Lecture 8: Interpolation

Mark Hasegawa-Johnson
All content CC-SA 4.0 unless otherwise specified.

ECE 401: Signal and Image Analysis, Fall 2022
1. Review: Sampling

2. Interpolation: Discrete-to-Continuous Conversion

3. Summary
Outline

1. Review: Sampling
2. Interpolation: Discrete-to-Continuous Conversion
3. Summary
Suppose you have some continuous-time signal, $x(t)$, and you’d like to sample it, in order to store the sample values in a computer. The samples are collected once every $T_s = \frac{1}{F_s}$ seconds:

$$x[n] = x(t = nT_s)$$
Outline

1. Review: Sampling
2. Interpolation: Discrete-to-Continuous Conversion
3. Summary
How can we get $x(t)$ back again?

We’ve already seen one method of getting $x(t)$ back again: we can find all of the cosine components, and re-create the corresponding cosines in continuous time.

There is an easier way. It involves multiplying each of the samples, $x[n]$, by a short-time pulse, $p(t)$, as follows:

$$y(t) = \sum_{n=-\infty}^{\infty} y[n] p(t - nT_s)$$
Rectangular pulses

For example, suppose that the pulse is just a rectangle,

\[
p(t) = \begin{cases} 
1 & -\frac{T_s}{2} \leq t < \frac{T_s}{2} \\
0 & \text{otherwise}
\end{cases}
\]
Rectangular pulses = Piece-wise constant interpolation

The result is a piece-wise constant interpolation of the digital signal:

Discrete-time signal $x[n]$

$x[n]$ interpolated using a rectangular pulse
The rectangular pulse has the disadvantage that $y(t)$ is discontinuous. We can eliminate the discontinuities by using a triangular pulse:

$$p(t) = \begin{cases} 
1 - \frac{|t|}{T_s} & -T_s \leq t < T_s \\
0 & \text{otherwise}
\end{cases}$$
Triangular pulses = Piece-wise linear interpolation

The result is a piece-wise linear interpolation of the digital signal:
Cubic spline pulses

The triangular pulse has the disadvantage that, although $y(t)$ is continuous, its first derivative is discontinuous. We can eliminate discontinuities in the first derivative by using a cubic-spline pulse:

$$p(t) = \begin{cases} 
1 - \frac{3}{2} \left( \frac{|t|}{T_s} \right)^2 + \frac{1}{2} \left( \frac{|t|}{T_s} \right)^3 & 0 \leq |t| \leq T_s \\
-\frac{3}{2} \left( \frac{|t|-2T_s}{T_s} \right)^2 \left( \frac{|t|-T_s}{T_s} \right) & T_s \leq |t| \leq 2T_s \\
0 & \text{otherwise}
\end{cases}$$
The triangular pulse has the disadvantage that, although \( y(t) \) is continuous, its first derivative is discontinuous. We can eliminate discontinuities in the first derivative by using a cubic-spline pulse:
Cubic spline pulses = Piece-wise cubic interpolation

The result is a piece-wise cubic interpolation of the digital signal:
The cubic spline has no discontinuities, and no slope discontinuities, but it still has discontinuities in its second derivative and all higher derivatives. Can we fix those? The answer: yes! The pulse we need is the inverse transform of an ideal lowpass filter, the sinc.
Sinc pulses

We can reconstruct a signal that has no discontinuities in any of its derivatives by using an ideal sinc pulse:

$$p(t) = \frac{\sin(\pi t/T_S)}{\pi t/T_S}$$
Sinc pulse = ideal bandlimited interpolation

The result is an ideal bandlimited interpolation:

Discrete-time signal $x[n]$

$x[n]$ interpolated using a sinc pulse
Outline

1. Review: Sampling
2. Interpolation: Discrete-to-Continuous Conversion
3. Summary
Summary

- Piece-wise constant interpolation = interpolate using a rectangle
- Piece-wise linear interpolation = interpolate using a triangle
- Cubic-spline interpolation = interpolate using a spline
- Ideal interpolation = interpolate using a sinc