FYE Take Home 2019 Re-take Report

Jianhong Chen

September 22, 2019

1 Summary

In this report, we have studied 1D heat equation with or without sources both analytically and numerically. Given the source term and its BC and IC, the original PDE is converted into a non-dimensional form. By utilizing separation of variables, the heat equation without source term is solved analytically. Then, three numerical methods, FTCS, BTCS, and Crank-Nicolson(C-N) are applied to solve the same problem computationally. Before implementing the numerical methods, the stablility condition of each method is checked by Von Neumann analysis. For both BTCS and C-N methods, because they are implicit, the problems are eventually converted into $Ax = b$ problem. To solve such classic linear algebra problem, Gaussian elimination and Jacobi iteration scheme are implemented for BTCS and C-N respectively. The numerical results are plotted at different time instances. Furthermore, the numerical errors, number of time step for the function to reach steady state, and the convergent rate of Jacobi iteration are recorded and studied.

2 Question 1

$$\frac{\partial U}{\partial T} = a\frac{\partial^2 U}{\partial X^2} + bU, 0 \le X \le L. \tag{1}$$

2.1 (i)

To convert eqn.(1) into non-dimensional form, the following are given,

$$x = \frac{X}{L},\ u = \frac{U}{U^0}.$$

Take the derivative of the two given expressions,

$$dx = \frac{dX}{L},\ dx^2 = \frac{dX^2}{L},$$

$$du = \frac{dU}{U^0},\ du^2 = \frac{dU^2}{U^0}.$$

Plug the derivative back into the PDE,

$$U^0\frac{\partial u}{\partial T} = \frac{aU^0}{L}\frac{\partial^2 u}{\partial x^2} + bU^0 u.$$

Cancel out U^0 and re-arrange,

$$\frac{L}{a}\frac{\partial u}{\partial T} = \frac{\partial^2 u}{\partial x^2} + \frac{bL}{a}u.$$

Let's redefine some new terms,

$$\frac{1}{\partial t} = \frac{L}{a}\frac{1}{\partial T},\ \hat{b} = \frac{bL}{a}.$$

Hence, we obtained a non-dimensional form of the original PDE with $L = 1$,

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

where

$$\boxed{t = aT},\ \boxed{\hat{b} = \frac{b}{a}}.$$

2.2 (ii)

Using central difference in space, the semi-discretized form of eqn (1),

$$\frac{du_i^n}{dt} = \frac{1}{(\Delta x)^2}(u_{i+1}^n - 2u_i^n + u_{i-1}^n) + \hat{b}u_i^n.$$

First, apply forward Euler method(FTCS),

$$u_i^{n+1} = u_i^n + \Delta t f(u_i^n, t_n),$$

$$u_i^{n+1} = u_i^n + \frac{\Delta t}{(\Delta x)^2}(u_{i+1}^n - 2u_i^n + u_{i-1}^n) + \Delta t\hat{b}u_i^n,$$

$$\boxed{u_i^{n+1} = (1 + \Delta t\hat{b})u_i^n + r(u_{i+1}^n - 2u_i^n + u_{i-1}^n)},\ \text{where } r = \frac{\Delta t}{(\Delta x)^2}.$$

Next, apply backward Euler method(BTCS),

$$u_i^{n+1} = u_i^n + \Delta t f(u_i^{n+1}, t_{n+1}),$$

$$u_i^{n+1} = u_i^n + \frac{\Delta t}{(\Delta x)^2}(u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}) + \Delta t\hat{b}u_i^{n+1},$$

$$\boxed{(1 - \Delta t\hat{b})u_i^{n+1} - r(u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}) = u_i^n},\ \text{where } r = \frac{\Delta t}{(\Delta x)^2}.$$

2.3 (iii)

Apply Von Neumann analysis to check the stability of the two methods from part (ii),

first, check FTCS by plugging in $u_i^n = \rho^n exp(\sqrt{-1}\xi i\Delta x)$,

$\rho^{n+1}exp(\sqrt{-1}\xi i\Delta x) = (1+\Delta t\hat{b})\rho^n exp(\sqrt{-1}\xi i\Delta x) + r\rho^n exp(\sqrt{-1}\xi i\Delta x)(e^{\sqrt{-1}\xi\Delta x} - 2 + e^{-\sqrt{-1}\xi\Delta x}).$

Divide both sides by $\rho^n exp(\sqrt{-1}\xi i\Delta x)$,

$$\rho = (1 + \Delta t\hat{b}) + 2r(cos(\xi\Delta x) - 1),$$

$$\rho = (1 + \Delta t\hat{b}) - 4r(sin^2(\frac{\xi\Delta x}{2})).$$

For the method to be stable, $|\rho| \leq 1 + C\Delta t$. Since ξ can take any value, $0 \leq sin^2(\frac{\xi\Delta x}{2}) \leq 1$,

$$\rho = (1 + \Delta t\hat{b}) - 4r, \text{ and } \rho \geq -1.$$

Re-arrange and re-write in term of Δt,

$$(\frac{4}{(\Delta x)^2} - \hat{b})\Delta t \leq 2,$$

$$\boxed{\Delta t \leq \frac{2}{\frac{4}{(\Delta x)^2} - \hat{b}}}.$$

Therefore, given Δx, Δt has to satisfy this relation in order for the method(FTCS) to be stable.

For special case, Heat equation without source($\hat{b} = 0$),

$$\Delta t \leq \frac{2}{\frac{4}{(\Delta x)^2}} \longrightarrow \Delta t \leq \frac{(\Delta x)^2}{2},$$

which is $r \leq \frac{1}{2}$, what we have derived in class!

Next, perform the same analysis for BTCS method by similar procedures,

$(1-\Delta t\hat{b})\rho^{n+1}exp(\sqrt{-1}\xi i\Delta x)-r\rho^{n+1}exp(\sqrt{-1}\xi i\Delta x)(e^{\sqrt{-1}\xi\Delta x}-2+e^{-\sqrt{-1}\xi\Delta x}) = \rho^n exp(\sqrt{-1}\xi i\Delta x)$.

Divide both sides by $\rho^n exp(\sqrt{-1}\xi i\Delta x)$,

$$(1 - \Delta t\hat{b})\rho - 2r(cos(\xi\Delta x) - 1)\rho = 1,$$

$$(1 - \Delta t\hat{b})\rho + 4rsin^2(\frac{\xi\Delta x}{2})\rho = 1,$$

$$\boxed{\rho = \frac{1}{1 - \Delta t\hat{b} + 4rsin^2(\frac{\xi\Delta x}{2})}}.$$

Again, given ξ can take any values, $0 \leq sin^2(\frac{\xi\Delta x}{2}) \leq 1$,

$$\rho = \frac{1}{1 - \Delta t\hat{b} + 4r}.$$

Since Δx and Δt are positive and usually small (< 1),

$$|\rho| \leq 1, \text{ for any } \xi, \ r, \ \hat{b}.$$

Therefore, BTCS is unconditional stable.

3 Question 2

$$\begin{cases} u_t = u_{xx}, \ 0 \leq x \leq 1, \\ IC : u(x, 0) = 2x, if \ 0 \leq x \leq 1/2, \\ \qquad u(x, 0) = 2(1 - x), if \ 1/2 \leq x \leq 1, \\ BC : u(0, t) = u(1, t) = 0, \ for \ all \ t > 0 \end{cases} \tag{2}$$

3.1 (i)

By separation of variables,

$$u(x,t) = X(x)T(t).$$

The heat equation from eqn(2),

$$X(x)T'(t) = X''(x)T(t), \Longrightarrow \frac{T'}{T} = \frac{X''}{X} = -k^2.$$

$$T(t) = e^{-k^2 t}, \quad X(x) = Acos(kx) + Bsin(kx).$$

Apply BC for $X(x)$,

$$X(0) = 0 \Longrightarrow A = 0,$$

$$X(1) = 0 \Longrightarrow Bsin(k) = 0 \Longrightarrow k = n\pi.$$

Hence, the general solution is,

$$u(x,t) = \sum_{n=1}^{\infty} a_n sin(n\pi x) e^{-(n\pi)^2 t}.$$

To find a_n, apply IC,

$$\sum_{n=1}^{\infty} a_n sin(n\pi x) = u(x,0),$$

By the definition of Fourier series coefficient,

$$a_n = \frac{2}{L}\int_0^L f(x)sin(n\pi x/L)dx = 2\int_0^1 u(x,0)sin(n\pi x)dx,$$

$$a_n = 4(\int_0^{1/2} xsin(n\pi x)dx + \int_{1/2}^1 (1-x)sin(n\pi x)dx).$$

Solve the two integrals separately, and apply integration by parts where needed,

$$I_1 = \int_0^{1/2} xsin(n\pi x)dx = -\frac{x}{n\pi}cos(n\pi x)|_0^{1/2} + \frac{1}{n\pi}\int_0^{1/2} cos(n\pi x)dx,$$

$$I_1 = 0 + \frac{1}{(n\pi)^2}sin(n\pi x)|_0^{1/2} = \frac{1}{(n\pi)^2}sin(\frac{n\pi}{2}).$$

$$I_2 = \int_{1/2}^1 sin(n\pi x)dx - \int_{1/2}^1 xsin(n\pi x)dx.$$

Again apply integration by parts(some steps are skipped),

$$I_2 = -\frac{1}{(n\pi)^2}sin(n\pi x)|_{1/2}^1 = \frac{1}{(n\pi)^2}sin(\frac{n\pi}{2}).$$

$$a_n = 4(I_1 + I_2) = \frac{8}{(n\pi)^2}sin(\frac{n\pi}{2}).$$

Finally, the general solution for the problem,

$$\boxed{u(x,t) = \sum_{n=1}^{\infty} \frac{8}{(n\pi)^2}sin(\frac{n\pi}{2})sin(n\pi x)e^{-(n\pi)^2 t}.}$$

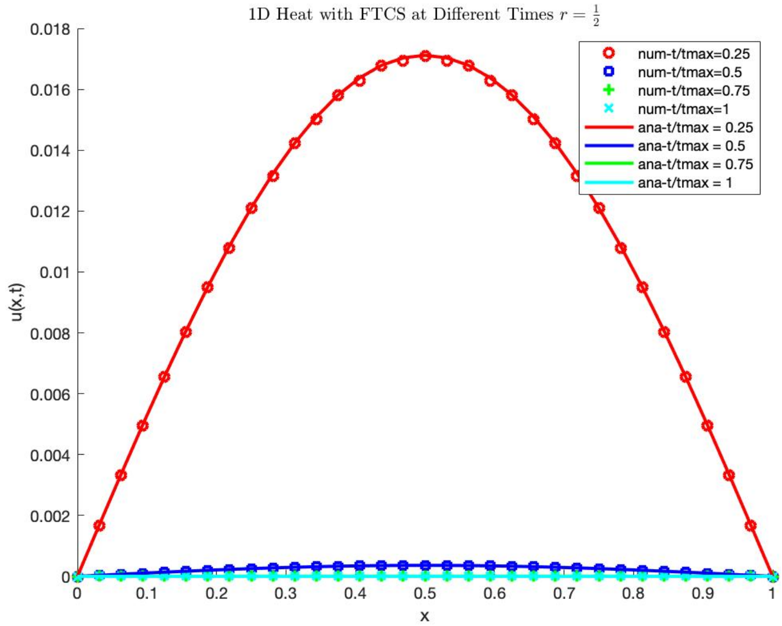

Figure 1: FTCS method for 1D heat equation with different time points.

3.2 (ii)

Given $\hat{b} = 0$ and $\Delta x = 1/32$, $\Delta t \leq \frac{(\Delta x)^2}{2}$, **I simply chose the boundary value for** $\boxed{\Delta t = \frac{1}{2048}}$ **which is good enough for FTCS method to be stable.** The number of step it takes to reach steady state is $\boxed{n_{ss} = 3202 \Longrightarrow t_{max} = n_{ss}\Delta t = 1.5635.}$
The numerical results are shown at figure(1).

3.3 (iii)

$$\theta \frac{u_i^{n+1} - u_i^{n-1}}{2\Delta t} + (1-\theta)\frac{u_i^n - u_i^{n-1}}{\Delta t} = \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{(\Delta x)^2} \tag{3}$$

Re-arrange and define $r = \frac{\Delta t}{(\Delta x)^2}$,

$$\frac{\theta}{2}(u_i^{n+1} - u_i^{n-1}) + (1-\theta)(u_i^n - u_i^{n-1}) = r(u_{i+1}^n - 2u_i^n + u_{i-1}^n).$$

LHS $= \frac{\theta}{2}(u_i^{n+1} - u_i^{n-1}) + (1-\theta)(u_i^n - u_i^{n-1})$, and perform Taylor expansion around (x_i, t_n),

$$\frac{\theta}{2}[(u + u_t\Delta t + \frac{u_{tt}}{2}(\Delta t)^2) - (u - u_t\Delta t + \frac{u_{tt}}{2}(\Delta t)^2] + [(1-\theta)[u - (u - u_t\Delta t + \frac{u_{tt}}{2}(\Delta t)^2)],$$

$$LHS = \frac{\theta}{2}(2u_t\Delta t) + (1-\theta)(u_t\Delta t - \frac{u_{tt}}{2}(\Delta t)^2),$$
$$LHS = u_t\Delta t + (\theta - 1)\frac{u_{tt}}{2}(\Delta t)^2 + H.O.T.$$

Perform Taylor expansion around (x_i, t_n) for the RHS,

$$RHS = r(2\frac{u_{xx}}{2}(\Delta x)^2 + 2\frac{u_{xxxx}}{4!}(\Delta x)^4 + H.O.T).$$

Plug in $r = \frac{\Delta t}{(\Delta x)^2}$ and compute truncation error(TE),

$$TE = \frac{LHS - RHS}{\Delta t} = (u_t - u_{xx}) + (\theta - 1)\frac{u_{tt}}{2}\Delta t - \frac{u_{xxxx}}{12}(\Delta x)^2 + O((\Delta t)^2 + (\Delta x)^4).$$

Given $u_t = u_{xx}$,

$$\boxed{TE = (\theta - 1)\frac{u_{tt}}{2}\Delta t - \frac{u_{xxxx}}{12}(\Delta x)^2 + O((\Delta t)^2 + (\Delta x)^4).}$$

where,

$$\boxed{A = (\theta - 1)\frac{u_{tt}}{2}}, \quad \boxed{B = -\frac{u_{xxxx}}{12}}, \quad \boxed{p = 2,\ q = 4}.$$

To find θ_{opt} that improves the order of accuracy of the method,

$$\frac{(\theta - 1)}{2} - \frac{1}{12} = 0,$$

$$12(\theta - 1) = 2 \Longrightarrow \boxed{\theta_{opt} = \frac{7}{6}}.$$

3.4 (iv)

In this section, Von Neumann analysis is performed to check the stability of eqn(3) for both $\theta = 1$ and $\theta = 0$.
First start with $\theta = 1$,

$$\frac{1}{2}(u_i^{n+1} - u_i^{n-1}) = r(u_{i+1}^n - 2u_i^n + u_{i-1}^n).$$

Plugging in $u_i^n = \rho^n exp(\sqrt{-1}\xi i\Delta x)$,

$$\frac{1}{2}(\rho^{n+1} exp(\sqrt{-1}\xi i\Delta x) - \rho^{n-1} exp(\sqrt{-1}\xi i\Delta x)) = r\rho^n exp(\sqrt{-1}\xi i\Delta x)(e^{\sqrt{-1}\xi\Delta x} - 2 + e^{-\sqrt{-1}\xi\Delta x}).$$

Divide both sides by $\rho^{n-1} exp(\sqrt{-1}\xi i\Delta x)$,

$$\frac{1}{2}(\rho^2 - 1) = -4r\rho(sin^2(\frac{\xi\Delta x}{2})).$$

Since ξ can takes any value, $sin^2(\frac{\xi\Delta x}{2}) = 1$ at most,

$$\frac{1}{2}\rho^2 + 4r\rho - \frac{1}{2} = 0,$$

$$\rho = \frac{-4r \pm \sqrt{16r^2 + 1}}{1} \approx -4r \pm 4r,$$

$$\rho = 0 \ or \ 8r = 8r \text{ (for non-trivial solution)}.$$

However,

$$\boxed{|\rho| = 8r > 1 + C\Delta t}.$$

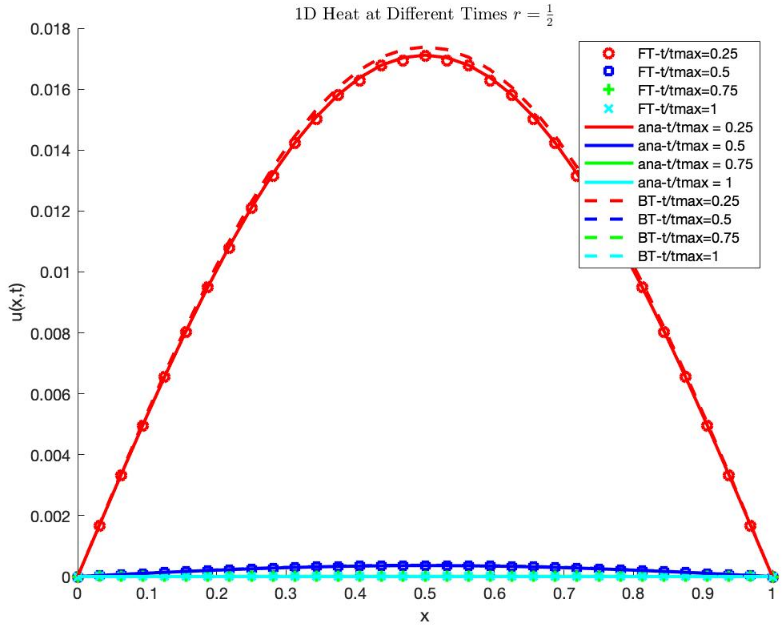

Figure 2: 1D Heat Equation with BTCS at different time

Therefore, the method when $\theta = 1$ is unstable.

Next, perform similar investigation when $\theta = 0$,

$$u_i^n - u_i^{n-1} = r(u_{i+1}^n - 2u_i^n + u_{i-1}^n).$$

This is exactly BTCS method for heat equation without source term, which is unconditionally stable for any r and ξ from Q1 part (iii).

3.5 (v)

The numerical results of the BTCS methods are shown in figure (2).

Because BTCS is implicit and we have to solve $Ax = b$ problem at the end, the linear algebra solver I choose is Gaussian elimination(GE). Because A is a sparse tridiagonal matrix which is already a nice condition for the GE algorithm, the algorithm will guarantee convergent once A is constructed correctly.
The table shows the different time steps, $\Delta t = \frac{C}{2}(\Delta x)^2$, are required to reach steatdy state for given different time step sizes,

C	n_{ss}
0.1	18653
1.0	2346
10	286

4 Question 3

4.1 (i)

Apply Crank-Nicolson method for the heat equation, eqn(2),

$$u_i^{n+1} = u_i^n + \frac{1}{2}\Delta t(f(u_i^n, t_n) + f(u_i^{n+1}, t_{n+1})),$$

$$u_i^{n+1} = u_i^n + \frac{1}{2}r(u_{i+1}^n - 2u_i^n + u_{i-1}^n) + \frac{1}{2}r(u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}), \quad r = \frac{\Delta t}{(\Delta x)^2}.$$

Convert spatial discretized terms into matrix-vector form, let $(u_{i+1}^n - 2u_i^n + u_{i-1}^n) = A_x U^n + b$, however, $b = 0$ in this case because of the zero BC,

$$(I - \frac{1}{2}rA_x)U^{n+1} = (I + \frac{1}{2}rA_x)U^n.$$

Finally, convert the method into $Ax = b$ form,

$$AX^{n+1} = (I + \frac{1}{2}rA_x)X^n, \ \boxed{A = I - \frac{1}{2}rA_x}, \text{ and, } \boxed{b = (I + \frac{1}{2}rA_x)X^0}.$$

From this, apply Jacobi iteration to solve $Ax = b$ problem,

$$A = D + R,$$

where D contains only the diagonal elements of A and R contains the rest elements of A.

$$(D + R)X = b \Longrightarrow DX = b - RX \Longrightarrow X = D^{-1}(b - RX).$$

$$X^{k+1} = -D^{-1}RX^k + D^{-1}b, \ \boxed{J = -D^{-1}R}, \ \boxed{c = D^{-1}b}.$$

4.2 (ii)

The numerical results of Crank-Nicolson are shown at figure(3).

For $N = 32$ and $\Delta t = 5 * 10^{-4}$, the step it takes to reach steady state is, $\boxed{n_{ss} = 3133}$, and number of step for the Jacobi iteration to converge at each time step is, $\boxed{N_{Jacobi} = 22}$, and finally the error between the numerical solution by Crank-Nicolson and the exact solution(analytic solution) is, $\boxed{error = 2.1714 * 10^{-8}}$.

4.3 (iii)

Δt	N_{Jacobi}	error
$5 * 10^{-5}$	10	$3.8276 * 10^{-9}$
$5 * 10^{-4}$	22	$2.1714 * 10^{-8}$
$5 * 10^{-3}$	96	$4.6935 * 10^{-7}$

For Jacobi iteration algorithm to converge, the spectral radius of J, has to satisfy the condition, $\rho(J) < 1$. Given $J = -D^{-1}R$, and both D and R depend on r since $A = D + R = I - \frac{1}{2}rA_x$. If Δt decreases, r decreases which also leads to A decreases. As a result, the elements of both D and R are smaller number. Then, the eigenvalues of J will become smaller, and ultimately, J is more diagonally dominant. **Therefore, it takes less steps for the algorithm to converge as Δt decreases.**

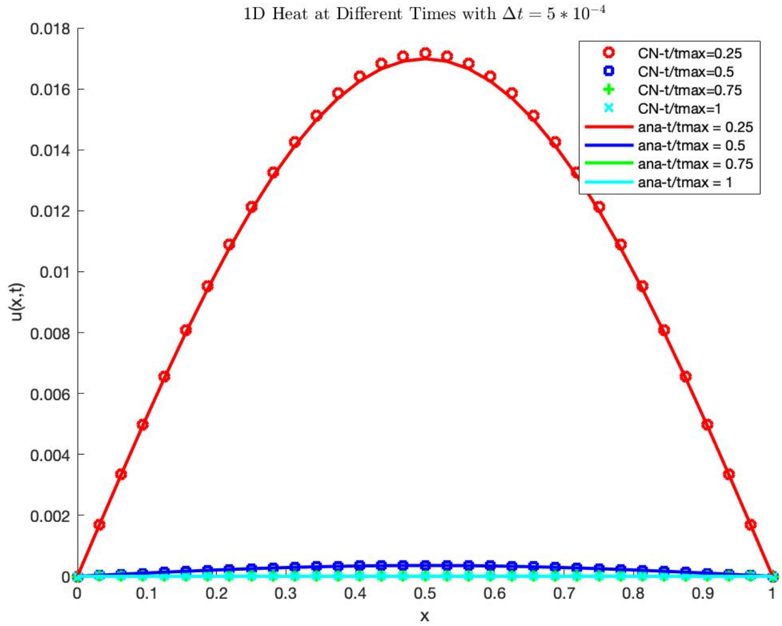

Figure 3: 1D Heat Equation with Crank-Nicolson at Different Times

5 Matlab Codes

5.1 Q2 Heat FTCS

```
%FYE take home 2019 retake Q2 part (ii)
% solve 1D Heat eqn with FTCS method.
% then compare numerical results vs analytic soluntions
% u_t = u_xx

% Jianhong Chen
% 09 20 2019

clear all
clc

% problem setup parameters
dx = 1/32; % given
dt = (dx^2)/2; % compute the stable time step
t0 = 0;
T = 5; % Terminal time that is long enough to reach steady state
a = 0;
b = 1;
Nx = (b a)/dx   1; %internal grid
Nt = (T t0)/dt; % time step point
r = dt/(dx^2);

% IC functions
f1 = @(x) 2*x;
f2 = @(x) 2*(1 x);

% analytic soln funciton
F = @(n,x,t) 8/(n*pi).^2 .*sin(n*pi/2).*sin(n*pi*x).*exp( (n*pi).^2.*t);

x = a + (1:Nx)*dx;%construct x step vector
x = [a;x.';b]; % transpose of A in matlab A.'

% allocate matrix vector arrays
u = zeros(Nx+2, Nt);

% BC
u(1, :) = 0;
u(Nx+2, :) = 0;
% IC
u(2:17, 1) =f1(x(2:17));
u(17:32,1) = f2(x(17:32));

% important!!! the whole Heat euqantion in space has t obe solve for each
% time step.
for n = 1:Nt % time has to be in the outer loop
```

```
    for i = 2:Nx+1 % solve all space first for every timeframe
        u(i,n+1) = u(i,n) + r*(u(i+1, n)    2*u(i,n) + u(i 1, n));
    end
end

% find t_max by computing the temporal change of the numerical soln
n_ss = 1; % steady state step
sigma = 10ˆ 6;
u_res = 1; %tempolar change residue

while u_res > sigma
    u_res = sum(abs(u(:,n_ss+1)    u(:,n_ss)))/(dt*(Nx+1));
    n_ss = n_ss + 1;
end

t_max = n_ss*dt; %  compute time that take to reach steady state

T_save = [0.25, 0.5, 0.75, 1]*t_max;
n_save = round(T_save/dt);

% now compute analytic soln as the exact soln
u_ana = zeros(Nx+2, length(T_save));
n_ana = 1:30; % only sum up the first 30 terms

for t = 1:length(T_save) % time loop
    u_dummy = zeros(length(n_ana),1);
    for i = 1:length(x) % x loop
        for n = n_ana % compute each term for n
            u_dummy(n) = F(n,x(i),T_save(t));
        end
        u_ana(i,t) = sum(u_dummy);% sum up all n terms
    end
end

% save workspace variables
save dat_heat_FTCS

%
figure(1)
clf
hold on
plot(x, u(:, n_save(1)), 'ro', 'linewidth',2)
plot(x, u(:, n_save(2)), 'bs','linewidth',2)
plot(x, u(:, n_save(3)), 'g+', 'linewidth',2)
plot(x, u(:, n_save(4)), 'cx', 'linewidth',2)

plot(x, u_ana(:,1), 'r ', 'linewidth',2)
plot(x, u_ana(:,2), 'b ', 'linewidth',2)
plot(x, u_ana(:,3), 'g ', 'linewidth',2)
plot(x, u_ana(:,4), 'c ', 'linewidth',2)

legend('num t/tmax=0.25', 'num t/tmax=0.5', ...
    'num t/tmax=0.75', 'num t/tmax=1', ...
    "ana t/tmax = 0.25", "ana t/tmax = 0.5", ...
    "ana t/tmax = 0.75", "ana t/tmax = 1")
xlabel("x")
ylabel("u(x,t)")
title("1D Heat with FTCS at Different Times $r = \frac{1}{2}$", ...
    'interpreter', 'latex')
```

5.2 Q2 Heat BTCS

```
%FYE take home 2019 retake Q2 part (ii)
% solve 1D Heat eqn with BTCS method.
% then compare numerical results vs analytic soluntions
% u_t = u_xx

% Jianhong Chen
% 09 20 2019

clear all
clc

% load the same parameters from previous solution
load dat_heat_FTCS

% allocate matrix vector arrays
u_BT = zeros(Nx+2, n_save(end));

% BC
u_BT(1, :) = 0;
u_BT(Nx+2, :) = 0;
% IC
u_BT(2:17, 1) =f1(x(2:17));
u_BT(17:32,1) = f2(x(17:32));

% BTCS method has the form
% u(n+1) = u(n) + r*(Ax*u(n+1)
% due to zero BC, b vector is zero in this case

% contruct sparse matrix Ax
Ia=zeros(1,3*Nx);     % for storing row indices of non zero entries
```

```
Ja=zeros(1,3*Nx);     % for storing column indices of non zero entries
Sa=zeros(1,3*Nx);     % for storing values of non zero entries
for i=1:Nx
  Ia(3*(i 1)+[1:3])=[i,i,i];
  Ja(3*(i 1)+[1:3])=[i 1,i,i+1];
  Sa(3*(i 1)+[1:3])=[1,  2,  1];
end
ind=find( (Ja>0)&(Ja<Nx+1) );
Ax=sparse(Ia(ind),Ja(ind),Sa(ind),Nx,Nx);
% rearrange BTCS into Ax = b form
% (I r*Ax)*u(n+1) = u(n)
% thus A = (I    r*Ax)
A = speye(Nx,Nx)    r*Ax;
clear Ax

for n = 1:n_save(end) % time loop
    % only update internal points
    % in this case it doesn't matter, becasue BC are zeros
    b = u_BT(2:Nx+1,n);
    %u_BT(2:Nx+1,n+1) = A\b;
    u_BT(2:Nx+1,n+1) = GaussianElimination(A, b); % solve Ax = b by GE
end
%
figure(1)
clf
hold on
plot(x, u(:, n_save(1)), 'ro', 'linewidth',2)
plot(x, u(:, n_save(2)), 'bs','linewidth',2)
plot(x, u(:, n_save(3)), 'g+', 'linewidth',2)
plot(x, u(:, n_save(4)), 'cx', 'linewidth',2)

plot(x, u_ana(:,1), 'r ', 'linewidth',2)
plot(x, u_ana(:,2), 'b ', 'linewidth',2)
plot(x, u_ana(:,3), 'g ', 'linewidth',2)
plot(x, u_ana(:,4), 'c ', 'linewidth',2)

plot(x, u_BT(:,n_save(1)), 'r  ', 'linewidth',2)
plot(x, u_BT(:,n_save(2)), 'b  ', 'linewidth',2)
plot(x, u_BT(:,n_save(3)), 'g  ', 'linewidth',2)
plot(x, u_BT(:,n_save(4)), 'c  ', 'linewidth',2)

legend('FT t/tmax=0.25', 'FT t/tmax=0.5', ...
    'FT t/tmax=0.75', 'FT t/tmax=1', ...
    "ana t/tmax = 0.25", "ana t/tmax = 0.5", ...
    "ana t/tmax = 0.75", "ana t/tmax = 1", ...
    'BT t/tmax=0.25', 'BT t/tmax=0.5', ...
    'BT t/tmax=0.75', 'BT t/tmax=1')
xlabel("x")
ylabel("u(x,t)")
title("1D Heat at Different Times $r = \frac{1}{2}$", ...
    'interpreter', 'latex')
```

5.3 Q2 Heat BTCS Steady State Analysis

```
%FYE take home 2019 retake Q2 part (ii)
% solve 1D Heat eqn with BTCS method.
% then analysis number of steps it takes to reach steady state vs vary
% time step dt.
% u_t = u_xx

% Jianhong Chen
% 09 20 2019

clear all
clc

% load the same parameters from previous solution

dx = 1/32; % given
C = 10; %constant choice for time step
dt = (dx^2)*C/2; % compute the stable time step
t0 = 0;
T = 5;
a = 0;
b = 1;
Nx = (b a)/dx    1; %internal grid
Nt = (T t0)/dt;
r = dt/(dx^2);

% IC functions
f1 = @(x) 2*x;
f2 = @(x) 2*(1 x);

% analytic soln funciton
F = @(n,x,t) 8/(n*pi).^2 .*sin(n*pi/2).*sin(n*pi*x).*exp( (n*pi).^2.*t);

x = a + (1:Nx)*dx;
x = [a;x.';b]; % transpose of A in matlab A.'

% allocate matrix vector arrays
u_BT = zeros(Nx+2, Nt);

```

```
% BC
u_BT(1, :) = 0;
u_BT(Nx+2, :) = 0;
% IC
u_BT(2:17, 1) =f1(x(2:17));
u_BT(17:32,1) = f2(x(17:32));

% BTCS method has the form
% u(n+1) = u(n) + r*(Ax*u(n+1)
% due to zero BC, b vector is zero in this case


% contruct sparse matrix Ax
Ia=zeros(1,3*Nx);     % for storing row indices of non zero entries
Ja=zeros(1,3*Nx);     % for storing column indices of non zero entries
Sa=zeros(1,3*Nx);     % for storing values of non zero entries
for i=1:Nx
  Ia(3*(i 1)+[1:3])=[i,i,i];
  Ja(3*(i 1)+[1:3])=[i 1,i,i+1];
  Sa(3*(i 1)+[1:3])=[1,  2,  1];
end
ind=find( (Ja>0)&(Ja<Nx+1) );
Ax=sparse(Ia(ind),Ja(ind),Sa(ind),Nx,Nx);
% rearrange BTCS into Ax = b form
% (I r*Ax)*u(n+1) = u(n)
% thus A = (I    r*Ax)
A = speye(Nx,Nx)     r(1)*Ax;
clear Ax

sigma = 10^ 6;

for n = 1:Nt % time loop
    % only update internal points
    % in this case it doesn't matter, becasue BC are zeros
    b = u_BT(2:Nx+1,n);
    %u_BT(2:Nx+1,n+1) = A\b;
    u_BT(2:Nx+1,n+1) = GaussianElimination(A, b); % solve Ax = b by GE
    u_res = 1/(dt*Nx+1)* sum(abs(u_BT(:,n+1)    u_BT(:,n)));
    if u_res < sigma
        n_ss = n;
        break % break the time loop once reach steady state
    end
end

t_max = n_ss*dt;
```

5.4 Q3 Heat Crank-Nicolson

```
%FYE take home 2019 retake Q3 part (ii)
% solve 1D Heat eqn with Crank Nicolson method.
% then compare numerical results vs analytic soluntions
% u_t = u_xx

% Jianhong Chen
% 09 20 2019

clear all
clc

% problem setup parameters
N = 32;
dx = 1/N; % given
dt = 5*10^ 4; % compute the stable time step
t0 = 0;
T = 5;
a = 0;
b = 1;
Nx = (b a)/dx    1; %internal grid
Nt = round((T t0)/dt);
r = dt/(dx^2);

% IC functions
f1 = @(x) 2*x;
f2 = @(x) 2*(1 x);

% analytic soln funciton
F = @(n,x,t) 8/(n*pi).^2 .*sin(n*pi/2).*sin(n*pi*x).*exp( (n*pi).^2.*t);

x = a + (1:Nx)*dx;
x = [a;x.';b]; % transpose of A in matlab A.'

u_CN = zeros(Nx+2, Nt);
% BC
u_CN(1, :) = 0;
u_CN(Nx+2, :) = 0;
% IC
u_CN(2:17, 1) =f1(x(2:17));
u_CN(17:32,1) = f2(x(17:32));

% CTCS method has the form
% u(n+1) = u(n) + r/2*(Ax(u(n)) +  r/2*(Ax*u(n+1))
% due to zero BC, b vector is zero in this case
```

```
% contruct sparse matrix Ax
Ia=zeros(1,3*Nx);     % for storing row indices of non zero entries
Ja=zeros(1,3*Nx);     % for storing column indices of non zero entries
Sa=zeros(1,3*Nx);     % for storing values of non zero entries
for i=1:Nx
  Ia(3*(i 1)+[1:3])=[i,i,i];
  Ja(3*(i 1)+[1:3])=[i 1,i,i+1];
  Sa(3*(i 1)+[1:3])=[1,  2,  1];
end
ind=find( (Ja>0)&(Ja<Nx+1) );
Ax=sparse(Ia(ind),Ja(ind),Sa(ind),Nx,Nx);
% rearrange CN method into Ax = b form
% (I r/2*Ax)*u(n+1) = (I+r/2*Ax)*u(n)
% thus A = (I    r*Ax)
A = speye(Nx,Nx)    r/2*Ax;

% impementing Jacobi iteration scheme

%construct D and R matrix
D = diag(diag(A)); % only the diagonal elements of A
R = A   D; % the rest elements except the diagonal ones of A
% inverse of diagonal matrix is simply the reciprocal
D_inv = diag(1./diag(A));
% compute J matrix
J =  D_inv * R;

% iterate the Jacobi scheme forward in time
N_Jacobi = zeros(Nt,1); %record the step for convergent for Jacobi method
for n = 1:Nt
    b = (speye(Nx,Nx)+r/2*Ax)*u_CN(2:Nx+1, n);
    c = D_inv *b;
    sigma = 10^ 6;
    X = zeros(Nx, 100);
    J_res = 1; % Jacobi iteration error
    k = 1;
    X(:,1) = 1; % initial guess of X for Jacobi scheme
    while J_res > sigma
        X(:,k+1) = J*X(:,k) + c;
        J_res = 1/(dt*N)*sum(abs(X(:,k+1)   X(:,k)));
        k = k +1;
    end
    N_Jacobi(n) = k; %save max Jacobi iteration step
    u_CN(2:Nx+1, n+1) = X(:,k);
    % check if steady state is reached
    u_res = 1/(dt*N)*sum(abs(u_CN(:,n+1)   u_CN(:,n)));
    if u_res < sigma
        n_ss = n;
        break % stop the time loop once reach steady state
    end

end

t_max = n_ss*dt;
T_save = [0.25, 0.5, 0.75, 1]*t_max;
n_save = round(T_save/dt);

% now compute analytic soln for as the benchmark comparison
u_ana = zeros(Nx+2, length(T_save));
n_ana = 1:30; % only sum up the first 30 terms

for t = 1:length(T_save)
    u_temp = zeros(length(n_ana),1);
    for i = 1:length(x)
        for n = n_ana
            u_temp(n) = F(n,x(i),T_save(t));
        end
        u_ana(i,t) = sum(u_temp);
    end
end

% compute error of CN with Jacobi scheme
error = sum(abs(u_CN(:,n_save(end))   u_ana(:,4)))/N;
disp("the error at t_max is ")
disp(error)


figure(1)
clf
hold on

plot(x, u_CN(:, n_save(1)), 'ro', 'linewidth',2)
plot(x, u_CN(:, n_save(2)), 'bs','linewidth',2)
plot(x, u_CN(:, n_save(3)), 'g+', 'linewidth',2)
plot(x, u_CN(:, n_save(4)), 'cx', 'linewidth',2)

plot(x, u_ana(:,1), 'r ', 'linewidth',2)
plot(x, u_ana(:,2), 'b ', 'linewidth',2)
plot(x, u_ana(:,3), 'g ', 'linewidth',2)
plot(x, u_ana(:,4), 'c ', 'linewidth',2)

legend('CN t/tmax=0.25', 'CN t/tmax=0.5', ...
    'CN t/tmax=0.75', 'CN t/tmax=1', ...
    "ana t/tmax = 0.25", "ana t/tmax = 0.5", ...
    "ana t/tmax = 0.75", "ana t/tmax = 1")
```

```
xlabel("x")
ylabel("u(x,t)")
title("1D Heat at Different Times with $\Delta t = 5* 10^{ 4}$", ...
    'interpreter', 'latex')
```

5.5 Function Gaussian Elimination

```
function X = GaussianElimination(A,b)
% perform Gaussian Elimination to solve Ax=b problem


% Jianhong Chen
% 09 21 2019


% useful parameters
[m,~] = size(A);

% perform Gaussian elimination to obtain upper trianglar matrix A
for j = 1:m 1
    for i = j+1:m
        a_ratio = A(i,j)/A(j,j);
        A(i, :) = A(i, :)   A(j, :) * a_ratio;
        b(i) = b(i)   b(j) * a_ratio;

    end
end
%

% perform back substitution to compute X vector (soln)
% calculate the solution of last row
X(m) = b(m)/A(m, m);
for i = m 1: 1:1 % row iteration start from the bottom
  sum = 0;
  for j = i+1:m % column iteration
    %sum up all the values from the solved terms
    sum(:) = sum(:) + A(i,j) * X(j);
  end
  X(i) = (b(i)   sum(:))/A(i,i);
end

X = reshape(X,m,1);

end
```