}\phi_{A,\mathbf{x}_1}(\mathbf{x}_1) & \overline{\mathcal{L}}\phi_{A,\mathbf{x}_2}(\mathbf{x}_1) & \cdots & \overline{\mathcal{L}}\phi_{A,\mathbf{x}_N}(\mathbf{x}_1) \\ \overline{\mathcal{L}}\phi_{A,\mathbf{x}_1}(\mathbf{x}_2) & \overline{\mathcal{L}}\phi_{A,\mathbf{x}_2}(\mathbf{x}_2) & \cdots & \overline{\mathcal{L}}\phi_{A,\mathbf{x}_N}(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \overline{\mathcal{L}}\phi_{A,\mathbf{x}_1}(\mathbf{x}_n) & \overline{\mathcal{L}}\phi_{A,\mathbf{x}_2}(\mathbf{x}_n) & \cdots & \overline{\mathcal{L}}\phi_{A,\mathbf{x}_N}(\mathbf{x}_n) \end{bmatrix}. \tag{3.19}$$

So far, we use radial basis functions to replace derivatives of target function u on spatial directions and only reserve the partial derivative on time variable t. It is reasonable to consider Equation (3.17) as a system of ODEs problems. The next thing is to employ a method to solve this system of ODEs, such as the Crank-Nicolson method.

Assuming time t is discretized into M uniformly distributed nodes with distance $\Delta t = \frac{b-a}{M-1}$ on $[a, b]$:

$$t_i = a + (i-1) \times \Delta t, \quad i = 1, 2, \ldots, M.$$

Equation (3.17) can be rewritten as:

$$\frac{\Phi^I \boldsymbol{\lambda}^{t_i} - \Phi^I \boldsymbol{\lambda}^{t_{i-1}}}{\Delta t} = \frac{1}{2}\overline{\mathcal{L}}\Phi^I \boldsymbol{\lambda}^{t_i} + \frac{1}{2}\overline{\mathcal{L}}\Phi^I \boldsymbol{\lambda}^{t_{i-1}}, \quad i = 1, 2, \ldots, M, \tag{3.20}$$

where $\boldsymbol{\lambda}^{t_i} = \boldsymbol{\lambda}(t_i)$. We can reorganize the above Equation (3.20) into the iteration expression

$$B\boldsymbol{\lambda}^{t_i} = D\boldsymbol{\lambda}^{t_{i-1}}. \tag{3.21}$$

Here B and D are $n \times N$ square matrices represented separately as

$$B = \Phi^I - \frac{1}{2}\overline{\mathcal{L}}\Phi^I \Delta t,$$

$$D = \Phi^I + \frac{1}{2}\overline{\mathcal{L}}\Phi^I \Delta t.$$

Previous researchers (Hon et al. [62]) applied a method called the Boundary Update Procedure (BUP) to force approximation values to satisfy boundary conditions. Here, we impose the boundary conditions (3.10) instead of the BUP as [105].

Suppose the matrix Φ^B applied on the boundary nodes $\Xi_2 = \{\mathbf{x}_{n+1}, \ldots, \mathbf{x}_N\}$ is

$$\Phi^B = \begin{bmatrix} \phi_{A,\mathbf{x}_1}(\mathbf{x}_{n+1}) & \phi_{A,\mathbf{x}_2}(\mathbf{x}_{n+1}) & \cdots & \phi_{A,\mathbf{x}_N}(\mathbf{x}_{n+1}) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{A,\mathbf{x}_1}(\mathbf{x}_N) & \phi_{A,\mathbf{x}_2}(\mathbf{x}_N) & \cdots & \phi_{A,\mathbf{x}_N}(\mathbf{x}_N) \end{bmatrix}. \tag{3.22}$$

So the boundary conditions at time step t_i is

$$\Phi^B \boldsymbol{\lambda}^{t_i} = \mathbf{g}^{t_i}, \tag{3.23}$$

where $\mathbf{g}^{t_i} = [g(t_i, \mathbf{x}_{n+1}), \ldots, g(t_i, \mathbf{x}_N)]^T$.

Combine the equations (3.21) and (3.23), we have

$$\begin{bmatrix} B \\ \Phi^B \end{bmatrix} \boldsymbol{\lambda}^{t_i} = \begin{bmatrix} D\boldsymbol{\lambda}^{t_{i-1}} \\ \mathbf{g}^{t_i} \end{bmatrix}. \tag{3.24}$$

If the parabolic problem is solved from the final time $t = b$, the iteration equation is

$$\begin{bmatrix} D \\ \Phi^B \end{bmatrix} \boldsymbol{\lambda}^{t_{i-1}} = \begin{bmatrix} B\boldsymbol{\lambda}^{t_i} \\ \mathbf{g}^{t_{i-1}} \end{bmatrix}. \tag{3.25}$$

3.2.2 Consider time as a spatial dimension

Considering time as a spatial dimension is known as the space-time method, which was firstly introduced into RBF field by Myers et al. in [90]. In order to describe this method clearly, we define some new notations firstly. In a parabolic problem we have a space domain $\Omega \subseteq \mathbb{R}^d$ and time interval $t \in [a, b]$, now suppose $\Omega_t = \Omega \times t \subseteq \mathbb{R}^d \times [a, b]$. Normally we only have conditions on boundary $\partial\Omega$ and at initial time given at starting time a or final time b, so in this section we use $\partial\Omega_t$ to represent the portion of boundary of Ω_t where the conditions are listed out in the problem. Correspondingly, there is a little change in uniformly distributed centre nodes $\Theta \in \Omega_t$. $\Theta = \Theta_1 \cup \Theta_2$, where $\Theta_1 = \{\boldsymbol{\eta_1}, \boldsymbol{\eta_2}, \ldots, \boldsymbol{\eta_n}\}$ are located in $\overline{\Omega}_t \setminus \partial\Omega_t$ and $\Theta_2 = \{\boldsymbol{\eta_{n+1}}, \boldsymbol{\eta_{n+2}}, \ldots, \boldsymbol{\eta_N}\}$ are on $\partial\Omega_t$.

Now our problem for target function u is displayed as:

$$\begin{aligned} \mathcal{L}_t u &= f, \quad in \quad \overline{\Omega}_t \setminus \partial\Omega_t, && (3.26) \\ u &= g, \quad on \quad \partial\Omega_t, && (3.27) \end{aligned}$$

where $\mathcal{L}_t$ is a parabolic operator, f, g are selected functions that makes u satisfies the PDE.

It is not necessary to explain how to solve this linear system as it is same to do a full grid RBF collocation (RBF-C) by Kansa's method introduced in Section 3.1. We also use $\hat{u}$ to represent the solution solved by the space-time method.

3.3 Numerical experiments

In 1973, the original analytical formula of non-dividend European options was proposed by Black and Scholes in [6]. Following this achievement, Roll extended the model to underlying stock paying dividends and introduced the first extension analytical formulation of the call option in [99]. After that, Geske [49] and Whaley [121] made corrections to the formulation separately in 1979 and 1981. The analytical solution of one asset European call option $C(t, S)$ is:

$$\begin{aligned} C(t,S) &= Se^{-q(T-t)}\mathcal{N}(d_1) - Ee^{-r(T-t)}\mathcal{N}(d_2), \\ d_1 &= \frac{\log(\frac{S}{E}) + (r - q + \frac{1}{2}\sigma^2)(T-t)}{\sigma\sqrt{T-t}}, \\ d_2 &= d_1 - \sigma\sqrt{T-t}, \end{aligned}$$

where $\mathcal{N}$ is the cumulative distribution function of the standard normal distribution, S means the stock price at time t, E is the strike price, σ is the volatility of the underlying stock, q is the continuous dividend payment for the asset, r is the risk-free rate and T means the maturity time.

The above option price $C(t, S)$ satisfies the following system on $[0, T] \times [0, +\infty)$

$$\frac{\partial C}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 C}{\partial C^2} + (r - q)S\frac{\partial C}{\partial S} - rC = 0, \quad t \in [0, T), \quad S \in [0, +\infty), \quad (3.28)$$

with initial condition and boundary conditions

$$\begin{aligned} C(T,S) &= \max\{S-E,0\}, \quad S\in[0,+\infty)\,, && (3.29)\\ C(t,0) &= 0, \quad t\in[0,T]\,, && (3.30)\\ \lim_{S\to+\infty} C(t,S) &= \lim_{S\to+\infty}\left(Se^{-q(T-t)}-Ee^{-r(T-t)}\right), \quad t\in[0,T]\,. && (3.31) \end{aligned}$$

Obviously, it is impossible to construct numerical results on an infinity domain. For computational purpose, we need to truncate the domain for stock price S from $[0,+\infty)$ to $[S_{min},S_{max}]$ even though there will be an unavoidable truncated error. However, according to the dominant diffusion process, it will reduce the influence from imperfect boundary condition. If we choose a domain large enough, the solution of the region of interest will not be affected by the truncation error. Let $u(t,S)$ stands for the one asset European call option value in the truncated domain $[S_{min},S_{max}]$. Hence $u(t,S)$ satisfies the following system on $\Omega_t=[0,T]\times[S_{min},S_{max}]$:

$$\frac{\partial u}{\partial t}+\frac{1}{2}\sigma^2S^2\frac{\partial^2u}{\partial S^2}+(r-q)S\frac{\partial u}{\partial S}-ru=0, \quad t\in[0,T)\,, \quad S\in[S_{min},S_{max}]\,, \tag{3.32}$$

with initial condition and boundary conditions

$$\begin{aligned} u(T,S) &= \max\{S-E,0\}, \quad S\in[S_{min},S_{max}]\,, && (3.33)\\ u(t,S_{min}) &= g_1(t), \quad t\in[0,T]\,, && (3.34)\\ u(t,S_{max}) &= g_2(t), \quad t\in[0,T]\,. && (3.35) \end{aligned}$$

Here, boundary conditions $g_1(t):=0$ and $g_2(t):=S_{max}e^{-q(T-t)}-Ee^{-r(T-t)}$. We also choose an enlarge spatial truncated domain $\left[\widetilde{S}_{min},\widetilde{S}_{max}\right]\supset[S_{min},S_{max}]$. Let $u^*(t,S)$ stands for the option value on the domain $\left[\widetilde{S}_{min},\widetilde{S}_{max}\right]$. The PDE system for u^* is very similar with the system for V, so we don't display that again.

In this section, we approximate the price of one asset non-dividend European call option following Parameter Set 1 taken from the BENCHOP [110] as Table 3.1. The numerical approximations are constructed by using the MOL and the space-time method. In order to compare the MOL and the space-time method

Parameter Values	
σ	0.15
r	0.03
T	1
E	100
q	0
S_{min}	0
S_{max}	3E

TABLE 3.1: Parameter Set 1 for non-dividend European call option.

more fair, in the following numerical results, center nodes are uniformly distributed on the domain $[0,T]\times[S_{min},S_{max}]$ and the domain $[0,T]\times\left[\widetilde{S}_{min},\widetilde{S}_{max}\right]$. In order to reduce changeable factors, all approximations are obtained with shape parameter $C=2$.

In the following tables, "Nodes" is the number of nodal points used for the approximation at different levels n. $\mathbf{T}$ is a testing sample points set which will be made explicit in each case. We measure the errors in the MOL and the space-time method at the level n respectively as

$$\begin{aligned} E^n_{\mathrm{MOL}} &= \max_{S\in\mathbf{T}} |C(0,S)-\hat{u}^n_{\mathrm{MOL}}(0,S)|, \\ E^n_{\mathrm{RBF-C}} &= \max_{S\in\mathbf{T}} |C(0,S)-\hat{u}^n(0,S)|. \end{aligned}$$

Correspondingly, let ρ stands for the slope of two adjacent points in different methods, for instance,

$$\rho_{\mathrm{MOL}} = \frac{\log\left(E^{n+1}_{\mathrm{MOL}}\right)-\log\left(E^n_{\mathrm{MOL}}\right)}{\log\left(\mathrm{Nodes}^{n+1}\right)-\log\left(\mathrm{Nodes}^n\right)},$$

here Nodes^n means the nodes number at the nth level.

When we apply the space-time method on the domain $\Omega_t=[S_{min},S_{max}]\times[0,T]$, we obtain the solution by solving Equations from (3.32) to (3.35). In Table 3.2, the max absolute errors are calculated on the sample points in the domain $[S_{min},S_{max}]=[0,3E]$ when we use the Gaussian basis function. We can see that the convergence rates for the MOL and the space-time method are both decreasing to zero. The bad performance of the collocation method using the Gaussian is also

Level	Nodes	E_{MOL}	ρ_{MOL}	$E_{\text{RBF-C}}$	$\rho_{\text{RBF-C}}$
1	9	12.3	—	11.8	—
2	25	7.01	-0.55	6.65	-0.56
3	81	5.10	-0.27	6.03	-0.08
4	289	5.48	0.06	6.97	0.11
5	1089	5.36	-0.02	7.11	0.02
6	4225	5.26	-0.01	7.07	-0.004
7	16641	5.20	-0.01	6.90	-0.02
8	66049	5.08	-0.02	6.82	-0.009

TABLE 3.2: The performance of the MOL and the space-time method using Gaussian on the spatial computational domain $[S_{min}, S_{max}] = [0, 3E]$. Error evaluated at 3000 uniform test points at time $t = 0$ on $[S_{min}, S_{max}] = [0, 3E]$.

Level	Nodes	E_{MOL}	ρ_{MOL}	$E_{\text{RBF-C}}$	$\rho_{\text{RBF-C}}$
1	9	15.9	—	15.7	—
2	25	6.40	-0.89	6.37	-0.88
3	81	1.87	-1.04	1.84	-1.06
4	289	2.64e-1	-1.54	2.46e-1	-1.58
5	1089	1.16e-1	-0.62	1.61e-1	-0.32
6	4225	5.73e-2	-0.52	9.26e-2	-0.41
7	16641	2.85e-2	-0.51	4.79e-2	-0.48
8	66049	1.39e-2	-0.52	2.39e-2	-0.50

TABLE 3.3: The performance of the MOL and the space-time method using MQ on the spatial computational domain $[S_{min}, S_{max}] = [0, 3E]$. Error evaluated at 3000 uniform test points at time $t = 0$ on $[S_{min}, S_{max}] = [0, 3E]$.

observed in Section 2.6, Chapter 4 and Chapter 5. In Table 3.3, the max absolute errors are also calculated on the sample points in the domain $[S_{min}, S_{max}] = [0, 3E]$ when we use the MQ. Both methods converge and the MOL seems to give more accurate solutions compared to the space-time method. However, the convergence rates based on the number of nodes for the MOL and the space-time method are both tending to be 0.5.

As this experiment is about option pricing, the truncation error has influence on the solutions, especially on the solutions near the boundary side. In Table 3.4, we show the performance of both methods using the MQ on the central region $\left[\widehat{S}_{min}, \widehat{S}_{max}\right] = [0.4E, 1.6E]$. The difference between Table 3.3 and Table 3.4 is only that the errors are calculated on the different testing points in different domains. It is obvious that both methods illustrate better performance for the

Level	Nodes	E_{MOL}	ρ_{MOL}	$E_{\text{RBF-C}}$	$\rho_{\text{RBF-C}}$
1	9	15.9	—	15.7	—
2	25	6.40	-0.89	6.37	-0.88
3	81	1.87	-1.04	1.84	-1.06
4	289	2.64e-1	-1.54	2.46e-1	-1.58
5	1089	7.06e-2	-0.99	6.33e-2	-1.02
6	4225	1.59e-2	-1.10	1.28e-2	-1.18
7	16641	4.11e-3	-0.99	2.69e-3	-1.14
8	66049	1.08e-3	-0.97	7.64e-4	-0.91

TABLE 3.4: The performance of the MOL and the space-time method using MQ on the spatial computational domain $[S_{min}, S_{max}] = [0, 3E]$. Error evaluated at 3000 uniform test points at time $t = 0$ on $\left[\widehat{S}_{min}, \widehat{S}_{max}\right] = [0.4E, 1.6E]$.

accuracy and the convergence rate. The rates based on the number of nodes tend to be 1 for both and the space-time method performs a little better in the accuracy. To have a visual view, we draw the performance of the MOL and the space-time method using MQ in Figure 3.1.

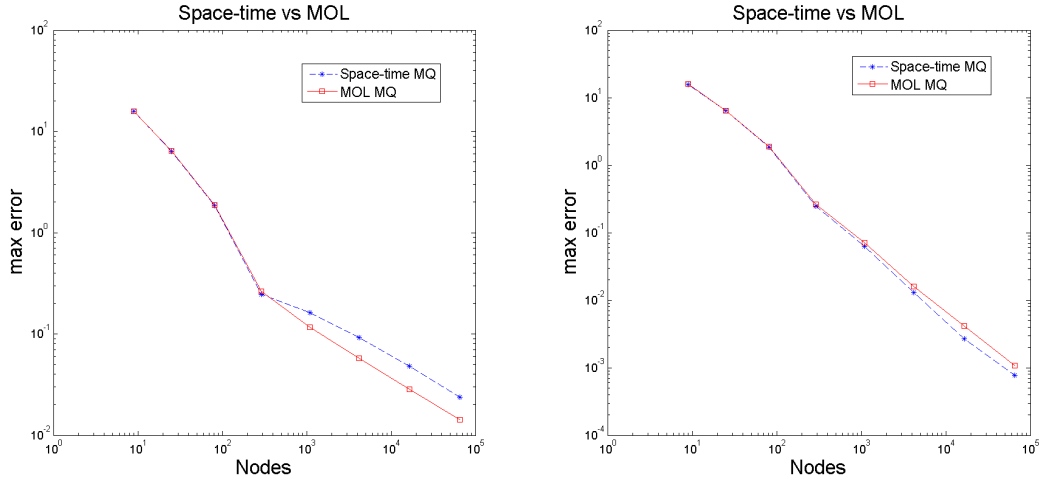

FIGURE 3.1: The performance of the space-time and the MOL using MQ on the spatial computational domain $[S_{min}, S_{max}] = [0, 3E]$. The left figure shows the max error at $t = 0$ on $[S_{min}, S_{max}] = [0, 3E]$, the right figure shows the max error at $t = 0$ on $\left[\widehat{S}_{min}, \widehat{S}_{max}\right] = [0.4E, 1.6E]$.

In Figure 3.2, we observe that the major errors in both figures are located near the boundaries. Meanwhile, the approximation using the MOL seems to be less influenced by the truncation error in the central region $\left[\widehat{S}_{min}, \widehat{S}_{max}\right] = [0.4E, 1.6E]$. This observation supports that choosing a large enough spatial computational domain is meaningful for option pricing. To have good approximations on the

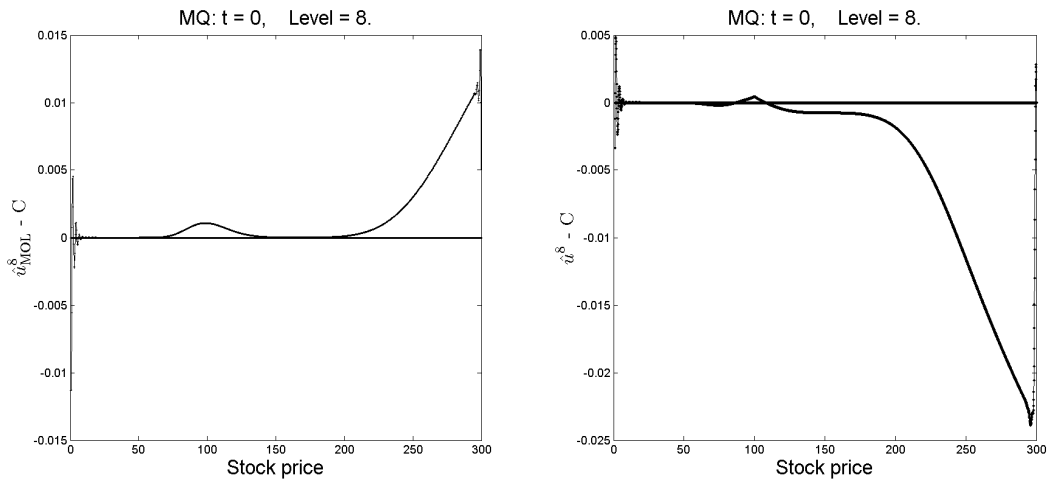

FIGURE 3.2: The performance of the MOL solution (left) and the space-time method (right) using MQ basis function on the spatial computational domain $[S_{min}, S_{max}] = [0, 3E]$. Errors are sampled on $[S_{min}, S_{max}] = [0, 3E]$ at $t = 0$.

whole domain $[S_{min}, S_{max}]$, we test using an enlarged region $\left[\widetilde{S}_{min}, \widetilde{S}_{max}\right] = [-E, 6E]$ and keep other parameter setting same as before. We define the boundary conditions for the domain $\left[\widetilde{S}_{min}, \widetilde{S}_{max}\right]$ as:

$$V^*(t, \widetilde{S}_{min}) = 0, \quad t \in [0, T], \tag{3.36}$$

$$V^*(t, \widetilde{S}_{max}) = \widetilde{S}_{max} e^{-q(T-t)} - E e^{-r(T-t)}, \quad t \in [0, T]. \tag{3.37}$$

As the MOL using MQ has similar performance as the space-time method using MQ, so we just display results from the MOL using MQ in the following table. For the MOL using MQ on the enlarged domain, we define the errors as E^*_{MOL} and the slope as ρ^*_{MOL} in Table 3.5.

Table 3.5 shows the performance of the MOL using MQ on different spatial computational domains: $[S_{min}, S_{max}] = [0, 3E]$ and $\left[\widetilde{S}_{min}, \widetilde{S}_{max}\right] = [-E, 6E]$. As we can see, choosing a larger spatial domain than the area that we are interested in yields a significant improvement based on the same cost of nodes.

In Figure 3.3, the errors are sampled at the region $[S_{min}, S_{max}]$ for levels 8 and 9. As we can see, the major errors locate in the kink around the strike price and there is a very good performance at the portions near S_{min} and S_{max}. Meanwhile, we obtain a better solution with more nodes. Based on these observations, in Chapter 5, we choose the MOL using MQ on an enlarged computational domain to approximate

Level	Nodes	E_{MOL}	ρ_{MOL}	E^*_{MOL}	ρ^*_{MOL}
1	9	15.9	—	51.1	—
2	25	6.40	-0.89	19.6	-0.94
3	81	1.87	-1.04	5.84	-1.03
4	289	2.64e-1	-1.54	2.87	-0.56
5	1089	1.16e-1	-0.62	3.45e-1	-1.60
6	4225	5.73e-2	-0.52	6.29e-2	-1.26
7	16641	2.85e-2	-0.51	3.10e-2	-0.52
8	66049	1.39e-2	-0.52	4.27e-3	-1.44

TABLE 3.5: The performance of the MOL using MQ on different spatial computational domain. Error evaluated at 3000 uniform test points at time $t = 0$ on $[S_{min}, S_{max}] = [0, 3E]$.

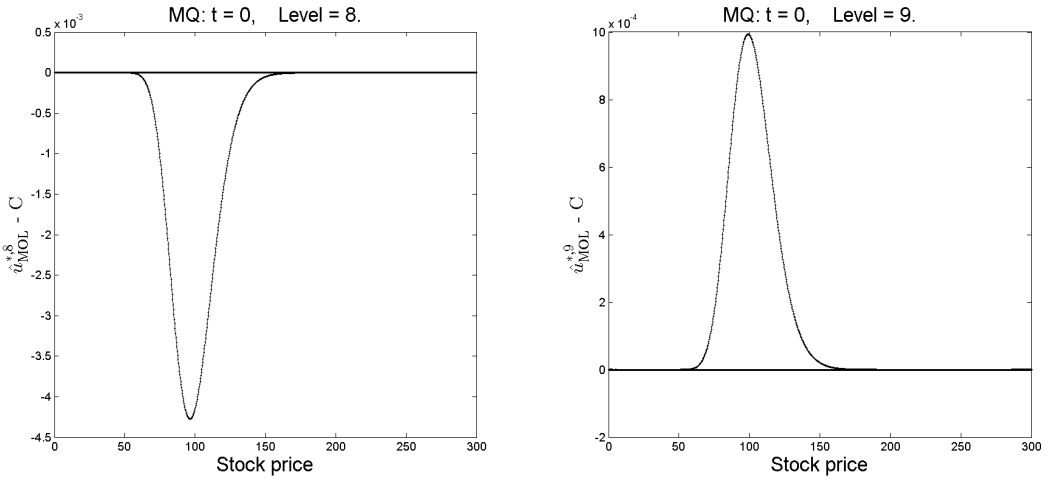

FIGURE 3.3: The performance of the MOL solution using MQ in the level 8 (left) and the level 9 (right) on the spatial computational domain $\left[\widetilde{S}_{min}, \widetilde{S}_{max}\right] = [-E, 6E]$. Errors are sampled on $[S_{min}, S_{max}] = [0, 3E]$ at $t = 0$.

the option price at an earlier time. A larger computational domain can block the truncation error from boundary side outside of our interesting domain. We could employ more spatial nodes to obtain better approximation in one spatial dimension problem, however, that will be a serious challenge in high dimensions.

3.4 Conclusion

In this chapter, we briefly review the Kansa method, the Method of Lines and the space-time method. From the numerical results, we can demonstrate that choosing a large enough computation domain yields a significant improvement in option

pricing. The space-time method using MQ can be directly used to solve parabolic problems, but the space-time method using the Gaussian cannot. When the nodal points are uniformly distributed, the space-time method using MQ has a similar convergence rate as the MOL using MQ in option pricing where an initial condition is non-smooth. Moreover, when using the MOL on a uniformly distributed grid, the number of nodes required is $\mathcal{O}(N^{d+1})$, here d is the spatial dimension and N means the nodes number in one direction. However, when we use sparse grids introduced in Chapter 4 in the space-time method, we only need $\mathcal{O}(N \log^d(N))$ nodes to construct the approximation. Even though we apply sparse grids in the spatial dimensions in the MOL. The number of nodes required is $\mathcal{O}(N^2 \log^{d-1}(N))$. We still save the memory cost with the space-time method. In Chapter 4, we show the performance of the sparse collocation method and the multilevel sparse collocation method in solving PDEs with smooth conditions.

Chapter 4

Multilevel sparse grid kernel collocation with RBFs

In the approximation field, high dimensional problems are always difficult because of the *curse of dimensionality*. Floater and Iske [36] proposed a multilevel interpolation scheme to circumvent this problem. The multilevel interpolation method requires decomposing the given data into a hierarchy of nested subsets. In [68, 69], Iske further studied the scheme and gave an efficient construction of such hierarchies. In [70], Iske and Levesley developed the multilevel scheme based on adaptive domain decomposition. Based on Floater-Iske setting, Narcowich, Schaback and Ward [91] demonstrated the multilevel method is a numerically stable method for the interpolation and gave some theoretical underpinnings. Further, Hales and Levesley [55] demonstrated the error estimates for the multilevel approximation using polyharmonic splines. Fasshauer and Jerome used the multilevel method with compactly supported radial basis functions (CSRBFs) to solve elliptic PDE in [28, 30]. In [26], Farrell and Wendland also used the multilevel RBF collocation method with CSRBFs to solve elliptic PDEs on bounded domains. Moreover, they demonstrated a convergence theory.

Another way to overcome the problem is the sparse grid method introduced by Zenger [126]. This method relies on a multi-scale basis via a tensor product construction and saves a massive amount of storage and memory cost without

loosing accuracy. Hemker [58] applied the finite volume method on sparse grids to solve three-dimensional elliptic problems. In [54], Griebel, Schneider and Zenger developed a combination technique for the sparse grid. They also demonstrated that the combination approach works for both smooth solutions and non-smooth solutions of linear problems, and even for non-linear problems. Griebel [52] employed finite difference in multilevel sparse grid method to solve elliptic PDEs. In 2013, Georgoulis, Levesley and Subhan [48] proposed an method called multilevel sparse grid kernel (MuSIK) for interpolation. Here, we extend this MuSIK method to the collocation problem.

4.1 Sparse grid kernel collocation

One of the advantages in using radial basis function is easy to construct even in high-dimensional problems. However, in order to achieve accuracy when dimension d is increasing, we have to fix the fill distance of full grid. That means the number of evenly distributed collocation points in every direction N is constant. As a result, the size of a full grid is growing exponentially as N^d. In contrast, the *sparse grid kernel* (SIK) algorithm which combines approximations based tensor product anisotropic radial basis functions on every sub-grid is a stable and efficient method when facing high dimension problem. The support of this matter is that under the assumption of sufficient smoothness of the data, the amount of nodes utilised can be reduced dramatically to guarantee a certain accuracy based on carefully constructed tensor product anisotropic basis function. Owing to the additional smoothness assumed, there is only a negligible loss of precision. The basic idea of SIK was first introduced about fifty years ago in [1, 106] and Zenger [126] proposed sparse grid methods in 1991.

4.1.1 Collocation with the combination technique

Schreiber discussed tensor product of one-dimensional RBFs applying directly sparse grid methods in her thesis [103], where numerical results corresponding to

the resulting method were not promising. On the other hand, the direct using of non-tensor product RBFs in the sparse grid setting is not straightforward, since the approximation spaces are characterised by basis functions with different anisotropic scaling in various directions. By utilising such scaling, the solution obtained from sparse grid method is infeasible as there is no guarantee about the well-posedness of the resulting kernel-based interpolation problems.

The strategy we adopt here is a sparse grid combination technique which was introduced in [54], afterwards this technique is operative in piecewise polynomial interpolation on sparse grids, for instance [12, 44, 46]. In sparse grid kernel collocation, the sparse grid is decomposed into a number of sub-grids firstly. In that case, all solutions that are constructed by solving collocation problems on each sub-grid are linearly combined to form a final solution on the sparse grid. The details about sparse grid kernel interpolation are discussed completely in [109], and here we present a particular case to introduce the collocation algorithm.

Suppose u is target function mapping from domain $\Omega \subseteq \mathbb{R}^d$ to $\mathbb{R}$. We define $\Omega = [0,1]^d$ in this section. Recall the collocation node set $\Xi = \Xi_1 \cup \Xi_2$, where $\Xi_1 = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ are interior points in Ω and $\Xi_2 = \{\mathbf{x}_{n+1}, \mathbf{x}_{n+2}, \ldots, \mathbf{x}_N\}$ are located on the boundary $\partial\Omega$. The approximation $\hat{u} : \Omega \to \mathbb{R}$ has a requirement to satisfy the collocation system for u:

$$\mathcal{L}\hat{u}(\mathbf{x}) = f(\mathbf{x}), \quad \mathbf{x} \in \Omega, \tag{4.1}$$

$$\hat{u}(\mathbf{x}) = g(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega. \tag{4.2}$$

We define a multi-index $\mathbf{l} = (l_1, l_2, \ldots, l_d) \in \mathbb{N}^d$ with condition $|\mathbf{l}|_1 = n + (d-1)$, here n represents the serial level and d is the number of dimensions. We use the symbol $\mathfrak{S}_{\mathbf{l}}^{n,d}$ to represent one decomposition of the sparse grid. The number of nodes of each sub-grid $\mathfrak{S}_{\mathbf{l}}^{n,d}$ in each direction are $\mathbf{N}^{\mathbf{l}} = 2^{\mathbf{l}}+\mathbf{1} := (2^{l_1}+1, \ldots, 2^{l_d}+1)$. The sub-grids are directionally uniform grids. In some directions fewer points may be needed and in others much more information is needed. The points $\mathbf{x}_{\mathbf{l},\mathbf{i}}$ of $\mathfrak{S}_{\mathbf{l}}^{n,d}$ are the points:

$$\mathbf{x}_{\mathbf{l},\mathbf{i}} := (x_{l_1,i_1}, \ldots, x_{l_d,i_d}),$$

where $x_{l_j,i_j} = i_j 2^{-l_j}$, for $i_j = 0, 1, 2, \ldots, 2^{l_j}$, $j = 1, \ldots, d$. Alternatively, the total amount of centres used in $\mathfrak{S}_{\mathbf{l}}^{n,d}$ can be presented by the following formula:

$$N_{\mathbf{l}}^{n,d} = \prod \mathbf{N}^{\mathbf{l}} = \prod_{i=1}^{d} (2^{l_i} + 1). \tag{4.3}$$

Furthermore, one sparse grid $\mathfrak{S}^{n,d}$ at level n and in d dimensions is the union of all possible grids $\mathfrak{S}_{\mathbf{l}}^{n,d}$:

$$\mathfrak{S}^{n,d} = \bigcup_{|\mathbf{l}|_1 = n+(d-1)} \mathfrak{S}_{\mathbf{l}}^{n,d}. \tag{4.4}$$

Here, we give an example of decomposition of sparse grid $\mathfrak{S}^{4,2}$ at level four and in two dimensions. The above equation reduces to

$$\mathfrak{S}^{4,2} = \bigcup_{l_1+l_2=5} \mathfrak{S}_{l_1,l_2}^{4,2}. \tag{4.5}$$

For a straightforward understanding, (4.5) is expressed in Figure 4.1.

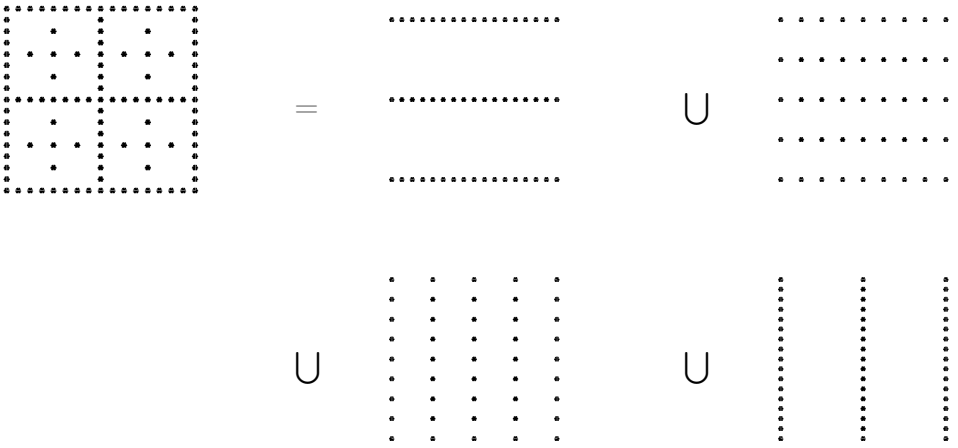

FIGURE 4.1: Sparse grid $\mathfrak{S}^{4,2}$ via (4.5).

However, if we construct approximation $\hat{u}$ based on the four sub-grids in Figure 4.1, it is clear that some nodes are utilised more than once. In order to fix this problem, we need to pick the redundant nodes out. In Figure 4.2, the first grid of the second row is the union of $\mathfrak{S}_{4,1}^{4,2}$ and $\mathfrak{S}_{3,2}^{4,2}$, and the first grid in the third row is the union of $\mathfrak{S}_{4,1}^{4,2}$, $\mathfrak{S}_{3,2}^{4,2}$ and $\mathfrak{S}_{2,3}^{4,2}$. The red points are redundancy nodes and the

red grids in the right-hand side column in Figure 4.2 are just sub-grids $\mathfrak{S}_{\mathbf{l}}^{3,2}$ of sparse grid $\mathfrak{S}^{3,2}$ at level three and in two dimensions.

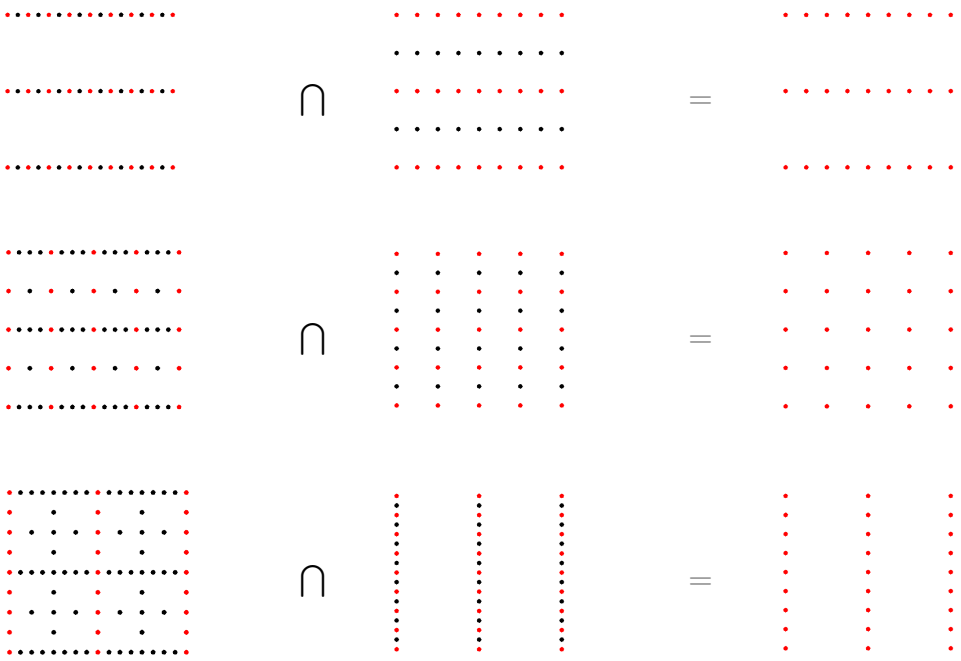

FIGURE 4.2: The redundant nodes of $\mathfrak{S}_{\mathbf{l}}^{4,2}$ in Figure 4.1.

As so far, we can formulate approximation $\hat{u}$ on the seven sub-grids in an arranged order as shown in Figure 4.3. This example is a particular case of the combination technique, and the combination formula will be displayed subsequently.

Now it is time to construct approximations on every coarser directionally uniform grids. As we employ tensor product RBFs in sparse grid kernel collocation (SIK-C), the scaling coefficients $A_{\mathbf{l}} \in \mathbb{R}^d$ for each multi-index $\mathbf{l}$ is defined as

$$A_{\mathbf{l}} := (2^{l_1}, 2^{l_2}, \ldots, 2^{l_d}).$$

We can solve the collocation system (4.1) and (4.2) on each sub grid $\mathfrak{S}_{\mathbf{l}}^{n,d}$ to have an approximation $\hat{u}_{\mathbf{l}}^{n,d}$. Here $\hat{u}_{\mathbf{l}}^{n,d}$ can be presented as

$$\hat{u}_{\mathbf{l}}^{n,d}(\mathbf{x}) = \sum_{i=1}^{N_{\mathbf{l}}^{n,d}} \lambda_i \phi_{A_{\mathbf{l}},\mathbf{x_i}}(\mathbf{x}). \tag{4.6}$$

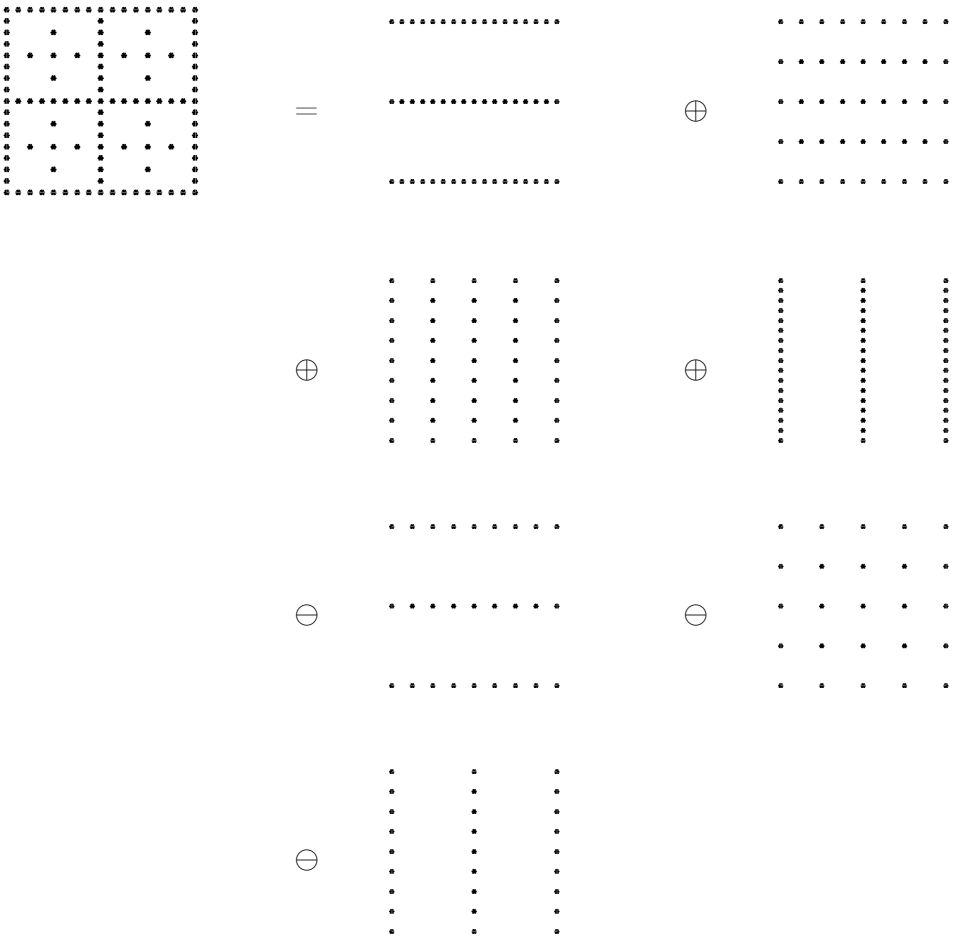

FIGURE 4.3: The construction of approximation $\hat{u}$ on $\mathfrak{S}^{4,2}$.

The final approximation $\hat{u}^{n,d}$ on sparse grid $\mathfrak{S}^{n,d}$ is constructed with these particular approximations $\hat{u}_{\mathbf{l}}^{n,d}$ following the rule named the combination formula [20, 44, 54]:

$$\hat{u}^{n,d}(\mathbf{x}) = \sum_{q=0}^{d-1} (-1)^q \begin{pmatrix} d-1 \\ q \end{pmatrix} \sum_{|\mathbf{l}|_1 = n+(d-1)-q} \hat{u}_{\mathbf{l}}^{n-q,d}(\mathbf{x}). \tag{4.7}$$

For the special case $d = 2$:

$$\hat{u}^{n,2}(\mathbf{x}) = \sum_{|\mathbf{l}|_1 = n+1} \hat{u}_{\mathbf{l}}^{n,2}(\mathbf{x}) - \sum_{|\mathbf{l}|_1 = n} \hat{u}_{\mathbf{l}}^{n-1,2}(\mathbf{x}), \tag{4.8}$$

when $n = 4$, it is the situation shown in Figure 4.3:

$$\begin{aligned}\hat{u}^{4,2}(\mathbf{x}) &= \hat{u}^{4,2}_{4,1}(\mathbf{x}) + \hat{u}^{4,2}_{3,2}(\mathbf{x}) + \hat{u}^{4,2}_{2,3}(\mathbf{x}) + \hat{u}^{4,2}_{1,4}(\mathbf{x}) \\ &- \hat{u}^{3,2}_{3,1}(\mathbf{x}) - \hat{u}^{3,2}_{2,2}(\mathbf{x}) - \hat{u}^{3,2}_{1,3}(\mathbf{x}).\end{aligned}$$

For $d = 3$:

$$\hat{u}^{n,3}(\mathbf{x}) = \sum_{|\mathbf{l}|_1=n+2} \hat{u}^{n,3}_{\mathbf{l}}(\mathbf{x}) - 2 \sum_{|\mathbf{l}|_1=n+1} \hat{u}^{n-1,3}_{\mathbf{l}}(\mathbf{x}) + \sum_{|\mathbf{l}|_1=n} \hat{u}^{n-2,3}_{\mathbf{l}}(\mathbf{x}). \tag{4.9}$$

For $d = 4$:

$$\begin{aligned}\hat{u}^{n,4}(\mathbf{x}) = &\sum_{|\mathbf{l}|_1=n+3} \hat{u}^{n,4}_{\mathbf{l}}(\mathbf{x}) - 3 \sum_{|\mathbf{l}|_1=n+2} \hat{u}^{n-1,4}_{\mathbf{l}}(\mathbf{x}) + \\ &3 \sum_{|\mathbf{l}|_1=n+1} \hat{u}^{n-2,4}_{\mathbf{l}}(\mathbf{x}) - \sum_{|\mathbf{l}|_1=n} \hat{u}^{n-3,4}_{\mathbf{l}}(\mathbf{x}).\end{aligned} \tag{4.10}$$

To sum up the above procedure, we describe the algorithm as Algorithm 1:

As we can see, one sparse grid cost less complexity than a corresponding full grid. Furthermore, the combination technique makes sparse grid decompose into sparse sub-grids. As a result, the memory and computation cost are reduced again. From the Equation (4.3), we notice that the size of each sub grid is $\mathcal{O}(2^{n+d-1})$. However, the number of nodes required in a corresponding full grid is $\mathcal{O}(2^{dn})$. It is obviously that our SIK-C algorithm save a lot of complexity compared to standard RBF algorithm, especially for the large class of data when $d > 2$. Another advantage of SIK-C is that it is possible to formulate approximation $\hat{u}^{n,d}_{\mathbf{l}}$ on each sub grid $\mathfrak{S}^{n,d}_{\mathbf{l}}$ with an individual code so that all approximations can be solved in a parallel system, see [44, 47, 51].

4.2 Multilevel sparse grid kernel collocation

For a significant amount of scattered data, a multilevel method is an efficient approach to accelerate convergence without breaking up the computational limitation.

Algorithm 1 Algorithm for Sparse grid kernel collocation

1. Input level n, dimension d for sparse grid $\mathfrak{S}^{n,d}$ and domain Ω.

2. Determine sequence $\mathbf{k} = \{n-d+1, n-d+2, \ldots, n\}$ that settle the amount of levels that we need to construct $\mathfrak{S}^{n,d}$.

3. Define multi-index set $\mathsf{L} = \{\mathsf{L}_{k_1}, \ldots, \mathsf{L}_{k_d}\}$, here $\mathsf{L}_{k_i} = \{\mathbf{l} = (l_1, \ldots, l_d) : |\mathbf{l}|_1 = k_i + d - 1\}$ for $i = 1, \ldots, d$.

4. Construct each directionally uniform sub grid $\mathfrak{S}_{\mathbf{l}}^{k_i,d}$ with evenly distributed nodes $\mathbf{N}^{\mathbf{l}} = \left(2^{l_1}+1, \ldots, 2^{l_d}+1\right)$ in different dimensions.

5. On each sub grid $\mathfrak{S}_{\mathbf{l}}^{k_i,d}$, formulate approximation $\hat{u}_{\mathbf{l}}^{k_i,d}$ to satisfy:

$$\begin{aligned} \mathcal{L}\hat{u}_{\mathbf{l}}^{k_i,d}(\mathbf{x}) &= f(\mathbf{x}), \quad \mathbf{x} \in \Omega, \\ \hat{u}_{\mathbf{l}}^{k_i,d}(\mathbf{x}) &= g(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega. \end{aligned}$$

6. Utilise combination formula

$$\hat{u}^{n,d}(\mathbf{x}) = \sum_{q=0}^{d-1} (-1)^q \begin{pmatrix} d-1 \\ q \end{pmatrix} \sum_{|\mathbf{l}|_1 = n+(d-1)-q} \hat{u}_{\mathbf{l}}^{n-q,d}(\mathbf{x})$$

to construct final approximation $\hat{u}^{n,d}$ on sparse grid $\mathfrak{S}^{n,d}$

Output $\hat{u}^{n,d}$.

The main idea of the multilevel method is to decompose scattered data into limited subspaces that form a nested sequence. Then in order to solve the collocation problem, all hierarchical approximations are summed up. On the coarsest subset approximation is formulated directly to original PDE model; other approximations are constructed to the PDE residuals on each subspace. The multilevel method is applied successfully in interpolation and collocation problems such as [26, 28, 30, 36, 68, 69, 70, 91, 93]. Error analysis of multilevel interpolation with RBFs is discussed in [30, 55].

4.2.1 Multilevel full grid collocation

Suppose $\mathbf{X}_i \subseteq \Omega \subseteq \mathbb{R}^d, i = 1, 2, \ldots, m$ are a sequence of nested full grids in domain Ω, each one has $2^i + 1$ evenly distributed nodes on one dimension, so they can be

presented as:

$$\mathbf{X}_1 \subseteq \mathbf{X}_2 \cdots \subseteq \mathbf{X}_m \subseteq \Omega \subseteq \mathbb{R}^d. \tag{4.11}$$

On the first grid $\mathbf{X}_1$, the approximation $\Delta\hat{u}_1$ is constructed to satisfy collocation system:

$$\begin{aligned} \mathcal{L}\Delta\hat{u}_1(\mathbf{x}) &= f(\mathbf{x}), \quad \mathbf{x} \in \mathbf{X}_1 \cap \Omega, \\ \Delta\hat{u}_1(\mathbf{x}) &= g(\mathbf{x}), \quad \mathbf{x} \in \mathbf{X}_1 \cap \partial\Omega. \end{aligned}$$

Approximation $\hat{u}_1 = \Delta\hat{u}_1$ on grid $\mathbf{X}_1$. On other grids $\mathbf{X}_i$ for $i \geq 2$, approximations $\Delta\hat{u}_i$ must satisfy the PDE residual system as:

$$\begin{aligned} \mathcal{L}\Delta\hat{u}_i(\mathbf{x}) &= f(\mathbf{x}) - \mathcal{L}\hat{u}_{i-1}(\mathbf{x}), \quad \mathbf{x} \in \mathbf{X}_i \cap \Omega, \\ \Delta\hat{u}_i(\mathbf{x}) &= g(\mathbf{x}) - \hat{u}_{i-1}(\mathbf{x}), \quad \mathbf{x} \in \mathbf{X}_i \cap \partial\Omega, \end{aligned}$$

here $\hat{u}_i = \hat{u}_{i-1} + \Delta\hat{u}_i, \quad i \geq 2$. Then let us use $\hat{u}^m_{\mathrm{ML}} = \hat{u}_m$ to represent the final approximation. The progress is drawn in following Figure 4.4.

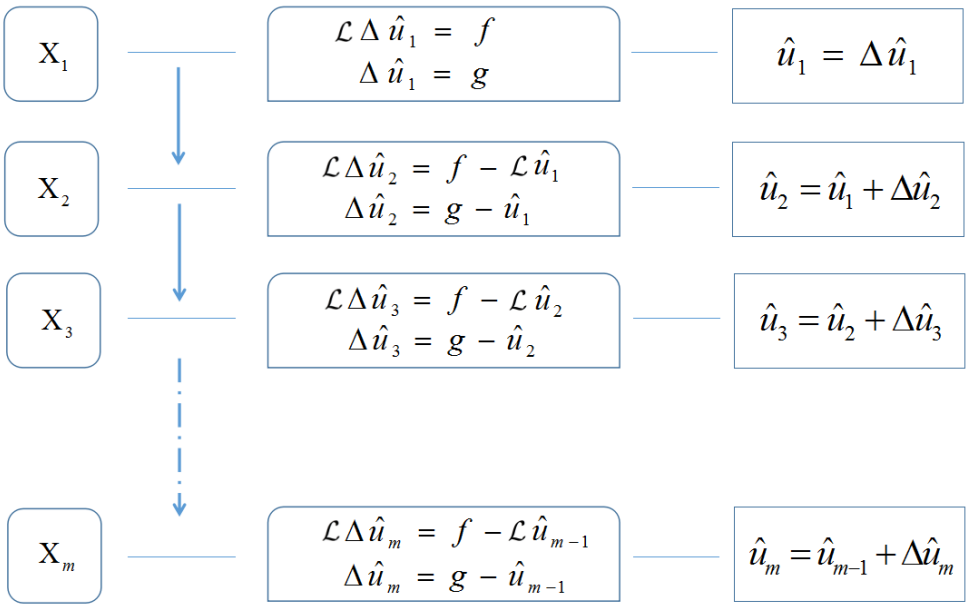

FIGURE 4.4: Multilevel full grid procedure.

4.2.2 Multilevel sparse grid kernel collocation

The essential idea of the multilevel method is to approximate the residuals in different levels. The setting of SIK-C is naturally suitable for a multilevel approximation algorithm. Firstly, the sparse grids from lower to higher level are nested, i.e., $\mathfrak{S}^{n,d} \subset \mathfrak{S}^{n+1,d}$ for $n \in \mathbb{N}$. Figure 4.5 shows a example in two dimensions, in which there are six nested sparse grids. Secondly, the anisotropic basis functions are scaled to fit the density of each particular sub-grid. That means the approximation spaces will not be nested for different levels. Finally, a multilevel algorithm requires the same number of nodal points and memory storage as SIK-C.

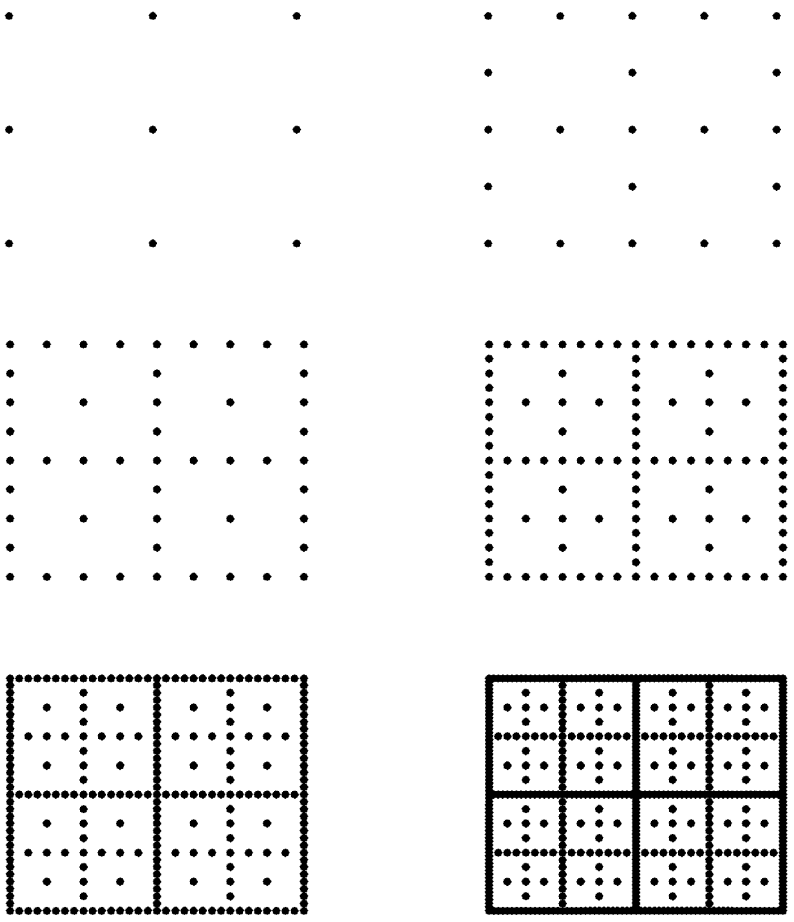

FIGURE 4.5: 6 nested sparse grids from $\mathfrak{S}^{1,2}$ to $\mathfrak{S}^{6,2}$.

The *multilevel sparse kernel-based collocation* (MuSIK-C, for short) algorithm is to solve the collocation problem on the coarsest sparse grid and then from the next level to solve on the residuals. Setting $\Delta_0 := 0$, for $k = 1, \ldots, n$, Δ_k is the sparse

grid approximation to the residuals $u - \sum_{j=0}^{k-1} \Delta_j$ on $\mathfrak{S}^{k,d}$ solved by collocation method from linear system:

$$\mathcal{L}\Delta_k = f - \sum_{j=0}^{k-1} \mathcal{L}\Delta_j, \tag{4.12}$$

$$\Delta_k = g - \sum_{j=0}^{k-1} \Delta_j. \tag{4.13}$$

The resulting multilevel sparse kernel based collocation is then given by

$$\hat{u}_{\mathrm{ML}}^{n,d} := \sum_{j=1}^{n} \Delta_j. \tag{4.14}$$

We summarise multilevel sparse grid kernel based collocation as Algorithm 2.

Algorithm 2 Algorithm for Multilevel sparse grid kernel based collocation

1. Input starting level m, final level n, dimension d and domain Ω.

2. Setting $\Delta_{m-1} := 0$.

for $i = m$ to n **do**

Reconstruct collocation system on $\mathfrak{S}^{i,d}$ as:

$$\mathcal{L}\Delta_i = f - \sum_{j=m-1}^{i-1} \mathcal{L}\Delta_j,$$

$$\Delta_i = g - \sum_{j=m-1}^{i-1} \Delta_j.$$

Recall Algorithm 1 to solve the above system.

Output residual approximation Δ_i.

end for

3. Sum up all residual approximations:

$$\hat{u}_{\mathrm{ML}}^{n,d} = \sum_{i=m}^{n} \Delta_i.$$

Output $\hat{u}_{\mathrm{ML}}^{n,d}$.

4.3 Numerical experiments

In this section, we mainly show the performance of MuSIK-C using MQ and Gaussian basis functions in solving elliptic and parabolic problems with smooth conditions up to 4 dimensions (including time). According to Schaback's uncertainty principle [101], we expect to have a better approximation with a larger condition number. As the condition number is depending on the choice of the connection constant parameter C, we illustrate the performance of our methods based on two connection constants ($C = 2$ and $C = 3$).

Because our implemented basis functions are tensor product, so there are two examples in the two-dimensional elliptic problem. In Example 4.1, we solve a PDE system with a non-tensor product target function. Here, we compare full grid RBF collocation (RBF-C), multilevel full grid RBF collocation (MLRBF-C), SIK-C and MuSIK-C. In Example 4.2, we approximate a tensor product target function in the two-dimensional elliptic problem. In this example, we compare MuSIK-C with a Q-basis tensor product FEM implemented in private communication Z. Dong [22]. In the three-dimensional (Example 4.3) and four-dimensional (Example 4.4) case, we compare MuSIK-C with a recent mesh-based method, called interior penalty discontinuous Galerkin (IPDG) method [116]. We also solve a four-dimensional elliptic problem whose analytical solution is non-tensor in Example 4.5.

In the parabolic problems, we firstly use MuSIK-C to approximate a non-tensor product function in one spatial dimension in Example 4.6. Then we compare MuSIK-C with another recent mesh-based method, called Isogeometric Analysis (IgA) method [74] in Example 4.7 and Example 4.8. In these two examples, the target functions are both tensor product. One is in two spatial dimensions and the other is in three spatial dimensions. Finally, we illustrate the performance of MuSIK-C in solving a three-dimensional spatial problem where the target function is non-tensor in Example 4.9.

In the following results, the condition (Cond) number of sparse grid is chosen to be the biggest from all condition numbers of every sub-grid in the same level. There are some notations to be declared. "Nodes" stands for the number of center points

that are used to construct estimations. Suppose u is the analytical solution. $\mathbf{T}$ is a testing sample points set whose elements are the Halton points [73]. $\mathbf{T}$ will be made explicit in each example. We measure the errors in RBF-C, MLRBF-C, SIK-C and MuSIK-C at level n respectively as

$$\begin{aligned} E^n_{\mathrm{RBF-C}}(\mathbf{x}) &= \max_{\mathbf{x}\in\mathbf{T}} |u(\mathbf{x}) - \hat{u}^n(\mathbf{x})|, \\ E^n_{\mathrm{MLRBF-C}}(\mathbf{x}) &= \max_{\mathbf{x}\in\mathbf{T}} |u(\mathbf{x}) - \hat{u}^n_{\mathrm{ML}}(\mathbf{x})|, \\ E^n_{\mathrm{SIK-C}}(\mathbf{x}) &= \max_{\mathbf{x}\in\mathbf{T}} \left|u(\mathbf{x}) - \hat{u}^{n,d}(\mathbf{x})\right|, \\ E^n_{\mathrm{MuSIK-C}}(\mathbf{x}) &= \max_{\mathbf{x}\in\mathbf{T}} \left|u(\mathbf{x}) - \hat{u}^{n,d}_{\mathrm{ML}}(\mathbf{x})\right|. \end{aligned}$$

Correspondingly, we define the slope ρ for two adjacent points in different methods, for instance

$$\rho_{\mathrm{RBF-C}} = \frac{\log\left(E^{n+1}_{\mathrm{RBF-C}}\right) - \log\left(E^n_{\mathrm{RBF-C}}\right)}{\log\left(\mathrm{Nodes}^{n+1}\right) - \log\left(\mathrm{Nodes}^n\right)},$$

here Nodes^n means the nodes number in the *nth* level.

4.3.1 Elliptic examples

Example 4.1. *In this example, we solve the following two-dimensional problem on* $\Omega = [0,1]^2$

$$\Delta u(\mathbf{x}) = -\pi^2 \sin(\pi x_1 x_2)(x_1^2 + x_2^2), \quad \mathbf{x} \in \Omega, \tag{4.15}$$

with boundary conditions

$$u(\mathbf{x}) = \sin(\pi x_1 x_2), \quad \mathbf{x} \in \partial\Omega. \tag{4.16}$$

The exact solution is a two-dimensional non-tensor product function

$$u(\mathbf{x}) = \sin(\pi x_1 x_2). \tag{4.17}$$

Level	Nodes	Cond	$E_{\text{MLRBF-C}}$	$\rho_{\text{MLRBF-C}}$	$E_{\text{RBF-C}}$	$\rho_{\text{RBF-C}}$
1	9	8e3	5.26e-2	—	5.26e-2	—
2	25	7e5	1.31e-2	-1.36	2.88e-2	-0.59
3	81	2e7	2.71e-3	-1.34	1.46e-2	-0.58
4	289	4e8	6.91e-4	-1.07	7.60e-3	-0.51
5	1089	8e9	1.75e-4	-1.04	3.96e-3	-0.49
6	4225	1e11	4.37e-5	-1.02	2.06e-3	-0.48

TABLE 4.1: MQ: Multilevel RBF collocation (MLRBF-C) and RBF collocation (RBF-C) on full grid with $C = 2$ for Example 4.1. Max error evaluated at 64,000 Halton points in the whole domain.

Level	Nodes	Cond	$E_{\text{MLRBF-C}}$	$\rho_{\text{MLRBF-C}}$	$E_{\text{RBF-C}}$	$\rho_{\text{RBF-C}}$
1	9	1e5	3.65e-2	—	3.65e-2	—
2	25	3e7	7.08e-3	-1.61	1.12e-2	-1.16
3	81	2e9	1.83e-3	-1.15	5.41e-3	-0.62
4	289	6e10	4.35e-4	-1.13	2.95e-3	-0.48
5	1089	1e12	9.38e-5	-1.16	1.59e-3	-0.47
6	4225	2e13	1.90e-5	-1.18	8.38e-4	-0.47

TABLE 4.2: MQ: Multilevel RBF collocation and RBF collocation on full grid with $C = 3$ for Example 4.1. Max error evaluated at 64,000 Halton points in the whole domain.

From Tables 4.1 and 4.2, it is obvious that the multilevel method really offers advantages in the solutions. The difference between these two tables is that there are two values of the connection constant parameter ($C = 2$ and 3). When C is bigger, the condition number grows faster. The convergence rate is faster and solutions are more accurate as we expected. In Tables 4.3 and 4.4, we use the Gaussian basis function in place of MQ. Both tables also demonstrate the superiority of the multilevel method and that RBF collocation with the Gaussian does not converge. In particular, when $C = 3$ the performance of multilevel RBF collocation with the Gaussian in Table 4.4 is better than the performance of multilevel RBF collocation with MQ in Table 4.2. Meanwhile, the corresponding condition numbers are also much bigger.

Level	Nodes	Cond	$E_{\text{MLRBF}-\text{C}}$	$\rho_{\text{MLRBF}-\text{C}}$	$E_{\text{RBF}-\text{C}}$	$\rho_{\text{RBF}-\text{C}}$
1	9	1e3	5.63e-2	—	5.63e-2	—
2	25	8e4	9.34e-3	-1.76	2.20e-2	-0.92
3	81	4e6	2.23e-3	-1.22	2.11e-2	-0.03
4	289	6e7	5.61e-4	-1.08	2.31e-2	0.07
5	1089	3e8	1.42e-4	-1.04	2.43e-2	0.04
6	4225	1e9	3.57e-5	-1.02	2.48e-2	0.02

TABLE 4.3: Gaussian: Multilevel RBF collocation and RBF collocation on full grid with $C = 2$ for Example 4.1. Max error evaluated at 64,000 Halton points in the whole domain.

Level	Nodes	Cond	$E_{\text{MLRBF}-\text{C}}$	$\rho_{\text{MLRBF}-\text{C}}$	$E_{\text{RBF}-\text{C}}$	$\rho_{\text{RBF}-\text{C}}$
1	9	2e4	3.00e-2	—	3.00e-2	—
2	25	2e7	4.47e-3	-1.86	5.60e-3	-1.64
3	81	5e10	6.09e-4	-1.70	2.53e-3	-0.67
4	289	2e14	8.54e-5	-1.54	2.22e-3	-0.10
5	1089	2e16	1.21e-5	-1.47	2.26e-3	0.01
6	4225	6e17	1.67e-6	-1.46	2.32e-3	0.02

TABLE 4.4: Gaussian: Multilevel RBF collocation and RBF collocation on full grid with $C = 3$ for Example 4.1. Max error evaluated at 64,000 Halton points in the whole domain.

Level	Nodes	Cond	$E_{\text{MuSIK}-\text{C}}$	$\rho_{\text{MuSIK}-\text{C}}$	$E_{\text{SIK}-\text{C}}$	$\rho_{\text{SIK}-\text{C}}$
2	21	1e5	4.51e-2	—	4.51e-2	—
3	49	7e5	1.61e-2	-1.21	1.69e-2	-1.16
4	113	8e6	3.85e-3	-1.71	9.55e-3	-0.68
5	257	5e7	8.66e-4	-1.82	5.91e-3	-0.58
6	577	2e8	2.09e-4	-1.76	4.15e-3	-0.44
7	1281	9e8	5.17e-5	-1.75	2.93e-3	-0.44
8	2817	4e9	1.27e-5	-1.78	2.11e-3	-0.42
9	6145	2e10	3.13e-6	-1.80	1.50e-3	-0.43
10	13313	6e10	7.68e-7	-1.81	1.08e-3	-0.43
11	28673	3e11	1.88e-7	-1.83	7.69e-4	-0.44
12	61441	1e12	4.62e-8	-1.84	5.27e-4	-0.50

TABLE 4.5: Multilevel sparse collocation compared with sparse collocation for MQ for Example 4.1 with $C = 2$. Max error evaluated at 64,000 Halton points in the whole domain.

Tables 4.5 and 4.6 show that the sparse grid collocation has the same performance as the full grid collocation. In Table 4.6, we can see the performance of SIK-C and MuSIK-C where we using MQ basis function when $C = 3$. Both have large

Level	Nodes	Cond	$E_{\text{MuSIK-C}}$	$\rho_{\text{MuSIK-C}}$	$E_{\text{SIK-C}}$	$\rho_{\text{SIK-C}}$
2	21	3e6	2.78e-2	—	2.78e-2	—
3	49	6e7	7.74e-3	-1.51	6.58e-3	-1.70
4	113	6e8	1.15e-3	-2.28	2.66e-3	-1.08
5	257	2e9	2.03e-4	-2.11	1.85e-3	-0.45
6	577	2e10	4.02e-5	-2.00	1.43e-3	-0.32
7	1281	1e11	8.59e-6	-1.93	1.07e-3	-0.36
8	2817	5e11	1.83e-6	-1.96	7.82e-4	-0.40
9	6145	2e12	3.73e-7	-2.04	5.76e-4	-0.39
10	13313	9e12	7.52e-8	-2.07	4.16e-4	-0.42
11	28673	4e13	1.54e-8	-2.07	2.98e-4	-0.43
12	61441	1e14	9.80e-9	-0.59	2.13e-4	-0.44

TABLE 4.6: Multilevel sparse collocation compared with sparse collocation for MQ for Example 4.1 when $C = 3$. Max error evaluated at 64,000 Halton points in the whole domain.

condition numbers ($1e14$) at the last level. However, SIK-C seems to keep its convergence while MuSIK-C starts to stop. The condition number is a very important index for the stability of the system but it is not the only determining factor to get accurate approximations. Even though the system is ill-conditioned, we can also achieve a good approximation. However, the solutions might not be such reliable.

Level	Nodes	Cond	$E_{\text{MuSIK-C}}$	$\rho_{\text{MuSIK-C}}$	$E_{\text{SIK-C}}$	$\rho_{\text{SIK-C}}$
2	21	2e4	2.82e-2	—	2.82e-2	—
3	49	8e4	1.69e-2	-0.60	2.17e-2	-0.31
4	113	2e6	4.44e-3	-1.60	2.32e-2	0.08
5	257	3e7	9.69e-4	-1.85	2.43e-2	0.06
6	577	2e8	2.43e-4	-1.71	2.48e-2	0.02
7	1281	8e8	6.21e-5	-1.72	2.51e-2	0.02
8	2817	3e9	1.57e-5	-1.74	2.52e-2	0.01
9	6145	1e10	3.93e-6	-1.78	2.52e-2	-0.00
10	13313	5e10	9.88e-7	-1.79	2.47e-2	-0.03
11	28673	2e11	2.44e-7	-1.82	2.52e-2	0.03
12	61441	9e11	5.92e-8	-1.86	2.43e-2	-0.05

TABLE 4.7: Multilevel sparse collocation compared with sparse collocation using the Gaussian for Example 4.1 when $C = 2$. Max error evaluated at 64,000 Halton points in the whole domain.

Similarly, we use the Gaussian basis functions instead of MQ in Tables 4.7 and 4.8. SIK-C with the Gaussian also does not converge. When $C = 3$, the condition

Level	Nodes	Cond	$E_{\text{MuSIK-C}}$	$\rho_{\text{MuSIK-C}}$	$E_{\text{SIK-C}}$	$\rho_{\text{SIK-C}}$
2	21	9e5	2.31e-2	—	2.31e-2	—
3	49	3e8	5.98e-3	-1.60	5.87e-3	-1.62
4	113	4e9	4.75e-4	-3.03	2.74e-3	-0.91
5	257	5e10	5.51e-5	-2.62	2.36e-3	-0.18
6	577	2e13	8.02e-6	-2.38	2.35e-3	-0.00
7	1281	2e15	1.12e-6	-2.47	2.36e-3	0.01
8	2817	2e16	1.50e-7	-2.55	2.37e-3	0.01
9	6145	3e17	1.99e-8	-2.59	2.37e-3	-0.00
10	13313	3e18	2.61e-9	-2.63	2.34e-3	-0.02
11	28673	1e19	1.59e-8	2.36	7.12e-3	1.45
12	61441	4e20	41	28	8e8	33

TABLE 4.8: Multilevel sparse collocation compared with sparse collocation using the Gaussian for Example 4.1 when $C = 3$. Max error evaluated at 64,000 Halton points in the whole domain.

number of MuSIK-C with the Gaussian reaches $1e19$ and $4e20$ at the levels 11 and 12. According to the ill-condition problem, the performance of MuSIK-C with the Gaussian breaks down at the levels 11 and 12. However, the condition number is $3e18$ at the level 10 and we also have an improving estimation.

From the Figure 4.6, we observe that SIK-C and RBF-C with MQ basis functions converge slowly and they have similar performance. Meanwhile, those two methods with the Gaussian do not converge. This phenomenon was also observed in the interpolation with RBFs, such as [115]. In Figure 4.6, MuSIK-C is always faster and more accurate than the other three methods. The illustration clearly shows the advantages of using the multilevel methods and the sparse version in particular. In the remaining examples, we only show the solutions from MuSIK-C.

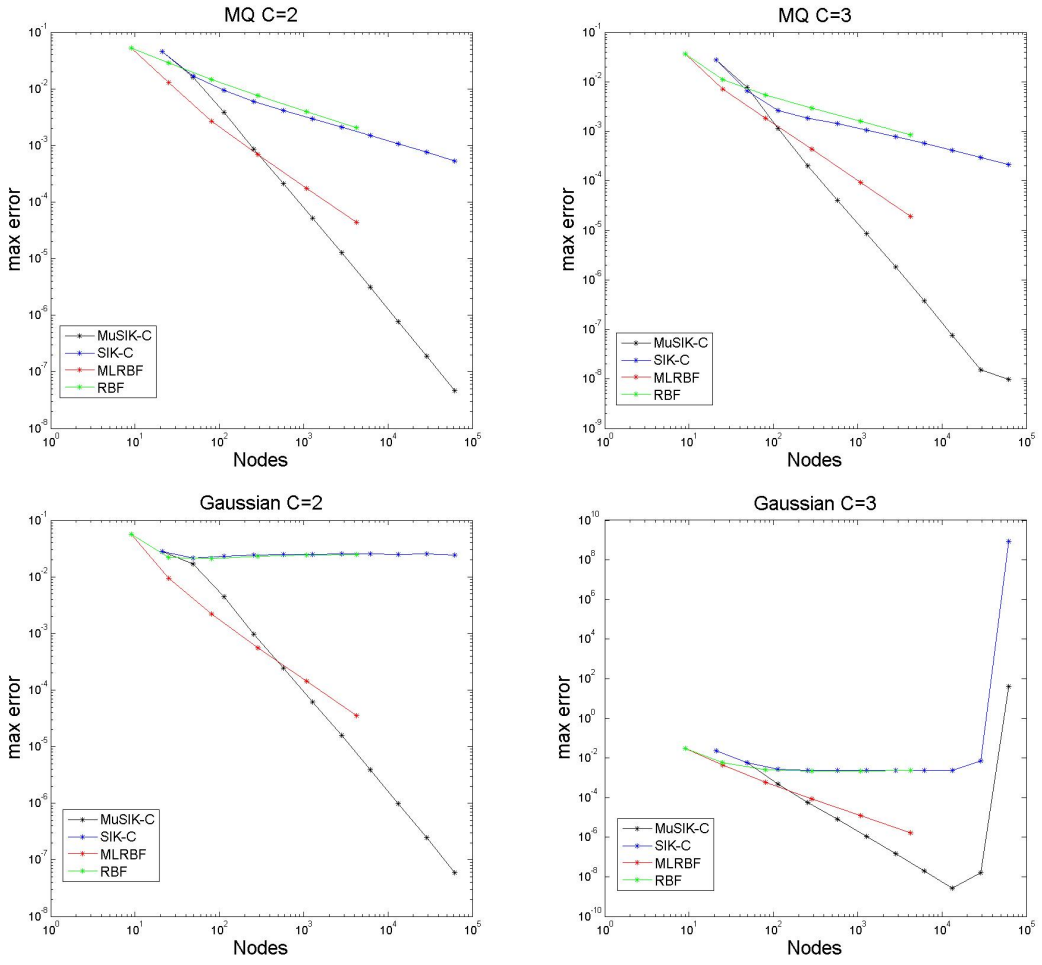

FIGURE 4.6: The performance of MLRBF-C, RBF-C, MuSIK-C and SIK-C with different basis functions and shape parameters for Example 4.1.

Example 4.2. *In this example, we solve the following two-dimensional problem on* $\Omega = [0,1]^2$

$$\Delta u(\mathbf{x}) = -2\pi^2 \sin(\pi x_1)\cos(\pi x_2), \quad \mathbf{x} \in \Omega, \tag{4.18}$$

with boundary conditions

$$u(\mathbf{x}) = \sin(\pi x_1)\cos(\pi x_2), \quad \mathbf{x} \in \partial\Omega. \tag{4.19}$$

The exact solution is a two-dimensional tensor product function

$$u(\mathbf{x}) = \sin(\pi x_1)\cos(\pi x_2). \tag{4.20}$$

Level	Nodes	$E_{\text{MuSIK}-\text{C}}$ (MQ)	$\rho_{\text{MuSIK}-\text{C}}$ (MQ)	$E_{\text{MuSIK}-\text{C}}$ (G)	$\rho_{\text{MuSIK}-\text{C}}$ (G)
2	21	2.81e-2	—	1.09e-2	—
3	49	7.53e-3	-1.55	5.01e-3	-0.91
4	113	2.07e-3	-1.55	1.70e-3	-1.29
5	257	5.33e-4	-1.65	5.50e-4	-1.38
6	577	1.33e-4	-1.71	1.62e-4	-1.51
7	1281	3.31e-5	-1.75	4.47e-5	-1.61
8	2817	8.14e-6	-1.78	1.18e-5	-1.69
9	6145	1.99e-6	-1.80	3.01e-6	-1.75
10	13313	4.89e-7	-1.82	7.55e-7	-1.79
11	28673	1.17e-7	-1.86	1.89e-7	-1.80
12	61441	2.83e-8	-1.86	4.69e-8	-1.83

TABLE 4.9: The performance of the multilevel sparse collocation method using MQ and Gaussian for Example 4.2 with $C = 2$. Max error evaluated at 64,000 Halton points in the whole domain.

Here the focus is on multilevel sparse collocation with MQ and Gaussian with two different constants C. In Table 4.10, MuSIK-C breaks down at the levels 11 and 12. However, the convergence rate increases slowly before the ill-conditioning problem arises. In Figure 4.7, the FEM is implemented by my colleague Z. Dong [22]. The FEM is based on the Q-basis tensor product polynomials on a full grid. In Figure 4.7 Qp refers to a degree p polynomial in each direction for the Q basis method. We can see that the slope for the dashed line is almost growing as $\frac{p+1}{2}$. That means the convergence order is increasing with polynomial order p. Similarly, the convergence rate of MuSIK-C can be accelerated by increasing

Level	Nodes	$E_{\text{MuSIK-C}}$ (MQ)	$\rho_{\text{MuSIK-C}}$ (MQ)	$E_{\text{MuSIK-C}}$ (G)	$\rho_{\text{MuSIK-C}}$ (G)
2	21	1.24e-2	—	1.20e-2	—
3	49	4.02e-3	-1.33	1.08e-3	-2.85
4	113	8.54e-4	-1.85	1.45e-4	-2.40
5	257	1.89e-4	-1.83	1.84e-5	-2.51
6	577	4.12e-5	-1.88	2.41e-6	-2.51
7	1281	8.59e-6	-1.97	3.25e-7	-2.51
8	2817	1.74e-6	-2.03	4.47e-8	-2.52
9	6145	3.45e-7	-2.07	5.98e-9	-2.58
10	13313	6.69e-8	-2.12	8.05e-10	-2.59
11	28673	1.32e-8	-2.11	3.64e-8	4.97
12	61441	6.72e-9	-0.89	8	25

TABLE 4.10: The performance of the multilevel sparse collocation method using MQ and Gaussian for Example 4.2 with $C = 3$. Max error evaluated at 64,000 Halton points in the whole domain.

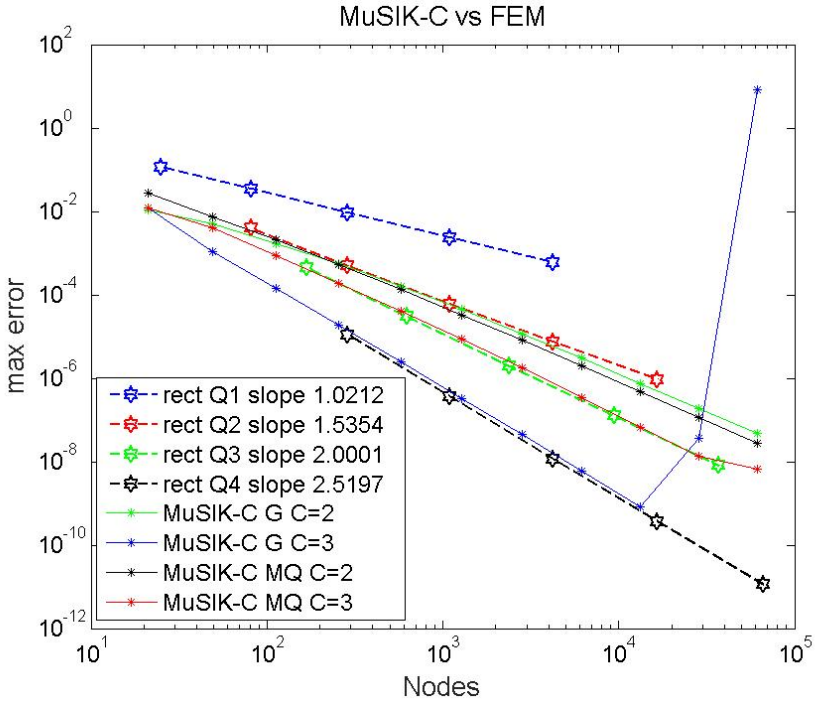

FIGURE 4.7: The performance of MuSIK-C and FEM for Example 4.2.

the shape parameter. The performance of MuSIK-C using both basis functions with $C = 2$ is better than the performance of the FEM with $p = 2$, for accuracy and convergence rate. With constant $C = 3$, MuSIK-C using MQ has similar performance with the FEM with $p = 3$. Moreover, MuSIK-C using the Gaussian has similar performance with the FEM with $p = 4$. However, MuSIK-C using both basis functions with $C = 3$ breaks down in the levels 11 and 12 because of

the ill-condition. This phenomenon demonstrates it is quite significant to reduce the condition number while utilising our MuSIK-C method. Therefore, one of our future research directions is to explore the use of preconditioning schemes.

Example 4.3. *In this example, we solve the following three-dimensional problem on* $\Omega = [0,1]^3$

$$\Delta u(\mathbf{x}) = 0, \quad \mathbf{x} \in \Omega, \tag{4.21}$$

with boundary conditions

$$u(\mathbf{x}) = \sin(\pi x_1)\sin(\pi x_2)\frac{\sinh(\sqrt{2}\pi x_3)}{\sinh(\sqrt{2}\pi)}, \quad \mathbf{x} \in \partial\Omega. \tag{4.22}$$

The exact solution is a three-dimensional tensor product function

$$u(\mathbf{x}) = \sin(\pi x_1)\sin(\pi x_2)\frac{\sinh(\sqrt{2}\pi x_3)}{\sinh(\sqrt{2}\pi)}. \tag{4.23}$$

Level	Nodes	MQ C=2	MQ C=3	Gaussian C=2	Gaussian C=3
3	225	3e8	4e10	1e7	4e10
4	593	2e9	7e11	2e8	1e13
5	1505	2e10	1e13	1e9	8e14
6	3713	1e11	1e14	9e9	3e16
7	8961	6e11	8e14	6e10	8e18
8	21249	3e12	6e15	3e11	5e20

TABLE 4.11: Sparse grid condition number for MQ and Gaussian with the two considered constants: 2 and 3, in the three dimensional Poisson problem.

Level	Nodes	$E_{\text{MuSIK-C}}$ (MQ)	$\rho_{\text{MuSIK-C}}$ (MQ)	$E_{\text{MuSIK-C}}$ (G)	$\rho_{\text{MuSIK-C}}$ (G)
3	225	3.01e-2	—	5.55e-2	—
4	593	7.83e-3	-1.39	1.39e-2	-1.43
5	1505	1.92e-3	-1.51	3.52e-3	-1.47
6	3713	3.49e-4	-1.89	6.89e-4	-1.81
7	8961	9.30e-5	-1.50	1.78e-4	-1.54
8	21249	2.29e-5	-1.62	4.49e-5	-1.60

TABLE 4.12: The performance of the multilevel sparse collocation method using MQ and Gaussian for Example 4.3 with $C = 2$. Max error evaluated at 120,000 Halton points in the whole domain.

Level	Nodes	$E_{\text{MuSIK-C}}$ (MQ)	$\rho_{\text{MuSIK-C}}$ (MQ)	$E_{\text{MuSIK-C}}$ (G)	$\rho_{\text{MuSIK-C}}$ (G)
3	225	2.15e-2	—	1.72e-2	—
4	593	3.74e-3	-1.80	1.79e-3	-2.34
5	1505	7.64e-4	-1.70	2.43e-4	-2.14
6	3713	1.17e-4	-2.08	2.78e-5	-2.40
7	8961	2.39e-5	-1.80	3.67e-6	-2.30
8	21249	4.67e-6	-1.89	9.74e-5	3.80

TABLE 4.13: The performance of the multilevel sparse collocation method using MQ and Gaussian for Example 4.3 with $C = 3$. Max error evaluated at 120,000 Halton points in the whole domain.

Here the focus is on multilevel sparse collocation with MQ and Gaussian with two different constants C. The convergence rate here seems to be growing. In Table 4.13, MuSIK-C with the Gaussian blows up more quickly.

In 2016, Wang et al. [116] developed an interior penalty discontinuous Galerkin (IPDG) method based on sparse grid to solve high-dimensional elliptic problems. The key idea is that IPDG utilises discontinuous elements over the computational domain, so the choice of local polynomial basis is flexible; see [14] for details. The authors used a hierarchical basis representation to construct a sparse finite element approximation space. The resulting degrees of freedom is substantially reduced without deteriorating the order of convergence too much. In particularly, the IPDG method can achieve accuracy of $\mathcal{O}\left(h^p|\log_2 h|^{d-1}\right)$ in the energy norm for d-dimensional problem, where p is the degree of polynomials used and h is the uniform mesh size in each dimension. In this example, the numerical results of IPDG are taken from [116]. In Figure 4.8, MuSIK-C with $C = 2$ performs much better than the IPDG when polynomial order $p = 1$. MuSIK-C with $C = 3$ converges faster and with more accuracy than IPDG when $p = 2$.

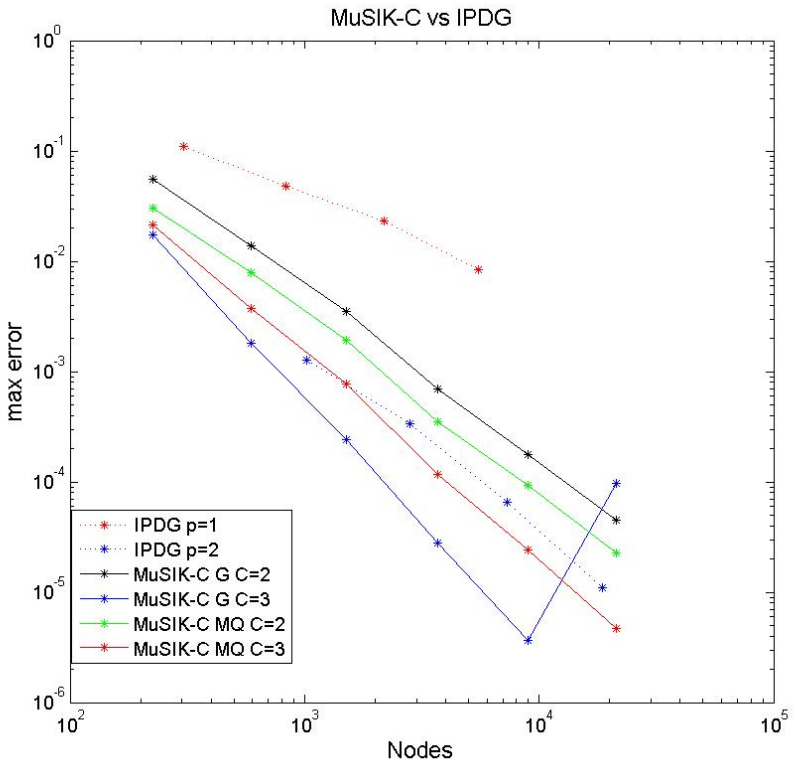

FIGURE 4.8: The performance of multilevel sparse collocation with MQ/Gaussian and sparse grid IPDG for Example 4.3.

Example 4.4. *In this example, we solve the following four-dimensional problem on* $\Omega = [0,1]^4$

$$\Delta u(\mathbf{x}) = 0, \quad \mathbf{x} \in \Omega, \tag{4.24}$$

with boundary conditions

$$u(\mathbf{x}) = \sin(\pi x_1)\sin(\pi x_2)\sin(\pi x_3)\frac{\sinh(\sqrt{3}\pi x_4)}{\sinh(\sqrt{3}\pi)}, \quad \mathbf{x} \in \partial\Omega. \tag{4.25}$$

The exact solution is a four-dimensional tensor product function

$$u(\mathbf{x}) = \sin(\pi x_1)\sin(\pi x_2)\sin(\pi x_3)\frac{\sinh(\sqrt{3}\pi x_4)}{\sinh(\sqrt{3}\pi)}. \tag{4.26}$$

In Table 4.14, we see the conditions all grow rapidly for MQ and Gaussian. As we explained before, even with a big condition number, it is also possible to achieve good estimations, but it might be not very reliable. In Table 4.15, the focus is on multilevel sparse collocation with MQ and Gaussian with the same connection constants $C = 2$. The convergence rate is also growing slowly. In Figure 4.9,

Level	Nodes	MQ C=2	Gaussian C=2
4	2769	4e11	4e9
5	7681	3e12	4e10
6	20481	3e13	3e11
7	52993	2e14	2e12
8	133889	1e15	2e13

TABLE 4.14: Illustration of how condition number grows on sparse grids for MQ and Gaussian with a connection consatant of $C = 2$ in the four dimensional Poisson problem.

Level	Nodes	$E_{\text{MuSIK-C}}$ (MQ)	$\rho_{\text{MuSIK-C}}$ (MQ)	$E_{\text{MuSIK-C}}$ (G)	$\rho_{\text{MuSIK-C}}$ (G)
4	2769	1.62e-2	—	4.26e-2	—
5	7681	3.93e-3	-1.39	1.02e-2	-1.40
6	20481	9.62e-4	-1.44	2.52e-3	-1.43
7	52993	2.42e-4	-1.45	6.40e-4	-1.44
8	133889	5.91e-5	-1.52	1.58e-4	-1.51

TABLE 4.15: The performance of the multilevel sparse collocation method using MQ and Gaussian for Example 4.4 with $C = 2$. Max error evaluated at 240,000 Halton points in the whole domain.

the numerical results are also taken from [116]. MuSIK-C with $C = 2$ show a rapid convergence rate, faster than the IPDG method with $p = 2$. In Figure 4.8, MuSIK-C with $C = 2$ performs better than the IPDG method with $p = 2$. That means even though the dimension is increasing, the performance of our method compared to others is not reduced.

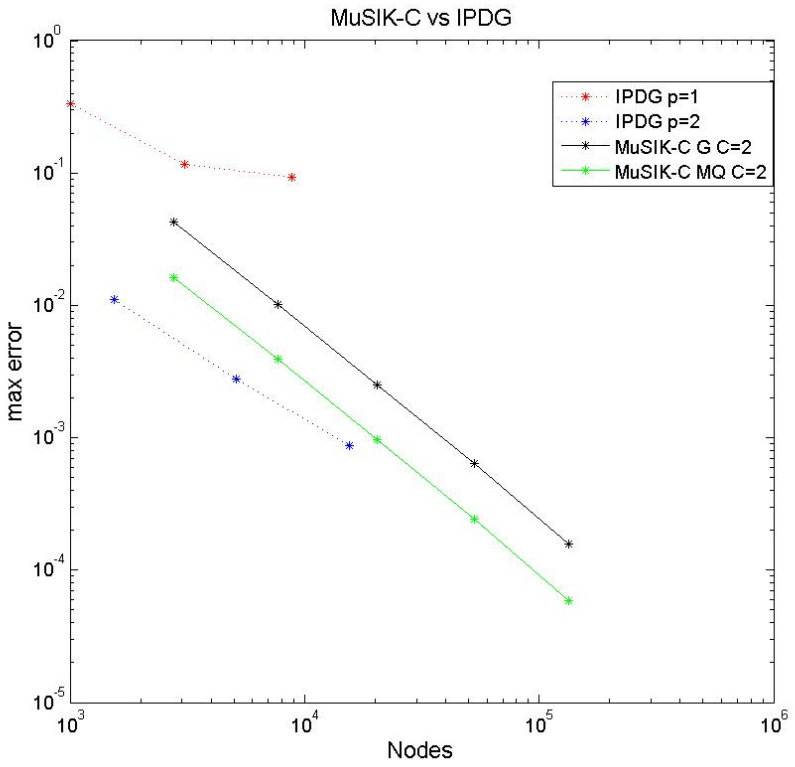

FIGURE 4.9: The performance of the multilevel sparse collocation methods using MQ and Gaussian with connection constant $C = 2$ for Example 4.4.

Example 4.5. *In this example, we solve the following four-dimensional problem on* $\Omega = [0,1]^4$

$$\Delta u(\mathbf{x}) = -\pi^2 \sin(\pi \prod_{i=1}^{4} x_i)(x_2^2 x_3^2 x_4^2 + x_1^2 x_3^2 x_4^2 + x_1^2 x_2^2 x_4^2 + x_1^2 x_2^2 x_3^2), \quad \mathbf{x} \in \Omega, \tag{4.27}$$

with boundary conditions

$$u(\mathbf{x}) = \sin(\pi \prod_{i=1}^{4} x_i), \quad \mathbf{x} \in \partial\Omega. \tag{4.28}$$

The exact solution is a four-dimensional non-tensor product function

$$u(\mathbf{x}) = \sin(\pi \prod_{i=1}^{4} x_i). \tag{4.29}$$

As we did not find any publications that solve PDE with a non-tensor product target function in high dimension, we just display our numerical results. Table 4.16 shows the performance when the target is a non-tensor product function in

Level	Nodes	$E_{\text{MuSIK-C}}$ (MQ)	$\rho_{\text{MuSIK-C}}$ (MQ)	$E_{\text{MuSIK-C}}$ (G)	$\rho_{\text{MuSIK-C}}$ (G)
4	2769	9.78e-3	—	2.84e-2	—
5	7681	5.28e-3	-0.60	1.16e-2	-0.88
6	20481	1.85e-3	-1.07	2.20e-3	-1.70
7	52993	5.08e-4	-1.36	7.08e-4	-1.19
8	133889	1.37e-4	-1.41	2.03e-4	-1.35

TABLE 4.16: The performance of multilevel sparse collocation methods using MQ and Gaussian for Example 4.5 with $C = 2$. Max error evaluated at 240,000 Halton points in the whole domain.

four dimensions. Here the focus is on multilevel sparse collocation with MQ and Gaussian with the same connection constants $C = 2$. Similarly, we observe the convergence rate is also growing slowly.

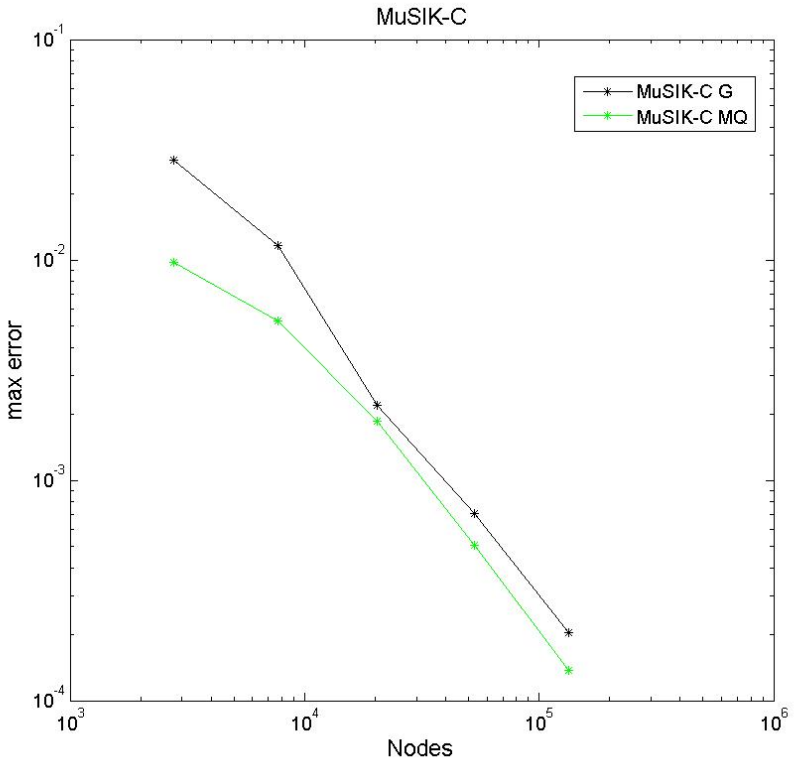

FIGURE 4.10: The performance of the multilevel sparse collocaiton methods using MQ and Gaussian with connection constant $C = 2$ for Example 4.5.

4.3.2 Parabolic examples

Example 4.6. *In this example, we solve the following one-dimensional spatial problem on* $\Omega_t = \Omega \times t = [0,1] \times [0,1]$

$$u_t - u_{xx} = \frac{\pi^2}{(t+0.5)^2} \sin\left(\frac{\pi x}{t+0.5}\right) - \frac{\pi x}{(t+0.5)^2} \cos\left(\frac{\pi x}{t+0.5}\right), \quad x \in \Omega, \quad t \in (0,1], \tag{4.30}$$

with boundary and initial conditions

$$u(t,x) = \sin\left(\frac{\pi x}{t+0.5}\right), \quad x \in \partial\Omega, \quad t \in (0,1], \tag{4.31}$$

$$u(0,x) = \sin(2\pi x), \quad x \in \Omega. \tag{4.32}$$

The analytical solution is a non-tensor product function

$$u(t,x) = \sin\left(\frac{\pi x}{t+0.5}\right). \tag{4.33}$$

Level	Nodes	MQ C=2	MQ C=3	Gaussian C=2	Gaussian C=3
2	21	4e4	1e6	3e3	6e5
3	49	4e5	2e7	4e4	2e8
4	113	5e6	7e7	1e6	8e10
5	257	3e7	2e9	2e7	6e12
6	577	2e8	2e10	1e8	7e13
7	1281	7e8	8e10	6e8	1e15
8	2817	3e9	4e11	3e9	1e16
9	6145	1e10	1e12	1e10	2e17
10	13313	5e10	6e12	4e10	6e18
11	28673	2e11	2e13	2e11	4e19
12	61441	7e11	1e14	7e11	3e20

TABLE 4.17: Sparse grid condition number using MQ and Gaussian with the two connection constants: 2 and 3, in the two dimensional heat problem.

In Tables 4.18 and 4.19, the focus is on multilevel sparse collocation with MQ and Gaussian with two constants: $C = 2$ and 3. The convergence rate here appears to be growing slowly. In Table 4.19, MuSIK-C with the Gaussian breaks down in the levels 11 and 12.

Level	Nodes	$E_{\text{MuSIK}-\text{C}}$ (MQ)	$\rho_{\text{MuSIK}-\text{C}}$ (MQ)	$E_{\text{MuSIK}-\text{C}}$ (G)	$\rho_{\text{MuSIK}-\text{C}}$ (G)
2	21	4.21e-1	—	4.22e-1	—
3	49	2.26e-1	-0.73	2.09e-1	-0.83
4	113	1.14e-1	-0.82	9.89e-2	-0.90
5	257	3.53e-2	-1.43	2.80e-2	-1.54
6	577	8.88e-3	-1.71	6.58e-3	-1.79
7	1281	2.20e-3	-1.75	1.73e-3	-1.67
8	2817	6.22e-4	-1.60	5.34e-4	-1.50
9	6145	1.88e-4	-1.54	1.75e-4	-1.43
10	13313	5.62e-5	-1.56	5.61e-5	-1.47
11	28673	1.62e-5	-1.62	1.68e-5	-1.57
12	61441	4.30e-6	-1.74	4.82e-6	-1.64

TABLE 4.18: The performance of the multilevel sparse collocation method using MQ and Gaussian for Example 4.6 with $C = 2$. Max error evaluated at 64,000 Halton points in the whole domain.

Level	Nodes	$E_{\text{MuSIK}-\text{C}}$ (MQ)	$\rho_{\text{MuSIK}-\text{C}}$ (MQ)	$E_{\text{MuSIK}-\text{C}}$ (G)	$\rho_{\text{MuSIK}-\text{C}}$ (G)
2	21	4.85e-1	—	5.06e-1	—
3	49	2.44e-1	-0.81	2.30e-1	-0.93
4	113	8.87e-2	-1.21	4.72e-2	-1.89
5	257	1.76e-2	-1.97	1.17e-2	-1.70
6	577	2.82e-3	-2.27	1.31e-3	-2.71
7	1281	6.67e-4	-1.81	1.39e-4	-2.81
8	2817	1.99e-4	-1.54	3.83e-5	-1.64
9	6145	5.46e-5	-1.66	6.26e-6	-2.32
10	13313	1.34e-5	-1.81	8.96e-7	-2.52
11	28673	2.99e-6	-1.96	5.23e-7	-0.70
12	61441	7.88e-7	-1.75	2.47e-6	2.04

TABLE 4.19: The performance of the multilevel sparse collocation method using MQ and Gaussian for Example 4.6 with $C = 3$. Max error evaluated at 64,000 Halton points in the whole domain.

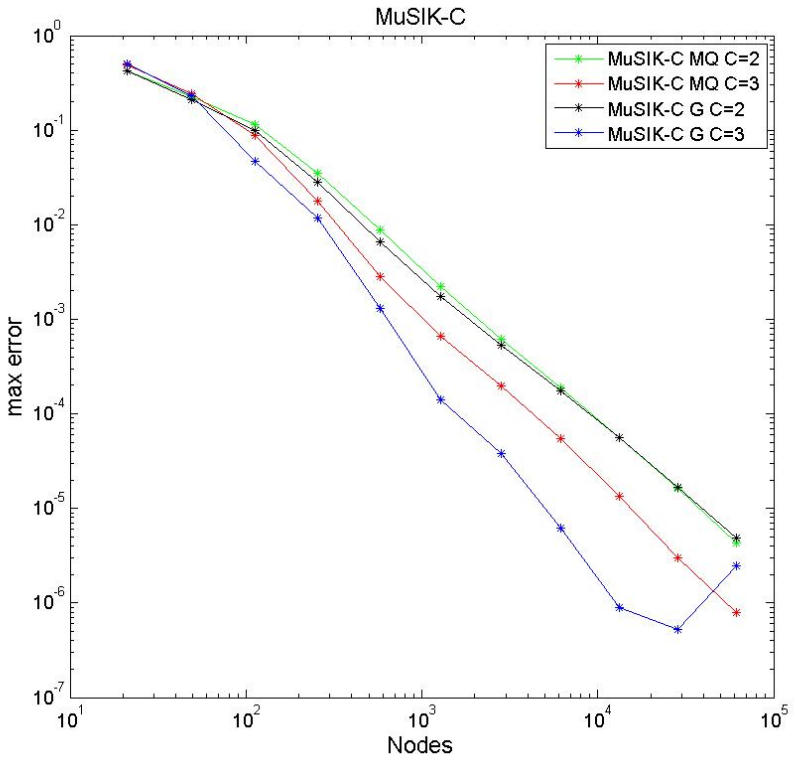

FIGURE 4.11: The performance of the multilevel sparse collocation method using MQ and Gaussian with the two contants: 2 and 3, for Example 4.6.

Example 4.7. *In this example, we solve the following two-dimensional spatial problem on* $\Omega_t = \Omega \times t = [0,1]^2 \times [0,1]$

$$u_t - \Delta u = \pi \sin(\pi x_1) \sin(\pi x_2) \left(\cos(\pi t) + 2\pi \sin(\pi t)\right), \quad \mathbf{x} \in \Omega, \quad t \in (0,1], \tag{4.34}$$

with boundary and initial conditions

$$u(t,\mathbf{x}) = \sin(\pi t) \sin(\pi x_1) \sin(\pi x_2), \quad \mathbf{x} \in \partial\Omega, \quad t \in (0,1], \tag{4.35}$$

$$u(0,\mathbf{x}) = 0, \quad \mathbf{x} \in \Omega. \tag{4.36}$$

The analytical solution is a tensor product function

$$u(t,\mathbf{x}) = \sin(\pi t) \sin(\pi x_1) \sin(\pi x_2). \tag{4.37}$$

Langer et al. [74] presented a new stable space-time Isogeometric Analysis (IgA) method in 2016. Isogeometric analysis is a collection of methods (henceforth

referred to as isogeometric methods) that use splines, or some of their extensions such as NURBS (non-uniform rational B-splines) and T-splines, as functions to build approximation spaces which are then used to solve partial differential equations numerically, see [21] for detail. As the authors just presented L_2 errors in [74], we change to RMS error here to compare. Let us define the error and slope as

$$\begin{aligned} E^n_{\mathrm{RMS}} &= \sqrt{\frac{1}{N_{\mathbf{T}}}\sum_{i=1}^{N_{\mathbf{T}}}\left(u(\mathbf{x}_i)-\hat{u}^{n,d}_{\mathrm{ML}}(\mathbf{x}_i)\right)^2}, \quad \mathbf{x}_i \in \mathbf{T}, \\ \rho_{\mathrm{RMS}} &= \frac{\log\left(E^{n+1}_{\mathrm{RMS}}\right)-\log\left(E^n_{\mathrm{RMS}}\right)}{\log\left(\mathrm{Nodes}^{n+1}\right)-\log\left(\mathrm{Nodes}^n\right)}, \end{aligned}$$

where Nodes^n means the nodes number at the level n, $N_{\mathbf{T}}$ is the number of nodes in the testing points set $\mathbf{T}$.

Level	Nodes	MQ C=2	MQ C=3	Gaussian C=2	Gaussian C=3
3	225	2e8	3e10	1e7	4e10
4	593	2e9	6e11	2e8	1e13
5	1505	2e10	9e12	1e9	1e15
6	3713	9e10	8e13	9e9	3e16
7	8961	5e11	6e14	6e10	9e18
8	21249	2e12	4e15	3e11	9e19

TABLE 4.20: Sparse grid condition number using MQ and Gaussian with the two connection constants: 2 and 3, in the three dimensional heat problem.

Level	Nodes	E_{RMS} (MQ)	ρ_{RMS} (MQ)	E_{RMS} (G)	ρ_{RMS} (G)
3	225	4.78e-3	—	8.54e-3	—
4	593	9.05e-4	-1.72	1.86e-3	-1.57
5	1505	2.09e-4	-1.57	4.45e-4	-1.53
6	3713	4.91e-5	-1.60	1.08e-4	-1.57
7	8961	1.21e-5	-1.59	2.68e-5	-1.58
8	21249	2.96e-6	-1.63	6.66e-6	-1.61

TABLE 4.21: The performance of the multilevel sparse collocation method using MQ and Gaussian for Example 4.7 with $C = 2$. Max error evaluated at 120,000 Halton points in the whole domain.

In Tables 4.21 and 4.22, the focus is on multilevel sparse collocation with MQ and Gaussian with two constants $C = 2$ and 3. The convergence rate here appears to increase slowly before the multilevel sparse collocation breaks down. In Figure

Level	Nodes	E_{RMS} (MQ)	ρ_{RMS} (MQ)	E_{RMS} (G)	ρ_{RMS} (G)
3	225	2.34e-3	—	1.74e-3	—
4	593	1.96e-4	-2.56	1.19e-4	-2.77
5	1505	3.59e-5	-1.82	1.55e-5	-2.18
6	3713	6.67e-6	-1.86	1.97e-6	-2.29
7	8961	1.28e-6	-1.87	2.55e-7	-2.32
8	21249	3.25e-7	-1.59	5.69e-8	-1.74

TABLE 4.22: The performance of the multilevel sparse collocation method using MQ and Gaussian for Example 4.7 with $C = 3$. Max error evaluated at 120,000 Halton points in the whole domain.

4.12, the numerical results of the IgA method are taken from [74]. MuSIK-C with $C = 2$ has better performance than the IgA with $p = 1$ and has faster convergence rate than the IgA with $p = 2$. MuSIk-C using the Gaussian with $C = 3$ performs better than the IgA with $p = 3$.

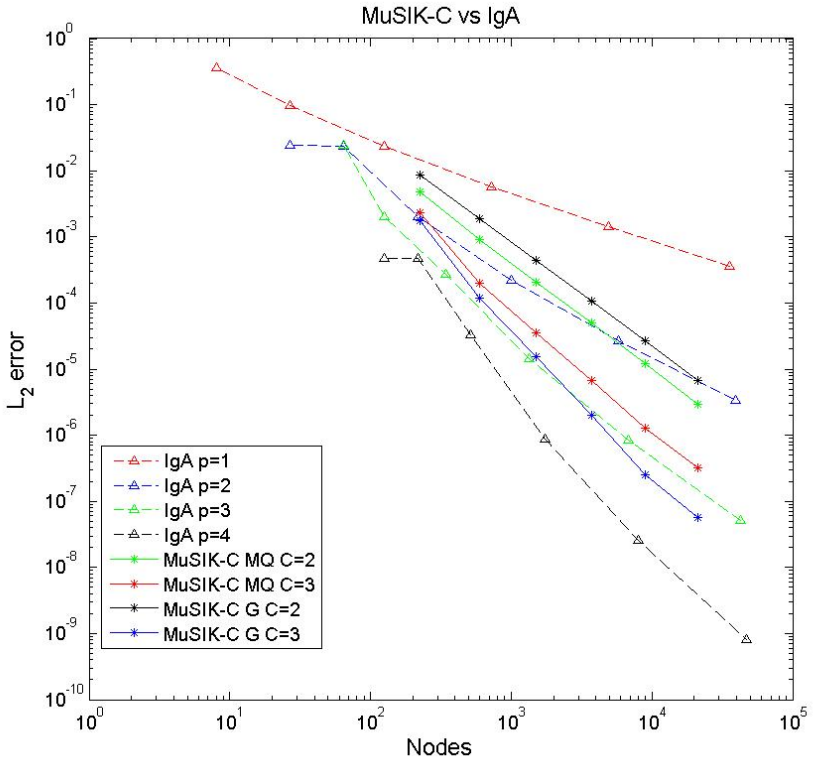

FIGURE 4.12: This compares mulitilevel sparse collocation with MQ/Gaussian and IgA for Example 4.7.

Example 4.8. *In this example, we solve the following three-dimensional spatial problem on* $\Omega_t = \Omega \times t = [0,1]^3 \times [0,1]$

$$u_t - \Delta u = \pi \sin(\pi x_1)\sin(\pi x_2)\sin(\pi x_3)\left(\cos(\pi t) + 3\pi \sin(\pi t)\right), \quad \mathbf{x} \in \Omega, \quad t \in (0,1], \tag{4.38}$$

with boundary and initial conditions

$$u(t,\mathbf{x}) = \sin(\pi t)\sin(\pi x_1)\sin(\pi x_2)\sin(\pi x_3), \quad \mathbf{x} \in \partial\Omega, \quad t \in (0,1], \tag{4.39}$$

$$u(0,\mathbf{x}) = 0, \quad \mathbf{x} \in \Omega. \tag{4.40}$$

The analytical solution is a tensor product function

$$u(t,\mathbf{x}) = \sin(\pi t)\sin(\pi x_1)\sin(\pi x_2)\sin(\pi x_3). \tag{4.41}$$

Level	Nodes	MQ C=2	Gaussian C=2
4	2769	3e11	5e9
5	7681	3e12	4e10
6	20481	2e13	3e11
7	52993	2e14	2e12
8	133889	1e15	3e13

TABLE 4.23: Sparse grid condition number using MQ and Gaussian with the connection constant $C = 2$ in the four dimensional heat problem.

Level	Nodes	E_{RMS} (MQ)	ρ_{RMS} (MQ)	E_{RMS} (G)	ρ_{RMS} (G)
4	2769	1.80e-3	—	3.19e-3	—
5	7681	3.71e-4	-1.55	6.98e-4	-1.49
6	20481	8.91e-5	-1.45	1.70e-4	-1.44
7	52993	2.18e-5	-1.48	4.23e-5	-1.46
8	133889	5.48e-6	-1.49	1.06e-5	-1.50

TABLE 4.24: The performance of the multilevel sparse collocation method using MQ and Gaussian for Example 4.8 with $C = 2$. Max error evaluated at 240,000 Halton points in the whole domain.

In Table 4.23, we see that the condition numbers grow fast for both basis functions. In Table 4.24, the focus is on multilevel sparse collocation using MQ and Gaussian with constant $C = 2$. The convergence rate here also appears to grow slowly. In

Figure 4.9, the numerical results of IgA are taken from [74]. So we also present RMS error here. MuSIK-C with $C = 2$ appears to have more capacity than the IgA with $p = 2$ in the square domain. As MuSIK-C is not applicable on a flexible domain, we cannot compare with IgA on irregular regions.

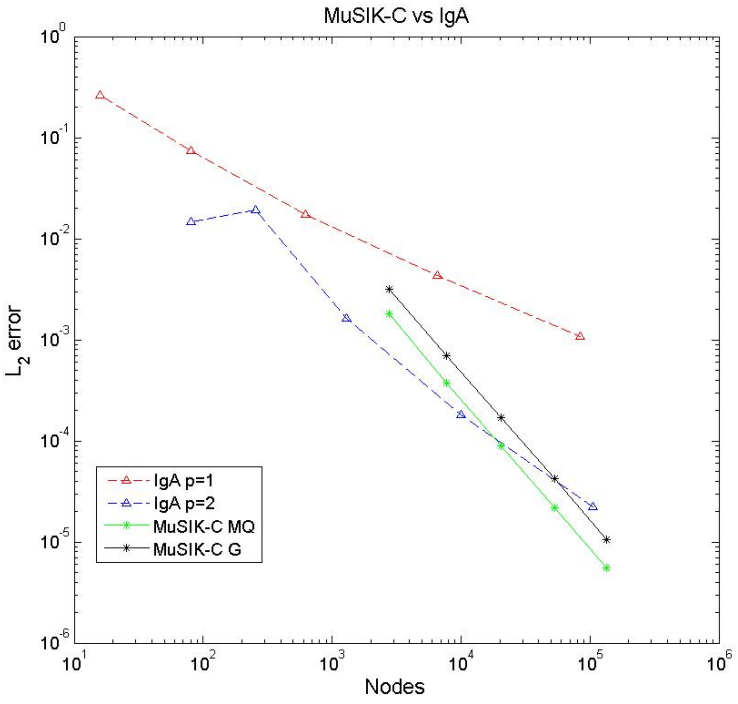

FIGURE 4.13: This compares multilevel sparse collocation with MQ/Gaussian and IgA for Example 4.8.

Example 4.9. *In this example, we solve the following three-dimensional spatial problem on* $\Omega_t = \Omega \times t = [0,1]^3 \times [0,1]$

$$u_t - \Delta u = e^{10(t-1)} \sin(\pi x_1 x_2 x_3) \left(10 + \pi^2 \left(x_2^2 x_3^2 + x_1^2 x_3^2 + x_1^2 x_2^2\right)\right), \quad \mathbf{x} \in \Omega, \quad t \in (0,1], \tag{4.42}$$

with boundary and initial conditions

$$u(t,\mathbf{x}) = e^{10(t-1)} \sin(\pi x_1 x_2 x_3), \quad \mathbf{x} \in \partial\Omega, \quad t \in (0,1], \tag{4.43}$$

$$u(0,\mathbf{x}) = e^{-10} \sin(\pi x_1 x_2 x_3), \quad \mathbf{x} \in \Omega. \tag{4.44}$$

The analytical solution is a non-tensor product function

$$u(t,\mathbf{x}) = e^{10(t-1)} \sin(\pi x_1 x_2 x_3). \tag{4.45}$$

Level	Nodes	$E_{\text{MuSIK-C}}$ (MQ)	$\rho_{\text{MuSIK-C}}$ (MQ)	$E_{\text{MuSIK-C}}$ (G)	$\rho_{\text{MuSIK-C}}$ (G)
4	2769	6.44e-2	—	9.18e-2	—
5	7681	2.70e-2	-0.85	4.33e-2	-0.74
6	20481	9.54e-3	-1.06	1.30e-2	-1.23
7	52993	3.30e-3	-1.12	4.01e-3	-1.24
8	133889	1.06e-3	-1.22	1.18e-3	-1.32

TABLE 4.25: The performance of the multilevel sparse collocation method using MQ and Gaussian for Example 4.9 with $C = 2$. Max error evaluated at 240,000 Halton points in the whole domain.

Table 4.25 shows the performance when the target is a non-tensor product function in three spatial dimensions. Here the focus is on multilevel sparse collocation with MQ and Gaussian with $C = 2$. We observe that the convergence rate here also appears to increase slowly.

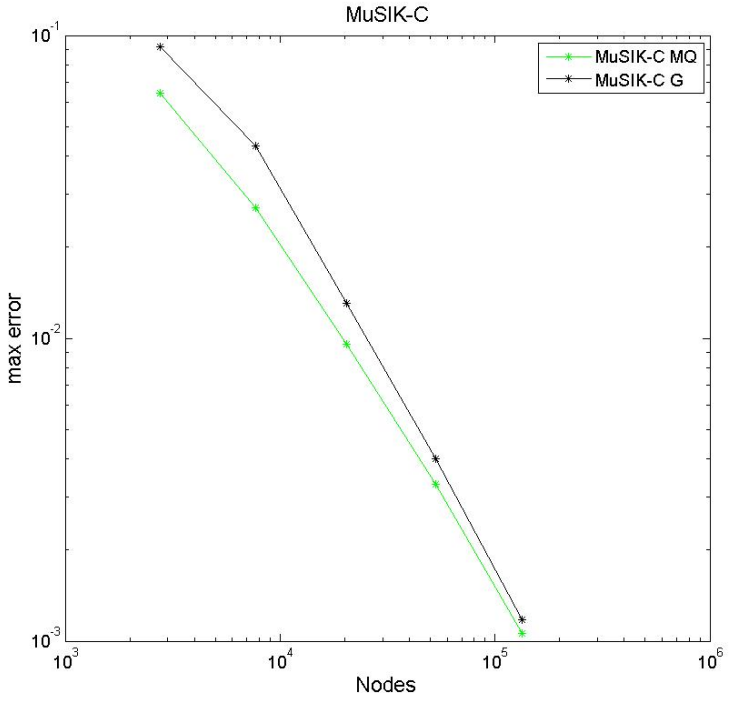

FIGURE 4.14: The performance of the multilevel sparse collocation method using MQ and Gaussian with $C = 2$ for Example 4.9.

4.4 Conclusion

From the experiments in last section, we can demonstrate that MuSIK-C is faster and more accurate than RBF-C, MLRBF-C and SIK-C when solving a PDE with smooth boundary conditions. MuSIK-C has similar performance as the recent mesh-based methods to solve PDEs with smooth conditions in low dimensions. In higher dimensions, we obtain better approximations. Moreover, it is a difficult task to construct elements for the mesh-based method in high dimensions, especially for high order polynomials. MuSIK-C, as a mesh-free method, still has its advantage of easy implementation. The convergence rates of MuSIK-C in the above examples are increasing slowly and might even be spectral. Furthermore, the solution displays more rapid convergence when utilising bigger shape parameters. However, the condition number also grows quickly so that the problem is ill-conditioned and solutions become unreliable. As we described in Chapter 2, there have been some developments in reducing the condition number while keeping the high convergence when using RBF. We will explore the use of techniques to improve our method in the near future. In Chapter 5, we shall test the performance of MuSIK-C in the space-time method to approximate an option price.

Chapter 5

MuSIK-C for option pricing

In this chapter, we discuss the implementation of MuSIK-C on option pricing. Recall the multi-assets *Black-Scholes equation* on domain $[0, T] \times \Omega$:

$$\frac{\partial C}{\partial t} + \frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{d}\rho_{ij}\sigma_i\sigma_j S_i S_j \frac{\partial^2 C}{\partial S_i S_j} + \sum_{i=1}^{d}(r - q_i)S_i\frac{\partial C}{\partial S_i} - rC = 0, \tag{5.1}$$

where C stands for the analytical option price which is determined by time t and d assets' prices, σ_i is the volatility of ith underlying stock, $\rho_{i,j}$ is the correlation between stocks i and j, q_i is the continuous dividend payment for ith asset, r is the risk-free rate. In 2015, the authors in [110] gathered together to join in the BENCHOP project, which provides the finance community with a common suite of benchmark problems for option pricing. At that time, we participated in the BENCHOP with the multilevel full grid radial basis function collocation (MLRBF-C). In this chapter, we hope to have a better approximation by utilising MuSIK-C method.

In Chapter 4, the numerical experiments have shown that MuSIK-C is stable and efficient in solving parabolic problem with smooth boundary conditions. However, we found MuSIK-C is not suitable to solve problem with non-smooth conditions like option pricing as shown in Section 5.1.1. In order to handle this problem, we modify the time interval by a small amount and take a relatively smooth initial condition at an earlier time τ. Then we apply MuSIK-C on the new domain

$[0, \tau] \times [S_{min}, S_{max}]$. The detail about how to determine the value of τ and estimate the fine initial condition is displayed in Section 5.1.2. In Section 5.1.3, we use the combination method to price one dimensional European call option. Afterwards, we also price the Margrabe option with the combination method in Section 5.2. In Section 5.3, we employed Richardson Extrapolation to improve the approximation based on existing results.

5.1 One asset European Option pricing

5.1.1 MuSIK-C in option pricing from different initial times

Recall the details about the one asset *Black-Scholes* PDE problem shown in Section 3.3. In this subsection, we display numerical results following the Parameter Set 1 in Table 3.1. To show the performance of MuSIK-C and SIK-C at different initial times, we set one initial time is the maturity time T and another is $\tau = 0.5T$. Initial conditions are $\max\{S - E, 0\}$ and analytical solution $C(\tau, S)$, respectively. The boundary conditions are the same as Equations (3.34) and (3.35) in both cases.

Even though the interval for stock price S is $[S_{min}, S_{max}]$, the most interesting area is around strike price E. Let us set the central region as $\left[\widehat{S}_{min}, \widehat{S}_{max}\right] = [0.4E, 1.6E]$. The errors in the following figures are sampled on the central region $\left[\widehat{S}_{min}, \widehat{S}_{max}\right]$. This is also required in the BENCHOP [110].

In Figure 5.1 shows max error of estimations at 3000 testing points evenly distributed in $[0.4E, 1.6E]$ at current time $t = 0$. It is clear that only SIK-C with MQ basis functions keeps converging and there is no significant difference in both cases: the initial time is at T and the initial time is at $\tau = 0.5T$. Meanwhile MuSIK-C only works when the initial time is $\tau = 0.5T$. The Figure 5.1 is an indication to show that our scheme is practical. We would like to make the approximation at time τ is close enough to the analytical solution. Otherwise, we will introduce another factor that will influence our numerical result. This restriction can be satisfied by

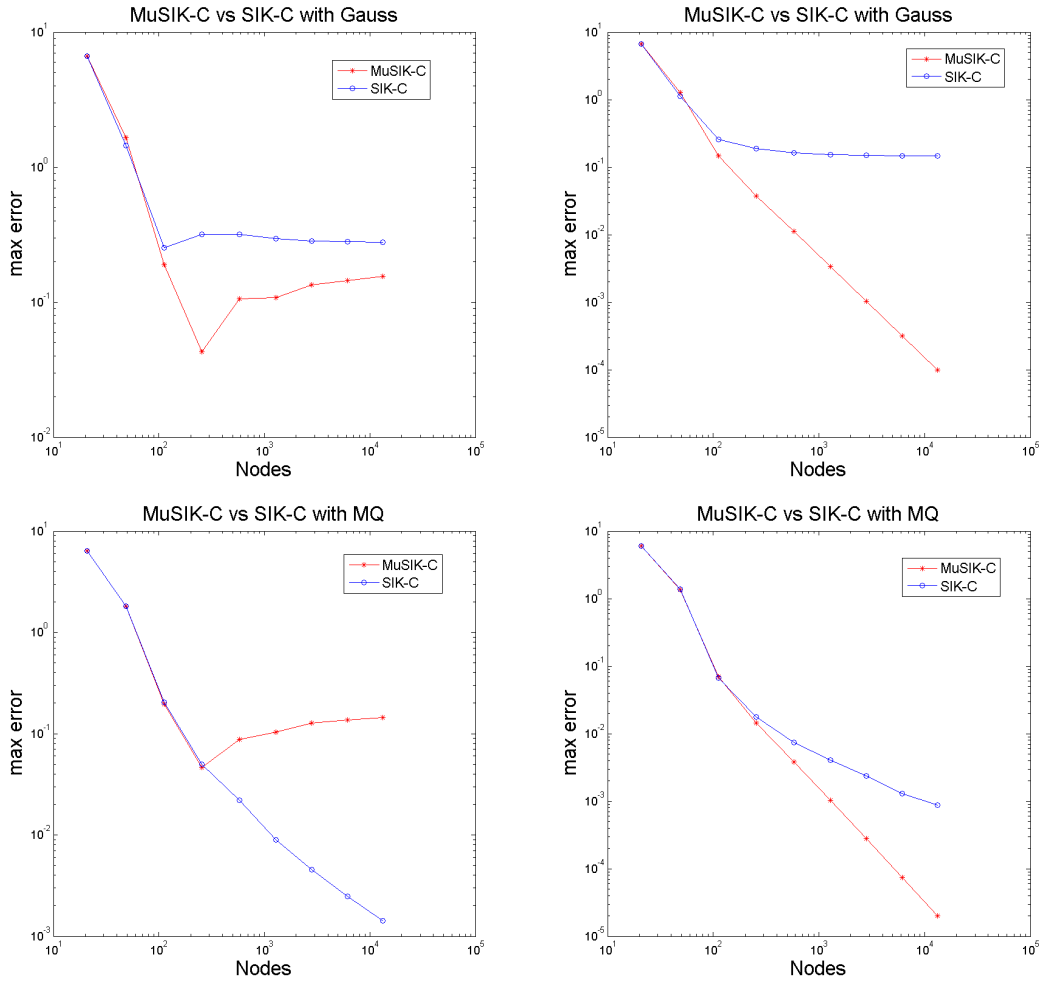

FIGURE 5.1: Max error of estimations at 3000 uniform points in $[0.4E, 1.6E]$ at $t = 0$ for Parameter Set 1. All initial conditions are analytical solutions and boundary condtions are the same. Initial times in left plots are maturity time T, that of right side are time $\tau = 0.5T$. Two figures above are using the Gaussian basic functions, the figures below are using MQ basic functions

employing plenty of nodes to construct a finer grid as discussed in Section 3.3. In next section, we introduce algorithm about the determination of time τ.

5.1.2 Algorithm to predetermine an earlier time

Our purpose is to take an approximation $\hat{u}^{\tau}$ at an earlier time τ before the maturity time T as an initial condition, in order to apply MuSIK-C on the remaining time interval. However, it is illogical if we take τ too earlier because MuSIK-C is our main method. It will be meaningless if the domain for MuSIK-C is quite small. So firstly, we define a symbol called τ^{end} to stand for minimum value of τ that we admit, see Figure 5.2. As we are moving initial condition backward step by

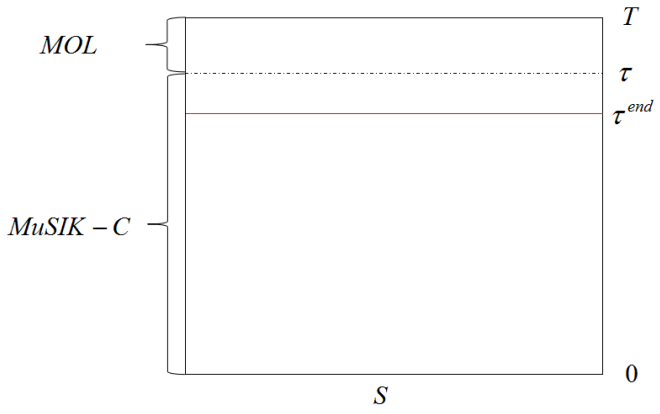

FIGURE 5.2: Dashed line is stop time τ, red solid line is terminate spot τ^{end}.

step, the method of lines (MOL) is proper to decide the time spot we should stop. Here, we use method introduced in Section 3.2.1. For the purpose of computing a good estimation on the whole domain $[S_{min}, S_{max}]$, we have to choose a bigger domain $\left[\widetilde{S}_{min}, \widetilde{S}_{max}\right] \supset [S_{min}, S_{max}]$ so as to prevent the truncation error from the boundary.

Owing to the facts that payoff functions of one asset European options only have singularity at strike price, the option value is monotone and convex around the strike price. So the strategy for selecting the modified initial condition in time is to ensure that the approximation is convex in a region will be described later. The test for this is the Gamma of the approximation is positive. There are no points of inflection (changes in sign of curvature). This is ensured by testing the slope of the second derivative of approximation, also known as the speed, is monotonic on a sub-interval around the strike price. Firstly we define the domain around exercise price E is $\left[\widehat{E}_{-}, \widehat{E}_{+}\right]$, here $\widehat{E}_{\pm} := E \pm 0.2E$. Supposing we put a large

amount of uniform nodes $\mathcal{X}$ in domain $\left[\widetilde{S}_{min}, \widetilde{S}_{max}\right]$, the nodes located in $\left[\widehat{E}_{-}, \widehat{E}_{+}\right]$ are defined as set $\mathcal{X}_1$. The approximation $\hat{u}^{\tau}$ satisfies conditions such as Gamma $\frac{\partial^2 \hat{u}^{\tau}}{\partial S^2} > 0$ on $\mathcal{X}_1$ and further the Speed $\frac{\partial^3 \hat{u}^{\tau}}{\partial S^3}$ is monotonic in subintervals of $\mathcal{X}_1$. The performance of the exact Gamma and Speed for Parameter Set 1 is drawn in Figure 5.3. As we can see, there are three subintervals in the figure of the Speed and the Speed is monotonic in every subinterval.

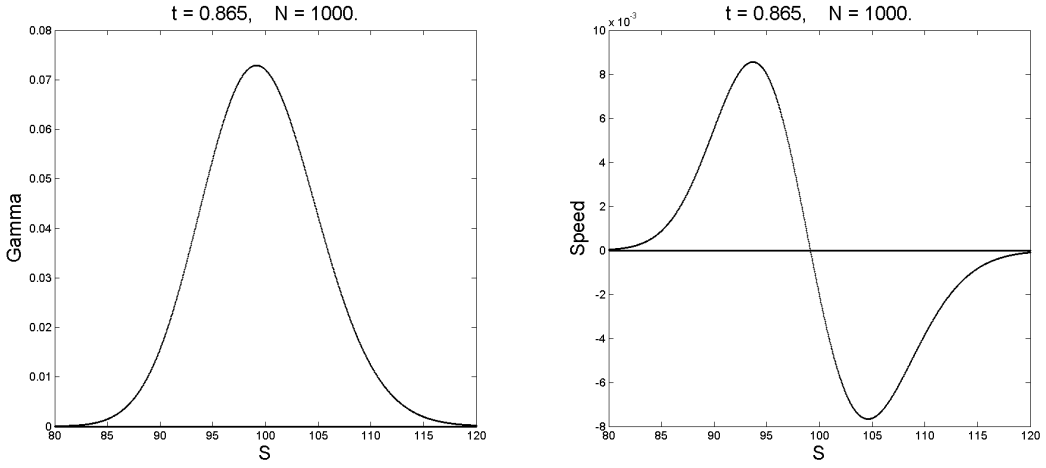

FIGURE 5.3: Examples of exact Gamma(left) and Speed(right) at $t = 0.865$ with 1000 uniform nodes in $\mathcal{X}_1 = [80, 120]$ for Parameter Set 1.

Currently, the scheme to determine τ is not perfect and is a method of choice. It might give us a relatively smooth initial condition rather than exactly smooth. This will be demonstrated again in the following sections. As so far, we can summarise the combination algorithm for one asset European option pricing as Algorithm 3.

Algorithm 3 Algorithm for the combination method on one asset European option pricing

1. Input parameter set, nodes number N, time step Δt and terminate time τ^{end}.

2. Determine domain for MOL as $\left[\widetilde{S}_{min}, \widetilde{S}_{max}\right]$.

3. Construct set $\mathcal{X}$ in $\left[\widetilde{S}_{min}, \widetilde{S}_{max}\right]$ with N uniform nodes and set $\mathcal{X}_1 = \mathcal{X} \cap \left[\widehat{E}_-, \widehat{E}_+\right]$, $\widehat{E}_\pm = E \pm 0.2E$.

4. Set $Tdone = 0$ and apply MOL.

while $Tdone < T - \tau^{end}$ **do**

Let x_{max}, x_{min} to be the prices where approximations $\hat{u}(x_{max})$ and $\hat{u}(x_{min})$ are extremum values on $\mathcal{X}_1$. Define $\mathcal{X}_{11} = \left[\widehat{E}_-, x_{max}\right]$, $\mathcal{X}_{12} = [x_{max}, x_{min}]$, $\mathcal{X}_{13} = \left[x_{min}, \widehat{E}_+\right]$ to be subintervals of $\mathcal{X}_1$.

if $Gamma = \frac{\partial^2 \hat{u}}{\partial S^2} > 0$ on $\mathcal{X}_1$ and $Speed = \frac{\partial^3 \hat{u}}{\partial S^3}$ is monotonic on three subintervals **then**

Break MOL;

else

$Tdone = Tdone + \Delta t$;

end if

end while

5. Assign $\tau = Tdone$ and set initial condition $f = \hat{u}$.

6. Reconstruct *Black-Scholes equation* on $\Omega_t^* = [0, \tau] \times [S_{min}, S_{max}]$ as:

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + (r - q) S \frac{\partial V}{\partial S} - rV = 0,$$

with initial condition and boundary conditions

$$\begin{aligned} V(t, S_{min}) &= 0, \quad t \in [0, \tau], \\ V(t, S_{max}) &= S_{max} e^{-q*(T-t)} - E e^{-r*(T-t)}, \quad t \in [0, \tau], \\ V(\tau, S) &= f(S), \quad S \in [S_{min}, S_{max}], \end{aligned}$$

7. Employ MuSIK-C Algorithm 2.

5.1.3 Numerical experiments

Our first example is to price the European call option following Parameter Set 1 in Table 3.1. In the algorithm to predetermine time τ, we utilise the necessary restrictions setting in the MOL as shown in Table 5.1.

Variables Values	
r^{end}	$0.8T$
N	5000
Δt	0.001
$\widetilde{S}_{min}$	$-E$
$\widetilde{S}_{max}$	$2S_{max}$

TABLE 5.1: Restricted variables set.

The size of set $\mathcal{X}$ is 5000 and the size of $\mathcal{X}_1 = \mathcal{X} \cap [E - 0.2E, E + 0.2E]$ is 286. Algorithm 3 stops using MOL when $\tau = 0.865$. In Figure 5.4, values of 1000 uniform points option values in $[S_{min}, S_{max}]$ are given, Gamma and Speed on $\mathcal{X}_1$ are graphed in left-hand side, and corresponding errors of approximations are given in right-hand side. To indicate $\tau = 0.865$ is necessary in this particular situation, we represent the performance of approximation at $\tau = 0.9$ in Figure 5.5. From the upper right corner in Figure 5.5, we can see there is no big difference in option values. However, we can observe apparent oscillation around $S = 100$ in Speed from bottom left corner figure in Figure 5.5. Meanwhile, the error plots of Gamma and Speed in Figure 5.5 are almost 10 times bigger than the corresponding graphs in Figure 5.4.

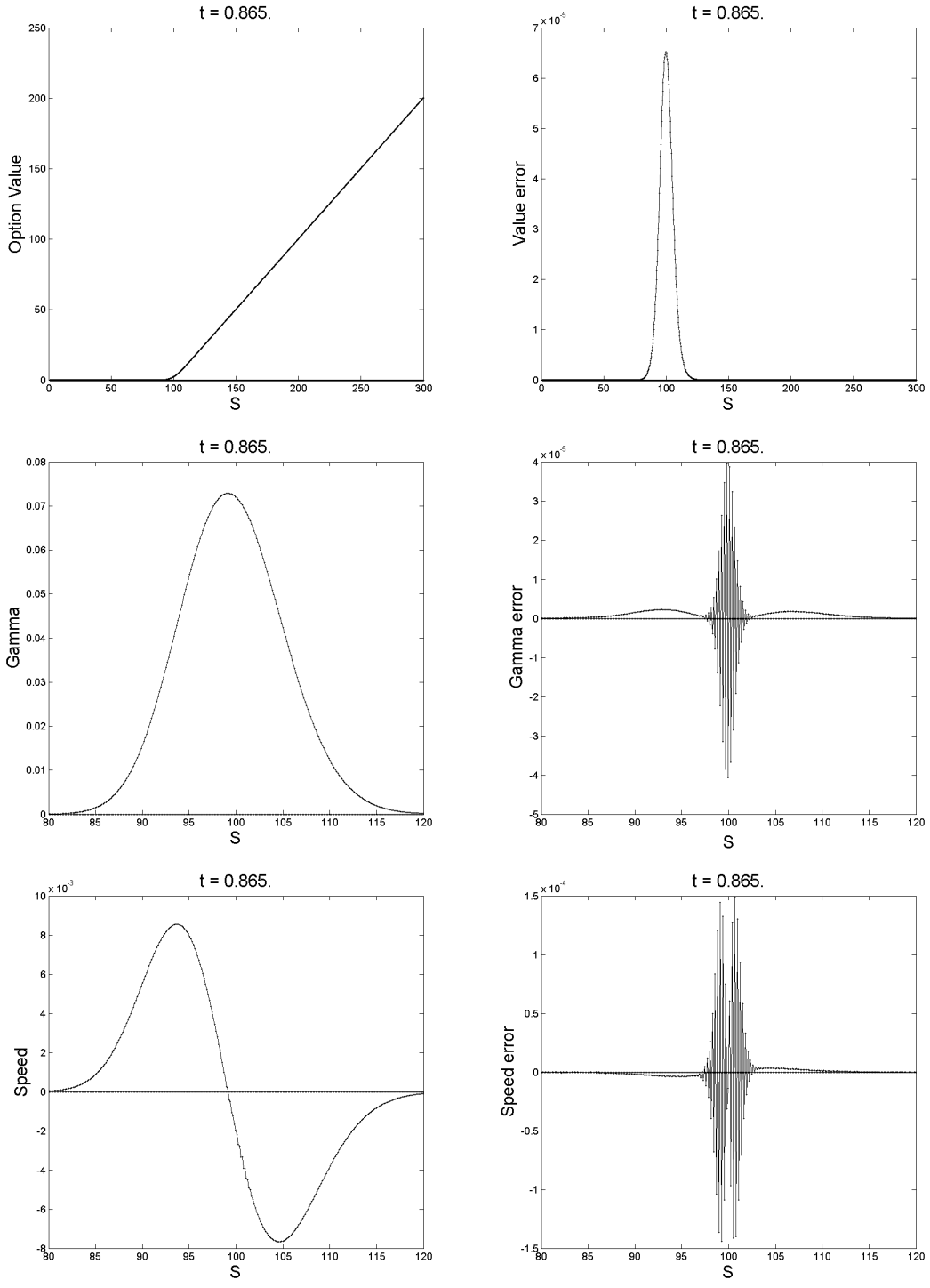

FIGURE 5.4: Approximations on the European call option price, Gamma and Speed (left three figures), and corresponding errors (right three figures) at $\tau = 0.865$ following variable set in Table 5.1 and Parameter Set 1 in Table 3.1.

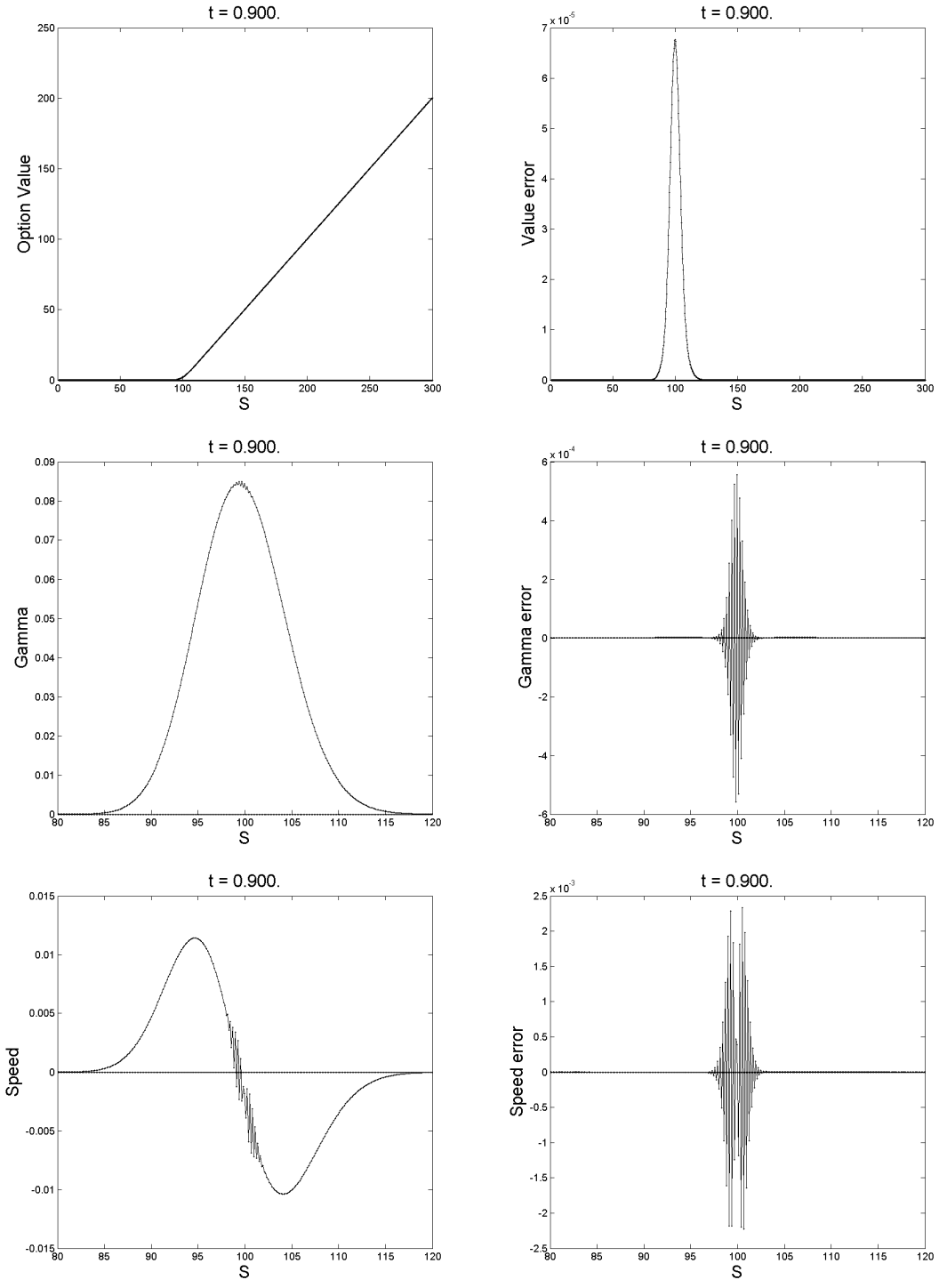

FIGURE 5.5: Approximations on the European call option price, Gamma and Speed (left three figures), and corresponding errors (right three figures) at $\tau = 0.9$ following variable set in Table 5.1 and Parameter Set 1 in Table 3.1.

Continuing Step 6 in Algorithm 3, we apply Algorithm 2 on domain $\Omega_t^* = [0, \tau] \times [S_{min}, S_{max}]$. To distinguish the notations, for the solutions obtained on the computational domain $[0, \tau] \times [S_{min}, S_{max}]$, we measure the errors at the level n as

$$\begin{aligned} E^{+,n}_{\text{SIK}-\text{C}} &= \max_{S \in \mathbf{T}} |C(0,S) - \hat{u}^{n,d}(0,S)|, \\ E^{+,n}_{\text{MuSIK}-\text{C}} &= \max_{S \in \mathbf{T}} |C(0,S) - \hat{u}^{n,d}_{\text{ML}}(0,S)|, \end{aligned}$$

here $\mathbf{T}$ is a testing domain which will be explicit in each case. Correspondingly, we also define ρ is the slope of two adjacent points, for instance,

$$\rho^{+}_{\text{SIK}-\text{C}} = \frac{\log\left(E^{+,n+1}_{\text{RBF}-\text{C}}\right) - \log\left(E^{+,n}_{\text{RBF}-\text{C}}\right)}{\log\left(\text{Nodes}^{n+1}\right) - \log\left(\text{Nodes}^{n}\right)},$$

here Nodes^n means the nodes number in the level n.

Level	Nodes	$E^{+}_{\text{MuSIK}-\text{C}}$	$\rho^{+}_{\text{MuSIK}-\text{C}}$	$E^{+}_{\text{SIK}-\text{C}}$	$\rho^{+}_{\text{SIK}-\text{C}}$
2	21	6.56	—	6.56	—
3	49	1.48	-1.76	1.29	-1.92
4	113	2.05e-1	-2.37	3.71e-1	-1.49
5	257	6.69e-2	-1.36	3.01e-1	-0.25
6	577	4.07e-2	-0.61	2.74e-1	-0.12
7	1281	1.34e-2	-1.39	2.57e-1	-0.08
8	2817	4.37e-3	-1.42	2.48e-1	-0.04
9	6145	1.36e-3	-1.50	2.44e-1	-0.02
10	13313	4.06e-4	-1.56	2.42e-1	-0.01
11	28673	1.09e-4	-1.72	2.41e-1	-0.005
12	61441	1.99e-5	-2.22	2.41e-1	-0.003

TABLE 5.2: The performance of multilevel sparse collocation and plain sparse collocation for one asset European call option price following Parameter Set 1 with that initial condition is estimation at $\tau = 0.865$ using Gaussian and a connection constant $C = 2$. Error evaluated at 4000 uniform points in $[0.4E, 1.6E]$ at time $t = 0$.

In Table 5.2, MuSIK-C using the Gaussian shows its increasing convergence rate while SIK-C using the Gaussian doesn't converge. In Table 5.3, we use the MQ to replace the Gaussian. Both MuSIK-C and SIK-C break down at the level 12 due to the ill-condition problem. As shown in Figure 5.6, MuSIK-C achieves good convergence at high levels. However, there are instances where MuSIK-C with MQ

Level	Nodes	$E^{+}_{\text{MuSIK}-\text{C}}$	$\rho^{+}_{\text{MuSIK}-\text{C}}$	$E^{+}_{\text{SIK}-\text{C}}$	$\rho^{+}_{\text{SIK}-\text{C}}$
2	21	6.24	—	6.24	—
3	49	1.63	-1.58	1.63	-1.58
4	113	7.54e-2	-3.68	7.14e-2	-3.75
5	257	2.74e-2	-1.23	3.02e-2	-1.05
6	577	2.86e-2	0.05	1.42e-2	-0.93
7	1281	9.39e-3	-1.40	6.05e-3	-1.07
8	2817	3.00e-3	-1.45	3.13e-3	-0.84
9	6145	8.95e-4	-1.55	1.72e-3	-0.77
10	13313	2.43e-4	-1.69	9.00e-4	-0.84
11	28673	5.52e-5	-1.93	5.45e-4	-0.65
12	61441	2.44e-3	4.97	4.28e-3	2.70

TABLE 5.3: The performance of multilevel sparse collocation and plain sparse collocation for one asset European call option price following Parameter Set 1 with that initial condition is estimation at $\tau = 0.865$ using MQ and a connection constant $C = 2$. Error evaluated at 4000 uniform points in $[0.4E, 1.6E]$ at time $t = 0$.

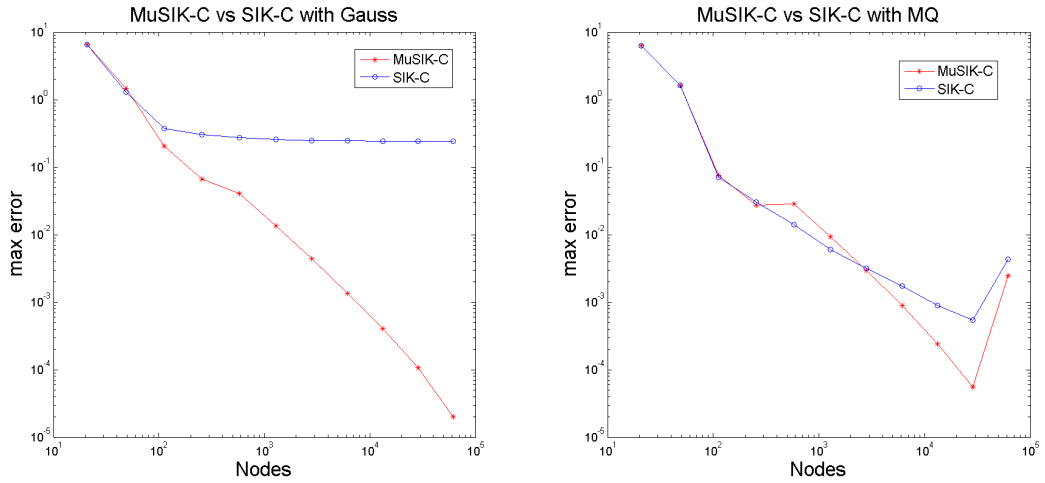

FIGURE 5.6: The performance of multilevel sparse collocation and plain sparse collocation for one asset European call option price following Parameter Set 1 with that initial condition is estimation at $\tau = 0.865$ using Gaussian(left) and MQ(right), $C = 2$. Error evaluated at 4000 uniform points in $[0.4E, 1.6E]$ at time $t = 0$.

basis functions performs worse than SIK-C with MQ in the mid-range. In Figure 5.7, the initial condition is the analytical solution at $\tau = 0.865$, we observe there is also a jump in the performance of multilevel sparse collocation as stated above. One possible explanation is that our algorithm to determine τ is still a method for choice. It just returns us a possible relatively smooth initial condition rather than one exact smooth one. Meanwhile, MuSIK-C is a method by approximating

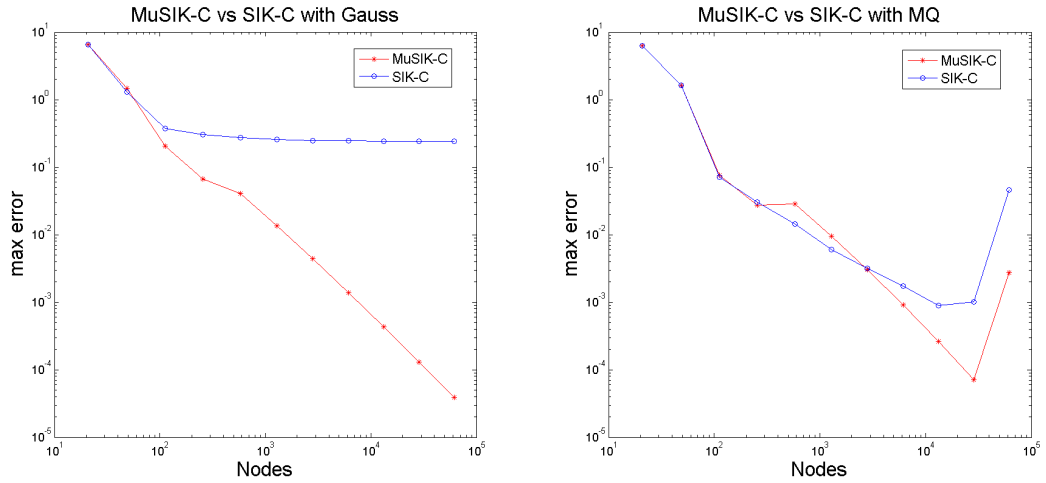

FIGURE 5.7: The performance of multilevel sparse collocation and plain sparse collocation for one asset European call option price following Parameter Set 1 with that initial condition is analytical solution at $\tau = 0.865$ using Gaussian(left) and MQ(right), $C = 2$. Error evaluated at 4000 uniform points in $[0.4E, 1.6E]$ at time $t = 0$.

the residuals, so it might take more steps to overcome the problem caused by non-smooth conditions. Figures 5.6 and 5.7 indicate the stop time $\tau = 0.865$ is not such perfect as we expected as there are some noise results. However, it is still a practical choice.

Now we present an additional Parameter Set 2 for continuous dividend European call option with a shorter expiry as shown in Table 5.4. We still use the restricted variable set in Table 5.1 into the MOL to predetermine a spot time τ, except we choose a smaller time step $\Delta t = 0.0001$ for the short maturity. Algorithm 3 terminates the MOL when $\tau = 0.236$.

Parameter Values	
σ	0.3
r	0.05
T	0.25
E	15
q	0.01
S_{min}	0
S_{max}	3E

TABLE 5.4: Parameter Set 2 for continuous dividend European call option.

Level	Nodes	$E^+_{\text{MuSIK}-\text{C}}$	$\rho^+_{\text{MuSIK}-\text{C}}$	$E^+_{\text{SIK}-\text{C}}$	$\rho^+_{\text{SIK}-\text{C}}$
2	21	1.04	—	1.04	—
3	49	2.30e-1	-1.78	2.06e-1	-1.91
4	113	2.54e-2	-2.63	4.11e-2	-1.93
5	257	1.02e-2	-1.11	3.58e-2	-0.17
6	577	1.24e-2	0.25	3.37e-2	-0.07
7	1281	4.73e-3	-1.21	3.07e-2	-0.11
8	2817	1.80e-3	-1.22	2.94e-2	-0.05
9	6145	6.57e-4	-1.29	2.88e-2	-0.03
10	13313	2.24e-4	-1.39	2.85e-2	-0.01
11	28673	7.08e-5	-1.50	2.84e-2	-0.006
12	61441	2.03e-5	-1.64	2.83e-2	-0.003

TABLE 5.5: The performance of multilevel sparse collocation and plain sparse collocation for one asset dividend European call option price following Parameter Set 2 with that initial condition is estimation at $\tau = 0.236$ using Gaussian and a connection constant $C = 2$. Error evaluated at 4000 uniform points in $[0.4E, 1.6E]$ at time $t = 0$.

Level	Nodes	$E^+_{\text{MuSIK}-\text{C}}$	$\rho^+_{\text{MuSIK}-\text{C}}$	$E^+_{\text{SIK}-\text{C}}$	$\rho^+_{\text{SIK}-\text{C}}$
2	21	9.76e-1	—	9.76e-1	—
3	49	2.53e-2	-1.59	2.56e-1	-1.58
4	113	1.94e-2	-3.07	2.13e-2	-2.98
5	257	5.30e-3	-1.58	5.42e-3	-1.66
6	577	1.03e-2	0.82	3.01e-3	-0.73
7	1281	3.98e-3	-1.19	1.05e-3	-1.32
8	2817	1.49e-3	-1.25	5.21e-4	-0.89
9	6145	5.47e-4	-1.28	2.82e-4	-0.79
10	13313	1.87e-4	-1.39	1.47e-4	-0.84
11	28673	5.74e-5	-1.54	7.51e-5	-0.87
12	61441	1.53e-5	-1.73	3.81e-5	-0.89

TABLE 5.6: The performance of multilevel sparse collocation and plain sparse collocation for one asset dividend European call option price following Parameter Set 2 with that initial condition is estimation at $\tau = 0.236$ using MQ and a connection constant $C = 2$. Error evaluated at 4000 uniform points in $[0.4E, 1.6E]$ at time $t = 0$.

Numerical results for Parameter Set 2 are displayed in Tables 5.5, 5.6 and Figure 5.8. They demonstrate similar performance as the previous example, but the convergence rates of MuSIK-C methods are not such good compared to the previous convergence rates in Tables 5.2 and 5.3. Moreover, the jump in the performance of MuSIK-C is much clearer. This phenomenon indicates that the initial condition at $\tau = 0.236$ is still not smooth enough.

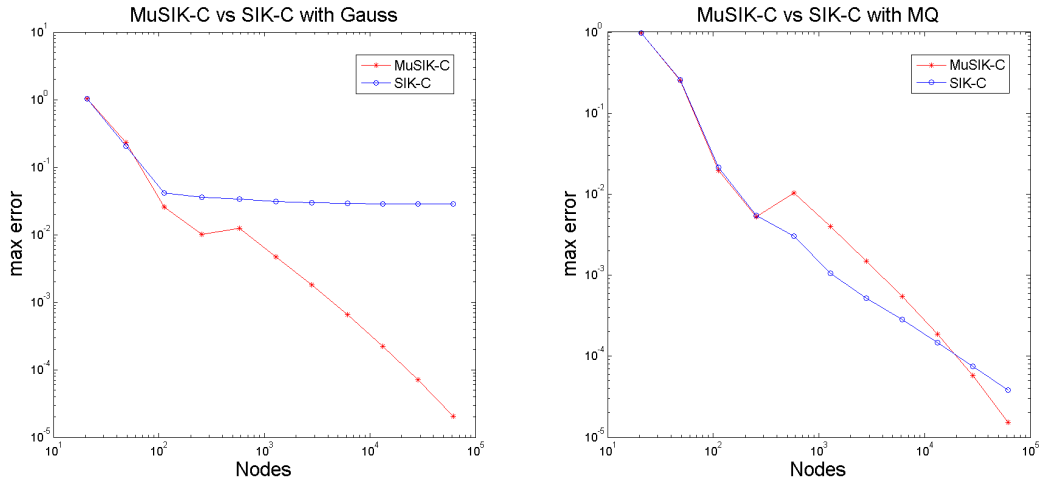

FIGURE 5.8: The performance of multilevel sparse collocation and plain sparse collocation for one asset dividend European call option price following Parameter Set 2 with that initial condition is estimation at $\tau = 0.236$ using Gaussian(left) and MQ(right), $C = 2$. Error evaluated at 4000 uniform points in $[0.4E, 1.6E]$ at time $t = 0$.

As so far, we can demonstrate that we can only apply MuSIK-C to solve PDEs with smooth conditions. So in option pricing, it is significant to take a relatively smooth approximation at an earlier time τ for the implementation of MuSIK-C. However, the current scheme we proposed is still a method of choice. Meanwhile, in order to guarantee the accuracy of the approximation at τ, we utilise a huge number of nodes. This procedure increases the whole computational cost and makes the combination method not comparable to other methods. As a result, one of our future direction is to employ other efficient methods to determine the time τ.

5.2 Margrabe Option pricing

In this section, we would like to exploit the performance of MuSIK-C and SIK-C methods under 2 assets options pricing. The two assets *Black-Scholes equation* is displayed as:

$$\begin{aligned} \frac{\partial u}{\partial t} &+ \frac{1}{2}\sigma_1^2 S_1^2 \frac{\partial^2 u}{\partial S_1^2} + \rho\sigma_1\sigma_2 S_1 S_2 \frac{\partial^2 u}{\partial S_1 S_2} + \frac{1}{2}\sigma_2^2 S_2^2 \frac{\partial^2 u}{\partial S_2^2} \\ &+ (r - q_1) S_1 \frac{\partial u}{\partial S_1} + (r - q_2) S_2 \frac{\partial u}{\partial S_2} - ru = 0. \end{aligned}$$

There is plenty of literature about high dimension option pricing, such as basket options [13, 75, 97], high dimension American option [32], spread options [67, 76]. For the purpose to check our method in a closed domain instead of particular locations, we choose the Margrabe option as our primary target in high dimensions. The Margrabe option is a special kind of spread option as strike price is zero and its analytical solution is proposed by Margrabe [88] in 1978. The holder can exchange the second asset for the first at maturity time T. So the pay off function P at time T is

$$P(T, S_1, S_2) = \max\{0, S_1(T) - S_2(T)\}.$$

An expression for the Margrabe option

$$C(t_0, s_1, s_2) = E\left[e^{-r(T-t_0)} \max\left(S_1(T) - S_2(T), 0\right)\right],$$

$S_1(t)$ and $S_2(t)$ satisfy

$$\begin{aligned} dS_1(t) &= (r - q_1)S_1(t)dt + \sigma_1 S_1(t)d\omega_1, \quad S_1(t_0) = s_1, \\ dS_2(t) &= (r - q_2)S_2(t)dt + \sigma_2 S_2(t)d\omega_2, \quad S_2(t_0) = s_2, \end{aligned}$$

where ω_1 and ω_2 are Brownian motions with $E\left[d\omega_1 d\omega_2\right] = \rho dt$. The solutions for $S_i(t)$ are

$$S_i(t) = s_i e^{(r-q_i)(t-t_0)} M_i(t_0, t), \quad i = 1, 2,$$

$$M_i(t_0, t) = \exp\left(\sigma_i(\omega_i(t) - \omega_i(t_0)) - \frac{1}{2}\sigma_i^2(t - t_0)\right), \quad i = 1, 2.$$

We observe that option price C does not depend on the riskless rate r. We shall assume that $r = 0$. So it is not accurate to estimate boundary values by backward formula. When we deal with boundaries, we fix one variable as the strike price and consider the problem as depending on just one European option. Once S_1 is fixed, it is a put option. Conversely, it is a call option. This method is also utilised by Fasshauer [32]. Therefore, we can use the procedure in Section 5.1 to construct

boundary grids. The analytical formula of the Margrabe option is

$$\begin{aligned} C(t, S_1, S_2) &= e^{-q_1(T-t)} S_1 N(d_1) - e^{-q_2(T-t)} S_2 N(d_2), && (5.2) \\ d_1 &= \frac{\log(\frac{S_1}{S_2}) + (q_2 - q_1 + \frac{1}{2}\sigma^2)(T-t)}{\sigma\sqrt{T-t}}, && (5.3) \\ d_2 &= d_1 - \sigma\sqrt{T-t}, && (5.4) \end{aligned}$$

where $\sigma = \sqrt{\sigma_1^2 + \sigma_2^2 - 2\sigma_1\sigma_2\rho}$. Other notations have the same meaning as those in the previous section.

Parameter Values	
σ_1	0.15
σ_2	0.15
ρ	0.5
T	1
E	0
q_1	0
q_2	0

TABLE 5.7: Parameter Set 3 for Margrabe option.

In the example, we use Parameter Set 3 in Table 5.7. The Margrabe option is a bet on the values of two risky assets. The most useful price of this option is when these two values near to each other. So we choose the interesting spatial domain as $\Omega = [80, 120]^2$ in this example.

As discussed before, we choose a larger domain $\widetilde{\Omega}$ to avoid the influence of the boundaries when applying the MOL to determine the estimated initial condition. Therefore, we set $\widetilde{\Omega} = [50, 150]^2$. As so far, we did not develop an effective and appropriate method to determine the stop time τ for the Margrabe option. Therefore, a terminate time spot τ^{end} when we must stop is chosen to be $0.8T$. We utilise the MOL with 10,000 uniform nodes in the domain $\widetilde{\Omega}$ and $\Delta t = 0.001$ to obtain an approximation at τ^{end}, see Figure 5.9. We can see that the largest deviation along the diagonal as expected.

Afterwards, we apply MuSIK-C and SIK-C method in the domain $\Omega_t^* = [0, 0.8T] \times [80, 120]^2$. We still use $E^+_{\text{MuSIK}-\text{C}}$ and $E^+_{\text{SIK}-\text{C}}$ to measure the errors and use $\rho^+_{\text{MuSIK}-\text{C}}$ and $\rho^+_{\text{IK}-\text{C}}$ to represent the slopes as defined before. Details are displayed in the Tables 5.9 and 5.8. The corresponding results are plotted in Figure 5.10.

As shown in Figure 5.10, MuSIK-C does not work from the first step. The reason might be that our estimated initial condition at $\tau = 0.8T$ is still not smooth enough, but it is not singular as that at the maturity time T. Therefore, our MuSIK-C method offsets the influence from initial condition firstly by producing a rough approximation. Afterwards, MuSIK-C using the Gaussian works well with a rapid convergence rate around 1.9 while MuSIK-C using MQ only expresses a slow convergence rate at the last step.

Level	Nodes	$E^+_{\mathrm{MuSIK-C}}$	$\rho^+_{\mathrm{MuSIK-C}}$	$E^+_{\mathrm{SIK-C}}$	$\rho^+_{\mathrm{SIK-C}}$
3	225	5.35e-2	—	5.35e-2	—
4	593	1.64e-1	1.16	5.58e-2	0.04
5	1505	3.54e-2	-1.65	4.81e-2	-0.16
6	3713	6.37e-3	-1.90	2.47e-2	-0.74
7	8961	1.27e-3	-1.83	1.17e-2	-0.84
8	21249	8.87e-4	-0.42	6.50e-3	-0.68

TABLE 5.8: The performance of multilevel sparse collocation and plain sparse collocation for Margrabe with an earlier terminal value of $0.8T$ using MQ and a connection constant $C = 2$. Error evaluated at 10,000 uniform points in $[90, 110]^2$ at time $t = 0$.

Level	Nodes	$E^+_{\mathrm{MuSIK-C}}$	$\rho^+_{\mathrm{MuSIK-C}}$	$E^+_{\mathrm{SIK-C}}$	$\rho^+_{\mathrm{SIK-C}}$
3	225	9.27e-1	—	9.27e-1	—
4	593	8.70e-1	-0.06	8.02e-1	-0.15
5	1505	1.69e-1	-1.76	6.84e-1	-0.17
6	3713	3.30e-2	-1.81	6.39e-1	-0.08
7	8961	6.64e-3	-1.82	6.15e-1	-0.04
8	21249	1.32e-3	-1.87	6.07e-1	-0.02

TABLE 5.9: The performance of multilevel sparse collocation and plain sparse collocation for Margrabe with an earlier terminal value of $0.8T$ using Gaussian and a connection constant $C = 2$. Error evaluated at 10,000 uniform points in $[90, 110]^2$ at time $t = 0$.

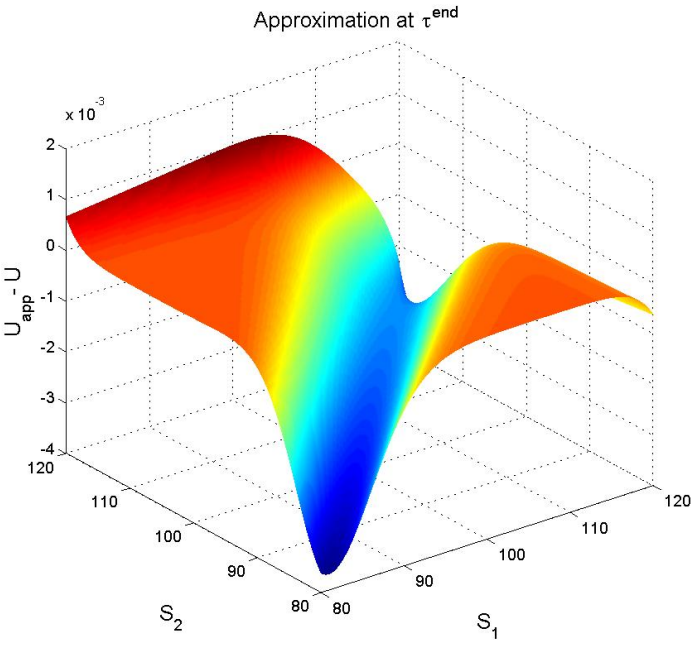

FIGURE 5.9: The error between the true surface at the new terminal condition and the approximated surface from MOL.

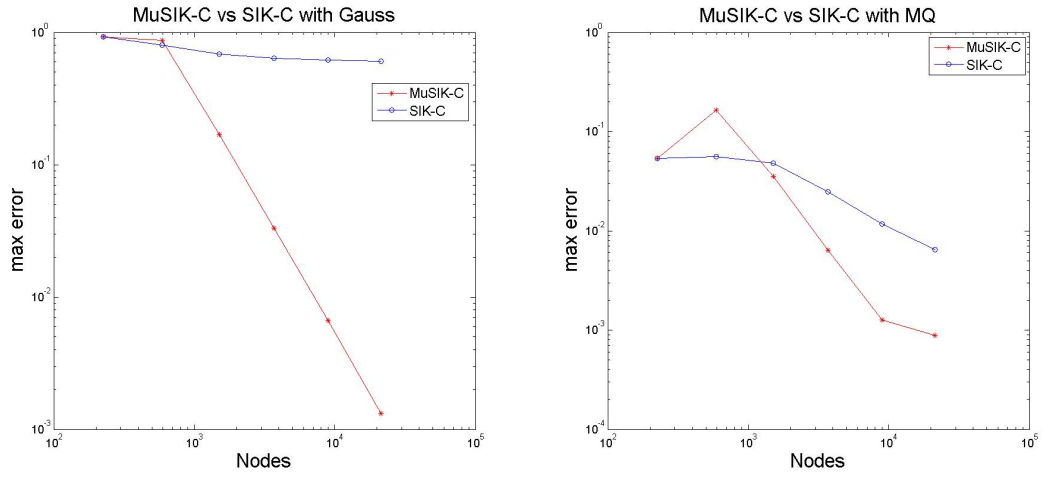

FIGURE 5.10: The performance of multilevel sparse collocation and plain sparse collocation for Margrabe option price following Parameter Set 3 with that initial condition is estimation at $\tau^{end} = 0.8T$ using Gaussian(left) and MQ(right), $C = 2$. Error evaluated at 10,000 uniform points in $[90, 110]^2$ at time $t = 0$.

5.3 Richardson Extrapolation

Richardson extrapolation (RE) is a quite useful and often used acceleration method. It is used to improve the rate of convergence of a sequence. In these pages, it is proposed that Richardson extrapolation can be employed to improve the convergence of the method. One famous classical application is Romberg integration. The basic idea behind that is to utilise Richardson extrapolation repeatedly on the trapezoidal rule. The general Romberg formula is

$$T_{m,j} = \frac{4^{j-1}T_{m,j-1} - T_{m-1,j-1}}{4^{j-1} - 1}, \quad j = 2, 3, \ldots (m \geq j). \tag{5.5}$$

The term $T_{m,1}$ means trapezoidal rule with $2^m + 1$ equally distributed points in the integration interval $[a, b]$ and it satisfies equation

$$I - T_{m,1} = C_1 h^2 + C_2 h^4 + \mathcal{O}\left(h^6\right), \quad h = \frac{b-a}{2^m}, \quad m \geq 1. \tag{5.6}$$

Replacing h with $\frac{h}{2}$, we can apply RE on Equation (5.6) to obtain

$$I - T_{m+1,2} = C_2' h^4 + \mathcal{O}\left(h^6\right), \quad m \geq 1 \tag{5.7}$$

$$T_{m+1,2} = \frac{4T_{m+1,1} - T_{m,1}}{3}, \quad m \geq 1, \tag{5.8}$$

where Equation (5.8) is equivalent to Simpson rule.

Now let us focus on the performance of Richardson extrapolation on our MuSIK-C method. Even though we do not know exactly what the convergence rate is, we also can utilise Richardson Extrapolation to advance our numerical results. Depending on the numerical results in Section 5.1.3, we just take partial numerical data. Because the rate of approximations are always oscillating at the beginning, and the condition number at the last step is a problem that we have to consider. The estimations are not helpful for utilising Richardson extrapolation.

Suppose our approximation $u\left(\frac{1}{N}\right)$ has a relation depending on the number of nodes N with the target function U as

$$u\left(\frac{1}{N}\right) = u + C\left(\frac{1}{N}\right)^{\alpha_1} + \mathcal{O}\left(\left(\frac{1}{N}\right)^{\alpha_2}\right), \quad 0 < \alpha_1 < \alpha_2. \tag{5.9}$$

For a sequence of number $N_i, i = 1, \ldots, n$, we set $B_i = \frac{N_{i+1}}{N_i}, i = 1, \ldots, n-1$, so

$$u\left(\frac{1}{N_i}\right) = u + C\left(\frac{1}{N_i}\right)^{\alpha_1} + \mathcal{O}\left(\left(\frac{1}{N_i}\right)^{\alpha_2}\right), \tag{5.10}$$

and

$$u\left(\frac{1}{N_{i+1}}\right) = u + C\left(\frac{1}{N_{i+1}}\right)^{\alpha_1} + \mathcal{O}\left(\left(\frac{1}{N_{i+1}}\right)^{\alpha_2}\right), \tag{5.11}$$

$$\Rightarrow u\left(\frac{1}{N_{i+1}}\right) = u + \frac{1}{B_i^{\alpha_1}} C\left(\frac{1}{N_i}\right)^{\alpha_1} + \mathcal{O}\left(\left(\frac{1}{N_i}\right)^{\alpha_2}\right). \tag{5.12}$$

By guessing an index β, we can form a new approximation

$$\frac{B_i^{\beta} u\left(\frac{1}{N_{i+1}}\right) - u\left(\frac{1}{N_i}\right)}{B_i^{\beta} - 1} = u + \frac{B_i^{\beta-\alpha_1} - 1}{B_i^{\beta} - 1} C\left(\frac{1}{N_i}\right)^{\alpha_1} + \mathcal{O}\left(\left(\frac{1}{N_i}\right)^{\alpha_2}\right). \tag{5.13}$$

From Equation (5.13), we know

- If $\beta = \alpha_1$, we can obtain high convergence rate α_2.
- If $\beta \gg \alpha_1$, the new Equation (5.13) will coincide with Equation (5.11).
- If $\beta > \alpha_1$ and β is not too big, we can achieve an approximation parallel to the Equation (5.11).

In the following tables, we use E_{RE}^{+} to represent the max absolute error from Richardson extrapolation and ρ_{RE}^{+} to represent the slope. The definitions are similar as before.

Level	Nodes	$E^+_{\text{MuSIK}-\text{C}}$	$\rho^+_{\text{MuSIK}-\text{C}}$	E^+_{RE}	ρ^+_{RE}
6	577	4.07e-2	-0.61	—	—
7	1281	1.34e-2	-1.39	4.09e-3	—
8	2817	4.37e-3	-1.42	1.47e-3	-1.30
9	6145	1.36e-3	-1.50	3.14e-4	-1.97
10	13313	4.06e-4	-1.56	6.10e-5	-2.12
11	28673	1.09e-4	-1.72	2.82e-6	-4.00

TABLE 5.10: Results of MuSIK-C with Gaussian from Table 5.2 and corresponding RE when $\beta = 1.7$.

Level	Nodes	$E^+_{\text{MuSIK}-\text{C}}$	$\rho^+_{\text{MuSIK}-\text{C}}$	E^+_{RE}	ρ^+_{RE}
6	577	2.86e-2	0.05	—	—
7	1281	9.39e-3	-1.40	5.78e-3	—
8	2817	3.00e-3	-1.45	1.77e-3	-1.51
9	6145	8.95e-4	-1.55	4.78e-4	-1.68
10	13313	2.43e-4	-1.69	1.11e-4	-1.88
11	28673	5.51e-5	-1.93	2.07e-5	-2.19

TABLE 5.11: Results of MuSIK-C with MQ from Table 5.3 and corresponding RE when $\beta = 2.3$.

Level	Nodes	$E^+_{\text{MuSIK}-\text{C}}$	$\rho^+_{\text{MuSIK}-\text{C}}$	E^+_{RE}	ρ^+_{RE}
6	577	1.24e-2	0.25	—	—
7	1281	4.73e-3	-1.21	2.59e-3	—
8	2817	1.80e-3	-1.22	7.46e-4	-1.58
9	6145	6.57e-4	-1.29	1.43e-4	-2.12
10	13313	2.24e-4	-1.39	2.83e-5	-2.10
11	28673	7.08e-5	-1.50	2.38e-6	-3.23

TABLE 5.12: Results of MuSIK-C with Gaussian from Table 5.5 and corresponding RE when $\beta = 1.5$.

Level	Nodes	$E^+_{\text{MuSIK}-\text{C}}$	$\rho^+_{\text{MuSIK}-\text{C}}$	E^+_{RE}	ρ^+_{RE}
6	577	1.03e-2	0.82	—	—
7	1281	3.98e-3	-1.19	2.86e-3	—
8	2817	1.49e-3	-1.25	5.80e-4	-2.03
9	6145	5.47e-4	-1.28	1.69e-4	-1.58
10	13313	1.87e-4	-1.39	3.93e-5	-1.89
11	28673	5.74e-5	-1.54	4.43e-6	-2.85

TABLE 5.13: Results of MuSIK-C with MQ from Table 5.6 and corresponding RE when $\beta = 1.6$.

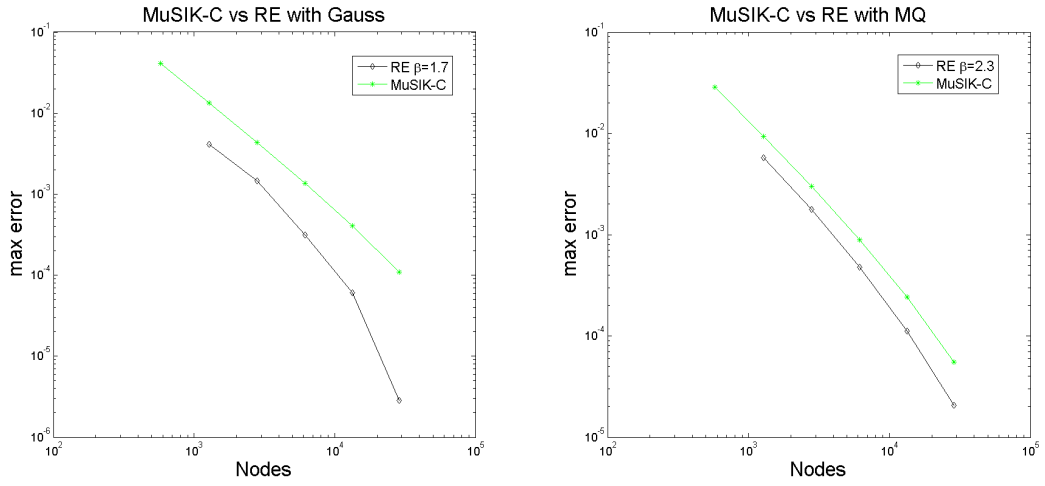

FIGURE 5.11: RE method applied on MuSIK-C with Gaussian(left) and MQ(right) from Table 5.10 and 5.11.

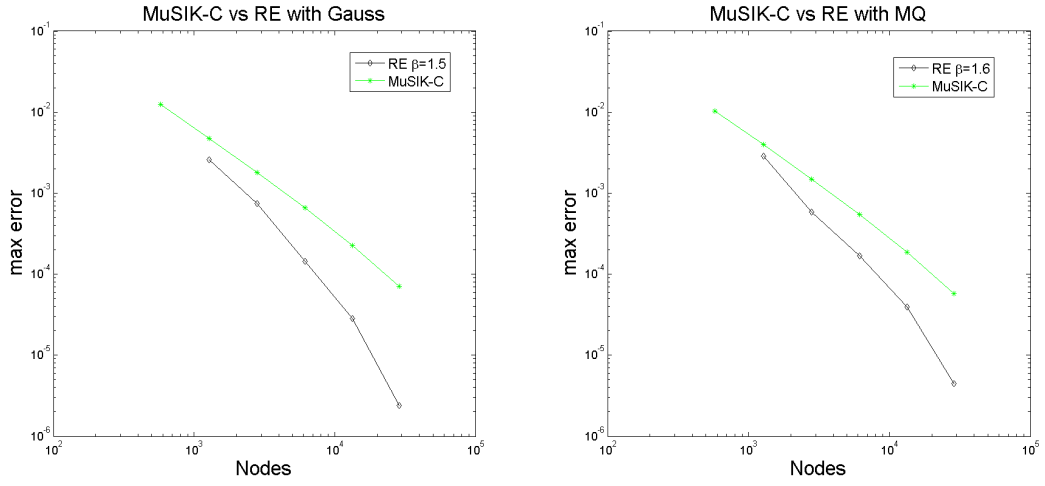

FIGURE 5.12: RE method applied on MuSIK-C with Gaussian(left) and MQ(right) from Table 5.12 and 5.13.

As we can see, the slopes of MuSIK-C are increasing slowly in Tables 5.10, 5.11, 5.12 and 5.13. So we guess the value of β in different situations is around the absolute value of the last slope. The Richardson extrapolation results seem to be exponential convergence in Figures 5.11 and 5.12. In the face of this phenomena, we just can have a guess the real convergence rate is a function of data size. It will be an interesting future work to figure out the reason.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In Chapter 3, we briefly reviewed the Kansa's collocation method, the Method of Lines (MOL) and the space-time method. As the numerical example shown in Section 3.3, we observed it is helpful to choose a larger spatial domain for using the MOL to solve the *Black-Scholes equation*. Meanwhile, when we utilised the space-time method with MQ basis function directly on the original domain, we achieved a series of convergence approximations on the whole spatial domain and it has a similar convergence rate as the MOL. Moreover, we reduced the number of nodal points by considering time as one spatial dimension.

In Chapter 4, we extended the multilevel sparse grid kernels (MuSIK) interpolation method into the collocation method to form MuSIK-C. MuSIK-C method has the desirable properties of the RBFs. These properties are easy to implement, mesh-free and applicability in high dimension. This method also inherits the advantages of MuSIK method, for instance, the combination technique that divides the sparse grid into a series of more coarsen directionally uniform sub-grids, rapid convergence rate, parallel approach. Each of these advantages makes MuSIK-C more appropriate for high dimensional situations. We successfully obtained good solutions for elliptic and parabolic problems with smooth boundary conditions and smooth initial condition up to four dimensions (including time). From the

experiments in Chapter 4, we observe that the convergence rates of MuSIK-C are increasing slowly when the system is not ill-conditioned. This phenomenon suggests MuSIK-C might be a spectral method even though we did not prove it theoretically. Compared to some very recent mesh-based methods, we have demonstrated that our MuSIK-C has the same level performance in low dimensional problems and has a better performance in high dimensional situations. Moreover, even though we are using tensor product radial basis functions in the collocation, MuSIK-C still shows its superiority in approximating non-tensor product functions.

We found MuSIK-C has its limitation in dealing with *Black-Scholes* PDE which is a parabolic problem whose initial condition is not smooth. In order to mitigate MuSIK-C method's drawback, we decide replace the non-smooth initial boundary, which is in the time direction, with a smooth estimation. Therefore, we developed a method to determine the stop time for the forcing movement and implement this method to price no dividend and continuous dividend European options. We not only make MuSIK-C applicable in *Black-Scholes* PDE, but also observe that it seems to have an exponential convergence rate. Then, in Section 5.3, we apply Richardson extrapolation method to accelerate the convergence speed and achieve more accurate approximations. In the Margrabe option pricing, we don't have a method to determine when we should stop the forcing movement currently. Instead, we choose an end time manually.

It is interesting to observe that SIK-C with the Gaussian basis function does not converge while SIK-C method with multiquadric basis function can have a slow convergence rate. Currently, we do not have a proper explanation. MuSIK-C method with both basis functions can provide superior performance. However, the ill-conditioning problem always accompanies with our method when using larger shape parameter, applying high levels and handling with high dimension problem. So the first requirement is to reduce the condition number under the situation that convergence rate is guaranteed. Afterwards, the performance of MuSIK-C will be more attractive.

6.2 Future work

The approximate cardinal basis function (ACBF) is a good choice to deal with the badly conditioned system. Ling and Kansa proposed preconditioning schemes that are based on domain decomposition method and least-squares method in [77] and [78] separately. In [8], the authors investigated and compared the performances of a class of preconditioners on Poisson, modified Helmholtz and Helmholtz equations. No matter to do interpolation problem or collocation problem with RBFs, there is a very significant process to find the coefficients. Usually, we obtain the coefficients by solving the global Gaussian elimination method which is expensive as it requires $\mathcal{O}(N^3)$ flops. Suppose a matrix-vector equation is

$$\mathbf{A}\boldsymbol{\lambda} = \mathbf{f}. \tag{6.1}$$

The idea behind preconditioning scheme is just to construct a preconditioner $\mathbf{W}$ such that $\mathbf{WA} = \mathbf{I}$, here $\mathbf{I}$ is identify matrix. So the coefficients $\boldsymbol{\lambda} = \mathbf{Wf}$. Assuming there are only α non-zero elements in each row of $\mathbf{W}$, multiplying the preconditioner by a vector can be performed in $\mathcal{O}(\alpha N)$ flops.

Recall Equation (2.18)

$$\Phi = \Phi_1 \otimes \cdots \otimes \Phi_d.$$

Φ means the interpolation matrix constructed by the anisotropic tensor product basis functions. Φ_i are the matrices constructed by the nodes in the ith direction. We can obtain the inverse matrix Φ^{-1} instead of solving the full matrix by equation

$$(\Phi)^{-1} = (\Phi_1)^{-1} \otimes \cdots \otimes (\Phi_d)^{-1}.$$

This is one kind of the preconditioner $\mathbf{W}$ as we talked before. What is more, it converts a d-dimensional problem to d one dimensional problems. Therefore, it will be very helpful for high-dimensional approximations. One of our future work is trying to apply a similar idea on the collocation matrix.

There are also many other jobs waiting for us. In order to improve the run time, we would like to employ quasi quadrature method as discussed in [115].

As mentioned in the introduction, the sparse grid was introduced to deal with the curse of dimension. We are always interested in exploiting more higher dimension problems, like $d = 10$. In this thesis, we just tested unsymmetric collocation with MuSIK-C. In the near future, we will try to solve PDEs with MuSIK-C based on symmetric collocation. Moreover, we observed that MuSIK-C has difficulty in solving the parabolic problem with a non-smooth initial condition. Even though we introduced a combination method to fix that problem, we still would like to apply sparse grid into Galerkin method to form a multilevel sparse grid kernels Galerkin method. Certainly, the error analysis is always our purpose.

Bibliography

[1] K. I. Babenko. Approximation by trigonometric polynomials in a certain class of periodic functions of several variables. *Soviet Mathematics Doklady*, 1:672–675, 1960.

[2] R. E. Bank, P. S. Vassilevski, and L. T. Zikatanov. Arbitrary dimension convection-diffusion schemes for space-time discretizations. *Journal of Computational and Applied Mathematics*, 2016.

[3] G. Baszenski, F. J. Delvos, and S. Jester. Blending approximations with sine functions. *In Numerical Methods in Approximation Theory*, 9:1–19, 1992.

[4] R. K. Beatson, J. B. Cherrie, and C. T. Mouat. Fast fitting of radial basis functions: methods based on preconditioned GMRES iteration. *Advance in Computational Mathematics*, 11(2-3):253–270, 1999.

[5] R. Belmann. *Adaptive Control process: a guide tour*. Princeton Universoty Press, Princeton, 1961.

[6] F. Black and M. Scholes. The pricing of corporate liabilities. *Journal of Political Economy*, 81(3):637–654, 1973.

[7] M. Bozzini, L. Lenarduzzi, M. Rossini, and R. Schaback. Interpolation with variably scaled kernels. *IMA Journal of Numerical Analysis*, 35(1):199-219, 2015.

[8] D. Brown, L. Ling, E. J. Kansa, and J. Levesley. On approximate cardinal preconditioning methods for solving PDEs with radial basis functions. *Engineering Analysis with Boundary Elements*, 29(4):343–353, 2005.

[9] M. D. Buhmann. Spectral convergence of multiquadric interpolation. *Proceeding of the Edinburgh Mathematical Society*, 36(-):319–333, 1993.

[10] M. D. Buhmann. *Radial Basis Functions: Theory and Implementations*. Cambridge University Press,Cambridge, 2003.

[11] H. J. Bungartz and M. Griebel. Sparse grids. *Acta Numerica*, 13:147–269, 2004.

[12] H. J. Bungartz, M. Griebel, and U. Rüde. Extrapolation, combination, and Sparse grids grid techniques for elliptic boundary value problems. *Computer Methods in Applied Mechanics and Engineering*, 116(1-4):243–252, 1994.

[13] H. J. Bungartz, A. Heinecke, D. Puger, and S. Schraufstetter. Option pricing with a direct adaptive sparse grid approach. *Journal of Computational and Applied Mathematics*, 236:3741–3750, 2012.

[14] A. Cangiani, E. H. Georgoulis, and P. Houston. hp-Version discontinuous Galerkin methods on polygonal and polyhedral meshes. *Mathematical Models and Methods in Applied Sciences*, 24(10):2009–2041, 2014.

[15] T. Cecil, J. Qian, and S. Osher. Numerical method for high dimensional hamilton-jacobi equation using radial basis functions. *Journal of Computational Physics*, 196(-):327–347, 2004.

[16] R. Chan and S. Hubbert. Options pricing under the one-dimensional jump-diffusion model using the radial basis function interpolation scheme. *Review of Derivatives Research*, 17(2):161–189, 2014.

[17] R. Charles, T. A. Driscoll, B. Fornberg, and G. Wright. Observations on the behavior of radial basis function approximations near boundaries. *Computers and Mathematics with Applications*, 43(3):473–490, 2002.

[18] C. S. Chen, M. A. Golberg, and S. Karur. Improved multiquadric approximation for partial differential equations. *Engineering Analysis with Boundary Elements*, 18(1):9–17, 1996.

[19] A. H-D Cheng and C. A. Brebbia. *Boundary Elements and Other Mesh Reduction Methods XXXVIII*. WIT press, 2015.

[20] F. J. Delvos. d-variate Boolean interpolation. *Journal of Approximation Theory*, 34:99–114, 1982.

[21] L. B. Da Veiga, A. Buffa, G. Sangalli, and R. Vázquez. Mathematical analysis of variational isogeometric methods. *Acta Numerica*, 23:157–287, 2014.

[22] Z. Dong. Provide communication.

[23] M. R. Dubal. Domain decomposition and local refinement for multiquadric approximations I. Second-order equations in one-dimension. *Journal of Applied Computer Science Methods*, 1(1):146–171, 1994.

[24] M. R. Dubal, R. A. Matzner, and S. R. Olvera. *In approaches to numerical relativity*. Cambridge University Press, Cambridge, 1993.

[25] T. A. Driscoll and B. Fornberg. Interpolation in the limit of increasingly flat radail basis functions. *Computers and Mathematics with Applications*, 43(3):413–422, 2002.

[26] P. Farrell and H. Wendland. RBF multiscale collocation for second order elliptic boundary value problems. *SIAM Journal on Numerical Analysis*, 51(4):2403–2425, 2013.

[27] G. E. Fasshauer. Solving partial differential equations by collocation with radial basis functions. *Proceedings of Chamonix*, 1997(-):1–8, 1996.

[28] G. E. Fasshauer. Solving differential equations with radial basis functions: multilevel methods and smoothing. *Advances in Computational Mathematics*, 11(-):139–159, 1999.

[29] G. E. Fasshauer. *Meshfree approximation methods with matlab*. World Scientific, 2007.

[30] G. E. Fasshauer and J. W. Jerome. Multistep approximation algorithms: improved convergence rates through postconditioning with smoothing kernels. *Advances in Computational Mathematics*, 10:1–27, 1999.

[31] G. E. Fasshauer and M. J. McCourt. Stable Evaluation of Gaussian Radial Basis Function Interpolants. *SIAM Journal on Scientific Computing*, 34(2):A737–A762, 2012.

[32] G. E. Fasshauer, A. Q. M. Khaliq, and D. A. Voss. Using meshfree approximation for multi asset American option problems. *Chinese Institute Engineers*, 24(-):563–571, 2004.

[33] G. E. Fasshauer and J. G. Zhang. On choosing optimal shape parameters for rbf approximation. *Numerical Algorithms*, 45(1-4):345–368, 2007.

[34] G. E. Fasshauer and J. G. Zhang. Preconditioning of Radial Basis Function Interpolation Systems via Accelerated Iterated Approximate Moving Least Squares Approximation. In *Progress on Meshless Methods Volume 11 of the series Computational Methods in Applied Sciences* pp:57–75, Springer, New York, 2009.

[35] A. I. Fedoseyev, M. J. Friedman, and E. J. Kansa. Improved multiquadric method for elliptic partial differential equations via PDE collocation on the boundary. *Computers and Mathematics with Applications*, 43(3-5):439–455, 2002.

[36] M. S. Floater and A. Iske. Multistep scattered data interpolation using compactly supported radial basis functions. *Journal of Computational and Applied Mathematics*, 73(1-2):65–78, 1996.

[37] B. Fornberg and E. Larsson. A numerical study of some radial basis function based solution methods for elliptic pdes. *Computers and Mathematics with Applications*, 46(5-6):891–902, 2003.

[38] B. Fornberg, E. Larsson, and N. Flyer. Stable computation with Gaussian radial basis functions. *SIAM Journal on Scientific Computing*, 33(2):869–892, 2011.

[39] B. Fornberg and C. Piret. A stable algorithm for flat radial basis functions on a sphere. *SIAM Journal on Scientific Computing*, 30(1):60–80, 2007.

[40] B. Fornberg and G. Wright. Stable computation of multiquadric interpolants for all values of the shape parameter. *Computers and Mathematics with Applications*, 48(5-6):853–867, 2004.

[41] R. Franke. Scattered data interpolation: test of some methods. *Mathematics of Computation*, 38(157):181–200, 1982.

[42] C. Franke and R. Schaback. Solving partial differential equations by collocation using radial basis functions. *Applied Mathematics and Computation*, 93(1):73–82, 1998.

[43] C. Franke and R. Schaback. Convergence order estimates of meshless collocation methods using radial basis functions. *Adavaces in Computational Mathematics*, 8(-):381–399, 1998.

[44] J. Garcke and M. Griebel. On the parallelization of the sparse grid approach for data mining. In S. Margenov, J. Wasniewski, and P. Yalamov, editors, *Large-Scale Scientific Computations, Third International Conference, LSSC 2001, Sozopol, Bulgaria*, volume 2179 of *Lecture Notes in Computer Science*, pages 22–32, Springer, 2001.

[45] J. Garcke and M. Griebel. *Sparse grids and applications*. Springer, 2013.

[46] J. Garcke and M. Hegland. Fitting multidimensional data using gradient penalties and the sparse grid combination technique. *Computing*, 84(1-2):1–25, 2009.

[47] J. Garcke, M. Hegland, and O. Nielsen. Parallelisation of sparse grids for large scale data analysis. *ANZIAM Journal*, 48(1):11–22, 2006.

[48] E. H. Georgoulis, J. Levesley, and F. Subhan. Multilevel sparse kernel-based interpolation. *SIAM Journal on Scientific Computing*, 35(2):A815–A831, 2013.

[49] R. Geske. The valuation of compound options. *Journal of Financial Economics*, 7: 63–81, 1979.

[50] M. B. Giles and R. Carter. Convergence analysis of Crank-Nicolson and Rannacher time-marching. 2005.

[51] M. Griebel. The combination technique for the sparse grid solution of PDEs on multiprocessor machines. *Parallel Processing Letters*, 2(1):61–70, 1992.

[52] M. Griebel. Adaptive sparse grid multilevel methods for elliptic PDEs based on finite difference. *Computing*, 61(2):151–179, 1998.

[53] M. Griebel, D. Oeltz. A sparse grid space-time discretization scheme for parabolic problems. *Computing*, 81(1):1–34, 2007.

[54] M. Griebel, M. Schneider, and C. Zenger. A combination technique for the solution of sparse grid problems. In *Iterative methods in linear algebra (Brussels, 1991)*, pages 263–281. North-Holland, Amsterdam, 1992.

[55] S. J. Hales and J. Levesley. Error estimates for multilevel approximation using polyharmonic splines. *Numerical Algorithms*, 30:1–10, 2002.

[56] R. L. Hardy. Multiquadric equations of topography and other irregular surfaces. *Journal of Geophysical Research*, 76(8):1905–1915, 1971.

[57] R. L. Hardy Theory and applications of the multiquadric-biharmonic method. *Computers and Mathematics with Applications*, 19(8/9):163–208, 1990.

[58] P. W. Hemker. Sparse-grid finite-volume multigrid for 3D-problems. *Advances in Computational Mathematics*, 4(1-2):83–110, 1995.

[59] A. Heryudono, E. Larsson, A. Ramage, and L. von Sydow. Preconditioning for Radial Basis Function Partition of Unity Methods. *Journal of Scientific Computing*, 67(3):1089–1109, 2016.

[60] A. Heryudono, E. Larsson, and A. Safdari-Vaighani. A radial basis function partition of unity collocation method for convection diffusion equations

arising in financial applications. *SIAM Journal on Scientific Computing*, 2014.

[61] Y. C. Hon. A quasi-radial basis functions methods for American option pricing. *Computers and Mathematics with Applications*, 43(-):513–524, 2002.

[62] Y. C. Hon and X. Z. Mao. A radial basis function method for solving option pricing model. *Financial Engineering*, 8(-):1–24, 1999.

[63] Y. C. Hon, M. W. Lu, W. M. Xue, and Y. M. Zhu. Multiquadric method for the numerical solution of a biphasic mixture model. *Appled Mathematics Computation*, 88(2-3):153–176, 1997.

[64] Y. C. Hon and R. Schaback. On unsymmetric collocation by radial basis functions. *Applied Mathematics and Computation*, 119(2-3):177–186, 2001.

[65] Y. C. Hon and Z. Wu. Convergence error estimate in solving free boundary diffusion problem by radial basis functions method. *Engineering Analysis with Boundary Elements*, 27(-):73–79, 2003.

[66] C. Hu and C. W. Shu. A discontinuous galerkin finite element method for hamilton–jacobi equations. *SIAM Journal on Scientific Computing*, 21(2):666–690, 2006.

[67] T. R. Hurd and Z. Zhou. A Fourier Transform Method for Spread Option Pricing. *SIAM Journal on Financial Mathematics*, 1(1):142–157, 2010.

[68] A. Iske. Multiresolution methods in scattered data modelling. *Lecture Notes in Computational Science and Engineering, Springer-Verlag Berlin Heidelberg*, 37(-):83–102, 2004.

[69] A. Iske. Hierarchical scattered data filtering for multilevel interpolation schemes. In *Mathematical methods for curves and surfaces (Oslo, 2000)*, Vanderbilt University Press, Nashville, TN, 211–221.

[70] A. Iske and J. Levesley. Multilevel scattered data approximation by adaptive domain decomposition. *Numerical Algorithms*, 39:187–198, 2005.

[71] E. J. Kansa. Multiquadrics - a scattered data approximation scheme with applications to computational fluid-dynamics - I. *Computers and Mathematics with Applications*, 19(8-9):127–145, 1990.

[72] E. J. Kansa. Multiquadrics - a scattered data approximation scheme with applications to computational fluid-dynamics - II. *Computers and Mathematics with Applications*, 19(8-9):147–161, 1990.

[73] L. Kuipers and H. Niederreiter. *Uniform distribution of sequences*. Dover Publications, 2005.

[74] U. Langer, S. Moore, and M. Neumuller. Space-time isogeometric analysis of parabolic evolution equations. *Computer Methods in Applied Mechanics and Engineering*, 306:342–263, 2016.

[75] E. Larsson, U. Pettersson, J. Presson, and G. Marcusson. Improved radial basis function methods for multi-dimensional option pricing. *Computers and Mathematics with Applications*, 222(1):82–93, 2008.

[76] M. Li, S. Deng, and J. Zhou. Closed-form approximations for spread option prices and Greeks. *Journal of Derivatives*, 15:58–80, 2008.

[77] L. Ling and E. J. Kansa. Preconditioning for Radial Basis Functions with Domain Decomposition Methods. *Mathematical and Computer Modelling*, 40(13):1413–1427, 2004.

[78] L. Ling and E. J. Kansa. A least-squares preconditioner for radial basis functions collocation methods. *Advances in Computational Mathematics*, 23(1):31–54, 2005.

[79] L. T. Luh. On Wu and Schaback's error bound. *International Journal of Numerical Methods and Applications*, 1(2):155–174, 2009.

[80] L. T. Luh. The shape parameter in the Gaussian function. *Computers and Mathematics with Applications*, 63(3):687–694, 2012.

[81] L. T. Luh. The shape parameter in the Gaussian function II. *Engineering Analysis with Boundary Elements*, 37(6):988–993, 2013.

[82] J. Levesley and Z. Luo. Error estimates and convergence rates for variational hermite interpolation. *Journal of Approximation Theory*, 95(-):264–279, 1998.

[83] X. Ma and N. Zabaras. An adaptive hierarchical sparse grid collocation algorithm for the solution of stochastic differential equations. *Journal of Computational Physics*, 228(8):3084–3113, 2009.

[84] W. R. Madych. Miscellaneous error bounds for multiquadric and related interpolators. *Computers and Mathematics with Applications*, 24:121–138, 1992.

[85] W. R. Madych and S. A. Nelson. Multivariate interpolation and conditionally positive definite functions. *Approximation Theory and its Applications*, 4(-):77–89, 1988.

[86] W. R. Madych and S. A. Nelson. Multivariate interpolation and conditionally positive definite functions. II. *Mathematics of Computation*, 54(189):211–230, 1990.

[87] W. R. Madych and S. A. Nelson. Bounds on multivariate polynomials and exponential error estimates for multiquadric interpolation. *Journal of Approximation Theory*, 70(1):94–114, 1992.

[88] W. Margrabe. The value of an option to exchange one asset for another. *Journal of Finance*, 33(-):177–186, 1978.

[89] C. A. Micchelli. Interpolation of scattered data: distance matrix and conditionally positive definite function. *Constructive Approximation*, 2(-):11–22, 1986.

[90] D. E. Myers, S. De Iaco, D. Posa, and L. De Cesare. Space-Time Radial Basis Functions. *Computers and Mathematics with Applications*, 43:539-549, 2002.

[91] F. J. Narcowich, R. Schaback, and J. D. Ward. Multilevel interpolation and approximation. *Applied and Computational Harmonic Analysis*, 7:243–261, 1999.

[92] F. Nobile, R. Tempone, and C. Webster. A sparse grid stochastic collocation method for partial differential equations with random input data. *SIAM Journal of Numerical Analysis*, 46(5):2309–2345, 2008.

[93] Y. Ohtake, A. Belyaev, and H. P. Seidel. 3d scattered data interpolation and approximation with multilevel compactly supported rbfs. *Graphical Models*, 67:150–165, 2005.

[94] K. Parand and J. A. Rad. Kansa method for the solution of a parabolic equation with an unknown spacewise-dependent coefficient subject to an extra measurement. *Computer Physics Communications*, 184(3):582–595, 2013.

[95] A. Pena. Option pricing with radial basis functions: a tutorial. Technical report, Wilmott Magazine, 2005.

[96] M. J. D. Powell. *The theory of radial basis function approximation in 1990, in Advances in numerical analysis II.* Oxford University Press, 105–210, 1992.

[97] C. Reisinger and G. Wittum. Efficient hierarchical approximation of high-dimensional option pricing problems. *SIAM Journal on Scientific Computing*, 29(1):440–458, 2006.

[98] A. La Rocca, A. Hernandez Rosales, and H. Power. Radial basis function Hermite collocation approach for the solution of time dependent convection-diffusion problems. *Engineering Analysis with Boundary Elements*, 29:359–370, 2005.

[99] R. Roll. An analytical formula for unprotected American call options on stocks with known dividends. *Journal of Financial Economics*, 5: 251–58, 1977.

[100] 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*, 2(2), 2009.

[101] R. Schaback. Error estimates and condition numbers for radial basis function interpolation. *Advances in Computational Mathematics*, 3(-):251–264, 1995.

[102] W. E. Schiesser and G. W. Griths. *A compendium of partial differential equation models: Method of lines analysis with Matlab.* Cambridge University Press, 2009.

[103] A. Schreiber. *The method of Smolyak in multivariate interpolation.* PhD thesis, der Mathematisch-Naturwissenschaftlichen Fakultäten, der Georg-August-Universität zu Göttingen, 2000.

[104] C. Schwab, E. Suli, and R. Todor. Sparse finite element approximation of high-dimensional transport-dominated diffusion problems. *ESAIM: Mathematical Modelling and Numerical Analysis*, 42(05):777–819, 2008.

[105] V. Shcherbakov and E. Larsson. Radial basis function partition of unity methods for pricing vanilla basket options. *Computers & Mathematics with Applications*, 71(1):185–200, 2016.

[106] S. A. Smolyak. Quadrature and interpolation of formulas for tensor product of certain classes of functions. *Soviet Mathematics Doklady*, 4:240–243, 1963.

[107] S. Song and L. Tang. A tvd-type method for 2d scalar Hamilton Jacobi equations on unstructured meshes. *Journal of Computational and Applied Mathematics*, 195(1-2):182–191, 2006.

[108] S. E. Stead. Estimation of gradients from scattered data. *Journal of Mathematics*, 14(1):265–279, 1984.

[109] F. Subhan. *Multilevel Sparse Kernel-Based Interpolation.* Ph.D. Thesis, University of Leicester, 2011.

[110] L. V. Sydow, L. J. Hook, E. Larsson, E. Lindstrom, S. Milovanovic, J. Persson, V. Shcherbakov, Y. Shpolyanskiy, S. Siren, J. Toivanen, J. Walden, M. Wiktorsson, M. B. Giles, J. Levesley, J. Li, C. W. Oosterlee, M. J. Ruijter, A. Toropov, Y. Zhao. BENCHOP-the BENCHmarking project in option pricing. *International Journal of Computer Mathematics*, forthcoming 2015.

[111] M. Tatari and M. Dehghan. A method for solving partial differential equations via radial basis functions: Application to the heat equation. *Engineering Analysis with Boundary Elements*, 34(3):206–212, 2010.

[112] V. Temlyakov. Approximations of functions with bounded mixed derivative. *Trudy Matematicheskogo Instituta imeni VA Steklova*, 178:3–113, 1986.

[113] L. N. Trefethen and David Bau, III. *Numerical linear algebra*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997.

[114] J. C. Urschel. A Space-Time Multigrid Method for the Numerical Valuation of Barrier Options. *Communications in Mathematical Finance*, 2(3):1–20, 2013.

[115] F. Usta. *Sparse Grid Approximation with Gaussians*. PhD thesis, University of Leicester, June 2015.

[116] Z. Wang, Q. Tang, W. Guo, and Y. Cheng. Sparse grid discontinuous Galerkin methods for high-dimensional elliptic equations. *Journal of Computational Physics*, 314:244–263, 2016.

[117] H. Wendland. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Advances in Computational Mathematics*, 4(4):389–396, 1995.

[118] H. Wendland. *Gaussian interpolation revisited, in: Trends in Approximation Theory eds. K. Kopotun, T. Lyche, and M. Neamtu.* Vanderbilt University Press, Nashville, 2001.

[119] H. Wendland. *Scattered data approximation*. Cambridge University Press, Cambridge, 2005.

[120] H. Wendland. Multiscale analysis in Sobolev spaces on bounded domains. *Numerische Mathematik*, 116:493–517, 2010.

[121] R. E. Whaley. On the valuation of American call options on stocks with known dividends. *Journal of Financial Economics*, 9: 207–211, 1981.

[122] Z. Wu. Hermite-Birkhoff interpolation of scattered data by radial basis functions. *Journal of Approximation Theory*, 8(2), 1–11, 1992.

[123] Z. Wu. Multivariate compactly supported positive definite and compactly supported radial functions. *Advances in Computational Mathematics*, 4(4):283–292, 1995.

[124] Z. Wu. Solving pde with radial basis function and the error estimation. *Advances in computational mathematics. Lecture notes on pure and applied mathematics*, 202, 1998.

[125] D. Xiu. Efficient collocational approach for parametric uncertainty analysis. *Communications in Computational Physics*, 2(2):293–309, 2007.

[126] C. Zenger. Sparse grids. In *Parallel algorithms for partial differential equations (Kiel, 1990)*, volume 31 of *Notes Numerical Fluid Mechanics*, pages 241–251. Vieweg, Braunschweig, 1991.