AM213B Numerical Methods for the Solution of Differential Equations

Lecture 12

List of topics in this lecture

- 2D IBVP of the heat equation, implementation of 2D FTCS and 2D BTCS
- Crank-Nicolson method, alternating direction implicit method (ADI)
- Numerical solution of hyperbolic PDEs, upwind method, downwind method, FTCS method, Lax-Friedrichs method, local truncation errors and stability of these methods, characteristics propagation, boundary conditions in IBVP

Review

Lax equivalence theorem: Consistency + Stability = Convergence

Method of lines (MOL), 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.	After MOL (method of lines) discretization, a PDE becomes an ODE system. *The stiffness of the system increases* as numerical resolution is refined.
As Δt is refined, eventually all methods will be well behaved for small Δt. We use L-stable ODE solvers to avoid tiny Δt.	As $(\Delta x, \Delta t)$ is refined, a constraint on $\Delta t/(\Delta x)^2$ is needed to ensure the numerical stability of explicit methods. We use L-stable ODE solvers to get out of this constraint.

Procedure of von Neumann stability analysis

- Substitute $u_i^n = \rho^n \exp(\sqrt{-1}\,\xi i \Delta x)$ into the numerical method to calculate ρ.
- 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).

Two-dimensional IBVP of the heat equation

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

FTCS method:

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

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

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

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

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. In each time step, we do two things:

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

Remark: We don't need to write $\left\{u^n(j,i)\right\}$ as a long vector.

BTCS method

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

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

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

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

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

BTCS in matrix-vector form

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

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

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

Remark:

- We need to write $\{u^{n+1}(j,i)\}$ as a long vector in the linear system.
- We need to construct matrices A_x and A_y and vector b^{n+1} in the linear system.

Mapping between 2D array and 1D vector

In the linear system, we need to write $\{u^{n+1}(j,i)\}$ as a long vector. We list all internal points by scanning each column of $\{u^{n+1}(j,i)\}$, storing $\{u_{i,j}^{n+1}\}$.

We write out the mapping between the Matlab 2D index and 1D index.

$$\underbrace{(i,j)}_{\text{2D index}} \xleftrightarrow{\text{Mapping}} \underbrace{k}_{\text{1D index}}$$

$$(i,j) \rightarrow k: \qquad k(i,j) = (i-1)N_y + j$$

$$k \rightarrow (i,j): \qquad \begin{cases} i(k) = \text{floor}((k-1)/N_y) + 1 \\ j(k) = (k-1) - N_y \text{floor}((k-1)/N_y) + 1 \end{cases}$$

The 1D Matlab vector u is

$$\left\{ u_{1D}^n(k) = u_{2D}^n(j(k), i(k)),\ k = 1, 2, \ldots, N_{xy} \right\}, \qquad N_{xy} \equiv N_x N_y$$

In Matlab, u1D = reshape(u2D, Nx*Ny, 1);

Constructing matrix A_x corresponding to $D_x^2\{u_{i,j}\} = \dfrac{1}{(\Delta x)^2}\left[u_{i+1,j} - 2u_{i,j} + u_{i-1,j}\right]$

Row k of A_x has at most 3 non-zero entries.

- map k to $(i(k), j(k))$
- two neighboring points in finite difference are $(i(k)-1, j(k))$, $(i(k)+1, j(k))$
- if $(i(k)-1, j(k))$ is still an internal point, map it to $k_{\text{Left}} \equiv k(i(k)-1, j(k))$
- if $(i(k)+1, j(k))$ is still an internal point, map it to $k_{\text{Right}} \equiv k(i(k)+1, j(k))$
- $A_x(k,k) = -2/(\Delta x)^2, \quad A_x(k, k_{\text{Left}}) = 1/(\Delta x)^2, \quad A_x(k, k_{\text{Right}}) = 1/(\Delta x)^2$

For 2D problems, matrix A_x needs to be stored as a sparse matrix.

Example:

$N_x = N_y = 400$ ==> $N_{xy} \equiv N_x N_y = 160000$

a 160000 × 160000 matrix is too big to store as a dense matrix.

Constructing matrix A_y corresponding to $D_y^2\left\{u_{i,j}\right\} = \frac{1}{(\Delta x)^2}\left[u_{i,j+1} - 2u_{i,j} + u_{i,j-1}\right]$

Row k of A_y has at most 3 non-zero entries.

- map k to $(i(k), j(k))$
- two neighboring points in finite difference are $(i(k), j(k)-1)$, $(i(k), j(k)+1)$
- if $(i(k), j(k)-1)$ is still an internal point, map it to $k_{\text{Down}} \equiv k(i(k), j(k)-1)$
- if $(i(k), j(k)+1)$ is still an internal point, map it to $k_{\text{Up}} \equiv k(i(k), j(k)+1)$
- $A_y(k,k) = -2/(\Delta x)^2$, $A_y(k, k_{\text{Down}}) = 1/(\Delta x)^2$, $A_y(k, k_{\text{Up}}) = 1/(\Delta x)^2$

Calculating vector b^{n+1} containing the effect of boundary conditions

Start with matrix version $\{b_{i,j}\} = 0$, stored in Matlab matrix $\{b(\text{j, i})\} = 0$.

Left boundary affects $\{b_{1,j}\}$, stored in $\{b(\text{j}, 1)\}$

$b(\text{j}, 1) = b(\text{j}, 1) + g_{\text{L}}(j\Delta y, (\text{n}+1)\Delta \text{t})/(\Delta x)^2$

Right boundary affects $\{b_{N_x,j}\}$, stored in $\{b(\text{j, Nx})\}$

$b(\text{j, Nx}) = b(\text{j, Nx}) + g_{\text{R}}(j\Delta y, (\text{n}+1)\Delta t)/(\Delta x)^2$

Bottom boundary affects $\{b_{i,1}\}$, stored in $\{b(1, \text{i})\}$

$b(1, \text{i}) = b(1, \text{i}) + g_{\text{B}}(i\Delta x, (\text{n}+1)\Delta t)/(\Delta y)^2$

Top boundary affects $\{b_{i,N_y}\}$, stored in $\{b(\text{j, Nx})\}$

$b(\text{Ny, i}) = b(\text{Ny, i}) + g_{\text{T}}(i\Delta x, (\text{n}+1)\Delta t)/(\Delta y)^2$

Map 2D array $\{b(\text{j, i})\}$ to 1D vector $\{b(\text{k})\}$

$$\left\{b_{\text{1D}}^{n+1}(k) = u_{\text{2D}}(j(k), i(k)),\ k = 1, 2, \ldots, N_{xy}\right\}, \qquad N_{xy} \equiv N_x N_y$$

In Matlab, `b1D = reshape(b2D, Nx*Ny, 1);`

Calculating vector u^{n+1} containing u at internal points at time t_{n+1}.

Once we obtain matrices A_x and A_y and vector b^{n+1}, we calculate vector u^{n+1}

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

Then the process is repeated in each time step.

Remarks:

- Matrices A_x and A_y only need to be calculated once and then are used in all time steps. Vector b^{n+1} needs to be calculated in each time step.
- We keep evolving 1D vector u forward in time until we need to output its 2D array version. The 2D array u^n gives a better description of function $u(x, y, t_n)$.

$$\left\{ u_{2D}^n(j,i) = u_{1D}^n((i-1)N_y + j),\ j=1,2,\ldots,N_y,\ i=1,2,\ldots,N_x \right\}$$

In Matlab, u2D = reshape(u1D, Ny, Nx);

Next study the Crank-Nicolson method for solving the 2D heat equation, which has the second order accuracy both in space and in time.

Crank-Nicolson method

$$u^{n+1} = u^n + \Delta t A_x \left(\frac{u^{n+1} + u^n}{2} \right) + \Delta t A_y \left(\frac{u^{n+1} + u^n}{2} \right) + \Delta t \left(\frac{b^{n+1} + b^n}{2} \right)$$

To calculate u^{n+1} from u^n, we need to solve a sparse linear system

$$\underbrace{\left(I - \frac{\Delta t}{2} A_x - \frac{\Delta t}{2} A_y \right)}_{\equiv B} u^{n+1} = \underbrace{\left(I + \frac{\Delta t}{2} A_x + \frac{\Delta t}{2} A_y \right)}_{\equiv B_2} u^n + \Delta t \underbrace{\left(\frac{b^{n+1} + b^n}{2} \right)}_{\equiv \tilde{b}^{n+1/2}}$$

==> $$u^{n+1} = B \backslash \left(B_2 u^n + \Delta t\, \tilde{b}^{n+1/2} \right)$$

On a numerical grid of 400×400, matrix B is 160000×160000, a huge matrix.

In the Crank-Nicolson method, the linear system is sparse. Nevertheless, it is still a huge linear system and its sparse structure is not easy to accommodate (as we saw in the implementation of BTCS). We like to replace the huge linear system with a collection of small tridiagonal linear systems. We now discuss such an approach.

Alternating direction implicit method (ADI)

Strategy:

i) divide each time step into two half steps,

ii) in each half step, use implicit difference along only one direction (x or y), and

iii) alternate the implicit direction in two half steps.

$$\begin{cases} u^{n+1/2} = u^n + \underbrace{\frac{\Delta t}{2} A_x u^{n+1/2}}_{\text{Implicit}} + \underbrace{\frac{\Delta t}{2} A_y u^n}_{\text{Explicit}} + \frac{\Delta t}{2}(b_x^{n+1/2} + b_y^n) \\ u^{n+1} = u^{n+1/2} + \underbrace{\frac{\Delta t}{2} A_x u^{n+1/2}}_{\text{Explicit}} + \underbrace{\frac{\Delta t}{2} A_y u^{n+1}}_{\text{Implicit}} + \frac{\Delta t}{2}(b_x^{n+1/2} + b_y^{n+1}) \end{cases}$$

where

A_x : matrix corresponding to difference operator D_x^2

A_y : matrix corresponding to difference operator D_y^2

$b_x^{n+1/2}$: vector containing the effects of boundary conditions in $D_x^2 u^{n+1/2}$,

b_y^{n+1} : vector containing the effects of boundary conditions in $D_y^2 u^{n+1}$,

The first half step $u^n \to u^{n+1/2}$

It is 1st order in time; it is implicit in x only; it is explicit in y.

In this half step, we only need to solve a collection of small tridiagonal linear systems, each corresponding to the discretization of D_x^2 at a fixed y.

The second half step $u^{n+1/2} \to u^{n+1}$

It is 1st order in time; it is implicit in y only; it is explicit in x.

In this half step, we only need to solve a collection of small tridiagonal linear systems, each corresponding to the discretization of D_y^2 at a fixed x.

The combined step $u^n \to u^{n+1}$

It is 2nd order in time and 2nd order in space!

This can be seen by writing out the combined step.

$$u^{n+1} = u^n + \Delta t A_x u^{n+1/2} + \Delta t A_y \frac{u^{n+1} + u^n}{2} + \Delta t \left(b_x^{n+1/2} + \frac{b_y^n + b_y^{n+1}}{2} \right)$$

Note: every term on the RHS is evaluated either directly or effectively at $t_{n+1/2}$.

Stability of numerical methods for solving the 2D heat equation

FTCS method:

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

It is stable if and only if $\dfrac{\Delta t}{(\Delta x)^2} + \dfrac{\Delta t}{(\Delta y)^2} \le \dfrac{1}{2}$.

Tool: 2D von Neumann stability analysis (see Appendix A).

BTCS method:

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

It is unconditionally stable.

Alternating direction implicit method (ADI)

$$\begin{cases} u^{n+1/2} = u^{n} + \underbrace{\frac{\Delta t}{2} D_x^2 u^{n+1/2}}_{\text{Implicit}} + \underbrace{\frac{\Delta t}{2} D_y^2 u^{n}}_{\text{Explicit}} \\ u^{n+1} = u^{n+1/2} + \underbrace{\frac{\Delta t}{2} D_y^2 u^{n+1/2}}_{\text{Explicit}} + \underbrace{\frac{\Delta t}{2} D_y^2 u^{n+1}}_{\text{Implicit}} \end{cases}$$

It is unconditionally stable (see Appendix B).

Numerical solution of hyperbolic PDEs

We first review a simple initial value problem.

IVP of $u_t + au_x = 0$

$$\begin{cases} u_t + au_x = 0, \qquad a = const \\ u(x,0) = u_0(x) \end{cases}$$

The initial condition in parametric form

$$\begin{cases} x_0(s) = s \\ t_0(s) = 0 \\ u(x_0(s), t_0(s)) = u_0(s) \end{cases} \qquad s \text{ is a parameter}$$

Characteristics:

$$\begin{cases} \dfrac{dt}{d\tau} = 1, \quad t(0) = t_0(s) = 0 \\ \dfrac{dx}{d\tau} = a, \quad x(0) = x_0(s) = s \\ \dfrac{du}{d\tau} = 0, \quad u(0) = u_0(s) \end{cases} \qquad s \text{ is a parameter; } \tau \text{ is the time along C-line.}$$

Solution in parametric form, as a function of (s, τ):

$$\begin{cases} t(s,\tau)=\tau \\ x(s,\tau)=s+a\tau \\ u(s,\tau)=u_0(s) \end{cases}$$

We like to express u as a function of (x, t).

The inverse mapping of $(t(s, \tau), x(s, \tau))$:

$$\begin{cases} \tau=t \\ s=x-at \end{cases}$$

Solution u, as a function of (x, t):

$$u(x,t)=u_0(s(x,t))=u_0(x-at)$$

Propagation of solution:

$a > 0$, $u(x, t) = u_0(x - a\,t)$ is a traveling wave to the right.

(Draw the characteristics and wave propagation in the x-t plane)

$a < 0$, $u(x, t) = u_0(x - a\,t)$ is a traveling wave to the left.

(Draw the characteristics and wave propagation in the x-t plane)

In all real applications, we solve the PDE in a finite domain.

IBVP of $u_t + au_x = 0$

For $a > 0$ (wave propagating to the right)

$$\begin{cases} u_t+au_x=0\,, & 0<x<L \\ u(x,0)=u_0(x)\,, & 0<x<L \\ u(0,t)=g(t) \end{cases}$$

Note: A boundary condition is imposed only at $x = 0$, not at $x = L$.

For $a < 0$ (wave propagating to the left)

$$\begin{cases} u_t+au_x=0\,, & 0<x<L \\ u(x,0)=u_0(x)\,, & 0<x<L \\ u(L,t)=g(t) \end{cases}$$

Note: A boundary condition is imposed only at $x = L$, not at $x = 0$.

Remark:

- A boundary condition is imposed only at the boundary where the characteristics go into the computational domain.

- No boundary condition is imposed at the boundary where the characteristics go out of the computational domain.

We consider the case of $a > 0$ and study several numerical methods.

The upwind method

Exact solution: $u(x, t) = u_0(x - at)$

Information is coming from the left. We go to the left side to get information.

$$u_x(x_i, t_n) \approx \frac{u_i^n - u_{i-1}^n}{\Delta x}, \qquad u_t(x_i, t_n) \approx \frac{u_i^{n+1} - u_i^n}{\Delta t}$$

$$u_t = -a u_x \qquad ==> \qquad \frac{u_i^{n+1} - u_i^n}{\Delta t} \approx -a \frac{u_i^n - u_{i-1}^n}{\Delta x}$$

The upwind method:

$$u_i^{n+1} = u_i^n - ar(u_i^n - u_{i-1}^n), \qquad r = \frac{\Delta t}{\Delta x}, \qquad a > 0$$

This is called the upwind method since we go upstream of characteristics propagation to seek information for calculating u^{n+1}.

Notice the new definition of r for numerical solution of hyperbolic PDEs.

Local truncation error of upwind method:

$$e_i^n(\Delta x, \Delta t) = \Delta t \cdot O(\Delta t + \Delta x)$$

It is 1st order in time and 1st order in space.

Stability of upwind method (an alternative analysis):

We write the upwind method as

$$u_i^{n+1} = (1 - ar) u_i^n + ar u_{i-1}^n$$

- The case of $0 \le (a\, r) \le 1$

$$|u_i^{n+1}| \le (1 - ar)|u_i^n| + ar|u_{i-1}^n| \le (1 - ar)\sup_k |u_k^n| + (ar)\sup_k |u_k^n| = \sup_k |u_k^n|$$

Since this is valid at any i, we have

$$\sup_k |u_k^{n+1}| \le \sup_k |u_k^n|$$

==> The method is stable for $(a\, r) \le 1$.

- The case of $(a\, r) > 1$

We try $u_i^n = \rho^n (-1)^i$. Substituting it into the upwind method, we get

$$\rho=(1-ar)+ar(-1)=1-2ar$$

==> $|\rho|=|1-2ar|=2ar-1>1$

==> The method is unstable for $(a\,r) > 1$.

The analysis above also shows that the method is unstable for $a < 0$.

$|\rho| = |1-2ar| = 1+(-2ar) > 1$ for $a < 0$

Therefore, the upwind method is stable if and only if $0 \le (a\,r) \le 1$.

von Neumann stability analysis (the universal tool)

We substitute $u_i^n=\rho^n e^{\sqrt{-1}\xi i\Delta x}$ into $u_i^{n+1}=(1-ar)u_i^n+(ar)u_{i-1}^n$.

$$\rho^{n+1}e^{\sqrt{-1}\xi i\Delta x}=(1-ar)\rho^n e^{\sqrt{-1}\xi i\Delta x}+(ar)\rho^n e^{\sqrt{-1}\xi(i-1)\Delta x}$$

==> $\rho=(1-ar)+(ar)e^{-\sqrt{-1}\xi\Delta x}$

==> $\rho=(1-ar)+(ar)\cos(\xi\Delta x)-\sqrt{-1}(ar)\sin(\xi\Delta x)$

==> $|\rho|^2=(1-ar)^2+2(1-ar)(ar)\cos(\xi\Delta x)+(ar)^2$

$$=\left((1-ar)^2+2(1-ar)(ar)+(ar)^2\right)+2(1-ar)(ar)\left(\cos(\xi\Delta x)-1\right)$$

$$=1-4(1-ar)(ar)\sin^2\left(\frac{\xi\Delta x}{2}\right)$$

- The case of $0 \le (a\,r) \le 1$

  $|\rho|^2 \le 1$ for all ξ

- The case of $(a\,r) > 1$

  At $\xi\Delta x/2 = \pi/2$, $\sin(\xi\Delta x/2) = 1$ and $|\rho|^2 = 1+4(ar-1)(ar) > 1$

- The case of $(a\,r) < 0$

  At $\xi\Delta x/2 = \pi/2$, $\sin(\xi\Delta x/2) = 1$ and $|\rho|^2 = 1+4(1-ar)(-ar) > 1$

In summary, the upwind method is stable if and only if $0 \le (a\,r) \le 1$.

The downwind method:

$$u_i^{n+1}=u_i^n-ar(u_{i+1}^n-u_i^n),\qquad r=\frac{\Delta t}{\Delta x},\qquad a>0$$

This is called the downwind method since we go downstream of characteristics propagation to seek information for calculating u^{n+1}.

Local truncation error of downwind method:

$$e_i^n(\Delta x, \Delta t) = \Delta t \cdot O(\Delta t + \Delta x)$$

It is 1st order in time and 1st order in space.

Stability of downwind method:

We write the downwind method as

$$u_i^{n+1} = (1 + ar)u_i^n - ar\, u_{i+1}^n$$

We carry out the von Neumann stability analysis

...

$$|\rho|^2 = 1 + 4(1 + ar)(ar)\sin^2\left(\frac{\xi \Delta x}{2}\right)$$

...

In summary, the downwind method is unstable for $a > 0$.

The FTCS method

We use central difference in the spatial dimension.

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

Local truncation error of FTCS method:

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

It is 1st order in time and 2nd order in space.

von Neumann stability analysis of FTCS method:

We substitute $u_i^n = \rho^n e^{\sqrt{-1}\xi i \Delta x}$ into $u_i^{n+1} = u_i^n - \frac{ar}{2}(u_{i+1}^n - u_{i-1}^n)$.

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

==> $\quad \rho = 1 - \frac{ar}{2}\left[e^{\sqrt{-1}\xi \Delta x} - e^{-\sqrt{-1}\xi \Delta x} \right]$

$$= 1 - \sqrt{-1}(ar)\sin(\xi \Delta x)$$

==> $\quad |\rho|^2 = 1 + (ar)^2 \sin^2(\xi \Delta x)$

At $\xi\Delta x = \pi/2$, $\sin(\xi\Delta x) = 1$ and

$$|\rho|^2 = 1 + (ar)^2 > 1$$

==> The FTCS method is unstable for any $(a\, r)$.

In summary, the FTCS method is unconditionally unstable.

Next we introduce the Lax-Friedrichs method, which is a stable variation of FTCS.

Lax-Friedrichs method

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

Note that the u_i^n in FTCS is replaced by $(u_{i+1}^n + u_{i-1}^n)/2$ in Lax-Friedrichs.

Local truncation error of Lax-Friedrichs method:

$$e_i^n(\Delta x, \Delta t) = \Delta t \cdot O(\Delta t + \Delta x) \qquad \text{for fixed } r$$

It is 1st order in time and 1st order in space.

Stability of Lax-Friedrichs method (an alternative analysis)

We write the Lax-Friedrichs method as

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

- For $|a\, r| \le 1$,

$$|u_i^{n+1}| \le \frac{1}{2}(1+ar)|u_{i+1}^n| + \frac{1}{2}(1-ar)|u_{i-1}^n|$$

$$\le \frac{1}{2}(1+ar)\sup_k |u_k^n| + \frac{1}{2}(1-ar)\sup_k |u_k^n| = \sup_k |u_k^n|$$

Since this is valid at any i, we have

$$\sup_k |u_k^{n+1}| \le \sup_k |u_k^n|$$

==> The method is stable for $|a\, r| \le 1$.

- For $|a\, r| > 1$, we try $u_i^n = \rho^n \left(\sqrt{-1}\right)^i$.

Substituting it into the Lax-Friedrichs method, we get

$$\rho = \frac{1}{2}(1+ar)\sqrt{-1} + \frac{1}{2}(1-ar)\left(\sqrt{-1}\right)^{-1} = \frac{1}{2}(1+ar)\sqrt{-1} - \frac{1}{2}(1-ar)\sqrt{-1} = (ar)\sqrt{-1}$$

==> $|\rho| = |ar| > 1$

==> The method is unstable for $|a\, r| > 1$.

In summary, the Lax-Friedrichs method is stable if and only if $|a r| \le 1$.

Remarks:

Lax-Friedrichs method has the properties below

- It is consistent.
- It is stable for $|a r| \le 1$.
- It has a unified form for both the case of $a > 0$ and the case of $a < 0$.

But Lax-Friedrichs is only 1st order in time and in space.

We want a method having these properties with 2nd order in time and space.

Appendix A Stability of the FTCS method for solving the 2D heat equation

2D FTCS method:

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

We substitute $u_{i,j}^{n} = \rho^n e^{\sqrt{-1}(\xi i \Delta x + \eta j \Delta y)}$ into the 2D FTCS method.

$$\rho^{n+1} e^{\sqrt{-1}(\xi i \Delta x + \eta j \Delta y)} = \rho^{n} e^{\sqrt{-1}(\xi i \Delta x + \eta j \Delta y)} + \rho^{n} e^{\sqrt{-1}(\xi i \Delta x + \eta j \Delta y)} \frac{\Delta t}{(\Delta x)^2}\left(e^{\sqrt{-1}\xi \Delta x} - 2 + e^{-\sqrt{-1}\xi \Delta x}\right)$$
$$+\rho^{n} e^{\sqrt{-1}(\xi i \Delta x + \eta j \Delta y)} \frac{\Delta t}{(\Delta y)^2}\left(e^{\sqrt{-1}\eta \Delta y} - 2 + e^{-\sqrt{-1}\eta \Delta y}\right)$$

==> $$\rho = 1 + \frac{\Delta t}{(\Delta x)^2}\left(e^{\sqrt{-1}\xi \Delta x} - 2 + e^{-\sqrt{-1}\xi \Delta x}\right) + \frac{\Delta t}{(\Delta y)^2}\left(e^{\sqrt{-1}\eta \Delta y} - 2 + e^{-\sqrt{-1}\eta \Delta y}\right)$$

==> $$\rho = 1 - 4\frac{\Delta t}{(\Delta x)^2}\sin^2\left(\frac{\xi \Delta x}{2}\right) - 4\frac{\Delta t}{(\Delta y)^2}\sin^2\left(\frac{\eta \Delta y}{2}\right)$$

Assuming the ratios $\Delta t/(\Delta x)^2$ and $\Delta t/(\Delta y)^2$ remain constant as $(\Delta x, \Delta y, \Delta t) \to 0$.

The 2D FTCS method is stable if and only if $\rho > -1$ for all (ξ, η)

if and only if $4\frac{\Delta t}{(\Delta x)^2}\sin^2\left(\frac{\xi \Delta x}{2}\right) + 4\frac{\Delta t}{(\Delta y)^2}\sin^2\left(\frac{\eta \Delta y}{2}\right) \le 2$ for all (ξ, η)

if and only if $\frac{\Delta t}{(\Delta x)^2} + \frac{\Delta t}{(\Delta y)^2} \le \frac{1}{2}$.

Appendix B Stability of the ADI method for solving the 2D heat equation

Alternating direction implicit method (ADI)

$$\begin{cases} u^{n+1/2} = u^n + \underbrace{\frac{\Delta t}{2} D_x^2 u^{n+1/2}}_{\text{Implicit}} + \underbrace{\frac{\Delta t}{2} D_y^2 u^n}_{\text{Explicit}} \\ u^{n+1} = u^{n+1/2} + \underbrace{\frac{\Delta t}{2} D_y^2 u^{n+1/2}}_{\text{Explicit}} + \underbrace{\frac{\Delta t}{2} D_y^2 u^{n+1}}_{\text{Implicit}} \end{cases}$$

We substitute $u_{i,j}^n = \rho^n e^{\sqrt{-1}(\xi i \Delta x + \eta j \Delta y)}$ and $u_{i,j}^{n+1/2} = \omega \rho^n e^{\sqrt{-1}(\xi i \Delta x + \eta j \Delta y)}$ into the first half.

$$\omega = 1 + \omega \frac{\Delta t}{(\Delta x)^2}\left(e^{\sqrt{-1}\xi \Delta x} - 2 + e^{-\sqrt{-1}\xi \Delta x}\right) + \frac{\Delta t}{(\Delta y)^2}\left(e^{\sqrt{-1}\eta \Delta y} - 2 + e^{-\sqrt{-1}\eta \Delta y}\right)$$

$$\Longrightarrow \quad \omega\left(1 + 4\frac{\Delta t}{(\Delta x)^2}\sin^2\left(\frac{\xi \Delta x}{2}\right)\right) = \left(1 - 4\frac{\Delta t}{(\Delta y)^2}\sin^2\left(\frac{\eta \Delta x}{2}\right)\right) \qquad \text{(E01A)}$$

We substitute $u_{i,j}^{n+1/2} = \omega \rho^n e^{\sqrt{-1}(\xi i \Delta x + \eta j \Delta y)}$ and $u_{i,j}^{n+1} = \rho^{n+1} e^{\sqrt{-1}(\xi i \Delta x + \eta j \Delta y)}$ into the second half.

$$\rho = \omega + \omega \frac{\Delta t}{(\Delta x)^2}\left(e^{\sqrt{-1}\xi \Delta x} - 2 + e^{-\sqrt{-1}\xi \Delta x}\right) + \rho \frac{\Delta t}{(\Delta y)^2}\left(e^{\sqrt{-1}\eta \Delta y} - 2 + e^{-\sqrt{-1}\eta \Delta y}\right)$$

$$\Longrightarrow \quad \rho\left(1 + 4\frac{\Delta t}{(\Delta y)^2}\sin^2\left(\frac{\eta \Delta x}{2}\right)\right) = \omega\left(1 - 4\frac{\Delta t}{(\Delta x)^2}\sin^2\left(\frac{\xi \Delta x}{2}\right)\right) \qquad \text{(E01B)}$$

Combining (E01A) and (E01B), we obtain

$$\rho = \frac{\left(1 - 4\frac{\Delta t}{(\Delta x)^2}\sin^2\left(\frac{\xi \Delta x}{2}\right)\right)}{\left(1 + 4\frac{\Delta t}{(\Delta x)^2}\sin^2\left(\frac{\xi \Delta x}{2}\right)\right)} \cdot \frac{\left(1 - 4\frac{\Delta t}{(\Delta y)^2}\sin^2\left(\frac{\eta \Delta x}{2}\right)\right)}{\left(1 + 4\frac{\Delta t}{(\Delta y)^2}\sin^2\left(\frac{\eta \Delta x}{2}\right)\right)}$$

It is straightforward to show that | ρ | ≤ 1 for all (ξ, η).