MATH 521

Assignment 5

February 8, 2018

Problem 1. In this assignment, we consider the linear elasticity problem

$$\begin{aligned} -c\Delta u + au &= f \quad \text{in } \Omega \\ u &= g \quad \text{on } \partial\Omega \end{aligned} \tag{1}$$

on a polygonal domain Ω. The function u can be interpreted as the elongation of a rubber membrane over the x_1x_2-plane. The boundary values g prescribe the elongation on $\partial\Omega$, e.g. by means of a wire frame construction in which the membrane has been fixed. The real number $c > 0$ is the stiffness of the rubber material, $a > 0$ is a constant proportional to its mass density and the inhomogeneity f models external forces that act on the membrane.

(a) Show that under the assumption of homogeneous boundary conditions, $g = 0$, the discretisation of (1) with linear finite elements reads

$$(cK^h + aM^h)\vec{u}^h = \vec{f}^h$$

where

$$\begin{aligned} k_{ij}^h &= \int_\Omega \nabla\phi_i^h \cdot \nabla\phi_j^h \,\mathrm{d}x \\ m_{ij}^h &= \int_\Omega \phi_i^h \phi_j^h \,\mathrm{d}x \\ f_i^h &= \int_\Omega f\phi_i^h \,\mathrm{d}x \\ \phi_i^h &= \text{hat function centred at the } i\text{-th vertex} \end{aligned}$$

for $i, j = 1, ..., N$. N is the number of effective degrees of freedom, i.e. the number of interior grid points which are not located on the boundary $\partial\Omega$. Note that since the domain is assumed to be a polygon, we can cover it exactly with a triangulation T^h such that $\Omega = \Omega^h$ (there is no mismatch on the boundary).

Solution. We will transform the strong form (1) of the problem in to the weak form in several steps, as in class.

- We begin with the strong form (1) of the problem. Here, $u \in C^2(\Omega) \cap C(\bar{\Omega})$.
- Next, as we are now looking for solutions to the weak form of the problem, we multiply by a test function $v \in H_0^1(\Omega)$ and integrate over the domain to obtain

$$-c\int_\Omega \Delta uv \,\mathrm{d}x + a\int_\Omega uv \,\mathrm{d}x = \int_\Omega fv \,\mathrm{d}x$$

.

- Now, we use Green's first identity

$$\int_\Omega (\psi\Delta\varphi + \nabla\psi \cdot \nabla\varphi) \,\mathrm{d}x = \oint_{\partial\Omega} \psi(\nabla\varphi \cdot \hat{n}) \,\mathrm{d}x$$

to integrate the first term by parts. Using the fact that both u and v are zero on the boundary, we arrive at the weak form of the problem:

$$c\int_\Omega \nabla u \cdot \nabla v \,\mathrm{d}x + a\int_\Omega uv \,\mathrm{d}x = \int_\Omega fv \,\mathrm{d}x. \tag{2}$$

- We aim to find solutions $u \in H_0^1(\Omega)$ to the weak problem (2). We project the true solution u onto the subspace $V^h \subset H_0^1(\Omega)$, where V^h is the space of functions spanned by the linear "hat" basis functions $\phi_i^h(x)$, where $i = 1, ..., N$. Note that it is indeed a true subspace due to the assumption of the domain Ω to be polygonal. The hat function $\phi_i^h(x)$ takes the value 1 on the i-th vertex, decreases linearly to 0 on all adjacent vertices, and is 0 on every other vertex.

  We expand u as a sum over the linear hat functions as

  $$u^h(x) = \sum_{j=1}^{N} \phi_j^h(x) u_j^h.$$

  The approximate solution u^h represents a finite dimensional approximation to the infinite dimensional solution u.

- It is sufficient to assert that the weak form holds against every linear basis function $v_i = \phi_i^h \in V^h$ for $i = 1, ..., N$. This leads to the system of equations

  $$\sum_{i=1}^{N} \left(c\int_\Omega \nabla\phi_i^h \cdot \nabla\phi_j^h \,\mathrm{d}x + a\int_\Omega \phi_i^h\phi_j^h \,\mathrm{d}x \right) u_j^h = \int_\Omega f\phi_i^h \,\mathrm{d}x.$$

  The solution u^h to this system of equations is optimal in the sense of Galerkin, wherein the error vector $u^h - u$ is orthogonal to all test functions $\phi_j(x)$.

- Interpreting the left-hand side sum as a matrix multiplication against a vector $\vec{u}^h = [u_1^h, ..., u_N^h]^T$, we now write

  $$\begin{aligned} k_{ij}^h &= \int_\Omega \nabla\phi_i^h \cdot \nabla\phi_j^h \,\mathrm{d}x \\ m_{ij}^h &= \int_\Omega \phi_i^h\phi_j^h \,\mathrm{d}x \\ f_i^h &= \int_\Omega f\phi_i^h \,\mathrm{d}x, \end{aligned}$$

  where k_{ij}^h and m_{ij}^h are the ij-th elements of the matrices K^h and M^h, respectively, and f_i^h is the i-th element of the vector $\vec{f}^h$. The resulting system of equations is the desired linear system

  $$(cK^h + aM^h)\vec{u}^h = \vec{f}^h. \tag{3}$$

□

(b) We can decompose the integrals that appear in the definition of the mass matrix into contributions from each triangle:

$$m_{ij}^h = \int_\Omega \phi_i^h \phi_j^h \,\mathrm{d}x = \sum_{k=1}^{n_T} \int_{T_k} \phi_i^h \phi_j^h \,\mathrm{d}x$$

n_T is the number of triangles in the triangulation T^h. Let's look at one such triangle T, the vertices of which have the indices 1, 2 and 3. Note that only the three hat functions ϕ_1^h, ϕ_2^h, and ϕ_3^h are non-zero on this triangle.

Show that the element mass matrix with entries

$$m_{ij,T}^h = \int_T \phi_i^h \phi_j^h \,\mathrm{d}x \qquad (i,j = 1,2,3)$$

is

$$M_T^h = \frac{|T|}{12} \begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix}$$

Hint: If you enjoy doing double integrals over triangles, you could use equation (2.21) from the notes. However, we don't want to mark endless calculations, so please use a different approach instead, similar to the mass matrix in 1D (Example 2.3.1, also available on Canvas under 'Pages' in the menu). Your solution should fit on the remainder of this page.

Solution. Products of linear basis functions are trivial to integrate on the triangular domain if we use quadrature rules which are exact for curves of degree ≤ 2. One such rule is the *trapezoidal rule*, in which the integrand is evaluated at the midpoints of the edges of the triangle T and summed with equal weightings of $w_i = 1/3$ for $i = 1, 2, 3$.

Since by definition the value of the linear basis functions is linear from node to node, it is clear that the value at the midpoint of the edge is either $\frac{1}{2}$ i.e. the edge connects node values of 0 and 1, or 0 i.e. the edge connects nodes each of value 0.

Now, the integration is trivial. We consider two cases:

(i) $i = j$, and so $m_{ii,T}^h = \int_T (\phi_i^h)^2 \,\mathrm{d}x = [\frac{1}{3}(\frac{1}{2})^2 + \frac{1}{3}(\frac{1}{2})^2 + \frac{1}{3}(0)^2]\,|T| = \frac{2}{12}\,|T|$

(ii) $i \neq j$, and so $m_{ij,T}^h = \int_T \phi_i^h \phi_j^h \,\mathrm{d}x = [\frac{1}{3}(\frac{1}{2})(\frac{1}{2}) + \frac{1}{3}(\frac{1}{2})(0) + \frac{1}{3}(0)(\frac{1}{2})]\,|T| = \frac{1}{12}\,|T|$

The resulting element mass matrix is therefore

$$M_T^h = \left[m_{ij,T}^h\right] = \frac{|T|}{12} \begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix}$$

as desired.

□

Problem 2. Download the file `discretiseLinearElasticity.m`. We will turn this function into a finite-element solver for Problem (1) next week. Today we implement some core components. The files `video10.mat` and `kiwi.mat` contain arrays P, E and T which define a triangulation on a polygonal computational domain Ω^h. Note that some versions of MATLAB's plotting functions from the PDE Toolbox require extra rows in E and T. If you are not using the PDE Toolbox, then you may delete all but the first two rows of E and all but the first three rows of T, as described in video #10. To import the variables from `video10.mat` or `kiwi.mat` into a structure `msh`, you may use the load command.

(a) Unlike the triangle in Question 1(b), the actual vertices of the k-th triangle are probably not 1, 2 and 3. For instance, the 5th triangle in video #10 has the vertices 7, 10 and 9. In general, the k-th triangle has the vertices `T(1,k)`, `T(2,k)` and `T(3,k)`.

Use Question 1(b) to complete the main function and the `assembleMass` subfunction. Can you do it without for loops?

Hint: In GNU Octave / MATLAB, the command sparse may be helpful.

Solution. The `assembleMass` subfunction is as follows:

```
function Mbar = assembleMass(msh)
%ASSEMBLEMASS Assembles the complete mass matrix including all boundary
%points, discretised with linear finite elements

% need all pairs points, including pairing with themselves:
pointRows = [1,2,3];
pointsAndNeighboursRows = [1,2,3, 2,3,1, 3,1,2];

% Assembly II and JJ arrays such that (II(idx), JJ(idx)) pairs contain all
% pairs of triangle points with themselves and their neighbours
%   NOTE: There will certainly be repeated points, as triangles share
%         vertices. However this is accounted for in the sparse constructor
%         as it adds all repeated indices together, i.e. all contributions
%         from all element mass matrices
II = msh.T(pointsAndNeighboursRows, :);
JJ = repmat( msh.T(pointRows, :), 3, 1);

% First row will be pairs of points with themselves, so value is 2/12*|T|
% 2nd/3rd rows are pairs of points with other points, so value is 1/12*|T|
S  = repmat( [ (2/12) * msh.A
               (1/12) * msh.A
               (1/12) * msh.A ], 3, 1);

% complete mass matrix
Mbar = sparse(II,JJ,S,msh.np,msh.np);

end
```

□

(b) Write a script `hw5.m` to plot the triangular mesh and the sparsity pattern of the mass matrix that the function `discretiseLinearElasticity` returns (you don't have to remove the rows/columns corresponding to boundary points). Do this for both data sets `video10.mat` and `kiwi.mat`. Make sure your plots are not distorted by using the `axis equal` command.

Hint: In installations of MATLAB with the PDE Toolbox, the command `pdemesh` may be helpful. In GNU Octave and MATLAB without the PDE Toolbox, the command `trimesh` may be helpful.

Solution. The `hw5.m` script is as follows:

```
% Plot triangular meshes and corresponding mass matrix sparsity patterns
% for 'video10.mat' and 'kiwi.mat' datasets

for fname = {'video10', 'kiwi'}
    msh = load(strcat(fname{1},'.mat'));
    Mbar = discretiseLinearElasticity(msh);

    figure, subplot(1,2,1);
    pdeplot(msh.P,msh.E,msh.T);
    axis equal

    subplot(1,2,2);
    spy(Mbar);
end
```

□

(c) Add extra commands to this script to plot your favourite function u^h on the kiwi domain and compute its L^2-norm. Constant functions are not allowed! Make sure your plots are not distorted.

Hint: The commands `pdeplot` or `trisurf` may be helpful.

Solution. The modified `hw5.m` script is as follows:

```
% Plot triangular meshes and corresponding mass matrix sparsity patterns
% for 'video10.mat' and 'kiwi.mat' datasets

for fname = {'video10', 'kiwi'}
    msh = load(strcat(fname{1},'.mat'));
    Mbar = discretiseLinearElasticity(msh);

    figure, subplot(1,2,1);
    pdeplot(msh.P,msh.E,msh.T);
    axis equal

    subplot(1,2,2);
    spy(Mbar);
end

% Plot my favourite function uh on the kiwi domain, and compute its L2-norm

u = @(x1,x2) sin(2.*pi.*x1).*cos(6.*pi.*x2);
msh = load('kiwi.mat');
Mbar = discretiseLinearElasticity(msh);

uh = u(msh.P(1,:),msh.P(2,:)).';
L2norm = uh'*Mbar*uh;

figure
pdeplot(msh.P,msh.E,msh.T,'xydata',uh);
title(sprintf('$||u^h||^2_2 = %6f$',L2norm));
axis equal
```

□

Your Learning Progress What is the one most important thing that you have learnt in this assignment?

I learned how to deal with the mess of indices that are finite element meshes! The pros of FEM seems to outweigh the cons so far, though - fumbling with indices is a small price to pay for being able to solve PDEs on irregular domains.

What is the most substantial new insight that you have gained from this course this week? Any *aha moment*?

I didn't have any particular *aha moment*, although finally getting the FEM indices correct (and without any `for`-loops) was satisfying, especially considering that all-in-all it looks rather elegant.