Numerical solution of PDEs

Arne Morten Kvarving

Department of Mathematical Sciences
Norwegian University of Science and Technology

November 6 2007

Problem and solution strategy

- This far we have considered ordinary differential equations where the solution only depends on one free variable.
- We now consider partial differential equations - equations where the solution depends on two (in general several) free variables.
- A general format for second order quasilinear PDE in two variables:

$$au_{xx} + 2bu_{xy} + cu_{yy} = F(x, y, u_x, u_y)$$

These can again be categorized in three subclasses.

Classification of problems

- $ac - b^2 > 0$: We have an elliptic equation.
  Example: Laplace

$$\nabla^2 = 0, \quad (a = c = 1,\ b = 0)$$

- $ac - b^2 = 0$: We have a parabolic equation.
  Example: The heat equation

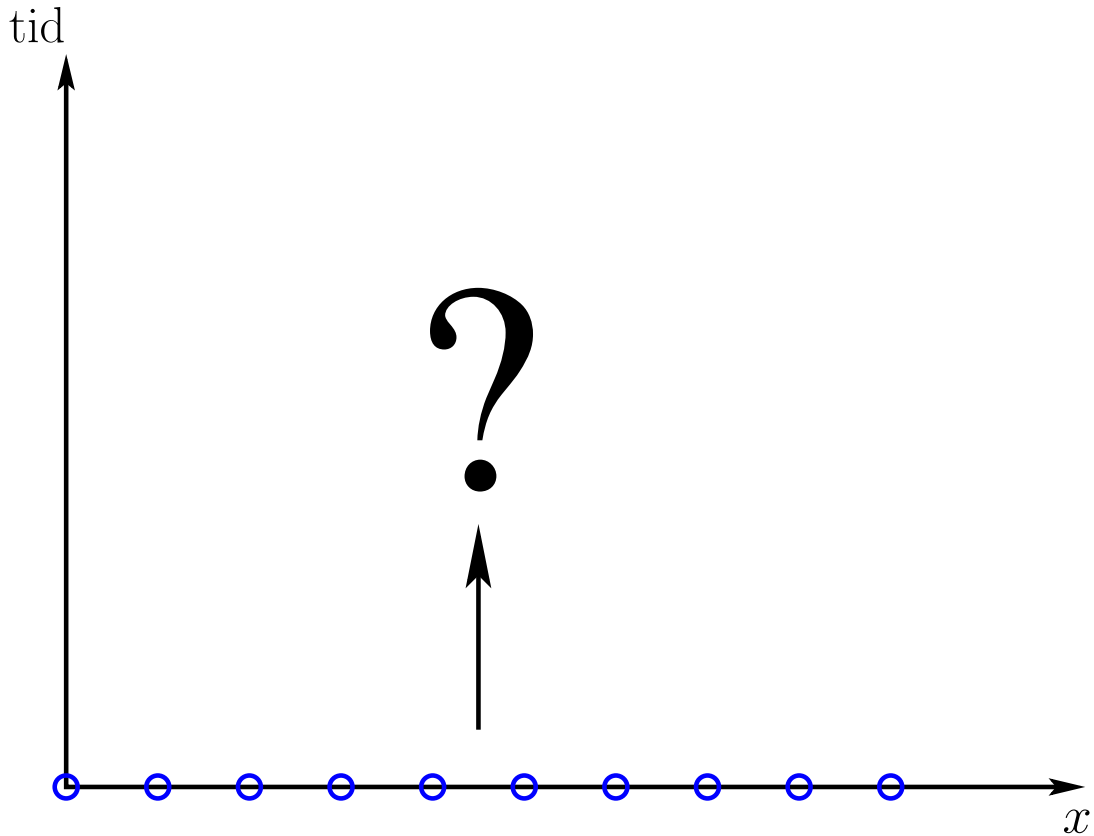
$$u_t = u_{xx}, \quad (a = 1,\ b = c = 0,\ F = u_t)$$

- $ac - b^2 < 0$: We have a hyperbolic equation.
  Example: The wave equation

$$u_{tt} = u_{xx}, \quad (a = 1,\ b = 0,\ c = -1)$$

These are not part of the curriculum.

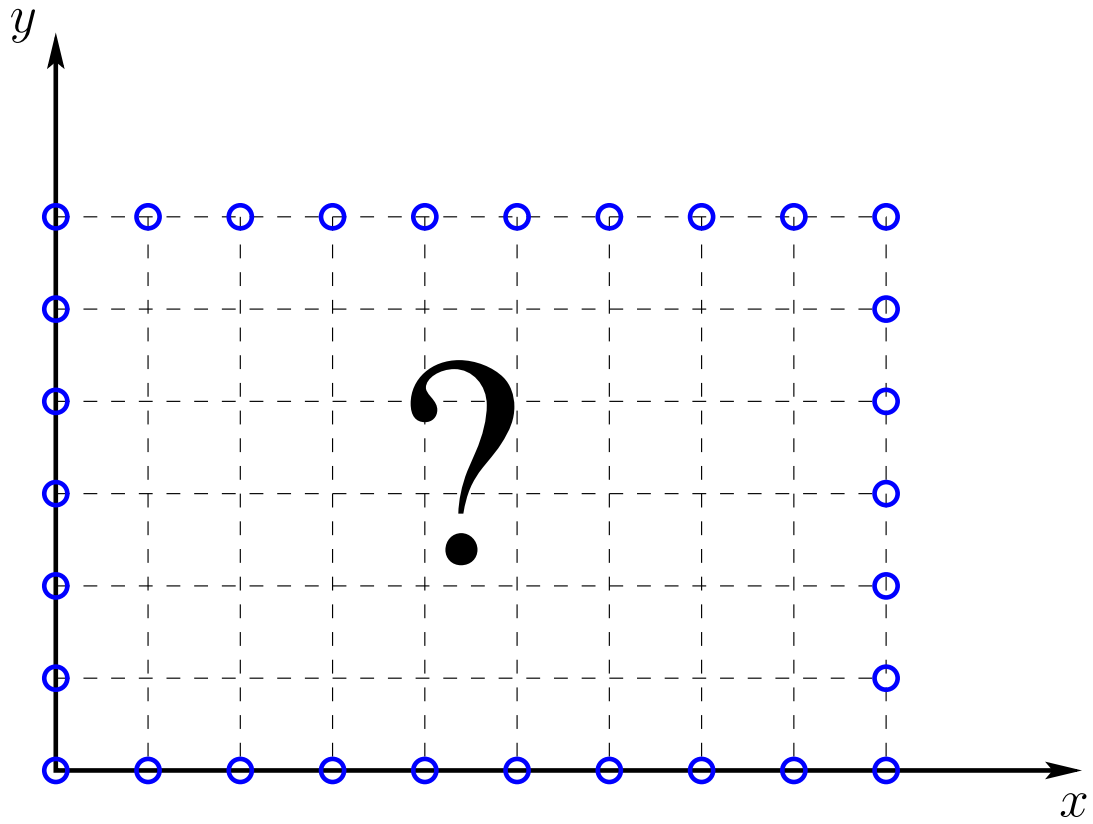
Classification of problems



Initial value problem - given a solution at $t = 0$, we seek to find the solution forward in time.

Classification of problems

- We now consider boundary value problems:



Fundamental difference!

Classification of problems

- Our problem now is: Given the values on the boundary, find the values within the domain.
- Two different types of boundary values we can be given are
  - Dirichlet boundary conditions: The solution is equal to a given function on the boundary. These are simple to handle.
  - Neumann boundary conditions: The derivative of the solution is known along the boundary. More difficult to handle, in particular in an implementation. Not part of the curriculum. One situation where these boundary values shows up is when we have a given heat flux over the boundary.

Solution strategy

- We introduce a grid and replace the derivative in the equation with numerical approximations. We then try to satisfy the equation approximately in the grid points. We end up with a system of linear equations we need to solve. The solution of this system is our approximation to the solution *in the grid points*. We do not obtain the solution inside the grid cells. If we need values within cells we have to resort to interpolation.
- These methods are known as difference methods.
- They are very inflexible with respect to geometry - hard to use in irregular geometries.
- There are methods which gives you a solution which is valid everywhere within your domain and which handles irregular geometries elegantly - the finite element method. It is not part of the curriculum.

How to approximate the derivative numerically

- As seen earlier, we can approximate the derivative as

$$f'(x_i) = \frac{f(x_i + h) - f(x_i)}{h} + \mathcal{O}(h)$$

  This is called a forward difference. It is only first order accurate.
- A better approximation can be obtained by

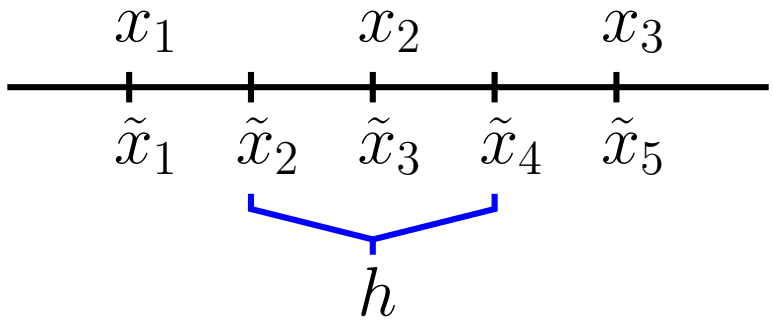
$$f'(x_i) = \frac{f\left(x_i + \frac{h}{2}\right) - f\left(x_i - \frac{h}{2}\right)}{h} + \mathcal{O}\left(h^2\right).$$

  This has one order higher precision, and the reason for that is that we choose points symmetrically around x_i.
- However, we are going to use these on a grid, and the grid would not include the point $x_i + \frac{h}{2}$. Hence we cannot use it to approximate the first derivative. But with a step size $2h$ we get

$$f'(x_i) = \frac{f\left(x_i + h\right) - f\left(x_i - h\right)}{2h} + \mathcal{O}\left(h^2\right)$$

How to approximate the derivative numerically

- The formula with step size h is still useful. Consider the grid



  We get

$$f'(\tilde{x}_2) = \frac{f(\tilde{x}_3) - f(\tilde{x}_1)}{h}$$
$$f'(\tilde{x}_4) = \frac{f(\tilde{x}_5) - f(\tilde{x}_3)}{h}.$$

- What happens if we use it to approximate the second derivative?

$$f''(x_2) = \frac{f'(\tilde{x}_4) - f'(\tilde{x}_2)}{h}$$

How to approximate the derivative numerically

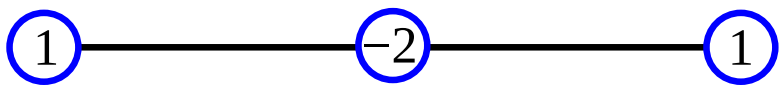
- We insert the expressions for the derivative

$$f''(x_2) = \frac{\left(\frac{f(\tilde{x}_3)-f(\tilde{x}_1)}{h}\right) - \left(\frac{f(\tilde{x}_5)-f(\tilde{x}_3)}{h}\right)}{h}$$
$$= \frac{f(\tilde{x}_5) + f(\tilde{x}_1) - 2f(\tilde{x}_3)}{h^2}$$

- But the points $\tilde{x}_1$, $\tilde{x}_3$ and $\tilde{x}_5$ are exactly the points that is part of our actual grid. This means that we can find our approximation of the second derivate as

$$f''(x_j) = \frac{f(x_{j+1}) + f(x_{j-1}) - 2f(x_j)}{h^2}.$$

- Such difference formulas are often represented as stencils.



Example - 1D Poisson

- To see how this works in a real life application, let us consider the one dimensional Poisson problem

$$u_{xx} = \cos\frac{\pi x}{2} \quad \text{in } \Omega = (0,1)$$
$$u(0) = 1$$
$$u(1) = 0.$$

- We introduce a grid with step size $h = 0.2$. This means that we have six grid points of which two lies on the boundary. We are given the boundary values hence the two grid points on the boundaries are not unknowns.

Example - 1D Poisson

- We proceed as previously announced. We replace the derivatives with numerical approximations and only consider the equation in the grid points. This yields

$$
\begin{aligned}
x = x_1: \quad & u_2 + u_0 - 2u_1 = h^2 \cos\frac{\pi h}{2} \\
x = x_2: \quad & u_3 + u_1 - 2u_2 = h^2 \cos\frac{\pi 2h}{2} \\
x = x_2: \quad & u_4 + u_2 - 2u_3 = h^2 \cos\frac{\pi 3h}{2} \\
x = x_4: \quad & u_5 + u_3 - 2u_4 = h^2 \cos\frac{\pi 4h}{2}
\end{aligned}
$$

Eksempel - 1D Poisson

- However, we know that $u_0 = 1$ and that $u_5 = -1$ which we can insert into the equations:

$$
\begin{aligned}
x = x_1 : \quad & u_2 - 2u_1 = h^2 \cos\frac{\pi h}{2} - 1 \\
x = x_2 : \quad & u_3 + u_1 - 2u_2 = h^2 \cos\frac{\pi 2h}{2} \\
x = x_2 : \quad & u_4 + u_2 - 2u_3 = h^2 \cos\frac{\pi 3h}{2} \\
x = x_3 : \quad & u_3 - 2u_4 = h^2 \cos\frac{\pi 4h}{2} - 0
\end{aligned}
$$

Example - 1D Poisson

- We can now state this as a matrix-vector system

$$\underline{A}\,\underline{u} = \underline{F}$$

$$\begin{bmatrix} -2 & 1 & & \\ 1 & -2 & 1 & \\ & 1 & -2 & 1 \\ & & 1 & -2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} h^2 \cos\frac{\pi h}{2} - u_0 \\ h^2 \cos\frac{\pi 2h}{2} \\ h^2 \cos\frac{\pi 3h}{2} \\ h^2 \cos\frac{\pi 4h}{2} - u_5 \end{bmatrix}$$

- We note that the matrix $\underline{A}$ is tridiagonal. The reason for this is that one equation only couples three unknowns (remember the stencil).
- This is structure which we can exploit when we want to solve the system.

The five point formula

- We now consider the operator

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}.$$

- As previously, we introduce a grid. We use a *local* numbering scheme. That is, we use two indices to identify a grid point,

$$u_i^j = u(x_i, y_j)$$

  To make the notation somewhat easier we will only consider grids where the step size is the same in both directions, denoted h.

The five point formula

- We can approximate the partial derivative using the central difference operator along each direction,

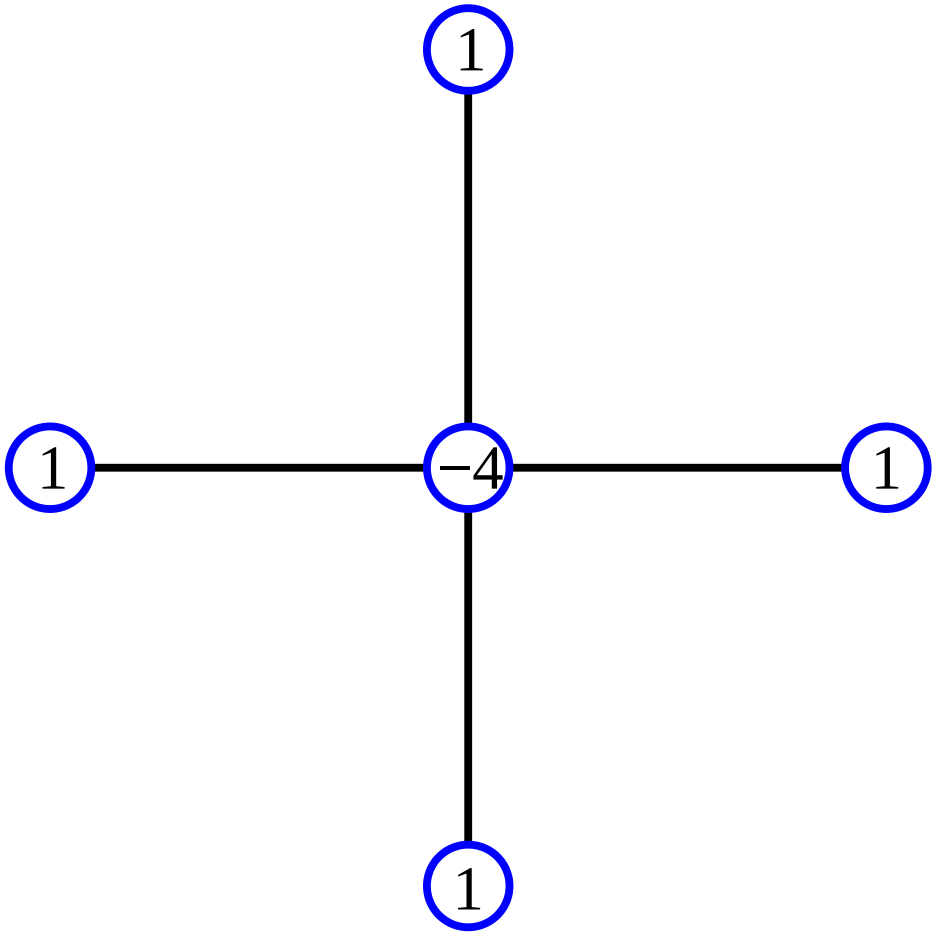
$$\frac{\partial^2}{\partial x^2} u(x_i, y_j) \approx \frac{u_{i+1}^j - 2u_i^j + u_{i-1}^j}{h^2}$$
$$\frac{\partial^2}{\partial y^2} u(x_i, y_j) \approx \frac{u_i^{j+1} - 2u_i^j + u_i^{j-1}}{h^2}$$

- We take the sum of these and get the famous *five point formula*.

$$\nabla^2 u(x_i, y_j) \approx \frac{u_{i+1}^j + u_{i-1}^j + u_i^{j+1} + u_i^{j-1} - 4u_i^j}{h^2}$$

It is named so since it uses five points to approximate the solution.

The five point formula



This is the stencil for the five point formula.

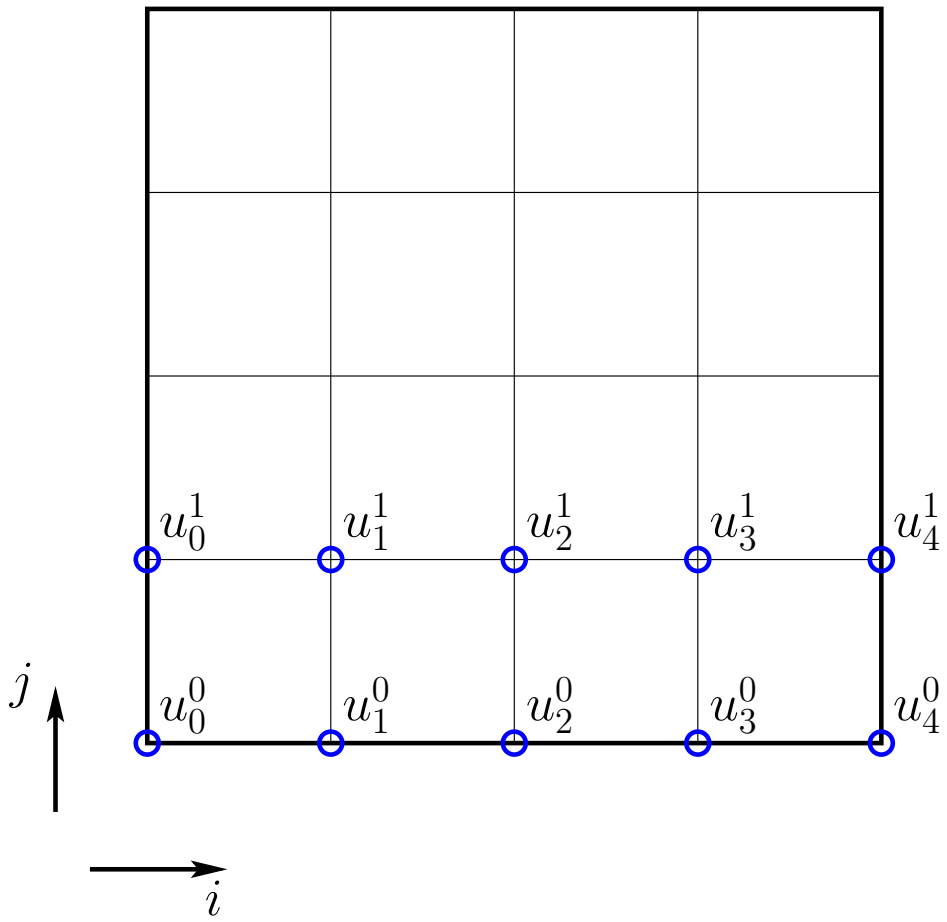
Example - 2D Poisson

- We now consider the problem

$$\begin{aligned} \nabla^2 u &= f, \quad \text{in } \Omega = (0,1) \times (0,1) \\ u(x,0) &= g_1(x) \\ u(1,y) &= g_2(y) \\ u(x,1) &= g_3(x) \\ u(0,y) &= g_4(y) \end{aligned}$$

- To make the presentation as easy to follow as possible, we use a grid with the minimal size needed to illustrate all the concepts involved. This means that we need five grid points in each direction which yields a total of 25 nodes of which 9 are unknowns.

Example - 2D Poisson

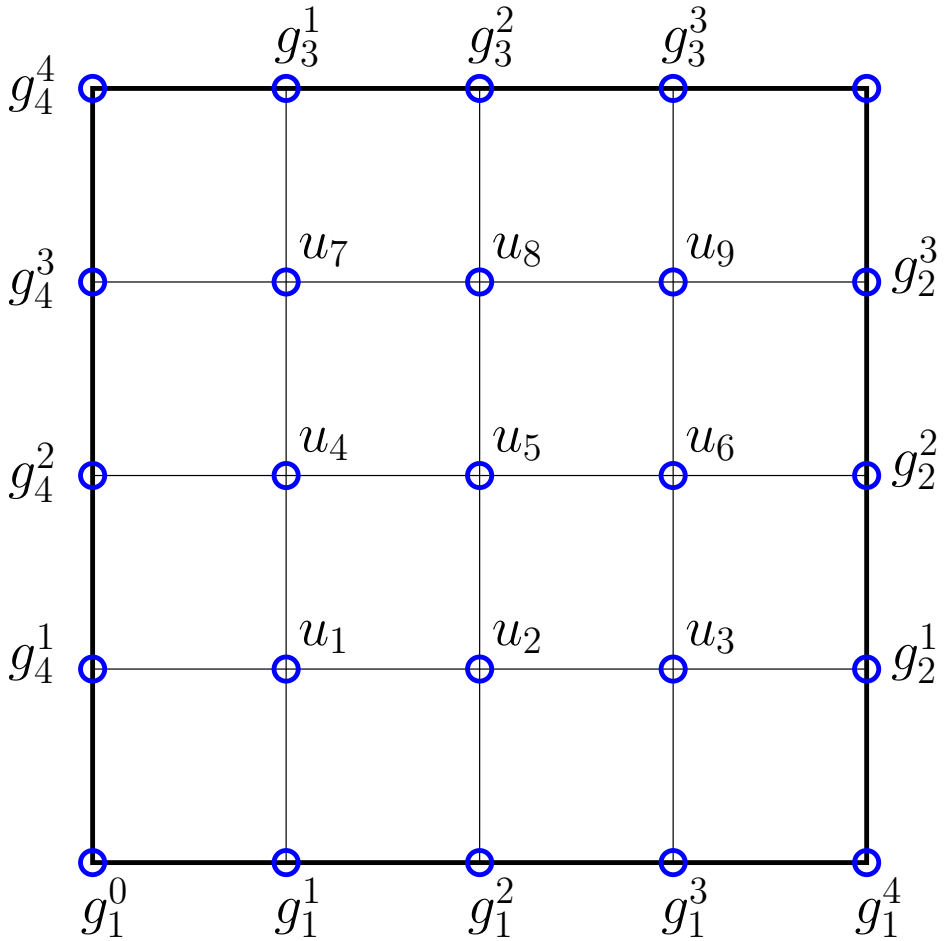


We number each node using two indices.

Example - 2D Poisson

- We get three classes of nodes
  - Nodes which couple to two boundary values - for instance node $(1, 1)$.
  - Nodes which couple to one boundary value - for instance node $(2, 1)$.
  - Nodes which couple to no boundary values - in this case the only node of this kind is node $(2, 2)$.
- We know that we end up with a linear system of equations we need to solve. Thus it is of preference to have a numbering scheme which corresponds to the vector components - known as a *global* numbering scheme.

Example - 2D Poisson



We only number the actual unknowns, successively along rows.

Example - 2D Poisson

- We approximate the solution in node 1:

$$u_2 + g_4^1 + u_4 + g_1^1 - 4u_1 = h^2 f_1 \Rightarrow$$
$$-4u_1 + u_2 + u_4 = h^2 f_1 - g_4^1 - g_1^1 \quad \text{couples 3 unknowns}$$

- We approximate the solution in node 2:

$$u_3 + u_1 + u_5 + g_1^2 - 4u_2 = h^2 f_2 \Rightarrow$$
$$u_1 - 4u_2 + u_3 + u_5 = h^2 f_2 - g_1^2 \quad \text{couples 4 unknowns}$$

- We approximate the solution in node 5:

$$u_2 + u_4 - 4u_5 + u_6 + u_8 = h^2 f_5 \qquad \text{couples 5 unknowns}$$

Example - 2D Poisson

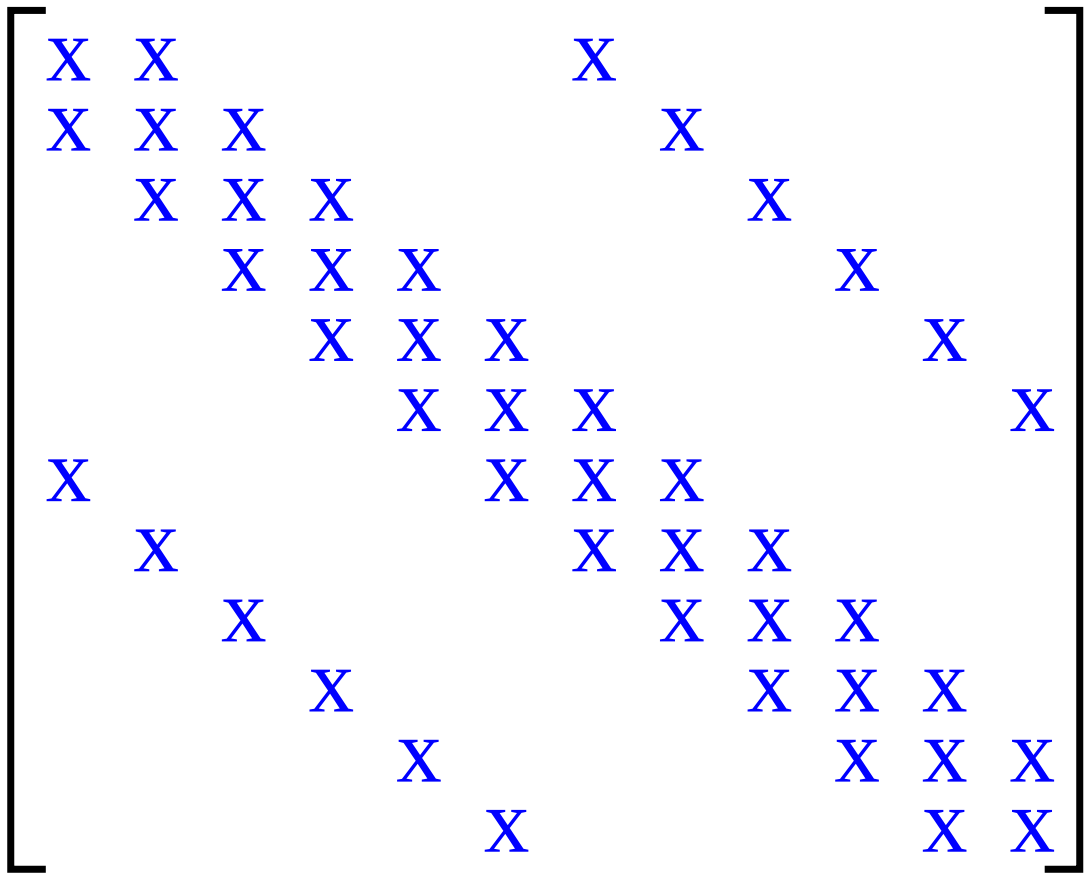
- In general this means that
  - For the top and bottom row we have two nodes of "type" 1.
  - We have a row of "type" 2 first and last because of the top and bottom boundary.
  - For every $N-2$ node we have two nodes of "type" 2 due to the left and right boundary. The exception is the first and last row since these elements are of "type" 1 there. Here N is the number of rows in one direction.
  - The rest of the nodes will be of "type" 3.

Example - 2D Poisson

$$
\left[\begin{array}{ccc|ccc|ccc}
-4 & 1 & & 1 & & & & & \\
1 & -4 & 1 & & 1 & & & & \\
 & 1 & -4 & & & 1 & & & \\
\hline
1 & & & -4 & 1 & & 1 & & \\
 & 1 & & 1 & -4 & 1 & & 1 & \\
 & & 1 & & 1 & -4 & & & 1 \\
\hline
 & & & 1 & & & -4 & 1 & \\
 & & & & 1 & & 1 & -4 & 1 \\
 & & & & & 1 & & 1 & -4
\end{array}\right]
\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \\ u_9 \end{bmatrix}
=
\begin{bmatrix} h^2 f_1 - g_4^1 - g_1^1 \\ h^2 f_2 - g_1^2 \\ h^2 f_3 - g_1^3 - g_2^1 \\ h^2 f_4 - g_2^2 \\ h^2 f_5 \\ h^2 f_6 - g_2^2 \\ h^2 f_7 - g_4^3 - g_3^1 \\ h^2 f_8 - g_3^2 \\ h^2 f_9 - g_2^3 - g_3^3 \end{bmatrix}
$$

Example - 2D Poisson

- Hopefully you are able to see the general system with more nodes.
- The matrix becomes sparse. General form



Simplified Gauss-Seidel iterations - exploiting the structure

- As previously stated, Gauss-Seidel iterations can be stated on component form as

$$x_j^{k+1} = \frac{1}{a_{jj}} \left(b - \sum_{i=1}^{j-1} a_{ji} x_i^{k+1} - \sum_{i=j+1}^{n} a_{ji} x_i^k \right).$$

- For our particular matrix, almost all the elements a_{ij} are zero. This we can exploit. For instance for node 5 we have that

$$u_5^{k+1} = \frac{1}{-4} \left(h^2 f_5 - u_2^{k+1} - u_4^{k+1} - u_6^k - u_8^k \right)$$

- There are *no* summations since we have control over the structure of the matrix.
- This formula exploits the sparseness of the matrix.

Simplified Gauss-Seidel iterations - exploiting the structure

- If we additionally use a sparse format for storing the matrix $\underline{A}$, we reduce the memory usage severely.

Format	$N = 1000$	$N = 100000$
nonspare	8Mb	80Gb
sparse	60Kb	6Mb

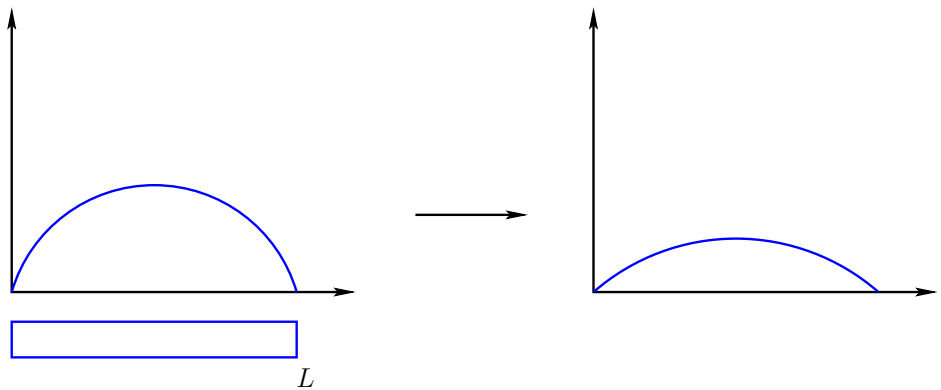
- This is important to keep in mind when applying difference methods - the resulting operators are almost exclusively sparse.

1D Heat equation

- We now consider a parabolic equation, the heat equation

$$\begin{aligned} u_t &= u_{xx}, \quad \text{i } \Omega = (0,1) \\ u(t_0) &= u_0(x) \\ u(0,t) &= g_0(t) \\ u(1,t) &= g_1(t) \end{aligned}$$

  This is a boundary-initial value problem.
- The equation smooth the initial data



- You can think of the problem as heat conduction in a 1D rod with a given temperature on the boundaries (end points).

1D heat equation - solution strategy

- We *semidiscretize* the equation. This means that the equation is discrete in some variables and continous in one or several others. Here we discretize in space and keep the equation continous in time.
- This means that we have to introduce a spatial grid - since this is in one space dimension this is simply cutting up the axis into intervals. We use the symbol h to denote spatial step size.

1D heat equation

- As usual we introduce a vector of unknowns, and we find that our semi-discrete system of equations can be stated on the form

$$\begin{aligned}
\frac{\mathrm{d}\underline{u}}{\mathrm{d}t} &= \frac{1}{h^2}\underline{A}\,\underline{u} \\
u_i(0) &= u_0(x_i) \;\forall\; i = 1, \cdots, N-1 \\
u_0(t) &= g_0(t) \\
u_N(t) &= g_1(t)
\end{aligned}$$

- This is exactly the format we had for the systems of ordinary differential equations we studied earlier - we can apply methods for solving systems of ODEs.

1D heat equation - Euler's method

- We denote the temporal step size by k.
- Euler's method can be written as

$$\underline{u}^{n+1} = \underline{u}^n + k\underline{f}\left(t^n, \underline{u}^n\right) = \underline{u}^n + \frac{k}{h^2}\underline{A}\,\underline{u}^n \Rightarrow$$
$$\frac{\underline{u}^{n+1} - \underline{u}^n}{k} = \frac{1}{h^2}\underline{A}\,\underline{u}^n.$$

- We introduce

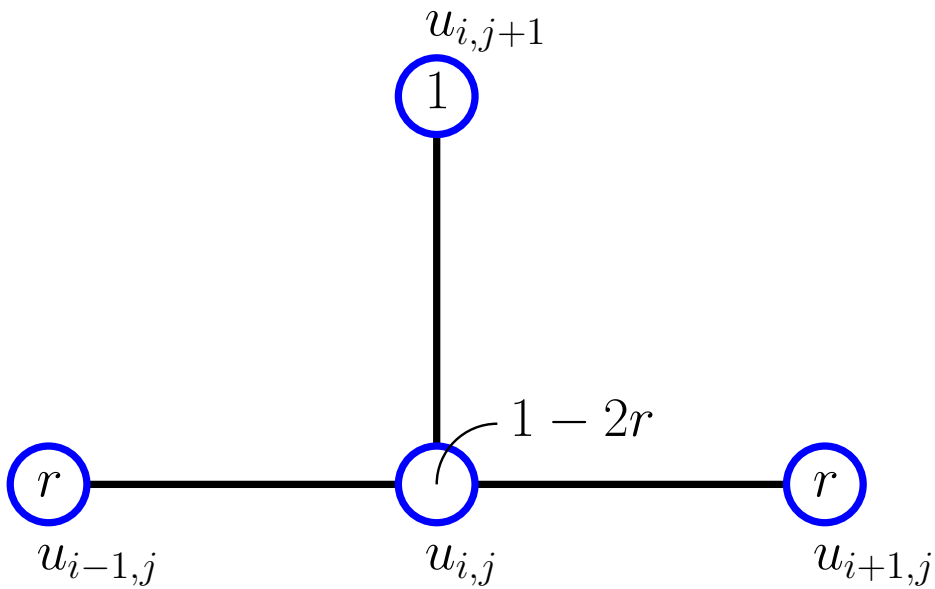
$$r = \frac{k}{h^2}$$

which let us state the method on component form as

$$u_i^{n+1} = (1-2r)u_i^n + r\left(u_{i-1}^n + u_{i+1}^n\right).$$

1D heat equation - Euler's method

- The stencil for this method is



- Euler's method for the heat equation is thus composed of
  - central difference in space
  - Euler's method in time.

1D heat equation - Euler's method - stability

- Do you recall the stability analysis we performed for Euler's method? We obtained the stability condition

$$|1 + \lambda k| < 1.$$

- Using diagonalization we showed that this translates to a condition on the eigenvalues for systems of equations. We considered systems on the form

$$\frac{\mathrm{d}\underline{u}}{\mathrm{d}t} = \underline{A}\,\underline{u}$$

which is exactly what we have here.

- It can be shown that for our matrix $\underline{A}$ we have that

$$\lambda_{\max}(\underline{A}) \sim \mathcal{O}(1)$$

which means that the total operator has

$$\lambda_{\max} \sim \frac{1}{h^2}.$$

1D heat equation - Euler's method - stability

- This can be used to show that for this particular equation we get the stability condition

$$r \leq \frac{1}{2}.$$

- This is a rather strict condition - if we half the step size in space we have to reduce the temporal step size by a factor of 4 to keep the solution stable!
- This turns out to be a catastrophy when applying this method in real life - consider the case where the materials involved are very slow conducting.
- The reason for this condition is the explicit temporal integration. We know what the cure is: Apply an implicit temporal integration instead.

1D heat equation - Crank Nicolson's method

- We here replace the temporal integration with an implicit method - namely the trapezoidal rule:

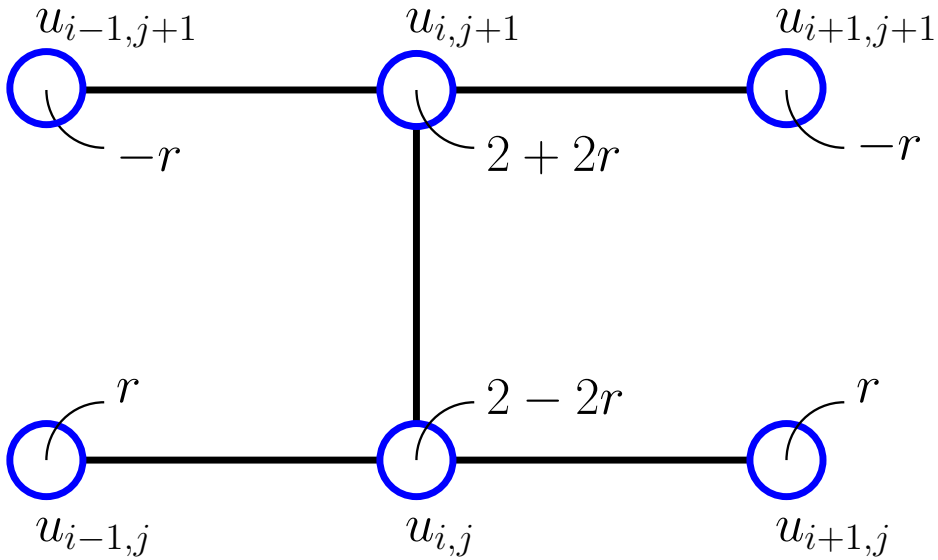
$$\begin{aligned}\underline{u}^{n+1} &= \underline{u}^{n} + \frac{k}{2}\left(\underline{f}\left(t^{n}, \underline{u}^{n}\right) + \underline{f}\left(t^{n+1}, \underline{u}^{n+1}\right)\right) \\ &= \underline{u}^{n} + \frac{k}{2h^2}\underline{A}\left(\underline{u}^{n} + \underline{u}^{n+1}\right) \Rightarrow \\ \left(\underline{I} - \frac{r}{2}\underline{A}\right)\underline{u}^{n+1} &= \left(\underline{I} + \frac{r}{2}\underline{A}\right)\underline{u}^{n}\end{aligned}$$

- On component form this can be stated as

$$\begin{aligned}&(2+2r)\,u_i^{n+1} - r\left(u_{i+1}^{n+1} + u_{i-1}^{n+1}\right) \\ &= (2+2r)\,u_i^{n} - r\left(u_{i+1}^{n} + u_{i-1}^{n}\right)\end{aligned}$$

1D heat equation - Crank Nicolson's method

- This method has the stencil



- This means that Crank-Nicolson's method for the heat equation consists of
  - central difference in space
  - the trapezoidal rule in time.

1D heat equation - Crank Nicolson's method - linear system

- We now assume that $r = 1$ to simplify the equations somewhat.
- We introduce a grid with 5 spatial grid points of which 3 are unknowns.
- We then state the equations in the three unknown grid points:

$$\begin{aligned} i = 1: \quad & 4u_1^{n+1} - u_2^{n+1} - u_0^{n+1} = u_2^n + u_0^n \\ i = 2: \quad & 4u_2^{n+1} - u_3^{n+1} - u_1^{n+1} = u_3^n + u_1^n \\ i = 3: \quad & 4u_3^{n+1} - u_4^{n+1} - u_2^{n+1} = u_4^n + u_2^n \end{aligned}$$

- Due to the given boundary values we have that

$$\begin{aligned} u_0^n &= g_0(kn) \\ u_4^n &= g_1(kn). \end{aligned}$$

1D heat equation - Crank Nicolson's method - linear system

- Hence we obtain a linear system on the form

$$\begin{bmatrix} 4 & -1 & \\ -1 & 4 & -1 \\ & -1 & 4 \end{bmatrix} \begin{bmatrix} u_1^{n+1} \\ u_2^{n+1} \\ u_3^{n+1} \end{bmatrix} = \begin{bmatrix} u_2^n + g_0(nk) \\ u_3^n + u_1^n \\ g_1(nk) + u_2^n \end{bmatrix}$$

- For an arbitrary N this yields a tridiagonal matrix

$$\begin{bmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 4 \end{bmatrix}$$

1D heat equation - Crank Nicolson's method - linear system

- These are known as *Toeplitz*-matrices.
- The fastest method to solve these systems of equations are a $\underline{L}\,\underline{U}$-factorization where we take the tridiagonal structure into consideration.
- Final comment: I again stress the fact that since we have employed an implicit temporal integration method we have *no* stability condition on the value of r, which means that we can choose step sizes based solely on accuracy considerations.