Multi-dimensional option pricing using radial basis functions and the generalized Fourier transform

Elisabeth Larsson [*,1] Krister Åhlander Andreas Hall

Address: Department of Information Technology, Uppsala University, Box 337, SE-751 05 Uppsala, Sweden.

Abstract

We show that the generalized Fourier transform can be used for reducing the computational cost and memory requirements of radial basis function methods for multi-dimensional option pricing. We derive a general algorithm, including a transformation of the Black–Scholes equation into the heat equation, that can be used in any number of dimensions. Numerical experiments in two and three dimensions show that the gain is substantial even for small problem sizes. Furthermore, the gain increases with the number of dimensions.

Key words: Radial basis function, RBF, generalized Fourier transform, GFT, Black–Scholes equation

1 Introduction

Trading of financial derivatives such as options is a continuous business going on all over the world today. Making sure that the prices are updated and correct at every time is of great importance for the traders.

For uncomplicated derivatives analytical formulas for the price are available, but for more complex derivatives, computer simulations are needed. One area with extreme computational demands is pricing of options with many underlying assets. This results in d-dimensional problems, where d is the number of underlying assets.

* *Email address*: Elisabeth.Larsson@it.uu.se

[1] The work was supported by a grant from the Swedish Research Council.

Among the most used methods for pricing options are Monte-Carlo or quasi-Monte Carlo simulations [10]. These methods scale linearly with the number of dimensions, but converge slowly. An alternative approach is to solve the Black-Scholes equation [4], which is a partial differential equation (PDE). A standard method for solving PDEs is finite difference approximation. In this work, we use an efficient adaptive finite difference method [20] for reference solutions in two dimensions. Another emerging method for high-dimensional financial problems is "sparse grids" [7].

Here, we are concerned with a relatively novel approach in this area, namely radial basis function (RBF) approximation. The main motivation for using RBFs is that they can provide spectral convergence rate [18,6], and that the methods are easy to implement in any number of dimensions. The basic idea in RBF approximation is to let the approximate solution have the form

$$u(x) = \sum_{j=1}^{N} \lambda_j \phi(\|x - x^j\|),$$

where $\phi(r)$ is the RBF, $x^j = (x_1^j, \ldots, x_d^j)$, $j = 1, \ldots, N$ are center points, and λ_j are coefficients that are determined through the conditions (equations) that the solution must fulfill. Applications of RBF methods to option pricing can be found in, e.g., [13,19,27,8,21].

One disadvantage with RBF methods is that the approximation problem typically results in a dense system of equations to solve. Therefore, both storage requirements and computational costs become prohibitive for large numbers of unknowns. Different ways of speeding up the computations that have been proposed are, e.g., fast evaluation through the Gauss transform [22] and preconditioned iterative methods together with domain decomposition [17].

The aim of this paper is to show that the generalized Fourier transform (GFT), under certain conditions, can be used both for reducing the memory requirements and the computational cost of RBF methods. The purpose of applying the GFT is to exploit geometrical symmetries. Use of the GFT when solving dense systems of equations was developed by Allgower et al. The computational overhead for applying the GFT to this type of problem is low, and the gain is substantial even for small problem sizes [3].

The multi-dimensional option pricing problem is not immediately suited for the use of the GFT. However, we show that the problem can be transformed in such a way that large parts exhibit the right symmetries.

All the techniques described here can be generalized to d dimensions. However, we have only implemented the algorithms in two and three dimensions so far.

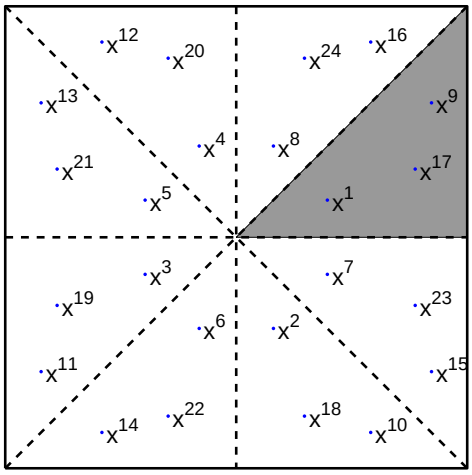

Fig. 1. A square with symmetrically positioned RBF points $x^i = (x_1^i,\ x_2^i)$.

The outline of the paper is as follows. In Section 2, we give a brief introduction to the GFT. Section 3 describes the problem, the transformations we apply, and the computational algorithm. Some details concerning the method are given in Section 4 and then Section 5 contains the numerical results. Finally we conclude the paper with a discussion in Section 6.

2 Exploiting symmetries through the GFT

The purpose of the GFT is to exploit geometrical symmetries [3]. To make the paper self-contained, we give a brief account for the machinery involved. Our notation follows a recent introduction to the subject [1] closely. First, we discuss the connection between geometrical symmetries and groups, and define equivariance. Then, we explain how the GFT block-diagonalizes equivariant matrices.

2.1 Symmetry and groups

Consider the computational domain in Figure 1. The square is mapped onto itself by a rotation 90 degrees, by a reflection in the x_2-axis, or by any composition of these transformations. In everyday life, we say informally that the square is "symmetric". More formally, introduce transformation matrices

$$\mathbf{A} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}, \ \mathbf{B} = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix},$$

which act on coordinates $x = (x_1,\ x_2)$ by matrix multiplication $x\mathbf{A}$ for the rotation and $x\mathbf{B}$ for the reflection. The square is then invariant under the *group* of transformation matrices generated by $\mathbf{A}$ and $\mathbf{B}$. It is easily seen that this group contains the following matrices

$$\mathcal{D} = \left\{\mathbf{I}, \mathbf{A}, \mathbf{A}^2, \mathbf{A}^3, \mathbf{B}, \mathbf{AB}, \mathbf{A}^2\mathbf{B}, \mathbf{A}^3\mathbf{B}\right\}. \tag{1}$$

It is readily verified that the identity belongs to the group, the group is associative, and each element has an inverse which belongs to the group, which means that the axioms of a group are satisfied. The group is not abelian, since for example $\mathbf{AB} \neq \mathbf{BA}$.

The transformations under which a geometrical object is invariant is referred to as the *symmetry group* of that object. Thus, $\mathcal{D}$ is the symmetry group of the square. This group has exactly the same structure as $\mathcal{D}_4$, the dihedral group with 8 elements. The abstract definition of this group is the group generated by two elements α and β such that the following relations hold:

$$\alpha^4 = e,\ \alpha\beta = \beta\alpha^{-1},$$

where e denotes identity. It is easy to verify that $\mathcal{D}$ and $\mathcal{D}_4$ has the same structure, by identifying α with $\mathbf{A}$ and β with $\mathbf{B}$. For later reference, we denote this mapping $\delta : \mathcal{D}_4 \to \mathcal{D}$. By verifying that δ is invertible and that $\delta(gh) = \delta(g)\delta(h)$ for all $g, h \in \mathcal{D}_4$, we confirm that $\mathcal{D}$ and $\mathcal{D}_4$ are *isomorphic*, i.e., they have the same structure.

Now consider the action of $\mathcal{D}$ on the single point x^1 in Figure 1. Each transformation maps x^1 to another point x^i, where $i \in \mathcal{O}_1 = \{1, \ldots, 8\}$. This is an example of an *orbit* under the action of a group. The figure illustrates $m = 3$ orbits. Each orbit may be identified by a single point in its orbit. In our example, such a selection may be chosen as $\{x^1, x^9, x^{17}\}$. All orbits shown in the figure have $|\mathcal{G}|$ elements, which is a consequence of the fact that the action is *free*, i.e., every non identity transformation moves every point x^i. If, for example, the origin had been included in the set of points, we would have had an orbit with just a single point. A point which is fixed under a transformation other than the identity is known as a "fix point".

The indices in $\mathcal{O}_1$ are enumerated such that $x^1 = x^1\mathbf{I}$, $x^2 = x^1\mathbf{A}$, $\ldots$, $x^8 = x^1\mathbf{A}^3\mathbf{B}$, in accordance with the enumeration (1) of $\mathcal{D}$.

By identifying a point x^i with its corresponding index i, and since we already via δ have identified $\mathbf{G} \in \mathcal{D}$ with $g \in \mathcal{D}_4$, we may as well describe the action by considering how $\mathcal{D}_4$ acts on the set of indices $\mathcal{I}$. We use the notation $j = ig$ to indicate that $x^j = x^i\mathbf{G}$, where $\mathbf{G} = \delta(g)$. When the action is regarded

like this, it is natural to represent the orbits by a selection $\mathcal{S}$ of indices, e.g., $\mathcal{S} = \{1, 9, 17\}$.

Another interpretation of the action is that a transformation $\mathbf{G}$ *permutes* the indices. In our example, it is easy to see that every transformation in $\mathcal{D}$ corresponds to a unique permutation. Every such permutation may be represented by a *permutation matrix*, and it can be shown that these permutation matrices form a group $\mathcal{P}$ which is isomorphic to $\mathcal{D}_4$. For later reference, we denote the isomorphism $\pi : \mathcal{D}_4 \to \mathcal{P}$.

We have now introduced the notation required to describe how geometrical symmetry may be exploited when discretizing many important PDE operators, e.g., the Laplacian Δ. Note that the Laplacian commutes with every rotation or reflection, i.e., if we apply the Laplacian to a field and then rotate or reflect the field, the result is the same as if we first rotate or reflect the field and then apply the Laplacian. In our example, we consider the rotations and reflections in $\mathcal{D}$. Since the Laplacian commutes with every transformation in $\mathcal{D}$, it is said to be *equivariant* with respect to $\mathcal{D}$.

In discretized form, a matrix is said to be *equivariant* with respect to a group $\mathcal{G}$ of matrices if it commutes with every matrix in the group. If every matrix in the group is a permutation matrix, equivariance may be expressed in terms of group actions on matrix indices. This is convenient for our application, and we use the following definition:

Definition 1 *Given a group $\mathcal{G}$ acting on $\mathcal{I} = \{1, \ldots, n\}$, an $(n \times n)$ matrix $\mathbf{A}$ is equivariant if*

$$\mathbf{A}_{ig,jg} = \mathbf{A}_{i,j} \tag{2}$$

holds for all indices $i, j \in \mathcal{I}$ and all $g \in \mathcal{G}$.

It is obvious from this definition, that an equivariant matrix $\mathbf{A}$ contains a large amount of redundant information. Let there be m orbits in $\mathcal{I}$, represented by a selection $\mathcal{S}$ of indices. All n^2 elements of $\mathbf{A}$ can then be represented by the nm elements, $\mathbf{A}_{i,j}, i \in \mathcal{I}, j \in \mathcal{S}$. If we assume that the action is free, all orbits have $|\mathcal{G}|$ elements and the memory requirement for dense matrices hence decreases with the size of the group. In our example, this means that a 24×24 matrix $\mathbf{A}$ which is equivariant with respect to $\mathcal{P}$ has 24×3 independent elements.

We conclude this section with the observation that many important RBF discretization matrices are inherently equivariant if the RBF node points are invariant under a group of isometric transformations. For example, cf. Section 3.4, if $\mathbf{A}_{i,j} = \phi(\|x^i - x^j\|)$, we have that

$$\mathbf{A}_{i,j} = \phi(\|x^{ig} - x^{jg}\|) = \mathbf{A}_{ig,jg},$$

since isometric transformations such as rotations and reflections are distance preserving. The same property holds for a discretization $\mathbf{B}$ of the Laplacian, where $\mathbf{B}_{i,j} = \Delta\phi(\|x^i - x^j\|)$. We stress that this observation saves not only memory, but also computations.

2.2 Block-diagonalization

If a matrix is equivariant, it is simple to use this fact to reduce memory requirement as well as the computations needed to construct it. In our applications it is even more important that an equivariant matrix can be block-diagonalized. In order to describe the formulas, we must first introduce some representation theory.

A (complex) representation $\rho : \mathcal{G} \to \mathbb{C}^{d\times d}$ of dimension d is a mapping from a group $\mathcal{G}$ to a group of matrices, in such a way that $\rho(gh) = \rho(g)\rho(h)$ for all $g, h \in \mathcal{G}$.

Examples of representations were given in the previous section. The mapping δ is a representation of dimension 2, and π is a representation of dimension 24. We mentioned that both $\mathcal{D}$ and $\mathcal{P}$ are isomorphic to $\mathcal{D}_4$, which implies that δ and π are invertible. This need not be the case. For example, every group $\mathcal{G}$ has the trivial representation $\tau : \mathcal{G} \to \mathbb{C}$, where $\tau(g) = 1$ for all $g \in \mathcal{G}$.

If $\rho : \mathcal{G} \to \mathbb{C}^{d\times d}$ is a representation and $\mathbf{T}$ is a nonsingular $d \times d$ matrix, we note that a representation $\sigma : \mathcal{G} \to \mathbb{C}^{d\times d}$ may be defined by $\sigma(g) = \mathbf{T}\rho(g)\mathbf{T}^{-1}$. Any two representations which only differs by such a shift of coordinates are said to be *equivalent*. If a representation ρ is equivalent to any block-diagonal matrix σ, ρ is said to be *reducible*, else, ρ is irreducible. A key result of representation theory says that every representation is reducible into irreducible representations. To every group, there exists a complete list $\mathcal{R}$ of nonequivalent irreducible representations, and this list is unique up to equivalence. For complex representations of a finite group $\mathcal{G}$, it holds that $\sum_{\rho\in\mathcal{R}} d_\rho^2 = |\mathcal{G}|$.

In our example, it is obvious that τ is irreducible, and it can be shown that δ is irreducible. There are three more nonequivalent irreducible representations, see Table 1.

The representation π is however reducible. Since every permutation matrix in $\mathcal{P}$ only permutes indices in the same orbit, every permutation matrix in $\mathcal{P}$ is block-diagonal with 3 blocks of size 8×8. This implies that π has three subrepresentations of dimension 8. More interesting, however, is the fact that each of these 3 subrepresentations is reducible as well and may be reduced into 4 nonequivalent representations of dimension 1, and 2 equivalent representations of dimension 2 [25]. The reduction of π can be used to block-diagonalize

Table 1
Number, notation (if applicable), dimension and generators for the complete list of irreducible representations for $\mathcal{D}_4$.

N:r	Notation	Dim.	a	b
1	τ	1	1	1
2	N/A	1	1	-1
3	N/A	1	-1	1
4	N/A	1	-1	-1
5	δ	2	$\mathbf{A}$	$\mathbf{B}$

equivariant matrices, as we have discussed in detail elsewhere [2]. In this paper, we will describe the theory for free actions.

Let $\mathbf{C}$ be an $n \times k$ matrix. Let $\mathcal{G}$ be a finite group acting freely on $\mathcal{I} = \{1 \ldots, n\}$, and let $\mathcal{S}$ be a selection of m orbits of size $|\mathcal{G}|$. Let $\mathcal{R}$ be a complete list of nonequivalent irreducible representations for $\mathcal{G}$. The GFT $\hat{C}$ of $\mathbf{C}$ is given by:

$$\hat{C}_{i,j}(\rho) = \sum_{g \in \mathcal{G}} \mathbf{C}_{ig,j} \rho(g), \tag{3}$$

for all $\rho \in \mathcal{R}$, $i \in \mathcal{S}$, and $j = 1, \ldots, k$. The inverse GFT (IGFT) is given by:

$$\mathbf{C}_{ig,j} = \sum_{\rho \in \mathcal{R}} \frac{d_\rho}{|\mathcal{G}|} \text{trace}\,(\hat{C}_{i,j}(\rho)\rho(g^{-1})), \tag{4}$$

for all $g \in \mathcal{G}$, $i \in \mathcal{S}$, and $j = 1, \ldots, k$.

We define the GFT of an equivariant matrix as a slight variation. It is obtained by applying the GFT to a selection of the columns. Let $\mathbf{A}$ be an equivariant matrix under the conditions above. The GFT of an equivariant matrix $\mathbf{A}$ is given by:

$$\hat{A}_{i,j}(\rho) = \sum_{g \in \mathcal{G}} \mathbf{A}_{ig,j}, \tag{5}$$

for all $\rho \in \mathcal{R}$, $i, j \in \mathcal{S}$. The IGFT is computed by (4) and by exploiting the equivariance property (2).

Block-diagonalization via the GFT can now be described. Assume $\mathbf{A}$ and $\mathbf{C}$ as above, and let $\mathbf{X}$ be a $n \times k$ matrix. The linear system of equations $\mathbf{AX} = \mathbf{C}$

with k right-hand sides can be solved as follows:

(1) Transform $\hat{C} = \text{gft}(\mathbf{C})$ according to (3).
(2) Transform $\hat{A} = \text{gft}(\mathbf{A})$ according to (5).
(3) For each $\rho \in \mathcal{R}$, solve the linear system of equations

$$\hat{A}(\rho)\hat{X}(\rho) = \hat{C}(\rho).$$

This is an $md_\rho \times md_\rho$ system with kd_ρ right-hand sides.
(4) Inverse transform $\mathbf{X} = \text{igft}(\hat{X})$ according to (4).

The block-diagonalization is thus realized in the transformed space, since the independent systems $\hat{A}(\rho)\hat{X}(\rho) = \hat{C}(\rho)$ can be written as one block-diagonal system $\hat{A}\hat{X} = \hat{C}$, where each matrix has as many blocks on the diagonal as there are irreducible representations in $\mathcal{R}$.

In the case of $\mathcal{D}_4$, the dimensions of the transformed matrices $\hat{A}$ and $\hat{C}$ can be deduced from Table 1: $\hat{A}$ has four $m \times m$ blocks and one $2m \times 2m$ block; $\hat{C}$ has four $m \times k$ blocks and one $2m \times 2k$ block. If a direct $\mathcal{O}(n^3)$ method is used for solving systems of equations, it is easy to see that the transformed systems require $|\mathcal{G}|^3/(\sum_{\rho\in\mathcal{R}} d_\rho^3)$ times the work required for solving the original system [1]. In two dimensions where we study symmetry under $\mathcal{D}_4$, this corresponds to a gain of about 42. In higher dimensions, the expected gain increases rapidly. For three dimensions, the symmetry group of the cube has 48 elements, 10 irreducible representations, and the estimated gain is 864. For the solution of dense equivariant systems, it is thus well established that block-diagonalization via the GFT speeds up the computations considerably [1,28]. The challenge that we address in this paper, is to study PDEs where the original PDE operator is not equivariant under a group of rotations and reflections, and where the boundary conditions destroy equivariance. The latter issue has also been studied by Bonnet [5].

3 Casting the multi-dimensional option pricing problem into symmetric form

We consider the valuation of European basket call options. The holder of such an option has the right to buy a specified combination of stocks at the strike price K at the exercise time T. The dimension d of the problem is given by the number of underlying stocks. First we give the usual mathematical formulation and then we show how to adapt it so that the problem symmetries can be exploited through the GFT.

The option value $F(t,s)$, at time t with underlying stock prices $s=(s_1,\ldots,s_d)^T$, is the solution of the following final value problem

$$\frac{\partial F}{\partial t}(t,s)+\mathcal{L}F(t,s)=0,\quad t<T,\quad s\in\mathbb{R}^d_+, \tag{6}$$

$$F(T,s)=\Phi(s),\quad s\in\mathbb{R}^d_+, \tag{7}$$

where (6) is the multi-dimensional Black–Scholes equation with the spatial operator

$$\mathcal{L}F(t,s)=r\sum_{i=1}^{d}s_i\frac{\partial F}{\partial s_i}+\frac{1}{2}\sum_{i,j=1}^{d}\left[\sigma\sigma^T\right]_{ij}s_is_j\frac{\partial^2 F}{\partial s_i\partial s_j}-rF, \tag{8}$$

where r is the risk free interest rate, and σ is the volatility matrix. The contract function $\Phi(s)$ can be defined in different ways. We use the example

$$\Phi(s)=\max\left(0\,,\,\frac{1}{d}\sum_{i=1}^{d}s_i-K\right). \tag{9}$$

When solving the problem numerically, it is convenient to rewrite it as an initial value problem through the transformation $\tau=T-t$ leading to

$$\frac{\partial F}{\partial \tau}(\tau,s)=\mathcal{L}F(\tau,s),\quad \tau>0,\quad s\in\mathbb{R}^d_+, \tag{10}$$

$$F(0,s)=\Phi(s),\quad s\in\mathbb{R}^d_+, \tag{11}$$

In order to facilitate transformation of the problem, we also reformulate the spatial operator as

$$\mathcal{L}F(\tau,s)=\frac{1}{2}\sum_{k=1}^{d}\sigma_k^T S\nabla_s(\nabla_s F)^T S^T\sigma_k+rs^T\nabla_s F-rF, \tag{12}$$

where σ_k is the kth column of the volatility matrix, $S=S^T=\mathrm{diag}(s)$, and the gradient of a scalar is taken as a column vector.

The following asymptotic boundary conditions are employed at the near and far field boundaries respectively

$$F(\tau,s)\to 0,\quad \|s\|\to 0, \tag{13}$$

$$F(\tau, s) \to \frac{1}{d}\sum_{i=1}^{d} s_i - Ke^{-r\tau}, \quad \|s\| \to \infty. \tag{14}$$

For a more extensive discussion of these boundary conditions, see [21].

3.2 Symmetrizing the equation

The operator $\mathcal{L}$ in (12) is not symmetry preserving under simple transformations such as those described in Section 2. In order to apply the GFT to the problem, we need to transform the equation into for example the heat equation. We generalize the steps outlined in [26, p. 110] for the one-dimensional problem to d dimensions. A similar transformation is also used in [19] for a two-dimensional problem. The first step is to turn the second-order term into a Laplacian. We use a transformation of the following form

$$s = \exp(A^T x),$$

where A is a $(d \times d)$ matrix. The Jacobian of the transformation is given by

$$J = \left[\frac{\partial s_j}{\partial x_i}\right]_{i,j=1}^{d} = AS.$$

The relations between gradients and Hessians are given by

$$\nabla_x F = J\nabla_s F = AS\nabla_s F,$$

$$\nabla_x(\nabla_x F)^T = AS\nabla_s(\nabla_s F)^T S^T A^T + ASG_s A^T,$$

where $G_s = \mathrm{diag}(\nabla_s F)$. Assuming that A is non-singular, (12) is transformed into

$$\mathcal{L}F(\tau, x) = \frac{1}{2}\sum_{k=1}^{d} \sigma_k^T A^{-1}\nabla_x(\nabla_x F)^T A^{-T}\sigma_k + \alpha^T A^{-1}\nabla_x F - rF, \tag{15}$$

where $\alpha_i = r - \frac{1}{2}\sum_{k=1}^{d}\sigma_{ik}^2$. The condition that the second order terms are reduced to the Laplacian becomes

$$\frac{1}{2}\sum_{k=1}^{d}(\sigma_k^T A^{-1})^T(A^{-T}\sigma_k)^T = \frac{1}{2}\sum_{k=1}^{d} A^{-T}\sigma_k\sigma_k^T A^{-1} = \frac{1}{2}A^{-T}\sum_{k=1}^{d}(\sigma_k\sigma_k^T)A^{-1} = I$$

or

$$A^T A = \frac{1}{2}\sum_{k=1}^{d}(\sigma_k \sigma_k^T). \tag{16}$$

Theorem 3.1 *The equation (16) has a non-singular solution A if the volatility matrix σ has full rank. Furthermore, if A is a solution, then $B = QA$, where Q is an orthogonal matrix, is also a solution.*

Proof: Let v be an arbitrary vector in $\mathbb{R}^d$ with $\|v\| = 1$. Then

$$v^T \left(\sum_{k=1}^{d}(\sigma_k \sigma_k^T) \right) v = \sum_{k=1}^{d}(v^T \sigma_k)(\sigma_k^T v) = \sum_{k=1}^{d} |\sigma_k^T v|^2 \geq 0.$$

Furthermore, if σ has full rank, $\{\sigma_k\}_{k=1}^d$ are d linearly independent vectors and v cannot be orthogonal to all of them. Hence,

$$v^T \left(\sum_{k=1}^{d}(\sigma_k \sigma_k^T) \right) v > 0,$$

meaning that the right hand side of equation (16) is positive definite. Then A can be found by Cholesky factorization of the right hand side. The computed A can always be augmented by an orthogonal matrix, since $B^T B = (QA)^T(QA) = A^T Q^T Q A = A^T A$. □

A Laplacian operator is invariant under rotations, reflections, and translations. The orthogonal matrix Q above can be interpreted as a rotation or a reflection. We can also add a translation b to the transformation without affecting the criterion (16). This leads to the more general transformation

$$s = \exp(A^T(Q^T x + b)).$$

This is used in Section 3.3 in order to position the computational domain in a beneficial way.

With this transformation and the choice of A described above, the PDE (10) is reduced to

$$\frac{\partial F}{\partial \tau} = \Delta_x F + \alpha^T (QA)^{-1} \nabla_x F - rF, \tag{17}$$

The next step is to transform the function in order to remove the gradient term. Let $F(\tau, x) = e^{\gamma\tau + \xi^T x} P(\tau, x) = g(\tau, x)P(\tau, x)$. The Laplacian and the

gradient now become

$$\nabla_x F = gP\xi + g\nabla_x P,$$

$$\Delta_x F = \xi^T \xi g P + 2g\xi^T \nabla_x P + g\Delta_x P.$$

Inserting these expressions and dividing by $g \neq 0$ we get a new PDE

$$\begin{aligned}\frac{\partial P}{\partial \tau} = \Delta_x P &+ (\alpha^T (QA)^{-1} + 2\xi^T)\nabla_x P \\ &+ (-\gamma + \alpha^T (QA)^{-1}\xi + \xi^T \xi - r)P. \end{aligned} \tag{18}$$

The final step is to choose

$$\xi^T = -\frac{1}{2}\alpha^T (QA)^{-1},$$

and

$$\gamma = \alpha^T (QA)^{-1}\xi + \xi^T \xi - r = -\xi^T \xi - r.$$

The multi-dimensional option pricing problem has now been reduced to the heat equation. Together with boundary conditions and initial condition, we have

$$\begin{aligned} \frac{\partial P}{\partial \tau} &= \Delta_x P, && x \in \Omega, \quad \tau > 0, \\ P(\tau, x) &= \mathrm{e}^{-\gamma\tau - \xi^T x} F(\tau, x) \equiv f(\tau, x), && x \in \Gamma_D, \, \tau > 0, \\ P(0, x) &= \mathrm{e}^{-\gamma\tau - \xi^T x} \Phi(x) \equiv f_0(x), && x \in \Omega, \end{aligned} \tag{19}$$

where Ω is the computational domain and Γ_D is the union of the near and far field part of the boundary where we enforce Dirichlet boundary conditions.

3.3 Choosing a symmetric computational domain

In the original s-coordinates, we want the computational domain to include the region $R = \{s \mid \frac{1}{d}\sum_{i=1}^{d} s_i \leq 4K, s_i \geq 0, i = 1, \ldots, d\}$. This is related to our choice of contract function, see [21]. Other choices of R could also be made. When moving to x-space, the far-field boundary is mapped onto a curved surface and $s = 0$ corresponds to $-\infty$ due to the logarithmic transformation.

Hence, we need to truncate the domain also in the near-field in order to make it finite. That is, we use $\tilde{R} = R \cap \{s \mid s_i \geq s_{\min}\}$ instead of R.

Since our aim is to exploit symmetries using GFT, we need to define a computational domain in x-space corresponding to a symmetry group. In two dimensions, we could for example use an equilateral triangle, a square, or any other shape enclosed by a polygon with sides of equal length. Finding the optimal shape and location of the computational domain is too large an issue to investigate within this study. Here, we restrict ourselves to hypercubes $[-L, L]^d$. Note that the domain does not have to be aligned with the coordinate axes. However, we use the aforementioned rotation and translation to position the domain for practical reasons.

A simple non-optimal algorithm to find a suitable hypercube is as follows:

- Compute the transformation matrix A in (16) through Cholesky factorization.
- Find a rotation Q such that the point $s = 4K(1, 1, \ldots, 1)^T$ is mapped onto $x = c(1, 1, \ldots, 1)^T$ for some $c \in \mathbb{R}$.
- Find the extreme points of the image of $\tilde{R}$ and let these define the corners in the region $\tilde{X}$.
- Compute the size of $\tilde{X}$ in each coordinate direction and let $2L$ be the maximum size. The computational domain X is formed by extending $\tilde{X}$ towards $-\infty$ in each coordinate until all sides have length $2L$.
- Finally, the computational domain is translated so that it is centered around the origin.

Figure 2 shows two examples of computational domains in the x-plane ($d = 2$) and their images in the s-plane. The volatility matrices in the examples are

$$\sigma^A = \begin{pmatrix} 0.30 & 0.05 \\ 0.05 & 0.30 \end{pmatrix} \quad \text{and} \quad \sigma^B = \begin{pmatrix} 0.30 & 0.05 \\ 0.01 & 0.25 \end{pmatrix}.$$

3.4 Discretization in time and approximation in space

We approximate the solution with a linear combination of RBFs centered at $N = N_I + N_B$ node points x^j. The N_I interior node points are distributed according to an equivariant pattern, whereas the N_B boundary node points are distributed uniformly along Γ_D. The approximation $p(\tau, x) \approx P(\tau, x)$ is

$$p(\tau, x) = \sum_{j=1}^{N} \lambda_j(\tau)\phi(\|x - x^j\|). \tag{20}$$

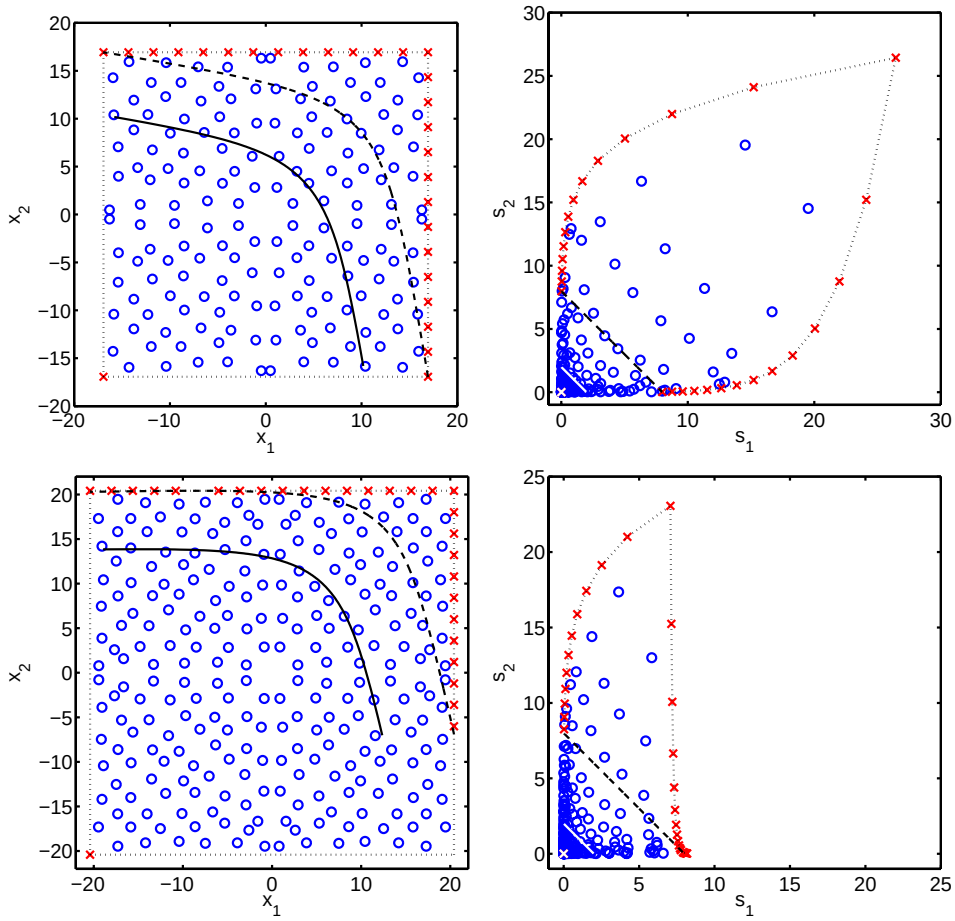

Fig. 2. The computational domain in the x-plane (left) and in the s-plane (right) for σ^A (top) and σ^B (bottom). Interior equivariant RBF node points are marked with circles and boundary node points with crosses. The solid lines (black to the left and white to the right) show where the discontinuous derivative of the contract function is located. The dashed lines show the far-field boundary of $\tilde{S}$.

Our solution approach is collocation of the equation at interior node points and of the boundary conditions at boundary node points. Therefore, the following vectors and their relations are of importance. Let $p_I(\tau) = (p(\tau, x^1), \ldots, p(\tau, x^{N_I}))^T$, $p_B(\tau) = (p(\tau, x^{N_I+1}), \ldots, p(\tau, x^N))^T$, $\lambda_I(\tau) = (\lambda_1(\tau), \ldots, \lambda_{N_I}(\tau))^T$, and $\lambda_B(\tau) = (\lambda_{N_I+1}(\tau), \ldots, \lambda_N(\tau))^T$, then

$$\begin{pmatrix} p_I(\tau) \\ p_B(\tau) \end{pmatrix} = \begin{pmatrix} A^{II} & A^{IB} \\ A^{BI} & A^{BB} \end{pmatrix} \begin{pmatrix} \lambda_I(\tau) \\ \lambda_B(\tau) \end{pmatrix}, \tag{21}$$

and

$$\Delta_x p_I(\tau) = \begin{pmatrix} B^{II} & B^{IB} \end{pmatrix} \begin{pmatrix} \lambda_I(\tau) \\ \lambda_B(\tau) \end{pmatrix}, \tag{22}$$

where the elements of the matrices A and B are $A_{ij} = \phi(\|x^i - x^j\|)$, $i, j = 1, \ldots, N$, and $B_{ij} = \Delta_x \phi(\|x^i - x^j\|)$, $i = 1, \ldots, N_I$, $j = 1, \ldots, N$.

We divide the time interval $[0, T]$ into M steps of length $k^n = \tau^n - \tau^{n-1}$, $n = 1, \ldots, M$. Let $p_I^n \approx p_I(\tau^n)$ and let $\omega_n = k^n / k^{n-1}$, $n = 2, \ldots, M$. Then the BDF-2 scheme [11, p. 401] applied to (19) takes the form

$$p_I^1 - p_I^0 = k^1 \Delta p_I^1, \tag{23}$$

$$p_I^n - \beta_1^n p_I^{n-1} + \beta_2^n p_I^{n-2} = \beta_0^n \Delta p_I^n, \quad n = 2, \ldots, M, \tag{24}$$

where

$$\beta_0^n = k^n \frac{1 + \omega_n}{1 + 2\omega_n}, \quad \beta_1^n = \frac{(1 + \omega_n)^2}{1 + 2\omega_n}, \quad \beta_2^n = \frac{\omega_n^2}{1 + 2\omega_n}. \tag{25}$$

The boundary conditions are enforced at each new time level through

$$p_B^n = f_B^n = (f(\tau^n, x^{N_I+1}), \ldots, f(\tau^n, x^N))^T, \quad n = 1, \ldots, M. \tag{26}$$

Inserting (21) and (22) into (24) and (26) gives the system of equations to solve for each time step

$$\begin{pmatrix} A^{II} - \beta_0^n B^{II} & A^{IB} - \beta_0^n B^{IB} \\ A^{BI} & A^{BB} \end{pmatrix} \begin{pmatrix} \lambda_I^n \\ \lambda_B^n \end{pmatrix} = \begin{pmatrix} f_I^n \\ f_B^n \end{pmatrix}, \tag{27}$$

where the superscript n indicates approximation at time τ^n, and

$$f_I^n = \begin{pmatrix} A^{II} & A^{IB} \end{pmatrix} \left[\beta_1^n \begin{pmatrix} \lambda_I^{n-1} \\ \lambda_B^{n-1} \end{pmatrix} - \beta_2^n \begin{pmatrix} \lambda_I^{n-2} \\ \lambda_B^{n-2} \end{pmatrix} \right]. \tag{28}$$

Note that we have chosen to work with the RBF coefficients $\lambda_{i,b}^n$. Another approach is to directly work with $p_{i,b}^n$. However, in this context, where we want to use the symmetries and where we do not need the solution at each time step, the coefficient approach saves some matrix operations.

3.5 Exploiting the partial symmetry

The interior node points are placed in such a way that the matrix blocks A^{II} and B^{II} are equivariant. Therefore, all operations involving these blocks can

be performed more efficiently using the GFT. Since the number of interior node points is large compared with the number of boundary points, these operations are also the most time consuming. Table 2 shows the theoretical gain in computational cost and in memory requirements in two and three dimensions for the symmetry groups of the square and the cube. Note that we have not counted the cost for performing the GFTs.

The first opportunity to use the GFT arises in the computation of the right hand side (28), where the multiplication with A^{II} can be performed in transform space.

Then we have the solution of system (27). In order to have a constant coefficient matrix, the time steps are chosen in such a way that $\beta_0^n \equiv \beta_0 = k^1$. This is further explained in Section 4.2. Let $C^{II} = A^{II} - \beta_0 B^{II}$ and $C^{IB} = A^{IB} - \beta_0 B^{IB}$. We perform block Gaussian elimination to obtain

$$\begin{pmatrix} C^{II} & C^{IB} \\ 0 & C^{BB} \end{pmatrix} \begin{pmatrix} \lambda_I^n \\ \lambda_B^n \end{pmatrix} = \begin{pmatrix} f_I^n \\ f_B^n - A^{BI}(C^{II})^{-1} f_I^n \end{pmatrix},$$

where $C^{BB} = A^{BB} - A^{BI}(C^{II})^{-1}C^{IB}$. The solution can then be obtained through the Schur complement algorithm:

(1) Solve $C^{II} w_I = f_I^n$.
(2) Solve $C^{BB}\lambda_B^n = f_B^n - A^{BI} w_I$.
(3) Solve $C^{II} v_I = C^{IB}\lambda_B^n$.
(4) Compute $\lambda_I^n = w_I - v_I$.

Steps 1 and 3 involve solving systems with the equivariant coefficient matrix C^{II}. Furthermore, we need to form C^{BB}, where the first step consists of solving a system with matrix C^{II} and N_B right hand sides. Since $\hat{C}^{II}$ is block-diagonal, LU-factoring the transformed matrix and solving in transform space leads to substantial savings.

Table 2
The gain in computational cost and memory when using the GFT in two and three dimensions.

Operation	Gain in 2D	Gain in 3D
$\mathcal{O}(N^2)$	8	48
$\mathcal{O}(N^3)$	128/3	864
Memory	8	48

4 Algorithmic details

In this section, we describe special cases and details that are not covered by the general description of the algorithm in the previous sections.

4.1 Hypercube symmetries in d dimensions

The symmetry group of a square and its irreducible representations were discussed extensively in Section 2. The symmetry group of the cube (in three dimensions) is the direct product $S_2 \times S_4$, discussed e.g. in [25], where S_k denotes the group of all permutations of k symbols. This means that all elements of $S_2 \times S_4$ as well as the group operation can easily be found from the subgroups S_2 and S_4. For example, the number of elements are $|S_2||S_4| = 2 \cdot 4! = 48$. Moreover, the representations of $S_2 \times S_4$ are constructed as tensor products of the representations of S_2 and S_4, respectively. The latter fact can be used to derive a fast GFT [14]. See [1] for a list of the representations of the cube and the GFT algorithm that we use.

Serre [25] also notes that another way of expressing the symmetry group of the cube is as a semidirect product of S_3 with S_2^3, where S_2^k denotes a direct product $S_2 \times \cdots \times S_2$ (k times). This means that many properties of the cube symmetry group can be expressed using these subgroups, for example is its size $|S_3||S_2|^3 = 3! \cdot 2^3 = 48$. An advantage with this description is that is easy to generalize to d dimensions. S_3 corresponds to permutations of the coordinates and S_2^3 corresponds to the reflections

$$(x_1, x_2, x_3)^T \mapsto (\pm x_1, \pm x_2, \pm x_3)^T.$$

The cube is clearly invariant under these transformations. Generalizing to d dimensions, we find that the symmetry group of a hypercube can be expressed as a semidirect product of S_d (permutations of the coordinates) with S_2^d (reflections of the coordinates). This group has $d!2^d$ elements.

Regarding its representations, it is possible to use the representations of the subgroups of a semidirect product to find representations of the product. However, it is more technical than in the case of a direct product. Representations of the symmetric groups S_k are well known, see e.g. [23], but we leave as future work to deduce the actual representations and the corresponding GFT algorithms of the hypercube symmetry groups.

If adaptive time-stepping is used with the BDF-2 method, the coefficient matrix in (27) changes with each time-step. Furthermore, even if a constant time-step is used, the coefficient matrix for the first time-step is different from the coefficient matrix for the subsequent time-steps. In the application considered here, time adaptivity is not really needed, but it is important to minimize the number of costly matrix factorizations. Therefore, we use the variable step size to keep the system matrix constant. We can derive a recursion formula from the condition $\beta_0^n = \beta_0^{n-1}$ or

$$k^n \frac{1+\omega_n}{1+2\omega_n} = k^{n-1} \frac{1+\omega_{n-1}}{1+2\omega_{n-1}} \quad \Leftrightarrow \quad \frac{\omega_n^2+\omega_n}{1+2\omega_n} = \frac{1+\omega_{n-1}}{1+2\omega_{n-1}} \equiv c_{n-1},$$

simplifying and solving for the positive root leads to

$$\begin{aligned}
\omega_n &= c_{n-1} - \frac{1}{2} + \frac{\sqrt{4c_{n-1}^2+1}}{2}, \quad n = 2, \ldots, M, \\
c_n &= \frac{1+\omega_n}{1+2\omega_n}, \quad n = 2, \ldots, M, \\
c_1 &= 1,
\end{aligned}$$

where the initial condition comes from (23). Given the length T of the time interval, we then choose the initial step size k^1, so that $\sum_{i=1}^{M} k^n = k^1(1 + \sum_{i=2}^{M} \prod_{j=2}^{i} \omega_n) = T$.

Figure 3 shows the time-step series normalized so that the step size converges to $k = 1$. As can be seen, the sequence rapidly converges toward a constant

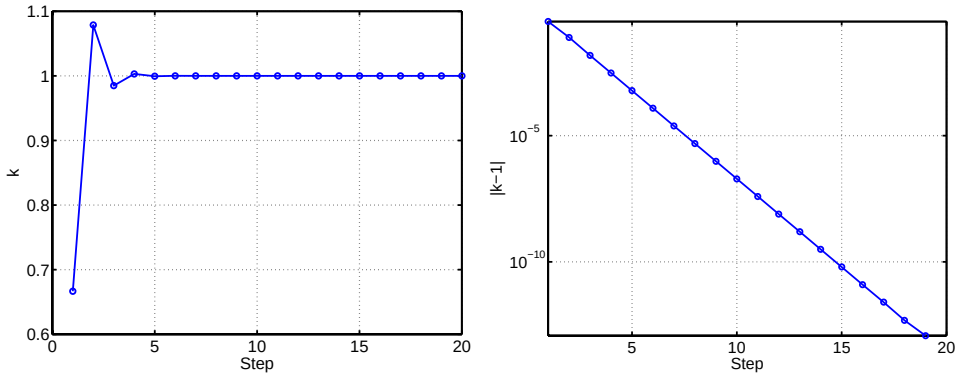

Fig. 3. The time-steps k^n that yield $\beta_0^n \equiv \beta_0$ as a function of the step number n.

step size. The variations are all well within the stability limit $\omega_n < 1 + \sqrt{2}$ [11, p. 405].

4.3 The two initial steps

The right hand sides of the first two time steps involves the initial data and are computed in a slightly different way. From (23), (24), and (28), we get

$$f_I^1 = p_I^0, \tag{29}$$

$$f_I^2 = \beta_1^n \begin{pmatrix} A^{II} & A^{IB} \end{pmatrix} \begin{pmatrix} \lambda_I^1 \\ \lambda_B^1 \end{pmatrix} - \beta_2^n \begin{pmatrix} p_I^0 \\ 0 \end{pmatrix}. \tag{30}$$

4.4 Evaluating the solution

Assuming we want to know the solution at time $\tau = T$ or equivalently $t = 0$, we can evaluate it at points y^i, $i = 1, \ldots, N_e$ using the RBF coefficients from the final time step through

$$p_y^M = E_I \lambda_I^M + E_B \lambda_B^M,$$

where $E_{ij} = \phi(\|y^i - x^j\|)$ with $j = 1, \ldots, N_I$ and $j = N_I + 1, \ldots, N$ respectively.

4.5 Error norms

When we compare the approximate solution $p(T, x)$ with an exact solution or a reference solution $P(T, x)$, we use either a relative maximum norm

$$E_\infty = \frac{\max_{x \in \Omega} |p(T, x) - P(T, x)|}{\max_{x \in \Omega} |P(T, x|},$$

or a weighted integral norm defined as

$$E_w = \int_\Omega w(x) |p(T, x) - P(T, x)| \, dx, \tag{31}$$

where the weight function is chosen as a product of d Gaussian functions, centered in the region of interest and normalized to have total weight 1. In two dimensions, we use $w(x) \propto \exp(-4(x_1 + x_2 - 2K)^2) \exp(-(x_1 - x_2)^2)$. The reason we use the integral norm is that the error is usually larger at the boundaries, but we are interested in the solution in the interior close to the

break-even line of the contract function. For further discussions about this norm, see [21].

5 Numerical experiments

In this section we present some numerical results designed to evaluate the performance of the RBF-GFT approach. In Section 5.1, we show results for the heat equation with known solution, and in section 5.2, we solve the option pricing problem in two and three dimensions.

We have chosen to use multiquadric RBFs in all experiments, i.e.,

$$\phi(r) = \sqrt{1 + \varepsilon^2 r^2},$$

where ε is called the shape parameter. A small shape parameter makes basis functions flatter. This has a significant influence on the approximation [16]. Other choices of infinitely smooth RBFs should work equally well, but this has not been explored here.

5.1 Validation tests for the heat equation with known solution

To get an idea of the effects of the method parameters, we present some tests for the heat equation (18), but with the known solution below substituted for the initial and boundary conditions,

$$u(\tau, x) = \sum_{k=1}^{d} \exp(-k^2 t) \sin(k x_k).$$

The computational domain is $\Omega = [-1, 1]^d$, the boundary Γ_D is taken as the union of the four sides of the square or the six sides of the cube, and $T = 1$. Except for the experiment where the number of time steps is varied, $M = 100$ is used.

Figure 4 shows that we get the right accuracy in time. Computations with and without the GFT are shown in both subfigures, but the only case where there is a visible difference is for $\varepsilon = 3$ in the left part of the figure. Our interpretation of this difference (based also on results that are not shown here) is that when the time-step is so small that it is of the same order as the error induced by the ill-conditioning of the RBF-matrices the time-stepping scheme breaks down. The condition number of an RBF matrix grows very rapidly with

decreasing ε and it also grows with N [24,16]. The conditioning of the matrix blocks after the GFT has been applied is approximately the same as for the original matrix [2,12]. However, the decreased size of the systems nevertheless improves the situation so that when the GFT is used, the numerical problems occur later. That is, we can solve a larger range of problems and hence achieve higher accuracy than with the full matrix method.

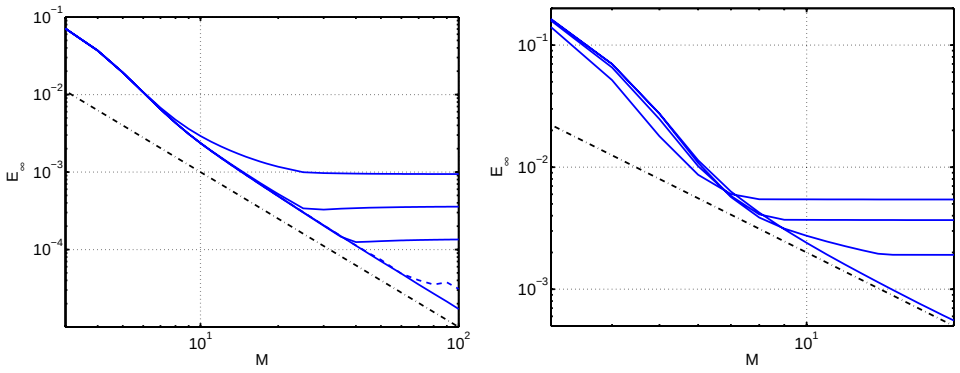

Fig. 4. The maximum error as a function of the number of time steps for $N = 692$ in two dimensions (left) and $N = 1082$ in three dimensions (right). The shape parameter values are $\varepsilon = 20, 10, 5, 3$ (left) from top to bottom, and $\varepsilon = 10, 5, 3, 1$ (right). Results with the GFT (solid) and without the GFT (dashed) are shown. The dash-dot lines are reference lines with slope 2.

Figure 5 shows how important the shape parameter is for the error. Usually there is an optimal, problem dependent, non-zero value of the shape parameter that gives the best result [15,9]. Here however, it is clear from the error plots that the ill-conditioning prevents us from reaching this optimum, and the best we can do is compute at the smallest ε where the method is stable. Again we can clearly see that the GFT actually improves the accuracy.

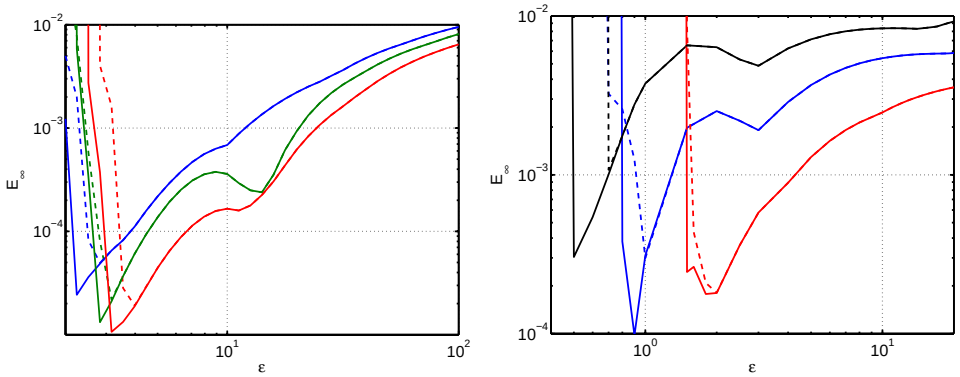

Fig. 5. The maximum error as a function of the shape parameter for, from top to bottom, $N = 512, 692, 972$ in two dimensions (left), and $N = 488, 1082, 2858$ in three dimensions (right), with the GFT (solid lines) and without the GFT (dashed lines).

Figure 6 shows the convergence when the shape parameter is fixed and the number of RBFs is increased. Note that the convergence rate is higher for smaller values of ε. In three dimensions (the right subfigure) we run out of memory and we can confirm that it really pays off to use a small shape parameter. Even though we are hit by ill-conditioning already at $N \approx 1000$, we get a more accurate result by using $\varepsilon = 1$ there than by using a larger ε with $N = 8000$.

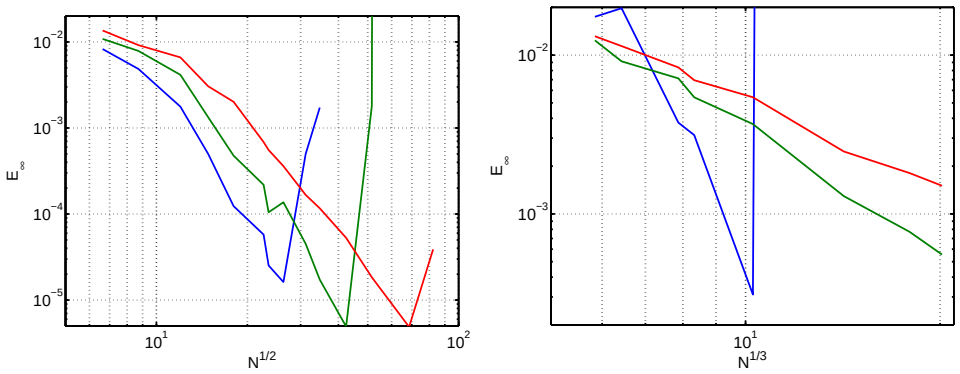

Fig. 6. The maximum error as a function of the approximate number of RBF points in one coordinate direction, in two dimensions for $\varepsilon = 3, 5, 10$ (left) and in three dimensions for $\varepsilon = 1, 5, 10$ (right). Only computations with the GFT are shown.

5.2 Performance tests for the Black–Scholes equation

Here we test the efficiency with respect to memory and computational time for the RBF-GFT method applied to the option pricing problem. The problem parameters used are exercise time $T = 1$, strike price $K = 1$, risk free interest rate $r = 0.05$, and volatility

$$\sigma^{2\mathrm{D}} = \begin{pmatrix} 0.30 & 0.05 \\ 0.05 & 0.30 \end{pmatrix} \quad \text{and} \quad \sigma^{3\mathrm{D}} = \begin{pmatrix} 0.30 & 0.05 & 0 \\ 0.05 & 0.30 & 0.05 \\ 0 & 0.05 & 0.30 \end{pmatrix}.$$

The minimal s-coordinate value used for positioning the domain was $s_{\min} = 0.02$. Note however that after the transformation, the point closest to the origin is actually closer than $s_{\min}$. For experiments where the number of time steps is fixed, $M = 15$. With the range of N that we use and for this problem, this is enough to reach the best possible accuracy. The shape parameter was optimized for each N. In two dimensions, we compare three different methods:

RBF-GFT The method derived in this paper.

RBF-E RBF approximation directly applied to the Black–Scholes equation (10) with the same **E**quivariant node points as the RBF-GFT method uses.

RBF-A Direct RBF approximation with an **A**daptive node point placement on a triangular domain [21].

Here, we do not have access to an exact solution, but we use a reference solution computed by the adaptive finite difference method described in [20]. Figures 7 and 8 shows that RBF-A is outstanding in 2D because it is so much more accurate. RBF-GFT and RBF-E reach approximately the same accuracy for equal N, but we are able to compute further with RBF-GFT. This means that the transformation we do to symmetrize the equation does not destroy the properties of the problem. With respect to time and memory, the RBF-GFT method is, as it should be, better than the RBF-E method.

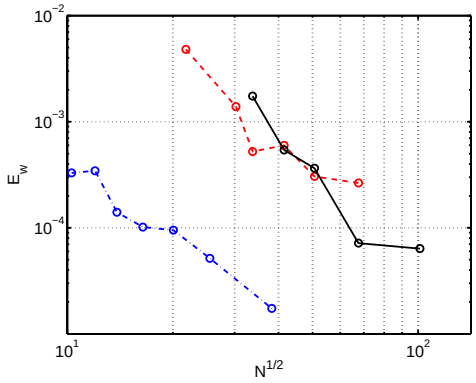

Fig. 7. The error as a function of N for RBF-GFT (solid), RBF-E (dashed), and RBF-A (dash-dot).

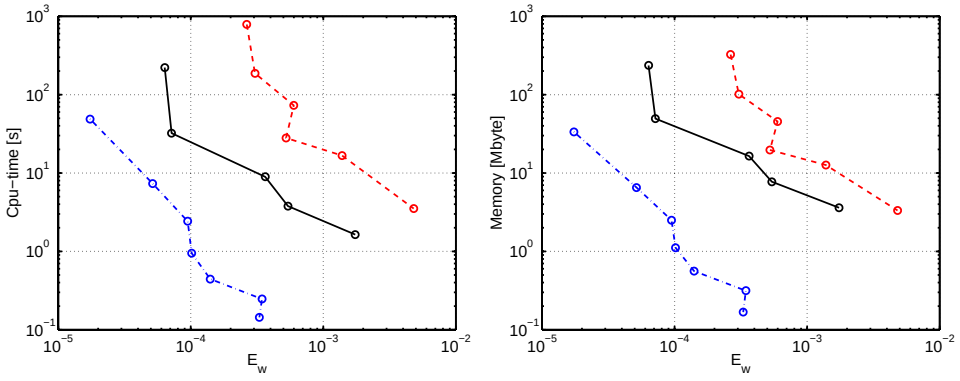

Fig. 8. The computational time and the memory requirements as a function of the desired accuracy for RBF-GFT (solid), RBF-E (dashed), and RBF-A (dash-dot).

In three dimensions we do not have access to a reference solution, so we only show the performance for the RBF-GFT method compared with solving the same problem without the GFT. In Figure 9, we see that we gain in memory already for very small problem sizes. The break-even point for the cpu-time

is at $N \approx 1000$. The reason that we do not see an immediate gain is that the number of boundary points is a large part of the total for small N, and only the interior points are handled with the GFT.

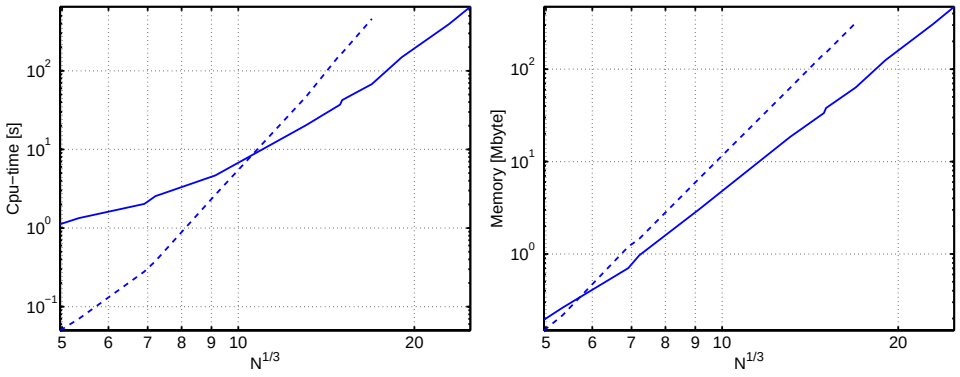

Fig. 9. The computational time and the memory requirements as a function of the approximate number of node points along one coordinate direction for the three-dimensional problem.

Figure 10 shows the gain when the GFT is employed. The gain is computed as the time or memory for the RBF method without the GFT divided by the time or memory for the RBF-GFT method. The theoretical maximum memory gain in two dimensions is 8, and we see that we are approaching the limit as N grows. For the three-dimensional problem we cannot compute for large enough problems to see the limit, but the gain is larger than in two dimensions for the same resolution. For the time gain the situation is similar. We gain more in three dimensions than in two dimensions and we do not see the limit.

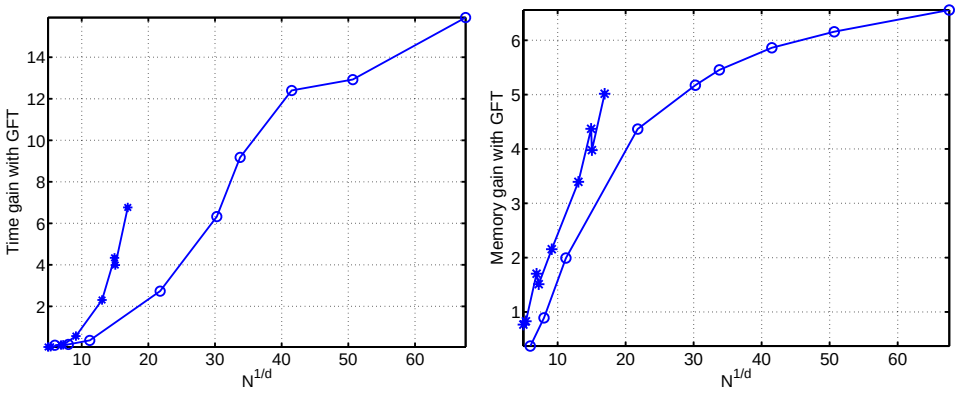

Fig. 10. The gain in memory requirements and computational time when using the GFT for the two-dimensional problem (∘) and the three-dimensional problem (*).

6 Discussion

First, we note that we managed to apply the GFT to a problem that at first appearance was not well suited for this. The proposed transformation and also the (non-optimal) method to choose the computational domain can be applied for any number of dimensions. The only components that we have not described in full generality are the specific representations needed for the GFT algorithm in higher dimensions. However, we believe this is possible to do.

The gain in both memory requirements and computational time with the RBF-GFT method compared with a standard RBF method is substantial and increases with the number of dimensions. In our computations, time was not a problem, but we ran out of memory. However, we only used one standard computer. With a high-performance computer or a cluster of computers and a parallel implementation, we could do much more. Due to the block-diagonal structure of the transformed matrices, the algorithm should respond well to a large grain parallelization [28].

In two dimensions, we saw that the RBF-A method performed best due to its superior accuracy for a given number of node points. However, it is not clear to us if an adaptive method or an RBF-GFT method will be better in more dimensions, since the gain from using the GFT increases rapidly with the dimension. Furthermore, it might be possible to improve also the equivariant point distribution by for example aligning points with the discontinuous derivative in the initial condition.

Finally, a positive side effect from using the GFT is that the accuracy is improved for small ε since the effect of the ill-conditioning is somewhat lessened by the reduction in size when the matrices are block-diagonalized.

Acknowledgment

The first author would like to thank Gail Gutierrez of Universidad Pontificia Bolivariana, Colombia for persistent questions about the use of the BDF-2 method, which challenged me to improve the treatment of the boundary conditions and also made me think of the variable time-step scheme.

References

[1] K. Åhlander, H. Munthe-Kaas, Applications of the generalized Fourier transform in numerical linear algebra, BIT 45 (2005) 819–850.

[2] K. Åhlander, H. Munthe-Kaas, Eigenvalues for equivariant matrices, J. Comput. Appl. Math. 192 (2006) 89–99.

[3] E. L. Allgower, K. Georg, R. Miranda, J. Tausch, Numerical exploitation of equivariance, Zeitschrift für Angewandte Mathematik und Mechanik 78 (1998) 185–201.

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

[5] M. Bonnet, Exploiting partial or complete geometrical symmetry in 3D symmetric Galerkin indirect BEM formulations, Intl. J. Numer. Meth. Engng 57 (2003) 1053–1083.

[6] M. Buhmann, N. Dyn, Spectral convergence of multiquadric interpolation, Proc. Edinburgh Math. Soc. (2) 36 (2) (1993) 319–333.

[7] H.-J. Bungartz, M. Griebel, Sparse grids, Acta Numer. 13 (2004) 147–269.

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

[9] B. Fornberg, G. Wright, Stable computation of multiquadric interpolants for all values of the shape parameter, Comput. Math. Appl. 48 (2004) 853–867.

[10] P. Glasserman, Monte Carlo methods in financial engineering, Vol. 53 of Applications of Mathematics (New York), Springer-Verlag, New York, 2004, Stochastic Modelling and Applied Probability.

[11] E. Hairer, S. P. Nørsett, G. Wanner, Solving ordinary differential equations. I, 2nd Edition, Vol. 8 of Springer Series in Computational Mathematics, Springer-Verlag, Berlin, 1993, Nonstiff problems.

[12] A. Hall, Pricing financial derivatives using radial basis functions and the generalized Fourier transform, UPTEC Report IT 05 036, School of Engineering, Uppsala Univ., Uppsala, Sweden (2005).

[13] Y.-C. Hon, X.-Z. Mao, A radial basis function method for solving options pricing models, J. Financial Engineering 8 (1999) 31–49.

[14] M. G. Karpovsky, Fast Fourier transforms on finite non-abelian groups, IEEE Trans. Computers C-26 (10) (1977) 1028–1030.

[15] E. Larsson, B. Fornberg, A numerical study of some radial basis function based solution methods for elliptic PDEs, Comput. Math. Appl. 46 (2003) 891–902.

[16] E. Larsson, B. Fornberg, Theoretical and computational aspects of multivariate interpolation with increasingly flat radial basis functions, Comput. Math. Appl. 49 (2005) 103–130.

[17] L. Ling, E. J. Kansa, Preconditioning for radial basis functions with domain decomposition methods, Math. Comput. Modelling 40 (2004) 1413–1427.

[18] W. R. Madych, Miscellaneous error bounds for multiquadric and related interpolators, Comput. Math. Appl. 24 (12) (1992) 121–138.

[19] M. D. Marcozzi, S. Choi, C. S. Chen, On the use of boundary conditions for variational formulations arising in financial mathematics, Appl. Math. Comput. 124 (2001) 197–214.

[20] J. Persson, L. von Sydow, Pricing European multi-asset options using a space-time adaptive FD-method, Tech. Rep. 2003-059, Dept. of Information Technology, Uppsala Univ., Uppsala, Sweden, to appear in Computing and Visualization in Science (2003).

[21] U. Pettersson, E. Larsson, G. Marcusson, J. Persson, Improved radial basis function methods for multi-dimensional option pricing, Tech. Rep. 2006-028, Dept. of Information Technology, Uppsala Univ., Uppsala, Sweden (2006).

[22] G. Roussos, B. J. C. Baxter, Rapid evaluation of radial basis functions, J. Comput. Appl. Math. 180 (2005) 51–70.

[23] B. E. Sagan, The symmetric group. Representations, combinatorial algorithms, and symmetric functions, 2nd Edition, Vol. 203 of Graduate Texts in Mathematics, Springer-Verlag, New York, 2001.

[24] R. Schaback, Multivariate interpolation by polynomials and radial basis functions, Constr. Approx. 21 (3) (2005) 293–317.

[25] J.-P. Serre, Linear representations of finite groups, Springer-Verlag, New York, 1977, translated from the second French edition by Leonard L. Scott, Graduate Texts in Mathematics, Vol. 42.

[26] R. Seydel, Tools for computational finance, 2nd Edition, Universitext, Springer-Verlag, Berlin, 2004.

[27] Z. Wu, Y.-C. Hon, Convergence error estimate in solving free boundary diffusion problem by radial basis functions method, Engrg. Anal. Bound. Elem. 27 (2003) 73–79.

[28] A. Yamba Yamba, K. Åhlander, M. Ljungberg, Designing for geometrical symmetry exploitation, Tech. Rep. 2006-017, Dept. of Information Technology, Uppsala Univ., Uppsala, Sweden (2006).