Assignment 5

David Fleischer
MACF 402 - Mathematical & Computational Finance II

December 6, 2015

Question 1

Question 1 (a): By expanding $y\left(\Delta(m+1)\right)$ as a Taylor series about the point (Δm) verify the Taylor series expansion

$$\delta^+ y_m = \Delta y'(\Delta m) + \frac{1}{2}(\Delta)^2 y''(\Delta m) + \cdots$$

What are the assumptions?

Solution 1 (a): If y has a continuous derivative up to order k on the interval $[\Delta m, \Delta(m+1)]$ and $y^{(k+1)}$ exists on $(\Delta m, \Delta(m+1))$ then, by Taylor's theorem, the expansion of $y(\Delta(m+1))$ at point (Δm) is

$$\begin{aligned} y(\Delta(m+1)) = y(\Delta m) + (\Delta(m+1) - \Delta m)y'(\Delta m) + \frac{1}{2}(\Delta(m+1) - \Delta m)^2 y''(\Delta m) + \cdots + \\ \frac{1}{k!}(\Delta(m+1) - \Delta m)^k y^{(k)}(\Delta m) + \frac{1}{(k+1)!}(\Delta(m+1) - \Delta m)^{k+1} y^{(k+1)}(\xi) \end{aligned}$$

for $\xi \in (\Delta m, \Delta(m+1))$. Note that

$$\Delta(m+1) - \Delta m = \Delta m + \Delta - \Delta m = \Delta$$

Hence

$$y(\Delta(m+1)) = y(\Delta m) + \Delta y'(\Delta m) + \frac{1}{2}(\Delta)^2 y''(\Delta m) + \cdots + \frac{1}{k!}(\Delta)^k y^{(k)}(\Delta m) + \frac{1}{(k+1)!}(\Delta)^{k+1} y^{(k+1)}(\xi)$$

So,

$$\begin{aligned} y(\Delta(m+1)) - y(\Delta m) &= \Delta y'(\Delta m) + \frac{1}{2}(\Delta)^2 y''(\Delta m) + \cdots + \frac{1}{k!}(\Delta)^k y^{(k)}(\Delta m) + \frac{1}{(k+1)!}(\Delta)^{k+1} y^{(k+1)}(\xi) \\ \iff \delta^+ y_m &= \Delta y'(\Delta m) + \frac{1}{2}(\Delta)^2 y''(\Delta m) + \cdots + \frac{1}{k!}(\Delta)^k y^{(k)}(\Delta m) + \frac{1}{(k+1)!}(\Delta)^{k+1} y^{(k+1)}(\xi) \end{aligned}$$

Now, if we assume that y is infinitely differentiable then we may write the expansion of $y(\Delta(m+1))$ at (Δm) as

$$\begin{aligned} y(\Delta(m+1)) &= y(\Delta m) + \Delta y'(\Delta m) + \frac{1}{2}(\Delta)^2 y''(\Delta m) + \cdots \\ \iff y(\Delta(m+1)) - y(\Delta m) &= \Delta y'(\Delta m) + \frac{1}{2}(\Delta)^2 y''(\Delta m) + \cdots \\ \iff \delta^+ y_m &= \Delta y'(\Delta m) + \frac{1}{2}(\Delta)^2 y''(\Delta m) + \cdots \end{aligned}$$

as desired.

Question 1 (b): Use Taylor's theorem to show that

$$\delta^+ y_m = \Delta y'(z_m)$$

for some point $z_m \in (\Delta m, \Delta(m+1))$

Solution 1 (b): Define α to satisfy

$$y(\Delta(m+1)) = y(\Delta m) + (\Delta(m+1) - \Delta m)\alpha$$

and consider

$$Y(x) = y(\Delta(m+1)) - y(x) - (\Delta(m+1) - x)\alpha$$

We see that Y is continuous on $[\Delta m, \Delta(m+1)]$ and differentiable on $(\Delta m, \Delta(m+1))$, and note

$$\begin{aligned}
Y(\Delta(m+1)) &= y(\Delta(m+1) - y(\Delta(m+1)) - (\Delta(m+1) - \Delta(m+1))\alpha \\
&= y(\Delta(m+1) - y(\Delta(m+1)) - 0 = 0 \\
Y(\Delta m) &= y(\Delta(m+1)) - y(\Delta m) - (\Delta(m+1) - \Delta m)\alpha \\
&= y(\Delta(m+1)) - y(\Delta(m+1)) = 0
\end{aligned}$$

Therefore, by the Mean Value Theorem, we are guaranteed the existence of some $z_m \in (\Delta m, \Delta(m+1))$ such that

$$Y'(z_m) = 0$$

Now, computing Y', we have

$$\begin{aligned}
Y'(x) &= \frac{d}{dx}\Big[y(\Delta(m+1)) - y(x) - (\Delta(m+1) - x)\alpha\Big] \\
&= -y'(x) + \alpha
\end{aligned}$$

Hence

$$\begin{aligned}
Y'(z_m) = 0 &\implies -y'(z_m) + \alpha = 0 \\
&\implies y'(z_m) = \alpha
\end{aligned}$$

So, we are guaranteed the existence of some point $z_m \in (\Delta m, \Delta(m+1))$ such that

$$\begin{aligned}
y(\Delta(m+1)) &= y(\Delta m) + (\Delta(m+1) - \Delta m)y'(z_m) \\
&= y(\Delta m) + \Delta y'(z_m) \\
\iff \delta^+ y_m &= \Delta y'(z_m)
\end{aligned}$$

as desired.

Question 2

Question 2 (a): Verify the Taylor series expansion

$$\delta^- y_m = \Delta y'(\Delta m) - \frac{1}{2}\Delta^2 y''(\Delta m) + \cdots$$

Solution 2 (a): By Taylor's theorem, the expansion of $y(\Delta(m-1))$ at point (Δm) is

$$\begin{aligned} y(\Delta(m-1)) = y(\Delta m) + (\Delta(m-1) - \Delta m)y'(\Delta m) + \frac{1}{2}(\Delta(m-1) - \Delta m)^2 y''(\Delta m) + \cdots + \\ \frac{1}{k!}(\Delta m - \Delta m)^k y^{(k)}(\Delta m) + \frac{1}{(k+1)!}(\Delta(m-1) - m)^{k+1} y^{(k+1)}(\xi) \end{aligned}$$

for $\xi \in (\Delta(m-1), \Delta m)$. Note that

$$\Delta(m-1) - \Delta m = \Delta m - \Delta + \Delta m = -\Delta$$

Hence

$$\begin{aligned} y(\Delta(m-1)) &= y(\Delta m) - \Delta y'(\Delta m) + \frac{1}{2}(-\Delta)^2 y''(\Delta m) - \cdots + \\ &\qquad \frac{1}{k!}(-\Delta)^k y^{(k)}(\Delta m) + \frac{1}{(k+1)!}(-\Delta)^{k+1} y^{(k+1)}(\xi) \\ &= y(\Delta m) - \Delta y'(\Delta m) + \frac{1}{2}(\Delta)^2 y''(\Delta m) - \cdots + \\ &\qquad \frac{(-1)^k}{k!}(\Delta)^k y^{(k)}(\Delta m) + \frac{(-1)^{k+1}}{(k+1)!}(\Delta)^{k+1} y^{(k+1)}(\xi) \end{aligned}$$

So,

$$\begin{aligned} y(\Delta(m-1)) - y(\Delta m) &= -\Delta y'(\Delta m) + \frac{1}{2}(\Delta)^2 y''(\Delta m) - \cdots + \\ &\qquad \frac{(-1)^k}{k!}(\Delta)^k y^{(k)}(\Delta m) + \frac{(-1)^{k+1}}{(k+1)!}(\Delta)^{k+1} y^{(k+1)}(\xi) \\ y(\Delta m) - y(\Delta(m-1)) &= \Delta y'(\Delta m) - \frac{1}{2}(\Delta)^2 y''(\Delta m) + \cdots + \\ &\qquad \frac{(-1)^{k+1}}{k!}(\Delta)^k y^{(k)}(\Delta m) + \frac{(-1)^{k+2}}{(k+1)!}(\Delta)^{k+1} y^{(k+1)}(\xi) \\ \iff \delta^- y_m &= \Delta y'(\Delta m) - \frac{1}{2}(\Delta)^2 y''(\Delta m) + \cdots + \\ &\qquad \frac{(-1)^{k+1}}{k!}(\Delta)^k y^{(k)}(\Delta m) + \frac{(-1)^{k+2}}{(k+1)!}(\Delta)^{k+1} y^{(k+1)}(\xi) \end{aligned}$$

Now, if we assume that y is infinitely differentiable then we may write the expansion of $y(\Delta(m+1))$ at (Δm) as

$$\begin{aligned} y(\Delta(m-1)) &= y(\Delta m) - \Delta y'(\Delta m) + \frac{1}{2}(\Delta)^2 y''(\Delta m) + \cdots \\ y(\Delta m) - y(\Delta(m-1)) &= \Delta y'(\Delta m) - \frac{1}{2}(\Delta)^2 y''(\Delta m) + \cdots \\ \iff \delta^+ y_m &= \Delta y'(\Delta m) - \frac{1}{2}(\Delta)^2 y''(\Delta m) + \cdots \end{aligned}$$

as desired.

Question 2 (b): Show that

$$\delta^- y_m = \Delta y'(z_m)$$

for some point $z_m \in (\Delta(m-1), \Delta m)$.

Solution 2 (b): Define α to satisfy

$$y(\Delta(m-1)) = y(\Delta m) + (\Delta(m-1) - \Delta m)\alpha$$

and consider

$$Y(x) = y(\Delta(m-1)) - y(x) - (\Delta(m-1) - x)\alpha$$

We see that Y is continuous on $[\Delta(m-1), \Delta m]$ and differentiable on $(\Delta(m-1), \Delta m)$, and note

$$\begin{aligned} Y(\Delta(m-1)) &= y(\Delta(m-1) - y(\Delta(m-1)) - (\Delta(m-1) - \Delta(m-1))\alpha \\ &= y(\Delta(m-1) - y(\Delta(m-1)) - 0 = 0 \\ Y(\Delta m) &= y(\Delta(m-1)) - y(\Delta m) - (\Delta(m-1) - \Delta m)\alpha \\ &= y(\Delta(m-1)) - y(\Delta(m-1)) = 0 \end{aligned}$$

Therefore, by the Mean Value Theorem, we are guaranteed the existence of some $z_m \in (\Delta(m-1), \Delta m)$ such that

$$Y'(z_m) = 0$$

Now, computing Y', we have

$$\begin{aligned} Y'(x) &= \frac{d}{dx}\Big[y(\Delta(m-1)) - y(x) - (\Delta(m-1) - x)\alpha\Big] \\ &= -y'(x) + \alpha \end{aligned}$$

Hence

$$\begin{aligned} Y'(z_m) = 0 &\implies -y'(z_m) + \alpha = 0 \\ &\implies y'(z_m) = \alpha \end{aligned}$$

So, we are guaranteed the existence of some point $z_m \in (\Delta m, \Delta(m+1))$ such that

$$\begin{aligned} y(\Delta(m-1)) &= y(\Delta m) + (\Delta(m-1) - \Delta m)y'(z_m) \\ &= y(\Delta m) - \Delta y'(z_m) \\ \iff y(\Delta(m-1)) - y(\Delta m) &= -\Delta y'(z_m) \\ \iff \delta^+ y_m &= \Delta y'(z_m) \end{aligned}$$

as desired.

Question 3

Question 3 (a): Give a Taylor series for $\delta^2 y_m$.

Solution 3 (a): We have

$$\delta^2 y_m = y_{m+1} - 2y_m + y_{m-1} = y_{m+1} - y_m - (y_m - y_{m-1}) = \delta^+ y_m - \delta^- y_m$$

So, from parts (1)-(2) we get

$$\begin{aligned}
\delta^2 y_m &= \delta^+ y_m - \delta^- y_m \\
&= \Big[\Delta y'(\Delta m) + \frac{1}{2}(\Delta)^2 y''(\Delta m) + \cdots\Big] - \Big[\Delta y'(\Delta m) - \frac{1}{2}(\Delta)^2 y''(\Delta m) + \cdots\Big] \\
&= (\Delta)^2 y''(\Delta m) + \frac{2}{4!}(\Delta)^4 y^{(4)}(\Delta m) + \cdots \\
&= (\Delta)^2 y''(\Delta m) + \frac{1}{12}(\Delta)^4 y^{(4)}(\Delta m) + \cdots
\end{aligned}$$

as desired.

Question 3 (b): Similar to part (b) of Questions 1 and 2 show that the error of the second order central difference is proportional to Δ^2.

Solution 3 (b): From (1)-(2) we have the first order Taylor expansions of $y(\Delta(m+1))$ and $y(\Delta(m-1))$ at point (Δm)

$$\begin{aligned}
y(\Delta(m+1)) &= y(\Delta m) + \Delta y'(\xi_1) \\
y(\Delta(m-1)) &= y(\Delta m) - \Delta y'(\xi_2)
\end{aligned}$$

for $\xi_1 \in (\Delta m, \Delta(m+1))$ and $\xi_2 \in (\Delta(m-1), \Delta m)$. Adding the two equations we get

$$\begin{aligned}
y(\Delta(m+1)) + y(\Delta(m-1)) &= 2y(\Delta m) + \Delta\Big[y'(\xi_1) - y'(\xi_2)\Big] \\
y(\Delta(m+1)) - 2y(\Delta m) + y(\Delta(m-1)) &= \Delta\Big[y'(\xi_1) - y'(\xi_2)\Big]
\end{aligned}$$

Since y'' is continuous on the interval $[\Delta(m-1), \Delta(m+1)]$ we may use the intermediate value theorem to guarantee the existence of some $z_m \in (\Delta(m-1), \Delta(m+1))$ such that

$$\frac{y'(\xi_1) + y'(\xi_2)}{2} = y'(z_m)$$

Hence

$$\begin{aligned}
y(\Delta(m+1)) - 2y(\Delta m) + y(\Delta(m-1)) &= \Delta y'(z_m) \\
\iff \delta^2 y_m &= \Delta y'(z_m)
\end{aligned}$$

as desired.

Question 3 (c): Show that $\delta^+\delta^- = \delta^2$.

Solution 3 (c): From the definition of the backwards operator, $\delta^- y_m$, we have

$$\delta^- y_m = y_m - y_{m-1}$$

Hence

$$\begin{aligned} \delta^+ \big(\delta^- y_m\big) &= \delta^+ \big(y_m - y_{m-1}\big) \\ &= \delta^+ y_m - \delta^+ y_{m-1} \\ &= \big(y_{m+1} - y_m\big) - \big(y_m - y_{m-1}\big) \\ &= y_{m+1} - 2y_m + y_{m-1} \\ &= \delta^2 y_m \end{aligned}$$

as desired.

Question 4

Question 4 (a): Verify that the BTCS algorithm may be written in the form

$$\mathbf{B}\vec{U}^{n+1} = \vec{U}^n + \vec{q}^n$$

for $0 \leq n \leq N-1$, a suitable $(J-1) \times (J-1)$ matrix $\mathbf{B}$ and a suitable $(J-1) \times 1$ vector $\vec{q}^n$ (in terms of the boundary conditions in the x coordinate). Specify the initial condition $\vec{U}^0$ in terms of the initial condition of the time coordinate $u(x,0)$.

Solution 4 (a): Under the BTCS algorithm we approximate $\frac{\partial u}{\partial t}$ at some point (x_j, t_{n+1}) by

$$\partial_t u_j^{n+1} \approx \frac{u_j^{n+1} - u_j^n}{\Delta t}$$

and approximate $\frac{\partial^2 u}{\partial x^2}$ at some point (x_j, t_{n+1}) by

$$\partial_x^2 u_j^n \approx \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{\Delta x}$$

Then, the balance equations are

$$\begin{aligned}
\frac{U_j^{n+1} - U_j^n}{\Delta t} &= \frac{U_{j+1}^{n+1} - 2U_j^{n+1} + U_{j-1}^{n+1}}{(\Delta x)^2} \\
\implies U_j^{n+1} &= U_j^n + \nu\left(U_{j+1}^{n+1} - 2U_j^{n+1} + U_{j-1}^{n+1}\right) \quad \text{with } \nu = \frac{\Delta t}{(\Delta x)^2} \\
\implies U_j^n &= -\nu_{j+1}^{n+1} + (1+2\nu)U_j^{n+1} - \nu U_{j-1}^{n+1}
\end{aligned}$$

If we have the initial condition

$$u(x,0) = \lambda(x) \implies U_j^0 = \lambda(x_j) \quad 0 \leq j < J$$

and boundary conditions

$$\begin{aligned}
U_0^n &= \alpha(n\Delta t) \quad 0 < n \leq N \\
U_J^n &= \beta(n\Delta t) \quad 0 < n \leq N
\end{aligned}$$

Then we may write this iterative process in matrix form as

$$\begin{bmatrix} U_1^n \\ \vdots \\ U_j^n \\ \vdots \\ U_{J-1}^n \end{bmatrix} = \begin{bmatrix} (1+2\nu) & -\nu & 0 & \cdots & 0 & 0 \\ -\nu & (1+2\nu) & -\nu & \cdots & 0 & 0 \\ 0 & -\nu & (1+2\nu) & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & -\nu & (1+2\nu) \end{bmatrix} \times \begin{bmatrix} U_1^{n+1} \\ \vdots \\ U_j^{n+1} \\ \vdots \\ U_{J-1}^{n+1} \end{bmatrix} - \begin{bmatrix} \nu\alpha((n+1)\Delta t) \\ 0 \\ \vdots \\ 0 \\ \nu\beta((n+1)\Delta t) \end{bmatrix}$$

Hence

$$\begin{aligned}
\vec{U}^n &= \mathbf{B}\vec{U}^{n+1} - \vec{q}^n \\
\implies \mathbf{B}\vec{U}^{n+1} &= \vec{U}^n + \vec{q}^n
\end{aligned}$$

for $0 \leq n \leq N-1$, $\mathbf{B}$ a $(J-1) \times (J-1)$ matrix, and $\vec{q}^n$ a $(J-1) \times 1$ vector. Finally, using our initial conditions specified above, we may express the vector of initial conditions $\vec{U}^0$ as

$$\vec{U}^0 = \begin{bmatrix} \lambda(x_1) \\ \vdots \\ \lambda(x_j) \\ \vdots \\ \lambda(x_{J-1}) \end{bmatrix}$$

Question 5

Question 5 (a): Investigate the performance of the FTCS algorithm relative to the Black-Scholes price for $J = 10, 30, 50, 70, 90, 110$.

Solution 5 (a): For our implementation of the FTCS solution we use parameters $S_0 = 100, \sigma = 0.2, r = 0.03, T = 1$ with boundaries $S_{min} = 0, S_{max} = 300, t_{min} = 0, t_{max} = 1$. Figure 1 show us the result of choosing N time steps too small to satisfy stability for a given value of J space steps. We see that small errors propagate from the terminal condition (an initial condition in our transformed PDE) through the system to explode at the other boundary.

Plotting the maximum absolute error over a grid of possible time and space steps gives us Figure 2. We see that a boundary of stability forms along a boundary that is (at least impressionistically) proportional to J^2, which is precisely to be expected given the FTCS algorithm's requirement of $\frac{\Delta t}{(\Delta x)^2} \leq \frac{1}{2}$. This proportionality is verified by fitting the model $\sqrt{\hat{N}} = \beta_0 + \beta_1 J$. Such a model was computed and the results are presented in Figure 3. The coefficients of the fitted model are

$$\beta_0 = -1.83529$$
$$\beta_1 = 0.09124584$$

and so we find that the fitted polynomial is

$$\hat{N} = 0.0083258J^2 - 0.334925J + 3.36829$$

Thus, given this domain of S and t, we find that the boundary of stability for N is indeed proportional to J^2.

Looking at the region of stability more closely we also see in Figure 2 periodic behaviour in J for the maximum error, as well as decreasing, with diminishing returns, error in increasing J.

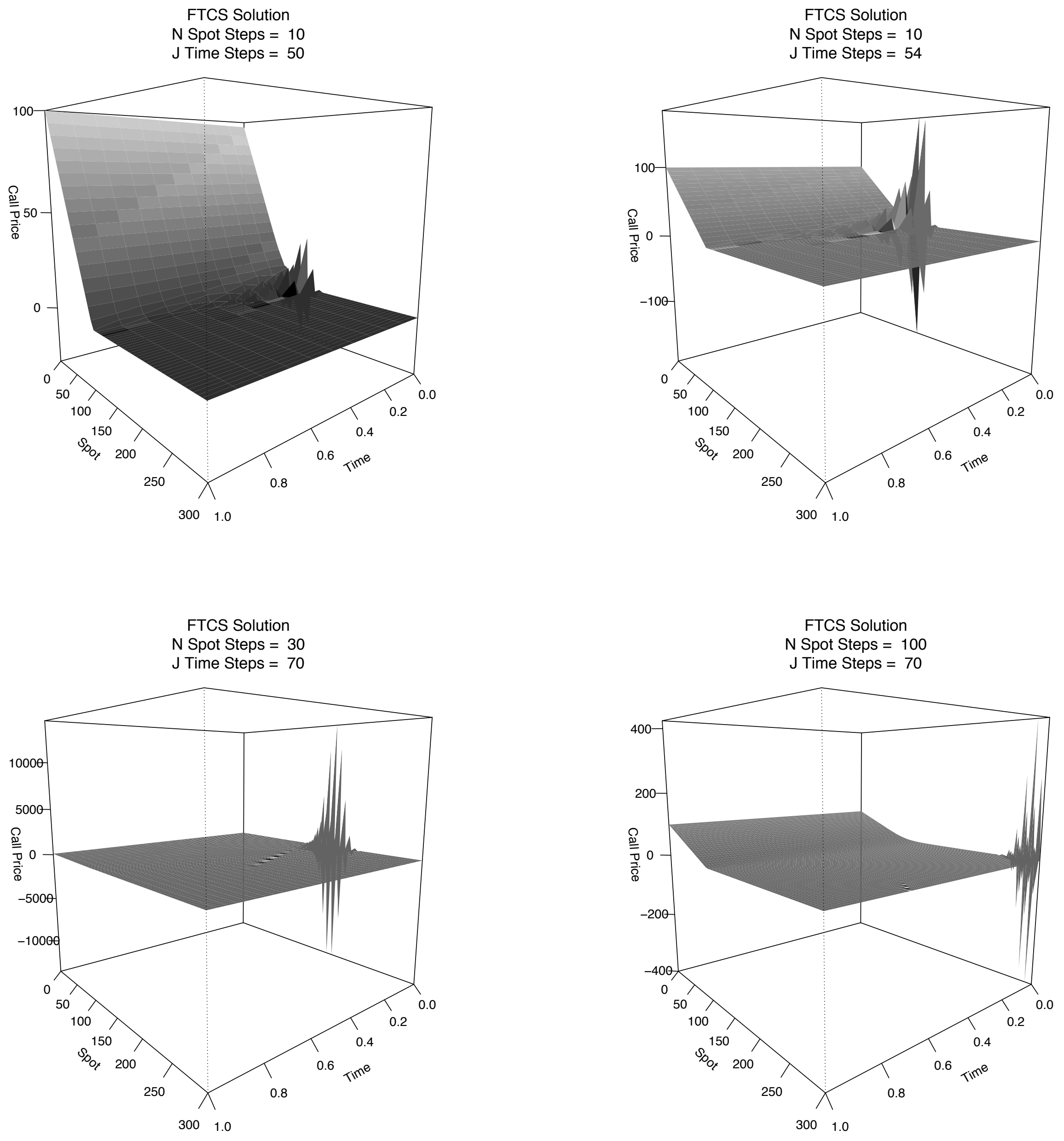
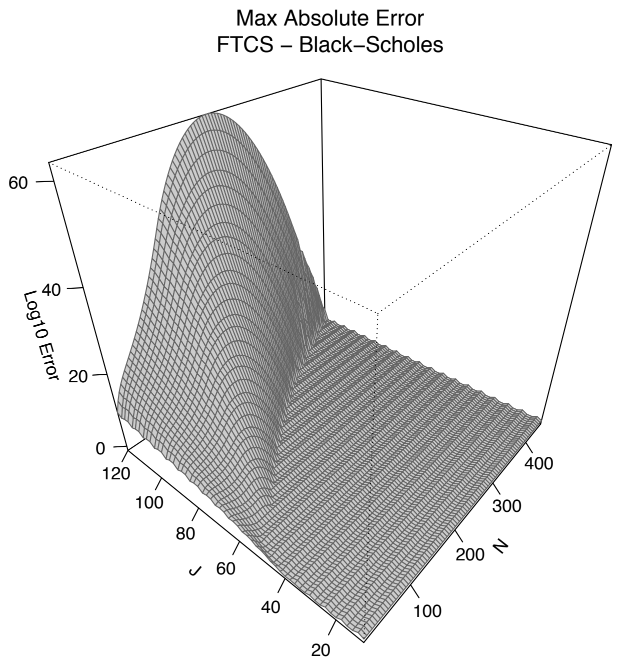


Figure 1: Some examples of unstable FTCS solutions to the Black-Scholes PDE.



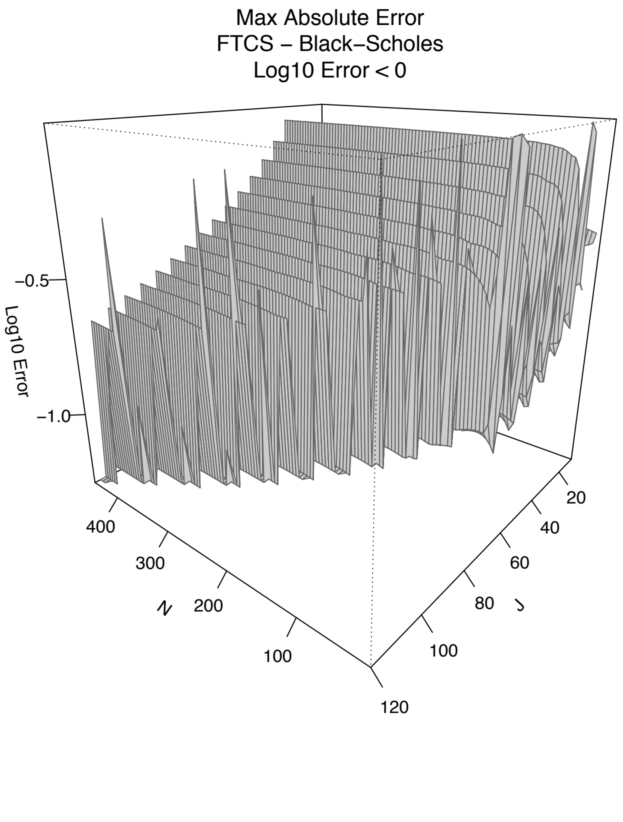


Figure 2: Base 10 log of the max absolute error between the FTCS approximation and the Black-Scholes solution given N time steps and J space steps. Note the downwards periodic behaviour in J.

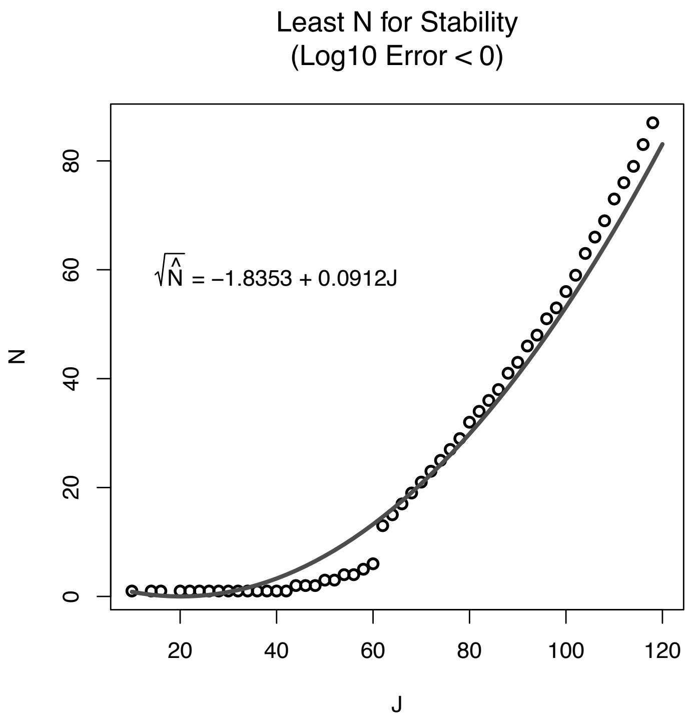


Figure 3: Values of the least N time steps corresponding to stable solutions for a given J number of space steps as well as the fitted linear model in $\sqrt{N}$.

Question 5 (b): Produce surface plots of the approximate value of the European put as a function of the stock-price and time to maturity.

Solution 5 (b): Presented in Figures 4 and 5 are the surfaces for $J = 10, 30, 50, 70, 90$ and 100 with the least N to satisfy stability.

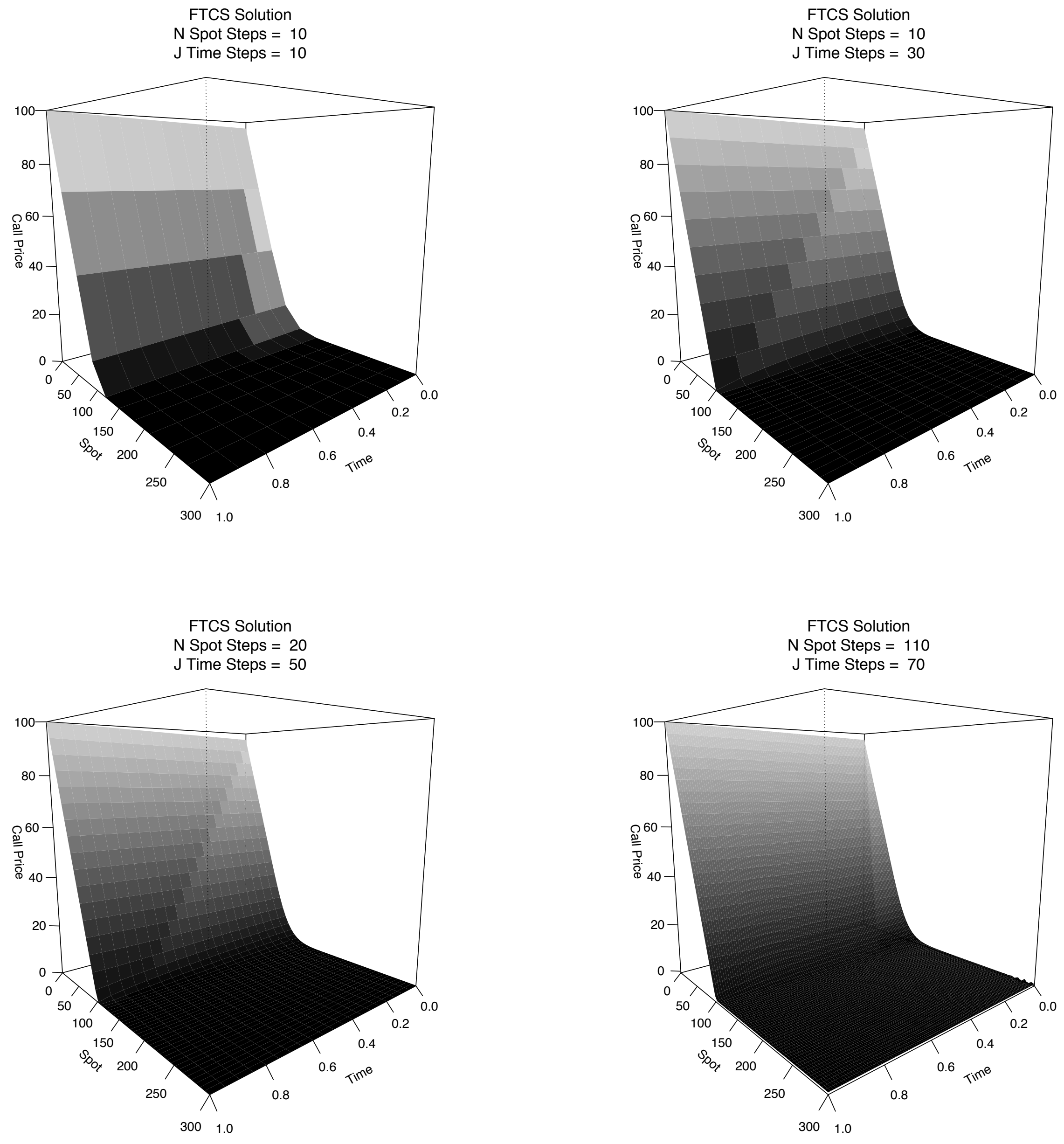
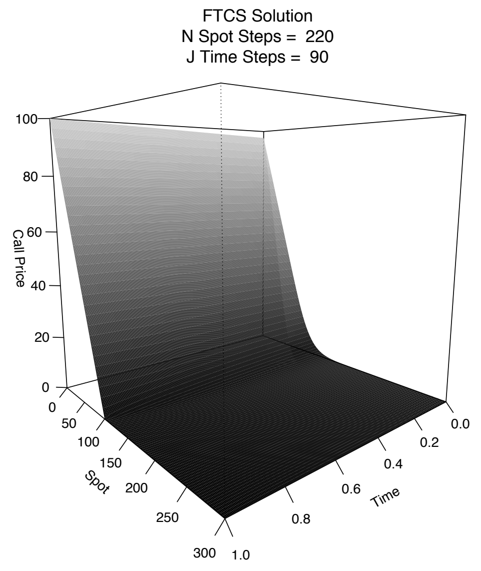


Figure 4: Surface approximations of a European call option from the FTCS algorithm. The N chosen for a given J was the least N such that the surfaces appeared stable.



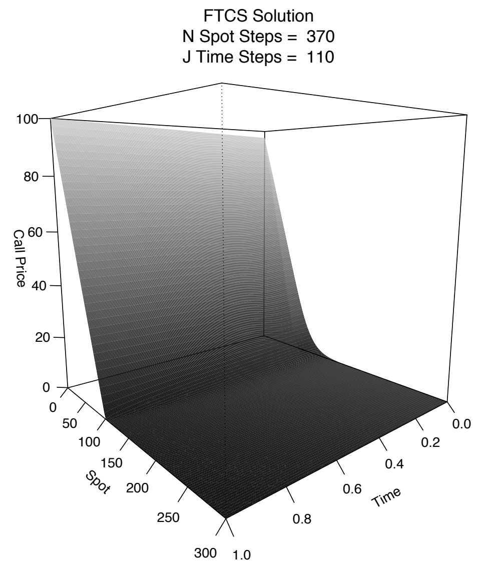


Figure 5: More surface approximations of a European call option from the FTCS algorithm. The N chosen for a given J was the least N such that the surfaces appeared stable.

Question 6

Solution 6 (preamble): We have lognormal dynamics for asset S, hence

$$\begin{aligned} S_t &= S_0 e^{(\mu - \frac{\sigma^2}{2})t + \sigma\sqrt{t}Z} \quad Z \sim N(0,1) \\ \implies \ln S_t &= \ln S_0 + \left(\mu - \frac{\sigma^2}{2}\right)t + \sigma\sqrt{t}Z \\ \implies \ln S_t - \ln S_0 - \left(\mu - \frac{\sigma^2}{2}\right)t &= \sigma\sqrt{t}Z \sim N\left(0, \sigma^2 t\right) \end{aligned}$$

For convenient application of the chain rule we let X_t be the risk factor for the European call option such that

$$X_t = \ln S_t - \ln S_0 - \left(\mu - \frac{\sigma^2}{2}\right)t$$

and note

$$\begin{aligned} \exp\left[X_t + (\mu - \frac{\sigma^2}{2})t + \ln S_0\right] &= \exp\left[\ln S_t - \ln S_0 - \left(\mu - \frac{\sigma^2}{2}\right)t + (\mu - \frac{\sigma^2}{2})t + \ln S_0\right] \\ &= \exp\left[\ln S_t\right] \\ &= S_t \end{aligned}$$

So, from the chain rule we get

$$V'(S_t) = \frac{\partial V}{\partial X_t} = \frac{\partial V}{\partial S_t}\frac{\partial S_t}{\partial X_t}$$

but for a European call option we have $\frac{\partial V}{\partial S_t} = \Phi(d_1)$ and

$$\begin{aligned} \frac{\partial S_t}{\partial X_t} &= \frac{\partial}{\partial X_t}\exp\left[X_t + (\mu - \frac{\sigma^2}{2})t + \ln S_0\right] \\ &= \exp\left[X_t + (\mu - \frac{\sigma^2}{2})t + \ln S_0\right] \\ &\equiv S_t \end{aligned}$$

so

$$V'(S_t) = \Phi(d_1)S_t =: \delta_0$$

Then, for some increment in the value of a call option ΔV over time horizon h, we have

$$\Delta V = V_{t+h} - V_t$$

and from the first order Taylor approximation we find

$$\Delta V \approx V'(S_t)\Delta X_t = \delta_0 \Delta X_t = \delta_0\left(X_{t+h} - X_t\right)$$

But $X_t \sim N\left(0, \sigma^2 t\right)$ so $X_{t+h} - X_t$ is some Brownian increment such that

$$\begin{aligned} \Delta X_t &= X_{t+h} - X_t \sim N\left(0, \sigma^2 h\right) \\ \implies \Delta X_t &= \sigma\sqrt{h}Z \end{aligned}$$

Therefore, over time horizon h and $\delta_0 = \Phi(d_1)S_t$, we may approximate $\Delta V = V_{t+h} - V_t$ as

$$\Delta V \approx \delta_0 \sigma \sqrt{h} Z$$

Question 6 (a): a short position

Solution 6 (a): From the preamble we have

$$\Delta V \approx \delta_0 \sigma \sqrt{h} Z$$

Therefore, for some significance level α and corresponding value-at-risk $\mathrm{VaR}_{1-\alpha}$, we want a short position to satisfy

$$\begin{aligned} 1-\alpha &= \mathbb{P}\left(\Delta V < \mathrm{VaR}_{1-\alpha}\right) \\ &\approx \mathbb{P}\left(\delta_0 \sigma \sqrt{h} Z < \mathrm{VaR}_{1-\alpha}\right) \\ &= \mathbb{P}\left(Z < \frac{\mathrm{VaR}_{1-\alpha}}{\sigma\sqrt{h}\delta_0}\right) \\ \implies \mathrm{VaR}_{1-\alpha} &= z_{1-\alpha}\sigma\sqrt{h}\delta_0 \end{aligned}$$

So, for a short European call position, the 95% VaR over a horizon of 5 days (given 365 days in a year) is

$$\mathrm{VaR}_{1-\alpha} = 1.645\sigma\sqrt{\frac{5}{365}}\delta_0$$

Question 6 (b): a long position

Solution 6 (b): From the preamble we have

$$\Delta V \approx \delta_0 \sigma \sqrt{h} Z$$

Therefore, for some significance level α and corresponding value-at-risk $\mathrm{VaR}_{1-\alpha}$, we want a long position to satisfy

$$\begin{aligned} 1-\alpha = \mathbb{P}\left(\Delta V > \mathrm{VaR}_{1-\alpha}\right) \iff \alpha &= \mathbb{P}\left(\Delta V < \mathrm{VaR}_{1-\alpha}\right) \\ &\approx \mathbb{P}\left(\delta_0 \sigma \sqrt{h} Z < \mathrm{VaR}_{1-\alpha}\right) \\ &= \mathbb{P}\left(Z < \frac{\mathrm{VaR}_{1-\alpha}}{\sigma\sqrt{h}\delta_0}\right) \\ \implies \mathrm{VaR}_{1-\alpha} &= z_{\alpha}\sigma\sqrt{h}\delta_0 \end{aligned}$$

So, for a long European call position, the 95% VaR over a time horizon of 5 days (given 365 days in a year) is

$$\mathrm{VaR}_{1-\alpha} = -1.645\sigma\sqrt{\frac{5}{365}}\delta_0$$

Question 6 (c): Let $S_0 = 100, K = 100, \mu = 0.02, \sigma = 0.5, T = 1$ and simulate the value of a long & short portfolio with $N = 10^2, 10^4, 10^5, 10^6$. Calculate the number of times the 95% VaR was broken.

Solution 6 (c):

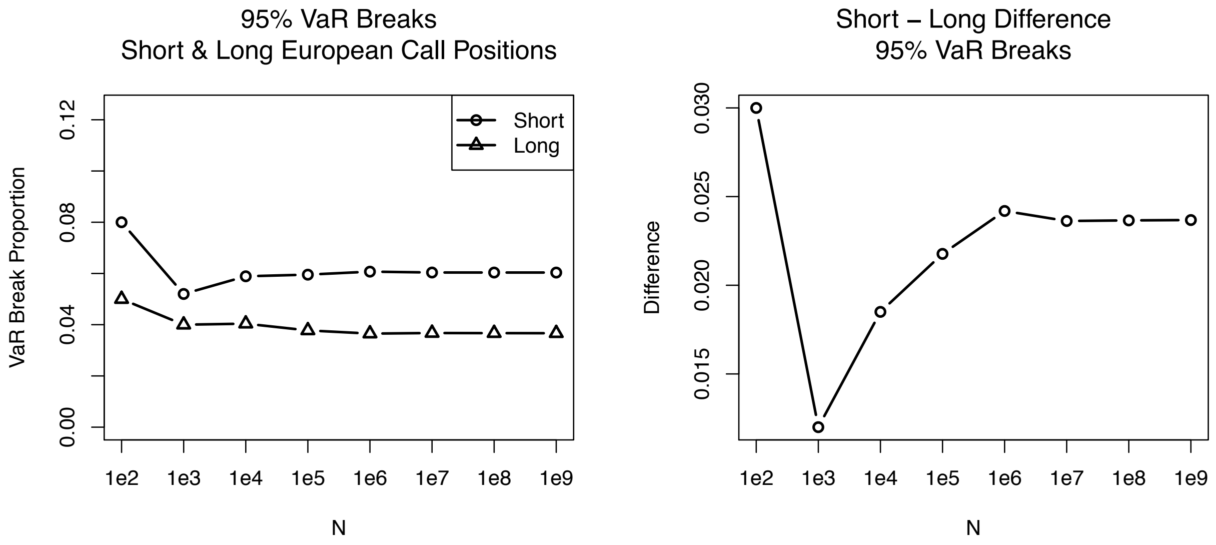


Figure 6: VaR break proportions for a short & long position in a European call. (Left) Average VaR break proportions and (right) Difference in break proportions as a function of N simulations of the underlying asset (lognormal dynamics).

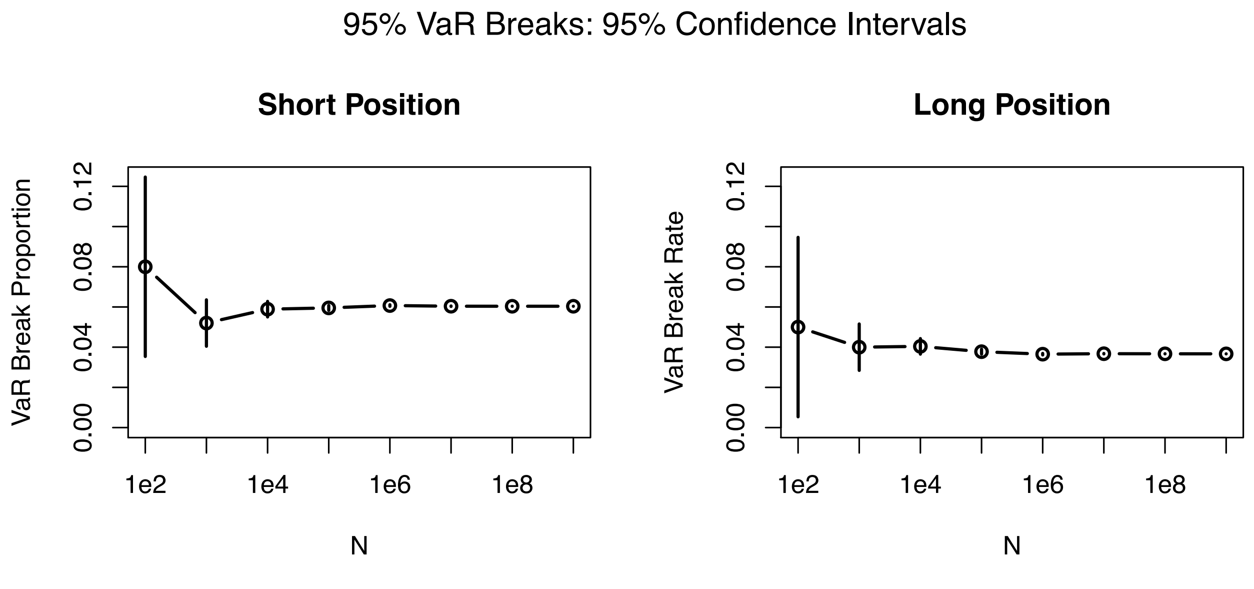


Figure 7: VaR break proportions with 95% confidence intervals for a (left) short position and (right) long position in a European call as a function of N simulations of the underlying asset (lognormal dynamics).

Table 1: Short and long European call position 95% VaR break proportions & standard errors as a function of N simulations of the underlying asset (lognormal dynamics).

N	Short Position	Short Std Error	Long Position	Long Std Error
1e2	0.08	0.0271293	0.05	0.0271293
1e3	0.052	0.00702111	0.04	0.00702111
1e4	0.0589	0.00235437	0.0404	0.00235437
1e5	0.05954	0.000748298	0.03777	0.000748298
1e6	0.060714	0.000238805	0.036524	0.000238805
1e7	0.0603785	7.53213e-05	0.0367551	7.53213e-05
1e8	0.0603627	2.38158e-05	0.0367049	2.38158e-05
1e9	0.0603584	7.53096e-06	0.0366808	7.53096e-06

Appendix A Transformation of the Black-Scholes PDE to the Heat Equation

Let V be the fair price of a European put option written on some risky asset S with risk free rate r and volatility σ^2. Then, it can be shown that V satisfies the partial differential with respect to $V = V(S,t)$

$$\frac{\partial V}{\partial t} + \frac{\sigma^2}{2}S^2\frac{\partial V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0$$

on the grid $S \in (0,\infty), t \in \times[0,T)$ in space and time, respectively. We have the terminal condition, for some strike price $K > 0$,

$$V(S,T) = \max(K - S, 0)$$

and boundary conditions

$$\lim_{S\to 0} V(S,t) = Ke^{-r(T-t)}$$
$$\lim_{S\to\infty} V(S,t) = 0$$

We wish to transform this PDE to the canonical heat equation $\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$. We perform the substitutions

$$S = e^x \iff \log S = x$$
$$t = T - \frac{2\tau}{\sigma^2} \iff \tau = \frac{\sigma^2}{2}(T-t)$$

and let $V(S,t) = W(x,\tau) = W\left(\log S, \frac{\sigma^2}{2}(T-t)\right)$, then

$$\frac{\partial V}{\partial t} = \frac{\partial W}{\partial \tau}\frac{\partial \tau}{\partial t} = -\frac{\sigma^2}{2}\frac{\partial W}{\partial \tau}$$
$$\frac{\partial V}{\partial S} = \frac{\partial W}{\partial x}\frac{\partial x}{\partial S} = \frac{1}{S}\frac{\partial W}{\partial x}$$
$$\frac{\partial^2 V}{\partial S^2} = \frac{\partial}{\partial S}\left(\frac{1}{S}\frac{\partial W}{\partial x}\right) = \frac{1}{S^2}\frac{\partial^2 W}{\partial x^2} - \frac{1}{S^2}\frac{\partial W}{\partial x}$$

Substituting these partial derivatives into our original PDE we get

$$\left[-\frac{\sigma^2}{2}\frac{\partial W}{\partial \tau}\right] + \frac{\sigma^2}{2}S^2\left[\frac{1}{S^2}\frac{\partial^2 W}{\partial x^2} - \frac{1}{S^2}\frac{\partial W}{\partial x}\right] + rS\left[\frac{1}{S}\frac{\partial W}{\partial x}\right] - r\left[W\right] = 0$$
$$\iff -\frac{\sigma^2}{2}\frac{\partial W}{\partial \tau} + \frac{\sigma^2}{2}\frac{\partial^2 W}{\partial x^2} - \frac{\sigma^2}{2}\frac{\partial W}{\partial x} + \frac{\partial W}{\partial x} - rW = 0$$
$$\iff -\frac{\partial W}{\partial \tau} + \frac{\partial^2 W}{\partial x^2} - \frac{\partial W}{\partial x} + \frac{2r}{\sigma^2}\frac{\partial W}{\partial x} - \frac{2r}{\sigma^2}W = 0$$
$$\iff -\frac{\partial W}{\partial \tau} + \frac{\partial^2 W}{\partial x^2} + \left(\frac{2r}{\sigma^2} - 1\right)\frac{\partial W}{\partial x} - \frac{2r}{\sigma^2}W = 0$$

For sanity let $\kappa := \frac{2r}{\sigma^2}$ so

$$\iff -\frac{\partial W}{\partial \tau} + \frac{\partial^2 W}{\partial x^2} + (\kappa - 1)\frac{\partial W}{\partial x} - \kappa W = 0$$

Set $W(x,\tau) = e^{\rho x + \xi\tau}u(x,\tau) = \psi \cdot u(x,\tau)$, then

$$\frac{\partial W}{\partial \tau} = \xi\psi u + \psi\frac{\partial u}{\partial \tau}$$
$$\frac{\partial W}{\partial x} = \rho\psi u + \psi\frac{\partial u}{\partial x}$$
$$\frac{\partial^2 W}{\partial x^2} = \rho^2\psi u + 2\rho\psi\frac{\partial u}{\partial x} + \psi\frac{\partial^2 u}{\partial x^2}$$

then our PDE in W becomes the PDE in u

$$\begin{aligned}
&-\left[\xi\psi u+\psi\frac{\partial u}{\partial\tau}\right]+\left[\rho^2\psi u+2\rho\psi\frac{\partial u}{\partial x}+\psi\frac{\partial^2 u}{\partial x^2}\right]+(\kappa-1)\left[\rho\psi u+\psi\frac{\partial u}{\partial x}\right]-\kappa\psi u=0\\
\iff &-\xi u-\frac{\partial u}{\partial\tau}+\rho^2 u+2\rho\frac{\partial u}{\partial x}+\frac{\partial^2 u}{\partial x^2}+(\kappa-1)\,\rho u+(\kappa-1)\frac{\partial u}{\partial x}-\kappa u=0\\
\iff &-\frac{\partial u}{\partial\tau}+\frac{\partial^2 u}{\partial x^2}+(\kappa-1+2\rho)\frac{\partial u}{\partial x}+(-\xi+\rho^2+(\kappa-1)\rho-\kappa)u=0
\end{aligned}$$

Now, we see that we essentially have the canonical heat equation from the two leftmost terms. In order to rid ourselves of the pesky convection and heat source terms u_x and u we must solve the following system for ρ and ξ,

$$\begin{aligned}
\kappa-1+2\rho&=0\\
-\xi+\rho^2+(\kappa-1)\rho-\kappa&=0
\end{aligned}$$

which has the solution

$$\begin{aligned}
\rho&=-\frac{1}{2}(\kappa-1)=-\frac{1}{2}\left(\frac{2r}{\sigma^2}-1\right)\\
\xi&=-\frac{1}{4}(\kappa+1)^2=-\frac{1}{4}\left(\frac{2r}{\sigma^2}+1\right)^2
\end{aligned}$$

and so we finally have the PDE

$$-\frac{\partial u}{\partial\tau}+\frac{\partial^2 u}{\partial x^2}=0$$

which is precisely the heat equation in x and τ. We now take a moment to notice that our substitution in $\tau:=T-\frac{2t}{\sigma^2}$ has led to a time reversal and so our terminal condition becomes the initial condition

$$\begin{aligned}
V(S,T)&=\max(K-S,0)\\
\implies e^{\rho x}u(x,0)&=\max(K-e^x,0)
\end{aligned}$$

with corresponding boundary conditions

$$\begin{aligned}
\lim_{S\to0}V(S,t)&=Ke^{-r(T-t)}\\
\implies \lim_{x\to-\infty}e^{\rho x+\xi\tau}u(x,\tau)&=Ke^{-r(T-(T-\frac{2\tau}{\sigma^2}))}=Ke^{-r\frac{2\tau}{\sigma^2}}\\
\lim_{S\to+\infty}V(S,t)&=0\\
\implies \lim_{x\to+\infty}e^{\rho x+\xi\tau}u(x,\tau)&=0
\end{aligned}$$

Appendix B Code

B.1 q5_main.cpp

```cpp
#include <iostream>
#include <fstream>
#include <cmath>
#include <vector>
#include "rng.h"

using namespace std;

double stdNormPdf(double x) {
   return 1/sqrt(2 * M_PI) * exp(-1/2.0 * pow(x, 2));
}
double stdNormCdf(double x_in) {
   double x = fabs(x_in);

   double b0 = 0.2316419;
   double b1 = 0.31938530;
   double b2 = -0.356563782;
   double b3 = 1.781477937;
   double b4 = -1.821255978;
   double b5 = 1.330274429;
   double t = 1/(1 + b0 * x);

   double y = 1 - stdNormPdf(x) * (b1*t + b2*pow(t,2) + b3*pow(t,3) + b4*pow(t,4) + b5*pow(t,5));

   if (x_in < 0) {
      y = 1 - y;
   }
   return y;
}
double computeD1(double S, double K, double r, double sigma, double dt) {
   return (log(S / K) + (r + 0.5 * pow(sigma, 2)) * dt) / (sigma * sqrt(dt));
}
double computeD2(double S, double K, double r, double sigma, double dt) {
   return computeD1(S, K, r, sigma, dt) - sigma * sqrt(dt);
}
double bs_price_put(double S, double K, double r, double sigma, double dt) {
   if (dt == 0)
      return fmax(S - K, 0);
   double d1 = computeD1(S, K, r, sigma, dt);
   double d2 = computeD2(S, K, r, sigma, dt);
   return stdNormCdf(-d2) * K * exp(-r * dt) - stdNormCdf(-d1) * S;
}

int main() {
   srand(402);

   // parameters
   double S0 = 100;
   double K = 100;
   double r = 0.03;
   double sigma = 0.2;
   double t_max = 1;
   double S_max = 3 * K;

   int N = 10;
   int J = 5;

   // increments
   double dS = S_max / J;
   double dt = t_max / N;
   double nu = dt / pow(dS, 2);

   // create the tridiagonal matrix values
   vector<double> aa(J - 1), bb(J - 1), cc(J - 1);
   for (int j = 1; j < J; j++) {
      aa.at(j - 1) = (-0.5 * r * dt * j + 0.5 * pow(sigma, 2) * dt * pow(j, 2)) / (1 + r * dt);
```

```
        bb.at(j - 1) = (1 - pow(sigma, 2) * dt * pow(j, 2)) / (1 + r * dt);
        cc.at(j - 1) = (0.5 * r * dt * j + 0.5 * pow(sigma, 2) * dt * pow(j, 2)) / (1 + r * dt);
    }

    // initialize matrix of solutions
    vector<vector<double> > V;
    // matrix.resize(num_of col, vector<double> (num_of_row , init_value));
    V.resize(N + 1, vector<double> (J + 1, 0));

    // place initial & boundary conditions
    for (int j = 0; j < V.at(0).size(); j++) { // initial conditions
        V.at(N).at(j) = fmax(K - j * dS, 0);
    }
    for (int n = 0; n < V.size(); n++) { // boundary conditions
        V.at(n).at(0) = exp(-r * (t_max - n * dt)) * K;
    }

    // FTCS algorithm
    for (int n = V.size() - 2; n > -1; n--) {
        for (int j = 1; j < V.at(0).size() - 1; j++) {
            V.at(n).at(j) = V.at(n + 1).at(j - 1) * aa.at(j - 1) +
                V.at(n + 1).at(j) * bb.at(j - 1) +
                V.at(n + 1).at(j + 1) * cc.at(j - 1);
        }
    }

    // write data
    string filepath = "../../data/q5_dat_N" + to_string(N) + "_J" +
        to_string(J) + ".csv";

    ofstream outfile;
    outfile.open(filepath);

    for (int i = 0; i < V.size(); i++) {
        for (int j = 0; j < V.at(0).size() - 1; j++) {
            outfile << V.at(i).at(j) << ",";
        }
        outfile << V.at(i).at(V.at(0).size() - 1) << endl;
    }
    outfile.close();
}
```

B.2 q6_main.cpp

```
#include <iostream>
#include <fstream>
#include <cmath>
#include <vector>
#include "rng.h"

using namespace std;

double stdNormPdf(double x) {
    return 1/sqrt(2 * M_PI) * exp(-1/2.0 * pow(x, 2));
}
double stdNormCdf(double x_in) {
    double x = fabs(x_in);

    double b0 = 0.2316419;
    double b1 = 0.31938530;
    double b2 = -0.356563782;
    double b3 = 1.781477937;
    double b4 = -1.821255978;
    double b5 = 1.330274429;
    double t = 1/(1 + b0 * x);

    double y = 1 - stdNormPdf(x) * (b1*t + b2*pow(t,2) + b3*pow(t,3) + b4*pow(t,4) + b5*pow(t,5));
```

```cpp
    if (x_in < 0) {
        y = 1 - y;
    }
    return y;
}
double computeD1(double S, double K, double r, double sigma, double dt) {
    return (log(S / K) + (r + 0.5 * pow(sigma, 2)) * dt) / (sigma * sqrt(dt));
}
double computeD2(double S, double K, double r, double sigma, double dt) {
    return computeD1(S, K, r, sigma, dt) - sigma * sqrt(dt);
}
double bs_price_call(double S, double K, double r, double sigma, double dt) {
    if (dt == 0)
        return fmax(S - K, 0);
    double d1 = computeD1(S, K, r, sigma, dt);
    double d2 = computeD2(S, K, r, sigma, dt);
    return stdNormCdf(d1) * S - stdNormCdf(d2) * K * exp(-r * dt);
}

int main() {
    srand(402);
    int n_min = 2; // log10 min to compute
    int n_max = 9; // log10 max to compute

    double S0 = 100;
    double K = 100;
    double mu = 0.02;
    double sigma = 0.5;
    double T = 1.0;
    double h = 5.0/365;
    double z_alpha = 1.645;
    string filepath = "../../data/q6_dat.csv";

    double C0 = bs_price_call(S0, K, mu, sigma, T);
    double d1 = computeD1(S0, K, mu, sigma, T);
    double delta0 = stdNormCdf(d1) * S0;
    double var = z_alpha * sigma * sqrt(h) * delta0;

    // initialize & open csv
    ofstream outfile;
    outfile.open(filepath);
    // write header
    outfile << "N,short,short_se, long, long_se" << endl;

    for (int n = n_min; n <= n_max; n++) {
        int N = pow(10, n);
        vector<double> z = rng::rnorm(N);
        double St, Ct, dC;

        int short_sum = 0;
        int long_sum = 0;

        // compute var breaks
        for (int i = 0; i < N; i++) { // loop over each asset simulation
            St = S0 * exp((mu - 0.5 * pow(sigma, 2)) * h + sigma * sqrt(h) * z.at(i));
            Ct = bs_price_call(St, K, mu, sigma, T - h);
            dC = Ct - C0;
            if (dC > var)
                short_sum++;
            if (dC < -var)
                long_sum++;
        }

        // compute proportions
        double short_pct = short_sum / (double)N;
        double long_pct = long_sum / (double)N;

        // compute SE
        double short_se = sqrt( short_pct * (1 - short_pct) / N );
        double long_se = sqrt( short_pct * (1 - short_pct) / N );

        // write data to csv
```

```
      outfile << N << "," << short_pct << "," << short_se << "," <<
         long_pct << "," << long_se << endl;
   }
   outfile.close();
}
```

B.3 rng.h

```
/*
   Header file for the random number generator
*/
#ifndef RNG_H
#define RNG_H

#include <vector>

class rng {
   public:
      static double runif();
      static double rnorm();
      static std::vector<double> rnorm(int n);
};
#endif
```

B.4 rng.cpp

```
#include "rng.h"
#include <stdlib.h>   /* srand, rand */
#include <cmath>

// constants for rnorm
const double a0 = 2.50662823884;
const double a1 = -18.61500062529;
const double a2 = 41.39119773534;
const double a3 = -25.44106049637;
const double a[] = {a0,a1,a2,a3};
const double b0 = -8.47351093090;
const double b1 = 23.08336743743;
const double b2 = -21.06224101826;
const double b3 = 3.13082909833;
const double b[] = {b0,b1,b2,b3};
const double c0 = 0.3374754822726147;
const double c1 = 0.9761690190917186;
const double c2 = 0.1607979714918209;
const double c3 = 0.0276438810333863;
const double c4 = 0.0038405729373609;
const double c5 = 0.0003951896511919;
const double c6 = 0.0000321767881768;
const double c7 = 0.0000002888167364;
const double c8 = 0.0000003960315187;
const double c[] = {c0,c1,c2,c3,c4,c5,c6,c7,c8};

double rng::runif() {
   return ((double) rand()) / RAND_MAX;
}

double rng::rnorm() {
   double v = runif();
   double u;
   double z = 0;
   double A = 0;
   double B = 0;

   u = v;
   if (v < 0.5) {
        u = 1 - v;
    }

    if (u <= 0.92 & u >= 0.5) {
        for (int i = 0; i <= 3; i++) {
            A += a[i] * pow(u - 0.5, 2 * i + 1);
            B += b[i] * pow(u - 0.5, 2 * i + 2);
        }
        z = A / (1 + B);
    }
    else {
        for (int i = 0; i <= 8; i++) {
            z += c[i] * pow(log( -log(1 - u) ), i);
        }
    }

    if (v < 0.5) return -z;
    else return z;
}
std::vector<double> rng::rnorm(int n) {
    std::vector<double> z(n);
    for (int i = 0; i < n; i++) {
        z.at(i) = rnorm();
    }
    return z;
}
```