THE DISCRETE AND THE FAST FOURIER TRANSFORM

J Liljencrants 84-05-09, rev 92-12-15, 97-11-27

1. THE DISCRETE FOURIER TRANSFORM, DFT

To make it possible to handle the signal in a computer we have to sample it, we must get a finite set of samples rather than the infinite number needed to represent it completely. In making Fourier analysis of this sampled signal we get the same problem once more, since the spectrum of the sampled signal is continuous, even though we can limit ourselves to the 'mother' spectrum limited by half the sampling frequency. To that end we represent the spectrum with a number of samples, as many as those of the time sequence, and define the Discrete Fourier Transform, DFT, as

$$X_k = \sum_{n=0}^{N-1} x_n \, e^{-j2\pi kn/N} ; \quad n, k = 0 \ldots N-1 , \qquad 1)$$

where x_n are the N samples in the time series (for instance N samples of a continuous signal), and X_k are corresponding samples in a frequncy series defining the complex discrete spectrum of x_n. DFT could be compared to the Fourier ,F , integral

$$S(f) = \int s(t) \, e^{-j2pft} \, dt \qquad (2)$$

with integration over a finite time interval. Our subscript k thus corresponds to the continuous frequency f, and n to time t.

To obtain one of the N frequency samples you have to accumulate the sum of all the N time samples, each multiplied by the rotation vector. This vector is somtimes called the twiddle factor and for simplicity it is notated as

$$W_N^{kn} ; \quad W_N = e^{-j2\pi/N} . \qquad (3)$$

The X_k samples, N in number, represent the spectrum of the signal between minus and plus half the sampling frequency. Since it is easier, especially when it comes to computer programming, to have the indexes to arrays positive, the usual convention is to let the negative frequencies be represented by the first half of the positive alias instead. This can be done since all the aliases are identical to the mother spectrum, something that is also seen from

$$W^{(k+N)*n} = W^{kn} * W^{Nn} = W^{kn} * e^{-j2\pi n} = W^{kn} * 1 .$$

The DFT, like the continuous F transform has some symmetry properties that are important for its application. If the time series is real, which is the case for a physical signal, then the real part of the transform has an even symmetry and the imaginary part has odd symmetry. Let us take just one sample x_n in the definition (1). Its contributions to X_{-k} and to X_k are obviously complex conjugate if x_n is real. Since this holds for all the real x_n the same holds for the sum.

Thus for a real sequence x_n:

$$Re\{x_n\} = x_n \quad Re\{X_k\} = Re\{X_{-k}\} = Re\{X_{N-k}\} \qquad even$$

$$Im\{x_n\} = 0 \quad Im\{X_k\} = -Im\{X_{-k}\} = -Im\{X_{N-k}\} \qquad odd$$

Similar relations can be found for other conditions, and we may put up the following table of symmetry rules:

$\mathrm{Re}\{x_n\}$	$\mathrm{Im}\{x_n\}$	$\mathrm{Re}\{X_k\}$	$\mathrm{Im}\{X_k\}$
x_n	0	*even*	*odd*
0	x_n	*odd*	*even*
even	*odd*	X_k	0
odd	*even*	0	X_k
even	*even*	*even*	*even*
odd	*odd*	*odd*	*odd*
even	*0*	*even*	0
odd	*0*	*odd*	0
0	*even*	0	*even*
0	*odd*	0	*odd*

It is fairly simple to show that the time series can be recovered from the frequency series by means of the Inverse DFT, IDFT, defined by

$$x_n = \frac{1}{N} \sum_{n=0}^{N-1} X_k \, e^{j2\pi kn/N} ; \qquad n, k = 0 \ldots N\text{-}1 . \tag{4}$$

In analogy to the case with the continuous transforms the IDFT differs from the DFT by a scaling factor $1/N$ and the sign of the exponent. Thus, to perform the IDFT we do not need a separate algorithm, we can instead

1. compute the DFT,
2. divide the result by N,
3. reverse the order of the samples.

Next page, fig 1a-f, shows a number of time-frequency DFT pairs on sequences of 16 points. The time function is in all cases real and consists of a single pulse. The DFT is shown with real and imaginary parts in separate graphs. The figures are quantitavely correct, but at instances a scale factor is used as indicated. A supplementary plot shows the complex DFT as a three-dimensional curve, projected at an angle to the paper.

First thing, remember again that the right end of frequency function represents the sampling rate, (the Nyquist rate), not half of this which is the highest frequency components the signal may have.

Some features of the DFT are illustrated:

- A number of the symmetries in the table above can be identified.
- When the pulse avances in time, fig 1a-e, the magnitude of the DFT is constantly the same, but the phase increases more rapidly with frequency (the delay theorem).
- When the impulse advances one step, the frequency sequence will contain exacly one more period of a cosine/sine. Ends always meet.

Of paramount importance when it comes to programming operations with the DFT is that you do not err in the exact numbering of the samples. The rectangles in the figures cover the range from sample number 0 to sample number N (here N=16). But there are only N samples, the last one at location N-1. There is no sample at the right edge. At this place would come sample number 0 of the next alias, identical to number 0 in the actual sequence. The points of symmetry are number 0 and number $N/2$ (here =8).

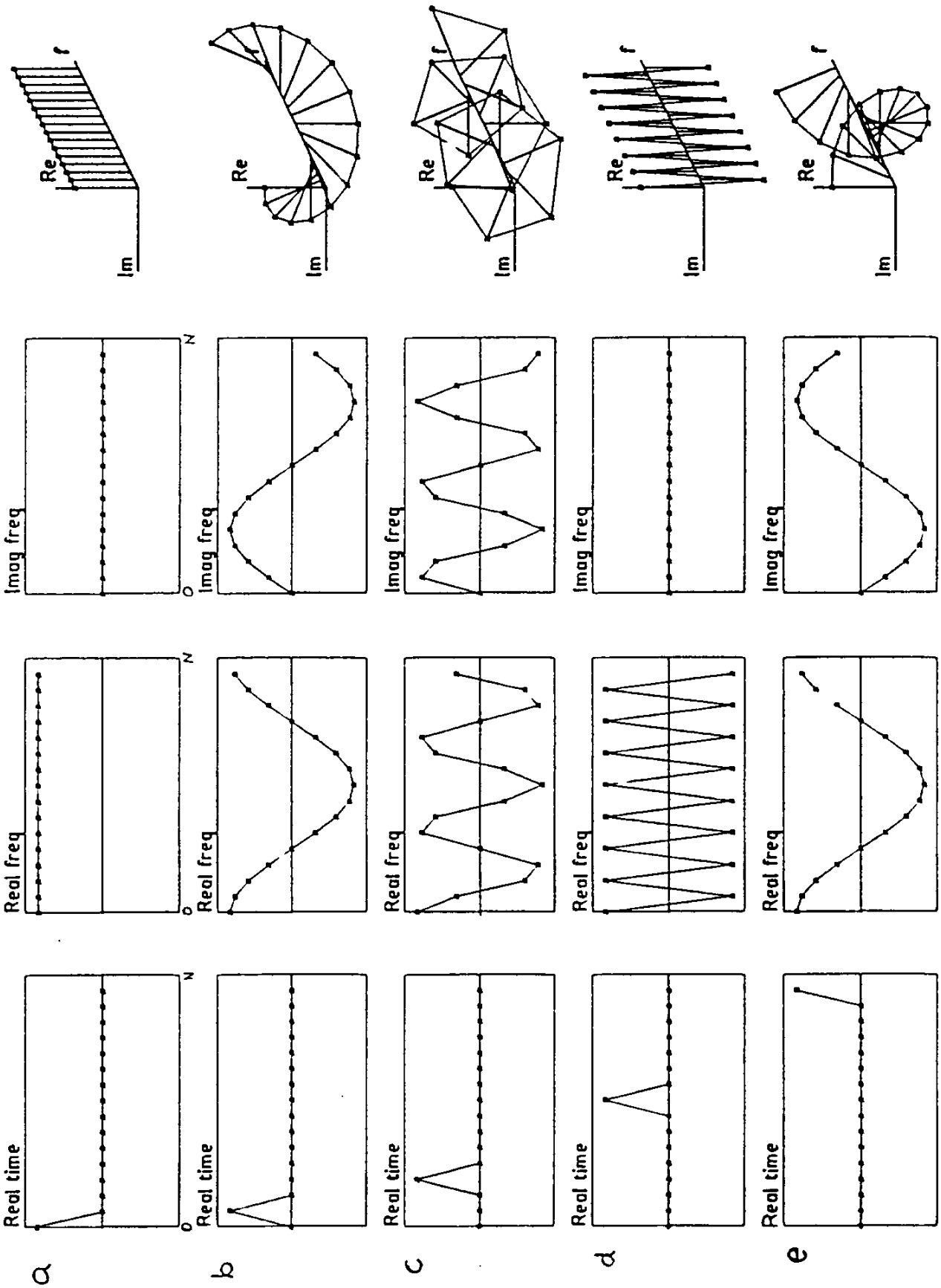


Fig 1. DFT of single impulses at different times.

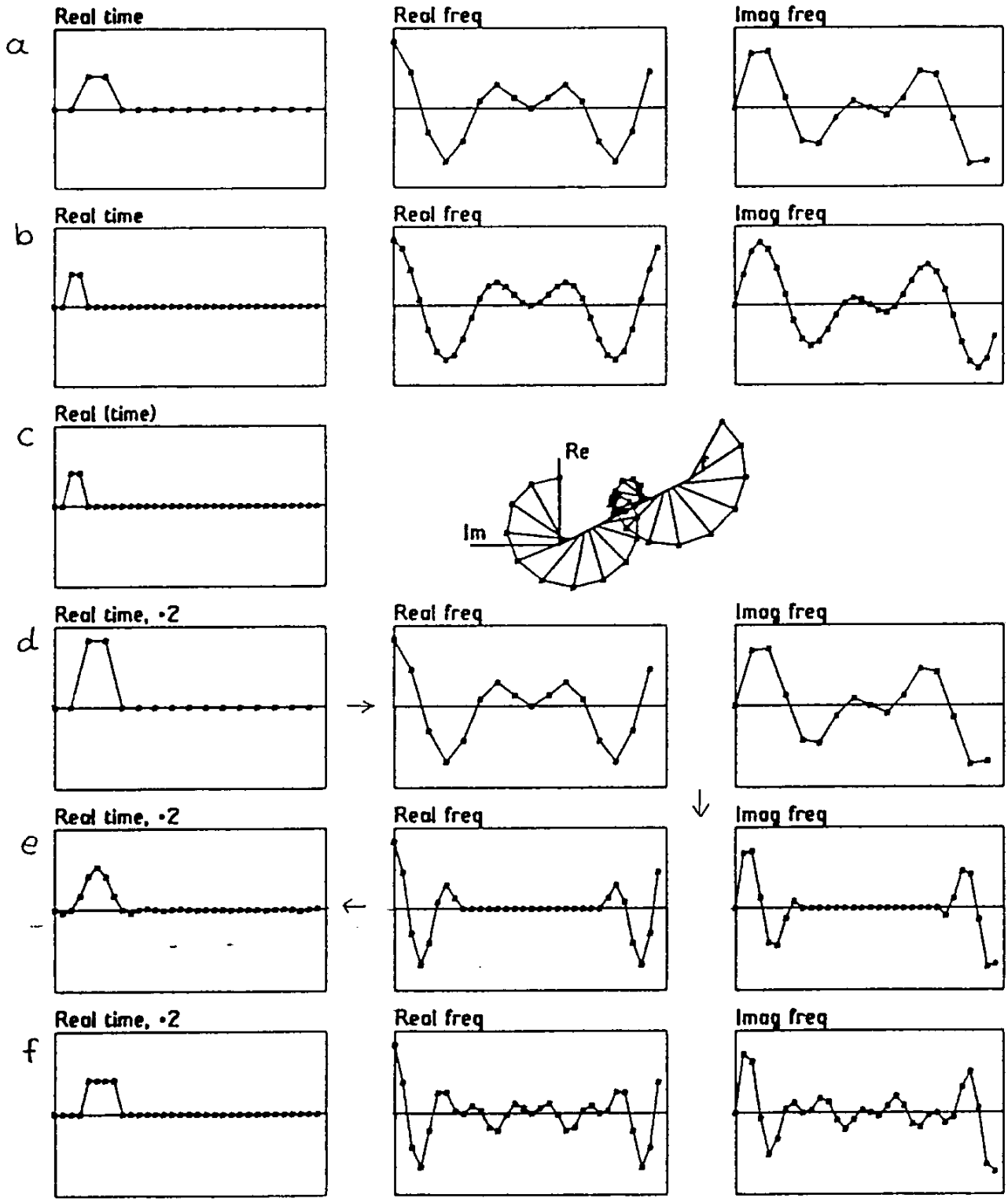


Fig 2. DFT of pulses, two samples in duration, and Fourier interpolation.

When reversing the order of samples, so as to get the IDFT from the DFT, do not touch number 0, exchange number 1 with number N-1 (=15), etc.

When extracting the odd and even components of a sequence you have

$$x_{odd,n} = (x_n - x_{N-n})/2 \; ; \quad n = 1 \ldots N/2\text{-}1$$

$$x_{odd,0} = 0.$$

$$x_{even,n} = (x_n + x_{N-n})/2; \quad n = 1 \ldots N/2\text{-}1$$

$$x_{even,0} = x_0.$$

Do not try to put n=0 in the general formula, in the computer there will be some irrelevant junk at x_N, it is outside your array.

Fig 2 shows the DFT of a pulse, two samples in duration. The spectrum is built up from two cosinoidal/sinusoidal shapes, having 2 and 3 periods along the frequency axis since input is at samples 2 and 3. Fig 2b-c shows how incrementing the number of time samples with zeroes at the end gives a smoother transform.

Fig 2d-e illustrates a related procedure called Fourier interpolation. The 16 sample time sequence is transformed. The DFT is cut in two at its middle and is spaced up into N=32 by inserting zeroes. Finally it is transformed back to the time domain with IDFT to render the Fourier interpolated result, fig 2e left. Additionally the result has to be multiplied by a factor of 2 in order to match in amplitude, rather than in amplitude*samples.

One application of this procedure is up-sampling, to increase the nominal sampling rate of a signal without adding spurious frequency components. Compare fig 2f. Here the time function is square as you would get by just duplicating the samples of fig 2d. The corresponding spectrum does have components at high frequencies.

This case can also illustrate ideal lowpass filtering: transform the fig 2f time sequence into the frequency domain by DFT, zero out high frequency components to get the fig 2e frequency sequence, and finally get lowpass filtered time sequence fig 2e by IDFT. (Spectra of 2e and 2f do not match exactly at low frequencies because of some time skew).

Another illustration of some symmetries is shown here in fig 3:

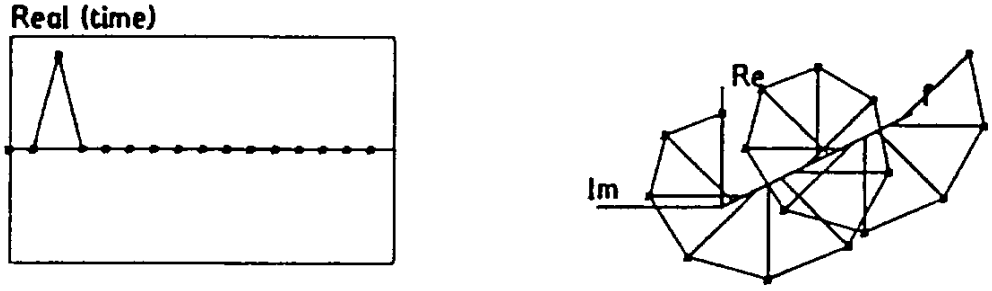


Fig 3a.

The transform of a single pulse is first shown as a spiral in the frequency domain. The spiral makes 2 turns because the pulse is at sample number 2. Adding an equal pulse in number 14 makes the time series even (and still real), and the transform is even and real. The transform is the sum of two spirals, equal in magnitude and pitch, but opposite in direction of rotation.

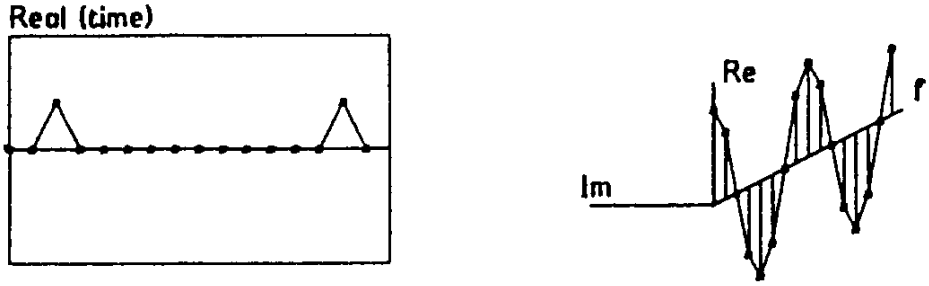


Fig 3b.

Similarly making an odd time series gives an odd and imaginary transform:

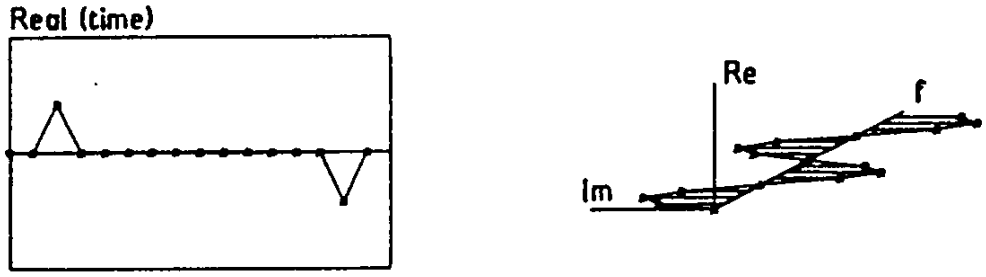


Fig 3c.

From these figures we can also comment transformation of a sinusoid assuming the right hand figure to represent time and the left hand to represent frequency. We see how the transform of a cosinusoid comes out as a pair of equal spectral lines at plus and minus the appropriate frequency. Changing the phase of the signal to make it a sinusoid shows up as a half revolution phase shift and an odd and real transform.

Now, what happens is we transform a sinusoidal signal that does not have an integer number of periods within the time interval? The spectrum should then ideally be a line at some frequency that is not represented by any sample. The continuous spectrum would have been this fictive spectral line convolved with a $\sin(x)/x$ function (the transform of our finite time window of N samples). When our sinusoidal time function does have an integer number of periods then the appropriate frequency sample coincides with the peak of the $\sin(x)/x$ shape, and all the other frequency samples come at its zero crossings. The time function with a non-integer number of periods gives a skewed sampling of the $\sin(x)/x$ shape, and all the spectrum samples are non-zero.

See fig 4 where only the lower half of the spectrum, up to half the sample rate is shown, and furthermore only the magnitude, converted into dB.

This effect is rather disastrous in making spectral analysis. The remedy is to modify the input prior to the transformation with a suitable window. The window function is then chosen such that its transform has sufficiently low sidelobes. Specially popular are the Hann window (inverted cosine period, raised to give zero values at its ends), and the Hamming window (same shape, but raised a little bit more, such that its transform sidelobes are approximately the same level everywhere, -40 dB).

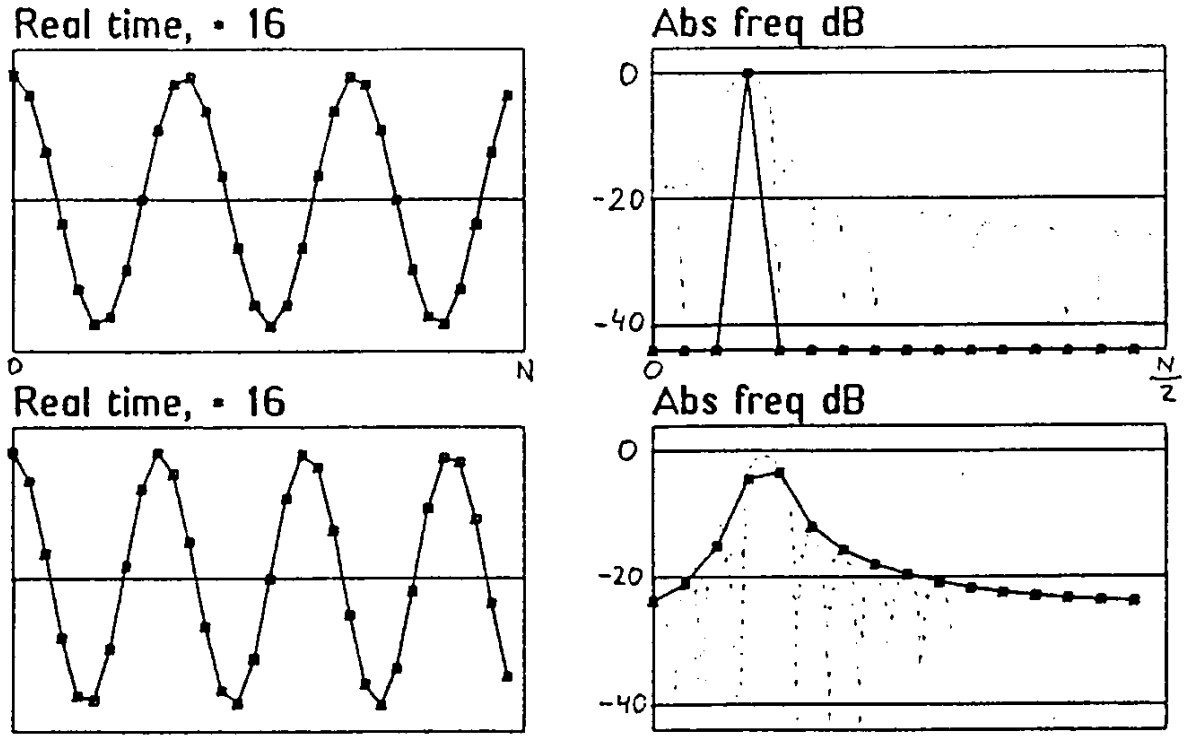


Fig 4.Transforms of 3 periods which has a clean spectral line, and of 3.5 periods which fills the entire spectrum.

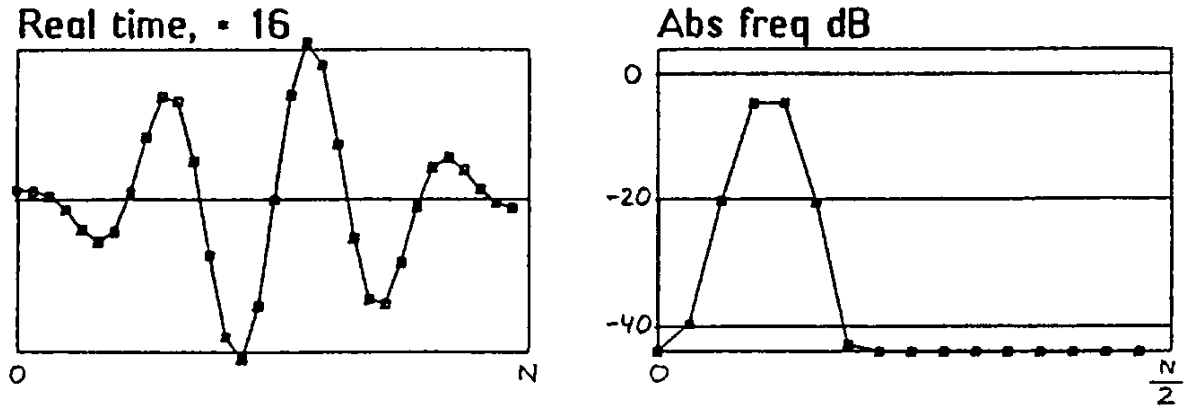


Fig 5. Transform of 3.5 periods windowed by a Hamming window such that spurious frequency components are reduced. The useful spectral peak is widened because the effective window time is smaller than the total window base time.

2. THE FAST FOURIER TRANSFORM ALGORITHM, FFT

The amount of computation to get the transform from the definition is N^2 complex multiplications and additions, an amount that may be inordinate for larger N.

FFT is an elegant algorithm that optimizes the order in which the partial computations are performed, the final result is exactly the same as from the definition (1) of the DFT. FFT stands for the feature of computational speed in making the DFT, it is not a transform in itself as is DFT. A prerequisite for FFT is that N can be resolved into factors, the greater the number of factors, the better. In practice one almost always uses $N = 2^n$. This is primarily because it makes the programming of the algorithm easy, but it is not necessary that N is a power of two.

The following example shows one way to proceed: Given a time series x_n, n=0 ..N-1. Split it up into two different sequences $x1_l$=$x2_l$ that is, with all the even numbered samples of x, and $x2_l$=$x2_l$+1 with all the odd numbered samples. These new sequences are the half as long as x, since $_l$=0 .. N/2-1.

The DFT can be computed for each of these sequences separately from the definition as

$$X1_k = \sum_{l=0}^{N/2-1} x1_l * W^{2kl}, \qquad (6)$$

$$X2_k = \sum_{l=0}^{N/2-1} x2_l * W^{2kl} .$$

It can be shown formally that the wanted result Xk can be obtained from these partial transforms as

$$X_k = X1_k + W_N^k * X2_k ; \qquad k=0 \ldots N/2-1 . \qquad (7)$$

The $X1_k$ sequence gives a direct contribution to the X_k sequence. The $X2_k$ sequence is delayed one sample interval and its contribution is thus multiplied by W_N^k. A direct application of the displacement theorem can thus be taken as a proof.

Since x_k is defined for N samples, but $x1$ and $x2$ for only half as many, some rule must be given for how to interpret the result when k=N/2 .. N-1. This is usually done as a direct periodic continuation:

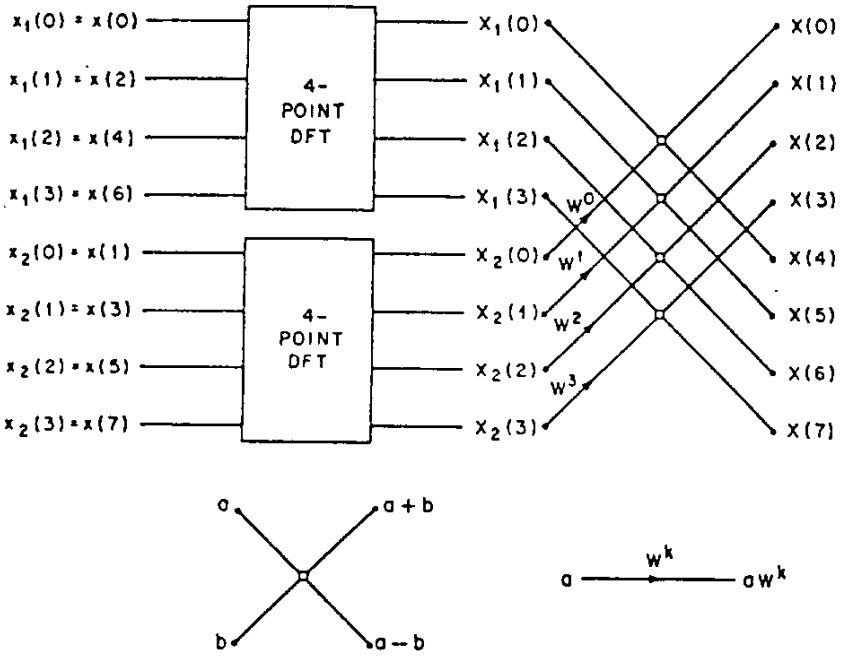


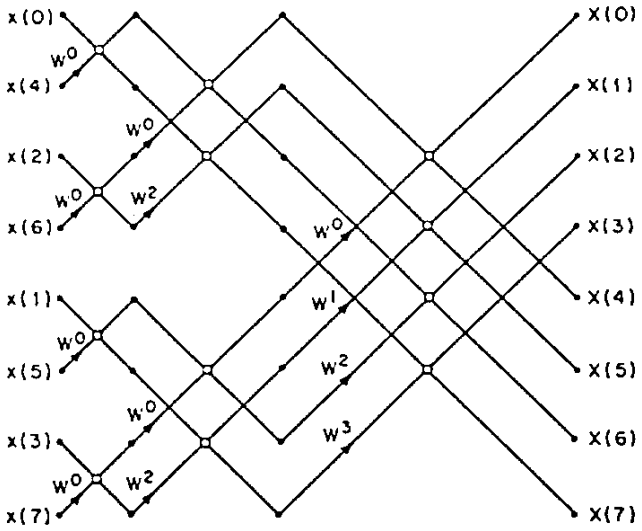
Fig 6. Construction of an eight-point DFT from two four-point DFTs.

$$X_k = X1_{k-N/2} + W_N^k * X2_{k-N/2} \; ; \qquad k = N/2 \ldots N\text{-}1 \; . \tag{8}$$

The meaningful point with this operation is that the volume of computation is reduced. Instead of making N^2 multiplications and additions as with the definition we now make $(N/2)^2$ operations for each half sequence, plus the little extra work of (7) and (8). The net gain is thus almost a factor of 2.

You can now repeat this procedure similarly to compute the two partial DFTs. These are then developed as 4 DFTs, each with only $N/4$ samples.

The principle of FFT is to continue this method of splitting the sequence into smaller sub-sequences, so called decimation, all the way possible. In the end we are to compute N transforms of sequences being only one sample each. The transform of such a one sample sequence is just the sample itself. The ultimate flow diagram will then be as in fig 7.



This procedure of splitting the time series in two with the odd and even numbered samples is called *decimation in time*. The FFT can be performed a different way, *decimation in frequency*. You then start by splitting the original sequence into two parts, one with the first $N/2$ samples, the other with the rest. Following procedures are very similar to the illustrated case.

Fig 7. Eight-point FFT obtained by successive splitting into two´s.

The total computational effort for FFT is proportional to $2N*^2\log(N)$ instead of N^2 as would be for the definition formula. For instance, with N=1024 the reduction of computation is about 50-fold. Another good point with the FFT is that the accuracy of the result is better than with the definition expression because you have a lesser number of rounding errors accumulated.

Looking at the flow diagram you see that in every stage of the computations you can take care of samples in a pair, you make the computation of an *FFT butterfly*. This is essential for the sake of memory economy because you need no tables of intermediate results. This feature is called that the operations are made *in place*. The drawback is that the original x_n sequence must be *scrambled*, the samples must be arranged in a specific order before the FFT is excecuted.

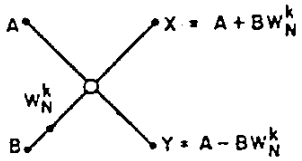


Fig 8. FFT butterfly

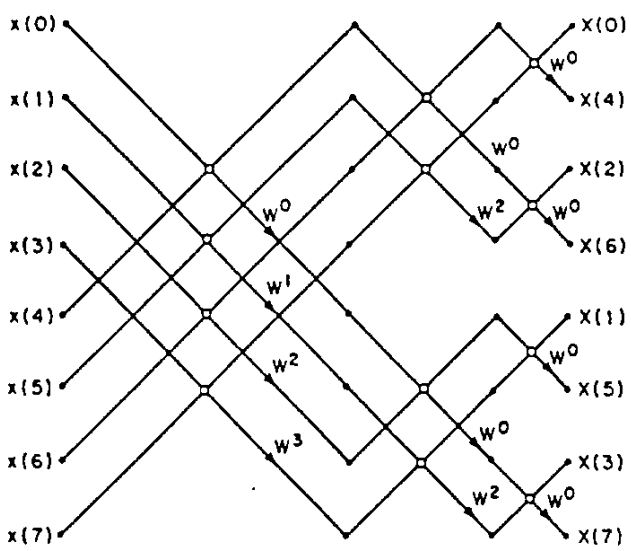


Fig 9. Complete eight-point decimation-in-frequency FFT

Alternatively decimation in frequency can be shown as a flow diagram like fig 9. Here the input sequence is ordered but the result sequence is scrambled, an unscrambling is to be made. Provided the ordinary case, that is $N=2n$, the FFT changes the input and output sequences into bit-reversed order. To find the places of the corresponding input and result samples you write the index in binary form, and reverse the order of the bits. For instance, in the $N=8$ FFT input sample number 3 has its index in binary as 011. Bit reversal gives 110, or decimal 6. Output sample number 3 is thus located at position 6 in the array. Like the FFT computations proper, also the scrambling/unscrambling operations can be done in place when you take care of samples in appropriate pairs.

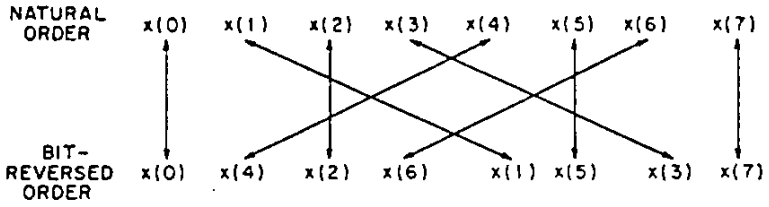


Fig 10. In-place bitreversed scrambling.

Both the result of the transform and the intermediate results are sequences of complex numbers that are represented by their real and imaginary parts. Thus also the input sequence is complex even if a physical signal would have zero imaginary part. This means that the transform is redundant in the sense that symmetrical samples are complex conjugate, as was shown in the table of symmetry relations. A consequence is that the FFT can transform two entirely independent real signals simultaneously. The two independent inputs are placed as real and imaginary parts of the input. After the FFT proper you extract the odd and even parts of the result to isolate the transforms of the two signals. These results are then single sided, the sequences extend from number 0 to $N/2-1$. The other halves representing negative frequencies have to be inferred from the symmetry rules.

FFT operations are therefore almost always complemented with some kind of shuffling operations. These may imply to isolate the odd and even parts of the transform into two sequences of half the original length, or the inverse operation, to develop an odd or even sequence twice as long as the input. Several examples on how shuffling can be applied are found in the figures of the DFT section.