Contents

1. Introduction

Interpolation is widely used in various fields of applied math, engineering, and computer science. It is a typical case that data are available only at a discrete set of points, while we would like to have them an an arbitrary point of a region containing this discrete set. The task of interpolation is to define a function in the whole region so that it coincides with the data at this discrete set. Various methods for interpolation are based on different assumptions regarding the function. Linear interpolation results at a continuous function while bicubic interpolation will make the function smooth. Take a look at the figures here.

Interpolation techniques are building blocks for various methods in computational mathematics, in particular, for

- quadrature rules,
- linear multistep ODE solvers with variable time step and order,
- finite element methods for solving partial differential equations.

We will discuss polynomial interpolation and spline interpolation in 1D.

2. Polynomial interpolation

2.1. **Lagrangian interpolation.** Suppose a continuous function $f(x)$ defined on the interval $[a, b]$ is given at a finite number of points x_j, $j = 0, 1, 2, \ldots, n$, $n + 1$ points in total. We will denote $f(x_j$ by f_j. The task of interpolation is to find a polynomial of degree at most n passing through all these points. Lagrange's approach to this task is to construct

$n+1$ polynomials $L_j(x)$ of degree n such that each $L_j(x)$ is 1 at x_j and zero at ll other x_k's. Then the linear combination

$$P(x) = f_0 L_0(x) + f_1 L_1(x) + \ldots + f_n(x) L_n(x)$$

is a polynomial of degree at most n, and $P(x_j) = f_j$ for all $j = 0, 1, \ldots, n$. The polynomial $P(x)$ written in the form above is called the Lagrange interpolation polynomial. The polynomials $L_j(x)$ are easily constructed. The error of interpolation can also be found. These results are summarized in the following theorem.

Theorem 1. *Given a function f that is defined at $n+1$ points $x_0 < x_1 < \ldots < x_n \in [a, b]$ there exists a unique polynomial of degree $\leq n$ such that*

$$P_n(x_j) = f(x_j), \quad j = 0, 1, 2, \ldots, n.$$

This polynomial is given by

$$P_n(x) = \sum_{j=0}^{n} f(x_j) L_j(x),$$

where $L_j(x)$ is defined by

$$L_j(x) = \frac{\pi_{n+1}(x)}{(x - x_j)\pi'_{n+1}(x_j)} = \frac{\prod_{k=0, k\neq j}^{n}(x - x_k)}{\prod_{k=0, k\neq j}^{n}(x_j - x_k)},$$

$\pi_{n+1}(x)$ *being the nodal polynomial*

$$\pi_{n+1}(x) = \prod_{k=0}^{n}(x - x_k).$$

Additionally, if f is $n+1$ times continuously differentiable in (a, b), then for any $x \in [a, b]$ there exists a value $\zeta_x \in (a, b)$ depending on x, such that

$$E_n(x) = f(x) - P_n(x) = \frac{f^{(n+1)}(\zeta_x)}{(n+1)!}\pi_{n+1}(x). \tag{1}$$

Proof. (1) *Existence* By construction, P_n is a polynomial of degree n passing through x_j, $j = 0, 1, 2, \ldots, n$.

(2) *Uniqueness* Suppose that there are two such polynomials, $P_n(x)$ and $Q_n(x)$. Then the polynomial $d(x) := P_n(x) - Q_n(x)$ is at most of degree n. On the other hand, it has at least $n+1$ roots x_j, $j = 0, 1, 2, \ldots, n$. Therefore, it must be identically zero.

(3) *Error formula* Consider the function

$$F(z) = f(z) - P_n(z) - [f(x) - P_n(x)]\frac{\pi_{n+1}(z)}{\pi_{n+1}(x)}.$$

This function has $n+2$ zeros at $z = x_j$, $j = 0, 1, 2, \ldots, n$, and $z = x$. Recall Rolle's theorem that says that if a function $f(z)$ is continuously differentiable on $[a, b]$ and

$f(a) = f(b)$ then there is $c \in (a, b)$ such that $f'(c) = 0$. Therefore, if we apply Rolle's theorem $n+1$ times we get that the function

$$F^{(n+1)}(z) = f^{(n+1)}(z) - P_n^{(n+1)}(z) - [f(x) - P_n(x)]\frac{(n+1)!}{\pi_{n+1}(x)}$$

has at least one zero in (x_0, x_n). We denote this zero by ζ_x. Therefore, taking into account that $P_n^{(n+1)}(z) \equiv 0$, we get

$$0 = f^{(n+1)}(\zeta_x) - [f(x) - P_n(x)]\frac{(n+1)!}{\pi_{n+1}(x)},$$

and Eq. (1) follows.

□

Example Let us find the Lagrange interpolant $p(x)$ passing through the points $(0, f_0)$, $(1, f_1)$, and $(2, f_2)$. The polynomial $p(x)$ is of the form

$$\begin{aligned} p(x) &= f_0\frac{(x-1)(x-2)}{(0-1)(0-2)} + f_1\frac{(x-0)(x-2)}{(1-0)(1-2)} + f_2\frac{(x-0)(x-1)}{(2-0)(2-1)} \\ &= \tfrac{1}{2}f_0(x^2 - 3x + 2) - f_1(x^2 - 2x) + \tfrac{1}{2}f_2(x^2 - x) \\ &= f_0 + \tfrac{1}{2}(-3f_0 + 4f_1 - f_2)x + \tfrac{1}{2}(f_0 - 2f_1 + f_2)x^2. \end{aligned} \tag{2}$$

2.2. **The Newton interpolation polynomial.** Lagrangian interpolation is convenient because it gives an explicit formula for the interpolant. However, it does not provide a convenient way to modify the polynomial to accommodate additional interpolation points. An alternative form of the interpolation polynomial, the Newton form, gives such a way. The Newton interpolation formula is used, for example, for deriving linear multistep methods with varying time step for solving ODE's. The Newton interpolation formula is defined via the divided differences. We set

$$\begin{aligned} f[x_0] &= f(x_0), \\ f[x_0, x_1] &= \frac{f[x_1] - f[x_0]}{x_1 - x_0}, \\ f[x_0, x_1, x_2] &= \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0}, \\ &\dots \\ f[x_0, x_1, \dots, x_k] &= \frac{f[x_1, \dots, x_k] - f[x_0, x_1, \dots, x_{k-1}]}{x_k - x_0}, \\ &\dots . \end{aligned}$$

Theorem 2. *The polynomial interpolating $f(x)$ at x_j, $j = 0, 1, 2, \dots, n$ is given by*

$$\begin{aligned} P_n(x) = f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) + \dots \\ + f[x_0, x_1, \dots, x_n](x - x_0)(x - x_1)\dots(x - x_{n-1}). \end{aligned} \tag{3}$$

Example Let us find the Newton interpolant $p(x)$ passing through the points $(0, f_0)$, $(1, f_1)$, and $(2, f_2)$. The polynomial $p(x)$ is of the form

$$p(x) = f[0] + f[0,1](x-0) + f[0,1,2](x-0)(x-1), \tag{4}$$

where

$$f[0] = f_0, \quad f[0,1] = \frac{f_1 - f_0}{1-0} = f_1 - f_0, \quad f[1,2] = \frac{f_2 - f_1}{2-1} = f_2 - f_1,$$

$$f[0,1,2] = \frac{f[1,2] - f[0,1]}{2-0} = \frac{1}{2}(f_2 - f_1 - (f_1 + f_0)) = \frac{1}{2}(f_0 - 2f_1 + f_2).$$

Plugging these coefficients into Eq. (4) we get

$$\begin{aligned} p(x) &= f_0 + (f_1 - f_0)x + \tfrac{1}{2}(f_0 - 2f_1 + f_2)x(x-1) \\ &= f_0 + \tfrac{1}{2}\left(-3f_0 + 4f_1 - f_2\right)x + \tfrac{1}{2}(f_0 - 2f_1 + f_2)x^2. \end{aligned} \tag{5}$$

The polynomial $p(x)$ in Eq. (5) coincides with the one in Eq. (2) as it should.

Proof. We will proceed by induction. For $n = 1$ Eq. (3) holds. Suppose the polynomials $P[x_0, \ldots, x_{n-1}](x)$ and $P[x_1, \ldots, x_n](x)$ of the form of Eq. (3) interpolate f at the points $x_0, \ldots, x_{n-1}$ and $x_1, \ldots, x_n$ respectively. Note that both of them are of degree $n-1$. Hence they differ by a polynomial of degree at most $n-1$, and this polynomial has zeros at x_1, ..., x_{n-1}. Therefore,

$$P[x_1, \ldots, x_n](x) - P[x_0, \ldots, x_{n-1}](x) = a(x - x_1)\ldots(x - x_{n-1})$$

where a is a number. Obviously, a is the difference of the leading coefficients of the polynomials $P[x_0, \ldots, x_{n-1}](x)$ and $P[x_1, \ldots, x_n](x)$, i.e.,

$$a = f[x_1, \ldots, x_n] - f[x_0, x_1, \ldots, x_{n-1}].$$

At the same time,

$$a = \frac{P[x_1, \ldots, x_n](x) - P[x_0, \ldots, x_{n-1}](x)}{(x - x_1)\ldots(x - x_{n-1})}.$$

Now consider the polynomial

$$P[x_0, x_1, \ldots, x_n](x) = P[x_0, \ldots, x_{n-1}](x) + \frac{a}{x_n - x_0}(x - x_0)(x - x_1)\ldots(x - x_{n-1}). \tag{6}$$

The last term of this polynomial is chosen so that it is zero at $x_0, \ldots, x_{n-1}$, and the coefficient $a/(x_n - x_0)$ is our guess that we will verify below. By construction, $P[x_0, x_1, \ldots, x_n](x)$ interpolates f at $x_0, \ldots, x_{n-1}$. Let us verify that it also does so at $x = x_n$. We evaluate

$P[x_0, x_1, \ldots, x_n](x)$ at x_n and obtain:

$$\begin{aligned}
&P[x_0, x_1, \ldots, x_n](x_n)\\
=&P[x_0, \ldots, x_{n-1}](x_n) + \frac{a}{x_n - x_0}(x_n - x_0)(x_n - x_1)\ldots(x_n - x_{n-1})\\
=&P[x_0, \ldots, x_{n-1}](x_n)\\
+&\frac{P[x_1, \ldots, x_n](x_n) - P[x_0, \ldots, x_{n-1}](x_n)}{(x_n - x_0)(x_n - x_1)\ldots(x_n - x_{n-1})}(x_n - x_0)(x_n - x_1)\ldots(x_n - x_{n-1})\\
=&P[x_1, \ldots, x_n](x_n) = f(x_n).
\end{aligned}$$

Therefore, the polynomial given by Eq. (6) interpolates f at x_j, $j = 0, 1, 2, \ldots, n$. □

Remark The function $f[x_0, \ldots, x_n]$ is a symmetric function of its arguments i.e., it does not change if we permute x_0, ..., x_n. This is because the interpolation polynomial is independent of the order of nodes.

2.3. **Hermite interpolation.** We have learned how to construct a polynomial of degree at most n matching with f at $n+1$ points x_0, x_1, ..., x_n. Now suppose that we want to construct a polynomial such that its values and its derivatives match with those of f at given points. For example, suppose we want to find a minimal degree polynomial $\mathcal{H}$ on the interval $[0, 1]$ such that $\mathcal{H}(0) = u_0$, $\mathcal{H}'(0) = u_0'$, $\mathcal{H}(1) = u_1$, $\mathcal{H}'(1) = u_1'$. Following Lagrange's idea, we design a polynomial $p_0(x)$ such that

$$p_0(0) = 1, \quad p_0'(0) = 0, \quad p_0(1) = 0, \quad p_0'(1) = 0.$$

Note that then the polynomial $p_1(x) := p_0(1 - x)$ satisfies:

$$p_1(0) = 0, \quad p_1'(0) = 0, \quad p_1(1) = 1, \quad p_1'(1) = 0.$$

We also need to design a polynomial $g_0(x)$ such that

$$g_0(0) = 0, \quad g_0'(0) = 1, \quad g_0(1) = 0, \quad g_0'(1) = 0.$$

Then the polynomial $g_1(x) := -g_0(1 - x)$ satisfies

$$g_1(0) = 0, \quad g_1'(0) = 0, \quad g_1(1) = 0, \quad g_1'(1) = 1.$$

For each polynomial $p_0(x)$ and $g_0(x)$ we have four conditions to meet. Hence they should have four coefficients which means that they should be cubic. We can find $p_0(x)$ by setting it to

$$p_0(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3, \quad \text{and, respectively,} \quad p_0'(x) = a_1 + 2a_2 x + 3a_3 x^2,$$

and solving the system of four linear equations:

$$\begin{aligned}
p_0(0) &= a_0 = 1,\\
p_0'(0) &= a_1 = 0,\\
p_0(1) &= a_0 + a_1 + a_2 + a_3 = 0,\\
p_0'(1) &= a_1 + 2a_2 + 3a_3 = 0.
\end{aligned}$$

We find:

$$p_0(x) = 1 - 3x^2 + 2x^3.$$

Similarly, we find

$$g_0(x) = x(1-x)^2.$$

Now we write out the desired interpolating polynomial:

$$\mathcal{H}(x) = u_0 p_0(x) + u_1 p_0(1-x) + u_0' g_0(x) - u_1' g_0(1-x). \tag{7}$$

The problem that we have just solved can be generalized to an arbitrary number of points x_j and arbitrary numbers of matching derivatives of f and the interpolation polynomial at each. The existence, uniqueness , and error estimate for such an interpolant are stated in the theorem below.

Theorem 3. *Let f be n times continuously differentiable in $[a,b]$ and $n+1$ times continuously differentiable in (a,b). Let $x_0 < x_1 < \ldots < x_k \in [a,b]$, and let $n_i \in \mathbb{N}$ be such that $n_0 + n_1 + \ldots + n_k = n+1$. Then there exists a unique polynomial P_n of degree $\leq n$ such that*

$$P_n^{(j)}(x_i) = f^{(j)}(x_i), \quad j = 0, 1, \ldots, n_i - 1, \quad i = 0, 1, \ldots, k.$$

Furthermore, given $x \in [a,b]$, there exists a value $\zeta_x \in (a,b)$ such that

$$f(x) = P_n(x) + \frac{f^{(n+1)}(\zeta_x)}{(n+1)!}\pi_{n+1}(x),$$

where $\pi_{n+1}(x)$ is the nodal polynomial

$$\pi_{n+1}(x) = (x - x_0)^{n_0} \ldots (x - x_k)^{n_k}.$$

Remark If all of the nodes coincide, i.e., $k = 0$ and $n_0 = n$ then the polynomial P_n is the Taylor polynomial and the error term is the error term for the Taylor series in the Lagrange form.

An explicit formula for such a polynomial exists but it is not as simple as for the Lagrangian interpolant. As in the case of distinct nodes, it is more convenient to calculate the Hermite interpolant using divided differences. However, we will need to generalize them.

Theorem 4. *Let*

$$t_0 \leq t_1 \leq \ldots \leq t_n,$$

where

$$t_0 = \ldots = t_{n_0} = x_0, \quad t_{n_0+1} = \ldots = t_{n_0+n_1+1} = x_1, \quad \ldots.$$

The points t_j are called the generalized abscissas.

The generalized divided differences are defined as follows. If $t_i = \ldots = t_{i+p} = x_l$ then

$$f[t_i, t_{i+1}, \ldots, t_{i+p}] = \frac{1}{p!} f_l^{(p)}.$$

If $t_i < t_{i+p}$ then

$$f[t_i, t_{i+1}, \ldots, t_{i+p}] = \frac{f[t_{i+1}, t_{i+2}, \ldots, t_{i+p}] - f[t_i, t_{i+1}, \ldots, t_{i+p-1}]}{t_{i+p} - t_i}.$$

The interpolation polynomial is given by

$$P_n(x) = f[t_0] + f[t_0, t_1](x - t_0) + \ldots + f[t_0, \ldots, t_n](x - t_0)\ldots(x - t_{n-1}). \tag{8}$$

Proof. The proof is by induction. The statement holds for $n = 0$. Suppose it holds for $n - 1$. Then the polynomials $P[t_0, \ldots, t_{n-1}](x)$ and $P[t_1, \ldots, t_n](x)$ of the form of Eq. (8) interpolate f at the abscissas $t_0 \leq \ldots \leq t_{n-1}$ and $t_1 \leq \ldots \leq t_n$ respectively. The difference between them is a polynomial $Q(x)$ of degree $n - 1$ with zeros at t_1, ..., t_{n-1}. Hence it is of the form

$$Q(x) = a(x - t_1)\ldots(x - t_{n-1}).$$

The coefficient a is equal, on one hand, to the difference between the leading coefficients of $P[t_1, \ldots, t_n](x)$ and $P[t_0, \ldots, t_{n-1}](x)$, i.e.,

$$a = f[t_1, \ldots, t_n] - f[t_0, \ldots, t_{n-1}],$$

on the other hand, it is equal to

$$a = \frac{P[t_1, \ldots, t_n](x) - P[t_0, \ldots, t_{n-1}](x)}{(x - t_1)\ldots(x - t_{n-1})}.$$

Consider the polynomial

$$P[t_0, \ldots, t_n](x) = P[t_0, \ldots, t_{n-1}](x) + b(x - t_0)\ldots(x - t_{n-1}).$$

We will distinguish two cases.

Case 1: $t_n = t_{n-1} = \ldots = t_{n-n_k}$. By construction, it interpolates f at the abscissas t_0, ..., t_{n-1}. Indeed,

$$(x - t_0)\ldots(x - t_{n-1}) = (x - x_0)^{n_0}\ldots(x - x_k)^{n_k - 1},$$

and it vanishes together with its n_i derivatives at x_0, ..., x_{k-1} and together with its $n_k - 1$ derivatives at x_k. We set $b = \frac{1}{n_k!} f^{(n_k)}$. Then

$$P[t_0, \ldots, t_n]^{(n_k)}(x_k) = b n_k! = f^{(n_k)}.$$

Case 2: $t_n > t_{n-1}$. We set

$$b = \frac{a}{t_n - t_0}$$

and proceed as in the proof of the Newton interpolant formula. This completes the proof. □

Example Suppose we are given $f(x_0) = f_0$, $f(x_1) = f_1$, $f'(x_1) = f'_1$, and $f''(x_1) = f''_1$. We need to write the interpolation polynomial $P(x)$ such that $P(x_0) = f_0$, $P^{(k)}(x_1) = f^{(k)}(x_1)$,

$k = 0, 1, 2$. We calculate the generalized divided differences:

$$
\begin{array}{l|l|l|l|l}
t_0 = x_0 & f[t_0] = f_0 & & & \\
 & & f[t_0, t_1] = \frac{f_1 - f_0}{x_1 - x_0} & & \\
t_1 = x_1 & f[t_1] = f_1 & & f[t_0, t_1, t_2] = \frac{f_1' - \frac{f_1 - f_0}{x_1 - x_0}}{x_1 - x_0} & \\
 & & f[t_1, t_2] = \frac{1}{1!} f_1' & & f[t_0, t_1, t_2, t_3] = \frac{\frac{1}{2} f_1'' - \frac{f_1' - \frac{f_1 - f_0}{x_1 - x_0}}{x_1 - x_0}}{x_1 - x_0} \\
t_2 = x_1 & f[t_2] = f_1 & & f[t_1, t_2, t_3] = \frac{1}{2} f_1'' & \\
 & & f[t_2, t_3] = \frac{1}{1!} f_1' & & \\
t_3 = x_1 & f[t_3] = f_1 & & &
\end{array}
$$

The interpolation polynomial is given by

$$P(x) = f_0 + f_1(x - x_0) + \frac{f_1' - \frac{f_1 - f_0}{x_1 - x_0}}{x_1 - x_0}(x - x_0)(x - x_1) + \frac{\frac{1}{2} f_1'' - \frac{f_1' - \frac{f_1 - f_0}{x_1 - x_0}}{x_1 - x_0}}{x_1 - x_0}(x - x_0)(x - x_1)^2.$$

The error is given by

$$E(x) \equiv f(x) - P(x) = \frac{f^{(4)}(\zeta)}{24}(x - x_0)(x - x_1)^3 \quad \text{for some } \zeta \in (x_0, x_1).$$

Example Let $f(x) = (1 + x^2)^{-1}$. This function is called the "Witch of Agnesi". It is a well-known example demonstrating the fault of high degree polynomial interpolation with equispaced points on a large interval. This will be discussed in Section 2.4. Here we take the interval $[0, 1]$ where the interpolation works pretty nicely and compare two interpolation polynomials: Newton's interpolant with four equispaced points, and the Hermite interpolant with the abscissas $0, 0, 1, 1$. The Matlab program below computes these interpolants and generates Fig. 1.

```
 function CompareInterpolants()
% Compares two cubic interpolating polynomials for 1/(x^2 + 1) on [0,1]:
% Newton's interpolant with four equispaced points and
% Hermite interpolant with abscissas 0, 0, 1, 1.
x = [0,1/3,2/3,1]'; % four equispaced points
t = linspace(0,1,100);
f = @(x)1./(x.^2 + 1); % Witch of Agnesi
fx = f(x);
ft = f(t);
figure;
hold on; grid;
plot(x,fx,'.','Markersize',30);
plot(t,ft,'Linewidth',1)
%% Newton's interpolation.
% compute divided differences
dd = zeros(4);
dd(:,1) = fx;
```

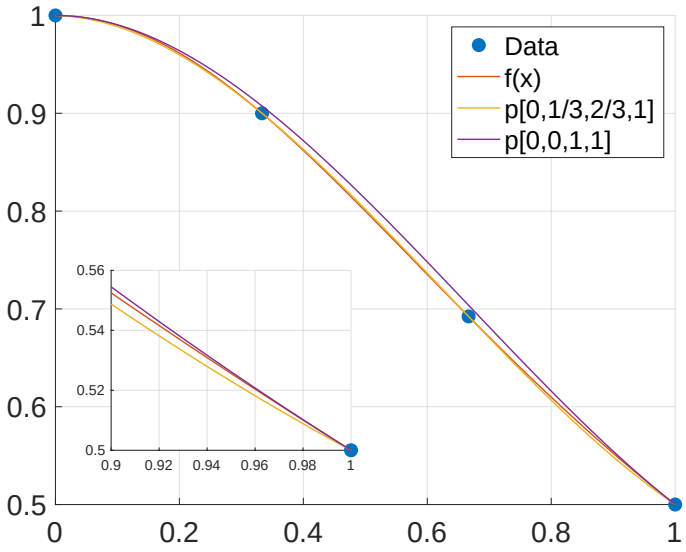

FIGURE 1. Comparison of two cubic interpolation polynomials: Newton's polynomial with four equispaced points $p[0, 1/3, 2/3, 1]$, and the Hermite polynomial with abscissas $t_0 = t_1 = 0$, $t_2 = t_3 = 1$: $p[0, 0, 1, 1]$.

```
dd(1:3,2) = (dd(2:4,1) - dd(1:3,1))./(x(2:4)-x(1:3));
dd(1:2,3) = (dd(2:3,2) - dd(1:2,2))./(x(3:4)-x(1:2));
dd(1,4) = (dd(2,3) - dd(1,3))./(x(4)-x(1));
% define Newton's interpolant
p = @(t) dd(1,1) + dd(1,2)*(t-x(1)) + dd(1,3)*(t-x(1)).*(t-x(2))...
    + dd(1,4)*(t-x(1)).*(t-x(2)).*(t-x(3));
pt = p(t);
plot(t,pt,'Linewidth',1);
%% Hermite interpolation
fp =  @(x)-2*x./(1+x.^2).^2; % derivative of f(x)
p0 = @(x)1-3*x.^2 + 2*x.^3; % p0(0) = 1, p0'(0) = p0(1) = p0'(1) = 0
g0 = @(x)x.*(1-x).^2; % g0(0) = g0(1) = g0'(1) = 0; g0'(0) = 1
% define Hermite interpolant
H = @(t)f(0)*p0(t) + fp(0)*g0(t) + f(1)*p0(1-t) - fp(1)*g0(1-t);
Ht = H(t);
plot(t,Ht,'Linewidth',1)
set(gca,'Fontsize',20);
legend('Data','f(x)','p[0,1/3,2/3,1]','p[0,0,1,1]');
%% zoom in near x = 1
ti = linspace(0.9,1,20);
```

```
ax2 = axes('Position',[.2 .2 .3 .3]);
hold(ax2,'on');
hold on; grid;
plot(x(4),fx(4),'.','Markersize',30);
plot(ti,f(ti),'Linewidth',1)
plot(ti,p(ti),'Linewidth',1)
plot(ti,H(ti),'Linewidth',1)
end
```

2.4. **Runge phenomenon.** In this section, we address the question whether the interpolation polynomial approaches f in the uniform norm as the number of nodes tends to infinity. The answer is negative in general.

Exercise (1) Build interpolation polynomials with n equispaced interpolation points in the interval $x \in [-1, 1]$ for $f(x) = |x|$ and show that the sequence with equally spaced nodes diverges as $n \to \infty$.

(2) Do the same for the function the "Witch of Agnesi"

$$f(x) = \frac{1}{1+x^2}, \quad x \in [-5, 5]. \tag{9}$$

The graph of the Witch of Agnesi and its polynomial interpolants with equispaced points of degrees 3, 6, 9, and 12 on the interval $[-5, 5]$ are shown in Fig. 2. We see that as we increase the degree of the interpolation polynomial, the interpolation error blows up. Fig. 2 was generated by the following Matlab code:

```
function RungePhenomenon()
% Demonstrates the Runge phenomenon on 1/(x^2 + 1) on [-5,5]:
a = -5; b = 5;
figure;
hold on; grid;
f = @(x)1./(x.^2 + 1); % Witch of Agnesi
t = linspace(a,b,400);
plot(t,f(t),'Linewidth',1)
set(gca,'Fontsize',20);
legend('f(x) = (1+x^2)^{-1}');
% build Newton's interpolants with equispaced points
for n = 3 : 3 : 12
    x = linspace(a,b,n + 1)'; % n+1 equispaced points
    fx = f(x);
    %% Newton's interpolation.
    % compute divided differences
    dd = zeros(n + 1);
    dd(:,1) = fx;
    for k = 2 : n + 1
```

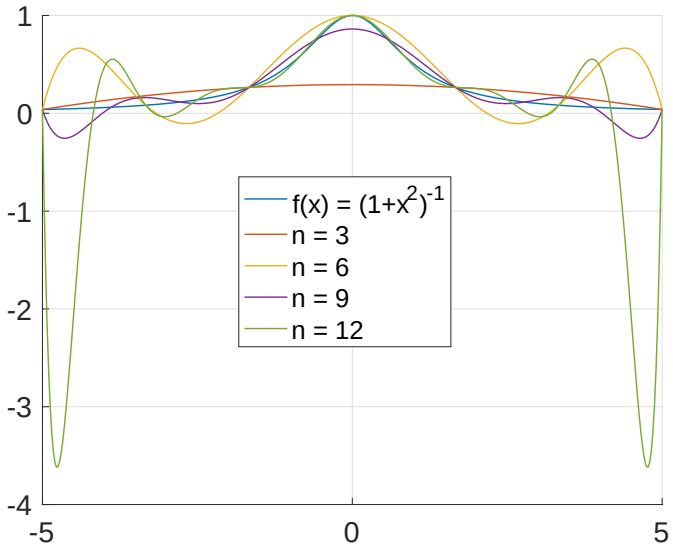

FIGURE 2. A demonstration of the Runge phenomenon. Polynomial interpolants with equispaced points of degrees 3, 6, 9, and 12 for $f(x) = (1+x^2)^{-1}$ on the interval $[-5, 5]$.

```
        dd(1:n-k+2,k) = (dd(2:n-k+3,k-1)-dd(1:n-k+2,k-1))./(x(k:n+1)-x(1:n-k+2));
    end
    % evaluate Newton's interpolant
    pt = dd(1,n+1);
    for k = n : -1 : 1
        pt = pt.*(t - x(k)) + dd(1,k);
    end
    plot(t,pt,'Linewidth',1,'DisplayName',sprintf('n = %d',n));
end
end
```

The lack of convergence in the first example is less surprising then in the second one, because the function in Eq. (9) is infinitely differentiable. This phenomenon was studied by Runge and is known as the *Runge phenomenon*.

The analysis explaining the Runge phenomenon uses theory of functions of complex variable.

Let us look at the error formula (1). There are two factors that can be large: the fraction $\frac{f^{(n+1)}}{(n+1)!}$ and the nodal polynomial $\pi_{n+1}(x)$.

The fraction $\frac{f^{(n+1)}}{(n+1)!}$ does not need to decay as $n \to \infty$. The estimate comes from the Cauchy integral formula

$$f^{(n+1)}(\zeta) = \frac{(n+1)!}{2\pi i} \int_{C_R} \frac{f(z)}{(z-\zeta)^{n+2}} dz, \tag{10}$$

where $C_R = \{z \in \mathbb{C} \mid |z - \zeta| = R\}$. Therefore,

$$\frac{\max_{x\in[\zeta-R,\zeta+R]} |f^{(n+1)}(x)|}{(n+1)!} \leq \frac{\max_{z\in C_R} |f(z)|}{2\pi R^{n+1}}.$$

Hence, what matters is not smoothness of f but its analyticity. Note that the Witch of Agnesi has poles at $z = \pm i$ (i.e. discontinuities of the form $a/(z-b)$). Therefore, the formula right above gives the upper bound for $\frac{f^{(n+1)}}{(n+1)!}$ equal to infinity.

The nodal polynomial $\pi_{n+1}(x)$ is highly oscillatory for equally spaced nodes. It can be shown that for large n it takes values approximately 2^n times larger near the endpoints than in the middle. However, if we pick the nodes nonuniformly, we can make the nodal polynomial deviate from zero as little as possible. The optimal choice of nodes is Chebychev's. A very nice explanation for these phenomena is found in L. N. Trefethen's book Spectral Methods in Matlab – read the beginning of Chapter 5.

In this connection, it becomes even surprising that the polynomial interpolants with equispaced nodes converge to the Witch of Agnesi on an interval much larger than $[-1, 1]$: on $[-a, a]$ where $a \approx 3.63$.

3. Chebyshev polynomials

We will study the Chebyshev polynomials in more details as they lead to a number of remarkable numerical tools:

- Chebyshev interpolation, where the Runge phenomenon is eliminated. Ref.: A. Gil, J. Segura, N. Temme, Numerical Methods for Special Functions, SIAM, 2007 available online via UMD library;
- Chebyshev least squares approximation which is close to the minimax one;
- Chebyshev spectral methods for solving PDEs with non-periodic boundary conditions – L. N. Trefethen's book Spectral Methods in Matlab.

Chebyshev polynomials are defined as follows:

$$T_n(x) = \cos\left[n \arccos(x)\right], \quad x \in [-1, 1], \quad n = 0, 1, 2, \ldots. \tag{11}$$

It follows from the definition that

$$T_n(\cos\theta) = \cos(n\theta), \quad \theta \in [0, \pi], \quad n = 0, 1, 2, \ldots. \tag{12}$$

3.1. **Properties of the Chebyshev polynomials.**

(A) The Chebyshev polynomials satisfy the following three-term recurrence relationships (TTRR):

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x), \quad T_0(x) = 1, \quad T_1(x) = x, \quad n = 0, 1, 2, \ldots. \tag{13}$$

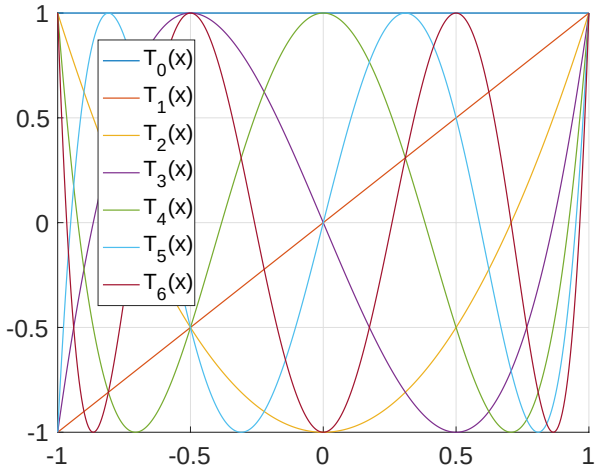

FIGURE 3. Graphs of the Chebyshev polynomials $T_0(x)$, $T_1(x)$, ..., $T_6(x)$.

This TTRR immediately follows from Eq. (43) and the trigonometric formula

$$\cos(a) + \cos(b) = 2\cos\frac{a+b}{2}\cos\frac{a-b}{2},$$

where we set $a = (n+1)\theta$ and $b = (n-1)\theta$.

(B) The leading coefficient of $T_n(x)$ is 2^{n-1}. This follows from Eq. (13).

(C)

$$T_n(-x) = (-1)^n T_n(x).$$

This follows from the formulas

$$\arccos(-x) = \pi - \arccos(x) \text{ and } \cos(\pi n - x) = (-1)^n \cos x.$$

(D) Zeros of $T_n(x)$ are

$$x_k = \cos\left(\frac{k+\frac{1}{2}}{n}\pi\right), \quad k = 0, 1, \ldots, n-1. \tag{14}$$

Extrema of $T_n(x)$ are

$$x'_k = \cos\left(\frac{\pi k}{n}\right), \quad k = 0, 1, \ldots, n. \tag{15}$$

$$T_n(x'_k) = (-1)^k.$$

(E) The deviation of $2^{-n}T_{n+1}(x)$ from zero on $[-1, 1]$ is minimal possible among all polynomials with leading coefficient 1 of degree $n+1$. The theorem establishing this fact is below.

Theorem 5. *Let*

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

Then the monic polynomial (i.e., its leading coefficient is 1)

$$\hat{T}_{n+1} = \prod_{k=0}^{n}(x - x_k)$$

of degree $n+1$ has the smallest possible uniform (maximum) norm 2^{-n} in $[-1,1]$ among all polynomials of degree $n+1$. I.e.,

$$2^{-n} = \max_{x\in[-1,1]} |\hat{T}_{n+1}(x)| = \min_{\substack{p\in\mathcal{P}_{n+1}\\ p=x^{n+1}+\ldots}} \max_{x\in[-1,1]} |p(x)|.$$

Proof. Suppose there is a monic polynomial $p(x)$ of degree $n+1$ such that $|p(x)| < 2^{-n}$ for all $x \in [-1,1]$. Let x'_k, $k = 0, 1, \ldots, n+1$ be the abscissas of the extreme values of Chebyshev polynomials of degree $n+1$. Hence we have

$$\begin{aligned} p(x'_0) &< 2^{-n}T_{n+1}(x'_0),\\ p(x'_1) &> 2^{-n}T_{n+1}(x'_1),\\ p(x'_2) &< 2^{-n}T_{n+1}(x'_2),\\ &\ldots. \end{aligned}$$

Therefore the polynomial

$$Q(x) = p(x) - 2^{-n}T_{n+1}$$

changes sign between each two consecutive extrema of T_{n+1}. $T_{n+1}(x)$ has $n+2$ extrema on $[-1,1]$. Hence $Q(x)$ has $n+1$ zeros. But $Q(x)$ is of degree $\le n$. Thus we have arrived to a contradiction. Hence there is no such monic polynomial p of degree $n+1$ such that $|p(x)| < 2^{-n}$ for $x \in [-1,1]$. □

(F) Relations with derivatives.

$$T_0(x) = T'_1(x), \tag{16}$$

$$T_1(x) = \frac{1}{4}T'_2(x), \tag{17}$$

$$T_n(x) = \frac{1}{2}\left(\frac{T'_{n+1}(x)}{n+1} - \frac{T'_{n-1}(x)}{n-1}\right), \quad n \ge 2. \tag{18}$$

The first two equalities are easy -to-check. The last one can be proven from the following observation:

$$T'_n(x) = \frac{n\sin(n\theta)}{\sin\theta}, \quad \text{where} \quad x = \cos\theta.$$

Exercise Prove Eq. (18).

(G) Multiplication relationship:

$$2T_r(x)T_q(x) = T_{r+q}(x) + T_{|r-q|}(x). \tag{19}$$

Exercise Prove Eq. (19).

(H) Orthogonality relationship:

$$\int_{-1}^{1} \frac{T_r(x)T_s(x)}{\sqrt{1-x^2}}dx = \begin{cases} 0, & r \neq s \\ \pi, & r = s = 0, \\ \frac{\pi}{2}, & r = s \neq 0. \end{cases} \tag{20}$$

Let us prove it. We make a variable change $x = \cos\theta$. Then $dx = -\sin\theta d\theta$, $T_r(x) = \cos(r\theta)$, $T_s(x) = \cos(s\theta)$, and the integration limits are changed to: $-1 \mapsto \pi$, $1 \mapsto 0$. Therefore, using the formula

$$\cos a \cos b = {}^1\!/_2 \cos(a+b) + {}^1\!/_2 \cos(a-b)$$

we get:

$$\begin{aligned} \int_{-1}^{1} \frac{T_r(x)T_s(x)}{\sqrt{1-x^2}}dx &= \int_0^{\pi} \cos(r\theta)\cos(s\theta)d\theta = \frac{1}{2}\int_0^{\pi} \left[\cos(r+s)\theta + \cos(r-s)\theta\right] d\theta \\ &= \frac{1}{2}\begin{cases} \left[\frac{\sin(r+s)\theta}{r+s} + \frac{\sin(r-s)\theta}{r-s}\right]_0^{\pi}, & r \neq s \\ 2\pi, & r = s = 0, \\ \pi + \left[\frac{\sin(2r\theta)}{2r}\right]_0^{\pi}, & r = s \neq 0. \end{cases} \\ &= \begin{cases} 0, & r \neq s \\ \pi, & r = s = 0, \\ \frac{\pi}{2}, & r = s \neq 0. \end{cases} \end{aligned}$$

(I) Discrete orthogonality relationship: Take the points

$$x_j = \cos\left(\frac{\pi(j+\frac{1}{2})}{n+1}\right), \quad j = 0, 1, \ldots, n$$

that are the zeros of $T_{n+1}(x)$. Then for all $0 \leq r, s \leq n$ we have

$$\sum_{j=0}^{n} T_r(x_j)T_s(x_j) = \begin{cases} 0, & r \neq s \\ n+1, & r = s = 0, \quad j = 0, 1, \ldots, n. \\ \frac{n+1}{2}, & r = s \neq 0, \end{cases} \tag{21}$$

Exercise Prove Eq. (25).

4. Chebyshev interpolation

Properties of the Chebyshev polynomials offer a nice way for computing the Chebyshev interpolant of degree n. Fix some integer n and consider the zeros of $T_{n+1}(x)$. They are

$$x_j = \cos\left(\frac{\pi(j+\frac{1}{2})}{n+1}\right), \quad j = 0, 1, \ldots, n.$$

For a given function $f(x)$ on the interval $[-1, 1]$, the polynomial p_n of degree n interpolating $f(x)$ at the Chebyshev points x_j, $j = 0, 1, \ldots, n$, is given by

$$p_n(x) = {\sum_{k=0}^{n}}' c_k T_k(x) \equiv \frac{c_0}{2} + \sum_{k=1}^{n} c_k T_k(x).$$

The symbol $'$ indicates that the first term in the sum should be divided by two. The coefficients c_k are found from the requirement that $p_n(x_j) = f(x_j)$, $j = 1, 2, \ldots, n$, i.e.,

$$f(x_j) = {\sum_{k=0}^{n}}' c_k T_k(x_j).$$

Then we have

$$\sum_{j=0}^{n} f(x_j) T_m(x_j) = {\sum_{k=0}^{n}}' c_k \sum_{j=0}^{n} T_k(x_j) T_m(x_j) = {\sum_{k=0}^{n}}' c_k \frac{n+1}{2} \delta_{mk} = \tfrac{1}{2}(n+1) c_k.$$

Hence the coefficients are given by

$$c_k = \frac{2}{n+1} \sum_{j=0}^{n} f(x_j) T_k(x_j), \quad x_j = \cos\left(\frac{\pi(j+\frac{1}{2})}{n+1}\right). \tag{22}$$

To summarize, the Chebyshev interplant of $f(x)$ is given by

$$p_n(x) = \frac{c_0}{2} + \sum_{k=1}^{n} c_k T_k(x), \tag{23}$$

where the coefficients c_k are given by Eq. (22).

4.1. Chebyshev polynomials shifted to the interval $[a, b]$.

Suppose we need to find the Chebyshev interpolant for $f(x)$ on the interval $[a, b]$ rather than on $[-1, 1]$. Then we need to shift Chebyshev polynomials to this interval using a linear transformation $l : [a, b] \to [-1, 1]$ such that $l(a) = -1$ and $l(b) = 1$. If $x \in [a, b]$ and $y \in [-1, 1]$ then

$$y = l(x) = \frac{x - \frac{a+b}{2}}{\frac{b-a}{2}} = \frac{2x - a - b}{b - a}.$$

Then the shifted Chebyshev polynomials to the interval $[a, b]$ $T_k^{[a,b]}(x)$ are defined by

$$T_k^{[a,b]}(x) = T_k(y) \equiv T_k\left(\frac{2x - a - b}{b - a}\right). \tag{24}$$

Exercise Show that the shifted Chebyshev polynomials satisfy the following orthogonality relationships

$$\int_a^b \frac{T_r^{[a,b]}(x)T_s^{[a,b]}(x)}{\sqrt{1-\left(\frac{2x-a-b}{b-a}\right)^2}}dx = \begin{cases} 0, & r \neq s \\ \frac{b-a}{2}\pi, & r = s = 0, \\ \frac{(b-a)\pi}{4}, & r = s \neq 0. \end{cases} \tag{25}$$

The inverse linear transformation for l is l^{-1}:

$$l^{-1}(y) = \tfrac{1}{2}(y(b-a)+b+a).$$

4.2. **Computing coefficients for Chebyshev interpolant.** Suppose first that $x \in [-1,1]$. In order to compute the coefficients given by Eq. (22), it is convenient to introduce angles

$$\theta_j = \frac{\pi(j+\frac{1}{2})}{n+1}, \quad , j = 0, 1, \ldots, n.$$

Then, recalling that $T_k(x) \equiv T_k(\cos\theta) = \cos(k\theta)$, we get

$$c_k = \frac{2}{n+1}\sum_{j=0}^{n} f(\cos\theta_j)\cos(k\theta_j). \tag{26}$$

If $x \in [a,b]$, we need to evaluate f in Eq. (26) at the points

$$x_j = l^{-1}(y_j) = \tfrac{1}{2}(y_j(b-a)+b+a) = \tfrac{1}{2}(\cos(\theta_j)(b-a)+b+a).$$

In this case we get

$$c_k = \frac{2}{n+1}\sum_{j=0}^{n} f(x_j)\cos(k\theta_j), \text{ where } x_j = \tfrac{1}{2}(\cos(\theta_j)(b-a)+b+a). \tag{27}$$

4.3. **Evaluating Chebyshev interpolant.** Chebyshev interpolant can be evaluated at the point $x \in [a,b]$ directly from Eq. (23)

$$p_n(x) = \frac{c_0}{2} + \sum_{k=1}^{n} c_k \cos(k\arccos(y)), \text{ where } y = \frac{2x-a-b}{b-a}. \tag{28}$$

This formula has a shortcoming that it requires evaluation of cos and arccos which are typically built-in functions, but their evaluation is time-consuming in comparison with basic floating point operations.

An elegant way for evaluation of Chebyshev interpolant that avoids calculations of cos and arccos was proposed by Clenshaw (1955). A detailed description of Clenshaw's algorithm is given in Chapter 3 "Chebyshev Expansions" from "Numerical Methods for Special Functions" by Amparo Gil, Javier Segura, and Nico Temme (see pages 75-76). Here we show how one can implement it in Matlab.

Suppose we need to evaluate the sum

$$p_n(x) = \frac{c_0}{2} + \sum_{k=1}^{n} c_k T_k(y), \quad \text{where} \quad y = \frac{2x-a-b}{b-a}.$$

We rewrite this sum in the vector form:

$$p_n(x) = \mathbf{c}^\top \mathbf{t} - \frac{c_0}{2},$$

where $\mathbf{c} := [c_0, c_1, \ldots, c_n]^\top$, $\mathbf{t} := [T_0(y), T_1(y), \ldots, T_n(y)]^\top$. Recall that the Chebyshev polynomials satisfy TTRR (13) that can be written in the matrix form as

$$\begin{bmatrix} 1 & & & & & \\ -2y & 1 & & & & \\ 1 & -2y & 1 & & & \\ & 1 & -2y & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2y & 1 \end{bmatrix} \begin{bmatrix} T_0(y) \\ T_1(y) \\ T_2(y) \\ \vdots \\ T_n(y) \end{bmatrix} = \begin{bmatrix} 1 \\ -y \\ 0 \\ \vdots \\ 0 \end{bmatrix} \equiv A\mathbf{t} = \mathbf{d}.$$

Let $\mathbf{b}$ be a vector such that

$$\mathbf{b}^\top A = \mathbf{c}^\top, \quad \text{or} \quad A^\top \mathbf{b} = \mathbf{c}. \tag{29}$$

Note that $\mathbf{b}$ is readily found starting from b_n as $A^\top$ is upper-triangular. Then

$$\begin{aligned} p_n(x) = \mathbf{c}^\top \mathbf{t} - \frac{c_0}{2} &= \mathbf{b}^\top A \mathbf{t} - \frac{c_0}{2} \\ &= \mathbf{b}^\top \mathbf{d} - \frac{c_0}{2} = b_0 - b_1 y - \frac{c_0}{2}. \end{aligned}$$

From (29) we find

$$c_0 = b_0 - 2yb_1 + b_2.$$

Therefore,

$$p_n(x) = b_0 - b_1 y - \frac{c_0}{2} = b_0 - b_1 y - \frac{1}{2}(b_0 - 2yb_1 + b_2) = \frac{1}{2}(b_0 - b_2). \tag{30}$$

A Matlab code computing Chebyshev's coefficients and evaluating Chebyshev's sum for $f(x) = (1+x^2)^{-1}$, $x \in [-5, 5]$ is given below.

```
function ChebClenshaw()
% Compute Chebyshev coefficients and evaluate Chebyshev sum using
% Clenshaw's method
a = -5;
b = 5;
nt = 200;
t = linspace(a,b,nt);
f = @(x)1./(x.^2 + 1); % Witch of Agnesi
ft = f(t);
%% Chebyshev grid
n = 30;
```

```
g = [0:n]';
theta = pi*(g + 0.5)/(n + 1) % angles theta
y = cos(theta); % grid on [-1,1]
x = 0.5*(y*(b-a) + a + b); % grid on [a,b]
%% compute Chebyshev coefficients
c = zeros(n+1,1);
fx = f(x);
for k = 0 : n
    c(k + 1) = sum(fx.*cos(k*theta));
end
c = 2*c/(n + 1);
fprintf('c = [\n');
fprintf('%d\n',c);
fprintf(']\n');
%% evaluate Chebyshev sum at points t using Clenshaw's method
e = ones(n+1,1);
pp = zeros(nt,1);
tt = (2*t - a - b)/(b - a);
for j = 1 : nt
    A = spdiags([e,-tt(j)*2*e,e],-2:0,n+1,n+1);
    bb = A'\c;
    pp(j) = 0.5*(bb(1) - bb(3));
end
figure;
hold on; grid;
plot(t,ft,'Linewidth',1,'Displayname','f(x)')
plot(x,fx,'.','Markersize',30,'Displayname','Chebyshev data');
plot(t,pp,'Linewidth',1,'Displayname','Chebyshev interpolant');
set(gca,'Fontsize',16)
legend
end
```

5. Interpolation by spline functions

Reference:

- J. Stoer and R. Bulirsch, Introduction to Numerical Analysis, Third Edition, Springer, 2002

Splines are widely used, for example, in

- applications in graphics,
- numerical methods (e.g., for BVP),
- signal processing.

The simplest splines are the cubic splines. We will restrict ourselves to them. Spline functions yield smooth interpolation curves that are less likely to exhibit large oscillations

than high degree polynomials. We will prove that a cubic spline interpolant converges to a given function together with its first three derivatives as soon as the function is four times continuously differentiable and the set of interpolation points has some minimal quality (this will be specified further in this section). Recall that this is not true for polynomial interpolation in general.

5.1. **Theoretical foundations.** Let

$$\Delta := \{a = x_0 < x_1 < \ldots < x_n = b\}$$

be a partition of the interval $[a, b]$. The points x_j are called *knots*.

Definition 1. *A cubic spline S_Δ on Δ is a function $S_\Delta : [a, b] \to \mathbb{R}$ such that:*

(1) $S_\Delta \in C^2[a, b]$, i.e., is twice continuously differentiable,

(2) S_Δ coincides on every subinterval $[x_j, x_{j+1}]$, $j = 0, 1, \ldots, n-1$ with a polynomial of degree at most 3.

Therefore, a cubic spline consists of cubic polynomials glued in such a manner that their values together with the values of their first two derivatives coincide at the interior knots x_j, $j = 1, 2, \ldots, n-1$.

Suppose a function f is given at the knots x_j, $j = 0, 1, \ldots, n$, i.e.,

$$f(x_j) = f_j, \quad j = 0, 1, \ldots, n-1.$$

We denote the vector $\{f_0, \ldots, f_n\}$ by F, and a spline function that interpolates f at these points by

$$S_\Delta(F; \cdot).$$

Note that $S_\Delta(F, \cdot)$ is not uniquely defined by the vector F. Indeed, we have $4n$ coefficients of the cubic polynomials (n subintervals, and the cubic polynomial on each subinterval has 4 coefficients) and $n + 1 + 3(n-1) = 4n - 2$ conditions to satisfy.

- At each interior knot x_j, $j = 1, \ldots, n-1$, the values and the first two derivatives of the cubic polynomials on the left and on the right must coincide. This gives $3(n-1)$ conditions.
- At each knot x_j, $j = 0, 1, \ldots, n$, the value of the spline must be f_j leading to $n+1$ more conditions.

Hence we a free to assign two additional conditions that would allow us to determine $S_\Delta(F; \cdot)$ uniquely. Typically, one of the following options is used:

(1) The second derivatives are zero at the end knots: $S''_\Delta(F; a) = S''_\Delta(F; b) = 0$.
(2) Periodic: $S'_\Delta(F; a) = S'_\Delta(F; b)$, $S''_\Delta(F; a) = S''_\Delta(F; b)$. Prerequisite: $f_0 = f_n$.
(3) The first derivatives are assigned at the end knots: $S'_\Delta(F; a) = f'_0$, $S'_\Delta(F; b) = f'_n$.
(4) Not-a-knot conditions: forcing third derivative continuity across the second and penultimate knots of the spline. What does this mean? Since the third derivative of a cubic is a constant function, if that third derivative is continuous, then, in effect, there is no break at all between those two segments at those knots. You can think of it as if those knots are really not knots at all, therein the name "not-a-knot" end conditions.

5.2. **Setting up a system of equations for a cubic spline.** Throughout this section, we will fix a partition $\Delta := \{a = x_0 < x_1 < \ldots < x_n = b\}$ and a vector of values $F = \{f_0, \ldots, f_n\}$. We will use the following notations for the intervals between the knots and their lengths:

$$I_j := [x_j, x_{j+1}], \quad j = 0, 1, \ldots, n-1, \quad \text{and}$$

$$h_{j+1} := x_{j+1} - x_j.$$

The second derivatives of cubic splines at the knots are called *moments* and denoted by

$$M_j := S''_\Delta(F, x_j), \quad j = 0, 1, \ldots, n.$$

We will show that *the spline function $S_\Delta(F; x)$ is completely characterized by its moments, and the moments are found by solving a system of linear equations.*

The second derivative of a $S_\Delta(F; x)$ coincides with a linear function on each subinterval, and these linear functions can be described in terms of the moments M_j:

$$S''_\Delta(F; x) = M_j \frac{x_{j+1} - x}{h_{j+1}} + M_{j+1} \frac{x - x_j}{h_{j+1}} \quad x \in I_j \equiv [x_j, x_{j+1}]. \tag{31}$$

Integrating Eq. (31) we obtain that for each $x \in I_j$, $j = 0, 1, \ldots, n$,

$$S'_\Delta(F; x) = -M_j \frac{(x_{j+1} - x)^2}{2h_{j+1}} + M_{j+1} \frac{(x - x_j)^2}{2h_{j+1}} + A_j, \tag{32}$$

$$S_\Delta(F; x) = M_j \frac{(x_{j+1} - x)^3}{6h_{j+1}} + M_{j+1} \frac{(x - x_j)^3}{6h_{j+1}} + A_j(x - x_j) + B_j, \tag{33}$$

where A_j and B_j are constants of integration. From the equalities

$$S_\Delta(F; x_j) = f_j \quad \text{and} \quad S_\Delta(F; x_{j+1}) = f_{j+1}$$

we obtain the following equations for A_j and B_j:

$$M_j \frac{h_{j+1}^2}{6} + B_j = f_j,$$

$$M_{j+1} \frac{h_{j+1}^2}{6} + A_j h_{j+1} + B_j = f_{j+1}.$$

Hence

$$B_j = f_j - M_j \frac{h_{j+1}^2}{6}, \tag{34}$$

$$A_j = \frac{f_{j+1} - f_j}{h_{j+1}} - \frac{h_{j+1}}{6}(M_{j+1} - M_j). \tag{35}$$

This yields the following representation of the spline function in terms of its moments:

$$S_\Delta(F; x) = \alpha_j + \beta_j(x - x_j) + \gamma_j(x - x_j)^2 + \delta_j(x - x_j)^3 \quad \text{for} \quad x \in I_j, \tag{36}$$

where

$$\alpha_j = f_j, \tag{37}$$

$$\gamma_j = \frac{M_j}{2}, \tag{38}$$

$$\beta_j = S'_\Delta(F;x_j) = -M_j\frac{h_{j+1}}{2} + A_j = \frac{f_{j+1}-f_j}{h_{j+1}} - \frac{h_{j+1}}{6}(M_{j+1}+2M_j), \tag{39}$$

$$\delta_j = \frac{S'''_\Delta(F,x_j^+)}{6} = \frac{M_{j+1}-M_j}{6h_{j+1}}. \tag{40}$$

Now we need to calculate the moments M_j. The continuity of $S'_\Delta(F;x)$ at the interior knots yields $n-1$ equations for the moments M_j. Using Eqs. (33) and (35) we obtain

$$S'_\Delta(F;x) = -M_j\frac{(x_{j+1}-x)^2}{2h_{j+1}} + M_{j+1}\frac{(x-x_j)^2}{2h_{j+1}} + \frac{f_{j+1}-f_j}{h_{j+1}} - \frac{h_{j+1}}{6}(M_{j+1}-M_j).$$

Therefore, for $j = 1, 2, \ldots, n-1$ we have

$$S'_\Delta(F;x_j^-) = \frac{f_j - f_{j-1}}{h_j} + \frac{h_j}{3}M_j + \frac{h_j}{6}M_{j-1},$$

$$S'_\Delta(F;x_j^+) = \frac{f_{j+1}-f_j}{h_{j+1}} - \frac{h_{j+1}}{3}M_j - \frac{h_{j+1}}{6}M_{j+1}.$$

Since $S'_\Delta(F;x_j^+) = S'_\Delta(F;x_j^-)$, we have

$$\frac{h_j}{6}M_{j-1} + \frac{h_j+h_{j+1}}{3}M_j + \frac{h_{j+1}}{6}M_{j+1} = \frac{f_{j+1}-f_j}{h_{j+1}} - \frac{f_j-f_{j-1}}{h_j} \tag{41}$$

for $j = 1, 2, \ldots, n-1$. These are $n-1$ equations for $n+1$ unknown moments. The remaining two equations can be gained from the boundary conditions. In the first case, $S''_\Delta(F;a) = S''_\Delta(F;b) = 0$ we set $M_0 = M_n = 0$. In the periodic case we set $M_0 = M_n$ and

$$\frac{h_n}{6}M_{n-1} + \frac{h_n+h_1}{3}M_n + \frac{h_1}{6}M_1 = \frac{f_1-f_n}{h_1} - \frac{f_n-f_{n-1}}{h_n}.$$

Exercise Obtain additional two conditions for the case of assigned first derivatives at the end points:

$$\frac{h_1}{3}M_0 + \frac{h_1}{6}M_1 = \frac{f_1-f_0}{h_1} - f'_0, \tag{42}$$

$$\frac{h_n}{6}M_{n-1} + \frac{h_n}{3}M_n = f'_n - \frac{f_n-f_{n-1}}{h_n}. \tag{43}$$

Eqs. (41), (42) and (43) can be written in the common format

$$\mu_j M_{j-1} + 2M_j + \lambda_j M_{j+1} = d_j, \quad j = 1, 2, \ldots, n-1,$$

where

$$\lambda_j := \frac{h_{j+1}}{h_j + h_{j+1}}, \tag{44}$$

$$\mu_j := 1 - \lambda_j = \frac{h_j}{h_j + h_{j+1}}, \tag{45}$$

$$d_j := \frac{6}{h_j + h_{j+1}} \left\{ \frac{f_{j+1} - f_j}{h_{j+1}} - \frac{f_j - f_{j-1}}{h_j} \right\}. \tag{46}$$

In the case $M_0 = M_n = 0$ we set

$$\lambda_0 = 0, \quad d_0 = 0, \quad \mu_n = 0, \quad d_n = 0.$$

As a result, weobtain the following system of equations

$$\begin{bmatrix} 2 & \lambda_0 & & & & 0 \\ \mu_1 & 2 & \lambda_1 & & & \\ & \mu_2 & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & & & \cdot & 2 & \lambda_{n-1} \\ 0 & & & & \mu_n & 2 \end{bmatrix} \begin{bmatrix} M_0 \\ M_1 \\ \cdot \\ \cdot \\ \cdot \\ M_n \end{bmatrix} = \begin{bmatrix} d_0 \\ d_1 \\ \cdot \\ \cdot \\ \cdot \\ d_n \end{bmatrix}. \tag{47}$$

Exercise Write out the system of equations for the momenta for the periodic case and for the case of assigned first derivatives at the endpoints.

Theorem 6. *The matrix in Eq.* (47) *is nonsingular for any partition* Δ *of* $[a, b]$.

Proof. We will denote the Matrix in Eq. (47) by A. If follows from the definitions of λ_j and μ_j that $\mu_j + \lambda_j = 1$, $\mu_j \geq 0$, $\lambda_j \geq 0$. Therefore, the matrix A is strictly diagonal dominant. Therefore, for any vector z

$$\max_j |z_j| \leq \max_j |Az_j|. \tag{48}$$

(Check this!) Hence $Az = 0$ if and only if $z = 0$. □

5.3. **A fast solver for tridiagonal matrices.** The matrix in Eq. (47) is tridiagonal. There is a fast solver for such kind of systems that involves $O(n)$ flops.

$$
\begin{aligned}
&q_0 = -\frac{\lambda_0}{2}; \quad u_0 = \frac{d_0}{2}; \quad \lambda_n = 0;\\
&\text{for} \quad k = 1 : n\\
&\qquad p_k = \mu_k q_{k-1} + 2;\\
&\qquad q_k = -\frac{\lambda_k}{p_k};\\
&\qquad u_k = \frac{d_k - \mu_k u_{k-1}}{p_k};\\
&\text{end for}\\
&M_n = u_n;\\
&\text{for} \quad k = n-1 : 0\\
&\qquad M_k = q_k M_{k+1} + u_k;\\
&\text{end for}
\end{aligned}
$$

In the first for-cycle, we successively express $M_j = M_{j+1}q_j + u_j$. Then we find $M_n = u_n$ as $\lambda_n = 0$. In the second for-cycle we successively find M_{n-1}, M_{n-2}, ..., M_0 using the relationships $M_j = M_{j+1}q_j + u_j$ created by the first for-cycle.

5.4. **Convergence properties of cubic spline functions.**

Definition 2. *The fineness of the given partition is*

$$\|\Delta\| := \max_j h_j.$$

Recall that interpolating polynomials do not necessarily converge to f even for smooth f as the fineness of the partition tends to zero (the Runge phenomenon!). In contrast, the spline interpolants do converge to f together with their first three derivatives under mild conditions.

First we will introduce some notations. We will denote by F'' the vector of the second derivatives of f at the knots. The vector of moments M satisfies Eq. (47). We will write

$$AM = d.$$

The residual r is defined as

$$r := d - AF'' = A(M - F'').$$

Theorem 7. *Suppose the first derivatives of f at the end knots are assigned. If $f \in C^4[a,b]$ and $|f^{(4)}(x)| \le L$ for $x \in [a,b]$, then*

$$\|M - F''\| \le \|r\| \le \frac{3}{4} L \|\Delta\|^2.$$

Proof. By definition of the residual we have

$$\begin{aligned} r_j &= d_j - \mu_j f''(x_{j-1}) - 2f''(x_j) - \lambda_j f''(x_{j+1}) \\ &= \frac{6}{h_j + h_{j+1}} \left\{ \frac{f_{j+1} - f_j}{h_{j+1}} - \frac{f_j - f_{j-1}}{h_j} \right\} \\ &- \frac{h_j}{h_j + h_{j+1}} f''(x_{j-1}) - 2f''(x_j) - \frac{h_{j+1}}{h_j + h_{j+1}} f''(x_{j+1}). \end{aligned}$$

Using Taylor's expansion around x_j we obtain

$$\begin{aligned} r_j &= \frac{6}{h_j + h_{j+1}} (f' + \frac{h_{j+1}}{2} f'' + \frac{h_{j+1}^2}{6} f''' + \frac{h_{j+1}^3}{24} f''''(\tau_1) \\ &- f' + \frac{h_j}{2} f'' - \frac{h_j^2}{6} f''' + \frac{h_j^3}{24} f''''(\tau_2)) \\ &- \frac{h_j}{h_j + h_{j+1}} \left[f'' - h_j f''' + \frac{h_j^2}{2} f''''(\tau_3) \right] - 2f'' - \\ &- \frac{h_{j+1}}{h_j + h_{j+1}} \left[f'' + h_{j+1} f''' + \frac{h_{j+1}^2}{2} f''''(\tau_4) \right] \\ &= \frac{1}{h_j + h_{j+1}} \left[\frac{h_{j+1}^3}{4} f''''(\tau_1) + \frac{h_j^3}{4} f''''(\tau_2) - \frac{h_j^3}{2} f''''(\tau_3) - \frac{h_{j+1}}{2} f''''(\tau_4) \right]. \end{aligned}$$

Here all omitted arguments are x_j and $\tau_j \in [x_{j-1}, x_{j+1}]$. Therefore, for $j = 1, 2, \ldots, n-1$

$$|r_j| \le \frac{3}{4} L \frac{h_{j+1}^3 + h_j^3}{h_{j+1} + h_j} = \frac{3}{4} L (h_j^2 - h_j h_{j+1} + h_{j+1}^2) \le \frac{3}{4} L \|\Delta\|^2.$$

The first and the last intervals are handled in a similar manner. For them we have

$$|r_0| \le \frac{3}{4} L \|\Delta\|^2 \quad \text{and} \quad |r_n| \le \frac{3}{4} L \|\Delta\|^2.$$

Since $r = A(M - F'')$ and A has the property given by Eq. (48) we get

$$\|M - F''\| \le \|r\| \le \frac{3}{4} L \|\Delta\|^2.$$

□

Theorem 8. *Suppose $f \in C^4[a, b]$ and $|f^{(4)}(x)| \le L$ for $x \in [a, b]$. Let Δ be a partition of the interval $[a, b]$ and K be a constant such that*

$$\frac{\|\Delta\|}{h_{j+1}} \le K, \quad j = 0, 1, \ldots, n-1.$$

If S_Δ is the spline function which interpolates the values of f at the knots of the partition Δ and satisfies

$$S'_\Delta(a) = f'(a) \quad \text{and} \quad S'_\Delta(b) = f'(b),$$

then there exist constants $c_k \le 2$ independent of the partition Δ, such that for $x \in [a,b]$

$$|f(x) - S_\Delta(x)| \le c_0 L\|\Delta\|^4, \tag{49}$$

$$|f'(x) - S'_\Delta(x)| \le c_1 L\|\Delta\|^3, \tag{50}$$

$$|f''(x) - S''_\Delta(x)| \le c_2 L\|\Delta\|^2, \tag{51}$$

$$|f'''(x) - S'''_\Delta(x)| \le c_3 LK\|\Delta\|. \tag{52}$$

Proof. First we prove Eq. (52). For $x \in [x_{j-1}, x_j]$,

$$\begin{aligned} S'''_\Delta(x) - f'''(x) &= \frac{M_j - M_{j-1}}{h_j} - f'''(x) \\ &= \frac{M_j - f''(x_j)}{h_j} - \frac{M_{j-1} - f''(x_{j-1})}{h_j} \\ &+ \frac{f''(x_j) - f''(x) - [f''(x_{j-1}) - f''(x)]}{h_j} - f'''(x). \end{aligned}$$

Using the previous theorem and the Taylor expansion at x we get

$$\begin{aligned} |S'''_\Delta(x) - f'''(x)| &\le \frac{3}{2}L\frac{\|\Delta\|^2}{h_j} + \frac{1}{h_j}|(x_j - x)f''' + \frac{(x_j - x)^2}{2}f''''(\eta_1) \\ &- (x_{j-1} - x)f''' - \frac{(x_{j-1} - x)}{2}f''''(\eta_2) - h_j f'''| \\ &\le \frac{3}{2}L\frac{\|\Delta\|^2}{h_j} + \frac{L}{2}\frac{\|\Delta\|^2}{h_j} = 2L\frac{\|\Delta\|^2}{h_j} \\ &\le 2LK\|\Delta\|. \end{aligned}$$

Here $\eta_1, \eta_2 \in [x_{j-1}, x_j]$.

To prove Eq. (51) we observe that for every $x \in (a,b)$ there is a closes knot $x_j = x_j(x)$. Assume without loss of generality that $x \le x_j(x)$ so that $x \in [x_{j-1}, x_j]$ and $|x_j(x) - x| \le h_j/2 \le \|\Delta\|/2$. Then

$$\begin{aligned} |f''(x) - S''_\Delta(x)| &= |f''(x_j) - S''_\Delta(x_j) + \int_{x_j(x)}^{x} (f'''(t) - S'''_\Delta(t))dt| \\ &\le \frac{3}{4}L\|\Delta\|^2 + \frac{h_j}{2}2L\frac{\|\Delta\|^2}{h_j} \le \frac{7}{4}L\|\Delta\|^2, \quad x \in (a,b). \end{aligned}$$

Next we prove Eq. (50). In addition to the boundary points $\xi_0 := a$ and $\xi_{n+1} := b$ there exist by Rolle's theorem n points $\xi_j \in (x_{j-1}, x_j)$ such that

$$f'(\xi_j) = S'_\Delta(\xi_j).$$

for any $x \in (a,b)$ there exists a closest one $\xi_j = \xi_j(x)$, and

$$|x - \xi_j(x)| < \|\Delta\|.$$

Therefore,

$$|f'(x) - S'_\Delta(x)| = |\int_{\xi(x)}^{x} (f''(t) - S''_\Delta(t))dt|$$
$$\leq \|\Delta\| \frac{7}{4} L \|\Delta\|^2 = \frac{7}{4} L \|\Delta\|^3.$$

Finally we prove Eq. (49). We have

$$|f(x) - S_\Delta(x)\| = |\int_{x_j(x)}^{x} (f'(t) - S'_\Delta(t))dt|$$
$$\leq \frac{\|\Delta\|}{2} \frac{7}{4} L \|\Delta\|^3 = \frac{7}{8} L \|\Delta\|^4, \quad x \in [a, b].$$

□

5.5. **Spline interpolation in Matlab.** Matlab has several tools for computing cubic splines. Functions `spline` and `interp1` are regular matlab functions. Function `spline` is designed for computing cubic splines. The boundary conditions can be chosen to be either 'Not-a-Knot' or 'Complete' (i.e., the values of the first derivatives are prescribed at the endpoints). Function `interp1` offers different kinds of interpolation. If its argument 'Method' is set to 'spline', it produces a cubic spline with the 'Not-a-Knot' boundary conditions.

The Matlab code below demonstrates the use of piecewise-linear interpolant and two cubic splines, with 'Not-a-Knot' and 'Complete' boundary conditions on the function $f(x) = (1+x^2)^{-1}$. Newton's interpolant is also plotted for comparison. Fig. 4 is generated by this code.

```
function CompareSplines()
n = 6;
a = -5; b = 5;
x = linspace(a,b,n + 1)'; % n+1 equispaced points
t = linspace(a,b,200);
f = @(x)1./(x.^2 + 1); % Witch of Agnesi
fx = f(x);
ft = f(t);
figure;
hold on; grid;
plot(x,fx,'.','Markersize',30,'Displayname','Data F');
plot(t,ft,'Linewidth',1,'Displayname','f(x)')
%% Newton's interpolation.
% compute divided differences
dd = zeros(n + 1);
dd(:,1) = fx;
for k = 2 : n + 1
    dd(1:n-k+2,k) = (dd(2:n-k+3,k-1)-dd(1:n-k+2,k-1))./(x(k:n+1)-x(1:n-k+2));
```

```
end
% evaluate Newton's interpolant
pt = dd(1,n+1);
for k = n : -1 : 1
    pt = pt.*(t - x(k)) + dd(1,k);
end
plot(t,pt,'Linewidth',1,'DisplayName','Newton interpolant');
%% Piecewise-linear interpolant
pl = interp1(x,fx,t);
plot(t,pl,'Linewidth',1,'Displayname','Piecewise-linear');
%% Cubic spline, not-a-knot condition
pc = interp1(x,fx,t,'spline');
plot(t,pc,'Linewidth',1,'Displayname','Cubic spline, not-a-knot');
%% Cubic spline, first derivaties at the end knots
fp = @(x) -2*x./((x.^2 + 1).^2);
pp = spline(x,[fp(a);fx;fp(b)]);
pc1 = ppval(pp,t);
plot(t,pc1,'Linewidth',1,'Displayname','Cubic spline, 1st ders');
set(gca,'Fontsize',16)
legend
end
```

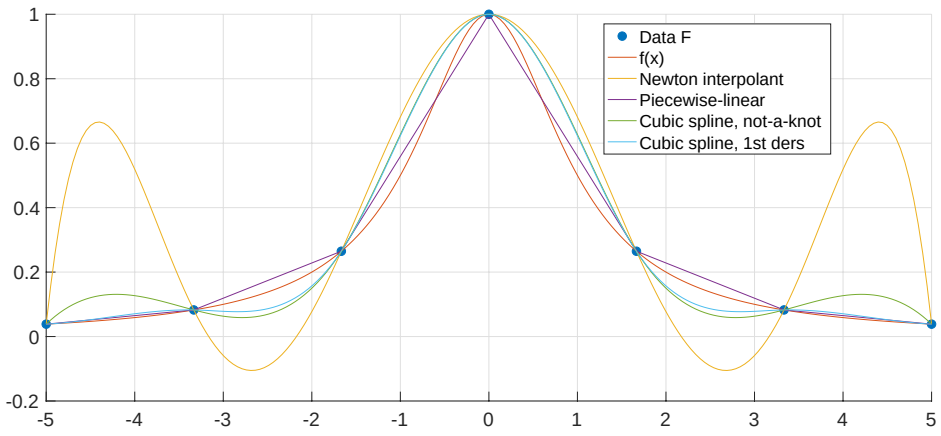

FIGURE 4. Comparison of Newton's intepolant with 7 equispaced knots, piecewise linear interpolant, and two cubic splines for $f(x) = (1+x^2)^{-1}$ on $[-5, 5]$