Spline Interpolation

Leigh J. Little SUNY Brockport November 9, 2018

Contents

1 Introduction

Interpolation is an important area in scientific computing. The essence of interpolation is this: Given a table of values for some data, determine values that are not in the table. Consider the data shown in Table 1. This data represents some function, but the formula for the function is unknown. All we know is the data in the table. How could we determine what the value of $T(x)$ is when $x = 0.33$? How could we determine what value(s) of x correspond to $T(x) = 1.00$? This a common situation. Often times, a formula for the function

x	$T(x)$
-1.00	3.08
-0.70	1.48
-0.40	0.90
-0.10	0.91
0.20	1.25
0.50	1.80
0.80	2.51
1.10	3.40
1.40	4.53
1.70	6.00
2.00	7.93

Table 1: Table of values for some function $T(x)$.

is not known. Instead, all that is known is a table of values. This table can come from either experimental measurements or the table might be the result of some other calculation (for example, the solution to a differential equation).

2 What We are Looking For

Interpolation is a broad field and the specific techniques used depend on the quality of the data and what information needs to be obtained from the data. So far, we have used linear interpolation. This computes the line that goes through successive points in the table. This is the most common form of interpolation but it suffers from lack of accuracy and flexibility.

Recall the example of determining the maximum value of a function. Linear interpolation will underestimate the true maximum. To address this, we used quadratic interpolation. In this case, a quadratic function was fit through three consecutive data points. This resulting quadratic approximating function can have curvature, which permits a better approximation of this maximum.

What we would like to have is a method of accurately predicting values in the table to the point that we can use this method as easily as we use any other function. In other words we would like to be able to compute

$$T = \text{Value of } T \text{ for some input value of } x.$$

as easily as we compute a known function like

$$y = \cos(x) - x.$$

In particular, we would like to be able to do all the types of things we have done to this point, for example, integrating the function, finding its roots or using it as the right-hand side function in a differential equation.

3 Cubic Spline Interpolation

The most common technique for doing this is known as cubic spline interpolation. A cubic spline is a set of cubic polynomials that go through the data points in the table, but its construction also permits accurate values for the first and second derivatives of the data. The cubic spline technique is optimal in the sense that it is reasonably accurate, can be efficiently computed and requires no additional information beyond the values given in the table.

Cubic spline interpolation is appropriate in the following circumstances:

- The raw data points form a smooth curve when plotted. Spline interpolation is not appropriate for many best-fit line data sets due to the lack of smoothness in the data.
- The range of x values we intend to input lie with the x range for the table. Attempting to predict function values that lie outside this range is called *extrapolation* and is a separate area of study.

Fortunately, MATLAB has built-in functions to construct the spline and to use it to evaluate the T value for any set of input values x. Type in the following commands

```
>> load testdata.dat      % Table 1 data to 5 digits instead  of 3
>> xdata = testdata(:,1);
>> Tdata = testdata(:,2);
>> T = spline(xdata,Tdata);
```

This is the process for creating the spline `T`. If you look at the spline, it is not a number. It a more complex way of storing data called a *structure*. For now, all we need to know is that `T` is what is needed to compute $T(x)$ values for any x value. To do this, use the `ppval` function. In order to compute the T value for $x = 1.33$, we would do

```
>> x = 1.33;
>> Tval = ppval(T,x)
Tval =
    4.2489
```

One property of the spline is that it should reproduce the data in the table. We can test this by sending `xdata` to the `ppval` function. We should recover the original `ydata` (with some small rounding errors).

```
>> Ttest = ppval(T,xdata);
>> [Tdata Ttest]
ans =
    3.0862    3.0862
    1.4833    1.4833
    0.9090    0.9090
    0.9159    0.9159
    1.2542    1.2542
    1.8004    1.8004
    2.5131    2.5131
    3.4069    3.4069
    4.5385    4.5385
    6.0019    6.0019
    7.9304    7.9304
```

We can also plot the spline on a finer set of points than `xdata`.

```
>> xval = linspace(-1,2,51)';
>> tval = ppval(T,x);
>> hold on
>> plot(xval,tval,'b-')
>> plot(xdata,tdata,'k*')
>> hold off
>> xlabel('x')
>> ylabel('T(x)')
```

The power of the spline is that we can now work the data in the table like any other function. For example, suppose we wanted to integrate the function.

$$\int_{-1}^{2} T(x)\,dx$$

We can do this using the `integral` function much like we did before

```
>> I = integral(@(x)ppval(T,x),-1,2)
I =
    8.3872
```

The only thing that changes is the calling sequence. We need to use the `@(x)ppval(T,x)` syntax to tell MATLAB that we are integrating with respect to x.

4 Accuracy

The cubic spline is an approximate technique and as such it will have limitations to its accuracy. Can this error be quantified? The answer is yes. We will just give the two main results, but the details can be found in any book on numerical analysis.

a) The accuracy is limited by the raw data. If the raw data is only accurate to n digits then any calculation that uses the spline (like the integral example above) is going to have at most n digits of accuracy.

b) There is also an error in the spline itself. This error is going to be bounded by

$$|\text{Absolute Spline Error in T(x)}| \le Ch^4$$

where C is a constant of moderate size (usually less than 10 or so) and

$$h = \max_{1\le i\le n-1} |x_{i+1} - x_i|\,.$$

is the maximum difference between any 2 successive x values.

c) As the number of data points increases, the error associated with b) decreases.

As an example, the exact value (to 5 digits) of the integral in the previous section is $I_{\text{exact}} = 8.3861$. The spline approximation agrees with the exact value to 3 digits. The raw data is accurate to 5 digits and $h = 0.3$ which gives $h^4 = 0.0081$. The resulting 3 digit accuracy for the integral is consistent with the statements above.