High-order adaptive space-discretizations for the Black–Scholes equation

Gunilla Linde, Jonas Persson and Lina von Sydow

April 19, 2006

Abstract

In this paper we develop a high-order adaptive finite difference space-discretization for the Black–Scholes (B–S) equation. The final condition is discontinuous in the first derivative yielding that the effective rate of convergence is two, both for low-order and high-order standard finite difference (FD) schemes. To obtain a sixth-order scheme we use an extra grid in a limited space- and time-domain. The new sixth-order method is called FD6G2. The FD6G2-method is combined with space- and time-adaptivity to further enhance the method. To obtain solutions of high accuracy in several dimensions the adaptive FD6G2-method is superior to both standard and adaptive second-order FD-methods.

1 Introduction

In option pricing problems, centered second-order finite differences (FDs) are commonly used for solving the Black–Scholes (B–S) equation. B–S is a parabolic final-value partial differential equation (PDE) that gives the arbitrage free price of an option. The main problem when solving the multi-dimensional B–S numerically is the "curse of dimensionality", i.e. the number of unknowns in the discretized problem grows exponentially in the number of dimensions. In recent years different approaches to improve the accuracy of the standard second-order FD-schemes have been presented. This offers the possibility to reduce the number of unknowns and hence increase the effectiveness of the method.

By using a coordinate transformation, stretching the region around the strike price where there is a discontinuity in the first derivative of the final condition, Clarke and Parrott found that the accuracy of their implicit FD-scheme was improved [2]. However, it is not obvious how this coordinate transformation generalizes for the multi-dimensional problem.

Another way of obtaining more grid-points around the discontinuity is to use a nonuniform mesh and adaptivity as Persson and von Sydow [10]. Zhao and Wei used discrete singular convolution together with an adaptive mesh and obtained results more accurate than FD-results from the same adaptive mesh [17].

Arciniega and Allen [1] used Richardson extrapolation and by that increased the accuracy of the results obtained by the Crank-Nicolson scheme. No convergence rate of the scheme was presented, and the numerical results presented indicate that the method is only second-order accurate. Düring, Fournié and Jüngel [3] used Rigal's schemes R2 and R3 [12] together with a new scheme called R3C on a nonlinear B–S equation. Those schemes are second-order accurate in time and fourth-order accurate in space. Düring et. al. obtained third-order convergence in space for the nonlinear problem, since they used the R2 or R3 schemes in combination with spline interpolation of higher-order to smooth the initial data.

McCartin and Labadie [9] used the Crandall-Douglas scheme. They showed that the results obtained with the Crandall-Douglas scheme are more accurate than results from Crank-Nicolson, but did not investigate the rate of convergence of the scheme for this particular problem. The authors only state that the Crandall–Douglas scheme "is (at least) fourth-order accurate and unconditionally stable" and that experiments with mesh refinement confirm the accuracy of their numerical results.

FD methods for sharp gradients and shock resolution have been studied in physics, for instance for problems in fluid dynamics [4],[14]. The most common procedure is to refine the mesh around discontinuities or sharp gradients. Gerritsen and Olsson [4] showed that for problems with a sharp gradient, all the considered FD methods, both lower-order classical and higher-order compact schemes, exhibited qualitatively the same rates of convergence when used on a uniform mesh.

In this paper, pricing of European call options is considered and the B–S equation is solved by a new sixth-order FD method called FD6G2. Standard sixth-order FD-discretizations yield only second-order accurate results due to the discontinuity in the final solution. Hence, FD6G2 uses an extra (fine) grid around the discontinuity in a limited space- and time-domain. The FD6G2-method is also used in combination with space- and time-adaptivity. The paper is an extension of a Master Thesis project presented in [7].

The paper is organized as follows. In Section 2 the B–S equation is described. In Section 3, the problem is discretized with different FD schemes. Section 3 also contains a description of the FD6G2-method. Space and time adaptivity is presented in Section 4. Convergence results are presented in Section 5 and results using our adaptive methods are studied in in Section 6. Finally, in

Section 7, we make some concluding remarks.

2 The Black-Scholes equation

When the option depends on several underlying assets, the B–S equation reads

$$\frac{\partial F}{\partial t} + \sum_{i=1}^{d} r s_i \frac{\partial F}{\partial s_i} + \frac{1}{2} \sum_{i,j=1}^{d} [\sigma\sigma^*]_{ij}\, s_i s_j \frac{\partial^2 F}{\partial s_i \partial s_j} - rF = 0, \qquad (1)$$
$$F(s,T) = \Phi(s),$$

where d is the number of underlying assets. In Equation (1), F is the value of the option and K is the strike price. The expiry date is denoted by T, s_i is the value of underlying stock i, and $s = (s_1, \ldots, s_d)$. Finally, $r \in \mathbb{R}^+$ is the short rate of interest and σ is the volatility-matrix. An example of a contract function for the multi-dimensional European call option is the mean-value European basket option

$$\Phi(s) = \Big(\frac{1}{d} \sum_{i=1}^{d} s_i - K \Big)^+.$$

We transform (1) from a final-value-problem into a non-dimensional initial-value problem and solve the transformed problem. The transformation of the time-scale has the advantage that standard texts on time integration is applicable. The following coordinate transformations are used:

$$\begin{array}{lll} s = Kx, & \bar{r} = \frac{r}{\hat{\sigma}^2}, & KP(x,\hat{t}) = F(s,t), \\ \bar{\sigma} = \frac{\sigma}{\hat{\sigma}}, & \hat{t} = \frac{1}{2}\hat{\sigma}^2 (T-t), & K\Psi(x) = \Phi(s), \end{array} \qquad (2)$$

where $\hat{\sigma}$ is a constant chosen as $\max_{ij}[\sigma\sigma^*]_{ij}$, and $x = (x_1, \ldots, x_d)$. These coordinate transformations result in

$$\frac{\partial P}{\partial \hat{t}} = \mathcal{L}P. \qquad (3)$$

where $\mathcal{L}$ is defined by

$$\mathcal{L} = 2\bar{r} \sum_{i=1}^{d} x_i \frac{\partial}{\partial x_i} + \sum_{i,j=1}^{d} [\bar{\sigma}\bar{\sigma}^*]_{ij} x_i x_j \frac{\partial^2}{\partial x_i \partial x_j} - 2\bar{r}. \qquad (4)$$

In the next section the operator $\mathcal{L}$ is discretized.

3 Discretization

3.1 Wide sixth-order scheme for a uniform grid

We start by presenting the discretization in 1D. The generalization to the multi-dimensional problem is discussed at the end of this subsection. The transformed B–S equation is discretized on a uniform grid defined by $x_k = x_1 + (k-1)h$, $k = 1, 2, \ldots, N$, where N is the total number of grid-points and h is the uniform grid spacing, see Figure 1. For the transformed B–S equation the domain is $[0, x_{max}]$ where $x_{max} = \frac{s_{max}}{K}$, hence $h = \frac{x_{max}}{N-1}$.

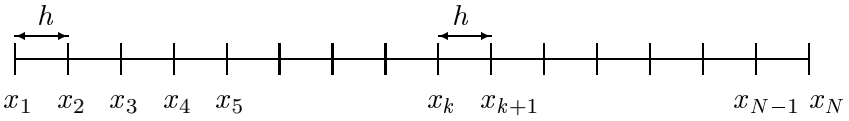

Figure 1: A uniform grid with N points and grid-spacing h.

The partial derivatives in the B–S equation are replaced by difference approximations. A centered second-order approximation is defined by

$$\frac{\partial P(x_k)}{\partial x} = (P(x_{k+1}) - P(x_{k-1}))/2h + \mathcal{O}(h^2), \tag{5}$$

$$\frac{\partial^2 P(x_k)}{\partial x^2} = (P(x_{k+1}) - P(x_k) + P(x_{k-1}))/h^2 + \mathcal{O}(h^2). \tag{6}$$

To obtain higher-order FD approximations of the first and second derivative Equation (5) and (6) is expanded in Taylor series and sufficiently many terms in the error are kept. If we want a sixth-order approximation all the error-terms up to h^6 must be kept. Then the higher derivatives in the error-terms are discretized yielding a seven points wide stencil. The coefficients for approximations of the first and second derivative are listed in [5].

The wide scheme requires special care when implementing boundary conditions. We have used boundary conditions of lower-order, see Section 3.3. These boundary conditions need fourth-order approximations of the first and second derivative. These approximations result in a stencil that is five points wide. These coefficients are also listed in [5].

The generalization to the multi-dimensional problem is straight-forward for the first- and second-order derivatives by (5) and (6). For the discretization of the mixed-derivative in the multi-dimensional problem see Section 3.2 and Linde's master thesis [7] where all coefficients are available.

3.2 Wide sixth-order scheme for a nonuniform grid

When using adaptivity in space it is necessary to create finite difference approximations on a nonuniform grid. In this case the coefficients will depend on the space coordinate in contrast to the coefficients for the uniform grid. For the first and second derivative a seven points wide stencil will be needed and by using Taylor expansions around the center point, like in the equidistant case, we can find a linear systems of equations that determine the coefficients. Of course, also here we need difference approximations of lower order at the boundaries. The coefficients for these approximations are derived in the same way. How to do this in practise is described in detail in [7].

For the multi-dimensional problem, the discretization of the mixed-derivatives results in a seven points wide stencil. Hence, the discretization of the mixed-derivative results in a stencil that involves $7 \times 7 = 49$ grid-points. The coefficients can be found in [7], where also coefficients for the fourth- and second-order approximations of the mixed-derivative are presented.

3.3 Boundary conditions

We present the boundary conditions used in 1D. The generalization to the multi-dimensional problem is straight-forward. For the standard second-order scheme (FD2) there are several possible boundary conditions for the B–S equation. Two boundary conditions often used are

$$\begin{array}{ll} \frac{\partial^2 P(0,\hat{t})}{\partial x^2} = 0, & \frac{\partial^2 P(x_{max},\hat{t})}{\partial x^2} = 0, \\ P(0,\hat{t}) = 0, & P(x_{max},\hat{t}) = x_{max} - e^{-2\bar{r}\hat{t}}. \end{array}$$

These and other boundary conditions are discussed in [16].

For the sixth-order scheme (FD6) additional numerical boundary conditions are needed since the wide stencils cannot be applied to grid-points near the boundary. Since we aim for problems in higher dimensions it is important with boundary conditions that easily can be applied also to these problems.

The boundary condition we have used is that the second derivative of the solution across a boundary is zero and the rest of the operator is discretized as in all inner grid-points. For the uniform grid the boundary condition in dimension i across the boundary is

$$\frac{P(x_{k-1}) - 2P(x_k) + P(x_{k+1})}{h^2} = 0, \qquad k = 2, N-1, \tag{7}$$

where a second-order approximation of the second derivative is used.

The boundary condition in Equation (7) is combined with a second-order approximation of B–S equation in x_2 and x_{N-1}. At grid-points x_3 and x_{N-2} the second-order FD2-scheme applied to B–S equation is used. At grid-points x_4 and x_{N-3} a fourth-order approximation of B–S equation is applied.

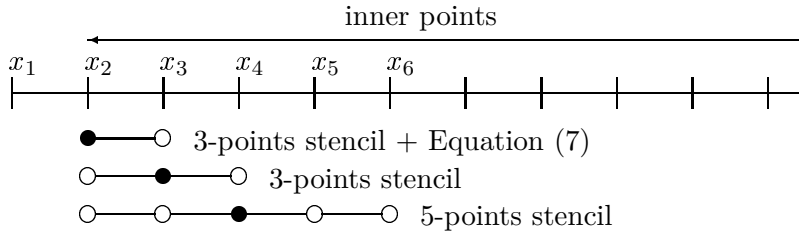

Figure 2: The different stencils used at the left boundary.

These boundary conditions could probably be improved for better accuracy, but in one space-dimension they work well enough if S_{max} is chosen large enough. Here we have used $S_{max} = 8K$.

The maximum error in the solution in one space dimension is usually located around K and the boundary points where the lower-order boundary conditions are applied are sufficiently far away from K and do not cause any problems. This is not the case in higher dimensions. For the two-dimensional problem we see larger errors where the discontinuity of the first derivative meets the boundary.

A thorough study of the boundary conditions for the B–S equation discretized with higher-order FDs is needed but is beyond the scope of this paper.

3.4 Two grids initially - the FD6G2-method

The initial condition for the transformed B–S equation for European call option is discontinuous in the first derivative at the strike price. Hence, the effective convergence rate of the higher-order schemes are reduced, see results in Section 5.1. There is a sharp gradient at the strike price during the first time-steps. The effects on the effective rate of convergence were studied by starting the calculations at time $\hat{t} = \frac{\hat{\sigma}^2}{2}0.1$ with the exact solution instead of at $\hat{t} = 0$ with the initial data. These calculations showed that the anticipated order of accuracy for the higher order scheme was achieved and we could conclude that the order of accuracy of a higher order standard scheme is lost due to the sharp gradient in the initial data.

To overcome the reduced rate of convergence a special procedure with an

extra fine mesh around the strike price is used during the first time-steps. This method, where two grids are used in parallel initially, will from here on be called FD6G2. This refers to a 6th order FD-scheme with 2 grids at the start. Sixth-order FD approximations were implemented also for $G2$ - a finer grid around the strike price - and the numerical results presented in Section 5.1 and 5.2 show that the FD6G2-method obtains results that are over-all sixth-order accurate.

The coarse grid over the whole domain has spacing h and the small grid $G2$ around the strike price has grid-spacing $\hbar < h$. This is illustrated for a 1D-problem in Figure 3.

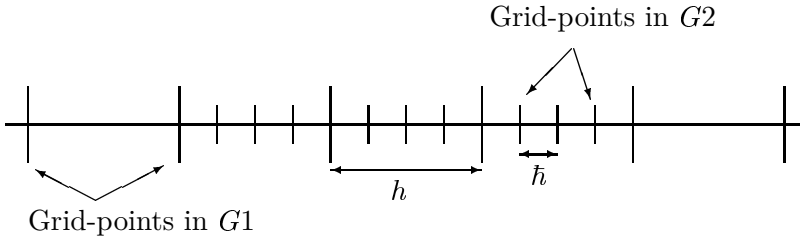

Figure 3: The coarse grid $G1$ and the second grid $G2$. In this case $G2$ has refinement 4.

We know from experiments that the error close to the strike price in the beginning is $\mathcal{O}(\hbar^2)$. To construct a method with over all spatial error $\mathcal{O}(h^6)$ the error close to the strike price must be reduced to $\mathcal{O}(h^6)$ even though the effective rate of convergence there is only $\mathcal{O}(\hbar^2)$. This can be achieved by using a smaller step there so that approximatively

$$\hbar^2 = C^2 h^6, \tag{8}$$

where C depends on higher derivatives of the solution. Equation (8) can be rewritten as

$$\frac{h}{\hbar} = \frac{1}{Ch^2}. \tag{9}$$

The refinement R of the grid is defined as

$$R = \lceil h/\hbar \rceil. \tag{10}$$

For a nonuniform grid with space-steps h_k, the refinement is defined as

$$R = \lceil 1/(C(\min h_k)^2) \rceil. \tag{11}$$

If $R = 4$, three equidistant grid-points are inserted between the grid-points in the coarse grid, see Figure 3.

The stencil of the sixth-order wide scheme is seven points wide and if the stencil is applied to a grid-point four points away from the strike price, points g_{-4} and g_4 in Figure 4, the stencil will not cross the discontinuity and the obtained error in the solution in this grid-point should be small. In the grid-points closer to the strike price the stencil will cross the discontinuity and a solution with a larger error is expected. Intuitively the second grid should stretch over six coarse grid-points, see Figure 4.

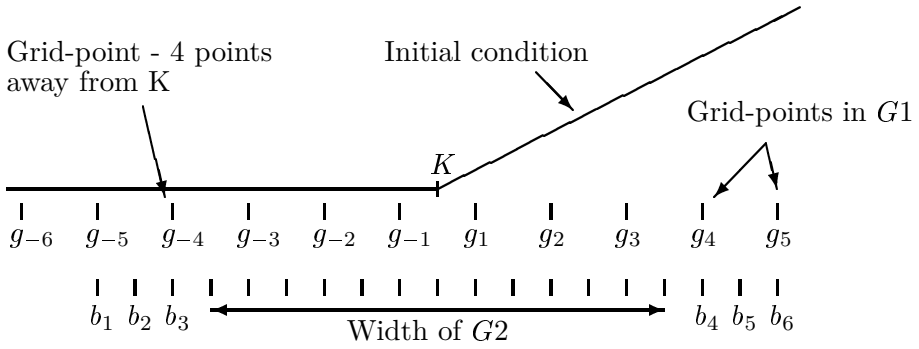

Figure 4: Intuitive width of $G2$. Here $R = 2$.

Every time-step with two grids consists of three phases. In the first phase the problem is solved on the coarse grid, $G1$, with the wide sixth-order scheme. In the second phase the problem is solved on $G2$ with the same sixth-order scheme. Since the stencil for the sixth-order scheme is seven points wide, values of three additional points, on each side of the grid are required. In Figure 4 these boundary points are called b_k, $k = 1, \ldots, 6$. The solution computed on the coarse grid is used to determine the values of the boundary points b_k. Interpolation from the coarse grid $G1$ is used if there is not a direct correspondence between points on the coarse grid and the points b_k.

In the third phase the values of the grid-points in $G1$ covered by $G2$, points g_k, $k = -3, -2, -1, 1, 2, 3$ in Figure 4, are updated using values computed in the second phase. The refinement R is chosen as an integer. Hence, the grid-points in the coarse grid will correspond to a grid-point in $G2$ in the region with two grids and the update process is straight forward.

The FD6G2-method can be summarized by this algorithm where T_{change} defines when to change from using two grids to one.

Algorithm 1

- For every time-step where $\hat{t} < T_{change}$.

1. Compute the solution on grid $G1$ for $\hat{t}$.
2. Compute the solution on grid $G2$ for $\hat{t}$.
3. Update the value of the grid-points in $G1$ that have a corresponding point in $G2$.

- For $\hat{t} \geq T_{change}$.
  1. Compute the solution on grid $G1$ for $\hat{t}$.

If the number of grid-points in $G1$ to the left of K are too few the left boundary condition used at $s = 0$ will be used also to the left of the fine grid $G2$.

Studying equations (9)-(11) we note that with a large value of C and a small N the refinement R can be one. It is obvious that in that case nothing is gained and the procedure with the second grid is worthless. Therefore a lower bound on R, called R_{min}, is needed to achieve a sixth-order space approximation. The use of an extra grid increases the memory requirements and the computational work to some extent, but higher accuracy is gained. To reduce the overhead introduced by the extra grid we want a high value on C, a narrow grid $G2$ and a low value of R_{min} and T_{change}, so that the extra work introduced is minimized. The FD6G2-method is discussed with respect to these parameters in the Master thesis by Linde, see [7].

For more than one space-dimension the fine grid $G2$ would ideally be used only around the discontinuity of the initial data. However, since that would complicate the implementation we have used a rectangle that enclose the discontinuity. Our regular boundary conditions are used when the boundary of the fine and coarse grid coincide and interpolation from the coarse grid is used on the other boundaries.

3.5 Temporal discretization

The time-derivative in Equation (3) is approximated with the second-order accurate backward differentiation formula, BDF2 [6]. It is an implicit and unconditionally stable scheme. The BDF2 algorithm for non-constant step-lengths reads

$$\begin{aligned} &\alpha_0^n P^n = \Delta t^n A P^n - \alpha_1^n P^{n-1} - \alpha_2^n P^{n-2}, \qquad (12)\\ &\alpha_0^n = \frac{1+2\theta^n}{1+\theta^n}, \quad \alpha_1^n = -(1+\theta^n), \quad \alpha_2^n = \frac{(\theta^n)^2}{1+\theta^n},\\ &\theta^n = \frac{\Delta t^n}{\Delta t^{n-1}} \end{aligned}$$

where A denotes the discretization of the spatial operator $\mathcal{L}$, see Equation (4). Note also that $\Delta t^n = \hat{t}^n - \hat{t}^{n-1}$ and $\hat{t}^n$ is the value of the scaled time at time-step n. Since BDF2 is a multi-step scheme another method must be used for the first time-step. We have used the first-order accurate implicit backward Euler method for this first step. This will not degrade the over all accuracy of the BDF2 method.

The method requires small time-steps for a short period of time in the beginning of the time-stepping process in order to achieve sixth-order accuracy in space (see Section 6). In Section 4.2 we show how we can use an adaptive BDF2 method that automatically adjusts the time-steps to keep the discretization error at a pre-described level. This technique was also successfully used in [8] and [10].

4 Adaptivity in space and time

4.1 Adaptivity in space

An adaptive technique is used to reduce the number of grid-points and still keep the discretization error at a prescribed level. We use the same idea of space adaptivity as presented in [10] and we refer all readers to that sorce for details on the method. Here we only present the basic idea about the adaptivity. The adaptive spatial process works like this:

Algorithm 2

1. Solve the problem once with a coarse grid with N_e grid-points, giving an estimate of the local discretization error.
2. Create a new grid in space such that the required accuracy criteria is met. The number of grid-points in the new grid will be denoted N_a.
3. Solve the problem again on the new grid.

Note that the problem has to be solved twice. The idea is that the first time the problem is solved it should be done quickly on a coarse grid. The solution will be of low accuracy but will give an estimate of where to place the grid-points in the second grid.

Assume that for any smooth solution $u(x)$ it holds that

$$A_h u_h = Au + \tau_h$$

where u_h is a vector of unknowns and A_h is the the discrete approximation of the operator $\mathcal{L}$. Au is the exact operator $\mathcal{L}u$ evaluated in the grid-points and thus τ_h is the discretization error for the FD-approximation with

step-length h. We estimate the discretization error using two grids with a technique similar to Richardson extrapolation which gives us an estimate of τ_h. Assuming that τ_h can be approximated by its leading order term and requiring that τ_h should be less than some constant $\epsilon_h > 0$ we find that we should choose our new discretization as

$$h(x) = \tilde{h} * \left(\frac{\epsilon_h}{|\tau_{\tilde{h}(x)}|} \right)^{\frac{1}{p}}. \tag{13}$$

Here $\tilde{h}$ is the step-length used to measure $\tau_{\tilde{h}(x)}$ and p is the order of the discretization. This technique to estimate the discretization error is presented in [15] in the context of option pricing and the adaptive technique was used and described in detail in [10] and [11].

In many dimensions we make the assumption that $[\sigma\sigma^*]_{ij} \ll [\sigma\sigma^*]_{ii}$ to simplify the theory . The same assumption was used in e.g. [10] and allows us to study the discretization one dimension at the time and create an adaptive grid for each dimension separately.

4.2 Adaptive time-integration

The adaptive time-stepping scheme used here consists of two parts, an explicit predictor and the implicit corrector BDF2 with non-equidistant time-steps. For the algorithm BDF2 see Equation (12) in Section 3.5. We use the adaptive algorithm used by e.g. Persson and von Sydow in [10] in the context of option pricing and by Söderberg et. al. in [8] where a slightly different approach was used for fluid problems. The difference to [10] is that here we use a logarithmic distribution of the time-steps for the first solve when the local discretization error is estimated while an equidistant distribution is used in [10]. The reason for using a logarithmic distribution is that it is necessary to have many small time-steps in the beginning, during the phase where we use two grids, to get a good approximation of the local discretization error. This is necessary for the second solve to be sixth order accurate.

The idea is to estimate τ_t, the local truncation error in the approximation of the time derivative in BDF2 by comparing a solution computed with an explicit and an implicit method. By using knowledge about the local truncation error in the two methods a good estimate of the local truncation error of BDF2 can be found. The information about the truncation error can then be used in different ways. In [8], Söderberg et. al. continously recompute the solution with smaller and smaller time-steps until they find a step that gives a small enough truncation error. Here we use a slightly different approach. Since we calculate the solution on a coarse grid to estimate the local errors in space we take the opportunity to simultaneously calculate the truncation

error from the time-discretization. Then we compute the time-steps needed to achieve the desired accuracy level ϵ_t on the adaptive time-grid using a method similar to the one used in space. Since the same method was used before by the authors we will not repeat the details here but refer the reader to [10] for details.

5 Convergence results

5.1 Results in one space-dimension

All computations have been performed with the transformed PDE, Equation (3), and were done in Matlab 6.5.0. Results presented with an execution time were carried out on a Sun Fire 15k server unless something else is stated. One UltraSPARC III+ CPU at 900MHz was used and 4 GB RAM was accessible. The default parameters used in one space-dimension are: $\sigma = 0.3$, $r = 0.05$ and $K = 20$. For the wide schemes, FD6 and FD6G2, $S_{max} = 8K$ is used due to the boundary conditions. When results from the FD6G2-method are compared to results from the FD2-scheme, the same value of S_{max} is used. Hence, $S_{max} = 8K$ is also used for the FD2-scheme in this case, except for when results of different methods are compared with respect to efficiency and the lowest possible value of S_{max} is used for each method.

BDF2 is used for time-integration and the length of the time-step is constant at $\Delta t = \frac{\hat{\sigma}^2}{2} 0.0002$ for all tests concerning the space adaptivity. Note that Δt and the value given for the parameter T_{change} are both given in scaled time $\hat{t}$. Usually $T = 0.1$ year or $T = 2$ years are used. The exercise date, T, is in non-transformed time, see Equation (2).

Since BDF2 is an implicit method, a sparse linear system of equations has to be solved in each time-step. For the FD6G2-method two systems must be solved during the first time-steps when two grids are used. In one space dimension Matlab's built-in solver '\' has been used.

On equidistant grids the errors calculated are the maximum of the absolute value of the error, denoted by $\max|\text{error}|$, and the root mean square, l_2 error.

$$\max|\text{error}| = \max|F_k - F(x_k)|, \tag{14}$$

$$l_2 \text{ error } = \sqrt{h \sum_{k=2}^{N-1} \big(F_k - F(x_k)\big)^2}, \tag{15}$$

Where $F(x_k)$ is the analytic solution in point x_k and $F_k = KP_k$ where P_k is the numerical solution in transformed coordinates. For adaptive grids only $\max|\text{error}|$ is computed. To verify the rate of convergence of the FD-schemes we are interested in the error from the approximations of the space derivatives. For these tests we have therefore used small time-steps to keep the error from the time-discretization negligible.

The discretization errors in space, for a pth-order method is of the form

$$\text{error} = ch^p + \mathcal{O}(h^{p+1}) \tag{16}$$

where c is independent of h. The logarithm of Equation (16) and the use of $h = \frac{x_{max}}{N-1}$ gives

$$\log(\text{error}) = -p\log(N-1) + c', \tag{17}$$

where c' depends on x_{max}, c and higher-order terms.

For the FD6G2-method $T_{change} = \frac{\hat{\sigma}^2}{2}35 \cdot 0.0002$, $C = 40$, $R_{min} = 4$ and the intuitive width of the grid were used. These values of the parameters are taken from [7] where appropriate values are determined.

Figure 5 show that the rate of convergence is two, both for the wide sixth-order scheme (FD6) and the standard FD-scheme of second-order (FD2). The rate of convergence for the FD6G2-method is close to six, when $T = 2.0$ years. For a much smaller T we found that the rate of convergence was reduced to fifth-order. This is probably due to errors from approximating derivatives over the discontinuity in space not being damped out.

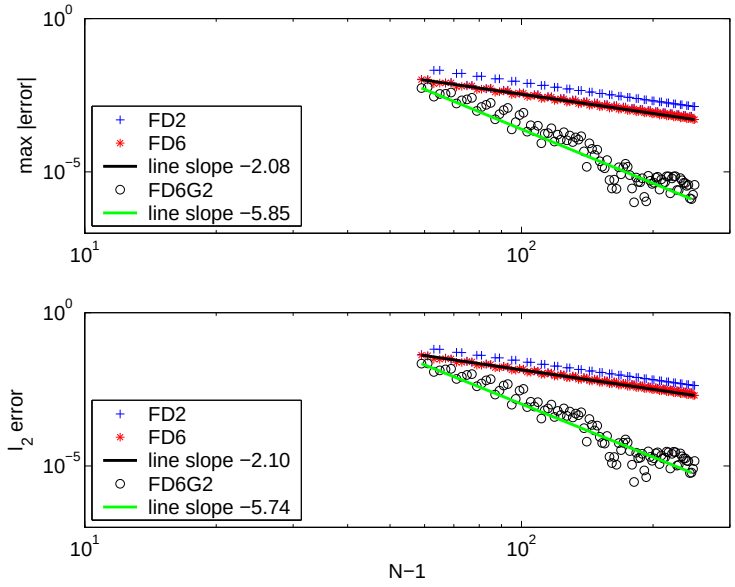

Figure 5: Convergence for the standard FD2 scheme, the wide sixth-order scheme (FD6) and the FD6G2-method. T=2.0 year.

In Figure 5 we show how the error is reduced with an increasing number of grid-points. However, it is impossible to give an exact rate of convergence from this figure. We chose a crude approximation and drew a straight line from the first to the last point in the data series and calculated the slope. We also see that the errors for different N using the FD2 scheme lie on two straight lines. The lower line is hard to distinguish because it coincides with the error from the FD6 scheme.

The results presented this far show that the wide sixth-order scheme (FD6) have an effective rate of convergence of two. This is in agreement with previously published results about FDs for problems with sharp gradients or discontinuities, see [12],[14],[17]. However, by using two grids initially, what we refer to as the FD6G2-method, we obtain a result that is sixth-order accurate in space.

The value R of the refinement of the fine grid is introduced in Section 3.4. Here we also introduce the parameter R_{min} since the refinement can be one when C is large and N is small. R_{min} is important to have for low values of N, since it is for few grid-points we can risk to get too little refinement. Experiments show that a minimum refinement of 4 is good. A larger minimum refinement does not give a more accurate solution.

Next, we will consider adaptivity in space. From now on we use T =

2. The local discretization error τ is measured at three different times, $[T/3, 2T/3, T]$. In each grid-point we use the largest of the three τ:s to compute the new grid, see Equation (13). In the first phase of the adaptive process, see Algorithm 2, an equidistant grid is used and throughout this section we have used $N_{e,G1} = 61$ ($N_{e,G2} = 27$) for the equidistant grid. We observed that other values did not change the final computational grid much.

Figure 6 shows max |error| for the adaptive FD6G2-method. Different values of ϵ are used to obtain grids with different number of points. For comparison the figure also contains the errors for the standard FD6G2, also presented in Figure 5, and errors for standard and adaptive FD2. Instead of using the same value of S_{max} for all schemes, the lowest possible S_{max}, that still gives the maximum error around K, was used in the experiment in this section.

The figure shows that the rate of convergence of the adaptive FD6G2-method is approximately six and that the adaptive FD6G2-method requires fewer or much fewer grid-points than the other methods examined to obtain a solution of the same accuracy.

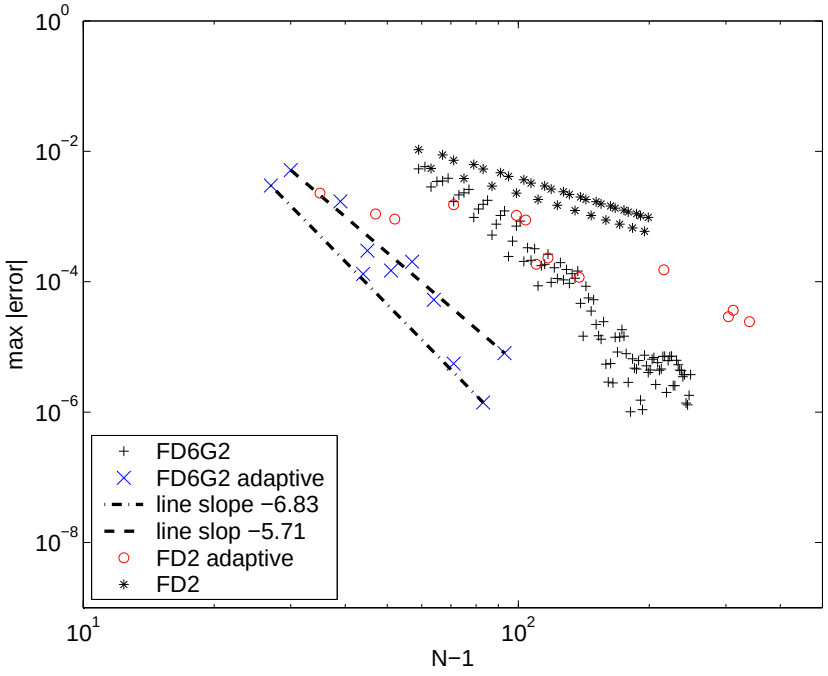

Figure 6: Convergence for the adaptive FD6G2-method.

The figure also shows that the second-order adaptive method is competitive with the higher-order methods only if low accuracy is acceptable. The

efficiency of the methods will be discussed in Section 6 where adaptive time-stepping have been used together with the space-adaptive method.

5.2 Results in two space-dimensions

All numerical experiments in two space-dimensions were performed on the same computer as the experiments in one dimension. Values for the parameters are $r = 0.05$, $K = 20$, $C = 40$, $R_{min} = 4$, $T_{change} = \frac{\hat{\sigma}^2}{2}35 \cdot 0.0002$ and $S_{max} = 2 \cdot 8K$ for the FD6G2-method and $S_{max} = 2 \cdot 6K$ for FD2. The volatility matrix σ has the value 0.3 on the diagonal and 0.05 in the other two elements in the matrix.

BDF2 with variable step-length, see Equation (12), is used for time integration. In the beginning, when two grids are used the constant time-step $\Delta t = \frac{\hat{\sigma}^2}{2}0.0002$ is used, then the length of the time-step is increased with 20% each time-step until $\Delta t_{\max} = \frac{\hat{\sigma}^2}{2}4 \cdot 0.0002$. The reason for this is to reduce the number of time-steps. We use these small time-steps so that we can study the discretization error in space. The increasing time-steps was confirmed to not lower the accuracy. Adaptive time-steps have also been studied, see Section 6.

Using the implicit BDF2 requires the solution of large sparse linear systems of equations. In two space-dimensions we used the iterative restarted `GMRES`, see [13], with an incomplete LU factorization as preconditioner. The LU-factorization was computed only for the first time-step and the iterations were stopped when the relative residual was less than 10^{-10}. The restart limit was set to 6.

For the two-dimensional problem we had no analytical solution so a reference solution was used instead. The reference solution was calculated with the adaptive FD6G2-method with time-steps $\Delta t = \frac{\hat{\sigma}^2}{2}0.0001$, $\Delta t_{\max} = \frac{\hat{\sigma}^2}{2}0.0004$ and $\epsilon = 10^{-8}$, resulting in $N_{a,G1} = 177$, $N_{a,G2} = 359$. These parameter values should give us a solution with small enough error to be used as reference solution.

Our experiments show that the largest errors are found at or close to the boundary. This is to be expected since we use lower order boundary conditions there. Since the error is of most interest in the inner of the domain (away from default of the stocks) we define the inner domain as all grid-points where a higher-order approximation is applied. Hence, the inner part is the region $[x_{1,5}, x_{1,N-4}] \times [x_{2,5}, x_{2,N-4}]$.

To calculate the error in a solution, the inner part of the reference solution is interpolated by cubic interpolation obtaining values in the same points as the numerical solution. All errors stated here are in the original variables

of B–S. Therefore |error| is the maximum absolute value of the error in the original variable F. The solver for the two-dimensional problem works for different number of grid-points in the first and second space-dimension, but all results presented in this paper are obtained with the same number of grid-points in both dimensions since our model problem is symmetric. In the first phase of the adaptive process we have used $N_{e,G1} = 61$ ($N_{e,G2} = 45$).

Our experiments show that the expected rate of convergence is obtained. We note that if the whole computational domain in space is studied we only have second order accuracy. That is the anticipated result though since the boundary conditions are only second-order accurate. If we study only the inner points we see that we have sixth-order accuracy. See the convergence plots in Figure 7 and 8 for details.

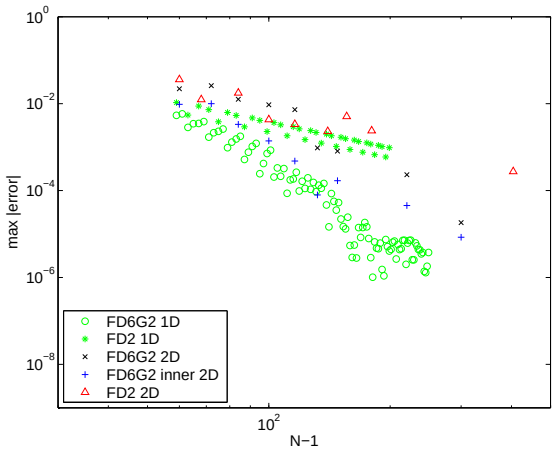

Figure 7: Convergence for FD2 and FD6G2 applied to the two-dimensional problem.

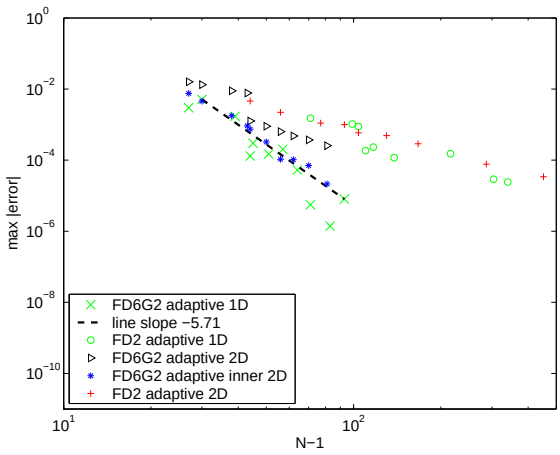

Figure 8: Convergence for adaptive FD2 and adaptive FD6G2 applied the two-dimensional problem.

6 Adaptivity results

In Section 5 we showed that our method FD6G2 can achieve sixth-order accuracy in space. Here we compare the efficiency of the adaptive second-order scheme and the adaptive FD6G2, both using the same second-order adaptive time-stepping described in Section 4.2. From here on FD6G2 will denote the adaptive sixth-order method.

In [10] a very coarse equidistant grid in both space and time was used to compute the initial solution where all local discretization errors are estimated. Here we must use a slightly different approach. For the space adaptivity to work properly we need to use small time-steps for a number of steps in the beginning of the time-stepping process. For the adaptive methods to be efficient, the first part where discretization errors are estimated has to be inexpensive and thus small time-steps must be avoided where possible. We have used a log-distribution of the time-steps in the initial solve in order to automatically use small time-steps in the beginning and then progressively larger time-steps towards the end. This allows us to use few time-steps during the first solve and still have a fine mesh in time in the very beginning. This is essential for the space-adaptivity to work properly as described earlier.

An example of how the algorithm controls the local truncation error τ_t is depicted in Figure 9. Here we have used $\epsilon_t = 10^{-3}$ as tolerance level.

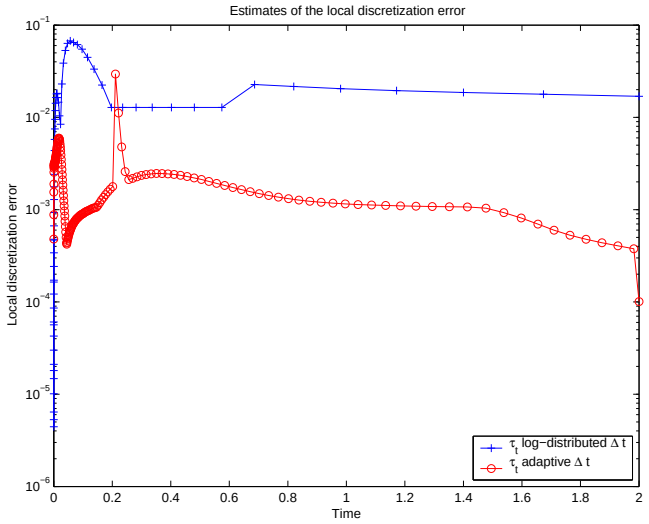

Figure 9: An example of τ_t with the log transformed and τ_t using the adaptive time-steps. Here ϵ_t was 10^{-3} and the problem one-dimensional.

Figure 9 shows that our adaptive method has problems in the beginning but after a while the time discretization error goes down to the desired tolerance level. The peak in the estimate on the adaptive grid (see the line with 'o'-markers), after about 0.25 years, occurs when we leave the fine mesh behind and use only the coarse grid. At that point the estimate of the local discretization error is influenced both by points that have been updated from the fine grid and points that have not been updated. This is visible in the figure where the estimate is much larger than expected. However, this is only a problem with the estimate. The actual error in the solution does not exhibit this behavior. In order to not use unnecessary small adaptive time-steps during the second solve at T_{change} we do not use the bad estimate. Instead we use the same estimate for a few time-steps (typically 5) just after T_{change}, this can be seen in Figure 9.

Note that in Figure 9 we have used the fine grid for a longer time than needed for illustrative reasons, from 0 to $0.1T$. Normally the fine grid is used only for a small fraction of the time-domain. In this case we used 65 log-distributed time-steps during the first solve. During the second solve, using adaptive time-steps, our method used 180 time-steps.

6.1 The two-dimensional problem

The experiments in this section were not performed on the same computer as the experiments in Section 5. Here we used an off-the-shelf standard

desktop PC. One should also know that the tolerance levels ϵ given are in the transformed variables and for each dimension separately. The errors mentioned are also in the transformed variable.

Here we study the computational times and the error when comparing with a fine reference solution for different levels of required local discretization error. The results are presented in Table 1. Note that N_a denotes the number of grid-points in each space-dimension on the adaptive grid. For the sixth-order method the numbers in parenthesis means the the number of grid-points used on the fine grid. The errors listed in the table is the maximum error measured in the area $[0.25{*}2K \quad 0.75{*}2K]\times[0.25{*}2K \quad 0.75{*}2K]$. We use this area because from a financial point of view the area closest to the boundaries are less important and we also know that our solution will be influenced by the lower order boundary conditions there. This means that we will measure the error in an area including the discontinuity in the derivative of the terminal condition but not close to the boundaries.

6th order	$\lvert\tau_h(T/3)\rvert$	$\lvert\tau_h(2T/3)\rvert$	$\lvert\tau_h(T)\rvert$	N_a	Time(s)	Error
$\epsilon = 10^{-3}$	2.80e-3	1.82e-3	1.51e-3	29(41)	36.9	9.51e-4
$\epsilon = 5 \cdot 10^{-4}$	2.48e-3	1.46e-3	1.22e-3	29(41)	40.1	8.95e-4
$\epsilon = 10^{-4}$	1.25e-3	5.62e-4	3.89e-4	37(41)	57.6	1.49e-4
$\epsilon = 10^{-5}$	3.99e-4	1.89e-4	1.34e-4	53(73)	198	4.06e-5
$\epsilon = 5 \cdot 10^{-6}$	1.05e-4	3.80e-5	2.38e-5	61(81)	365	1.10e-5
$\epsilon = 10^{-6}$	3.18e-5	1.18e-5	7.95e-6	77(97)	1113	4.95e-6
$(10^{-6}, 10^{-4})$	3.10e-5	1.15e-5	7.95e-6	77(97)	236	6.27e-6
2nd order	$\lvert\tau_h(T/3)\rvert$	$\lvert\tau_h(2T/3)\rvert$	$\lvert\tau_h(T)\rvert$	N_a	Time(s)	Error
$\epsilon = 10^{-3}$	1.29e-3	4.70e-4	2.86e-4	101	62.6	2.85e-5
$\epsilon = 5 \cdot 10^{-4}$	6.48e-4	2.49e-4	1.45e-4	141	182	1.98e-5
$\epsilon = 10^{-4}$	1.33e-4	5.01e-5	2.90e-5	305	2911	2.65e-6

Table 1: 2D experimental results. FD6G2 and FD2 with second-order time-adaptivity. $\#N$ denotes the number of grid-points in each dimension and $\tau_h(t)$ is the maximum absolute measured discretization error in each dimenion.

Studying the errors in Table 1 we see that we get much lower errors with the same number of grid-points using the higher order method. For the sixth order method the estimated local truncation errors are generally too large. This is probably due to the fact that this method uses very few grid-points, and the low-order boundary approximations are still present in the estimates.

For higher accuracy, errors as low as 10^{-6}, we see that the computational time for the second-order method is longer but for lower accuracy the second-order method is competitive. Note that for the two tolerance levels 10^{-3} and

$5 \cdot 10^{-4}$ there were really too few time-steps used with the fine grid in the FD6G2 method which could give a larger error than expected.

The last row in the table for '6th order' shows the results for $\epsilon_h = 10^{-6}$ in space and $\epsilon_t = 10^{-4}$ in time. We see that this results in very little loss of accuracy, compared to $\epsilon_h = \epsilon_t = 10^{-6}$, even though the computational time needed is less than a fourth. This is because the number of adaptive time-steps is much smaller. A natural next step to improve the method is to use a high-order discretization also in time.

In all experiments performed the higher order method needs less or much less grid-points than the second-order method. This means that they will in general use less memory. This is even more important when the number of dimensions in the problem increases.

7 Conclusions

The final condition is discontinuous in the first derivative at the strike price and therefore only second-order accurate results are obtained, both with the standard FD2 scheme and higher-order schemes using one grid. We have shown that with the new FD6G2-method that uses two grids initially, sixth-order accurate results are obtained. The FD6G2-method can easily be combined with space adaptivity obtaining more accurate results than the standard FD6G2, using the same number of grid-points.

The two dimensional results presented here show that for high accuracy the higher order method FD6G2 outperforms the second-order method. This is due to the much lower number of grid-points needed in each space dimension. It also means that the higher order method needs much less memory than the second-order method. However, the linear systems of equations are more dense than for the second order method, but since the number of unknowns is substantially smaller the execution times is shorter.

References

[1] A. Arciniega, E. Allen. Extrapolation of difference methods in option valuation, Applied Mathematics and Computation **153,** 165-186, 2004.

[2] N. Clarke, K. Parrott. Multigrid for American option pricing with stochastic volatility, Applied Mathematical Finance **6,** 177-195, 1999.

[3] B. Düring, M. Fournié, A. Jüngel. High order compact finite difference schemes for a nonlinear Black–Scholes equation, International Journal of Theoretical and Applied Finance Vol. 6, No. 7, 767-789, 2003.

[4] M. Gerritsen, P. Olsson. Designing an Efficient Solution Strategy for Fluid Flows 1. A Stable High Order Finite Difference Scheme and Sharp Shock Resolution for the Euler Equations, Journal of Computational Physics **129,**245-262, 1996.

[5] Bengt Fornberg. Generation of Finite Difference Formulas on Arbitrarily Spaced Grids, Mathematics of Computation Vol. 51, No. 184, 1988.

[6] E. Hairer, S.P. Norsett, G. Wanner. Solving ordinary differential equations 2nd ed.. Springer-Verlag, Berlin 1993.

[7] G. Linde. High-order Adaptive Space Discretizations for the Black-Scholes Equation. Master thesis. Uppsala University. 2005

[8] P. Lötstedt, S. Söderberg, A. Ramage, L. Hemmingsson-Frändén. Implicit solution of hyperbolic equations with space-time adaptivity, BIT, **42:1**:128-153, 2002.

[9] B.J. McCartin, S.M Labadie. Accurate and efficient pricing of vanilla stock options via the Crandall–Douglas scheme, Applied Mathematics and Computation **143,** 39-60, 2003.

[10] J. Persson, L. von Sydow. Pricing European Multi-asset Options Using a Space-time Adaptive FD-method, Department of Information Technology, Uppsala University, Technical report 2003-59, 2003 (Accepted in Computing and Visualization in Science).

[11] J. Persson, L. von Sydow, P. Lötstedt and J. Tysk. Space-Time Adaptive Finite Difference Method for European Multi-Asset Option s,Dept. of Information Technology, Uppsala University, 2004.

[12] A. Rigal. High Order Schemes for Unsteady One-Dimensional Diffusion–Convection Problems, Journal of Computational Physics **114,** 59-76, 1994.

[13] Y. Saad, M.H. Schultz. `GMRES`: A generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM Journal of Scientific Computing, **7**:856-869, 1986.

[14] A.S. Sabau, P.E. Raad. Comparisons of compact and classical finite difference solutions of stiff problems on nonuniform grids, Computer & Fluids **28,** 361-384, 1999.

[15] R. Seydel. *Tools for Computational Finance.*, Springer Verlag, Second edition, 2003

[16] D. Tavella, C. Randall. *Pricing Financial Instruments- The Finite Difference Method*, John Wiley & Sons, Chichester, 2000.

[17] S. Zhao, G.W. Wei. Option valuation by using a discrete singular convolution, Applied Mathematics and Computation, article in press.