ASSIGNMENT 3
MATINF4170
SPLINE METHODS

Ivar Haugaløkken Stangeby

March 17, 2017

Exercise 3.1

In this exercise we wish to show the linear independence of two cubic B-splines on the knot vector $\hat{t} := (0, 0, 1, 3, 4, 5)$ over the interval $[1, 3)$. Using the knot dependencies, these two B-splines are

$$\hat{B}_{1,3}(x \mid 0, 0, 1, 3, 4), \qquad \hat{B}_{2,3}(x \mid 0, 1, 3, 4, 5).$$

In order to show the linear independence, we wish to apply Theorem 3.9, restated here:

Theorem (3.9). Suppose that t is a $p+1$-extended knot vector. Then the B-splines in $\mathbb{S}_{p,t}$ are linearly independent on the interval $[t_{p+1}, t_{n+1})$.

Note that the vector $\hat{t}$ is *not* $p+1$-extended, so we introduce the new knot vector $t := (0, 0, 0, 1, 3, 4, 5, 5)$. We now have enough knots to define four cubic B-splines, and these are

$$B_{1,3}(x \mid 0, 0, 0, 1, 3), \qquad B_{2,3}(x \mid 0, 0, 1, 3, 4),$$
$$B_{3,3}(x \mid 0, 1, 3, 4, 5), \qquad B_{4,3}(x \mid 1, 3, 4, 5, 5).$$

By the above theorem, we now know that the B-splines $B_{i,3,t}$ for $i = 1, \ldots, 4$ are linearly independent on the interval $[t_4, t_5) = [1, 3)$. By examining the knot dependencies, we see that $B_{2,3} = \hat{B}_{1,3}$ and $B_{3,3} = \hat{B}_{2,3}$. Hence, the two B-splines on the coarse knot vector are linearly independent on the interval $[1, 3)$.

Exercise 4.5

In this exercise we wish to prove Lemma 4.2 in generality:

Lemma (4.2). Let p be a positive integer and let τ be a knot vector with at least $p+2$ knots. If t is a knot vector containing τ as a subsequence, then $\mathbb{S}_{p,\tau} \subseteq \mathbb{S}_{p,t}$.

The requirement that τ contains at least $p+2$ knots ensures that we have at least one B-spline in the coarse spline space. We want to apply Curry-Schoenberg, however our knot vectors are not $p+1$-regular. Setting $a := \min(t_1, \tau_1) - 1$ and $b := \max(t_n, \tau_m) + 1$ we can introduce the augmented knot vectors

$$t' := (\overbrace{a, \dots, a}^{p+1}, t_1, \dots, t_n\ , \overbrace{b, \dots, b}^{p+1}),$$
$$\tau' := (\underbrace{a, \dots, a}_{p+1}, \tau_1, \dots, \tau_m, \underbrace{b, \dots, b}_{p+1}).$$

By Curry-Schoenberg, we have the following inclusions:

$$\mathbb{S}_{p,\tau} \subseteq \mathbb{S}_{p,\tau'} = \mathbb{S}_p^{r_{\tau'}}(\Delta_{\tau'}) \subseteq \mathbb{S}_p^{r_{t'}}(\Delta_{t'}) = \mathbb{S}_{p,t'} \supseteq \mathbb{S}_{p,t},$$

where $r_{\tau'}$ and $r_{t'}$ are the continuity requirements imposed by τ' and t' respectively. This implies that any $f \in \mathbb{S}_{p,\tau}$ also lies in $\mathbb{S}_{p,t'}$. It now remains to show that f also must lie in $\mathbb{S}_{p,t}$.

By construction, the first $p+1$, and the last $p+1$ spline coefficients of a spline f in $\mathbb{S}_{p,\tau}$ with respect to the knot vector τ' are zero. Consequently, the first $p+1$ and last $p+1$ spline coefficients of f with respect to the knot vector t' are zero. This means, that only the B-splines originally defined on the refined knot vector t contributes to the spline f. Hence, $f \in \mathbb{S}_{p,t}$.

Exercise 4.6

In this exercise we want to show that Theorem 4.6 holds in the general case, where τ and t not necessarily are $p+1$ regular. For this, we recall the following:

Recall. These results will be needed:

i) $\mathbf{R}_1(x_1) \cdots \mathbf{R}_p(x_p)\boldsymbol{\rho}_p(y) = (y - x_1) \cdots (y - x_p)$.

ii) If $\tau \subseteq t$ are two $p+1$ regular knot vectors with common end knots, then

$$\rho_{i,p,t}(y) = \boldsymbol{\alpha}_p(i)\boldsymbol{\rho}_{p,\tau}(y) \qquad\qquad b_i = \boldsymbol{\alpha}_p(i)c_p.$$

iii) The dual polynomials in $\boldsymbol{\rho}_{p,\tau}(y)$ are linearly independent.

Let τ' and t' be the augmented knot vectors as defined in Exercise 4.5. For $p = 0$, if $\tau'_\mu \leqslant t'_i < \tau'_{\mu+1}$, then $\boldsymbol{\alpha}_p(i) = 1$. If $p > 0$, then we know that

$$\mathbf{R}_1(x_1) \cdots \mathbf{R}_p(x_p)\boldsymbol{\rho}_{p,\tau'}(y) = (y - x_1) \cdots (y - x_p)$$

and letting $x_j = t'_{i+j}$ yields

$$\mathbf{R}_1(t'_{i+1}) \cdots \mathbf{R}_p(t'_{i+p})\boldsymbol{\rho}_{p,\tau'}(y) = (y - t'_{j+1}) \cdots (y - t'_{j+p}) = \rho_{i,p,t'}(y).$$

As we recalled, we can write $\rho_{i,p,t'}(y) = \boldsymbol{\alpha}_p(i)\boldsymbol{\rho}_{p,\tau}$, so this reduces to

$$\mathbf{R}_1(t'_{i+1}) \cdots \mathbf{R}_p(t'_{i+p})\boldsymbol{\rho}_{p,\tau'}(y) = \boldsymbol{\alpha}_p(i)\boldsymbol{\rho}_{p,\tau'}(y).$$

By the linear independence of the dual polynomials, this means that

$$\mathbf{R}_1(t'_{i+1}) \cdots \mathbf{R}_p(t'_{i+p})) = \boldsymbol{\alpha}_p(i).$$

This has proven the result for the $p+1$ regular vectors τ' and t'. Assume we have n B-splines in $\mathbb{S}_{p,\tau}$ and m B-splines in $\mathbb{S}_{p,t}$. In our augmented knot vectors, as the first $p+1$ and last $p+1$ knots are equal, the first $p+1$ and last $p+1$ rows of the knot insertion matrix is zero. For each non-zero row, that is rows $i = p+2, \ldots, m+p+1$, only columns $j = p+2, \ldots, n+p+1$ correspond to the old coarse B-splines. Hence, we can consider the submatrix obtained by dropping the $p+1$ first and last rows and columns.

Exercise 4.7

In this exercise, we show that the $p+1$ *discrete B-splines* of degree p sums to one, provided that τ and t are $p+1$ regular and $\tau \subseteq t$ with knots agreeing at the end.

Recall. The discrete B-splines satisfy the following recurrence relation:

$$\alpha_{j,p}(i) = \frac{t_{i+p} - \tau_j}{\tau_{j+p} - \tau_j}\alpha_{j,p-1}(i) + \frac{\tau_{j+p+1} - t_{i+p}}{\tau_{j+1+p} - \tau_{j+1}}\alpha_{j+1,p-1}(i),$$

where $\alpha_{j,0}(i) = B_{j,0}(t_i)$.

We proceed by induction.

Base case: The base case consists of $p = 0$ where we have

$$\sum_j \alpha_{j,0}(i) = \sum_j B_{j,0}(i) = 1,$$

as the B-splines sum to 1.

Induction step: We assume for the sake of induction that the claim holds for degree $p-1$, and wish to show that it also holds for degree p. We know that $\alpha_{j,p}(i) = 0$ for $j < \mu - p$ and $j > \mu$, so we can discard the discrete B-splines not contributing to the sum. We then have that

$$\sum_{j=\mu-p}^{\mu} \alpha_{j,p}(i) = \sum_{j=\mu-p}^{\mu} \frac{t_{i+p} - \tau_j}{\tau_{j+p} - \tau_j}\alpha_{j,p-1}(i) + \sum_{j=\mu-p}^{\mu} \frac{\tau_{j+p+1} - t_{i+p}}{\tau_{j+1+p} - \tau_{j+1}}\alpha_{j+1,p-1}(i),$$

by the recurrence relation. Now, since we are working with discrete B-splines of one less degree, we have that for $j < \mu - p + 1$ and $j > \mu$ the discrete B-splines $\alpha_{j,p-1}(i) = 0$. We can therefore discard $\alpha_{\mu-p,p-1}(i)$ and $\alpha_{\mu+1,p-1}(i)$ as they are both zero. By a change of summation index, we end up with the following:

$$\sum_{j=\mu-p}^{\mu} \alpha_{j,p}(i) = \sum_{j=\mu-p+1}^{\mu} \frac{\tau_{j+p} - \tau_j}{\tau_{j+p} - \tau_j} \alpha_{j,p-1}(i) = 1,$$

where the last equality follows by the induction hypothesis. This closes the induction.

Exercise 4.8

In this exercise, we implement the *Oslo algorithm* for computing knot insertion matrices and the refinement of coefficient vectors. The Oslo algorithms have a striking resemblance to the spline evaluation algorithms, where the spline is "evaluated" at the fine knots.

We first implement a routine for inserting midpoints in the non-trivial knot intervals in a given $p+1$-extended knot vector for a given spline degree p. A PYTHON implementation is given in Listing 1. This method will be used for repeated refinement of a knot vector, to illustrate that the control polygon of a spline converges under repeated knot insertion.

The Oslo algorithm for computing the coefficients of a coarse B-spline as a linear combination of finer B-splines is implemented in PYTHON as shown in Listing 2.

The algorithm was used to examine empirically the convergence of the control polygon to the spline under repeated knot insertion. Here we simply insert midpoints where possible. The results are given in Figures 1 and 2.

Exercise 4.10

In this exercise, we find the trivariate blossoms of some given polynomials.

Recall. The p-variate *blossom* of $g \in \Pi_p$ satisfies:

i) $\mathcal{B}[g](x_1, \dots, x_p) = \mathcal{B}[g](x_{\sigma(1)}, \dots, x_{\sigma(p)})$ where $\sigma\colon \{1, \dots, p\} \to \{1, \dots, p\}$ is a permutation;

ii) $\mathcal{B}[g](\dots, \alpha x + \beta y, \dots) = \alpha\mathcal{B}[g](\dots, x, \dots) + \beta\mathcal{B}[g](\dots, y, \dots)$ when $\alpha + \beta = 1$;

iii) $\mathcal{B}[g](x, \dots, x) = g(x)$.

It then follows that the blossoms of the following polynomials are:

a) $\mathcal{B}[x^3] = x_1x_2x_3$;

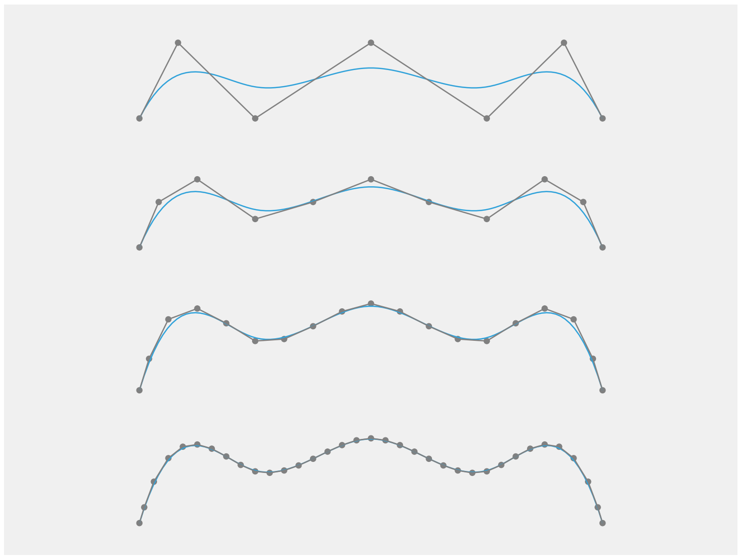

Figure 1: A cubic spline over the knot vector $t = [0, 0, 0, 0, 1, 2, 3, 4, 4, 4, 4]$ with corresponding spline coefficients $c = [-1, 1, -1, 1, -1, 1, -1]$. The top image is the control polygon with no refinements of the knot vector. Then, the three following splines are with one, two and three midpoint refinements respectively.

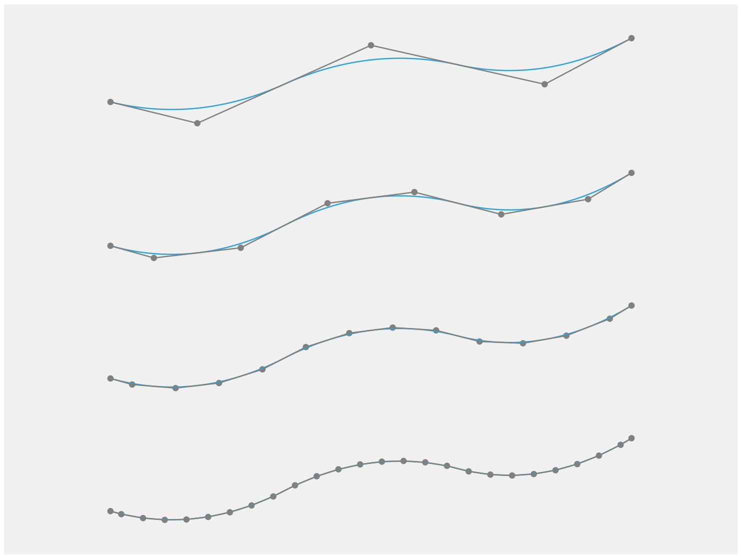

Figure 2: A quadratic spline over the knot vector $t = [0, 0, 0, 1, 2, 3, 3, 3]$ with corresponding spline coefficients $c = [-3, -6, 5, -0.5, 6]$. The top image is the control polygon with no refinements of the knot vector. Then, the three following splines are with one, two and three midpoint refinements respectively.

```
def insert_midpoints(x, p):
    """
    Routine for inserting n - 1 midpoints in a p+1
    extended knot vector, using numpy vector operations.
    :param x: the knot vector x
    :param p: the spline degree p
    :return: knot vector with inserted midpoints
    """
    midpoints = (x[p:-p - 1] + x[p + 1:-p]) / 2
    m = len(midpoints)
    new_array = np.zeros(len(x) + m, dtype=np.float64)

    new_array[:p + 1] = x[:p + 1]
    new_array[-p - 1:] = x[-p - 1]
    new_array[p + 1:p + 2 * m:2] = midpoints
    new_array[p + 2:p + 2 * m - 1:2] = x[p + 1:-p - 1]

    return new_array
```

Listing 1: Inserts midpoints in the non-trivial intervals of a $p + 1$ extended knot vector. The interior knots are supposed to be strictly increasing.

b) $\mathcal{B}[1] = 1$;

c) $\mathcal{B}[2x + x^2 - 4x^3] = 2(x_1 + x_2 + x_3)/3 + (x_1x_2 + x_1x_3 + x_2x_3)/3 - 4x_1x_2x_3$;

d) $\mathcal{B}[0] = 0$;

e) $\mathcal{B}[(x - a)^2] = (x_1x_2 + x_1x_3 + x_2x_3)/3 - 2a(x_1 + x_2 + x_3)/3 + a^2$.

```python
def algorithm_4_10(p, tau, t, c):
    """
    Computes the spline coefficients representing a coarse
    spline in a finer spline space.
    :param p: The spline degree
    :param tau: The coarse knot vector
    :param t: The fine knot vector
    :param c: The set of coarse spline coefficients
    :return: The set of fine spline coefficients
    """

    m = len(t) - (p + 1)
    n = len(tau) - (p + 1)
    c = np.array(c, dtype=np.float64)
    t = np.array(t, dtype=np.float64)
    tau = np.array(tau, dtype=np.float64)
    b = np.zeros(m)

    for i in range(m):
        mu = index(t[i], tau)
        if p == 0:
            b[i] = c[mu]
        else:
            C = c[mu - p:mu + 1]
            for j in range(0, p):
                k = p - j
                tau1 = tau[mu - k + 1:mu + 1]
                tau2 = tau[mu + 1:mu + k + 1]
                omega = np.divide(
                    (t[i + k] - tau1), (tau2 - tau1),
                    out=np.zeros_like(tau1),
                    where=((tau2 - tau1) != 0))
                C = (1 - omega) * C[:-1] + omega * C[1:]
            b[i] = C
    return b
```

Listing 2: The Oslo algorithm implemented in PYTHON. Given a coarse knot vector, we can refine the corresponding spline space and represent the coarse splines as linear combinations of the finer B-splines.