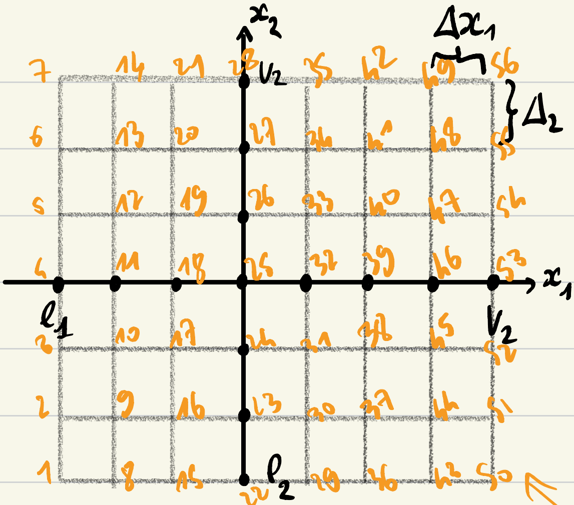
What we have to deal with is the solution of the following PDE :

$$\begin{cases} \dfrac{\partial V}{\partial t} + rS_1 \dfrac{\partial V}{\partial S_1} + rS_2 \dfrac{\partial V}{\partial S_2} + \dfrac{1}{2}\sigma_1^2 S_1^2 \dfrac{\partial^2 V}{\partial S_1^2} + \dfrac{1}{2}\sigma_2^2 S_2^2 \dfrac{\partial^2 V}{\partial S_2^2} + \rho S_1 S_2 \sigma_1 \sigma_2 \dfrac{\partial^2 V}{\partial S_1 \partial S_2} \\ \quad - rV = 0 \qquad \forall t \in [0,T], \ \forall s_1 \in (L_1, U_1), \ \forall S_2 \in (L_2, U_2); \\ \\ V(t, L_1, S_2) = V(t, U_1, S_2) = V(t, S_1, L_2) = V(t, S_1, U_2) = 0 \ \forall t \in [0,T]; \\ V(T, S_1, S_2) = (w_1 S_1 + w_2 S_2 - K)^+ \mathbb{1}_{\{L_1 \le S_1(t) \le U_1 \ \forall t \in [0,T]\}} \mathbb{1}_{\{L_2 \le S_2(t) \le U_2 \ \forall t \in [0,T]\}} \end{cases}$$

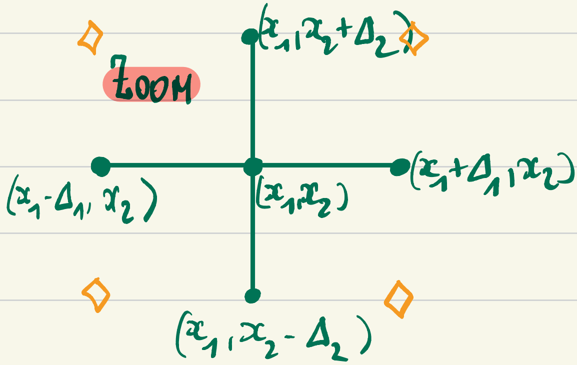
We can also decide to work on the log-prices : $x_1 = \log(S_1)$, $x_2 = \log(S_2)$, and $V(t, x_1, x_2) = V(t, e^{x_1}, e^{x_2})$. The log-price PDE is :

$$\frac{\partial V}{\partial t} + \left(r - \frac{\sigma_1^2}{2}\right)\frac{\partial V}{\partial x_1} + \left(r - \frac{\sigma_2^2}{2}\right)\frac{\partial V}{\partial x_2} + \frac{\sigma_1^2}{2}\frac{\partial^2 V}{\partial x_1^2} + \frac{\sigma_2^2}{2}\frac{\partial^2 V}{\partial x_2^2} + \rho \sigma_1 \sigma_2 \frac{\partial^2 V}{\partial x_1 \partial x_2} - rV = 0$$

We want to solve those PDEs numerically.



$l_1 = \log(L_1)$, $l_2 = \log(L_2)$.



$$x_{1,j} = l_1 + j\Delta_1$$
$$x_{2,i} = l_2 + i\Delta_2$$

The best way to store it in Matlab.

Notice that we have:

$$\frac{\partial V}{\partial x_1}(x_1,x_2) \simeq \frac{V(x_1+\Delta_1, x_2) - V(x_1-\Delta_1, x_2)}{2\Delta_1} + o(\Delta_1^2)$$ EXACTLY AS IN 1D.

$$\frac{\partial V}{\partial x_2}(x_1,x_2) \simeq \frac{V(x_1, x_2+\Delta_2) - V(x_1, x_2-\Delta_2)}{2\Delta_2} + o(\Delta_2^2)$$ EXACTLY AS IN 1D.

$$\frac{\partial^2 V}{\partial x_1^2}(x_1,x_2) \simeq \frac{V(x_1+\Delta_1, x_2) - 2V(x_1,x_2) + V(x_1-\Delta_1, x_2)}{\Delta_1^2} + o(\Delta x_1^2)$$ Like in 1D.

$$\frac{\partial^2 V}{\partial x_2^2}(x_1,x_2) \simeq \frac{V(x_1, x_2+\Delta_2) - 2V(x_1,x_2) + V(x_1, x_2-\Delta_2)}{\Delta_2^2} + o(\Delta x_2^2)$$ Like in 1D.

Now, we need to approximate:

$$\frac{\partial^2 V}{\partial x_1 \partial x_2}(x_1, x_2) = \underbrace{\frac{\partial}{\partial x_1}\left(\frac{\partial V}{\partial x_2}(x_1, x_2)\right)}_{\text{OR, in the reverse order.}}$$

Use the previous expression.

$$\cong \frac{\partial}{\partial x_1}\left(\frac{V(x_1, x_2+\Delta_2) - V(x_1, x_2-\Delta_2)}{2\Delta_2}\right)$$

Linearity

$$\cong \frac{1}{2\Delta_2}\left(\frac{\partial}{\partial x_1}V(x_1, x_2+\Delta_2) - \frac{\partial}{\partial x_1}V(x_1, x_2-\Delta_2)\right)$$

We can compute using ◇ on the Zoom

$$\cong \frac{1}{2\Delta_2}\left[\frac{V(x_1+\Delta_1, x_2+\Delta_2) - V(x_1-\Delta_1, x_2+\Delta_2)}{2\Delta_1} - \frac{V(x_1+\Delta_1, x_2-\Delta_2) - V(x_1-\Delta_1, x_2-\Delta_2)}{2\Delta_1}\right]$$

$$\cong \frac{1}{4\Delta_1\Delta_2}\begin{bmatrix} V(x_1+\Delta_1, x_2+\Delta_2) + V(x_1-\Delta_1, x_2-\Delta_2) \\ - V(x_1-\Delta_1, x_2+\Delta_2) - V(x_1+\Delta_1, x_2-\Delta_2) \end{bmatrix} + O(\Delta_1^2+\Delta_2^2)$$

So in MATLAB :

$$\frac{\partial V_i}{\partial x_2} = \frac{V_{i+1} - V_{i-1}}{2\Delta_2} \quad \text{with} \quad \Delta_2 = \frac{U_2 - L_2}{(M_2 - 1)}$$

← number of pts on the x_2 axis.

Analogous for the x_1 axis.

Let's look at the code :
of "Basket_KnockOut.m"

↳ by default, $W_1 = W_2 = 1$.

↳ Usually : calibrate S_1 & S_2 on call options on S_1 & S_2 <u>separately</u> ; and calibrate ρ by <u>historical analysis</u> (like correlation between S_1-logreturns & S_2-logreturns).

↳ To build a grid :

$$x_1 = \text{linspace}(x_1\text{min}, x_1\text{max}, M_1)$$
$$x_2 = \text{linspace}(x_2\text{min}, x_2\text{max}, M_2)$$
$$[X_1, X_2] = \text{meshgrid}(x_1, x_2)$$

↳ This gives us 2 matrices : coordinates of X_1, and of X_2.

↳ Then we have to transform the grid (2 matrices) into a vector :

X = [X1(:) X2(:)]

transforms a matrix to a vector.

In this form :
$$\begin{bmatrix} 0 & 10 \\ 0 & 12.25 \\ 0 & 14.5 \\ \cdots & \cdots \end{bmatrix}$$

↳ We have to be able to know WHEN we are on the boundary of the domain: we have 4 edges: N, S, W, E. } This about how W, S, N, E are coded (using the scheme we wrote above) to check it's clear in you head.

↳ use "unique".

↳ Then the matrix M is built according to the equation (and its discretization).

↳ Then we use for ex I.E: backward in time...

↳ we get the price and price surface.

Idea behind this code:

$$\frac{\partial V}{\partial t} + \left(r - \frac{\sigma_1^2}{2}\right)\frac{\partial V}{\partial x_1} + \left(r - \frac{\sigma_2^2}{2}\right)\frac{\partial V}{\partial x_2} + \frac{\sigma_1^2}{2}\frac{\partial^2 V}{\partial x_1^2} + \frac{\sigma_2^2}{2}\frac{\partial^2 V}{\partial x_2^2} + \rho\sigma_1\sigma_2\frac{\partial^2 V}{\partial x_1 \partial x_2} - rV = 0$$

$V_{j,i} = V(t_j, x_1(i), x_2(i))$: because we use a unique index in order to define the points of our grid. The idea is that:

I.E

$\downarrow$

j+1: known

j: unknown

$$\frac{V_{j+1,i}\boxed{-}V_{j,i}}{\boxed{\Delta t}} + \left(r-\frac{\sigma_1^2}{2}\right)\frac{V_{j,i+M_2}-V_{j,i-M_2}}{2\Delta_1}$$

$$+\left(r-\frac{\sigma_2^2}{2}\right)\frac{V_{j,i+1}-V_{j,i-1}}{2\Delta_2}$$

$$+\frac{\sigma_1^2}{2}\,\frac{V_{j,i+M_2}\boxed{-2}V_{j,i}+V_{j,i-M_2}}{\boxed{\Delta_1^2}}$$

$$+\frac{\sigma_2^2}{2}\,\frac{V_{j,i+1}\boxed{-2}V_{j,i}+V_{j,i-1}}{\boxed{\Delta_2^2}}$$

$$+\rho\sigma_1\sigma_2\,\frac{V_{j,i+M_2+1}-V_{j,i+M_2-1}-V_{j,i-M_2+1}+V_{j,i-M_2-1}}{4\Delta_1\Delta_2}$$

$$\boxed{-r}V_{j,i}=0$$

see the MATLAB code

all the coefficients associated to each term.

The MATLAB code that the professor provided us is exactly the implementation of this equation.

⚠ The 2D PDE code part is not asked at the exam.

END OF THE PART ABOUT PDEs.