LECTURE 7: *Trigonometric Interpolation*

1.9 Trigonometric interpolation for periodic functions

Thus far all our interpolation schemes have been based on polynomials. However, if the function f is *periodic*, one might naturally prefer to interpolate f with some set of periodic functions.

To be concrete, suppose we have a continuous 2π-periodic function f that we wish to interpolate at the uniformly spaced points $x_k = 2\pi k/n$ for $k = 0, \ldots, n$ with $n = 5$. We shall build an interpolant as a linear combination of the 2π-periodic functions

'2π-periodic' means that f is continuous throughout $\mathbb{R}$ and $f(x) = f(x + 2\pi)$ for all $x \in \mathbb{R}$.

The choice of period 2π makes the notation a bit simpler, but the idea can be easily adapted for any period.

$$b_0(x) = 1, \quad b_1(x) = \sin(x), \quad b_2(x) = \cos(x), \quad b_3(x) = \sin(2x), \quad b_4(x) = \cos(2x).$$

Note that we have *six* interpolation conditions at x_k for $k = 0, \ldots, 5$, but only *five* basis functions. This is not a problem: since f is periodic, $f(x_0) = f(x_5)$, and the same will be true of our 2π-periodic interpolant: the last interpolation condition is automatically satisfied.

We shall construct an interpolant of the form

$$t_5(x) = \sum_{k=0}^{4} c_k b_k(x)$$

such that

$$t_5(x_j) = f(x_j), \quad j = 0, \ldots, 4.$$

To compute the unknown coefficients $c_0, \ldots, c_4$, set up a linear system as usual,

$$\begin{bmatrix} b_0(x_0) & b_1(x_0) & b_2(x_0) & b_3(x_0) & b_4(x_0) \\ b_0(x_1) & b_1(x_1) & b_2(x_1) & b_3(x_1) & b_4(x_1) \\ b_0(x_2) & b_1(x_2) & b_2(x_2) & b_3(x_2) & b_4(x_2) \\ b_0(x_3) & b_1(x_3) & b_2(x_3) & b_3(x_3) & b_4(x_3) \\ b_0(x_4) & b_1(x_4) & b_2(x_4) & b_3(x_4) & b_4(x_4) \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix} = \begin{bmatrix} f(x_0) \\ f(x_1) \\ f(x_2) \\ f(x_3) \\ f(x_4) \end{bmatrix},$$

which can be readily generalized to accommodate more interpolation points. We could solve this system for $c_0, \ldots, c_n$, but we prefer to express the problem in a more convenient basis for the trigonometric functions. Recall Euler's formula,

You would $b_6(x) = \sin(3x)$, $b_7(x) = \cos(3x)$, etc.: one function for each additional interpolation point. Generally you would use an odd value of n, to include pairs of sines and cosines.

$$e^{i\theta x} = \cos(\theta x) + i \sin(\theta x),$$

which also implies that

$$e^{-i\theta x} = \cos(\theta x) - i \sin(\theta x).$$

To prove this, write the Taylor expansion of $e^{i\theta x}$, then separate the real and imaginary components to give Taylor expansions for $\cos(\theta x)$ and $\sin(\theta x)$.

From these formulas it follows that

$$\operatorname{span}\{e^{i\theta x}, e^{-i\theta x}\} = \{\cos(\theta x), \sin(\theta x)\}.$$

Note that we can also write $b_0(x) \equiv 1 = e^{i0x}$. Putting these pieces together, we arrive at an alternative basis for the trigonometric interpolation space:

$$\text{span}\{1, \sin(x), \cos(x), \sin(2x), \cos(2x)\} = \text{span}\{e^{-2ix}, e^{-ix}, e^{0ix}, e^{ix}, e^{2ix}\}.$$

The interpolant t_n can thus be expressed in the form

$$t_4(x) = \sum_{k=-2}^{2} \gamma_k e^{ikx} = \sum_{k=-2}^{2} \gamma_k (e^{ix})^k.$$

This last sum is written in a manner that emphasizes that t_4 is *a polynomial in the variable* e^{ix}, and hence t_n is a *trigonometric polynomial*. In this basis, the interpolation conditions give the linear system

$$\begin{bmatrix} e^{-2ix_0} & e^{-ix_0} & e^{0ix_0} & e^{ix_0} & e^{i2x_0} \\ e^{-2ix_1} & e^{-ix_1} & e^{0ix_1} & e^{ix_1} & e^{i2x_1} \\ e^{-2ix_2} & e^{-ix_2} & e^{0ix_2} & e^{ix_2} & e^{i2x_2} \\ e^{-2ix_3} & e^{-ix_3} & e^{0ix_3} & e^{ix_3} & e^{i2x_3} \\ e^{-2ix_4} & e^{-ix_4} & e^{0ix_4} & e^{ix_4} & e^{i2x_4} \end{bmatrix} \begin{bmatrix} \gamma_{-2} \\ \gamma_{-1} \\ \gamma_0 \\ \gamma_1 \\ \gamma_2 \end{bmatrix} = \begin{bmatrix} f(x_0) \\ f(x_1) \\ f(x_2) \\ f(x_3) \\ f(x_4) \end{bmatrix},$$

again with the natural generalization to larger odd integers n. At first blush this matrix looks no simpler than the one we first encountered, but a fascinating structure lurks. Notice that a generic entry of this matrix has the form $e^{\ell i x_k}$ for $\ell = -(n-1)/2, \ldots, (n-1)/2$ and $k = 0, \ldots, n-1$. Since $x_k = 2\pi k/n$, rewrite this entry as

$$e^{\ell i x_k} = \left(e^{ix_k}\right)^\ell = \left(e^{2\pi i k/n}\right)^\ell = \left(e^{2\pi i/n}\right)^{k\ell} = \omega^{k\ell},$$

where $\omega = e^{2\pi i/n}$ is an *nth root of unity*. In the $n = 5$ case, the linear system can thus be written as

This name comes from the fact that $\omega^n = 1$.

$$\begin{bmatrix} \omega^0 & \omega^0 & \omega^0 & \omega^0 & \omega^0 \\ \omega^{-2} & \omega^{-1} & \omega^0 & \omega^1 & \omega^2 \\ \omega^{-4} & \omega^{-2} & \omega^0 & \omega^2 & \omega^4 \\ \omega^{-6} & \omega^{-3} & \omega^0 & \omega^3 & \omega^6 \\ \omega^{-8} & \omega^{-4} & \omega^0 & \omega^4 & \omega^8 \end{bmatrix} \begin{bmatrix} \gamma_{-2} \\ \gamma_{-1} \\ \gamma_0 \\ \gamma_1 \\ \gamma_2 \end{bmatrix} = \begin{bmatrix} f(x_0) \\ f(x_1) \\ f(x_2) \\ f(x_3) \\ f(x_4) \end{bmatrix}. \tag{1.30}$$

Denote this system by $\mathbf{F}\boldsymbol{\gamma} = \mathbf{f}$. Notice that each column of $\mathbf{F}$ equals some (entrywise) power of the vector

$$\begin{bmatrix} \omega^0 \\ \omega^1 \\ \omega^2 \\ \omega^3 \\ \omega^4 \end{bmatrix}.$$

In other words, *the matrix* $\mathbf{F}$ *has Vandermonde structure*. From our past experience with polynomial fitting addressed in Section 1.2.1, we

might fear that this formulation is ill-suited to numerical computations, i.e., solutions γ to the system $\mathbf{F}\gamma = \mathbf{f}$ could be polluted by large numerical errors.

In the language of numerical linear algebra, we might fear that the matrix $\mathbf{F}$ is *ill-conditioned*, i.e., the *condition number* $\|\mathbf{F}\|\,\|\mathbf{F}^{-1}\|$ is large.

Before jumping to this conclusion, examine $\mathbf{F}^*\mathbf{F}$. To form $\mathbf{F}^*$ note that $\overline{\omega^{-\ell}} = \omega^{\ell}$, so

$\mathbf{F}^*$ is the conjugate-transpose of $\mathbf{F}$:

$$\mathbf{F}^* = \overline{\mathbf{F}}^{\mathrm{T}},$$

so $(\mathbf{F}^*)_{j,k} = \overline{\mathbf{F}_{k,j}}$.

$$\mathbf{F}^*\mathbf{F} = \begin{bmatrix} \omega^0 & \omega^2 & \omega^4 & \omega^6 & \omega^8 \\ \omega^0 & \omega^1 & \omega^2 & \omega^3 & \omega^4 \\ \omega^0 & \omega^0 & \omega^0 & \omega^0 & \omega^0 \\ \omega^0 & \omega^{-1} & \omega^{-2} & \omega^{-3} & \omega^{-4} \\ \omega^0 & \omega^{-2} & \omega^{-4} & \omega^{-6} & \omega^{-8} \end{bmatrix} \begin{bmatrix} \omega^0 & \omega^0 & \omega^0 & \omega^0 & \omega^0 \\ \omega^{-2} & \omega^{-1} & \omega^0 & \omega^1 & \omega^2 \\ \omega^{-4} & \omega^{-2} & \omega^0 & \omega^2 & \omega^4 \\ \omega^{-6} & \omega^{-3} & \omega^0 & \omega^3 & \omega^6 \\ \omega^{-8} & \omega^{-4} & \omega^0 & \omega^4 & \omega^8 \end{bmatrix}.$$

The (ℓ, k) entry for $\mathbf{F}^*\mathbf{F}$ thus takes the form

$$(\mathbf{F}^*\mathbf{F})_{\ell,k} = \omega^0 + \omega^{(k-\ell)} + \omega^{2(k-\ell)} + \omega^{3(k-\ell)} + \omega^{4(k-\ell)}.$$

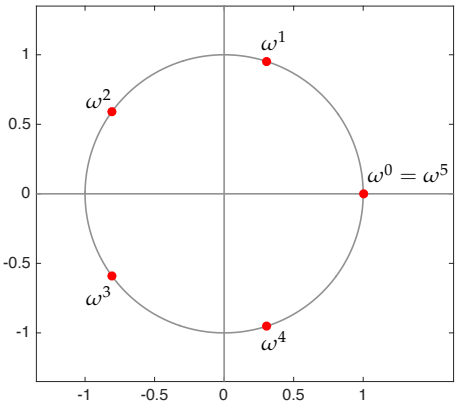
On the diagonal, when $\ell = k$, we simply have

$$(\mathbf{F}^*\mathbf{F})_{k,k} = \omega^0 + \omega^0 + \omega^0 + \omega^0 + \omega^0 = n.$$

On the off-diagonal, use $\omega^n = 1$ to see that all the off diagonal entries simplify to

$$(\mathbf{F}^*\mathbf{F})_{\ell,k} = \omega^0 + \omega^1 + \omega^2 + \omega^3 + \omega^4, \qquad \ell \neq k.$$

You can think of this last entry as n times the average of ω^0, ω^1, ω^2, ω^3, and ω^4, which are uniformly spaced points on the unit circle, shown in the plot to the right.



As these points are uniformly positioned about the unit circle, their mean must be zero, and hence

$$(\mathbf{F}^*\mathbf{F})_{\ell,k} = 0, \qquad \ell \neq k.$$

We thus must conclude that

$$\mathbf{F}^*\mathbf{F} = n\mathbf{I},$$

thus giving a formula for the inverse:

$$\mathbf{F}^{-1} = \frac{1}{n}\mathbf{F}^*.$$

The system $\mathbf{F}\gamma = \mathbf{f}$ can be immediately solved without the need for any factorization of $\mathbf{F}$:

$$\gamma = \frac{1}{n}\mathbf{F}^*\mathbf{f}.$$

The ready formula for $\mathbf{F}^{-1}$ is reminiscent of a *unitary* matrix. In fact, the matrices

$$\frac{1}{\sqrt{n}}\mathbf{F} \quad \text{and} \quad \frac{1}{\sqrt{n}}\mathbf{F}^*$$

$\mathbf{Q} \in \mathbb{C}^{n\times n}$ is unitary if and only if $\mathbf{Q}^{-1} = \mathbf{Q}^*$, or, equivalently, $\mathbf{Q}^*\mathbf{Q} = \mathbf{I}$.

are indeed unitary, and hence $\|n^{-1/2}\mathbf{F}\|_2 = \|n^{-1/2}\mathbf{F}^*\|_2 = 1$.

From this we can compute the condition number of $\mathbf{F}$:

$$\|\mathbf{F}\|_2\|\mathbf{F}^{-1}\|_2 = \frac{1}{n}\|\mathbf{F}\|_2\|\mathbf{F}^*\|_2 = \|n^{-1/2}\mathbf{F}\|_2\|n^{-1/2}\mathbf{F}^*\|_2 = 1.$$

This special Vandermonde matrix is perfectly conditioned! One can easily solve the system $\mathbf{F}\boldsymbol{\gamma} = \mathbf{f}$ to high precision. The key distinction between this case and standard polynomial interpolation is that now we have a Vandermonde matrix based on *points* e^{ix_k} *that are equally spaced about the unit circle in the complex plane*, whereas before our points were distributed over an interval on the real line. This distinction makes all the difference between an unstable matrix equation and one that is not only perfectly stable, but also forms the cornerstone of modern signal processing.

In fact, we have just computed the 'Discrete Fourier Transform' (DFT) of the data vector

$$\begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_{n-1}) \end{bmatrix}.$$

The coefficients $\gamma_{-(n-1)/2}, \ldots, \gamma_{(n-1)/2}$ that make up the vector

$$\boldsymbol{\gamma} = \frac{1}{n}\mathbf{F}^*\mathbf{f}$$

are the *discrete Fourier coefficients* of the data in $\mathbf{f}$. From where does this name derive?

1.9.1 Connection to Fourier series

In a real/functional analysis course, one learns that a 2π-periodic function f can be written as the Fourier series

$$f(x) = \sum_{k=-\infty}^{\infty} c_k \mathrm{e}^{\mathrm{i}kx},$$

where the *Fourier coefficients* c_ℓ are defined via

$$c_k := \frac{1}{2\pi}\int_0^{2\pi} f(x)\mathrm{e}^{-\mathrm{i}kx}.$$

Notice that $\gamma_k = ((1/n)\mathbf{F}^*\mathbf{f})_k$ is an approximation to this c_k:

$$\begin{aligned}\gamma_k &= \frac{1}{n}\sum_{\ell=0}^{n-1} f(x_\ell)\omega^{-\ell k} \\ &= \frac{1}{n}\sum_{k=0}^{n-1} f(x_k)\mathrm{e}^{-(2\pi k/n)\mathrm{i}\ell} = \frac{1}{n}\sum_{k=0}^{n-1} f(x_k)\mathrm{e}^{-\mathrm{i}\ell x_k}.\end{aligned}$$

The matrix 2-norm is defined as

$$\|\mathbf{F}\|_2 = \max_{\mathbf{x}\neq\mathbf{0}} \frac{\|\mathbf{F}\mathbf{x}\|_2}{\|\mathbf{x}\|_2},$$

where the vector norm on the right hand side is the Euclidean norm

$$\|\mathbf{y}\|_2 = \Big(\sum_k |y_k|^2\Big)^{1/2} = (\mathbf{y}^*\mathbf{y})^{1/2}.$$

The 2-norm of a unitary matrix is one: If $\mathbf{Q}^*\mathbf{Q} = \mathbf{I}$, then

$$\|\mathbf{Q}\mathbf{x}\|_2^2 = \mathbf{x}^*\mathbf{Q}^*\mathbf{Q}\mathbf{x} = \mathbf{x}^*\mathbf{x} = \|\mathbf{x}\|^2,$$

so $\|\mathbf{Q}\|_2 = 1$.

To ensure pointwise convergence of this series for all $x \in [0, 2\pi]$, f must be a continuous 2π-periodic function with a continuous first derivative. The functions $e_k(x) = \mathrm{e}^{\mathrm{i}kx}/\sqrt{2\pi}$ form an orthonormal basis for the space $L^2[0, 2\pi]$ endowed with the inner product

$$(f, g) = \int_0^{2\pi} f(x)\overline{g(x)}\,\mathrm{d}x.$$

The Fourier series is simply an expansion of f in this basis: $f = \sum_k (f, e_k)\, e_k$.

Now use the fact that $f(x_0)e^{-i\ell x_0} = f(x_n)e^{-i\ell x_n}$ to view the last sum as a *composite trapezoid rule* approximation of an integral:

The composite trapezoid rule will be discussed in Chapter 3.

$$\begin{aligned} 2\pi\gamma_\ell &= \frac{2\pi}{n}\left(\frac{1}{2}f(x_0)e^{-i\ell x_0} + \sum_{k=1}^{n-1} f(x_k)e^{-i\ell x_k} + \frac{1}{2}f(x_n)e^{-i\ell x_n}\right) \\ &\approx \int_0^{2\pi} f(x)e^{-i\ell x}\,dx \\ &= 2\pi c_\ell. \end{aligned}$$

The coefficient γ_ℓ that premultiplies $e^{i\ell x}$ in the trigonometric interpolating polynomial is actually an approximation of the Fourier coefficient c_ℓ.

Let us go one step further. Notice that the trigonometric interpolant

$$t_n(x) = \sum_{k=-(n-1)/2}^{(n-1)/2} \gamma_k e^{ikx}$$

is an approximation to the Fourier series

$$f(x) = \sum_{k=-\infty}^{\infty} c_k\, e^{ikx}$$

obtained by (1) truncating the series, and (2) replacing c_k with γ_k. To assess the quality of the approximation, we need to understand the magnitude of the terms dropped from the sum, as well as the accuracy of the composite trapezoid rule approximation γ_k to c_k. We will thus postpone discussion of $f(x) - t_n(x)$ until we develop a few more analytical tools in the next two chapters.

1.9.2 *Computing the discrete Fourier coefficients*

Normally we would require $O(n^2)$ operations to compute these coefficients using matrix-vector multiplication with $\mathbf{F}^*$, but Cooley and Tukey discovered in 1965 that given the amazing structure in $\mathbf{F}^*$, one can arrange operations so as to compute $\boldsymbol{\gamma} = n^{-1}\mathbf{F}^*\mathbf{f}$ in only $O(n \log n)$ operations: a procedure that we now famously call the *Fast Fourier Transform* (FFT).

Apparently the FFT was discovered earlier by Gauss, but it was forgotten, given its limited utility before the advent of automatic computation. Jack Good (Bletchley Park codebreaker and, later, a Virginia Tech statistician) published a similar idea in 1958. Good recalls: 'John Tukey (December 1956) and Richard L. Garwin (September 1957) visited Cheltenham and I had them round to steaks and fries on separate occasions. I told Tukey briefly about my FFT (with little detail) and, in Cooley and Tukey's well known paper of 1965, my 1958 paper is the only citation.' See D. L. Banks, 'A conversation with I. J. Good,' *Stat. Sci.* 11 (1996) 1–19.

We can summarize this section as follows.

> The FFT of a vector of uniform samples of a 2π-periodic function f gives the coefficients for the trigonometric interpolant to f at those sample points. These coefficients approximate the function's Fourier coefficients.

Example 1.8 (Trig interpolation of a smooth periodic function)**.** Figure 1.14 shows the degree $n = 5, 7, 9$ and 11 trigonometric interpolants to the 2π-periodic function $f(x) = e^{\cos(x)+\sin(2x)}$. Notice that although all the interpolation points are all drawn from the interval $[0, 2\pi)$ (indicated by the gray region on the plot), the interpolants are just as accurate outside this region. In contrast, a standard polynomial fit through the same points will behave very differently: (non-constant) polynomials must satisfy $|p_n(x)| \to \infty$ as $|x| \to \infty$. Figure 1.15 shows this behavior for $n = 7$: for $x \in [0, 2\pi]$, the polynomial fit to f is about as accurate as the $n = 7$ trigonometric polynomial in Figure 1.14. Outside of $[0, 2\pi]$,the polynomial is much worse.

Example 1.9 (Trig interpolation of non-smooth function)**.** Figure 1.15 shows that a standard (non-perioidic) polynomial fit to a periodic function can yield a good approximation, at least over the interval from which the interpolation points are drawn. Now turn the tables: how well does a (periodic) trigonometric polynomial fit a smooth but non-periodic function? Simply take $f(x) = x$ on $[0, 2\pi]$, and construct the trigonometric interpolant as described above for $n = 11$. The top plot in Figure 1.16 shows that t_{11} gives a very poor approximation to f, constrained by design to be periodic even though f is not. The fact that $f(2\pi) \neq f(0)$ acts like a discontinuity, vastly impairing the quality of the approximation. The bottom plot in Figure 1.16 repeats this exercise for $f(x) = (x - \pi)^2$. Since in this case $f(0) = f(2\pi)$ we might expect better results; indeed, the approximation looks quite reasonable. Note, however, that $f'(0) \neq f'(\pi)$, and this lack of smoothness severely slows convergence of t_n to f as $n \to \infty$. Figure 1.17 contrasts this slow rate of convergence with the much faster convergence observed for $f(x) = e^{\cos(x)+\sin(2x)}$ used in Figure 1.14. Clearly the periodic interpolant is much better suited to smooth f.

1.9.3 Fast MATLAB implementation

MATLAB organizes its Fast Fourier Transform is a slightly different fashion than we have described above. To fit with MATLAB, reorder the unknowns in the system (1.30) to obtain

$$\begin{bmatrix} \omega^0 & \omega^0 & \omega^0 & \omega^0 & \omega^0 \\ \omega^0 & \omega^1 & \omega^2 & \omega^{-2} & \omega^{-1} \\ \omega^0 & \omega^2 & \omega^4 & \omega^{-4} & \omega^{-2} \\ \omega^0 & \omega^3 & \omega^6 & \omega^{-6} & \omega^{-3} \\ \omega^0 & \omega^4 & \omega^8 & \omega^{-8} & \omega^{-4} \end{bmatrix} \begin{bmatrix} \gamma_0 \\ \gamma_1 \\ \gamma_2 \\ \gamma_{-2} \\ \gamma_{-1} \end{bmatrix} = \begin{bmatrix} f(x_0) \\ f(x_1) \\ f(x_2) \\ f(x_3) \\ f(x_4) \end{bmatrix}, \tag{1.31}$$

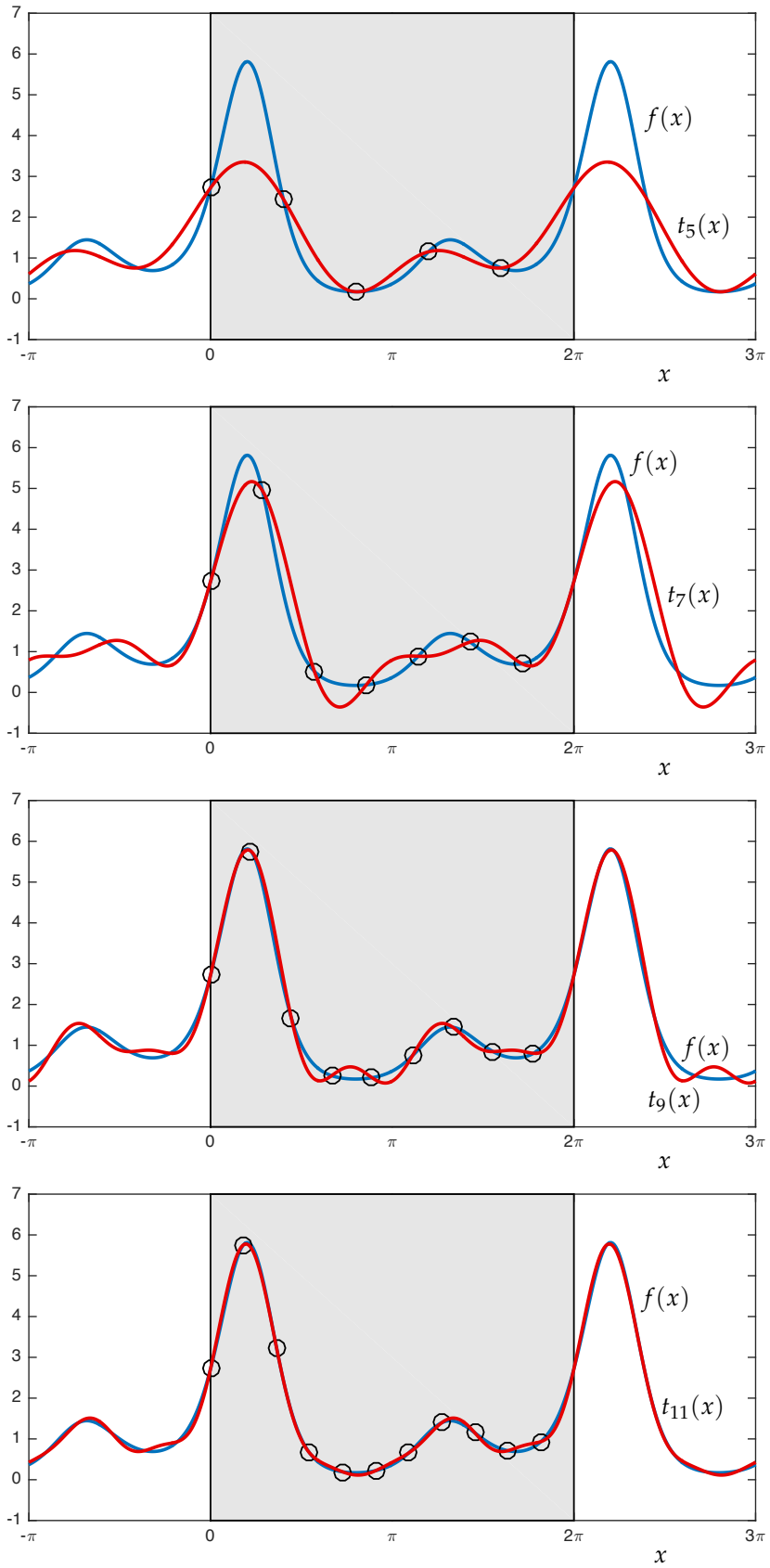


Figure 1.14: Trigonometric interpolant to 2π-periodic function $f(x) = e^{\cos(x)+\sin(2x)}$, using $n = 5, 7, 9$ and 11 points uniformly spaced over $[0, 2\pi)$ ($\{x_k\}_{k=0}^{n}$ for $x_k = 2\pi k/n$). Since both f and the interpolant are periodic, the function fits well throughout $\mathbb{R}$, not just on the interval for which the interpolant was designed.

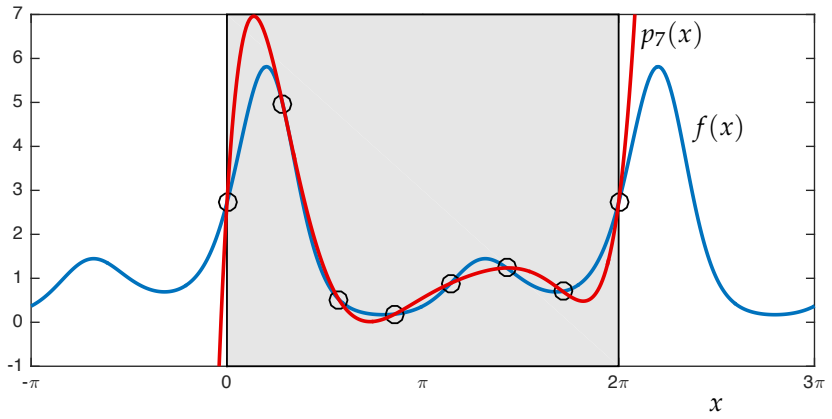


Figure 1.15: Polynomial fit of degree $n = 7$ through uniformly spaced grid points $x_0, \ldots, x_n$ for $x_j = 2\pi j/n$, for the same function $f(x) = e^{\cos(x)+\sin(2x)}$ used in Figure 1.14. In contrast to the trigonometric fits in the earlier figure, the polynomial grows very rapidly outside the interval $[0, 2\pi]$. Moral: if your function is periodic, fit it with a trigonometric polynomial.

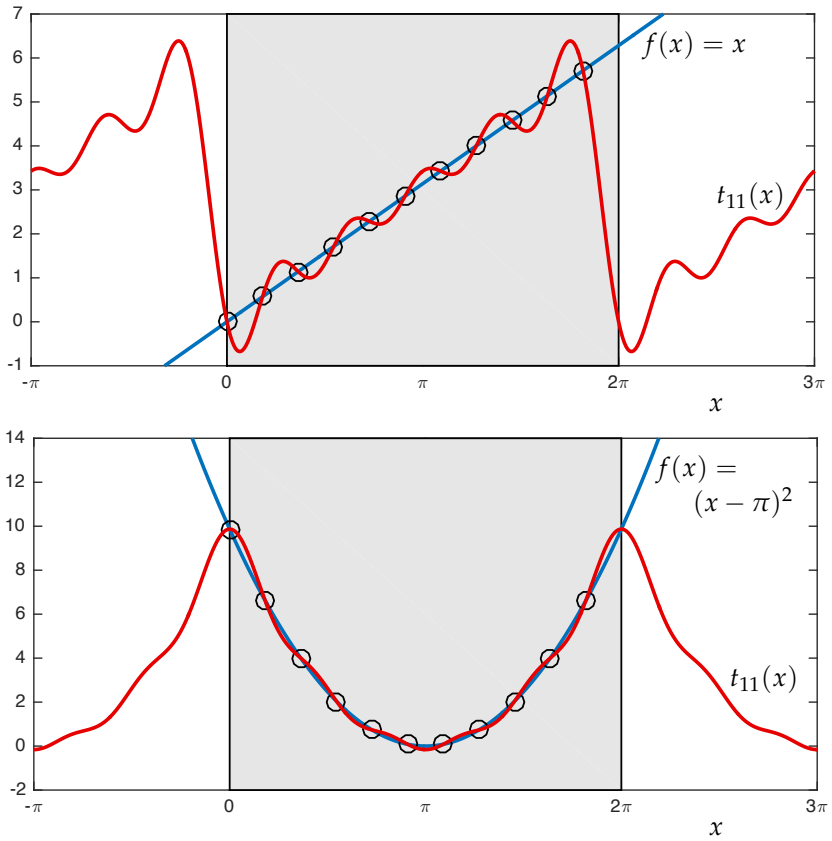


Figure 1.16: Trigonometric polynomial fit of degree $n = 11$ through uniformly spaced grid points $x_0, \ldots, x_n$ for $x_j = 2\pi j/n$, for the *non-periodic* function $f(x) = x$ (top) and for $f(x) = (x-\pi)^2$ (bottom). By restricting the latter function to the domain $[0, 2\pi]$, one can view it as a continuous periodic function with a jump discontinuity in the first derivative. The interpolant t_{11} seems to give a good approximation to f, but the discontinuity in the derivative slows the convergence of t_n to f as $n \to \infty$.

which amounts to reordering the *columns* of the matrix in (1.30). You can obtain this matrix by the command `ifft(eye(n))`. For $n = 5$,

$$5 * \texttt{ifft(eye(5))} = \begin{bmatrix} \omega^0 & \omega^0 & \omega^0 & \omega^0 & \omega^0 \\ \omega^0 & \omega^1 & \omega^2 & \omega^{-2} & \omega^{-1} \\ \omega^0 & \omega^2 & \omega^4 & \omega^{-4} & \omega^{-2} \\ \omega^0 & \omega^3 & \omega^6 & \omega^{-6} & \omega^{-3} \\ \omega^0 & \omega^4 & \omega^8 & \omega^{-8} & \omega^{-4} \end{bmatrix}.$$

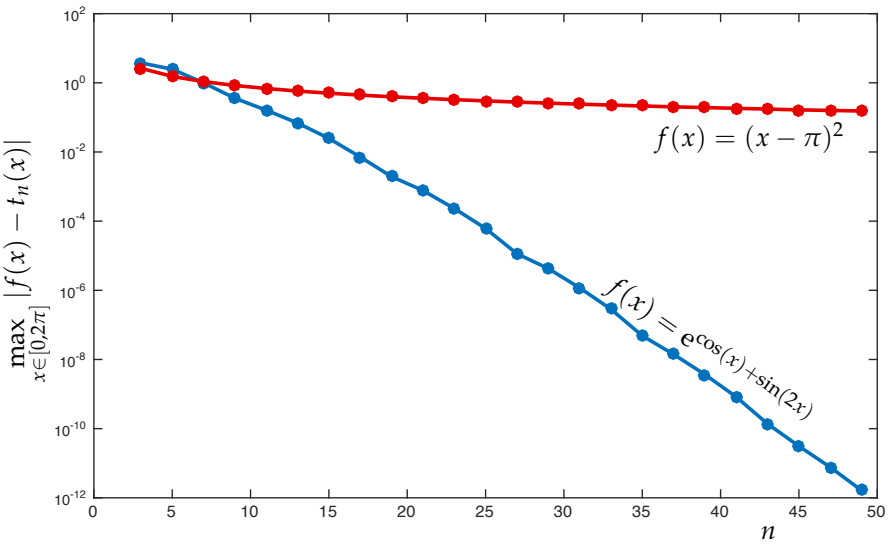


Figure 1.17: Convergence of the trigonometric polynomial interpolants to $f(x) = e^{\cos(x)+\sin(2x)}$ and $f(x) = (x-\pi)^2$. For the first function, convergence is extremely rapid as $n \to \infty$. The second function, restricted to $[0, 2\pi]$, can be viewed as a continuous but not continuously differentiable function. Though the approximation in Figure 1.16 looks good over $[0, 2\pi]$, the convergence of t_n to f is slow as $n \to \infty$.

Similarly, the inverse of this matrix can be computed from `fft(eye(n))` command:

$$\texttt{fft(eye(5))}/5 = \frac{1}{5}\begin{bmatrix} \omega^0 & \omega^0 & \omega^0 & \omega^0 & \omega^0 \\ \omega^0 & \omega^{-1} & \omega^{-2} & \omega^{-3} & \omega^{-4} \\ \omega^0 & \omega^{-2} & \omega^{-4} & \omega^{-6} & \omega^{-8} \\ \omega^0 & \omega^2 & \omega^4 & \omega^6 & \omega^8 \\ \omega^0 & \omega^1 & \omega^2 & \omega^3 & \omega^4 \end{bmatrix}.$$

We could construct this matrix and multiply it against $\mathbf{f}$ to obtain γ, but that would require $\mathcal{O}(n^2)$ operations. Instead, we can compute γ directly using the `fft` command:

$$\gamma = \texttt{fft}(\mathbf{f})/\texttt{n}.$$

Recall that this reordered vector gives

$$\gamma = \begin{bmatrix} \gamma_0 \\ \gamma_1 \\ \gamma_2 \\ \gamma_{-2} \\ \gamma_{-1} \end{bmatrix},$$

which must be taken into account when constructing t_n.

Example 1.10 (MATLAB code for trigonometric interpolation)**.**
We close with a sample of MATLAB code one could use to construct the interpolant t_n for the function $f(x) = e^{\cos(x)+\sin(2x)}$. First we present a generic code that will work for any (real- or complex-valued) 2π-periodic f. Take special note of the simple one line command to find the coefficients γ.

```
f = @(x) exp(cos(x)+sin(2*x));                % define the function
n = 5;                                        % # terms in trig polynomial (must be odd)
xk = [0:n-1]'*2*pi/n;                         % interpolation points
xx = linspace(0,2*pi,500)';                   % fine grid on which to plot f, t_n
tn = zeros(size(xx));                         % initialize t_n

gamma = fft(f(xk))/n;                         % solve for coefficients, gamma

for k=1:(n+1)/2
   tn = tn + gamma(k)*exp(1i*(k-1)*xx);       % add in gamma_0, gamma_1, ... gamma_{(n-1)/2}
end
for k=(n+1)/2+1:n
   tn = tn + gamma(k)*exp(1i*(-n+k-1)*xx);    % add in gamma_{-(n-1)/2}, ..., gamma_{-1}
end
plot(xx,f(xx),'b-'), hold on                  % plot f
plot(xx, tn,'r-')                             % plot t_n
```

In the case that f is real-valued (as with all the examples shown in this section), one can further show that

$$\gamma_{-k} = \overline{\gamma_k},$$

indicating that the imaginary terms will not make any contribution to t_n. Since for $k = 1, \ldots, (n-1)/2$,

$$\gamma_{-k} e^{-1ikx} + \gamma_k e^{1ikx} = 2\Big(\mathrm{Re}(\gamma_k)\cos(kx) - \mathrm{Im}(\gamma_k)\sin(kx)\Big),$$

the code can be simplified slightly to construct t_n as follows.

```
gamma = fft(f(xk))/n;                         % solve for coefficients, gamma
tn = gamma(1)*exp(1i*0*xx);                   % initialize t_n(x) = gamma_0
for k=2:(n+1)/2
   tn = tn + 2*real(gamma(k))*cos((k-1)*xx) ...
           - 2*imag(gamma(k))*sin((k-1)*xx);  % exploit gamma_{-k} = conj(gamma_k)
end
```