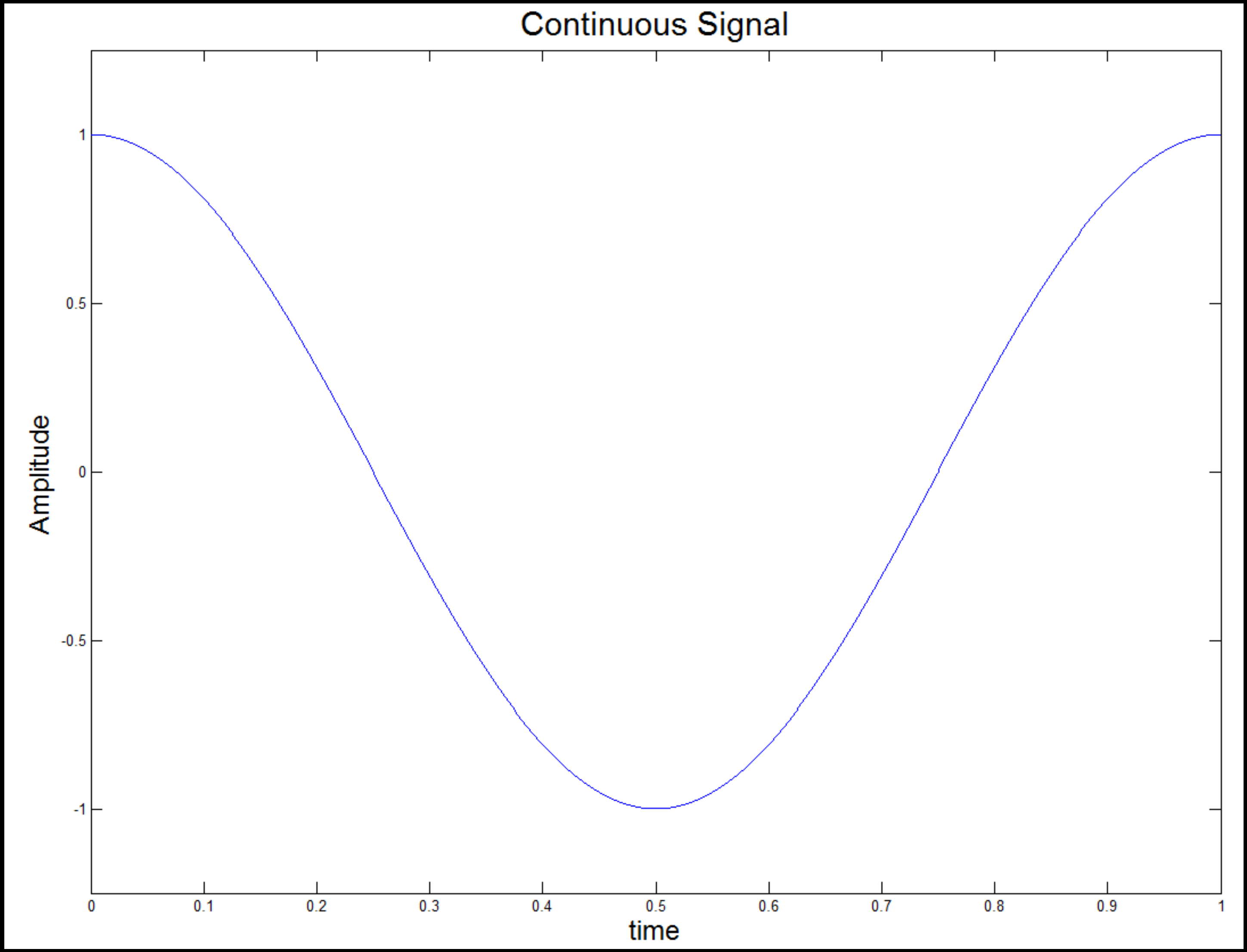
Sampling and Signal Reconstruction

Lab 10

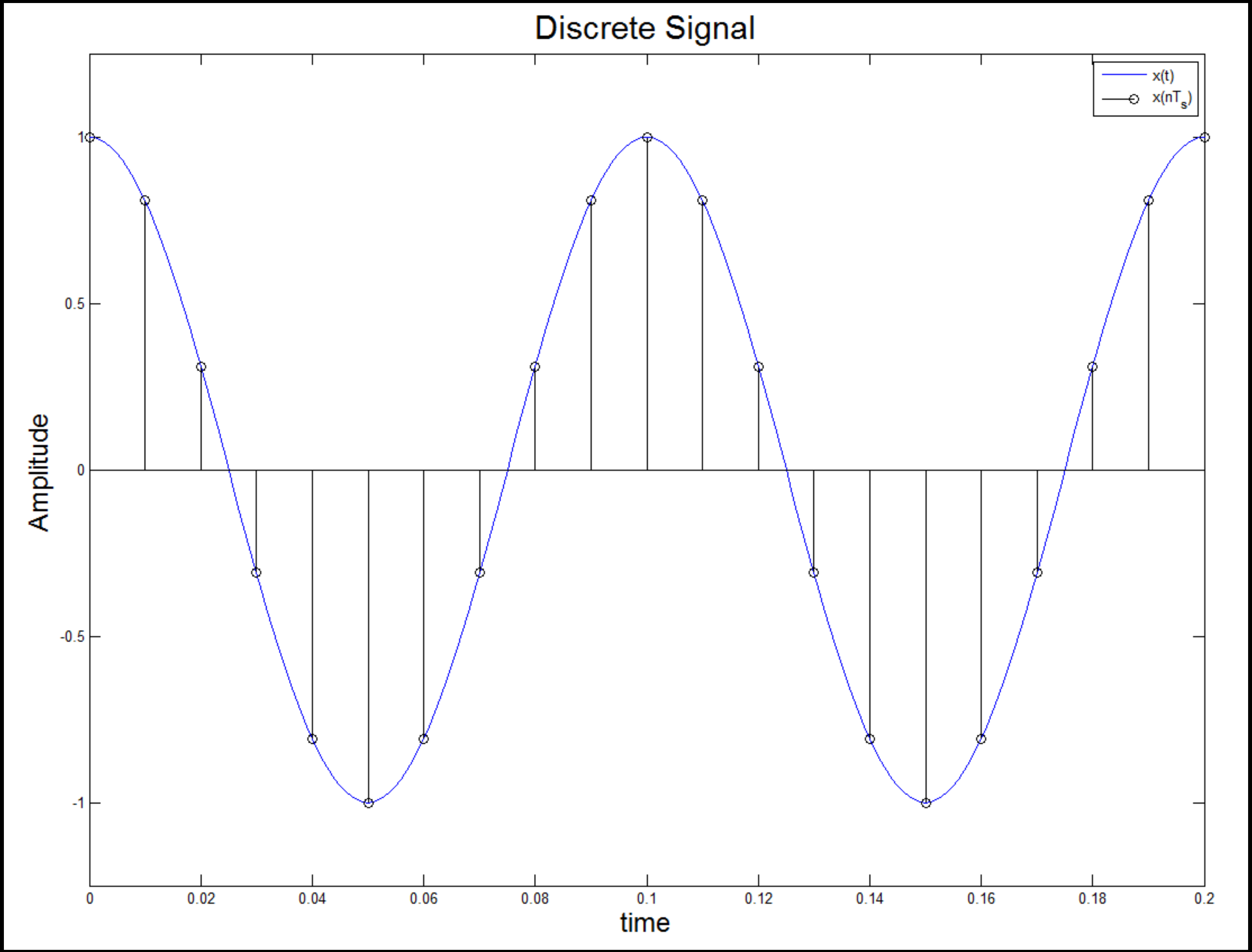
$x(t)$

Continuous Signals



$X(nT_s) \rightarrow x[n]$

Sampling



Nyquist Sampling Theorem

$$\therefore x(t)\delta_{T_s} \overset{\mathcal{F}}{\leftrightarrow} \frac{1}{T_s}\delta_{f_s} * X(f)$$

Remember this result? It says that when you sample a signal every T_s in the time domain the frequency domain is periodic (it repeats every f_s). The “copies” that occur every multiple of $\pm f_s$ are called aliases.

Reconstruction

In some instances we would like to reconstruct the original signal.

$$x[n] \rightarrow X(nT_s)$$

This may not yield a signal that even remotely resembles the original signal!

Nyquist in terms of Reconstruction

If the sampling rate, f_s, is not large enough (larger than twice the bandlimit, f_m) then the aliases will overlap: an effect known as Aliasing.

$$\boxed{f_s > 2f_m}$$

<u>If and only if</u> a signal is sampled at this frequency (or above) can the original signal be reconstructed in the time-domain.

Reconstruction Methods

Zeroth-Order Interpolation

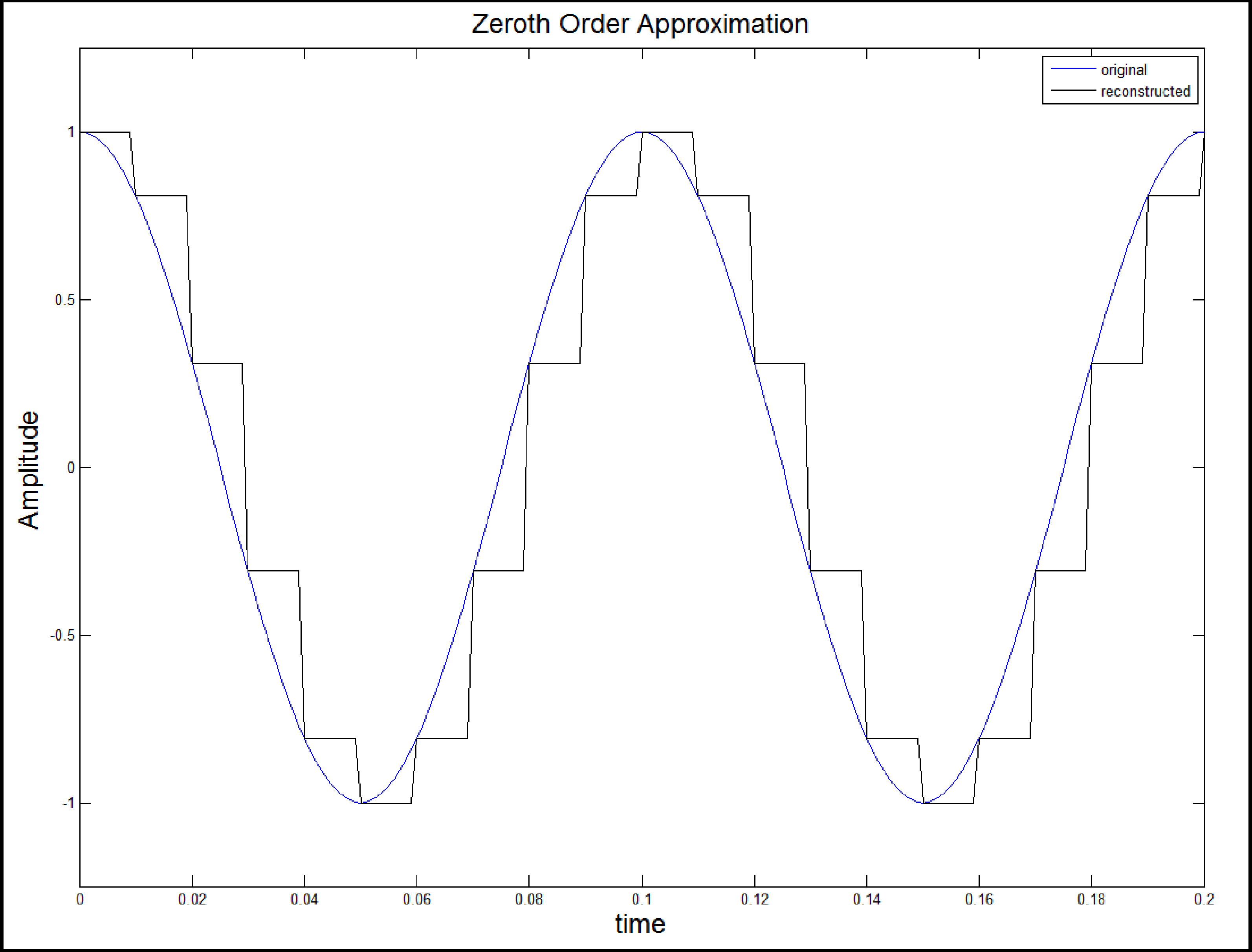
Zeroth-Order Interpolation means we accept the value on the discrete sample for the time window that the sample was taken from originally.

This is basically just approximating each time window with a constant.

Zeroth-Order Code

```
clc,clear,close all
deltat=0.01;            %time window
n=0:deltat:1;           %time index
N=length(n);            %number of sampled points
x=cos(20*pi*n);         %signal
ta=0:0.001:1;           %reconstruction time
y1=[];
for i=1:N-1
        y1=[y1 ones(1,10)*x(i)];
end
y1=[y1 x(end)];         %it was one element too short
```

Zeroth- Order Interpolation

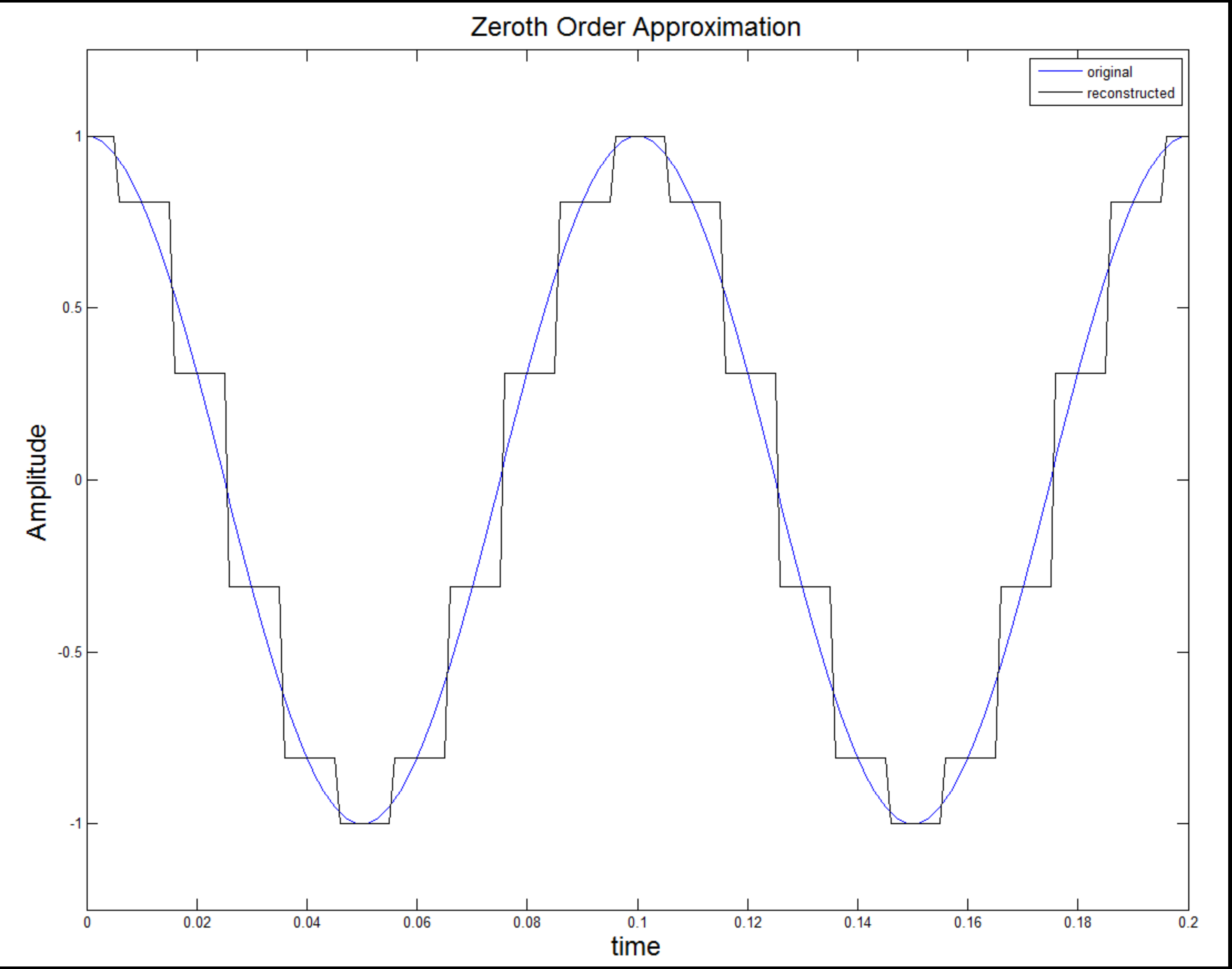


Zeroth-Order Code

```
clc,clear,close all
deltat=0.01;            %time window
n=0:deltat:1;           %time index
N=length(n);            %number of sampled points
x=cos(20*pi*n);         %signal
ta=0:0.001:1;           %reconstruction time
y1=[];
for i=1:N-1
        y1=[y1 ones(1,10)*x(i)];
end
y1=[y1 x(end)];
y1=[y1(5:end) x(1:4)];
```

Phase correction, this works because the signal is periodic!

Zeroth-Order Interpolation – With phase correction



rectpuls()

Matlab has a function which does this zeroth-order interpolation. It’s called rectpuls().

This function operates by multiplying each sampled amplitude by a shifted and compressed rectangle pulse signal.

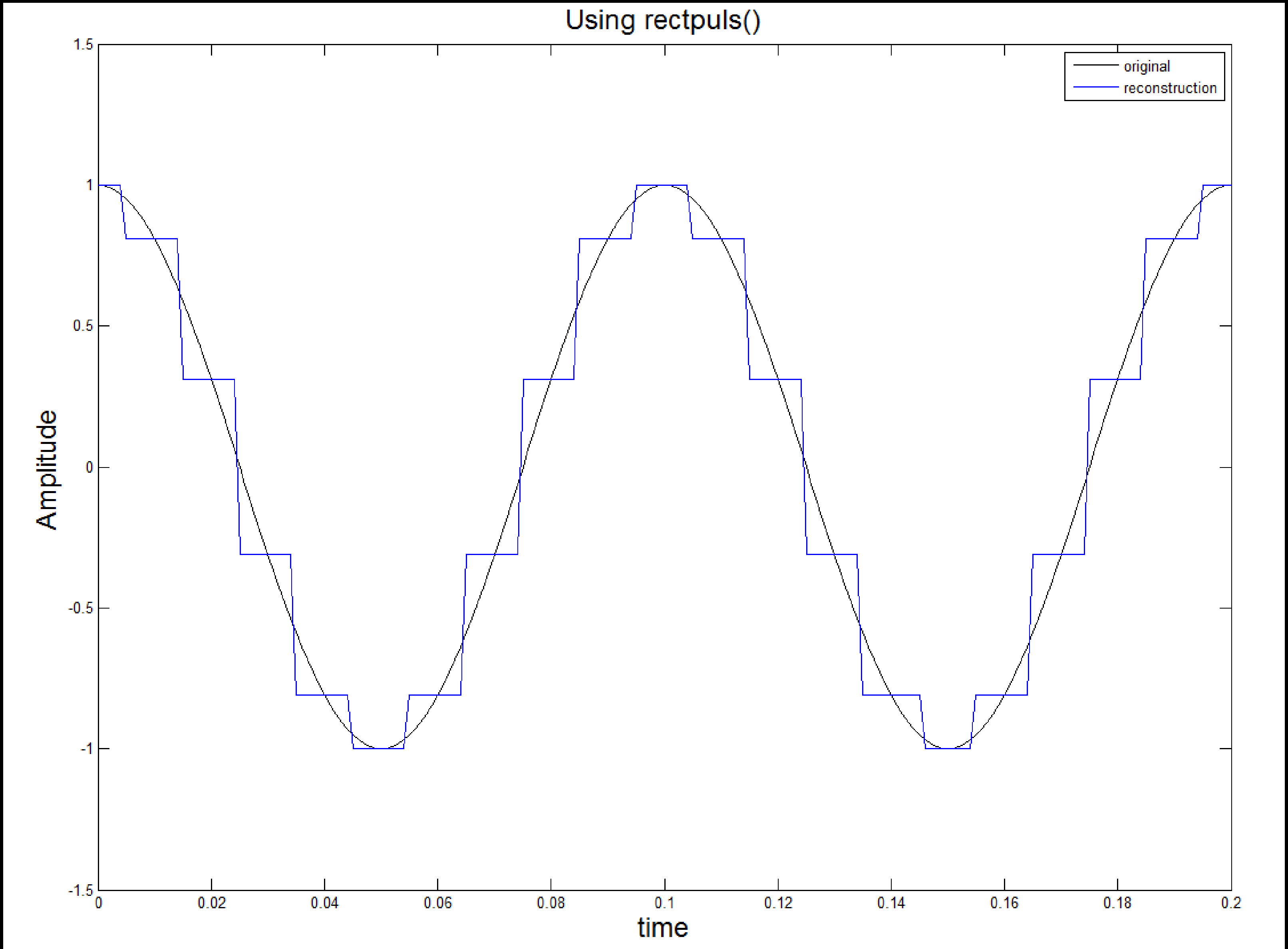
Code using rectplus()

```
Ts=0.01;                                   %time window
n=0:Ts:1;                                  %time index
Fs=1/Ts;                                   %sample rate
N=length(n);                               %number of sampled points
x=cos(20*pi*n);                            %original signal
ta=0:0.001:1;                              %reconstruction time
 y=zeros(N,length(ta));                    %reconstruction vector
for i=1:N
   y(i,:)=x(i)*rectpuls(Fs*ta-i+1);
end
plot(ta,sum(y))
```

time shift

compression

Same result as Zeroth-Order Approximation!



Matrix Operations instead of For-Loop

```
Ts=0.01;                    %time window
n=0:Ts:1;                   %time index
Fs=1/Ts;                    %sample rate
N=length(n);                %number of sampled points
x=cos(20*pi*n);             %original signal

ta=0:0.001:1;               %reconstruction time
Na=length(ta);              %reconstruction length

y=x*rectpuls(Fs*(ones(N,1)*ta-n'*ones(1,Na)));
```

Left for your report

Reconstruct the signal using tripuls() and sinc().

Higher-Order Methods

Cubic Spline Interpolation

There is a higher order polynomial interpolation known as the spline method.

$$Y_i(t) = a_i + b_i t + c_i t^2 + d_i t^3$$

This approximation is used a lot as it results in a very smooth curve.

Spline Code

```
Ts=0.01;            %time window
n=0:Ts:1;           %time index
x=cos(20*pi*n);     %sampled signal
ta=0:0.001:1;       %reconstruction time
```

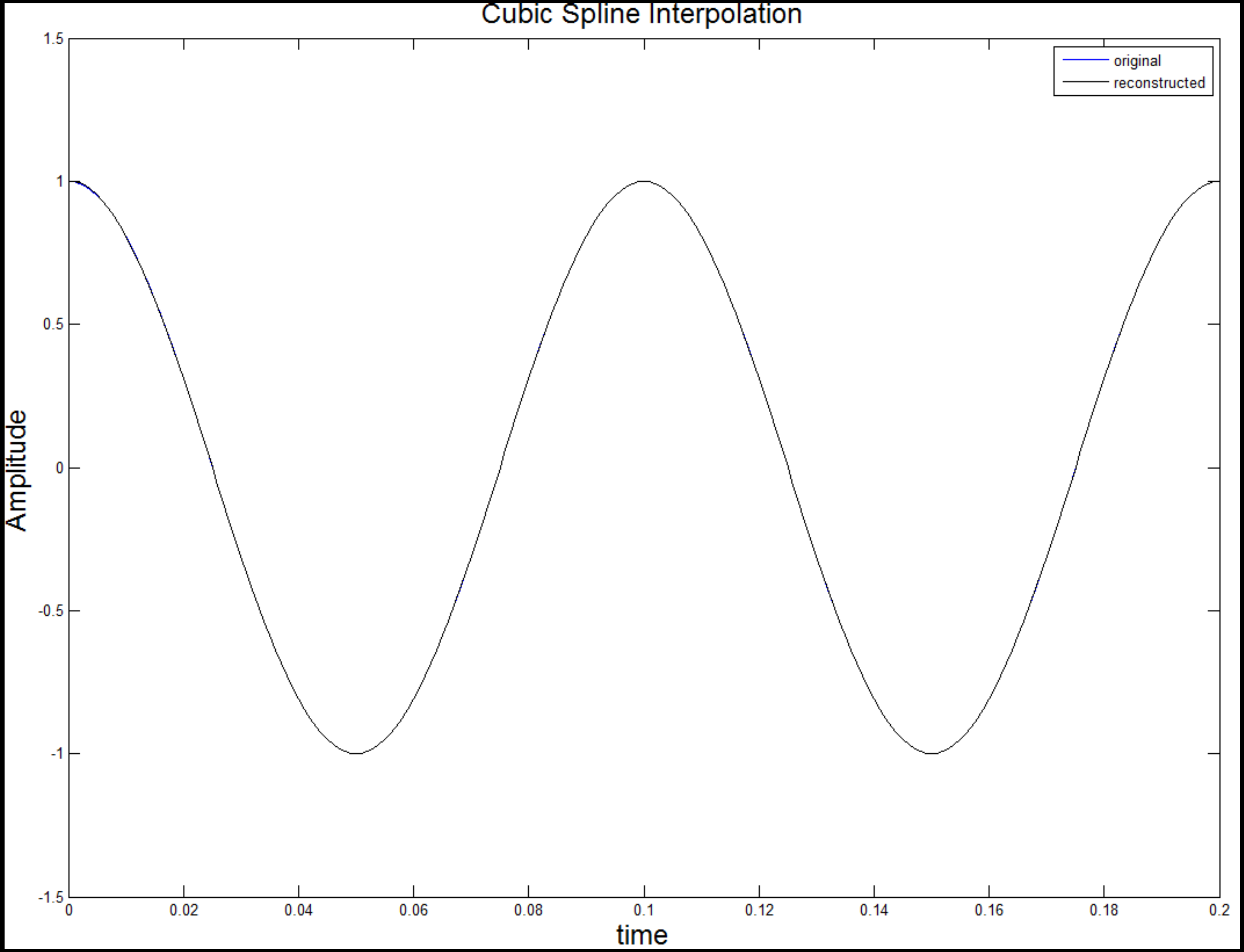
Sampled Amplitudes

```
y=spline(n,x,ta);
```

Reconstruction Indexes

Sampled Indexes

Cubic Spline Interpolation



Why is there only one?