Piecewise polynomials and splines

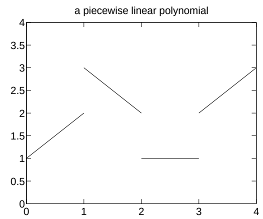
Let $\Delta = \{a = x_0, x_1, \cdots, x_n = b\}$ be a set of distinct points, called *knots* or *nodes* or *breakpoints* or *gridpoints*, partitioning the interval $I = [a, b]$ into n subintervals. W.l.g. assume that the knots are in ascending order.

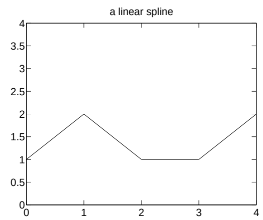
A **piecewise polynomial** (pp) $p(x)$ of degree N, w.r.t. the knots x_i, $i = 0, \cdots, n$, is a polynomial of degree (at most) N on each interval (x_{i-1}, x_i), $i = 1, \cdots, n$.

A polynomial of degree N is always a pp of degree N, but the opposite is not always true.

A pp $p(x)$ of degree N, w.r.t. the knots x_i, $i = 0, \cdots, n$, is called a **spline** (of degree N w.r.t. the knots x_i, $i = 0, \cdots, n$), if $p(x)$ is continuous on the knots and has continuous derivatives up to some order on the knots.

Usually, but not always, the term spline implies continuity up to the $(N-1)$-st derivative. (This is the maximum continuity possible for a pp of degree N.) Sometimes though, the terms pp and spline are used interchangeably.





Piecewise polynomials and splines

A pp $p(x)$ of degree N w.r.t. Δ is often written as a function with n branches.
Each branch shows the form of the pp in one subinterval (x_{i-1}, x_i).
This form is, of course, a polynomial, that is, it can be written as a linear combination of the monomials $1, x, x^2, \cdots, x^N$ with $N+1$ coefficients.
That is, a total of $(N+1)n$ coefficients uniquely define the pp. Thus,

$$p(x) = \begin{cases} a_{01} + a_{11}x + \cdots + a_{N1}x^N & \text{for} \quad x_0 \le x < x_1 \\ a_{02} + a_{12}x + \cdots + a_{N2}x^N & \text{for} \quad x_1 \le x < x_2 \\ \cdots & \cdots \quad \cdots \\ a_{0n} + a_{1n}x + \cdots + a_{Nn}x^N & \text{for} \quad x_{n-1} \le x \le x_n \end{cases}. \tag{1}$$

If continuity conditions are imposed at the interior knots, some coefficients are bound due to these conditions, and fewer coefficients remain free parameters to uniquely define the pp.
For example, if $p(x)$ is continuous at the interior knots, $(N+1)n - (n-1)$ coefficients uniquely define the pp.

Piecewise polynomials and splines -- dimension of pp space

In general, a pp of degree N w.r.t. Δ with continuous derivatives up to order k is uniquely defined by $d = (N+1)n - (k+1)(n-1)$ coefficients.
We then say that the **dimension** of the space $\mathbf{P}^N_{\Delta,k}$ of pps of degree N w.r.t. Δ with continuous derivatives up to order k is d.

The dimension of maximum continuity (smooth) splines of degree N w.r.t. Δ is $d = (N+1)n - N(n-1) = n + N$.

Piecewise polynomials and splines -- Example of dimension

Example: A piecewise linear polynomial w.r.t. Δ has the form

$$p(x) = \begin{cases} a_{01} + a_{11}x & \text{for} \quad x_0 \le x < x_1 \\ a_{02} + a_{12}x & \text{for} \quad x_1 \le x < x_2 \\ \cdots & \cdots \quad \cdots \\ a_{0n} + a_{1n}x & \text{for} \quad x_{n-1} \le x \le x_n \end{cases} \tag{2}$$

where the coefficients a_{ij}, $i = 0, 1$, $j = 1, \cdots, n$, are free parameters. The dimension of the space $\mathbf{P}^1_{\Delta,-1}$ of linear pps is $2n$.

A linear spline w.r.t. Δ has the same form (2) as above, but the coefficients a_{ij}, $i = 0, 1$, $j = 1, \cdots, n$, satisfy the $n-1$ continuity conditions

$$\begin{array}{c} a_{01} + a_{11}x_1 = a_{02} + a_{12}x_1 \\ a_{02} + a_{12}x_2 = a_{03} + a_{13}x_2 \\ \cdots \\ a_{0n-1} + a_{1n-1}x_{n-1} = a_{0n} + a_{1n}x_{n-1} \end{array}. \tag{3}$$

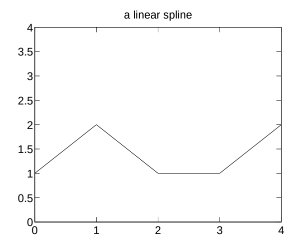
Thus only $2n - (n-1) = n+1$ coefficients are free parameters. The dimension of the space $\mathbf{P}^1_{\Delta,0}$ of linear splines is $n+1$. When writing a linear spline in the form shown in (2), we can use $x_{i-1} \le x \le x_i$ instead of $x_{i-1} \le x < x_i$ in all branches. However, this does not hold for the derivative of a linear spline.

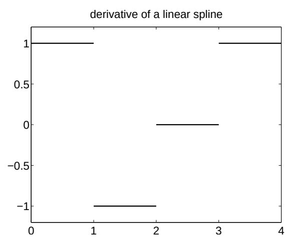
Piecewise polynomials and splines -- derivative

The derivative of a pp is formed by differentiating each branch of the pp. That is,

$$p'(x) = \begin{cases} a_{11} + 2a_{11}x + \cdots + Na_{N1}x^{N-1} & \text{for} \quad x_0 \le x < x_1 \\ a_{12} + 2a_{12}x + \cdots + Na_{N2}x^{N-1} & \text{for} \quad x_1 \le x < x_2 \\ \cdots & \cdots \quad \cdots \\ a_{1n} + 2a_{1n}x + \cdots + Na_{Nn}x^{N-1} & \text{for} \quad x_{n-1} \le x \le x_n \end{cases} \tag{4}$$

Higher derivatives of the pp are formed in a similar way. Note that, even if the pp happens to be continuous on the knots, the derivative may not be so. But it can always be defined so that it is continuous from one side of each knot. (In the above, p and p' are continuous from the right.)





Linear spline interpolation

Given a partition $\Delta = \{a = x_0, x_1, \cdots, x_n = b\}$ of $[a, b]$ and the data (x_i, y_i), $i = 0, \cdots, n$, the linear spline given by

$$L(x) = \left\{ y_{i-1} \frac{x - x_i}{x_{i-1} - x_i} + y_i \frac{x - x_{i-1}}{x_i - x_{i-1}} \text{ for } x_{i-1} \le x \le x_i,\ i = 1, \cdots, n \right. \tag{5}$$

is the unique linear spline w.r.t. Δ interpolating the data.

The linear spline $L(x)$ can be graphed by graphing the broken line that connects the points (x_i, y_i), $i = 0, \cdots, n$.

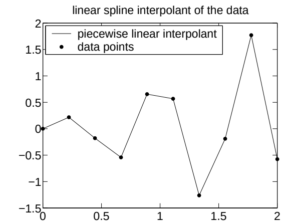
Notes:

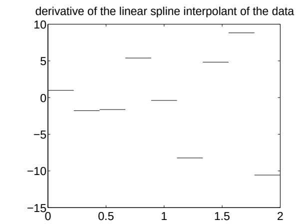
- The form of $L(x)$ in each subinterval $[x_{i-1}, x_i]$ is the Lagrange form of the polynomial of degree at most 1, interpolating (x_{i-1}, y_{i-1}) and (x_i, y_i).
- It is easy to see that $L(x)$ is continuous at the knots.
- The derivative $L'(x)$ in the interior of the subintervals is given by

$$L'(x) = \left\{ \frac{y_i - y_{i-1}}{x_i - x_{i-1}} \text{ for } x_{i-1} < x < x_i\ ,\ \ i = 1, \cdots, n. \right. \tag{6}$$

Linear spline interpolation

Note that $L'(x)$ is *piecewise constant*, and since, in general, $\frac{y_i - y_{i-1}}{x_i - x_{i-1}} \ne \frac{y_{i+1} - y_i}{x_{i+1} - x_i}$, $L'(x)$ is discontinuous at the knots.





Error of the linear spline interpolant

It can be shown that the error of the linear spline interpolant $L(x)$ of a function $f(x) \in C^2$ satisfies

$$|f(x) - L(x)| \le \frac{1}{8} \max_{x \in [a,b]} |f''(x)| \max_{i=1,\cdots,n} (x_i - x_{i-1})^2 \tag{7}$$

for all $x \in [a, b]$.

If $h = \max_{i=1,\cdots,n} \{x_i - x_{i-1}\}$, and f'' is bounded in $[a, b]$, we can write

$$\max_{x \in [a,b]} |f(x) - L(x)| = O(h^2). \tag{8}$$

Linear spline interpolation -- error

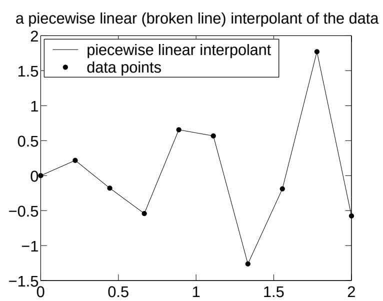
Remarks:

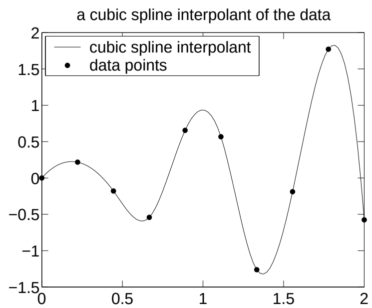
0. If f happens to be a polynomial of degree up to 1, since $f'' = 0$, the error is zero, that is, the interpolant is the function itself.
1. The bound for the error does not involve high derivatives of f. It involves f'', independently of how large n is.
2. Assume f'' is bounded in $[a, b]$. If we double n (i.e. we take about twice as many data points) and halve the stepsizes $x_i - x_{i-1}$, the error bound decreases by a factor of 4. This leads to suggest that the maximum error is expected to decrease by a factor of about 4.
   Thus, it is guaranteed that the bound for the error decreases as n increases, and we can quantify the rate of improvement.
   We say that the linear spline interpolant error is of *second order*. In general, the order of the error is the exponent α of h in the relation $\max |\text{error}| = O(h^\alpha)$.
3. No computation is required to construct a linear spline interpolant.
   For the evaluation of the linear spline interpolant at an arbitrary point $x \in [a, b]$, we need to find which subinterval x belongs to, pick the appropriate branch of $L(x)$, then apply a small ($O(1)$) number of operations to obtain the value of L at x.

Cubic spline interpolation

Interpolating linear splines are easy to construct, and cheap to evaluate. They also give reasonable error bounds and do not suffer from oscillations. However, linear splines have angles (discontinuity of the first derivative). Thus linear splines are not as smooth and visually pleasing as we would like.

To get a smoother interpolant, we try higher degree splines or piecewise polynomials, e.g. piecewise cubics.





Cubic spline interpolation

Cubic splines

A cubic spline w.r.t. $\Delta = \{a = x_0, x_1, \cdots, x_n = b\}$ is defined by

$$C(x) = \begin{cases} C_i(x) \equiv a_i + b_i x + c_i x^2 + d_i x^3 \text{ for } x_{i-1} \le x \le x_i,\ i = 1, \cdots, n, \end{cases} \quad (10)$$

where the coefficients a_i, b_i, c_i, d_i, $i = 1, \cdots, n$, satisfy the continuity conditions

$$C_i(x_i) = C_{i+1}(x_i),\ \ i = 1, \cdots, n-1, \quad (11)$$

$$C_i'(x_i) = C_{i+1}'(x_i),\ \ i = 1, \cdots, n-1, \quad (12)$$

$$C_i''(x_i) = C_{i+1}''(x_i),\ \ i = 1, \cdots, n-1. \quad (13)$$

Relations (11)-(13) are $3(n-1)$ continuity conditions.

Thus, although $C(x)$ involves $4n$ coefficients, there are only $4n - 3(n-1) = n+3$ free parameters in the representation of a cubic spline.

Note that $C'''(x)$ is piecewise constant, and, in general, discontinuous at the knots.

Cubic spline interpolation

Cubic spline interpolants

Assume the data (x_i, y_i), $i = 0, \cdots, n$ are given. In order to construct a cubic spline w.r.t Δ interpolating the data, we need to find $n+3$ linearly independent conditions, so that we determine the $n+3$ free parameters of the cubic spline.

Since we are looking for an *interpolating* cubic spline, some conditions that obviously need to be satisfied are the interpolating conditions

$$C(x_i) = y_i,\ \ i = 0, \cdots, n. \quad (14)$$

These conditions can be shown to be linearly independent. However, they are not enough to uniquely determine a cubic spline (i.e. to uniquely determine the $n+3$ free parameters of the cubic spline).

We need another two conditions. These conditions are often imposed at the endpoints or close to the endpoints, and refereed to as *end-conditions*, or *boundary conditions*.

Cubic spline interpolation -- commonly used end-conditions

(1a) If the derivative values $y_0' = y'(x_0)$ and $y_n' = y'(x_n)$ are given, then the two extra conditions are

$$C'(x_0) = y_0',\ \ C'(x_n) = y_n'. \quad (15)$$

These end-conditions give rise to the so-called *clamped cubic spline interpolant*.

(1b) If the derivative values $y_0' = y'(x_0)$ and $y_n' = y'(x_n)$ are not given, we can first construct approximations to them. For example, to construct an approximation to y_0', first construct a cubic polynomial interpolant of $\{(x_i, y_i),\ i = 0, 1, 2, 3\}$, differentiate it, and evaluate it at x_0. Once approximations $\tilde{y_0'}$ and $\tilde{y_n'}$ to y_0' and y_n', respectively, are available, then the two extra conditions are

$$C'(x_0) = \tilde{y_0'},\ \ C'(x_n) = \tilde{y_n'}. \quad (16)$$

(2) Let the two extra conditions be

$$C''(x_0) = 0,\ \ C''(x_n) = 0. \quad (17)$$

These end-conditions give rise to the so-called *natural cubic spline interpolant* (or *free cubic spline interpolant*). Though these conditions seem to set the second derivative of the spline at the endpoints to an arbitrary value, they are shown to work very well.

Cubic spline interpolation -- commonly used end-conditions

(3) Form the two extra conditions by forcing $C'''(x)$ to be continuous at x_1 and x_{n-1}. That is, the two extra conditions are

$$C_1'''(x_1) = C_2'''(x_1), \;\; C_{n-1}'''(x_{n-1}) = C_n'''(x_{n-1}). \tag{18}$$

These end-conditions give rise to the so-called *not-a-knot cubic spline interpolant*, because they are equivalent to eliminating x_1 and x_{n-1} from the knot sequence (i.e. $C(x)$ is a cubic polynomial in (x_0, x_2) and also a cubic polynomial in (x_{n-2}, x_n)).

Cubic spline interpolation -- error

Error of the cubic spline interpolants

It can be shown that the error of the clamped cubic spline interpolant $C(x)$ of a function $f(x) \in \mathbb{C}^4$ satisfies

$$|f(x) - C(x)| \le \frac{5}{384} \max_{x\in[a,b]} |f^{(4)}(x)| \max_{i=1,\cdots,n} (x_i - x_{i-1})^4 \tag{19}$$

for all $x \in [a, b]$.

If $h = \max_{i=1,\cdots,n} \{x_i - x_{i-1}\}$, and $f^{(4)}$ is bounded in $[a, b]$, we can write

$$\max_{x\in[a,b]} |f(x) - C(x)| = O(h^4). \tag{20}$$

Similar order error bounds hold for the not-a-knot and the natural cubic spline interpolants.

Remarks:

0. If f happens to be a polynomial of degree up to 3, since $f^{(4)} = 0$, the error is zero, that is, the interpolant is the function itself.
1. The bound for the error does not involve high derivatives of f. It involves $f^{(4)}$, independently of how large n is.

Cubic spline interpolation -- error

2. Assume $f^{(4)}$ is bounded in $[a, b]$. If we double n (i.e. we take about twice as many data points) and halve the stepsizes $x_i - x_{i-1}$, the error bound decreases by a factor of 16. This leads to suggest that the maximum error is expected to decrease by a factor of about 16.
   Thus, it is guaranteed that the bound for the error decreases as n increases, and we can quantify the rate of improvement.
   We say that the cubic spline interpolant error is of *fourth order*.
3. It can be shown that the computation to construct a cubic spline interpolant is equivalent to solving an $(n+1)\times(n+1)$ tridiagonal symmetric linear syst., i.e. $O(n)$.
   For the evaluation of the cubic spline interpolant at an arbitrary point $x \in [a, b]$, we need to find which subinterval x belongs to, pick the appropriate branch of $C(x)$, then apply a small ($O(1)$) number of operations to obtain the value of C at x.

Re-cap

- A cubic spline is a cubic pp (thus it can be written in the form (10)) satisfying the continuity conditions (11)-(13).
- A cubic spline interpolant is a cubic spline (i.e. a cubic pp of the form (10)) satisfying, besides the continuity conditions (11)-(13), the interpolating conditions (14), and one type of end-conditions among (15), (16), (17) or (18).

Spline interpolation -- MATLAB

MATLAB's function `spline(x, y, z)` takes as input pairs (x_i, y_i), $i = 0, \cdots, n$, constructs the ($\mathbb{C}^2$) not-a-knot cubic spline interpolant of the data, and returns its values at the points z. If two more y values are given, the first and last values in y are taken as endslopes and the clamped cubic spline interpolant is contructed and evaluated.

MATLAB's function `interp1(x, y, z, 'linear')` takes as input pairs (x_i, y_i), $i = 0, \cdots, n$, constructs the ($\mathbb{C}^0$) linear spline (broken line) interpolant of the data, and returns its values at the points z.

The form `interp1(x, y, z, 'spline')` is equivalent to `spline(x, y, z)`.

Spline interpolation -- MATLAB

Examples: What is the expected output of

```
x  = linspace(0, 2, 100); y  = x.^3;
xi = linspace(0, 2, 5);   yi = xi.^3;
max(abs(spline(xi, yi, x) - y))
>> ans =
  8.8818e-016
```

If the output of

```
x  = linspace(0, 2, 100); y  = x.^4;
xi = linspace(0, 2, 10);  yi = xi.^4;
max(abs(spline(xi, yi, x) - y))
```

is

```
>> ans =
>>  1.6541e-003
```

what is the expected output of

```
xi = linspace(0, 2, 19);  yi = xi.^4;
max(abs(spline(xi, yi, x) - y))
>> ans =
>>  1.0337e-004
```

