Project - TMA4220 - 2025

Deadline : November 14, at 11:59 pm
Weight on the final grade : 25%

To submit, send an email to `james.jackaman@ntnu.no` attaching a folder with

- the code,
- a PDF file with the explanations of the obtained results.

An alternative would be a Jupyter notebook with the theoretical derivations and code. Specify the group and the group members when you submit. Please also indicate how the project work was distributed between group members.

For this project you may choose one of the following two topics. You may use any high-level finite element library to implement your project. There will be advise/support offered if you choose to use Firedrake. If using Firedrake, you may use this notebook to familiarise yourself with the software.

Option 1: Adaptive finite elements

In this task you will work with a posteriori indicators and use them to adapt meshes with an aim to make the error uniform. In this task you may use any (open source) software you like, but are recommended to use Firedrake.

Task 1.1 Theory

Consider the 2D Helmholtz like equation

$$\begin{aligned} -\Delta u + cu &= f \text{ in } \Omega \\ u &= 0 \text{ on } \partial\Omega. \end{aligned}$$

Find a finite element discretisation of this problem. Show a residual based a posteriori upper bound.

Task 1.2 Sharp and reliable

Fix c to be your favourite (non-zero) constant. Fix $f(x)$ such that you know the exact solution (you may also choose $f(x)$ to be a forcing term to enforce an exact solution). Use the Firedrake utility mesh `UnitSquareMesh(nx,ny)` (or similar) to construct a mesh with N elements on each edge of a unit square domain. Write a function which takes in N, a polynomial degree k which solves the finite element approximation with continuous Lagrange elements. The function should return the (global) a posteriori indicator and the error in the energy norm.

Vary N and investigate the convergence of the a posteriori indicator. Is it sharp and reliable? Does this depend on k?

Task 1.3 Mesh generation

Utilising `mesher.py`, which is available here, write a function which takes an unrefined mesh (with roughly N points along an edge) and refines about a collection of points. This function should take in a list of points to refine. Bonus points if you can generalise this to an iterative list (to perform multiple refinements). This function should write a mesh to file which can then read into Firedrake using `firedrake.Mesh(``mesh_name.msh'')`

Task 1.4 Localisation of indicator

Localise your indicator by projecting/interpolating it into the space of discontinuous constant functions over the mesh, which is written in Firedrake as

```
DG0 = firedrake.FunctionSpace(mesh, ``Discontinuous Lagrange'',0).
```

Build a function which finds the location of the elements which are greater than some tolerance γ.
Hint: Here's a hack to find the locations of points:

- Project the indicator into `DG0`, obtaining a Firedrake function `eta`.
- Convert this function to a vector through `eta.dat.data`.
- Interpolate the function x and the function y into `DG0` as follows.

```
x,y = firedrake.SpatialCoordinate(mesh)
f_x = firedrake.Function(DG0).interpolate(x)
f_y = firedrake.Function(DG0).interpolate(y)
```

- Convert `f_x` and `f_y` to vectors to find the points in x and y associated to a location in a `DG0` vector.

Task 1.5 Adaptive refinement over the unit square

Choose $f(x,y) = \sin(16\pi x)\sin(2\pi y)$. Perform (at least one step of) mesh refinement through

- Solving over a uniform mesh.
- Computing the local indicator and finding the locations where it is greater than some tolerance γ.
- Remeshing your problem (with gmsh).
- Re-solving over the new mesh.

Do the same with

$$f(x,y) = \begin{cases} 10 \text{ for } x,y \in [0.45, 0.55] \\ 1 \text{ for } x,y \in [0.4, 0.45) \cup (0.55, 0.6] \\ 0 \text{ else.} \end{cases}$$

Task 1.6 L-shaped domain

Here we generalise our set up to mixed boundary conditions. Consider the Helmholtz equation

$$\begin{aligned} -\Delta u + cu &= 0 \text{ in } \Omega \\ u &= 0 \text{ on } \Gamma_D \\ \frac{\partial u}{\partial n} &= g \text{ on } \Gamma_N, \end{aligned}$$

where g is defined such that the boundary conditions are consistent with

$$u(x,y)\big|_{\Gamma_N} = \sin\left(2\pi\sqrt{(x-1)^2 + (y-1)^2}\right).$$

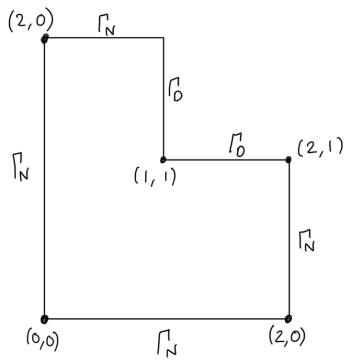

Figure 1: L-shaped domain

Let our domain Ω be L-shaped, as described in Figure 1. We note that at $(1,1)$ the Neumann condition is consistent with the Dirichlet boundary conditions.

Repeat the adaptive refinement procedure given in Task 1.5 for $k = 1$ for this domain. What is the ideal mesh? Why is the case?

Option 2: Biharmonic problem

In this task you will work with the biharmonic problem (a fourth order elliptic PDE), investigate conforming and nonconforming approximations, and study the condition number of the underlying linear system.

Task 2.1 Theory

Consider the fourth order scalar PDE

$$\nabla^4 u := \nabla^2 \nabla^2 u = \nabla \cdot \nabla (\nabla \cdot \nabla u) = f$$

where $u \in H^4(\Omega)$.

1. Show that the weak formulation is described by

$$\left\langle \nabla^2 u, \nabla^2 v \right\rangle + \int_{\partial\Omega} \left(\nabla \left(\nabla^2 u \right) \cdot \boldsymbol{n} \right) v \, \mathrm{d}S - \int_{\partial\Omega} \left(\nabla^2 u \right) (\nabla v \cdot \boldsymbol{n}) \, \mathrm{d}S = \langle f, v \rangle \qquad \forall v \in H^2(\Omega),$$

   where $\boldsymbol{n}$ is the outward unit normal vector to the boundary. How much regularity is needed on u (if you assume zero Dirichlet boundary conditions)?

2. Assuming purely Neumann boundary conditions, i.e.,

$$\begin{aligned} \nabla^2 u &= f \text{ on } \partial\Omega \\ \nabla \left(\nabla^2 u \right) \cdot \boldsymbol{n} &= g \text{ on } \partial\Omega, \end{aligned}$$

   is this problem well posed? Hint: Find a non-trivial nullspace.

3. For the remainder of this project consider the stabilised weak form

$$\left\langle \nabla^2 u, \nabla^2 v \right\rangle + \langle \nabla u, \nabla v \rangle + \langle u, v \rangle + \int_{\partial\Omega} \left(\nabla \left(\nabla^2 u \right) \cdot \boldsymbol{n} - \nabla u \cdot \boldsymbol{n} \right) v \, \mathrm{d}S - \int_{\partial\Omega} \left(\nabla^2 u \right) (\nabla v \cdot \boldsymbol{n}) \, \mathrm{d}S = \langle f, v \rangle \qquad \forall v \in H^2(\Omega). \tag{1}$$

   What is the PDE we are now solving? What are the Dirichlet and Neumann boundary conditions of this modified problem. Prove this form, with zero Neumann boundary conditions, is stable if $f(x) \in L_2(\Omega)$. In the sequel you may fix pure Neumann boundary conditions for this modified problem.

Task 2.2 H^2-conforming approximation with Argyris elements

Discretise (1) with finite elements and implement the method over a unit square domain with non-zero boundary conditions. Choose boundary conditions and $f(x)$ such that you know the exact solution ($f(x)$ may be used as a forcing term). Use the Firedrake utility mesh `UnitSquareMesh(nx,ny)` (or similar) to construct a mesh with N elements on each edge of a unit square domain. Write a function which takes in N, computes the finite element approximation using 5-th order Argyris elements (as introduced in the lectures) over a triangular mesh. The function should output the error in the L_2, H^1 and H^2 norms. Vary N and investigate the convergence of this method.

Task 2.3 Non-conforming approximation with continuous Lagrange elements

Here we consider H^1 methods and modify the bilinear form $a(u,v) := \left\langle \nabla^2 u, \nabla^2 v \right\rangle$ in the spirit of a discontinuous Galerkin approximation. That is to say, in the discrete setting we replace the bilinear form with

$a_h : H^1 \times H^1 \to \mathbb{R}$ such that

$$\begin{aligned} a_h(u_h, v_h) := &\sum_{T \in \mathcal{T}_h} \left\langle \nabla^2 u, \nabla^2 v \right\rangle_T \\ &+ \sum_{e \in \mathcal{E}_h} \Big(\{\boldsymbol{n} \cdot \nabla(\nabla u \cdot n)\} [\![\nabla v \cdot \boldsymbol{n}]\!] + \{\boldsymbol{n} \cdot \nabla(\nabla v \cdot n)\} [\![\nabla u \cdot \boldsymbol{n}]\!] \Big) \mathrm{d}S \\ &+ \sum_{e \in \mathcal{E}_h} \frac{\sigma}{|e|} \int_e [\![\nabla u \cdot \boldsymbol{n}]\!] [\![\nabla v \cdot \boldsymbol{n}]\!] \, \mathrm{d}S, \end{aligned}$$

where $[\![\cdot]\!]$ represents the jump of the solution over the edge e and $\{\cdot\}$ the average. We note that the derivative of a H^1 function is discontinuous. This choice is informed by [2]. The jump term is known as the penalty term as it penalises the discontinity of the derivative and brings the nonconforming solution closer to being in H^2. Note that σ is a user specified parameter, and must be sufficiently large for existence/uniqueness. For ease of implementation, the following Firedrake functions are recommended

- Element diameter: `h = CellSize(mesh)` (given a Firedrake mesh)
- Normal vector: `n = FacetNormal(mesh)` (given a Firedrake mesh)
- Penalty term: `sigma/h * jump(grad(u),n) * jump(grad(v),n) * dS` (given a Firedrake function `u` and test function `v`).

We further define the broken H^2 norm such that for $w \in H^1(\Omega) \cap H^2(T)$ for all $T \in \mathcal{T}_h$ as

$$\|w\|_h := a_h(w, w) + \|w\|_{H^1(\Omega)}.$$

This is sometimes referred to as a mesh dependent norm due to the mesh dependency of the nonconforming bilinear form a_h. When applied to $w \in H^2(\Omega)$ it is equivalent to the H^2 norm.

What is the minimal degree of Lagrange elements required for this discretisation? Build a function similar to that built in Task 2.2 using continuous Lagrange elements. Investigate the dependency of the solution on σ and how this effects smoothness in practice (for example, through plotting the derivative of the solution). Investigate the convergence of this approximation, and compare it against the conforming approximation with Argyris elements.

Warning: If you choose σ to be too small the method may become unstable. The lowest stable value of this parameter depends on the polynomial degree k.

Task 2.4 Preconditioning

Investigate the condition number of the underlying matrix of the nonconforming finite element discretisation. Defining your form as F, this matrix can be accessed as a sparse matrix through

```
A = firedrake.assemble(firedrake.lhs(F),mat_type='aij').M.handle.getValuesCSR()
A = scipy.sparse.csr_matrix((A[2],A[1],A[0]))
```

How does the condition number scale with N and k? Assemble a finite element discretisation of the Poisson equation and use the resulting matrix to precondition your problem. How does this change things? You *do not* need to solve the preconditioned system.

References

[1] Mark Ainsworth and J Tinsley Oden. A posteriori error estimation in finite element analysis. *Computer methods in applied mechanics and engineering*, 142(1-2):1–88, 1997.

[2] Susanne C. Brenner and Li-Yeng Sung. C^0 interior penalty methods for fourth order elliptic boundary value problems on polygonal domains. *J. Sci. Comput.*, 22/23:83–118, 2005.

[3] Patrick E Farrell, Abdalaziz Hamdan, and Scott P MacLachlan. A new mixed finite-element method for H^2 elliptic problems. *arXiv preprint arXiv:2105.07289*, 2021.

[4] David A Ham, Paul HJ Kelly, Lawrence Mitchell, C Cotter, Robert C Kirby, Koki Sagiyama, Nacime Bouziani, Sophia Vorderwuelbecke, T Gregory, Jack Betteridge, et al. Firedrake user manual. 2023.