Pricing of an American option using Black and Scholes model under Gamma constraint (project #4)

Eldiias **Dzhamankulov**, Aidar **Mussabekov**

December 18, 2017
Module : FQ301 (PDE in finance) *

Contents

*Encadrant : O. Bokanowski

1 Introduction

1.1 Main Problem

Consider a portfolio consisting of a risk-free asset S^0 and a risk asset $S(u) = S_{t,s}(u)$, evolving according to $dS^0(u) = S^0(u)rdu$ and

$$dS(u) = S(u)(\mu du + \sigma(u, S(u))dW(u))$$

Let Y(u) be the risky part of the asset at time u. In the Black and Scholes model, the classical hedging strategy consists in taking $Y(u) = \frac{\partial v}{\partial s}(u, S(u))$. In practice market constraints mean that this optimal strategy is not always possible, and we examine here a model where one imposes a constraint on the variations of Y. More precisely, we give ourselves a constant $\Gamma > 0$ and we consider constraint

$$s\frac{\partial^2 v}{\partial s^2} \leq \Gamma$$

(One can say that v is Γ-concave).

It's shown [14] that a model for the price of the option v with Gamma constraints is the solution of the following PDE:

$$min(-\frac{\partial v}{\partial t} - \frac{\sigma^2}{2}s^2\frac{\partial^2 v}{\partial s^2} - rs\frac{\partial v}{\partial s} + rv) = 0 \tag{1}$$

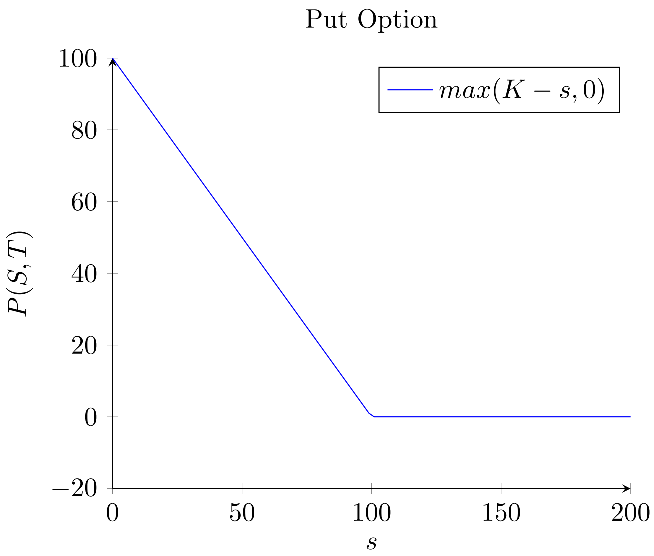
with the terminal condition

$$v(T, s) = \hat{g}(s). \tag{2}$$

The function $\hat{g}(s)$ which is increasing function of g(s) The function $\hat{g}(s)$ is itself denoted as the smallest Γ-concave function increasing in g(s), and one can show that this function is solution of the equation

$$min(\hat{g}(s) - g(s), \Gamma - s\frac{\partial^2 \hat{g}}{\partial s^2}) = 0, s > 0. \tag{3}$$

1.2 American Put Option



An American option is an option that can be exercised anytime during its life - at any time prior to and including its maturity date[1], thus increasing the value of the option to the holder relative to European options, which can only be exercised at maturity date. The holder of an American Put option has the right (but he is not obliged) to exercise the option at any time until its expiration date. In case he exercises it, he sells previously defined amount of assets to option seller at strike price.

American Put option P(S,t) is required to satisfy following conditions:

$$\begin{cases} P(\inf, t) = 0 \\ P(S,T) = \max(K-S,0) \\ P(S,t) \geq \max(K-S,0) \end{cases}$$

[1] According to https://www.investopedia.com/terms/a/americanoption.asp

2 Development of the model

2.1 Black and Scholes model transformation

We look for a numerical approximation of the American Put function $v = v(t,s), t \in (0,T), s \in [0, Smax]$.It satisfies in first approximation the Black and Scholes backward PDE on the truncated domain $\Omega = [S_{min}, S_{max}]$:

$$\begin{cases} -\frac{\partial v}{\partial t} - \frac{\sigma^2}{2} s^2 \frac{\partial^2 v}{\partial t^2} - rs\frac{\partial v}{\partial s} + rv = 0, t \in (0,T), s \in (S_{min}, S_{max}) \\ v(t, S_{min}) = v_\ell(t) \equiv K - S_{min}, t \in (0,T) \\ v(t, S_{max}) = v_r(t) \equiv 0, t \in (0,T) \\ v(t,s) = \varphi(s) := (K-s)_+, s \in (S_{min}, S_{max}) \end{cases}$$

With terminal condition $v(T,s) = q$ and defining $\hat{v}(t,s) = v(T-t,s)$ we end up with:

$$\begin{cases} \frac{\partial \hat{v}}{\partial t} + \frac{\sigma^2}{2} s^2 \frac{\partial^2 \hat{v}}{\partial t^2} - rs\frac{\partial \hat{v}}{\partial s} + r\hat{v} = 0, t \in (0,T), s \in (S_{min}, S_{max}) \\ \hat{v}(t, S_{min}) = \hat{v}_\ell(t) \equiv K - S_{min}, t \in (0,T) \\ \hat{v}(t, S_{max}) = \hat{v}_r(t) \equiv 0, t \in (0,T) \\ \hat{v}(t,s) = \varphi(s) := (K-s)_+, s \in (S_{min}, S_{max}) \\ \hat{v}(0,s) = q \end{cases}$$

For simplicity of notes let's denote $\hat{v}$ as v. We first introduce a discrete mesh as follows. Let $h := \frac{S_{max}-S_{min}}{I+1}$ be (spatial) mesh step, and $\Delta t := \frac{T}{N}$ be the time step. Then $s_j = S_{min} + jh, j = 0, ..., I+1$ are the mesh points, and $t_n = n\Delta t, n = 0, ..., N$ the time mesh.

We are looking for U_n^j, an approximation of $v(t_n; s_j)$.

For any function $v \in C^2$ (or $v \in C^3$ for (4)), we recall the following approximations, as $h \to 0$

$$v'(s_j) = \frac{v(s_j) - v(s_{j-1})}{h} + O(h) \tag{4}$$

$$v'(s_j) = \frac{v(s_{j+1}) - v(s_j)}{h} + O(h) \tag{5}$$

$$v'(s_j) = \frac{v(s_{j+1}) - v(s_{j-1})}{2h} + O(h^2) \tag{6}$$

We therefore obtain several possible approximations by finite differences for the first order derivative :

$$\partial_s v(t_n, s_j) \simeq \frac{U_j^n - U_{j-1}^n}{h} \text{ (backward difference approximation)} \tag{7}$$

$$\partial_s v(t_n, s_j) \simeq \frac{U_{j+1}^n - U_j^n}{h} \text{ (forward difference approximation)} \tag{8}$$

$$\partial_s v(t_n, s_j) \simeq \frac{U_{j+1}^n - U_{j-1}^n}{2h} \text{ (center difference)} \tag{9}$$

The first two approximations are said to be consistent of order 1 (in space), while the second one is consistent of order 2.

We also recall the approximation

$$-\partial_{ss}^2(t_n, s_j) \simeq \frac{-U_{j-1}^n + 2U_j^n - U_{j+1}^n}{h^2}, \qquad (10)$$

which is consistent of order 2 in space.

Hence we obtain the so-called "Euler Backward scheme" (or Implicit Euler scheme), abbreviated "EI" using the centered approximation, as follows :

Below this line a boxed environment is used

$$\begin{array}{ll}
\frac{U_j^{n+1}-U_j^n}{\Delta t} + \frac{\sigma^2}{2}s_j^2\frac{-U_{j-1}^{n+1}+2U_j^{n+1}-U_{j+1}^{n+1}}{h^2} - rs_j\frac{U_{j+1}^{n+1}-U_{j-1}^{n+1}}{2h} + rU_j^{n+1} = 0, & n = 0, ..., N-1, j = 1, ..., I \\
U_0^{n+1} = v_\ell(t_{n+1}) = K - S_{min}, & n = 0, ..., N \\
U_{I+1}^{n+1} = v_r(t_{n+1}) = q, & n = 0, ..., N \\
U_j^0 = \varphi(s_j) = (K - s_j)_+, & j = 1, ..., I
\end{array} \qquad (11)$$

Let us remark that we have taken $j = 1$ and $j = I$ as extreme indices in j. For $j = 1$, the scheme utilizes the known value $U_0^{n+1} = v_l(t_n)$ (left boundary value). For $j = I$, the scheme utilizes the known value $U_{I+1}^{n+1} = v_r(t_n)$ (right boundary value).

2.2 Gamma constraint transformation

The most optimal way to solve given equations (1) and (3) is Newton's method which is numerical method. To solve equations using Newton's method, we need to write them in vector form. For Newton's method we need transform equations to the form:

$$\min(KX - B, LX - C) = 0 \qquad (12)$$

Equation (3) can be transformed in the next way:

$$\min(\hat{g}(s) - g(s), \Gamma - s\frac{\partial^2 v}{\partial^2}) = 0$$

For this equation we can denote ĝ as X. Then we have:

$$K = Id = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix}$$

and $B = U_j^n$. Now let's take a look on a $\Gamma - constraint$:

$$\Gamma - s\frac{\partial^2 v}{\partial^2} = \Gamma + s(\frac{-U_{j-1}^{n+1} + 2U_j^{n+1} - U_{j+1}^{u+1}}{h^2})$$

From here we have:

$$L = \begin{bmatrix} \frac{2s}{h^2} & \frac{-s}{h^2} & \cdots & 0 & 0 \\ \frac{-s}{h^2} & \frac{2s}{h^2} & \frac{-s}{h^2} & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & \frac{-s}{h^2} & \frac{2s}{h^2} & \frac{-s}{h^2} \end{bmatrix}$$

$$C = \begin{bmatrix} \frac{s}{h^2}U_0 - \Gamma \\ -\Gamma \\ \vdots \\ -\Gamma \\ \frac{s}{h^2}U_{I+1} - \Gamma \end{bmatrix}$$

Hence, we end up with following equation:

$$\min(KX - B, LX - D) = 0. \tag{13}$$

2.3 Programming Euler Backward

We choose to work with the unkown the vector corresponding to $(v(t_n, s_j))_j = 1, ..., I$:

$$U^{n+1} = \begin{pmatrix} U_1^{n+1} \\ \cdots \\ U_I^{n+1} \end{pmatrix}$$

We would like to write (11) under the vector form as follows :

$$\frac{U^{n+1} - U^n}{\Delta t} + AU^{n+1} + q(t_n) = 0, \tag{14}$$

where A is a square matrix of dimension I and q(t) is a column vector of size I.

Let us denote

$$\alpha_j = \frac{\sigma^2}{2}\frac{s_j^2}{h^2}, \beta_j := r\frac{s_j}{2h}$$

We look for A and q(t) such that

$$(-\alpha_i + \beta_i)U_{i-1}^n + (2\alpha_i + r)U_i^n + (-\alpha_i - \beta_i)U_{i+1}^n \equiv (AU + q(t_n))_i$$

By identification we can see that A is a tridiagonal matrix

$$A = \begin{bmatrix} 2\alpha_1 + r & -\alpha_1 - \beta_1 & & & & 0 \\ -\alpha_2 + \beta_2 & 2\alpha_2 + r & -\alpha_2 - \beta_2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & -\alpha_i + \beta_i & 2\alpha_i + r & -\alpha_i - \beta_i & \\ & & & \ddots & \ddots & \ddots \\ 0 & & & & -\alpha_I + \beta_I & 2\alpha_I + r \end{bmatrix}$$

and $q(t)$ will contain the known boundary values U_0 and U_{n+1}:

$$q(t) = \begin{bmatrix} (-\alpha_1 + \beta_1)v\ell(t) \\ 0 \\ \vdots \\ 0 \\ (-\alpha_I - \beta_I)v_r(t) \end{bmatrix}$$

Now we can work with equation (13).

$$\frac{U^{n+1} - U^n}{\Delta t} + AU^{n+1} + q(t_n) = 0,$$

$$U^{n+1} - U^n + \Delta t A U^{n+1} + \Delta t q(t_n) = 0,$$

$$(Id + \Delta t A)U^{n+1} - (U^n - \Delta t q(t_n)), \tag{15}$$

$$U^{n+1} = (Id + \Delta t A)^{-1}(U^n - \Delta t q(t_n)), \tag{16}$$

The equation (15) should be used for Newton's method implementation and as for equation (16) - it solves classical unconstrained Black and Scholes model.

Considering equation (15) we can notice, that it actually looks perfectly combined for Newtons's method: we have everything in vector form and we can define for (12) $K = (Id + \Delta t A), X = U^{n+1}, B = (U^n - \Delta t q(t_n))$ and second part of the comparison is same as in (13).

3 Numerical Results

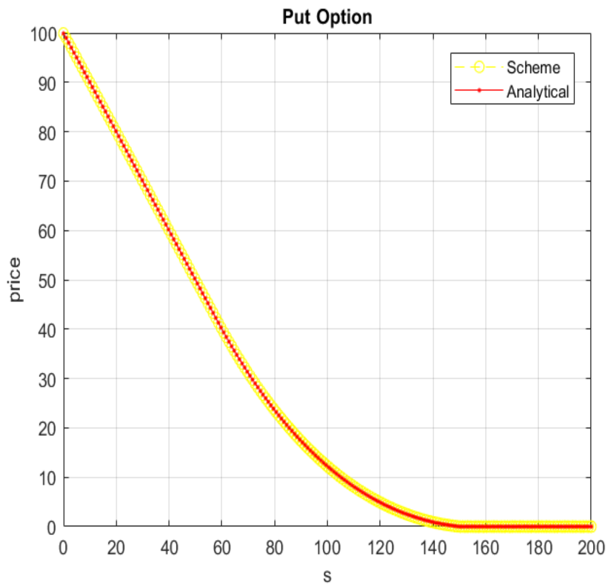
3.1 Conclusions of coding

In the Appendix A the Matlab code solving the problem above can be found. Here only main results will be mentioned.

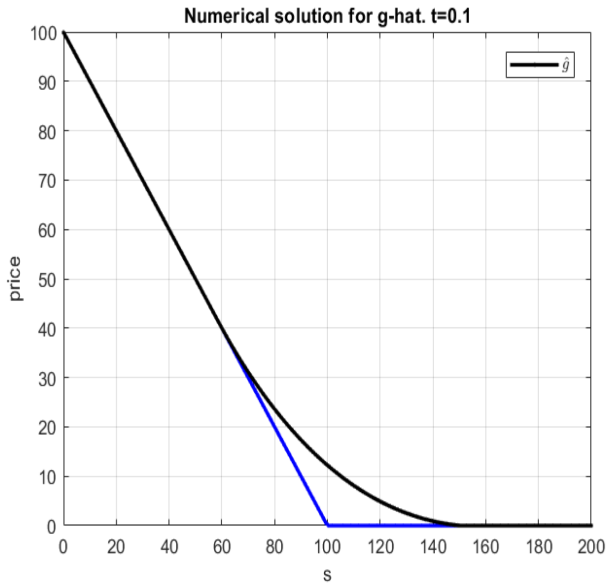
1. Depending on the number of observations, result vary a lot. Decreasing N leads to increase a time step that decreases the quality of approximation (mane function is built incorrectly). But uncontrolled increasing of N leads to an increase in the execution time of the code.

2. The value of $\hat{g}$ changes on a very small value. It can be explained by stability of the gradient of min-function. In this problem during almost all the operations we ended up with gradient equal to Id-matrix. Eventually it didn't changed the value of $\hat{g}$.

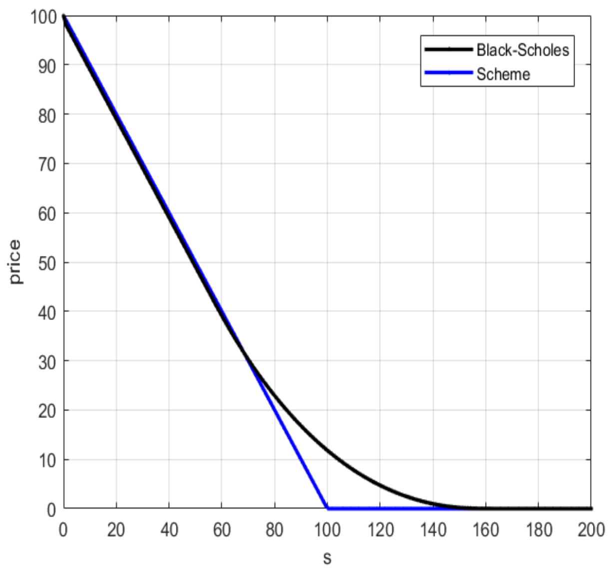
3. The result obtained by Newton's method is almost the same as the one provided by H.M.Sonner and N. Touzi [1]. Functions act similarly enough to prove the quality of numerical approximations.



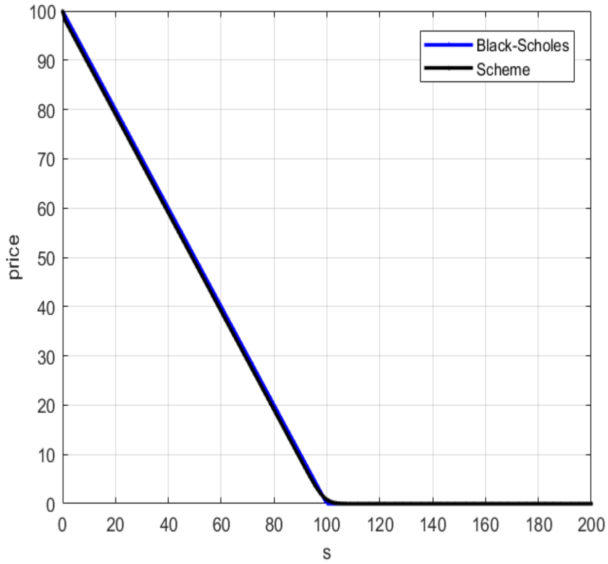
4. Below someone can find the plot of the $\hat{g}$-function at time T for case 1. It is easy to see that the approximation is quite well describes the function. The biggest deviation of a function is less then 11.5.



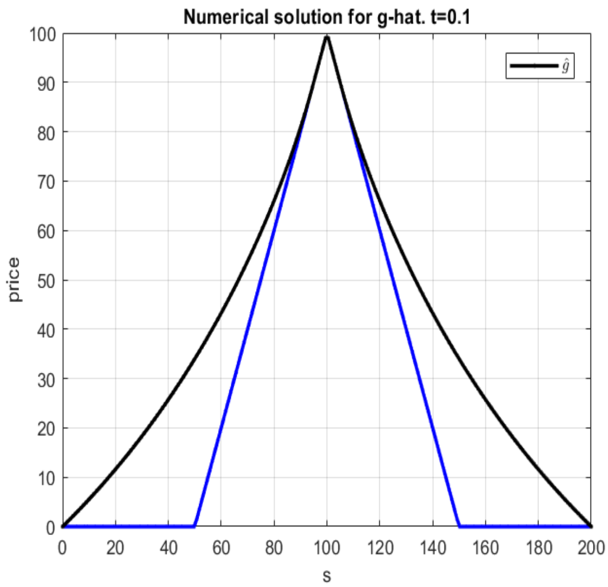
5. On the next figure it is easy to see that the value of the v - the price of the option in first case at the last iteration is also quite close to the values of scheme. But it's also shown that there is a deviation between v and $\hat{v}$.



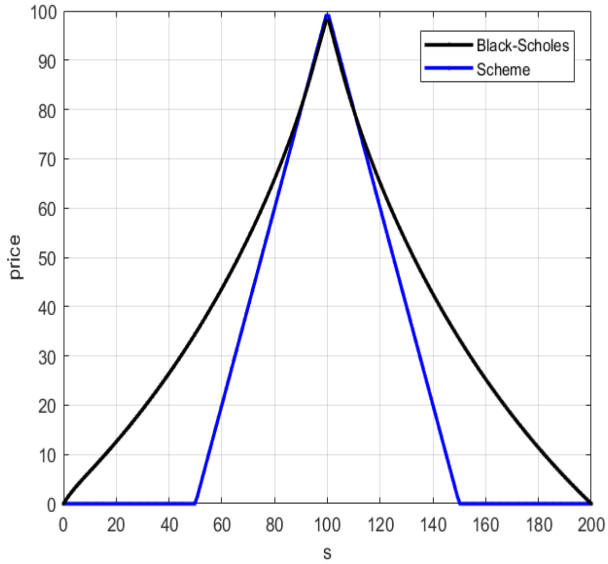
6. But if someone compare the classical Black-Scholes model to Newton's approximation for 1 case, the classical model has a better accuracy.



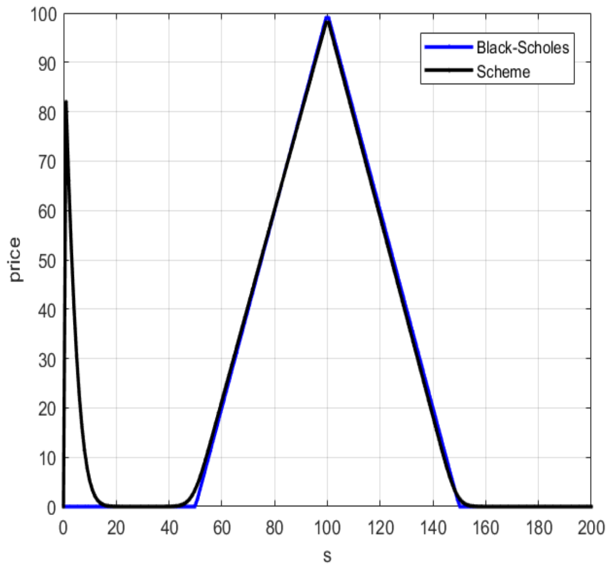
7. For second case we have a bit different situation. The $\hat{g}$-function on the last iteration gives following result:



8. The approximation of v has bigger difference from scheme - less then 100, when $\hat{g}$ had a difference around 245.Even though the figures look the same, the result is more than correct.



9. The classical BS model for second case acts quite strange. It increases right after the start, but then deviation between the model and scheme decreases.



3.2 Black and Scholes model under Gamma constraint and transfers

Here we need to solve the folowwing equation:

$$min(-\frac{\partial v}{\partial t} - \frac{s^2}{2}(\sigma^2 \partial_{s,s} v - \varepsilon|\partial_{s,s} v|) - rs\frac{\partial v}{\partial s} + rv, \Gamma - s\frac{\partial^2 v}{\partial^2}) = 0 \qquad (17)$$

Here we need to transform into vector form the following:

$$-\frac{\partial v}{\partial t} - \frac{s^2}{2}(\sigma^2 \partial_{s,s} v - \varepsilon|\partial_{s,s} v|) - rs\frac{\partial v}{\partial s} + rv$$

Here we have (11) as a part of equation which transformation was shown above. But still we need to transform the part $\varepsilon|\partial_{s,s} v|$. Using same approximation method (via U_j^n) we can obtain the following:

$$-\varepsilon|\partial_{s,s} v| = \frac{\varepsilon}{h^2}\left|-U_{j-1}^{n+1} + 2U_j^{n+1} - U_{j+1}^{u+1}\right|$$

If we define $\eta = \frac{\varepsilon s^2}{2h^2}$ then we have

$$\eta\left|-U_{j-1}^{n+1} + 2U_j^{n+1} - U_{j+1}^{u+1}\right| = \eta(\max(-U_{j-1}^{n+1}+2U_j^{n+1}-U_{j+1}^{u+1},0)-\min(-U_{j-1}^{n+1}+2U_j^{n+1}-U_{j+1}^{u+1},0))$$

Let's define $sign(-U_{j-1}^{n+1} + 2U_j^{n+1} - U_{j+1}^{u+1})$ as sgn_j. Then we can rewrite previous equation as

$$\begin{aligned}\eta(\max(-U_{j-1}^{n+1} + 2U_j^{n+1} - U_{j+1}^{u+1},0) - \min(-U_{j-1}^{n+1} + 2U_j^{n+1} - U_{j+1}^{u+1},0)) =\\ \eta(\max(sgn_j(-U_{j-1}^{n+1}+2U_j^{n+1}-U_{j+1}^{u+1}),0)+\max(sgn_j(-U_{j-1}^{n+1}+2U_j^{n+1}-U_{j+1}^{u+1}),0)) =\\ \eta sgn_j(-U_{j-1}^{n+1} + 2U_j^{n+1} - U_{j+1}^{u+1})\end{aligned}$$

The last equation is easily can be rewritten as $A'U^{n+1} + Q'$:

$$A' = \begin{bmatrix} 2\eta sgn_1 & -\eta sgn_1 & 0 & 0 & \cdots & 0 \\ -\eta sgn_2 & 2\eta sgn_2 & -\eta sgn_2 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & \cdots & -\eta sgn_{I-1} & 2\eta sgn_{I-1} & -\eta sgn_{I-1} \\ 0 & 0 & 0 & \cdots & -\eta sgn_I & 2\eta sgn_I \end{bmatrix}$$

$$Q' = \begin{bmatrix} -\eta sgn_1 U_0^{n+1} \\ 0 \\ \vdots \\ 0 \\ -\eta sgn_I U_{I+1}^{n+1} \end{bmatrix}$$

Now the equation (17) can be easily transformed into vector form:

$$\begin{aligned} min(-\frac{\partial v}{\partial t} - \frac{s^2}{2}(\sigma^2 \partial_{s,s} v - \varepsilon|\partial_{s,s} v|) - rs\frac{\partial v}{\partial s} + rv, \Gamma - s\frac{\partial^2 v}{\partial^2}) =\\ min((Id + \Delta tA - \Delta tA')U^{n+1} - (U^n - \Delta tq(t_n) + \Delta tQ'), \quad (18)\end{aligned}$$

Equation (18) is a generalized version of equation given by Black and Scholes model under Gamma-constraint.

Appendices

A Main code

```
%% Black and Scholes with Gamma constraint
%% Reboot matlab
%%
clc
clear all
tic()
%% Global variables
%%
global K sigma r T G Smin Smax Xmin Xmax Ymin Ymax I N P0
global ur %Boundary conditions
%% Numerical Data
%%
K=100; sigma=0.1; r=0.1; T=0.1; Smin=0; Smax=200; ur=0;
I=200; N=200; ID=eye(I);
G=ones (I,1);
%% Defining the functons
%%
P0=@(s) max(K-s,0); %value at each point
%% Choosing the case
%%
disp('Choose the case');
disp('1:max(K-s,0)');
disp('2:max(K-2|s-K|,0');
choice=input('Your choice: ');
%% Graphics
%%
Xmin=Smin; Xmax=Smax; Ymin=0;Ymax=K;
err_scale=0; %Error graph scale
deltan=N/10;
%% Info for user
%%
fprintf('sigma=%5.2f, r=%5.2f, Smax=%5.2f\n',sigma,r,Smax);
fprintf('Mesh I= %5i, N=%5i\n',I,N);
fprintf('CENTRAGE : CENTRE');
fprintf('SCHEMA: Euler Implicit for American Style Option');
%% Defining the mesh
%%
dt=T/N; %time step
h=(Smax-Smin)/(I+1); %mesh step
s=Smin+(1:I)'*h; %vector of mesh
if (choice==2)
    s=2*abs(s-K);
end
g=P0(s);
```

```
%% Condition Gamma
%%
B1=zeros(I);
gamma=s/h^2;
for i=1:I;   B1(i,i)=2*gamma(i);  end
for i=2:I;   B1(i,i-1)=-gamma(i); end
for i=1:I-1; B1(i,i+1)=-gamma(i); end

b1=@(t) [gamma(1)*P0(t)-G(1);-G(2:I-1,1);gamma(end)*ur-G(end)];
%% Newton for g_hat
%%
for n=0:N-1
    t=n*dt;
    if (choice==2)
        t=2*abs(t-K);
    end
    if (n==0); B=ID; end
    b=P0(s);bh=b1(t); Bh=B1;
    gold=g;
    x0=g; eps=1e-10;kmax=50;
    [g_hat,k]=newton_sol(Bh,bh,B,b,x0,eps,kmax);
    %- Verification
    err=norm(g_hat-gold);
    fprintf('Verif: err=%10.5f\n',err);

    if mod(n+1,deltan)==0 %- Affichage tous les deltan pas.

        %- Graphiques:
        t1=(n+1)*dt;
        if (choice==2)
            t1=2*abs(t1-K);
        end
        ploot(t1,s,g_hat,1);% pause(1e-3);

        %- Calculs d'erreurs:
        %COMPLETER
        Pex=BS(t1,s);   %- Appel de la solution Black et Scholes
        errLI=norm(g_hat-Pex,'inf');        %- Calcul erreur Linfty
        fprintf('t=%5.2f; Err.Linf=%8.5f',t1,errLI);
        fprintf('\n');
    end
end
hold off;
disp('Press any button to continue'); pause;

%% Sonner and Touzi. Compare the results
%
if (choice==1)
g_hat_a=g_hat_SonTou(s); % An analytical solution of g_hat in the case g(s)=max(K-s,0)
```

```
% Graphical representation
figure(3);
clf; % Clear current figure window
axis =[Xmin,Xmax,Ymin,Ymax];
sgraph  =[Smin;s;Smax];
Pgraph  =[P0(t);g_hat;ur];
Pexgraph=[P0(t);g_hat_a;ur];
R1 = plot(sgraph,Pexgraph,'y--o');
hold on;
R2 =plot(sgraph,Pgraph,'red.-');
titre=strcat('Put Option');
title(titre);
xlabel('s');
ylabel('price');
legend([R1 R2],'Scheme','Analytical');
grid;
disp('Press any button to continue'); pause;

% Precision
mape=nanmean(abs(((g_hat-g_hat_a)./g_hat_a*100))); % mean absolute percentage error
end
%% Matrices for v
%%
A=zeros(I);
alpha=sigma^2/2 * s.^2 /h^2;
bet=r*s/(2*h);
for i=1:I;   A(i,i) = 2*alpha(i) + r; end
for i=2:I;   A(i,i-1) = -alpha(i) + bet(i); end
for i=1:I-1; A(i,i+1) = -alpha(i) - bet(i); end

q= @(t) [(-alpha(1) + bet(1))* P0(t);  zeros(I-2,1);  (-alpha(end) - bet(end))* ur];
%% Newton for v
%%
V=g_hat;
for n=0:N-1
    t=n*dt;
    if (choice==2)
        t=2*abs(t-K);
    end
    if (n==0); B=ID+dt*A; end
    Vold=V;
    b=V-dt*q(t); x0=V; eps=1e-10; kmax=50; Bh=B1; bh=b1(t);
    [V,k]=newton_sol(B,b,Bh,bh,x0,eps,kmax);
    %- Verification
    err=norm(V-Vold);
    fprintf('Verif: err=%10.5f\n',err);

    if mod(n+1,deltan)==0 %- Affichage tous les deltan pas.
```

```

        %- Graphiques:
        t1=(n+1)*dt;
        if (choice==2)
            t1=2*abs(t1-K);
        end
        ploot(t1,s,V,2); pause(1e-3);

        %- Calculs d'erreurs:
        %COMPLETER
        Pex=BS(t1,s);   %- Appel de la solution Black et Scholes
        errLI=norm(V-Pex,'inf');        %- Calcul erreur Linfty
        fprintf('t=%5.2f; Err.Linf=%8.5f',t1,errLI);
        fprintf('\n');
    end
end
hold off;
disp('Press any button to continue'); pause;
%% Classical Black and Scholes
%%
P=P0(s);
for n=0:N-1
    t=n*dt;
    t1=t+dt;
    P = (ID + dt*A)\(P-dt*q(t1));
end

if mod(n+1,deltan)==0         %- Printings at each deltan steps.

    %- Graphs:
    t1=(n+1)*dt;
    if (choice==2)
        t1=2*abs(t1-K);
    end
    ploot(t1,s,P,3);
    disp('Press any button to continue'); pause;
end
dif=norm(P-V,'inf');
fprintf('difference is equal to %5.2f\n',dif);
%% Timer end
%%
toc()
```

B Newton's algorithm

```
function [x,k,err]=newton(B,b,Bh,bh,x0,eps,kmax)
%- Methode de Newton pour resoudre min(Bx-b,Bhx-bh)=0;

k=0;
x=x0;
err=eps+1;
while( k<kmax & err>eps )

    k=k+1;

    xold=x;

    %- Definition de F(x) et de F'(x):
    F=min(B*x-b,Bh*x-bh);
    Fp=Bh;
    i=find(B*x-b<=Bh*x-bh); Fp(i,:)=B(i,:);

    %- Definition nouvel x
    x=x-inv(Fp)*F;

    %- Estimateur pour convergence
    err=norm(x-xold,'inf');
end
end
```

C Black and Scholes scheme

```
%Black and Scholes Scheme
function P=BS(t,s)
global K r sigma
if t==0
    P=max(K-s,0);
else
    P=ones(size(s))*K*exp(-r*t);
    i=find(s>0);
    tau=sigma^2*t;
    dm=(log(s(i) /K) + r*t - 0.5*tau) / sqrt(tau);
    dp=(log(s(i) /K) + r*t + 0.5*tau) / sqrt(tau);
    P(i)=K*exp(-r*t)*(Normal(-dm)) - s(i).*(Normal(-dp));
end

function y=Normal(x)
  %y=cdf('normal',x,0,1);
  y=0.5*erf(x/sqrt(2))+0.5;
```

D Figures

```
function ploot(t,s,P,a)
global Xmin Xmax Ymin Ymax
global Smin Smax I P0 K
Pex=P0(s);
figure(1);
clf;
axis =[Xmin,Xmax,Ymin,Ymax];
h=(Smax-Smin)/(I+1);
s1=Smin+(1:I)'*h;
sgraph  =[Smin;s1;Smax];
Pgraph  =[P0(t);P;  0];
Pexgraph=[P0(t);Pex;0];
if (t>5)
    if (t<K)
        t=t/2+K;
    else
        t=K-t/2;
    end
end
if (Pgraph~=Pexgraph)
    line1=plot(sgraph,Pexgraph,'blue.-','Linewidth',2);
else
    if (a==1)
        line1=plot(sgraph,Pexgraph,'blue.-','Linewidth',2); % Payoff function
        hold on;
        line2=plot(sgraph,Pgraph,'black.-','Linewidth',2);
        legend([line2],'Black-Scholes','Scheme');
        titre=strcat('Numerical solution for g-hat. t=',num2str(t)); title(titre);
        legend(line2,{'$\hat{g}$'},'Interpreter','latex');
    elseif (a==2)
        line1=plot(sgraph,Pexgraph,'blue.-','Linewidth',2); % Payoff function
        hold on;
        line2=plot(sgraph,Pgraph,'black.-','Linewidth',2);
        legend([line2,line1],'Black-Scholes','Scheme');
    elseif (a==3)
        line1=plot(sgraph,Pexgraph,'blue.-','Linewidth',2); % Payoff function
        hold on;
        line2=plot(sgraph,Pgraph,'black.-','Linewidth',2);
        legend([line1,line2],'Black-Scholes','Scheme');
    end
end
xlabel('s');
ylabel('price');
grid;
end
```

4 References

1. H.M.Sommer, N.Touzi, *Superreplication under gamma constraints*, SIAM J. Control Optim.,Vol.39 (1):73-96,2000. 2. H.Zidani, O.Bokanowski, S. Maroso, *Some convergence results for Howard's algorithm*, Preprint Inria, 2007. (http://hal.inria.fr/inria-00179579/fr/)

3. P. Wilmott, S. Howison, and J. Dewynne, *The Mathematics of Financial Derivatives*, Cambridge: Cambridge University Press, 1995.

4. M.J. Brennan and E.S. Schwartz, *The valuation of American put options*, Journal of Finance 32 (1977), 449–462. (https://web.stanford.edu/class/msande316/slides/060104.pdf)

5. M. R. Grossinho, Y. F. Kord, D. Sevcovic, *Pricing American Call Options by the Black-Scholes Equation with a Nonlinear Volatility Function* (https://arxiv.org/pdf/1707.00358.pdf)