Numerical Methods PDE-BTCS Class Exercise.   April-23-2018

Last name ___________________________ First name _______________ ID ___________________________

PROBLEM: BTCS-PDE. Solve the following PDE using the BTCS method:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + b\frac{\partial u}{\partial x}$$

BCs:

$$u(0,t) = 100, \qquad u(10,t) = 50$$

IC:

$$u(x,0) = 0 \quad 0 \le x \le 10$$

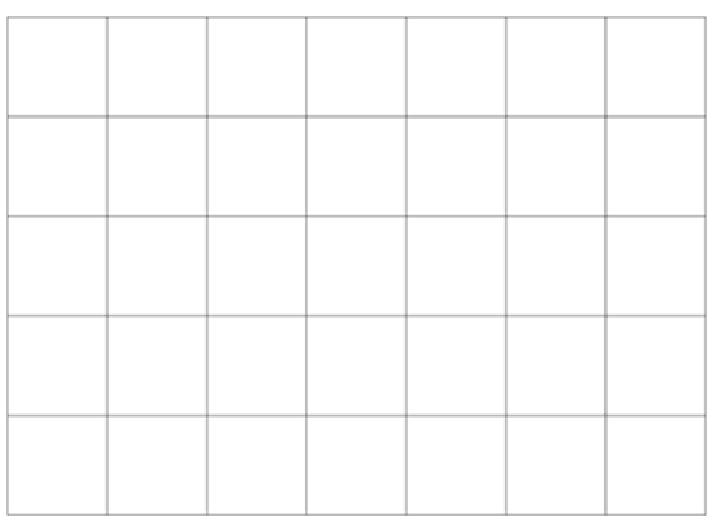
Use the BTCS finite difference method with a matlab program for the solution (e.g., modified the one available for CN). Follow two strategies

(A) Solve with a constant $\Delta t = 0.1$, Δx=2 for values of *b* = 4, 2, 0, -2, -4 and plot u for specific time points, t=[3, 6, 9, 12]

(B) Once solved with a constant $\Delta t = 0.1$ attempt a numerical experiment increasing the value of Δt by 5% for each new time step to more quickly obtain the steady-state solution. For the experiment solve it for an extended tmax. Start with Δt=0.1, Δx=2, b=0 and plot u for specific time points, t.

In APPENDIX, you have the solution to this problem with the CN method. Use it for testing your own solution with the BTCS method. Follow the steps:

(1) **Discretize the domain of the problem.** For two independent variables use a mesh grid, each axis representing one of the independent variables. Use as running indices, **i** for space and **k** for time, e.g., i=0,1,2,… ix, and k=0,1,2,…,kx. Both starting in zero. Use the grid below:



(2) Discretization of the Differential equation.

Starting with

$$\frac{\partial U}{\partial t} = \frac{\partial^2 U}{\partial x^2} + b\frac{\partial U}{\partial x} \quad [Eq\ 1]$$

To develop the BTCS scheme discretize each term of Eq 1 at the grid point (*i,k+1*):

a. Approximate the 1st order time derivative with _______ finite difference approximation of $\vartheta(\Delta t^n)$ [n=_____] at the grid point (*i,k+1*):

b. Evaluate $\left.\frac{\partial^2 U}{\partial x^2}\right|_{i,k+1}$ with ______ finite difference approximation of $\vartheta(\Delta x^n)$ [n=__________]

c. Approximate the 1st order space derivative $\frac{\partial U}{\partial x}$ with central finite difference approximation of $\vartheta(\Delta t^n)$ [n=_____] at the grid point (*i,k+1*):

d. Prove that the recurrence formulation with characteristic parameters, e.g., $\lambda = \frac{\Delta t}{\Delta x^2}$ and $\alpha = \frac{b\Delta t}{\Delta x}$ can be obtained from:

$$\frac{U_i^{K+1} - U_i^K}{\Delta t} = \left[\frac{U_{i-1}^{k+1} - 2U_i^{k+1} + U_{i+1}^{k+1}}{\Delta x^2}\right] + b\left[\frac{-\left(\frac{1}{2}\right)U_{i-1}^{k+1} + \left(\frac{1}{2}\right)U_{i+1}^{k+1}}{\Delta x}\right]$$

e. Obtain the recurrence formula

(3) Discretize the initial and boundary conditions.

Original Mathematical Expression	Discretized
$U(0,t) = T_1$ for t>0	
$U(L,t) = T_2$ for t>0	
$U(x,t_0) = T_0$ for 0<x<L	

Where L=10, $T_0 = 0, T_1 = 100, T_2 = 50$

(4) Show the BTCS computational molecule within the grid of your discretization.

see GRID above

Answer the following questions:

a) How many unknowns do you have at each time step?____________
b) How many times does the computational molecule fit in the grid during the computation of one time step?______

(5) **System of Equations**. According to your problem discretization and the number of unknowns find the appropriate number of equations to solve the system. Assume your work is for the first time step (i.e. $[t_1, U_i^1]$. We use indexed-variable algebraic equations—Not numbers yet.

(6) Express the problem with numbers for a clean system of linear equations and also in matrix form.

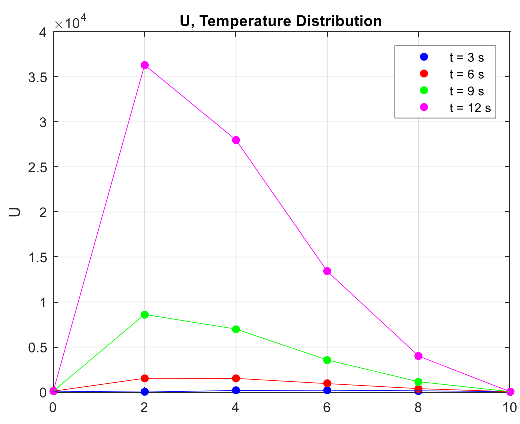
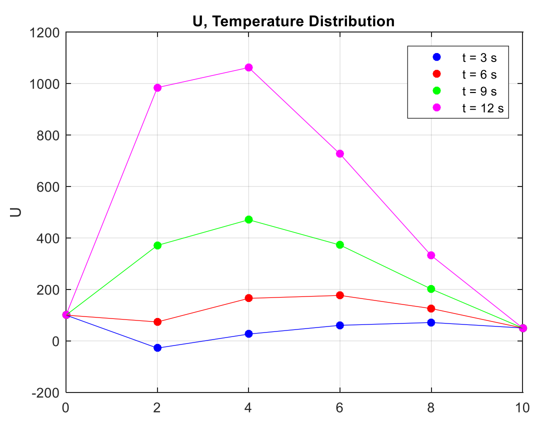
a. System of equations:

b. Matrix Form

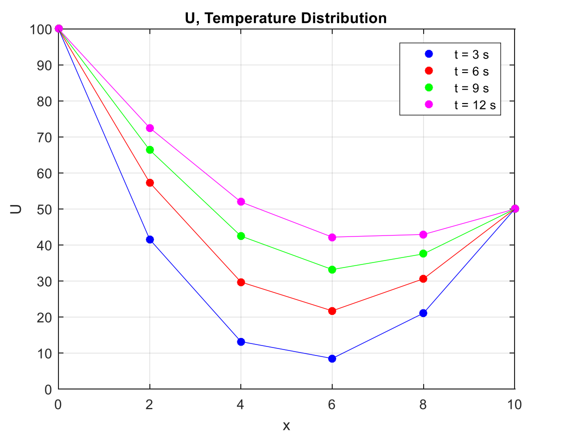
(7) Solve (6.b) with matlab code for (A) and (B) specifications above.

APPENDIX

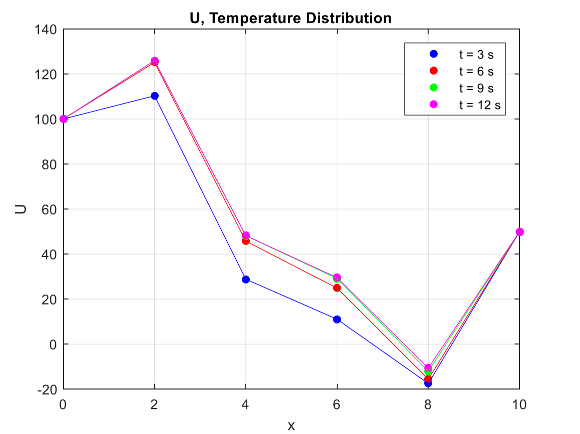
Below is a solution with the Crank Nicolson Scheme use it for testing your solution:

<table>
<tr><th>b=4</th><th>b=2</th></tr>
<tr><td>
</td><td>
</td></tr>
</table>

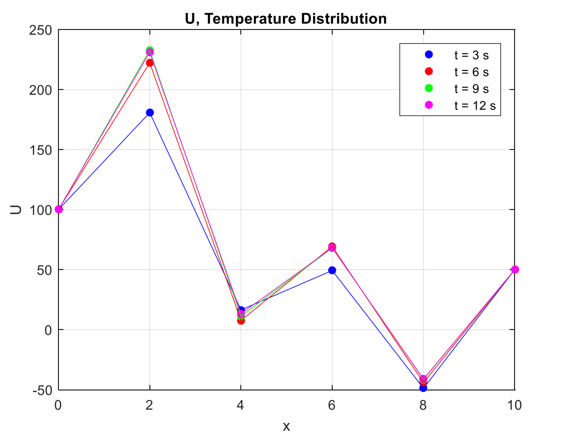
b=0



b=-2



b=-4



For b=4:

Command Window

New to MATLAB? See resources for Getting Started.

t	x(1)	x(2)	x(3)	x(4)	x(5)	x(6)
0.0	100.0	0.0	0.0	0.0	0.0	50.0
3.0	100.0	6.5	177.8	206.6	142.5	50.0
6.0	100.0	1529.2	1518.2	941.4	380.1	50.0
9.0	100.0	8616.1	7000.3	3567.8	1151.8	50.0
12.0	100.0	36322.3	28001.9	13410.1	3999.0	50.0

Elapsed time is 0.153795 seconds.

```
% The Crank Nicolson Method
% Example
% This CN solution uses theta=1/2 and the matlab backslash operator
% File: CrankNicolsonP5.m
```

```
clc, clear, close

tic

%---Compute parameters:
t(1)=0; tmax = 12;  dt =0.1;
t=[t(1):dt:tmax];           % t vector
nt = tmax/dt + 1;           % if t(1)=0, nt is number of total time steps (or nodes in time)
                            % i.e., k=[1,nt] number of nodes in t

L=10;                       % rod length
dx =2;                      % Delta x
bb = 4;                     % b parameter
La = dt/dx^2;               % Lambda
Al = bb*dt/dx;              % Alfa
nx=[L/dx]+1;                % number of nodes in x
x=([1:nx]-1).*dx;           % x-vector

% --- Constant Coefficients of the tridiagonal Matrix
b = Al+2*La;                % Super diagonal: coefficients of u(i+1)
c = Al-2*La;                % Subdiagonal: coefficients of u(i-1)
a = 4*(1 + La);             % Main Diagonal: coefficients of u(i)

% Boundary conditions and Initial Condition
Uo(1)=100;  Uo(2:nx-1)=0; Uo(nx)=50;
Un(1)=100; Un(nx)=50;

% Store results for future use
UUU(1,:)=Uo;

% Loop over time
for k=2:nt

   for ii=1:nx-2
      if ii==1  % d(1) has specific formula due to BC
         d(ii)=-c*Uo(ii)+(4+(c-b))*Uo(ii+1)+b*Uo(ii+2)-c*Un(1);
      elseif ii==nx-2 % d(nx-2) has specific formula due to BC
         d(ii)=-c*Uo(ii)+(4+(c-b))*Uo(ii+1)+b*Uo(ii+2)+b*Un(nx);
      else
         d(ii)=-c*Uo(ii)+(4+(c-b))*Uo(ii+1)+b*Uo(ii+2);
      end
   end  % d is a row vector
```

```
   % Transform a, b, c scalar constants in column vectors:
   bb=b*ones(nx-3,1);
   cc=bb;
   aa=a*ones(nx-2,1);

   % Use column vectors to construct tridiagonal matrix
   AA=diag(aa)+ diag(-bb,1)+ diag(-cc,-1);          % AA is triadiagonal Matrix

   % Find the solution for interior nodes i=2,3,4,5
   UU=AA\d';     % UU is temp at interior nodes only

   % Build the whole solution by including BCs
   Un=[Un(1),UU',Un(nx)];  % row vector

   % Store results for future use, one column at a time
   UUU(k,:)=Un;

   % to start over
   Uo=Un;

end

% Output
t=t';   % from row to column

uuu=[t,UUU];               % uuu stores whole solution: t plus u;  first column is t

fprintf('%10s %10s %10s %10s %10s %10s %10s\n','t','x(1)','x(2)','x(3)','x(4)','x(5)','x(6)');
   for jj=1:30:nt    % Output selected results
      for ii=1:nx+1
         fprintf('%10.1f ', uuu(jj,ii));
      end
      fprintf('\n');
   end
   fprintf('\n');


% For the graph

tPlot= 0:3:tmax;

plot(x,uuu(31,2:7), '.b','markers',20);
hold on                                         % allows to print other solutions in same graph
plot(x,uuu(61,2:7), '.r','markers',20);
```

```
plot(x,uuu(91,2:7), '.g','markers',20);
plot(x,uuu(121,2:7), '.m','markers',20);

line(x,uuu(31,2:7),'Color','b');
line(x,uuu(61,2:7),'Color','r');
line(x,uuu(91,2:7),'Color','g');
line(x,uuu(121,2:7),'Color','m');

title('U, Temperature Distribution');
ylabel('U');    xlabel('x');     legend('t = 3 s','t = 6 s','t = 9 s','t = 12 s');
% axis([0 10 -0.5e7 2.5e7]); % axis([xmin xmax ymin ymax]);
grid on
hold off                            % free the current figure

toc
```