LSI systems, Fourier transforms, Sampling theory

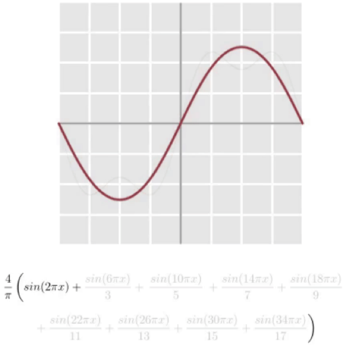
Math Tools Tutorial Oct 17, 2019

Lyndon Duong

Learning objectives:

- Linear shift-invariant systems
    - Convolution framework

- Sinusoids as inputs to LSI systems
    - Complex numbers
    - Fourier series and transforms
    - Convolution theorem



- Sampling theory
    - Aliasing
    - Nyquist theorem

Exercise 1: Testing LSI systems

labsystems.p is a function given to you that has 2 systems in it.

To pass a signal through system1, type:

```
output = labsystems(signal, 1) % 1 means sys 1, replace w 2 for sys 2
```

1) Is **System 1** linear?
2) Is it shift in variant?
3) How does it handle boundaries?

Exercise 1.1 : Testing LSI systems

How do I check for linearity in a system?

```
%% Checking for linearity
close all
clear all

n = 64;
x = 0:n-1;
in1 = randn(n,1);
in2 = randn(n,1);

out1 = labsystems(in1, 1);
out2 = labsystems(in2, 1);

out_sum = labsystems(in1+in2,1);

disp((out_sum - (out1+out2)<1e-8)')
```

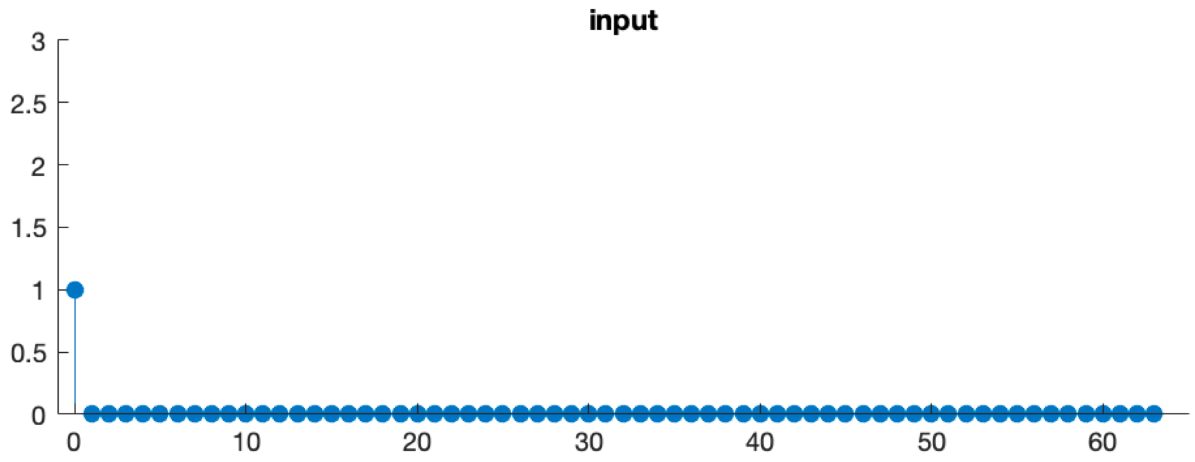
```
  Columns 1 through 30

   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1

  Columns 31 through 60

   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1

  Columns 61 through 64

   1   1   1   1
```

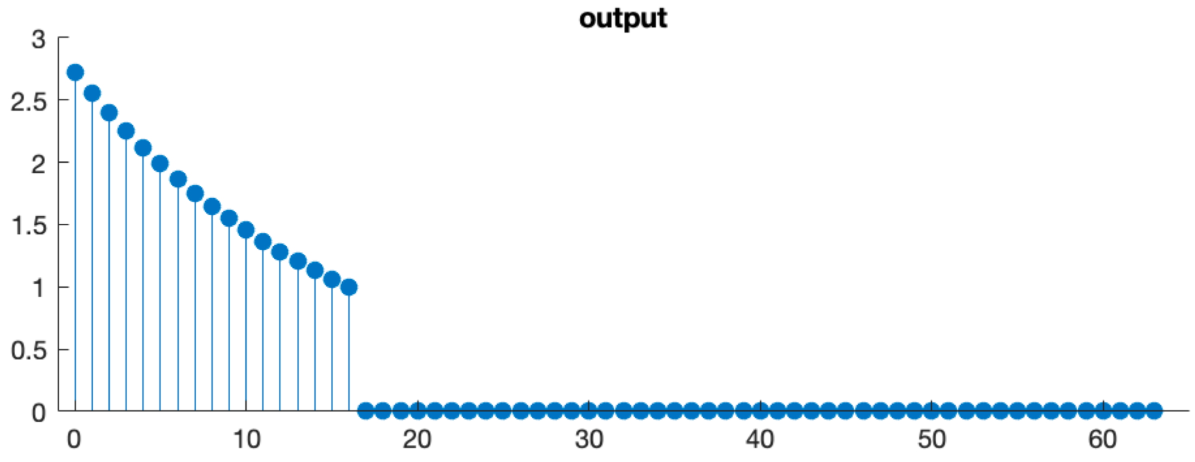
It is linear!

Exercise 1.2: Testing LSI systems

Is it shift-invariant?

How do I test for shift-invariance?





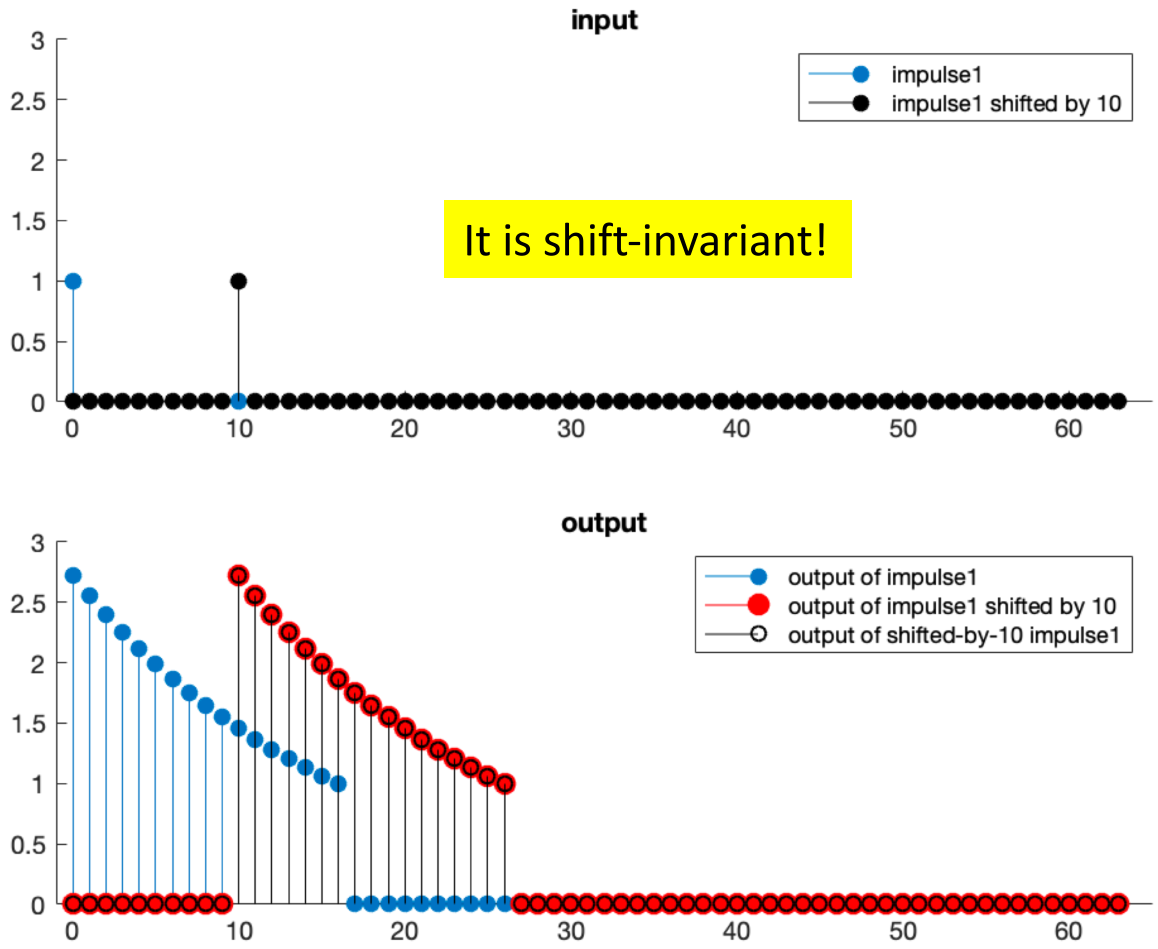
Exercise 1.3: Testing LSI systems

How do you test for shift-invariance?

```
➢ impulse = zeros(64,1);
➢ impulse(1) = 1;
➢ impulse_10 = circshift(impulse, 10)

➢ output = labsystems(impulse,1);
➢ output_10 = labsystems(impulse_10,1);
➢ disp(output_10 - circshift(output,10) < 1e-8);
```

You want to test if the shifted input's output equals the original input's output, shifted.

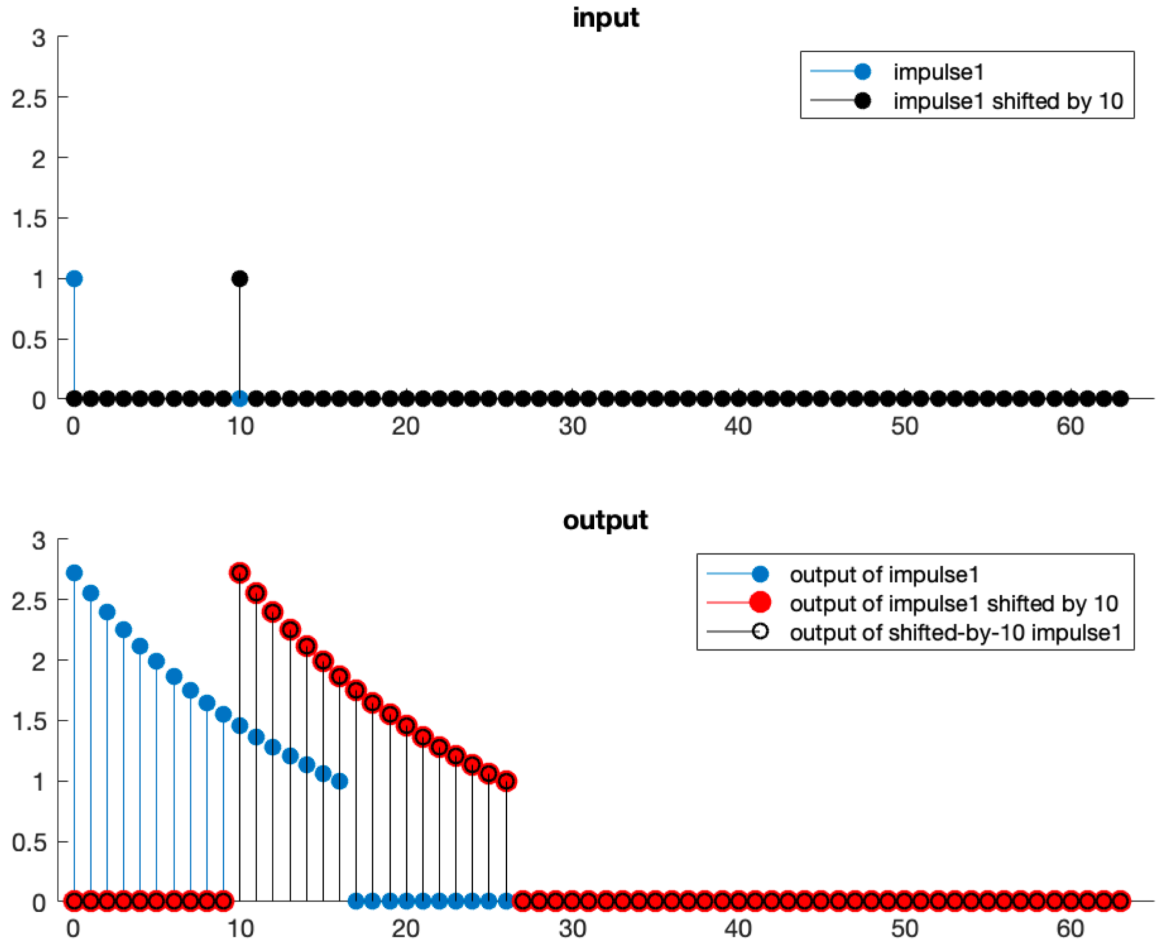
(Which is a super confusing statement but ponder on it a little bit and it will make sense)



What operation is a LSI system effectively doing?

Hint:
Each output is a ***shifted copy*** of the original impulse response.

Answer: Convolution!



How I think of LSI systems:

Input: vector of size n=9

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_5 \end{bmatrix}$$

Reverse order so x_1 enters system first

$$\vec{y}[m] = \sum_{m=1}^{n} \vec{x}\,[n-m]\vec{k}[m]$$

Output: vector of size n=9

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_5 \end{bmatrix} = \vec{y}$$

Reverse the order back

Black box LSI system

$[x_5, x_4, x_3, x_2, x_1]$

Gif source: https://www.youtube.com/watch?v=HQp6n_oJpLI

Looking inside the black box…

$$\vec{y}[m] = \sum_{m=1}^{n} \vec{x}\,[n-m]\vec{k}[m]$$

X-ray of LSI system

DOT PRODUCT overlapping components

$[k_1, k_2, k_3]$

$[x_5, x_4, x_3, x_2, x_1]$

Gif source: https://www.youtube.com/watch?v=HQp6n_oJpLI

Having our input $\vec{x}$ be oriented normally

$$\vec{y}[m] = \sum_{m=1}^{n} \vec{x}\,[n-m]\vec{k}[m] = \sum_{m=1}^{n} \vec{x}\,[m]\vec{k}[n-m]$$

Instead of flipping the input, we've now flipped the systems, i.e. 'kernel'.

X-ray of LSI system

DOT PRODUCT overlapping components

$[k_3, k_2, k_1]$

$[x_1, x_2, x_3, x_4, x_5]$

Gif source: https://www.youtube.com/watch?v=HQp6n_oJpLI

But wait…

Despite the dimensions of input and output being the same (n=5), we did **more** than n dot products (7 to be exact)

1. $[k_3, k_2, k_1]$
$[x_1, x_2, x_3, x_4, x_5]$

2. $[k_3, k_2, k_1]$
$[x_1, x_2, x_3, x_4, x_5]$

3. $[k_3, k_2, k_1]$
$[x_1, x_2, x_3, x_4, x_5]$

4. $[k_3, k_2, k_1]$
$[x_1, x_2, x_3, x_4, x_5]$

5. $[k_3, k_2, k_1]$
$[x_1, x_2, x_3, x_4, x_5]$

6. $[k_3, k_2, k_1]$
$[x_1, x_2, x_3, x_4, x_5]$

7. $[k_3, k_2, k_1]$
$[x_1, x_2, x_3, x_4, x_5]$

What do we do with the overhangs at step 1,2,6,7?
How can we compute a dot-product if there are gaps?

Solution 1: Ignore the dot products that had gaps

1. $[k_3, k_2, k_1]$
$[x_1, x_2, x_3, x_4, x_5]$

2. $[k_3, k_2, k_1]$
$[x_1, x_2, x_3, x_4, x_5]$

3. $[k_3, k_2, k_1]$
$[x_1, x_2, x_3, x_4, x_5]$

4. $[k_3, k_2, k_1]$
$[x_1, x_2, x_3, x_4, x_5]$

5. $[k_3, k_2, k_1]$
$[x_1, x_2, x_3, x_4, x_5]$

6. $[k_3, k_2, k_1]$
$[x_1, x_2, x_3, x_4, x_5]$

7. $[k_3, k_2, k_1]$
$[x_1, x_2, x_3, x_4, x_5]$

Only keep the dot products that had **complete overlap** between $\vec{k}$ and $\vec{x}$

$$\vec{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} ? \\ ? \\ x_1 k_3 + x_2 k_2 + x_3 k_1 \\ x_2 k_3 + x_3 k_2 + x_4 k_1 \\ x_3 k_3 + x_4 k_2 + x_5 k_1 \\ ? \\ ? \end{bmatrix} = \begin{bmatrix} x_1 k_3 + x_2 k_2 + x_3 k_1 \\ x_2 k_3 + x_3 k_2 + x_4 k_1 \\ x_3 k_3 + x_4 k_2 + x_5 k_1 \end{bmatrix}$$

```
% Matlab
xx = [1 1 1 1 1];
kk = [1 1 1];
conv(xx,kk,'valid')
```

```
ans = 1×3
     3     3     3
```

Solution 2: Zero-padding

1. $[k_3, k_2, k_1]$
   $[0, 0, x_1, x_2, x_3, x_4, x_5, 0, 0]$
2. $[k_3, k_2, k_1]$
   $[0, 0, x_1, x_2, x_3, x_4, x_5, 0, 0]$
3. $[k_3, k_2, k_1]$
   $[0, 0, x_1, x_2, x_3, x_4, x_5, 0, 0]$
4. $[k_3, k_2, k_1]$
   $[0, 0, x_1, x_2, x_3, x_4, x_5, 0, 0]$
5. $[k_3, k_2, k_1]$
   $[0, 0, x_1, x_2, x_3, x_4, x_5, 0, 0]$
6. $[k_3, k_2, k_1]$
   $[0, 0, x_1, x_2, x_3, x_4, x_5, 0, 0]$
7. $[k_3, k_2, k_1]$
   $[0, 0, x_1, x_2, x_3, x_4, x_5, 0, 0]$

$$\vec{y} = \begin{bmatrix} 0k_3 + 0k_2 + x_1k_1 \\ 0k_3 + x_1k_2 + x_2k_1 \\ x_1k_3 + x_2k_2 + x_3k_1 \\ x_2k_3 + x_3k_2 + x_4k_1 \\ x_3k_3 + x_4k_2 + x_5k_1 \\ x_4k_3 + x_5k_2 + 0k_1 \\ x_5k_3 + 0k_2 + 0k_1 \end{bmatrix} = \begin{bmatrix} x_1k_1 \\ x_1k_2 + x_2k_1 \\ x_1k_3 + x_2k_2 + x_3k_1 \\ x_2k_3 + x_3k_2 + x_4k_1 \\ x_3k_3 + x_4k_2 + x_5k_1 \\ x_4k_3 + x_5k_2 \\ x_5k_3 \end{bmatrix}$$

```
% Matlab
xx = [1 1 1 1 1];
kk = [1 1 1];
conv(xx,kk,'full') % default is 'full'
```

```
ans = 1×7
     1     2     3     3     3     2     1
```

```
 1 1 1];
kk = [1 1 1];
conv(xx,kk,'same')
```

```
ans = 1×5
     2     3     3     3     2
```

Solution 3: assume your input signal is **periodic**

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}$$

1. $[k_3, k_2, k_1]$
$[x_1, x_2, x_3, x_4, x_5, x_1, x_2, x_3, x_4, x_5, x_1$

2. $[k_3, k_2, k_1]$
$[x_1, x_2, x_3, x_4, x_5, x_1, x_2, x_3, x_4, x_5, x_1, x_2$

⋮

6. $[k_3, k_2, k_1]$
$[x_1, x_2, x_3, x_4, x_5, x_1, x_2, x_3, x_4, x_5, x_1, x_2, x_3, x_4, x_5]$

7. $[k_3, k_2, k_1]$
$[x_1, x_2, x_3, x_4, x_5, x_1, x_2, x_3, x_4, x_5, x_1, x_2, x_3, x_4, x_5]$

8. $[k_3, k_2, k_1]$
$[x_1, x_2, x_3, x_4, x_5, x_1, x_2, x_3, x_4, x_5, x_1, x_2, x_3, x_4, x_5]$

9. $[k_3, k_2, k_1]$
$[x_1, x_2, x_3, x_4, x_5, x_1, x_2, x_3, x_4, x_5, x_1, x_2, x_3, x_4, x_5]$

10. $[k_3, k_2, k_1]$
$[x_1, x_2, x_3, x_4, x_5, x_1, x_2, x_3, x_4, x_5, x_1, x_2, x_3, x_4, x_5]$

11. $[k_3, k_2, k_1]$
$[x_1, x_2, x_3, x_4, x_5, x_1, x_2, x_3, x_4, x_5, x_1, x_2, x_3, x_4, x_5]$

12. $[k_3, k_2, k_1]$
$[x_1, x_2, x_3, x_4, x_5, x_1, x_2, x_3, x_4, x_5, x_1, x_2, x_3, x_4, x_5]$

⋮

Periodic boundary handling, cont'd

Our input is n=5, and we want our output to be n=5

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} \qquad \vec{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_6 \\ y_7 \\ y_8 \\ y_9 \\ y_{10} \\ \vdots \\ y_{17} \end{bmatrix} \longrightarrow$$

With zero-padding $\vec{y} = \begin{bmatrix} 0k_3 + 0k_2 + x_1k_1 \\ 0k_3 + x_1k_2 + x_2k_1 \\ x_1k_3 + x_2k_2 + x_3k_1 \\ x_2k_3 + x_3k_2 + x_4k_1 \\ x_3k_3 + x_4k_2 + x_5k_1 \\ x_4k_3 + x_5k_2 + 0k_1 \\ x_5k_3 + 0k_2 + 0k_1 \end{bmatrix} = \begin{bmatrix} x_1k_1 \\ x_1k_2 + x_2k_1 \\ x_1k_3 + x_2k_2 + x_3k_1 \\ x_2k_3 + x_3k_2 + x_4k_1 \\ x_3k_3 + x_4k_2 + x_5k_1 \\ x_4k_3 + x_5k_2 \\ x_5k_3 \end{bmatrix}$

With periodic boundary padding $\vec{y} = \begin{bmatrix} x_4k_3 + x_5k_2 + x_1k_1 \\ x_5k_3 + x_1k_2 + x_2k_1 \\ x_1k_3 + x_2k_2 + x_3k_1 \\ x_2k_3 + x_3k_2 + x_4k_1 \\ x_3k_3 + x_4k_2 + x_5k_1 \\ x_4k_3 + x_5k_2 + x_1k_1 \\ x_5k_3 + x_1k_2 + x_2k_1 \end{bmatrix} = \begin{bmatrix} x_4k_3 + x_5k_2 + x_1k_1 \\ x_5k_3 + x_1k_2 + x_2k_1 \\ x_1k_3 + x_2k_2 + x_3k_1 \\ x_2k_3 + x_3k_2 + x_4k_1 \\ x_3k_3 + x_4k_2 + x_5k_1 \end{bmatrix}$

Exercise 1.4: Testing LSI systems

1) Is **System 1** linear?

Yes!

2) Is it shift in variant?

Yes!

3) How does it handle boundaries?

<u>Let’s see (white board)!</u>

Solution:

- System 1 is linear, shift invariant, and has periodic boundary handling

1) Is **System 1** linear?

   Yes.

2) Is it shift in variant?

   Yes.

3) How does it handle boundaries?

   Periodically.

Summary: LSI Systems pt. 1

- Saying a system is LSI is a *stronger* statement than just saying it is linear
  - Measure one input/output pair (the impulse response), and you are **done**

- The action of an LSI system on a signal is *equivalent* to the convolution operation
  - Convolution can be written as matrix multiplication (remember to flip the kernel)

- Convolution (and thus LSI systems too), must ”deal with” signal boundaries
  - E.g. zero-padding, periodic/circular boundaries

LSI systems part 2: Sines and cosines

Definition

Sinusoids:

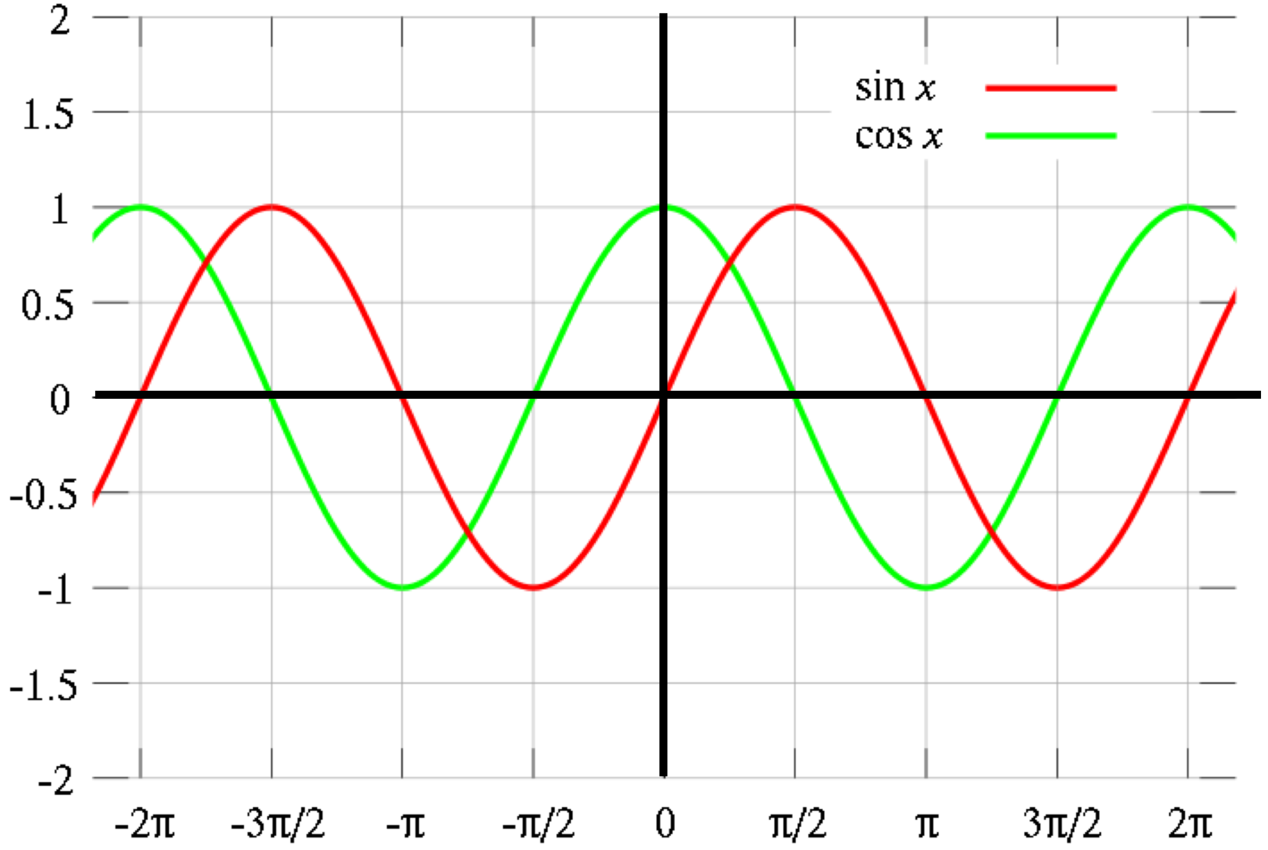
$\cos(\omega x), \sin(\omega x)$ or equivalently, $\cos(2\pi f x), \sin(2\pi f x)$

ω: omega, angular frequency in radians

$2\pi f$: where f is regular frequency (e.g. Hz); can alternatively use k instead of f

$\omega = 2\pi f = 2\pi k/n$

Recall sines and cosines



Source: wiki

Exercise: Sinusoids are sinusoids

Plot 2 sinusoids:

1) What is the frequency?

2) What is the period?

3)What is the phase shift of sig3?

4) How does sig3 relate to sig1+sig2?

```
k = 5;
n = 64; % total number of points
x = 0:(n-1); % domain on which to plot

sig1 = 3*cos(2*pi*k/n * x);
sig2 = 4*sin(2*pi*k/n * x);

sig3 = 5*cos(2*pi*k/n * x - atan(4/3));

close all
hold on
plot(x, sig3,'r','linewidth',2)
plot(x, sig1+sig2 ,'--k','linewidth',2)
```

Solution:

1) What is the frequency?

They each have a frequency k/n = 5/64

2) What is the period?

Period is 1/f = 64/5, i.e. it takes 64/5 points until the signal repeats

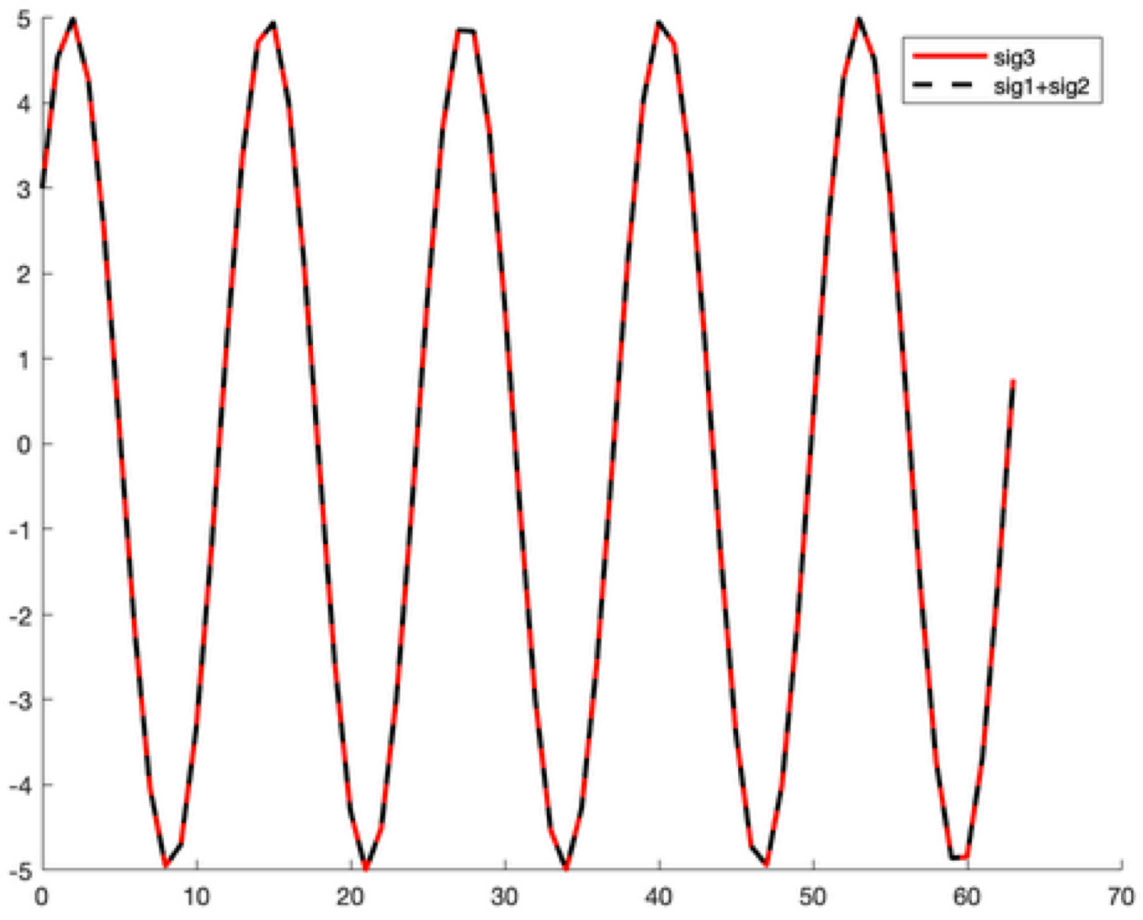
3) What is the phase shift of sig3?

Sig3's phase shift is atan(4/3) which is 0.9273 rads

4) How does sig3 relate to sig1+sig2?

sig3=sig1+sig2

$$5\cos\left(2\pi\frac{k}{n}\mathrm{x}-0.9273\right)=3\cos\left(2\pi\frac{k}{n}\mathrm{x}\right)+4\sin(2\pi\frac{k}{n}\mathrm{x})$$



Any phase-shifted sinusoid can instead be described as a combo of a cos() plus a sin() of the same frequency ***without*** *a phase shift*

Fourier series, informally

ANY function can be represented as:
a sum of sinusoids of different frequencies

The Fourier transform -- intuition

- Project a signal onto orthogonal basis (i.e. a *rotation of the space)*
- Each axis represents a different (cos+sin) pair of a different frequency

Exercise 3.1: Sinusoids as building blocks

```
n=1000;
x = (0:n-1)';

hold on
y = sin(2*pi*1/n * x);
plot(y)

y = y+ 1/3*sin(2*pi*3/n * x);
plot(y)

y= y+ 1/5*sin(2*pi*5/n * x);
plot(y)

y= y+ 1/7*sin(2*pi*7/n * x);
plot(y)
```

$$y = \sin\left(2\pi\frac{1}{n}x\right)$$

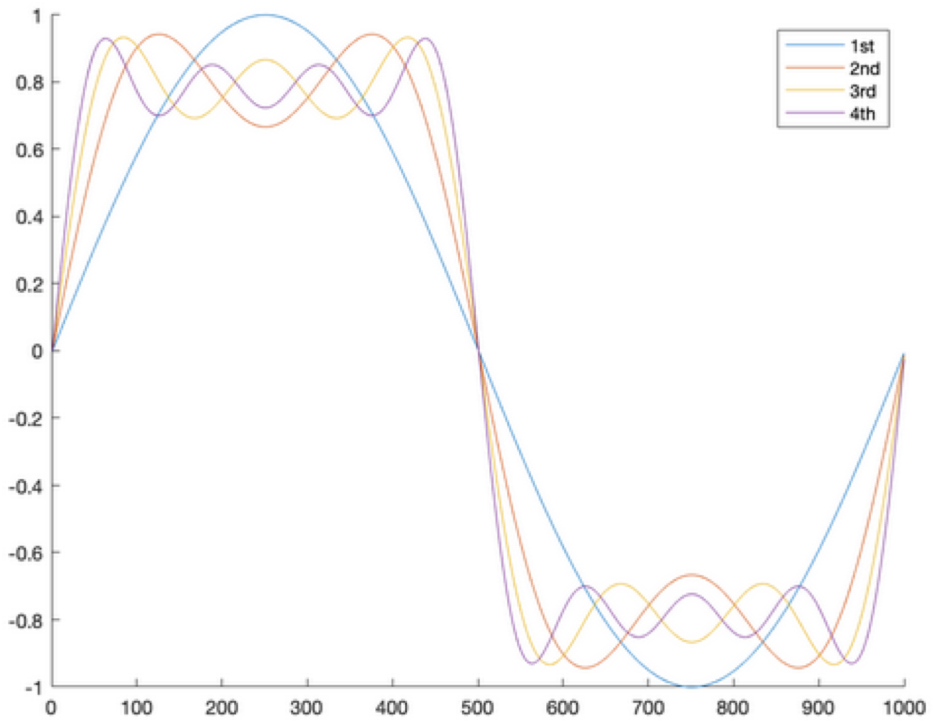
$$+ \frac{1}{3}\sin\left(2\pi\frac{3}{n}x\right)$$

$$+ \frac{1}{5}\sin\left(2\pi\frac{5}{n}x\right)$$

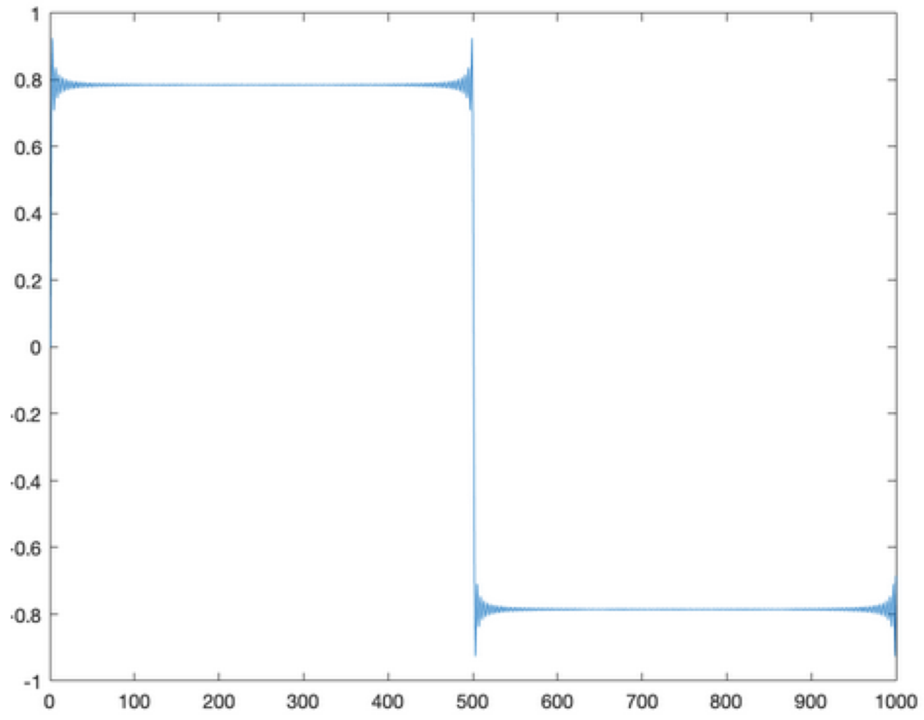
$$+ \frac{1}{7}\sin\left(2\pi\frac{7}{n}x\right)$$

Can you see the pattern?

Solution 3.1: Sinusoids as building blocks



After adding the 125th sine wave…



$y = \sum_{k=1,3,5,\ldots}^{\infty} \frac{1}{k} \sin\left(2\pi \frac{k}{n} x\right)$; This forms a ‘square wave’

You can synthesize (almost) **any** function/signal from a bunch of sinusoids of different frequencies.

Summary: Sinusoids

Any signal can be **synthesized** from many cosines and sines at different frequencies

Conversely…

Any signal can be **decomposed** into many cosines and sines at different frequencies

↑ This is what the Fourier transform does.

Imaginary numbers and sinusoids are intimately related

Introduce Euler's formula:

$$z = e^{i\omega x} = \cos(\omega x) + i\, sin(\omega x)$$

e: Euler's number
i: imaginary const.
ω: angular frequency

Each frequency ω can be described by one cosine and one sine.

If we pair them together then that looks like:

‘Complex math’ does **not** have to mean ‘difficult math’!

Introduction:

The complex, ‘imaginary’ number i (AKA less commonly but still frequent, j)

[In the study of electrical circuits, which necessitates complex math, i means current (e.g. v=ir) so they use the letter j instead]

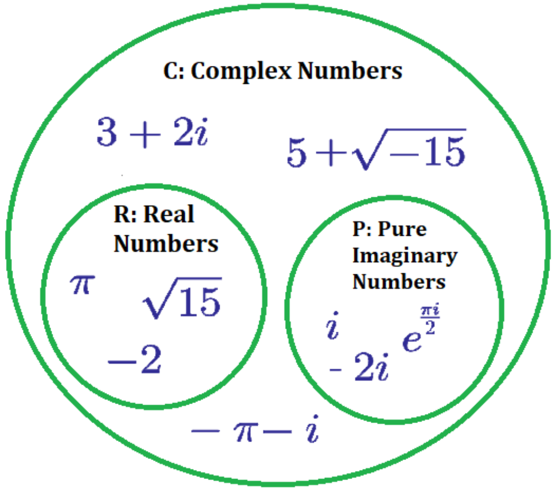
$$\sqrt{-1} = i = j$$

Command Window

```
Matlab defaults the imaginary number to i and j.
i=
   0.0000 + 1.0000i

j=
   0.0000 + 1.0000i

sqrt(-1)=
   0.0000 + 1.0000i
```

This is one reason why you shouldn’t name your for-loop variables *i* or *j*, and should use *ii* or *jj* instead

The rules of complex algebra



$$c = a + bi$$

Real part Imag. part

c is just ***one*** number comprising 2 parts

$$\mathrm{real}(c) = a$$
$$\mathrm{imag}(c) = b$$

In MATLAB:

Command Window

```
c =
   5.0000 + 8.0000i


ans =

    'real(c) = 5.000000'


ans =

    'imag(c) = 8.000000'
```

How i think of complex numbers:

A notational convenience in order to pack 2 numbers into 1

The rules of complex algebra

$$x = a + bi$$
$$y = c + di$$
$$x + y = (a + c) + (b + d)i$$

The sum is **still one** number with a real and imaginary part

$$\begin{aligned} x * y &= (a + bi) * (c + di) \\ &= ac + adi + bci + bdi^2 \\ &= ac + adi + bci + bd(-1) \\ &= ac + adi + bci - bd \\ &= (ac - bd) + (ad + bc)i \end{aligned}$$

$i = \sqrt{-1}$, so $i^2 = (\sqrt{-1})^2 = -1$

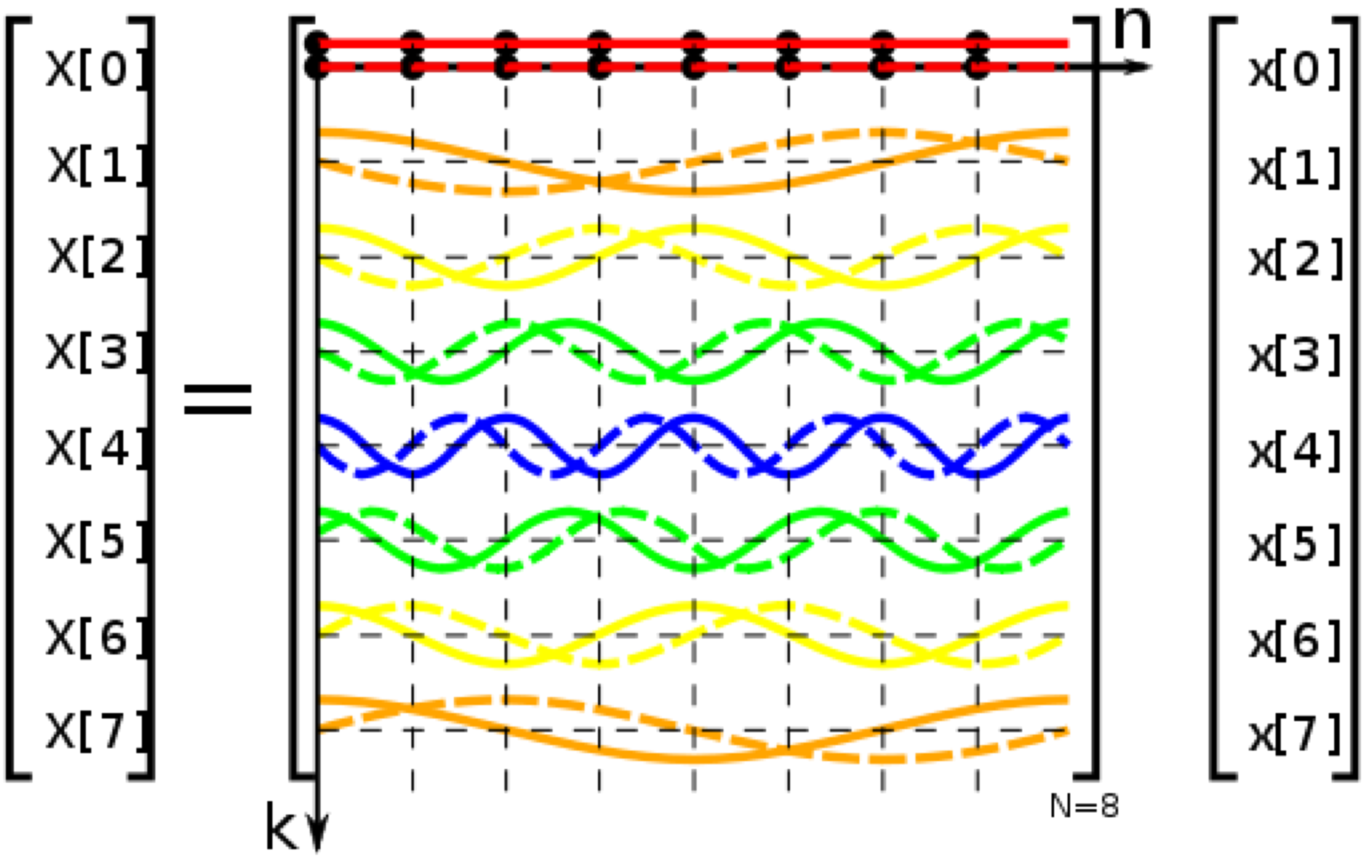
Real part

Imag. part

The product is again **one** number with a real and imaginary part

The (discrete) Fourier transform (DFT)

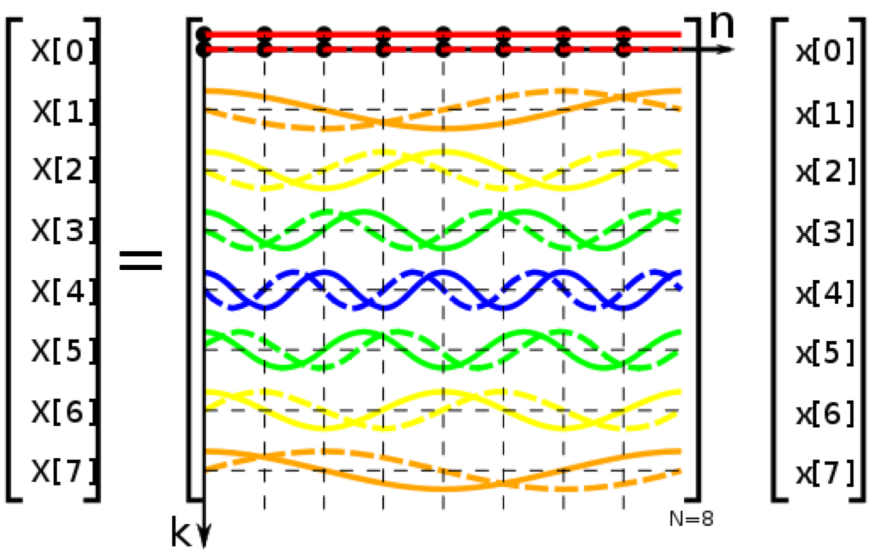
Capital letters usually denote Fourier transformed signal



Lowercase letters denote original signal

$X = F(x) = Dx$; where D is the DFT matrix

The DFT is an orthogonal matrix!



$$z = e^{i\omega x} = \cos(\omega x) + i\, sin(\omega x)$$

```
>> dftmtx(8)

ans =

   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i
   1.0000 + 0.0000i   0.7071 - 0.7071i   0.0000 - 1.0000i  -0.7071 - 0.7071i  -1.0000 + 0.0000i  -0.7071 + 0.7071i   0.0000 + 1.0000i   0.7071 + 0.7071i
   1.0000 + 0.0000i   0.0000 - 1.0000i  -1.0000 + 0.0000i   0.0000 + 1.0000i   1.0000 + 0.0000i   0.0000 - 1.0000i  -1.0000 + 0.0000i   0.0000 + 1.0000i
   1.0000 + 0.0000i  -0.7071 - 0.7071i   0.0000 + 1.0000i   0.7071 - 0.7071i  -1.0000 + 0.0000i   0.7071 + 0.7071i   0.0000 - 1.0000i  -0.7071 + 0.7071i
   1.0000 + 0.0000i  -1.0000 + 0.0000i   1.0000 + 0.0000i  -1.0000 + 0.0000i   1.0000 + 0.0000i  -1.0000 + 0.0000i   1.0000 + 0.0000i  -1.0000 + 0.0000i
   1.0000 + 0.0000i  -0.7071 + 0.7071i   0.0000 - 1.0000i   0.7071 + 0.7071i  -1.0000 + 0.0000i   0.7071 - 0.7071i   0.0000 + 1.0000i  -0.7071 - 0.7071i
   1.0000 + 0.0000i   0.0000 + 1.0000i  -1.0000 + 0.0000i   0.0000 - 1.0000i   1.0000 + 0.0000i   0.0000 + 1.0000i  -1.0000 + 0.0000i   0.0000 - 1.0000i
   1.0000 + 0.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i  -0.7071 + 0.7071i  -1.0000 + 0.0000i  -0.7071 - 0.7071i   0.0000 - 1.0000i   0.7071 - 0.7071i
```

Exercise 3.2: Decomposing a signal using DFT

Using the same signal that we just created in 3.1.

Compute its DFT and plot its amplitude spectrum

The inverse (discrete) Fourier transform (IDFT)

If the DFT is an orthogonal matrix, what is the inverse DFT?

The transposed complex conjugate of the DFT matrix!

Power and amplitude spectrum

- Amplitude spectrum: $|F(x)|$
- Power spectrum: $|F(x)|^2$

Signal vs. Fourier domain

Signal and Fourier representations are two sides of the same coin

Exercise:

- Power spectrum of a signal y: $|F(y)|^2$

Predict these signals' power spectra:

- 1) $y = \text{zeros}(64,1)\,;$
- 2) $y = \sin\left(2\pi \frac{5}{64}\ (0{:}\,63)\right);$
- 3) $y = 5 * \text{ones}(64,1)\,;$
- 4) $y = \text{randn}(64,1);$
- 5) $y = \sin\left(2\pi \frac{5}{64}\ (0{:}\,63)\right) + \cos\left(2\pi \frac{11}{64}\ (0{:}\,63)\right)\ ;$

Solution:

Predict these signals' power spectra:

- 1) $y = \mathrm{zeros}(64,1)\,;$
- 2) $y = \sin\left(2\pi \frac{5}{64}\ (0{:}\,63)\right);$
- 3) $y = 5 * \mathrm{ones}(64,1)\,;$
- 4) $y = \mathrm{randn}(64,1);$
- 5) $y = \sin\left(2\pi \frac{5}{64}\ (0{:}\,63)\right) + \cos\left(2\pi \frac{11}{64}\ (0{:}\,63)\right)\ ;$

Summary 2: Fourier transforms

- How does this relate to Linear shift invariant systems?

Sinusoids are ‘eigenfunctions’ of LSI systems!
You put in an input of cos+sin of a given frequency, and the output is a cos+sin of that same frequency but each with possibly different amplitudes.

Recommended short videos/articles:

- Convolution:
  - [Convolution as matrix multiplication]
- Fourier stuff:
  - [Sines and cosines from vectors (Khan Academy, 3mins)]
  - [Complex numbers and vectors (Khan Academy, 8mins)]
- Nyquist sampling:
  - [Nyquist frequency]
  - [Aliasing]