Numerical Methods

PDE Matlab code

==========

```
function [ u, x, t ] = he_feuler(u0, g0, g1, T, N, J)
%HE_FEULER - Forward Euler scheme for the heat equation
%...

h  = 1./J;
dt = T/N;
nu = dt/h^2

t = [0:dt:T];
x = [0:h:1.0];
u = zeros(J+1,N+1);

u(:,1) = u0(x);

for k=1:N
    u(2:J,k+1) = u(2:J,k) + nu*(u(3:J+1,k)-2*u(2:J,k)+u(1:J-1,k));
    u(  1,k+1) = g0(t(k+1));
    u(J+1,k+1) = g1(t(k+1));
end

end
```

==========

```
function [ u, x, t ] = he_beuler(u0, g0, g1, T, N, J)
%HE_FEULER - Backward Euler scheme for the heat equation
% u0    - function describing the initial condition
% g0,g1 - functions describing the boundary conditions
% T     - Max time
% N, J  - Number of grid points in time / space directions

h  = 1./J;
dt = T/N;
nu = dt/h^2;

t = [0:dt:T];
x = [0:h:1.0];
u = zeros(J+1,N+1);

u(:,1) = u0(x);
```

```
A0 = [1;ones(J-1,1)*(1+2*nu);1];
Am = [-ones(J-1,1)*nu; 0];
Ap = [0; -ones(J-1,1)*nu];

for k=1:N
    b = u(:,k);
    b(  1) = g0(t(k+1));
    b(J+1) = g1(t(k+1));
    u(:,k+1) = tridiag(Am, A0, Ap, b);
end

end
```

==========

```
function [ u, x, t ] = he_cn(u0, gmin, gmax, xmin, xmax, T, N, J)
%HE_CN - Cranck-Nicolson scheme for the heat equation
% u0    - function describing the initial condition
% g0,g1 - functions describing the boundary conditions
% xmin, xmax - "x" range
% T     - Max time
% N, J  - Number of grid points in time / space directions

h  = (xmax - xmin)/J;
dt = T/N;
nu = dt/h^2;

t = [0:dt:T];
x = [xmin:h:xmax]';
u = zeros(J+1,N+1);
b = zeros(J+1,1);

u(:,1) = u0(x);

A0 = [1;ones(J-1,1)*(1+nu);1];
Am = [-0.5*ones(J-1,1)*nu; 0];
Ap = [0; -0.5*ones(J-1,1)*nu];

for k=1:N
    b(2:J) = (1.-nu) * u(2:J,k) + 0.5*nu*(u(3:J+1,k)+u(1:J-1,k));
```

```
    b(  1) = gmin(t(k+1));
    b(J+1) = gmax(t(k+1));
    u(:,k+1) = tridiag(Am, A0, Ap, b);
end

end
```

==========

```
function [V, S] = euro_call(Smin, Smax, E, T, r, sigma, N, J)
% EURO_CALL - Pricing of European call option
% Smin Smax - The underlying price range
% E         - Strike price
% T         - Time to expiration
% r         - The interest rate
% sigma     - The volatility
% J,N       - Number of grid points in price, time directions
% S         - The underlying price at t=0
% V         - The corresponding option price

k = 2*r/sigma^2;
alfa = -0.5*(k-1.);
beta = -0.25 * (k+1.)^2;

xmin = log(Smin/E);
xmax = log(Smax/E);
taumax = T * sigma^2 * 0.5;

w0 = @(x) max(exp(x)-1.,0.)./exp(alfa*x);
bmin = @(x) 0;
bmax = @(tau) (exp(xmax) - exp(-tau*k))./exp(alfa*xmax+beta*tau);


[w x tau] = he_cn(w0, bmin, bmax, xmin, xmax, taumax, N, J);

S = E * exp(x);
V = w(:,end) .* exp(alfa*x+beta*taumax)*E;
end
```

```
==================================================

function [V, S] = american_put(Smin, Smax, E, T, r, sigma, N, J)
% AMERICAN_PUT - Pricing of American put option
% Smin Smax - The underlying price range
% E         - Strike price
% T         - Time to expiration
% r         - The interest rate
% sigma     - The volatility
% J,N       - Number of grid points in price, time directions
% S         - The underlying price at t=0
% V         - The corresponding option price

omega = 1.6;

k = 2*r/sigma^2;
alfa = -0.5*(k-1.);
beta = -0.25 * (k+1.)^2;

g = @(x,t) exp(-beta*t).*max(exp(-alfa*x)-exp(0.5*(k+1.).*x),0);

xmin = log(Smin/E);
xmax = log(Smax/E);
taumax = T * sigma^2 * 0.5;

dt = taumax/N;
h  = (xmax-xmin)/J;
nu = dt/h^2;
t = [0:dt:T];
x = [xmin:h:xmax]';

w = g(x,0);
b = zeros(J+1,1);

counter = 0;
for n=1:N
   gn = g(x,t(n));
   b(1) = gn(1);
   b(2:J) = (1-nu)*w(2:J) + 0.5*nu*(w(3:J+1)+w(1:J-1));
   b(J+1) = gn(J+1);
   u = max(w,gn);
   while 1
       counter = counter+1;
      u(1) = gn(1);
      er = 0.;
      for j=2:J-1
```

```
            du = 1./(1.+nu)*(b(j)+0.5*nu*(u(j-1)+u(j+1))) - u(j);
            u1 = u(j) + omega * du;
            if u1>gn(j)
                u(j) = u1;
                er = max(er, abs(du));
            else
                u(j) = gn(j);
            end
        end
        u(J+1) = gn(J+1);
        if er < 0.0000001
            break
        end
    end
    w = u;
end

counter/N

S = E * exp(x);
V = w .* exp(alfa*x+beta*taumax)*E;
end
```

==========