

Homework Problem Sheet 4

Problem 4.1 Establishing empirical convergence rates

In [NPDE, Section 1.6] the concept of “convergence”, and in particular of *algebraic* and *exponential* convergence, has been introduced. Through several practical examples you have learned, how use error norms obtained from numerical experiments to extract the necessary information to describe *qualitatively* and *quantitatively* the convergence of a numerical method. Detailed instructions are given in [NPDE, § 1.6.26]. This problem is meant to make you more familiar with concepts and techniques connected with the notion of “convergence”.

(4.1a) In the supplementary material distributed with this problem sheet you can find the ASCII files `Nvalues.txt`, `Error1.txt`, `Error2.txt` and `Error3.txt`.

The files `Error1.txt`, `Error2.txt` and `Error3.txt` contain sequences of error norms obtained from three different numerical experiments and three different discretization schemes for some boundary value problem. For each case the associated values N of degrees of freedom used for discretization are contained in the file `Nvalues.txt`. Thus pairs of problem sizes N and related norms of the discretization error are available.

For each of the three convergence studies, describe qualitatively and quantitatively the empirical convergence that you observe.

HINT: Valuable information can be extracted from doubly logarithmic (`loglog`) or semi-logarithmic (`semilogy`) plots. In MATLAB, you may use the function `polyfit` to estimate the convergence rates. See [NPDE, § 1.6.26] for details.

HINT: The MATLAB function `lsqcurvefit` that solves nonlinear least squares problems may come into help.

Solution: For the values e_1 from `Error1.txt`, a double logarithmic plot (see left plot in Figure 4.1) shows that we have algebraic convergence; doing a first order polynomial fitting of $\log(N)$ and $\log(e_1)$, we see that the convergence rate is $0.496540 \approx \frac{1}{2}$.

For the values e_2 from `Error2.txt`, a semi-logarithmic (`semilogy`) plot (see right plot in Figure 4.1) shows that the convergence is exponential. A first order polynomial fitting of N and $\log(e_2)$ shows that $e_2 \sim 2^{-N}$.

For the values from `Error3`, we get the semi-logarithmic and double logarithmic plots shown in Figure 4.2 (a) and (b). From the two plots we conjecture that the error behaves as $e_3 \sim e^{-\gamma N^\delta}$; since the error has to decrease as $N \rightarrow +\infty$, we have that either $\gamma > 0, \delta > 0$, either $\gamma < 0, \delta < 0$. Since the error curve is convex in the semi-logarithmic plot and concave in the double logarithmic

plot, we deduce that we are in the case that $\gamma > 0$ and $\delta > 0$. Thus, we have that

$$(e_3)_i = C_1 e^{-\gamma N_i^\delta}, \quad C_1, \gamma, \delta > 0, i = 1, \dots, M, \quad (4.1.1)$$

where N_i denotes the i -th entry of the vector N of the number of degrees of freedom, $(e_3)_i$ the corresponding measured error, and M the length of the array N .

Using the MATLAB function `lsqcurvefit` on $\log((e_3)_i) = \log(C_1) - \gamma N_i^\delta$ (which works better than if we use (4.1.1)), then we get $C_1 \approx 170$, $\gamma \approx 0.5409$ and $\delta \approx 0.5030 \approx \frac{1}{2}$. Thus, in the end we have that $e_3 \approx 170\sqrt{3}^{-N^{\frac{1}{2}}} = 170 \cdot 3^{-\frac{\sqrt{N}}{2}}$ (where we have obtained $\sqrt{3}$ as e^γ).

A double logarithmic plot of N and $\log(C_1) - \log(e_3)$ gives a curve that is very close to a line, see Figure 4.2 (c).

See Listing 4.1 for more details on the procedure we have used.

Listing 4.1: Implementation for tasks (4.3a) and (4.3b).

```

1 N = load('Nvalues.txt', '-ascii');
2 e1 = load('Error1.txt', '-ascii');
3 e2 = load('Error2.txt', '-ascii');
4 e3 = load('Error3.txt', '-ascii');
5
6 figure(1)
7 loglog(N, e1);
8 p = polyfit(log(N), log(e1), 1);
9 sprintf('Algebraic convergence with rate %f', -p(1))
10
11 figure(2)
12 semilogy(N, e2);
13 p = polyfit(N, log(e2), 1);
14 sprintf('Exponential convergence with basis %f and rate %f', p(1), exp(-p(1)))
15
16 figure(3)
17 semilogy(N, e3)
18 figure(4)
19 loglog(N, e3)
20 x0 = [1, 1, 1];
21 x = lsqcurvefit(@(x, N) log(x(1)) - x(2) .* N.^(x(3)), x0, N, log(e3));
22 sprintf('Exponential convergence with basis %f and exponent %f', x(2), x(3))
23 figure(5)
24 loglog(N, log(x(1)) - log(e3))

```

(4.1b) For each of the three convergence studies from subproblem (4.3a), provide a plot of the error norms versus N , for which the measured error norms approximately lie on a straight line.

Solution: See Figure 4.1 and right plot in Figure 4.2.

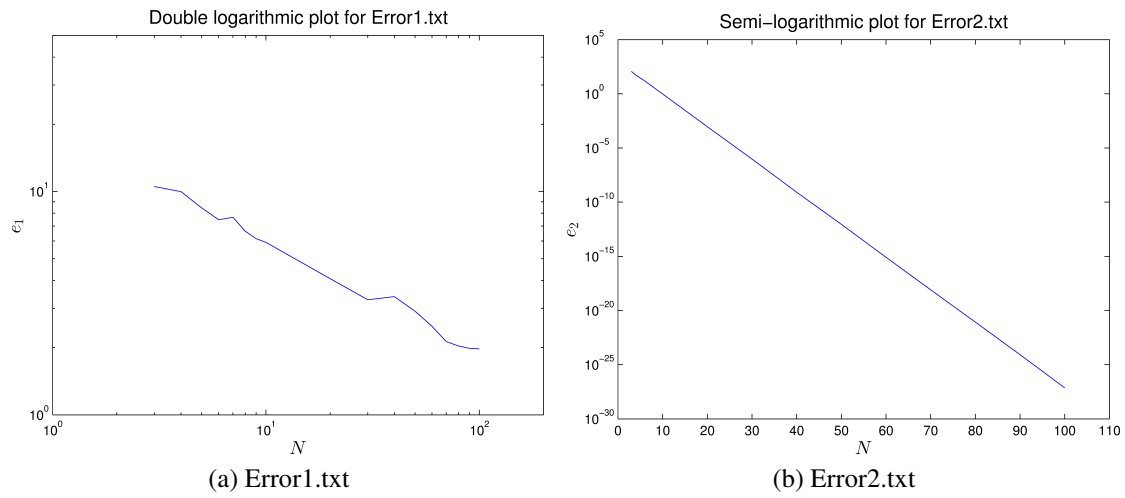


Figure 4.1: Convergence plots for Error1.txt and Error2.txt.

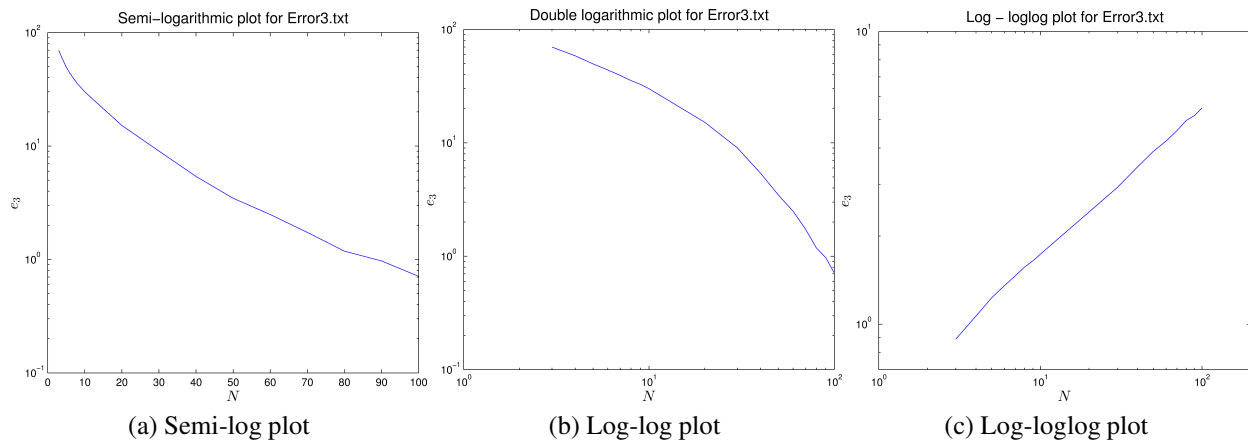


Figure 4.2: Convergence plots for Error3.txt.

Problem 4.2 $L^2(0, 1)$ -Orthogonal Projection onto Linear Finite Element Space

In this problem we deal with a very simple *quadratic* minimization problem that does not even involve derivatives. We derive the associated variational formulation, and then discretize it with linear finite elements as in [NPDE, Section 1.5.2.2]. A careful re-examination of this section is recommended. You will be asked to implement the method in MATLAB and perform a numerical study of its convergence.

Given $f \in C_{\text{pw}}^0([0, 1])$, the minimization problem reads:

$$u^* = \operatorname{argmin}_{v \in C_{\text{pw}}^1([0, 1])} \underbrace{\int_0^1 |v(x) - f(x)|^2 dx}_{:=J(v)} \quad (4.2.1)$$

(4.2a) Show that $J = J(v)$ from (4.2.1) is a quadratic functional and identify its building blocks according to [NPDE, Def. 2.2.24].

Solution: Since

$$J(v) = \int_0^1 |v(x) - f(x)|^2 dx = \int_0^1 (v(x) - f(x))(v(x) - f(x)) dx = \int_0^1 v^2(x) - 2v(x)f(x) + f^2(x) dx,$$

we can write it as

$$J(v) = \frac{1}{2}a(v, v) - \ell(v) + \gamma,$$

with $a(u, v) := 2 \int_0^1 u(x)v(x) dx$, $\ell(v) := 2 \int_0^1 f(x)v(x) dx$ and $\gamma := \int_0^1 f^2(x) dx$.

(4.2b) Derive the variational problem associated with the minimization problem (4.2.1).

HINT: Don't forget to specify the trial and test spaces.

Solution: The variational problem reads: seek $u \in C_{\text{pw}}^1([0, 1])$:

$$\int_0^1 u(x)v(x) dx = \int_0^1 f(x)v(x) dx \quad \forall v \in V := C_{\text{pw}}^1([0, 1]). \quad (4.2.2)$$

Remark: Variational problems of this kind are encountered when computing the $L^2(0, 1)$ -orthogonal projection of f onto subspaces.

(4.2c) Show that the solutions to (4.2.1) are unique, that is, if two solutions are known to be global minimizers of J , then they must agree.

HINT: A theorem from [NPDE, Section 2.2.3] may come handy. If you plan to use it, please give a precise citation.

Solution: A function u is a solution to (4.2.1) if and only if it is a solution to (4.2.2). Suppose now we have two minimizers u_1 and $u_2 \in C_{\text{pw}}^1([0, 1])$; then we have

$$\begin{aligned} \int_0^1 u_1(x)v(x) dx &= \int_0^1 f(x)v(x) dx \\ \int_0^1 u_2(x)v(x) dx &= \int_0^1 f(x)v(x) dx. \end{aligned}$$

for every $v \in V$.

Subtracting the two equations we obtain

$$\int_0^1 (u_1(x) - u_2(x))v(x) dx = 0 \quad \text{for all } v \in V,$$

which, thanks to the fundamental lemma of calculus of variations ([NPDE, Lemma 1.3.33]), implies $u_1 = u_2$.

Alternatively, one can observe that the bilinear form $\tilde{a}(u, v) := \int_0^1 u(x)v(x) dx$, $u, v \in V$, satisfies $a(u, u) = \|u\|_{L^2([0,1])}^2$ and thus it is positive definite. Then Theorem [NPDE, Thm. 2.2.41] implies uniqueness of the solution.

To begin with, we consider Galerkin discretization based on the space $V_N = \mathcal{S}_1^0(\mathcal{M})$ of piecewise linear continuous functions on a uniform mesh \mathcal{M} of $[0, 1]$ with M cells of size $h = \frac{1}{M}$, see [NPDE, Section 1.5.2.2]. In the following, use tent functions according to [NPDE, Eq. (1.5.62)] as a basis for the Galerkin trial and test space.

(4.2d) What is the Galerkin matrix for this problem? Write a MATLAB function

$$A = \text{galmatrix_tent}(M)$$

which takes the number of equal mesh cells as input in M , computes the Galerkin matrix for the variational problem from subproblem (4.2b), and returns it in the *sparse matrix* A .

Solution: The Galerkin matrix should have the format

$$A = \begin{pmatrix} \frac{h_1}{3} & \frac{h_1}{6} & & & & \\ \frac{h_1}{6} & \frac{h_1+h_2}{3} & \frac{h_2}{6} & & & \\ & \frac{h_2}{6} & \frac{h_2+h_3}{3} & \frac{h_3}{6} & & \\ & & \ddots & \ddots & \ddots & \\ & & & \frac{h_{N-1}}{6} & \frac{h_{N-1}+h_N}{3} & \frac{h_N}{6} \\ & & & & \frac{h_N}{6} & \frac{h_N}{3} \end{pmatrix},$$

where h_1, \dots, h_N are the mesh widths.

For equidistant mesh points, this results in

$$A = \begin{pmatrix} \frac{h}{3} & \frac{h}{6} & & & & \\ \frac{h}{6} & \frac{2h}{3} & \frac{h}{6} & & & \\ & \frac{h}{6} & \frac{2h}{3} & \frac{h}{6} & & \\ & & \ddots & \ddots & \ddots & \\ & & & \frac{h}{6} & \frac{2h}{3} & \frac{h}{6} \\ & & & & \frac{h}{6} & \frac{h}{3} \end{pmatrix},$$

where h is the mesh widths.

Listing 4.2: Implementation for `galmatrix_tent`

```
1 function A = galmatrix_tent(mesh)
2 h = mesh(2:end) - mesh(1:end-1);
```

```

3 v1 = h./6;
4 v2 = [h;0]./3 + [0;h]./3;
5
6 A = gallery('tridiag',v1,v2,v1);
7
8 %Modification for zero b.c.
9 %A = A(2:end-1,2:end-1);
10
11 end

```

See [Listing 4.2](#).

(4.2e) Write down the entries of the right-hand side vector for the variational problem from [subproblem \(4.2b\)](#), using the composite trapezoidal rule for numerical quadrature of integral involving the generic function f .

Solution: For equidistant mesh points, the right-hand side is given by

$$\mathbf{l} = \begin{pmatrix} f(0)\frac{h}{2} \\ f(h)h \\ f(2h)h \\ \vdots \\ f((M-1)h)h \\ f(1)\frac{h}{2} \end{pmatrix},$$

where h is the mesh widths.

(4.2f) Write a MATLAB function

$$\mathbf{L} = \text{rhs_tent}(M, f)$$

which takes as input the number M of equal mesh cells and a function handle to f , computes the right-hand side vector, and returns it in the column vector \mathbf{L} .

Solution: See [Listing 4.3](#).

Listing 4.3: Implementation for rhs_tent

```

1 function rhs = rhs_tent(mesh, f)
2
3     fvals = f(mesh);
4     hvals = mesh(2:end) - mesh(1:end-1);
5
6     rhs = ([hvals;0].*fvals + [0;hvals].*fvals)/2;
7
8     %Modification for zero b.c.
9     %rhs = rhs(2:end-1);
10
11 end

```

(4.2g) Write a MATLAB function

$$U = \text{l2proj_tent}(M, f)$$

that solves the variational problem from task (4.2b) approximately based on linear finite element Galerkin discretization on an equidistant mesh with M cells. The arguments M and f are the same as before. The column vector U should contain the value of the solution in each node of the mesh.

Solution: See [Listing 4.4](#).

Listing 4.4: Implementation for `l2proj_tent`

```
1 function U = l2proj_tent(mesh, f)
2
3     A = galmatrix_tent(mesh);
4     L = rhs_tent(mesh, f);
5     U = A \ L;
6
7     % Modification for zero boundary conditions
8     %U = [0; U(:); 0]';
9 end
```

We now want to investigate the convergence for the L^p -norm of the discretization error.

For $u \in C_{pw}^0(I)$ on a closed interval $I \subset \mathbb{R}$ and a real number $1 \leq p \leq \infty$, the L^p -norm of u is defined as

$$\|u\|_{L^p(I)} := \left(\int_I |u(x)|^p dx \right)^{\frac{1}{p}}, \quad \text{for } 1 \leq p < \infty, \quad (4.2.3)$$

and

$$\|u\|_{L^\infty(I)} := \sup_{x \in I} |u(x)|. \quad (4.2.4)$$

(4.2h) Write a MATLAB function

$$\text{function rate} = \text{lpcvg}(\text{sol}, p)$$

that, given $1 \leq p \leq \infty$, performs a convergence study for the L^p -norm of the discretization error and returns in `rate` the (algebraic) convergence rate. Use the convention that $p = 0$ for the L^∞ -norm. The argument `sol` is a function handle to the exact solution. Use the values $N = 10 \cdot 2^i$, $i = 1, \dots, 9$, for the number of mesh intervals.

For the computation of the norms, use the 2-point Gauss quadrature rule, with quadrature points $\zeta_1 = -\frac{\sqrt{3}}{3}$, $\zeta_2 = \frac{\sqrt{3}}{3}$ and weights $\omega_1 = \omega_2 = 1$ on the reference interval $[-1, 1]$.

Solution: See [Listing 4.5](#) for the code.

Listing 4.5: Implementation for `lpcvg`

```
1 function rate = lpcvg(sol, p)
2
3 Nvals = 10*2.^(1:9);
```

```

4 errors = zeros(1,9);
5
6 for i = 1:9
7     mesh = linspace(0,1,Nvals(i)+1)';
8     h = 1/Nvals(i);
9
10    U = l2proj_tent(mesh, sol);
11
12    qpts = sort([mesh(1:end-1) + h/2 - h/(2*sqrt(3));
13               mesh(1:end-1) + h/2 + h/(2*sqrt(3))]);
14    appr = linterp(mesh, U, qpts);
15    ex = sol(qpts);
16
17    if (p~=0)
18        errors(i) = (sum(abs((appr-ex')).^p)*(h/2)).^(1/p);
19    else
20        errors(i) = max(abs(appr(:)-ex(:)));
21    end
22 end
23
24 loglog(Nvals,errors)
25 hold on
26 pol = polyfit(log(Nvals), log(errors), 1);
27 rate=-pol(1);

```

(4.2i) Write a MATLAB script

convergence

to compute

$$\|u - u_N\|_{L^p([0,1])} \quad \text{for } N \rightarrow \infty. \quad (4.2.5)$$

for the values $p = 1.0, 1.5, 2, 4, \infty$. In (4.2.5), u and u_N denote the exact and Galerkin solutions, respectively. Plot the error curves for these values of p in the case that the exact solution is

$$u(x) = \sin(x^2) \text{ and in the case that it is } u(x) = \begin{cases} 1 & x \leq \frac{\sqrt{2}}{2} \\ 0 & x > \frac{\sqrt{2}}{2} \end{cases}.$$

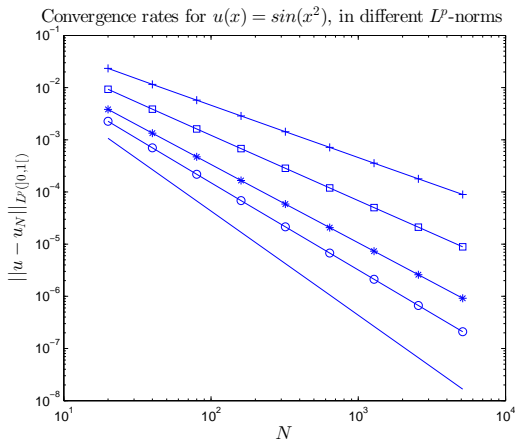
Solution: See Figure [Figure 4.3](#) and code [4.6](#).

Listing 4.6: Implementation for convergence

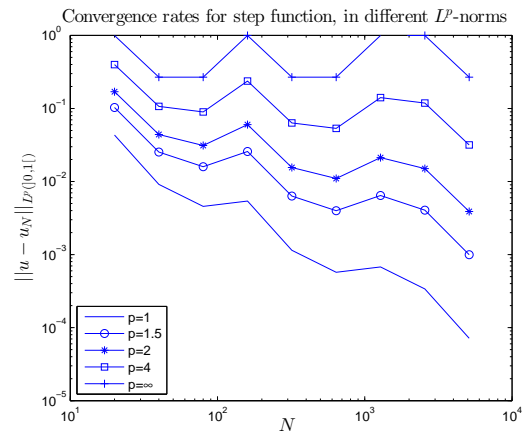
```

1
2 pvals=[1,1.5,2,4,0];
3 rates = zeros(size(pvals));
4 exact = @(x) sin(x.^2);
5 %exact = @(x) (x<sqrt(2)/2);
6
7 for i=1:length(pvals)

```

(a) $\sin(x^2)$



(b) Step function

Figure 4.3: Convergence plots for different L^p -norms (double logarithmic plots), subproblem (4.2i).

```

8     rates(i)=lpcvg(exact,pvals(i));
9 end
10
11 pvals(end) = Inf;
12 figure
13 plot(pvals,rates)
14
15 % p=polyfit(log(pvals(1:end-1)),log(log(rates(1:end-1))),1);
16 % gamma=exp(p(2));
17 % plot(log(pvals),log(log(rates)))
18 % hold on
19 % plot(log(pvals),log(gamma*pvals.^p(1)),'r')
20
21 p = polyfit(log(pvals(1:end-1)),log(rates(1:end-1)),1);
22 figure
23 loglog(pvals,rates)
24 hold on
25 loglog(pvals,p(2).*pvals.^p(1),'r')

```

Listing 4.7: Testcalls for Problem 4.2

```

1 mesh = linspace(0,1,5)';
2 fprintf('\n\n##galmatrix_tent:')
3 full(galmatrix_tent(mesh))
4
5 fprintf('\n\n##rhs_tent:')
6 rhs_tent(mesh, @(x) x)
7
8 fprintf('\n\n##l2proj_tent:')
9 l2proj_tent(mesh, @(x) x)
10

```

```

11 fprintf('\n\n#lpcvg:')
12 rate = lpcvg(@(x) sin(x.^2),2)

```

Listing 4.8: Output for Testcalls for Problem 4.2

```

1 >> test_call_lp
2
3 ##galmatrix_tent:
4 ans =
5
6     0.0833    0.0417         0         0         0
7     0.0417    0.1667    0.0417         0         0
8         0    0.0417    0.1667    0.0417         0
9         0         0    0.0417    0.1667    0.0417
10        0         0         0    0.0417    0.0833
11
12 ##rhs_tent:
13 ans =
14
15         0    0.0625    0.1250    0.1875    0.1250
16
17 ##l2proj_tent:
18 ans =
19
20    -0.1429    0.2857    0.5000    0.7143    1.1429
21
22 ##lpcvg:
23 rate =
24
25     1.5028

```

Problem 4.3 Quadratic and non-quadratic functionals

In [NPDE, Section 1.3] you have seen many examples of energy functionals, and you have learned the connection between the minimization of a functional and the solution of a variational formulation. Moreover, in [NPDE, Section 1.4] and, again, in [NPDE, Section 2.2.3], you have studied *quadratic* minimization problems, which lead to *linear* variational formulations as explained in [NPDE, Section 1.4.1] and [NPDE, Section 2.4.1]. In this problem, we will study the properties of further functionals.

Let $V := \mathcal{C}_{\text{pw}}^1([0, 1])$. We consider the functionals $J_i : V \rightarrow \mathbb{R}$, $i = 1, 2, 3$:

$$J_1(v) := \int_0^1 |v(x)|^2 - v'(x) \, dx, \quad (4.3.1)$$

$$J_2(v) := \int_0^1 v(x)v'(x) + v^2(x) \, dx, \quad (4.3.2)$$

$$J_3(v) := \int_0^1 \cosh(v'(x)) + v(x) \, dx, \quad (4.3.3)$$

$$\cosh(x) := \frac{1}{2}(e^x + e^{-x}).$$

(4.3a) Which J_i , $i = 1, 2, 3$, is a quadratic functional?

For these, identify the associated linear forms and (symmetric) bilinear forms. Which of the latter are symmetric positive definite?

Solution: We remind that a functional $J : V \rightarrow \mathbb{R}$ is quadratic if and only if it can be written as $J(v) = \frac{1}{2}a(u, v) - \ell(v) + \gamma$, $v \in V$, where $a(\cdot, \cdot)$ is a bilinear form, $\ell(\cdot)$ is a linear form and $\gamma \in \mathbb{R}$ (see [NPDE, Def. 1.4.3]).

J_1 is quadratic with

$$a_1(u, v) := 2 \int_0^1 u(x)v(x) dx, \quad \ell_1(v) := \int_0^1 v'(x) dx = v(1) - v(0), \quad \gamma_1 = 0, \quad \text{for all } u, v \in V.$$

$a_1(u, v)$ is positive definite because $a_1(u, u) = 2\|u\|_{L^2([0,1])}^2$, for every $u \in V$, which is always positive and equal to 0 if and only if $u = 0$ (where 0 is the zero function).

J_2 is quadratic with

$$a_2(u, v) := \int_0^1 u'(x)v(x) + u(x)v'(x) + 2u(x)v(x) dx, \quad \ell_2(v) = 0, \quad \gamma_2 = 0, \quad \text{for all } u, v \in V.$$

We have that $a_2(u, u) = u^2(1) - u^2(0) + 2\|u\|_{L^2([0,1])}^2$, for every $u \in V$, which is not positive definite because of the presence of $-u^2(0)$.

J_3 is not quadratic because of the presence of $\cosh(x)$.

(4.3b) Show that J_1 and J_3 are convex.

HINT: Recall the following definition:

A functional $J : V \rightarrow \mathbb{R}$ on an affine space V is called convex if

$$J(\lambda x + (1 - \lambda)y) \leq \lambda J(x) + (1 - \lambda)J(y), \quad \text{for all } \lambda \in [0, 1] \text{ and all } x, y \in V. \quad (4.3.4)$$

HINT: Use that $f(x) : \mathbb{R} \rightarrow \mathbb{R}$, $f(x) = x^2$, and $\cosh : \mathbb{R} \rightarrow \mathbb{R}$ are convex on $V = \mathbb{R}$.

Solution: For J_1 we have:

$$\begin{aligned} J_1(\lambda u + (1 - \lambda)v) &= \int_0^1 (\lambda u(x) + (1 - \lambda)v(x))^2 - \lambda u'(x) - (1 - \lambda)v'(x) dx \leq \\ &\leq \int_0^1 \lambda u^2(x) + (1 - \lambda)v^2(x) - \lambda u'(x) - (1 - \lambda)v'(x) dx = \\ &= \lambda J_1(u) + (1 - \lambda)J_1(v), \end{aligned}$$

for every $\lambda \in [0, 1]$ and every $u, v \in V = C_{pw}^1([0, 1])$. In the second passage, we have used the convexity of $f(x) = x^2$.

For J_3 , using the convexity of $\cosh(x)$, we have:

$$\begin{aligned} J_3(\lambda u + (1 - \lambda)v) &= \int_0^1 \cosh(\lambda u'(x) + (1 - \lambda)v'(x)) + \lambda u(x) + (1 - \lambda)v(x) dx \leq \\ &\leq \int_0^1 \lambda \cosh(u'(x)) + (1 - \lambda) \cosh(v'(x)) + \lambda u(x) + (1 - \lambda)v(x) dx = \\ &= \lambda J_3(u) + (1 - \lambda)J_3(v), \end{aligned}$$

for every $\lambda \in [0, 1]$ and every $u, v \in V = C_{\text{pw}}^1([0, 1])$.

(4.3c) Derive the variational formulations that have to be satisfied by potential minimizers of $J_i, i = 1, 2, 3$.

HINT: For J_3 , remember that $(\cosh(x))' = \sinh(x)$.

Solution: The variational formulation for J_1 reads:

Find $u \in V$ such that

$$2 \int_0^1 u(x)v(x) \, dx = v(1) - v(0) \quad \text{for all } v \in V. \quad (4.3.5)$$

For J_2 , we have:

Find $u \in V$ such that

$$\int_0^1 u'(x)v(x) + u(x)v'(x) + 2u(x)v(x) \, dx = u(1)v(1) - u(0)v(0) + 2 \int_0^1 u(x)v(x) \, dx = 0, \quad \text{for all } v \in V. \quad (4.3.6)$$

Finally, for J_3 , from $\lim_{t \rightarrow 0} \frac{J_3(u+tv) - J_3(u)}{t}$ and using that $(\cosh(x))' = \sinh(x)$:

Find $u \in V$ such that

$$\int_0^1 \sinh(u'(x))v'(x) \, dx = - \int_0^1 v(x) \, dx, \quad \text{for all } v \in V.$$

In all the cases, we have denoted $V := C_{\text{pw}}^1([0, 1])$.

(4.3d) State the 2-point boundary value problems satisfied by solutions of the variational equations from subproblem (4.3c), when $V_0 = C_{\text{pw},0}^1([0, 1])$ is used as trial and test space. In all cases, assume the solution u to be smooth.

Solution: For $v \in V_0$, the variational problem associated to J_1 becomes:

Find $u \in V_0$ such that

$$\int_0^1 u(x)v(x) \, dx = 0 \quad \text{for all } v \in V_0.$$

Then, for the fundamental lemma of calculus of variations (see [NPDE, Lemma 1.3.33]), this leads to the following 2-point boundary value problem:

$$u(x) = 0 \text{ in } (0, 1), \quad u(0) = u(1) = 0.$$

From the variational problem associated to J_2 , using the integration by parts or simply using the boundary conditions, we get

$$u(x) = 0 \text{ in } (0, 1), \quad u(0) = u(1) = 0.$$

Finally, applying integration by parts and using the zero-boundary conditions of the test functions, we get that the 2-point boundary value problem associated to J_3 reads:

$$u''(x) \cosh(u'(x)) = 1 \text{ in } (0, 1), \quad u(0) = u(1) = 0.$$

(4.3e) Show that no minimizer exists for J_1 .

Solution: Let us consider $v_n = nx^{\frac{n^2}{2}}$, $n \in \mathbb{N}$. First of all, it's trivial that $v \in V$. We have that

$$J_1(v_n) = \int_0^1 v_n^2(x) dx - (v_n(1) - v_n(0)) = \frac{n^2}{n^2 + 1} - n.$$

$J_1(v_n) \rightarrow -\infty$ as $n \rightarrow +\infty$, and thus J_1 is unbounded from below and has no minimizer.

(4.3f) Show that no minimizer exists for J_2 .

Solution: This time we present another approach and proceed by contradiction.

We observe that, if u is a minimizer, then it solves (4.3.6), which in turn implies $J_2(u) = 0$. Notice that we can choose u not to be the zero function, because, for instance, $J_2(u) = 0$ for $u(x) = \cos(\frac{\pi}{2}x)$.

Let now $w \in V$ be such that $w(x) = \sqrt{f(x)}u(x)$, with $f = f(x)$ a function that is strictly positive and strictly convex on $(0, 1)$, and such that $f(0) = f(1) = 1$. Take, for example, $f(x) = x^2 - x + 1$ or $f(x) = \sin(\pi x) + 1$.

Then we have:

$$\begin{aligned} J_2(w) &= w^2(1) - w^2(0) + \|w\|_{L^2([0,1])}^2 = f(1)u^2(1) - f(0)u^2(0) + \|\sqrt{f}u\|_{L^2([0,1])}^2 \\ &= u^2(1) - u^2(0) + \|\sqrt{f}u\|_{L^2([0,1])}^2; \end{aligned}$$

since we can choose $u \neq 0$ (where here 0 denotes the zero function), the strict convexity of f implies that

$$\|\sqrt{f}u\|_{L^2([0,1])}^2 = \int_0^1 f(x)u^2(x) dx < \int_0^1 u^2(x) dx.$$

This means that $J_2(w) < J_2(u)$, which contradicts the fact that u is a minimizer.

Alternatively, if we proceed as in the solution of task (4.3f), we can choose $v_n(x) = ne^{n^2x} \in V$, $n \in \mathbb{N}$. It holds that $v_n(1) = ne^{-n^2}$, $v_n(0) = n$ and $\|v_n\|_{L^2([0,1])}^2 = 1 - e^{-n^2}$.

Then $J_2(v_n) = ne^{-n^2} - n + 1 - e^{-n^2} \rightarrow -\infty$ as $n \rightarrow +\infty$.

Published on March 11.

To be submitted on March 18.

References

[NPDE] [Lecture Slides](#) for the course “Numerical Methods for Partial Differential Equations”.SVN revision # 74461.

[1] M. Struwe. Analysis für Informatiker. Lecture notes, ETH Zürich, 2009. <https://moodle-app1.net.ethz.ch/lms/mod/resource/index.php?id=145>.

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

Last modified on March 30, 2015