Computer Assignment EE4550: Block 2

Finite Element Solver of a Poisson Equation in One Dimension

The objective of this assignment is to guide the student to the development of a finite element solver from scratch. The assignment consists of a compulsory and an elective part. The former in turn consists of both pen-and-paper and implementation exercises.

In this assignment we aim at solving the Poisson equation on the domain $\Omega = (0, 1)$. We in particular set out to find the function $u(x)$ that is solution of the equation

$$-\frac{d^2u}{dx^2} = g(x) \text{ for } 0 < x < 1 \tag{1}$$

where the source function $g(x)$ is defined as

$$g(x) = 2\sin(\pi x) + 4\pi(x-1)\cos(\pi x) - \pi^2(x-1)^2\sin(\pi x) \tag{2}$$

and where $u(x)$ satisfies homogeneous Dirichlet and homogeneous Neumann boundary conditions at the left and the right end point of Ω, respectively. This means that

$$u(x=0) = 0 \quad \text{and} \quad \frac{du}{dx}(x=1) = 0. \tag{3}$$

Compulsory Theoretical Part

Assignment 1 Verify that the function $u(x) = -(x-1)^2\sin(\pi x)$ is the exact solution of the above problem. Do so by verifying that the second derivative of $u(x)$ equals $-g(x)$ and that $u(x)$ verifies the boundary conditions at both end points. Plot the function $u(x)$ for $0 \leq x \leq 1$ for future reference.

Assignment 2 Cast the above Poisson equation supplied with the boundary conditions at the left and right end point in its weak or variational form. Pay in particular attention to the order of derivatives used and to the choice of the vector space of test functions to resolve the boundary conditions. Please refer to the lecture notes for completing this assignment.

Assume that Ω is discretized by a mesh consisting of $n+1$ vertices (or nodes) x_i where i runs from 1 to $n+1$. This enumeration includes the end points of Ω, that is, $x_1 = 0$ and $x_{n+1} = 1$. Assume furthermore that the mesh on Ω consists of n elements $e_i = [x_i, x_{i+1}]$ where $i = 1, \ldots, n$ such that $\Omega = \cup_{i=1}^{n} e_i$. The points x_i and x_{i+1} are the end points of e_i. Assume furthermore that on each element e_i the solution to the previously introduced Poisson problem is approximated by linear (first order) Lagrangian shape functions. On each element e_i the weak form is discretized. This requires the computation of an element 2×2 matrix S_{e_i} and the element 2×1 vector f_{e_i} on each element e_i. This matrix and vector are a representation of the second derivative operator $-d^2/dx^2$ and the source function $g(x)$ locally on the element e_i.

Assignment 3 Give the expression for the element 2×2 matrix S_{e_i} and the element 2×1 vector f_{e_i} on the element e_i. Use the trapezoidal rule to approximate the integrals on e_i that appear in these expressions. Please refer to the lecture notes for completing this assignment.

The contribution S_{e_i} and f_{e_i} on element e_i need to be assembled to the global matrix S and f. This assembly requires the mapping from the local enumeration of the nodes on element e_i to the global enumeration of nodes on the mesh of Ω. The local and global enumeration run from 1 to 2 and from 1 to $n+1$, respectively. This mapping gives a unique global index to each left and right node of the element e_i and is defined by the connectivity matrix or topology of the mesh.

Assignment 4 Explain why the global matrix S and the global vector f are of size $(n+1) \times (n+1)$ and $(n+1) \times 1$, respectively.

Having the matrix S and the vector f the discrete finite element solution $u^h(x)$ can be computed by a linear system solve.

Compulsory Implementation Part

This part guides the student to the development of a finite element code for solving the above Poisson equation by guiding in the construction of a mesh, the assembly of the linear system on the mesh and solution of the linear system.

Assignment 5 Write a Matlab (or Python) routine, called *GenerateMesh.m* (or GenerateMesh.py), that returns as output a vector of size $n+1$ of equidistant grid points x_i on Ω.

Assignment 6 Write the routines *SourceFct.m* and *ExactSolu.m* that given the coordinate x_i return as output the source function $g(x_i)$ and exact solution $u(x_i)$, respectively.

Assignment 7 Write a routine, called *GenerateTopology.m*, that generates an $n \times 2$ matrix called *elmat* such that the ith row of *elmat* contains the indices of the left and right node of the ith element in the global enumeration on mesh on Ω, that is

$$\begin{aligned} elmat(i,1) &= i \quad \text{for } i = 1, \ldots, n \\ elmat(i,2) &= i+1 \end{aligned} \tag{4}$$

Assignment 8 Write a routine, called *GenerateElementMatrix.m*, that given the coordinates x_i and x_{i+1} of the end points of element e_i generates the 2×2 element matrix S_{e_i} such that

$$S_{e_i} = \frac{1}{x_{i+1} - x_i} \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} . \tag{5}$$

Assignment 9 Write a routine, called *GenerateElementVector.m*, that given the coordinates x_i and x_{i+1} of the end point of element e_i generates the 2×1 element vector f_{e_i} such that

$$f_{e_i} = \frac{x_{i+1} - x_i}{2} \begin{pmatrix} g(x_i) \\ g(x_{i+1}) \end{pmatrix} . \tag{6}$$

Assignment 10 Write a routine, called *AssembleMatrix.m*, that assembles the element matrices S_{e_i} on each element into the global matrix $(n+1) \times (n+1)$ matrix S. To so by first initializing S to be an empty $(n+1) \times (n+1)$ matrix and subsequently performing a loop over the elements. In this loop the element matrices are generated and added to the global matrix. In this addition the connectivity of the mesh defined by the matrix *elmat* needs to be taken into account. Write therefore a triple for-loop in which

- the outermost loop indexed by $i = 1, \ldots, n$ traverses the elements;
- on each element e_i the element matrix S_{e_i} on the element e_i is computed;
- the innermost two loops indexed by $j, k = 1, 2$ traverse the nodes on ith element;
- the following statement is placed in the innermost loop

$$S(elmat(i,j), elmat(i,k)) = S(elmat(i,j), elmat(i,k)) + S_{e_i}(j,k) . \tag{7}$$

Assignment 11 Write a routine, called *AssembleVector.m*, that assembles the element vectors f_{e_i} on each element into the global matrix $(n+1) \times 1$ vector f. To so by first initializing f to be an empty $(n+1) \times 1$ vector and subsequently performing a loop over the elements. In this loop the element vectors are generated and added to the global vector. In this addition the connectivity of the mesh defined by the matrix *elmat* needs to be taken into account. Write therefore a double for-loop in which

- the outermost loop indexed by $i = 1, \ldots, n$ traverses the elements;
- on each element e_i the element matrix f_{e_i} on the element e_i is computed;
- the innermost loop indexed by $j = 1, 2$ traverse the nodes on ith element;
- the following statement is placed in the innermost loop

$$f(elmat(i,j)) = f(elmat(i,j)) + f_{e_i}(j)\,. \tag{8}$$

Assignment 12 Modify the first equation of the linear system $S\,u^u = f$ in such a way that the finite element solution $u^h(x)$ satisfied the Dirichlet boundary conditions at $x = 0$. This can be accomplished by modifying the first equation of the linear system in the following way. Modify the first row of the matrix S by setting $S(1,1) = 1$ and $S(1,2) = 0$ and modify the first element of f by setting $f(1) = 0$.

Assignment 13 Run the assembly routines to get the matrix S and vector f for $n = 100$. Visualize the matrix S using the command *spy* in matlab. Can you give an interpretation of the picture you obtain?

Assignment 14 Compute the finite element solution u^h using $u^h = S\backslash f$ in Matlab. Compare this solution with the analytical solution found in the first assignment.

Elective Part

- change the analytical solution and recompute the finite element solution;
- compute the solution on a sequence of increasing finer meshes and verify how the error decreases. Use to this end a set of uniform meshes with N elements and allow N to gradually increase by setting N equal to $2, 4, 8, 16, 32, 64, \ldots$. Given a solution computed on a mesh, compute the error $u(x) - u^h(x)$ on all of the mesh points and define the error norm to be largest component of the error vector in absolute value. Plot the error norm as a function of N. Use a logarithmic scale for the error norm axis. What do you observe? Can you explain the observed behavior;
- change the Neumann boundary condition to $\frac{du}{dx}(x = 1) = \alpha$ and solve the resulting problem numerically for different values of α;
- plot the error defined as $u(x) - u^h(x)$ as a function of x over Ω. Implement a non-uniform mesh and try refine the mesh in areas in which the error is largest. Recompute the solution and verify whether the error has decreased as expected;
- change the differential equation to be solved to

$$-\frac{d}{dx}\left(c(x)\,\frac{du}{dx}\right) = g(x) \text{ for } 0 < x < 1 \tag{9}$$

  where the diffusion coefficient is a non-constant function;
- implement second order Lagrangian elements;