Approximation of functions : Interpolation

The main theme here is the following : Given a function $f(x)$, we would like to approximate it by a polynomial. We already know one way to do this, namely using the Taylor polynomial. There are, however, two drawbacks that come to mind concerning this approach.

(i) The Taylor polynomial requires f to be smooth.
(ii) All the information used in constructing the Taylor polynomial comes from values of f and its derivatives at a single point a. Hence the error increases away from a.

In interpolation, on the other hand, we use information of values of f, and possibly its derivative(s), at a number of points called nodes.

To fix ideas, suppose we have a set of $n+1$ distinct points $x_0 < x_1 < \cdots < x_n$ belonging to some interval $[a,b]$ where f is defined.

Defn. Let $f \in C^0[a,b]$ and $n+1$ distinct nodes $a \le x_0 < x_1 < \cdots < x_n \le b$ be given. The Lagrange polynomial interpolant of f on the given nodes is a polynomial $p(x)$ such that

$$p(x_i) = f(x_i), \quad i = 0, \ldots, n.$$

Remark There may not be a function f that is given. Rather, we may only have a set of data $y_0, \ldots, y_n$ corresponding to the nodes, in the same order. In this more general case also we may interpolate the data, i.e. look for a polynomial $p(x)$ satisfying

$$p(x_i) = y_i, \quad i = 0, \ldots, n.$$

Theorem. Let $f \in C^0[a,b]$. Given the $n+1$ distinct nodes $a \le x_0 < x_1 < \cdots < x_n \le b$, there exists a unique Lagrange interpolant p_n of degree $\le n$.

Furthermore, if $f \in C^{n+1}[a,b]$, for any $x \in [a,b]$, there exists $\xi = \xi(x) \in (a,b)$ such that

$$(E) \quad f(x) = p_n(x) + \frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x-x_0)\cdots(x-x_n).$$

proof. We shall use the method of cardinal or basis functions. For $k = 0, 1, \ldots, n$, let $L_{n,k}(x)$ be given by

$$L_{n,k}(x) = \frac{(x-x_0)\cdots(x-x_{k-1})(x-x_{k+1})\cdots(x-x_n)}{(x_k-x_0)\cdots(x_k-x_{k-1})(x_k-x_{k+1})\cdots(x_k-x_n)}.$$

It turns out that the collection $\{L_{n,k}(x)\}_{k=0}^n$ is a basis for P_n, the space of all polynomials in x of degree $\le n$, i.e. every polynomial of degree can be expressed, in a unique way, as a linear combination of $\{L_{n,k}(x)\}_{k=0}^n$.

Indeed, we immediately observe that these functions satisfy

$$L_{n,k}(x_j) = \left\{ \begin{array}{ll} 1 & \text{if } j = k \\ 0 & \text{if } j \neq k. \end{array} \right\} \equiv \delta_{kj} \quad \text{"Kronecker delta"}$$

(see next page for some cases).

Then, we simply define the polynomial $p_n(x)$ by

$$p_n(x) = \sum_{k=0}^n f(x_k) L_{n,k}(x).$$

we see that for any $j \in \{0, 1, \ldots, n\}$

$$p_n(x_j) = \sum_{k=0}^n f(x_k) L_{n,k}(x_j) = \sum_{k=0}^n f(x_k)\delta_{kj} = f(x_j).$$

Thus $p_n(x)$ as defined above, is a Lagrange interpolant.

Also note that degree $P_n \le n$ (since it is a linear combination of polynomials of degree n. On the other hand, it is possible that degree P_n is less than n. Furthermore, we can construct infinitely many Lagrange interpolants of f, there is only one with degree $\le n$. To see this (uniqueness), suppose P_n and $\tilde{P}_n$ are two Lagrange interpolants of degree $\le n$. Let $e(x) = P_n(x) - \tilde{P}_n(x)$. We have

$$e(x_j) = P_n(x_j) - \tilde{P}_n(x_j) = f(x_j) - f(x_j) = 0, \quad j = 0, \dots, n.$$

Now $e(x)$ is a polynomial of degree $\le n$, and has, as seen above $n+1$ distinct roots $(x_0, x_1, \dots, x_n)$. But this is impossible, unless $e(x) \equiv 0$, i.e. $P_n(x) = \tilde{P}_n(x)$ $\forall x$.

It remains to establish the "Error formula" (E). Obviously it holds for $x = x_j$, $j = 0, \dots, n$. So let x be a number in $[a,b]$ but different from the nodes, and fix it. Consider the function $g(t)$ of the variable t defined by

$$g(t) \equiv f(t) - P_n(t) - [f(x) - P_n(x)] \overbrace{(t-x_0)(t-x_1)\cdots(t-x_n)}^{\omega(t)}.$$

Now it is easy to see that g vanishes at the $n+2$ distinct points $x_0, \dots, x_n$ and x. By the generalized Rolle's Theorem there exists $\xi \in (a,b)$ which depends on $x_0, \dots, x_n$ and x also, such that g

$$0 = g^{(n+1)}(\xi) = f^{(n+1)}(\xi) - P_n^{(n+1)}(\xi) - [f(x) - P_n(x)]\, \omega^{(n+1)}(\xi).$$

Now $P_n^{(n+1)}(\xi) = 0$ since degree $P_n \le n$. Also

$\omega(t) = t^{n+1} + c_n t^n + \cdots + c_0$. Hence $\omega^{(n+1)}(t) = (n+1)!$.

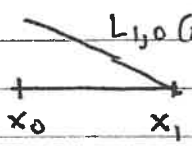
Thus

$$0 = g^{(n+1)}(\xi) = f^{(n+1)}(\xi) - [f(x) - P_n(x)]\,(n+1)!.$$

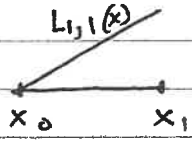
Rearranging this, gives (E). ∎

Examples of Cardinal or basis functions

n=1



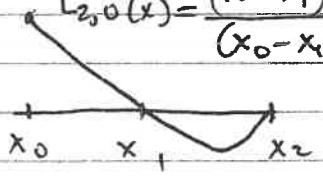
$$L_{1,0}(x) = \frac{x - x_1}{x_0 - x_1}$$



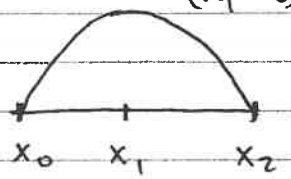
$$L_{1,1}(x) = \frac{x - x_0}{x_1 - x_0}$$

n=2

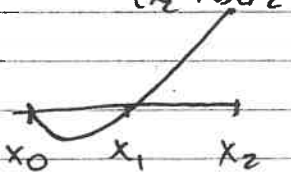
$$L_{2,0}(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)}$$



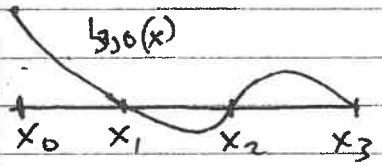
$$L_{2,1}(x) = \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)}$$



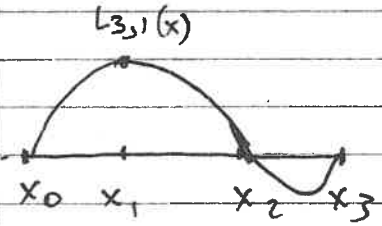
$$L_{2,2}(x) = \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)}$$



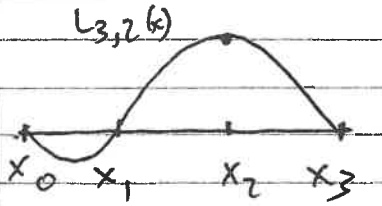
n=3



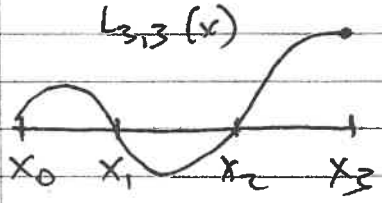
$$L_{3,0}(x) = \frac{(x - x_1)(x - x_2)(x - x_3)}{(x_0 - x_1)(x_0 - x_2)(x_0 - x_3)}$$



$$L_{3,1}(x) = \frac{(x - x_0)(x - x_2)(x - x_3)}{(x_1 - x_0)(x_1 - x_2)(x_1 - x_3)}$$



$$L_{3,2}(x) = \frac{(x - x_0)(x - x_1)(x - x_3)}{(x_2 - x_0)(x_2 - x_1)(x_2 - x_3)}$$



$$L_{3,3}(x) = \frac{(x - x_0)(x - x_1)(x - x_2)}{(x_3 - x_0)(x_3 - x_1)(x_3 - x_2)}$$

Ex (i) Construct the Lagrange interpolant of $f(x)=\cos x$ with $x_0=0$, $x_1=\frac{\pi}{4}$, $x_2=\frac{\pi}{2}$.
(ii) Use it to approximate $\cos\frac{\pi}{3}$. Compute the error.
(iii) Use the Remainder term in (E) to obtain an upper bound for the error.

(i) $\boxed{n=2}$. $L_{2,0}(x)=\dfrac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}=\dfrac{(x-\frac{\pi}{4})(x-\frac{\pi}{2})}{(0-\frac{\pi}{4})(0-\frac{\pi}{2})}=\dfrac{8}{\pi^2}(x-\frac{\pi}{4})(x-\frac{\pi}{2})$

$L_{2,1}(x)=\dfrac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}=\dfrac{(x-0)(x-\pi/2)}{(\frac{\pi}{4}-0)(\frac{\pi}{4}-\frac{\pi}{2})}=\dfrac{-16}{\pi^2}x(x-\frac{\pi}{2})$

$L_{2,2}(x)=\dfrac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}=\dfrac{(x-0)(x-\pi/4)}{(\frac{\pi}{2}-0)(\frac{\pi}{2}-\frac{\pi}{4})}=\dfrac{8}{\pi^2}x(x-\frac{\pi}{4})$.

$\Rightarrow$

$P_2(x)=f(x_0)L_{2,0}(x)+f(x_1)L_{2,1}(x)+f(x_2)L_{2,2}(x)$

$=\overset{=\cos 0}{1}\cdot L_{2,0}(x)+\overset{=\cos\frac{\pi}{4}}{\frac{1}{\sqrt{2}}}L_{2,1}(x)+\overset{=\cos\frac{\pi}{2}}{0}\cdot L_{2,2}(x)$

$=\dfrac{8}{\pi^2}(x-\frac{\pi}{4})(x-\frac{\pi}{2})-\dfrac{8\sqrt{2}}{\pi^2}x(x-\frac{\pi}{2})$.

(ii) $P_2(\frac{\pi}{3})=\dfrac{8}{\pi^2}(\frac{\pi}{3}-\frac{\pi}{4})(\frac{\pi}{3}-\frac{\pi}{2})-\dfrac{8\sqrt{2}}{\pi^2}(\frac{\pi}{3}-\frac{\pi}{2})(\frac{\pi}{3})$

$=-\frac{1}{9}+\frac{4\sqrt{2}}{9}\approx .517428.$

Error $=\left|\frac{1}{2}-.517428\right|\cong\boxed{.017}$

(iii) Remainder $=\dfrac{\cos^{(3)}(\xi)}{3!}(x-x_0)(x-x_1)(x-x_2)$

$\Rightarrow$ Error $=\left|\text{Remainder}(\frac{\pi}{3})\right|=\dfrac{|\cos^{(3)}(\xi(\frac{\pi}{3}))|}{6}\left|\frac{\pi}{3}-0\right|\left|\frac{\pi}{3}-\frac{\pi}{4}\right|\left|\frac{\pi}{3}-\frac{\pi}{2}\right|$

we use the bound $|\cos^{(3)}(\xi(\frac{\pi}{3}))|\le 1$. Hence

Error $\le$ Error bound $\cong\dfrac{1}{6}\cdot\dfrac{\pi}{3}\cdot\dfrac{\pi}{12}\cdot\dfrac{\pi}{6}=\boxed{.024}$

Divided Differences, Newton's form of the Lagrange Interpolant

Let $P_n(x)$ denote the Lagrange interpolant of f on the nodes $a \le x_0 < x_1 < \cdots < x_n \le b$. We write $P_n(x)$ in the so-called Newton's form:

$$P_n(x) = a_0 + a_1(x-x_0) + a_2(x-x_0)(x-x_1) + \cdots + a_n(x-x_0)\cdots(x-x_{n-1})$$

where $a_0, a_1, \ldots, a_n$ are coefficients to be determined. Indeed, using the Lagrange interpolating conditions $P_n(x_j) = f(x_j)$, $j = 0, 1, \ldots, n$ leads to the linear system

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 1 & x_1-x_0 & 0 & \cdots & 0 \\ 1 & x_2-x_0 & (x_2-x_0)(x_2-x_1) & & \vdots \\ \vdots & \vdots & \vdots & \ddots & 0 \\ 1 & x_n-x_0 & (x_n-x_0)(x_n-x_1) & \cdots & \prod_{i=0}^{n-1}(x_n-x_i) \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix}$$

which is lower triangular. The coefficients $a_0, a_1, \ldots, a_n$ are easily solved by Forward Substitution:

```
a_0 = f(x_0)
For i = 1 : n
    prod = 1
    sum = 0
    for j = 0 : i-1
        sum = sum + prod * a_j
        prod = prod * (x_i - x_j)
    }
    a_i = (f(x_i) - sum) / prod
}
```

There is a mathematically equivalent alternative way to calculate the coefficients $a_0, a_1, \cdots, a_n$ in terms of <u>divided differences</u>.

<u>Zeroth (order) divided differences</u>: $f[x_i]$, $i = 0, \cdots, n$

$$f[x_i] \equiv f(x_i), \quad i = 0, 1, \cdots, n$$

<u>First (order) divided differences</u>: $f[x_i, x_{i+1}]$, $i = 0, \cdots, n-1$

$$f[x_i, x_{i+1}] \equiv \frac{f[x_{i+1}] - f[x_i]}{x_{i+1} - x_i}$$

<u>Second (order) divided differences</u>: $f[x_i, x_{i+1}, x_{i+2}]$, $i = 0, \cdots, n-2$

$$f[x_i, x_{i+1}, x_{i+2}] \equiv \frac{f[x_{i+1}, x_{i+2}] - f[x_i, x_{i+1}]}{x_{i+2} - x_i}$$

$$\vdots$$

<u>K-th (order) divided differences</u>:

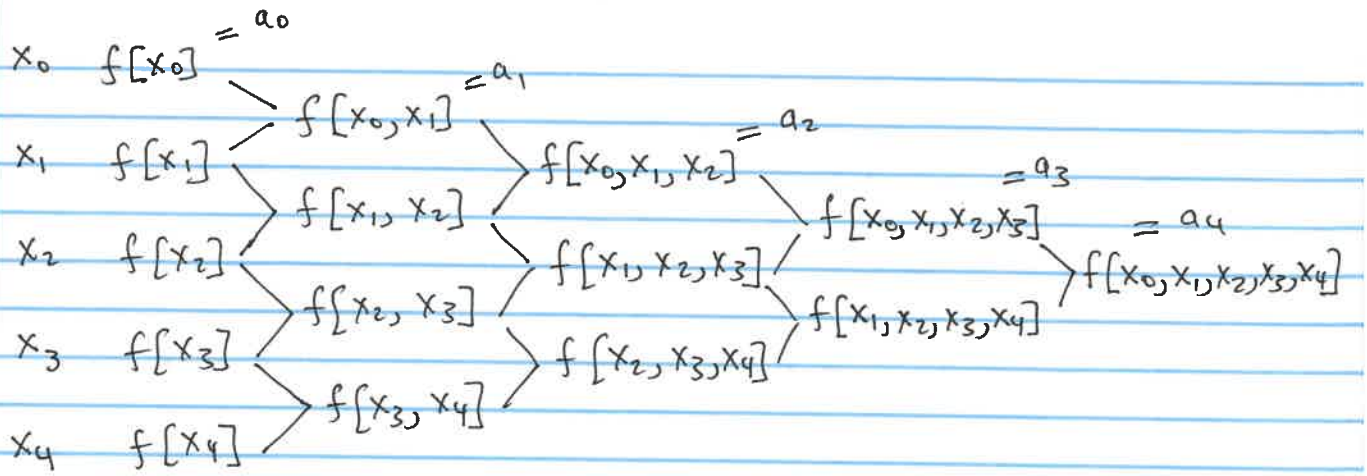
$$f[x_i, x_{i+1}, \cdots, x_{i+k}] \equiv \frac{f[x_{i+1}, \cdots, x_{i+k}] - f[x_i, \cdots, x_{i+k-1}]}{x_k - x_i}$$

<u>Theorem</u> The coefficients $a_0, a_1, \cdots, a_n$ in Newton's form of the Lagrange polynomial interpolant are given by the recursive formula

$$a_k = f[x_0, x_1, \cdots, x_k] = \frac{f[x_1, \cdots, x_k] - f[x_0, \cdots, x_{k-1}]}{x_k - x_0}$$

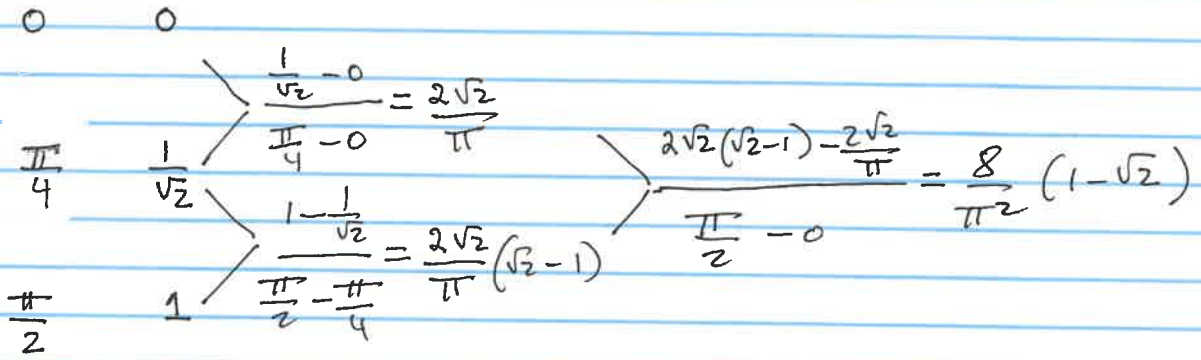
$$k = 0, 1, \cdots, n. \qquad \square$$

<u>Remark</u> In practice, the successive divided differences are calculated in the <u>forward</u> direction as shown in the following table.



More nodes may be added as needed.

Ex. $f(x) = \sin x$, $x_0 = 0$, $x_1 = \frac{\pi}{4}$, $x_2 = \frac{\pi}{2}$



$$\Rightarrow \boxed{P_2(x) = 0 + \frac{2\sqrt{2}}{\pi}(x-0) + \frac{8}{\pi^2}(1-\sqrt{2})\,x\left(x-\frac{\pi}{4}\right)}$$

In particular, let $x = \frac{\pi}{3}$.

$$\Rightarrow P_2\left(\frac{\pi}{3}\right) = \frac{2\sqrt{2}}{\pi}\cdot\frac{\pi}{3} + \frac{8}{\pi^2}(1-\sqrt{2})\frac{\pi}{3}\left(\frac{\pi}{3}-\frac{\pi}{4}\right) = \frac{4\sqrt{2}+2}{9} \approx \boxed{.8507}$$

Exact value: $\sin\frac{\pi}{3} = \frac{\sqrt{3}}{2} = .866025$, $\boxed{\text{Error} \approx 0.015}$

Algorithm: Newton's Divided Differences

Input: Nodes $x_0, x_1, \ldots, x_n$

Function or data values $F_{0,0}; F_{1,0}; \ldots; F_{n,0}$, $F_{n,i} = f(x_i)$

For $i = 1 : n$

For $j = 1 : i$

$$F_{i,j} = \frac{F_{i,j-1} - F_{i-1,j-1}}{x_i - x_{i-j}}, \quad (F_{i,j} = f[x_{i-j}, \ldots, x_i])$$

}

}

OUTPUT: Coefficients $\boxed{a_i = F_{i,i}, \; i = 0, \ldots, n}$

Inverse Interpolation

Used to approximate the inverse function $f^{-1}(x)$ of $f(x)$

=> Approximate a root of $f(x)$.

Using the table corresponding to $y_i = f(x_i)$,

y	y_0	y_1	- - - -	y_n
x	x_0	x_1		x_n

we can form the interpolating polynomial

$y_i = f(x_i)$, $i = 0, \dots, n$

$$p(y) = \sum_{i=0}^{n} c_i \prod_{j=0}^{i-1} (y - y_j) .$$

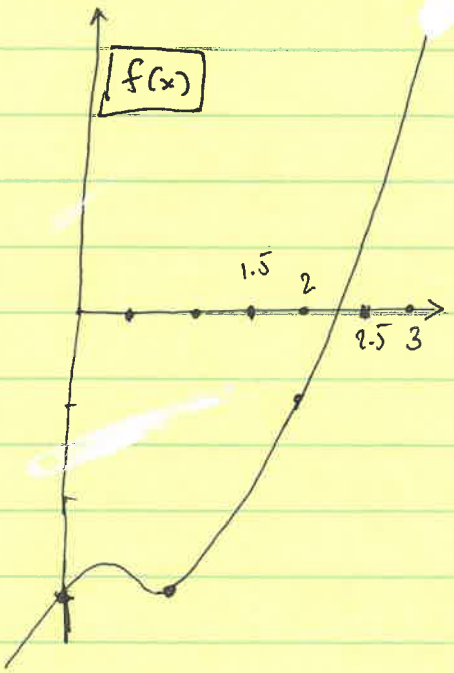
The polynomial $x = p(y)$ approximates the inverse function of $y = f(x)$, if the inverse exists.

Inverse interpolation can be used to approximate a root of $y = f(x)$. Indeed $\boxed{0 = f(x) \Leftrightarrow x = f^{-1}(0) \approx p(0).}$

Ex. $f(x) = x^3 - 2x^2 + x - 3$

⇓

y	-2.625	-1	2.625	9
x	1.5	2	2.5	3



-2.625	1.5			
		.3077		
-1	2		-.032314	
		.1379		.0022697
2.625	2.5		-0.00595	
		.07843		
-9	3			

$$p(y) = 1.5 + .3077(y + 2.625) - 0.032314(y+2.625)(y+1)$$

$$+ 0.0022697(y+2.625)(y+1)(y-2.625) \Rightarrow p(0) = 2.2042$$

"Exact" root 2.17456 rel. Error $\approx$ .03

Hermite interpolation

Given $n+1$ distinct nodes $a \le x_0 < x_1 < \cdots < x_n \le b$ and a function $f \in C^1[a,b]$, the Hermite interpolant of f at the nodes $\{x_i\}_{i=0}^n$ is a polynomial, call it $H(x)$, such that

$$\text{and} \qquad \begin{aligned} H(x_i) &= f(x_i), \quad i = 0, 1, \ldots, n \\ H'(x_i) &= f'(x_i), \quad i = 0, 1, \ldots, n. \end{aligned} \tag{H}$$

This is similar to Lagrange interpolation except that in addition to matching function values, we also match derivative values.

There are $2n+2$ conditions in (H). Hence we expect that a polynomial of degree $\le 2n+1$ should be capable of satisfying these conditions. Indeed, we have

Theorem. Let $f \in C^1[a,b]$. Given $n+1$ distinct nodes $a \le x_0 < x_1 < \cdots < x_n \le b$, there exists a unique polynomial $H(x)$ of degree $\le 2n+1$ that satisfies (H). Furthermore, if $f \in C^{2n+2}[a,b]$, then, given $x \in [a,b]$, there exists $\xi = \xi(x) \in (a,b)$ such that

$$f(x) = H(x) + \frac{f^{(2n+2)}(\xi(x))}{(2n+2)!}(x-x_0)^2 \cdots (x-x_n)^2. \qquad \square$$

There are a variety of ways for constructing the Hermite interpolant. In particular, this can be done using divided differences. Furthermore, the procedure for constructing the Lagrange interpolant can be modified to yield the Hermite interpolant.

Given nodes $x_0, x_1, \ldots, x_n$, we introduce $2n+2$ points $z_0, z_1, \ldots, z_{2n+1}$ essentially by repetition of the nodes $x_0, \ldots, x_n$

$$\begin{array}{cccccccccc} z_0 & z_1 & z_2 & z_3 & \cdots & z_{2n-2} & z_{2n-1} & z_{2n} & z_{2n+1} \\ x_0 & x_0 & x_1 & x_1 & \cdots & x_{n-1} & x_{n-1} & x_n & x_n \end{array}$$

Divided differences $f[z_0], f[z_0, z_1], \ldots, f[z_0, z_1, \ldots, z_{2n+1}]$ are defined, as in the Lagrange case but with a slight adjustment in order to input derivative values of f

Zeroth divided differences: Defined as function values

$$f[z_j] = f(z_j), \quad j = 0, 1, \ldots, 2n+1$$

First Divided Differences: Defined in two distinct ways

$$f[z_i, z_{i+1}] = \begin{cases} f'(z_{2m}) = f'(x_m) & i \text{ is even, i.e. } i = 2m \\ \dfrac{f(z_{i+1}) - f(z_i)}{z_{i+1} - z_i} & i \text{ is odd, i.e. } i = 2m+1 \end{cases}$$

$$= \frac{f(2m+2) - f(2m+1)}{z_{2m+2} - z_{2m+1}} = \frac{f(x_{m+1}) - f(x_m)}{x_{m+1} - x_m}$$

2nd and

Higher-order divided differences are defined as in the Lagrange case:

K-th divided differences

$$f[z_0, z_1, \dots, z_k] = \frac{f[z_1, \dots, z_k] - f[z_0, \dots, z_{k-1}]}{z_k - z_0}.$$

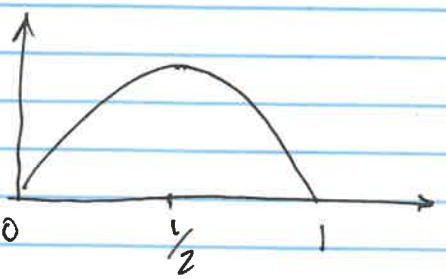
Theorem The Hermite polynomial interpolant of f at the nodes $x_0, \dots, x_n$ can be written in the Newton form

$$H(x) = f[z_0] + f[z_0, z_1](x - z_0) + f[z_0, z_1, z_2](x - z_0)(x - z_1) +$$
$$\dots + f[z_0, \dots, z_{2n+1}](x - z_0) \dots (x - z_{2n}).$$

In terms of the nodes $x_0, \dots, x_n$,

$$H(x) = f[x_0] + f'(x_0)(x - x_0) + f[x_0, x_0, x_1](x - x_0)^2$$
$$+ f[x_0, x_0, x_1, x_1](x - x_0)^2(x - x_1) + \dots$$
$$+ \dots + f[x_0, x_0, \dots, x_n, x_n](x - x_0)^2 \dots (x - x_{n-1})^2(x - x_n)$$

Ex. (i) Construct the Hermite interpolant of $f(x) = \sin(\pi x)$ with nodes $x_0 = 0$, $x_1 = 1/2$, $x_2 = 1$.



(ii) Use it to approximate $\sin \frac{\pi}{4}$. Compute the error.

(iii) Obtain a bound on the error $\left| H\left(\frac{1}{4}\right) - \sin \frac{\pi}{4} \right|$ using the error term.

$f'(x) = \pi \cos \pi x$

0 | 0 | | |

$f'(0) = \pi$

0 | 0 | $4-2\pi$ |

2 | | $-16+4\pi$

$\frac{1}{2}$ | 1 | -4 |

| | $16-4\pi$

$f'(\frac{1}{2}) = 0$ | | 0 | | 0

$\frac{1}{2}$ | 1 | -4 | $16-4\pi$

-2 | | $16-4\pi$

1 | 0 | $4-2\pi$ |

$f'(1) = -\pi$

1 | 0 |

$$\Rightarrow H_5(x) = 0 + \pi(x-0) + (4-2\pi)(x-0)^2 + (-16+4\pi)(x-0)^2(x-\tfrac{1}{2}) + (32-8\pi)(x-0)^2(x-\tfrac{1}{2})^2 + 0(x-0)^2(x-\tfrac{1}{2})^2(x-1)$$

$$\boxed{H_5(x) = \pi x + (4-2\pi)x^2 + (4\pi-16)x^2(x-\tfrac{1}{2}) + (16-4\pi)x^2(x-\tfrac{1}{2})^2.}$$

$$H_5(\tfrac{1}{4}) = \frac{\pi}{4} + (4-2\pi)\frac{1}{16} + (4\pi-16)\frac{1}{16}\left(\frac{1}{4}\right) + (16-4\pi)\frac{1}{16}\frac{1}{16}$$

$$= .70976215$$

$$\text{Error} = \left|\frac{1}{\sqrt{2}} - .70976\right| \approx 2.6 \times 10^{-3}$$

Error bound: $n=2 \Rightarrow 2n+2 = 6$

$$\text{Remainder} = \frac{f^{(6)}(s(\frac{1}{4}))}{6!}(x-0)^2(x-\tfrac{1}{2})^2(x-1)^2$$

$$f^{(6)}(s) = -\pi^6 \sin \pi(s) \Rightarrow |f^{(6)}(s)| \leq \pi^6$$

Error bound is $\frac{\pi^6}{720}\left(\frac{1}{4}-0\right)^2\left(\frac{1}{4}-\frac{1}{2}\right)^2\left(\frac{1}{4}-1\right)^2$

$\approx 2.9 \times 10^{-3}$.

Piecewise Polynomial Interpolation

The remainder/error terms in Lagrange or Hermite interpolation appear to indicate that arbitrarily high accuracies may be obtained by increasing the degree of the interpolant. While this may be true in some cases, there are examples where increasing the degree leads to larger errors.

Indeed consider the example of the Runge function

$$f(x) = \frac{1}{1+x^2} \qquad -5 \le x \le 5.$$

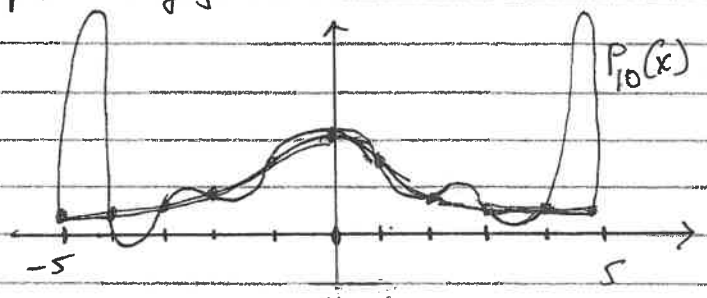
Suppose we take 11 equidistant nodes $x_k = -5 + k$, $k = 0, 1, \ldots, 10$

Let P_n be the Lagrange interpolant of f

It can be shown that

$\lim_{n \to \infty} \|f - P_n\|_\infty = \infty$,

when using $n+1$ uniformly spaced nodes.



The idea behind piecewise polynomial interpolation is to

(i) Subdivide the interval $[a,b]$ into a number of subintervals

(ii) Interpolate f on each subinterval by a polynomial. This gives a number of distinct interpolants. The collection of these interpolants is the piecewise polynomial interpolant of f.

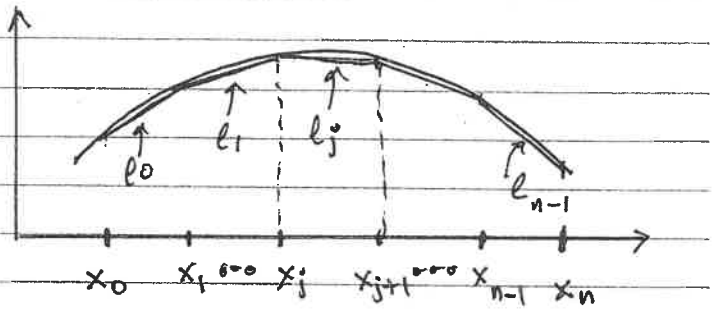
Piecewise linear interpolation

on each subinterval

$[x_j, x_{j+1}]$, $j = 0, 1, \ldots, n-1$

we construct the affine

interpolant, denoted $\ell_j(x)$,
using the two nodes x_j, x_{j+1}.



$a = x_0 < x_1 < \cdots < x_j, x_{j+1} < \cdots < x_n = b$

As an interpolation technique, this is not new

$$\ell_j(x) = f(x_j)\frac{(x - x_{j+1})}{(x_j - x_{j+1})} + f(x_{j+1})\frac{x - x_j}{x_{j+1} - x_j},$$

indeed, we recognize $\frac{x - x_{j+1}}{x_j - x_{j+1}}$ and $\frac{x - x_j}{x_{j+1} - x_j}$ as the basis functions for linear interpolation encountered earlier.

Defn. The collection of the n affine functions $\ell_0, \ldots, \ell_{n-1}$ is the piecewise linear interpolant of f subject to the partition (subdivision) $a = x_0 < \cdots < x_n = b$ of $[a, b]$.

The error estimation formula can be used to estimate the error on each interval: For $x \in [x_j, x_{j+1}]$, there exists $\xi_j(x)$ such that

$$f(x) = \ell_j(x) + \frac{f^{(2)}(\xi_j(x))}{2}(x - x_j)(x - x_{j+1}),$$

assuming of course that $f \in C^2[a, b]$. The remainder term can be used as before to estimate the errors.

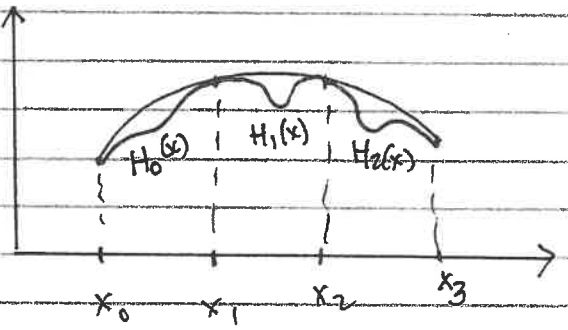
Remark In order to approximate f at some $x \in [a, b]$, we need to find an interval $[x_j, x_{j+1}]$ which contains x and then use $\ell_j(x)$ to approximate $f(x)$. If x happens to be x_k, then there may be two subintervals that could be used.

Piecewise cubic Hermite Interpolation

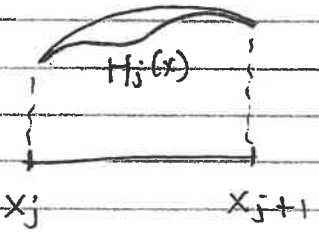
Again we consider a partition

$a = x_0 < x_1 < \cdots < x_j < x_{j+1} < \cdots < x_n = b$

consisting of n subintervals of the form $[x_j, x_{j+1}]$, $j = 0, \cdots, n-1$.



On each subinterval $[x_j, x_{j+1}]$ we shall denote the cubic Hermite interpolant by $H_j(x)$. Indeed, recall that a cubic Hermite interpolant can be constructed uniquely using two nodes, x_j and x_{j+1} here.



The collection of the n cubic Hermite interpolants $H_0(x), \cdots, H_{n-1}(x)$ is the piecewise cubic Hermite interpolant.

Each $H_j(x)$ can be constructed using a variety of ways.

Using Divided differences

$$
\begin{array}{lll}
x_j & f[x_j] & \\
 & & A_j \\
 & f[x_j, x_j] & \\
 & & B_j \\
x_j & f[x_j] & \\
 & & C_j \\
 & f[x_j, x_{j+1}] & \\
 & & D_j \\
x_{j+1} & f[x_{j+1}] & \\
 & & \\
x_{j+1} & f[x_{j+1}] &
\end{array}
\qquad
\begin{array}{l}
\\
f[x_j, x_j] \\
\\
f[x_j, x_j, x_{j+1}] \\
\\
f[x_j, x_j, x_{j+1}, x_{j+1}] \\
\\
f[x_j, x_{j+1}, x_{j+1}] \\
\\
f[x_{j+1}, x_{j+1}]
\end{array}
$$

Newton's form

$$\boxed{H_j(x) = A_j + B_j(x - x_j) + C_j(x - x_j)^2 + D_j(x - x_j)^2(x - x_{j+1})}$$

Note that $f[x_j, x_j] = f'(x_j)$, $f[x_{j+1}, x_{j+1}] = f'(x_{j+1})$.

The other terms are computed as usual differences.

Adopting the general result for Hermite interpolation to each H_j, we have

Thm. Suppose $f \in C^4[a,b]$. Then for each $j = 0, \ldots, n-1$, and $x \in [x_j, x_{j+1}]$, there exists $\xi_j(x) \in (x_j, x_{j+1})$ such that

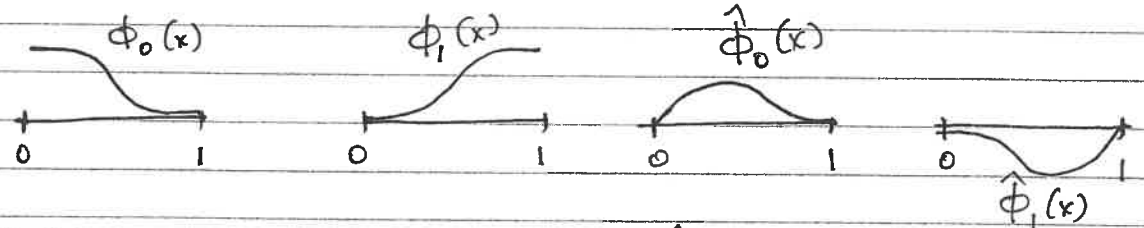
$$\boxed{f(x) = H_j(x) + \frac{f^{(4)}(\xi_j(x))}{4!}(x-x_j)^2(x-x_{j+1})^2.}$$

Construction of piecewise Hermite interpolant using basis functions

We shall next describe an efficient way for constructing the functions $H_0(x), \ldots, H_{n-1}(x)$. This approach consists in

(i) constructing a set of 4 canonical basis functions for cubic Hermite interpolation on the interval $[0,1]$. These functions are denoted by ϕ_0, ϕ_1, $\hat{\phi}_0$ and $\hat{\phi}_1$.

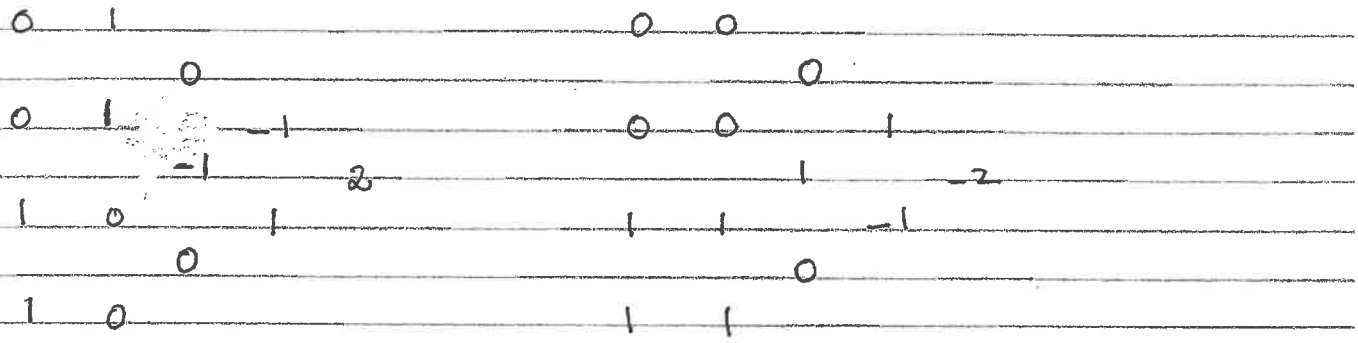
(ii) Construction of $H_j(x)$ by mapping the interval $[0,1]$ into $[x_j, x_{j+1}]$.



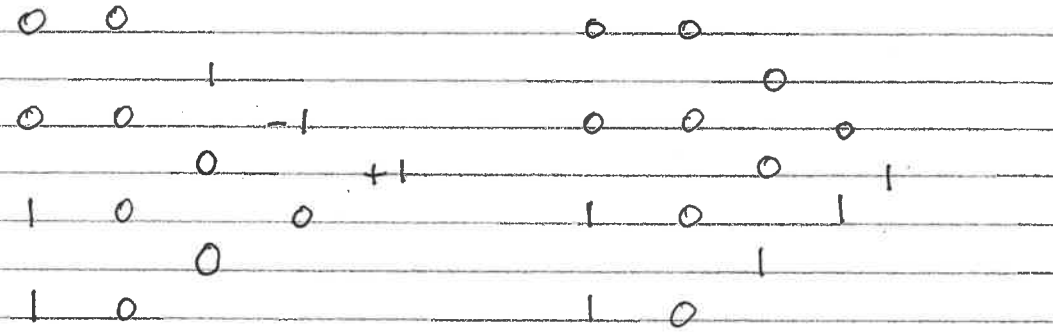
$\phi_0(0) = 1$	$\phi_1(0) = 0$	$\hat{\phi}_0(0) = 0$	$\hat{\phi}_1(0) = 0$
$\phi_0(1) = 0$	$\phi_1(1) = 1$	$\hat{\phi}_0(1) = 0$	$\hat{\phi}_1(1) = 0$
$\phi_0'(0) = 0$	$\phi_1'(0) = 0$	$\hat{\phi}_0'(0) = 1$	$\hat{\phi}_1'(0) = 0$
$\phi_0'(1) = 0$	$\phi_1'(0) = 0$	$\hat{\phi}_0'(1) = 0$	$\hat{\phi}_1'(1) = 1$

These basis functions can be constructed using divided differences and expressed in Newton's form.

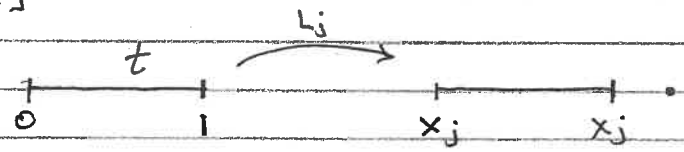
$$\boxed{\phi_0(x) = 1 - x^2 + 2x^2(x-1)} \qquad \boxed{\phi_1(x) = x^2 - 2x^2(x-1)}$$



$$\boxed{\hat{\phi}_0(x) = x - x^2 + x^2(x-1)} \qquad \boxed{\hat{\phi}_1(x) = x^2(x-1)}$$



It is easy to see that the map $t = L_j(x) = x_j + t(\overbrace{x_{j+1} - x_j}^{h_j})$ establishes a bijection between the intervals $[0,1]$ and $[x_j, x_{j+1}]$



Lemma The cubic Hermite interpolant $H_j(x)$ of f on $[x_j, x_{j+1}]$ is given by

$$\boxed{\begin{aligned} H_j(x) = {} & f(x_j)\phi_0\left(\frac{x-x_j}{h_j}\right) + f(x_{j+1})\phi_1\left(\frac{x-x_j}{h_j}\right) \\ & + h_j f'(x_j)\hat{\phi}_0\left(\frac{x-x_j}{h_j}\right) + h_j f'(x_{j+1})\hat{\phi}_1\left(\frac{x-x_j}{h_j}\right) \end{aligned}} \quad (*)$$

proof. Note that $x = x_j + t h_j \iff t = \dfrac{x - x_j}{h_j}$, which is the argument used in the above formula.

$$H_j(x_j) = f(x_j)\underbrace{\phi_0(0)}_{=1} + f(x_{j+1})\underbrace{\phi_1(0)}_{0} + h_j f'(x_j)\underbrace{\hat{\phi}_0(0)}_{0} + h_j f'(x_{j+1})\underbrace{\hat{\phi}_1(0)}_{0}$$
$$= f(x_j).$$

$$H_j(x_{j+1}) = f(x_j)\underbrace{\phi_0(1)}_{0} + f(x_{j+1})\underbrace{\phi_1(1)}_{1} + h_j f'(x_j)\underbrace{\hat{\phi}_0(1)}_{0} + h_j f'(x_{j+1})\underbrace{\hat{\phi}_1(1)}_{0}$$
$$= f(x_{j+1}).$$

Also, from the chain rule

$$H_j'(x) = \frac{1}{h_j} f(x_j)\phi_0'\left(\frac{x - x_j}{h_j}\right) + \frac{1}{h_j} f(x_{j+1})\phi_1'\left(\frac{x - x_j}{h_j}\right) + f'(x_j)\hat{\phi}_0'\left(\frac{x - x_j}{h_j}\right)$$
$$+ f'(x_{j+1})\hat{\phi}_1'\left(\frac{x - x_j}{h_j}\right).$$

Hence

$$H_j'(x_j) = \frac{1}{h_j} f(x_j)\underbrace{\phi_0'(0)}_{0} + \frac{1}{h_j} f(x_{j+1})\underbrace{\phi_1'(0)}_{0} + f'(x_j)\underbrace{\hat{\phi}_0'(0)}_{1} + f'(x_{j+1})\underbrace{\hat{\phi}_1'(0)}_{0}$$
$$= f'(x_j).$$

Finally,

$$H_j'(x_{j+1}) = \frac{1}{h_j} f(x_j)\underbrace{\phi_0'(1)}_{0} + \frac{1}{h_j} f(x_{j+1})\underbrace{\phi_1'(1)}_{0} + f'(x_j)\underbrace{\hat{\phi}_0'(1)}_{0} + f'(x_{j+1})\underbrace{\hat{\phi}_1'(1)}_{1}$$
$$= f'(x_{j+1}).$$

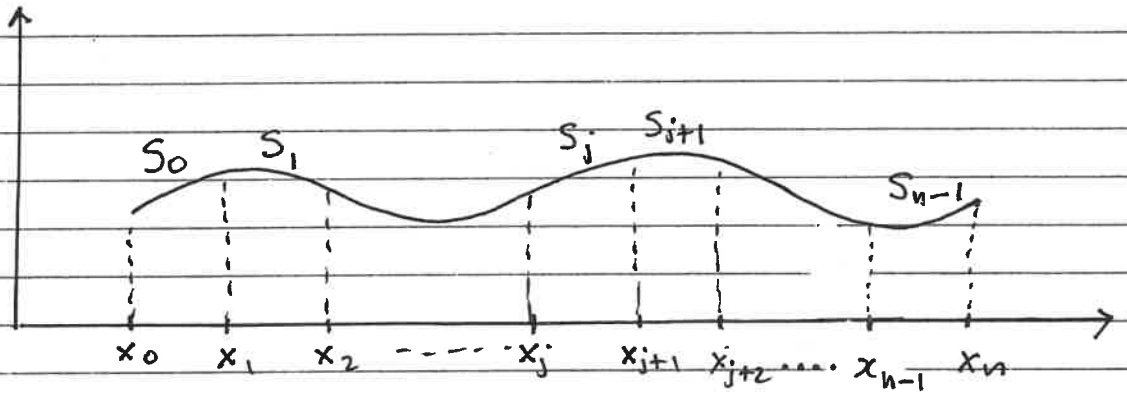
This shows that $H_j(x)$ as defined in (*) has the properties required of the cubic Hermite interpolant of f on $[x_j, x_{j+1}]$ and hence by uniqueness, it is the cubic Hermite interpolant of f on $[x_j, x_{j+1}]$. ∎

Cubic Spline Interpolation

Is an instance of piecewise polynomial (cubic) interpolation over the partition $x_0 < x_1 < \cdots < x_n$. It has similarities to Lagrange and Hermite interpolation:

(i) we match function values at the nodes.

(ii) the derivative values of the interpolants are matched to <u>each other</u> rather than to derivative values of f.



The (global) cubic spline $S(x)$ is the collection of the n cubic polynomials $S_j(x)$, $j = 0, \ldots, n-1$, i.e.

$$S|_{[x_j, x_{j+1}]} = S_j, \quad j = 0, \ldots, n-1.$$

The cubic spline interpolant is constructed in such a way as to satisfy 3 sets of conditions

(I) <u>Internal regularity conditions</u>: We want $S(x)$ be as smooth as possible on $[x_0, x_n]$. It turns out that the maximum smoothness we can achieve without losing the piecewise nature of S is $S \in C^2[a,b]$

(I_1) $\quad S_j(x_{j+1}) = S_{j+1}(x_{j+1}) \qquad j = 0, \ldots, n-2$

(I_2) $\quad S_j'(x_{j+1}) = S_{j+1}'(x_{j+1}) \qquad j = 0, \ldots, n-2$

(I_3) $\quad S_j''(x_{j+1}) = S_{j+1}''(x_{j+1}) \qquad j = 0, \ldots, n-2$

(II) Interpolation conditions : we want S to agree with f at all the nodes

$$S(x_j) = f(x_j) \qquad j = 0, 1, \ldots, n .$$

Counting the number of conditions imposed so far, we see that the number is $3(n-1)+n+1 = \boxed{4n-2}$. On the other hand, we have available $\boxed{4n}$ degrees of freedom, i.e. coefficients 4 for each S_j, which we could use to satisfy the conditions imposed. So we have a surplus of 2 degrees of freedom with which we could satisfy two extra conditions.

One way of achieving this is to impose so-called boundary conditions. It is common to use one or the other of

(III) Natural or Free boundary conditions

(N) $S''(x_0) = S_0''(x_0) = 0$ and $S''(x_n) = S_{n-1}''(x_n) = 0$

(III) Clamped boundary conditions

(C) $S'(x_0) = S_0'(x_0) = f'(x_0)$ and $S'(x_n) = S_{n-1}'(x_n) = f'(x_n)$.

Defn The function $S(x)$ defined by (I), (II) and (III N) is called the Natural or Free cubic spline interpolant of f.

The function $S(x)$ defined by (I), (II) and (III C) is called the clamped cubic spline interpolant of f.

We shall next show that both types of splines are uniquely well-defined. To do this, we write each "piece" S_j in terms of coefficients a_j, b_j, c_j, d_j

(1) $$\boxed{S_j(x) = a_j + b_j(x-x_j) + c_j(x-x_j)^2 + d_j(x-x_j)^3, \quad j = 0, 1, \ldots, n-1}$$

Note that (I) and (II) are common to both types of splines which differ only because of the boundary conditions adopted.

Now let $x = x_j$ in (1). we see that from (II)

$$f(x_j) = S_j(x_j) = a_j \quad , \quad j = 0, 1, \dots, n-1 .$$

At this point we introduce the quantity $a_n \equiv f(x_n)$. Thus, we have

(2) $$\boxed{a_j = f(x_j), \quad j = 0, 1, \dots, n} .$$

Note that a_n is not used in (1), yet it will be used to evaluate the coefficients b_j, c_j, d_j, $j = 0, \dots, n-1$.

From (I_1) we have $S_{j+1}(x_{j+1}) = S_j(x_{j+1})$, $j = 0, 1, \dots, n-2$, hence

(*) $$a_{j+1} = f(x_{j+1}) = a_j + b_j h_j + c_j h_j^2 + d_j h_j^3 \quad , \quad j = 0, 1, \dots, n-2$$

where $$\boxed{h_j = x_{j+1} - x_j \quad , \quad j = 0, \dots, n-1} .$$

Since we defined $a_n = f(x_n)$, we have

$$a_n = f(x_n) = S_{n-1}(x_n) = a_{n-1} + b_{n-1} h_{n-1} + c_{n-1} h_{n-1}^2 + d_{n-1} h_{n-1}^3 .$$

This last equation can be incorporated with (*) to give

(3) $$a_{j+1} = a_j + b_j h_j + c_j h_j^2 + d_j h_j^3 \quad , \quad j = 0, 1, \dots, n-1 .$$

Next, using (I_2) we have for $j = 0, 1, \dots, n-2$

(**) $$b_j + 2c_j h_j + 3d_j h_j^2 = S_j'(x_{j+1}) = S_{j+1}'(x_{j+1}) = b_{j+1} \quad , \quad j = 0, 1, \dots, n-2$$

Defining the "new" quantity b_n by $\boxed{b_n = S'(x_n) = S_{n-1}'(x_n)}$ we see that

$$b_n = S_{n-1}'(x_n) = a_{n-1} + 2c_{n-1} h_{n-1} + 3d_{n-1} h_{n-1}^2$$

we incarporate This with (**) to obtain

(4) $$b_{j+1} = b_j + 2c_j h_j + 3d_j h_j^2, \quad j = 0, 1, \dots, n-1.$$

we next use (I_3) to obtain

(***) $$2c_{j+1} = S_{j+1}''(x_{j+1}) = S_j''(x_{j+1}) = 2c_j + 6d_j h_j, \quad j = 0, 1, \dots, n-2.$$

we extend the range of j to n-1 in the above by defining c_n:

$$c_n = \frac{S''(x_n)}{2} = \frac{S_{n-1}''(x_n)}{2} = c_{n-1} + 3d_{n-1}h_{n-1}.$$

Combining this with (***) we set

(5) $$c_{j+1} = c_j + 3d_j h_j, \quad j = 0, 1, \dots, n-1.$$

This allows solving for d_j in terms of the c_j's:

(6) $$\boxed{d_j = \frac{c_{j+1} - c_j}{3h_j}, \quad j = 0, \dots, n-1}.$$

we can now express the b_j's in terms of $\{a_j, c_j, d_j\}$.

$$b_j = \frac{a_{j+1} - a_j}{h_j} - c_j h_j - d_j h^2, \text{ which simplifies to}$$

(7) $$\boxed{b_j = \frac{a_{j+1} - a_j}{h_j} - \frac{c_{j+1} + 2c_j}{3} h_j, \quad j = 0, 1, \dots, n-1.}$$

It remains to find an equation relating the c_j's. To do this,

(8) $$b_j - b_{j-1} = 2c_{j-1}h_{j-1} + 3d_{j-1}h_{j-1}^2 = (c_j + c_{j-1})h_{j-1}, \quad j = 1, \dots, n,$$

whereas using (7) with j and j-1 we set

(9) $$b_j - b_{j-1} = \frac{a_{j+1} - a_j}{h_j} - \frac{c_{j+1} + 2c_j}{3} h_j - \frac{a_j - a_{j-1}}{h_{j-1}} + \frac{c_j + 2c_{j-1}}{3} h_{j-1}$$
$$j = 1, \dots, n-1.$$

Equating the right hand sides of (8) and (9) for $j = 1, \dots, n-1$, and multiplying both sides by 3, we obtain,

(10) $$h_{j-1} c_{j-1} + 2(h_{j-1} + h_j) c_j + h_j c_{j+1} = \frac{3}{h_j}(a_{j+1} - a_j) - \frac{3}{h_{j-1}}(a_j - a_{j-1}), \quad j = 1, \dots, n-1$$

This constitutes a system of $n-1$ linear equations in the $n+1$ coefficients $c_0, c_1, \dots, c_n$. These are arranged as a matrix vector equation

$$\begin{bmatrix} h_0 & 2(h_0+h_1) & h_1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} & \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-2} \\ c_{n-1} \\ c_n \end{bmatrix} = \begin{bmatrix} \\ \frac{3}{h_1}(a_2 - a_1) - \frac{3}{h_0}(a_1 - a_0) \\ \\ \vdots \\ \frac{3}{h_{n-1}}(a_n - a_{n-1}) - \frac{3}{h_{n-2}}(a_{n-1} - a_{n-2}) \\ \end{bmatrix}$$

The first and last rows are left blank in order to accomodate two additional equations resulting from boundary conditions.

Natural or Free boundary conditions: $S''(x_0) = S''(x_n) = 0$.

$S''(x_0) = 0 \Rightarrow S_0''(x_0) = 0 = 2c_0 \Rightarrow \boxed{c_0 = 0}$.

Recall that we defined $c_n = \frac{S''(x_n)}{2}$. Hence $\boxed{c_n = 0}$

$$\begin{bmatrix} 1 & 0 & \cdots & 0 \\ h_0 & 2(h_0+h_1) & h_1 & \\ & \ddots & \ddots & \ddots \\ & & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\ 0 & \cdots & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{3}{h_1}(a_2-a_1) - \frac{3}{h_0}(a_1-a_0) \\ \vdots \\ \frac{3}{h_{n-1}}(a_n-a_{n-1}) - \frac{3}{h_{n-2}}(a_{n-1}-a_{n-2}) \\ 0 \end{bmatrix}$$

Clamped boundary conditions: $S'(x_0) = f'(x_0)$, $S'(x_n) = f'(x_n)$.

$f'(x_0) = S_0'(x_0) = b_0$.

Also, from (7) we have $b_0 = \frac{a_1 - a_0}{h_0} - \frac{h_0}{3}(c_1 + 2c_0)$.

Hence we have

$$\boxed{h_0(2c_0 + c_1) = \frac{3}{h_0}(a_1 - a_0) - 3f'(x_0)}$$

Now

$f'(x_n) = S'(x_n) = S_{n-1}'(x_n)$

$= b_{n-1} + 2c_{n-1}h_{n-1} + 3d_{n-1}h_{n-1}^2$

$= \frac{a_n - a_{n-1}}{h_{n-1}} + \frac{h_{n-1}}{3}(c_{n-1} + 2c_n)$

upon using (6) and (7). we write this as

$$h_{n-1}c_{n-1} + 2h_{n-1}c_n = 3f'(x_n) - \frac{3}{h_{n-1}}(a_n - a_{n-1}).$$

Hence the completed system in this case is

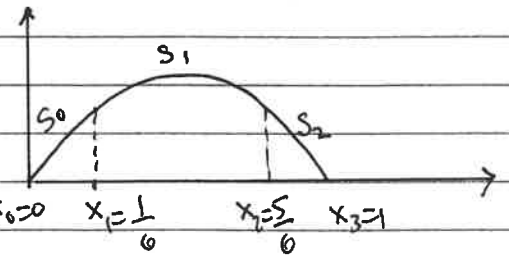
$$\begin{bmatrix} 2h_0 & h_0 & 0 & \cdots & & & 0 \\ h_0 & 2(h_0+h_1) & h_1 & & & & \\ & & & \ddots & & & \\ & & & & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\ 0 & & & \cdots & 0 & h_{n-1} & 2h_{n-1} \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-2} \\ c_{n-1} \\ c_n \end{bmatrix} = \begin{bmatrix} \frac{3}{h_0}(a_1 - a_0) - 3f'(x_0) \\ \frac{3}{h_1}(a_2 - a_1) - \frac{3}{h_0}(a_1 - a_0) \\ \vdots \\ \frac{3}{h_{n-1}}(a_n - a_{n-1}) - \frac{3}{h_{n-2}}(a_{n-1} - a_{n-2}) \\ 3f'(x_n) - \frac{3}{h_{n-1}}(a_n - a_{n-1}) \end{bmatrix}$$

Remark The coefficient matrices for both types of splines are strictly, row diagonally dominant, hence invertible. Therefore the cubic spline for both types of boundary conditions is uniquely defined.

Ex. Calculate the cubic spline interpolant of $\sin \pi x$ on $[0,1]$ using the 4 nodes ($n=3$)

$$x_0=0,\quad x_1=\frac{1}{6},\quad x_2=\frac{5}{6},\quad x_3=1$$

Implement both Natural and Free boundary conditions.



Natural B.C.'s $S''(0)=S''(1)=0$

$$a_0=\sin\pi 0=0$$
$$a_1=\sin\frac{\pi}{6}=1/2$$
$$a_2=\sin\frac{5\pi}{6}=1/2$$
$$a_3=\sin\pi=0$$

$$h_0=\frac{1}{6},\quad h_1=\frac{2}{3},\quad h_2=\frac{1}{6}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ h_0 & 2(h_0+h_1) & h_1 & 0 \\ 0 & h_1 & 2(h_1+h_2) & h_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}=\begin{bmatrix} 0 \\ \frac{3}{h_1}(a_2-a_1)-\frac{3}{h_0}(a_1-a_0) \\ \frac{3}{h_2}(a_3-a_2)-\frac{3}{h_1}(a_2-a_1) \\ 0 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ \frac{1}{6} & \frac{5}{3} & \frac{2}{3} & 0 \\ 0 & 2/3 & \frac{5}{3} & 1/6 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}=\begin{bmatrix} 0 \\ -9 \\ -9 \\ 0 \end{bmatrix} \Rightarrow \begin{aligned} c_0&=0 \\ c_1&=-3.85714 \\ c_2&=-3.85714 \\ c_3&=0 \end{aligned}$$

$$b_j=\frac{a_{j+1}-a_j}{h_j}-\frac{c_{j+1}+2c_j}{3}h_j \qquad d_j=\frac{c_{j+1}-c_j}{3h_j}$$

$$\Rightarrow \begin{aligned} b_0&=3.21429 & d_0&=-7.71429 \\ b_1&=2.57143 & d_1&=0 \\ b_2&=2.57143 & d_2&=7.71429 \\ b_3&=f'(1)=-1 & d_3&\text{ not defined} \end{aligned}$$

These are not used in the spline

$$S_j(x)=a_j+b_j(x-x_j)+c_j(x-x_j)^2+d_j(x-x_j)^3 \qquad j=0,1,2$$

$$S_0(x)=3.21429x-7.71429x^3,\quad 0\le x\le 1/6$$

$$S_1(x)=\frac{1}{2}+2.57143\left(x-\frac{1}{6}\right)-3.85714\left(x-\frac{1}{6}\right)^2,\quad \frac{1}{6}\le x\le\frac{5}{6}$$

$$S_2(x)=\frac{1}{2}+2.57143\left(x-\frac{5}{6}\right)-3.85714\left(x-\frac{5}{6}\right)^2+7.71429\left(x-\frac{5}{6}\right)^3,\quad \frac{5}{6}\le x\le 1$$

Cubic spline with Clamped B.C.

In this case the system is

$$\begin{bmatrix} \frac{1}{3} & \frac{1}{6} & 0 & 0 \\ \frac{1}{6} & \frac{5}{3} & \frac{2}{3} & 0 \\ 0 & \frac{2}{3} & \frac{5}{3} & \frac{1}{6} \\ 0 & 0 & \frac{1}{6} & \frac{1}{3} \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 9-3\pi \\ -9 \\ -9 \\ -3\pi+9 \end{bmatrix} \begin{matrix} \leftarrow \frac{3}{h_0}(a_1-a_0)-3f'(0) \\ \\ \\ \leftarrow 3f'(1)-\frac{3}{h_2}(a_3-a_2) \end{matrix}$$

$a_0 = 0$	$b_0 = 3.14159$	$c_0 = .67847$	$d_0 = -9.16815$
$a_1 = 1/2$	$b_1 = 2.60374$	$c_1 = -3.90560$	$d_1 = 0$
$a_2 = 1/2$	$b_2 = -2.60374$	$c_2 = 3.90560$	$d_2 = 9.16815$
$a_3 = 0$		$c_3 = .67847$	

$$S_0(x) = 3.14159\,x + .67847x^2 - 9.16815x^3$$

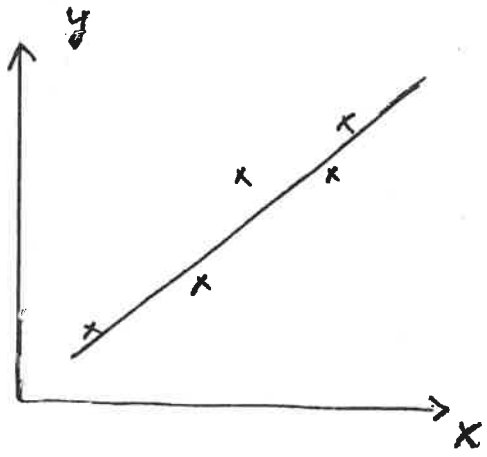
$$S_1(x) = \frac{1}{2} + 2.60374\left(x-\tfrac{1}{6}\right) - 3.90560\left(x-\tfrac{1}{6}\right)^2$$

$$S_2(x) = \frac{1}{2} - 2.60374\left(x-\tfrac{5}{6}\right) - 3.90560\left(x-\tfrac{5}{6}\right)^2 + 9.16815\left(x-\tfrac{5}{6}\right)^3$$

Least-Squares polynomial approximation

The idea here is find a polynomial which approximates a given data $(x_0, y_0), \dots, (x_n, y_n)$ not by interpolation but to minimize a certain measure, or norm, of the error.

Linear Least squares approximation



Given data $(x_0, y_0), \dots, (x_n, y_n)$ we want to construct an affine function

$$y(x) = mx + b$$

which "best fits" the data in the sense that the error

$$E \equiv \sum_{i=0}^{n} (y_i - y(x_i))^2 = \sum_{i=0}^{n} (y_i - (mx_i + b))^2$$

is minimized. Note that E is a function of the unknown coefficients m and b.

To minimize E as a fn. of m and b it is necessary to have (cf. multivariate calculus) that

$$0 = \frac{\partial E}{\partial m} = 2 \sum_{i=0}^{n} (y_i - (mx_i + b))(-x_i)$$

and

$$0 = \frac{\partial E}{\partial b} = 2 \sum_{i=0}^{n} (y_i - (mx_i + b))(-1)$$

This is equivalent to the system of 2 linear equations in the two unknowns m and b

$$\begin{bmatrix} \sum_{i=0}^{n} x_i^2 & \sum_{i=0}^{n} x_i \\ \sum_{i=0}^{n} x_i & \underbrace{\sum_{i=0}^{n} 1}_{n+1} \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^{n} x_i y_i \\ \sum_{i=0}^{n} y_i \end{bmatrix}.$$

It is easy to show that the matrix is invertible thus has a unique solution.

Ex. $n=5$

x_i	1.0	1.1	1.3	1.5	1.9	2.1
y_i	1.84	1.96	2.21	2.45	2.94	3.18

$$\sum_{i=0}^{5} x_i^2 = 14.17, \quad \sum_{i=0}^{5} x_i = 8.9, \quad \sum_{i=0}^{5} x_i y_i = 22.808$$
$$\sum_{i=0}^{5} 1 = 6 \qquad \sum_{i=0}^{5} y_i = 14.58$$

$$\Rightarrow \begin{bmatrix} 14.17 & 8.9 \\ 8.9 & 6 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} = \begin{bmatrix} 22.808 \\ 14.58 \end{bmatrix}$$

$$\begin{bmatrix} m \\ b \end{bmatrix} = \frac{1}{(14.17)6 - (8.9)^2} \begin{bmatrix} 6 & -8.9 \\ -8.9 & 14.17 \end{bmatrix} \begin{bmatrix} 22.808 \\ 14.58 \end{bmatrix}$$

$$\Rightarrow m = 1.20430$$
$$b = .64528$$

Ex. plot data and $1.2043x + .64528$ on same plot and calculate E.

Higher degree polynomial Least squares data fit

We seek a polynomial of degree m that fits best the data $(x_0, y_0), \ldots, (x_n, y_n)$.
Typically we want $m \ll n$.
What happens if $m \geq n$??
The procedure is very similar to the linear fit:

$$P_m(x) = a_m x^m + a_{m-1} x^{m-1} + \cdots + a_1 x + a_0$$

The error at each point x_i is $(y_i - P_n(x_i))^2$
The total error is

$$E = E(a_m, \ldots, a_0) = \sum_{i=0}^{n} \left[y_i - (a_m x_i^2 + \cdots + a_0) \right]^2$$

The necessary conditions to minimize E in terms of the unknown coefficients $a_m, \ldots, a_0$ are

$$\frac{\partial E}{\partial a_m} = \frac{\partial E}{\partial a_{m-1}} = \cdots = \frac{\partial E}{\partial a_0} = 0$$

These give $m+1$ equations in the $m+1$ unknowns $a_m, \ldots, a_0$

$$\frac{\partial E}{\partial a_k} = 2\sum_{i=0}^{n}\left[y_i - (a_m x_i^m + a_k x_i^k + a_0)\right](-x_i^k) = 0,\ k=0,\dots,m$$

$$\Rightarrow \sum_{i=0}^{n}(a_m x_i^m + \cdots + a_1 x_i + a_0)x_i^k = \sum_{i=0}^{n} y_i x_i^k,\ k=0,\dots,m$$

$$\sum_{i=0}^{n}\sum_{j=m}^{0} a_j x_i^{m+k} = \sum_{i=0}^{n} y_i x_i^k,\ j=0,\dots,m$$

$$\sum_{j=m}^{0} a_j \sum_{i=0}^{n} x_i^{m+k} = \sum_{i=0}^{n} y_i x_i^k,\ j=0,\dots,m$$

These can be arranged in the form an $(m+1)\times(m+1)$ linear system

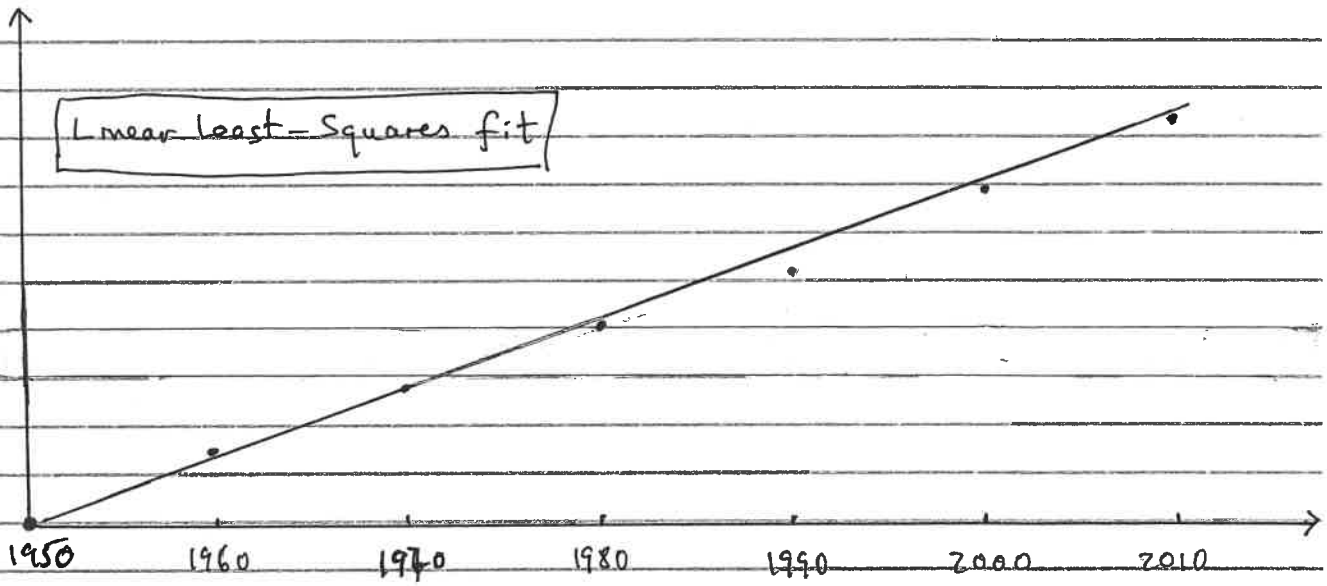
$$\begin{bmatrix} \sum_{i=0}^{n} x_i^{2m} & \sum_{i=0}^{n} x_i^{2m-1} & \cdots & \sum_{i=0}^{n} x_i^{m} \\ \sum_{i=0}^{n} x_i^{2m-1} & \sum_{i=0}^{n} x_i^{2m-2} & & \sum_{i=0}^{n} x_i^{m-1} \\ \vdots & & & \vdots \\ \sum_{i=0}^{n} x_i^{m} & \sum_{i=0}^{n} x_i^{m-1} & & \sum_{i=0}^{n} x_i^{0} \end{bmatrix} \begin{bmatrix} a_m \\ a_{m-1} \\ \vdots \\ a_1 \\ a_0 \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^{n} y_i x_i^m \\ \sum_{i=0}^{n} y_i x_i^{m-1} \\ \vdots \\ \sum_{i=0}^{n} y_i x_i \\ \sum_{i=0}^{n} y_i \end{bmatrix}$$

Ex. $m=2$

$$\begin{bmatrix} \sum_{i=0}^{n} x_i^4 & \sum_{i=0}^{n} x_i^3 & \sum_{i=0}^{n} x_i^2 \\ \sum_{i=0}^{n} x_i^3 & \sum_{i=0}^{n} x_i^2 & \sum_{i=0}^{n} x_i \\ \sum_{i=0}^{n} x_i^2 & \sum_{i=0}^{n} x_i & \sum_{i=0}^{n} 1 \end{bmatrix} \begin{bmatrix} a_2 \\ a_1 \\ a_0 \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^{n} y_i x_i^2 \\ \sum_{i=0}^{n} y_i x_i \\ \sum_{i=0}^{n} y_i \end{bmatrix}$$

As an example where it does not even make sense of talking of a Taylor polynomial, consider the Census data for the U.S. population in millions

Year	1950	1960	1970	1980	1990	2000	2010
population	150.7	179.3	203.4	226.5	248.7	281.4	308.7



$$\begin{bmatrix} 27,445,600 & 13,860 \\ 13,860 & 7 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} = \begin{bmatrix} 3,172,661 \\ 1,598.7 \end{bmatrix} \Rightarrow \begin{cases} m = 2.5839 \\ b = -4,887.8 \end{cases}$$

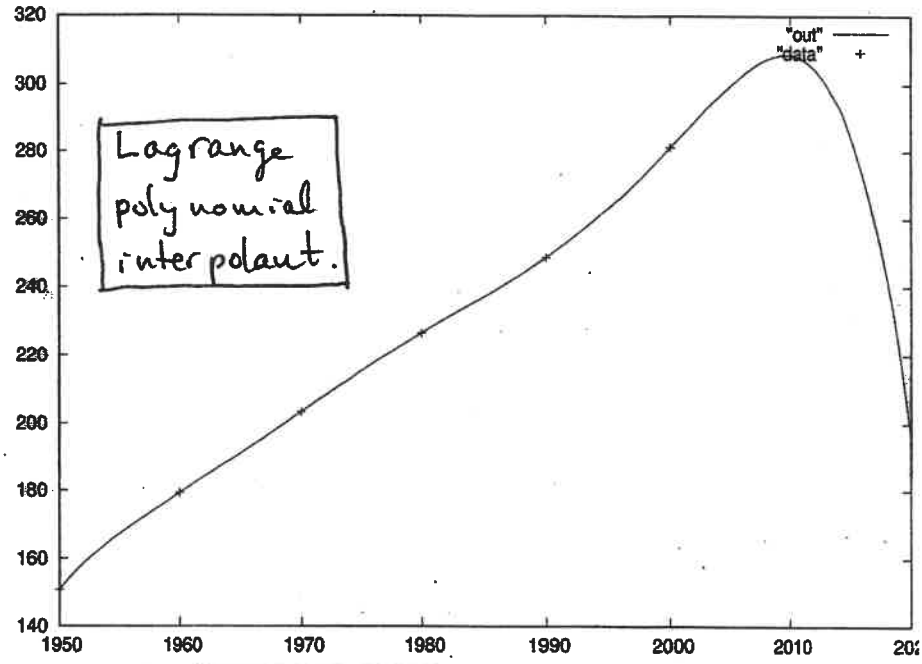
$$\boxed{\ell(t) = 2.5839\,t - 4,887.8}.$$

The error is $E = \left(\sum_{i=0}^{6} \left(y_i - (m t_i + b) \right)^2 \right)^{1/2} \approx 7.1530$

which can be considered as small in the scales present. Indeed, we see that the line is a reasonable fit to the data. We can also use ℓ to "predict" the population in future years, e.g. $\ell(2018) = 326.5$. The actual census figure for 2018 is ≈ 328.

The graph below shows the behaviour of the Lagrange interpolant $P_6(x)$. While it appears "reasonably accurate" in the range [1950, 2010] its predictive potential for any year beyond 2010 appears to be highly suspect.

Indeed, this is yet another example of a cautionary tale about the behaviour of polynomials of degree even moderately high.



Exponential data fit

If it is suspected that the data is exponentially related, then one can try to fit the data with a function of the form

$$y = be^{ax}$$

where the coefficients b and a are determined so that $y = be^{ax}$ "best fits" the data. In this case

$$E(a,b) = \sum_{i=0}^{n} \left[y_i - be^{axi} \right]^2$$

If E is minimized, it is necessary to have

$$\frac{\partial E}{\partial a} = \frac{\partial E}{\partial b} = 0$$

$$0 = \frac{\partial E}{\partial a} = 2 \sum_{i=0}^{n} \left(y_i - be^{axi} \right)\left(-bx_i e^{axi} \right)$$

$$0 = \frac{\partial E}{\partial b} = 2 \sum_{i=0}^{n} \left(y_i - be^{axi} \right)\left(-e^{axi} \right)$$

$$\Rightarrow b^2 \sum_{i=0}^{n} x_i e^{2axi} = b \sum_{i=0}^{n} y_i x_i e^{axi}$$

$$b \sum_{i=0}^{n} e^{2axi} = \sum_{i=0}^{n} y_i e^{axi}$$

A major problem here is that unlike the case of polynomial least squares fit, these equations are <u>nonlinear</u> in a and b and cannot be easily solved.

Instead, we argue as follows:

$$y_i \approx b e^{ax_i} \Rightarrow \ln y_i \approx \ln(b e^{ax_i}) = \ln b + ax_i$$

So we do a linear least-squares fit with the data $(x_i, \ln y_i)$ with the unknown coefficients being $\ln b$ and a. The correspondences are as follows

$b \leftarrow \ln b$
$m \leftarrow a$
$x_i \leftarrow x_i$
$y_i \leftarrow \ln y_i$

$\ln y_i \approx \ln b + a x_i$ Exp.
$y_i \approx b + m x_i$ Lin.

$$\Rightarrow \begin{bmatrix} \sum_{i=0}^{n} x_i^2 & \sum_{i=0}^{n} x_i \\ \sum_{i=0}^{n} x_i & \sum_{i=0}^{n} 1 \end{bmatrix} \begin{bmatrix} a \\ \ln b \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^{n} x_i \ln y_i \\ \sum_{i=0}^{n} \ln y_i \end{bmatrix}$$

Ex.

x	1	1.25	1.50	1.75	2.00
y	5.10	5.79	6.53	7.45	8.46

$$\sum_{i=0}^{4} x_i = 7.50, \quad \sum_{i=0}^{4} x_i^2 = 11.875, \quad \sum_{i=0}^{4} \ln y_i = 9.404$$

$$\sum_{i=0}^{4} x_i \ln y_i = 14.422$$

$$\Rightarrow \begin{bmatrix} 11.875 & 7.50 \\ 7.50 & 5 \end{bmatrix} \begin{bmatrix} a \\ \ln b \end{bmatrix} = \begin{bmatrix} 14.422 \\ 9.404 \end{bmatrix}$$

$$\Rightarrow a = .5056, \quad \ln b = 1.12240$$

$$\Rightarrow b = 3.0722$$

$$\Rightarrow \boxed{y = 3.0722\, e^{.5056x}}$$

x_i	1	1.25	1.50	1.75	2.00
y_i	5.10	5.79	6.53	7.45	8.46
$y(x_i)$	5.0937	5.78	6.559	7.442	8.44

Another choice of an exponential function that is to fit data is

$$\boxed{y = bx^a}$$

Ex. Develop the system of equations for this case