Math 563 Lecture Notes
The discrete Fourier transform

Spring 2020

The point: A brief review of the relevant review of Fourier series; introduction to the DFT and its good properties (spectral accuracy) and potential issues (aliasing, error from lack of smoothness). There are a vast number of sub-topics and applications that could be discussed; a few examples are included at the end (we'll likely make use of the DFT again later in solving differential equations).

Related reading: Details on the DFT can be found in Quarteroni, . Many other sources have good descriptions of the DFT as well (it's an important topic), but beware of slightly different notation. Reading the documentation for `numpy` or Matlab's `fft` is suggested as well, to see how the typical software presents the transform for practical use.

1 Fourier series (review/summary)

We consider functions in $L^2[0, 2\pi]$ (with weight $w(x) = 1$), which have a **Fourier series**

$$f = \sum_{k=-\infty}^{\infty} c_k e^{ikx}, \quad c_k = \frac{1}{2\pi} \int_0^{2\pi} f(x) e^{-ikx}\, dx.$$

The basis functions $\phi_k = e^{ikx}$ are orthogonal in the inner product $\langle f, g \rangle = \int_0^{2\pi} f(x)g(x)\, dx$.

In this section, the space $L^2[0, 2\pi]$ is regarded as the space of 2π-**periodic** functions, i.e. functions defined for $x \in \mathbb{R}$ that satisfy

$$f(x) = f(x + 2\pi) \text{ for all } x.$$

By this property, the function is defined by its values on one period, so it suffices to consider the function on the interval $[0, 2\pi]$. The main point is that continuity of the periodic function requires that the endpoints match:

$$f \text{ continuous in } [0, 2\pi] \textbf{ and } f(0) = f(2\pi).$$

The same goes for differentiability. The 'periodic' definition is key, as error properties will depend on the smoothness of f **as a periodic function**.

If f is real-valued, it is not hard to show that

$$c_{-k} = \overline{c_k}.$$

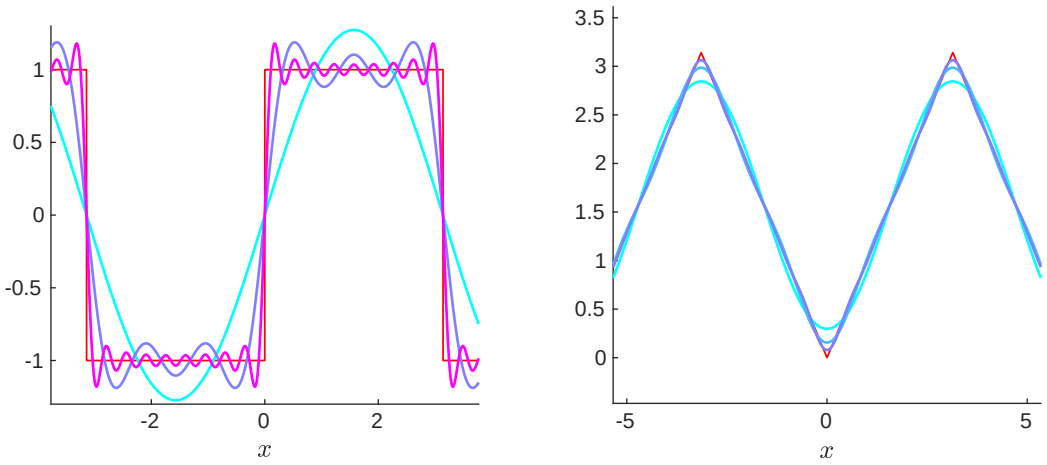
Writing the coefficients as $c_n = \frac{1}{2}(a_k - ib_k)$ we have (by grouping $+k$ and $-k$ terms)

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty} a_n \cos kx + b_n \sin kx$$

which is the Fourier series in real form. Two standard examples of Fourier series:

$$\textbf{square wave:} \quad f_S(x) = \begin{cases} 1 & x > 0 \\ -1 & x < 0 \end{cases}, \qquad x \in [-\pi, \pi] \tag{1}$$

$$\textbf{triangle wave:} \quad f_T(x) = |x|, \qquad x \in [-\pi, \pi] \tag{2}$$



The Fourier series for the square wave is straightforward to calculate:

$$f_S(x) = \frac{4}{\pi} \sum_{n \text{ odd}} \frac{1}{n} \sin nx \quad \text{or} \quad f_S(x) = \frac{4}{\pi} \sum_{n=1}^{\infty} \frac{1}{2n-1} \sin((2n-1)x).$$

Similar to the square wave, we get for the triangle wave that

$$f_T(x) = \frac{1}{2} - \frac{4}{\pi} \sum_{n=1}^{\infty} \frac{1}{(2n-1)^2} \cos((2n-1)x).$$

Convergence: The partial sums of the Fourier series are least-squares approximations with respect to the given basis. In particular, if f is real then

$$\lim_{N \to \infty} \|f - S_N\|_2 = 0, \quad S_N := \sum_{k=-N}^{N} c_k e^{ikx} = \frac{a_0}{2} + \sum_{k=1}^{N} a_k \cos kx + b_k \sin kx.$$

More can be said about the coefficients; by integrating the formula for c_k by parts we can obtain the following bounds:

These bounds, coupled with Parseval's theorem, connect the convergence rate of the series to the smoothness of f **as a periodic function**.

The Fourier series allows for some discontinuities. Define

$$f(x^-) = \lim_{\xi \nearrow x} f(\xi), \quad f(x^+) = \lim_{\xi \searrow x} f(\xi).$$

Theorem (Pointwise/uniform convergence): Let $S_N(x)$ be the N-th partial sum of the Fourier series for $f \in L^2[-\ell, \ell]$. If f and f' are piecewise continuous, then

$$\lim_{N\to\infty} S_N(x) = \overline{f}(x) := \begin{cases} f(x) & \text{if } f \text{ is continuous at } x \\ \frac{1}{2}(f(x^-) + f(x^+)) & \text{if } f \text{ has a jump at } x. \end{cases}$$

That is, the partial sums converge **pointwise** to the average of the left and right limits.

Near a discontinuity x_0, one has **Gibbs' phenomenon**, where the series fails to converge uniformly, leaving a maximum error near x_0 of about 9% of the discontinuity (evident in the square wave example above).

1.1 Decay of Fourier coefficients

Consider L^2 functions in $[0, 2\pi]$ and the Fourier series

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos nx + b_n \sin nx.$$

Worst case: Suppose f as a periodic function is piecewise continuous but has a jump (e.g. the square wave). Then

$$a_n, b_n = O(1/n).$$

This is enough to get norm convergence:

$$\|f - S_N\|^2 = \text{const.} \cdot \sum_{n=N+1}^{\infty} (a_n^2 + b_n^2) = O(1/N).$$

However, $1/n$ decay is not enough to get convergence at each point ($\sum 1/n$ diverges).

Nicer case: However, if the function is smoother then we can do better. If $f \in L^2[-\pi, \pi]$ and its derivatives up to $f^{(p-1)}$ are continuous **as periodic functions** and $f^{(p)}$ is continuous except at a set of jumps then, for some constant C,

$$|a_n| \le \frac{C}{n^{p+1}}, \quad |b_n| \le \frac{C}{n^{p+1}}.$$

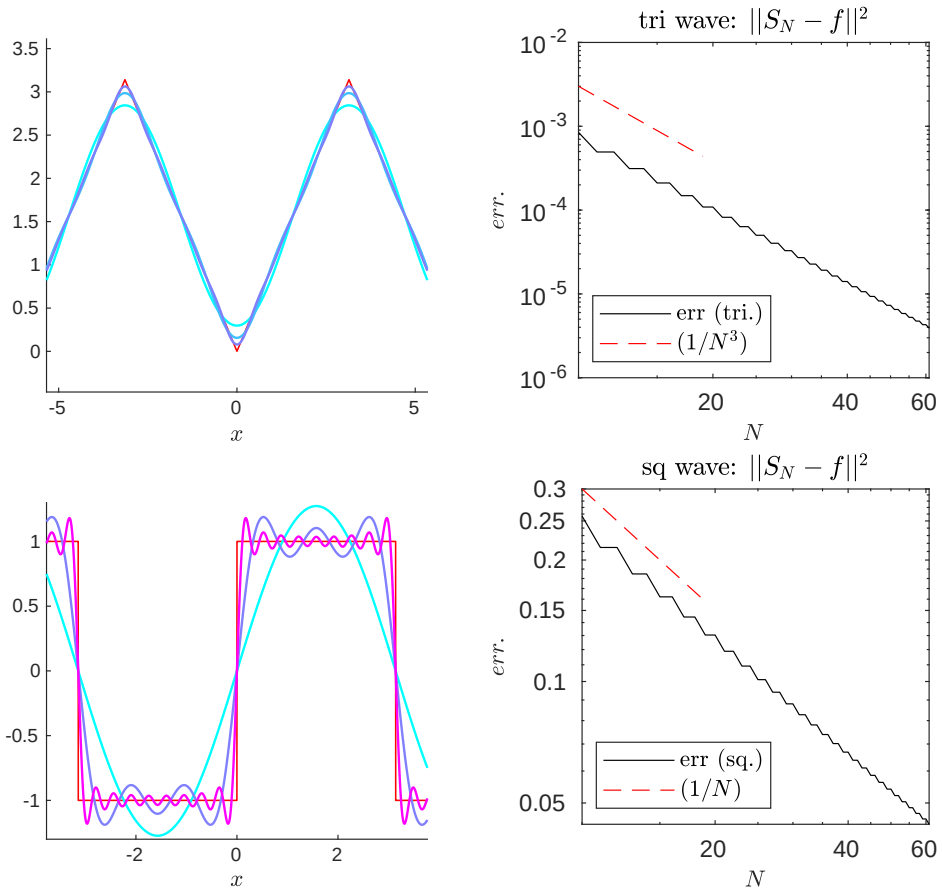


Figure 1: Partial sums for the triangle/square wave and log-log plot of the squared L^2 error $\int_{-\pi}^{\pi} |S_N(x) - f(x)|^2 \, dx$. For the triangle, $c_n \sim 1/n^2$ and the error decreases like $\sum_{n=N}^{\infty}(1/n^2)^2 \sim 1/N^3$. For the square, $c_n \sim 1/n$; the error decreases as $\sum_{n=N}^{\infty}(1/n)^2 \sim 1/N$.

Thus, in general, smoother $f \implies$ faster convergence of its Fourier series. Informally, we get one factor of $1/n$ for each derivative (as a periodic function), starting with the 0-th and ending with the first one that has a jump. For the square/triangle:

$$\text{square} \implies \text{jump in } f \implies c_n \sim 1/n$$

$$\text{tri.} \implies f \text{ cts.} + \text{jump in } f' \implies c_n \sim 1/n^2.$$

Smooth functions: If the function $f(x)$ is smooth and 2π-periodic (derivatives of all orders), the the coefficients have the exponential bound

$$|a_n| \leq e^{-\alpha n}$$

where α (The decay rate) is the distance from $[0, 2\pi]$ to the nearest singularity of the function in the complex plane. That is, less singular functions have coefficients that decay faster.

Informal definition: When an error has a bound of the form

$$\|f - \tilde{f}_n\| \leq Cn^{-p} \text{ if } f \in C^p,$$

$$\|f - \tilde{f}_n\| \leq Ce^{-an} \text{ if } f \in C^\infty$$

we say the approximation has **spectral accuracy**. That is, the error behaves like the Fourier series error (you will sometimes see this referred to as **exponential type**). We have already seen that the trapezoidal rule, for instance, has spectral accuracy while a spline approximation does not.

The benefit (and disadvantage, sometimes) of the property is that the **smoothness of** f determines the convergence rate. When f is smooth, this is great; when f is not smooth, the lack of smoothness can be a problem.

2 The discrete Fourier transform

2.1 Getting to the discrete form: using the trapezoidal rule

We know from the previous theory that the partial sums are best approximations in the least-squares sense (minimizing the error in the L^2 norm):

$$g = S_N = \sum_{k=-N}^{N} c_k e^{ikx} \text{ minimizes } \|f - g\|_2 \text{ for functions of the form } g = \sum_{k=-N}^{N} (\cdots) e^{ikx}.$$

Computing the coefficients c_k requires estimating the integral.

As we will see, the trapezoidal rule is the right choice. Let $x_j = jh$ with $h = 2\pi/N$. Assume that f is periodic and continuous, so $f_0 = f_N$. Using the trapezoidal rule,

$$\begin{aligned} c_k &= \frac{1}{2\pi} \int_0^{2\pi} f(x) e^{-ikx}\, dx \\ &\approx \frac{h}{2\pi} \left(\frac{f_0}{2} + \frac{f_N}{2} + \sum_{j=1}^{N-1} f_j e^{-ikx_j} \right) \\ &= \frac{1}{N} \sum_{j=0}^{N-1} f_j e^{-ikhj} \\ &= \frac{1}{N} \sum_{j=0}^{N-1} f_j e^{-2\pi ikj/N} \end{aligned}$$

The transformation from f_j to c_k is the **discrete Fourier transform**:

$$\{f_j\}_{j=0}^{N-1} \to \{F_k\}_{k=0}^{N-1}, \qquad F_k = \frac{1}{N} \sum_{j=0}^{N-1} f_j e^{-2\pi ikj/N}.$$

We may then use this transform to estimate partial sums of the Fourier series (the least-squares approximation to f) using data at equally spaced points.

Note that if f is real and $m < N/2$ then, writing $F_k = (a_k - ib_k)/2$,

$$f(x) \approx \sum_{k=-m}^{m} F_k e^{ikx} = \frac{a_0}{2} + \sum_{k=1}^{m} a_k \cos kx + b_k \sin kx \tag{3}$$

where, from the identity $e^{iNx_j} = 1$,

$$c_k = c_{N-k} \text{ for } k < 0.$$

In this way the DFT $\{c_k\}$ is 'cyclic' (it wraps around for indices past N). Note that the DFT approximation (3) **is not quite the Fourier series partial sum**, because the F_k's are not equal to the Fourier series coefficients (but they are close!).

To get a better understanding, we should be more careful; at present, it is not clear why the trapezoidal rule should be used for the integral.

2.2 The discrete form (from discrete least squares)

Instead, we derive the transform by considering 'discrete' approximation from data. Let $x_0, \cdots, x_N$ be equally spaced nodes in $[0, 2\pi]$ and suppose the function data is given at the nodes. Remarkably, the basis $\{e^{ikx}\}$ is also orthogonal in the discrete inner product

$$\langle f, g \rangle_d = \sum_{j=0}^{N-1} f(x_j)\overline{g(x_j)}.$$

Precisely, for integers j, k we have

$$\langle e^{ijx}, e^{ikx} \rangle_d = \begin{cases} 0 \text{ if } j \neq k \\ N \text{ if } j = k \end{cases}.$$

The general theory for least-squares approximation then applies (see HW). For ease of notation, the Fourier coefficients for f are denoted with a capital letter. It follows that

$$f \approx \sum_{k=-m}^{m} F_k e^{ikx}, \quad F_k = \frac{\langle f, e^{ikx} \rangle_d}{\langle e^{ikx}, e^{ikx} \rangle_d} = \frac{1}{N} \sum_{j=0}^{N-1} f(x_j) e^{-ikx_j}$$

which is exactly the discrete Fourier transform. Moreover, the orthogonality relation gives a formula for the inverse transform. The result is the following:

Definition (Discrete Fourier transform): Suppose $f(x)$ is a 2π-periodic function. Let $x_j = jh$ with $h = 2\pi/N$ and $f_j = f(x_j)$. The **discrete Fourier transform** of the data $\{f_j\}_{j=0}^{N-1}$ is the vector $\{F_k\}_{k=0}^{N-1}$ where

$$F_k = \frac{1}{N}\sum_{j=0}^{N-1} f_j e^{-2\pi i k j/N} \tag{4}$$

and it has the **inverse transform**

$$f_j = \sum_{k=0}^{N-1} F_k e^{2\pi i k j/N}. \tag{5}$$

Letting $\omega_N = e^{-2\pi i/N}$, the transform and inverse can be written as

$$F_k = \frac{1}{N}\sum_{j=0}^{N-1} f_j \omega_N^{jk}, \quad f_j = \sum_{j=0}^{N-1} F_k \omega_N^{-jk}.$$

Proof. The inverse formula is not hard to prove; just use the definitions and the discrete orthogonality relation to calculate

$$\begin{aligned}
\sum_{k=0}^{N-1} F_k e^{ikx_j} &= \sum_{k=0}^{N-1}\sum_{m=0}^{N-1} f_m e^{-ikx_m} e^{ikx_j} \\
&= \sum_{m=0}^{N-1} f_m \sum_{k=0}^{N-1} e^{ik(x_j - x_m)} \\
&= \sum_{m=0}^{N-1} f_m \langle e^{ijx}, e^{imx}\rangle_d \\
&= N f_j.
\end{aligned}$$

□

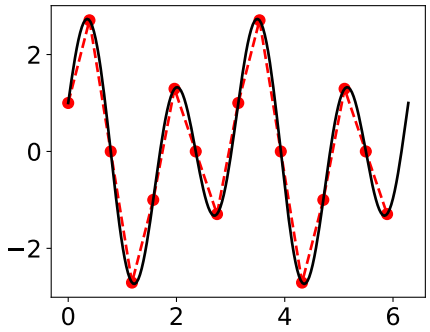
2.3 Aliasing

As a simple (but subtle) example, consider

$$f(x) = \cos 2x + 2\sin 4x.$$

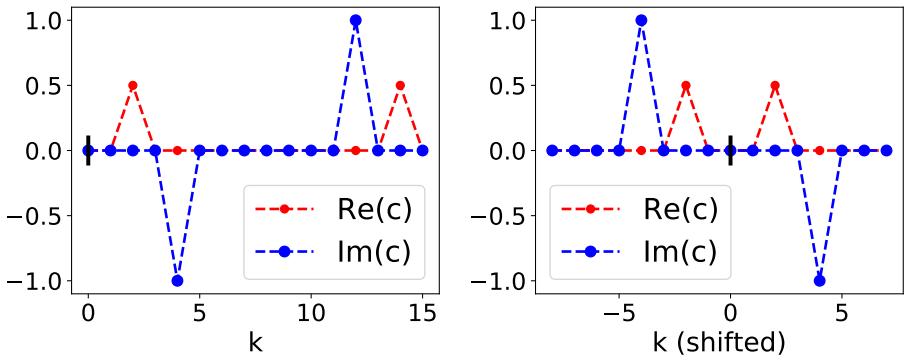
The frequencies in this function are ± 2 (with coeffs. $1/2$) and ± 4 (with coeffs. $\pm 2/i$). The DFT, therefore, should have peaks at these values of k.

Consider taking N equally spaced samples ($x_j = jh$ with $h = 2\pi/N$ and $j = 0, \cdots, N-1$) with $N = 6$ and $N = 16$ as shown below.

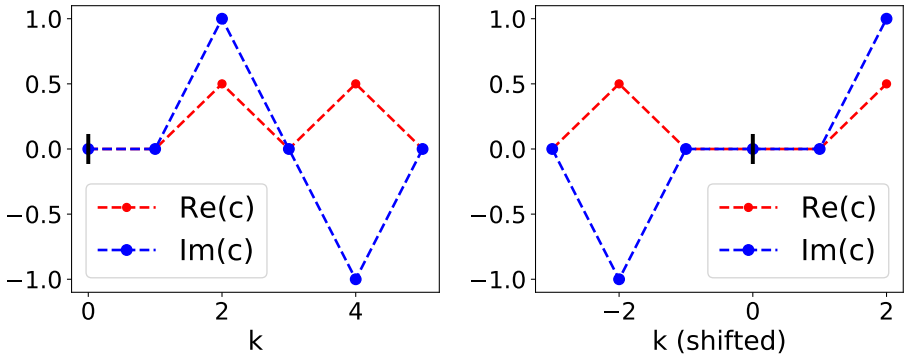


Since $e^{16ix_j} = 1$ for all j, the frequencies -2 and -4 appear in the transform in the $k = 14$ and $k = 12$ places (they 'wrap around' from zero to the other side of the array).

The real/imaginary parts of the DFT are shown below, confirming that it works. Since the underlying function is real, it makes sense to 'center' the DFT by shifting the data by $N/2$ so that $k = 0$ is in the center; the plot then has a nice symmetry.



However, when $N = 6$, the available frequencies are only $k = 0, 1, 2, 3$. The DFT sees $k = 4$ as -2 and $k = -4$ as 2 - as if the function were $-2\sin 2x$. There are not enough samples to resolve the frequency, which causes it to 'wrap back around' and appear to be a different value (this effect is called **aliasing**).



The example here suggests that the range of frequencies used in the DFT (the 'bandwidth') must be wide enough to fit the bandwidth of the function to avoid aliasing. In particular, the number of sample points N must be at least twice the largest frequency (the **Nyquist sampling rate**).

2.4 Real functions, interpolation

Definition A **trigonometric polynomial** of degree d is a polynomial in $e^{\pm ix}$ of degree d, e.g. $\cos^2\theta + \sin^2\theta$ is a trig. polynomial of degree 2.

The inverse formula (5) shows that the trigonometric polynomial

$$p_N(x) = \sum_{k=0}^{N-1} F_k e^{ikx} \tag{6}$$

satisfies

$$p_N(x_j) = f_j \text{ for } j = 0, \cdots, N-1.$$

Thus, it is a 'trigonometric interpolant' for the data (x_j, f_j). However, there is a catch! If $f(x)$ is real, then $p_N(x)$ given by (6) may be complex (away from the interpolation points).

When the function $f(x)$ is real, we should instead 'symmetrize' the formula to construct a real-valued interpolant. Let $N = 2m$ be even. Observe that if $x = x_j$ is a node then

$$\begin{aligned}
p_N(x_j) &= \sum_{k=0}^{N-1} F_k e^{ikx_j} \\
&= \sum_{k=0}^{N/2} F_k e^{2\pi ijk/N} + \sum_{k=N/2+1}^{N-1} F_k e^{2\pi ijk/N} \\
&= \sum_{k=0}^{N/2} F_k e^{2\pi ijk/N} + \sum_{k=-(N/2-1)}^{-1} F_{k+N} e^{2\pi ijk/N} && \text{(shift sum by } N\text{)} \\
&= \sum_{k=-(m-1)}^{m} \tilde{F}_k e^{ikx_j} \\
&= \sum_{k=-(m-1)}^{m-1} \tilde{F}_k e^{ikx_j} + \frac{\tilde{F}_m}{2} e^{iNx_j/2} + \frac{\tilde{F}_m}{2} e^{-iNx_j/2} && \text{(split into } \pm N/2 \text{ terms)}
\end{aligned}$$

where

$$\tilde{F}_k = \begin{cases} F_{k+N} & 1 - N/2 \le k < 0 \\ F_k & 0 \le k \le N/2 \end{cases}.$$

One can check that

$$\tilde{F}_{-k} = \overline{\tilde{F}_k}, \qquad \tilde{F}_m, \tilde{F}_0 \text{ are real.}$$

Writing $\tilde{F}_k = \frac{1}{2}(a_k - b_k i)$ and $\tilde{F}_m = a_m$, we therefore have an interpolant in the form

$$p_N(x) = \frac{a_0}{2} + \frac{a_m}{2}\cos mx + \sum_{k=1}^{m-1} a_k \cos kx + b_k \sin kx.$$

Note that the $k = N/2$ coefficient gets split into $\pm N/2$ parts (it is aliased!) so it becomes both halves of the $\cos mx$ term; it has no imaginary part so there is no $\sin mx$ term.

Caution (uniqueness): Observe that the calculation shows that the interpolant is not unique; we are allowed to shift by various multiples in the exponent and still have the interpolation property. The DFT can only distinguish frequencies k up to a multiple of N; all functions $e^{i(k+pN)x}$ are the same at the given data points for any integer p.

2.5 Example (interpolant)

Consider the trigonometric interpolant for

$$f(x) = x(2\pi - x), \quad x \in [0, 2\pi]$$

using the points $0, \pi/2, \pi, 3\pi/2$ (so $N = 4$). Note that 2π is also an interpolation point, but is equivalent to zero.

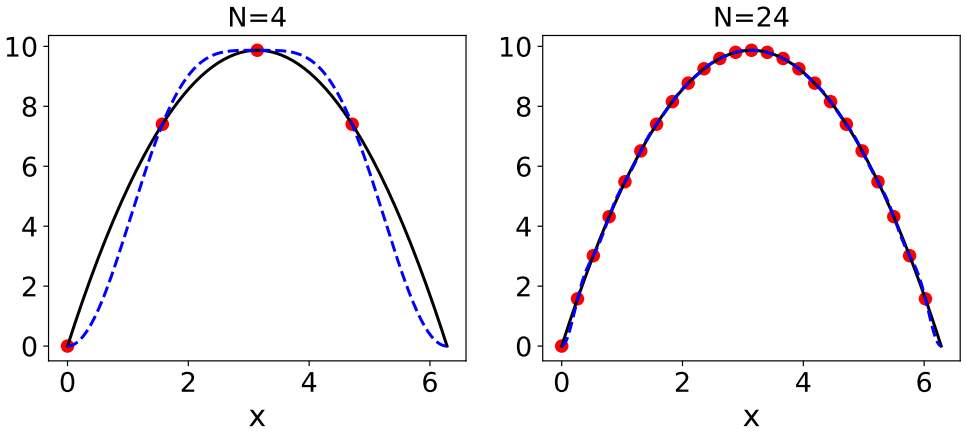
In complex form, the interpolant is

$$p_2(x) = \frac{F_2}{2}e^{-2ix} + F_3 e^{-ix} + F_0 + F_1 e^{ix} + \frac{F_2}{2}e^{2ix}.$$

All the F's are real here (check this!). Setting $F_k = a_k/2$ gives

$$p(x) = \frac{a_0}{2} + a_1 \cos x + \frac{a_2}{2} \cos 2x.$$

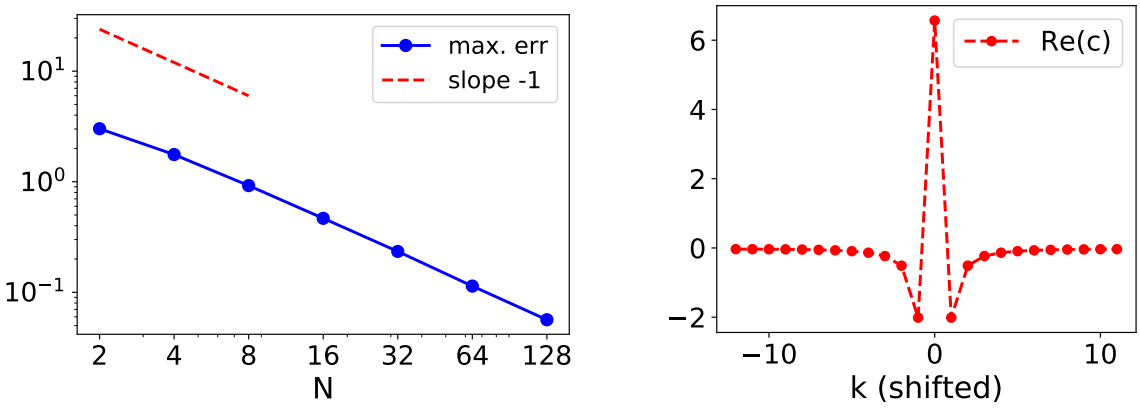
With more points, the approximation improves. To better understand the error, we can use the Fourier series theory.



Since $f(x)$ (as a periodic function) is continuous but f' has a jump at $x = 0$, the coefficients decay like $1/k^2$ (see subsection 1.1). This suggests that

$$\max_{x\in[0,2\pi]} |p_N(x) - f(x)| \sim \frac{C}{N} \text{ as } N \to \infty$$

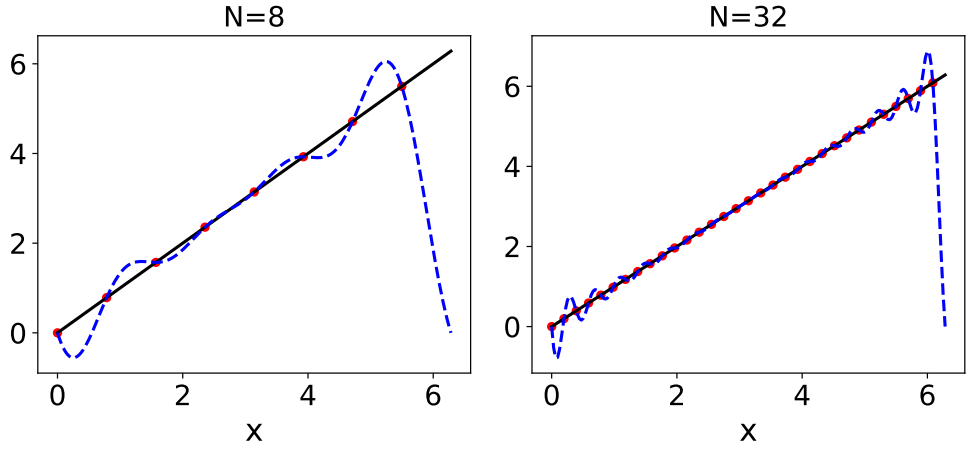
which is indeed the case. The error is not exciting here, because the lack of differentiability at the endpoint causes trouble (on the other hand, if f is smooth and periodic, then the approximation will be much better!).[1]



For a more dramatic example, suppose instead we interpolate

$$f(x) = x, \quad x \in [0, 2\pi].$$

The function is not continuous (with a jump $x = 0$). Using the N points $0, 2\pi/N, \cdots 2\pi(N-1)/N)$ means the interpolant matches the function at $x = 0$, but has the wrong behavior as $x \to 2\pi$ (the interpolant must be continuous!).



In addition, we observe Gibbs' phenomnenon in the approximation; the effect of the discontinuity spoils the approximation nearby.

To get around this, the standard solution is to use **smoothing** - the coefficients in the Fourier series are 'smoothed out' by adding a numerical factor that improves the smoothness without changing the function much (see, for instance, **Lanczos smoothing** in the textbook).

[1] The result is true for Fourier series, as discussed. From the first derivation of the DFT, we saw that the Fourier series and DFT approximation differ by a trapezoidal rule application. Thus, it's plausible that the DFT inherits the same convergence properties as the Fourier series, which is more or less the case.

2.6 The Fast Fourier transform (the short version)

The discrete Fourier transform has the form

$$F_k = \frac{1}{N}\sum_{j=0}^{N-1} f_j \omega_N^{kj}$$

where $\omega_N = e^{-2\pi i/N}$. Each component takes $O(N)$ operations to compute and there are N components, so naively, evaluating the DFT takes $O(N^2)$ operations.

However, the coefficients in the sum are not arbitrary! They are powers of ω, and that structure can be exploited to save work. The idea is worth seeing to understand why it is faster; the nasty implementation details are omitted here in favor of showing the idea.[2]

Let $\omega_N = e^{-2\pi i/N}$ and assume that

$$N = 2^M \text{ is a power of 2.}$$

The idea is to split the sum into 'odd' and 'even' parts. We write

$$F_k = S_k + T_k$$

where (with the sum over indices in the range 0 to $N-1$)

$$S_k = \sum_{j \text{ even}} f_j \omega_N^{kj}, \quad T_k = \sum_{j \text{ odd}} f_j \omega_N^{kj}$$

Since N is even, the even sum can be written in terms of powers of $\omega_{N/2}$

$$S_k = \sum_{m=0}^{N/2-1} f_{2m} e^{-2\pi i k(2m)/N} = \sum_{m=0}^{N/2} f_{2m} (\omega_{N/2})^{km}$$

$$\implies S_k = DFT((f_0, f_2, \cdots, f_{N-2}))_k$$

and the odd sum can be written the same way, factoring one power of ω out:

$$T_k = \sum_{m=0}^{N/2-1} f_{2m+1} e^{-2\pi i k(2m+1)/N} = e^{-2\pi i k/N} \sum_{m=0}^{N/2} f_{2m+1} (\omega_{N/2})^{km}$$

$$\implies T_k = e^{-2\pi i k/N} DFT((f_1, f_3, \cdots, f_{N-1}))_k$$

The DFT therefore requires

- two DFTs of length $N/2$ for the odd/even parts
- $O(N)$ operations to combine the odd/even parts T_k and S_k (by addition)

[2] Numerical Recipes, 3rd edition has a good description of the implementation details.

This calculation defines a recursive process for computing the DFT. If N is a power of two then $\log_2(N)$ iterations of the process are required, so a length N DFT requires $O(N \log N)$ operations.

The algorithm described here is a first version of the **Fast Fourier transform**. Three components are missing:

- Proper handling of cases when N is not a power of 2
- Mkaing the algorithm not use true recursion. By some clever encoding, one can write the algorithm in a non-recursive way (using loops), which avoids overhead and allows for optimization.
- The 'base case' when N is small. One can go all the way to $N = 1$ or 2; in practice it is most efficient to implement a super-efficient base case at certain values of N.

The gain in efficiency is quite profound here - going from $O(N^2)$ to $O(N \log N)$ converts an N to $\log N$, which is often orders of magnitude better. The fact that the scaling with N is almost linear means that the FFT is quite fast in practice - and its speed and versatile use makes it one of the key algorithms in numerical computing.

2.7 Convolution

Suppose f, g are periodic functions with period 2π. A important quantity one often cares about is the 'periodic' or 'circular' **convolution**

$$(f * g)(y) = \int_0^{2\pi} f(x - y)g(y)\, dy.$$

For instance, the convolution of f with a Gaussian smooths out that function (used e.g. in blurring images - 'Gaussian blurring').

The **discrete** circular convolution of data vectors $(f_0, \cdots, f_{N-1})$ and $(g_0, \cdots, g_{N-1})$, presumably sampled from periodic data as in the DFT, is

$$(f * g)_j = \sum_{\ell=0}^{N-1} f_\ell g_{j-\ell}, \qquad j = 0, \cdots, N-1.$$

where the indices are 'mod N' (so they wrap around, e.g. N become 0, $N+1$ becomes 1 and so on). As you may recall from Fourier analysis, the Fourier transform of a convolution is the product of the transforms; the same property holds for the discrete Fourier transform.

Precisely, we have that if

$$\mathrm{DFT}(f) = (F_0, \cdots, F_{N-1}), \quad \mathrm{DFT}(g) = (G_0, \cdots G_{N-1})$$

then

$$\mathrm{DFT}(f * g)_k = F_k G_k.$$

The proof is straightforward (if one is careful with interchanging summation order):

$$\begin{aligned}
\mathrm{DFT}(f*g)_k &= \sum_{j=0}^{N-1}\left(\sum_{\ell=0}^{N-1} f_\ell g_{j-\ell}\right)\omega^{kj} \\
&= \sum_{j=0}^{N-1}\left(\sum_{\ell=0}^{N-1} f_\ell g_{j-\ell}\right)\omega^{k\ell}\omega^{k(j-\ell)} \\
&= \sum_{\ell=0}^{N-1} f_\ell \omega^{k\ell}\sum_{j=0}^{N-1} g_{j-\ell}\omega^{k(j-\ell)} \\
&= \left(\sum_{\ell=0}^{N-1} f_\ell \omega^{k\ell}\right) G_k \\
&= F_k G_k
\end{aligned}$$

using periodicity of $\omega^{k(j-\ell)}$ to simplify the shifted sum for g:

$$\sum_{j=0}^{N-1} g_{j-\ell}\omega^{k(j-\ell)} = \sum_{\tilde{j}=-\ell}^{N-1-\ell} g_{\tilde{j}}\omega^{k\tilde{j}} = \sum_{\tilde{j}=0}^{N-1} g_{\tilde{j}}\omega^{k\tilde{j}} = G_k.$$

The DFT thus provides an efficient way to compute convolutions of two functions.

2.8 The trapezoidal rule, again

Let $T_N(x)$ denote the composite trapezoidal rule using points $x_0, \cdots x_N$ in $[0, 2\pi]$ (equally spaced). Then we claim that

$$T_N f = \int_0^{2\pi} f(x)\,dx \text{ if } f(x) = e^{ikx},\ |k| < N.$$

To see this, calculate (with $h = 2\pi/N$)

$$T_N f = \frac{1}{2}f(0) + \frac{1}{2}f(2\pi) + h\sum_{j=1}^{N-1} e^{ikx_j} = \frac{2\pi}{N}\sum_{j=0}^{N-1} e^{ikx_j} = \mathrm{DFT}(g(x)=2\pi) = \begin{cases} 2\pi & k=0 \\ 0 & |k|<N \end{cases}.$$

In both cases, the right hand side is the exact value of the integral, which proves the claim.

Now suppose $f(x)$ has a Fourier series

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty} a_k \cos kx + b_k \sin kx = \lim_{N\to\infty} S_N.$$

Then each point added to the trapezoidal rule **improves the accuracy by one more term in the Fourier series**, so the error in the trapezoidal rule should inherit the same spectral accuracy as in the Fourier series itself. This fact provides the reason for the error behavior we observed before - the trapezoidal rule behaves nicely on terms of Fourier series.

2.9 The discrete cosine transform

(Adapted from Numerical Recipes, 3rd edition, 12.4.2). Now suppose $f(x)$ is a real function in $[0, \pi]$ and consider the points

$$x_j = \pi j/N, \quad j = 0, \cdots, N-1.$$

We can define a transform using these points by creating an **extension** of the data f_j to the points x_j for $j = N, \cdots, 2N-1$, effectively extending f from $[0, \pi]$ to $[\pi, 2\pi]$. Then, take the DFT and cut off the extra half to get a new transform.

One option is to define an **even extension**

$$f_{2N-j} = f_j, \quad j = 0, \cdots N-1.$$

equivalent to setting $f(x) = f(2\pi - x)$ (an even reflection around $x = \pi$). We then find that the DFT, in terms of the values known in $[0, \pi]$, is

$$F_k = \frac{1}{2}(f_0 + (-1)^k f_N) + \sum_{k=0}^{N-1} f_j \cos(\pi jk/N), \quad k = 0, \cdots, N-1$$

plus another set of values that are discarded. The above is version 1 of the **discrete cosine transform**. It has the benefit that all the pieces are real functions.

To invert, we take the IDFT; one can show that the DCT is its own inverse (up to a factor of $1/N$ in front). A second version is the 'staggered' transform

$$F_k = \sum_{j=0}^{N-1} f_j \cos kx_j, \quad f_j = \frac{2}{N} \sum_{k=0}^{N-1} F_k \cos kx_j$$

where

$$x_j = \pi(j + 1/2)/N, \ j = 0, \cdots, N-1$$

are the Chebyshev nodes. Here the trick is to extend by even reflection across the $N - 1/2$ point rather than $x_N = \pi$.

3 Derivatives, pseudo-spectral methods

3.1 Differentiation matrices, (pseudo)-spectral differentiation

Suppose we have nodes $\{x_j\}$ (for $j = 0, \cdots, n$) in an interval $[a, b]$, data $\{f_j\}$ from a real function $f(x)$ and wish to construct a 'global' approximation to the derivative $f(x)$ in the interval (or at least at the nodes).

One approach is to compute the Lagrange interpolant and then differentiate:

$$p_n(x) = \sum_{k=0}^{N} f_k \ell_k(x)$$

$$f'(x_j) \approx p_n'(x_j) = \sum_{k=0}^{N} f_k \ell_k'(x_j).$$

Observe that the formula takes the form

$$\vec{f'} = D\vec{f}$$

where $\vec{f'}$ denotes the vector of $f'(x_j)$'s and D is the **differentiation matrix** with

$$D_{ij} = \ell_k'(x_j).$$

We know that for equally spaced points $x_j = a + jh$, interpolation can be problematic, so this approach is not recommended. One could instead use local approximations at each node. Using central differences, for instance,

$$f'(x_j) \approx \frac{1}{2h}(f(x_{j+1}) - f(x_{j-1})) \implies D = \frac{1}{h}\begin{bmatrix} a & b & c & \cdots & & \\ -1/2 & 0 & 1/2 & \cdots & 0 & \\ 0 & \ddots & \ddots & \ddots & 0 & \\ \vdots & \ddots & \ddots & \ddots & \vdots & \\ 0 & \cdots & \cdots & -1/2 & 0 & 1/2 \\ \cdots & \cdots & \cdots & d & e & f \end{bmatrix}$$

where the boundary rows must be suitably modified.

However, with a better chocie of nodes, the 'Lagrange interpolant' approach can work. The Chebyshev nodes work here. There are two variants (both in $[-1, 1]$):

$$\text{zeros: } x_j = -\cos((j + 1/2)\pi/N),\ j = 0, \cdots, N-1$$

$$\text{extrema: } x_j = -\cos(j\pi/N),\ j = 0, \cdots, N.$$

Remarkably, the differentiation matrix D can be evaluated in closed form for the Chebyshev extrema. There is a deeper connection, however, discussed in the next section.

If $f(x)$ is periodic in $[0, 2\pi]$, we can use the DFT to compute its derivative. Let $\{x_j\}$ be the usual points and suppose we want to compute $\vec{f'} = (f'_0, \cdots, f'_{N-1})$.

Observe that for a Fourier series,

$$f(x) = \sum c_k e^{ikx} \implies f'(x) = \sum ikc_k e^{ikx}.$$

Thus

$$\text{differentiation in } x \iff \text{multiplication by } ik.$$

In the transformed 'frequency domain', differentiation becomes multiplying the k-th element by ik, which is trivial to compute.

The derivative is estimated by constructing the interpolant. If $N = 2m$ is then

$$f(x) \approx \sum_{k=-(m-1)}^{m} F_k e^{ikx} \implies f'(x) \approx \sum_{k=-(m-1)}^{m} ikF_k e^{ikx}.$$

The multiplication by ik property makes Letting

$$\widehat{D} = i\,\mathrm{diag}(-(m-1), \cdots, -1, 0, 1, \cdots, m)$$

it follows that

$$\vec{f'} = \mathrm{IDFT}(\widehat{D} \cdot \mathrm{DFT}(\vec{f})).$$

Note that the DFT can be written in the form of matrix multiplication, with

$$\mathcal{F}_{jk} = \frac{1}{N} e^{-2\pi ijk/N}, \quad (\mathcal{F})^{-1}_{jk} = e^{2\pi ijk/N}.$$

It follows that differentation by trigonemtric interpolant ($\vec{f'} = Df$) is **diagonalized** by the DFT - it converts the differentiation matrix to a trivial diagonal matrix ($D = \mathcal{F}^{-1}\hat{D}\mathcal{F}$).

The property makes the DFT a powerful tool for solving differential equations when spectral accuracy is beneficial (when f is smooth and periodic, for instance). When f is not periodic, however, the method cannot be used directly!

3.2 Chebyshev polynomials vs. Fourier series

Adapted from *Finite Difference and Spectral Methods for Ordinary and Partial Differential Equations* by Lloyd Trefethen (1997, available online). See, for example, *Spectral methods in Matlab* (a later book) for further details on spectral methods.

If a function $f(x)$ is smooth and **periodic**, we can transfer it to the interval $[0, 2\pi]$ (with period 2π) and use a Fourier series. Then, we reap the benefits of spectral accuracy - the approximation error will decay exponentially.

However, suppose $f(x)$ is just some smooth function on an interval. Then the Fourier series will **not** have good convergence properties, since it is very unlikely the derivatives of f match at the endpoints.

Instead, we can map it to $[-1, 1]$ and use a Chebyshev series

$$f = \sum_{k=0}^{\infty} c_k T_k(x).$$

Recalling the defintiion of T, we have, with $x = \cos\theta$,

$$f(\cos\theta) = \sum_{k=0}^{\infty} c_k \cos k\theta \text{ for } \theta \in [0, \pi].$$

The expansion is revealed to be a Fourier (cosine) series in disguise. It follows that we may use the powerful techniques (FFT and variants) to compute the expansion.

Moreover, $f(x)$ does not have to be periodic; even if it is not, $f(\cos\theta)$ will be, and thus the Chebyshev series tends to yield exponential convergence where a Fourier series may not. This trick is the basis of **spectral** (or **pseudo-spectral**) methods.

Note also that **equally spaced interpolation** of $f(\cos\theta)$ by trigonometric functions is equivalent to interpolation at the **Chebyshev extrema**

$$x_j = -\cos(j\pi/N), \quad j = 0, \cdots, N$$

which are the minima/maxima of the Chebyshev polynomials:

$$x_j = -\cos(j\pi/N) \in [-1, 1] \iff \theta_j = j\pi/N \in [0, \pi].$$

The FFT can therefore be used to construct the interpolant. Moreover, it illustrates that the extrema can be a good choice of interpolation nodes, because it makes the (polynomial) interpolant in x,

$$f(x) = \sum_{k=0}^{N} a_k T_k(x),$$

really a Fourier series (that can have good convergence properties).

Not-quite Fourier: Note that from the least-squares theory, the coefficients that minimize the least squares error are

$$c_k = \frac{\int_{-1}^{1} T_k(x) w(x)\, dx}{\int_{-1}^{1} T_k^2(x) w(x)\, dx}, \quad w(x) = 1/\sqrt{1 - x^2}.$$

The coefficients for the Chebyshev interpolant (obtained by FFT) are not quite the same, but are close. The same is true of the Fourier series vs. the approximation constructed via the FFT - the integrals are replaced by discrete versions. For this reason, methods of this sort are called **pseudo-spectral methods**.

3.3 Pseudo-spectral derivatives

Suppose we have data at the Chebyshev extraema $(x_0, \cdots, x_N)$. To compute the derivative, we first find the associated cosine series (via the FFT or discrete cosine transform)

$$f(\cos\theta) = \sum_{k=0}^{N} a_n \cos n\theta$$

Then,

$$f'(x) = \frac{df}{d\theta}\frac{d\theta}{dx} = \sum_{k=0}^{N} \frac{n a_n}{\sin\theta} \sin n\theta.$$

Since the a_n's are known, we can just evaluate the sum to compute

$$f(x_j) = \cdots, \quad j = 1, \cdots, N-1.$$

At the endpoints $x_0 = -1$ and $x_N = 1$, the $\sin\theta$ leads to a singularity, so one has to be careful (left as an exercise).

The point here is that 'Chebyshev' methods are very similar to Fourier methods but may (a) require some transformations and (b) need to have boundaries handled properly. Details aside, the power of the Fourier transform for problems with derivatives is hopefully clear from these examples.