OPTION PRICING USING NUMERICAL METHODS

B.V. Rathish Kumar[1], J.H.M. ten Thije Boonkkamp[2],
Abhinav Mehta, Prakhar Aggarwal, Lakshya Khandelwal

Abstract

We implement various numerical methods to solve for Black-Scholes option pricing model. We also intoduce a new numerical technique called finite volume-complete flux scheme. We survey their performance in terms of accuracy at different number of grid points and evaluate their relative performance.

1 European Option

The Black-Scholes equation is a partial differential equation(PDE) governing the price evolution of a European call under Black-Scholes model. For a European call on an uderlying stock paying no dividends, the equation is:

$$\frac{\partial V}{\partial t}+\frac{1}{2}\sigma^2 S^2\frac{\partial^2 V}{\partial S^2}+rS\frac{\partial V}{\partial S}-rV=0$$

The boundary conditions for the above equation are:

$$\begin{aligned}
&V(0,t)=ke^{-r(T-t)} &&,0\leqslant t\leqslant T\\
&V(S,T)=\max\{K-S,0\} &&,0<S<L\\
&V(S_{\max},t)=S_{\max}-e^{-r(T-t)}K &&,0\leqslant t\leqslant T
\end{aligned}$$

where

$$\begin{aligned}
K &= \text{strike price}\\
S &= \text{stock price}\\
T &= \text{expiration time}\\
V &= \text{option price}\\
t &= \text{time}
\end{aligned}$$

1. Dr. B. V. Rathish Kumar(corresponding author), Professor, Department of Mathematics and Statistics, IIT Kanpur, Kanpur, Uttar Pradesh, India-208016, email: bvrk@iitk.ac.in

2. Dr. J.H.M. ten Thije Boonkkamp, Professor, Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, The Netherlands, e-mail: tenthije@win.tue.nl

2 Complete Flux Scheme

The model problem for the method is :

$$\varphi_t + (u\varphi - \epsilon\varphi_x)_x = S(\varphi) \qquad 0 < x < 1$$

$$\begin{aligned}
\text{Boundary conditions:}\quad & \varphi(0,t) = \varphi_L(t)\\
& \varphi(1,t) = \varphi_R(t)\\
\text{Assume:}\quad & \epsilon = \epsilon(x) \geqslant \epsilon_{\min} > 0\\
& u = u(x)\\
& s = S(\varphi(x)) = s(x)\\
\text{Define:}\quad & f = (u\varphi - e\varphi_x)\\
& \hat{S} = S(\varphi) - \varphi_t\\
& \lambda = \frac{u}{\epsilon}\\
& P = \lambda \triangle x\\
& \Lambda = \int_{x_{j+1/2}}^{x} \lambda(\xi) d\xi\\
& S(x) = \int_{x_{j+1/2}}^{x} \hat{S}(\xi) d\xi
\end{aligned}$$

For numerical solution, we divide the spatial domain into N points such that $\triangle x = \frac{1}{N-1}$, so, $x_j = (j-1)\triangle x$ for $j = 1, 2, 3 \ldots, N$.

$$\begin{aligned}
\text{Define:}\quad & x_{j+1/2} = \frac{1}{2}(x_j + x_{j+1})\\
& f(x_{j+1/2}) = f^h_{j+1/2} + f^i_{j+1/2}\\
& \Lambda_{j+1/2} = \frac{1}{2}(\Lambda_j + \Lambda_{j+1})
\end{aligned}$$

Suppose $\Omega_j = (x_{j-1/2}, x_{j+1/2})$

Now using the above notation, we get the model problem as :

$$\begin{aligned}
& \varphi_t + f_x = s\\
& \frac{\partial}{\partial t}\int_{\Omega_j} \varphi\, dx + \int_{\Omega_j} F_x\, dx = \int_{\Omega_j} s\, dx\\
& \frac{\partial}{\partial t}\int_{\Omega_j} \varphi\, dx + F(x_{j+1/2}) - F(x_{j-1/2}) = \int_{\Omega_j} s\, dx\\
& \dot{\varphi}_j \triangle x + F_{j+1/2} - F_{j-1/2} = s_j \triangle x
\end{aligned}$$

We know that : $f' = \hat{S}$

On integrating between $x_{j+1/2}$ and x :

$$f(x) - f(x_{j+1/2}) = S(x)$$

Now substituting for $f(x) = -\epsilon(\varphi' - \lambda\varphi)$ from above substitutions, we get :

$$\begin{aligned}
& -\epsilon e^{\Lambda}(\varphi e^{-\Lambda})' - f(x_{j+1/2}) = S(x)\\
& (\varphi e^{-\Lambda})' + \frac{e^{-\Lambda}}{\epsilon} f(x_{j+1/2}) = -\frac{e^{-\Lambda}}{\epsilon} S(x)
\end{aligned}$$

On integrating between x_j and x_{j+1} :

$$\varphi_{j+1}\, e^{-\Lambda_{j+1}} - \varphi_j\, e^{-\Lambda_j} + \left(\int_{x_j}^{x_{j+1}} \frac{e^{-\Lambda}}{\epsilon}\, dx\right) f(x_{j+1/2}) = -\int_{x_j}^{x_{j+1}} \frac{e^{-\Lambda}}{\epsilon}\, S(x) dx$$

We know that $\Lambda_{j+1} - \Lambda_j = \int_{x_j}^{x_{j+1}} \lambda dx = <\lambda, 1>$

$$\begin{aligned}\Lambda_j &= \Lambda_{j+1/2} - \frac{<\lambda, 1>}{2}\\ \Lambda_{j+1} &= \Lambda_{j+1/2} + \frac{<\lambda, 1>}{2}\end{aligned}$$

Separating the homogeneous and non homogeneous term, we get :

$$\begin{aligned}f^h_{j+1/2} &= -\frac{e^{-\Lambda_{j+1/2}}\big(\varphi_{j+1}\, e^{-<\lambda,1>/2} - \varphi_j\, e^{<\lambda,1>/2}\big)}{<\epsilon^{-1}, e^{-\Lambda}>}\\ f^i_{j+1/2} &= \frac{-<\epsilon^{-1}e^{-\Lambda}, S>}{<\epsilon^{-1}, e^{-\Lambda}>}\end{aligned}$$

Now we need to evaluate $e^{-\Lambda_{j+1/2}}$:

$$\begin{aligned}<\lambda, e^{-\Lambda}> &= \int_{x_j}^{x_{j+1}} \lambda e^{-\Lambda}\, dx\\ &= [-e^{-\Lambda}]_{x_j}^{x_{j+1}}\\ &= e^{-\Lambda_j} - e^{-\Lambda_{j+1}}\\ &= e^{-\Lambda_{j+1/2}}\big(e^{<\lambda,1>/2} - e^{-<\lambda,1>/2}\big)\end{aligned}$$

$$e^{-\Lambda_{j+1/2}} = \frac{<\lambda, e^{-\Lambda}>}{e^{<\lambda,1>/2} - e^{-<\lambda,1>/2}}$$

Substituting, we get :

$$\begin{aligned}f^h_{j+1/2} &= \frac{<\lambda, e^{-\Lambda}>}{<\epsilon^{-1}, e^{-\Lambda}><\lambda, 1>}[B(-<\lambda,1>)\varphi_j - B(<\lambda,1>)\varphi_{j+1}]\\ f^i_{j+1/2} &= \frac{-<\epsilon^{-1}e^{-\Lambda}, S>}{<\epsilon^{-1}, e^{-\Lambda}>}\end{aligned}$$

where $B(z) = \frac{z}{e^z - 1}$

$$\begin{aligned}\text{Define:}\quad \sigma &= \frac{x - x_j}{\triangle x} && ;0 \leqslant \sigma \leqslant 1, x_j \leqslant x \leqslant x_{j+1}\\ x &= x_j + \sigma\triangle x && ;dx = \triangle x\, d\sigma\\ \xi &= x_j + \eta\triangle x && ;d\xi = \triangle x\, d\eta\end{aligned}$$

Now,

$$\begin{aligned}<\epsilon^{-1}e^{-\Lambda}, S> &= \int_{x_j}^{x_{j+1}} \epsilon^{-1}e^{-\Lambda} S dx\\ &= \int_{x_j}^{x_{j+1/2}} \epsilon^{-1}e^{-\Lambda}\int_{x_{j+1/2}}^{x} \hat{S}(\xi) d\xi dx + \int_{x_j+1/2}^{x_{j+1}} \epsilon^{-1}e^{-\Lambda}\int_{x_{j+1/2}}^{x} \hat{S}(\xi) d\xi dx\\ &= \triangle x^2 \int_0^{1/2} \epsilon^{-1}e^{-\Lambda}\int_{1/2}^{\sigma} \hat{S}(\eta) d\eta d\sigma + \triangle x^2 \int_{1/2}^{1} \epsilon^{-1}e^{-\Lambda}\int_{1/2}^{\sigma} \hat{S}(\eta) d\eta d\sigma\\ &= \triangle x^2 \int_0^{1/2} \left(\int_0^{\eta} \epsilon^{-1}e^{-\Lambda} d\sigma\right)\hat{S}(\eta) d\eta + \triangle x^2 \int_{1/2}^{1} \left(\int_{\eta}^{1} \epsilon^{-1}e^{-\Lambda} d\sigma\right)\hat{S}(\eta) d\eta d\sigma\end{aligned}$$

Substituting,

$$f^i_{j+1/2} = \triangle x \int_0^1 \varsigma(\eta)\hat{S}(\eta) d\eta$$

where $\varsigma(\eta)$: flux's Greens function such that

$$\begin{aligned}
\varsigma(\eta) &= \frac{\triangle x \int_0^\eta \epsilon^{-1} e^{-\Lambda} d\sigma}{<\epsilon^{-1}, e^{-\Lambda}>} \text{ for } 0 \leqslant \eta \leqslant 1/2 \\
&\quad \frac{-\triangle x \int_\eta^1 \epsilon^{-1} e^{-\Lambda} d\sigma}{<\epsilon^{-1}, e^{-\Lambda}>} \text{ for } 1/2 \leqslant \eta \leqslant 1
\end{aligned}$$

Suppose u is constant, then

$$\begin{aligned}
<\epsilon^{-1}, e^{-\Lambda}> &= \frac{1}{u} < \lambda, e^{-\Lambda} > \\
&= \frac{1}{u} \int_{x_j}^{x_{j+1}} \lambda e^{-\Lambda} \, dx \\
&= \frac{1}{u} [-e^{-\Lambda}]_{x_j}^{x_{j+1}} \\
&= \frac{1}{u} [e^{-\Lambda_j} - e^{-\Lambda_{j+1}}]
\end{aligned}$$

Next,

$$\begin{aligned}
\triangle x \int_0^\eta \epsilon^{-1} e^{-\Lambda} d\sigma &= \int_{x_j}^\xi \epsilon^{-1} e^{-\Lambda} \, dx \\
&= \frac{1}{u} \int_{x_j}^\xi \lambda e^{-\Lambda} \, dx \\
&= \frac{1}{u} [e^{-\Lambda}]_{x_j}^\xi \\
&= \frac{1}{u} \left[e^{-\Lambda_j} - e^{-\Lambda(\xi)} \right]
\end{aligned}$$

Now,

$$\begin{aligned}
\Lambda_j - \Lambda(\xi) &= \int_{x_{j+1/2}}^{x_j} \lambda(x) \, dx - \int_{x_{j+1/2}}^\xi \lambda(x) \, dx \\
&= -\int_{x_j}^\xi \lambda(x) \, dx \\
&= -\sigma <\lambda, 1>
\end{aligned}$$

where $\sigma = \frac{\int_{x_j}^x \lambda(\xi) \, d\xi}{<\lambda, 1>} = \frac{x - x_j}{\triangle x}$

Also,

$$\begin{aligned}
<\lambda, 1> &= \int_{x_j}^{x_{j+1}} \lambda(x) \, dx \\
&= \frac{\triangle x}{2} (\lambda_j + \lambda_{j+1}) \\
&= \frac{P_j + P_{j+1}}{2} \\
&= P_{j+1/2}
\end{aligned}$$

and

$$\begin{aligned}
\frac{<a, e^{-\Lambda}>}{<1, e^{-\Lambda}>} &= a_{j+1/2} \\
&= W(-P_{j+1/2}) a_j + W(P_{j+1/2}) a_{j+1}
\end{aligned}$$

where $W(z) = \frac{e^z - 1 - z}{z(e^z - 1)}$We approximate some quantities,

$$\begin{aligned}
\hat{S}(\sigma) &= \hat{S}(k) \\
&= \hat{S}_j \text{ if } u_{j+1/2} > 0 \\
&\quad \hat{S}_{j+1} \text{ if } u_{j+1/2} < 0
\end{aligned}$$

Now we substitute :

$$\varsigma(\eta) = \begin{array}{ll} \dfrac{1-e^{-P\sigma}}{1-e^{-P}}, & ,0 \leqslant \sigma < \dfrac{1}{2} \\ -\dfrac{1-e^{-P(1-\sigma)}}{1-e^{P}} & ,\dfrac{1}{2} < \sigma \leqslant 1 \end{array}$$

$$\int_0^1 \varsigma(\sigma, <\lambda, 1>) d\sigma \;=\; \frac{1}{2} - W(P_{j+1/2})$$

$$\epsilon_{j+1/2} \;=\; \tilde{\lambda}_{j+1/2}\, \tilde{\epsilon}_{j+1/2}$$

Thus, we get :

$$\begin{aligned} F^h_{j+1/2} &= \frac{\epsilon_{j+1/2}}{\Delta x}[B(-P_{j+1/2})\varphi - B(P_{j+1/2})\varphi_{j+1}] \\ F^i_{j+1/2} &= \left(\frac{1}{2} - W(P_{j+1/2})\right) S_{u_{j+1/2}} \Delta x \end{aligned}$$

Now, we are at position to introduce :

$$\begin{aligned} \alpha_{j+1/2} &= \frac{\epsilon_{j+1/2}}{\Delta x}\, B(-P_{j+1/2}) \\ \beta_{j+1/2} &= \frac{\epsilon_{j+1/2}}{\Delta x}\, B(P_{j+1/2}) \\ \gamma_{j+1/2} &= \max\left(\frac{1}{2} - W(P_{j+1/2}), 0\right) \\ \delta_{j+1/2} &= \min\left(\frac{1}{2} - W(P_{j+1/2}), 0\right) \end{aligned}$$

So, we get :

$$F_{j+1/2} = \alpha_{j+1/2}\, \varphi_j - \beta_{j+1/2}\, \varphi_{j+1} + \Delta x \left[\gamma_{j+1/2}\, \hat{S}_j + \delta_{j+1/2}\, \hat{S}_{j+1}\right]$$

Substituting the value of the numerical flux in our original equation :

$$\begin{aligned} S_j \Delta x = \dot{\varphi}_j \Delta x\, \alpha_{j+1/2} + \varphi_j - \beta_{j+1/2}\, \varphi_{j+1} + \Delta x \left[\gamma_{j+1/2}\, \hat{S}_j + \delta_{j+1/2}\, \hat{S}_{j+1}\right] - \alpha_{j-1/2}\, \varphi_{j-1} \\ + \beta_{j-1/2}\, \varphi_j + \Delta x \left[\gamma_{j-1/2}\, \hat{S}_{j-1} + \delta_{j-1/2}\, \hat{S}_j\right] \end{aligned}$$

Now, we write it in matrix form :

$$\mathbf{B}\dot{\Phi} + \mathbf{A}\Phi = \mathbf{B}\mathbf{S} + \mathbf{b}$$

where,

$$\mathbf{A} = \begin{pmatrix} \alpha_{5/2} + \beta_{5/2} & -\beta_{5/2} & \cdots & \\ -\alpha_{5/2} & \alpha_{-1/2} + \beta_{5/2} & \cdots & \\ \vdots & \vdots & \ddots & \beta_{N-3/2} \\ & & -\alpha_{N-3/2} & \alpha_{N-1/2} \end{pmatrix}$$

$$\mathbf{B} = \begin{pmatrix} 1-\gamma_{5/2} & & & 0 \\ \gamma_{5/2} & 1-\gamma_{7/2} & & \\ & \ddots & \ddots & \\ 0 & & -\alpha_{N-3/2} & 1-\gamma_{N-1/2} \end{pmatrix}$$

$$\mathbf{b} = \begin{pmatrix} -\Delta x\, \gamma_{3/2}\, \dot{\varphi}_L + \alpha_{3/2}\, \varphi_L + \Delta x\, \gamma_{3/2}\, S(\varphi_L) \\ 0 \\ \vdots \\ \beta_{N-1/2}\, \varphi_R \end{pmatrix}$$

Now, we will use time integration by $\theta-$method :

$$(\mathbf{B}+\triangle t\theta\,\mathbf{A})\Phi^{n+1}-\triangle t\theta\mathbf{BS}(\Phi^{n+1})-r^n=0$$

where

$$r^n=\ (\mathbf{B}-\triangle t\,(1-\theta)\,\mathbf{A})\Phi^n+\triangle t(1-\theta)\mathbf{BS}^n+[(1-\theta)\mathbf{b}^n+\theta\mathbf{b}^{n+1}]\triangle t$$

This can be solved using the Newton-Iteration procedure. In the next section, we will apply this scheme on Black-Scholes equation . Before that, we need to transform the equation for the scheme.

We substitute $S=kz$ to the black scholes equation to get :

$$\frac{\partial V}{\partial t}+\left((r-\sigma^2)zV-\left(-\frac{1}{2}\sigma^2z^2\right)V_z\right)_z=(2r-\sigma^2)V$$

And the boundary conditions become :

$$\begin{array}{ll} V(0,t)=ke^{-r(T-t)} & ,0\leqslant t\leqslant T\\ V(S,T)=\max\{K-S,0\} & ,0<S<L\\ V(S_{\max},t)=S_{\max}-e^{-r(T-t)}K & ,0\leqslant t\leqslant T\\ V(L,t)=0,\,0\leqslant t\leqslant T & ,0\leqslant t\leqslant T \end{array}$$

where

$$\begin{array}{rl} V & =\text{option price}\\ t & =\text{time}\\ S & =\text{stock price}\\ L & =\text{cut}-\text{off price}\\ T & =\text{expiration time}\\ K & =\text{strike price} \end{array}$$

Comparing the equation to one in the complete flux scheme, we get:

$$\begin{array}{rcl} \varphi & = & V\\ x & = & z\\ u & = & (r-\sigma^2)z\\ \epsilon & = & -\frac{1}{2}\sigma^2z^2\\ s & = & (r-\sigma^2)V \end{array}$$

Using the values from the last section, we get the following values for the coefficients :

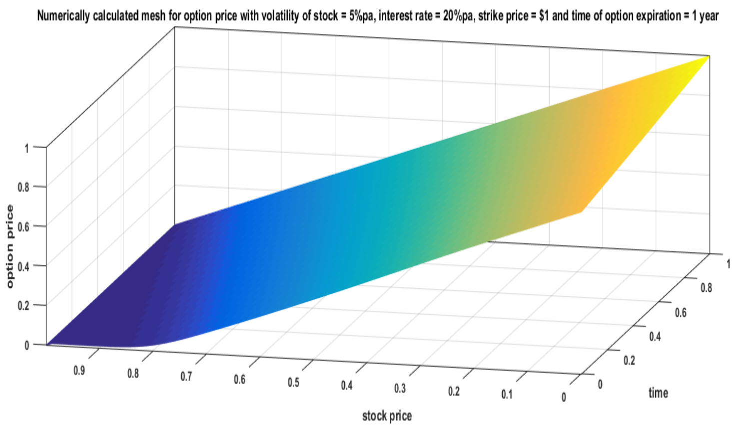
$$\begin{array}{rcl} \alpha_{j+1/2} & = & -\dfrac{1}{2\triangle x}\sigma^2x_{j+1/2}^2B\left(\dfrac{2\triangle x(\sigma^2-r)}{\sigma^2x_{j+1/2}}\right)\\ \beta_{j+1/2} & = & -\dfrac{1}{2\triangle x}\sigma^2x_{j+1/2}^2B\left(-\dfrac{2\triangle x(\sigma^2-r)}{\sigma^2x_{j+1/2}}\right)\\ \gamma_{j+1/2} & = & \max\left(\dfrac{1}{2}-W\left(-\dfrac{2\triangle x\,(\sigma^2-r)}{\sigma^2x_{j+1/2}}\right),0\right)\\ \delta_{j+1/2} & = & \min\left(\dfrac{1}{2}-W\left(-\dfrac{2\triangle x(\sigma^2-r)}{\sigma^2x_{j+1/2}}\right),0\right) \end{array}$$

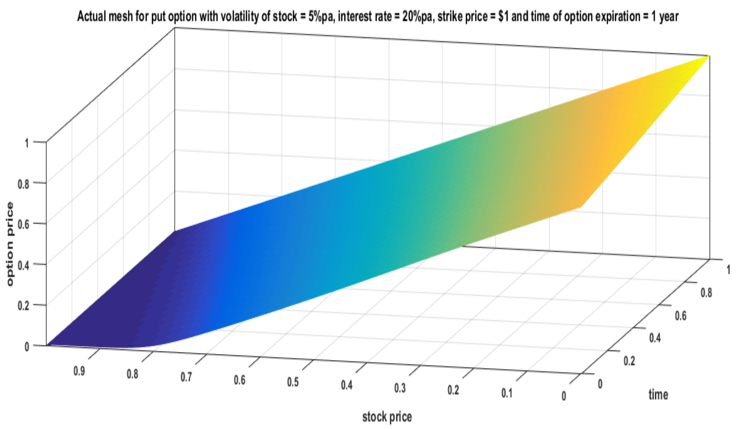
We plug these values in the matrix and compute them numerically using newton-iteration scheme to get the value of the call option.

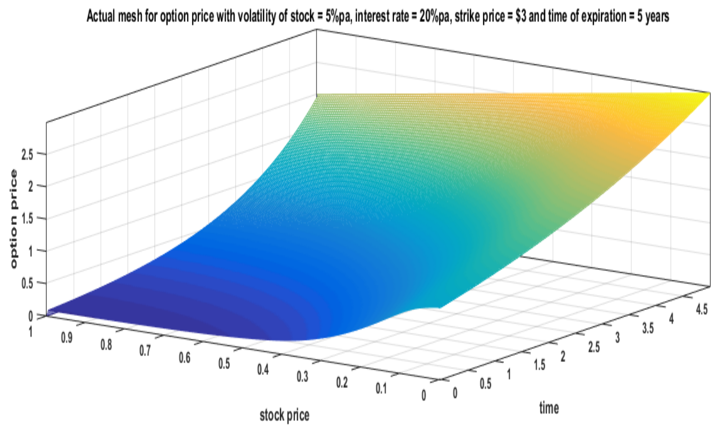
3 Stability of solution

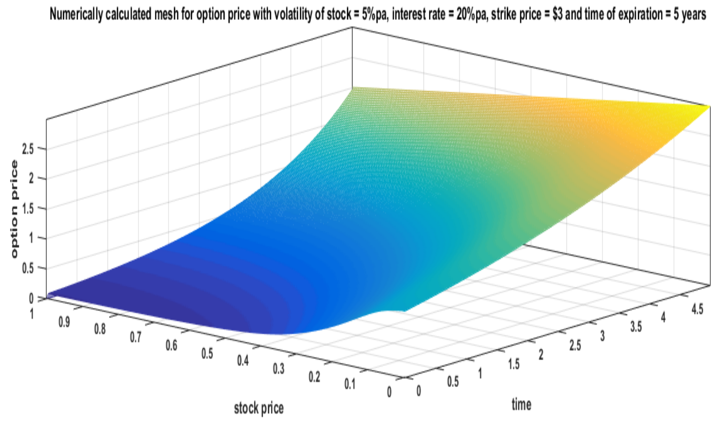
We have to ensure the stability of $\mathbf{B}\dot{\Phi} + \mathbf{A}\Phi = \mathbf{BS} + \mathbf{b}$. We can write it in the form $\dot{\Phi} = -\mathbf{B}^{-1}\mathbf{A}\Phi + \mathbf{BS} + \mathbf{b}$. If we ensure the stability of $\dot{\Phi} = -\mathbf{B}^{-1}\mathbf{A}\Phi + \mathbf{BS}$, it will be sufficient as $\mathbf{b}$ is constant. Now we have to prove $\boldsymbol{\rho}(\mathbf{B}^{-1}\mathbf{A}) < 1$. We know that $\boldsymbol{\rho}(\mathbf{B}^{-1}\mathbf{A}) \leqslant \boldsymbol{\rho}(\mathbf{B}^{-1})\boldsymbol{\rho}(\mathbf{A})$. We have to prove $\boldsymbol{\rho}(\mathbf{B}^{-1}) \leqslant 1$ and $\boldsymbol{\rho}(\mathbf{A}) \leqslant 1$.

4 Plots

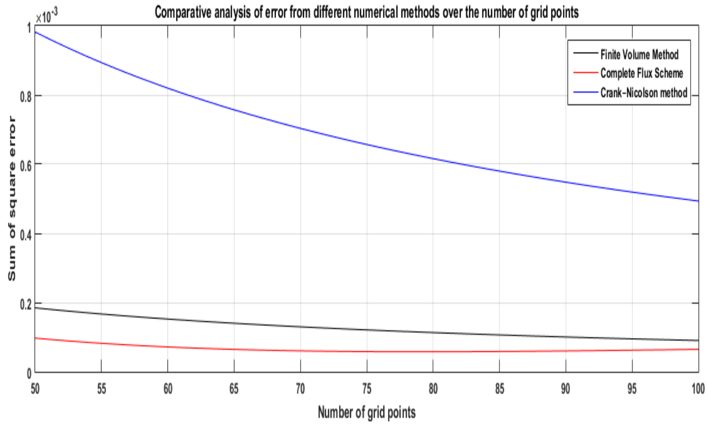








5 Error plot



6 References

1. ten Thije Boonkkamp, J.H.M., van Dijk, J., Liu, L., Peerenboom, K.S.C. : Extension of the Complete Flux Scheme to Systems of Conservation Laws, J Sci Comput (2012) 53:552–568, DOI 10.1007/s10915-012-9588-5

7 MATLAB Code

```
% program code
clear all
time(200) = 0;
error(200) = 0;
q= 35;
for k2 = 50 :1 : 60
tic
% parameters
r=0.2; % interest rate
sigma=0.05; % volatility
K=1; % strike price
Smax=1; % maximal asset price
```

```
T=1.1; % maturity time

% grid
ns=k2; % space points
nt=k2; % time points
dS=Smax/ns; % space step, l(i)
dtau=T/nt; % time step
S=0:dS:Smax; % asset prices grid
tau=0:dtau:T; % time grid

% put option
V=zeros(ns+1,nt+1); % array for option

%PUT
for i=1:ns+1
V(i,1)=max(K-S(i),0); % payoff t=T or tau=0
end

for i=1:nt+1
V(1,i)=K*exp(-r*tau(i)); % boundary condition S=0
V(ns+1,i)=0; % boundary condition S=Smax
end

% constants
a = 0.5*sigma^2; b = r-sigma^2; c = r+b; % eq. (2.4)
k=b/a;

% arrays
xi = zeros(ns,1);
phi = zeros(ns,1);
psi = zeros(ns,1);
M = zeros(ns-1,ns-1);
R = zeros(ns-1,1);

% coefficients, eq-s. (2.14), (2.15)
xi(2) = (S(2)*(a-b)) / (4*dS);
phi(2) = (0.5*b*S(3)^k*(S(2)+S(3))) / (dS*(S(3)^k - S(2)^k));
psi(2) = -(S(2)*(a+b))/(4*dS) ...
- (0.5*b*S(2)^k*(S(2)+S(3))) / (dS*(S(3)^k - S(2)^k)) -c;

for j=3:ns
xi(j) = 0.5*b*S(j-1)^k*(S(j-1)+S(j)) / (S(j)^k-S(j-1)^k) /dS;
phi(j) = 0.5*b*S(j+1)^k*(S(j)+S(j+1)) / (S(j+1)^k-S(j)^k) /dS;
psi(j) = -0.5*b*S(j)^k*(S(j-1)+S(j)) / (S(j)^k-S(j-1)^k) /dS-...
0.5*b*S(j)^k*(S(j+1)+S(j)) / (S(j+1)^k-S(j)^k) /dS -c;
end

for j=2:nt+1 % main time loop
% Inhomogeneity, eq. (2.22)
R(1) = xi(2)*V(1,j-1);
R(ns-1) = phi(ns)*V(ns+1,j-1);
% Matric, eq. (2.23)

for i=1:ns-2
M(i+1,i) = xi(i+2);
M(i,i+1) = phi(i+1);
```

```
M(i,i) = psi(i+1);
end

M(ns-1,ns-1) = psi(ns);
V(2:ns,j) = (eye(ns-1,ns-1) - dtau*M)\(V(2:ns,j-1) + dtau*R);
end

% EXACT SOLUTION
Vex=zeros(ns+1,1);
coeff = r+0.5*sigma^2;

for i=1:ns+1
d1=(log(S(i)/K)+coeff*T)/(sigma*sqrt(T));
d2=d1-sigma*sqrt(T);
D1=normcdf(d1,0,1);
D2=normcdf(d2,0,1);
Vex(i)=K*exp(-r*T)*(1-D2)-S(i)*(1-D1);
end
% end EXACT SOLUTION

%graph
% plot (S,V(:,nt+1),'--b');
% xlabel ('S')
% ylabel ('V(S,T)')
% title ('Option Price')
% hold on
% plot(S,Vex,'-r')
% hold off
%
% % graph for error
% plot (S, (Vex(:,1)-V(:,nt+1)),'-g')
% xlabel ('S')
% ylabel ('Error value')
% title ('Error')
toc
time(k2) = toc;
N = ns; M = nt;
V2(N,M) = 0;
%%tm(M)= 0;
for i=1:N+1
    for j = 1: M+1
        %%tm(j) = abs(T-t(j));
        [call,put] = blsprice((i*1/N)-1/N, K, r,(j*1/M)-1/M, sigma, 0);
        V2(i,j) = put;
    end
end
error3(k2) = norm(V2-V,'fro');
error2(k2) = norm(V2-V,'fro')/norm(V2,'fro');
error(k2) = norm(V2-V,'fro')/(M*N);
end
figure;
%plot(error3,'k');
%plot(error2,'k');
plot(error,'k');
hold on;
```

```
%-----------------------------------------------------------------------------------------------

clear all;
q=35;
time(200) = 0;
error(200) = 0;
h=3;
for k = 50-h : 1 : 60-h
tic;
sigma = 0.05; r = 0.2; K = 1;
X = 1; T = 1.1;
N = k+h; M = k+h;
x(N) = X; x(1) = 0;
t(M) = T; t(1) = 0;
dx = X/(N-1); dt = T/(M-1);
for i = 2:N
    x(i) = x(i-1) + dx;
end
for i = 2:M
    t(i) = t(i-1) + dt;
end
V3(N,M) = 0;
for j = 1:M
    for i = 1:N
        V3(i,j) = 0;
    end
end
for i = 2:N
    V3(i,M) = max(-x(i) + K,0);
end
% for i = 1:M-1
%       V3(N,i) = exp(-(-t(i)+N)*M);
% end
for i = 1:M
    V3(1,i) = K*exp(-r*(T-t(i)));
end
l(N) = 0;
l(1) = 2*(sigma^2 - r)/(sigma^2*x(2));
for i = 2:N
    l(i) = 2*(sigma^2 - r)/(sigma^2*x(i)) ;
end
e(N)=0;
for i = 1:N
    e(i) = -(sigma^2*x(i)^2)/2 ;
end
p(N-1) = 0; alpha(N-1) = 0;
beta(N-1) =0; gamma(N-1) =0;
for i = 1:N-1
    p(i) = dx*(l(i)/2 + l(i+1)/2);
    alpha(i) = ((N-1)/X)*b((-1)*p(i))*(2/(l(i)+l(i+1)))*(w((-1)*p(i))*l(i) +
w(p(i))*l(i+1))*(w((-1)*p(i))*e(i) + w(p(i))*e(i+1));
    beta(i) = ((N-1)/X)*b(p(i))*(2/(l(i)+l(i+1)))*(w((-1)*p(i))*l(i) +
w(p(i))*l(i+1))*(w((-1)*p(i))*e(i) + w(p(i))*e(i+1));
    gamma(i) = max((0.5 - w(p(i))) , 0);
```

```
end
A(N-2,N-2) = alpha(N-1);
for i = 1:N-3
    A(i,i) = alpha(i+1) + beta(i+1);
    A(i,i+1) = (-1) * beta(i+1);
    A(i+1,i) = (-1) * alpha(i+1);
end
B(N-2,N-2) = 1 - gamma(N-1);
for i = 1:N-3
    B(i,i) = 1 - gamma(i+1);
    B(i+1,i) = gamma(i+1) ;
end
B = B*dx;
b_vector(N-2,M) = 0;
for i = 1:M
    b_vector(1,i) = (-dx*gamma(1)*(K*r*exp(-r*(T-t(i)))) + alpha(1)*V3(1,i) +
dx*gamma(1)*(-(sigma^2 - r)*V3(1,i)));
    %b_vector((N-2),i) = beta(N-1)*V3(N,i);
end
theta = 0.5;
A_matrix = B - dt*(1-theta)*A + (r-sigma^2)*dt*(1-theta)*B;
A_matrix_inv = inv(A_matrix);
B_matrix(N-2,M-2) = 0;
source(N-2,1) = 0;
for i = (M-1) : -1 : 1
    source(:,1) = -(sigma^2 - r)*V3(2:N-1,i+1);
    B_matrix(:,i) = -dt*((1-theta)*b_vector(:,i) + theta*b_vector(:,i+1))
-dt*theta*B*source(:,1) + B*V3(2:(N-1),i+1) + dt*theta*A*V3(2:(N-1),i+1);
    V3(2:(N-1), i) = A_matrix\B_matrix(:,i);
end
toc;
time(k) = toc;
V2(k,M) = 0;
tm(M)= 0;
for i=1:k
    for j = 1: M
        tm(j) = abs(T-t(j));
        [call,put] = blsprice(x(i), K, r, tm(j), sigma, 0);
        V2(i,j) = put;
    end
end
V5(1:k,1:M) = V3(1:k,1:M);
error3(k) = norm(V2-V5,'fro');
error2(k) = norm(V2-V5,'fro')/norm(V2,'fro');
error(k+h) = norm(V2-V5,'fro')/(M*k);
end
%plot(error3,'r');
%plot(error2,'g');
plot(error,'r');


%-----------------------------------------------------------------------------------------
clear all
q=35;
time(200) = 0;
```

```
error(200) = 0;
for k = 50 : 1 : 60
tic
% parameters
r=0.2; % interest rate
sigma=0.05; % volatility
K=1; % strike price
Smax=1; % maximal asset price
T=1.1; % maturity time

% grid
ns=k; % space points
nt=k; % time points
dS=Smax/ns; % space step, l(i)
dtau=T/nt; % time step
S=0:dS:Smax; % asset prices grid
tau=0:dtau:T; % time grid

%option
V4=zeros(ns+1,nt+1); %array for option

%PUT
for i=1:ns+1
V4(i,1)=max(K-S(i),0); %payoff t=T or tau=0
end

for i=1:nt+1
V4(1,i)=K*exp(-r*tau(i)); %boundary condition S=0
V4(ns+1,i)=0; %boundary condition S=Smax
end

% coefficients
for i=1:ns-1
ad(i) = -0.25*dtau*(sigma*sigma*S(i)*S(i) - r*S(i));
ac(i) = 1 + 0.5*dtau*(sigma*sigma*S(i)*S(i)+r);
au(i) = -0.25*dtau*(sigma*sigma*S(i)*S(i) + r*S(i));
bd(i) = 0.25*dtau*(sigma*sigma*S(i)*S(i) - r*S(i));
bc(i) = 1 - 0.5*dtau*(sigma*sigma*S(i)*S(i)+r);
bu(i) = 0.25*dtau*(sigma*sigma*S(i)*S(i) + r*S(i));
end

% matrixes A and B
A=zeros(ns-1,ns-1);
B=zeros(ns-1,ns-1);
A(ns-1,ns-1) = ac(ns-1);
B(ns-1,ns-1) = bc(ns-1);
for i=1:ns-2
A(i,i) = ac(i);
A(i+1,i) = ad(i+1);
A(i,i+1) = au(i);
B(i,i) = bc(i);
B(i+1,i) = bd(i+1);
B(i,i+1) = bu(i);
end

% method
```

```
for j=2:nt+1
V4(2:ns,j)=A\B*V4(2:ns,j-1);
end

% EXACT SOLUTION
Vex=zeros(ns+1,1);
coeff = r+0.5*sigma^2;
for i=1:ns+1
d1=(log(S(i)/K)+coeff*T)/(sigma*sqrt(T));
d2=d1-sigma*sqrt(T);
D1=normcdf(d1,0,1);
D2=normcdf(d2,0,1);
Vex(i)=K*exp(-r*T)*(1-D2)-S(i)*(1-D1);
end
% end EXACT SOLUTION

toc
time(k) = toc;
N = ns; M = nt;
V2(N,M) = 0;
%%tm(M)= 0;
for i=1:N+1
    for j = 1: M+1
        %%tm(j) = abs(T-t(j));
        [call,put] = blsprice((i*1/N)-1/N, K, r,(j*1/M)-1/M, sigma, 0);
        V2(i,j) = put;
    end
end
error3(k) = norm(V2-V4,'fro');
error2(k) = norm(V2-V4,'fro')/norm(V2,'fro');
error(k) = norm(V2-V4,'fro')/(M*N);
end
%plot(error3,'y');
%plot(error2,'m');
plot(error,'b');
%---------------------------------------------------------------------------------------
fileID = fopen('example.txt','r');
[A11,count] = fscanf(fileID, ['%f'])
fclose(fileID);
for g=50:55
r=0.2; % interest rate
sigma=0.05; % volatility
K=1; % strike price
Smax=1; % maximal asset price
T=1.1; % maturity time
N = g; M = g;
V2(N,M) = 0;
%%tm(M)= 0;
for i=1:N+1
    for j = 1: M+1
        %%tm(j) = abs(T-t(j));
        [call,put] = blsprice((i*1/N)-1/N, K, r,(j*1/M)-1/M, sigma, 0);
        V2(i,j) = put;
    end
end
sum=0;
```

```
for k=50:g-1
    sum = sum + k^2;
end
for k2 = 1 : g
    for k3 = 1: g
        V4(k2,k3) = A11(sum+g*k2+k3);
    end
end
error3(g) = norm(V2-V4,'fro');
error2(g) = norm(V2-V4,'fro')/norm(V2,'fro');
error(g) = norm(V2-V4,'fro')/(M*N);
end
plot(error,'g');
% k=1;
% for i=1:16870
%
%     if A11(i)>49
%
%         j=i+1;
%         q=1;
%         for j=i+1:i+A11(i)
%             B11(k,q) = A11(j);
%             q=q+1;
%         end
%         k=k+1;
%     end
% end



legend('Finite Volume Method','Complete Flux Scheme','Crank-Nicolson
method','Finite Element Method')
hold off;
```

Grid	cfs(norm(V-V’)	cfs(norm(V-V’)) (norm(V))	cfs(norm(V-V’)) N*M	fvm(norm(V-V’))	fvm(norm(V-V’)) (norm(V))	fvm(norm(V-V’)) N*M
25x25	0.2758	0.0217	4.4135e-04	0.0917	0.0070	1.4671e-04
50x50	0.2698	0.0107	1.0790e-04	0.0782	0.0030	3.1296e-05
75x75	0.3319	0.0088	5.8997e-05	0.0649	0.0017	1.1535e-05
100x100	0.6125	0.0122	6.1248e-05	0.0552	0.0011	5.5166e-06
200x200	4.0812	0.0407	1.0203e-04	0.0355	3.5246e-04	8.8824e-07
400x400	20.5303	0.1024	1.2831e-04	NaN	NaN	NaN
1000x1000	125.1218	0.2498	1.2512e-04	NaN	NaN	NaN

Grid	cfs(norm(V-V’)	cfs(norm(V-V’)) (norm(V))	cfs(norm(V-V’)) N*M	cns(norm(V-V’))	cns(norm(V-V’)) (norm(V))	cns(norm(V-V’)) N*M
25x25	0.2758	0.0217	4.4135e-04	1.2989	0.0985	0.0021
50x50	0.2698	0.0107	1.0790e-04	2.6140	0.1017	0.0010
75x75	0.3319	0.0088	5.8997e-05	3.9299	0.1028	6.9865e-04
100x100	0.6125	0.0122	6.1248e-05	5.2461	0.1034	5.2461e-04
200x200	4.0812	0.0407	1.0203e-04	10.5111	0.1043	2.6278e-04
400x400	20.5303	0.1024	1.2831e-04	21.0415	0.1047	1.3151e-04
1000x1000	125.1218	0.2498	1.2512e-04	189.9293	0.1178	1.7467e-04

Grid	Cfs time	Fvm time	Cns time
25x25	0.088062	0.112905	0.027845
50x50	0.091488	0.125092	0.034292
75x75	0.096351	0.145529	0.040988
100x100	0.105750	0.171671	0.053278
200x200	0.188637	0.407237	0.243788
400x400	1.072633	3.000069	1.988612
1000x1000	25.667480	29.17166	61.14708

Grid	Cfs time	Monte carlo simulation
25x25	0.088062	0.672637
50x50	0.091488	0.933636
75x75	0.096351	7.328784
100x100	0.105750	29.874657
200x200	0.188637	257.87467
400x400	1.072633	1349.3465
1000x1000	25.667480	>25 min