Piecewise Polynomials and Cubic Splines

Math 131: Numerical Analysis

J.C. Meza

March 14, 2024

Section 1

Review

Polynomial Interpolation Error

Theorem (Polynomial Interpolation Error)

Let $n \geq 0$ and $f \in C^n[a, b]$. Suppose we are given $x_0, x_1, \ldots, x_n$ distinct points in $[a, b]$. Then

$$f(x) - p_n(x) = \frac{f^{n+1}(\xi(x))}{(n+1)!}\Psi_n(x), \tag{1}$$

with

$$\Psi_n(x) = \prod_{i=0}^{n}(x - x_i),$$

for $a \leq x \leq b$, where $\xi(x)$ is an unknown between the min and max of $x_0, x_1, \ldots, x_n$ and x.

Practical Tips

1. We can use Equation 1 to give us a sense of how the error behaves throughout the interval. Clearly, at the node points themselves, the error will be zero, but what about the rest of the points in $[a, b]$?
2. When considering the error bounds, the interpolation error is likely to be smaller when evaluated at points close to the middle of the domain or near a node.
3. High degree polynomials with equally spaced nodes are not suitable for interpolation because of oscillatory behavior and dependence on higher order derivatives.
4. Sometimes low-order polynomials with a set of suitably chosen data points may be useful in obtaining approximations of some functions.

Section 2

Piecewise Polynomials

Piecewise Polynomials Motivation

1. Theorem showed that the error bound depended on both the size of the interval as well as the higher derivatives, which could be large.
2. Higher order polynomials oscillate (wiggle) a lot. When fitting data that doesn't have a lot of oscillations the polynomial will not approximate the function well in certain areas of the interval (usually near the ends of the intervals).
3. Data are often only piecewise smooth, but polynomials are infinitely differentiable. Asking a polynomial to fit data that isn't as smooth as itself may not be fair.
4. Changing a single data point could drastically alter the entire interpolant.

What to do?

As we discussed in the practical tips section, it is best to think of using:

1. low-order polynomials,
2. within small intervals,
3. and only think of them as local approximations.

This leads us to think about using an alternative approach, which can be briefly described as:

Idea

Instead of finding one single polynomial to fit all the data find a set of polynomials for different regions within the given interval.

Mathematically

In more detail, this approach can be described as:

1. Divide the interval $[a, b]$ into a set of smaller subintervals (elements)

$$a = t_0 < t_1 < ... < t_r = b.$$

   The t_i are often referred to as break points, or sometimes just ***knots***.

2. Fit a ***low-degree polynomial*** $s_i(x)$ in each of the subintervals $[t_i, t_{i+1}], \quad i = 0, ... r - 1$

3. Patch (glue) the polynomials together so that

$$v(x) = s_i(x), \quad t_i \leq x \leq t_{i+1} \quad i = 0, ... r - 1.$$

Piecewise Linears Example

Suppose we were given the following data points

i	0	1	2	3	4
x	1	2	4	5	6
y	1	1.8	2	1.8	0.5

Consider using the Newton form for a linear polynomial ***within*** each of the sub-intervals.

$$s_i(x) = f(x_i) + f[x_i, x_{i+1}](x - x_i),$$

where

$$t_i \leq x \leq t_{i+1}, \quad 0 \leq i \leq 4.$$

Here note that for now, $t_i = x_i$

Example (cont.)

For example, for $i = 0$, we would have:

$$
\begin{aligned}
s_0(x) &= f(x_0) + f[x_0, x_1](x - x_0), \\
&= 1 + \frac{1.8 - 1}{2 - 1}(x - 1), \\
&= 1 + 0.8(x - 1).
\end{aligned}
$$

Likewise, we would then compute s_1, s_2, s_3, s_4.

Exercise: Compute s_1.

Section 3

Error Analysis

Error Bounds

It turns out to be fairly easy to compute an error bound for this case (we've done most of the heavy lifting in the previous sections already).

First let's provide some notation to help us in this new situation.

Let

$$
\begin{aligned}
n &= r && \text{number of subintervals} \\
t_i &= x_i && \text{knots = data points} \\
h &= \max_{1 \le i \le n} (t_i - t_{i-1}) && \text{maximum subinterval size}
\end{aligned}
$$

Error Bounds (cont.)

Theorem

For any $x \in [a, b]$.

$$\left| f(x) - v(x) \right| \leq \frac{h^2}{8} \max_{a \leq \xi \leq b} f''(\xi),$$

Proof:

First, let's note that for any $x \in [a, b]$, it must lie in some interval, say i, therefore $t_{i-1} \leq x \leq t_i$. We can now apply our Polynomial Interpolation Error theorem, and since we are using linear interpolation, $n = 1$, the bound states that:

$$f(x) - v(x) = \frac{f''(\xi)}{2!}(x - t_{i-1})(x - t_i). \tag{2}$$

Proof (cont.)

Next we note that the maximum of the quantity $(x - t_{i-1})(x - t_i)$ occurs at the point

$$x = \frac{t_{i-1} + t_i}{2}.$$

(Why?)

Therefore we can say that:

$$\begin{aligned}\left|(x - t_{i-1})(x - t_i)\right| &\leq \left(\frac{t_i - t_{i-1}}{2}\right)^2, \\ &\leq \frac{h^2}{4}.\end{aligned}$$

The result now follows by substituting this back into Equation 2.

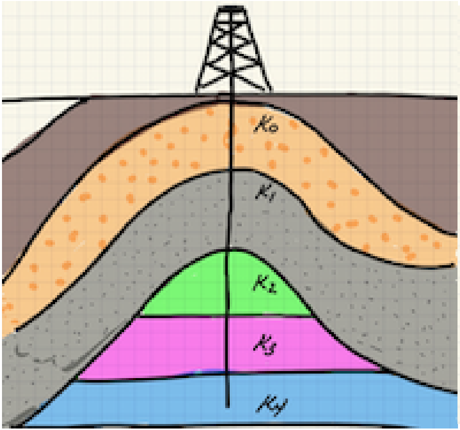
Section 4

Some Special Cases

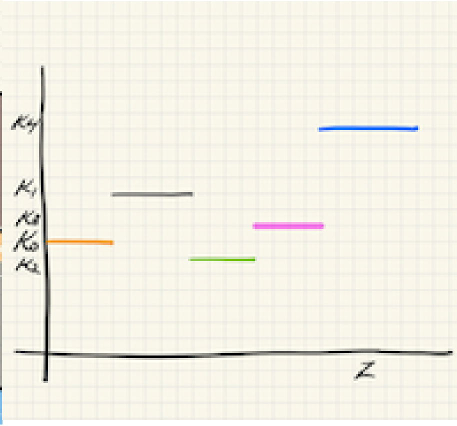
Piecewise Constants

- Before moving to higher-order piecewise interpolation, it might be good to note that sometimes it is useful to consider piecewise constants.
- This approach could be used, for example in applications where the data is known to have discontinuities.
- Consider the case of modeling the subsurface of the earth in oil reservoir and geophysical exploration models.

Oil reservoir modeling



(a) Reservoir Model with different layers



(a) Piecewise constant data for layers

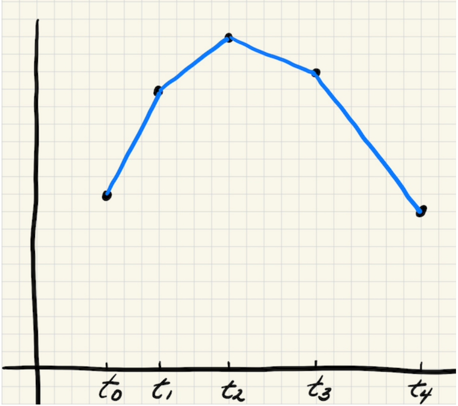
Figure 2: Oil Reservoir Model with piecewise constant data

Section 5

Cubic Splines

Smoothness

Piecewise linear polynomials appear to be a good compromise but they do have one clear disadvantage – the final interpolant will likely have corners at the knots.



Smoothness (cont.)

- What if we want to have a ***smoother*** interpolant?
- This could be quite important if we are trying to approximate a function that is known to have certain smoothness properties, or
- if we are modeling some physical or engineering problem that we wish to have smoothness, such as an airplane wing or a car body.

Cubic Splines

The most popular approach for creating a smooth piecewise interpolant is known as ***cubic splines***.

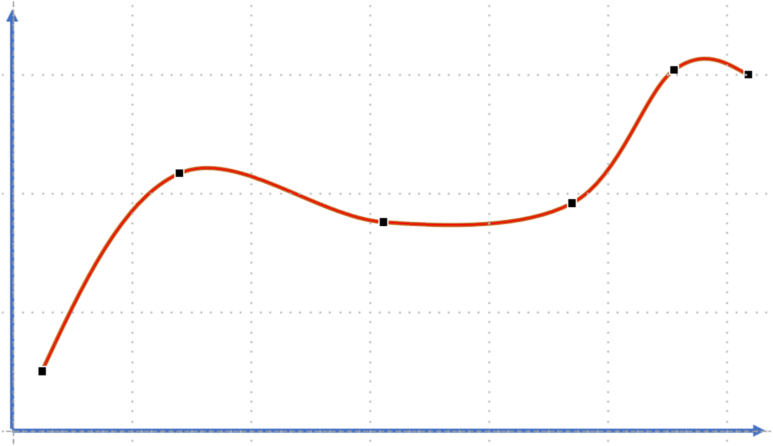


Figure 3: Cubic Spline with 6 data points

Mathematically

Let's consider a cubic interpolant for the ith interval, which we can write as:

$$v(x) = s_i(x) = a_i + b_i(x - t_i) + c_i(x - t_i)^2 + d_i(x - t_i)^3.$$

where

$$t_i \leq x \leq t_{i+1} \qquad i = 0, 1, \ldots, n - 1$$

Note that there are 4 unknowns a_i, b_i, c_i, d_i per sub-interval. If we had n intervals, then there are $4n$ unknowns in total.

That means if we want to have a unique solution, we need to also have $4n$ equations (conditions) specified.

Approach

The usual approach is to generate these equations through a combination of:

1. interpolation conditions, and
2. continuity conditions

Let's first note that in the linear case we had:

$$\begin{aligned} s_i(t_i) &= f(t_i), \qquad & i = 0, 1, \ldots, n-1, \\ s_i(t_{i+1}) &= f(t_{i+1}), & i = 0, 1, \ldots, n-1. \end{aligned} \tag{3}$$

This gave us $2n$ conditions for the $2n$ unknowns. In addition, continuity was implied because

$$s_i(t_{i+1}) = f(t_{i+1}) = s_{i+1}(t_{i+1}).$$

Approach (cont.)

- With cubic splines, we have $4n$ unknowns. ***Why?***
- We can use the interpolating conditions (Equation 3) to give us $2n$ conditions.
- The question before us now is how to choose the additional $2n$ conditions required to give us a unique solution.

Approach (cont.)

Idea

Use remaining $2n$ conditions so as to satisfy $v(x) \in C^2[a, b]$.

- In other words, ensure that each of the splines is twice-continuously differentiable:
  - $s_i(x)$ is continuous at the knots
  - $s_i'(x)$ is continuous at the knots
  - $s_i''(x)$ is continuous at the knots

Mathematically

This idea translates into:

$$
\begin{aligned}
s_i(t_i) &= f(t_i), & i &= 0, 1, \dots, n-1, & &n \text{ conditions} \\
s_i(t_{i+1}) &= f(t_{i+1}), & i &= 0, 1, \dots, n-1, & &n \text{ conditions} \\
s_i'(t_{i+1}) &= s_{i+1}'(t_{i+1}), & i &= 0, 1, \dots, n-2, & &n-1 \text{ conditions} \\
s_i''(t_{i+1}) &= s_{i+1}''(t_{i+1}), & i &= 0, 1, \dots, n-2, & &n-1 \text{ conditions}
\end{aligned}
\tag{4}
$$

- It is important to note that the last two conditions only hold at the internal knots since that is where two splines meet and need to be aligned to maintain continuity of the derivatives.
- Counting up the conditions therefore leaves us with only $4n-2$ conditions.

Approaches

There are two popular approaches to resolving this problem:

1. ***Free boundary*** (natural spline):

$$v''(t_0) = v''(t_n) = 0 \tag{5}$$

2. ***Clamped boundary***:

$$\begin{aligned} v''(t_0) &= f''(t_0), \\ v''(t_n) &= f''(t_n). \end{aligned} \tag{6}$$

Free vs. Clamped B.C. Comparison

Free Bounday

- Easiest to implement and apply.
- Rather arbitrary and there is no *a priori* reason to expect it to be true.

Clamped Boundary

- More realistic,
- Disadvantage of requiring the second derivative of the function.
- If the second derivative is known (or can be approximated) however, this approach would be preferred.

Section 6

Cubic Splines Example

Cubic Splines Example

Suppose we were given the following data points

i	0	1	2
x	0	1	2
y	1.1	0.9	2.0

Compute the cubic spline that interpolates the above data

Recall:

$$v(x) = s_i(x) = a_i + b_i(x - t_i) + c_i(x - t_i)^2 + d_i(x - t_i)^3, \qquad (7)$$

where we have chosen to use $t_i = x_i$, $i = 0, 1$ for this particular case.

Solution

First, let's set up a roadmap of what we need to do:

Given the form for the cubic spline (Equation 7), we need to determine each of 4 coefficients a_i, b_i, c_i, d_i for each of the individual splines, $s_i(x), i = 0, 1$ such that they:

1. Satisfy the interpolation conditions
2. Satisfy the continuity conditions
3. Satisfy the first derivative conditions
4. Satisfy the second derivative conditions

In other words we need to satisfy Equation 3 plus either one of the free or clamped boundary conditions.

Solution (cont.) Interpolation conditions

First note that there are only 3 points, hence we have $n = 2$ intervals. As a result we know that we have $4n = 8$ coefficients that need to be determined and hence we will need 8 equations.

First let's write down what the general form of each of the s_i functions

$$v(x) = \begin{cases} s_0(x) = a_0 + b_0(x - t_0) + c_0(x - t_0)^2 + d_0(x - t_0)^3 & t_0 \leq x < t \\ s_1(x) = a_1 + b_1(x - t_1) + c_1(x - t_1)^2 + d_1(x - t_1)^3 & t_1 \leq x \leq t \end{cases}$$

And plugging in for the actual values of t_0, t_1, we have

$$\begin{aligned} s_0(x) &= a_0 + b_0 x + c_0 x^2 + d_0 x^3 & 0 \leq x < 1 \\ s_1(x) &= a_1 + b_1(x - 1) + c_1(x - 1)^2 + d_1(x - 1)^3 & 1 \leq x \leq 2 \end{aligned} \tag{8}$$

Solution (cont.) Interpolation conditions

We can set up the first 2 equations by using the interpolation conditions: $s_i(t_i) = f(t_i)$.

Setting $x = t_i, i = 0, 1,$ into Equation 8 we get

$$
\begin{aligned}
s_0(0) &= a_0 = 1.1 \\
s_1(1) &= a_1 = 0.9
\end{aligned}
$$

Here we could also have noticed that in each case all the coefficients, except the first one, drop out of the equations for the given value of $x = t_i$.

Solution (cont.) Continuity conditions

Repeating the same procedure and substituting the right-end points of each interval into Equation 8 we have:

$$\begin{aligned} s_0(1) &= a_0 + b_0 + c_0 + d_0 = 0.9 \\ s_1(2) &= a_1 + b_1 + c_1 + d_1 = 2.0 \end{aligned}$$

or substituting the now known values for a_0, a_1 we have:

$$\begin{aligned} s_0(1) &= 1.1 + b_0 + c_0 + d_0 = 0.9 \\ s_1(2) &= 0.9 + b_1 + c_1 + d_1 = 2.0 \end{aligned} \tag{9}$$

At this point continuity is satisfied and we can now look at satisfying the first derivative conditions.

Solution (cont.) First derivative conditions

First let's notice that the only place to apply the derivative conditions is at the one internal know that we have, namely $t_1 = 1$.

Starting with Equation 8 and taking the derivative we have:

$$\begin{aligned} s_0'(x) &= b_0 + 2c_0 x + 3d_0 x^2 & \quad 0 \le x < 1 \\ s_1'(x) &= b_1 + 2c_1(x-1) + 3d_1(x-1)^2 & \quad 1 \le x \le 2 \end{aligned}$$

for which the two equations must be equl at $x = t_1 = 1$, namely

$$s_0'(1) = b_0 + 2c_0 + 3d_0 = s_1'(1) = b_1 + 0 + 0$$

or simplifying

$$b_0 + 2c_0 + 3d_0 = b_1 \tag{10}$$

Solution (cont.) Second derivative conditions

The last step is to enforce continuity of the second derivative at $t_1 = 1$.

Taking the second derivatives:

$$
\begin{aligned}
s_0''(x) &= 2c_0 + 6d_0 x & \qquad 0 \le x < 1 \\
s_1''(x) &= 2c_1 + 6d_1(x-1) & \qquad 1 \le x \le 2
\end{aligned}
$$

for which the two equations must be equl at $x = t_1 = 1$.

Following the same procedure as before leads us to the following equation:

$$
2c_0 + 6d_0 = 2c_1 \tag{11}
$$

That gives us a total of 4 equations. As discussed in class, we now have the freedom to choose either one of the free or clamped boundary conditions to complete the cubic spline.

Solution (cont.) Boundary conditions

To complete the example, let's suppose we choose to use the free boundary conditions.

That means that the second derivatives must equal 0 at both of the end knots, $t_0 = 0, t_2 = 2$

$$
\begin{aligned}
s_0''(0) &= 0 = 2c_0 \\
s_1''(2) &= 0 = 2c_1 + 6d_1
\end{aligned}
$$

That gives us 6 equations with 6 unknowns, so we can solve for all of the remaining coefficients. (Yes, c_0 is actually known in this case, but it won't be in general.)

Complete Solution

If you'd like to complete the example, the final solution is:

$$v(x) = \begin{cases} s_0(x) = 1.1 - 0.525x + 0.325x^3 \\ s_1(x) = 0.9 + 0.45(x-1) + 0.975(x-1)^2 \end{cases}$$

Section 7

Summary

Summary

- ***Idea*** - use piecewise interpolating polynomials within subintervals of the domain as opposed to using one single polynomial over the entire domain.
- Approach has several advantages over using one polynomial including the ability to take into account more of the structure of the problem
- Produces smoother interpolants over the entire region.
- Widely used in practice