Εργαστήριο Υπολογιστικής Επιστήμης ΙΙ
Πέμπτη 5 Φεβρουαρίου 2008

Άσκηση 1η

```
disp('   ')
k = input( 'enter the value of the diffusion coefficient k   ')
disp(' Enter the number of timesteps between snapshots ')
n = input (' In the form [n1, n2, n3, n4 ]            ')
delt = .05

n1 = n(1); n2= n(2); n3 = n(3); n4 = n(4);
t1 = n1*delt
t2 = t1 +n2*delt
t3 = t2 +n3*delt
t4 = t3 +n4*delt

delx = .05
r = .5*k*delt/(delx^2);

J= 10/delx;

x = 0:delx:10;

   snap0 = heatf(x);
   %snap0 =  -sin(((2*%pi)*mtlb_double(x))/10) .*exp(0.5*mtlb_double(x));

v = snap0(2:1:J);

D = sparse(1:J-1, 1:J-1,(1+2*r), J-1, J-1, J-1);
E = sparse(2:J-1, 1:J-2, -r, J-1, J-1, J-1);
A = D + E +E';

[L,U] = lu(A);


%% Note: the original index j runs from j = 1 ( x = 0) to j = J ( x = 10 -delx).
%% The index in v (and hence in b and z) runs from j = 1 (x= delx) to
%% j = J-1 (x = 10-delx).
for n = 1:n1
```

```
    b = 2*v ;
    b(1) = b(1) + r*( left(n*delt) +left((n-1)*delt) );
    b(J-1) = b(J-1) + r*( right(n*delt) +right((n-1)*delt) ) ;
    y = L\b';
    z = (U\y)';
    v = z- v;
end

snap1 = [left(n*delt),v,right(n*delt)];
disp('Computed up to time t1')

for n = n1+1: n1+n2
    b = 2*v;
    b(1) = b(1) + r*( left(n*delt) + left((n+1)*delt) );
    b(J-1) = b(J-1) + r*( right(n*delt) + right((n+1)*delt) );
    y = L\b';
    z = (U\y)';
    v = z-v;
end

disp('Computed up to time t2 ')
snap2 = [left(n*delt),v,right(n*delt)];

for n = n1+n2+1: n1 + n2 +n3
    b = 2*v;
    b(1) = b(1) +r*( left(n*delt) + left((n+1)*delt) );
    b(J-1) = b(J-1) + r*( right(n*delt) + right((n+1)*delt) );
    y = L\b';
    z = (U\y)';
    v = z-v;
end

disp('Computed up to time t3 ')
snap3 = [left(n*delt),v,right(n*delt)];

for n = n1 + n2 +n3+1: n1+n2+n3+n4
    b = 2*v;
    b(1) = b(1) +r*( left(n*delt) + left((n+1)*delt) );
    b(J-1) = b(J-1) + r*( right(n*delt) + right((n+1)*delt) );
    y = L\b';
    z = (U\y)';
```

```
    v = z-v;
end

disp('Computed up to time t4 ')
snap4 = [left(n*delt),v,right(n*delt)];

plot(x,snap0,x,snap1,x,snap2,x,snap3,x,snap4)
```

```
%                          Matlab program heat4
%
%      This program integrates the heat equation u_t - ku_xx = q(x) on
%   the interval [0, 10] with Neumann boundary conditions u_x(0,t) =
%   u_x(10,t) = 0.  Crank-Nicholson method is used with delx = .05
%   and delt = .05.
%      At run time user must enter the value of the diffusion
%   coefficient k. User also must enter the number of time steps
%   n1, n2, n3, and n4 between snapshots as in program heat3.
%   Output is written into the vectors 'snap0',snap1', 'snap2','snap3'
%   and 'snap4'.  The vectors may be plotted alone or together.
%      The initial data comes from the file heatf.m.  User must
%   provide a function mfile q.m for the source.



disp('  ')
k = input('enter the value of the diffusion coefficient k  ')
disp(' Enter the number of time steps between snapshots, n1,n2, n3, n4 ')
n = input(' in the form [n1 n2 n3 n4]    ')
n1 = n(1);  n2 = n(2); n3 = n(3);  n4 = n(4);
delt = .05
t1 = n1*delt
t2 = (n1 +n2)*delt
t3 = (n1 + n2 +n3)*delt
t4 = (n1 + n2 + n3 +n4)*delt

delx = .05
r = .5*k*delt/(delx^2);

J = 200

x = 0:delx:10;

   snap0= heatf(x);
   qq = q(x);
```

```
v= snap0;

D= sparse(1:J+1, 1:J+1, (1+2*r), J+1,J+1, J+1);

D(1,1) = D(1,1) - r;

D(J+1, J+1) = D(J+1, J+1) - r;

E = sparse(2:J+1, 1:J, -r, J+1,J+1,J+1);

A = D + E+ E';

[L,U] = lu(A);


for n = 1:n1
    b = 2*v + delt*qq;
    y = L\b';
    z = (U\y)';
    v = z - v;
end
    snap1 = v;
disp('Computed up to time t1 ')

for n = n1+1: n1+n2

    b = 2*v +delt*qq;
    y = L\b';
    z = (U\y)';
    v = z-v;
end
    snap2 = v;
disp('Computed up to time t2')

for n = n1 + n2+1:n1 + n2 +n3

    b = 2*v +delt*qq;
    y = L\b';
    z = (U\y)';
    v = z-v;
end
```

```
    snap3 = v;
disp('Computed up to time t3')

for n = n1 + n2 + n3+1:n1 + n2 + n3 +n4

     b = 2*v +delt*qq;
     y = L\b';
     z = (U\y)';
     v = z-v;
end

    snap4 = v;
disp('Computed up to time t4')

plot(x,snap0,x,snap1,x,snap2,x,snap3,x,snap4)
```

```
%                        Matlab program heat5
%
%     This program integrates the heat equation u_t -ku_xx = 0 on
%   the interval 0 < x < 10, with prescribed boundary values
%   u(0,t) = left(t), and u(10,t) = right(t). The diffusion
%   is piecewise constant: for x > 0, k = 1, while for x < 0
%   k = kleft which must be entered at run time. The Crank-Nicholson
%   is used with delx = .05 but now delt = .025.
%     As in programs heat3 and heat4 the programs computes four
%   snapshots of the solution at times t1, t2,t3,and t4 in
%   addition to the initial data which is snap0.
%     At run time, user must enter the diffusion coefficient kleft,
%   and the number of time steps n1, n2, n3, and n4 between snapshots.
%   n1 is the number of time steps to snap1, n2 the number of time steps
%   from snap1 to snap2, etc. Vectors snap0, .. .. snap4 may be plotted
%   separately or together.
%      Also required: a function file heatf.m for the inital data
%   f(x), and function files left.m and right.m for the boundary values
%   u(0,t) = left(t), and u(10,t) = right(t).




kleft = input('enter the value of the diffusion coeff. kleft  ')
disp(' Enter the number of time steps between snapshots, n1, n2, n3, n4 ')
n = input(' in the form [n1 n2 n3 n4]        ')

n1 = n(1);  n2 = n(2);
n3= n(3);  n4 = n(4);
delt = .025
t1 = n1*delt
t2 = (n1 +n2) *delt
t3 = (n1 + n2 +n3)*delt
t4 = (n1 + n2 + n3+n4)*delt

delx = .05

sright = .5*delt/(delx^2);
```

```
sleft = .5*kleft*delt/(delx^2);

J =  10/delx;

x = 0:delx:10;

    snap0 = heatf(x);



for j = 1:J/2 -1
    diag(j) = 1+2*sleft;
    diag(j+J/2) = 1+2*sright;
end
diag(J/2) = sleft + sright;

D= sparse(1:J-1, 1:J-1, diag, J-1, J-1, J-1);

for j = 1:J/2-1
    lower(j) = -sleft;
    lower(j+J/2 -1) = -sright;
end

E = sparse(2:J-1, 1:J-2, lower,J-1, J-1,J-1);

A = D + E + E';

[L,U] = lu(A);

e= [-sleft, sleft + sright - 2, -sright ];

C = sparse( J/2, [J/2-1, J/2, J/2+1], e, J-1, J-1, J-1);


v = snap0(2:1:J);

% special setup for the first time step when the initial data and
% the boundary data are discontinuous in the corners.

    b = 2*v' + C*v';
    b(1) = b(1) + sleft*( left(delt)+ snap0(1)  );
    b(J-1) = b(J-1) +sright*( right(delt) + snap0(J+1) );
```

```
    y = L\b;
    z = (U\y)' ;
    v = z-v ;

for n = 2:n1
     b = 2*v'+C*v' ;
     b(1) = b(1) + sleft*( left(n*delt) + left((n-1)*delt) );
     b(J-1) = b(J-1) +sright*( right(n*delt) + right((n-1)*delt) );
     y = L\b;
     z = (U\y)';
     v = z - v;
end
     snap1 = [left(n*delt),v,right(n*delt)];
disp('Computed up to time t1 ')

for n = n1+1:  n1 + n2

     b = 2*v' + C*v';
     b(1) = b(1) +sleft*( left(n*delt) + left((n-1)*delt) );
     b(J-1) = b(J-1) +sright*( right(n*delt) +right((n-1)*delt) );
     y = L\b;
     z = (U\y)';
     v = z-v;
end
     snap2 = [left(n*delt),v,right(n*delt)];
disp('Computed up to time t2')

for n = n1 + n2+1: n1 + n2 +n3
     b = 2*v' +C*v';
     b(1) = b(1) + sleft*( left(n*delt) + left((n-1)*delt) );
     b(J-1) = b(J-1) +sright*( right(n*delt) + right((n-1)*delt) );
     y = L\b;
     z = (U\y)';
     v = z-v;
end
    snap3 = [left(n*delt),v,right(n*delt)];
disp('Computed up to time t3')

for n =n1 + n2 + n3+1:n1 + n2 + n3 +n4
     b = 2*v' + C*v';
     b(1) = b(1) +sleft*( left(n*delt) + left((n-1)*delt) );
```

```
    b(J-1) = b(J-1) + sright*(right(n*delt) + right((n-1)*delt) );
    y = L\b;
    z = (U\y)';
    v = z-v;
end

   snap4 = [left(n*delt),v,right(n*delt)];
disp('Computed up to time t4')

plot(x,snap0,x,snap1,x,snap2,x,snap3,x,snap4)
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%% Required files %%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function y = heatf(x)
%
% This mfile contains all the initial data choices for the three programs
% heat3, heat4, and heat5. Remove the % sign from in front of the
% formula you want to use.  Then replace it before making the
% data choice.


%            Initial Data Choices for heat3

%   1)
    y = -sin(2*pi*x/10.0).*exp(.5*x);

%   2)
%   y1 = x;  y2 = 10 -x;  y = (x < 5.001).*y1 + (x > 5).*y2 ;

%   3)
%   y = .5*x - 2 + sin(.2*pi*x);

%   4)
%   y = zeros(size(x));

%%%%%%%%%%%%%%%%%%%%%%%%


%            Initial Data Choices for heat4

%  1)
%  y = exp(-(x-5).^2) ;

%   2)
%   y = .01*(.25*x.^4 - (17*x.^3)/3 + 35*x.^2 ) - 1 ;

%  3) and 4)
%  y = zeros(size(x));
```

```
%%%%


%                 Initial Data Choices for heat5

%%   1)
%   y = sin(pi*x/10);

%   2)
%   y = exp(-2*(x-7.5).^2) ;

%   3)
   y = .5*x - 2 + 3*exp(-.5*(x-5).^2);

%   4)
%   y = zeros(size(x));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



function y = left(t)
%
%                These are the left boundary data choices

%      y = t/(1+t) + t.*exp(-.2*t);
      y = -2;
%      y = 100.0;
%      y = 0;


function y = q(x)

%     y = exp(-(x-5).^2);

%      y = 0;

      y = (x < 5).*exp(-2*(x-2).^2) - 2*(x >=5).*exp(-8*(x-7).^2);
```

```
function y = right(t)
%
%                 These are the right boundary data choices
%
%    y = t/(1+t) + t.*exp(-.2*t)

   y = 3;
%     y = 0;
```

```
function w = u1(x,t)

    w = 40/(pi.^2)*sin(pi*x/10)*exp(-(pi/10)^2*t);
```