MATH-6840
Vignesh Ramakrishnan
RIN: 662028006
Due: Monday February 7, 2022

Problem Set 3

NPDE is the textbook *Numerical Partial Differential Equations.* Submissions are due in the LMS, and must be typeset (e.g. LaTeX).

1. (10 pts.) Consider the smooth function $u(x)$ to be known at integer grid points $x_j = j\Delta x$ and use the notation $u_j = u(x_j)$

   (a) Is it possible to approximate $u_x(0)$ with error $\mathcal{O}(\Delta x^3)$ for general $u(x)$ using solution values u_j, for $j = -1, 0, 1$?

   This exercise is to figure out if $u_x(0)$ can be approximated with error $\mathcal{O}(\Delta x^3)$ using solution values u_j for $j = -1, 0, 1$.

$$
\begin{aligned}
\text{Let, } u_x(0) &= au(-\Delta x) + bu(0) + cu(\Delta x) + \text{Error} \\
u_x(0) &= a\left[u(0) - \Delta x u_x(0) + \frac{\Delta x^2}{2!} u_{xx}(0) - \frac{\Delta x^3}{3!} u_{xxxx}(0) + \ldots.\right] \\
&\quad + bu(0) \\
&\quad + c\left[u(0) + \Delta x u_x(0) + \frac{\Delta x^2}{2!} u_{xx}(0) + \frac{\Delta x^3}{3!} u_{xxxx}(0)\right] + \text{Error} \\
u_x(0) &= (a + b + c)\, u(0) + \Delta x\,(-a + c)\, u_x(0) \\
&\quad + \frac{\Delta x^2}{2!}(a + c)\, u_{xx}(0) + \frac{\Delta x^3}{3!}(-a + c)\, u_{xxxx}(0) + \ldots \ldots + \text{Error}
\end{aligned}
$$

   Matching the coefficients on the LHS with the coefficients on RHS results in

$$
\begin{aligned}
& a + b + c = 0 \\
& \Delta x\,(-a + c) = 1 \\
& \frac{\Delta x^2}{2!}(a + c) = 0 \\
& \text{Solving for, } a, b, \text{and, } c \text{ gives} \\
& a = \frac{-1}{2\Delta x} \\
& b = 0 \\
& c = \frac{1}{2\Delta x}
\end{aligned}
$$

   Substituting this back in the previous equation, we can find what Error term is,

$$
\begin{aligned}
u_x(0) &= 0 + u_x(0) + 0 + \frac{\Delta x^3}{3!}\frac{2}{2\Delta x} u_{xxxx}(0) + \frac{\Delta x^5}{5!}\frac{2}{2\Delta x} u^{(5)}(0) + \ldots \ldots + \text{Error} \\
\text{Error} &= -\Delta x^2\left[\frac{1}{2!} u_{xxxx}(0) + \frac{\Delta x^2}{5!} u^{(5)}(0) + \ldots + \ldots\right] = \mathcal{O}(\Delta x^2)
\end{aligned}
$$

   Even if all the three points are used, it is impossible to get an accuracy beyond $\mathcal{O}(\Delta x^2)$.

(b) Under what restrictions on $u(x)$ can one approximate $u_x(0)$ with error $\mathcal{O}(\Delta x^3)$ using solution values u_j, for $j = -1, 0, 1$?

If the smooth function $u(x)$ is a polynomial of order 2, like $u(x) = a_0 + a_1 x + a_2 x^2$, then $u_{xx}(0) = 2a_2$ and $u_{xxxx}(0) = 0$. In this case, the finite difference approximation developed should be of $\mathcal{O}(\Delta x^3)$ accuracy.

(c) Using the solution values u_j, for $j = -2, -1, 0, 1, 2$, derive as accurate an approximation to $u_x(0)$ as possible. What is the order of accuracy?

In order to get an approximation as accurate as possible, all points $j = -2, -1, 0, 1, 2$ should be used.

$$
\begin{aligned}
u_x(0) &= au(-2\Delta x) + bu(-\Delta x) + cu(0) + du(\Delta x) + eu(2\Delta x) + \text{Error} \\
&= a\left[u(0) - 2\Delta x u_x(0) + \frac{(2\Delta x)^2}{2!}u_{xx}(0) - \frac{(2\Delta x)^3}{3!}u_{xxxx}(0) + ...\right] \\
&\quad + b\left[u(0) - \Delta x u_x(0) + \frac{\Delta x^2}{2!}u_{xx}(0) - \frac{\Delta x^3}{3!}u_{xxxx}(0) + ...\right] \\
&\quad + cu(0) \\
&\quad + d\left[u(0) + \Delta x u_x(0) + \frac{\Delta x^2}{2!}u_{xx}(0) + \frac{\Delta x^3}{3!}u_{xxxx}(0) + ...\right] \\
&\quad + e\left[u(0) + 2\Delta x u_x(0) + \frac{(2\Delta x)^2}{2!}u_{xx}(0) + \frac{(2\Delta x)^3}{3!}u_{xxxx}(0) + ...\right] + \text{Error}
\end{aligned}
$$

$$
\begin{aligned}
u_x(0) = \; & (a+b+c+d+e)\,u(0) + \Delta x\,(-2a - b + d + 2e)\,u_x(0) + \\
& \frac{\Delta x^2}{2!}(4a + b + d + 4e)\,u_{xx}(0) + \frac{\Delta x^3}{3!}(-8a - b + d + 8e)\,u_{xxxx}(0) + \\
& \frac{\Delta x^4}{4!}(16a + b + d + 16e)\,u^{(4)}(0) + \dots \dots + \text{Error}
\end{aligned}
$$

Equating coefficients on LHS and RHS results in solving the set of linear equations,

$$
\begin{aligned}
& a + b + c + d + e = 0 \\
& \Delta x\,(-2a - b + d + 2e) = 1 \\
& 4a + b + d + 4e = 0 \\
& -8a - b + d + 8e = 0 \\
& 16a + b + d + 16e = 0
\end{aligned}
$$

Solving for the variables yields,

$$
a = \frac{1}{12\Delta x}, b = \frac{-2}{3\Delta x}, c = 0, d = \frac{2}{3\Delta x}, e = \frac{-1}{12\Delta x}
$$

$$
u_x(0) = \frac{u(-2\Delta x) - 8u(-\Delta x) + 8u(\Delta x) - u(2\Delta x)}{12\Delta x} + \text{Error}
$$

Expanding these terms further gives,

$$
\begin{aligned}
u_x(0) =& u_x(0) + a\left[\frac{(2\Delta x)^4}{4!}u^{(4)}(0) - \frac{(2\Delta x)^5}{5!}u^{(5)}(0) + ...\right] \\
&+ b\left[\frac{\Delta x^4}{4!}u^{(4)}(0) - \frac{\Delta x^5}{5!}u^{(5)}(0) + ...\right] \\
&+ d\left[\frac{\Delta x^4}{4!}u^{(4)}(0) + \frac{\Delta x^5}{5!}u^{(5)}(0) + ...\right] \\
&+ e\left[\frac{(2\Delta x)^4}{4!}u^{(4)}(0) + \frac{(2\Delta x^5)}{5!}u^{(5)}(0) + ...\right] + \text{Error} \\
0 =& (a+e)\left[\frac{(2\Delta x)^4}{4!}u^{(4)}(0) + \frac{(2\Delta x)^6}{6!}u^{(6)}(0) + ...\right] \\
&(b+d)\left[\frac{\Delta x^4}{4!}u^{(4)}(0) + \frac{\Delta x^6}{6!}u^{(6)}(0) + ...\right] \\
&(-a+e)\left[\frac{(2\Delta x)^5}{5!}u^{(5)}(0) + \frac{(2\Delta x)^7}{7!}u^{(7)}(0) + ...\right] \\
&(-b+d)\left[\frac{\Delta x^5}{5!}u^{(5)}(0) + \frac{\Delta x^7}{7!}u^{(7)}(0) + ...\right] + \text{Error}
\end{aligned}
$$

Here, $a+e = b+d = 0$ and substituting the variables give,

$$
\begin{aligned}
\text{Error} =& \frac{\Delta x^4}{6}\left[\frac{2^5}{5!}u^{(5)}(0) + \frac{2^7\Delta x^2}{7!}u^{(7)}(0) + ...\right] \\
&- \frac{4\Delta x^4}{3}\left[\frac{1}{5!}u^{(5)}(0) + \frac{\Delta x^2}{7!}u^{(7)}(0) + ...\right] = \mathcal{O}(\Delta x^4)
\end{aligned}
$$

(d) Using the solution values u_j, for $j = -2, -1, 0, 1, 2$, derive as accurate an approximation to $u_{xxx}(0)$ as possible. What is the order of accuracy?

Following the same procedure as above and matching coefficients will lead to the system of linear equations,

$$
\begin{aligned}
&a + b + c + d + e = 0 \\
&-2a - b + d + 2e = 0 \\
&4a + b + d + 4e = 0 \\
&-8a - b + d + 8e = 0 \\
&16a + b + d + 16e = 0
\end{aligned}
$$

Solving,

$$
a = \frac{-1}{2\Delta x^3}, b = \frac{1}{\Delta x^3}, c = 0, d = \frac{-1}{\Delta x^3}, e = \frac{1}{2\Delta x^3}
$$

$$
u_{xxxx}(0) = \frac{-u(-2\Delta x) + 2u(-\Delta x) - 2u(\Delta x) + u(2\Delta x)}{2\Delta x^3}
$$

Substituting these terms and expanding once again yields,

$$
\begin{aligned}
u_{xxxx}(0) = u_{xxxx}(0) + a &\left[-\frac{(2\Delta x)^5}{5!}u^{(5)}(0) + \frac{(2\Delta x)^6}{6!}u^{(6)}(0) + \dots \dots\right] \\
b &\left[-\frac{\Delta x^5}{5!}u^{(5)}(0) + \frac{\Delta x^6}{6!}u^{(6)}(0) + \dots \dots\right] \\
d &\left[\frac{\Delta x^5}{5!}u^{(5)}(0) + \frac{\Delta x^6}{6!}u^{(6)}(0) + \dots \dots\right] \\
e &\left[\frac{(2\Delta x)^5}{5!}u^{(5)}(0) + \frac{(2\Delta x)^6}{6!}u^{(6)}(0) + \dots \dots\right] + \text{Error} \\
0 = \frac{1}{\Delta x^3} &\left[\frac{(2\Delta x)^5}{5!}u^{(5)}(0) + \frac{(2\Delta x)^7}{7!}u^{(7)}(0) + \dots \dots\right] \\
- \frac{2}{\Delta x^3} &\left[\frac{\Delta x^5}{5!}u^{(5)}(0) + \frac{\Delta x^7}{7!}u^{(7)}(0) + \dots \dots\right] + \text{Error} \\
\text{Error} =& \mathcal{O}(\Delta x^2)
\end{aligned}
$$

2. (15 pts.) Again consider the smooth function $u(x)$ to be known at integer grid points $x_j = j\Delta x$ and continue to use the notation $u_j = u(x_j)$.

   (a) Derive an infinite expansion for the exact value of $u_{xx}(0)$ using the discrete operators $D_\pm$ and D_0 and assuming u_j is know at all relevant locations.

$$
\begin{aligned}
D_{+x}D_{-x}u(x_j) =& \frac{u_{j+1} - 2u_j + u_{j-1}}{\Delta x^2} \\
=& \frac{1}{\Delta x^2}\left[2\frac{\Delta x^2}{2!}u_{xx}(x_j) + 2\frac{\Delta x^4}{4!}u_{xxxx}(x_j) + \dots\dots\right] \\
=& u_{xx}(x_j) + \frac{2\Delta x^2}{4!}u_{xxxx}(x_j) + \frac{2\Delta x^4}{6!}u^6(x_j) + \dots. \\
u_{xx}(x_j) =& D_{+x}D_{-x}u_j - \frac{2\Delta x^2}{4!}u_{xxxx}(x_j) - \frac{2\Delta x^4}{6!}u^{(6)}(x_j) + \dots. \\
u_{xx}(x_j) =& D_{+x}D_{-x}u_j - \frac{2\Delta x^2}{4!}\left(D_{+x}D_{-x}\right)^2 u_j - \frac{2\Delta x^4}{6!}\left(D_{+x}D_{-x}\right)^3 u_j - \dots\dots. \\
u_{xx}(x_j) =& \left[\sum_{m=0}^{\infty} \alpha_m \Delta x^{2m}\left(D_{+x}D_{-x}\right)^{m+1}\right] u_j
\end{aligned}
$$

   (b) Using the representation in (a) above, derive a nonlinear equation whose solution gives the coefficients in the expansion in (a).

$$
\begin{aligned}
u &= \hat{u}e^{ikx} \\
u_x &= iku \\
u_{xx} &= -k^2 u \\
D_{+x}D_{-x}u_j &= \frac{-4}{\Delta x^2}\sin^2\left(\frac{\xi}{2}\right)u_j \\
-k^2 u_j &= \sum_{m=0}^{\infty}\alpha_m \Delta x^{2m}\frac{1}{\Delta x^2 \Delta x^2}\left(-4\sin^2\left(\frac{\xi}{2}\right)\right)^{m+1} u_j \\
-\xi^2 &= \sum_{m=0}^{\infty}\alpha_m\left(-4\sin^2\left(\frac{\xi}{2}\right)\right)^{m+1}
\end{aligned}
$$

$$
\xi^2 + \sum_{m=0}^{\infty}\alpha_m\left(-4\sin^2\left(\frac{\xi}{2}\right)\right)^{m+1} = 0
$$

(c) Using Taylor series, solve for the coefficients in your expansion and derive a 10th order accurate approximation to $u_{xx}(0)$. Present the discrete approximation. Note you are permitted to use symbolic software such as Maple or Mathematica.

I don't have Mathematica or Maple installed in my laptop, so I used online services (emathhelp.net) to compute the Taylor series expansions in order to derive a 10th order accurate approximation to $u_{xx}(0)$.
Let, $\xi = 2\eta$:

$$
\begin{aligned}
0 &= \xi^2 + \sum_{m=0}^{\infty}\alpha_m\left(-4\sin^2\left(\frac{\xi}{2}\right)\right)^{m+1} \\
&= 4\eta^2 + \sum_{m=0}^{\infty}\alpha_m\left(-4\sin^2\eta\right)^{m+1}
\end{aligned}
$$

$$
\begin{aligned}
0 = 4\eta^2 - 4\alpha_0\eta^2 &+ \frac{4\alpha_0}{3}\eta^4 - \frac{8\alpha_0}{45}\eta^6 + \frac{4\alpha_0}{215}\eta^8 - \frac{8\alpha_0}{14175}\eta^{10} + \dots \dots \\
&+ 16\alpha_1\eta^4 - \frac{32\alpha_1}{3}\eta^6 + \frac{16\alpha_1}{5}\eta^8 - \frac{544\alpha_1}{945}\eta^{10} + \dots \dots \\
&- 64\alpha_2\eta^6 + 64\alpha_2\eta^8 - \frac{448\alpha_2}{15}\eta^{10} + \dots \dots \\
&+ 256\alpha_3\eta^8 - \frac{1024\alpha_3}{3}\eta^{10} + \dots \dots \\
&- 1024\alpha_4\eta^{10} + \dots
\end{aligned}
$$

Gathering all coefficients and setting them to 0 results in:

$$
\begin{aligned}
0 = (4-4\alpha_0)\eta^2 &+ \left(\frac{4\alpha_0}{3}+16\alpha_1\right)\eta^4 - \left(\frac{8\alpha_0}{45}+\frac{32\alpha_1}{3}+64\alpha_2\right)\eta^6 \\
&+ \left(\frac{4\alpha_0}{215}+\frac{16\alpha_1}{5}+64\alpha_2+256\alpha_3\right)\eta^8 + \\
&- \left(\frac{8\alpha_0}{14175}+\frac{448\alpha_2}{15}+\frac{448\alpha_2}{15}+\frac{1024\alpha_3}{3}+1024\alpha_4\eta^{10}\right)\eta^{10} + \ldots \ldots
\end{aligned}
$$

Solving for the coefficients,

$$
\begin{aligned}
\alpha_0 &= 1 \\
\alpha_1 &= -0.0833 \\
\alpha_2 &= 0.0111 \\
\alpha_3 &= -0.0018 \\
\alpha_4 &= 3.1746 \times 10^{-4}
\end{aligned}
$$

Therefore,

$$
\begin{aligned}
u_{xx}(0) = \alpha_0 D_{+x}D_{-x}u_j &+ \alpha_1 \Delta x^2 \left(D_{+x}D_{-x}\right)^2 u_j + \\
&\alpha_2 \Delta x^4 \left(D_{+x}D_{-x}\right)^3 u_j + \alpha_3 \Delta x^6 \left(D_{+x}D_{-x}\right)^4 u_j + \\
&\alpha_4 \Delta x^8 \left(D_{+x}D_{-x}\right)^5 u_j + \mathcal{O}(\Delta x^{10})
\end{aligned}
$$

3. (10 pts.) *Adopted from NPDE exercise 1.5.12:*

   (a) Write a code to approximately solve

$$
\begin{aligned}
u_t &= \nu u_{xx}, && x \in (0,1), && t > 0 \\
u(x,0) &= \sin(2\pi x), && x \in (0,1) && \\
u(0,t) &= 0, && && t \geq 0 \\
u(1,t) &= 0, && && t \geq 0.
\end{aligned}
$$

   Use the grid $x_j = j\Delta x$, with $j = -1, 0, 1, \ldots, N, N+1$, and $\Delta x = 1/N$ (as described in the text), and apply the fourth-order centered spatial discretization with forward Euler time integration for $j = 1, 2, \ldots, N-1$, (BCs specified below) i.e.

$$
D_{+t}v_j^n = \nu D_{+x}D_{-x}\left(I - \frac{\Delta x^2}{12}D_{+x}D_{-x}\right)v_j^n.
$$

   The scheme given above can be decomposed into the following format

$$
\begin{aligned}
D_{+t}v_j^n &= \nu D_{+x}D_{-x}\left(Iv_j^n - \frac{\Delta x^2}{12}\frac{v_{j+1}^n - 2v_j^n + v_{j-1}^n}{\Delta x^2}\right) \\
&= \nu\left[\frac{v_{j+1}^n - 2v_j^n + v_{j-1}^n}{\Delta x^2} - \frac{1}{12}\left(D_{+x}D_{-x}v_{j+1}^n - 2D_{+x}D_{-x}v_j^n + D_{+x}D_{-x}v_{j-1}^n\right)\right] \\
&= \nu\left[\frac{v_{j+1}^n - 2v_j^n + v_{j-1}^n}{\Delta x^2} - \frac{1}{12}\left\{\frac{v_{j+2}^n - 4v_{j+1}^n + 6v_j^n - 4v_{j-1}^n + v_{j-2}^n}{\Delta x^2}\right\}\right] \\
v_j^{n+1} &= v_j^n + \frac{\nu\Delta t}{12\Delta x^2}\left(-v_{j+2}^n + 16v_{j+1}^n - 30v_j^n + 16v_{j-1}^n - v_{j-2}^n\right)
\end{aligned}
$$

A fourth order scheme for all the internal nodes ($j = 1, 2, 3, ..., N-1$)and a second order scheme for boundary nodes is applied and the MATLAB code of the same is attached below. For the boundary nodes a second order accurate central difference u_{xx} approximation is used. It is derived at the left and the right boundary as follows,

$$u_{xx}(0) = \frac{v_1^n - 2v_0^n + v_{-1}^n}{\Delta x^2} + \mathcal{O}(\Delta x^2)$$
$$u_{xx}(N) = \frac{v_{N+1}^n - 2v_N^n + v_{N-1}^n}{\Delta x^2} + \mathcal{O}(\Delta x^2)$$

As described above, ghost nodes (v_{-1}^n, v_{N+1}^n) are introduced and they are calculated using the compatibility boundary condition technique and that is described in part.b) of this question.

Listing 1: Heat Equation - Order 4 (Q.3b)

```
function [x,uhat,u_ex] = HeatEqn_Order4(N,dt,xlim1,xlim2,tlim1,tlim2)
% $Author: Vignesh Ramakrishnan$
% $RIN: 662028006$
% v_t = \nu v_xx, x \in (0,1), t > 0
% v(x,t=0) = sin(2\pi*x), v(0,t) = v(1,t) = 0
% \nu = 1/6;
% This function intends to use Finite Difference method to get a numerical
% solution to the heat equation described above. The compatability Boundary
% condtions are utilized to enforce Boundary conditions at the Boundary
% points. Euler forward stepping is utilized to march in time. A fourth
% order discretization scheme is used for approximating u_xx

% Inputs: N     - Number of elements - describes spatial discretization
%         dt    - temporal discretization
%         xlim1 - left end of the spatial boundary
%         xlim2 - right end of the spatial boundary
%         tlim1 - start time (initial condition time)
%         tlim2 - end time of simulation
% Output: Prints the time evolution across the spatial grid for the heat
%         equation described above

%% Trial runs

% HeatEqn_Order4(10,0.02,0,1,0,0.1); - stable solution
%% code

nu = 1/6;        % coefficient of heat transfer
a  = 0  ;        % time varying function at left boundary
b  = 0  ;        % time varying function at right boundary
dx = 1/N;        % dx -spatial discretization

r  = nu*dt/dx^2;% r = CFL number

```

```
ng = 1   ;         % number of ghost points at one boundary
NP = N+1+2*ng;  % Total number of spatial points
ja = ng+1;        % xlim1's index number
jb = NP-ng;       % xlim2's index number

x   = (xlim1:dx:xlim2);
t   = (tlim1:dt:tlim2);

u_prev   = zeros(1,NP);
u_curr   = zeros(1,NP);

u_if      = @(xv,tv) sin(2*pi*xv)* exp(-nu*4*pi^2*tv);

% set initial conditions for the spatial grid
u_prev(ja:jb)      = sin(2*pi*x);
u_prev(ja)         = a;
u_prev(jb)         = b;
u_prev(1)          = 2*u_prev(ja) - u_prev(ja+1);
u_prev(NP)         = 2*u_prev(1,jb) - u_prev(jb-1);

for i=2:length(t)
    for j = ja:jb
        if (j==ja || j==jb)
%               u_curr(j) = u_prev(j) + ...
%                   r*(u_prev(j+1)-2*u_prev(j)+ u_prev(j-1));
            u_curr(j) = 0;
        else
            u_curr(j) = u_prev(j) + ...
                (r/12)*(-u_prev(j+2) + 16*u_prev(j+1) -30*u_prev(j)...
                + 16*u_prev(j-1) - u_prev(j-2));
        end
    end

    u_curr(ng) = 2*u_curr(ja) - u_curr(ja+1);
    u_curr(NP) = 2*u_curr(jb) - u_curr(jb-1);
    u_prev         = u_curr;

end

u_ex      = u_if(x,tlim2);
uhat      = u_prev(ja:jb);

end
```

(b) Set $\nu = 1/6$, $\Delta t = 0.02$, and $N = 10$. Define ghost values using

$$\begin{aligned} v_{-1}^n &= 2v_0^n - v_1^n \\ v_0^n &= 0 \\ v_N^n &= 0 \\ v_{N+1}^n &= 2v_N^n - v_{N-1}^n, \end{aligned}$$

and compute approximate solutions at $t = 0.06$, $t = 0.1$, and $t = 0.9$.

The approximate solutions generated using this scheme are presented in Fig 1

(c) Again set $\nu = 1/6$, $\Delta t = 0.02$, and $N = 10$. Now define ghost values using

$$\begin{aligned} v_{-1}^n &= 0 \\ v_{N+1}^n &= 0, \end{aligned}$$

and compute approximate solutions at $t = 0.06$, $t = 0.1$, and $t = 0.9$.

If the ghost nodes are assigned the values of 0 each, boundary conditions are not satisfied with this approach. It can be seen in Fig 2.

(d) Discuss your results in comparison to each other and to those from PS2 #1.
From Fig 1., and Fig 2., it does seem like the second order solution works better in terms of approximating or arriving at a numerical solution. One reason I can think of is that, by using only 2 ghost nodes and also developing a second order scheme to find the values at the ghost nodes, the overall fourth order accuracy is not met, especially at the boundaries. Maybe, adding 2 ghost nodes on either sides of the boundaries and using the developed fourth order scheme to generate the values at the boundaries might help in inching towards overall fourth order accuracy. This is my takeaway. But, it could also be that something was wrong in the way I coded this problem. But I have checked it multiple times already and I don't know if I'm missing an error that's obvious.
Now, if the two fourth order schemes are compared, the scheme in which the ghost nodes are calculated using a second order accurate compatibility boundary condition (3.b), works better than the scheme where the ghost nodes are given a constant 0 value at all times. This increases my confidence in my initial guess that, if a higher order accuracy is maintained at the boundaries, the overall performance of the fourth order scheme should be better than the second order scheme.

4. (15 pts.) Consider the heat equation

$$u_t - u_{xx} = f(x,t), \qquad 0 < x < 1, \qquad t > 0$$

with initial conditions $u(x, t = 0) = u_0(x)$ and boundary conditions of the form

$$\begin{aligned} u(x = 0, t) &= \gamma_L(t) \\ u_x(x = 1, t) &= \gamma_R(t). \end{aligned}$$

(a) Determine $f(x,t)$, $u_0(x)$, $\gamma_L(t)$, and $\gamma_R(t)$ so that the exact solution to the problem is $u(x,t) = 2\cos(x)\cos(t)$.

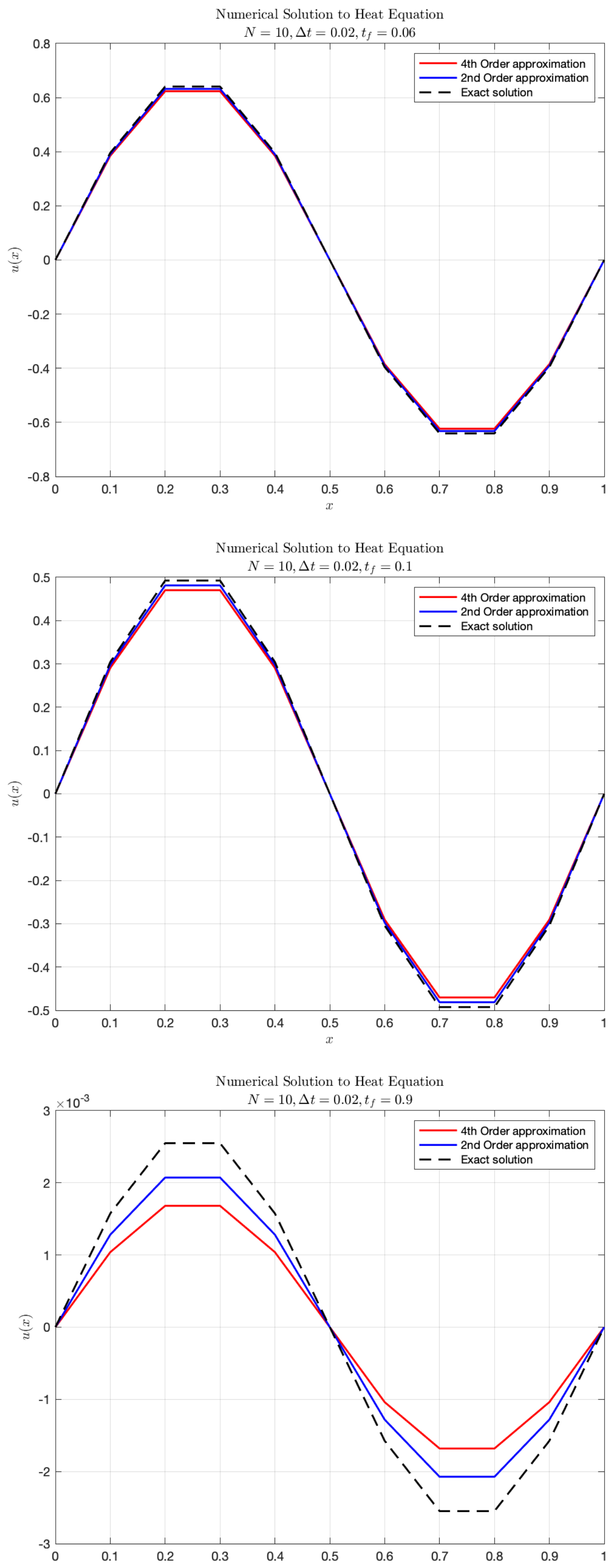

Figure 1: Fourth Order Solution

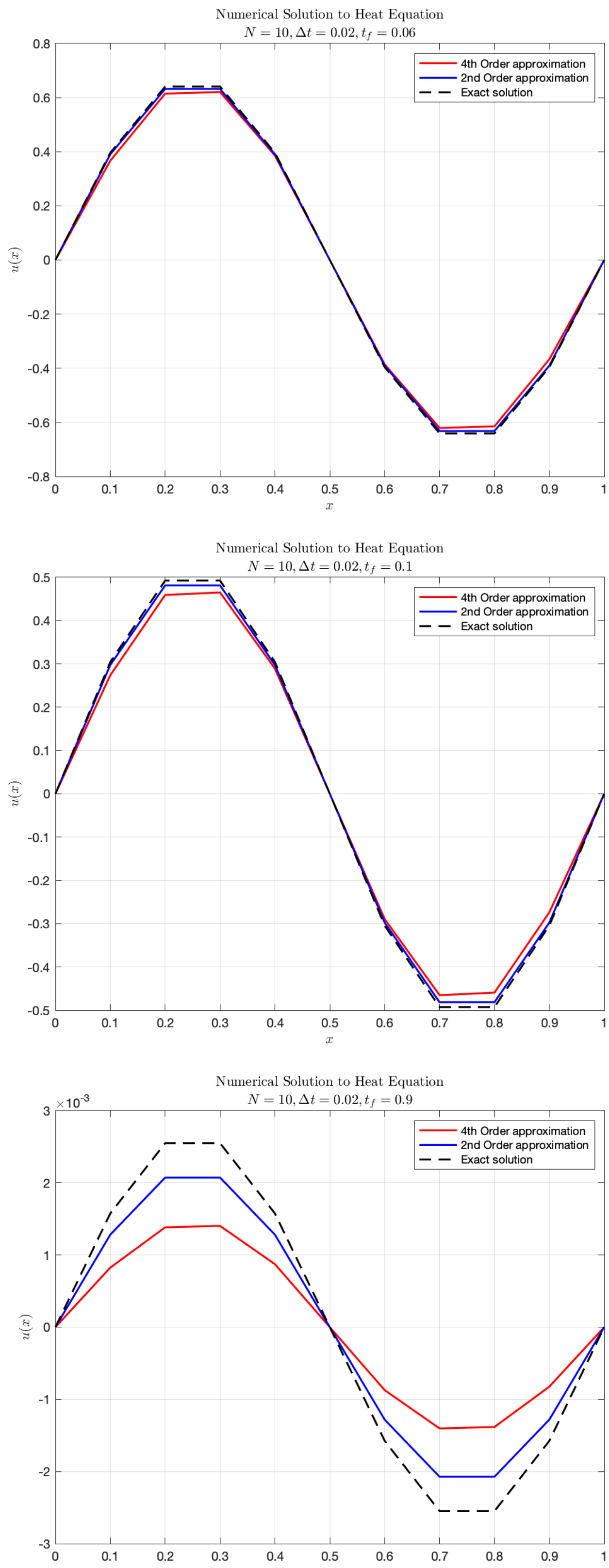

Figure 2: Solution with 0 values at ghost points

This technique can be used to verify that the numerical solution is indeed trying to approach closer and closer to the exact analytical solution and all the tricks of adding higher order of accuracy, etc is not just to blindly ensure stability and convergence. This technique does prove that the Finite Difference approach tries to converge to the analytical/exact solution.

$$
\begin{aligned}
u(x,y) &= 2\cos(x)\cos(t) \\
u(x,t=0) &= 2\cos(x) = u_0(x) \\
u(x=0,t) &= 2\cos(t) = \gamma_L(t) \\
u_x(x,t) &= -2\sin(x)\cos(t) \\
u_x(x=1,t) &= -2\sin(1)\cos(t) = \gamma_R(t)
\end{aligned}
$$

Now,

$$
\begin{aligned}
u_{xx} &= -2\cos(x)\cos(t) \\
u_t &= -2\cos(x)\sin(t)
\end{aligned}
$$

Substituting it into the PDE,

$$
u_t - u_{xx} = 2\cos(x)\left(\cos(t) - \sin(t)\right) = f(x,t)
$$

(b) Write a code to solve this problem using the scheme

$$
D_{+t}v_j^n = \nu D_{+x}D_{-x}v_j^n + f_j^n
$$

on the grid defined by $x_j = j\Delta x$, $j = 0, 1, \ldots, N$, $\Delta x = 1/N$, with the parameter $r = \Delta t/\Delta x^2$. You can include ghost points as you need them, but you must ensure that your boundary conditions are at least second-order accurate.

To maintain second order accuracy at all nodes, two ghost nodes are added at each of the boundaries. Compatibility boundary condition at the left boundary and the central difference first derivative approximate D_{0x} at the right boundary will ensure second order accuracy at all node points. The code has been attached in the listing below. The scheme is developed as shown,

$$
v_j^{n+1} = v_j^n + \frac{\Delta t}{\Delta x^2}\left(v_{j+1}^n - 2v_j^n + v_{j-1}^n\right) + \Delta t f_n^j
$$

Listing 2: Heat Equation -Forced (Q.4)

```
function [err_norm,x,uhat,u_ex] = HeatEqn_Forcing(N,r,xlim1,xlim2,...
                                                  tlim1,tlim2)
% $Author: Vignesh Ramakrishnan$
% $RIN: 662028006$
% u_t - \nu u_{xx} = f(x,t)
% s.t u(x,0) = u_0(x)
% u(0,t)     = \Gamma_L(t)
% u_x(1,t)   = \Gamma_R(t)
% u_{ex} = 2\cos(x)\cos(t)
```

```
% This function is a method to prove that the Difference methods work and
% will be a good approximate to the exact solution. The task is to find
% functions f(x,t), u_0(x), Gamma_L(t) and Gamma_R(t), plug it in
% and solve using the scheme: D+t v^n_j = \nu D+xD-xv^n_j + f^n_j
% Inputs: N        - Number of elements
%         r        - CFL number
%         xlim1    - left end of the spatial boundary
%         xlim2    - right end of the spatial boundary
%         tlim1    - start time of simulation
%         tmin2    - end time of simulation
% Output: err_norm - L2 norm of the error between the exact solution and
%                    numerical solution

nu  = 1;                                % Co-efficient of heat conduction

u0  = @(x) 2*cos(x);                    % Initial condition
gL  = @(t) 2*cos(t);                    % Drichlet BC on Left Boundary
gR  = @(t) -2*sin(1)*cos(t);            % Neumann BC on Right Boundary
u0t = @(t) -2*sin(t);                   % u_t @ x=0
f   = @(x,t) 2*cos(x)*(cos(t)-sin(t)); % Forcing function f

dx  = (xlim2-xlim1)/N;                  % dx -spatial discretization

dt    = r*dx^2;                         % r = CFL number

ng  = 1   ;                             % number of ghost points at BC
NP  = N+1+2*ng;                         % Total number of spatial points
ja  = ng+1;                             % xlim1's index number
jb  = NP-ng;                            % xlim2's index number

x   = (xlim1:dx:xlim2);                 % Spatial locations  x
t   = (tlim1:dt:tlim2);                 % Temporal locations t

u_prev  = zeros(1,NP);                  % Solution at previous tstep
u_curr  = zeros(1,NP);                  % Solution at current tstep

% set initial conditions for the spatial grid
u_prev(ja:jb)     = u0(x);
u_prev(ja)        = gL(tlim1);

% Set Compatability boundary condition
u_prev(ng)        = (u0t(tlim1) - f(xlim1,tlim1))*dx^2 ...
                         + 2*u_prev(ja) - u_prev(ja+1);
% Set Neumann boundary condition
u_prev(NP)        = u_prev(jb-1) + 2*gR(tlim1)*dx;

for i=2:length(t)
    for j = ja:jb
```

```
            u_curr(j) = u_prev(j) + ...
                        r*(u_prev(j+1)-2*u_prev(j)+ u_prev(j-1)) + ...
                          dt*f(x(j-1),t(i));
        end

        u_curr(ng) = 2*u_curr(ja) - u_curr(ja+1) + ...
                     (u0t(t(i)) - f(xlim1,t(i)))*dx^2;
        u_curr(NP) = u_curr(jb-1) + 2*gR(t(i))*dx;
        u_prev     = u_curr;

end

u_ex = 2*cos(x)*cos(tlim2);
uhat = u_prev(ja:jb);
err      = u_ex - u_prev(ja:jb);
err_norm = norm(err);

end
```

The solution at $N = 640, CFL = 0.4$, is shown in Fig 3. The error between the numerical

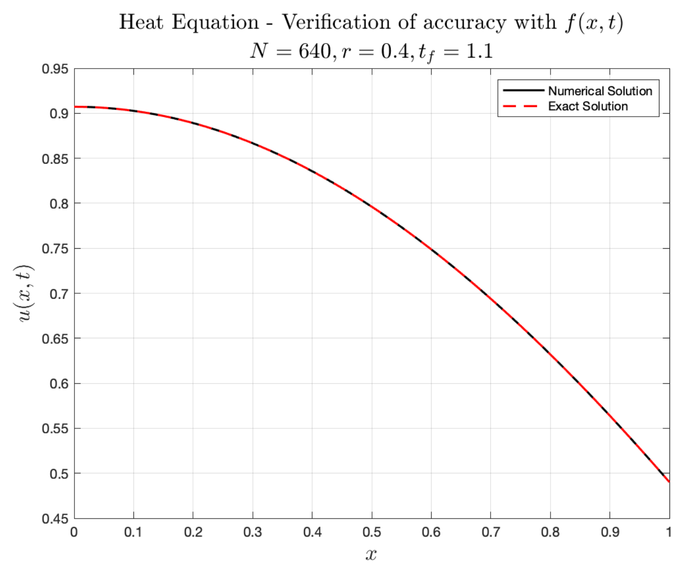

Figure 3: Heat Equation under forcing

solution and the exact solution is $= \ 4.9605 \times 10^{-5}$

(c) Perform a grid refinement study using $N = 20, 40, 80, 160, 320, 640$ by computing the maximum errors in the approximation at $t = 1.1$.Discuss the observed order of accuracy of the method.

Grid refinement study was performed and a $\log - \log$ plot is made. If the order of accuracy is $\mathcal{O}(\Delta x^p)$, then,

$$\text{error} = k\Delta x^p$$
$$\log(\text{error}) = p\log(\Delta x) + \log(k)$$

The slope of this line is p and it is an indication of the order of accuracy. From Fig 4, the observed order of accuracy is approximately 1.5.

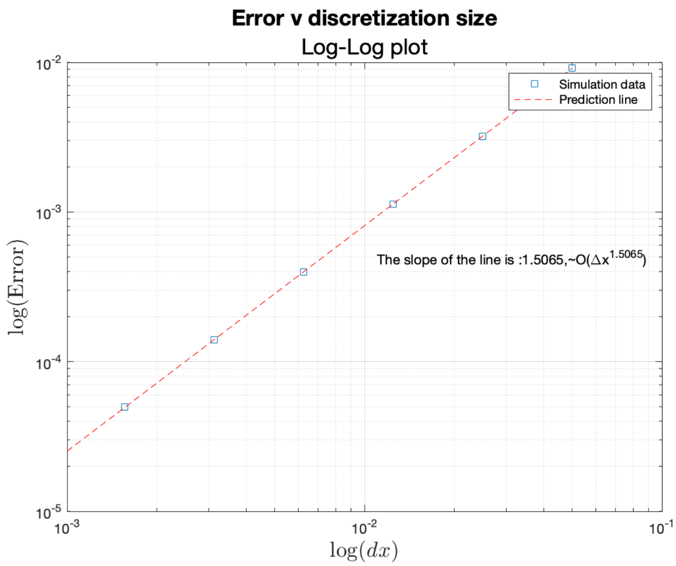

Figure 4: Log-Log plot of Error v spatial discretization