R. Hiptmair
L. Scarabosio
C. Urzua Torres

Spring Term 2015

Numerical Methods for Partial Differential Equations

ETH Zürich
D-MATH

Exam Preparation Sheet 14

Introduction. This problem sheet collects a few rather complex problems involving both theoretical and C++ implementation parts. They all have a concrete application background and touch on various topics addressed during the course. So they are very well suited to test one's skills when preparing for the main examination on August 22, 2015.

Problem 14.1 Far field computation

In [NPDE, Section 5.6.1] we discussed the convergence of linear output functionals. More precisely, by means of duality techniques, one can prove that the convergence of the functional is of one order higher than the convergence of the solution in the energy norm ([NPDE, Thm. 5.6.7]).

In [NPDE, Section 5.6.2], the computation of heat boundary flux was considered and it was demonstrated how a modification of an output functional can render it continuous with respect to the energy norm, which is an essential prerequisite for applying the duality argument. In this problem, we pursue a similar policy for the computation of another linear output functional, which is important in the simulation of electromagnetic waves.

This problem focuses on time-harmonic wave propagation in linear media. As in Problem 13.4, in this case all time-dependent fields can be represented as

$$U(\boldsymbol{x},t) = \mathrm{Re}(u(\boldsymbol{x})\exp(\mathrm{i}\omega t)) \;, \tag{14.1.1}$$

where $u(\boldsymbol{x}) \in \mathbb{C}$ is a *complex amplitude*, $\omega > 0$ stands for the so-called angular frequency, and Re extracts the real part of a complex number. All equations will be equations for complex amplitudes, from which the actual wave can be recovered by (14.1.1). Hence, in this problem, all unknowns will be *complex valued*. The file `Pardiso.hpp` in the repository has been modified so that it can also handle complex-valued matrices and vectors.

Note: If you work on a local copy of the NPDE library, you must substitute your old `Pardiso.hpp` header with the new one.

The propagation of the so-called TE-mode (TE for transverse electric) of an electromagnetic wave and its interaction with an (infinitely long and straight) penetrable scatterer can be described by the following two-dimensional second-order elliptic boundary value problem for the complex amplitude u of the axial component of the electric field:

$$\begin{aligned} -\Delta u - k^2(\boldsymbol{x})u &= f && \text{in } D \subset \mathbb{R}^2 \;, \\ \mathbf{grad}\, u \cdot \boldsymbol{n} + \mathrm{i}k_d u &= 0 && \text{on } \partial D \;. \end{aligned} \tag{14.1.2}$$

Here, i is the imaginary unit, $D \subset \mathbb{R}^2$ an artificially truncated bounded computational domain, and the piecewise constant discontinuous coefficient $k(\boldsymbol{x})$ is the *wave number*, given by

$$k(\boldsymbol{x}) = \begin{cases} k_s, & \text{for } \boldsymbol{x} \in S \ , \\ k_d, & \text{for } \boldsymbol{x} \in D \setminus \overline{S} \ , \end{cases} \qquad k_s, k_d > 0 \ . \tag{14.1.3}$$

In the following, we use the concrete values $k_s = \sqrt{2}k_0$, $k_d = k_0$, $k_0 = \frac{2\pi}{3}$.

The bounded sub-domain $S \subset D$ is the space occupied by the scattering object, see Figure 14.1. The source function $f = f(\boldsymbol{x})$ is given by

$$f(\boldsymbol{x}) = (k^2(\boldsymbol{x}) - k_d^2)u_i(\boldsymbol{x}) \ , \tag{14.1.4}$$

with the so-called incident wave

$$u_i(\boldsymbol{x}) = \mathrm{e}^{\mathrm{i}k_d x_1} \ , \tag{14.1.5}$$

a plane wave impinging from the right, see Figure 14.1.

Remark: The solution u of (14.1.2) represents the so-called scattered field, that is, the perturbation of the incident field due to the presence of the scattering objects. The total field that can be measured is described by the complex amplitude $u_{\mathrm{tot}} = u + u_i$.

Remark: Actually the wave propagation problem is posed on the unbounded domain $\mathbb{R}^2$, which, however, is outside the scope of every mesh based discretization. Therefore, computations are done on an artificially truncated domain D, and one tries to take into account the effect of the discarded part of space $\mathbb{R}^2 \setminus \overline{D}$ by means of so-called *absorbing boundary conditions*. The Robin boundary condition in (14.1.2) is a simple variant of these.

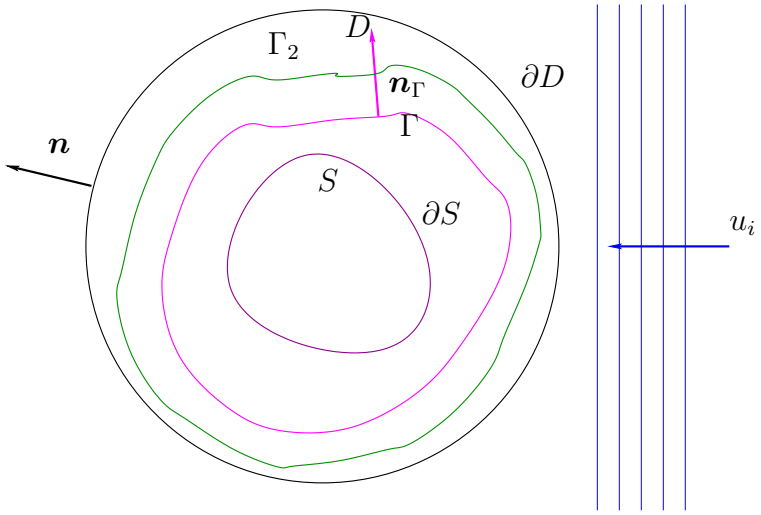

Figure 14.1: Arrangement for 2D scattering problem

Once the scattered field has been computed, the *far field mapping* $F : H^2(D) \mapsto \mathcal{C}^\infty(\mathbb{S}^1)$ is given by

$$F(u)(\hat{\boldsymbol{x}}) = \frac{\mathrm{e}^{\mathrm{i}\pi/4}}{\sqrt{8\pi k_d}} \int_\Gamma u(\boldsymbol{y})(\mathbf{grad}\, w_{\hat{\boldsymbol{x}}})(\boldsymbol{y}) \cdot \boldsymbol{n}_\Gamma(\boldsymbol{y}) \\ - (\mathbf{grad}\, u)(\boldsymbol{y}) \cdot \boldsymbol{n}_\Gamma(\boldsymbol{y}) w_{\hat{\boldsymbol{x}}}(\boldsymbol{y}) \ \mathrm{d}S(\boldsymbol{y}) \ , \quad \hat{\boldsymbol{x}} \in \mathbb{S}^1 \ . \tag{14.1.6}$$

with

$$w_{\hat{\boldsymbol{x}}}(\boldsymbol{y}) = \exp(-\mathrm{i}k_d\hat{\boldsymbol{x}}\cdot\boldsymbol{y})\ , \quad \hat{\boldsymbol{x}}\in\mathbb{S}^1,\ \boldsymbol{y}\in\mathbb{R}^3\ . \tag{14.1.7}$$

This means that the image of F is a function defined on the unit circle $\mathbb{S}^1 = \{\hat{\boldsymbol{x}}\in\mathbb{R}^2 \,|\, \|\hat{\boldsymbol{x}}\| = 1\}$. Here, $\Gamma\subset D\setminus\overline{S}$ is a simple closed path around the scatterer with (exterior) unit normal vector field $\boldsymbol{n}_\Gamma$, see Figure 14.1. For fixed $\hat{\boldsymbol{x}}\in\mathbb{S}^1$ $u\to F(u)(\hat{\boldsymbol{x}})$ is a linear output functional depending on the solution of (14.1.2). The objective of this problem is to investigate its stable numerical evaluation.

Remark: The far field $F(u_s)$ represents the intensity of the scattered field at large distances away S. The integral of $|F(u)|^2$ over some part of the sphere tells us the power carried through that sector by the wave coming back from S.

I. The first part of this problem is concerned with preparatory considerations about (14.1.2) and the far field mapping.

(14.1a) State the variational formulation of (14.1.2) complete with appropriate function spaces.

HINT: The derivation is given in [NPDE, Ex. 2.9.6]. You simply ignore the fact that we deal with $\mathbb{C}$-valued functions. Also recall subproblem (13.4e).

(14.1b) Prove that, if $f\equiv 0$ in (14.1.2), then $u(\boldsymbol{x}) = 0$ and $\mathbf{grad}\, u\cdot\boldsymbol{n}(\boldsymbol{x}) = 0$ for $\boldsymbol{x}\in\partial D$.

HINT: Here you have to use complex conjugation $z\mapsto\overline{z}$ at some point and that $|u(\boldsymbol{x})|^2 = u(\boldsymbol{x})\overline{u}(\boldsymbol{x})$. Test with a function depending on u and consider imaginary and real part of the resulting equation separately.

The result in subproblem (14.1b) implies uniqueness of the solution to the variational problem from subtask (14.1a). Indeed, if u_1 and u_2 are two solutions to the variational formulation that you derived, then u_2-u_1 satisfies the associated homogeneous equation. From subproblem (14.1b), we have that $u_1-u_2\equiv 0$ and $\mathbf{grad}(u_1-u_2)\cdot\boldsymbol{n}\equiv 0$ on ∂D. Then, the so-called *unique continuation principle* implies that $u_1-u_2\equiv 0$ in the whole D.

(14.1c) Explain why, for fixed $\hat{\boldsymbol{x}}\in\mathbb{S}^1$, the functional $u\mapsto F(u)(\hat{\boldsymbol{x}})$ is not continuous on $H^1(D\setminus\overline{S})$.

HINT: You may appeal to the result presented in [NPDE, § 5.6.13].

II. In the second part of this problem we devise a finite element discretization of (14.1.2) based on the linear Lagrangian finite element space $\mathcal{S}_1^0(\mathcal{M})$, where $\mathcal{M}$ is a triangular mesh of D, which is compatible with ∂S in the sense that ∂S is represented by a closed polygon ∂S_N consisting of edges of $\mathcal{M}$. This permits us to associated every cell of $\mathcal{M}$ with either S or $D\setminus\overline{S}$, depending on which side of ∂S_N it is located.

The location of mesh cells is encoded in a vector of integers `ElemFlag`. Each subdomain has a flag, namely the flag 1 is associated to $D\setminus S$, and the flag 2 is associated to S. Then the vector `ElemFlag` has length equal to the number of elements and

$$\begin{aligned} &\texttt{ElemFlag[k]==1} \quad\Leftrightarrow\quad \text{mesh cell with global index } k \text{ belongs to } D\setminus S\ .\\ &\texttt{ElemFlag[k]==2} \quad\Leftrightarrow\quad \text{mesh cell with global index } k \text{ belongs to } S\ . \end{aligned} \tag{14.1.8}$$

Throughout we are going to use the standard tent function basis of $\mathcal{S}_1^0(\mathcal{M})$.

(14.1d) We want to implement a class representing the wave number $-k^2 = -k^2(\boldsymbol{x})$, $\boldsymbol{x} \in D$, with k as given in (14.1.3). To that end, complete the class

```
template <class GridView>
class KappaFunc{
public:
  using calc_t=double;

  KappaFunc(GridView const& gv, std::vector<int> const& ElemFlags,
            calc_t ks_sq, calc_t kd_sq)
    : idset(gv.indexSet()), ElemFlag(ElemFlags), ks_sq_(ks_sq),
      kd_sq_(kd_sq) {}

  template <class Element>
  calc_t operator()(Element const& e) const{

  }

private:
  typename GridView::IndexSet const& idset;
  std::vector<int> const& ElemFlag;
  calc_t ks_sq_,kd_sq_;
};
```

contained in the header `KappaFunc.hpp`, implementing the method

```
template <class Element>
    calc_t operator()(Element const& e) const
```

that, given an element, computes the value of $-k^2 = -k^2(\boldsymbol{x})$ (we assume that the value of k cannot change inside an element).

HINT: Use the information contained in the vector ElemFlag.

(14.1e) Due to the boundary condition in (14.1.2), we also to evaluate $k = k(\boldsymbol{x})$ on the boundary, that is for $\boldsymbol{x} \in \partial D$. To this aim, complete the class

```
 class KappaBdFunc{
public:
  using calc_t=double;

  KappaBdFunc(calc_t kd_sq)
    : kd_sq_(kd_sq) {}

  template <class Element>
  std::complex<calc_t> operator()(Element const& e) const{

  }

private:
  calc_t kd_sq_;
};
```

contained in the header `KappaFunc.hpp`, with the implementation of the method

```
template <class Element>
    calc_t operator()(Element const& e) const
```

that, given an element, computes the value of $k = k(\boldsymbol{x})$. This function will be called only to evaluate k boundary edges, so you don't have to make the distinction made in (14.1.3).

(14.1f) We now implement a class for the source function $f = f(\boldsymbol{x})$ as defined in (14.1.4). For this, complete the class

```
template <class GridView>
class LoadFunc{
public:
  using calc_t=double;

  LoadFunc(GridView const& gv, std::vector<int> const& ElemFlags,
        calc_t ks_sq, calc_t kd_sq)
    : idset(gv.indexSet()), ElemFlag(ElemFlags), ks_sq_(ks_sq),
      kd_sq_(kd_sq) {}

  template <class Coordinate, class Element>
  std::complex<calc_t> operator()(Coordinate const& x, Element
     const& e) const{
  }

private:
  typename GridView::IndexSet const& idset;
  std::vector<int> const& ElemFlag;
  calc_t ks_sq_,kd_sq_;
};
```

with the implementation of the method

```
template <class Coordinate, class Element>
std::complex<calc_t> operator()(Coordinate const& x, Element
   const& e) const
```

that computes the source function in the point with coordinates contained in `x` and belonging to the element `e`.

HINT: Use again the information contained in the vector `ElemFlag`.

(14.1g) Complete the file `main.cc`, provided in the handout, to compute the finite element solution of (14.1.2)–(14.1.5) and write it in a vtk file.

HINT: Of course you should make use of the functions coded in the previous sub-problems.

HINT: For the local mass matrix, both on the domain and on the boundary, you can use the class `LocalMassFarfield` from the file `LocalMassFarfield.hpp` given in the handout. This class is a slight modification of the class `LocalMass` contained in the folder `local/` of the `npde15` library.

HINT: To assemble the right-hand side, use the class `LocalFunctionFarfield` contained in the header `LocalFunctionFarfield.hpp`, which is a slight modification of the function

`LLocalFunction` that you already used to solve the previous assignments. To assemble the finite element matrix, use the routine `MatrixAssembler.hpp` contained in the folder of the handout.

HINT: To select the boundary nodes, you can proceed as in the Radiative Cooling problem that you solved in one of the previous assignments, using the class `LBoundaryNodes`.

III. This part of the problem examines the far field mapping (14.1.6) and its accurate evaluation. This will demonstrate another application of the techniques presented in [NPDE, Section 5.6.2].

(14.1h) Refresh yourself on the "cut-off function trick" used to convert the boundary flux functional to the form [NPDE, Eq. (5.6.15)]. Try to understand again, why this "manipulation" is admissible.

(14.1i) Show that the function $w_{\hat{\boldsymbol{x}}}$ from (14.1.7) satisfies

$$(-\Delta - k_d^2) w_{\hat{\boldsymbol{x}}} = 0 \quad \text{for all } \hat{\boldsymbol{x}} \in \mathbb{S}^1 \,, \tag{14.1.9}$$

where the Laplacian Δ ($\rightarrow$ [NPDE, Rem. 2.5.14]) acts on the independent variable $\boldsymbol{y}$ only.

(14.1j) In formula (14.1.6), Γ stands for any simple closed path around the scatterer. Show that the far field mapping is independent of the path Γ, more precisely, that, for any fixed $\hat{\boldsymbol{x}} \in \mathbb{S}^1$, you get the same value for $F(u)(\hat{\boldsymbol{x}})$ (u a solution of (14.1.2)), no matter whether you use the paths Γ or Γ_2 drawn in Figure 14.1.

HINT: First prove, appealing to Green's formula [NPDE, Thm. 2.5.9], that for smooth functions u and w on a bounded domain Ω:

$$\int_\Omega \Delta u\, w - u\, \Delta w \,\mathrm{d}\boldsymbol{x} = \int_{\partial\Omega} \mathbf{grad}\, u \cdot \boldsymbol{n}\, w - u\, \mathbf{grad}\, w \cdot \boldsymbol{n} \,\mathrm{d}S \,, \tag{14.1.10}$$

where $\boldsymbol{n}$ is the *outward pointing* unit normal vector field on $\partial\Omega$. Then apply this formula to (14.1.6) for a suitable Ω (enclosed between the two paths) and make use of (14.1.9). Watch the orientation of the normal vectors.

(14.1k) As in [NPDE, Section 5.6.2], we choose a cut-off function $\psi \in \mathcal{C}^0(D\setminus S) \cap H^1(D \setminus \overline{S})$ satisfying

$$\psi\big|_{\partial D} = 1 \quad , \quad \psi_{\partial S} = 0 \quad , \quad \mathbf{grad}\, \psi \text{ bounded.} \tag{14.1.11}$$

Show that for $u, w \in H^1(D \setminus \overline{S})$ with $(-\Delta - k_d^2)w = 0$ in $D \setminus \overline{S}$, we have

$$\begin{aligned} &\int_{\partial D} u(\boldsymbol{y})(\mathbf{grad}\, w)(\boldsymbol{y}) \cdot \boldsymbol{n}(\boldsymbol{y}) \,\mathrm{d}S(\boldsymbol{y}) \\ &\qquad = \int_{D\setminus\overline{S}} u(\boldsymbol{y})\psi(\boldsymbol{y}) k_d^2 w(\boldsymbol{y}) + \mathbf{grad}(u\psi)(\boldsymbol{y}) \cdot \mathbf{grad}\, w(\boldsymbol{y}) \,\mathrm{d}\boldsymbol{y} \,. \end{aligned} \tag{14.1.12}$$

HINT: Use Green's formula [NPDE, Thm. 2.5.9].

(14.1l) Show that for the far field mapping $F(u)$ from (14.1.6) holds

$$F(u)(\hat{\boldsymbol{x}}) = \frac{\mathrm{e}^{\mathrm{i}\pi/4}}{\sqrt{8\pi k_d}} \int\limits_{D\setminus\overline{S}} \mathbf{grad}\,\psi(\boldsymbol{y})(u(\boldsymbol{y})\,(\mathbf{grad}\,w_{\hat{\boldsymbol{x}}})(\boldsymbol{y}) - (\mathbf{grad}\,u)(\boldsymbol{y})\,w_{\hat{\boldsymbol{x}}}(\boldsymbol{y}))\,\mathrm{d}\boldsymbol{y}\ , \quad (14.1.13)$$

for any $\hat{\boldsymbol{x}} \in \mathbb{S}^1$, provided that u solves (14.1.2).

HINT: First switch to the integration path ∂D, using the result of sub-problem (14.1j). Then apply (14.1.12) taking into account (14.1.9).

(14.1m) The result of the previous sub-problem suggests that we consider the modified far field mapping

$$F^*(u)(\hat{\boldsymbol{x}}) = \frac{\mathrm{e}^{\mathrm{i}\pi/4}}{\sqrt{8\pi k_d}} \int\limits_{D\setminus\overline{S}} \mathbf{grad}\,\psi(\boldsymbol{y}) \cdot (u(\boldsymbol{y})\,(\mathbf{grad}\,w_{\hat{\boldsymbol{x}}})(\boldsymbol{y}) - (\mathbf{grad}\,u)(\boldsymbol{y})\,w_{\hat{\boldsymbol{x}}}(\boldsymbol{y}))\,\mathrm{d}\boldsymbol{y}\ . \quad (14.1.14)$$

Why is $u \mapsto F^*(u)(\hat{\boldsymbol{x}})$ for fixed $\hat{\boldsymbol{x}} \in \mathbb{S}^1$ a *continuous* linear functional on the energy space $H^1(D \setminus \overline{S})$?

(14.1n) In the previous sub-problem we have seen that F^* is bounded on $H^1(D \setminus \overline{S})$. Well, we can even do better, when choosing special cut-off functions, which satisfy, in addition to (14.1.17),

$$\psi \equiv 1 \quad \text{close to } \partial D\ , \quad \psi \equiv 0 \quad \text{close to } \partial S\ , \quad \psi \in \mathcal{C}^2(\overline{D})\ . \quad (14.1.15)$$

Show that for this choice

$$F^*(u)(\hat{\boldsymbol{x}}) = \frac{\mathrm{e}^{\mathrm{i}\pi/4}}{\sqrt{8\pi k_d}} \int\limits_{D\setminus\overline{S}} u(\boldsymbol{y})(\mathbf{grad}\,\psi(\boldsymbol{y}) \cdot (\mathbf{grad}\,w_{\hat{\boldsymbol{x}}})(\boldsymbol{y}) + \mathrm{div}(w_{\hat{\boldsymbol{x}}}\,\mathbf{grad}\,\psi)(\boldsymbol{y}))\,\mathrm{d}\boldsymbol{y}\ . \quad (14.1.16)$$

Explain, why $u \mapsto F^*(u)(\hat{\boldsymbol{x}})$ is even bounded on $L^2(\Omega)$, if (14.1.15) is satisfied.

HINT: Hardly surprising, an application of Green's formula from [NPDE, Thm. 2.5.9] does the trick.

(14.1o) For fixed $\hat{\boldsymbol{x}} \in \mathbb{S}^1$ we consider the output error $|F^*(u)(\hat{\boldsymbol{x}}) - F^*(u_N)(\hat{\boldsymbol{x}})|$, where $u_N \in \mathcal{S}_1^0(\mathcal{M})$ is the finite element solution introduced in Part **II** of the problem. Moreover, we assume (14.1.15).

Establish what will be the asymptotic dependence of this output error on the meshwidth $h_{\mathcal{M}}$, if $\psi \in \mathcal{C}^2(\overline{D} \setminus S)$ and

- both D and S are discs,
- and we deal with a family of triangular meshes whose shape regularity measures ($\rightarrow$ [NPDE, Def. 5.3.36]) are uniformly small.

HINT: You may take for granted that polygonal boundary approximation does not affect the asymptotic convergence for lowest order Lagrangian finite elements, *cf.* [NPDE, Section 5.5.2]. Then rely on [NPDE, Thm. 5.6.7], state the dual problem in strong form based on (14.1.16) and use elliptic regularity theory from [NPDE, Thm. 5.4.2].

(14.1p) Implement the function

```
template <class GradPsiFunc>
complex_t Farfield(DofHandler const& dofh, GridView const& gv,
   Vector const& u, Coordinate const& p, calc_t const& k,
   GradPsiFunc const& GradPsi)
```

to compute the far field in the point p of the unit sphere. We will use the stable formula (14.1.14) for the far field.

The input argument k denotes the wavenumber for $w_{\hat{\boldsymbol{x}}}$ and u is the vector containing the solution u to Helmholtz equation.

As cut-off function we use

$$\psi(\boldsymbol{y}) = \frac{\|\boldsymbol{y}\|^2 - R_{\text{in}}^2}{R_{\text{out}}^2 - R_{\text{in}}^2}, \tag{14.1.17}$$

where R_{out} is the radius of ∂D and R_{in} the radius of ∂S.

The gradient of $\psi = \psi(\boldsymbol{y})$ is implemented in the class GradPsiFunc, provided in the file GradPsiFunc.hpp of the handout.

(14.1q) Modify the file main.cc implemented in task (14.1g) in order including the far field computation at the current angle (the angles can be initialized at the beginning of the file).

Produce a plot of the absolute value of the far field.

(14.1r) Finally, we want to study the convergence of the far field mapping. Fixing a point $\hat{x} \in \mathbb{S}^1$, we want to estimate the asymptotic behavior of $|F^*(u)(\hat{\boldsymbol{x}}) - F^*(u_N)(\hat{\boldsymbol{x}})|$, as stated in subproblem (14.1o). Of course, as $F^*(u)(\hat{\boldsymbol{x}})$ we consider (14.1.14), as in the previous subproblem. We have at our disposal the meshes, `Helmholtz_mesh1`, ..., `Helmhotz_mesh5`, ordered from the coarser to the finest one, and obtained by successive refinements. Since we don't have an analytical solution for the far field, we consider the discrete solution on the finest grid as reference solution.
Adapt the file `main.cc` including the convergence study. Test the routine with some points in the unit circle $\mathbb{S}^1$; which order of convergence do you observe?

Problem 14.2 Electrostatic Force (Part I)

A straight cylindrical wire is enclosed in a straight cylindrical conducting pipe as depicted in 14.2. There is a voltage drop between both. If the axial extension of both wire and pipe is very long, translational symmetry along the axis can be assumed, which allows two-dimensional modelling. As explained in [NPDE, Section 2.2.2], the electrostatic potential u in the homogeneous space Ω between the surface Γ_1 of the wire and the inner wall Γ_0 of the pipe, is given as the solution of the 2nd-order elliptic boundary value problem

$$-\Delta u = 0 \qquad \text{in } \Omega, \tag{14.2.1}$$

with Dirichlet boundary conditions

$$u = 0 \quad \text{on } \Gamma_0, \qquad u = 1 \quad \text{on } \Gamma_1. \tag{14.2.2}$$

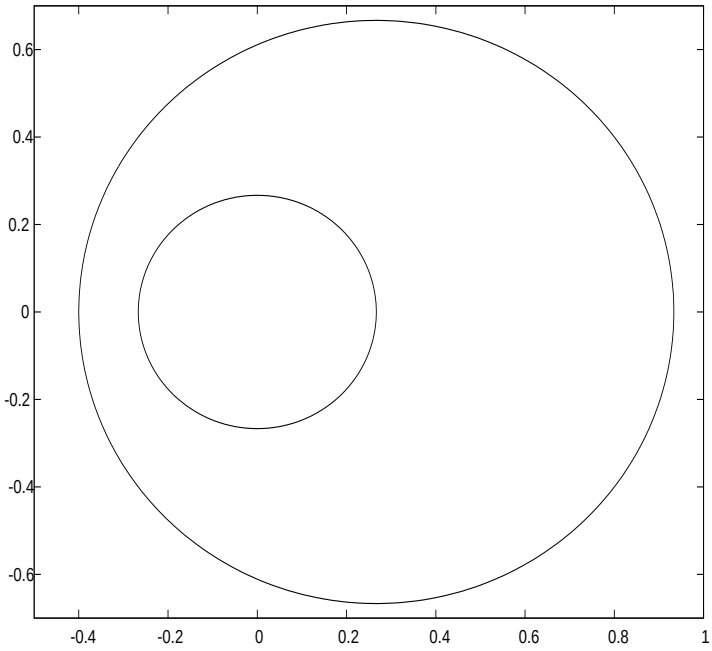

Figure 14.2: The domain used in Problem 14.2

Of course, a suitable scaling is assumed that yielded the non-dimensional equation (14.2.1). In this problem the use of a finite element method to compute the total attracting *force* between wire and pipe approximately is explored.

Caution: In contrast to the cases discussed in class, the output functionals studied in this problem are *non-linear*!

In the above model the electrostatic force on the wire can be computed from the potential u according to

$$\boldsymbol{F}(u) = \frac{1}{2}\int_{\Gamma_1} (\operatorname{grad} u(\boldsymbol{x}) \cdot \boldsymbol{n}(\boldsymbol{x}))\operatorname{grad} u(\boldsymbol{x})\,\mathrm{d}S \tag{14.2.3}$$

To understand this formula recall $\mathbf{E} = -\mathbf{grad}\, u$, see [NPDE, Eq. (2.2.11)], and that $\mathbf{E}\cdot\boldsymbol{n}$ gives the surface charge density on a conductor. Thus the integrand in (14.2.3) can be read as the force density effected by the electric field "pulling at the surface charges".

In order to obtain an analytical solution for u the following special geometric situation will be considered. The pipe wall Γ_0 is a circle centered in $(4/15, 0)$ with radius $2/3$, and the wire boundary Γ_1 is a circle centered in the origin, with radius $4/15$.

(14.2a) Show that the solution to (14.2.1)-(14.2.2) is

$$u(\boldsymbol{x}) = \frac{1}{\log 2}(\log\|\boldsymbol{x}-\boldsymbol{a}\| - \log\|\boldsymbol{x}-\boldsymbol{b}\|) - 1 \tag{14.2.4}$$

where $\boldsymbol{a} = (-16/15, 0)$ and $\boldsymbol{b} = (-1/15, 0)$ are the positions of the point charges.

HINT: You may use a MATLAB or C++ code to verify that the boundary conditions are satisfied. Alternatively, you may appeal to the Apollonius circle theorem that you may have heard about in

secondary school.

(14.2b) Compute the force between wire and pipe for the exact solution found in subproblem (14.2a)

HINT: Exact computation is tedious. Therefore, you may use "overkill" numerical quadrature (e.g. trapezoidal rule with 10^6 points) within a MATLAB or C++ code to obtain a good approximation for the exact value of the force functional. The gradient of u is

$$\operatorname{grad} u(\boldsymbol{x}) = \frac{1}{\log 2}\left(\frac{\boldsymbol{x}-\boldsymbol{a}}{\|\boldsymbol{x}-\boldsymbol{a}\|^2} - \frac{\boldsymbol{x}-\boldsymbol{b}}{\|\boldsymbol{x}-\boldsymbol{b}\|^2}\right).$$

HINT: You should obtain the force value $\boldsymbol{F} = (13.0776, 0)$.

(14.2c) Complete `main.cc` in order to solve (14.2.1)-(14.2.2) using a linear Lagrangian finite element method.

HINT: Use your already implemented classes `DofHandler`, `MatrixAssembler`, `VectorAssembler`, `BoundaryDofs`, and local assemblers, developed in the previous assignments (also available in the corresponding solution folders).

HINT: Use `BoundaryDofs` to find the indices of the boundary edges. Then, loop through each boundary edge and check the norm of the vertices of that edge. If the norms are not much more than $4/15$, the edge is part of the inner boundary.

(14.2d) In `main.cc` add a function

```
template <class Function>
double L2Error(DofHandler dofh, Vector const& FN, Function const&
    Fex)
```

which calculates $\|F_{ex} - F_N\|_{L^2(\Omega)}$ when given the DofHandler `dofh`, the coefficient vector `FN` associated to the finite element function F_N, and `Fex` given in procedural form.

Use it to compute and plot approximate L^2-errors $\|u - u_N\|_{L^2(\Omega)}$ for the given sequence of meshes in terms of the meshwidth or the number of degrees of freedom. Which type of convergence do you (and should you in light of [NPDE, Section 5.6.3]) observe?

HINT: Types of convergence are defined [NPDE, Section 1.6.2]. The sequence of meshes is given by `Annulus_i.msh`, `i=1..3` available in the repository folder.

Problem 14.3 Electrostatic Force (Part II)

In [NPDE, Section 5.6.1] we learned that we can expect enhanced rates of algebraic h-convergence for continuous linear output functionals when evaluating them for finite element Galerkin solutions of linear variational problems. We also saw in [NPDE, Section 5.6.2] for the example of boundary flux functionals that switching to an equivalent continuous form of the functional can bring about dramatic gains in accuracy.

In this problem we apply this policy to the *non-linear* electrostatic force functional from Problem 14.2. We are going to replace the original force functional with an equivalent one that enjoys better continuity properties. This will be explored in numerical experiments.

(14.3a) Show that the functional $u \mapsto \boldsymbol{F}(u)$ from (14.2.3) is *not* linear.

(14.3b) Complete the function

```
Dune::FieldVector<double,2> Force(DofHandler const& dofh, GridView
    const& gv, Vector const& U)
```

(in the handout file `Force.hpp`) that returns the force functional $\boldsymbol{F}(u)$ from (14.2.3) for a finite element solution $u_N \in \mathcal{S}_1^0(\mathcal{M})$, where information on the triangular mesh data structure and boundary nodes are contained in `dofh` and `gv`. The vector `U` passes the nodal basis expansion coefficients of a finite element function.

HINT: This subproblem requires you to compute normals of edges on the (inner) boundary. To get them, you can use the method `unitOuterNormal` contained in the `Dune::Intersection` class.

HINT: Formula (14.2.3) requires integration only on part of the boundary (Γ_1). To select the correct boundary edges, proceed as you did in (14.2c) to set the boundary conditions.

The integrand can be evaluated exactly, so there is no need for quadrature.

(14.3c) Show that there are functions $u \in H^1(\Omega)$ for which $\boldsymbol{F}(u)$ from (14.2.3) is not defined (i.e. "$\boldsymbol{F}(u) = \infty$").

HINT: Examine [NPDE, § 5.6.13]. Try $u(\boldsymbol{x}) = \left(\|\boldsymbol{x}\| - \frac{4}{15}\right)^{\alpha}$ for some suitably chosen α.

(14.3d) Prepare for the three given meshes $\ell = 1, 2, 3$ a plot for the convergence in term of $\boldsymbol{F}(u_\ell)$. What (kind and rate) of convergence $\boldsymbol{F}(u_N) \to \boldsymbol{F}(u)$ in terms of the meshwidth do you observe?

HINT: You can solve this task updating the file `main.cc` from Problem 14.2 with the computation of the force.

(14.3e) According to subproblem (14.3c) it might be possible to encounter infinitely large electrostatic forces attracting the wire towards the pipe. Why will this never be observed?

(14.3f) [NPDE, Section 5.6.2] strikingly demonstrated the benefit of replacing an unbounded linear output functional with an equivalent bounded one. By subproblem (14.3c) the output functional $\boldsymbol{F}$ from (14.2.3) may be haunted by the same problems as [NPDE, Eq. (5.6.11)], though it is non-linear. Thus, it may pay off to reformulate (14.2.3) in a form $\widetilde{\boldsymbol{F}}$ that is continuous on $H^1(\Omega)$, and which agrees with $\boldsymbol{F}$ for all solutions of (14.2.1).

Show that, for all $\boldsymbol{x} \in \Gamma_1$, we have

$$\frac{1}{2}(\operatorname{grad} u(\boldsymbol{x}) \cdot \boldsymbol{n}(\boldsymbol{x}))\operatorname{grad} u(\boldsymbol{x}) = \boldsymbol{T}(u)(\boldsymbol{x}) \cdot \boldsymbol{n}(\boldsymbol{x}),$$

where $\boldsymbol{T}$ is the so-called Maxwell stress tensor

$$\boldsymbol{T}(u)(\boldsymbol{x}) = \operatorname{grad} u(\boldsymbol{x}) \cdot \operatorname{grad} u(\boldsymbol{x})^\top - \frac{1}{2}\|\operatorname{grad} u(\boldsymbol{x})\|^2 \cdot \boldsymbol{I} \in \mathbb{R}^{2,2}\,,$$

where $\boldsymbol{I}$ is the 2×2 identity matrix.

HINT: If u is constant on Γ_1, $\operatorname{grad} u$ will be parallel to $\boldsymbol{n}$ there.

(14.3g) Show that if u solves (14.2.1), then

$$\operatorname{div} \boldsymbol{T}(u)(\boldsymbol{x}) = \mathbf{0}$$

for all $\boldsymbol{x} \in \Omega$. Here, div is row-wise divergence, i.e. it takes the divergence of each *row* $\boldsymbol{T}_i$ of the matrix $\boldsymbol{T}$ (regarded as a vector) and returns a vector:

$$\operatorname{div}\begin{pmatrix} t_{11}(\boldsymbol{x}) & t_{12}(\boldsymbol{x}) \\ t_{21}(\boldsymbol{x}) & t_{22}(\boldsymbol{x}) \end{pmatrix} = \begin{pmatrix} \dfrac{\partial t_{11}}{\partial x_1}(\boldsymbol{x}) + \dfrac{\partial t_{12}}{\partial x_2}(\boldsymbol{x}) \\ \dfrac{\partial t_{21}}{\partial x_1}(\boldsymbol{x}) + \dfrac{\partial t_{22}}{\partial x_2}(\boldsymbol{x}) \end{pmatrix}. \tag{14.3.1}$$

(14.3h) Show that if u solves (14.2.1), then

$$\boldsymbol{F}(u) = \widetilde{\boldsymbol{F}}(u) := \int_\Omega \boldsymbol{T}(u)(\boldsymbol{x}) \operatorname{grad} \Psi(\boldsymbol{x}) \,\mathrm{d}\boldsymbol{x}, \tag{14.3.2}$$

where $\Psi \in \mathcal{C}^\infty(\overline{\Omega})$ satisfies $\Psi(\boldsymbol{x}) = 1$ on Γ_1 and $\Psi(\boldsymbol{x}) = 0$ on Γ_0.

HINT: First, show that $\boldsymbol{F}(u) = \int_{\partial\Omega} \Psi \boldsymbol{T} \cdot \boldsymbol{n} \,\mathrm{d}S$, and then use Gauss' theorem (the divergence theorem, [NPDE, Thm. 2.5.7]). Be inspired by the derivation of [NPDE, Eq. (5.6.15)].

(14.3i) Show that $\widetilde{\boldsymbol{F}}(v)$, defined in (14.3.2), is is bounded for all $v \in H^1(\Omega)$, that is

$$|\widetilde{\boldsymbol{F}}(v)| \le C|v|_{H^1}^2,$$

where C does not depend on v.

(14.3j) Code the function

```
Dune::FieldVector<double,2> ForceStable(DofHandler const& dofh,
    GridView const& gv, Vector const& U)
```

(in the handout file `ForceStable.hpp`) that implements the force functional $\widetilde{\boldsymbol{F}}(u)$ from (14.3.2).

HINT: Use $\Psi = u^{\mathrm{ex}}$ (the exact solution from (14.2.4)) and the three-point local quadrature rule that relies on the midpoints of the edges of a triangle.

(14.3k) Compute and plot the errors $\widetilde{\boldsymbol{F}}(u_N) - \boldsymbol{F}(u)$ for the given meshes, describe the observed convergence in terms of the meshwidth and cpompare with subproblem (14.3d).

HINT: Again, modify the file `main.cc` from Problem 14.2.

Problem 14.4 Least-Squares Galerkin Discretization

On a bounded polygon $\Omega \subset \mathbb{R}^2$ we consider the stationary linear advection problem

$$\begin{aligned} \boldsymbol{v}(\boldsymbol{x}) \cdot \operatorname{grad} u &= f \quad \text{in } \Omega, \\ u &= g \quad \text{on } \Gamma_{\mathrm{in}} := \{\boldsymbol{x} \in \partial\Omega \,|\, \boldsymbol{v}(\boldsymbol{x}) \cdot \boldsymbol{n} < 0\}, \end{aligned} \tag{14.4.1}$$

where $\mathbf{v} : \overline{\Omega} \mapsto \mathbb{R}^2$ is a given continuous velocity field, $f \in \mathcal{C}^0(\overline{\Omega})$ a source term, and $g \in \mathcal{C}^0(\overline{\Gamma}_{\mathrm{in}})$ boundary values for the unknown u on the inflow boundary Γ_{in}.

The so-called *least squares variational formulation* of (14.4.1) boils down to a linear variational problem

$$u \in V: \quad \mathsf{a}(u, w) = \ell(w) \quad \forall w \in V, \tag{14.4.2}$$

with

$$\mathsf{a}(u, w) := \langle \boldsymbol{v} \cdot \operatorname{grad} u, \boldsymbol{v} \cdot \operatorname{grad} w \rangle_{L^2}, \quad \ell(w) := \langle \boldsymbol{v} \cdot \operatorname{grad} w, f \rangle_{L^2}. \tag{14.4.3}$$

(14.4a) Specify an appropriate function space V for the least squares variational formulation.

HINT: The Dirichlet boundary conditions in (14.4.1) should be treated as *essential boundary conditions*.

(14.4b) The least squares variational formulation (14.4.2) is equivalent to a minimization problem for a functional J of the form

$$J(u) := \|T(u,f)\|^2_{L^2(\Omega)}, \tag{14.4.4}$$

where T is an expression involving the functions u and f. What is $T(u,f)$ in concrete terms.

(14.4c) Consider the linear 2nd-order scalar elliptic boundary value problem

$$\begin{aligned} -\operatorname{div}(\mathbf{A}(\boldsymbol{x})\operatorname{grad} u) &= f \quad \text{in } \Omega, \\ u &= g \quad \text{on } \Gamma_{\text{in}}, \\ (\mathbf{A}(\boldsymbol{x})\operatorname{grad} u)\cdot \boldsymbol{n} &= 0 \quad \text{on } \partial\Omega \setminus \Gamma_{\text{in}}, \end{aligned} \tag{14.4.5}$$

where $\mathbf{A} : \overline{\Omega} \mapsto \mathbb{R}^{2\times 2}$ is a continuous matrix-valued function with $\mathbf{A}(\boldsymbol{x}) = \mathbf{A}(\boldsymbol{x})^\top$ for all $\boldsymbol{x} \in \Omega$. Which choice of $\mathbf{A}$ makes the bilinear forms of the *standard* (i.e. not least squares) variational formulation of (14.4.5) and the variational problem (14.4.2) agree?

HINT: For the standard variational formulation of (14.4.5), see [NPDE, Eq. (2.4.5)].

(14.4d) Implement the class LocalLaplace which provides a method

```
template <class Element>
void operator()(Element const& e, ElementMatrix &local) const
```

to compute the element matrix associated to the bilinear form for (14.4.5) using linear Lagrangian finite elements and Dune::QuadratureRule<calc_t, elem_dim>.

HINT: Implementation of `LocalMass` in subproblem (8.1e) and `LocalLaplaceFromVector` in subproblem (8.2c) might be useful.

HINT: Keep in mind the constructor LocalLaplaceC(Function const& q) takes a function `q` in procedural form which returns a 2×2-matrix.

(14.4e) In `main.cc` write a method

```
template < class VectorFunction, class Function >
  void solveAdvBVP(DofHandler const& dofh, VectorFunction const& v,
     Function const& g, Vector & U)
```

that solves (14.4.1) in the case $f \equiv 0$ by means of the least squares Galerkin approach based on the variational formulation (14.4.2) and piecewise linear Lagrangian finite elements. The argument `v` provides the velocity field in procedural form. This function should return a column vector $\in \mathbb{R}^2$. The `g`-argument is also given in procedural form and passes the real valued function g. The Vector `U` is filled with the obtained solution.

(14.4f) Complete the class

```
template <class Function, class VectorFunction>
LocalFunctionLSQ(Function const& f, VectorFunction const& v)
```

by writing the method

```
template <class Element, class Vector>
void operator()(Element const& e, Vector &local) const
```

that computes the right-hand side vector for the variational problem (14.4.2), when piecewise linear Lagrangian finite elements are employed for its Galerkin discretization.

The arguments `v` and `f` in the constructor, give the velocity field $\mathbf{v}$ and source term f in procedural form, see subproblem (14.4e). Vertex based quadrature (2D trapezoidal rule) is to be used for local computations.

HINT: Base your implementation on `LocalFunction`.

(14.4g) Assume $g = 0$. Code a method

```
template < class VectorFunction, class Function >
  void solveAdvBVP_LSQ(DofHandler const& dofh, VectorFunction
     const& v, Vector & U)
```

in `main.cc`, that computes the coefficient vector `U` of the least squares solution of (14.4.1) obtained by a linear Lagrangian finite element Galerkin solution of the related least squares variational problem (14.4.2). The arguments have the same meaning as in subproblem (14.4f).

HINT: You may copy large parts of your implementation of `solveAdvBVP` from subproblem (14.4e). Also use `LocalFunctionLSQ`.

HINT: For debugging, you may found the output `test_call_out.txt` corresponding to the given file `main.cc`.

References

[NPDE] Lecture Slides for the course "Numerical Methods for Partial Differential Equations".SVN revision # 79326.

[NCSE] Lecture Slides for the course "Numerical Methods for CSE".

[LehrFEM] LehrFEM manual.

Last modified on July 16, 2015