Introduction to Scientific Computing
Lecture 7

Professor Hanno Rein

Last updated: October 26, 2017

1 Cubic spline interpolation

Instead of going to higher and higher order, there is another way of creating a smooth function that interpolates data-points. A cubic spline is a piecewise continuous curve that passes through all of the values of a given dataset. This works particularly well for smooth datasets with no noise. Each of the piecewise curves is a cubic polynomial with coefficients a_i, b_i, c_i and d_i:

$$S_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i \qquad \text{for } x \in [x_i, x_{i+1}]$$

If we have N data-points, there are $N - 1$ intervals, hence $(N - 1) \cdot 4$ coefficients that we need to find. Two conditions in each interval arise because we have to match the two data-points at each end.

$$S_i(x_i) = y_i, \quad S_i(x_{i+1}) = y_{i+1}$$

How about the other two parameters? We want to have a smooth function! Thus, we require that the derivatives of the piecewise functions match at the interval boundaries:

$$S'_{i-1}(x_i) = S'_i(x_i), \quad S''_{i+1}(x_i) = S''_i(x_i)$$

We now have almost as many conditions as we have free parameters, except at the boundaries.What could we possibly do there? There are multiple choices and it depends on the problem. We use one called *natural* boundary condition which says that we set the second derivatives to zero at the boundary.

As you can probably guess, this set of equations that we are generating will become a matrix equation. Let's go through the individual steps. Finding the value for the d_is is simple. Our requirement gives us

$$d_i = S_i(x_i) = y_i$$

The condition to match the point at $i + 1$ gives

$$S_i(x_{i+1}) = a_i(x_{i+1} - x_i)^3 + b_i(x_{i+1} - x_i)^2 + c_i(x_{i+1} - x_i) + d_i = y_{i+1}$$

Let's call the derivatives at point i, D_i, i.e.

$$S'_i(x_i) = D_i = c_i$$

and therefore

$$S'_i(x_{i+1}) = D_{i+1} = 3a_i(x_{i+1} - x_i)^2 + 2b_i(x_{i+1} - x_i) + D_i$$

We can now setup an equation system for a, b, c and d:

$$\begin{aligned} a_i &= (D_{i+1} + D_i)(x_{i+1} - x_i)^{-2} - 2(y_{i+1} - y_i)(x_{i+1} - x_i)^{-3} \\ b_i &= (-D_{i+1} - 2D_i)(x_{i+1} - x_i)^{-1} + 3(y_{i+1} - y_i)(x_{i+1} - x_i)^{-2} \\ c_i &= D_i \\ d_i &= y_i \end{aligned}$$

We have one requirement left to play with, to match second derivatives at each interval. This gives us:

$$\begin{aligned} S''_{i-1}(x_i) &= S''_i(x_i) \\ S''_i(x_{x+i}) &= S''_{i+1}(x_{i+1}) \end{aligned}$$

which equates to

$$6a_i(x_{i+1} - x_i) + 2b_i = 2b_{i+1}$$

Let us now combine this with the earlier equation to get

$$\begin{aligned} 3(y_i - y_{i-1})(x_i - x_{i-1})^{-1}(x_{i+1} - x_i) + 3(y_{i+1} - y_i)(x_{i+1} - x_i)^{-1}(x_i - x_{i-1}) \\ = D_{i-1}(x_{i+1} - x_i) \\ + D_i(3(x_{i+1} - x_i) + (x_i - x_{i-1})) \\ + D_{i+1}(x_i - x_{i-1}) \end{aligned}$$

Let us define

$$Y_i \equiv 3\frac{y_i - y_{i-1}}{x_i - x_{i-1}}(x_{i+1} - x_i) + 3\frac{y_{i+1} - y_i}{x_{i+1} - x_i}(x_i - x_{i-1})$$

We can write this as a matrix equation

$$\begin{pmatrix} \ddots & & & & \\ & (x_{i+1} - x_i) & (3x_{i+1} - 2x_i - x_{i-1}) & (x_i - x_{i-1}) & \\ & & & & \ddots \end{pmatrix} \cdot \begin{pmatrix} \vdots \\ D_i \\ \vdots \end{pmatrix} = \begin{pmatrix} \vdots \\ Y_i \\ \vdots \end{pmatrix}$$

At the end points, we have not enough information to fully determine all variables. We therefore come up with new requirements there, let the second derivatives be zero.

$$\begin{aligned} S''_0(x_0) &= 2b_0 = 0 \\ S''_{N-2}(x_{N-1}) &= 6a_{N-2}(x_{N-1} - x_{N-2}) + 2b_{N-2} = 0 \end{aligned}$$

This gives in terms of the D coefficients:

$$\begin{aligned} 2D_0 + D_1 &= 3\frac{y_1 - y_0}{x_1 - x_0} \equiv Y_0 \\ D_{N-2} + 2D_{N-1} &= 3\frac{y_{N-1} - y_{N-2}}{x_{N-1} - x_{N-2}} \equiv Y_{N-1} \end{aligned}$$

We can now complete the matrix from above to

$$\begin{pmatrix} 2 & 1 & 0 & \cdots & & \\ (x_2-x_1) & (3x_2-2x_1-x_0) & (x_1-x_0) & 0 & \cdots & \\ 0 & (x_3-x_2) & (3x_3-2x_2-x_1) & (x_2-x_1) & 0 & \cdots \\ & & & \ddots & & \\ & & (x_{i+1}-x_i) & (3x_{i+1}-2x_i-x_{i-1}) & (x_i-x_{i-1}) & \\ & & & \ddots & & \\ & & & & 0 & 1 \quad 2 \end{pmatrix}$$

$$\cdot \begin{pmatrix} D_0 \\ \vdots \\ D_i \\ \vdots \\ D_{N-1} \end{pmatrix} = \begin{pmatrix} Y_0 \\ \vdots \\ Y_i \\ \vdots \\ Y_{N-1} \end{pmatrix}$$

This is a tridiagonal system and can easily be solved. Here, we just use the Gaussian eliminiation that we already know. It is easy to find more efficient ways to solve it, but we don't bother.

Once solved for the D_is, we can solve for the a_i, b_i, c_i and d_is. This then defined all parameters for the piecewise cubic function.

In the following figure, we apply this method to the average temperature in Toronto. As you can see, it gives a very smooth and reasonable fit.

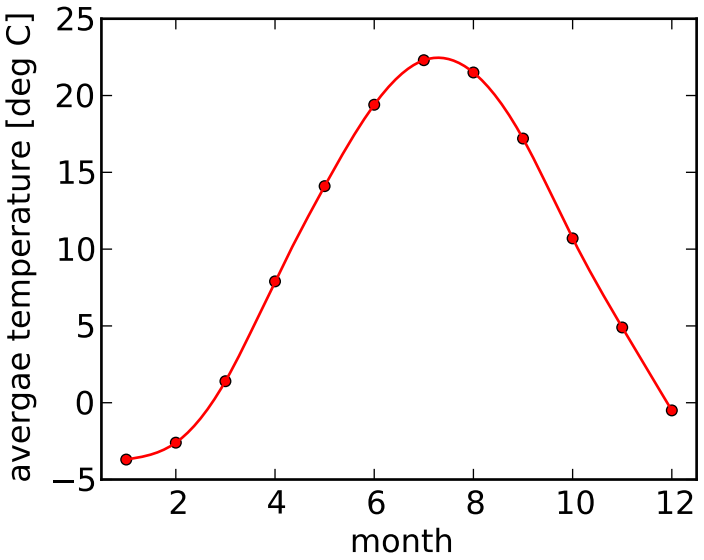

Figure 1: Temperature in Toronto. Cubic spline interpolation.

One way to further improve this spline is to take advantage of the fact that the temperature is periodic. This removes the extra criteria at the boundaries. The matrix will get some extra components (i.e. is not tridiagonal anymore).

Now, we have this great spline interpolation method. Can you use it for any problem? No. Here's a word of caution. Look at the following dataset.

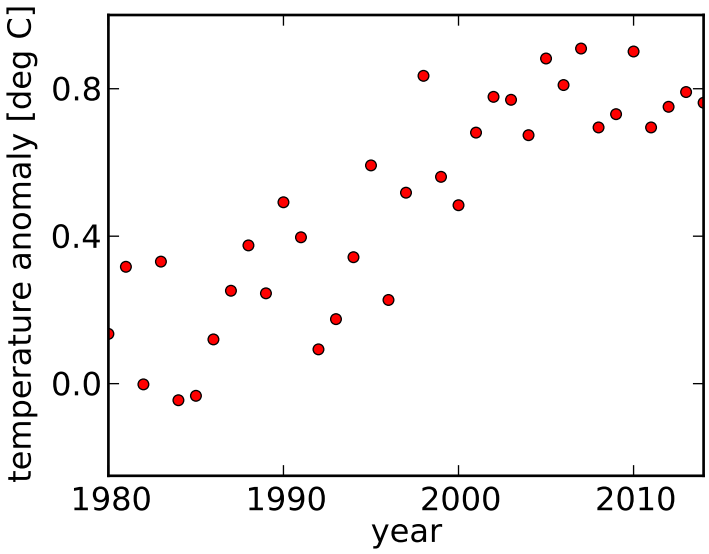

Figure 2: Global temperature anomaly. Source: Climatic Research Unit (University of East Anglia).

The above figure shows the global temperature anomaly. This is an indication of climate change. Clearly, there is a lot of noise in the data. Nevertheless, one can see a very dominant trend towards higher temperatures. We could just fit a spline to this curve. The result is shown in the following figure.

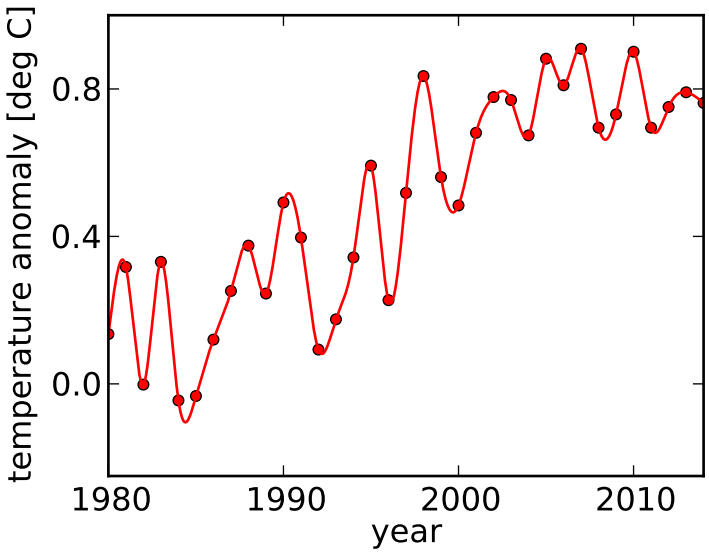

Figure 3: Global temperature anomaly fitted with a cubic spline.

Although the spline is clearly going through all data-points and the curve is smooth at every point in the interval, it is not a good indication of the trend. Why is that? We have fitted a smooth function to noisy data. It's the same issue that we encountered earlier and is sometimes called *ringing* or *Runge's phenomenon*.

Thus, interpolating this data with a constant or piece-wise linear function or even a cubic spline does not make much sense. We would effectively try to interpolate the noise, not the data. So how can we interpolate this data in a meaningful way. We can use a least square fit!

For example, a straight line fit will give us an indiction of how fast the temperature anomaly has risen over time. We can even use it to extrapolate how much temperatures will be rising in the future. The idea of a straight line fit is to find a linear function

$$f(x) = a_1 + a_2 x$$

which is the *best* fit to the data (see our earlier discussion on least square fits). Figure 4 shows the straight line for for the climate data discussed above.

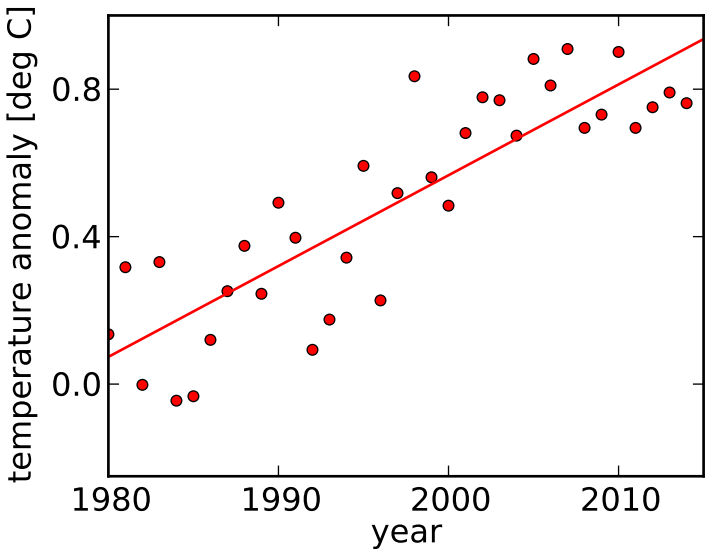

Figure 4: Global temperature anomaly with a straight line fit.

2 Differential equations

Differential equations are equations where we are solving for a function rather than a variable. For example, instead of solving for x in a normal algebraic equation such as

$$a \cdot x = b$$

we now solve for a function $y(t)$ in a differential equation such as

$$y(t) = a \cdot \frac{\partial y(t)}{\partial t}.$$

Differential equations are closely related to integral equations. The differential and the integral operators are the inverse of each other. So for every differential equation there is one corresponding integral equation. For the above example, this would be

$$\int y(t)\, dt = a \cdot y(t).$$

We'll usually work with differential equations rather than integral equations because they are more intuitive most of the time.

Let's discuss a few examples of differential equations. For these differential equations we can write down exact solutions.

Example 1: The following differential equation pops up in a variety of places in biology and physics:

$$N(t) = a \cdot N'(t).$$

It says that the growth of the function, $N'(t)$, is proportional to the value of the function $N(t)$. The solution is an exponential:

$$N(t) = N_0 \cdot e^{a \cdot t}$$

Here, N_0 is an initial value that is determined by the boundary conditions.

Example 2: The first example leads to a exponentially growing or decreasing function. Another class of differential equations leads to periodic solutions. For example

$$m \cdot x''(t) = -k \cdot x(t)$$

has a solution

$$x(t) = A \cdot \cos(\omega t + \phi) \qquad \text{with} \qquad \omega^2 = \frac{k}{m}.$$

This is the solution for a harmonic oscillator, a very common problem in physics.

So, let's go back to the initial conditions. What are these values N_0, A and ϕ? They are given to you as part of the problem description. For example, in the first example, if you think of N as the number of creatures in a biology model, then N_0 is the number od creatures at the beginning of the experiment. Similarly, in the second example, the initial position and velocity of your pendulum determine the constants A and ϕ.

The unknown function is not entirely specified by the differential equation itself. We also need to define certain boundary conditions. The nature of the boundary conditions varies depending on the problem. They can be as simple as requiring that the function has a certain value at $t = 0$. But they can also be complex algebraic equations. Different boundary conditions lead to qualitatively different problems and solutions. There are two main kinds of differential equations.

- The first kind has the boundary conditions specified at one point.
- The second kins has boundary conditions for two points

In this course we will focus on the first kind.

2.1 Euler Method

The simplest method to solve a differential equation of the form

$$y'(t) = F(y,t)$$

is the Euler method. We start from an initial condition consisting of two values, t_0 and $y_0 = y(t_0)$. We then integrate the equation forward in time, one timestep dt at a time. Often the timestep is also denoted by the letter h. The size of the timestep determines the precision of the scheme. The smaller the timestep the higher the accuracy but the more calculations we have to do. This statement is true not only for the Euler method but for all ODE (ordinary differential equation) solvers.

To perform an Euler step, we evaluate the derivate of the function $y(t)$ at the beginning of the timestep. For the first timestep, t is t_0, for the second timestep $t_0 + dt$, and so on. We then multiply the derivative with the timestep dt and add it to the initial value y_0. If we label subsequent steps with the index n, then the Euler method can be written as

$$y_{n+1} = y_n + dt \cdot F(y_n, t_n).$$

Note that you can think of the the function y as either a scalar function or a vector function. The Euler method (and any other ODE solver) works exactly the same in both cases. In other words, it works in multiple dimensions as well as in just one. This can be used to convert an ODE with a second order derivative of y into a system of first order derivatives in vector notation. As an example, suppose we want to solve the ODE

$$y'' = F(y)$$

then we can rewrite this as a vector equation and first derivatives as

$$\begin{pmatrix} y' \\ y'' \end{pmatrix} = \begin{pmatrix} y' \\ F(y) \end{pmatrix}$$

and thus

$$\begin{pmatrix} y \\ y' \end{pmatrix}' = \begin{pmatrix} y' \\ F(y) \end{pmatrix}$$

or equally

$$Y' = G(Y)$$

where we introduced new variables Y and the function G.

We now perform an error analysis on the Euler method. We do this to find out how well the Euler method does, how large the error is and how it depends on the timestep dt. So, let's go back to our generic example

$$y'(t) = F(y,t)$$

and expand the function y as a Taylor series around the $t = t_0$

$$\begin{aligned} y(t) &= y_0 + (t - t_0) \cdot \frac{\partial y}{\partial t} + \frac{1}{2}(t - t_0)^2 \cdot \frac{\partial^2 y}{\partial t^2} + \cdots \\ &= y_0 + dt \cdot F(y,t) + \frac{1}{2} dt^2 \cdot \frac{\partial^2 y}{\partial t^2} + \cdots \end{aligned}$$

The Euler method gets the first two terms right. The error after one timestep is therefor of the order

$$E \sim \frac{1}{2} dt^2 \cdot \frac{\partial^2 y}{\partial t^2}$$

plus higher order terms. The higher order terms will be small if dt is small and the dominant error term will be of order dt^2.

Suppose we want to integrate a system forward in time for some finite time T. Then the number of timesteps N depends on the size of the timestep dt:

$$N = \frac{T}{dt}$$

The smaller the timestep, the more timesteps we need to take. Thus, the error of the Euler method after a finite time T is now not of order dt^2 anymore, but of order dt. This is an important result. The Euler method is a first order method. If we reduce the timestep by a factor of two, then the error will be smaller by a factor of two. This is good because we eventually converge to the correct solution if we only make the timestep small enough. However, we might hope to find a better method that is more accurate than the Euler method and converges faster, i.e. quadratically. This is important because in most situations the evaluation of the function F is very computationally expensive and we want to minimize the number of evaluations. The Euler method has one function evaluation per timestep.

2.2 Runge-Kutta Methods

As we indicated before, we might be able to cancel out higher order terms in the expansion of y. This would lead us to a high order scheme which is better and faster than the simple Euler Methods. A class of higher order methods that is widely used is the class of Runge-Kutta Methods. These methods use the Euler method to make predictions of y during the timestep and then use this information to improve the result at the end.

The mid-point method, or second-order Runge-Kutta method, is one such method. The idea is simple. Let's first take half an Euler step, to get an estimate of the function F in the middle of the timestep. Then use this estimate to advance y from the beginning of the timestep to the end. In terms of equations, it can be written as follows (using the same notation as above):

$$\begin{aligned} k_1 &= dt \cdot F(y_n, t_n) \\ k_2 &= dt \cdot F\left(y_n + \frac{1}{2} \cdot k_1, t_n + \frac{1}{2}dt\right) \\ y_{n+1} &= y_n + k_2 \end{aligned}$$

Let's do an error analysis for this method. The Taylor expansion of the function y to third order is

$$y(t) = y_0 + (t - t_0) \cdot \frac{\partial y}{\partial t} + \frac{1}{2}(t - t_0)^2 \cdot \frac{\partial^2 y}{\partial t^2} + \frac{1}{3}(t - t_0)^3 \cdot \frac{\partial^3 y}{\partial t^3} + \cdots .$$

An integration step of the mid-point method gives

$$\begin{aligned} y_{n+1} &= y_n + k_2 \\ &= y_n + dt \cdot F\left(y_n + \frac{1}{2} \cdot k_1, t_n + \frac{1}{2}dt\right) \\ &= y_n + dt \cdot F\left(\underbrace{y_n + \frac{1}{2} \cdot dt \cdot F(y_n, t_n)}_{=y_{n+\frac{1}{2}} + O(dt^2)}, t_n + \frac{1}{2}dt\right) \\ &= y_n + dt \cdot \underbrace{F\left(y_n + \frac{1}{2} \cdot dt \cdot F(y_n, t_n), t_n + \frac{1}{2}dt\right)}_{=F_{n+\frac{1}{2}} + O(dt^2)} \end{aligned}$$

We've seen this before. This is the central difference scheme from last lecture (ignoring the higher order error term). The central difference scheme was used to estimate the derivate of a function. It is symmetric with respect to a reflection around the mid-point. This makes it higher order (2nd). Here, this is what makes the mid-point method second order accurate. We now have an error of order $O(dt^3)$ after one timestep. If the error of a scheme is $O(dt^{n+1})$ after one timestep, then we call the scheme $n-$th order.

The most commonly used Runge-Kutta method is the fourth order Runge-Kutta Method, or RK4. It uses four function evaluations and can be written in a similar way as the mid-point method.

$$\begin{aligned} k_1 &= dt \cdot F(y_n, t_n) \\ k_2 &= dt \cdot F\left(y_n + \frac{1}{2} \cdot k_1, t_n + \frac{1}{2}dt\right) \\ k_3 &= dt \cdot F\left(y_n + \frac{1}{2} \cdot k_2, t_n + \frac{1}{2}dt\right) \\ k_4 &= dt \cdot F\left(y_n + \cdot k_3, t_n + dt\right) \\ y_{n+1} &= y_n + \frac{1}{6}k_1 + \frac{1}{3}k_2 + \frac{1}{3}k_3 + \frac{1}{6}k_4 \end{aligned}$$

As one can show (we won't) the error after one timestep is $O(h^5)$, thus it is indeed a fourth order method.

There are many different Runge-Kutta scheme. They are usually written in block form. For example, the coefficients for RK4 can be summarized as

0				
$\frac{1}{2}$	$\frac{1}{2}$			
$\frac{1}{2}$		$\frac{1}{2}$		
1			1	
	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{6}$

Note that the 0's are not printed in the block form. Similarly, the Euler method can be summarized as

0	
	1

And the midpoint method can be summarized as

0		
$\frac{1}{2}$	$\frac{1}{2}$	
	0	1