Natural cubic splines

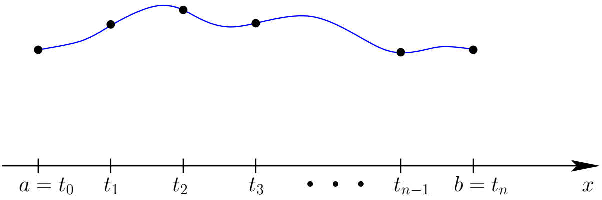
Arne Morten Kvarving

Department of Mathematical Sciences
Norwegian University of Science and Technology

October 21 2008

Motivation

- We are given a "large" dataset, i.e. a function sampled in many points.
- We want to find an approximation in-between these points.
- Until now we have seen one way to do this, namely high order interpolation - we express the solution over the whole domain as one polynomial of degree N for $N+1$ data points.



Motivation

- Let us consider the function

$$f(x) = \frac{1}{1+x^2}.$$

  Known as Runge's example.
- While what we illustrate with this function is valid in general, this particular function is constructed to really amplify the problem.

Motivation

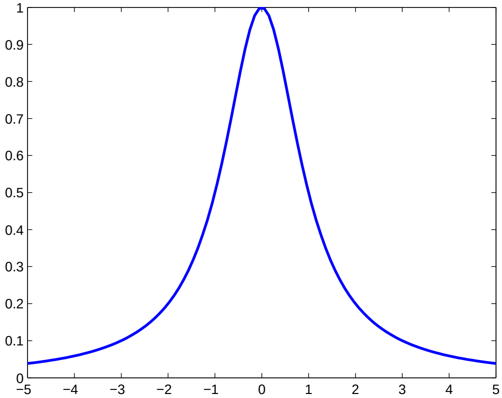


Figure: Runge's example plotted on a grid with 100 equidistantly spaced grid points.

Motivation

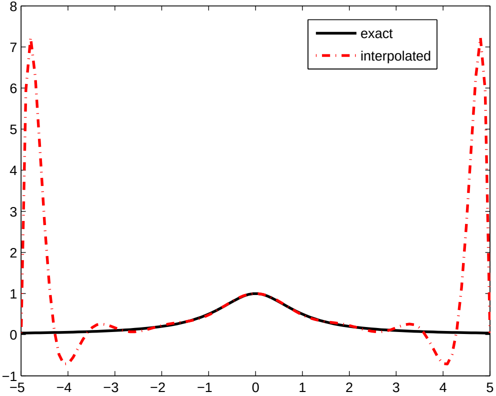
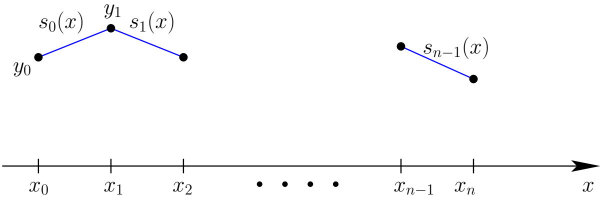


Figure: Runge's example interpolated using a 15th order polynomial based on equidistant sample points.

Motivation

- It turns out that high order interpolation using a global polynomial often exhibit these oscillations hence it is "dangerous" to use (in particular on equidistant grids).
- Another strategy is to use piecewise interpolation. For instance, piecewise linear interpolation.



Motivation

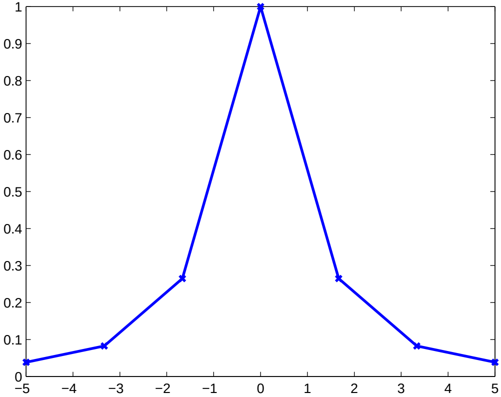
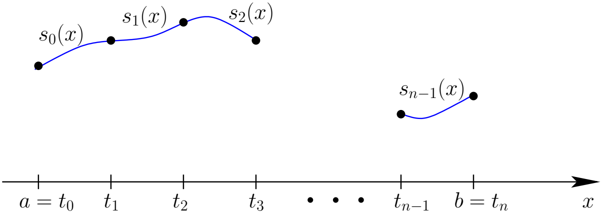


Figure: Runge's example interpolated using piecewise linear interpolation. We have used 7 points to interpolate the function in order to ensure that we can actually see the discontinuities on the plot.

A better strategy - spline interpolation

- We would like to avoid the Runge phenomenon for large datasets $\Rightarrow$ we cannot do higher order interpolation.
- The solution to this is using piecewise polynomial interpolation.
- However piecewise linear is not a good choice as the regularity of the solution is only C^0.
- These desires lead to splines and spline interpolation.



Splines - definition

A function $S(x)$ is a spline of degree k on $[a, b]$ if

- $S \in C^{k-1}[a, b]$.
- $a = t_0 < t_1 < \cdots < t_n = b$ and

$$S(x) = \begin{cases} S_0(x), & t_0 \le x \le t_1 \\ S_1(x), & t_1 \le x \le t_2 \\ \vdots & \\ S_{n-1}(x), & t_{n-1} \le x \le t_n \end{cases}$$

where $S_i(x) \in \mathbb{P}^k$.

Cubic spline

$$S(x) = \begin{cases} S_0(x) = & a_0x^3 + b_0x^2 + c_0x + d_0, \qquad t_0 \le x \le t_1 \\ \quad \vdots & \\ S_{n-1}(x) = & a_{n-1}x^3 + b_{n-1}x^2 + c_{n-1}x + d_{n-1}, \;\; t_{n-1} \le x \le t_n. \end{cases}$$

which satisfies

$$S(x) \in C^2[t_0, t_n] : \left. \begin{array}{r} S_{i-1}(x_i) = S_i(x_i) \\ S'_{i-1}(x_i) = S'_i(x_i) \\ S''_{i-1}(x_i) = S''_i(x_i) \end{array} \right\} , i = 1, 2, \cdots , n-1.$$

Cubic spline - interpolation

Given $(x_i, y_i)_{i=0}^{n}$. Task: Find $S(x)$ such that it is a cubic spline interpolant.

- The requirement that it is to be a cubic spline gives us $3(n-1)$ equations.
- In addition we require that

$$S(x_i) = y_i, \qquad i = 0, \cdots, n$$

  which gives $n+1$ equations.
- This means we have $4n-2$ equations in total.
- We have $4n$ degrees of freedom $(a_i, b_i, c_i, d_i)_{i=0}^{n-1}$.
- Thus we have 2 degrees of freedom left.

Cubic spline - interpolation

We can use these to define different subtypes of cubic splines:

- $S''(t_0) = S''(t_n) = 0$ - natural cubic spline.
- $S'(t_0), S'(t_n)$ given - clamped cubic spline.
-

$$\left.\begin{aligned} S_0'''\left(t_1\right) = S_1'''\left(t_1\right) \\ S_{n-2}''\left(t_{n-1}\right) = S_{n-1}'''\left(t_{n-1}\right) \end{aligned}\right\} \text{ - Not a knot condition (MATLAB)}$$

Natural cubic splines

Task: Find $S(x)$ such that it is a natural cubic spline.

- Let $t_i = x_i, i = 0, \cdots, n$.
- Let $z_i = S''(x_i), i = 0, \cdots, n$. This means the condition that it is a natural cubic spline is simply expressed as $z_0 = z_n = 0$.
- Now, since $S(x)$ is a third order polynomial we know that $S''(x)$ is a linear spline which interpolates (t_i, z_i).
- Hence one strategy is to first construct the linear spline interpolant $S''(x)$, and then integrate that twice to obtain $S(x)$.

Natural cubic splines

- The linear spline is simply expressed as

$$S_i''(x) = z_i \frac{x - t_{i+1}}{t_i - t_{i+1}} + z_{i+1} \frac{x - t_i}{t_{i+1} - t_i}.$$

- We introduce $h_i = t_{i+1} - t_i, i = 0, \cdots, n$ which leads to

$$S''(x) = z_{i+1} \frac{x - t_i}{h_i} + z_i \frac{t_{i+1} - x}{h_i}.$$

- We now integrate twice

$$\begin{aligned} S_i(x) = {} & \frac{z_{i+1}}{6h_i} (x - t_i)^3 + \frac{z_i}{6h_i} (t_{i+1} - x)^3 \\ & + C_i (x - t_i) + D_i (t_{i+1} - x). \end{aligned}$$

Natural cubic splines

- Interpolation gives:

$$S_i(t_i) = y_i \Rightarrow \frac{z_i}{6} h_i^2 + D_i h_i = y_i, i = 0, \cdots, n.$$

- Continuity yields:

$$S_i\left(t_{i+1}\right) = y_{i+1} \Rightarrow \frac{z_{i+1}}{6} h_i^2 + C_i h_i = y_{i+1}.$$

Natural cubic splines

- We insert these expressions to find the following form of the system

$$
\begin{aligned}
S_i(x) = {} & \frac{z_{i+1}}{6h_i}(x - t_i)^3 + \frac{z_i}{6h_i}(t_{i+1} - x)^3 \\
& + \left(\frac{y_{i+1}}{h_i} - \frac{z_{i+1}}{6}h_i\right)(x - t_i) \\
& + \left(\frac{y_i}{h_i} - \frac{h_i}{6}z_i\right)(t_{i+1} - x) .
\end{aligned}
$$

- We then take the derivative.

Natural cubic splines

- The derivative reads

$$S_i'(x) = \frac{z_{i+1}}{2h_i}(x - t_i)^2 - \frac{z_i}{2h_i}(t_{i+1} - x)^2$$
$$+ \underbrace{\frac{1}{h_i}(y_{i+1} - y_i)}_{b_i} - \frac{h_i}{6}(z_{i+1} - z_i).$$

- In our abscissas this gives

$$S_i'(t_i) = -\frac{1}{2}z_i h_i + b_i - \frac{h_i}{6}z_{i+1} + \frac{1}{6}h_i z_i$$
$$S_i'(t_{i+1}) = \frac{z_{i+1}}{2}h_i + b_i - \frac{h_i}{6}z_{i+1} + \frac{1}{6}h_i z_i$$
$$S_{i-1}(t_i) = \frac{1}{3}z_i h_{i+1} + \frac{1}{6}h_{i-1}z_{i-1} + b_{i-1}$$
$$S_i'(t_i) = S_{i-1}(t_i) \Rightarrow$$
$$6(b_i - b_{i-1}) = h_{i-1}z_{-1} + 2(h_{i-1} + h_i)z_i + h_i z_{i+1}.$$

Natural cubic splines - algorithm

This means that we can find our solution using the following procedure:

- First do some precalculations

$$
\begin{aligned}
h_i &= t_{i+1} - t_i, & i &= 0, \cdots, n-1 \\
b_i &= \frac{1}{h_i}\left(y_{i+1} - y_i\right), & i &= 0, \cdots, n-1 \\
v_i &= 2\left(h_{i-1} + h_i\right), & i &= 1, \ldots, n-1 \\
u_i &= 6\left(b_i - b_{i-1}\right), & i &= 1, \cdots, n-1 \\
z_0 &= z_n = 0
\end{aligned}
$$

Natural cubic splines - algorithm

- Then solve the tridiagonal system

$$\begin{bmatrix} v_1 & h_1 & & & & \\ h_1 & v_2 & h_2 & & & \\ & h_2 & v_3 & h_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & h_{n-2} \\ & & & & h_{n-2} & v_{n-1} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_{n-2} \\ z_{n-1} \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{n-2} \\ u_{n-1} \end{bmatrix}.$$

Natural cubic splines - example

- Given the dataset

i	0	1	2	3
x_i	0.9	1.3	1.9	2.1
y_i	1.3	1.5	1.85	2.1
$h_i = x_{i+1} - x_i$	0.4	0.6	0.2	
$b_i = \frac{1}{h_i}\left(y_{i+1} - y_i\right)$	0.5	0.5833	1.25	
$v_i = 2\left(h_{i-1} + h_i\right)$		2.0	1.6	
$u_i = 6\left(b_i - b_{i-1}\right)$		0.5	4	

- The linear system reads

$$\begin{bmatrix} 2.0 & 0.4 \\ 0.4 & 1.6 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.4 \end{bmatrix}$$

Natural cubic splines - example

- We find $z_0 = 0.5, z_1 = 0.125$. This gives us our spline functions

$$S_0(x) = 0.208\,(x - 0.9)^3 + 3.78\,(x - 0.9) + 3.25\,(1.3 - x)$$
$$S_1(x) = 0.035\,(x - 1.3)^3 + 0.139\,(1.9 - x)^3 + 0.664 - 0.62x$$
$$S_2(x) = 0.104\,(x - 1.9)^3 + 10.5\,(x - 1.9) + 9.25\,(2.1 - x)$$

Gaussian elimination of tridiagonal systems

- Assume we are given a general tridiagonal system

$$\begin{bmatrix} d_1 & c_1 & & & \\ a_1 & d_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & & & c_{n-1} \\ & & & a_{n-1} & d_n \end{bmatrix}, \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{n-1} \\ b_n \end{bmatrix}.$$

- First elimination (second row) yields

$$\begin{bmatrix} d_1 & c_1 & & & \\ 0 & \tilde{d}_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & & & c_{n-1} \\ & & & a_{n-1} & d_n \end{bmatrix}, \begin{bmatrix} b_1 \\ \tilde{b}_2 \\ \vdots \\ b_{n-1} \\ b_n \end{bmatrix}, \qquad \begin{aligned} \tilde{d}_2 &= d_2 - \frac{a_1}{d_1} c_1 \\ \tilde{b}_2 &= b_2 - \frac{a_1}{d_1} b_1 \end{aligned}$$

Gaussian elimination of tridiagonal systems

- This means that the elimination stage is

$$
\begin{aligned}
&\text{for } i = 2, \cdots, n \\
&\qquad m = a_{i-1}/d_{i-1} \\
&\qquad \tilde{d}_i = d_i - mc_{i-1} \\
&\qquad \tilde{b}_i = b_i - mb_{i-1} \\
&\text{end}
\end{aligned}
$$

- And the backward substitution reads

$$
\begin{aligned}
&x_n = \tilde{b}_n/d_n \\
&\text{for } i = n-1, \cdots, 1 \\
&\qquad x_i = \left(\tilde{b}_i - c_i x_{i+1}\right)/\tilde{d}_i \\
&\text{end}
\end{aligned}
$$

where $\tilde{b}_1 = b_1$.

Gaussian elimination of tridiagonal systems

- This will work out fine as long as $\tilde{d}_i \neq 0$.
- Assume that $|d_i| > |a_{i-1}| + |c_i|$ - i.e. diagonal dominance.
- For the eliminated system diagonal domiance means that

$$|\tilde{d}_i| < |c_i|.$$

- We now want to show that diagonal domiance of the original system implies that the eliminated system is also diagonal dominant.

Gaussian elimination of tridiagonal systems

- We now assume that $|\tilde{d}_{i-1}| > |c_{i-1}|$. This is obviously satisfied for $\tilde{d}_1 = d_1$.

$$
\begin{aligned}
|\tilde{d}_i| = |d_i - \frac{a_{i-1}}{\tilde{d}_{i-1}} c_{i-1}| &\geq |d_i| - \frac{|a_{i-1}|}{|\tilde{d}_{i-1}|} |c_{i-1}| \\
&> |a_{i-1} - |c_i| - |a_{i-1}| = |c_i|.
\end{aligned}
$$

- Hence the diagonal domiance is preserved which means that $\tilde{d}_i \neq 0$. The algorithm produces a unique solution.

Why cubic splines?

- Now to motivate why we use cubic splines.
- First, let us introduce a measure for the smoothness of a function:

$$\mu(f) = \int_a^b (f''(x))^2 \, \mathrm{d}x. \tag{1}$$

- We then have the following theorem

Theorem

Given interpolation data $(t_i, y_i)_{i=0}^n$. *Among all functions* $f \in C^2[a, b]$ *which interpolates* (t_i, y_i), *the natural cubic spline is the smoothest, where smoothness is measured through* (1).

Why cubic splines?

- We need to prove that

$$\mu(f) \geq \mu(S) \, \forall \, f \in C^2[a, b].$$

- Introduce

$$g(x) = S(x) - f(x), \quad \begin{array}{c} g(x) \in C^2[a, b] \\ g(t_i) = 0, i = 0, \cdots, n. \end{array}$$

- Inserting this yields

$$\begin{aligned} \mu(f) &= \int_a^b \left(S''(x) - g''(x)\right)^2 \, \mathrm{d}x \\ &= \mu(S) + \mu(g) - 2\int_a^b S''(x) g''(x) \, \mathrm{d}x \end{aligned}$$

Now since $\mu(g) > 0$, we have proved our result if we can show that

$$\int_a^b S''(x) g''(x) \, \mathrm{d}x = 0.$$

Why cubic splines?

- We have that

$$\int_a^b S''(x)g''(x)\,\mathrm{d}x = g'(x)S''(x)\big|_a^b - \int_a^b g'(x)S'''(x)\,\mathrm{d}x$$

First part on the right hand side is zero since $z_0 = z_n = 0$. Second part we split in an integral over each subdomain

$$\begin{aligned} -\int_a^b g'(x)S'''(x)\,\mathrm{d}x &= -\sum_{i=0}^{n-1} \int_{t^i}^{t^{i+1}} g'(x)S'''(x)\,\mathrm{d}x \\ &= -\sum_{i=0}^{n-1} 6a_i \int_{t_i}^{t_{i+1}} g'(x)\,\mathrm{d}x \\ &= -\sum_{i=0}^{n-1} 6a_i\, g(x)\big|_{t_i}^{t_{i+1}} = 0. \end{aligned}$$

Cubic spline result

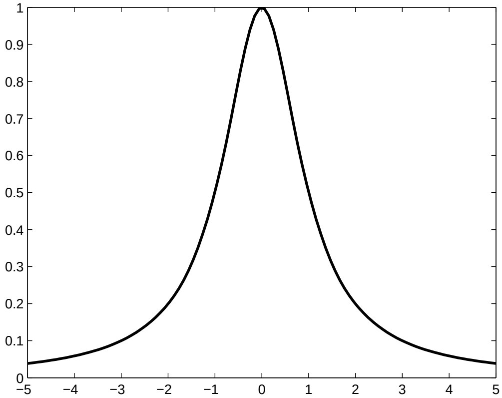


Figure: Runge's example interpolated using cubic spline interpolation based on 15 equidistant samples.