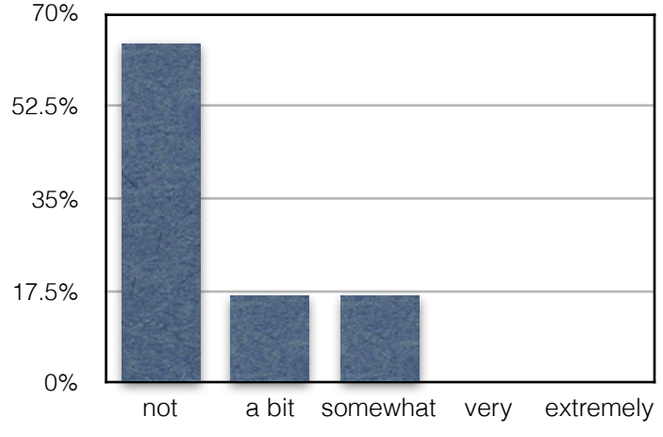
Muddiest Points from Class 02/27

- *"So does Crank-Nicholson require two phi arrays like Point-Jacobi did?"*
  - Not really, since the n-time level is used to calculate the right-hand-side vector d for the Gaussian elimination and the result of Gaussian elimination can be directly copied into the solution variable array (please ignore the comment in the slide 7 of class 14)
  - 2 different arrays for the n and n+1 time levels are required for explicit methods though

- *"I don't understand why do we need to incorporate boundary conditions using ghost cell values when coding Crank-Nicolson method. If we want to build the tridiagonal matrix, we just need to build array a,b,c,d, when do we need to incorporate boundary conditions? "*
- *"To code Crank-Nicolson, do we need to recalculate ghost points after solving for the interior points with the tridiagonal solver?"*
  - the boundary conditions will change the content of the arrays a,b,c, and d.
  - using boundary conditions to set ghost values at least allows the d array to be calculated for the old time level data using the regular interior domain stencil

- *"Is there an easier way to transition from 1D to 2D while coding or is does it have to be done separately each time, i feel that there should be a way to make it universal so 1D or 2D work with a given function but i cant quite seem to get it."*
  - You can use 2D code to run 1D cases, if you allow for different number of elements in the x-direction (M) and y-direction (N).
  - Setting N=1 in a 2D code with zero-Neumann boundary conditions in the y-direction essentially does 1D simulations, but at a higher cost, since all the y-direction terms have to be calculated.
  - it's usually more efficient to just add an if statement outside the mesh-loop to check if N == 1 and then use the 1D formulas



2D Parabolic Equation

What about 2D?

$$\frac{\partial \varphi}{\partial t} = \alpha \left(\frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial^2 \varphi}{\partial y^2} \right)$$

- additional finite difference terms for the y-direction derivative

Example: FTCS

$$\varphi_{j,k}^{n+1} = \varphi_{j,k}^{n} + B_x \left(\varphi_{j+1,k}^{n} - 2\varphi_{j,k}^{n} + \varphi_{j-1,k}^{n} \right) + B_y \left(\varphi_{j,k+1}^{n} - 2\varphi_{j,k}^{n} + \varphi_{j,k-1}^{n} \right)$$

$$B_x = \frac{\alpha \Delta t}{\Delta x^2} \qquad B_y = \frac{\alpha \Delta t}{\Delta y^2}$$

- von Neumann Stability Analysis

$$\varphi_{j,k}^{n} = \rho^n e^{ik_x x_j} e^{ik_y y_k} \qquad \varphi_{j,k}^{n+1} = \rho^{n+1} e^{ik_x x_j} e^{ik_y y_k}$$

$$\varphi_{j\pm1,k}^{n} = \rho^n e^{ik_x x_{j\pm1}} e^{ik_y y_k} \qquad \varphi_{j,k\pm1}^{n} = \rho^n e^{ik_x x_j} e^{ik_y y_{k\pm1}}$$

- substitute in sinusoidal solution:

$$\rho^{n+1} e^{ik_x x_j} e^{ik_y y_k} = \rho^n e^{ik_x x_j} e^{ik_y y_k} + B_x \left(\rho^n e^{ik_x x_{j+1}} e^{ik_y y_k} - 2\rho^n e^{ik_x x_j} e^{ik_y y_k} + \rho^n e^{ik_x x_{j-1}} e^{ik_y y_k} \right)$$
$$+ B_y \left(\rho^n e^{ik_x x_j} e^{ik_y y_{k+1}} - 2\rho^n e^{ik_x x_j} e^{ik_y y_k} + \rho^n e^{ik_x x_j} e^{ik_y y_{k-1}} \right)$$

von Neumann Stability Analysis

$$\frac{\partial \varphi}{\partial t} = \alpha \left(\frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial^2 \varphi}{\partial y^2} \right)$$

$$\rho^{n+1} e^{ik_x x_j} e^{ik_y y_k} = \rho^n e^{ik_x x_j} e^{ik_y y_k} + B_x \left(\rho^n e^{ik_x x_{j+1}} e^{ik_y y_k} - 2\rho^n e^{ik_x x_j} e^{ik_y y_k} + \rho^n e^{ik_x x_{j-1}} e^{ik_y y_k} \right)$$
$$+ B_y \left(\rho^n e^{ik_x x_j} e^{ik_y y_{k+1}} - 2\rho^n e^{ik_x x_j} e^{ik_y y_k} + \rho^n e^{ik_x x_j} e^{ik_y y_{k-1}} \right)$$

$$\Big| : e^{ikx_j} e^{iky_k}$$

$$\rho^{n+1} = \rho^n + B_x \left(\rho^n e^{ik_x \Delta x} - 2\rho^n + \rho^n e^{-ik_x \Delta x} \right) + B_y \left(\rho^n e^{ik_y \Delta y} - 2\rho^n + \rho^n e^{-ik_y \Delta y} \right)$$

$$\rho^{n+1} = \rho^n \left[1 + B_x \left(e^{ik_x \Delta x} + e^{-ik_x \Delta x} - 2 \right) + B_y \left(e^{ik_y \Delta y} + e^{-ik_y \Delta y} - 2 \right) \right]$$

$$\frac{\rho^{n+1}}{\rho^n} = \left[1 + B_x \left(2\cos(k_x \Delta x) - 2 \right) + B_y \left(2\cos(k_y \Delta y) - 2 \right) \right]$$

Stable if $\left| \frac{\rho^{n+1}}{\rho^n} \right| \leq 1$

$$1 + 2B_x \underbrace{\left(\cos(k_x \Delta x) - 1\right)}_{\leq 0} + 2B_y \underbrace{\left(\cos(k_y \Delta y) - 1\right)}_{\leq 0} \leq 1 \quad \checkmark$$

≤ 0

$$\wedge \quad 1 + 2B_x \left(\cos(k_x \Delta x) - 1\right) + 2B_y \left(\cos(k_y \Delta y) - 1\right) \geq -1$$

$$2B_x \left(\cos(k_x \Delta x) - 1\right) + 2B_y \left(\cos(k_y \Delta y) - 1\right) \geq -2$$

$$B_x \underbrace{\left(1 - \cos(k_x \Delta x)\right)}_{0 \ldots 2} + B_y \underbrace{\left(1 - \cos(k_y \Delta y)\right)}_{0 \ldots 2} \leq 1$$

worst case: $2B_x + 2B_y \leq 1$

$$B_x + B_y \leq \frac{1}{2}$$

$$\Delta t \leq \frac{1}{2\alpha \left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2} \right)}$$

if $\Delta x = \Delta y = h$: $\Delta t \leq \frac{1}{4} \frac{h^2}{\alpha}$

Implicit treatment of the parabolic terms

$$\frac{\partial u}{\partial t} = \alpha \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right)$$

- FTCS stability: $\left(\frac{\alpha \Delta t}{\Delta x^2} + \frac{\alpha \Delta t}{\Delta y^2} \right) \leq \frac{1}{2}$
- let's define some shorthands: $d_x = \frac{\alpha \Delta t}{\Delta x^2} \quad d_y = \frac{\alpha \Delta t}{\Delta y^2} \quad \Rightarrow \quad d_x + d_y \leq \frac{1}{2}$
- stability limit is typically very restrictive ⇒ implicit schemes are preferable!
- BTCS:

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

$$\Rightarrow \quad d_x u_{i+1,j}^{n+1} + d_x u_{i-1,j}^{n+1} + d_y u_{i,j+1}^{n+1} + d_y u_{i,j-1}^{n+1} - (2d_x + 2d_y + 1) u_{i,j}^{n+1} = -u_{i,j}^{n}$$

  - penta-diagonal system (block tri-diagonal)
    - not very efficient to solve using Gaussian elimination
    - could use iterative methods, especially V-cycle multigrid methods

ADI-Methods

Alternating **D**irectional **I**mplicit Methods

$$d_x = \frac{\alpha \Delta t}{\Delta x^2} \qquad d_y = \frac{\alpha \Delta t}{\Delta y^2} \qquad d_1 = \frac{d_x}{2} \qquad d_2 = \frac{d_y}{2}$$

Idea:

- we don't really need to find the exact solution to the system
  - even the exact solution will have truncation errors!
  - we can introduce modifications, as long as we preserve the formal order of accuracy!

⇒ splitting methods: ADI

$$d_x u_{i+1,j}^{n+1} + d_x u_{i-1,j}^{n+1} + d_y u_{i,j+1}^{n+1} + d_y u_{i,j-1}^{n+1} - (2d_x + 2d_y + 1) u_{i,j}^{n+1} = -u_{i,j}^{n}$$

- Step 1: implicit only in the x-direction for *Δt/2*

$$\frac{d_x}{2} u_{i+1,j}^{n+1/2} + \frac{d_x}{2} u_{i-1,j}^{n+1/2} + \frac{d_y}{2} u_{i,j+1}^{n} + \frac{d_y}{2} u_{i,j-1}^{n} - (2\frac{d_x}{2} + 1) u_{i,j}^{n+1/2} - 2\frac{d_y}{2} u_{i,j}^{n} = -u_{i,j}^{n}$$

$$-d_1 u_{i+1,j}^{n+1/2} + (1 + 2d_1) u_{i,j}^{n+1/2} - d_1 u_{i-1,j}^{n+1/2} = d_2 u_{i,j+1}^{n} + (1 - 2d_2) u_{i,j}^{n} + d_2 u_{i,j-1}^{n}$$

- Step 2: implicit only in the y-direction for *Δt/2*

$$-d_2 u_{i,j+1}^{n+1} + (1 + 2d_2) u_{i,j}^{n+1} - d_2 u_{i,j-1}^{n+1} = d_1 u_{i+1,j}^{n+1/2} + (1 - 2d_1) u_{i,j}^{n+1/2} + d_1 u_{i-1,j}^{n+1/2}$$

ADI-Methods

$$d_x = \frac{\alpha \Delta t}{\Delta x^2} \qquad d_y = \frac{\alpha \Delta t}{\Delta y^2} \qquad d_1 = \frac{d_x}{2} \qquad d_2 = \frac{d_y}{2}$$

- Step 1: implicit only in the x-direction for *Δt/2*

$$-d_1 u_{i+1,j}^{n+1/2} + (1+2d_1) u_{i,j}^{n+1/2} - d_1 u_{i-1,j}^{n+1/2} = d_2 u_{i,j+1}^{n} + (1-2d_2) u_{i,j}^{n} + d_2 u_{i,j-1}^{n}$$

- Step 2: implicit only in the y-direction for *Δt/2*

$$-d_2 u_{i,j+1}^{n+1} + (1+2d_2) u_{i,j}^{n+1} - d_2 u_{i,j-1}^{n+1} = d_1 u_{i+1,j}^{n+1/2} + (1-2d_1) u_{i,j}^{n+1/2} + d_2 u_{i-1,j}^{n+1/2}$$

- introduce two more shorthands for 2nd derivative, 2nd-order central finite difference operators:

$$\delta_x^2 u_{i,j} = u_{i+1,j} - 2u_{i,j} + u_{i-1,j} \qquad \delta_y^2 u_{i,j} = u_{i,j+1} - 2u_{i,j} + u_{i,j-1}$$

- Step 1:

$$-d_1 (u_{i+1,j}^{n+1/2} - 2u_{i,j}^{n+1/2} + u_{i-1,j}^{n+1/2}) + u_{i,j}^{n+1/2} = d_2 (u_{i,j+1}^{n} - 2u_{i,j}^{n} + u_{i,j-1}^{n}) + u_{i,j}^{n}$$

$$(1 - d_1 \delta_x^2) u_{i,j}^{n+1/2} = (1 + d_2 \delta_y^2) u_{i,j}^{n}$$

Step 1: $(1 - d_1 \delta_x^2) u_{i,j}^{n+1/2} = (1 + d_2 \delta_y^2) u_{i,j}^{n}$

Step 2: $(1 - d_2 \delta_y^2) u_{i,j}^{n+1} = (1 + d_1 \delta_x^2) u_{i,j}^{n+1/2}$

ADI

Crank-Nicholson

$$d_x = \frac{\alpha \Delta t}{\Delta x^2} \qquad d_y = \frac{\alpha \Delta t}{\Delta y^2} \qquad d_1 = \frac{d_x}{2} \qquad d_2 = \frac{d_y}{2}$$

$$\delta_x^2 u_{i,j} = u_{i+1,j} - 2u_{i,j} + u_{i-1,j} \qquad \delta_y^2 u_{i,j} = u_{i,j+1} - 2u_{i,j} + u_{i,j-1}$$

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

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

$$\left[1 - \left(d_1 \delta_x^2 + d_2 \delta_y^2\right)\right] u_{i,j}^{n+1} = \left[1 + \left(d_1 \delta_x^2 + d_2 \delta_y^2\right)\right] u_{i,j}^n$$

- How does this compare to ADI? eliminate $u^{n+1/2}$

Step 1: $(1 - d_1 \delta_x^2) u_{i,j}^{n+1/2} = (1 + d_2 \delta_y^2) u_{i,j}^n \qquad \big| \cdot (1 + d_1 \delta_x^2)$

Step 2: $(1 - d_2 \delta_y^2) u_{i,j}^{n+1} = (1 + d_1 \delta_x^2) u_{i,j}^{n+1/2} \qquad \big| \cdot (1 - d_1 \delta_x^2)$

add

$$(1 + d_1 \delta_x^2)(1 - d_1 \delta_x^2) u_{i,j}^{n+1/2} + (1 - d_1 \delta_x^2)(1 - d_2 \delta_y^2) u_{i,j}^{n+1} = (1 + d_1 \delta_x^2)(1 + d_2 \delta_y^2) u_{i,j}^n + (1 - d_1 \delta_x^2)(1 + d_1 \delta_x^2) u_{i,j}^{n+1/2}$$

$$(1 - d_1 \delta_x^2)(1 - d_2 \delta_y^2) u_{i,j}^{n+1} = (1 + d_1 \delta_x^2)(1 + d_2 \delta_y^2) u_{i,j}^n$$

$$(1 - (d_1 \delta_x^2 + d_2 \delta_y^2) + d_1 d_2 \delta_x^2 \delta_y^2) u_{i,j}^{n+1} = (1 + (d_1 \delta_x^2 + d_2 \delta_y^2) + d_1 d_2 \delta_x^2 \delta_y^2) u_{i,j}^n$$

⇒ the same as Crank-Nicholson, except for the $d_1 d_2 \delta_x^2 \delta_y^2$ terms!

but $d_1 d_2 \delta_x^2 \delta_y^2$ is $O(h^4)$, thus a higher order term compared to the truncation error, $O(h^2)$

⇒ we can neglect these terms!

⇒ up to the order of Crank-Nicholson, ADI & Crank-Nicholson are identical!

ADI

- ADI belongs to a class of methods called “Approximate Factorization”
- Idea: split multi-dimensional FDEs into series of FDEs that can be solved in tri-diagonal form

ADI is the approximate factorization of Crank-Nicholson

- ADI is simply a technique to solve Crank-Nicholson in 2D (and 3D)
- Results of ADI and Crank-Nicholson are different
  - BUT: the difference is < $O(h^2)$!!!
  - The error we make by using ADI instead of Crank-Nicholson is not relevant since it will go to zero as fast or faster than our Taylor-Series discretization errors

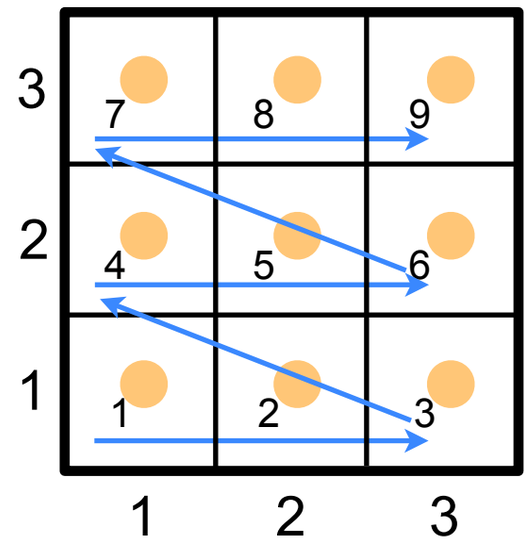
How to code ADI

$$d_x = \frac{\alpha \Delta t}{\Delta x^2} \quad d_y = \frac{\alpha \Delta t}{\Delta y^2} \quad d_1 = \frac{d_x}{2} \quad d_2 = \frac{d_y}{2} \quad \frac{\partial \varphi}{\partial t} = \alpha \left(\frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial^2 \varphi}{\partial y^2} \right)$$

- Step 1: implicit only in the x-direction for $\Delta t/2$

$$\underbrace{-d_1 u_{i+1,j}^{n+1/2}}_{c} + \underbrace{(1+2d_1) u_{i,j}^{n+1/2}}_{b} \underbrace{- d_1 u_{i-1,j}^{n+1/2}}_{a} = \underbrace{d_2 u_{i,j+1}^{n} + (1-2d_2) u_{i,j}^{n} + d_2 u_{i,j-1}^{n}}_{d}$$

  - sort interior of 2D mesh into 1D vector
    - sorting of a,b,c,d:
  - apply bc for b,d but also a,c!
  - add source terms to d @ t^n



- Must solve step 1 for entire mesh and update ghost cells before step 2.

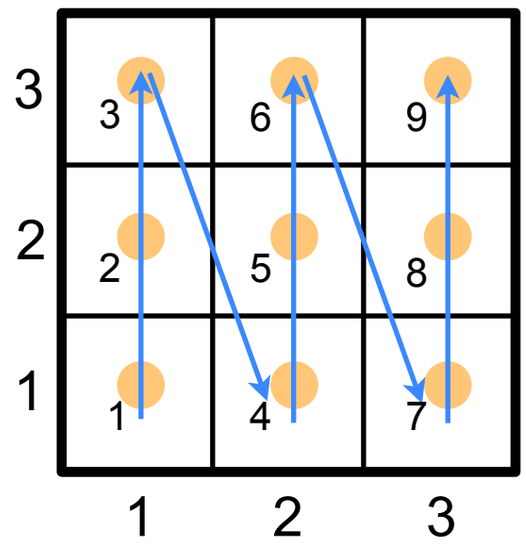
How to code ADI

$$d_x = \frac{\alpha \Delta t}{\Delta x^2} \quad d_y = \frac{\alpha \Delta t}{\Delta y^2} \quad d_1 = \frac{d_x}{2} \quad d_2 = \frac{d_y}{2} \quad \frac{\partial \varphi}{\partial t} = \alpha \left(\frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial^2 \varphi}{\partial y^2} \right)$$

- Step 2: implicit only in the y-direction for $\Delta t/2$

$$\underbrace{-d_2 u_{i,j+1}^{n+1}}_{c} + \underbrace{(1+2d_2) u_{i,j}^{n+1}}_{b} \underbrace{- d_2 u_{i,j-1}^{n+1}}_{a} = \underbrace{d_1 u_{i+1,j}^{n+1/2} + (1-2d_1) u_{i,j}^{n+1/2} + d_1 u_{i-1,j}^{n+1/2}}_{d}$$

  - sort interior of 2D mesh into 1D vector
    - sorting of a,b,c,d:
  - apply bc for b,d but also a,c!
  - add source terms to d @ $t^{n+1/2}$



Parabolic Equations with Source Terms

What changes for 2D vs 1D problems?

- additional finite difference terms for the y-direction derivate
- different stability constraint for time step for explicit methods
- implicit methods are no longer tridiagonal
  - use iterative method (V-cycle multigrid)
  - **use ADI with Gaussian elimination for the tridiagonal systems**

- Comments on coding parabolic equation solvers
  - Explicit methods have a stable time step limitation

1D: $\Delta t \leq \frac{1}{2}\frac{h^2}{\alpha}$ 2D: $\Delta t \leq \frac{1}{4}\frac{h^2}{\alpha}$

- implement this with a security factor CFL, typically CFL = 0.5 … 0.9

1D: $\Delta t = CFL \cdot \frac{1}{2}\frac{h^2}{\alpha}$ 2D: $\Delta t = CFL \cdot \frac{1}{4}\frac{h^2}{\alpha}$

  - If time step is set by above equations, how to “hit” exactly requested output times?

```
if (time < outputTime) .and. (time + dt >= outputTime) then
    dt = outputTime - time;
    setFlagforOuput;
end if
```

  - code layout for parabolic solver

```
setInitialConditions;
applyBoundaryConditions;
time = intialTime;
while (time < endTime)
    dt = calculateStableTimeStep;
    calculateNewTimeStepSolution;
    applyBoundaryConditions;
    time = time+dt;
    doOutputIfRequired;
end if
```