Discrete Fourier Analysis

We now look at the problem where the variables in Fourier transforms have compact support which means that they are only considered over a finite range rather than the infinite range. In this case we obtained Fourier transforms that can be expressed as sums over discrete points rather then integrals.

Bandwidth Limited Transforms

We consider first the case where the frequency variable ω is restricted so that the Fourier transform is bandwidth limited. This means that

$$FT^{-1}\{f(t)\} = F(\omega), \quad F(\omega) = 0 \quad for \quad |\omega| > 2\pi W$$

making the transformation $\omega = 2\pi x$ (so that $F(2\pi x) = 0$ for $|x| > W$) then for $W_1 \geq W$

$$f(t) = \int_{-W_1}^{W_1} F(2\pi x) exp(2i\pi xt) dx$$

If we now examine the function at the discrete points

$$t = \frac{n}{2W_1}, \quad n = 0, \pm 1, \pm 2....$$

we have

$$f(\frac{n}{2W_1}) = \int_{-W_1}^{W_1} F(2\pi x) exp(\frac{i\pi nx}{W_1}) dx$$

Thus we can identify

$$c_n = \frac{1}{2W_1} f(\frac{n}{2W_1}) = \frac{1}{2W_1} \int_{-W_1}^{W_1} F(2\pi x) exp(\frac{i\pi nx}{W_1}) dx$$

as the coefficients of the exponential form of the Fourier series for $F(2\pi x)$ in $-W_1 < x < W_1$. This means that this discrete set of points determines $F(\omega)$ *completely* and, furthermore, $F(\omega)$ determines $f(t)$. Thus this set of discrete points is sufficient in principle, to determine $f(t)$ for all t.

For all this to be valid we require $W_1 > W$ so that all of the bandwidth where $F(\omega)$ is non-zero is within the interval used. Now the time interval between these discrete points is

$$\frac{n+1}{2W_1} - \frac{n}{2W_1} = \frac{1}{2W_1}$$

so that the *sampling frequecy* is

$$f_S = 2W_1$$

Since we need $W_1 \geq W$ then $f_S \geq 2W$ so that the minimum frequency is given by $f_S = 2W$. This is known as *The Nyquist frequency.* Note that is does not matter if we use a frequency greater than the Nyquist frequency, apart from the fact that it is less efficient , but if we we use a sampling frequency less than the Nyquist rate then the Fourier transform and hence $f(t)$ are not determined completely and gives rise to what is known as alliasing. (This also happens if $F(\omega)$ is only approximately zero outside the bandwidth $-W \leq x \leq W$)

If $f_S = 2W$ then we can recover an *exact* expression for $f(t)$. We have

$$f(t) = \int_{-W}^{W} F(2\pi x) exp(2i\pi xt)dx = \sum_{n=-\infty}^{\infty} c_n \int_{-W}^{W} exp(-i\frac{xn\pi}{W}) exp(i2\pi xt)dx$$

$$= \sum_{n=-\infty}^{\infty} c_n \int_{-W}^{W} exp(i\mu_n x)dx, \quad \mu_n = 2\pi t - \frac{n\pi}{W}$$

$$= 2\sum_{n=-\infty}^{\infty} c_n \int_{0}^{W} cos(\mu_n x)dx$$

$$= \sum_{n=-\infty}^{\infty} f(\frac{n}{2W}) \frac{sin(\mu_n W)}{\mu_n W}$$

$$= \sum_{n=-\infty}^{\infty} f(\frac{n}{2W}) sinc(2\pi Wt - n\pi)$$

Note that at the sample points $\frac{n}{2W}$ we have $sinc(0) = 1$ as required.

Discrete Fourier Transforms

We now use the fact that any interval of length $2W$ can be used to express the bandwidth limited function and, for convenience, we now use $0 <\leq x \leq 2W$ so that

$$f(\frac{n}{2W}) = \int_{0}^{2W} F(2\pi x) exp(i\frac{\pi nx}{W})dx$$

We can use a simple integration rule to estimate the integral with the discrete points $x_k = \frac{2Wk}{N}, \quad k = 0, 1, ...N-1$. This leads to

$$f(\frac{n}{2W}) = \frac{2W}{N}\sum_{k=0}^{N-1} F(2\pi x_k)exp(i\frac{2\pi nk}{N})$$

Writing $f_n = f(\frac{n}{2W})$ and $F_k = 2WF(2\pi x_k)$ we may write this as

$$f_n = \frac{1}{N}\sum_{k=0}^{N-1} F_k exp(\frac{i2\pi nk}{N})$$

(We note that this rule replaces the continuous curve by a sequence of rectangles and if F is a finite Fourier series then for sufficiently large N the integration rule is exact; therefore the error depends on how well we can represent F by a finite Fourier series.)

This analysis is valid for all integer n. If we now make assumptions about the time function , $f(t)$ so that it is transient function that only has non-zero values in the interval $0 \leq t \leq \frac{N-1}{2W}$ then

$$\mathbf{f_n = \frac{1}{N}\sum_{k=0}^{N-1} F_k exp(\frac{i2\pi nk}{N}), \quad n = 0...N-1}$$

This is the *Inverse Discrete Fourier Transform.* From this we can define the *Discrete Fourier Transform* :

$$\sum_{n=0}^{N-1} f_n exp(-i\frac{2\pi nm}{N}) = \frac{1}{N}\sum_{n=0}^{N-1}\sum_{k=0}^{N-1} F_k exp(i(k-m)\frac{2\pi n}{N})$$

$$= \frac{1}{N}\sum_{k=0}^{N-1} F_k\{\sum_{n=0}^{N-1} exp(i\theta_k n)\}, \quad \theta_k = \frac{2\pi}{N}(k-m)$$

If $k = m$ we have $\theta_k = 0$ and the inner sum is N otherwise the sum is

$$\frac{(1 - exp(iN\theta_k)}{(1 - exp(i\theta_k))} = 0$$

since the numerator is zero and the denominator is non-zero. Thus the only non-zero value of the inner sum is $k = m$ with the value N and consequently we obtain

$$\mathbf{F_m = \sum_{n=0}^{N-1} f_n exp(-i\frac{2\pi nm}{N}), \quad m = 0...N-1}$$

Examples

1. (a) Find the DFT (Discrete Fourier Transform) for $f_n = n^2, n = 0...3$ using $N = 4$.

   (b) Find the Inverse DFT for $F_m = (-1)^m, m = 0..1$ using $N = 2$.

2. Let F be the 14-point DFT of a length -14 real sequence f. The first 8 values of $F[k]$ are:

$$F[0] = 12, F[1] = -1 + 3i, F[2] = 3 + 4i, F[3] = 1 - 5i, F[4] = -2 + 2i$$
$$F[5] = 6 + 3i, F[6] = -2 - 3i, F[7] = 10.$$

   (a) Determine the remaining values of $F[k]$.
   Evaluate the following , *without computing the inverse DFT* and *justify your answers.*

   (b) $f[0]$

   (c) $f[7]$

   (d) $\sum_{n=0}^{13} f[n]$

   (e) $\sum_{n=0}^{13} f[n]exp(4\pi in/7)$

   (f) $\sum_{n=0}^{13} |f[n]|^2$

3. Show that the inverse Fourier transform of $A(u(\omega + a) - u(\omega - a))$ is

$$\frac{Aa}{\pi} sinc(at)$$

   and hence find the inverse Fourier transform of

   (a)

$$F(\omega) = 2(u(\omega + 1) - u(\omega - 1))$$

   and

   (b)

$$G(\omega) = F(\omega) + \frac{1}{2} exp(-2\omega i\pi)F(\omega)$$

   For both of these show that the formula for a band-limited signal:

$$\sum_{n=-\infty}^{\infty} f(\frac{n}{2W}) sinc(2\pi Wt - n\pi)$$

   can be written as a finite sum of terms. Write down these terms explicitly.

Fast Fourier Transform

Defining

$$W_N = exp(-2\pi i/N)$$

the DFT and the inverse DFT can be written as

$$F_k = \sum_{j=0}^{N-1} f_j W_N^{jk}$$

and

$$f_j = \frac{1}{N} \sum_{k=0}^{N-1} F_k W_N^{-jk}$$

respectively. Both of these require N^2 multiplications. The Fast Fourier Transform is an algorithm for computing the transforms efficiently so as to reduce the number of complex multiplications required.

Initially we consider an example where $N = N_1 N_2$ giving rise to two stages in the algorithm but in general use may be made of the factors of N and the algorithm may be extended to the case where

$$N = \prod_{j=1}^{m} N_j$$

That is N has m factors. The usual implementation of the algorithm is when N is taken to be a power of 2 and each N_j is 2. Thus in our example $N = 2^2 = 4$ The 4 values of f_j are assumed given and can be written as

$$f_j = f(a, b), \quad j = 2a + b, \quad a, b = 0..1$$

so that we are essentially using a binary code for the elements.In this notation the DFT can be written

$$F(c, d) = \sum_{a=0}^{1} \sum_{b=0}^{1} f(a, b) W_4^{(2a+b)(2c+d)}$$

$$= \sum_{b=0}^{1} W_4^{b(2c+d)} \{ \sum_{a=0}^{1} f(a, b) W_4^{4ac+2ad} \}$$

But $W_4^{4ac} = 1 \Rightarrow$

$$F(c, d) = \sum_{b=0}^{1} W_4^{b(2c+d)} \{ \sum_{a=0}^{1} f(a, b) W_4^{2ad} \}$$

Defining

$$X(b,d) = \sum_{a=0}^{1} f(ab) W^{2ad}$$

we have

$$F(c,d) = \sum_{b=0}^{1} W_4^{b(2c+d)} X(b,d)$$

These two sums replace the DFT and use 2 multiplications for each of the N values so that in this example the number of complex multiplications is $4(2+2) = 16$ and is the same as the straightforward approach which uses $N^2 = 16$ multiplications. . However if we use $N = 2^3$ then we obtain three such sums with 2N multiplications required at each stage i.e. 6N = 48 multiplications compared with the $N^2 = 64$ in the conventional approach. The process becomes more and more efficient as N increases. Explicitly for $N = 2^3$ the number of multiplications is

$$2^3(2+2+2) = 2^3(2\log_2 2^3) = N(2\log_2(N))$$

This result can be generalized to any $N = 2^m$.

Numerical Example

$$N = 4, W_4^0 = 1, W_4 = -i, W_4^2 = -1, W_4^3 = i$$

$$f(00) = 1, f(01) = 2, f(10) = 3, f(11) = 4.$$

To calculate $X(b,d)$:

1. $d = 0$,

$$X(00) = f(00)W_4^0 + f(10)W_4^0 = 1 + 3 = 4$$
$$X(1,0) = f(01)W_4^0 + f(11)W_4^0 = 2 + 4 = 6$$

2. $d = 1$

$$X(01) = f(00)W_4^0 + f(10)W_4^2 = 1 - 3 = -2$$
$$X(11) = f(01)W_4^0 + f(11)W_4^2 = 2 - 4 = -2$$

At the next stage we calculate F.

1. $d = 0$:

$$F(00) = X(00)W_4^0 + X(10)W_4^0 = 4 + 6 = 10$$
$$F(10) = X(00)W_4^0 + X(10)W_4^2 = 4 - 6 = -2$$

Note the order that the DFT elements have been calculated: (00), (10) (Not (00), (01)).

2. $d = 1$:

$$F(01) = X(01)W_4^0 + X(11)W_4 = -2 + 2i$$
$$F(11) = X(01)W_4^0 + X(11)W_4^3 = -2 + 2i$$

Note the order that the values are calculated: (01), (11).

Thus we have obtained the DFT elements in the *bit reversed order* : (00) (10) (01) (11) (rather than numerical order (00) (01) (10) (11)).

Extension to larger N

The details of the use of this algorithm for larger N are easily illustrated using $N = 8 = 2^3$. Writing

$$a = 2^2 a_2 + 2a_1 + a_0, \quad b = 2^2 b_2 + 2b_1 + b_0$$

we have

$$F(b_2 b_1 b_0) = \sum_{a_2, a_1, a_0 = 0}^{1} f(a_2 a_1 a_0) W^{ab}$$

Summing first over a_2 we have

$$F(b_2 b_1 b_0) = \sum_{a_1, a_0 = 0}^{1} W^{\hat{a}b} \sum_{a_2 = 0}^{1} f(a_2 a_1 a_0) W^{2^2 a_2 b}$$

where $\hat{a} = 2a_1 + a_0$ and

$$W^{2^2 a_2 b} = W^{2^2 a_2 b_0}$$

depending only on b_0 . (The other terms in the exponential are of the form $W^{2^3 m} = 1$). Thus we may *define* the inner sum

$$X_0(a_1 a_0 b_0) = \sum_{a_2 = 0}^{1} f(a_2 a_1 a_0) W^{2^2 a_2 b_0}$$

The binary symbols a_1, a_0, b_0 lead to $N = 2^3$ such sums each taking 2 multiplications. We then have

$$F(b_2b_1b_0) = \sum_{a_1,a_0=0}^{1} W^{(2a_1+a_0)b} X_0(a_1a_0b_0)$$

We now sum over a_1 :

$$F(b_2b_1b_0) = \sum_{a_0=0}^{1} W^{a_0b} \sum_{a_1=0}^{1} X_0(a_1a_0b_0) W^{2a_1b}$$

Now

$$W^{2a_1b} = W^{4a_1b_1+2a_1b_0}$$

The remaining factor of W is 1 as before. Thus this *term depends only on* b_1 *and* b_0 . Consequently we may *define*

$$X_1(a_0b_0b_1) = \sum_{a_1=0}^{1} X_0(a_1a_0b_0) W^{2a_1b}$$

which may also be written

$$X_1(a_0b_0b_1) = \sum_{a_1=0}^{1} X_0(a_1a_0b_0) W^{2a_1(2b_1+b_0)}$$

taking into account the fact that the last term only depends on b_1 and b_0. (Note if we code this we do not need to replace b by $2b_1 + b_0$ since this result will arise naturally).

We may now complete the calculation summing over a_0

$$F(b_2b_1b_0) = \sum_{a_0=0}^{1} W^{a_0b} X_1(a_0b_0b_1)$$

In each of the three steps in the calculation we use $2N$ multiplications so that as indicated earlier , there are $N(2\log_2 N) = 48$ multiplications compared with the $N^2 = 64$ multiplications in the straightforward algorithm. Also, in each calculation of F , we need to specify b_0 to find X_0 and then b_1 to obtain X_1 and finally do this for each b_2. Thus the DFT sequence is calculated in the bit reverse order. Thus in any program we need to adjust the order of the results at the end.

Examples

1. Given $f[n] = n(7-n), n = 0..7$, use the 8-point Fast Fourier transform to obtain the DFT.

2. Let $N = N_1 N_2 = 2.3 = 6$. Show how the FFT algorithm can be adapted for this coding the numbers 0...5 in the form

$$j = 2a + b, a = 0...2, b = 0...1., \quad k = 3c + d, c = 0...1, d = 0...2$$

   Write down the 2 steps in the algorithm explicitly .