UPPSALA
UNIVERSITET

High Dimensional Option Pricing Using Kernel Methods

Hampus Björklin
Kevin Coleman
Oskar Falkeström
Oscar Widenfalk

Project in Computational Science: Report

January 2023

PROJECT REPORT

Institutionen för informationsteknologi

Abstract

This study takes on the task of pricing a high dimensional basket option using a deterministic and efficient kernel method. The original method is proposed in a pre-print by Christian Rieger and Holger Wendland. The method is used to approximate high dimensional functions by their lower dimensional parts in a Reproducing Kernel Hilbert Space. The high dimensional function is solved on a set of anchor points using freezing projections. To approximate the function, the result from this set of anchor points is subsequently interpolated onto the rest of the domain. It can be shown that under certain assumptions on the smoothness on a closed Hilbert subspace, the curse of dimensionality can be circumvented with the method. This method is used for approximating the solution to the Black-Scholes PDE using a multiquadric radial basis function as kernel inside the reproducing kernel proposed by Rieger and Wendland. The method is tailored to the Black-Scholes PDE by deriving the first and second order derivatives to the reproducing kernel as well as performing a coordinate system shift in order to capture the directions of highest importance in the solution domain. The implementation result in reasonable errors at polynomial complexity instead of the exponential complexity faced by conventional methods.

Acknowledgements

We are grateful for having the opportunity to take part in this project and would like to express our gratitude to some individuals who made the study possible.

A special thank you to prof. **Elisabeth Larsson** for supervising the project and for her contributions in supplying relevant literature, input on the report and supporting us at weekly meetings.

Furthermore, we would like to thank prof. **Christian Rieger** and prof. **Holger Wendland** for their rigorous work in presenting the underlying theory that this study is based on.

Contents

1 Introduction

Options are a type of financial derivatives which give the holder the right but not the obligation to either buy - *call option* - or sell - *put option* - an underlying asset at a predetermined price at some point in the future. The underlying assets could be everything from equity, swaps, commodities, etc. The agreed upon price is called the strike price K, and the option is either exercised or not at a specified expiry time or date $t = T$. A simple example would be holding a call option with strike price $K = \$110$ and expiry one year from now on a stock currently trading at $S = \$100$. If the stock trades higher at expiry than the strike price - the price the holder possesses the right to buy it for - it is beneficial to exercise this right. The dollar value the holder would obtain from exercising the option is thus $K - S$. However, if $K \geq S$ the holder has no obligation to exercise the option and would thus abstain from doing so. The value of a call option for an underlying asset S, with strike price K and expiry time T can be described by the so-called payoff function

$$C_T = \max(S - K, 0). \tag{1}$$

The respective put option has a similar payoff function

$$P_T = \Phi(S, K) = \min(K - S, 0). \tag{2}$$

The option described above is the standard European option, but other option types exist. The American option has all the properties of its European counterpart, but while the European option is only allowed to be exercised at expiry, the holder of an American option can choose to exercise at any point prior to expiry.

The value of an option obtained when exercising is thus determined by the payoff function. However, determining the fair price of an option expiring in the future may seem like a more vague question. Luckily, there exists a partial differential equation (PDE) - The Black-Scholes equation, named after Fisher Black and Myron Scholes who in 1973 first made it public in the paper *The Pricing of Options and Corporate Liabilities* - which models this question [1]. Surprisingly, the PDE has an analytical solutions for European options under some mild assumptions. For the American option, the ability to exercise early yields a free boundary problem which one would have to resort to numerical methods such as stochastic Monte Carlo methods or deterministic finite difference schemes.

The standard Black-Scholes model discussed above is one-dimensional in the asset space, it is however generalizable to multiple dimensions in the asset space. This is useful for pricing another type of option, namely the *basket option*. The basket option is similar to the European option, but includes multiple underlying assets. For a basket option, the payoff is usually the mean price of all assets. The number of dimensions in the option is thus proportional to the number of the underlying assets. As is well-known, approximating high dimensional functions is difficult due to the so-called *curse of dimensionality*. The curse of dimensionality entails that the amount of numerical data needed to approximate a function using finite difference or finite element based grows exponentially with the number of dimensions in the approximated function. This makes finite difference schemes unviable for pricing basket options with numerous underlying assets. Monte Carlo methods is the standard approach to price such options, but they converge quite slowly and do not result in deterministic answers. This is however not as much of an issue if the uncertainty of the approximation is small enough, which can be estimated using the standard deviation of the result.

1.1 Objectives

The goal of this study is to test a method proposed by Wendland and Rieger that hopefully can be used to circumvent the curse of dimensionality when pricing basket options with a deterministic method. The goal is to reduce the complexity of solving the pricing problem to polynomial complexity rather than exponential.

2 Theory

2.1 The Black-Scholes Financial Market Model

The Black-Scholes model for the financial market consists of two assets, one risk free asset B, typically a bond, and risky asset S, e.g., the share of a company. The price dynamics of the given assets are given below.

$$\begin{aligned} dB_t &= rB_t dt, \\ dS_t &= \mu S_t dt + \sigma S_t dW_t, \end{aligned} \tag{3}$$

where r is the interest rate the risk free asset yields. The risky asset is modelled by a standard geometric Brownian motion (GBM). The change in expected price over time of an asset modeled by a GBM is deterministic and proportional to the current price of the asset and to the drift term μ. The risky asset also has a noise term modelled by a Wiener-process W_t whose magnitude is proportional to the current price of the asset and the volatility term σ [2].

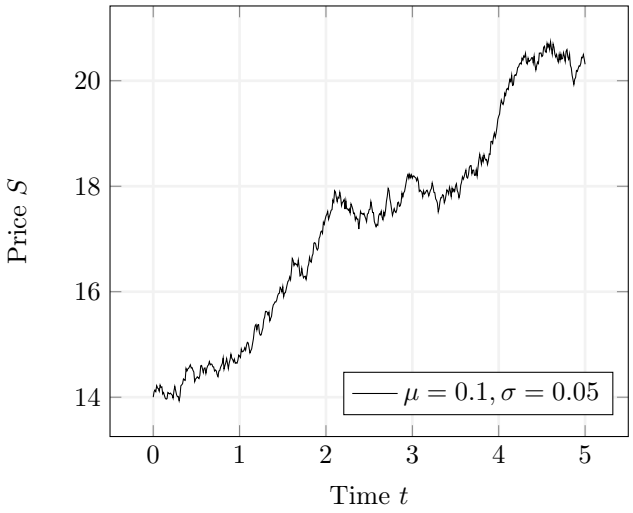


Figure 1: Example of geometric brownian motion with drift $\mu = 0.1$ and volatility $\sigma = 0.05$

2.2 The Feynman-Kač Representation Formula

Given three scalar functions $\mu(t,x)$, $\sigma(t,x)$ and $\Phi(x)$, the following Cauchy problem can be constructed.
Find a function F which on $[0,T] \times R$ satisfies the boundary value problem:

$$\begin{aligned} \frac{\partial F}{\partial t}(t,x) + \mu(t,x)\frac{\partial F}{\partial x} + \frac{1}{2}\sigma^2 \frac{\partial^2 F}{\partial x^2}(t,x) = 0, \\ F(T,x) = \Phi(x), \end{aligned} \tag{4}$$

where $\Phi(x)$ is the payoff function and T is the expiry time. Assuming a solution to equation (4) exists, an alternative approach to the purely analytical solution of the problem is to construct a stochastic representation formula. This is done by fixing a point in space x and time t and defining the stochastic process X on the time interval $[t,T]$ as the solution to the stochastic differential equation.

$$\begin{aligned} dX_s = \mu(x, X_s)ds + \sigma(s, X_s)dW_s, \\ X_t = x. \end{aligned} \tag{5}$$

The infinitesimal operator $\mathcal{A}$ for the stochastic process is given by the Îto-formula

$$\mathcal{A} = \mu(t,x)\frac{\partial}{\partial x} + \frac{1}{2}\sigma^2(t,x)\frac{\partial^2}{\partial x^2}. \tag{6}$$

as the operator $\mathcal{A}$ appears in the original PDE, the boundary value problem can be rewritten in the form

$$\begin{aligned} \frac{\partial F}{\partial t}(t,x) + \mathcal{A}F(t,x) = 0, \\ F(T,x) = \Phi(x). \end{aligned} \tag{7}$$

Applying the Îto-formula once again to the process $F(s, X(s))$

$$F(T,x) = F(t,X_t) + \sum_t^T \frac{\partial F}{\partial t}(s,X_s) + \mathcal{A}F(s,X_s)ds + \sum_t^T \sigma(s,X_s)\frac{\partial F}{\partial x}(s,X_s)dW_s. \tag{8}$$

By assumption, from equation (7), the time integral vanishes, as does the stochastic integral if the expected values are studied. The boundary condition $F(T,x) = \Phi(x)$ together with the initial condition $X_t = x$ yields the recognized solution known as the Feynman-Kač representation formula, cf., [2].

$$F(t,x) = \mathrm{E}_{t,x}[\Phi(X_T)]. \tag{9}$$

The closely related problem

$$\begin{aligned} \frac{\partial F}{\partial t}(t,x) + \mu(t,x)\frac{\partial F}{\partial x} + \frac{1}{2}\sigma^2\frac{\partial^2 F}{\partial x^2}(t,x) - rF(t,x) = 0, \\ F(T,x) = \Phi(x), \end{aligned} \tag{10}$$

can be solved by applying the same approach, yielding the useful solution

$$F(t,x) = e^{-r(T-t)}\mathrm{E}_{t,x}[\Phi(X_T)]. \tag{11}$$

2.3 Risk-Neutral Valuation and the Black-Scholes Equation

Assuming a financial derivative contains the risky asset S as a underlying asset, a reasonable approach to determine a fair price of the derivative is to discount the expected payoff of the derivative. Less intuitive but nonetheless important is not to use an objective probability measure but instead a risk neutral measure Q. Suppose that the agents trading at the market are risk-neutral, then all assets will have an expected rate of return of r - the rate the risk free asset yields. Considering the Black-Scholes model, this implies a drift term μ equal to r.
A risk-neutral valuation of a financial derivative with payoff function $\Phi(s)$ is thus

$$u(t,s) = e^{-r(T-t)}\mathrm{E}^Q_{t,x}[\Phi(S_T)], \tag{12}$$

where under the risk neutral measure Q the dynamics of S_t is given by

$$dS_u = \mu S_u dt + \sigma S_u dW_u \tag{13}$$

$$dS_t = s. \tag{14}$$

The risk-neutral valuation shown in equation (12) takes the form of the Feynman-Kač representation formula, which in turn is the solution to the boundary value problem described in equation (10). The risk-neutral valuation of the considered financial derivative can thus also be solved from the BVP

$$\begin{aligned} \frac{\partial F}{\partial t}(t,s) + r\frac{\partial F}{\partial s}(t,s) + \frac{1}{2}\sigma^2\frac{\partial^2 F}{\partial s^2}(t,s) - rF(t,s) = 0, \\ F(T,s) = \Phi(s), \end{aligned} \tag{15}$$

which is famously known as the Black-Scholes Equation.

2.4 Solutions to the Black-Scholes Equation

In their paper published in 1973, Fisher Black and Myron Scholes shows that an analytical solution exists to the Black-Sholes equation (15) under the following assumptions:

- interest rate r and volatility σ are constant,
- the distribution of of possible stock prices at the end of any finite time interval is log-normal,
- the stock pays no dividend and trading it involves no transaction fees,
- it is possible to borrow any fraction of the price of a security to buy it or to hold in, at interest rate r,
- there are no penalties for short selling,
- the option is of European type.

The price of a call option with strike price K and time to maturity $\tau := (T - t)$ is given by

$$u(s,t) = sN(d_1) - e^{-r\tau}KN(d_2), \tag{16}$$

$$d_1 = \frac{ln\frac{s}{K} + (r + \frac{\sigma^2}{2})\tau}{\sigma\sqrt{\tau}}, \tag{17}$$

$$d_2 = \frac{ln\frac{s}{K} + (r - \frac{\sigma^2}{2})\tau}{\sigma\sqrt{\tau}}, \tag{18}$$

where N is the standard normal cumulative distribution function [1].
Even though analytical solutions are preferable, the assumption of the option being of European type and that the underlying has constant volatility is often too restrictive for many real world applications. If the underlying asset is modelled by a process where the volatility is variable, or even non-deterministic such as in the SABR model or Heston model, numerical methods are required. Similarly, if the option is of American type - which accounts for a considerable fraction of all traded options - or of most other exotic types of options such as the basket option, analytical solutions do not exist either.

2.4.1 Numerical Methods for Option Pricing

Two common approaches to pricing options numerically are the Monte Carlo methods and the finite difference-based methods. Recalling the Feynman-Kač stochastic representation from equation (12), the price of an option can be determined by

$$u(t,s) = e^{-r(T-t)}\mathrm{E}^Q_{t,x}[\Phi(S_T)]. \tag{19}$$

This formula has a natural Monte Carlo-based interpretation. By simulating numerous paths of the underlying assets price, given its dynamics S_t, the mean of the payoff function $\Phi(S_T)$ for each simulated path gives an approximation of the expectation term in the formula. Discounting this to present time gives an approximation of the current value of the option.

Another approach to determine the fair price of an option is to solve the initial-boundary value problem given by the Black-Scholes equation shown in equation (15). This problem is solved on a mesh constructed with discrete points representing the value of the option for different prices of the underlying and at different times using finite difference based schemes. It is possible to use the payoff function at all discrete points of stock price s at maturity T as the given final condition. Due to arbitrage reasons, the value of the option will always be zero if the price of the underlying asset at some point is zero, giving another boundary condition at $s = 0$. Finally, utilizing the fact that if the price of the asset is significantly larger than the strike price, there is a large probability

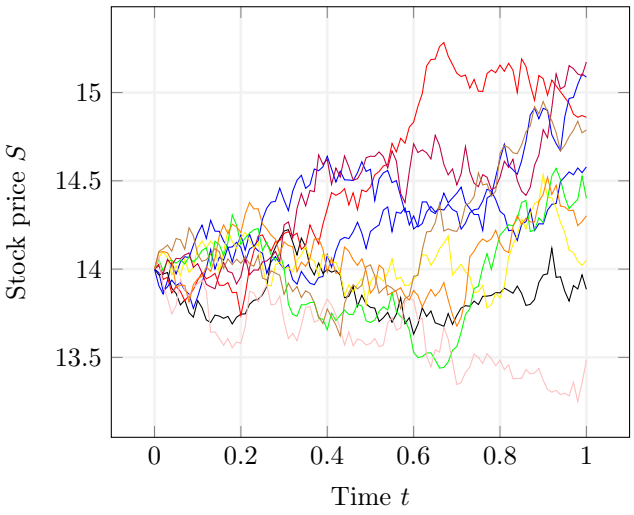


Figure 2: Example of ten simulated stock paths with drift $= \mu = r = 0.05$ and volatility $\sigma = 0.05$

at maturity of the underlying asset being *in the money*, i.e., higher than the strike price. This also truncates the otherwise infinite domain in price at S_{max}. The low probability of the option expiring worthless thus having a negligible negative contribution to the option value then given by $S_{max} - e^{-r(T-t)}K$. To price a call option, the following PDE has to be solved.

$$\begin{aligned} \frac{\partial u}{\partial t} = rs\frac{\partial u}{\partial s} + \frac{1}{2}\sigma^2 s^2 \frac{\partial^2 u}{\partial t^2} - ru, \\ t \in [0, T], \\ s \in [0, S_{max}], \end{aligned} \tag{20}$$

with boundary conditions

$$u(0,t) = 0, \quad u(S_{max}, t) = S_{max} - e^{-r(T-t)}K, \quad 0 \leq t \leq T, \tag{21}$$

and final condition

$$u(s,T) = \max(S - K, 0), \quad 0 \leq s \leq S_{max}. \tag{22}$$

2.4.2 Basket options and High-Dimensional Problems

The Black-Scholes equation can be generalized to model multiple dimensional problems. Such problems arise when pricing basket options, i.e. options with multiple underlying assets. The assets could be everything from a pair of stocks to all components of an index. The price of a basket option u is given by the PDE below.

$$\begin{aligned} &\frac{\partial u}{\partial t} + \frac{1}{2}\sum_{i,j}^{d} \mathbf{C}_{i,j} s_i s_j \frac{\partial^2 u}{\partial s_i \partial s_j} + \sum_{i}^{d} r s_i \frac{\partial u}{\partial s_i} - ru = 0, \\ &u(\mathbf{s},t)|_{\|\mathbf{s}\| \geq s_{max}} = \phi(\mathbf{s},t) = \max\left(\frac{1}{d}\sum_{i}^{d} s_i - Ke^{T-t}, 0\right), \\ &u(\mathbf{s},t)|_{\|\mathbf{s}\|=0} = 0, \\ &u(\mathbf{s},t)|_{t=T} = \phi(\mathbf{s},T), \end{aligned} \tag{23}$$

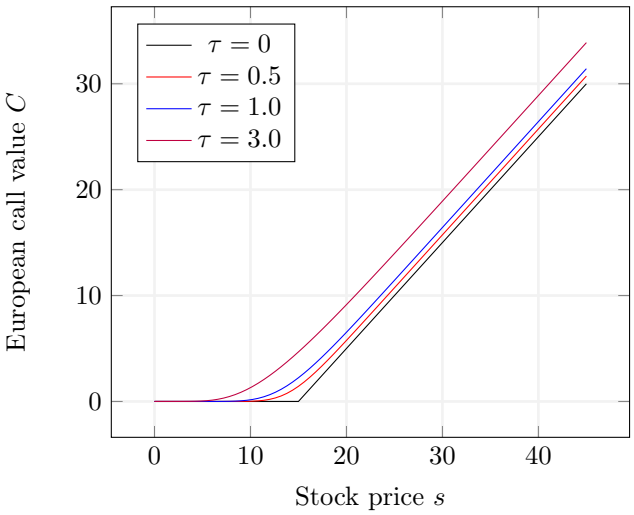


Figure 3: Fair price of European call determined with finite difference method at different times to expiry τ

where $\mathbf{C}$ is the volatility- and correlation matrix of the underlying asset prices $\mathbf{s}$. For the standard one-dimensional case, using a finite difference method is often the preferable approach for pricing options as depending on the choice of scheme because comparably high convergence rates of the discretization error can be achieved. For standard Monte Carlo methods, square root convergence is expected.

It is also of interest to study how the problem complexity of pricing an option grows with increased number of dimensions. Here Monte Carlo methods often excel as the complexity scales linearly in dimensions. For grid based methods such as finite difference methods, the accuracy of an approximation of a function is strongly related to the density of discrete points used for evaluation, i.e. how fine the grid is. With each added dimension, the number of discrete points needed to retain similar density grows exponentially, and with exponentially growing evaluation points, the computationally complexity also grows exponentially.

This phenomenon is known as the curse of dimensionality and renders finite difference methods impractical to use for high-dimensional options. A currently unconventional method to deterministically determine the fair price of an option, which also circumvents the curse of dimensionality is to use radial basis function approximation.

2.5 Radial Basis Functions

Solving the Black-Scholes equation for a basket option without any analytical solution requires some form of numerical approximation. Radial basis function methods are used to find such an approximation s to a multivariate function f. As implied by the name, these methods use radial basis functions (RBFs). The name RBF comes from the fact that these basis functions are radially symmetric [3].

RBF methods are mesh-free, meaning that the points in the domain are not required to lie on a structured grid or mesh [4]. Geometrically, these approximation methods only require distances between point pairs because of the inherent radial symmetry of RBFs [5]. This also makes RBFs suitable for approximating high-dimensional functions, as distances are easy to compute and independent of dimension [3]. In addition to this, RBF methods are often spectrally accurate and stable for large numbers of nodes in high dimensions [4].

An RBF is a real-valued function φ with values depending on the distance between the input $\mathbf{x}$

and some fixed point $\mathbf{x_i}$, such that $\varphi(\mathbf{r}) = \varphi(\|\mathbf{x}-\mathbf{x}_i\|)$. The distance metric $\mathbf{r} = \|\cdot\|$ is measured on a vector space $V \to [0, \infty)$ and is usually the *Euclidian Distance*. Because of radial symmetry, i.e., that the function value of an RBF depends only on the distance from some fixed point, rotations do not affect the value of the function [5]. The function $\varphi(\mathbf{r}) = \varphi(\|\mathbf{x}-\mathbf{x}_i\|)$ is also called the radial kernel centred at point $\mathbf{x}_i$. A set of radial kernels are also RBFs for any set of nodes $\{\mathbf{x}_k\}_{k=1}^n$ if the following criteria are met.

1. The kernels $\varphi_{x_1}, \varphi_{x_2}, ..., \varphi_{x_n}$ are linearly independent.

2. The kernels $\varphi_{x_1}, \varphi_{x_2}, ..., \varphi_{x_n}$ form a basis for the so-called *Haar Space*. Such a space means that the so-called interpolation matrix A is non-singular [6]. The interpolation matrix A is presented in equation 27 later in this section.

The input parameter to an RBF is usually scaled with a so-called *shape parameter* ϵ [6]. Because of this shape parameter, RBFs have different shapes and thus different interpolation surfaces between points. One form of RBF are the infinitely smooth RBFs $\varphi(\mathbf{x}) \in C^\infty(\mathcal{R})$, meaning that they have an infinite amount of continuous derivatives [5]. Some of these RBFs are also positive definite. Two examples of infinitely smooth radial basis functions with tuneable shape parameters are:

$$\text{Gaussian} : \varphi(\mathbf{r}) = e^{-(\epsilon \cdot \mathbf{r})^2}. \tag{24}$$

$$\text{Multiquadric} : \varphi(\mathbf{r}) = \sqrt{1 + (\epsilon \cdot \mathbf{r})^2}. \tag{25}$$

In RBF methods the approximation s to a function f is usually given by finite linear combinations of RBFs. The RBFs are then used to approximate f by interpolating between all evaluated points. To use the RBF method, we define the linear space S where $s \in S$ is the approximation to the function f. Then, we let f be the function evaluated for the data points $x \in \mathcal{R}^n$ in n dimensions. These points x are a part of the discrete set $\Omega \subseteq C(\mathcal{R}^n)$ where the function f will be evaluated. Interpolation explicitly requires that all function values for f equal the function values for s, i.e $s(x) = f(x)$ for all x in Ω. The interpolation is subsequently performed in between these discrete points x [5].

Radial basis functions define a space $S \subseteq C(\mathcal{R}^n)$ which depends on the finite set Ω containing center points x_i. Generally, RBF methods can approximate a function f by a linear combination of radial RBFs

$$s(\mathbf{x}) = \sum_{i=1}^{N} \lambda_i \varphi(||\mathbf{x} - \mathbf{x}_i||), \tag{26}$$

where λ_i are real coefficients, N are the number of center points $x_i \in \Omega$, and φ is an RBF. We then choose λ_i for $i = 0, 1, ..., N$ such that $s(x_i) = f(x_i) \quad \forall i$. Performing this, s will interpolate f at the chosen points $\mathbf{x}$. By solving the linear system

$$A\lambda = f, \tag{27}$$

the coefficients λ_i can be calculated. Here, the *interpolation matrix* A contains elements $a_{i,j} = \varphi(||x_i - x_j||)$ for $i, j = 0, 1, ..., N$. The interpolation matrix A is invertible for the Gaussian and multiquadric RBFs presented above. According to the *uncertainty principle*, using an RBF approximation leads to a trade-off between a well conditioned matrix A and achieving a high accuracy. Generally, choosing a small shape parameter ϵ results in a high accuracy, but this choice is usually associated with an ill-conditioned interpolation matrix. Choosing the optimal shape parameter to obtain both an accurate solution and obtaining numerical stability is a highly researched topic [7].

Furthermore, RBF methods have in recent times emerged as a good method for solving and numerically approximating partial differential equations (PDEs) [7]. The function values at points $\mathbf{x}$ in the domain are approximated using the linear combination of RBFs in equation (26). The derivatives can subsequently be approximated by a linear combination of the derivatives of the radial kernels

$$\frac{\partial^n u(\mathbf{x})}{\partial x^n} = \sum_{i=1}^{N} \lambda_i \frac{\partial^n}{\partial x^n} \varphi(||\mathbf{x} - \mathbf{x}_i||), \quad \mathbf{x} \in \mathcal{R}^d, \tag{28}$$

where N is the number of points in the discretized domain, d is the dimension of the domain and λ_i are scalar coefficients unaltered by the linear differential operator.

2.6 Reproducing Kernel Hilbert Spaces

The following section considers Hilbert spaces only over a set of real numbers $\mathcal{R}$. Let X be an arbitrary set of points and $F(X, \mathcal{R})$ be a set of functions from X to $\mathcal{R}$. The set $F(X, \mathcal{R})$ is a vector space with operations of addition $(f + g)(x) = f(x) + g(x)$ and scalar multiplication $(\lambda \cdot f)(x) = \lambda \cdot (f(x))$. A subset $\mathcal{H} \subseteq F(X, \mathcal{R})$ is a *Reproducing Kernel Hilbert Space* (RKHS) on X if the following holds.

(i) $\mathcal{H}$ is a vector subspace of $F(X, \mathcal{R})$.

(ii) $\mathcal{H}$ has an inner product $\langle \cdot \rangle$.

(iii) For every $x \in X$, the evaluation functional $L_x : \mathcal{H} \to \mathcal{R}$ is defined by:

$$L_x(f) = f(x), \quad \text{for all} \quad f \in \mathcal{H}. \tag{29}$$

The linear evaluation functional L_x can thus evaluate each function in $\mathcal{H}$ at a point x [8]. L_x also has to be a bounded operator, that is, there exists some constant $M > 0$ such that

$$|L_x(f)| := |f(x)| \leq M_x ||f||_H, \quad \text{for all} \quad f \in H. \tag{30}$$

Equation (30) presents the weakest condition that ensures the existence of both an inner product and the evaluation of every function $\mathcal{H}$ at every point on the domain [8]. The Riesz representation theorem guarantees that in an RKHS, the linear evaluation functional $L_x \in H$ is given by taking the inner product of f with a function K_x. Thus, for each $x \in X$, there exists a unique, so called *reproducing kernel* $K_x \in \mathcal{H}$ for the point x, such that:

$$f(x) = L_x(f) = \langle f, K_x \rangle, \quad \text{for all} \quad f \in \mathcal{H}. \tag{31}$$

Thus, for each point in space, there exists a reproducing kernel that allows for functions to be reconstructed from their evaluations at that point. This reproducing kernel is positive semi-definite because for any set $X = \{x_1, ...x_N\} \subseteq \Omega$ of points, the matrices $K(x_i, x_j) \in \mathcal{R}^{NxN}$ are always positive semi-definite. This holds true if evaluations of points over $\mathcal{H}$ are linearly independent [9]. As will be presented in the upcoming sections, the properties of a RKHS can be useful for high-order function approximation.

2.7 Approximating High-Dimensional Functions and the Curse of Dimensionality

As previously mentioned, one consequence and immense challenge when numerically approximating high-dimensional functions is the curse of dimensionality. This numerical dilemma entails that the number of data points needed to adequately approximate a function grows exponentially with the number of dimensions d. Fortunately, a new method presented by Rieger and Wendland allows for high-dimensional functions to be approximated as finite sums of lower dimensional functions $f(x_1), ..., f(x_n)$ that do not suffer from the curse of dimensionality. This method uses reproducing kernels in an RKHS to reconstruct functions of higher dimensions using only lower dimensional basis functions.

This method stems from using sampling inequalities to show that the curse of dimensionality does not apply under certain assumptions for functions from Sobolev spaces $H^{\sigma}_{\Lambda}(\Omega)$ and mixed regularity Sobolev spaces $H^{\sigma}_{mix,\Lambda}(\Omega)$, where Ω is the computational domain with points in the region $[0,1]^d$. Functions in a Sobolev space are required to have weak derivatives up to a certain order, which is here presented by a smoothness parameter σ. A higher σ means a higher smoothness and higher numbers of weak derivatives are required. In a mixed-regularity Sobolev space, the functions are allowed to have varying regularity and smoothness in different directions. For such spaces under some assumptions that will be presented in later sections, the cost of approximating a normed function up to an accuracy $\epsilon > 0$ depends only polynomially on the space dimension d. This result is far superior to the exponential increase of other conventional methods [9].

2.7.1 Anchored Decomposition of High-Dimensional Functions

The method proposed by Rieger and Wendland uses a decomposition called *High-dimensional model representation*, where a function $f : \mathcal{R}^d \to \mathcal{R}$ can be decomposed into a sum of functions that depend on fewer variables

$$\begin{aligned} f(x_1, ..., x_d) = f_0 + f_1(x_1) + f_2(x_2) + ... + f_d(x_d) + f_{1,2}(x_1, x_2) + \\ f_{1,3}(x_1, x_3) + ... + f_{d-1,d}(x_{d-1}, x_d). \end{aligned} \tag{32}$$

The decomposition of a function f into lower dimensional parts can be represented in terms of sets containing the dimensions of interest. First, we define a set of all dimensions up to d, $\mathcal{D} = 1, ...d$. Then we define a subset $u \subseteq \mathcal{D}$ and $P(d) = \{u : u \subseteq \mathcal{D}\}$ as the set of all subsets of $\mathcal{D}$. $\Lambda \subseteq P(d)$ is consequently defined as a fixed set of subsets of $\mathcal{D}$. Then a function $f : \Omega \to \mathcal{R}$ can be decomposed into

$$f = \sum_{u \in \Lambda} f_u, \tag{33}$$

where f_u only depends on variables with indices in u. If f can be decomposed as in equation (33), then f is said to have a Λ-*representation*.

High-dimensional functions can be described by this decomposition as long as the set of subsets Λ is bounded reasonably. This can be achieved by defining the set of subsets Λ according to a max order m, which determines the maximum amount of dimensions allowed in a subset u. In other words, m determines then number of elements in each subset u. Thus, the max order m determines how many dimensions a decomposition of f is allowed to depend on. The set Λ can be downwards closed and bounded in two different ways as presented by Rieger and Wendland. However, we chose to concentrate on bounding Λ by

$$\Lambda = \{u \subseteq \mathcal{D} : \#u \leq m\}, \tag{34}$$

where $m \leq d$ is the mentioned max order which decides how many dimensions are allowed in each subset $u \in \Lambda$. Consequently, all functions f can be written as sums of functions that depend only on a maximum of m dimensions. For Λ given as in equation (34) the decomposition of f can be constructed as

$$f = \sum_{u \subseteq \mathcal{D}, \#u \leq m} f_u. \tag{35}$$

The function f can now theoretically be decomposed into functions of reduced dimensions according to equation (35). This concept of dimensionality reduction can be applied to any type of projection. We will put these concepts into practice by using the so-called *anchored decomposition*. In anchored decomposition, the function f is approximated by solving for the lower dimensional functions on a specific set of points, called *anchor points*. This decomposition method utilizes the projection known as *freezing*. If $\mathbf{a} \in \Omega$ is a fixed set of points $\mathbf{a} = (a_1, ..., a_d)$, then the freezing projection at point $a_{i,j}$ is defined by

$$P_j^{fr} f(x_1, ...x_d) = f(x_1, ..., x_{j-1}, a_j, x_{j+1}, ..., x_d), \quad \mathbf{x} \in \Omega. \tag{36}$$

Here the j:th dimension is frozen and kept unchanged. In other words, using freezing, projects the function onto the *anchored space* at a certain anchor $\mathbf{a}$. The idea is that the decomposed and lower dimensional functions of f, can then be solved for this anchored space. The combined set of all anchor points is called the *anchored set*, and the values of the lower dimensional functions are calculated for this set of anchor points.

To illustrate how to define an anchored set, let the Hilbert space $H_\Lambda^\sigma(\Omega)$ be a set of all functions such that $f \in H_\Lambda^\sigma(\Omega)$ has an Λ-representation for a downwards closed $\Lambda \subseteq P(d)$. For each $u \in \Lambda$, a set of points should be chosen corresponding to points in the dimensions included in u

$$X_u = \left(\tilde{x}_1^u, ..., \tilde{x}_n^u \right) \subseteq \Omega, \tag{37}$$

where n is the amount of anchor points in dimension u. For an anchor $\mathbf{a} \in \Omega$, the set above can subsequently be extended to an anchored set. This is the following set of points for every $u \in \Lambda$

$$X_u = \left(x_1^u, ..., x_n^u \right) \subseteq \Omega. \tag{38}$$

In this anchored set, each point x_j^u is defined as

$$e_k^T x_j^u = \begin{cases} e_k^T \tilde{x}_j^u, & \text{if } k \in u \\ e_k^T \mathbf{a}, & \text{if } k \notin u \end{cases}, \tag{39}$$

where e_k is the k-th unit vector in $\mathcal{R}^d$. Thus, if a point corresponding to a dimension is not included in the subset u, this point will be considered an anchor $\mathbf{a}$. Subsequently, that dimension will be frozen and unchanged [9].

2.7.2 Circumventing the Curse of Dimensionality

From the *Sobolev embedding theorem*, Rieger and Wendland deduce that for a smoothness order $\sigma > \frac{d}{2}$, the Sobolev space $H^\sigma(\Omega)$ is in fact a reproducing kernel Hilbert space (RKHS). As before, Ω is the computational domain with points in $[0, 1]^d$. Subsequently, a high dimensional function in this Sobolev space with large enough order of smoothness σ, can be reconstructed by its reproducing kernel as shown in equation (31). This reproducing kernel $K_\Lambda(\cdot, \mathbf{x})$ works

for dimensions in the set Λ defined in equation (34). This reproducing kernel can however only be used for a closed subspace of the Hilbert space $H^\sigma(\Omega)$, called $H^\sigma_m(\Omega)$. $H^\sigma_m(\Omega)$ is defined for $1 \leq m \leq d$ and is the set consisting of all functions $f \in H^\sigma(\Omega)$ of max order m [9]. Furthermore, for smoothness $\sigma > \frac{d}{2}$, the positive definite reproducing kernel $K_\Lambda(\cdot, \mathbf{x}) : \Omega \times \Omega \to \mathcal{R}$ of the downward closed set $\Lambda \subseteq P(d)$ is defined by

$$K_\Lambda(\cdot, \mathbf{x}) = P^{pr} K(\cdot, \mathbf{x}), \quad \mathbf{x} \in \Omega, \tag{40}$$

where $P^{pr} : H^\sigma(\Omega) \to H^\sigma_\Lambda(\Omega)$ is the orthogonal projection. Using the RKHS and bounding the number of points in a set $X_N = \{\mathbf{x}_1, ...\mathbf{x}_N\}$, Rieger and Wendland show that the curse of dimensionality does not apply to the closed subspace $H^\sigma_m(\Omega)$. To then approximate the function f, an interpolant s_f can be computed according to

$$s_f = \sum_{j=1}^{N} \lambda_j K_\Lambda(\cdot, \mathbf{x}_j). \tag{41}$$

The subsequent linear system can be solved to determine the λ-coefficients

$$\mathbf{A}\lambda = f|X_N, \tag{42}$$

where $\mathbf{A} := K_\Lambda(\mathbf{x}, \mathbf{y})$ is a symmetric and positive definite $N \times N$ matrix. The evaluated points $\mathbf{x}, \mathbf{y}$ are in X_N. Numerically, this interpolant can be calculated in $\mathcal{O}(N^3)$ computational complexity. The interpolant s_f satisfies the following stability condition [9]

$$||f - s_f||_{H^\sigma(\Omega)} \leq ||f||_{H^\sigma(\Omega)}, \quad f \in H^\sigma_n(\Omega). \tag{43}$$

From equation (43), Rieger and Wendland derive an error estimate using sampling inequalities, seen in their *Theorem 4.9*. With some further reconstructions and looking at the limit of $d \to \infty$, they conclude that the error between the function and the interpolant that is computed with the reproducing kernel only increases with dimension d polynomially. For comparison, the number of node points needed in finite difference methods increases exponentially with d resulting in huge computational complexity. Thus, for the closed subspace $H^\sigma_m(\Omega)$, of Hilbert space $H^\sigma(\Omega)$, with smoothness $\sigma > \frac{d}{2}$ and a fixed max order $m < d$, the curse of dimensionality is circumvented. This was also shown to be true by Wendland and Rieger (*Corollary 5.9*) for a closed subspace of a mixed Sobolev space $H^\sigma_{\text{mix},m}$, which is the set of all m-order functions $f \in H^\sigma_{\text{mix}}$ [9].

2.7.3 Application of Anchored Sobolev Spaces

To put the method in practice, Rieger and Wendland provide an application of their method on an anchored Sobolev space, solved in the region Ω with points in $[0, 1]$. Firstly, the specific set of dimensional subset Λ can be defined from equation (34).

$$\Lambda = \{\emptyset, \{1\}, \ldots, \{d\}, \{1, 2\}, \ldots \{d-1, d\}, \ldots, \}\,. \tag{44}$$

This set thus contains reduced dimensional subsets. The maximum size of each subset is decided by the max order m. As explained in Section 2.7.2, for smoothness $\sigma > \frac{d}{2}$, the Sobolev space $H^\sigma(\Omega)$ is a RKHS. Thus, the high-dimensional function f can be approximated by its evaluations at specific points by the reproducing kernel $K_\Lambda(x, y)$ on the closed subspace $H^\sigma_m(\Omega)$. The subsequent reproducing Kernel K_Λ proposed by Rieger and Wendland for the RKHS denoted $H^m_{\text{mix},\Lambda}$ is given by

$$K_\Lambda(\mathbf{x},\mathbf{y}) = \sum_{u\in\Lambda} K_u(x,y) = \sum_{u\in\Lambda}\prod_{j\in u} k(x_j,y_j), \quad x,y\in[0,1]^d, \tag{45}$$

for a certain dimension d, and max order of dimension m. $k(x_j, y_j)$ is a kernel function. In words, the reproducing kernel is the sum of all kernel functions over the subsets of dimensions u present in the set Λ. Here, the product with the empty set is defined to equal 1. For each subset u, the kernels $k(x_j, y_j)$ for all $j \in u$ are multiplied [9].

The next step is to define which points should be evaluated by the reproducing Kernel to reconstruct function f. As mentioned in Section 2.7.2, the function is reconstructed by evaluating the reproducing kernel for the point set X_N. Here, this point set will be the anchored set, as defined in equations (38) and (39). The approximation is thus first evaluated for an anchored set, and then interpolated to the rest of the domain. Using the defined reproducing Kernel and point set above, the interpolant s_f in equation (41) can be found by evaluating the λ-coefficients in equation (42). The implementation of this approach will be presented in Section 3 for three dimensions and also extended to work for d-dimensional functions.

3 Method

The following section gives an overview of the methodology used. The implementation is done in MATLAB.

3.1 Higher Dimensional Reproducing Kernel

In Section 2.7.3, an application of Rieger and Wendland's method of dimensionality reduction is presented. In order to diminish the curse of dimensionality when pricing a high dimensional basket option with the Black Scholes PDE, we chose to implement this method. Thus, the reproducing Kernel in equation (45) is used for a set of reduced dimensions defined in equation (44).

We first test the method in two dimensions, using the same kernel $k(x, y)$ (see equation (6.1) in [9]) and test function used by Rieger and Wendland. We reproduce the results presented by the authors in *Fig 1.* in their study [9]. We chose to implement the method using the multiquadric kernel function instead, as presented in equation (25). One reason for this choice is the fact that the multiquadric RBF is easily differentiable, which is beneficial for approximating the Black-Scholes solution. This is important because the approximation requires taking first and second order derivatives of the reproducing kernel. The multiquadric RBF has also been shown to be efficient for pricing European basket options in particular. Since these are infinitely smooth and less sensitive to the choice of shape parameter ϵ compared to other alternatives [10]. In equation (22), it is clear that the Black-Scholes equation for pricing a put option has a discontinuity in its final condition. The smoothness of the multiquadric RBF makes it more suitable for approximating smooth functions, which is why it performs well for the continuous parts of the solution. The discontinuity in the final condition leads to oscillatory behavior in the approximation. This downside can however be reduced by choosing a larger shape parameter ϵ, resulting in a flatter RBF function.

To make sure the multiquadric RBF is also a suitable choice of kernel in practice, we use the method to approximate the same test function from Rieger and Wendlands 2D example in their report. However, instead of using their proposed kernel function, we use the multiquadric RBF and compare the results. With the multiquadric RBF, the approximation error decreases with an increasing number of points used. However, the convergence rate with the multiquadric RBF kernel is lower compared to that presented in *Fig 1.* in Rieger and Wendlands study. Despite this, we conclude that the multiquadric RBF is an appropriate choice of kernel for testing the method for higher order functions as well. We base this on results from previous studies approximating European basket options [10] and a promising convergence rate when approximating the simpler test function using the proposed method. During the rest of the study we thus test the method using a multiquadric RBF as kernel function.

3.1.1 Rieger and Wendlands Method for a Multiquadric RBF in 3D

Rieger and Wendland implemented the method on a 2D-problem [9]. We extend this to a 3D example for the multiquadric kernel. A 3D example of how we implement Rieger and Wendlands method using a multiquadric RBF is presented below. In this case we choose $d = 3$ and max order $m = 2$. In accordance with equation (44), Λ becomes

$$\Lambda = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}\}, \tag{46}$$

where we exclude the subset $\{1,2,3\}$ because it contains 3 dimensions which exceeds the max order of $m = 2$. As the max order is $m = 2$, two three dimensional point sets $\mathbf{x}$ and $\mathbf{y}$ are used for each evaluation of the reproducing kernel. The sets of points are denoted $\mathbf{x} = (x_1, x_2, x_3)$ and $\mathbf{y} = (y_1, y_2, y_3)$. Following equation (45) we derive the reproducing kernel

$$\begin{aligned} K_\Lambda(\mathbf{x},\mathbf{y}) = 1 &+ k(x_1,y_1) + k(x_2,y_2) + k(x_3,y_3) \\ &+ k(x_1,y_1)\cdot k(x_2,y_2) + k(x_1,y_1)\cdot k(x_3,y_3) \\ &+ k(x_2,y_2)\cdot k(x_3,y_3), \end{aligned} \tag{47}$$

where the term containing only 1 stems from the inclusion of the empty set in Λ. The multiquadric kernel basis function with shape parameter ϵ as seen in equation (25) is simplified to

$$k(x,y) = \sqrt{1 + (||\epsilon(x-y)||)^2} = \{\text{1D points}\} = \sqrt{1+\epsilon^2(x-y)^2},$$

because we compare only points of one dimension. This simplifies the Euclidean distance measure $||\cdot||$. Substituting the kernel function above into the reproducing kernel given in equation (47), results in the three dimensional reproducing kernel

$$\begin{aligned} K_\Lambda(\mathbf{x},\mathbf{y}) = 1 &+ \sqrt{1+\epsilon^2(x_1-y_1)^2} + \sqrt{1+\epsilon^2(x_2-y_2)^2} \\ &+ \sqrt{1+\epsilon^2(x_3-y_3)^2} + \sqrt{1+\epsilon^2(x_1-y_1)^2}\cdot\sqrt{1+\epsilon^2(x_2-y_2)^2} \\ &+ \sqrt{1+\epsilon^2(x_1-y_1)^2}\cdot\sqrt{1+\epsilon^2(x_3-y_3)^2} \\ &+ \sqrt{1+\epsilon^2(x_2-y_2)^2}\cdot\sqrt{1+\epsilon^2(x_3-y_3)^2}. \end{aligned} \tag{48}$$

As the goal of the study is to circumvent the curse of dimensionality by approximating the Black-Scholes equation applied on a multi-dimensional basket option, the reproducing kernel must also be used to approximate the derivatives in the PDE. Thus, derivatives of 1st and 2nd order need to be derived for the reproducing Kernel, as these operators are part of the Black-Scholes equation, see equation (23). For dimensions $i, j, k \in u$, the first order derivative up to three dimensions (thus for m=1,2 or 3) is derived as

$$\frac{\partial}{\partial x_i}(K_\Lambda(\mathbf{x},\mathbf{y})) = \frac{\epsilon^2(x_i-y_i)}{\sqrt{1+\epsilon^2(x_i-y_i)^2}}\left(1 + \sqrt{1+\epsilon^2(x_j-y_j)^2} + \sqrt{1+\epsilon^2(x_k-y_k)^2}\right), \tag{49}$$

where the reproducing kernel is differentiated with respect to dimension i. In a similar fashion, the second order derivative up to three dimensions (for m=1,2 or 3) is given by

$$\frac{\partial^2}{\partial x_i^2}(K_\Lambda(\mathbf{x},\mathbf{y})) = \frac{\epsilon^2}{(1+\epsilon^2(x_i-y_i)^2)^{\frac{3}{2}}}\cdot\left(1 + \sqrt{1+\epsilon^2(x_j-y_j)^2} + \sqrt{1+\epsilon^2(x_k-y_k)^2}\right). \tag{50}$$

Unsurprisingly, the derivatives of a certain dimension i, include only factors with points corresponding to that dimension. For example, in the case $m = 2$, the terms involving the third dimension $k \in u$ are zero, as more than two dimensions per subset $u \in \Lambda$ are not allowed. This can be seen in equation (48), where the maximum amount of factors in a term is 2, coming from the max order $m = 2$. Consequently, the mixed derivatives of the reproducing kernel up to three dimensions is

$$\frac{\partial^2}{\partial x_i \partial x_j}(K_\Lambda(\mathbf{x}, \mathbf{y})) = \frac{\epsilon^4 (x_i - y_j)(x_j - y_j)}{\sqrt{1 + \epsilon^2 (x_i - y_i)^2} \cdot \sqrt{1 + \epsilon^2 (x_j - y_j)^2}}, \tag{51}$$

where the derivatives are taken with respect to dimension $i, j \in u$. The mixed derivative thus only contains factors with points from dimensions i and j.

3.1.2 Differentiating the Reproducing Kernel for a Multiquadric RBF in d Dimensions

The derivatives of the reproducing kernel for a multiquadric kernel function has to be generalized to d dimensions and a max order of m. In a similar way as in Section 3.1.1, the derivatives with respect to a certain dimension should only contain terms with points from that dimension.

Firstly, $\mathcal{D}$ is defined as the set of all dimensions up to d. For the subset of dimensions $u \in \Lambda$, and for a dimension $i \in u$ we define a set of subsets $\tilde{\Lambda}_i := \Lambda \backslash i$. In other words, $\tilde{\Lambda}_i$ is a subset of Λ, including all 2^{m-1} dimensional subsets $u \in \Lambda$ without the dimension i. With this definition, we write the first order derivative of the reproducing Kernel in d dimensions as

$$\frac{\partial}{\partial x_i}(K_\Lambda(x, y)) = \frac{\epsilon^2 (x_i - y_i)}{\sqrt{1 + \epsilon^2 (x_i - y_i)^2}} \left(1 + \sum_{u \in \tilde{\Lambda}_i} \prod_{j \in u} \left(\sqrt{1 + \epsilon^2 (x_j - y_j)^2} \right) \right). \tag{52}$$

Explained differently, we multiply the term corresponding to dimension i with the terms excluding this dimension. This works without losing any dimensions because the set of all subsets with max order m, $\Lambda = \left\{ \tilde{\Lambda}_i \cup \{i\} \cup \{\emptyset\} \right\}$. Thus, λ can be reconstructed by combining all the 2^{m-1} subsets without dimension i with the one set containing only dimension i and the empty set corresponding to the 1 in equation (52). Similarly, the second derivative of the reproducing kernel is

$$\frac{\partial^2}{\partial x_i^2}(K_\Lambda(\mathbf{x}, \mathbf{y})) = \frac{\epsilon^2}{(1 + \epsilon^2 (x_i - y_i)^2)^{\frac{3}{2}}} \left(1 + \sum_{u \in \tilde{\Lambda}_i} \prod_{j \in u} \left(\sqrt{1 + \epsilon^2 (x_j - y_j)^2} \right) \right). \tag{53}$$

The mixed derivatives can be calculated in a similar manner. However, another subset of Λ is defined and denoted $\tilde{\Lambda}_{i,j}$. This set includes all possible 2^{m-2} dimensional subsets up to order m, excluding dimensions $\{i, j\} \in u$. Now Λ can be rewritten a $\Lambda = \left\{ \tilde{\Lambda}_{i,j} \cup \{i\} \cup \{j\} \cup \{i, j\} \cup \{\emptyset\} \right\}$, showing that there are no loss of dimensional subsets u. Consequently, the mixed derivatives become

$$\begin{aligned} &\frac{\partial^2}{\partial x_i \partial x_j}(K_\Lambda(\mathbf{x}, \mathbf{y})) = \\ &\frac{\epsilon^4 (x_i - y_i)(x_j - y_j)}{\sqrt{1 + \epsilon^2 (x_i - y_i)^2} \cdot \sqrt{1 + \epsilon^2 (x_j - y_j)^2}} \cdot \left(1 + \sum_{u \in \tilde{\Lambda}_{i,j}} \prod_{k \in u} \left(\sqrt{1 + \epsilon^2 (x_k - y_k)^2} \right) \right). \end{aligned} \tag{54}$$

3.2 Placement of Anchor Points

As explained in Section 2.7.1, the reproducing kernel approximates the targeted function on a set of points in an anchored space. It is at these points that the PDE is solved. These points are generated according to the problem dimension d and maximum order m. The points are generated from a fixed anchor $\mathbf{a} = [a_1, \ldots, a_d]$ and iterative unfreezing of the dimensions in the subsets of Λ. This iteration is performed according to equations (38) and (39). Each iteration, points from the lower-dimensional anchored space are sampled and added to the set of anchor points. The amount of points used in each dimension is denoted n. In this project these n points are linearly spaced within a domain Ω, e.g, [0, 1].

Consequently, as m and therefore the sizes of the subsets in Λ increases, the amount of anchor points rapidly increases. The total number of anchor points is denoted N and follows equation (55). While N is dependent on d through the binomial term, the dominant term is n^m.

$$N = \sum_{i=0}^{m} \binom{d}{i} n^i. \tag{55}$$

For a three-dimensional problem with $m = 2$ the set of anchor points is shown below. A visual example of the same problem is given in Figure 4.

$$\begin{aligned}
\Lambda =& \{\emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}\}, \\
X =& \Big\{ \{(a_1, a_2, a_3)\} \\
& \cup \{(x_i, a_2, a_3)\}_i^n \cup \{(a_1, x_i, a_3)\}_i^n \cup \{(a_1, a_2, x_i)\}_i^n \\
& \cup \{(x_i, x_j, a_3)\}_{i,j}^n \cup \{(x_i, a_2, x_j)\}_{i,j}^n \cup \{(a_1, x_i, x_j,)\}_{i,j}^n \Big\}, \\
x_i \in& \Omega.
\end{aligned}$$

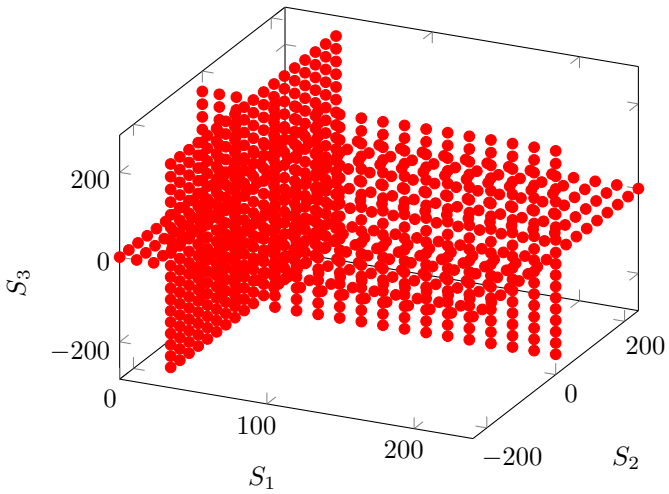


Figure 4: Example set of anchor points, prior to removal of undefined points, for a three-dimensional problem with $m = 2$

3.3 Transformation of Coordinates

As mentioned, given that the lower dimensional solutions are lacking information encoded in higher dimensions, the goal of the method is to represent the high dimensional function as well as

possible with the lower dimensional decompositions. When solving equation (23) with a maximum order lower than the dimension the placement of the anchor points is important. For an adequate approximation the anchor points must be placed so the lower dimensional solution captures the defining features of the high dimensional solution. However, the anchor points can not be placed anywhere. On the contrary, the placement of the anchor points is highly limited as they are generated from freezing projections, as seen in equations (38) and (39).

In the original basis, the anchor points fail to accurately capture the behaviour of the pricing space. The defining behaviour of the solution is in the direction were all assets grow, i.e. the diagonal. Because this direction is not captured by the anchor points in the original basis, the approximation is poor. Therefore, the space needs to be rotated such that the anchor points align with the diagonal. A coordinate transformation is applied to rotate the space accordingly. The transformation should also result in an orthonormal basis.

The Gram-Schmidt process for orthogonalization was used, taking a linear independent basis making it orthonormal. A feature of this process is that the direction of the first vector is kept. Given the d-dimensional linearly independent basis in equation (56), the Gram-Schmidth process is used to make the transformation matrix orthonormal.

$$\mathcal{T} = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 1 & 1 & & 0 \\ \vdots & & \ddots & 0 \\ 1 & 0 & \dots & 1 \end{bmatrix}. \tag{56}$$

A three dimensional orthonormal example-matrix can be seen in equation (57).

$$\mathcal{T} = \frac{1}{\sqrt{3}} \begin{bmatrix} 1 & -1/\sqrt{2} & -\sqrt{3}/\sqrt{2} \\ 1 & \sqrt{2} & 0 \\ 1 & -1/\sqrt{2} & \sqrt{3}/\sqrt{2} \end{bmatrix}. \tag{57}$$

3.3.1 Solving the PDE in the New Basis

As the PDE is defined in the basis of the underlying assets $\vec{s}$, after the coordinates are transformed the PDE also has to be transformed in accordance with this new basis $\vec{v}$. The first and second order derivatives are transformed using the chain-rule, as follows,

$$\begin{aligned} \frac{\partial^2 u}{\partial s_i \partial s_j} &= \sum_{k,l=1}^{d} \left(\frac{\partial^2 u}{\partial v_k \partial v_l} \frac{\mathrm{d}v_k}{\mathrm{d}s_i} \frac{\mathrm{d}v_l}{\mathrm{d}s_j} \right), \\ \frac{\partial u}{\partial s_i} &= \sum_{k=1}^{d} \left(\frac{\partial u}{\partial v_k} \frac{\mathrm{d}v_k}{\mathrm{d}s_i} \right). \end{aligned} \tag{58}$$

Since the transformation is linear, the terms $\mathrm{d}v_i / \mathrm{d}s_i$ in equation (58) are constants. These constants are obtained directly from the transformation matrix.

$$\frac{\mathrm{d}v_k}{\mathrm{d}s_i} = \mathcal{T}_{k,i} := a_{k,i}, \qquad \frac{\mathrm{d}v_k}{\mathrm{d}s_i} \frac{\mathrm{d}v_l}{\mathrm{d}s_j} = \mathcal{T}_{l,j}\, \mathcal{T}_{k,i} := b_{\{k,l\},\{i,j\}}. \tag{59}$$

The coefficient matrices D_1 and D_2 transforming the derivatives in v to s follow naturally. The size of D_1 is $d \times d$, and such the indexing goes from 1 to d. For the matrix D_2 the size does not grow linearly with d. Rather, the size of the matrix is $d + \binom{d}{2} \times d + \binom{d}{2}$ since there are be d non-mixed derivatives and $\binom{d}{2}$ mixed derivatives. The indexing of the second order derivatives is

therefore not as trivial. In D_2 the first d elements are the non-mixed derivatives, 11 to dd. After this the mixed derivatives follow in the order $12, 13, ..., 23, ..., d-1\,d$. This ordering is seen and used throughout the project.

$$\begin{bmatrix} \frac{\partial u}{\partial s_1} \\ \vdots \\ \frac{\partial u}{\partial s_d} \end{bmatrix} = \underbrace{\begin{bmatrix} a_{1,1} & \dots & a_{1,d} \\ \vdots & \ddots & \vdots \\ a_{d,1} & \dots & a_{d,d} \end{bmatrix}}_{D_1} \begin{bmatrix} \frac{\partial u}{\partial v_1} \\ \vdots \\ \frac{\partial u}{\partial v_d} \end{bmatrix}, \tag{60}$$

$$\begin{bmatrix} \frac{\partial^2 u}{\partial s_1^2} \\ \vdots \\ \frac{\partial^2 u}{\partial s_{d-1}\partial s_d} \end{bmatrix} = \underbrace{\begin{bmatrix} b_{\{1,1\},\{1,1\}} & \dots & b_{\{d-1,d\},\{1,1\}} \\ \vdots & \ddots & \vdots \\ b_{\{1,1\},\{d-1,d\}} & \dots & b_{\{d-1,d\},\{d-1,d\}} \end{bmatrix}}_{D_2} \begin{bmatrix} \frac{\partial^2 u}{\partial v_1^2} \\ \vdots \\ \frac{\partial^2 u}{\partial v_{d-1}\partial v_d} \end{bmatrix}. \tag{61}$$

PDE equation (23) can thus be rewritten with equation (58). Using equations (60) and (61) the transformed PDE is simplified as follows,

$$V^1 = \begin{bmatrix} | & & | \\ \mathcal{T}v_1 & \dots & \mathcal{T}v_d \\ | & & | \end{bmatrix}, \; V^2 = \begin{bmatrix} | & & | \\ \mathcal{T}(v_1 v_1) & \dots & \mathcal{T}(v_{d-1} v_d) \\ | & & | \end{bmatrix},$$

$$\hat{C} = \begin{bmatrix} C_{1,1} & 0 & \dots & 0 \\ 0 & C_{2,2} & & 0 \\ \vdots & & \ddots & 0 \\ 0 & 0 & \dots & C_{d-1,d} \end{bmatrix}, \tag{62}$$

$$\frac{\partial u}{\partial t} + \frac{1}{2}\sum_{k,l}^{d} V^2 \hat{C} b_{\{l,k\},\{\cdot,\cdot\}} \frac{\partial^2 u}{\partial v_k \partial v_l} + \sum_{k}^{d} r V^1 a_{\{k,\cdot\}} \frac{\partial u}{\partial v_k} - ru = 0, \tag{63}$$

where $a_{\{k,\cdot\}}, b_{\{l,k\},\cdot,\cdot\}}$ are the columns in D_1, D_2 corresponding with differentiation in v_k and $v_k v_l$ respectively.

3.3.2 Squeezing the Anchor Points

As the Black-Scholes PDE is defined for positive values of the underlying assets, one must be careful not to place anchor points beyond the defined region. In the standard coordinate system this is trivial. However, as the transformation effectively rotates space it is important to impose limits on the domain of the anchor points, corresponding to the defined region.

Given an anchor $\mathbf{a}$ and a transformation matrix $\mathcal{T}$ the defined domain in each dimension i is given as

$$x_i \in \Omega_i = \begin{cases} [0,\, s_{max}], & i = 1, \\ [-\min((\mathbf{a}/\mathcal{T}_{:,i}) > 0)), -\max((\mathbf{a}/\mathcal{T}_{:,i}) < 0))], & i \neq 1. \end{cases} \tag{64}$$

Solving the transformed PDE, equation (63), with the domain defined by equation (64) squeezes the anchor points such that in the standard system the points are bounded by the defined region.

For maximum order $m \geq 2$ however, the placement of the anchor points in the span of the axes may result in undefined points. There are currently removed from the set of anchor points. While this approach is not perfect, the points that are removed are far away from the evaluation zone and removing them might only speed up computations.

3.4 Solving the PDE

In the implementation, the user supplies the program with a set of evaluation points X_e as well as numerical and financial parameters. The user gives these points defined in the standard coordinate system $\vec{s}$ so they are transformed to the transformed system $\vec{v}$ before further computations.

The PDE is not solved at the evaluation points but at the anchor points described in Section 3.2 with domain from equation (64). The price at the evaluation points is only calculated when the PDE is solved on the anchor points.

Since the far boundary condition, cf. equation (23), is only applicable for points far away from strike the computational domain must be sufficiently large. In accordance with earlier implementation, this is chosen at $4dK$. This value, called s_{max}, is then used to scale down the problem to a numerically smaller domain. The evaluation points X_e, strike K and anchor $\mathbf{a}$, is all scaled by this factor. When computations are finished, the evaluation points are rescaled.

Furthermore, to correctly apply the boundary condition the set of anchor points X are split in subgroups corresponding with the appropriate conditions:
$X_{close}, X_{far}, X_{in}$ with size N_c, N_f and N_i respectively. Since these boundary conditions are defined in the standard coordinate system, and the anchor points in the transformed coordinate system, the anchor point are transformed prior to subgroup classification,

$$\begin{gathered} \vec{v} \in X, \\ \begin{cases} \vec{v} \in X_{close}, & \textbf{if } \|\mathcal{T}\vec{v}\|_2 = 0, \\ \vec{v} \in X_{far}, & \textbf{if } \|\mathcal{T}\vec{v}\|_2 \geq 1, \\ \vec{v} \in X_{in}, & \textbf{else}, \end{cases} \\ X = X_{close} \cup X_{far} \cup X_{in}. \end{gathered} \tag{65}$$

3.4.1 Derivatives with Kernel Methods

Continuing from equation (41), letting $U = [u(\vec{v}, t), \ldots]^T_{\vec{v} \in X}$ be the vector with the price at the anchor points in X. The coefficients λ can be written as:

$$\lambda = \underbrace{\left([K_\Lambda(\mathbf{x}, \mathbf{y})]_{\mathbf{x},\mathbf{y} \in X} \right)^{-1}}_{\mathbf{A}^{-1}} U. \tag{66}$$

Consequently, equation (66) can be substituted in equation (42) yielding the system for approximating the function at an arbitrary point $\vec{v}'$:

$$\underbrace{s_f(\vec{v}', t)}_{\mathcal{R}^{1\times 1}} = \underbrace{[K(\vec{v}', \mathbf{x}), \ldots]_{\mathbf{x} \in X}}_{\mathcal{R}^{1\times N}} \underbrace{\mathbf{A}^{-1}}_{\mathcal{R}^{N\times N}} \underbrace{U}_{\mathcal{R}^{N\times 1}}. \tag{67}$$

Equation (67) extends naturally when approximating at multiple points. Given a set X_e containing N_e evaluation points, the system becomes:

$$\underbrace{[s_f(\mathbf{x}, t)]_{\mathbf{x} \in X_e}}_{\mathcal{R}^{N_e\times 1}} = \underbrace{[K_\Lambda(\mathbf{x}, \mathbf{y})]_{\mathbf{x} \in X_e, \mathbf{y} \in X}}_{\mathcal{R}^{N_e\times N}} \underbrace{\mathbf{A}^{-1}}_{\mathcal{R}^{N\times N}} \underbrace{U}_{\mathcal{R}^{N\times 1}}. \tag{68}$$

Recalling equation (28), since the coefficients λ are only dependent on time, calculating the spatial derivative approximations at the evaluation points only requires differentiating the reproducing

kernel, namely,

$$\begin{aligned}\Big[\frac{\partial}{\partial x_i} s_f(\mathbf{x},t)\Big]_{\mathbf{x}\in X_e} &= \Big[\frac{\partial}{\partial x_i} K_\Lambda(\mathbf{x},\mathbf{y})\Big]_{\mathbf{x}\in X_e,\mathbf{y}\in X}\ \mathbf{A}^{-1}\ U,\\ \Big[\frac{\partial^2}{\partial x_i \partial x_j} s_f(\mathbf{x},t)\Big]_{\mathbf{x}\in X_e} &= \Big[\frac{\partial^2}{\partial x_i \partial x_j} K_\Lambda(\mathbf{x},\mathbf{y})\Big]_{\mathbf{x}\in X_e,\mathbf{y}\in X}\ \mathbf{A}^{-1}\ U.\end{aligned} \tag{69}$$

3.4.2 Black-Scholes PDE with Kernel Methods

The rotated PDE discussed in Section 3.3 and displayed in equation (63) is approximated with the kernel method described in Section 3.4.1, equations (68) and (69). With the differentiated reproducing kernels from Section 3.1.2, equations (52) through (54), the matrices used in the approximation are:

$$\begin{aligned}\mathbf{A_0} &= \Big[K_\Lambda(x,y)\Big]_{x,y\in X},\\ \mathbf{B_0} &= \Big[K_\Lambda(x,y)\Big]_{x\in X_{in},y\in X},\\ \mathbf{B_i} &= \Big[\frac{\partial}{\partial x_i} K_\Lambda(x,y)\Big]_{x\in X_{in},y\in X},\\ \mathbf{B_{ij}} &= \Big[\frac{\partial^2}{\partial x_i \partial x_j} K_\Lambda(x,y)\Big]_{x\in X_{in},y\in X}.\end{aligned} \tag{70}$$

The spatial part of the Black-Scholes PDE, equation (63), can be rewritten with the matrices in equation (70) as the operator $\mathcal{O}_{BS}$. The resulting system of ODEs becomes

$$\begin{aligned}\frac{\partial U(\vec{v})}{\partial \tau} &= \mathcal{O}_{BS} U(\cdot), && \vec{v}\in X_{in},\\ U(\vec{v}) &= \Phi(\mathcal{T}\vec{v},\tau), && \vec{v}\in X_{far},\\ U(\vec{v}) &= 0, && \vec{v}\in X_{close},\end{aligned} \tag{71}$$

$$\begin{aligned}\mathcal{O}_{BS} = \Big(&\frac{1}{2}\sum_k^d \sum_l^d V^2 \hat{C} b_{\{l,k\},\{\cdot,\cdot\}} \mathbf{B_{kl}}\\ &+ \sum_k^d r V^1 a_{\{k,\cdot\}} \mathbf{B_k} - r\mathbf{B_0}\Big)\mathbf{A_0}^{-1}.\end{aligned} \tag{72}$$

For convenience, the time is inverted to $\tau = T - t$.

3.4.3 Time Discretization - BDF2

The system of ODEs is solved with the *Backward differentiation formula 2* (BDF2). Time is discretized with $\vec{\tau} = \{\tau_j\}_{j=1}^M$ with time step $\Delta\tau_j = \tau_{j+1} - \tau_j$. Equation (71) is expanded as:

$$\begin{aligned}U^{j+2} - \frac{4}{3}U^{j+1} + \frac{1}{3}U^j &= \Delta\tau_j \mathcal{O}_{BS} U^{j+2},\\ \Leftrightarrow (I - \Delta\tau_j \mathcal{O}_{BS}) U^{j+2} &= \frac{4}{3}U^{j+1} - \frac{1}{3}U^j,\\ \Leftrightarrow U^{j+2} &= (I - \Delta\tau_j \mathcal{O}_{BS})^{-1}(\frac{4}{3}U^{j+1} - \frac{1}{3}U^j).\end{aligned} \tag{73}$$

To eliminate the boundary conditions, the matrices are expanded to size $N \times N$. This new matrix, C, has identical rows to $(I - \Delta\tau_j \mathcal{O}_{BS})$ for the indices corresponding with the interior points while being the identity matrix for the boundary rows. The boundary condition is then enforced by

applying the condition to the boundary-indices of the right-hand side $U_{RHS} = (\frac{4}{3}U^{j+1} - \frac{1}{3}U^j)$. This is done before solving the linear system. The final system is then given by

$$\begin{aligned} U^{j+2} &= C^{-1}U_{RHS}, && \vec{v} \in X, \\ \begin{cases} U_{RHS} = \frac{4}{3}U^{j+1} - \frac{1}{3}U^j, & \vec{v} \in X_{in}, \\ U_{RHS} = 0, & \vec{v} \in X_{close}, \\ U_{RHS} = \Phi(\mathcal{T}\vec{v}, \tau_{j+1}), & \vec{v} \in X_{far}. \end{cases} \end{aligned} \tag{74}$$

3.4.4 Interpolating to Desired Evaluation Points

Lastly, the final step is using the solution at the anchor points U to interpolate the solution at the evaluation points. These point are still defined in the standard basis and are transformed with the transformation matrix $\mathcal{T}$. Equation (68) was rewritten for this final step as seen below in equation (75).

$$[s_f(\mathbf{x}, t)]_{\mathbf{x} \in (\mathcal{T}^{-1}X_e)} = [K_\Lambda(\mathbf{x}, \mathbf{y})]_{\mathbf{x} \in (\mathcal{T}^{-1}X_e), \mathbf{y} \in X} \, \mathbf{A}^{-1} \, U. \tag{75}$$

The choice of evaluation points is arbitrary but choosing points far away or beyond the anchor points may result in highly inaccurate results.

4 Numerical Results

Here we present the numerical results obtained from the implemented solver. Unless specified otherwise, the used numerical and financial parameters are the ones shown below. While realistic and quite standard parameters, the domain from which the evaluation points are taken from is substantially smaller than the domain of the anchor points, equation 64.

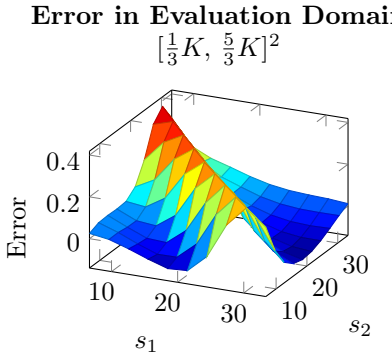
$$\begin{gathered} K = 20, \; T = 0.5, \; r = 0.02, \\ M = 10, \; n = 20, \; s_{max} = 4dK, \\ \epsilon = 50, \; \mathbf{a} = [21, 21, \ldots] \\ N_e = 10^d, \; x_e \in [\frac{1}{3}K, \frac{5}{3}K]^d, \\ C_{d=2} = \begin{bmatrix} \sigma_1^2 & \sigma_1\sigma_2\rho_{12} \\ \sigma_2\sigma_1\rho_{12} & \sigma_2^2 \end{bmatrix} = \begin{bmatrix} 0.0225 & 0.0150 \\ 0.0150 & 0.0400 \end{bmatrix}, \end{gathered}$$

$$C_{d=3} = \begin{bmatrix} \sigma_1^2 & \sigma_1\sigma_2\rho_{12} & \sigma_1\sigma_3\rho_{13} \\ \sigma_2\sigma_1\rho_{12} & \sigma_2^2 & \sigma_2\sigma_3\rho_{23} \\ \sigma_3\sigma_1\rho_{13} & \sigma_3\sigma_2\rho_{23} & \sigma_3^2 \end{bmatrix} = \begin{bmatrix} 0.0225 & 0.0150 & 0.0900 \\ 0.0150 & 0.0400 & 0.0060 \\ 0.0090 & 0.0060 & 0.0100 \end{bmatrix}.$$

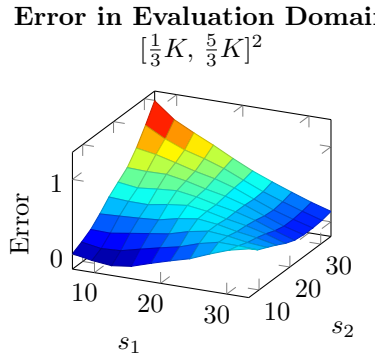
4.1 Two-Dimensional Problem

The experiments for the two-dimensional problem are done by an already existing solver. It calculates the fair basket option price with a kernel method and partition of unity. This solver has multiple kernels implemented, again using the muliquadric kernel.

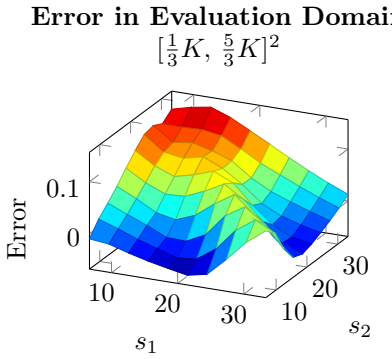
For all the following graphs the error is computed as the difference between the solver implemented in this study and the provided solver. Figure 5 shows the error when using max order $m = 1$ for three different problem variants: The standard parameters, increased time to maturity and increased number of spatial points.



(a) Standard parameters



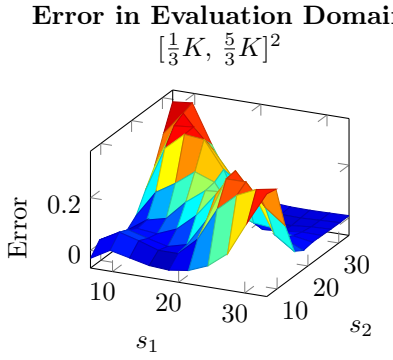
(b) Increased time to maturity $T = 3$



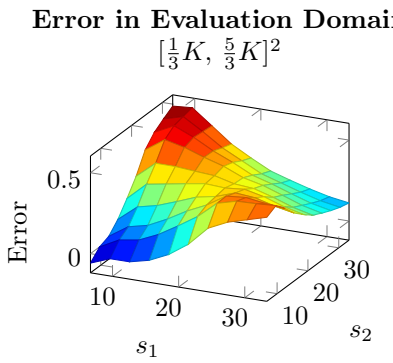
(c) Increased number of spatial points $n = 40$.

Figure 5: Error when solving a few variants of the two dimensional, max order one problem

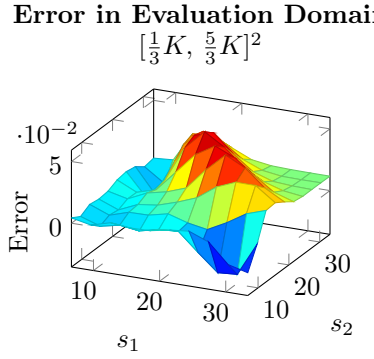
The same problems are solved with $m = 2$. These resulting errors are seen in Figure 6. Solving a problem with $m = d$ no longer constitutes a dimensional reduction but is still of intrest.



(a) Error, $d = 2, m = 2$.



(b) Error, $d = 2, m = 2$.Increased time to maturity $T = 3$

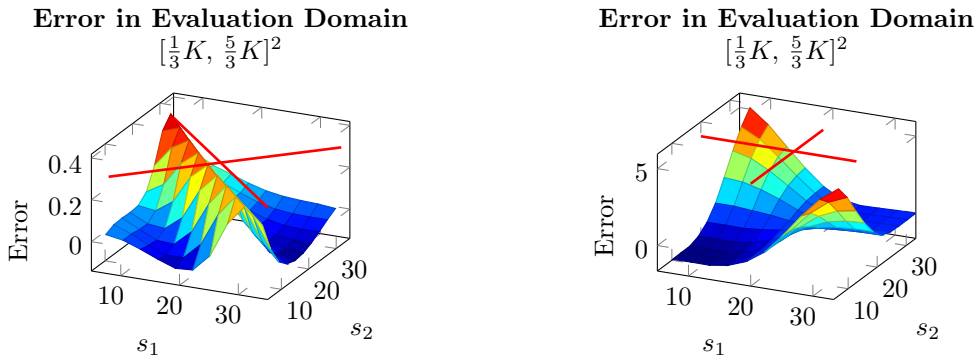


(c) Error, $d = 2, m = 2$. Increased number of points $n = 40$

Figure 6: Error when solving a few variants of the two dimensional, max order two problem

4.1.1 Transformed PDE

As discussed in Section 3.3, the PDE is transformed to better align the anchor points with the features of the solution. Investigating the improvement, Figure 7 compares the error when solving a two-dimensional problem with max order $m = 1$. As clearly visible in the plot, the solution of the original system is not able to capture the behaviour of the true solution, resulting in massive errors at the corners. The maximum error for the standard and transformed problems are 5.3 and 0.37 respectively.



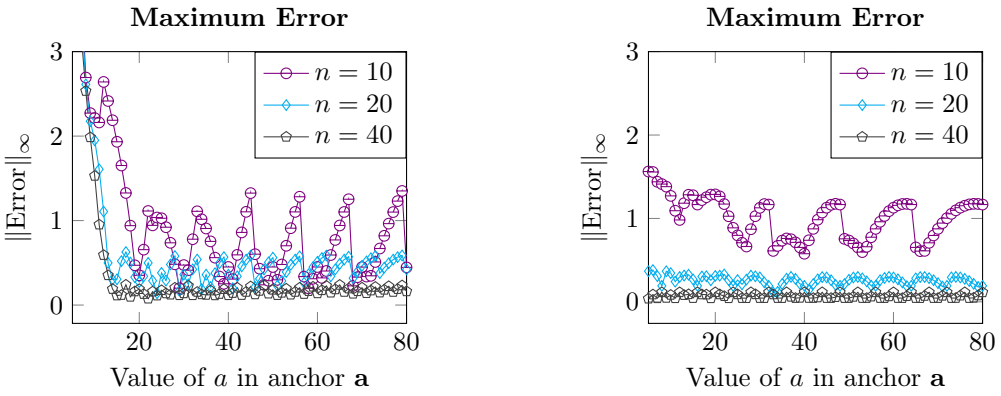
(a) Solved in Transformed Coordinates

(b) Solved in Standard Coordinates

Figure 7: Figures showing $d = 2, m = 1$ error when solved in different coordinate Systems. Red lines visualise placement of anchor points

4.1.2 Placement of Anchor

The choice of anchor $\mathbf{a}$ affects the placement of the anchor points. The maximum (infinity norm) error is calculated for increasing values of the elements in $\mathbf{a}$. The results for maximum order one and two is seen in Figure 8. The error initially drops, followed by oscillations. The amplitudes of the oscillations decrease with n but does not seem to be affected by m.



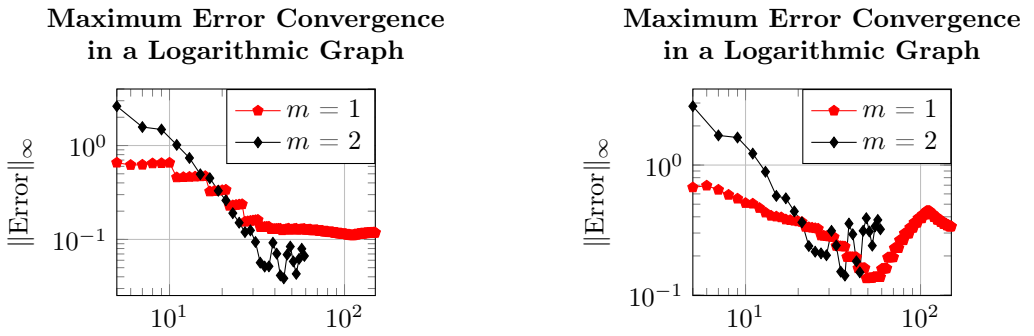
(a) $m = 1$

(b) $m = 2$

Figure 8: Figures showing maximum error when solved with different anchors $\mathbf{a}$

4.1.3 Convergence

For the two-dimensional problem, the maximum (infinity norm) error convergence when increasing number time steps M and spatial points n is seen in Figures 9 and 10. For spatial convergence, $T = 0$ is also tested. The error does, as expected, decrease when increasing n and M. Around $n = 60$ the error stagnates or, in the $T = 0$ case, starts increasing rapidly.



(a) Standard Parameters *(b) Computed at Maturity $T = 0$*

Figure 9: Logarithmic graphs Showing the infinity norm error convergence when increasing number of spatial points n

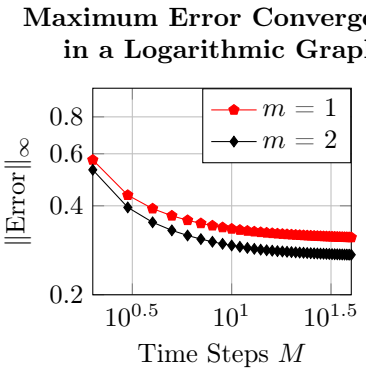


Figure 10: Logarithmic graphs showing the infinity norm error convergence when increasing number of time steps M

4.2 Higher Dimensional Problem

For the higher dimensional problem there are no good reference solution. Instead, the differences between high dimensional solutions generated with different m are investigated. These comparisons are done for $n = 10, 20$ and 30. Tables 1 to 3 displays the results. In these tables $U_{d,m}$ represent the d-dimensional solution computed with max order m. The higher dimensional C matrices are not shown.

Table 1: Comparison between generated solutions U with spatial points $n = 10$

d	$\|\|\mathbf{U_{d,1}} - \mathbf{U_{d,2}}\|\|_\infty$	$\|\|\mathbf{U_{d,2}} - \mathbf{U_{d,3}}\|\|_\infty$	$\|\|\mathbf{U_{d,1}} - \mathbf{U_{d,3}}\|\|_\infty$	$\|\|\mathbf{U_{d,1}}\|\|_\infty$
2	0.9401	N/A	N/A	13.4784
3	0.4472	0.8621	0.8209	13.5479
4	0.4621	0.9759	1.0818	13.2123
5	0.2976	0.7436	0.9822	13.0400
6	0.2660	-	-	12.8749

Table 2: Comparison between generated solutions U with spatial points $n = 20$

$\mathbf{d}$	$\|\|\mathbf{U_{d,1}} - \mathbf{U_{d,2}}\|\|_\infty$	$\|\|\mathbf{U_{d,2}} - \mathbf{U_{d,3}}\|\|_\infty$	$\|\|\mathbf{U_{d,1}} - \mathbf{U_{d,3}}\|\|_\infty$	$\|\|\mathbf{U_{d,1}}\|\|_\infty$
2	0.2667	N/A	N/A	13.5428
3	0.1435	0.3471	0.3865	13.5479
4	0.1796	-	-	13.5578
5	0.1368	-	-	13.5740
6	0.1131	-	-	13.5827

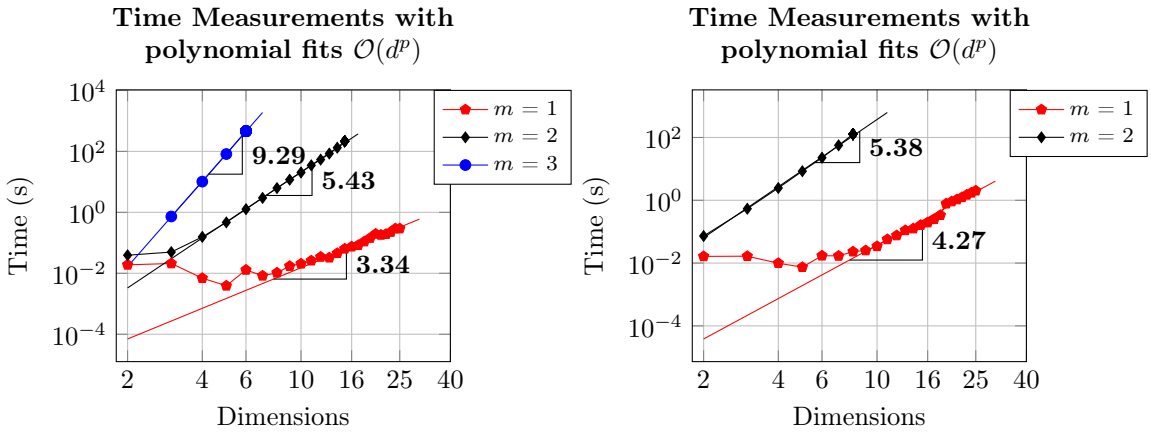
Table 3: Comparison between generated solutions U with spatial points $n = 40$

$\mathbf{d}$	$\|\|\mathbf{U_{d,1}} - \mathbf{U_{d,2}}\|\|_\infty$	$\|\|\mathbf{U_{d,2}} - \mathbf{U_{d,3}}\|\|_\infty$	$\|\|\mathbf{U_{d,1}} - \mathbf{U_{d,3}}\|\|_\infty$	$\|\|\mathbf{U_{d,1}}\|\|_\infty$
2	0.1186	N/A	N/A	13.5442
3	0.1607	-	-	13.5432
4*	0.1263	-	-	13.5437
5**	$10^{29.3}$	-	-	13.5427

*$m = 2$ *computations resulted in badly conditioned matrices*
**$m = 2$ *computations resulted in very badly conditioned matrices*

4.3 Complexity

The computational complexity of the implementation is studied by measuring run time for problems with increasing dimensions. These measurements are presented in Figure 11. For these runs only one evaluation point, placed at strike, is used.



(a) Spatial Points in Each Dimension $n = 10$

(b) Spatial Points in Each Dimension $n = 20$

Figure 11: Time complexity of different max order solution with different number of spatial points in logarithmic graph

4.4 Removed Points

Lastly, as the current sub-optimal implementation removes all points generated outside the defined domain the actual number of center points N might differ massively from the theoretical

amount. Figure 12 shows what percentage of points generated points were invalid and subsequently removed.

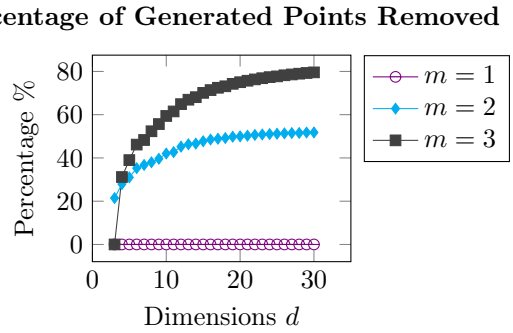


Figure 12: Graph showing the percentage of removed anchor points due to placement within undefined region

5 Discussion

5.1 Discussion Over Results

The method results in polynomial time scaling for all tested max order m, which can be seen in Figures 11 (a) and 11 (b). Still, the scaling differences between different degree polynomials is large — which becomes evident here. The polynomial scaling to the power of 9–10 (i.e m = 3) becomes impractical to run after about six dimensions, with a run time well into hundreds of seconds. The case when m=2, with the behaviour like a 5–6 degree polynomial, is better — dealing with up to about 16 dimensions before reaching impractical run times. The case with m=1, on the other hand, behaves like a 3–4 degree polynomial, and is much faster. Even the 25 dimensional problem is solved in under a second. Given that the error convergence, which can be seen in Figures 9 (a) and 9 (b), is fairly similar for the case when m=1 and m=2, it fortunately enough shows that the much faster m=1 case is not performing worse than the m=2 case.

The placement of the anchor does have an impact on the maximum error, but this impact is not too significant, which can be seen in Figures 8 (a) and 8 (b). Some oscillations can be seen in these figures, which are more prominent when using 10 points. A possible explanation could be that the points are spaced by two linspace commands in matlab, and the number of points are determined by the rounded fraction of the anchors placed on the grid. The distance in between the points on each side of the anchor is going to vary slightly when the anchor is moved just enough not to make a point change to the other side. This could affect how good the interpolation becomes around the discontinuous first order derivative at strike. The fact that the oscillations seems to double in frequency when going from 10 to 20 points suggests that this could be at least a partial explanation. The main difference between m=1 and m=2 seems to be that the behaviour is more stable when placing the anchor close to the origin.

The error from the time solver is not the dominant part, which can be seen in Figure 10. By the use of only 10 time points, the error from the time stepping has already converged to a very small value of about 0.02. This shows that the error from the time stepping quickly becomes smaller than the other errors present, and staying in the range 10–30 time steps is enough.

It is worth to mention the special case when the dimensionality of the problem is equal to the maximum order approximation used. It is chosen in this case to not perform the transformation/rotation, and let the coordinate system stay unchanged. This ensures that points which could fill out the region are not dropped unnecessarily, as a a large percentage of points are dropped for higher maximum orders, which can be seen in Figure 12.

5.2 Sources of Error

There are as always multiple sources of error present in a numerical approximation, and this is no exception. There are three types of errors worth special mentioning here:

In the approximation of derivatives with finite steps, we get a discretization error, which appears both in space and time. This error can be made smaller by the use of more points (which we see in figures 9 (a), 9 (b), and 10) — and as the spatial error is both larger and shrinks more slowly than the discretization error from the time solver, the discretization error from the time solver is probably not the dominating error encountered. The spatial discretization error is not as easy to reduce as the time one, and this stems from the discontinuous first order derivative at strike during the payoff. This is a main component of the observed error.

The main goal is to gain computational speed which comes at the cost of sacrificing some accuracy. The loss of information that occurs by approximating a high-dimensional function with a sum of low-dimensional ones introduces an approximation error. This error is made smaller by the use of a higher order approximation, and for lesser order approximations leads to some offset from the true value. The error estimations at Table 1 shows that there is a difference between different maximum order solutions, but that this is somewhat small. We can conclude that this type of error has an impact on the quality of the solution, but probably not as much as the error stemming from the kink in the payoff function.

After the solution in every anchor point is retrieved, the solution is interpolated to the evaluation region using the proposed kernel method. Depending on the choice of anchor points, this interpolation can be of varying quality. As an example, when using the unchanged coordinate system and a max order of one, the sampling is made by freezing all coordinates to the anchor, with the exception of one. In each of these orthogonal sampling lines, the values will resemble a ramp function, as the value along each of these lines almost is the one-dimensional payoff. The interpolation between these lines will have a linear shape, but this causes an overshoot in the corners along strike. This error is visible in Figure 7 (b). The error can also be understood by the straight line between two points on a ramp function graph, with one point to the left and one point to the right of where the ramp activates. The line will overshoot the real value, and it will be the most noticeable at the ramp function's equivalent of the strike — the origin. By instead using the implemented coordinate rotation, the region around strike will be better sampled, so this type of error will be reduced, which can be seen in Figure 7 (a).

6 Conclusions

In conclusion, the method proposed by Rieger and Wendland performs quite well for a high dimensional option pricing purpose, achieving low error with a polynomial dependence on the number of dimensions.

6.1 Possible Improvements and Future Research

6.1.1 Measurement of Error in Higher Dimensions

One such improvement could be to introduce a better measurement of the error for higher dimensions. The current way to quantify the error is to compare solutions with different max orders to each other, and see how much they differ. Investigating a better measurement is thus of interest. One could for example compare solutions with different grid resolution and max order.

6.1.2 Handling of Out-Of-Bounds Points

When using a max order higher than 1, there is a trade-off between filling the region and ensuring that every point lies in the viable region. The points outside of the region, i.e. with at least one negative coordinate, have to be dropped. The percentage of dropped points with respect to the

number of is seen in Figure 12. The optimisation of this balance is worth studying further. It is also of interest to study if there is a reliable way to generate points in any other shape than grids, e.g. a triangularly shaped mesh, as this could avoid having to drop as many points and therefore give rise to better quality approximation.

6.1.3 Other Ways of Placing the Anchor Points

The current implementation uses linearly spaced points for simplicity, but this leads to exponential ill-conditioning [11]. A better idea is to use e.g. Chebyshev points, which can avoid this exponential ill-conditioning.

6.1.4 Pricing of Put Options

If call options can be priced, then the step to apply the method for pricing put options is not that large. Allowing the pricing of put option would require changing the boundary and terminal conditions, and perhaps an investigation into where to place the anchor points. Because the payoff of a put option is non-zero between zero and the strike, this means that a solution based on fewer points would have very few points placed along the first coordinate in the rotated system being placed in the region of interest between just above strike and the origin, where the put option has a value larger than approximately zero.

6.1.5 Choosing the Components of Interest

The current implementation only allows the maximum order to be set for all components. A possible extension is to adjust model so that one can choose only a few important components of a larger max order. This would entail that the increased computational time is spent where it is most likely to improve the solution.

References

[1] Fischer Black and Myron Scholes. *The Pricing of Options and Corporate Liabilities. Journal of Political Economy vol. 81 (3)*, pages 554–637, 1973.

[2] Tomas Björk. *Arbitrage Theory in Continuous Time.* Oxford University Press, 4th edition, 2020.

[3] A.J.M. Ferreira. *A formulation of the multiquadric radial basis function method for the analysis of laminated composite plates, pages 385–392.* 2003. Departamento de Engenharia Mecânica e Gestão Industrial, Faculdade de Engenharia, Universidade do Porto.

[4] Martin D. Buhmann and Nira Dyn. *Spectral convergence of multiquadric interpolation. Proceedings of the Edinburgh Mathematical Society vol. 36(2)*, pages 319–333, 1993. Cambridge University press.

[5] Martin D. Buhmann. *Radial Basis Functions - Theory and Implementations. Cambridge Monographs on Applied and Computational Mathematics vol. 12*, 2009. Cambridge University press.

[6] Gregory E. Fasshauer. *Meshfree Approximation Methods with MATLAB.* 2007. Illinois Institute of Technology, USA.

[7] S.A Sarra. *Integrated Multiquadric Radial Basis Function Approximation Methods. Computers and Mathematics With Applications vol. 51, pages 1283–1296*, 2006. Department of Mathematics, Marshall University.

[8] Vern I Paulsen and Mrinal Raghupathi. *An Introduction to the Theory of Reproducing Kernel Hilbert Spaces.* 2016. Cambridge University Press.

[9] Christian Rieger and Holger Wendland. *On the Approximability and Curse of Dimensionality of Certain Classes of High-Dimensional Functions.* Preprint.

[10] Victor Shcherbakov. *Radial Basis Function Methods for Pricing Multi-Asset Options.* 2016. Division of Scientific Computing - Department of Information Technology, Uppsala.

[11] Rodrigo B. Platte, Lloyd N. Trefethen, and Arno B. J. Kuijlaars. *Impossibility of Fast Stable Approximation of Analytic Functions from Equispaced Samples. SIAM review 53(2)*, pages 308–318, 2011.