

CMPSC/Math 451, Numerical Computation

Wen Shen

Department of Mathematics, Penn State University

Finite Difference Methods for Some Partial Differential Equations

In this chapter we consider several well-known second order linear partial differential equations, and study finite difference approximations.

These include:

- Laplace Equation in 2D, Poisson Equation in 2D, with Dirichlet and/or Neumann boundary conditions.
- Heat equation in 1D, or other simple parabolic type PDE in 1D, with various boundary conditions.

Laplace Equation in 2D: Finite Difference Methods

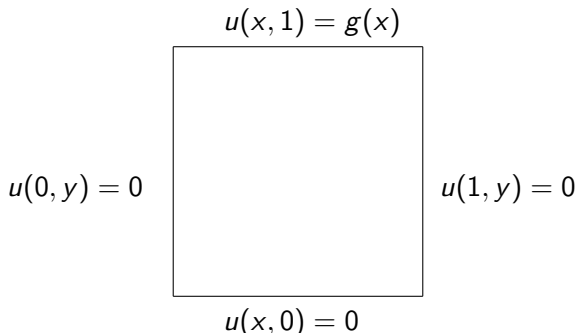
We consider the Laplace equation on a unit square

$$u_{xx} + u_{yy} = 0, \quad (0 < x < 1, \quad 0 < y < 1),$$

with boundary Dirichlet conditions, homogeneous on three sides, namely

$$u(0, y) = 0, \quad u(1, y) = 0, \quad (0 \leq x \leq 1)$$

$$u(x, 0) = 0, \quad u(x, 1) = g(x), \quad (0 \leq y \leq 1)$$



We use a uniform grid in both x and y direction.

Choose N , the number of intervals in x and y directions, and let

$$h = 1/N, \quad x_i = ih, \quad y_j = jh, \quad i, j = 0, 1, 2, \dots, N.$$

Here h is the grid size.

Our goal is to find approximate solution at the grid points only, i.e., we want to find $u_{i,j} \approx u(x_i, y_j)$.

Finite differences approximations to the partial derivatives at the grid point (x_i, y_j) are

$$u_{xx}(x_i, y_j) \approx \frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{h^2},$$

$$u_{yy}(x_i, y_j) \approx \frac{u_{i,j-1} - 2u_{i,j} + u_{i,j+1}}{h^2}.$$

Plugging these approximations into the Laplace equation at the point (x_i, y_j) , we get

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

After some simplification, we get

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

We see that the discrete equation (1) holds at every grid point (x_i, y_j) that is not on the boundary, i.e., $i = 1, 2, \dots, N-1, j = 1, 2, \dots, N-1$.

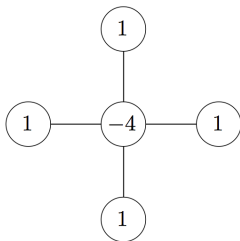
This gives us $(N-1) \times (N-1)$ equations.

The discrete boundary conditions are

$$u_{0,j} = 0, \quad u_{N,j} = 0, \quad 0 \leq j \leq N$$

$$u_{i,0} = 0, \quad u_{i,N} = g_i = g(x_i), \quad 0 \leq i \leq N.$$

Each discrete equation in (1) involves 5 grid points, which forms a **computational stencil**:



Note that only the interior points will be the unknowns, and the number of unknowns is $(N - 1)^2$. which matches the number of the equations.

This leads to a $(N - 1)^2 \times (N - 1)^2$ system of linear equations. – next video.

System of Linear Equations for Discrete Laplace Equation

The $(N-1)^2 \times (N-1)^2$ system of linear equations can be written as $A\vec{v} = \vec{b}$ where $\vec{v} = (v_1, v_2, \dots, v_{(N-1)^2})^T$.

To form the unknown vector \vec{v} out of the double indexed data u_{ij} , we go through a process called **natural ordering**.

We sweep through x-direction, then y-direction, as follows:

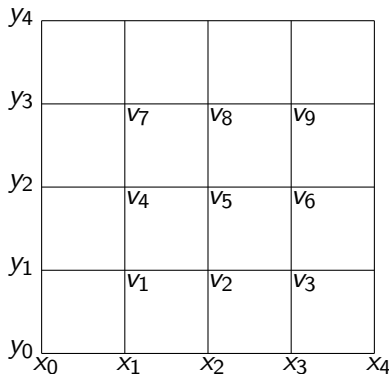
$$\begin{aligned}\vec{v} &= (v_1, v_2, \dots, v_{(N-1)^2})^T \\ &= (u_{1,1}, u_{2,1}, \dots, u_{N-1,1}, u_{1,2}, u_{2,2}, \dots, u_{N-1,2}, \dots, u_{N-1,N-1})^T\end{aligned}$$

Take for example $N = 4$. Total number unknowns is 9, and the unknown vector is

$$v_1 = u_{1,1}, \quad v_2 = u_{2,1}, \quad v_3 = u_{3,1},$$

$$v_4 = u_{1,2}, \quad v_5 = u_{2,2}, \quad v_6 = u_{3,2},$$

$$v_7 = u_{1,3}, \quad v_8 = u_{2,3}, \quad v_9 = u_{3,3}.$$



We can form the equations by using the computational stencil, placing it over each interior grid point, and write out the equation.

We must take consideration of the boundary conditions, and move them to the righthand side.

We obtain 9 equations:

$$-4v_1 + v_2 + v_4 = 0$$

$$v_1 - 4v_2 + v_3 + v_5 = 0$$

$$v_2 - 4v_3 + v_6 = 0$$

$$v_1 - 4v_4 + v_5 + v_7 = 0$$

$$v_2 + v_4 - 4v_5 + v_6 + v_8 = 0$$

$$v_3 + v_5 - 4v_6 + v_9 = 0$$

$$v_4 - 4v_7 + v_8 = -g(x_1)$$

$$v_5 + v_7 - 4v_8 + v_9 = -g(x_2)$$

$$v_6 + v_8 - 4v_9 = -g(x_3)$$

We can write them out in the following more organized way:

$-4v_1$	$+v_2$		$+v_4$			$=$	0
v_1	$-4v_2$	$+v_3$		$+v_5$		$=$	0
	v_2	$-4v_3$			$+v_6$	$=$	0
v_1			$-4v_4$	$+v_5$		$+v_7$	$= 0$
	v_2		$+v_4$	$-4v_5$	$+v_6$	$+v_8$	$= 0$
		v_3		$+v_5$	$-4v_6$	$+v_9$	$= 0$
			v_4			$-4v_7$	$+v_8 = -g(x_1)$
				v_5		$+v_7$	$-4v_8 + v_9 = -g(x_2)$
					v_6	$+v_8$	$-4v_9 = -g(x_3)$

This gives us a linear system $A\vec{v} = b$, where

$$A = \left(\begin{array}{ccc|ccc|ccc} -4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -4 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -4 & 0 & 0 & 1 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & -4 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & -4 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & -4 & 0 & 0 & 1 \\ \hline 0 & 0 & 0 & 1 & 0 & 0 & -4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & -4 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -4 \end{array} \right), \quad \vec{b} = \left(\begin{array}{c} 0 \\ 0 \\ 0 \\ \hline 0 \\ 0 \\ 0 \\ \hline -g(x_1) \\ -g(x_2) \\ -g(x_3) \end{array} \right)$$

Note the tri-diagonal block-structure of the A matrix!
The matrix A is symmetric, and diagonal dominant.

In the general case, the same block-structure is preserved, if we form the unknown vector using natural ordering.

Let D be an $(N - 1) \times (N - 1)$ tri-diagonal matrix with -4 on the diagonal and 1 on the sup and sub diagonal.

Let I be an $(N - 1) \times (N - 1)$ identity matrix.

Then, the A matrix is block-tridiagonal, $(N - 1) \times (N - 1)$, block structured:

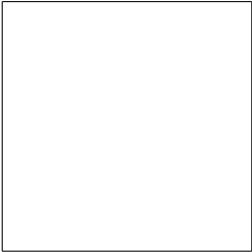
$$A = \begin{pmatrix} D & I & & & \\ I & D & I & & \\ & \ddots & \ddots & \ddots & \\ & & I & D & I \\ & & & I & D \end{pmatrix}$$

The final dimension of A is $(N - 1)^2 \times (N - 1)^2$.

In the load vector b we will collect all boundary conditions.

Laplace equation with non-homogeneous Dirichlet BCs

Consider the Laplace equation with non-homogeneous Dirichlet boundary conditions on all 4 sides of the unit squares:


$$\begin{aligned} u(x, 1) &= g(x) \\ u(0, y) &= m(y) \\ u(1, y) &= f(y) \\ u(x, 0) &= h(x) \end{aligned}$$

Since the boundary conditions only enter the load vector \vec{b} , the coefficient matrix A remains unchanged.

One may go through the grid again with the computational stencil, and collect all the boundary terms and move them to the load vector.

With $N = 4$, this gives us

		$g(x_1)$	$g(x_2)$	$g(x_3)$	
$m(y_3)$		v_7	v_8	v_9	$f(y_3)$
$m(y_2)$		v_4	v_5	v_6	$f(y_2)$
$m(y_1)$		v_1	v_2	v_3	$f(y_1)$
		$h(x_1)$	$h(x_2)$	$h(x_3)$	

$$\vec{b} = \begin{pmatrix} -h(x_1) - m(y_1) \\ -h(x_2) \\ -h(x_3) - f(y_1) \\ -m(y_2) \\ 0 \\ -f(y_2) \\ -g(x_1) - m(y_3) \\ -g(x_2) \\ -g(x_3) - f(y_3) \end{pmatrix}$$

$$N = 5 : \quad \vec{b} = \begin{pmatrix} -h(x_1) - m(y_1) \\ -h(x_2) \\ -h(x_3) \\ -h(x_4) - f(y_1) \\ \hline -m(y_2) \\ 0 \\ 0 \\ 0 \\ -f(y_2) \\ \hline -m(y_3) \\ 0 \\ 0 \\ 0 \\ -f(y_3) \\ \hline -g(x_1) - m(y_4) \\ -g(x_2) \\ -g(x_3) \\ -g(x_4) - f(y_4) \end{pmatrix} .$$

The pattern shall be clear by now.

Poisson equation on a unit square with Dirichlet boundary condition

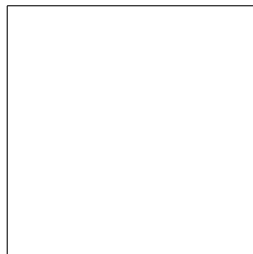
We now consider the Poisson equation on a unit square

$$u_{xx} + u_{yy} = \phi(x, y)$$

with Dirichlet boundary condition, given as

$$u(x, 1) = g(x)$$

$$u(0, y) = m(y)$$



$$u(1, y) = f(y)$$

$$u(x, 0) = h(x)$$

Finite different discretization gives

$$\frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{h^2} + \frac{u_{i,j-1} - 2u_{i,j} + u_{i,j+1}}{h^2} = \phi(x_i, y_j) = \phi_{ij} \quad (1)$$

so

$$u_{i-1,j} + u_{i+1,j} - 4u_{i,j} + u_{i,j-1} + u_{i,j+1} = h^2 \phi_{ij}, \quad i, j = 1, 2, \dots, N-1 \quad (2)$$

We see that the left-hand side of the discrete equation is unchanged. The source term $h^2 \phi_{ij}$ will only enter the load vector.

Take for example $N = 4$, we get

$$\vec{b} = \begin{pmatrix} h^2\phi_{1,1} - h(x_1) - m(y_1) \\ h^2\phi_{2,1} - h(x_2) \\ h^2\phi_{3,1} - h(x_3) - f(y_1) \\ \hline h^2\phi_{1,2} - m(y_2) \\ h^2\phi_{2,2} \\ h^2\phi_{3,2} - f(y_2) \\ \hline h^2\phi_{1,3} - g(x_1) - m(y_3) \\ h^2\phi_{2,3} - g(x_2) \\ h^2\phi_{3,3} - g(x_3) - f(y_3) \end{pmatrix}.$$

Laplace Equation with Neumann Boundary Condition

Consider the Laplace equation

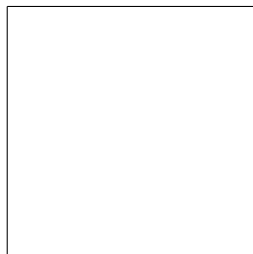
$$u_{xx} + u_{yy} = 0$$

on a unit square, with boundary conditions

$$u(x, 1) = g(x)$$

$$u_x(0, y) = a(y)$$

$$u(1, y) = f(y)$$



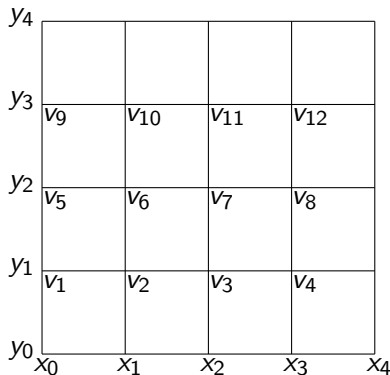
$$u(x, 0) = h(x)$$

We assign Neumann boundary condition on one side of the square.

This changes quite a bit the discretization.

Since now $u_{0,j}$ are also unknowns, we need to take them into the ordering.

The natural ordering is illustrated below:



In order to use a second order central finite difference for the Neumann boundary condition, we add a layer of ghost boundary, i.e.,

$$u_{-1,j}, \quad \text{for } j = 1, 2, \dots, N-1.$$

A central finite difference for the boundary condition can be

$$u_x(0, y_j) \approx \frac{u_{1,j} - u_{-1,j}}{2h} = a(y_j) = a_j, \quad \Rightarrow \quad u_{-1,j} = u_{1,j} - 2ha_j.$$

The discrete Laplace equation at $i = 0$ for any j gives

$$u_{-1,j} - 4u_{0,j} + u_{1,j} + u_{0,j-1} + u_{0,j+1} = 0.$$

Eliminating the ghost $u_{-1,j}$ from the previous two equations, we get

$$u_{1,j} - 2ha_j - 4u_{0,j} + u_{1,j} + u_{0,j-1} + u_{0,j+1} = 0,$$

$$\Rightarrow \quad -4u_{0,j} + 2u_{1,j} + u_{0,j-1} + u_{0,j+1} = 2ha_j.$$

This leads to a system of linear equations $A\vec{v} = \vec{b}$.

For $N = 4$, we get

$$A = \left(\begin{array}{cccc|cccc|cccc} -4 & 2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -4 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & -4 & 2 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 & 0 & 0 & 0 & 1 \\ \hline 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -4 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 \end{array} \right)$$

We see that A still preserves the blocked tri-diagonal structure.

The load vector

$$\vec{b} = \begin{pmatrix} -h(x_0) + 2ha_1 \\ -h(x_1) \\ -h(x_2) \\ -h(x_3) - f(y_1) \\ \hline 2ha_2 \\ 0 \\ 0 \\ -f(y_2) \\ \hline -g(x_0) + 2ha_3 \\ -g(x_1) \\ -g(x_2) \\ -g(x_3) - f(y_3) \end{pmatrix}.$$

The pattern for A and \vec{b} with larger value of N is now clear.

Heat Equation in 1D

We consider a simple heat equation, where $u(t, x)$ is the temperature in a rod:

$$u_t = u_{xx}, \quad 0 \leq x \leq 1$$

with boundary condition

$$u(t, 0) = 0, \quad u(t, 1) = 0, \quad t \geq 0$$

and initial condition

$$u(0, x) = f(x).$$

Set up the grid in x :

$$\Delta x = \frac{1}{M}, \quad x_j = j\Delta x, \quad j = 0, 1, 2, \dots, M$$

and the grid in t :

$$t_0 = 0, \quad t_n = n\Delta t, \quad n = 0, 1, 2, \dots$$

Goal: Find approximation to solution $u_j^n \approx u(t_n, x_j)$.

Tool: finite differences.

$$u_t(t_n, x_j) \approx \frac{u_j^{n+1} - u_j^n}{\Delta t}, \quad u_{xx}(t_n, x_j) \approx \frac{u_{j-1}^n - 2u_j^n + u_{j+1}^n}{\Delta x^2}$$

Forward-Euler (Explicit Euler):

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = \frac{u_{j-1}^n - 2u_j^n + u_{j+1}^n}{\Delta x^2}.$$

We can clean up a bit. Writing $\gamma = \Delta t / \Delta x^2$, we get

$$u_j^{n+1} = \gamma u_{j-1}^n + (1 - 2\gamma)u_j^n + \gamma u_{j+1}^n, \quad (1)$$

with initial and boundary conditions

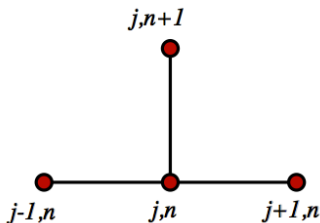
$$u_j^0 = f(x_j), \quad j = 0, 1, 2, \dots, M$$

and boundary conditions

$$u_0^n = 0, \quad u_M^n = 0, \quad n \geq 0.$$

The method is first order in time and second order in space, i.e., $\mathcal{O}(\Delta t, \Delta x^2)$.

The computational stencil:



Stability condition for explicit scheme

Maximum Principle of Heat Equation. The exact solution $u(t, x)$ of the heat equation satisfies the following maximum principle:

$$\min_{0 \leq y \leq 1} u(t_1, y) \leq u(t_2, x) \leq \max_{0 \leq y \leq 1} u(t_1, y) \quad (1)$$

for any $t_2 \geq t_1$.

In particular, (1) implies

$$\max_{0 \leq x \leq 1} |u(t_2, x)| \leq \max_{0 \leq x \leq 1} |u(t_1, x)|, \quad (2)$$

for any $t_2 \geq t_1$.

\Rightarrow The maximum value of $|u(t, x)|$ over x is non-increasing in time t .

Discrete Maximum principle.

It is desirable to require a discrete version of the maximum principle for our approximate solution, i.e.,

$$\max_j \left| u_j^{n+1} \right| \leq \max_j \left| u_j^n \right|, \quad \text{for every } n. \quad (3)$$

We now provide a sufficient condition for (3). Assume now

$$1 - 2\gamma \geq 0, \quad \text{i.e.} \quad \gamma \leq \frac{1}{2}, \quad \Rightarrow \quad \Delta t \leq \frac{1}{2} \Delta x^2. \quad (4)$$

This is called the **CFL stability condition**.

Then

$$\begin{aligned} |u_j^{n+1}| &\leq \gamma |u_{j-1}^n| + (1 - 2\gamma) |u_j^n| + \gamma |u_{j+1}^n| \\ &\leq \gamma \max_i |u_i^n| + (1 - 2\gamma) \max_i |u_i^n| + \gamma \max_i |u_i^n| \\ &= \max_i |u_i^n|. \end{aligned}$$

This means

$$|u_j^{n+1}| \leq \max_i |u_i^n|, \quad \text{for every } j.$$

Since this holds for all j , we conclude

$$\max_j |u_j^{n+1}| \leq \max_j |u_j^n|, \quad \text{for every } n.$$

The CFL stability condition

$$\Delta t \leq \frac{1}{2}(\Delta x)^2$$

It actually puts a very strict constraint on the time step size Δt , especially when space grid size Δx is small.

For example, if $\Delta x = 10^{-3}$, then the time step size Δt must be $\Delta t \leq (\frac{1}{2})10^{-6}$, which is extremely small!

We are now forced to take many many time steps!

Backward-Euler (Implicit-Euler)

We instead use:

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = \frac{u_{j-1}^{n+1} - 2u_j^{n+1} + u_{j+1}^{n+1}}{\Delta x^2}.$$

Writing $\gamma = \Delta t / \Delta x^2$, we can write

$$-\gamma u_{j-1}^{n+1} + (1 + 2\gamma)u_j^{n+1} - \gamma u_{j+1}^{n+1} = u_j^n, \quad (1)$$

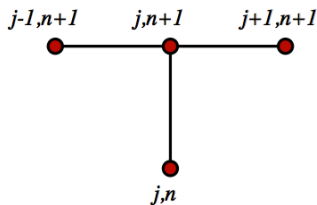
with initial and boundary conditions

$$u_j^0 = f(x_j), \quad u_0^n = 0, \quad u_M^n = 0.$$

(1) gives a tri-diagonal system to solve at every time step.

The method is first order in time and second order in space, i.e., of $\mathcal{O}(\Delta t, \Delta x^2)$.

Computational stencil:



Discrete Maximum Principle:

Scheme (1) could also be written as

$$(1 + 2\gamma)u_j^{n+1} = u_j^n + \gamma u_{j-1}^{n+1} + \gamma u_{j+1}^{n+1}.$$

$$\begin{aligned}(1 + 2\gamma) \left| u_j^{n+1} \right| &\leq \left| u_j^n \right| + \gamma \left| u_{j-1}^{n+1} \right| + \gamma \left| u_{j+1}^{n+1} \right| \\ &\leq \max_i \left| u_i^n \right| + \gamma \max_i \left| u_i^{n+1} \right| + \gamma \max_i \left| u_i^{n+1} \right| \\ &= \max_i \left| u_i^n \right| + 2\gamma \max_i \left| u_i^{n+1} \right|\end{aligned}$$

Since this holds for all j , it also holds when the left reaches the max. We conclude

$$\begin{aligned}(1 + 2\gamma) \max_j \left| u_j^{n+1} \right| &\leq \max_j \left| u_j^n \right| + 2\gamma \max_j \left| u_j^{n+1} \right| \\ \rightarrow \max_j \left| u_j^{n+1} \right| &\leq \max_j \left| u_j^n \right|.\end{aligned}$$

Note that the Maximum Principle is satisfied for any choices of γ , i.e, any choices of grid size $\Delta t, \Delta x$!

This is called *unconditionally stable*.

Discussion:

+ Using implicit time step, we can take larger time step Δt , even though at each step it takes longer time to solve. In the end, it may still save us time!

- The method is of order $\mathcal{O}(\Delta t, \Delta x^2)$, meaning error $\approx C_1 \Delta t + C_2 \Delta x^2$. With large Δt , the error is larger.

Wish: A second order method both in time and space, and unconditionally stable.

Crank-Nicholson time step

Both explicit and implicit Euler are first order in time. Since the space discretization is second order, it would be desirable to get a method that is also second order in time.

Here we introduce a method that is also second order in time. We use

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = \frac{u_{j-1}^n - 2u_j^n + u_{j+1}^n}{2\Delta x^2} + \frac{u_{j-1}^{n+1} - 2u_j^{n+1} + u_{j+1}^{n+1}}{2\Delta x^2}$$

This is like central finite difference for the time derivative.

Writing $r = \Delta t / (2\Delta x^2)$, (note this is different from γ !), and cleaning up a bit, we get

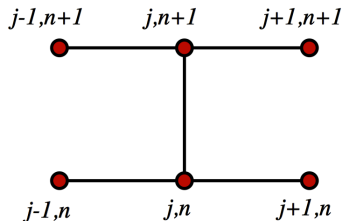
$$-ru_{j-1}^{n+1} + (1 + 2r)u_j^{n+1} - ru_{j+1}^{n+1} = ru_{j-1}^n + (1 - 2r)u_j^n + ru_{j+1}^n \quad (1)$$

with initial and boundary conditions

$$u_j^0 = f(x_j), \quad u_0^n = 0, \quad u_M^n = 0.$$

Note that scheme (1) gives a tri-diagonal system to solve at every time step.

Computational stencil:



This method is second order in both time and space, i.e, of $\mathcal{O}(\Delta t^2, \Delta x^2)$.

One can prove that Crank-Nicolson's method is *unconditionally stable*, although the stability is NOT with respect to the discrete Maximum Principle.