QF4102 - Financial Modelling

Assignment 2

G46

Joel Liew - A0004624U

Samuel Tow - A0102724U

1a)

|  |  |
| --- | --- |
|  | Output |
| European Call Option (Finite Difference)  fd\_ex3(0.005,0.02,0.5,0.45,0.35,0.25) | -5.8188e+12 |
| European Call Option (BS-formula) bs\_call(0.5,0.45,0.005,0.25,0.35,0.02) | 0.062441584897532 |

1b)

To determine the lower bound, we just need to ensure that the monotonicity condition is fulfilled.

Hence, from coef\_0 in fd\_ex3.m,

1-sig^2\*isq\*dt-(r-q)\*i\*dt > 0

After some manipulations,

dt = 1/((sig\*I)^2+r\*I)

(Note: We drop the ‘q’ term because it is positive, and we want to get the highest upper bound of N)

1c)

|  |  |
| --- | --- |
| European Call Option (Finite Difference with Condition)  fd\_ex3\_cond(0.005,0.02,0.5,0.45,0.35,0.25) | 0.062358767 |

1d)

|  |  |
| --- | --- |
| N=559, dt=T/559 | Value=0.06236 |
| N=434 dt=T/434 | 0.06626 |
| N=429 dt=T/429 | -0.10682 |
| N=424 dt=T/424 | -2.2070\*10^2 |
| N=419 dt=T/419 | -1.5167\*10^4 |
| N=409 dt/T/409 | -6.506\*10^7 |

**Comment**

The value obtained when N is bigger than 434 are is rather accurate. However, beyond that, the value becomes more unpredictable. When N goes smaller to 429 the answer already becomes negative. Going even lower will return meaningless results which decreases at an increasing rate.

1e)

|  |  |
| --- | --- |
| American Call Option (Finite Difference with Condition)  FD\_ex\_call(0.5,0.45,0.005,0.25,0.35,0.01,0.01,0.02)  N=559 dt=T/559 | 0.062671 |
| European Call Option (Finite Difference with Condition)  fd\_ex3\_cond(0.005,0.02,0.5,0.45,0.35,0.25) | 0.062358767 |

**Comment**

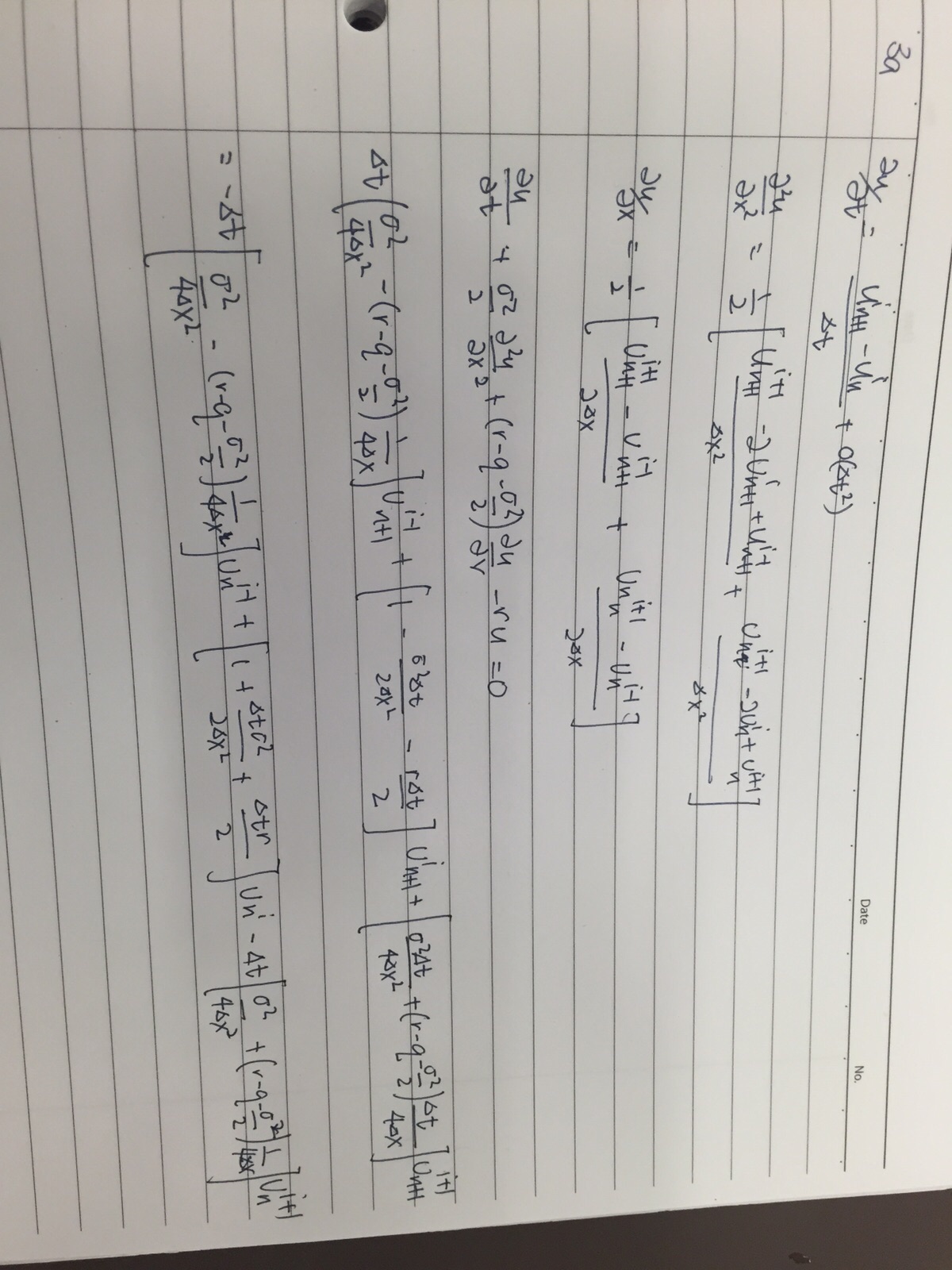
The amercian call value obtained is slightly larger than the value from the european call formula. This is because of the additional value of the option to exercise earlier i.e. before maturity.

2)

|  |  |  |
| --- | --- | --- |
| European Call Option (BS-formula)  bs\_call(0.88,0.45,0.05,0.25,0.34,0.03) | 0.429015522127195 | |
| European Call Option, Transformed (Thomas Algorithm) | dt | Output Value |
| fd\_eur\_call\_thomas(0.05,0.03,0.88,0.45,0.34,0.25,0.025,0.025) | 0.025 | 0.431497158493728 |
| fd\_eur\_call\_thomas(0.05,0.03,0.88,0.45,0.34,0.25,0.0125,0.025) | 0.0125 | 0.431495032346675 |
| fd\_eur\_call\_thomas(0.05,0.03,0.88,0.45,0.34,0.25,0.005,0.025) | 0.005 | 0.431494122942625 |
| fd\_eur\_call\_thomas(0.05,0.03,0.88,0.45,0.34,0.25,0.0025,0.025) | 0.0025 | 0.431493886593200 |
| fd\_eur\_call\_thomas(0.05,0.03,0.88,0.45,0.34,0.25,0.00125,0.025) | 0.00125 | 0.431493781529807 |

**Comment**

The output from the FD method with Thomas Algorithm is above the value of the call option using the formula (for all dt). This can be explained by O(dt + dx^2). For this question, we only change the value of dt, without changing the value of dx. Since the order of dt is 1, the convergence is slow. If we were to change the dx also, we should get approximately the same results.

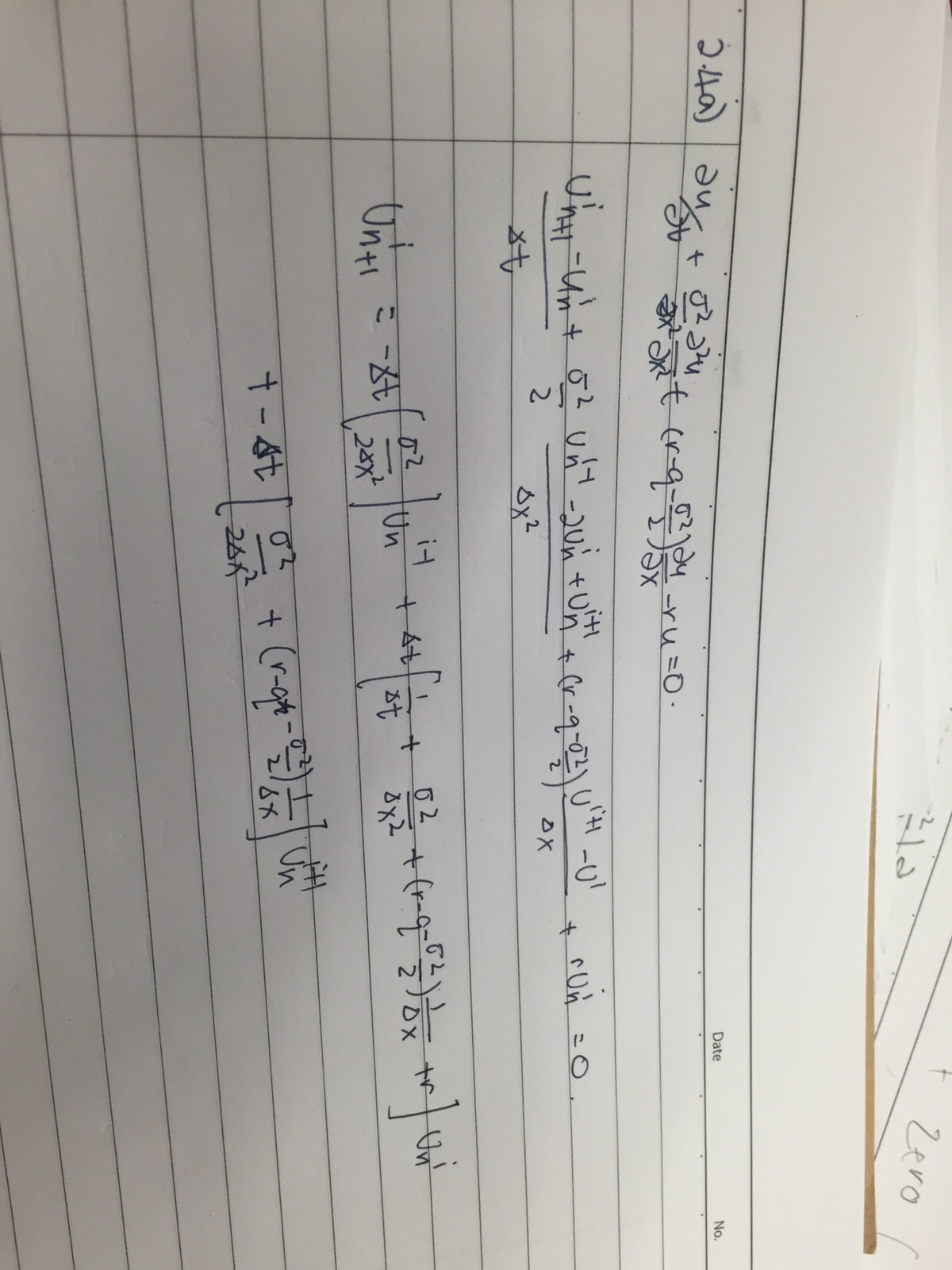
3)

**Derivation of the scheme:**

|  |  |
| --- | --- |
| European call Option (Crank-Nicolson Finite Difference)  FD\_im\_crank(0.95,1,.05,.25,.4,.05,.01,.03) | 0.056562735 |
| European Call Option (BS-formula)  bs\_call(0.95,1,0.05,0.25,0.4,0.03) | 0.056591991 |

**Comment**

The values obtained are very close to each other. This is probably due to the O(dt^2+dx^2), whereby the approximate answer converges very quickly to the true value.

4)

**Derivation of the scheme:**

|  |  |  |
| --- | --- | --- |
| Relaxation Factor | No of Iterations | Output Value |
| 1.1 | 14 | 0.1313637 |
| 1.2 | 10 | 0.13136719 |
| 1.3 | 15 | 0.131364834 |
| 1.4 | 15 | 0.1313624238 |
| 1.5 | 21 | 0.13136864458 |
| 1.6 | 26 | 0.13136505699 |
| 1.7 | 35 | 0.13136899879 |
| 1.8 | 53 | 0.13136877112148 |
| 1.9 | 113 | 0.131365786036172 |

Hence we will choose relaxation factor = 1.2

|  |  |
| --- | --- |
| European Put Option (SOR Method) , Relaxation Factor = 1.2  FD\_newSOR\_put(S0,.X,r,T,sig,ds,dt,q,w) | Output Value |
| FD\_newSOR\_put(.73,.85,0.03,0.25,.35,0.05,0.05,0.01,1.2) | 0.131365 |
| FD\_newSOR\_put(.73,.85,0.03,0.25,.35,0.025,0.025,0.01,1.2) | 0.1315436 |
| FD\_newSOR\_put(.73,.85,0.03,0.25,.35,0.0125,0.0125,0.01,1.2) | 0.1316305 |
| FD\_newSOR\_put(.73,.85,0.03,0.25,.35,0.00625,0.00625,0.01,1.2) | 0.131735 |

4c) Getting the vectors

v1 = FD\_newSOR\_putvector(.73,.85,0.03,0.25,0.35,0.05,0.05,0.01,1.2)

v2 = FD\_newSOR\_putvector(.73,.85,0.03,0.25,0.35,0.025,0.025,0.01,1.2)

v3 = FD\_newSOR\_putvector(.73,.85,0.03,0.25,0.35,0.0125,0.0125,0.01,1.2)

v4 = FD\_newSOR\_putvector(.73,.85,0.03,0.25,0.35,0.00625,0.00625,0.01,1.2)

Note: The vectors generated are the values of the option when T=0

We then compute (dt,dx)=max|V(dt,dx)-V(0.5\*dt,0.5%dx)|

|  |  |
| --- | --- |
| e1  compare(v1,v2) | 0.015351343039817 |
| e2  compare(v2,v3) | 0.007686424515324 |
| e3  compare(v3,v4) | 0.003831500419844 |

|  |  |
| --- | --- |
| e1/e2 | 1.9972020813 |
| e2/e3 | 2.006113447232 |

We can clearly see that the convergence rate is 2^1 hence order=1.

5)

10,000 Price Bundles

|  |  |  |  |
| --- | --- | --- | --- |
| Monte Carlo for Call Option | X | Value | Standard Errors |
| mc\_call(10000,20,0.9) | 0.9 | 0.348745478948539 | 0.002117523131967 |
| mc\_call(10000,20,1.0) | 1.0 | 0.257419635041072 | 0.001929266946372 |
| mc\_call(10000,20,1.1) | 1.1 | 0.176565031222452 | 0.001811847088917 |

50,000 Price Bundles

|  |  |  |  |
| --- | --- | --- | --- |
| Monte Carlo for Call Option | X | Value | Standard Errors |
| mc\_call(50000,20,0.9) | 0.9 | 0.349210051659551 | 0.001026433240045 |
| mc\_call(50000,20,1.0) | 1.0 | 0.25753526298537 | 0.000566158551258 |
| mc\_call(50000,20,1.1) | 1.1 | 0.177109758819777 | 0.000874893109258 |

100,000 Price Bundles

|  |  |  |  |
| --- | --- | --- | --- |
| Monte Carlo for Call Option | X | Value | Standard Errors |
| mc\_call(100000,20,0.9) | 0.9 | 0.349133764645447 | 0.000728605042913 |
| mc\_call(100000,20,1.0) | 1.0 | 0.257724398997125 | 0.000617779851916 |
| mc\_call(100000,20,1.1) | 1.1 | 0.177274991686365 | 0.000504417951007 |

**With Control Variate (using ⅓\*(max(S1-X,0)+max(S2-X,0)+max(S3-X,0)))**

10,000 Price Bundles

|  |  |  |  |
| --- | --- | --- | --- |
| Monte Carlo for Call Option | X | Value | Standard Errors |
| MC\_MA\_EurCall(0.9,10000) | 0.9 | 0.348721368114775 | 0.000529424775985 |
| MC\_MA\_EurCall(1,10000) | 1.0 | 0.257625551990408 | 0.000566402227206 |
| MC\_MA\_EurCall(1.1,10000) | 1.1 | 0.177479974337519 | 0.000575962593609 |

50,000 Price Bundles

|  |  |  |  |
| --- | --- | --- | --- |
| Monte Carlo for Call Option | X | Value | Standard Errors |
| MC\_MA\_EurCall(0.9,50000) | 0.9 | 0.349026055992790 | 0.000288929699255 |
| MC\_MA\_EurCall(1,50000) | 1.0 | 0.257566491221280 | 0.000191370314789 |
| MC\_MA\_EurCall(1.1,50000) | 1.1 | 0.177327743270607 | 0.000213838306288 |

100,000 Price Bundles

|  |  |  |  |
| --- | --- | --- | --- |
| Monte Carlo for Call Option | X | Value | Standard Errors |
| MC\_MA\_EurCall(0.9,100000) | 0.9 | 0.34899903656124 | 0.000205049550516 |
| MC\_MA\_EurCall(1,100000) | 1.0 | 0.257492541271792 | 0.000194676723516 |
| MC\_MA\_EurCall(1.1,100000) | 1.1 | 0.177292932914285 | 0.000116039939022 |

**Comment**

The value of the option is similar to what we obtained using the crude monte carlo. However, standard errors are much smaller than the crude monte carlo. Interestingly, the standard errors from the MC with control variate does not decrease much as number of price bundles increases.