R. Hiptmair
L. Scarabosio
E. Fonn

Spring Term 2013

Numerical Methods for Partial Differential Equations

ETH Zürich
D-MATH

Homework Problem Sheet 10

Introduction. The focus of this problem sheet is on parabolic evolution problems and their numerical treatment as discussed in [NPDE, Sect. 6.1]. Only the first problem addresses convergence of finite element results for linear output functionals, see [NPDE, Sect. 5.6.1].

Problem 10.1 Far field computation

In [NPDE, Sect. 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. 5.6.5]).

In [NPDE, Sect. 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. In this case all time-dependent fields can be represented as

$$U(\boldsymbol{x},t) = \mathrm{Re}(u(\boldsymbol{x})\exp(i\omega t)) \;, \tag{10.1.1}$$

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 (10.1.1). Hence, in this problem, all unknowns will be *complex valued*. However, MATLAB can deal with complex vectors and matrices as easily as with real ones, so that LehrFEM can immediately be used to compute complex amplitudes.

The propagation of the so-called TM-mode 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:

$$\begin{aligned} -\Delta u - k(\boldsymbol{x})u &= f \quad \text{in } D \subset \mathbb{R}^2 \;, \\ \mathbf{grad}\, u \cdot \boldsymbol{n} + \imath k_d u &= 0 \quad \text{on } \partial D \;. \end{aligned} \tag{10.1.2}$$

Here, $\imath$ 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

$$k(\boldsymbol{x}) = \begin{cases} k_s & \text{, for } \boldsymbol{x} \in S \;, \\ k_d & \text{, for } \boldsymbol{x} \in D \setminus \overline{S} \;, \end{cases} \qquad k_s, k_d > 0 \;. \tag{10.1.3}$$

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 Fig. 10.1. The source function $f = f(\boldsymbol{x})$ is given by

$$f(\boldsymbol{x}) = (k^2(\boldsymbol{x}) - k_d^2)u_i(\boldsymbol{x}) \ , \tag{10.1.4}$$

with the so-called incident wave

$$u_i(\boldsymbol{x}) = e^{\imath k_d x_1} \ , \tag{10.1.5}$$

a plane wave impinging from the right, see Fig. 10.1.

Remark. The solution u of (10.1.2) 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 $u_{\text{tot}} = u + u_i$.

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 (10.1.2) is a simple variant of these.

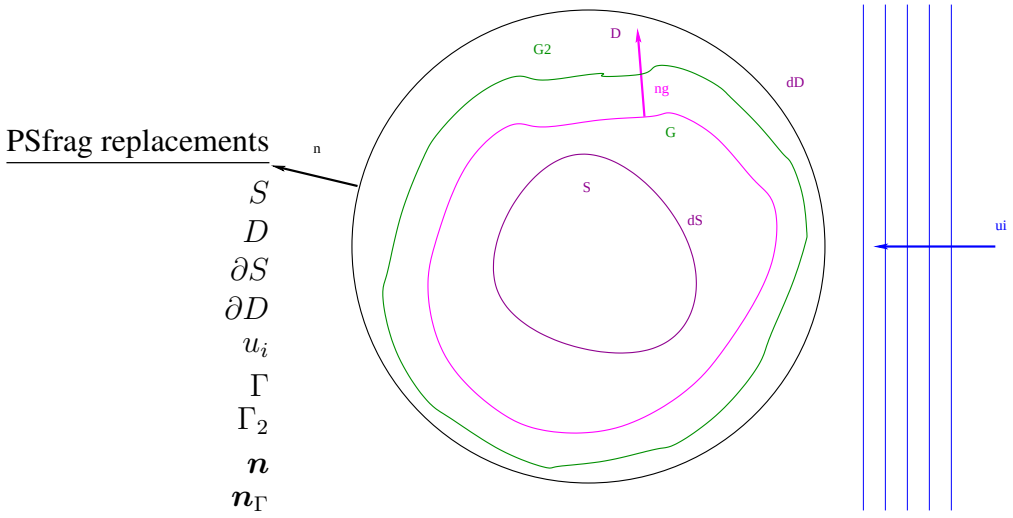

Figure 10.1: Arrangement for 2D scattering problem

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

$$F(u)(\hat{\boldsymbol{x}}) = \frac{e^{i\pi/4}}{\sqrt{8\pi k_d}} \int_\Gamma u(\boldsymbol{y})(\mathbf{grad}\, w_{\hat{\boldsymbol{x}}})(\boldsymbol{y}) \cdot \boldsymbol{n}_\Gamma(\boldsymbol{y}) \\ - (\mathbf{grad}\, u)(\boldsymbol{y}) \cdot \boldsymbol{n}_\Gamma(\boldsymbol{y}) w_{\hat{\boldsymbol{x}}}(\boldsymbol{y})\, dS(\boldsymbol{y}) \ , \quad \hat{\boldsymbol{x}} \in S^1 \ . \tag{10.1.6}$$

with

$$w_{\hat{\boldsymbol{x}}}(\boldsymbol{y}) = \exp(-ik_d \hat{\boldsymbol{x}} \cdot \boldsymbol{y}) \ , \quad \hat{\boldsymbol{x}} \in S^1, \ \boldsymbol{y} \in \mathbb{R}^3 \ . \tag{10.1.7}$$

This means that the image of F is a function defined on the unit circle $S^1 = \{\hat{\boldsymbol{x}} \in \mathbb{R}^2 : \|\hat{\boldsymbol{x}}\| = 1\}$. 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 Fig. 10.1. For fixed $\hat{\boldsymbol{x}} \in S^1$ $u \to F(u)(\hat{\boldsymbol{x}})$ is a linear output functional depending on the solution of (10.1.2). The objective of this problem is to investigate its stable numerical evaluation.

I. The first part of this problem is concerned with preparatory considerations about (10.1.2) and the far field mapping.

(10.1a) State the variational formulation of (10.1.2) complete with appropriate function spaces.

HINT: The derivation is given in [NPDE, Ex. 2.8.5]. You simply ignore the fact that we deal with $\mathbb{C}$-valued functions.

(10.1b) Argue, why the variational formulation obtained in sub-problem (10.1a) has a unique solution.

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.

(10.1c) Explain why, for fixed $\hat{\boldsymbol{x}} \in S^1$, 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, Ex. 5.6.11].

II. In the second part of this problem we will devise a finite element discretization of (10.1.2) 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 will be encoded by the extra field `ElemFlag` in the LehrFEM `mesh` data structure, see [LehrFEM, Sect. **??**], [LehrFEM, Table **??**]. The convention is

$$\texttt{mesh.ElemFlag[k]} == 2 \quad \Leftrightarrow \quad \text{mesh cell with global index } k \text{ belongs to } S\,. \tag{10.1.8}$$

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

(10.1d) Implement a MATLAB function

```
M = MassScattering(Vertices,flag,ks_sq,kd_sq)
```

to compute the local Galerkin matrix ("mass matrix") for the bilinear form

$$(u, v) \mapsto \int_D k(\boldsymbol{x})u(\boldsymbol{x})v(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}\,, \quad u, v \in H^1(D)\,, \tag{10.1.9}$$

and the finite element space $\mathcal{S}_1^0(\mathcal{M})$. Here `Vertices` contains the coordinates of the element vertices; `flag` whether the element is inside or outside the scatterer and uses the same convention (10.1.8). The other arguments are `ks_sq` = k_s^2 and `kd_sq` = k_d^2.

HINT: You may use the LehrFEM function `MASS_LFE` to obtain the element mass matrices (without wave number coefficient).

(10.1e) Implement a LehrFEM compatible MATLAB function

```
Floc = FHandle(x,flag,ks_sq,kd_sq)
```

that evaluates the source function f from (10.1.4) at the point $\boldsymbol{x}$ that is located inside a mesh cell, whose associated element flag is passed in `flag`. The convention (10.1.8) applies and the form (10.1.5) for the incident wave is assumed.

HINT: A reference implementation is provided in `FHandle.p`.

(10.1f) Following the conventions of LehrFEM write a MATLAB function

```
L = assemLoad_LFEScattering(Mesh,ks_sq,kd_sq)
```

that computes the right hand side vector for the Galerkin discretization of (10.1.2)–(10.1.5) by means of $\mathcal{S}_1^0(\mathcal{M})$. Here `Mesh` contains the LehrFEM mesh data structure for $\mathcal{M}$.

HINT: You can rely on the LehrFEM function `assemLoad_LFE` and the function `FHandle` from sub-problem (10.1e).
As quadrature rule you may use `P3O2()`

(10.1g) In LehrFEM style, code a MATLAB function

```
A = assemMat_LFEScattering(Mesh,ks_sq,kd_sq)
```

that computes the sparse Galerkin matrix for the finite element discretization of (10.1.2) based on $\mathcal{S}_1^0(\mathcal{M})$. Here, `Mesh` passes information on the triangular mesh $\mathcal{M}$ in a LehrFEM mesh data structure complete with edge information, see [NPDE, Ex. 3.5.8]. The other arguments are `ks_sq` = k_s^2 and `kd_sq` = k_d^2.

HINT: You may use the LehrFEM functions `assemMat_LFE`, `STIMA_Lap_LFE`, and `assemMat_Bnd_Robin`. The latter function treats the contributions to the bilinear forms arising from the Robin boundary conditions. Also use `assemMat_MassScattering` from sub-problem (10.1d).

A reference implementation is supplied in `assemMat_LFEScattering.p`.

(10.1h) Complete the function `Helmholtz.m`, for which a template is available, that computes the finite element solution of (10.1.2)–(10.1.5) and returns the corresponding coefficient vector.

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

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

(10.1i) Refresh yourself on the "cut-off function trick" used to convert the boundary flux functional to the form [NPDE, Eq. (5.6.12)]. Try to understand again, why this "manipulation" is admissible.

(10.1j) Show that the function $w_{\hat{\boldsymbol{x}}}$ from (10.1.7) satisfies

$$(-\Delta - k_d^2) w_{\hat{\boldsymbol{x}}} = 0 \quad \text{for all } \hat{\boldsymbol{x}} \in S^1 \,, \tag{10.1.10}$$

where the Laplacian Δ ($\rightarrow$ [NPDE, Rem. 2.4.12]) acts on the independent variable $\boldsymbol{y}$ only.

(10.1k) In formula ((10.1h)) Γ stands for any simple closed path around the scatterer. Show that the far field mapping is independent of the path Γ, mor precisely, that for any fixed $\hat{\boldsymbol{x}} \in S^1$ you get the same value for $F(u)(\hat{\boldsymbol{x}})$ (u a solution of (10.1.2)) no matter whether you use the paths Γ of Γ_2 drawn in Figure 10.1.

HINT: First prove, appealing to Green's formula [NPDE, Thm. 2.4.7] that for smooth functions u and w on a bounded domain Ω

$$\int_{\Omega} \Delta u\, w - u\, \Delta w \,\mathrm{d}\boldsymbol{x} = \int_{\partial\Omega} \mathbf{grad}\, u \cdot \boldsymbol{n}\, w - u\ \mathbf{grad}\, w \cdot \boldsymbol{n} \,\mathrm{d}S\ , \tag{10.1.11}$$

where $\boldsymbol{n}$ is the *outward pointing* unit normal vector field on $\partial\Omega$. Then apply this formula to ((10.1h)) for a suitable Ω (enclosed between the two paths) and make use of (10.1.10). Watch the orientation of the normal vectors.

(10.1l) As in [NPDE, Sect. 5.6.2] we choose a cut-off function $\psi \in C^0(D \setminus S) \cap H^1(D \setminus \overline{S})$ satisfying

$$\psi_{|\partial D} = 1 \quad , \quad \psi_{\partial S} = 0 \quad , \quad \mathbf{grad}\, \psi \text{ bounded.} \tag{10.1.12}$$

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

$$\begin{aligned} &\int_{\partial D} u(\boldsymbol{y})(\mathbf{grad}\, w)(\boldsymbol{y}) \cdot \boldsymbol{n}(\boldsymbol{y}) \,\mathrm{d}S(\boldsymbol{y}) \\ &\qquad = \int_{D\setminus\overline{S}} u(\boldsymbol{y})\psi(\boldsymbol{y})k_d^2 w(\boldsymbol{y}) + \mathbf{grad}(u\psi)(\boldsymbol{y}) \cdot \mathbf{grad}\, w(\boldsymbol{y}) \,\mathrm{d}\boldsymbol{y}\ . \end{aligned} \tag{10.1.13}$$

HINT: Use Green's formula [NPDE, Thm. 2.4.7].

(10.1m) Show that for the far field mapping $F(u)$ from ((10.1h)) holds

$$F(u)(\hat{\boldsymbol{x}}) = \frac{e^{i\pi/4}}{\sqrt{8\pi k_d}} \int_{D\setminus\overline{S}} \mathbf{grad}\, \psi(\boldsymbol{y}) \left(u(\boldsymbol{y}) \left(\mathbf{grad}\, w_{\hat{\boldsymbol{x}}}\right)(\boldsymbol{y}) - (\mathbf{grad}\, u)(\boldsymbol{y})\, w_{\hat{\boldsymbol{x}}}(\boldsymbol{y}) \right) \mathrm{d}\boldsymbol{y}\ , \tag{10.1.14}$$

for any $\hat{\boldsymbol{x}} \in S^1$, provided that u solves (10.1.2)

HINT: First switch to the integration path ∂D, using the result of sub-problem (10.1k). Then apply (10.1.13) taking into account (10.1.10).

(10.1n) The result of the previous sub-problem suggests that we consider the modified far field mapping

$$F^*(u)(\hat{\boldsymbol{x}}) = \frac{e^{i\pi/4}}{\sqrt{8\pi k_d}} \int_{D\setminus\overline{S}} \mathbf{grad}\, \psi(\boldsymbol{y}) \cdot \left(u(\boldsymbol{y}) \left(\mathbf{grad}\, w_{\hat{\boldsymbol{x}}}\right)(\boldsymbol{y}) - (\mathbf{grad}\, u)(\boldsymbol{y})\, w_{\hat{\boldsymbol{x}}}(\boldsymbol{y}) \right) \mathrm{d}\boldsymbol{y}\ . \tag{10.1.15}$$

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

(10.1o) 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 (10.1.12),

$$\psi \equiv 1 \quad \text{close to } \partial D\,, \quad \psi \equiv 0 \quad \text{close to } \partial S\,, \quad \psi \in C^2(\overline{D})\,. \tag{10.1.16}$$

Show that for this choice

$$F^*(u)(\hat{\boldsymbol{x}}) = \frac{e^{i\pi/4}}{\sqrt{8\pi k_d}} \int\limits_{D\setminus\overline{S}} u(\boldsymbol{y})\,(\mathbf{grad}\,\psi(\boldsymbol{y}) \cdot (\mathbf{grad}\,w_{\hat{\boldsymbol{x}}})(\boldsymbol{y}) + \operatorname{div}(w_{\hat{\boldsymbol{x}}}\,\mathbf{grad}\,\psi)(\boldsymbol{y}))\,\mathrm{d}\boldsymbol{y}\,. \tag{10.1.17}$$

Explain, why $u \mapsto F^*(u)(\hat{\boldsymbol{x}})$ is even bounded on $L^2(\Omega)$, if (10.1.16) is satisfied.

HINT: Hardly surprising, an application of Green's formula from [NPDE, Thm. 2.4.7] does the trick.

(10.1p) For fixed $\hat{\boldsymbol{x}} \in S^1$ 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 (10.1.16).

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

- both D and S are discs,
- and we deal with a family of triangular meshes whose shape regularity measures ($\rightarrow$ [NPDE, Def. 5.3.26]) 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, Sect. 5.5.2]. Then rely on [NPDE, Thm. 5.6.5], state the dual problem in strong form based on (10.1.17) and use elliptic regularity theory from [NPDE, Thm. 5.4.2].

(10.1q) Implement a MATLAB function

```
J = farfield(Mesh,U,point,k,R_in,R_out)
```

to compute the far field in the point `point` of the unit sphere. We will use the stable formula 10.1.15 for the far field.
As cut-off function use

$$\psi(\boldsymbol{y}) = \frac{\|\boldsymbol{y}\|^2 - R_{\text{in}}^2}{R_{\text{out}}^2 - R_{\text{in}}^2} \tag{10.1.18}$$

where R_{out} is the radius of ∂D and R_{in} the radius of ∂S.
The input argument `k` denotes the wavenumber for $w_{\hat{\boldsymbol{x}}}$ and `U` is the array containing the solution u to Helmholtz equation.

For the implementation, rely on the LehrFEM function `SmartFlux_LFE` (in the LehrFEM folder `Lib/Functionals`) and modify it accordingly. As quadrature rule, you may use P3O2().

HINT: A reference implementation is provided in `farfield.p`.

(10.1r) Complete the MATLAB script `farfieldplot.m` given in the handout to plot the absolute value of the far field versus the angle ϕ used to identify the points on the unit circle.

(10.1s) Finally, we want to study the convergence of the far field mapping. Fixing a point $\hat{x} \in S^1$, we want to estimate the asymptotic behavior of $|F^*(u)(\hat{\boldsymbol{x}}) - F^*(u_N)(\hat{\boldsymbol{x}})|$, as stated in subproblem (10.1p). Of course, as $F^*(u)(\hat{\boldsymbol{x}})$ we consider (10.1.15), as in the previous subproblem. Since we don't have an analytical solution for the far field, we consider the discrete solution on the finest grid as reference solution.
Complete the script `farfieldconv.m` attached in the handout. Test the routine with some points in the unit circle S^1; which order of convergence do you observe?

Listing 10.1: Testcalls for Problem 10.1

```
epr_d = 1;
epr_s = 2;

freq=10e7;
omega=2*pi*freq;
c0=3e8;               %speed velocity
lambda=c0/freq;
k=omega/c0;

r1=lambda/5;         %radius of the inner circle
L=lambda;            %radius of the outer circle

h=(L-r1)/12;
Mesh = mesh_generation(h);

k = Mesh.k;

epr_d = 1;
epr_s = 2;
ks_sq = k*k*epr_s;
kd_sq = k*k*epr_d;

fprintf('\n##MassScattering')
MassScattering(Mesh.Coordinates(Mesh.Elements(1,:),:),2,ks_sq,kd_sq)

fprintf('\n##FHandle');
FHandle([0.5,0.7],2,ks_sq,kd_sq)

fprintf('\n##assemLoad_LFEScattering');
L=assemLoad_LFEScattering(Mesh,ks_sq,kd_sq);
L(1:10)

fprintf('\n##assemMat_LFEScattering');
A=assemMat_LFEScattering(Mesh,ks_sq,kd_sq);
A(1:10,1:10)

fprintf('\n##Helmholtz');
U=Helmholtz(Mesh,k,epr_d,epr_s);
```

```
U(1:10)
```

Listing 10.2: Output for Testcalls for Problem 10.1

```
##MassScattering
ans =

    0.0234    0.0117    0.0117
    0.0117    0.0234    0.0117
    0.0117    0.0117    0.0234

##FHandle
ans =

   2.1932 + 3.7988i

##assemLoad_LFEScattering
ans =

         0
         0
   0.0134 + 0.0381i
   0.0220 - 0.0465i
         0
         0
   0.0516 + 0.0103i
   0.0642 - 0.0068i
         0
         0

##assemMat_LFEScattering
ans =

   (1,1)       1.6880 + 0.2741i
   (9,1)      -0.2218 + 0.0685i
   (2,2)       1.6173 + 0.2741i
   (3,3)       3.4778
   (4,4)       3.4036
   (5,5)       1.6632 + 0.2741i
   (6,6)       1.7906 + 0.2741i
   (7,7)       3.6120
   (8,8)       3.4463
   (1,9)      -0.2218 + 0.0685i
   (9,9)       1.6563 + 0.2741i
  (10,9)      -0.3086 + 0.0685i
   (9,10)     -0.3086 + 0.0685i
  (10,10)      1.8089 + 0.2741i

##Helmholtz
ans =
```

```
-0.0548 - 0.0738i
 0.0830 - 0.3412i
-0.1370 + 0.0210i
-0.7215 - 0.4200i
-0.0002 - 0.1819i
 0.0144 - 0.2001i
-0.3456 - 0.1886i
-0.3857 - 0.2339i
-0.0558 - 0.0746i
-0.0550 - 0.0737i
```

Problem 10.2 Decaying Solution by Implicit Euler Timestepping

In this problem we practice the diagonalization technique that was a key tool in the stability analysis of single step methods for semi-discrete parabolic evolution problems, see the presentation in [NPDE, Sect. 6.1.4.3] and [NPDE, Eq. (6.1.44)].

In class we learned that solutions of the abstract variational linear parabolic evolution problem

$$\begin{aligned} \mathsf{m}(\dot{u},v) + \mathsf{a}(u,v) = 0 \quad \forall v \in V_0, \quad 0 < t < T, \\ u(0) = u_0 \in V_0, \end{aligned} \tag{10.2.1}$$

where both m and a are symmetric positive definite bilinear forms on V_0 that satisfy, see [NPDE, Eq. (6.1.12)],

$$\exists \gamma > 0 : \quad \mathsf{a}(v,v) \geq \gamma \mathsf{m}(v,v) \quad \forall v \in V_0, \tag{10.2.2}$$

display an exponential decay of their m-norm and energy norm, see [NPDE, Lem. 6.1.14].

A simple L(π)-stable [NPDE, Def. 6.1.61] implicit timestepping scheme for the semi-discrete linear parabolic evolution problem

$$\begin{aligned} \mathbf{M}\left\{\frac{\mathrm{d}}{\mathrm{d}t}\vec{\boldsymbol{\mu}}(t)\right\} + \mathbf{A}\vec{\boldsymbol{\mu}} = 0, \quad 0 < t < T, \\ \vec{\boldsymbol{\mu}}(0) = \vec{\boldsymbol{\mu}}_0, \end{aligned} \tag{10.2.3}$$

arising from the *method of lines* ($\rightarrow$ [NPDE, Sect. 6.1.3]) is the implicit Euler method, see [NPDE, Eq. (6.1.23)]. Here, $\mathbf{M}$ and $\mathbf{A}$ denote the Galerkin matrices ($\rightarrow$ [NPDE, Sect. 3.1]) associated with m and a w.r.t. a finite-dimensional trial and test space $V_{0,N} \subset V_0$.

The question is, to what extent the sequence $\vec{\boldsymbol{\mu}}^{(j)}$ generated by the implicit Euler method inherits the exponential decay of the norms stated in [NPDE, Lem. 6.1.14].

(10.2a) Write down the formula for a step of the implicit Euler method producing $\vec{\boldsymbol{\mu}}^{(j)}$ from $\vec{\boldsymbol{\mu}}^{(j-1)}$. Use $\tau > 0$ for the length of the timestep.

(10.2b) From the generalized eigenvalue problem $\mathbf{A}\vec{\boldsymbol{\mu}} = \lambda\mathbf{M}\vec{\boldsymbol{\mu}}$ we deduced the existence of a regular matrix $\mathbf{T} \in \mathbb{R}^{N\times N}$ such that (*cf.* the diagonalization technique discussed in [NPDE, Sect. 6.1.4.3])

$$\mathbf{T}^{-1}\mathbf{M}^{-1}\mathbf{A}\mathbf{T} = \mathbf{D} := \operatorname{diag}(\lambda_1,\ldots,\lambda_N)\,, \quad \mathbf{T}^{\top}\mathbf{M}\mathbf{T} = \mathbf{I}\,.$$

Rewrite the implicit Euler step from subproblem (10.2a) in terms of the transformed coefficient vectors $\vec{\boldsymbol{\eta}}^{(k)} := \mathbf{T}^{\top}\mathbf{M}\vec{\boldsymbol{\mu}}^{(k)}$, $k = j-1, j$.

(10.2c) What will the norms $\left\|\vec{\boldsymbol{\xi}}\right\|_{\mathbf{M}} := (\vec{\boldsymbol{\xi}}^{\top}\mathbf{M}\vec{\boldsymbol{\xi}})^{\frac{1}{2}}$ and $\left\|\vec{\boldsymbol{\xi}}\right\|_{\mathbf{A}} := (\vec{\boldsymbol{\xi}}^{\top}\mathbf{A}\vec{\boldsymbol{\xi}})^{\frac{1}{2}}$ of a coefficient vector $\vec{\boldsymbol{\mu}}$ become for the transformed vector $\vec{\boldsymbol{\eta}} := \mathbf{T}^{\top}\mathbf{M}^{\frac{1}{2}}\vec{\boldsymbol{\mu}}$?

(10.2d) Express $\mathsf{m}(u_N, u_N)$ and $\mathsf{a}(u_N, u_N)$ through the coefficient vector $\vec{\boldsymbol{\mu}}$ associated with a function $u_N \in V_{0,N}$ from the discrete Galerkin trial/test space and through the Galerkin matrices $\mathbf{A}$ and $\mathbf{M}$. How does (10.2.2) read when stated in terms of matrices and coefficient vectors? What is the relationship of γ and the generalized eigenvalues λ_i?

(10.2e) In the sequel we assume a uniform timestep $\tau > 0$. Show that

$$\left\|\vec{\boldsymbol{\mu}}^{(j)}\right\|_{\mathbf{M}} \leq \frac{1}{1+\gamma\tau}\left\|\vec{\boldsymbol{\mu}}^{(j-1)}\right\|_{\mathbf{M}}, \tag{10.2.4}$$

where $\gamma > 0$ is the constant from (10.2.2).

HINT: There are two ways to tackle the problem: rephrase (10.2.4) in terms of $\vec{\boldsymbol{\eta}}^{(k)}$ and look at the implicit Euler method for these transformed coefficient vectors, see subproblem (10.2b). This is another application of the diagonalization technique.

Alternatively, you may use the Cauchy-Schwarz inequality

$$\vec{\boldsymbol{\xi}}^{\top}\mathbf{M}\vec{\boldsymbol{\zeta}} \leq \left\|\vec{\boldsymbol{\xi}}\right\|_{\mathbf{M}}\left\|\vec{\boldsymbol{\zeta}}\right\|_{\mathbf{M}}, \tag{10.2.5}$$

and the implicit Euler recursion formula from subproblem (10.2a)

(10.2f) Now show (10.2.4) with $\|\cdot\|_{\mathbf{M}}$ replaced with $\|\cdot\|_{\mathbf{A}}$.

HINT: Here it is recommended to use diagonalization and the result of subproblem (10.2c).

(10.2g) What is the relationship of the estimates obtained in subproblems (10.2e) and (10.2f) with [NPDE, Lem. 6.1.14].

HINT: Bound $\left\|\vec{\boldsymbol{\mu}}^{(j)}\right\|_{\mathbf{M}}$ and $\left\|\vec{\boldsymbol{\mu}}^{(j)}\right\|_{\mathbf{A}}$ in terms of $\left\|\vec{\boldsymbol{\mu}}^{(0)}\right\|_{\mathbf{M}}$ and $\left\|\vec{\boldsymbol{\mu}}^{(0)}\right\|_{\mathbf{A}}$, respectively. Then, for fixed t consider $\tau \to 0$ and $j \approx t/\tau \to \infty$. Remember the limit

$$e^t = \lim_{n\to\infty}(1+\tfrac{t}{n})^n\,, \quad t \in \mathbb{R}\,.$$

Problem 10.3 Radiative Cooling (Core problem)

This problem is dedicated to the full spatial and temporal discretization of a 2nd-order parabolic evolution problem, see [NPDE, Sect. 6.1]. It will also ask for implementation in LehrFEM in later sub-problems.

The evolution of the temperature distribution $u = u(\boldsymbol{x}, t)$ in a homogeneous "2D body" (occupying the space $\Omega \subset \mathbb{R}^2$) with convective cooling (→ [NPDE, Ex. 2.6.4]) is modelled by the linear second-order parabolic initial-boundary value problem (IBVP) with flux (spatial) boundary conditions (→ [NPDE, Rem. 6.1.7])

$$\begin{aligned} \frac{\partial u}{\partial t} - \Delta u &= 0 && \text{in } \Omega \times [0,T], \\ -\mathbf{grad}\, u \cdot n &= \gamma u && \text{on } \partial\Omega \times [0,T], \\ u(\boldsymbol{x},0) &= u_0(\boldsymbol{x}) && \text{in } \Omega, \end{aligned} \tag{10.3.1}$$

with $\gamma > 0$.

We pursue a method of lines approach, see [NPDE, Sect. 6.1.3]. For the spatial Galerkin semi-discretization of (10.3.1) we employ linear finite elements on a triangular mesh $\mathcal{M}$ of Ω (FE space $\mathcal{S}_1^0(\mathcal{M})$) with polygonal boundary approximation.

(10.3a) Derive the spatial variational formulation of the form $\mathsf{m}(\dot{u}, v) + \mathsf{a}(u, v) = \ell(v)$ for (10.3.1), with suitable bilinear forms a and m, and linear form ℓ. Do not forget to specify the function spaces for $u(t, \cdot)$ and the test function v.

HINT: Combine the considerations leading to [NPDE, Eq. (6.1.8)] with the approach explained in [NPDE, Ex. 2.8.5].

(10.3b) Argue why the total thermal energy

$$E(t) := \int_\Omega u(\boldsymbol{x}, t) \, \mathrm{d}\boldsymbol{x} \, ,$$

decreases with time, if $u_0(\boldsymbol{x}) > 0$ for all $\boldsymbol{x} \in \Omega$.

HINT: Appeal to the heat conduction background to justify the assumption that $u(\boldsymbol{x}, t) \geq 0$ for all $(\boldsymbol{x}, t)$. Use test function $v \equiv 1$ in the variational formulation.

(10.3c) Compute the local mass matrix $\mathbf{M}_{\widehat{K}}$ corresponding to $\mathsf{m}(\cdot, \cdot)$ and the local stiffness matrix $\mathbf{A}_{\widehat{K}}$ corresponding to $\mathsf{a}(\cdot, \cdot)$ for the unit triangle $\widehat{K}$ with vertices $\binom{0}{0}$, $\binom{1}{0}$ and $\binom{0}{1}$. Assume that the edge connecting $\binom{0}{0}$ and $\binom{1}{0}$ forms part of $\partial\Omega$ and that the coefficient γ is constant along this edge.

HINT: See [NPDE, Def. 3.5.13] for the definition of local matrices, and [NPDE, Sect. 3.2.5] for concrete formulas. [NPDE, Lemma 3.5.30] for $d = 2$ may also come handy.

(10.3d) Now we turn to the full spatial semi-discretization of (10.3.1). Implement a MATLAB function

```
[M,A] = getMatLFE(mesh,gamma)
```

that computes the matrices $\mathbf{M}, \mathbf{A} \in \mathbb{R}^{N \times N}$, $N := \dim \mathcal{S}_1^0(\mathcal{M})$, for the semi-discrete evolution

$$\mathbf{M} \frac{\mathrm{d}}{\mathrm{d}t} \vec{\boldsymbol{\mu}}(t) + \mathbf{A} \vec{\boldsymbol{\mu}}(t) = 0 \tag{10.3.2}$$

resulting from the $\mathcal{S}_1^0(\mathcal{M})$-based finite element semi-discretization of (10.3.1), when standard nodal bassis functions are used. Here, the argument `mesh` passes a LehrFEM mesh data structure, and `gamma` supplies the value of γ.

HINT: Do not use the result of sub-problem (10.3c) (which is only valid for a very special triangle). Use already existing LehrFEM functions instead, see [LehrFEM, Sect. **??**].

> In this and all following sub-problems all integrals are to be evaluated by local vertex based quadrature formulas (the trapezoidal rule in one and two dimensions, see [NPDE, Eq. 3.2.21]).

To facilitate the implementation, a MATLAB template is provided in `getMatLFE.m` which already computes the stiffness matrix for $-\Delta$. A reference file is provided as `getMatLFERef.p`.

The edge flag for Robin boundary conditions is `-1`. You may use the provided implementation for `assemMat_Bnd_Cooling`, type "`help assemMat_Bnd_Cooling`" for further information on the function call.

(10.3e) Implement a MATLAB function

```
mufinal = RadTEvl(u0,gamma,mesh,m)
```

that carries out m uniform timesteps of the L-stable SDIRK-2 implicit 2-stage Runge-Kutta method with Butcher scheme

$$\begin{array}{c|cc} \lambda & \lambda & 0 \\ 1 & 1-\lambda & \lambda \\ \hline & 1-\lambda & \lambda \end{array} \qquad \lambda := 1 - \tfrac{1}{2}\sqrt{2}, \tag{10.3.3}$$

in order to solve (10.3.1) over the time interval $[0, 1]$. The finite element Galerkin discretization from subproblem (10.3d) is used in space. The argument `u0` is a column vector that passes the values of the initial temperature distribution in the vertices of the mesh. The return value provides the basis coefficients of the approximation of $u(\cdot, 1)$ of u at $t = 1$. This function will be called within the driver routine `driver_evl.m`.

HINT: From [NPDE, Def. 6.1.26] and [NPDE, Eq. 6.1.27] it should be clear how to obtain the linear systems of equations [NPDE, Eq. (6.1.28)], [NPDE, Eq. (6.1.29)] for the Runge-Kutta increments.

A MATLAB template is provided in `RadTEvl.m`, and a reference implementation in `RadTEvlRef.p`.

(10.3f) Write a MATLAB function

```
avg = lfeavg(u,mesh)
```

that computes $\int_\Omega u \,\mathrm{d}\boldsymbol{x}$ for $u \in \mathcal{S}_1^0(\mathcal{M})$. The argument `u` passes the coefficients of u w.r.t. the standard nodal basis of $\mathcal{S}_1^0(\mathcal{M})$, while `mesh` contains a LehrFEM mesh data structure.

HINT: A reference implementation is provided as `lfeavgRef` (File `lfeavgRef.p`).

(10.3g) For the evolution problem (10.3.1) on $\Omega = (0, 1)^2$ track the behavior of the thermal energy

$$E(t) = \int_\Omega u(\boldsymbol{x}, t) \,\mathrm{d}\boldsymbol{x} \tag{10.3.4}$$

over the period $[0, T]$ for $u_0 \equiv 1$, $\gamma = 1$. Use the fully discrete evolution implemented in `RadTEvl` and extend it to

```
[mufinal,E] = RadTEvl(u0,gamma,mesh,m),
```

where `E` returns approximations for $E(t_k)$ for $k = 0, \ldots, m$ (t_k are the points of the equidistant temporal grid).

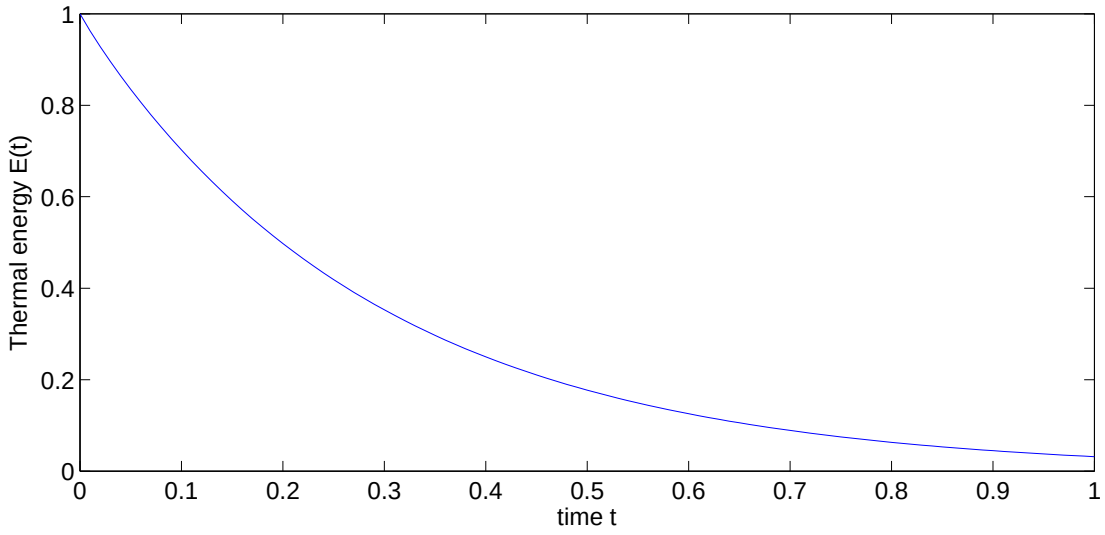

Figure 10.2: Result of subproblem (10.3g)

Extend the MATLAB script `driver_evl.m` to compute $u(x,t)$ for $t = 1$ for the mesh supplied in the file `evlmesh.mat`. Make a plot of u for $t = 0$ and $t = 1$. Plot the approximation for $E(t)$ that you have computed as a function of t for $m = 100$.

HINT: The plot for E is depicted in Figure 10.2.

Listing 10.3: Testcalls for Problem 10.3

```
clear Mesh
g = 1/sqrt(2);
circ = [g g; -g g; -g -g; g -g];
Mesh.Coordinates = [4/15*circ; 4/15 * [ones(4,1),zeros(4,1)] +
   2/3*circ];
Mesh.Elements = [1 8 5; 1 5 6; 1 6 2; 2 6 7; 2 7 3; 3 7 4; 4 7 8; 4
   8 1];
Mesh.ElemFlag = ones(size(Mesh.Elements,1),1);
Mesh = add_Edges(Mesh);
Mesh = add_Edge2Elem(Mesh);
Mesh.BdFlags = transpose([0 0 0 -1 -1 0 0 -1 0 -1 0 0 0 0 0 0]);

fprintf('## getMatLFE')
x = Mesh.Coordinates;
gamma = 2;
[M, A] = getMatLFE(Mesh, gamma); it = 1:5;
M_part = full(M(it, it))
A_part = full(A(it, it))

fprintf('## RadTEvl')
x = Mesh.Coordinates;
u0 = x(:,1) + x(:,2);
gamma = 2;
m = 3;
mufinal = RadTEvl(u0, gamma, Mesh, m)
```

```
24
25 fprintf('## lfeavg')
26 x = Mesh.Coordinates;
27 u = x(:,1) + x(:,2);
28 lfeavg(u, Mesh)
29
30 fprintf('## RadTEvl (with E)')
31 x = Mesh.Coordinates;
32 u0 = x(:,1) + x(:,2);
33 gamma = 2;
34 m = 3;
35 [mufinal, E] = RadTEvl(u0, gamma, Mesh, m);
36 E
```

Listing 10.4: Output for Testcalls for Problem 10.3

```
1  >> test
2  ## getMatLFE
3  M_part =
4
5      0.0916    0.0044         0    0.0086    0.0327
6      0.0044    0.0107    0.0003         0         0
7           0    0.0003    0.0094    0.0044         0
8      0.0086         0    0.0044    0.0484         0
9      0.0327         0         0         0    0.0654
10
11 A_part =
12
13     4.3252   -0.2791         0   -1.0532   -1.2958
14    -0.2791   37.3235  -15.1954         0         0
15          0  -15.1954   37.3235   -0.2791         0
16    -1.0532         0   -0.2791    4.3252         0
17    -1.2958         0         0         0    1.1518
18
19 ## RadTEvl
20 mufinal =
21
22     0.0193
23     0.0136
24     0.0139
25     0.0194
26     0.0269
27     0.0138
28     0.0144
29     0.0297
30
31 ## lfeavg
32 ans =
33
34     0.2370
```

```
## RadTEvl (with E)
E =

    0.2370
    0.1125
    0.0396
    0.0162
```

Problem 10.4 Radau-3 Timestepping (Core problem)

This short problem studies the full discretization of a linear second-order parabolic initial-boundary value problem as discussed in [NPDE, Sect. 6.1.1]. The focus will be on the implementation of higher-order implicit Runge-Kutta timestepping.

We consider the parabolic IBVP with homogeneous Dirichlet boundary conditions

$$\begin{aligned} \dot{u} - \Delta u &= f(\boldsymbol{x},t) \quad \text{in } \Omega \times (0,1), \\ u &= 0 \qquad\quad \text{on } \partial\Omega \times (0,1), \\ u &= 0 \qquad\quad \text{on } \Omega \times \{0\}, \end{aligned} \tag{10.4.1}$$

where the spatial domain Ω is the unit disk

$$\Omega := \{\boldsymbol{x} \in \mathbb{R}^2,\ \|\boldsymbol{x}\| < 1\}.$$

The time-dependent source function is given by

$$f(\boldsymbol{x},t) = \begin{cases} 1 & \text{, if } \left\| \boldsymbol{x} - \frac{1}{2}\begin{pmatrix}\cos \pi t \\ \sin \pi t\end{pmatrix} \right\| < \frac{1}{2}, \\ 0 & \text{elsewhere.} \end{cases}$$

We pursue the method of lines approach, see [NPDE, Sect. 6.1.3]. Spatial discretization relies on linear Lagrangian finite elements on a triangular mesh $\mathcal{M}$, that is, $V_{0,N} = \mathcal{S}^0_{1,0}(\mathcal{M})$, using a polygonal approximation of $\partial\Omega$.

Mesh generation and assembly of the stiffness matrix and the mass matrix using the LehrFEM MATLAB finite element library are demonstrated in [NPDE, Code 6.1.48]. This MATLAB script is available for download from the lecture page (see `Parabolic_LFE.m` on the course page).

When higher order timestepping schemes are desired for the initial value problem arising from the method of lines approach, L(π)-stable implicit Runge-Kutta methods ($\rightarrow$ [NPDE, Def. 6.1.26]) are the methods of choice. One example is the Radau-3 scheme, see [NPDE, Eq. (6.1.63)].

(10.4a) Devise and implement (in MATLAB) a numerical experiment for determining the order of (algebraic) convergence of the Radau-3 method, when applied to the initial value problem for a scalar ODE $\dot{y} = -y$, $y(0) = 1$, on $[0,2]$. Which order do you find?

HINT: [NPDE, Rem. 1.6.22] discussed "measurements" for rates of convergence. Measure the error $\max_k |y(t_k) - y^{(k)}|$ for various timestep sizes $\tau > 0$ (equidistant timesteps).

(10.4b) Design a MATLAB function

```
u = radauevl(mesh,tau)
```

for solving (10.4.1) approximately based on the spatial and temporal discretizations described above. The argument `mesh` passes a simple mesh data structure (without edges, see [NPDE, Sect. 3.5.2]), and `tau` specifies the (fixed) timestep. The function should return the coefficient vector for the solution at final time $T = 1$.

(10.4c) Plot the approximate solution of (10.4.1) at final time $T = 1$ obtained by the method from subproblem (10.4b) using the mesh generated by `Circ_mesh_arc.m` (from the lecture page, use $N = 10$) and the timestep $\tau = 0.02$. Invoke the plotting function `plot_LFE` of the MATLAB finite element library.

HINT: For validation purposes you may compare your plot with that provided on the lecture website.

Listing 10.5: Testcalls for Problem 10.4

```
% Initialize mesh
[Mesh, X] = Circ_mesh_arc(10);
Mesh.ElemFlag = ones(size(Mesh.Elements,1),1);

U = Radauev1(Mesh,0.02);
U(end-10:end)
```

Listing 10.6: Output for Testcalls for Problem 10.4

```
>> test_call

ans =

    0.0229
    0.0654
    0.0286
    0.0328
    0.0852
    0.0446
    0.0515
    0.1004
    0.0641
    0.0691
    0.0761
```

Published on May 6.

To be submitted on May 13.

MATLAB: Submit all files in the online system. Include the files that generate the plots. Label all your plots. Include commands to run your functions. Comment on your results.

References

[NPDE] Lecture Slides for the course “Numerical Methods for Partial Differential Equations”, SVN revision # 54620.

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

[LehrFEM] LehrFEM manual.

Last modified on May 10, 2013