9.1 Polynomial Interpolation

We have already seen how to use nonlinear functions to data. Next we consider a special fitting problem called interpolation. Given (x_i, y_i), where $x_0 < x_1 < x_2 < \ldots < x_n$, we want to find a function $f : \mathbb{R} \to \mathbb{R}$ such that

$$f(x_i) = y_i, \quad i = 0, 1, \ldots, n \tag{9.1}$$

It is too ambitious to explore the whole space of all the functions. Rather than that, we assume that the functions are from the class of polynomials. A polynomial of degree n has the form

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots a_1 x + a_0.$$

The set of all polynomials of degree no more than n is denoted by $\mathcal{P}_n = \{p : \deg p \leq n\}$.

Proposition 1. *Let $(x_i)_{0 \leq i \leq n}$ be distinct points. There exists a unique polynomial p of degree at most n that satisfies $p(x_i) = y_i$, $i = 0, 1, \ldots, n$.*

The existence of such polynomial will be given by an explicit construction. Let us prove the uniqueness first. If there are $q \neq p$ such that $p(x_i) = y_i$. Then $p(x_i) - q(x_i) = 0$, $0 \leq i \leq n$. Since k-th polynomial has at most k roots, and $p - q$ has degree $n$and $n + 1$ roots, we must have $p - q \equiv 0$.

For simplicity, let us consider $\{x_i\}$ fixed, and $\{y_i\}$ unspecified. Our strategy is to express the polynomials using some basic polynomial functions, namely the cardinal functions:

$$l_k(x_j) = \begin{cases} 1 & j = k \\ 0 & \text{o.w.} \end{cases}.$$

Therefore, given any values of $\{y_j\}$, we can denote the interpolated polynomial by

$$p(x) = \sum_{i=0}^{n} y_i l_i(x) \tag{9.2}$$

Immediately, we see that it satisfies the relation (9.1).

Lagrange interpolation There is a lot of variability in the form of $l_k(x_j)$. It is natural to consider Lagrange polynomial:

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

The corresponding expression (9.2) is called Lagrange formula or Lagrange interpolation. Depsite its simplicity, the Lagrange interpolation has several drawbacks. Specifically, it is not efficient because each evaluation requires $\mathcal{O}(n^2)$ flops. Moreover, when adding new data point (x_{n+1}, y_{n+1}), it requires new computation from scratch. Another issue is the instability arising from the fractionals. Consider when $x_k \approx x_{k-1}$, then there is potential overflow or cancelation error.

Barycentric interpolation To alleviate those issues, we consider an equivalent but more convenient form called Barycentric interpolation. Denote the nodal polynomial:

$$\psi_{n+1}(x) = \prod_{i=0}^{n}(x - x_i), \tag{9.3}$$

and the barycentric weight

$$w_k = \left(\prod_{j=0, j\neq k}^{n} (x_k - x_j) \right)^{-1}.$$

Therefore, we can rewrite ℓ_k as

$$\ell_k(x) = \psi_{n+1}(x) \frac{w_k}{x - x_k}, \quad k = 0, 1, \ldots, n.$$

Then we have the equivalent interpolating polynomial of (9.2) as follows:

$$p(x) = \psi_{n+1}(x) \sum_{j=0}^{n} \frac{w_j}{x - x_j} y_j. \tag{9.4}$$

This is called the first form of barycentric interpolation formula. We can further simplify $p(x)$ as follows. Setting $y_j \equiv 1$, $0 \le j \le n$, we should have $p(x) \equiv 1$ (degree n with $n+1$ roots). Therefore, we can derive $\psi_n(x)$ from $1 = \psi_n(x) \sum_{j=0}^{n} \frac{w_j}{x - x_j}$. Going back to $p(x)$, we have

$$p(x) = \frac{\sum_{j=0}^{n} \frac{w_j}{x - x_j} y_j}{\sum_{j=0}^{n} \frac{w_j}{x - x_j}}. \tag{9.5}$$

This is called the second form of barycentric interpolation formula, which has a few computational advantages. First, (9.5) can be viewed as a weighted sum of $\{y_k\}$. And Evaluation at new point $x$or updating the formula to incorporate new data x_{n+1}, y_{n+1} can be done in $\mathcal{O}(n)$, much faster than that of Lagrange formula. Moreover, Barycentric form is numerically stable, you might worry about dividing by zero when $x \approx x_j$, since the same fractional appears in numerator and denominator, it can be safely handled.

Interpolation Error

We estimate the error of using $p(x)$ for interpolation.

Theorem 2. *Let $\{x_i\}_{0 \le i \le n}$ be distinct points in $[a, b]$. Suppose that $f(x)$ has at least $n + 1$ continuous derivatives. Then for any $x \in (a, b)$, the interpolation error is*

$$\varepsilon_n = f(x) - p(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \psi_{n+1}(x), \tag{9.6}$$

for some $\xi(x) \in [a, b]$ and ψ_{n+1} is the nodal polynomial of degree $n + 1$.

We immediately get more concrete bound for equally spaced nodes.

Corollary 3. *Let $x_i = x_0 + ih$. Suppose that $f(x)$ has at least $n+1$ continuous derivatives and the all the high order derivatives are bounded from above. Then there exists $C > 0$, and for any $x \in (x_0, x_n)$, we have*

$$|f(x) - p(x)| \leq C f^{(n+1)}(\xi) h^{n+1}.$$

Brief. In Theorem (9.6) note that $|x - x_i| \leq nh$ and $|\psi_{n+1}(x)| \leq \prod_{i=0}^{n} |x - ih| \leq \mathcal{O}\left((n+1)! h^{n+1}\right)$. □

Note that our goal is to find a polynomial that guarantee

$$p \approx \operatorname*{argmin}_p \|f - p\|_\infty$$

where the function norm $\|f\|_\infty = \max_{x\in[a,b]} |f(x)|$. From Corollary 3, as $h \to 0$, the approximation error decreases at the rate $\mathcal{O}(h^{n+1})$. Nevertheless, the above result doesn't guarantee universal stability across the whole domain $[a, b]$. (As $h \to 0$, we also have $x_n \to x_0$.). We illustrate this phenomenon in the following example.

Example 4 (Runge phenomenon)**.** The function $f(x) = \frac{1}{12x^2+1}$, $x \in [-1, 1]$. Figure 9.1 shows interpolation with polynomial of degreee 10. $x_0, x_1, x_2, \ldots, x_{10}$ are evenly distributed in $[-1, 1]$. One can observe that the approximation error is relatively large at the end point.

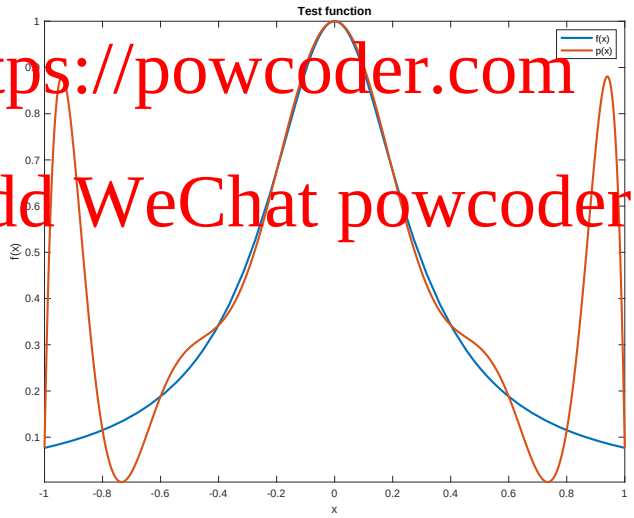

Figure 9.1: We use polynomial interpolation with equal spacing

This phenomenon can be explained by looking into the nodal polynomials $\psi_{n+1}(x)$. We plot $\psi_{n+1}(x)$ for various n in Figure 9.2. It can be immediately checked that $\psi_{n+1}(x)$ is not uniformly convergent when $\{x_k\}$ are equally spaced.

Chebyshev noted that is possible to improve the accuracy (minimizing $\psi_{n+1}(x)$) by non-uniformly placing $\{x_k\}$. Consider, for example we have $a = -1$, $b = 1$, then The Chebyshev points (second kind) are defined by:

$$x_i = -\cos\left(\frac{i\pi}{n}\right), \quad 0 \leq i \leq n.$$

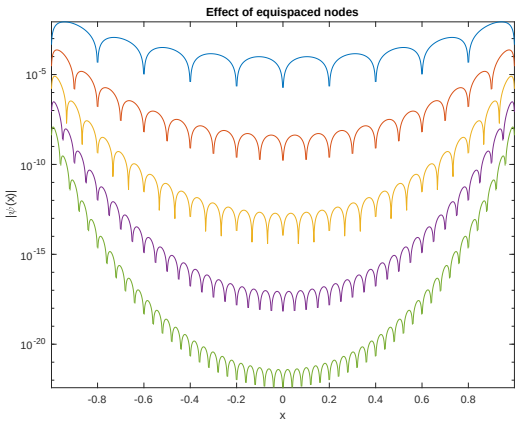

Figure 9.2: Plot the $\psi_{n+1}(x)$ function for $n = 10, 20, 30, 40, 50$.

These are the projections onto the x-axis of n equally spaced points on a unit circle. In Figure 9.3 we plot the fitted polynomial and the corresponding nodal polynomial.

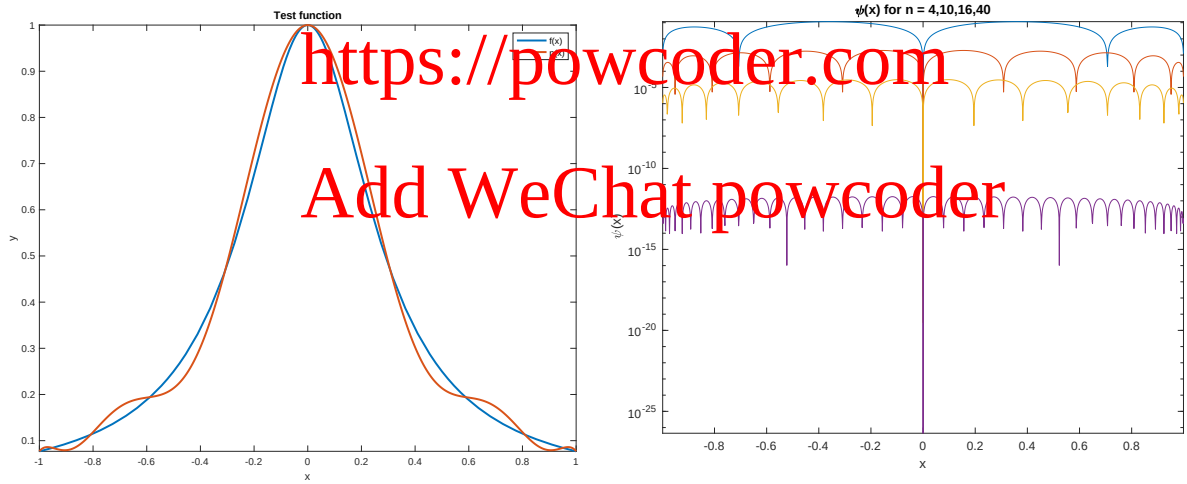

Figure 9.3: Left: Interpolation by Chebyshev points. Right: $\psi_{n+1}(x)$.

Chebyshev interpolation pays more attention to the endpoints near which large errors may occur. It can be checked that the interpolant uniformly converges to f.

Theorem 5. *Suppose $f : [-1, 1] \to \mathbb{R}$. Then there exists a constant $K > 1$ such that*

$$\|f - p\|_\infty \lesssim K^{-n}$$

where p is the unique polynomial of degree n or less defined by interpolation on $n + 1$ Chebyshev points.

9.2 Piecewise Polynomial Interpolation

9.2.1 Piecewise Linear Interpolation

In this section, we present alternative ways to interpolate nonlinear functions using piecewise polynomials. As a starting point, we consider piecewise linear (PL) interpolation, for which we define

$$p(x) = y_k + \frac{y_{k+1} - y_k}{x_{k+1} - x_k}(x - x_k), \quad \text{for } x \in [x_k, x_{k+1}], \quad k = 0, 1, ..., n-1.$$

Rather than expressing this function in conditional statement, we are interested in expressing $p(x)$ by some other cardinal functions. Similar to the Lagrange form, we can define the hat functions

$$h_k(x) = \begin{cases} \frac{x - x_{k-1}}{x_k - x_{k-1}} & \text{if } x \in [x_{k-1}, x_k] \\ \frac{x_{k+1} - x}{x_{k+1} - x_k} & \text{if } x \in [x_k, x_{k+1}] \\ 0 & \text{o.w.} \end{cases}.$$

Immediately, we observe that $h_k(x)$ is continuous and is linear inside the interval $[x_k, x_{k+1}]$. Since the linear combination of PL function is still PL, we can write $p(x) = \sum_{i=0}^{n} y_i h_i(x)$.

Since PL function fits different intervals based on local information, it is somewhat less sensitive to the global pattern of nonlinearity. The following theorem presents the approximation error of PL interpolation.

Theorem 6. *Suppose $f(x)$ has a continuous second order derivative in $[a, b]$. Let $p(x)$ be a PL interpolant with equal spacing $x_i = a + ih$ and $h = (b - a)/n$. Then*

$$\|f - p\|_\infty = \max_{x \in [a,b]} |f(x) - p(x)| \lesssim \sup_x |f''(x)| h^2.$$

This implies that if $f(x)$ grows at most quadratically ($\sup_x |f''(x)| < \infty$), then error converges to zero at the rate $\mathcal{O}(h^2)$ as $h \to \infty$.

9.2.2 Cubic Spline

PL interpolant is continuous but the derivative is not continuous. To guarantee smooth interpolation, we consider high order polynomials which have continuous derivatives. Particularly, we use the cubic spline—a piecewise cubic function which has continuous first and second derivatives. For convention, we use $S(x)$ to denote the spline interpolant, which satisfies $S(x) = S_k(x)$ when $x \in [x_{k-1}, x_k]$. The cubic polynomial $S_k(x)$ is given by

$$S_k(x) \triangleq a_k + b_k(x - x_{k-1}) + c_k(x - x_{k-1})^2 + d_k(x - x_{k-1})^3, \quad k = 1, 2, ..., n$$

Note that cubic pieces has two more degrees of freedom than PL. Therefore, we need to impose more constraints on function and high order derivatives.

1. Continuity of $S(x)$. We have

$$S_k(x_{k-1}) = y_{k-1}, \; S_k(x_k) = y_k, \quad k = 1, 2, ..., n. \tag{9.7}$$

2. Continuity of $S'(x)$. We have

$$S'_k(x_k) = S'_{k+1}(x_k), \quad k = 1, 2, ..., n-1. \tag{9.8}$$

3. Continuity of $S''(x)$. We have

$$S_k''(x_k) = S_{k+1}''(x_k), \quad k = 1, 2, ..., n-1. \tag{9.9}$$

Let us express the constraint formally. Denoting, $h_k = x_k - x_{k-1}$, then Equation (9.7)-(9.9) reads

$$\begin{aligned} a_k &= y_{k-1} \\ a_k + b_k h_k + c_k h_k^2 + d_k h_k^3 &= y_k \end{aligned} \quad k = 1, 2, ..., n. \tag{9.10}$$

$$b_k + 2c_k h_k + 3d_k h_k^2 = b_{k+1} \quad k = 1, 2, ..., n-1. \tag{9.11}$$

$$c_k + 3d_k h_k - c_{k+1} = 0 \quad k = 1, ..., n-1. \tag{9.12}$$

Since equations (9.10)-(9.12) have $4n$ variables with $-2 \cdot n + 2 \cdot (n-1) = 4n - 2$ constraints, the fitted polynomial is not unique yet. Nevertheless, that the above equations can be futher simplified. From (9.10) and 9.12) we have

$$d_k = \frac{1}{3h_k}(c_{k+1} - c_k), \quad k = 1, 2, ..., n-1 \tag{9.13}$$

$$b_k = \begin{cases} u_k - \frac{1}{3}h_k[c_{k+1} + 2c_k], & 1 \le k \le n-1. \\ u_n - c_n h_n - d_n h_n^2 & k = n \end{cases} \tag{9.14}$$

where $u_k = \frac{y_k - y_{k-1}}{h_k}$. Using (9.11) and (9.12) we have

$$(c_k + c_{k+1})h_k = b_{k+1} - b_k, \quad k = 1, 2, ..., n-1 \tag{9.15}$$

Putting (9.14) and (9.15) together we have

$$c_k h_k + 2(h_k + h_{k+1})c_{k+1} + h_{k+1}c_{k+2} = 3(u_{k+1} - u_k), \quad k = 1, 2, ..., n-2. \tag{9.16}$$

$$h_{n-1}c_{n-1} + 2(2h_{n-1} + 3h_n)c_n + 3h_n^2 d_n = 3(u_n - u_{n-1}). \tag{9.17}$$

This computational form is precisely a tridiagonal system of $n+1$ variables and $n-1$ constraints. We need two more constraint to form a square system.

Endpoint constraints The last ingredient of our linear system is two more linear constraints on the derivatives of the endpoints $S'(x_i), S''(x_i)$, $i = \{0, n\}$. The endpoint constraints are quite flexible, their choice reflecting certain assumptions of the smoothness. To this end, we give a few important cases.

1. In the *natural spline*, we enforce that the endpoints are smooth:

$$S_1''(x_0) = S_n''(x_n) = 0. \tag{9.18}$$

   That is $c_1 = c_n = 0$.

2. In the *clamped cubic spline*, we know the derivative $S'(x_0) = f'(x_0)$ and $S'(x_n) = f'(x_n)$. That is $b_1 = f'(x_0)$ and $b_n + 2c_n h_n + 3d_n h_n^2 = f'(x_n)$.

3. In the not-a-knot spline, we assume $S_1'''(x_1) = S_2'''(x_1)$ and $S_{n-1}'''(x_{n-1}) = S_n'''(x_{n-1})$. Since each S_k is cubic, it means that S_1 and S_2 are identical. We can add $d_1 = d_2$ and $d_{n-1} = d_n$ to the system.

So what is good about cubic spline? The following theorems (cf. Property 8.2, 8.3 [1]) list a few properties of cublic spline.

Theorem 7. *Let $a = x_0 < x_1 < x_2 \ldots < x_n = b$. Suppose $f(x) \in C^2[a,b]$ (twice continuously differentiable on $[a,b]$). Let $S(x)$ be the natural splinef. Then $S(x)$ has the minimum norm property:*

$$\int_a^b (S''(x))^2 dx \leq \int_a^b (f''(x))^2 dx.$$

Theorem 8. *Suppose that $f(x) \in C^4[a,b]$, then $h_i = x_i - x_{i-1}$. Let $h = \max_i h_i$ and $\beta = h/\min_i h_i$. Let $S(x)$ be a cubic spline interpolating f. Then*

$$\|f^{(r)} - S^{(r)}\|_\infty \leq C_r h^{4-r} \|f\|_\infty, \quad r = 0, 1, 2, 3.$$

In above, $f^{(r)}$ is the r-th derivative. Compared with PL interpolation, cubic spline improves the order of accuracy to four.

Example 9. We use PL functions and cubic spline to interpolate $f(x) = \exp(\sin(7x))$. In Figure 9.4 we plot the interpolation result of PL functions ($n = 5$) and plot the convergence of error as n grows in log scale. Figure 9.5 visualize the interpolant of cubic spline and its error convergence. It can be readily observed that cubic spline is more smoothly changing away from the interpolating points x_i and achieving a faster rate of convergence.

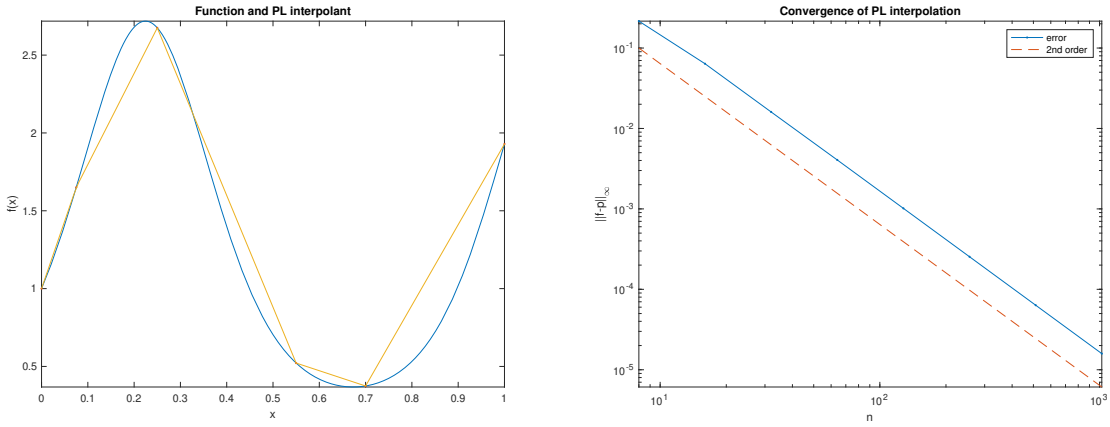

Figure 9.4: Left: interpolation by piecewise linear function. Right: Convergence of error w.r.t. n.

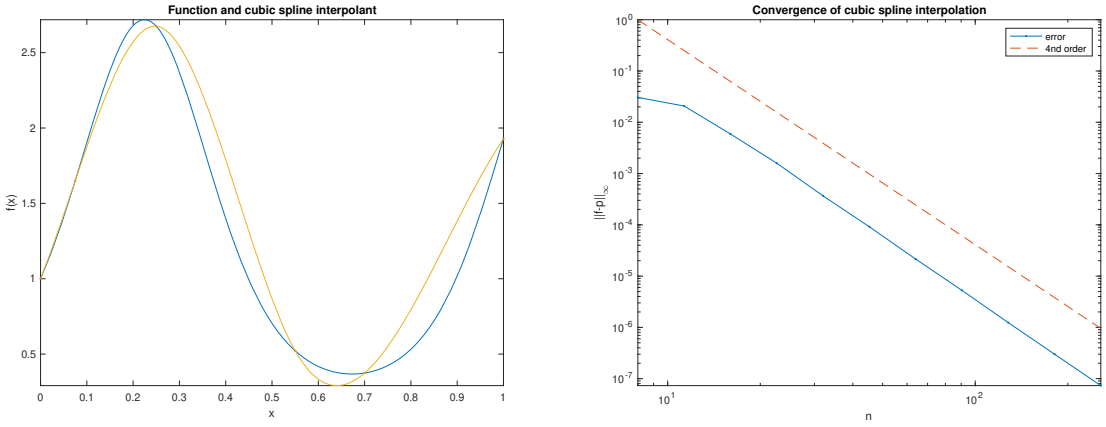

Figure 9.5: Left: interpolation by piecewise linear function. Right: Convergence of error w.r.t. n.

9.3 Conditioning

Our nest goal is to give a unified view of the studied interpolation techniques. Fixing the interpolation points $\{x_i\}$, let $\mathcal{L}(x) : \mathbb{R}^{n+1} \to \mathcal{F}$ be an abstract map of values $\{y_i\}$ to interpolation functions. Here the function space $\mathcal{F}$, depending on the specific interpolation techniques, consists of all the candidate interpolants. Recall that $\ell_i(x)$ is a cardinal function satisfying

$$l_k(x_j) = \begin{cases} 1 & j = k \\ 0 & \text{o.w.} \end{cases}. \tag{9.19}$$

Furthermore, we assume that $\mathcal{L}(y)$ is the linear combination of basic interpolants, namely, the cardinal functions:

$$\mathcal{L}(y) = \sum_{i=0}^{n} y_i \ell_i(x).$$

It follows that $\mathcal{L}(\cdot)$ has linearity, that is,

$$\mathcal{L}(\alpha y_1 + \beta y_2) = \alpha \mathcal{L}(y_1) + \beta \mathcal{L}(y_2).$$

In Polynomial interpolation, ℓ_k are the Lagrange polynomials. In PL interpolation, ℓ_k are the hat functions. In cubic spline, we can similarly define ℓ_k to be the piecewise cubic functions. We want to know how the change in function value $\{y_i\}$ affects the resulting interpolant. Let us define the conditioning of the interpolation:

$$\kappa = \lim_{\delta y \to 0} \frac{\|\mathcal{L}(y + \delta y) - \mathcal{L}(y)\|_\infty}{\|\delta y\|_\infty}.$$

Above, the denominator is the infinity norm of vector while the numerator is the norm of a function. Note that $\mathcal{L}(y + \delta y) - \mathcal{L}(y) = \mathcal{L}(\delta y) = \sum_{i=0}^{n} (\delta y)_i \ell_i(x)$. Based on triangular inequality, we thus have

$$\|\mathcal{L}(y + \delta y) - \mathcal{L}(y)\|_\infty \le \sum_{i=0}^{n} |(\delta y)_i| \cdot \|\ell_i(x)\|_\infty \le \|\delta y\|_\infty \sum_{i=0}^{n} \|\ell_i(x)\|_\infty.$$

We immediately have $\kappa \leq \sum_{i=0}^{n} \|\ell_i(x)\|_\infty$. Let $j = \text{argmax}_j \|\ell_j(x)\|_\infty$, then taking $\delta y = e_j$, we have $\kappa \geq \max_j \|\ell_j(x)\|_\infty$. Putting these two pieces together, we have

$$\kappa \in \left[\max_j \|\ell_j(x)\|_\infty, \sum_{i=0}^{n} \|\ell_i(x)\|_\infty\right].$$

Therefore, we could examine the conditioning of interplolation by looking into those cardinal functions.

Example 10. We compare the interpolation results of using piecewise cubic cardinal functions and polynomial cardinal functions. Let $-1 = x_0 < x_1 < ... < x_{18} = 1$ be evenly distributed in $[-1, 1]$. Let $y_9 = 1$ and $y_j = 0$ for $j \neq 9$. We can see that piecewise cubic interpolant nevers jumps above 1, but polynomial interpolant has significantly large values off the interpolating points.

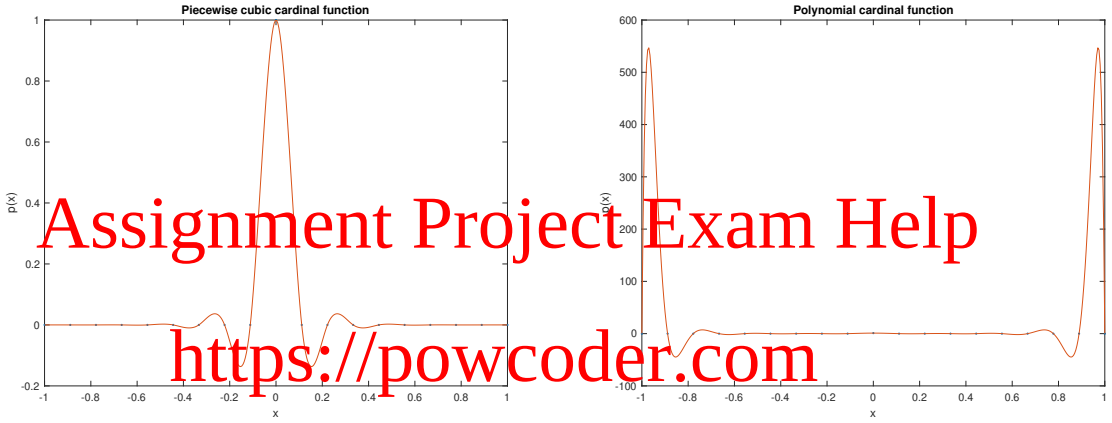

Figure 9.6: Interpolation by cardinal functions

References

[1] Alfio Quarteroni, Riccardo Sacco, and Fausto Saleri. Numerical mathematics (texts in applied mathematics). *Numerical Mathematics (Texts in Applied Mathematics)*, 2006.