EP375 Computational Physics

Topic 11

FOURIER TRANSFORM

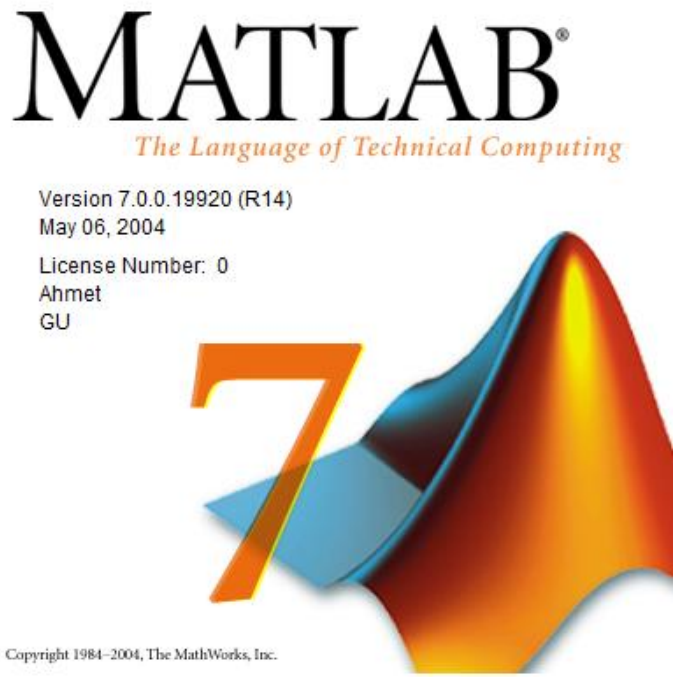
Department of
Engineering Physics

University of Gaziantep

Apr 2014

Content

1. Introduction
2. Continues Fourier Trans.
3. DFT in MATLAB and C++
4. Power Spectrum
5. MATLAB fft() function
6. Examples



Introduction

- The Fourier transform is a mathematical operation that decomposes a function into its constituent frequencies, known as its frequency spectrum.
- For instance, the transform of a musical chord made up of pure notes is a mathematical representation of the amplitudes and phases of the individual notes that make it up.

- The composite waveform depends on time, and therefore is called the time domain representation. The frequency spectrum is a function of frequency and is called the frequency domain representation.
- Each value of the function is a complex number (called complex amplitude) that encodes both a magnitude and phase component. The term "Fourier transform" refers to both the transform operation and to the complex-valued function it produces.

Continues Fourier Transform

Any periodic function f(t) = f(t+T) = f(t+2T) = f(t+2T)= …

can be written as a sum of simple harmonic (sinusoidal) functions having various frequencies as follows:

$$
\begin{aligned}
f(t) = a_0 & \\
& + a_1 \cos(\omega t) \quad + b_1 \sin(\omega t) \\
& + a_2 \cos(2\omega t) \quad + b_2 \sin(2\omega t) \\
& + a_3 \cos(3\omega t) \quad + b_3 \sin(3\omega t) \\
& + \cdots \qquad\qquad + \cdots
\end{aligned}
$$

or

$$
f(t) = a_0 + \sum_{k=1}^{\infty} [a_k \cos(k\omega t) + b_k \sin(k\omega t)]
$$

where T is the period and w = 2pi/T.

$$f(t) = a_0 + \sum_{k=1}^{\infty} [a_k \cos(k\omega t) + b_k \sin(k\omega t)]$$

Here the numerical constants can be found from:

$$a_0 = \frac{1}{T} \int_0^T f(t) dt$$

$$a_k = \frac{2}{T} \int_0^T f(t) \cos(k\omega t) dt$$

$$b_k = \frac{2}{T} \int_0^T f(t) \sin(k\omega t) dt$$

$$f(t) = a_0 + \sum_{k=1}^{\infty} [a_k \cos(k\omega t) + b_k \sin(k\omega t)]$$

By using Euler equations:

$$e^{-iAt} = \cos(At) - i\sin(At)$$

$$e^{+iAt} = \cos(At) + i\sin(At)$$

One can get the complex form of the exapnsion:

$$f(t) = \sum_{k=-\infty}^{\infty} c_k e^{ik\omega t}$$

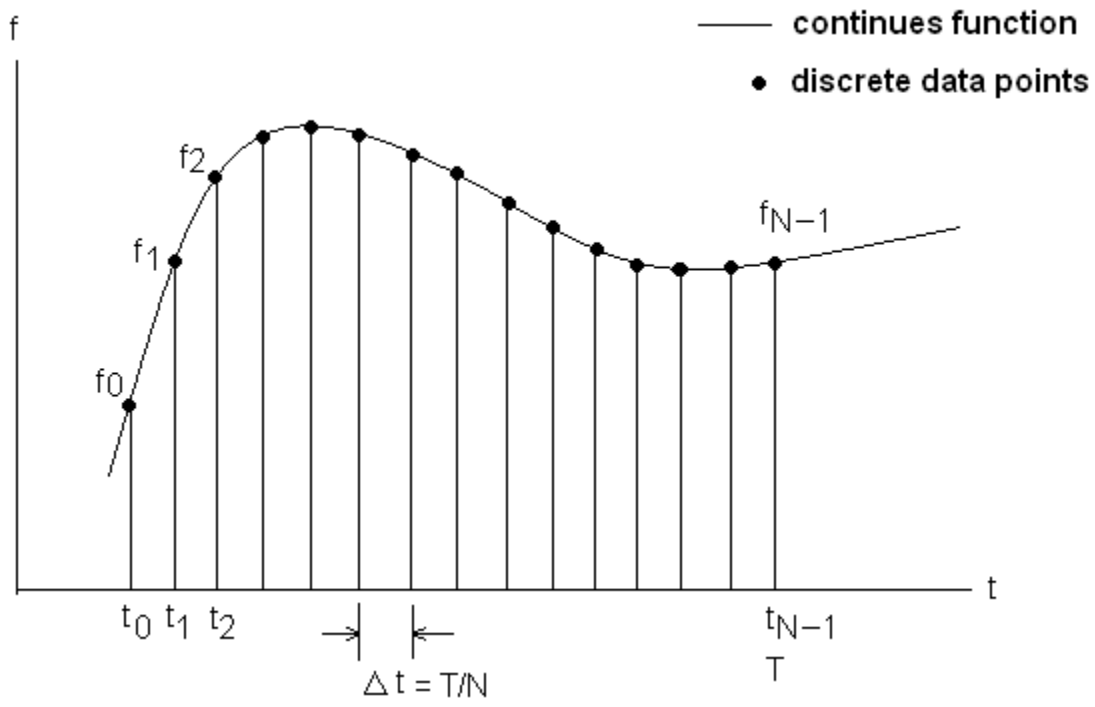
where

$$c_k = \frac{1}{T} \int_{-T/2}^{T/2} f(t)\, e^{-ik\omega t}$$

Fourier transform of f(t) is basically finding the coefficients c_k.

Discrete Fourier Transform (DFT)

- In engineering, we often encounter with finite set of discrete values.
- Suppose you have a signal stored as a series of N data points (tn , fn) equally spaced in time by time interval T
  from t = 0 to t = tfinal = (N-1)T.



t	f
t0	f0
t1	f1
t2	f2
.	.
.	.
.	.

- For totally N points then:

$$f_n = \sum_{k=0}^{N-1} [c_k \cos(k\omega n) + i\, c_k \sin(k\omega n)]$$

$$c_k = \sum_{n=0}^{N-1} [f_n \cos(k\omega n) - i\, f_n \sin(k\omega n)]$$

where w = 2pi/N.

Vector c is full of complex numbers because c stores the phase relationship between frequency components as well as amplitude information, just like the Fourier transform.

Basic DFT Algorithm

```
Define fn  (vector of N-discreate values)
Define w = 2*pi/N
for k=0, N-1
  real = 0
  imag = 0
  for n=0, N-1
     A = k*w*n
     real = real + fn * cos(A)
     imag = imag + fn * sin(A)
  end
  ck = complex(real,imag)
end
```

A MATLAB DFT Function

```
function c = dft(x, npoints)
  if nargin==1
     N = length(x);
  else
     N = npoints;
  end
  c = zeros(N-1, 1); % c_k is the fourier trasform of x
  w = 2*pi/N;
  for k = 0:N-1
    real = 0;
    imag = 0;
    for n = 0:N-1
      A = k*w*n;
      real = real + x(n+1) * cos(A);
      imag = imag - x(n+1) * sin(A);
    end
    c(k+1) = complex(real, imag);
  end
end
```

A C++ DFT Function

```
std::vector< std::complex<double> >
dft(std::vector<double> x, int npoints=-1)
{
  const double pi = 3.141592654;
  int N = x.size();
  if(npoints !=-1 && npoint<N) N = npoints;
  std::vector< complex<double> > c; // c_k is the fourier trasform of x
  double w = 2.0*pi/N, real, imag;
  for(int k=0; k<N-1; k++){
     real = imag = 0.0;
     for(int n=0; n<N-1; n++){
       double A = k*w*n;
       real = real + x[n] * cos(A);
       imag = imag - x[n] * sin(A);
     }
     c.push_back( std::complex<double>(real, imag) );
  }
  return c;
}
```

MATLAB fft() function

In MATLAB we can use the build-in function (**`fft()`**)
to perform Fast Fourier Transform.

`Y = fft(X)`

returns the discrete Fourier transform (DFT) of vector X,
computed with a fast Fourier transform (FFT) algorithm.

`Y = fft(X,n)`

returns the n-point DFT.

- The functions **`X = fft(x)`** and **`x = ifft(X)`** implement the transform and inverse transform pair given for vectors of length *N* by:

$$X(k) = \sum_{j=1}^{N} x(j)\omega_N^{(j-1)(k-1)}$$

$$x(j) = (1/N)\sum_{k=1}^{N} X(k)\omega_N^{-(j-1)(k-1)}$$

where

$$\omega_N = e^{(-2\pi i)/N}$$

is an *N* root of unity.

Power Spectrum

Both **`dft()`** and **`fft()`** functions returns a list of complex values:

```
>> c = fft(x)
```

Here vector c is full of complex numbers because it stores the phase relationship between frequency components as well as amplitude information, just like the Fourier transform.

When we don't care about the phase information contained in c we can work instead with the power spectrum $P = |c|^2$.

There are many applications of Power Spectra such as Vibration Analysis, Quantum Mechanics, Signal Processing, …

Power of a signal is defined as:

$$P = \frac{1}{T}\int_0^T f^2(t)dt = \sum_{k=-\infty}^{\infty} |\, c_k \,|^2$$

In MATLAB:

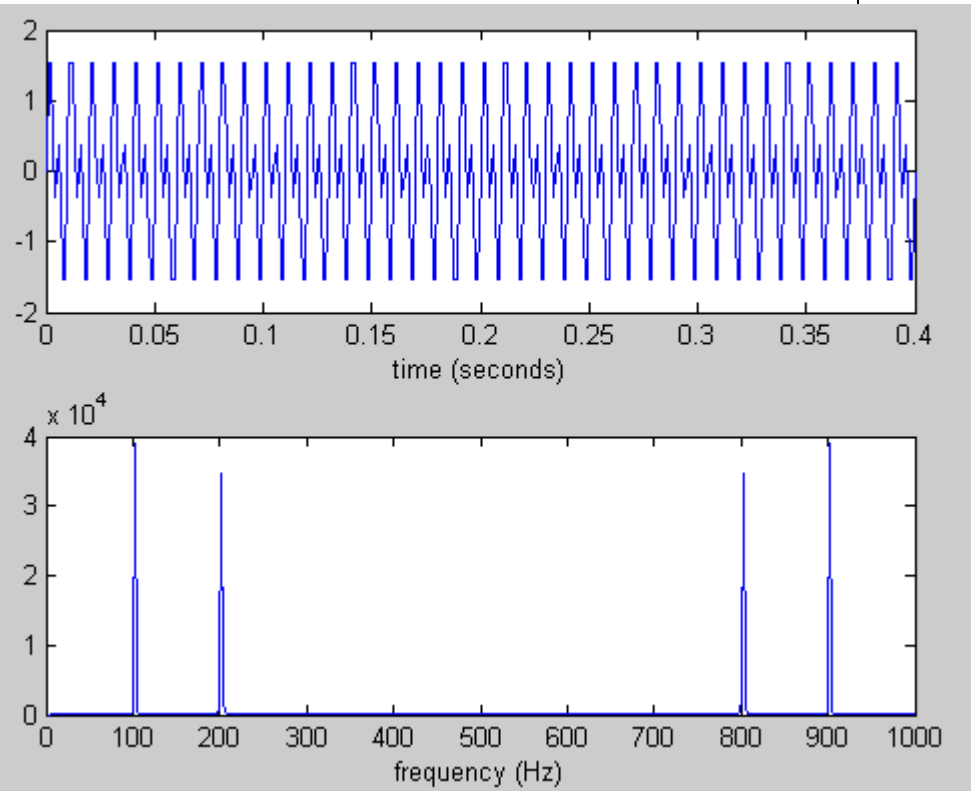
```
>> c = fft(x)
>> P = abs(c).^2
```

fft1.m

```
dt= 0.001;
% time
t = 0:dt:0.4;
% signal (time domain)
x = sin(2*pi*100*t) + sin(2*pi*200*t);

subplot(2,1,1)
plot(t, x)
xlabel('time (seconds)')

subplot(2,1,2)
c = fft(x);
N = length(c);
% frequency
f = (1/dt) * (1:N)/N;
% power (frequency domain)
P = abs(c).^2;
plot(f, P)
xlabel('frequency (Hz)')
```

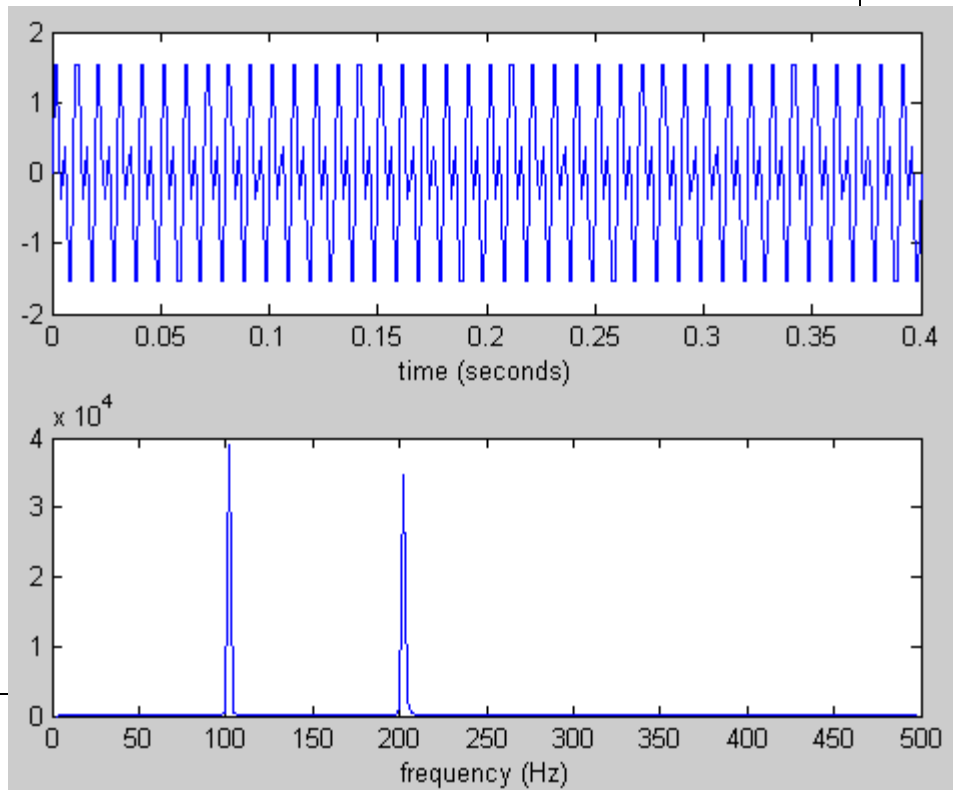


fft2.m

```
dt= 0.001;
% time
t = 0:dt:0.4;
% signal (time domain)
x = sin(2*pi*100*t) + sin(2*pi*200*t);

subplot(2,1,1)
plot(t, x)
xlabel('time (seconds)')

subplot(2,1,2)
c = fft(x);
N = length(c);
% frequency
f = (1/dt) * (1:N/2)/N;
% power (frequency domain)
P = abs(c).^2;
plot(f, P(1:N/2))
xlabel('frequency (Hz)')
```

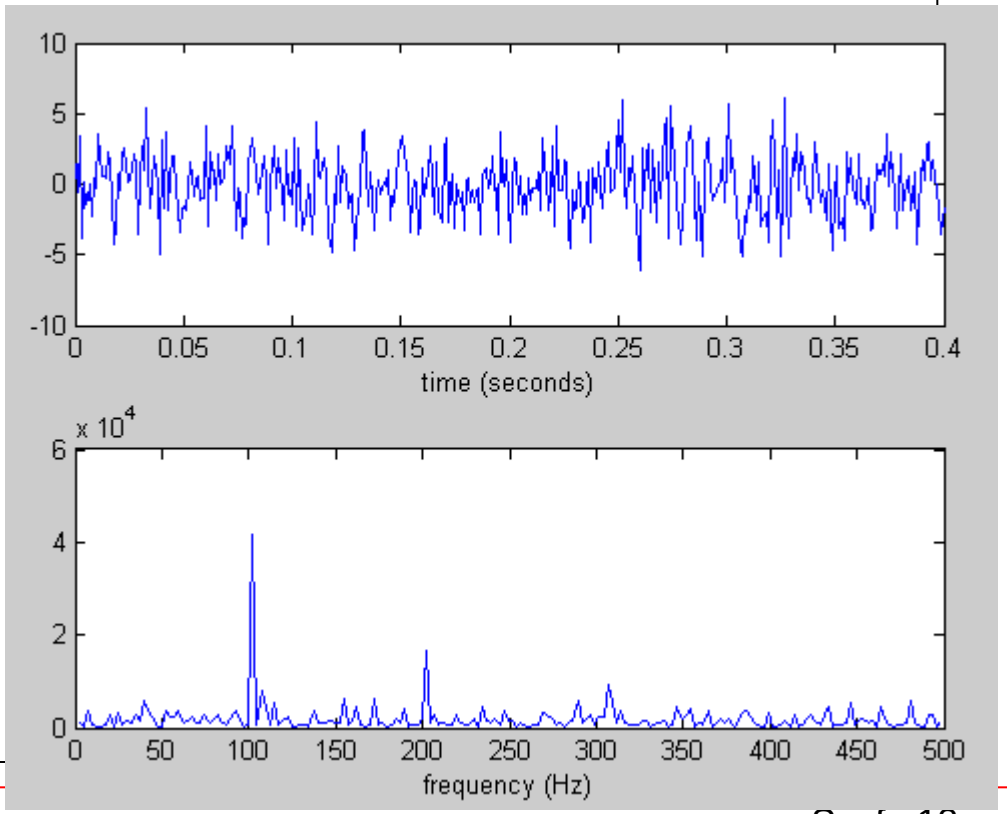


fft3.m

```
dt= 0.001;
% time
t = 0:dt:0.4;
% signal (time domain)
x = sin(2*pi*100*t) + sin(2*pi*200*t);
% random noise
x = x + 2*randn(1,length(t));

subplot(2,1,1)
plot(t, x)
xlabel('time (seconds)')

subplot(2,1,2)
c = fft(x);
N = length(c);
% frequency
f = (1/dt) * (1:N/2)/N;
% power (frequency domain)
P = abs(c).^2;
plot(f,P(1:N/2))
xlabel('frequency (Hz)')
```



References:

[1]. http://www.mathworks.com/products/matlab

[2]. Numerical Methods in Engineering with MATLAB,
J. Kiusalaas, Cambridge University Press (2005)

[3]. Numerical Methods for Engineers, 6th Ed.
S.C. Chapra, Mc Graw Hill (2010)

[4]. http://en.wikipedia.org/wiki/Fourier_transform