Improved radial basis function methods for multi-dimensional option pricing

Ulrika Pettersson [a,1], Elisabeth Larsson [a,2,*], Gunnar Marcusson [b] and Jonas Persson [a,1]

[a] *Address: Department of Information Technology, Uppsala University, Box 337, SE-751 05 Uppsala, Sweden*

[b] *Address: Försäkringsmatematik, Box 5148, SE-102 43 Stockholm, Sweden.*

Abstract

In this paper, we have derived a radial basis function (RBF) based method for the pricing of financial contracts by solving the Black–Scholes partial differential equation. As an example of a financial contract that can be priced with this method we have chosen the multi-dimensional European basket call option. We have shown numerically that our scheme is second order accurate in time and spectrally accurate in space for constant shape parameter. For other, non-optimal choices of shape parameter values, the resulting convergence rate is algebraic. We propose an adaptive node point placement that improves the accuracy compared with a uniform distribution. Compared with an adaptive finite difference method, the RBF method is 20–40 times faster in one and two space dimensions and has approximately the same memory requirements.

Key words: Radial basis function, Black–Scholes equation, multi-dimensional, boundary conditions.

1 Introduction

The financial markets are becoming more and more complex, with trading not only of stocks, but also of numerous types of financial derivatives. The market requires updated information about the values of these derivatives

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

[1] Funded by FMB, the Swedish Graduate School in Mathematics and Computing.

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

every second of the day. This leads to a huge demand for fast and accurate computer simulations.

In this study we consider the problem of pricing financial contracts on several underlying assets. These contracts are receiving more and more interest as the demand of complex derivatives from the customers and the speed of computers have increased over the years. We have chosen to use a European basket option as an example. This is a rather simple contract but works well as an indicator of the usefulness of our method.

One way of pricing financial contracts is to solve the Black–Scholes equation [1], a partial differential equation (PDE) in which the number of spatial dimensions is determined by the number of underlying assets. When the number of dimensions grows, this becomes very computationally demanding. Thus, it is necessary to use fast and memory efficient algorithms.

Other methods to price high-dimensional contracts is, e.g., Monte Carlo methods, "sparse grids", and finite difference methods. Monte Carlo methods have the advantage of scaling linearly with the number of dimensions, but have the drawback of converging very slowly. There are several ways of speeding up the convergence, e.g., different variance reduction and quasi random sequence techniques. A good reference for Monte Carlo algorithms and theory is [6]. Sparse grids is an approximation technique rediscovered in the 1990's. By combining several grids with different step sizes, a small number of grid points can be used to achieve an accurate approximation and keep the memory requirements low. The reference [3] gives an introduction to sparse grids with applications. Finite difference methods are generally well known, but for details about the method we have used here for comparisons, we refer to Section 5.

Here, we consider RBF approximation as a potentially effective approach for solving the multi-dimensional Black–Scholes equation. A typical RBF approximant has the form

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

where $\phi(r)$ is the RBF, $\vec{x}_j$, $j = 1, \ldots, N$ are center points, and ε is a shape parameter. A small value of ε leads to flatter RBFs. The shape parameter is an important method parameter, with a significant effect on the accuracy of the method. With infinitely smooth RBFs the method can be spectrally accurate [17,2], meaning that the required number of node points for a certain desired accuracy is potentially very small. Since the method only needs pairwise distances between points, it is meshfree. Therefore, it is easy to use in higher dimensions and it also allows for adaptive node placement.

Option pricing using RBFs has been explored in one dimension for European and American options by Hon et al. [10,22] and in both one and two dimensions by Fasshauer et al. [4] and Marcozzi et al. [18] with promising results. Hon has also applied a quasi-radial basis function method to option pricing in one dimension [9].

The contribution of this paper is a thorough numerical study of the effects of the method parameters on the accuracy and performance of the method, providing some insights regarding the possibilities and limitations of RBF methods. We look at sample problems in one and two dimensions and we also compare the results of the RBF method with those of an adaptive finite difference scheme [20]. Furthermore, we discuss boundary conditions both from a theoretical and an implementational viewpoint.

The outline of the paper is as follows. In Section 2, we present the sample problems and boundary conditions. Then, in Section 3, we derive the space approximation and time discretization of the problem. Section 4 contains numerical experiments for the RBF method and Section 5 shows the results of the comparison with the adaptive finite difference method. Finally, Section 6 gives some conclusions.

2 The multi-dimensional Black–Scholes problem

2.1 The Black–Scholes equation

The Black–Scholes equation is a time-dependent linear PDE, in its original formulation posed as a final value problem. Here we use a transformed version of the PDE. Time is reversed to make standard texts on time-integration for PDEs applicable, and all variables have been scaled to be dimensionless. The details of the transformation can be found in [20]. The transformed problem reads

$$\begin{cases} \dfrac{\partial}{\partial \hat{t}} P(\hat{t}, \vec{x}) = \mathcal{L}P(\hat{t}, \vec{x}), \quad \hat{t} \in \mathbb{R}_+, \quad \vec{x} \in \mathbb{R}_+^d, \\ P(0, \vec{x}) \quad\quad = \Phi(\vec{x}), \quad \vec{x} \in \mathbb{R}_+^d, \end{cases} \tag{1}$$

where

$$\mathcal{L}P = 2\bar{r} \sum_{i=1}^{d} x_i \frac{\partial P}{\partial x_i} + \sum_{i,j=1}^{d} \left[\bar{\sigma}\bar{\sigma}^T\right]_{ij} x_i x_j \frac{\partial^2 P}{\partial x_i \partial x_j} - 2\bar{r}P, \tag{2}$$

where $P(\hat{t}, \vec{x})$ is the value of the option at time $\hat{t}$ when the underlying assets have the values given by $\vec{x}$. Furthermore, the coefficient $\bar{r}$ is the scaled short interest rate, $\bar{\sigma}$ is the scaled volatility and d denotes the number of underlying assets and thus the number of spatial dimensions of the problem. An example of a contract function for a European basket call option is the average option

$$\Phi(\vec{x}) = \max\left(\frac{1}{d}\sum_{i=1}^{d} x_i - \bar{K},\, 0\right), \tag{3}$$

where the scaled strike price in our case is $\bar{K} = 1$. The weights could also be different from $1/d$, but that would just be another scaling of the variables. This type of contract function is considered in e.g. [21].

2.2 Boundary conditions for the finite difference method

As mentioned previously, we use the adaptive finite difference method derived in [20] for reference solutions and for comparisons. For a finite difference discretization of the Black–Scholes problem, (numerical) boundary conditions are needed at all parts of the boundary. This implementation employs

$$\frac{\partial^2 P(\vec{x}, \hat{t})}{\partial n^2} = 0, \tag{4}$$

where $\partial/\partial n$ indicates differentiation in the direction normal to the boundary. This is an approximation discussed and used in [21]. It has also been successfully used in [20] and [16]. There are of course other possible boundary conditions, but this choice has proven to work very well for this problem. The condition (4) is approximated by a second order discretization of the second derivative and can be considered as a linear extrapolation of the solution up to the boundary. For all interior grid points a second order discretization of the PDE is used.

2.3 Boundary conditions for the RBF method

The condition (4) does not work well with an RBF approximation method. One reason is that it does not imply linearity in a region near the boundary, since the condition is enforced only at the boundary and the infinitely smooth RBFs that we use are not in themselves linear.

In [11], Janson and Tysk show that the problem we consider here is actually well posed without boundary conditions as long as the growth at infinity is

restricted. Therefore, we only use near- and far-field boundary conditions. This means that no boundary conditions are employed at boundaries of the type $\Gamma_i = \{\vec{x} \mid \vec{x} \in \mathbb{R}^d_+, \quad \vec{x} \neq \vec{0}, \quad x_i = 0\}$, $i = 1, \ldots, d$.

The near-field boundary can be seen as the single point $\vec{x} = \vec{0}$, and there we enforce

$$P(\hat{t}, \vec{0}) = 0. \tag{5}$$

At the far-field boundary, which we have not yet defined, we use the asymptotic solution

$$P(\hat{t}, \vec{x}) \to \frac{1}{d} \sum_{i=1}^{d} x_i - \bar{K} e^{-2\bar{r}\hat{t}}, \quad \|\vec{x}\| \to \infty. \tag{6}$$

A different approach to boundary conditions for the RBF method was used in [4]. There $d-1$ dimensional problems are solved at the parts of the boundary where we do not enforce any boundary conditions at all. Our arguments against doing this is (i) errors in the computations in the lower dimensions are transferred to and possibly enlarged in the higher dimensions, (ii) with the need to recursively solve PDEs in all dimensions up to d, it becomes more difficult to implement the algorithm, and (iii) since the PDE at the boundaries collapses into lower dimensional versions, time stepping the boundary points along with rest should automatically provide the correct behavior.

2.4 Computational domain

The problem is defined on $\mathbb{R}^d_+$, but for computational reasons we need to restrict the problem to a finite domain. For the finite difference method the domain is $[0, a_1] \times [0, a_2] \times \cdots \times [0, a_d]$, in order to easily construct the structured grid that is needed. However, the RBF method is meshfree, which gives us the opportunity to choose the artifical far-field boundary as we like. With the contract function (3), it makes sense to use a boundary surface of the type $\sum_{i=1}^{d} x_i = C$, where the constant C is chosen to bring the surface far enough from the origin for the far-field solution (6) to be an accurate approximation.

2.5 Measuring the error

When measuring the error in the approximate solutions it is important to remember the real-life background of the problems we are solving. Firstly, when we solve the Black-Scholes equation, we want to know the price today

($\hat{t} = T$) of an option with exercise time T years from today. That is, the error function is given by

$$E(\vec{x}) = P(T, \vec{x}) - u(T, \vec{x}). \tag{7}$$

Secondly, in option trading the region of most interest is when the mean of the stock prices is close to the strike price. Typically, the probability for a stock to default or to be very far from the strike price is small. Therefore we define the region of interest Ω_i to be all $\vec{x}$ for which

$$\frac{1}{d}\sum_{i=1}^{d} x_i \in \left[\frac{\bar{K}}{3}, \frac{5\bar{K}}{3}\right]$$

holds, and propose a financial error norm given by

$$E_f = \max_{\vec{x}\in\Omega_i} |E(\vec{x})|. \tag{8}$$

The region of interest Ω_i is depicted in Figure 1(a) for a one-dimensional problem and in Figure 2(a) for a two-dimensional problem.

We have also used a weighted integral norm defined as

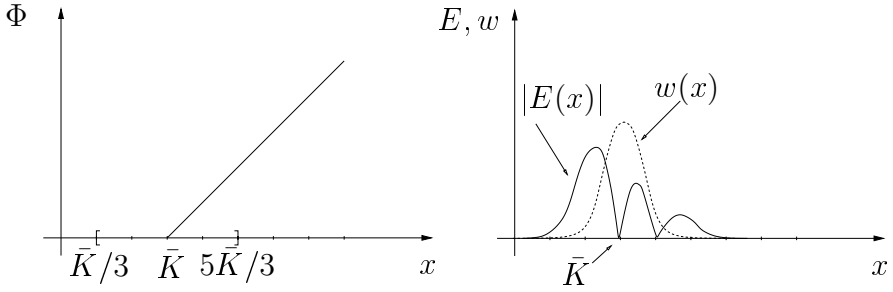
$$E_w = \int_{\Omega} w(\vec{x})|E(\vec{x})|\, d\vec{x}, \tag{9}$$

where Ω is the whole computational domain. The weight function is chosen as a product of d Gaussian functions, centered in the region of interest and with $\int_\Omega w(\vec{x})\, d\vec{x} = 1$. In one dimension, we use $w(\vec{x}) \propto \exp(-5(x - \bar{K})^2)$ and in two dimensions, we use $w(\vec{x}) \propto \exp(-4(x_1 + x_2 - 2\bar{K})^2)\exp(-(x_1 - x_2)^2)$. The weight functions in one and two dimensions are shown in Figures 1(b) and 2(b), respectively. The idea can be extended to several dimensions and other contract functions by changing the function $w(\vec{x})$ accordingly.

3 RBF approximation and time-stepping

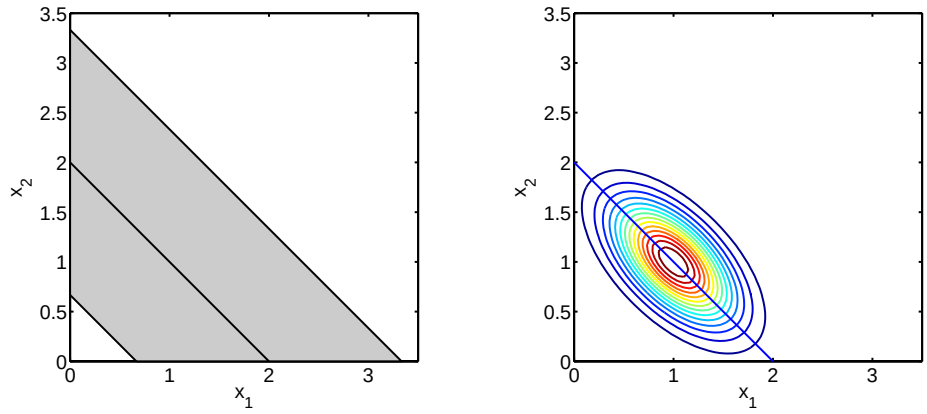
We approximate the solution of (1) with a time-dependent linear combination of RBFs centered at the node points $\vec{x}_k$, $k = 1, \ldots, N$,

$$u(\hat{t}, \vec{x}) = \sum_{k=1}^{N} \lambda_k(\hat{t})\phi(\varepsilon\|\vec{x} - \vec{x}_k\|) = \sum_{k=1}^{N} \lambda_k(\hat{t})\phi_k(\vec{x}), \tag{10}$$



(a) The contract function and Ω_i. (b) The weight function (dashed) and $|E(x)|$ (solid).

Fig. 1. Illustrations of the error norms in one dimension.



(a) The region Ω_i. (b) Contours of $w(\vec{x})$.

Fig. 2. Illustrations concerning the error norms in two dimensions. The line $s_1 + s_2 = 2$ in both subfigures is where the contract function has a discontinuous first derivative.

where $\phi(r)$ is the radial basis function, ε is the shape parameter, and $\lambda_k(\hat{t})$ are coefficients to determine.

Our method of determining these coefficients is collocation at the node points. For interior node points $\vec{x}_k$, $k = 1, \ldots, N_i$ we use equation (1) and for node points at the near or far field boundaries, $\vec{x}_k$, $k = N_i + 1, \ldots, N$, we enforce (5) or (6), respectively. Let $\vec{u}_i(\hat{t}) = (u(\hat{t}, \vec{x}_1), \ldots, u(\hat{t}, \vec{x}_{N_i}))^T$ and $\vec{u}_b(\hat{t}) = (u(\hat{t}, \vec{x}_{N_i+1}), \ldots, u(\hat{t}, \vec{x}_N))^T$. Then from (10)

$$\begin{pmatrix} \vec{u}_i(\hat{t}) \\ \vec{u}_b(\hat{t}) \end{pmatrix} = \begin{pmatrix} A_{ii} & A_{ib} \\ A_{bi} & A_{bb} \end{pmatrix} \begin{pmatrix} \vec{\lambda}_i(\hat{t}) \\ \vec{\lambda}_b(\hat{t}) \end{pmatrix}, \tag{11}$$

where the total coefficient matrix A has elements $a_{ij} = \phi(\varepsilon \|\vec{x}_i - \vec{x}_j\|)$ and the indicated block structure is due to the decomposition of interior and boundary node points. Furthermore, A is non-singular for standard choices of RBFs [19], and

$$\mathcal{L}\vec{u}_i(\hat{t}) = \begin{pmatrix} B_{ii} & B_{ib} \end{pmatrix} \begin{pmatrix} \vec{\lambda}_i(\hat{t}) \\ \vec{\lambda}_b(\hat{t}) \end{pmatrix} = \begin{pmatrix} B_{ii} & B_{ib} \end{pmatrix} A^{-1} \begin{pmatrix} \vec{u}_i(\hat{t}) \\ \vec{u}_b(\hat{t}) \end{pmatrix}$$
$$\equiv \begin{pmatrix} C_{ii} & C_{ib} \end{pmatrix} \begin{pmatrix} \vec{u}_i(\hat{t}) \\ \vec{u}_b(\hat{t}) \end{pmatrix}, \tag{12}$$

where the matrix elements of B are $b_{ij} = \mathcal{L}\phi(\varepsilon\|\vec{x}_i - \vec{x}_j\|)$, for $i = 1, \ldots, N_i$ and $j = 1, \ldots, N$.

For the time evolution of the problem, we use the BDF2 method [8] with a constant time step k. Let $\hat{t}^n = kn$ and let $\vec{u}_i^n \approx \vec{u}_i(\hat{t}^n)$. The time-stepping scheme applied to (1) yields

$$\vec{u}_i^n + \beta_1 \vec{u}_i^{n-1} + \beta_2 \vec{u}_i^{n-2} = k\beta_0 \mathcal{L}\vec{u}_i^n, \tag{13}$$

where $\beta_0 = 1$, $\beta_1 = -1$, and $\beta_2 = 0$ for the first time step and $\beta_0 = \frac{2}{3}$, $\beta_1 = -\frac{4}{3}$, and $\beta_2 = \frac{1}{3}$ for subsequent steps.

The boundary conditions are enforced at each new time level through

$$\vec{u}_b^n = \vec{g}_b^n, \tag{14}$$

where $\vec{g}_b^n = (g(\hat{t}^n, \vec{x}_{N_i+1}), \ldots, g(\hat{t}^n, \vec{x}_N))^T$, and

$$g(\hat{t}, \vec{x}) = \begin{cases} 0, & \vec{x} = \vec{0} \\ d^{-1} \sum_{i=1}^{d} x_i - \bar{K}\, \mathrm{e}^{-2\bar{r}\hat{t}}, & \|\vec{x}\|_1 \geq C. \end{cases} \tag{15}$$

Combining (12), (13), and (14) gives the overall scheme for advancing all unknowns one step in time,

$$\begin{pmatrix} I - k\beta_0 C_{ii} & -k\beta_0 C_{ib} \\ 0 & I \end{pmatrix} \begin{pmatrix} \vec{u}_i^n \\ \vec{u}_b^n \end{pmatrix} = \begin{pmatrix} -\beta_1 \vec{u}_i^{n-1} - \beta_2 \vec{u}_i^{n-2} \\ \vec{g}_b^n \end{pmatrix} \tag{16}$$

The initial condition from (1) in discrete form is

$$\vec{u}_i^0 = \vec{f}_i = (\Phi(\vec{x}_1), \ldots, \Phi(\vec{x}_{N_i}))^T. \tag{17}$$

Due to the change in β_0 between the first and second time step, we need to factorize the matrix block $I - k\beta_0 C_{ii}$ twice. However, this can be avoided by choosing the time step in a special way [12].

In, e.g., [10], the authors claim that the time-stepping is the major source of numerical errors. However, we suspect that this is related to how the boundary conditions are implemented. In our scheme (16) the boundary conditions are incorporated in a correct way, and we show in Section 4.3 that we get the expected second order convergence in time. If instead the boundary unknowns are adjusted separately after each time step, an error is introduced in the whole domain through the global coupling of the unknowns and time continuity is lost.

4 Numerical experiments

We have used multiquadric RBFs in all the experiments, i.e., $\phi(r) = \sqrt{1+r^2}$. The far-field boundary surface was given by all $\vec{x}$ for which $\frac{1}{d}\sum_{i=1}^{d} x_i = 4\bar{K}$. The problem parameters were set to $\bar{r} = 5/9$, corresponding to $r = 0.05$, and $\bar{\sigma} = 1$, corresponding to $\sigma = 0.3$, in one dimension. For the two-dimensional problem we used

$$\bar{\sigma} = \begin{pmatrix} 1 & 1/6 \\ 1/6 & 1 \end{pmatrix}, \quad \text{corresponding to} \quad \sigma = \begin{pmatrix} 0.30 & 0.05 \\ 0.05 & 0.30 \end{pmatrix}.$$

The number of time-steps M is optimized to give the best result in some experiments and is otherwise chosen large enough not to affect the accuracy of the solution. The exercise time used was $T = 0.045$, corresponding to 1 year.

The accuracy of the RBF method naturally depends on the number of node points N. However, the accuracy is also very much influenced by the choice of shape parameter and to a lesser degree by the distribution of the node points. In the following subsections, we first discuss how to make these choices, and then we look at space and time accuracy.

4.1 Node distribution

Since we are concerned with making the error small in the region of interest, we can adapt the node point distribution to reduce the financial error norms, while allowing a larger error in the far-field region.

We have not tried to optimize the node distribution, but we have tried some different approaches and found one that gives a clear improvement compared with a uniform distribution. Examples are shown in Figure 3. In one dimension,

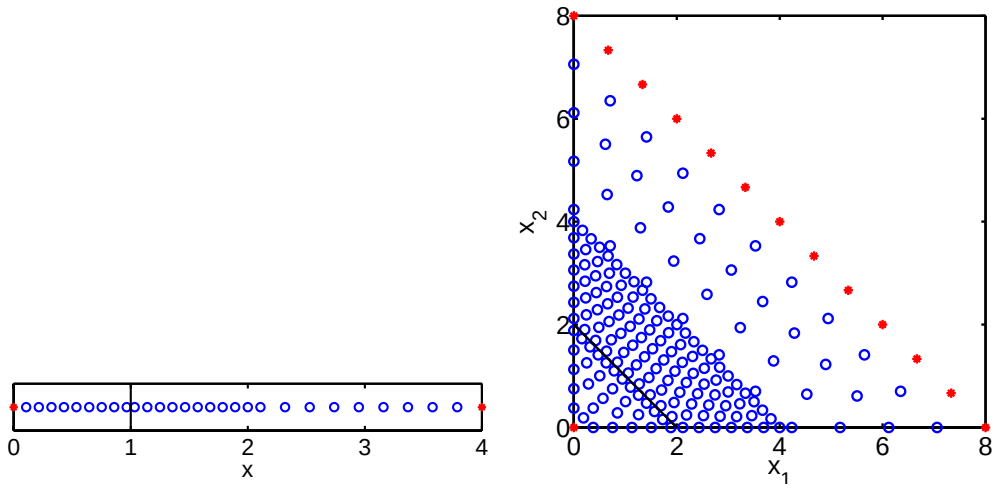


Fig. 3. Non-uniform node distributions in 1D (left) and 2D (right).

the node points are placed in the following way. If $N = 3p+2$, for some integer p, we distribute $p+1$ points uniformly in the intervals $[0, \bar{K}-\delta]$ and $[\bar{K}+\delta, 2\bar{K}]$. Then we place the remaining p points in the last part of the computational domain. The small distance, δ, from $\bar{K}$ is chosen as $\delta = 1/(N-1)$. In two dimensions a similar distribution is chosen in the diagonal direction. Figure 4 shows the difference between using a uniform and non-uniform distribution in one dimension. Figure 5 also shows an example of the error $E(x)$ for the two distributions. A comparison of the errors for the two types of distributions

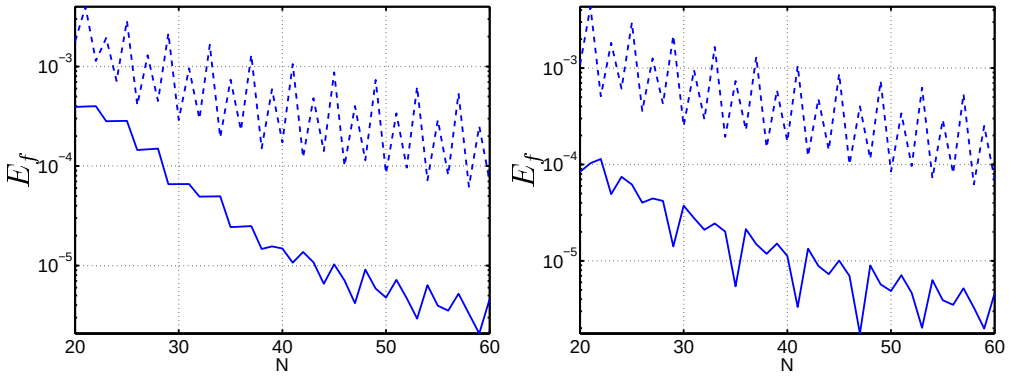


Fig. 4. Financial error norms for uniform (dashed) and non-uniform (solid) node distributions in 1D with $\varepsilon = 4$ (left) and $\varepsilon = 1 + N/20$ (right).

in two dimensions is shown in Figure 6. To compute these errors we used a reference solution computed by the finite difference method on a very fine grid.

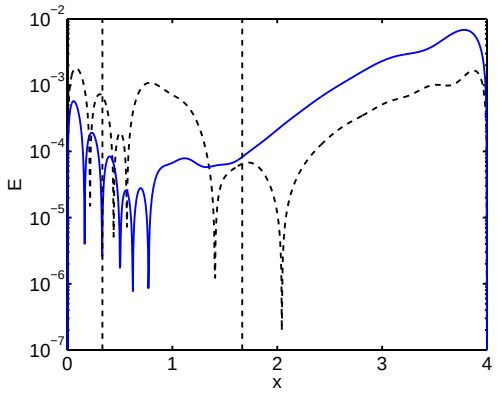


Fig. 5. The absolute value of the error $E(x)$ for $N = 20$ and $\varepsilon = 2$ for uniform (dashed) and non-uniform (solid) node distributions in 1D.

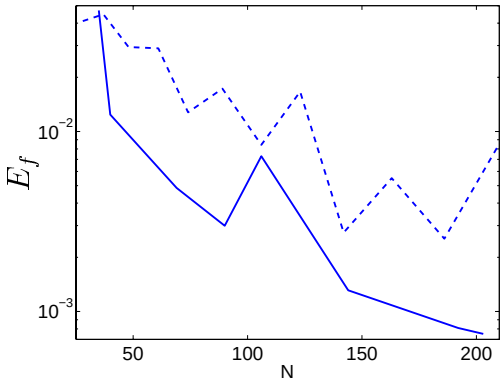


Fig. 6. Financial error norms for uniform (dashed) and non-uniform (solid) distributions in 2D with $\varepsilon = 1$.

4.2 Choosing the shape parameter value

The best choice of shape parameter is problem dependent [14] and there is (currently) no easy way to determine it a priori. Furthermore, the RBF matrices become increasingly ill-conditioned when ε decreases, making it impossible to compute the approximation at small optimal shape parameters using standard methods. However, there are methods to get around this for moderate numbers of node points [5,13].

The best shape parameter value, for N ranging from 20 to 60 in the one dimensional problem, can be reasonably well approximated by $\varepsilon = 1 + N/20$. The difference between using a constant ε and the formula above is illustrated in Figure 4. As can be seen in Figure 7 the optima are not always well defined and we are very close to the ill-conditioned zone.

It is easy to believe that the formula that works well for small N is a suitable choice also for larger N. However, the asymptotic convergence rate can be very different from the initial behavior [15]. This is illustrated in the left

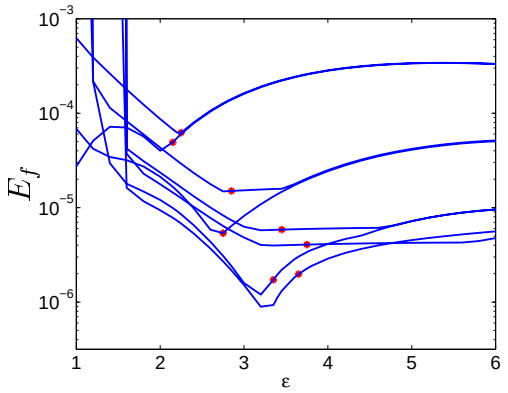


Fig. 7. The financial error norm as function of ε for $N \in [20, 60]$. The stars show $\varepsilon = 1 + N/20$.

part of Figure 8, where the error is plotted for a larger range of N. The error is computed with the non-uniform distribution and plotted against the corresponding uniform step size $h = 4\bar{K}/(N-1)$. The fitted slopes are 1.3 and 4.4 and indicates an algebraic rate of convergence in both regions. Here, the number of time steps M has been optimized to remove a bit of saw-tooth behavior. The right part of the figure shows that by letting ε grow slower with N, we improve the asymptotic rate of convergence. The slopes are 1.5, 1.9 and 2.4 respectively. The convergence rates could be improved even more by taking smaller ϵ, but the ill-conditioning prevents us from doing this.

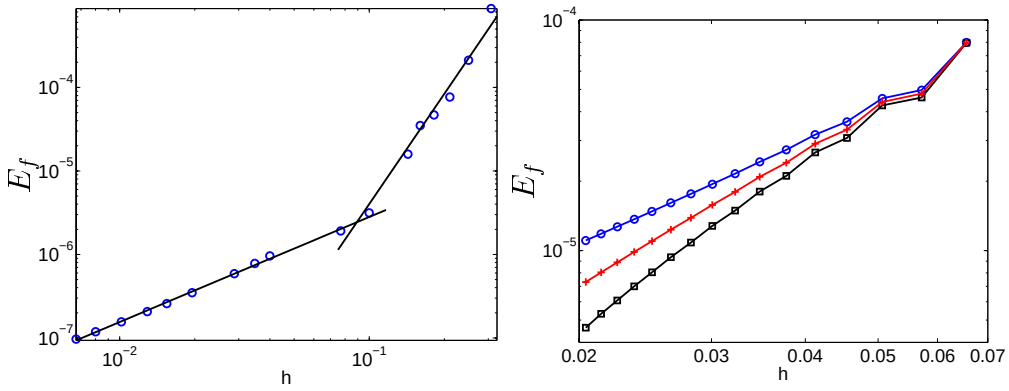


Fig. 8. Left: Financial error norm for the choice $\varepsilon = 1 + N/20$. Right: Financial error norm for $\varepsilon = 1 + N/20$ (o), $\varepsilon = a + bN^{3/4}$ (+) and $\varepsilon = c + dN^{1/2}$ (□), where a, b, c, and d are constants.

4.3 Accuracy in space

One of the main advantages of the RBF method is that it can provide spectral accuracy. However, the experiments in the previous section only showed algebraic convergence. The reason is that ε was increased with N. The spectral

accuracy holds for fixed ε and in Figure 9 the spectral convergence rate can be observed. That is,

$$E_f = C \exp(-\alpha N).$$

Also here the asymptotic rate of convergence is different from that for small N. The value of α is approximately 0.2 for all three values in the small N region and in the large N part, $\alpha \approx 0.032$, 0.026, and 0.021. As can be seen, the spectral rate is higher for smaller ε. This was observed also in [15].

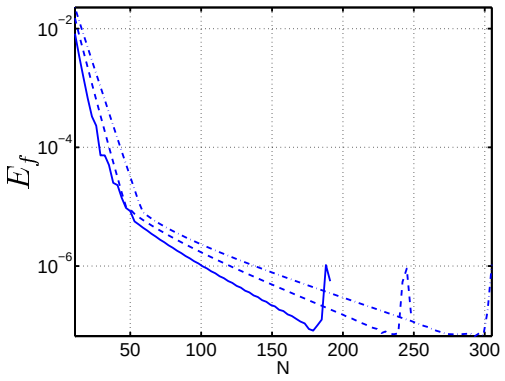


Fig. 9. The financial error norm as a function of N for constant $\varepsilon = 6$ (solid), $\varepsilon = 8$ (dashed), and $\varepsilon = 10$ (dash-dot).

4.4 Accuracy in time

The accuracy in time was studied by fixing the spatial part of the approximation to $N = 98$ RBFs with shape parameter $\epsilon = 1 + 98/20 = 5.9$, and then varying the number of time steps from $M = 2$ to $M = 10^4$. The results are displayed in Figure 10.

The different curves correspond to different ways of measuring. For all errors we see that the expected order of accuracy 2 is realized. However, measuring the maximum error over the whole interval (+) includes large errors at the boundary that increasing M cannot remove. We can draw this conclusion since when measuring the maximum error in the interior of the domain, with the financial norm (solid line), it is possible to get smaller values of the error without increasing N or changing any other parameter. Using the weighted integral norm (o) it is possible to reduce the error even further. Studying the error locally at the strike price shows that we can get errors as low as 10^{-8} with 98 basis functions. For the three last ways of measuring the error it is very clear when the error from the space approximation takes over and starts to dominate. When this happens and on what level depends on where and how the error is measured.

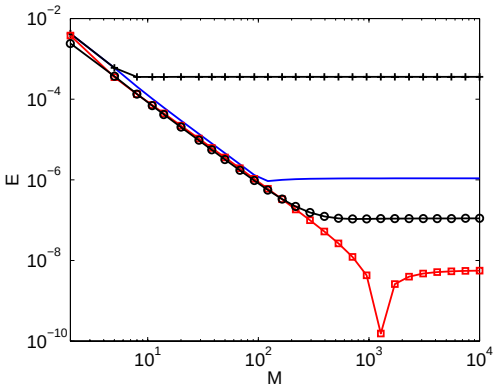


Fig. 10. The error E as a function of the number of time steps M. Maximum error over the whole region (+), financial norm (solid), weighted integral norm (o) and error at $x = \bar{K}$ ($\Box$).

5 Efficiency of the RBF method compared with the finite difference method

In order to investigate the efficiency of the RBF method, we measured the execution time for the one dimensional problem and the execution time and memory requirements for the two dimensional problem, and compared the results with those of the finite difference method presented in [20]. Both implementations are in MATLAB and none of the codes is optimized. A brief description of the finite difference method and the results of the comparisons are given below.

5.1 Adaptive finite differences

The generalized Black–Scholes equation (1) can be solved by approximating the derivatives in space and time by finite difference, see e.g. [21]. In [20] centered second order finite differences on a structured but not equidistant grid are used in space, and the second order implicit, unconditionally stable BDF2 scheme [8] is used for the time discretization.

The adaptive algorithm in space automatically adjusts the discretization to achieve a predefined truncation error. This allows the user to choose the error level instead of the number of grid-points as is standard in non-adaptive finite difference implementations. The adaptive method can alternatively be used to minimize the memory usage by restricting the number of grid-points used in each dimension.

Time adaptivity is implemented through a variable step size BDF2 version combined with an explicit multi-step method used for estimating the local

truncation error at each time-step. The time-step is then chosen so that the error is controlled.

The adaptive method has been successfully used for European basket options in [20] where the local truncation error is controlled and in [16] where a functional of the global error is estimated and controlled using a similar technique.

Since the time-stepping algorithm is implicit and the approximation of space derivatives is local, the solution of large, but very sparse, systems of equations is necessary. For this purpose, the iterative restarted GMRES method [7] has been used, together with a preconditioner (incomplete LU factorization) to speed up the computations.

5.2 Results of the efficiency tests

The results of the tests can be seen in Figures 11 and 12. In Figure 11 the effect of the shape parameter choice can again be observed. The formula $\varepsilon = 1+N/20$ was used, and the two different convergence rates are reflected by the time consumption of the RBF method. With a lower convergence rate, N must increase more to get to a desired tolerance, and hence the computational time grows faster. This illustrates that another choice of shape parameter values should preferably be made for larger N, i.e., when extremely high accuracy is desired.

For the 2D experiments the choice of ε was not made in a rigourous way. The shape parameter value was optimized locally in a small interval, typically around $\varepsilon = [0.5, 3.5]$, close to the ill-conditioned zone for each experiment. Again the results in Figure 12 suggest that these choices of shape parameters are not optimal for larger N, corresponding with the left part of the figure.

Measured in the right hand parts of the two figures, i.e., where the choices of ε are relatively good, the RBF method is approximately 20–40 times faster than the finite difference method. Although the speedup is lower in two dimensions than in one dimension, it is still good enough in two dimensions to suggest that the RBF method can be more efficient than the finite difference method for (even) higher dimensional problems. The memory requirements are rather similar for the two methods in two dimensions, which is a positive result considering that the RBF method works with dense matrices, whereas the finite difference method uses sparse matrices. There are also possibilities to improve the performance of the RBF methods further both with respect to memory and time usage [12].

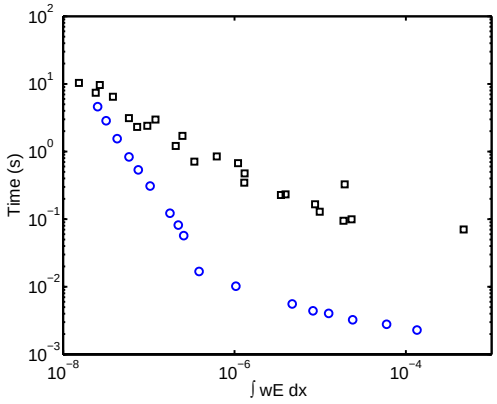


Fig. 11. Time efficiency comparison between the RBF method (circles) and the second order accurate adaptive finite difference method (squares) for the one-dimensional problem.

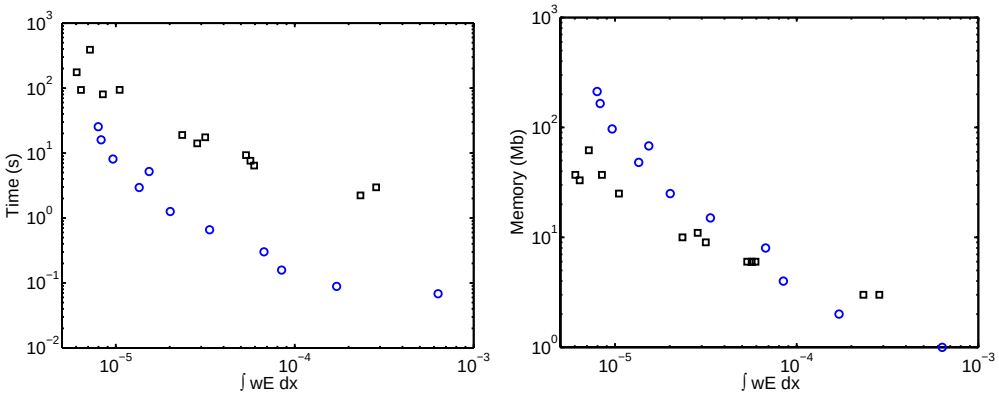


Fig. 12. Time and memory efficiency comparison between the RBF method (circles) and the second order accurate adaptive finite difference method (squares) for the two dimensional problem.

6 Conclusions

In this work we have derived a streamlined RBF method for option pricing in several dimensions, including boundary conditions. We have shown that it is second order in time (due to the second order time-stepping scheme) and spectrally accurate in space. We have also shown that it can be difficult to take full advantage of the spectral property due to the ill-conditioning of the RBF matrices for small shape parameter values.

Furthermore, we have shown how an adaptive placement of the node points, instead of a standard uniform choice, can increase the accuracy by up to an order of magnitude. We believe that even better node distribution strategies can be found for problems in two or more dimensions. By exploiting the meshfree nature of RBF approximation, we can also reduce the size of the computational domain by two in two dimensions and more in higher dimensions.

We have investigated how the convergence rate in space is affected by the choice of shape parameter and found that if the shape parameter is increased according to some formula $\varepsilon \propto N^q$, $q \geq \frac{1}{2}$, the resulting convergence rate becomes algebraic. Probably, there is a gradual transition from spectral to algebraic convergence rate for $0 \leq q \leq \frac{1}{2}$.

The new RBF method has been compared with an existing second order adaptive finite difference method and the experiments show that the RBF method can be 20–40 times than the finite difference method. The memory requirements of the two methods are comparable for the problems considered here.

We conclude that overall, the RBF method performs well and we expect to beat at least the finite difference methods also in higher dimensions.

References

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

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

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

[4] 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.

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

[6] 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.

[7] A. Greenbaum, Iterative methods for solving linear systems, Vol. 17 of Frontiers in Applied Mathematics, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997.

[8] 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.

[9] Y. C. Hon, A quasi-radial basis functions method for American options pricing, Comput. Math. Appl. 43 (3) (2002) 513–524.

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

[11] S. Janson, J. Tysk, Feynman–Kac formulas for Black–Scholes type operators, Bull. London Math. Soc. 38 (2006) 269–282.

[12] E. Larsson, K. Åhlander, A. Hall, Option pricing using radial basis functions and the generalized Fourier transform, manuscript in preparation (2006).

[13] 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.

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

[15] E. Larsson, U. Pettersson, Fixed shape parameter radial basis function approximations for PDE problems, manuscript in preparation (2006).

[16] P. Lötstedt, J. Persson, L. von Sydow, J. Tysk, Space-time adaptive finite difference method for European multi-asset options, Tech. Rep. 2004-055, Dept. of Information Technology, Uppsala Univ., Uppsala, Sweden, available at `http://www.it.uu.se/research/publications/` (2004).

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

[18] 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.

[19] C. A. Micchelli, Interpolation of scattered data: distance matrices and conditionally positive definite functions, Constr. Approx. 2 (1) (1986) 11–22.

[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] D. Tavella, C. Randall, Pricing Financial Instruments: The Finite Difference Method, John Wiley & Sons, Inc., New York, 2000.

[22] 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.