MEC3075F - Computer Methods for Mechanical Engineering

University of Cape Town:
Department of Mechanical Engineering

Project 2

Overview

In this project you will find the temperature distribution across a two-dimensional plate based on boundary conditions using the finite element method. The project requires the solution of systems of linear equations and Gauss quadrature. These linear system have to be algorithmically assembled based on boundary conditions and discretisation size. You are asked to use several different techniques and to evaluate their suitability for this problem.

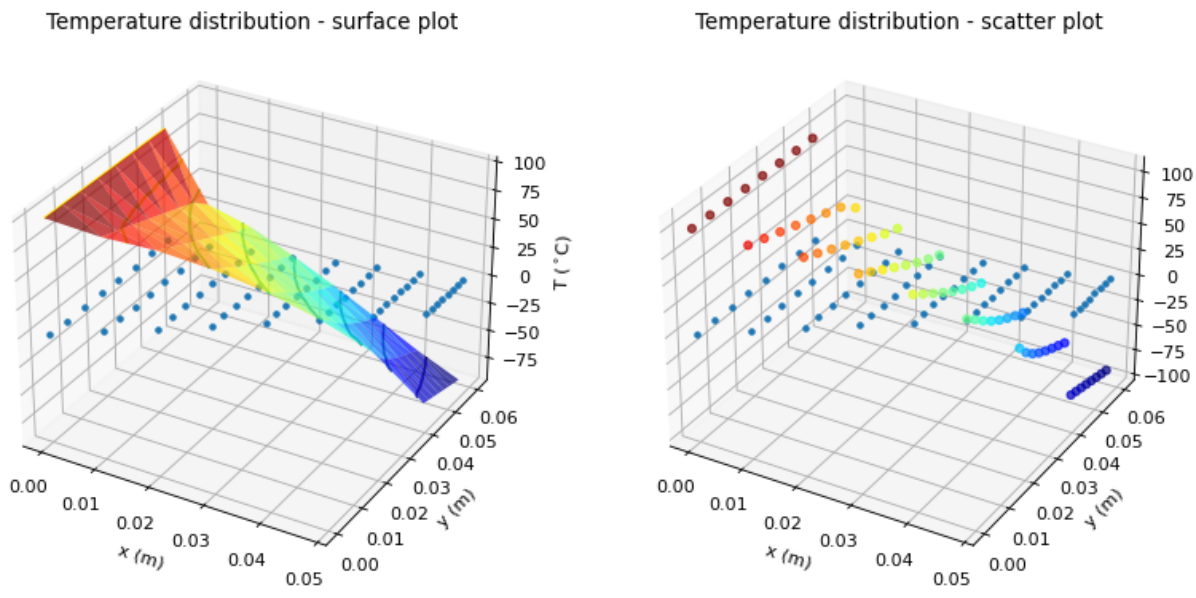

Figure 1: Temperature distribution over a trapezoidal plate. Nodes shown in blue.

Core skills

In completing this project you will demonstrate that you are able to:

- Extract data from text files using Python
- Use Python to approximate double integrals using Gauss quadrature with your own implementation.
- Use Python to assemble an element conduction matrix using the finite element method.
- Use Python to obtain an LU factorisation of a square matrix.
- Use Python to solve repeated systems of linear equations with different right-hand sides using LU factorisation.
- Use matplotlib or seaborn to create graphics and present data.
- Discuss the relative merits of various approaches to solving linear systems with reference to a specific problem.

Motivation

The purpose of this project is to assess your ability to use Python to analyse an engineering problem. The project is intended to serve the second of two opportunities you have in this course to demonstrate the associated graduate attribute at the exit level for your degree.

ECSA graduate attribute 5

Graduate attribute 5 is defined as:
"Engineering methods, skills and tools, including information technology."

The graduate attribute requires that you:
Demonstrate competence to use appropriate engineering methods, skills and tools, including those based on information technology.

The range statement for the GA is:
A range of methods, skills and tools appropriate to the disciplinary designation of the program including:

- Discipline-specific tools, processes or procedures;
- Computer packages for computation, modelling, simulation, and information handling;
- Computers and networks and information infrastructures for accessing, processing, managing, and storing information to enhance personal productivity and teamwork.

Where and how is this learning objective assessed?

This outcome is assessed in each of the projects. You are required to demonstrate your ability to select appropriate analysis tools, implement them, and perform the required analyses. Marks are awarded for the implementation as well as for the project report. The mark breakdown is indicated in the marking rubric later in this document.

What constitutes satisfactory performance?

A mark of 50% in either of the projects will be considered sufficient demonstration of the graduate attribute. This is ensured through the structure of the mark rubric for each project.

What strategy is to be followed should this learning objective not be satisfactorily attained?

If you do not obtain more than 50% for either either project you will be given an opportunity to submit and addendum to Project 2 to address any shortcomings. If a you fail to achieve 50% for either project after the submission of the addendum, you will be refused DP for the course and will be required to repeat it.

Outline

In this project we will be considering the steady state transfer of heat across a uniform plate. We will assume that the plate can be modelled in two dimensions and that the temperature distribution will be governed by the partial differential equation

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0$$

which is known as the Poisson equation.

The finite element method (FEM) is widely used in mechanical engineering, primarily in the stress analysis of components and structures, but also in many other applications including heat conduction, vibration analysis, and more. The method is also extensively used in financial modelling and several scientific fields.

The FEM is taught in detail in a pair of post-graduate courses at UCT, MEC5063Z and MEC5064Z. It is therefore not feasible to teach all the fine detail of the method for this project. What follows is a brief, non-rigorous description of the method, and the formulae you will need. To reduce the complexity of the project, you will also be given a Python file with useful functions.

In the FEM we divide the *domain*, the area or volume we want to analyse, into discrete parts. These parts are known as *elements*. Each element is defined by *nodes* which are points in space. The elements are connected to each other at the nodes, and by making assumptions about what happens on each single element, we are able to form an approximation across the entire domain.

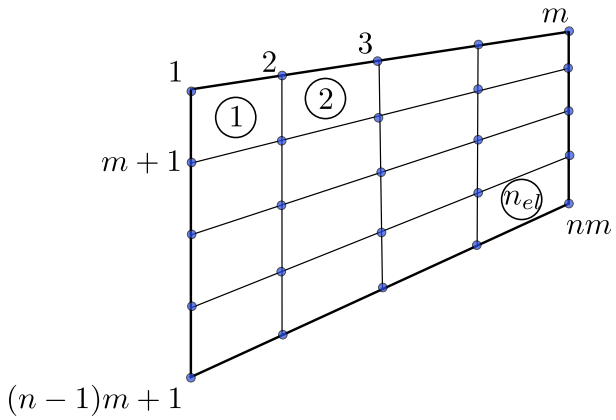

Figure 2: Nodes indicated with blue circles and numbered. Elements are numbered in circles.

In this project we will start numbering nodes and elements in the top left of the domain and continue first horizontally right and then vertically down. Figure 2 shows node numbering for a mesh with m horizontal nodes and n vertical nodes for a total of nm nodes. There are n_{el} elements in the mesh.

In this project we will be considering four-noded quadrilaterals. For each element we will calculate an *element conductivity matrix* (ECM), $\boldsymbol{k}$, which represents the differential equation. The ECMs will then be assembled into a *global conductivity matrix* (GCM), $\boldsymbol{K}$, which, together with a *heat vector*, $\boldsymbol{h}$, will form a linear system. This linear system can then be solved to find values at all of the nodes that approximate the temperatures which correspond to the defined *boundary conditions*.

The finite element method works by transforming the PDE from the differential or *strong* form into an integral or *weak* form. When we convert the PDE from strong to weak form we can use integration by parts to shift the derivatives of the unknown values (in this case the temperatures) in the PDE onto functions known as *shape functions*. In doing so we can create a linear system where the only unknowns are the values we are looking for, without worrying about their derivatives. Thus, when we find each ECM, we need to integrate. This integration is usually performed using Gauss quadrature. As you will see when we cover Gauss quadrature in the course, it is required to transform the limits of integration.

We call the new coordinate system the *isoparametric* coordinates and define our shape functions in terms of these coordinates which are symbolically represented by ξ and η. In higher dimensions, this is usually done using a *Jacobian matrix*, which maps the physical coordinates to coordinates suitable for the quadrature.

The shape functions are

$$N_1 = \frac{1}{4}(1-\xi)(1-\eta); \quad N_2 = \frac{1}{4}(1+\xi)(1-\eta); \quad N_3 = \frac{1}{4}(1+\xi)(1+\eta); \quad N_4 = \frac{1}{4}(1-\xi)(1+\eta)$$

and their derivatives are

$$\begin{aligned}
\frac{\partial N_1}{\partial \xi} &= -\frac{1}{4}(1-\eta); & \frac{\partial N_2}{\partial \xi} &= \frac{1}{4}(1-\eta); & \frac{\partial N_3}{\partial \xi} &= \frac{1}{4}(1+\eta); & \frac{\partial N_4}{\partial \xi} &= -\frac{1}{4}(1+\eta) \\
\frac{\partial N_1}{\partial \eta} &= -\frac{1}{4}(1-\xi); & \frac{\partial N_2}{\partial \eta} &= -\frac{1}{4}(1+\xi); & \frac{\partial N_3}{\partial \eta} &= \frac{1}{4}(1+\xi); & \frac{\partial N_4}{\partial \eta} &= \frac{1}{4}(1-\xi)
\end{aligned}$$

It is convenient to describe a matrix $\boldsymbol{D}$ such that

$$\boldsymbol{D} = \begin{bmatrix} \dfrac{\partial N_1}{\partial \xi} & \dfrac{\partial N_2}{\partial \xi} & \dfrac{\partial N_3}{\partial \xi} & \dfrac{\partial N_4}{\partial \xi} \\ \dfrac{\partial N_1}{\partial \eta} & \dfrac{\partial N_2}{\partial \eta} & \dfrac{\partial N_3}{\partial \eta} & \dfrac{\partial N_4}{\partial \eta} \end{bmatrix}$$

and a the Jacobian matrix $\boldsymbol{J}$ which can be found from

$$\boldsymbol{J} = \boldsymbol{D} \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ x_4 & y_4 \end{bmatrix}$$

where x_i and y_i are the coordinates of node i.

We then define a matrix $\boldsymbol{B}$ as

$$\boldsymbol{B} = \boldsymbol{J}^{-1}\boldsymbol{D}$$

The determinant of the Jacobian matrix is confusingly called the *Jacobian* which for two dimensions is

$$J = \det \boldsymbol{J} = J_{11}J_{22} - J_{21}J_{12}$$

the inverse of the Jacobian matrix for two dimensions is

$$\boldsymbol{J}^{-1} = \frac{1}{J}\begin{bmatrix} J_{22} & -J_{12} \\ -J_{21} & J_{11} \end{bmatrix}$$

The ECM can be constructed from

$$\boldsymbol{k} = \int_{-1}^{1}\int_{-1}^{1} \boldsymbol{B}^T\boldsymbol{B}J\,\mathrm{d}\xi\,\mathrm{d}\eta$$

Each of the ECMs can be assembled into the GCM. This procedure is not complex, but you are given a python function to perform the assembly to reduce the requirements of the project. We can write the assembly procedure as

$$\boldsymbol{K} = \mathop{\mathsf{A}}_{e=1}^{n_{el}} \boldsymbol{k}^e$$

where $\boldsymbol{k}^e$ is the ECM for the e^{th} element.

There is a similar procedure to assemble the heat vector based on internal heat sources and sinks and heat flux on the boundary. However, for this project, we will assume there to be no heat sources or sinks, and boundaries are either insulated (no heat flux) or the temperature is defined. Thus

$$\boldsymbol{h} = \boldsymbol{0}$$

Note that $\boldsymbol{h}$ will have as many rows as the number of nodes.

Once the system is assembled the boundary values can be applied. This is done by modifying $\boldsymbol{K}$ and $\boldsymbol{h}$. Here, we will call the *constrained* GCM and global heat vector $\boldsymbol{K}_c$ and $\boldsymbol{h}_c$ respectively.

The nodal temperatures can now be found by solving the linear system

$$\boldsymbol{K}_c\boldsymbol{t} = \boldsymbol{h}_c$$

where $\boldsymbol{t}$ is a vector listing the temperature at each node.

Finding allocated values

Each student is provided three unique set of boundary conditions.

On the left hand side of the domain the temperature is set to be $100°C$. The top and bottom edges are insulated. The temperature of the right hand side of the domain varies from $a°C$ at the top right corner to $b°C$ at the bottom right corner. Each student is provided three values for a and b.

You should write a python function `get_parameters` which locates your PeopleSoft number in the file `allocation.csv` and stores the values allocated to you for later use.

The file `allocation.csv` is structured in the following format:

student no	a_1	b_1	a_2	b_2	a_3	b_3

where the subscripts refer to the test cases.

Generation of node and element lists

You are provided a function `get_node_element_arrays`. This function takes in three parameters. The first parameter is a NumPy array defining the positions of the corners of the domain and correspond with the coordinates below.The array should be structured as

$$\texttt{corners} = \begin{bmatrix} x_{c1} & y_{c1} \\ x_{c2} & y_{c2} \\ x_{c3} & y_{c3} \\ x_{c4} & y_{c4} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0.044 \\ 0.048 & 0.060 \\ 0.048 & 0.044 \end{bmatrix}$$

where (x_{c1}, y_{c1}) are the coordinates of the four corners in metres.

The second and third inputs should be the number of nodes required along the width and height of the domain respectively. The first function output will be a NumPy array defining the (x, y) coordinates of each node sequentially

$$\texttt{node\_list} = \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \text{etc} & \end{bmatrix}$$

The second function output will be a NumPy array defining the node numbers of each element, starting with the top left node and proceeding counter-clockwise. These will be listed sequentially. For example, for a mesh with 4 nodes horizontally

$$\texttt{element\_list} = \begin{bmatrix} 1 & 5 & 6 & 2 \\ 2 & 6 & 7 & 3 \\ & \text{etc} & & \end{bmatrix}$$

Calculation of ECM

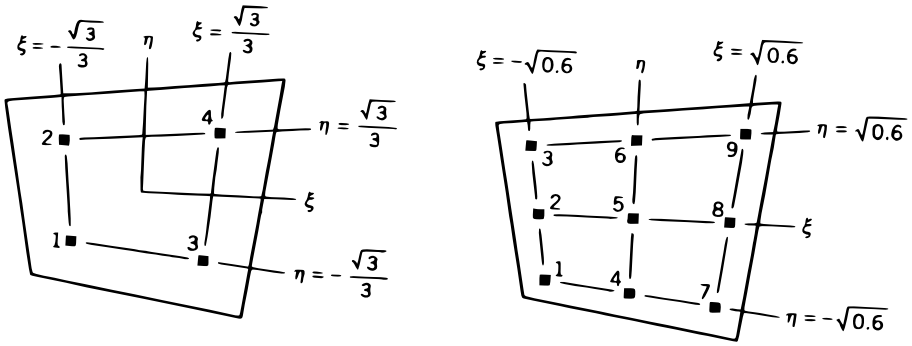

Figure 3: Sampling points for 2D Gauss quadrature of orders 2 and 3

If we have a scalar value $\psi(\xi, \eta)$, a double integral can be approximated by

$$I = \int_{-1}^{1} \int_{-1}^{1} \psi(\xi, \eta) \, \mathrm{d}\xi \, \mathrm{d}\eta = \sum_i \sum_j W_i W_j \psi(\xi_i, \eta_j)$$

where W are the weights.

You should write a function called `find_ecm` which should take as inputs an element number, the NumPy array of node coordinates, the NumPy array of element definitions, and an order

of integration. The function should loop over the Gauss points and use the value of the shape functions at those points, together with the weights to calculate and return the 4×4 element conductivity matrix according to

$$\boldsymbol{k} = \int_{-1}^{1} \int_{-1}^{1} \boldsymbol{B}^T \boldsymbol{B} J \, \mathrm{d}\xi \, \mathrm{d}\eta$$

Assembly of unconstrained GCM

You are provided a function `assemble_gcm` which takes as inputs a NumPy array which lists the nodes of the problem, a NumPy array which lists the elements of the problem, and the function `find_ecm`. The GCM will be returned.

Applying constraints

To apply the constraints (boundary conditions) you need to modify the global linear system.

Create a vector of known temperatures $\boldsymbol{t}_k$ where each non-zero entry corresponds to prescribed temperature at a node. A partially constrained right-hand side vector can then be found according to

$$\boldsymbol{h}_{c1} = -\boldsymbol{K}\boldsymbol{t}_k$$

Now, if a node numbered l is constrained, set the entry in the l^{th} row of $\boldsymbol{h}_{c1}$ to be the temperature value the node is constrained to. Once all the constrained nodes' *degrees of freedom* in $\boldsymbol{h}_{c1}$ have been replaced with the appropriate temperature the resulting vector will be $\boldsymbol{h}_c$.

If a node numbered l is constrained, you should set every entry in the l^{th} row and column of the GCM to 0 and then set the entry $K_{ll} = 1$. The resulting matrix will be $\boldsymbol{K}_c$.

You should write a function `make_constrained_system` which takes in the unconstrained GCM as a NumPy array and the values a and b and returns the constrained GCM and heat vector as NumPy arrays. Remember to account for the fact that node numbers start from 1 but python is zero-indexed when constructing $\boldsymbol{K}_c$ and $\boldsymbol{h}_c$.

Solving for nodal temperatures

The constrained linear system

$$\boldsymbol{K}_c\boldsymbol{t} = \boldsymbol{h}_c$$

can now be solved for the temperatures at each node.

These temperatures can be visualised using matplotlib or seaborn. Creating surface plots on unstructured meshes can be quite difficult, so you may with to only provide contour or scatter plots.

Note that you will have to solve several different linear systems corresponding to the various test cases you are asked to consider.

Project requirements

For the first case you have been allocated, use a range of mesh sizes and the NumPy linear solver to determine an appropriate mesh size. You should motivate for your mesh size in your report.

Once you have selected a mesh, construct the constrained GCM, and find an LU factorisation of this mesh once for all three cases using Cholesky factorisation.

You should write a function called `cholesky` that decomposes the matrix according to $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^T$. The matrix only needs to return the matrix $\boldsymbol{L}$. However, if you perform any row swaps make sure to also return a vector indicating what swaps you performed so that this can be used when you solve the cases. Your function should check that the correct factorisation has been found.

You can then solve the three cases by constructing the constrained heat vector and using forward and backward substitution.

Coding

You are required to create a single Python file named with your student number (*e.g.* `ABCDEF001.py`). This file should contain all the functions you define, as well as execute all the calculations required.

The test cases you must solve are indicated in the file `allocation.csv`.

You must write the following Python functions

1. `get_parameters`
2. `find_ecm`
3. `make_constrained_system`
4. `cholesky`

Notes

This project is meant to encompass one hour of reading time, in which you should understand the problem, one hour of reporting at the end, and an additional eight hours. You are encouraged to adapt the codes provided throughout the course to minimise the amount of coding you have to do.

To get you started with the project, you are provided $\boldsymbol{K}$ for several mesh sizes in compressed format. For instance, the file `K5.npz` contains a NumPy array corresponding to the conductivity matrix for a mesh of 5×5 nodes. The file `load_K.py` shows how to load the values of K into a Python script.

It is recommended that you start the project by loading $\boldsymbol{K}$ and tackling the various parts of the project using NumPy and SciPy functions. This will allow you to quickly cover most aspects of the project (see the marking rubric on the last page). Once you have done this, you can go back and write the function to create $\boldsymbol{k}$ and thus assemble $\boldsymbol{K}$, or replace the NumPy and SciPy ones with your own, as appropriate and time allows.

Specifically, if you are spending too much time on the project you can save time (with an associated reduction in mark) by:

- using a provided NumPy array $\boldsymbol{K}$ instead of constructing it algorithmically
- manually fetching the boundary values and constructing the NumPy vector $\boldsymbol{h}_c$.
- using the `scipy.linalg.cholesky` function instead of your own. Take care, this function returns $\boldsymbol{U}$ instead of $\boldsymbol{L}$ by default.

Submission requirements

The project is due at 09:00 on Monday 8 May 2023. You are only required to submit a digital submission. For your submission you should include your Python script, named with your student number `ABCDEF001.py` and a *brief* report written in Word or LaTeX and saved in PDF format. This report should include a flow diagram of the main script you used, an algorithmic description of the function `get_parameters`, the minimum number of mesh points required to approximate set 1 appropriately, and copies of the temperature distribution plots you created. Discuss when using Gauss reduction, LU factorisation, and iterative techniques is preferable, using examples from your analyses where appropriate.

Late submissions will be penalised 10% for each day or part thereof after the due date and time.

Marking rubric

The following marking rubric will be used:

Requirement	Mark
Project submission in correct format	**2**
Functions	**50**
`get_parameters`	10
`find_ecm`	20
`make_constrained_system`	10
`cholesky`	10
Plotting	**10**
Graphical representation of approximations	7
Appropriate labels and formatting	3
Report	**8**
Total	**70**