```matlab
% EuCallExpl1.m
function price = EuCallExpl1(S0,X,r,T,sigma,Smax,dS,dt)
% set up grid and adjust increments if necessary
M = round(Smax/dS);
dS = Smax/M;
N = round(T/dt);
dt = T/N;
matval = zeros(M+1,N+1);
vetS = linspace(0,Smax,M+1)';
vetj = 0:N;
veti = 0:M;
% set up boundary conditions
matval(:,N+1) = max(vetS-X,0);
matval(1,:) = X*exp(-r*dt*(N-vetj));
matval(M+1,:) = 0;
% set up coefficients
a = 0.5*dt*(sigma^2*veti - r).*veti;
b = 1- dt*(sigma^2*veti.^2 + r);
c = 0.5*dt*(sigma^2*veti + r).*veti;
% solve backward in time
for j=N:-1:1
   for i=2:M
      matval(i,j) = a(i)*matval(i-1,j+1) +
b(i)*matval(i,j+1)+ c(i)*matval(i+1,j+1);
   end
end
% find closest point to S0 on the grid and return price
% possibly with a linear interpolation
idown = floor(S0/dS);
iup = ceil(S0/dS);
if idown == iup
   price = matval(idown+1,1);
else
   price = matval(idown+1,1) + (S0 -
(idown+1)*dS)*(matval(iup+1,1) - matval(iup,1))/dS;
end
```

```
>> CompBlsExple

p04 =

    4.0760


Exps04 =

   1.1516e+12


p03 =

    2.8446


Exps03dS2 =

    4.8453


ExpdS15 =

  -8.3267e+09


ExpdS1 =

  -2.7711e+46
```

```matlab
% EuPutImpl.m
function price = EuPutImpl(S0,X,r,T,sigma,Smax,dS,dt)
% set up grid and adjust increments if necessary
M = round(Smax/dS);
dS = Smax/M;
N = round(T/dt);
dt = T/N;
matval = zeros(M+1,N+1);
vetS = linspace(0,Smax,M+1)';
vetj = 0:N;
veti = 0:M;

% set up boundary conditions
matval(:,N+1) = max(vetS-X,0);
matval(1,:) = X*exp(-r*dt*(N-vetj));
matval(M+1,:) = 0;

% set up the tridiagonal coefficients matrix
a = 0.5*(r*dt*veti-sigma^2*dt*(veti.^2));
b = 1+sigma^2*dt*(veti.^2)+r*dt;
c = -0.5*(r*dt*veti+sigma^2*dt*(veti.^2));
coeff = diag(a(3:M),-1) + diag(b(2:M)) + diag(c(2:M-
1),1);
[L,U] = lu(coeff);

% solve the sequence of linear systems
aux = zeros(M-1,1);
for j=N:-1:1
   aux(1) = - a(2) * matval(1,j);
   %matval(2:M,j) = U \ (L \ (matval(2:M,j+1) + aux));
   matval(2:M,j) = coeff \ (matval(2:M,j+1) + aux);
end
% find closest point to S0 on the grid and return price
% possibly with a linear interpolation
idown = floor(S0/dS);
iup = ceil(S0/dS);
if idown == iup
   price = matval(idown+1,1);
else
   price = matval(idown+1,1) + (S0 -
idown*dS)*(matval(idown+2,1) - matval(idown+1,1))/dS;
end
```

```
>> CompBlsExpl

c =

    6.1165


Impl =

    5.7185
```

```matlab
% UOCallCK.m
function price = UOCallCK(S0,X,r,T,sigma,Sb,Smax,dS,dt)
% set up grid and adjust increments if necessary
M = round((Smax-Sb)/dS);
dS = (Smax-Sb)/M;
N = round(T/dt);
dt = T/N;
matval = zeros(M+1,N+1);
vetS = linspace(Sb,Smax,M+1)';
vetj = 0:N;
veti = vetS / dS;
% set up boundary conditions
matval(:,N+1) = max(vetS-X,0);
matval(1,:) = 0;
matval(M+1,:) = 0;

% set up the coefficients matrix
alpha = 0.25*dt*( sigma^2*(veti.^2) - r*veti );
beta = -dt*0.5*( sigma^2*(veti.^2) + r );
gamma = 0.25*dt*( sigma^2*(veti.^2) + r*veti );
M1 = -diag(alpha(3:M),-1) + diag(1-beta(2:M)) -
diag(gamma(2:M-1),1);
[L,U] = lu(M1);
M2 = diag(alpha(3:M),-1) + diag(1+beta(2:M)) +
diag(gamma(2:M-1),1);

% solve the sequence of linear systems
for j=N:-1:1
   matval(2:M,j) = U \ (L \ (M2*matval(2:M,j+1)));
end

% find closest point to S0 on the grid and return price
% possibly with a linear interpolation
idown = floor((S0-Sb)/dS);
iup = ceil((S0-Sb)/dS);
if idown == iup
   price = matval(iup+1,1);
else
   price = matval(iup+1,1) + (S0-Sb-
iup*dS)*(matval(iup+2,1) - matval(iup+1,1))/dS;
end
```

```
>> CompUOCallCK

UpOutCallCK =

    5.4916
```

```matlab
% AmPutExpl1.m
function price = AmPutExpl1(S0,X,r,T,sigma,Smax,dS,dt)
% set up grid and adjust increments if necessary
M = round(Smax/dS);
dS = Smax/M;
N = round(T/dt);
dt = T/N;
matval = zeros(M+1,N+1);
vetS = linspace(0,Smax,M+1)';
veti = 0:N;
vetj = 0:M;
% set up boundary conditions
matval(:,N+1) = max(X-vetS,0);
matval(1,:) = 0; %Am
matval(M+1,:) = X-Smax; %Put
% set up coefficients
a = 0.5*dt*(sigma^2*vetj - r).*vetj;
b = 1- dt*(sigma^2*vetj.^2 + r);
c = 0.5*dt*(sigma^2*vetj + r).*vetj;
% solve backward in time
for i=N:-1:1
   for j=2:M
      matval(j,i) = max(X-vetS(j),a(j)*matval(j-1,i+1)+
b(j)*matval(j,i+1)+ c(j)*matval(j+1,i+1));
   end
end
% find closest point to S0 on the grid and return price
% possibly with a linear interpolation
jdown = floor(S0/dS);
jup = ceil(S0/dS);
if jdown == jup
   price = matval(jdown+1,1);
else
   price = matval(jdown+1,1) + (S0 -
jdown*dS)*(matval(jdown+2,1) - matval(jdown+1,1))/dS;
end
```

```matlab
% AmPutImpl.m
function price = AmPutImpl(S0,X,r,T,sigma,Smax,dS,dt)
% set up grid and adjust increments if necessary
M = round(Smax/dS);
dS = Smax/M;
N = round(T/dt);
dt = T/N;
matval = zeros(M+1,N+1);
vetS = linspace(0,Smax,M+1)';
veti = 0:N;
vetj = 0:M;

% set up boundary conditions
matval(:,N+1) = max(X-vetS,0);
matval(1,:) = 0;
matval(M+1,:) = X-Smax;

% set up the tridiagonal coefficients matrix
a = 0.5*(r*dt*vetj-sigma^2*dt*(vetj.^2));
b = 1+sigma^2*dt*(vetj.^2)+r*dt;
c = -0.5*(r*dt*vetj+sigma^2*dt*(vetj.^2));
coeff = diag(a(3:M),-1) + diag(b(2:M)) + diag(c(2:M-
1),1);
[L,U] = lu(coeff);

% solve the sequence of linear systems
aux = zeros(M-1,1);
for j=N:-1:1
   aux(1) = - a(2) * matval(1,j);
   %matval(2:M,j) = U \ (L \ (matval(2:M,j+1) + aux));
   matval(2:M,j) = max(coeff \ (matval(2:M,j+1) +
aux),X*ones(M-1,1)-vetS(2:M));
end
% find closest point to S0 on the grid and return price
% possibly with a linear interpolation
jdown = floor(S0/dS);
jup = ceil(S0/dS);
if jdown == jup
   price = matval(jdown+1,1);
else
   price = matval(jdown+1,1) + (S0 -
jdown*dS)*(matval(jdown+2,1) - matval(jdown+1,1))/dS;
end
```

```
%AmPutCK.m
function price = AmCallCK(S0,X,r,T,sigma,Smax,dS,dt)

% set up grid and adjust increments if necessary
M = round(Smax/dS);
dS = Smax/M;
N = round(T/dt);
dt = T/N;
matval = zeros(M+1,N+1);
vetS = linspace(0,Smax,M+1)';
veti = 0:N;
vetj = 0:M;

% set up boundary conditions
matval(:,N+1) = max(X-vetS,0);
matval(1,:) = 0;
matval(M+1,:) = X-Smax;

alpha = 0.25*dt*( sigma^2*(vetj.^2) - r*vetj );
beta = -dt*0.5*( sigma^2*(vetj.^2) + r );
gamma = 0.25*dt*( sigma^2*(vetj.^2) + r*vetj );
M1 = -diag(alpha(3:M),-1) + diag(1-beta(2:M)) -
diag(gamma(2:M-1),1);
[L,U] = lu(M1);
M2 = diag(alpha(3:M),-1) + diag(1+beta(2:M)) +
diag(gamma(2:M-1),1);

for i=N:-1:1
  matval(2:M,i) = max(U \ (L \
(M2*matval(2:M,i+1))),X*ones(M-1,1)-vetS(2:M));
end

jdown = floor(S0/dS);
jup = ceil(S0/dS);
if jdown == jup
  price = matval(jdown+1,1);
else
  price = matval(jdown+1,1) + (S0-
jdown*dS)*(matval(jdown+2,1) - matval(jdown+1,1))/dS;
end
```

```
%CompExpImplCK_AmPut.m
S0=50;
X=50;
r=0.1;
T=5/12;
sigma=0.3;
Sb = 40;
Smax=100;
dS=2;
dT=5/1200;

Explp = AmPutExpl1(S0,X,r,T,sigma,Smax,dS,dT)
Implp = AmPutImpl(S0,X,r,T,sigma,Smax,dS,dT)
CKp = AmPutCK(S0,X,r,T,sigma,Smax,dS,dT)
```

```
>> CompExpImplCK

Explp =

    3.0616


Implp =

    3.0207


CKp =

    3.0289
```