HW2, Math 228B

Date due 2/15/2011

March 2011
University of California, Davis

Nasser M. Abbasi

March 2011 Compiled on February 27, 2021 at 3:04am

Contents

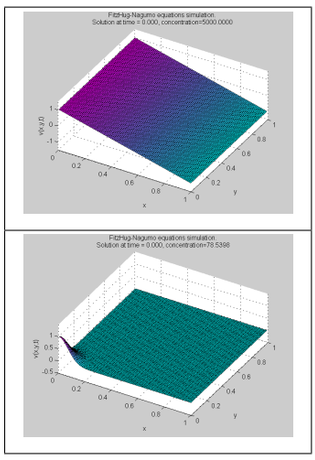
1 Animation of FitzHugh-Nagumo equations

The following are animated GIFs showing the solution to problem 3, parts (b) and (c). These will show only in the HTML version.

Assuming that $f(v) = (a - v)(v - 1)v$, the equations solved are the following

$$\frac{\partial v}{\partial t} = D\Delta v + f(v) - w + I$$
$$\frac{\partial w}{\partial t} = \epsilon\left(v - \gamma w\right)$$

Click on image to see the animation run, it will open in new window.



2 Problem1

Math 228B
Homework 2
Due Tuesday, February 15th

1. In class, we showed that the two-dimensional Peaceman-Rachford ADI scheme is unconditionally stable and second-order accurate in time. This can be thought of as either an approximate factorization or as a fractional step method. By adapting the fractional step idea to three-dimensions we get the scheme

$$\left(I - \frac{b\Delta t}{3}L_x\right)u^* = \left(I + \frac{b\Delta t}{3}L_y + \frac{b\Delta t}{3}L_z\right)u^n$$
$$\left(I - \frac{b\Delta t}{3}L_y\right)u^{**} = \left(I + \frac{b\Delta t}{3}L_x + \frac{b\Delta t}{3}L_z\right)u^*$$
$$\left(I - \frac{b\Delta t}{3}L_z\right)u^{n+1} = \left(I + \frac{b\Delta t}{3}L_x + \frac{b\Delta t}{3}L_y\right)u^{**}.$$

   (a) Use von Neumann analysis to show that this scheme is conditionally stable. This is an example of how certain desirable properties of a numerical scheme can be lost when using fractional stepping.

   (b) What temporal accuracy do you expect from this scheme? Explain.

Figure 1: Problem description

2.1 Part (a)

The diffusion PDE is given by

$$u_t - D\Delta u = 0$$

Where D is the diffusion constant. The ADI scheme in 3D [1] is given by

$$\left(I - \frac{D\Delta t}{3}L_x\right)u^* = \left(I + \frac{D\Delta t}{3}L_y + \frac{D\Delta t}{3}L_z\right)u^n \tag{1}$$
$$\left(I - \frac{D\Delta t}{3}L_y\right)u^{**} = \left(I + \frac{D\Delta t}{3}L_x + \frac{D\Delta t}{3}L_z\right)u^* \tag{2}$$
$$\left(I - \frac{D\Delta t}{3}L_z\right)u^{n+1} = \left(I + \frac{D\Delta t}{3}L_x + \frac{D\Delta t}{3}L_y\right)u^{**} \tag{3}$$

Where L_x, L_y, L_z are each the 1D Laplacian given by $\frac{1}{h^2}\begin{bmatrix} -2 & 1 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & 0 \\ \dots & \dots & \dots & \ddots & \dots & \dots \\ 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 1 & -2 \end{bmatrix}$

Assuming that the spatial frequencies in each of the three Cartesian directions (x, y, z) are given by ξ_x, ξ_y, ξ_z where $\frac{\pi}{h} \leq \xi_i \leq \frac{\pi}{h}$ and by setting $r = \frac{D\Delta t}{3h^2}$, $u^* = g^* e^{i\left(\xi_1 x + \xi_2 y + \xi_3 z\right)}$ and $u = e^{i\left(\xi_1 x + \xi_2 y + \xi_3 z\right)}$ and substituting these into Eq. (1) and dividing throughout by $e^{i\left(\xi_1 x + \xi_2 y + \xi_3 z\right)}$

[1] Please see the appendix of this problem at the end of the HW report showing how these equations came about.

gives the following

$$\begin{aligned} g^*\left(1 - r\left(e^{-i\xi_1 h} - 2 + e^{i\xi_1 h}\right)\right) &= 1 + r\left(e^{-i\xi_2 h} - 2 + e^{i\xi_2 h}\right) + r\left(e^{-i\xi_3 h} - 2 + e^{i\xi_3 h}\right) \\ g^* &= \frac{1 - 4r + r\left(e^{-i\xi_2 h} + e^{i\xi_2 h}\right) + r\left(e^{-i\xi_3 h} + e^{i\xi_3 h}\right)}{\left(1 + 2r - r\left(e^{i\xi_1 h} + e^{-i\xi_1 h}\right)\right)} \\ &= \frac{1 - 4r + 2r\cos(\xi_2 h) + 2r\cos(\xi_3 h)}{1 + 2r - 2r\cos(\xi_1 h)} \\ &= \frac{1 - 4r\left(\sin^2\left(\frac{\xi_2 h}{2}\right) + \sin^2\left(\frac{\xi_3 h}{2}\right)\right)}{1 + 4r\sin^2\left(\frac{\xi_1 h}{2}\right)} \end{aligned} \tag{4}$$

The last step above was obtained by the use of the relation $\cos A = 1 - 2\sin^2\left(\frac{A}{2}\right)$.

Applying the same method used above to Eq. (2), but now letting $u^* = g^* e^{i\left(\xi_1 x + \xi_2 y + \xi_3 z\right)}$ and $u^{**} = g^{**} e^{i\left(\xi_1 x + \xi_2 y + \xi_3 z\right)}$ and dividing throughout by $e^{i\left(\xi_1 x + \xi_2 y + \xi_3 z\right)}$ gives

$$\begin{aligned} g^{**}\left(1 - r\left(e^{-i\xi_2 h} - 2 + e^{i\xi_2 h}\right)\right) &= 1 + r\left(e^{-i\xi_1 h} - 2 + e^{i\xi_1 h}\right) + r\left(e^{-i\xi_3 h} - 2 + e^{i\xi_3 h}\right) \\ g^{**} &= \frac{1 - 4r + r\left(e^{-i\xi_1 h} + e^{i\xi_1 h}\right) + r\left(e^{-i\xi_3 h} + e^{i\xi_3 h}\right)}{1 + 2r - r\left(e^{i\xi_2 h} + e^{-i\xi_2 h}\right)} g^* \\ &= \frac{1 - 4r + 2r\cos(\xi_1 h) + 2r\cos(\xi_3 h)}{1 + 2r - 2r\cos(\xi_2 h)} g^* \\ &= \frac{1 - 4r\left(\sin^2\left(\frac{\xi_1 h}{2}\right) + \sin^2\left(\frac{\xi_3 h}{2}\right)\right)}{1 + 4r\sin^2\left(\frac{\xi_2 h}{2}\right)} g^* \end{aligned} \tag{5}$$

Again, applying the same method to Eq. (3), but now letting $u^{**} = g^{**} e^{i\left(\xi_1 x + \xi_2 y + \xi_3 z\right)}$ and $u^{n+1} = g e^{i\left(\xi_1 x + \xi_2 y + \xi_3 z\right)}$ and dividing by $e^{i\left(\xi_1 x + \xi_2 y + \xi_3 z\right)}$ gives

$$\begin{aligned} g\left(1 - r\left(e^{-i\xi_3 h} - 2 + e^{i\xi_3 h}\right)\right) &= 1 + r\left(e^{-i\xi_1 h} - 2 + e^{i\xi_1 h}\right) + r\left(e^{-i\xi_2 h} - 2 + e^{i\xi_2 h}\right) \\ g &= \frac{1 - 4r + r\left(e^{-i\xi_1 h} + e^{i\xi_1 h}\right) + r\left(e^{-i\xi_2 h} + e^{i\xi_2 h}\right)}{1 - r\left(e^{-i\xi_3 h} - 2 + e^{i\xi_3 h}\right)} g^{**} \\ &= \frac{1 - 4r + 2r\cos(\xi_1 h) + 2r\cos(\xi_2 h)}{1 + 2r - 2r\cos(\xi_3 h)} g^{**} \\ &= \frac{1 - 4r\left(\sin^2\left(\frac{\xi_1 h}{2}\right) + \sin^2\left(\frac{\xi_2 h}{2}\right)\right)}{1 + 4r\sin^2\left(\frac{\xi_3 h}{2}\right)} g^{**} \end{aligned} \tag{6}$$

Substituting (4) into (5) and substituting the resulting expression into (6) gives the overall magnification factor for the ADI scheme:

$$g = \left[\frac{1 - 4r\left(\sin^2\left(\frac{\xi_1 h}{2}\right) + \sin^2\left(\frac{\xi_2 h}{2}\right)\right)}{1 + 4r\sin^2\left(\frac{\xi_3 h}{2}\right)}\right]\left[\frac{1 - 4r\left(\sin^2\left(\frac{\xi_1 h}{2}\right) + \sin^2\left(\frac{\xi_3 h}{2}\right)\right)}{1 + 4r\sin^2\left(\frac{\xi_2 h}{2}\right)}\right]\left[\frac{1 - 4r\left(\sin^2\left(\frac{\xi_2 h}{2}\right) + \sin^2\left(\frac{\xi_3 h}{2}\right)\right)}{1 + 4r\sin^2\left(\frac{\xi_1 h}{2}\right)}\right] \tag{7}$$

Letting $A \equiv \sin^2\left(\frac{\xi_1 h}{2}\right), B \equiv \sin^2\left(\frac{\xi_2 h}{2}\right), C \equiv \sin^2\left(\frac{\xi_1 h}{2}\right) = C$ in Eq. (7) results in

$$g(\xi_1, \xi_2, \xi_3) = \left(\frac{1 - 4r(A + B)}{1 + 4rC}\right)\left(\frac{1 - 4r(A + C)}{1 + 4rB}\right)\left(\frac{1 - 4r(B + C)}{1 + 4rA}\right) \tag{8}$$

The scheme is conditionally stable if $\left|g(\xi_1, \xi_2, \xi_3)\right| \leq 1$ for some value of r and $\left|g(\xi_1, \xi_2, \xi_3)\right| > 1$ for some other value of r (this is the same as using different values of Δt in place of r, since $r = \frac{D\Delta t}{3h^2}$ and h and D would be kept constant).

Now the scheme can be shown to be conditionally stable by letting $A = B = C = 1$ in Eq. (8) and then by finding one value of r which makes the magnification factor to become

less than one and then by looking for another value of r which makes the magnification factor to becomes larger than one.

Therefore, when $A = B = C = 1$, Eq. (8) becomes

$$\left|g(\xi_1, \xi_2, \xi_3)\right| = \left(\frac{1-8r}{1+4r}\right)\left(\frac{1-8r}{1+4r}\right)\left(\frac{1-8r}{1+4r}\right) \tag{8A}$$

Now, putting $r = 2$ in the above gives $\left|g(\xi_1, \xi_2, \xi_3)\right| = 2.744 > 1$ implying that the scheme is unstable.

Putting $r = 0.5$ in Eq. (8A) gives $\left|g(\xi_1, \xi_2, \xi_3)\right| = 0.125 < 1$ implying that the scheme is stable.

Hence the scheme is conditionally stable, because by fixing h and D, it was possible to find a time step Δt which made some mode become unstable. If one mode is unstable, the overall scheme is also unstable. This result shows that the above given ADI scheme for 3D is conditionally stable.

2.2 Part (b)

Expectation: Temporal accuracy is expected to be $O(\Delta t)$ since at each 1/3 time step there is one implicit step compared to two explicit steps. Starting from the main equations shown in part (a)

$$\overbrace{(I - rL_x)u^*}^{implicit\ \text{(backward Euler)}} = \overbrace{\left(I + rL_y + rL_z\right)}^{\text{explicit (2 forward Euler)}} u^n \tag{1}$$

$$\left(I - rL_y\right)u^{**} = (I + rL_x + rL_z)u^* \tag{2}$$

$$(I - rL_z)u^{n+1} = \left(I + rL_x + rL_y\right)u^{**} \tag{3}$$

There will be an $O(\Delta t)$ error resulting from the application of Euler approximation to each of the terms in each equation above. One of the implicit errors will cancel exactly with one of the errors from the explicit part of the equation (due to sign difference), leaving an extra $O(\Delta t)$ error after each third step. Hence at the completion of one full time step, the temporal error will be $3O(\Delta t)$ or $O(\Delta t)$.

Explanation: The derivation below follows the method explained in class for the 2D ADI case, but being applied to the 3D case. Starting by pre-multiplying Eq. (1) by the operator $(I + rL_x + rL_z)$ gives

$$(I + rL_x + rL_z)(I - rL_x)u^* = (I + rL_x + rL_z)\left(I + rL_y + rL_z\right)u^n$$

But since $(I + rL_x + rL_z)$ commutes[2] with $(I - rL_x)$, then the two terms in the LHS of the above equation can be interchanged giving

$$(I - rL_x)\ \overbrace{(I + rL_x + rL_z)u^*}^{\text{now replace this from (2)}} = (I + rL_x + rL_z)\left(I + rL_y + rL_z\right)u^n$$

Replacing the term marked above by its LHS value from Eq. (2) yields

$$(I - rL_x)\left(I - rL_y\right)u^{**} = (I + rL_x + rL_z)\left(I + rL_y + rL_z\right)u^n$$

Pre-multiplying the above by the operator $\left(I + rL_x + rL_y\right)$ gives

$$\left(I + rL_x + rL_y\right)(I - rL_x)\left(I - rL_y\right)u^{**} = \left(I + rL_x + rL_y\right)(I + rL_x + rL_z)\left(I + rL_y + rL_z\right)u^n$$

But since $\left(I + rL_x + rL_y\right)$ commutes with $(I - rL_x)\left(I - rL_y\right)$ the above can be written as

$$(I - rL_x)\left(I - rL_y\right)\overbrace{\left(I + rL_x + rL_y\right)u^{**}}^{\text{replace this from (3)}} = \left(I + rL_x + rL_y\right)(I + rL_x + rL_z)\left(I + rL_y + rL_z\right)u^n$$

[2]To show these operators commute, similar argument can be made as was done for the 2D case in class, which is by saying that each operator L_x, L_y, L_z on its own commutes with the other 2, hence the result will follow.

Replacing the term marked above by its LHS value from Eq. (3) gives

$$(I - rL_x)\Big(I - rL_y\Big)(I - rL_z)u^{n+1} = \Big(I + rL_x + rL_y\Big)(I + rL_x + rL_z)\Big(I + rL_y + rL_z\Big)u^n$$

Expanding all terms by multiplying all operators and simplifying the result and using $L = L_x + L_y + L_z$ gives the following

$$\begin{aligned}\Big(I - rL + r^2\Big(L_xL_z + L_yL_z\Big) + r^2L_xL_y - r^3L_xL_yL_z\Big)u^{n+1} = \\ \Big(I + rL + rL + 3r^2L_xL_y + 3r^2L_xL_z + 3r^2L_xL_y + 3r^2L_yL_z\Big)u^n + (H.O.T.)\end{aligned} \tag{4}$$

Where H.O.T. are terms from operators of order 2 and higher. These terms produce errors of order $O\Big(\Delta t^2\Big)$, $O\Big(\Delta t^3\Big)$ and higher. Moving all these terms to the RHS simplifies Eq. (4) to the following

$$\begin{aligned}(I - rL)u^{n+1} &= (I + rL + rL)u^n + O\Big(\Delta t^2\Big) + O\Big(\Delta t^3\Big) + \cdots \\ u^{n+1} - u^n &= rLu^{n+1} + 2rLu^n + O\Big(\Delta t^2\Big) + O\Big(\Delta t^3\Big) + \cdots\end{aligned}$$

Since $r = \frac{D\Delta t}{3}$ the above becomes

$$u^{n+1} - u^n = \frac{D\Delta t}{3}Lu^{n+1} + 2\frac{D\Delta t}{3}Lu^n + O\Big(\Delta t^2\Big) + O\Big(\Delta t^3\Big) + \cdots$$

Dividing the above equation by Δt gives

$$\frac{u^{n+1} - u^n}{\Delta t} = \frac{D}{3}Lu^{n+1} + \frac{2D}{3}Lu^n + O(\Delta t) + O\Big(\Delta t^2\Big) + \cdots$$

Now $\boxed{\text{adding } \frac{D}{6}Lu^{n+1} \text{ and subtracting } \frac{D}{6}Lu^{n+1}}$ and $\boxed{\text{subtracting } \frac{D}{6}Lu^{n} \text{ and adding } \frac{D}{6}Lu^{n}}$ from the RHS of the above equation gives

$$\begin{aligned}\frac{u^{n+1} - u^n}{\Delta t} &= \frac{D}{3}Lu^{n+1} + \frac{D}{6}Lu^{n+1} + \frac{2D}{3}Lu^n - \frac{D}{6}Lu^n + \frac{D}{6}Lu^n - \frac{D}{6}Lu^{n+1} + O(\Delta t) + O\Big(\Delta t^2\Big) + \cdots \\ \overbrace{\frac{u^{n+1} - u^n}{\Delta t}}^{C-N} &= \frac{1}{2}\Big(Lu^{n+1} + Lu^n\Big) + \frac{1}{6}\Big(Lu^n - Lu^{n+1}\Big) + O(\Delta t) + O\Big(\Delta t^2\Big) + \cdots\end{aligned}$$

The C-N scheme is known to be $O\Big(\Delta t^2 + h^2\Big)$. Multiplying the term $\frac{1}{6}\Big(Lu^n - Lu^{n+1}\Big)$ by $\frac{\Delta t}{\Delta t}$ in the above yields

$$u_t = \frac{1}{2}\Delta u + \overbrace{O\Big(\Delta t^2\Big) + O\Big(h^2\Big)}^{\text{from C-N part}} + \frac{\Delta t}{6}L\left(\frac{u^n - u^{n+1}}{\Delta t}\right) + O(\Delta t) + O\Big(\Delta t^2\Big) + \cdots$$

Taking the limits $\Delta t \to 0$ results in

$$\begin{aligned}u_t &= \frac{1}{2}\Delta u + \overbrace{O\Big(\Delta t^2\Big) + O\Big(h^2\Big)}^{\text{from C-N part}} + \overbrace{\frac{\Delta t}{6}\frac{\partial^7}{\partial^2 x \partial^2 y \partial^2 z \partial t} + O(\Delta t)}^{\text{still } O(\Delta t)} + O\Big(\Delta t^2\Big) + \cdots \\ &= \frac{1}{2}\Delta u + \overbrace{O\Big(\Delta t^2\Big) + O\Big(h^2\Big)}^{\text{from C-N part}} + O(\Delta t) + O\Big(\Delta t^2\Big) + \cdots \\ &= \frac{1}{2}\Delta u + \overbrace{O\Big(h^2\Big) + O(\Delta t)}^{} + O\Big(\Delta t^2\Big) + \cdots\end{aligned}$$

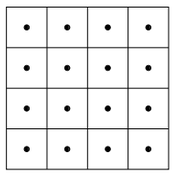
Since in the above, the dominant temporal error term is $O(\Delta t)$ the scheme is a first order in time accurate. It is also a second order in space accurate.

3 Problem 2

2. Consider

$$u_t = 0.1\,\Delta u \text{ on } \Omega = (0,1)\times(0,1)$$
$$\frac{\partial u}{\partial \vec{n}} = 0 \text{ on } \partial\Omega$$
$$u(x,y,0) = \exp\left(-10((x-0.4)^2 + (y-0.4)^2)\right)$$

(a) Write a program to solve this PDE using the Peaceman-Rachford ADI scheme on a cell-centered grid. Use a direct solver for the tridiagonal systems. In a cell-centered discretization the solution is stored at the grid points $(x_i, y_j) = (h(i-0.5), h(j-0.5))$ for $i, j = 1 \ldots N$ and $h = 1/N$. This discretization is natural for handling Neumann boundary conditions, and it is often used to discretize conservation laws. At the grid points adjacent to the boundary, the one-dimensional discrete Laplacian for homogeneous Neumann boundary conditions is



$$u_{xx}(x_1) \approx \frac{-u_1 + u_2}{h^2}.$$

(b) Perform a refinement study to show that your numerical solution is second-order accurate in space and time (refine time and space simultaneously using $\Delta t = h$) at time $t = 1$.

(c) Show that the spatial integral of the solution to the PDE does not change in time. That is

$$\frac{d}{dt}\int_\Omega u\,dV = 0.$$

(d) Show that the solution to the discrete equations satisfies the discrete conservation property

$$\sum_{i,j} u^n_{i,j} = \sum_{i,j} u^0_{i,j}$$

for all n. Demonstrate this property with your code.

Figure 2: Problem description

The following diagram shows the discretization using cell-centered scheme for the case of $N = 4$. The center of the cells moves closer to the physical boundaries of the unit square as N becomes larger.

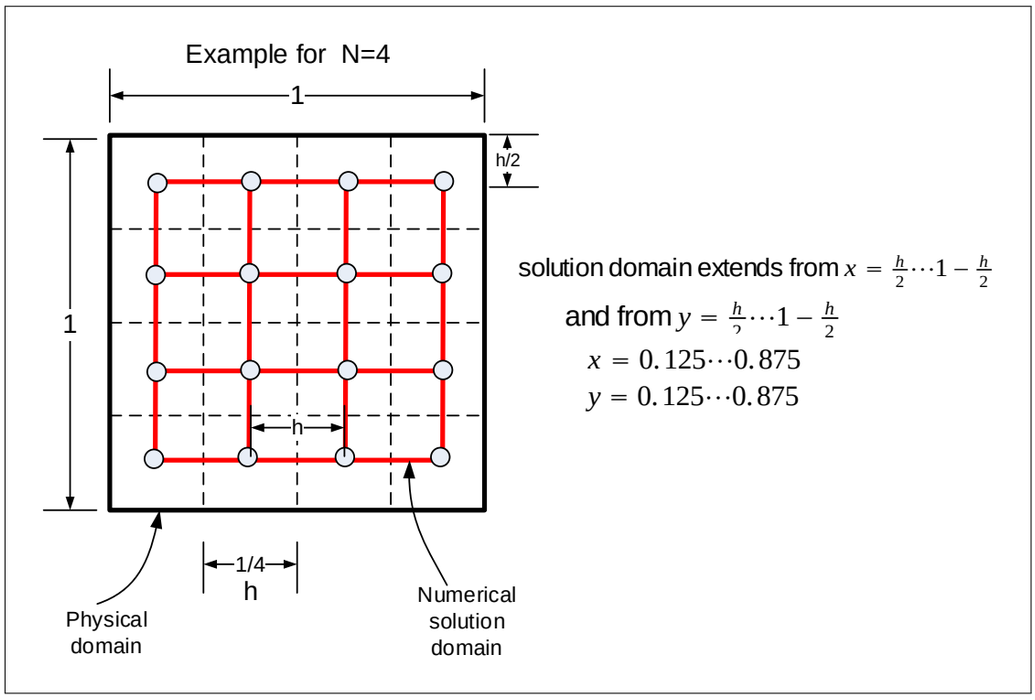


Figure 3: Grid used

The physical domain is always the unit square $x = 0 \cdots 1$, $y = 0 \cdots 1$, but the discrete solution domain is the one at corners of the red grid above. A small example is used below to help determine the layout of the operator used in the direct solver. The 2D ADI scheme for the diffusion problem is

$$\left(I - \frac{D\Delta t}{2} L_x\right)\mathbf{u}^* = \left(I + \frac{D\Delta t}{2} L_y\right)\mathbf{u}^n \qquad (1A)$$
$$\left(I - \frac{D\Delta t}{2} L_y\right)\mathbf{u}^{n+1} = \left(I + \frac{D\Delta t}{2} L_x\right)\mathbf{u}^*$$

Where $L_x = L_y = \frac{1}{h^2}\begin{bmatrix} -1 & 1 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & 0 \\ \cdots & \cdots & \cdots & \ddots & \cdots & \cdots \\ 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix}$ for the case of homogenous Neumann boundary conditions. The solution given below uses an overall L operator which is used by the direct solver. Another approach would have been to use the above L_x, L_y operator, and iterate over each each row and column applying the direct solver each time.

The following derives the overall L operator used. Eq. (1A) can be written as

$$u^*_{ij} - \frac{D\Delta t}{2}\frac{u^*_{i-1,j} - 2u^*_{ij} + u^*_{i+1,j}}{h^2} = u^n_{ij} + \frac{D\Delta t}{2}\frac{u^n_{i,j-1} - 2u^n_{ij} + u^n_{i,j+1}}{h^2}$$
$$u^{n+1}_{ij} - \frac{D\Delta t}{2}\frac{u^{n+1}_{i,j-1} - 2u^{n+1}_{ij} + u^{n+1}_{i,j+1}}{h^2} = u^{n*}_{ij} + \frac{D\Delta t}{2}\frac{u^*_{i-1,j} - 2u^*_{ij} + u^*_{i+1,j}}{h^2}$$

letting $r = \frac{D\Delta t}{2h^2}$ and simplifying the above gives

$$u^*_{ij}(1+2r) - ru^*_{i-1,j} - ru^*_{i+1,j} = u^n_{ij}(1-2r) + ru^n_{i,j-1} + ru^n_{i,j+1} \qquad (1)$$
$$u^{n+1}_{ij}(1+2r) - ru^{n+1}_{i,j-1} - ru^{n+1}_{i,j+1} = u^*_{ij}(1-2r) + ru^*_{i-1,j} + ru^*_{i+1,j} \qquad (2)$$

The above finite difference equations are applied at all the grid points, except for those for the rows and columns at the boundaries. In order to determine the equations to use for the boundary grid points, the approximation $L_x \approx \frac{-u_1+u_2}{h^2}$ is used. Similar one is used for L_y. The result of using the above approximation is the following finite difference equations

used for the boundary grid points

$$u^*_{ij} - \frac{D\Delta t}{2}\frac{-u^*_{ij} + u^*_{i+1,j}}{h^2} = u^n_{ij} + \frac{D\Delta t}{2}\frac{-u^n_{ij} + u^n_{i,j+1}}{h^2}$$
$$u^{n+1}_{ij} - \frac{D\Delta t}{2}\frac{-u^{n+1}_{ij} + u^{n+1}_{i,j+1}}{h^2} = u^{n*}_{ij} + \frac{D\Delta t}{2}\frac{-u^*_{ij} + u^*_{i+1,j}}{h^2}$$

Simplifying the above gives

$$u^*_{ij}(1+r) - ru^*_{i+1,j} = u^n_{ij}(1-r) + ru^n_{i,j+1} \tag{1A}$$
$$u^{n+1}_{ij}(1+r) - ru^{n+1}_{i,j+1} = u^*_{ij}(1-r) + ru^*_{i+1,j} \tag{2A}$$

To help obtain L, a small example is used to help see the structure of the matrix. This small example exhibits all the needed information to generate the pattern for L from. Using $n_x = n_y = 4$, the grid becomes

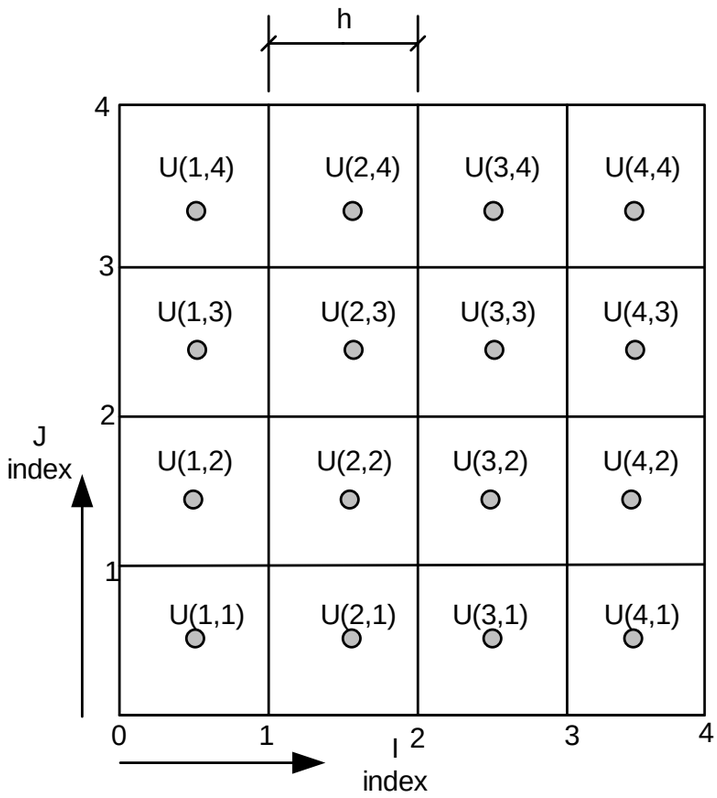


Figure 4: Updated Grid

Eq. (1) and (1A) are now written for all the nodes resulting in the following 16 equations

$$
\begin{aligned}
u^*_{11}(1+r) - ru^*_{21} &= u^n_{11}(1-r) + ru^n_{12} \\
u^*_{21}(1+2r) - ru^*_{1,1} - ru^*_{3,1} &= u^n_{21}(1-r) + ru^n_{22} \\
u^*_{31}(1+2r) - ru^*_{2,1} - ru^*_{4,1} &= u^n_{31}(1-r) + ru^n_{32} \\
u^*_{41}(1+r) - ru^*_{31} &= u^n_{41}(1-r) + ru^n_{42} \\
u^*_{12}(1+r) - ru^*_{22} &= u^n_{12}(1-2r) + ru^n_{1,1} + ru^n_{1,3} \\
u^*_{22}(1+2r) - ru^*_{1,2} - ru^*_{3,2} &= u^n_{22}(1-2r) + ru^n_{2,1} + ru^n_{2,3} \\
u^*_{32}(1+2r) - ru^*_{2,2} - ru^*_{4,2} &= u^n_{32}(1-2r) + ru^n_{3,1} + ru^n_{3,3} \\
u^*_{42}(1+r) - ru^*_{32} &= u^n_{42}(1-2r) + ru^n_{4,1} + ru^n_{4,3} \\
u^*_{13}(1+r) - ru^*_{23} &= u^n_{13}(1-2r) + ru^n_{1,2} + ru^n_{1,4} \\
u^*_{23}(1+2r) - ru^*_{1,3} - ru^*_{3,3} &= u^n_{23}(1-2r) + ru^n_{2,2} + ru^n_{2,4} \\
u^*_{33}(1+2r) - ru^*_{2,3} - ru^*_{4,3} &= u^n_{33}(1-2r) + ru^n_{3,2} + ru^n_{3,4} \\
u^*_{43}(1+r) - ru^*_{33} &= u^n_{43}(1-2r) + ru^n_{4,2} + ru^n_{4,4} \\
u^*_{14}(1+r) - ru^*_{24} &= u^n_{14}(1-r) + ru^n_{13} \\
u^*_{24}(1+2r) - ru^*_{1,4} - ru^*_{3,4} &= u^n_{24}(1-r) + ru^n_{23} \\
u^*_{34}(1+2r) - ru^*_{2,4} - ru^*_{4,4} &= u^n_{34}(1-r) + ru^n_{33} \\
u^*_{44}(1+r) - ru^*_{34} &= u^n_{44}(1-r) + ru^n_{43}
\end{aligned}
$$

In matrix form, the above gives $Au^* = b$ which is then used to solve for u^*. The matrix A is now written out. To save space and to allow the matrix to fit on the page, the following terms are used

$$
\begin{aligned}
r &= \frac{D\Delta t}{2h^2} \\
\alpha &\equiv 1 + r \\
\beta &\equiv 1 + 2r \\
\gamma &\equiv 1 - r \\
\theta &\equiv 1 - 2r
\end{aligned}
$$

$$
\overbrace{\begin{bmatrix}
\alpha & -r & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-r & \beta & -r & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -r & \beta & -r & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -r & \alpha & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \alpha & -r & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -r & \beta & -r & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -r & \beta & -r & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -r & \alpha & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha & -r & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -r & \beta & -r & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -r & \beta & -r & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -r & \alpha & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha & -r & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -r & \beta & -r & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -r & \beta & -r \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -r & \alpha
\end{bmatrix}}^{\mathbf{A}}
\overbrace{\begin{bmatrix} u^*_{11} \\ u^*_{21} \\ u^*_{31} \\ u^*_{41} \\ u^*_{12} \\ u^*_{22} \\ u^*_{32} \\ u^*_{4,2} \\ u^*_{13} \\ u^*_{2,3} \\ u^*_{3,3} \\ u^*_{4,3} \\ u^*_{14} \\ u^*_{2,4} \\ u^*_{3,4} \\ u^*_{4,4} \end{bmatrix}}^{\mathbf{x}}
=
\overbrace{\begin{bmatrix}
\gamma & 0 & 0 & 0 & r & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \gamma & 0 & 0 & 0 & r & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \gamma & 0 & 0 & 0 & r & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \gamma & 0 & 0 & 0 & r & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
r & 0 & 0 & 0 & \theta & 0 & 0 & 0 & r & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & r & 0 & 0 & 0 & \theta & 0 & 0 & 0 & r & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & r & 0 & 0 & 0 & \theta & 0 & 0 & 0 & r & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & r & 0 & 0 & 0 & \theta & 0 & 0 & 0 & r & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & r & 0 & 0 & 0 & \theta & 0 & 0 & 0 & r & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & r & 0 & 0 & 0 & \theta & 0 & 0 & 0 & r & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & r & 0 & 0 & 0 & \theta & 0 & 0 & 0 & r & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & r & 0 & 0 & 0 & \theta & 0 & 0 & 0 & r \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r & 0 & 0 & 0 & \gamma & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r & 0 & 0 & 0 & \gamma & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r & 0 & 0 & 0 & \gamma & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r & 0 & 0 & 0 & \gamma
\end{bmatrix}}^{\mathbf{b}}
\begin{bmatrix} u^n_{11} \\ u^n_{21} \\ u^n_{31} \\ u^n_{41} \\ u^n_{12} \\ u^n_{22} \\ u^n_{32} \\ u^n_{4,2} \\ u^n_{13} \\ u^n_{2,3} \\ u^n_{3,3} \\ u^n_{4,3} \\ u^n_{14} \\ u^n_{2,4} \\ u^n_{3,4} \\ u^n_{4,4} \end{bmatrix}
\tag{7}
$$

A sparse direct solver can now be used to solve for u^*.

Starting the second ADI step to find u^{n+1}, the process is similar to the one shown above, but the equations are written column-wise instead of row-wise as was the case earlier. For non-boundary grid points the following equation is used

$$u^{n+1}_{ij}(1+2r) - ru^{n+1}_{i,j-1} - ru^{n+1}_{i,j+1} = u^*_{ij}(1-2r) + ru^*_{i-1,j} + ru^*_{i+1,j}$$

And for the boundary grid points the following equation is used

$$u^{n+1}_{ij}(1+r) - ru^{n+1}_{i,j+1} = u^*_{ij}(1-r) + ru^*_{i+1,j}$$

Applying the above to each grid point results in the the following 16 equations

$$
\begin{aligned}
u_{11}^{n+1}(1+r) - ru_{12}^{n+1} &= u_{11}^{*}(1-r) + ru_{21}^{*} \\
u_{21}^{n+1}(1+r) - ru_{22}^{n+1} &= u_{21}^{*}(1-2r) + ru_{1,1}^{n} + ru_{3,1}^{n} \\
u_{31}^{n+1}(1+r) - ru_{32}^{n+1} &= u_{31}^{*}(1-2r) + ru_{2,1}^{n} + ru_{4,1}^{n} \\
u_{41}^{n+1}(1+r) - ru_{42}^{n+1} &= u_{41}^{*}(1-r) + ru_{31}^{*} \\
u_{12}^{n+1}(1+2r) - ru_{1,1}^{n+1} - ru_{1,3}^{n+1} &= u_{12}^{*}(1-r) + ru_{22}^{*} \\
u_{22}^{n+1}(1+2r) - ru_{2,1}^{n+1} - ru_{2,3}^{n+1} &= u_{22}^{*}(1-2r) + ru_{1,2}^{n} + ru_{3,2}^{n} \\
u_{32}^{*}(1+2r) - ru_{3,1}^{*} - ru_{3,3}^{*} &= u_{32}^{n}(1-2r) + ru_{2,2}^{n} + ru_{4,2}^{n} \\
u_{42}^{*}(1+2r) - ru_{4,1}^{*} - ru_{4,3}^{*} &= u_{42}^{n}(1-r) + ru_{32}^{n} \\
u_{13}^{*}(1+2r) - ru_{1,2}^{*} - ru_{1,4}^{*} &= u_{13}^{n}(1-r) + ru_{23}^{n} \\
u_{23}^{*}(1+2r) - ru_{2,2}^{*} - ru_{2,4}^{*} &= u_{23}^{n}(1-2r) + ru_{1,3}^{n} + ru_{3,3}^{n} \\
u_{33}^{*}(1+2r) - ru_{3,2}^{*} - ru_{3,4}^{*} &= u_{33}^{n}(1-2r) + ru_{2,3}^{n} + ru_{4,3}^{n} \\
u_{43}^{*}(1+2r) - ru_{4,2}^{*} - ru_{4,4}^{*} &= u_{43}^{n}(1-r) + ru_{33}^{n} \\
u_{14}^{*}(1+r) - ru_{13}^{*} &= u_{14}^{n}(1-r) + ru_{24}^{n} \\
u_{24}^{*}(1+r) - ru_{23}^{*} &= u_{24}^{n}(1-2r) + ru_{1,4}^{n} + ru_{3,4}^{n} \\
u_{34}^{*}(1+r) - ru_{33}^{*} &= u_{34}^{n}(1-2r) + ru_{2,4}^{n} + ru_{4,4}^{n} \\
u_{44}^{*}(1+r) - ru_{43}^{*} &= u_{44}^{n}(1-r) + ru_{34}^{n}
\end{aligned}
$$

The above equations are now written as $Au = b$ but the unknowns are listed column-wise

in order to keep the tridiagonal form. The resulting matrix A is the following

$$\overbrace{\begin{bmatrix}
\alpha & -r & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-r & \beta & -r & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -r & \beta & -r & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -r & \alpha & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \alpha & -r & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -r & \beta & -r & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -r & \beta & -r & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -r & \alpha & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha & -r & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -r & \beta & -r & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -r & \beta & -r & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -r & \alpha & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha & -r & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -r & \beta & -r & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -r & \beta & -r \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -r & \alpha
\end{bmatrix}}^{\mathbf{A}}
\overbrace{\begin{bmatrix} u_{11}^{n+1} \\ u_{12}^{n+1} \\ u_{13}^{n+1} \\ u_{14}^{n+1} \\ u_{21}^{n+1} \\ u_{22}^{n+1} \\ u_{23}^{n+1} \\ u_{2,4}^{n+1} \\ u_{31}^{n+1} \\ u_{32}^{n+1} \\ u_{3,3}^{n+1} \\ u_{3,4}^{n+1} \\ u_{41}^{n+1} \\ u_{4,2}^{n+1} \\ u_{4,3}^{n+1} \\ u_{4,4}^{n+1} \end{bmatrix}}^{\mathbf{x}} =$$

$$\overbrace{\begin{bmatrix}
\gamma & 0 & 0 & 0 & r & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \gamma & 0 & 0 & 0 & r & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \gamma & 0 & 0 & 0 & r & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \gamma & 0 & 0 & 0 & r & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
r & 0 & 0 & 0 & \theta & 0 & 0 & 0 & r & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & r & 0 & 0 & 0 & \theta & 0 & 0 & 0 & r & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & r & 0 & 0 & 0 & \theta & 0 & 0 & 0 & r & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & r & 0 & 0 & 0 & \theta & 0 & 0 & 0 & r & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & r & 0 & 0 & 0 & \theta & 0 & 0 & 0 & r & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & r & 0 & 0 & 0 & \theta & 0 & 0 & 0 & r & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & r & 0 & 0 & 0 & \theta & 0 & 0 & 0 & r & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & r & 0 & 0 & 0 & \theta & 0 & 0 & 0 & r \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r & 0 & 0 & 0 & \gamma & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r & 0 & 0 & 0 & \gamma & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r & 0 & 0 & 0 & \gamma & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r & 0 & 0 & 0 & \gamma
\end{bmatrix}}^{\mathbf{b}}
\begin{bmatrix} u_{11}^{*} \\ u_{12}^{*} \\ u_{13}^{*} \\ u_{14}^{*} \\ u_{21}^{*} \\ u_{22}^{*} \\ u_{23}^{*} \\ u_{2,4}^{*} \\ u_{31}^{*} \\ u_{32}^{*} \\ u_{3,3}^{*} \\ u_{3,4}^{*} \\ u_{41}^{*} \\ u_{4,2}^{*} \\ u_{4,3}^{*} \\ u_{4,4}^{*} \end{bmatrix} \tag{7}$$

Now u^{n+1} is solved for using a direct sparse solver. The above 2 steps are repeated for each one time step. One can see that the A matrix is the same for both solving $Au^{*} = b$ and $Au^{n+1} = b$. Therefore, in the implementation only one A and one B matrix was allocated initially and used for solving for u^{*} and u^{n+1}. Both matrices (A and B) are created as sparse matrices to save storage. The A matrix represent the implicit part of the scheme, while the B matrix for the explicit part.

Since the edges of the domain are insulated, no concentration will diffuse to the outside. Therefore the result of diffusion will be that the concentration will diffuse internally and will spread out. Therefore, at steady state as $t \to \infty$ the solution is known and given by

$$u\left(x, y, \infty\right) = \int_{h/2}^{1-\frac{h}{2}} \int_{h/2}^{1-\frac{h}{2}} u\left(x, y, 0\right) dx dy$$

The following plot shows the solution at $t = 1$ second with the steady state solution displayed as the blue horizontal flat surface superimposed on the same plot. The steady state solution is what would result if run time was made to be very long.

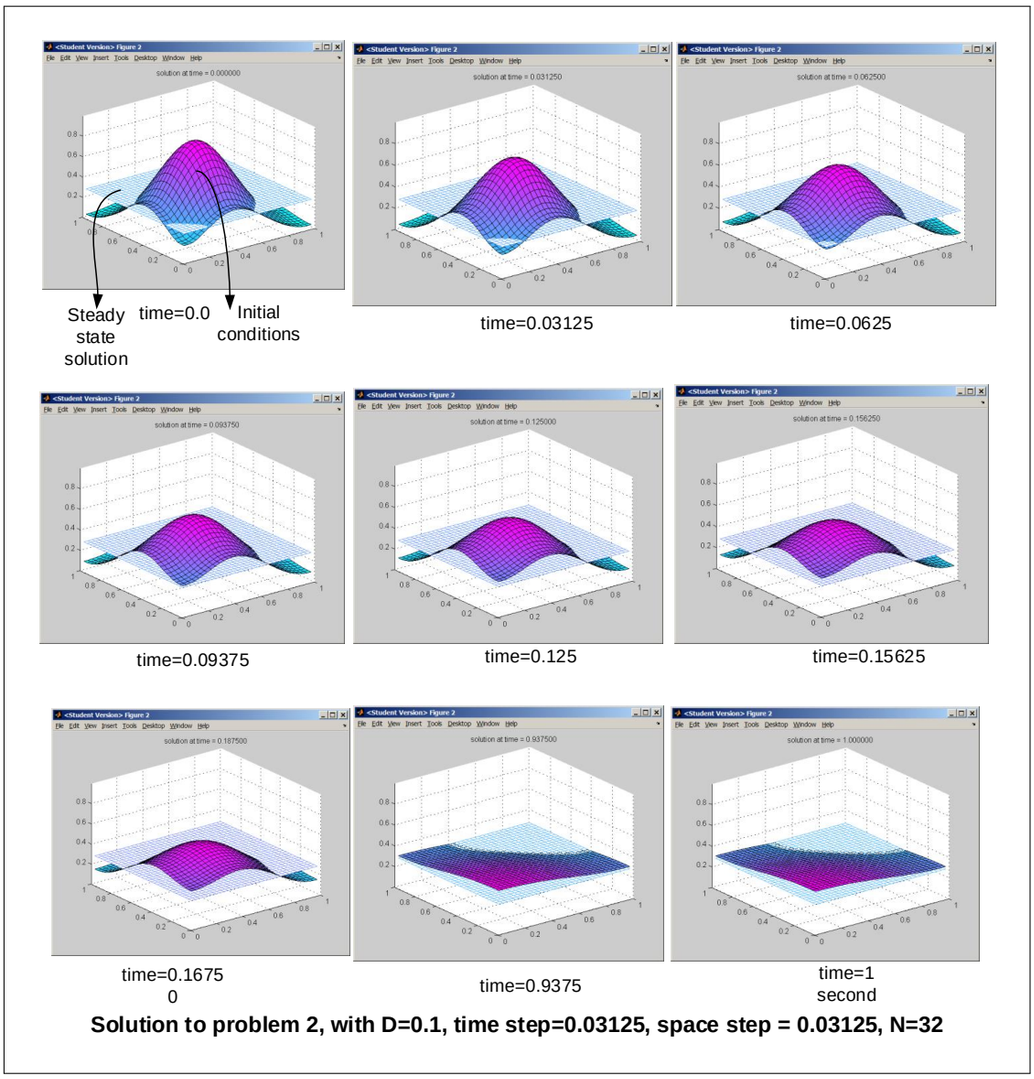


Figure 5: steady state solution

To verify that the numerical solution converges to the steady state solution, the plot below was generate which represents the solution of the above problem taken at $t = 4$ seconds. The gap in the diagram below is the difference between the steady state solution and the solution at $t = 4$ seconds. This gap became smaller the longer the time to run is made (keeping everything else fixed).

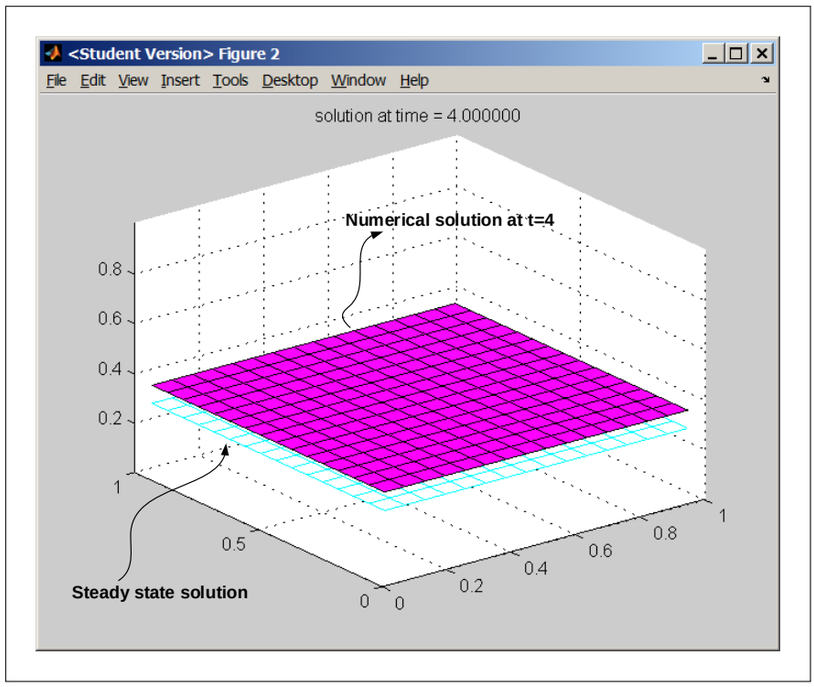


Figure 6: steady state solution

3.1 Part(b)

Refinement study was carried out to show that the 2D ADI scheme is a second order accurate in time and in space. The method used successive errors between numerical solutions. The algorithm of the refinement study is given below at the end of this part of the problem.

Recalling that In HW1, the spatial grid was divided by half each time. However, in this problem, since cell centered grid is used, h and Δt were divided by 3 each time. This was done so that the new grid will contain some grid locations that are still aligned in the same physical location as the previous step. The error between both solutions is obtained by taking the difference of only these points that are aligned. These points will be the grid point of the coarse grid. The following diagram illustrate this for the case of $n = 3$ and $n = 9$

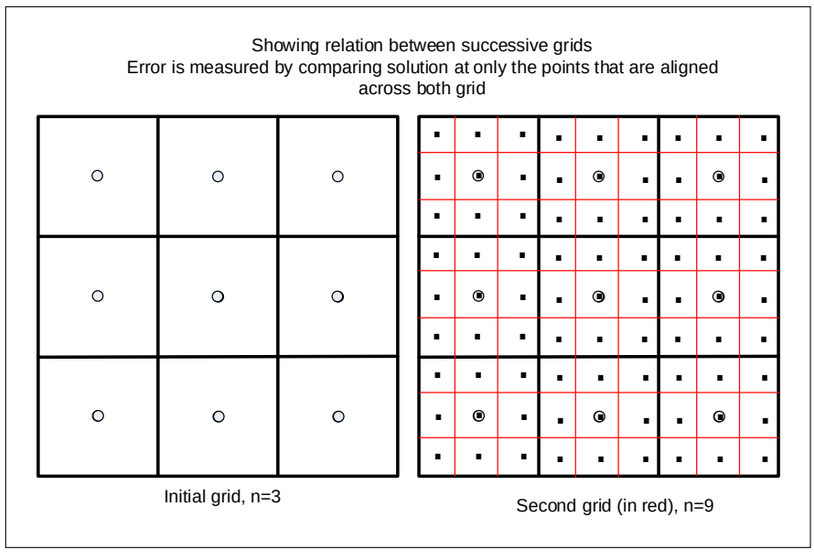


Figure 7: case of $n = 3$ and $n = 9$

The result of the refinement study shows second order accuracy as the error ratio came out to be 9.

Below is the result obtained. In addition to the ratio table, it can be seen that the slope of the line in the log plot is 2, implying the scheme is second order accurate.

Refinement study result, HW 2, problem 2, showing second order accuracy

#	N	delt	h	|u|	mean(u)	|e|	ratio	|u steady|
2	9	0.11111	0.111111	0.293146	0.290318	1.19490e-002	**1.000000**	0.289080
3	27	0.03704	0.037037	0.293146	0.289369	1.23779e-003	**9.653448**	0.289080
4	81	0.01235	0.012346	0.293146	0.289262	1.36668e-004	**9.056912**	0.289080
5	243	0.00412	0.004115	0.293146	0.289251	1.51752e-005	**9.006018**	0.289080
6	729	0.00137	0.001372	0.293146	0.289249	1.68601e-006	**9.000673**	0.289080

EDU>>

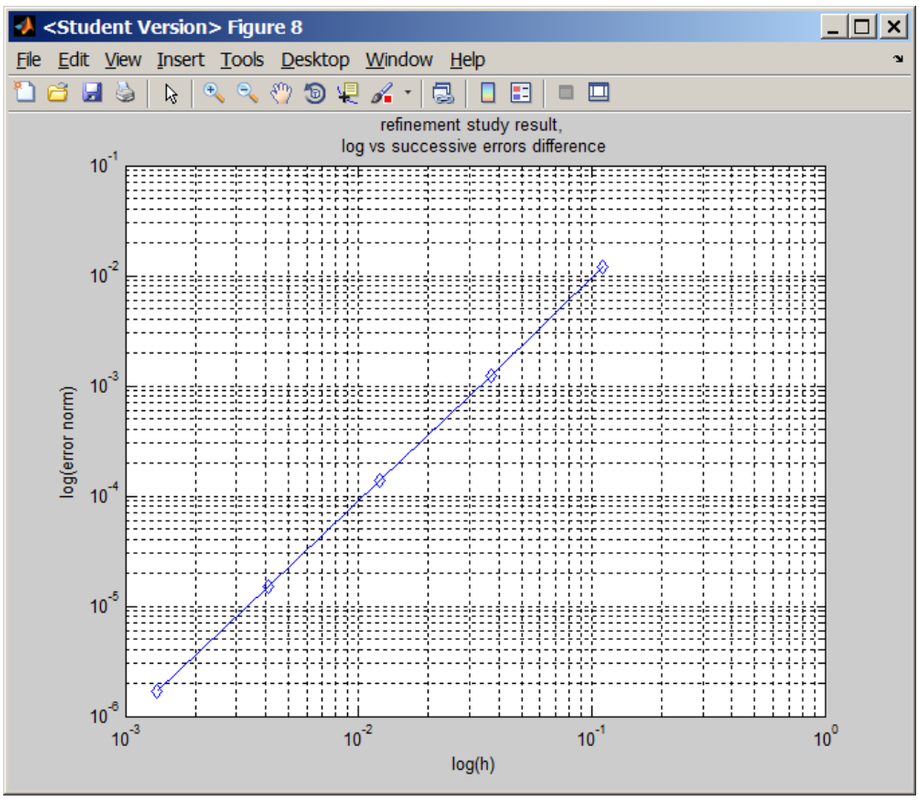


Figure 8: refinement study plot

3.2 Refinement algorithm

The following is the general outline of the algorithm used in the refinement study. The important part was to make sure when finding the error between the current and last solution, is to use the same physical locations that are aligned between both grids, and to use the coarse grid spacing when determining the grid norm of the error grid.

```
last_error   = 0
h            = 1/3
last_h       = h
delt         = h
last_u       = Solve_2D_ADI(h,delt)

LOOP
      h    = h/3
      delt = h

      current_u = Solve_2D_ADI(h,delt)

      -- now extract from current_u only locations that aligned with last_u grid
      current_u_mapped = extracted_u(last_u)

      error = last_h *  norm(last_u - current_u_mapped,2)
```

```
      ratio = last_error/error

      last_h = h
      last_error = error

      loop_counter++

      IF loop_counter > some_maximum THEN -- normally 5,6 iterations is enough
         EXIT LOOP
      END IF
END LOOP
```

3.3 Part(c)

The spatial integral represents the total concentration in the domain. Since the boundary are insulated, matter will only diffuse internally and no loss will occur to the outside. Hence, from the conservation of mass principle, initial concentration will remain the same, but will spread out to the mean in space. Therefore, it is known physically total concentration will not change with time

$$\frac{d}{dt}\int_{\Omega} u\big(x,y,t\big)dA = 0$$

The problem is asking to show this mathematically.

Since $u_t = D\Delta u$, then

$$\begin{aligned} I &= \frac{d}{dt}\int_{\Omega} u\big(x,y,t\big)dA \\ &= D\int_{\Omega} \Delta u \; dA \end{aligned}$$

But $\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$, hence

$$\left(\frac{1}{D}\right)I = \int_{y=0}^{1}\int_{x=0}^{1} \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \; dxdy$$

To show that $I = 0$, the above is written as

$$\begin{aligned} \left(\frac{1}{D}\right)I &= \int_{y=0}^{1}\int_{x=0}^{1} \frac{\partial^2 u}{\partial x^2}dxdy + \int_{y=0}^{1}\int_{x=0}^{1} \frac{\partial^2 u}{\partial y^2} \; dxdy \\ &= \int_{y=0}^{1}\left(\int_{x=0}^{1} \frac{\partial^2 u}{\partial x^2}dx\right)dy + \int_{x=0}^{1}\left(\int_{y=0}^{1} \frac{\partial^2 u}{\partial y^2} \; dy\right)dx \end{aligned} \tag{1}$$

By applying the fundamental theory of calculus (or using integration by parts) results in

$$\int_{x=0}^{1} \frac{\partial^2 u}{\partial x^2}dx = \left.\frac{\partial u}{\partial x}\right|_{x=1} - \left.\frac{\partial u}{\partial x}\right|_{x=0}$$

However $\left.\frac{\partial u}{\partial x}\right|_{x=1}$ is the normal derivative at the right boundary, and $\left.\frac{\partial u}{\partial x}\right|_{x=0}$ is the normal derivative at the left boundaries. These are both zero due to the homogenous Neumann boundary conditions given in the problem statement. therefore

$$\int_{x=0}^{1} \frac{\partial^2 u}{\partial x^2}dx = 0 \tag{2}$$

Similar argument shows that

$$\int_{y=0}^{1} \frac{\partial^2 u}{\partial y^2} \; dy = 0 \tag{3}$$

Substituting Eqs. (2) and (3) into (1) gives

$$\left(\frac{1}{D}\right)I = 0$$

Therefore

$$\frac{d}{dt}\int_{\Omega} u\left(x,y,t\right)dA = 0$$

3.4 Part(d)

The finite difference equations for the 2D ADI scheme is given by

$$\left(I - \frac{D\Delta t}{2}L_x\right)\mathbf{u}^* = \left(I + \frac{D\Delta t}{2}L_y\right)\mathbf{u}^n \tag{1}$$

$$\left(I - \frac{D\Delta t}{2}L_y\right)\mathbf{u}^{n+1} = \left(I + \frac{D\Delta t}{2}L_x\right)\mathbf{u}^* \tag{2}$$

Summing the equations over all entries in the 2D solution domain gives

$$\sum_i\sum_j\left(I - \frac{D\Delta t}{2}L_x\right)u^* = \sum_j\sum_i\left(I + \frac{D\Delta t}{2}L_y\right)u^n \tag{1A}$$

$$\sum_j\sum_i\left(I - \frac{D\Delta t}{2}L_y\right)u^{n+1} = \sum_i\sum_j\left(I + \frac{D\Delta t}{2}L_x\right)u^* \tag{2A}$$

In the above i represents the row number and j represents the column number of the solution grid u. The above two equations can be rewritten as

$$\sum_i\sum_j u_{ij}^* - \frac{D\Delta t}{2}\sum_i\sum_j L_x u_{ij}^* = \sum_j\sum_i u_{ij}^n + \frac{D\Delta t}{2}\sum_j\sum_i L_y u_{ij}^n \tag{1B}$$

$$\sum_j\sum_i u_{ij}^{n+1} - \frac{D\Delta t}{2}\sum_j\sum_i L_y u_{ij}^{n+1} = \sum_i\sum_j u_{ij}^* + \frac{D\Delta t}{2}\sum_i\sum_j L_x u_{ij}^* \tag{2B}$$

Looking at the term $\sum_i\sum_j L_x u_{ij}^*$ from Eq. (1B) and rewriting this as follows

$$\sum_i\sum_j L_x u_{ij}^* = \sum_i\left(\sum_j L_x u_{ij}^*\right)$$

$$= \sum_i \overbrace{\left(\sum_j L_x u_{ij}^*\right)}^{L_x \text{ operator applied to } i^{th}\text{row}}$$

In other words, $\sum_j L_x u_{ij}^*$ is the result of applying L_x to each entry in the i^{th} row, then summing the result.

Therefore, L_x is applied to entry $u^*(i,1)$ then to entry $u^*(i,2)$ and so on, until the last entry in the row which is $u^*(i,n)$.

How to find the result of applying L_x on each row? Given that L_x for 1D with homogenous Neumann boundary conditions is

$$L_x = \frac{1}{h^2}\begin{bmatrix} -1 & 1 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & 0 \\ \cdots & \cdots & \cdots & \ddots & \cdots & \cdots \\ 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix}$$

Then applying the operator to each entry in the i^{th} row gives the following

$$
\begin{array}{cccccccccccc}
j=1 & j=2 & j=3 & 4 & 5 & 6 & 7 & \cdots & n-2 & n-1 & j=n \\
-u_{i1} & +u_{i2} & & & & & & & & & \\
+u_{i1} & -2u_{i2} & +u_{i3} & & & & & & & & \\
 & +u_{i2} & -2u_{i3} & +u_{i4} & & & & & & & \\
 & & +u_{i3} & -2u_{i4} & +u_{i5} & & & & & & \\
 & & & +u_{i4} & -2u_{i5} & +u_{i6} & & & & & \\
 & & & & +u_{i5} & -2u_{i6} & +u_{i7} & & & & \\
 & & & & & & \ddots & & & & \\
 & & & & & & & +u_{i,n-3} & -2u_{i,n-2} & +u_{i,n-1} & \\
 & & & & & & & & +u_{i,n-2} & -2u_{i,n-1} & +u_{i,n} \\
 & & & & & & & & & +u_{i,n-1} & -u_{i,n}
\end{array}
$$

$$\left(\sum_j L_x u_{ij}^*\right) =$$

In the above, L_x was applied directly on the i^{th} row. The first line above shows the column index j which goes from $1 \cdots n$. The following diagram is made to help illustrate the above process, showing how L_x and L_y are applied to the solution in the u matrix.

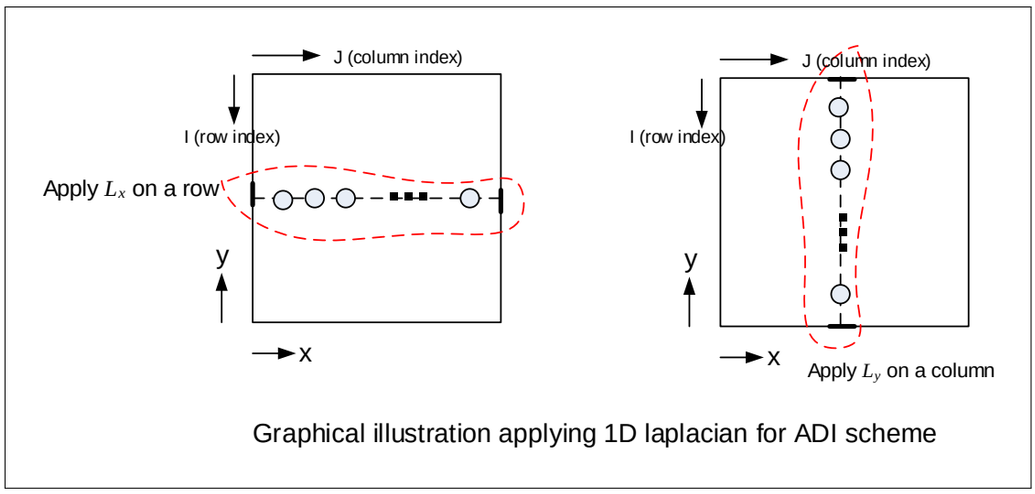


Figure 9: Illustrating the above process

One can see now that thesum is zero due to terms cancellation. The sum is zero in this case due to the homogenous Neumann boundary conditions which caused the first and last entries to cancel out.

Using the same procedure, then applying L_y to each column of u^n will also result in zero sum, since the north and south boundaries also have homogenous Neumann boundary conditions. Since boundary conditions do not change going from u^* to u^{n+1}, the same result is obtained when applying L_y operator to each column of u^{n+1}. From the above discussion, it is found that

$$\sum_i \sum_j L_x u_{ij}^* = 0$$

$$\sum_j \sum_i L_y u_{ij}^n = 0$$

$$\sum_j \sum_i L_y u_{ij}^{n+1} = 0$$

$$\sum_i \sum_j L_x u_{ij}^* = 0$$

Substituting the above 4 equations back into Eqs. (1B),(2B) gives

$$\sum_i \sum_j u_{ij}^* = \sum_j \sum_i u_{ij}^n \tag{1C}$$

$$\sum_j \sum_i u_{ij}^{n+1} = \sum_i \sum_j u_{ij}^* \tag{2C}$$

Substituting Eq. (1C) into (2C) gives

$$\sum_i j u_i j^n + 1 = \sum_i j u_i j^n$$

Since the above is valid for any n (boundary conditions do not change with time) then setting $n = 0$ in the above results in

$$\sum_{ij} u_{ij}^1 = \sum_{ij} u_{ij}^0$$

Similarly, setting $n = 1$ results in

$$\sum_{ij} u_{ij}^2 = \sum_{ij} u_{ij}^1$$

and so on all the way any n value. Hence in general the following result is obtained

$$\sum_{ij} u_{ij}^n = \sum_{ij} u_{ij}^{n-1}$$

By repeated back substitution on the RHS, the following is obtained

$$\sum_{ij} u_{ij}^n = \sum_{ij} u_{ij}^0$$

Therefore, the discrete conservation property is satisfied.

Verification in code To verify part(d) in the code, a table was generated during one run, where $\sum_{ij} u_{ij}^n$ was calculated at the end of each time step using the Matlab command sum(sum(u)), and this value was printed at each time step. The result shows that this value is constant implying the discrete conservation property is satisfied. Here is the result below

```
current_time   sum(U(current_time)
  0.00000      1897.85094
  0.01235      1897.85094
  0.02469      1897.85094
  0.03704      1897.85094
  0.04938      1897.85094
  0.06173      1897.85094
  0.07407      1897.85094
  0.08642      1897.85094
  0.09877      1897.85094
  0.11111      1897.85094
....
  0.92593      1897.85094
  0.93827      1897.85094
  0.95062      1897.85094
  0.96296      1897.85094
  0.97531      1897.85094
  0.98765      1897.85094
```

4 Problem 3

for all n. Demonstrate this property with your code.

3. The FitzHugh-Nagumo equations

$$\frac{\partial v}{\partial t} = D\Delta v + (a - v)(v - 1)v - w + I$$
$$\frac{\partial w}{\partial t} = \epsilon(v - \gamma w).$$

are used in electrophysiology to model the cross membrane electrical potential (voltage) in cardiac tissue and in neurons. Assuming that the spatial coupling is local and passive results the term which looks like the diffusion of voltage. The state variables are the voltage v and the recovery variable w.

(a) Write a program to solve the FitzHugh-Nagumo equations on the unit square with homogeneous Neumann boundary conditions for v (meaning electrically insulated). Use a fractional step method to handle the diffusion and reactions separately. Use an ADI method for the diffusion solve. Describe what ODE solver you used for the reactions and what fractional stepping your chose.

(b) Use the following parameters $a = 0.1$, $\gamma = 2$, $\epsilon = 0.005$, $I = 0$, $D = 5 \cdot 10^{-5}$, for $h = 0.01$ and initial conditions

$$v(x, y, 0) = \exp\left(-100(x^2 + y^2)\right)$$
$$w(x, y, 0) = 0.0.$$

Note that $v = 0$, $w = 0$ is a stable steady state of the system. Call this the rest state. For these initial conditions the voltage has been raised above rest in the bottom corner of the domain. Generate a numerical solution up to time $t = 300$. What time step did you use and why? Visualize the voltage and describe the solution.

(c) Use the same parameters from part (b), but use the initial conditions

$$v(x, y, 0) = 1 - 2x$$
$$w(x, y, 0) = 0.05y,$$

and run the simulation until time $t = 600$. Show the voltage at several points in time (pseudocolor plot, or contour plot, or surface plot $z = V(x, y, t)$) and describe the solution.

The dynamics of excitable media is a fascinating subject from both the mathematical and physiological perspectives. The electrical patterns that you simulated in part (c) are related to cardiac arrhythmias. For more information see the book *Mathematical Physiology* by Keener and Sneyd.

Just for fun, (there is no need to turn this in or even do it) try to find an input current $I(x, y, t)$ in the form of a short pulse (e.g. $I(x, y, t) = f(x, y) \exp\left(-\kappa(t - t_p^2)\right)$) so that the normal electrical wave from part (b) degenerates into an arrhythmia like that from part (c). Then try to find a pulse of current that will eliminate the arrhythmia. This second task may be easier. What to the doctors on TV do?

Figure 10: Problem statement

4.1 Part(a)

The equations to solve are the following

$$\frac{\partial v}{\partial t} = D\Delta v + (a - v)(v - 1)v - w + I$$
$$\frac{\partial w}{\partial t} = \epsilon\Big(v - \gamma w\Big)$$

The first PDE $\frac{\partial v}{\partial t} = D\Delta v + (a - v)(v - 1)v - w + I$ was solved by the splitting method by solving the diffusion equation $\frac{\partial v}{\partial t} = D\Delta v$ using ADI method separately and then by solving the reaction (non-linear) equation $\frac{\partial v}{\partial t} = (a - v)(v - 1)v - w + I$ along with $\frac{\partial w}{\partial t} = \epsilon\Big(v - \gamma w\Big)$ separately. The following is a coupled first order non-linear differential equations system (the reaction ODE is nonlinear in votage v)

$$\frac{dv}{dt} = (a - v)(v - 1)v - w + I$$
$$\frac{dw}{dt} = \epsilon\left(v - \gamma w\right)$$

The above system was solved using Runge-Kutta order 4. The following diagram illustrates the time line for one full splitting step.

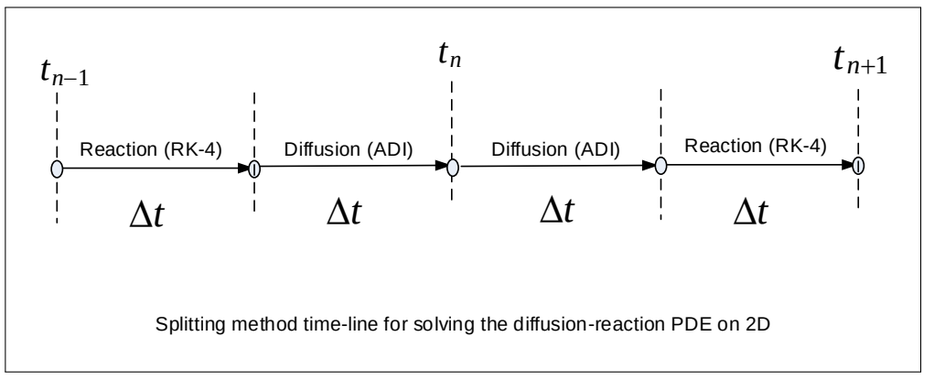


Figure 11: time line for one full splitting step

ADI was described in problem 2, and the same function was reused for this problem for the diffusion solver. For solving the reaction system of equations, RK4 was implemented as follows. define

$$f(v, w) = (a - v)(v - 1)v - w + I$$

and also define

$$g(v, w) = \epsilon\left(v - \gamma w\right)$$

Therefore, the RK4 solver for the above system becomes

$$v^{n+1} = v^n + \frac{1}{6}(m_1 + 2m_2 + 2m_3 + m_4)$$
$$w^{n+1} = w^n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

Where

$$m_1 = \Delta t f(v, w)$$
$$m_2 = \Delta t f\left(v + \frac{1}{2}m_1, w + \frac{1}{2}k_1\right)$$
$$m_3 = \Delta t f\left(v + \frac{1}{2}m_2, w + \frac{1}{2}k_2\right)$$
$$m_4 = \Delta t f(v + m_3, w + k_3)$$

And

$$k_1 = \Delta t g(v, w)$$
$$k_2 = \Delta t g\left(v + \frac{1}{2}m_1, w + \frac{1}{2}k_1\right)$$
$$k_3 = \Delta t g\left(v + \frac{1}{2}m_2, w + \frac{1}{2}k_2\right)$$
$$k_4 = \Delta t g(v + m_3, w + k_3)$$

Another point regarding the splitting method. It was required to decide which splitting method to use. Should a simple splitting, Strang splitting or the 2-step splitting method which was described in class be used? To make sure the second order accuracy of ADI 2D in time is preserved, simple splitting was not used (unless the operators commute, this would have caused the scheme to become first order accurate in time). Instead, the two step splitting method was used, as it was found to be simpler than Strang method to implement.

4.2 Part(b)

The program written in part(a) was run using the parameters given. The time step used was set to be the same as the space step. This time step is recommended for the ADI The diffusion solver as it is a second order accurate in time and space. This is the fast system (the stiff part of the system), hence making the time step larger than the space step would not give accurate results, even though it will remain a stable scheme. Keeping the time step the same as the space step seemed to be a good choice, as it kept the time resolution and the space resolution the same. The same time step was then used for the reaction solver, as was required by the splitting method to keep each step the same length.

The following shows the visualization of the voltage solution for up to 300 seconds as required using the surf() command.

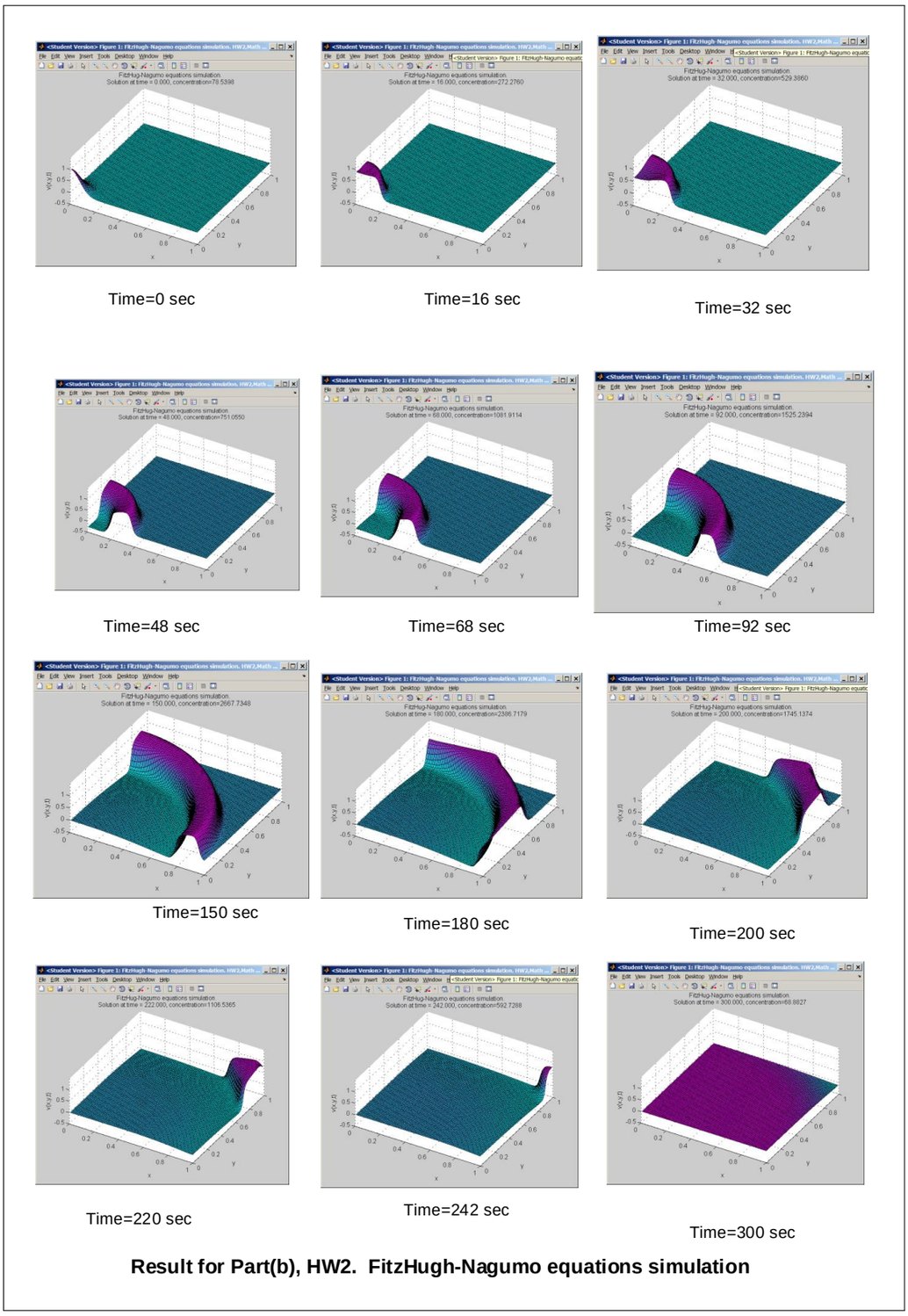
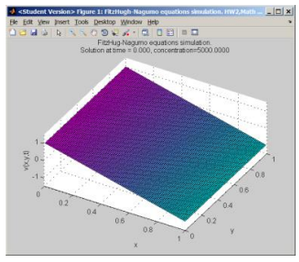


Figure 12: visualization of the voltage solution for up to 300 seconds

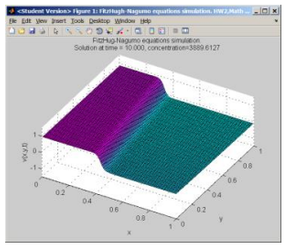
The solution $v(t)$ started from a peak value at one corner of the square. Shortly after, at about 50 seconds, a wave started to form, the wave front became large and it spread out and advanced with time. When the pulse reached the boundary on the other corner, it started to diffuse and by $t = 300$ seconds, the pulse has completely disappeared.

4.3 Part(c)

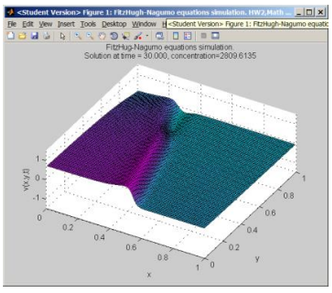
The following is the result of the simulation for this part.



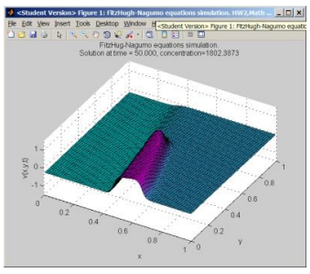
t=0 seconds



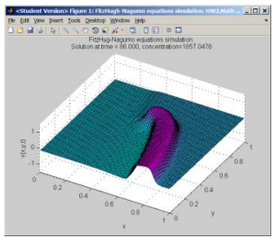
t=10 seconds



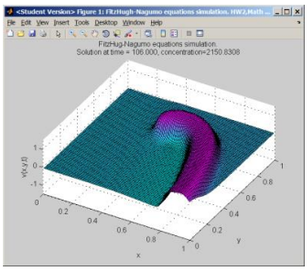
t=30 seconds



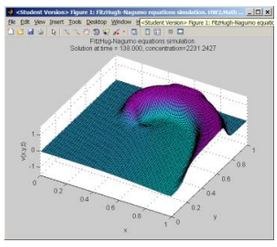
t=50 seconds



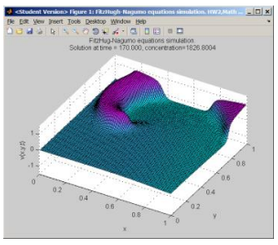
t=86 seconds



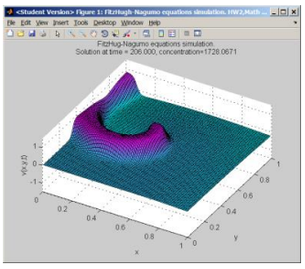
t=106 seconds



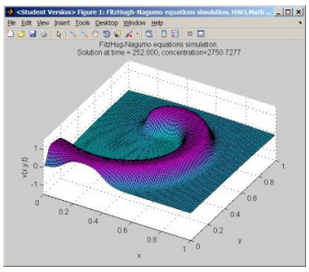
t=138 seconds



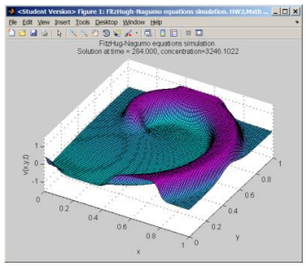
t=170 seconds



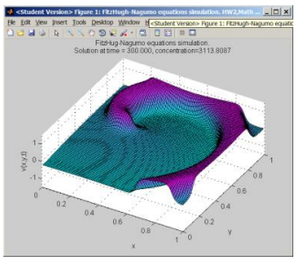
t=206 seconds



t=250 seconds



t=284 seconds



t=300 seconds

Result of part c problem 3. First 300 seconds

Figure 13: simulation of part c

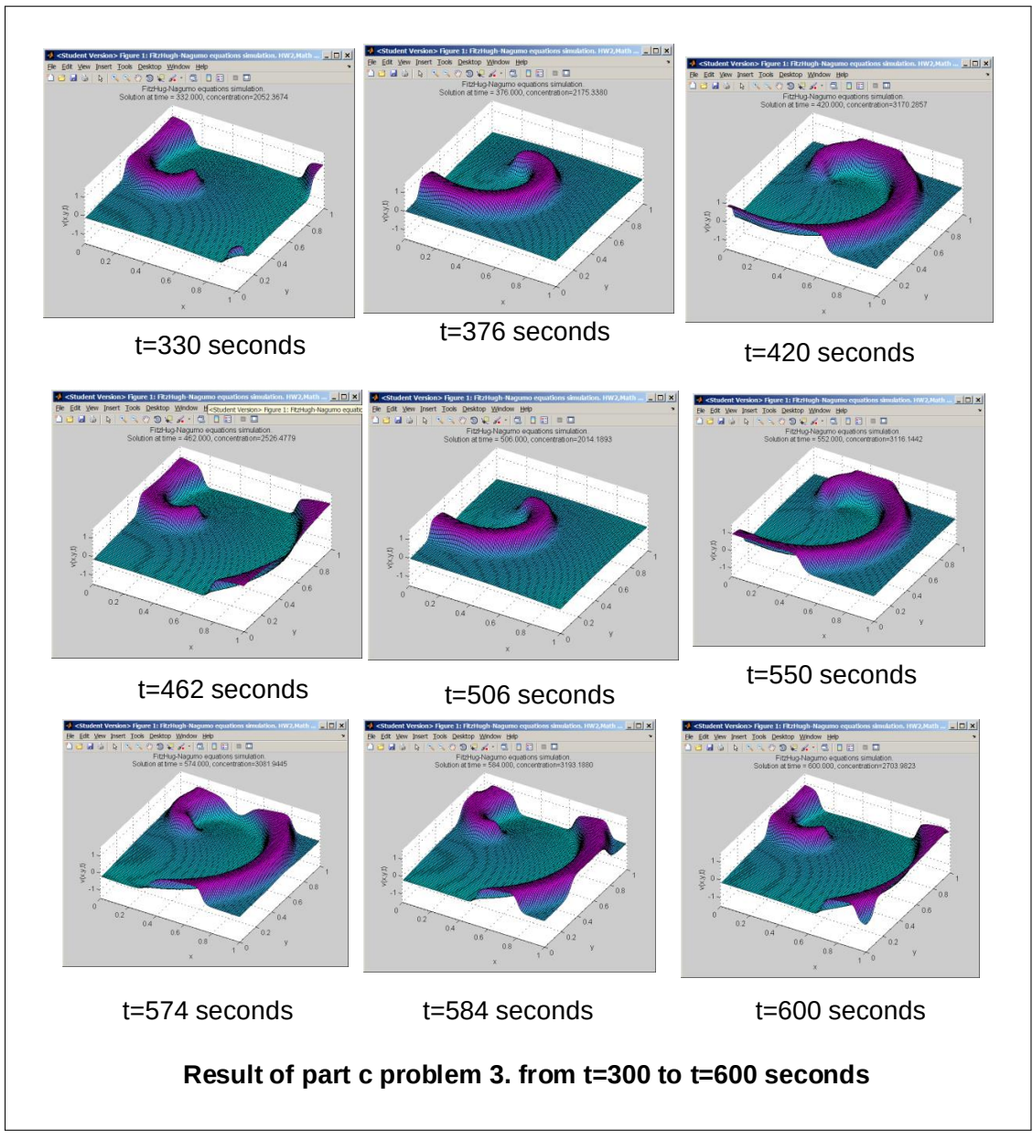


Figure 14: Up to 300 seconds

In this simulation, the pulse that appeared after about 50 second, quickly became a spiral, and did not appear to diffuse as was the case in part (d). At the end of the simulation, the pulse was continuing to spiral in the same rotation direction it started with. The above phenomena seem to be termed an arrhythmia pulse.

One common theme between part(c) and part (b), is the formation of a wave like motion that traveles across the domain. The difference was in the shape of the pulse, the direction it moves to and the amount of diffusion that occured.

5 Appendix

5.1 Problem 1 appendix

Derivation of ADI equations for 2D and 3D for the diffusion problem Given $u_t = \Delta u$ (D is assumed 1), then in 2D forward Euler (explicit) gives

$$\frac{u_{ij}^{n+1} - u_{ij}^{n}}{k} = \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right)_{ij}^{n}$$

While C-N method gives

$$\frac{u_{ij}^{n+1} - u_{ij}^{n}}{k} = \frac{1}{2} \left[\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right)_{ij}^{n+1} + \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right)_{ij}^{n} \right]$$

However, in ADI, the time step itself is divided by half, and in the first half step, one of the spatial second derivatives is implicit while and the other spatial second derivative is explicit. In the second half step, these are reversed.

$$\frac{u_{ij}^{n+\frac{1}{2}} - u_{ij}^{n}}{k/2} = \overbrace{\left(\frac{\partial^2 u}{\partial x^2}\right)_{ij}^{n+\frac{1}{2}}}^{implicit} + \overbrace{\left(\frac{\partial^2 u}{\partial y^2}\right)_{ij}^{n}}^{\text{explicit}}$$

$$\frac{u_{ij}^{n+1} - u_{ij}^{n+\frac{1}{2}}}{k/2} = \overbrace{\left(\frac{\partial^2 u}{\partial x^2}\right)_{ij}^{n+\frac{1}{2}}}^{\text{explicit}} + \overbrace{\left(\frac{\partial^2 u}{\partial y^2}\right)_{ij}^{n+1}}^{implicit}$$

Writing $\frac{\partial^2 u}{\partial x^2} = L_x = \frac{u_{i-1,j} - 2u_{ij} + u_{i+1,j}}{h^2}$ and $\frac{\partial^2 u}{\partial y^2} = L_y = \frac{u_{i,j-1} - 2u_{ij} + u_{i,j+1}}{h^2}$, the above 2 equations become

$$\frac{u_{ij}^{n+\frac{1}{2}} - u_{ij}^{n}}{k/2} = \frac{u_{i-1,j}^{n+\frac{1}{2}} - 2u_{ij}^{n+\frac{1}{2}} + u_{i+1,j}^{n+\frac{1}{2}}}{h^2} + \frac{u_{i,j-1}^{n} - 2u_{ij}^{n} + u_{i,j+1}^{n}}{h^2}$$

$$\frac{u_{ij}^{n+1} - u_{ij}^{n+\frac{1}{2}}}{k/2} = \frac{u_{i-1,j}^{n+\frac{1}{2}} - 2u_{ij}^{n+\frac{1}{2}} + u_{i+1,j}^{n+\frac{1}{2}}}{h^2} + \frac{u_{i,j-1}^{n+1} - 2u_{ij}^{n+1} + u_{i,j+1}^{n+1}}{h^2}$$

Moving all implicit terms to the LHS and rearranging results in

$$u_{ij}^{n+\frac{1}{2}} - \frac{k}{2}\frac{u_{i-1,j}^{n+\frac{1}{2}} - 2u_{ij}^{n+\frac{1}{2}} + u_{i+1,j}^{n+\frac{1}{2}}}{h^2} = u_{ij}^{n} + \frac{k}{2}\frac{u_{i,j-1}^{n} - 2u_{ij}^{n} + u_{i,j+1}^{n}}{h^2}$$

$$u_{ij}^{n+1} - \frac{k}{2}\frac{u_{i,j-1}^{n+1} - 2u_{ij}^{n+1} + u_{i,j+1}^{n+1}}{h^2} = u_{ij}^{n+\frac{1}{2}} + \frac{k}{2}\frac{u_{i-1,j}^{n+\frac{1}{2}} - 2u_{ij}^{n+\frac{1}{2}} + u_{i+1,j}^{n+\frac{1}{2}}}{h^2}$$

Hence, in operator form the above becomes

$$\left(I - \frac{k}{2}L_x\right)u_{ij}^{n+\frac{1}{2}} = \left(I + \frac{k}{2}L_y\right)u_{ij}^{n}$$

$$\left(I - \frac{k}{2}L_y\right)u_{ij}^{n+1} = \left(I + \frac{k}{2}L_x\right)u_{ij}^{n+\frac{1}{2}}$$

In the class notes, u_{ij}^{*} was used to represent $u_{ij}^{n+\frac{1}{2}}$ but they are the same. The above is the ADI scheme for 2D. The 3D equations are now derived. Since three different directions exist now, the time step is divided into 3. This results in

$$\frac{u_{ij}^{n+\frac{1}{3}} - u_{ij}^{n}}{\Delta t/3} = \overbrace{\left(\frac{\partial^2 u}{\partial x^2}\right)_{ij}^{n+\frac{1}{3}}}^{implicit} + \overbrace{\left(\frac{\partial^2 u}{\partial y^2}\right)_{ij}^{n}}^{\text{explicit}} + \overbrace{\left(\frac{\partial^2 u}{\partial z^2}\right)_{ij}^{n}}^{\text{explicit}}$$

$$\frac{u_{ij}^{n+\frac{2}{3}} - u_{ij}^{n+\frac{1}{3}}}{\Delta t/3} = \overbrace{\left(\frac{\partial^2 u}{\partial x^2}\right)_{ij}^{n+\frac{1}{3}}}^{\text{explicit}} + \overbrace{\left(\frac{\partial^2 u}{\partial y^2}\right)_{ij}^{n+\frac{2}{3}}}^{implicit} + \overbrace{\left(\frac{\partial^2 u}{\partial z^2}\right)_{ij}^{n+\frac{1}{3}}}^{\text{explicit}}$$

$$\frac{u_{ij}^{n+1} - u_{ij}^{n+\frac{2}{3}}}{\Delta t/3} = \overbrace{\left(\frac{\partial^2 u}{\partial x^2}\right)_{ij}^{n+\frac{2}{3}}}^{\text{explicit}} + \overbrace{\left(\frac{\partial^2 u}{\partial y^2}\right)_{ij}^{n+\frac{2}{3}}}^{\text{explicit}} + \overbrace{\left(\frac{\partial^2 u}{\partial z^2}\right)_{ij}^{n+1}}^{implicit}$$

Similar to what done in the 2D case, the above are rearranged resulting in

$$\left(I - \frac{D\Delta t}{3}L_x\right)\mathbf{u}^{n+\frac{1}{3}} = \left(I + \frac{D\Delta t}{3}L_y + \frac{D\Delta t}{3}L_z\right)\mathbf{u}^{n} \tag{1}$$

$$\left(I - \frac{D\Delta t}{3}L_y\right)\mathbf{u}^{n+\frac{2}{3}} = \left(I + \frac{D\Delta t}{3}L_x + \frac{D\Delta t}{3}L_z\right)\mathbf{u}^{n+\frac{1}{3}} \tag{2}$$

$$\left(I - \frac{D\Delta t}{3}L_z\right)\mathbf{u}^{n+1} = \left(I + \frac{D\Delta t}{3}L_x + \frac{D\Delta t}{3}L_y\right)\mathbf{u}^{n+\frac{2}{3}} \tag{3}$$

6 Matlab Source code developed for this HW

6.1 nma_math228b_HW2_prob2.m

```
function nma_math228b_HW2_prob2
% This function implements refinement study for HW2
% problem 2, Math 228B, Winter 2011, UC Davis
%
%
% By Nasser M. Abbasi

% set up initialization for the error table, such as headings
% and formating

close all;

% for formating of error table
titles = {'#','N','delt','h','|u|','mean(u)','|e|','ratio'};
fms    = {'d','d','.5f','.5f','.5f','.5f','.4e','.5f'};
wid    = 13;
fileID = 1;

% use 8 runs, and allocate the table to store the error and ratios
N=5;
table=zeros(N,8); % #, t, h, |u|,|u-u_last|, ratio, N, mean(u), |exact|

% Initialize space and time steps.
grid_size = 9;
h1 = figure();
D  = 0.1;  %diffusion constant
time_to_run = 1; % 1 second

for n = 1:N

   % Simulatiously divide space step and time step.

   grid_size = grid_size * 3;
   h = 1/grid_size;
   k = h;

   [u,u_steady_state] = solve_2D_diffusion(grid_size,h,k,D,time_to_run);

   % the numerical solution now is stored in u. Make
   % a new entry in the error table for this iteration.
   table(n,1) = n;
   table(n,7) = grid_size;
   table(n,8) = mean(mean(u));

   table(n,2) = k;
   table(n,3) = h;

   table(n,4) = h*norm(u(:),2);  % use grid 2-norm

   if n>1
      table(n,5) = abs(table(n-1,4)-table(n,4)); %e
      %table(n,5) = h*norm( u-u_steady_state,2); %e

      if n==2
         table(n,6)=1;
      else
         table(n,6) = table(n-1,5)/table(n,5); %e ratio
      end
```

```
      [hd,bdy]=nma_format_matrix(titles, ...
         [table(2:n,1) table(2:n,7) table(2:n,2) table(2:n,3) table(2:n,4)  table(2:n,8) tab
         wid,fms,fileID,true );

      clf(h1);
      set(0,'CurrentFigure',h1);
      ax = axes();
      set(h1, 'CurrentAxes',ax);
      cla('reset');

      text(.1,.60,bdy,'FontSize',10);
      set(ax,'YTick',[]);
      set(ax,'XTick',[]);
      text(.1,.9,hd,'FontSize',10);
      title('result of refinement study');
      drawnow();

   end
end

% The refinment study is completed. Generate plots and error table

h2  = figure();
ax2 = axes();
set(0,'CurrentFigure',h2);
set(h2, 'CurrentAxes',ax2);
cla('reset');
set(0,'defaultaxesfontsize',8) ;

loglog(table(2:end,3),table(2:end,5),'-d');
xlabel('log(h)','FontSize',8);
title({'refinement study result, ';'log vs successive errors difference'},...
   'FontSize',8);
ylabel('log(error norm)','FontSize',8);
grid on;
end

%-----------------------
function [u,u_steady_state]=solve_2D_diffusion(...
   grid_size,...
   h,... % space step size
   k,... % time step size
   D,... % diffusion constant
   max_t... % maximum time to run solver for
   )

n  = grid_size-2; %internal nodes
ic = @(X,Y) exp( -10*((X-0.4).^2 + (Y-0.4).^2 ));   % initial data
[X,Y]  = meshgrid(h/2:h:1-h/2,1-h/2:-h:h/2); % coordinates
u_mean = quad2d(ic,h/2,1-(h/2),h/2,1-(h/2));
%u_mean = quad2d(ic,0,1,0,1);

%ic= @(X,Y) -exp( -(X-0.25).^2 - (Y-0.6).^2 );

% create sparse matrices for A and B (implicit and explicit) see HW report
A     = lap2D_diffusion_ADI_A(n,D,k,h);
A_RHS = lap2D_diffusion_ADI_A_RHS(n,D,k,h);

u = ic(X,Y); % initial U
u_steady_state = zeros(size(u));
u_steady_state(:,:)=u_mean;
u_max  = max(max(u));
u_min  = min(min(u));
```

```
h1        = figure();
current_t = 0;
done      = false;

while not(done)

   % solve for U*
   tmp = reshape(flipud(u(2:end-1,2:end-1))',n^2,1);
   sol = A\(A_RHS*tmp);
   u(2:end-1,2:end-1) = flipud(reshape(sol,n,n)');

   %update the boundaries
   u = update_BC(u);

   % solve for U_n+1
   tmp = reshape(flipud(u(2:end-1,2:end-1)),n^2,1);
   sol = A\(A_RHS*tmp);
   u(2:end-1,2:end-1) = flipud(reshape(sol,n,n));

   u = update_BC(u);

   set(0,'CurrentFigure',h1);
   surf(X,Y,u);
   colormap cool;
   title(sprintf('solution at time = %1.3f, D=%.3f, N=%d\nsteady state=%1.4f, h=%1.5f',...
      current_t,D,grid_size,u_mean,h));

   hold on;
   mesh(X,Y,u_steady_state);
   zlim([u_min u_max]);
   drawnow();
   hold off;

   %update current time and check if reached end of time
   current_t = current_t + k;
   if current_t > max_t
      done = true;
   end


end
close(h1);
end

%------------------------
function A=lap2D_diffusion_ADI_A(...
   n, ... %size of matrix (1D size)
   D, ... %diffusion constant
   k, ... %time step
   h)     %space_step

r  = D*k/(2*h^2);
e  = ones(n,1);
B  = [-r*e (1+2*r)*e -r*e];
Lx = spdiags(B,[-1 0 1],n,n);
Ix = speye(n);
A  = kron(Ix,Lx);

% adjust A, see HW report

pos = 1:n:n^2;

for i = 1:length(pos)
```

```
   A(pos(i),pos(i))=1+r;
end

pos = n:n:n^2;

for i=1:length(pos)
   A(pos(i),pos(i))=1+r;
end

end
%---------------------------------------
function A=lap2D_diffusion_ADI_A_RHS(...
   n, ... %size of matrix (1D)
   D, ... %diffusion constant
   k, ... %time step
   h)     %space_step

r = D*k/(2*h^2);

e = ones(n^2,1);
B = [r*e (1-r)*e r*e];
A = spdiags(B,[-n 0 n],n^2,n^2);

%adjust matrix, see HW report
pos = n+1:n^2-n;
for i=1:length(pos)
   A(pos(i),pos(i))=1-2*r;
end

end

%----------------------
function    u = update_BC(u)

   u(1,2:end-1)   = u(2,2:end-1);
   u(end,2:end-1) = u(end-1,2:end-1);
   u(2:end-1,1)   = u(2:end-1,2);
   u(2:end-1,end) = u(2:end-1,end-1);


   u(1,1)     = u(1,2);
   u(end,1)   = u(end-1,1);
   u(end,end) =  u(end,end-1);
   u(1,end)   =  u(1,end-1);

end
```