Lecture 6

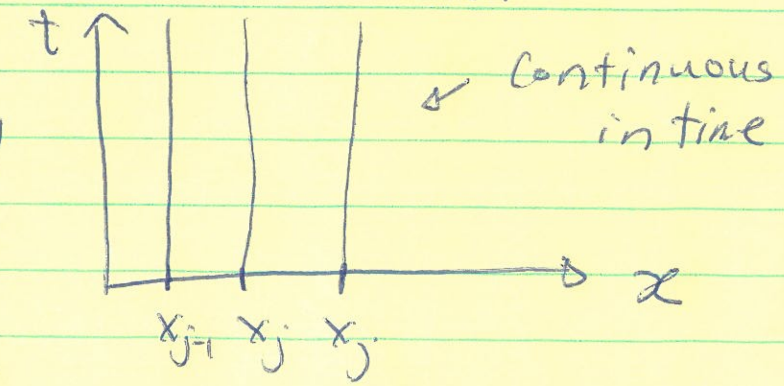
The method of lines and general 1D parabolic equations

- Method of lines
- general 1D linear PDEs
- divergence form
- nonlinear PDEs
- MOVFIDM

The method of lines

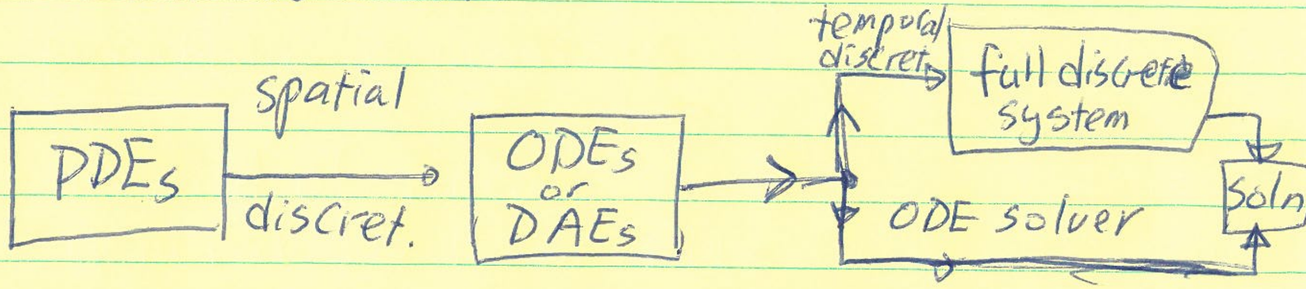
- PDEs are discretized simultaneously in time and space.
- Rothe's method: in time first and in space later
- The Method of lines (MOL): in space first and in time later

Advantages of MOL:



- Temporal and spatial discretizations are treated separately. Special attention can be paid to each of them.

- Existing software on ODE integration can be used.



The heat equation (Model problem)

$$\begin{cases} \dfrac{\partial u}{\partial t} = \dfrac{\partial^2 u}{\partial x^2} \\ u(0,t) = 0, \ u(1,t) = 0 \end{cases}$$

Let

$$U_j(t) \approx u(x_j, t)$$

$$\begin{cases} \dfrac{dU_j(t)}{dt} = \dfrac{1}{\Delta x^2}\left[U_{j+1}(t) - 2U_j(t) + U_{j-1}(t)\right] \\ \qquad\qquad j = 1, \cdots, J-1 \\ \left.\begin{aligned} U_0(t) &= 0 \\ U_J(t) &= 0 \end{aligned}\right\} \Rightarrow \begin{cases} \dfrac{d}{dt}U_0(t) = 0 \\ \dfrac{d}{dt}U_J(t) = 0 \end{cases} \end{cases}$$

$$M \frac{d\vec{u}}{dt} = \vec{f}(t, \vec{u})$$

(ODE or DAE system)

They can be solved using existing ODE/DAE codes.

Or: they can be discretized in time to give a full discrete system. For example, Using Implicit Euler, we get

$$\begin{cases} \dfrac{u_j^{n+1} - u_j^n}{\Delta t} = \dfrac{1}{\Delta x^2}\left[u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}\right] \\ u_0^{n+1} = 0 \\ u_J^{n+1} = 0. \end{cases}$$

General 1D linear PDEs

$$\begin{cases} \dfrac{\partial u}{\partial t} = a(x,t)\dfrac{\partial^2 u}{\partial x^2} + b(x,t)\dfrac{\partial u}{\partial x} + c(x,t)u + f(x,t) \\ \qquad\qquad x \in (0,1) \\ u(0,t) = g_0(t) \\ u(1,t) = g_1(t) \end{cases}$$

$$\begin{cases} \dfrac{du_j}{dt} = \dfrac{a_j}{\Delta x^2}(u_{j+1} - 2u_j + u_{j-1}) + \dfrac{b_j}{2\Delta x}(u_{j+1} - u_{j-1}) \\ \qquad + c_j u_j + f_j \\ u_0(t) = g_0(t) \\ u_J(t) = g_1(t) \end{cases}$$

- Treatment of Neumann BC

$$\frac{\partial u}{\partial x}(0,t) = g_0(t)$$

- firstorder approximation

$$\boxed{\frac{u_1 - u_0}{\Delta x} = g_0(t)} \qquad O(\Delta x)$$

- 2nd order approximation — fictitious point

$$x_{-1} = x_0 - \Delta x$$

$$\begin{cases} \dfrac{u_1 - u_{-1}}{2\Delta x} = g_0(t) \\[2ex] \dfrac{du_0}{dt} = \dfrac{a_0}{\Delta x^2}\left(u_1 - 2u_0 + u_{-1}\right) + \dfrac{b_0}{2\Delta x}\left(u_1 - u_{-1}\right) \end{cases}$$

$$+ c_0 u_0 + f_0$$

Use these eqns to eliminate u_{-1}, we get

$$\boxed{\begin{aligned} \frac{du_0}{dt} &= \frac{a_0}{\Delta x^2}\left(u_1 - 2u_0 + \underline{u_1 - 2\Delta x\, g_0(t)}\right) \\ &\quad + \frac{b_0}{2\Delta x}\left(u_1 - \left(u_1 - 2\Delta x\, g_0(t)\right)\right) \\ &\quad + c_0 u_0 + f_0 \end{aligned}} \qquad O(\Delta x^2)$$

- 2nd order approximation - extrapolation

$$
\begin{array}{r|l}
4\times & u_1 = u_0 + \Delta x\, u_x + \Delta x^2 u_{xx} + O(\Delta x^3) \\
- & u_2 = u_0 + 2\Delta x\, u_x + 4\Delta x^2 u_{xx} + O(\Delta x^3)
\end{array}
$$

$$4u_1 - u_2 = 3u_0 + 2\Delta x\, u_x + O(\Delta x^3)$$

$$u_x = \frac{4u_1 - u_2 - 3u_0}{2\Delta x} + O(\Delta x^2)$$

Thus:

$$\boxed{\frac{4u_1 - u_2 - 3u_0}{2\Delta x} = g_0(t)} \qquad O(\Delta x^2)$$

Divergence form of PDEs

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left(a(x,t) \frac{\partial u}{\partial x} \right)$$

better to use this form

$$\frac{\partial u}{\partial t} = a(x,t)\frac{\partial^2 u}{\partial x^2} + \frac{\partial a}{\partial x}\frac{\partial u}{\partial x}$$

$$\frac{du_j}{dt} = \frac{1}{\Delta x^2}\left[a_{j+1/2}(u_{j+1} - u_j) - a_{j-\frac{1}{2}}(u_j - u_{j-1}) \right]$$

- Symmetric
- diagonally dominant.

Nonlinear PDEs

Example – Burgers' equation

$$\frac{\partial u}{\partial t} = \underbrace{\varepsilon \frac{\partial^2 u}{\partial x^2}}_{\text{diffusion}} \underbrace{- u \frac{\partial u}{\partial x}}_{\text{convection}}$$

$\varepsilon = \frac{1}{Re}$ Reynolds #

$$\Rightarrow \frac{du_j}{dt} = \frac{\varepsilon}{\Delta x^2}\left(u_{j+1} - 2u_j + u_{j-1}\right) - u_j \frac{u_{j+1} - u_{j-1}}{2\Delta x}$$

$$\frac{\partial u}{\partial t} = \varepsilon \frac{\partial^2 u}{\partial x^2} - \frac{\partial}{\partial x}\left(\frac{1}{2}u^2\right)$$

$$\Rightarrow \frac{du_j}{dt} = \frac{\varepsilon}{\Delta x^2}\left(u_{j+1} - 2u_j + u_{j-1}\right) - \frac{1}{4\Delta x}\left(u_{j+1}^2 - u_{j-1}^2\right)$$

▶ MOVFDM

—— For 1D system

$$\begin{cases} F(t, x, u, u_x, u_t) = \frac{\partial}{\partial x} G(t, x, u, u_x, u_t) & a < x < b \\ B_l(t, x, u, u_x, u_t) = 0 & x = a \\ B_r(t, x, u, u_x, u_t) = 0 & x = b \end{cases}$$

The user needs to provide F, G, B_l and B_r.

—— Central FD discretization

—— ODE15i for time integration

Example Burgers' eqn

$$\begin{cases} \dfrac{\partial u}{\partial t} = \varepsilon \dfrac{\partial^2 u}{\partial x^2} - u \dfrac{\partial u}{\partial x} & x \in (0,1) \\ u(0,t) = 0 \\ u(1,t) = 0 \end{cases}$$

① $F = \dfrac{\partial u}{\partial t}$, $G = \varepsilon \dfrac{\partial u}{\partial x} - \frac{1}{2} u^2$

$B_\ell = u_t - 0$, $B_r = u_t - 0$.

② $F = \dfrac{\partial u}{\partial t} + u \dfrac{\partial u}{\partial x}$, $G = \varepsilon \dfrac{\partial u}{\partial x}$

$B_\ell = u_t - 0$, $B_r = u_t - 0$.

40

```
function sec1_2_burgersFDM(jmax)
%
% example driver for Burgers' equation in section 1.2.
% it calls movfdm().


%
% Copyright (C) 2010 Weizhang Huang and Robert D. Russell
% all rights reserved.
%
% This program is provided "as is", without warranty of any
kind.
% Permission is hereby granted, free of charge, to use this
program
% for personal, research, and education purposes. Distribution
or use
% of this program for any commercial purpose is permissible
% only by direct arrangement with the copyright owner.
%

cpu0=clock;

   % job = 1 for solution
   % job = 2 for mesh trajectories

   job = 1;

   %jmax = 41;
   npde = 1;
   nn = npde*jmax;

   x=zeros(jmax,1);
   u=zeros(npde,jmax);

% define initial solution

   x=linspace(0,1,jmax)';
   u(1,:)=(sin(2*pi*x)+0.5*sin(pi*x))';
   xt=zeros(jmax,1);
   ut=zeros(npde,jmax);

% call the moving mesh function
% monitor = 0 (fixed mesh), 1, 2, 3, 4, or 5
% mmpde = 4, 5, 6, 7 (mmpde = 7 ==> modified MMPDE5)
% alpha_type = 1, 2, or 3: (integral def, integral def with
flooring, or alpha = constant)

   monitor = 3;
   mmpde = 7;
   alpha_type = 2;
   alpha = 1.0;
   reltol=1e-6; abstol=1e-4;

   if job==1 % for solution

      tspan = [0 0.2 0.4 0.6 0.8 1.0];

[t,X,U]=movfdm(npde,jmax,tspan,x,xt,u,ut,@PDE_F,@PDE_G,@BC_L,@BC
_R,...
```

41

```
[],monitor,reltol,abstol,[],mmpde,alpha_type,alpha);
      fprintf('\n cpu time used = %e \n', etime(clock,cpu0));
      % output the solution
      N=size(t,1);
          figure
          labels={};
      marks='+osdv^<>';
      lines1='--:-';
      lines2='-  .';
      colors='mkbgrc';
      for n=1:N
         u=U(:,:,n);
         x=X(:,n);
         hold on
         labels{n} = ['t = ' num2str(t(n)) ];
         style=['-' marks(1+rem(n,8))];
         plot(x,u(1,:)',style);%,'LineWidth',2);
      end
      hold off
      legend(labels{:});
      xlabel('x');
      ylabel('u');
      axis([0 1 -1 2]);
          box on;

   else % for trajectories

      tspan = [0 1.0];

  X,U]=movfdm(npde,jmax,tspan,x,xt,u,ut,@PDE_F,@PDE_G,@BC_L,@BC
_R,...

[],monitor,reltol,abstol,[],mmpde,alpha_type,alpha);
          fprintf('\n cpu time used = %e \n',
etime(clock,cpu0));
      plot(X',t);
          axis([0 1 0 1]);
      xlabel('x')
      ylabel('t')
   end

fprintf('cpu time used = %e \n', etime(clock,cpu0));

% ----------------------------------------------------

function f=PDE_F(t,x,u,ux,ut)
  f(1,:) = ut(1,:);

function f=PDE_G(t,x,u,ux,ut)
  f(1,:) = 1e-4*ux(1,:)-u(1,:).*u(1,:)*0.5;

function f=BC_L(t,x,u,ux,ut)
  f(1) = ut(1)-0.0;

function f=BC_R(t,x,u,ux,ut)
  (1) = ut(1)-0.0;
```

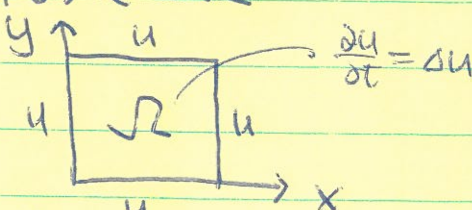
Lecture 7

FTCS scheme for 2D parabolic PDEs

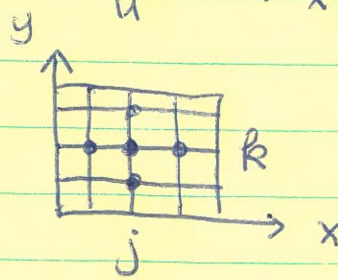
- Model problem
- Mesh
- FTCS scheme
- local truncation error
- Convergence
- stability

The 2D Model problem

$$\begin{cases} \dfrac{\partial u}{\partial t} = \dfrac{\partial^2 u}{\partial x^2} + \dfrac{\partial^2 u}{\partial y^2} \equiv \Delta u & (x,y) \in \Omega = (0,1)\times(0,1) \\ u(x,y,t) = g(x,y,t) & (x,y) \in \partial\Omega \\ u(x,y,0) = u^0(x,y) & \end{cases}$$



Mesh – Cartisian grid



J subintervals of equal length in x direction

$$x_j = j\Delta x, \; j = 0,1,\cdots,J$$
$$\Delta x = \frac{1}{J}$$

K subintervals of equal length in y direction

$$y_k = k\Delta y, \; k = 0,1,\cdots,J$$
$$\Delta y = \frac{1}{K}$$

N subintervals of equal length in t direction

$$t_n = n\Delta t, \; n = 0,\cdots,N \quad \Delta t = \frac{T}{N}$$

FTCS scheme

$$u_{j,k}^n \approx u(x_j, y_k, t_n)$$

$$\begin{cases} \dfrac{u_{j,k}^{n+1} - u_{j,k}^n}{\Delta t} = \dfrac{1}{\Delta x^2}\left(u_{j+1,k}^n - 2u_{j,k}^n + u_{j-1,k}^n\right) \\ \qquad\qquad + \dfrac{1}{\Delta y^2}\left(u_{j,k+1}^n - 2u_{j,k}^n + u_{j,k-1}^n\right) \\ \qquad\qquad\qquad j = 1,\dots,J-1 \\ \qquad\qquad\qquad k = 1,\dots,K-1 \\ u_{j,k}^{n+1} = g(x_j, y_k, t_{n+1}), \quad j=0 \text{ or } j=J,\ k=0,\dots,K \\ \qquad\qquad\qquad k=0 \text{ or } k=K,\ j=1,\dots,J-1 \end{cases}$$

Local truncation error

The error equation:

$$\frac{e_{j,k}^{n+1} - e_{j,k}^n}{\Delta t} = \frac{1}{\Delta x^2}\delta_x^2 e_{j,k}^n + \frac{1}{\Delta y^2}\delta_y^2 e_{j,k}^n - \tau_{j,k}^n$$

$$\tau_{j,k}^n = \frac{u(x_j, y_k, t_{n+1}) - u(x_j, y_k, t_n)}{\Delta t} - \frac{1}{\Delta x^2}\delta_x^2 u(x_j, y_k, t_n) - \frac{1}{\Delta y^2}\delta_y^2 u(x_j, y_k, t_n)$$

$$= \frac{\Delta t}{2} u_{tt}(x_j, y_k, \eta_n) - \frac{\Delta x^2}{12} u_{xxxx}(\xi_j, y_k, t_n) - \frac{\Delta y^2}{12} u_{yyyy}(x_j, \sigma_k, t_n)$$

$$= O(\Delta t) + O(\Delta x^2) + O(\Delta y^2)$$

▶ Convergence in L^∞ norm

$$\|e^n\|_\infty = \max_{j,k} |e^n_{j,k}|$$

If
$$\Delta t \le \frac{1}{\frac{2}{\Delta x^2} + \frac{2}{\Delta y^2}}$$

then
$$\|e^n\|_\infty \le T\left(\frac{\Delta t}{2} M_{tt} + \frac{\Delta x^2}{12} M_{xxxx} + \frac{\Delta y^2}{12} M_{yyyy}\right)$$

▶ Stability

— Fourier stability analysis

$$u^n_{j,k} = (\lambda)^n e^{im\pi x_j} e^{i\ell\pi y_k}$$

$$m = 1, \cdots, J-1$$
$$\ell = 1, \cdots, K-1$$

$$\frac{\lambda - 1}{\Delta t} = \frac{1}{\Delta x^2}\left(e^{im\pi\Delta x} - 2 + e^{-im\pi\Delta x}\right) + \frac{1}{\Delta y^2}\left(e^{i\ell\pi\Delta y} - 2 + e^{-i\ell\pi\Delta y}\right)$$

$$\frac{\lambda - 1}{\Delta t} = \frac{1}{\Delta x^2}\left(2\cos(m\pi\Delta x) - 2\right) + \frac{1}{\Delta y^2}\left(2\cos(\ell\pi\Delta y) - 2\right)$$

$$\frac{\lambda - 1}{\Delta t} = \frac{-4}{\Delta x^2} \sin^2\left(\frac{m\pi\Delta x}{2}\right) - \frac{4}{\Delta y^2} \sin^2\left(\frac{\ell\pi\Delta y}{2}\right)$$

$$\lambda = 1 - \frac{4\Delta t}{\Delta x^2} \sin^2\left(\frac{m\pi\Delta x}{2}\right) - \frac{4\Delta t}{\Delta y^2} \sin^2\left(\frac{\ell\pi\Delta y}{2}\right)$$

$|\lambda| \leq 1$ for all $m = 1, \cdots, J-1$
$\ell = 1, \cdots, K-1$

$$\Rightarrow \quad \boxed{\Delta t \leq \frac{1}{\frac{2}{\Delta x^2} + \frac{2}{\Delta y^2}}} \quad \text{(CFL)}$$

(Condition)

- L^∞ stability analysis

$\|u^n\|_\infty \leq \|u^0\|_\infty$ for all n

$$\Rightarrow \quad \Delta t \leq \frac{1}{\frac{2}{\Delta x^2} + \frac{2}{\Delta y^2}} \quad \text{(CFL)}.$$

#

Lecture 8

BTCS scheme for 2D parabolic PDEs

- BTCS scheme
- local truncation error, stability, convergence
- Linear system for $u_{j,k}^{n+1}$ and natural ordering
- Direct solution
- ~~Solution~~ iterative methods ~~Techniques~~ for large scale sparse systems

BTCS scheme

$$\begin{cases} \dfrac{\partial u}{\partial t} = \dfrac{\partial^2 u}{\partial x^2} + \dfrac{\partial^2 u}{\partial y^2} & \text{in } \Omega \\ u = g & \text{on } \partial\Omega \end{cases}$$

$$\frac{u_{j,k}^{n+1} - u_{j,k}^{n}}{\Delta t} = \frac{1}{\Delta x^2}\delta_x^2 u_{j,k}^{n+1} + \frac{1}{\Delta y^2}\delta_y^2 u_{j,k}^{n+1}$$

$$j = 1, \cdots, J-1 \qquad k = 1, \cdots, K-1$$

$$\delta_x^2 u_{j,k}^{n+1} = u_{j+1,k}^{n+1} - 2u_{j,k}^{n+1} + u_{j-1,k}^{n+1}$$

$$\delta_y^2 u_{j,k}^{n+1} = u_{j,k+1}^{n+1} - 2u_{j,k}^{n+1} + u_{j,k-1}^{n+1}$$

$$u_{j,k}^{n+1} = g(x_j, y_k, t_{n+1}) \qquad (x_j, y_k) \in \partial\Omega$$

Consistency, stability, and convergence

$$\tau_{jk}^{n} = O(\Delta t) + O(\Delta x^2) + O(\Delta y^2)$$

Stability: (L^∞ and Fourier analysis)

on constraint on choice of Δt

$\Rightarrow$ unconditionally stable

Convergence:

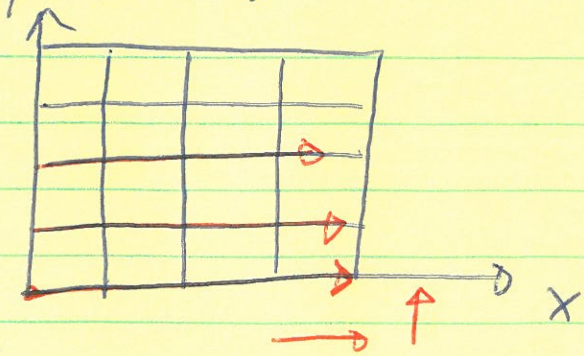
$$e_{jk}^{n} = O(\Delta t) + O(\Delta x^2) + O(\Delta y^2)$$

Linear system for u_{jk}^{n+1} and natural ordering

- BTCS scheme is implicit.

  $\Rightarrow$ $\{u_{jk}^{n+1},\ j=1,\cdots,J-1,\ k=1,\cdots,K-1\}$ must be solved simultaneously.

- natural ordering for unknown variables (and for equations):



we need to convert 2D arrays u_{jk} into 1D array $\vec{u}$ so that we can express the linear system in a matrix form.

$$(j,k) \longrightarrow (J+1)k+j$$

$$u_{j,k} \longrightarrow \vec{u}_{(J+1)k+j}$$

$$\{u_{j,k}\} \longrightarrow (u_{0,0}, u_{1,0}, \cdots, u_{J,0}, u_{0,1}, \cdots, u_{J,1}, \cdots$$
$$\cdots, u_{0,K}, \cdots, u_{J,K})$$

- matrix form

$$A\vec{u}^{n+1} = \vec{b}$$

For $(x_j, y_k) \in \Omega$:

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

$$A_{(j,k),(j,k)} = 1 + \frac{2\Delta t}{\Delta x^2} + \frac{2\Delta t}{\Delta y^2}$$

$$A_{(j,k),(j+1,k)} = -\frac{\Delta t}{\Delta x^2}$$

$$A_{(j,k),(j-1,k)} = -\frac{\Delta t}{\Delta x^2}$$

$$A_{(j,k),(j,k+1)} = -\frac{\Delta t}{\Delta y^2}$$

$$A_{(j,k),(j,k-1)} = -\frac{\Delta t}{\Delta y^2}$$

$$\vec{b}_{(j,k)} = u_{j,k}^{n}$$

For $(x_j, y_k) \in \partial\Omega$:

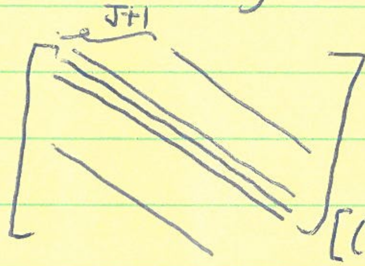
$$u_{j,k}^{n+1} = g(x_j, y_k, t_{n+1})$$

$$A_{(j,k),(j,k)} = 1$$

$$\vec{b}_{(j,k)} = g(x_j, y_k, t_{n+1})$$

Properties of A:

(i) A is pentadiagonal

$A =$ 

$[(J+1)(K+1)] \times [(J+1)\times(K+1)]$

(ii) Sparse:

$$\frac{\#\text{ of non-zero entries}}{\#\text{ of entries}} \approx \frac{5(JK)}{(JK)^2} = \frac{5}{JK}$$

small when J, K large

(iii) diagonally dominant

$$|A_{j,j}| \geq \sum_{k \neq j} |A_{j,k}|$$

Direct solution of the linear system

— Direct solvers (full matrix)

Gaussian elimination, LU decomposition

Cost $$O((JK)^3) = O(J^3K^3)$$

— Direct solvers (banded matrix)

band width $= J+1$

Cost $$O(J^2(JK)) = O(J^3K)$$

$J=K$, # of operations/sec $= 10^8$

J	$O(J^3K^3)$	CPU (sec)	$O(J^3K)$	CPU (sec)
10	10^6	10^{-2}	10^4	10^{-4}
100	10^{12}	10^4	10^8	1
1000	10^{18}	10^{10}	10^{12}	10^4

— Direct solvers (sparse matrix)

Example: UMFPACK

Iterative methods for large scale sparse systems:

— Convential iterative methods

$$\{u_{j,k}^{n+1,0}\} \longrightarrow \{u_{j,k}^{n+1,1}\} \rightarrow \cdots$$

$$\{u_{j,k}^{n+1,i} \longrightarrow u_{j,k}^{n+1,\infty} \quad \text{as } i \rightarrow +\infty.$$

Jacobi iteration:

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

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

$$j = 1, \cdots, J-1$$
$$k = 1, \cdots, K-1$$

Gauss-seidel iteration

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

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

$$j = 1, \cdots, J-1$$
$$k = 1, \cdots, K-1$$

SOR (successive over relaxation)

$$\begin{cases} -\frac{\Delta t}{\Delta x^2}\left(u_{j+1,k}^{n+1,i} + u_{j-1,k}^{n+1,i+1}\right) - \frac{\Delta t}{\Delta y^2}\left(u_{j,k+1}^{n+1,i} + u_{j,k-1}^{n+1,i+1}\right) \\ \quad + \left(1 + \frac{2\Delta t}{\Delta x^2} + \frac{2\Delta t}{\Delta y^2}\right) u_{j,k}^{n+1,*} = u_{j,k}^{n} \\ u_{j,k}^{n+1,i+1} = (1-\omega) u_{j,k}^{n+1,i} + \omega u_{j,k}^{n+1,*} \end{cases}$$

$$j = 1, \cdots, J-1$$
$$k = 1, \cdots, K-1$$

initial guess:

$$u_{j,k}^{n+1,0} = u_{j,k}^{n}$$

— Krylov Subspace methods + preconditioning techniques

Conjugate gradient (CG)

CR, GMRES, BiCG, BiCGstab, BiCGstab2,

preconditioning techniques:

ILU (incomplete LU decomposition)

and many others

— Multigrid (or multilevel) methods

algebraic multigrid methods

(MG, AMG).

#

Lecture 9

ADI methods

ADI: alternating direction implicit

- ADI method
- ~~order~~ accuracy ~~analysis~~
- Stability analysis
- implementation
- limitations

ADI method

Consider BTCS scheme

$$\frac{u_{j,k}^{n+1} - u_{j,k}^{n}}{\Delta t} = \frac{1}{\Delta x^2}\delta_x^2 u_{j,k}^{n+1} + \frac{1}{\Delta y^2}\delta_y^2 u_{j,k}^{n+1} + O(\Delta t) + O(\Delta x^2) + O(\Delta y^2)$$

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

Introduce: increment:

$$u_{j,k}^{n+1} = u_{j,k}^{n} + \Delta u_{j,k}^{n}$$

Then, we have

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

Notice that

$$\left(1-\frac{\Delta t}{\Delta x^2}\delta_x^2\right)\left(1-\frac{\Delta t}{\Delta y^2}\delta_y^2\right)$$
$$=1-\frac{\Delta t}{\Delta x^2}\delta_x^2-\frac{\Delta t}{\Delta y^2}\delta_y^2+\frac{\Delta t^2}{\Delta x^2\Delta y^2}\delta_x^2\delta_y^2$$

Thus

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

drop this term?

$$+O(\Delta t^2)+O(\Delta t\Delta x^2)+O(\Delta t\Delta y^2)$$

$\Rightarrow$ ADI scheme:

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

(i) The system is much more econonic to solve than the BTCS scheme
(ii) Good stability
(iii) Good accuracy

Accuracy

We just need to estimate the term dropped:

$$\frac{\Delta t^2}{\Delta x^2 \Delta y^2} \delta_x^2 \delta_y^2 \Delta u_{j,k}^n$$

$$\Delta u_{j,k}^n = u_{j,k}^{n+1} - u_{j,k}^n = O(\Delta t)$$

$$\frac{1}{\Delta x^2} \delta_x^2 u \approx u_{xx} + O(\Delta x^2) = O(1)$$

$$\Rightarrow \quad \frac{\Delta t^2}{\Delta x^2 \Delta y^2} \delta_x^2 \delta_y^2 \Delta u_{j,k}^n = O(\Delta t^3)$$

this order is higher than the local truncation error $O(\Delta t^2) + O(\Delta t \Delta x^2) + O(\Delta t \Delta y^2)$.

Thus, the local truncation error for the ADI scheme is

$$\tau_{j,k}^n = O(\Delta t) + O(\Delta x^2) + O(\Delta y^2)$$

Stability

$$u_{j,k}^{n} = (\lambda)^{n} e^{im\pi x_j} e^{il\pi y_k}$$

$$\delta_x^2 u_{j,k}^{n} = -4\sin^2\left(\frac{m\pi}{2}\Delta x\right) u_{j,k}^{n}$$

$$\delta_y^2 u_{j,k}^{n} = -4\sin^2\left(\frac{l\pi}{2}\Delta y\right) u_{j,k}^{n}$$

$$\xi \equiv \sin^2\left(\frac{m\pi}{2}\Delta x\right), \quad \eta \equiv \sin^2\left(\frac{l\pi}{2}\Delta y\right)$$

$$\cancel{(\lambda - 1)\left(1 + \frac{4\Delta t}{\Delta x^2}\sin^2\left(\frac{m\pi}{2}\Delta x\right)\right)} \qquad \underline{0 < \xi, \eta < 1}$$

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

$$= -\frac{4\Delta t}{\Delta x^2}\xi - \frac{4\Delta t}{\Delta y^2}\eta$$

$$\lambda = \frac{1 + \frac{16\Delta t^2}{\Delta x^2 \Delta y^2}\xi\eta}{\left(1 + \frac{4\Delta t}{\Delta x^2}\xi\right)\left(1 + \frac{4\Delta t}{\Delta y^2}\eta\right)}$$

$$|\lambda| \leq 1$$

unconditionallly stable!

Implementation:

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

Define

$$v_{j,k}=\left(1-\frac{\Delta t}{\Delta y^2}\delta_y^2\right)\Delta u_{j,k}^n$$

then

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

Step 1 X-sweeps

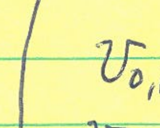
for $k=1,\cdots,K-1$

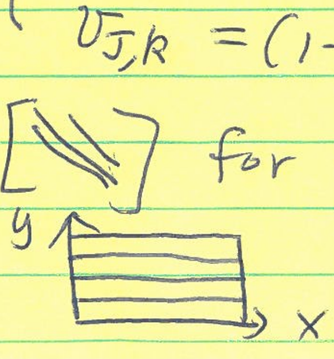
Solve

$$\begin{cases}\left(1-\frac{\Delta t}{\Delta x^2}\delta_x^2\right)v_{j,k}=\left(\frac{\Delta t}{\Delta x^2}\delta_x^2+\frac{\Delta t}{\Delta y^2}\delta_y^2\right)u_{j,k}^n\\ v_{0,k}=\left(1-\frac{\Delta t}{\Delta y^2}\delta_y^2\right)\Delta u_{0,k}^n\\ v_{J,k}=\left(1-\frac{\Delta t}{\Delta y^2}\delta_y^2\right)\Delta u_{J,k}^n\end{cases}$$

or

$$\begin{cases}-\frac{\Delta t}{\Delta x^2}v_{j+1,k}+\left(1+\frac{2\Delta t}{\Delta x^2}\right)v_{j,k}-\frac{\Delta t}{\Delta x^2}v_{j-1,k}=\left(\frac{\Delta t}{\Delta x^2}\delta_x^2+\frac{\Delta t}{\Delta y^2}\delta_y^2\right)u_{j,k}^n\\ \qquad j=1,\cdots,J-1\\ v_{0,k}=\left(1-\frac{\Delta t}{\Delta y^2}\delta_y^2\right)\left(u_{0,k}^{n+1}-u_{0,k}^n\right)\\ v_{J,k}=\left(1-\frac{\Delta t}{\Delta y^2}\delta_y^2\right)\left(u_{J,k}^{n+1}-u_{J,k}^n\right)\end{cases}$$

 for $(v_{0,k}, v_{1,k}, \cdots, v_{J,k})$



Step 2 y-sweeps

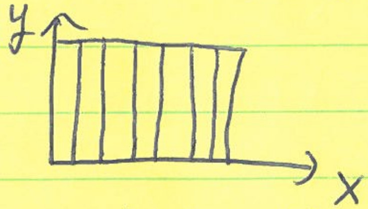
For $j = 1, \cdots, J-1$:

$$\begin{cases} \left(1 - \frac{\Delta t}{\Delta y^2}\delta_y^2\right)\Delta u_{j,k}^n = v_{j,k} \\ \Delta u_{j,0}^n = u_{j,0}^{n+1} - u_{j,0}^n \\ \Delta u_{j,K}^n = u_{j,K}^{n+1} - u_{j,K}^n \end{cases}$$

or

$$\begin{cases} -\frac{\Delta t}{\Delta y^2}\Delta u_{j,k+1}^n + \left(1 + \frac{2\Delta t}{\Delta y^2}\right)\Delta u_{j,k}^n - \frac{\Delta t}{\Delta y^2}\Delta u_{j,k-1}^n = v_{j,k} \\ \qquad k = 1, \cdots, K-1 \\ \Delta u_{j,0}^n = u_{j,0}^{n+1} - u_{j,0}^n \\ \Delta u_{j,K}^n = u_{j,K}^{n+1} - u_{j,K}^n \end{cases}$$

$[\diagdown\!\diagdown\!\diagdown]$ for $(\Delta u_{j,0}^n, \Delta u_{j,1}^n, \cdots, \Delta u_{j,K}^n)$



Step 3

Finally update the soln:

$$u_{j,k}^{n+1} = u_{j,k}^n + \Delta u_{j,k}^n$$

▶ limitations of ADI

- work only for regular, simple domain
- work only for FD mesh (Cartisian mesh)
- doesnot work for problems with mixed derivatives

$\left(1 - \frac{\Delta t}{\Delta x^2}\delta_x^2\right)\left(1 - \frac{\Delta t}{\Delta y^2}\delta_y^2\right)$ can be used as a preconditioner. #

$\boxed{\text{Lecture 10}}$

MOL for general 2D problems

- Model PDE
- general PDEs
- PDEs in divergence form

MOL for model PDE

$$\boxed{\text{PDEs}} \xrightarrow[\text{discretizats}]{\text{spatial}} \boxed{\begin{array}{c}\text{ODE}\\ \text{systems}\end{array}} \xrightarrow[\text{solver}]{\text{ODE}} \boxed{\text{Solns}}$$

- adavantage is that we can focus on the spatial discretization and leave time discrezation, solution of nonlinear equations, and efficient solution of large scale (sparse) systems to the ODE solver:

$$u_{j,k}(t) \approx u(x_j, y_k, t)$$

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$$

$$\frac{d u_{j,k}}{dt} = \frac{1}{\Delta x^2}\delta_x^2 u_{j,k} + \frac{1}{\Delta y^2}\delta_y^2 u_{j,k}$$

$$(x_j, y_k) \in \Omega$$

+ ~~approximation~~ $u_{j,k} = g(x_j, y_k, t)$. $(x_j, y_k) \in \partial\Omega$

$$\Rightarrow \quad M \frac{d\vec{u}}{dt} = \vec{f}(t, \vec{u})$$

$$u_{j,k} \to \vec{u}_{(j,k)} = \vec{u}_{(J+1)k+j}$$

General 2D parabolic PDEs

$$\frac{\partial u}{\partial t} = d_{11} \frac{\partial^2 u}{\partial x^2} + 2 d_{12} \frac{\partial^2 u}{\partial x \partial y} + d_{22} \frac{\partial^2 u}{\partial y^2} + b_1 \frac{\partial u}{\partial x} + b_2 \frac{\partial u}{\partial y} + c u + f(t, x, y)$$

$$\begin{aligned} \frac{d}{dt} u_{j,k} = {} & d_{11,jk} \frac{1}{\Delta x^2} \delta_x^2 u_{j,k} + d_{22,jk} \frac{1}{\Delta y^2} \delta_y^2 u_{j,k} \\ & + 2 d_{12,jk} \frac{1}{4 \Delta x \Delta y} \big(u_{j+1,k+1} - u_{j+1,k-1} \\ & \quad - u_{j-1,k+1} + u_{j-1,k-1} \big) \\ & + \frac{b_1}{2\Delta x} \left(u_{j+1,k} - u_{j-1,k} \right) + \frac{b_2}{2\Delta y} \left(u_{j,k+1} - u_{j,k-1} \right) \\ & + c_{j,k} u_{j,k} + f_{j,k} \end{aligned}$$

Nonlinear equations

2D Burgers' equation

$$\frac{\partial u}{\partial t} = \varepsilon \frac{\partial^2 u}{\partial x^2} + \varepsilon \frac{\partial^2 u}{\partial y^2} - u \frac{\partial u}{\partial x} - u \frac{\partial u}{\partial y}$$

$$\frac{du_{j,k}}{dt} = \frac{\varepsilon}{\Delta x^2}\delta_x^2 u_{j,k} + \frac{\varepsilon}{\Delta y^2}\delta_y^2 u_{j,k}$$
$$- \frac{1}{4\Delta x}\left(u_{j+1,k}^2 - u_{j-1,k}^2\right)$$
$$- \frac{1}{4\Delta y}\left(u_{j,k+1}^2 - u_{j,k-1}^2\right)$$

PDEs in divergence form

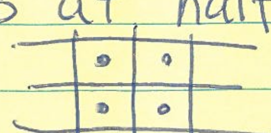
$$\nabla = \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix}$$

$$\frac{\partial u}{\partial t} = \nabla\cdot(\mathbb{D}\nabla u) - \nabla\cdot(\vec{b}u) + cu + f$$

Define diffusion fluxes:

$$q_1 = d_{11}\frac{\partial u}{\partial x} + d_{12}\frac{\partial u}{\partial y}$$
$$q_2 = d_{21}\frac{\partial u}{\partial x} + d_{22}\frac{\partial u}{\partial y}$$

Approximate these fluxes at half-half points $(j+\frac{1}{2}, k+\frac{1}{2})$:



$$q_{1,j+\frac{1}{2},k+\frac{1}{2}} = d_{11,j+\frac{1}{2},k+\frac{1}{2}}\frac{1}{2\Delta x}\left(u_{j+1,k} - u_{j,k} + u_{j+1,k+1} - u_{j,k+1}\right)$$
$$+ d_{12,j+\frac{1}{2},k+\frac{1}{2}}\frac{1}{2\Delta y}\left(u_{j,k+1} - u_{j,k} + u_{j+1,k+1} - u_{j+1,k}\right)$$

$$q_{2,j+\frac{1}{2},k+\frac{1}{2}} = d_{21,j+\frac{1}{2},k+\frac{1}{2}} \frac{1}{2\Delta x}(\cdot - - \cdot) + d_{22,j+\frac{1}{2},k+\frac{1}{2}} \frac{1}{2\Delta y}(\cdot - - \cdot)$$

Then

$$\nabla \cdot (\mathbb{D} \nabla u)\Big|_{j,k} = \nabla \cdot (\vec{q})\Big|_{j,k}$$

$$= \left(\frac{\partial q_1}{\partial x} + \frac{\partial q_2}{\partial y}\right)_{j,k}$$

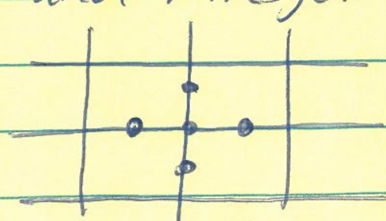
$$\approx \frac{1}{2\Delta x}\Big(q_{1,j+\frac{1}{2},k+\frac{1}{2}} - q_{1,j-\frac{1}{2},k+\frac{1}{2}} + q_{1,j+\frac{1}{2},k-\frac{1}{2}} - q_{1,j-\frac{1}{2},k-\frac{1}{2}}\Big)$$

$$+ \frac{1}{2\Delta y}\Big(q_{2,j+\frac{1}{2},k+\frac{1}{2}} - q_{2,j+\frac{1}{2},k-\frac{1}{2}} + q_{2,j-\frac{1}{2},k+\frac{1}{2}} - q_{2,j-\frac{1}{2},k-\frac{1}{2}}\Big)$$

Define ~~tran~~ convection fluxes:

$$\vec{p} = \vec{b}\, u$$

Approximate convection fluxes at half-integer and integer-half points



$$p_{1,j+\frac{1}{2},k} = b_{1,j+\frac{1}{2},k} \frac{u_{j+1,k} + u_{j,k}}{2}$$

$$p_{2,j,k+\frac{1}{2}} = b_{2,j,k+\frac{1}{2}} \frac{u_{j,k+1} + u_{j,k}}{2}$$

$$\nabla \cdot (\vec{b} u)_{j,k} = \nabla \cdot \vec{p}_{j,k}$$

$$= \left(\frac{\partial p_1}{\partial x} + \frac{\partial p_2}{\partial y}\right)_{j,k}$$

$$\approx \frac{1}{\Delta x}\left(p_{1,j+\frac{1}{2},k} - p_{1,j-\frac{1}{2},k}\right)$$

$$+ \frac{1}{\Delta y}\left(p_{2,j,k+\frac{1}{2}} - p_{2,j,k-\frac{1}{2}}\right)$$

Advantages:

- symmetric
- diagonally dominant

\#