Poisson Equation

Many systems in science and engineering are described by PDEs (Partial Differential Equations). A simple example for a PDE is the *Poisson Equation*, describing e.g. the gravitational potential or the electric potential, if the mass or charge distribution is given. For simplicity, we consider here a system with two spatial dimensions and Cartesian coordinates x and y.

Background

Given a mass distribution $\rho(x, y)$, the resulting gravitational potential $U(x, y)$ is given by

$$\Delta U(x, y) \; = \; 4\pi\rho(x, y) \quad , \tag{1}$$

where Δ is the Laplace operator div $\cdot$ grad, i.e. $\partial_x^2 + \partial_y^2$ in our case.

A particularly simple way of calculating $U(x, y)$ is the *Jacobi Method*, which is defined as follows:

1. Choose any initial guess for the potential, e.g. $U(x, y) := 0$
2. Evaluate the residual: $r(x, y) := \Delta U(x, y) - 4\pi\rho(x, y)$
3. Calculate the L_2 norm of the residual: $L_2[r] := \left(\int |r(x, y)|^2 \, dx \, dy / V\right)^{1/2}$, where V is the volume of the domain
4. If the L_2 norm of the residual is small enough, we are done
5. Otherwise, add a small multiple of the residual to the potential: $U(x, y) \to U(x, y) + \alpha r(x, y)$
6. Repeat from step 2

It is important to choose a good value for $\alpha > 0$. If α is too large, the Jacobi method is unstable, and the residual will grow without bounds. If α is small enough, the residual will converge to zero.

> *Note:* This algorithm is both simple and spectacularly inefficient. We use it here only because it leads to a simple code. Do not use this algorithm to solve a real-world problem. Two much better classes of algorithms are *multi-grid methods* and *Krylov subspace methods*. There exist efficient, generic, ready-to-use libraries for these methods, such as e.g. PETSc.

We discretise the Poisson equation by employing *finite differences*. We represent the domain by two-dimensional arrays $[0, n] \times [0, n]$, where $n + 1$ is the number of grid points, and $h = 1/n$ is the spatial resolution. We discretise the Laplace operator via second-order centred finite differences:

$$\left(\partial_x^2 + \partial_y^2\right) U(x, y) \; := \; \frac{U_{i-1,j} - 2\,U_{i,j} + U_{i+1,j}}{h^2} + \frac{U_{i,j-1} - 2\,U_{i,j} + U_{i,j+1}}{h^2} \quad . \tag{2}$$