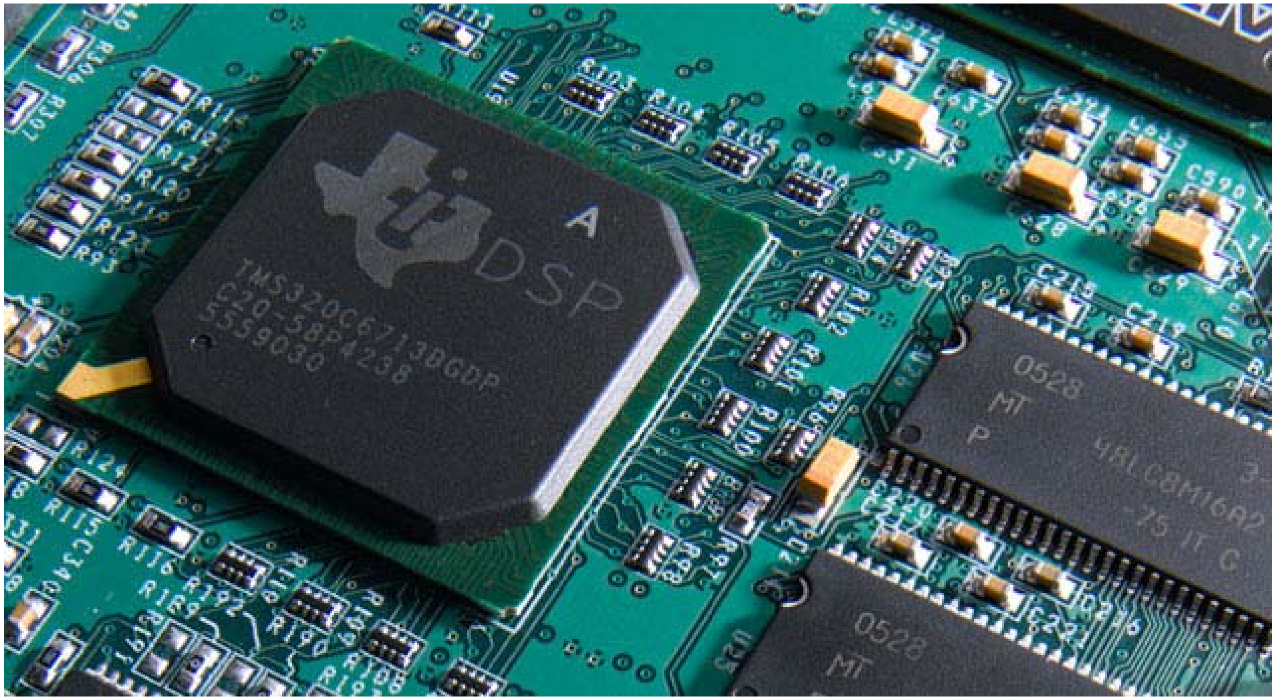
NUS National University of Singapore
Department of Electrical & Computer Engineering
Faculty of Engineering

Lecture 4: Signal Processing in Matlab



Outline

- Background
- Fourier series and Fourier transform
- Sampling and aliasing
- DTFT, DFT, and FFT
- Filter Design

Why signals should be processed?

- Signals are carriers of information
  - Useful and unwanted
  - Extracting, enhancing, storing and transmitting the useful information

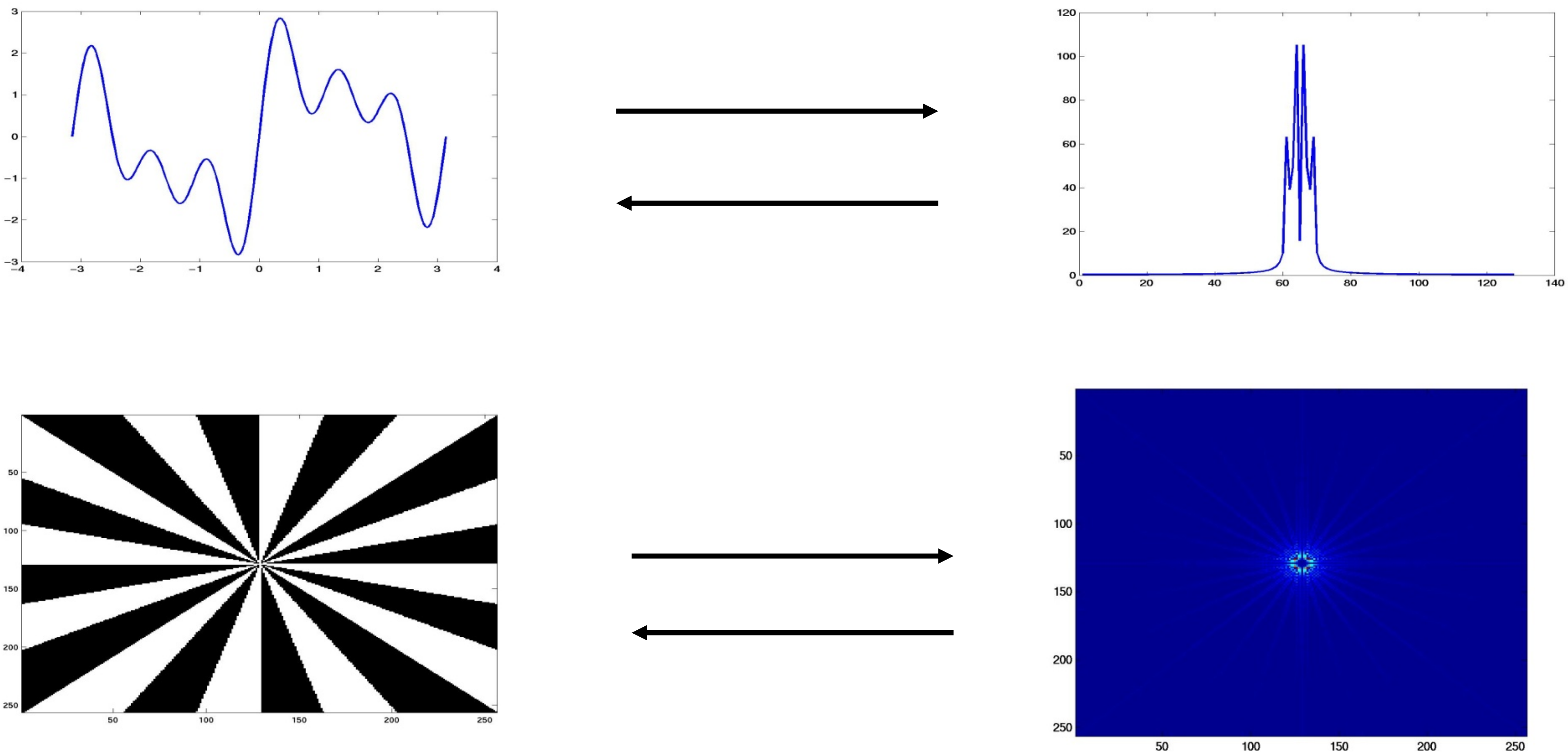
- How signals are being processed?
  - Analog Signal Processing
  - Digital Signal Processing

Two categories of tasks

- Signal Analysis:
  - Measurement of signal properties
  - Spectrum(frequency/phase) analysis
  - Target detection, verification, recognition
- Signal Filtering:
  - Signal-in-signal-out, filter
  - Removal of noise/interference
  - Separation of frequency bands

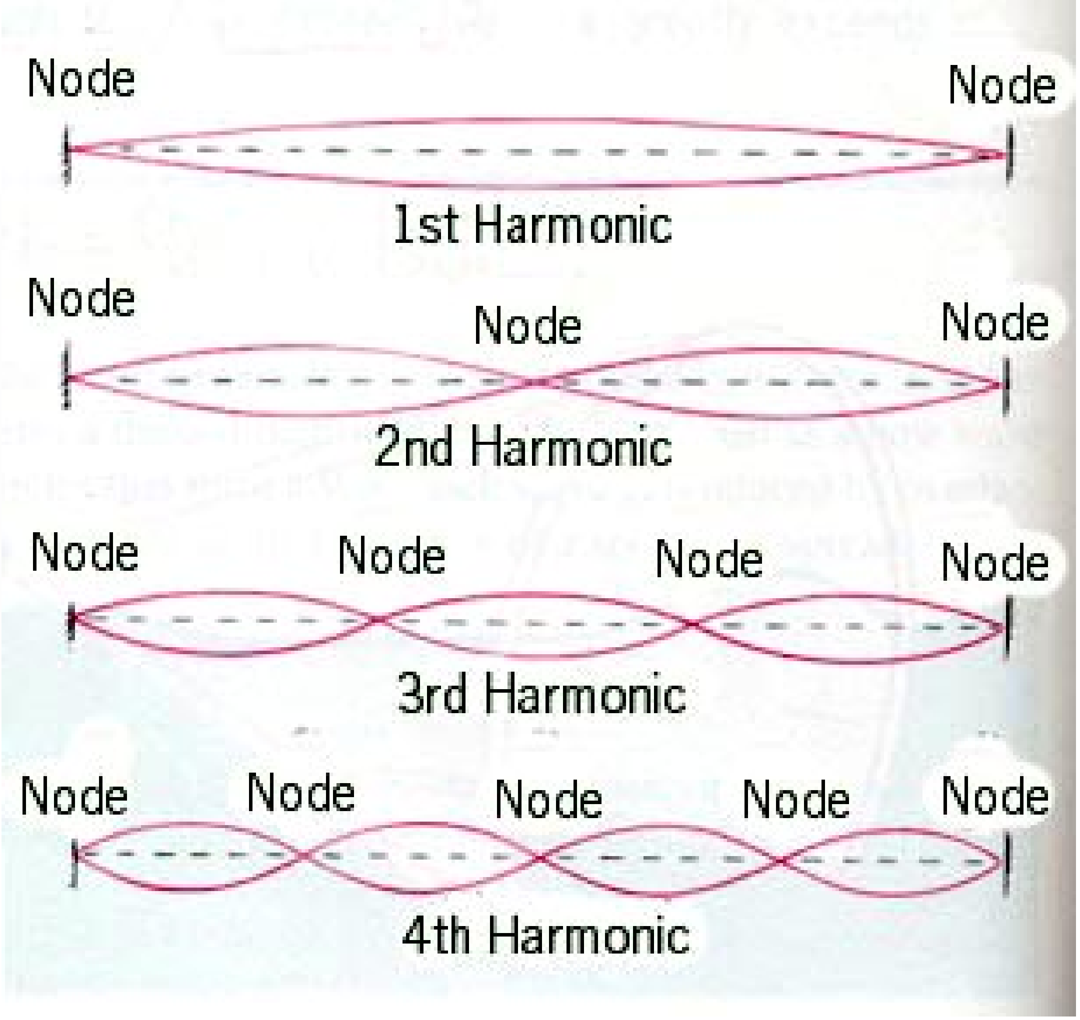
What is frequency domain analysis ?

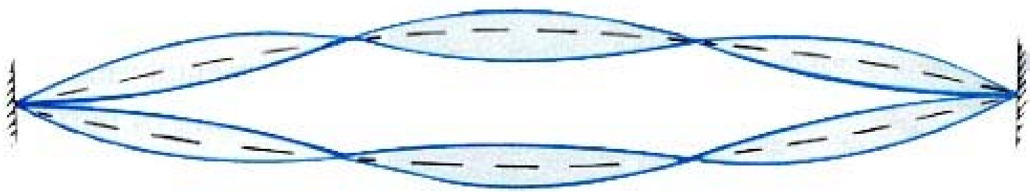
- Analyzes the signals in the frequency space.
- Primarily involves interpreting the spectrum.



Fourier Analysis

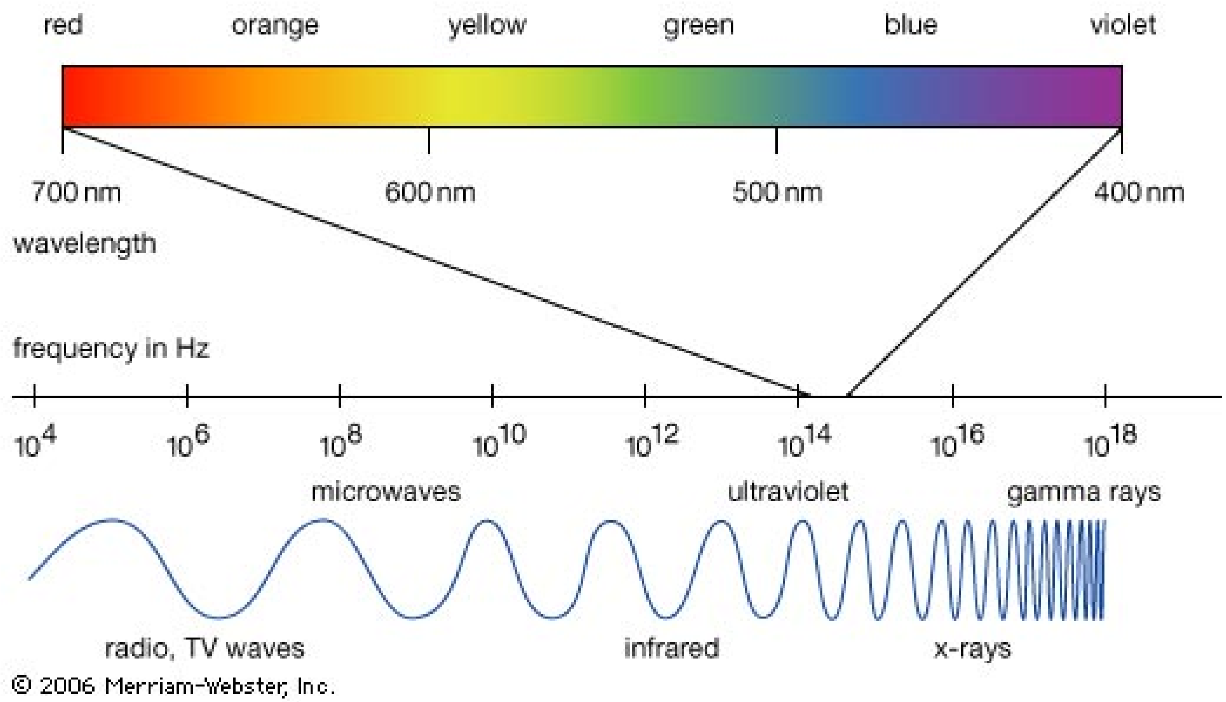
Harmonic Frequency

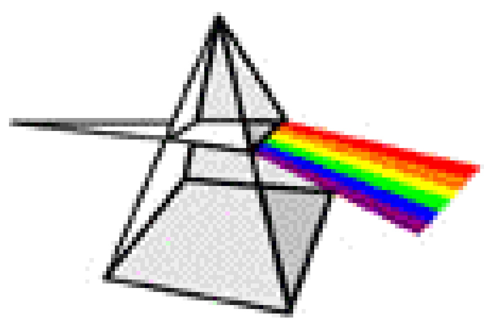




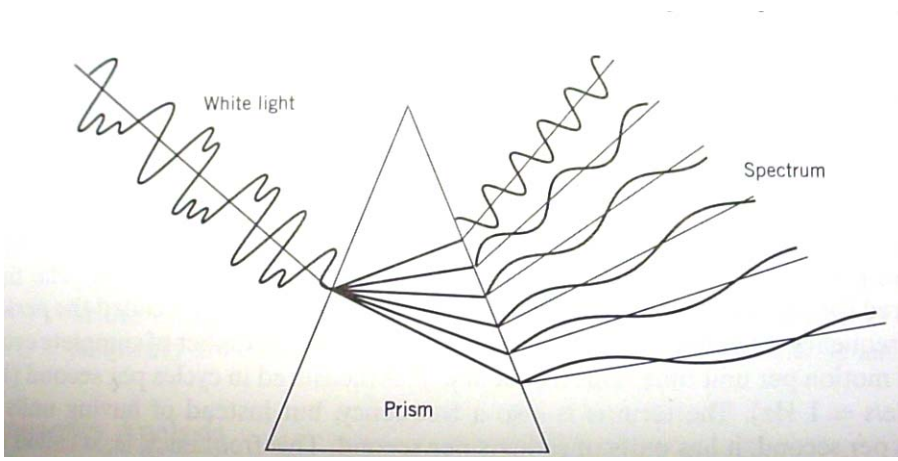
NUS National University of Singapore

Electromagnetic Spectrum





triangular prism diffraction



Fourier Series

$$x(t)=\sum_{k=-\infty}^{\infty} X_k e^{jk\frac{2\pi}{T_0}t} \quad \Leftrightarrow \quad X_k=\frac{1}{T_0}\int_{T_0} x(t)e^{-jk\frac{2\pi}{T_0}t}dt$$

IF	THEN...
$x(t)$ is real	$X_{-k}=X_k^*$
$x(t)$ is even	$X_{-k}=X_k$
$x(t)$ is odd	$X_{-k}=-X_k$
$x(t)$ is real and even	$X_{-k}=X_k^*,\ X_{-k}=X_k$
$x(t)$ is real and odd	$X_{-k}=X_k^*,\ X_{-k}=-X_k$

NUS National University of Singapore

Fourier Transform

$$x(t) = \int_{-\infty}^{+\infty} X(f) e^{j2\pi f t} df \quad \Leftrightarrow \quad X(f) = \int_{-\infty}^{+\infty} x(t) e^{-j2\pi f t} dt$$

IF	**THEN…**
$x(t)$ is real	$X(-f) = X(f)^*$
$x(t)$ is even	$X(-f) = X(f)$
$x(t)$ is odd	$X(-f) = -X(f)$
$x(t)$ is real and even	$X(-f) = X(f)^*,\ X(-f) = X(f)$
$x(t)$ is real and odd	$X(-f) = X(f)^*,\ X(-f) = -X(f)$

NUS National University of Singapore

Property

Uniqueness

$$x_1(t) = x_2(t) \Leftrightarrow X_1(j\omega) = X_2(\mathrm{j}\omega)$$

Homogeneity

$$\mathcal{F}(Kx(t)) = K\mathcal{F}(x(t))$$

$$\mathcal{F}^{-1}(KX(j\omega)) = K\mathcal{F}^{-1}(X(j\omega))$$

Addition

$$\mathcal{F}(x_1(t) + x_2(t)) = \mathcal{F}(x_1(t)) + \mathcal{F}(x_2(t))$$

$$\mathcal{F}^{-1}(X_1(j\omega) + X_2(j\omega)) = \mathcal{F}^{-1}(X_1(j\omega)) + \mathcal{F}^{-1}(X_2(j\omega))$$

Differentiation

$$\mathrm{D}x(t) = \frac{\mathrm{d}x(t)}{\mathrm{d}t}$$

$$\mathcal{F}(\mathrm{D}x(t)) = j\omega\mathcal{F}(x(t)) = j\omega X(j\omega)$$

$$\mathcal{F}^{-1}(j\omega X(j\omega)) = \mathrm{D}\mathcal{F}^{-1}(X(j\omega)) = \mathrm{D}x(t)$$

NUS National University of Singapore

Property

Convolution

$$[g*h](t) \equiv \int_{-\infty}^{+\infty} g(\tau)h(t-\tau)d\tau \quad \Leftrightarrow \quad G(f)H(f)$$

Correlation

$$\langle g(\tau)h(\tau+t)\rangle \equiv \int_{-\infty}^{+\infty} g(\tau)h(\tau+t)d\tau \quad \Leftrightarrow \quad G(-f)H(f)$$

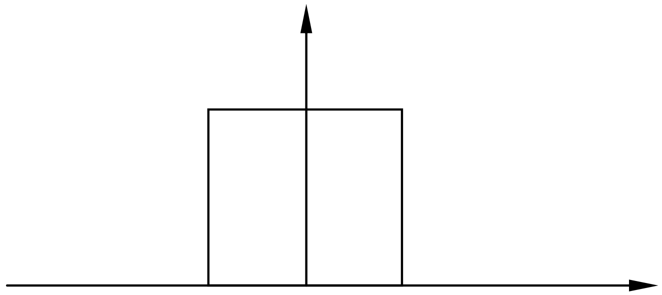
Total power:

$$\int_{-\infty}^{+\infty} |x(t)|^2\, dt = \int_{-\infty}^{+\infty} |X(f)|^2\, df$$

Autocorrelation if $g = h$. Autocorrelation is equal to power spectrum $|G(f)|^2$ in frequency space.

Fourier Transform

$$x(t) = \int_{-\infty}^{+\infty} X(f) e^{j2\pi f t} df \quad \Leftrightarrow \quad X(f) = \int_{-\infty}^{+\infty} x(t) e^{-j2\pi f t} dt$$

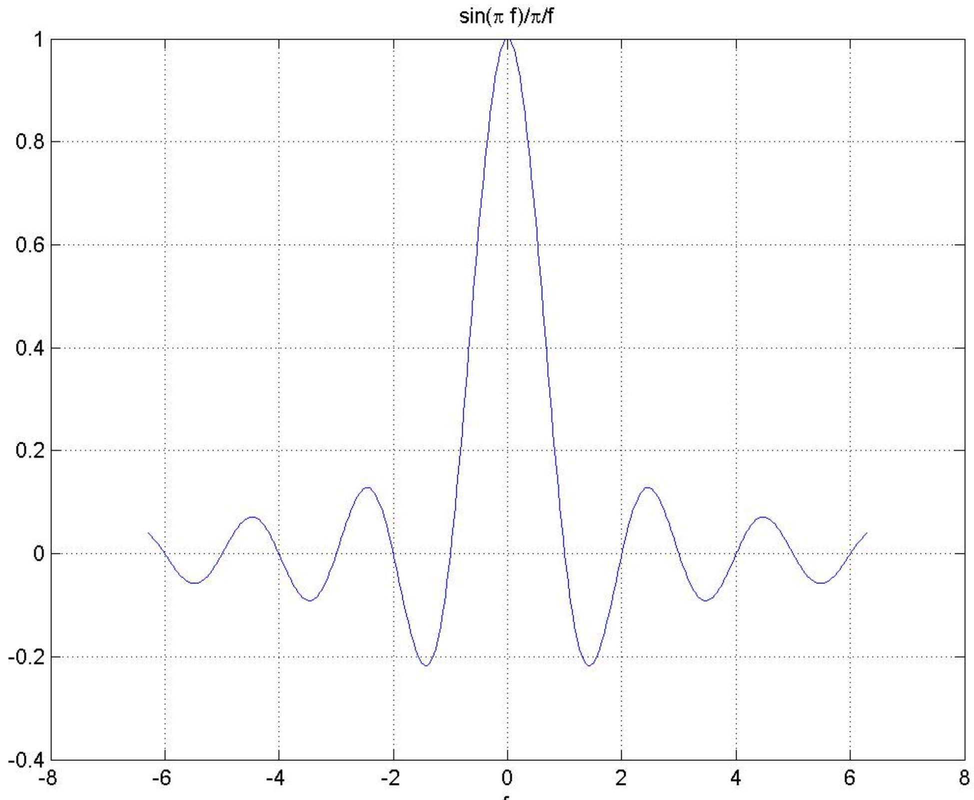


```
>> clear,
>> syms f t;
>> g = exp(-i*2*pi*f*t);
>> Xf = int(g,t,-.5,.5);
>> pretty(Xf)

               -1/2 i (exp(2 i pi f) - 1) exp(-i pi f)
               ---------------------------------------
                                pi f
```

Exercise: Is the result a sinc function? Try **`simple`** and **`ezplot`**

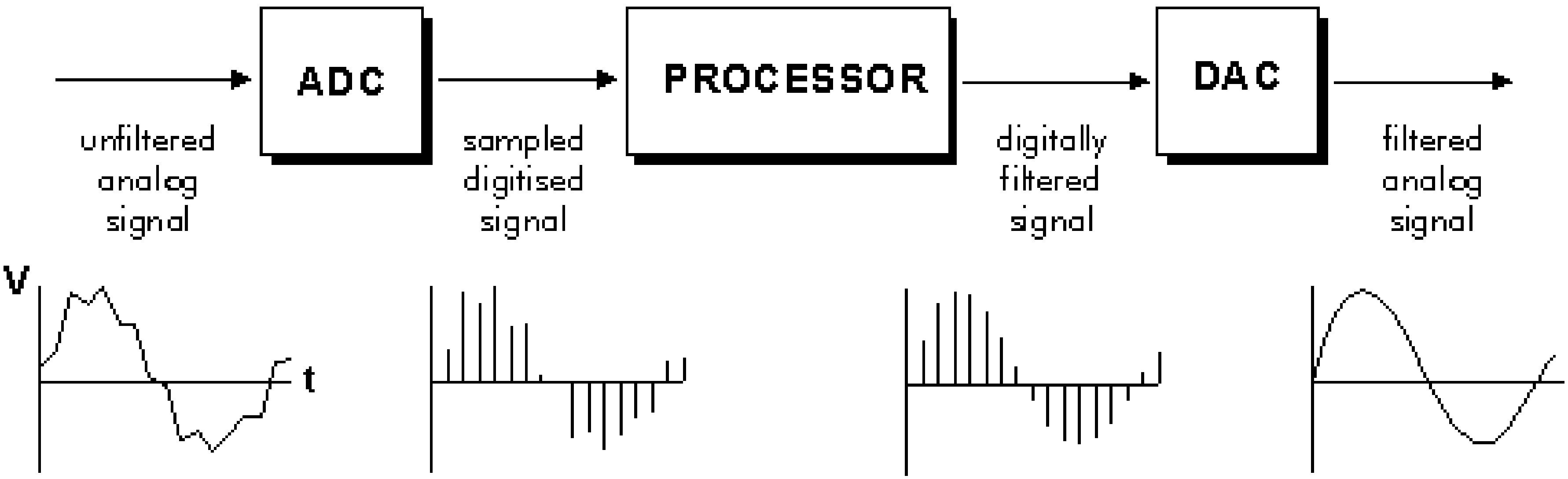
Fourier Transform

```
>> Xfs = simple(Xf);
>> pretty(Xfs)
                                    sin(pi f)
                                    ---------
                                      pi f

>> ezplot(Xfs)
>> axis auto, grid on
```



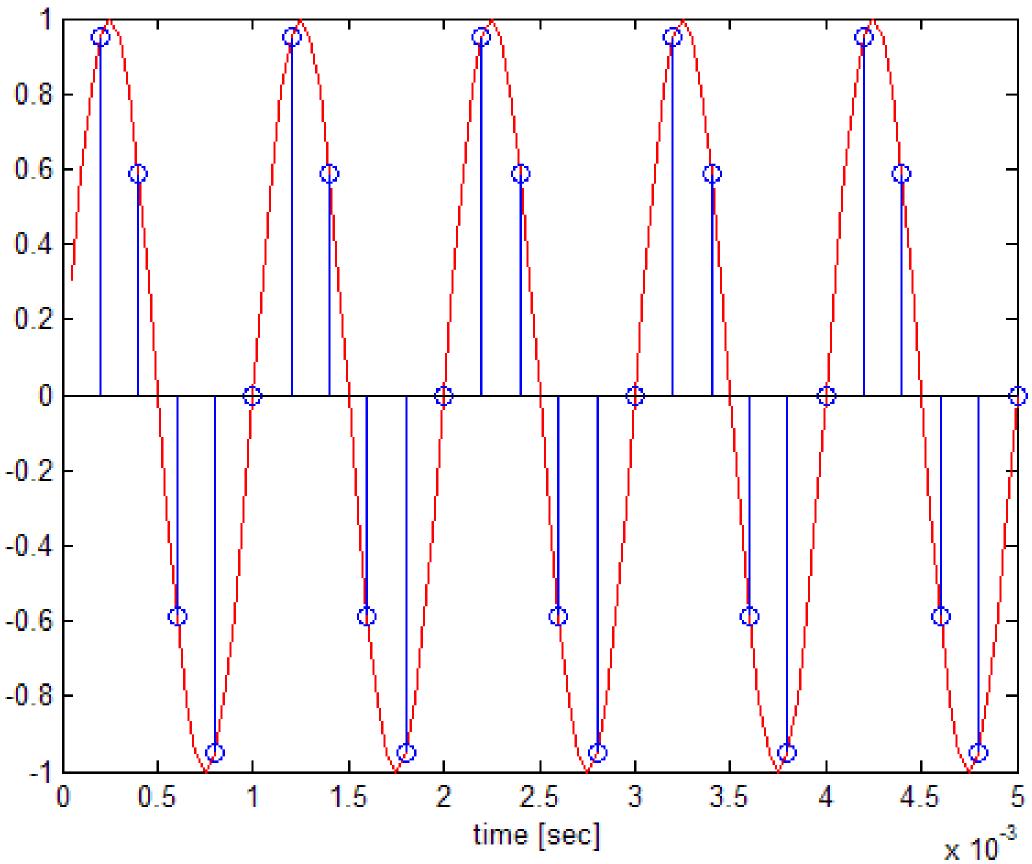
Sampling

The framework of DSP



Sampling

- Read values from a continuous signal
- Equally spaced time interval (sampling frequency)

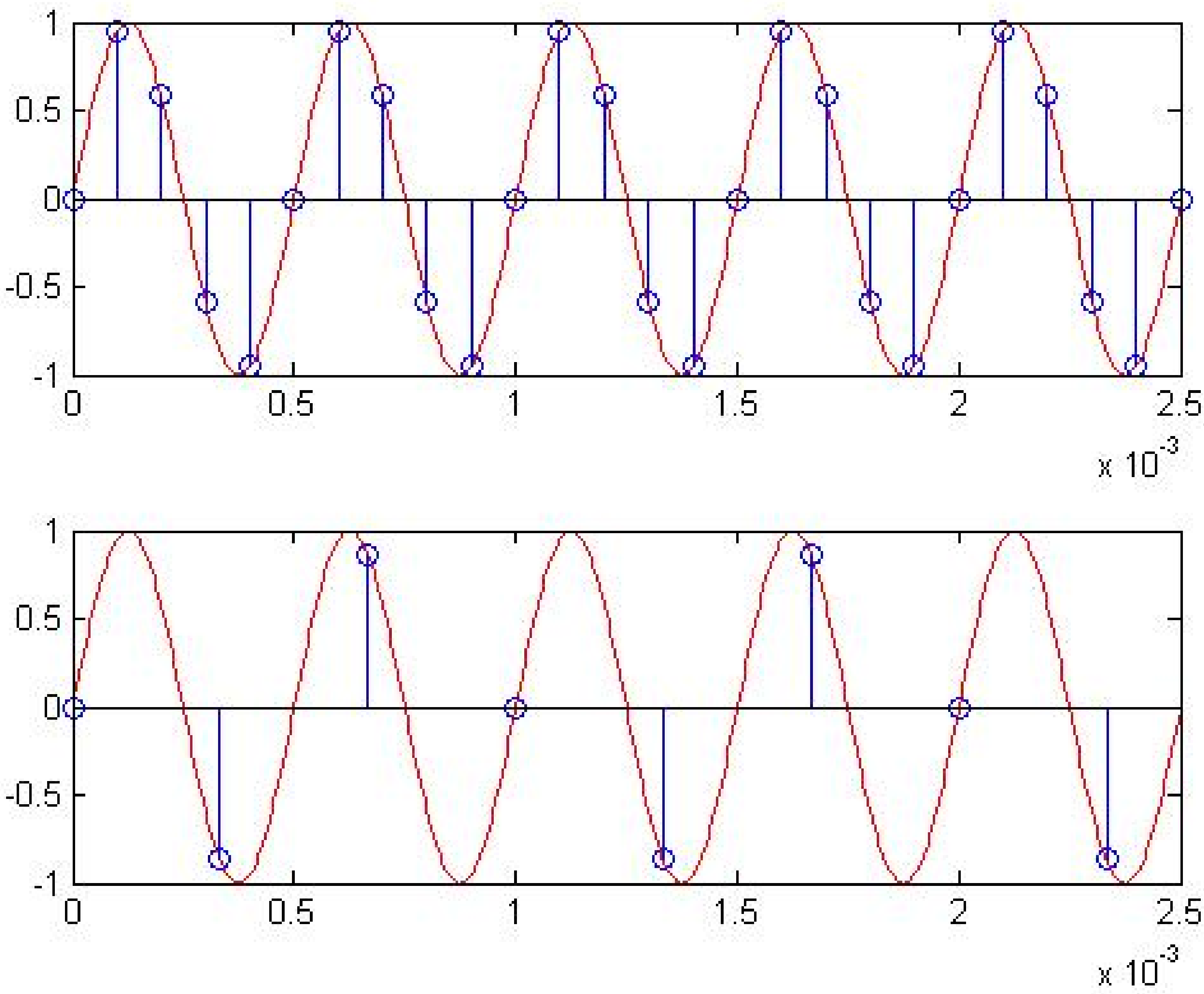


Sampling: Example

```
f = 2000; T = 1/f; tmin = 0; tmax = 5*T;
dt = T/100;
t = tmin:dt:tmax;
x = sin(2*pi*f*t); %original sinusoid signal with f = 2 kHz

dt1 = 1/10000; %sampled at 10 kHz
t1 = tmin:dt1:tmax;
x1 = sin(2*pi*f*t1);

dt2 = 1/3000; %sampled at 3 kHz
t2 = tmin:dt2:tmax;
x2 = sin(2*pi*f*t2);

subplot(211)
plot(t,x,'r'); hold on; stem(t1,x1);
subplot(212)
plot(t,x,'r'); hold on; stem(t2,x2);
```

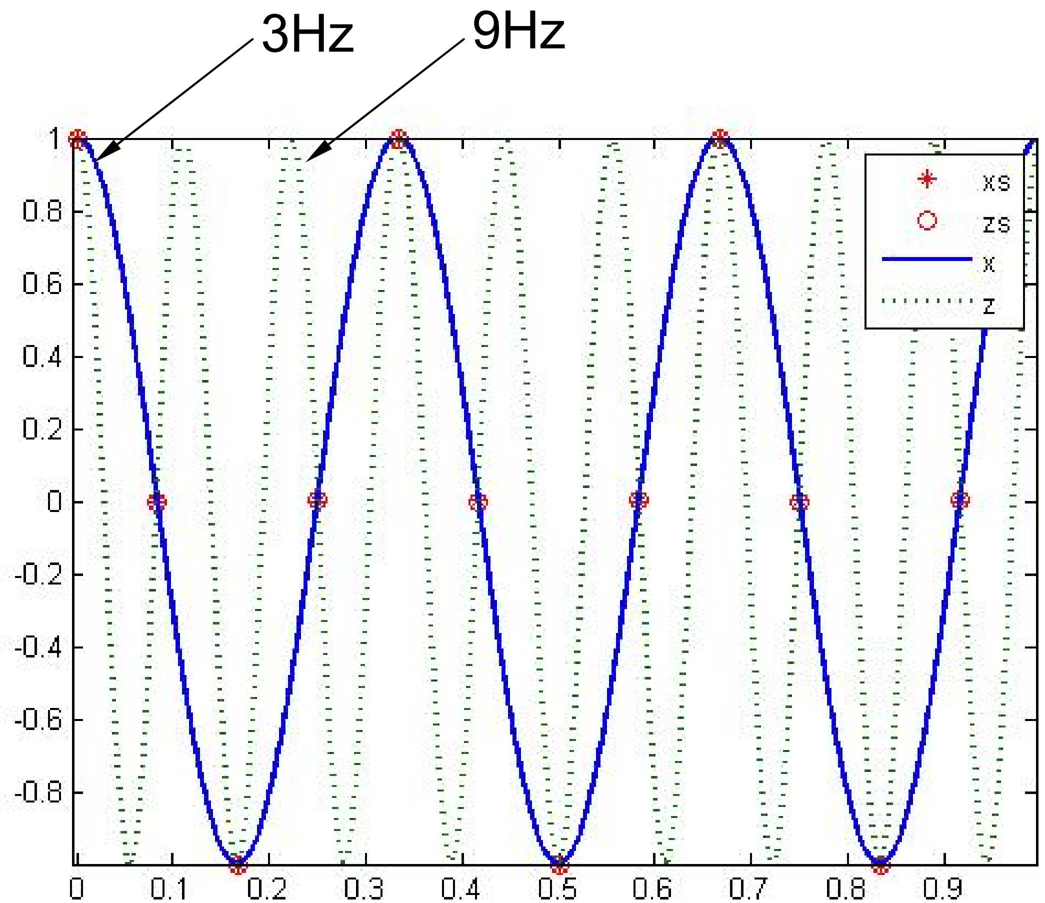

Exercise: Try this example

Sampling



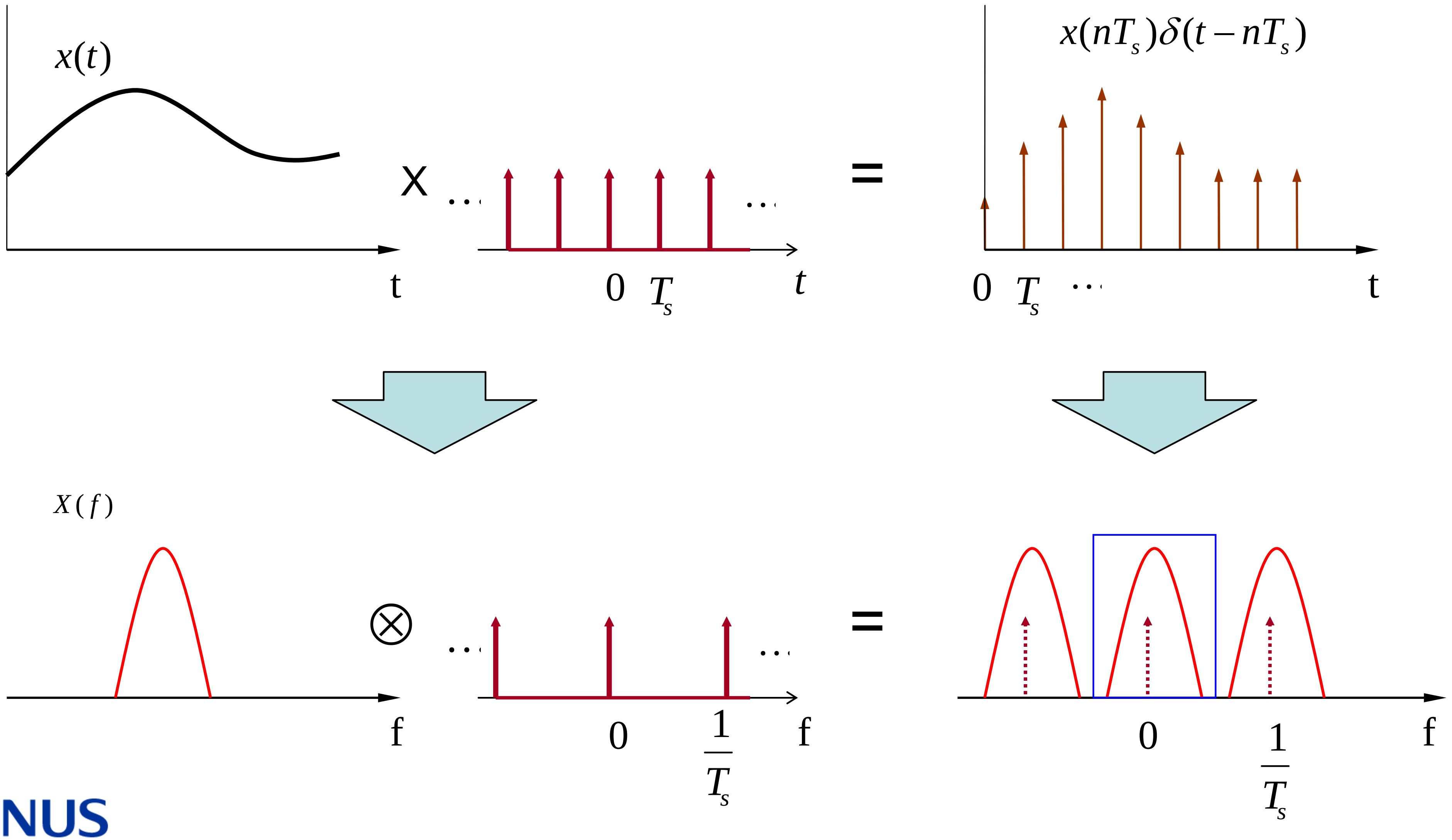
Aliasing: Example

```
f=3; w=2*pi*f;
fs=12;ts=1/fs;

t=0:ts/100:1;
x=cos(w*t);
z=cos((2*pi*fs-w)*t);

nts=0:ts:1;
xs=cos(w*nts);
zs=cos((2*pi*fs-w)*nts);

plot(nts,xs,'r*',nts,zs,'ro')
hold on
plot(t,x,t,z,':')

legend('xs','zs','x','z')
```

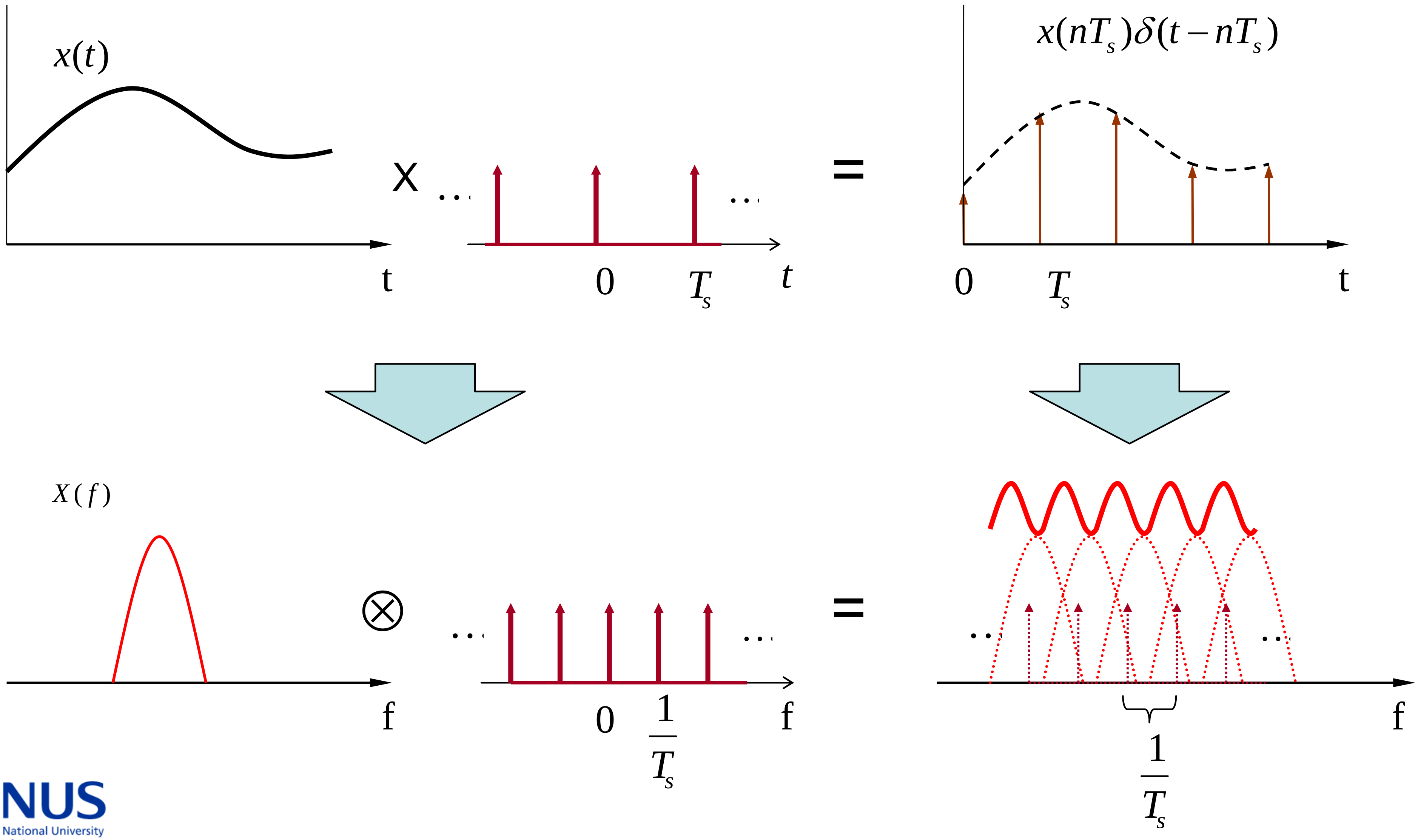


Exercise: Try this example and observe the sampled signals

From Continuous to Discrete



Aliasing

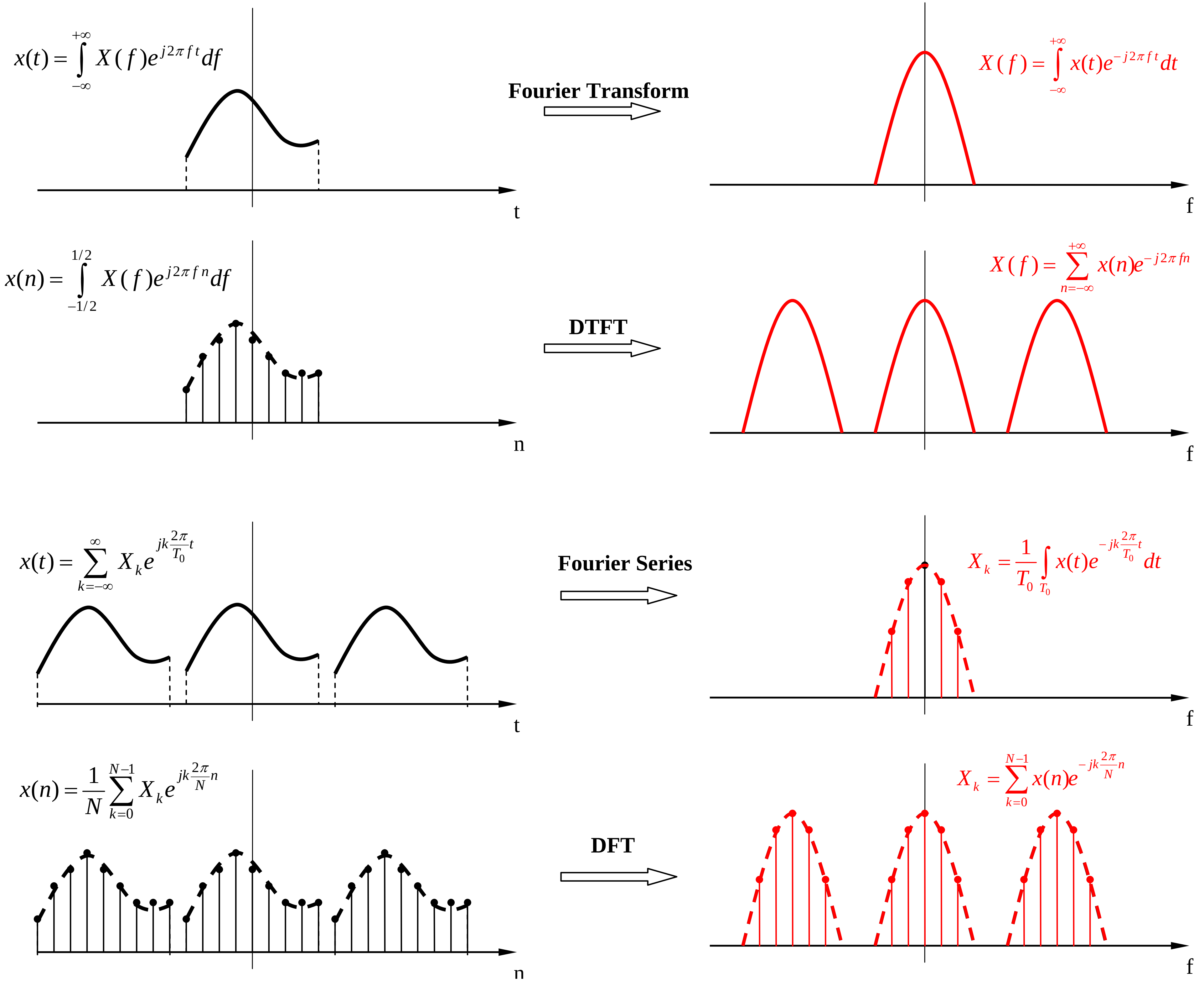


$x(t)$
t
X
0
T_s
t
=
$x(nT_s)\delta(t-nT_s)$
0
T_s
t
$X(f)$
f
$\otimes$
0
$\frac{1}{T_s}$
f
=
$\frac{1}{T_s}$
f

Aliasing

- When sampling is too slow for a signal's band width, high frequency content cannot be observed and it leaks into lower frequencies, thus distorting the signal.

- A fundamental law in signal processing states that the sampling frequency must be at least twice the highest frequency present in the signal.


$x(t) = \int_{-\infty}^{+\infty} X(f) e^{j2\pi f t} df$
Fourier Transform
$X(f) = \int_{-\infty}^{+\infty} x(t) e^{-j2\pi f t} dt$
t
f
$x(n) = \int_{-1/2}^{1/2} X(f) e^{j2\pi f n} df$
DTFT
$X(f) = \sum_{n=-\infty}^{+\infty} x(n) e^{-j2\pi fn}$
n
f
$x(t) = \sum_{k=-\infty}^{\infty} X_k e^{jk\frac{2\pi}{T_0}t}$
Fourier Series
$X_k = \frac{1}{T_0} \int_{T_0} x(t) e^{-jk\frac{2\pi}{T_0}t} dt$
t
f
$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{jk\frac{2\pi}{N}n}$
DFT
$X_k = \sum_{k=0}^{N-1} x(n) e^{-jk\frac{2\pi}{N}n}$
n
f

Discrete Fourier Transform

$$X_k^N = \sum_{n=0}^{N-1} x_n e^{-j\frac{2\pi}{N}kn}, \quad or \quad X = F_N x, \quad (F_N)_{nk} = e^{-j\frac{2\pi}{N}nk}$$

$$F_1 = [1], \quad F_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad F_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}$$

Some properties:

- F is symmetric, $F^T = F$
- $(F^T)^* F = N\,I$
- $F^{-1} = F^*/N$ (inverse transform is obtained by replacing j by $-j$, and dividing by N)

NUS National University of Singapore

Fast Fourier Transform

- Although the DFT is *computable* transform, the straightforward implementation is very *inefficient*, especially when the sequence length N is large.
- In 1965, Cooley and Tukey showed the a procedure to substantially reduce the amount of computations involved in the DFT.
- This led to the *explosion* of applications of the DFT.
- All these efficient algorithms are collectively known as *fast Fourier transform* (FFT) algorithms.

National University of Singapore

Basic Idea of FFT

$$X_k^N = \sum_{n=0}^{N-1} x_n e^{-j\frac{2\pi}{N}kn}$$

$$= \sum_{n=even} x_n e^{-j\frac{2\pi}{N}kn} + \sum_{n=odd} x_n e^{-j\frac{2\pi}{N}kn}$$

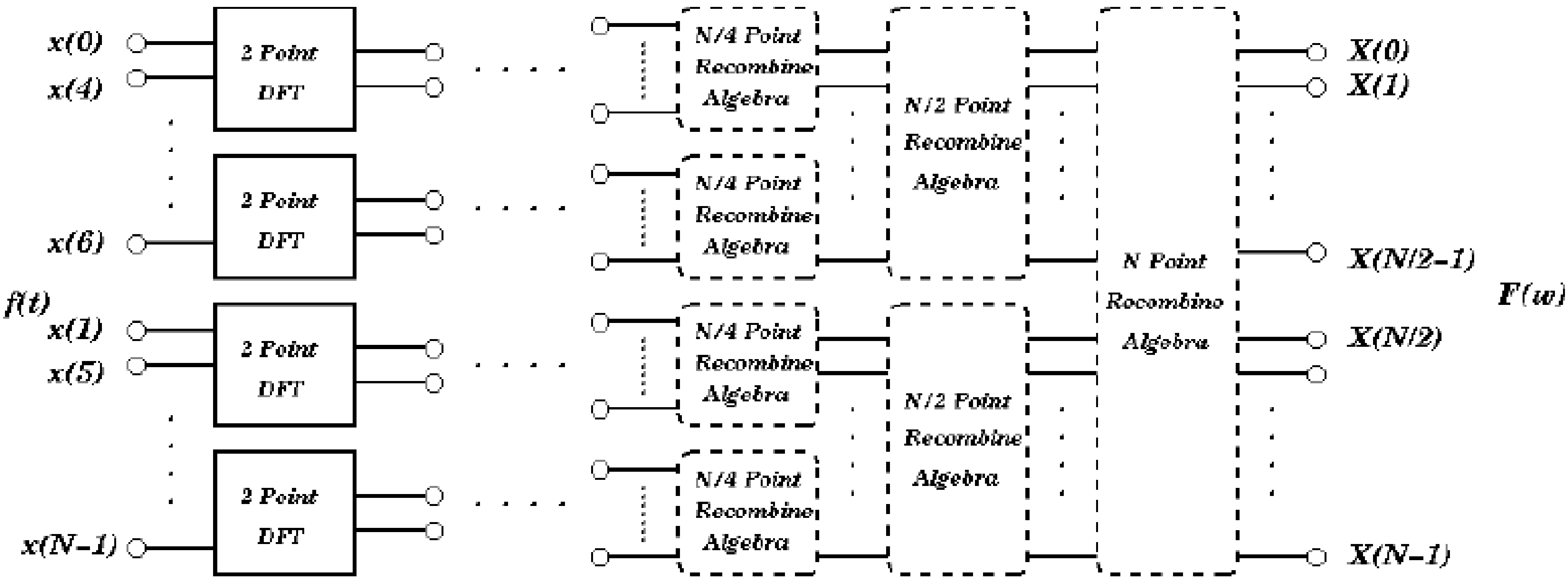
$$= \sum_{m=0}^{N/2-1} x_{2m} e^{-j\frac{2\pi}{N}k(2m)} + \sum_{m=0}^{N/2-1} x_{2m+1} e^{-j\frac{2\pi}{N}k(2m+1)}$$

$$= \sum_{m=0}^{N/2-1} x_{2m} e^{-j\frac{2\pi}{N}k(2m)} + e^{-j\frac{2\pi}{N}k} \sum_{m=0}^{N/2-1} x_{2m+1} e^{-j\frac{2\pi}{N}k(2m)}$$

$$= \sum_{m=0}^{N/2-1} x_m^e e^{-j\frac{2\pi}{N/2}km} + e^{-j\frac{2\pi}{N}k} \sum_{m=0}^{N/2-1} x_m^o e^{-j\frac{2\pi}{N/2}km}$$

NUS National University of Singapore

Basic Idea of FFT



Matlab Implementation

- **Function:** `X = fft(x,N)`
  - **If length(x)< N, x is padded with zeros.**
  - **If the argument N is omitted, N = length(x)**
  - **If x is matrix, fft computes the N-point DFT of each column of x.**

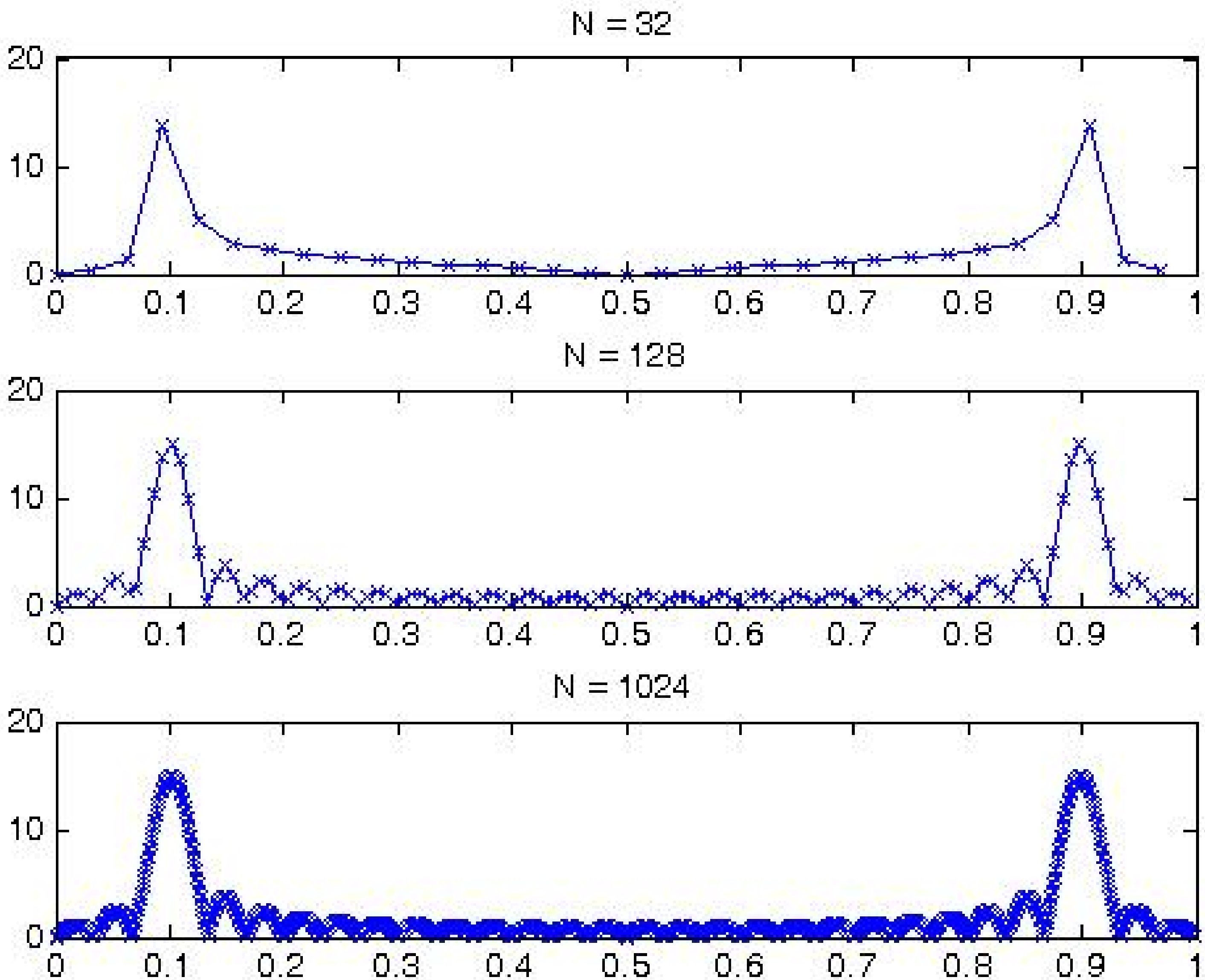
NUS National University of Singapore

FFT Example 1

```
n = [0:29];
x = cos(2*pi*n/10);

N1 = 32; N2 = 128; N3 = 1024;
X1 = abs(fft(x,N1));
X2 = abs(fft(x,N2));
X3 = abs(fft(x,N3));

F1 = [0 : N1 - 1]/N1;
F2 = [0 : N2 - 1]/N2;
F3 = [0 : N3 - 1]/N3;

subplot(3,1,1)
plot(F1,X1,'-x'),title('N = 32'),axis([0 1 0 20])
subplot(3,1,2)
plot(F2,X2,'-x'),title('N = 128'),axis([0 1 0 20])
subplot(3,1,3)
plot(F3,X3,'-x'),title('N = 1024'),axis([0 1 0 20])
```

Exercise: Try this example and check help for `fft`

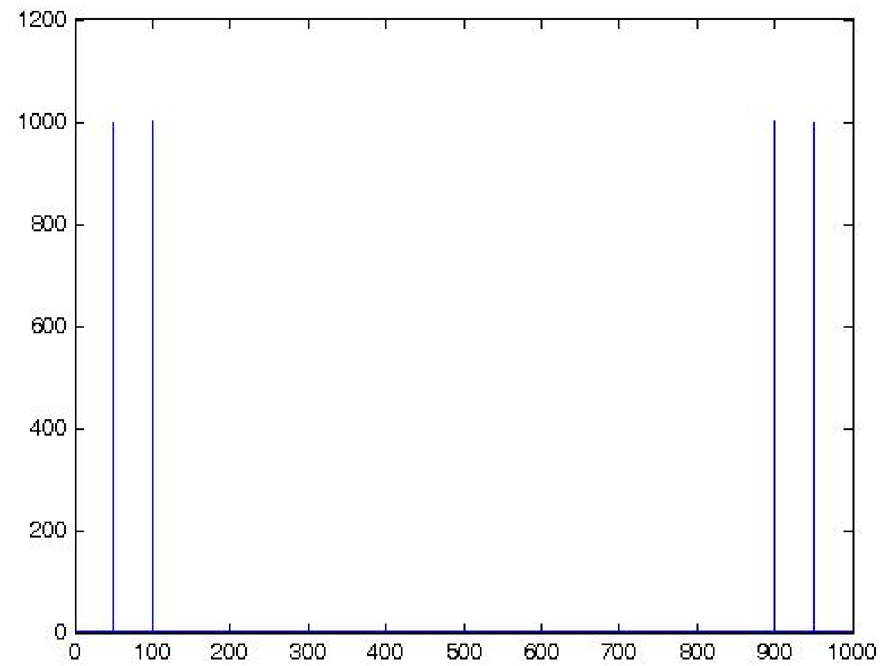
FFT Example 1



FFT Example 2

```
Fs = 1000; % Sampling frequency
t = 0:1/Fs:2-1/Fs; % Time vector

% Signal with two frequencies
y = sin(2*pi*t*50) + sin(2*pi*t*100);

% Frequency vector
f = Fs*(0:length(y)-1)/length(y);

plot(f,abs(fft(y)))
```

Exercise: Try this example

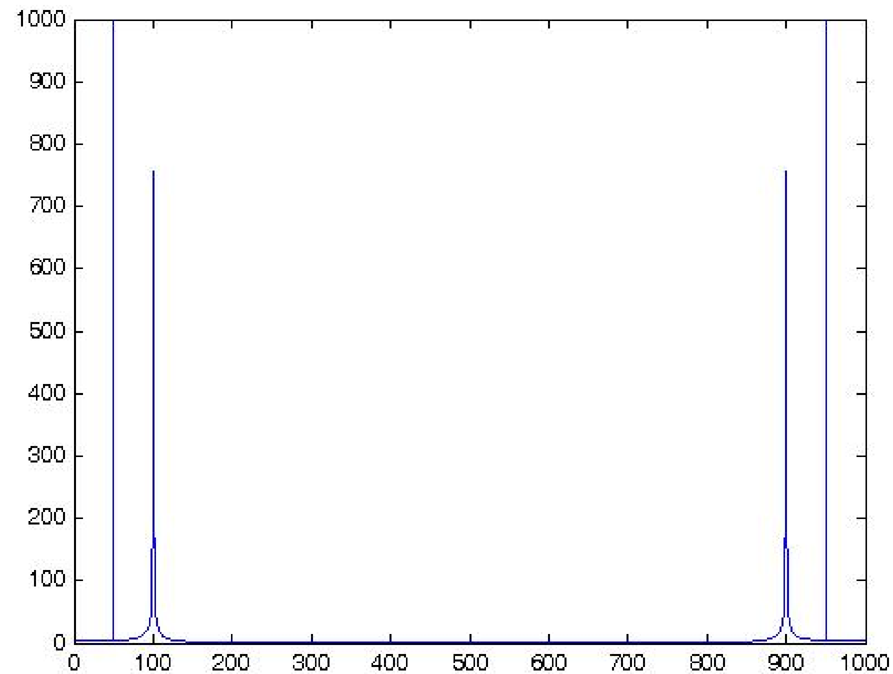


FFT Example 2

```
Fs = 1000; % Sampling frequency
t = 0:1/Fs:2-1/Fs; % Time vector

% Signal with two frequencies
y = sin(2*pi*t*50) + sin(2*pi*t*100.3);

% Frequency vector
f = Fs*(0:length(y)-1)/length(y);

plot(f,abs(fft(y)))
```

Exercise: Try this example

Recall:
Frequency Resolution: Fs/N
Time Resolution: N/Fs



NUS
National University of Singapore

FFT Example 3

Create an example signal:

```
>> Fs = 1000;                       % Sampling frequency
>> T = 1/Fs;                        % Sample time
>> L = 1000;                        % Length of signal
>> t = (0:L-1)*T;                   % Time vector

% Sum of a 50 Hz sinusoid and a 120 Hz sinusoid
>> x = 0.7*sin(2*pi*50*t) + sin(2*pi*120*t);
>> y = x + 2*randn(size(t));        % Sinusoids plus noise

>> plot(Fs*t(1:50),y(1:50))
>> title('Signal Corrupted with Zero-Mean Random Noise')
>> xlabel('time (milliseconds)')
```

National University of Singapore

FFT Example 3

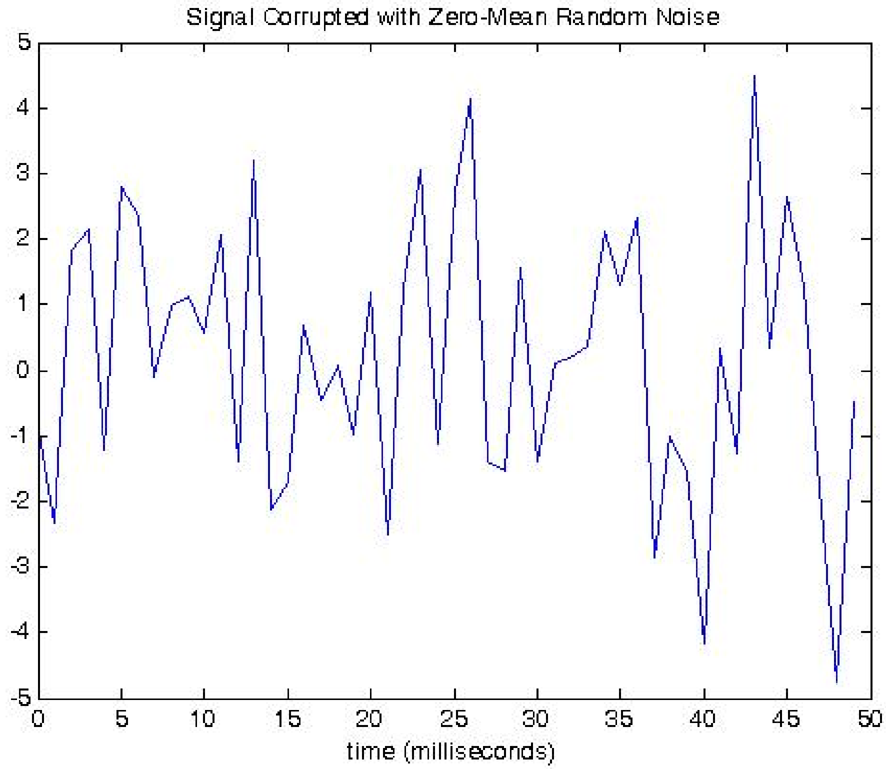
Now find the frequency response using the DFT

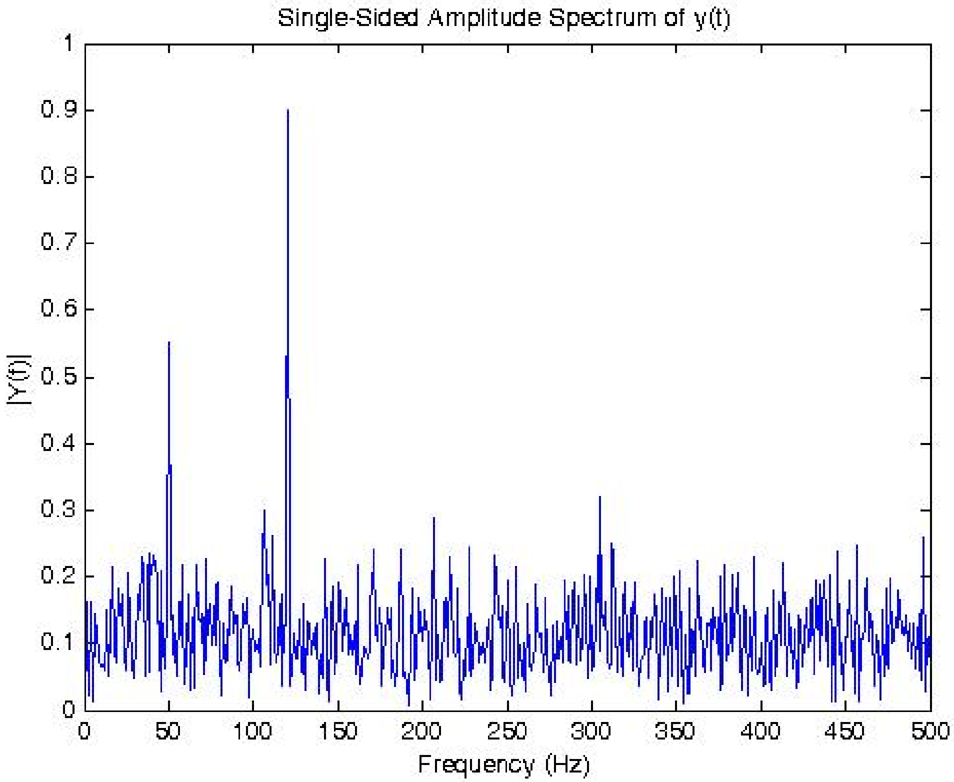
```
>> NFFT = 2^nextpow2(L); % Next power of 2 from length of y
>> Y = fft(y,NFFT)/L;
>> f = Fs/2*linspace(0,1,NFFT/2);

% Plot single-sided amplitude spectrum.
>> plot(f,2*abs(Y(1:NFFT/2)))
>> title('Single-Sided Amplitude Spectrum of y(t)')
>> xlabel('Frequency (Hz)')
>> ylabel('|Y(f)|')
```

Exercise: Try this example

NUS National University of Singapore

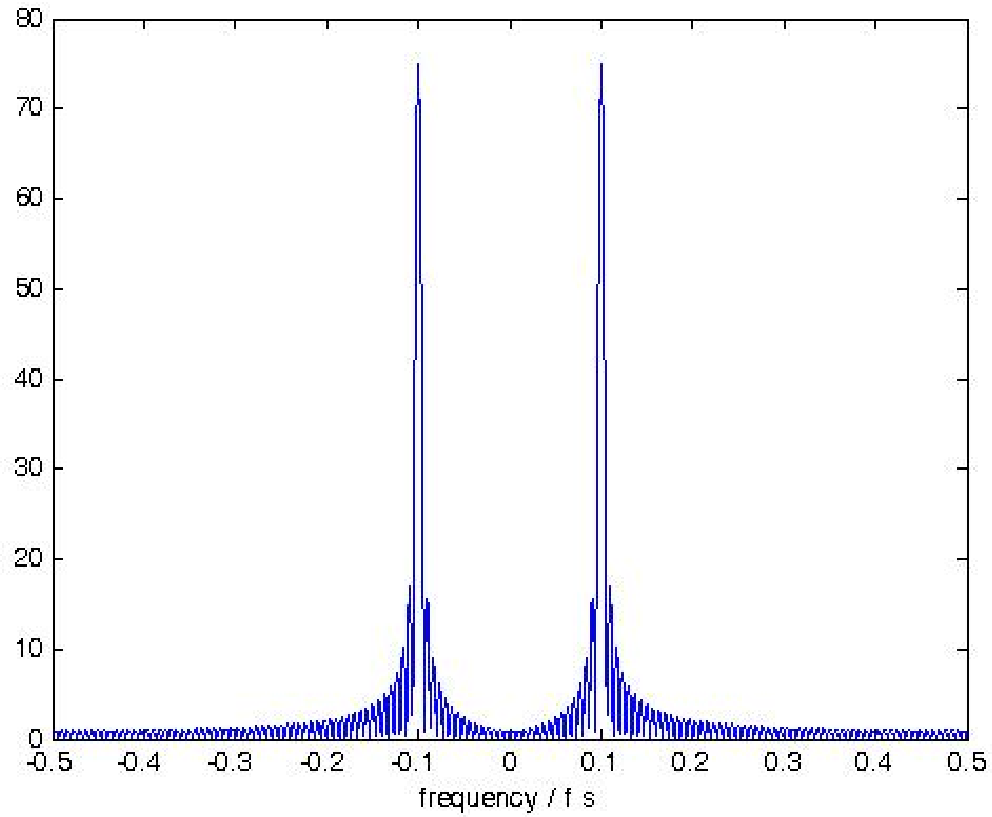
FFT Example 3





FFT Example 4

```
n = [0:149];
x1 = cos(2*pi*n/10);

N = 2048;
X = abs(fft(x1,N));
X = fftshift(X);

F = [-N/2:N/2-1]/N;
plot(F,X),
xlabel('frequency / f s')
```



Exercise: Try this example

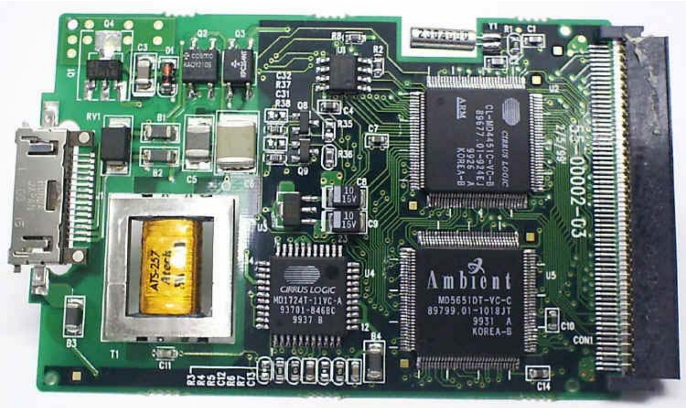
FFT Function Summary

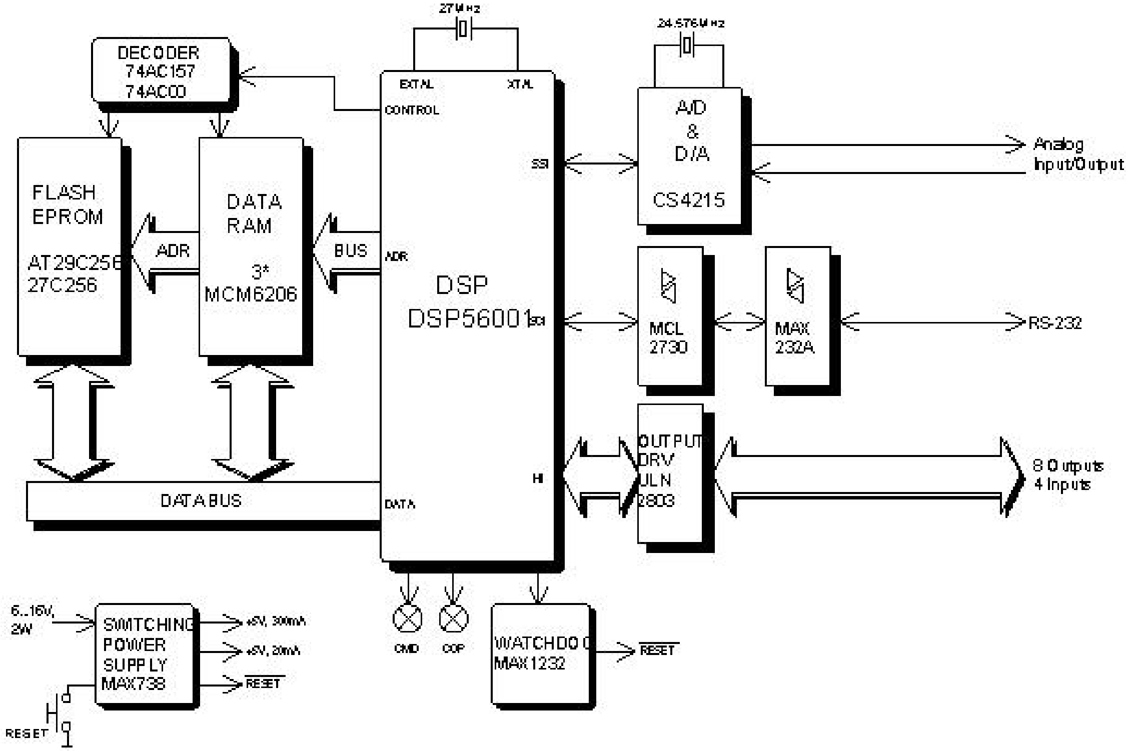
Function	Description
`abs`	Absolute value and complex magnitude
`angle`	Phase angle
`cplxpair`	Sort numbers into complex conjugate pairs
`fft`	One-dimensional discrete Fourier transform, computed with a fast Fourier transform (FFT) algorithm
`fft2`	Two-dimensional discrete Fourier transform
`fftn`	N-dimensional discrete Fourier transform
`fftshift`	Shift DC component of the discrete Fourier transform to the center of spectrum
`ifft`	Inverse one-dimensional discrete Fourier transform
`ifft2`	Inverse two-dimensional discrete Fourier transform
`ifftn`	Inverse N-dimensional discrete Fourier transform
`ifftshift`	Inverse FFT shift
`nextpow2`	Next higher power of 2
`unwrap`	Unwrap phase angle in radians

Filter Design

Digital Filters

A digital filter uses a digital processor to perform numerical calculations on sampled values of the signal. The processor may be a general-purpose computer such as a PC, or a specialized DSP (Digital Signal Processor) chip.





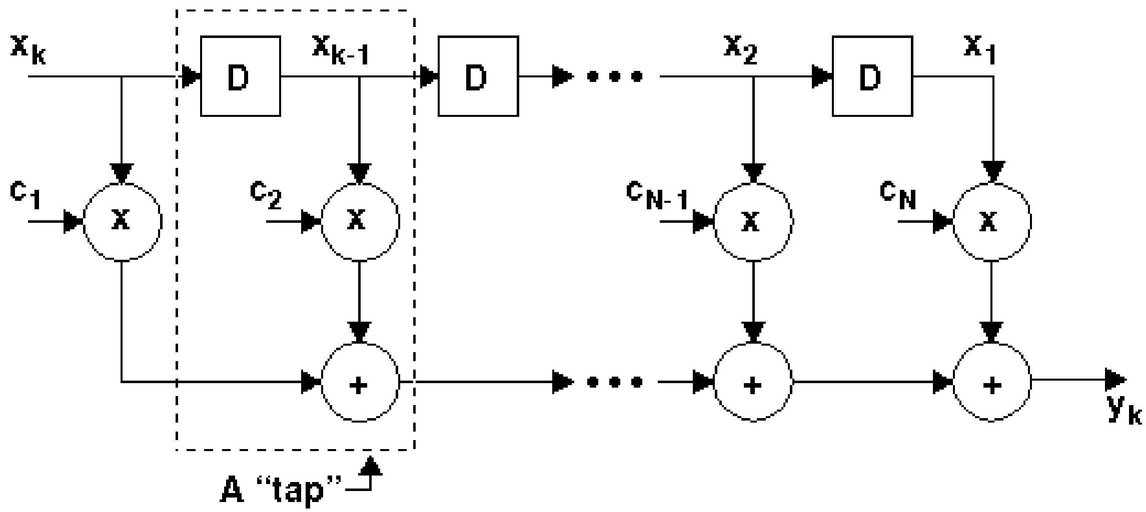
FIR Filters

- Finite impulse response (FIR), i.e., non-recursive equation

$$y[n] = b_0 x[n] + b_1 x[n-1] + \cdots + b_N x[n-N]$$

N is known as the *filter order*; an *N*th-order filter has (*N* + 1) terms on the right-hand side; these are commonly referred to as *taps*.

$$\begin{aligned} H(z) &= Z\{h[n]\} \\ &= \sum_{n=-\infty}^{\infty} h[n] z^{-n} \\ &= \sum_{n=0}^{N} b_n \, z^{-n}. \end{aligned}$$



$$y_k = x_k \times c_1 + x_{k-1} \times c_2 + \cdots + x_2 \times c_{N-1} + x_1 \times c_N$$

FIR Filter Example

- Five-tap discrete-time averaging FIR filter with input *x*[*k*] and output *y*[*k*]

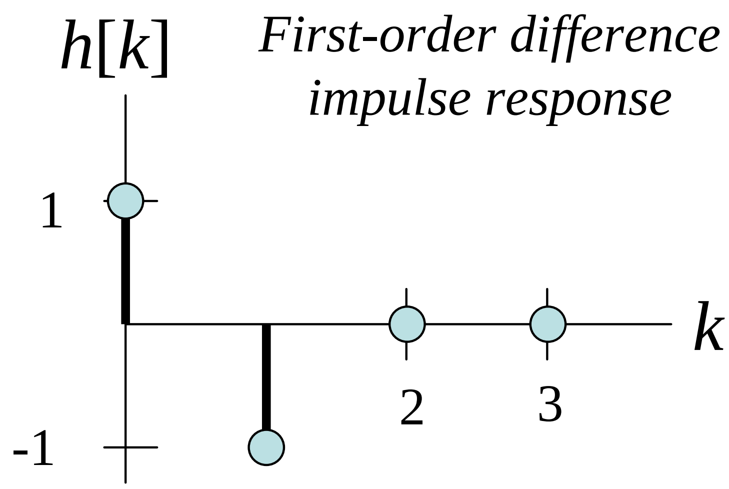
$$y[k] = x[k] + x[k-1] + x[k-2] + x[k-3] + x[k-4]$$

Standard averaging filtering scaled by 5.
Lowpass filter (smooth/blur input signal)
Impulse response is {1, 1, 1, 1, 1}

- First-order difference FIR filter

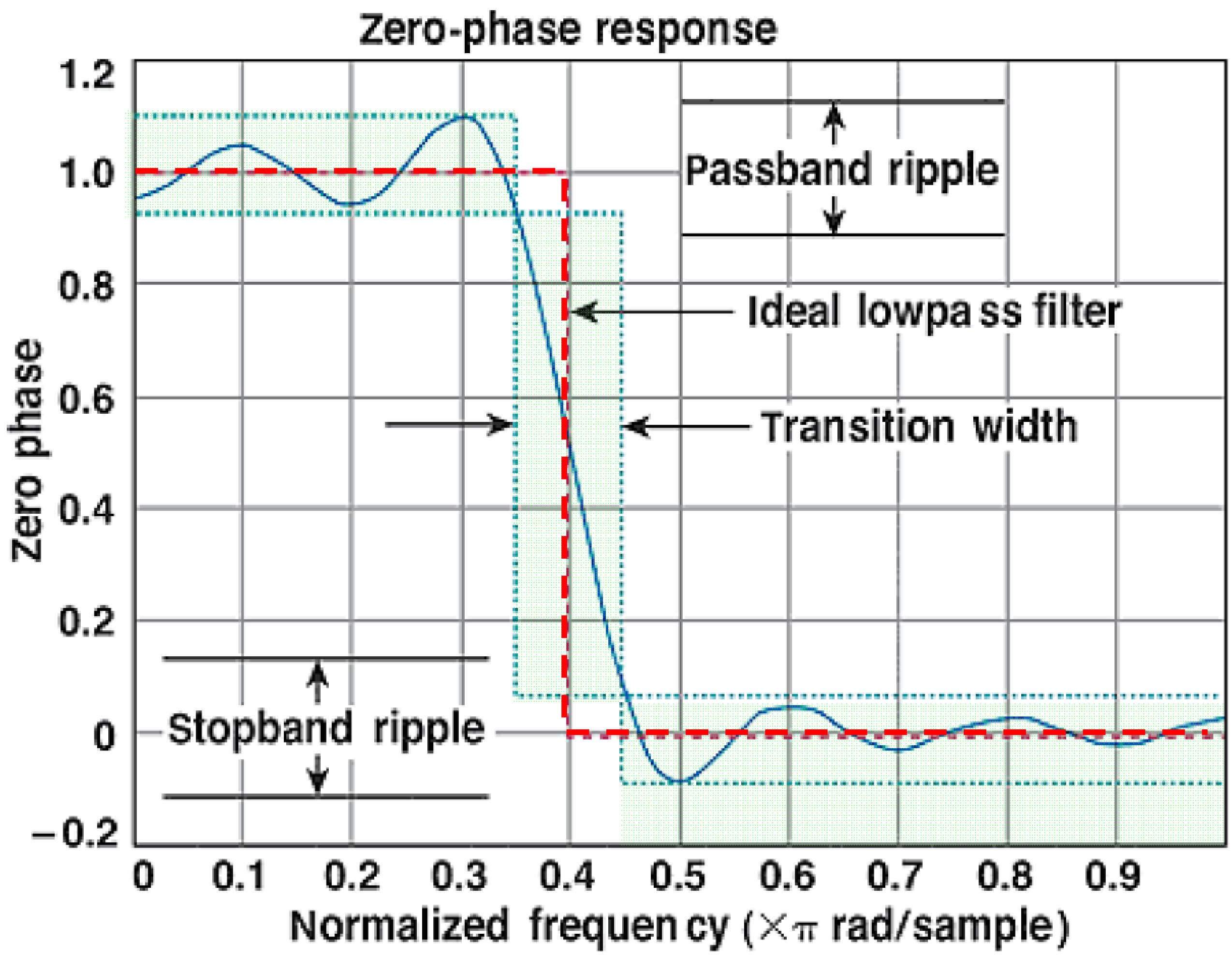
$$y[k] = x[k] - x[k-1]$$

Highpass filter (sharpens input signal)
Impulse response is {1, -1}



Filters Design

To design a filter means to select the coefficients such that the system has specific characteristics.



Matlab: `fir1`

`b = fir1(n,Wn)` returns row vector b containing the n+1 coefficients of an order n low-pass FIR filter with normalized cutoff frequency Wn. The output filter coefficients, b, are ordered in descending powers of z.

$$B(z) = b(1) + b(2)z^{-1} + \cdots + b(n+1)z^{-n}$$

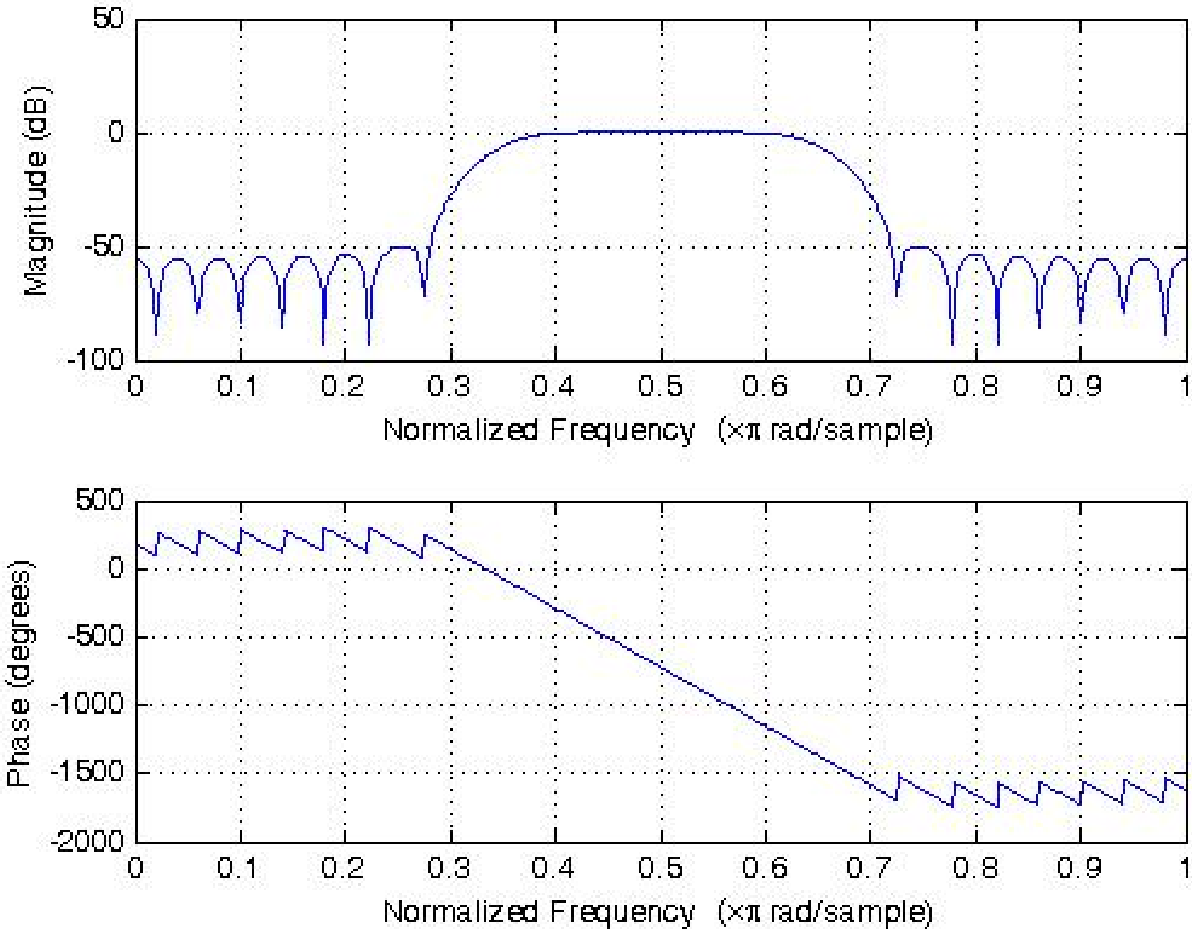
`b = fir1(n,Wn,'ftype')` specifies a filter type, where **`'ftype'`** is:

- **`'high'`** for a highpass filter with cutoff frequency Wn.
- **`'stop'`** for a bandstop filter, if Wn = [w1 w2]. The stopband frequency range is specified by this interval.
- **`'DC-1'`** to make the first band of a multiband filter a passband.
- **`'DC-0'`** to make the first band of a multiband filter a stopband.

FIR Filter Example

Design a 48th-order FIR bandpass filter with passband 0.35 ≤ w ≤ 0.65:

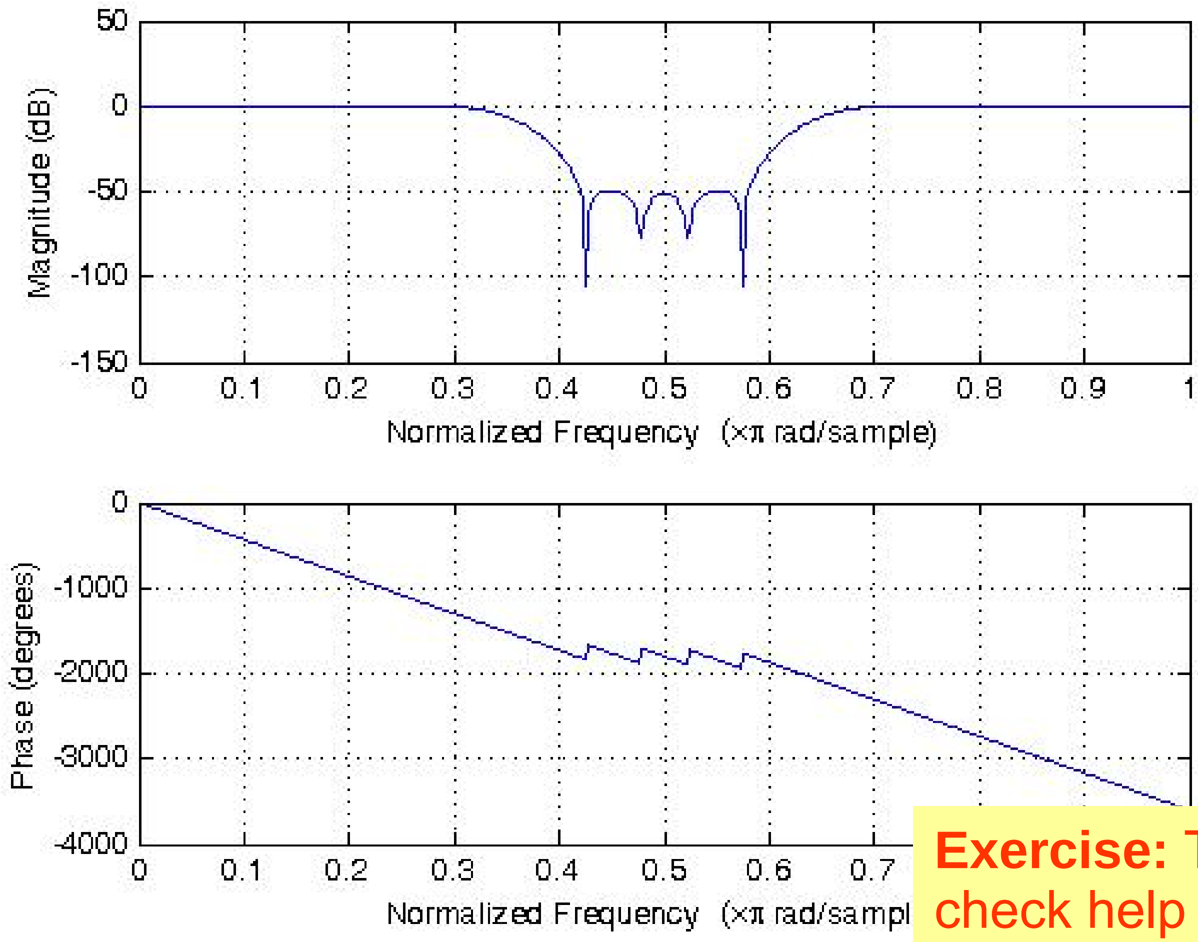
```
>> b = fir1(48,[0.35 0.65]);
>> freqz(b,1)
```



FIR Filter Example

Design a 48th-order FIR bandstop filter with stopband $0.35 \leqslant w \leqslant 0.65$:

```
>> b = fir1(48,[0.35 0.65],'DC-1');
>> freqz(b,1)
```



Exercise: Try these examples and check help for **`fir1, freqz`**

IIR Filters

- Infinite impulse response (IIR), i.e., recursive equation

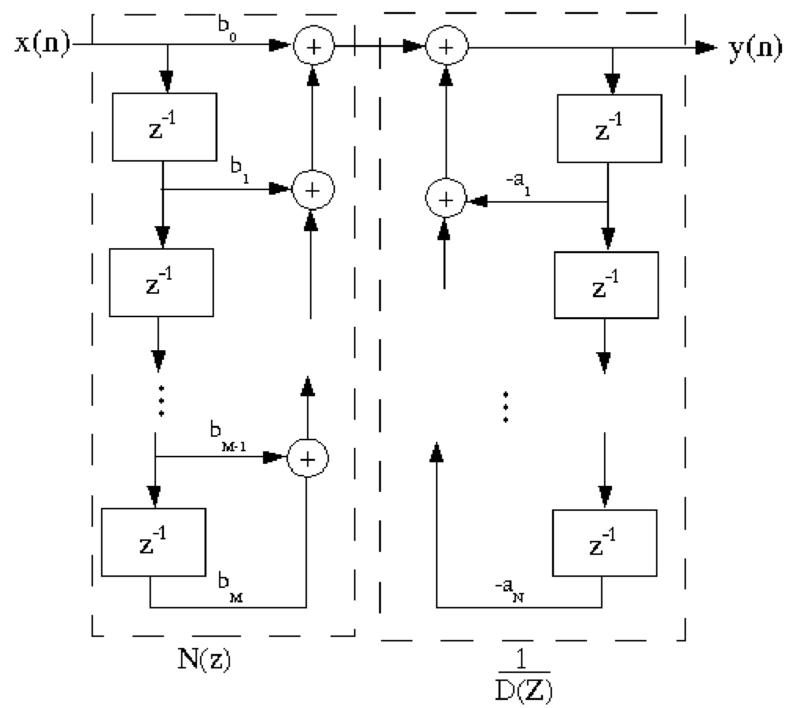
$$y(n) = -(a_1 y(n-1)) - a_2 y(n-2) + \cdots - a_N y(n-N)$$
$$+ b_0 x(0) + b_1 x(n-1) + \cdots + b_M x(n-M)$$

$$H(z) = \frac{N(z)}{D(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_M z^{-M}}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_N z^{-N}}$$

For Example,

$$y[n] - \frac{1}{2} y[n-1] = x[n]$$

$$h[n] = \left(\frac{1}{2}\right)^n u[n]$$



Matlab: `butter`

Butterworth analog and digital filter design

Syntax

```
[z,p,k] = butter(n,Wn,'ftype','s')
[b,a] = butter(n,Wn,'ftype','s')
[A,B,C,D] = butter(n,Wn,'ftype','s')
```

Input

n – filter order

Wn – cutoff frequency

ftype – filter type: high, low, stop

s – analog filter

Output

[z,p,k] – pole-zero-gain form

$$W(z) = \frac{k(z-z_1)(z-z_2)\ldots(z-z_m)}{(z-p_1)(z-p_2)\ldots(z-p_n)}$$

[b,a] – transfer function form

$$W(z) = \frac{b_{m-1} + b_{m-2}z^{-1} + \ldots + b_2 z^{m-2} + b_1 z^{m-1}}{a_{n-1} + a_{n-2}z^{-1} + \ldots + a_2 z^{n-2} + a_1 z^{n-1}}$$

[A,B,C,D] – state space form

$$\begin{cases} x(n+1) = Ax(n) + Bu(n) \\ y(n) = Cx(n) + Du(n) \end{cases}$$

NUS National University of Singapore

Butterworth Filter Design

Analog low pass Filter

Design a 3rd-order low pass Butterworth filter with cutoff frequency of 300 Hz:

```
>> [b,a] = butter(2,300,'low','s');
>> freqz(b,a)
```

Digital Highpass Filter

For data sampled at 1000 Hz, design a 5th-order highpass Butterworth filter with cutoff frequency of 300 Hz, which corresponds to a normalized value of 0.6:

```
>> [z,p,k] = butter(5,300/500,'high');
>> [b,a] = zp2tf(z,p,k);
>> freqz(b,a)
```

To implement: assume that you have input data vector x

```
>> y = filter(b, a, x );
```

NUS National University of Singapore

Exercise: Try these examples and check help for **`butter, filter`**

We reviewed…

- Fourier series and Fourier transform
- Sampling and aliasing
- DTFT, DFT, and FFT
- How to do frequency analysis in Matlab
- Filter design in Matlab