R. Hiptmair
L. Scarabosio
C. Urzua Torres

Spring Term 2015

Numerical Methods for Partial Differential Equations

ETH Zürich
D-MATH

Homework Problem Sheet 2

Problem 2.1 A Curve with Tension

In [NPDE, Sect. 1.3] we examined the energy minimization problem for an elastic string in an external force field. We faced a minimization problem on a space of curves. A similar problem is investigated in this task in order to practice the derivation of variational equations and of the associated two-point boundary value problems.

Given a curve $\mathbf{u} \in V := (\mathcal{C}^1_{\mathrm{pw}}([0,1]))^2$, its tension energy is proportional to the square of its length:

$$J_{\mathrm{tens}}(\mathbf{u}) := \left(\int_0^1 \|\mathbf{u}'(\xi)\| \,\mathrm{d}\xi \right)^2 . \tag{2.1.1}$$

On the curve acts as an external force field $\mathbf{f} = \mathbf{f}(\xi)$, which results in a potential energy contribution according to

$$J_{\mathrm{f}}(\mathbf{u}) := -\int_0^1 \mathbf{f}(\xi) \cdot \mathbf{u}(\xi) \,\mathrm{d}\xi \tag{2.1.2}$$

The curve seeks to attain a shape such that its total energy $J(\mathbf{u}) = J_{\mathrm{tens}}(\mathbf{u}) + J_{\mathrm{f}}(\mathbf{u})$ becomes minimal. In addition, it is pinned at its ends, that is, we have to deal with the constraints

$$\mathbf{u}(0) = \mathbf{u}_0, \qquad \mathbf{u}(1) = \mathbf{u}_1. \tag{2.1.3}$$

We follow the approach of [NPDE, Sect. 1.3.1] to determine which variational problem needs to be solved to compute the shape of the curve.

(2.1a) Determine the *test space* V_0, i.e. the functional space of all admissible variations $\mathbf{v}$.

HINT: Remember that any direction/variation $\mathbf{v} \in V_0$ must vanish wherever the argument function $\mathbf{u}$ is *fixed*, because the test space has to be a vector space. Look up [NPDE, Def. 1.3.16] again.

(2.1b) Following [NPDE, Sect. 1.3.1], compute the *directional/configurational derivative* of $J_{\mathrm{f}} = J_{\mathrm{f}}(\mathbf{u})$ in direction $\mathbf{v} \in V_0$ at a generic curve $\mathbf{u}$ that is compute

$$\lim_{t \to 0} \frac{J_f(\mathbf{u} + t\mathbf{v}) - J_{\mathrm{f}}(\mathbf{u})}{t} .$$

HINT: Observe that J_{f} is a *linear* functional and that the derivative of a linear mapping is easy to compute.

(2.1c) Compute the *directional derivative* of $J_{\text{tens}} = J_{\text{tens}}(\mathbf{u})$ in direction $\mathbf{v} \in V_0$ at a generic curve shape $\mathbf{u}$, that is, compute

$$\lim_{t \to 0} \frac{J_{\text{tens}}(\mathbf{u} + t\mathbf{v}) - J_{\text{tens}}(\mathbf{u})}{t},$$

for any $\mathbf{v} \in V$.

HINT: It is strongly recommended to study [NPDE, §1.3.3] before tackling this problem. In particular [NPDE, Eq. (1.3.5)] will be useful. The identity $A^2 - B^2 = (A-B)(A+B)$ may also come handily.

(2.1d) Determine, which variational problem needs to be solved to compute the shape of the curve. Explain, why we face a non-linear variational problem.

HINT: Use the results from subproblems (2.1a), (2.1c) and (2.1b).

HINT: Do not forget to specify the trial and test spaces.

(2.1e) Assuming that $\mathbf{u}$ is sufficiently smooth, find a two-point boundary value problem, whose solution provides the shape of the curve. Take the cue from the approach in [NPDE, Sect. 1.3.3].

HINT: Remember to use the boundary conditions for test functions $\mathbf{v} \in V_0$.

Problem 2.2 The Brachistochrone Problem

This task retraces all the essential considerations employed in elastic string modeling in class for a different problem from classical mechanics. The purpose of this problem is to practice all the techniques introduced in [NPDE, Sect. 1.2.2], [NPDE, Sect. 1.2.3], [NPDE, Sect. 1.3.1], and [NPDE, Sect. 1.3.3]. In addition, it involves some MATLAB implementation and calculus drill (which will not do you any harm).

The Brachistochrone Problem is a classical problem of variational calculus, already tackled by Newton and Bernoulli in the 17th century: Given two points $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^2$, such that $a_2 > b_2$, we are looking for a curve $\mathbf{u} = (u_1, u_2)^\top : [0,1] \to \mathbb{R}^2$ connecting $\boldsymbol{a}$ to $\boldsymbol{b}$ so that a ball rolling down the curve $\mathbf{u}$ reaches $\mathbf{b}$ in minimal time.

Following our approach to the modeling of an elastic string in [NPDE, Sect. 1.2.2], we first consider an approximate discrete model in order to arrive at a continuous minimization problem describing the Brachistochrone Problem by a limit process.

The **discrete model** approximates the Brachistochrone curve by a polygon; let $\mathbf{u}^i$ for $i = 0, \dots, N+1$, be points (knots) along the curve, with $\mathbf{u}^0 = \boldsymbol{a}$ and $\mathbf{u}^{N+1} = \boldsymbol{b}$, so that there are N "free" points. To find the speed v of the ball at a point $\mathbf{u}$, we assume that the ball starts at rest and appeal to conservation of total (kinetic and potential) energy. Thus, in non-dimensional form already, we have for its speed

$$v(\mathbf{u})^2 = a_2 - u_2 \quad \Longrightarrow \quad v(\mathbf{u}) = \sqrt{a_2 - u_2};.$$

In the following, we will assume $a_2 = 0$, which can always be achieved by choosing a suitable coordinate system, so that $v(\mathbf{u}) = \sqrt{-u_2}$. Note that this, of course, requires $u_2 < 0$.

Each segment $[\mathbf{u}^i, \mathbf{u}^{i+1}]$ has length $\|\mathbf{u}^i - \mathbf{u}^{i+1}\|$, and we approximate the speed of the ball on this

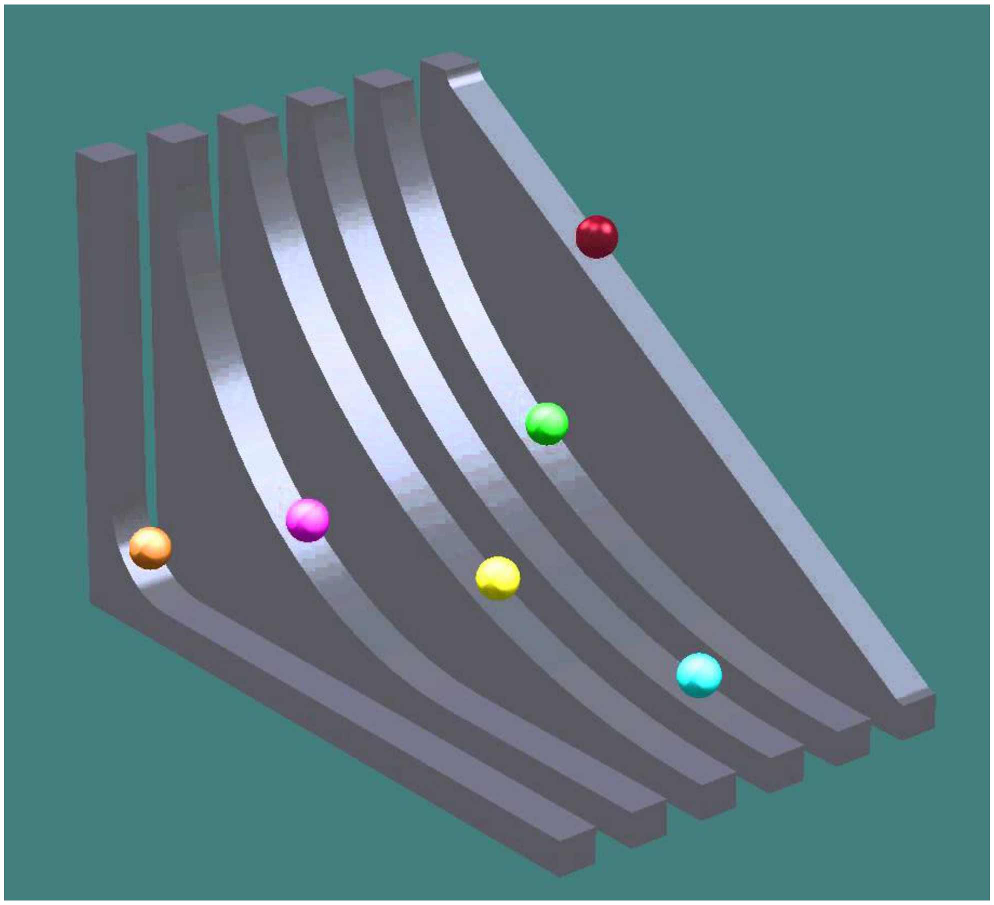

Figure 2.1: Different curves a ball may roll down. On which does it reach the bottom in shortest time?

segment by the constant speed

$$v^i := \sqrt{-\frac{1}{2}(u_2^i + u_2^{i+1})}\ , \quad i = 0, \dots, N\ .$$

This means that the time t_i it takes the ball to cross segment $[\mathbf{u}^i, \mathbf{u}^{i+1}]$ is approximately

$$t_i(\mathbf{u}^i, \mathbf{u}^{i+1}) := \frac{\|\mathbf{u}^{i+1} - \mathbf{u}^i\|}{v^i} = \sqrt{\frac{\|\mathbf{u}^i - \mathbf{u}^{i+1}\|^2}{-\frac{1}{2}(u_2^i + u_2^{i+1})}}\ ,$$

and the total time required for rolling along the curve is

$$T_N(\mathbf{u}^1, \dots, \mathbf{u}^N) = \sum_{i=0}^{N} t_i(\mathbf{u}^i, \mathbf{u}^{i+1})\ .$$

The knot positions of the optimal polygon minimize $T_N(\mathbf{u}^1, \dots, \mathbf{u}^N)$.

(2.2a) We consider the discrete model described above. Write a MATLAB function

```
time = time(u, a, b)
```

that accepts arguments $\boldsymbol{a}$ and $\boldsymbol{b}$ (2×1-vectors) and $\mathbf{u}$ (a $2N \times 1$-vector) containing $\mathbf{u}^1, \dots, \mathbf{u}^N$ stacked on top each other. It should compute and return $T_N(\mathbf{u})$.

HINT: Use the MATLAB function `reshape` to convert `u` to an $N \times 2$-matrix instead, i.e.

```
u = reshape(u, 2, N);
```

Other useful MATLAB functions are `diff` and `sum`.

In order to efficiently minimize the function `time` we will also require the gradient of T_N.

(2.2b) In this and the following sub-problems we examine the derivative of the travel time with respect to the point positions. The obtained expressions will be instrumental in the implementation of an iterative solution strategy.

Show that the gradients of t_i with respect to $\mathbf{u}^i$ and $\mathbf{u}^{i+1}$ are

$$\begin{aligned}\mathbf{grad}_{\mathbf{u}^i}\, t_i &= -\frac{1}{(u_2^i + u_2^{i+1})^2 t_i}\left[2(u_2^i + u_2^{i+1})(\mathbf{u}^i - \mathbf{u}^{i+1}) - \|\mathbf{u}^i - \mathbf{u}^{i+1}\|^2 \begin{pmatrix}0\\1\end{pmatrix}\right]\\ \mathbf{grad}_{\mathbf{u}^{i+1}}\, t_i &= -\frac{1}{(u_2^i + u_2^{i+1})^2 t_i}\left[2(u_2^i + u_2^{i+1})(\mathbf{u}^{i+1} - \mathbf{u}^{i}) - \|\mathbf{u}^i - \mathbf{u}^{i+1}\|^2 \begin{pmatrix}0\\1\end{pmatrix}\right].\end{aligned}$$

HINT: You may use the fact that $\mathbf{grad}_{\mathbf{x}}\, \|\mathbf{x} - \mathbf{y}\|^2 = 2(\mathbf{x} - \mathbf{y})$. Otherwise, this is an exercise in the rules of differentiation (chain rule, quotient rule).

(2.2c) Show that the gradient of T_N regarded as a function $\mathbb{R}^{2N} \mapsto \mathbb{R}$ is the $2N$-vector

$$\mathbf{grad}_{\mathbf{u}}\, T_N(\mathbf{u}) = \begin{pmatrix} \mathbf{grad}_{\mathbf{u}^1}\, t_0 + \mathbf{grad}_{\mathbf{u}^1}\, t_1 \\ \mathbf{grad}_{\mathbf{u}^2}\, t_1 + \mathbf{grad}_{\mathbf{u}^2}\, t_2 \\ \vdots \\ \mathbf{grad}_{\mathbf{u}^N}\, t_{N-1} + \mathbf{grad}_{\mathbf{u}^N}\, t_N \end{pmatrix}.$$

(2.2d) Write a MATLAB function

```
dt = difftime(u, a, b)
```

that accepts the same arguments as the function `time` in sub-problem (2.2a), and returns the gradient as a $2N$-vector.

HINT: Use the MATLAB function `reshape` to convert `u` to an $N \times 2$-matrix instead, i.e.

```
u = reshape(u, 2, N);
```

Other useful MATLAB functions are `diff` and `sum`.

Our policy is to use a successive minimization-refinement algorithm. that is, we want to solve the minimization problem for T_1 using just a single free point between $\boldsymbol{a}$ and $\boldsymbol{b}$. Then, we can divide each curve segment into two by inserting its midpoint as a new knot of the polygon, and solve the minimization problem again for T_3, and so on.

(2.2e) To this end, write a MATLAB function

```
newu = refine(u, a, b)
```

that accepts the same arguments as the function `time` in sub-problem (2.2a) (with $\mathbf{u}$ of length $2N$), and returns the extended $\mathbf{u}$-vector of length $4N+2$. The extra knots are located at the midpoint positions of the sides of the original polygon.

(2.2f) Write a MATLAB script

```
minimize.m
```

that solves the Brachistochrone problem for $\boldsymbol{a} = (0,0)^\top, \boldsymbol{b} = (6,-1)^\top$ using the successive minimization-refinement technique described earlier. Use the midpoint $(3,-0.5)$ as your initial guess for $\mathbf{u}^1$.

To solve the minimization problems in MATLAB, you may use the builtin MATLAB function `fminunc` from MATLAB's optimization toolbox in the following way:

```
options = optimset('GradObj', 'on');
u = fminunc(@minfunc, u, options);
```

Here, `minfunc` must be a function that takes *only* $\mathbf{u}$ as argument (not $\mathbf{a}$ or $\mathbf{b}$), and returns the function value and the gradient. See the provided code template for details.

HINT: If you are tired of `fminunc` writing gibberish to your console, you can provide the options
```
options = optimset('GradObj', 'on', 'Display', 'off');
```

HINT: For more information about `fminunc`, type `help fminunc` in MATLAB.

HINT: You can use the supplied function `plot_curve` to draw your curve.

(2.2g) Now we tackle the continuum limit $N \to \infty$ for the polygon model. Please study again [NPDE, Sect. 1.2.3], where the corresponding considerations are pursued for the mass-spring model of an elastic string.

(2.2h) As in [NPDE, Figure 10] assume that the points $\mathbf{u}^i$ lie on a smooth curve $\mathbf{u} : [0,1] \to \mathbb{R}^2$, i.e. that

$$\mathbf{u}^i = \mathbf{u}(\xi^i) = \mathbf{u}\left(\frac{i}{N+1}\right).$$

Show that in the limit $N \to \infty$, we get

$$T(\mathbf{u}) := \lim_{N\to\infty} T_N(\mathbf{u}^1, \dots, \mathbf{u}^N) = \int_0^1 \frac{\|\mathbf{u}'(\xi)\|}{\sqrt{-u_2(\xi)}}\, \mathrm{d}\xi \,. \tag{2.2.1}$$

HINT: You may use [NPDE, Eq. (1.2.39)] and [NPDE, Eq. (1.3.6)].

(2.2i) Analogous to [NPDE, Sect. 1.3.1] derive the variational problem arising from the minimization of $T(\mathbf{u})$ from 2.2.1 over the space of curves

$$V := \left\{ \mathbf{v} \in (\mathcal{C}^1_{\text{pw}}([0,1]))^2 \,\middle|\, \mathbf{v}(0) = \boldsymbol{a},\ \mathbf{v}(1) = \boldsymbol{b} \right\} .$$

HINT: Use [NPDE, Eq. (1.3.5)] and the Taylor expansion of $x \mapsto x^{-\frac{1}{2}}$:

$$\frac{1}{\sqrt{-x-th}} = \frac{1}{\sqrt{-x}}\left(1 - \tfrac{1}{2}t\frac{h}{x}\right) + \mathcal{O}(t^2) \; .$$

(2.2j) Using integration by parts as in [NPDE, Sect. 1.3.3] derive the differential equation (Euler-Lagrange equation) spawned by the variational problem obtained in sub-problem (2.2i).

(2.2k) Show that the cycloid curve

$$\mathbf{u}(\xi) = \begin{pmatrix} \pi\xi - \sin(\pi\xi) \\ \cos(\pi\xi) - 1 \end{pmatrix} , \quad 0 \le \xi \le 1 \; , \tag{2.2.2}$$

satisfies the differential equation found in sub-problem (2.2j).

(2.2l) Finally, we switch to a graph description of the curve. The considerations run parallel to those of [NPDE, Sect. 1.4.2], which should be read again before starting with this sub-problem.

We re-parameterize the integral (2.2.1) using the variable $x = u_1(\xi)$. For this to work, we assume that the function u_1 maps $[0,1]$ one-to-one and onto $[a_1, b_1]$. (It is sufficient, for example, that $u_1'(\xi) > 0$ everywhere.) Then, there exists an inverse

$$\xi : [a_1, b_1] \to [0,1]$$

so that $u_1(\xi(x)) = x$. Now, define $y(x) = u_2(\xi(x))$. Then $(x, y(x))$ is the *graph* for the curve $\mathbf{u}$ on the x-interval $[a_1, b_1]$.

Show that under this parametrization, (2.2.1) becomes

$$T(y) = \int_{a_1}^{b_1} \sqrt{\frac{1 + y'(x)^2}{-y(x)}} \,\mathrm{d}x \; .$$

HINT: Use the chain rule to express $\partial_x u_2(\xi(x))$ and $\partial_x u_1(\xi(x))$, see [NPDE, Eq. (1.4.21)]. Also apply the transformation formula for integrals ("integration by substitution") [NPDE, Eq. (1.4.22)].

Listing 2.1: Testcalls for Problem 2.2

```
test_time = time([3;-3], [0;0], [6;-1])
test_difftime = difftime([3;-3], [0;0], [6;-1])'
test_extend = extend([3;-3], [0;0], [6;-1])'
```

Listing 2.2: Output for Testcalls for Problem 2.2

```
test_time =

    6.0136

test_difftime =

   -0.0110   -0.0735

test_extend =

   1.5000   -1.5000    3.0000   -3.0000    4.5000   -2.0000
```

Published on February 25.

To be submitted on March 4.

References

[NPDE] Lecture Slides for the course "Numerical Methods for Partial Differential Equations".SVN revision # 73669.

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

Last modified on February 26, 2015