EPFL
Mathematics section
Dr Martin Licht

Numerical Approximation of Partial Differential Equations

MATH-451 EXAM **04.07.2023** **15h15–18h15**

Last name: .... **First name:** .... **Sciper:** ....

EXAM RULES:

- **CAMIPRO card is mandatory and will be checked.**
- **The exam is recorded only after the student has signed.**
- **Do not detach any page.**
- **Write with blue or black ink. No other colors are allowed.**
- **Mobile phones and other electronic devices must be turned off and stored in the bags.**
- **Please copy all Python code into the exam. Results without code will not be graded.**
- **Justify all your answers. The clarity of the answers will be evaluated as well.**

☐ I read and understood the above rules. Signature: ....

Exercises	Points	Grades
1	10	
2	10	
3	10	
4	10	
TOTAL	40	

Problem 1 (10 points)

Let $\Omega \subset \mathbb{R}^d$ be a bounded Lipschitz domain with boundary $\partial\Omega$. We consider the elliptic boundary value problem

$$\begin{cases} -\epsilon\Delta u + ku &= f \quad \text{in } \Omega, \\ u &= 0 \quad \text{on } \partial\Omega. \end{cases}$$

Here, $f \in L^2(\Omega)$, and $\epsilon > 0$ and $k > 0$ are positive numbers.

(a) Write a weak formulation of the problem in suitable Sobolev spaces and state the bilinear form.

(b) Show that the problem is well-posed by using the Lax-Milgram theorem. How do the constants in the Lax-Milgram theorem depend on k and ϵ? Find a lower estimate for the coercivity constant that depends only on ϵ.

(c) Suppose that V_h is a subspace of the relevant Sobolev space. Write down the Galerkin formulation, explain why it is well-posed, and compare the Galerkin error to the best approximation error (Cea's lemma).

(d) Describe in words what happens in the situation that ϵ is much larger than k.

Problem 2 (10 points)

Consider the one-dimensional reference interval $\hat{K} = [0,1]$.

(a) Describe an affine transformation that maps $[0,1]$ onto $[a,b]$ for any $a < b$.

(b) Define degrees of freedom for the space $\mathbb{P}_1(\hat{K})$ and the corresponding Lagrange basis.

(c) Take the Lagrange basis from the previous subtask together with the two functions

$$\hat{x}^2(1-\hat{x}), \quad \hat{x}(1-\hat{x})^2.$$

Show that these together constitute a basis of $\mathbb{P}_3(\hat{K})$ and state suitable degrees of freedom.

(d) Compute the mass matrix $\hat{M}_1$ for the above basis of $\mathbb{P}_1(\hat{K})$.

(e) Give a formula for the condition number of a symmetric positive-definite matrix. Calculate the condition number of the mass matrix $\hat{M}_1$.

Problem 3 (10 points)

Let $\Omega \subseteq \mathbb{R}^d$ be a bounded Lipschitz domain with boundary $\partial\Omega$. We consider the convection-dominated problem

$$\begin{cases} -\epsilon\Delta u + \beta \cdot \nabla u &= f \quad \text{in } \Omega, \\ u &= 0 \quad \text{on } \partial\Omega. \end{cases}$$

Here, $\beta : \Omega \to \mathbb{R}^d$ is a continuous vector field, $f \in L^2(\Omega)$, and $\epsilon > 0$. Suppose that $\mathcal{T}$ is a triangulation of Ω and write $V_h \subseteq H_0^1(\Omega)$ for the linear finite element space.

(a) Describe the weak formulation (with the suitable choice of Sobolev space) and the Galerkin formulation. State the bilinear form a of the weak formulation.

(b) Describe the linear system of equations for the finite element problem. Describe the entries of the matrix and the right-hand side in terms of integrals. Can you restrict the integrals to subsets of Ω?

(c) Compute the integrals if we use the modified bilinear form

$$a_h(u, v) := a(u, v) + \delta \int_\Omega \nabla u(x) \nabla v(x)\, dx.$$

Problem 4 (10 points)

Let $\Omega = (0,1)^2 \subset \mathbb{R}^2$ be a bounded polyhedral domain. We consider the heat equation

$$\begin{cases} \partial_t u - \Delta u &= f & \text{in } \Omega, \\ u &= 0 & \text{on } \partial\Omega, \\ u = u_0 &= 0 & \text{at } t = 0. \end{cases}$$

over some time interval $[0, T]$ with $T > 0$. Here, $f \in C(0, T; L^2(\Omega))$ and $u_0 \in L^2(\Omega)$. Suppose that $\mathcal{T}$ is a triangulation of Ω and write $V_h \subseteq H_0^1(\Omega)$ for the linear finite element space. We let N be the dimension of that space.

(a) State the semidiscrete formulation in terms of functions and in terms of coefficients. Briefly define all matrices and vectors.

(b) Write the coefficient-form of the semidiscrete formulation in the form $\partial_t U(t) = G(t, U(t))$.

(c) For a fully discrete scheme with time step $\Delta t > 0$, we consider the Runge-Kutta method

$$\begin{aligned} V_1 &= G\left(t^n,\ U^n\right), \\ V_2 &= G\left(t^n + \tfrac{2}{3}\Delta t,\ U^n + \tfrac{2}{3}\Delta t V_1\right), \\ U^{n+1} &= U^n + \Delta t\left(\tfrac{1}{4}V_1 + \tfrac{3}{4}V_2\right). \end{aligned}$$

Implement this scheme by filling out the Python code below, where $u_0 = 1$ and $f = 1$.

For testing your implementation, you can use the time step $\Delta t = 0.0001$.

```python
from util import np  # import numpy
from integrate import assemble_matrix_from_iterables, assemble_rhs_from_iterables, \
                      stiffness_with_diffusivity_iter, mass_with_reaction_iter, \
                      poisson_rhs_iter
from quad import seven_point_gauss_6
from solve import solve_with_dirichlet_data
from mesh import Triangulation


def main():

  # define the mesh vertices of (0, 1)^2 in counterclockwise direction TODO: complete the blank line
  mesh_vertices =

  mesh = Triangulation.from_polygon( mesh_vertices, mesh_size=.1 )

  # as quadrature rule we utilise the severn point gauss scheme of order 6
  quadrule = seven_point_gauss_6()

  # dimension of the FEM space
  ndofs = len(mesh.points)

  # we are freezing the entire boundary
  freezeindices = mesh.boundary_indices

  # we enforce zero Dirichlet on the boundary TODO: complete after `=`
  data =

  Ntimesteps = 15000
  dt = 0.0001

  # assemble the mass and stiffness matrices
  M = assemble_matrix_from_iterables(mesh, mass_with_reaction_iter(mesh, quadrule))
  A = assemble_matrix_from_iterables(mesh, stiffness_with_diffusivity_iter(mesh, quadrule))

  # the source term as a function of x TODO: complete after `:`
  f = lambda x:

  rhs = assemble_rhs_from_iterables(mesh, poisson_rhs_iter(mesh, quadrule, f=f))

  # first iterate TODO complete after `=`
  u0 =

  # initialise
  un = u0

  for iiter in range(Ntimesteps):

    # the matrix we need to invert for V1 TODO complete after `=`
    mat1 =

    # the right hand side term corresponding to V1 TODO complete after `=`
    rhs1 =

    V1 = solve_with_dirichlet_data(mat1, rhs1, freezeindices, data)

    # the matrix we need to invert for V2 TODO complete after `=`
    mat2 =
```

```python
        # the right hand side term corresponding to V2 TODO complete after `=`
        rhs2 =

        V2 = solve_with_dirichlet_data(mat2, rhs2, freezeindices, data)

        # update iterate TODO complete after `+`
        un = un +

    mesh.tripcolor(un)


if __name__ == '__main__':
    main()
```