Mobile Systems

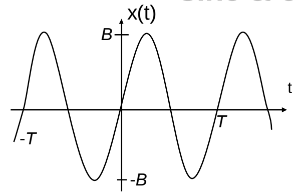
Lecture 3

Frequency-domain processing

COMP28512

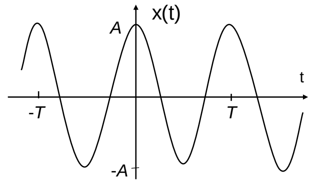
Steve Furber & Barry Cheetham

sine & cosine waves



x(t) = $B\sin(2\pi Ft)$

$T=1/F$



x(t) = $A\cos(2\pi Ft)$

$T=1/F$

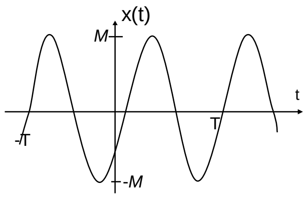
$x(t) = A\times\cos(2\pi Ft) + B\times\sin(2\pi Ft)$

$= M\times\cos(2\pi Ft + \phi)$ where $T=1/F$

$= M\times\cos(2\pi F(t - D))$,

$M = \sqrt{(A^2+B^2)}$, $\phi = \mathrm{atan2}(B,A) = \tan^{-1}(B/A)$;

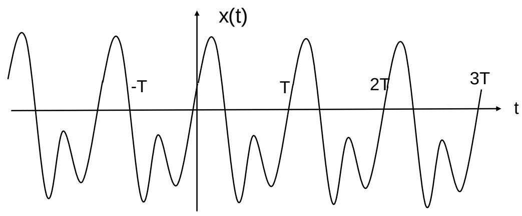
$D = -\phi / (2\pi F)$ = some delay (s)



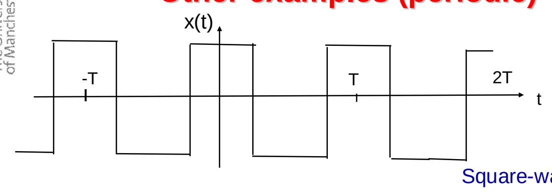
Adding a sine & cosine of same frequency produces a sine-wave of the same frequency delayed by *D* s. Call it a 'sinusoid'

Periodic wave-form

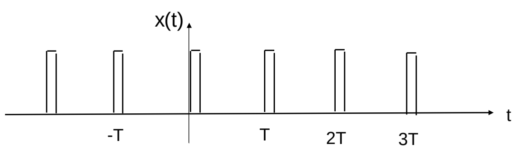
- Any wave-form that repeats itself at time-intervals of *T*.
- *T* is the period. 1/*T* is the fundamental frequency



Other examples (periodic)

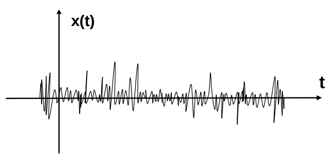


Square-wave



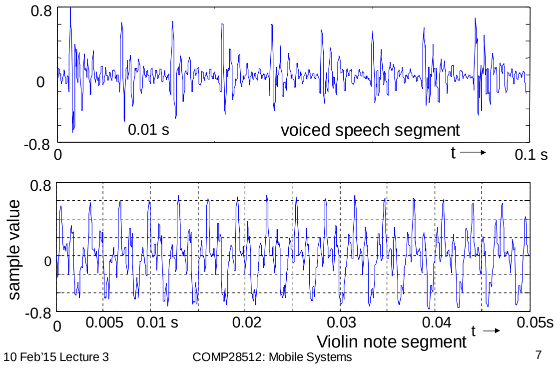
Pulse-stream

Non-periodic wave-form



- e.g. Recording of 'white noise', the sea, unvoiced speech,
- Or a series of random numbers applied to a D to A converter.
- Sound pressure becomes 'random'.

Pseudo-periodic wave-forms



Pseudo-periodicity

- Speech & music waveforms are changing constantly.
- They are not purely periodic
  (They would be pretty boring if they were).
- But if we take a small segment: say 1/20 or 1/50 s, they can be considered periodic.
- They are approximately 'short term periodic' i.e. pseudo-periodic.

Fourier series

- A periodic x(t) with fundamental freq $\mathbf{F}$ can be written as:

$$x(t) = A_0/2 + A_1\cos(2\pi \mathbf{F} t) + B_1\sin(2\pi \mathbf{F} t)$$
$$+ A_2\cos(2\pi(\mathbf{2F})t) + B_2\sin(2\pi(\mathbf{2F})t)$$
$$+ A_3\cos(2\pi(\mathbf{3F})t) + B_3\sin(2\pi(\mathbf{3F})t)$$
$$+ \ \ldots$$

'cos & sin' form

Jean-Baptiste-Joseph Fourier (1768-1830)

Fourier series (re-expressed)

- A periodic x(t) with fundamental freq F can be written as:

$$x(t) = M_0/2 + M_1\cos(2\pi F t + \phi_1)$$
$$+ M_2\cos(2\pi(2F)t + \phi_2)$$
$$+ M_3\cos(2\pi(3F)t + \phi_3)$$
$$+ \ \ldots$$

'Harmonics'

- 'Modulus & phase' form

$$x(t) = M_0/2 \ + \ \sum_{k=1}^{\infty} M_k \cos(2\pi(kF)t + \phi_k)$$

- First harmonic (with k=1) is called the 'fundamental'

Sampled Fourier Series

- Cannot sample the whole infinite series (aliasing)
- Need to remove all sinusoids of freq $\geq Fs/2$

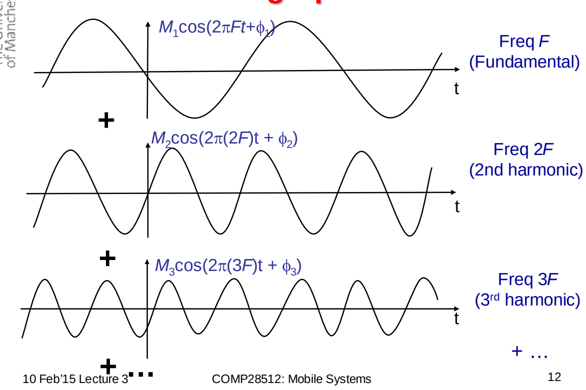
$$x(t) = M_0/2 \ + \ \sum_{k=1}^{N/2-1} M_k \cos(2\pi(kF)t + \phi_k)$$

- where N is even & $(N/2 - 1)\times F < Fs/2$.
- Take Fs samples/second to obtain an array of samples x:

$$x[n] = M_0/2 \ + \ \sum_{k=1}^{N/2-1} M_k \cos(2\pi(kF)n/Fs + \phi_k)$$

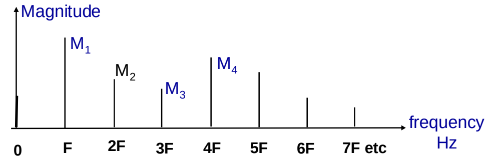
for n = 0,1,2, …, *size(x)*

In graphical form



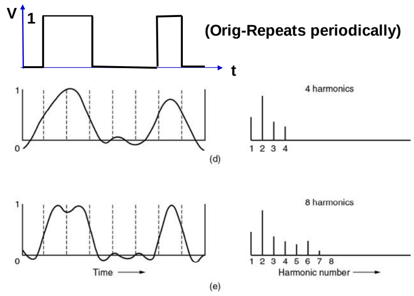
Line spectrum

- Represent magnitudes of FSeries sine-waves graphically.



- Tells us which harmonics are present & how strong they are.
- Represents a periodic signal in the 'frequency-domain'.

Fourier series for a waveform



- The more harmonics we take, the closer it gets
- With 8 harmonics, it starts to look like the original wave

Some questions

1. What would a line-spectrum for a 100 Hz sine-wave of amplitude 0.8 look like ?
2. What is the Fourier series for 'white noise'
3. Do the phases (ϕ_1, ϕ_2, ϕ_3, …) matter?

Sampled Fourier series & complx numbers

$e^{j\theta} = \cos(\theta) + j\times\sin(\theta)$ where $j = \sqrt{(-1)}$ or 'i'

$e^{-j\theta} = \cos(\theta) - j\times\sin(\theta)$

$e^{j\theta} + e^{-j\theta} = 2\times\cos(\theta)$

$$\cos(2\pi(kF)t+\phi_k) = \left(e^{j(2\pi(kF)t+\phi_k)} + e^{-j(2\pi(kF)t+\phi_k)}\right)/2$$

Sampled Fourier series & complx numbers

$e^{j\theta} = \cos(\theta) + j\times\sin(\theta)$ where $j = \sqrt{(-1)}$

$e^{-j\theta} = \cos(\theta) - j\times\sin(\theta)$

$e^{j\theta} + e^{-j\theta} = 2\times\cos(\theta)$

$$\cos(2\pi(kF)t+\phi_k) = \left(e^{j(2\pi(kF)t+\phi_k)} + e^{-j(2\pi(kF)t+\phi_k)}\right)/2$$

$$\therefore\ x[n] = (1/2)\sum_{k=-(N/2-1)}^{N/2-1} M_k e^{j(2\pi(kF)n/Fs+\phi_k)}$$

if we define $\phi_k = 0$ & $M_{-k} = M_k$ and $\phi_{-k} = -\phi_k$ for all k

Sampled Fourier series & complx numbers

$e^{j\theta} = \cos(\theta) + j\times\sin(\theta)$ where $j = \sqrt{(-1)}$

$e^{-j\theta} = \cos(\theta) - j\times\sin(\theta)$

$e^{j\theta} + e^{-j\theta} = 2\times\cos(\theta)$

$$\cos(2\pi(kF)t+\phi_k) = \left(e^{j(2\pi(kF)t+\phi_k)} + e^{-j(2\pi(kF)t+\phi_k)}\right)/2$$

$$\therefore\ x[n] = (1/2)\sum_{k=-(N/2-1)}^{N/2-1} M_k e^{j(2\pi(kF)n/Fs+\phi_k)}$$

$$= (1/2)\sum_{k=-(N/2-1)}^{N/2-1} M_k e^{j\phi_k}\ e^{2\pi j(kF)n/Fs}$$

With $\phi_k = 0$ & $M_{-k} = M_k$ and $\phi_{-k} = -\phi_k$ for all k

Sampled Fourier series & complx numbers

$e^{j\theta} = \cos(\theta) + j\times\sin(\theta)$ where $j = \sqrt{(-1)}$

$e^{-j\theta} = \cos(\theta) - j\times\sin(\theta)$

$e^{j\theta} + e^{-j\theta} = 2\times\cos(\theta)$

$$\cos(2\pi(kF)t+\phi_k) = \left(e^{j(2\pi(kF)t+\phi_k)} + e^{-j(2\pi(kF)t+\phi_k)}\right)/2$$

$$\therefore\ x[n] = (1/2)\sum_{k=-(N/2-1)}^{N/2-1} M_k e^{j(2\pi(kF)n/Fs+\phi_k)}$$

$$= (1/2)\sum_{k=-(N/2-1)}^{N/2-1} \underbrace{M_k e^{j\phi_k}}_{X[k]} e^{2\pi j(kF)n/Fs}$$

We defined $\phi_k = 0$ & $M_{-k} = M_k$ and $\phi_{-k} = -\phi_k$ for all k

This means that X[-k] = complx conj of X[k].

Sampled Fourier series (more neatly)

- To avoid negative values of k, invent values of M_k and ϕ_k for $k \geq N/2$.
- Define: $M_{N/2+k} = M_{N/2-k}$ and $\phi_{N/2+k} = \phi_{N/2-k}$ for k=1,2,…,N/2-1
- Also allow X[N/2] to be some real number (often zero).
- Then X[N/2+k] = complx conj of X[N/2-k] for k=0,1,2, …
- And sampled F Series is more neatly expressed as:

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]e^{2\pi j(kF)n/Fs}$$

N-point Inverse-DFT

- Set $F = Fs/N$

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]e^{2\pi j(k/N)n}$$

- Have just invented the 'inverse Discrete Fourier Transform'.
- If $Fs = 80$ Hz & $N = 10$, it expresses a signal x[n] as the sum of sinusoids of frequencies:

  0, F, 2F, 3F, 4F

  i.e. 0, 8, 16, 24, 32 Hz
- With N points, we only get N/2 – 1 sinusoids & a 'dc term'

Example

- Assume N=8, and:

$X_0 = 1$ (for freq 0 (dc value): always real).

$X_1 = 5.0e^{0.93j} = 3 + 4j$

$X_2 = 4.12e^{-1.33j} = 1 - 4j$

$X_3 = 1.4e^{3\pi j/4} = -1 + j$

$X_4 = 0$ (for freq $Fs/2$: must be real)

(X_{-3}) $X_5 = 1.4e^{-3\pi j/4} = -1 - j$

(X_{-2}) $X_6 = 4.12e^{1.33j} = 1 + 4j$

(X_{-1}) $X_7 = 5.0e^{-0.93j} = 3 - 4j$

The 'neat' formula has introduced $X_{N/2}$. Set it to zero.

Discrete Fourier Transform (DFT)

- Mathematically, it may be shown that if

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]\exp\left(j(2\pi(k/N)n)\right) \quad \text{for n} = 0, 1, 2, ..., \text{N -1}$$

then, $$X[k] = \sum_{n=0}^{N-1} x[n]\exp\left(-j(2\pi(k/N)n)\right) \quad \text{for k} = 0,1,..., \text{N -1}$$

- This new equation is the 'N-point' DFT.
- Converts a sampled waveform segment $\{x[n]\}_{0,N-1}$ into a Fourier series representation.
- Converts from time-domain to frequency-domain.

'N-pt' DFT & inverse-DFT

- As we have seen, the 'N-pt' DFT is:

$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-j(2\pi kn/N)} \quad \text{for k} = 0,1,..., \text{N -1}$$

- From the frequency-dom representation, we can always go back to the time-domain by:

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]e^{j(2\pi kn/N}} \quad \text{for n} = 0, 1, 2, ..., \text{N -1}$$

- This was on previous slide, & it is the 'n-pt' inverse-DFT

DFT & inverse-DFT (in short)

- DFT converts $\{x[n]\}_{0,N-1}$(time-dom) to $\{X[k]\}_{0,N-1}$ (freq-dom)

$$X[k] = \sum_{n=0}^{N-1} x[n]\exp(-j(2\pi kn/N)) \quad \text{for k} = 0,1,\ldots,\text{N-1}$$

- IDFT converts $\{X[k]\}_{0,N-1}$(freq-dom) to $\{x[n]\}_{0,N-1}$ (time-dom)

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]\exp(j(2\pi kn/N)) \quad \text{for n} = 0, 1, 2, \ldots, \text{N-1}$$

- Note similarities & differences betw DFT & inverse-DFT.

10 Feb'15 Lecture 3 COMP28512: Mobile Systems 25

DFT & complex numbers

- In freq-dom, each X[k] has real & imaginary parts.
- Difficult to interpret,
- We usually convert to modulus & phase form:

$$M_k = \sqrt{real(X_k)^2 + imag(X_k)^2} \qquad \phi_k = \text{atan2}(imag(X_k), real(X_k))$$

- Time-dom signal x[n] is usually real.
- Strangely, we make it complex with zero imag part.
- Result of DFT usually complex, even when signal is real.

10 Feb'15 Lecture 3 COMP28512: Mobile Systems 26

'Direct' DFT in Pseudo-code

- Transforms $\{x[n]\}_{0,N-1}$ to $\{X[k]\}_{0,N-1}$

```
for k=0:N-1,
  Xre(k)=0;  Xim(k)=0 ;
  W= k * 2.0 * pi / N ;
  for n=0:N-1,
    Xre(k) = Xre(k) + xre(n)*cos(n*W) +xim(n)*sin(n*W);
    Xim(k) = Xim(k) + xim(n)*cos(n*W) -xre(n)*sin(n*W);
  end;
end;
```

10 Feb'15 Lecture 3 COMP28512: Mobile Systems 27

'Direct' inverse-DFT

- Transforms $\{X[k]\}_{0,N-1}$ to $\{x[n]\}_{0,N-1}$
- Small letters for time-dom, large letters for freq-dom

```
for n=0:N-1,
  xre(n)=0;  xim(n)=0 ;
  W= - n * 2.0 * pi / N ;
  for k=0:N-1,
    xre(n) = xre(n) + Xre(k)*cos(k*W) + Xim(k)*sin(k*W);
    xim(n) = xim(n) + Xim(k)*cos(k*W) – Xre(k)*sin(k*W);
  end;
  xre(n) = xre(n)/N;  xim(n)=xim(n)/N;
end;
```

10 Feb'15 Lecture 3 COMP28512: Mobile Systems 28

Comment on 'direct' versions

- Presented only to show how simple the equations are.
- Even simpler if we used NumPy's complex arith
- Direct versions are much too slow.
- Instead we use much faster FFT:

  X = np.fft.fft(x)
- x can be a complex array, but is normally purely real.
- X will normally be complex
- Inverse-FFT gives complex array x

  x = np.fft.ifft(X)
- If you have been careful, imag part of x will be zero.

10 Feb'15 Lecture 3 COMP28512: Mobile Systems 29

Example

- Apply FFT to {1 -2 3 4 5 -6 7 8}
- Real pt:

  20 1.66 -4 -9.66 12 -9.66 -4 1.66
- Imag pt:

  0 4 20 -4 0 4 -20 -4
- Always get this 'mirror' property for a real signal.
- Normally, only look at the first 5 terms (0,1,…,N/2)
- If we want to use inverse DFT to get a real signal, we must make sure the 'mirror' is correct.

10 Feb'15 Lecture 3 COMP28512: Mobile Systems 30

Uses of DFT & its inverse

1. Spectral analysis:
   - finding out what signal components are present
2. Signal processing:
   - some processes best done by converting to freq-dom first
   - convert back to time-domain after the processing
   - e.g. filtering out an unwanted sine-wave

Matrix representation (N=6)

$$X = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & e^{-2\pi j/N} & e^{-4\pi j/N} & e^{-6\pi j/N} & e^{-8\pi j/N} & e^{-10\pi j/N} \\ 1 & e^{-4\pi j/N} & e^{-8\pi j/N} & e^{-12\pi j/N} & e^{-16\pi j/N} & e^{-20\pi j/N} \\ 1 & e^{-6\pi j/N} & e^{-12\pi j/N} & e^{-18\pi j/N} & e^{-24\pi j/N} & e^{-30\pi j/N} \\ 1 & e^{-8\pi j/N} & e^{-16\pi j/N} & e^{-24\pi j/N} & e^{-32\pi j/N} & e^{-40\pi j/N} \\ 1 & e^{-10\pi j/N} & e^{-20\pi j/N} & e^{-30\pi j/N} & e^{-40\pi j/N} & e^{-50\pi j/N} \end{bmatrix} \times \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ x[3] \\ x[4] \\ x[5] \end{bmatrix}$$

Test in Pseudo-code

```
% DFT in matrix form
xre=[1; -2; 3; 4; 5; -6; 7; 8]
for I=0:N-1
  for J=0:N-1,
    G(I,J) =  exp(complex(0, -2*pi*I*J/N ));
   end;
end;
XM = G*xre
```


Equivalent representation (*N*=6)

- **Noting that when *N*=6,**

$$e^{-12\pi j/N} = \cos(-2\pi) + j\sin(-2\pi) = 1$$

$$\text{and} \quad e^{-6\pi j/N} = \cos(-\pi) + j\sin(-\pi) = -1$$

it may be shown that:

$$X = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & e^{-2\pi j/N} & -e^{2\pi j/N} & -1 & -e^{-2\pi j/N} & e^{2\pi j/N} \\ 1 & -e^{2\pi j/N} & -e^{-2\pi j/N} & 1 & -e^{2\pi j/N} & -e^{-2\pi j/N} \\ 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & -e^{-2\pi j/N} & -e^{2\pi j/N} & 1 & -e^{-2\pi j/N} & -e^{2\pi j/N} \\ 1 & +e^{2\pi j/N} & -e^{-2\pi j/N} & -1 & -e^{2\pi j/N} & e^{-2\pi j/N} \end{bmatrix} \times \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ x[3] \\ x[4] \\ x[5] \end{bmatrix}$$

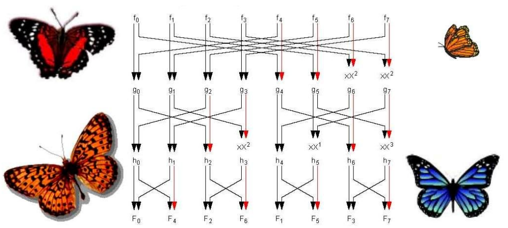
Simplification (*N*=6)

$$X = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & e^{-2\pi j/N} & 0 & -e^{2\pi j/N} \\ 1 & 0 & -e^{2\pi j/N} & 0 & -e^{-2\pi j/N} & 0 \\ 0 & 1 & 0 & -1 & 0 & 1 \\ 1 & 0 & -e^{-2\pi j/N} & 0 & -e^{2\pi j/N} & 0 \\ 0 & 1 & 0 & e^{2\pi j/N} & 0 & -e^{-2\pi j/N} \end{bmatrix} \times \begin{bmatrix} x[0]+x[3] \\ x[0]-x[3] \\ x[1]+x[4] \\ x[1]-x[4] \\ x[2]+x[5] \\ x[2]-x[5] \end{bmatrix}$$

- Computation simplified into 9 ‘butterfly’ calculations.
- Can even continue this process for further simplification

Fast Fourier Transform (FFT)

- Take advantage of structure, using, classic “butterfly”:



   - coefficients are precomputed and held in a table
   - This diag is for N=8 with inputs $\{f_0, f_1, \ldots, f_7\}$

MANCHESTER 1824

The University of Manchester

Fast Fourier Transform (FFT)

- Gives 'exactly' same results as DFT only much faster.
- Some versions work best when N is a power of 2, e.g. 32, 64, 512, 1024, etc.
- 'C' version of FFT for reference on last slide.
- Direct DFT programs of academic interest only.

10 Feb'15 Lecture 3 COMP28512: Mobile Systems 37

MANCHESTER 1824

The University of Manchester

Summary

- Periodic signal of fundamental freq F Hz repeats itself F times per second.
- Has a Fourier series with sinusoids of freq 0, F, 2F, 3F, ...
- Each sinusoid (or 'harmonic') is a delayed sine-wave.
- Some signals are pseudo-periodic (e.g. music & vowels)
- Fourier Series of sampled signals leads to inverse-DFT
- Convenient to use complex numbers
- Transforms to signal $\{x[n]\}_{0,N-1}$ from amplitudes & phases of its harmonics expressed as complex $\{X[k]\}_{0,N-1}$
- DFT transforms from $\{x[n]\}_{0,N-1}$ to $\{X[k]\}_{0,N-1}$
- FFT computes DFT & inverse DFT very efficiently.

10 Feb'15 Lecture 3 COMP28512: Mobile Systems 38

MANCHESTER 1824

The University of Manchester

FFT algorithm in C

```
void fft(void)   {  int N1,N2,NM,N2M,i,j,k,l;  float e,c,s,a,xt,yt;
   N2=N ; NM=N-1;
   if(Invers==1) e= -PI/(float)N;  else e= PI/(float)N;
   for ( k=2; k<N+1; k=k+k)
      { N1=N2; N2=N2/2; N2M=N2-1 ; e=e*2.0; a=0.0;
        for (j=0; j <  N2M+1; j++)
          {  c=cos(a); s=sin(a); a=a+e;
             for ( i=j; i<NM+1; i=i+N1)
                 { l=i+N2; xt=x[i]-x[l]; x[i]=x[i]+x[l];
                   yt=y[i]-y[l]; y[i]=y[i]+y[l];   x[l]=xt*c+yt*s; y[l]=yt*c-xt*s;}
          }   //end of j loop
      } //end of k loop
  if(Invers==1)  for (k=0; k<(NM+1); k++) {x[k]=x[k]/(float)N; y[k]=y[k]/(float)N ;}
  j=0;
  for(i=0; i<N-1; i++)
       { if (i<j) {xt=x[j]; x[j]=x[i]; x[i]=xt; yt=y[j]; y[j]=y[i]; y[i]=yt;}
         k=N/2;   while((k-1)<j) { j=j-k; k=k/2;}  j=j+k;  } // end of i loop
}    // end of procedure FFT
```

10 Feb'15 Lecture 3 COMP28512: Mobile Systems 39