Name :- Kunal Ghosh

Course :- M. Tech. (Course)

Subject :- Computational Heat Transfer and Fluid Flow (ME 282)

Assignment No. :- 3

S.A.P. No. :- 6000007645

S.R. No. :- 05-01-00-10-42-22-1-21061

Problem (1):

Consider the leap frog scheme for numerical integration of the ODE

$$\frac{dy}{dt} = f(y,t)$$

$$y(0) = y_0$$

Analyze the stability of this scheme and express it in the form of a stability diagram.

Now consider the application of leap frog method with central in space discretization of

$$u_t = b\, u_{xx}$$

$$u(0,t) = u(1,t) = 0$$

$$u(x,0) = u_0(x)$$

constructed using uniform grid points in $[0,1]$ and the standard three-point second-order discretization to approximate u_{xx} (let $x_i = \frac{i}{N}$, $i = 0, 1, 2, \ldots, N$). Derive a stability criterion for this method using the stability diagram obtained from the first part.

Solution:

$$\frac{dy}{dt} = f(y,t)$$

$$y_0 = y(0)$$

Let's assume there is some fixed point y_* and we perturb it by amount $\bar{y}$.

So, $y = y_* + \bar{y}$

So, $\frac{dy}{dt} = \frac{d}{dt}(y_* + \bar{y})$

$$\frac{dy}{dt} = \frac{d\bar{y}}{dt}$$

As $\frac{dy}{dt} = f(y, t)$

~~So, $\frac{d\bar{y}}{dt}$~~

So, $\frac{d\bar{y}}{dt} = f(y_* + \bar{y}, t)$

$$= f(y_*, t) + \bar{y} f_y(y_*, t) + H.O.T.$$

(Taylor Series)

So, $\frac{d\bar{y}}{dt} = \bar{y} f_y(y_*, t)$

[As $f(y_*, t) = 0$ (fixed Point)

Neglecting the Higher Order Terms (H.O.T.).]

So, $\frac{d\bar{y}}{dt} = \bar{y} f_y(y_*, t)$

If we <u>consider y as a vector</u> then,

$[J] = f_y(y_*, t) =$ Jacobian.

So, $\bar{y}_t = [J]\bar{y}$

Diagonalizing $[J]$ as $[J] = P \Lambda P^{-1}$

So, $\bar{y}_t = (P \Lambda P^{-1})\bar{y}$

or $P^{-1}\bar{y}_t = \Lambda P^{-1}\bar{y}$

~~Let $\tilde{y} = \Lambda P^{-1}$~~

Let $\tilde{y} = P^{-1}\bar{y}$

So, $\tilde{y}_t = P^{-1}\bar{y}_t$

So, $\boxed{\tilde{y}_t = \Lambda \tilde{y}}$

Using leap frog scheme, we can analyze stability as

$$\frac{\tilde{y}^{n+1} - \tilde{y}^{n-1}}{2\Delta t} = \lambda \tilde{y}^n$$

Let Amplification factor $= K = \dfrac{\tilde{y}^{n+1}}{\tilde{y}^{n}} = \dfrac{\tilde{y}^{n}}{\tilde{y}^{n-1}}$

So, $\dfrac{\tilde{y}^{n+1} - \tilde{y}^{n-1}}{\tilde{y}^{n}} = (\lambda \Delta t)(2)$

Let $z = (\Delta t)\lambda$

So, $K + \left(-\frac{1}{K}\right) = (z)(2)$

or $\dfrac{K^2 - 1}{K} = 2z$

or $K^2 - 2zK - 1 = 0$

$$K = \frac{(2z) \pm \sqrt{(-2z)^2 - 4(1)(-1)}}{2(1)}$$

or $\underline{\underline{K = z \pm \sqrt{z^2 + 1}}}$

For $z \in \mathbb{R}$, we have

$\underline{\underline{|K| \geq 1}}$

Even ~~from~~ for complex numbers $|K| \geq 1$
(Using MATLAB)

But for stability we need $|K| \leq 1$

This, only possible if $|K| = 1$

This happen when z lies on the imaginary axis between $-i$ & i.

If z is on imaginary axis $\in [-i, i]$

Let $z = Pi$ $\quad (|P| \leq 1)$

So, $\pm\sqrt{z^2+1} = \pm\sqrt{1-P^2}$

This the real part of $z \pm \sqrt{z^2+1}$

So, Pi is the imaginary part of $z \pm \sqrt{z^2+1}$

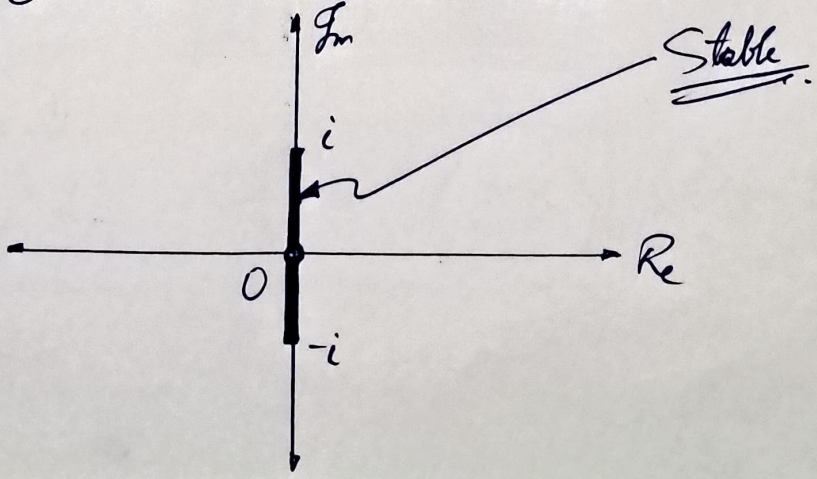
So, $\mathrm{Im}\left(z \pm \sqrt{z^2+1}\right) = P$

So, $|z|^2 = \text{Real}^2 + \text{Imaginary}^2$

$= 1 - P^2 + P = 1$

Also, <u>NOTE</u>, $z = 0 \rightarrow$ Repeated roots

So, $z = 0$ is NOT stable.



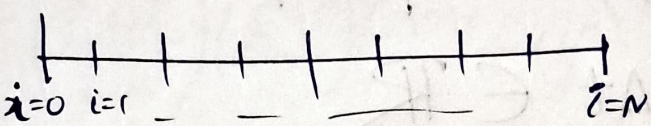
Consider the following equation

$$u_t = b\, u_{xx},$$

$$u(0,t) = u(1,t) = 0$$

$$u(x,0) = u_0(x)$$

<u>Discretization</u> (Uniform grid):



$$\frac{du_i}{dt} = b\left[\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2}\right]$$

So, $$\frac{d\bar{u}}{dt} = \frac{b}{h^2}\begin{bmatrix} -2 & 1 & 0 & \cdots & 0 \\ 1 & -2 & 1 & \cdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & 1 & -2 & 1 \\ 0 & \cdots & 0 & 1 & -2 \end{bmatrix}\bar{u}$$

So, $$\frac{d\bar{u}}{dt} = [J]\,\bar{u}$$

So, $$\boxed{\lambda_J = -\frac{4b}{h^2}\sin^2\left(\frac{\pi K}{2N}\right)}$$ (Eigen Values)

$K = 1, 2, \ldots, (N-1)$

From the stability diagram, ~~of~~ leap frog scheme, ~~$|R| \leq 1$~~ is stable if $|R| \leq 1$

So, $\left| \lambda_J \Delta t \pm \sqrt{(\lambda_J \Delta t)^2 + 1} \right| \leq 1$

So, $\left| z \pm \sqrt{z^2 + 1} \right| \leq 1$

But $z = \lambda_J \Delta t \in \mathbb{R}$

So, z lies on Real axis which is NOT Stable

So, CTCS schemes for $u_t = b\, u_{xx}$ is unconditionally unstable

Problem ②

Determine the order of accuracy and discuss the stability of the following difference scheme

$$\left(1+\frac{1}{12}\delta_x^{(2)}\right)\left(u_j^{n+1}-u_j^n\right)$$

$$=\left(\frac{b\,\Delta t}{2h^2}\right)\left(\delta_x^{(2)}\right)\left(u_j^{n+1}+u_j^n\right)$$

$$+\frac{\Delta t}{2}\left[f_j^{n+1}+\left(1+\frac{1}{6}\delta_x^{(2)}\right)f_j^n\right]$$

for the solution of $u_t = b\,u_{xx} + f$,

where $\delta_x^{(2)} u_j^n = u_{j+1}^n - 2u_j^n + u_{j-1}^n$

Solution :

$$\left(1+\frac{1}{12}\delta_x^{(2)}\right)\left(u_j^{n+1}-u_j^n\right)$$

$$=\left(\frac{b\,\Delta t}{2h^2}\right)\left(\delta_x^{(2)}\right)\left(u_j^{n+1}+u_j^n\right)$$

$$+\left(\frac{\Delta t}{2}\right)\left[f_j^{n+1}+\left(1+\frac{1}{6}\delta_x^{(2)}\right)f_j^n\right]$$

Let Term ① $=\left(1+\frac{1}{12}\delta_x^{(2)}\right)\left(u_j^{n+1}-u_j^n\right)$

$$\text{Term } ① = u_j^{n+1} - u_j^n + \frac{1}{12}\left(u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}\right)$$
$$- \frac{1}{12}\left(u_{j+1}^n - 2u_j^n + u_{j-1}^n\right)$$

$$\underline{\underline{\text{Term } ①}} = u_j^{n+1} - u_j^n + \frac{1}{12}\left(u_{j+1}^{n+1} - u_{j+1}^n\right)$$
$$- \frac{2}{12}\left(u_j^{n+1} - u_j^n\right) + \frac{1}{12}\left(u_{j-1}^{n+1} - u_{j-1}^n\right)$$

Let, $\underline{\underline{\text{Term } ②}} = \left(\frac{b\,\Delta t}{2h^2}\right) \delta_x^{(2)} \left(u_j^{n+1} + u_j^n\right)$

$$\underline{\underline{\text{Term } ②}} = \left(\frac{b\,\Delta t}{2h^2}\right)\left[\left(u_{j+1}^{n+1} + u_{j+1}^n\right)\right]$$
$$+ \left(\frac{b\,\Delta t}{2h^2}\right)\left[-2\left(u_j^{n+1} + u_j^n\right)\right]$$
$$+ \left(\frac{b\,\Delta t}{2h^2}\right)\left[\left(u_{j-1}^{n+1} + u_{j-1}^n\right)\right]$$

Let $\underline{\underline{\text{Term } ③}} = \frac{\Delta t}{2}\left[f_i^{n+1} + \left(1 + \frac{1}{6}\delta_x^{(2)}\right) f_j^n\right]$

$$\underline{\underline{\text{Term } ③}} = \frac{\Delta t}{2}\left[f_j^{n+1} + f_j^n + \frac{1}{6}\left(f_{j+1}^n - 2f_j^n + f_{j-1}^n\right)\right]$$

Stability Analysis:-

For stability analysis we can neglect the forcing term (Term ③).

(As stability is the inherent property of the system)

Let $u_j^n \sim g^n e^{ij\theta}$ (Von-Neumann Stability Method)

$$\text{Term ①} = u_j^{n+1} - u_j^n + \frac{1}{12}\left(u_{j+1}^{n+1} - u_{j+1}^n\right)$$

$$-\frac{2}{12}\left(u_j^{n+1} - u_j^n\right) + \frac{1}{12}\left(u_{j-1}^{n+1} - u_{j-1}^n\right)$$

$$= g^{n+1} e^{ij\theta} - g^n e^{ij\theta} + \frac{1}{12}\left(g^{n+1} e^{i(j+1)\theta} - g^n e^{i(j+1)\theta}\right)$$

$$-\frac{2}{12}\left(g^{n+1} e^{ij\theta} - g^n e^{ij\theta}\right)$$

$$+\frac{1}{12}\left(g^{n+1} e^{i(j-1)\theta} - g^n e^{i(j-1)\theta}\right)$$

$$= g^n e^{ij\theta}\left[(g-1) + \frac{1}{12}\left(g e^{i\theta} - e^{i\theta}\right) - \frac{2}{12}(g-1)\right]$$

$$+\frac{g^n e^{ij\theta}}{12}\left[g e^{-i\theta} - e^{-i\theta}\right]$$

$$\text{Term ①} = g^n e^{ij\theta}\left[(g-1)\left(1-\frac{2}{12}\right)\right]$$

$$+ g^n e^{ij\theta}\left[\frac{1}{12}\left(g e^{i\theta} - e^{i\theta}\right) + \frac{1}{12}\left(g e^{-i\theta} - e^{-i\theta}\right)\right]$$

$$= g^n e^{ij\theta}(g-1)\left[1 - \frac{2}{12} + \left(\frac{e^{i\theta} + e^{-i\theta}}{12}\right)\right]$$

$$= g^n e^{ij\theta}(g-1)\left[1 - \frac{2}{12} + \frac{\cos\theta}{6}\right]$$

$$\text{Term ①} = g^n e^{ij\theta}(g-1)\left[\frac{10}{12} + \frac{\cos\theta}{6}\right]$$

$$\text{Term ②} = \left(\frac{6\Delta t}{2h^2}\right)\left[\left(u_{j+1}^{n+1} + u_{j+1}^{n}\right) - 2\left(u_j^{n+1} + u_j^{n}\right) + \left(u_{j-1}^{n+1} + u_{j-1}^{n}\right)\right]$$

$$\text{Term ②} = \left(\frac{6\Delta t}{2h^2}\right)\left[g^{n+1} e^{i(j+1)\theta} + g^n e^{i(j+1)\theta}\right]$$

$$+ \left(\frac{6\Delta t}{2h^2}\right)\left[-2\left(g^{n+1} e^{ij\theta} + g^n e^{ij\theta}\right)\right]$$

$$+ \left(\frac{6\Delta t}{2h^2}\right)\left[g^{n+1} e^{i(j-1)\theta} + g^n e^{i(j-1)\theta}\right]$$

$$\text{Term ②} = \left(\frac{b\,\Delta t}{2h^2}\right) g^n e^{ij\theta}\left[g e^{i\theta} + e^{i\theta} - 2(g+1) + \left(g e^{-i\theta} + e^{-i\theta}\right)\right]$$

$$= \left(\frac{b\,\Delta t}{2h^2}\right) g^n e^{ij\theta}\left[g\left(e^{i\theta} + e^{-i\theta}\right) - 2(g+1) + \left(e^{i\theta} + e^{-i\theta}\right)\right]$$

$$= \left(\frac{b\,\Delta t}{2h^2}\right) g^n e^{ij\theta}\left[g(2\cos\theta) - 2(g+1) + (2\cos\theta)\right]$$

$$= \left(\frac{b\,\Delta t}{h^2}\right) g^n e^{ij\theta}\left[(g+1)(\cos\theta - 1)\right]$$

$$\text{Term ②} = \left(\frac{b\,\Delta t}{h^2}\right) g^n e^{ij\theta}(g+1)(\cos\theta - 1)$$

Equating Term ① & Term ②.
(Term ③, <u>forcing term is neglected</u>)

$$g^n e^{ij\theta}(g-1)\left[\frac{10}{12} + \frac{\cos\theta}{6}\right]$$

$$= \left(\frac{b\,\Delta t}{h^2}\right) g^n e^{ij\theta}(g+1)(\cos\theta - 1)$$

or $(g-1)\left(\frac{10}{12}+\frac{\cos\theta}{6}\right)$

$= \left(\frac{6\,\Delta t}{h^2}\right)(g+1)(\cos\theta-1)$

or $g\left(\frac{10}{12}+\frac{\cos\theta}{6}\right)-\left(\frac{10}{12}+\frac{\cos\theta}{6}\right)$

$= g\left(\frac{6\Delta t}{h^2}\right)(\cos\theta-1)+\left(\frac{6\Delta t}{h^2}\right)(\cos\theta-1)$

Let $\mu = \frac{6\Delta t}{h^2}$

So, $g\left(\frac{10}{12}+\frac{\cos\theta}{6}\right)-\left(\frac{10}{12}+\frac{\cos\theta}{6}\right)$

$= g\mu(\cos\theta-1)+\mu(\cos\theta-1)$

or $g\left(\frac{10}{12}+\frac{\cos\theta}{6}+\mu(1-\cos\theta)\right)$

$= \mu(\cos\theta-1)+\left(\frac{10}{12}+\frac{\cos\theta}{6}\right)$

or $g = \dfrac{\left(\mu(\cos\theta-1)+\left(\frac{10}{12}+\frac{\cos\theta}{6}\right)\right)}{\left(\frac{10}{12}+\frac{\cos\theta}{6}+\mu(1-\cos\theta)\right)}$

So, $g = \dfrac{12\mu(\cos\theta - 1) + 10 + 2\cos\theta}{10 + 2\cos\theta + 12\mu(1 - \cos\theta)}$

or $g = \dfrac{6\mu(\cos\theta - 1) + (5 + \cos\theta)}{(5 + \cos\theta) + 6\mu(1 - \cos\theta)}$

As $\cos\theta \leq 1$

So, $(1 - \cos\theta) \geq 0$

So, $6\mu(1 - \cos\theta) \geq 0$

So, Let $m = 6\mu(1 - \cos\theta) \geq 0$

$K = 5 + \cos\theta > 0$

So, $g = \left(\dfrac{K - m}{K + m}\right)$

So, $|g| \leq 1$

So, the scheme is <u>Unconditionally Stable</u>

Order of Accuracy:

$$u_{j+1}^{n+1} = u + h\,u_x + (\Delta t)\,u_t + \frac{h^2}{2!}u_{xx} + \frac{\Delta t^2}{2!}u_{tt} + \frac{h\,\Delta t}{1!\,1!}u_{xt} + H.O.T.$$

$$u_{j+1}^{n} = u + h\,u_x + \frac{h^2}{2!}u_{xx} + \frac{h^3}{3!}u_{xxx} + HOT$$

$$u_{j}^{n+1} = u + (\Delta t)\,u_t + \frac{(\Delta t)^2}{2!}u_{tt} + \frac{(\Delta t)^3}{3!}u_{ttt} + HOT$$

$$u_{j}^{n} = u$$

$$u_{j-1}^{n+1} = u - h\,u_x + (\Delta t)\,u_t + \frac{h^2}{2!}u_{xx} + \frac{\Delta t^2}{2!}u_{tt} - \frac{h\,\Delta t}{1!\,1!}u_{xt} + H.O.T.$$

$$u_{j-1}^{n} = u - h\,u_x + \frac{h^2}{2!}u_{xx} - \frac{h^3}{3!}u_{xxx} + H.O.T.$$

Consider,

$$u_j^{n+1} - u_j^{n} = (\Delta t)\,u_t + \frac{(\Delta t)^2}{2!}u_{tt} + \frac{\Delta t^3}{3!}u_{ttt} + H.O.T.$$

$$u_{j+1}^{n+1} - u_j^{n+1} = h\,u_x + \frac{h^2 u_{xx}}{2!} + \frac{h\,\Delta t}{1!\,1!}u_{xt} + H.O.T.$$

$$u_{j+1}^{n+1} - u_{j+1}^{n} = (\Delta t)\,u_t + \frac{(\Delta t)^2}{2!}u_{tt} + \frac{h\,\Delta t}{1!\,1!}u_{xt} + H.O.T$$

$$u_{j-1}^{n+1} - u_{j-1}^{n} = (\Delta t) u_t + \frac{(\Delta t)^2}{2!} u_{tt} - h \Delta t\, u_{xt} + H.O.T.$$

So, Term ① $= (u_j^{n+1} - u_j^n) + \frac{1}{12}\left(u_{j+1}^{n+1} - u_{j+1}^n\right)$

$$- \frac{2}{12}\left(u_j^{n+1} - u_j^n\right) + \frac{1}{12}\cdot\left(u_{j-1}^{n+1} - u_{j-1}^n\right)$$

$$= \left((\Delta t)(u_t) + \frac{\Delta t^2}{2!} u_{tt} + \cdots\right)$$

$$+ \frac{1}{12}\left((\Delta t)(u_t) + \frac{(\Delta t)^2}{2!} u_{tt} + h\Delta t\, u_{xt} + \cdots\right)$$

$$- \frac{2}{12}\left((\Delta t) u_t + \frac{(\Delta t)^2}{2!} u_{tt} + \cdots\right)$$

$$+ \frac{1}{12}\left((\Delta t) u_t + \frac{(\Delta t)^2}{2!} u_{tt} + (-h\Delta t)\, u_{xt} + \cdots\right)$$

Term ① $= (\Delta t)(u_t) + \frac{(\Delta t)^2}{2!} u_{tt} + \frac{2}{12} u_{xxt} \frac{(\Delta t) h^2}{2!\,!!}$

$+ H.OT.$

Term ② $= \frac{\mu}{2}\left(u_{j+1}^{n+1} + u_{j+1}^n\right) - \mu\left(u_j^{n+1} + u_j^n\right)$

$$+ \frac{\mu}{2}\left(u_{j-1}^{n+1} + u_{j-1}^{n+1}\right)$$

$$= \frac{\mu}{2}\left(2\left(u + h u_x + \frac{h^2}{2!} u_{xx} + \cdots\right) + \left(\Delta t\, u_t + \frac{\Delta t^2}{2!} u_{tt} + \cdots\right)\right)$$

$$- \mu\left[2\left(u + \Delta t\, u_t + \frac{\Delta t^2}{2!} u_{tt} + \cdots\right)\right]$$

$$+ \frac{\mu}{2}\left[2\left(u - h u_x + \frac{h^2}{2!} u_{xx} + \cdots\right) + \left(\Delta t\, u_t + \frac{\Delta t^2}{2} u_{tt} + \cdots\right)\right]$$

Term ③ $= \frac{\Delta t}{2}\left[f_j^{n+1} + (1+\delta_x^{(2)}) f_j^n\right]$

$$= \frac{\Delta t}{2}\left[f_j^{n+1} + f_j^n + \frac{1}{6}\left(f_{j+1}^n - 2f_j^n + f_{j-1}^n\right)\right]$$

$$= \frac{\Delta t}{2}\left[2\left(f + \Delta t f_t + \frac{\Delta t^2}{2!} f_{tt} + \cdots\right) + \frac{1}{6}\left(\frac{2h^2}{2!} f_{xx} + \frac{2h^4 f_{xxxx}}{4!} + \cdots\right)\right]$$

Equating Term ①, Term ② and Term ③, and dividing by Δt, we have

$$\left[u_t + \frac{\Delta t}{2!} u_{tt} + \cdots\right] + \frac{1}{12}\left[u_{xxxt} \frac{h^2 \Delta t}{2!1!} + \cdots\right]$$

$$= b\left[u_{xx} + \frac{\Delta t}{2!1!} u_{xxxt} + \frac{h^2}{12} u_{xxxx} + \cdots\right] + \left[f + \frac{\Delta t f_t}{2} + \frac{(\Delta t)^2 f_{tt}}{2!} + \cdots\right]$$

As $u_t = b\, u_{xx} + f$

So, the truncation Error (TE) is.

$$TE = \frac{\Delta t}{2!} u_{tt} + \cdots + \frac{u_{xxxt}\, h^2 \Delta t}{48} + \cdots$$

So, $\underline{TE = O(\Delta t, h^2)}$

So, truncation error has leading order terms of $O(\Delta t)$ in time and $O(h^2)$ in space.

Problem 3

Consider the DuFort-Frankel scheme for the diffusion equation $u_t = b\, u_{xx}$, where $b > 0$, on a uniform mesh

$$u_j^{n+1} = u_j^{n-1} + \frac{2b\Delta t}{h^2}\left(u_{j+1}^n - u_j^{n+1} - u_j^{n-1} + u_{j-1}^n\right)$$

Determine the restrictions on the time step Δt and grid spacing h required to ensure convergence.

Solution :-

DuFort Frankel Scheme :-

$$u_j^{n+1} = u_j^{n-1} + \frac{2b\Delta t}{h^2}\left(u_{j+1}^n - u_j^{n+1} - u_j^{n-1} + u_{j-1}^n\right) \quad \text{— (1)}$$

To ensure the convergence of the scheme, it is necessary and sufficient to have both consistency and stability.

Stability Analysis :-

According to Von-Neumann stability method, we consider $u_j^n \sim g^n e^{ij\theta}$ — (2)

where g = Amplification factor, $\theta = Kh$

K = Wave Number, h = grid spacing.

Using ① and ②,

$$g^{n+1}e^{ij\theta} = g^{n-1}e^{ij\theta} + \frac{2b\Delta t}{h^2} g^n e^{i(j+1)\theta}$$

$$- \frac{2b\Delta t}{h^2} g^{n+1} e^{ij\theta} - \frac{2b\Delta t}{h^2} g^{n-1} e^{ij\theta}$$

$$+ \frac{2b\Delta t}{h^2} g^n e^{i(j-1)\theta}$$

or $g^n e^{ij\theta}(g) = g^n e^{ij\theta}\left(\frac{1}{g}\right)$

$$+ g^n e^{ij\theta}\left(\frac{2b\Delta t}{h^2}\right)\left[e^{i\theta} - g - \frac{1}{g} + e^{-i\theta}\right]$$

Let $\mu = \left(\frac{b\,\Delta t}{h^2}\right)$ — ③

So, $g = \frac{1}{g} + 2\mu\left[2\cos\theta - g - \frac{1}{g}\right]$

or $g^2 = 1 + 2\mu\left[2g\cos\theta - g^2 - 1\right]$

or $(1+2\mu)g^2 - (4\mu\cos\theta)g + (2\mu - 1) = 0$

So, $g = \dfrac{-(-4\mu\cos\theta) \pm \sqrt{(-4\mu\cos\theta)^2 - 4(1+2\mu)(2\mu-1)}}{2(1+2\mu)}$

or $g = \dfrac{4\mu\cos\theta \pm \sqrt{(4\mu\cos\theta)^2 + 4(1-4\mu^2)}}{2(1+2\mu)}$

So, $$g = \frac{4\mu\cos\theta \pm \sqrt{4 + (4\mu)^2(\cos^2\theta - 1)}}{2(1+2\mu)}$$

or $$g = \frac{2\mu\cos\theta \pm \sqrt{1 + 4\mu^2(\cos^2\theta - 1)}}{(1+2\mu)}$$

or $$g = \frac{2\mu\cos\theta \pm \sqrt{1 - 4\mu^2\sin^2\theta}}{(1+2\mu)}$$

So, $$\boxed{g = \frac{2\mu\cos\theta \pm \sqrt{1 - 4\mu^2\sin^2\theta}}{(1+2\mu)}} \quad \text{— (4)}$$

<u>Case I</u> :- $2\mu \leq 1$

As, $0 \leq \sin^2\theta \leq 1$

or $0 \leq 4\mu^2\sin^2\theta \leq 4\mu^2$

(As $b > 0$, So, $\mu = \frac{b\Delta t}{h} > 0$)

So, $0 \geq -4\mu^2\sin^2\theta \geq -4\mu^2$

or $1 \geq 1 - 4\mu^2\sin^2\theta \geq 1 - 4\mu^2$

So, $1 - 4\mu^2\sin^2\theta \leq 1$ — (5)

If $$g = \frac{2\mu\cos\theta + \sqrt{1 - 4\mu^2\sin^2\theta}}{(1+2\mu)}$$

Using (5),

$$g \leq \frac{2\mu\cos\theta + 1}{1+2\mu} \leq \frac{2\mu+1}{1+2\mu}$$

(As $\cos\theta \leq 1$)

So, $g \leq 1$ — (6)

if $g = \dfrac{2\mu\cos\theta + \sqrt{1-4\mu^2\sin^2\theta}}{1+2\mu}$

Considering,

$$g = \frac{2\mu\cos\theta - \sqrt{1-4\mu^2\sin^2\theta}}{1+2\mu}$$

Or,

$$\frac{2\mu\cos\theta - \sqrt{1-4\mu^2\sin^2\theta}}{1+2\mu} < \frac{2\mu\cos\theta + \sqrt{1-4\mu^2\sin^2\theta}}{1+2\mu}$$

So, $g = \dfrac{2\mu\cos\theta - \sqrt{1-4\mu^2\sin^2\theta}}{1+2\mu} \leq 1$

Using (6)

— (7)

So, if $2\mu \leq 1$, then

$g \leq 1$ using ⑥ & ⑦

So, $\underline{\underline{|g| \leq 1}}$

<u>Case II</u>: $2\mu > 1$

So, $g = \dfrac{2\mu\cos\theta \pm \sqrt{1 - 4\mu^2\sin^2\theta}}{1+2\mu}$

$= \dfrac{2\mu\cos\theta \pm i\sqrt{4\mu^2\sin^2\theta - 1}}{(1+2\mu)}$

$|g|^2 = \dfrac{(2\mu\cos\theta)^2 + (4\mu^2\sin^2\theta - 1)}{(1+2\mu)^2}$

or $|g|^2 = \dfrac{4\mu^2(\sin^2\theta + \cos^2\theta) - 1}{(1+2\mu)^2}$

$= \dfrac{4\mu^2 - 1}{(1+2\mu)^2}$

$= \dfrac{(2\mu-1)(2\mu+1)}{(2\mu+1)(2\mu+1)}$

or $|g|^2 = \dfrac{2\mu - 1}{2\mu + 1}$

or $|g| = \sqrt{\dfrac{2\mu - 1}{2\mu + 1}}$

(As, $2\mu > 1$)

So, $|g| < 1$ if $2\mu > 1$

So, the scheme is <u>Unconditionally Stable</u>.

<u>Consistency Check:</u>

The given scheme

$$\frac{u_j^{n+1} - u_j^{n-1}}{2\Delta t} = \frac{b}{h^2}\left(u_{j+1}^n - u_j^{n+1} - u_j^{n-1} + u_{j-1}^n\right)$$

$$= b\left(\frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{h^2}\right) - b\left(\frac{u_j^{n+1} - 2u_j^n + u_j^{n-1}}{h^2}\right) \quad \text{—— (8)}$$

So,

$$u_j^{n+1} = u + (\Delta t)u_t + \frac{(\Delta t)^2}{2!}u_{tt} + \cdots$$

$$u_j^{n-1} = u - (\Delta t)u_t + \frac{(\Delta t)^2}{2!}u_{tt} - \cdots$$

$$u^n_{j+1} = u + (\Delta x)\, u_x + \frac{(\Delta x)^2}{2!} u_{xx} + \frac{(\Delta x)^3}{3!} u_{xxx} + \cdots$$

$$u^n_{j-1} = u - (\Delta x)\, u_x + \frac{(\Delta x)^2}{2!} u_{xx} - \frac{(\Delta x)^3}{3!} u_{xxx} + \cdots$$

Substituting these in (8),

$$\frac{\left(u + (\Delta t)\, u_t + \frac{(\Delta t)^2}{2!} u_{tt} + \cdots\right) - \left(u - \Delta t\, u_t + \frac{(\Delta t)^2}{2!} u_{tt} - \cdots\right)}{2\Delta t}$$

$$= \frac{b}{h^2}\left(u + (\Delta x)\, u_x + \frac{(\Delta x)^2}{2!} u_{xx} - 2u + u - (\Delta x)\, u_x + \frac{(\Delta x)^2}{2!} u_{xx} + H.O.T\right)$$

$$- \frac{b}{h^2}\left(u + (\Delta t)\, u_t + \frac{(\Delta t)^2}{2!} u_{tt} + \frac{(\Delta t)^3}{3!} u_{ttt} - 2u + u - \Delta t\, u_t + \frac{(\Delta t)^2}{2!} u_{tt} - \frac{(\Delta t)^3}{3!} u_{ttt} + H.O.T.\right)$$

or $$u_t + \frac{\Delta t^2}{6} u_{ttt} = b\left[u_{xx} + \frac{h^2}{12} u_{xxxx}\right] - b\left[\frac{\Delta t^2}{h^2} u_{tt} + \frac{(\Delta t)^4}{12 h^2} u_{tttt}\right] + H.O.T.$$

As $u_t = b\, u_{xx}$

So, $$\frac{\Delta t^2}{6} u_{ttt} = b\left[\frac{h^2}{12} u_{xxxx} + \frac{\Delta t^2}{h^2} u_{tt} + H.O.T\right]$$

So, the truncation error is

$$T.E. = \frac{(\Delta t)^2}{6} u_{ttt} - \frac{b h^2}{12} u_{xxxx} + \frac{b \Delta t^2}{h^2} u_{tt} + \underline{\underline{H.O.T.}}$$

Because of the term $\frac{b \Delta t^2}{h^2} u_{tt}$ as

$\Delta t \to 0$ and $h \to 0$ independently, the

T.E. ~~does NOT do~~ may NOT tends to 0.

So, the given scheme is <u>inconsistent</u>. ~~if~~ $\frac{\Delta t}{\Delta h}$ NOT tends to 0.

But if $\underline{\underline{\frac{\Delta t}{h} \to 0}}$, then the scheme

<u>becomes consistent</u> and hence we can achieve

<u>convergence</u> (It is unconditionally stable)

So, the <u>restriction</u> on Δt and h is that

$$\underline{\underline{\left(\frac{\Delta t}{h}\right) \to 0}} \qquad \left(\begin{matrix} h > 0 \\ \underline{\underline{\Delta t > 0}} \end{matrix}\right)$$

<u>to ensure convergence.</u>

So, the scheme is <u>conditionally convergent</u>.

Problem 4

Write a computer code to solve

$$u_t = \nabla^2 u + \sin(2\pi x)\sin(2\pi y)\sin(2\pi t)$$

for $(x, y) \in (0,1)^2$

$$u(x,0,t) = u(x,1,t) = 0$$

$$u(0,y,t) = u(1,y,t) = 0$$

Using Crank-Nicolson method. Use the matrix diagonalization method discussed in the class to solve the resulting linear system of equations: Plot the L_2 and L_∞ errors as a function of number of mesh nodes $N\left(h_x = h_y = \frac{1}{N}\right)$ and the time step size Δt. What rate of convergence do you observe in space and in time?

Solution:-

$$u_t = \nabla^2 u + \sin(2\pi x)\sin(2\pi y)\sin(2\pi t)$$

Let $f(x,y,t) = \sin(2\pi x)\sin(2\pi y)\sin(2\pi t)$

So, we have

$$u_t = \nabla^2 u + f(x,y,t)$$

Using <u>Crank - Nicolson</u> method,

$$\frac{u_{i,j}^{n+1} - u_{i,j}^{n}}{\Delta t} = \frac{1}{2h^2}\left[\frac{u_{i+1,j}^{n+1} - 2u_{i,j}^{n+1} + u_{i-1,j}^{n+1}}{1}\right]$$

$$+ \frac{1}{2h^2}\left[u_{i+1,j}^{n} - 2u_{i,j}^{n} + u_{i-1,j}^{n}\right]$$

$$+ \frac{1}{2h^2}\left[u_{i,j+1}^{n+1} - 2u_{i,j}^{n+1} + u_{i,j-1}^{n+1}\right]$$

$$+ \frac{1}{2h^2}\left[u_{i,j+1}^{n} - 2u_{i,j}^{n} + u_{i,j-1}^{n}\right]$$

$$+ \frac{1}{2}\left[f_{i,j}^{n+1} + f_{i,j}^{n}\right]$$

So,

$$u_{i,j}^{n+1} = u_{i,j}^{n} + \frac{\Delta t}{2h^2}\Big[\left(u_{i,j}^{n+1} - 2u_{i,j}^{n+1} + u_{i-1,j}^{n+1}\right)$$

$$+ \left(u_{i+1,j}^{n} - 2u_{i,j}^{n} + u_{i-1,j}^{n}\right)$$

$$+ \left(u_{i,j+1}^{n+1} - 2u_{i,j}^{n+1} + u_{i,j-1}^{n+1}\right)$$

$$+ \left(u_{i,j+1}^{n} - 2u_{i,j}^{n} + u_{i,j-1}^{n}\right)$$

$$+ h^2 f_{i,j}^{n+1} + h^2 f_{i,j}^{n}\Big]$$

For $D = \begin{bmatrix} -2 & 1 & 0 & \cdots & 0 & 0 \\ 1 & -2 & 1 & \cdots & 1 & 1 \\ \vdots & & \ddots & & \vdots & \vdots \\ 0 & \cdots & & 1 & -2 & 1 \\ 0 & \cdots & & 0 & 1 & -2 \end{bmatrix}$

Let $\nu = \frac{\Delta t}{2h^2}$

$F^n_{i,j} = h^2 f^n_{i,j}$

So, we can write this equation in the matrix form as,

$$[u^{n+1}] = [u^n] + \nu\Big[[D][u^{n+1}] + [u^{n+1}][D] + [F^{n+1}] + [D][u^n] + [u^n][D] + [F^n]\Big]$$

or $[u^{n+1}] - \nu[D][u^{n+1}] - \nu[u^{n+1}][D]$

$$= [u^n] + \nu\Big[[F^{n+1}] + [F^n] + [D][u^n] + [u^n][D]\Big]$$

or $\frac{1}{2}[I][u^{n+1}] - \nu[D][u^{n+1}]$

$$+ [u^{n+1}]\left(\frac{1}{2}[I]\right) - \nu[u^{n+1}][D]$$

$$= [u^n] + \nu\Big[[F^{n+1}] + [F^n] + [D][u^n] + [u^n][D]\Big]$$

or $\left[\frac{1}{2}[I] - \nu[D]\right][u^{n+1}]$

$$+ [u^{n+1}]\left[\frac{1}{2}[I] - \nu[D]\right]$$

$$= [u^n] + \nu\Big[[F^{n+1}] + [F^n] + [D][u^n] + [u^n][D]\Big]$$

Let $[B] = \frac{1}{2}[I] - \nu[D]$

So,

$$[B][u^{n+1}] + [u^{n+1}][B]$$

$$= [u^n] + \nu\left[[F^{n+1}] + [F^n] + [D][u^n] + [u^n][D]\right]$$

On diagonalizing $[B]$; we have

$$[B] = [P][\Lambda][P]^{-1}$$

So,

$$[P][\Lambda][P]^{-1}[u^{n+1}] + [u^{n+1}][P][\Lambda][P]^{-1}$$

$$= [u^n] + \nu\left[[F^{n+1}] + [F^n] + [D][u^n] + [u^n][D]\right]$$

or $[\Lambda][P]^{-1}[u^{n+1}] + [P]^{-1}[u^{n+1}][P][\Lambda][P]^{-1}$

$$= [P]^{-1}[u^n] + \nu[P]^{-1}\left[[F^{n+1}] + [F^n] + [D][u^n] + [u^n][D]\right]$$

or $[\Lambda][P]^{-1}[u^{n+1}][P]$

$$+ [P]^{-1}[u^{n+1}][P][\Lambda]$$

$$= [P]^{-1}[u^n][P]$$

$$+ \nu[P]^{-1}\left[[F^{n+1}] + [F^n] + [D][u^n] + [u^n][D]\right][P]$$

Let, ~~[V]~~

$$[V^{n+1}] = [P]^{-1}[u^{n+1}][P]$$

So,

$$[\Lambda][V^{n+1}] + [V^{n+1}][\Lambda]$$

$$= [P]^{-1}[U^n][P]$$
$$+ \nu [P]^{-1}\left[[F^{n+1}] + [F^n] + [D][U^n] + [U^n][D]\right][P]$$

So, $$V_{i,j}^{n+1} = \left(\frac{[P]^{-1}[U^n][P]}{\lambda_i + \lambda_j}\right)_{i,j}$$
$$+ \left(\frac{\nu [P]^{-1}\left[[F^{n+1}] + [F^n] + [D][U^n] + [U^n][D]\right][P]}{\lambda_i + \lambda_j}\right)_{i,j}$$

So, we can calculate $[V^{n+1}]$.

As $[V^{n+1}] = [P]^{-1}[U^{n+1}][P]$

So, $[U^{n+1}] = [P][V^{n+1}][P]^{-1}$

So, we can calculate $[U^{n+1}]$

<u>NOTE</u>: We know eigen values (λ_D) of D is

$$\lambda_D^{(K)} = -4\sin^2\left(\frac{\pi K}{2N}\right) \quad [K = 1, 2, \ldots (N-1)]$$

As $[B] = \frac{1}{2}[I] - \nu[D]$

So the eigen values (λ) of $[B]$ is

$$\lambda^{(K)} = \frac{1}{2} - \nu\left[-4\sin^2\left(\frac{\pi K}{2N}\right)\right]$$

$$\lambda^{(K)} = \frac{1}{2} + 4\nu\left[\sin^2\left(\frac{\pi K}{2N}\right)\right]$$

$$(K = 1, 2, \ldots, N-1)$$

Analytical Solution :-

Let $u(x,y,t) = \sum_{n=1}^{\infty} \sum_{m=1}^{\infty} a_{nm}(t) \sin(n\pi x) \sin(m\pi x)$

We have

$$u_t = \nabla^2 u + \sin(2\pi x)\sin(2\pi y)\sin(2\pi t)$$

$$R.H.S. = \nabla^2 u + \sin(2\pi x)\sin(2\pi y)\sin(2\pi t)$$

$$= -(n^2\pi^2 + m^2\pi^2) \sum_{n=1}^{\infty} \sum_{m=1}^{\infty} a_{nm}(t) \sin(n\pi x)\sin(m\pi x)$$

$$+ \sin(2\pi x)\sin(2\pi y)\sin(2\pi t)$$

$$L.H.S. = u_t$$

$$= \sum_{n=1}^{\infty} \sum_{m=1}^{\infty} \dot{a}_{nm}(t) \sin(n\pi x)\sin(m\pi x)$$

On comparing LHS and RHS we have,

$$\dot{a}_{nm}(t) = -(n^2\pi^2 + m^2\pi^2)\, a_{nm}(t) \quad \forall$$

(except $n = m = 2$) — Case ①

If $n = m = 2$, then

$$\dot{a}_{22}(t) = -(2^2\pi^2 + 2^2\pi^2)\, a_{22}(t) + \sin(2\pi t)$$

— Case ②

Considering Case ①,

$$\frac{da_{nm}}{dt} = -(n^2 + m^2)\pi^2 a_{nm}$$

So, $a_{nm}(t) = C e^{-(m^2+n^2)\pi^2 t}$

[except $(n = m = 2)$]

Initial condition, $u(x, y, 0) = 0$

So, $a_{nm}(0) = 0$

or $C = 0$

So, $a_{nm}(t) = 0$ (Except $n = m = 2$)

Considering case ②, $(m = n = 2)$

$\dot{a}_{22}(t) = -8\pi^2 a_{22}(t) + \sin(2\pi t)$

or $\dot{a}_{22}(t) + 8\pi^2 a_{22}(t) = \sin(2\pi t)$

Solving for homogeneous part,

$\dot{a}_{22}(t) + 8\pi^2 a_{22}(t) = 0$

So, $a_{22}(t) = C_1 e^{-8\pi^2 t}$

For the non homogeneous part,

Let $a_{22}(t) = A\sin(2\pi t) + B\cos(2\pi t)$

So, $\dot{a}_{22}(t) + 8\pi^2 a_{22}(t) = \sin(2\pi t)$

or $2\pi A\cos(2\pi t) - B(2\pi)\sin(2\pi t)$
$+ 8\pi^2 (A\sin(2\pi t) + B\sin(2\pi t)) = \sin(2\pi t)$

or $(A(8\pi^2) - B(2\pi))\sin(2\pi t)$
$+ (B(8\pi^2) + A(2\pi))\cos(2\pi t) = \sin(2\pi t)$

On comparing,

$$(A)(8\pi^2) - (B)(2\pi) = 1$$

$$(B)(8\pi^2) + A(2\pi) = 0$$

So, $A = -4\pi B$

~~So, $(A)(8\pi^2) - (-4\pi B)($~~

So, $(-4\pi B)(8\pi^2) - B(2\pi) = 1$

or $B(-2\pi - 32\pi^3) = 1$

or $B = \dfrac{-1}{2\pi + 32\pi^3}$

So, $A = -4\pi B$

or $A = \dfrac{4\pi}{2\pi + 32\pi^3}$

or $A = \dfrac{2}{1 + 16\pi^2}$

So, $a_{22}(t) = \dfrac{2\sin(2\pi t)}{1+16\pi^2} - \dfrac{\cos(2\pi t)}{2\pi(1+16\pi^2)}$

So, $a_{22}(t) = Ge^{-8\pi^2 t} + \dfrac{2\sin(2\pi t)}{(1+16\pi^2)}$

$$- \frac{\cos(2\pi t)}{2\pi(1+16\pi^2)}$$

As $u(x,y,0)=0$

So, $a_{22}(0)=0$

So, $C_1 = \dfrac{1}{2\pi(1+16\pi^2)}$

~~So, $a_{22}(t) = e^{-8\pi^2 t}$~~

So,

$$a_{22}(t) = \frac{e^{-8\pi^2 t}}{2\pi(1+16\pi^2)} + \frac{2\sin(2\pi t)}{(1+16\pi^2)} - \frac{\cos(2\pi t)}{2\pi(1+16\pi^2)}$$

So,

$$u(x,y,t) = \left(\frac{e^{-8\pi^2 t}\sin(2\pi x)\sin(2\pi y)}{2\pi(1+16\pi^2)}\right) + \left(\frac{2\sin(2\pi t)\sin(2\pi x)\sin(2\pi y)}{(1+16\pi^2)}\right) + \left(\frac{-\cos(2\pi t)\sin(2\pi x)\sin(2\pi y)}{2\pi(1+16\pi^2)}\right)$$

This is the analytical solution.

Question 4:

CODE:

```
#include <iostream>
#include <cassert>
#include<algorithm>
#include<vector>
#include<fstream>
#include<cmath>
#include<string>
#include<unistd.h>
#include<functional>

using namespace std;

typedef double real;

double const pi = 4.0*atan(1.0);

// Generate 1D grid
void generate_1D_grid(vector<real> &x, real X0, real Xl, real N)
{
    for (int i = 0; X0 + ((i/N)*(Xl-X0)) <= Xl;i++)
    {
        x.push_back(X0 + ((i/N)*(Xl-X0)));
        //cout<<X0 + ((i/N)*(Xl-X0))<<endl;
    }
}

// This function calculates the transpose of a given matrix
void matrix_transpose(vector<vector<real>>& A, vector<vector<double>> &A_Transpose)
{
    int rows = A.size();
    int cols = A[0].size();

    for (int row = 0; row < rows; row++)
    {
        for (int col = 0; col < cols; col++)
        {
            A_Transpose[col][row] = A[row][col];
        }
    }
}

// This function calculates the eigen values , eigen vector matrix and inverse of eigen vector matrix
void compute_eigen_value_vector_inverse(vector<vector<real>> &P, vector<vector<real>>
&PINV, vector<real> &eigen_values, vector<real> &x,real Nx,real hx,real nu)
{
    // Set up eigenvalues and matrices
    int i, j;
```

```
        for(i = 1 ; i < Nx ; i++) {
                eigen_values.push_back(0.5-(-nu*4.0*sin(pi*0.5*x[i])*sin(pi*0.5*x[i])));
                for(j = 1 ; j < Nx ; j++) {
                        P[i-1][j-1] = sin(i*pi*x[j])/sqrt((Nx-1.0)/2.0) ;
                        PINV[i-1][j-1] = sin(j*pi*x[i])/sqrt((Nx-1.0)/2.0) ;
                }
        }
}

// This function calculate the F of the equation at time t
void compute_F(vector<vector<real>> &F, vector<real> &x, vector<real> &y, real h, real t)
{
   int rows = F.size();
   int cols = F[0].size();
   real pi = 4.0*atan(1);

   for (int row = 0; row < rows; row++)
   {
      for (int col = 0; col < cols; col++)
      {
         F[row][col] = h*h*sin(2.0*pi*x[row+1])*sin(2.0*pi*y[col+1])*sin(2.0*pi*t);
      }
   }
}

// This function calculate the analytical solution
void compute_analytical_sol(vector<vector<real>> &sol, vector<real> &x, vector<real> &y, real h,
real t)
{
   int rows = sol.size();
   int cols = sol[0].size();
   real pi = 4.0*atan(1);
   real temp = (1.0 + (16*pi*pi));

   for (int row = 0; row < rows; row++)
   {
      for (int col = 0; col < cols; col++)
      {
         sol[row][col] =
sin(2.0*pi*x[row+1])*sin(2.0*pi*y[col+1])*((exp(-8.0*pi*pi*t)/(2*pi*temp)) +
(2.0*sin(2*pi*t)/temp) - (cos(2.0*pi*t)/(2*pi*temp)));
      }
   //cout<<x[row+1]<<endl;
   }
}



// This function computes the product of two given matrix
void matrix_multiply(vector<vector<real>> &A, vector<vector<double>> &B,
vector<vector<double>> &C)
{
```

```
    int m = A.size();
    int n = B.size();
    int p = B[0].size();

    // vector<vector<double>> C(m, vector<double>(p, 0));

    for (int i = 0; i < m; i++) {
        for (int j = 0; j < p; j++) {
            for (int k = 0; k < n; k++) {
                C[i][j] += A[i][k] * B[k][j];
            }
        }
    }

    //return C;
}

// This function computes the addition of two given matrix
void matrix_addition(vector<vector<real>> &A, vector<vector<double>> &B,
vector<vector<double>> &C)
{
    int m = A.size();
    int n = A[0].size();
    // int n = B.size();
    // int p = B[0].size();

    // vector<vector<double>> C(m, vector<double>(p, 0));

    for (int j = 0; j < n; j++) {
            for (int i = 0; i < m; i++) {
                C[i][j] = A[i][j] + B[i][j];
            }
        }

    //return C;
}

// This function computes the addition of two given matrix
void matrix_value_transfer(vector<vector<real>> &A, vector<vector<double>> &B)
{
    int m = A.size();
    int n = A[0].size();
    // int n = B.size();
    // int p = B[0].size();

    // vector<vector<double>> C(m, vector<double>(p, 0));

    for (int j = 0; j < n; j++) {
            for (int i = 0; i < m; i++) {
                A[i][j] = B[i][j];
            }
        }
```

```
    //return C;
}

// This function computes the scalar multiplication of a given matrix
void matrix_scalar_multiply(vector<vector<real>> &A,real nu, vector<vector<double>> &C)
{
    int m = A.size();
    int n = A[0].size();

    for (int j = 0; j < n; j++) {
            for (int i = 0; i < m; i++) {
                C[i][j] = nu*A[i][j];
            }
        }

    //return C;
}

// This function computes the V matrix
void compute_V(vector<vector <real>> &F_tilde, vector<real> &eigen_values,
vector<vector<real>> &V)
{
    int rows = F_tilde.size();
    int cols = F_tilde[0].size();

    for (int row = 0; row < rows; row++)
    {
        for (int col = 0; col < cols; col++)
        {
            V[row][col] = (F_tilde[row][col])/(eigen_values[row] + eigen_values[col]);
        }
    }
}

// This function computes the V matrix
void compute_D(vector<vector <real>> &D)
{
    int rows = D.size();
    int cols = D[0].size();

    for (int row = 0; row < rows; row++)
    {
            D[row][row] = -2.0;
    }
    for (int row = 0; row < rows-1; row++)
    {
            D[row][row+1] = 1;
    }
    for (int row = 1; row < rows; row++)
    {
            D[row][row-1] = 1;
```

```
    }
}

// Function to save a vector<double> to a file of given name
void write_to_file(vector<real> &u, string str)
{
    ofstream file;
    // Open the file
    file.open(str);

    // If the file is NOT opened
    if( !file )
    {
        // Error Message if the file couldn't be opened
        cerr << "Error: file could not be opened" << endl;
        exit(1);
    }

    cout<<"Output file "<< str <<" is opened."<<endl;

    // Writing the vector values to the file in scientific notation
    for(int i=0;i<u.size();i++)
    {
        file<<u[i]<<scientific<<endl;
    }

    // Closing the file
    file.close();

    cout<<"Output file "<< str <<" is closed."<<endl;
    cout<<endl;
}

// Function to save a vector<string> to a file of given name
void write_to_file(vector<string> &u, string str)
{
    ofstream file;
    // Open the file
    file.open(str);

    // If the file is NOT opened
    if( !file )
    {
        // Error Message if the file couldn't be opened
        cerr << "Error: file could not be opened" << endl;
        exit(1);
    }

    cout<<"Output file "<< str <<" is opened."<<endl;

    // Writing the vector values to the file in scientific notation
    for(int i=0;i<u.size();i++)
```

```
    {
        file<<u[i]<<endl;
    }

    // Closing the file
    file.close();

    cout<<"Output file "<< str <<" is closed."<<endl;
    cout<<endl;
}

// Function to save a vector<vector <double>> (MATRIX) to a file of given name
void write_to_file(vector<vector <real> > &u, string str)
{
    ofstream file;
    // Open the file
    file.open(str);

    // If the file is NOT opened
    if( !file )
    {
        // Error Message if the file couldn't be opened
        cerr << "Error: file could not be opened" << endl;
        exit(1);
    }

    cout<<"Output file "<< str <<" is opened."<<endl;

    int rows = u.size();
    int cols = u[0].size();

    // Writing the vector values to the file in scientific notation

    for (int row = 0; row < rows; row++)
    {
        for (int col = 0; col < cols; col++)
        {
            file<<u[row][col]<<scientific<<",";
        }
        file<<endl;
    }

    // Closing the file
    file.close();

    cout<<"Output file "<< str <<" is closed."<<endl;
    cout<<endl;
}

double calculate_average(const std::vector<real>& v) {
    real sum = 0;
    for (int i = 0; i < v.size(); i++) {
```

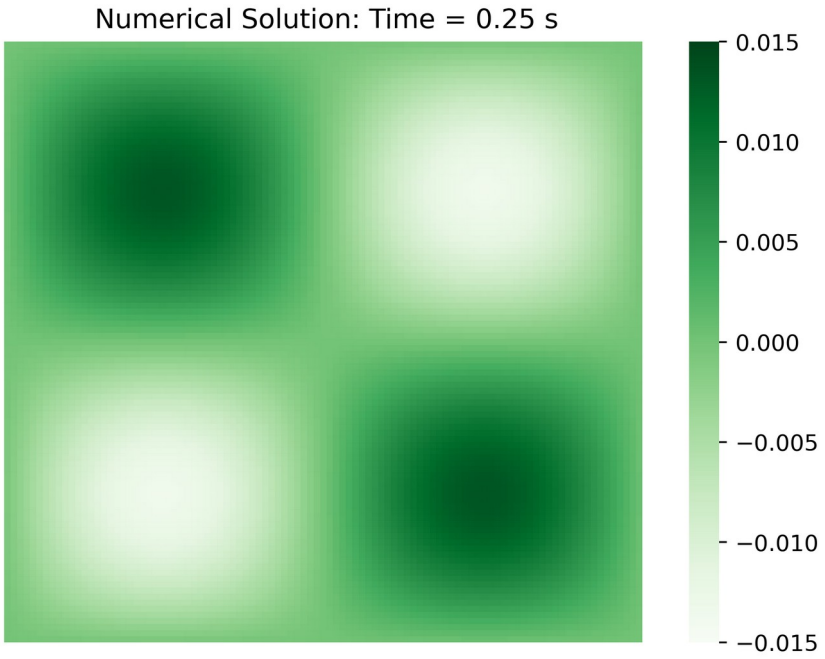
```
        sum += v[i];
    }
    double avg = static_cast<double>(sum) / v.size();
    return avg;
}

int calculate_max(const std::vector<real>& v) {
    real max = v[0];
    for (int i = 1; i < v.size(); i++) {
        if (v[i] > max) {
            max = v[i];
        }
    }
    return max;
}


int main()
{
    cout << "Hello world!" << endl;
    real dt,nu;

    real X0,Xl,Y0,Yl,h;
    int N=100;

    // Given the domain of the solution
    X0 = 0;
    Xl = 1;
    Y0 = 0;
    Yl = 1;

    h = 1.0/N;

    dt = 0.005;
    nu = dt/(2.0*h*h);

    nu = dt/(2.0*h*h);
    int rows, cols;
    rows = N-1;
    cols = N-1;


    vector<real> x;
    vector<real> y;
    vector<vector <real>> U_extended(N,vector<real> (N,0.0));

    // Generating the Grids
    generate_1D_grid(x,X0,Xl,N);
    generate_1D_grid(y,Y0,Yl,N);

    vector<vector <real> > P(rows,vector<real> (cols,0.0));
    vector<vector <real> > P_INV(rows,vector<real> (cols,0.0));
```

```
vector<real> eigen_values;
// Computing the eigen values, eigen vectors and the inverse of the matrix of eigen vectors
compute_eigen_value_vector_inverse(P,P_INV,eigen_values,x,N,h,nu);

write_to_file(eigen_values,"Eigen_Value_.csv");

// Vector to store the names of all the output files generated
vector<string> Output_file_names;
vector<string> Analytical_Output_file_names;

vector<vector <real> > U_n(rows,vector<real> (cols,0.0));
vector<vector <real> > Analytical_U_n(rows,vector<real> (cols,0.0));

vector<real> L2_error_Vector, max_error_vector;

real t_end;
t_end = 1.0;
t_end = t_end+dt;
for (real t = 0.0; t <= t_end; t = t+dt){
    cout<<"Time = "<<t<<endl;
    vector<vector <real> > F_n(rows,vector<real> (cols,0.0));
    compute_F(F_n,x,y,h,t);

    vector<vector <real> > F_n_plus_1(rows,vector<real> (cols,0.0));
    compute_F(F_n_plus_1,x,y,h,t+dt);

    vector<vector <real> > D(rows,vector<real> (cols,0.0));
    compute_D(D);

    vector<vector <real> > temp(rows,vector<real> (cols,0.0));
    matrix_multiply(U_n,D,temp);

    vector<vector <real> > temp_1(rows,vector<real> (cols,0.0));
    matrix_multiply(D,U_n,temp_1);

    vector<vector <real> > temp_2(rows,vector<real> (cols,0.0));
    matrix_addition(F_n,F_n_plus_1,temp_2);

    vector<vector <real> > temp_3(rows,vector<real> (cols,0.0));
    matrix_addition(temp_2,temp_1,temp_3);

    vector<vector <real> > temp_4(rows,vector<real> (cols,0.0));
    matrix_addition(temp_3,temp,temp_4);

    vector<vector <real> > temp_5(rows,vector<real> (cols,0.0));
    matrix_scalar_multiply(temp_4,nu,temp_5);

    vector<vector <real> > temp_6(rows,vector<real> (cols,0.0));
    matrix_addition(U_n,temp_5,temp_6);

    vector<vector <real> > temp_7(rows,vector<real> (cols,0.0));
    matrix_multiply(temp_6,P,temp_7);
```

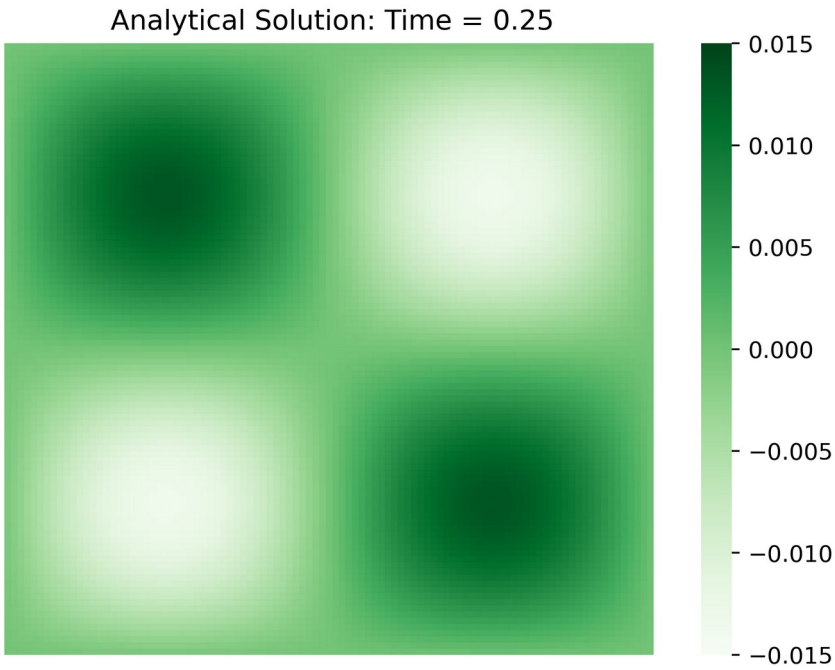
```
vector<vector <real> > F_tilde(rows,vector<real> (cols,0.0));
matrix_multiply(P_INV,temp_7,F_tilde);


vector<vector <real> > V(rows,vector<real> (cols,0.0));
compute_V(F_tilde,eigen_values,V);

vector<vector <real> > temp_9(rows,vector<real> (cols,0.0));
matrix_multiply(P,V,temp_9);

vector<vector <real> > U_n_plus_1(rows,vector<real> (cols,0.0));
matrix_multiply(temp_9,P_INV,U_n_plus_1);

compute_analytical_sol(Analytical_U_n,x,y,h,t);

// Writing the Numerical Solution to the file
write_to_file(U_n,"Q_4_Numerical_Solution_at_time_"+to_string(t)+"_.csv");

// Writing the Analytical Solution to the file
write_to_file(Analytical_U_n,"Q_4_Analytical_Solution_at_time_"+to_string(t)+"_.csv");

Output_file_names.push_back("Q_4_Numerical_Solution_at_time_"+to_string(t)+"_.csv");

Analytical_Output_file_names.push_back("Q_4_Analytical_Solution_at_time_"+to_string(t)
+"_.csv");

write_to_file(Output_file_names,"Output_file_names.csv");

write_to_file(Analytical_Output_file_names,"Analytical_Output_file_names.csv");

real Max_Error, L2_Error;
Max_Error = L2_Error = 0.0 ;


// Write a Output file showing values of x,y, Numerical Solution and Analytical Solution
ofstream File("Q_4_Output_"+to_string(N)+"_.dat", ios::out) ;
File.flags( ios::dec | ios::scientific );
File.precision(16) ;
if(!File) {cerr<< "Error: Output file couldnot be opened.\n";}

File << "TITLE = Flow" << endl << "VARIABLES = X, Y, u, Exact " << endl;
File << "Zone T = psi I = " << N+1 << " J = " << N+1 << endl ;
int i,j;

for(i = 0 ; i < N-1 ; i++) {
    for(j = 0 ; j < N-1 ; j++) {
        if( fabs(U_n[i][j] - Analytical_U_n[i][j] ) > Max_Error)
        {
            Max_Error = fabs( U_n[i][j] - Analytical_U_n[i][j] );
        }
```

```
            L2_Error += (U_n[i][j] - Analytical_U_n[i][j])*(U_n[i][j] - Analytical_U_n[i][j] )/
( ( N+1.0 )*(N+1.0) ) ;

            File << x[i] << "\t" << y[j] << "\t" << U_n[i][j] << "\t" << Analytical_U_n[i][j] << endl ;
        }
    }


    L2_Error = sqrt(L2_Error) ;
    File.close() ;
    // Printing the value of N
    cout<<"For N = "<<N<<endl;

    L2_error_Vector.push_back(L2_Error);
    max_error_vector.push_back(Max_Error);

    matrix_value_transfer(U_n,U_n_plus_1);

  }

  vector<real> L2_Error;
  vector<real> Max_Error;

  L2_Error.push_back(calculate_average(L2_error_Vector));

  Max_Error.push_back(calculate_max(max_error_vector));

  // Writing the L2 Errors in a file
  write_to_file(L2_Error,"Q_4_L2_Error_h_"+to_string(h)+"_dt_"+to_string(dt)+".csv");

  // Writing the Max Errors in a file
  write_to_file(Max_Error,"Q_4_max_error_h_"+to_string(h)+"_dt_"+to_string(dt)+".csv");

  return 0;
}
```

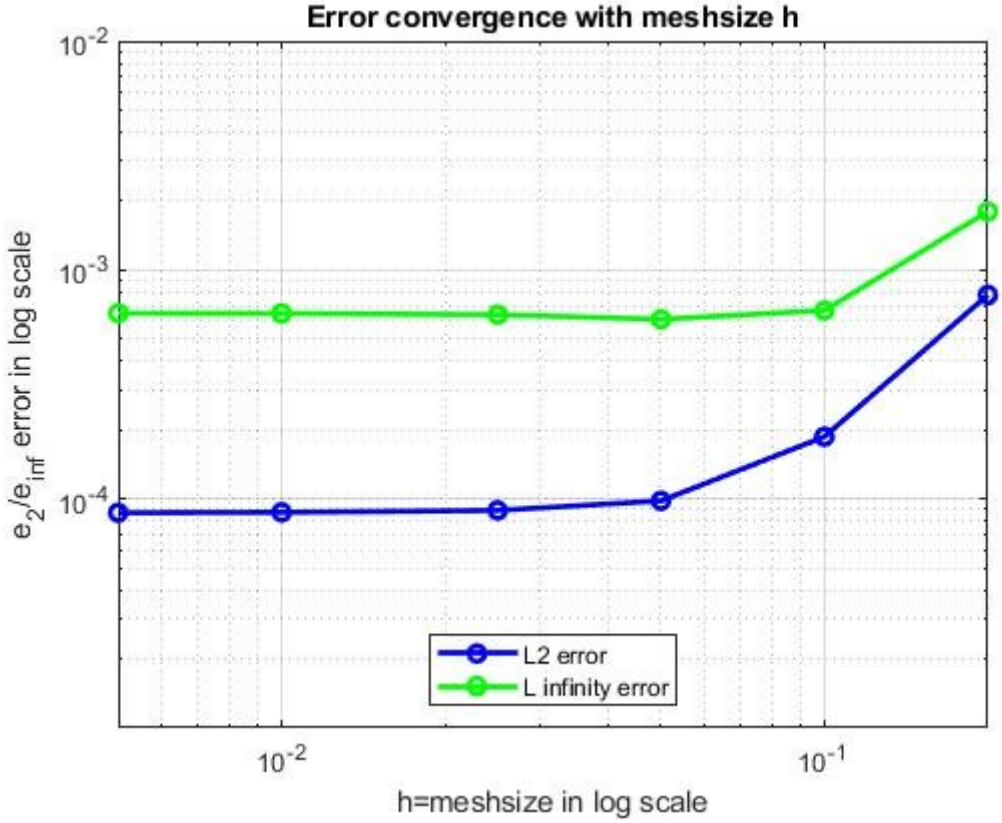
Numerical Solution at t = 0.25
h = 0.01
dt = 0.005



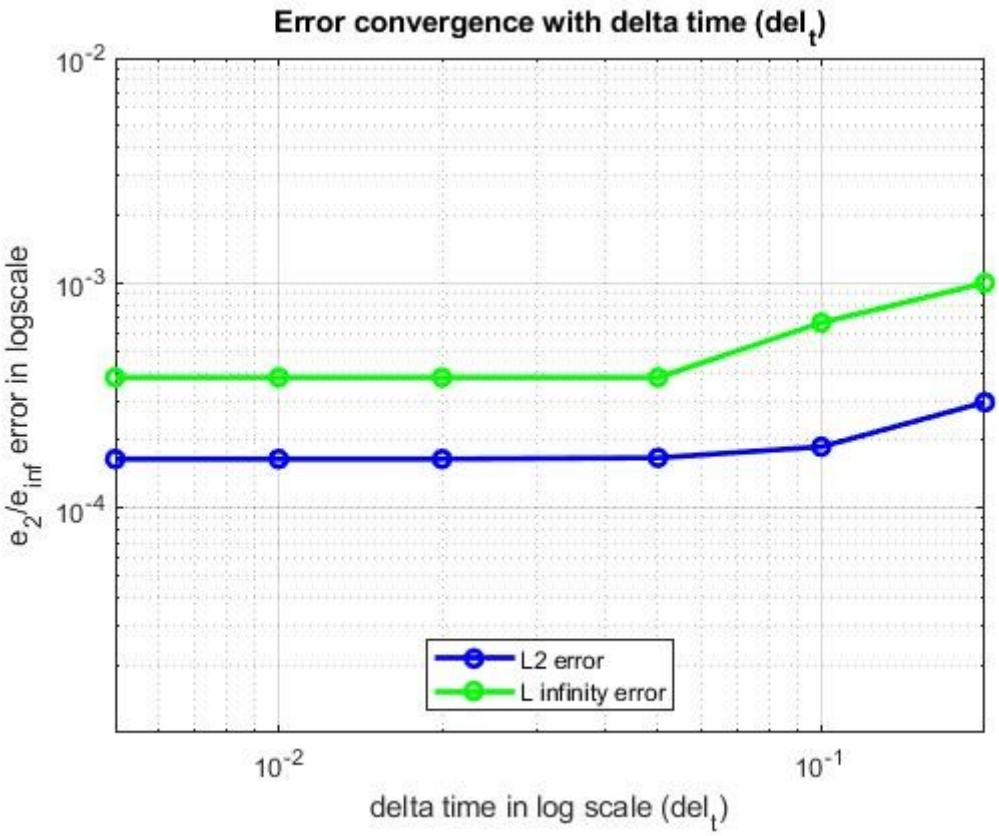
Analytical Solution at t = 0.25
h = 0.01
dt = 0.005

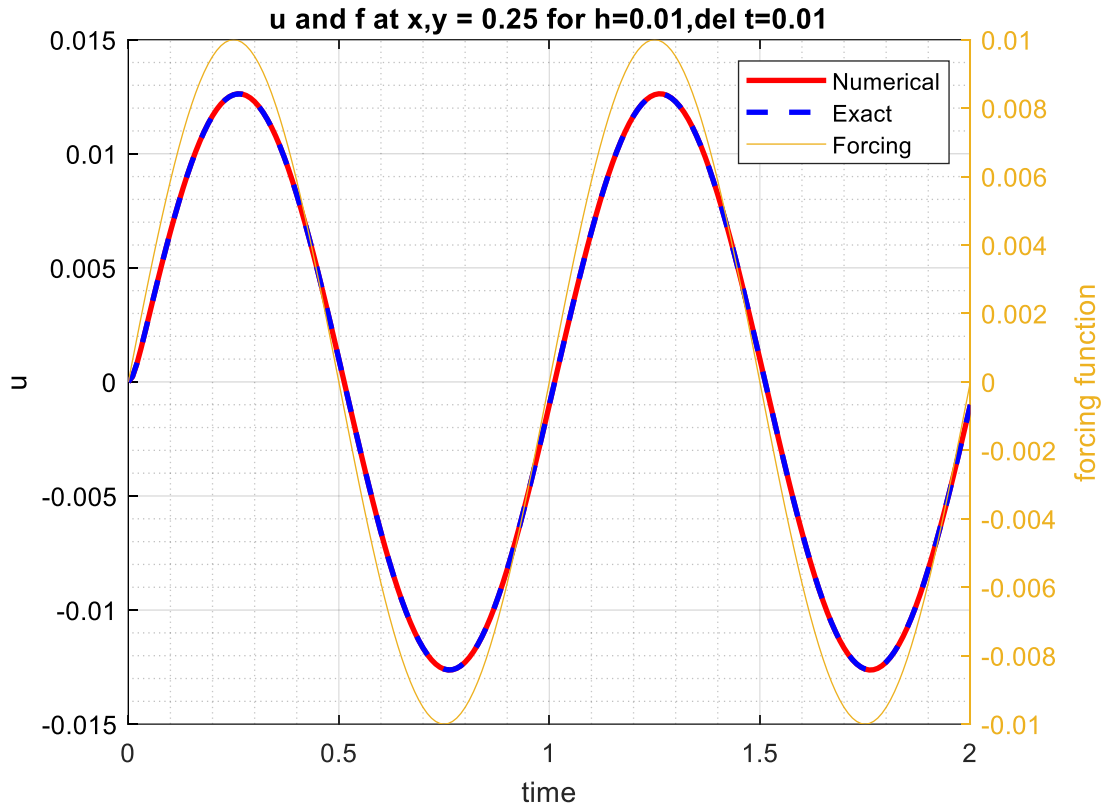


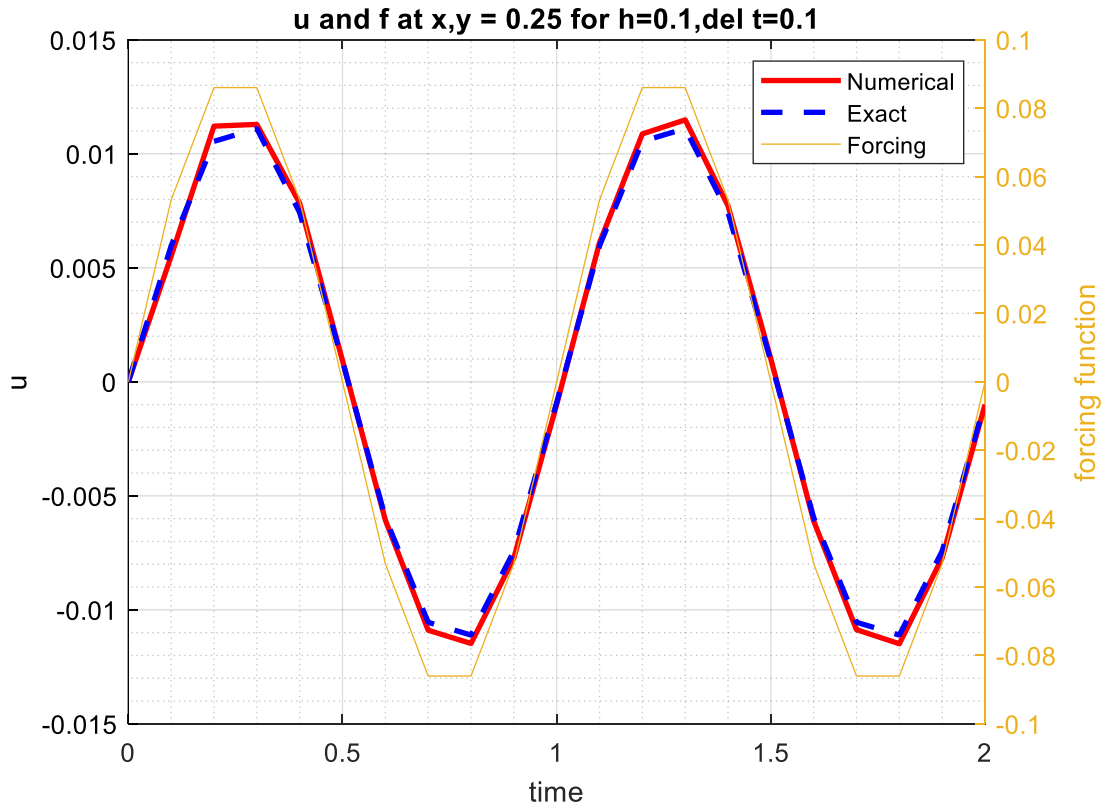
Delta time = dt = 0.1 (constant)



Mesh Size = h = 0.1 (constant)




u and f at x,y = 0.25 for h=0.01,del t=0.01
Numerical
Exact
Forcing
u
forcing function
time
0.015
0.01
0.005
0
-0.005
-0.01
-0.015
0.01
0.008
0.006
0.004
0.002
0
-0.002
-0.004
-0.006
-0.008
-0.01
0
0.5
1
1.5
2


u and f at x,y = 0.25 for h=0.1,del t=0.1
Numerical
Exact
Forcing
u
forcing function
time
0.015
0.01
0.005
0
-0.005
-0.01
-0.015
0.1
0.08
0.06
0.04
0.02
0
-0.02
-0.04
-0.06
-0.08
-0.1
0
0.5
1
1.5
2