CAGD - Homework 4

Josefine Stål & Erik Ackzell

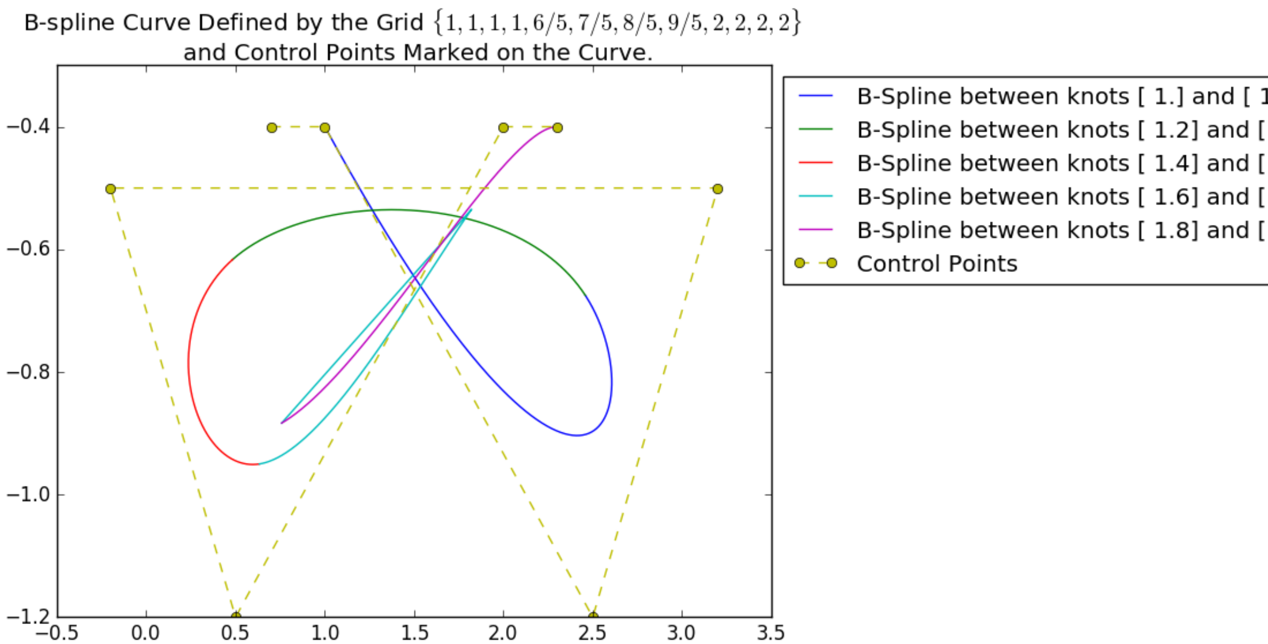
October 16, 2016

Task 1

The B-Spline algorithm can be found in Appendix I.

Task 2

The plot shows the B-Spline curved constructed by the grid $\{1, 1, 1, 1, 6/5, 7/5, 8/5, 9/5, 2, 2, 2, 2\}$ and the control points $(0.7, -0.4)$, $(1, -0.4)$, $(2.5, -1.2)$, $(3.2, -0.5)$, $(-0.2, -0.5)$, $(0.5, -1.2)$, $(2, -0.4)$ and $(2.3, -0.4)$.



Task 3

In this task we want to derive a relation between the number of control points $n+1$ of a clamped B-spline curve of degree p and with simple knots, and the total number of control points of the curve's Bézier segments. We first determine the number of Bézier segments expressed in n and then find the number of control points.

Number of Bézier segments

In the case of only simple knots, including the endpoints, we know that the number of subintervals are given by

$$\#\text{subintervals} = \#\text{knots} - 1. \tag{1}$$

From (1), it is evident that in the present case, the number of Bézier segments is given by

$$\#\text{segments} = \#\text{knots} - 2p - 1. \tag{2}$$

Furthermore, we know that

$$\#\text{knots} - 1 = \#\text{controlpoints} + \text{degree},$$

or equivalently

$$\#\text{knots} - 1 = n + 1 + p. \tag{3}$$

Using (2) and (3), we have that

$$\#\text{segments} = n + 1 - p. \tag{4}$$

Number of control points

Every Bézier segments has the same degree p as the original curve, resulting in every segment needing $p+1$ control points. As the inner control points are each used by two Bézier segments, one of the segment has $p+1$ unique control points, while the others have p unique control points each. Using (4), we have that the total number of control points are given by

$$\begin{aligned} \#\text{total} &= p + 1 + p(n + 1 - 1) \\ &= p(n + 1 - p) + 1. \end{aligned}$$

Comparison with control points of B-spline

In the plot below, the relation of $n+1$ and the total number of control points of the Bézier segments can be seen. From (4) we have that $n+1 > p$, yielding the somewhat strange appearance of the plot. We see that the number of total number of control points of the Bézier segments is larger than the number of control points of the corresponding B-splines, except for the case $p = 1$, where they are equal.

Task 4

Task 5

The curve constructed by the knots

$$\{0, 1/11, 2/11, 3/11, 4/11, 5/11, 6/11, 7/11, 8/11, 9/11, 10/11, 1\}$$

with control points $(0, 0)$, $(3, 2)$, $(9, -2)$, $(7, -5)$, $(1, -3)$, $(1, -1)$, $(3, 1)$, $(9, -1)$ is shown in the first figure. The following figures show how the curve changes if we replace the last knot by the first one followed by the second last one with the second one and so on.

What we can se from the pictures is that..

Task 6

In this task we want to give a rational parametric representation of the segment of the unit circle in $\mathbb{R}^2$ for $x, y \leq 0$.
A natural parametrization of the circle is simply

$$\varphi(t) = \begin{pmatrix} \sin(t) \\ \cos(t) \end{pmatrix}.$$

We recall the Taylor expansions of sine and cosine

$$\begin{aligned} \sin(t) &= \sum_{k=0}^{\infty} (-1)^k \frac{t^{2k+1}}{(2k+1)!} \\ \cos(t) &= \sum_{k=0}^{\infty} (-1)^k \frac{t^{2k}}{(2k)!} \end{aligned} \tag{5}$$

$\forall t \in \mathbb{R}$. It is thus not possible to express the points on the circle as polynomials of finite degrees.
Consider the lines passing through the point (1, 0) and the segment of the unit circle with $x, y \leq 0$. All such lines are on the form

$$y = t(x-1) \quad t \in [0, 1] \quad x \in [-1, 0]. \tag{6}$$

All points on the circle segment fulfills

$$y^2 + x^2 = 1. \tag{7}$$

Now substitute y in (7) with y in (6). This yields

$$x^2 + t^2(x-1)^2 = 1. \tag{8}$$

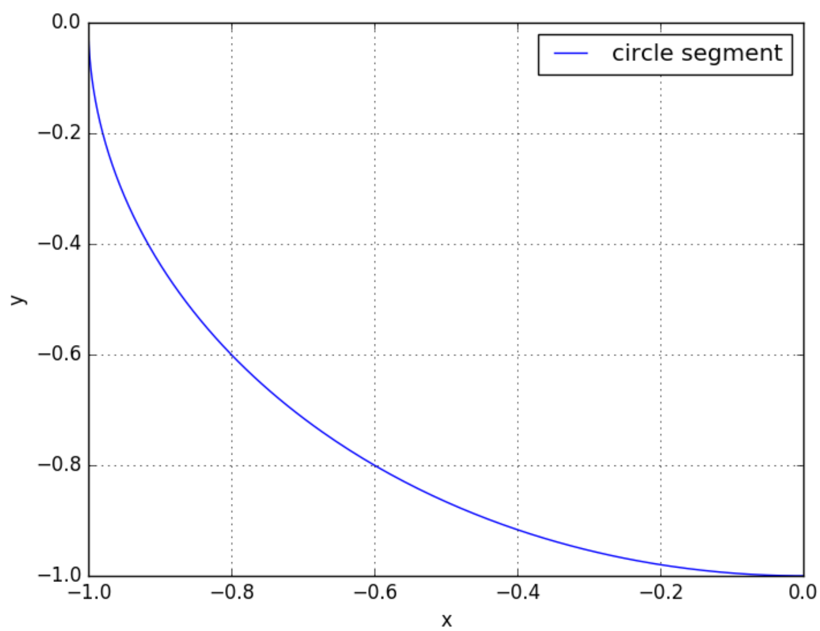
The solutions to (8) are given by $x = 1$ and $x = \frac{t^2-1}{t^2+1}$ and since we need $x \in [-1, 0]$ only the second is suitable. Inserting this expression for x into (7) yields

$$y^2 + \left(\frac{t^2-1}{t^2+1}\right)^2 = 1. \tag{9}$$

The solutions to (9) are given by $y = \frac{2t}{t^2+1}$ and $y = -\frac{2t}{t^2+1}$ and we are only interested in the second solution as $y \in [-1, 0]$.
Thus, the circle segment can be parametrized by

$$\varphi(t) = \frac{1}{t^2+1}\begin{pmatrix} t^2-1 \\ -2t \end{pmatrix}.$$

A plot of the circle segment using the above parametrization can be seen below.



Appendix I

```
import scipy
from matplotlib import pyplot as plt
import numpy as np


class Bspline(object):
    def __init__(self, grid, controlpoints, degree):
        """
        grid (iterable): grid points should have multiplicity 3 in order
            to
        have the spline starting and ending in the first and last control
        point, respectively.
        controlpoints (array): should be on the form
                controlpoints = array([
                                    [d_00, d01],
                                    [d_10, d11],
                                    ...
                                    [d_L0, d_L1],
                                    ]),
        i.e. an (L+1)x2 array.
        """
        try:
            grid = scipy.array(grid)
            grid = grid.reshape((len(grid), 1))
        except ValueError:
            raise ValueError('Grid should be a one-dimensional list or
                array')
        if controlpoints.shape[1] != 2:
            raise ValueError('Controlpoints should be an (L+1)x2 array.')
        self.grid = grid.reshape((len(grid), 1))
        self.controlpoints = controlpoints
        self.degree = degree
        self.dim = scipy.shape(controlpoints)[1]

    def __call__(self, u):
        index = self.get_index(u)
        r = self.get_mult(index, u)
        current_controlpoints = self.get_controlpoints(index, r, u)
        numControlpoints = len(current_controlpoints)
        d = scipy.zeros((numControlpoints, numControlpoints, self.dim))
        d[0] = current_controlpoints
        for s in range(1, numControlpoints): # column
            for j in range(s, numControlpoints): # rows
                left = index - self.degree + j
                right = index + j - s + 1
                a = (u - self.grid[left])\
                    / (self.grid[right] - self.grid[left])
```

```
            d[s,j] = (1-a)*d[s-1,j-1] + a*d[s-1,j]
        return d[-1, -1]

    def get_mult(self, index, u):
        if u == self.grid[index]:
            return len([i for i in self.grid if i == self.grid[index]])
        elif u == self.grid[-1]: # IS THERE A NICER WAY OF FIXING THIS?
            return len([i for i in self.grid if i == self.grid[-1]])
        else:
            return 0

    def get_controlpoints(self, index, r, u):
        """
        Method to obtain the current control points, d_{i-n}, ..., d_i,
            for de Boor's algorithm, where n is the degree
        and i is the index of the interval for which u lies in:
            [u_i,u_{i+1}). If u=u_i and u_i has multiplicity r the
        current control points changes to be
            d_{i-n},...,d_{i-r-1},d_{i-r}.
        index (int): the index depending on the point u at which to
            evaluate
        the spline (see get_index method).
        """
        # Assert an error if the degree is bigger than the index, which
            only happens if the curve is not clamped
        assert (index - self.degree) >= 0, 'The curve is not clamped.'
        if r > self.degree:
            # We are only working with clamped curves, therefore there
                are always have n+1 knots at the endpoints
            # A knot can not have multiciply higher than the degree
                unless it is at the end points
            if u == self.grid[0]:
                # startpoint
                current_controlpoints = self.controlpoints[:self.degree +
                    1]
            else:
                # endpoint
                current_controlpoints = self.controlpoints[-(self.degree
                    + 1):]
        elif r == 1: # CAN THIS BE DONE IN A NICER WAY?
            current_controlpoints = self.controlpoints[index -
                self.degree:index]
        else:
            current_controlpoints = self.controlpoints[index -
                self.degree:index - r]
        return current_controlpoints


    def get_index(self, u):
        """
```

```
        Method to get the index of the grid point at the left endpoint
            of the
        gridpoint interval at which the current value u is. If u belongs
            to
            [u_I, u_{I+1}]
        it returns the index I.
        u (float): value at which to evaluate the spline
        """
        if u == self.grid[-1]: # check if u equals last knot
            index = (self.grid < u).argmin() - 1
        else:
            index = (self.grid > u).argmax() - 1
        return index

    def plot(self, title, filename=None, points=300, controlpoints=True,
         markSeq=False, clamped=False):
        """
        Method to plot the spline.
        points (int): number of points to use when plotting the spline
        controlpoints (bool): if True, plots the controlpoints as well
        markSeq (bool): if true, marks each spline sequence in the plot
        clamped (bool): if true, skips the multiples in the endpoints
            when each sequence is marked
        """
        fig = plt.figure()
        ax = fig.add_subplot(111)
        if markSeq:
            if clamped:
                start, end = self.degree, len(self.grid) - self.degree - 1
            else:
                start, end = 0, len(self.grid) - 1
            for i in range(start, end):
                ulist = scipy.linspace(self.grid[i], self.grid[i+1],
                    points)
                ax.plot(*zip(*[self(u) for u in ulist]), label='B-Spline
                    between knots {} and {}'.format(self.grid[i],
                                                                                    self.g
                                                                                    ))

        else:
            # list of u values for which to plot
            ulist = scipy.linspace(self.grid[0], self.grid[-1], points)
            ax.plot(*zip(*[self(u) for u in ulist]), label='B-Spline
                Curve')
        if controlpoints: # checking whether to plot control points
            ax.plot(*zip(*self.controlpoints), 'o--', label='Control
                Points')

        lgd = ax.legend(loc='upper left', bbox_to_anchor=(1,1))
        plt.title(title)
        plt.show()
```

```
        if filename:
            fig.savefig(filename, bbox_extra_artists=(lgd,),
                bbox_inches='tight')

if __name__ == '__main__':
    """
    ### Task 2 ###
    numPoints = 200
    grid = scipy.array([1,1,1,1,6/5,7/5,8/5,9/5,2,2,2,2])
    controlpoints = scipy.array([[0.7,-0.4],
                                 [1.0,-0.4],
                                 [2.5,-1.2],
                                 [3.2,-0.5],
                                 [-0.2,-0.5],
                                 [0.5,-1.2],
                                 [2.0,-0.4],
                                 [2.3,-0.4]])
    bspline = Bspline(grid, controlpoints, 3)
    title = 'B-Spline Curve with Knots
        $\{1,1,1,1,6/5,7/5,8/5,9/5,2,2,2,2\}$ \n' \
           'and Control Points Marked in the Plot'
    bspline.plot(title, filename='task2_1', markSeq=True, clamped=True)
    """
    #"""
    ### Task 4 ###
    grid = scipy.array([0,0,0,0,0,1/3,2/3,1,1,1,1,1])
    controlpoints = scipy.array([[0,0],
                                 [-4,0],
                                 [-5,2],
                                 [-4,4.5],
                                 [-2,5],
                                 [1,5.5],
                                 [1,0]])
    bspline = Bspline(grid, controlpoints, 4)
    title = 'B-spline Curve Defined by the Grid
        $\{0,0,0,0,0,1/3,2/3,1,1,1,1,1\}$ \n and Control Points Marked
        on the Curve.'
    bspline.plot(title, filename='task4')
    #"""
    """
    ### Task 5 ###
    grid =
        scipy.array([0,1/11,2/11,3/11,4/11,5/11,6/11,7/11,8/11,9/11,10/11,1])
    controlpoints = scipy.array([[0,0],
                                  [3,2],
                                  [9,-2],
                                  [7,-5],
                                  [1,-3],
                                  [1,-1],
                                  [3,1],
```

```
                          [9,-1]])
#for i in range(len(controlpoints) - 1):
 #   if i > 0:
  #      controlpoints[-i] = controlpoints[i-1]
   #    print('i=', i, controlpoints)
bspline = Bspline(grid, controlpoints, 3)
title = 'B-spline Curve Defined by the Grid
    $\{0,1/11,2/11,3/11,4/11,5/11,6/11,7/11,8/11,9/11,10/11,1\}$ \n
    and ' \
       'Control Points Marked on the Curve.'
bspline.plot(title)
"""
```