UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia

Pengolahan Sinyal Digital

Adhi Harmoko Saputro

UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia

Sampling of Continuous-Time Signals

Adhi Harmoko Saputro

Signal Types

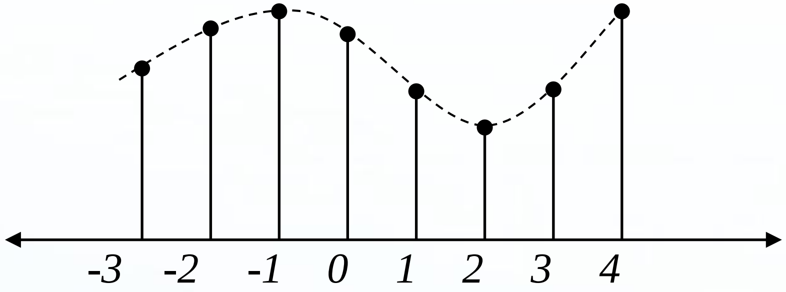
- Analog signals: continuous in time and amplitude
  - Example: voltage, current, temperature,…
- Digital signals: discrete both in time and amplitude
  - Example: attendance of this class, digitizes analog signals,…
- Discrete-time signal: discrete in time, continuous in amplitude
  - Example: hourly change of temperature in Jakarta

Signal Types

- Theory for digital signals would be too complicated
  - Requires inclusion of nonlinearities into theory
- Theory is based on discrete-time continuous-amplitude signals
  - Most convenient to develop theory
  - Good enough approximation to practice with some care
- In practice we mostly process digital signals on processors
  - Need to take into account finite precision effects
- Our text book is about the theory hence its title
  - Discrete-Time Signal Processing

Periodic (Uniform) Sampling

- Sampling is a continuous to discrete-time conversion



- Most common sampling is periodic

$$x[n] = x_c(nT) \quad -\infty < n < \infty$$

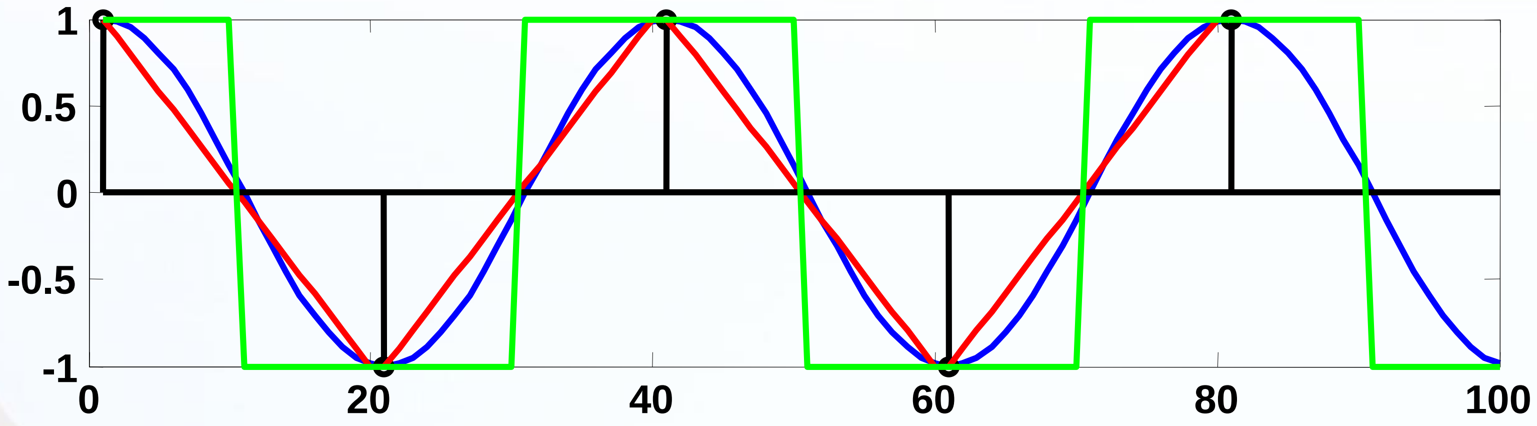
- T is the sampling period in second
- $f_s = 1/T$ is the sampling frequency in Hz

Periodic (Uniform) Sampling

- Sampling frequency in radian-per-second $\Omega_s = 2\pi f_s$ rad/sec
- Use [.] for discrete-time and (.) for continuous time signals
- This is the ideal case not the practical but close enough
  - In practice it is implement with an analog-to-digital converters
  - We get digital signals that are quantized in amplitude and time

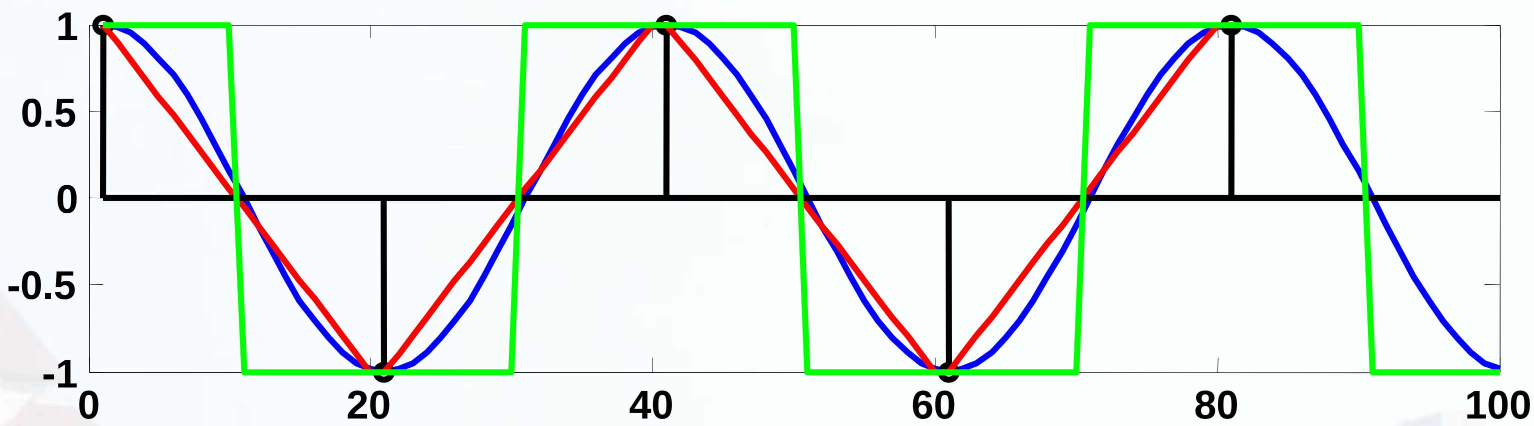
Periodic Sampling

- Sampling is, in general, not reversible
- Given a sampled signal one could fit infinite continuous signals through the samples



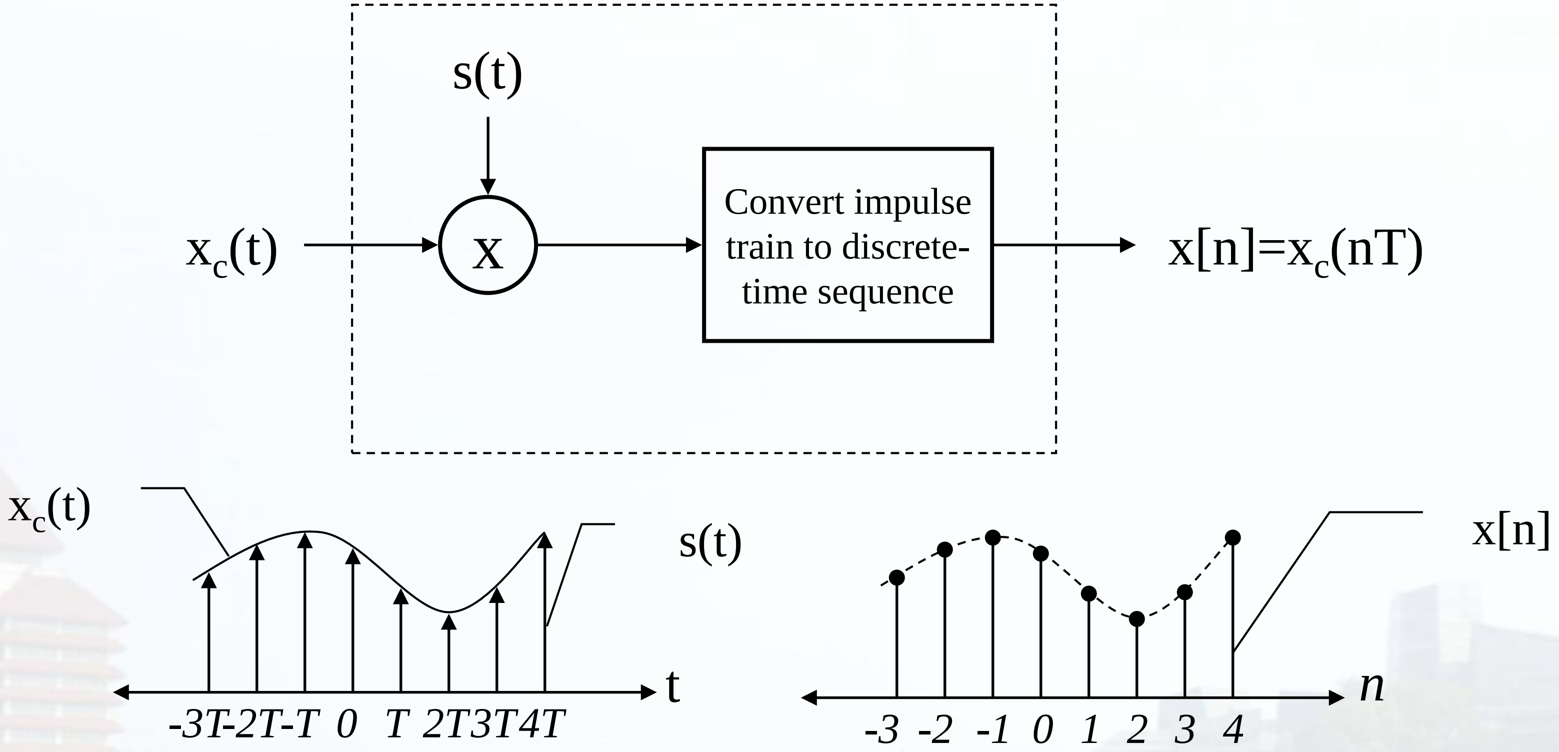
Periodic Sampling

- Fundamental issue in digital signal processing
  - If we loose information during sampling we cannot recover it
- Under certain conditions an analog signal can be sampled without loss so that it can be reconstructed perfectly



Representation of Sampling

- Mathematically convenient to represent in two stages
  - Impulse train modulator
  - Conversion of impulse train to a sequence



Discrete-Time Signals and Systems

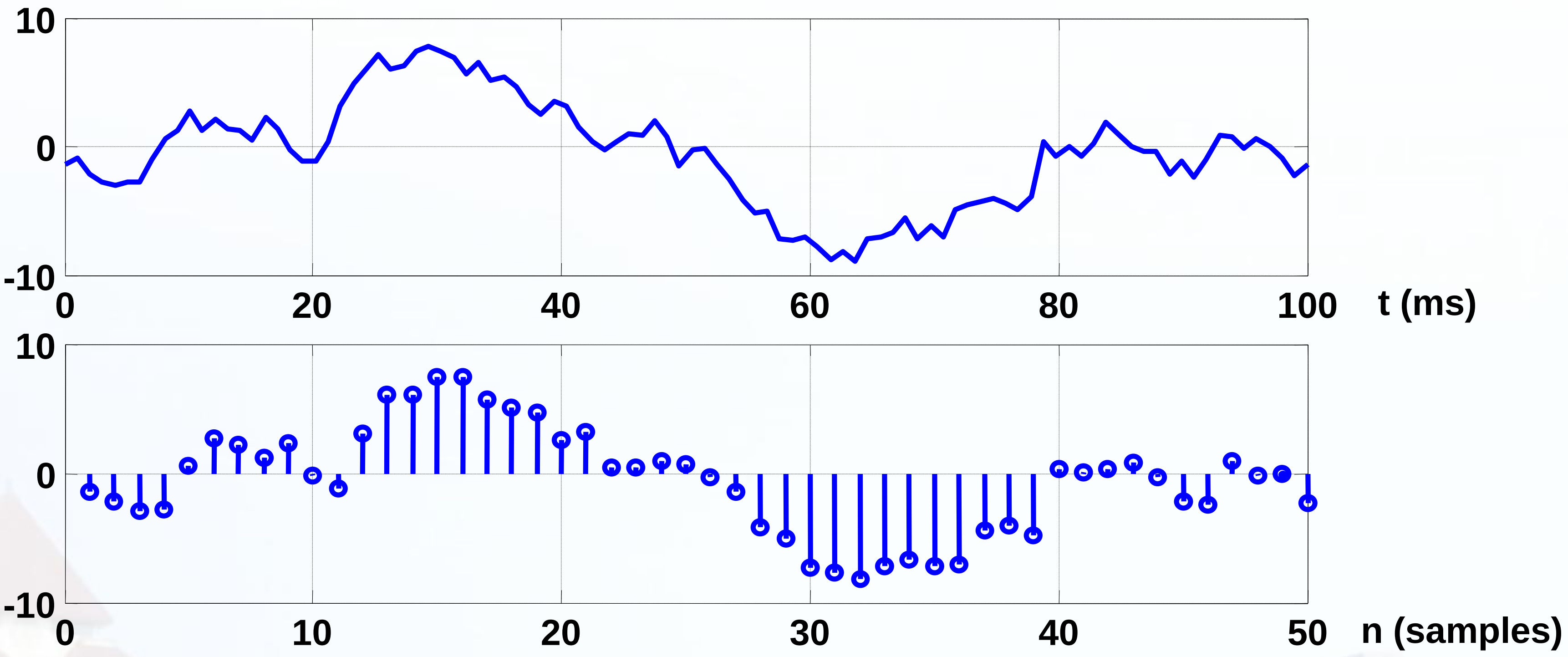
Discrete-Time Signals: Sequences

- Discrete-time signals are represented by sequence of numbers
  - The n^{th} number in the sequence is represented with x[n]

$$x[n] = \{x[n]\} = \{\ldots, x[-1], x[0], x[1], \ldots\}$$

- Often times sequences are obtained by sampling of continuous-time signals
  - In this case x[n] is value of the analog signal at $x_c(nT)$
  - Where T is the sampling period

Discrete-Time Signals: Sequences



Discrete-Time Signals: Sequences

- In MATLAB we can represent a *finite-duration* sequence by a *row vector* of appropriate values
- A vector does not have any information about sample position *n*.
  - A correct representation of $x(n)$ would require two vectors, one each for x and n.

$$x[n] = \left\{2, 1, -1, \underset{\uparrow}{0}, 1, 4, 3, 7\right\}$$

  - represented in MATLAB

```
>> n=[-3,-2,-1,0,1,2,3,4]; x=[2,1,-1,0,1,4,3,7];
```

Basic Sequences and Operations

- Delaying (Shifting) a sequence

$$y[n] = x[n - n_o]$$

- Unit sample (impulse) sequence

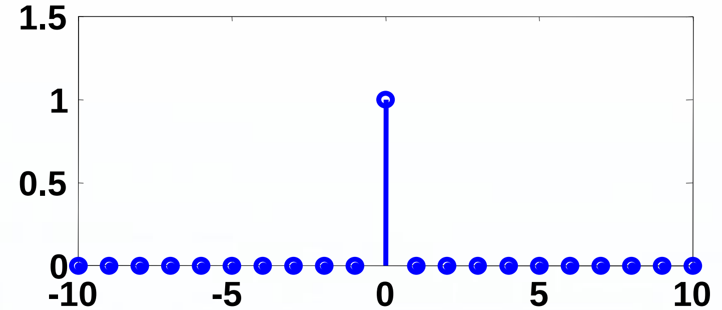
$$\delta[n] = \begin{cases} 0 & n \neq 0 \\ 1 & n = 0 \end{cases}$$

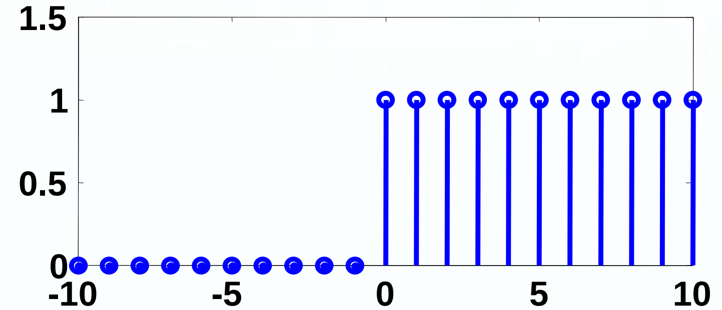
- Unit step sequence

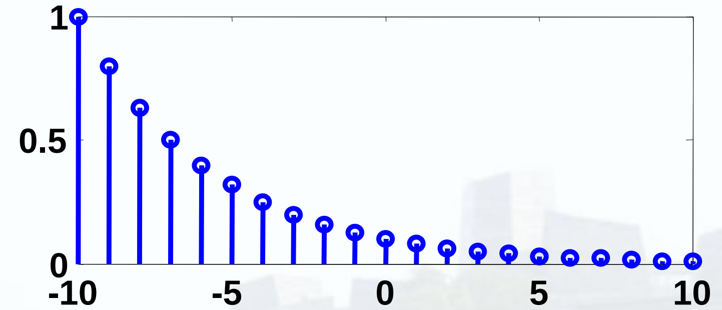
$$u[n] = \begin{cases} 0 & n < 0 \\ 1 & n \geq 0 \end{cases}$$

- Exponential sequences

$$x[n] = A\alpha^n$$







Sinusoidal Sequences

- Important class of sequences

$$x[n] = \cos(\omega_o n + \varphi)$$

- An exponential sequence with complex

$$\alpha = |\alpha| e^{j\omega_o} \text{ and } A = |A| e^{j\varphi}$$

$$x[n] = A\alpha^n = |A| e^{j\varphi} |\alpha|^n e^{j\omega_o n} = |A||\alpha|^n e^{j(\omega_o n + \varphi)}$$

$$x[n] = |A||\alpha|^n \cos(\omega_o n + \varphi) + j|A||\alpha|^n \sin(\omega_o n + \varphi)$$

- x[n] is a sum of weighted sinusoids

Sinusoidal Sequences

- Different from continuous-time, discrete-time sinusoids
  - Have ambiguity of $2\pi k$ in frequency

$$\cos\left(\left(\omega_o + 2\pi k\right)n + \varphi\right) = \cos\left(\omega_o n + \varphi\right)$$

  - Are not necessary periodic with $2\pi/\omega_o$

$$\cos\left(\omega_o n + \varphi\right) = \cos\left(\omega_o n + \omega_o N + \varphi\right) \text{ only if } N = \frac{2\pi k}{\omega_o} \text{ is an integer}$$

MATLAB Example 1

- Create sequence

$$\delta[n-n_o]=\begin{cases}0 & n\neq 0\\ 1 & n=0\end{cases}$$

- over the $n_1 \leq n_0 \leq n_2$ interval

```
function [x,n] = impseq(n0,n1,n2)
% Generates x(n) = delta(n-n0); n1 <= n <= n2
% ----------------------------------------------
% [x,n] = impseq(n0,n1,n2)
%
n = [n1:n2]; x = [(n-n0) == 0];
```

MATLAB Example 2

- Create sequence

$$u[n-n_o]=\begin{cases}0 & n<0\\ 1 & n\geq 0\end{cases}$$

- over the $n_1 \leq n_0 \leq n_2$ interval

```
function [x,n] = stepseq(n0,n1,n2)
% Generates x(n) = u(n-n0); n1 <= n <= n2
% ------------------------------------------
% [x,n] = stepseq(n0,n1,n2)
%
n = [n1:n2]; x = [(n-n0) >= 0];
```

MATLAB Example 3

- Create sequence

$$x[n] = (0.9)^n$$

- over the $0 \le n \le 10$ interval

```
>> n = [0:10]; x = (0.9).^n;
```

MATLAB Example 4

- Create sequence

$$x[n] = (0.9)^n$$

- over the $0 \le n \le 10$ interval

```
>> n = [0:10]; x = (0.9).^n;
```

MATLAB Example 5

- Create sequence

$$x[n] = \exp\left[(2 + j3)n\right]$$

- over the $0 \le n \le 10$ interval

```
>> n = [0:10]; x = exp((2+3j)*n);
```

MATLAB Example 6

- Create sequence

$$x[n] = 3\cos(0.1\pi n + \pi / 3) + 2\sin(0.5\pi n)$$

- over the $0 \le n \le 10$ interval

```
>> n = [0:10];
>> x = 3*cos(0.1*pi*n+pi/3) + 2*sin(0.5*pi*n);
```

Operations on Sequences

- **Signal addition**:

$$\{x_1[n]\}+\{x_2[n]\}=\{x_1[n]+x_2[n]\}$$

- **Signal multiplication:**

$$\{x_1[n]\}.\{x_2[n]\}=\{x_1[n]x_2[n]\}$$

- **Scaling:**

$$\alpha\{x[n]\}=\{\alpha x[n]\}$$

Operations on Sequences

- **Shifting**: each sample of $x[n]$ is shifted by an amount k to obtain a shifted sequence $y[n]$.

$$y[n] = \{x[n-k]\}$$

  - If $m = n-k$, then $n = m+k$ and the above operation is given by

$$y[m+k] = \{x[m]\}$$

Operations on Sequences

- **Folding**: In this operation each sample of $x(n)$ is flipped around $n = 0$ to obtain a folded sequence $y(n)$.

$$y[n] = \{x[-n]\}$$

- **Sample summation**: This operation differs from signal addition operation. It adds all sample values of $x(n)$ between n_1 and n_2.

$$\sum_{n=n_1}^{n_2} x[n] = x[n_1] + \ldots + x[n_2]$$

Operations on Sequences

- **Sample products**: This operation also differs from signal multiplication operation.
  - It multiplies all sample values of $x(n)$ between n_1 and n_2.

$$\prod_{n_1}^{n_2} x[n] = x[n_1] \times \ldots \times x[n_2]$$

- **Signal energy**: The energy of a sequence $x(n)$ is given by

$$\varepsilon_x = \sum_{-\infty}^{\infty} x[n]x^*[n] = \sum_{-\infty}^{\infty} |x[n]|^2$$

Operations on Sequences

- **Signal power**: The average power of a periodic sequence $x[n]$ with fundamental period N is given by

$$P_x = \frac{1}{N}\sum_{0}^{N-1}\left|\tilde{x}[n]\right|^2$$

Matlab Function

```
function [y,n] = sigadd(x1,n1,x2,n2)
% implements y(n) = x1(n)+x2(n)
% -----------------------------
% [y,n] = sigadd(x1,n1,x2,n2)
% y = sum sequence over n, which includes n1 and n2
% x1 = first sequence over n1
% x2 = second sequence over n2 (n2 can be different from n1)
%
n = min(min(n1),min(n2)):max(max(n1),max(n2));
% duration of y(n)
y1 = zeros(1,length(n)); y2 = y1;
% initialization
y1(find((n>=min(n1))&(n<=max(n1))==1))=x1;
% x1 with duration of y
y2(find((n>=min(n2))&(n<=max(n2))==1))=x2;
% x2 with duration of y
y = y1+y2; % sequence addition
```

Matlab Function

```
function [y,n] = sigmult(x1,n1,x2,n2)
% implements y(n) = x1(n)*x2(n)
% -----------------------------
% [y,n] = sigmult(x1,n1,x2,n2)
% y = product sequence over n, which includes n1 and n2
% x1 = first sequence over n1
% x2 = second sequence over n2 (n2 can be different from n1)
n = min(min(n1),min(n2)):max(max(n1),max(n2));
% duration of y(n)
y1 = zeros(1,length(n)); y2 = y1; %
y1(find((n>=min(n1))&(n<=max(n1))==1))=x1;
% x1 with duration of y
y2(find((n>=min(n2))&(n<=max(n2))==1))=x2;
% x2 with duration of y
y = y1 .* y2; % sequence multiplication
```

Matlab Function

```
function [y,n] = sigshift(x,m,k)
% implements y(n) = x(n-k)
% -------------------------
% [y,n] = sigshift(x,m,k)
%
n = m+k; y = x;
```

Matlab Function

```
function [y,n] = sigfold(x,n)
% implements y(n) = x(-n)
% -----------------------
% [y,n] = sigfold(x,n)
%
y = fliplr(x); n = -fliplr(n);
```

MATLAB Example 7

- Generate and plot each of the following sequences over the indicated interval

$$x(n) = 2\delta(n+2) - \delta(n-4), \; -5 \le n \le 5$$

```
>> n = [-5:5];
>> x = 2*impseq(-2,-5,5) - impseq(4,-5,5);
>> stem(n,x); title('Sequence in MATLAB Example 7')
>> xlabel('n'); ylabel('x(n)');
```

MATLAB Example 8

- Generate and plot each of the following sequences over the indicated interval

$$x(n) = n[u(n)-u(n-10)]+10e^{-0.3(n-10)}[u(n-10)-u(n-20)],$$

$$0 \le n \le 20$$

```
>> n = [0:20];
>> x1 = n.*(stepseq(0,0,20)-stepseq(10,0,20));
>> x2 = 10*exp(-0.3*(n-10)).*(stepseq(10,0,20)- ...
        stepseq(20,0,20));
>> x = x1+x2;
>> subplot(2,2,3); stem(n,x);
>> title('Sequence in MATLAB Example 8')
>> xlabel('n'); ylabel('x(n)');
```

MATLAB Example 9

- Generate and plot each of the following sequences over the indicated interval

$$x(n) = \cos(0.04\pi n) + 0.2w(n),\ 0 \le n \le 50$$

$w(n)$ is a Gaussian random sequence with zero mean and unit variance

```
>> n = [0:50];
>> x = cos(0.04*pi*n)+0.2*randn(size(n));
>> subplot(2,2,2); stem(n,x);
>> title('Sequence in MATLAB Example 9')
>> xlabel('n'); ylabel('x(n)');
```

MATLAB Example 10

- Generate and plot each of the following sequences over the indicated interval

$$x[n] = \{\ldots, 5, 4, 3, 2, 1, \underset{\uparrow}{5}, 4, 3, 2, 1, 5, 4, 3, 2, 1, \ldots\}; \quad -10 \le n \le 9$$

- Note that over the given interval, the sequence $\tilde{x}(n)$ has four periods

```
>> n = [-10:9]; x = [5,4,3,2,1];
>> xtilde = x' * ones(1,4); xtilde = (xtilde(:))';
>> subplot(2,2,4); stem(n,xtilde);
>> title('Sequence in MATLAB Example 10')
>> xlabel('n'); ylabel('xtilde(n)');
```

MATLAB Example 10

- Let

$$x[n] = \{1, 2, \underset{\uparrow}{3}, 4, 5, 6, 7, 6, 5, 4, 3, 2, 1\}$$

- Determine and plot the following sequences.
  - $x_1[n] = 2x(n - 5) - 3x(n + 4)$
  - $x_2[n] = x(3 - n) + x(n)\, x(n - 2)$

MATLAB Example 10

$$x[n] = \{1, 2, \underset{\uparrow}{3}, 4, 5, 6, 7, 6, 5, 4, 3, 2, 1\}$$

- The sequence $x(n)$ is nonzero over $-2 \leq n \leq 10$.

```
>> n = -2:10; x = [1:7,6:-1:1];
```

MATLAB Example 10a

- $x_1[n] = 2x[n - 5] - 3x[n + 4]$
- The first part is obtained by shifting $x[n]$ by 5 and the second part by shifting $x[n]$ by -4.
- This shifting and the addition can be easily done using the sigshift and the sigadd functions.

```
>> [x11,n11] = sigshift(x,n,5);
>> [x12,n12] = sigshift(x,n,-4);
>> [x1,n1] = sigadd(2*x11,n11,-3*x12,n12);
>> subplot(2,1,1); stem(n1,x1);
>> title('Sequence in MATLAB Example 10a')
>> xlabel('n'); ylabel('x1(n)');
```

MATLAB Example 10b

- $x_2[n] = x[3 - n] + x[n]\, x[n - 2]$
- The first term can be written as $x[-[n - 3]]$. Hence it is obtained by first folding $x[n]$ and then shifting the result by 3.
- The second part is a multiplication of $x[n]$ and $x[n-2]$, both of which have the same length but different support (or sample positions).

MATLAB Example 10b

- $x_2[n] = x[3-n] + x[n]\,x[n-2]$
- These operations can be easily done using the sigfold and the sigmult functions

```
>> [x21,n21] = sigfold(x,n);
>> [x21,n21] = sigshift(x21,n21,3);
>> [x22,n22] = sigshift(x,n,2);
>> [x22,n22] = sigmult(x,n,x22,n22);
>> [x2,n2] = sigadd(x21,n21,x22,n22);
>> subplot(2,1,2); stem(n2,x2);
>> title('Sequence in MATLAB Example 10b')
>> xlabel('n'); ylabel('x2(n)');
```

MATLAB Example 11

- Generate the complex-valued signal

$$x[n] = e^{[-0.1+j0.3]n}, \; -10 \leq n \leq 10$$

- and plot its magnitude, phase, the real part, and the imaginary part in four separate subplots.

```
>> n = [-10:1:10]; alpha = -0.1+0.3j;
>> x = exp(alpha*n);
>> subplot(2,2,1); stem(n,real(x));
>> title('real part');xlabel('n')
>> subplot(2,2,2); stem(n,imag(x));
>> title('imaginary part');xlabel('n')
>> subplot(2,2,3); stem(n,abs(x));
>> title('magnitude part');xlabel('n')
>> subplot(2,2,4); stem(n,(180/pi)*angle(x));
>> title('phase part');xlabel('n')
```

Discrete-Time Sinusoidal

- Basis for Fourier transform and in system theory as a basis for steady-state analysis

$$x[n] = A\cos(\omega_o n + \theta_o)$$

- Conveniently related to the continuous-time sinusoid

$$x_a(t) = A\cos(\Omega_o t + \theta_o)$$

Periodicity in time

- the sinusoidal sequence is periodic if

$$x[n+N] = A\cos(\omega_o n + \omega_o N + \theta) = A\cos(\omega_o n + \theta_o) = x[n]$$

- This is possible if and only if $\omega_o N = 2\pi k$, where k is an integer

Unit sample synthesis

- Any arbitrary sequence $x[n]$ can be synthesized as a weighted sum of delayed and scaled unit sample sequences, such as

$$x[n] = \sum_{k=-\infty}^{\infty} x[k]\delta[n-k]$$

Even and odd synthesis

- A real-valued sequence $x_e[n]$ is called even (symmetric) if

$$x_e[-n] = x_e[n]$$

- A real-valued sequence $x_o[n]$ is called odd (antisymmetric) if

$$x_o[-n] = -x_o[n]$$

Even and odd synthesis

- Then any arbitrary real-valued sequence $x[n]$ can be decomposed into its even and odd components

$$x[n] = x_e[n] + x_o[n]$$

- where the even and odd parts are given by

$$x_e[n] = \frac{1}{2}\left[x[n] + x[-n]\right] \qquad x_o[n] = \frac{1}{2}\left[x[n] - x[-n]\right]$$

Even and odd synthesis

- MATLAB function to decompose a given sequence into its even and odd components

```
function [xe, xo, m] = evenodd(x,n)
% Real signal decomposition into even and odd parts
% ---------------------------------------------------
% [xe, xo, m] = evenodd(x,n)
%
if any(imag(x) ~= 0)
error('x is not a real sequence')
end
m = -fliplr(n);
m1 = min([m,n]); m2 = max([m,n]); m = m1:m2;
nm = n(1)-m(1); n1 = 1:length(n);
x1 = zeros(1,length(m)); x1(n1+nm) = x; x = x1;
xe = 0.5*(x + fliplr(x)); xo = 0.5*(x - fliplr(x));
```

MATLAB Example

- Let $x[n] = u[n] - u[n-10]$.
  - Decompose $x[n]$ into even and odd components.

MATLAB Example

- Let $x[n] = u[n] - u[n-10]$.
  - Decompose $x[n]$ into even and odd components.
- The sequence $x[n]$, which is nonzero over $0 \leq n \leq 9$, is called a *rectangular pulse*

```
>> n = [0:10]; x = stepseq(0,0,10)-stepseq(10,0,10);
>> [xe,xo,m] = evenodd(x,n);
>> subplot(2,2,1); stem(n,x);
>> title('Rectangular pulse')
>> xlabel('n'); ylabel('x(n)'); axis([-10,10,0,1.2])
>> subplot(2,2,2); stem(m,xe); title('Even Part')
>> xlabel('n'); ylabel('xe(n)');
>> axis([-10,10,0,1.2])
>> subplot(2,2,4); stem(m,xo); title('Odd Part')
>> xlabel('n'); ylabel('xe(n)');
>> axis([-10,10,-0.6,0.6])
```

The geometric series

- A one-sided exponential sequence of the form

$$\{\ \alpha^n,\ n \geq 0\ \}$$

  - where α is an arbitrary constant

- The convergence and expression for the sum of this series are used in many applications

Correlations of sequences

- Measure of the degree to which two sequences are similar
- the *crosscorrelation* of $x[n]$ and $y[n]$ is a sequence $r_{xy}[l]$ defined as

$$r_{xy}[\ell] = \sum_{n=-\infty}^{\infty} x[n]\, y[n-\ell]$$


Terima Kasih