Pricing European Multi-asset Options Using a Space-time Adaptive FD-method

Jonas Persson   *Lina von Sydow*

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

Technical report 2003-059
December 2003
ISSN 1404-3203

Pricing European Multi-asset Options Using a Space-time Adaptive FD-method

Jonas Persson[*] Lina von Sydow

28 Nov. 2003

Abstract

In this paper we present an adaptive technique to solve the multi-dimensional Black-Scholes equation. The number of grid-points required for a given tolerance of the local discretization errors is reduced substantially when compared to a standard equidistant grid. Using our adaptive methods in space and time we have control of the local discretization errors and can refine the grid where needed for accuracy reasons.

Keywords: Finite-difference methods; Option pricing; Adaptive methods; Black-Scholes model

1 Introduction

In this paper, we consider the pricing of multi-asset European options in the generalized Black-Scholes model. The standard Black-Scholes model of the financial market consists of two assets – a bond B and a stock S – with dynamics given by

$$\begin{array}{rcl} dB(t) & = & rB(t)dt \\ dS(t) & = & \alpha(t,S(t))S(t)dt + \sigma(t,S(t))S(t)dW(t). \end{array} \tag{1}$$

Here W is a Wiener process, $r \in \mathbb{R}^+$ is the short rate of interest, $\alpha \in \mathbb{R}$ is the local mean of return of S and $\sigma \in \mathbb{R}$ is the volatility of S.

We consider contingent claims X on S of the form $X = \Phi(S(T))$. Here T is the time of maturity of the claim and Φ is the contract function. If we

[*]funded by FMB, the Graduate School in Mathematics and Computing

consider for instance a European call (put) option, the owner of the option has the right (but not the obligation) to buy (sell) the stock at a certain price K – the strike price – at time T. The contract function of a European call option is $\Phi(x) = \max(x - K, 0)$ (for the put option it is $\max(K - x, 0)$). We will denote $\max(a, 0)$ by a^+ for simplicity reasons. The problem is to determine the arbitrage free price of this contingent claim for every $t < T$. An arbitrage opportunity is when there is a possibility to make a risk-free profit with probability 1. If the market is free of arbitrage we say that the market is efficient.

Using the Feynman-Kac formula, see e.g. [8], we can derive the following formula for the arbitrage free price F at time t of the contingent claim $X = \Phi(S(T))$

$$F(t,s) = e^{-r(T-t)} E_{t,s}[\Phi(S(T))], \tag{2}$$

where S follows the following dynamics

$$\begin{aligned} dS(\tau) &= rS(t)d\tau + \sigma(\tau, S(\tau))dW(\tau), \\ S(t) &= s, \end{aligned} \tag{3}$$

and $E[\cdot]$ denotes expected value. Thus, we can determine the arbitrage free price of the contingent claim by calculating the expected value in (2) with a Monte-Carlo method and then discount the obtained price of the option to get today's price. This way of computing the arbitrage free price is widely used, see e.g. [3].

An alternative method to determine the arbitrage free price of a contingent claim is to solve the Black-Scholes equation

$$\begin{aligned} F_t(t,s) + rsF_s(t,s) + \tfrac{s^2\sigma^2}{2}(t,s)F_{ss}(t,s) &- rF(t,s) = 0, \\ F(T,s) &= \Phi(s), \end{aligned} \tag{4}$$

where a subscript denotes a partial derivative with respect to that variable. This partial differential equation can be derived under standard assumptions on the mathematical model and the financial market. By solving this equation backwards in time, we obtain the arbitrage free price of the contingent claim for all s that we are interested in. The interested reader is referred to [1] for a thorough discussion on this matter. This option pricing model was introduced by F. Black and M. Scholes in [2] and by R.C. Merton in [6], both in 1973.

The disadvantage with a Monte-Carlo method is that it converges very slowly – the statistical error for a standard Monte Carlo method is proportional to $M^{-1/2}$, where M is the number of simulations. Hence, we need to do a huge amount of simulations to reduce the statistical error. The main advantage of the method is that it is simple to generalize to higher dimensions, i.e. when the contingent claim depends on several underlying assets. Using e.g. finite differences to solve the PDE (4) the main advantage is that there is no statistical error. On the other hand, the main disadvantage is that for multi-dimensional problems, the number of degrees of freedom in the solution vector grows exponentially in d, where d is the number of underlying assets. If we have d underlying assets and discretize each dimension using N grid-points, the total number of unknowns is N^d. For a Monte-Carlo method the extension to multi-dimensional problems is straight forward. However, we believe that it is the Black-Scholes setting with finite differences that is the right problem to attack, up to 5-6 space dimensions. This strong belief is due to the extremely slow convergence in the statistical error for standard Monte-Carlo methods. Another advantage with the finite differences method is that we get the price of the option in a neighborhood of the current spot-price. These values are used when the derivative of the pricing function is needed to find hedging parameters. This comes for free with the finite difference method while for the Monte-Carlo method the problem must be solved several times. The solution of PDEs using finite differences was introduced in the financial literature by [10] in 1977 and has been widely used ever since.

In this paper we present an adaptive technique to reduce the number of grid-points to a minimum, still keeping the discretization error at a pre-described level. Our idea does not break the exponential grows in the unknowns but it reduces the coefficient from 1 to $\sim 2^{-d}$. It also makes sure that the grid-points are placed so that the discretization errors are kept at a certain level and the number of grid-points is at a minimum.

The adaptive process works like this:

Algorithm 1:

(i) Solve the problem once with a coarse grid and a fixed, large, time-step (giving low accuracy).

(ii) Create a new grid in space and time (to get required accuracy).

(iii) Solve again with the new grids.

This means that we have to solve the problem two times but the idea is that the first time the problem should be solved quite quickly and to very low accuracy but giving us a good estimate of how to place the grid-points for the second solve with more grid-points. The space- and time-adaptive processes are described in more detail below.

The outline of the paper is the following. In Section 2 we formulate the generalized version of the Black-Scholes partial differential equation, in Section 3 we show the numerical approximation methods used and in Section

4 and 5 respectively we show the adaptive techniques used in space and time. Section 6 and 7 finally presents the results of the numerical experiments and the conclusions drawn from them.

2 The multi-dimensional model-problem

Pricing options on several underlying assets is of great interest in the financial industry. We solve the generalized Black-Scholes partial differential equation

$$\begin{aligned} \frac{\partial F}{\partial t} + \sum_{i=1}^{d} r s_i \frac{\partial F}{\partial s_i} + \frac{1}{2}\sum_{i,j=1}^{d} [\sigma\sigma^*]_{ij} s_i s_j \frac{\partial^2 F}{\partial s_i \partial s_j} - rF &= 0 \\ F(T,s) &= \Phi(s). \end{aligned} \tag{5}$$

A simple example of an option on several underlying assets is the European basket option, this is the call version

$$\Phi(s) = \left(\frac{1}{d}\sum_{i=1}^{d} s_i - K\right)^{+}. \tag{6}$$

Before solving the partial differential equation (5) we transform it from a final-value-problem into a non-dimensional initial-value–problem. The transformation of the time-scale has the advantage that standard texts on time-integrators are applicable. The following transformations give the desired properties:

$$\begin{array}{lll} S = Kx, & \bar{r} = \frac{r}{\hat{\sigma}^2}, & KP(\hat{t},x) = F(t,s), \\ \bar{\sigma} = \frac{1}{\hat{\sigma}}\sigma, & \hat{t} = \frac{1}{2}\hat{\sigma}^2(T-t), & K\Psi(x) = \Phi(s), \end{array} \tag{7}$$

where $\hat{\sigma}$ is a constant chosen as $\max_{ij}\sigma_{ij}$. Other choices of $\hat{\sigma}$ to scale σ can be used. These transformations result in the following linear partial differential equation

$$\begin{aligned} \frac{\partial P}{\partial \hat{t}} = 2\bar{r}\sum_{i=1}^{d} x_i \frac{\partial P}{\partial x_i} + \sum_{i,j=1}^{d} [\bar{\sigma}\bar{\sigma}^*]_{ij} x_i \quad x_j \ \frac{\partial^2 P}{\partial x_i \partial x_j} - 2\bar{r}P \\ P(0,x) = \Psi(x). \end{aligned} \tag{8}$$

Let $\mathcal{L}$ be the operator

$$\mathcal{L} = 2\bar{r}\sum_{i=1}^{d} x_i \frac{\partial}{\partial x_i} + \sum_{i,j=1}^{d} [\bar{\sigma}\bar{\sigma}^*]_{ij} x_i x_j \frac{\partial^2}{\partial x_i \partial x_j} - 2\bar{r}. \tag{9}$$

The partial differential equation (8) can then be written as

$$\frac{\partial P}{\partial \hat{t}} = \mathcal{L}P \tag{10}$$

and in the next section we will show how to discretize the operator $\mathcal{L}$ using a finite difference discretization.

3 FD discretization

We make a semi-discretization in space by using centered second order finite differences on a structured but not equidistant grid, see Figure 1. The number

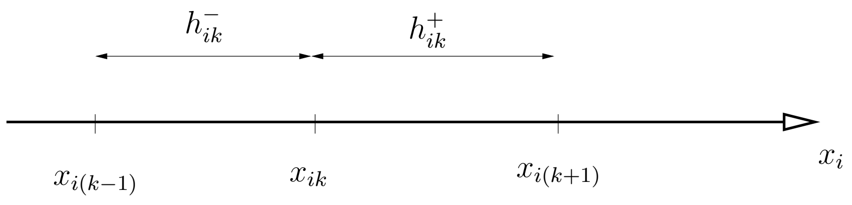

Figure 1: The x_i-axis. Here x_{ik}, $k = 1 \ldots m_i$ denotes the k:th node of dimension x_i.

of grid-points in the i:th dimension is $m_i, i = 1, \ldots, d$. If we let P_h be a vector of the lexicographically ordered unknowns, then

$$\frac{dP_h}{d\hat{t}} = A_h P_h \tag{11}$$

where A_h is the second order FD-discretization of $\mathcal{L}$. To make the following expressions more readable, the short notation

$$P(x_{ik}) = P(x_1, \ldots, x_{ik}, \ldots x_d)$$

and

$$P(x_{ik}, x_{jl}) = P(x_1, \ldots, x_{ik}, \ldots, x_{jl}, \ldots, x_d),$$

is used. For the first derivatives we have

$$\begin{aligned} \frac{\partial P(x_{ik})}{\partial x_i} &= P_{x_i}(x_{ik}) \approx \\ \approx a_{x_{ik}} P(x_{i(k+1)}) + b_{x_{ik}} P(x_{ik}) &+ c_{x_{ik}} P(x_{i(k-1)}), \end{aligned} \tag{12}$$

where

$$\begin{cases} a_{x_{ik}} = \frac{h_{ik}^-}{h_{ik}^-(h_{ik}^- + h_{ik}^+)}, \\ \\ b_{x_{ik}} = \frac{h_{ik}^+ - h_{ik}^-}{h_{ik}^- h_{ik}^+}, \\ \\ c_{x_{ik}} = -\frac{h_{ik}^+}{h_{ik}^-(h_{ik}^- + h_{ik}^+)}, \end{cases}$$

and for the second derivatives

$$\begin{aligned} \frac{\partial^2 P(x_{ik})}{\partial x_i^2} &= P_{x_i x_i}(x_{ik}) \approx \\ \approx a_{x_{ik}x_{ik}} P(x_{i(k+1)}) &+ b_{x_{ik}x_{ik}} P(x_{ik}) + \\ &+ c_{x_{ik}x_{ik}} P(x_{i(k-1)}), \end{aligned} \tag{13}$$

where

$$\begin{cases} a_{x_{ik}x_{ik}} = \frac{2}{h_{ik}^+(h_{ik}^- + h_{ik}^+)}, \\ \\ b_{x_{ik}x_{ik}} = -\frac{2}{h_{ik}^- h_{ik}^+}, \\ \\ c_{x_{ik}x_{ik}} = \frac{2}{h_{ik}^-(h_{ik}^- + h_{ik}^+)}, \end{cases}$$

$$\begin{aligned} &\left.\frac{\partial^2 P(x_{ik}, x_{jl})}{\partial x_i \partial x_j}\right|_{i \neq j} = P_{x_i x_j}(x_{ik}, x_{jl}) \approx \\ &\approx a_{x_{jl}} P_{x_i}(x_{j(l+1)}) + b_{x_{jl}} P_{x_i}(x_{jl}) + c_{x_{jl}} P_{x_i}(x_{j(l-1)}) \approx \\ &\approx a_{x_{jl}} \Big[a_{x_{ik}} P(x_{i(k+1)}, x_{j(l+1)}) + b_{x_{ik}} P(x_{ik}, x_{j(l+1)}) + \\ &+ c_{x_{ik}} P(x_{i(k-1)}, x_{j(l+1)}) \Big] + \\ &+ b_{x_{jl}} \Big[a_{x_{ik}} P(x_{i(k+1)}, x_{jl}) + b_{x_{ik}} P(x_{ik}, x_{jl}) + \\ &+ c_{x_{ik}} P(x_{i(k-1)}, x_{jl}) \Big] + \\ &+ c_{x_{jl}} \Big[a_{x_{ik}} P(x_{i(k+1)}, x_{j(l-1)}) + b_{x_{ik}} P(x_{ik}, x_{j(l-1)}) + \\ &+ c_{x_{ik}} P(x_{i(k-1)}, x_{j(l-1)}) \Big]. \end{aligned} \tag{14}$$

The matrix A_h in (11) is a very large, sparse matrix with the number of non-zeros of each row depending on the number of space dimensions, i.e. the number of stocks.
There are several possible boundary conditions that can be used for these kind of problems. The boundary condition we use on all boundaries is

$$\frac{\partial^2 P(x,\hat{t})}{\partial x_i^2} = 0, \tag{15}$$

which implies that the option price is nearly linear with respect to the spot price at the boundaries. Another boundary condition often used at the far-field boundary is

$$P(x_{max},\hat{t}) = x_{max} - Ke^{-2\bar{r}\hat{t}}$$

and $P(0,\hat{t}) = 0$ (in the one-dimensional case). Note that in financial problems there is often a dominant diffusion part of the equation, depending on the volatility, which is forgiving when having imperfect boundary conditions. Disturbances reflected back into the domain are smoothed out and does not destroy the solution as is the case in many hyperbolic problems. These and other boundary conditions are discussed in [11]. The boundary condition we use is applicable also for other options than European, as long as the price is nearly linear with respect to the spot price at the boundaries.

Something very crucial for finite difference discretizations of partial differential equations of convection-diffusion type is to avoid oscillatory solutions, see e.g. [7]. For simplicity we will only consider a one-dimensional problem with constant coefficients. Let us define the local mesh Péclet number Pe by

$$Pe = \frac{2\bar{r}h}{\bar{\sigma}^2 x} \tag{16}$$

In order to have no oscillatory solutions this mesh Péclet number has to be less than 2 for constant coefficient problems, see e.g. [7]. We will study this by examining the size of the two quotients $\bar{r}/\bar{\sigma}^2$ and h/x. The first quotient can be larger than one for some choices of $\bar{r}$ and $\bar{\sigma}$ but not arbitrarily large for realistic $\bar{r}$ and $\bar{\sigma}$. The second quotient is equal to one in the first inner grid-point but is lower than one further away from zero. The size of this quotient can be controlled by the largest allowed step-size, hence controlling Pe. If oscillations in the solution would cause problems it would be near the origin where the local mesh Péclet number can be larger than 2, but no such problems have been observed.

4 Space adaptivity

We will start by presenting the basic idea with the space adaptivity for a one-dimensional problem. At the end of this section we will demonstrate how this idea extends to a problem in more than one space-dimension. Assume that for any smooth solution $u(x)$ it holds that

$$A_h u_h = Au + \tau_h \tag{17}$$

where u_h is a vector of the unknowns and A_h is the discrete approximation of the operator $\mathcal{L}$ (the FD-matrix) as before. 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-lengths h. We further assume that we can approximate τ_h with the leading term

$$\begin{aligned} \tau_h &= h^p \eta(x) + \mathcal{O}(\text{h.o. terms}) = \\ &= h^p \left(r\phi(x) + \sigma^2 \psi(x) \right) + \mathcal{O}(\text{h.o. terms}). \end{aligned} \tag{18}$$

For a second order discretization in space $p = 2$. Then we define δ_h and δ_{2h} in the following way

$$\begin{aligned} \delta_h &= A_h u_h = Au + \tau_h \\ \delta_{2h} &= A_{2h} u_{2h} = Au + \tau_{2h}. \end{aligned} \tag{19}$$

The local discretization error of the approximation of order p on the fine grid with step-lengths h can then be approximated in the coarse grid with step-lengths $2h$ by using (18) and (19) omitting higher order terms yielding

$$\tau_h = \frac{1}{2^p - 1} (\delta_{2h} - \delta_h). \tag{20}$$

Note that only every second grid-point can be used in δ_h. The local discretization error τ_h is calculated in this way at several points in time. What one really would like to do is approximate the global discretization error. This, however, is more complicated, especially in several space dimensions. Instead we look at the local error at several, equally distributed points in time. The local τ_h is calculated at these times and the maximum of the absolute value of these τ_hs is found and used to calculate the new grid distribution in each dimension.

The adaptive process aims at creating a grid in which the grid-points are distributed efficiently in each space dimension. To do this we need to keep the discretization error at or below a predescribed level, here denoted by ϵ. To find an approximation of $\eta(x)$ in (18), we compute a solution using space-step $\bar{h}$ and then create a new matrix $A_{2\bar{h}}$ corresponding to the coarse grid with step-lengths $2\bar{h}$. The solution $u_{2\bar{h}}$ in Equation (19) is not computed, instead we use the restriction of the solution u_h to the coarse grid-points. From Equations (18), (19) and (20) we then obtain

$$\eta(x) = \frac{\tau_{\bar{h}}(x)}{\bar{h}^p(x)}.$$

So, in order to control the local discretization error and keep $|\tau_h(x)| \leq \epsilon$ for any $\epsilon > 0$, we have that

$$|\tau_h(x)| = |h^p(x)\eta(x)| \approx \left| h^p(x) \frac{\tau_{\bar{h}}(x)}{\bar{h}^p(x)} \right| \leq \epsilon \tag{21}$$

This gives us a way to choose the step-lengths for the adaptive grid. Using the initial grid $\bar{h}$ and the $\tau_{\bar{h}}$ calculated on this grid we can find a discrete function $h(x)$ as

$$h(x) = \bar{h}(x) \left(\frac{\epsilon}{|\tau_{\bar{h}}(x)|} \right)^{\frac{1}{p}}. \tag{22}$$

An example of such a function is depicted in Figure 2. There we also see how the new step-lengths are chosen. The method is simple, we start at the first old x-value and look at the discrete $h(x)$. The value of $h(x)$ in that point tells how big the time-step should be and we arrive at a new x-value. We again look at the value of the discrete $h(x)$ in this new x-value and get the new step, and so on. This is depicted in Figure 2. Of course, since $h(x)$ is a discrete function we must interpolate between known function values. A simple linear interpolation has been used.

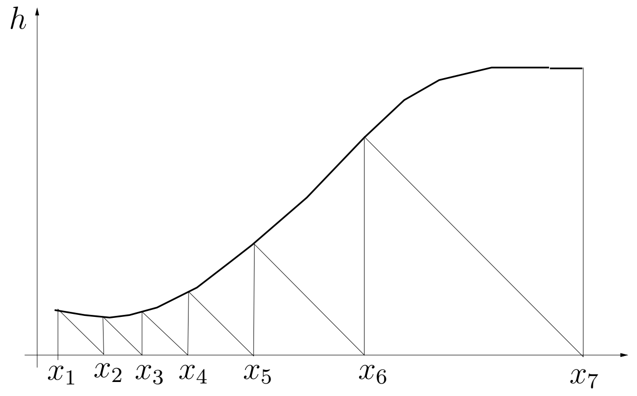

Figure 2: Creating the new adaptive grid in space using the discrete function $h(x)$.

Next, we again consider a second order discretization, but now of a two-dimensional problem. Now the discretization error τ_{h_1,h_2} can be expressed in the following way

$$\begin{aligned} \tau_{h_1,h_2} &= h_1^2(r\phi_1(x_1,x_2) + [\sigma\sigma^*]_{11}\psi_1(x_1,x_2)) + \\ &+ h_2^2(r\phi_2(x_1,x_2) + [\sigma\sigma^*]_{22}\psi_2(x_1,x_2)) + \\ &+ 2h_1h_2[\sigma\sigma^*]_{12}\psi_{12}(x_1,x_2) + \mathcal{O}(\text{h.o. terms}). \end{aligned} \tag{23}$$

We will assume that $[\sigma\sigma^*]_{ij} \ll [\sigma\sigma^*]_{ii}$ to simplify the theory. We then arrive in the following approximation

$$\tau_{h_1,h_2} \approx h_1^2\eta_1(x_1,x_2) + h_2^2\eta_2(x_1,x_2) = \tau_{h_1} + \tau_{h_2}$$

Note though that this is for the theoretic derivation of the step-length function. In practice the mixed terms will have influence of the τ. Proceeding as in the one-dimensional case we then get that we can estimate $\eta_1(x_1,x_2)$ by performing computations with $\bar{h}_1$ and $2\bar{h}_1$ and similarly for $\eta_2(x_1,x_2)$. Then, we define

$$\begin{aligned} \hat{\tau}_{\bar{h}_1}(x_1) &= \max_{x_2} |\tau_{\bar{h}_1}(x_1,x_2)|, \\ \hat{\tau}_{\bar{h}_2}(x_2) &= \max_{x_1} |\tau_{\bar{h}_2}(x_1,x_2)|. \end{aligned} \tag{24}$$

I.e., for each dimension i we take the maximum over the absolute value of $\tau_{\bar{h}_i}(x_1,\ldots,x_i,\ldots,x_d)$ over all other dimensions. In the more general case with d assets we have

$$\hat{\tau}_{\bar{h}_i}(x_i) = \max_{x_{I\setminus\{i\}}} |\tau_{\bar{h}_i}(x_1,\ldots,x_d)| \quad \forall i \in I = \{1,\ldots,d\}.$$

Finally, by prescribing the maximal discretization error ϵ_d in each space-direction, we can compute a new spatial grid from

$$\begin{aligned} h_1(x_1) &= \bar{h}_1(x_1)\left(\frac{\epsilon_1}{\hat{\tau}_{\bar{h}_1}(x_1)}\right)^{\frac{1}{2}}, \\ h_2(x_2) &= \bar{h}_2(x_2)\left(\frac{\epsilon_2}{\hat{\tau}_{\bar{h}_2}(x_2)}\right)^{\frac{1}{2}}. \end{aligned} \tag{25}$$

The generalization to higher dimensions than two is now straight-forward. Since $\hat{\tau}(x)$ in some cases is non-smooth we apply a weighted mean-value filter on these approximations of the local discretization errors. The principle is simple, in each dimension i, the value $\tau_h(x_i^j)$ in point j in that dimension is changed according to

$$\hat{\tau}_{\bar{h}_i}(x_{i,j}) = (\hat{\tau}_h(x_{i,j-1}) + 2\hat{\tau}_h(x_{i,j}) + \hat{\tau}_h(x_{i,j+1}))/4. \tag{26}$$

At the end-points $\hat{\tau}_{\bar{h}_i}(x_i)$ is then extrapolated using zeroth order extrapolation. This filter is repeated a number of times to ensure a smooth $\hat{\tau}_{\bar{h}_i}$.

Finally, we will add an extra parameter γ in order to make sure that the space-step is not chosen too large when the discretization error is very small. For the one-dimensional problem we use

$$h(x) = \bar{h}(x) \left(\frac{\epsilon}{\epsilon\gamma + |\tau_{\bar{h}}(x)|} \right)^{\frac{1}{p}}, \tag{27}$$

where γ is a chosen constant. Throughout the calculations we have used $\gamma = 0.01$. How to calculate the discrete function $h(x)$ is generalized to higher dimensions in an obvious manner.

5 Time-integration and time-adaptivity

For time-integration we use the backward differentiation formula of order two (BDF-2), see [4]. In the adaptive form the coefficients have to be modified if the time-step is changing from one step to the next. Since BDF-2 is A-stable the time-step has to be changed due to accuracy only, not for stability reasons. A disadvantage though is that a system of equations must be solved in each time-step and for several space dimensions these systems, though sparse, are very large.
For the adaption of the time-grid we use an explicit predictor and an implicit corrector (the BDF-2) to find an approximation of the local truncation error in BDF-2 in the same way as Lötstedt et. al. in [5]. But, where they continuously change the time-step as τ_k is computed at each time-step, we store the information about τ_k and choose the new time-steps after the first solve (i) in **Algorithm 1** is completed. This is because we have to solve once for the space adaption anyway. Let us assume that we want to approximate the solution u^n at time $\hat{t}_n$, given that we know the solutions u^{n-1} and u^{n-2} at the times $\hat{t}_{n-1}$ and $\hat{t}_{n-2}$.
As predictor we choose the explicit method:

$$\begin{aligned} &\tilde{\alpha}_0^n \tilde{u}^n = k_n \mathcal{L}(u^{n-1}) - \tilde{\alpha}_1^n u^{n-1} - \tilde{\alpha}_2^n u^{n-2} \\ &\tilde{\alpha}_0^n = 1/(1+\theta^n), \\ &\tilde{\alpha}_1^n = \theta^n - 1, \\ &\tilde{\alpha}_2^n = -(\theta^n)^2/(1+\theta^n) \end{aligned} \tag{28}$$

with the local truncation error

$$\begin{aligned} &u(\hat{t}_n) - \tilde{u}^n = C_P(\theta^n) k_n^3 u^{'''} + O(k^4) \\ &C_P(\theta^n) = (1 + 1/\theta^n)/6. \end{aligned} \tag{29}$$

Note that $\theta^n = \frac{k_n}{k_{n-1}}$, where k_n is the time-step between $\hat{t}_{n-1}$ and $\hat{t}_n$.
As corrector we choose the implicit method BDF-2:

$$\begin{aligned} &\alpha_0^n u^n = k_n \mathcal{L}(u^n) - \alpha_1^n u^{n-1} - \alpha_2^n u^{n-2} \\ &\alpha_0^n = (1+2\theta^n)/(1+\theta^n), \\ &\alpha_1^n = -(1+\theta^n), \\ &\alpha_2^n = (\theta^n)^2/(1+\theta^n) \end{aligned} \tag{30}$$

with the local truncation error

$$\begin{aligned} &u(\hat{t}_n) - u^n = C_I(\theta^n) k_n^3 u^{'''} + O(k^4) \\ &C_I(\theta^n) = -(1+\theta^n)^2/(6\theta^n(1+2\theta^n)) \end{aligned} \tag{31}$$

Since BDF-2 is a multi-step method we need to use a different method for the first time-step. We have used Euler-backward (take $\alpha_0^1 = 1, \alpha_1^1 = -1$ and $\alpha_2 = 0$ in Equation (30)) for the first time-step.

The leading term $C_I(\theta^n) k_n^3 u^{'''}$ in the local error in time in Equation (31) can be approximated by computing the difference between the numerical solution u^n computed by BDF-2 and the explicit solution $\tilde{u}^n$ computed with the method in (28). The solution $\tilde{u}^n$ is also used as initial solution at each time-step for the iterative solution of the linear systems of equations arising from the implicit solver. The solution of these systems of equations are addressed later in this section.
Now we can infer from (29) and (31) that the leading error term in the time discretization (30) can be approximated by

$$\tau_k^n = -\alpha_0^n C_I k_n^2 u^{'''} \approx \alpha_0 C_I (u^n - \tilde{u}^n)/(k_n(C_I - C_P)). \tag{32}$$

When the approximation of the discretization error is found the step-lengths in time are chosen in the same way as the step-lengths in space. We assume that the local discretization error in time can be approximated by the leading term

$$\tau_k(\hat{t}) = k^q \xi(\hat{t}) + \mathcal{O}(\text{h.o. terms}). \tag{33}$$

For a second order discretization in time $q = 2$. To keep the local discretiaz-tion error $|\tau_k(\hat{t})| \leq \epsilon_t$ for any $\epsilon_t > 0$ we find an approximation of $\xi(\hat{t})$ using an initial time-grid $\bar{k}(\hat{t})$ and $\tau_{\bar{k}}(\hat{t})$ computed from Equation (32), we get

$$\xi(\hat{t}) = \frac{\tau_{\bar{k}}}{\bar{k}^q(\hat{t})}.$$

In this way we can choose the step-lengths in time using the equation

$$k(\hat{t}) = \bar{k}(\hat{t}) \left(\frac{\epsilon_t}{|\tau_{\bar{k}}(\hat{t})|} \right)^{\frac{1}{q}}. \tag{34}$$

Please note that $\tau_{\bar{k}}$ really also depends on x but since we cannot, at least not in an easy way, have different time-steps for different x-values at the same point in time we only use the maximum value of τ_k over all space dimensions for each time-step. As in the case with space adaptivity, the approximation of the local discretization error τ_k in time can be non-smooth and thus give rise to too abrupt changes in the chosen step-lengths. A weighted mean-value filter is applied in the same way as in space. The value of τ_k in each point (except the first and last point in time) is modified according to

$$\tau_k(\hat{t^n}) = (\tau_k(\hat{t}^{n-1}) + 2\tau_k(\hat{t^n}) + \tau_k(\hat{t}^{n+1}))/4. \tag{35}$$

Again, as a security measure, we use a parameter to avoid using a too large time-step when τ_k is very small

$$k(\hat{t}) = \bar{k}(\hat{t}) \left(\frac{\epsilon_t}{\epsilon_t \beta + |\tau_{\bar{k}}(\hat{t})|} \right)^{\frac{1}{q}}. \tag{36}$$

We have used $\beta = 0.01$ in our calculations.

Since BDF-2, see (30), is an implicit method in time, we must solve large, linear, sparse systems of equations in each time-step. For the solution of these systems the iterative method `GMRES`, see [9], has been used. As described earlier, the solution computed with the explicit method is used as initial-guess in each time-step. The `GMRES`-iterations are stopped when the relative residual norm is small enough. To be efficient and memory lean, the iterative method is restarted after 6 iterations. To keep the number of iterations low, Matlabs built-in incomplete LU preconditioner (ILU) has been used. The ILU-factors, L and U, are only computed at the first time-step for efficiency reasons. However, since the explicit solution in most cases is a very good initial guess for `GMRES` the convergence of the iterative method is fast (few iterations) even though the ILU factors aren't recomputed. This will be addressed again in the following section. In a forthcoming paper we will study the solution of these linear systems in more detail.

6 Numerical results

The adaptive methods have been studied in several test-cases for one, two and four underlying assets. The contract function is the European Basket option, Equation (6), of the mean of the assets. The main focus in the numerical experiments have been on the performance of the adaptive techniques and their benefit to find accurate solutions efficiently.

The following parameters have been used as the standard setup: the local mean rate of return r have been fixed at 0.05 and the volatility matrix σ have the value 0.3 on the diagonal and 0.05 just above and below the diagonal, and all other entries are zero. The volatility matrix could without complications depend on the space variables but not on time. This is because the FD-matrix then need to be recomputed at each time-step which would be possible but very expensive. All computations have been performed with the transformed PDE, Equation (8), in forward time from zero to T instead of in backward time from T to zero. In all numerical examples we have $T = 0.1$. Note also that the local discretization error in space has been measured at three times for all examples, $[T/3 \quad 2T/3 \quad T]$ as described in Section 4 and that T is different in transformed and untransformed time. Since there is a price on the option for arbitrarily large stock-prices, at least in theory, we must choose an $S_{\max}$ to truncate the domain. In all computations we have used an extension of the general "rule of thumb", $S_{\max} = 4dK$, i.e. we truncate the domain at four times the strike-price multiplied by the number of dimensions d. The reason for multiplying by d is to have the far-field boundary at four times the location of the discontinuity of the derivative of the initial function Φ in each dimension.

6.1 Some one-dimensional numerical examples

In the first test-example in one space dimension we have used few equidistantly distributed grid-points to calculate an approximation of the local discretization error. This rather coarse approximation is then used to distribute grid-points demanding a certain level of the local discretization error. After the problem has been solved on the adaptive grid we solve the problem once more to find an equidistant grid on which the local discretization errors on the adaptive and equidistant grids match in size. The tolerance levels 10^{-3} and 10^{-4} have been studied. The adaptive method in time has not been used in this test-example, the time-step has been held constant at $k = 0.001$. The result is displayed in Table 1.

This numerical example shows that giving the method a small number of equidistantly distributed grid-points, N_e in Table 1, we create an adaptive grid with N_a grid-points on which the local discretization errors are at the level of the required tolerance. In Table 1 we also see the number of equidistant grid-points that are needed to get the same low local discretization error as on the adaptive grid. We have also computed the true error for

Adaptive	$N_e \Rightarrow$	N_a	$\lvert\tau_h(T/3)\rvert$	$\lvert\tau_h(2T/3)\rvert$	$\lvert\tau_h(T)\rvert$
10^{-3}	29	73	2.61E-3	1.01E-3	6.1E-4
	33	69	2.65E-3	1.04E-3	6.3E-4
	37	69	2.54E-3	9.9E-4	6.2E-4
Equidist.	117	–	2.64E-3	1.04E-3	6.3E-4
Adaptive	$N_e \Rightarrow$	N_a	$\lvert\tau_h(T/3)\rvert$	$\lvert\tau_h(2T/3)\rvert$	$\lvert\tau_h(T)\rvert$
10^{-4}	61	197	2.10E-4	8.3E-5	1.2E-4
	69	193	2.05E-4	8.2E-5	1.3E-4
	77	193	1.98E-4	9.2E-5	1.4E-4
Equidist.	389	–	2.34E-4	9.3E-5	5.7E-5

Table 1: Maximum of the absolute value of the local discretization errors on adaptive and equidistant grids. Three different initial equidistant initial distributions N_e result in adaptive grids with N_a grid-points. Results for two tolerance levels, 10^{-3} and 10^{-4}, are presented.

our problem using the exact solution for the European call

$$\begin{aligned} F(t,s) &= s\mathcal{N}(d_1(t,s)) - Ke^{-rt}\mathcal{N}(d_2(t,s)) \\ \text{where} & \\ d_1(t,s) &= \frac{ln(s/K)+(r+\frac{1}{2}\sigma^2)t}{\sigma\sqrt{t}} \\ \text{and} & \\ d_2(t,s) &= d_1(t,s) - \sigma\sqrt{t} \end{aligned} \tag{37}$$

which is available for the one-dimensional problem under certain conditions, see e.g. [8]. Note that $\mathcal{N}(x)$ is the standard Gaussian cumulative distribution function and t here is the time left to maturity. Our solution is transformed back from $P(\hat{t},x)$ to $F(t,s)$ according to (7) in Section 2 and then compared to the exact solution to get the error. This error at time T, is shown for the adaptive grid with 69 grid-points and the equidistant grid with 117 grid-points in Figure 3. We see that the error on our adaptive grid is about the same or lower than the error on the equidistant grids, even though less grid-points has been used. Note that the errors are large at the right part of the domain but when pricing options one is mainly interested in the domain close to the strike-price which was 30 in this case. Choosing a larger S_{max} would decrease the error at the far-field boundary but give our adaptive method an unfair advantage compared to the equidistant grid. The comparison would be unfair since our adaptive method will use very few grid-points when the solution is very smooth but using an equidistant grid one can not exploit the smoothness of the solution in that way.

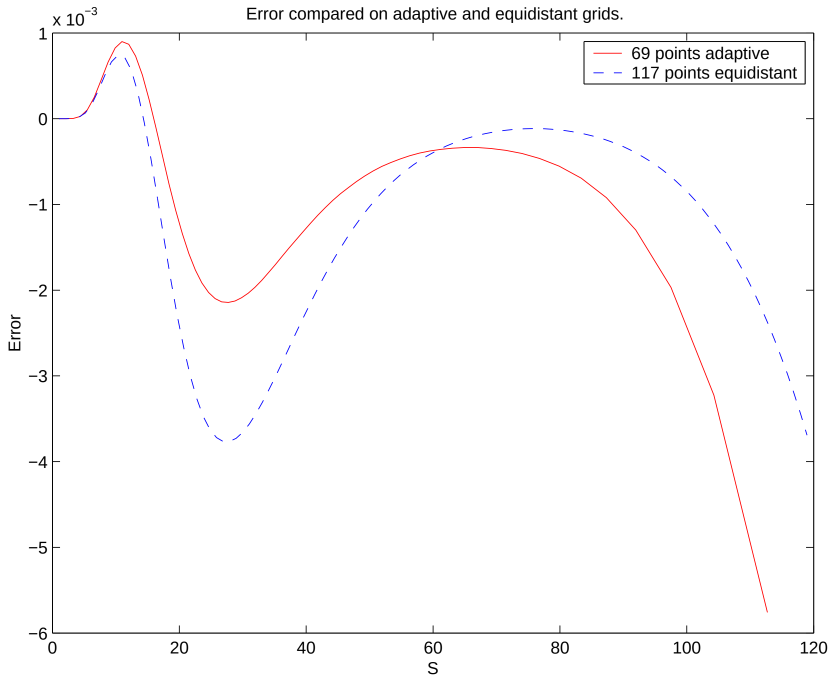

Figure 3: The errors with 69 adaptively and 117 equidistantly placed grid-points

6.2 Two-dimensional numerical examples

In the first numerical example in two space dimensions we repeat the experiments from one space dimension. I.e., we use few grid-points in each dimension to get an approximation of the local truncation errors. Using these approximations we then distribute the grid-points in each dimension according to Equation (27) to achieve the desired level of the local discretization error. The problem is then solved again with the new grid (step (iii) in **Algorithm 1**). To check that the method works we also measure the discretization error after we have solved the problem on the new grid. This is not necessary for the solution of the problem but it is a good test that the method produces the expected result and the extra cost is not so large. Note that the adaptive algorithm in time has not been used in these experiments, it has been tested separately. In Table 2 we see the results from the two-dimensional experiments. We see that using relatively few equidistantly distributed grid-points in each dimension we get an adaptive grid on which the measured local discretization errors are of the level that has been prescribed. The levels set in this experiment was 10^{-3} and 10^{-4} and the local

Adapt.	$N_e^2 \Rightarrow$	N_a^2	$\lvert\tau_h(T/3)\rvert$	$\lvert\tau_h(2T/3)\rvert$	$\lvert\tau_h(T)\rvert$
10^{-3}	33^2	89^2	1.50E-3	5.85E-4	6.61E-4
	37^2	89^2	1.50E-3	5.82E-4	8.12E-4
	41^2	85^2	1.52E-3	5.92E-4	8.77E-4
Equid.	157^2	–	1.81E-3	7.13E-4	4.26E-4
Adapt.	$N_e \Rightarrow$	N_a	$\lvert\tau_h(T/3)\rvert$	$\lvert\tau_h(2T/3)\rvert$	$\lvert\tau_h(T)\rvert$
10^{-4}	61^2	249^2	1.45E-4	9.22E-5	1.43E-4
	69^2	249^2	1.55E-4	1.04E-4	1.52E-4
	77^2	249^2	1.63E-4	1.08E-4	1.57E-4
Equid.	441^2	–	2.23E-4	8.93E-5	5.3E-5

Table 2: Maximum of the absolute value of the local discretization errors on the adaptive grids. Three different initial equidistant initial distributions N_e^2 result in adaptive grids with N_a^2 grid-points. Results for two tolerance levels, 10^{-3} and 10^{-4}, are presented.

discretization errors on the adaptive grids were of this size. If the computer memory is of no problem one can just set the desired level of local accuracy and let the computer compute the solution with this accuracy. If the size of the computer memory is a problem and we want to use it efficiently we can instead study what local accuracy we can get out of a certain number of grid-points.

The test has been performed in two space dimensions. We experimentally find the number of grid-points in each dimension that, given a certain tolerance level, returns the same number of grid-points in each dimension. As in the first two-dimensional example we have looked at two different levels of local discretization errors. In Table 3 we display the maximum of the

Tol ϵ_1		Before adaption		After adaption	
		$N_e \times N_e$	$\max(\lvert\tau_h\rvert)$	$N_a \times N_a$	$\max(\lvert\tau_h\rvert)$
10^{-3}	$T/3$	81×81	7.0E-3	81×81	1.6E-3
	$2T/3$	-"-	2.7E-3	-"-	8.8E-4
	T	-"-	1.6E-3	-"-	1.2E-3
10^{-4}	$T/3$	241×241	7.6E-4	241×241	2.7E-4
	$2T/3$	-"-	3.0E-4	-"-	1.6E-4
	T	-"-	1.8E-4	-"-	1.8E-4

Table 3: Local discretization errors before and after adaption in two space dimensions. Maximum of the local discretization error at time $T/3$, $2T/3$ and T.

absolute value of the approximations of the local dicscretization errors at the times $T/3$, $2T/3$ and T. This example shows that with the same number of grid-points in an adaptive grid as in an equidistant, it is possible to get lower local discretization errors using our adaptive grid. I.e., we can keep the memory required at a constant level and get better accuracy using our adaptive technique than using the standard equidistant grid.

6.3 A four-dimensional numerical example

In four space dimensions only one simple test-case has been performed. We set the desired level of the local discretization error to be $5 \cdot 10^{-3}$ and start with 25 grid-points in each dimension. The strike-price was set to $K = 10$. The local discretization errors are not the same in every dimension

Tol ϵ_1		Before adaption		After adaption	
		Dim.	$\max(\lvert\tau_h\rvert)$	Dim.	$\max(\lvert\tau_h\rvert)$
$5 * 10^{-3}$	$T/3$	1	0.06513	1	0.008723
	-”-	2	0.07333	2	0.007684
	-”-	3	0.07333	3	0.007684
	-”-	4	0.06513	4	0.008723
	$2T/3$	1	0.021151	1	0.003671
	-”-	2	0.023070	2	0.003180
	-”-	3	0.023070	3	0.003180
	-”-	4	0.021151	4	0.003671
	T	1	0.009260	1	0.002416
	-”-	2	0.010189	2	0.002309
	-”-	3	0.010189	3	0.002309
	-”-	4	0.009260	4	0.002416

Table 4: A four-dimensional example. With 25^4 grid-points before adaption and [37 41 41 37] grid-points after adaption.

because of the structure of the σ-matrix. With our choice of σ-matrix the coefficients in front of the mixed-derivatives will are different and the problem is not symmetric in the dimensions. Therefore τ_h will differ in the results for different dimensions and the adaptive grid may have different number of grid-points in each dimension. In Table 4 we see that given a coarse equidistant initial distribution of the grid-points we can really lower the local discretization errors. The result on the adaptive grid is much better then what could be expected by refining the equidistant grid from 25^4 grid-points to [37 41 41 37] grid-points. This has been done and the result is shown in Table 5. Comparing the second column, 'After adaption', in Table 4 with

Tol ϵ_1		Before adaption	
		Dim.	$\max(\lvert\tau_h\rvert)$
$5 * 10^{-3}$	$T/3$	1	0.04701
	-"-	2	0.03782
	-"-	3	0.03782
	-"-	4	0.04701
	$2T/3$	1	0.01548
	-"-	2	0.01253
	-"-	3	0.01253
	-"-	4	0.01548
	T	1	0.00913
	-"-	2	0.00774
	-"-	3	0.00774
	-"-	4	0.00913

Table 5: The maximum of the absolute value of the local discretization errors on an equidistant grid with [37 41 41 37] grid-points.

the results in Table 5 it is evident that the corresponding local discretization errors are much lower, a factor three to five.

6.4 Testing the adaption in time

The local discretization error in time is approximated as in Equation (32) in Section 5 and the new time-steps are chosen according to Equation (34). The idea with the adaptive algorithm in time is to approximate the local discretization error on a coarse grid, with large time-steps, to get an idea of how the time-steps should be chosen for the second solve (step (iii) in **Algorithm 1**) when adaptive grids in both space and time is used. When the local discretization errors have been estimated a new time-grid is created to get required accuracy in the local errors. A few time-steps in the first solve is of course an extra cost in comparison to using a non-adaptive method solving the problem with just one solve. However, when solving just once, one has to guess at an appropriate time-step or have before hand knowledge of the problem. This can be difficult and in general give too large errors in time, extra work due to too small time-steps or even instabilities if an explicit method is used. If the time-step turns out to be too large, the problem nevertheless has to be solved again with a smaller time-step. With our method a level for the local discretization error is chosen by the user. In Figure 4 we see that the discretization error is too large in the beginning. This is because of the discontinuity of the space derivative in the first time-step.

In the first test-example we used 20 equally sized time-steps to get from 0 to 0.1. The example-problem was two-dimensional and 89 equidistantly distributed grid-points was used in each space-dimension. The adaption in space generated an adaptive grid with 85 grid-points in each dimension. The tolerance of the local discretization error in time and space was set to 10^{-3}. All other parameters was the same as the standard setup. The approximation of τ_k on the equidistant and on the adaptive time-grid is shown in Figure 4. From the approximation of τ_k on the coarse equidistant grid we calculate the discrete function $k(t)$, depicted in Figure 5. Choosing the time-step from the discrete function $k(t)$ generates an adaptive grid with 148 time-steps. In Figure 5 we also see that the last time-step has to be smaller than the next to last time-step since we have to reach $T = 0.1$ exactly. When solving the problem again with the adaptive grid we compute the approximation of the local discretization error on the refined grid. The result is shown in Figure 4. There we also see that the smaller last time-step also give a smaller τ_k at that point in time.

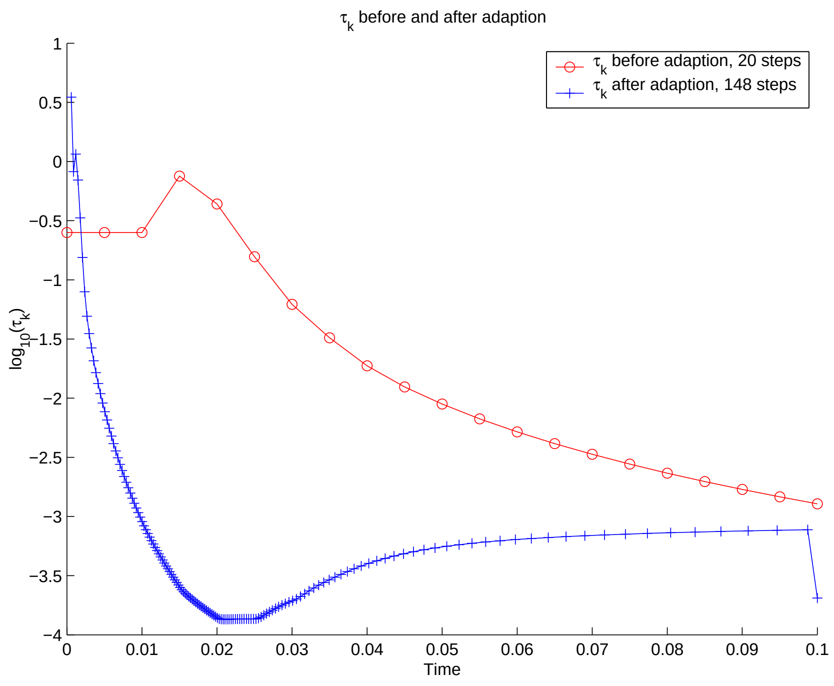

Figure 4: The approximation of the local discretization errors in time on equidistant and non-equidistant grid. The chosen level of the local discretization error in this case was 10^{-3}.

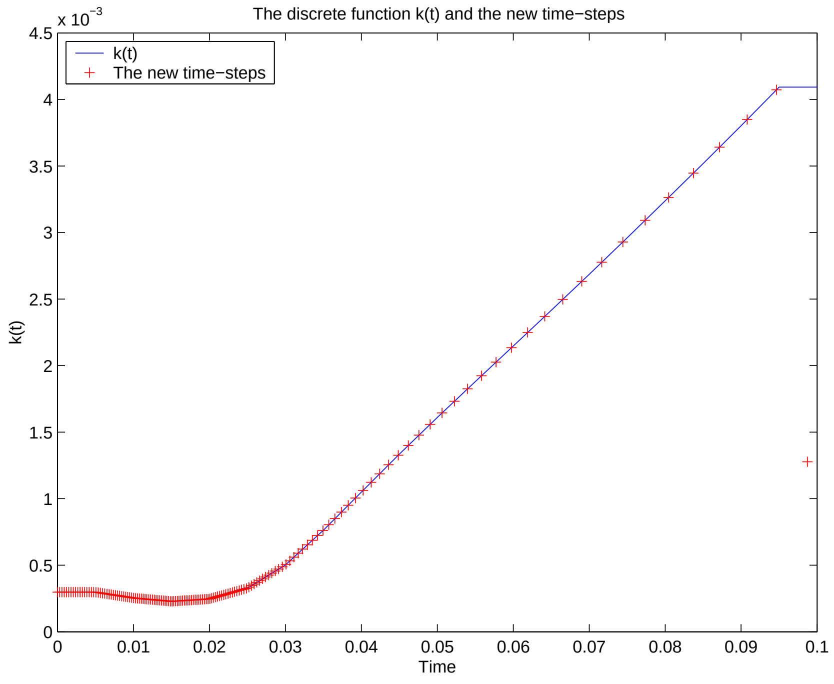

Figure 5: The discrete time-step function $k(t)$.

The large system of equations mentioned in Section 5 is solved with the iterative method `GMRES`. In the first time-step with backward Euler, on both equidistant and adaptive grid, no preconditioner has been used but for the rest of the time-steps, where BDF-2 is used, an incomplete LU-factorization has been used as preconditioner to speed up the convergence. The ILU-factors L and U are computed only for the first time-step with BDF-2, they are not recomputed when the length of the time-step changes between two steps since it has turned out in the experiments that `GMRES` converges fast anyway. The explicit solution (28) used as initial guess for the iterative solver is usually so good that only a few iterations is needed. An example is shown in Figure 6 where we have solved a two-dimensional problem on an adapted grid with 89×89 grid-points. We see that the total number of iterations in `GMRES` is very low and that the number of iterations grows only at the end of the computation. This is probably due to the large time-steps used here and the now inaccurate ILU-factors.

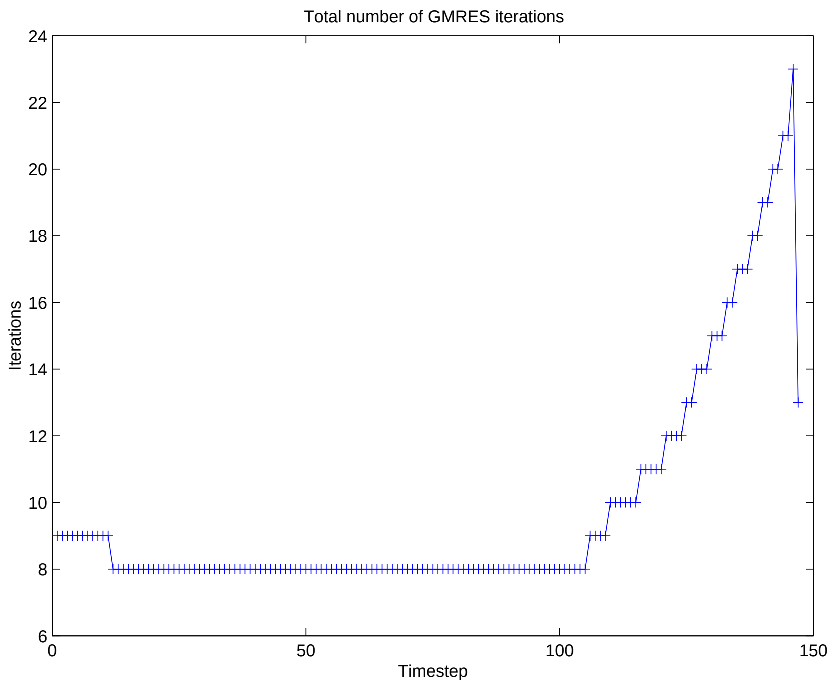

Figure 6: The total number of iterations in each time-step in `GMRES` for a two-dimensional example with 85×85 grid-points.

7 Conclusions

From our numerical tests we conclude that from a coarse initial grid, our adaptive method in space can generate an adaptive grid on which we can control the local discretization errors and keep them at a predefined levels. When comparing our method to solving on an equidistant grid we find that it requires almost twice as many grid-points in each space dimension to get the same size of the local discretization errors. Note that using our method one does not have to guess at the required number of grid-points to get a certain local discretization error, the method takes care of that for you.

If memory is the limitation, our adaptive technique can be used to efficiently use the amount of memory available and increase the accuracy compared to an equidistant grid. This is accomplished by rearranging the grid-points to make the local discretization errors smaller.

The adaptive algorithm in time chooses time-steps to keep the local discretization error in time below or at a predefined level. The total computational cost from solving the problem twice on a coarse and a refined time-grid will in most cases be lower than finding an appropriate time-step by solving the problem several times and experimentally finding a suitable time-step.

In a forthcoming paper we will study discretizations of higher order than 2, which will further reduce the number of grid-points needed. Altogether our space-time adaptive method is simple and easy to implement, even for a multi-dimensional problem.

References

[1] T. Björk. Arbitrage Theory in Continuous Time, Oxford University Press, New York 1998

[2] F. Black, M. Scholes. The pricing of options and corporate liabilities. Journal of Political Economy, **81**:637–659, 1973

[3] P.P. Boyle, M. Broadie, P. Glasserman. Monte Carlo methods for security pricing. Journal of Economic Dynamics and Control, **21**:1267–1321, 1997

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

[5] P. Lötstedt, S. Söderberg, A. Ramage, L. Hemmingsson-Frändén. Implicit Solution of Hyperbolic Equations with Space-Time Adaptivity. BIT, **42:1**:128–153, 2002

[6] R.C. Merton. Theory of rational option pricing. Bell J. Econom. Manag. Sci., **4**:141–183, 1973

[7] K.W. Morton. Numerical solution of convection-diffusion problems. Chapman & Hall, London 1996.

[8] M. Musiela, M. Rutkowski. Martingale Methods in Financial Modelling. Springer-Verlag, 1997

[9] Y. Saad, M.H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. Siam J. of Sci. Comp., **7**:856–869, 1986

[10] E.S. Schwartz. The valuation of warrants: implementing a new approach. Journal of Financial Economics, **4**:79–93, 1977

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

Recent technical reports from the Department of Information Technology

2003-039	Therese Berg, Bengt Jonsson, Martin Leucker, and Mayank Saksena: *Insights to Angluin's Learning*
2003-040	Wendy Kress: *Error Estimates for Deferred Correction Methods in Time*
2003-041	Wendy Kress: *A Compact Fourth Order Time Discretization Method for the Wave Equation*
2003-042	Gerardo Schneider: *Invariance Kernels of Polygonal Differential Inclusions*
2003-043	Kajsa Ljungberg, Sverker Holmgren, and Örjan Carlborg: *Simultaneous Search for Multiple QTL Using the Global Optimization Algorithm DIRECT*
2003-044	Dan Wallin, Henrik Johansson, and Sverker Holmgren: *Cache Memory Behavior of Advanced PDE Solvers*
2003-045	Sven-Olof Nyström: *A Polyvariant Type Analysis for Erlang*
2003-046	Martin Karlsson: *A Power-Ef£cient Alternative to Highly Associative Caches*
2003-047	Jimmy Flink: *Simuleringsmotor för tågtra£k med stöd för experimentell kon£guration*
2003-048	Timour Katchaounov and Tore Risch: *Interface Capabilities for Query Processing in Peer Mediator Systems*
2003-049	Martin Nilsson: *A Parallel Shared Memory Implementation of the Fast Multipole Method for Electromagnetics*
2003-050	Alexandre David: *Hierarchical Modeling and Analysis of Timed Systems*
2003-051	Pavel Krcal and Wang Yi: *Decidable and Undecidable Problems in Schedulability Analysis Using Timed Automata*
2003-052	Magnus Svärd and Jan Nordström: *Well Posed Boundary Conditions for the Navier-Stokes Equations*
2003-053	Erik Bängtsson and Maya Neytcheva: *Approaches to Reduce the Computational Cost when Solving Linear Systems of Equations Arising in Boundary Element Method Discretizations*
2003-054	Martin Nilsson: *Stability of the Fast Multipole Method for Helmholtz Equation in Three Dimensions*
2003-055	Martin Nilsson: *Rapid Solution of Parameter-dependent Linear Systems for Electromagnetic Problems in the Frequency Domain*
2003-057	Erik Berg: *Low-Overhead Spatial and Temporal Data Locality Analysis*
2003-058	Erik Berg: *StatCache: A Probabilistic Approach to Ef£cient and Accurate Data Locality Analysis*
2003-059	Jonas Persson and Lina von Sydow: *Pricing European Multi-asset Options Using a Space-time Adaptive FD-method*

UPPSALA
UNIVERSITET

December 2003
ISSN 1404-3203
http://www.it.uu.se/