Reference Material for Review of the concepts

Taken from Wikipedia.com

Fast Fourier transform (FFT) is an efficient algorithm to compute the discrete Fourier transform (DFT) and its inverse.

Let x_0, ..., x_{N-1} be complex numbers. The DFT is defined by the formula

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} nk} \qquad k = 0, \dots, N-1.$$

Since the inverse DFT is the same as the DFT, but with the opposite sign in the exponent and a $1/N$ factor, any FFT algorithm can easily be adapted for it as well.

Discrete Fourier transform (DFT) is one of the specific forms of Fourier analysis.

The sequence of N complex numbers x_0, ..., x_{N-1} is transformed into the sequence of N complex numbers X_0, ..., X_{N-1} by the DFT according to the formula:

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

where e is the base of the natural logarithm, i is the imaginary unit ($i^2 = -1$), and π is pi. The transform is sometimes denoted by the symbol $\mathcal{F}$, as in $\mathbf{X} = \mathcal{F}\left\{\mathbf{x}\right\}$ or $\mathcal{F}\left(\mathbf{x}\right)$ or $\mathcal{F}\mathbf{x}$.

The **Inverse Discrete Fourier transform (IDFT)** is given by

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{\frac{2\pi i}{N} kn} \qquad n = 0, \dots, N-1.$$

Name	**Time domain**	**Frequency domain**	**Formula**		
	Domain property	Function property	Domain property	Function property	
(Continuous) Fourier transform	Continuous	Aperiodic	Continuous	Aperiodic	$S(f) = \int_{-\infty}^{\infty} s(t) \cdot e^{-i2\pi ft} dt$
Fourier series	Continuous	Periodic (τ)	Discrete	Aperiodic	$S[k] = \frac{1}{\tau} \int_0^{\tau} s(t) \cdot e^{-i2\pi \frac{k}{\tau} t} dt$
Discrete-time Fourier transform	Discrete	Aperiodic	Continuous	Periodic (f_s)	$S(f) = \frac{1}{f_s} \sum_{n=-\infty}^{\infty} s(n/f_s) \cdot e^{-i2\pi \frac{f}{f_s} n}$
ete Fourier transform	Discrete	Periodic (N)	Discrete	Periodic (N)	$S[k] = \sum_{n=0}^{N-1} s[n] \cdot e^{-i2\pi \frac{k}{N} n}$

Some Fourier transform properties

Notation: $f(t) \overset{\mathcal{F}}{\Longleftrightarrow} F(\omega)$ denotes that *f*(*t*) and *F*(ω) are a Fourier transform pair.

Linearity

$$a \cdot f(t) + b \cdot g(t) \quad \overset{\mathcal{F}}{\Longleftrightarrow} \quad a \cdot F(\omega) + b \cdot G(\omega)$$

Multiplication

$$f(t) \cdot g(t) \overset{\mathcal{F}}{\Longleftrightarrow} \quad \frac{1}{\sqrt{2\pi}} \cdot F(\omega) * G(\omega)$$ (unitary normalization convention)

$$\overset{\mathcal{F}}{\Longleftrightarrow} \quad \frac{1}{2\pi} \cdot F(\omega) * G(\omega)$$ (non-unitary convention)

$$\overset{\mathcal{F}}{\Longleftrightarrow} \quad F(f) * G(f)$$ (ordinary frequency)

e.g., **Modulation**

$$f(t) \cdot \cos\omega_0 t \quad \overset{\mathcal{F}}{\Longleftrightarrow} \quad \frac{1}{2}[F(\omega + \omega_0) + F(\omega - \omega_0)], \qquad \omega_0 \in \mathbb{R}$$

$$f(t) \cdot \sin\omega_0 t \quad \overset{\mathcal{F}}{\Longleftrightarrow} \quad \frac{i}{2}[F(\omega + \omega_0) - F(\omega - \omega_0)]$$

$$f(t) \cdot e^{i\omega_0 t} \quad \overset{\mathcal{F}}{\Longleftrightarrow} \quad F(\omega - \omega_0)$$

Convolution

$$f(t) * g(t) \quad \overset{\mathcal{F}}{\Longleftrightarrow} \quad \sqrt{2\pi} \cdot F(\omega) \cdot G(\omega)$$ (unitary convention)

$$\overset{\mathcal{F}}{\Longleftrightarrow} \quad F(\omega) \cdot G(\omega)$$ (non-unitary convention)

$$\overset{\mathcal{F}}{\Longleftrightarrow} \quad F(f) \cdot G(f)$$ (ordinary frequency)

e.g., **Integration**

$$f(t) * u(t) = \int_{-\infty}^{t} f(\tau)\, d\tau \quad \overset{\mathcal{F}}{\Longleftrightarrow} \quad \frac{1}{i\omega} F(\omega) + \pi F(0) \cdot \delta(\omega)$$

Conjugation

$$\overline{f(t)} \quad \overset{\mathcal{F}}{\Longleftrightarrow} \quad \overline{F(-\omega)}$$

Scaling

$$f(at) \quad \overset{\mathcal{F}}{\Longleftrightarrow} \quad \frac{1}{|a|} F\left(\frac{\omega}{a}\right), \qquad a \in \mathbb{R}, a \neq 0$$

Time reversal

$$f(-t) \quad \overset{\mathcal{F}}{\Longleftrightarrow} \quad F(-\omega)$$

Time shift

$$f(t - t_0) \quad \overset{\mathcal{F}}{\Longleftrightarrow} \quad e^{-i\omega t_0} \cdot F(\omega)$$

Parseval's theorem

$$\int_{-\infty}^{\infty} f(t) \cdot \overline{g(t)}\, dt = \int_{-\infty}^{\infty} F(\omega) \cdot \overline{G(\omega)}\, d\omega$$ (unitary convention)

$$= \frac{1}{2\pi} \cdot \int_{-\infty}^{\infty} F(\omega) \cdot \overline{G(\omega)}\, d\omega$$ (non-unitary convention)

$$= \int_{-\infty}^{\infty} F(f) \cdot \overline{G(f)}\, df$$ (ordinary frequency)

Sampling is the process of converting a signal (for example, a function of continuous time or space) into a numeric sequence (a function of discrete time or space).

Nyquist–Shannon sampling theorem

Exact reconstruction of a continuous-time baseband signal from its samples is possible if the signal is bandlimited and the sampling frequency is greater than twice the signal bandwidth

Taken from Matlab Help

For length *N* input sequence *x*, the DFT is a length N vector, *X*. `fft` and `ifft` implement the relationships

$$X(k) = \sum_{n=1}^{N} x(n) e^{-j2\pi(k-1)\left(\frac{n-1}{N}\right)} \qquad 1 \le k \le N$$

$$x(n) = \frac{1}{N} \sum_{k=1}^{N} X(k) e^{j2\pi(k-1)\left(\frac{n-1}{N}\right)} \qquad 1 \le n \le N$$

FFT

If $x(n)$ is real, you can rewrite the above equation in terms of a summation of sine and cosine functions with real coefficients:

$$x(n) = \frac{1}{N} \sum_{k=1}^{N} a(k) \cos\left(\frac{2\pi(k-1)(n-1)}{N}\right) + b(k) \sin\left(\frac{2\pi(k-1)(n-1)}{N}\right)$$

where

$$a(k) = \mathrm{real}(X(k)), \quad b(k) = -\mathrm{imag}(X(k)), \quad 1 \le n \le N$$

The FFT of a column vector `x`

```
x = [4 3 7 -9 1 0 0 0]' ;
```

is found with

```
y = fft(x)
```

which results in

```
y =
6.0000
11.4853 - 2.7574i
-2.0000 -12.0000i
-5.4853 +11.2426i
18.0000
-5.4853 -11.2426i
-2.0000 +12.0000i
11.4853 + 2.7574i
```

Notice that although the sequence `x` is real, `y` is complex. The first component of the transformed data is the constant contribution and the fifth element corresponds to the Nyquist frequency. The last three values of `y` correspond to negative frequencies and, for the real sequence `x`, they are complex conjugates of three components in the first half of `y`.

sinc

```
y = sinc(x)
```

Description

`sinc` computes the sinc function of an input vector or array, where the sinc function is

$$\operatorname{sinc}(t) = \begin{cases} 1, & t = 0 \\ \dfrac{\sin(\pi t)}{\pi t}, & t \neq 0 \end{cases}$$

This function is the continuous inverse Fourier transform of the rectangular pulse of width 2π and height 1.

$$\operatorname{sinc}(t) = \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{j\omega t} d\omega$$

`y = sinc(x)` returns an array `y` the same size as `x`, whose elements are the `sinc` function of the elements of `x`.

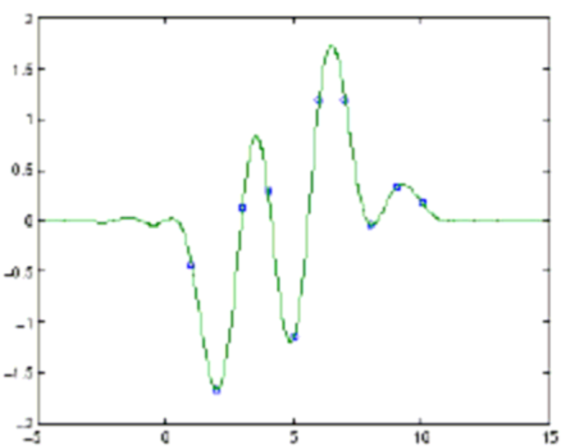
The space of functions bandlimited in the frequency range $\omega \in [-\pi,\pi]$ is spanned by the infinite (yet countable) set of sinc functions shifted by integers. Thus any such bandlimited function $g(t)$ can be reconstructed from its samples at integer spacings.

$$g(t) = \sum_{n=-\infty}^{\infty} g(n) \operatorname{sinc}(t-n)$$

Examples

Perform ideal bandlimited interpolation by assuming that the signal to be interpolated is 0 outside of the given time interval and that it has been sampled at exactly the Nyquist frequency:

```
t = (1:10)';                    % Column vector of time samples
randn('state',0);
x = randn(size(t));             % Column vector of data
ts = linspace(-5,15,600)';      % Times at which to interpolate
y = sinc(ts(:,ones(size(t))) - t(:,ones(size(ts)))')*x;
plot(t,x,'o',ts,y)
```



Signum function

Y = sign(X)

Description

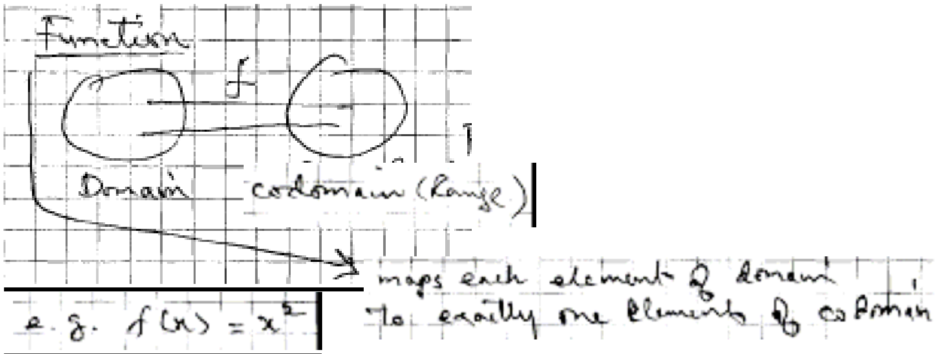
`Y = sign(X)` returns an array `Y` the same size as `X`, where each element of `Y` is:

- `1` if the corresponding element of `X` is greater than zero
- `0` if the corresponding element of `X` equals zero
- `-1` if the corresponding element of `X` is less than zero

For nonzero complex `X`, `sign(X) = X./abs(X)`.

Functions

(Class discussion)



Odd , Even. Neither, Complex, Real, Periodic ,aperiodic, Continuous, Discontinuous functions Bandlimited , Basband

Sampling is the process of converting a signal (for example, a function of continuous time or space) into a numeric sequence (a function of discrete time or space).

Nyquist–Shannon sampling theorem

Exact reconstruction of a continuous-time baseband signal from its samples is possible if the signal is bandlimited and the sampling frequency is greater than twice the signal bandwidth

From class work

Convolution

Fourier Transform of a constant is dirac delta fn.

$f(t)$	$F(\omega)$
df/dt	$j\omega F(\omega)$
$\int_{-\infty}^{t} f(t)\,dt$	$\frac{F(\omega)}{j\omega}$
$f(t-t_0)$	$F(\omega)e^{-j\omega t_0}$
$f(t)e^{j\omega_0 t}$	$F(\omega-\omega_0)$
$f(t) * g(t)$	$F(\omega) \cdot G(\omega)$

Convolution

$w[k] = \sum_j u(j)\, v[k+1-j]$

$w(1) = u(1) * v(1)$

$w(2) = u(1) * v(2) + u(2) * v(1)$

$w(3) = u(1) * v(3) + u(2) * v(2) + u(3) * v(1)$

$\vdots$

$w(n) = u(1) * v(n) \cdots \cdots u(n) * v(1)$

$w(2n-1) = u(n) * v(n)$ $\quad T_0 = t \pm n t_0$

T_0= t+- nt_0

Convolution

$$f(t) * g(t) = \int^{\infty} f(t')\, g(t-t')\, dt'$$

something which gives you estimation of $f(t)$

only $g(t)$ is compact & $f(t)$ doesn't change considerably then u can say $g(t)$ is sampling $f(t)$

	$f(t)$	$F(\omega)$	$\Theta(\omega)$
Single Sided Exp.	$e^{-at}u(t)$	$\frac{1}{\sqrt{a^2+\omega^2}}$	$e^{-j\tan^{-1}(\omega/a)}$
Double sided Exp	$e^{-a\|t\|}$	$\frac{2a}{a^2+\omega^2}$	
Gate fn	$G(t) = \begin{cases} 1 & \|t\| < \tau/2 \\ 0 & \|t\| > \tau/2 \end{cases}$	$A\tau\, Sa(\frac{\omega\tau}{2})$	
Impulse	$f(t) = \delta(t)$	1	
Constant Sign(t)	$Sgn(t) = \begin{cases} 1 & t>0 \\ -1 & t<0 \end{cases}$	$2u(t)-1$	

From MATLAB HELP

Convolution Theorem

Theorem: For any $x, y \in \mathbf{C}^N$, $\boxed{x \circledast y \longleftrightarrow X \cdot Y.}$

Proof:

$$
\begin{aligned}
\mathrm{DFT}_k(x \circledast y) &\triangleq \sum_{n=0}^{N-1} (x \circledast y)_n e^{-j2\pi nk/N} \\
&\triangleq \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} x(m) y(n-m) e^{-j2\pi nk/N} \\
&= \sum_{m=0}^{N-1} x(m) \sum_{n=0}^{N-1} \underbrace{y(n-m) e^{-j2\pi nk/N}}_{e^{-j2\pi mk/N} Y(k)} \\
&= \left(\sum_{m=0}^{N-1} x(m) e^{-j2\pi mk/N} \right) Y(k) \quad \text{(by the Shift Theorem)} \\
&\triangleq X(k) Y(k)
\end{aligned}
$$

This is perhaps the most important single Fourier theorem of all. It is the basis of a large number of FFT applications. Since an FFT provides a fast Fourier transform, it also provides *fast convolution*, thanks to the convolution theorem. It turns out that using an FFT to perform convolution is really more efficient in practice only for reasonably long convolutions, such as $N > 100$. For much longer convolutions, the savings become enormous compared with ``direct'' convolution. This happens because direct convolution requires on the order of N^2 operations (multiplications and additions), while FFT-based convolution requires on the order of $N \lg(N)$ operations, where $\lg(N)$ denotes the logarithm-base-2 of N.

<u>Taken from Standfor Uni and Matlab Help</u>

The simple matlab example and figure illustrates how much faster convolution can be performed using an FFT. We see that for a length $N = 1024$ convolution, the `fft` function is approximately 300 times faster in Octave, and 30 times faster in Matlab. (The `conv` routine is much faster in Matlab, even though it is a built-in function in both cases.)

```
N = 1024;          % FFT much faster at this length
t = 0:N-1;         % [0,1,2,...,N-1]
h = exp(-t);       % filter impulse reponse
H = fft(h);        % filter frequency response
x = ones(1,N);     % input = dc (any signal will do)
Nrep = 100;        % number of trials to average
t0 = clock;        % latch the current time
for i=1:Nrep, y = conv(x,h); end
                   % Direct convolution
t1 = etime(clock,t0)*1000;
    % elapsed time in msec
t0 = clock;
for i=1:Nrep, y = ifft(fft(x) .* H); end
                   % FFT convolution
t2 = etime(clock,t0)*1000;
disp(sprintf([...
    'Average direct-convolution time = %0.2f 
msec\n',...
    'Average FFT-convolution time = %0.2f msec\n',...
    'Ratio = %0.2f (Direct/FFT)'],...
    t1/Nrep,t2/Nrep,t1/t2));
```