Cubic Spline Interpolation

Leigh J. Little SUNY Brockport October 13, 2021

# Contents



# 1 Introduction

We have been using interpolation to estimate missing entries from a table of values for some function $y = f(x)$. These have been tables of known values so that we can compare the accuracy of the methods but in most cases the function defined in the table is unknown.

What we have observed is that simple methods (linear interpolation) are easy to understand and compute by hand but lack accuracy. Higher order methods are more accurate, but their implementation is complicated. Finally, using the whole table to build a single polynomial of very high degree polynomial can be disastrous. The results are summarized in Table 1 below for the table of values used in Homework 17.

| $x^*$ | Linear | Quadratic | Cubic | Whole Table |
|---|---|---|---|---|
| 0.01 | $9.90 \cdot 10^{-1}$ | $1.74 \cdot 10^{-4}$ | $2.44 \cdot 10^{-5}$ | $3.26 \cdot 10^{-1}$ |
| 1.40 | $5.40 \cdot 10^{-2}$ | $1.36 \cdot 10^{-4}$ | $3.88 \cdot 10^{-5}$ | $5.04 \cdot 10^{-5}$ |
| 3.10 | $4.16 \cdot 10^{+0}$ | $3.21 \cdot 10^{-3}$ | $1.06 \cdot 10^{-3}$ | $4.17 \cdot 10^{+2}$ |

Table 1: Relative errors at specified $x^*$ values for interpolation techniques using the table of values from Homework 17. Notice the large errors present in the Whole Table polynomial.

It turns out there is no one-size-fits-all solution to our problem. What we need is a method that 'works most of the time.' More specifically, what we need is a method with the following characteristics:

a) The method should be easy to work with.

b) The method should not require any additional information beyond what is in the table of values. This is an important item because oftentimes, the values in the table are all that are known about the function $f(x)$.

c) The method should be reliable and reasonably accurate (relative errors of magnitude $10^{-4}$ or less).

# 2 Cubic Splines

Our previous observations and Table 1 imply that cubic polynomial approximations seem to be a good compromise. They are reliable and accurate, but using them is complicated.

A cubic spline is a special function that consists of a series of cubic polynomials that span the entire range of the table. An example of a cubic spline is shown in Figure 1. Starting from a table with $n + 1$ values, a sequence of $n$ cubic polynomials is constructed. The polynomials are continuous at the locations where they join but also have additional smoothness properties imposed to ensure that the first and second derivatives of the curves are also continuous where the curves join.

It turns out that this is exactly what we are looking for. The spline satisfies characteristics a) and c) in the previous section. However, characteristic b) is not met. There is not enough information in the table alone to compute the spline completely. Two additional conditions must be stated. Because of this, there are multiple versions of cubic splines. The three most common are:

- **Natural Spline** - For this spline, the second derivative $f''(x)$ of $f(x)$ is assumed to be zero at the endpoints. With these conditions no additional information beyond what is in the table of values is required.

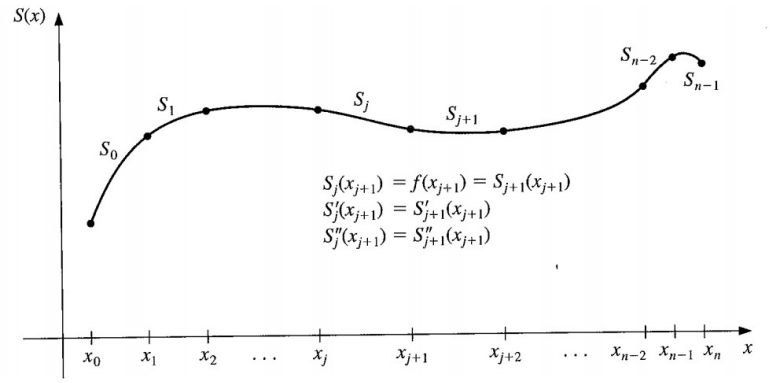


Figure 1: An example of a cubic spline. A table of $n+1$ points is used to construct a sequence of $n$ cubic polynomials, $S_j$. Taken from *Numerical Analysis*, Burden, Faires, 9th Edition.

- **Not-a-Knot** - For this spline, the condition that the spline have a continuous third derivative at points $x_1$ and $x_{n-1}$ in Figure 1 is imposed. Again, with this assumption no additional information beyond what is in the table of values is required.
- **Clamped Spline** - For this spline, known values of $f'(x)$ at the endpoints of the table are required. If these values are not known exactly, estimates can be obtained from the values in the table.

# 3 Computing the Spline

MATLAB has a built-in function that will compute the non-a-knot and clamped splines. I'm not sure why the natural spline is not included. If you have your table of values in two column vectors $x$ and $y$, you can compute the not-a-knot spline using

```
S = spline(x,y);
```

If you also have values of $f'(x)$ at the endpoints of the table (call these $f_1$ and $f_n$) you can compute the clamped spline using

```
S = spline(x,[f1; y; fn]);
```

In either case, you can compute the value of the spline at a given $x^*$ (which can be either a vector or a scalar) by doing

```
ystar = ppval(S,xstar);
```

With the use of splines, the process of interpolating accurate values (most of the time) is very easy. For example,

```
load hw17.dat
x = hw17(:,1);
y = hw17(:,2);

S = spline(x,y);
xstar = input('input xstar: ');
ystar = ppval(S,xstar);
ye = 1 + xstar^2*cos(xstar)*sin(xstar);
ae = abs(ystar - ye);
re = ae/abs(ye);
```