

5. THE FINITE FOURIER TRANSFORM

We consider the approximation of a periodic function  $f$  with period  $2\pi$ , i.e.,  $f(t + 2\pi) = f(t)$ . Note that a function with a more general period can be reduced to this case in the following simple way. If  $g(t + \tau) = g(t)$ , and we set  $f(t) = g(\tau t/(2\pi))$ , then

$$f(t + 2\pi) = g(\tau(t + 2\pi)/(2\pi)) = g(\tau t/(2\pi) + \tau) = g(\tau t/(2\pi)) = f(t).$$

**5.1. Trigonometric interpolation.** We wish to approximate  $f$  by the trigonometric polynomial

$$p_n(t) = a_0 + \sum_{j=1}^n [a_j \cos(jt) + b_j \sin(jt)],$$

where we assume  $|a_n| + |b_n| \neq 0$ . Since  $p_n$  has  $2n + 1$  coefficients, we consider the interpolation problem:

Given  $0 \leq t_0 < t_1 < \dots < t_{2n} < 2\pi$ , find  $p_n(t)$  satisfying  $p_n(t_k) = f(t_k)$ ,  $k = 0, 1, \dots, 2n$ . Since  $\cos \theta = (e^{i\theta} + e^{-i\theta})/2$ ,  $\sin \theta = (e^{i\theta} - e^{-i\theta})/(2i)$ , we may rewrite the above as

$$p_n(t) = a_0 + \sum_{j=1}^n \left[ a_j \frac{e^{ijt} + e^{-ijt}}{2} + b_j \frac{e^{ijt} - e^{-ijt}}{2i} \right] = \sum_{j=-n}^n c_j e^{ijt},$$

where

$$c_0 = a_0, \quad c_j = (a_j - ib_j)/2, \quad c_{-j} = (a_j + ib_j)/2, \quad 1 \leq j \leq n.$$

Now consider the case of equally spaced points  $t_k = 2\pi k/(2n + 1)$ ,  $k = 0, 1, \dots, 2n$ . To solve the interpolation problem, we need to find  $c_j$ ,  $j = -n, \dots, n$  such that

$$\sum_{j=-n}^n c_j e^{ijt_k} = f(t_k), \quad k = 0, 1, \dots, 2n.$$

To do so, we will need the following result.

**Lemma 2.** For all integers  $m$ ,

$$\sum_{k=0}^{2n} e^{imt_k} = \begin{cases} 2n + 1, & \text{if } e^{it_m} = 1, \\ 0, & \text{if } e^{it_m} \neq 1. \end{cases}$$

*Proof.* Since  $mt_k = m2\pi k/(2n + 1) = kt_m$ , we get by the sum formula for a geometric series that

$$\sum_{k=0}^{2n} e^{imt_k} = \sum_{k=0}^{2n} e^{ikt_m} = \sum_{k=0}^{2n} [e^{it_m}]^k = \begin{cases} 2n + 1, & \text{if } e^{it_m} = 1, \\ ([e^{it_m}]^{2n+1} - 1)/(e^{it_m} - 1), & \text{if } e^{it_m} \neq 1. \end{cases}$$

But

$$[e^{it_m}]^{2n+1} = e^{(2n+1)it_m} = e^{i2\pi m} = 1,$$

so the right hand side in the second case is zero. □

We use this result in the following way. Multiply the equation

$$\sum_{j=-n}^n c_j e^{ijt_k} = f(t_k)$$

by  $e^{-ilt_k}$ , where  $-n \leq l \leq n$ , and sum from  $k = 0$  to  $2n$  to get

$$\sum_{k=0}^{2n} \sum_{j=-n}^n c_j e^{i(j-l)t_k} = \sum_{k=0}^{2n} e^{-ilt_k} f(t_k).$$

Reversing the order of summation and applying the lemma, we have

$$\begin{aligned} \sum_{k=0}^{2n} \sum_{j=-n}^n c_j e^{i(j-l)t_k} &= \sum_{j=-n}^n c_j \sum_{k=0}^{2n} e^{i(j-l)t_k} \\ &= \sum_{j=-n}^{l-1} c_j \sum_{k=0}^{2n} e^{i(j-l)t_k} + c_l \sum_{k=0}^{2n} 1 + \sum_{j=l+1}^n c_j \sum_{k=0}^{2n} e^{i(j-l)t_k} = c_l(2n+1). \end{aligned}$$

Note that since  $-n \leq j, l \leq n$ ,  $-2n \leq j-l \leq 2n$ , and hence  $|j-l|/(2n+1) < 1$ . Thus  $e^{it_{j-l}} \neq 1$  unless  $j = l$ . Replacing  $l$  by  $j$ , we conclude that

$$(5.1) \quad c_j = \frac{1}{2n+1} \sum_{k=0}^{2n} e^{-ijt_k} f(t_k), \quad j = -n, \dots, n.$$

We then recover  $p_n(t)$  by determining the  $a_j$  and  $b_j$  from  $c_j$ , i.e.,

$$a_0 = c_0, \quad a_j = c_j + c_{-j}, \quad b_j = (c_{-j} - c_j)/i = i(c_j - c_{-j}).$$

The coefficients  $\{c_{-n}, \dots, c_n\}$  are called the finite Fourier transform of the data  $f(t_0), \dots, f(t_{2n})$ . Formula (5.1) is related to the formula for the Fourier coefficients of  $f(t)$ , i.e.,

$$f(t) = \sum_{j=-\infty}^{\infty} \gamma_j e^{ijt}, \quad \gamma_j = \frac{1}{2\pi} \int_0^{2\pi} e^{-ijt} f(t) dt.$$

To see this relationship, we approximate the above integral by the composite trapezoidal rule using  $N$  subdivisions of  $[0, 2\pi]$ . If  $s_k = 2\pi k/N$ ,  $k = 0, \dots, N$ , we get

$$\begin{aligned} \gamma_j &= \frac{1}{2\pi} \int_0^{2\pi} e^{-ijt} f(t) dt = \frac{1}{2\pi} \sum_{k=0}^{N-1} \int_{s_k}^{s_{k+1}} e^{-ijt} f(t) dt \\ &\approx \frac{1}{2\pi} \sum_{k=0}^{N-1} \frac{2\pi}{N} \frac{1}{2} [e^{-ijs_k} f(s_k) + e^{-ijs_{k+1}} f(s_{k+1})] = \frac{1}{N} \sum_{k=0}^{N-1} e^{-ijs_k} f(s_k), \end{aligned}$$

where we have applied the basic trapezoidal rule  $\int_a^b f(x) dx \approx (b-a)[f(a) + f(b)]/2$  on each subinterval and have used the periodicity of  $f$  (i.e.,  $f(2\pi) = f(0)$ ) in the last step. The coefficients  $c_j$  in (5.1) correspond to the choice  $N = 2n + 1$  (for which  $s_k = t_k$ ).

**5.2. The Fast Fourier Transform (FFT).** We next consider a fast method, called the Fast Fourier Transform for computing the coefficients  $\{c_j\}$ , when the number of data points is large. To describe the FFT, we consider a more general problem, in which the data  $f$  is of length  $N$ , where  $N = 2^r$  for some integer  $r$ . Note, in formula (5.1), we considered the special case  $N = 2n + 1$ . Then letting  $w_N = e^{2\pi i/N}$ , the generalization of (5.1) becomes

$$c_j = \frac{1}{N} \sum_{k=0}^{N-1} e^{-ijk2\pi/N} f(t_k) = \frac{1}{N} \sum_{k=0}^{N-1} (w_N)^{-kj} f(t_k), \quad j = 0, 1, \dots, N-1.$$

Note that since  $(w_N)^{-k(j+N)} = (w_N)^{-kj}$ ,  $c_{j+N} = c_j$ . Thus, the range  $j = -n, \dots, n$  in (5.1) can be changed to  $j = 0, \dots, 2n$ , which in our generalized problem becomes  $j = 0, \dots, N-1$ .

To evaluate  $c_j$  if  $w_N^j$  is known requires  $N-1$  additions,  $N-1$  multiplications, and 1 division (if we use nested multiplication). For example, to calculate  $p(x) = a_3x^3 + a_2x^2 + a_1x + a_0$ , we write  $p(x)$  in the form:  $p(x) = x[x(a_3x + a_2) + a_1] + a_0$ . Then, evaluation of  $p(x)$  takes 3 additions and 3 multiplications. If we compute  $w_N^{-(l+1)}$  by  $w_N^{-1}w_N^{-l}$ , then the computation of  $w_N^j$  for  $j = 0, \dots, N$  requires  $N$  multiplications. Hence, the cost of computing all the  $c_j$ ,  $j = 0, \dots, N-1$  is  $N(N-1) + N = N^2$  multiplications and  $N(N-1)$  additions. We now present a method (FFT) that substantially reduces this cost from  $O(N^2)$  operations to  $O(N \ln_2 N)$  operations.

The basic idea of the FFT is to reduce the computation of the finite Fourier transform of a vector  $\{f_k\}$  of size  $2m$  to the transform of two vectors of size  $m$ . Let

$$\mathbf{F} = (f_0, \dots, f_{2m-1}), \quad \mathbf{F}' = (f_0, f_2, \dots, f_{2m-2}), \quad \mathbf{F}'' = (f_1, f_3, \dots, f_{2m-1}).$$

The first step is to show how to compute  $\{c_j\}$  assuming we know

$$c'_j = \frac{1}{m} \sum_{l=0}^{m-1} f_{2l} w_m^{-lj}, \quad c''_j = \frac{1}{m} \sum_{l=0}^{m-1} f_{2l+1} w_m^{-lj}, \quad j = 0, 1, \dots, m-1.$$

Now for  $\mathbf{F} = (f_0, \dots, f_{2m-1})$ ,

$$c_j = \frac{1}{2m} \sum_{k=0}^{2m-1} f_k w_{2m}^{-kj} = \frac{1}{2m} \left[ \sum_{l=0}^{m-1} f_{2l} w_{2m}^{-2lj} + \sum_{l=0}^{m-1} f_{2l+1} w_{2m}^{-(2l+1)j} \right].$$

Since  $w_m = e^{2\pi i/m} = [e^{2\pi i/(2m)}]^2 = w_{2m}^2$ , we get  $w_{2m}^{-2lj} = w_m^{-lj}$ . Hence, for  $j = 0, \dots, m-1$ ,

$$(5.2) \quad c_j = \frac{1}{2m} \left[ \sum_{l=0}^{m-1} f_{2l} w_m^{-lj} + w_{2m}^{-j} \sum_{l=0}^{m-1} f_{2l+1} w_m^{-lj} \right] = (c'_j + w_{2m}^{-j} c''_j)/2.$$

To calculate the coefficients  $c_m, \dots, c_{2m-1}$ , we use the following identities.

$$\begin{aligned} w_m^{-l(m+j)} &= w_m^{-lm} w_m^{-lj} = [e^{2\pi i/m}]^{-lm} w_m^{-lj} = e^{-2\pi il} w_m^{-lj} = w_m^{-lj}. \\ w_{2m}^{-(m+j)} &= w_{2m}^{-m} w_{2m}^{-j} = [e^{2\pi i/(2m)}]^{-m} w_{2m}^{-j} = e^{-\pi i} w_{2m}^{-j} = -w_{2m}^{-j}. \end{aligned}$$

Then using (5.2), with  $j$  replaced by  $m + j$ , (and choosing  $N = 2m$ ), we get for  $j = 0, 1, \dots, m - 1$ ,

$$c_{m+j} = \frac{1}{2m} \left[ \sum_{l=0}^{m-1} f_{2l} w_m^{-lj} - w_{2m}^{-j} \sum_{l=0}^{m-1} f_{2l+1} w_m^{-lj} \right] = (c'_j - w_{2m}^{-j} c''_j)/2.$$

Hence, if for  $j = 0, 1, \dots, m - 1$ ,  $c'_j$  and  $c''_j$  are known, the calculation of  $c_j$  for  $j = 0, 1, \dots, 2m - 1$  requires the following operations. First, the computation of  $w_{2m}^{-j}$ ,  $j = 0, 1, \dots, m$ . Starting from  $w_{2m}^{-1}$  and using the formula  $w_{2m}^{-j} = w_{2m}^{-1} w_{2m}^{-(j-1)}$ , this requires a total of  $m - 1$  multiplications. Next, the formation of  $m$  products  $w_{2m}^{-j} c''_j$ ,  $j = 0, 1, \dots, m - 1$  which requires  $m$  multiplications. Finally, we need  $m$  additions and  $m$  subtractions, a total of  $2m$  additive operations. We ignore division by 2, which is a fast operation. Thus, we see that the computation of  $\{c_j\}$ ,  $j = 0, 1, \dots, 2m - 1$  requires essentially  $2m$  multiplications and  $2m$  additions plus the evaluation of 2 finite Fourier transforms of size  $m$ . To evaluate a finite Fourier transform of size  $N = 2^r$ , we use repeated application of this idea. There will be  $r$  levels in this process, ending in the evaluation of a finite Fourier transform of size one. Hence, to calculate the finite Fourier transform of  $\{f_0, \dots, f_N\}$ , where  $N = 2^r$ , the total number of multiplications will be:

$$\begin{aligned} 2N + 2 \text{ FFT of size } N/2 &= 2N + 2 \left[ 2 \frac{N}{2} + 2 \text{ FFT of size } N/4 \right] \\ &= 2N + 2 \left[ 2 \frac{N}{2} + 2 \left( 2 \frac{N}{4} + 2 \text{ FFT of size } N/8 \right) \right] \\ &= \dots = 2N + 2 \left[ 2 \frac{N}{2} \right] + 4 \left[ 2 \frac{N}{4} \right] + \dots + 2^r \left[ 2 \frac{N}{2^r} \right] \\ &= 2N(r + 1) = 2N(\ln_2 N + 1) = O(N \ln_2 N), \end{aligned}$$

and a similar number of additions/subtractions. Thus, the number of operations in the FFT is proportional to  $N \ln_2 N$ , compared to  $N^2$ , if we do it in a naive way. So, if  $N = 1,000$ , this reduces the cost from 1,000,000 operations to 10,000.