Take Home Exam 4 — Computational Finance

Mezzetti Anita 307498

① $I(\tau) = \int_0^\tau S_\eta \, d\eta$ ⇒ I depends on S

$\Psi(S_T, I_T) = \left(S_T - \frac{I_T}{T}\right)^+$ where $dS_\tau = r S_\tau d\tau + \sigma S_\tau dW_\tau$, $dI_\tau = S_\tau d\tau$

$$u(S,I,\tau) = e^{-\int_t^T r d\eta} \, \mathbb{E}[\Psi(S_T, I_T) \mid S_\tau = S, I_\tau = I]$$

$U := U(S,I,t)$

$$\begin{cases} \partial_t U - \frac{1}{2}\sigma^2 S^2 \partial_{SS} U - rS\partial_S U - S\partial_I U + rU \overset{(*)}{=} 0 & \forall S, I \in \mathbb{R}^+ \quad t \in [0,T] \\ U(S,I,0) = \max\left(S - \frac{I}{T}, 0\right) \end{cases}$$

$R_t = \frac{I_t}{S_t}$

Show that the solution $U := U(S,I,t)$ of * can be rewritten as

$$U(S,I,t) = S \cdot H(R,t)$$

where $H(R,t)$ satisfies: $H := H(R,t)$

$$\begin{cases} \partial_t H - \frac{\sigma^2 R^2}{2} \partial_{RR} H - (1 - rR)\partial_R H \overset{(*)}{=} 0 & R \in \mathbb{R}^+ \quad t \in [0,T] \\ H(R,0) = \max\left(1 - \frac{R}{T}, 0\right) \end{cases}$$

Solution:

- Let us start from the initial conditions:

$$U(S,I,0) = \max\left(S - \frac{I}{T}, 0\right)$$

Remind that $R = \frac{I}{S}$

$$H(R,0) = \frac{U(S,I,0)}{S} = \max\left(0 - \frac{I}{S \cdot T}, 0\right) = \max\left(0 - \frac{R}{T}, 0\right)$$ ok!

- We need to rewrite $\partial_t U$, $\partial_S U$, $\partial_{SS} U$ and $\partial_I U$:

$U := U(S,I,t)$, $H := H(R,t)$ ⇒ $U = SH$

Keep in mind that $\partial_S R = -\frac{I}{S^2}$

- $\partial_t U = \partial_t (S \cdot H) = S \partial_t H$
- $\partial_I U = \partial_I (S \cdot H) = S \partial_I H = \partial_R H$ ← $\partial_I H = \partial_R H \frac{dR}{dI} = \frac{1}{S}\partial_R H$
- $\partial_S U = \partial_S (S \cdot H) = H + \partial_S H = H + S \partial_R H \frac{dR}{dS} = H - \frac{I}{S}\partial_R H = H - R\partial_R H$
- $\partial_{SS} U = \partial_S (H - R\partial_R H) = \underbrace{\partial_S H}_{\partial_R H\left(-\frac{I}{S^2}\right)} - \underbrace{\partial_S R}_{-\frac{I}{S^2}} \partial_R H - R \underbrace{\partial_{SR} H}_{\partial_{RR} H\left(-\frac{I}{S^2}\right)} = -\cancel{\frac{I}{S^2}\partial_R H} + \cancel{\frac{I}{S^2}\partial_R H} + \frac{R^2}{S}\partial_{RR} H$

Therefore $\partial_t U = S\,\partial_t H$

$\partial_I U = \partial_R H$

$\partial_S U = H - R\,\partial_R H$

$\partial_{SS} U = \frac{R^2}{S}\,\partial_{RR} H$

Substituting in $*$:

$$S\,\partial_t H - \frac{1}{2}\sigma^2 S^2 \frac{R^2}{S}\,\partial_{RR} H - rS(H - R\,\partial_R H) - S\,\partial_R H + r \cdot S \cdot H = 0$$

Dividing by S $(S \in \mathbb{R}^+)$:

$$\partial_t H - \frac{1}{2}\sigma^2 R^2\,\partial_{RR} H - rH + rR\,\partial_R H - \partial_R H + rH = 0$$

$$\partial_t H - \frac{1}{2}\sigma^2 R^2\,\partial_{RR} H + (rR - 1)\,\partial_R H = 0$$

which is equal to $*$

$\Rightarrow$ we have proven that the solution of U can be written as $U = S \cdot H$ if H solves this equation.

In the previous point we also took care of the initial condition.

(2) $V := V(S,t)$ $\quad S_{min} = 0$ $\quad$ bounded domain: $[0, S_{max}) \times [0, T)$

$$(*) \quad \begin{cases} \partial_t V - \frac{\sigma^2}{2} S^2 \partial_{SS} V - rS\,\partial_S V + rV = 0 \\ V(S_{max}, t) = 0 \\ V(S,0) = G(S) = \max\{K - S, 0\} \end{cases}$$

a) In the BS model $\sigma(S,t) = \sigma$ and $r(t) = r$.

$$\frac{\partial}{\partial \sigma}\left(\begin{cases} \partial_t V - \frac{\sigma^2}{2} S^2 \partial_{SS} V - rS\,\partial_S V + rV = 0 \\ V(S_{max}, t) = 0 \quad V(S,0) = G(S) = \max(K - S, 0) \end{cases} \right)$$

$$\overset{\upsilon = \partial_\sigma V}{\Rightarrow} \begin{cases} \partial_t \upsilon - S^2 \frac{\partial}{\partial \sigma}\left(\frac{\sigma^2}{2} \partial_{SS} V\right) - rS\,\partial_S \upsilon + r\upsilon = 0 \\ \upsilon(S_{max}, t) = 0 \qquad \upsilon(S,0) = 0 \end{cases}$$

$$\begin{cases} \partial_t \upsilon - S^2 \frac{\sigma^2}{2} \partial_{SS} \upsilon - rS\,\partial_S \upsilon + r\upsilon \overset{*}{=} S^2 \sigma\, \partial_{SS} V \\ \upsilon(S_{max}, t) = 0 \qquad \upsilon(S,0) = 0 \end{cases} \quad \leftarrow \text{PDE for } \upsilon$$

The PDE for the vega $*$ is similar to the PDE for V $*$.
The additional term derives from the application of the product rule.

To compute vega: 1) first compute V (FD scheme)
2) then compute υ (FD scheme) using the computed V for the right hand side.

In other words, we must numerically solve the PDE $*$ along with the BS PDE $*$ on the same grid and in the same way.

To approximate $\partial_{SS} V$, we can use the second derivative of an option price can be approximated with

$$D_h^2 V(S) = \frac{V(S+h) - 2V(S) + V(S-h)}{h^2} \quad (*)$$

We obtain a non homogeneous PDE, similar to $*$ apart from the RHS.
Therefore, it is a non-homogeneous second order linear PDE.

b) After setting the given parameters, we calculate S_max using the given max error of truncation. We use the same formula of exercise 4c in problem set 7:

$$S\_max = K \cdot \exp\left\{ \frac{1}{2} \cdot \sigma^2 \cdot T - \sigma\sqrt{T} \cdot \text{norminv}\left(\frac{max\_tol}{K} \right) \right\}$$

norminv(p) returns the inverse of the standard cumulative distribution function, evaluated at the probability values in p.

The goal is to numerically solve the equation for υ:

i) Compute the numerical solution for problem * using the θ-method (θ = 0,5):

The θ-method is a linear combination of the forward and backward Euler methods:

$$U^{m+1} = U^m + \theta \Delta t \left(F^{m+1} - A^{m+1} U^{m+1}\right) + (1-\theta)\,\Delta t \left(F^m - A^m U^m\right)$$

$\theta = \frac{1}{2} \Rightarrow$ Crank-Nicolson method (centered finite difference in time)

We use the previously implemented bs_timestepping function, which needs 3 functions as input. Therefore, we create and pass them:

1) forcing: describe the right hand side - Zero for *
2) bc_right: describe the boundary conditions of the right border, which is zero for os.
3) initial_cond: describe the initial conditions. In * we find that the initial condition is $\max\{K-S, 0\}$

The function bs_timestepping returns a matrix 61 × 101 with the option prices and two vectors corresponding to the two axis of the grid:

- FD_grid: Contains the nodes at which the solution U is evaluated
- time_step: time steps at which U is computed.

ii) Use the numerical result of i) to approximate by finite differences the right hand side of the PDE solved by V. Numerically solve the PDE for V obtaining an approximate solution V_a.

vega_timestepping is a θ-method time-stepping differences solver for vega. Also in this case, before calling this function, we have to create its still missing arguments:

- vega_rhs: the right hand side (rhs) of vega can be found using find_rhs ↘

find_rhs is a function that returns the rhs of the vega PDE *. Looking at point a), the rhs is given by $S^2 \sigma \partial_{ss} V$

$\sigma \rightarrow$ sigma
$S \rightarrow$ FD_grid
$\partial_{ss} V \rightarrow$ approx_second_der
(obtained using ⊛ defined in part a))

- vega_bc_right
- vega_init_condition

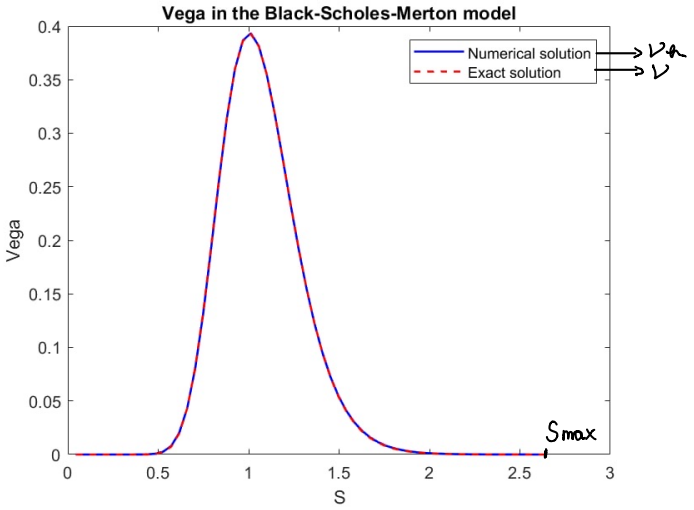
} they are defined as explained before. Also for vega, they both are zero vectors.

The result V_a vega_approx is an approximation of the analytical solution V,

iii) The numerical solution V is obtained used besvega, which computes the BS sensitivity to underlying price volatility.
Therefore, we have:

vega_timestepping ⇒ V_a (vega_approx)
besvega ⇒ V (vega_exact)

We plot both for all the Si values that belong to FD_grid(2:end):



c) Verify the convergence property of the finite difference solver:
Solve numerically the eq. for V on a sequence of increasingly finer computational grids

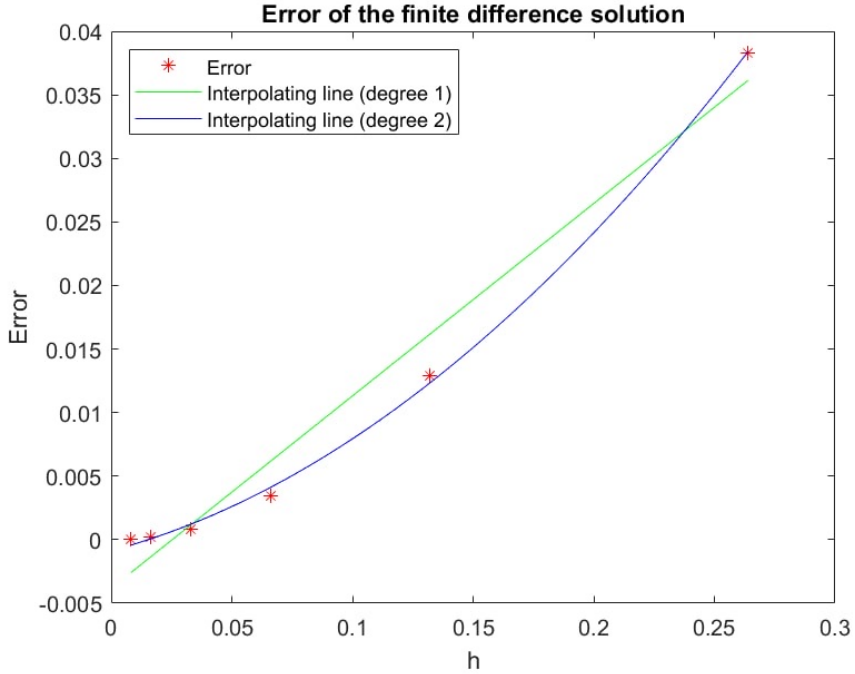
N_x : number of intervals in space $\quad N_x = 10 \times 2^i \quad i = 0, 1, 2, \ldots 5$
N_t : number of time-steps $\quad N_t = 0.6 \cdot N_x$

and, for each discretization level compute the error of the finite difference solution:

$$e(h,T) = \max_i |V_h(S_i,T) - V(S_i,T)| \qquad h = \frac{S_{max}}{N_x}$$

To find the error for each level of discretization, we need a loop. Each iteration corresponds to a different i and, consequently, to a different grid. Therefore, for each loop we find a different error.
Inside the loop, we repeat the same steps described above to calculate vega_approx and vega_exact.

The results are 6 h values to which correspond 6 errors. We plot these points as *. We also add to the plot two interpolation lines, created using polyfit and polyval, one of degree 1 and one of degree 2:



As expected, as high h corresponds to a less fine grid and the corresponding error is quite large.
On the other side, when $h \to 0$ (more fine grade), the error tends to zero as well.

Order of convergence:

def: A sequence $x^{(k)} \in \mathbb{R}^m$, $k \in \mathbb{N}_0$, converges with order $p \in [1, \infty)$ to x^* if $x^* = \lim_{k \to \infty} x^{(k)}$ and

$$\exists C \in [0, \infty) \;/\; \forall k \in \mathbb{N}_0 : \|x^{(k+1)} - x^*\| \leq C \|x^{(k)} - x^*\|^p$$

Looking at the previous plot, we observe a nice second-order convergence behaviour.

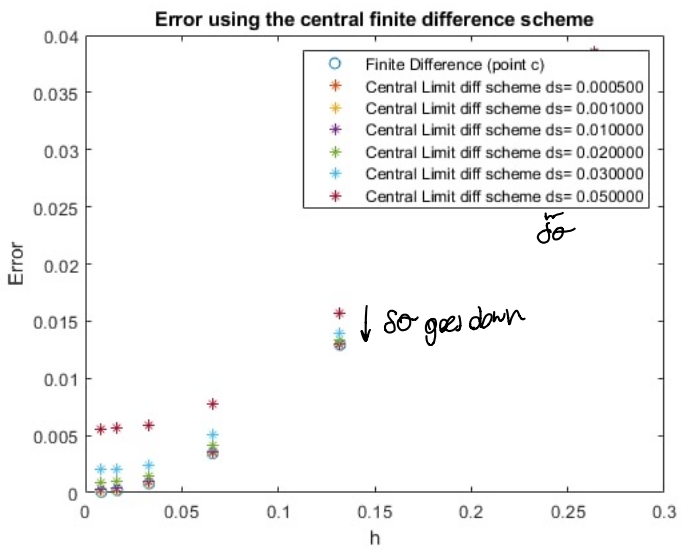
a) i) Use the central limit difference scheme to directly approximate ν as

$$\nu_{\delta\sigma} \overset{*}{=} \frac{V(S,t,\sigma+\delta\sigma) - V(S,t,\sigma-\delta\sigma)}{2\delta\sigma}$$

where V is the numerical solution of $*$.
Compute $\nu_{\delta\sigma}$ for the given $\delta\sigma$.

In order to solve this point we need to calculate the price V two times, one for $V(S,t,\sigma+\delta\sigma)$ and one for $V(S,t,\sigma-\delta\sigma)$. We call them `V_plus` and `V_minus`.
As usual, we use the function `bs_timestepping` to which we pass sigma+$\delta\sigma$ and sigma-$\delta\sigma$ respectively. We thus find $\nu_{\delta\sigma}$, called `vega_sigma`, by $*$.
Of course we repeat these steps for all the given $\delta\sigma$.

ii) To make things easier, we do part i) and ii) in the same loop. Therefore, once we have found `vega_sigma`, we directly proceed to find `vega_exact` using `blsvega`.
Then, we find the error as in c).
We plot $e_{\delta\sigma}(h,T)$ for each h and for each $\delta\sigma$.



We note that, for all values of δo, the error goes down decreasing h (finer grid). This was easily predictable. It was expected also that we obtain a smaller error when δo decreases.
These two results are the basis of the FD method.

What is more interesting is that, for any δo the error is bigger than the one given by point c).
The plot show that, under the same h, point d) errors always exceed point c error.
Therefore, c) seems more accurate from the previous plot.
Of course, decreasing δo, the difference between the errors of the two methods drops:

$$\text{if } \delta o(i) \downarrow \Rightarrow e_{\delta o(i)} - e_{\text{point c}} \downarrow$$

In terms of order of convergence, we can conclude that, for sufficiently small δo, the two manifest the same order of convergence.

One may ask if there exists a proper value of δo (a kind of threshold) under which $e_{\delta o} < e_{\text{point c}}$.
We try to answer to this question repeating the previous steps for extremely low values of δo and zooming the resulting plot for a certain h.

We try with $\delta o = [10^{-5}\ \ 10^{-6}\ \ 10^{-7}]$ and we obtain that the threshold exists and it is a value between 10^{-6} and 10^{-4}. However, we are working with extremely low numbers and we should further check not to run into any machine error.