§5.4.0. Introduction to the Discrete Fourier Transform matrices F_n.

A. Polynomial interpolation and F_n.

Recall from §2.1 that finding the polynomial of degree n

$$p(x) := \sum_1^n b_j\, x^{j-1} \tag{1}$$

for the interpolation of the data points (x_i, y_i), that is satisfying

$$p(x_i) = y_i\,, \qquad 1 \le i \le n$$

is equivalent to solving the set of n equations

$$b_1 + b_2 x_i + b_3 x_i^2 + \cdots + b_n x_i^{n-1} = y_i\,, \qquad 1 \le i \le n \tag{2}$$

for the unknown coefficients b_j. The equations (2) written in matrix form read

$$V b = y \tag{3}$$

where

$$V \;=\; \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1} \\ & & \vdots & & \\ 1 & x_n & x_n^2 & \cdots & x_n^{n-1} \end{bmatrix} \tag{4}$$

is called the **Vandermonde matrix** for the n by 1 array of nodes x .

The Discrete Fourier Transform matrices F_n arise in this way from a very special choice of interpolating points x_j. Recall that the n^{th} roots of unity, that is the solutions of the equation $z^n = 1$, are the numbers

$$1, \omega_n, \omega_n^2, \ldots, \omega_n^{n-1}\,, \qquad \omega_n = e^{2\pi i/n}\,. \tag{5}$$

These are the vertices of a regular n-gon inscribed in the unit circle in the complex plane.

Definition. *The discrete Fourier Transform is the $n \times n$ Vandermonde matrix with $x_{j+1} = \omega_n^j$, $0 \le j \le n-1$. The matrix is denoted F_n and the i,j element of F_n is $\omega_n^{(i-1)(j-1)}$.*

The solution of the equation $F_n b = y$ gives a polynomial (1) which interpolates the data y at the points (5).

Consider the case $n = 4$. Then $\omega_4 = e^{2\pi i/4}$ and the the roots of unity are $1, \omega_4, \omega_4^2, \omega_4^3 = 1, i, -1, -i$. The equations (2) take the form

$$\begin{array}{ccccccccc} b_1 & + & b_2 & + & b_3 & + & b_4 & = & y_1 \\ b_1 & + & b_2 e^{2\pi i/4} & + & b_3 e^{4\pi i/4} & + & b_4 e^{6\pi i/4} & = & y_2 \\ b_1 & + & b_2 e^{4\pi i/4} & + & b_3 e^{8\pi i/4} & + & b_4 e^{12\pi i/4} & = & y_3 \\ b_1 & + & b_2 e^{6\pi i/4} & + & b_3 e^{12\pi i/4} & + & b_4 e^{18\pi i/4} & = & y_4 \end{array} \tag{6}$$

or

$$F_4\, b = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & e^{2\pi i/4} & e^{4\pi i/4} & e^{6\pi i/4} \\ 1 & e^{4\pi i/4} & e^{8\pi i/4} & e^{12\pi i/4} \\ 1 & e^{6\pi i/4} & e^{12\pi i/4} & e^{18\pi i/4} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}. \tag{7}$$

When reading these equations note that $\omega_4^4 = 1$ implies that $\omega_4^n = \omega_4^m$ whenever $m-n$ is a multiple of 4. Evaluating the exponentials, one finds

$$F_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix} . \tag{8}$$

B. Trigonometric polynomial interpolation and F_n.

The polynomial interpolation problem is to find constants b_j so that the function

$$p(x) = \sum_{k=1}^{n} b_k \phi_k(x) , \qquad \phi_k(x) := x^{k-1} ,$$

interpolates the data (x_i, y_i). In practice other basis functions ϕ_k are also important. The two most widespread occur in the finite element method and trigonometric interpolation. A trigonometric polynomial of degree $n-1$ is a linear combination of the basis functions

$$\phi_k(x) := e^{i(k-1)x} , \qquad 1 \le k \le n . \tag{9}$$

These trigonometric polynomials are the functions of the form

$$b_1 + b_2 e^{ix} + b_3 e^{2ix} + \cdots + b_n e^{i(n-1)x} .$$

This is exactly what one gets from (2) if x is replaced by e^{ix}. The basis functions and therefore the interpolating function is periodic in x with period 2π, that is

$$p(x + 2\pi) = p(x)$$

for all real numbers x. This is equivalent to the fact that the graph of p repeats itself every 2π units on the x axis.

When dealing with 2π periodic functions, it is convenient to call the variable θ. At the same time the interpolating function will be denoted by Q. A trigonometric polynomial of degree 3 is a function

$$Q(\theta) := b_0 + b_1 e^{i\theta} + b_2 e^{2i\theta} + b_3 e^{3i\theta} = \sum_{n=0}^{3} b_n \, e^{in\theta} . \tag{10}$$

where the indices have been shifted to match the conventional notation in discrete Fourier analysis where the subscripts on the coefficients and the data points run from 0 to $n-1$. The interpolation problem for $p(z) = \sum b_j z^j$ at the fourth roots of unity, that is the equations $p(\omega_4^j) = y_j$ is equivalent to the equations

$$Q(\theta_j) = y_j , \qquad (\theta_0, \theta_1, \theta_2, \theta_3) = 2\pi \left(0, 1/4, 2/4, 3/4\right).$$

Polynomial interpolation at the roots of unity is equivalent to trigonometric polynomial interpolation at equally spaced points in the interval $[0, 2\pi]$. Thus there is one and only one trigonometric polynomial $Q(\theta)$ of degree $n-1$,

$$Q(\theta) = \sum_{k=0}^{n-1} b_k \, e^{ik\theta} ,$$

solving the interpolation problem

$$Q(2\pi j/n) = y_j\,, \qquad 0 \le j \le n-1\,,$$

and the solution is given by solving the linear system

$$F_n\, b = y\,. \tag{11}$$

Equation (11) is the relation between the coefficients of the trigonometric polynomial Q and the values y at the equally spaced points $x_j = 2\pi j/n$. The problem of finding the interpolating trigonometric polynomial at equally spaced points reduces to solving this equation. It is not at all obvious, but solving this system can be performed very rapidly. The algorithm is called the Fast Fourier Transform and is the subject of §5.4.1.

The inverse of F_n is easy to compute and resembles F_n. First note that the elements of the matrix $_n$ all have modulus equal to 1, so the rows of F_n are each of length $\sqrt{n}$. Less obvious but easily verified is that distinct rows of F_n are orthogonal. This is easy to check for F_4. To verify the general case, recall that the scalar product of complex vectors has a complex conjugate in it and use the formula $1 + r + r^2 + \ldots + r^{n-1} = 0$ whenever $r = \omega_n^k$ and $k \in \mathbb{Z}$ is not a multiple of n. This shows that $(F_n)^* F_n = nI$ where F_n^* denotes the conjugate transpose of F_n which is equal to the complex conjugate of F_n since F_n is equal to its own transpose. Thus, the equation for b is solved by

$$b = \frac{1}{n}\,(F_n^*) * y \tag{12}$$

Since F_n^* is equal to the complex conjugate of F_n, $(F_n^*) * y$ is the complex conjugate of $F_n * \overline{y}$. Hence, $(F_n^*) * y$ is rapidly computable once we know how to rapidly multiply by F_n.

The algorithm in §5.4.1 shows that for n equal to a power of 2, multiplication by F_n or F_n^* can be performed in $5n \log_2 n$ operations which shows that the trigonometric interpolation can be performed very rapidly. These speeds should be compared to $O(n^3)$ for solving a general linear system in the form of equation (11), to $O(n^2)$ for multiplying a vector by an unstructured matrix, and to $O(n^2)$ for the Vandermonde related matrices from Chapter 3, where the rapid algorithms depend on the relations of Pascal triangle type among the divided differences.

C. Computing Fourier coefficients and the matrix F_N.

The matrices F_N come up in many applications where Fourier analysis plays a role. Recall that every 2π periodic function $f(\theta)$ can be represented as a linear combination of the infinite set of functions $e^{in\theta}$ where n runs over all of the integers both positive and negative. That is, there are complex coefficients a_n such that

$$f(\theta) = \sum_{n \in \mathbb{Z}} a_n\, e^{in\theta}\,. \tag{13}$$

This remarkable fact shows that all 2π periodic functions are combinations of the basic functions $\sin nx$ and $\cos nx$ from elementary geometry. More remarkable still is that the coefficients are given by a simple explicit formula

$$a_n = \frac{1}{2\pi} \int_0^{2\pi} f(\theta)\, e^{-in\theta}\, d\theta\,. \tag{14}$$

For rare functions, these integrals can be explicitly computed. More often, one evaluates formula (14) using numerical quadrature rules. Toward that end, define equally spaced mesh points $\theta_j := jh$ for $j \in \mathbb{Z}$ with $h := 2\pi/N$. The values of f at these points are denoted by $f_j := f(\theta_j)$.

Note that $\theta_{j+N} = \theta_j + 2\pi$ so that the infinite sequence of function values f_j is N-periodic in j, that is $f_{j+N} = f_j$.

Since the values of the integrand in (14) are equal at the two end points, the left hand rule, right hand rule, and trapezoid rule all give the same approximate quadrature for (14). Using the formula for the left hand quadrature rule yields

$$a_n \approx a_n^{\text{approx}} := \frac{h}{2\pi} \sum_{j=0}^{N-1} f(\theta_j)\, e^{-in\theta_j} = \frac{1}{N} \sum_{j=0}^{N-1} f_j\, e^{-2\pi i n j/N} . \tag{15}$$

Note that $a_n \to 0$ as $|n| \to \infty$, while, a_n^{approx} is N-periodic in n. That is for all n, $a_n^{\text{approx}} = a_{n+N}^{\text{approx}}$.
Equation (15) shows that the approximate Fourier coefficients $a^{\text{approx}}(0 : N-1)$ are given by

$$a^{\text{approx}}(0 : N-1) = \frac{1}{N} F_N * f(0 : N-1) . \tag{16}$$

D. Evaluating a Fourier Series at equally spaced points.

Suppose that one has exact or approximate values for the Fourier coefficients a_n of a 2π periodic function. It is reasonable to have values for $|n| \le C$ for some C since $|n|$ is essentially a frequency. In order to have an even number of coefficients, we use the a_n for $-M \le n \le M-1$. The expression (13) then furnishes an approximation to f

$$f(\theta) \approx f^{\text{approx}}(\theta) := \sum_{n=-M}^{M-1} a_n^{\text{approx}}\, e^{in\theta} . \tag{17}$$

To see what $f^{\text{approx}}(\theta)$ looks like one samples at N equally spaced points in $[0, 2\pi]$,

$$\theta_j = \frac{2\pi j}{N}, \qquad j = 0, 1, \cdots, N-1 \tag{18}$$

to find

$$f^{\text{approx}}(\theta_j) = \sum_{n=-M}^{M-1} a_n\, e^{2\pi i j n/N} = \sum_{n=-M}^{M-1} a_n\, \omega^{jn}, \qquad \omega := \omega_N = e^{2\pi i/N} . \tag{19}$$

Shifting indices to $m = n + M$ yields

$$f^{\text{approx}}(\theta_j) = \sum_{n=-M}^{M-1} a_n\, \omega^{jn} = \sum_{m=0}^{2M-1} a_{m-M}\, \omega^{j(m-M)} = \omega^{-jM} \sum_{m=0}^{2M-1} a_{m-M}\, \omega^{jm} . \tag{20}$$

Take N to be a power of 2, with N larger than M so that there are several points sampled within each of the periods of the most rapidly varying exponential in (17), say $N \ge 8M$.

The next step must be carefully implemented to avoid bugs. Define a vector $A(1 : 2N)$ so that

$$A(1 : M) = a(0 : M-1) , \qquad A(M+1 : 2M+1) = a(-M : -1) , \qquad A(2M+2 : N) = 0 . \tag{21}$$

Then

$$\sum_{m=0}^{2M-1} a_{m-M}\, \omega^{jm} = F_N * A . \tag{22}$$

Thus (20) shows that the evaluation of f reduces to multiplication of the vector A by F_N followed by componentwise multiplication by ω^{-jM},

$$f^{\text{approx}} = \left[\omega^{-jM}\right]_{j=0:N-1} \cdot \Big(F_N * A\Big)\,.$$

E. Fourier Transforms.
If one wants to use the Fourier Inversion Formulas

$$f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \hat{f}(\xi)\, e^{ix\xi}\, d\xi\,, \qquad \hat{f}(\xi) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x)\, e^{-ix\xi}\, d\xi\,,$$

evaluating the integrals with familiar quadrature rules (e.g., Composite Trapezoid or Simpson) both the computation of the Fourier Transform $\hat{f}$ and the recovery of f from $\hat{f}$ reduce to multiplications by matrices of type F and so can be performed at the remarkable rate of $O(n \log n)$ operations. Roughly speaking this means that passing from equally spaced values of f to equally spaced values of $\hat{f}$ has hardly any computational cost.

Problem 1. Let f be the 2π periodic function such that

$$f(\theta) := \begin{cases} 1 & \text{for } \theta \in [2\pi/8, 2\pi/2] \\ \\ 0 & \text{for } \theta \in [0, 2\pi] \setminus [2\pi/8, 2\pi/2]\,. \end{cases}$$

For $M = 8, 16, 32$ compute approximate values of the Fourier coefficients a_n for $-M \le n \le M-1$. Compare the approximate values with exact values computed by integrating (14) exactly. **Hints.** First use (16) to generate $a^{\text{approx}}(0 : 2M-1)$. Fast multiplication by F_N^* is done by the Matlab command **ifft**. Then use the $2M$-periodicity of a_n^{approx} to find approximate values for the desired coefficients.

Problem 2. For $M = 8, 16, 32$ use the approximate Fourier coefficients from Problem 1, to define the Fourier approximation (17). Sample this function at 512 equally spaced points in $[0, 2\pi]$ using the FFT in (20)-(22) to compute the values. Plot the values of f^{approx} against θ for $0 \le \theta \le 2\pi$ along with the exact values of f. **Discussion.** The approximation overshoots the jump discontinuity on the high side and undershoots on the low side. Both overshoots are by about 9%. This together with oscillations in the approximation near the jumps is called **Gibb's phenomenon**.

If you want to avoid the oscillations in the Fourier approximation of discontinuous functions, this can be achieved by **filtering** the high frequencies. The simplest filter is to replace a_n in (13) and a_n^{approx} in (17), by $\psi_n a_n$ and $\psi_n a_n^{\text{approx}}$, where

$$\psi_n \;:=\; \begin{cases} \frac{2}{N}\left(\frac{N}{2} - |n|\right) & \text{for } |n| \le N \\ \\ 0 & \text{for } |n| \ge N+1\,. \end{cases}$$

The filter ψ_n is equal to 1 for $n = 0$ and decreases linearly to 0 as $|n|$ increases to $N/2$. These yield the Féjer approximations

$$f(\theta) \approx f^{\text{filtered}}(\theta) := \sum_{-N/2}^{N/2} \psi_n\, a_n^{\text{approx}}\, e^{in\theta} = \sum_{0}^{N-1} \psi_{n-N/2}\; a_{n-N/2}^{\text{approx}}\, e^{i(n-N/2)\theta}\,. \tag{23}$$

Note that $\psi_{\pm N/2} = 0$ so whether the sum runs to $N/2$ or to $N/2 - 1$ makes no difference. Define

$$b_n := \begin{cases} \psi_{n-N/2}\; a^{\text{approx}}_{n-N/2} & \text{for } \; 0 \le n \le N \\ 0 & \text{otherwise}\,. \end{cases}$$

Equation (23) reads

$$f^{\text{filtered}}(\theta) = e^{-iN\theta/2} \sum_{0}^{N-1} b_n\, e^{in\theta}\,. \tag{24}$$

Problem 3. To see the suppression of Gibb's phenomenon, use the approximate Fourier coefficients of the function f from Problem 1. Sample $f^{\text{filtered}}(\theta)$ at the equally spaced points $\theta_j = 2\pi j/512$ for $j = 0 : 512$ and graph along with the graph of f for $0 \le \theta \le 2\pi$. **Hints.** You need $b(0 : 511)$. For $b(N/2 : N-1)$ use the N-periodicity of a^{approx}_n. You can use the FFT to perform the sum on the right of (24) by writing it as

$$\text{conj}\Big[\, F_{512} * \text{conj}\big[b(0:511)\big] \,\Big]\,.$$

Alternatively you can directly use the Matlab command **ifft**. **Hint.** Beware of the index shift defining b. This is a natural place for bugs.