Linear Systems and Signals

Representing signals computationally

Anand D. Sarwate

Department of Electrical and Computer Engineering
Rutgers, The State University of New Jersey

2020

Learning objectives

The learning objectives for this section are:

- represent CT signals in numerical simulation using the sampling frequency
- generate and plot standard DT and CT signals mentioned above
- window signals in time

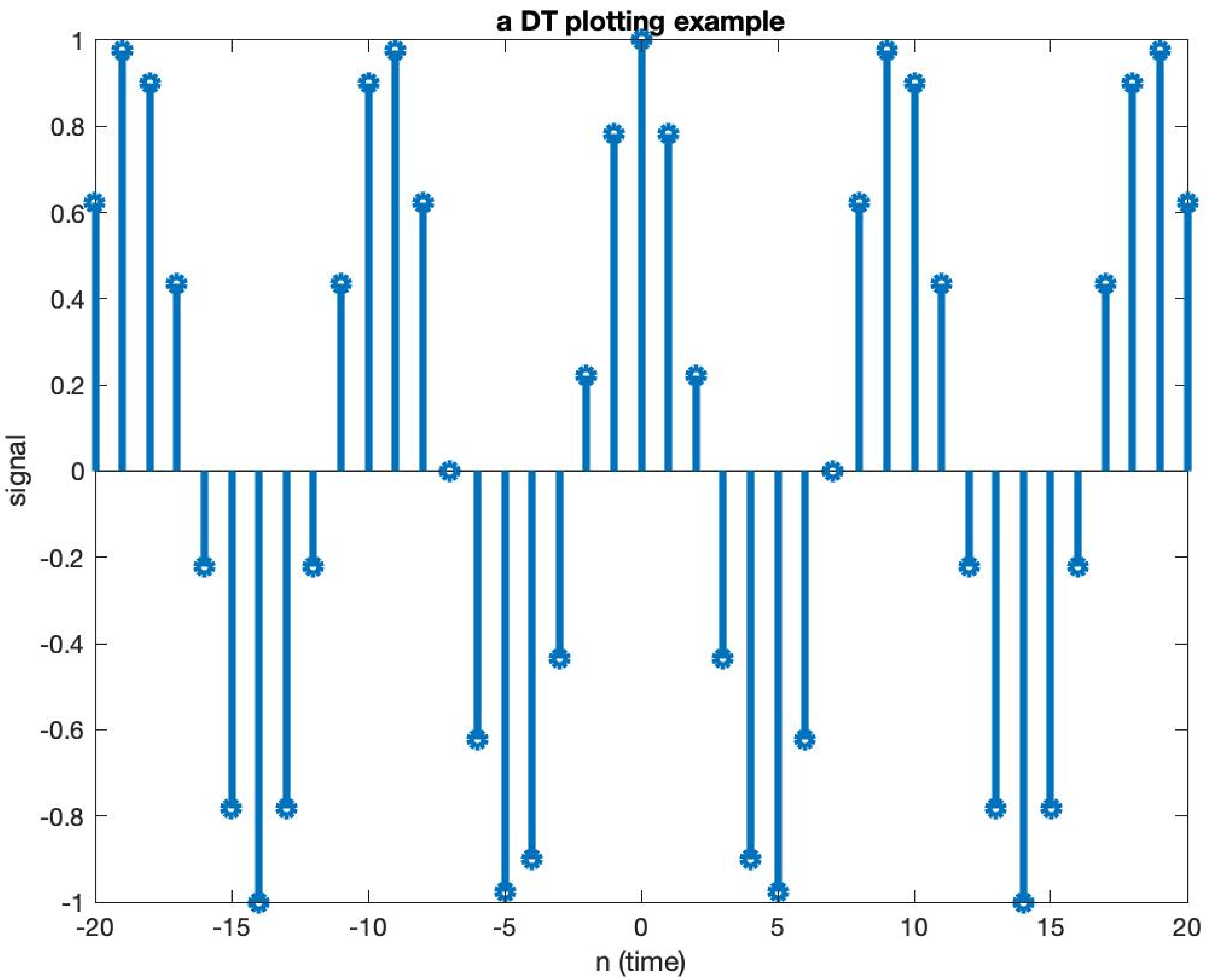
Representing a signal in DT

Representing a signal in DT in MATLAB is easy. It's just a vector. You can plot it using `stem()`.

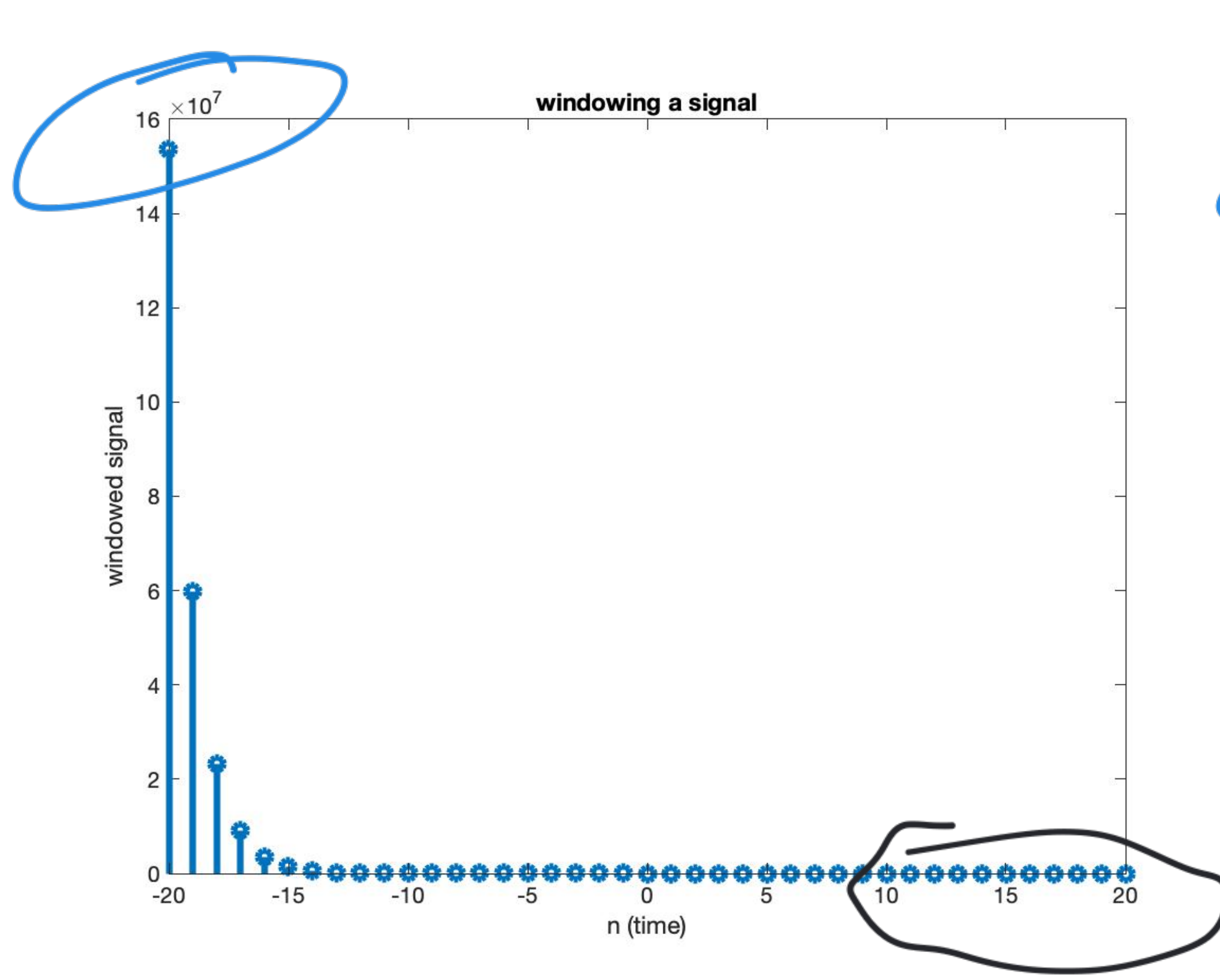
Code Example 1: a simple DT signal

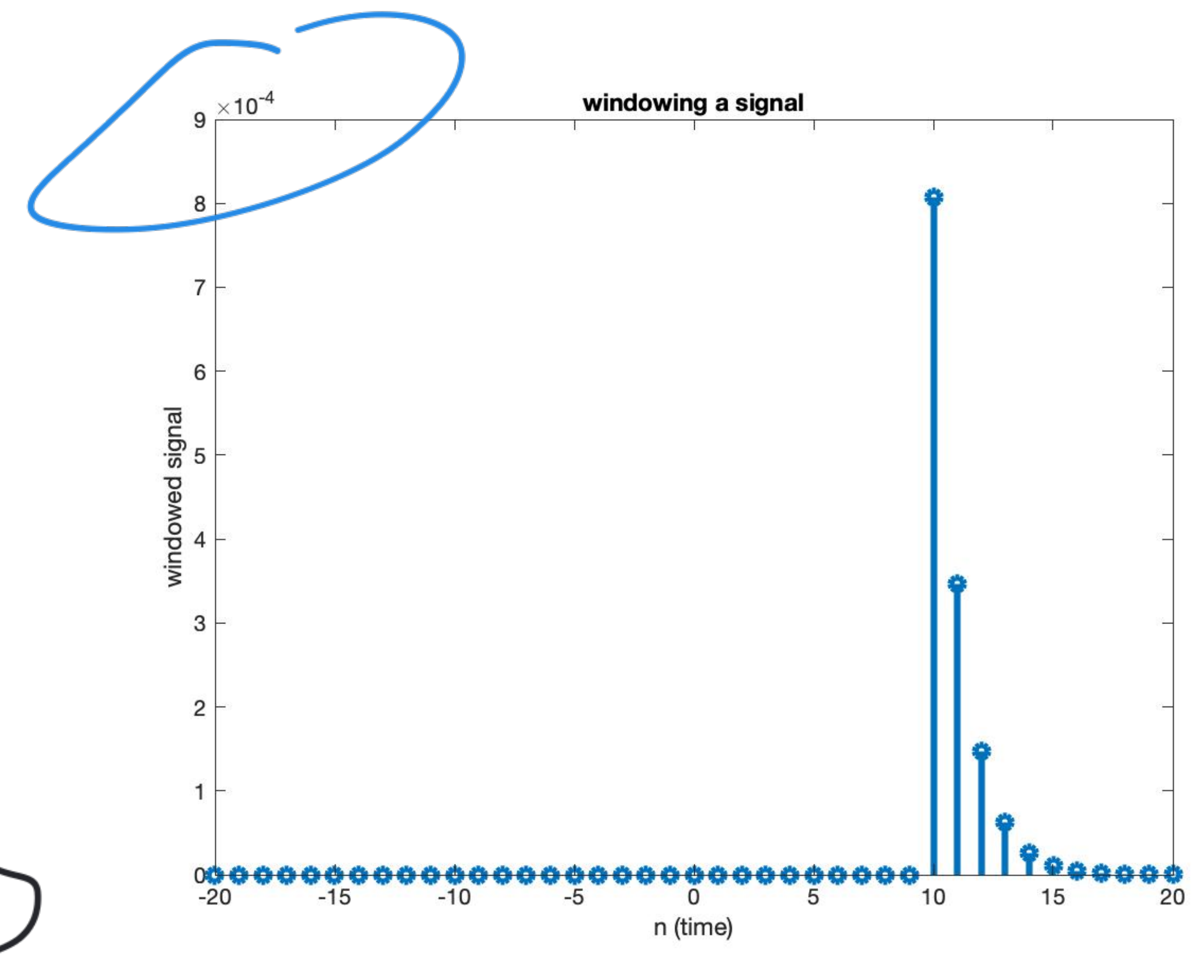
```
xsig = @(n) cos((3*pi/14)*n);
n = (-20):20;
x = xsig(n);
figure;
stem(n,x,'LineWidth',3);
xlabel('n (time)');
ylabel('signal');
title('a DT plotting example')
print('DT_plot_example.png', '-dpng');
```

The plot


a DT plotting example
signal
n (time)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-20
-15
-10
-5
0
5
10
15
20

Windowing signals





We can window signals by multiplying by the window:

Code Example 2: windowing a signal

```
xsig = @(n) exp(-0.4*n);
n = (-20):20;
window = n [redacted] .* (n >= 10);
x = xsig(n) .* window;
figure; stem(n,xsig(n),'LineWidth',3);
figure; stem(n,x,'LineWidth',3);
```

Real and imaginary parts

We can also get the real and imaginary parts of a complex signal:

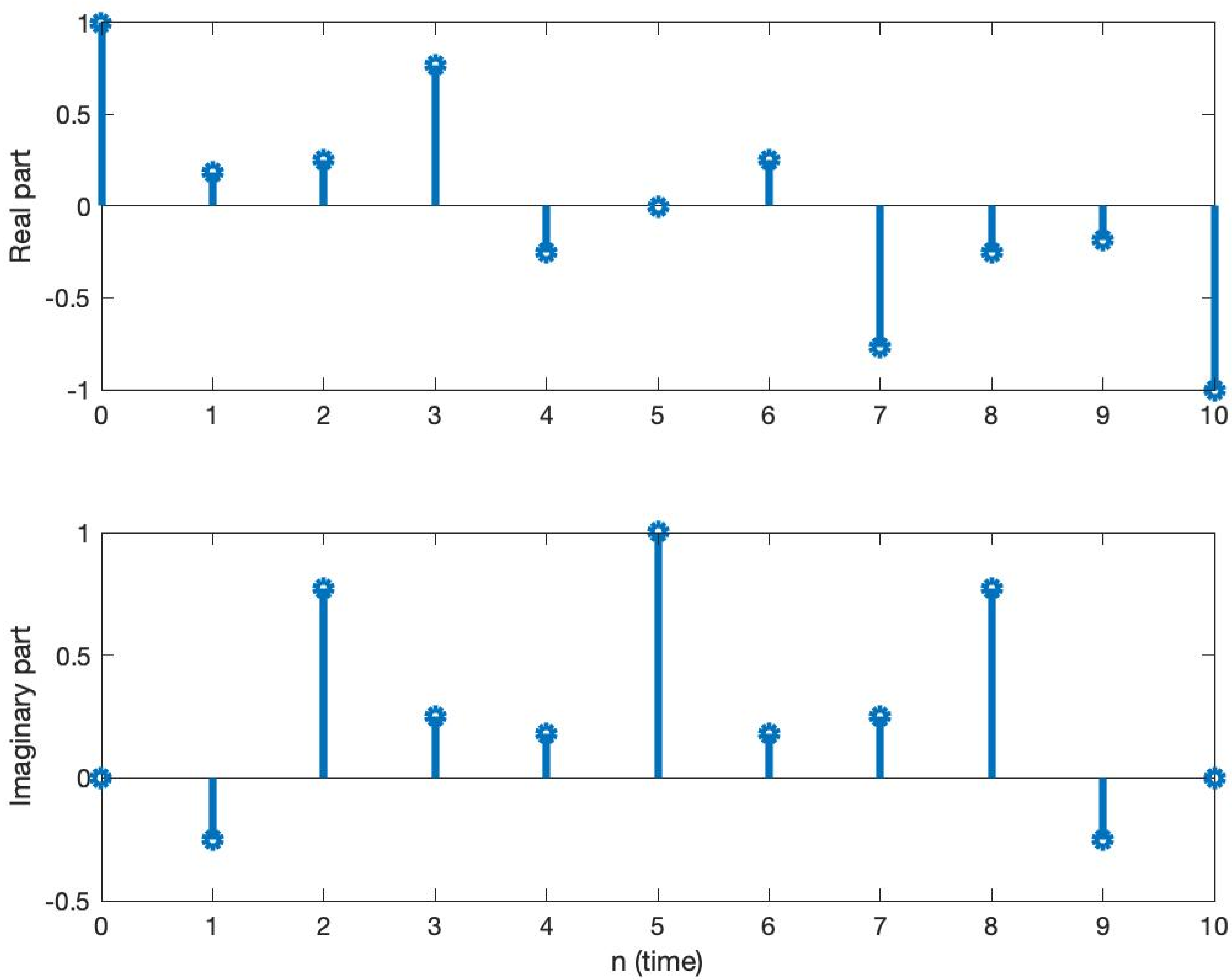
$$x(t) = e^{-j0.3\pi n} \cos(0.4\pi n) \quad (1)$$

Code Example 3: real/imaginary

```
xsig = @(n) exp(-j*0.3*pi*n) .* cos(0.4*pi*n);
n = 0:10;
figure;
subplot(2,1,1); stem(n,real(xsig(n)),'LineWidth',3);
ylabel('Real part');
subplot(2,1,2); stem(n,imag(xsig(n)),'LineWidth',3);
xlabel('n (time)');
ylabel('Imaginary part');
print('DT_reim_example.png', '-dpng');
```

1 2

The plot



1
0.5
0
-0.5
-1
Real part
0
1
2
3
4
5
6
7
8
9
10
1
0.5
0
-0.5
Imaginary part
n (time)

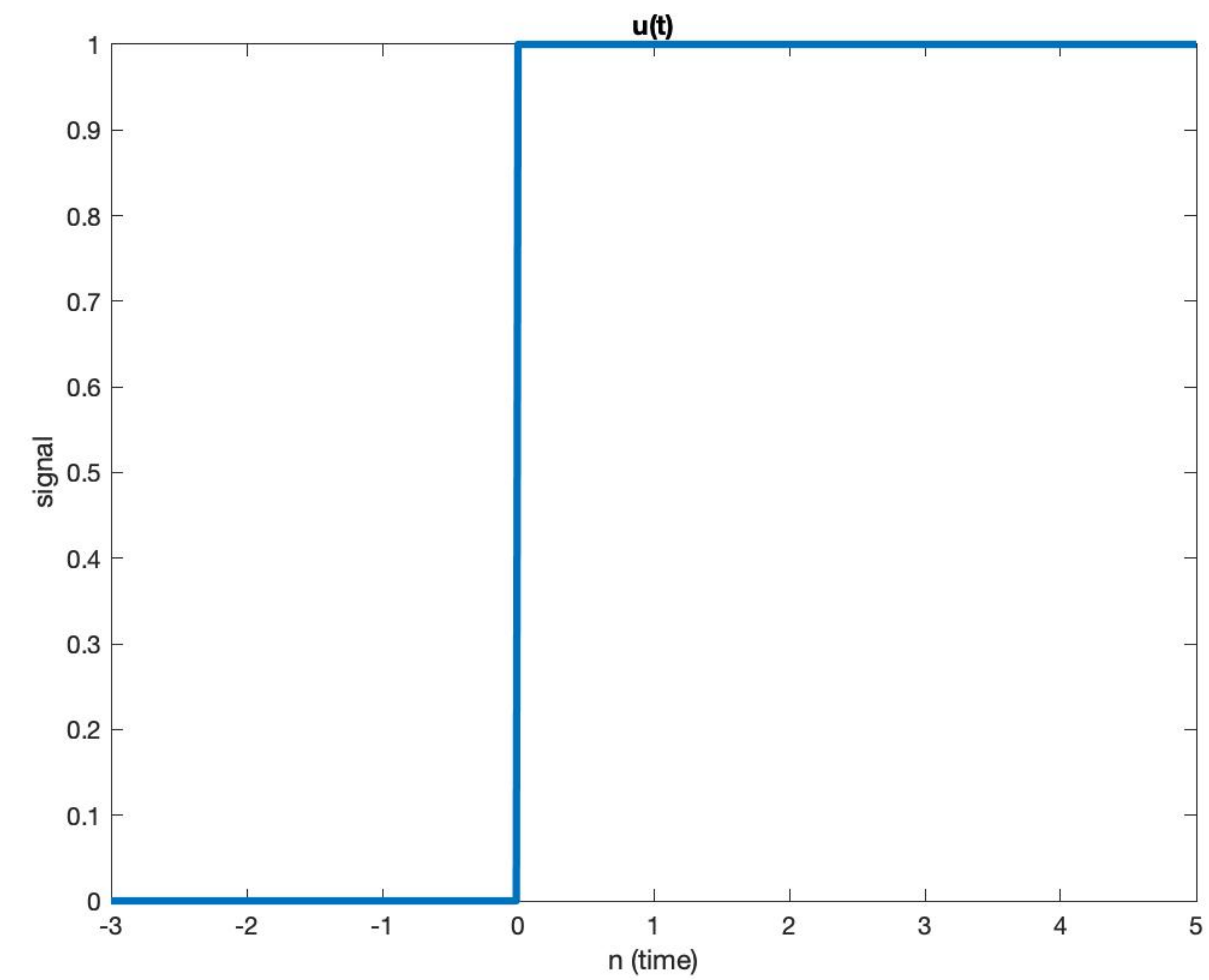
Representing a signal in CT

We have to *simulate a CT signal with a DT signal*. The idea is to use a small enough interval Δt so that

$$x[n] = x(n\Delta t) \tag{2}$$

is a good approximation of $x(t)$. Each element of $x[n]$ is a sample of $x(t)$ spaced every Δt. The sampling frequency is $f_s = \frac{1}{\Delta t}$.
How do you choose f_s? It depends on what your signal is like – if it has a lot of high-frequency elements you need f_s to be larger. We will revisit this at the end of class when we discuss the *Nyquist rate* which says how fast you have to sample a signal to be able to accurately reconstruct it.

Example: step

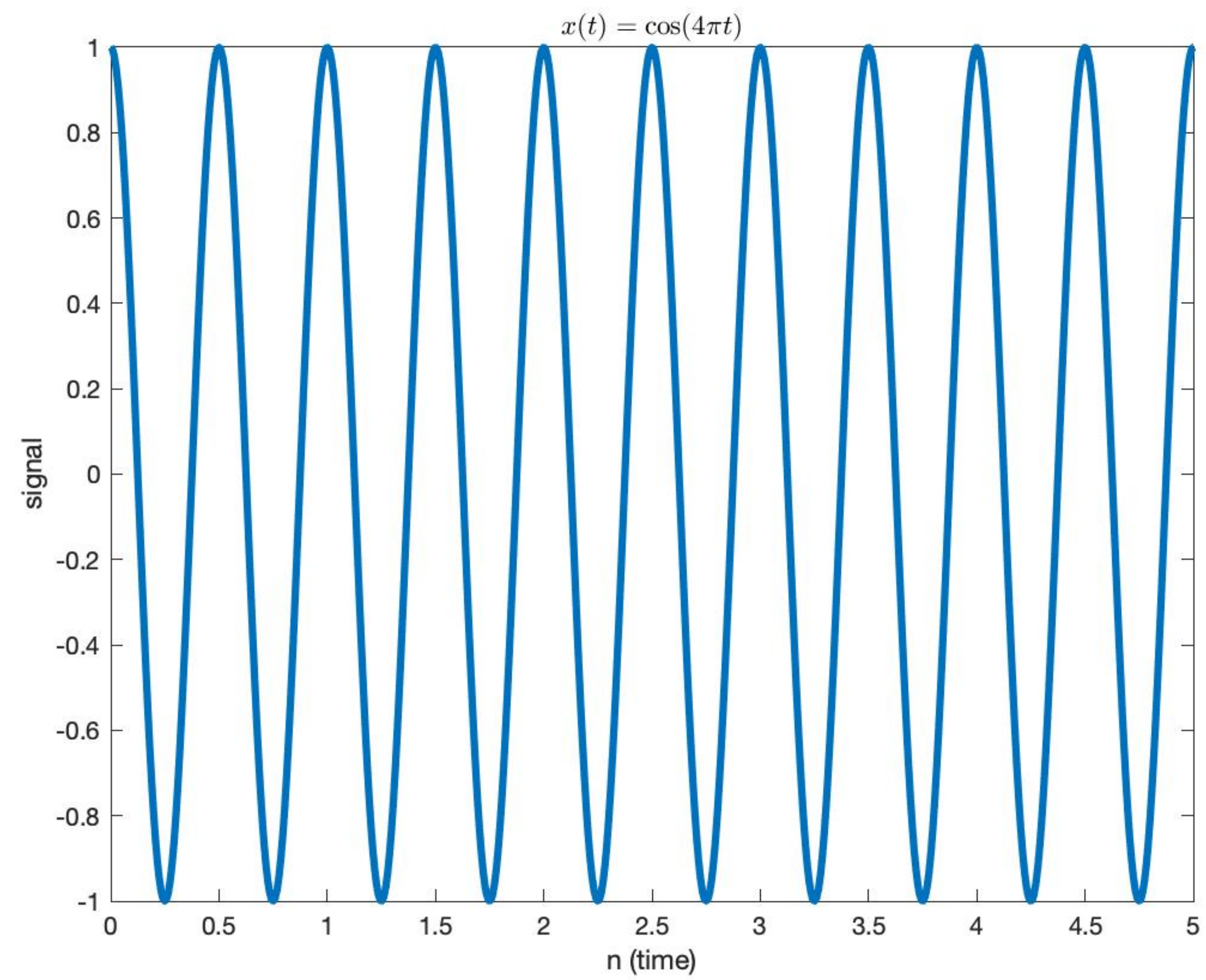


Code Example 4: step function

```
fs = 1e4;
t = (-3):(1/fs):5;
figure; plot(t,heaviside(t),'LineWidth',3);
```

u(t)

Example: a sinusoid



Code Example 5: sinusoid

```
fs = 1e6;
t = 0:(1/fs):5;
xsig = @(t) cos(4*pi*t);
figure; plot(t,xsig(t),'LineWidth',3);
```

Try it yourself

Problem

Try to simulate some of the signals we have seen so far as examples. What happens if you choose a sampling rate that is too low?