CS601: Software Development for Scientific Computing

Autumn 2021

Week6:

- Structured Grids (Elliptic PDEs contd..)

Last Week..

- Intermediate C++
  - Class templates, STL, Operator overloading
- Structured Grids (Elliptic PDEs - introduction)

Elliptic Equation – Numerical Solution

1. Approximate the derivatives of $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x, y)$ using central differences
2. Choose step sizes δx and δy for x and y axis resp.
    1. Both and x and y are independent variables here.
    2. Choose δx= δy = h
3. Write difference equation for approximating the PDE above

Elliptic Equation – Numerical Solution

1. Approximate the derivatives of $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x, y)$ using central differences

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

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

Where, δx and δy are step sizes along x and y direction resp.

Elliptic Equation – Numerical Solution

- Substituting in $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x, y)$:

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

$$+$$

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

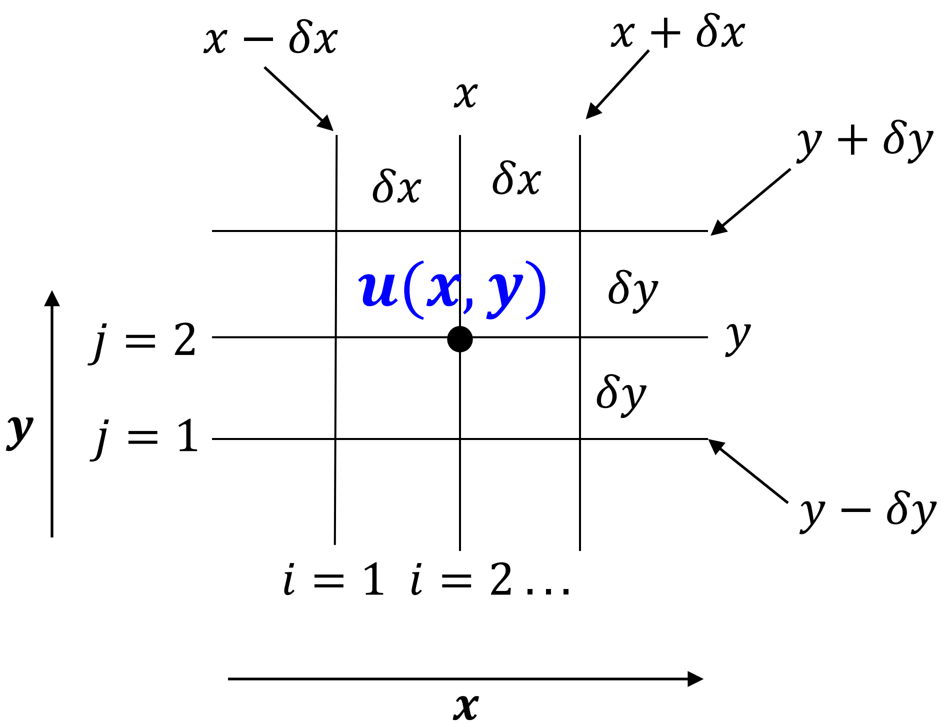
$$=$$

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

$$= f(x, y)$$

Elliptic Equation – Numerical Solution

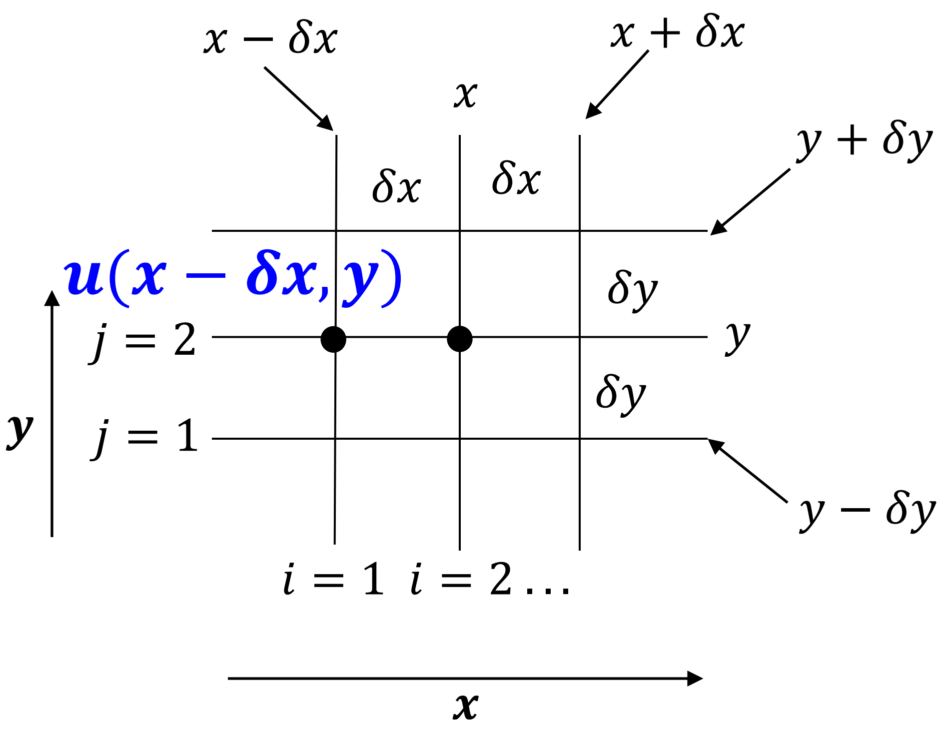
- Representing $u(x, y)$



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

Elliptic Equation – Numerical Solution

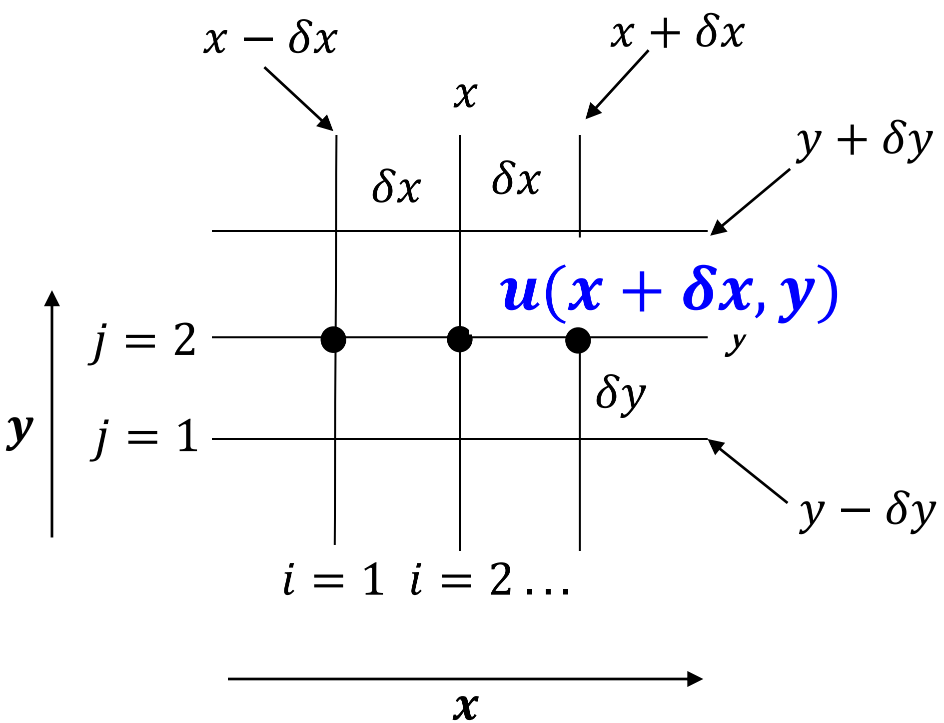
- Representing $u(x - \delta x, y)$



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

Elliptic Equation – Numerical Solution

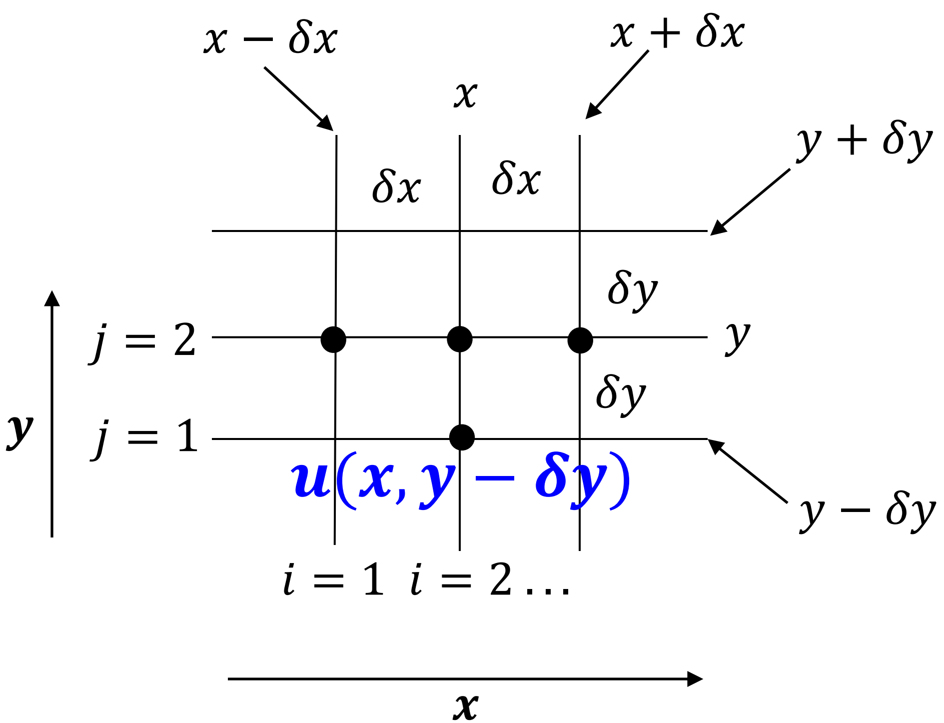
- Representing $u(x+\delta x, y)$



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

Elliptic Equation – Numerical Solution

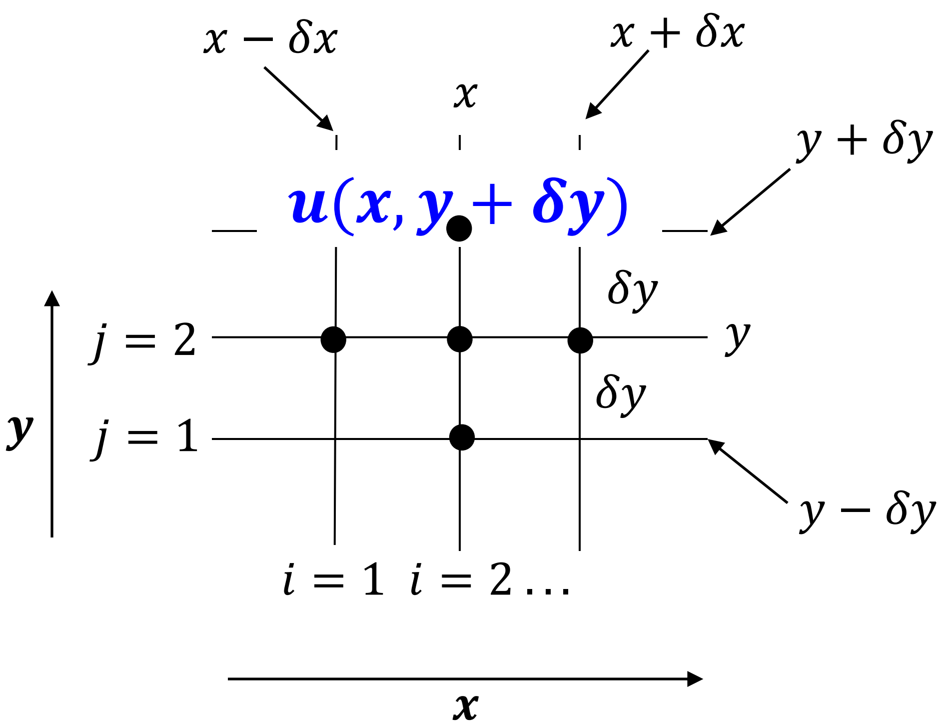
- Representing $u(x, y - \delta y)$



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

Elliptic Equation – Numerical Solution

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



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

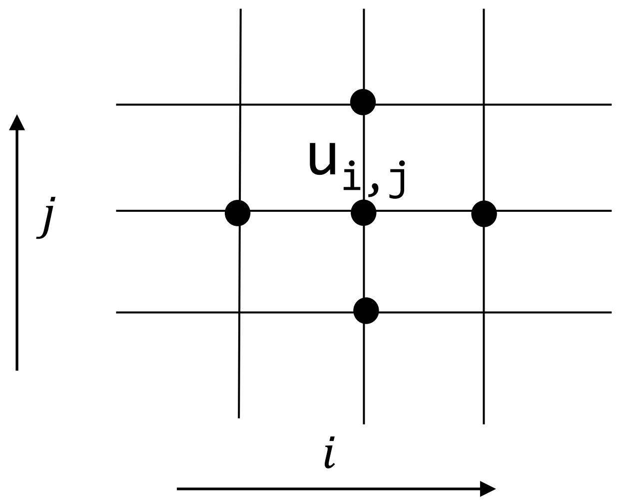
Elliptic Equation – Numerical Solution

- Rewriting:

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

$$= f(x, y)$$

$$\frac{\mathbf{u}_{i+1,j} + \mathbf{u}_{i,j+1} - 4\mathbf{u}_{i,j} + \mathbf{u}_{i-1,j} + \mathbf{u}_{i,j-1}}{\mathbf{h}^2} = \mathbf{f}_{i,j}$$



5-point stencil

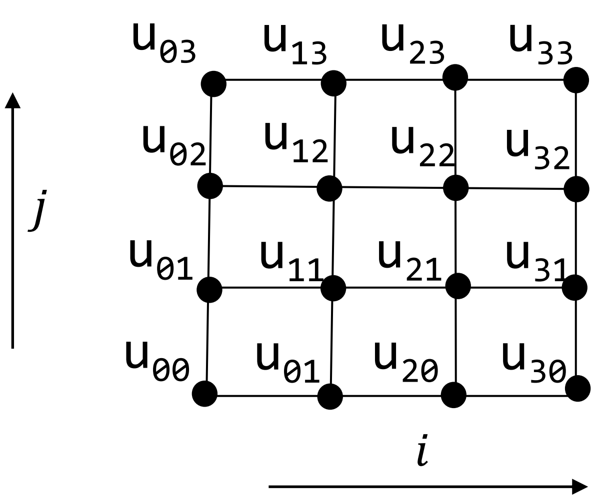
Elliptic Equation – Computing Stencil

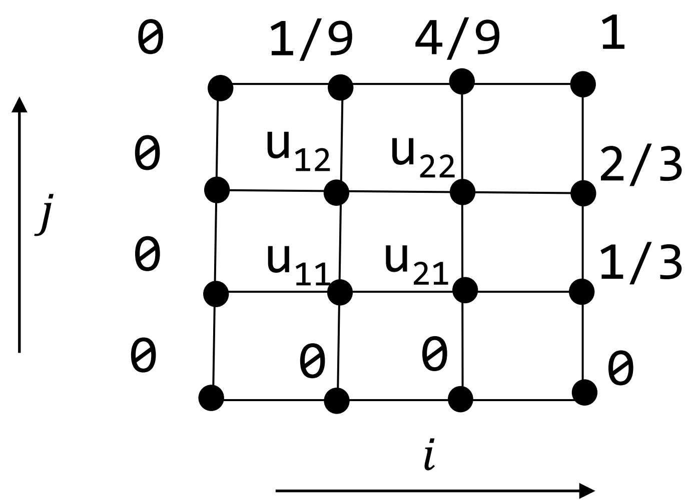
- Consider the *boundary-value* problem:

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

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

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





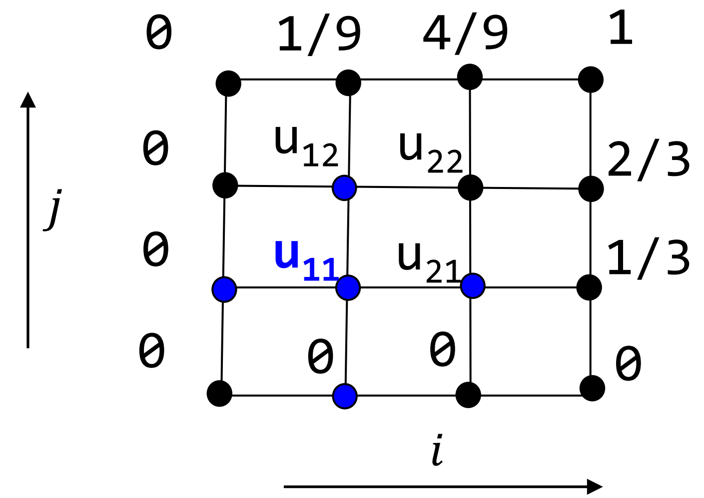
Elliptic Equation – Computing Stencil

- Computing u_{11}

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

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

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



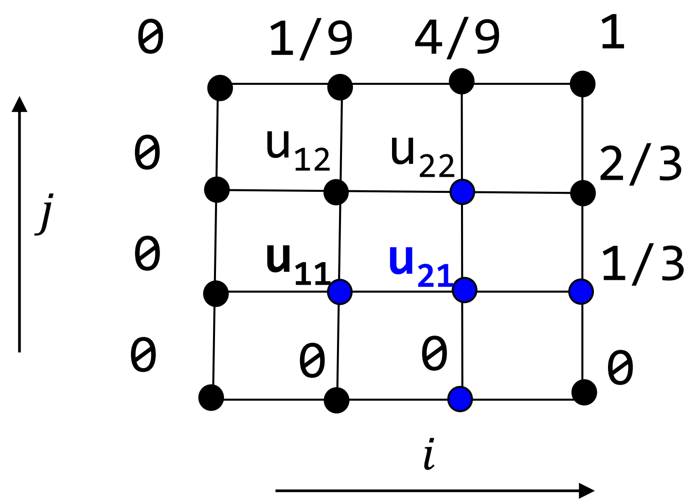
Elliptic Equation – Computing Stencil

- Computing u_{21}

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

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

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



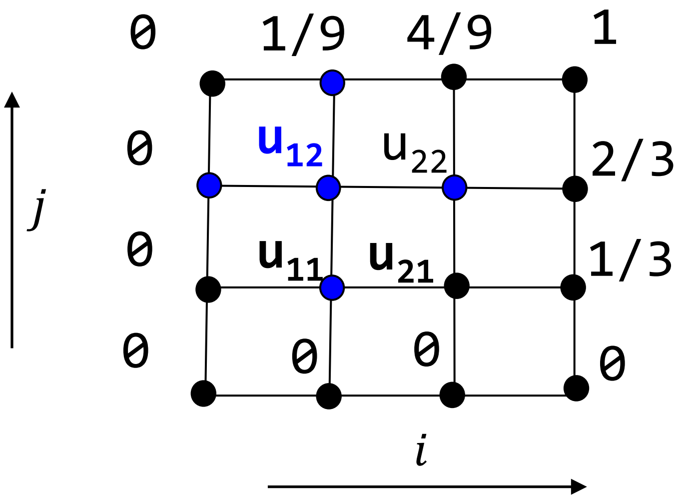
Elliptic Equation – Computing Stencil

- Computing u_{12}

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

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

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



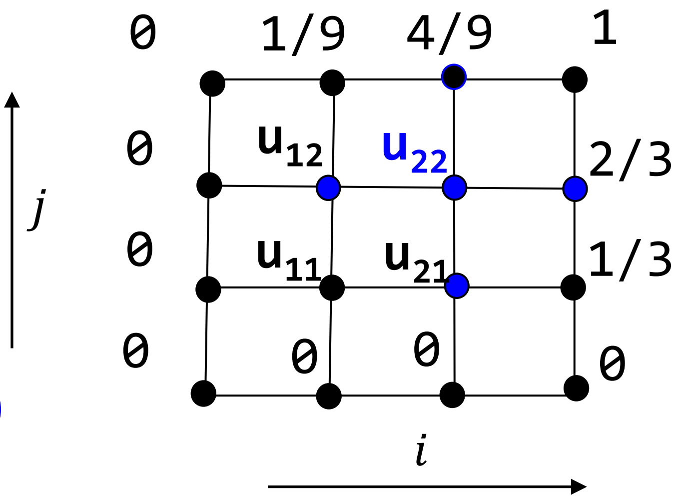
Elliptic Equation – Computing Stencil

- Computing u_{22}

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

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

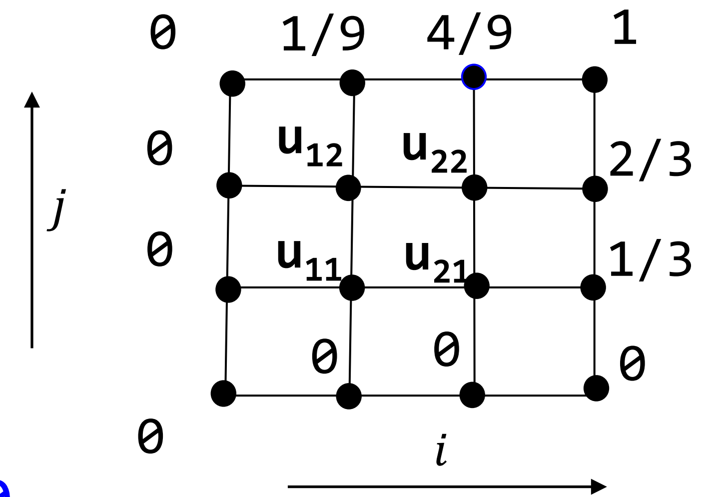
$$2/3 + 4/9 - 4u_{22} + u_{12} + u_{21} = 0$$



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

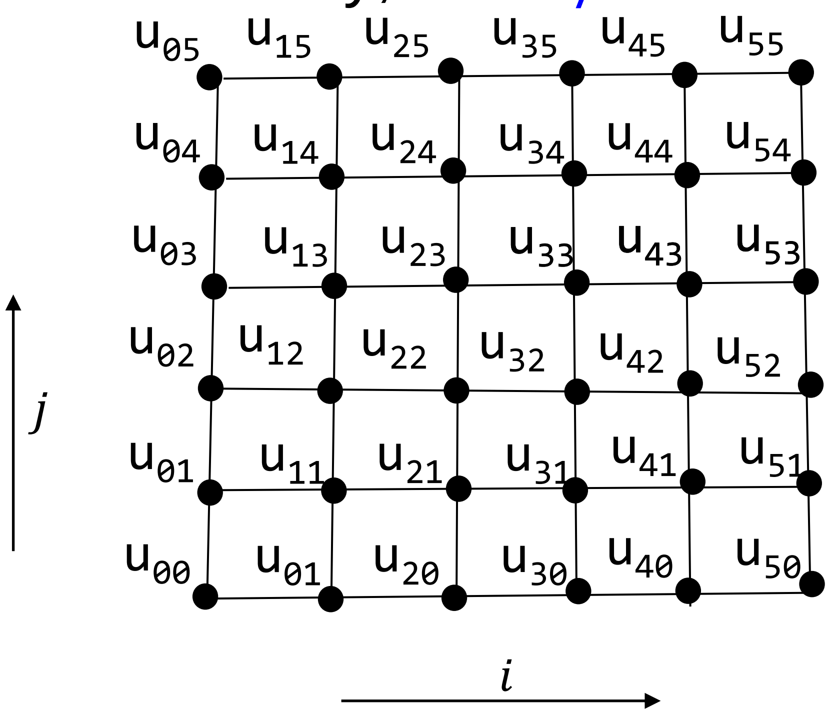
	1	
1	**-4**	**1**
	1	

Elliptic Equation – Computing Stencil

- Consider the *boundary-value* problem:

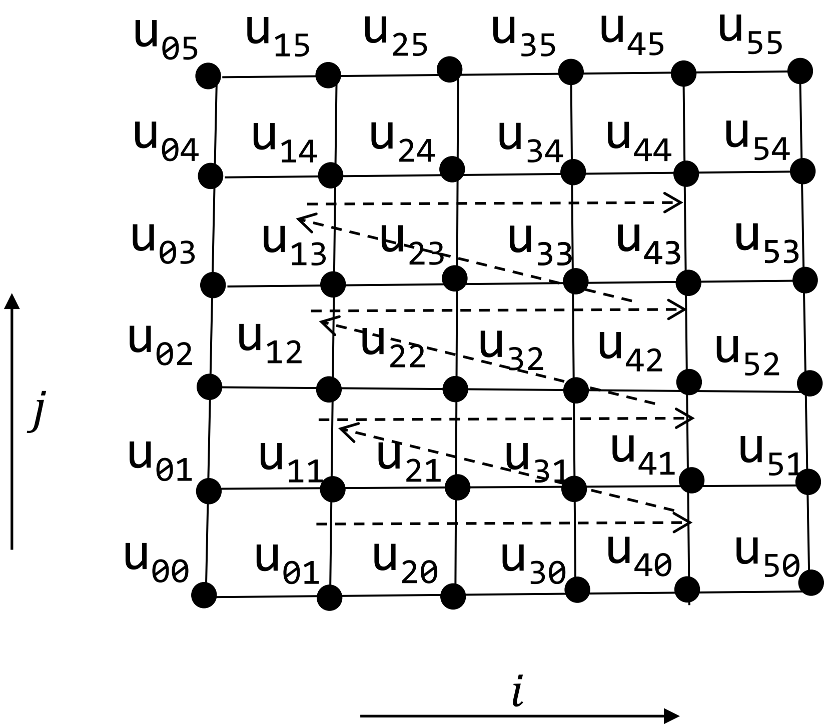
$u_{xx} + uyy = 0$ in the square $0 < x < 1, 0 < y < 1$

$u = x^2 y$ 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.

$$\begin{pmatrix} -4 & 1 & 1 & 0 \\ 1 & -4 & 0 & 1 \\ 1 & 0 & -4 & 1 \\ 0 & 1 & 1 & -4 \end{pmatrix} = \underbrace{\begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix}}_{L} + \underbrace{\begin{pmatrix} -4 & 0 & 0 & 0 \\ 0 & -4 & 0 & 0 \\ 0 & 0 & -4 & 0 \\ 0 & 0 & 0 & -4 \end{pmatrix}}_{D} + \underbrace{\begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}}_{U}$$

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

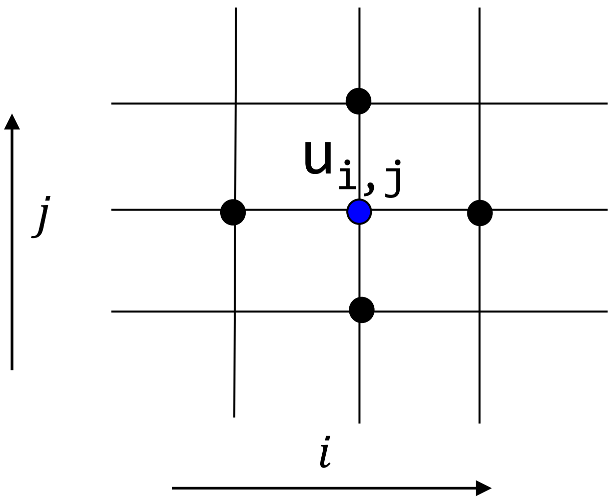
u_{ij} 's value in $(\mathbf{1})^{st}$ iteration is computed based on u_{ij} values computed in $(\mathbf{0})^{th}$ iteration

Background – Jacobi Iteration

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

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

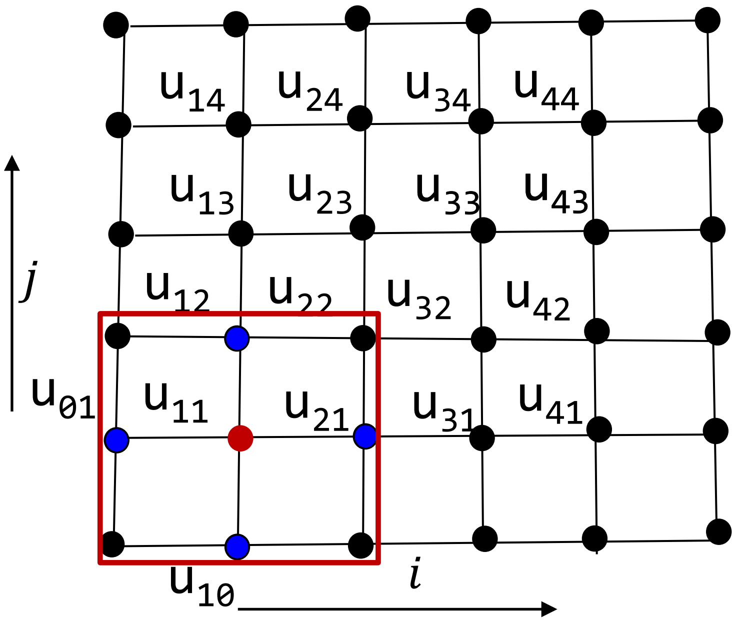
- $u_{right} + u_{top} - 4u_{center} + u_{left} + u_{bottom} = 0$

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

- Applying Jacobi Iteration:
  - $u_{center}^{(k+1)} = 1/4(u_{center}^{(k)} + u_{top}^{(k)} + u_{left}^{(k)} + u_{bottom}^{(k)})$

Computing Stencil – Recap

- Example: applying Jacobi Iteration:
  - $u_{center}^{(k+1)} = 1/4(u_{center}^{(k)} + u_{top}^{(k)} + u_{left}^{(k)} + u_{bottom}^{(k)})$

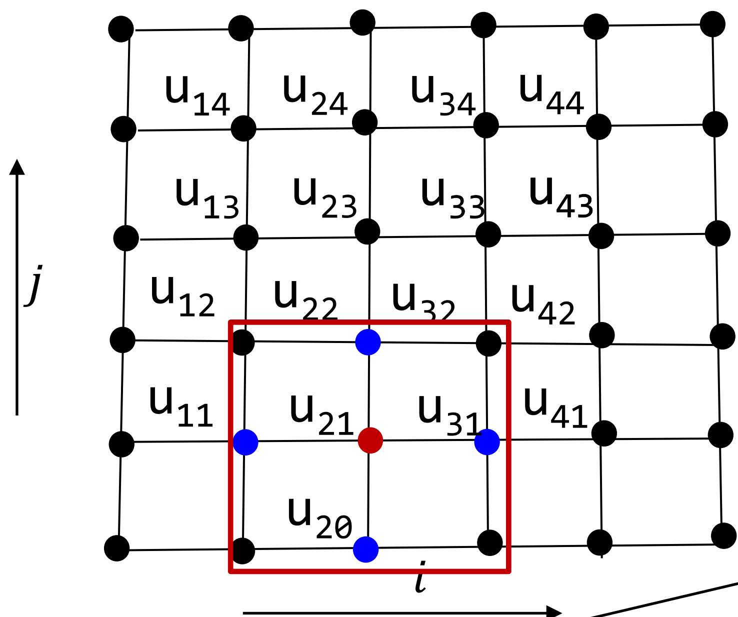


Iteration 1

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



Iteration 1

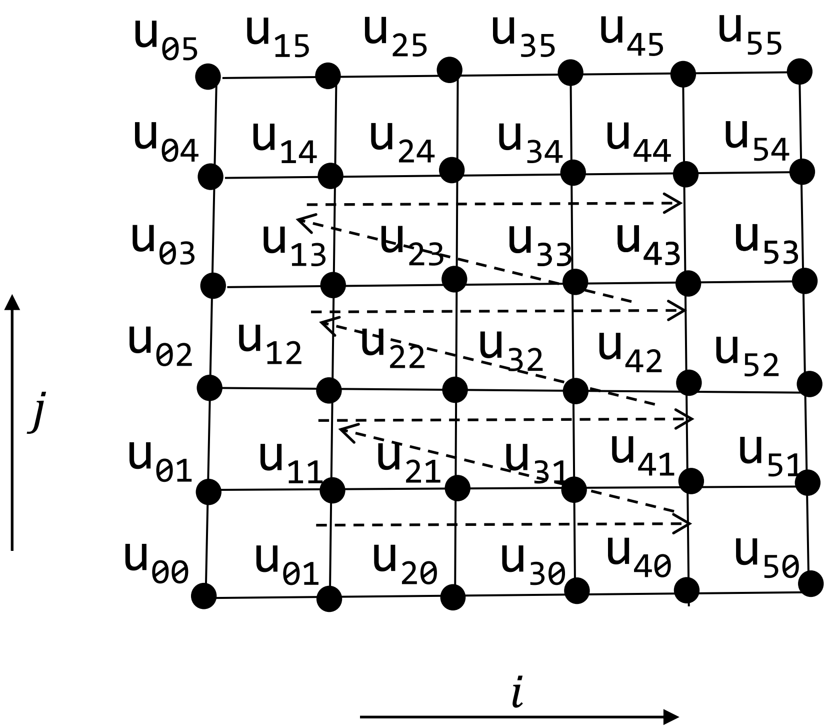
1) Compute u_{11} using initial guess for u_{12} and u_{21}. u_{01} and u_{10} are known from boundary conditions

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

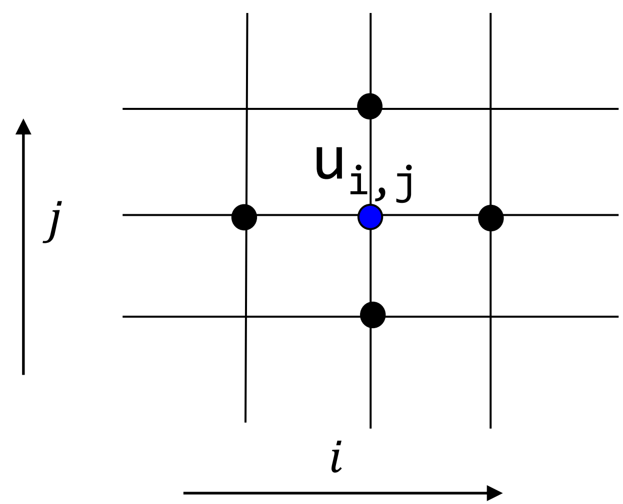
In 2), note that the initial guess for u_{11} is used even though u_{11} was updated just before in 1)

Elliptic Equation – Computing Stencil

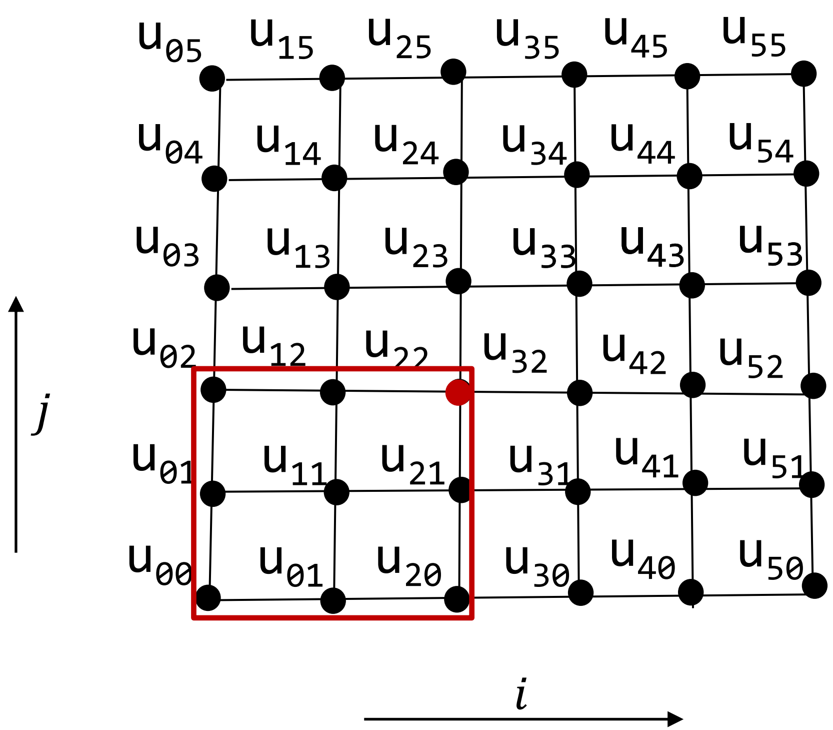
- In every iteration, suppose we follow the computing order as shown (dashed):



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

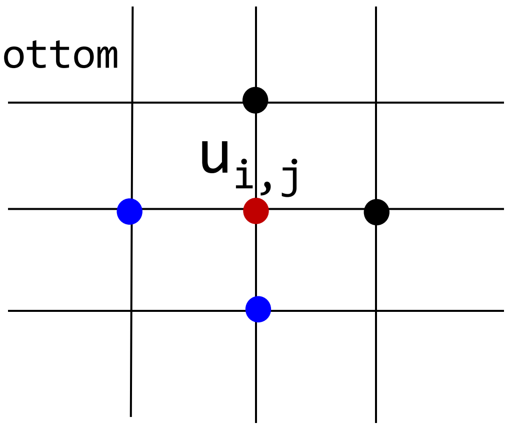


Elliptic Equation – Computing Stencil



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
  - $\boldsymbol{u_{center}^{(k+1)}} = 1/4(\boldsymbol{u_{center}^{(k)}} + \boldsymbol{u_{top}^{(k)}} + \boldsymbol{u_{left}^{(k+1)}} + \boldsymbol{u_{bottom}^{(k+1)}})$
- Gauss-Seidel: Observations
  - For a given problem and initial guess, Gauss-seidel *converges faster* than Jacobi
  - An iteration in Jacobi can be parallelized

Program Representation – Structured Grids

- Requirements:
  - Grid step size shall not be hardcoded E.g. h=1/3, h=1/5 etc.
    - Consequence: can't define int arr[m][n]; //m,n to be constant expr.
  - Grid dimension shall not be hardcoded
    - Consequence: implementations must define a compile-time constant
  - A grid point shall be identified with cartesian coordinates / polar coordinates (e.g. with angle and radius from origin)
    - Shall be able to generate a structured grid given number of points, xi, and eta.
  - Shall allow access to any grid point
  - Shall allow for implementation of grid operators

Structured Grids - Representation

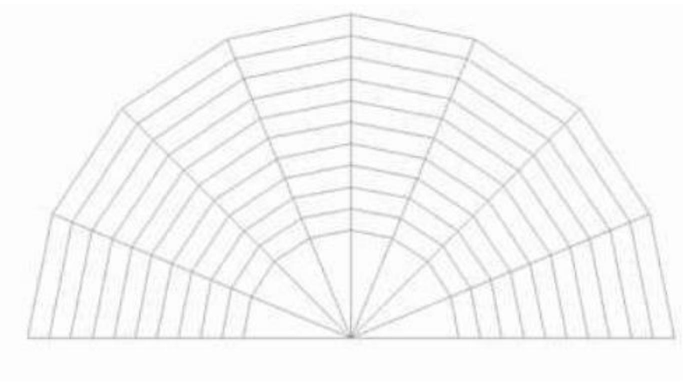
- Because of regular connectivity between cells
  - Cells can be identified with indices (x,y) or (x,y,z) and neighboring cell info can be obtained.
  - How about identifying a cell here?

    Given:

    ξ = ("Xi") radius

    η = ("Eta") angle

    Compute:



$$x = \left(\frac{1}{2} + \xi\right)\cos(\pi\eta)$$

$$y = \left(\frac{1}{2} + \xi\right)\sin(\pi\eta)$$

class `Domain`

- We discretize the domain using a grid

```
class Domain{
    public:
       generate_grid(int m, int n);
       Domain(); // constructor
       //...
    private:
       //...
};
```

Method `generate_grid`

- What is the shortcoming of the following method?

```
void Domain::GenerateGrid(int m, int n) {
    if (m <=0 || n<=0)
      throw std::invalid_argument("ERR_generate_grid");
    else if( m > 0) {
//there already exists a grid! Attempt to create a grid again
             delete [] x; delete [] y;
    }
    xlen=m;ylen=n; // initialize members
    x=new double[xlen*ylen]; y=new double[xlen*ylen];
}
```

Grid Function

- We let a grid function to operate on the grid points
  - Example of an operator: numerical differentiation
  - Different operations possible
  - Note: grid function always operates on some grid.
  - Many functions may operate on the same grid.

```
class GridFn{
    public:
       //...
    private:
       Domain* d; //aggregation
       //...
};
```

Boundary conditions

- Multiple options: affect the accuracy of the solution

Name	Prescription	Interpretation
Dirichlet	u	Fixed temperature
Neumann	$\partial u/\partial n$	Energy Flow
Mixed	$\partial u/\partial n + f(u)$	Temperature dependent flow

- How to represent boundary conditions?

Solution

- Heat equation 1D: How about?

```
Domain dom;
GridFn g(dom);
Solution u(&gridfn)
u.initcond() //set initial conditions
for(int step=0; step<maxsteps; step ++) {
        u += δt * u.D2X();
        t += δt;
        u.boundarycond();
}
```

class `Solution`

- We discretize the domain using a grid

```
class Solution{
    public:
       Solution(GridFn* d): sol(d) {}
       initcond();
       boundarycond();
    private:
       GridFn* sol;
};
```

What is missing?

- Data array?
- Type of data as template parameter?
- Operation on subgrids (Box)?