Computer Graphics and HCI Group
AG Computergrafik und HCI

FACHBEREICH INFORMATIK

Algorithmic Geometry WS 2016/2017

Prof. Dr. Hans Hagen, Benjamin Karer M.Sc., Alina Freund M.Sc.

2016-11-11

Notes:
Due date: 2016-11-23 (theory) & 2016-11-25 (programming)
Web: gfx.uni-kl.de/~alggeom
If you have questions or encounter any problems, feel free to visit me in room 36-218 or to send an email to karer@rhrk.uni-kl.de.

Sheet No.1: Interpolation

1) Vandermonde Matrix (T)

The coefficients of an interpolation polynomial for $n+1$ pairs of real numbers (t_i, f_i) can be calculated by solving a system of linear equations using the Vandermonde matrix V_{n+1}:

$$\underbrace{\begin{pmatrix} 1 & t_0 & t_0^2 & \dots & t_0^n \\ 1 & t_1 & t_1^2 & \dots & t_1^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & t_n & t_n^2 & \dots & t_n^n \end{pmatrix}}_{V_{n+1}} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{pmatrix} = \begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_n \end{pmatrix}$$

Prove, that $\det(V_{n+1}) = \prod_{i,j=0, i>j}^{n} (t_i - t_j)$.

Hint: Bring the entries of the first row to the form $(1, 0, 0, \dots, 0)$, then calculate the determinant using Laplace expansion along the first row. Use induction for the rest of the proof.

2) Interpolation (T)

a) **Newton Method**: The following points are given:

i	0	1	2
x_i	0.0	1.0	2.0
y_i	8.0	5.0	4.0

Calculate the interpolating Newton-polynomial.

b) **Lagrange Method**: The following points are given:

i	0	1	2
x_i	0.0	0.5	1.0
y_i	1.0	0.8	0.5

Calculate the interpolating Lagrange-polynomial.

3) Interpolating Splines (T)

a) **natural end conditions** The following two-dimensional points are given:

i	0	1	2	3	4
x_i	-1.0	-0.5	0.0	0.5	1.0
y_i	0.5	0.8	1.0	0.8	0.5

Calculate the interpolating cubic spline with natural end conditions.

b) **periodic end conditions** The following two-dimensional points are given:

i	0	1	2	3
x_i	-1.0	0.0	0.5	1.0
y_i	0.5	0.0	-1.0	0.5

Calculate the interpolating cubic spline with periodic end conditions.

4) Lagrange Interpolation (P)

In this exercise, you will implement the Lagrange Interpolation algorithm and test it for several sets of points. You can use whatever programming language you want but to make it easier, a QT project has been prepared and uploaded on the website for you.

a) **Getting started**
Download the given project file linked as “Ex1_ LagrangePolynomial" from the homepage of the lecture.
Open the files. You will find shaders for drawing the polynomial later on. The actual curve object is instantiated in the main file. In the class Ex1_LagrangePolynomial, you will further find the member variables for storing the support and curve points. Support points are of type Vector(4), and of the form $(x, y = f(x), 0, 1)^T$ Three function bodies for this task are given which are yet empty.

b) **Interpolation Algorithm**
Implement the methods "computeLagrangeBasisNumerators" and "computeLagrangeBasisDenominators". As you cannot actually calculate the polynomial straight forward, a possible idea is as follows: Calculating the divisions directly is rather difficult to achieve. Hence, we treat denominator and numerators separately in the form of $f(x) \cdot n(x)/d(x)$. $d(x)$ has to be calculated only once for each polynomials. $n(x)$ is meant to return a function body (as a std:vector) for each support point.

c) **Computing the polynomial**
Have a look at the function called "evaluate". Leave all code inside as it is but add the blending, i.e. add the functionality to calculate the polynomial for the number of values range_min to range_max. Store the values as vectors (of class Vector. Keep in mind that the last entry has to be a 1). Combine your previous results here, and complete the computation to obtain the points on the curve.

d) **Interpolation**
Test your curve on several nontrivial example point sets.