

Exercise 1

$$\begin{cases} \partial_t V(s,t) - \frac{\sigma^2}{2} S^2 \partial_{ss} V(s,t) - rS \partial_s V(s,t) + rV(s,t) = 0 & \text{in } [0, S_{\max}) \times (0, T] \\ V(S_{\max}, t) = 0 \\ V(s, 0) = G(s) = \max\{K-s, 0\} \end{cases} \quad (1)$$

a) Take $v = \partial_r V$ and differentiate the previous equation.

$$\bullet \partial_r \left(\partial_t V(s,t) - \frac{\sigma^2}{2} S^2 \partial_{ss} V(s,t) - rS \partial_s V(s,t) + rV(s,t) \right) = 0$$

$$\Rightarrow \partial_t \partial_r V(s,t) - \sigma S^2 \partial_{ss} V(s,t) - \frac{\sigma^2}{2} S^2 \partial_{ss} \partial_r V(s,t) - rS \partial_s \partial_r V(s,t) + r \partial_r V(s,t) = 0$$

$$\Rightarrow \partial_t v(s,t) - \frac{\sigma^2}{2} S^2 \partial_{ss} v(s,t) - rS \partial_s v(s,t) + r v(s,t) = \sigma S^2 \partial_{ss} V(s,t)$$

$$\bullet \partial_r V(S_{\max}, t) = 0 \Rightarrow v(S_{\max}, t) = 0$$

$$\bullet \partial_r V(s, 0) = \partial_r G(s) = 0 \Rightarrow v(s, 0) = 0$$

Then, we obtain the PDE

$$\begin{cases} \partial_t v(s,t) - \frac{\sigma^2}{2} S^2 \partial_{ss} v(s,t) - rS \partial_s v(s,t) + r v(s,t) = \sigma S^2 \partial_{ss} V(s,t) \\ v(S_{\max}, t) = 0 \\ v(s, 0) = 0 \end{cases} \quad \text{in } [0, S_{\max}) \times (0, T]$$

This is a parabolic PDE since the coefficients of the differential operator relating to the $\partial_{tt} v$ and $\partial_s \partial_t v$ are equal to 0.

Moreover, unlike equation (1), the PDE for v is not homogeneous.

Take Home Exam 4: exercise 1 b) c) g) h)

Alessandro La Farciola

December 1, 2022

In this file there are the Matlab codes for *Exercises b) c) d)*.

Exercise b)

- (i) Compute the solution of the first PDE, namely $V(S,t)$.
- (ii) Here, I want to approximate with finite differences the RHS of the PDE for ν . In particular, I use

$$D_h^2 u(x) = \frac{u(x+h) - 2u(x) + u(x-h)}{h^2}.$$

Since I have the vector V that is the numerical solution of the first PDE, for each $i = 2, \dots, \text{end}-1$

$$D_h^2 V(i) = \frac{V(i+1) - 2V(i) + V(i-1))}{h^2}, \quad \text{where } h = \frac{S_{max}}{N_x}.$$

- (iii) Plot the exact solution computed with *blsvega* and the one computed before.

The maximum error of the approximation is 0.0015 and the following figure shows the result obtained.

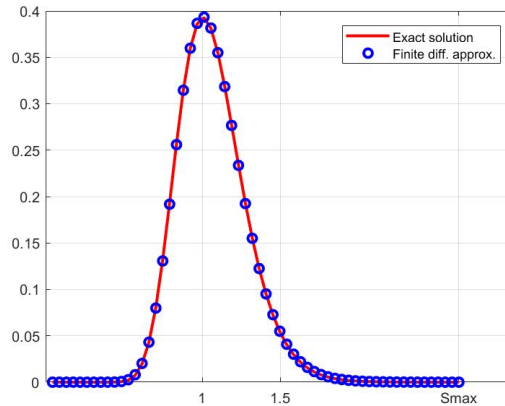


Figure 1: Comparison the exact solution and the approximation of *vega* .

Here it is the code for the solution of the 1.b.

```

1  %(i)
2  sigma=0.2; r=0.015; K=1; T=1; tol=1e-6; Nx=60; Nt=100;
3  initial_cond=@(S) max(K-S,0);
4  %forcing
5  rhs= @(x,t) 0*x;
6  % compute the right boundary
7  Smax = K*exp(sigma^2*T/2-sigma*sqrt(T)*norminv(tol/K));
8  bc_right=@(t) 0*t;
9  theta=0.5;
10
11 % numerical solution for V
12 [V,FD_grid,time_steps] = bs_timestepping_const_sigma(sigma,r,rhs,bc_right
    ,initial_cond,Smax,Nx,T,Nt,theta);
13
14 %discretization of rhs
15 h = Smax/Nx;
16 ddS=@(t) (V(3:end,t+1)-2*V(2:end-1,t+1)+V(1:end-2,t+1))/h^2;
17 rhs_nu=@(S,t) [S(1) sigma*S(2:end).^2.*ddS(t)'];
18 initial_cond_nu=@(S) 0*S;
19 % numerical solution for nu
20 [nu,FD_grid_nu,time_steps] = bs_timestepping_const_sigma_vega(sigma,r,
    rhs_nu,bc_right,initial_cond_nu,Smax,Nx,T,Nt,theta);
21
22 % exact solution with blsvega
23 Vexact=[];
24 for v=FD_grid(2:end)
25     Put = blsvega(v, K, r, T, sigma);
26     Vexact=[Vexact Put];
27 end
28 %comparison with blsvega
29 figure
30 set(gca,'FontSize',20)
31 plot(FD_grid(2:end),Vexact,'r','LineWidth',2,'DisplayName','Exact
    solution')
32 hold on
33 plot(FD_grid(2:end),nu(2:end,end),'ob','LineWidth',2,'DisplayName','
    Finite diff. approx.')
34 grid on
35 legend show
36 set(legend,'Location','NorthEast')
37 set(gca,'XTick',[1 1.5 Smax])
38 set(gca,'XTickLabel',{'1','1.5','Smax'})
39 % difference between exact solution and finite differences approximation
40 max(abs(nu(2:end,end))-Vexact))

```

In the previous code, I used two functions:

- `bs_timestepping_const_sigma`, that is the one written in Exercise 1 of Homework 8.
- `bs_timestepping_const_sigma_vega`, that is the same of the previous one, but at line 108 I changed `f_new = forcing(inner_grid,t_new)` with `f_new = forcing(inner_grid,tn)` (This func-

tion is included in the matlab file *TH4_b.m*).

Exercise c)

In this section, I firstly present the script to obtain the plot to show the rate of convergence.

```

1  %1.c
2
3  sigma=0.2; r=0.015; K=1; T=1; tol=1e-6;
4  initial_cond=@(S) max(K-S,0);
5  %forcing
6  rhs= @(x,t) 0*x;
7  % compute the right boundary
8  Smax = K*exp(sigma^2*T/2-sigma*sqrt(T)*norminv(tol/K));
9  bc_right=@(t) 0*t;
10 theta=0.5;
11
12
13 %check of convergence
14 err=zeros(6,1);
15 h=zeros(6,1);
16 for i=0:5
17     Nx=10*2^i;
18     Nt=0.6*Nx;
19     % finite differences solver
20     [V,FD_grid,time_steps] = bs_timestepping_const_sigma(sigma,r,rhs,
        bc_right,initial_cond,Smax,Nx,T,Nt,theta);
21     h(i+1)=Smax/Nx;
22     ddS=@(t) (V(3:end,t+1)-2*V(2:end-1,t+1)+V(1:end-2,t+1))/h(i+1)^2;
23     rhs_nu=@(S,t) [S(1) sigma*S(2:end).^2.*ddS(t)'];
24     initial_cond_nu=@(S) 0*S;
25     [nu,FD_grid_nu,time_steps] = bs_timestepping_const_sigma_vega(sigma,r
        ,rhs_nu,bc_right,initial_cond_nu,Smax,Nx,T,Nt,theta);
26     % exact solution with blsprice
27     Vexact=[];
28     for v=FD_grid(2:end)
29         Put = blsvega(v, K, r, T, sigma);
30         Vexact=[Vexact Put];
31     end
32     err(i+1)=max(abs(nu(2:end,end)'-Vexact));
33 end
34 figure
35 set(gca,'FontSize',20)
36 loglog(h,h*err(1)/h(1),'-','LineWidth',2,'MarkerSize',10,'DisplayName','
    linear')
37 hold on
38 loglog(h,h.^2*err(1)/h(1)^2,'—','LineWidth',2,'MarkerSize',10,'
    DisplayName','quadratic')
```

```

39 hold on
40 loglog(h,err,'-xk','LineWidth',1.5,'MarkerSize',8,'DisplayName','errors')
41 grid on
42 legend show
43 set(legend,'Location','SouthEast')

```

Now, the following figure shows the plot obtained with the previous script

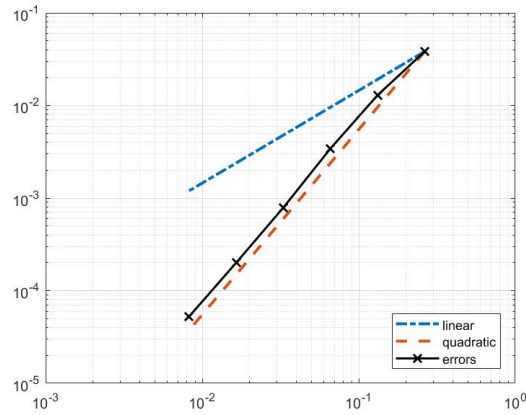


Figure 2: Order of convergence in a log-log plot.

As we can see from the previous plot, the order of convergence is quadratic.

Exercise d)

- (i) In this section, I firstly present the script to *directly approximate* ν using the central difference scheme for the derivative with respect to σ .

```

1 sigma=0.2; r=0.015; K=1; T=1; tol=1e-6;
2 initial_cond=@(S) max(K-S,0);
3 %forcing
4 rhs= @(x,t) 0*x;
5 % compute the right boundary
6 Smax = K*exp(sigma^2*T/2-sigma*sqrt(T)*norminv(tol/K));
7 bc_right=@(t) 0*t;
8 theta=0.5;
9 Nx=60;
10 Nt=100;
11
12 %1.d
13
14 %(i)
15 %central finite difference scheme to directly approximate nu
16 deltasigma= [0.0005; 0.001; 0.01; 0.02; 0.03; 0.05];

```

```

17 figure
18 set(gca,'FontSize',13)
19 for i=6:-1:1
20     [V,FD_grid,time_steps] = bs_timestepping_const_sigma(sigma+
        deltasigma(i),r,rhs,bc_right,initial_cond,Smax,Nx,T,Nt,theta);
21     [V1,FD_grid,time_steps] = bs_timestepping_const_sigma(sigma-
        deltasigma(i),r,rhs,bc_right,initial_cond,Smax,Nx,T,Nt,theta);
22     nu1=(V(:,end)-V1(:,end))/(2*deltasigma(i));
23     hold on
24     plot(FD_grid(1:end),nu1(1:end,end),'DisplayName',sprintf('Approx
        ds=%.4f',deltasigma(i)))
25 end
26 Vexact=[];
27 for v=FD_grid(2:end)
28     Put = blsvega(v, K, r, T, sigma);
29     Vexact=[Vexact Put];
30 end
31 hold on
32 plot(FD_grid(2:end),Vexact,'r','LineWidth',0.5,'DisplayName','Exact
    solution')
33 grid on
34 legend show
35 set(legend,'Location','NorthEast')
36 set(gca,'XTick',[1 1.5 Smax])
37 set(gca,'XTickLabel',{'1','1.5','Smax'})
38
39
40 %(ii)
41
42 %check of convergence
43 err=zeros(6,1);
44 h=zeros(6,1);
45 for i=0:5
46     Nx=10*2^i;
47     Nt=0.6*Nx;
48     % finite differences solver
49     [V,FD_grid,time_steps] = bs_timestepping_const_sigma(sigma,r,rhs,
        bc_right,initial_cond,Smax,Nx,T,Nt,theta);
50     h(i+1)=Smax/Nx;
51     ddS=@(t) (V(3:end,t+1)-2*V(2:end-1,t+1)+V(1:end-2,t+1))/h(i+1)^2;
52     rhs_nu=@(S,t) [S(1) sigma*S(2:end).^2.*ddS(t)'];
53     initial_cond_nu=@(S) 0*S;
54     [nu,FD_grid_nu,time_steps] = bs_timestepping_const_sigma_vega(
        sigma,r,rhs_nu,bc_right,initial_cond_nu,Smax,Nx,T,Nt,theta);
55     % exact solution with blsprice
56     Vexact=[];
57     for v=FD_grid(2:end)
58         Put = blsvega(v, K, r, T, sigma);
59         Vexact=[Vexact Put];

```

```

60     end
61     err(i+1)=max(abs(nu(2:end,end))-Vexact));
62 end
63 figure
64 set(gca,'FontSize',20)
65 loglog(h,h*err(1)/h(1),'-.','LineWidth',2,'MarkerSize',10,'
    DisplayName','linear')
66 hold on
67 loglog(h,h.^2*err(1)/h(1)^2,'—','LineWidth',2,'MarkerSize',10,'
    DisplayName','quadratic')
68 hold on
69 loglog(h,err,'-x','LineWidth',1.5,'MarkerSize',8,'DisplayName','
    errors')
70 grid on
71 legend show
72 set(legend,'Location','SouthEast')
73
74 %adding the new errors for the central finite differe scheme for all
    deltasigmas
75 for j=6:-1:1
76     for i=0:5
77         Nx=10*2^i;
78         Nt=0.6*Nx;
79         h(i+1)=Smax/Nx;
80         % finite differences solver
81         [V,FD_grid,time_steps] = bs_timestepping_const_sigma(sigma+
            deltasigma(j),r,rhs,bc_right,initial_cond,Smax,Nx,T,Nt,theta);
82         [V1,FD_grid,time_steps] = bs_timestepping_const_sigma(sigma-
            deltasigma(j),r,rhs,bc_right,initial_cond,Smax,Nx,T,Nt,theta);
83         nu1=(V(:,end)-V1(:,end))/(2*deltasigma(j));
84         % exact solution with blsprice
85         Vexact=[];
86         for v=FD_grid(2:end)
87             Put = blsvega(v, K, r, T, sigma);
88             Vexact=[Vexact Put];
89         end
90         err(i+1)=max(abs(nu1(2:end,end))-Vexact));
91     end
92     hold on
93     loglog(h,err,'-x','LineWidth',1,'MarkerSize',6,'DisplayName',
        sprintf('err ds=%4f',deltasigma(j)))
94 end
95 grid on
96 legend show
97 set(legend,'Location','SouthEast')

```

The following figure shows the results obtained.

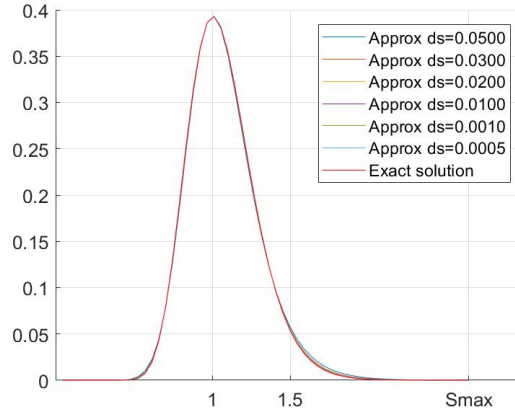


Figure 3: Approximation of *vega* with central difference scheme.

- (ii) Now, I compare the methods just applied and the one used in part b) to approximate ν . In particular, I add the errors of the previous methods for each $\delta\sigma$ on the log-log plot obtained in part c).

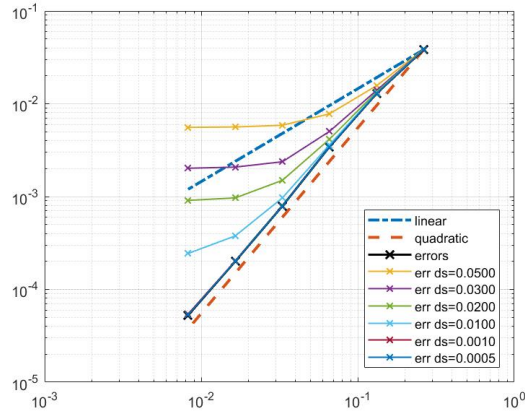


Figure 4: Approximation of *vega* with central difference scheme.

As we can see from the last plot, the errors for the methods implemented with the central difference scheme are higher than the ones computed in part b). Moreover, the rate of convergence of the error increases when $\delta\sigma$ decreases. In particular, for $\delta\sigma = 0.001$ and $\delta\sigma = 0.0005$ we reach the quadratic order of convergence as in part b).