PS403 - Digital Signal processing

7. DSP – The Discrete and Fast Fourier Transforms

Key Text:

Digital Signal Processing with Computer Applications (2nd Ed.)

Paul A Lynn and Wolfgang Fuerst, (Publisher: John Wiley & Sons, UK)

We will cover in this section

The rudiments of the DFT

An introduction to FFTs

We have also already looked at the discrete *Fourier Series* of a periodic discrete sampled signal (data set)

For such a discrete (periodic) signal x[n] we can compute the corresponding *'line' spectrum* using:

$$a_k = \frac{1}{N}\sum_{n=0}^{N} x[n]\exp\left(\frac{-j2\pi kn}{N}\right)$$

where the a_k's are the complex harmonic amplitudes from which the magnitude $|a_k|$ and phase Φ_k spectra can be obtained.

$$|a_k| = \sqrt{\mathrm{Re}(a_k)^2 + \mathrm{Im}(a_k)^2} \qquad \Phi_k = Tan^{-1}\left\{\frac{\mathrm{Im}(a_k)}{\mathrm{Re}(a_k)}\right\}$$

Note:

1. x[n] is periodic and repeats every 'N' values.
2. a_k is also periodic with period 'N' samples per cycle

We have also already looked at the continuous *Fourier Transform* of an aperiodic discrete sampled signal

For such a discrete (aperiodic) signal x[n] we can compute the corresponding 'continuum' spectrum using:

$$X(\Omega) = Na_k = \sum_{n=-\infty}^{n=\infty} x[n]\exp(-jn\Omega)$$

where $X(\Omega)$ is complex and a continuous function of 'frequency' Ω in radians (or samples/cycle). The magnitude $|X(\Omega)|$ and phase $\Phi_k(\Omega)$ spectra can be obtained from:

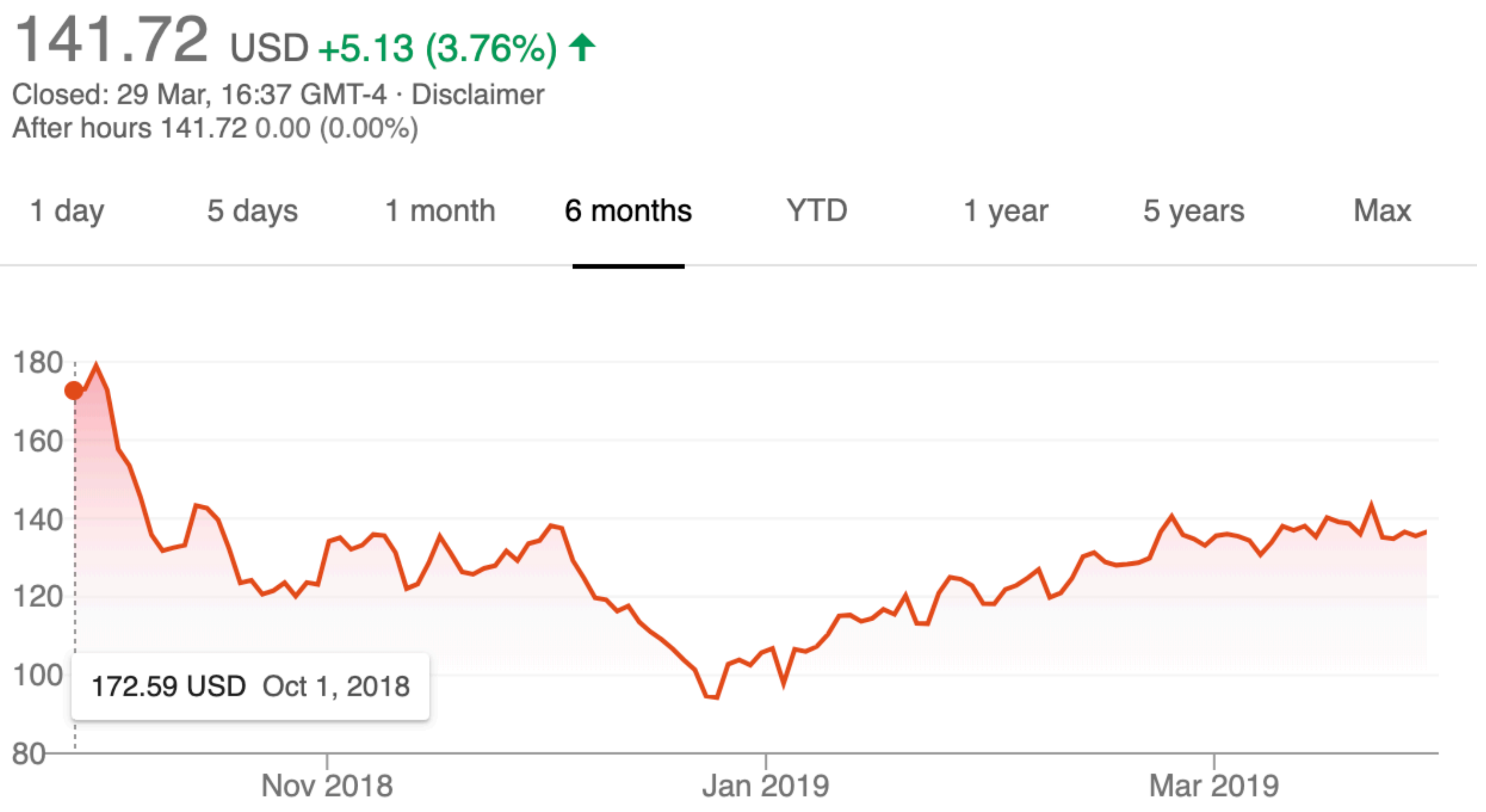
$$|X(\Omega)| = \sqrt{X(\Omega)^* X(\Omega)} \qquad \Phi_k(\Omega) = Tan^{-1}\left\{\frac{Im[X(\Omega)]}{Re[X(\Omega)]}\right\}$$

Note:

1. x[n] *is not periodic.*
2. $X(\Omega)$ *is periodic* and repeats every 2π radians !!

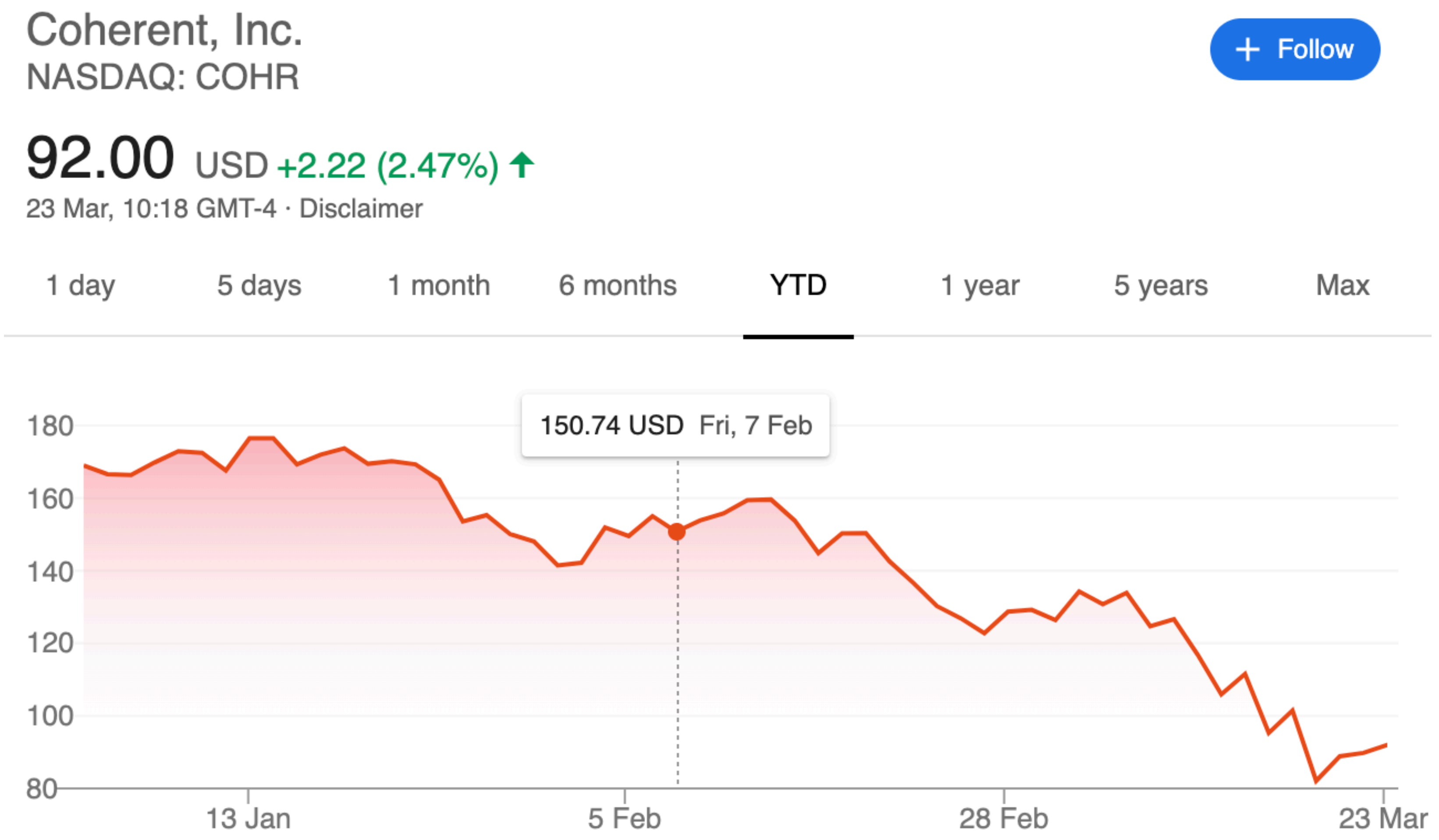
The Discrete Fourier Transform……

In general, in DSP one rarely comes across truly periodic signals. Most often one deals with a digital signal or set of data like e.g., the daily price of an equity like Coherent Inc. (NASDAQ: COHR) over say 180 days.



The Discrete Fourier Transform……

In general, in DSP one rarely comes across truly periodic signals. Most often one deals with a digital signal or set of data like e.g., the daily price of an equity like Coherent Inc. (NASDAQ: COHR) over say 180 days.


Coherent, Inc.
NASDAQ: COHR

92.00 USD +2.22 (2.47%)
23 Mar, 10:18 GMT-4 · Disclaimer

150.74 USD Fri, 7 Feb

The Discrete Fourier Transform (DFT)……

So here x[n] contains the dollar prices and the data set contains 180 values. x[n] is defined for 0 ≤ n ≤ N-1. The DFT of x[n] is defined as:

$$X[k] = \sum_{n=0}^{N-1} x[n] exp(-j2\pi k\, n/N)$$

We can write X[k] as:

$$X[k] = \sum_{n=0}^{N-1} x[n]\, W_N^{kn}$$

Where:

$$W_N^{kn} = exp(-j2\pi k\, n/N)$$

One can write the inverse DFT (IDFT) as:

$$x[n] = \frac{1}{N}\sum_{n=0}^{N-1} X[k]\, W_N^{-kn}$$

where x[n] is evaluated for 0 ≤ n ≤ N-1

The Discrete Fourier Transform (DFT)……

Looking at:

$$X[k] = \sum_{n=0}^{N-1} x[n] exp(-j2\pi k\, n/N)$$

You can see that if you try to evaluate X[k] outside the range:
0 ≤ k ≤ N-1 that X[k] (the DFT of an aperiodic signal) repeats itself. So we see that it forms a periodic spectrum just like we saw with the discrete Fourier Series of a *periodic signal*.

So the DFT considers an aperiodic signal to be periodic for the purposes of computation !!!

The Discrete Fourier Transform (DFT)……

Of course why not just use:

$$X(\Omega) = Na_k = \sum_{n=-\infty}^{n=\infty} x[n]\exp(-jn\Omega)$$

and discretise the Ω scale in the range $0 \leq \Omega \leq \pi$. But how many values of Ω should we choose ? What criterion (or criteria) should we use to obtain the optimal number of Ω values for computational economy ?

To find out consider the sampling theorem but this time transformed to the frequency domain. It will state that:

"The continuous or continuum spectrum of a signal of limited duration 'T_0' seconds may be completed represented by (or restored from) equally spaced frequency domain samples provided the samples are spaced not more than $1/T_0$ Hz apart"

The Discrete Fourier Transform (DFT)……

For ‘N’ samples the total duration of the signal then is $T_0 = NT$. Hence we require that the frequency domain samples are separated by:

$\Delta\nu = 1/NT$ (Hz) or equally $\Delta\omega = 2\pi/NT$ (radians/second).

Remembering that $\Omega = \omega T$ we have that $\Delta\Omega = \Delta\omega.T$

Hence $\Delta\Omega$ must be equal to $2\pi/N$ (radians).

In summary, if we want to discretize and compute $X(\Omega)$, the continuous FT of a discrete sampled signal (or data set) with a total of ‘N’ samples, then to satisfy the sampling theorem one must have at least ‘N’ frequency points….

One can see that the IDFT does exactly this. It returns ‘N’ values of x[n] from ‘N’ values of X[k]……..

Properties of the Discrete Fourier Transform (DFT)……

These are as for the DFS and FT of a discrete sampled signal (data set):

1. $x[n] = x[n + N]$ and $X[k] = X[k + N]$
2. Linearity
3. Time shifting: $x[n] \leftrightarrow X[k] \Rightarrow x[n - n_0] \leftrightarrow X[k].\,W_N^{kn_0}$
4. Convolution
5. Modulation
6. If $x[n]$ is symmetric about $n = 0$, then the DFT is purely real.
7. If $x[n]$ is anti-symmetric about $n = 0$, then the DFT is purely imaginary.

Rudiments of the Fast Fourier Transform (FFT)……

Work on the FFT started in the 1960s.

Continuous development since then.

The FFT covers a whole range of algorithms designed to speed up the calculation of the DFT.

If you wish to calculate the DFT of a data string of ‘N’ elements you must carry out N x N arithmetic operations. For N = 1 million, that’s 1 trillion calculations. For a 1 GHz processor and assuming 10 CPU cycles operation that’s $10^{13} / 10^{9}$ = 10,000 seconds or almost 3 hours. A 1k x 1k image would be an example. With modern FFTs and 1 GHz processor that drops to typically 0.1 – 0.5 seconds (potentially faster with GPUs).

All FFTs work on the principle of eliminating repeated calculations.

Rudiments of the Fast Fourier Transform (FFT)……

$$X[k] = \sum_{n=0}^{N-1} x[n]\, W_N^{kn}$$

which is evaluated in the range 0 ≤ k ≤ N-1

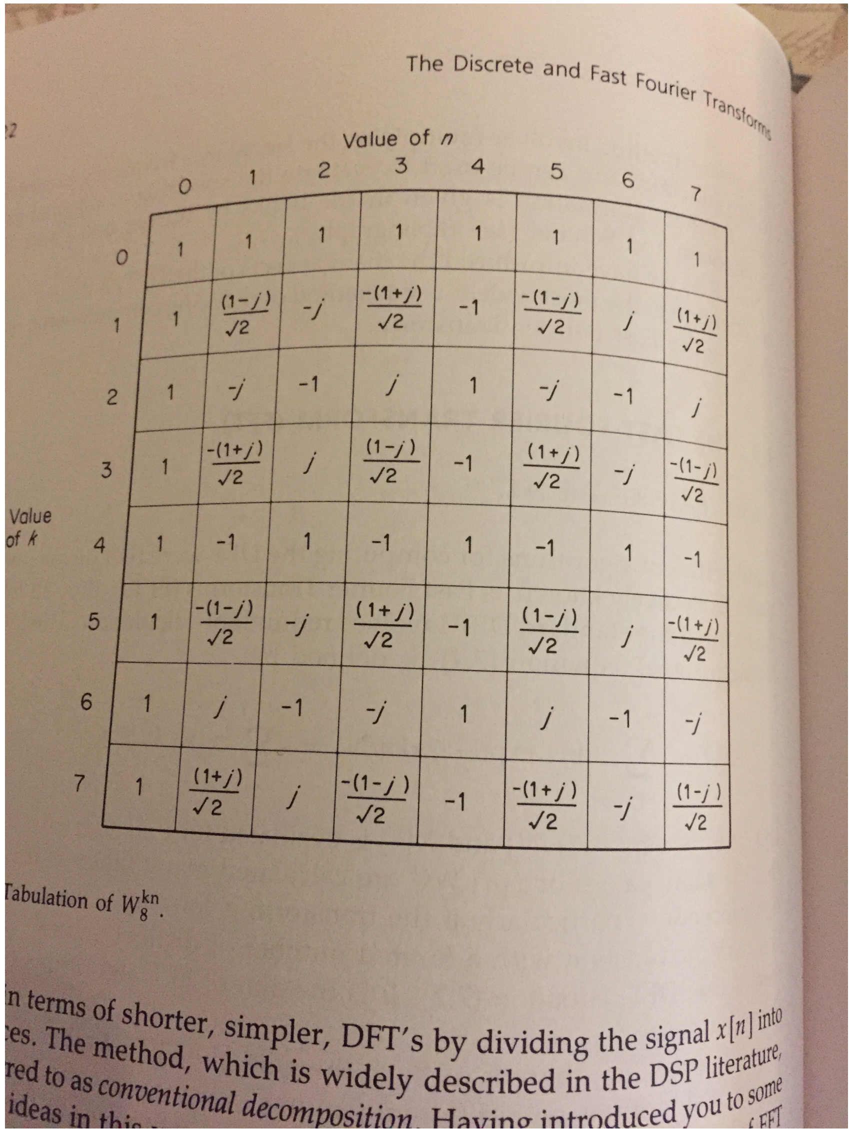
$$W_N^{kn} = exp(-j2\pi k\, n/N)$$

It turns out that the same values of $x[n].W_N^{kn}$ are evaluated many times over. FFTs try to take advantage of this point by reducing the replication of values.

Consider a compact data set of just 8 values for x[n] where ’k’ and ‘n’ each run from 0 – 7. So the product kn runs from 0 – 49. But there are only 8 unique, distinct values of W_N^{kn}.

Rudiments of the Fast Fourier Transform (FFT)……

Look at the 8 x 8 (k x n) matrix below.



Value of k \ Value of n	0	1	2	3	4	5	6	7
0	1	1	1	1	1	1	1	1
1	1	$\frac{(1-j)}{\sqrt{2}}$	$-j$	$\frac{-(1+j)}{\sqrt{2}}$	-1	$\frac{-(1-j)}{\sqrt{2}}$	j	$\frac{(1+j)}{\sqrt{2}}$
2	1	$-j$	-1	j	1	$-j$	-1	j
3	1	$\frac{-(1+j)}{\sqrt{2}}$	j	$\frac{(1-j)}{\sqrt{2}}$	-1	$\frac{(1+j)}{\sqrt{2}}$	$-j$	$\frac{-(1-j)}{\sqrt{2}}$
4	1	-1	1	-1	1	-1	1	-1
5	1	$\frac{-(1-j)}{\sqrt{2}}$	$-j$	$\frac{(1+j)}{\sqrt{2}}$	-1	$\frac{(1-j)}{\sqrt{2}}$	j	$\frac{-(1+j)}{\sqrt{2}}$
6	1	j	-1	$-j$	1	j	-1	$-j$
7	1	$\frac{(1+j)}{\sqrt{2}}$	j	$\frac{-(1-j)}{\sqrt{2}}$	-1	$\frac{-(1+j)}{\sqrt{2}}$	$-j$	$\frac{(1-j)}{\sqrt{2}}$

Tabulation of W_8^{kn}.

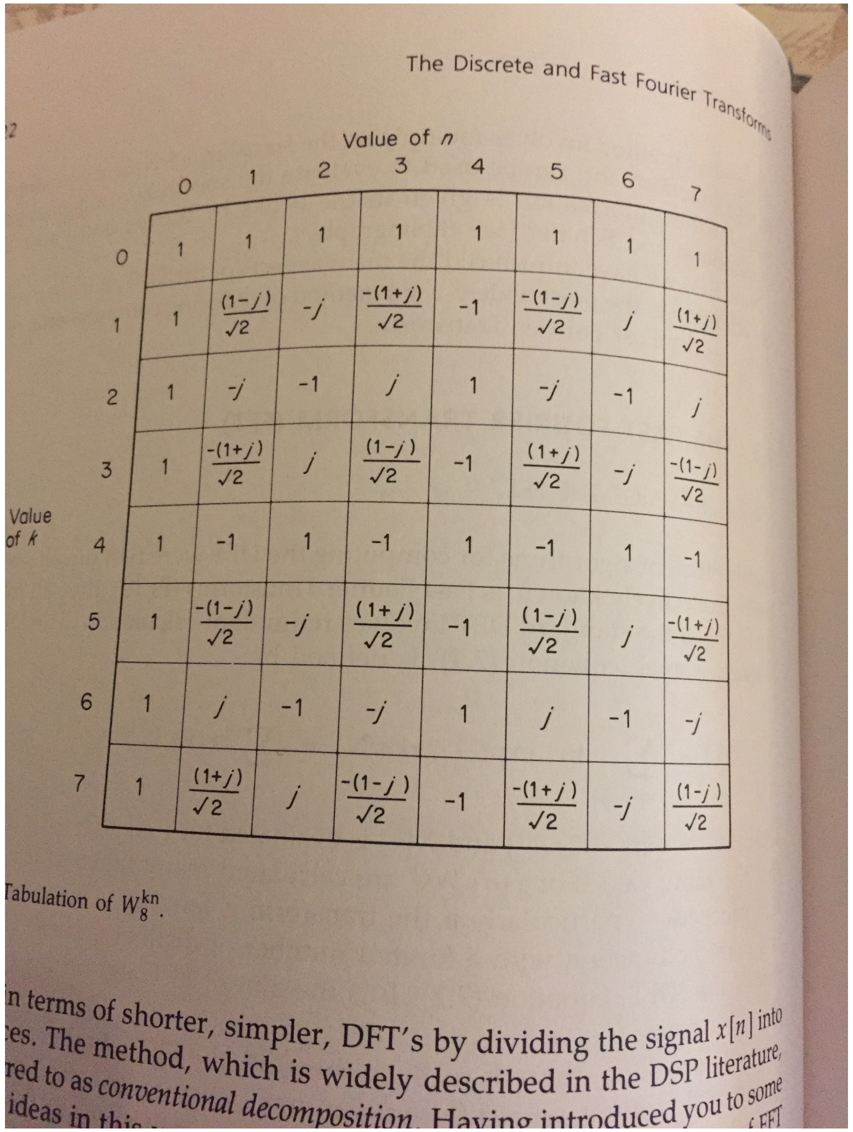
n terms of shorter, simpler, DFT's by dividing the signal $x[n]$ into
es. The method, which is widely described in the DSP literature,
red to as *conventional decomposition*. Having introduced you to some
ideas in thi

Look at this 8 x 8 (k x n) matrix. Not only are there just 8 distinct values, if you ignore the sign, there are actually only 4 unique numeric values. So 64 possible arithmetic calculations result in only 4 values (if you know which elements should be positive and which ones should be negative). So the features of periodicity and symmetry contribute to the inherent redundancy in the DFT.

All FFT algorithms exploit these features.

Rudiments of the Fast Fourier Transform (FFT)……

Look at the 8 x 8 (k x n) matrix below.



Value of n

Value of k

	0	1	2	3	4	5	6	7
0	1	1	1	1	1	1	1	1
1	1	$\frac{(1-j)}{\sqrt{2}}$	$-j$	$\frac{-(1+j)}{\sqrt{2}}$	-1	$\frac{-(1-j)}{\sqrt{2}}$	j	$\frac{(1+j)}{\sqrt{2}}$
2	1	$-j$	-1	j	1	$-j$	-1	j
3	1	$\frac{-(1+j)}{\sqrt{2}}$	j	$\frac{(1-j)}{\sqrt{2}}$	-1	$\frac{(1+j)}{\sqrt{2}}$	$-j$	$\frac{-(1-j)}{\sqrt{2}}$
4	1	-1	1	-1	1	-1	1	-1
5	1	$\frac{-(1-j)}{\sqrt{2}}$	$-j$	$\frac{(1+j)}{\sqrt{2}}$	-1	$\frac{(1-j)}{\sqrt{2}}$	j	$\frac{-(1+j)}{\sqrt{2}}$
6	1	j	-1	$-j$	1	j	-1	$-j$
7	1	$\frac{(1+j)}{\sqrt{2}}$	j	$\frac{-(1-j)}{\sqrt{2}}$	-1	$\frac{-(1+j)}{\sqrt{2}}$	$-j$	$\frac{(1-j)}{\sqrt{2}}$

Tabulation of W_8^{kn}.

n terms of shorter, simpler, DFT's by dividing the signal $x[n]$ into
es. The method, which is widely described in the DSP literature,
red to as *conventional decomposition*. Having introduced you to some
ideas in this

So-called DFT decomposition is the fundamental operation in FFT. Essentially a DFT is expressed in terms of shorter and hence simpler DFTs. There are two approaches – decimation in time and index mapping.

We will look at the former. The latter is for your next course or module on DSP

Rudiments of the Fast Fourier Transform (FFT)……

Suppose we have a ‘N’ sample long data set where $N = 2^m$.

1. Separate x[n] into two data sets of length N/2 each.
2. The first ‘subsequence’ contains only the even numbered data points in x[n].
3. The second contains only the odd numbered points in x[n].
4. We set $n = 2r$ for ‘n’ even and $n = 2r + 1$ for ‘n’ odd.

The DFT may now be recast as follows

Rudiments of the Fast Fourier Transform (FFT)……

$$X[k] = \sum_{n=0}^{N-1} x[n]\, W_N^{kn}$$

$$X[k] = \sum_{\mathrm{r}=0}^{N/2-1} x[2r]\, W_N^{2rk} + \sum_{\mathrm{r}=0}^{N/2-1} x[2r+1]\, W_N^{(2r+1)k}$$

$$X[k] = \sum_{\mathrm{r}=0}^{N/2-1} x[2r](W_N^2)^{rk} + W_N^k \sum_{\mathrm{r}=0}^{N/2-1} x[2r+1](W_N^2)^{rk}$$

Rudiments of the Fast Fourier Transform (FFT)……

Noting that:

$$(W_N^2) = \exp(-2j2\pi/N) = W_{N/2}$$

We get that:

$$X[k] = \sum_{\mathrm{r=0}}^{N/2-1} x[2r].W_{N/2}^{rk} + W_N^k \sum_{\mathrm{r=0}}^{N/2-1} x[2r+1].W_{N/2}^{rk}$$

$$X[k] = G[k] + W_N^k.H[k]$$

So we have broken the N-[point transform into two shorter N/2 point transforms G[k] and H[k]. Note the first point for the evaluation of G[k] is x[0] but for H[k] it is x[1] and hence the it is multiplied by the phase factor: W_N^k

Rudiments of the Fast Fourier Transform (FFT)……

You continue the process of decimation-in-time until you finally end up with with N/2 x 2-point DFTs to evaluate. For example:

n = {0 1 2 3 4 5 6 7} is broken down into

n = {0 2 4 6} and n = {1 3 5 7}

These two data sets are further broken down into

n = {0 4} and n = {2 6} and n = {1 5} and n = {3 7}

A 2-point DFT requires only one addition and one subtraction. However the phase factors do add complexity, especially for large ‘N’.

Rudiments of the Fast Fourier Transform (FFT)……

In general for a ‘N’ point data set, highly efficient FFT algorithms will get you down from N^2 calculations to of order N.Log(N) calculations.