Chapter 3

An Introduction to the Approximation of Functions

In this Chapter, we will look at various ways of approximating functions from a given set of discrete data points.

Interpolation is a method for constructing a function $f(x)$ that fits a known set of data points (x_k, y_k), i.e. a method for constructing new data points within the range of a discrete set of known data points. There are various ways in which this can be done.

Given a sequence of $n+1$ distinct numbers x_k (called *knots*) with corresponding numbers y_k we are looking for a function $f(x)$ such that

$$f(x_k) = y_k \ , \qquad k = 0, 1, ..., n$$

Each pair (x_k, y_k) is called a *data point* and f is called an *interpolant* for the data points.

3.1 Linear Interpolation

Given 2 discrete data points, for instance, (x_0, y_0) and (x_1, y_1), then it is possible to find a unique straight line that fits these two points (Figure 3.1). The unique straight line can found by solving the linear system

$$\begin{aligned} y_0 &= a_1 x_0 + a_0 \ , \\ y_1 &= a_1 x_1 + a_0 \ . \end{aligned}$$

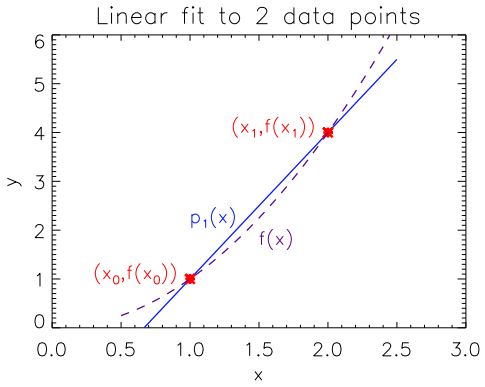


Figure 3.1: Linear approximation (solid blue line) to the 2 data points (red), $(x_0, f(x_0))$ and $(x_1, f(x_1))$, where $f(x)$ is the function given by the purple dashed line.

This pair of simultaneous equations yields the result

$$a_1 = \frac{y_1 - y_0}{x_1 - x_0}\,, \qquad \text{and} \qquad a_0 = y_0 - \frac{(y_1 - y_0)}{(x_1 - x_0)} x_0\,.$$

So the interpolant for any $x \in [x_0, x_1]$ is

$$\begin{aligned} p_1(x) &= \frac{(y_1 - y_0)}{(x_1 - x_0)} x + y_0 - \frac{(y_1 - y_0)}{(x_1 - x_0)} x_0\,, \\ &= y_0 + \frac{(y_1 - y_0)}{(x_1 - x_0)}(x - x_0)\,. \end{aligned}$$

$p_1(x)$ is a first-degree polynomial and if $y_0 = f(x_0)$ and $y_1 = f(x_1)$ the equation above can easily be re-arranged to give

$$\begin{aligned} p_1(x) &= \frac{(x_0 - x_1)}{(x_0 - x_1)} f(x_0) + \frac{(x - x_0)}{(x_0 - x_1)} f(x_0) - \frac{(x - x_0)}{(x_0 - x_1)} f(x_1)\,. \\ &= \frac{(x - x_1)}{(x_0 - x_1)} f(x_0) + \frac{(x - x_0)}{(x_1 - x_0)} f(x_1)\,. \end{aligned} \tag{3.1}$$

3.2 Piecewise Linear Interpolation

Given a known set of $n+1$ discrete data points $(x_k, f(x_k)) \quad k = 0, 1, \ldots, n$ that partition an interval $[a, b]$ into n sub-intervals $[x_k, x_{k+1}]$ where

$$a = x_0 < x_1 < \cdots < x_n = b\,.$$

One of the easiest ways of approximating a given function f in $C[a, b]$ is to form a function connecting consecutive pairs of data points with straight line segments. This is known as *piecewise linear interpolation.*

To find the unique straight line through any pair of consecutive data points $(x_k, f(x_k))$ and $(x_{k+1}, f(x_{k+1}))$ we generalise the formula in (3.1). So the interpolant for any $x \in [x_k, x_{k+1}]$ is equal to

$$p_{1k}(x) \quad = \quad \frac{(x - x_{k+1})}{(x_k - x_{k+1})} f(x_k) + \frac{(x - x_k)}{(x_{k+1} - x_k)} f(x_{k+1}) \, . \tag{3.2}$$

By choosing n sufficiently large, i.e. ensuring the sub-intervals are sufficiently small, we can approximate f as closely as we wish. In general, this interpolating function will be continuous but not differentiable.

Example 3.1.1: *Using the 4 data points given below and find the piecewise linear approximation to these points.*

k	$(x_k, f(x_k))$
0	$(0.0, 0.0)$
1	$(\pi/3, 0.8660)$
2	$(2\pi/3, 0.8660)$
3	$(\pi, 0.0)$

By applying the formula (3.2) to each pair of adjacent data points we find the following 3 straight lines:

$$P(x) = \begin{cases} 0.8270x & x \in [0.0, \pi/3] \\ 0.0000x + 0.8660 & x \in [\pi/3, 2\pi/3] \\ -0.8270x + 2.5981 & x \in [2\pi/3, \pi] \end{cases}$$

This produces the plot given in Figure 3.2

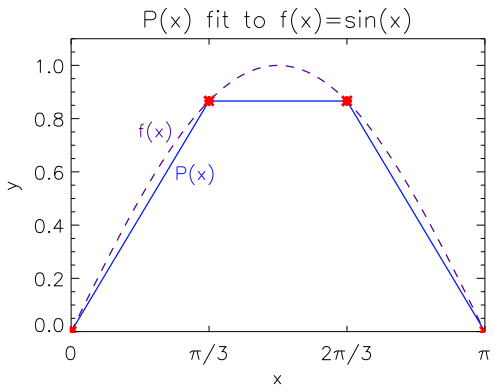


Figure 3.2: (Example 3.1.1) Piecewise linear approximation (solid blue lines) to the 4 data points (red). The purple dotted line is the function that created the data points.

3.2.1 The Linear Interpolation Error

An obvious question is how large is the error of our fit?

The error associated with a linear fit $p_1(x)$ between the points $[a, b]$ to the function $f(x)$ is

$$f(x) - p_1(x) = \frac{1}{2}(x-a)(x-b)f''(u_x) \; .$$

where u_x lies somewhere in the interval $[a, b]$.

Proof

First, we fix $x \in [a, b]$ and use Taylor's theorem to express the values of $f(a)$ and $f(b)$ in terms of $f(x)$, $f'(x)$, $f''(u_a)$ and $f''(u_b)$ where $u_a \in (a, x)$ and $u_b \in (x, b)$. This gives us

$$f(a) = f(x) + (a-x)f'(x) + \frac{1}{2}(a-x)^2 f''(x) + \dots \; , \tag{3.3}$$

$$f(b) = f(x) + (b-x)f'(x) + \frac{1}{2}(b-x)^2 f''(x) + \dots \; , \tag{3.4}$$

From (3.2) we rewrite $p_1(x)$ such that

$$p_1(x) = \frac{x-b}{a-b}f(a) - \frac{x-a}{a-b}f(b) \; ,$$

and substituting in (3.3) for $f(a)$ and (3.4) for $f(b)$ gives

$$\begin{aligned} p_1(x) &= \frac{(x-b)-(x-a)}{a-b}f(x) + \frac{(x-b)(a-x)-(x-a)(b-x)}{a-b}f'(x) + \\ &\quad + \frac{1}{2}\left(\frac{(x-b)(a-x)^2-(x-a)(b-x)^2}{a-b}\right)f''(x) + \dots \; , \\ &= f(x) + \frac{1}{2}\left(\frac{(x-b)(a-x)^2-(x-a)(b-x)^2}{a-b}\right)f''(x) + \dots \; . \end{aligned}$$

However, note that

$$\begin{aligned} \frac{((x-b)(a-x)^2-(x-a)(b-x)^2)}{a-b} &= (x-a)(x-b)\frac{x-a-x+b}{a-b} \; , \\ &= (x-a)(x-b) \; . \end{aligned}$$

So

$$f(x) - p_1(x) = \frac{1}{2}(x-b)(x-a)f''(x) + \dots \qquad x \in [a, b] \; .$$

Now for each $x \in [a, b]$ there exists a $u_x \in [a, b]$ such that

$$\frac{1}{2}(x-b)(x-a)f''(x) + \mathcal{O}(f^{(3)}(x)) = \frac{1}{2}(x-b)(x-a)f''(u_x) \; .$$

In a similar way that the remainder for a Taylor's series is equal to a specific value of the highest-order remaining term.

Thus, the error in the linear interpolant is

$$f(x) - p_1(x) = \frac{1}{2}(x-b)(x-a)f''(u_x) \qquad u_x \in [a,b] \qquad \text{for each } x \in [a,b] \ .$$

□

Note, that when $x = a$ or $x = b$ then the error is exactly zero.

Furthermore, the maximum error in the linear interpolant is given by

$$|f(x) - p_1(x)| \leq \frac{1}{2}|x-b||x-a| \max_{u\in[a,b]} |f''(u)| \qquad x \in [a,b] \ .$$

It is easy to show that

$$|x-b||x-a| \leq \frac{(b-a)^2}{4} \qquad x \in [a,b] \ ,$$

(see tutorial sheet for proof) and hence the maximum linear interpolation error is

$$|f(x) - p_1(x)| \leq \frac{(b-a)^2}{8} \max_{u\in[a,b]} |f''(u)| \ .$$

The overall maximum error of the piecewise linear fit to a sequence of uniformly spaced data points $(x_k, f(x_k)) \quad k = 0, 1, \ldots, n$, with an interval of h between points giving $x_k = hk$ is

$$|f(x) - P(x)| \leq \frac{h^2}{8} \max_{u\in[x_0,x_n]} |f''(u)| \ .$$

Clearly, the error in the piecewise linear fit decreases as h decreases and the number of data points n increases. This is what many graphics packages do - simply draw lines between adjacent points on the screen.

Example 3.2.1: *We return to Example 3.1.1. The function that gave the data points in this example*

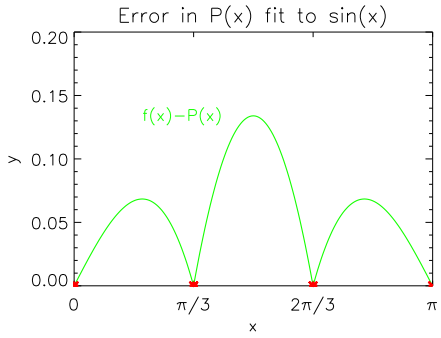


Figure 3.3: (Example 3.2.1) Error of the piecewise linear fit $f(x) - P(x)$.

was

$$f(x) = \sin x \ , \qquad \text{and} \qquad f''(x) = -\sin x \ .$$

Thus, the maximum error on the first interpolant for $x \in [0, \pi/3]$ is

$$|f(x) - P(x)| \leq \frac{(\pi/3 - 0)^2}{8} \max_{u \in [0,\pi/3]} |-\sin(u)| = \frac{(\pi/3)^2}{8} |\sin(\pi/3)| = 0.1187 \; .$$

The maximum errors on the other interpolants are similarly easy to find and are equal to

$$|f(x) - P(x)| \leq \frac{(\pi/3)^2}{8} |\sin(\pi/2)| = 0.1370, \;\; x \in [\pi/3, 2\pi/3] \; ,$$

$$|f(x) - P(x)| \leq \frac{(\pi/3)^2}{8} |\sin(2\pi/3)| = 0.1187, \;\; x \in [2\pi/3, \pi] \; .$$

Thus, the largest overall error for the full piecewise linear fit is

$$|f(x) - P(x)| \leq \frac{(\pi/3)^2}{8} |\sin(\pi/2)| = 0.1370, \;\; x \in [0, \pi] \; .$$

The largest error is actually $1.0 - 0.8660 = 0.1340$ *and so the above estimate is actually an over estimate.*

In Figure 3.3 the error $f(x) - P(x)$ is plotted. Note, that the error is zero at the nodes and the maximum errors for each segment are overly pessimistic.

3.3 Polynomial Interpolation

Piecewise linear interpolation is quick and easy, but it is not very precise if only a few data points are known (n is small). Another disadvantage is that the interpolant is not differentiable at the points x_k and so is does not lead to smooth curves. However, it is possible to generalise linear interpolation to higher-order polynomials to produce a much better fit to the data points.

In piecewise linear interpolation we considered just single pairs of data points in turn. This gave us two equations (one for each data point) and hence two unknowns a_0 and a_1. However, if we consider all the data points in one go we will have $n + 1$ equations and hence can solve for $n + 1$ unknowns.

Given $n+1$ distinct discrete data points $(x_k, f(x_k)) \quad k = 0, 1, \ldots, n$ we can construct the interpolant

$$p_n(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 \; ,$$

which is an n-dimensional polynomial, by solving the equations of the following linear system

$$\begin{bmatrix} x_0^n & x_0^{n-1} & x_0^{n-2} & \ldots & x_0 & 1 \\ x_1^n & x_1^{n-1} & x_1^{n-2} & \ldots & x_1 & 1 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ x_n^n & x_n^{n-1} & x_n^{n-2} & \ldots & x_n & 1 \end{bmatrix} \begin{bmatrix} a_n \\ a_{n-1} \\ \vdots \\ a_0 \end{bmatrix} = \begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix} \; . \tag{3.5}$$

Since all the data points are distinct, the above matrix (known as the Vandermonde matrix) will be non-singular and, hence, there exists a unique polynomial that satisfies,

$$p_n(x_k) = f(x_k), \qquad \forall k = 0, 1, \ldots, n\,.$$

Recall from §1.6 that if the condition number of a matrix is large then the errors in the solution may also be large, i.e. the coefficients a_i of our polynomial may be inaccurate. Even if the condition number is small and we can find $p_n(x)$ to a high degree of accuracy we would like to know how well it fits to the true function, $f(x)$.

3.3.1 The Polynomial Interpolation Error

The error associated with the polynomial fit to the n-dimensional sequence $(x_k, f(x_k))$ compared to the function $f(x)$ can be expressed as

$$f(x) - p_n(x) = \frac{(x - x_0)(x - x_1)\ldots(x - x_n)}{(n+1)!} f^{(n+1)}(u_x)$$

for $x, u_x \in [x_0, x_n]$.

Note, that when $n = 1$ we regain the result for linear interpolation between two data points. This result can be derived in a similar way to the result for linear interpolation, although we do not give a proof here.

It is convenient to write this result as

$$f(x) - p_n(x) = \frac{\prod_{k=0}^{n}(x - x_k)}{(n+1)!} f^{(n+1)}(u_x)\,.$$

$x, u_x \in [x_0, x_n]$.

The maximum error is equal to

$$|f(x) - p_n(x)| = \frac{\max\left(\prod_{k=0}^{n}|x - x_k|\right)}{(n+1)!} \max_{x_0 \le u \le x_n} |f^{(n+1)}(u)|\,. \tag{3.6}$$

Usually, by taking more points (i.e. increasing the degree of the polynomial) the error gets smaller, but this is not always true.

Let us consider some examples:

Example 3.3.1: *Let us return to Example 3.2.1 and see how well 3rd order polynomial interpolation does in comparison.*

We solve the following linear system

$$\begin{bmatrix} 0.0 & 0.0 & 0.0 & 1 \\ (\pi/3)^3 & (\pi/3)^2 & \pi/3 & 1 \\ (2\pi/3)^3 & (2\pi/3)^2 & 2\pi/3 & 1 \\ \pi^3 & \pi^2 & \pi & 1 \end{bmatrix} \begin{bmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix} = \begin{bmatrix} 0.0 \\ 0.8660 \\ 0.8660 \\ 0.0 \end{bmatrix} ,$$

which gives the 3rd order polynomial interpolant

$$p_3(x) = -1.3663 \times 10^{-8} x^3 + -0.3948 x^2 + 1.2405 x - 0.000 \ .$$

We plot this interpolant in Figure 3.4.

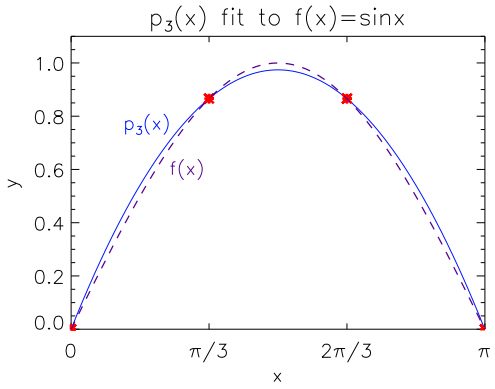


Figure 3.4: (Example 3.3.1) 3rd order polynomial approximation (solid blue lines) to the 4 data points (red). The purple dashed line is the function that created the data points.

Clearly, this fit is considerable better than the piecewise linear fit and thus we expect to find a smaller maximum error:

$$|f(x) - p_3(x)| \leq \frac{\max\left(\prod_{k=0}^{n}(x - x_k)\right)}{4!} \max_{u\in[0,\pi]} |f^4(u)| = \frac{1.2026}{24} |\sin(\pi/2)| = 0.0501 \ .$$

The maximum error is a factor of 2.6 smaller than the maximum piecewise linear error.

Table 3.1 below we consider how the maximum error varies as we increase n, i.e. increase the number of equally spaced data points.

Using the maximum error formula (3.6) to obtain the worst possible error that might occur is always pessimistic but in this case, even a pessimistic estimate decrease quickly with increasing n.

Note, the behaviour of the maximum value of the $n+1$th derivative of $f(x)$

$$f^{(n+1)}(x) = \frac{d^{(n+1)} \sin(x)}{dx^{(n+1)}} = \begin{cases} \sin(x) & n \ - \ odd \\ \cos(x) & n \ - \ even \end{cases}$$

Dimension of polynomial	Interval size, h	maximum error
$n = 3$	$\pi/3$	0.0501
$n = 4$	$\pi/4$	0.0090
$n = 5$	$\pi/5$	0.0014
$n = 6$	$\pi/6$	0.0002

Table 3.1: Error in $p_n(x)$ fits to $f(x) = \sin(x)$.

and hence, for $x \in [0, \pi]$,

$$|f^{(n+1)}(x)| < 1 \qquad \forall n\,.$$

In other words, for all values of n, *the maximum derivative is always the same.*

Example 3.3.2: *Let us now consider* $f(x) = e^{-4x^2}, \quad x \in [-1, 1]\,.$

Here we calculate the polynomial approximations with dimensions $n = 1,\ 2,\ 3,\ 4,\ 5$ *and* 6 *and plot the results in Figure 3.5.*

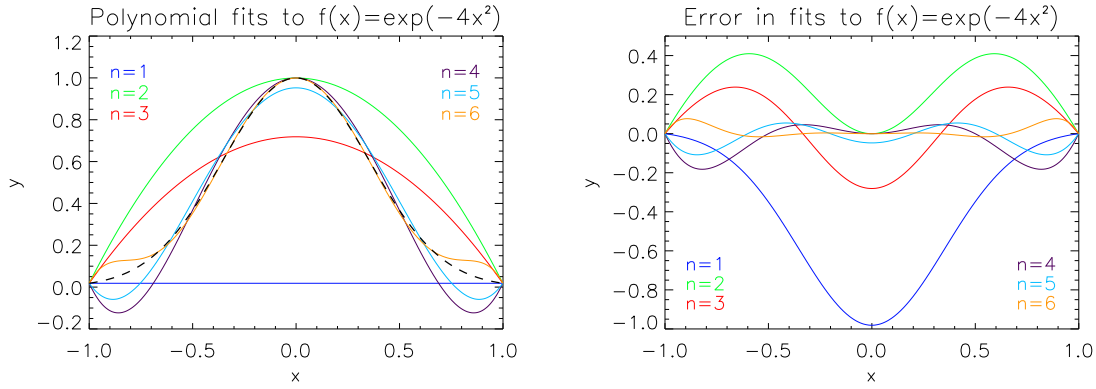


Figure 3.5: (Example 3.2.2) Polynomial approximation for $f(x) = e^{-4x^2}, \quad x \in [-1, 1]$ with $n = 1$ (blue), $n = 2$ (green), $n = 3$ (red), $n = 4$ (purple), $n = 5$ (cyan) and 6 (orange). The black dashed line is the actual function. (b) Error $p_n(x) - f(x)$ of the polynomial fits.

Here, we can see that polynomial interpolation does improve as n *increases, but rather slowly. The errors for these polynomials are given in Table 3.2 Here, we see that the overall error does indeed decrease as* n *increases, but at a slower rate than in the earlier case.*

The reason for this is because as n *increases so does the maximum of the* $n + 1th$ *derivative, as shown below:*

$$f(x) = e^{-4x^2} \qquad \text{and} \qquad f'(x) = -8xe^{-4x^2}$$

So the derivatives important for the errors are

Dimension of polynomial	Interval size, h	maximum error
$n = 1$	2.	4.0000
$n = 2$	2/3	2.0031
$n = 3$	0.5	1.5802
$n = 4$	2/5	0.9899
$n = 5$	1/3	0.7384
$n = 6$	2/7	0.1084

Table 3.2: Error in $p_n(x)$ fits to $f(x) = \exp(-4x^2)$.

- $n = 1$*:* $f''(x) = 8e^{-4x^2}(-1 + 8x^2) \Rightarrow x_{max} = 0$ *and* $|f''(x_{max})| = 8$
- $n = 2$*:* $f^{(3)}(x) : x_{max} = \pm 0.26$ *and* $|f^{(3)}(x_{max})| = 31.23$
- $n = 3$*:* $f^{(4)}(x) : x_{max} = \pm 0.0$ *and* $|f^{(4)}(x_{max})| = 192.0$
- $n = 4$*:* $f^{(5)}(x) : x_{max} = \pm 0.22$ *and* $|f^{(5)}(x_{max})| = 1046.75$
- $n = 5$*:* $f^{(6)}(x) : x_{max} = \pm 0.0$ *and* $|f^{(6)}(x_{max})| = 7680.00$
- $n = 6$*:* $f^{(7)}(x) : x_{max} = \pm -0.97$ *and* $|f^{(7)}(x_{max})| = 12467.00$

The derivatives of $f(x)$ grow in maximum size as n increases.

Example 3.3.3: *In our final example we consider the polynomial approximations to*

$$f(x) = (1 + 25x^2)^{-1}, \qquad x \in [-1, 1]$$

. Here we calculate the polynomial approximations with dimensions $n = 5,\ 9$ and 10 and plot the results in Figure 3.6.

Fitting polynomials to this curve seems much more problematic. Even when $n = 9$ (10 data points) or $n = 10$ (11 data points) the fit is poor. The reason is that the derivatives of $f(x)$ grow in maximum size very quickly. Table 3.3 shows the maximum error found and the size of the maximum value for $|f^{(n+1)}(x)|$.

Dimension of polynomial	maximum error	maximum $n+1$ derivative
$n = 1$	25.	50.
$n = 2$	37.5	584.6
$n = 3$	129.4	15720.7

Table 3.3: Error in $p_n(x)$ fits to $f(x) = (1 + 25x^2)^{-1}$.

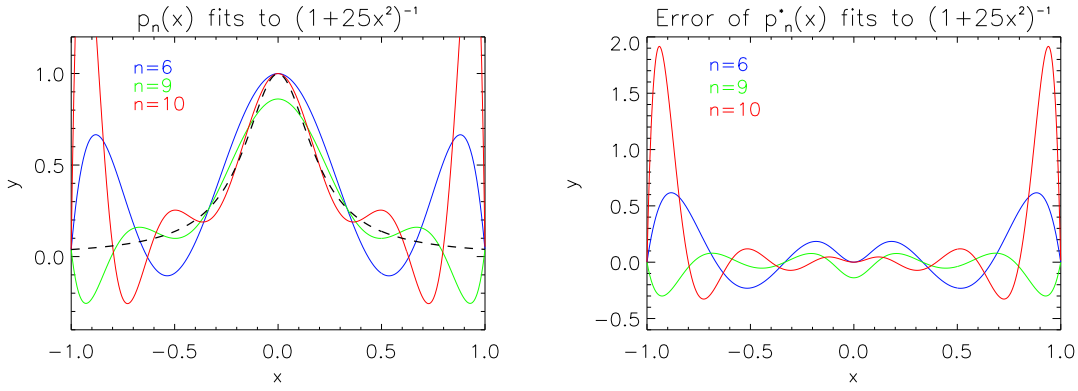


Figure 3.6: (Example 3.2.3) (a) Polynomial approximation for $f(x) = 1/(1+25x^2), \quad x \in [-1,1]$ with $n = 5$ (blue), $n = 9$ (green) and $n = 10$ (red). The black dotted line is the actual function. (b) Error $p_n(x) - f(x)$ of the polynomial fits.

3.3.2 Reducing the Error

The examples illustrate the difficulty of ensuring a "good fit" (with a low degree polynomial).

Recall that the formula for the maximum error in the interpolating polynomial

$$\max_{x_0 \le x \le x_1} |f(x) - p_n(x)| \le \frac{1}{(n+1)!} \max_{x_0 \le x \le x_1} \left| \prod_{k=0}^{n} (x - x_k) \right| \max_{x_0 \le x \le x_1} \left| f^{(k+1)}(x) \right| ,$$

and increasing the degree n will eventually reduce the maximum error. However, if one is using modest values of n, the success will depend on the nature of $f(x)$.

The Weierstrass Theorem states that **there is always a polynomial arbitrarily close to any continuous function.**

i.e. A polynomial which is a "good fit" always exists, but how do you find it, and just how big does n have to be?

For practical reasons, however, it is often better to constraint n to remain small (or modest). With equally spaced points (as in the above examples), we are at the mercy of the derivatives of $f(x)$. There are two possibilities on how to improve our fit with polynomial interpolation without having to resort to large n :

- Try and pick the points $x_0, x_1, \ldots, x_n$ so that the term $\prod(x - x_k)$ is minimised (Chebyshev Polynomials, §3.4).
- Try and find a "piecewise fit" by using a collection of low degree polynomials joined together (similar to the piecewise linear fit §3.2) (Splines §3.5).

3.4 Chebyshev Polynomials

Here, we try to reduce the error in the polynomial approximation by minimising the term

$$\prod_{k=0}^{n}(x - x_k)$$

.

Note, that for any $x \in [-1, 1]$ there exists a θ such that $x = \cos\theta$.

Let us define the set of polynomials $T_n(x) = \cos n\theta$ where $\cos\theta = x$ for $-1 \leq x \leq 1$. These polynomials, called *Chebyshev polynomials*, exist for all x but the definition of $T_n(x)$ only makes sense for $x \in [-1, 1]$.

To determine the form of these polynomials we recall the trigonometric formula

$$\cos(n+1)\theta + \cos(n-1)\theta = 2\cos n\theta\cos\theta\,. \tag{3.7}$$

Note, that we can rewrite the trigonometric identity given in (3.7) as

$$T_{n+1}(x) + T_{n-1}(x) = 2T_n(x)x\,,$$

which can be rearrange to give the recurrence relation:

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)\,. \tag{3.8}$$

The Chebyshev polynomials can be generated by this relation giving

$$\begin{aligned}
T_0(x) &= \cos 0 = 1\\
T_1(x) &= \cos\theta = x\\
T_2(x) &= \cos 2\theta = 2x^2 - 1\\
T_3(x) &= \cos 3\theta = 2x(2x^2-1) - x = 4x^3 - 3x\\
T_4(x) &= \cos 4\theta = 2x(4x^3-3x) - (2x^2-1) = 8x^4 - 8x^2 + 1\\
T_5(x) &= \cos 5\theta = 2x(4x^3-3x) - (2x^2-1) = 16x^5 - 20x^3 + 5x\\
&\text{etc.}
\end{aligned}$$

Observe that if n is even $T_n(x)$ contains even powers of x and if n is odd then $T_n(x)$ contains odd powers of x.

Figure 3.7 shows plots of $T_3(x)$, $T_4(x)$, $T_5(x)$ and $T_6(x)$ for $-1 \leq x \leq 1$. The graphs suggest the following three properties which hold true for all the Chebyshev polynomials:

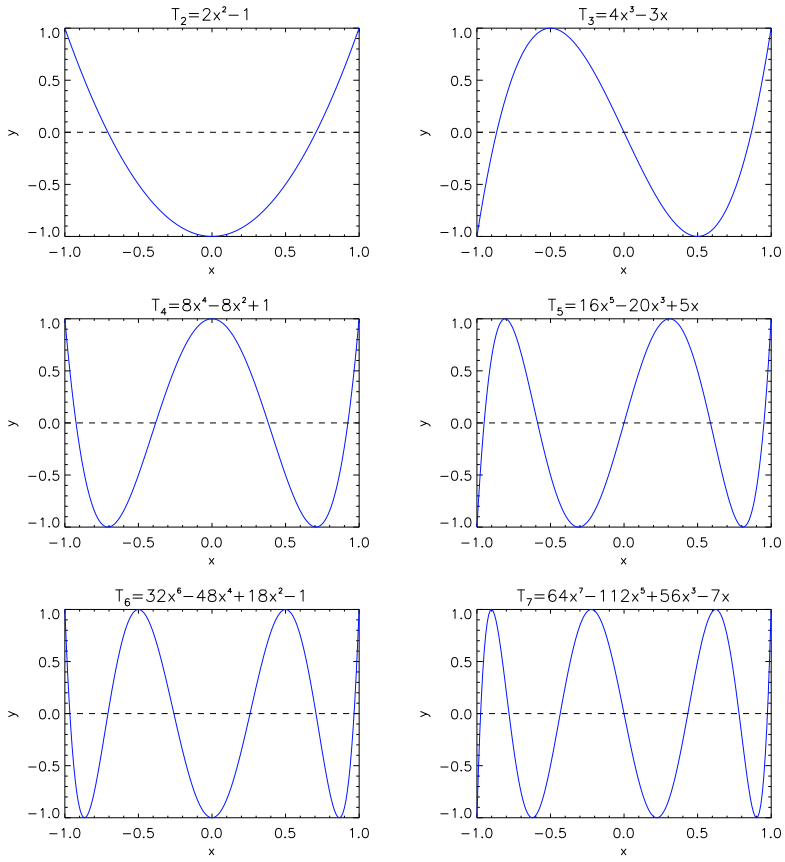


Figure 3.7: Chebyshev polynomials for $n = 2,\ 3,\ 4,\ 5\ , 6$ and $n = 7$.

Properties of the Chebyshev Polynomials:

1. $|T_n(x)| \leq 1$ for $-1 \leq x \leq 1\ \ \forall n$
2. $|T_n|$ attains its maximum value of 1 on $x \in [-1, 1]$ at $n + 1$ points, including both endpoints, and takes the values ± 1 alternately on these points.
3. T_n has n distinct zeros in the interior of $[-1, 1]$.

Proof of Properties

1. **Property 1:** $-1 \leq \cos n\theta \leq 1$, so by definition, $-1 \leq T_n(x) \leq 1$ for $x \in [-1, 1]$.
2. **Property 2:** The maximum of $|T_n(x)| = 1$ since the maximum of $|\cos n\theta| = 1$.

   Now, $\cos n\theta = \pm 1$, when

$$\begin{aligned} n\theta &= k\pi \qquad k = 0, 1, \ldots\ n, \\ \Rightarrow \theta &= \frac{k\pi}{n} \qquad k = 0, 1, \ldots\ n, \end{aligned} \tag{3.9}$$

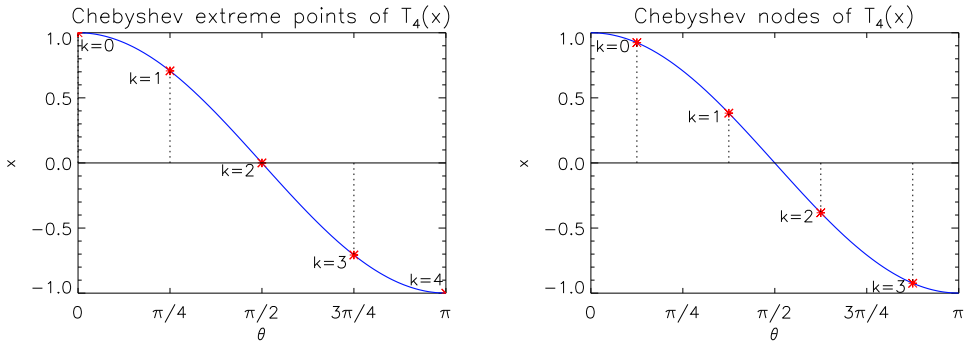


Figure 3.8: The (a) extreme points and (b) nodes of the Chebyshev polynomial $T_4(x)$.

so

$$x = \cos\theta = \cos\left(\frac{k\pi}{n}\right) \qquad k = 0, 1, \ldots n, \,,$$

There are $(n+1)$ values of θ $(0, \pi/n, 2\pi/n, \ldots, \pi)$ for which $|T_n(x)| = |\cos n\theta| = 1$. Hence, there are $(n+1)$ values of $x \in [-1, 1]$ $(x = 1, \cos\frac{\pi}{n}, \cos\frac{2\pi}{n}, \ldots, \cos\frac{(k-1)\pi}{n}, -1)$ at which $|T_n(x)| = 1$. We call this set of $(n+1)$ points the *extreme points* of $T_n(x)$.

The extreme points of the Chebyshev polynomial $T_4(x)$ are shown in Figure 3.8(a).

Furthermore, observe that $\cos n\theta = \cos(k\pi) = (-1)^k$, $k = 0, 1, \ldots, n$. So $\mathbf{T_n(x)}$ oscillates between +1 and -1 at these $(\mathbf{n+1})$ points.

3. **Property 3:** Obviously, between each maximum and minimum of +1 and -1 is a zero, and hence, there are n zeros in [-1,1], given by

$$\begin{aligned} &\cos n\theta = 0\,, \\ &\Rightarrow n\theta = (2k+1)\frac{\pi}{2}, \qquad k = 0, 1, 2, \ldots, (n-1)\,, \\ &\Rightarrow \theta = (2k+1)\frac{\pi}{2n}, \qquad k = 0, 1, 2, \ldots, (n-1)\,. \end{aligned}$$

In terms of x let use denote the kth zero as

$$x_k = \cos\theta = \cos\left((2k+1)\frac{\pi}{2n}\right) \qquad k = 0, 1, 2, \ldots, (n-1)\,.$$

The zeros (or nodes) of the Chebyshev polynomial $T_4(x)$ are shown in Figure 3.8(b).

Observe that $x_0 = \cos\left(\frac{\pi}{2n}\right)$ and $x_{n-1} = \cos\left(\frac{(2n-2+1)\pi}{2n}\right) = \cos\left(\pi - \frac{\pi}{2n}\right)$, which implies

$$x_0 = -x_{n-1}\,.$$

So in general, we have

$$x_k = -x_{n-1-k},$$

i.e. the zeros are placed symmetrically about $x = 0$ in $[-1, 1]$.

3.4.1 Minimising the Error Bound

The objective of the following section is to minimise $\max\{|\prod_{k=0}^{n}(x-x_k)|\}$ over $x_0 \le x \le x_n$ by selecting suitable nodes $x_0, x_1, \ldots, x_n$ for interpolation.

Consider $\prod(x-x_k)$ for $-1 \le x \le 1$,

$$\prod_{k=0}^{n}(x-x_k) = (x-x_0)(x-x_1)\ldots(x-x_n), \qquad x_k \in [-1,1]$$

$$= x^{n+1} - x^n \sum_{k=0}^{n} x_k + \cdots + x_0 x_1 x_2 \ldots x_n \,,$$

i.e. a polynomial of degree $n+1$ with leading coefficient 1. This polynomial has $n+1$ roots (zeros), namely the nodes $x_0,\ \ldots,\ x_k,\ \ldots,\ x_n$.

Now from §3.4, we know that $T_{n+1}(x)$ is a polynomial of degree $(n+1)$ with $(n+1)$ zeros in $[-1,1]$ and

$$|T_{n+1}(x)| \le 1 \qquad \forall x \in [-1,1]\,.$$

From the recurrence relation $T_{n+1} = 2xT_n - T_{n-1}$ we can see that

$$T_{n+1}(x) = 2^n x^{n+1} + \ldots$$

and so has leading coefficient 2^n.

If we choose the nodes of the interpolation $(x_0,\ \ldots,\ x_k,\ \ldots,\ x_n)$ to be equal to the zeros of $T_{n+1}(x)$ then this is equivalent to saying $\prod_{k=0}^{n}(x-x_k) = \frac{1}{2^n}T_{n+1}$. Now,

$$|T_{n+1}(x)| \le 1 \quad \Rightarrow \quad \left|\prod_{k=0}^{n}(x-x_k)\right| \le \frac{1}{2^n} \quad \Rightarrow \quad \max_{-1\le x\le 1}\left|\prod_{k=0}^{n}(x-x_k)\right| = \frac{1}{2^n}\,.$$

Thus,

$$\left\|\prod_{k=0}^{n}(x)\right\|_\infty = \frac{1}{2^n}\,.$$

Is this the minimum value for $\left\|\prod_{k=0}^{n}(x)\right\|_\infty$ we can find?

Theorem:

$$\left\|\frac{\mathbf{1}}{\mathbf{2^n}}\mathbf{T_{n+1}}\right\|_\infty \le \|\mathbf{q(x)}\|_\infty \qquad \mathbf{x} \in [\mathbf{-1},\mathbf{1}]$$

for all $\mathbf{q(x)} \in \mathbf{P_{n+1}}$ [set of polynomials of degree $n+1$] **with leading coefficients of 1**.

Proof:

Suppose the theorem is FALSE. That is assume there exists a polynomial $r(x)$ of degree $n+1$, with leading coefficient 1, such that:

$$\|r(x)\|_\infty < \frac{1}{2^n} = \left\|\frac{1}{2^n}T_{n+1}\right\|_\infty\,.$$

Consider $[r(x) - \frac{1}{2^n}T_{n+1}]$. This is a polynomial of degree n, since the leading terms cancel (both have same coefficient of 1).

From the 2nd property of T_{n+1} we know it has $n+2$ extreme points which oscillate in sign. Also, from the definition of $r(x)$, we know these extremes are larger than the extremes of $r(x)$. Hence, at the extreme points of T_{n+1}

$$[r(x) - \frac{1}{2^n}T_{n+1}]$$

will oscillate in sign.

This means $[r(x) - \frac{1}{2^n}T_{n+1}]$ has $(n+1)$ zeros (at least).

But a polynomial of degree n has at most n zeros. Thus, there can be no such $r(x)$, which implies that

$$\min_{q(x)\in P_{n+1}} \|q(x)\|_\infty = \left\| \frac{1}{2^n}T_{n+1} \right\|_\infty ,$$

where $q(x) \in P_{n+1}$ is of the form $q(x) = x^{n+1} + \ldots$. □

Since $\prod_{k=0}^{n}(x - x_k) = (x - x_0)(x - x_1)\ldots(x - x_n)$ is a polynomial belonging to P_{n+1}, with leading coefficient 1, we have:

$$\min\left\{ \max_{-1\le x\le 1} \left| \prod_{k=0}^{n}(x - x_k) \right| \right\} = \min \left\| \prod_{k=0}^{n}(x - x_k)) \right\|_\infty = \frac{1}{2^n} ,$$

where $\prod_{k=0}^{n}(x - x_k) = \frac{1}{2^n}T_{n+1}$. In other words, $\prod_{k=0}^{n}(x - x_k)$ is minimised (with minimum value $\frac{1}{2^n}$) by choosing $x_0, x_1, \ldots, x_n$ as the zeros of the Chebyshev polynomial $T_{n+1}(x)$.

Summary: If we choose $p_n(x) \in P_n$ as the polynomial which interpolates $f(x)$ on the zeros of T_{n+1}, then

$$\|f(x) - p_n(x)\|_\infty \le \frac{1}{2^n(n+1)!} \max_{x_0\le u\le x_n} \left| f^{(n+1)}(u) \right| , \qquad x \in [-1, 1] .$$

Note, we can always transform the Chebyshev nodes $x_{ck} \in [-1, 1]$ to nodes $x_k \in [a, b]$, using

$$x_k = \frac{(a+b)}{2} + \frac{(b-a)}{2}x_{ck} .$$

So to minimise the error bound in interpolation, we choose the points

$$x_k = \frac{(a+b)}{2} + \frac{(b-a)}{2}x_{ck} ,$$

where $T_{n+1}(x_{ck}) = 0$ for some selected n.

This is the best we can do with polynomial interpolation, fitting a single polynomial through data points.

Example 3.4.1: *Let us return again to Example 3.3.1 and see how well a 3rd order polynomial does when the data points are chosen to be at the Chebyshev nodes.*

Recall, in this example $f(x) = sin(x), \quad x =\in [0, \pi]$. *Using* $n = 3$ *giving 4 equally spaced points* $x_0 = 0,\ x_1 = \pi/3,\ x_3 = 2\pi/3$ *and* $x_4 = \pi$ *we found an approximation* $p_3(x)$.

We would like to compare this with the approximation $p_3^*(x)$ *based on the "Chebyshev zeros" of* $T_{n+1}(x) = T_4(x)$, *given by*

$$T_{n+1}(x) = 0 \qquad \Rightarrow \qquad T_4(x) = 8x^4 - 8x^2 + 1 = 0\ ,$$

$$x^2 = \frac{8}{16} \pm \frac{1}{16}\sqrt{64 - 32} = \frac{1}{2} \pm \frac{\sqrt{2}}{4}$$

Hence,

$$\Rightarrow \begin{cases} x_{c0} &= \sqrt{\frac{1}{2} + \frac{\sqrt{2}}{4}} = 0.9239 \\ x_{c1} &= \sqrt{\frac{1}{2} - \frac{\sqrt{2}}{4}} = 0.3827 \\ x_{c2} &= -\sqrt{\frac{1}{2} - \frac{\sqrt{2}}{4}} = -0.3827 \\ x_{c3} &= -\sqrt{\frac{1}{2} + \frac{\sqrt{2}}{4}} = -0.9239 \end{cases}$$

or,alternatively,

$$T_{n+1}(x) = 0 \qquad \Rightarrow \qquad x_{ck} = \cos\left((2k+1)\frac{\pi}{2(n+1)}\right) \quad \text{where} \quad k = 0, 1, 2, 3$$

$$\Rightarrow \begin{cases} x_{c0} &= \cos\frac{\pi}{8} = 0.9239 \\ x_{c1} &= \cos\frac{3\pi}{8} = 0.3827 \\ x_{c2} &= \cos\frac{5\pi}{8} = -0.3827 \\ x_{c3} &= \cos\frac{7\pi}{8} = -0.9239 \end{cases}$$

Clearly, these nodes lie between -1 *and* 1. *To map them onto the range* $[a, b] = [0, \pi]$ *we use the following*

$$x_k = \frac{1}{2}(a + b) + \frac{1}{2}(b - a)x_{ck} = \frac{\pi}{2} + \frac{\pi}{2}x_{ck}\ .$$

Hence, the Chebyshev nodes for our particular problem are

$$\begin{cases} x_0 &= \frac{\pi}{2} + \frac{\pi}{2}0.9239 = 3.0220 = 0.9619\pi \\ x_1 &= \frac{\pi}{2} + \frac{\pi}{2}0.9239 = 2.1719 = 0.6913\pi \\ x_2 &= \frac{\pi}{2} + \frac{\pi}{2}0.9239 = 0.9697 = 0.3087\pi \\ x_3 &= \frac{\pi}{2} + \frac{\pi}{2}0.9239 = 0.1196 = 0.0381\pi \end{cases}$$

Using these nodes implies we should use the data points

$$(x_0, f(x_0)) = (0.961\pi, 0.1192), \quad (x_1, f(x_1)) = (0.6913\pi, 0.8247), \quad (x_2, f(x_2)) = (0.3087\pi, 0.8247)$$

$$\text{and} \quad (x_3, f(x_3)) = (0.0381\pi, 0.1192)\ ,$$

we now find the 3rd order polynomial interpolant

$$p_3(x) = -1.3663 \times 10^{-8}x^3 + -0.3948x^2 + 1.2405x - 0.000 \ .$$

We plot this interpolant in Figure 3.9(a). In Figure 3.9(b) both the error from this polynomial fit

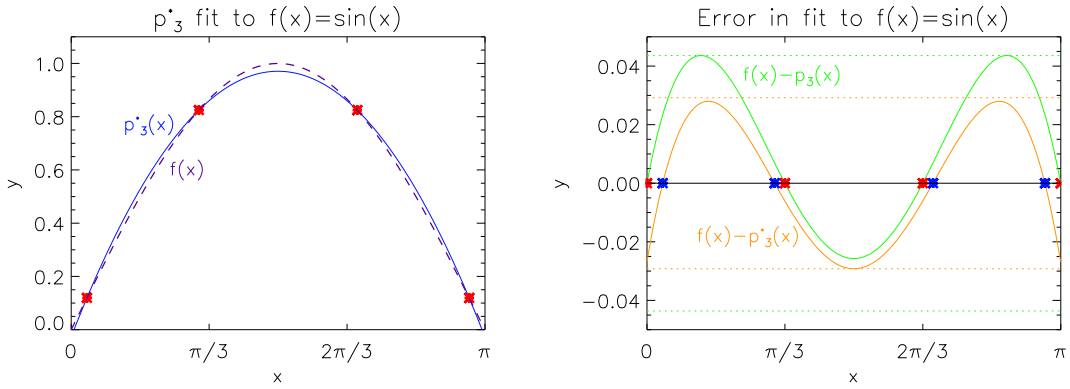


Figure 3.9: (Example 3.4.1) (a) 3rd order polynomial approximation (solid blue lines) to the 4 data points (red) found using the Chebyshev nodes. The purple dotted line is the function that created the data points.(b) The error $f(x) - p_3^*(x)$ (orange) and the error for the original polynomial found using uniformly spaced points $f(x) - p_3(x)$ (green).

$f(x) - p_3^*(x)$ and the error of the original polynomial fit $f(x) - p_3(x)$ are plotted. Clearly, overall using the Chebyshev nodes a more accurate fit is achieved, even if in places the other polynomial fit does better.

Example 3.4.2: *In this example we consider the polynomial approximations to $f(x) = (1+25x^2)^{-1}$, $x \in [-1, 1]$, the expression previously considered in Example 3.3.1. In this example, we found that the polynomial fit did not seem to improve very well as n increased and the polynomial fit seemed to oscillate about $f(x)$.*

Here, we again calculate the polynomial approximations with dimensions $n = 5,\ 9$ and 10, but this time instead of using uniformly spaced data points we use data points given by the Chebyshev nodes. The results are plotted in Figure 3.10.

The table below compares the maximum error found for the polynomial approximations found using the Chebyshev nodes $\max(|f(x) - p_n^*(x)|)$ *and those obtained using uniformly spaced data points* $\max(|f(x) - p_n(x)|)$*. The Chebyshev nodes clearly help, especially when n is even..*

Summary of polynomial interpolation: So far, we have discussed polynomial interpolation to a continuous function f, defined on the interval $[a, b]$. This was justified by Weierstrass' Theorem which assures us that by choosing a polynomial of sufficiently high degree we can approximate the given function as closely as we wish.

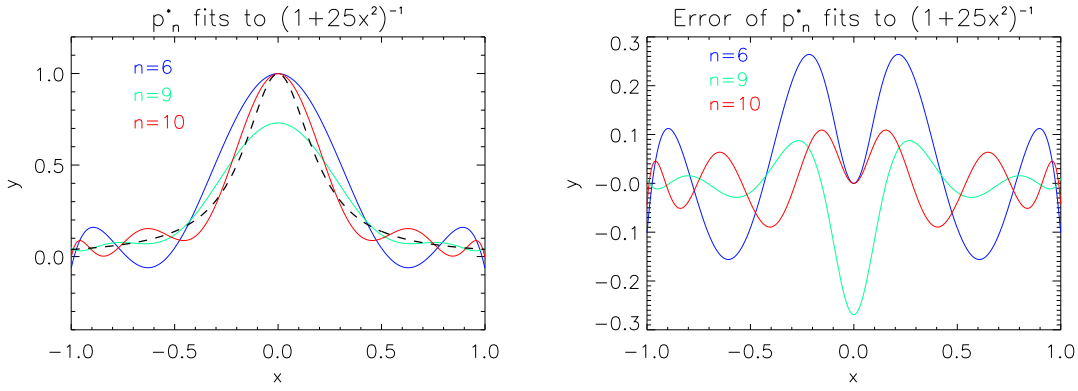


Figure 3.10: (Example 3.4.2.) (a) Polynomial approximation for $f(x) = 1/(1+25x^2)$, $x \in [-1,1]$ obtained using data points relating to the Chebyshev nodes with $n = 6$ (blue), $n = 9$ (green) and $n = 10$ (red). The black dotted line is the actual function. (b) Error $p_n^*(x) - f(x)$ of the polynomial fits.

Dimension of polynomial	$\max(\lvert f(x) - p_n^*(x)\rvert)$	$\max(\lvert f(x) - p_n(x)\rvert)$
$n = 6$	0.2642	0.6169
$n = 9$	0.2691	0.3003
$n = 10$	0.1091	1.916

However, as the degree of the interpolating polynomial increases, its oscillatory behaviour increases as well. This makes it harder to find a 'good' approximation.

Using 'modest' n-degree polynomials, the best we can do is to choose the interpolation points $x_0, x_1, \ldots, x_n$ as the zeroes of the Chebyshev polynomial $T_{n+1}(x)$.

3.5 Splines (Piecewise Approximation)

An alternative approach to minimising the error is to restrict the approximation to low degree polynomials fitted over sections of the function. i.e. to perform piecewise approximations.

We considered piecewise linear approximation in §3.2, but in order to understand how to produce piecewise approximations using other n-degree polynomials (known as splines) we revisit piecewise linear approximation.

3.5.1 Linear Splines

Recall that if the interval $[a, b]$ is partitioned into n equal sub-intervals $[x_k, x_{k+1}]$ $(0 \leq k \leq n-1)$ where $a = x_0 < x_1 < \cdots < x_n = b$ and each consecutive pair of data points is connected by a straight line equal to (3.2)

$$p_{1k}(x) = \begin{cases} \frac{(x-x_{k+1})}{(x_k-x_{k+1})} f(x_k) + \frac{(x-x_k)}{(x_{k+1}-x_k)} f(x_{k+1}) & x \in [x_k, x_{k+1}] \\ 0 & \text{elsewhere} \end{cases} . \tag{3.10}$$

The complete piecewise linear function is

$$S_1(x) = \sum_{k=0}^{n-1} p_{1k}(x) \, .$$

Alternatively, it can be rewritten as

$$S_1(x) = \sum_{k=0}^{n} f(x_k)\phi_k(x) \, .$$

What do these $\phi_k(x)$ look like?

From (3.10) it is clear it is made up of two components, one in the interval $[x_{k-1}, x_k]$ and the other from the interval $[x_k, x_{k+1}]$, i.e.

$$\phi_k(x) = \begin{cases} \frac{x-x_{k+1}}{x_k-x_{k+1}} & x_k \leq x \leq x_{k+1} \\ \frac{x-x_{k-1}}{x_k-x_{k-1}} & x_{k-1} \leq x \leq x_k \\ 0 & \text{elsewhere} \, . \end{cases}$$

$\phi_k(x)$ forms a 'hat' function. In particular,

$$\phi_k(x_{k-1}) = 0 \, , \quad \phi_k(x_k) = 1 \quad \text{and} \quad \phi_k(x_{k+1}) = 0 \, .$$

The sum of all these hat functions $\phi_k(x)$ times by the values $f(x_k)$ produces the function S_1 which is called a *first degree spline*. The 'hat' functions are known as the *basis* for this linear spline. S_1 is continuous, but may have discontinuities in its first derivatives at the knots x_k.

Example 3.5.1 *Consider $f(x) = \sin(x)$ $x \in [0, \pi]$ again, as seen in Examples 3.1.1, in which we found a piecewise linear fit with $n + 1 = 4$ data points. The piecewise linear solution $P_1(x)$ can be written as $S_1(x)$ the linear spline solution (the result is the same).*

$$S_1(x) = \sum_{k=0}^{3} f(x_k)\phi_k(x) \, .$$

where

$$\phi_k(x) = \begin{cases} \frac{x-x_{k+1}}{\pi/3} & x_k \leq x \leq x_{k+1} \\ \frac{x-x_{k-1}}{\pi/3} & x_{k-1} \leq x \leq x_k \\ 0 & \textit{elsewhere} \end{cases} \qquad k = 0, 1, 2, 3 \, ,$$

Since $n = 3$ $x_k = \pi k/3$ *with* $k = 0, 1, 2$ *and 3.*

Figure 3.11(a) shows a plot of the $\phi_k(x)$ $k = 0, 1, 2, 3$ *functions.*

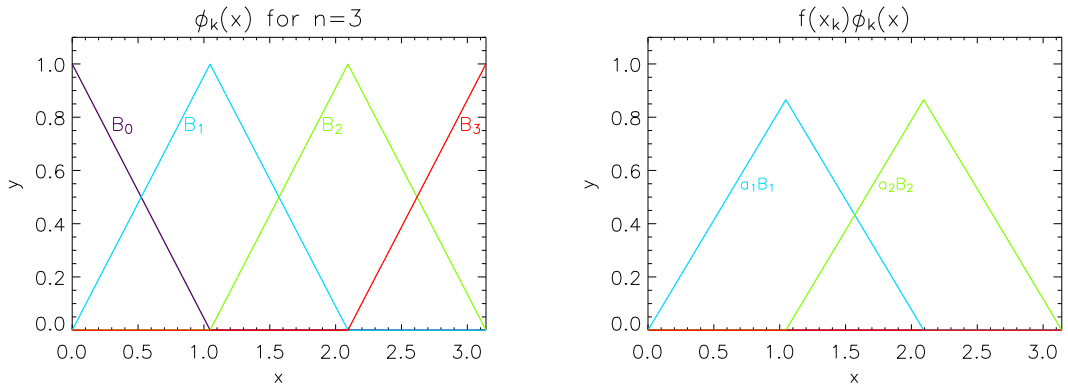


Figure 3.11: (Example 3.5.1) (a) The $n + 1$ $\phi_k(x)$ functions for the $n = 3$ linear spline. (b) The $n + 1$ $f(x_k)\phi_k(x)$ weighted spline components for the $n = 3$ linear spline fit to $f(x) = \sin(x)$.

These function are weighted according to the components $f(x_k)$*, which in this case equal*

$$f(x_0) = 0\,, \quad f(x_1) = 0.8660\,, \quad f(x_2) = 0.8660 \text{ and } f(x_3) = 0\,.$$

These leave us with just two weighted spline components ϕ_1 *and* ϕ_2 *seen in Figure 3.11(b).*

Finally, summing all these components up gives the $S1(x)$*, the linear spline fit to* $f(x) = \sin(x)$ *(Figure 3.12).*

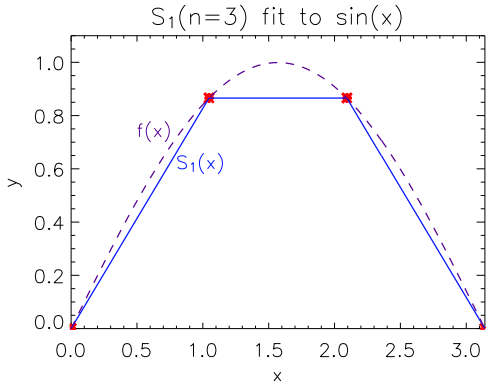


Figure 3.12: (Example 3.5.1) The $n = 3$ linear spline $S_1(x)$ fit to $f(x) = \sin(x)$.

Linear piecewise approximations are fine for graphics since the error does not have to be particularly small to create a satisfactory visual effect. However, we need to do something better than this for accurate computations. In many practical applications, we will desire a greater degree of

'smoothness', so let us use higher degree functions, say cubics, as interpolants on the sub-intervals.

3.5.2 Splines: General Definition

In general, such a piecewise function forming a spline of degree m, S_m say, is defined as follows:
Let $[a, b]$ be partitioned into sub-intervals $[x_{k-1}, x_k]$, $1 \leq k \leq n$,where $a = x_0 < x_1 < \cdots < x_n = b$.
A spline S_m of degree m in $[a, b]$ satisfies the following properties:

(i) $S_m \in P_m$: S_m restricted to $[x_{k-1}, x_k]$ is a polynomial of degree at most $m \geq 1$.

(ii) $S_m \in C^{m-1}[a, b]$: the $(m-1)$th derivatives are continuous.

Thus, for a given partition of $[a, b]$, the spline consists of n m-degree polynomial segments with the appropriate continuity condition at each of the interior knots. This will ensure that the spline has a certain degree of 'smoothness'.

3.5.3 Cubic Splines

We consider the most commonly used spline, *the cubic spline*, when $m = 3$.

Given $(n+1)$ points, $x_0, x_1, \ldots, x_n$, which are equally spaced, we wish to construct a piecewise cubic $S_3(x)$ such that

$$S_3(x_k) = f(x_k), \qquad k = 0, 1, \ldots, n\,,$$

and

$$S_3(x) \in C^2[x_0, x_n]\,,$$

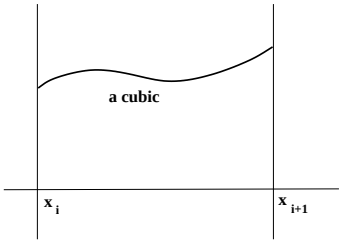
i.e. $S_3(x)$ is continuous and has continuous 1st and 2nd derivatives for $x_0 < x < x_n$ (at the interior knots).

Question: Can this be done?

A typical cubic has 4 unknowns $(ax^3 + bx^2 + cx + d)$. We have n intervals so there are n cubics to be constructed, i.e. $4n$ unknowns.

When fitting the data points, there are 2 values of $f(x)$ for each sub-interval, and n intervals, so a total of $2n$ conditions.

There are $(n-1)$ interval data points, $x_1, x_2, \ldots, x_{n-1}$, so to ensure that $S'(x)$ is continuous we have $(n-1)$ conditions.



Likewise, to ensure $S''(x)$ is continuous we have a further $(n-1)$ conditions.

Hence, a total number of $4n-2$ conditions. We are 2 short!

The two further conditions can be found in several ways:

• Case (i): If we know $f'(x_0)$ and $f'(x_n)$ we could use these to provide 2 further conditions.

• Case (ii): We could estimate $f'(x_0)$ and $f'(x_n)$ from the values of $f(x_k)$ and use these estimates.

• Case (iii): We could impose the condition that $S''(x_0) \equiv 0$ and $S''(x_n) \equiv 0$. A spline constructed in this way is called a *natural spline*

Question: How do we construct $S_3(x)$? With n cubics, $4n$ unknowns and $4n$ equations, the whole exercise looks horrendous!

One of the easiest ways to do this is to define a basis for the cubic spline similar to the basis of the linear spline.

3.5.3.1. The Basis Approach

Given a set of $(n+1)$ uniformly spaced data points (n equal sub-intervals), then a cubic spline

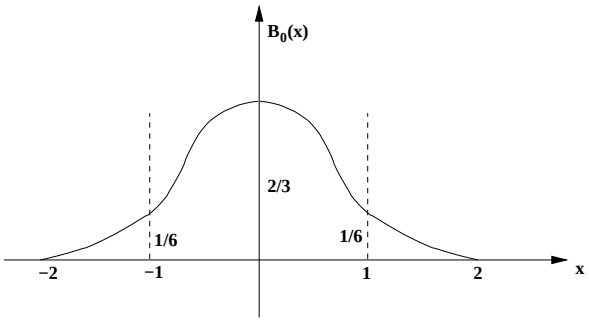
$$S_3(x) = \sum_{k=-1}^{n+1} a_k B_k(x)\,, \tag{3.11}$$

where $B_k(x)$ is the *basis* of the cubic spline.

Let the $(n+1)$ knots be $x_0, x_1, \ldots, x_n$ and assume that these knots are uniformly spaced at intervals of $h = 1$, for simplicity (this assumption will be relaxed later so we can consider more general cases). Thus, $x_k = k$.

The Cubic B-Spline $\mathbf{B_0}(\mathbf{x})$ is defined as:

$$\begin{aligned}
B_0(x) &= 0, && x \le -2\,,\\
B_0(x) &= \frac{1}{6}(2+x)^3, && -2 \le x \le -1\,,\\
B_0(x) &= \frac{2}{3} - \frac{1}{2}x^2(2+x), && -1 \le x \le 0\,,\\
B_0(x) &= \frac{2}{3} - \frac{1}{2}x^2(2-x), && 0 \le x \le 1\,,\\
B_0(x) &= \frac{1}{6}(2-x)^3, && 1 \le x \le 2\,,\\
B_0(x) &= 0, && x \ge 2\,,
\end{aligned}$$



Note, that at the knots $x_k = 0,\ \pm 1$ and ± 2

$$B_0(0) = \frac{2}{3},\ \ B_0(\pm 1) = \frac{1}{6},\ \ B_0(\pm 2) = 0\,.$$

$$B_0'(0) = 0, \;\; B_0'(\pm 1) = \pm\frac{1}{2}, \;\; B_0'(\pm 2) = 0\,.$$

$$B_0''(0) = -2, \;\; B_0''(\pm 1) = \pm 1, \;\; B_0''(\pm 2) = 0\,.$$

$B_0(x)$ is an example of a cubic spline. It is simple show that $B_0'(x)$ and $B_0''(x)$ are continuous (See tutorial sheet 4).

In general, $B_k(x)$ is defined as $B_0(x-k)$.

A linear combination of such functions will be continuous on $C^2[0,1]$, so they will automatically fulfill $(n-1)\times 3 = 3n-3$ of the $4n$ conditions. Hence, another $4n-(3n-3) = n+3$ conditions are required.

Now consider the function $S_3(x)$ (3.11) on the points, $0, 1, \ldots, n$,

$$S_3(x) = \sum_{k=-1}^{n+1} a_k B_k(x)\,,$$

which represents a linear combination of $(n+3)$ functions with the prescribed continuities.

• Consider an interval, $[0,1]$ say, depicted in Figure 3.13

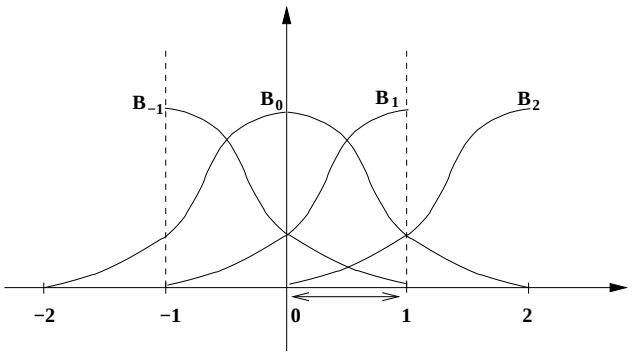


Figure 3.13: The B-splines lying in the interval $[0,1]$.

In this interval, $[0,1]$, there are non-zero contributions from 4 B-splines:

$$B_{-1}(x), \quad B_0(x), \quad B_1(x) \quad \text{and} \quad B_2(x).$$

Similarly, the general interval $[x_k, x_{k+1}]$ $(= [k, k+1])$ has non-zero contributions from B_{k-1}, B_k, B_{k+1} and B_{k+2} (Figure 3.14).

Thus, in the interval $[k, k+1]$,

$$\begin{aligned} S_3(x) &= a_{k-1}B_{k-1}(x) + a_k B_k(x) + a_{k+1}B_{k+1}(x) + a_{k+2}B_{k+2}(x) \\ &= a_{k-1}B_0(x-k+1) + a_k B_0(x-k) + a_{k+1}B_0(x-k-1) + a_{k+2}B_0(x-k-2)\,. \end{aligned}$$

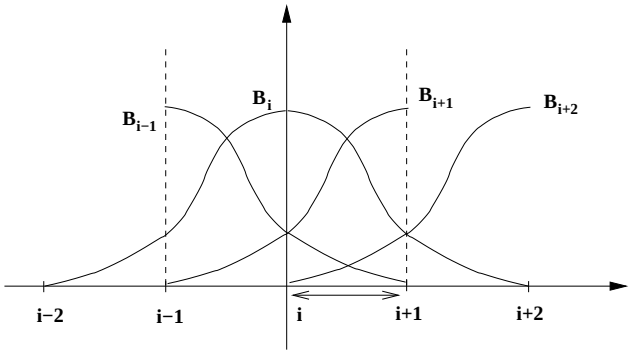


Figure 3.14: The B-splines lying in the interval $[k, k+1]$.

At the knots, $x_k = k$, our cubic spline must equal the values $f(x_k)$. Hence, we require

$$S_3(x_k) = f(x_k) = f(k)\,.$$

So,

$$S_3(k) = a_{k-1}B_0(1) + a_k B_0(0) + a_{k+1}B_0(-1) + a_{k+2}B_0(-2) = f(k) = f(x_k)\,,$$

which gives

$$f(x_k) = f(k) = \frac{1}{6}a_{k-1} + \frac{2}{3}a_k + \frac{1}{6}a_{k+1} + 0\,,$$

and thus,

$$a_{k+1} + 4a_k + a_{k-1} = 6f(x_k), \qquad x_k = k\,. \tag{3.12}$$

We have $(n+3)$ unknowns, $a_{-1}, a_0, \ldots, a_{n+1}$ and $(n+1)$ data points $(x_k, f(x_k))$, $k = 0, 1, \ldots, n$.

Thus, we require <u>2 extra conditions</u> to find all a_k's.

Suppose we insist that $S_3''(0) = 0$ as one extra condition:

$$(k=0) \qquad a_{-1}B_{-1}''(0) + a_0 B_0''(0) + a_1 B_1''(0) + a_2 B_2''(0) = 0\,,$$

and hence,

$$a_{-1}B_0''(1) + a_0 B_0''(0) + a_1 B_0''(-1) + a_2 B_0''(-2) = 0\,,$$

or,

$$a_{-1} \times (1) + a_0 \times (-2) + a_1 \times (1) + 0 = 0\,,$$

so (3.12) becomes

$$a_{-1} = 2a_0 - a_1\,, \tag{3.13}$$

and,

$$a_{-1} + 4a_0 + a_1 = 6f(x_0) = 6f(0)\,,$$

so

$$6a_0 = 6f(0)\,.$$

We can produce a similar equation by requiring that

$$S_3''(n) = 0\,,$$

which gives us

$$a_{n-1} - 2a_n + a_{n+1} = 0 \quad \Rightarrow \quad a_{n+1} = 2a_n - a_{n-1}\,, \tag{3.14}$$

and hence,

$$6a_n = 6f(n)\,.$$

Thus, for a natural spline we have the following system of equations

$$\begin{aligned}
&a_0 = f(0)\\
&a_0 + 4a_1 + a_2 = 6f(1)\\
&\cdots\\
&a_{n-2} + 4a_{n-1} + a_n = 6f(n-1)\\
&a_n = f(n)\,.
\end{aligned}$$

Or, in matrix form:

$$\begin{bmatrix}
1 & 0 & 0 & \dots & & \dots & 0\\
1 & 4 & 1 & 0 & & \dots & \vdots\\
0 & 1 & 4 & 1 & 0 & \dots & 0\\
\vdots & & & & & & \vdots\\
\vdots & & \dots & 0 & 1 & 4 & 1\\
0 & & \dots & & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix} a_0\\ a_1\\ \vdots\\ \vdots\\ a_{n-1}\\ a_n \end{bmatrix}
=
\begin{bmatrix} f(0)\\ 6f(1)\\ \vdots\\ \vdots\\ 6f(n-1)\\ f(n) \end{bmatrix} \tag{3.15}$$

This is a linear system of $(n+1)$ equations which we can solve to find the $(n+1)$ unknowns a_k, $k = 0, \dots, n$. Then a_{-1} and a_{n+1} can be found using equations (3.13) and (3.14).

Finally, once we know all the a_k, the full spline $S_3(x)$ can be obtained from equation (3.11).

Note, that the system given above in (3.15) is 'well-posed' - the condition number is modest (≤ 3).

Given the basic idea, it is easy to generalise to higher-order piecewise polynomials, but the most commonly used are cubic splines.

3.5.3.2. General Cubic B-Spline

In the previous section, we considered the cubic spline fitted to a set of $n+1$ data points uniformly distributed on $[0, n]$. i.e. the data points had an interval of 1 such that $x_k = k$, $k = 0, \ldots, n$. However, in general, data points are unlikely to be spaced in such a manner.

Cubic splines are relatively simple on uniformly spaced data points and so we consider the case of $n+1$ data points uniformally spaced with an interval of h over the range [a,b]. Hence $x_k = a + kh$ and $h = (b-a)/n$

The basis for the cubic B-Spline then becomes

$$
\begin{aligned}
B_0(x) &= 0, \qquad x - a \le -2h\,, \\
B_0(x) &= \frac{1}{6}(2h + (x-a))^3, \qquad -2h \le x - a \le -h\,, \\
B_0(x) &= \frac{2h^3}{3} - \frac{1}{2}(x-a)^2(2h + (x-a)), \qquad -h \le x - a \le 0\,, \\
B_0(x) &= \frac{2h^3}{3} - \frac{1}{2}(x-a)^2(2h - (x-a)), \qquad 0 \le x - a \le h\,, \\
B_0(x) &= \frac{1}{6}(2h - (x-a))^3, \qquad h \le x - a \le 2h\,, \\
B_0(x) &= 0, \qquad x - a \ge 2h\,,
\end{aligned}
$$

The overall cubic spline is still equal to

$$S_3(x) = \sum_{k=-1}^{n+1} a_k B_k(x)\,,$$

with $B_k(x) = B_0(x - kh)$ and the a_k , $k = -1, \ldots, n+1$ given by

$$
\begin{aligned}
a_0 &= \frac{1}{h^3} f(x_0) \\
a_0 + 4a_1 + a_2 &= \frac{6}{h^3} f(x_1) \\
&\ldots \\
a_{n-2} + 4a_{n-1} + a_n &= \frac{6}{h^3} f(x_{n-1}) \\
a_n &= \frac{1}{h^3} f(x_n)\,.
\end{aligned}
$$

with, in the natural spline case,

$$a_{-1} = 2a_0 - a_1 \quad \text{and} \quad a_{n+1} = 2a_n - a_{n-1}\,.$$

Example 3.5.2: *In this example, we consider the cubic spline approximations to* $f(x) = \sin(x)$, $x \in [0, \pi]$, *the function previously considered in Examples 3.1.1, 3.3.1, 3.4.1. and 3.5.1. We will compare our cubic spline approximations found with the polynomial fits obtained for uniformly spaced*

data points (Example 3.3.1) and data points obtained from the Chebyshev nodes (Example 3.4.1) and the linear spline fit (Example 3.5.1).

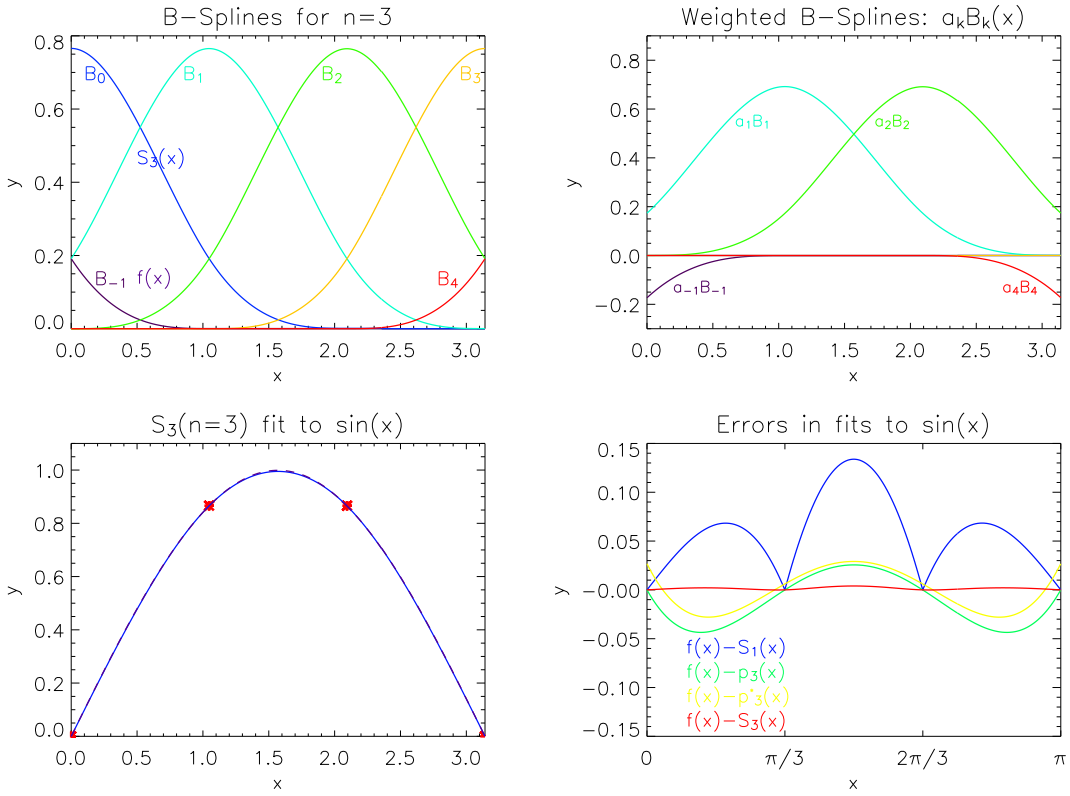


Figure 3.15: (Example 3.5.1) (a) Cubic Bsplines for $n = 3$ uniformly spaced data points. (b) Individual weighted splines ($a_k B_k(x)$). (c) Cubic spline approximation to $f(x) = sin(x), \quad x \in [0, \pi]$ obtained using $n = 3$ uniformly spaced data points (blue). The black dotted line is the actual function. (b) Error $S_3(x) - f(x)$ of the cubic spline.

Here, we calculate the cubic spline approximations using $(n+1)$ data points with $n = 3$, as in all the previous cases. The data points are uniformly spaced with $h = \pi/3$ and are given in the table below

k	$(x_k, f(x_k))$
0	$(0.0, 0.0)$
1	$(\pi/3, 0.8660)$
2	$(2\pi/3, 0.8660)$
3	$(\pi, 0.0)$

Thus, the linear system we must solve is

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 4 & 1 & 0 \\ 0 & 1 & 4 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} \left(\frac{3}{\pi}\right)^3 f(x_0) \\ 6\left(\frac{3}{\pi}\right)^3 f(x_1) \\ 6\left(\frac{3}{\pi}\right)^3 f(x_2) \\ \left(\frac{3}{\pi}\right)^3 f(x_3) \end{bmatrix} = \begin{bmatrix} 0.0 \\ 4.525 \\ 4.525 \\ 0.0 \end{bmatrix}$$

Hence,

$$a_0 = a_3 = 0 \quad \text{and} \quad 4a_1 + a_2 = a_1 + 4a_2 = 4.525 \,,$$

$$\Rightarrow \quad a_1 = a_2 = \frac{4.525}{5} = 0.905 \,.$$

In addition,

$$a_{-1} = 2a_0 - a_1 = -0.905 \quad \text{and} \quad a_4 = 2a_3 - a_2 = -0.905 \,.$$

The individual Bsplines, $B_k(x)$, and weighted Bsplines, $a_k B_k(x)$, are plotted in Figure 3.15a & 3.15b, whilst the sum of these terms, $S_3(x)$ is plotted in Figure 3.15c.

Clearly, this cubic spline is the best fit to the $f(x) = \sin(x)$. We compare it with the simple piecewise linear fit (linear spline), single 3rd degree polynomial derived from uniformly spaced data, and a 3rd degree polynomial found using the Chebyshev nodes in Figure 3.15d. The maximum error for each of these fits is given in the table below.

Fit $Z(x)$	$\max(\lvert f(x) - Z(x) \rvert)$
$S_1(x)$	0.134
p_3	0.026
p_3^*	0.029
$S_3(x)$	0.004

Example 3.5.3: *In this example, we consider the cubic spline approximations to $f(x) = (1 + 25x^2)^{-1}$, $x \in [-1, 1]$, the function previously considered in Examples 3.3.2 and 3.4.2. We will compare our spline approximations found with the polynomial fits obtained for uniformly spaced data points (Example 3.3.2) and data points obtained from the Chebyshev nodes (Example 3.4.2)*

Here, we calculate the cubic spline approximations using $(n+1)$ data points where, as before, $n = 6,\ 9$ and 10. The data points are uniformly spaced. The results are plotted in Figure 3.11.

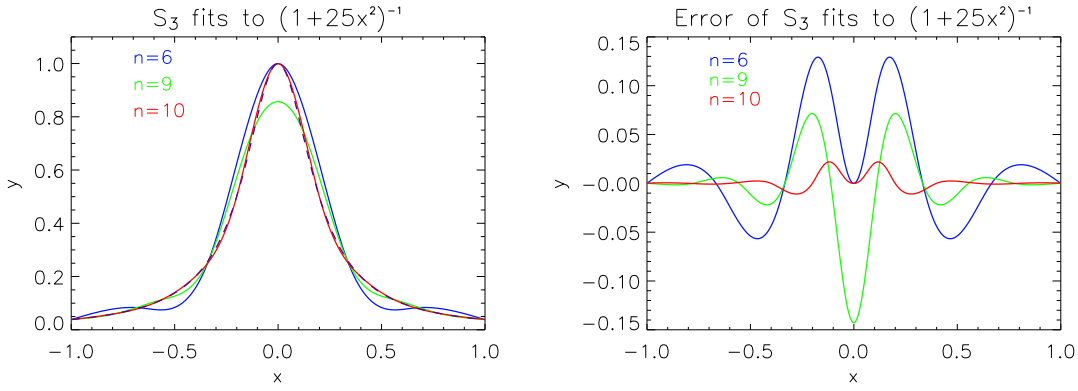


Figure 3.16: (Example 3.5.1) (a) Cubic spline approximations for $f(x) = 1/(1+25x^2), \quad x \in [-1,1]$ obtained using n+1 uniformly spaced data points with $n = 6$ (blue), $n = 9$ (green) and $n = 10$ (red). The black dotted line is the actual function. (b) Error $S_3(x) - f(x)$ of the cubic Splines.

The table below compares the maximum error found for the polynomial approximations found using the cubic splines $\max(|f(x)-S_3(x)|)$, *polynomials determined using the Chebyshev nodes* $\max(|f(x)-p_n^*(x)|)$ *and polynomials obtained using uniformly spaced data points* $\max(|f(x)-p_n(x)|)$. *The cubic splines seem to give the best fits.*

Dimension of polynomial	$\max(\|f(x) - S_3(x)\|)$	$\max(\|f(x) - p_n^*(x)\|)$	$\max(\|f(x) - p_n(x)\|)$
$n = 6$	0.1293	0.2642	0.6169
$n = 9$	0.1429	0.2691	0.3003
$n = 10$	0.0220	0.1091	1.916