Nima Fariborzi MAE 290b Final Project

Table of Contents

Problem 1 1
Problem 1 part C 3
Functions 3

Problem 1

```
clear all;
close all;
clc;
%format long;

alpha=.1;
a=2;
omega=50;
dt=.002;
h=0.0125;
kappa=alpha.*dt./(2.*h.^2);
tol = 1e-5;

x=0:h:1;
y=0:h:1;
[X,Y]=meshgrid(x,y);
N=length(X)-2;

%Location where T(x=.55,y=.45)
pLoc=[find(x==0.55),find(y==0.45)];

%Q is source term without f yet
Q=2.5.*sin(4.*pi.*X).*sin(8.*pi.*Y);
Q=Q(2:end-1,2:end-1);
Qvec=reshape(Q,N*N,1); %Making it into a vector

%Applying f(t) to Q
f=@(t) (1-exp(-a.*t)*sin(omega.*t).*cos(2.*omega.*t))*Qvec;

%Diag matrices used in ADI function
posxx=speye(N*N)+kappa.*fdaX(N,N);
negxx=speye(N*N)-kappa.*fdaX(N,N);
posyy=speye(N*N)+kappa.*fdaY(N,N);
negyy=speye(N*N)-kappa.*fdaY(N,N);

%itr time to reach steady state counter
ntime=0;
```

```
%IC of T
ic=0.01.*sin(pi.*X).*sin(pi.*Y);
icB=ic(2:end-1,2:end-1);
icvec=reshape(icB,N*N,1); %Making it into a vector

%IC of time and T applied
tvec=0;
sol=ic;
solloc=sol(pLoc(1),pLoc(2));

%Variable for convergence
conv=norm(sol(:,:,1),'fro');

%Variable for previous step to update
solold=icvec;

tic;
while conv>=tol
    %Apply ADI for next solution
    unew=ADI(solold,ntime,tvec(end),dt,f,posxx,negxx,posyy,negyy);

    %Append solution output and time vector
    tvec=[tvec,tvec(ntime+1)+dt];
    sol(:,:,ntime+2)=applyBC(reshape(unew,N,N));
    solloc(ntime+2)=sol(pLoc(1),pLoc(2),ntime+2);

    %Update variables
    conv=norm(sol(:,:,ntime+2)-sol(:,:,ntime+1),'fro');
    solold=unew;
    ntime=ntime + 1;
end
fprintf('itr n=%d done, conv = %1.4e. \n',ntime-1,conv);
fprintf('Realtime to reach steadystate %1.2f, \n',toc);

figure(1)
plot(tvec,solloc)
xlabel('Time')
ylabel('Temperature')
title('Time evolution of T(@x=.55,@y=.45,t)')

figure(2)
solend=sol(:,:,end);
contourf(X,Y,solend,'ShowText','on');
xlabel('x')
ylabel('y')
title('Contour plot at T(end)')

%part b solving K and showing error anyalsis
K=1/(3.2*pi*pi);

Tepoint=K*sin(4*pi*.55)*sin(8*pi*.45);
Teerror=(solloc(end)-Tepoint)/Tepoint;
fprintf('Steady state error on the point is %1.4e \n',Teerror);
```

```
Te=K*sin(4*pi*X).*sin(8*pi*Y);
figure(3)
contourf(X,Y,Te,'ShowText','on');
xlabel('x')
ylabel('y')
title('Expected steady state temperature distribution')

%Error on each point at the end
Terror=solend-Te;

%Creating a heatmap to show exactly where the error is

figure(4)
mapscheme=[linspace(1,1)',linspace(0,1)',linspace(0,1)';linspace(1,0)',linspace(1,0)',lins

hoterror=heatmap(flip(Terror,2),'Colormap',mapscheme, 'ColorLimits',
[-3E-4,3E-4]);
hmLabel = string(0:h:1);
hmLabel(mod(0:length(hmLabel)-1,8)~=0) = " ";
hoterror.XDisplayLabels = hmLabel;
hoterror.YDisplayLabels = flip(hmLabel);
hoterror.NodeChildren(3).XAxis.Label.Interpreter = 'latex';
hoterror.NodeChildren(3).YAxis.Label.Interpreter = 'latex';
hoterror.NodeChildren(3).Title.Interpreter = 'latex';
hoterror.XLabel = '$x$';
hoterror.YLabel = '$y$';
hoterror.Title = {'Error in Computed Steady State','Temperature
 Distribution'};
```

Problem 1 part C

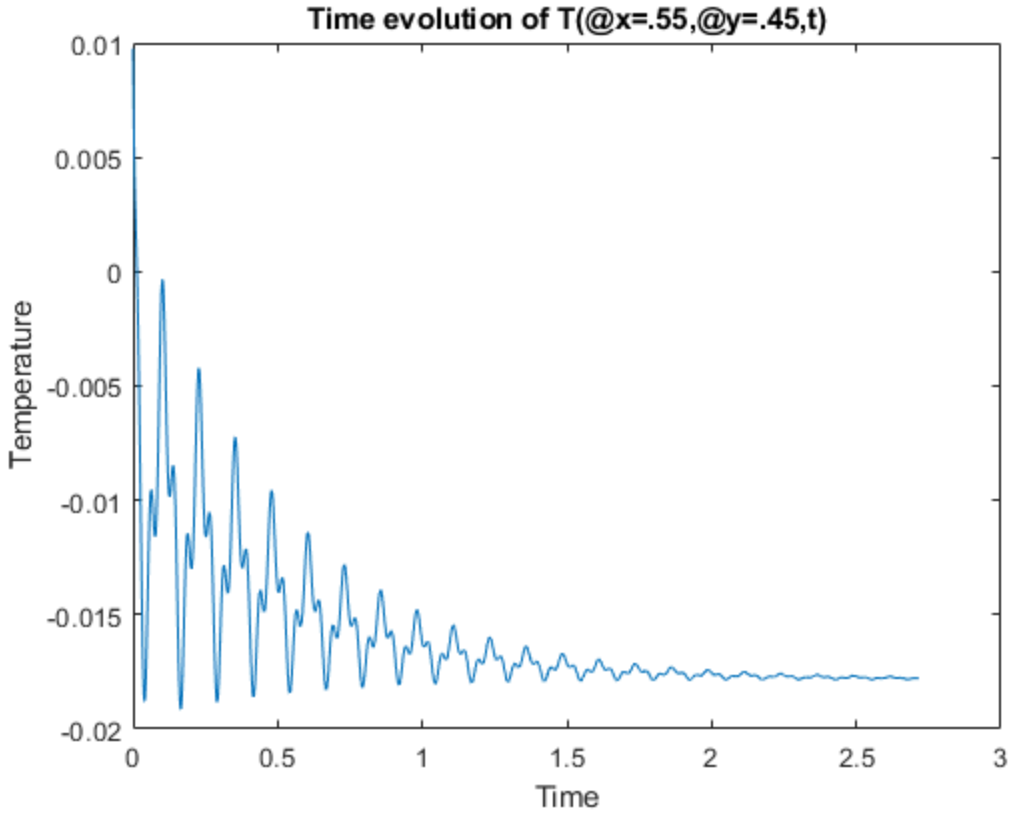
```
%Since the original equation is to be solved over a rectangluar domain with
%zero boundary conditions. Lets apply the spectral method to this problem
%so that we will achieve higher accuracy without needing finite difference
%methods. I would use a fast fouriour transform to then get a linear ODE
%for each fourir coefficient. By symmetry I can make my points only go to
%0to 39 in the spatial domain. Then the decoupled system of linear
%first-order ODE's will be converted to a single vector equation using RK4
%Also,since the system is expected to be stiff, Matlabs ode23s can be used.
%Then perform an inverse fourier transform that will result in a
%highly accurate solution in both time and space.
```

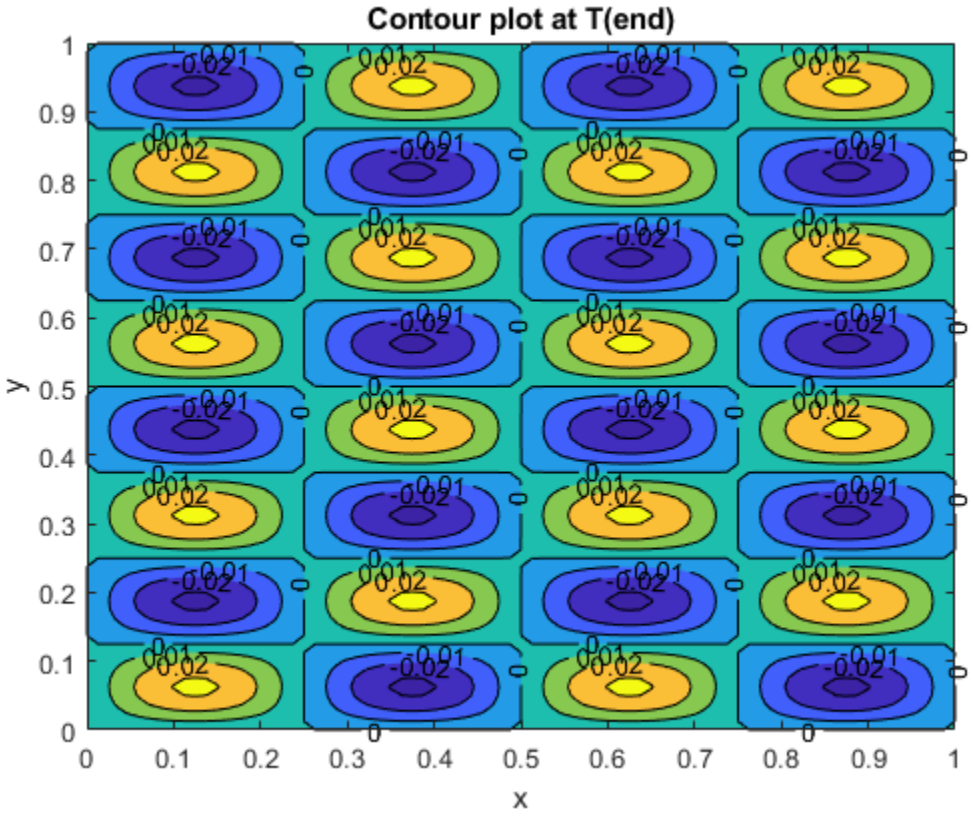
Functions

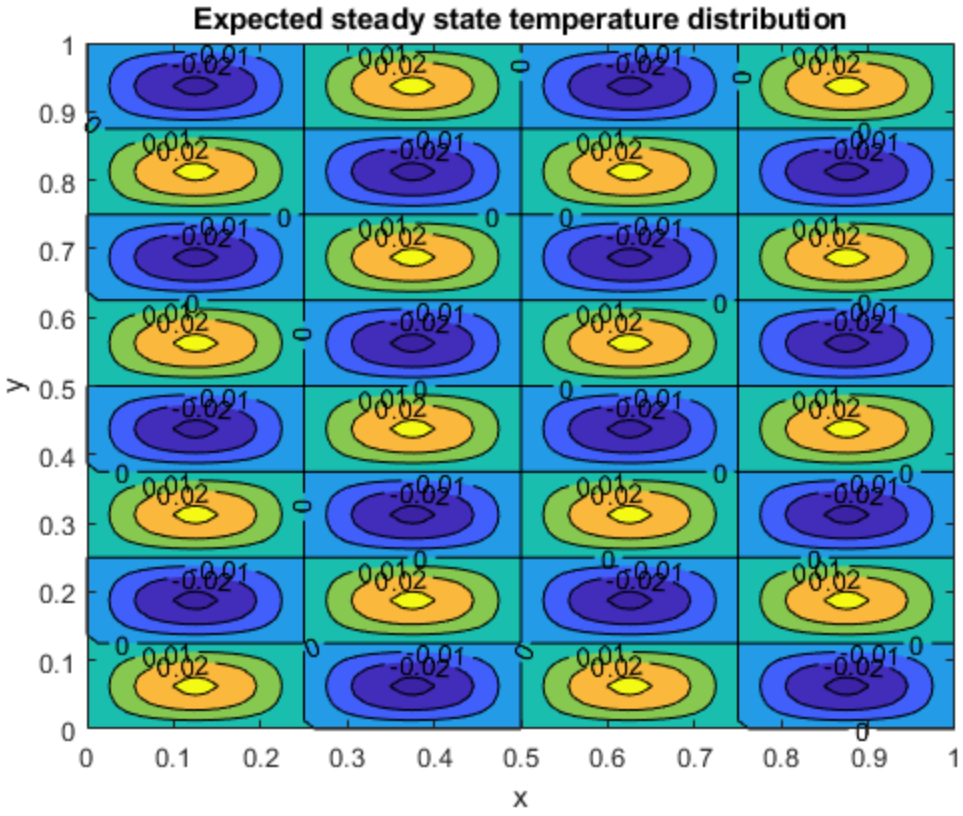
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

                              %Using diag%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

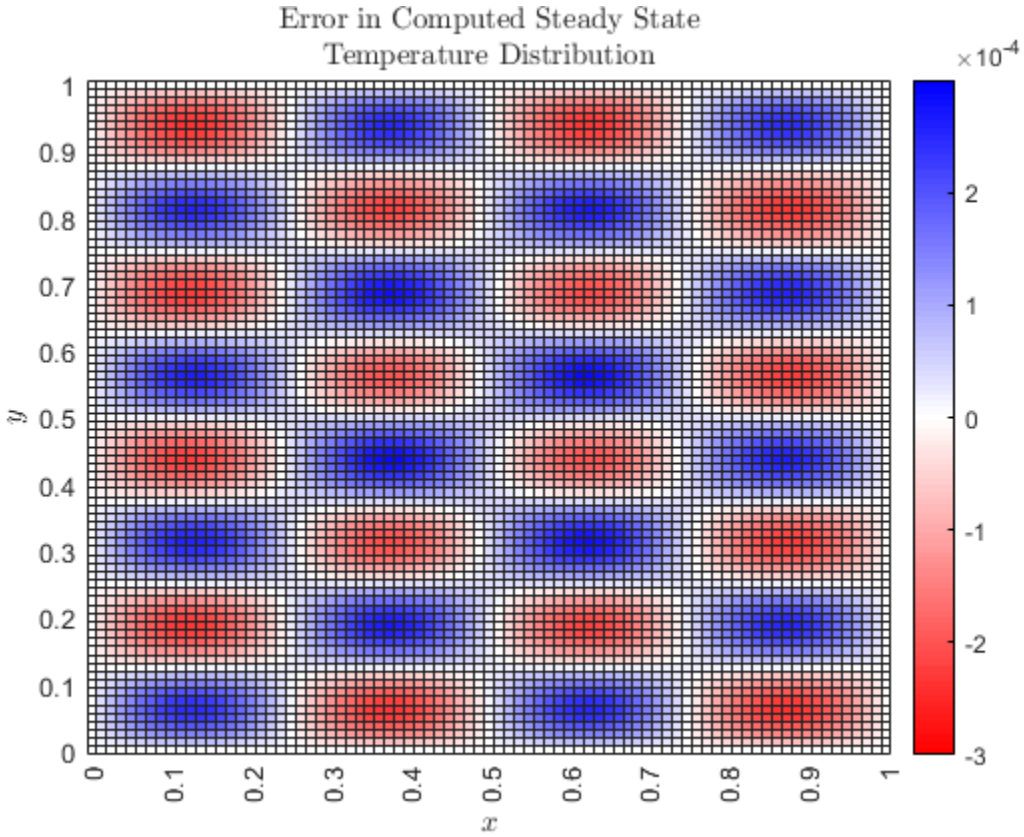
```
function Mat=fdaX(Nx,Ny)
%Making FDA matrix for x with second order central
I=speye(Ny);
P=spdiags([ones(Nx,1),-2.*ones(Nx,1),ones(Nx,1)],-1:1,Nx,Nx);

Mat=kron(P,I);
end

function Mat=fdaY(Nx,Ny)
%Making FDA matrix for y with second order central
P=speye(Nx);
A=spdiags([ones(Ny,1),-2.*ones(Ny,1),ones(Ny,1)],-1:1,Ny,Ny);

Mat=kron(P,A);
end

function BCS=applyBC(solution)
%Apply BCS after FDA has been applied
BCS=zeros(size(solution,1)+2,size(solution,2)+2);
BCS(2:end-1,2:end-1)=solution;
end


function x=tridiagSolve(A,b)
%Confirming Matrix is equal
if size(A,1)==size(A,2)
    s=length(A);
else
    disp('error in matrix');
    return;
end
x=zeros(s,1);

[B,ds]=spdiags(A);
d=ds(end);
bs=B(:,2);
cs=B(d+1:end,3);
ks=zeros(s-d,1);
ss=zeros(s,1);

ks(1:d)=(cs(1:d))./(bs(1:d));
ss(1:d)=(b(1:d))./(bs(1:d));

for ii=d+1:s-d
    ks(ii)=(A(ii,ii+d))/(A(ii,ii)-A(ii,ii-d)*ks(ii-d));
    ss(ii)=(b(ii)-A(ii,ii-d)*ss(ii-d))/(A(ii,ii)-A(ii,ii-d)*ks(ii-d));
end

for ii=s-d+1:s
    ss(ii)=(b(ii)-A(ii,ii-d)*ss(ii-d))/(A(ii,ii)-A(ii,ii-d)*ks(ii-d));
end

x(s-d+1:s)=ss(s-d+1:s);
```

```
for ii=s-d:-1:1
    x(ii)=ss(ii)-ks(ii)*x(ii+d);
end
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                                    %ADI%

function solvec=ADI(solspot,nspot,tspot,tstep,funcQ,pxx,nxx,pyy,nyy)
%Check if even or odd ADI must be computed
if mod(nspot,2)==0
    solvec=evenADI(solspot,tspot,tstep,funcQ,pxx,nxx,pyy,nyy);
else
    solvec = oddADI(solspot,tspot,tstep,funcQ,pxx,nxx,pyy,nyy);
end
end

function solvec=oddADI(solspot,tspot,tstep,funcQ,pxx,nxx,pyy,nyy)
%same thing as even except different RHS
qog=funcQ(tspot);
qhalf=funcQ(tspot+tstep/2);
q1=funcQ(tspot+tstep);

r=pxx*pyy*solspot+(tstep/4)*pyy*(qog+qhalf)+(tstep/4)*nyy*(qhalf+q1);
i=tridiagSolve(nxx,r);

solvec = tridiagSolve(nyy,i);
end

function solvec=evenADI(solspot,tspot,tstep,funcQ,pxx,nxx,pyy,nyy)
%Define Q at time locations
qog=funcQ(tspot);
qhalf=funcQ(tspot+tstep./2);
q1=funcQ(tspot+tstep);
%rhs then use Ax=b from tridiagonal matrix to solve for the solution
r=pxx*pyy*solspot+(tstep/4)*pxx*(qog+qhalf)+(tstep/4)*nxx*(qhalf+q1);
i=tridiagSolve(nxx,r);

solvec=tridiagSolve(nyy,i);
end

itr n=1357 done, conv = 8.2410e-06.
Realtime to reach steadystate 400.56,
Steady state error on the point is 5.0595e-03
```


Time evolution of T(@x=.55,@y=.45,t)
Temperature
Time
0.01
0.005
0
-0.005
-0.01
-0.015
-0.02
0
0.5
1
1.5
2
2.5
3


Contour plot at T(end)
y
x
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
-0.02
-0.01
0
0.01
0.02


Expected steady state temperature distribution
y
x


Error in Computed Steady State
Temperature Distribution
×10^-4
y
x

Published with MATLAB® R2022b