Université de Rouen
M2 AIMAF
Année 2012-2013

MNF
Programmes Scilab

1. Changement de variables (compliqué)

Voici tout d'abord les programmes Scilab
- équation de la chaleur, schéma explicite
- équation de la chaleur, schéma de Crank-Nicolson
- call, schéma de Crank-Nicolson, condition sur le bord indépendante du temps
- call, schéma de Crank-Nicolson, condition sur le bord dépendante du temps (donne *a priori* une meilleure approximation)
- put, schéma de Cranck-Nicolson (programme non vérifié)
- call, formule de Black et Scholes (permet de vérifier les schémas Crank-Nicolson)

```
function [tpt,xpt,MU]=heat_expl(T,a,b,f,u0,alp,bet,N,M)
 // schéma explicite
  k=T/(M+1)
  h=(b-a)/(N+1)
  disp(k/h^2)
  tpt=linspace(0,T,M+2)
  xpt=linspace(a,b,N+2)
  MU=zeros(M+2,N+2)
  A=2*diag(ones(N,1))-diag(ones(N-1,1),1)-diag(ones(N-1,1),-1)
  G=eye(N,N)-k*A/h^2
  U=u0(xpt(2:N+1))'
  MU(1,:)=[alp,U',bet]
  for i=1:M+1
    F=f(tpt(i),xpt(2:N+1))'
    F(1)=alp/h^2+F(1)
    F(N)=bet/h^2+F(N)
    U=G*U+k*F
    MU(i+1,:)=[alp,U',bet]
  end
endfunction

function [tpt,xpt,MU]=heat_crnic(T,a,b,f,u0,alp,bet,N,M)
  // schéma de Crank-Nicolson
  k=T/(M+1)
  h=(b-a)/(N+1)
  disp(k/h^2)
  tpt=linspace(0,T,M+2)
  xpt=linspace(a,b,N+2)
  MU=zeros(M+2,N+2)
  A=2*diag(ones(N,1))-diag(ones(N-1,1),1)-diag(ones(N-1,1),-1)
  G=eye(N,N)-k*A/(2*h^2)
  G1=eye(N,N)+k*A/(2*h^2)
  U=u0(xpt(2:N+1))'
  MU(1,:)=[alp,U',bet]
  for i=1:M+1
    F=(f(tpt(i),xpt(2:N+1))' + f(tpt(i+1),xpt(2:N+1))')/2
    F(1)=alp/h^2+F(1)
    F(N)=bet/h^2+F(N)
    // résout G1*U(i+1)=G*U(i)+k*F
    U=linsolve(G1,-G*U-k*F)
```

```
    MU(i+1,:)=[alp,U',bet]
  end
endfunction



function [Spt,Vpt]=callbs_crnic(T,a,b,r,sig,K,N,M)
// call, équation de Black et Scholes avec changement de variable "complet"
// schéma de Crank-Nicolson sur le bord approximation 1er ordre,
// indépendante du temps
  T1=T*sig^2/2
  k=T1/(M+1)
  h=(b-a)/(N+1)
  q=2*r/sig^2
  tpt=linspace(0,T1,M+2)
  xpt=linspace(a,b,N+2)
  A=2*diag(ones(N,1))-diag(ones(N-1,1),1)-diag(ones(N-1,1),-1)
  G=eye(N,N)-k*A/(2*h^2)
  G1=eye(N,N)+k*A/(2*h^2)
  U=exp((q-1)*xpt(2:N+1)/2)'.*max(K*exp(xpt(2:N+1))-K,0)'/K
  for i=1:M+1
    F=zeros(N,1)
    F(1)=0
    F(N)=(1/(2*h^2))*(exp((q+1)*b/2+(q+1)^2*k*i/4)+exp((q+1)*b/2+(q+1)^2*k*(i+1)/4))
    U=linsolve(G1,-G*U-k*F)
  end
  Spt=K*exp(xpt(2:N+1))'
  Vpt=K*exp(-(q-1)*xpt(2:N+1)/2-(q+1)^2/4*T*sig^2/2)'.*U
endfunction


function [Spt,Vpt]=callbs_crnic2(T,a,b,r,sig,K,N,M)
// call, équation de Black et Scholes avec changement de variable "complet"
// schéma de Crank-Nicolson, sur le bord approximation ordre supérieur
// dépendante du temps
  T1=T*sig^2/2
  k=T1/(M+1)
  h=(b-a)/(N+1)
  q=2*r/sig^2
  tpt=linspace(0,T1,M+2)
  xpt=linspace(a,b,N+2)
  A=2*diag(ones(N,1))-diag(ones(N-1,1),1)-diag(ones(N-1,1),-1)
  G=eye(N,N)-k*A/(2*h^2)
  G1=eye(N,N)+k*A/(2*h^2)
  U=exp((q-1)*xpt(2:N+1)/2)'.*max(K*exp(xpt(2:N+1))-K,0)'/K
  for i=1:M+1
    F=zeros(N,1)
    F(1)=0
    f1=(exp(b)-exp(-r*k*i/sig^2))*exp((q-1)*b/2+(q+1)^2*k*i/4)
    f2=(exp(b)-exp(-r*k*(i+1)/sig^2))*exp((q-1)*b/2+(q+1)^2*k*(i+1)/4)
    F(N)=(1/(2*h^2))*(f1+f2)
```

```
    U=linsolve(G1,-G*U-k*F)
  end
  Spt=K*exp(xpt(2:N+1))'
  Vpt=K*exp(-(q-1)*xpt(2:N+1)/2-(q+1)^2/4*T*sig^2/2)'.*U
endfunction

function [Spt,Vpt]=putbs_crnic(T,a,b,r,sig,K,N,M)
// put, équation de Black et Scholes avec changement de variable "complet"
// schéma de Crank-Nicolson  et sur le bord approximation ordre supérieur
// dépendante du temps
// non vérifié !
  T1=T*sig^2/2
  k=T1/(M+1)
  h=(b-a)/(N+1)
  q=2*r/sig^2
  tpt=linspace(0,T1,M+2)
  xpt=linspace(a,b,N+2)
  A=2*diag(ones(N,1))-diag(ones(N-1,1),1)-diag(ones(N-1,1),-1)
  G=eye(N,N)-k*A/(2*h^2)
  G1=eye(N,N)+k*A/(2*h^2)
  // donnnée intiale après changement de variable
  U=exp((q-1)*xpt(2:N+1)/2)'.*max(K-K*exp(xpt(2:N+1)),0)'/K
  for i=1:M+1
    F=zeros(N,1)
    F(1)=1/2*(exp((q-1)*a/2+(q-1)^2*k*i/4)+exp((q-1)*a/2+(q-1)^2*k*(i+1)/4))/h^2
    F(N)=0
    U=linsolve(G1,-G*U-k*F)
  end
  Spt=K*exp(xpt(2:N+1))'
  Vpt=K*exp(-(q-1)*xpt(2:N+1)/2-(q+1)^2/4*T*sig^2/2)'.*U
endfunction

function V=callbs_formule(S,K,T,r,sig)
  // Formule de Black et Scholes, call
  d1=(log(S/K)+(r+sig^2/2)*T)/(sig*T^.5)
  d2=d1-sig*T^.5
  D1=cdfnor("PQ",d1,0,1)
  D2=cdfnor("PQ",d2,0,1)
  V=S*D1-K*exp(-T*r)*D2
endfunction
```

Les tests réalisés durant le cours sont basés sur les commandes suivantes (dans l'éditeur on peut exécuter la partie du code sélectionnée)

```
// exemple 1
deff("y=f(t,x)", "y=-1+x-x")
deff("y=u0(x)", "y=x.*(x-1)/2")
[tpt,xpt,MU]=heat_expl(1,0,1,f,u0,0,0,10,1000)
plot3d(tpt,xpt,MU)

[tpt,xpt,MU]=heat_expl(1,0,1,f,u0,0,0,10,194)
plot2d(xpt,MU(140,:))
```

```
clf()
plot2d(xpt,MU(10,:))


// exemple 2
// u(t,x)=sin(t).* sin (%pi*x)
// nulle en t=0
// second membre qui se calcule
deff("y=f(t,x)", "y=cos(t).*sin(%pi*x)+%pi^2*sin(t).*sin(%pi*x)")
deff("y=u0(x)", "y=0-x+x")
[tpt,xpt,MU]=heat_crnic(4,0,1,f,u0,0,0,100,100)
plot3d(tpt,xpt,MU)
clf()
plot2d(xpt,MU(11,:))
plot2d(xpt,sin(4*10/100)*sin(%pi*xpt),style=-1)

//exemple 3
// effet régularisant + décroissant second membre nul
// u0(x)=x si x>1, 0 sinon
deff("y=u0(x)", "y=(x>1).*x")
deff("y=f(t,x)", "y=x-x") // f=0 vectorisé
[tpt,xpt,MU]=heat_crnic(1,0,2,f,u0,0,0,100,200)
plot3d(tpt,xpt,MU)

//exemple 4
// condition non nulle sur le bord
deff("y=f(t,x)", "y=%pi^2*cos(%pi*x)")
deff("y=u0(x)", "y=cos(%pi*x)")
[tpt,xpt,MU]=heat_crnic(1,0,1,f,u0,1,-1,10,50)
plot3d(tpt,xpt,MU)


/// putbs_crnic(1,-2,1,.06,.3,10,100,100)
// T=1, K=10, r=.06, sigma=.3
// observation sur le bord
[Spt,Vpt]=putbs_crnic(1,-2,1,.06,.3,10,100,100)


// call , comparaison avec les
// formules de B& S
[Spt,Vpt]=callbs_crnic(1,-1,1,.25,.6,10,100,100)
plot(Spt,Vpt,'linewidth',3)
plot(Spt(20),callbs_formule(Spt(20),10,1,.25,.6),'k+','MarkSize',10)
plot(Spt(40),callbs_formule(Spt(40),10,1,.25,.6),'k+','MarkSize',10)
plot(Spt(60),callbs_formule(Spt(60),10,1,.25,.6),'k+','MarkSize',10)
plot(Spt(80),callbs_formule(Spt(80),10,1,.25,.6),'k+','MarkSize',10)
plot(Spt(100),callbs_formule(Spt(100),10,1,.25,.6),'k+','MarkSize',10)
[Spt,Vpt]=callbs_crnic(1,-1.5,1.5,.25,.6,10,100,100)
clf()
plot(Spt,Vpt,'linewidth',3)
plot(Spt(20),callbs_formule(Spt(20),10,1,.25,.6),'k+','MarkSize',10)
```

```
plot(Spt(40),callbs_formule(Spt(40),10,1,.25,.6),'k+','MarkSize',10)
plot(Spt(60),callbs_formule(Spt(60),10,1,.25,.6),'k+','MarkSize',10)
plot(Spt(80),callbs_formule(Spt(80),10,1,.25,.6),'k+','MarkSize',10)
plot(Spt(100),callbs_formule(Spt(100),10,1,.25,.6),'k+','MarkSize',10)

// call conditions aux limites plus précises
// et formules de B et S

clf()
[Spt,Vpt]=callbs_crnic(1,-1,1,.25,.6,10,100,100)
[Spt2,Vpt2]=callbs_crnic2(1,-1,1,.25,.6,10,100,100)
plot(Spt,Vpt,'linewidth',3)
plot(Spt2,Vpt2,'k','linewidth',3)
plot(Spt(20),callbs_formule(Spt(20),10,1,.25,.6),'k+','MarkSize',10)
plot(Spt(40),callbs_formule(Spt(40),10,1,.25,.6),'k+','MarkSize',10)
plot(Spt(60),callbs_formule(Spt(60),10,1,.25,.6),'k+','MarkSize',10)
plot(Spt(80),callbs_formule(Spt(80),10,1,.25,.6),'k+','MarkSize',10)
plot(Spt(100),callbs_formule(Spt(100),10,1,.25,.6),'k+','MarkSize',10)

clf()
[Spt,Vpt]=callbs_crnic(1,-1.5,1.5,.25,.6,10,100,100)
[Spt2,Vpt2]=callbs_crnic2(1,-1.5,1.5,.25,.6,10,100,100)
plot(Spt,Vpt,'linewidth',3)
plot(Spt2,Vpt2,'k','linewidth',3)
plot(Spt(20),callbs_formule(Spt(20),10,1,.25,.6),'k+','MarkSize',10)
plot(Spt(40),callbs_formule(Spt(40),10,1,.25,.6),'k+','MarkSize',10)
plot(Spt(60),callbs_formule(Spt(60),10,1,.25,.6),'k+','MarkSize',10)
plot(Spt(80),callbs_formule(Spt(80),10,1,.25,.6),'k+','MarkSize',10)
plot(Spt(100),callbs_formule(Spt(100),10,1,.25,.6),'k+','MarkSize',10)


clf()
[Spt,Vpt]=callbs_crnic(1,-1.5,2,.25,.6,10,100,100)
[Spt2,Vpt2]=callbs_crnic2(1,-1.5,2,.25,.6,10,100,100)
plot(Spt,Vpt,'linewidth',3)
plot(Spt2,Vpt2,'k','linewidth',3)
plot(Spt(20),callbs_formule(Spt(20),10,1,.25,.6),'k+','MarkSize',10)
plot(Spt(40),callbs_formule(Spt(40),10,1,.25,.6),'k+','MarkSize',10)
plot(Spt(60),callbs_formule(Spt(60),10,1,.25,.6),'k+','MarkSize',10)
plot(Spt(80),callbs_formule(Spt(80),10,1,.25,.6),'k+','MarkSize',10)
plot(Spt(100),callbs_formule(Spt(100),10,1,.25,.6),'k+','MarkSize',10)
```

2. Variable logarithmique et inversion du temps

Voici tout d'abord les programmes (uniquement le call) :

- `call_crnic1` : call, schéma de Crank-Nicolson, condition sur le bord indépendante du temps
- `call_crnic2` : call, schéma de Crank-Nicolson, condition sur le bord dépendante du temps (donne *a priori* une meilleure approximation)
- `call_crnic3` : call, schéma de Cranck-Nicolson, condition Dirichlet homogène ($u = 0$ sur le bord) peu réaliste
- `callbs_formule` : call, formule de Black et Scholes (permet de vérifier les schémas Crank-Nicolson)

Les valeurs sur le bord n'ont pas été ajoutées ; il faudrait le faire.

```
function [Spt,Vpt]=call_crnic1(T,a,b,r,sig,K,N,M)
  // inversion du temps et variable logarithmique
  // sur le bord approximation 1er ordre indépendant du temps
  k=T/(M+1)
  h=(b-a)/(N+1)
  q=2*r/sig^2
  tpt=linspace(0,T,M+2)
  xpt=linspace(a,b,N+2)
  A=(sig^2/h^2+r)*diag(ones(N,1))+((r-sig^2/2)/(2*h)-sig^2/(2*h^2))*diag(ones(N-1,1),-1)
  A=A+(-(r-sig^2/2)/(2*h)-sig^2/(2*h^2))*diag(ones(N-1,1),1)
  G=eye(N,N)-k*A/2
  G1=eye(N,N)+k*A/2
  U=max(exp(xpt(2:N+1))-K,0)'
  for i=1:M+1
    F=zeros(N,1)
    F(1)=0
    F(N)=exp(b)*((r-sig^2/2)/(2*h)+sig^2/(2*h^2))
    U=linsolve(G1,-G*U-k*F)
  end
  Spt=exp(xpt(2:N+1))'
  Vpt=U
endfunction


function [Spt,Vpt]=call_crnic2(T,a,b,r,sig,K,N,M)
  // inversion du temps et variable logarithmique
  // sur le bord approximation dépendante du temps
  k=T/(M+1)
  h=(b-a)/(N+1)
  q=2*r/sig^2
  tpt=linspace(0,T,M+2)
  xpt=linspace(a,b,N+2)
  A=(sig^2/h^2+r)*diag(ones(N,1))+((r-sig^2/2)/(2*h)-sig^2/(2*h^2))*diag(ones(N-1,1),-1)
  A=A+(-(r-sig^2/2)/(2*h)-sig^2/(2*h^2))*diag(ones(N-1,1),1)
  G=eye(N,N)-k*A/2
  G1=eye(N,N)+k*A/2
  U=max(exp(xpt(2:N+1))-K,0)'
  for i=1:M+1
    F=zeros(N,1)
    F(1)=0
    F(N)=(exp(b)-K/2*exp(-r*k*i)-K/2*exp(-r*k*(i+1)))*((r-sig^2/2)/(2*h)+sig^2/(2*h^2))
    U=linsolve(G1,-G*U-k*F)
  end
  Spt=exp(xpt(2:N+1))'
  Vpt=U
endfunction

function [Spt,Vpt]=call_crnic3(T,a,b,r,sig,K,N,M)
  // inversion du temps et variable logarithmique
```

```
  // sur le bord dirichlet homogène
  // zero, comme dans Lamberton-Lapeyre
  k=T/(M+1)
  h=(b-a)/(N+1)
  q=2*r/sig^2
  tpt=linspace(0,T,M+2)
  xpt=linspace(a,b,N+2)
  A=(sig^2/h^2+r)*diag(ones(N,1))+((r-sig^2/2)/(2*h)-sig^2/(2*h^2))*diag(ones(N-1,1),-1)
  A=A+(-(r-sig^2/2)/(2*h)-sig^2/(2*h^2))*diag(ones(N-1,1),1)
  G=eye(N,N)-k*A/2
  G1=eye(N,N)+k*A/2
  U=max(exp(xpt(2:N+1))-K,0)'
  for i=1:M+1
    U=linsolve(G1,-G*U)
  end
  Spt=exp(xpt(2:N+1))'
  Vpt=U
endfunction

function V=callbs_formule(S,K,T,r,sig)
  d1=(log(S/K)+(r+sig^2/2)*T)/(sig*T^.5)
  d2=d1-sig*T^.5
  D1=cdfnor("PQ",d1,0,1)
  D2=cdfnor("PQ",d2,0,1)
  V=S*D1-K*exp(-T*r)*D2
endfunction
```

Voici le programme Scilab qui permet de tester trois possibles conditions aux limites.

```
[Spt1,Vpt1]=call_crnic1(1,-2,3,.25,.6,10,100,100);
[Spt2,Vpt2]=call_crnic2(1,-2,3,.25,.6,10,100,100);
[Spt3,Vpt3]=call_crnic3(1,-2,3,.25,.6,10,100,100);
clf()
plot(Spt3,Vpt3, Spt2,Vpt2, Spt1,  Vpt1,'linewidth',3);


[Spt3,Vpt3]=call_crnic1(1,-2,3,.25,.6,10,100,100);
clf()
plot(Spt3,Vpt3,'r','linewidth',2)
for i=10:5:100
plot(Spt3(i),callbs_formule(Spt3(i),10,1,.25,.6),'k+','MarkSize',10)
end

[Spt3,Vpt3]=call_crnic3(1,-2,3,.25,.6,10,100,100);
clf()
plot(Spt3,Vpt3,'r','linewidth',2)
for i=10:5:100
plot(Spt3(i),callbs_formule(Spt3(i),10,1,.25,.6),'k+','MarkSize',10)
end


[Spt3,Vpt3]=call_crnic2(1,-2,3,.25,.6,10,100,100);
```

```
clf()
plot(Spt3,Vpt3,'r','linewidth',2)
for i=10:5:100
plot(Spt3(i),callbs_formule(Spt3(i),10,1,.25,.6),'k+','MarkSize',10)
end
```

```
[Spt1,Vpt1]=call_crnic1(1,-4,6,.25,.6,10,100,100);
[Spt2,Vpt2]=call_crnic2(1,-4,6,.25,.6,10,100,100);
[Spt3,Vpt3]=call_crnic3(1,-4,6,.25,.6,10,100,100);
clf()
plot(Spt3,Vpt3, Spt2,Vpt2, Spt1,  Vpt1,'linewidth',3);
```

```
[Spt3,Vpt3]=call_crnic1(1,-4,6,.25,.6,10,100,100);
clf()
plot(Spt3,Vpt3,'r','linewidth',2)
for i=10:5:100
plot(Spt3(i),callbs_formule(Spt3(i),10,1,.25,.6),'k+','MarkSize',10)
end

[Spt3,Vpt3]=call_crnic3(1,-4,6,.25,.6,10,100,100);
clf()
//plot(Spt3,Vpt3,'r','linewidth',2)
for i=10:5:50
plot(Spt3(i),Vpt3(i)-callbs_formule(Spt3(i),10,1,.25,.6),'k+','MarkSize',10)
end
```

```
[Spt3,Vpt3]=call_crnic2(1,-4,6,.25,.6,10,100,100);
clf()
//plot(Spt3,Vpt3,'r','linewidth',2)
for i=10:5:100
plot(Spt3(i),Vpt3(i)-callbs_formule(Spt3(i),10,1,.25,.6),'k+','MarkSize',10)
end
```

Quelques explications. Tout d'abord les programmes `call_crnic1` et `call_crnic2` de la séance du 23 février n'étaient pas corrects ! Il y avait un problème au niveau de la condition sur le bord en b (si les valeurs aux deux extrémités avaient été ajoutées, cette erreur aurait été probablement détectée avant ou pendant la séance).

Rappelons que $u(\tau,x) = V(\exp(x), T-\tau)$ est solution de

$$\frac{\partial u}{\partial \tau} - \frac{1}{2}\sigma^2 \frac{\partial^2 u}{\partial x^2} - \big(r - \frac{1}{2}\sigma^2\big)\frac{\partial u}{\partial x} + r\,u = 0, \tag{1}$$

avec la condition initiale « fonction payoff ». Pour approchée cette solution on « tronque » le domaine $]-\infty,+\infty[$ en $[a,b]$ et on effectue un schéma aux différences finies. Il est donc indispensable d'ajouter des conditions aux limites (en a et b).

Prenons le cas d'un call (européen). En a ($a<0$) ce sera $u(\tau,a)=0$. En b selon l'approximation qui vient de la parité call-put au moins 3 choix sont possibles

- $u(\tau,b) = \exp(b)$, programme 1 ou `call_crnic1`
- $u(\tau,b) = \exp(b) - K\exp(-r\tau)$, programme 2 ou `call_crnic2`

– $u(\tau, b) = 0$, programme 3 ou `call_crnic3`.

Du point de vue théorique les programmes 1, 2 et 3 calculent respectivement une approximation des problèmes

$$\text{Prog 1}\begin{cases} \dfrac{\partial u}{\partial \tau} - \dfrac{1}{2}\sigma^2 \dfrac{\partial^2 u}{\partial x^2} - \left(r - \dfrac{1}{2}\sigma^2\right)\dfrac{\partial u}{\partial x} + r\,u = 0 & (\tau, x) \in [0, T] \times [a, b] \\ u(\tau, a) = 0 \quad u(\tau, b) = \exp(b) & \tau \in [0, T] \\ u(\tau = 0, x) = \text{fonction payoff} & x \in [a, b] \end{cases}$$

$$\text{Prog 2}\begin{cases} \dfrac{\partial u}{\partial \tau} - \dfrac{1}{2}\sigma^2 \dfrac{\partial^2 u}{\partial x^2} - \left(r - \dfrac{1}{2}\sigma^2\right)\dfrac{\partial u}{\partial x} + r\,u = 0 & (\tau, x) \in [0, T] \times [a, b] \\ u(\tau, a) = 0 \quad u(\tau, b) = \exp(b) - K\exp(-r\tau) & \tau \in [0, T] \\ u(\tau = 0, x) = \text{fonction payoff} & x \in [a, b] \end{cases}$$

$$\text{Prog 3}\begin{cases} \dfrac{\partial u}{\partial \tau} - \dfrac{1}{2}\sigma^2 \dfrac{\partial^2 u}{\partial x^2} - \left(r - \dfrac{1}{2}\sigma^2\right)\dfrac{\partial u}{\partial x} + r\,u = 0 & (\tau, x) \in [0, T] \times [a, b] \\ u(\tau, a) = 0 \quad u(\tau, b) = 0 & \tau \in [0, T] \\ u(\tau = 0, x) = \text{fonction payoff} & x \in [a, b] \end{cases}$$

Il est raisonnable de penser que les solutions de (1) et des trois équations précédentes sont différentes. Du point de vue théorique il est possible (mais difficile) d'étudier la différence entre chacune d'entre elles et de donner une estimation de l'erreur commise. Nous nous contenterons d'un test sur deux « fenêtres » $[a, b]$ différentes.

Pour un call, $T = 1$, $r = .25$, $\sigma = 0.6$, $K = 10$ et une discrétisation en temps et en espace identiques :$N = M = 100$.

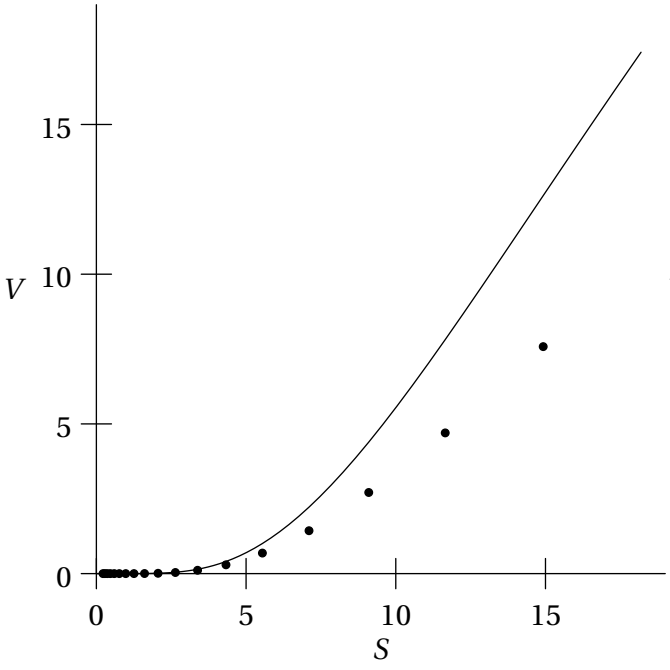


FIGURE 1. Programme 1, fenêtre $[-2, 3]$

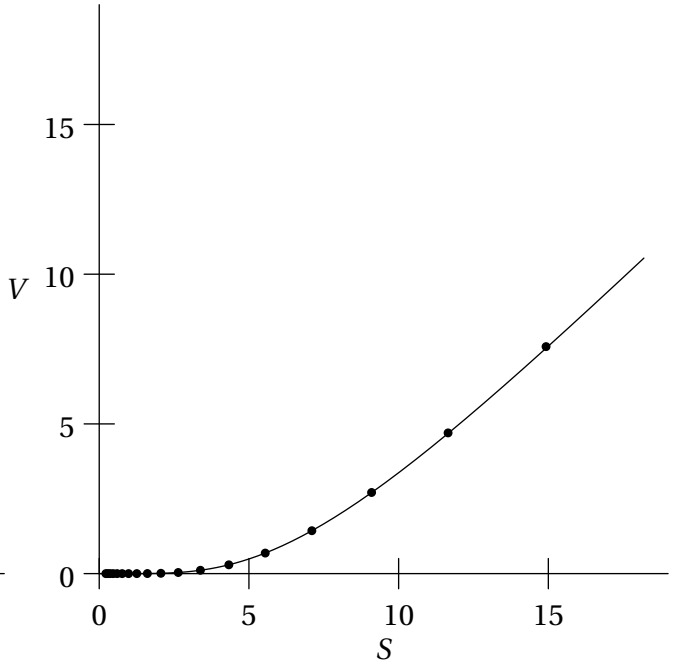


FIGURE 2. Programme 2, fenêtre $[-2, 3]$

Test 1. On prend $a = -2$ et $b = 3$. Après changement de variables, nous avons (approximativement) $0.135 \leq S \leq 20$. Le strike valant 10, on pourrait (devrait) prendre une plus grande valeur. L'important est

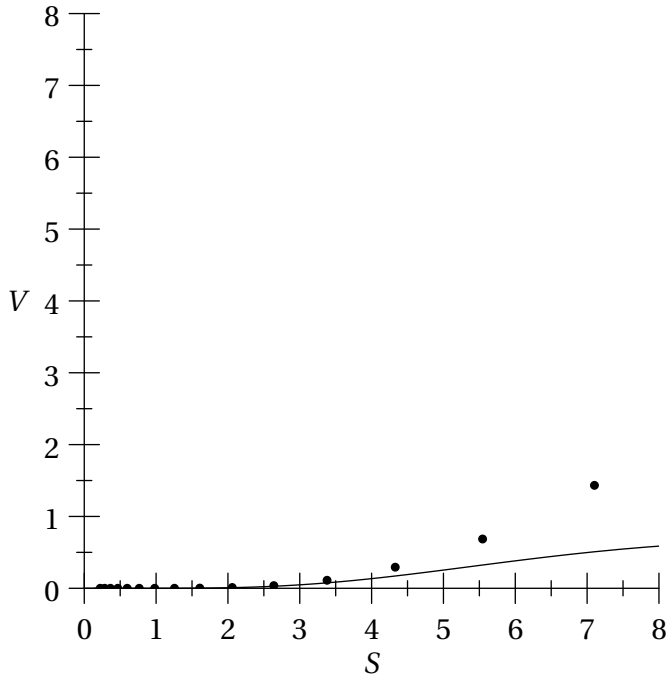


FIGURE 3. Programme 3, fenêtre $[-2,3]$

de voir la différence dans les figures 1, 2 et 3. Les points sur les figures correspondent aux valeurs données par les formules de Black et Scholes, l'idéal est donc que ces points soient sur la courbe tracée. Les programmes 1 et 3 sont catastrophiques tandis que le programme 2 se comporte très bien.

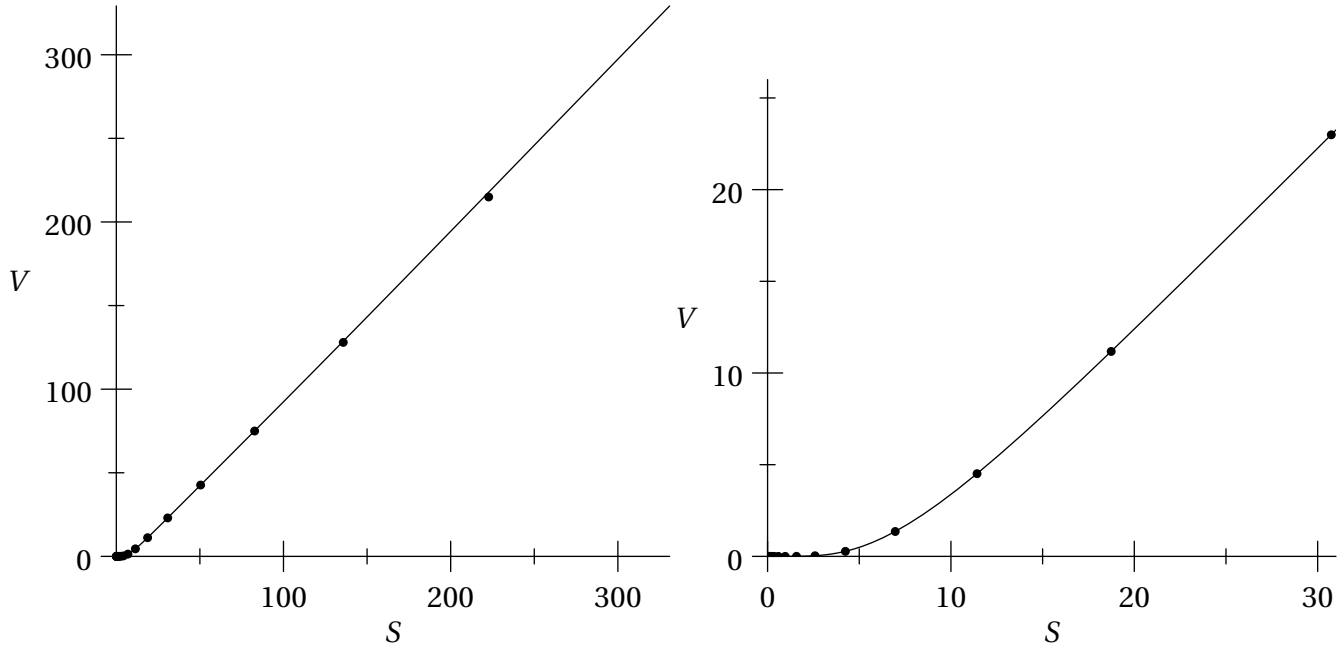


FIGURE 4. Programme 1, fenêtre $[-4,6]$

Test 2. On prend $a = -4$ et $b = 6$. Après changement de variables, nous avons (approximativement) $0.018 \leq S \leq 403.4$. Le strike valant 10, c'est assez exagéré mais on peut dans la suite se limiter à S appartenant à $[0,30]$. Pour chaque programme on trace la simulation sur l'intervalle $[0,403]$ puis sur $[0,30]$

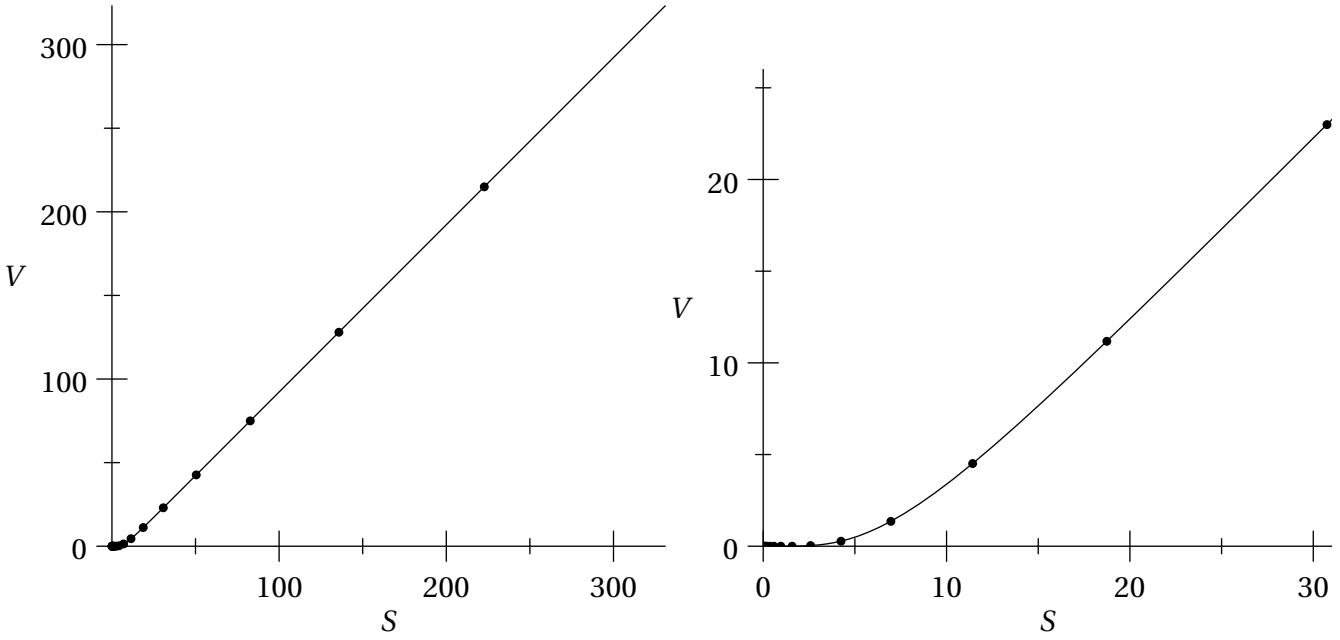


FIGURE 5. Programme 2, fenêtre $[-4,6]$

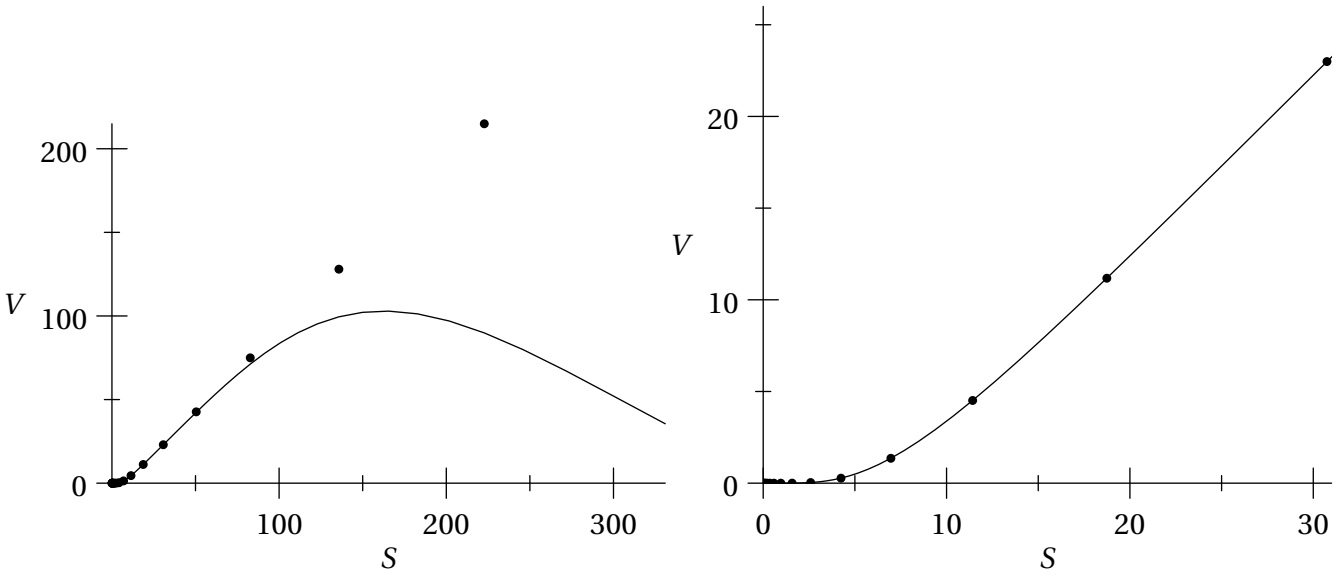


FIGURE 6. Programme 3, fenêtre $[-4,6]$

et toujours nos valeurs exactes, voir les figures 4, 5 et 6. Les programmes 1 et 2 se comportent bien sur tout l'intervalle (la différence commise entre les deux données sur le bord est négligeable devant $\exp(b)$, mais il faudrait regarder plus précisément ce qui se passe pour les grandes valeurs de S) tandis que le programme 3 donne un résultat non satisfaisant pour $S \geq 80$.

Enfin pour se faire une idée plus précise, les figures 7, 8 et 9 représentent le tracé de l'erreur sur un échantillon de S dans $[0^+,50]$. Les résultats des programmes 1 et 2 sont similaires sur $[0,50]$. Le programme 3 est équivalent sur $[0,30]$ mais dès la valeur $S = 50$ il y a un saut au niveau de l'erreur.

Conclusion. La condition aux limites a un impact important sur le domaine de validité des simulations numériques. Prendre celle qui est la plus réaliste permet de faire quelques économies en temps de calcul. Évidemment pour des taux et volatilité constantes les formules de Black et Scholes nous le permettent,

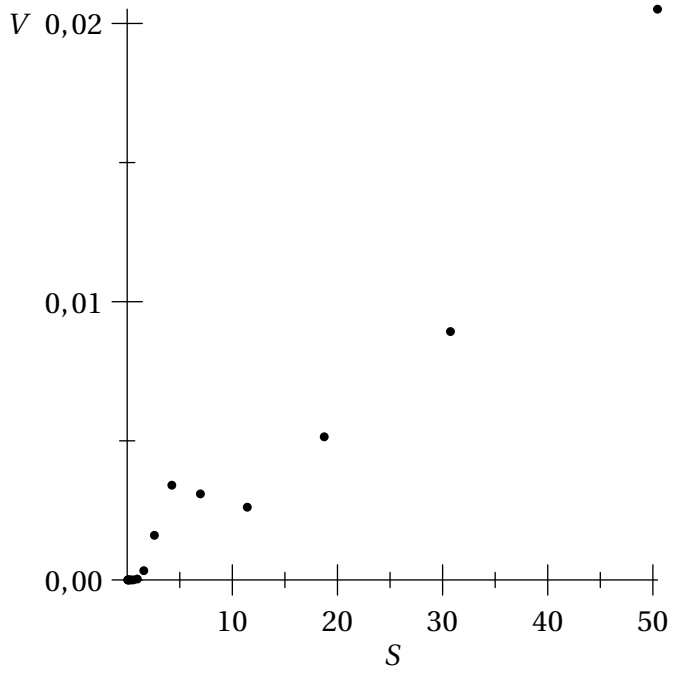


FIGURE 7. Erreur, programme 1, fenêtre [−4,6], puis zoom

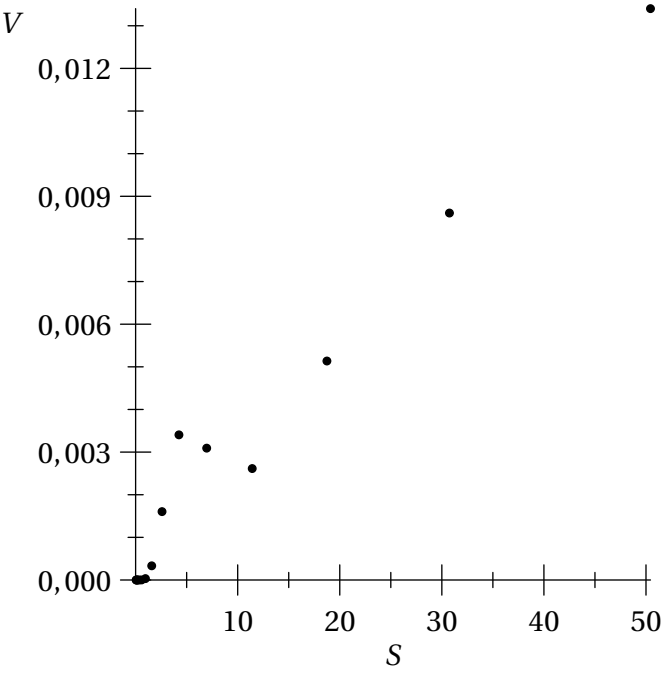
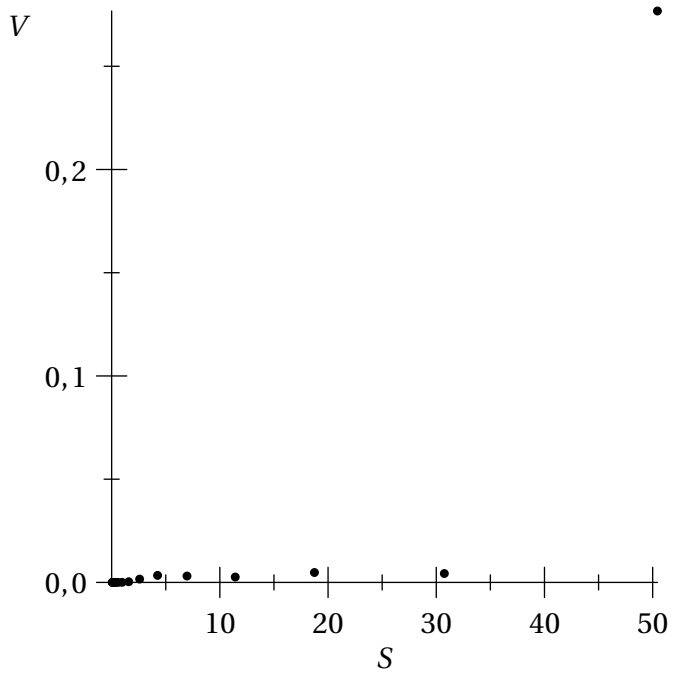


FIGURE 8. Erreur, programme 2, fenêtre [−4,6] puis zoom



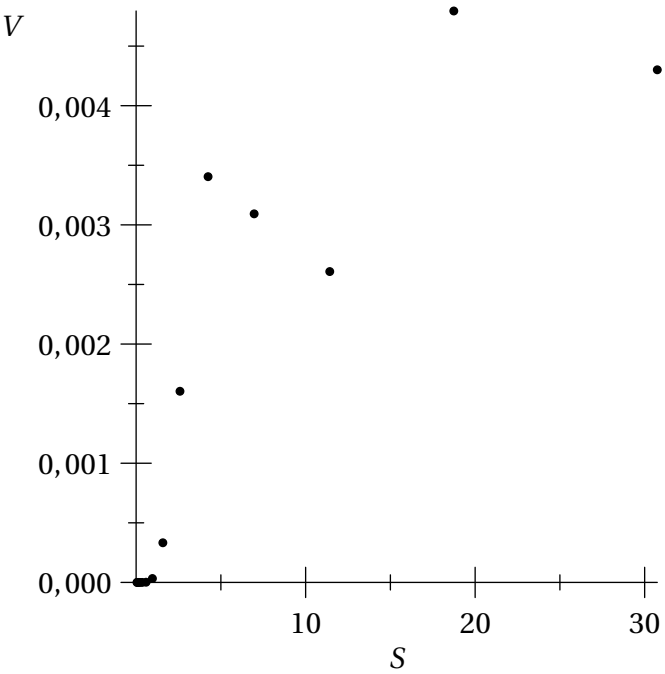


FIGURE 9. Erreur, programme 3, fenêtre [−4,6], deux zooms

mais pour des problèmes plus complexes ce n'est pas toujours possible. Il faut donc toujours garder un esprit critique vis à vis des simulations numériques.