Series 3

Computational Methods for
Engineering Applications
Last edited: October 26, 2020
Due date: November 8 at 23:59

Template codes are available on the course's webpage at `https://moodle-app2.let.ethz.ch/course/view.php?id=13412`.

Exercise 1 Finite Differences for the Poisson equation in 1D

We consider the 1D Poisson equation with homogeneous Dirichlet boundary conditions:

$$\begin{aligned} -u''(x) &= f(x), \quad \forall x \in \Omega = (0,1) \\ u(0) &= u(1) = 0. \end{aligned} \tag{1}$$

Where $f : \mathbb{R} \to \mathbb{R}$ is a continuous function. We want to discretize eq. (1) using the Finite Differences method, namely the centered finite differences. To this aim, we subdivide the interval $[0,1]$ in $N+1$ subintervals using equispaced grid points

$$\{x_0 = 0, x_1, \ldots, x_N, x_{N+1} = 1\}.$$

The discretized problem can be written as a linear system

$$\mathbf{A}\mathbf{u} = \mathbf{F}, \tag{2}$$

where $\mathbf{A}$ is a $N \times N$ matrix, $\mathbf{F}$ a $N \times 1$ vector and $\mathbf{u}$ the $N \times 1$ vector containing the unknowns $u_j \approx u(x_j)$, $j = 1, \ldots, N$, the approximate values of the function u at the grid points.
Let us denote by $h = |x_1 - x_0|$ the meshsize.

1a)

Using central finite differences, write the matrix $\mathbf{A}$ and the right hand side vector $\mathbf{F}$. For the right hand side, write it in terms of a generic force term $f(x)$ in (1).

1b)

In the template file `finite_difference.cpp`, implement the function

```
void createPoissonMatrix(SparseMatrix& A, int N),
```

where `typedef Eigen::SparseMatrix<double> SparseMatrix`. This function computes the matrix $\mathbf{A}$ for (2). Here the input parameter `N` denotes the number of *interior* grid points. Assume that the size of the input matrix `A` has not been initialized.

1c)

In the template file `finite_difference.cpp`, implement the function

```
void createRHS(Vector& rhs, FunctionPointer f, int N, double dx),
```

where `typedef double(*FunctionPointer)(double)` and `typedef Eigen::VectorXd Vector`. This function computes the right hand side $\mathbf{F}$ for (2). The input parameter `f` is the function pointer for the right-handside $f(x)$, `N` is again the number of interior grid points, and `dx` is the length of a cell. Assume that the size of the input vector `rhs` has not been initialized.

1d)

In the template file `finite_difference.cpp`, implement the function

```
void poissonSolve(int order, Vector& u, FunctionPointer f, int N),
```

to solve the Poisson problem (1).
The input parameters `f` and `N` are as in subproblem **1c)**. Parameter `order` should be 2 for now. The vector `u` is assumed to have not been initialized in size, and at the end of the routine it has to correspond to the array $\{u_h(x_j)\}_{j=1}^{N}$ containing the approximate values of the solution u at the interior grid points $\{x_j\}_{j=1}^{N}$.

Hint: Use the routines from subproblems **1b)** and **1c)**.

1e)

Run the routine `poissonSolve` for $f(x) = \sin(2\pi x)$ and $N = 50$ and plot the solution.

1f)

We saw in the lecture that the centered finite difference scheme is stable and consistent, and thus it converges to the exact solution u to (1) when the mesh is refined.
Here we are going to study the convergence of our scheme.

In the template file `finite_difference.cpp`, implement the function

```
void poissonConvergence(int order, std::vector<double>& errors, std::vector<double>&
↪ resolutions, FunctionPointer f)
```

to perform the convergence study. The input argument `f` is a function pointer to the right hand side $f(x)$. The vectors `resolutions` and `errors`, assumed to be passed in input with uninitialized size, must contain, in output, the array $[N_1, \ldots, N_7]$ of numbers of degrees of freedom and the array $[e_1, \ldots, e_7]$ of computed errors, respectively. `order` should be passed to `PoissonSolve`.

As error between the discrete solution $\boldsymbol{u}^h$ and the exact solution u, we consider the maximum norm error

$$\|u - \boldsymbol{u}^{h_i}\|_\infty = \max_{1 \leq j \leq N_i} |u(x_j) - u_j^{h_i}|, \qquad i = 1, \ldots, 7.$$

The standard steps for a convergence study then:

1. compute the *exact solution* u to (1);
2. start from a meshsize $h_1 = \frac{1}{2^4}$, corresponding to $N_1 = 15$ interior grid points;
3. compute the discrete solution $\boldsymbol{u}^{h_1}$ to (1);
4. compute the error $e_1 = \|u - \boldsymbol{u}^{h_1}\|_\infty$
5. refine the grid, considering $h_2 = \frac{h_1}{2} = \frac{1}{2^5}$ (corresponding to $N_2 = 31$) and repeat the algorithm from step 3;
6. repeat the previous step till $h_7 = \frac{1}{2^{10}}$.

1g)

Run the routine `poissonConvergence` for $f(x) = \sin(2\pi x)$.
Make a double logarithmic plot of the errors $e_1, \ldots, e_7$ versus the resolutions $N_1, \ldots, N_7$. What do you observe? Which is the order of convergence?

Exercise 2 Finite Differences for Poisson Equation in 2D

In this problem we consider the Finite Differences discretization of the Poisson problem on the unit square:

$$
\begin{aligned}
-\Delta u &= f \quad \text{in } \Omega := (0,1)^2, \\
u &= 0 \quad \text{on } \partial\Omega,
\end{aligned}
\tag{3}
$$

for a bounded and continuous function $f \in \mathcal{C}^0(\overline{\Omega})$.

We consider a regular tensor product grid with meshwidth $h := (N+1)^{-1}$ and we assume a lexicographic numbering of the interior vertices of the mesh as depicted in Fig.1.

[b]

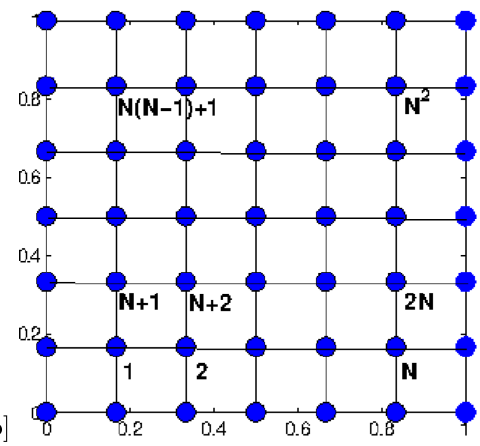

Figure 1: Lexicographic numbering of vertices of the equidistant tensor product mesh.

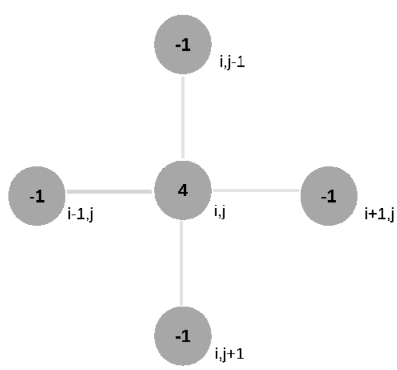

Figure 2: 5-point stencil used in this problem.

We consider the 5-point stencil finite difference scheme for the operator $-\Delta$ described by the 5-points stencil shown in Fig. 2.

2a)

Write the system

$$\mathbf{A}\mathbf{u} = \mathbf{F} \tag{4}$$

corresponding to the discretization of (3) using the stencil in Fig. 2, specifying the matrix $\mathbf{A}$ and the vectors $\mathbf{F}$ and $\mathbf{u}$.

2b)

In the template file `finite_difference.cpp`, implement the function

```
void createPoissonMatrix2D(SparseMatrix& A, int N),
```

to construct the matrix $\mathbf{A}$ in (4), where N denotes the number of interior grid points along one dimension, with `typedef Eigen::SparseMatrix<double> SparseMatrix`. Assume the matrix `A` to have an uninitialized size at the beginning.

2c)

In the template file `finite_difference.cpp`, implement the function

```
void createRHS(Vector& rhs, FunctionPointer f, int N, double dx),
```

to build the vector $\mathbf{F}$ in (4), with `typedef Eigen::VectorXd Vector` and `typedef double(*FunctionPointer)(double, double)`. The argument `f` is a function pointer to the function f in (3), `N` is the number of interior grid points and `dx` is cell width. Again, assume that the vector `rhs` has uninitialized size when passed in input.

2d)

In the template file `finite_difference.cpp`, implement the function

```
void poissonSolve(Vector& u, FunctionPointer f, int N),
```

to solve the system (4), with `u` the vector containing the values of the approximate solution at all the grid points, *including those on the boundary*, and the other arguments as in the previous subproblems.

2e)

Using attached files `plot_fd2d.py/m` plot the discrete solution that you get from subproblem **2d)** for $f(x, y) = 8\pi^2 \sin(2\pi x) \sin(2\pi y)$ and $N = 50$, and compare it to the exact solution $u(x, y) = \sin(2\pi x) \sin(2\pi y)$.