ELEX 7121 Scientific Computing-2

Lecture 4, 5: Fast Fourier Transforms for discrete data

Consider a set of $2m$ data points. We previously studied the Least Square polynomial of order n and how to find the coefficients a_k , b_k . We then used orthogonality and Least Squares error minimization to to find:

$$S_n(x) = \frac{a_0}{2} + a_n \cos nx + \sum_{k=1}^{n-1} (a_k \cos kx + b_k \sin kx)$$

$$a_k = \frac{1}{m} \sum_{j=0}^{2m-1} y_j \cos kx_j \quad , \quad k = 0,1, \dots, n$$

$$b_k = \frac{1}{m} \sum_{j=0}^{2m-1} y_j \sin kx_j \quad , \quad k = 0,1, \dots, n$$

The *interpolatory* polynomial of order m on those $2m$ data points is very similar to the *LS* polynomial but requires some modifications. Since the lemma we used to find the coefficients required r not to be factor of m:

Moreover, if r is not a multiple of m, then

- $\sum_{j=0}^{2m-1} (\cos rx_j)^2 = m$ and $\sum_{j=0}^{2m-1} (\sin rx_j)^2 = m.$

For $n = m$ we need to modify the polynomial since $\sum_{j=0}^{2m-1} (\cos mx_j)^2$ will be equal to $2m$, not m.

Since $(\cos x)^2 = \frac{1}{2} + \frac{1}{2} \cos 2x$,

$$\sum_{j=0}^{2m-1} (\cos mx_j)^2 = \frac{1}{2} \sum_{j=0}^{2m-1} 1 + \frac{1}{2} \sum_{j=0}^{2m-1} \cos 2mx_j = m + \frac{1}{2} \sum_{j=0}^{2m-1} \cos 2mx_j.$$

However,

$$\cos 2mx_j = \cos 2m \left(-\pi + \frac{j}{m} \pi \right) = \cos(2j\pi - 2m\pi) = \cos(2j - 2m)\pi = (-1)^{2j-2m} = 1.$$

Thus

$$\sum_{j=0}^{2m-1} (\cos mx_j)^2 = m + \frac{1}{2} \sum_{j=0}^{2m-1} 1 = m + m = 2m.$$

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

$$a_k = \frac{1}{m} \sum_{j=0}^{2m-1} y_j \cos kx_j, \quad \text{for each } k = 0, 1, \ldots, m, \text{ and}$$

$$b_k = \frac{1}{m} \sum_{j=0}^{2m-1} y_j \sin kx_j \quad \text{for each } k = 1, 2, \ldots, m-1.$$

In practical applications we have a limit on the maximum order of Fourier series terms we can use which is referred to as Nyquist frequency. Nyquist frequency is the highest frequency FFT can detect (or include) and it is half of the sampling frequency a pictured below:

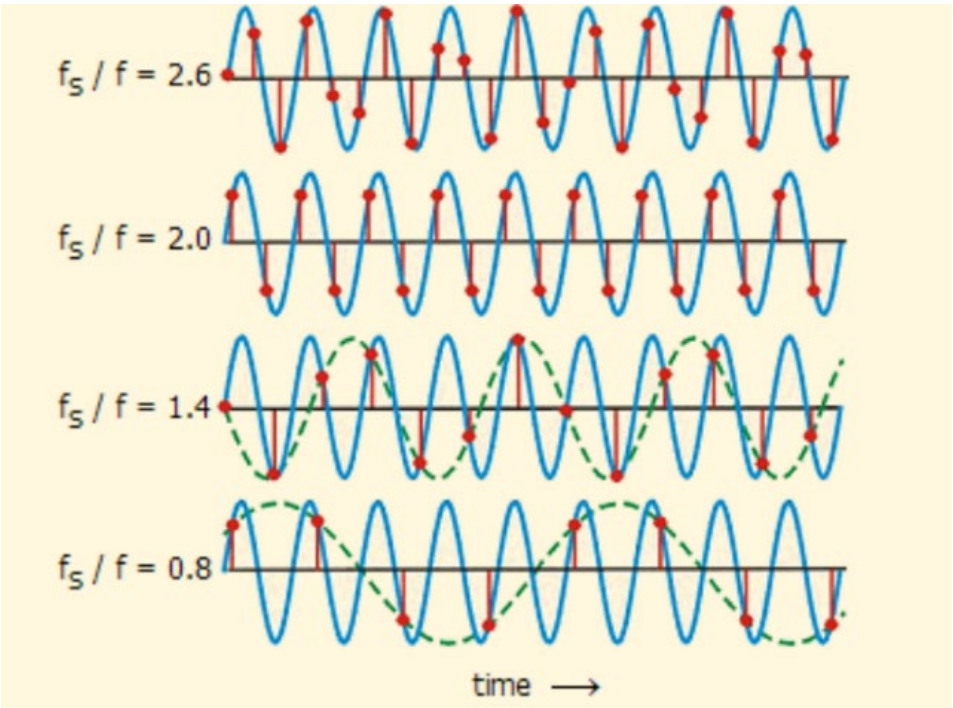


Figure 1. Sampling of a sinusoidal signal of frequency f at different sampling rates fs. With dashed lines are shown the alias frequencies, occurring when fs/f <2.

For applying FFT, instead of finding the constants a_k and b_k , we write Fourier Series in terms of exponential terms with complex coefficients, c_k , instead of sin and cos terms:

$$c_k = \sum_{j=0}^{2m-1} y_j e^{\frac{ik\pi j}{m}} = \sum_{j=0}^{2m-1} y_j e^{\frac{i\pi}{m}jk} = \sum_{j=0}^{2m-1} y_j \left(e^{\frac{i\pi}{m}}\right)^{kj}$$

Exercise 1:

$$e^{\frac{i\pi}{m}} = e^{\frac{i2\pi}{2m}} = W_N \ , \ c_k = \sum_{j=0}^{2m-1} y_j \, (W_N)^{jk} \qquad for\ k = 0, \dots N-1$$

where W_N is one of the N, N^{th} roots of one. We can write the above equation in a matrix form as:

$$\begin{bmatrix} c_0 \\ \cdot \\ \cdot \\ \cdot \\ c_{N-1} \end{bmatrix} = W \cdot \begin{bmatrix} y_0 \\ \cdot \\ \cdot \\ \cdot \\ y_{N-1} \end{bmatrix} \text{ and } \quad W_{jk} = \quad W_N{}^{(j-1)(k-1)}$$

For example, for $N = 4$: $q = W_4 = e^{\frac{i\pi}{2}} = i \quad q^0 = 1 \ , \ q^1 = i \ , q^2 = -1 \ , , q^3 = -i$ and it rotates after that. So

$$W = \begin{bmatrix} q^0 & q^0 & q^0 & q^0 \\ q^0 & q^1 & q^2 & q^3 \\ q^0 & q^2 & q^4 & q^6 \\ q^0 & q^3 & q^6 & q^9 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix}$$

$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix} \cdot \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} \quad ,$$

$$c_0 = y_0 + y_1 + y_2 + y_3 = \ (y_0 + y_2) + (y_1 + y_3)$$

$$c_1 = y_0 + iy_1 - y_2 - iy_3 = \ (y_0 - y_2) + i(y_1 - y_3)$$

$$c_2 = y_0 - y_1 + y_2 - y_3 = \ (y_0 + y_2) - (y_1 + y_3)$$

$$c_3 = y_0 - iy_1 - y_2 + iy_3 = \ (y_0 - y_2) - i(y_1 - y_3)$$

We can also use the inverse of the W matrix and recover the data values, y_j for

$$j = 0 \dots N-1 \; as:$$

$$W^{-1} = \frac{1}{N} W^* \quad \text{and} \quad y_j = \frac{1}{N}\sum_{k=0}^{2m-1} c_k \, (W_N^*)^{jk} \;\; , \;\; y_j = \frac{1}{N}\sum_{k=0}^{2m-1} c_k \left(e^{-\frac{i\pi}{m}}\right)^{jk}$$

It should be noted that the c values and the matrix W , are dependent only on y values and the size of the data set and finding them does not require any information about the time scaling or sampling frequency.

Exercise 2

Assuming $-\pi \le t \le \pi$, $t_j = -\pi + \frac{\pi}{m} j$ and Euler's formula we can find a_k and b_k as:

$$a_k + ib_k = \frac{1}{m}\sum_{j=0}^{2m-1} y_j \cos(kt_j) + i \cdot y_j \sin(kt_j) = \frac{1}{m}\sum_{j=0}^{2m-1} y_j \cos\left(k\left(-\pi + \frac{\pi}{m} j\right)\right) + i \cdot y_j \sin\left(k\left(-\pi + \frac{\pi}{m} j\right)\right)$$

$$= \frac{1}{m}\sum_{j=0}^{2m-1} y_j e^{ik\left(-\pi + \frac{\pi}{m} j\right)} = \frac{1}{m}(-1)^k c_k = A_k \;\; \rightarrow \;\; \boldsymbol{a_k = real\,(A_k)} \;\; , \;\; \boldsymbol{b_k = imag\,(A_k)}$$

Using A_k complex values, we will be able to find and recover the interpolatory trig polynomial as well.

1- Interpolatory Polynomial

Exercise 2

This polynomial interpolates the data points and is an accurate continuous fit:

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

Note that the sum is only over the first half of the C vector as the second half is complex conjugate of the first half and does not contain extra information. It also matches the **Nyquist** principle which states we can not detect frequencies higher than half of the sampling frequency in the signal. Information on sampling frequency and time axis will be used to scale the argument of sin and cos functions in the trig polynomial, so that the real time interval of $t_1 \le t \le t_2$ will be always mapped to $-\pi \le z \le \pi$ Which means: $z = -\pi + \frac{2\pi}{T}(t - t_1)$ where $T = t_2 - t_1$.

This can be easily achieved with the following `linspace` statements where we have also adjusted for the correct number of data points, $N = 2m$.

```
d=T/N
dp=2*pi/N
X = linspace(t1, t2-d,128);
Z = linspace(-pi,pi-dp,128);
```

If the sampling frequency, f_s, is given instead of $T = t_2 - t_1$, we can use $T = \frac{N}{f_s}$ and $t_1 = 0$.

2- Frequency spectrum for the given data set

Exercise 4

The sampling interval of $T = t_2 - t_1$, corresponds to the fundamental frequency of

$$f_1 = f_{k=1} = \frac{1}{T}$$

(the lowest non -zero frequency of the signal) and the higher harmonics are $f_k = kf_1$.

Frequency spectrum of a time series data is the magnitude (phase might also be included) of

A_k for $0 \le k \le m$. We should also adjust for $k = 0\ and\ k = m$ by dividing the magnitude by 2. So:

$$AF_k = abs(A_k) = \sqrt{a_k^2 + b_k^2} \quad for \quad 1 < k < m \qquad and\ AF_k = \frac{1}{2} abs(A_k)\ for\ k = 1, m$$

The frequency axis should be scaled to start from zero to represent DC component and the correct higher harmonics as:

$$k = (1{:}\, m+1) \ \rightarrow \quad ff = \frac{1}{T} * (0{:}\, m)$$

(ff is used in the exercise for the correct frequency).

Finding vector c of complex numbers:

1- **Direct method:** vector c can be found by applying direct matrix product as in:

$$c_k = \sum_{j=0}^{2m-1} y_j \, (W_N)^{jk} \quad , \quad \begin{bmatrix} c_0 \\ \cdot \\ \cdot \\ \cdot \\ c_{N-1} \end{bmatrix} = W \cdot \begin{bmatrix} y_0 \\ \cdot \\ \cdot \\ \cdot \\ y_{N-1} \end{bmatrix} \text{ and } \quad W_{jk} = \quad W_N{}^{(j-1)(k-1)}$$

Which requires $2N^2$ multiplications.

2- **Fast Fourier Transform:** This method takes advantage of the nature of W matrix to simplify the products into $\log_2 N$ stages of butterfly blocks thus reducing the number of multiplication drastically to $2N \log_2 N$. Each butterfly consists of one product and two adds as pictured below:

$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{bmatrix} = \begin{bmatrix} W_8^0 & W_8^0 & W_8^0 & W_8^0 & W_8^0 & W_8^0 & W_8^0 & W_8^0 \\ W_8^0 & W_8^1 & W_8^2 & W_8^3 & W_8^4 & W_8^5 & W_8^6 & W_8^7 \\ W_8^0 & W_8^2 & W_8^4 & W_8^6 & W_8^0 & W_8^2 & W_8^4 & W_8^6 \\ W_8^0 & W_8^3 & W_8^6 & W_8^4 & W_8^4 & W_8^7 & W_8^2 & W_8^5 \\ W_8^0 & W_8^4 & W_8^0 & W_8^4 & W_8^0 & W_8^4 & W_8^0 & W_8^0 \\ W_8^0 & W_8^5 & W_8^2 & W_8^7 & W_8^4 & W_8^4 & W_8^6 & W_8^3 \\ W_8^0 & W_8^6 & W_8^4 & W_8^2 & W_8^0 & W_8^6 & W_8^4 & W_8^2 \\ W_8^0 & W_8^7 & W_8^6 & W_8^5 & W_8^4 & W_8^3 & W_8^2 & W_8^1 \end{bmatrix} \cdot \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix}$$

$$W_8^0 = 1 \quad , \quad W_8^1 = e^{\left(\frac{i\pi}{4}\right)} = \frac{1+i}{\sqrt{2}} \, , \quad W_8^2 = i \, , \quad W_8^3 = \frac{i-1}{\sqrt{2}}$$

$$W_8^4 = -1 \quad , \quad W_8^5 = -\frac{1+i}{\sqrt{2}} \, , \quad W_8^6 = -i \, , \quad W_8^7 = -\frac{i-1}{\sqrt{2}}$$

By direct calculations the complex constants are:

$$c_0 = y_0 + y_1 + y_2 + y_3 + y_4 + y_5 + y_6 + y_7;$$

$$c_1 = y_0 + \left(\frac{i+1}{\sqrt{2}}\right) y_1 + iy_2 + \left(\frac{i-1}{\sqrt{2}}\right) y_3 - y_4 - \left(\frac{i+1}{\sqrt{2}}\right) y_5 - iy_6 - \left(\frac{i-1}{\sqrt{2}}\right) y_7;$$

$$c_2 = y_0 + iy_1 - y_2 - iy_3 + y_4 + iy_5 - y_6 - iy_7;$$

$$c_3 = y_0 + \left(\frac{i-1}{\sqrt{2}}\right) y_1 - iy_2 + \left(\frac{i+1}{\sqrt{2}}\right) y_3 - y_4 - \left(\frac{i-1}{\sqrt{2}}\right) y_5 + iy_6 - \left(\frac{i+1}{\sqrt{2}}\right) y_7;$$

$$c_4 = y_0 - y_1 + y_2 - y_3 + y_4 - y_5 + y_6 - y_7;$$

$$c_5 = y_0 - \left(\frac{i+1}{\sqrt{2}}\right) y_1 + iy_2 - \left(\frac{i-1}{\sqrt{2}}\right) y_3 - y_4 + \left(\frac{i+1}{\sqrt{2}}\right) y_5 - iy_6 + \left(\frac{i-1}{\sqrt{2}}\right) y_7;$$

$$c_6 = y_0 - iy_1 - y_2 + iy_3 + y_4 - iy_5 - y_6 + iy_7;$$

$$c_7 = y_0 - \left(\frac{i-1}{\sqrt{2}}\right) y_1 - iy_2 - \left(\frac{i+1}{\sqrt{2}}\right) y_3 - y_4 + \left(\frac{i-1}{\sqrt{2}}\right) y_5 + iy_6 + \left(\frac{i+1}{\sqrt{2}}\right) y_7.$$

Applying Fast Fourier Transform we can go backwards and define constant vectors: f, e, d to find c as follows:

Exercise 5:

Find the constants f, e, d and c based on the given formulas for an $8-$point FFT:

The f_k:

$$f_0 = y_0; \quad f_1 = y_4; \quad f_2 = iy_2; \quad f_3 = iy_6;$$

$$f_4 = \left(\frac{i-1}{\sqrt{2}}\right) y_1; \quad f_5 = \left(\frac{i-1}{\sqrt{2}}\right) y_5; \quad f_6 = -\left(\frac{i+1}{\sqrt{2}}\right) y_3; \quad f_7 = -\left(\frac{i+1}{\sqrt{2}}\right) y_7.$$

The e_k:

$$e_0 = f_0 + f_1; \quad e_1 = -i(f_2 + f_3); \quad e_2 = -\left(\frac{i-1}{\sqrt{2}}\right)(f_4 + f_5);$$

$$e_3 = -\left(\frac{i+1}{\sqrt{2}}\right)(f_6 + f_7);$$

$$e_4 = f_0 - f_1; \quad e_5 = f_2 - f_3; \quad e_6 = f_4 - f_5; \quad e_7 = f_6 - f_7.$$

The d_k:

$$d_0 = e_0 + e_1; \quad d_1 = -i(e_2 + e_3); \quad d_2 = e_4 + e_5; \quad d_3 = -i(e_6 + e_7);$$

$$d_4 = e_0 - e_1; \quad d_5 = e_2 - e_3; \quad d_6 = e_4 - e_5; \quad d_7 = e_6 - e_7.$$

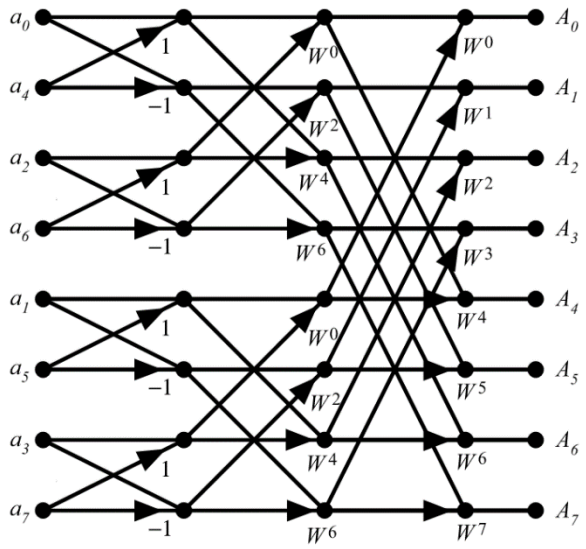
The c_k:

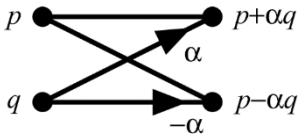
$$c_0 = d_0 + d_1; \quad c_1 = d_2 + d_3; \quad c_2 = d_4 + d_5; \quad c_3 = d_6 + d_7;$$

$$c_4 = d_0 - d_1; \quad c_5 = d_2 - d_3; \quad c_6 = d_4 - d_5; \quad c_7 = d_6 - d_7.$$

And c will be the result of an 8 points FFT on the time series y. It should match the result of the direct matrix multiplication resulting from the direct method and `fft` function in MATLAB.

FFT works based on regrouping the manipulations and creating butterflies blocks as follows:





This algorithm reduces the number of multiplications drastically.
To compute DFT of an N point sequence using direct matrix multiplication we have N^2 complex multiplies and adds which adds up to $4N^2$ real multiplies and $4N^2$ adds. The basic computational step of the FFT is a butterfly. Each butterfly computes two complex numbers of $p + \alpha q$ and of $p - \alpha q$. So it requires one complex multiply and two complex adds. This works out to 4 real multiplies and 6 real adds per butterfly. There are $N/2$ butterflies per stage, and $\log_2 N$ stages, so that means about $4(\frac{N}{2}\log_2 N)$=2 $N \log_2 N$ real multiplies and 3 $N \log_2 N$ adds for an N-point FFT.

N	$r = \log_2 N$	**BRUTE FORCE** $4N^2$	**FFT** $2N\log_2 N$	speedup
2	1	16	4	4
4	2	64	16	4
8	3	256	48	5
1,024	10	4,194,304	20,480	205
65,536	16	$1.7 \cdot 10^{10}$	$2.1 \cdot 10^6$	~10^4