Project 2 - TMA4220 - 2021

Deadline : End of the course

Do real-life experimentation using your FEM code

In the second part of the project you are going to make use of your finite element library which you have now built. We are going to apply this to one of several real-life applications.

You should choose **one** of the 4 following options and solve it.

The three first ones are based on the following equations

- $\frac{\partial^2 u}{\partial t^2} = \alpha \nabla^2 u$ - the heat equation
- $\nabla \boldsymbol{\sigma}(\boldsymbol{u}) = -\boldsymbol{f}$ - the linear elasticity equation
- $\rho \frac{\partial^2 \boldsymbol{u}}{\partial t^2} = \nabla \boldsymbol{\sigma}(\boldsymbol{u})$ - the free vibration equation.

1 The heat equation

In this exercise we will model heat transfer in a thin metal sheet when a candle is placed under it. The heat equation reads

$$\begin{aligned} \frac{\partial u}{\partial t} &= \alpha \nabla^2 u + f \\ u(t,x,y)|_{\partial\Omega} &= u^D(t,x,y) \\ u(t,x,y)|_{t=0} &= u_0(x,y) \end{aligned} \tag{1}$$

where α is a positive constant defined by

$$\alpha = \frac{\kappa}{c_p \rho}$$

with κ being the thermal conductivity, ρ the mass density, c_p the specific heat capacity of the material and f a heat source term. We consider, as computational domain Ω, a two dimensional metal sheet, and we want to find the temperature distribution in the metal sheet when a candle fire is placed under it.

1.1 Semi-discretization

We are going to semidiscretize the system by projecting the spatial variables to a finite element subspace X_h. First, assume the source term f is 0 , and multiply (1) by a test function v and integrate over the domain Ω to get

$$\iint_\Omega \frac{\partial u}{\partial t} v dA = -\iint_\Omega \alpha \nabla u \nabla v dA$$

Note that we have only semidiscretized the system, so our unknown u is given as a linear combination of the spatial basis functions, and is continuous in time, i.e.

$$u_h(x, y, t) = \sum_{i=1}^{n} u_h^i(t) \varphi_i(x, y).$$

The Galerkin formulation of the problem, thus reads: Find $u_h \in X_h^D$ such that

$$\begin{aligned} \iint_\Omega \frac{\partial u}{\partial t} v dA &= -\iint_\Omega \alpha \nabla u \nabla v dA, \quad \forall v \in X_h \\ \Rightarrow \quad \sum_i \iint_\Omega \varphi_i \varphi_j dA \frac{\partial u_h^i}{\partial t} &= -\sum_i \iint_\Omega \alpha \nabla \varphi_i \nabla \varphi_j dA u_h^i \quad \forall j \end{aligned}$$

which in turn can be written as the linear system

$$M \frac{\partial \boldsymbol{u}}{\partial t}(t) = -A\boldsymbol{u}(t) \tag{2}$$

which is an ordinary differential equation (ODE) with the matrices defined as

$$\begin{aligned} A = [A_{ij}] &= \iint_\Omega \alpha \nabla \varphi_i \nabla \varphi_j dV \\ M = [M_{ij}] &= \iint_\Omega \varphi_i \varphi_j dV \end{aligned}$$

Construct the matrices A and M as defined above.

1.2 Include the source term

Assume the heat from the candle light is given by a function

$$f(t, x, y) = e^{-\beta\left(x^2+y^2\right)}$$

for some parameter $\beta > 0$. What changes in the formulation from equation (2) do you need in order to include this? Implement these changes and obtain the semi-discretized version of the non-homogeneous heat equation (i.e. with $f \neq 0$).

1.3 Time integration

The system (2) is an ODE, which should be familiar from previous courses. Very briefly an ODE is an equation on the form

$$\frac{d}{dt}\boldsymbol{y}(t) = \boldsymbol{g}(t, \boldsymbol{y})$$

where $\boldsymbol{y} \in \mathbb{R}^n$ and $\boldsymbol{g} : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$ is a regular-enough vector field. Rarely ODEs can be solved exactly, usually the solution is again approximated with a numerical method. The simplest ODE solver available is explicit-Euler method which is based on the following updating rule:

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\boldsymbol{g}(t_n, \boldsymbol{y}_n).$$

More sophisticated methods might be the implicit mid-point method

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\boldsymbol{g}\left(t_n + \frac{h}{2}, \frac{1}{2}\left(\boldsymbol{y}_n + \boldsymbol{y}_{n+1}\right)\right)$$

or higher-order Runge-Kutta methods. Choose an ODE scheme (based on your previous experience and expertise) and implement your time integrator to solve the semi-discretized PDE obtained in previous task. Why did you choose the solver you did?

1.4 Experimentation

The parameters κ, ρ and c_p are problem specific. Try to find (by for example googling it) typical values for these for different types of metal. The free parameters that you can change in this problem are the boundary conditions

$$u(t, x, y)|_{\partial\Omega} = u^D(t, x, y)$$

and the initial condition

$$u(t, x, y)|_{t=0} = u_0(x, y).$$

What do you think are appropriate boundary and initial conditions? Why? The parameter β in the source term decides how large the candle light is. What happens if you let β be large (for example $\beta = 1000$)? Use a source function

$$f(t, x, y) = e^{-\beta\left((x - a\sin(t))^2 + y^2\right)}$$

for different values of a. What happens to the solution when a is large? Do you have to change the time integration somehow?

1.5 Compute the convergence rate

When the analytical solution is not known, a way to compute the convergence rate is to compare the numerical solution with the numerical solution computed using a very refined mesh, i.e. with many elements. Compute the solution for element sizes $h \in \left\{\frac{1}{2^i}, i = 0, 1, 2, 3\right\}$ and compare these with the solution using element size $h = \frac{1}{2^5}$. Compute the relative error and the convergence rate.

2 Linear elasticity equation

We are going to consider the linear elasticity equation. The equations describe deformation and motion in a continuum. While the entire theory of continuum mechanics is an entire course by itself, it will here be sufficient to only study a small part of this: the linear elasticity. This is governed by three main variables $\boldsymbol{u}, \boldsymbol{\varepsilon}$ and $\boldsymbol{\sigma}$. We now define them:

- $\boldsymbol{u} = [u_x, u_y]$: the displacement vector measures how much each spatial point has moved in $(x, y)-$direction.
- $$\boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_{xx} & \varepsilon_{xy} \\ \varepsilon_{xy} & \varepsilon_{yy} \end{bmatrix}$$
  the strain tensor $\boldsymbol{\varepsilon}$ measures how much each spatial point has deformed or stretched,
- $$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} \\ \sigma_{xy} & \sigma_{yy} \end{bmatrix}$$
  the stress tensor $\boldsymbol{\sigma}$ measures how much forces per area are acting on a particular point.

Note that the subscript denotes vector component and **not** the derivative, i.e. $u_x \neq \partial_x \boldsymbol{u}$. These three variables can be expressed in terms of each other in the following way:

$$\boldsymbol{u} = \boldsymbol{u}(\boldsymbol{x})$$
$$\boldsymbol{\varepsilon} = \boldsymbol{\varepsilon}(\boldsymbol{u})$$
$$\boldsymbol{\sigma} = \boldsymbol{\sigma}(\varepsilon)$$

The primary unknown $\boldsymbol{u}$ (the displacement) is the one we are going to find in our finite element implementation. We will have two displacement values for each finite element "node", one in each of the spatial directions.

The relation between strain $\boldsymbol{\varepsilon}$ and the unknown is purely geometric. Consider an infinitesimal small square of size dx and dy, and its deformed geometry as depicted in figure 1. The strain is defined as the stretching of the element, i.e.

$$\varepsilon_{xx} = \frac{\text{length(ab)-length(AB)}}{\text{length(AB)}}.$$

The complete derivations of these quantities is described well in the **Wikipedia article on strain**, and the result is the following relations

$$\varepsilon_{xx}(\boldsymbol{u}) = \frac{\partial u_x}{\partial x}$$
$$\varepsilon_{yy}(\boldsymbol{u}) = \frac{\partial u_y}{\partial y}$$
$$\varepsilon_{xy}(\boldsymbol{u}) = \frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x}.$$

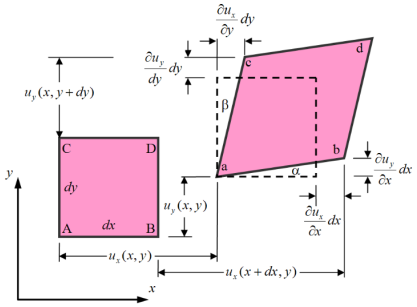

Figure 1: An infinitesimal small deformed rectangle

Note that these relations are the linearized quantities, which will only be true for small deformations.

For the final relation, which connects the deformation to the forces acting upon it, we turn to the material properties. Again, there is a rich literature on the subject, and different relations or physical laws to describe different materials. In our case, we will study small deformations on solid materials like metal, wood or concrete. It is observed that such materials behave elastically when under stress of a certain limit, i.e. a deformed geometry will return to its initial state if all external forces are removed. Experiment has shown that the Generalized Hooks Law is proving remarkable accurate under such conditions. It states the following. Consider a body being dragged to each side by some stress σ_{xx} as depicted in Figure 2. Hooks law states that the forces on a spring is linearly

Figure 2: Deformed geometry under axial stresses

dependent on the amount of stretching multiplied by some stiffness constant, i.e. $\sigma_{xx} = E\varepsilon_{xx}$. The constant E is called Young's modulus. Generalizing upon this law, we see that materials typically contract in the y-direction, while being dragged in the x-direction. The ratio of compression over expansion is called Poisson's ratio ν and is expressed as $\varepsilon_{yy} = -\nu\varepsilon_{xx}$. This gives the following

relations

$$\varepsilon_{xx} = \frac{1}{E}\sigma_{xx}$$
$$\varepsilon_{yy} = -\frac{\nu}{E}\sigma_{xx}$$

Due to symmetry conditions, we see that when applying a stress σ_{yy} in addition to σ_{xx} we get

$$\varepsilon_{xx} = \frac{1}{E}\sigma_{xx} - \frac{\nu}{E}\sigma_{yy}$$
$$\varepsilon_{yy} = \frac{1}{E}\sigma_{yy} - \frac{\nu}{E}\sigma_{xx}.$$

Finally, it can be shown that the relation between the shear strain and shear stress is $\varepsilon_{xy} = 2\frac{1+\nu}{E}\sigma_{xy}$. Collecting the components of ε and σ in a vector, gives us the compact notation

$$\bar{\varepsilon} = C^{-1}\bar{\sigma}$$
$$\begin{bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \varepsilon_{xy} \end{bmatrix} = \begin{bmatrix} \frac{1}{E} & -\frac{\nu}{E} & 0 \\ -\frac{\nu}{E} & \frac{1}{E} & 0 \\ 0 & 0 & 2\frac{1+\nu}{E} \end{bmatrix} \begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{bmatrix}$$

or conversely

$$\bar{\sigma} = C\bar{\varepsilon}$$
$$\begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{bmatrix} = \frac{E}{1-\nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} \begin{bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \varepsilon_{xy} \end{bmatrix} \tag{3}$$

For a body at static equilibrium, we have the governing equations

$$\nabla\boldsymbol{\sigma}(\boldsymbol{u}) = -\boldsymbol{f}$$
$$\begin{bmatrix} \frac{\partial}{\partial x}, \frac{\partial}{\partial y} \end{bmatrix} \begin{bmatrix} \sigma_{xx} & \sigma_{xy} \\ \sigma_{xy} & \sigma_{yy} \end{bmatrix} = -\left[f_x, f_y\right] \tag{4}$$

and some appropriate boundary conditions

$$\boldsymbol{u} = \boldsymbol{g}, \quad \text{on} \quad \partial\Omega_D$$
$$\boldsymbol{\sigma}\cdot\hat{\boldsymbol{n}} = \boldsymbol{h}, \quad \text{on} \quad \partial\Omega_N$$

2.1 Weak form

Show that (4) can be written as the scalar equation

$$\sum_{i=1}^{2}\sum_{j=i}^{2}\int_\Omega \varepsilon_{ij}(\boldsymbol{v})\sigma_{ij}(\boldsymbol{u})dA = \sum_{i=1}^{2}\int_\Omega v_i f_i dA + \sum_{i=1}^{2}\sum_{j=1}^{2}\int_{\partial\Omega} v_i\sigma_{ij}n_j d\boldsymbol{S}$$

(where we have exchanged the subscripts (x, y) with $(1, 2)$) by multiplying with a test function $\boldsymbol{v} = \begin{bmatrix} v_1(x,y) \\ v_2(x,y) \end{bmatrix}$ and integrating over the domain Ω. Moreover, show that this can be written in compact vector form as

$$\begin{aligned}\int_\Omega \bar{\varepsilon}(\boldsymbol{v})^T C \bar{\varepsilon}(\boldsymbol{u}) dA &= \int_\Omega \boldsymbol{v}^T \boldsymbol{f} dA + \int_{\partial\Omega} \boldsymbol{v}^T \boldsymbol{\sigma}\hat{\boldsymbol{n}} d\boldsymbol{S} \\ &= \int_\Omega \boldsymbol{v}^T \boldsymbol{f} dA + \int_{\partial\Omega} \boldsymbol{v}^T \boldsymbol{h} d\boldsymbol{S}\end{aligned}$$

2.2 Galerkin projection

Let now $\boldsymbol{v}$ be a test function in the space X_h of piecewise linear functions on some triangulation T. Note that unlike before, we now have vector test functions. This means that for each node $\hat{i}$, we will have two test functions

$$\begin{aligned}\boldsymbol{\varphi}_{\hat{i},1}(\boldsymbol{x}) &= \begin{bmatrix} \varphi_{\hat{i}}(\boldsymbol{x}) \\ 0 \end{bmatrix} \\ \boldsymbol{\varphi}_{\hat{i},2}(\boldsymbol{x}) &= \begin{bmatrix} 0 \\ \varphi_{\hat{i}}(\boldsymbol{x}) \end{bmatrix}\end{aligned}$$

Let these functions be numbered by a single running index $i = 2\hat{i} + d$, where i is the node number in the triangulation and d is the vector component of the function. Show that by inserting $\boldsymbol{v} = \boldsymbol{\varphi}_j$ and $\boldsymbol{u} = \sum_i \boldsymbol{\varphi}_i u_i$ into (11) you get the system of linear equations

$$A\boldsymbol{u} = \boldsymbol{b}$$

where

$$\begin{aligned}A = [A_{ij}] &= \int_\Omega \bar{\varepsilon}\left(\boldsymbol{\varphi}_i\right)^T C \bar{\varepsilon}\left(\boldsymbol{\varphi}_j\right) dA \\ b = [b_i] &= \int_\Omega \boldsymbol{\varphi}_i^T \boldsymbol{f} dA + \int_{\partial\Omega} \boldsymbol{\varphi}_i^T \boldsymbol{h} d\boldsymbol{S}\end{aligned}$$

(**Hint**: $\bar{\varepsilon}(\cdot)$ is a linear operator).

2.3 Test case

Show that

$$\boldsymbol{u} = \begin{bmatrix} \left(x^2-1\right)\left(y^2-1\right) \\ \left(x^2-1\right)\left(y^2-1\right) \end{bmatrix}$$

is a solution to the problem

$$\begin{aligned}\nabla\boldsymbol{\sigma}(\boldsymbol{u}) &= -\boldsymbol{f} \text{ in } \Omega \\ \boldsymbol{u} &= \boldsymbol{0} \text{ on } \partial\Omega\end{aligned} \tag{5}$$

where

$$\begin{aligned}f_x &= \frac{E}{1-\nu^2}\left(-2y^2 - x^2 + \nu x^2 - 2\nu xy - 2xy + 3 - \nu\right) \\ f_y &= \frac{E}{1-\nu^2}\left(-2x^2 - y^2 + \nu y^2 - 2\nu xy - 2xy + 3 - \nu\right)\end{aligned}$$

and $\Omega = \{(x, y) : \max(|x|, |y|) \leq 1\}$ is the reference square $[-1, 1]^2$.

2.4 Implementation

Modify your Poisson solver to solve the problem (5). Verify that you are getting the correct result by comparing your approximation with the exact solution. The mesh may be obtained through the Grid function `getPlate( )`.

2.5 Stress analysis

Solving (4) with a finite element method gives you the primary unknown: the displacement $\boldsymbol{u}$. If you are interested in derived quantities such as the stresses, these can be calculated from (3). Note that σ is in essence the derivative of u which means that since u is C^0 across element boundaries, then σ will be discontinuous. To get stresses at the nodal values, we propose to average the stresses over all neighbouring elements.

Loop over all elements and evaluate (the constant) stresses on that element. For each node, assign the stresses to be the average stress over all neighbouring elements. This method is called "Stress Recovery".

2.6 Convergence rate

When the analytical solution is not known, a way to compute the convergence rate is to compare the numerical solution with the numerical solution computed using very small elements. Compute the solution for element sizes $h \in \left\{\frac{1}{2^i}, i = 0, 1, 2, 3\right\}$ and compare these with the solution using element size $h = \frac{1}{2^5}$. Compute the relative error and the convergence rate.

3 Vibration analysis

Read the theory on linear elasticity in the option 2, and solve the points 2.1, 2.2, 2.3, 2.4. Then, read this section and conclude with the points 3.1, 3.2, 3.3 described below.

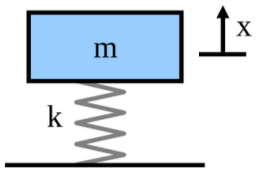

Figure 3: Mass-spring-model

The forces acting on a point mass m by a spring are given by the well known Hooks law:

$$m\ddot{x} = -kx.$$

This can be extended to multiple springs and multiple bodies as in figure 4. The physical laws will now become a system of equations instead of the scalar one above. The forces acting on m_1 is the spring k_1 dragging in negative direction and k_2 dragging in the positive direction. We thus get

$$m_1\ddot{x}_1 = -k_1x_1 + k_2\left(x_2 - x_1\right).$$

This is symmetric, and we have an analogue expression for m_2. The system can be written in matrix form as

$$\begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix}\begin{bmatrix} \ddot{x}_1 \\ \ddot{x}_2 \end{bmatrix} = \begin{bmatrix} -k_1 - k_2 & k_2 \\ k_2 & -k_2 - k_3 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$
$$M\ddot{\boldsymbol{x}} = A\boldsymbol{x}$$

When doing continuum mechanics, it is the exact same idea, but the actual

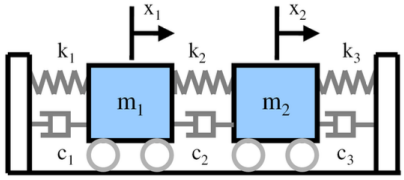

Figure 4: 2 degree-of-freedom mass spring model

equations have some differences. Instead of discrete equations, we have continuous functions in space and the governing equations are

$$\rho\ddot{\boldsymbol{u}} = \nabla\boldsymbol{\sigma}(\boldsymbol{u}).$$

The semi-discretization (discretize just in space) yields the following system of equations

$$M\ddot{\boldsymbol{u}} = -A\boldsymbol{u} \tag{6}$$

with the usual stiffness and mass matrix

$$\boldsymbol{A} = [A_{ij}] = \iint_\Omega \bar{\varepsilon}\left(\boldsymbol{\varphi}_i\right)^T C\bar{\varepsilon}\left(\boldsymbol{\varphi}_j\right)dV$$

$$\boldsymbol{M} = [M_{ij}] = \iint_\Omega \rho\boldsymbol{\varphi}_i^T\boldsymbol{\varphi}_j dV$$

3.1 Building the matrices

Build the 2d mass matrix M as given above (recall that A should already be computed in the previous points). We are now going to search for solutions of the type: $\boldsymbol{u}e^{\omega_i t}$ which inserted into (6) yields

$$\omega^2 M\boldsymbol{u} = A\boldsymbol{u}.$$

3.2 Generalized eigenvalue problem

Note that the previous expression can be rewritten as $(A - \omega^2 M)u = 0$ and this turns out to be a generalized-eigenvalue problem. The term genereliz ed comes fron the fact that 'usual' eigenvalues are associated to the specific choice $M = Id$. Find the 20 first eigenvalues ω_i and eigenvectors $\boldsymbol{u}_i$ corresponding to this problem.

3.3 Animation of the eigenmodes

Let $\boldsymbol{x}_0$ be your initial geometric description (the nodal values). Plot an animation of the eigenmodes by

$$\boldsymbol{x} = \boldsymbol{x}_0 + \alpha \boldsymbol{u}_i \sin(t)$$

You may want to scale the vibration amplitude by some visually pleasing scalar α, and choose the time steps appropriately. Note that for visualization purposes, you will not use the eigenfrequency ω_i since you are interested in viewing (say) $1 - 5$ complete periods of the vibration, but for engineering purposes this is a very important quantity.

4 Improve on the code from Project 1

In this problem you are required to give the most accurate finite element code you can make. It is going to be a technical problem, and will focus on improving your finite element library built with Project 1.

You are going to compute a finite element solution to problems with known exact solutions u and the goal is to get the error $\|u - u_h\|$ as small as possible.

From the theory we know that $\|u - u_h\| \approx Ch^p$, so there are two ways of improving the error

1. decrease h
2. increase p.

Both strategies will require major revisions to your existing library from part 1 of the project; either from implementing new basis functions (p-refinement), or optimizing existing algorithms to handle large number of unknowns (h-refinement).

4.1 Profile your code

Read about profiling at the Python documentation page. Use some profile library to find out which part of your program is the slowest.

4.2 Compute the error

To compute the error of you finite element solution, you will need to integrate this in some norm. One usually measures the error of finite element problems in one of the three norms,

$$
\begin{aligned}
|u - u_h|_{H^1}^2 &= \int_\Omega \nabla (u - u_h) \cdot \nabla (u - u_h) \, dA \\
\|u - u_h\|_{L^2}^2 &= \int_\Omega (u - u_h)^2 \, dA \\
\|u - u_h\|_{L^\infty} &= \sup_{x \in \Omega} |u(x) - u_h(x)|
\end{aligned}
$$

The H^1-seminorm is usually denoted as the energy norm, and satisfies

$$
a(u, u) = |u|_{H^1}^2.
$$

From the Galerkin orthogonality, we know that the finite element solution is the best possible solution (in our solution space) when measured in the energy norm. It is the most natural, and predictive way of computing the error.

Compute your error in energy norm on the problem from task 2 and task 3 of Project 1. To compute the continuous integral of the norms, you will need to split it into a sum of integration over single elements, and evaluate the error functions here.

4.3 Get convergence plots

Again, we will use the analytical solutions from the Project 1. Compute your solution on a series of meshes of different sizes. Plot the error you are getting against the element size h in a log-log plot. What do you see? Explain your findings.

4.4 Convergence rate with reference solution

When the analytical solution is not known, a way to compute the convergence rate is to compare the numerical solution with the numerical solution computed using very small elements. Compute the solution for element sizes $h \in \left\{ \frac{1}{2^i}, i = 0, 1, 2, 3 \right\}$ and compare these with the solution using element size $h = \frac{1}{2^5}$. Compute the relative error and the convergence rate.

4.5 Machine precision

Computers operate with a limited number of decimal digits. For your typical computing environment, this is of the order $\mathcal{O}\left(10^{-15}\right)$. Can you create a 2D finite element solution which reaches machine precision? How large system did you need, and how long time did the computations take? You can use the time-module in Python to measure this.

4.6 The $L-$shape

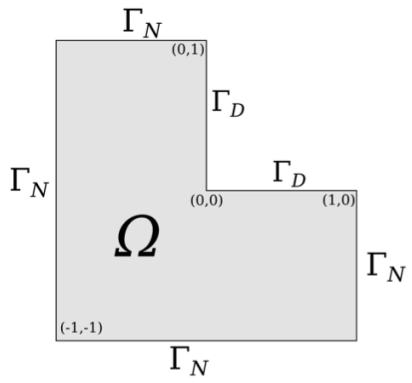

Figure 5: L-shape

Let the domain Ω be the L-shaped domain around the origin (see Figure 5). Solve the Poisson problem

$$\begin{aligned} \nabla^2 u &= 0 \text{ in } \Omega \\ u &= 0 \text{ on } \Gamma_D \\ \frac{\partial u}{\partial n} &= g \text{ on } \Gamma_N. \end{aligned}$$

With the known exact solution

$$u(r,\theta) = r^{2/3} \sin\left(\frac{2\theta + 2\pi}{3}\right), \quad \theta \in \left[\frac{\pi}{2}, 2\pi\right]$$

compute g and run your finite element program on this problem. How fine solution are you able to obtain? How long did your simulation take?