Computational Finance with Saxo

Karl Engelund/QVT701
Michael Dalby/LMD389

January 16, 2023

0. Parameters and notation

We use the following parameters for our finite difference solver unless otherwise stated

$$T = 5, K = 1.025, S(0) = 1, r = 0.04, \mu = -0.03, \sigma = 0.2, n = 100, \theta = 0.5.$$

In particular $n = 100$ means a grid of 101 points for S. Furthermore we use

$$\texttt{numStd} = 5, m = 6400.$$

For the grid points of S we do the following transformation

$$S_i = S(0)e^{(i-\lfloor (n+1)/2 \rfloor)\Delta x)}$$

for $i = 0, \dots, n$ where $\Delta x = 2 \cdot \texttt{numStd} \cdot \sigma\sqrt{T}n^{-1}$. Finally we will in general assume constant BS parameters μ and σ across time and space.

1. Convergence of Vanilla

We start by looking at the rate of convergence for a European call option as function of Δt, for each of the schemes. We run the backwards solver across the number of time steps m recorded in Table 1. Note first that the implicit scheme and Cranck-Nicolson is always von Neumann stable, however for the explicit scheme stability is not ensured. We observe nonsensical prices for $m \leq 75$ hence these are note used for the analysis.

We assume that the error on each scheme is

$$\left|\hat{\sigma^{iv}}_m - \sigma\right| = \beta \Delta t^\alpha \tag{1}$$

Taking logarithm that is

$$\log\left|\hat{\sigma^{iv}}_m - \sigma\right| = \log\beta + \alpha\log\Delta t. \tag{2}$$

Hence we can estimate the order of convergence $\mathcal{O}(\Delta t^\alpha)$ by a linear regression on log-log scale. Instead of using the true $\sigma = 0.2$ we account for machine precision by calculating the implied volatility for each of the schemes with $m = 4 \cdot 10^6$. This yields

$$\sigma_0^{(\text{mp})} = 0.200004152, \qquad \sigma_{0.5}^{(\text{mp})} = 0.200004144, \qquad \sigma_1^{(\text{mp})} = 0.200004137$$

Doing a linear regression with intercept by $(X^\top X)^{-1}X^\top Y$ we get

$$\hat{\alpha}_0 = 1.03010175, \qquad \hat{\alpha}_{0.5} = 1.93501041, \qquad \hat{\alpha}_1 = 0.999403937$$

$m \setminus \theta$	0	0.5	1
10		0.200025340	0.197136166
25		0.200020781	0.198851659
50		0.200008305	0.199427066
75		0.200005994	0.199619243
100	0.200376137	0.200005185	0.199715401
150	0.200196863	0.200004607	0.199811604
200	0.200148667	0.200004404	0.199859722
400	0.200076393	0.200004209	0.199931921
800	0.200040266	0.200004161	0.199968029
1600	0.200022204	0.200004148	0.199986086
3200	0.200013174	0.200004145	0.199995115
6400	0.200008659	0.200004145	0.199999630

Table 1: Calculated IV across different number of time steps

We have plotted the errors as function on Δt on $\log - \log$ scale in figure 1. We note that the explicit and implicit errors are almost identical. Our estimates of $\alpha \approx 1$ for the implicit and explicit schemes while $\alpha \approx 2$ for the Crank-Nicolson is in line with the theoretical convergence since we are essentially doing a Taylor approximation, where for the implicit and explicit schemes we take the first order approximation and while we make a second order approximation in the Crank-Nicolson.

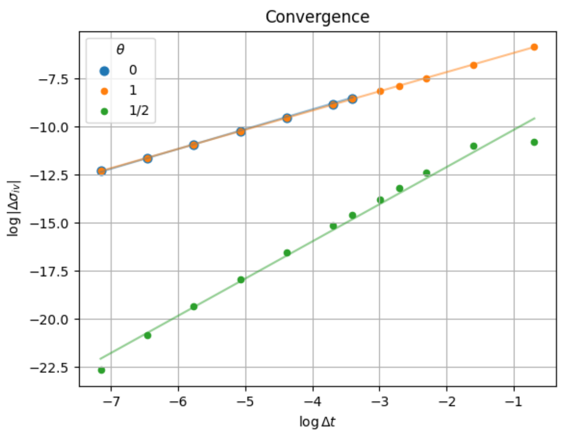


Figure 1: Convergence of finite difference schemes

2. Exercise premium

When pricing american options we would like to study the early exercise premium, being $E = C^E - C^A$. In our finite difference solver we can easily price Bermudan options, f.x. C^B, by evaluating whether the calculated option price or the current exercise value is optimal at each time step. That is, after rolling the finite difference to get call prices $C(t_{h+}, S_i)$ we set the option value to

$$C(t_h, S_i) = \max\left(C(t_{h+}, S_i), (S_i - K)^+\right) \tag{3}$$

After this operation we roll again to $t_{(h-1)+}$ and evaluate the call price at t_{h-1}. In our implementation we do not allow for exercise when the calculated price is negative.

So if we want to study the convergence of our finite difference solver to E, we need convergence of both the Bermudan to the American as well as the convergence of the Crank-Nicolson scheme. Thus we expect the convergence to be slower then in section 1.

We can calculate the early exercise premium on the same m grid as in section 1. We get the values in table 2. To get an estimate of the order of convergence we again assume the error is of order $\mathcal{O}(\Delta t^\alpha)$ and perform a linear regression on the log-log space where we compare with $E^{(mp)} = 0.0235229828$. The estimated convergence parameter is $\hat{\alpha} = 0.9779399075$. We have plotted the errors as function on Δt on $\log - \log$ scale in figure 2. We note that the assumption of linearity on $\log - \log$ space is very well backed by the data. Since $\alpha \approx 1$ we see that the convergence of early

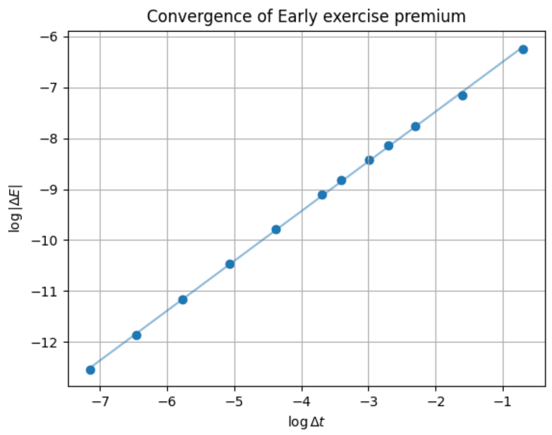


Figure 2: Convergence of early exercise premium, $\theta = 0.5$

exersice premium in the Crank-Nicolson scheme is as fast as the convergence

$m \setminus \theta$	0.5
10	0.0215733212
25	0.0227374087
50	0.0230979301
75	0.0232328172
100	0.0233012959
150	0.0233740507
200	0.0234111644
400	0.0234666393
800	0.0234947399
1600	0.0235088547
3200	0.0235159160
6400	0.0235194506

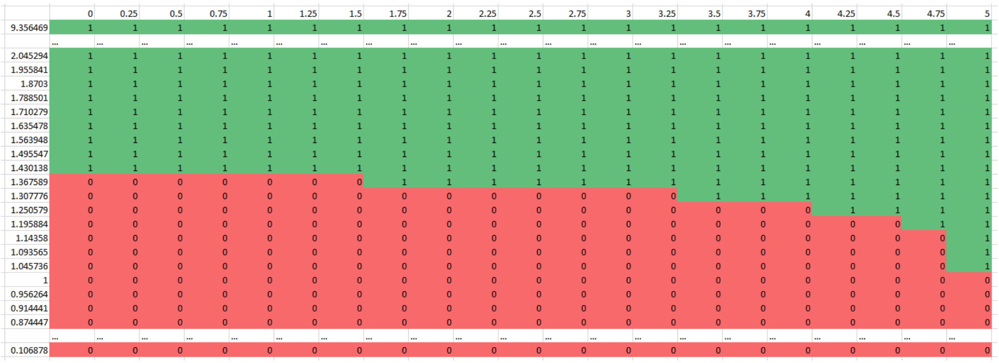
Table 2: Calculated early exercise premium across different number of time steps

of a vanilla in the explicit and implicit schemes.

We could also be interested in looking at when exercising is optimal on the grid. This can easily be implemented in the finite difference solver by recording which value is the maximal in equation (3). This can be stored in a matrix $\mathbf{E} \in \mathbb{R}^{(n+1)\times m}$ with

$$\mathbf{E}_{i,h} = 1_{[C(t_h,S_i)=(S_i-K)^+]}.$$

For presentation purposes we add a parameter `timeFactor` to the finite difference solver that only presents every `timeFactor`'th time point on the grid. Furthermore we make the output as a grid an option in the `ae` parameter. The **E** matrix is shown in figure 3 (Mirrored on space axis with `timeFactor` $= 2$ and $m = 100$). Another way to calculate the exercise boundary is to update a vector with the earliest time that the finite difference solver deems it optimal to exercise. For this to be accurate we need the exercise boundary to be decreasing, this is well known in the zero dividend case with constant parameters. The above example indicates that this also seems to be the case when having continuous dividends. Implementing this as another option in the `ae` parameter we can find the (first time) exercise boundary, which is plotted in figure 4 where we use $m = 6400$ again.



	0	0.25	0.5	0.75	1	1.25	1.5	1.75	2	2.25	2.5	2.75	3	3.25	3.5	3.75	4	4.25	4.5	4.75	5
9.356469	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2.045294	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1.955841	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1.8703	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1.788501	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1.710279	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1.635478	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1.563948	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1.495547	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1.430138	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1.367589	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1.307776	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
1.250579	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
1.195884	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
1.14358	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1.093565	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1.045736	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.956264	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.914441	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.874447	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
0.106878	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 3: Exercise indicator on the grid

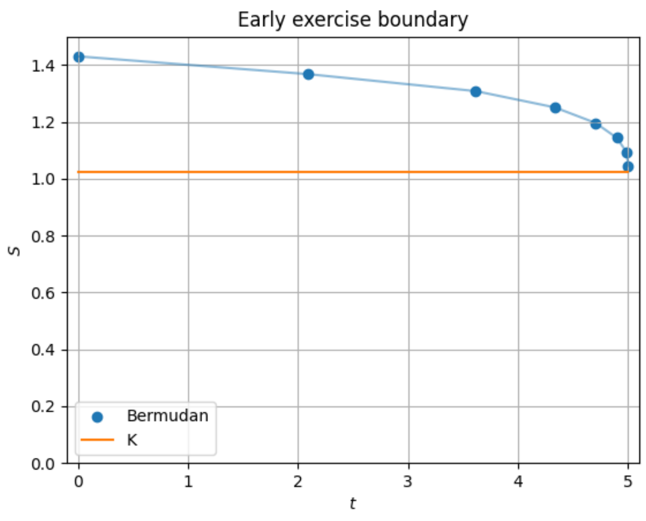


Figure 4: First time exercise boundary

3. The effects of smoothing

When running the backward finite difference solver we calculate the boundary conditions on the grid points. For a digital option with strike K the terminal condition will depend on where the grid points lay in respect to K. In figure 5 we have plotted the real payoff and the payoff used as boundary condition when the strike is between the grid points. As an alternative we can smooth out the payoff by taking the terminal condition to be

$$P(t_H, S(i), K) = \begin{cases} 0, & S(i+1) \leq K \\ 1 & S(i) \geq K \\ \frac{S(i+1)-K}{S(i+1)-S(i)}, & S(i) < K < S(i+1) \end{cases}$$

that is when the strike is between the grid points we calculate $\frac{1}{\Delta S_i}\int_{S(i)}^{S(i+1)} 1_{(s>K)} ds$. Note that for varying K between two grid points will yield exactly the same

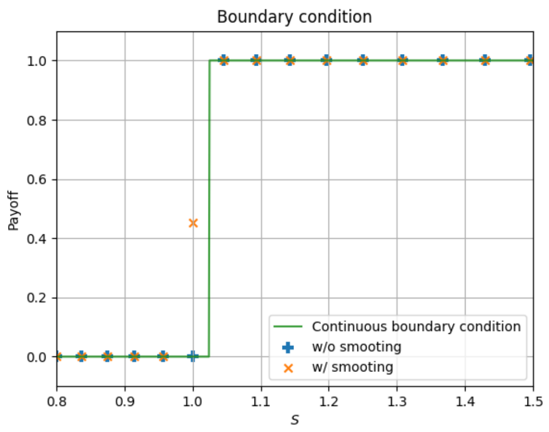


Figure 5: Boundary condition for digital option with K between S points on grid

boundary conditions when smoothing is not used. Although the error on the grid points of the terminal condition is larger using smoothing we will study the effect on the price.

Note that theoretically we have

$$P(0,K) = \mathbb{E}\left[e^{-rT}1_{(S_T>K)}\right] = e^{-rT}\mathbb{P}\left(S_T > K\right)$$

and since we have smooth densities the price should be smooth as a function of K. We run the backward solver using the two methods while varying the strike. The output is plotted in figure 6. As reasoned above we see that without using smoothing the price is step-wise constant, while smoothing yields a smooth price.

4. Width of grid

When choosing the grid to evaluate on we want to have enough points such that the price converges. However having a too large S grid means the computer will have to do a lot of calculations, even though the probability of S ever reaching these extreme values is virtually zero. It's therefore just computationally heavier with negligible benefits. In this section we will study how wide a grid is needed for convergence.

To do this we calculate the implied volatility of a European call option, where we vary `numStd` and n while keeping $\Delta \log S = \Delta x$ fixed. With our

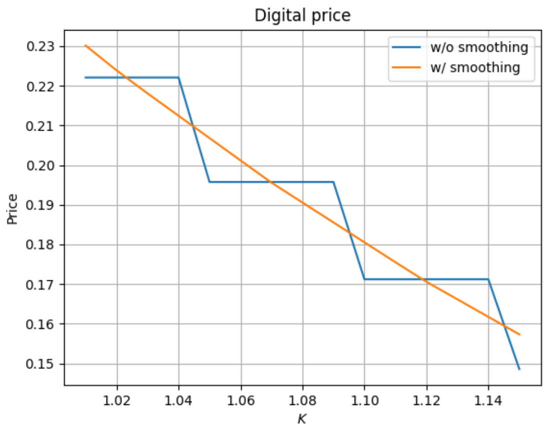


Figure 6: Price of digital option as function of strike

standard parameters we have $\overline{\Delta x} = \frac{5}{100} 2\sigma\sqrt{T}$ so we use $n = \texttt{numStd} \cdot \frac{2\sigma\sqrt{T}}{\Delta x} = 20 \cdot \texttt{numStd}$. The volatilities are shown in table 3 We see that having `numStd`

`numStd`	n	σ_{iv}
1	20	0.1886787083521000
2	40	0.1999902717141600
3	60	0.2000041439121220
4	80	0.2000041445497550
5	100	0.2000041445497600
6	120	0.2000041445497560
7	140	0.2000041445497560
8	160	0.2000041445497560
9	180	0.2000041445497560
10	200	0.2000041445497600

Table 3: Calculated IV across different number of time steps

less then 3 is not sufficient for pricing. Having it greater then or equal to 4 will yield pretty much perfect results. So for time considerations it should be optimal to take `numStd`$\in \{4, 5\}$. To explain why taking high `numStd` yields no difference in the result recall that

$$\mathbb{P}(S_t \geq S_u) = \mathbb{P}\left(W_1 \geq \frac{\log \frac{S_u}{S_0} - (\mu - \frac{\sigma^2}{2})t}{\sigma\sqrt{T}}\right) \approx 10^{-11}$$

at expiry for $S_u = S_0 \exp\{\texttt{numStd} \cdot \sigma\sqrt{T}\}$ with `numStd`=6.

5. Forward equations

We implemented the kBlack::FwdRunner() by solving the following equation for each time step:

$$p(t_{h+1}) = [I + (1-\theta)\Delta t \bar{A}'][I - \theta \Delta t \bar{A}']^{-1} p(t_h)$$

With initial boundary condition at $t_0 = 0$ of $p(t_0) = 1$ in $S_0 = 1$ and 0 everywhere else. We had S_0 as one of our grid points so this was no problem, but if S_0 lied between points we could have made these two points both hold positive probabilities as the boundary. Most simply we could have set the probabilities equal to the relative distance from said point.

After doing the forward roll we were left with a matrix or more specifically a vector of vectors of the probabilities (with discounting) for all our discrete time points and underlying asset points, where the vector for each time point summed to 1 times the discount factor. These discounted probabilities were under the Q measure so we could find call option prices for many different strikes by the formula:

$$c(t_h, s_i) = E_0^Q \left[e^{\int_0^{t_h} r_u du} (s(t_h) - s_i)^+ \right] = \sum_{j=i}^{n-1} (s_j - s_i) p(t_h, s_j)$$

Where the first equality is the classic risk neutral valuation, and the second is where we have discretised the distribution of s. Here the sum is simply over all our grid points where the option is in the money. We added to this function that you could input a vector of strikes yourself such that the final grid would be prices for specified strikes and maturities, where the strikes typically wouldn't be the points as the spot price points.

Using $m = 6400$ and $n = 100$ we get the following prices for the call options shown in table 4.

Now, assuming that our k and s points are the same, we have in the

$t_h \setminus K$	0.7	0.75	0.8	0.85	0.9	0.95	1	1.05	1.1	1.15	1.2	1.25	1.3
0	0.3	0.25	0.2	0.15	0.1	0.05	0	0	0	0	0	0	0
0.25	0.28963	0.24023	0.19117	0.14374	0.09990	0.06265	0.03472	0.01753	0.00797	0.00329	0.00125	0.00044	0.00017
0.5	0.27981	0.23181	0.18529	0.14233	0.10413	0.07213	0.04716	0.02961	0.01769	0.01009	0.00550	0.00288	0.00154
0.75	0.27101	0.22509	0.18148	0.14221	0.10769	0.07867	0.05544	0.03815	0.02547	0.01652	0.01043	0.00642	0.00398
1	0.26327	0.21957	0.17866	0.14229	0.11044	0.08352	0.06158	0.04469	0.03176	0.02214	0.01514	0.01017	0.00686
1.25	0.25638	0.21483	0.17630	0.14230	0.11255	0.08725	0.06637	0.04992	0.03697	0.02699	0.01943	0.01382	0.00985
1.5	0.25015	0.21060	0.17417	0.14216	0.11413	0.09017	0.07020	0.05420	0.04135	0.03120	0.02329	0.01722	0.01277
1.75	0.24444	0.20673	0.17216	0.14186	0.11529	0.09248	0.07331	0.05775	0.04506	0.03486	0.02674	0.02036	0.01554
2	0.23913	0.20311	0.17020	0.14140	0.11611	0.09431	0.07586	0.06073	0.04824	0.03805	0.02982	0.02323	0.01813
2.25	0.23413	0.19967	0.16826	0.14079	0.11663	0.09574	0.07795	0.06323	0.05097	0.04085	0.03256	0.02583	0.02054
2.5	0.22940	0.19637	0.16632	0.14006	0.11692	0.09684	0.07966	0.06535	0.05332	0.04331	0.03501	0.02820	0.02276
2.75	0.22489	0.19319	0.16439	0.13921	0.11700	0.09767	0.08106	0.06713	0.05536	0.04546	0.03720	0.03034	0.02480
3	0.22056	0.19009	0.16244	0.13826	0.11690	0.09827	0.08219	0.06864	0.05711	0.04736	0.03915	0.03228	0.02667
3.25	0.21639	0.18707	0.16048	0.13723	0.11666	0.09866	0.08309	0.06990	0.05862	0.04902	0.04089	0.03403	0.02838
3.5	0.21236	0.18412	0.15852	0.13612	0.11628	0.09889	0.08379	0.07095	0.05991	0.05047	0.04243	0.03561	0.02994
3.75	0.20845	0.18122	0.15654	0.13495	0.11579	0.09896	0.08432	0.07181	0.06102	0.05175	0.04381	0.03703	0.03137
4	0.20465	0.17837	0.15456	0.13371	0.11520	0.09891	0.08469	0.07251	0.06196	0.05285	0.04502	0.03831	0.03266
4.25	0.20095	0.17557	0.15258	0.13243	0.11452	0.09874	0.08493	0.07306	0.06274	0.05381	0.04610	0.03945	0.03383
4.5	0.19735	0.17281	0.15059	0.13111	0.11377	0.09846	0.08504	0.07348	0.06340	0.05464	0.04704	0.04047	0.03490
4.75	0.19383	0.17009	0.14860	0.12975	0.11295	0.09810	0.08506	0.07378	0.06393	0.05534	0.04787	0.04138	0.03585
5	0.19039	0.16741	0.14661	0.12835	0.11207	0.09766	0.08497	0.07398	0.06435	0.05594	0.04859	0.04219	0.03671

Table 4: Calculated call option prices for different maturities (rows) and strikes (columns)

discrete setting that :

$$\begin{aligned}
\delta_k^+ c &= \frac{1}{\Delta k^+}\left[\sum_{j=i+1}^{n-1}(s_j - k_{i+1})p(s_j) - \sum_{j=i}^{n-1}(s_j - k_i)p(s_j)\right] \\
&= \frac{1}{\Delta k^+}\left[-(s_i - k_i)p(s_i) - \sum_{j=i+1}^{n-1}\Delta k^+ p(s_j)\right] \\
&= -\sum_{j=i+1}^{n-1} p(s_j) \iff \\
\delta_k^- c &= -\sum_{j=i}^{n-1} p(s_j)
\end{aligned}$$

By substituting this into our standard result for discrete double differentiation we get:

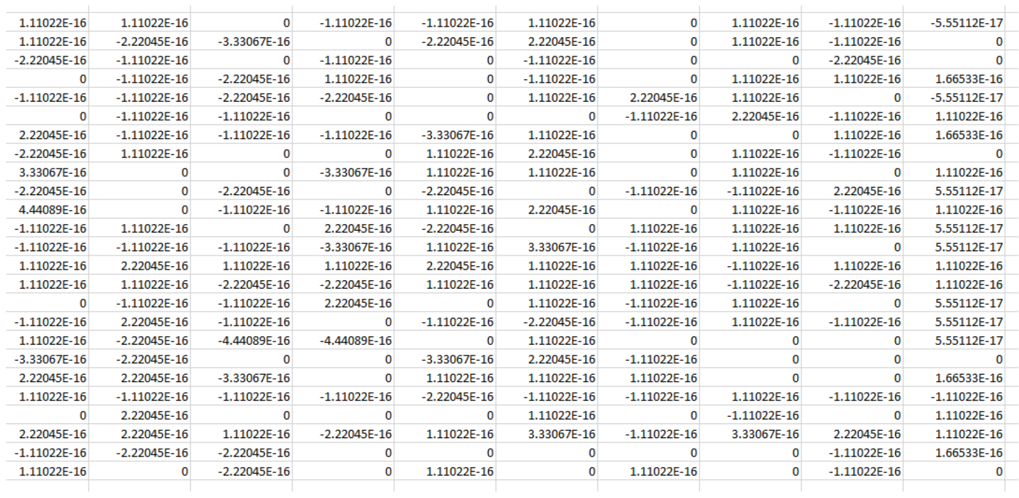
$$\begin{aligned}
\delta_{kk} c &= \frac{2}{k_{i+1} - k_{i-1}}\left[\delta_k^+ c - \delta_k^- c\right] \\
&= \frac{2}{k_{i+1} - k_{i-1}}\left[-\sum_{j=i+1}^{n-1} p(s_j) + \sum_{j=i}^{n-1} p(s_j)\right] \\
&= \frac{2p(s_i)}{k_{i+1} - k_{i-1}}
\end{aligned}$$

This is greater or equal to 0 if $p(s_i)$ is non negative. Using $\theta = 1$ i.e. the fully implicit scheme this is always the case for $\mu = 0$ or if dynamic up- and down winding is used for the first derivative.

We're then asked to show numerically that for $r = \mu = 0$ the initial option prices satisfy

$$c(t_{h+1}) = [I + (1-\theta)\Delta t\bar{A}][I - \theta\Delta t\bar{A}]^{-1}c(t_h)$$

which is the discetized Dupire forward equation. We made another excel function in C++ called xNumericTest, which would do this exact forward roll and compare the difference in price between our forward grid and the value from the roll. This time we did $n = m = 25$ to keep the grid compact. We then used the strikes equal to the underlying s points. Otherwise than setting $r = \mu = 0$ we did not change any parameters. Our errors while using the Crank-Nicolson scheme can be seen in figure 7.



1.11022E-16	1.11022E-16	0	-1.11022E-16	-1.11022E-16	1.11022E-16	0	1.11022E-16	-1.11022E-16	-5.55112E-17
1.11022E-16	-2.22045E-16	-3.33067E-16	0	-2.22045E-16	2.22045E-16	0	1.11022E-16	-1.11022E-16	0
-2.22045E-16	-1.11022E-16	0	-1.11022E-16	0	-1.11022E-16	0	0	-2.22045E-16	0
0	-1.11022E-16	-2.22045E-16	1.11022E-16	0	-1.11022E-16	0	1.11022E-16	1.11022E-16	1.66533E-16
-1.11022E-16	-1.11022E-16	-2.22045E-16	-2.22045E-16	0	1.11022E-16	2.22045E-16	1.11022E-16	0	-5.55112E-17
0	-1.11022E-16	-1.11022E-16	0	0	0	-1.11022E-16	2.22045E-16	-1.11022E-16	1.11022E-16
2.22045E-16	-1.11022E-16	-1.11022E-16	-1.11022E-16	-3.33067E-16	1.11022E-16	0	0	1.11022E-16	1.66533E-16
-2.22045E-16	1.11022E-16	0	0	1.11022E-16	2.22045E-16	0	1.11022E-16	-1.11022E-16	0
3.33067E-16	0	0	-3.33067E-16	1.11022E-16	1.11022E-16	0	1.11022E-16	0	1.11022E-16
-2.22045E-16	0	-2.22045E-16	0	-2.22045E-16	0	-1.11022E-16	-1.11022E-16	2.22045E-16	5.55112E-17
4.44089E-16	0	-1.11022E-16	-1.11022E-16	1.11022E-16	2.22045E-16	0	1.11022E-16	-1.11022E-16	1.11022E-16
-1.11022E-16	1.11022E-16	0	2.22045E-16	-2.22045E-16	0	1.11022E-16	1.11022E-16	1.11022E-16	5.55112E-17
-1.11022E-16	-1.11022E-16	-1.11022E-16	-3.33067E-16	1.11022E-16	3.33067E-16	-1.11022E-16	1.11022E-16	0	5.55112E-17
1.11022E-16	2.22045E-16	1.11022E-16	1.11022E-16	2.22045E-16	1.11022E-16	1.11022E-16	-1.11022E-16	1.11022E-16	1.11022E-16
1.11022E-16	1.11022E-16	-2.22045E-16	-2.22045E-16	1.11022E-16	1.11022E-16	1.11022E-16	-1.11022E-16	-2.22045E-16	1.11022E-16
0	-1.11022E-16	-1.11022E-16	2.22045E-16	0	1.11022E-16	-1.11022E-16	1.11022E-16	0	5.55112E-17
-1.11022E-16	2.22045E-16	-1.11022E-16	0	-1.11022E-16	-2.22045E-16	-1.11022E-16	1.11022E-16	-1.11022E-16	5.55112E-17
1.11022E-16	-2.22045E-16	-4.44089E-16	-4.44089E-16	0	1.11022E-16	0	0	0	5.55112E-17
-3.33067E-16	-2.22045E-16	0	0	-3.33067E-16	2.22045E-16	-1.11022E-16	0	0	0
2.22045E-16	2.22045E-16	-3.33067E-16	0	1.11022E-16	1.11022E-16	1.11022E-16	0	0	1.66533E-16
1.11022E-16	-1.11022E-16	-1.11022E-16	-1.11022E-16	-2.22045E-16	-1.11022E-16	-1.11022E-16	1.11022E-16	-1.11022E-16	-1.11022E-16
0	2.22045E-16	0	0	0	1.11022E-16	0	-1.11022E-16	0	1.11022E-16
2.22045E-16	2.22045E-16	1.11022E-16	-2.22045E-16	1.11022E-16	3.33067E-16	-1.11022E-16	3.33067E-16	2.22045E-16	1.11022E-16
-1.11022E-16	-2.22045E-16	-2.22045E-16	0	0	0	0	0	-1.11022E-16	1.66533E-16
1.11022E-16	0	-2.22045E-16	0	1.11022E-16	0	1.11022E-16	0	-1.11022E-16	0

Figure 7: Errors between forward scheme prices and the Dupire forward roll

This is a excerpt of the full error grid, but these were the biggest errors as for higher strikes the errors was orders of magnitude smaller. All values are so close to 0 that we can safely state that the initial option prices satisfy the Dupire forward equation. Here the roll is done from maturity $T = 0$ to $T = 0.2$ and so on meaning our time step is in terms of maturity and not time as with the backwards scheme.

This is useful in two ways. Firstly instead of constructing a grid of strikes and maturities with option prices by calculating probabilities, and then transforming these to the prices, we can do it all in one go by using boundary conditions for the option price at $T = 0$ and the Dupire forward scheme. We can in fact even use a set of forward volatilities for the individual volatilities.

Secondly we can for a set of given option prices across strikes and maturities find an implied volatility surface by solving the following equation for σ:

$$c(t_{h+1}) = \left[I + (1-\theta)\Delta t\left(\frac{1}{2}K^2\sigma^2\delta_{KK}\right)\right]\left[I - \theta\Delta t\left(\frac{1}{2}K^2\sigma^2\delta_{KK}\right)\right]^{-1} c(t_h)$$

6. Barrier option

Another exotic that we could easily price with our finite difference grid is a down-and-out call option. We can implement this as with the American option by checking whether the stock is below the barrier at each time step. If it is then we set the price equal to 0.

We do the same convergence analysis as in section 2 and get the prices in table 5, where the barrier is set to 0.874446574818369 which is on the S grid. We compare with $P^{(mp)} = 0.0443055992485955$ and do a linear regression on the $\log - \log$ space to get $\hat{\alpha} = 0.88168996136$

However as we saw in the American case when doing this we are essentially pricing a Bermudan down-and-out, i.e. the option is only out if the price of the underlying is below the barrier at specific time points, rather then checking continuously. So when increasing the number of timepoints the price has to converge both to a real down and out option, and it has to converge to the right price.

To circumvent this we could in stead kill the price by setting $\mu = \sigma = 0$ for gridpoints below the barrier. Thus the backward equation for $S(i) \leq B$ is

$$\partial_t P = rP$$

and since the boundary condition is zero the price will be zero. We have also done the convergence analysis with this implementation where we compare with $P'^{(mp)} = 0.0443055304612597$ and get $\hat{\alpha'} = 2.00535578285$.

Thus, as with the American in section 2 we see that the precision in the Bermudan case is $\alpha \approx 1$ while adapting μ and σ gives the same convergence as for a European call.

m	Simple	Better
10	0.0554373950296327	0.0442795454691022
25	0.0506411824710417	0.0443094140974540
50	0.0481677673539233	0.0443065034062958
75	0.0471216610581398	0.0443059630420463
100	0.0465312346873536	0.0443057738194710
150	0.0458802723252707	0.0443056386302840
200	0.0455264054022633	0.0443055913081206
400	0.0449506747696226	0.0443055456731115
800	0.0446381809194380	0.0443055342639124
1600	0.0444745939135425	0.0443055314115832
3200	0.0443907776265891	0.0443055306985104
6400	0.0443483370213223	0.0443055305202421

Table 5: Calculated IV across different number of time steps with both methods