ASSIGNMENT 4
MATINF4170
SPLINE METHODS

Ivar Haugaløkken Stangeby

April 5, 2017

Contents

1 Least square spline approximation

In this assignment, we are to model a heart using cross sectional contour data and least squares approximation. Before we start dealing with the specifics of the heart model, we first recall the least squares approximation problem. Being a generalization of the scalar version, we only go through the parametric least squares approximation, as that is what we will employ in this assignment.

1.1 Parametric curves

Assume we are given a set of data on the form $(u_i, \mathbf{x}_i)_{i=1}^m$, where the $u_i \in \mathbb{R}$ are the *parameter values*, and the $\mathbf{x}_i$ are points in $\mathbb{R}^d$ for some $d \geqslant 1$. Moreover, we fix a spline space $\mathcal{S} \coloneqq \mathbb{S}_{p,\mathbf{t}}$ for some knot vector $\mathbf{t}$ and some non-negative

degree p. We set $n \coloneqq \dim(\mathcal{S})$. We seek the spline function f in the spline space $\mathcal{S}$ that *minimizes* the squared error, i.e.,

$$f = \min_{g \in \mathcal{S}} \sum_{i=1}^{m} w_i (\mathbf{x}_i - g(\mathbf{x}_i))^2, \tag{1}$$

where the w_is are specified weights. This can be reformulated in linear algebra terms as

$$\min_{\mathbf{c} \in \mathbb{R}^n} \|\mathbf{A}\mathbf{c} - \mathbf{b}\|^2, \tag{2}$$

where the matrix $\mathbf{A} \in \mathbb{R}^{m,n}$ is given by $a_{i,j} = \sqrt{w_i} B_j(\mathbf{x}_i)$ and the vector $\mathbf{b} \in \mathbb{R}^m$ is given by $b_i = \sqrt{w_i}\mathbf{x}_i$.

Under the assumption that $m \geqslant n$, we can find the spline coefficients of f by solving the linear system

$$\mathbf{A}^T\mathbf{A}\mathbf{c} = \mathbf{A}^T\mathbf{b}.$$

1.2 Parametric surfaces

Finding the least squares approximation to a parametric surface is completely analogous to the parametric curve case. Here we are instead dealing with tensor product spline spaces $\mathcal{S} \coloneqq \mathcal{S}_1 \otimes \mathcal{S}_2$.[1] The difference then lies in the data set, where we are given two sets of parameter values, i.e., the data set is of the form $(u_i, v_j, \mathbf{x}_{i,j})_{i,j}^{m_1,m_2}$. The linear system to solve is then

$$\mathbf{A}^T\mathbf{A}\mathbf{c}\mathbf{B}^T\mathbf{B} = \mathbf{A}^T\mathbf{G}\mathbf{B},$$

for matrices similar to those in the curve case.

2 Modelling cross sections of a heart.

We are supplied with a collection of nine data sets, each on the form $\mathbf{x}_i = (x_i, y_i, z_i)$ for $i = 1, \ldots, m_j$ where m_j denotes the number of data points for the jth data set. We step through the process for a fixed j.

Parametrization: In order to apply least squares spline approximation to data set j, we first need to find a suitable parametrization of the data. In this assignment we chose the chord length parametrization, in general given as

$$u_i = u_{i-1} + \|\mathbf{x}_i - \mathbf{x}_{i-1}\|^\mu$$

for $i = 2, \ldots, m_j$, where μ is a blending parameter. Setting $\mu = 1$ yields chord length parametrization, and $\mu = 0$ is equivalent to uniform parametrization. After having parametrized the data, we can introduce the augmented data set $(u_i, \mathbf{x}_i)_{i=1}^{m_j}$.

[1] This of course generalizes to the multi-variate case where the tensor product spline space is $\mathcal{S} = \mathcal{S}_1 \otimes \cdots \otimes \mathcal{S}_k$.

Determining the spline space $\mathcal{S}_j$: In order to apply the method outlined in Section 1.1 we still need to fix the spline space $\mathcal{S}_j$. We want cubic splines, so setting $p = 3$, we introduce the $p+1$-regular knot vector $\mathbf{t} \coloneqq (t_1, \ldots, t_{n+p+1})$ where $t_i = t_{p+1} = u_1$ and $t_{n+1} = t_{n+p+1} = u_m$ and the interior knots are uniformly spaced. We can chose the number of basis functions n to be any number between one and m_j, and the choice will certainly have an effect on the resulting spline. In order to be consistent across the data sets we set the number of basis functions to be

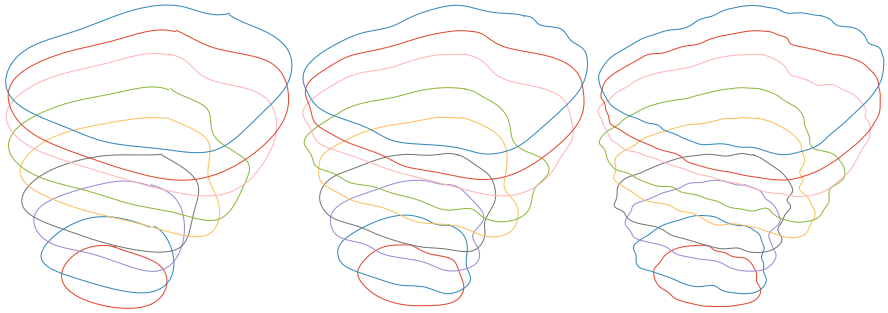
$$n = \lfloor \lambda m_j \rfloor$$

where $\lambda = 5\%, 10\%$, or 20%.

Solving the least square problem: We now have everything we need to solve the least squares approximation problem. We denote for further reference the spline solving the least squares problem on data set j by f_j, where

$$f_j = \min_{g \in \mathcal{S}_j} \sum_{i=1}^{m_j} w_i (\mathbf{x}_i - g(\mathbf{x}_i))^2. \tag{3}$$

The splines have been visualized and can be seen in Figure 1.



(a) $\lambda = 5\%$ (b) $\lambda = 10\%$ (c) $\lambda = 20\%$

Figure 1: The least square spline approximations to the nine contour data sets for varying number of basis splines as indicated by the parameter λ. As we can see, more fine scaled structured is preserved for higher values of λ.

3 Tensor product spline representation of heart

In order to find a least squares tensor product spline surface approximating the data, we first need to construct a data set suitable for approximation. Having already found the splines f_j for the contour curves, we can sample these in

order to create a data set of the form $\mathbf{x}_{i,j} \coloneqq (x_{i,j}, y_{i,j}, z_{i,j})$ for $i = 1, \dots, m$ and $j = 1, \dots, n$ where m and n denote the number of samples in each "direction". From such a data set it is possible to compute the least squares spline surface $\mathbf{f}(u, v)$.

Sampling the spline curves: Having only nine spline curves f_j where the z-component is constant for fixed j, we can only hope to have at most nine samples *across* the curves. This means that $n = 9$. When it comes to sampling *along* the curves, we can choose more freely. In this assignment we let $m = 20$. This choice immediately puts a constraint on the dimension of the spline spaces we are allowed to consider, as we require the number of basis splines in each direction to be no more than the number of data points in each direction.

Assuming now that the jth spline f_j has parameter domain $[0, u_j]$, we sample m uniformly spaced points along the curve. This can be achieved by the following rule:

$$\mathbf{x}_{i,j} \coloneqq f_j(u_j \cdot i/(m-1))$$

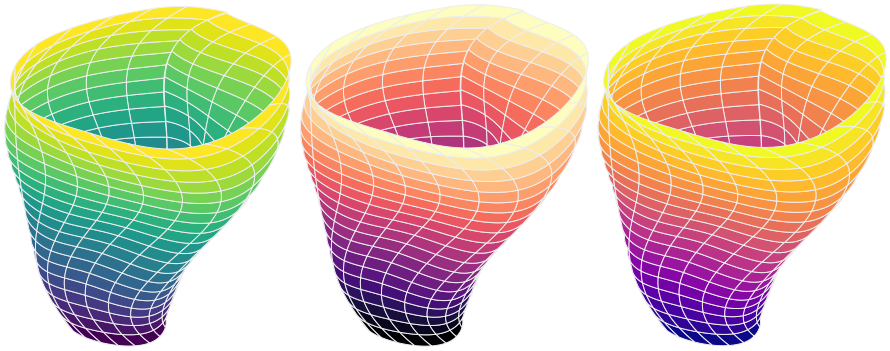
for $i = 0, \dots, m-1$, and then repeating this for each spline $f_1, \dots, f_9$.

Parametrization: Since we sampled the data points $\mathbf{x}_{i,j}$ points uniformly along the curves, it would seem reasonable to also parametrize the rectangular data using uniform parametrization. Setting $\mu = 0$ in the parametrization routine from above, and normalizing the parameters to the interval $[0, 1]$ we obtain the parametrization rule $(u_i, v_j) = (i/19, j/8)$.

Determining the spline space $\mathcal{S}_u \otimes \mathcal{S}_v$: We now have everything we need to choose the knot vectors. In order to obtain non-singular collocation matrices, we need to satisfy the *Schoenberg-Whitney* conditions. This amounts to choosing knots so that the support of each basis spline contains at least one parameter point. We decide on the knot vectors

$$\boldsymbol{\tau}_u \coloneqq (0, 0, 0, 0, \overbrace{\dots}^{7}, 1, 1, 1, 1),$$
$$\boldsymbol{\tau}_v \coloneqq (0, 0, 0, 0, \underbrace{\dots}_{1}, 1, 1, 1, 1),$$

with seven and one interior knots respectively. This ensures that our univariate spline spaces that make up the tensor product spline space has dimensions 11 and 5 respectively. The tensor product spline surfaces have been visualized and can be seen in Figure 2.



(a) $\lambda = 5\%$ (b) $\lambda = 10\%$ (c) $\lambda = 20\%$

Figure 2: The least square approximating surface $\mathbf{f}(u, v)$ to the rectangular heart data set. Due to the rectangular data relying on the least square approximations to the original contour data, we see that the fine scaled structure is lost. It therefore seems that there is no real benefit to choosing a particularity high λ when it comes to modelling purposes.

A Implementation

During this spline course I have been tinkering on a small PYTHON-library, opportunistically named SEAL (SplinE Algorithm Library). It provides high level methods for computing with splines, and I have employed this during this assignment. The library revolves around two objects: `SplineSpace` and `SplineFunction`. SplineSpaces can be instantiated with a spline degree `p` and a knot vector `t`. An example of least squares approximation of data of the same form as the contour data from above is given following:

I) We first load and parametrize the data:

```
import SEAL
import numpy as np

curve = np.loadtxt('hj1.dat')
u = SEAL.parametrize(curve, data_type='curve')
```

II) We can now define our desired spline space, by specifiying a degree, knot end points, and the desired dimension:

```
p = 2
n = 30
t = SEAL.create_knots(u[0], u[-1], p, n)
S = SEAL.SplineSpace(p, t)
```

III) We can now find the least squares approximation in the spline space `S`, and evaluate it over the knot vector `t`.

```
f = SEAL.least_squares_spline_approximation(u, curve, S)
t_values = S.parameter_values(resolution=100)
f_values = f(t_values)
```

IV) The plotting of the spline, its control polygon, and the initial data can now easily be done:

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure()
axs = Axes3D(fig)
axs.plot(*zip(*f_values))
axs.plot(*zip(*f.control_polygon))
axs.scatter(*zip(*f.control_polygon))
axs.scatter(*zip(*curve))
plt.show()
```

The result is shown in Figure 3. The workflow is completely analogous for tensor product surfaces, and methods for computing interpolants and variation diminishing spline approximation are also implemented, both for scalar and parametric splines.

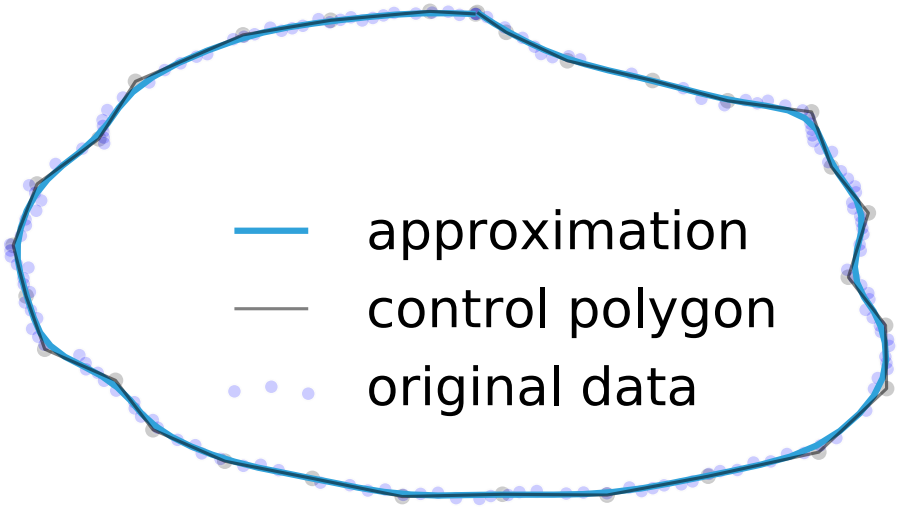


Figure 3: The least square approximation to the first contour data set. Here visualized with the control polygon and the original data points.