Assignment 2: The Poisson Problem

Slides by: Stefan L. Glimberg

Technical University of Denmark

02614 High-Performance Computing
January, ~~2013~~ 2024

Modified for 3D by: Hans Henrik B. Sørensen

Deadline: Friday, January 12, 2024 - at midnight!

DTU

Assignment 2

Problem: *Solve the Poisson problem; an elliptic partial differential equation in three space dimensions* x, y, *and* z

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} = -f(x, y, z), \qquad (x, y, z) \in \Omega, \tag{1}$$

where $u(x, y, z)$ is the unknown function, $f(x, y, z)$ is a given source term, and Ω defines the domain.

Partial differential equations play important roles in many fields of science and engineering. The Poisson equation describe e.g. the steady state heat distribution in a media with constant heat capacity.

Agenda

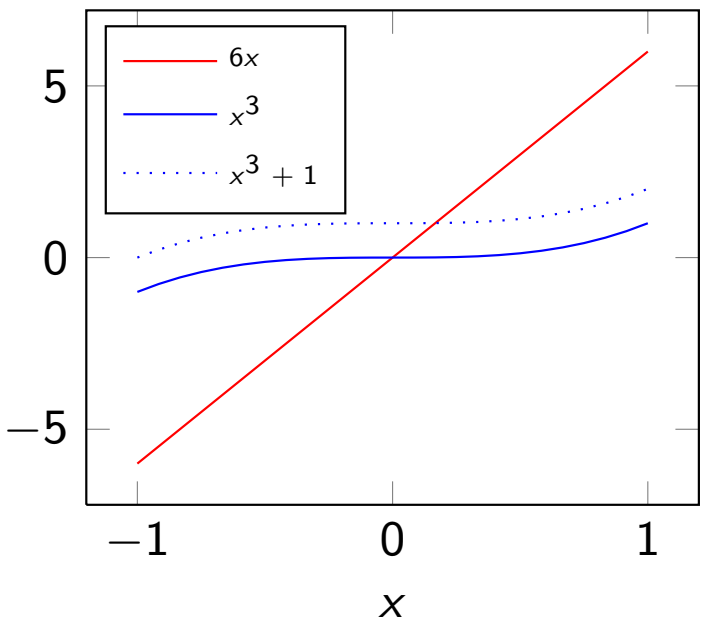
1. The Poisson problem
2. Numerical discretization methods
3. Jacobi iterations (what you will do)

1D Poisson Problem

Consider the simple one dimensional case $\frac{\partial^2 u}{\partial x^2} = -f$ and assume $-f = 6x$ and $\Omega = \{x : |x| \leq 1\}$,

$$\frac{\partial^2 u}{\partial x^2} = 6x, \qquad \Omega = \{x : |x| \leq 1\}. \tag{2}$$

Then $u = x^3$ satisfies (2), but so does $u = 1 + x^3$. Therefore conditions are required on the boundary $\partial\Omega$ to uniquely identify the solution. We consider only *Dirichlet boundary conditions*, where the exact solution on the boundary is known, e.g. if $u(-1) = -1$ and $u(1) = 1$, then $u = x^3$ is the only solution.



2D Poisson Problem

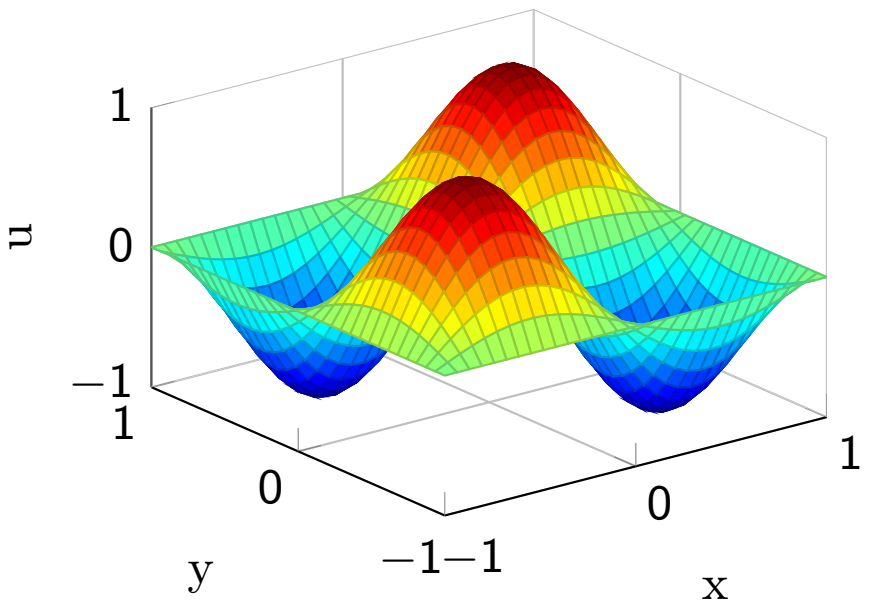
Consider the two dimensional case $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = -f$, where

$$f = 2\pi^2 \sin(\pi x) \sin(\pi y), \quad \Omega = \{(x, y) : |x| \leq 1, |y| \leq 1\}, \tag{3}$$

with Dirichlet boundary conditions

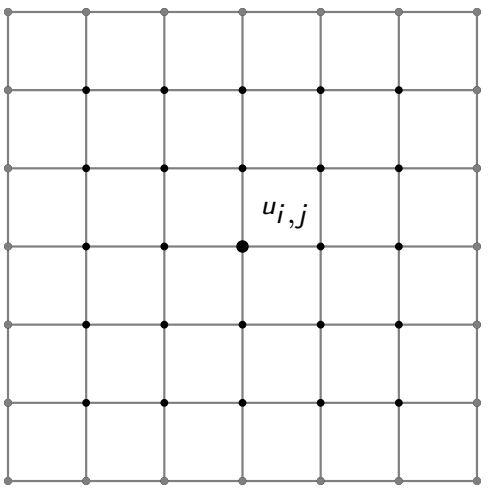
$$u(x, y) = 0, \quad (x, y) \in \partial\Omega. \tag{4}$$

Then $u = \sin(\pi x) \sin(\pi y)$ is the solution.



Numerical discretization

It is often not possible to determine u analytically, so we need to make a discrete representation of Ω and numerically approximate the derivatives $\frac{\partial^2 u}{\partial x^2}$ and $\frac{\partial^2 u}{\partial y^2}$.
Consider a regular equidistant grid with N interior grid points in each direction, such that $u_{i,j} = u(i\Delta, j\Delta)$, where $i, j = 1, 2, \ldots, N$ and Δ is the grid spacing.



$N = 5$

Finite difference approximation

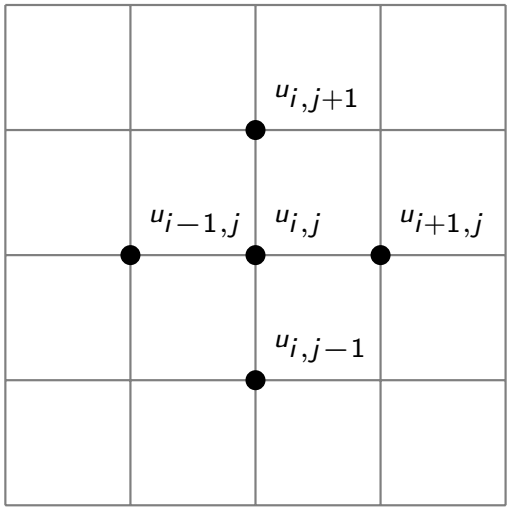
Using the finite difference method we can approximate second order derivatives at any point (i, j) using adjacent grid points

$$\frac{\partial^2 u_{i,j}}{\partial x^2} \approx \frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{\Delta^2} \tag{5}$$

$$\frac{\partial^2 u_{i,j}}{\partial y^2} \approx \frac{u_{i,j-1} - 2u_{i,j} + u_{i,j+1}}{\Delta^2}. \tag{6}$$

Inserting these approximations back into the Poisson equation we get for all interior points

$$\frac{-4u_{i,j} + u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1}}{\Delta^2} = -f_{i,j}. \tag{7}$$



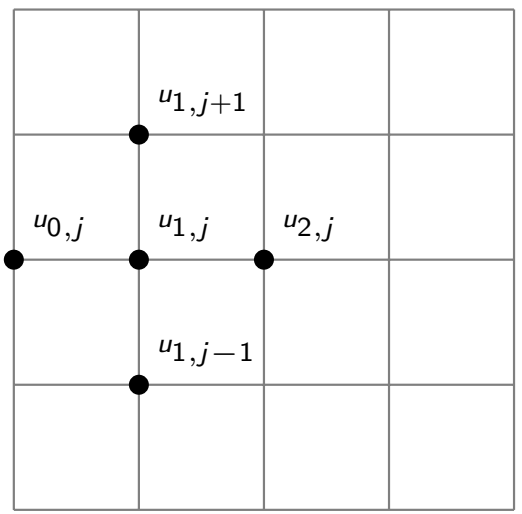
DTU

Incorporating boundary conditions

For internal grid points next to the boundary, the boundary point values are known and can be incorporated into the equations. For example, on the left wall $i = 1$ and we get

$$\frac{-4u_{1,j} + \overbrace{u_{0,j}}^{\text{known}} + u_{2,j} + u_{1,j-1} + u_{1,j+1}}{\Delta^2} = -f_{1,j}, \tag{8}$$

then $\frac{u_{0,j}}{\Delta^2}$ can be moved to the right hand side.



System of linear equations

If all N^2 unknowns are arranged into a vector $\mathbf{u}$, then the discrete Poisson equation can be written as a system of linear equations

$$\mathcal{A}\mathbf{u} = \mathbf{b}, \qquad \mathcal{A} \in \mathbb{R}^{N^2 \times N^2}, \quad \mathbf{u} \in \mathbb{R}^{N^2}, \tag{9}$$

where $\mathbf{b}$ holds f and the boundary information.

If for example $N = 3$ then we get

$$\frac{1}{\Delta^2}\begin{bmatrix} -4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -4 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & -4 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & -4 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & -4 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & -4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & -4 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -4 \end{bmatrix} \begin{bmatrix} u_{1,1} \\ u_{2,1} \\ u_{3,1} \\ u_{1,2} \\ u_{2,2} \\ u_{3,2} \\ u_{1,3} \\ u_{2,3} \\ u_{3,3} \end{bmatrix} = \begin{bmatrix} b_{1,1} \\ b_{2,1} \\ b_{3,1} \\ b_{1,2} \\ b_{2,2} \\ b_{3,2} \\ b_{1,3} \\ b_{2,3} \\ b_{3,3} \end{bmatrix} \tag{10}$$

DTU

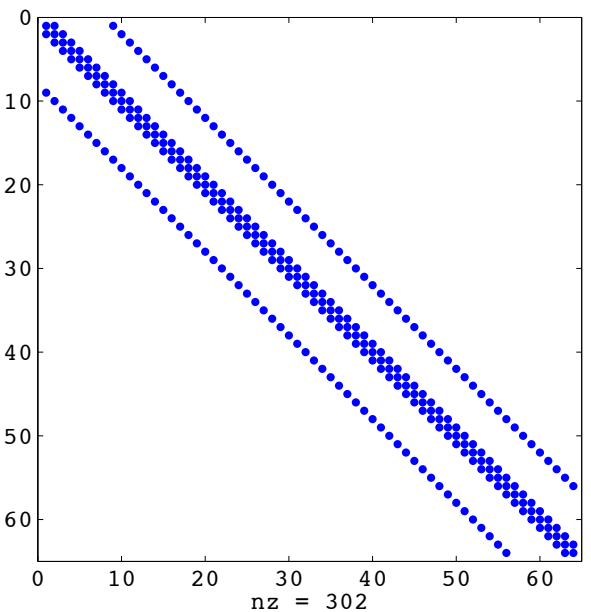
System of linear equations

If all N^2 unknowns are arranged into a vector $\mathbf{u}$, then the discrete Poisson equation can be written as a system of linear equations

$$\mathcal{A}\mathbf{u} = \mathbf{b}, \qquad \mathcal{A} \in \mathbb{R}^{N^2 \times N^2}, \quad \mathbf{u} \in \mathbb{R}^{N^2}, \tag{9}$$

where $\mathbf{b}$ holds f and the boundary information.

Or for $N = 8$



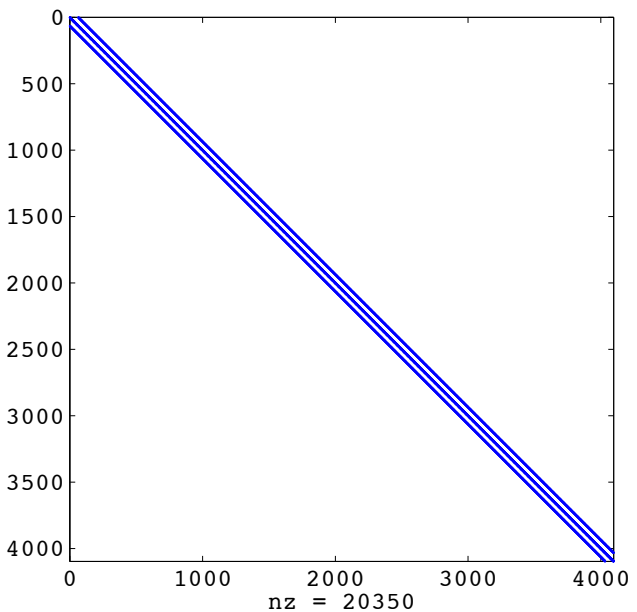
System of linear equations

If all N^2 unknowns are arranged into a vector $\mathbf{u}$, then the discrete Poisson equation can be written as a system of linear equations

$$\mathcal{A}\mathbf{u} = \mathbf{b}, \qquad \mathcal{A} \in \mathbb{R}^{N^2 \times N^2}, \quad \mathbf{u} \in \mathbb{R}^{N^2}, \tag{9}$$

where $\mathbf{b}$ holds f and the boundary information.

Or for $N = 64$



So, it all comes down to efficiently solving

$$\mathcal{A}\mathbf{u} = \mathbf{b}. \tag{10}$$

Using a *direct method* such as gaussian elimination to solve (10) can be expensive; it has arithmetic complexity of $\mathcal{O}(m^3)$, where m is the number of equations. In our case $m = N^2$ and we get $\mathcal{O}(N^6)$. Also, it requires $\mathcal{O}(m^2)$ storage.

Instead we should take advantage of the sparse and known structure of $\mathcal{A}$ and use an iterative method. Then matrix $\mathcal{A}$ never needs be formed explicitly.

DTU

Jacobi iterations

The basic ide of on *iterative method* is to choose an *initial guess* and *continuously improve* it until a certain tolerance is achieved.

You will be implementing the method known as **Jacobi iterations**:

1. Start with initial guess $u^{(0)}$
2. Outer loop: for $k = 0, 1, \ldots$
3. Inner loop: for all interior grid point update

$$u_{i,j}^{(k+1)} = \frac{1}{4}(u_{i-1,j}^{(k)} + u_{i+1,j}^{(k)} + u_{i,j-1}^{(k)} + u_{i,j+1}^{(k)} + \Delta^2 f_{i,j})$$

Notice that new values are always computed based on old data, so we need to arrays to hold new and old data.

Jacobi uses a constant number of operations per point, thus the total number of arithmetic operations per iteration scale linearly with the problem size, $\mathcal{O}(m)$, where $m = N^2$. It takes $O(N^2 \log(N))$ iterations to converge.

Gauss-Seidel iterations

An alternative to the Jacobi method is to use Gauss-Seidel iterations, where data is overwritten immediately

1. Start with initial guess $u^{(0)}$
2. Outer loop: for $k = 0, 1, \ldots$
3. Inner loop: for all interior grid point update

$$u_{i,j} = \frac{1}{4}(u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} + \Delta^2 f_{i,j})$$

How data is reused depends on how the grid points are traversed. In lexicographical order it would be equal to

$$u_{i,j}^{(k+1)} = \frac{1}{4}(u_{i-1,j}^{(k+1)} + u_{i+1,j}^{(k)} + u_{i,j-1}^{(k+1)} + u_{i,j+1}^{(k)} + \Delta^2 f_{i,j})$$

Gauss-Seidel generally converges about twice as fast as Jacobi.

When to stop the iterations

There are several ways to determine when to stop, the simplest one being to use a finite number of iterations. However, this will almost certainly lead to either too many or too few iterations compared to what is needed.

Alternatively we can compute the change from one iteration to the next, and stop when the difference is sufficiently(?) small.

1. $k = 0$
2. $d = \infty$
3. while $d > threshold$ && $k < k_{max}$
4. $\quad u_{old} = u$
5. $\quad u =$ update function
6. $\quad d = ||u - u_{old}||_P$ (where P is some norm)
7. $\quad k = k + 1$
8. end

3D Poisson Problem

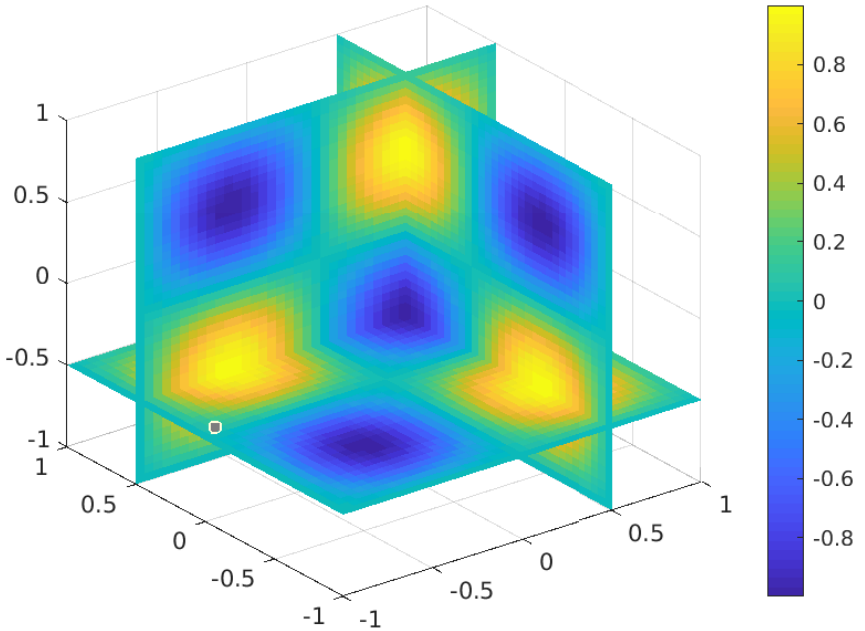
Consider the 3-dimensional case $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} = -f$, using

$$f = 3\pi^2 \sin(\pi x)\sin(\pi y)\sin(\pi z), \quad \Omega = \{(x, y, z) : |x| \leq 1, |y| \leq 1, |z| \leq 1\},$$

with Dirichlet boundary conditions

$$u(x, y, z) = 0, \quad (x, y, z) \in \delta\Omega \tag{12}$$

Then $u = \sin(\pi x)\sin(\pi y)\sin(\pi z)$ is the solution.



3D Jacobi iterations

The three dimensional Jacobi method can be obtained as the straight forward *extension* of the two dimensional method.

You will be implementing the method known as **Jacobi iterations**:

- Start with initial guess $u^{(0)}$
- Outer loop: for $n = 0, 1, \ldots$
- Inner loop: for all interior grid point update

$$u_{i,j,k}^{(n+1)} = \frac{1}{6}\left(u_{i-1,j,k}^{(n)} + u_{i+1,j,k}^{(n)} + u_{i,j-1,k}^{(n)} + u_{i,j+1,k}^{(n)} + u_{i,j,k-1}^{(n)} + u_{i,j,k+1}^{(n)} + \Delta^2 f_{i,j,k}\right)$$

And likewise for the 3D Gauss-Seidel iterations:

$$u_{i,j,k}^{(n+1)} = \frac{1}{6}\left(u_{i-1,j,k}^{(n+1)} + u_{i+1,j,k}^{(n)} + u_{i,j-1,k}^{(n+1)} + u_{i,j+1,k}^{(n)} + u_{i,j,k-1}^{(n+1)} + u_{i,j,k+1}^{(n)} + \Delta^2 f_{i,j,k}\right)$$

Such that $u_{i,j,k} = u(i\Delta, j\Delta, k\Delta)$, where $i, j, k = 1, 2, \ldots, N$ and Δ is the grid spacing.

The Assignment

In this assignment we consider the heat distribution in a small cubic room (ignoring convection and other effects) with a radiator (with a radiation $= 200^\circ$ C/m^2) placed somewhat near the cold wall, and with the temperature kept fixed at the walls: 20° C at five walls and 0° C degrees at the sixth wall. Hence, we can take Ω as the cube

$$\Omega = \{(x, y, z) : |x| \leq 1 \, , |y| \leq 1 \, , |z| \leq 1\}$$

and we have the Dirichlet boundary conditions

$$\begin{aligned} u(x, 1, z) = 20, u(x, -1, z) = 0, \qquad & |x| \leq 1 \, , |z| \leq 1 \\ u(1, y, z) = u(-1, y, z) = 20, \qquad & |y| \leq 1, |z| \leq 1 \\ u(x, y, -1) = u(x, y, 1) = 20, \qquad & |x| \leq 1, |y| \leq 1. \end{aligned}$$

Finally, the radiator is represented by the function

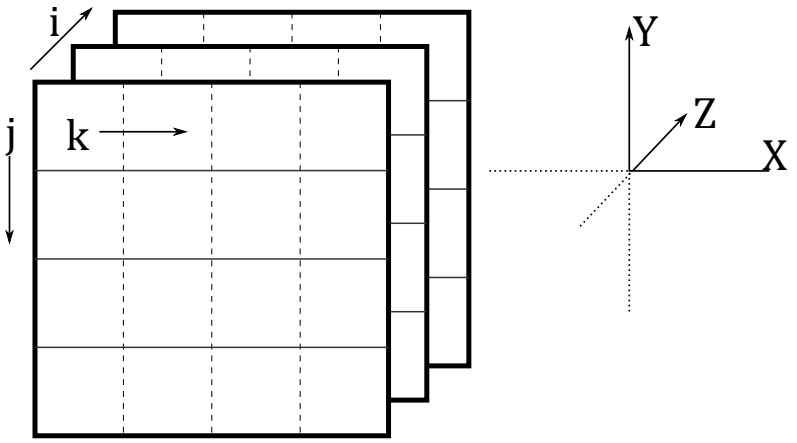
$$f(x, y, z) = \begin{cases} 200 \, , -1 \leq x \leq -3/8, -1 \leq y \leq -1/2, -2/3 \leq z \leq 0 \\ 0 \quad \, , \text{ elsewhere.} \end{cases}$$

DTU

Hints

- Allocate memory for u and f of size $(N+2) \times (N+2) \times (N+2)$ to account for boundary points.
- Loop through and initialize f and the boundary values of u.
- Write your Jacobi iteration method as separate functions; makes it easier to change/add new.
- Be aware of numerical and algorithmic performance.
- Flops is based on the number of floating points operations for the Jacobi update times the problem size times the number of iterations.
- Consider carefully how the array indices [i][j][k] relate to the memory locations and the physical dimensions of the volume

Figure by:
Tim O. &
Mathias L.



Good luck!