RADIX-2 FAST FOURIER TRANSFORM

http://www.phys.uconn.edu/~rozman/Courses/m3511_19s/

Last modified: January 22, 2019

1 Discrete Fourier transform

The *discrete*, or *finite*, *Fourier transform* (DFT) of a (complex) vector $\mathbf{x}$ with N elements $(x_0, x_1, \dots, x_{N-1}) = \{x_n\}$ is another vector $\mathbf{X}$ with N elements $(X_0, X_1, \dots, X_{N-1}) = \{X_k\}$,

$$X_k = \sum_{n=0}^{N-1} x_n \, e^{-\frac{2\pi i}{N} k\, n} = \sum_{n=0}^{N-1} \omega^{k\, n} \, x_n, \tag{1}$$

where $i = \sqrt{-1}$, $k = 0, 1, \dots, N-1$, and we introduce the notation

$$\omega = e^{-\frac{2\pi i}{N}}. \tag{2}$$

The discrete Fourier transform can be expressed with matrix-vector notation:

$$\mathbf{X} = \mathrm{F}_N \, \mathbf{x}, \tag{3}$$

where the Fourier matrix F has the elements

$$\Big(F_N\Big)_{kn} = \omega^{k\, n}, \qquad k, n = 0, 1, \dots, N-1. \tag{4}$$

$$F_N = \begin{pmatrix} \omega^0 & \omega^0 & \omega^0 & \dots & \omega^0 \\ \omega^0 & \omega^1 & \omega^2 & \dots & \omega^{N-1} \\ \omega^0 & \omega^2 & \omega^4 & \dots & \omega^{2(N-1)} \\ \omega^0 & \omega^3 & \omega^6 & \dots & \omega^{3(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \omega^0 & \omega^{N-1} & \omega^{2(N-1)} & \dots & \omega^{(N-1)^2} \end{pmatrix}. \quad (5)$$

In matlab the DFT can be codes, for example, as shown in Listing 1.

```
function X = mynaivedft(x)
% MYNAIVEDFT - naive implementation of the discrete Fourier transform
    np = length(x);
    omega = exp(-2*pi*1i/np);
    n = 0:np-1;
    k = n';
    F = omega.^(k*n);
    X = F*x;
end
```

Listing 1: Naive MATLAB implementation of the discrete Fourier transform

Direct application of the definition Eq. (1) shown Listing 1 in requires N multiplications and N additions for each of the N components of $\mathbf{X}$ for a total of $2N^2$ floating-point operations. This does not include the generation of the matrix F.

2 Radix-2 algorithm

Radix-2 algorithm is a member of the family of so called Fast Fourier transform (FFT) algorithms. It computes separately the DFTs of the even-indexed inputs $(x_0, x_2, \dots, x_{N-2})$ and of the odd-indexed inputs $(x_1, x_3, \dots, x_{N-1})$, and then combines those two results to produce the DFT of the whole sequence. This idea can then be performed recursively to reduce the overall runtime from $O(N^2)$ to $O(N \log N)$. Radix-2 algorithm requires that N

is a power of two; since the number of sample points N can usually be chosen freely by the application, this is often not an important restriction.

To derive the algorithm, lets rearrange the DFT of $\mathbf{x}$, Eq. (1), into two parts: a sum over the even-numbered indices and a sum over the odd-numbered indices:

$$X_k = \sum_{m=0}^{N/2-1} x_{2m} e^{-\frac{2\pi i}{N}(2m)k} + \sum_{m=0}^{N/2-1} x_{2m+1} e^{-\frac{2\pi i}{N}(2m+1)k}. \tag{6}$$

One can factor a common multiplier $e^{-\frac{2\pi i}{N}k}$ out of the second sum.

$$X_k = \sum_{m=0}^{N/2-1} x_{2m} e^{-\frac{2\pi i}{N}(2m)k} + e^{-\frac{2\pi i}{N}k} \sum_{m=0}^{N/2-1} x_{2m+1} e^{-\frac{2\pi i}{N}(2m)k}. \tag{7}$$

The two sums in Eq. (7) are the DFT of the even-indexed part and the DFT of odd-indexed part of x_n. Denote the DFT of the even-indexed inputs by E_k and the DFT of the odd-indexed inputs by O_k and we obtain:

$$X_k = \underbrace{\sum_{m=0}^{N/2-1} x_{2m} e^{-\frac{2\pi i}{(N/2)}mk}}_{\text{DFT of even-indexed part}} + e^{-\frac{2\pi i}{N}k} \underbrace{\sum_{m=0}^{N/2-1} x_{2m+1} e^{-\frac{2\pi i}{(N/2)}mk}}_{\text{DFT of odd-indexed part}} = E_k + e^{-\frac{2\pi i}{N}k} O_k. \tag{8}$$

As the functions of k E_k and O_k are periodic with the period $N/2$:

$$E_{k+\frac{N}{2}} = E_k \tag{9}$$

and

$$O_{k+\frac{N}{2}} = O_k. \tag{10}$$

Therefore, we can rewrite Eq. (8) as

$$X_k = \begin{cases} E_k + e^{-\frac{2\pi i}{N}k} O_k & \text{for } 0 \le k < N/2, \\ E_{k-N/2} + e^{-\frac{2\pi i}{N}k} O_{k-N/2} & \text{for } N/2 \le k < N, \end{cases} \tag{11}$$

where we used the periodicity of O_k and E_k to translate the index k.

Using the following property of the *twiddle* factor $e^{-2\pi ik/N}$,

$$e^{\frac{-2\pi i}{N}(k+N/2)} = e^{\frac{-2\pi ik}{N} - \pi i} = e^{-\pi i} e^{\frac{-2\pi ik}{N}} = -e^{\frac{-2\pi ik}{N}}$$

we can rewrite X_k as:

$$\begin{aligned} X_k &= E_k + e^{-\frac{2\pi i}{N}k} O_k, \\ X_{k+\frac{N}{2}} &= E_k - e^{-\frac{2\pi i}{N}k} O_k. \end{aligned}$$

This result, expressing the DFT of length N recursively in terms of two DFTs of size $N/2$, is the core of the radix-2 fast Fourier transform.

```
function X = myradix2dft(x)
% MYRADIX2DFT radix-2 discrete Fourier transform
    np = length(x); % must be a power of two
    if np == 1
        X = x;
    else
        xe = x(1:2:end);
        xo = x(2:2:end);
        xe = myradix2dft(xe);
        xo = myradix2dft(xo);
        omega = exp(-2*pi*1i/np);
        k = (0:(np/2-1))';
        w = omega.^k;
        xo = w.*xo;
        X = [xe+xo; xe-xo];
    end
end
```

Listing 2: MATLAB implementation of radix-2 discrete Fourier transform

3 The number of floating point operations

The DFT of length N is expressed in terms of two DFTs of length $N/2$, then four DFTs of length $N/4$, then eight DFTs of length $N/8$, and so on until we reach N DFTs of length one. An DFT of length one is just the number itself. If $N = 2^p$, the number of steps in the recursion is $p = \log_2 N$. There is $O(N)$ work at each step, independent of the step number, so the total amount of work is $O(Np) = O(N \log_2 N)$.

4 Inverse DFT

The Fourier matrix F_N has the explicit inverse:

$$\left(\mathrm{F}_N^{-1}\right)_{kn} = \frac{1}{N}\,\omega^{-kn}, \qquad k,n = 0,1,\dots,N-1, \tag{12}$$

or

$$\mathrm{F}_N^{-1} = \frac{1}{N}\begin{pmatrix} \omega^0 & \omega^0 & \omega^0 & \dots & \omega^0 \\ \omega^0 & \omega^{-1} & \omega^{-2} & \dots & \omega^{-(N-1)} \\ \omega^0 & \omega^{-2} & \omega^{-4} & \dots & \omega^{-2(N-1)} \\ \omega^0 & \omega^{-3} & \omega^{-6} & \dots & \omega^{-3(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \omega^0 & \omega^{-(N-1)} & \omega^{-2(N-1)} & \dots & \omega^{-(N-1)^2} \end{pmatrix}. \tag{13}$$

Eq. (1) can be inverted as following:

$$x_k = \frac{1}{N}\sum_{n=0}^{N-1} X_n\,\omega^{-kn} = \frac{1}{N}\sum_{n=0}^{N-1} X_n\,e^{\frac{2\pi i}{N}kn}. \tag{14}$$

To prove that indeed,

$$\mathrm{F}_N\,\mathrm{F}_N^{-1} = \mathrm{F}_N^{-1}\,\mathrm{F}_N = \mathrm{I}, \tag{15}$$

where I is the identity matrix, notice that

$$1 + \omega + \omega^2 + \omega^3 + \ldots + \omega^{N-1} = 0, \tag{16}$$

since the sum on the left of Eq. (16) is an N-terms geometric progression with the start value $\omega^0 = 1$ and the common ratio ω. The value of the sum is

$$\frac{1-\omega^N}{1-\omega} = 0 \tag{17}$$

since

$$\omega^N = \left(e^{-\frac{2\pi i}{N}}\right)^N = e^{-2\pi i} = 1, \tag{18}$$

and thus the numerator in Eq. (17) is zero whereas the denominator is not.

Similarly, we can show that

$$1 + \omega^2 + \omega^4 + \omega^6 + \ldots + \omega^{2(N-1)} = 0, \tag{19}$$

$$1 + \omega^3 + \omega^6 + \omega^9 + \ldots + \omega^{3(N-1)} = 0, \tag{20}$$

and in general

$$1 + \omega^k + \omega^{2k} + \omega^{3k} + \ldots + \omega^{(N-1)k} = 0, \qquad k = 1, \ldots, N-1 \text{ and } k = -N+1, \ldots, -1. \tag{21}$$

However, when $k = 0$ the sum in Eq. (21)

$$1 + \omega^k + \omega^{2k} + \omega^{3k} + \ldots + \omega^{(N-1)k} = 1 + 1 + 1 + \ldots + 1 = N. \tag{22}$$

Summarizing Eqs. (21), (22):

$$\sum_{l=0}^{N-1} \omega^{lk} = N\,\delta_{k0} = \begin{cases} 0, & k = 1, \ldots, N-1 \text{ and } k = -N+1, \ldots, -1 \\ N, & k = 0, \end{cases} \tag{23}$$

where δ_{mn} is the Kronecker symbol.

Finally,

$$\left(\mathrm{F}_N\,\mathrm{F}_N^{-1}\right)_{kp} = \sum_{l=0}^{N-1} (\mathrm{F}_N)_{kl} \left(\mathrm{F}_N^{-1}\right)_{lp} = \frac{1}{N} \sum_{l=0}^{N-1} \omega^{kl} \omega^{-lp} = \frac{1}{N} \sum_{l=0}^{N-1} \omega^{l(k-p)} = \delta_{kp}. \tag{24}$$