Sampling and Reconstruction of Analog Signals

Chapter Intended Learning Outcomes:

(i) Ability to convert an analog signal to a discrete-time sequence via sampling

(ii) Ability to construct an analog signal from a discrete-time sequence

(iii) Understanding the conditions when a sampled signal can uniquely present its analog counterpart

Sampling

- Process of converting a continuous-time signal $x(t)$ into a discrete-time sequence $x[n]$

- $x[n]$ is obtained by extracting $x(t)$ every T s where T is known as the sampling period or interval

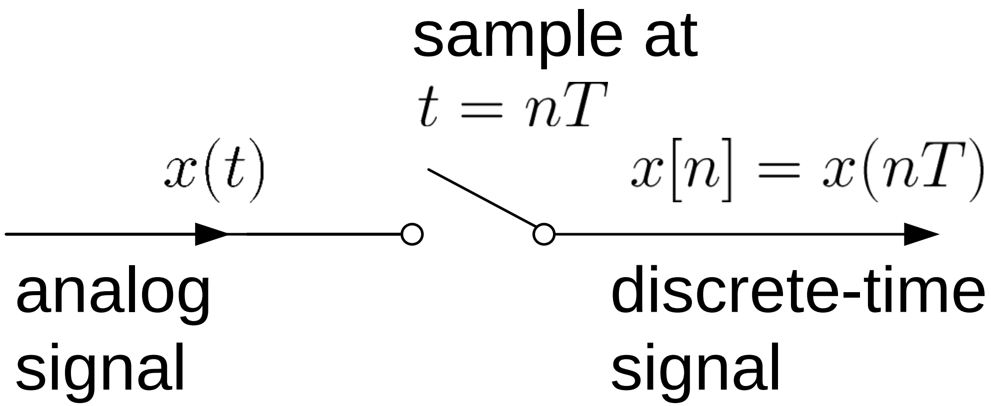

Fig.4.1: Conversion of analog signal to discrete-time sequence

- Relationship between $x(t)$ and $x[n]$ is:

$$x[n] = x(t)|_{t=nT} = x(nT), \quad n = \cdots - 1, 0, 1, 2, \cdots \qquad (4.1)$$

- Conceptually, conversion of $x(t)$ to $x[n]$ is achieved by a continuous-time to discrete-time (CD) converter:

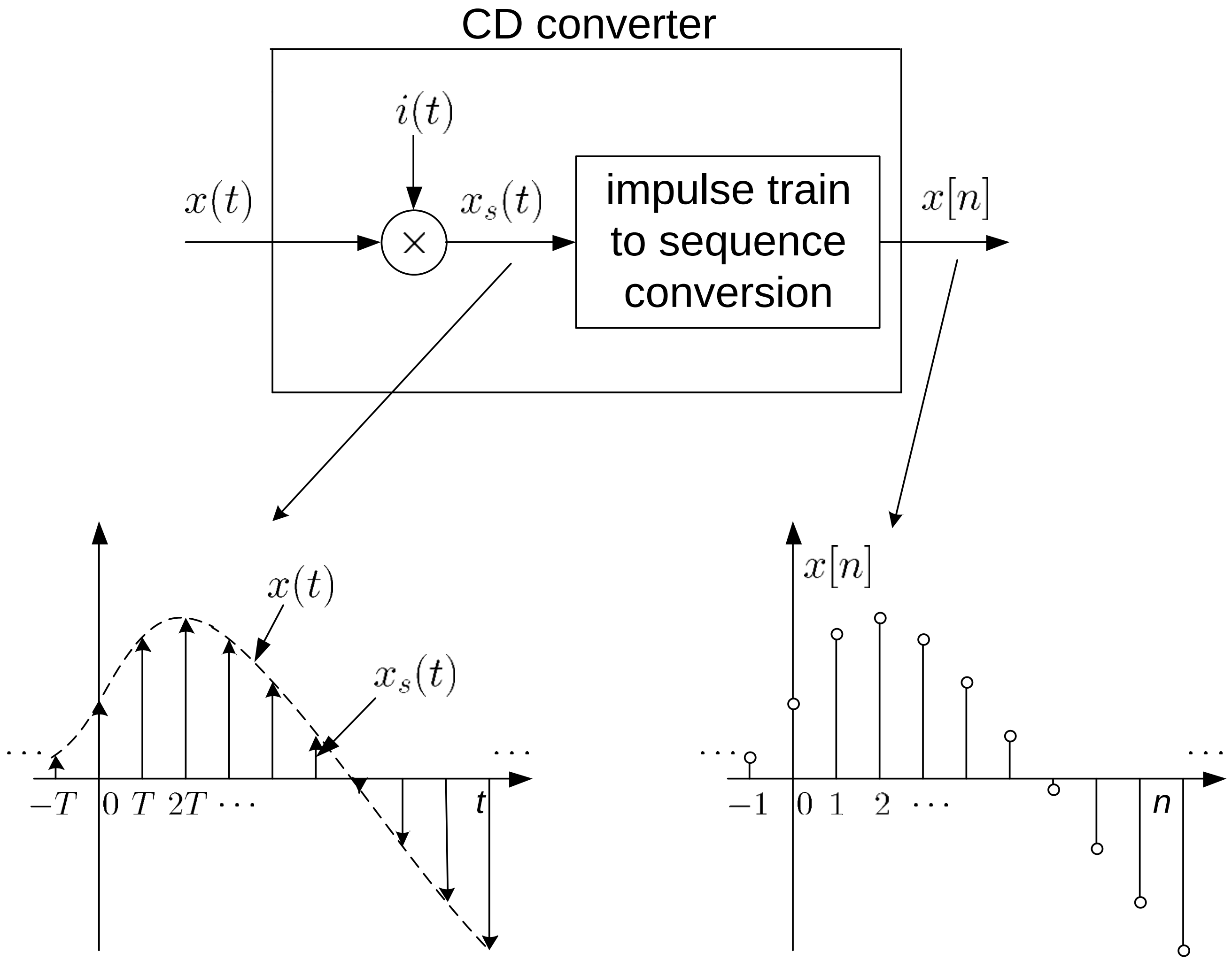

Fig.4.2: Block diagram of CD converter

▪ A fundamental question is whether $x[n]$ can uniquely represent $x(t)$ or if we can use $x[n]$ to reconstruct $x(t)$

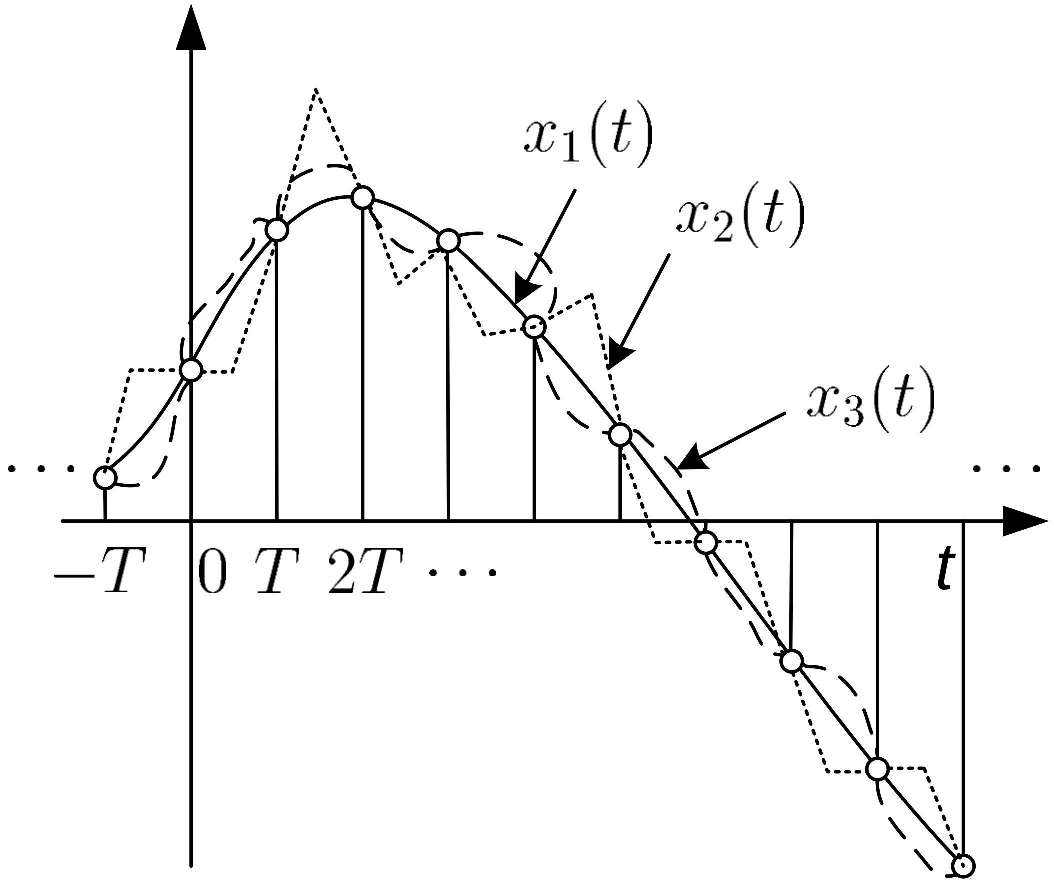

Fig.4.3: Different analog signals map to same sequence

But, the answer is yes when:

(1) $x(t)$ is bandlimited such that its Fourier transform $X(j\Omega) = 0$ for $|\Omega| \geq \Omega_b$ where Ω_b is called the bandwidth

(2) Sampling period T is sufficiently small

Example 4.1

The continuous-time signal $x(t) = \cos(200\pi t)$ is used as the input for a CD converter with the sampling period $1/300$ s. Determine the resultant discrete-time signal $x[n]$.

According to (4.1), $x[n]$ is

$$x[n] = x(nT) = \cos(200n\pi T) = \cos\left(\frac{2\pi n}{3}\right), \quad n = \cdots - 1, 0, 1, 2, \cdots$$

The frequency in $x(t)$ is $200\pi \ \mathrm{rads}^{-1}$ while that of $x[n]$ is $2\pi/3$

Frequency Domain Representation of Sampled Signal

In the time domain, $x_s(t)$ is obtained by multiplying $x(t)$ by the impulse train $i(t) = \sum_{k=-\infty}^{\infty} \delta(t-kT)$:

$$x_s(t) = x(t) \sum_{k=-\infty}^{\infty} \delta(t-kT) = \sum_{k=-\infty}^{\infty} x[k]\delta(t-kT) \qquad (4.2)$$

with the use of the sifting property of (2.12)

Let the sampling frequency in radian be $\Omega_s = 2\pi/T$ (or $F_s = 1/T = \Omega_s/(2\pi)$ in Hz). From Example 2.8:

$$I(j\Omega) = \Omega_s \sum_{k=-\infty}^{\infty} \delta(\Omega - k\Omega_s) \qquad (4.3)$$

Using multiplication property of Fourier transform:

$$x_1(t) \cdot x_2(t) \leftrightarrow \frac{1}{2\pi} X_1(j\Omega) \otimes X_2(j\Omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X_1(j\tau) X_2(j(\Omega - \tau)) d\tau \quad (4.4)$$

where the convolution operation corresponds to continuous-time signals

Using (4.2)-(4.4) and properties of $\delta(t)$, $X_s(j\Omega)$ is:

$$
\begin{aligned}
X_s(j\Omega) &= \frac{1}{2\pi}\int_{-\infty}^{\infty} I(j\tau)X(j(\Omega-\tau))d\tau \\
&= \frac{1}{2\pi}\int_{-\infty}^{\infty}\left(\Omega_s\sum_{k=-\infty}^{\infty}\delta(\tau-k\Omega_s)\right)X(j(\Omega-\tau))d\tau \\
&= \frac{1}{T}\sum_{k=-\infty}^{\infty}\left(\int_{-\infty}^{\infty}X(j(\Omega-\tau))\delta(\tau-k\Omega_s)d\tau\right) \\
&= \frac{1}{T}\sum_{k=-\infty}^{\infty}X(j(\Omega-k\Omega_s))\left(\int_{-\infty}^{\infty}\delta(\tau-k\Omega_s)d\tau\right) \\
&= \frac{1}{T}\sum_{k=-\infty}^{\infty}X(j(\Omega-k\Omega_s)) \qquad (4.5)
\end{aligned}
$$

which is the sum of infinite copies of $X(j\Omega)$ scaled by $1/T$

When Ω_s is chosen sufficiently large such that all copies of $X(j\Omega)/T$ do not overlap, that is, $\Omega_s - \Omega_b > \Omega_b$ or $\Omega_s > 2\Omega_b$, we can get $X(j\Omega)$ from $X_s(j\Omega)$

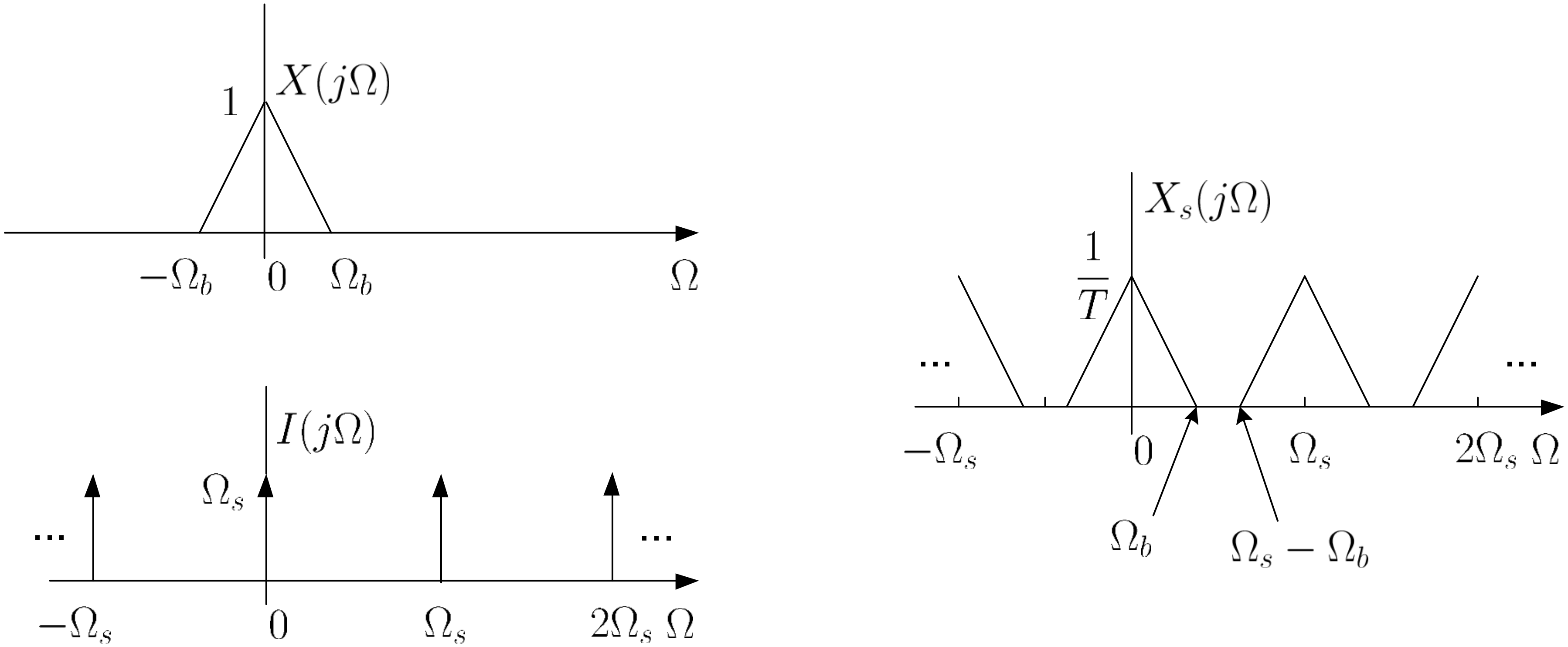

Fig.4.4: $X_s(j\Omega) = X(j\Omega) \otimes I(j\Omega)$ for sufficiently large Ω_s

When Ω_s is not chosen sufficiently large such that $\Omega_s < 2\Omega_b$, copies of $X(j\Omega)/T$ overlap, we cannot get $X(j\Omega)$ from $X_s(j\Omega)$, which is referred to aliasing

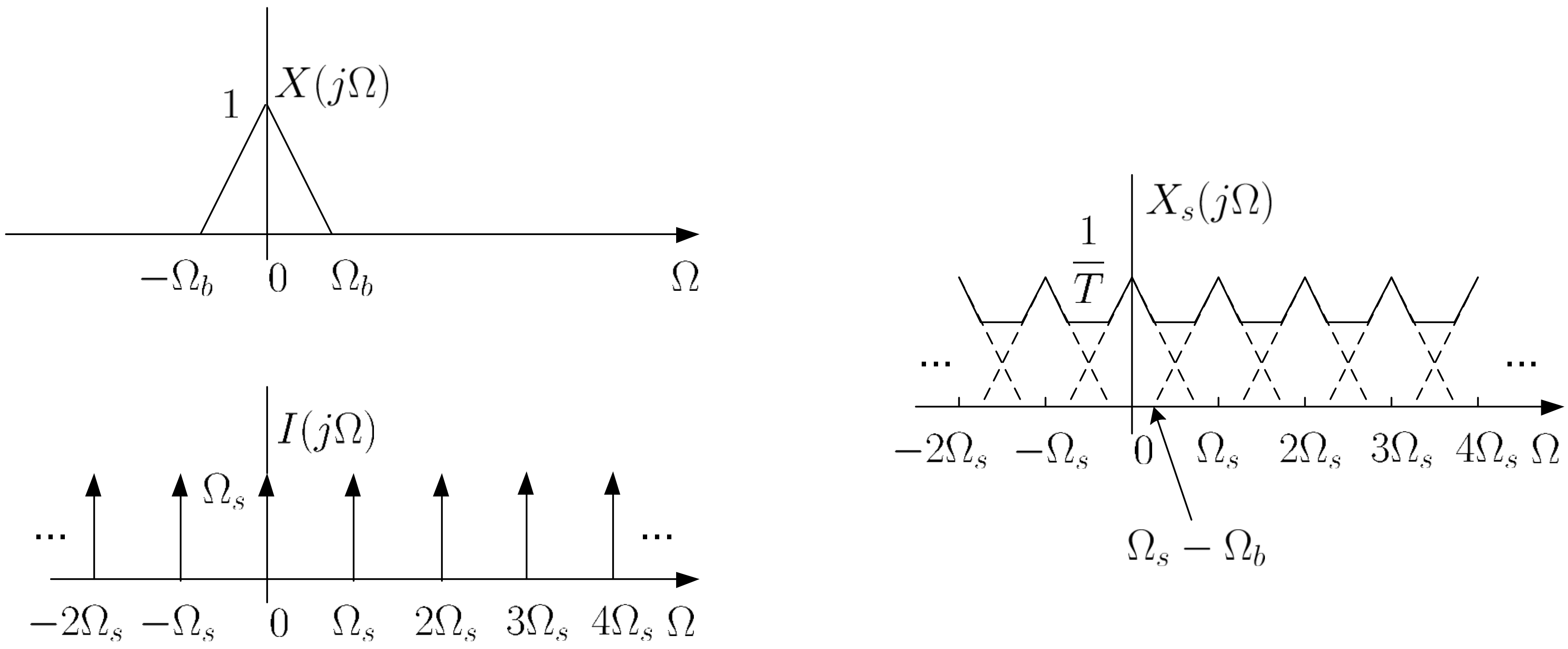

Fig.4.5: $X_s(j\Omega) = X(j\Omega) \otimes I(j\Omega)$ when Ω_s is not large enough

Nyquist Sampling Theorem (1928)

Let $x(t)$ be a bandlimited continuous-time signal with

$$X(j\Omega) = 0, \quad |\Omega| \geq \Omega_b \tag{4.6}$$

Then $x(t)$ is uniquely determined by its samples $x[n] = x(nT)$, $n = \cdots -1, 0, 1, 2, \cdots$, if

$$\Omega_s = \frac{2\pi}{T} > 2\Omega_b \tag{4.7}$$

The bandwidth Ω_b is also known as the Nyquist frequency while $2\Omega_b$ is called the Nyquist rate and Ω_s must exceed it in order to avoid aliasing

Application	$f_b = \Omega_b/(2\pi)$	$f_s = \Omega_s/(2\pi)$
Biomedical	< 500 Hz	1 kHz
Telephone speech	< 4 kHz	8 kHz
Music	< 20 kHz	44.1 kHz
Ultrasonic	< 100 kHz	250 kHz
Radar	< 100 MHz	200 MHz

Table 4.1: Typical bandwidths and sampling frequencies in signal processing applications

Example 4.2

Determine the Nyquist frequency and Nyquist rate for the continuous-time signal $x(t)$ which has the form of:

$$x(t) = 1 + \sin(2000\pi t) + \cos(4000\pi t)$$

The frequencies are 0, 2000π and 4000π. The Nyquist frequency is $4000\pi \text{ rads}^{-1}$ and the Nyquist rate is $8000\pi \text{ rads}^{-1}$

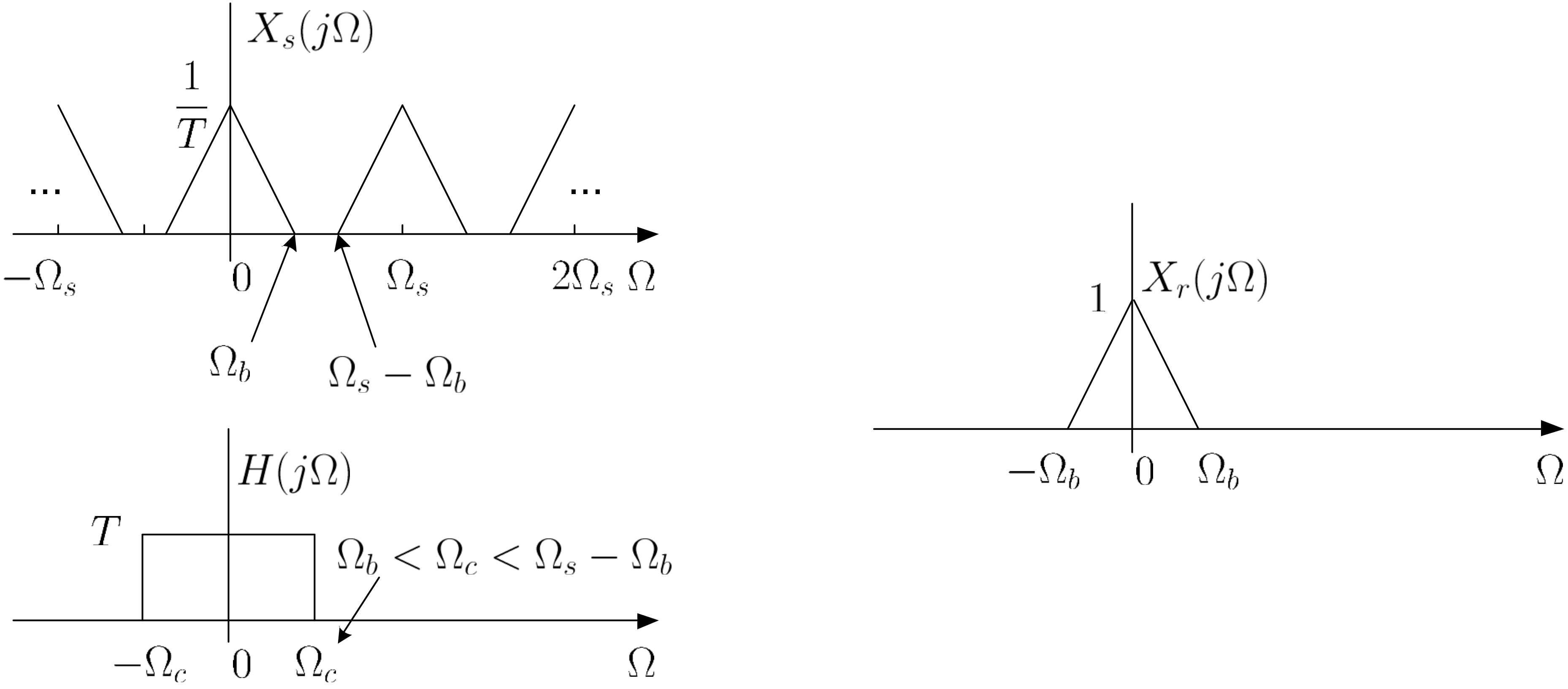

Fig.4.6: Multiplying $X_s(j\Omega)$ and $H(j\Omega)$ to recover $X(j\Omega)$

In frequency domain, we multiply $X_s(j\Omega)$ by $H(j\Omega)$ with amplitude T and bandwidth Ω_c with $\Omega_b < \Omega_c < \Omega_s - \Omega_b$, to obtain $X_r(j\Omega)$, and it corresponds to $x_r(t) = x_s(t) \otimes h(t)$

Reclaim

- Process of transforming $x[n]$ back to $x(t)$

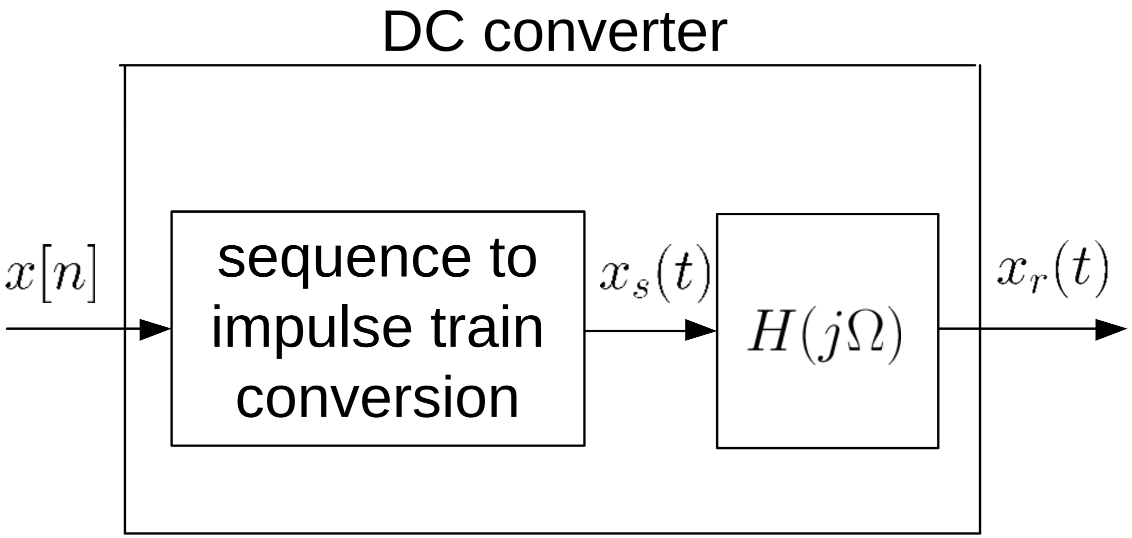

Fig.4.7: Block diagram of DC converter

From Fig.4.6, $H(j\Omega)$ is

$$H(j\Omega) = \begin{cases} T, & -\Omega_c < \Omega < \Omega_c \\ 0, & \text{otherwise} \end{cases} \quad (4.8)$$

where $\Omega_b < \Omega_c < \Omega_s - \Omega_b$

For simplicity, we set Ω_c as the average of Ω_b and $(\Omega_s - \Omega_b)$:

$$\Omega_c = \frac{\Omega_s}{2} = \frac{\pi}{T} \tag{4.9}$$

To get $h(t)$, we take inverse Fourier transform of $H(j\Omega)$ and use Example 2.5:

$$\begin{aligned} h(t) &= \frac{1}{2\pi} \int_{-\infty}^{\infty} H(j\Omega) e^{j\Omega t} d\Omega = \frac{1}{2\pi} \int_{-\pi/T}^{\pi/T} T e^{j\Omega t} d\Omega = \frac{T \sin(\pi t/T)}{\pi t} \\ &= \mathrm{sinc}\left(\frac{t}{T}\right) \end{aligned} \tag{4.10}$$

where $\mathrm{sinc}(u) = \sin(\pi u)/(\pi u)$

Using (2.23)-(2.24), (4.2) and (2.11)-(2.12), $x_r(t)$ is:

$$
\begin{aligned}
x_r(t) &= x_s(t) \otimes h(t) \\
&= \left(\sum_{k=-\infty}^{\infty} x[k]\delta(t-kT) \right) \otimes h(t) \\
&= \int_{-\infty}^{\infty} \sum_{k=-\infty}^{\infty} x[k]\delta(\tau-kT)h(t-\tau)d\tau \\
&= \sum_{k=-\infty}^{\infty} x[k]h(t-kT) \\
&= \sum_{k=-\infty}^{\infty} x[k]\mathrm{sinc}\left(\frac{t-kT}{T}\right) \qquad (4.11)
\end{aligned}
$$

which holds for all real values of t

The interpolation formula can be verified at $t = nT$:

$$x_r(nT) = \sum_{k=-\infty}^{\infty} x[k]\text{sinc}\,(n-k) \tag{4.12}$$

It is easy to see that

$$\text{sinc}\,(n-k) = \frac{\sin((n-k)\pi)}{(n-k)\pi} = 0, \quad n \neq k \tag{4.13}$$

For $n = k$, we use L'Hôpital's rule to obtain:

$$\text{sinc}(0) = \lim_{m\to 0} \frac{\sin\,(m\pi)}{m\pi} = \lim_{m\to 0} \frac{\dfrac{d\sin\,(m\pi)}{dm}}{\dfrac{dm\pi}{dm}} = \lim_{m\to 0} \frac{\pi\cos(m\pi)}{\pi} = 1 \tag{4.14}$$

Substituting (4.13)-(4.14) into (4.12) yields:

$$x_r(nT) = x[n] = x(nT) \tag{4.15}$$

which aligns with $x_r(t) = x(t)$

Example 4.3

Given a discrete-time sequence $x[n] = x(nT)$. Generate its time-delayed version $y[n]$ which has the form of

$$y[n] = x(nT - \Delta)$$

where $\Delta \neq mT > 0$ and m is a positive integer. Applying (4.11) with $t = nT - \Delta$:

$$y[n] = x(nT - \Delta) = \sum_{k=-\infty}^{\infty} x[k]\mathrm{sinc}\left(\frac{nT - kT - \Delta}{T}\right)$$

By employing a change of variable of $l = n - k$:

$$y[n] = \sum_{l=-\infty}^{\infty} x[n-l]\mathrm{sinc}\left(\frac{lT - \Delta}{T}\right)$$

Is it practical to get *y*[n]?

Note that when $\Delta = mT$, the time-shifted signal is simply obtained by shifting the sequence $x[n]$ by m samples:

$$y[n] = x(nT - mT) = x[n-m]$$

Sampling and Reconstruction in Digital Signal Processing

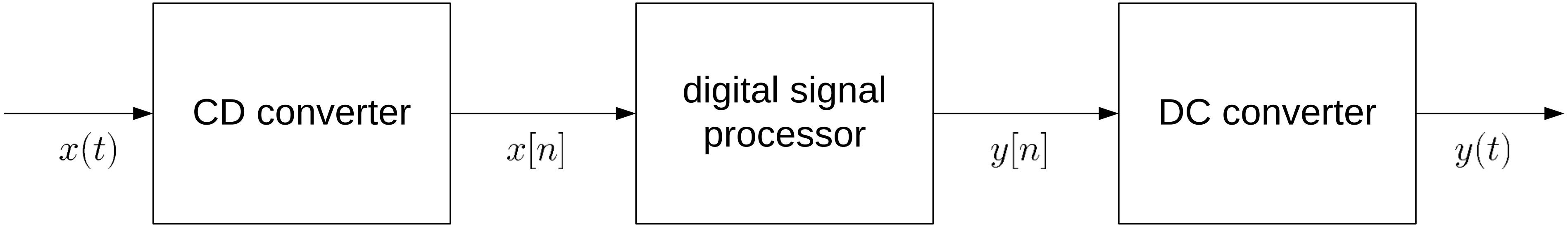

Fig.4.8: Ideal digital processing of analog signal

- CD converter produces a sequence $x[n]$ from $x(t)$
- $x[n]$ is processed in discrete-time domain to give $y[n]$
- DC converter creates $y(t)$ from $y[n]$ according to (4.11):

$$y(t) = \sum_{k=-\infty}^{\infty} y[k] \mathrm{sinc}\left(\frac{t-kT}{T}\right) \quad (4.16)$$

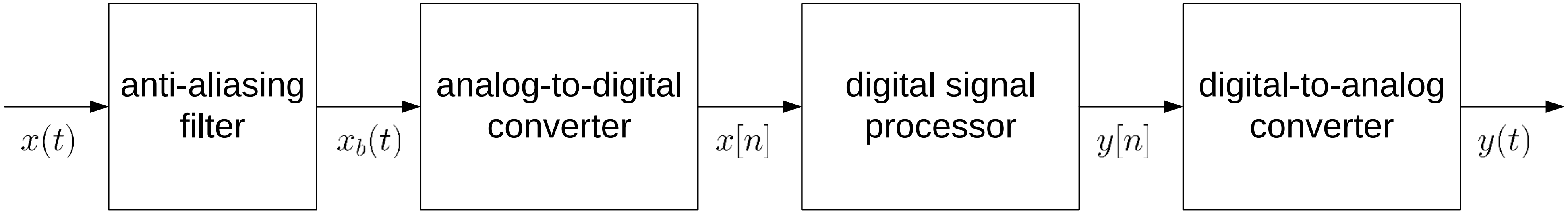

Fig.4.9: Practical digital processing of analog signal

- $x(t)$ may not be precisely bandlimited $\Rightarrow$ a lowpass filter or anti-aliasing filter is needed to process $x(t)$
- Ideal CD converter is approximated by AD converter
    - Not practical to generate $\delta(t)$
    - AD converter introduces quantization error
- Ideal DC converter is approximated by DA converter because ideal reconstruction of (4.16) is impossible
    - Not practical to perform infinite summation
    - Not practical to have future data
- $x[n]$ and $y[n]$ are quantized signals

Example 4.4

Suppose a continuous-time signal $x(t)=\cos(\Omega_0 t)$ is sampled at a sampling frequency of 1000Hz to produce $x[n]$:

$$x[n]=\cos\left(\frac{\pi n}{4}\right)$$

Determine 2 possible positive values of Ω_0, say, Ω_1 and Ω_2. Discuss if $\cos(\Omega_1 t)$ or $\cos(\Omega_2 t)$ will be obtained when passing $x[n]$ through the DC converter.

According to (4.1) with $T=1/1000$ s:

$$\cos\left(\frac{\pi n}{4}\right)=x[n]=x(nT)=\cos\left(\frac{\Omega_0 n}{1000}\right)$$

Ω_1 is easily computed as:

$$\frac{\pi n}{4}=\frac{\Omega_1 n}{1000}\Rightarrow \Omega_1=\frac{1000\pi}{4}=250\pi$$

Ω_2 can be obtained by noting the periodicity of a sinusoid:

$$\cos\left(\frac{\pi n}{4}\right) = \cos\left(\frac{\pi n}{4} + 2n\pi\right) = \cos\left(\frac{9\pi n}{4}\right) = \cos\left(\frac{\Omega_2 n}{1000}\right)$$

As a result, we have:

$$\frac{9\pi n}{4} = \frac{\Omega_2 n}{1000} \Rightarrow \Omega_2 = \frac{9000\pi}{4} = 2250\pi$$

This is illustrated using the MATLAB code:

```
O1=250*pi;                    %first frequency
O2=2250*pi;                   %second frequency
Ts=1/100000;%successive sample separation is 0.01T
t=0:Ts:0.02;%observation interval
x1=cos(O1.*t);                %tone from first frequency
x2=cos(O2.*t);                 %tone from second frequency
```

There are 2001 samples in 0.02s and interpolating the successive points based on `plot` yields good approximations

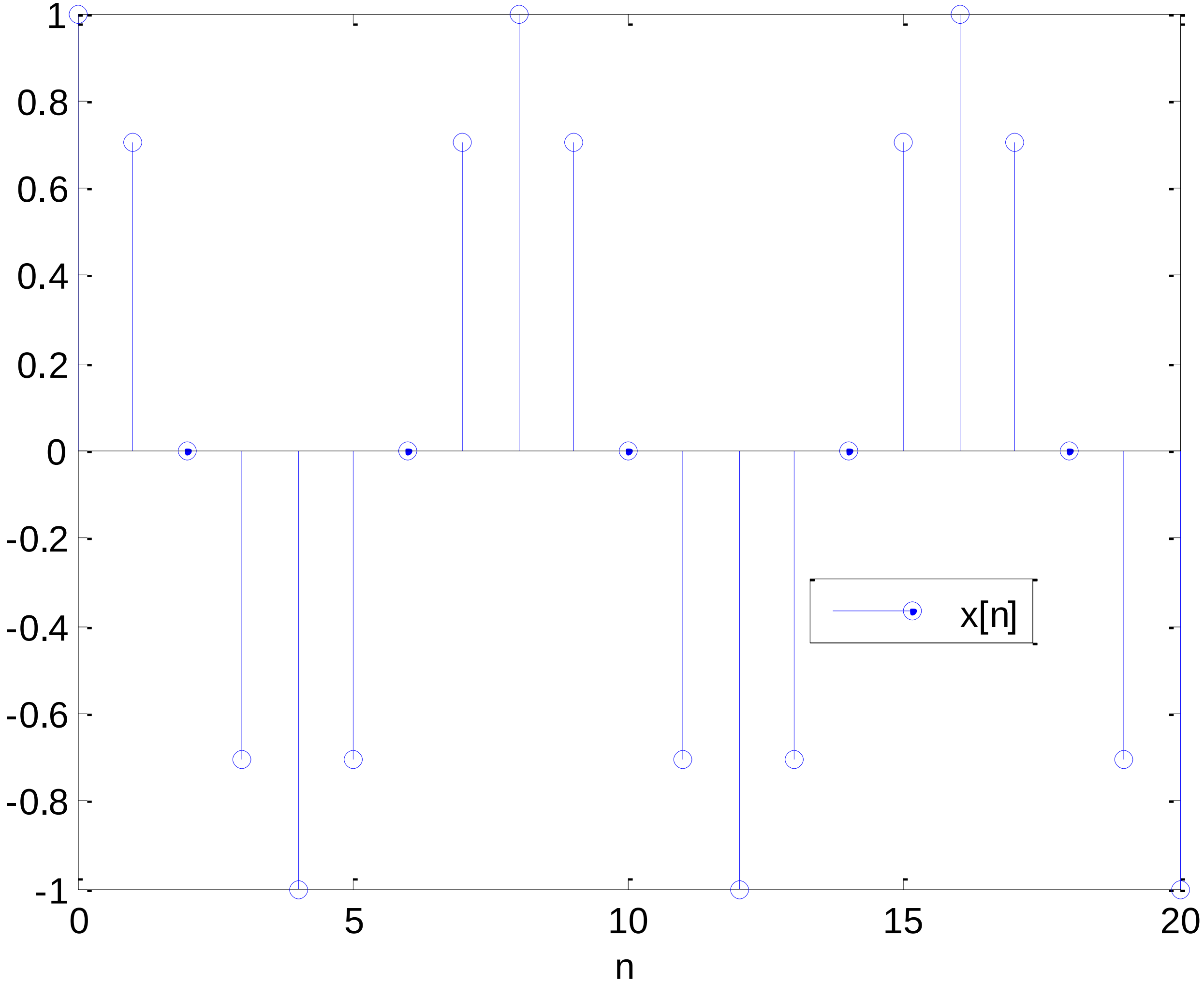

Fig.4.10: Discrete-time sinusoid

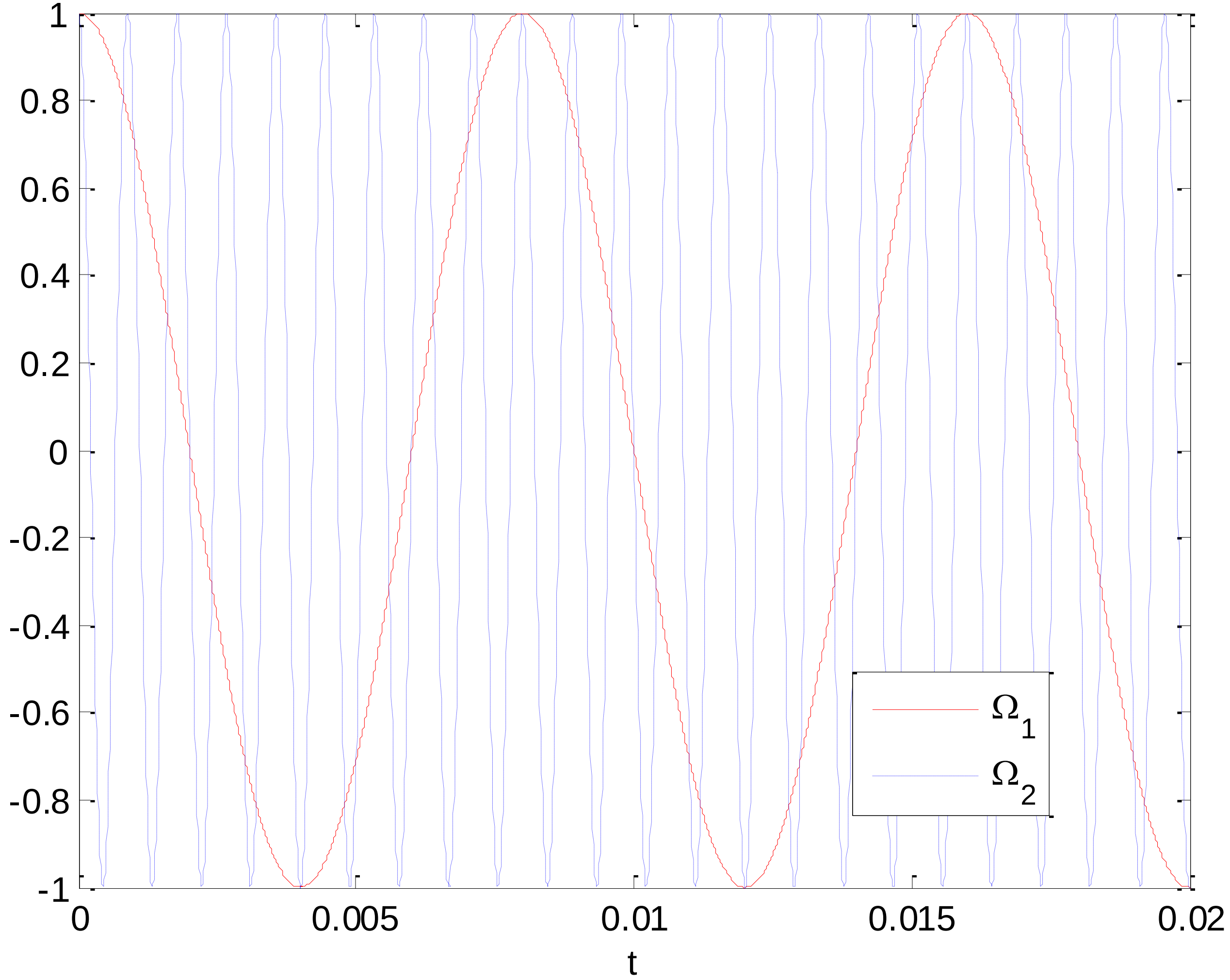

Fig.4.11: Continuous-time sinusoids

Passing $x[n]$ through the DC converter only produces $\cos(\Omega_1 t)$ but not $\cos(\Omega_2 t)$

The Nyquist frequency of $\cos(\Omega_2 t)$ is $2250\pi \ \mathrm{rads}^{-1}$ and hence the sampling frequency without aliasing is $\Omega_s > 4500\pi$

Given $F_s = 1000$ Hz or $\Omega_s = 2000\pi \quad \mathrm{rads}^{-1}$, $\cos(\Omega_2 t)$ does not correspond to $x[n]$

We can recover $x_r(t) = \cos(\Omega_1 t)$ because the Nyquist frequency and Nyquist rate for $\cos(\Omega_1 t)$ are $250\pi \ \mathrm{rads}^{-1}$ and $500\pi \ \mathrm{rads}^{-1}$

Based on (4.11), $x_r(t) = \cos(\Omega_1 t)$ is:

$$x_r(t) = \sum_{k=-\infty}^{\infty} x[k]\mathrm{sinc}\left(\frac{t-kT}{T}\right) \approx \sum_{k=-10}^{30} x[k]\mathrm{sinc}\left(\frac{t-kT}{T}\right)$$

with $T = 1/1000$ s

The MATLAB code for reconstructing $\cos(\Omega_1 t)$ is:

```
n=-10:30;           %add 20 past and future samples
x=cos(pi.*n./4);
T=1/1000;           %sampling interval is 1/1000
for l=1:2000        %observed interval is [0,0.02]
t=(l-1)*T/100;%successive sample separation is 0.01T
h=sinc((t-n.*T)./T);
xr(l)=x*h.'; %approximate interpolation of (4.11)
end
```

We compute 2000 samples of $x_r(t)$ in $t \in [0, 0.02]$s

The value of each $x_r(t)$ at time `t` is approximated as `x*h.'` where the sinc vector is updated for each computation

The MATLAB program is provided as `ex4_4.m`

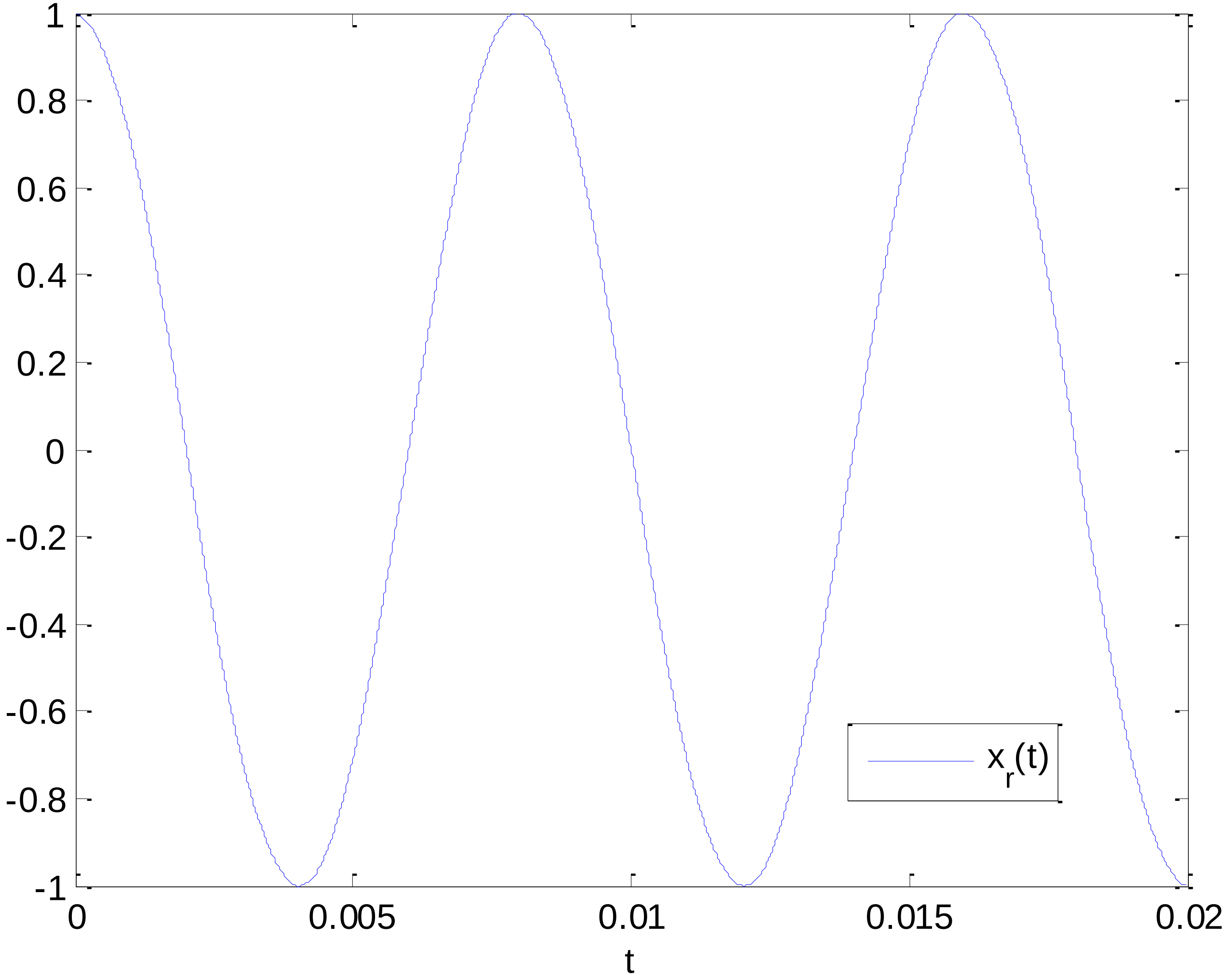

Fig.4.12: Reconstructed continuous-time sinusoid

Example 4.5

Play the sound for a discrete-time tone using MATLAB. The frequency of the corresponding analog signal is 440 Hz which corresponds to the A note in the American Standard pitch. The sampling frequency is 8000 Hz and the signal has a duration of 0.5 s.

The MATLAB code is

```
A=sin(2*pi*440*(0:1/8000:0.5));%discrete-time A
sound(A,8000);                 %DA conversion and play
```

Note that sampling frequency in Hz is assumed for `sound`