AM213B Numerical Methods for the Solution of Differential Equations

Lecture 10

List of topics in this lecture

- Crank-Nicolson method, 2nd order in time and in space
- Lax equivalence theorem: Consistency + Stability = Convergence
- Method of lines (MOL), numerical stability of ODE solvers vs PDE solvers in the framework of MOL
- Von Neumann Stability Analysis
- Numerical Solution of 2D Problems

Review:

<u>The heat equation:</u> $u_t = u_{xx}$

<u>Numerical grid</u>: (x_i, t_n)

<u>Notation of num. approx:</u> $u_i^n \approx u(x_i, t_n)$

<u>Vector notation:</u> $u^n = \left\{ u_i^n \ , \ 1 \le i \le N \right\}$

<u>Numerical operator:</u> $u^{n+1} = L_{num}(u^n)$

<u>Local truncation error:</u> $\left\{ e_i^n(\Delta x, \Delta t) \right\} = \left\{ u(x_i, t_{n+1}) \right\} - L_{num}\left\{ u(x_i, t_n) \right\}$

<u>Consistency</u>: $\displaystyle \lim_{\substack{\Delta x \to 0 \\ \Delta t \to 0}} \frac{e_i^n(\Delta x, \Delta t)}{\Delta t} = 0$

<u>Stability</u>: $\left\| (L_{num})^n \right\| \le C_T \quad \text{for all } n\Delta t \le T$

<u>The FTCS method:</u> $u_i^{n+1} = u_i^n + r\left(u_{i+1}^n - 2u_i^n + u_{i-1}^n \right), \qquad r = \dfrac{\Delta t}{(\Delta x)^2}$

It is stable for $r \le 1/2$, unstable for $r > 1/2$.

<u>The BTCS method:</u> $u_i^{n+1} = u_i^n + r\left(u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1} \right), \qquad r = \dfrac{\Delta t}{(\Delta x)^2}$

It is unconditionally stable (i.e., it is stable for any $r > 0$).

Norm of numerical solution: $$\|u^n\|_p = \left(\sum_{i=1}^{N} |u_i^n|^p \Delta x \right)^{\frac{1}{p}}$$

- It is a simple scaling of the regular vector norm.
- It converges to the function norm as $\Delta x \to 0$.

Both FTCS and BTCS have only first order accuracy in time.

We introduce a method that has second order in time and in space.

Crank-Nicolson method:

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

It is also called the CTCS method (Central-time, Central-space).

Its local truncation error is

$$e_i^n(\Delta x, \Delta t) = \Delta t\, O\left((\Delta t)^2 + (\Delta x)^2\right)$$

It is second order in time and in space. We will look at its stability later.

Next we study convergence.

Global error:

$$E_i^n(\Delta x, \Delta t) = u(x_i, t_n) - u_i^n$$

Convergence:

We say a method is convergent if $\lim\limits_{\substack{\Delta x \to 0 \\ \Delta t \to 0}} E_i^n(\Delta x, \Delta t) = 0$.

Lax equivalence theorem:

For a linear method, $u^{n+1} = L_{\text{num}}(u^n)$,

Consistency + Stability = Convergence.

Proof:

We use the vector notation:

$$u^n = \{u_i^n\}, \qquad e^n = \{e_i^n\}, \qquad E^n = \{E_i^n\}$$

We write the global error E^{n+1} as the sum of two parts:

$$E^{n+1} = \{u(x_i, t_{n+1})\} - u^{n+1}$$

$$= \underbrace{\{u(x_i,t_{n+1})\} - L_{num}\{u(x_i,t_n)\}}_{\text{Local truncation error } e^n} + \underbrace{L_{num}\{u(x_i,t_n)\} - L_{num}(u^n)}_{\text{Propagation of } E^n = \{u(x_i,t_n)\} - u^n}$$

This gives us a recursive equation on the global error.

$$E^{n+1} = e^n + L_{num}(E^n)$$

Applying this recursive equation repeatedly, we have

$$E^1 = e^0 + L_{num}(E^0)$$

$$E^2 = e^1 + L_{num}(E^1) = e^1 + L_{num}\left(e^0 + L_{num}(E^0)\right) = e^1 + L_{num}(e^0) + (L_{num})^2(E^0)$$

$$\cdots$$

$$E^n = \sum_{k=0}^{n-1}\left((L_{num})^k(e^{n-1-k})\right) + (L_{num})^n(E^0)$$

Taking the norm of both sides, we obtain

$$\left\|E^n\right\| \le \sum_{k=0}^{n-1}\left\|(L_{num})^k\right\|\cdot\left\|e^{n-1-k}\right\| + \left\|(L_{num})^n\right\|\cdot\left\|E^0\right\| \le C_T\cdot\sum_{k=0}^{n-1}\left\|e^{n-1-k}\right\| + C_T\cdot\left\|E^0\right\|$$

$$\le C_T n\left(\max_{0\le k\le n-1}\left\|e^k(\Delta x,\Delta t)\right\|\right) + C_T\left\|E^0\right\|$$

$$\le C_T(n\Delta t)\left(\max_{0\le k\le n-1}\frac{\left\|e^k(\Delta x,\Delta t)\right\|}{\Delta t}\right) + C_T\cdot\left\|E^0\right\| \to 0 \text{ as } (\Delta x,\Delta t)\to 0$$

End of proof

We cast the 3 methods, FTCS, BTCS and CTCS, into a unified framework. The framework allows us to utilize high-order L-stable Runge-Kutta solvers. The framework also provides theoretical insight into the numerical stability of PDE solvers.

Method of lines (MOL)

Recall the IBVP

$$\begin{cases} u_t = u_{xx} \\ u(x,0) = f(x) \\ u(0,t) = g_1(t), \qquad u(L,t) = g_2(t) \end{cases}$$

We first discretize only in the spatial dimension.

Let $u_i(t)$ = the numerical approximation of $u(x_i, t)$.

The IBVP of PDE becomes an IVP of ODE system for $\{u_i(t), i = 1, 2, \ldots, N\}$.

$$\frac{du_i(t)}{dt} = \frac{1}{(\Delta x)^2}\Big(u_{i+1}(t) - 2u_i(t) + u_{i-1}(t)\Big), \qquad i = 1, 2, \ldots, N \qquad \text{(E01)}$$

$u_i(0) = f(x_i)$, $\quad i = 1, 2, \ldots, N$

$u_0(t) = g_1(t)$, $\qquad u_{N+1}(t) = g_2(t)$

Formulation (E01) is called the method of lines (MOL).

We write (E01) in the matrix-vector form

$$\frac{d\vec{u}(t)}{dt} = A\vec{u}(t) + \vec{b}(t) \qquad \text{(E01B)}$$

where vector u, vector b and matrix A are

$$\vec{u}(t) = \begin{pmatrix} u_1(t) \\ u_2(t) \\ \vdots \\ u_N(t) \end{pmatrix}, \qquad \vec{b}(t) = \frac{1}{(\Delta x)^2}\begin{pmatrix} g_1(t) \\ 0 \\ \vdots \\ 0 \\ g_2(t) \end{pmatrix}$$

$$A = \frac{1}{(\Delta x)^2}\begin{pmatrix} -2 & 1 & & \\ 1 & -2 & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & -2 \end{pmatrix}_{N\times N}, \qquad \Delta x = \frac{L}{N+1}$$

In the discretization, # of sub-intervals = $N + 1$; # of internal points = N. The spatial step (Δx) and matrix size (N) are related by $\Delta x = \dfrac{L}{N+1}$.

Observations on vector u, vector b and matrix A:

- Vector u contains values of u at internal points.
- Vector b contains the effects of boundary conditions.
- Matrix A corresponds to the second order difference operator

$$D_x^2 : \{u_i\} \longrightarrow \left\{\frac{u_{i+1} - 2u_i + u_{i-1}}{(\Delta x)^2}\right\}$$

with zero boundary conditions: $u_0 = 0$ and $u_{N+1} = 0$

Remarks:

- When the zero BCs, D_x^2 is a linear operator acting on $\{u_i, 1 \le i \le N\}$ (u at internal points) even though its expression involves $u_0 = 0$ and $u_{N+1} = 0$.

- To separate the unknown u at internal points from the prescribed BCs, we use the zero BCs with operator D_x^2 to construct matrix A in the linear system, regardless of the real BSc in the IBVP.
- The real BCs in the IBVP are taken care of in vector b, not in matrix A.

Once we have the MOL ODE system (E01B), we can use any ODE solver to solve it.

- If we use forward Euler, we get FTCS.
- If we use backward Euler, we get BTCS.
- If we use trapezoidal, we get Crank-Nicolson (CTCS).
- We can use the 2-stage DIRK or the 3-stage DIRK, which are L-stable and have second order accuracy and third order accuracy, respectively.

Next we study the stability in the framework of method of lines (MOL).

Numerical stability in the framework of MOL

We consider the model IBVP (with homogeneous BCs)

$$\begin{cases} u_t = u_{xx} \\ u(x,0) = f(x) \\ u(0,t) = 0, \qquad u(1,t) = 0 \end{cases}$$

MOL ODE system in the matrix-vector form:

$$\frac{d\vec{u}(t)}{dt} = A\vec{u}(t), \qquad A = \frac{1}{(\Delta x)^2}\begin{pmatrix} -2 & 1 & & \\ 1 & -2 & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & -2 \end{pmatrix}_{N\times N}, \qquad \Delta x = \frac{1}{N+1}$$

Eigenvalues and eigenvectors of matrix A

Matrix A is tridiagonal, real, and symmetric.

==> It has real eigenvalues and orthogonal eigenvectors.

Theorem:

The eigenvalues and eigenvectors of matrix A are

$$\left.\begin{array}{l} \lambda^{(k)} = \dfrac{2}{(\Delta x)^2}\left(\cos(k\pi\Delta x) - 1\right) \\ w^{(k)} = \left\{\sin(k\pi i\Delta x),\ \ i = 1,2,\ldots,N\right\} \end{array}\right\}, \quad k = 1,2,\ldots,N \qquad \text{(E02)}$$

Proof: (homework problem)

Remark:

It is worthwhile to compare with the continuous Sturm-Liouville problem

$$\begin{cases} u'' = \lambda u \\ u(0)=0, \quad u(1)=0 \end{cases}$$

which has eigenvalues and eigenfunctions

$$\lambda^{(k)} = -k^2\pi^2$$

$$w^{(k)}(x) = \sin(k\pi x), \quad k = 1, 2, \ldots,$$

All eigenvalues of matrix A are negative, each corresponding to the decay of a mode.

We examine the slowest decay and the fastest decay.

The smallest eigenvalue (in absolute value) of matrix A:

$$\lambda^{(1)} = \frac{2}{(\Delta x)^2}\left[\cos(\pi\Delta x) - 1\right] = \frac{2}{(\Delta x)^2}\left[-\frac{(\pi\Delta x)^2}{2} + O\left((\Delta x)^4\right)\right] = -\pi^2 + O\left((\Delta x)^2\right)$$

In the above, we used the expansion $\cos(\varepsilon) = 1 - \frac{1}{2}\varepsilon^2 + O(\varepsilon^4)$.

The largest eigenvalue (in absolute value) of matrix A:

$$\lambda^{(N)} = \frac{2}{(\Delta x)^2}\left[\cos(N\pi\Delta x) - 1\right] = \frac{2}{(\Delta x)^2}\left[\cos(\pi - \pi\Delta x) - 1\right], \qquad (N+1)\Delta x = 1$$

$$= \frac{2}{(\Delta x)^2}\left[-\cos(\pi\Delta x) - 1\right] = \frac{2}{(\Delta x)^2}\left[-2 + O\left((\Delta x)^2\right)\right] = \frac{-4}{(\Delta x)^2} + O(1)$$

The ratio of the two extreme eigenvalues is

$$\frac{\lambda^{(N)}}{\lambda^{(1)}} = \frac{4}{\pi^2}\cdot\frac{1}{(\Delta x)^2} + O(1)$$

For small Δx, this is a *stiff system*, and the stiffness increases, as Δx is refined!

Behavior of ODE solvers on the MOL ODE system

Recall our previous study of ODE solvers applied to $u' = -\lambda u$:

Euler method:

Numerical solution is bounded if

$$\Delta t \le \frac{2}{|\lambda_{\max}|} = \frac{2}{4/(\Delta x)^2} = \frac{(\Delta x)^2}{2} \quad \text{<==>} \quad r \equiv \frac{\Delta t}{(\Delta x)^2} \le \frac{1}{2}$$

This is the same as the stability condition we obtained for FTCS.

Backward Euler method:

Numerical solution is bounded for any Δt.

This corresponds to the unconditional stability of BTCS.

Trapezoidal:

Numerical solution is bounded for any Δt.

This corresponds to the unconditional stability of CTCS.

Observation on numerical stability of ODE solvers vs PDE solvers

ODE solvers	PDE solvers
An ODE system may be stiff but the *stiffness is fixed* as numerical resolution is refined. As Δt is refined, $\Delta t \le \frac{2}{\lvert\lambda_{\max}\rvert}$ will eventually be satisfied, and all methods will be well behaved for small Δt. The practical issue: can we afford a tiny time step satisfying $\Delta t \le 2/\lvert\lambda_{\max}\rvert$? To make the numerical solution well behaved for relatively large time steps, we use L-stable methods.	After MOL (method of lines) discretization, a PDE becomes an ODE system. *The stiffness of the system increases* as numerical resolution is refined. As $(\Delta x, \Delta t)$ is refined, $\Delta t \le \frac{2}{\lvert\lambda_{\max}\rvert} = \frac{1}{2}(\Delta x)^2$ is not satisfied automatically. Condition $\frac{\Delta t}{(\Delta x)^2} \le \frac{1}{2}$ must be imposed to ensure the numerical stability for explicit methods. To relax this condition, we use L-stable ODE solvers with MOL.

Von Neumann Stability Analysis

Consider a linear method

$$L_{\text{Left}}(u^{n+1}) = L_{\text{Right}}(u^n)$$

Here the left-hand side and right-hand side are written out explicitly.

Procedure of von Neumann stability analysis

- We try solution of the form

$$u_i^n = \rho^n \exp(\sqrt{-1}\,\xi i \Delta x), \qquad \text{a discrete version of } \rho^n \exp(\sqrt{-1}\,\xi x)$$

where ρ is called the magnification factor.

- Substitute into the numerical method to calculate ρ.

$$\rho^{n+1} L_{\text{Left}}\left\{\exp(\sqrt{-1}\,\xi i \Delta x)\right\} = \rho^n L_{\text{Right}}\left\{\exp(\sqrt{-1}\,\xi i \Delta x)\right\}$$

$$\Longrightarrow \qquad \rho = \frac{L_{\text{Right}}\left\{\exp(\sqrt{-1}\,\xi i\Delta x)\right\}}{L_{\text{Left}}\left\{\exp(\sqrt{-1}\,\xi i\Delta x)\right\}}$$

Note: ρ depends on $(\xi, \Delta t)$ and $r = (\Delta t)/(\Delta x)^2$.

- Determine the stability as follows.

If $\left|\rho(\xi,\Delta t)\right| \le 1 + C\Delta t$ for small Δt and all ξ is valid for r within a certain range, then the method is stable for r in that range.

If $\left|\rho(\xi,\Delta t)\right| \le 1 + C\Delta t$ for small Δt and all ξ is valid for all r, then the method is unconditionally stable (stable for all r).

If $\left|\rho(\xi,\Delta t)\right| \le 1 + C\Delta t$ for small Δt and all ξ is valid for none of r, then the method is unconditionally unstable (unstable for all r).

Examples:

von Neumann stability analysis on FTCS

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

Substitute $u_i^n = \rho^n e^{\sqrt{-1}\xi i\Delta x}$ into the method to calculate ρ

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

Dividing by $\rho^n e^{\sqrt{-1}\xi i\Delta x}$

$$\rho = 1 + 2r\left(\cos(\xi\Delta x) - 1\right)$$

Recall $\cos(\alpha) - 1 = -2\sin^2\left(\alpha/2\right)$

$$\Longrightarrow \qquad \rho = 1 - 4r\sin^2\left(\xi\Delta x/2\right)$$

For stability we need $|\rho| \le 1 + C\Delta t$. The expression of ρ above gives $\rho \le 1$.

We only need to check whether or not $\rho \ge -(1 + C\Delta t)$ is true.

$$1 - 4r\sin^2\left(\xi\Delta x/2\right) \ge -(1 + C\Delta t) \quad \text{for small } \Delta t \text{ and all } \xi$$

if and only if $\quad 4r\sin^2\left(\xi\Delta x/2\right) \le 2 \quad$ for all ξ

if and only if $\quad 4r \le 2$

if and only if $\quad r \le 1/2$

Therefore, we conclude that the FTCS method is stable if and only if $r \le 1/2$.

This is the same as the stability condition obtained in our previous stability analysis.

von Neumann stability analysis on BTCS

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

Move u^{n+1} terms to the left side

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

Substitute $u_i^n = \rho^n e^{\sqrt{-1}\xi i\Delta x}$ into the method to calculate ρ

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

Dividing by $\rho^n e^{\sqrt{-1}\xi i\Delta x}$

$$\rho\left(1 - 2r(\cos(\xi\Delta x) - 1)\right) = 1 \qquad \text{==>} \qquad \rho\left(1 + 4r\sin^2\left(\xi\Delta x/2\right)\right) = 1$$

$$\text{==>} \qquad \rho = \frac{1}{1 + 4r\sin^2\left(\xi\Delta x/2\right)}$$

==> $|\rho| \le 1$ for any r

Therefore, we conclude that the BTCS method is unconditionally stable.

This is the same as what we obtained in our previous stability analysis.

Numerical Solution of 2D Problems

Two-dimensional IBVP of the heat equation

$$\begin{cases} u_t = u_{xx} + u_{yy}, & (x,y) \in \Omega \\ u(x,y,0) = f(x,y), & (x,y) \in \Omega \\ u(x,y,t) = g(x,y,t), & (x,y) \in \partial\Omega \end{cases}$$

We consider the case of a rectangular region: $\Omega = [0,\ L_x] \times [0,\ L_y]$.

Numerical grid:

$$\Delta x = \frac{L_x}{N_x + 1}, \qquad x_i = i\Delta x, \quad x_0 = 0, \quad x_{N_x+1} = L_x$$

$$\Delta y = \frac{L_y}{N_y+1}, \quad y_j = j\Delta y, \quad y_0 = 0, \quad y_{N_y+1} = L_y$$

$$t_n = n\Delta t$$

$$u_{i,j}^n = \text{Numerical approximation of } u(x_i, y_j, t_n)$$

<u>FTCS method:</u>

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

$$u_{i,j}^0 = f(x_i, y_j)$$

$$u_{0,j}^n = g(0, y_j, t_n), \quad u_{(N_x+1),j}^n = g(L_x, y_j, t_n)$$

$$u_{i,0}^n = g(x_i, 0, t_n), \quad u_{i,(N_y+1)}^n = g(x_i, L_y, t_n)$$

<u>Caution:</u>

In Matlab implementation, a function of two variables, $u(x, y)$, on a numerical grid, is stored in a matrix $u = \{u(k, \ell)\}$. In the Matlab matrix,

row index, k, corresponds to y direction;

column index, ℓ, corresponds to x direction;

$u(k, \ell)$ stores the numerical approximation of $u(x_{\ell-1}, y_{k-1})$ or $u(x_\ell, y_k)$ depending on whether or not boundary values are included.

In an explicit method, we usually include boundary values in matrix u and use boundary values directly in the time evolution.

In an implicit method, matrix u contains only unknown values of u at internal points; it does not include boundary values.

The exact solution, the theoretical numerical solution and the numerical solution stored in Matlab are related by

$$\underbrace{u(x_i, y_j)}_{\text{Exact solution}} \longrightarrow \underbrace{u_{i,j}}_{\substack{\text{Theoretical}\\ \text{numerical solution}}} \longrightarrow \begin{cases} \underbrace{u(j+1, i+1)}_{\substack{\text{Numerical solution}\\ \text{stored in Matlab}}} & \text{explicit method} \\ \underbrace{u(j, i)}_{\substack{\text{Numerical solution}\\ \text{stored in Matlab}}} & \text{implicit method} \end{cases}$$

So be careful!!!

<u>Implementation of FTCS</u>

Matlab matrix $\left\{u^n(j,i),\ 1 \le j \le N_y + 2,\ 1 \le i \le N_x + 2\right\}$ stores $\left\{u_{i-1,j-1}^n\right\}$, values of u at all points, including boundary points.

- Update $\left\{u^n(j,i)\right\}$ at boundary points, according to the IBVP

$$\begin{aligned}
&u^n(j,1)=g(0,(j-1)\Delta y,t_n) && \text{for } 2\le j\le N_y+1\\
&u^n(j,N_x+2)=g(L_x,(j-1)\Delta y,t_n) && \text{for } 2\le j\le N_y+1\\
&u^n(1,i)=g((i-1)\Delta x,0,t_n) && \text{for } 2\le i\le N_x+1\\
&u^n(N_y+2,i)=g((i-1)\Delta x,L_y,t_n) && \text{for } 2\le i\le N_x+1
\end{aligned}$$

- Calculate $\left\{u^{n+1}(j,i)\right\}$ at internal points

$$\begin{aligned}
u^{n+1}(j,i)=u^n(j,i)&+\frac{\Delta t}{(\Delta x)^2}\left(u^n(j,i+1)-2u^n(j,i)+u^n(j,i-1)\right)\\
&+\frac{\Delta t}{(\Delta y)^2}\left(u^n(j+1,i)-2u^n(j,i)+u^n(j-1,i)\right)
\end{aligned}$$

for $2 \le i \le N_x +1$, $2 \le j \le N_y +1$

- Repeat the process until we get to time T.

BTCS method:

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

$$u_{i,j}^0=f(x_i,y_j)$$

$$u_{0,j}^{n+1}=g(0,y_j,t_{n+1}),\quad u_{(N_x+1),j}^{n+1}=g(L_x,y_j,t_{n+1})$$

$$u_{i,0}^{n+1}=g(x_i,0,t_{n+1}),\quad u_{i,(N_y+1)}^{n+1}=g(x_i,L_y,t_{n+1})$$

We write it in the theoretical matrix-vector form

$$u^{n+1}=u^n+\Delta t A_x u^{n+1}+\Delta t A_y u^{n+1}+\Delta t b^{n+1}$$

==> $\quad (I-\Delta t A_x-\Delta t A_y)u^{n+1}=u^n+\Delta t b^{n+1}$

==> $\quad u^{n+1}=(I-\Delta t A_x-\Delta t A_y)\backslash(u^n+\Delta t b^{n+1})$ solution of a linear system

where

$u^{n+1}=\left\{u_{i,j}^{n+1}\right\}$: theoretical vector containing values of u at internal points.

$N_{xy} = N_x \times N_y$ = total number of internal points in the rectangle

A_x: theoretical matrix corresponding to difference operator D_x^2.

$$D_x^2\left\{u_{i,j}^{n+1}\right\}=\frac{1}{(\Delta x)^2}\left[u_{i+1,j}^{n+1}-2u_{i,j}^{n+1}+u_{i-1,j}^{n+1}\right]$$

with zero boundary conditions: $u_{0,j}^{n+1}=u_{(N_x+1),j}^{n+1}=u_{i,0}^{n+1}=u_{i,(N_y+1)}^{n+1}=0$.

Note: Here the zero boundary conditions are set for the sole purpose of defining matrix A_x. They are not the real boundary conditions in the IBVP.

A_y: theoretical matrix corresponding to difference operator D_y^2.

$$D_y^2\left\{u_{i,j}^{n+1}\right\}=\frac{1}{(\Delta y)^2}\left[u_{i,j+1}^{n+1}-2u_{i,j}^{n+1}+u_{i,j-1}^{n+1}\right]$$

with zero boundary conditions: $u_{0,j}^{n+1}=u_{(N_x+1),j}^{n+1}=u_{i,0}^{n+1}=u_{i,(N_y+1)}^{n+1}=0$.

b^{n+1} : vector containing the effects of the *real* boundary conditions in IBVP.

Remarks:

- FTCS is much easier to implement. It does not require the construction of Matlab matrices A_x and A_y. It does not require the solution of a linear system.
- In the implementation of BTCS, we first need to set up Matlab vector u^n, which involves arranging 2D internal grid points into a long vector.
- To set up the linear system in the implicit BTCS, we need to construct Matlab matrices A_x and A_y. The construction involves mapping the neighboring points (i, j), (i+1, j), (i−1, j), (i, j+1), (i, j−1) on the 2D grid to their corresponding indices in the 1D vector u^{n+1}.
- The real boundary conditions in the IBVP are taken care of in vector b^{n+1}.

Summary:

- To implement FTCS, we include boundary points in $\{u\}$, and we update/calculate $\{u\}$ directly. There is no need to solve a linear system.
- To implement BTCS, we include only internal points in $\{u\}$, and we need to set up the linear system $(I-\Delta t A_x-\Delta t A_y)u^{n+1}=u^n+\Delta t b^{n+1}$.
- In the 1-D case, the linear system is much simpler than that of the 2-D case.