## Trigonometric approximation and interpolation

Suppose $f$ is a periodic function with period $T$, i.e. $f(x+T) = f(x)$ for all $x \in \mathbb{R}$.
The *Fourier series* of $f$ is defined to be

$$\frac{a_0}{2} + \sum_{k=1}^{\infty} \left\{ a_k \cos\left(\frac{k\,2\pi x}{T}\right) + b_k \sin\left(\frac{k\,2\pi x}{T}\right) \right\}$$

where the coefficients $a_0, a_1, b_1, a_2, b_2, \ldots$ are given by the definite integrals

$$\begin{aligned} a_k &= \frac{2}{T} \int_{-T/2}^{T/2} f(x) \cos\left(\frac{k\,2\pi x}{T}\right) dx \qquad \text{for } k = 0, 1, 2, 3, \ldots \\ b_k &= \frac{2}{T} \int_{-T/2}^{T/2} f(x) \sin\left(\frac{k\,2\pi x}{T}\right) dx \qquad \text{for } k = 1, 2, 3, \ldots \end{aligned}$$

Owing to the periodicity, $\displaystyle\int_{-T/2}^{T/2}$ can be replaced by integration over *any* interval of length $T$, e.g. $\displaystyle\int_0^T$.

If $f$ is a function defined on a *finite* interval of length $T$, we can construct its "periodic extension" by making an infinite number of copies of it along the real line. The extended function will have period equal to $T$, the length of the original interval, and we can define its Fourier series as above.

The *Fourier convergence theorem* says that if $f$ is a function defined on an interval of length $T$ such that $f$ and $f'$ are both *piecewise continuous*, then the Fourier series of $f$ converges to the periodic extension of $f$, except at points of jump discontinuity, where the Fourier series converges to $\frac{1}{2}[f(x+) + f(x-)]$ (the midway point between the two sides of the jump).

"Convergence" means that the graph of the $M$th partial sum of the series,

$$(\star) \qquad \frac{a_0}{2} + \sum_{k=1}^{M} \left\{ a_k \cos\left(\frac{k\,2\pi x}{T}\right) + b_k \sin\left(\frac{k\,2\pi x}{T}\right) \right\},$$

approaches the graph of $f(x)$ more and more closely as $M$ gets larger. So the theorem tells us that the Fourier series of $f$ provides a very good approximation to the function $f(x)$ as you add on more terms. In practice, however, the integrals in the formulas for $a_k$ and $b_k$ can be difficult (and slow) to calculate.

### Finite Fourier transform

We would like to find a good approximation to the function $f(x)$ of the form $(\star)$ but without having to calculate integrals. First, let us simplify $(\star)$. Write $\dfrac{2\pi}{T} = \omega$ and use Euler's formula $e^{i\theta} = \cos\theta + i\sin\theta$ to get

$$\begin{aligned} e^{ik\omega x} &= \cos(k\omega x) + i\sin(k\omega x) \\ e^{-ik\omega x} &= \cos(k\omega x) - i\sin(k\omega x) \end{aligned} \quad \Longrightarrow \quad \begin{aligned} \cos(k\omega x) &= \frac{e^{ik\omega x} + e^{-ik\omega x}}{2} \\ \sin(k\omega x) &= \frac{e^{ik\omega x} - e^{-ik\omega x}}{2i} = -i\,\frac{(e^{ik\omega x} - e^{-ik\omega x})}{2} \end{aligned}$$

So $(\star)$ becomes

$$\begin{aligned} &\frac{a_0}{2} + \sum_{k=1}^{M} \left\{ a_k \frac{(e^{ik\omega x} + e^{-ik\omega x})}{2} - b_k i \frac{(e^{ik\omega x} - e^{-ik\omega x})}{2} \right\} \\ &= \frac{a_0}{2} + \sum_{k=1}^{M} \left\{ \tfrac{1}{2}(a_k - ib_k) e^{ik\omega x} + \tfrac{1}{2}(a_k + ib_k) e^{-ik\omega x} \right\} = c_0 + \sum_{k=1}^{M} \left\{ c_k e^{ik\omega x} + c_{-k} e^{-ik\omega x} \right\} \end{aligned}$$

where $c_0 = \frac{1}{2}a_0$, $c_k = \frac{1}{2}(a_k - ib_k)$ and $c_{-k} = \frac{1}{2}(a_k + ib_k) = c_k^*$.

We can further simplify this representation by writing it as

$$(\dagger) \qquad \sum_{k=-M}^{M} c_k\, e^{ik\omega x}$$

There are $2M+1$ coefficients $c_k$ in this formula. In order for $(\dagger)$ to be an approximation of $f(x)$ on the interval $[0,T]$, we shall require that it *interpolate* $f$ at the $2M+1$ "data" points $x_j = \dfrac{jT}{2M+1}$ for $j = 0,1,2,3,\ldots,2M$. In other words, we want

$$\sum_{k=-M}^{M} c_k\, e^{ik\omega x_j} = f(x_j) \qquad \text{for } \ j = 0,1,\ldots,2M$$

Writing $N = 2M+1$ for the number of data points at which we do the interpolation, and recalling that $\omega = 2\pi/T$, the above is equivalent to

$$\sum_{k=-M}^{M} c_k\, e^{i\,2\pi jk/N} = f(x_j) \qquad \text{for } \ j = 0,1,\ldots,2M,$$

This linear system for the "$c$"s can be solved to give

$$c_k = \frac{1}{N}\sum_{j=0}^{N-1} f(x_j)\, e^{-i\,2\pi kj/N} \qquad \text{for } \ k = -M,-M+1,\ldots,-1,0,1,\ldots,M-1,M$$

Matlab's `fft` command computes the sums $\sum_{j=0}^{N-1} f(x_j)\, e^{-i\,2\pi kj/N}$, given an array of $f(x_j)$ values of length $N$. The actual coefficients $c_k$ are obtained by dividing the `fft` result by $N$.
It is important to note that `fft` outputs the coefficients in the order

$$N \times (c_0,\, c_1,\ldots,c_{M-1},\, c_M,\, c_{-M},\, c_{-M+1},\ldots,c_{-2},\, c_{-1}).$$

To get the coefficients in the "natural" order

$$N \times (c_{-M},\, c_{-M+1},\ldots,c_{-2},\, c_{-1},\, c_0,\, c_1,\ldots,c_{M-1},\, c_M),$$

use the `fftshift` command.
The method underlying `fft`, namely the "fast Fourier transform" algorithm, is one of the most famous algorithms in scientific computing. It works most efficiently when the number of interpolation points $N$ is *even*, and is particularly fast when $N$ is a power of 2. Note, however, that $(\dagger)$ has $2M+1$ terms, and $2M+1$ is always an odd number. So, in practice, `fft` calculates the coefficients in the sum

$$\sum_{k=-M}^{M-1} c_k\, e^{ik\omega x}$$

i.e. the last term in $(\dagger)$ is dropped. Now there are $2M$ coefficients and we use $N = 2M$ points (an even number) for the interpolation.

The set of coefficients $\{c_k\}$ is called the "Fourier spectrum" of the function $f$.
For $k = 0,1,2,\ldots,M-1$, the "power" of $f$ in the $k$th Fourier mode is defined to be

$$P_k = \begin{cases} |c_0|^2 & \text{for } k = 0 \\ |c_k|^2 + |c_{-k}|^2 & \text{for } k = 1,\ldots,M-1 \end{cases}$$

The set $\{P_k\}$ constitutes the "power spectrum" of $f$.