CS601, Lecture 17/10/2022 – Numerical Solution for a 2D problem using FDM

The previous lecture focused on computing the numerical solution for two example problems that are modeled using PDEs and for which the boundary (and initial) conditions given. This lecture introduces you to the 2D problem (so far, we have been looking at 1D problems) and also some terminology along the way.

Conditionally stable **and** ***unconditionally stable*** **methods**: recall that while deriving the difference equations, we truncated the Taylor series, and this resulted in *truncation error*. Also recall that we may have an analytical solution (A) for a PDE and that the analytical solution is typically expensive to compute and may or may not exist for all the problems. Hence, we use a numerical solution (D). The numerical solution will have the truncation error. When this numerical solution is implemented in computer with finite accuracy, a *round-off error* is also introduced additionally. So, we have:

round-off error $= \epsilon = N - D$

we say that the algorithm is *stable* if $\frac{\epsilon^{n+1}}{\epsilon^{n}} \leq 1$

if this were not the case, then it would mean that the error at time step $t = n + 1$ is more than the error at time step $t = n$. So, the error would continuously increase as time progresses.

For the time marching problem of heat diffusion through 1D rod discussed in the previous lecture, if we are using the explicit method, then the method is stable only if $\Delta t <= \Delta x/C$, where C is a constant called as wave speed. In other words, if this condition is violated, we could end up producing a computed result that is impractical / impossible (e.g. temperature at a point on the rod showing as colder and colder as time progresses.)

For the heat equation problem, if we are using the explicit method, the method is stable only if $\Delta t <= \Delta x^2/2\alpha$.

The implicit method (Crank-Nicholson) discussed in previous class requires imposition of no such condition on time step Δt. Hence, the implicit methods are referred to as *unconditionally stable* methods. Being an unconditionally stable method, can a method choose any value Δt ? Not actually. Because, if you chose a large value of time step Δt, you could lose accuracy of the result that you produce (e.g. suppose you chose $\Delta t = 10mins,$ you can compute the temperature at a point on the rod after 10 mins with computation that time-stepped once. However, it is very likely that this computation would not be accurate.).

The explicit methods used in time-marching problems are also referred to as *conditionally stable* methods because of the requirement of a condition on the time step Δt.

For problems that do not have time as an independent variable (i.e. *spatial problems*), what errors we can expect? Only the truncation error or *discretization error.* We do not have any concept of imposition of a condition on the spatial variables here.

2D Heat Conduction Problem:

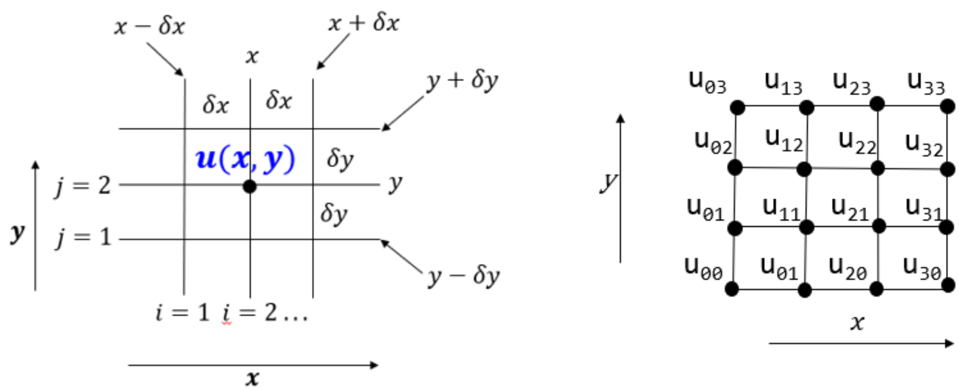
The following PDE models this problem:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0, \qquad \text{———— } \mathbf{(1)}$$

where $u(x, y)$ is the dependent variable dependent on spatial variables x and y.

The above equation is an example of an elliptic PDE and is also called as *Laplace equation*. If the RHS of above equation is non-zero and is $u(x, y)$, then the equation is also called as *Poisson equation.*

When you discretize the 2D domain, you get a small rectangle as the sub-domain. Contrast this with 1D problems where you got a line as a sub-domain. In *structured grids* the shape and size of the sub-domain is same. In *unstructured grids* this is not the case. The right-hand-side of the following figure illustrates the discretization of a 2D rectangular domain and the resulting sub-domains. We have multiple grid points, each identified with a pair (i, j), where the i denotes the index along the x direction and j denotes the index along the y direction. The left-hand-side of the figure focuses on a grid point (2,2). The value of the dependent variable at a grid point (x, y) is shown as $u(x, y)$ and denoted by u_{xy}. The discretization steps along the x and y direction are δx and δy respectively. We can choose the values of δx an δy (can be same or different). Typically, they are the same value.



u_{ij} denotes the dependent variable at grid point (i,j)

Numerical solution for PDE of equation (1):

Using difference equations for $\frac{\partial u}{\partial x}$ and $\frac{\partial u}{\partial y}$, we have

$$\frac{\partial u}{\partial x} = \frac{u(x+\delta x) - 2u(x) + u(x-\delta x)}{\delta x}, \qquad \frac{\partial u}{\partial y} = \frac{u(y+\delta y) - 2u(y) + u(y-\delta y)}{\delta y}$$

Similarly, using difference equations for $\frac{\partial^2 u}{\partial x^2}$ and $\frac{\partial^2 u}{\partial y^2}$, we have

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{\left(u(x+\delta x,\ y) - 2u(x,\ y) + u(x-\delta x, y)\right)}{(\delta x)^2}$$

$$\frac{\partial^2 u}{\partial y^2} \approx \frac{\left(u(x, y+\delta y) - 2u(x,\ y) + u(x, y-\delta y)\right)}{(\delta y)^2}$$

Substituting the above in equation (1):

$$\frac{\left(u(x+\delta x,\ y) - 2u(x,\ y) + u(x-\delta x, y)\right)}{(\delta x)^2}$$

$$+$$

$$\frac{\left(u(x, y+\delta y) - 2u(x,\ y) + u(x, y-\delta y)\right)}{(\delta y)^2}$$

$$=$$

$$\frac{\left(u(x+\delta x,\ y) +\ u(x, y+\delta y) - 4u(x,\ y) + u(x-\delta x, y) +\ u(x, y-\delta y)\right)}{(h)^2} \qquad \text{———} \qquad \textbf{(2)}$$

The above equation is also called as *algebraic equation*.

For each grid point, write equation (2) and obtain a system of equations.

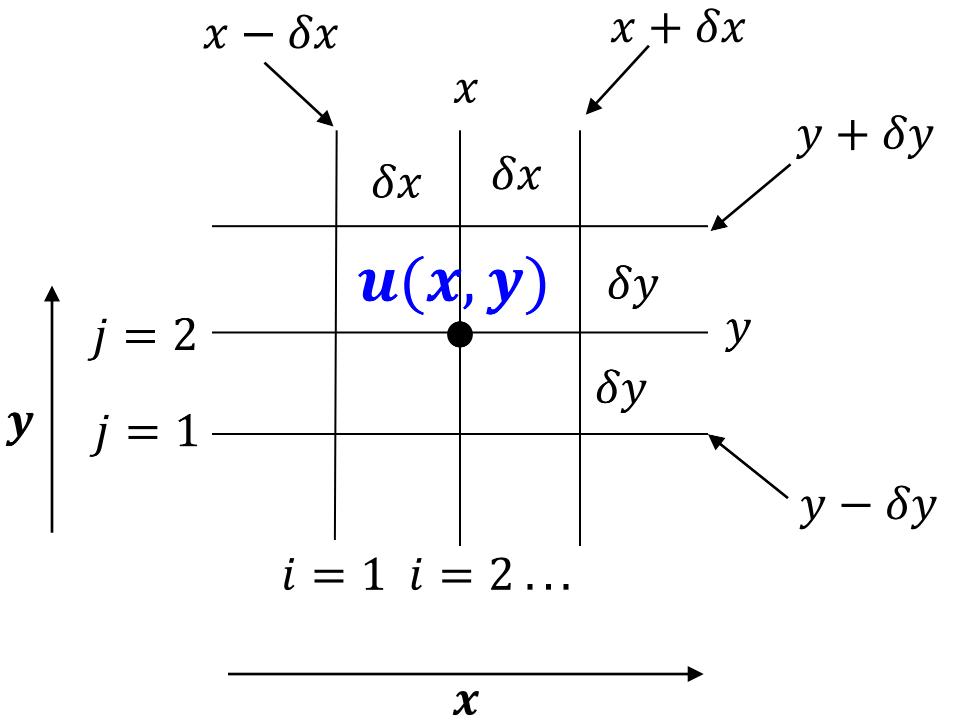
IMPORTANT: it is absolutely critical to number the grid points in such a way that you get a nice structure to the matrix A. if you do not number the grid points properly, your non-zeros will be scattered all over the matrix. If the numbering is incorrect, your computation will not be able to exploit the structure inherent in the matrix. However, your solution will still be as expected.

Once you have a system of equations, how does this system of equations look like and how to solve them?

Next: slides from previous years' offering of CS601

Elliptic Equation – Numerical Solution

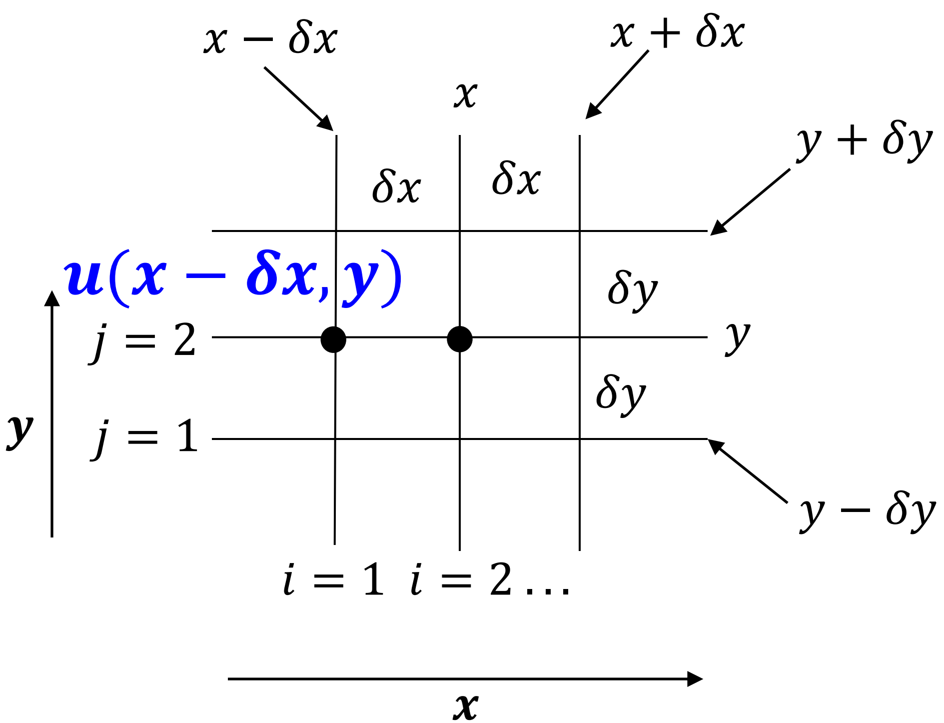
- Representing $u(x, y)$



Notation: $\mathbf{u_{i,j}}$

Elliptic Equation – Numerical Solution

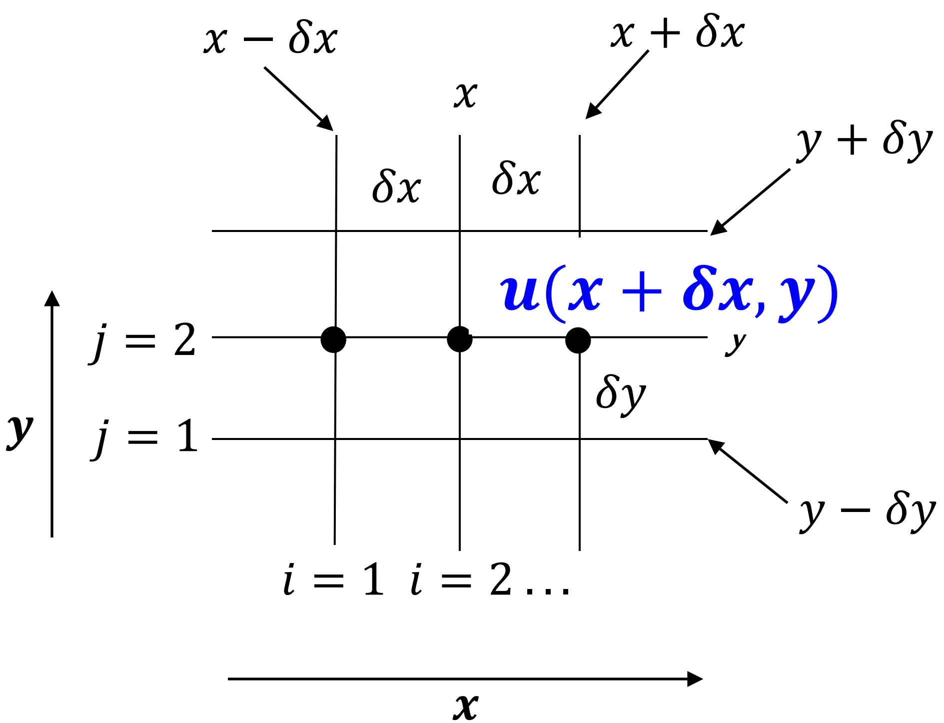
- Representing $u(x - \delta x, y)$



Notation: $\mathbf{u_{i-1,j}}$

Elliptic Equation – Numerical Solution

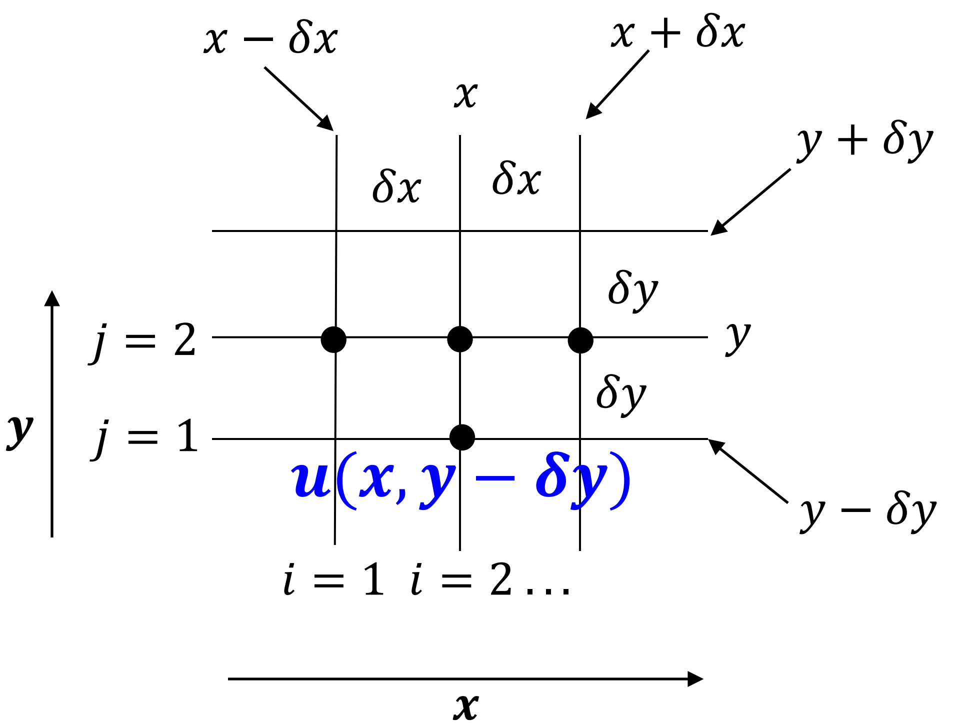
- Representing $u(x + \delta x, y)$



Notation: $\mathbf{u_{i+1,j}}$

Elliptic Equation – Numerical Solution

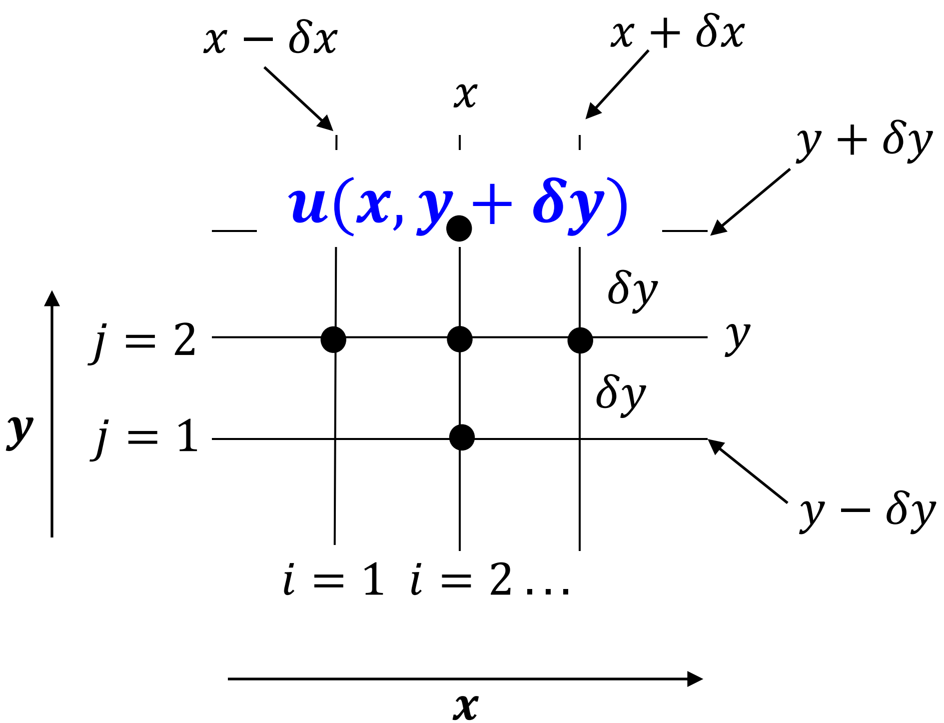
- Representing $u(x, y - \delta y)$



Notation: $\mathbf{u_{i,j-1}}$

Elliptic Equation – Numerical Solution

- Representing $u(x, y + \delta y)$



Notation: $\mathbf{u_{i,j+1}}$

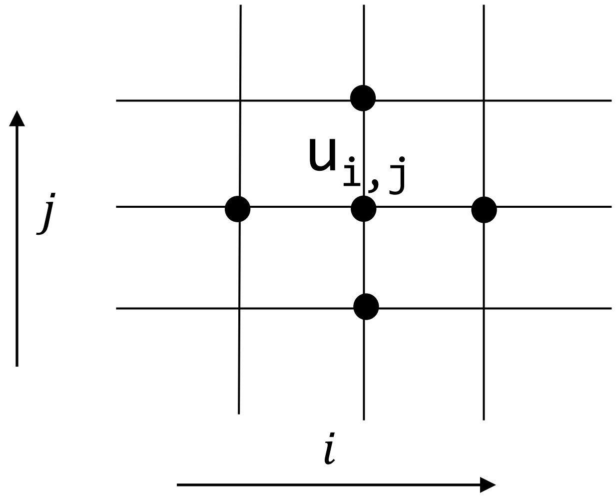
Elliptic Equation – Numerical Solution

- Rewriting:

$$\frac{\left(u(x+\delta x,y)+\ u(x,y+\delta y)-4u(x,y)+u(x-\delta x,y)+\ u(x,y-\delta y)\right)}{(h)^2}$$

$$= f(x,y)$$

$$\frac{\mathbf{u_{i+1,j} + u_{i,j+1}} - 4\mathbf{u_{i,j}} + \mathbf{u_{i-1,j} + u_{i,j-1}}}{\mathbf{h^2}} = \mathbf{f_{i,j}}$$



5-point stencil

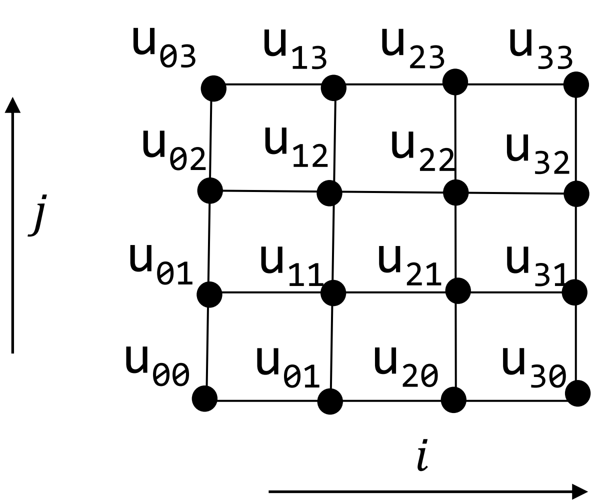
Elliptic Equation – Computing Stencil

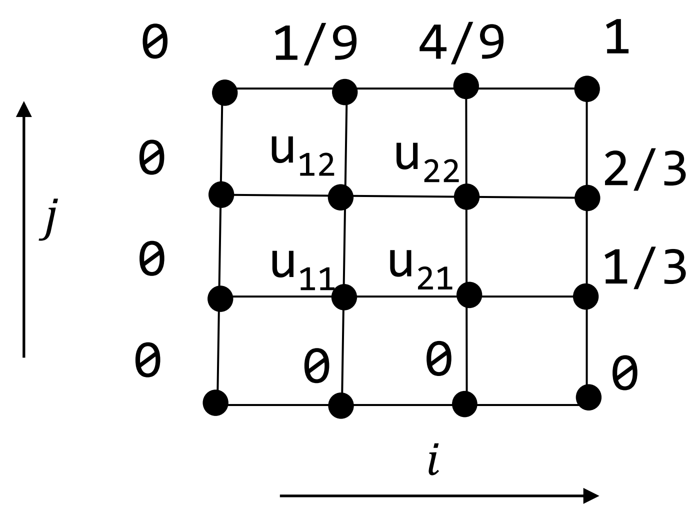
- Consider the *boundary-value* problem:

$u_{xx} + uyy = 0$ in the square $0 < x < 1, 0 < y < 1$

$u = x^2y$ on the boundary, $h = 1/3$

$$\frac{u_{i+1,j} + u_{i,j+1} - 4u_{i,j} + u_{i-1,j} + u_{i,j-1}}{h^2} = 0$$





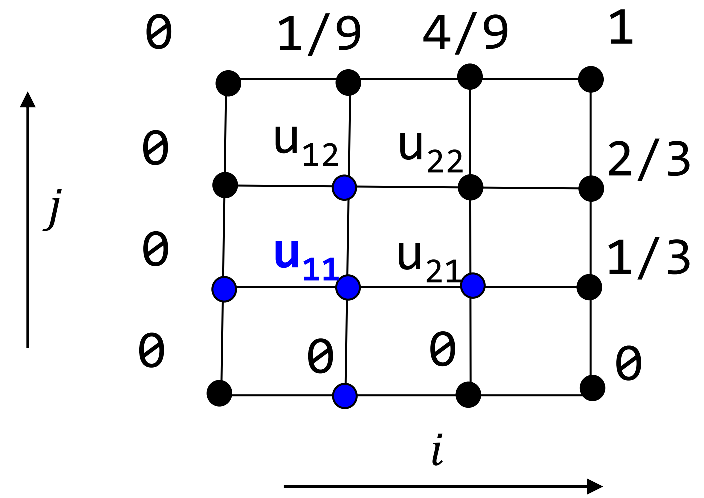
Elliptic Equation – Computing Stencil

- Computing u_{11}

$(u_{i+1,j} + u_{i,j+1} - 4u_{i,j} + u_{i-1,j} + u_{i,j-1} = 0)$

$$u_{21} + u_{12} - 4u_{11} + u_{01} + u_{10} = 0$$

$$u_{21} + u_{12} - 4u_{11} + 0 + 0 = 0$$



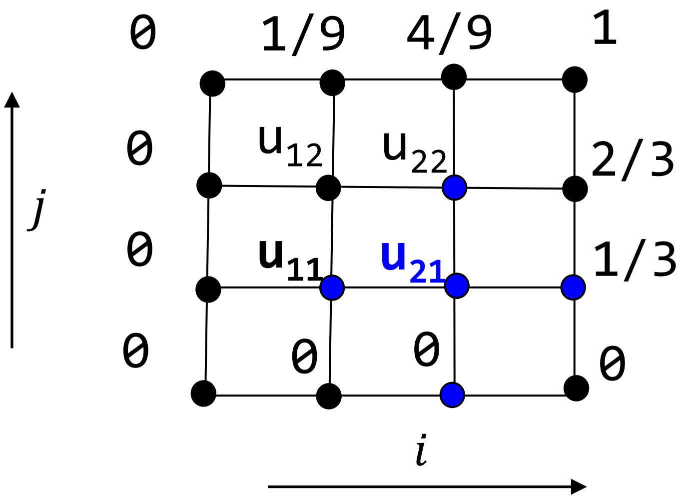
Elliptic Equation – Computing Stencil

- Computing u_{21}

$(u_{i+1,j} + u_{i,j+1} - 4u_{i,j} + u_{i-1,j} + u_{i,j-1} = 0)$

$$u_{31} + u_{22} - 4u_{21} + u_{11} + u_{20} = 0$$

$$1/3 + u_{22} - 4u_{21} + U_{11} + 0 = 0$$



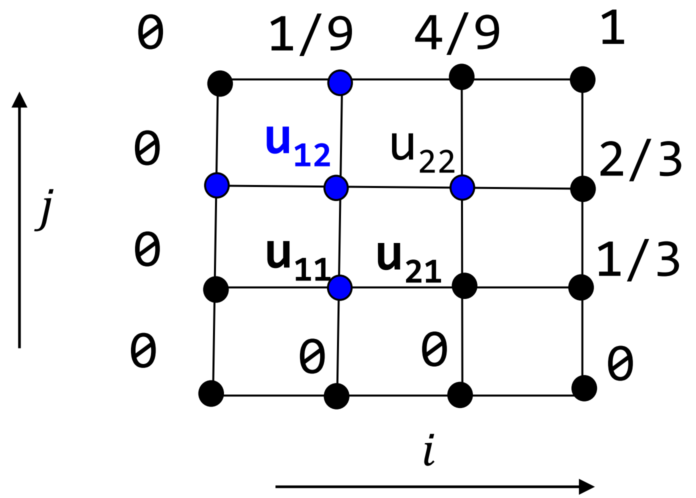
Elliptic Equation – Computing Stencil

- Computing u_{12}

$(u_{i+1,j} + u_{i,j+1} - 4u_{i,j} + u_{i-1,j} + u_{i,j-1} = 0)$

$$u_{22} + u_{13} - 4u_{12} + u_{02} + u_{11} = 0$$

$$u_{22} + 1/9 - 4u_{12} + 0 + u_{11} = 0$$



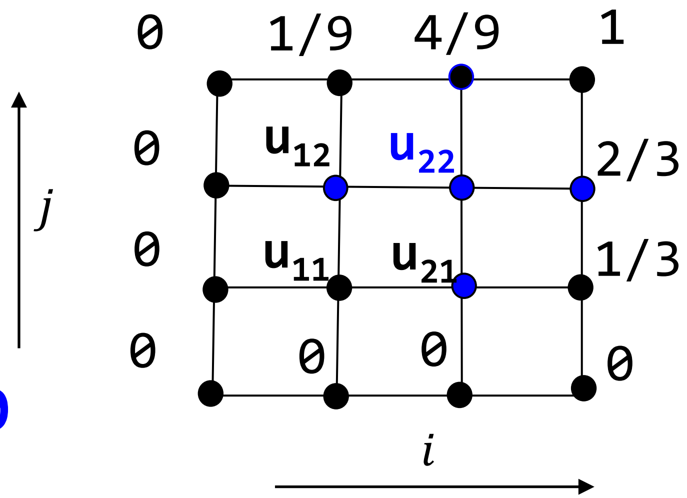
Elliptic Equation – Computing Stencil

- Computing u_{22}

$(u_{i+1,j} + u_{i,j+1} - 4u_{i,j} + u_{i-1,j} + u_{i,j-1} = 0)$

$u_{32} + u_{23} - 4u_{22} + u_{12} + u_{21} = 0$

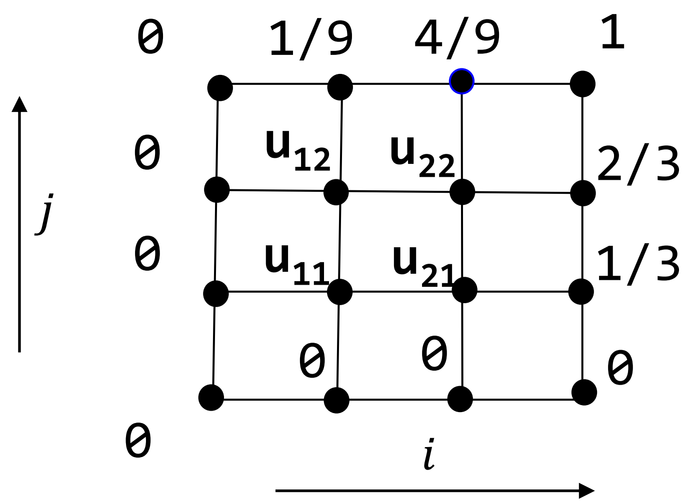
$2/3 + 4/9 - 4u_{22} + u_{12} + u_{21} = 0$



Elliptic Equation – Computing Stencil

- System of Equations

$(u_{i+1,j} + u_{i,j+1} - 4u_{i,j} + u_{i-1,j} + u_{i,j-1} = 0)$



Right Top Center Left Bottom

$$u_{21} + u_{12} - 4u_{11} + 0 + 0 = 0$$

$$1/3 + u_{22} - 4u_{21} + u_{11} + 0 = 0$$

$$u_{22} + 1/9 - 4u_{12} + 0 + u_{11} = 0$$

$$2/3 + 4/9 - 4u_{22} + u_{12} + u_{21} = 0$$

Elliptic Equation – Computing Stencil

- Computing System of Equations:

$$u_{21} + u_{12} - 4u_{11} + 0 + 0 = 0$$

$$1/3 + u_{22} - 4u_{21} + u_{11} + 0 = 0$$

$$u_{22} + 1/9 - 4u_{12} + 0 + u_{11} = 0$$

$$2/3 + 4/9 - 4u_{22} + u_{12} + u_{21} = 0$$

$$\begin{pmatrix} -4 & 1 & 1 & 0 \\ 1 & -4 & 0 & 1 \\ 1 & 0 & -4 & 1 \\ 0 & 1 & 1 & -4 \end{pmatrix} \begin{pmatrix} u_{11} \\ u_{21} \\ u_{12} \\ u_{22} \end{pmatrix} = \begin{pmatrix} 0 \\ -1/3 \\ -1/9 \\ -10/9 \end{pmatrix}$$

A x = B

Ax=B

Matrix A has only coefficients

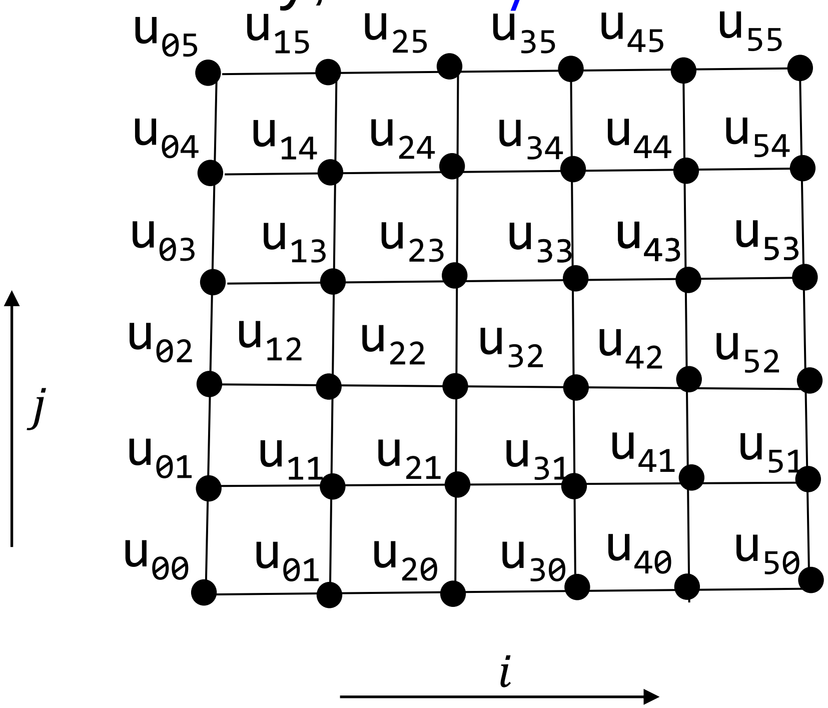
	1	
1	**-4**	**1**
	1	

Elliptic Equation – Computing Stencil

- Consider the *boundary-value* problem (here u_{xx} denotes $\partial^2 u/\partial x^2$):

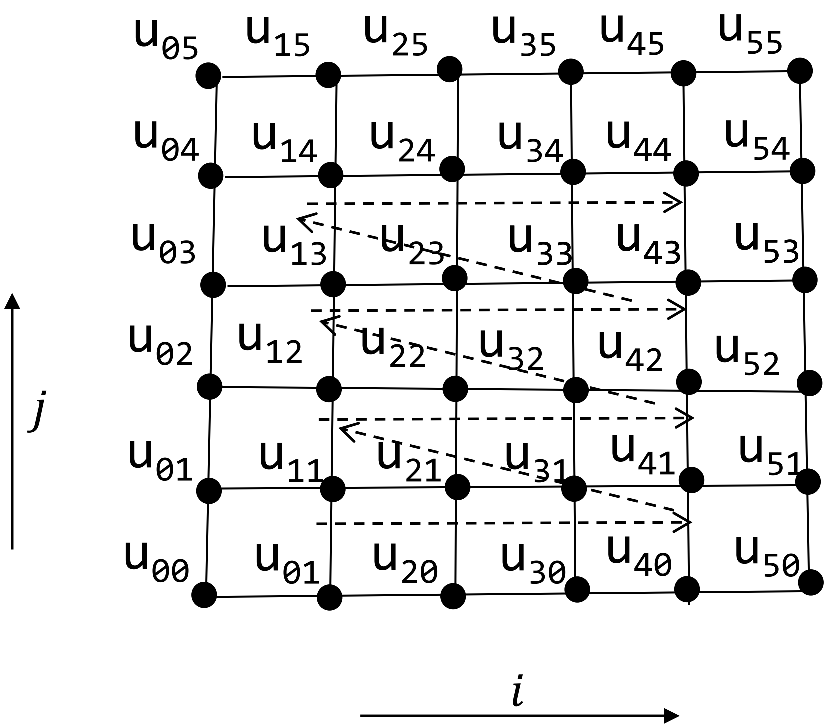
$u_{xx} + u_{yy} = 0$ in the square $0 < x < 1, 0 < y < 1$

$u = x^2 y$ on the boundary, $\boldsymbol{h = 1/5}$



Elliptic Equation – Computing Stencil

- Computing stencil (boundary values are all given): 16 unknowns (u_{11} to u_{44}), So, 16 equations.



Elliptic Equation – Computing Stencil

-4	1	0	0	1								
1	-4	1	0	0	1							
0	1	-4	1	0	0	1						
0	0	1	-4	1	0	0	1					
1	0	0	1	-4	1	0	0	1				
	1	0	0	1	-4	1	0	0	1			
		1	0	0	1	-4	1	0	0	1		
			1	0	0	1	-4	1	0	0	1	
				1	0	0	1	-4	1	0	0	1

- Lot of Zeros!
- Five non-zero bands
  - Top-left to bottom-right diagonals
- Main diagonal is all -4 (from center of the stencil)
- What about others?

Elliptic Equation – Computing Stencil

-4	1	0	0	1								
1	-4	1	0	0	1							
0	1	-4	1	0	0	1						
0	0	1	-4	1	0	0	1					
1	0	0	1	-4	1	0	0	1				
	1	0	0	1	-4	1	0	0	1			
		1	0	0	1	-4	1	0	0	1		
			1	0	0	1	-4	1	0	0	1	
				1	0	0	1	-4	1	0	0	1

Left

- Lot of Zeros!
- Five non-zero bands
    - Top-left to bottom-right diagonals
- Main diagonal is all -4 (from center of the stencil)
- What about others?

Elliptic Equation – Computing Stencil

-4	1	0	0	1								
1	-4	1	0	0	1							
0	1	-4	1	0	0	1						
0	0	1	-4	1	0	0	1					
1	0	0	1	-4	1	0	0	1				
	1	0	0	1	-4	1	0	0	1			
		1	0	0	1	-4	1	0	0	1		
			1	0	0	1	-4	1	0	0	1	
				1	0	0	1	-4	1	0	0	1

Right

- Lot of Zeros!
- Five non-zero bands
  - Top-left to bottom-right diagonals
- Main diagonal is all -4 (from center of the stencil)
- What about others?

Elliptic Equation – Computing Stencil

-4	1	0	0	1								
1	-4	1	0	0	1							
0	1	-4	1	0	0	1						
0	0	1	-4	1	0	0	1					
1	0	0	1	-4	1	0	0	1				
	1	0	0	1	-4	1	0	0	1			
		1	0	0	1	-4	1	0	0	1		
			1	0	0	1	-4	1	0	0	1	
				1	0	0	1	-4	1	0	0	1

Bottom

- Lot of Zeros!
- Five non-zero bands
  - Top-left to bottom-right diagonals
- Main diagonal is all -4 (from center of the stencil)
- What about others?

Elliptic Equation – Computing Stencil

-4	1	0	0	1								
1	-4	1	0	0	1							
0	1	-4	1	0	0	1						
0	0	1	-4	1	0	0	1					
1	0	0	1	-4	1	0	0	1				
	1	0	0	1	-4	1	0	0	1			
		1	0	0	1	-4	1	0	0	1		
			1	0	0	1	-4	1	0	0	1	
				1	0	0	1	-4	1	0	0	1

Top

- Lot of Zeros!
- Five non-zero bands
  - Top-left to bottom-right diagonals
- Main diagonal is all -4 (from center of the stencil)
- What about others?

Computing Stencil – Iterative Methods

- Jacobi and Gauss-Seidel
  - Start with an initial guess for the unknowns u^0_{ij}
  - Improve the guess u^1_{ij}
  - Iterate: derive the new guess, u^{n+1}_{ij} , from old guess u^n_{ij}
- Solution (Jacobi):
  - Approximate the *value of the center* with old values of (left, right, top, bottom)

Background – Jacobi Iteration

- **Goal:** find solution to system of equations represented by $AX=B$
- **Approach:** find sequence of approximations X^0 X^1 X^2 . . . X^n, which gradually approach X.
  - X^0 is called initial guess, X^i's called *iterates*
- **Method:**
  - Split A into $A=L+D+U$ e.g.

$$\begin{pmatrix} -4 & 1 & 1 & 0 \\ 1 & -4 & 0 & 1 \\ 1 & 0 & -4 & 1 \\ 0 & 1 & 1 & -4 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix} + \begin{pmatrix} -4 & 0 & 0 & 0 \\ 0 & -4 & 0 & 0 \\ 0 & 0 & -4 & 0 \\ 0 & 0 & 0 & -4 \end{pmatrix} + \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

L D U

Background – Jacobi Iteration

- **Compute:** $AX=B$ is $(L+D+U)X=B$

  $\Rightarrow DX = -(L+U)X+B$

  $\Rightarrow DX^{(k+1)} = -(L+U)X^{k}+B$ **(iterate step)**

  $\Rightarrow X^{(k+1)} = D^{-1}(-(L+U)X^{k}) + D^{-1}B$

  (As long as D has no zeros in the diagonal $X^{(k+1)}$ is obtained)

- E.g. $$\begin{pmatrix} -4 & 0 & 0 & 0 \\ 0 & -4 & 0 & 0 \\ 0 & 0 & -4 & 0 \\ 0 & 0 & 0 & -4 \end{pmatrix} \begin{pmatrix} u_{11} \\ u_{21} \\ u_{12} \\ u_{22} \end{pmatrix}^{\mathbf{1}} = -\begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} u_{11} \\ u_{21} \\ u_{12} \\ u_{22} \end{pmatrix}^{\mathbf{0}} + \begin{pmatrix} 0 \\ -1/3 \\ -1/9 \\ -10/9 \end{pmatrix},$$

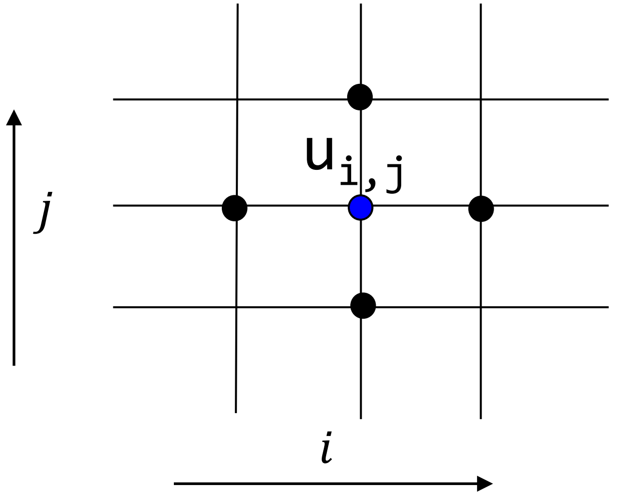
u_{ij} 's value in $(\mathbf{1})^{st}$ iteration is computed based on u_{ij} values computed in $(\mathbf{0})^{th}$ iteration

Background – Jacobi Iteration

- E.g. $$\begin{pmatrix} -4 & 0 & 0 & 0 \\ 0 & -4 & 0 & 0 \\ 0 & 0 & -4 & 0 \\ 0 & 0 & 0 & -4 \end{pmatrix} \begin{pmatrix} u_{11} \\ u_{21} \\ u_{12} \\ u_{22} \end{pmatrix}^{k+1} = -\begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} u_{11} \\ u_{21} \\ u_{12} \\ u_{22} \end{pmatrix}^{k} + \begin{pmatrix} 0 \\ -1/3 \\ -1/9 \\ -10/9 \end{pmatrix},$$

u_{ij} 's value in $(k+1)^{st}$ iteration is computed based on u_{ij} values computed in $(k)^{th}$ iteration

- Center's value is updated. Why?



5-point stencil

Computing Stencil – Recap

- Jacobi and Gauss-Seidel (Solution approach)
  - Start with an initial guess for the unknowns u^0_{ij}
  - Improve the guess u^1_{ij}
  - Iterate: derive the new guess, u^{n+1}_{ij} , from old guess u^n_{ij}
- Solution (Jacobi):
  - Approximate the *value of the center* with *old values* of (left, right, top, bottom)

Computing Stencil – Recap

- $u_{right} + u_{top} - 4u_{center} + u_{left} + u_{bottom} = 0$

  => $u_{center} = 1/4(u_{right} + u_{top} + u_{left} + u_{bottom})$

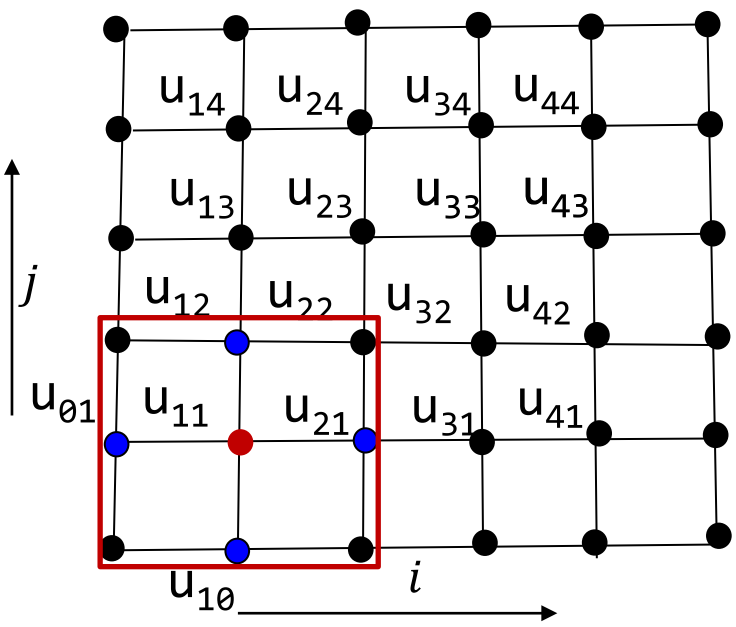
- Applying Jacobi Iteration:

  $u_{center}^{(k+1)} = 1/4(u_{right}^{(k)} + u_{top}^{(k)} + u_{left}^{(k)} + u_{bottom}^{(k)})$

Computing Stencil – Recap

- Example: applying Jacobi Iteration:

$$u_{center}^{(k+1)} = 1/4(u_{right}^{(k)} + u_{top}^{(k)} + u_{left}^{(k)} + u_{bottom}^{(k)})$$



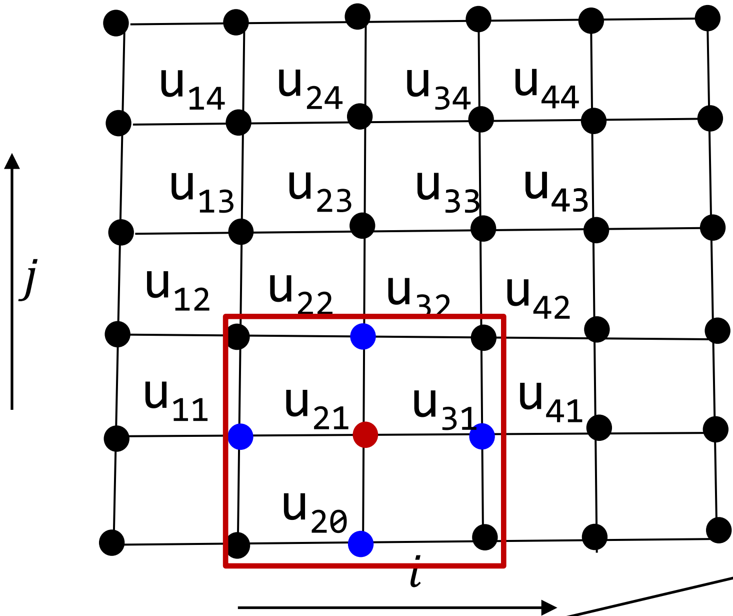
Iteration 1

1) Compute u_{11} using initial guess for u_{12} and u_{21}. u_{01} and u_{10} are known from boundary conditions

Computing Stencil – Recap

- Example: applying Jacobi Iteration:

$$u_{center}^{(k+1)} = 1/4(u_{right}^{(k)} + u_{top}^{(k)} + u_{left}^{(k)} + u_{bottom}^{(k)})$$



<u>Iteration 1</u>

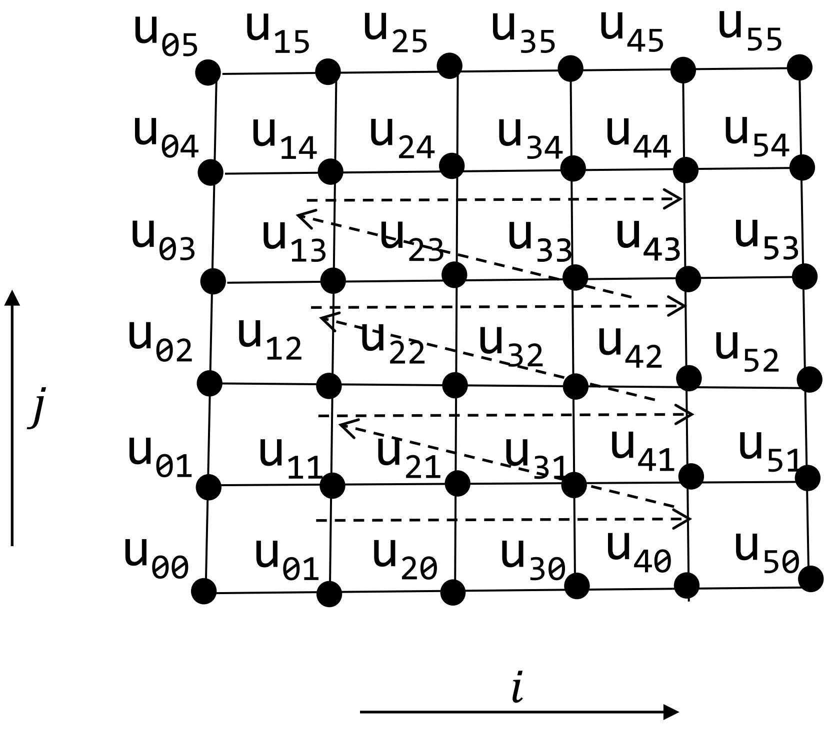
1) Compute u_{11} using initial guess for u_{12} and u_{21}. u_{01} and u_{10} are known from boundary conditions

2) Compute u_{21} using initial guess for u_{11}, u_{31}, and u_{22}. u_{20} are known from boundary conditions

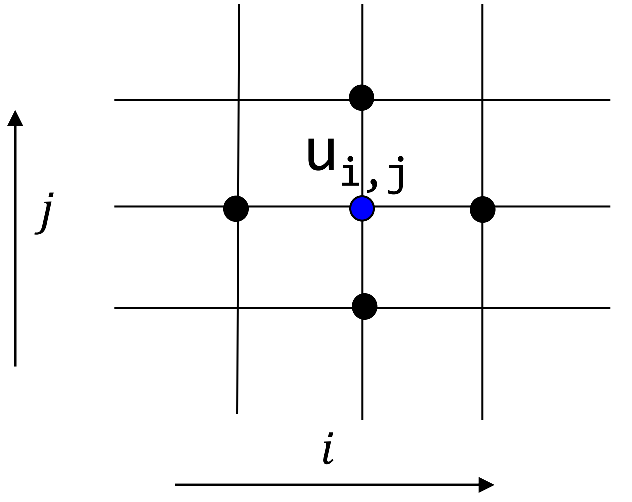
In 2), note that the initial guess for u_{11} is used even though u_{11} was updated just before in 1)

Elliptic Equation – Computing Stencil

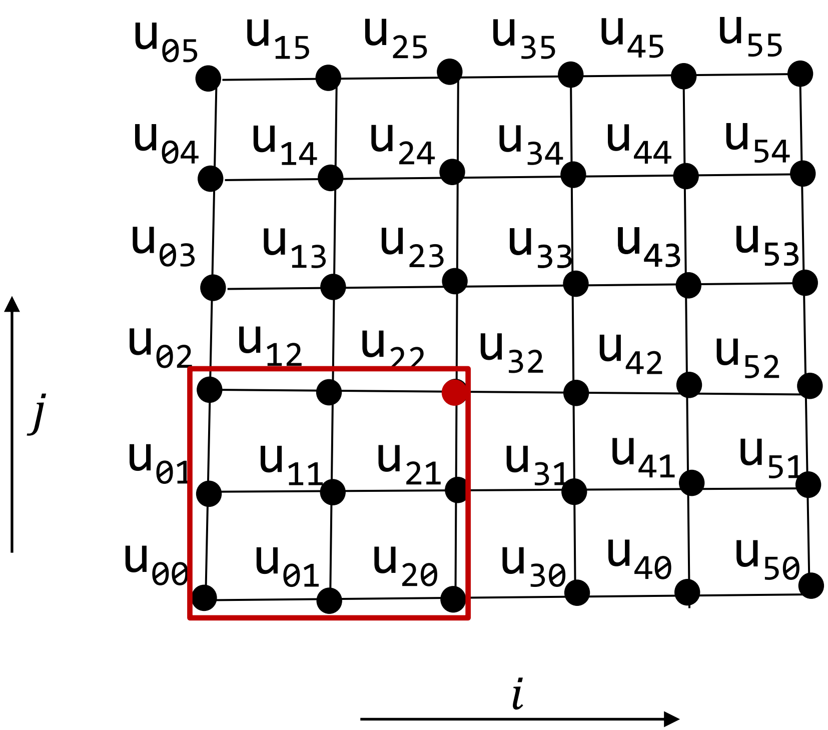
- In every iteration, suppose we follow the computing order as shown (dashed):



In any iteration, what are all the points of a 5-point stencil already updated while computing u_{ij} *?*

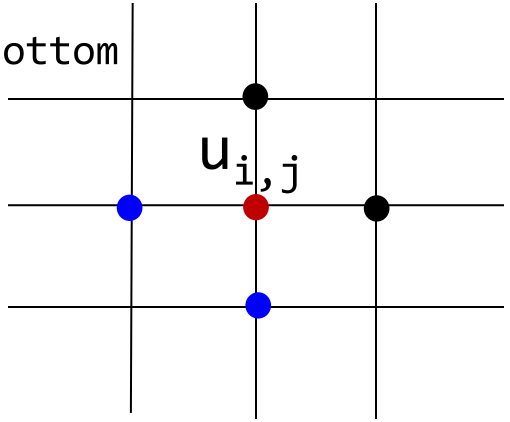


Elliptic Equation – Computing Stencil



What are the points that are already computed at $u_{i,j}$*?*

u_{left}, u_{bottom}



Background – Gauss-Seidel Iteration

- **Compute:** $AX=B$ is $(L+D+U)X=B$

$\Rightarrow (L+D)X = -UX+B$

$\Rightarrow (L+D)X^{(k+1)} = -UX^{k}+B$ **(iterate step)**

$\Rightarrow X^{(k+1)} = (L+D)^{-1}(-UX^{k}) + (L+D)^{-1}B$

(As long as $L+D$ has no zeros in the diagonal $X^{(k+1)}$ is obtained)

- E.g. $\begin{pmatrix} -4 & 0 & 0 & 0 \\ 1 & -4 & 0 & 0 \\ 1 & 0 & -4 & 0 \\ 0 & 1 & 1 & -4 \end{pmatrix} \begin{pmatrix} u_{11} \\ u_{21} \\ u_{12} \\ u_{22} \end{pmatrix}^{1} = - \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} u_{11} \\ u_{21} \\ u_{12} \\ u_{22} \end{pmatrix}^{0} + \begin{pmatrix} 0 \\ -1/3 \\ -1/9 \\ -10/9 \end{pmatrix}$

Computing Stencil – Gauss-Seidel

- Gauss-Seidel: Applying for 2D Laplace Equation

$$u_{center}^{(k+1)} = 1/4(u_{right}^{(k)} + u_{top}^{(k)} + u_{left}^{(k+1)} + u_{bottom}^{(k+1)})$$

- Gauss-Seidel: Observations
  - For a given problem and initial guess, Gauss-seidel *converges faster* than Jacobi
  - An iteration in Jacobi can be parallelized