Discrete Time Signals

Jee-Hwan Ryu

School of Mechanical Engineering
Korea University of Technology and Education

Sampled Signals

- Converting continuous time signal into discrete time system is unambiguous—e.g., sample value every $T$ [note that we are throwing away information]

Given $f(t)$ to be a continuous time signal, $f(kT)$ is the value of $f(t)$ at $t = kT$. The discrete-time signal (or sequence) $f[k]$ is defined only for $k$ an integer. So if we derive $f[k]$ from $f(t)$ by sampling every $T$ seconds, where $T$ is the sample period, we get:

$$f[k] = f(kT) = f(t)_{t=kT}$$
Discrete Time Unit Step Function

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

Alternate notation: \(1[k]\)

Time-shifted unit step function

\[ u[n-n_0] = \begin{cases} 
1, & n \geq n_0 \\
0, & n < n_0 
\end{cases} \]

Let \(n_0 = 4\)
Discrete Time Unit Impulse Function

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

Shifted Impulse Function

\[ \delta[n-n_0] = \begin{cases} 
1, & n = n_0 \\
0, & n \neq n_0 
\end{cases} \]

Let \( n_0 = 3 \)
Comparison—DT and CT

<table>
<thead>
<tr>
<th>Continuous time</th>
<th>Discrete time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u(t) = \int_{-\infty}^{t} \delta(\tau) d\tau$</td>
<td>$u[n] = \sum_{k=-\infty}^{n} \delta[k]$</td>
</tr>
<tr>
<td>$\delta(t) = \frac{d}{dt} u(t)$</td>
<td>$\delta[n] = u[n] - u[n-1]$</td>
</tr>
<tr>
<td>$x(t)\delta(t-t_0) = x(t_0)\delta(t-t_0)$</td>
<td>$x[n]\delta[n-n_0] = x[n_0]\delta[n-n_0]$</td>
</tr>
<tr>
<td>$\int_{-\infty}^{\infty} x(t)\delta(t-t_0) dt = x(t_0)$</td>
<td>$\sum_{n=-\infty}^{\infty} x[n]\delta[n-n_0] = x[n_0]$</td>
</tr>
</tbody>
</table>

Adding and Subtracting Signals

- Do it "point by point"
- Can do using a table, or graphically (or by computer program)
- Example: $x[n] = u[n] - u[n - 4]$

<table>
<thead>
<tr>
<th>$n$</th>
<th>$\leq -1$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>$\geq 4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x[n]$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Adding and Subtracting Signals

- Example: \( x[n] = u[n] - u[n-4] \)

\[
\begin{array}{c}
\hline
0 & 1 & 2 & 3 & 4 & 5 & \cdots \\
\hline
\end{array}
\quad - \quad
\begin{array}{c}
\hline
1 & 2 & 3 & 4 & 5 & 6 & \cdots \\
\hline
\end{array}
\quad = \quad
\begin{array}{c}
\hline
1 & 2 & 3 & \cdots \\
\hline
\end{array}
\]

Time-Reversal of a Signal

\( y[n] = x[m]|_{m=-n} = x[-n] \)

This reversal operation precisely flips a signal about the vertical axis.
Time-Reversal of a Signal

Time-Scaling a Signal

\[ y[n] = x[m] \big|_{m=an} = x[an] \]

If \(|a| > 1\), then SPEED UP by a factor of \(a\). If \(|a| < 1\), then SLOW DOWN by a factor of \(a\).

Unlike continuous time, there are restrictions on \(a\)!

For speeding up (also known as “subsampling”), \(a\) must be an integer.
Time-Scaling a Signal-Subsampling

Find $w_1[n] = x[2n]$ and $w_2[n] = x[2n+1]$

Time-Scaling a Signal-Subsampling

$w_1[n] = x[2n]$
Time-Scaling a Signal-Subsampling

\[
x[n]
\]

\[
w_2[n] = x[2n+1]
\]

Same in this case

Time-Scaling a Signal-Subsampling

Try a different example…

\[
y[n]
\]

Find \( w_3[n] = y[2n] \) and \( w_4[n] = y[2n+1] \)
Time-Scaling a Signal - Subsampling

\[ w_3[n] = y[2n] \]

\[ w_4[n] = y[2n+1] \]

Time-Scaling a Signal - Slowing Down

For slowing down (expanding) a signal, you need \( a = 1/K \) where \( K \) is an integer.

Example: Let \( K = 2 \) (\( a = 1/2 \)) and find \( z[n] = b[n/2] \)

<table>
<thead>
<tr>
<th>( n )</th>
<th>( z[n] )</th>
<th>( b[n/2] )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>( z[0] )</td>
<td>( b[0] )</td>
</tr>
<tr>
<td>1</td>
<td>( z[1] )</td>
<td>??</td>
</tr>
<tr>
<td>2</td>
<td>( z[2] )</td>
<td>( b[1] )</td>
</tr>
<tr>
<td>3</td>
<td>( z[3] )</td>
<td>??</td>
</tr>
</tbody>
</table>

Values like \( b[\frac{1}{2}] \) and \( b[\frac{3}{2}] \) are not defined so how do we find \( z[1] \) and \( z[3] \)??
Time-Scaling a Signal-Slowing Down

One solution is to **INTERPOLATE**. A simple, but sub-optimal interpolation rule is linear interpolation

\[
z[n] = \begin{cases} 
    b[n/2], & n \text{ even} \\
    \frac{1}{2} \{b[(n-1)/2] + b[(n+1)/2]\}, & n \text{ odd}
\end{cases}
\]

<table>
<thead>
<tr>
<th>( n )</th>
<th>( z[n] )</th>
<th>( b[n/2] )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>( z[0] )</td>
<td>( b[0] )</td>
</tr>
<tr>
<td>1</td>
<td>( z[1] )</td>
<td>??</td>
</tr>
<tr>
<td>2</td>
<td>( z[2] )</td>
<td>( b[1] )</td>
</tr>
<tr>
<td>3</td>
<td>( z[3] )</td>
<td>??</td>
</tr>
</tbody>
</table>

Interpolation can be used in a simple compression scheme – just transmit every other sample and fill in missing the values at the receiver.

Recall earlier example…

\[
w_3[n] = y[2n] \quad \text{and} \quad w_4[n] = y[2n+1]
\]

What does \( z_3[n] = w_3[n/2] \) look like? (compute from \( w_3 \), using table and **linear interpolation**)

Looks just like \( y[n] \)!
Recall earlier example...

\[ w_3[n] = y[2n] \quad \text{and} \quad w_4[n] = y[2n+1] \]

What does \( z_4[n] = w_4[n/2] \) look like? (compute from \( w_4 \), using table and linear interpolation)

Looks just like \( w_4[n] \)!

Recall earlier example...

\[ w_3[n] = y[2n] \quad \text{and} \quad w_4[n] = y[2n+1] \]

What does \( y[n/2] \) look like, using linear interpolation?
More Time-Shifting

Ex.: Given $x[n] = a^n u[n], \ |a| < 1$, find and plot $y[n] = x[n-3]$
Combining Time Shifting and Scaling

Ex. Find \( u[3-n] \),

There are two direct ways to solve this example:

**Method 1.** Reverse (time scale of -1) then shift (delay) in time \((x[a(n+\frac{b}{a})])\):

\[
z[n] = u[-n]
\]
\[
y[n] = z[n-3] = u[-(n-3)] = u[-n + 3]
\]

**Method 2.** Advance in time then reverse \((x[an+b])\):

\[
w[n] = u[n + 3]
\]
\[
y[n] = w[-n] = u[-n + 3]
\]
Combining Time Shifting and Scaling

Be careful—for some cases method 1 doesn't work!
For example, if you want to form

\[ z[n] = x[3 - 2n] = x[-2 \left( n - \frac{3}{2} \right)] , \]

What does it mean to shift a signal by \( 3/2 \)? To make sure, plug values into the table to check:

<table>
<thead>
<tr>
<th>( n )</th>
<th>( z[n] )</th>
<th>( x[3 - 2n] )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>( z[0] )</td>
<td>( x[3] )</td>
</tr>
<tr>
<td>1</td>
<td>( z[1] )</td>
<td>( x[1] )</td>
</tr>
<tr>
<td>2</td>
<td>( z[2] )</td>
<td>( x[-1] )</td>
</tr>
<tr>
<td>-1</td>
<td>( z[-1] )</td>
<td>( x[5] )</td>
</tr>
<tr>
<td>-2</td>
<td>( z[-2] )</td>
<td>( x[7] )</td>
</tr>
</tbody>
</table>

For other cases, method 2 doesn't work

\[ z[n] = x[3 - 2n] \]

**Ex.** Let \( x[n] = 2u[n + 2] \). Find \( z[n] = x[3 - 2n] \).

\[ x[n] = 2u[n + 2] \]

\[ z[n] = x[3 - 2n] \]

Get this from table, using \( x[n] \) values. Note method 1 problem described before this example.
Combining Time Shifting and Scaling

Ex. Find \( y[n] = x[2 - 2n] \):
\[
x[2 - 2n] = x[-2(n-1)]
\]
For method 1 approach, \( y[n] = x[-2n] \), then delay by 1. Or, just plug in values of \( n \) in a table.

Ex. Let \( y[n] = a^n u[n] \), where \( a > 1 \). Find and plot \( z[n] = y[-2n + 2] \).

\[
\begin{align*}
v[0] &= y[-2(0) + 2] = y[2] = a^2 \\
v[1] &= y[-2(1) + 2] = y[0] = 1 \\
v[2] &= y[-2(2) + 2] = y[-2] = 0 \\
v[-1] &= y[-2(-1) + 2] = y[4] = a^4 \\
\end{align*}
\]
Amplitude Scaling

- Do point by point

Example: Find \(x[n] = (u[n+1] - u[n-5])(nu[2-n])\)
Even and Odd Signals

Any discrete-time signal can be expressed as the sum of an even signal and an odd signal.

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

Even: \( x_e[n] = x_e[-n] \)
Odd: \( x_o[n] = -x_o[-n] \)

\[ x_e[n] = \frac{1}{2}(x[n] + x[-n]) \]
\[ x_o[n] = \frac{1}{2}(x[n] - x[-n]) \]
\[ x[n] = x_e[n] + x_o[n] \]

Ex. Given \( x[n] \), find \( x_e[n] \) and \( x_o[n] \).