```
%% Lab#3 For u_t = mu u_xx with u(0) = 0, u(pi) = 0
%%
%%                _                                _       _
%%  Solved using u_t = mu (1/dx^2) B(1,-2,1) u = [A]u
%%
%% The solution is initialized using the eigensystem of [A]
%%                -1                    _0                  T
%%      [L] = [X]  [A] [X]  , setting C    = [1,1,1,...,1]
%%
%%       _           _0
%%       u(0) = [X] C   which excites all the eigenmodes.
%%
%%    A standard ODE solver is use to integrate the equations in time.
%%    At each step the solution is decomposed in the eigensystem
%%     _ n      -1 _
%%     C   = [X]   u(t)  to track the decay of each of the eigenmodes.
%%
%%   Problem Definition:
%%   The MATLAB internal routine ODE23 is used to integrate in time.
%%   After each iteration the current solution can be decomposed in
%%                                                          _n
%%   terms of the eigenvectors, giving the current state of C .
%%   This shows how the ODE solver attacks each component.
%%
%%   Try running the code for jmax = 31, exciting all the modes and
%%   total time = 100.
%%   A plot of the eigenvalues, decompostion coefficients C  in
%%   bar graph and line format and the final solution is presented.
%%   Note that with all the modes excited the initial C's are all 1.
%%   Try just one mode (say 3 and 20) and watch what happens.
%%

global A;

jmax = input('Enter Jmax = ')
mu = 1.e-2;
%%mu   = input('Enter mu = ')

%  Grid spacing
%dx = 1./(jmax-1)*pi;
dx = 1./(jmax-1)*pi;

%  Grid
x = [0:jmax-1]*dx;

%  Load Matrix and get Eigensytem
[A,X,L] = diffusion(jmax,dx,mu);

%  Sort Eigensystem smallest lambda to largest
```

```
[X,L] = sort_eig(jmax,X,L);

%  Set up initial condition
fprintf('Enter 0 to initalize a particular element of the spectrum \n');
fprintf('Enter 1 to initalize all the elements of spectrum to 1 \n');
init = input('Enter init = ');
if init == 1
  C0 = ones(jmax,1);
elseif init==0
  C0 = zeros(jmax,1);
  i_element = input('Enter index of nonzero spectrum element (<=jmax) = ');
  C0(i_element) = 1;
end

%  Initial u
u = X*C0;

t0 = 0.0;
t1 = 100.0;
t1 = input('Enter total time of integration (D = 100) = ');

%  Use ODE solver to integrate ODE between t0 and t1
%  Returns a matrix U whose rows are the solution at times T(i)
i_ode = input('Choose ode solver: 0 ode23 / 1 ode45 / 2 ode113 = ');
if i_ode == 0 [T,U] = ode23('ode_fun',[t0,t1],u); end
if i_ode == 1 [T,U] = ode45('ode_fun',[t0,t1],u); end
if i_ode == 2 [T,U] = ode113('ode_fun',[t0,t1],u); end

%  N = total number of time steps used by ode23
N = size(T,1);  fprintf('Total number of steps = %g \n',N);

np = input('Enter pause delay (0 for none) (-1 for full pause) = ')
XI = inv(X);
for n = 1:N
  u = U(n,1:jmax)';
  w = XI*u;

  subplot(2,2,1);
  plot([1:jmax], diag(L),'r+');
  xlabel('m');
  ylabel('Eigenvalues');

  subplot(2,2,2);
  bar(w);
%$$$   title(['Iteration # ',int2str(n)]);
  axis([0 jmax+1 0 1]);
  xlabel('m');
  ylabel('C_m');
  text(-10-jmax/2,1.1,['ODE for u_t = mu u_xx using B(1,-2,1)', ...
    '  Iteration # ',int2str(n)]);
```

```
  subplot(2,2,3);
  semilogy([1:jmax],abs(w),'r');
  xlabel('m');
  ylabel('C_m');

  subplot(2,2,4);
  plot(x,u,'r');
  xlabel('x');
  ylabel('u');

  if np >= 0 pause(np); end
  if np < 0 pause; end

end


%%diffusion.m
function [A,X,L] = diffusion(jmax,dx,mu)
a = mu*ones(jmax,1)/dx^2;
b = -2.*a;
c = a;
A = load_tri(jmax,a,b,c);
[X,L] = eig(A);


%%load_tri.m
function [B] = load_tri(jmax,a,b,c);
B = diag(a(2:jmax),-1) + diag(b) + diag(c(1:jmax-1),1);

%%ode_fun.m
function yprime = ode_fun(t,y)
global A;
yprime = A*y;

%%sort_eig.m
function [X,L] = sort_eig(m,X,L)
Lam = abs(diag(L));
[LL,I] = sort(Lam);
P = zeros(m,m);
for j = 1:m
  P(j,I(j)) = 1;
end
Y = X*P';
X = Y;
Z = P*diag(L);
L = diag(Z);
```