M.Sc. Simulation Sciences, Summer Semester 2021

Fast Iterative Solvers

Prof. Georg May

Work Package for Multigrid Assignment – Part 1

Summary

This is preparatory work for multigrid methods. Here you will *not* assemble sparse matrices explicitly, so there is no need to use `CSR` storage format.

Poisson Solver

Consider the Poisson equation with homogeneous boundary conditions

$$\begin{aligned} Lu := -\nabla^2 u &= f \qquad \text{in } \Omega \\ u &= 0 \qquad \text{on } \partial\Omega \end{aligned}$$

where $\Omega = (0,1) \times (0,1)$. Consider a Finite Difference discretization on a Cartesian Grid

$$\mathcal{G}_h := \{(ih, jh) : i, j = 0, \ldots, N;\ hN = 1\}. \tag{1}$$

This means finding $u_{i,j} := u(x_i, y_j)$ such that

$$-f_{i,j} = \frac{1}{h^2}\left(u_{i-1,j} - 2u_{i,j} + u_{i+1,j}\right) + \frac{1}{h^2}\left(u_{i,j-1} - 2u_{i,j} + u_{i,j+1}\right) \ , \quad i, j = 1, \ldots N-1 \ , \tag{2}$$

Note that we don't iterate over the boundary points! These should be initialized to zero, and then not be touched again. Implement a Gauss-Seidel Relaxation with lexicographical ordering (GS-LEX) to solve the problem. Lexicographical ordering means that the iteration is "line by line": For indexing as above, first $i = 1, 2 \ldots, N-1$ for $j = 1$, then $i = 1, 2, \ldots, N-1$ for $j = 2$ and so forth until $j = N-1$, which completes one iteration.

You should implement this as a function, i.e. $\mathbf{u} = GS(\mathbf{u}_0, \mathbf{f}; \nu)$, where $\mathbf{u}$ and $\mathbf{f}$ are two-dimensional arrays for the solution and right-hand sides, respectively. $\mathbf{u}_0 = 0$ is the initial guess, and ν is the number of iterations.

- Test this for the right-hand side $f(x,y) = 8\pi^2 \sin(2\pi x)\sin(2\pi y)$
- For this choice the solution is $u(x,y) = \sin(2\pi x)\sin(2\pi y)$.
- Choose ν large enough that $||\mathbf{u}_\nu - \mathbf{u}_{\nu-1}||_\infty < 10^{-10}$,
- Measure the *converged* maximum error, i.e. $\max_{i,j} |u_{i,j} - u(x_i, y_j)|$ for $N_x = N_y = N$, and $N = 10$, as well as $N = 100$
- Note: For this Cartesian grid, you should not assemble *any* matrix to implement the Gauss-Seidel iteration!