Lecture 7: Interpolation

Mark Hasegawa-Johnson
These slides are in the public domain.

ECE 401: Signal and Image Analysis, Fall 2023

1. Review: Sampling
2. Interpolation: Discrete-to-Continuous Conversion
3. Interpolation: Upsampling a signal
4. Summary

Outline

1. Review: Sampling
2. Interpolation: Discrete-to-Continuous Conversion
3. Interpolation: Upsampling a signal
4. Summary

How to sample a continuous-time signal

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. Interpolation: Upsampling a signal
4. 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.

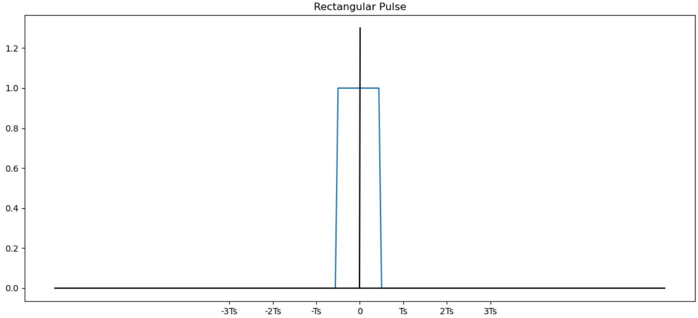
There is a more general method, that we can use for any signal, even signals that are not composed of pure tones. It involves multiplying each of the samples, $x[n]$, by a short-time pulse, $p(t)$, as follows:

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

Rectangular pulses

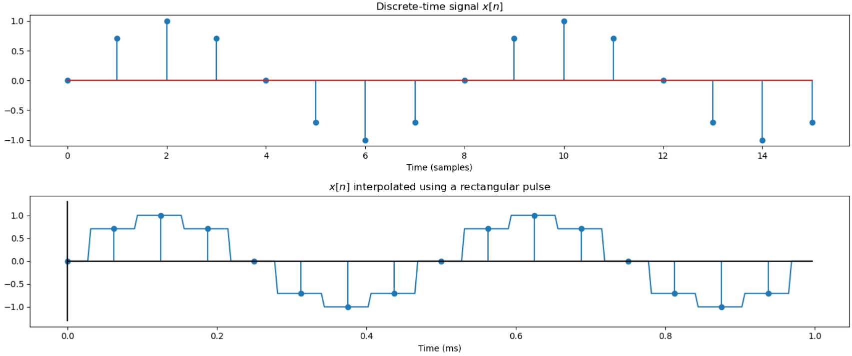
For example, suppose that the pulse is just a rectangle,

$$p(t) = \begin{cases} 1 & -\frac{T_S}{2} \le t < \frac{T_S}{2} \\ 0 & \text{otherwise} \end{cases}$$



Rectangular pulses = Piece-wise constant interpolation

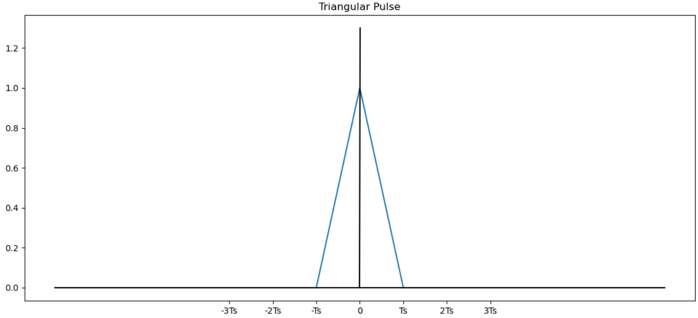
The result is a piece-wise constant interpolation of the digital signal:



Triangular pulses

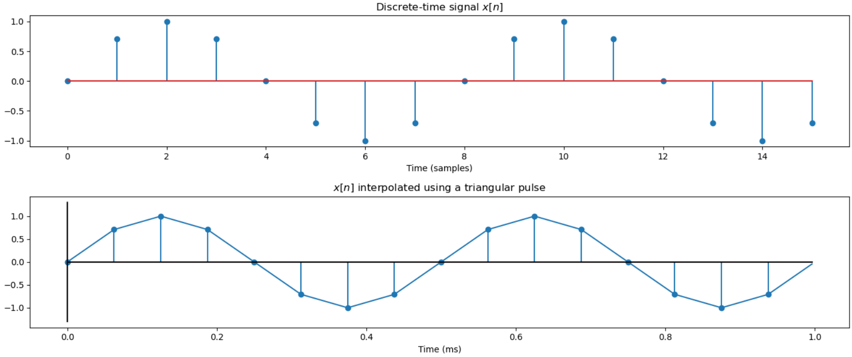
The rectangular pulse has the disadvantage that $y(t)$ is discontinuous. We can eliminate the discontinuities by using a triangular pulse:

$$p(t) = \begin{cases} 1 - \frac{|t|}{T_S} & -T_S \leq t < T_S \\ 0 & \text{otherwise} \end{cases}$$



Triangular pulses = Piece-wise linear interpolation

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



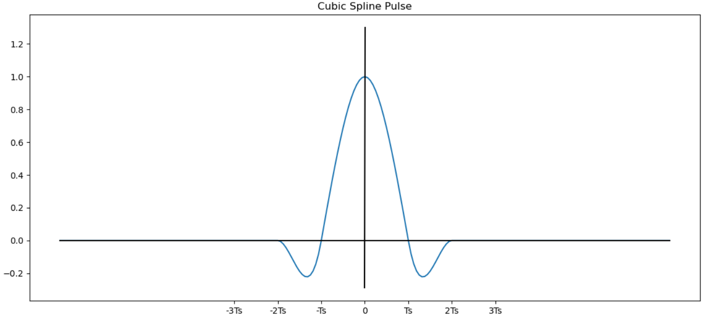
Cubic spline pulses

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:

$$p(t) = \begin{cases} 1 - \frac{3}{2}\left(\frac{|t|}{T_S}\right)^2 + \frac{1}{2}\left(\frac{|t|}{T_s}\right)^3 & 0 \le |t| \le T_S \\ -\frac{3}{2}\left(\frac{|t|-2T_s}{T_S}\right)^2\left(\frac{|t|-T_s}{T_S}\right) & T_S \le |t| \le 2T_S \\ 0 & \text{otherwise} \end{cases}$$

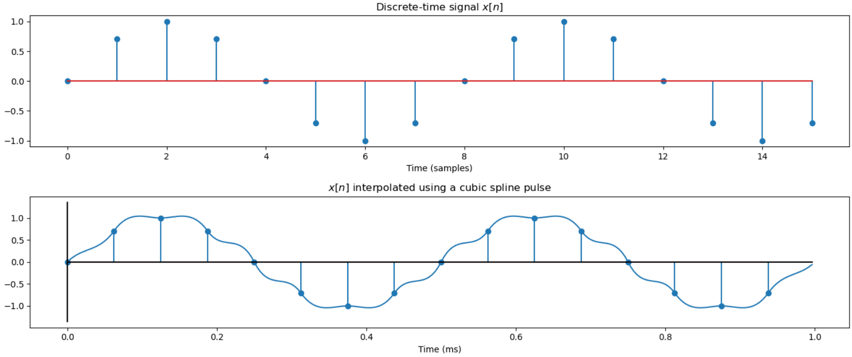
Cubic spline pulses

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:



Cubic spline pulses = Piece-wise cubic interpolation

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



Sinc pulses

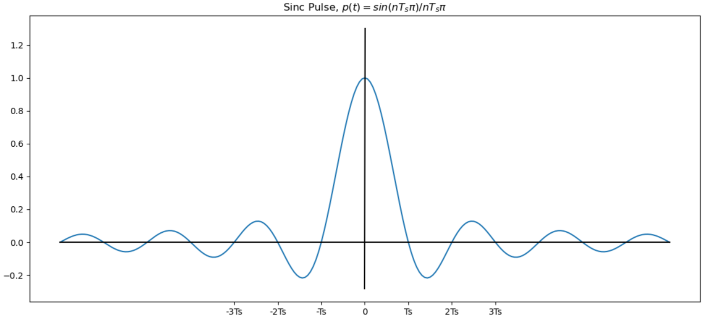
The cubic spline has no discontinuities, and no slope discontinuities, but it still has discontinuities in its second derivative and all higher derivatives. Can we fix those?

The answer: yes! The pulse we need is the inverse transform of an ideal lowpass filter, the sinc.

Sinc pulses

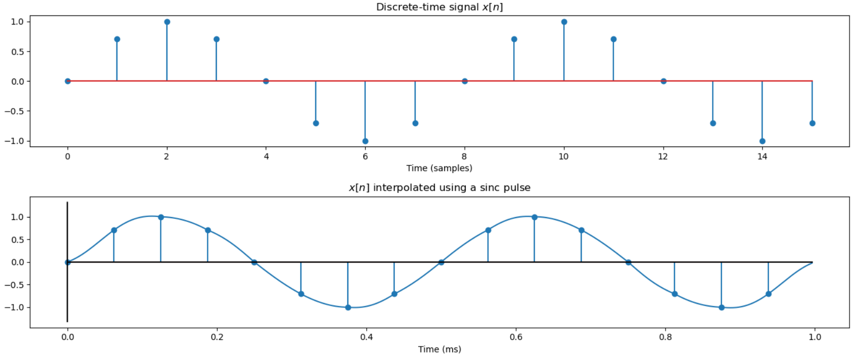
We can reconstruct a signal that has no discontinuities in any of its derivatives by using an ideal sinc pulse:

$$p(t) = \frac{\sin(\pi t / T_S)}{\pi t / T_S}$$



Sinc pulse = ideal bandlimited interpolation

The result is an ideal bandlimited interpolation:



Outline

1 Review: Sampling

2 Interpolation: Discrete-to-Continuous Conversion

3 Interpolation: Upsampling a signal

4 Summary

Changing the sampling rate of a signal

Suppose we have an audio signal ($x[n]$) sampled at 11025 samples/second, but we really want to play it back at 44100 samples second. We can do that by creating a new signal, $y[n]$, at $M = 4$ times the sampling rate of $x[n]$:

$$y[n] = \begin{cases} x[n/M] & n = \text{integer multiple of } M \\ \text{interpolated value} & \text{otherwise} \end{cases}$$

Upsampling

We split this process into two steps. First, **upsampling** means that we just insert zeros between the samples of $x[n]$:

$$u[n] = \begin{cases} x[n/M] & n = \text{integer multiple of } M \\ 0 & \text{otherwise} \end{cases}$$

Interpolation

Second, we generate the missing samples by interpolation:

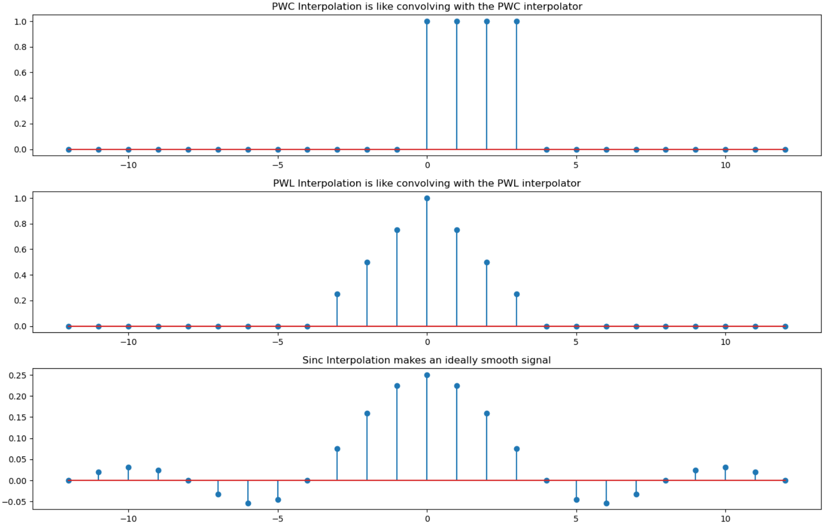
$$
\begin{aligned}
y[n] &= \sum_{m=-\infty}^{\infty} u[m]p[n-m] \\
&= \begin{cases} x[n/M] & n = \text{integer multiple of } M \\ \text{interpolated value} & \text{otherwise} \end{cases}
\end{aligned}
$$

The second line of the equality holds if

$$
p[n] = \begin{cases} 1 & n = 0 \\ 0 & n = \text{nonzero integer multiple of } M \\ \text{anything} & \text{otherwise} \end{cases}
$$

Interpolation Kernels

All of these interpolation kernels satisfy the condition on the previous slide:



Outline

1 Review: Sampling

2 Interpolation: Discrete-to-Continuous Conversion

3 Interpolation: Upsampling a signal

4 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