EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Assignment #2: Implicit Finite Difference Methods

B473 Numerical Methods in Finance
Prof. Dr.-Ing. Rainer Schöbel

Winter Term 2016-2017

Konstantin Smirnov
Student-ID: 3980253
Economics and Finance
Tübingen, January 16, 2017

List of Figures

1 European Put value for different S - Implicit Scheme vs. Theoretical 5
2 Implicit Scheme - Error vs. Stock Price 6
3 Implicit Scheme - Error vs. Stock Price vs. Time to Maturity 6
4 European Put value for different S: Crank-Nicolson Scheme vs. Theoretical . 11
5 Crank-Nicolson Scheme - Error vs. Stock Price 11
6 Crank-Nicolson Scheme - Error vs. Stock Price vs. Time to Maturity 12
7 2D-Errors - Standard Schemes vs. Extrapolation 13
8 Standard Implicit Scheme vs. Extrapolation (3D-Errors) 13
9 Standard Crank-Nicolson Scheme vs. Extrapolation (3D-Errors) 14
10 Errors Development for increasing step-sizes MxN (log-log-scale)) 15
11 Trade-Off: Computational Time vs. Error 15
12 Errors Development for increasing step-sizes MxN (log-log-scale)) 16

Contents

1 Introduction **1**

2 Implicit Finite Difference Method for an European Put **1**
2.1 Theoretical Framework 1
2.1.1 Defining a Grid 1
2.1.2 Difference Approximation of the Implicit Scheme 2
2.1.3 Truncation Error 3
2.1.4 Boundary Conditions 3
2.1.5 Matrix Notation and Solution of the System 4
2.2 Discussion of Results 5
2.3 Technical Implementation 7

3 Crank-Nicolson Finite Difference Method for an European Put **7**
3.1 Theoretical Framework 8
3.1.1 Difference Approximation of the Crank-Nicolson Scheme 8
3.1.2 Truncation Error 9
3.1.3 Boundary Conditions 9
3.1.4 Matrix Notation and Solution of the System 10
3.2 Discussion of Results 11
3.3 Technical Implementation 12

4 Richardson extrapolation method **12**
4.1 Discussion of Results 13
4.2 Technical Implementation 14

5 Efficiency Discussion: Accuracy and Computational Effort **14**
5.1 Technical Implementation 16

6 References **18**

1 Introduction

This paper examines how the Implicit and the Crank-Nicolson Finite Difference Method is used to solve numerically for the Black-Scholes Merton partial differential equation (PDE) with the focus on an European Put. Generally speaking the Finite-Difference methods uses a Taylor series expansions in order to approximate the partial derivatives terms of the PDE (Brandimarte, 2006). The implicit scheme proceeds forward in time to solve the PDE, while the Crank-Nicolson Method can be described as a combination of the implicit and explicit scheme method.

Also both methods are extended with a Richardson extrapolation to increase accuracy. In the end of this paper it will be analysed which of the the four methods is the most efficient regarding accuracy and computational effort.

2 Implicit Finite Difference Method for an European Put

2.1 Theoretical Framework

The Black-Scholes-Merton-PDE is defined as:

$$\frac{1}{2}\eta^2 S^2 F_{SS}(S,T) + (r-q)SF_S(S,t) - rF(S,t) + F_t(S,T) = 0. \tag{1}$$

where η^2 is the volatility, S is the stock price, r is the risk-free rate, q is the continuous payed dividend and F(S,T) is the payoff function of the option depended of the stock price S and time to maturity T. Since we are interested in the values of an European Put option, we use the following payoff function (Hirsa, 2013):

$$F(S,T) = max(E - S(T), 0) \tag{2}$$

where S is the stock price at time-to-maturity T and E as the strike price of the option.

2.1.1 Defining a Grid

To solve the PDE (1) with the Finite Difference method, we have to set up a grid first by discretizing our input variables S and T as it was the case in the explicit scheme assignment. Hence, we define our input variables as

$$x = S = ih, \text{ where } i \in [0, M]$$

and

$$\tau = T - t = nk, \text{ where } n \in [0, N]$$

where i and n are index numbers, and h and k are step sizes with respect to the stock price and time. Hence one can define the rectangular grid as

$$x \in [0, x_{max}], \text{ with } x_{max} = M \cdot h$$

and

$$\tau \in [0, T], \text{ with } T = N \cdot k$$

Discretizing the PDE of the European Put at the points x and τ leads to the following equation:

$$a(i,n)u_{xx}(i,n) + b(i,n)u_x(i,n) + c(i,n)u(i,n) - u_\tau(i,n) = 0 \tag{3}$$

with

$$\begin{aligned} a(i,n) &= \frac{1}{2}\eta^2 i^2 h^2 \\ b(i,n) &= (r-q)ih \\ c(i,n) &= -r \end{aligned}$$

2.1.2 Difference Approximation of the Implicit Scheme

In order to approximate the discretized PDE (3) with the implicit scheme, we use following Taylor series expansions in order to approximate the partial derivatives (Fusai and Roncoroni, 2008).
The central approximation of the gamma term Γ:

$$[U_{xx}]_i^{n+1} = \frac{1}{h^2}(U_{i+1}^{n+1} - 2U_i^{n+1} + U_{i-1}^{n+1}) + \mathcal{O}(h^2) \tag{4}$$

The central approximation of the delta term Δ:

$$[U_x]_i^{n+1} = \frac{1}{2h}(U_{i+1}^{n+1} - U_{i-1}^{n+1}) + \mathcal{O}(h^2) \tag{5}$$

the forward approximation for the theta term Θ:

$$[U_\tau]_i^{n+1} = \frac{1}{k}(U_i^{n+1} - U_i^n n) + \mathcal{O}(k) \tag{6}$$

Plugging these approximations (4),(5), and (6) into our discretized PDE (3), we get following equation:

$$\frac{a}{h^2}(U_{i+1}^{n+1} - 2U_i^{n+1} + U_{i-1}^{n+1}) + \frac{b}{2h}(U_{i+1}^{n+1} - U_{i-1}^{n+1}) + cU_i^{n+1} - \frac{1}{k}(U_i^{n+1} - U_i^n n) = 0 \tag{7}$$

Rearranging (7) in order to solve for U_i^n leads to the implicit solution for the grid points at $x = ih$ and $\tau = nk$ for all $i \in [1, M-1]$ and $n \in [0, N-1]$:[1]

$$d_1(i,n+1)U_{i-1}^{n+1} + d_2(i,n+1)U_i^{n+1} + d_3(i,n+1)U_{i+1}^{+1n} = d_4(i,n+1) \tag{8}$$

with

$$\begin{aligned} d_1(i,n+1) &= -a\frac{k}{h^2} + b\frac{k}{2h} = -\frac{1}{2}\eta^2 i^2 k + \frac{1}{2}(r-q)ik \\ d_2(i,n+1) &= a\frac{2k}{h^2} - ck + 1 = \eta^2 i^2 k + (r-q)k + 1 \\ d_3(i,n+1) &= -a\frac{k}{h^2} - b\frac{k}{2h} = -\frac{1}{2}\eta^2 i^2 k - \frac{1}{2}(r-q)ik \\ d_4(i,n+1) &= U_i^n \end{aligned}$$

At this point it should be made clear that the discretized PDE can be described as recursive algorithm consisting of linear equations. Only the RHS is known while on the LHS we have M unknowns at each time step n. Hence methods like the Gaussian elimination can be used to solve for the system. This will be discussed in section 2.1.5 in more detail.

[1] These approximations are still independent of the time index n.

2.1.3 Truncation Error

In order to derive the truncation error of the fully implicit scheme, we need a closer look at the Taylor series expansion around $x = ih$ and $\tau = (n+1)k$ (Sturm, 2017):

$$u(x \pm h, \tau) = u(x, \tau) \pm hu_x + \frac{1}{2}h^2 u_{xx} \pm \frac{1}{6}h^3 u_{xxx} + \frac{1}{24}h^4 u_{xxxx} \pm \dots$$
$$u(x, \tau - k) = u(x, \tau) - ku_\tau + \frac{1}{2}k^2 u_{\tau\tau} - \dots$$

Plugging them into the rearranged discretized PDE in (7) leads to:

$$au_{xx} + \frac{a}{12}h^2 u_{xxxx} + bu_x + b\frac{h^2}{6}u_{xxx} + cu - u_\tau + \frac{1}{2}ku_{\tau\tau} + O(h^4, k^2) = 0 \ .$$

Since (3) holds we get following truncation error for the implicit scheme:

$$T^{Imp}(i,n) = h^2\left(\frac{a}{12}u_{xxxx} + \frac{b}{6}u_{xxx}\right) + k\left(\frac{1}{2}u_{\tau\tau}\right) + O(h^4, k^2)$$
$$T^{Imp}(i,n) = O(h^2, k) \ .$$

2.1.4 Boundary Conditions

One of the major aspects of the finite difference method are the boundary conditions. It should be made clear that in the implicit scheme boundaries have to be also established for the LHS and RHS of equation system (8).

The terminal condition is simply defined as the payoff function of the European Put because the value at maturity is known:

$$U_i^0 = max(E - ih, 0)$$

For the left-hand boundary we use a *Dirichlet* boundary condition, which in our case is described as the discounted exercise price $U_0^{n+1} = Ee^{-r(n+1)k}$.
For $i = 0$, we hence get:

$$d_2 U_1^{n+1} + d_3 U_2^{n+1} = d_4^* \text{ with } d_4^*(1, n+1) = d_4(1, n+1) - d_1(1, n+1)U_0^{n+1}.$$

For the right-side boundary, we will use a von Neumann boundary where we assume that the first derivative of the value at the grid point $(n+1, M)$ with respect to stock price is zero.

Economically speaking this means that if the stock price varies, the value of the option remains constant. Using the second central difference approximation (5) at the right-side border M with respect to our *von-Neumann* boundary condition leads to:

$$(U_x)_M^{n+1} \approx \frac{1}{2h}\left(U_{M+1}^{n+1} - U_{M-1}^{n+1}\right) \stackrel{!}{=} 0.$$

As already discussed in the explicit scheme framework, we can get rid of the 'ghost points' U_{M+1}^{n+1} because we assume that the boundary point next to our boundary point M+1 should also have approximately a slope of 0.
Hence we get:

$$d_1^* U_{M-1}^{n+1} + d_2^* U_M^{n+1} = d_4 \text{ with } d_1^*(M, n+1) = d_1(M, n+1) + d_3(M, n+1).$$

2.1.5 Matrix Notation and Solution of the System

In order to solve for the linear equation system of of the discretized PDE with respect to the boundaries, we use matrix notation in order to solve for the system.
Since we have M linear equations with M unknowns, we hence get:[2]

$$\underbrace{\begin{bmatrix} d_2(1) & d_3(1) & & & \\ d_1(2) & d_2(2) & d_3(2) & & \\ & & \ddots & & \\ & & d_1(M-1) & d_2(M-1) & d_3(M-1) \\ & & & d_1^*(M) & d_2(M) \end{bmatrix}}_{\mathbf{A}} \cdot \underbrace{\begin{bmatrix} U_1^{n+1} \\ U_2^{n+1} \\ \vdots \\ U_{M-1}^{n+1} \\ U_M^{n+1} \end{bmatrix}}_{U^{n+1}} = \underbrace{\begin{bmatrix} d_4(1) - d_1(1)U_0^{n+1} \\ d_4(2) \\ \vdots \\ d_4(M-1) \\ d_4(M) \end{bmatrix}}_{d_4}$$

As already discussed this system can be solved by using Gaussian elimination. However, as the $\mathbf{A}$ is of tridiagonal structure, it is more efficient to use a LR-decomposition to solve for the system (Brandimarte, 2006).
The system is described as:

$$\underbrace{\mathbf{A}}_{MxN} \underbrace{u}_{Mx1} = \underbrace{d}_{Mx1}$$

In the LR-Decomposition we first transform the A matrix into two matrices, where L is a lower triangular matrix and R is an upper triangular matrix:

$$(\mathbf{L}\ \mathbf{R})u = d$$

This can be rearranged to:

$$\mathbf{L}\underbrace{(\mathbf{R}u)}_{y} = d$$

Now the system can be solved with respect to:

$$\mathbf{L}y = d$$
$$\mathbf{R}u = d$$

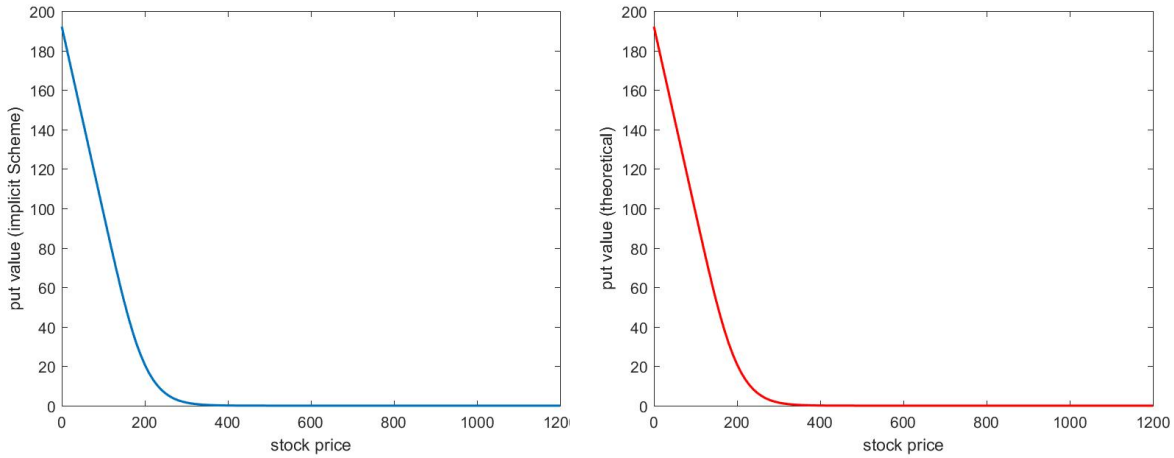
which is far more efficient because the number of operations with the LR-decomposition method only increases linearly compared to Gaussian elimination.

[2] d_1^* as in (2.1.4)

2.2 Discussion of Results

In our computations following parameters are used:

$$
\begin{aligned}
&S_{min} = 200 && M = 120\\
&S_{max} = 1200 && N = 100\\
&E = 200\\
&T - t = 1.00\\
&r = 0.04\\
&q = 0.06\\
&\eta = 0.25
\end{aligned}
$$



Implicit Scheme Theoretical

Figure 1: European Put value for different S - Implicit Scheme vs. Theoretical

The left plot of figure 1 shows the value of an European Put for different stock price levels with a time to maturity of 1 with respect to the other parameters and step-sizes calculated with the implicit scheme algorithm. The right plot depicts the theoretical computation (benchmark). As it can clearly be seen, the values only diverge minimally in those illustrations. For a stock price of 200 we get an explicit finite difference value of 20.7655 and for the closed-form solution we get a value of 20.8886.

In order to get a closer examination on the divergence of the implicit scheme method, the approximated values will directly be compared with the theoretical benchmark values. We define this divergence from now on as our error term:

$$Error = u_0(M, N) - BSMP$$

where u_0 and the BSMP are vectors which contain all calculated Put values solved with the explicit scheme or respectively with the closed form solution.

These errors are illustrated in figure 2 for different levels of S (from 0 to 1200) with a time to maturity of 1. As it can be seen the biggest deviation of 0.1231 can be found when the option is at-the-money. As it was already described in the explicit scheme assignment, at this

points it is not possible to take the derivatives due to discontinuity. Hence we get relatively poor approximations at these grid points.

As we have seen in the derivations in the theoretical part, the error vanishes with a speed of h^2 at each following approximation in space dimension.

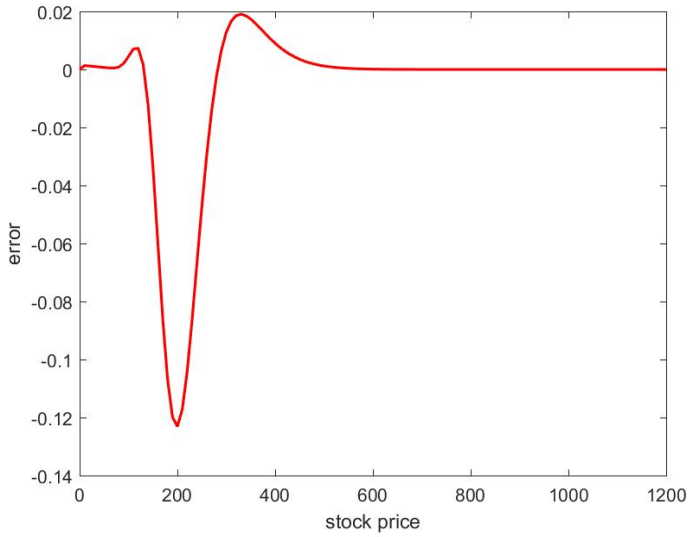


Figure 2: Implicit Scheme - Error vs. Stock Price

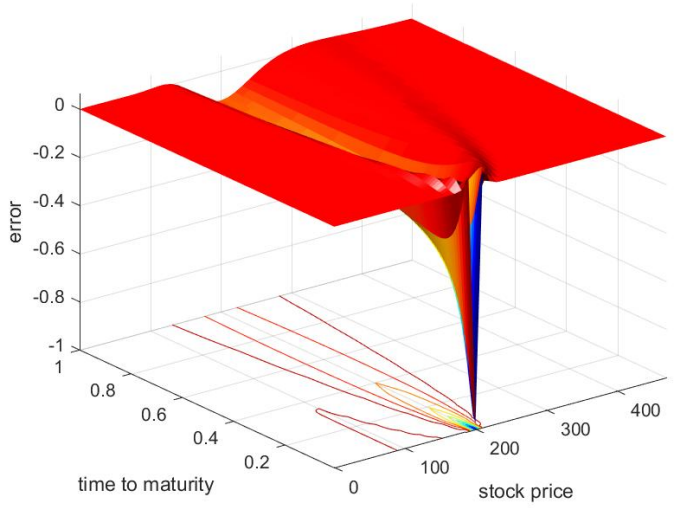


Figure 3: Implicit Scheme - Error vs. Stock Price vs. Time to Maturity

In figure 3 the 3D errors for increasing time to maturity are illustrated. Also here we can clearly see that the biggest deviations can be found when the option is at the money for the same reason as just discussed.

Also relatively big deviations can be observed when the option has a short time-to-maturity level of n. The upside probability for options with a very short time to maturity are mis-approximated due to the same reason as we discussed in the explicit scheme in the former assignment. The maximum absolute deviation which can be found in this analysis is 0.9740.

The error itself vanishes with the speed of the step-size k and takes longer to vanish com-

pared to the truncational error of the stock price approximations h^2.

2.3 Technical Implementation

The first part of the the main script $Task71\_73.m$ contains all necessary computations and plots to compute the price of an European Put for different stock prices. The script basically calls the function $Implicit.m$ which consists of the main algorithm for the implicit scheme.

To preserve the outline only the step-size parameters i and n serve as input variables. To change other input parameters one can adjust it in the algorithm function $Explicit\_EuroPut.m$ itself.

The algorithm basically follows all steps which I described in the theoretical framework of this chapter. The idea of the algorithm is to define all time independent variables out of the main for-loop to improve efficiency. First we discretize our framework with respect to *d1,d2 and d3* and implement the time independent boundary, as it was discussed in the theoretical part of this chapter.

In the next steps we start with the LR decomposition. Note here that we decided to use the self-written algorithm of the LR-decomposition instead of the implemented *Matlab* function *lu*. As we experienced the self-written algorithm is far more efficient than the *lu* function regarding computational time. As the A matrix is of triangular structure it is more efficient to adjust the LR-decomposition to this special case which can probably not be considered by the implemented *Matlab* function *lu*.

First we decompose the A-Matrix with a loop into a lower and an upper triangular matrix LR. After that we start with our main for-loop which counts for all time steps n in order to solve the equation system at each time step. First the time depended boundaries have to be implemented into the main-loop as it was described in the theoretical part. After that we first solve the system Ly = d with a further for-loop, before we finally continue to solve the system for Ru=y. These values are stored in a *u0* which is the final European Put price for the given S at the time to maturity level of 1. To compute the 3D-errors we subtract the theoretical Put price, which is again calculated with an self-written functions *BSMP.m and N.m* at this certain time step, and are stored in an own *ERROR*-matrix. After the main-loop we compute the 2D-error by subtracting the theoretical price from the last u0 vector. Furthermore we search for absolute maximum errors of the 2D error and 3D error cases.

Taking everything into account, we output the last *u0_Implicit* vector, the errors *ERROR_2D_Implicit* and the *ERROR_3D_Implicit*, as well as the maximum errors *ME_Implicit_2D* and *ME_Implicit_3D*, for the 2D and 3D cases, respectively.

In order to plot, the output additionally contains the discretized stock price vector S and the theoretical European Put vector *TheoPut*. All illustrations are simply plotted with the *plot* function with adjusted settings.

3 Crank-Nicolson Finite Difference Method for an European Put

This section analyses how the Crank-Nicolson Finite-Difference Method is applied to solve the BSM-PDE.[3] The Crank-Nicolson Method (CN) can be described as a combination of the implicit and explicit schemes. It proceed forward and backward in time and hence uses six points on the grid to approximate the price of the European Put option.

[3]The BSM-PDE equation described in chapter 2 still holds.

3.1 Theoretical Framework

3.1.1 Difference Approximation of the Crank-Nicolson Scheme

For the discretized PDE in the Crank-Nicolson framework we use following approximations of the partial derivatives (Fusai and Roncoroni, 2008):
The central approximation of the gamma term Γ:

$$[U_{xx}]_i^{n+0.5} = \frac{1}{2h^2}(U_{i+1}^{n+1} - 2U_i^{n+1} + U_{i-1}^{n+1} + U_{i+1}^n - 2U_i^n + U_{i-1}^n) + \mathcal{O}(h^2) + \mathcal{O}(\frac{k^2}{4}) \tag{9}$$

The central approximation of the delta term Δ:

$$[U_x]_i^{n+0.5} = \frac{1}{4h}(U_{i+1}^{n+1} - U_{i-1}^{n+1} + U_{i+1}^n - U_{i-1}^n) + \mathcal{O}(h^2) + \mathcal{O}(\frac{k^2}{4}) \tag{10}$$

the forward approximation for the theta term Θ:

$$[U_\tau]_i^{n+0.5} = \frac{1}{k}(U_i^{n+1} - U_i^n) + \mathcal{O}(\frac{k^2}{4}) \tag{11}$$

and the approximation of $[U]_i^{n+0.5}$:

$$[U]_i^{n+0.5} = \frac{1}{2}(U_i^{n+1} + U_i^n) + \mathcal{O}(\frac{k^2}{4}) \tag{12}$$

Plugging these approximations into the discretized PDE (3) we get:

$$\frac{a}{2h^2}\left(U_{i+1}^{n+1} - 2U_i^{n+1} + U_{i-1}^{n+1} + U_{i+1}^n - 2U_i^n + U_{i-1}^n\right) + \frac{b}{4h}\left(U_{i+1}^{n+1} - U_{i-1}^{n+1} + U_{i+1}^n - U_{i-1}^n\right)$$
$$+\frac{c}{2}\left(U_i^{n+1} + U_i^n\right) - \frac{1}{k}\left(U_i^{n+1} - U_i^n n\right) = 0\ .$$

Rearranging this equation with respect to U_i^n leads for $x = ih$ and $\tau = nk$ for all $i \in [1, M-1]$ and $n \in [0, N-1]$ to :

$$d_1(i, n+\frac{1}{2})U_{i-1}^{n+1} + d_2(i, n+\frac{1}{2})U_i^{n+1} + d_3(i, n+\frac{1}{2})U_{i+1}^{n+1} = d_4(i, n+\frac{1}{2}) \tag{13}$$

with

$$d_1(i, n+\frac{1}{2}) = -a\frac{k}{2h^2} + b\frac{k}{4h} = -\frac{1}{4}\eta^2 i^2 k + \frac{1}{4}(r-q)ik$$
$$d_2(i, n+\frac{1}{2}) = a\frac{k}{h^2} - c\frac{k}{2} + 1 = \frac{1}{2}\eta^2 i^2 k + \frac{1}{2}(r-q)k + 1$$
$$d_3(i, n+\frac{1}{2}) = -a\frac{k}{2h^2} - b\frac{k}{4h} = -\frac{1}{4}\eta^2 i^2 k - \frac{1}{4}(r-q)ik$$

$$d_4(i, n+\frac{1}{2}) = -d_1(i, n+0.5)U_{i-1}^n + (2 - d_2(i, n+0.5))U_i^n - d_3(i, n+0.5)U_{i+1}^n$$

3.1.2 Truncation Error

To analyse the truncation error of the CN scheme we again have to take a closer look of the corresponding Taylor series expansions (Sturm, 2017):

$$\begin{aligned}
u(x \pm h, \tau + \frac{k}{2}) &= u(x,\tau) \pm hu_x + \frac{1}{2}h^2u_{xx} \pm \frac{1}{6}h^3u_{xxx} + \frac{1}{24}h^4u_{xxxx} + \frac{k}{2}u_\tau + \frac{k^2}{8}u_{\tau\tau} + \frac{k^3}{48}u_{\tau\tau\tau} \\
&\quad \pm h\frac{k}{2}u_{x\tau} + \frac{1}{2}h^2\frac{k}{2}u_{xx\tau} \pm \frac{1}{2}h\frac{k^2}{4}u_{x\tau\tau} + \frac{1}{4}h^2\frac{k^2}{4}u_{xx\tau\tau} + \dots \\
u(x \pm h, \tau - \frac{k}{2}) &= u(x,\tau) \pm hu_x + \frac{1}{2}h^2u_{xx} \pm \frac{1}{6}h^3u_{xxx} + \frac{1}{24}h^4u_{xxxx} - \frac{k}{2}u_\tau + \frac{k^2}{8}u_{\tau\tau} - \frac{k^3}{48}u_{\tau\tau\tau} \\
&\quad \mp h\frac{k}{2}u_{x\tau} - \frac{1}{2}h^2\frac{k}{2}u_{xx\tau} \pm \frac{1}{2}h\frac{k^2}{4}u_{x\tau\tau} + \frac{1}{4}h^2\frac{k^2}{4}u_{xx\tau\tau} + \dots \\
u(x, \tau + \frac{k}{2}) &= u(x,\tau) \pm \frac{k}{2}u_\tau + + \frac{k^2}{8}u_{\tau\tau} \pm \frac{k^3}{48}u_{\tau\tau\tau} + \dots
\end{aligned}$$

Plugging them into the rearranged discretized PDE yields to:

$$\begin{aligned}
&\frac{a}{2h^2}\left(2h^2u_{xx} + \frac{1}{6}h^4u_{xxxx} + h^2\frac{k^2}{4}u_{xx\tau\tau}\right) + \frac{b}{4h}\left(4hu_x + \frac{2}{3}h^3u_{xxx} + 2h\frac{k^2}{4}u_{x\tau\tau}\right) \\
&+\frac{c}{2}\left(2u + \frac{k^2}{4}u_{\tau\tau}\right) - \frac{1}{k}\left(ku_\tau + \frac{1}{3}\frac{k^3}{8}u_{\tau\tau\tau}\right) = 0
\end{aligned}$$

which can be simplified to

$$\underbrace{au_xx + bu_x + cu - u_\tau}_{=0} + h^2\left(\frac{a}{12}u_{xxxx} + \frac{b}{6}u_{xxx}\right) + k^2\left(\frac{a}{8}u_{xx\tau\tau} + \frac{b}{8}u_{x\tau\tau} + \frac{c}{8}u_{\tau\tau} - \frac{1}{24}u_{\tau\tau\tau}\right) = 0$$

$$\implies T^{CN}\left(i, n+\frac{1}{2}\right) = O(h^2, k^2)\ .$$

As you can see, The CN method is more accurate because the error vanishes faster compared to the implicit scheme of chapter 2. This will be discussed in the further chapters in more detail.

3.1.3 Boundary Conditions

The terminal condition is again simply defined as the payoff function of the European Put:

$$U_i^0 = max(E - ih, 0)$$

For the left-hand boundary we also use a *Dirichlet* boundary condition, which is described as the discounted exercise price $U_0^{n+1} = Ee^{-r(n+1)k}$.
For $i = 0$, we hence get:

$$d_2U_1^{n+1} + d_3U_2^{n+1} = d_4^*$$

with

$$d_4^*(1, n+\frac{1}{2}) = d_4(1, n+\frac{1}{2}) - d_1(1, n+\frac{1}{2})U_0^{n+1}.$$

For the right boundary of the grid at i=M, we again use a *von-Neumann* boundary where we assume that the first derivative of the value at the grid point $(n+1, M)$ with respect to stock price is zero.

Hence:

$$\frac{1}{2h}\left(U_{M+1}^{n} - U_{M-1}^{n}\right) \stackrel{!}{=} 0$$
$$\frac{1}{2h}\left(U_{M+1}^{n+1} - U_{M-1}^{n+1}\right) \stackrel{!}{=} 0$$

Plugging this into the discretized PDE this leads to:

$$d_1 U_{M-1}^{n+1} + d_2 U_M^{n+1} + d_3 U_{M+1}^{n+1} = -d_1 U_{M-1}^{n} + (2 - d_2) U_M^{n} - d_3 U_{M+1}^{n}$$

Eliminating the 'ghost points' at U_{M+1}^{n} and U_{M+1}^{n+1} leads to:

$$d_1^* U_{M-1}^{n+1} + d_2 U_M^{n+1} = d_4^*$$

with

$$d_1^*(M, n+\frac{1}{2}) = d_1(M, n+\frac{1}{2}) + d_3(M, n+\frac{1}{2})$$
$$d_4^*(M, n+\frac{1}{2}) = -d_1^* U_{M-1}^{n} + (2 - d_2) U_M^{n}$$

3.1.4 Matrix Notation and Solution of the System

As in the implicit scheme, we rewrite the grid into matrix notation in order to solve the system with a LR decomposition. The matrix notation of the CN scheme with respect to the boundaries is defined as:

$$\mathbf{A}U^{n+1} = d_4 = \mathbf{B}U^{n} + f$$

$$\underbrace{\begin{bmatrix} -d_2(1) & d_3(1) & & & & \\ d_1(2) & -d_2(2) & d_3(2) & & & \\ & & \ddots & & & \\ & & d_1(M-1) & -d_2(M-1) & d_3(M-1) & \\ & & & d_1^*(M) & -d_2(M) \end{bmatrix}}_{\mathbf{A}} \cdot \underbrace{\begin{bmatrix} U_1^{n+1} \\ U_2^{n+1} \\ \vdots \\ U_{M-1}^{n+1} \\ U_M^{n+1} \end{bmatrix}}_{U^{n+1}} = \underbrace{\begin{bmatrix} d_4(1) \\ d_4(2) \\ \vdots \\ d_4(M-1) \\ d_4(M) \end{bmatrix}}_{d_4}$$

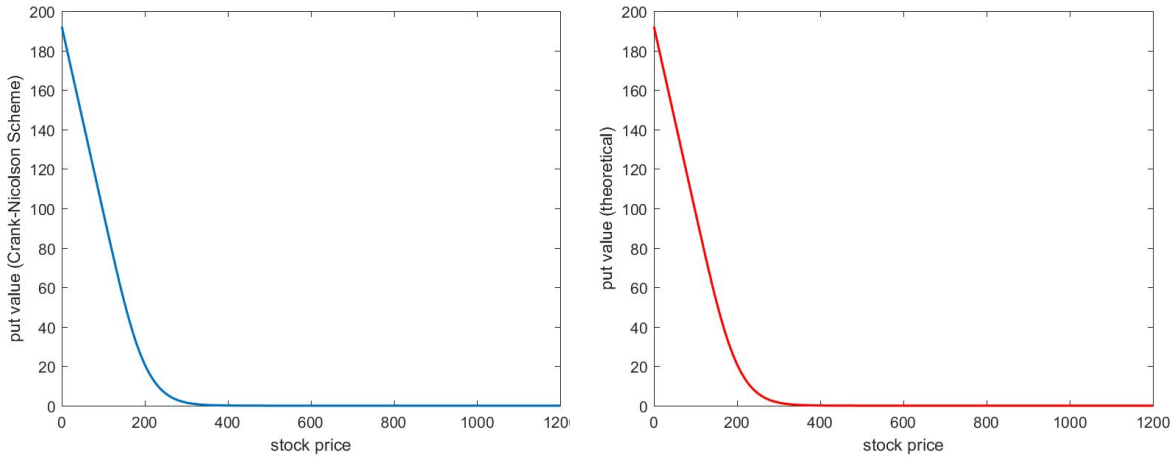
where

$$\underbrace{\begin{bmatrix} d_4(1) \\ d_4(2) \\ \vdots \\ d_4(M-1) \\ d_4(M) \end{bmatrix}}_{\mathbf{d_4}} = \underbrace{\begin{bmatrix} 2-d_2 & d_3 & & & \\ d_1 & 2-d_2 & d_3 & & \\ & & \ddots & & \\ & & d_1 & 2-d_2 & d_3 \\ & & & d_1^* & 2-d_2 \end{bmatrix}}_{\mathbf{B}} \cdot \underbrace{\begin{bmatrix} U_1^{n} \\ U_2^{n} \\ \vdots \\ U_{M-1}^{n} \\ U_M^{n} \end{bmatrix}}_{U^{n}} + \underbrace{\begin{bmatrix} -d_1(1)U_0^{n+1} - d_1(1)U_0^{n} \\ 0 \\ \vdots \\ \vdots \\ 0 \end{bmatrix}}_{f}$$

The system can again be solved with a LR decomposition in the following way:

$$\mathbf{A}u = (\mathbf{L}\ \mathbf{R})u = \mathbf{L}\underbrace{(\mathbf{R}u)}_{y} = \mathbf{L}y = d$$

where d equals d_4.



Crank-Nicolson Scheme Theoretical

Figure 4: European Put value for different S: Crank-Nicolson Scheme vs. Theoretical

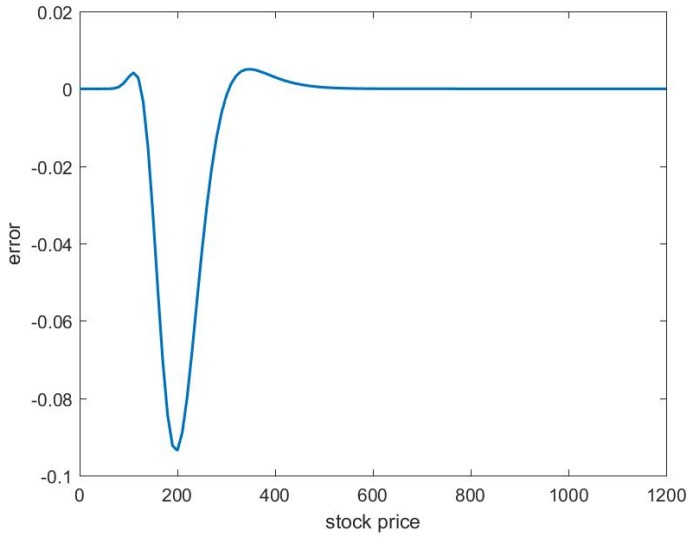


Figure 5: Crank-Nicolson Scheme - Error vs. Stock Price

3.2 Discussion of Results

Again it can clearly be seen that the CN scheme prices are almost identical to the theoretical price. For an underlying price of 200 the price of the European Put is 20.7951 (benchmark price = 20.8886). These errors in figure 5 are similar to our former findings. The biggest deviation can be found when the option is at-the-money due to the same reasons. In this case the error is 0.0936.

Also the deviations decrease with increasing time steps as it can be seen in the 3D plot in 6. The maximum deviation which can be observed is 0.8762.

Comparing these results with the implicit scheme, one can see that the CN scheme computes more precise results.[4] This results are also in line with the theory because the truncation error of the CN scheme decreases,as it was discussed in 3.2.2., with a speed k^2, instead of only k, as it was the case of the implicit scheme. This will be analysed in more detail in the last chapter of this paper.

[4]Implicit Scheme 2D-Error: 0.1231 and 3D-Error: 0.9740.

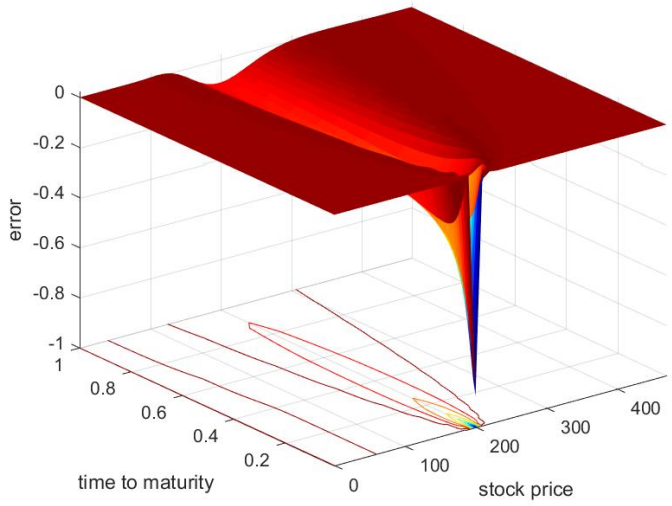


Figure 6: Crank-Nicolson Scheme - Error vs. Stock Price vs. Time to Maturity

3.3 Technical Implementation

The second part of the the main script $Task71\_73.m$ contains all necessary computations and plots for the Crank-Nicolson scheme. The script basically calls the function $CN.m$ which consists of the main algorithm for the Crank-Nicolson scheme.

The technical implementation is similar to the implicit scheme in the former chapter 2. The LR decomposition in this algorithm has to consider the B-Matrix on the right hand side. Furthermore other boundaries have to be implemented.

As an output we get again the last *u0_CN* vector, as well as the errors for *ERROR_2D_CN* for the 2D case and the *ERROR_3D_CN* for the 3D case, as well as the maximum errors *ME_CN_2D* and *ME_CN_3D* for the 2D and 3D case, respectively.

4 Richardson extrapolation method

To improve accuracy we introduce an Richardson extrapolation method for the same fixed combination of M and N to the implicit and Crank-Nicholson scheme.
For the implicit scheme we use:

$$\frac{1}{3}(4U_{4M}^{2N} - U_M^N) \tag{14}$$

For the Crank-Nicolson-Scheme we use:

$$\frac{1}{3}(4U_{2M}^{2N} - U_M^N) \tag{15}$$

Following the same analysis as in the chapters, one can derive the truncation errors of the extrapolated functions. (Sturm, 2017):
For the extrapolated implicit scheme we get:

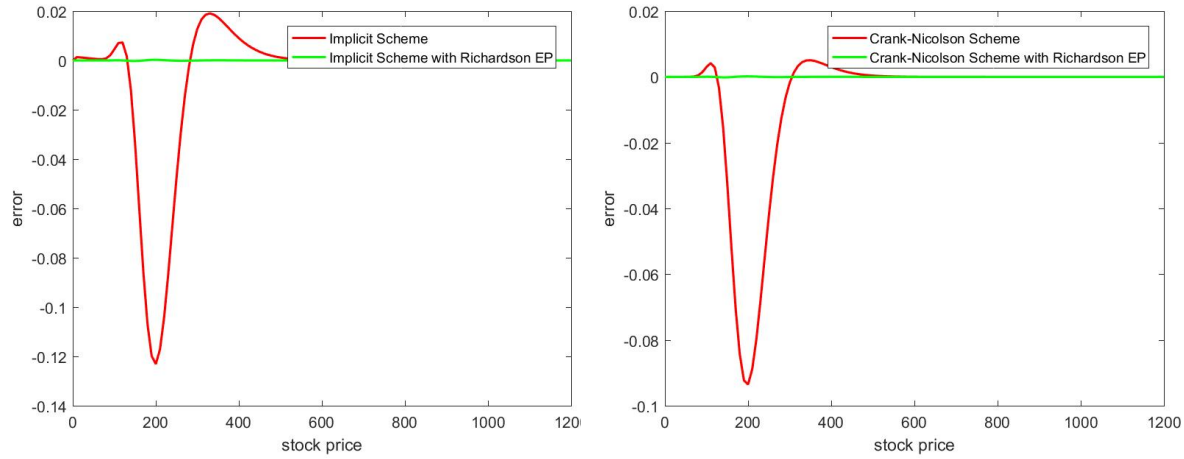
$$O(h^4 + k^2) \tag{16}$$

and for the extrapolated Crank-Nicolson scheme:

$$O(h^4 + k^2h^2 + k^4) \tag{17}$$

One can show that, the truncation error of the Crank-Nicolson scheme is of higher order and hence will deliver more precise results.

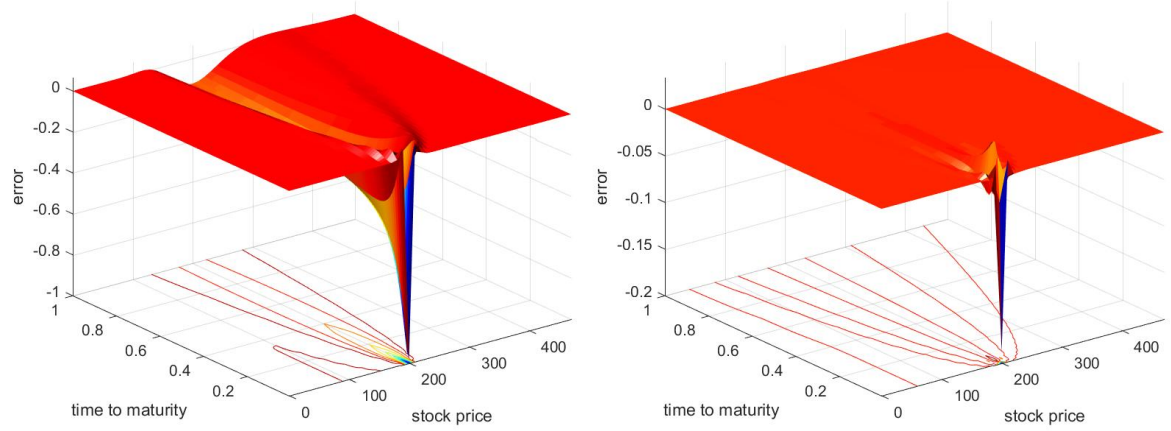
4.1 Discussion of Results



Implicit Scheme CN Scheme

Figure 7: 2D-Errors - Standard Schemes vs. Extrapolation

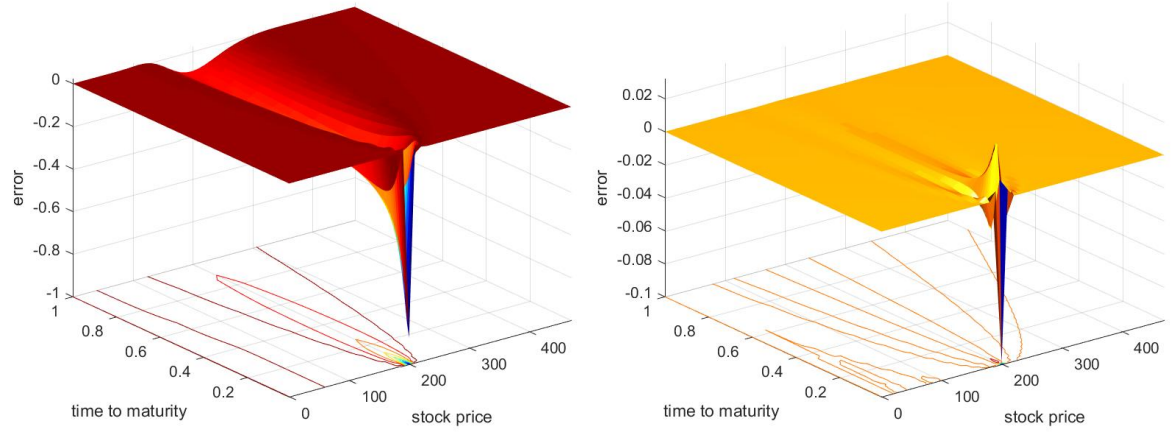
Figure 7 compares the standard implicit and Crank-Nicolson scheme methods with the extrapolated functions. The prices for the European Put at S=200 for the extended implicit and CN schemes are 20.8889 and 20.8888, respectively. As you can clearly see the methods decrease the errors significantly. The maximum errors of the extrapolated implicit and Crank-Nicolson cases are 2.6765e-04 (vs. 0.1231) and 1.7175e-04 (vs. 0.0936) respectively, which is a huge improvement compared to the calculation without extrapolation.



Standard Implicit Scheme Implicit Scheme with Richardson Extrapolation

Figure 8: Standard Implicit Scheme vs. Extrapolation (3D-Errors)

Same holds for the 3D-error shown in figures 8 and 9. Both extrapolation methods improves accuracy also for the 3D case as you can also observe in the maximum absolute error. In the 3D case the maximum absolute error of the extrapolated implicit and Crank-Nicolson



Standard Implicit Scheme | Implicit Scheme with Richardson Extrapolation

Figure 9: Standard Crank-Nicolson Scheme vs. Extrapolation (3D-Errors)

schemes are 0.1881 (vs. 0.9740) and 0.0992 (vs. 0.8762).

The reason of this improvement can be explained by the truncation errors in (16) and (17). The errors converge faster compared to the standard methods and hence deliver more precise results.

Note that higher accuracy comes with a higher computational effort. As you can see in equations (14) and (15) bigger grids are necessary which can increase computational time significantly. This can be described as a trade off between accuracy and time, which will be discussed more in detail in chapter 5.

4.2 Technical Implementation

The third part of the the main script $Task71\_73.m$ contains all necessary computations and plots for the Richardson Extrapolation functions. The script calls the function $Implicit\_RE.m$ and $CN\_RE.m$. Each function basically calls the scheme functions *Implicit.m* or *CN.m* with respect to the grid-sizes defined by the Richardson extrapolation in Task 7.3. in the assignment. Since the Richardson extrapolation creates vectors of different sizes MxN, we have to cut the vector to the specific form (every 2nd row). The computations of the extrapolation are then just stored in an own *u0 vector*.

To get the 3D-errors of the extrapolation, one has also to adjust the error matrices in order to fit. For the implicit scheme (4Nx2M) we only take every 4th column into account and cut every 2nd row away. For the CN scheme (2Mx2N) we only take every 2nd column into account and cut away every 2nd row.

As an output, we again get the *u0* vectors, swell as the error vector (2D errors), matrix (3D errors) and its maximum absolute errors, for the implicit scheme and Crank-Nicolson scheme respectively.

5 Efficiency Discussion: Accuracy and Computational Effort

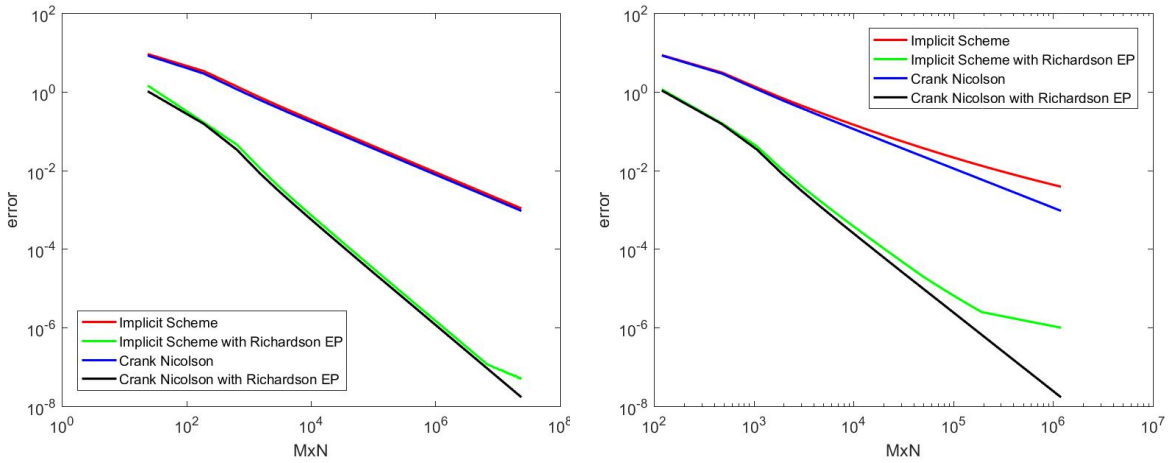
In this section we compare which method is the most efficient by computing the price of the European Put with the given parameters for increasing combinations of M and N. Therefore we fix the time dimension on the space dimension for M $\in [12 : 12 : 1200]$ in the following way.

For the first case:

$$N = \frac{1}{72}M^2$$

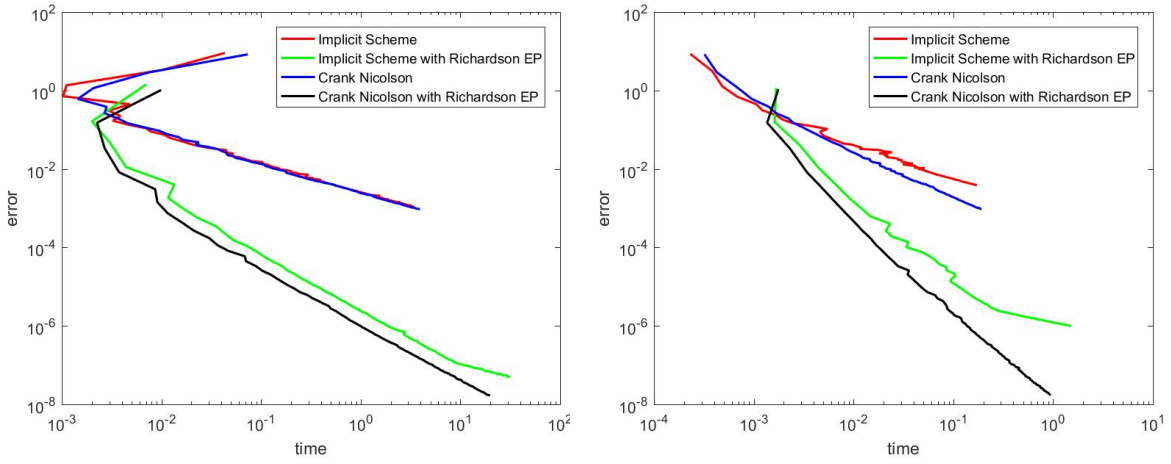
For the second case:

$$N = \frac{1}{1.2}M$$



$N = \frac{1}{72}M^2$ $N = \frac{1}{1.2}M$

Figure 10: Errors Development for increasing step-sizes MxN (log-log-scale))



$N = \frac{1}{72}M^2$ $N = \frac{1}{1.2}M$

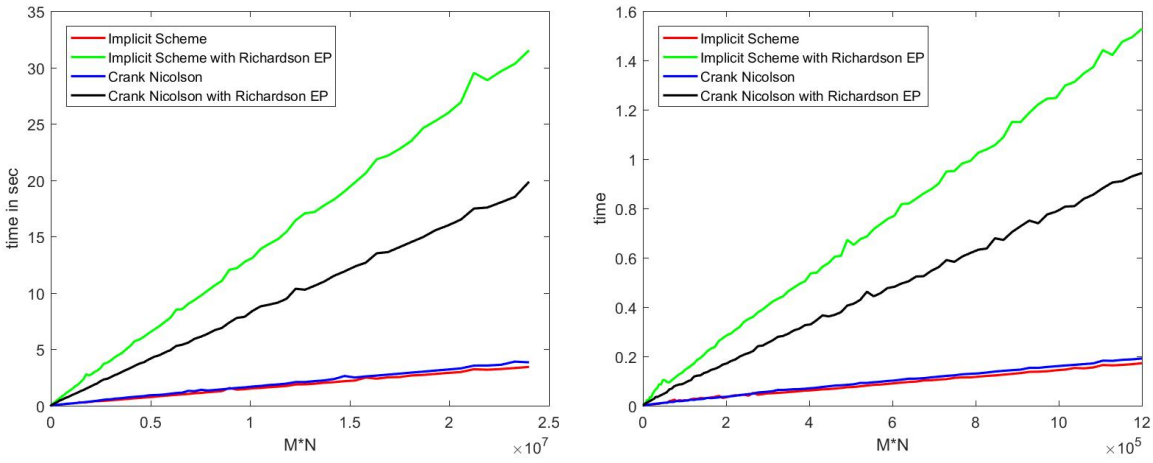
Figure 11: Trade-Off: Computational Time vs. Error

In figure 10 one can see that the error decreases with increasing MxN combinations and the Crank-Nicolson scheme delivers, due to the smaller truncation error, a more precise approximation then the implicit scheme.

Also one can easily observe that the Richardson extrapolation of the schemes clearly improves accuracy. It is also noticeable that in the extrapolated case, the CN function dominates the implicit function, even tough it uses fewer grid points to extrapolate (CN uses 2Nx2M

while the implicit scheme uses 2Nx4M grid points). Again, this is in line with the theory. As we have shown the truncation errors of the extrapolated functions are of higher order compared to the case without extrapolation. Also the truncation errors of the CN scheme in the extrapolated case are of higher order compared to the implicit case.

Plot 11 illustrates the trade off between computational time and accuracy.[5] One can see that for a decreasing error size the extrapolated functions are more efficient. Hence, for our programmed algorithm, the extrapolated CN function, delivers the most efficient results for relatively big grid sizes MxN. But this, in fact, also implies that efficiency is depended on the size of grid. For small grids it may makes sense to dispense with extrapolation as you can see in figure 12.



$$N = \frac{1}{72}M^2 \qquad N = \frac{1}{1.2}M$$

Figure 12: Errors Development for increasing step-sizes MxN (log-log-scale))

Comparing the left and right figures of illustrations 10, 11 and 12 makes clear that an increase in the grid-size does not necessary increase accuracy, especially when it comes to efficiency regarding computational effort. In particular, this can be seen in figure 12. A strong increase in the grid size may come with an increase in precision but again with an large increase in computational time and inefficient computations.

Taking everything together, when it comes to the trade off between accuracy and computational effort, the Crank-Nicolson scheme delivers not only the most accurate but also the most efficient results with respect to computational time. Hence the Crank-Nicolson scheme should always be preferred to the implicit method, especially for modified Richardson extrapolation functions.

5.1 Technical Implementation

Task 7.4 and 7.5 can be solved by running the scripts $Task74.m$ and $Task75.m$. In these scripts we constructed for each of the four methods a for-loop, where the function is called with respect to the computational 2D-error, for each specific MxN combination. Since we are interested in the computational time we additionally count the time by including the command 'tic' and 'toc' into each for-loop. Note that due to system performance, the time measurement might differ slightly from run to run. The output vectors are later plotted with a *loglog*-command to get a log-scale on the x and y axis.

Note that all computations were run on an Windows 10 64-bit operation system with

[5]Note that time plots might differ from run to run of the function.

an Intel(R) Core(TM) i7-5500U CPU 2.40 Ghz. Since the computation might take several minutes, the data of task 7.4. and 7.5. in *74_Errors.mat* and *75_Errors.mat* were additionally added.

6 References

Brandimarte, Paolo. *Numerical Methods in Finance and Economics - A MATLAB-Based Introduction*, 2^{nd} ed., John Wiley & Sons, Hoboken NJ, 2006.

Fusai, Gianluca and Roncoroni, Andrea. *Implementing Models in Quantitative Finance: Methods and Cases.* Springer-Verlag, Berlin Heidelberg New York 2008.

Hirsa, Ali. *Computational Methods in Finance.* Chapman & Hall/ CRC Press, Boca Raton FL, 2013.

Sturm, Pascal. *Assignment #2: Implicit Finite Difference Methods.* Numerical Methods in Finance Assignments, Tübingen, 2017.