# MASTER MATLAB THROUGH GUIDED PROBLEM SOLVING

Udemy Hosted Course: https://www.udemy.com/course/master-matlab-through-guided-problem-solving/

Instructor: Mike X Cohen: http://sincxpress.com/

Notes and code completion (student): Ron Fredericks: http://BiophysicsLab.com/

**Course-work: Feb 10 to May 16, 2020** **Notes last updated: Wednesday, May 20, 2020**

## MATLAB TOOLBOXES USED IN THIS COURSE

| | |
|---|---|
| MATLAB R2020a | https://www.mathworks.com/help/matlab/ |
| Signal Processing Toolbox v:8.4 | https://www.mathworks.com/help/signal/ |
| Statistics and Machine Learning Toolbox v:11.7 | https://www.mathworks.com/help/stats/ |
| Image Processing Toolbox v:11.1 | https://www.mathworks.com/help/images/ |
| Symbolic Math Toolbox v:8.5 | https://www.mathworks.com/help/symbolic/ |
| Optimization Toolbox v:8.5 | https://www.mathworks.com/help/optim/ |

GUIDE to App Designer Migration Tool for MATLAB v:2.0
https://www.mathworks.com/matlabcentral/fileexchange/66087-guide-to-app-designer-migration-tool-for-matlab

## TABLE OF CONTENTS

## SECTION 1: COURSE INTRODUCTION

## SECTION 2: GETTING STARTED

### 5. CREATE, EDIT, AND OPEN SCRIPTS

Command window controls:

is: Display files in current directory

what: like is but without subdirectories

<tab>: will try to expand a partially entered file name in all known paths, such as when using the edit command

edit: open a script in the editor

==: Comparison

=: Assignment

<ctrl>s: save script

Type script name to run the script

## 6. WRITE COMMENTS IN LINES AND BLOCKS

%: comment

%%: start a new block or cell of code, may include header comment

Create a block of code:

```
%{
Comments go here for as many lines as needed
%}
```

…: Continue to next line

Keyboard short cuts:

<ctrl>c: copy

<ctrl>v: paste

<ctrl>z: undo

<ctrl>a: select all

<ctrl>s: save

<ctrl>t: uncomment

<ctrl>r: comment lines of code

<ctrl><enter>: run results of current cell of code

## 7. USING MATLAB FOR A PERSONAL BUDGET

### MATLAB

Code: MasterMATLAB_0120_personalBudget.m

```
>> MasterMATLAB_0120_personalBudget
I can spend 5 extra each day.
I can spend 23.3333 during the weekend, and 2.3333 each day.
```

## 9. START MATLAB WITH AN ENCOURAGING NOTE

Create startup.m using notepad++ (if startup.m is not found)

Original default color map:

```
colormap parula
```

Test colormap:

```
figure
imagesc(randn(20))
```

MATLAB

Code: startup.m

Default Location for startup: D:\Ron\Documents\MATLAB\startup.m

```
set(0, 'DefaultFigureWindowStyle', 'docked')
set(0, 'DefaultFigureColormap', jet), close all
```

## SECTION 3: VECTORS AND VARIABLES

D:\Ron\Documents\Ron Fredericks Consulting\Education\Master MATLAB Problem Solving - Udemy\Section 03\

### 10. CREATE VECTORS AND MATRICES

Use semicolons, square brackets to create column vectors, row vectors, and matrices.

Use functions *ones, zeroes,* and *randn* to make vectors and matrices

Skills: transpose (‘), transpose function, ones, zeroes, randn []

MATLAB

Code: MasterMATLAB_0160_createVectors.m

### 12. WORKING WITH TEXT (CHARACTERS AND STRINGS)

Parse text according to spaces and remove 4-letter words.

Identify text patterns and replace with other patterns of different length.

Skills: regexp, cellfun, strfind, concatenate

==: Boolean result produces a 1 for true

~=: Boolean result produces a 1 for false

MATLAB

Code: MasterMATLAB_0180_text.m

## 13. HTML TABLE FROM MATLAB CODE

Generate a table of letters and numbers, and print them into an html table that can be read in a web browser.

Skills: char, randn, randi, disp ,for, fprintf, and num2str

MATLAB

Code: MasterMATLAB_0200_html.m

## 14. ROUND PI TO N SIGNIFICANT DIGITS

Skills: num2str, round, format, pi

MATLAB

Code: RonMATLAB_2020_02_Round_pi_N_Signif_Digs.m

## 15. FILE/FOLDER INFORMATION USING STRUCTURES

Get nonselective and selective folder queries to identify subfolders and find filenames containing pattern "variable."

Extract the length of a filenames (number of characters).

Skills: dir, cellfun

MATLAB

Code: MasterMATLAB_0220_fileInfoFromStructures.m

## SECTION 4: COMMAND STATEMENTS

Folder: D:\Ron\Documents\Ron Fredericks Consulting\Education\Master MATLAB Problem Solving - Udemy\Section 04

### 16. CREATE A HILBERT MATRIX USING FOR-LOOPS

Use for-loops to create an MxM Hilbert matrix.

In the same for-loop, create a checkerboard matrix.

Skills: zeros, for, imagesc

Plotting skills: figure, clf, subplot, axis square, title('Checkerboard matrix'), set(gca,'xtick',[],'ytick',[])

Hilbert matrix:

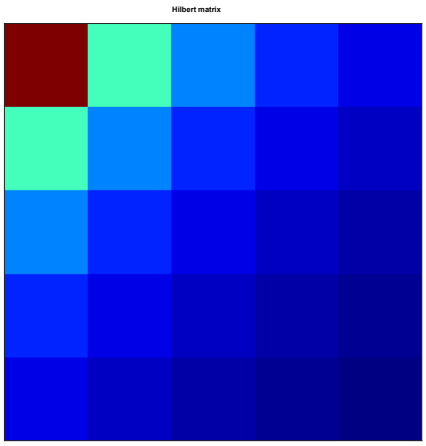
$$M_{i,j} = \frac{1}{i + j - 1}$$

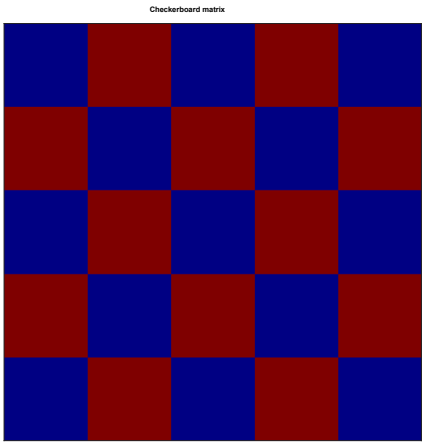
Checkerboard matrix:

$$M_{i,j} = -1^{i+j-1}$$

MATLAB

Code: MasterMATLAB_0240_HilbertMatrix.m




## 18. CREATE AN UPPER-TRIANGULAR MATRIX

Use for=loops and if-statements to fill the upper-triangular elements of an MxM matrix.

Populate the lower triangle of the matrix to be the flipped version of the upper-triangle.

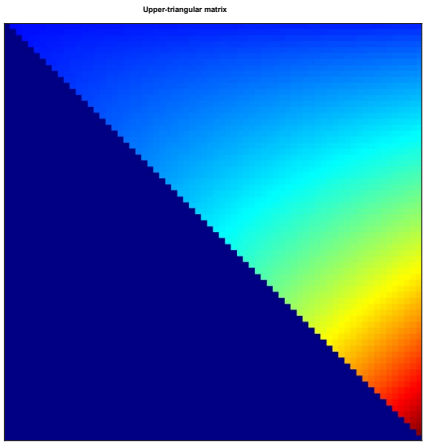
Skills: for, zeros, if

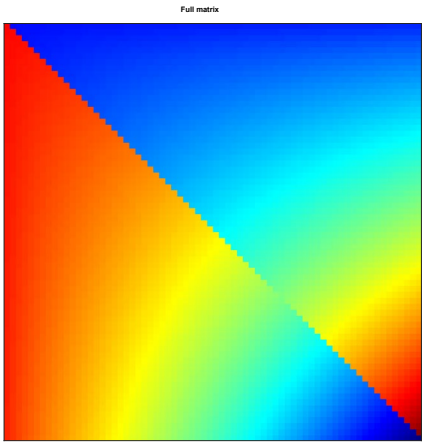
## Upper triangle matrix:

$$M_{i,j} = 1.03^{\sqrt{ij}}$$

### MATLAB

Code: MasterMATLAB_0260_TriangularMatrix.m




## 19. RANDOM COUNT-DOWN TIMER (POISSON-LIKE)

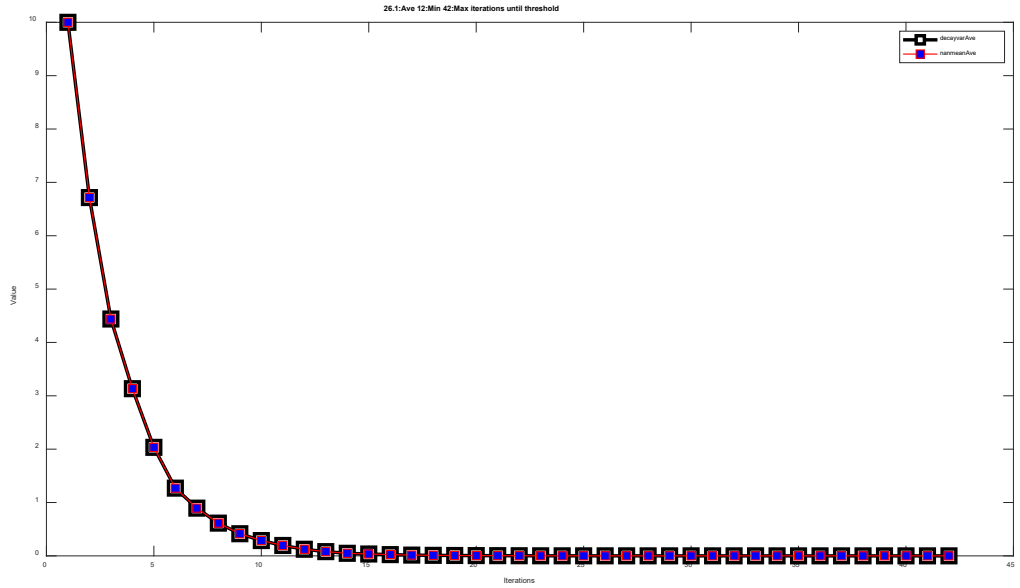
A number iteratively decreases by being multiplied by a random number between 0 and 1. A counter tracks the number of iterations until a set threshold is reached.

Modulate the rate of decrease using fractional exponents. Repeat 100 iterations and plot the average decay.

Skills: while loop, Poisson process, random numbers, for loop, fractional exponents (exp), plotting, deal, initialize a vector of nan (nan(1, 1000))

### MATLAB

Code: MasterMATLAB_0280_PoissonCounter.m




## 20. DISPLAY THE DAY OF 1 JANUARY

Use Gauss' date formula to identify the day of 1 January (e.g., Monday Wednesday…) of the year. Use switch-case to convert the number 0-6 into a day, e.g., 0=Sunday.

Use appropriate grammar for past tense vs. future tense. Solve program without using switch!

Skills: mod (modulus), switch, fprintf, clock (returns current date and time).

### MATLAB

Code: MasterMATLAB_0300_DayOf1January.m

```
>> MasterMATLAB_0300_DayOf1January
  1 January 2021 will be a Friday
  1 January 2021: Friday
```

# SECTION 5: IMPORT AND EXPORT DATA

## 21. SAVE AND LOAD MULTIPLE FILES

Use a for-loop to create multiple data files. Then import those files into a cell array and a matrix,

Avoid overwriting files with the same name.

Initialize the matrix containing all data.

Skills: for, continue, mod, disp, dir, zeros, cell

### MATLAB

Code: MasterMATLAB_0320_exportInput.m

## 22. IMPORT FORMATTED TEXT DATA

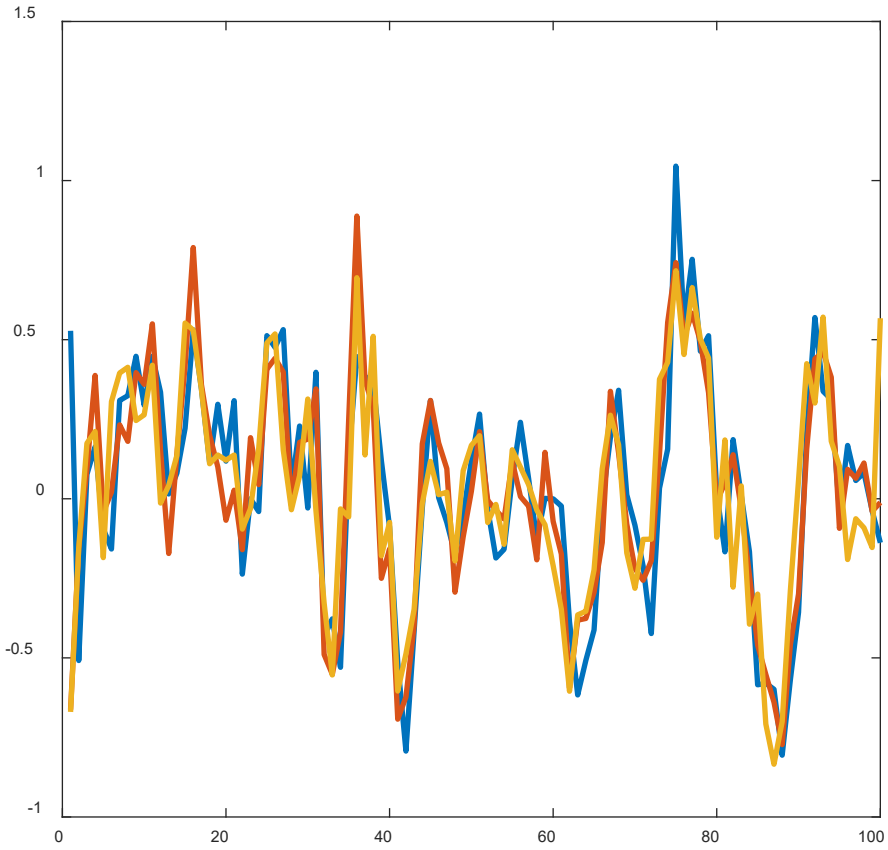
Read numerical data from a formatted tab-separated text file.

Skills: fgetl (file get line), while, toggle, str2double, regexp, strcmpi

### MATLAB

Code: MasterMATLAB_0340_inputtxt.m

Data: datafile.txt




## 23. IMPORT EXCEL-FORMAT DATA

Read data from an Excel file and plot the sensor time series.

Identify missing data points and print a warning message to the Command Window.

Skills: xlsread (trust only the raw data), find, cell2mat, unique, isnan, ind2sub, warning (an alternative to disp), help vs doc command in command window.

### MATLAB

Code: MasterMATLAB_0360_importExcel.m

Data: sensordata.xlsx

## 24. CONVERT US$ TO EUROS USING UP-TO-DATE INFO

Read data from a website to be able to determine the value of $50 US in Euros.

Convert both ways (€ → $ and $ → €)

Skills: urlread, strfind, sscanf

https://transferwise.com/us/currency-converter/usd-to-lkr-rate?amount=1

in html code search for: data-rate

### MATLAB

Code: MasterMATLAB_0380_convertCurrency.m

## SECTION 6: TRANSLATE FORMULAS INTO CODE

### 25. TRIG FUNCTIONS AND GAUSSIANS

Produce and plot trig functions (sine, cosine, tangent) over time, from 0 to 4π. Create and plot a Gaussian ("bell curve"), and explore various parameters of the Gaussian.

Taper (or windowing function) the tangent with a Gaussian.

Skills: sin, cos, tan, exp, Gaussian

Sine wave:

$$\sin(2\pi f t + \phi)$$

- f = frequency
- t = time
- Phi = phase

Gaussian:

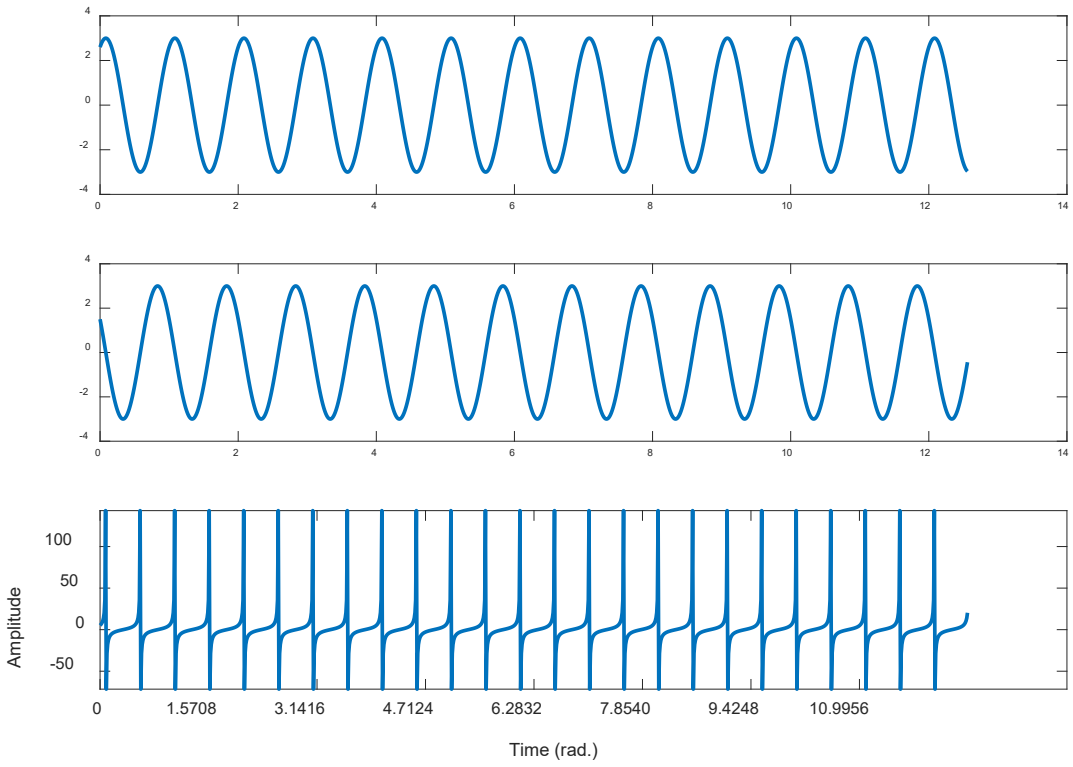
$$ae^{\frac{-(t-c)^2}{2s^2}}$$

- a is amplitude

- t = time
- c = center of curve
- s = spread or width

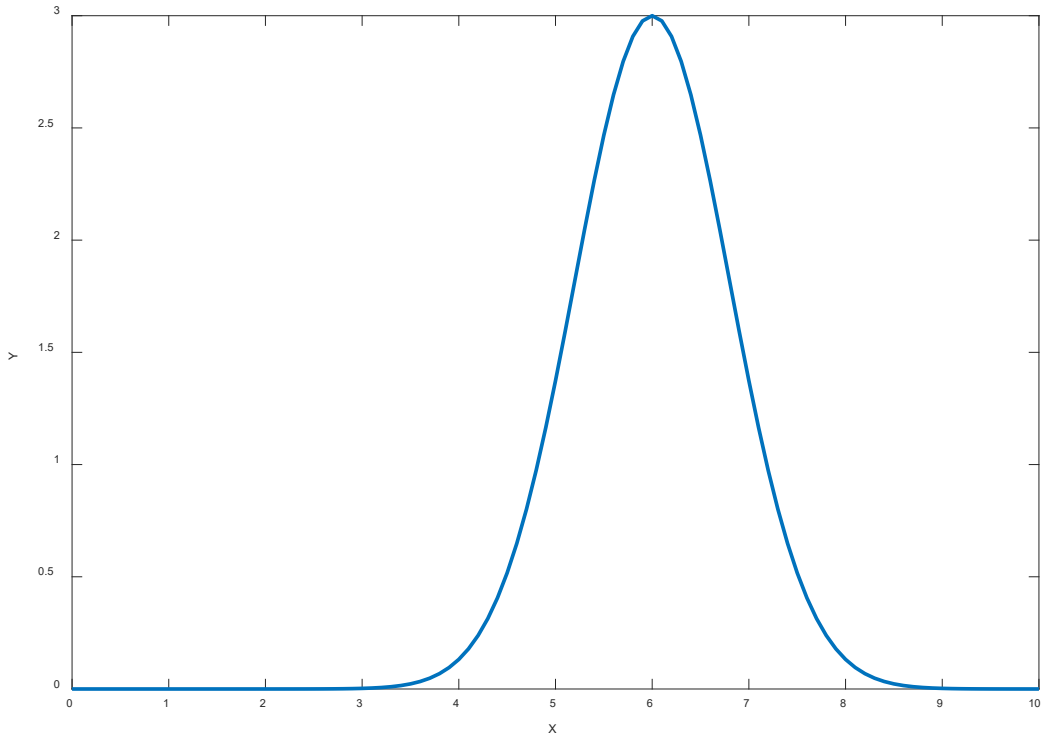
## MATLAB

Code: MasterMATLAB_0400_trigGauss.m

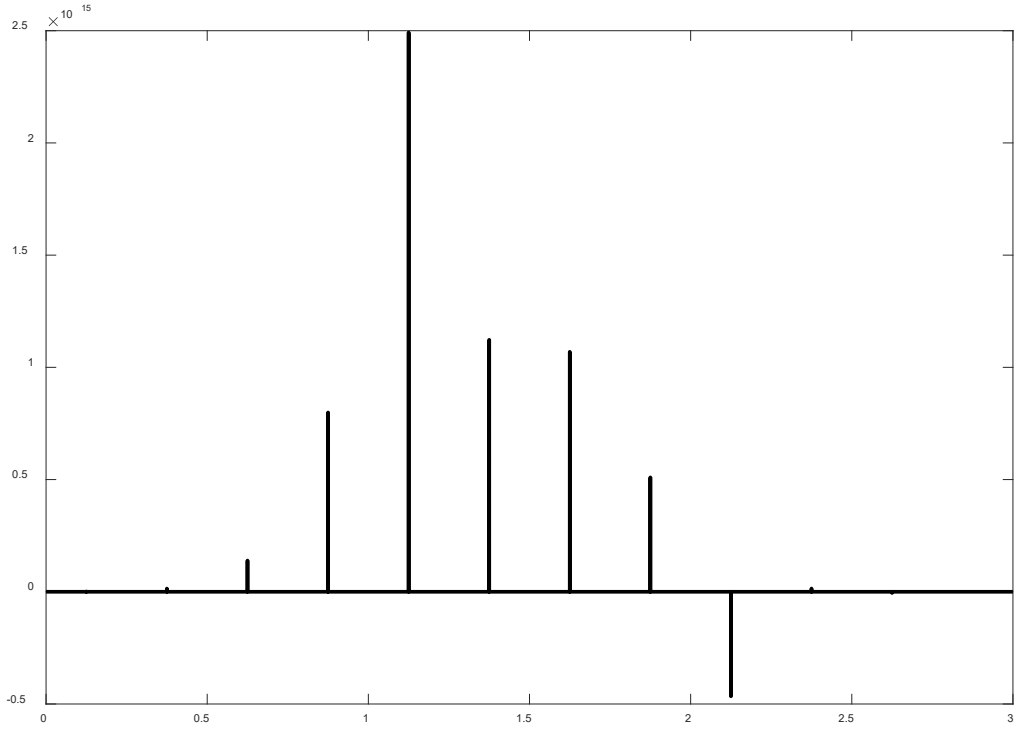



Lines of code from 7 to 36

Note: Tangent plot does not go from +-infinity. Instead the plot is based on the time increment. The smaller the time increment, the higher the plot values will go towards +-infinity since we are only plotting values at time points. The plots are not continuous plots. But the symbolic toolbox can be used to make continues plots.




Lines of code from 36 to 56




Lines of code from 58 to 89

Note: trig function frequency is very sensitive to plot. A frequency of 2 produces this plot, but a frequency of 2.3 produces only one line in this plot.

## 26. LAPLACE AND LOG-NORMAL DISTRIBUTIONS

Produce and plot the Laplace distribution and the log-normal distribution (equations on the following slides).

Normalize to probability densities (PDFs). Plot the CDFs (cumulative density functions).

Note: Probability Density: sum of all values should equal 1

Skills: exp (natural exponential), sqrt, subplot (use commas when more than 9 dimensions or when using variables), cumsum (cumulative sum), linspace

## Laplace distribution:

$$f(x) = .5\lambda e^{-\lambda|x|}$$

$$x = [-5,5]$$

$$\lambda = 3$$

Natural Exponential function:

$$f(x) = \frac{e^{-(\log(x)-m)^2/2s^2}}{sx\sqrt{2\pi}}$$

$$x = [0,5]$$

$$m = 0$$

$$s = .5$$

PDF(probability distribution function) on top, CDF (cumulative distribution function) on bottom

Log-normal distribution useful in physics and biology.

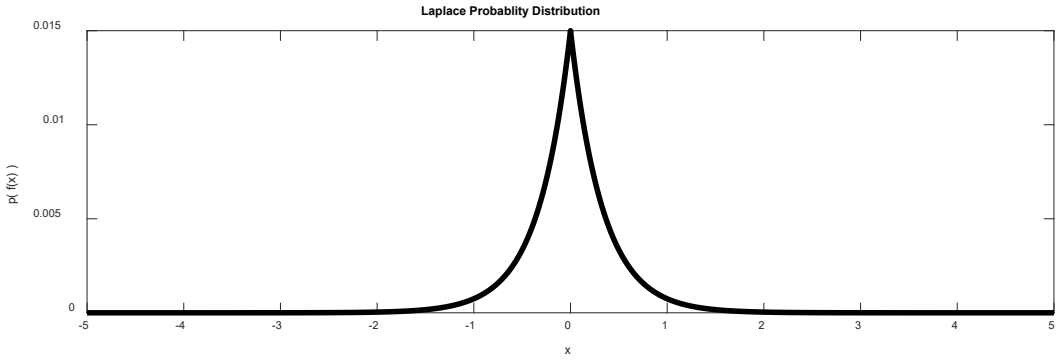
m is a center point.

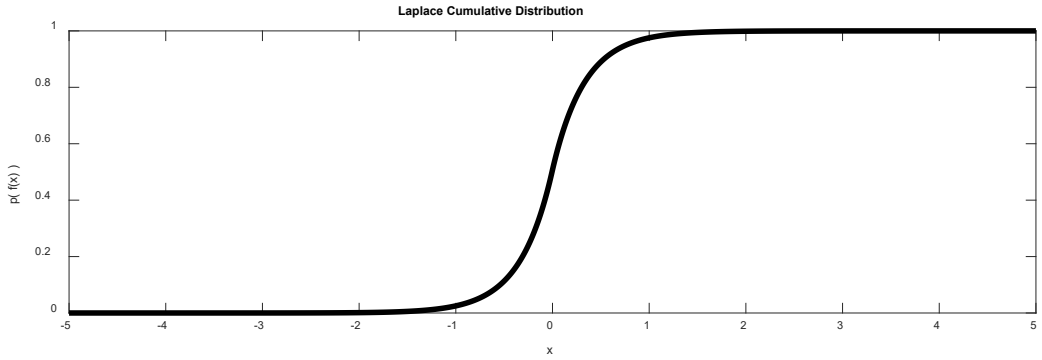
s is a width parameter.

Defined only for positive values of x

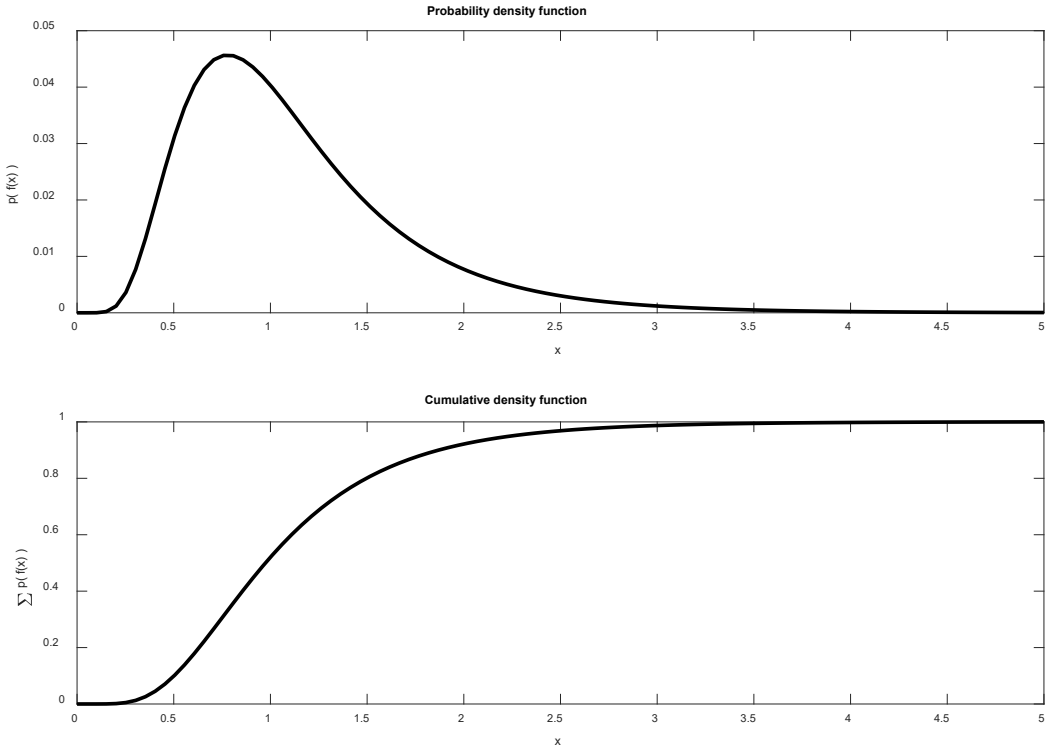
MATLAB

Code: MasterMATLAB_0420_e.m




Lines of code: 9 to 33




Lines of code from 34 to 63

## 27. COMPLEX NUMBERS AND EULER'S FORMULA

Given two real numbers (a,b), create a complex number a+ib.

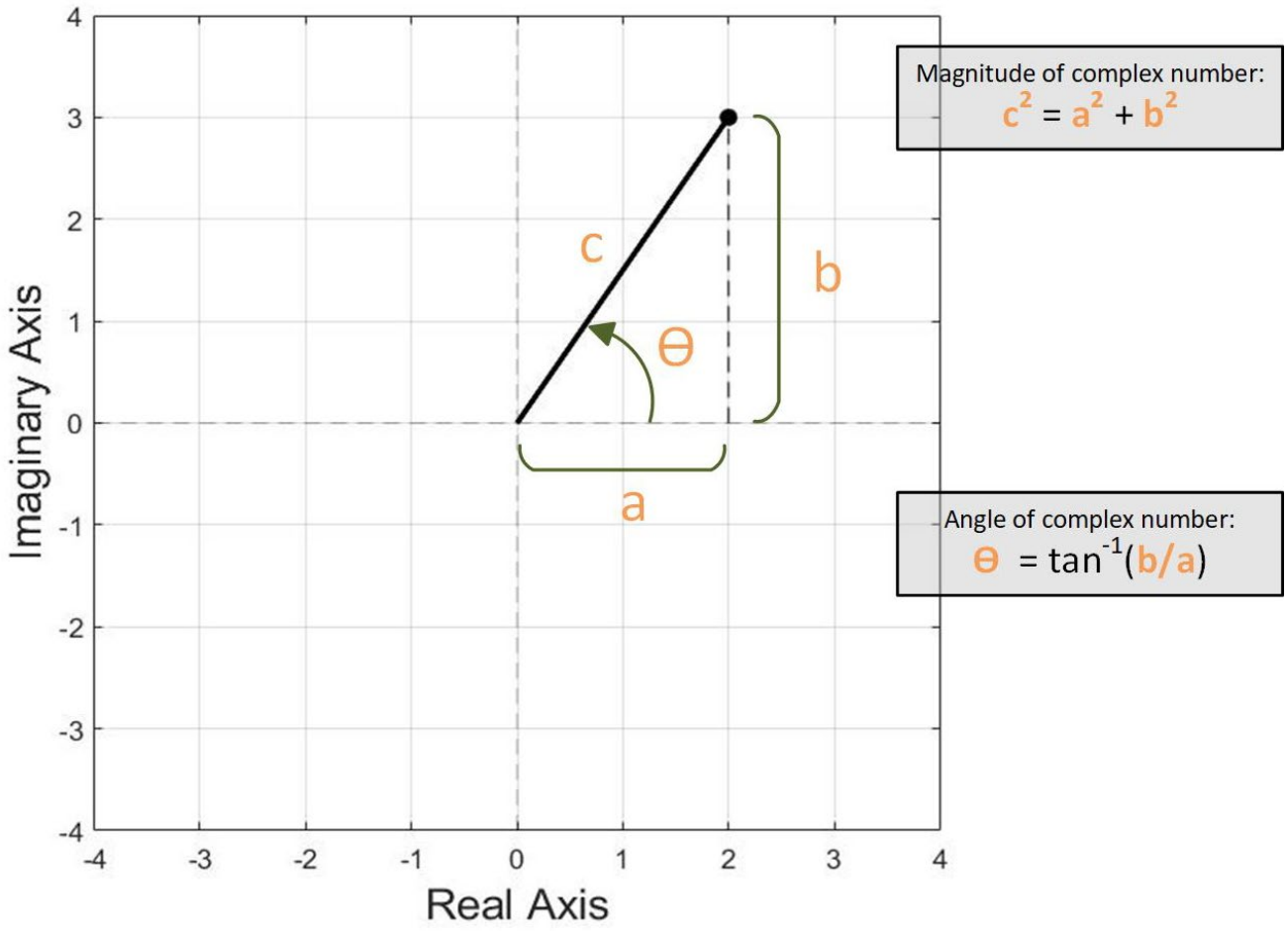
Create a complex number using Euler's formula and show that $e^{ik} = \cos(k) + isin(k)$.

Extract magnitude and phase angle of Euler's format number.

Demonstrate the law of exponents: $e^{p+q} = e^{p}e^{q}$

Skills: complex, 1i, exp, abs (for complex numbers abs is distance away from origin, i.e. magnitude), angle

# Complex Numbers and Euler's Formula




To calculate phase angle using arc tangent function:

- Phase = atan( imag(C)/real(C) )
- But the value in radians will depend on which of the 4 quadrants the complex number resides.
- One solution that almost works: atan(y./x) + (x<0)*pi - (x<0 & y<0)*2*pi, but will not work when x (real part) is zero
- So the most common solution is to use the 4 quadrant atan2 function: atan2( imag(C),real(C) )

## MATLAB

Code: MasterMATLAB_0440_complexEuler.m

```
>> MasterMATLAB_0440_complexEuler
m*exp( 1i*p ) produces the same result as m*( cos(p) + 1i*sin(p) )

ans =
```

5.5511e-17

## 28. PIECEWISE FUNCTIONS

Implement the piecewise function (following slide) in MATLAB using a for-loop and if-else statements.

Use *dsearchn* to implement the function more accurately and without control statements.

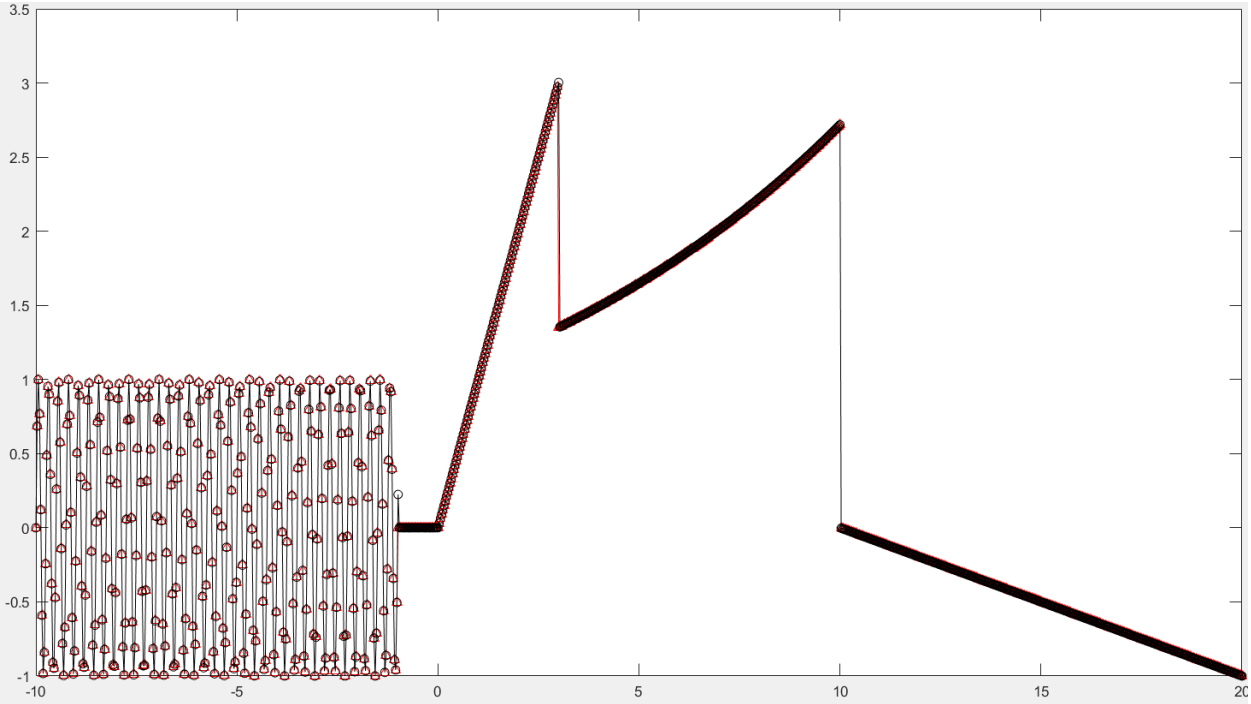
Skills: elseif, linspace, dsearchn

## Piecewise Functions:

$$y = \begin{cases} \sin(2\pi 4x), & x \leq -1 \\ 0, & 1 < x \leq 0 \\ x, & 0 < x \leq 3 \\ e^{x/10}, & 3 < x \leq 10 \\ 1 - x/10, & 3 < x \end{cases}$$

## MATLAB

Code: MasterMATLAB_0460_piecewise.m




Lines of code from 7 to 43 using if-elseif-else, and 66 to 88 using dsearchn

## 29. PIECEWISE FUNCTION IN ONE LINE OF CODE

Implement a piecewise function using a single line of MATLAB code.

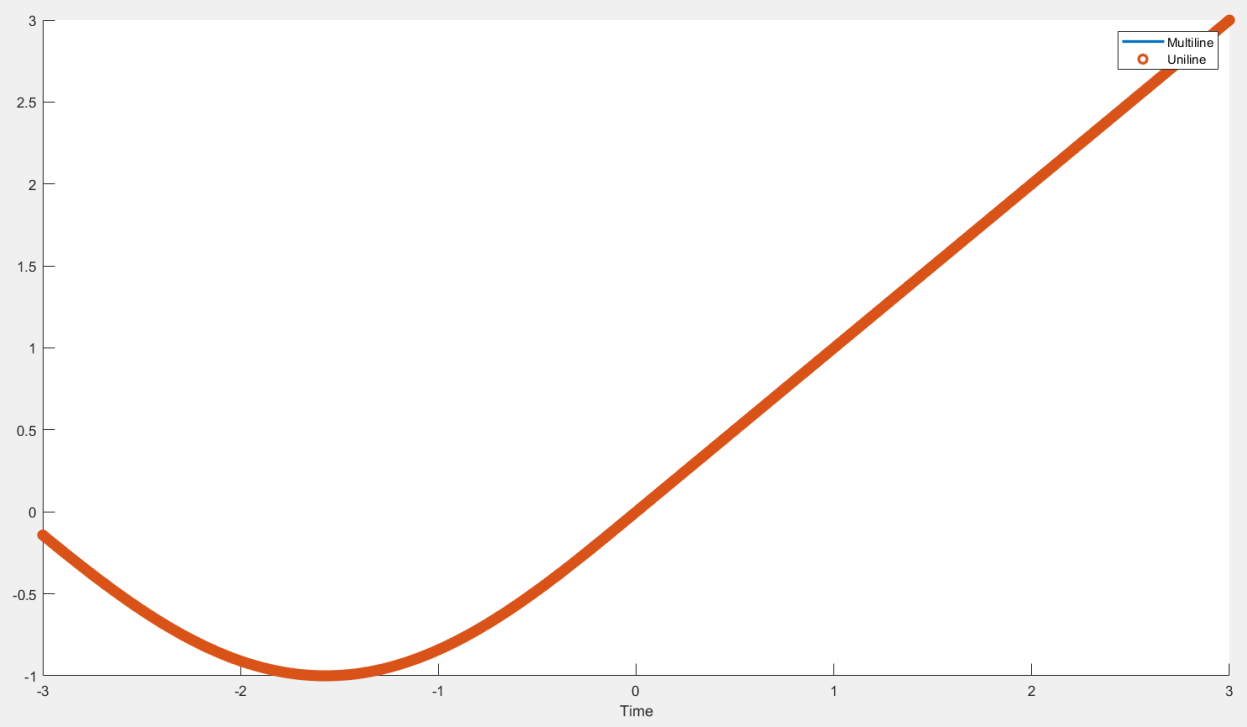
Skills: dsearchn, sin

## Piecewise Function in One Line of Code:

$$f(x) = \begin{cases} \sin(t), & t \leq 0 \\ t, & 0 > t \end{cases}$$

### MATLAB

Code: MasterMATLAB_0480_piecewise1line.m




## 30. SIGMOID FUNCTION

Generate and plot a sigmoid function.

A.k.a. Asymptote function.

Plot dotted lines at the sigmoid parameter crossing (inflection point a.k.a. equivalence point).

Skills: Exp, linspace, get

# Sigmoid Function:

$$f(x) = \frac{a}{1 + e^{-b(x-c)}}$$

Where:

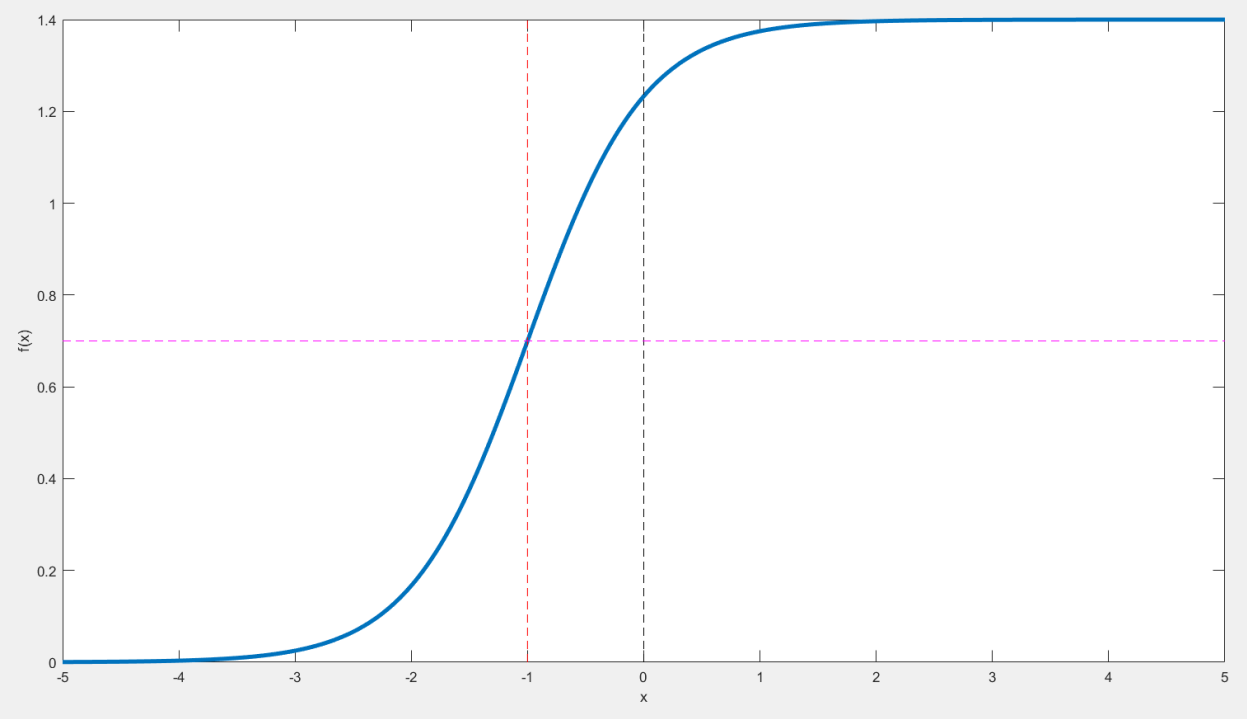
a is the height of the plot – set to 1.4 in this graph

b is the temperature or heat parameter – larger b makes the curve look more like a square wave; small b creates a gentle slope between the two asymptotes.

c is the center point, default is zero, but it is set to -1 in this graph. Also known as the inflection point or equivalence point. This is also where the derivative is zero.

### MATLAB

Code: MasterMATLAB_0500_sigmoid.m

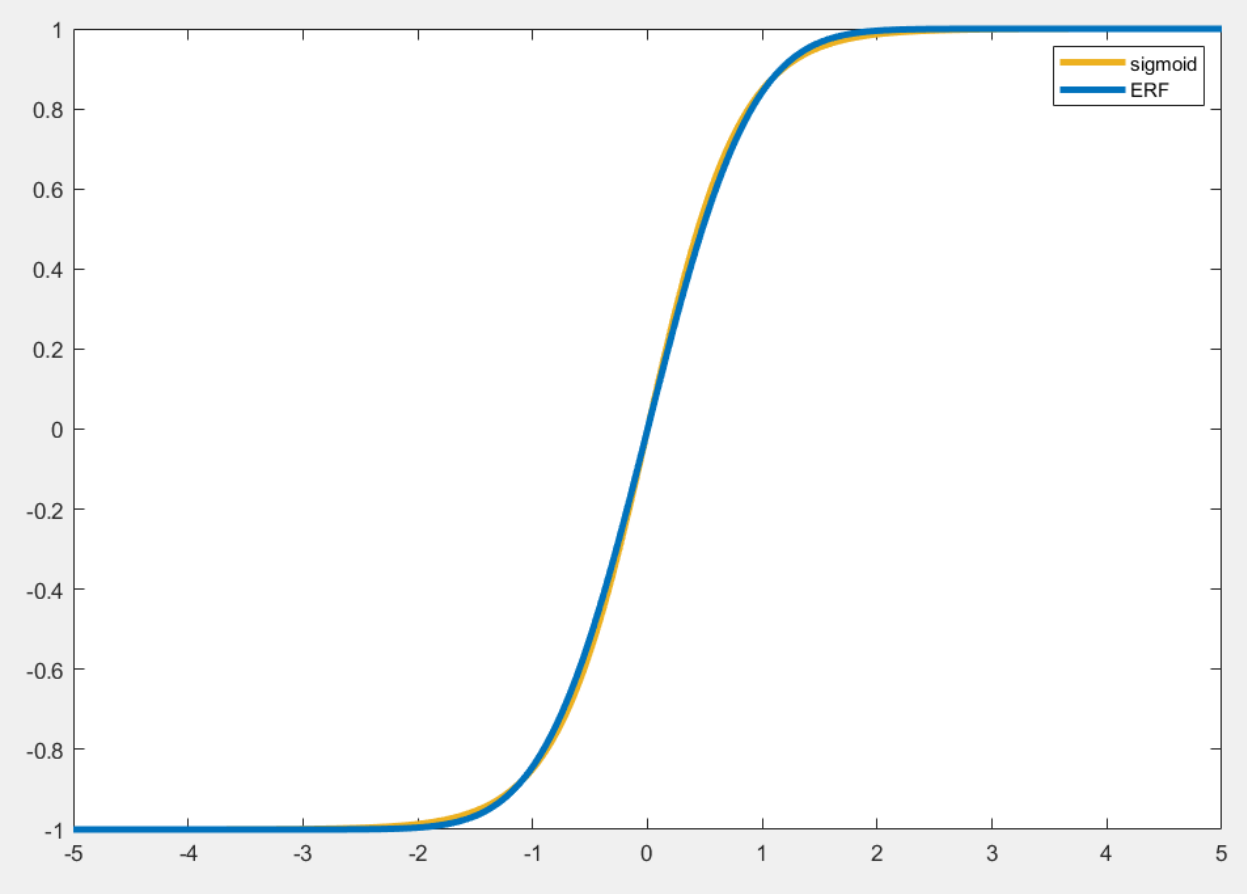



## 31. SOLVED: SIGMOID AND ERROR FUNCTION

Parameterize a sigmoid function to match the error function.

### MATLAB

Code: solvedProject_sigmoid_erf.m




## 32. CIRCULAR P-VALUE AND ITS APPROXIMATION

Implement a 2-parameter p-value computation and its approximation for a range of input parameters.

Plot the two functions and their difference. Determine whether the approximation I useful.

Skills: exp, sqrt, contour

## Circular p-value and its Approximation:

$$p_f = e^{\sqrt{1+4n+4(n^2-(nz)^2)}-1+2n}$$

$$p_a = e^{-nz^2}$$

$$n \in \mathbb{Z}(1,100)$$

$$z \in \mathbb{R}(0,1)$$

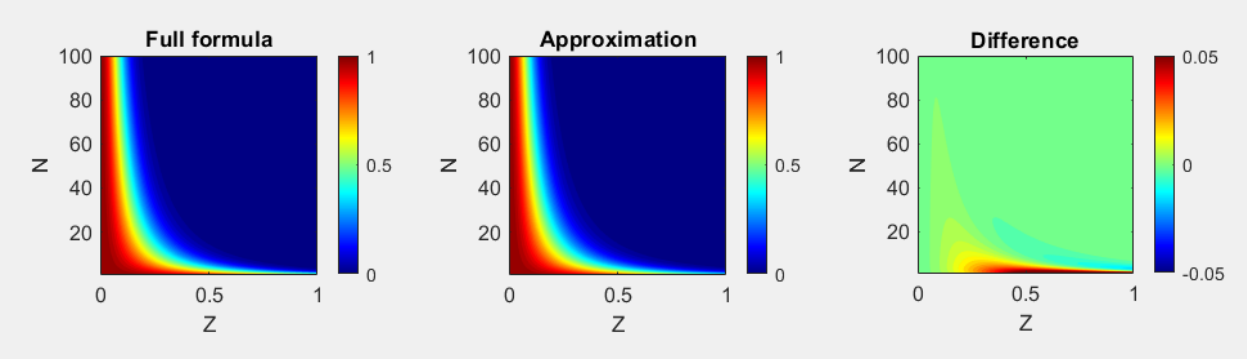
Where:

- n is the number of trials, an integer ranging from 1 to 100
- z is a circular plotting control, a real value between 0 and 1
- Pf is the full formula
- Pa is the approximate formula

Difference is very good for trial values, n, above 10

### MATLAB

Code: MasterMATLAB_0520_circPval.m




## SECTION 7: DESCRIPTIVE STATISTICS

### 33. COMPUTE MEASURES OF CENTRAL TENDENCY

Implement algorithms to compute the mean, median, and mode of a log-normal distribution.

Plot lines at these three distribution values on top of the data histogram.

Skills: hist, sort, mean, median, mode, unique, max, randn (normal distributed random numbers), numel (number of elements in the dataset, similar to length() )

Monotonic: (of a function or quantity) varying in such a way that it either never decreases or never increases.

Skills related to round: floor (rounds down), ceil (rounds up), fix (round towards 0).

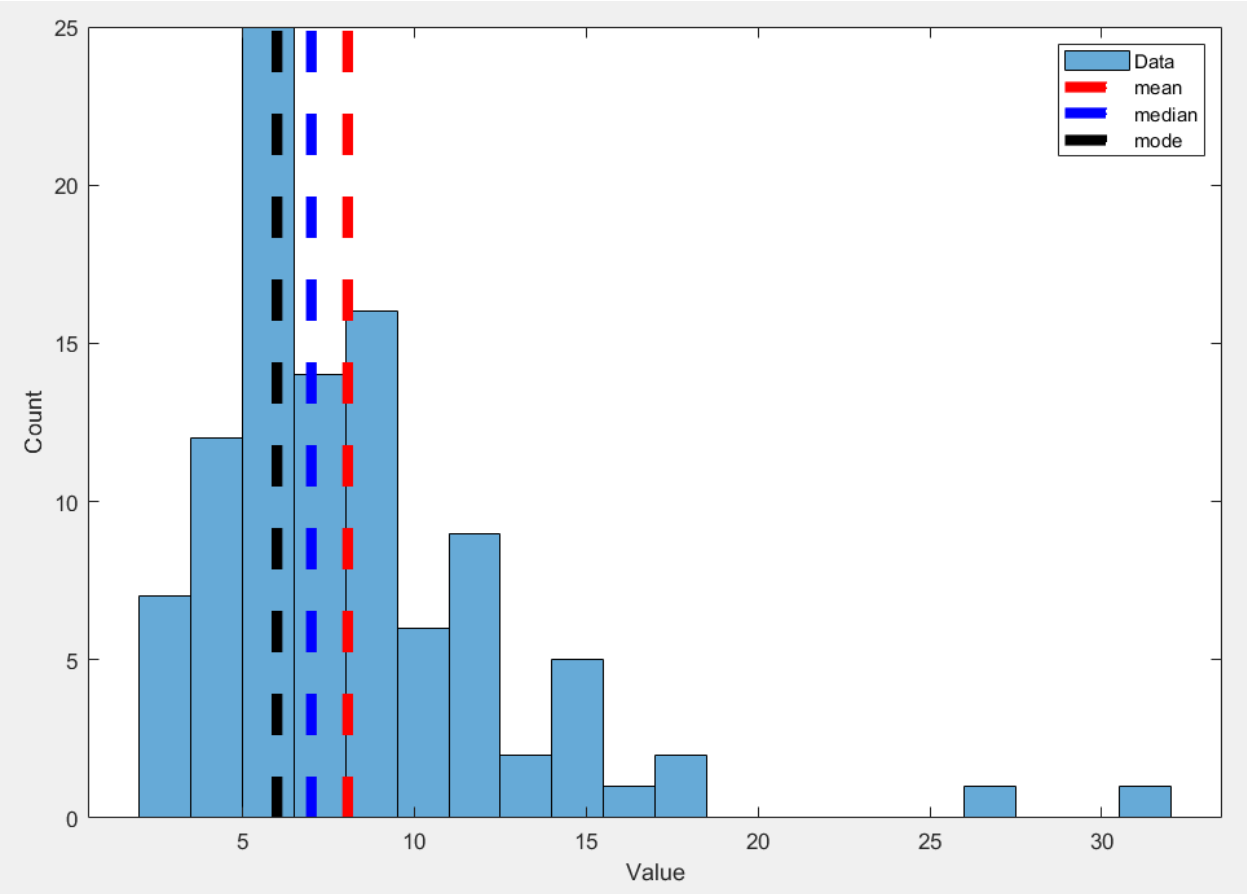
**Mean**: sum/N

**Median**: middle of the distribution

**Mode**: most common value

### MATLAB

Code: MasterMATLAB_0540_centralTendency.m




## 34. COMPUTE VARIANCE AND STANDARD DEVIATION

Implement algorithms to compute the variance and standard deviation of a log-normal distribution.

Implement variance and standard deviation either with or without first mean-centering.

Skills: var, std

**Mean centering**: the act of subtracting a variable's mean from all observations on that variable in the dataset such that the variable's new mean is zero.

**Log-normal distribution**: is a continuous probability distribution of a random variable whose logarithm is normally distributed.

## Standard deviation:

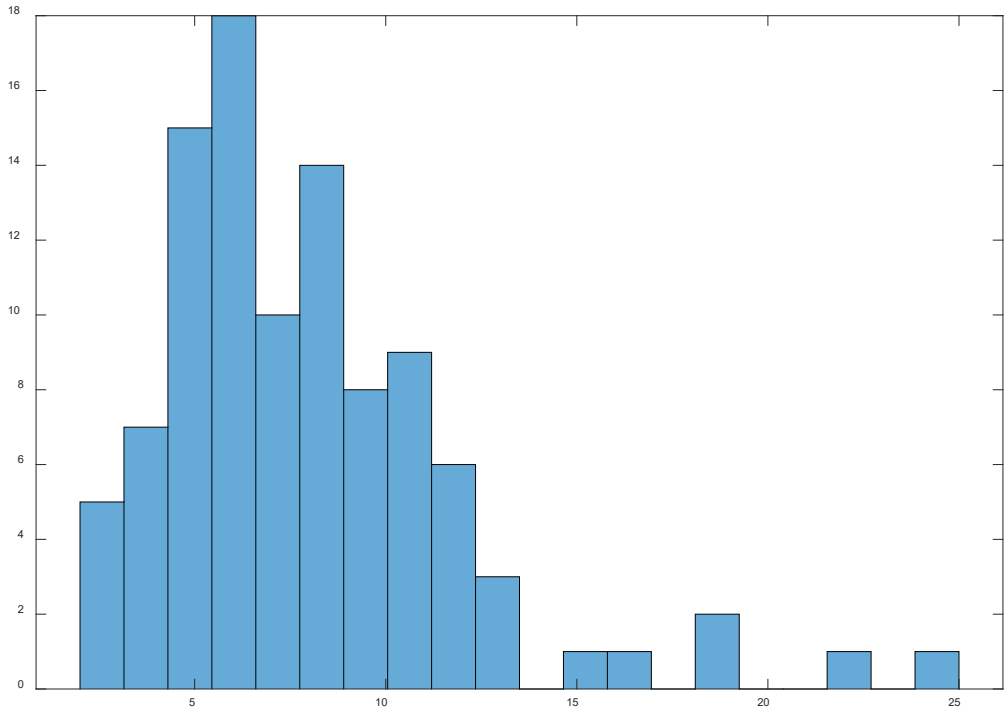
$$\sigma = \sqrt{(n-1)^{-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

## Variance:

$$\sigma^2 = (n-1)^{-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

MATLAB

Code: MasterMATLAB_0560_varStDev.m




## 34. SOLVED: SORT DATA P AND DOWN

Generate 5 random integers (variance=10) and sort up and down.

MATLAB

Code: solved_Data_Sort.m

```
v =
    0    3    2   -1    3
vup =
   -1    0    2    3    3
vdn =
    3    3    2    0   -1
```

## 35. DATA TRANSFORMATIONS (LOG, SQRT, RANK)

Apply several transformations (log, square root, rank) to log-normal distributed data.

Use the 3-step RSH algorithm to transform any distribution into a normal (Gaussian) distribution.

Skills: hist, sqrt, log, tiedrank, atanh (and Fisher z-transformation)

Data Transformations:

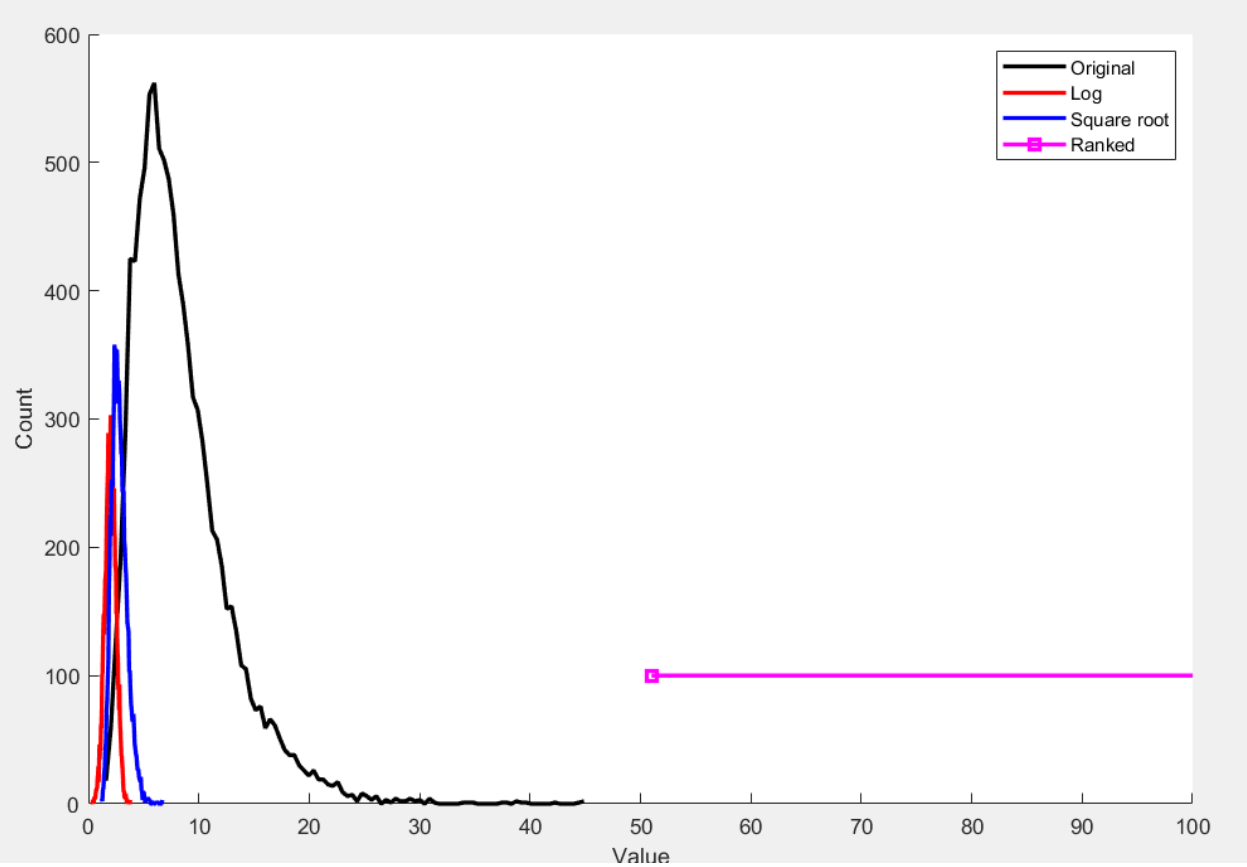
**Rank**: rank-order

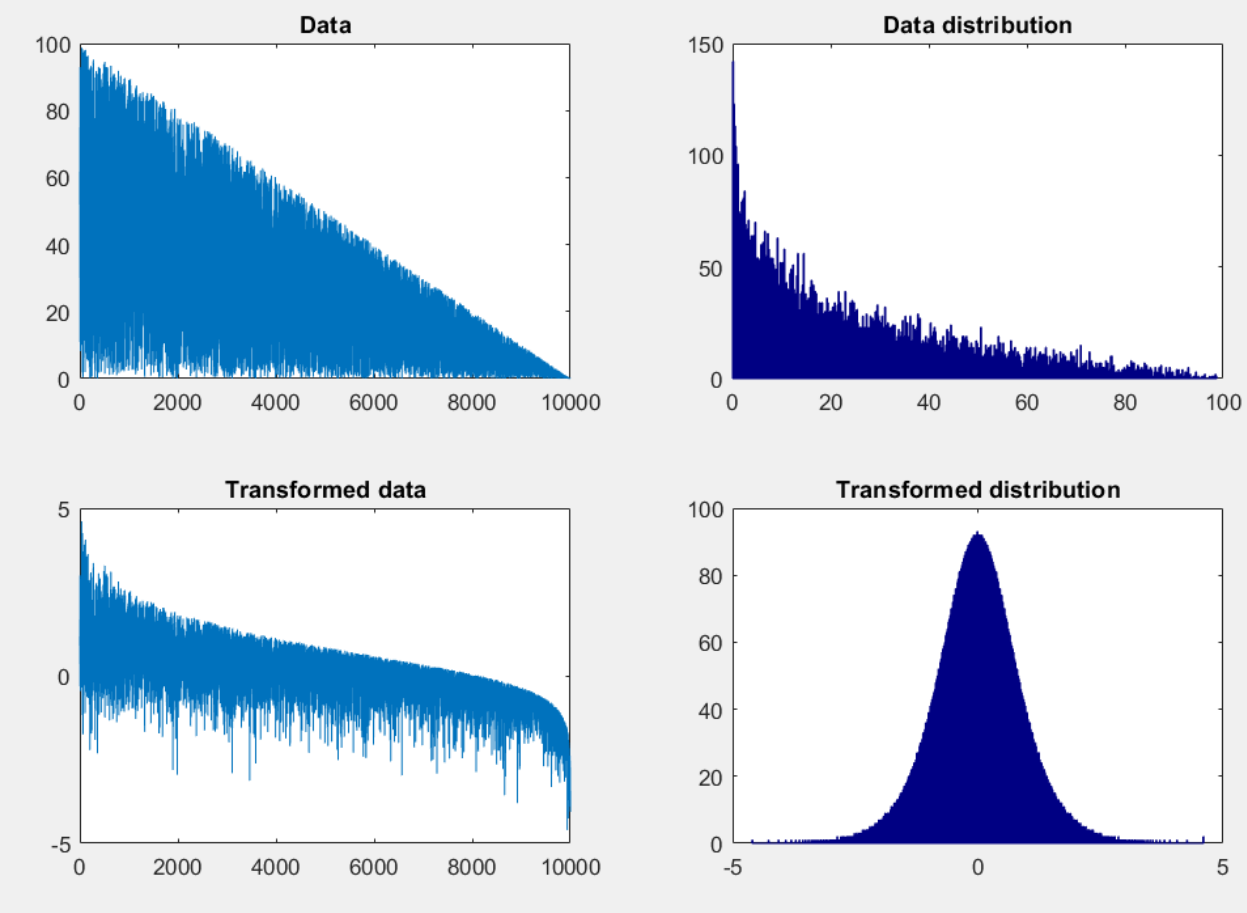
**Scale**: to [-1 +1]

**Htan**: inverse hyperbolic tangent

## MATLAB

Code: MasterMATLAB_0580_transformations.m






## SECTION 8: 2D PLOTTING

### 37. LINES

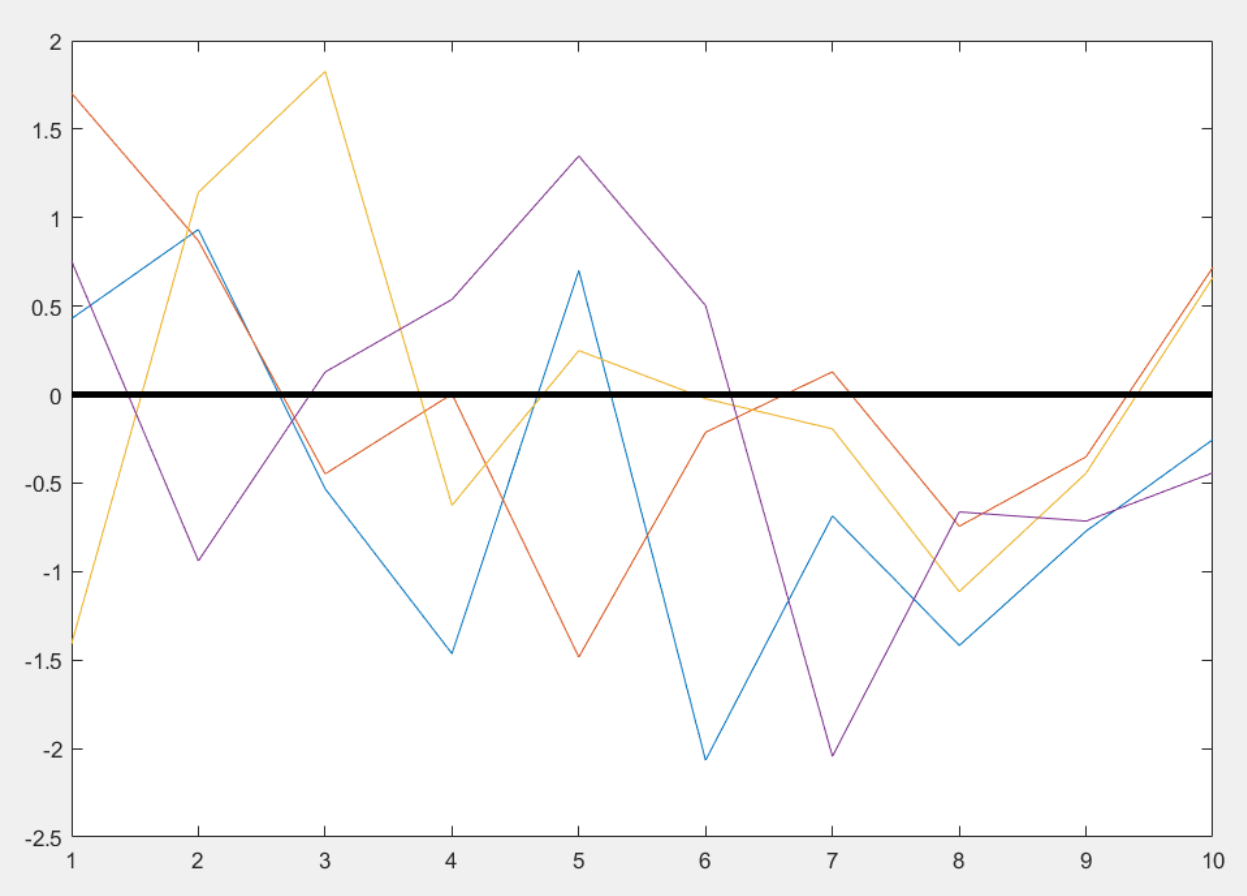
Use a for-loop to plot a curve from straight lines.

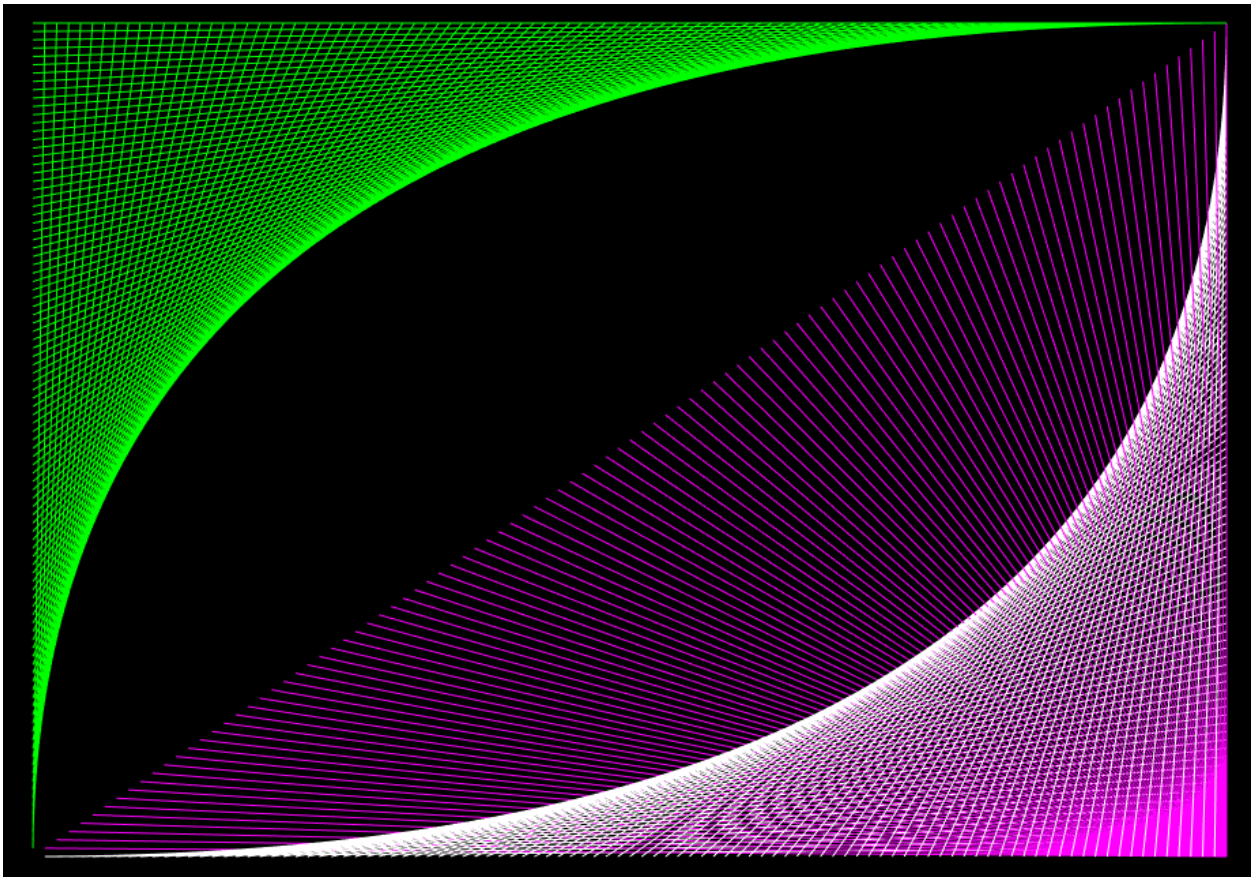
1970's disco party!

Skills: for, plot, hold on, set

### MATLAB

Code: masterMATLAB_0600_lines.m


2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
1
2
3
4
5
6
7
8
9
10



## 38. BARS AND ERRORBARS

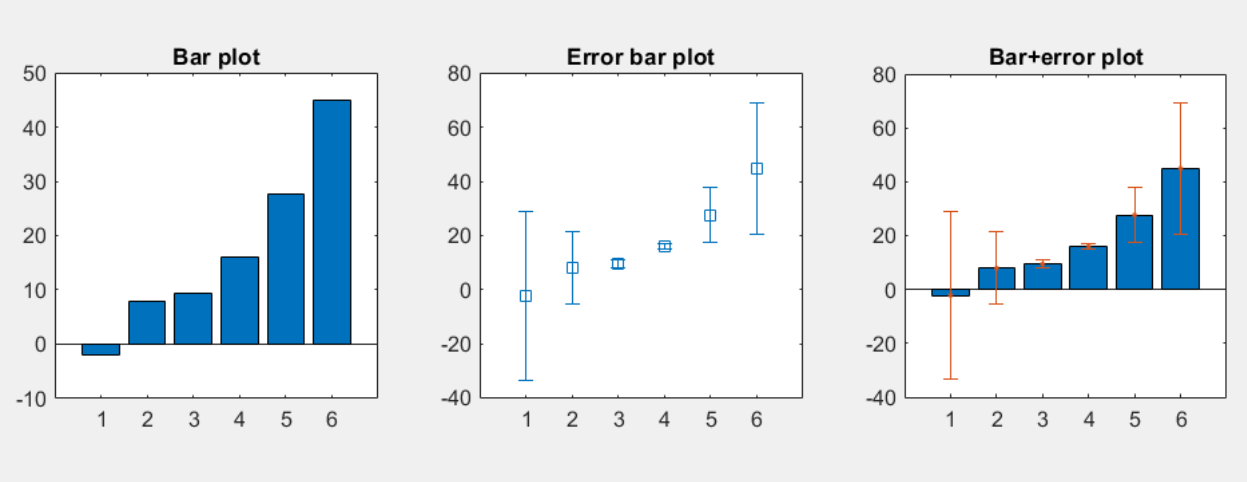
Create a 30 (observations) by 6 (features) random matrix, and show the means and standard deviations of each feature using bars and errorbars.

Create the dataset such that the mean increases linearly, and the variance increases quadratically, as a function of feature number.

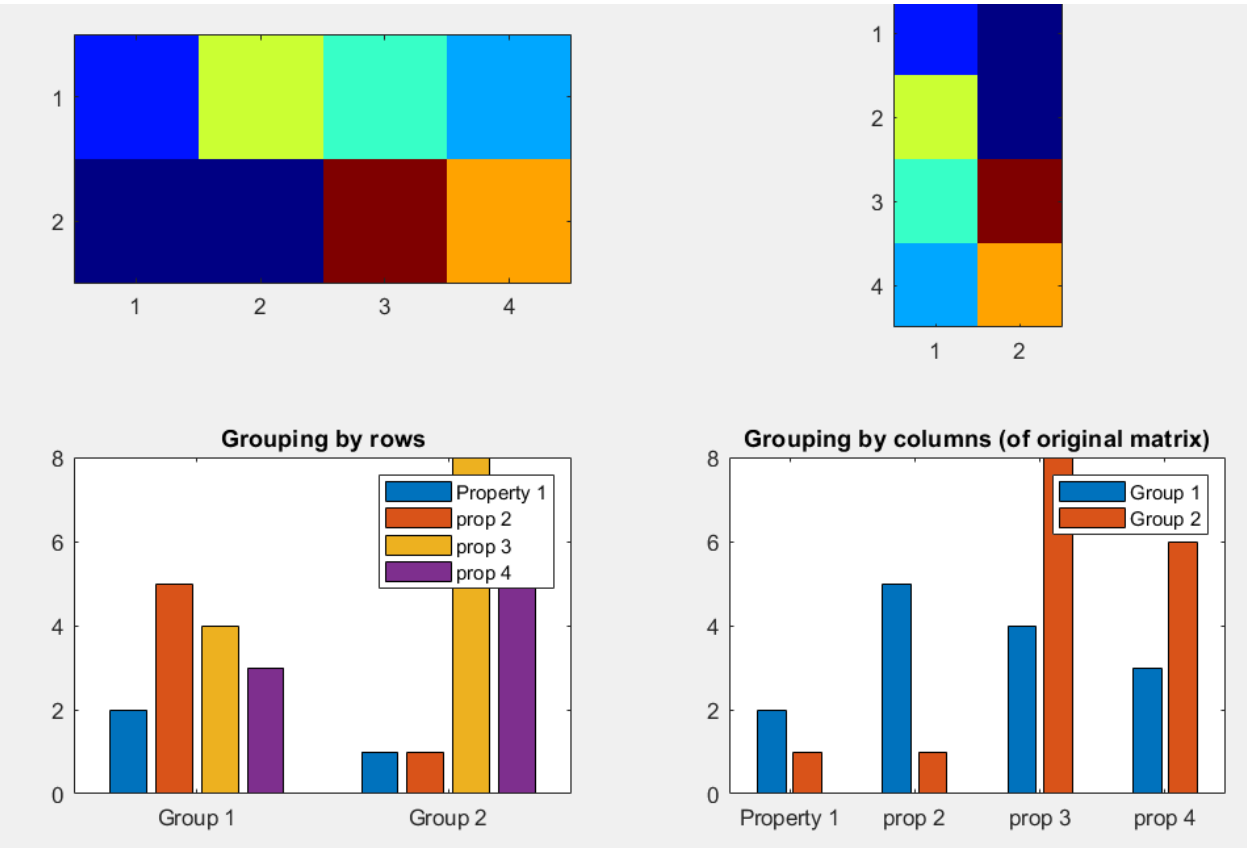
Skills: bar, errorbars, hold, bsxfun, randn, linspace, imagesc

### MATLAB

Code: masterMATLAB_0620_bars.m




Code lines 21 to 56




Code lines from 58 to 88

Plot normally distributed random numbers that become exponentially further apart. Use set to change the line to red stars with blue outline.

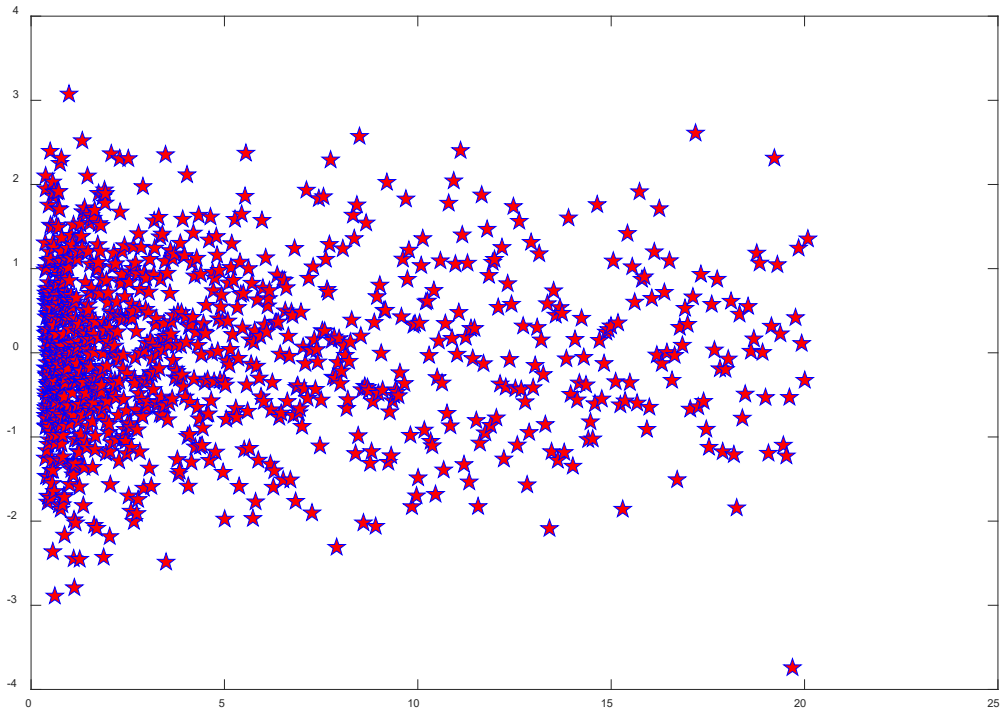
Use the scatter function to specify a different color for each dot.

Skills: Bar errorbar, hold bsxfun, randn, scatter, set

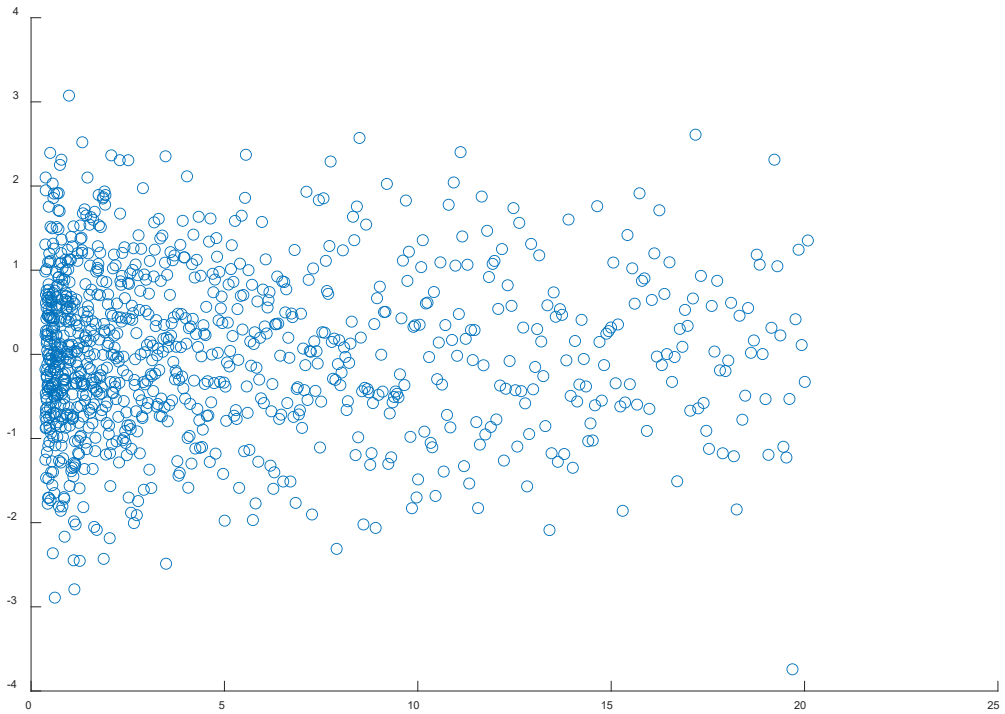
Short hand notation for normal distribution with mean of 0 and variance of 1: ~N(0,1)

## MATLAB

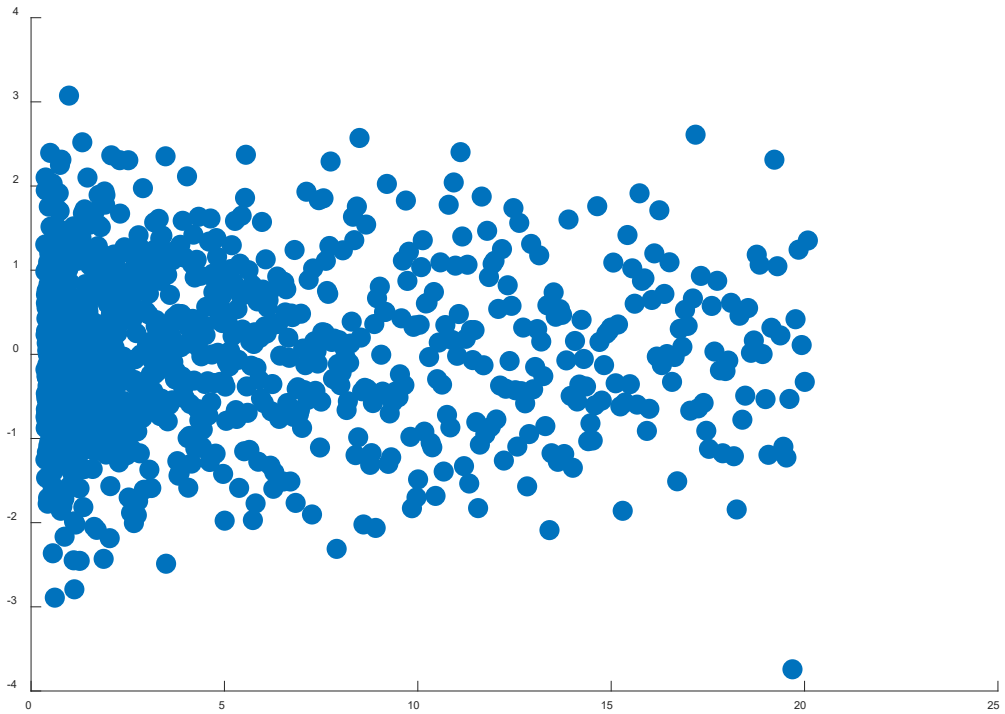
Code: masterMATLAB_0640_dots.m



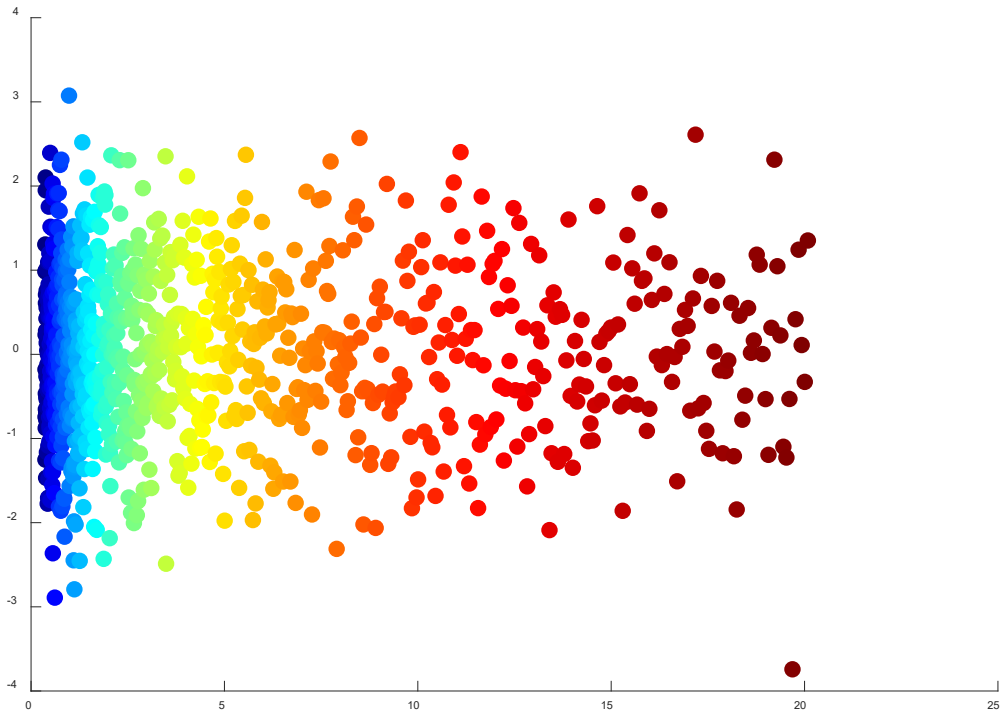

Lines of code from 7 to 27



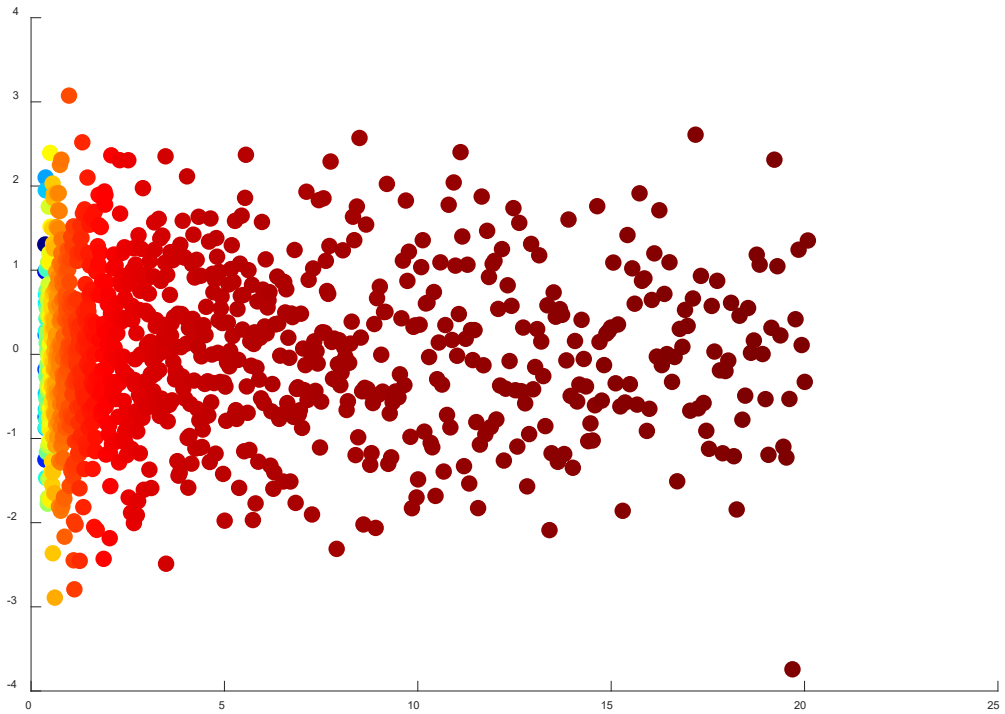

+ Line 31



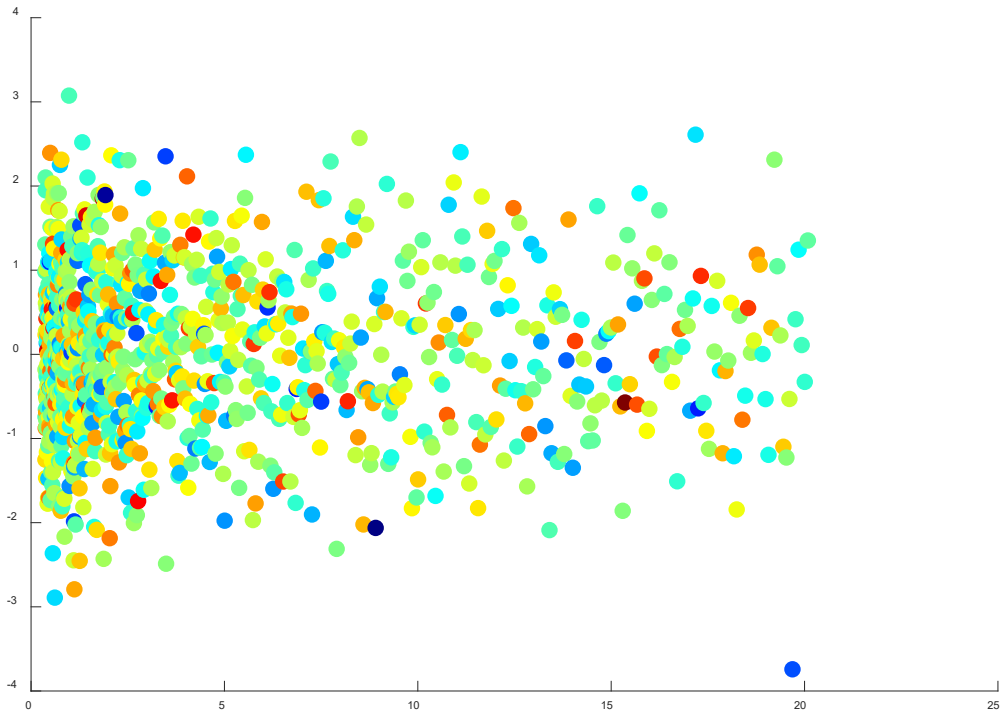

+Line 34




+Line 37




+ Line 38




+Line 39

## 40. MULTIDIMENSIONAL DATA WITH COLORED SCATTER

Generate a 3D nonlinear dataset. Make a scatterplot of the first two dimensions while using the third dimension to define the color of dots.

Explore the effects of noise on the results.

Skills: linspace, sin, exp, scatter

## Equations Used:

$$x: (-1, 1)$$

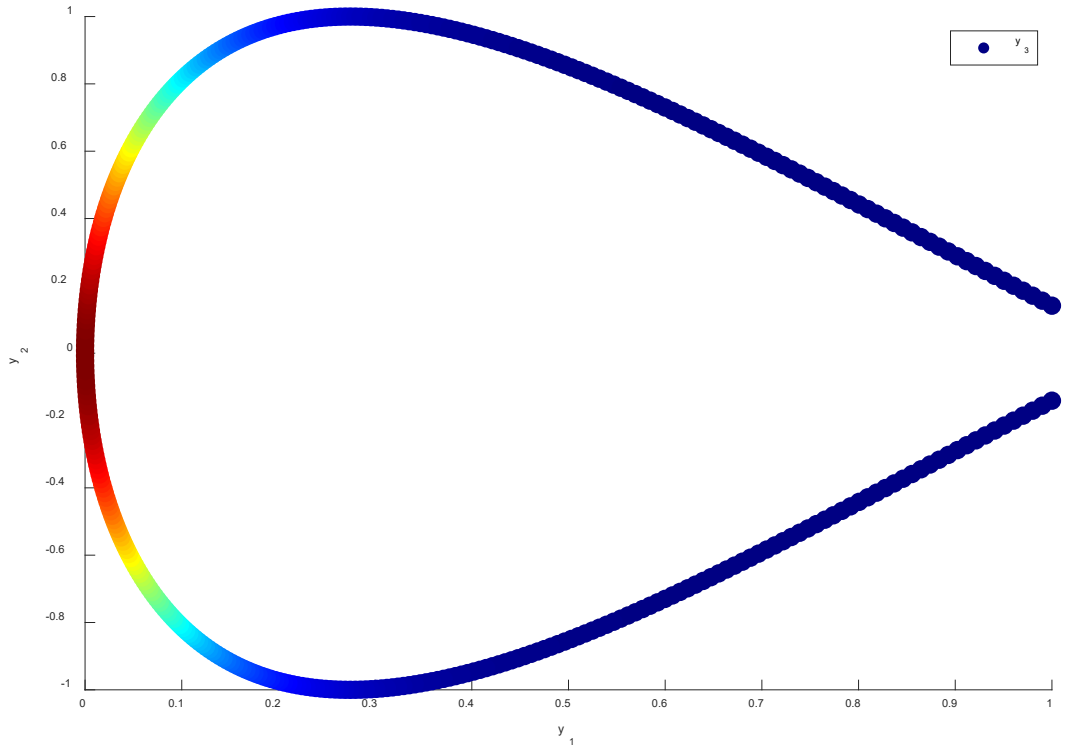
$$y_1: x^2$$

$$y_2: \sin(3x)$$

$y_3 : e^{-10x^2}$

## MATLAB

Code: masterMATLAB_0660_multiDots.m




## 41 SOLVED: IMAGESC VS. PCOLOR

Understand the difference between *imagesc* and *pcolor* plotting commands

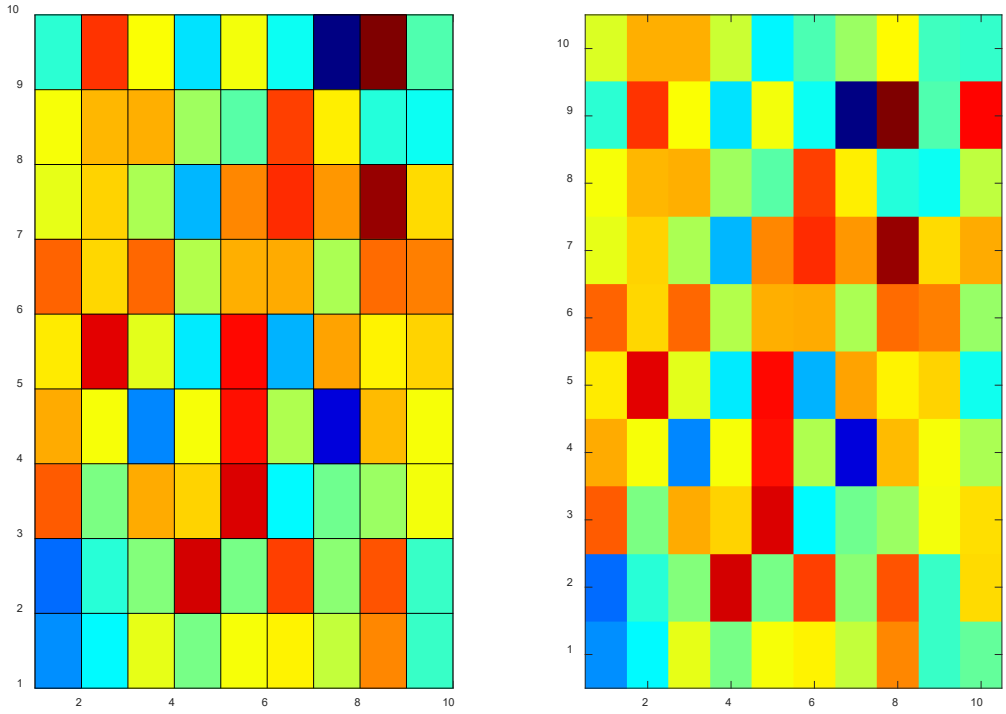
Skill: pcolor, imagesc, axis xy

```
ri = randn(10);
subplot(121),   pcolor(ri)
```

```
subplot(122),    imagesc(ri), axis xy
```

### MATLAB:

Code: solved_imagesc_pcolor.m




Pcolor (left), imagesc, axis xy (right)

## 42 HISTOGRAMS

Generate log-normal random data $e^{r/2}$, r~N(0,1)) and show its histogram using 40 bins. Then apply the Freedman-Diaconis rule to identify the "optimal" number of bins.

Generate a movie of histograms with bin counts of 5 to N/2 (N data points).

Skills: histogram, hist, set, ceil, iqr

Freedman-Diaconis rule:

$$Bin\ size = 2\frac{IQR(z)}{\sqrt[3]{n}}$$

Where:

IQR(z) is the interquartile range: the distance between 25% of the data and 75% of the data

Calculate number of bins k:

$$k = \left\lceil \frac{\max x - \min x}{h} \right\rceil$$

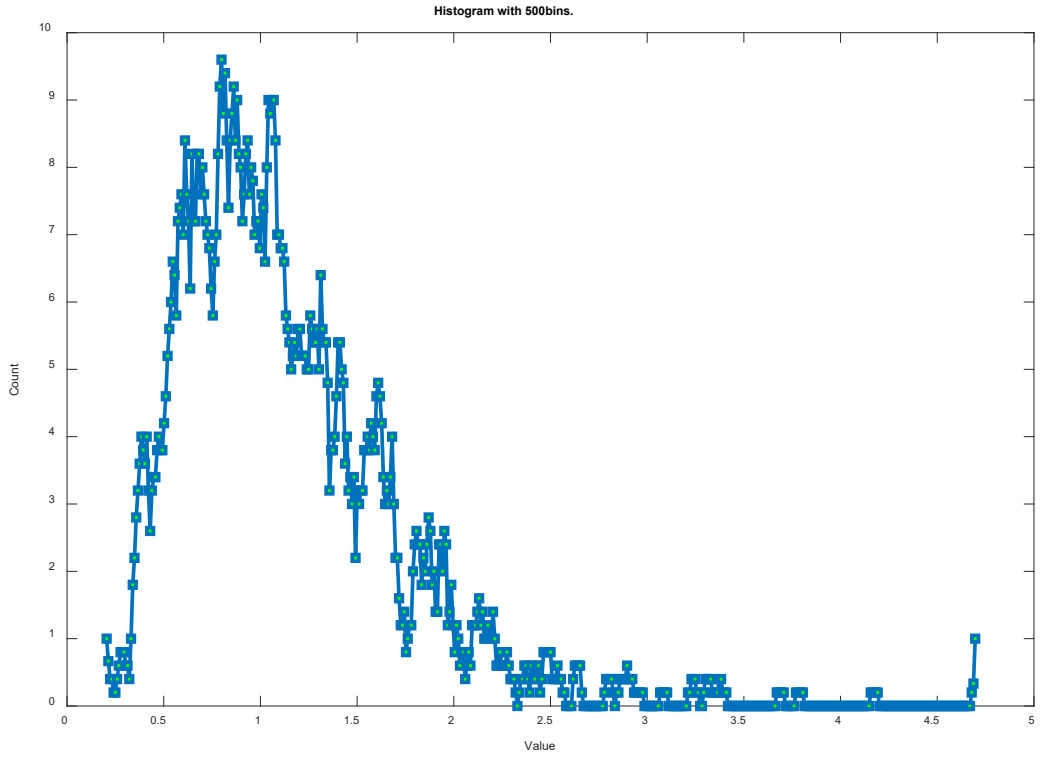
Where:

k is the number of bins

h is the bin width.

MATLAB

Code: masterMATLAB_0680_hist.m




## 43. UNCERTAINTY IN FUTURE MONEY (USING PATCH)

Calculate the present-day value of $100,000 over 10 years, assuming inflation rates of 1%, 2%, ,and 3%. Plot the value ranges using lines and patches.

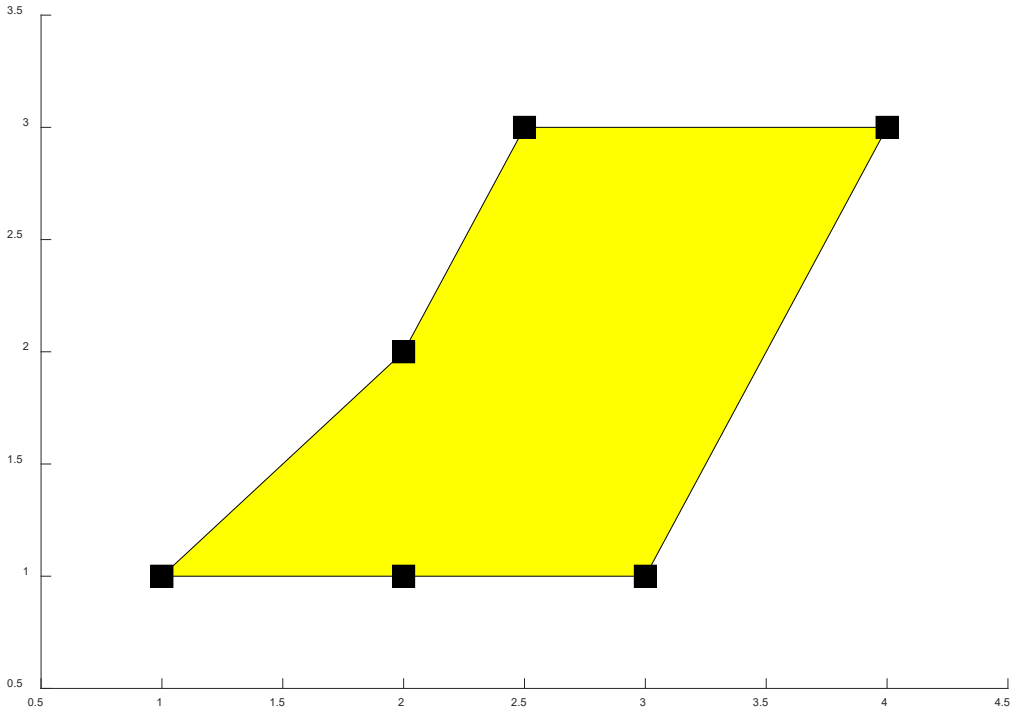
Bonus Understand how to use the patch function with matrix inputs.

Skills: patch

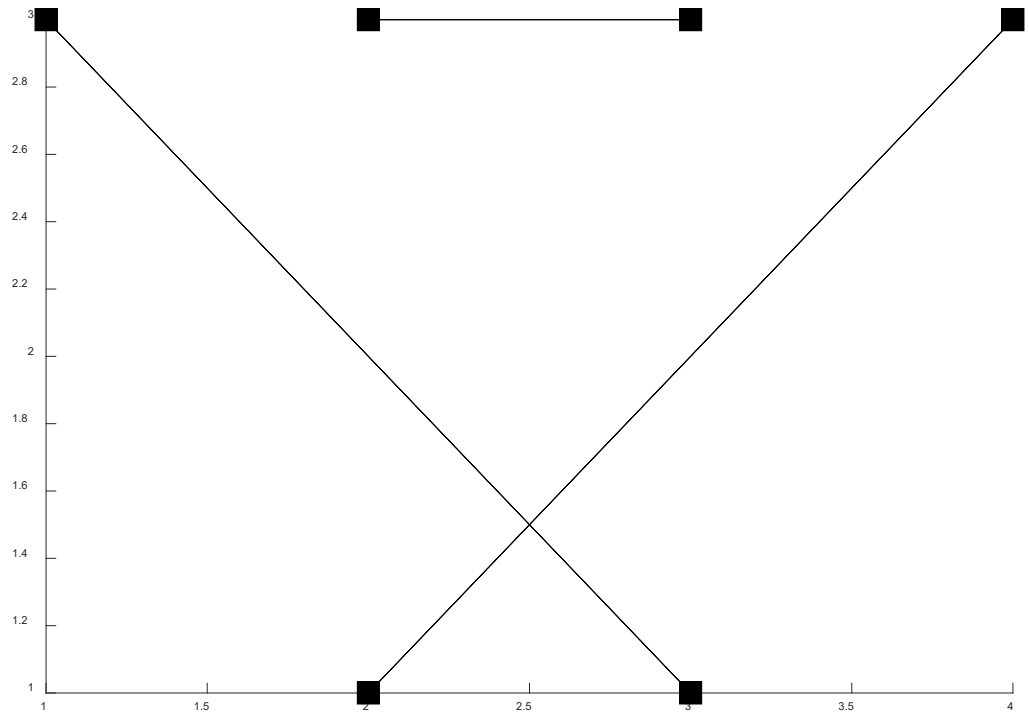
MATLAB

Code: masterMATLAB_0700_moneyPatch.m

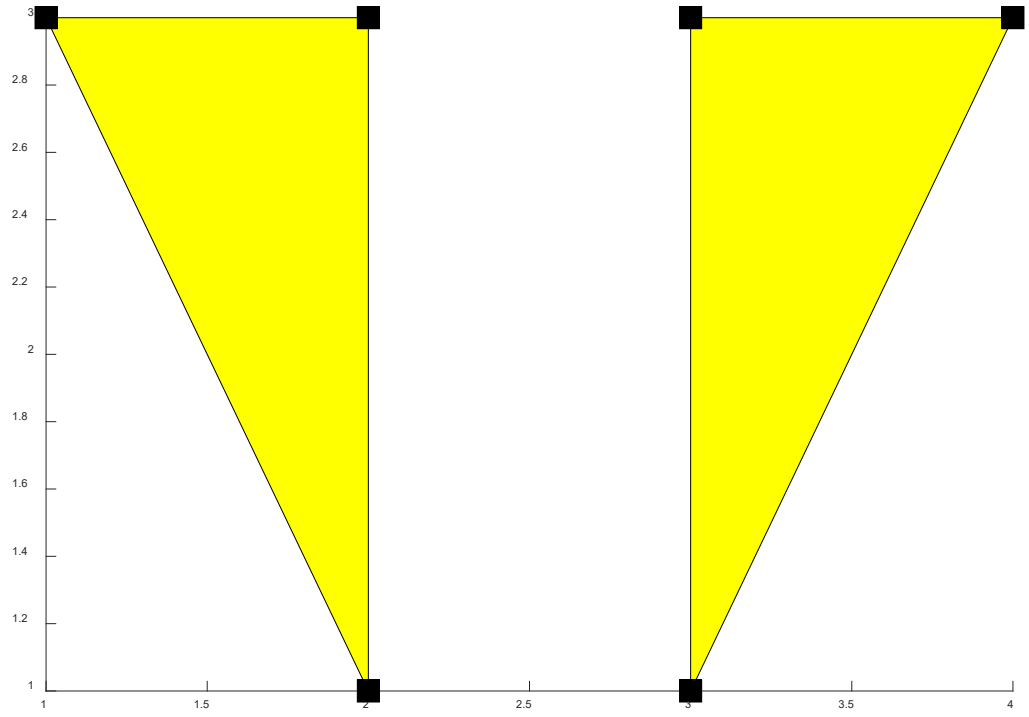
Lines of code from 10 to 22



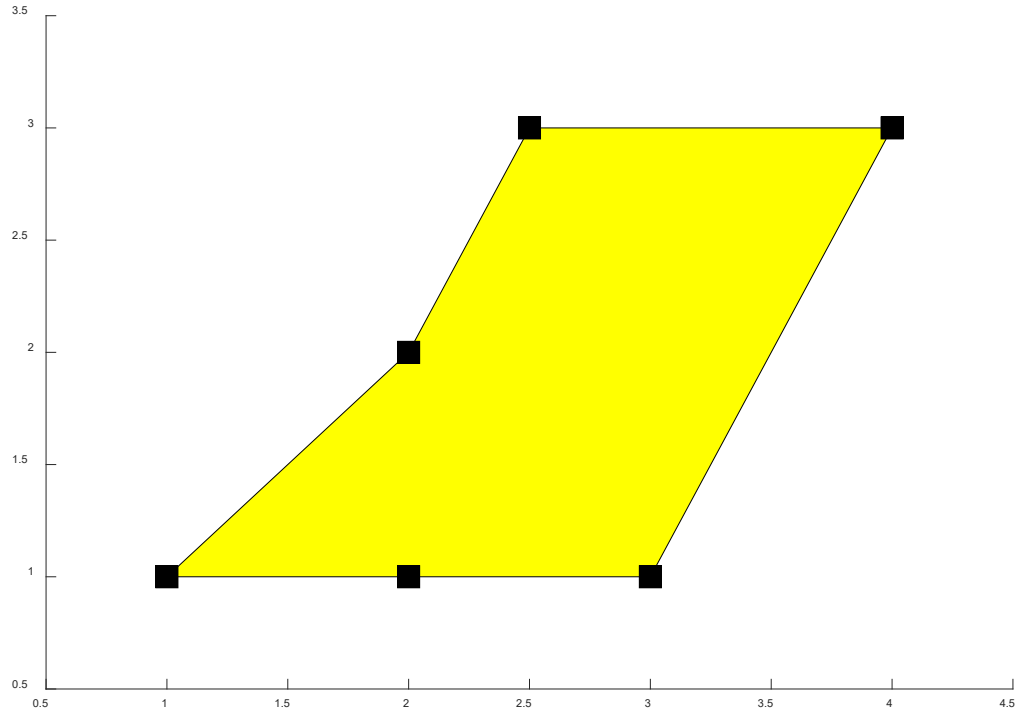

Lines of code from 10 to 22




Lines of code 23 to 34 drawing patches down the columns

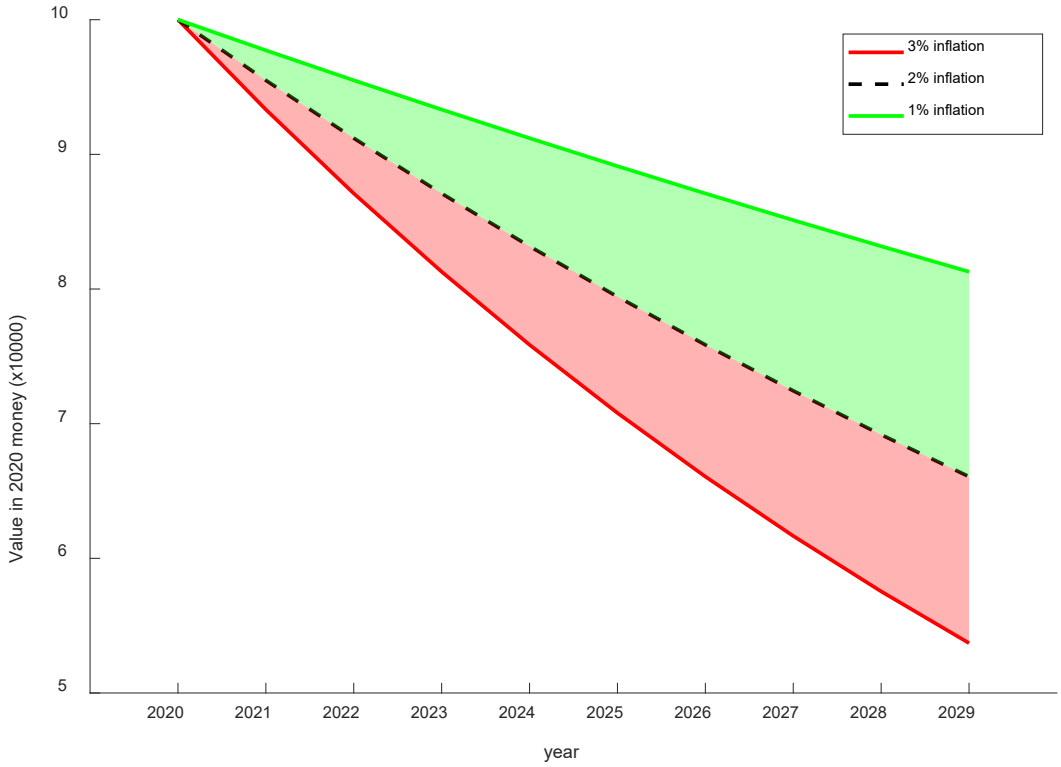



Same lines of code 23 to 34 except drawing patches across the rows




Uncertainty in Future Money

$$v_{y+1} = v_y 10^{-i}$$




Lines of code: 25 to 62

## 44. BLEND PICTURES USING TRANSPARENCY

Load the “car1” and “tape” images. Blend them together using (1) averaging and (2) AlphaData transparency. Then fade from the car to the tape using an alpha (transparency) map.

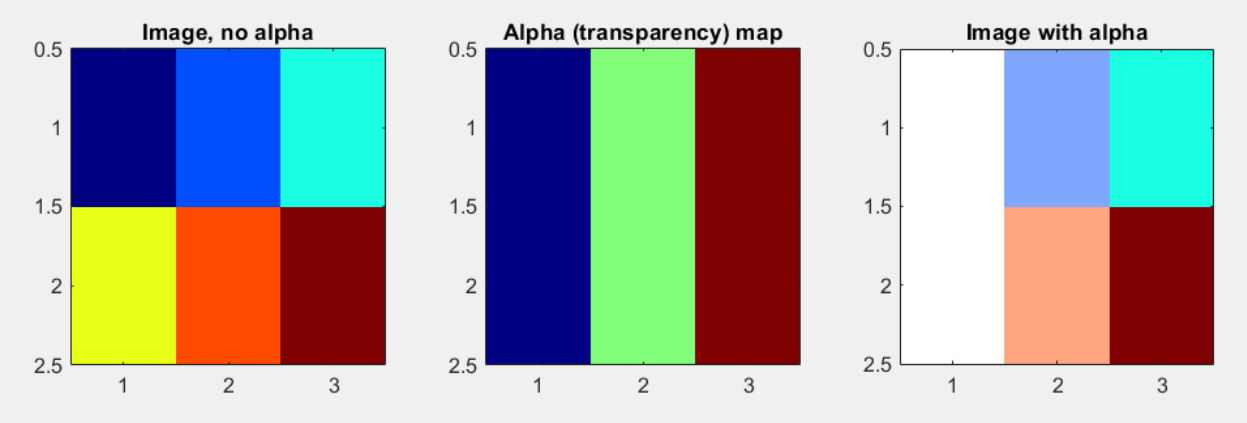
Test spatial patterns of alpha maps. Make an old-school movie-style fade from tape to car.

Skills: imagesc (image scale), alphadata, imread, colormap, imresize (resize using the image toolbox) , meshgrid and sinterp2 (resize without using the image toolbox), logspace (Generate logarithmically spaced vector, like linspace) set

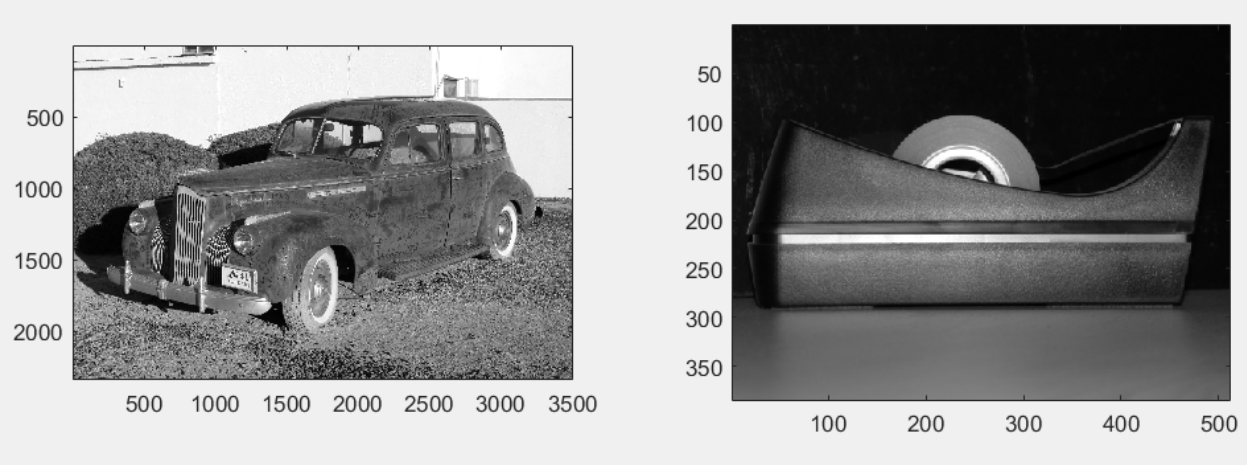
### MATLAB

Code: masterMATLAB_0720_blendImages.m

Data: car1.jpg and tape.png

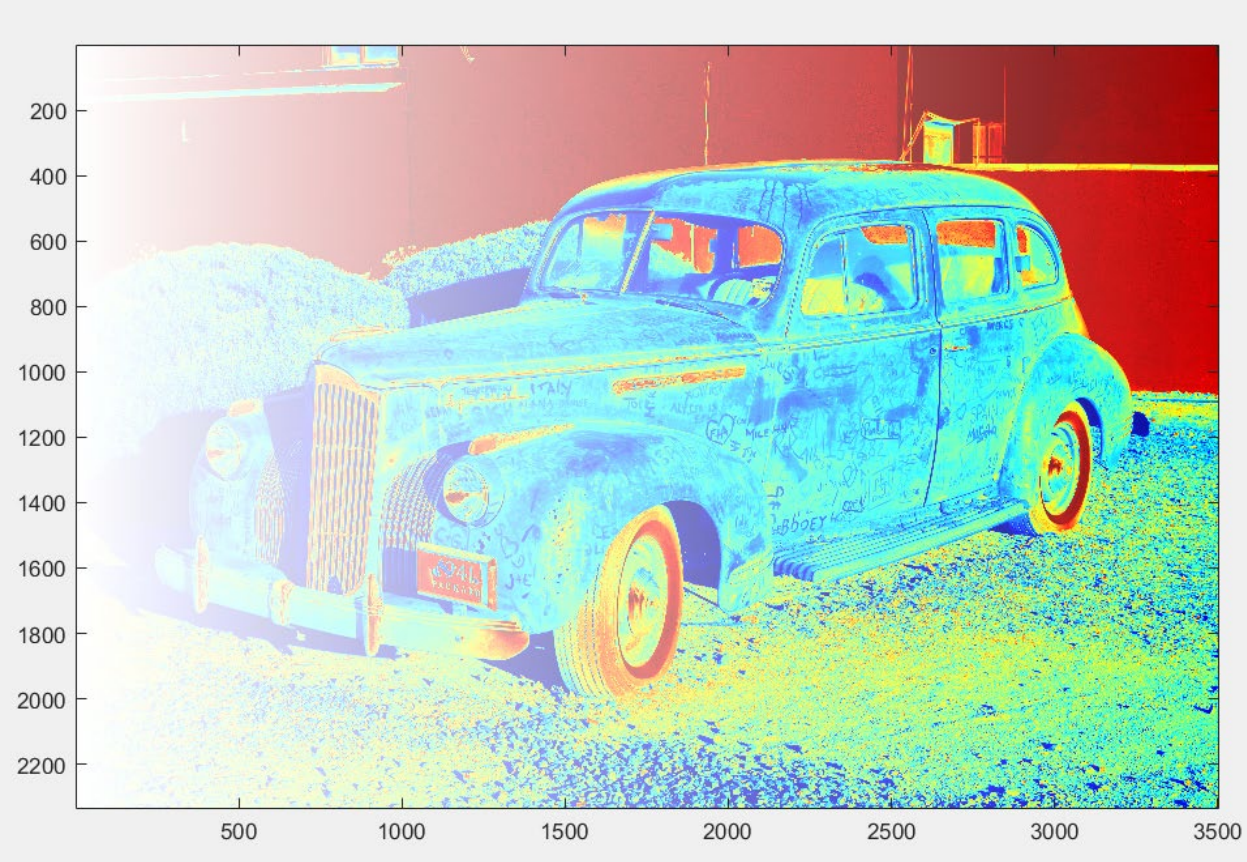



Using lines of code from 7 to 30

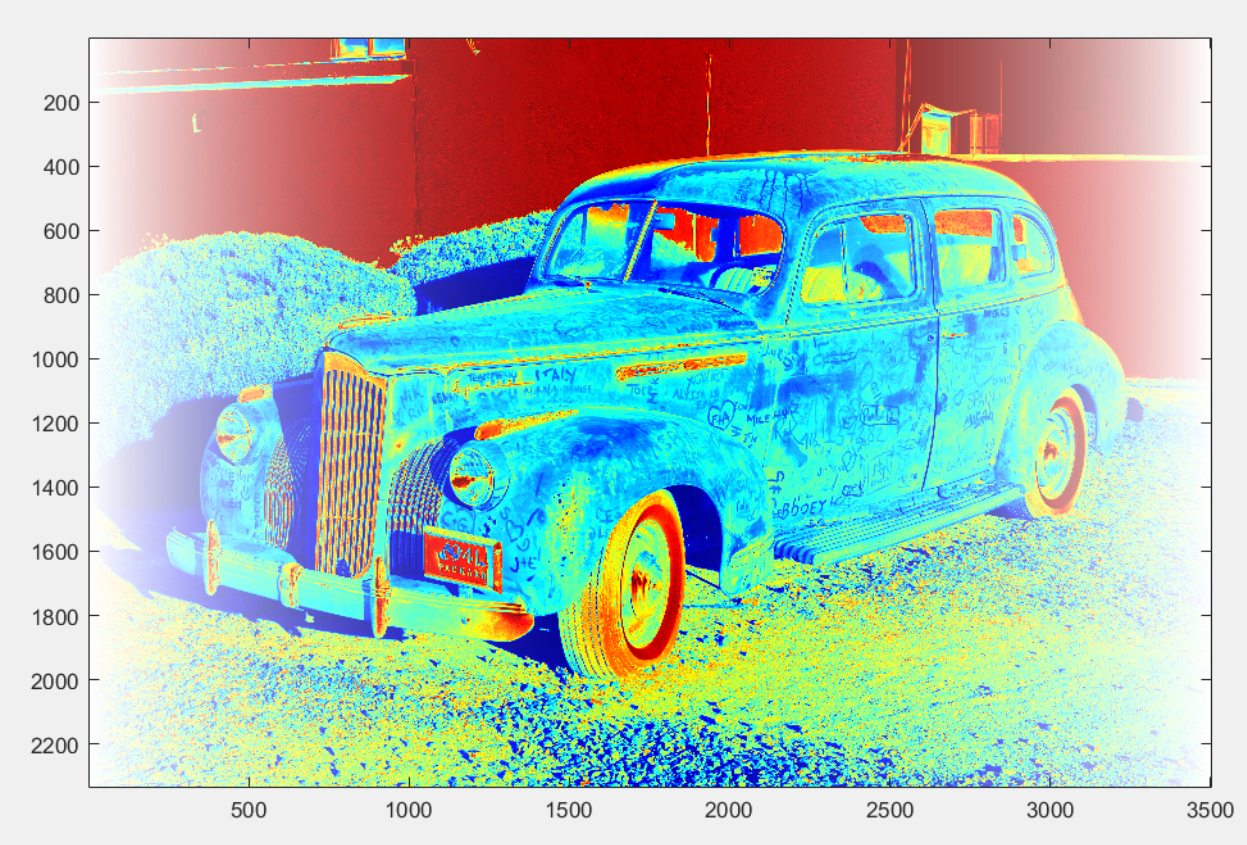



Using lines of code from 31 60 61

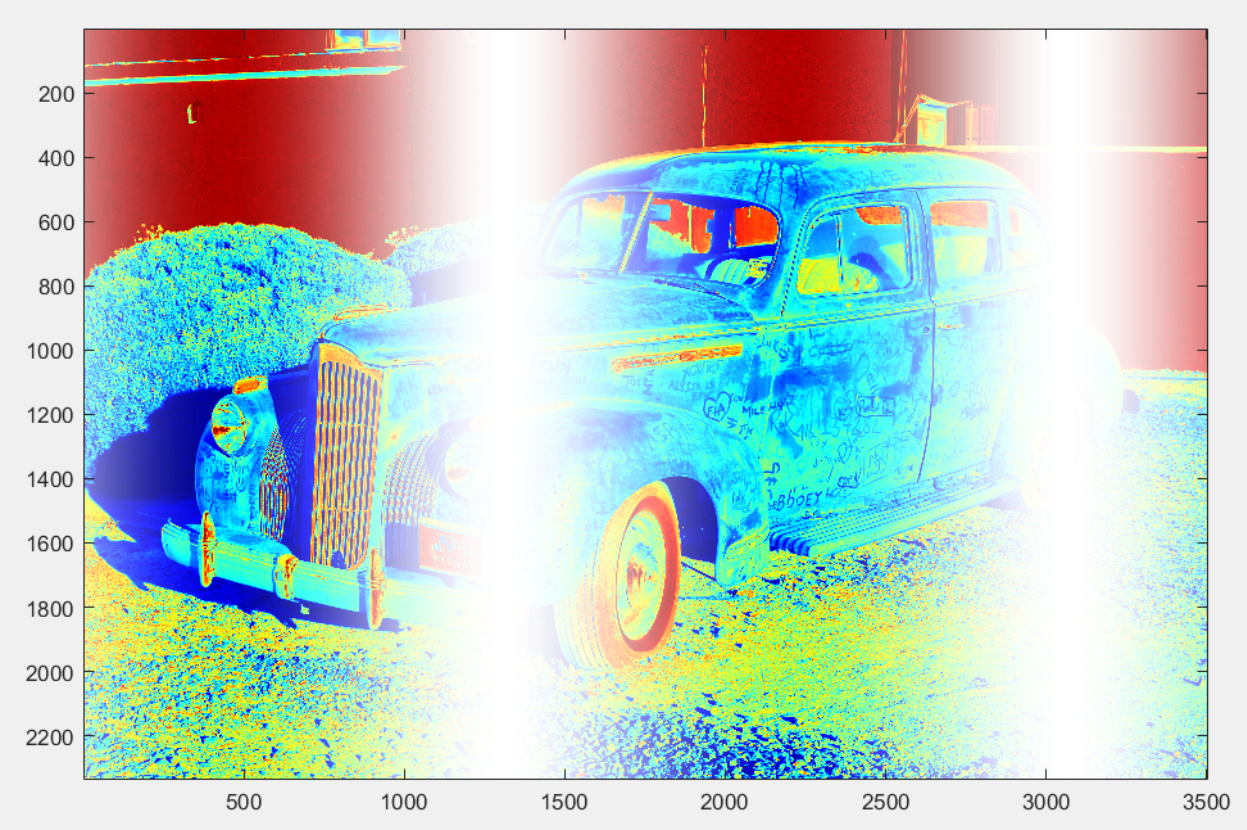

Using lines of code from 62 to 75




Vector of transparency values – linear decrease: Lines of code from 76 to 84




Vector of transparency values – quadradic decrease: Lines of code from 76 to 84




Vector of transparency values – sine wave decrease: Lines of code from 76 to 84


Lines of code from 88 to99 – mix pictures using alpha transparency


Lines of code from 100 to 116 – transparency wipe from one image to another as a movie using sigmoid function

## 45. VERTICALLY STACKING DATA SERIES

Generate correlated multivariate time series, and plot all M channels using vertical stacking.

Stack the time series without using a loop.

Skills: rand, eig (eigen decomposition), detrend, cumsum, bsxfun

To create multivariate time series data:

**X**: MxM matrix (~U[0,1)]

**R**: MxN matrix (~N[0,1])

**V,D** = eig($\mathbf{XX}^T$)

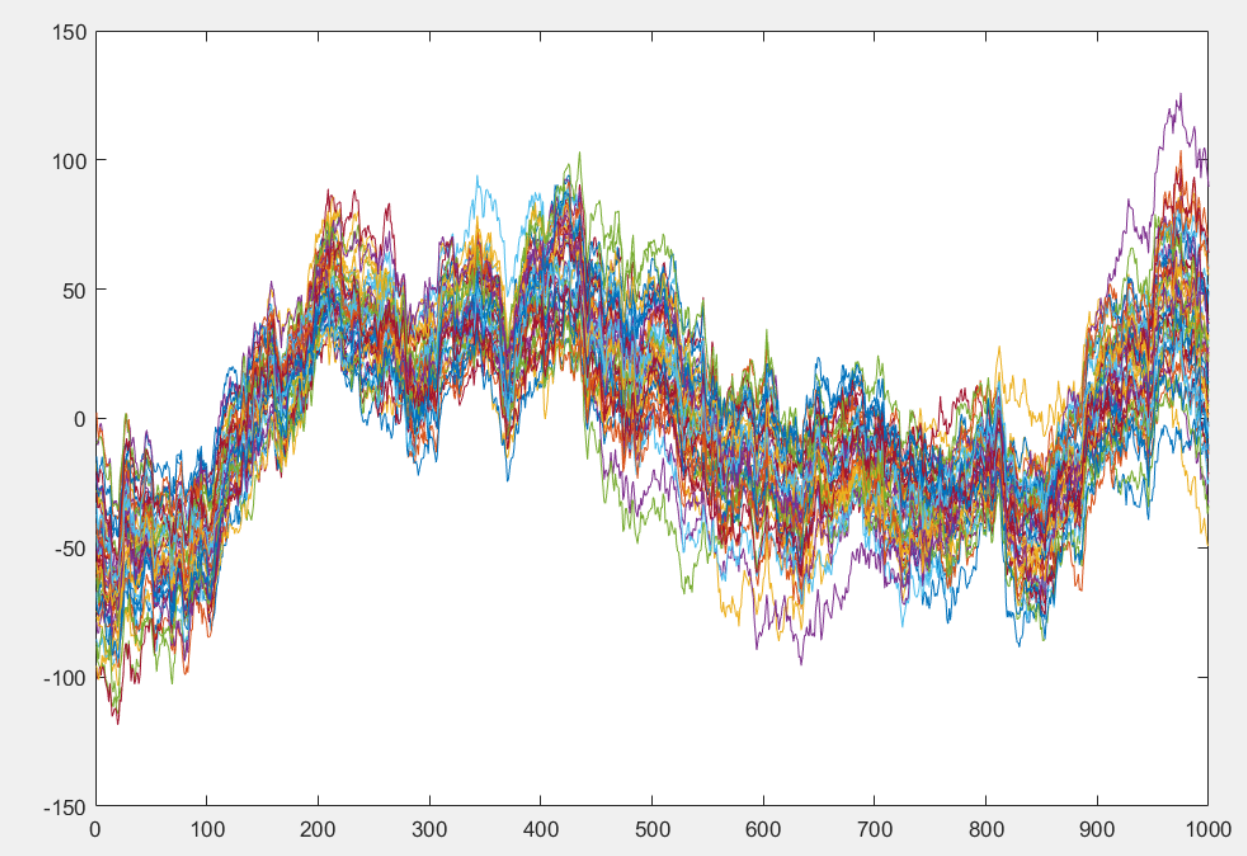
$$\mathbf{Y = VD^{1/2}R}$$

Where:

- X is uniformly distributed between 0 and 1
- R is normally (Gaussian) distributed with a mean of 0 and a STD of 1
- M is the number of channels
- N is the number of time points
- V are eigen vectors from the eigen decomposition
- D are eigen values from the eigen decomposition
- Y is the multivariate time-series data

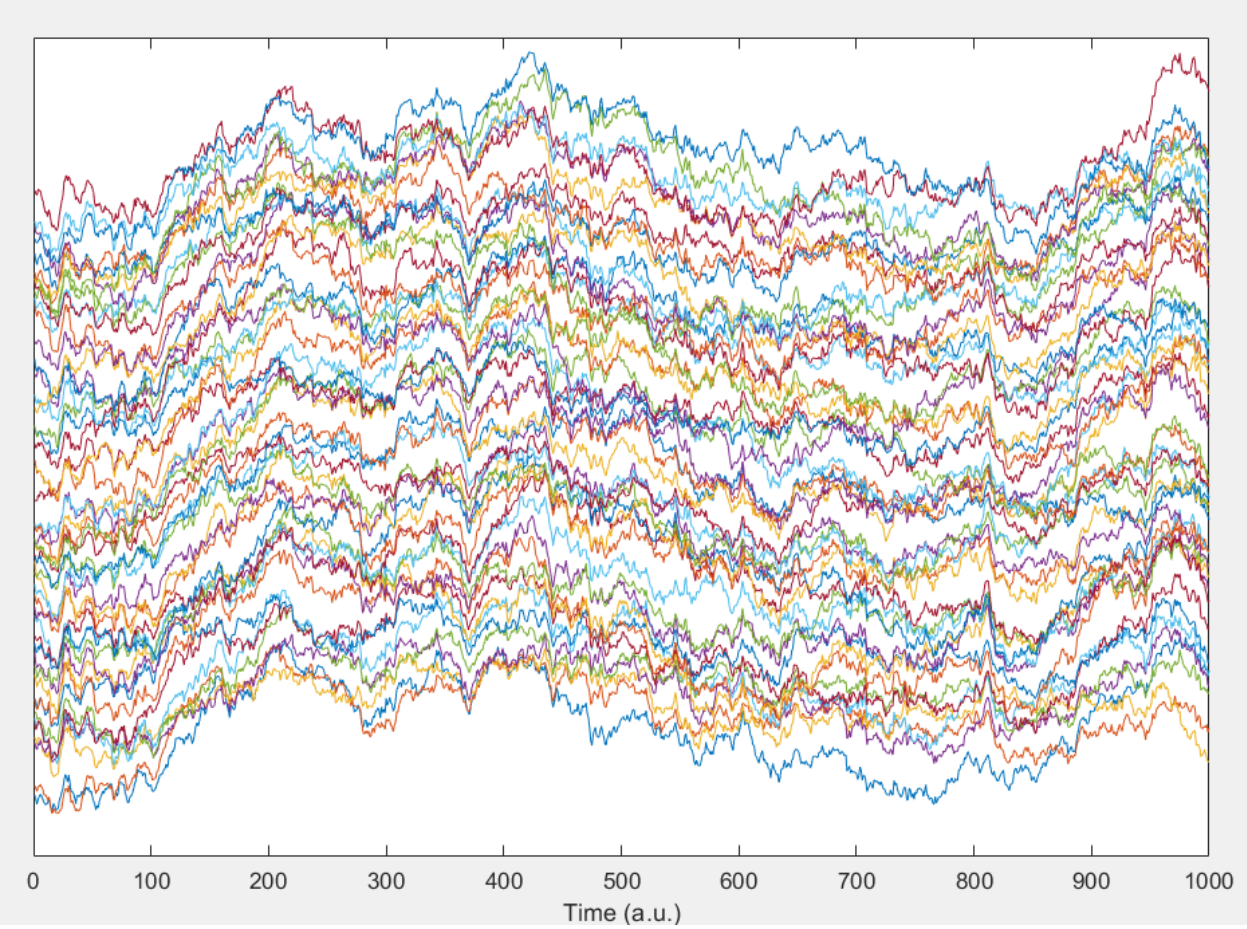
## MATLAB

Code: masterMATLAB_0740_vertStack.m



Lines of code from 7 to 22 – figure 1 – create correlated multivariate time series – Using an Eigen Decomposition


Lines of code from 23 to 29 – Figure 2 – Vertically stack data with 10 unit y offset per trace




Lines of code from 330 to 36 – Figure 3 (same as Figure 1 except…) no for -loop used and y-axis values blanked out

## 46. DISTANCE MATRIX FROM GENERATED POINTS

Create an arbitrary 2D spatial dataset using *ginput* and then compute and display the Euclidean distance matrix.

Compute all-to-all distance without any loops

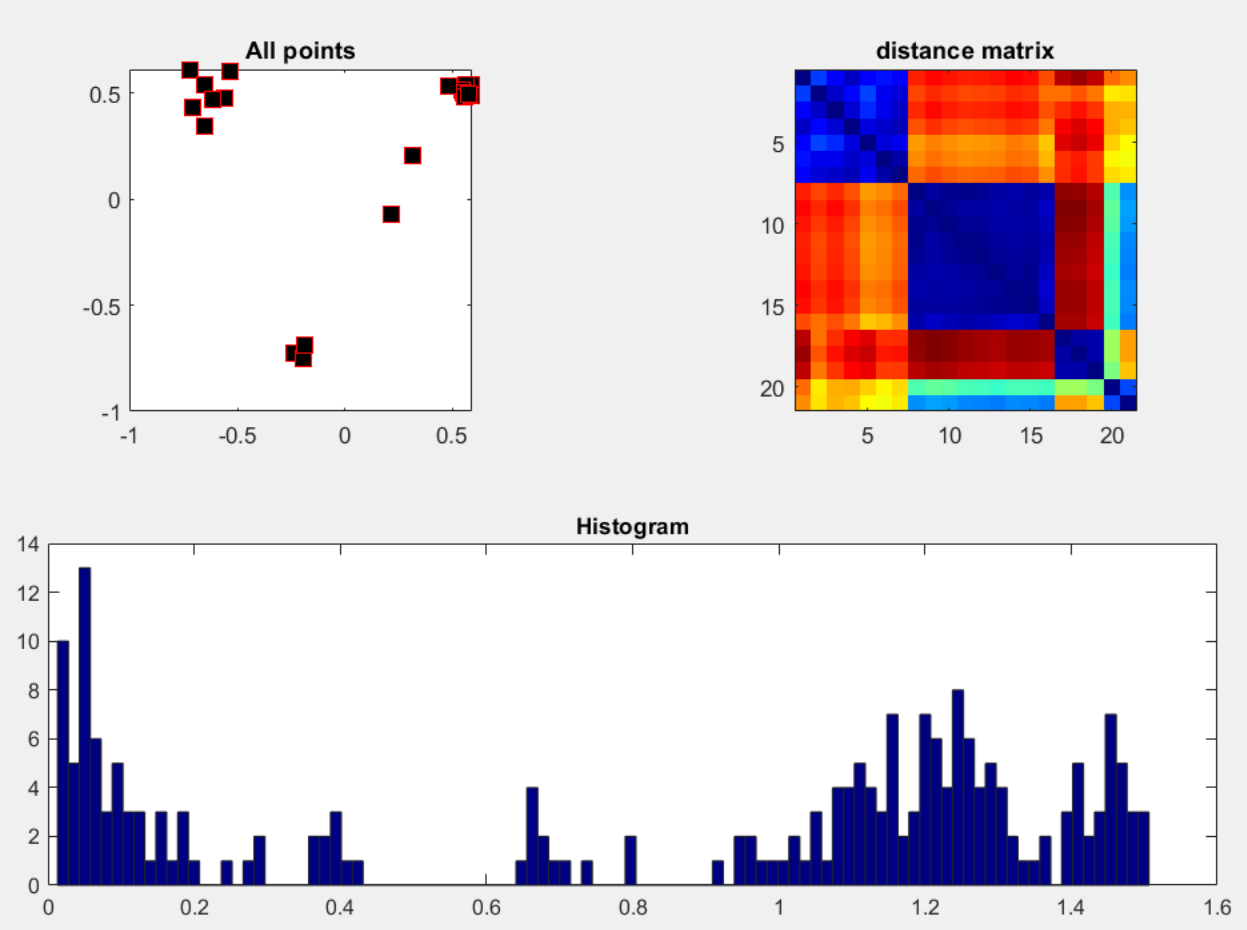
Skills: deal, ginput, set, strcomp, toggle, while, sqrt, bsxfun, triu (Upper triangular part of matrix), hist, nonzeros (convert matrix into a vector of nonzero values)

## Distance between points:

$$d_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

### MATLAB

Code: masterMATLAB_0760_distMat.m




```
>> masterMATLAB_0760_distMat
Difference between loop and one-line distance calculations: 0
>>
```

## 47. GABOR PATCH MARGINAL HISTOGRAMS

Create and plot the absolute value of a Gabor patch (combination of a sine wave and a Gaussian).

Draw the marginal histograms in separate axes to the right and above the Gabor image.

Skills: ndgrid, sin, cos, exp, axes, set, imagesc

Note: Hadamard multiplication is built into certain programming languages under various names. In MATLAB, GNU Octave, GAUSS and HP Prime, it is known as array multiplication, or in Julia broadcast multiplication, with the symbol .* or ⊙.

## Gabor Patch:

$$X_p = X\cos(\phi) + Y\sin(\phi)$$

$$S = \sin(2\pi f X_p)$$

$$G = e^{-((X-m_x)^2+(Y-m_y)^2)/2s^2}$$

$$P = |S \odot G|$$

Where:

Xp: is a matrix of time points (instead of a vector of time points)

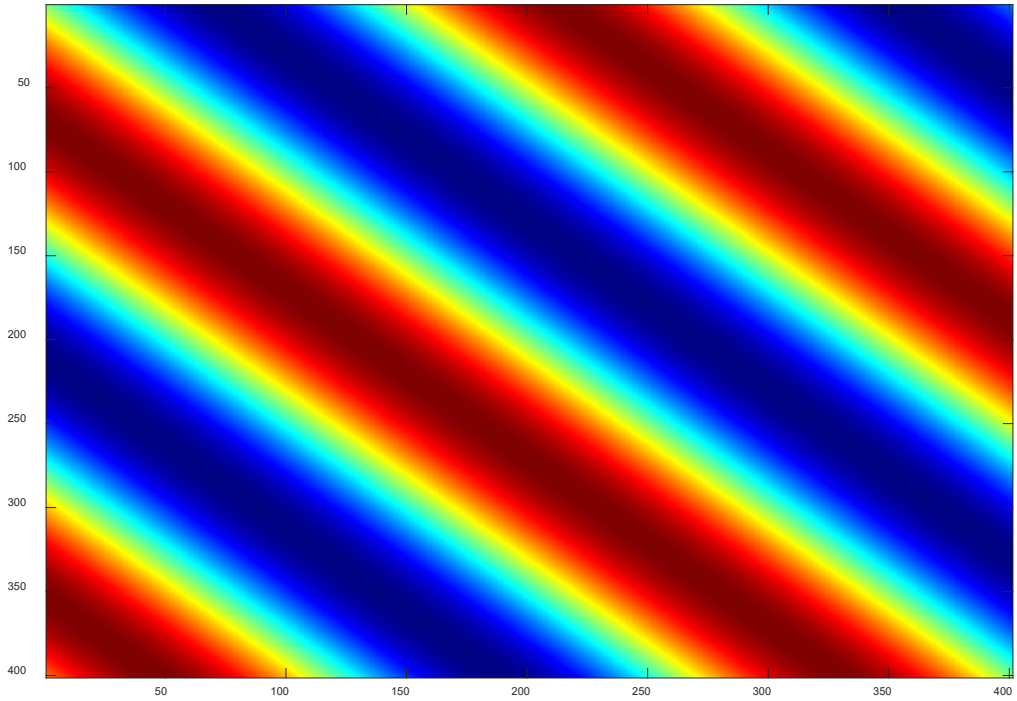
S: is a 2D sine wave

G: is a 2D Gaussian

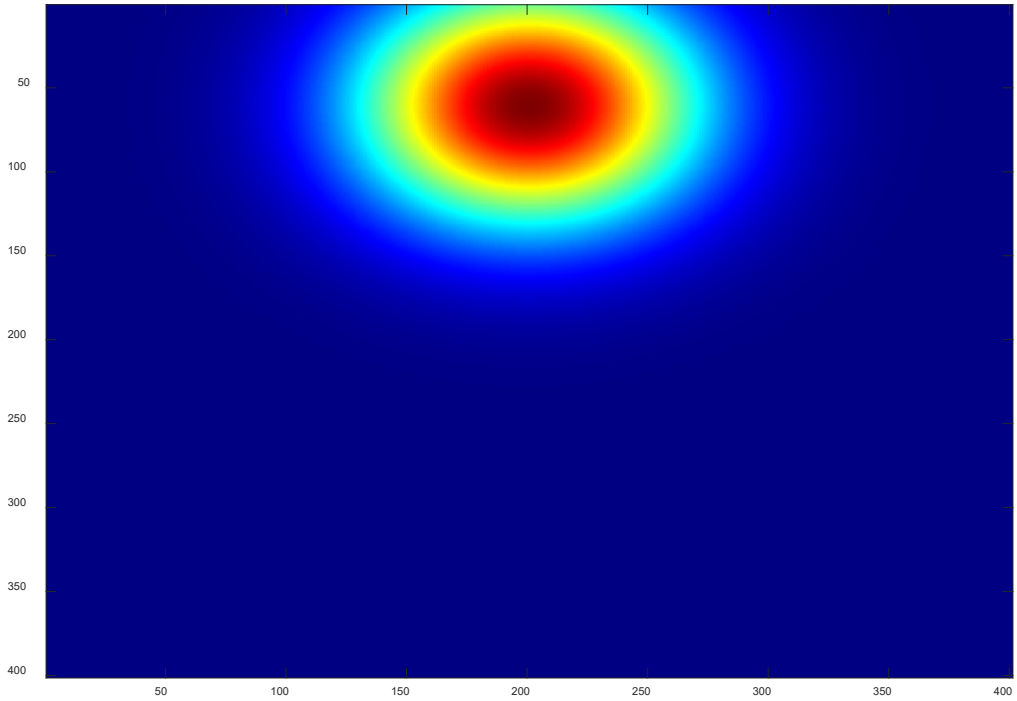
P: is the absolute value of the patch

### MATLAB

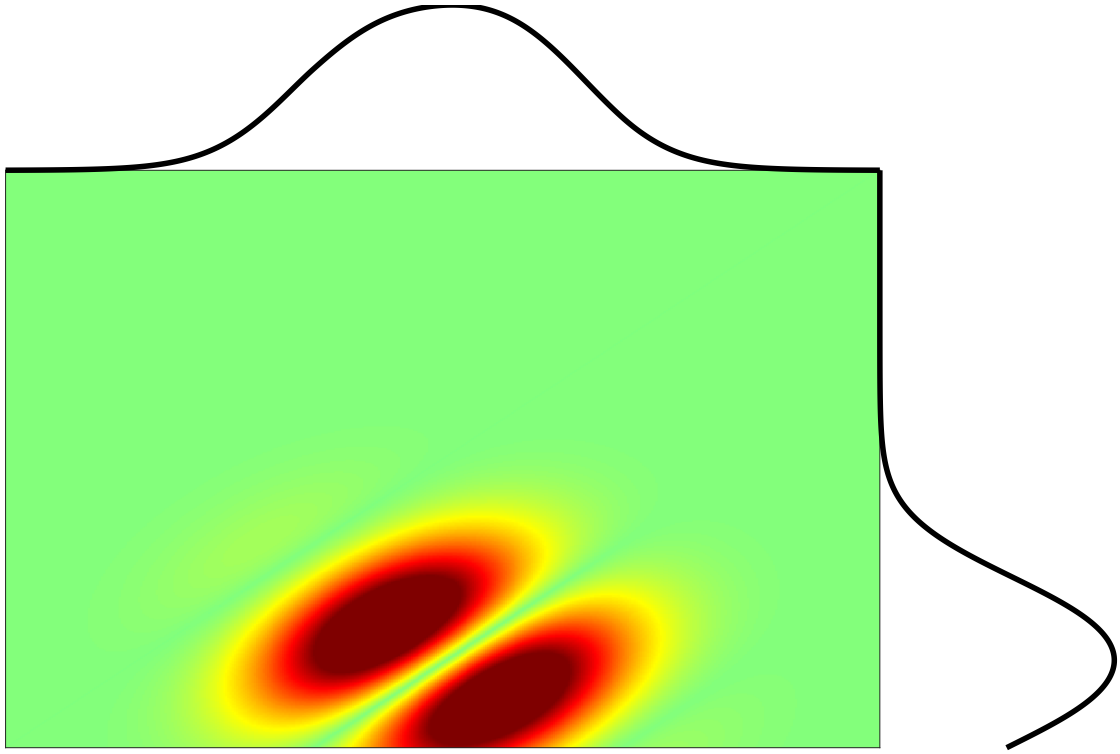
Code: masterMATLAB_0780_GaborHist.m




Imagesc(sine2d)




Imagesc(gaus2d)



## SECTION 9: 3D PLOTTING

### 49. SPHERE IN A CUBE

Use plot3 to draw the 12 edges of a unit cube, then draw a sphere using dots inside that cube.

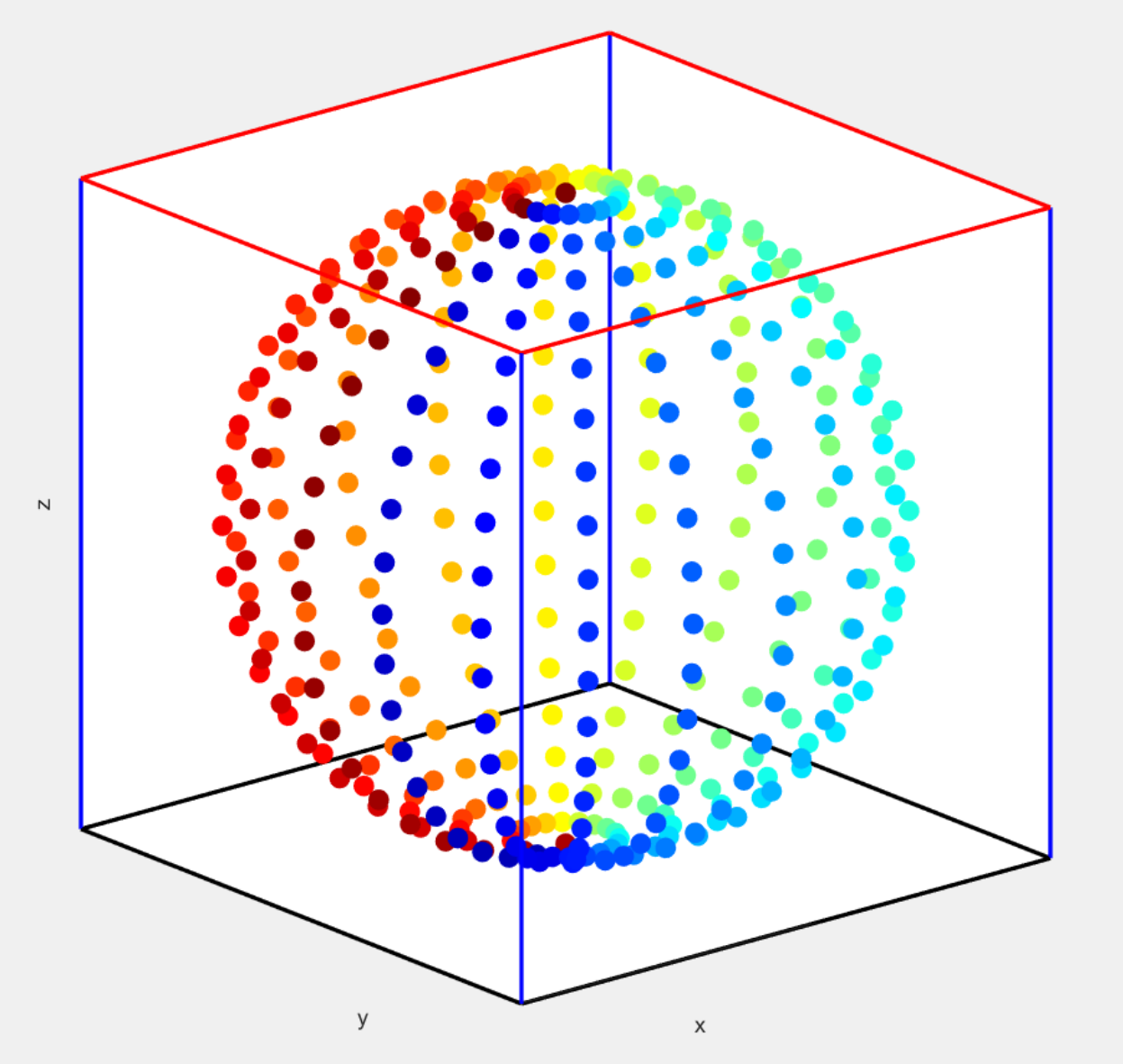
Use scatter3 to make a colorful disco ball!

Skills: plot3, scatter3, sphere

### MATLAB

Code: MasterMATLAB_0800_sphereCube.m

Command line: rotate3d on (allows user to rotate a 3d plot inside figure)




## 50. COLORFUL CUBE (A.K.A. THE HAPPY BORG SHIP)

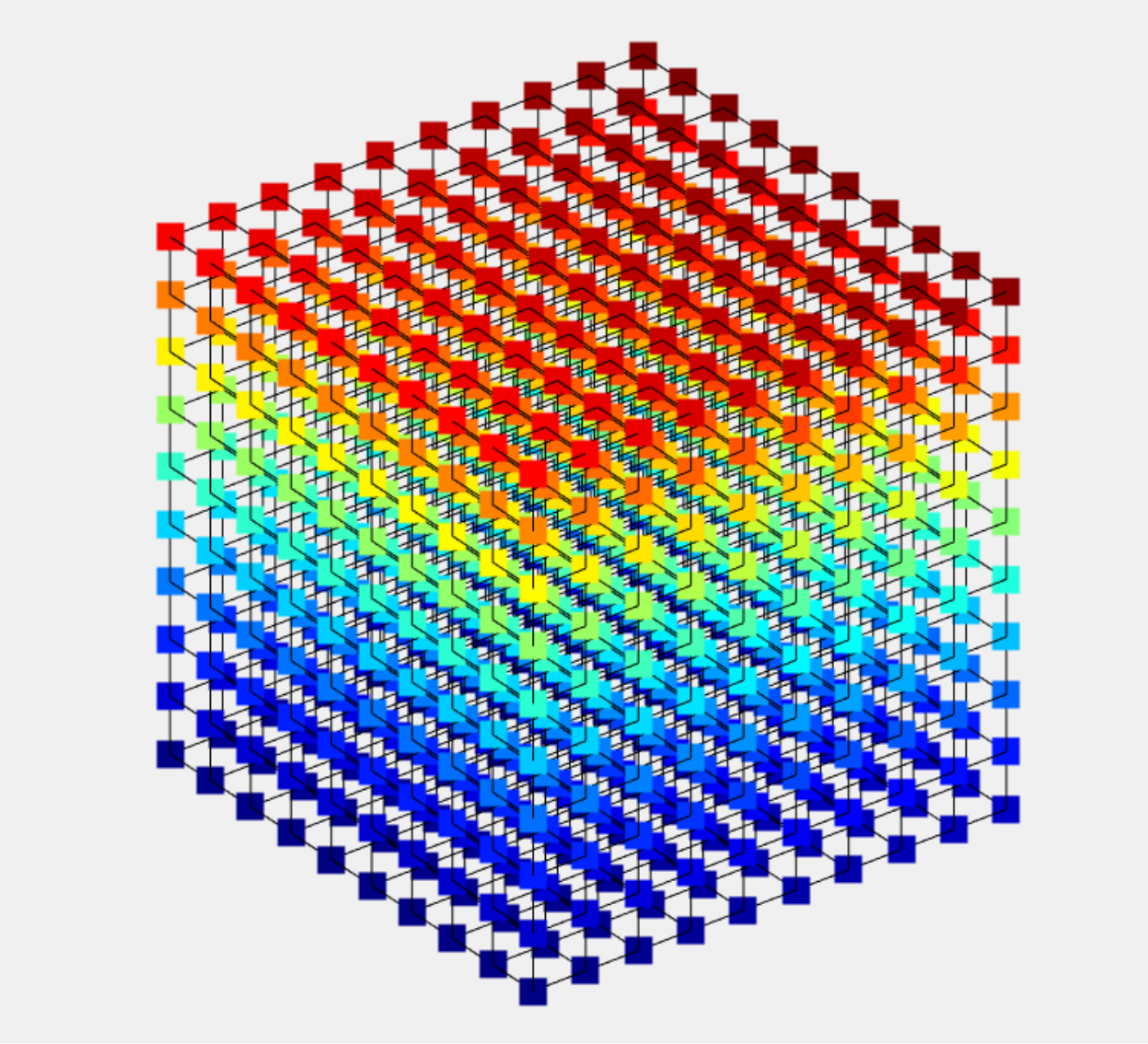
Use plot3 to draw lines connecting NxNxN nodes in a cube, and use scatter3 to draw happy-colored squares at each node.

Use logarithmic color scaling

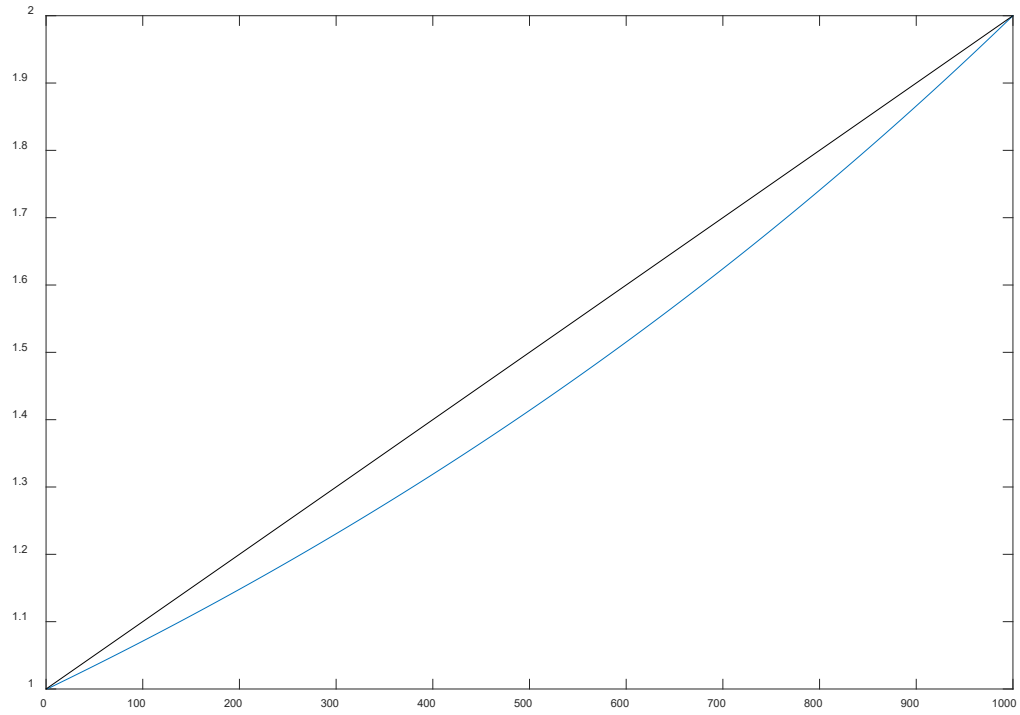
Skills: plot3, statter3, meshgrid

MATLAB

Code: MasterMATLAB_0820_colorfulCube.m



Lines of code from 8 to 29




Lines of code from 31 to 35

## 51. EXPANDING WAVELETS WITH SURFACES

Create a surface plot showing a family of Morlet wavelets.

Set the color scaling according to wavelet frequency instead of the default map height.

Skills: meshgrid, shading, surf

## Equation:

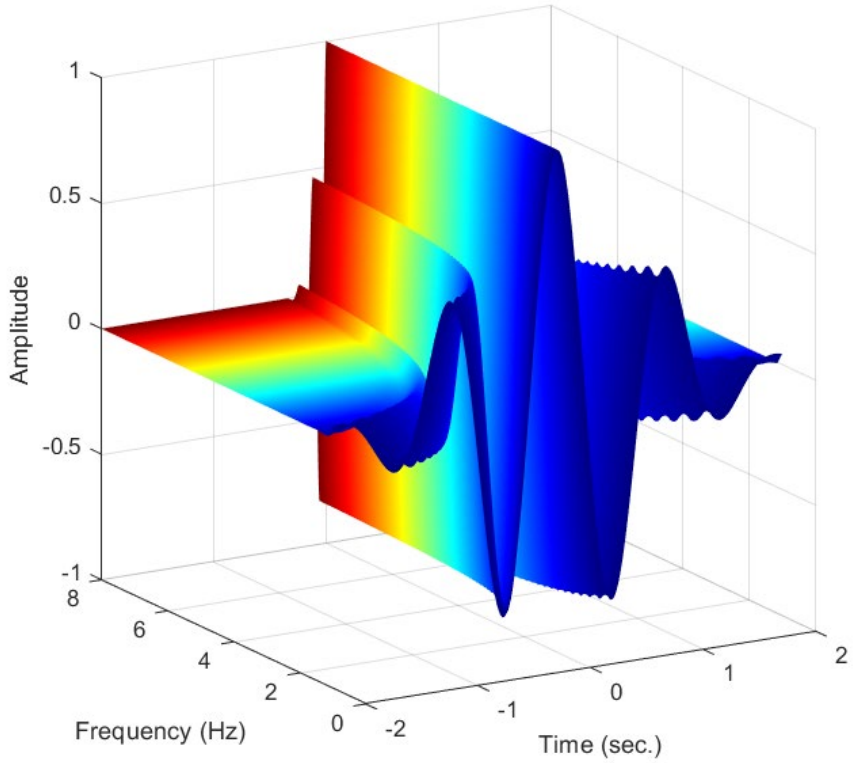
$$w_f = \cos(2\pi f t)e^{-t^2/2s^2}$$

$$s = \frac{5}{2\pi f}$$

$$f: [1,8]\ Hz$$

## MATLAB

Code: MasterMATLAB_0840_3Dwavelets.m




## 52. TEXTURED GAUSSIAN SURFACES

Create a surface plot showing a 2D Gaussian with random texture map instead of the default height. Then map a picture (a hand) onto the surface.

Use an anisotropic Gaussian (different widths on the two dimensions).

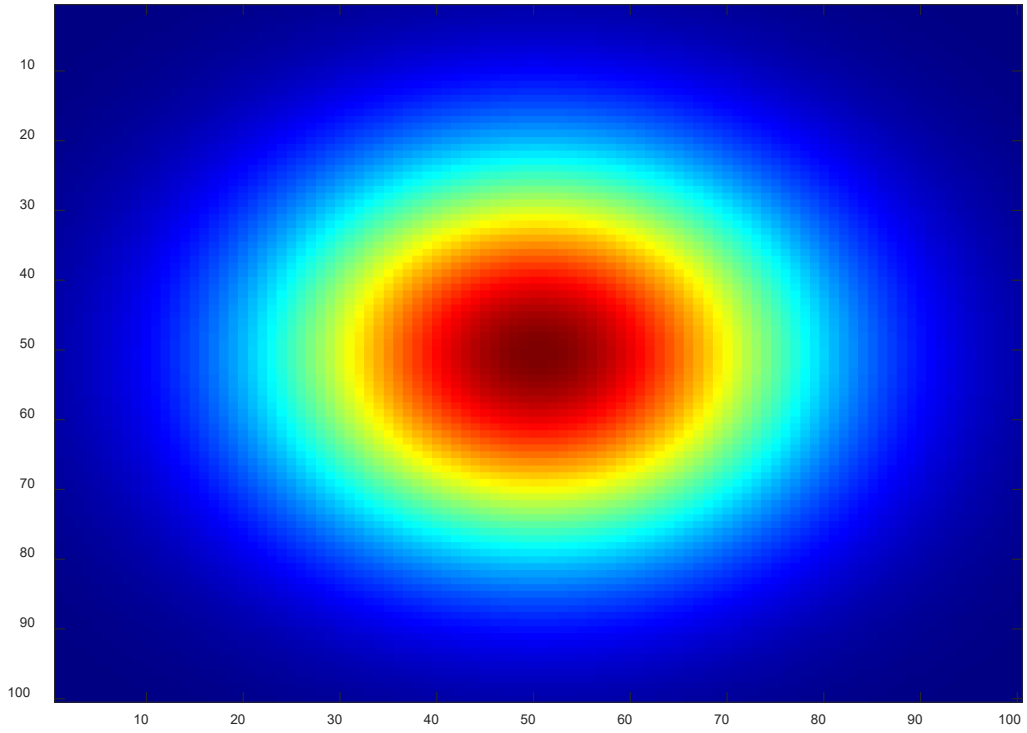
Skills: linspace, meshgrid, surf, rotate3d, imread

Equation:

$$e^{-(X^2+Y^2)/2s^2}$$

## MATLAB

Code: MasterMATLAB_0860_GaussianSurf.m




imagesc(gaus2d) – 2D Gaussian

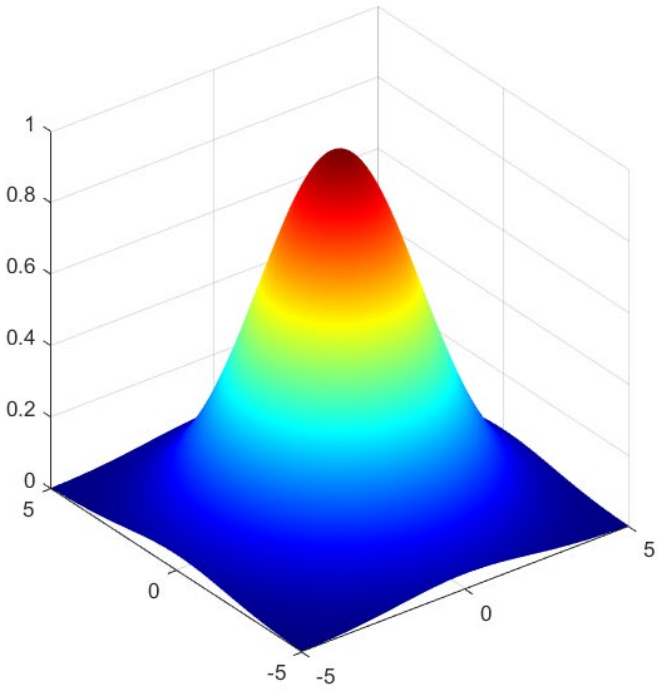



Figure 1

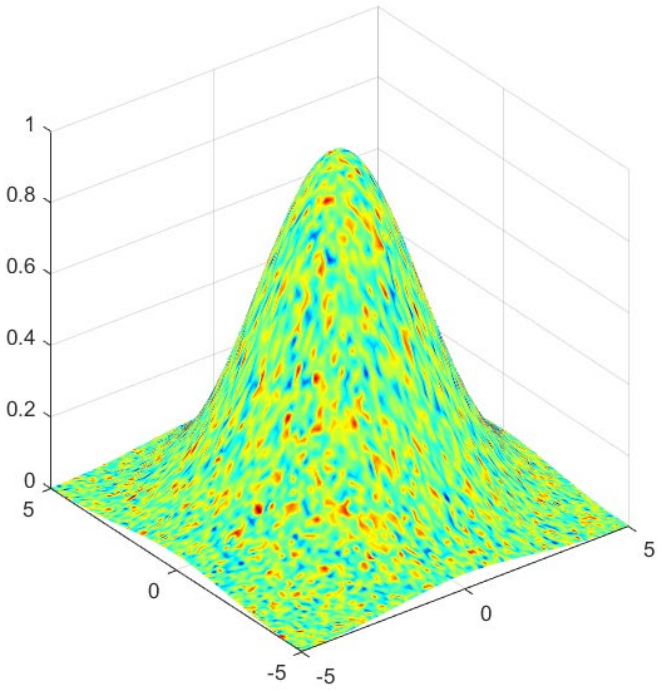



Figure 2

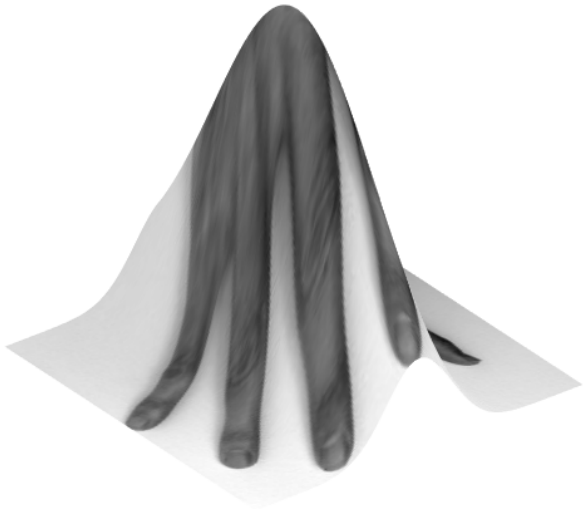


Figure 3

## 53. A BALL IN 3D COLOR SPACE

Think of RGB color space as a 3D cube. Then use dots to draw a sphere of specified location and diameter inside that cube, with the dots colored according to their location in the 3D cube.

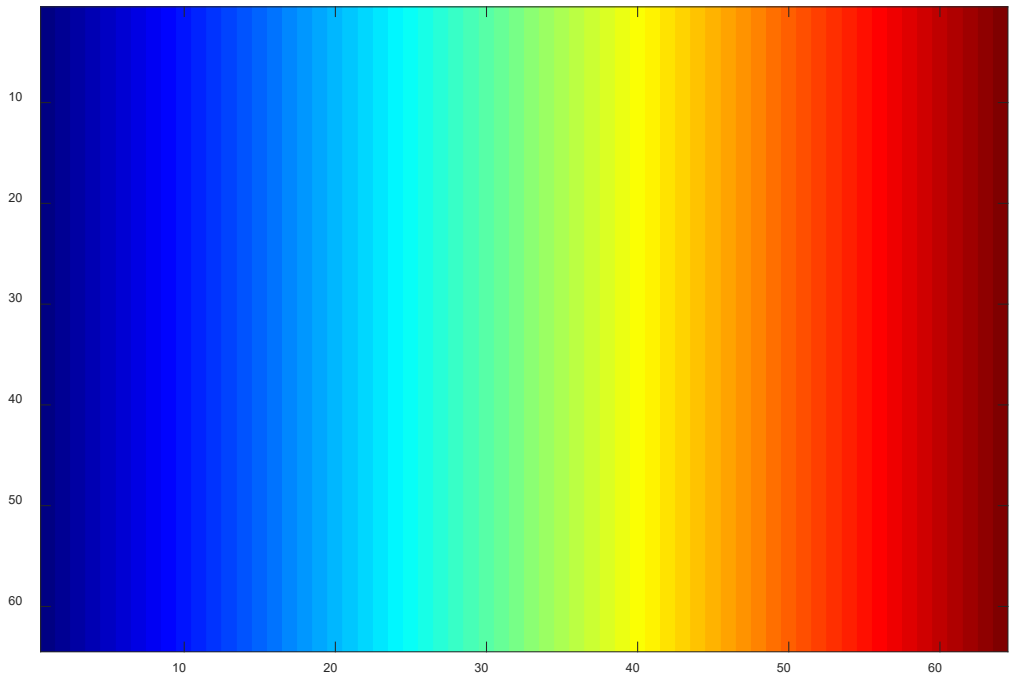
Change the axis labels to fraction of maximum possible color.

Skills: meshgrid, sqrt, ind2sub (index to subscript), scatter3, squeeze (Remove dimensions of length 1)

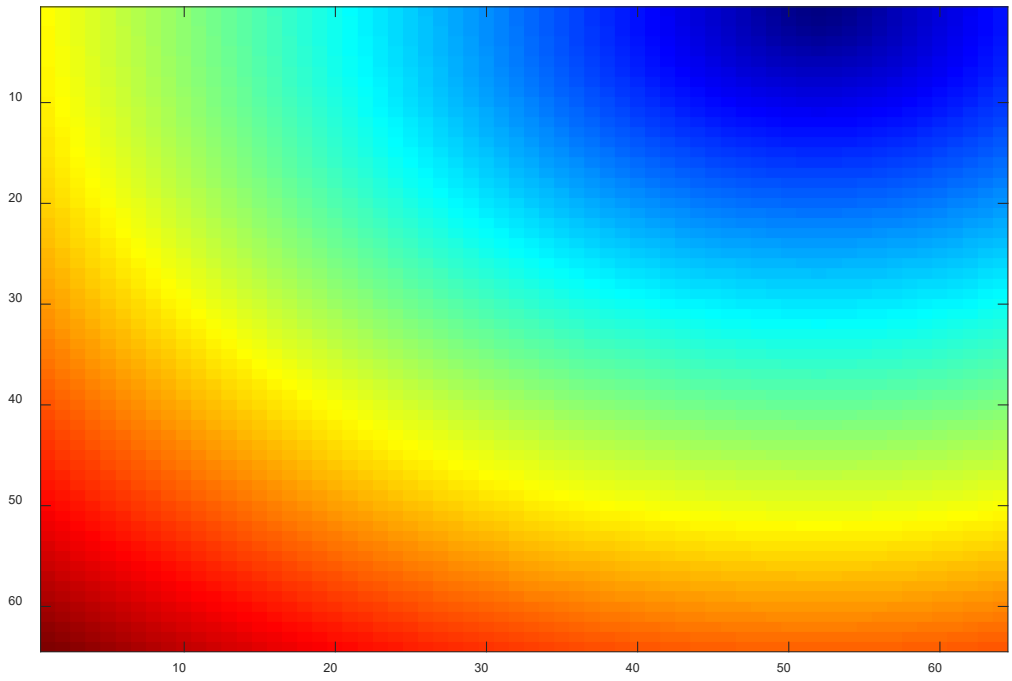
Matrix indexing vs linear indexing

### MATLAB

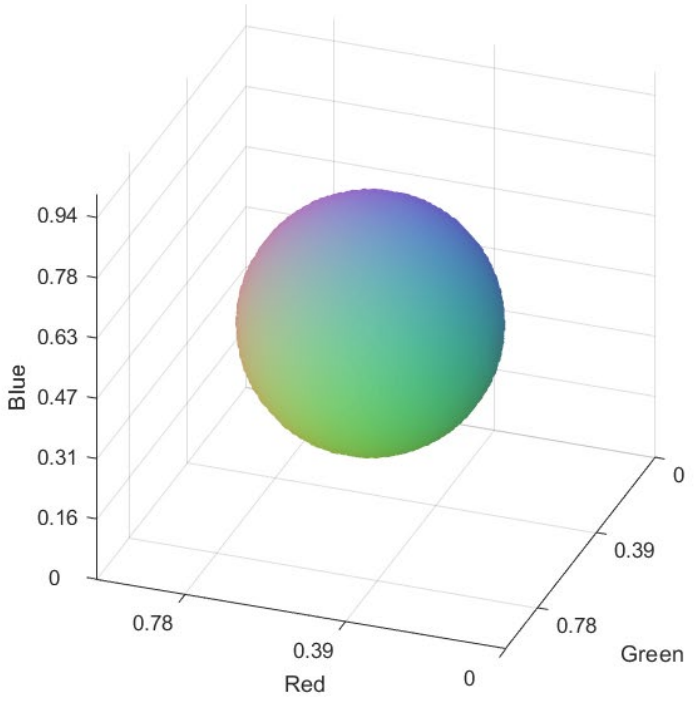
Code: MasterMATLAB_0880_ballInRGB.m




```
skip   = 4;
cent   = round([ .3 .0 .8 ]*255/skip); % center of sphere
radius = round( .3 *255/skip);
% define the indices in the RGB space
x = 0:255/skip;
[R,G,B] = meshgrid(x-cent(1),x-cent(2),x-cent(3));
>> imagesc(squeeze(R(:,:,14)))
```




```
% show those points as a physical sphere with color mapped on
distSph = sqrt( R.^2 + G.^2 + B.^2 ); % Euclidean distance
>> imagesc(squeeze(distSph(:,19,:)))
```




## 54. PLANE IN R3 SPANNED BY TWO VECTORS

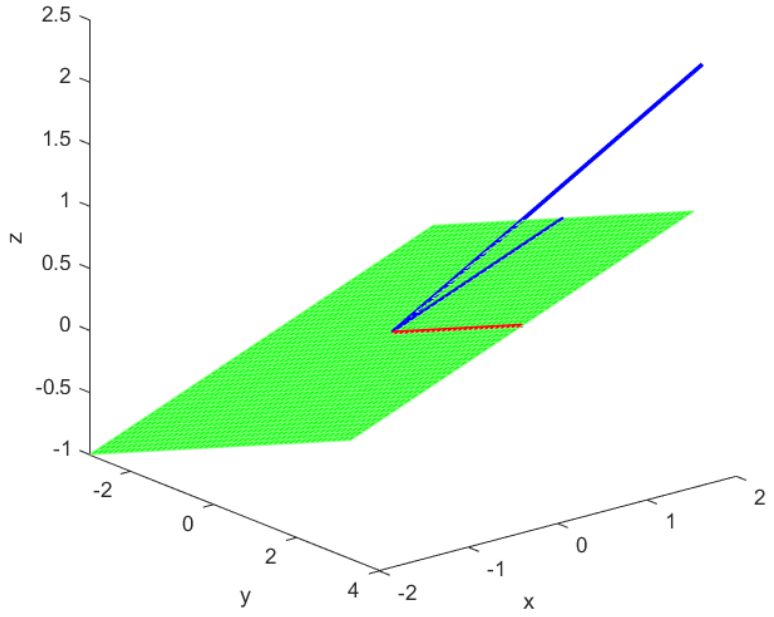
Two lines in a 3D space can form a plane. Define two (independent) vectors, draw them as lines, and draw the plane spanned by those vectors.

Visualize that any arbitrary linear combination of the two vectors is also in the plane.

Skills: ezmesh, plot3, set

### MATLAB

Code: MasterMATLAB_0900_planeR3.m




## 55. COMPLEX SINC SURFACE

Generate a complex 2D sinc function, and show the real part + magnitude as a surface.

Implement the function in two lines without for-loops!

Skills: complex, sin, transpose surf, real, abs

## Equation:

$$f(z) = Re(y) - |y|$$

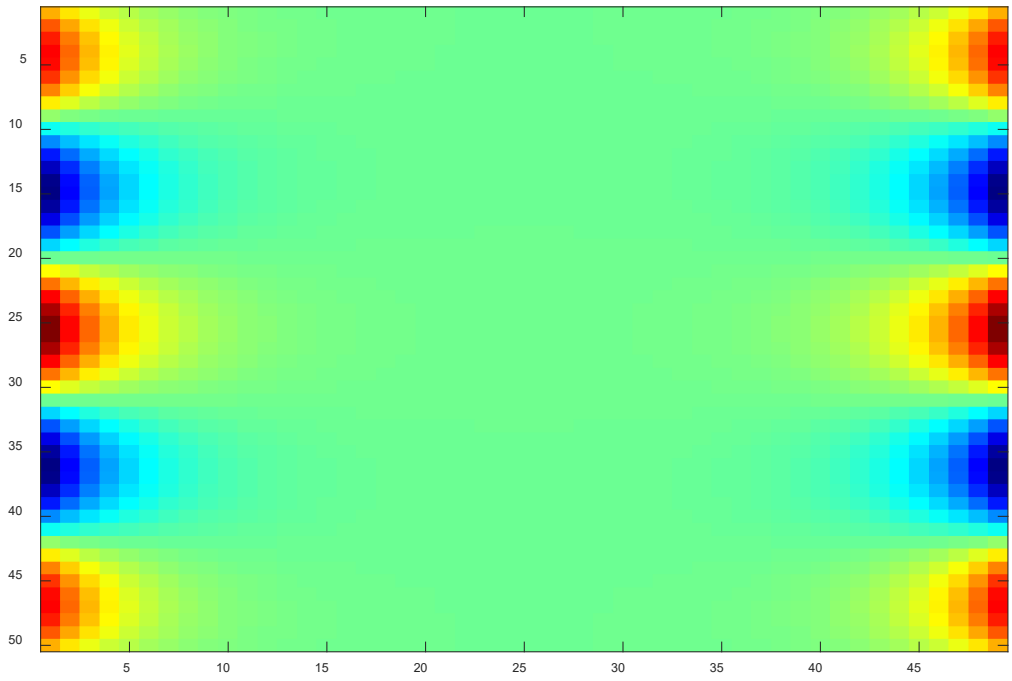
$$y = \frac{\sin(z)}{z}$$

$$Re(z), Im(z): (-8, 8)$$

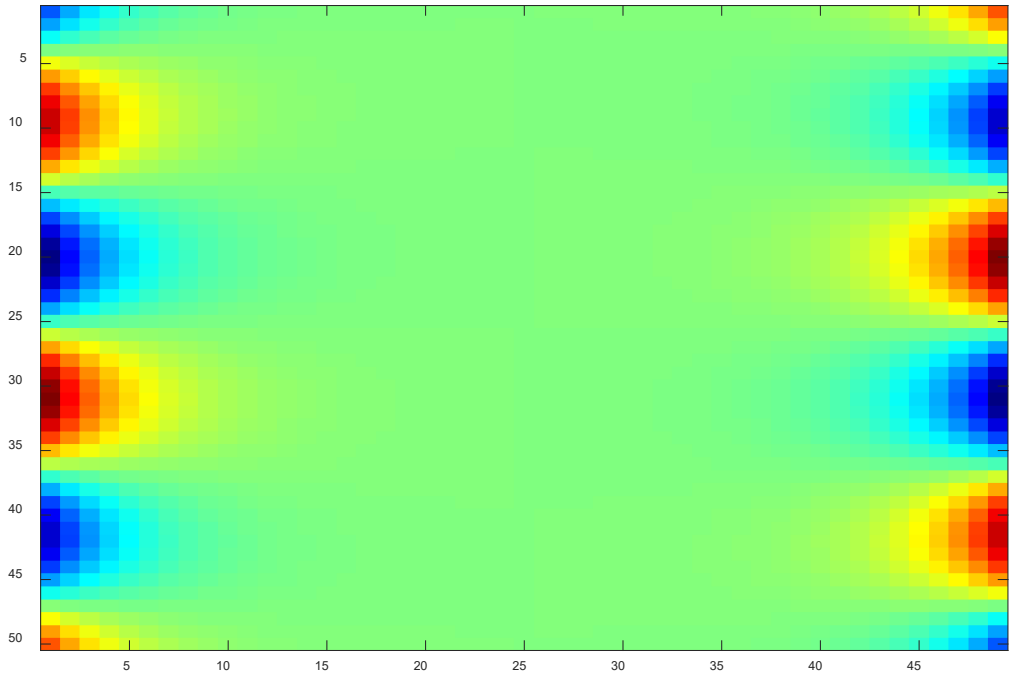
Where y is the sinc function

### MATLAB

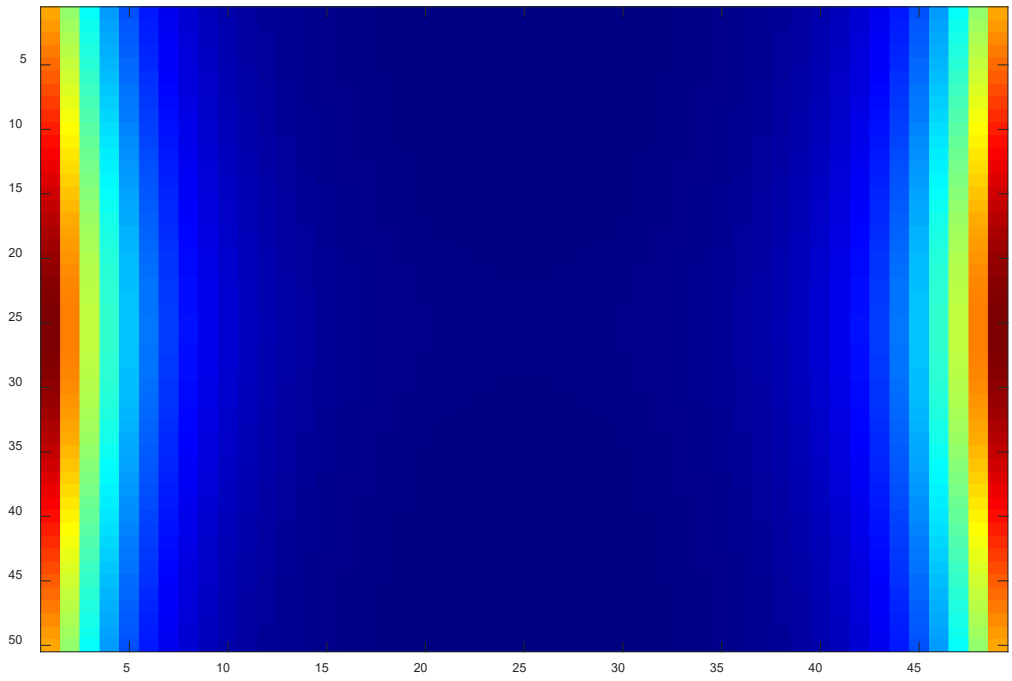
Code: MasterMATLAB_0920_sincSurface.m



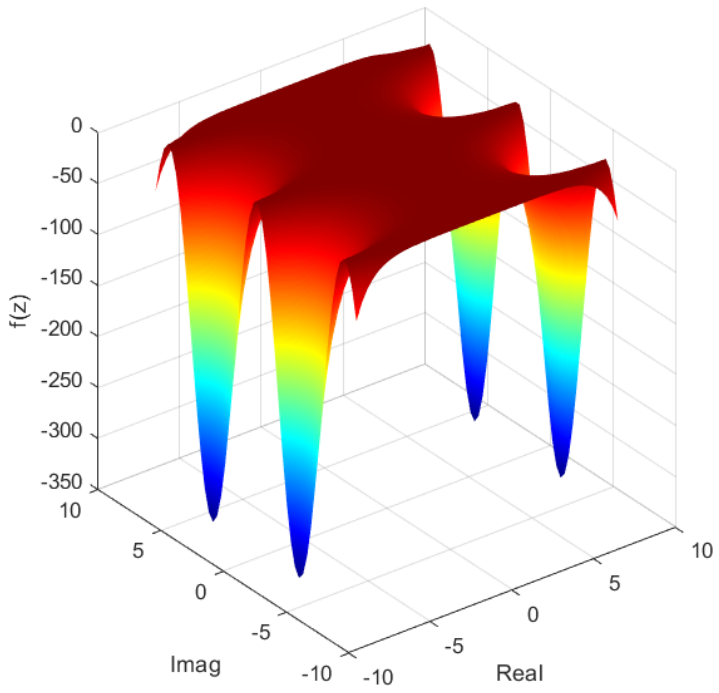

imagesc(real(sincsurf))




imagesc(imag(sincsurf))




imagesc(abs(sincsurf)) % maginitude



## 56. THE PRICKLY GABOR PATCH

Generate a Gabor patch, and show it as a surface with its normal.

Also display the patch as a 3D grid.

Skills: ndgrid, exp, surfnorm, mesh

Gabor patches are used in artificial intelligence, vision processing and vision neuroscience. Neurons in brain implement Gabor filter to convert visual input into functions that make sense of images.

## Gabor equation:

$$X_p = X\cos(\phi) + Y\sin(\phi)$$

$$S = \sin(2\pi f X_p)$$

$$G = e^{-((X-m_x)^2+(Y-m_y)^2)/2s^2}$$

$$P = |S \odot G|$$

Where:

Xp: is a matrix of time points (instead of a vector of time points)

S: is a 2D sine wave

G: is a 2D Gaussian

P: is the absolute value of the patch

### MATLAB

Code: MasterMATLAB_0940_pricklyGabor.m

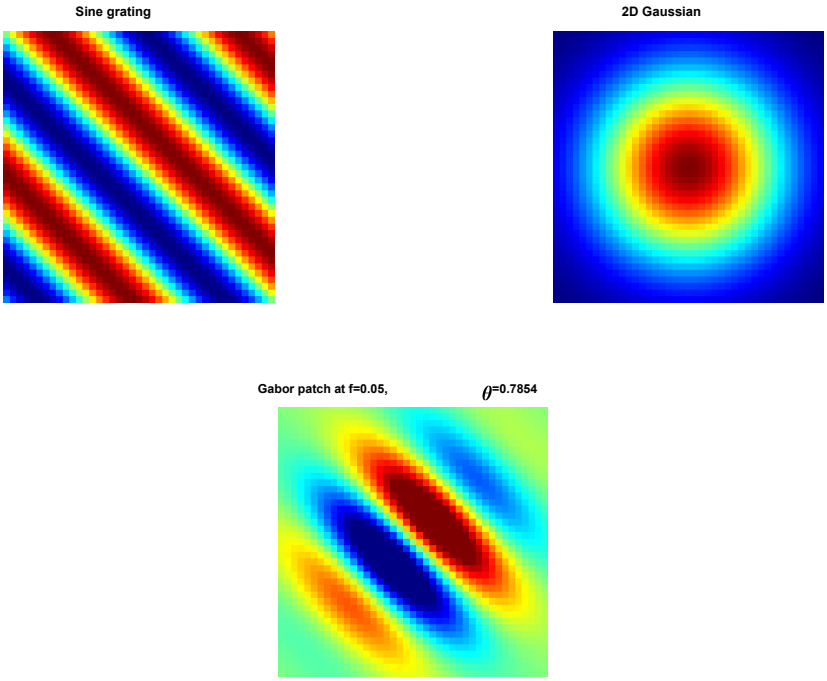



Figure 1

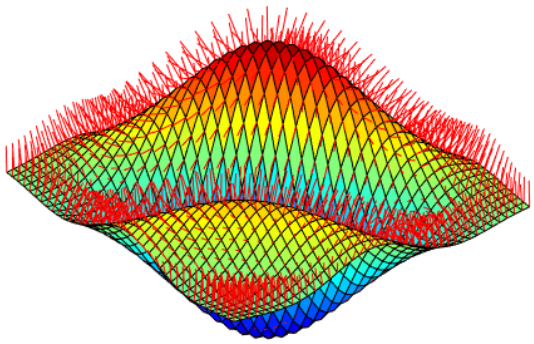


Figure 2

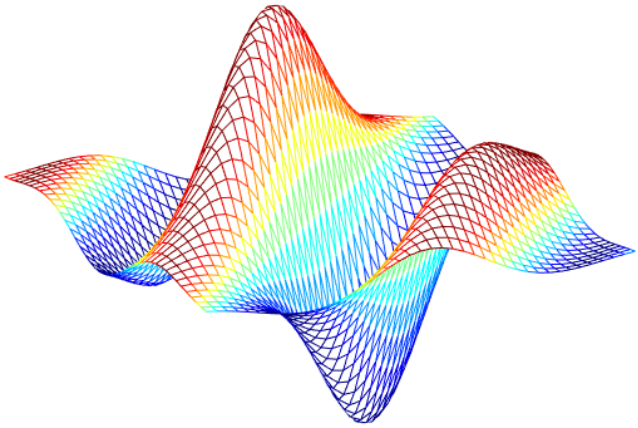


Figure 3

## SECTION 10. SEGMENTATION

### 57. THRESHOLD-BASED TIME SERIES

Generate a random time series with large positive and negative fluctuations. Identify the 10% most extreme points (positive and negative) and draw patches from those points to the y = 0 line.

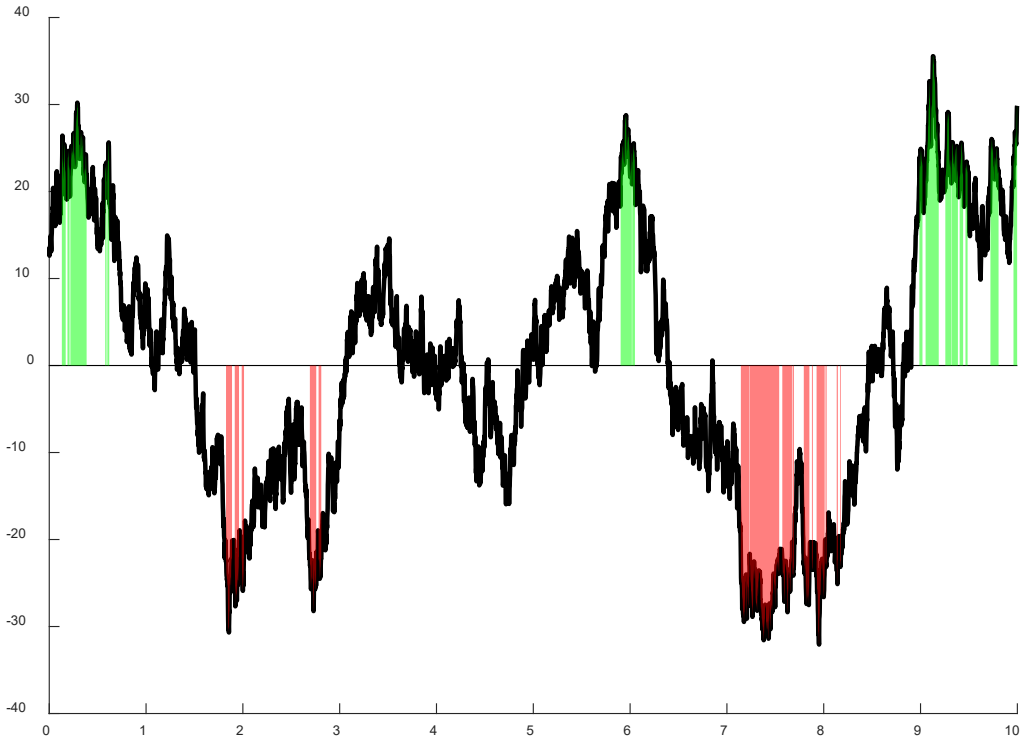
Exclude any extreme segments that have fewer than N=5 consecutive extreme points.

Skills: detrend, eval, cellfun. Patch, bwconncomp (image processing toolbox)

bwconncomp useful for image and time-series segmentation

#### MATLB

Code: MasterMATLAB_0960_threshSegment.m




## 58. DERIVATIVE-BASED TIME SERIES SEGMENTATION

Simulate 1000 days of stock prices, and mark sharp increases or decreases of price using red or green thick lines.

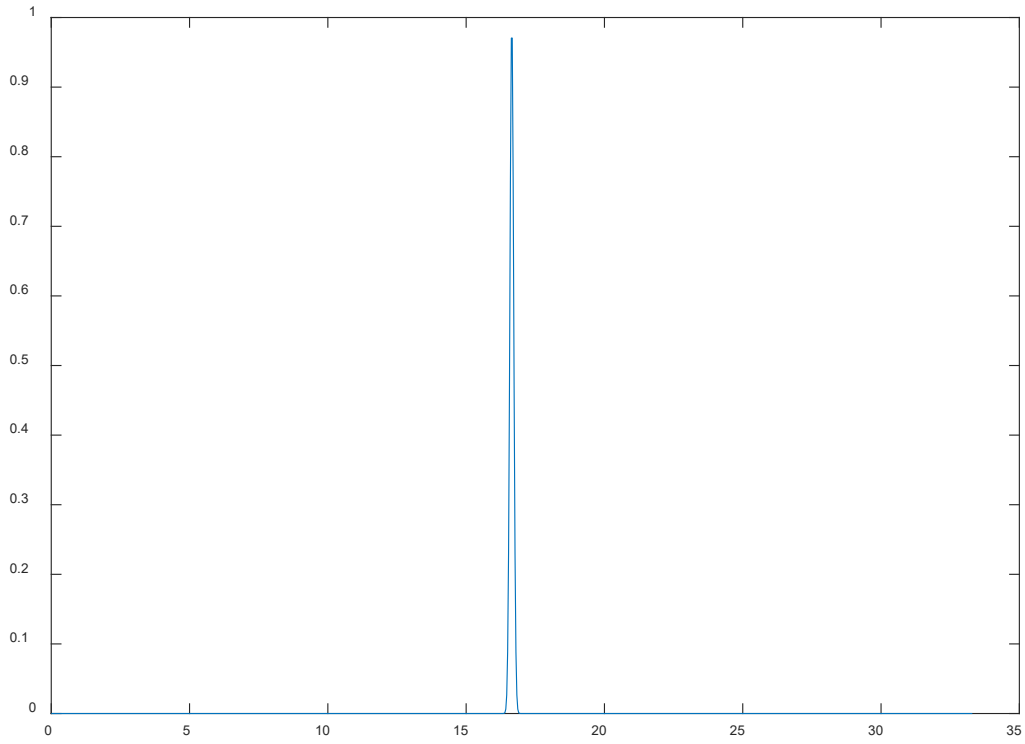
Skills: zscore (requires statistics toolbox), cumsum, conv, diff, nan, find

zscore: Simply put, a z-score (also called a standard score) gives you an idea of how far from the mean a data point is.

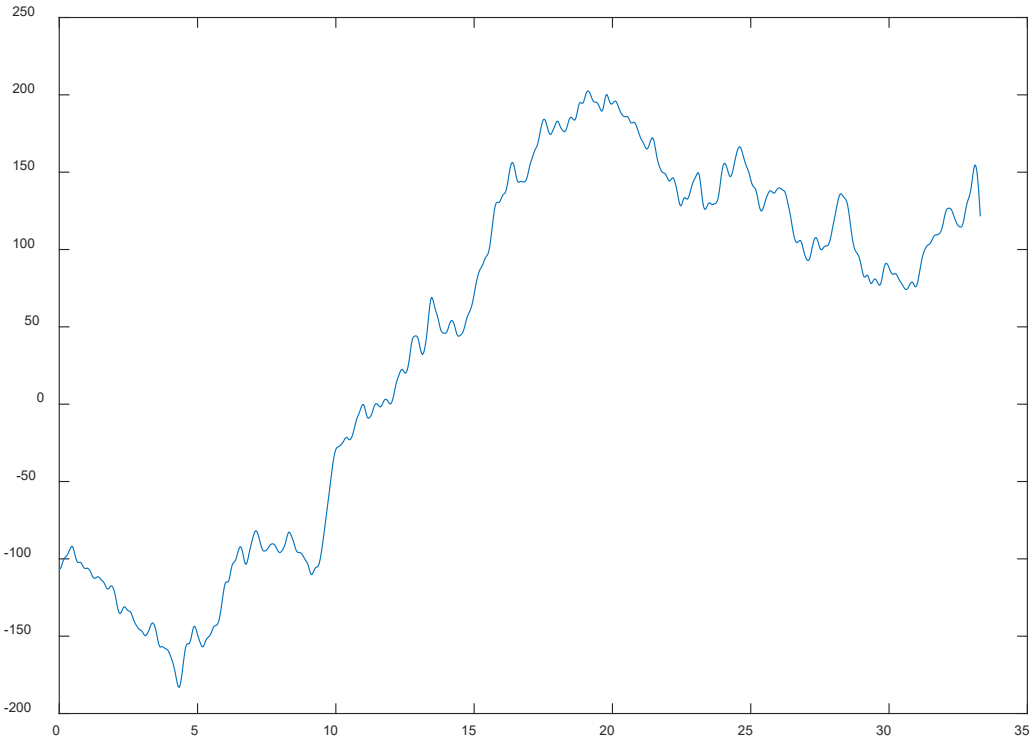
find: Find indices and values of nonzero elements

### MATLAB

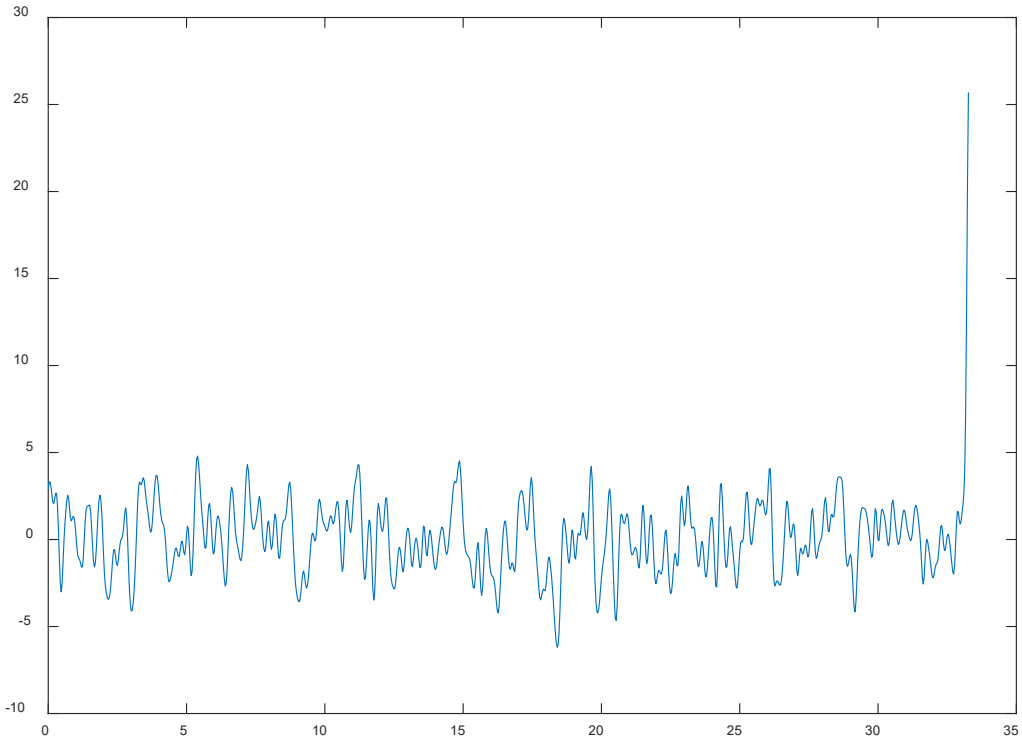
Code: MasterMATLAB_0980_derivativeSeg.m




```
% random smooth time series
N = 1000; % days
tv = (0:N-1)/30; % simulate time vector (tv) in months
gwin = exp( -zscore(tv).^2/.0001 ); % Gaussian window
>> plot(tv,gwin)
```




```
% "stock market": smoothed noise plus linear trend
signal = conv(cumsum(randn(N,1)),gwin,'same') + linspace(-100,100,N)';
>>plot(tv,signal)
```




```
% compute derivative
signalD = diff( signal );
signalD(N) = signalD(end); % increase the size by 1 to match size(tv)
>> plot(tv,signalD)
```

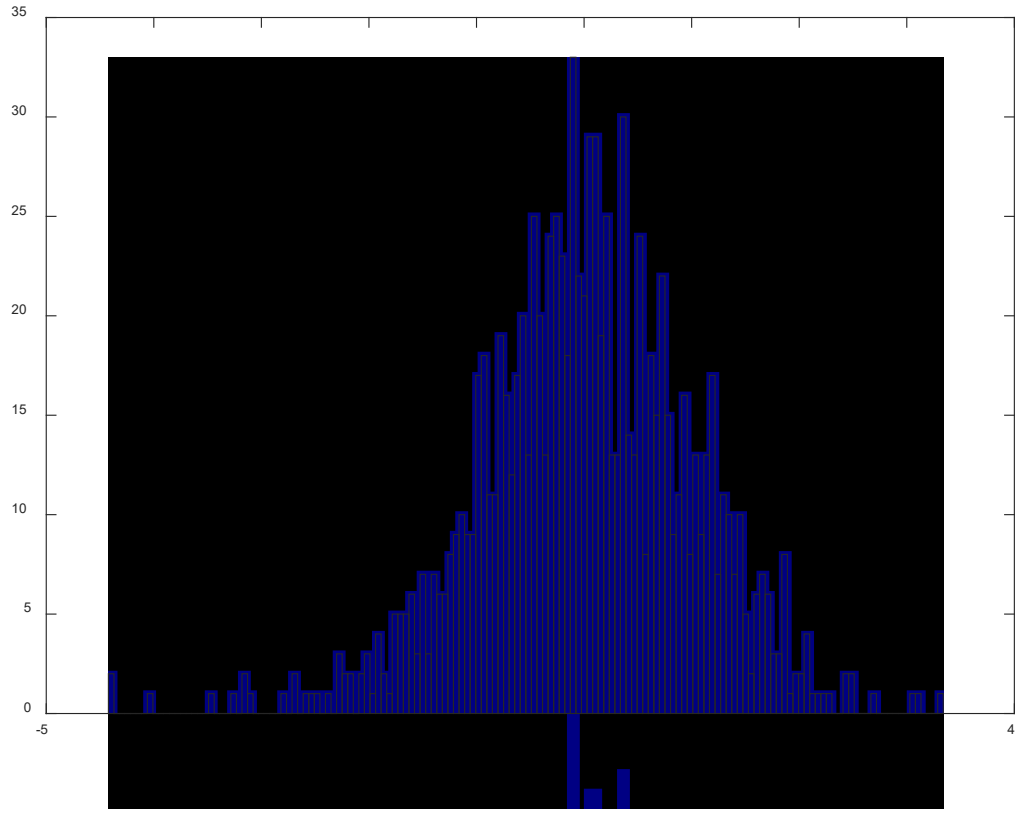



Figure 1

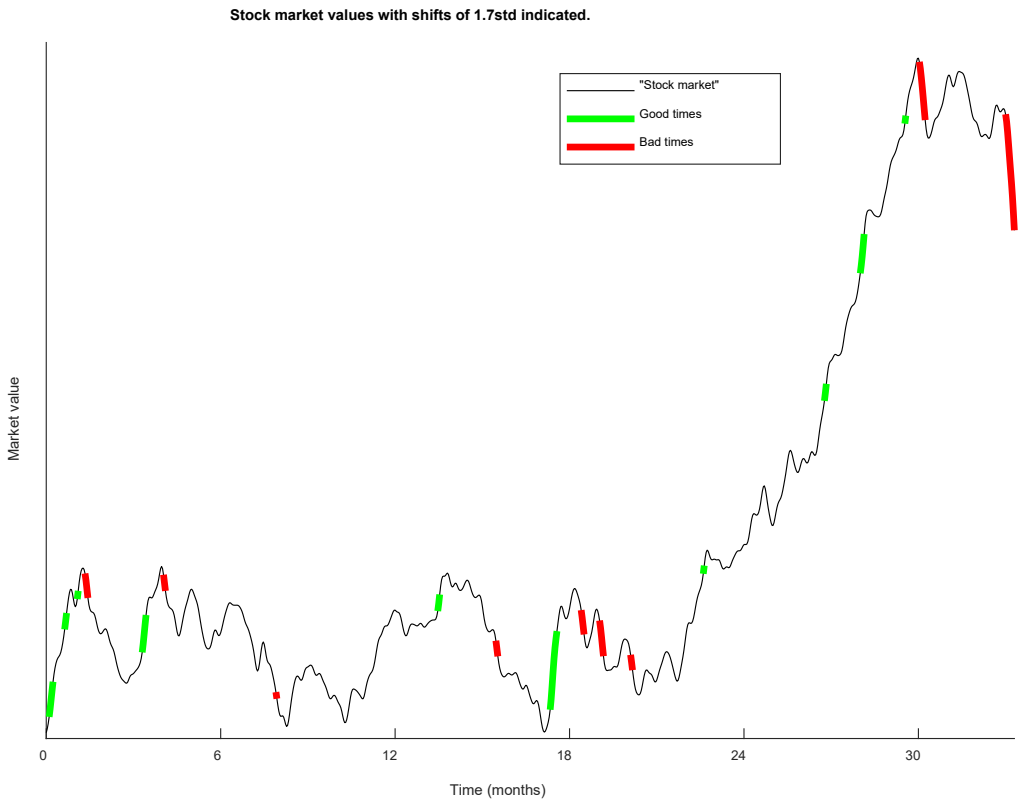



Figure 2

## 59. INTENSITY-BASED IMAGE SEGMENTATION

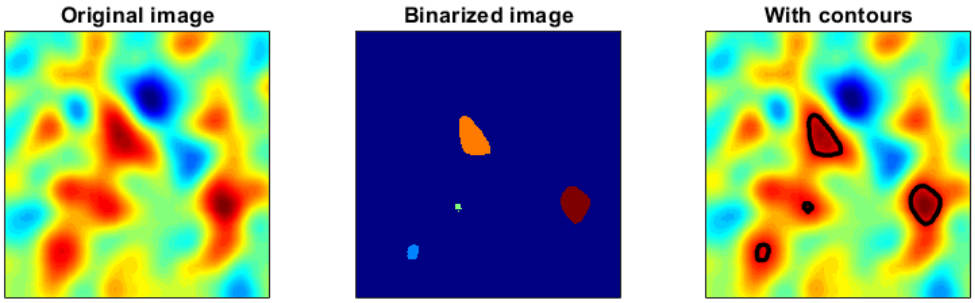
Generate a 2D smoothed random image by convolving noise with a Gaussian. Apply intensity-based thresholding and show the binarized map.

Draw contours around extreme features in the original map.

Skills: meshgrid, conv2, contour, bwlabeln (image processing toolbox)

### MATLAB

Code: MasterMATLAB_1000_intensitySeg.m




## 60. IDENTIFY NEURONS IN A MOUSE BRAIN SLICE

Use intensity thresholding to identify cells in a high-resolution picture of a brain slice. Remove any clusters that are unusually small.

Color clusters according to their size.

Use a transparency map to remove non-neuron pixels from the image.

Skills: imshow, alphadata, bwconncomp (image processing toolbox)

### MATLAB

Code: MasterMATLAB_1020_findTheNeurons.m

Image (mouse Neurons): 100048576_197.jpg

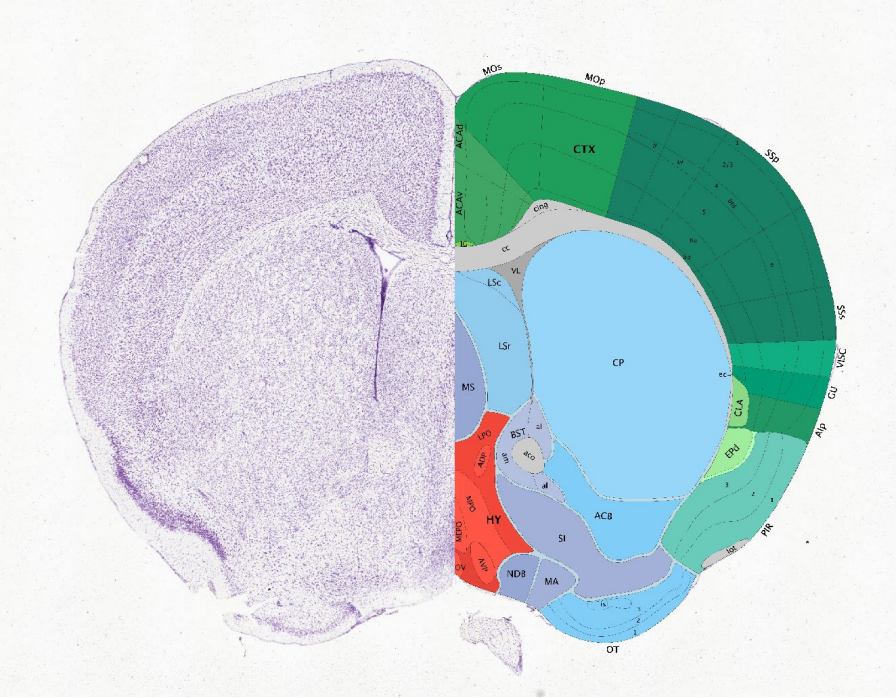



Figure 1. Mouse brain with left sise show Nissl staining. Nissl staining is convenient for measuring the density of neurons because stained cells are clearly defined and easily measured.

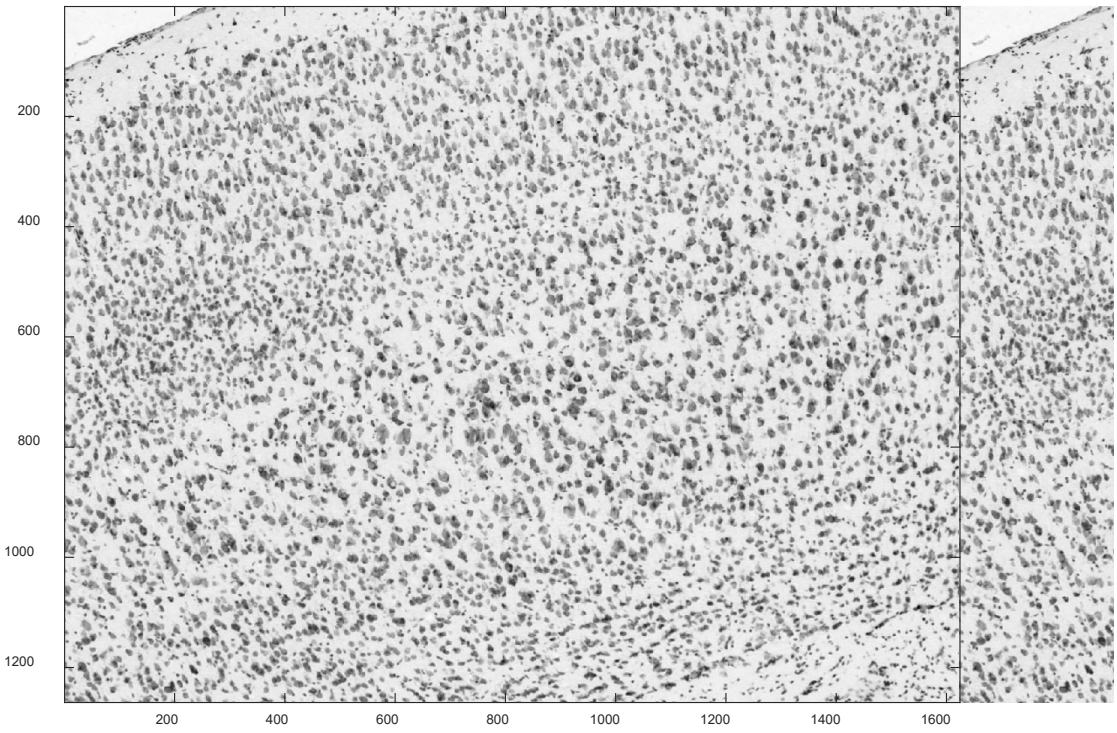



Figure 2. Close-up of section in upper left area.

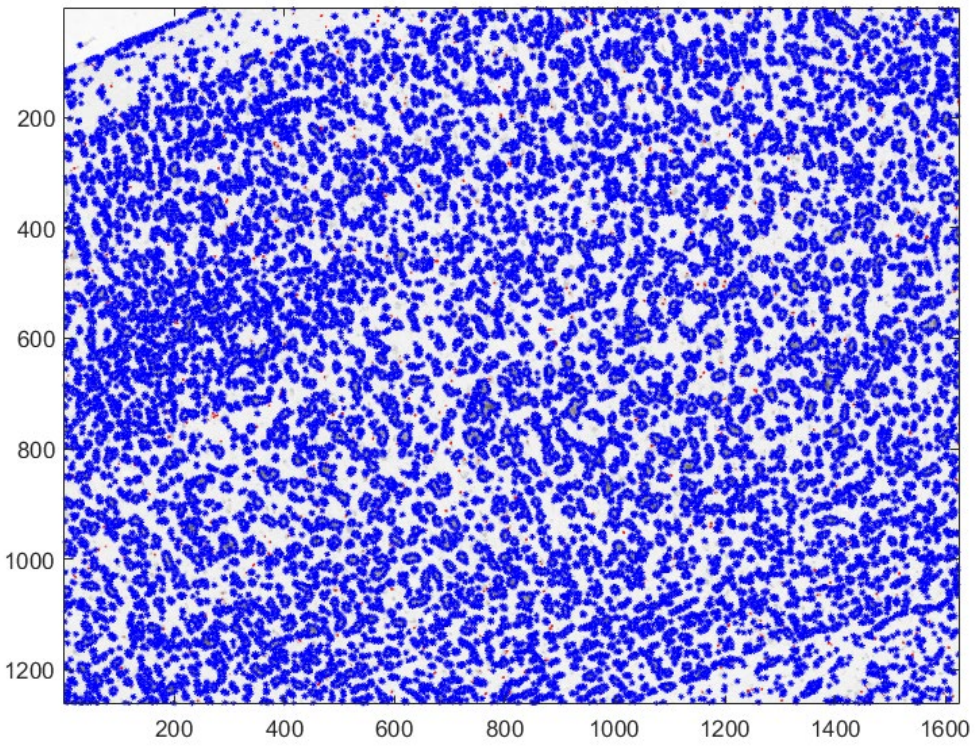



Figure 3. Contour neurons.

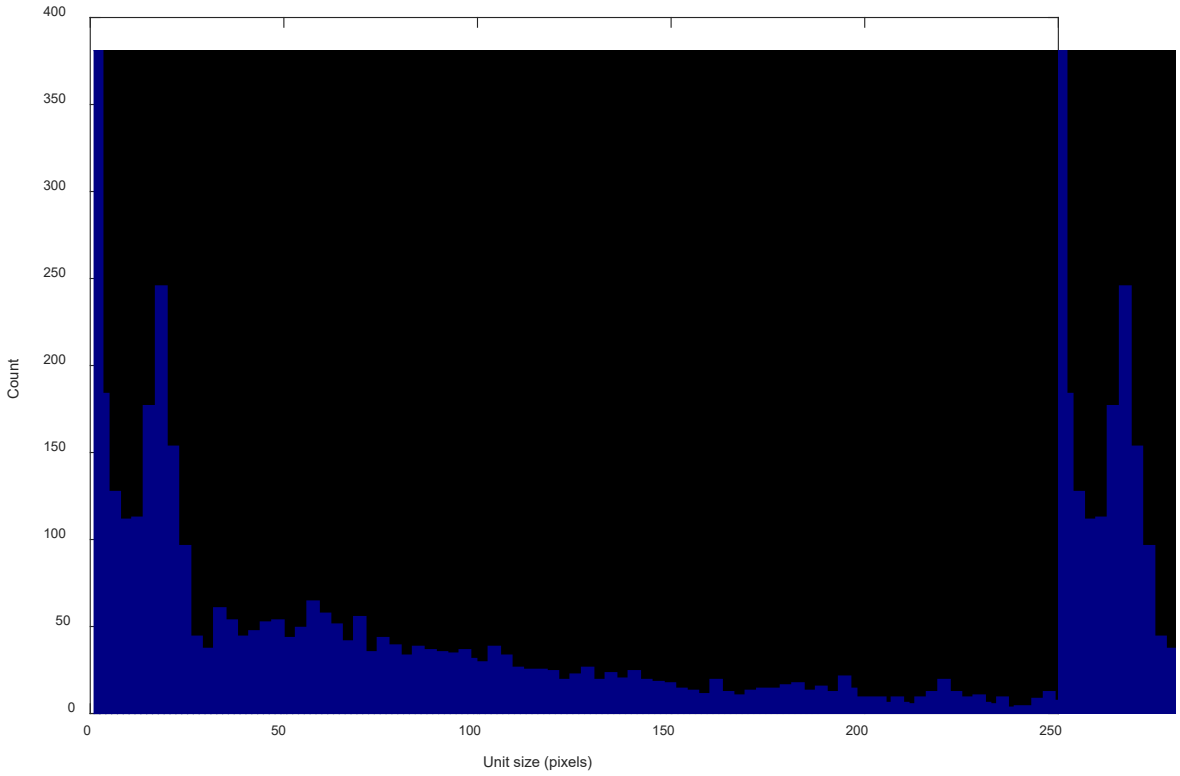

400
350
300
250
200
150
100
50
0
Count
0
50
100
150
200
250
Unit size (pixels)

Figure 4.

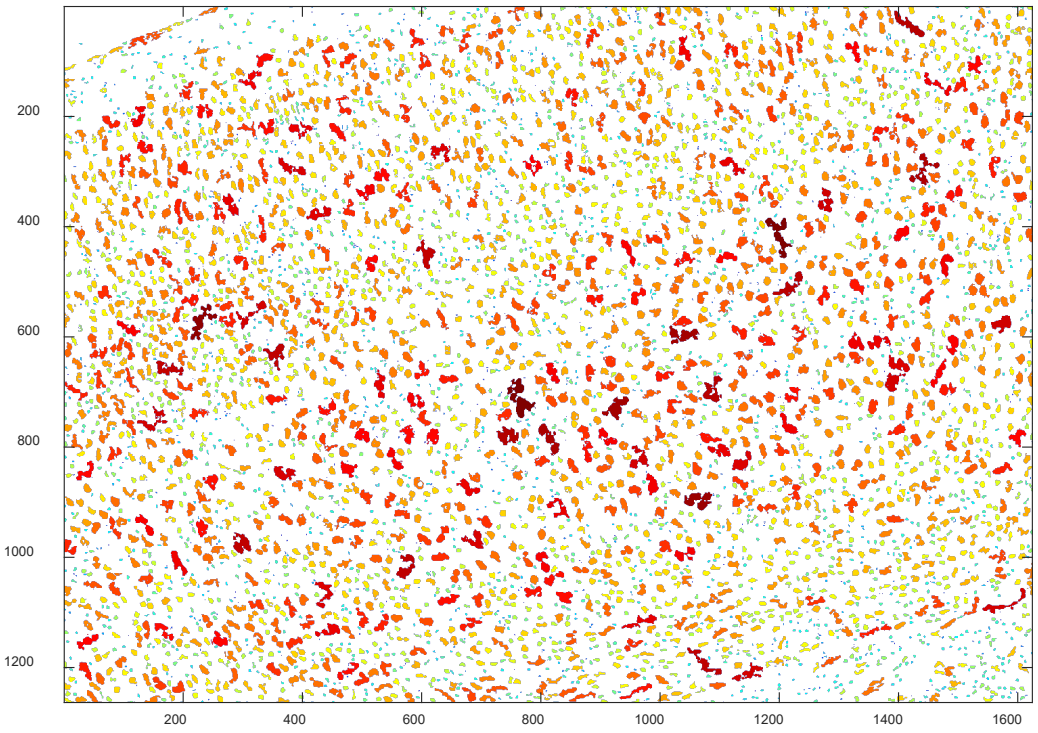



Figure 5.

## SECTION 11: DATA ANIMATIONS

### 61. RANDOM FLOATING BALL

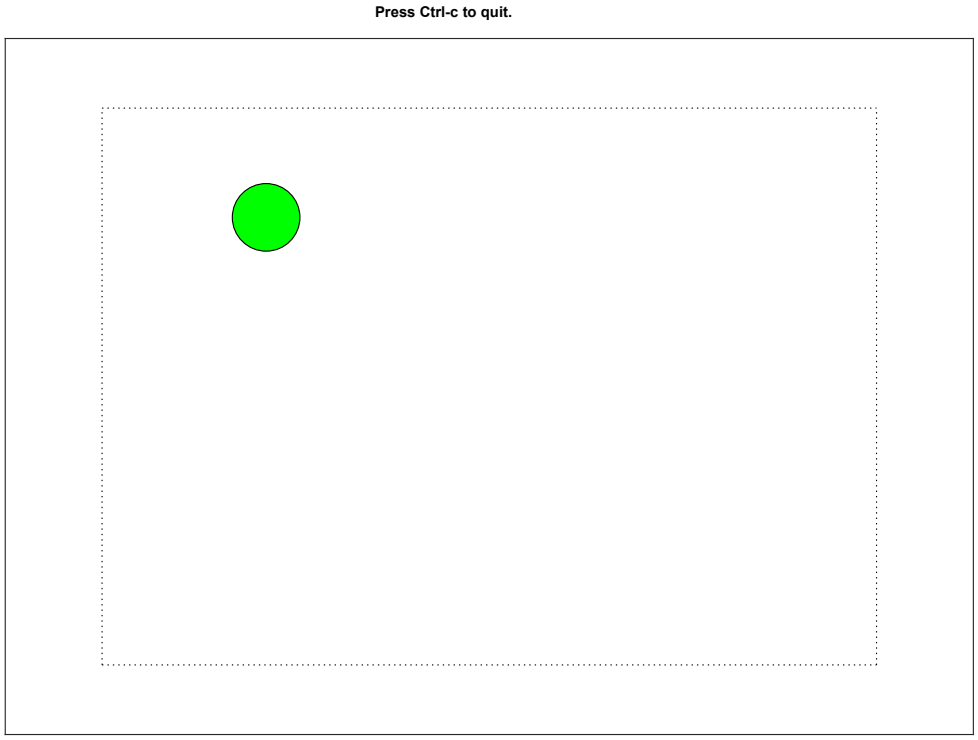
Make a movie of a ball that floats around randomly in the figure. Transport the ball to the other side when it reaches the edge.

Change the color of the ball when it's near the edge.

Skills: while, set, any

### MATLAB

Code: MasterMATLAB_1040_floatingBall.m




## 62. THE SQUARE CHASES THE MOUSE

Make a movie of a square that moves to the most recent mouse-click.

Keep the square inside the axis, even if the mouse was clicked outside the axis.

Skills: while, set, pause, sign

### MATLAB

Code: MasterMATLAB_1060_squareChasesMouse.m


## 63. THE MAGICALLY MATERIALIZING PEAKS

Make a movie of the MATLAB “peaks” surface slowly materializing.

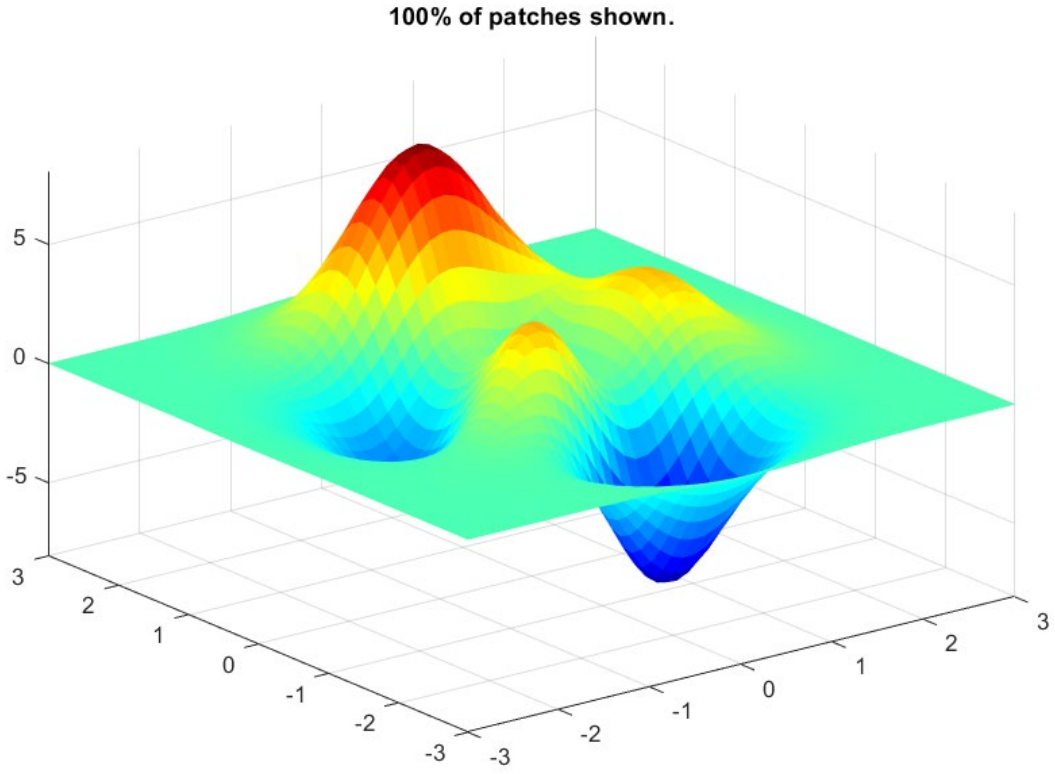
Update the plot title to show progress.

Skills: peaks, surf, set, pause, randsample (Statistics Toolbox), randperm

*peaks* is a function of two variables, obtained by translating and scaling Gaussian distributions, which is useful for demonstrating *mesh, surf, pcolor, contour,* and so on.

### MATLAB

Code: MasterMATLAB_1080_magicPeaks.m




## 64. SMOOTH SAILING: THE MOVIE

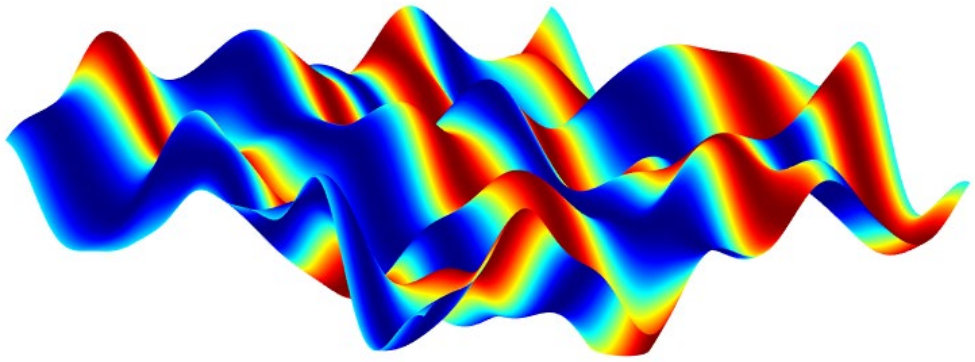
Make a movie of 2D sine waves traveling on a “sea” of Perlin waves. Export the movie as an .avi file.

Thanks to Ken Perlin for the development of Perlin Noise, a technique used to produce natural appearing textures on computer generated surfaces for motion picture visual effects. The development of Perlin Noise has allowed computer graphics artists to better represent the complexity of natural phenomena in visual effects for the motion picture industry. Simplex noise alleviates some of the problems with Perlin's "classic noise", among them computational complexity and visually-significant directional artifacts. https://en.wikipedia.org/wiki/Perlin_noise

Skills: surf, shading, rotate3d, ifft2, fftshift, Videowriter, writeVideo

### MATLAB

Code: MasterMATLAB_1081_SailingMovie.m



Generates an avi movie called tumblingSea.avi to disk.

## 65. REAL-TIME AUDIO SPECTRUM FROM MIC

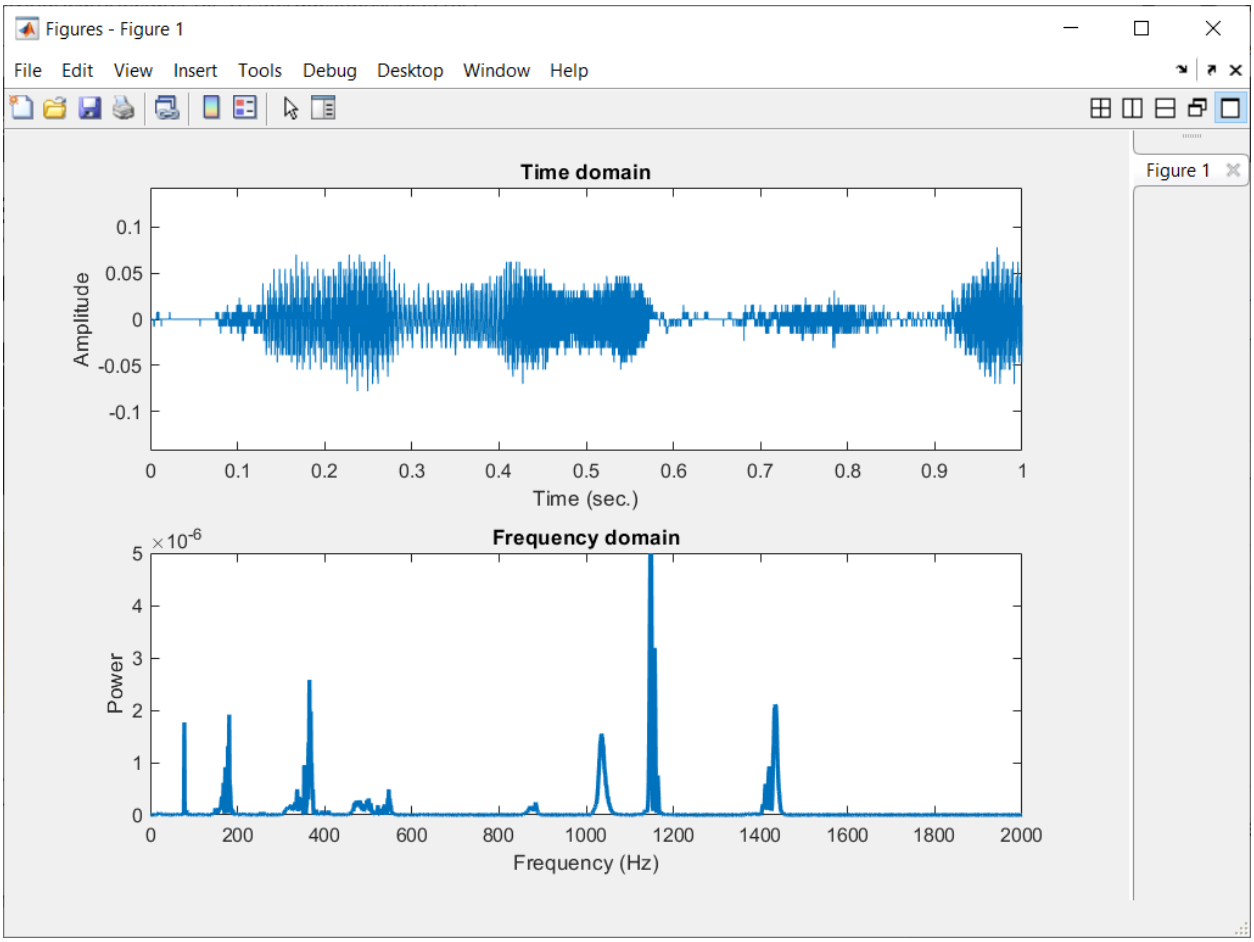
Record data from your computer's microphone. Display your voice in the time domain and in the frequency domain (via FFT).

Make adjustments to get the spectrum to show whistling.

Skills: audiorecorder, record, fft, getaudiodata

### MATLAB

Code: MasterMATLAB_1100_realTime_mic.m




## 66. MOBIUS TRANSFORMATION

Make a movie of the Möbius transformation.

Get rid of two for-loops!

Skills: complex, set, imagesc, bsxfun

Möbius transformation is a function of a complex number and time. Use this transformation in physics to study electromagnetism.

## Equation:

$$f = \frac{(t-1)+(1+t)z}{(1+t)+(t-1)z}$$

$$z = a + ib$$

$$t \in (.2, 2)$$

## MATLAB

Code: MasterMATLAB_1120_MobiusTransform.m

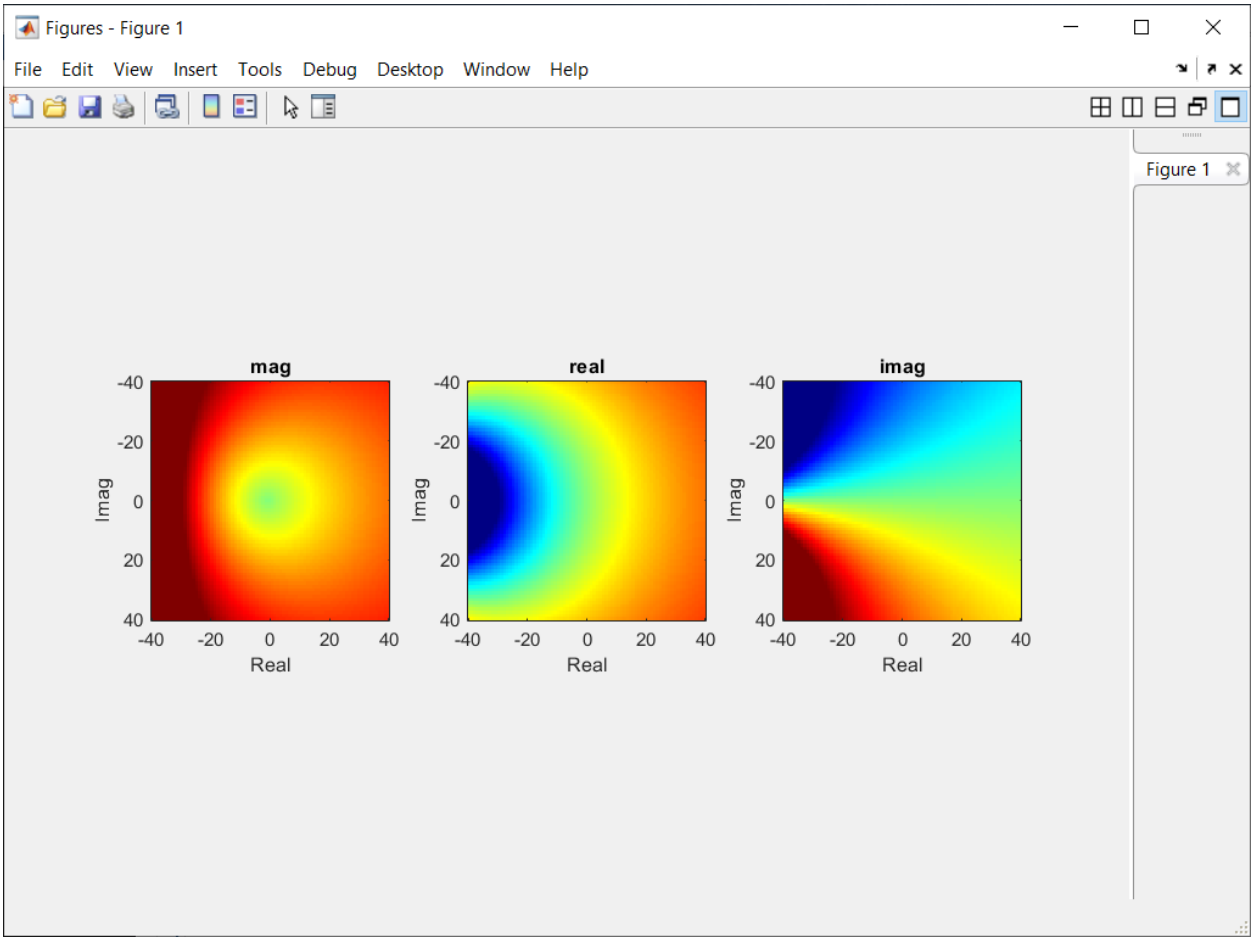
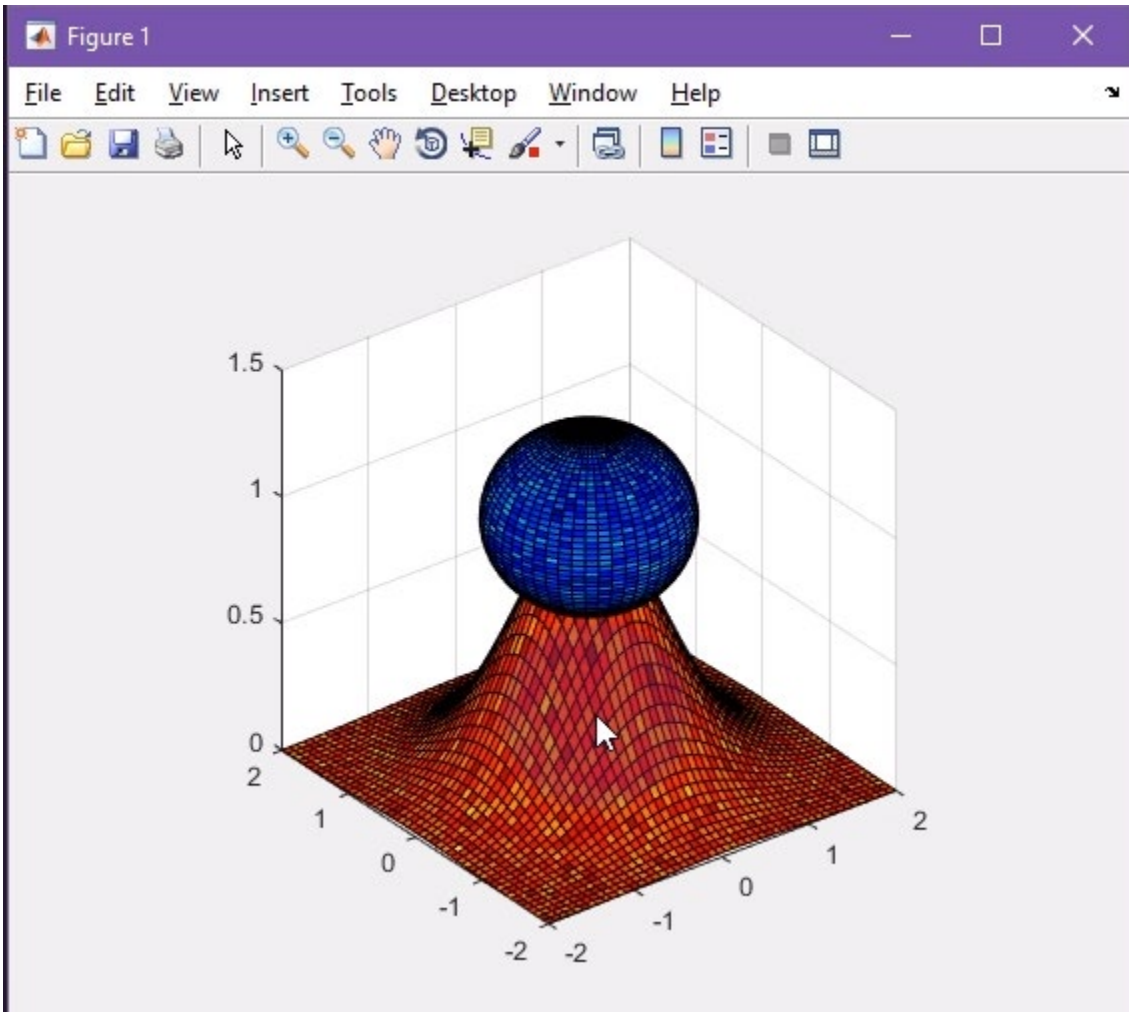



Figure 1. Snapshot in time.

### 67. SOLVED: UFO ON A SANDCASTLE

Make a movie of a color-throbbing ball on a traffic cone.




#### MATLAB

Code: solved_Plot_the_Sphere.m

Note: Still need to fix color before sharing final image

## SECTION 12: GRAPHICAL USER INTERFACES

### 68. DIALOG BOX FOR USER INPUT

Use a dialog box to get matrix size, title, and filename information.

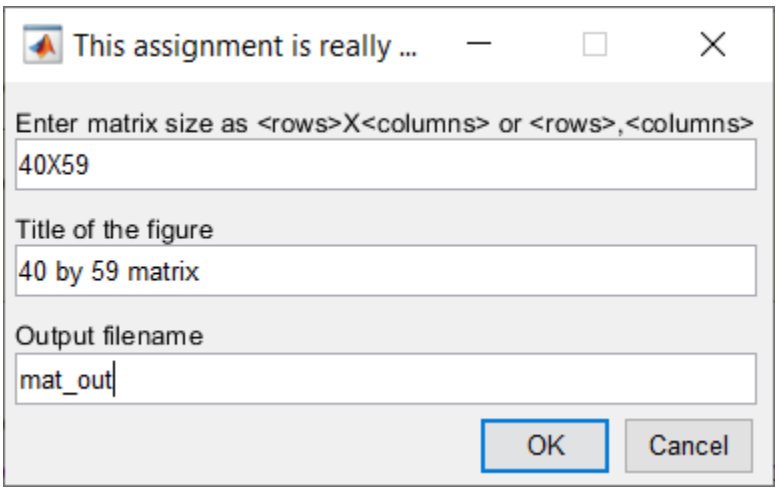
Then create a random matrix with that size and save as png.

Make the input work for MxN or M,N

Skills: inputdlg, strfind, sscanf, print

MATLAB

Code: MasterMATLAB_1140_dialogBox.m



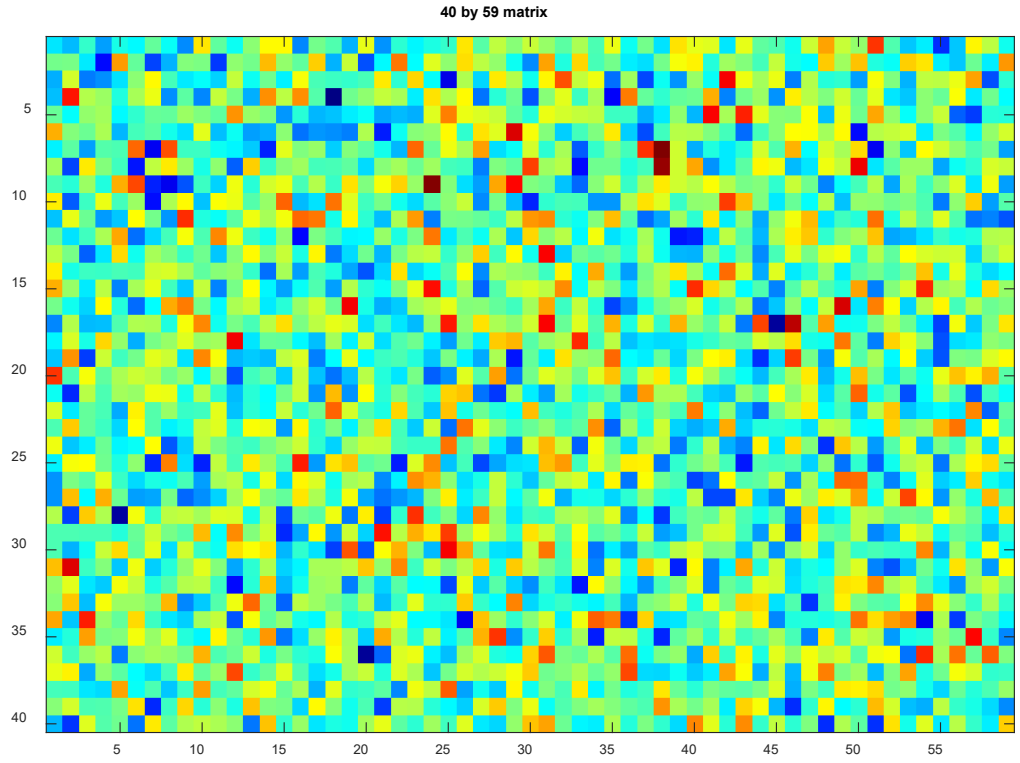



Figure 1. Result of entering data into dialog box: 40x59, 40 by 59 matrix, mat_out (png image file)

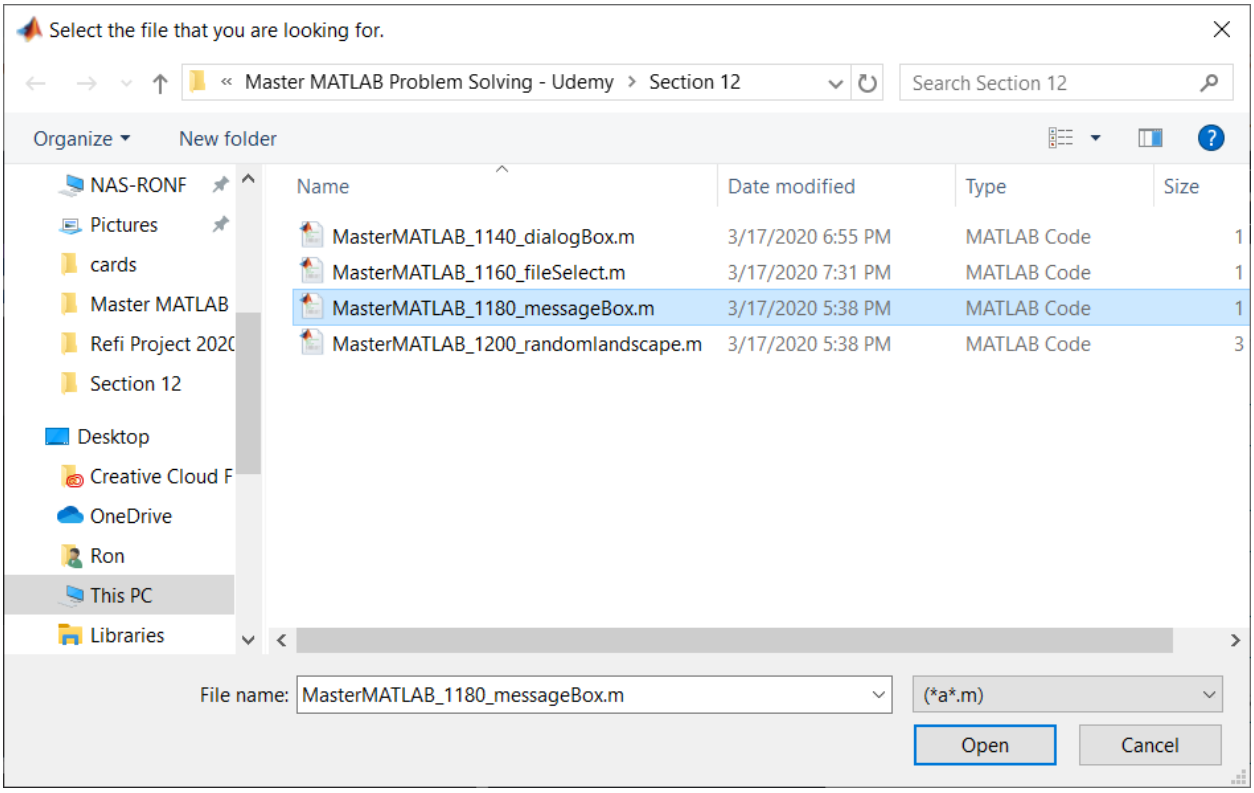
## 69. INTERFACE TO SELECT A FILE

User file interfaces to select a specific file and a specific folder.

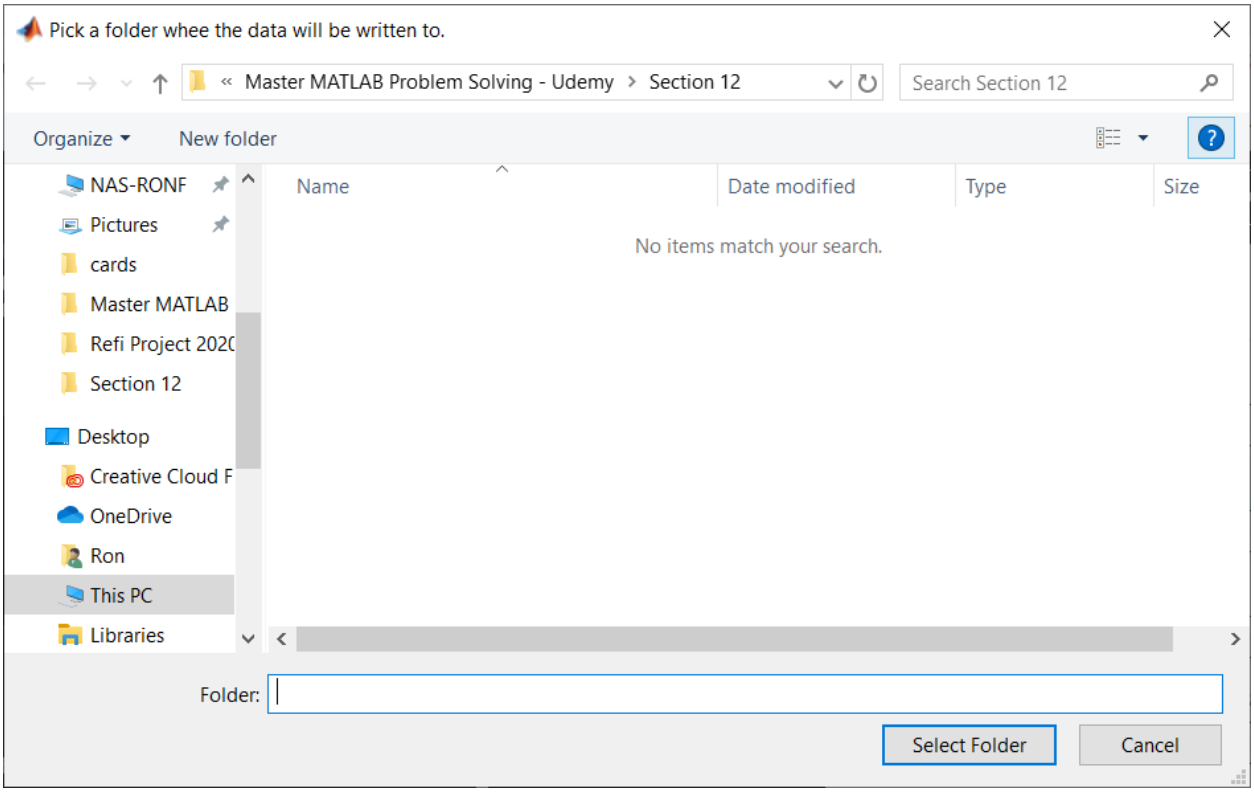
Skills: uigetfile, uigetdir

### MATLAB

Code: MasterMATLAB_1160_fileSelect.m




[filename,pathname] = uigetfile('*a*.m','Select the file that you are looking for.');



```
% select a folder
pathname = uigetdir(pwd,'Pick a folder whee the data will be written to.'); % pwd present working
directory
pathname = [ pathname '\' ]; % uigetdir requires a final path character before cat with a file name

save([ pathname 'testfile.txt' ],'pathname', '-nocompression')
```

## 70. INPUT AND MESSAGE BOXES

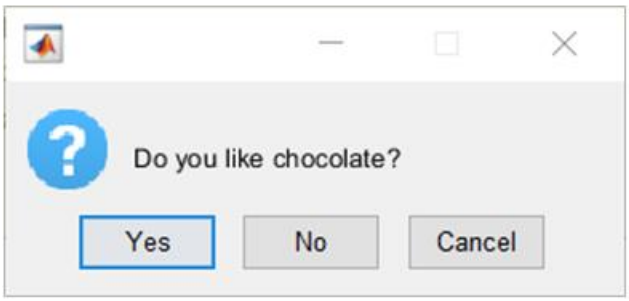
Ask the user whether they like chocolate and how much chocolate they eat. Then inform them that the amount of chocolate is not enough!

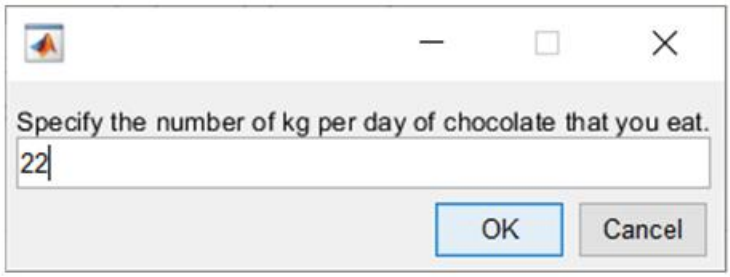
Make the message box purple and set the window size.

Skills: questdlg, inputdlg, msgbox

### MATLAB

Code: MasterMATLAB_1180_messageBox.m





## 71. GUI TO CREATE RANDOM LANDSCAPES

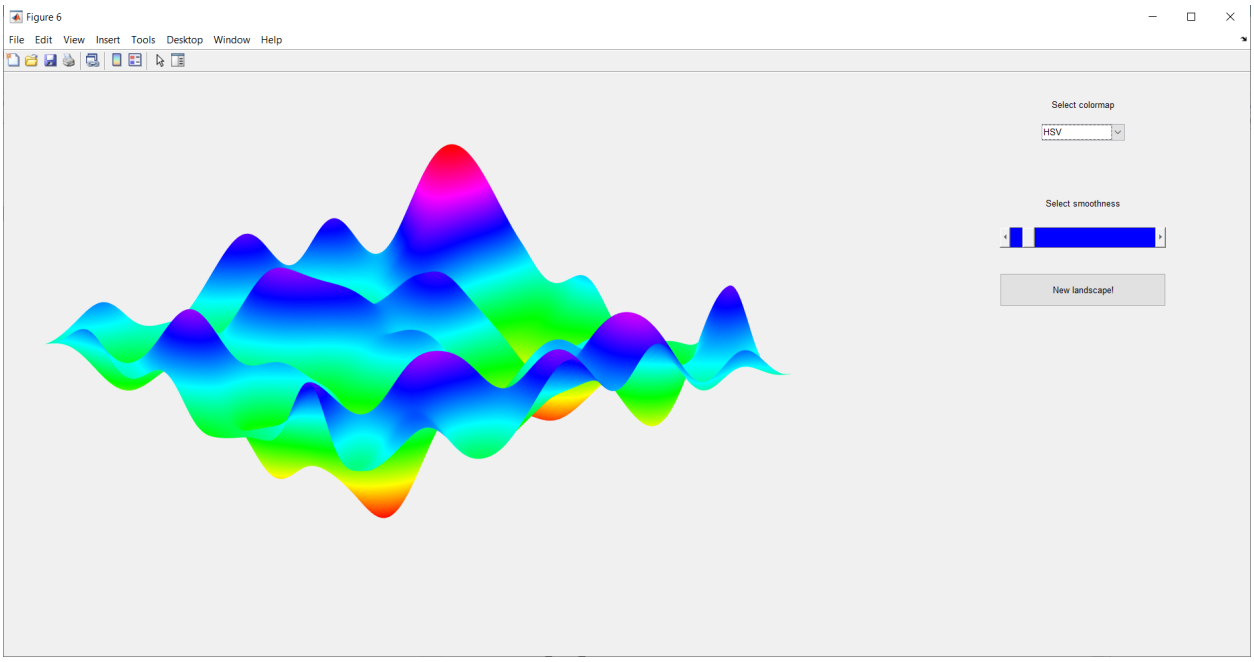
Understand the code. Move the locations and sizes of the GUI components.

Skills: uicontrol, axes, align, function

*dbquit [all]*: Quit debug mode from command window.

### MATLAB

Code: MasterMATLAB_1200_randomlandscape.m




## 72. GUIDE TO SIGMOID PARAMETER SPACE

Use GUIDE to create a UI that facilitates understanding the parameterization of the sigmoid function.

Skills: get, set, handles, Callback, exp

## Sigmoid Equation:

$$f(x) = -\frac{a}{1 + e^{-b(x-c)}}$$

### MATLAB:

Code:

GUIDE layout: guide2sigmoid.fig

GUIDE code: guide2sigmoid.m

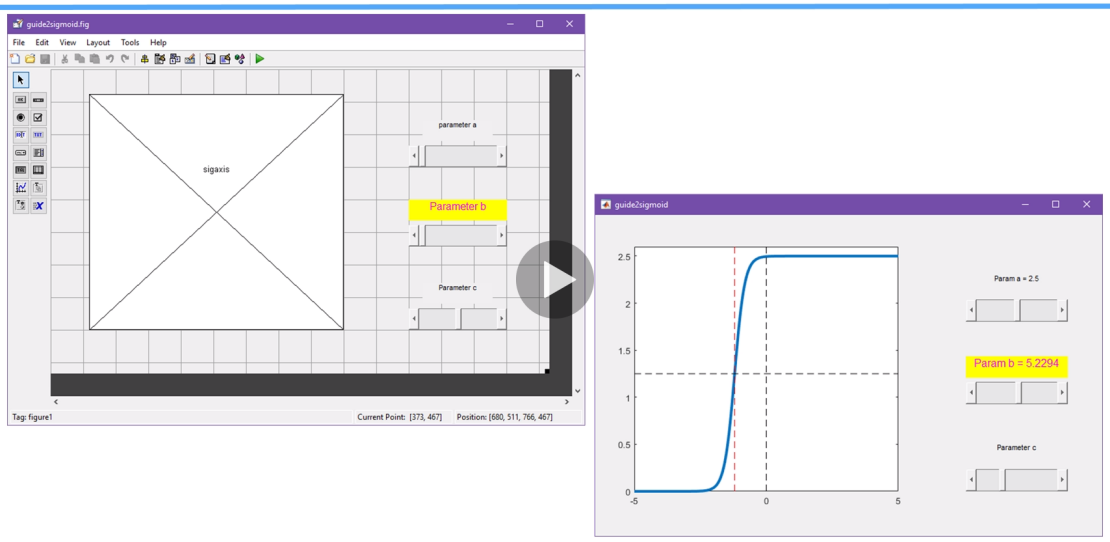
GUIDE image: sigmoidEquation.JPG

App Developer (code, layout): guide2sigmoid_App.mlapp

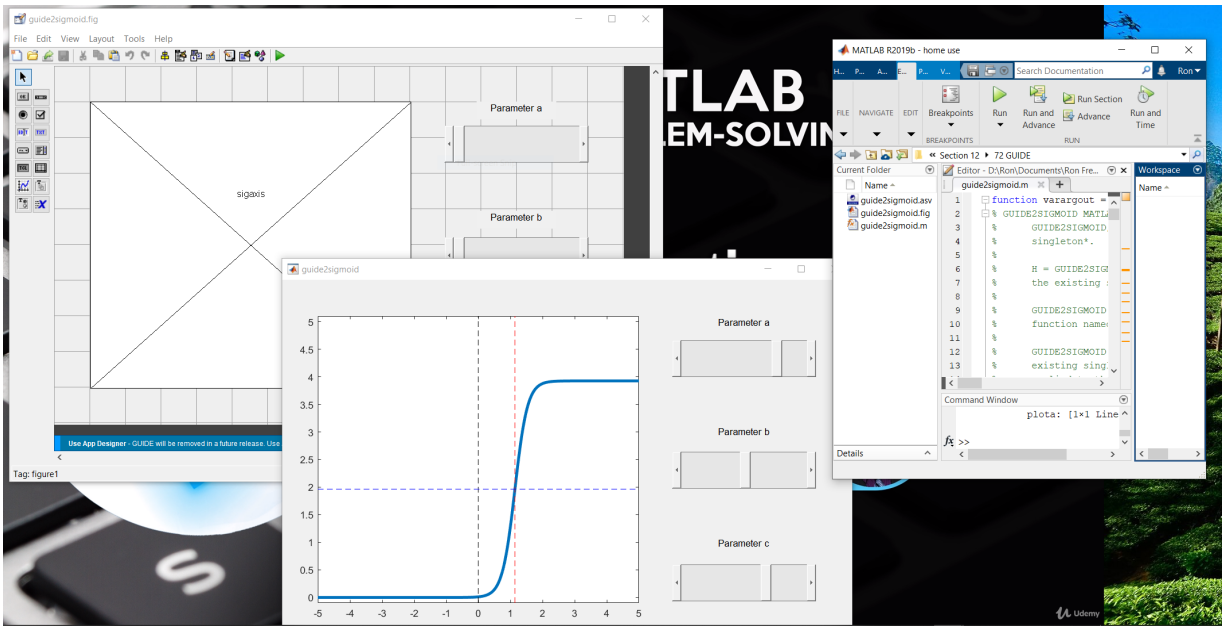
App Developer conversion tool report: guide2sigmoid_App_report.html

App Developer image: sigmoidEquation.JPG

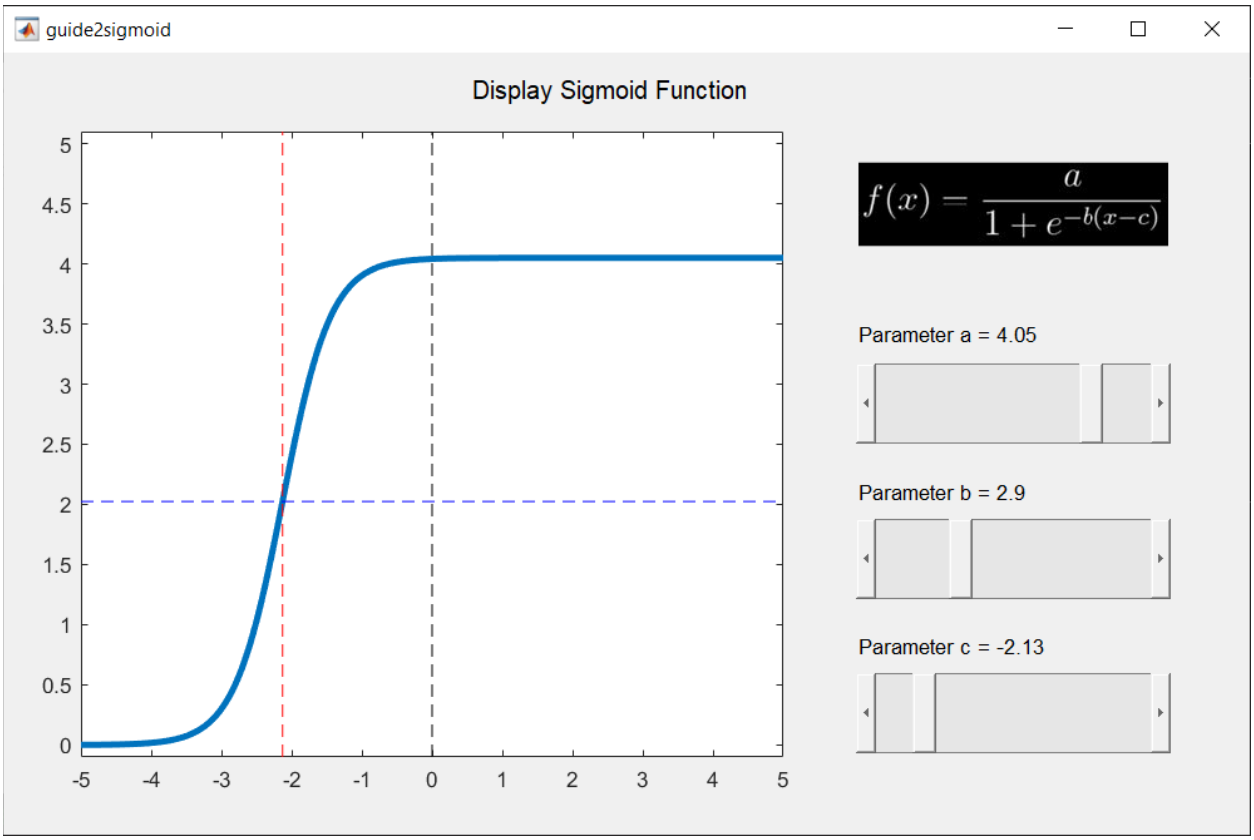
# GUIDE to Sigmoid Parameter Space



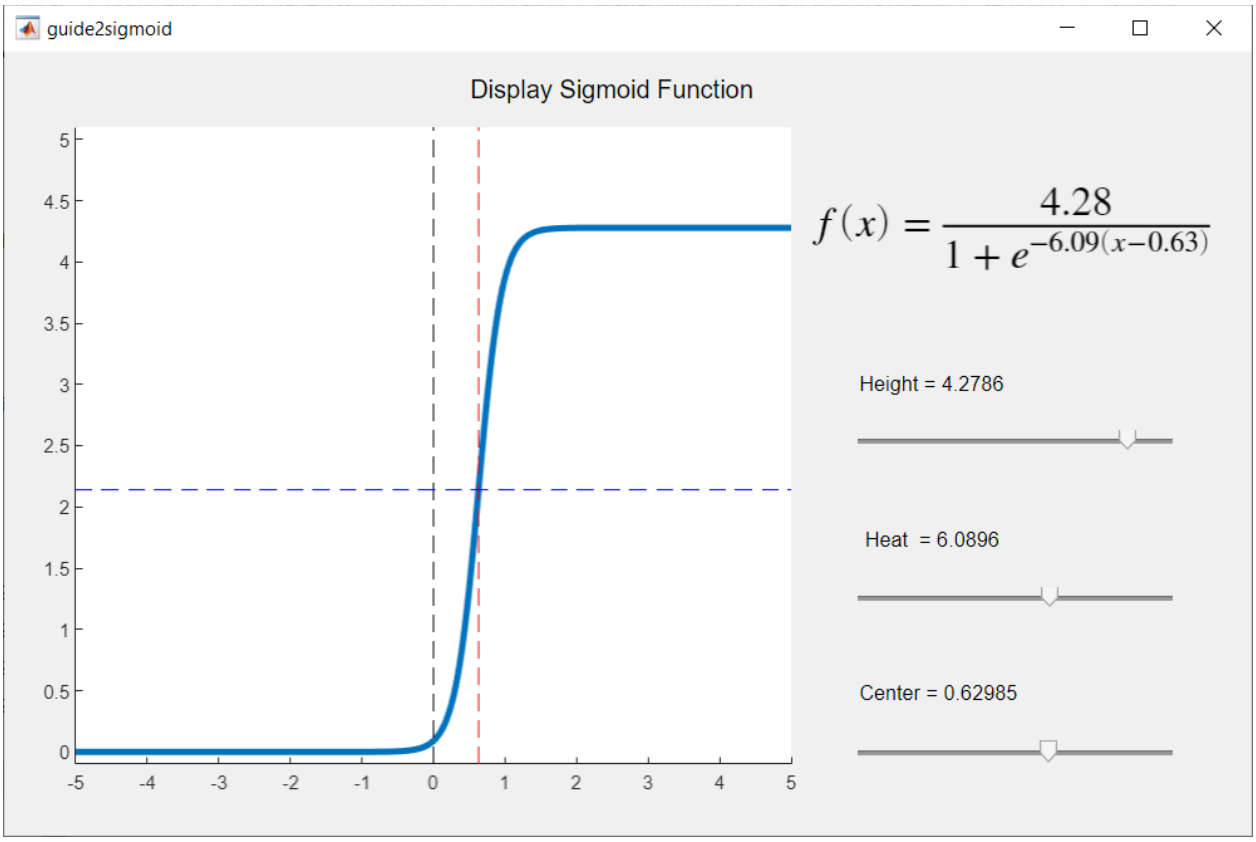

MASTER MATLAB THROUGH GUIDED PROBLEM-SOLVING
GUIDE to Sigmoid Parameter Space




GUIDE + script + running code




Final GUIDE application




App Developer version after using the GUIDE to App Developer Conversion Tool showing parameter values and graphic image.

## SECTION 13: FUNCTIONS AND ANONYMOUS FUNCTIONS

### 73. SAME-LENGTH DIFFERENTIATION

Write a function that computes the 1st and 2nd derivative and provides outputs of the same length as the original signal.

Implement input-checks to ensure the input is (1) numeric and (2) a vector.

Extract only the second output of a function without the first.

Skills: diff, error, isnumeric

#### MATLAB

Code: MasterMATLAB_1240_diffx.m

Function: diffx.m

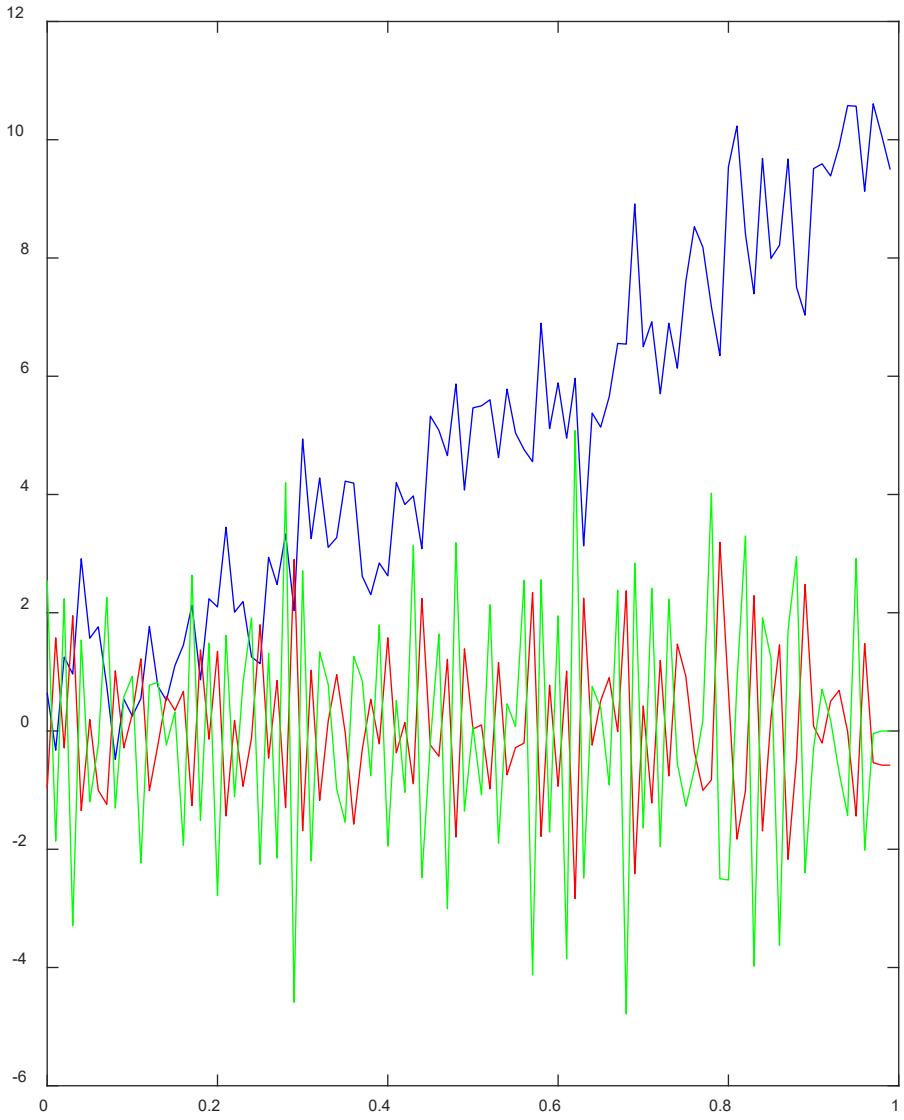



Figure 2: Shows diffx(x) function with end padding so all plots have the same vector sizes

## 74. DAMPED OSCILLATOR

Write an anonymous function that returns a damped oscillator, given inputs of frequency and decay.

Produce an image of the oscillator output for a range of decay parameters.

Skills: function handle (@) contour, func2str

Equation:

$$\sin(2\pi f t)\, e^{-\tau t}$$

## MATLAB

Code: MasterMATLAB_1260_dampOsc.m

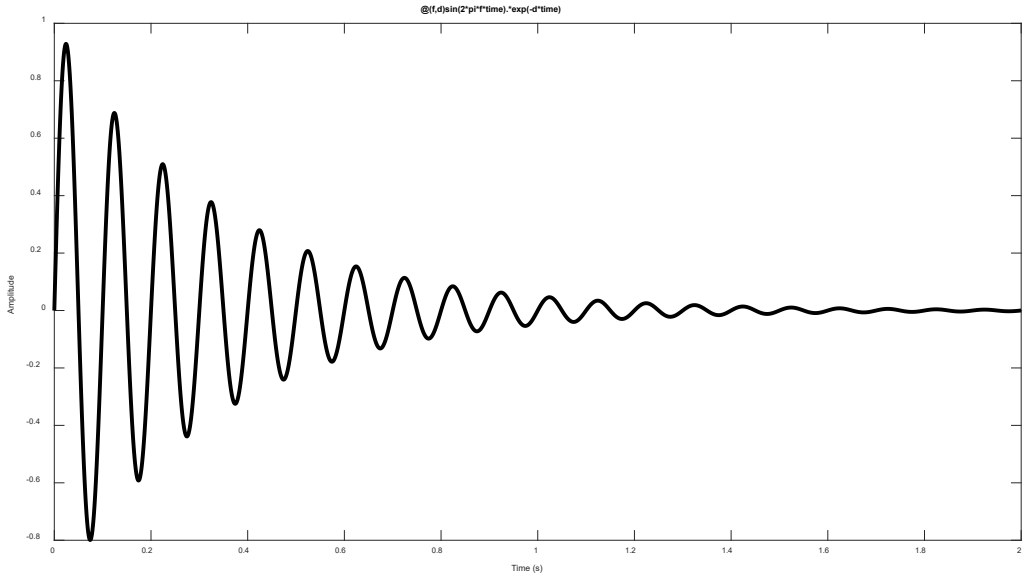



Figure 1

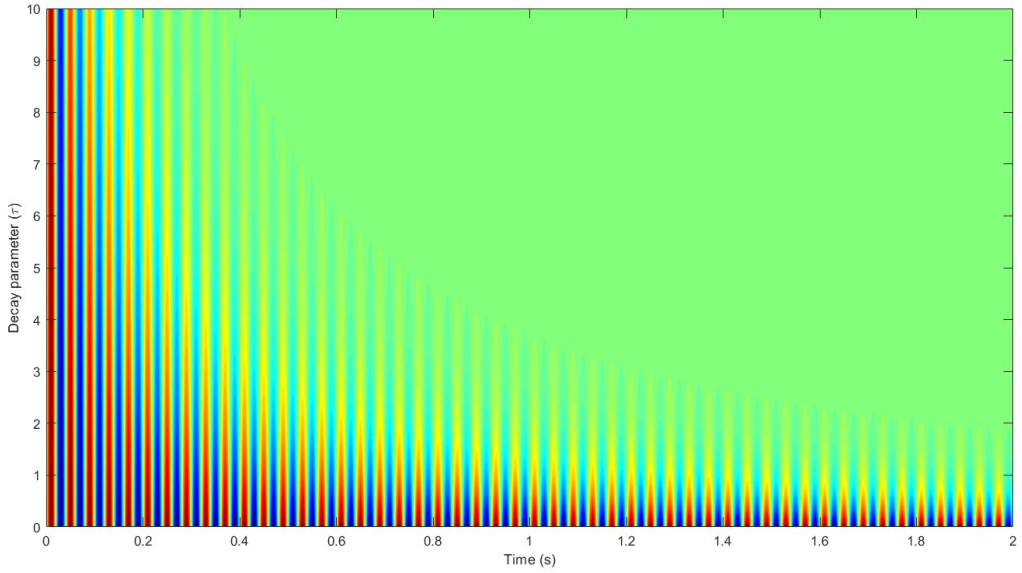



Figure 2

## 75. UNSOLVED: DAMPED ARCSINE

Repeat the damped oscillator exercise but use the arcsine function over time (0, 2pi). Show a surface of its magnitude for tau (0,3).

Skill: arcsin

Part 1: plot the real (blue), imaginary (orange), and magnitude (yellow)

Part 2: plot the magnitude of the surface

## 76. FIND AND EXTRACT A FUNCTION CORE

Search through the MATLAB 'median' function to extract the key lines that implement the operation. Get that code to work outside the function.

Skills: Understanding and mining existing code.

Median is a descriptive statistic that describes the middle value in a distribution of numbers. The number closest to the center of the distribution is the median.

### MATLAB

Code: MasterMATLAB_1280_searchFunc.m

```
% some random numbers
x = randn(1001,1);

% compute the median from code extracted from MATLAB (via edit mean)
x = sort(x);
half = floor(length(x)/2);
y = x(half+1)

% compute the median using MATLAB function
median(x)
```

## 77. SMOOTH PLOTTING FUNCTION WITH OPTIONS

Write an external function that smooths an input vector, and optionally plots the result with a specified title.

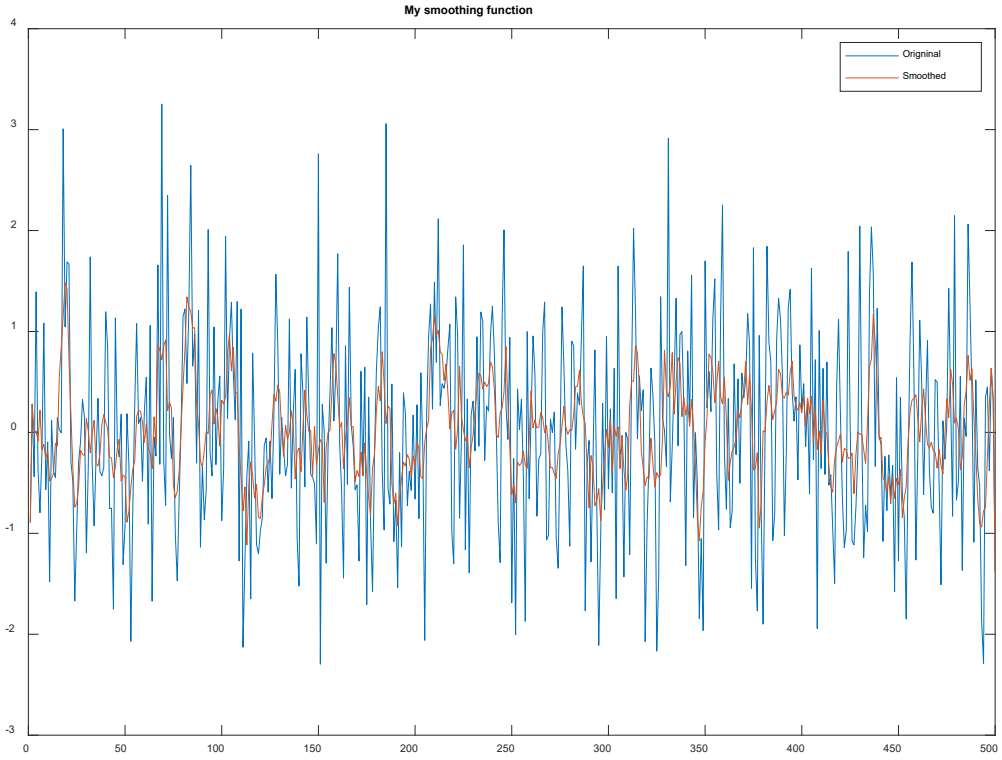
Set default options if none are provided.

Skills: nargin, varargin, error, title

### MATLAB

Code: MasterMATLAB_1300_smoothPlotter.m

Function: MM_meansmoothplot.m




## 78. SOLVED: ZSCORE FUNCTION

Write a function that will compute the zscore of a vector, the columns of a matrix, or the entire matrix.

MATLAB

Code: test_myzscore.m

Function: myzscore.m

# SECTION 14: FIND, MIN, MAX

## 79. FIND POINT CLOSEST TO SPECIFIED VALUE

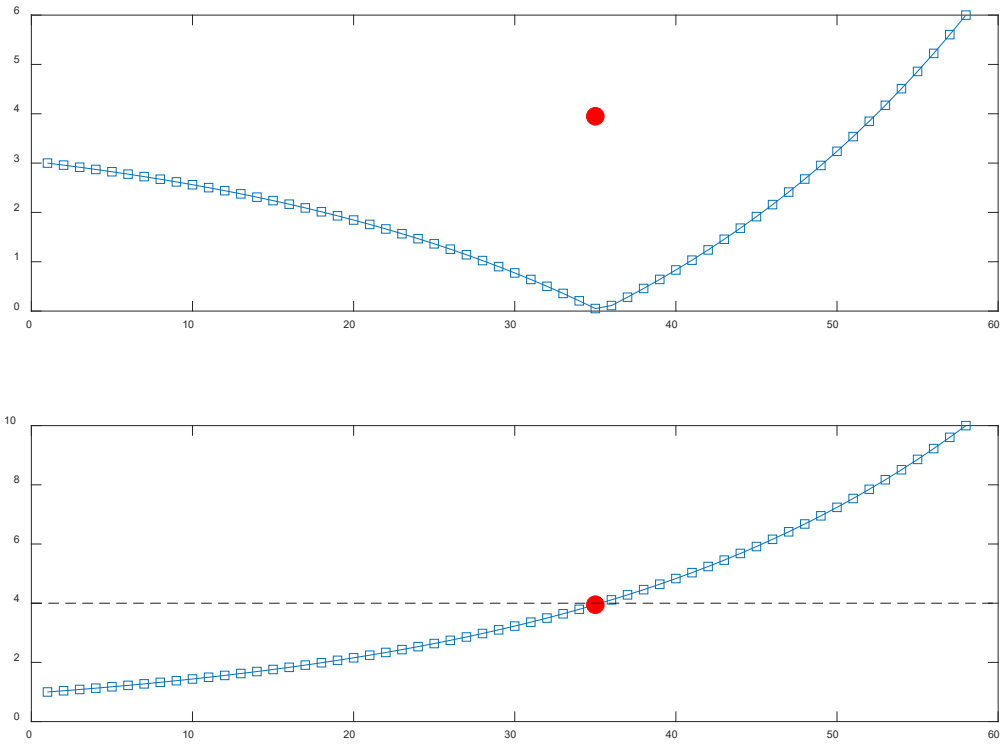
Given a monotonically increasing list of numbers, find the index in that list closest to a specified value.

Repeat for matrices instead of vectors.

Skills: min-abs, dsearchn, ind2sub, reshape

## MATLAB

Code: masterMATLAB_1320_closestPoint.m




## 80. SOLVED: MANUAL PEAK-PICKING

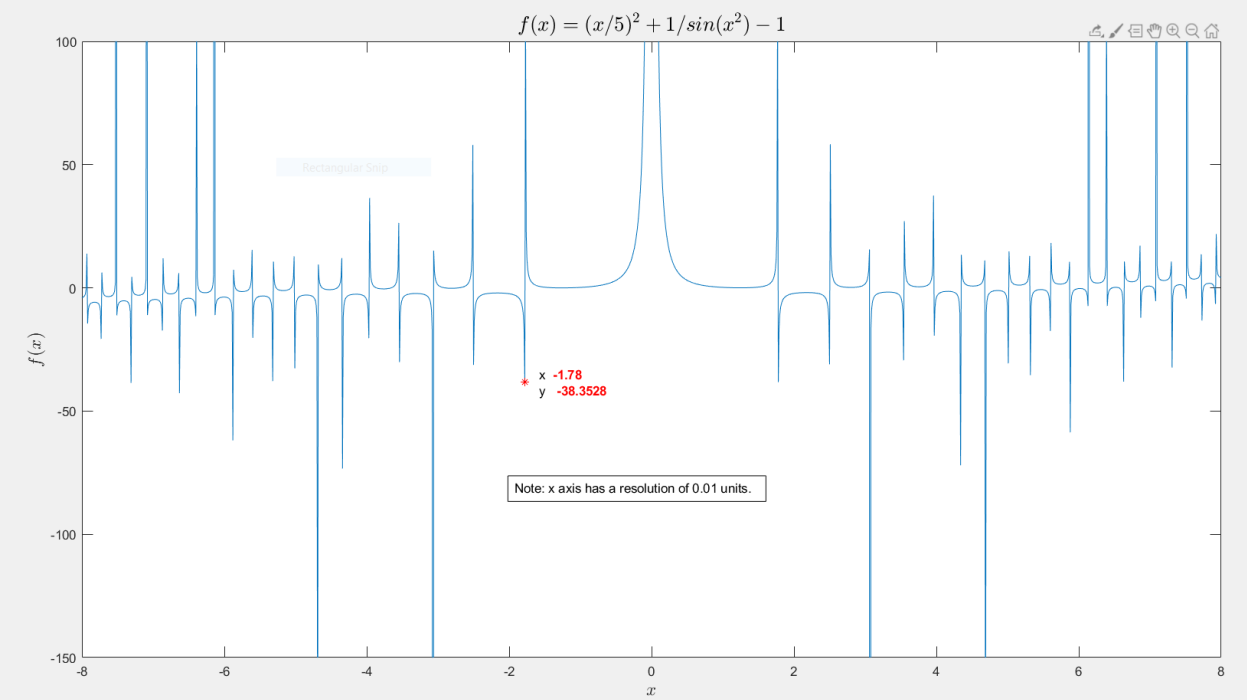
Identify the XY coordinates of the local minimum around x = -1.8

Equation:

$$y = (\frac{x}{5})^3 + \frac{1}{\sin(x^2)} - 1$$

## MATLAB

Code: manualPeakPicking.m




## 81. FIND NEGATIVE EXTREMA IN A 2D FUNCTION

Implement a 2D nonlinear function (see next slide) and find the global minimum. Use bsxfun!

Identify minima points with a threshold of 0.01.

Skills: log, bsxfun, min, max, sub2ind

## Equation:

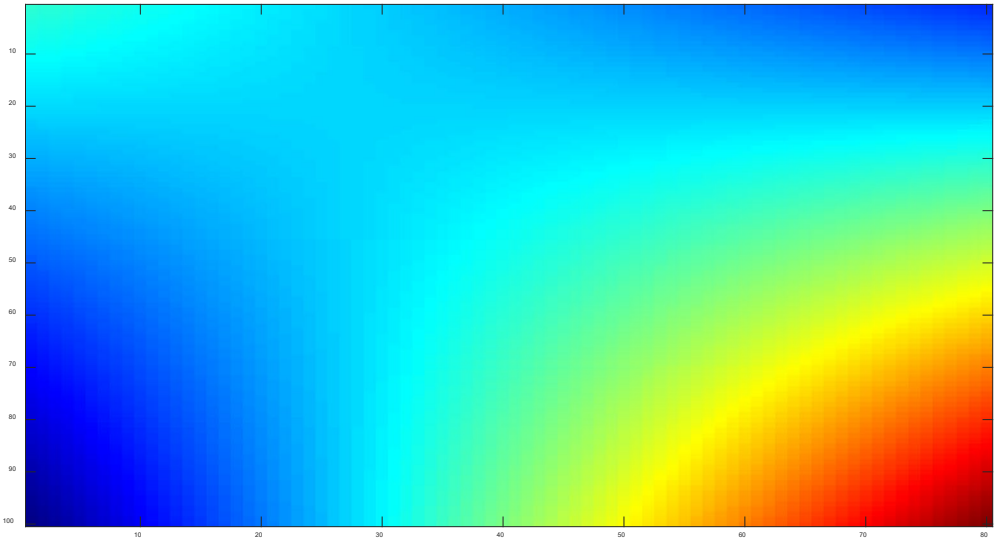
$$\|\log(xy^T)\|$$

$x$: $(1, 4)$

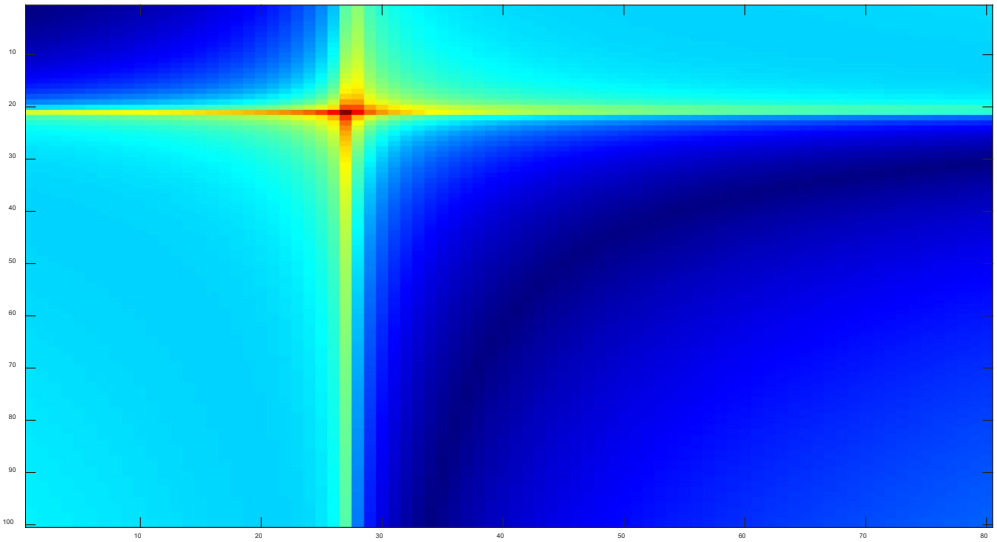
$y$: $(-1, 2)$

## MATLAB

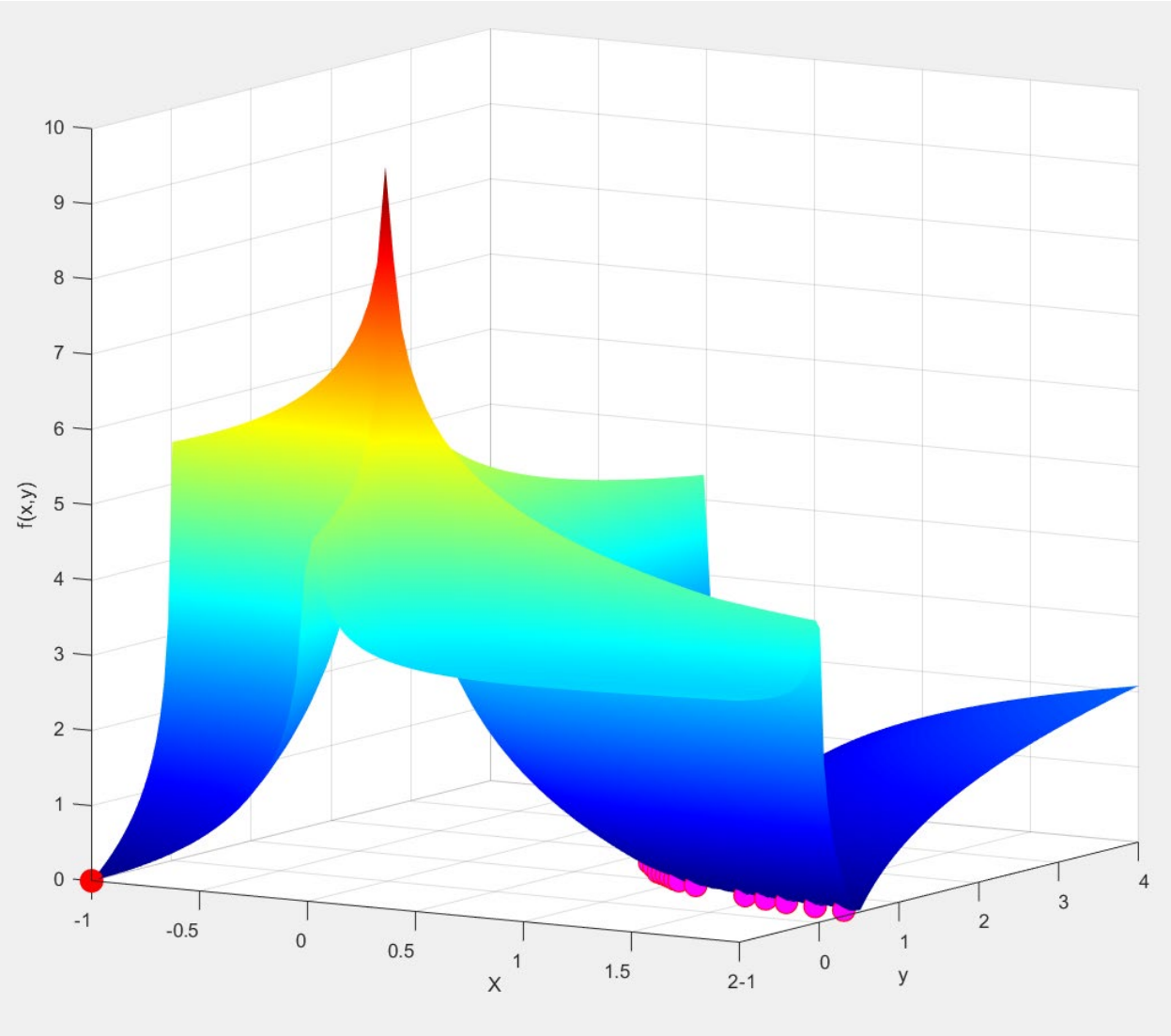
Code: masterMATLAB_1340_negativeExtrema.m



```
% a neat-looking 2D function:
x   = linspace(-1,4,100);
y   = linspace(-1,2,80);
fxy = bsxfun(@times,x',y);
>> imagesc(fxy)
```



```
% a neat-looking 2D function:
x   = linspace(-1,4,100);
y   = linspace(-1,2,80);
fxy = abs(log( bsxfun(@times,x',y) ));
>> imagesc(fxy)
```




## 82. UNSOLVED: FIND RIDGES OF A 2D SURFACE

Plot green/black dots on the upper “ridges” of the landscape.

## 83. FIND LOCAL MAXIMA

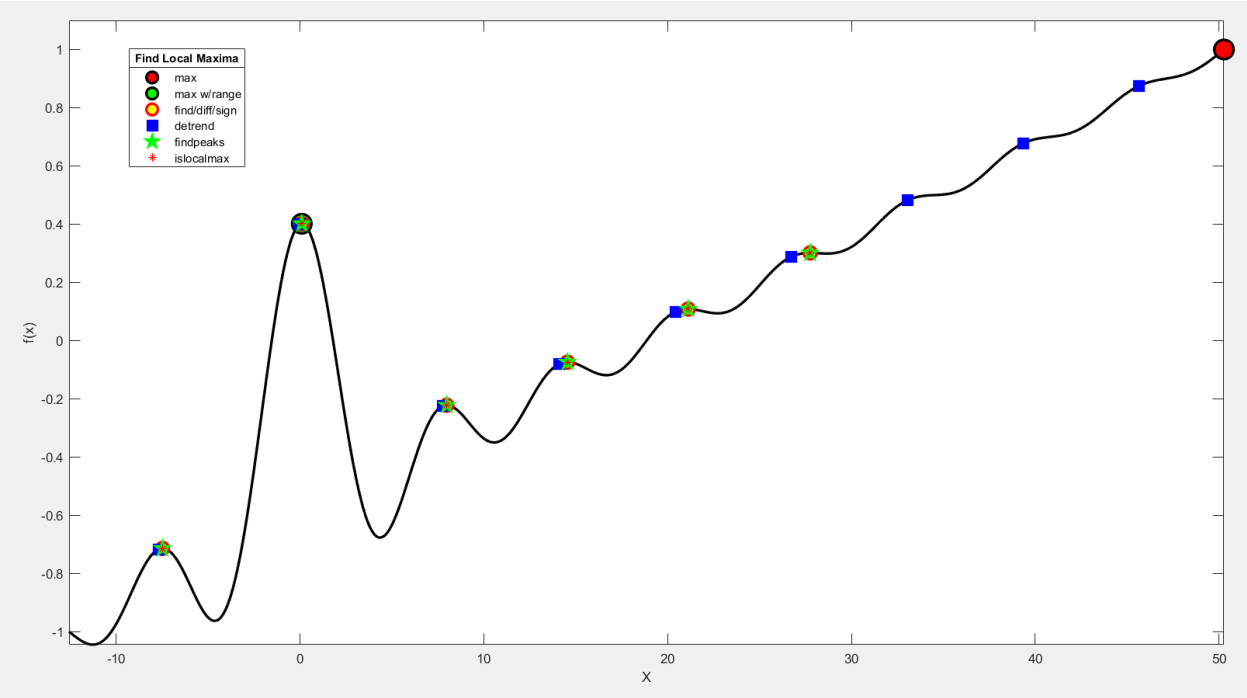
Find local maxima in a smooth function.

Understand the effects of detrending on local maxima identification.

Skills: max, dsearchn, diff, sign, detrend

### MATLAB

Code: masterMATLAB_1360_localMaxima.m




Note: Blue squares are offset due to the detrend function skewing peaks by a small amount.

## 84. REPLACE IMAGE PIXELS IN AND INTENSITY RANGE

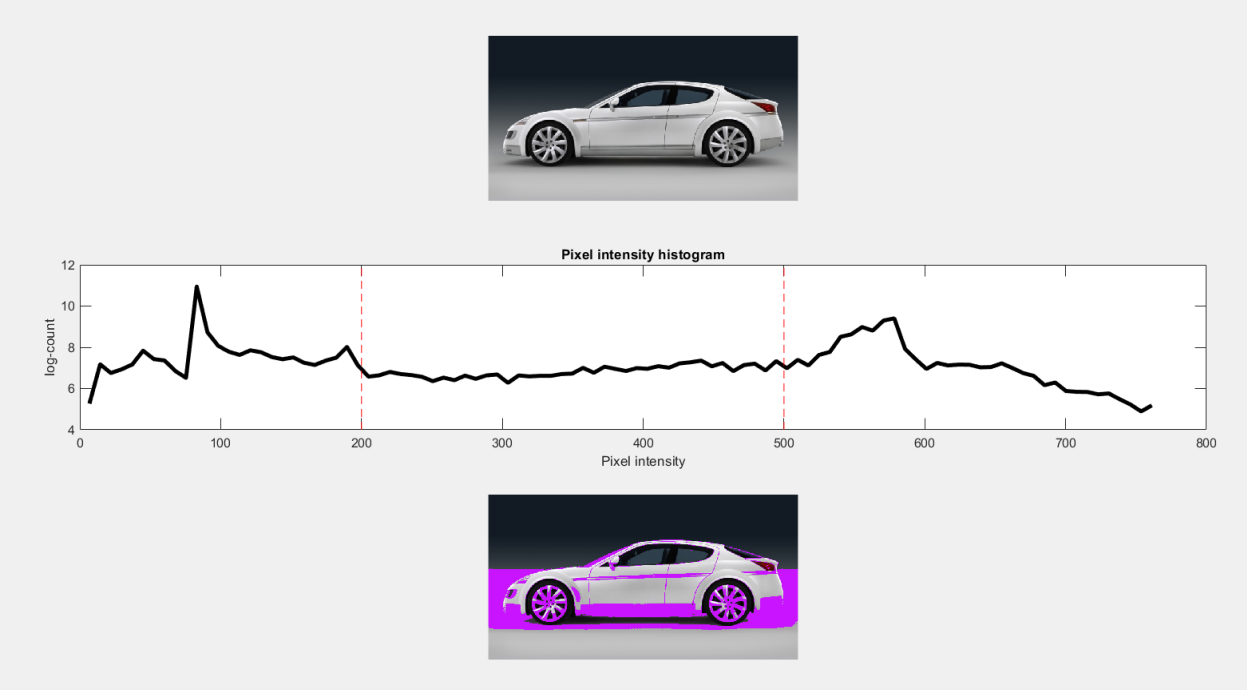
Find pixel values within a range, based on the pixel histogram, and replace those values with a different color.

Draw two lines in a plot using matrix inputs into the plot function.

Skills: imagesc, hist, repmat

### MATLAB

Code: masterMATLAB_1380_replacePixels.m




## 85. FIND SIGNAL CLIPPING POINTS

Generate a clipped signal based on a formula.

Mark the points in a signal that define the clipping boundary.

Skills: diff, mode, find

## Equation to generate clipping points

$$g(t) = \sin\left(2t - \frac{\pi}{2}\right)$$

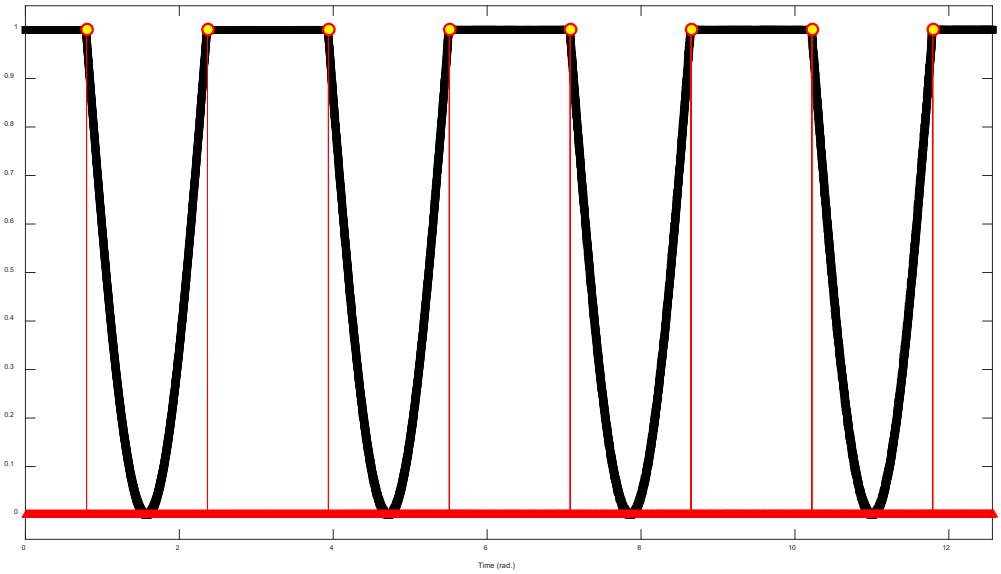
$$f(t) = \begin{cases} 1 - g & g > 0 \\ 1 & g \leq 0 \end{cases}$$

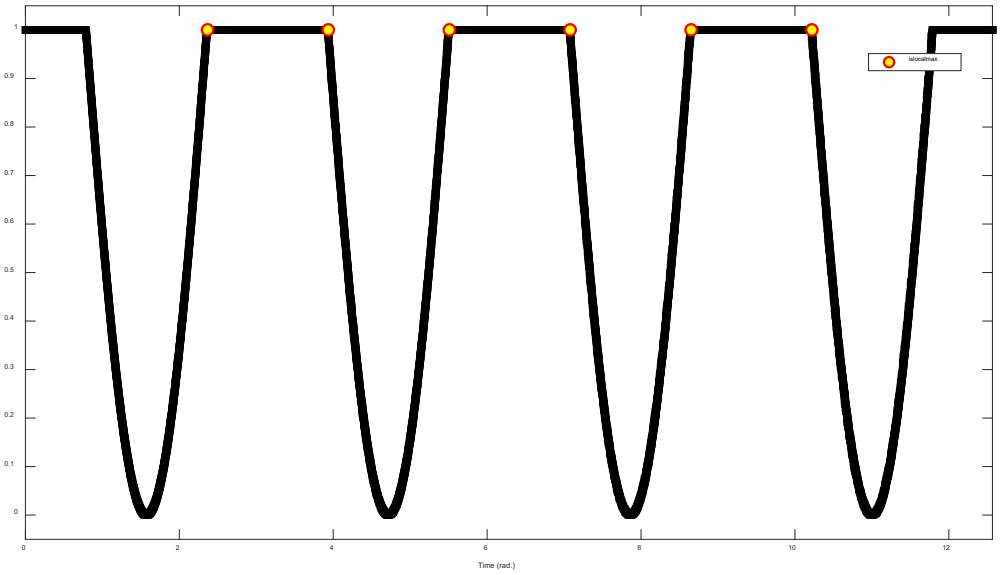
$$t \in (0, 4\pi)$$

### MATLAB

Code: masterMATLAB_1400_findClips.m

Note: Code is not a one size fits all solution. Often in signal processing, the specifics of the problem dictate changes required to find the solution such as the resolution of the data, etc…




## SECTION 15: CALCULUS AND DIFFERENTIAL EQUATIONS

### 86. PLOTTING SYMBOLIC FUNCTIONS

Plot several functions using *ezplot*, *fplot*, *fimplicit3*, and *ezpolar*.

Skills: ezplot, fplot, fimplicit3, ezpolar

Function to plot:

$$f(x) = 10\sin(x) + \frac{\tan(\frac{x}{2})}{10}$$

Implicit function to plot:

$$f(r,s,t) = 2^{ts} + rs - ts^{1/3}$$

Cardioid Polar function:

$$f(\phi) = 2(1 - \cos(\phi))$$

Note: these utilities plot formulas not data

## MATLAB

Code: MasterMATLAb_1420_plottingSymbolic.m

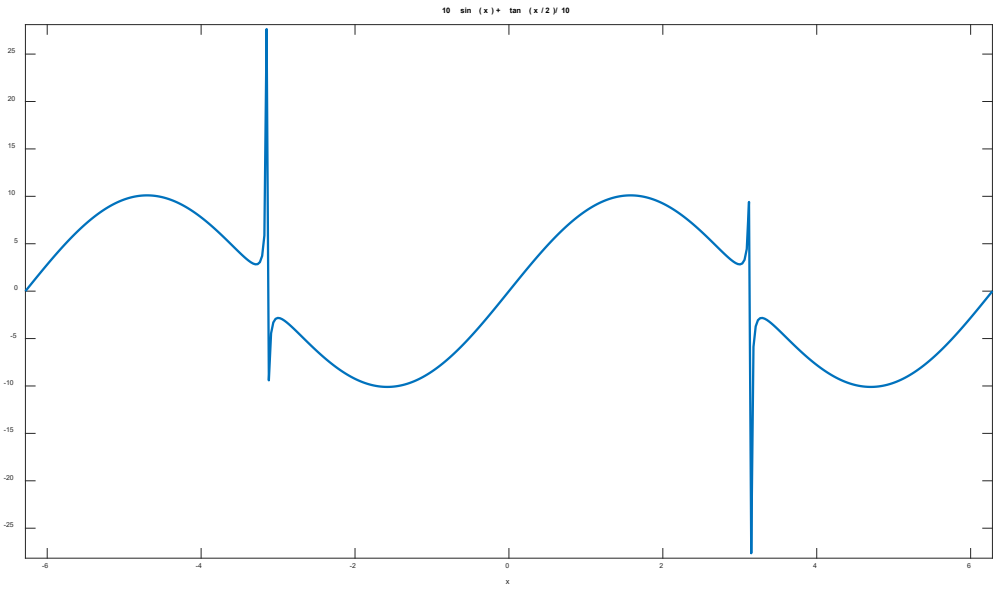


Figure 1: ezplot(fx)

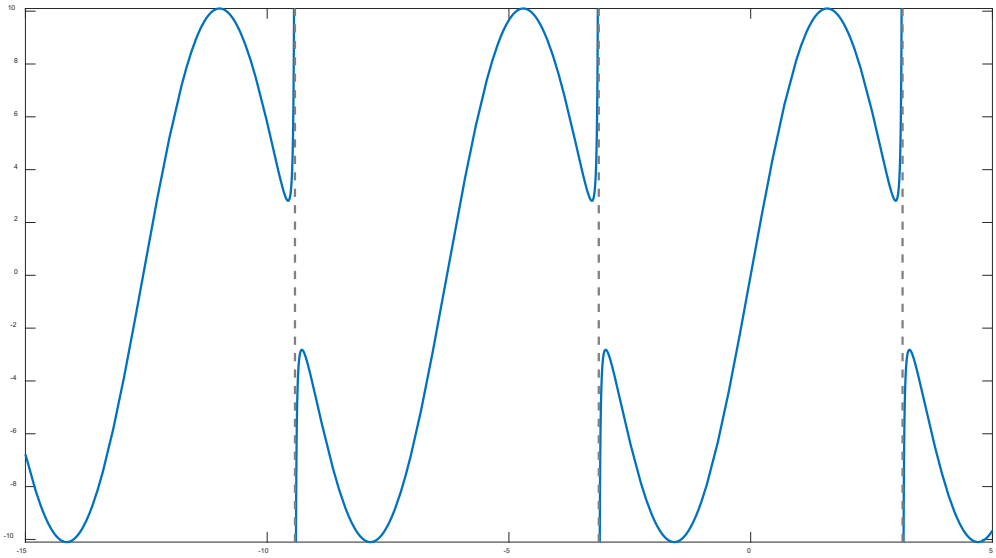



Figure 2: fplot(fy,[-15 5])

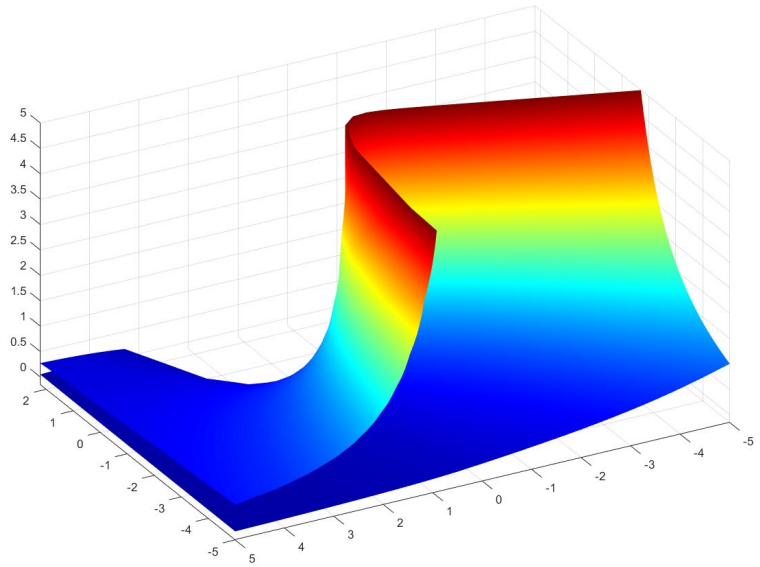



Figure 3: fimplicit3(fts)

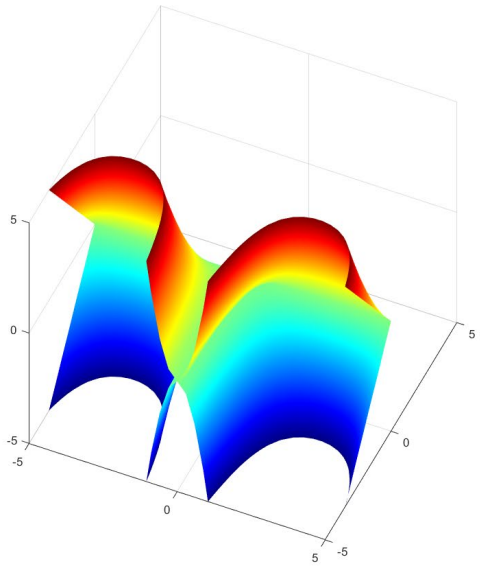



Figure 4: fimplicit3(fts)

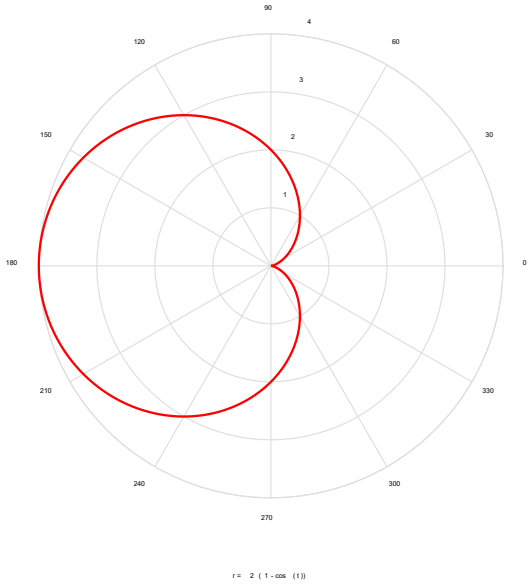


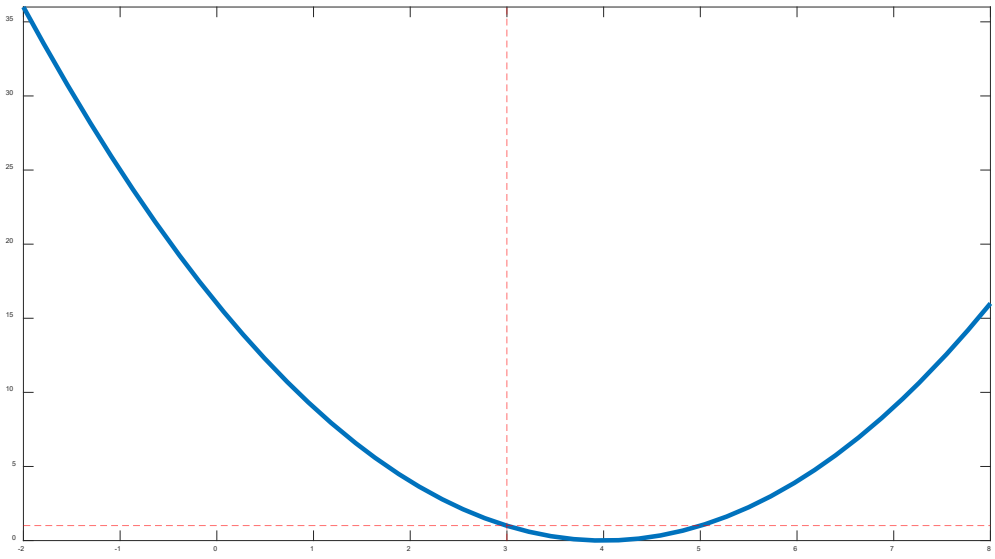

Figure 5: ezpolar(cardioid)

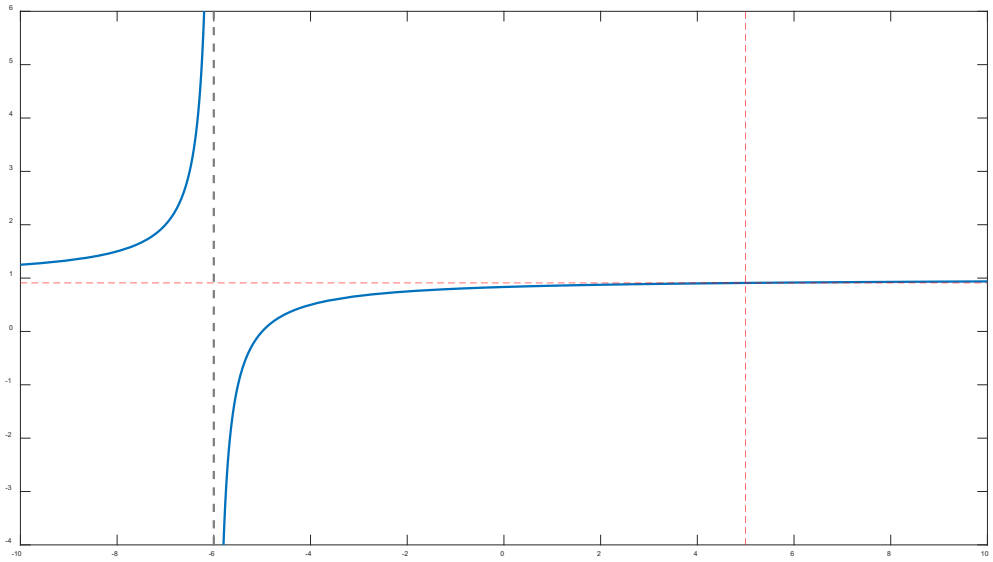
Compute the limits of several functions as they approach interesting values.

Skills: syms (requires symbolic toolbox), limit, fplot

## MATLAB

Code: MasterMATLAb_1440_limits.m





```
% determine what happens as f(x) approaches x=-6 from left and right
limit(fx,x,-6,'left')
limit(fx,x,-6,'right')
ans = Inf
ans =-Inf
```

## 88. FUNCTION DERIVATIVES

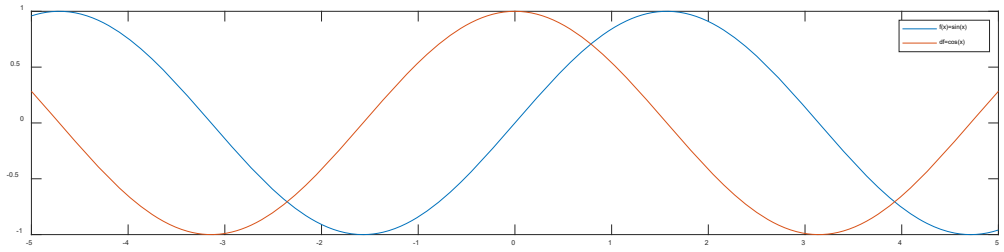
Compute the derivative of trig functions. Compute the value of the function and its derivative at a specific value.

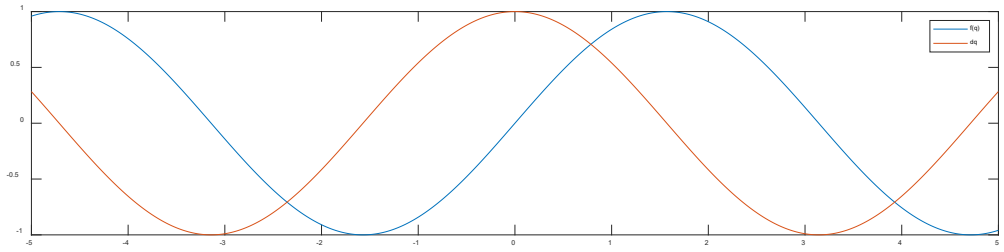
Understand the difference between symbolic and numeric math in MATLAB.

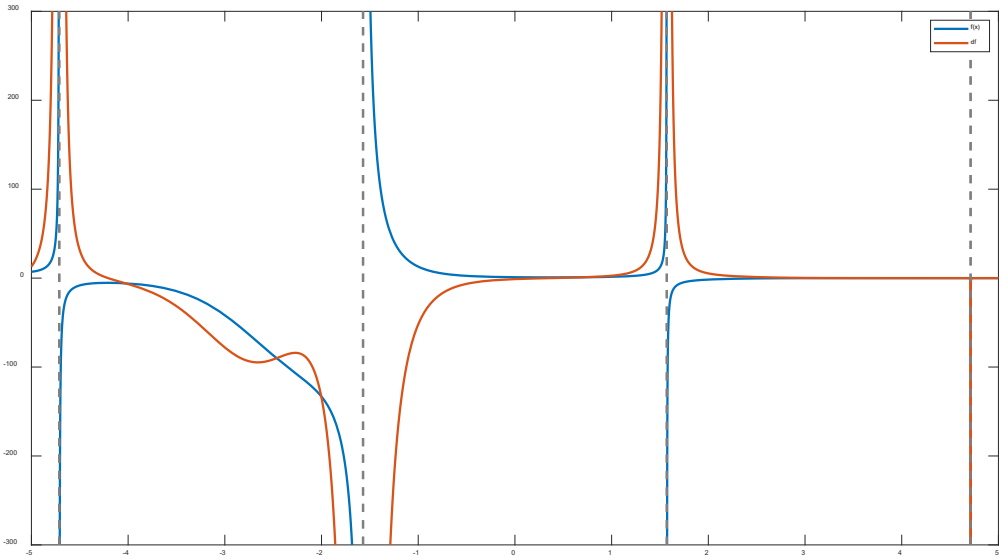
Skills: syms, diff, subs, pretty

### MATLAB

Code: MasterMATLAb_1460_derivatives.m



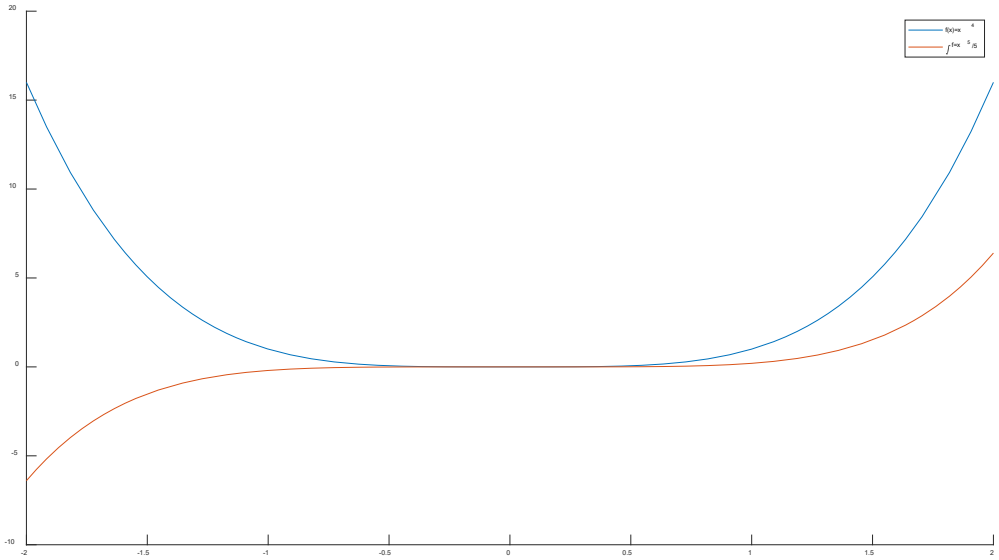

## 89. FUNCTION INTEGRATION

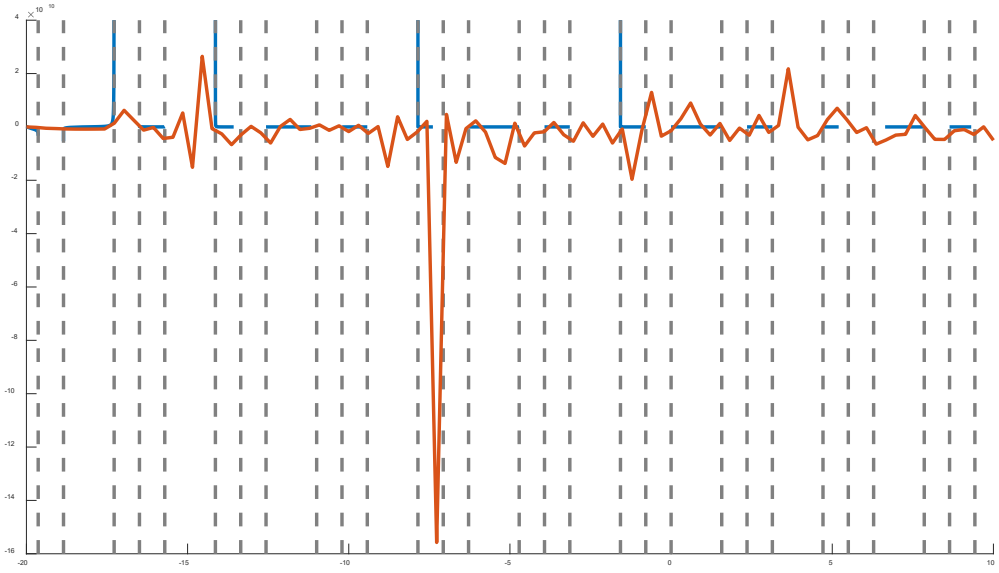
Compute the indefinite integral (antiderivative) of easy and “hard” functions (“hard” meaning no closed-form solution). Understand how to combine symbolic and numeric solutions in the same plot.

Skills: syms, int, integral, fplot, vpa (variable precision arithmetic), subs (substitute)

## MATLAB

Code: MasterMATLAb_1480_integration.m





## 90. SOLVING DIFFERENTIAL EQUATIONS

Solve a simple differential equation, and plot several solutions given different initial values.

Plot slope curves underneath the solution curves.

Skills: syms, diff, meshgrid, ezplot, quiver, dsolve

## Find solution to following initial value equation:

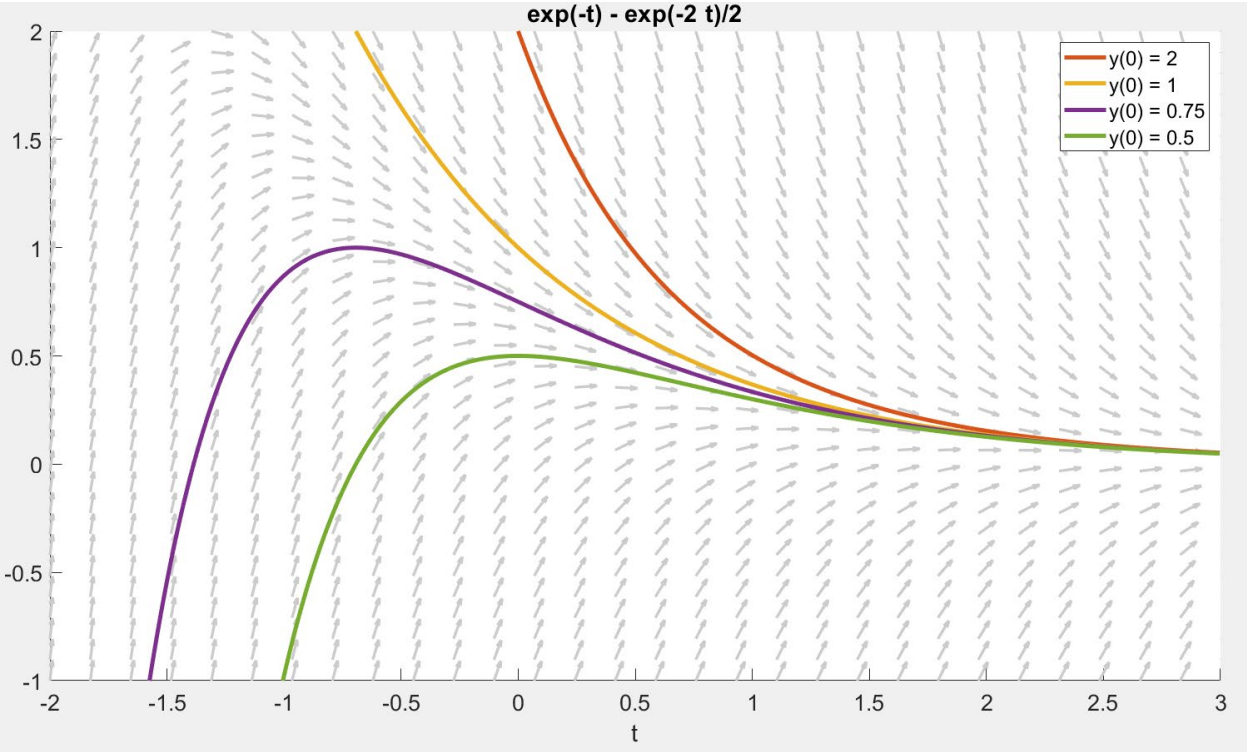
$$\frac{dy}{dt} = e^{-1} - 2y$$

Where:

$$y(0) = \{2, 1, .75, .5\}$$

### MATLAB

Code: MasterMATLAb_1500_diffEQ.m




## SECTION 16: CLEANING UNIVARIATE TIME SERIES

## 91. RUNNING MEAN FILTER

Generate a noisy signal and implement a mean-smoothing filter on that signal.

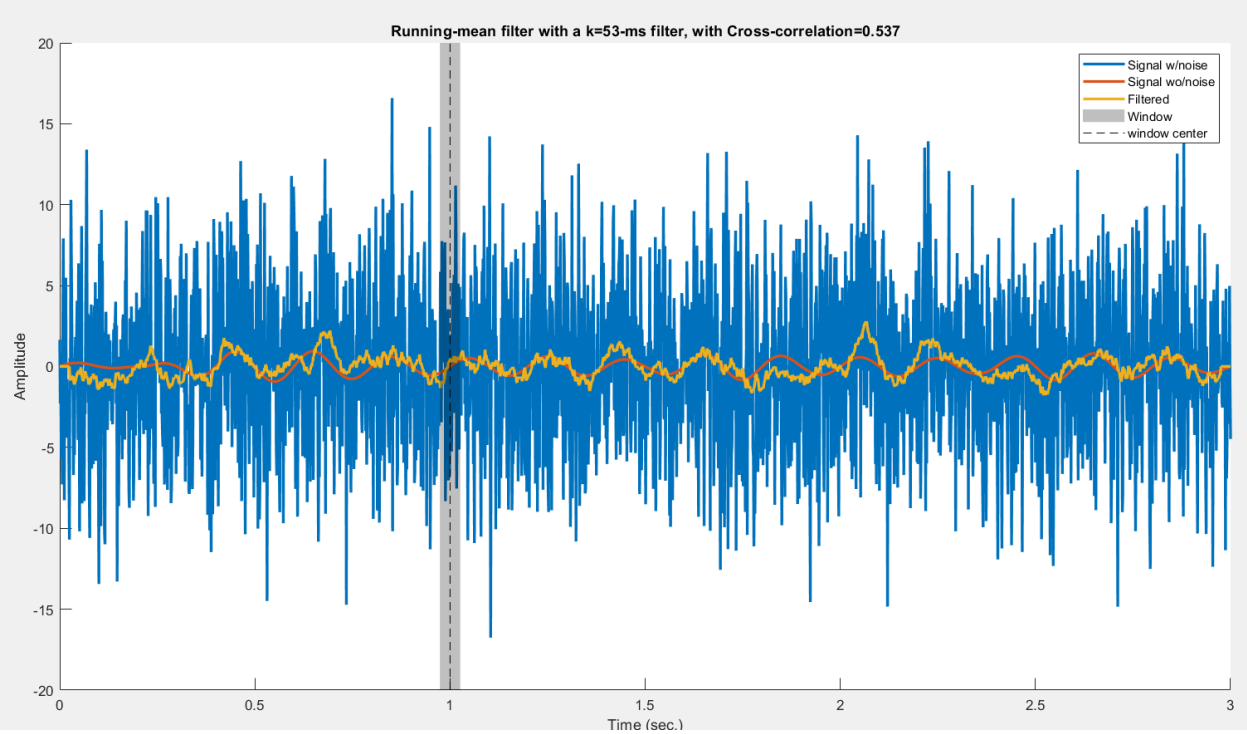
Draw a patch on the time series to illustrate the size of the time window used in the filter.

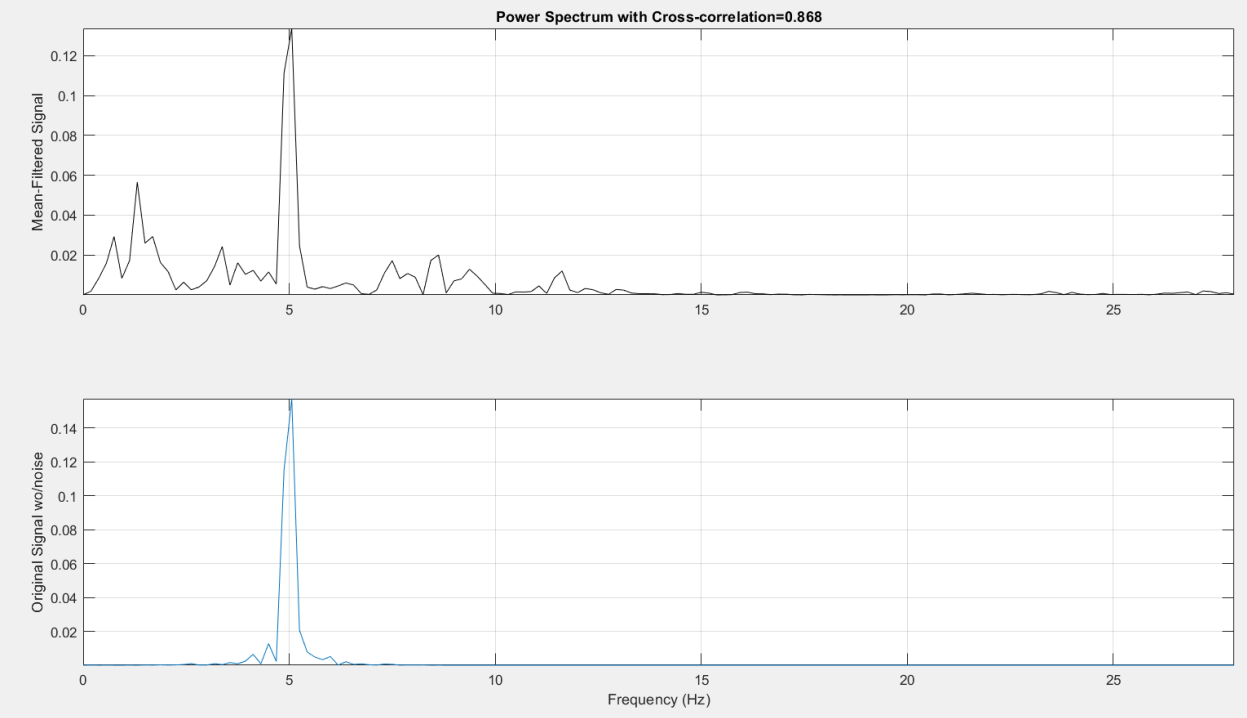
Skills: interp1, mean, for, patch, dsearchn

### MATLAB

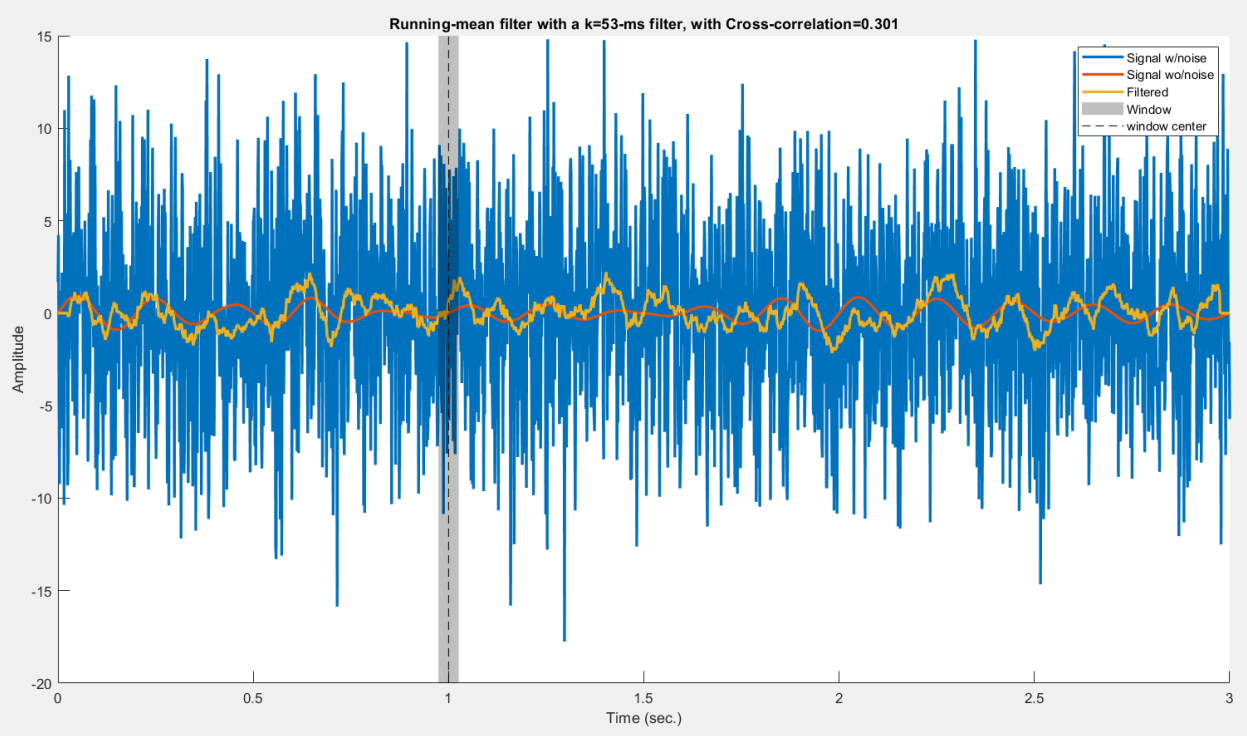
Code: MasterMATLAB_1520_runningMean.m

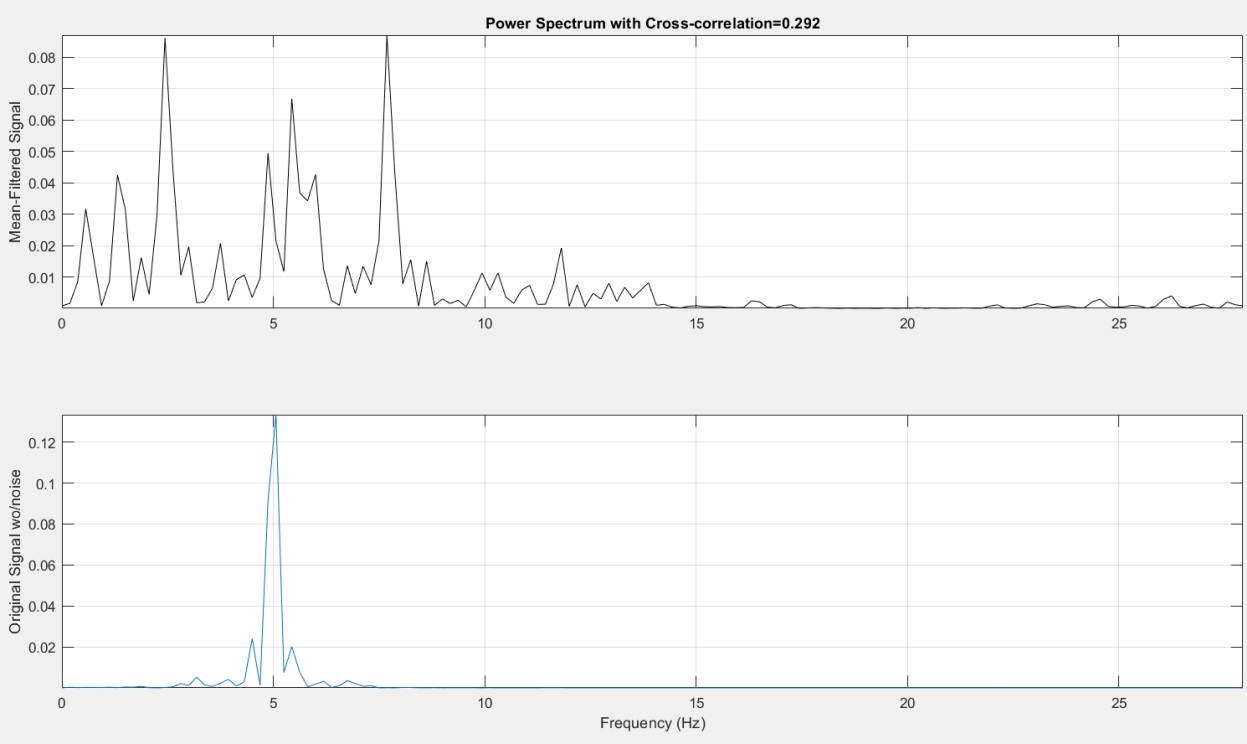
Test 1: Power Spectrum shows some similarity






Test 2: Power Spectrum not very similar






## 92. THRESHOLD MEDIAN FILTER

Generate a signal with spike-noise and implement a media filter to remove the artifacts.

Derive a data-driven threshold based on the derivative of the sorted data values.

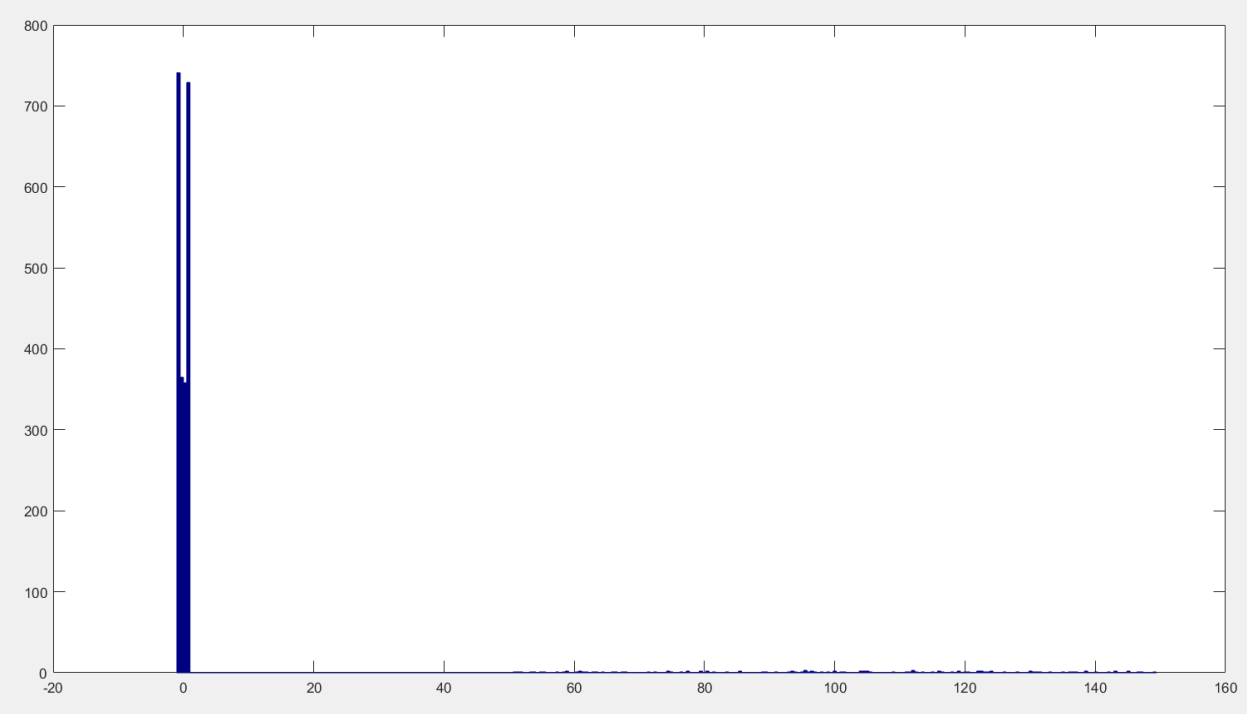
Skills: histogram, sin, sort, diff, median, find, randperm

*randperm(n)* returns a row vector containing a random permutation of the integers from 1 to n without repeating elements.

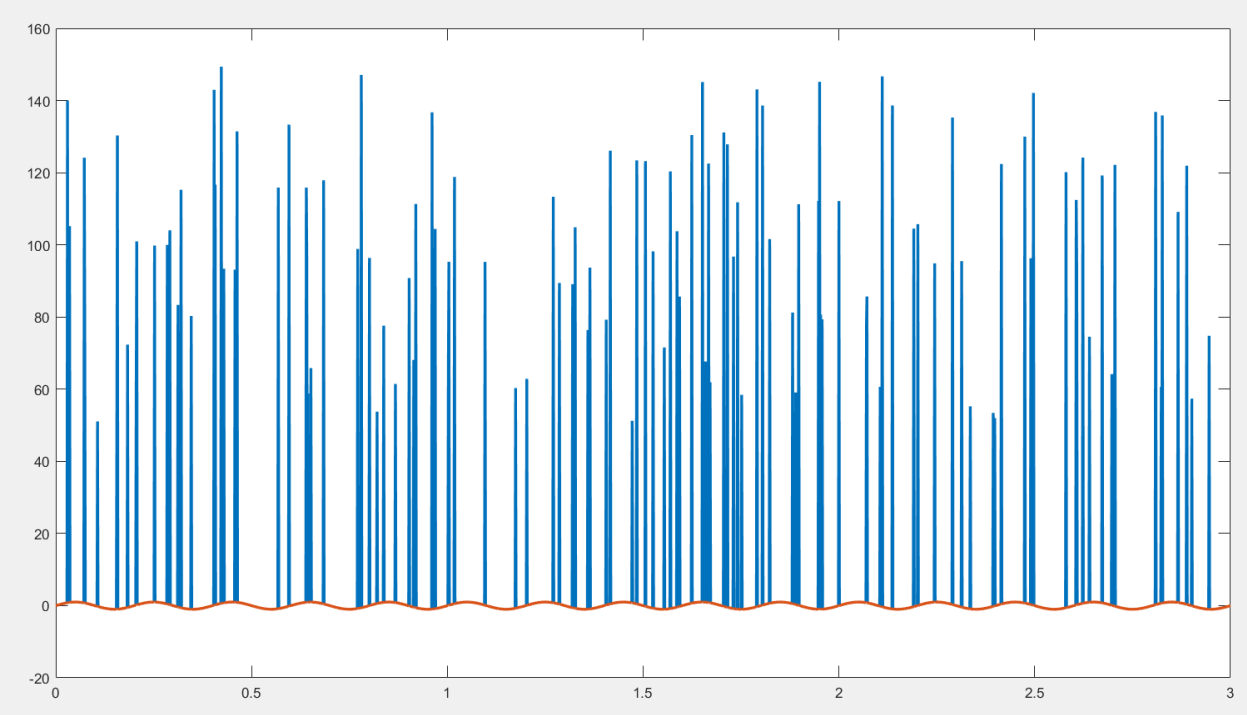
### MATLAB

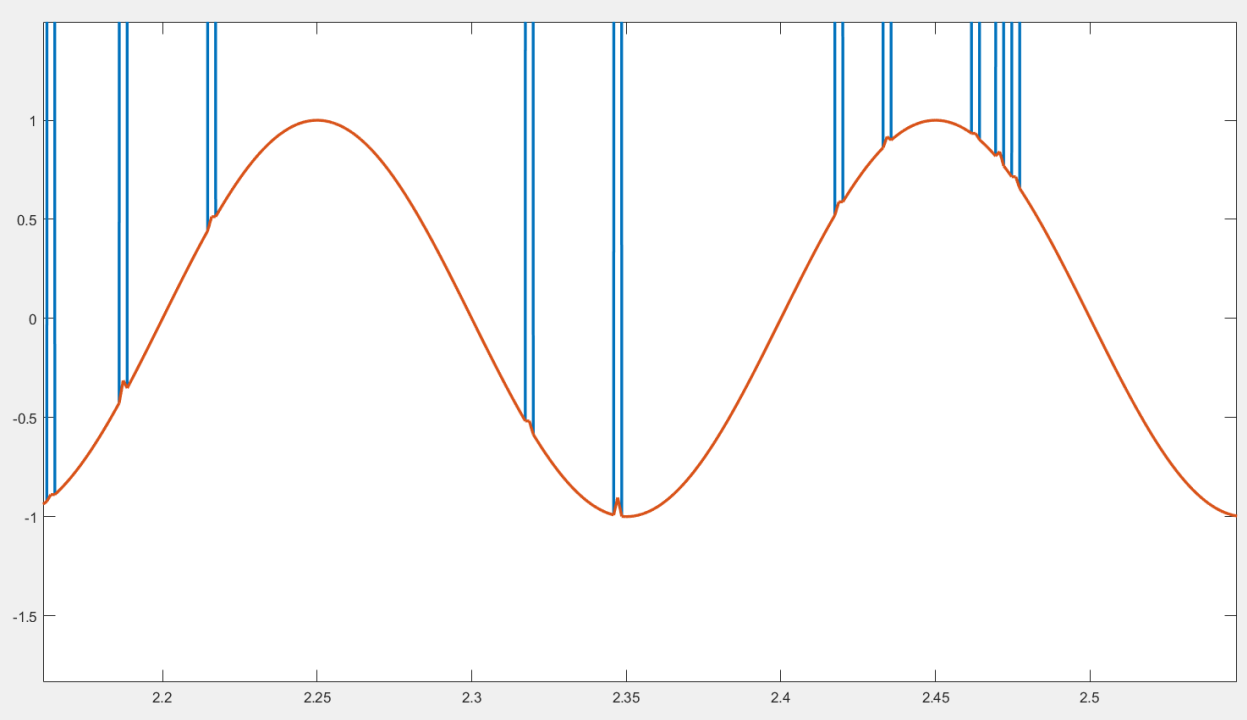
Code: MasterMATLAB_1540_thresholdMedian.m

Use Histogram to visually inspect threshold value




Orange line shows signal after threshold median filtering



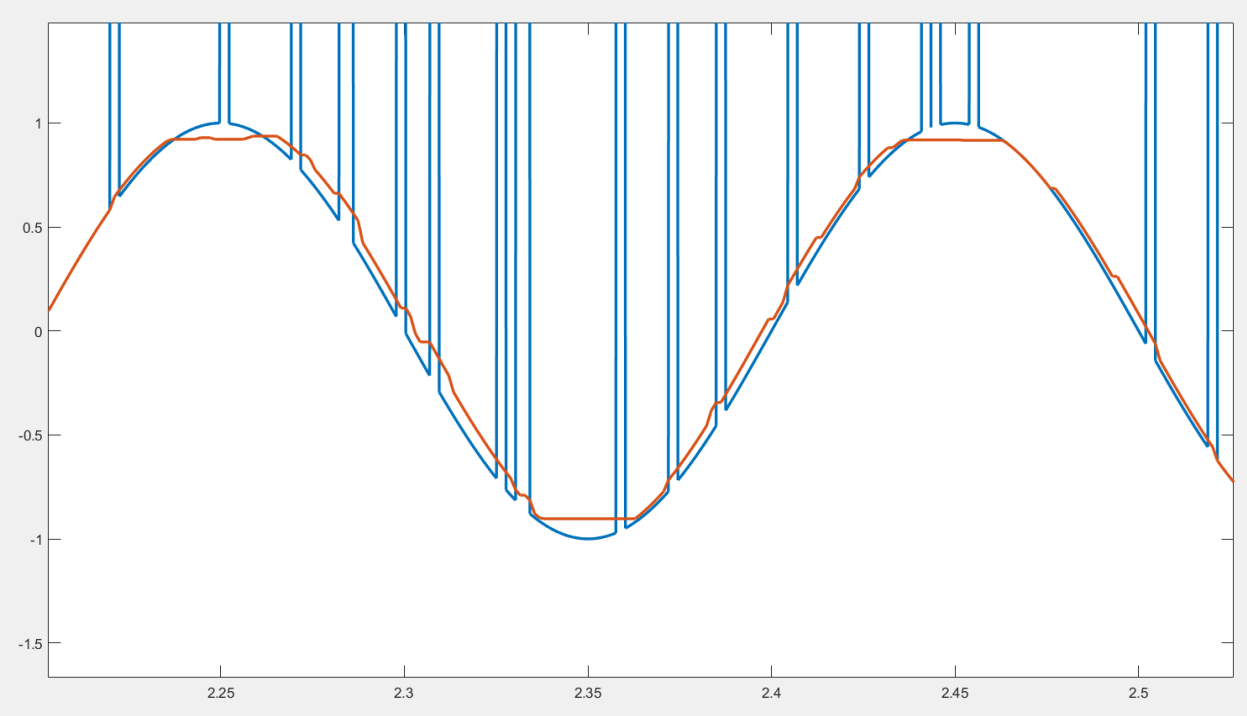



## 93. SOLVED: ALL-POINTS MEDIAN FILTER

Repeat the median filter exercise, but compute the median for each time point instead of each suprathreshold time point.

### MATLAB

Code: Unknown_Median_Threshold.m




## 94. INTERPOLATE MISSING TIME POINTS

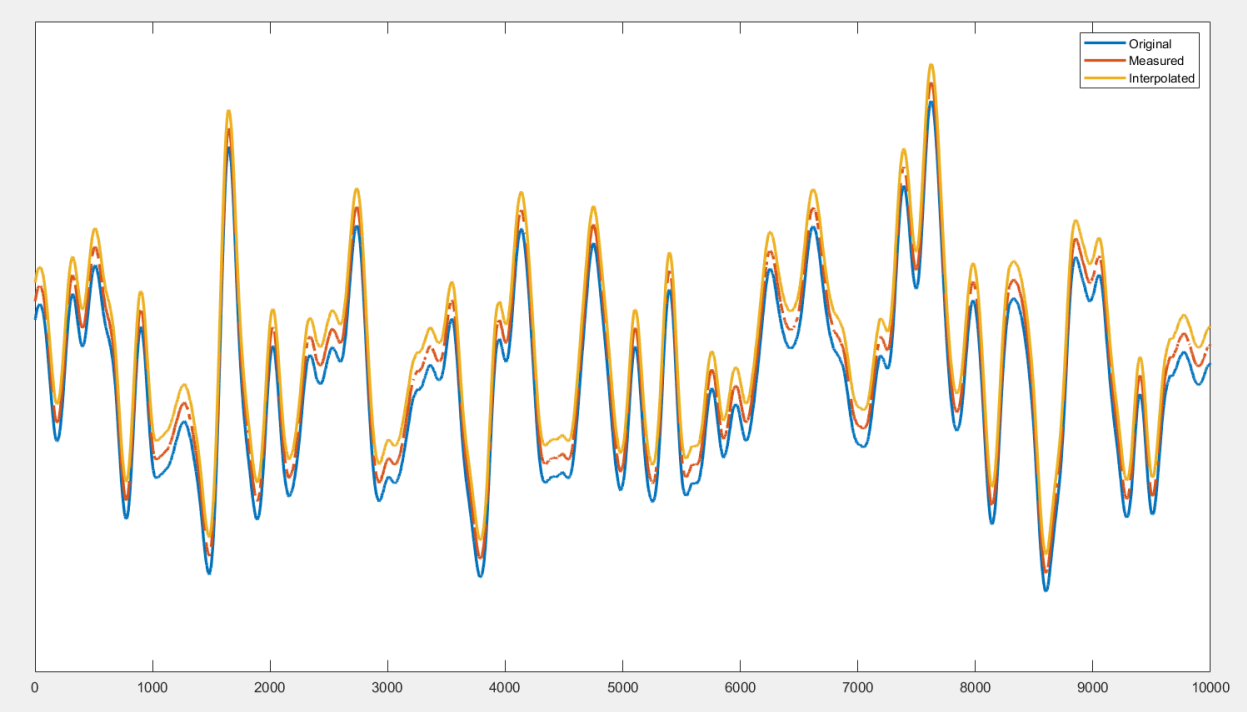
Identify missing points from a time series. Use linear interpolation to fill in the missing points.

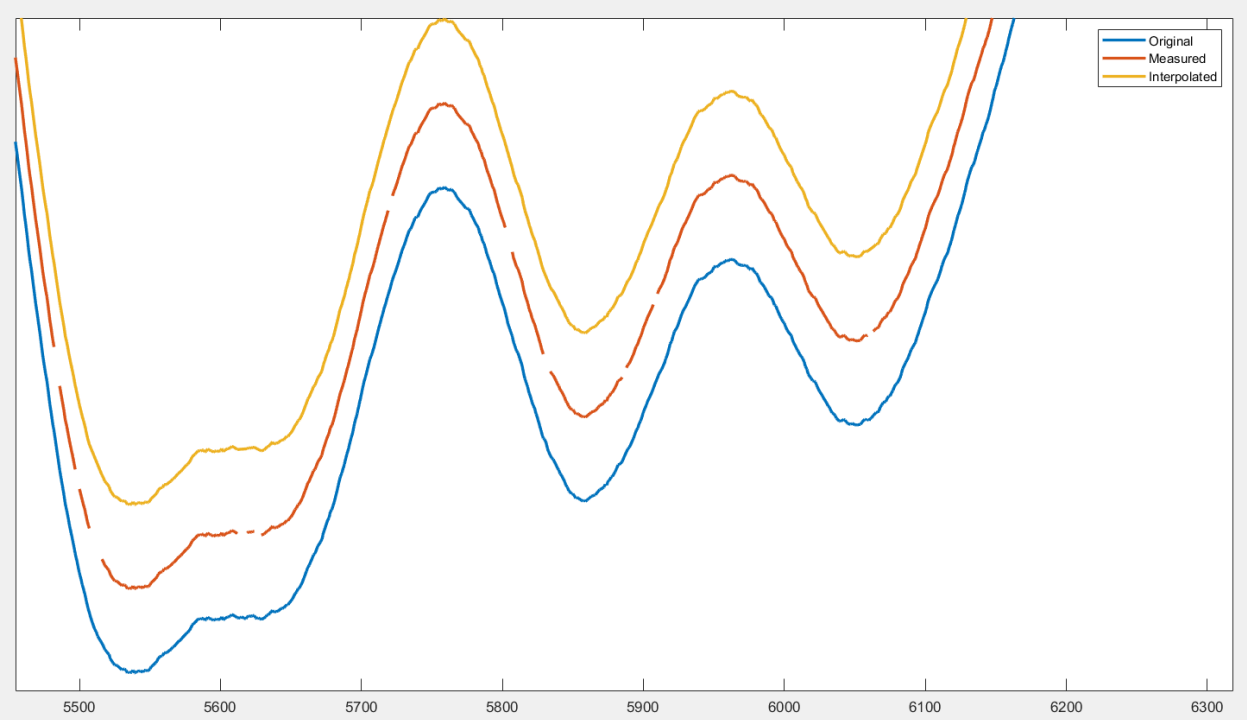
Plot all three time courses (original, measured, interpolated) with y-axis offsets using one line of code.

Skills: exp, conv, deal, cellfun, bwconncomp (image processing toolbox)

### MATLAB

Code: MasterMATLAB_1560_interpolate.m


Original
Measured
Interpolated
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000


Original
Measured
Interpolated
5500
5600
5700
5800
5900
6000
6100
6200
6300

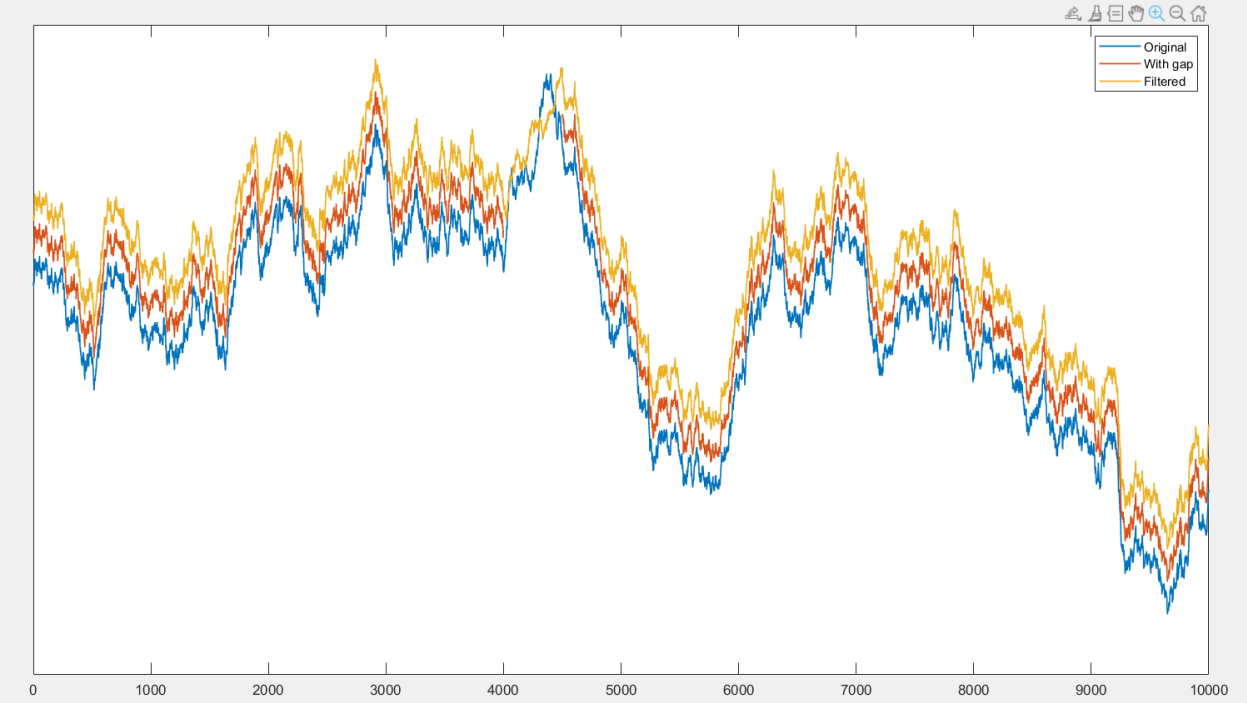
Cut a large chunk of data out of a time random series. Compute the FFT of the pre-cut and post-cut data, and interpolate the missing data by mixing the pre/post spectra.

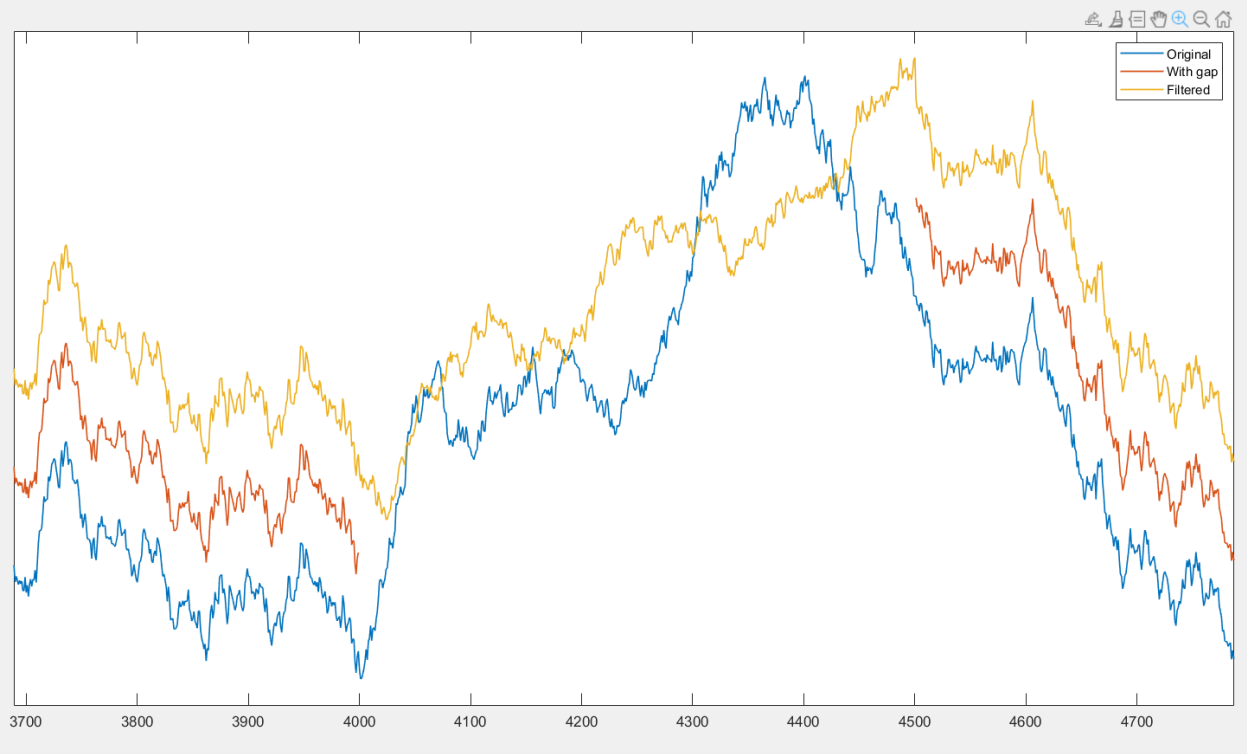
Make sure there are no sharp edges at the boundaries of the missing data.

Skills: cumsum, deal, fft, detrend, linspace

## MATLAB

Code: MasterMATLAB_1580_spectralInterp.m






## 96. POLYNOMIAL FITTING TO REMOVE DRIFTS

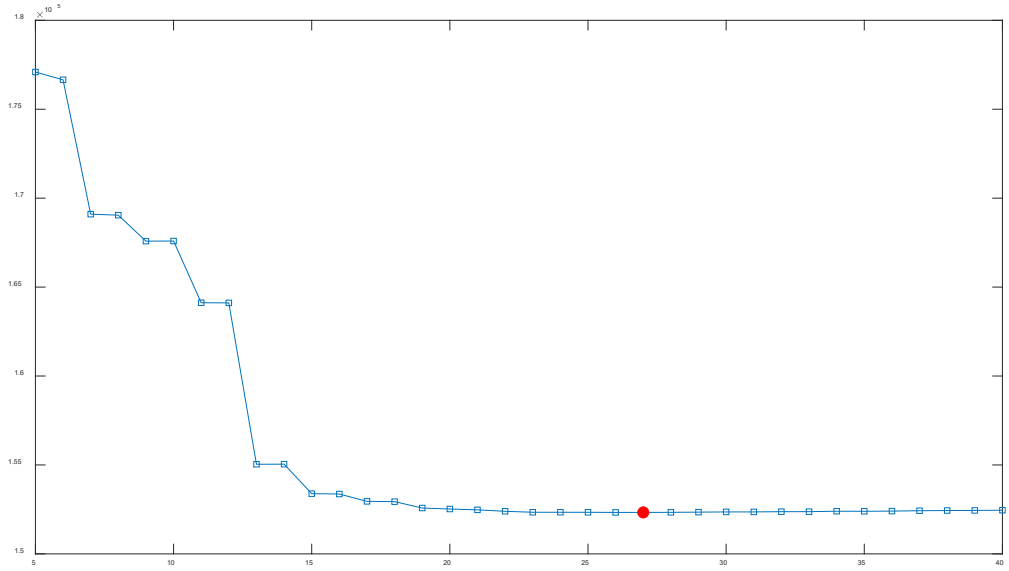
Generate data that contains slow drifts and high-frequency activity. Use polynomial fitting (order=25) to model and remove the slow drift.

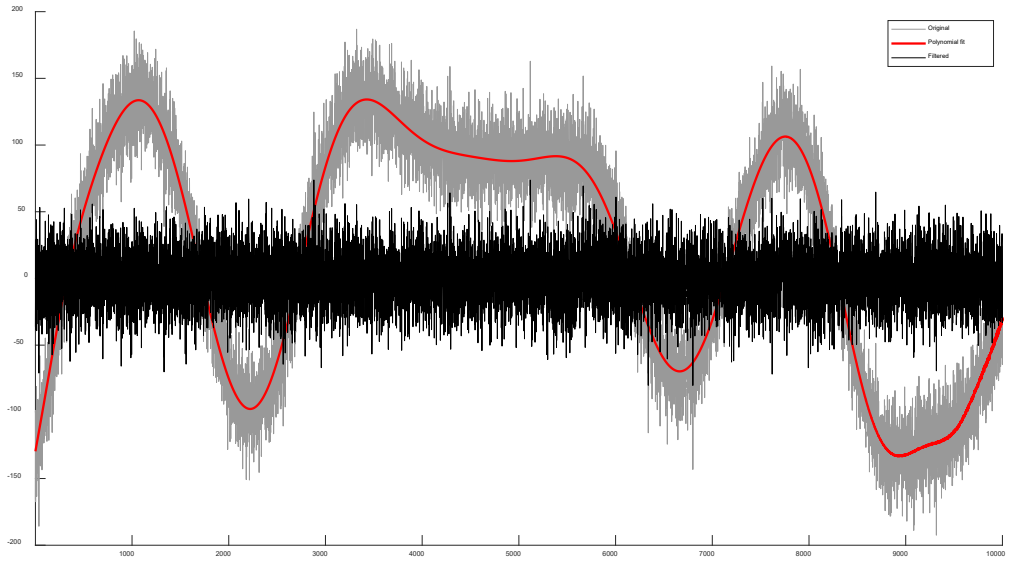
Use the Bayes information criteria to determine the optimal model order in the range of 5 and 40.

Skills: interp1, polyval, log

### MATLAB

Code: MasterMATLAB_1600_polynomial.m




## 97. UNSOLVED: POLYNOMIAL FITTING TO ISOLATE DRIFTS

Use polynomial fitting to keep the slow drift and remove the noise.

Skills: Sinc function, circshift (to find roots of a function), Bayes Information Criteria (to find order of a polynomial)

## 98. SOLVED: LOCAL MAXIMIMA IN NOISY DATA

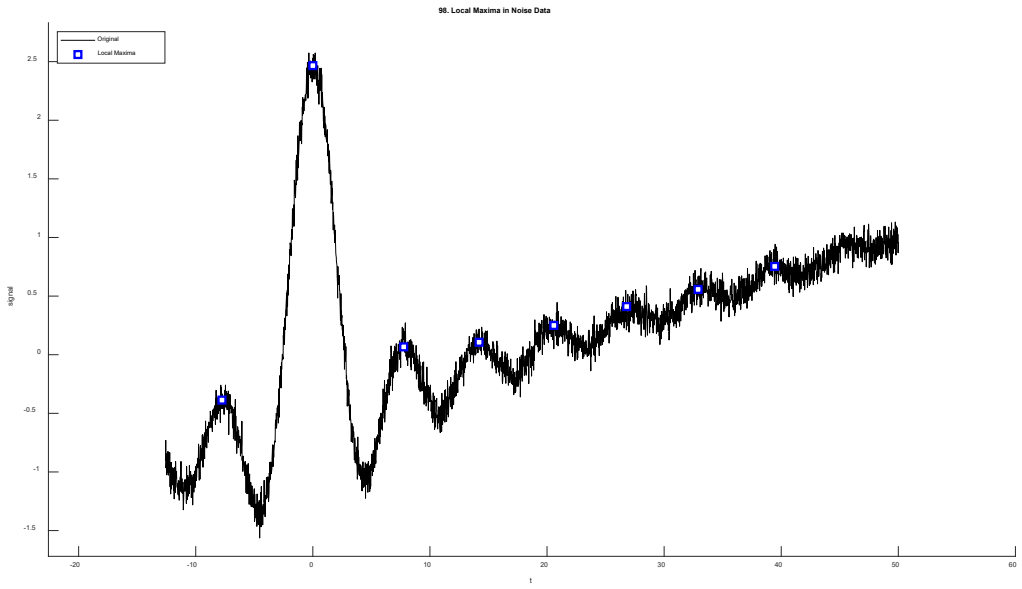
Find a way to solve this problem.

Skills: sinc function, anonymous function, circshift, islocalmax

### MATLAB

Code: solved_local_maxima_in_noisy_data2.m

Image: unknown_local_maxima_in_noise_data.JPG




# SECTION 17: CLEANING MULTIVARIATE TIME SERIES

## 99. THREE WAYS TO MAKE A COVARIANCE MATRIX

Compute a channel-by-channel covariance matrix using loops, direct matrix algebra, and the MATLAB *cov* function.

Use the *eval* function to display the three matrices in a loop.

Skills: for, cov, eval

Covariance matrix is a matrix of all of the relationships (covariations) between all possible pairs of channels (or variables). Basis for analysis steps pre-processing signal processing, machine learning, clustering.

## Equation:

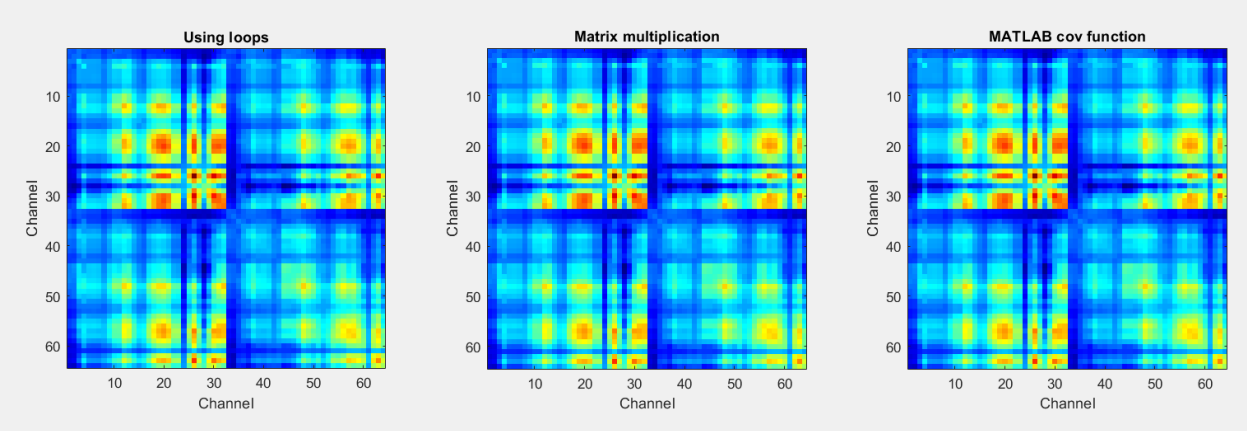
$$c_{x,y} = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y}) = x^T y$$

$$C = \frac{1}{n-1}X^T X$$

Note: matrices should be mean-centered for transpose shorthand to be used.

### MATLAB

Code: MasterMATLAB_1620_covariance.m



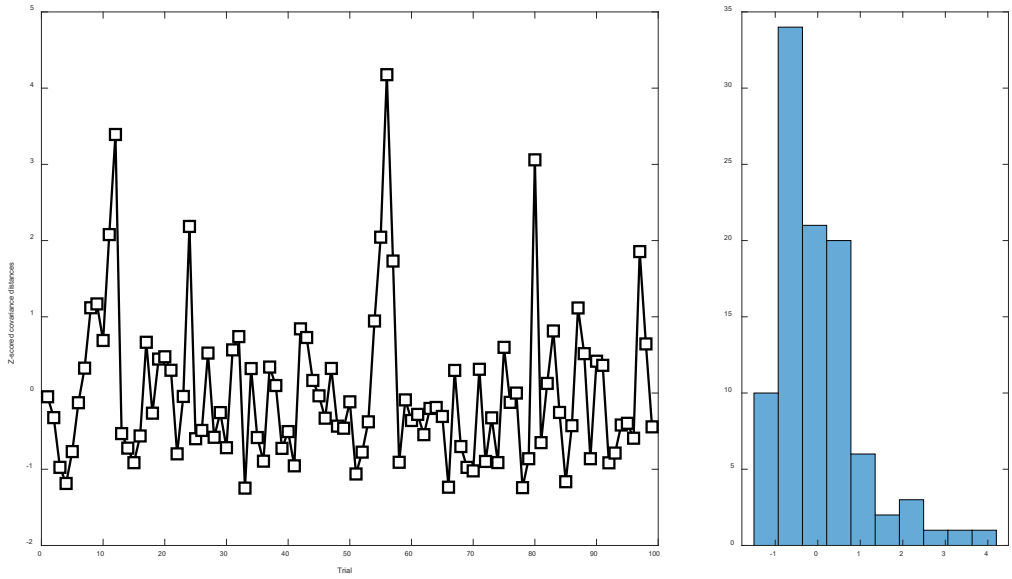

## 100. REJECT DATA BASED ON EXTREME COVARIANCE

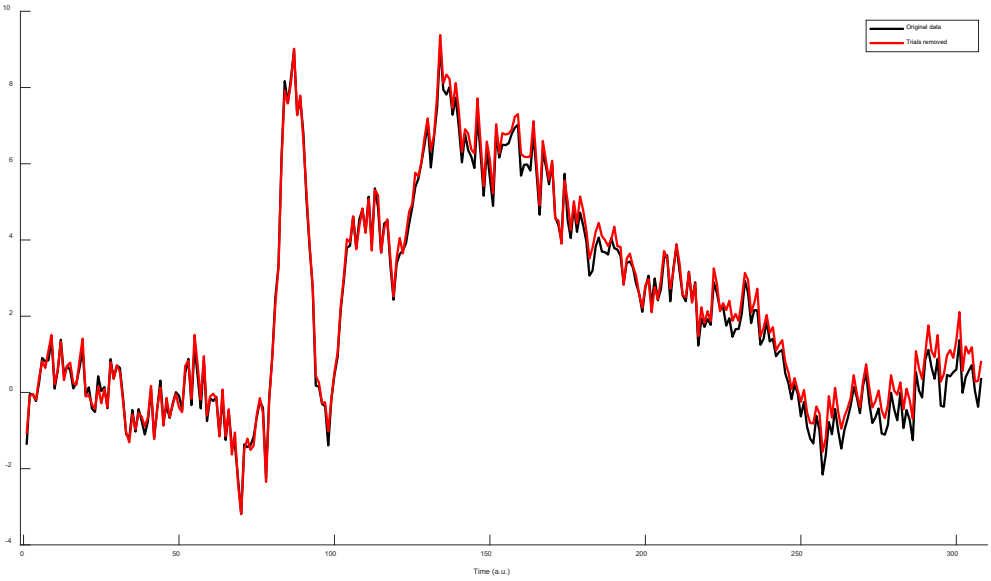
Compute the distance from single-trial covariance matrices to the grand-average covariance. Identify trials with extreme distances, and reject those trials from the dataset. Plot the average over trials for the original and cleaned data.

Skills: cov, sqrt, zscore, histogram

## MATLAB

Code: MasterMATLAB_1640_rejectCovar.m






## 101. EFFECTS OF AVERAGING ON COVARIANCE MATRICES

Generate correlated multivariate data by channel-modulated sine waves plus noise. Compute the covariance matrices in three ways:

1. Averaging in the time domain first, then computing covariances.
2. Computing covariances first, then averaging them together.
3. Covariance per trial, then averaging.

Use *eval* to image the covariance matrices in a double-loop.

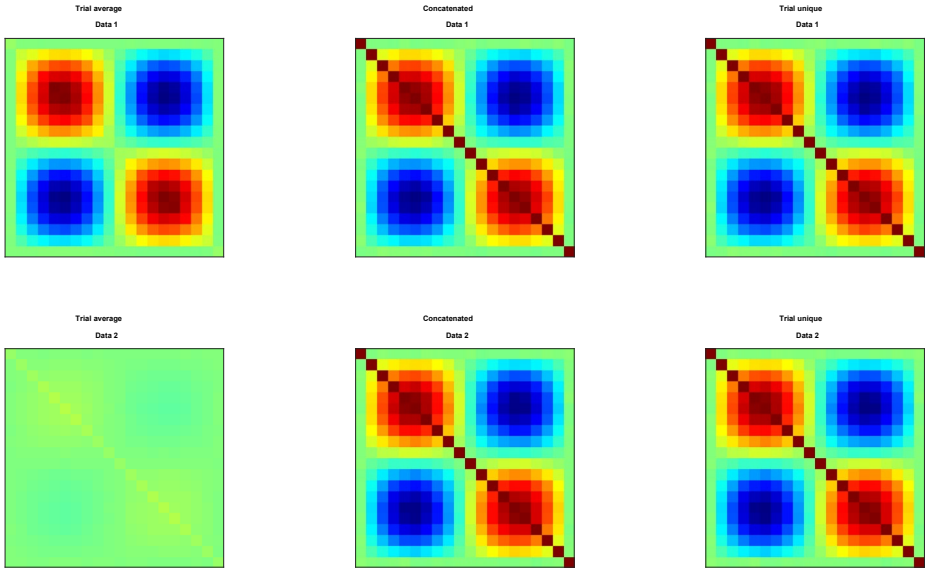
Skills: linspace, sin, bsxfun, eval, repmat, reshape, squeeze (used during plotting in this example)

Covariance methods:

- Average trials, then covariance
- Concatenate trials, the covariance
- Covariance per trial, then average

## MATLAB

Code: MasterMATLAB_1641_averagingCovars.m




# 102. SIMULATE TRI-COMPONENT TIME-SPACE DATA

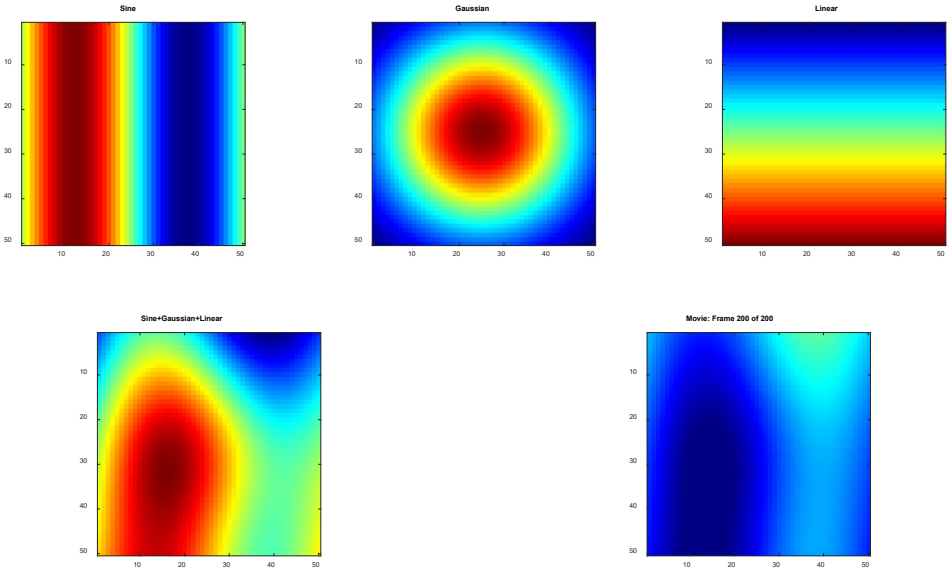
Generate time-space data by amplitude-modulating the sum of three spatial patterns: Sine wave, Gaussian, and linear trend.

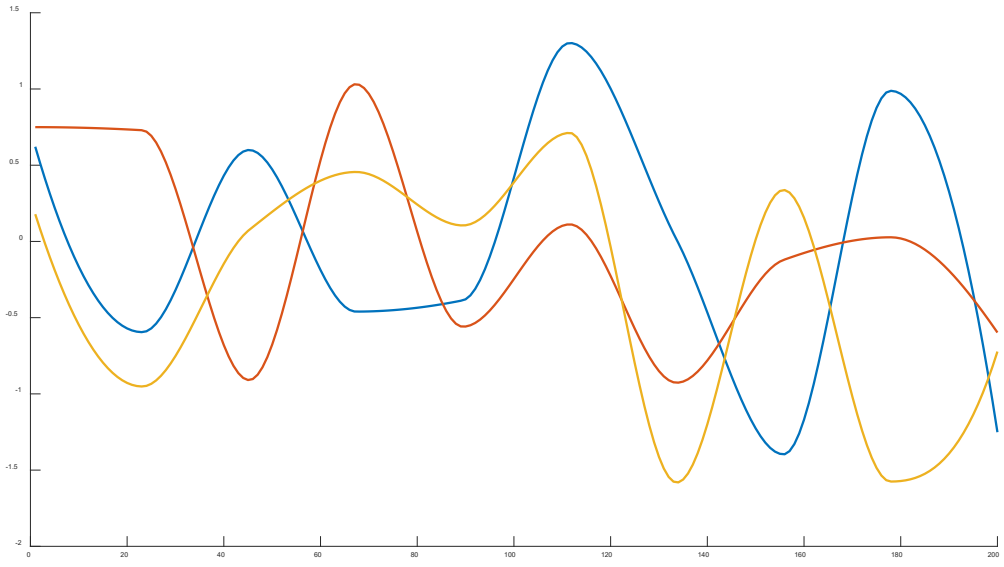
Use *set* to generate a movie of the time-space data and dynamically update the plot title on each frame.

Skills meshgrid, exp, set, interp1, squeeze

## MATLAB

Code: MasterMATLAB_1660_simulate3comp.m



## 103. UNSOLVED: TRI-COMPONENT DATA WITHOUT LOOPS

Generate the exact same time-space data as the previous video, but with NO LOOPS! (Loop only to display the movie.)

## 104. SPACE-BASED SINGLE CHANNEL INTERPOLATION

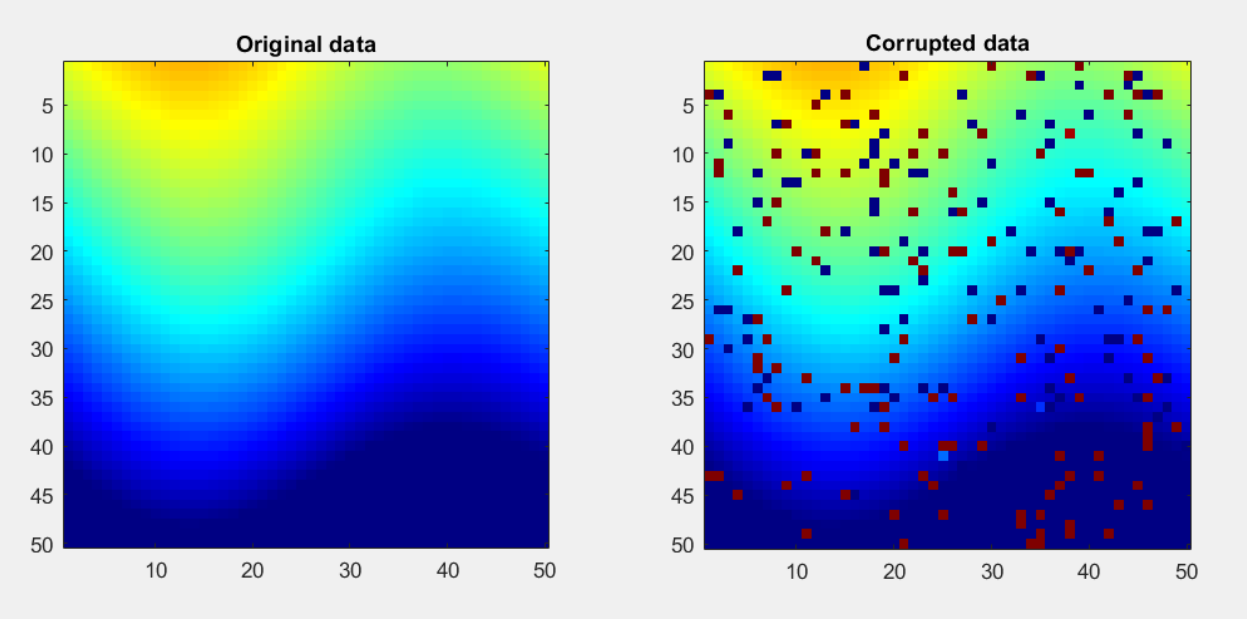
Corrupt random pixels in a space-time dataset. Use *scatteredInterpolant* to interpolate those missing data values.

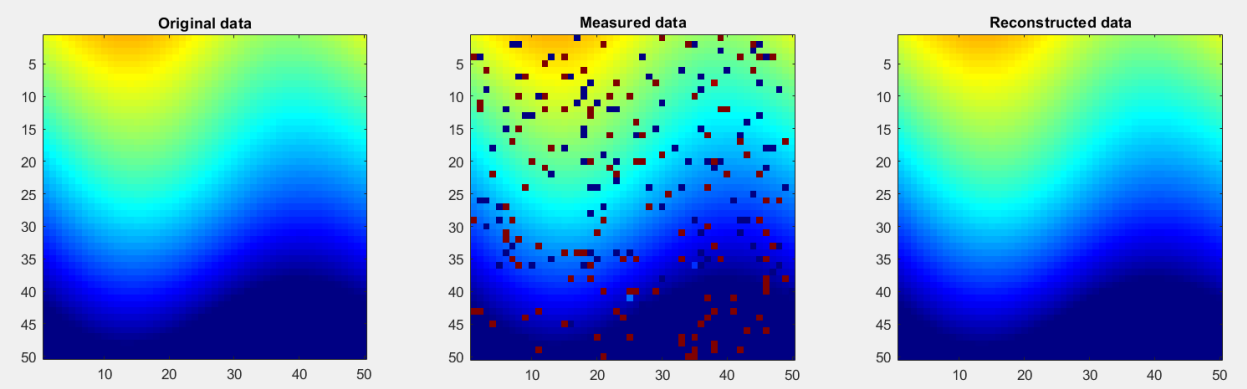
Use *set* to generate a movie of all three datasets (original, corrupted, interpolated).

Skills scatteredInterpolant, histogram, set, randperm

### MATLAB

Code: MasterMATLAB_1680_spaceInterp.m






## 105. SPATIAL SMOOTHING ON A GRID OF CHANNELS

Add random noise to the space-time dataset. Mean-smooth the data using a 2D raised kernel. Show a movie of the noisy and smoothed datasets.

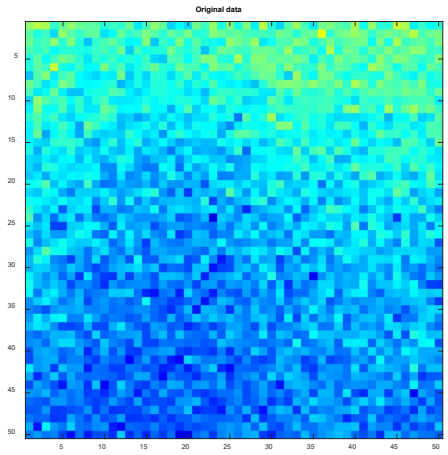
Use *conv2* to perform the same operation faster. Use *tic/toc* to determine how much faster *conv2* is.

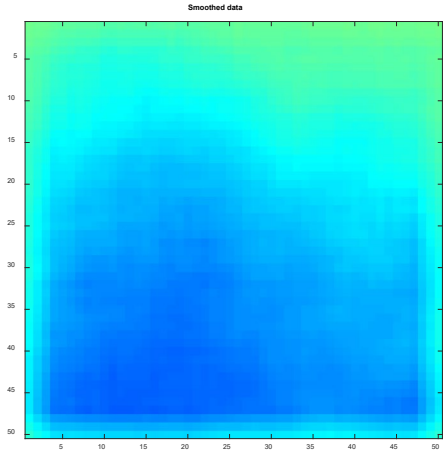
Skills: conv2, mean

### MATLAB

Code: MasterMATLAB_1700_gridSmoothing.m

Run this code first: MasterMATLAB_1701_simulate3compMod.m




## 106. SPATIAL SHARPENING VIA LAPLACIAN

Generate a 2D Laplacian kernel and apply it to the space-time dataset in order to sharpen the edges.

Skills: conv2

## Definition [ edit ]

The Laplace operator is a second-order differential operator in the *n*-dimensional Euclidean space, defined as the divergence ($\nabla \cdot$) of the gradient ($\nabla f$). Thus if $f$ is a twice-differentiable valued function, then the Laplacian of $f$ is defined by:

$$\Delta f = \nabla^2 f = \nabla \cdot \nabla f$$

where the latter notations derive from formally writing:

$$\nabla = \left( \frac{\partial}{\partial x_1}, \ldots, \frac{\partial}{\partial x_n} \right).$$

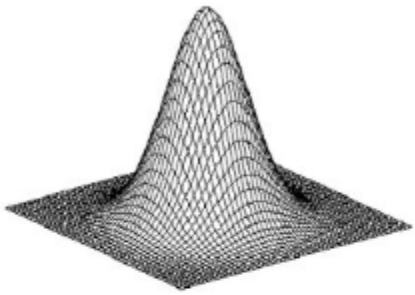
Equivalently, the Laplacian of $f$ is the sum of all the *unmixed* second partial derivatives in the Cartesian coordinates $x_i$:

$$\Delta f = \sum_{i=1}^{n} \frac{\partial^2 f}{\partial x_i^2}$$

As a second-order differential operator, the Laplace operator maps $C^k$ functions to $C^{k-2}$ functions for $k \geq 2$. The expression (**1**) (or equivalently (**2**)) defines an operator $\Delta : C^k(\mathbb{R}^n) \to C^{k-2}(\mathbb{R}^n)$, or more generally, an operator $\Delta : C^k(\Omega) \to C^{k-2}(\Omega)$ for any ope set $\Omega$.

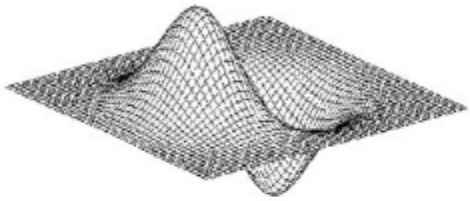
Reference: https://en.wikipedia.org/wiki/Laplace_operator

# 2D edge detection filters



Gaussian

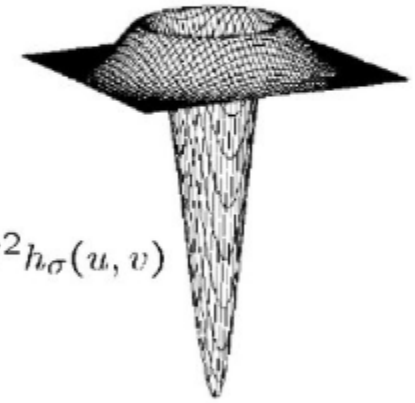
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



derivative of Gaussian

$$\frac{\partial}{\partial x} h_\sigma(u,v)$$

Laplacian of Gaussian



$$\nabla^2 h_\sigma(u,v)$$

$\nabla^2$ is the **Laplacian** operator:

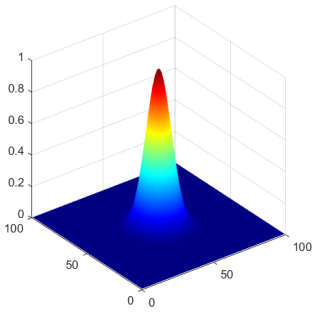
$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

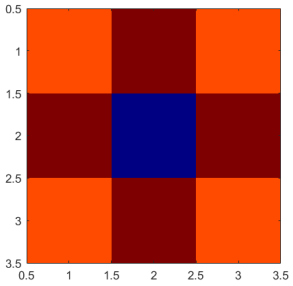
Reference: slide 62 of 63: “Computational Photography” https://en.ppt-online.org/77166

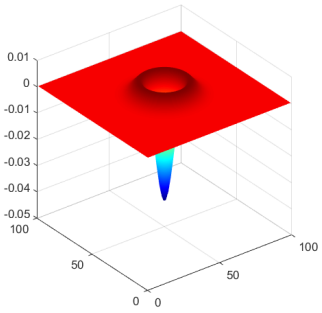
MATLAB

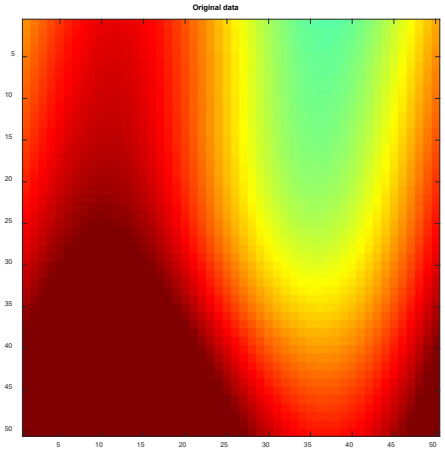
Code: MasterMATLAB_1720_spaceLaplace.m

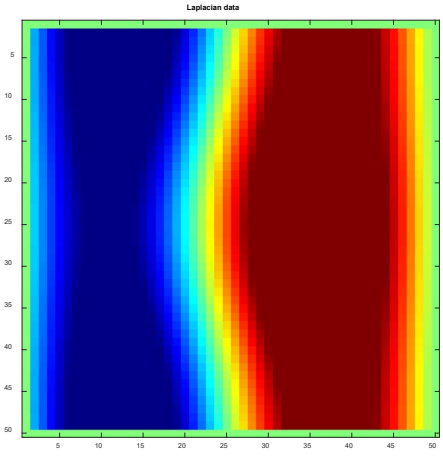
Run this code to set up variables: MasterMATLAB_1660_simulate3comp.m




## SECTION 18: TIME SERIES ANALYSIS

### 107. CONVOLUTION

Generate Brownian noise and a Gaussian kernel. Use convolution to smooth the noise.

Implement convolution in the frequency domain as multiplication of the Fourier spectra of the signal and kernel.

Skills: cumsum, exp, dot, fft, ifft

Convolution in the Time Domain:

Signal

$$x_i = \sum_{n=1}^{i} r_n$$

$$r \sim \mathrm{N}(0, 1)$$

Kernel

$$h = e^{\frac{-(-k:k)^2}{k}}$$

$$g(\tau) = h(2k + 1 - \tau)$$

Convolution

$$c_i = g^T x_{i-k:i+k}$$

Convolution in the Frequency Domain:

Signal

$$x_i = \sum_{n=1}^{i} r_n$$

$$r \sim \mathrm{N}(0, 1)$$

Kernel

$$g = e^{-(-k:k)^2/k}$$

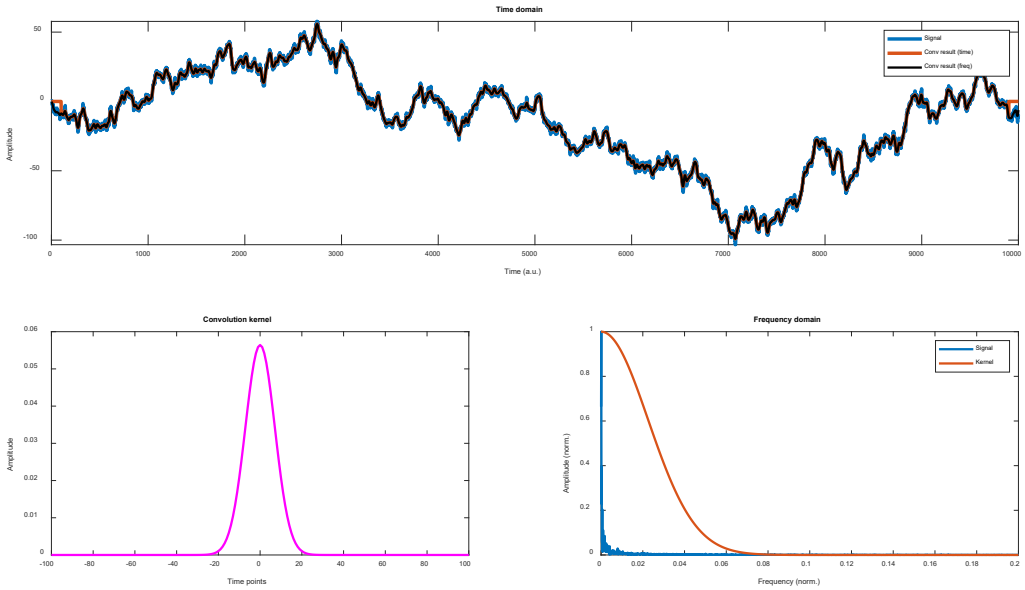
Convolution

$$c = IFFT\left( FFT_p(g)FFT_p(x) \right)$$

$$p = (2k + 1) + N - 1$$

## MATLAB

Code: MasterMATLAB_1740_convolution.m




## 108. HIGH-PASS FILTER USING FIR FILTER

Generate Brownian noise using a sampling rate of 1000 Hz

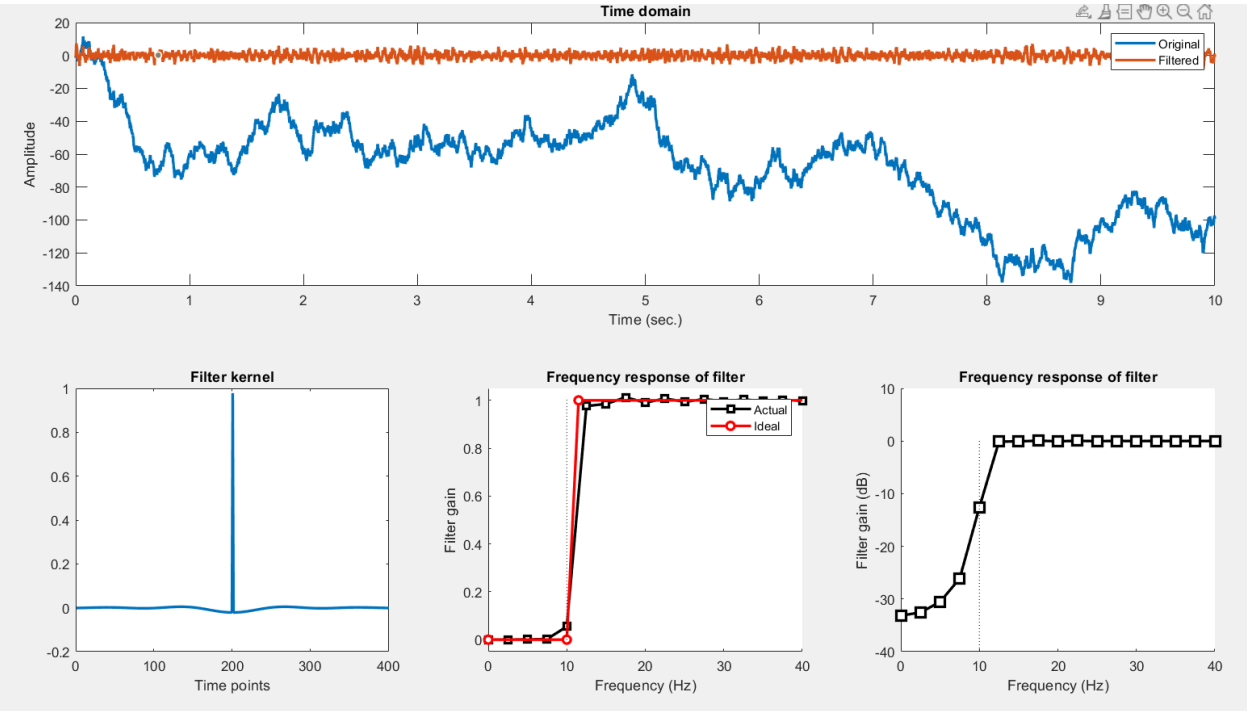
Create and apply a high-pass filter using *firls*.

Plot the filter kernel and its frequency-domain magnitude response.

Skills: firls, filtfilt, freqz, fft

## MATLAB

Code: MasterMATLAB_1760_highPassFIR.m




## 109. NARROW-BAND FILTER VIA FREQUENCY-DOMAIN GAUSSIAN

Generate a broadband signal in the time domain, and apply a narrowband filter by multiplying its spectra content by a frequency-domain Gaussian.

Compute the empirical full-width at half maximum of the Gaussian.

Skills: ifft, bsxfun

Equation for a Gaussian in the Frequency Domain:

$$g = e^{-.5((h-p)/s)^2}$$

$$s = \frac{w(2\pi - 1)}{4\pi}$$

Where:

$h$: frequencies (Hz)

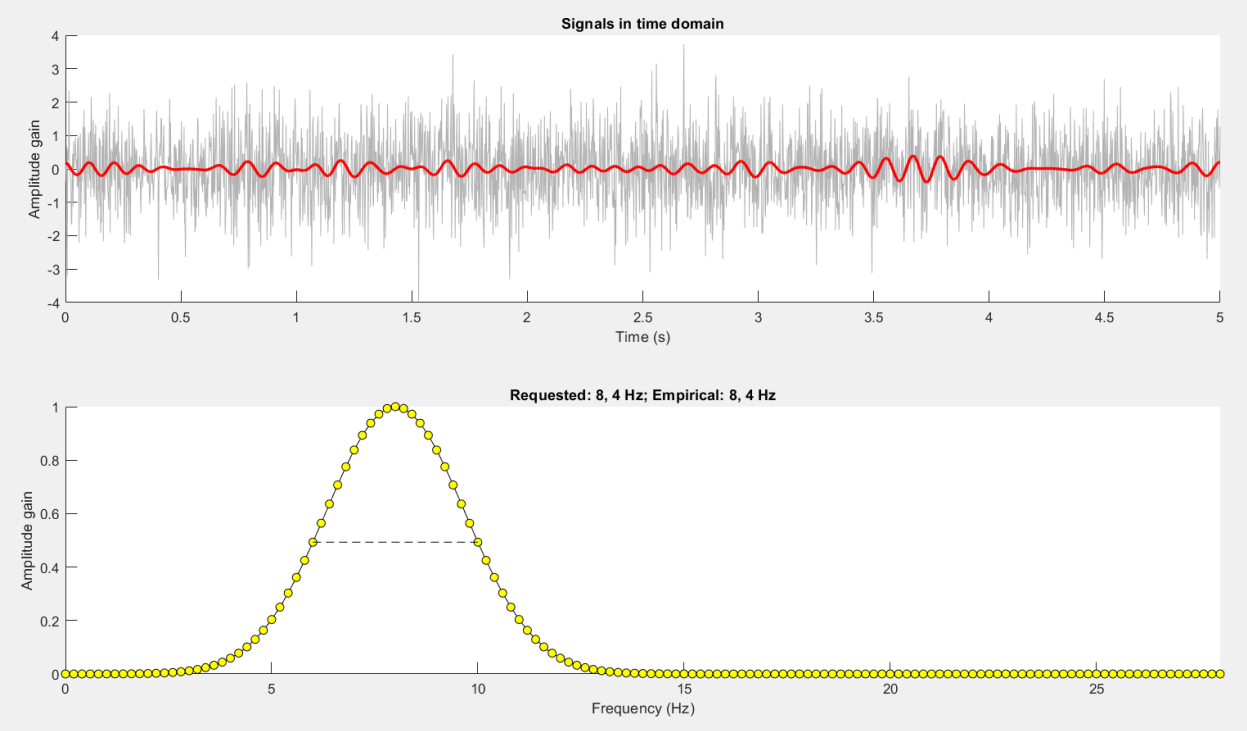
$p$: peak frequency (Hz)

$w$: FWHM (Hz)

FWHM: Full Width Half Maximum

### MATLAB

Code: MasterMATLAB_1780_narrowband.m




## 110. CAUSAL VS. ZERO-PHASE-SHIFT FILTER

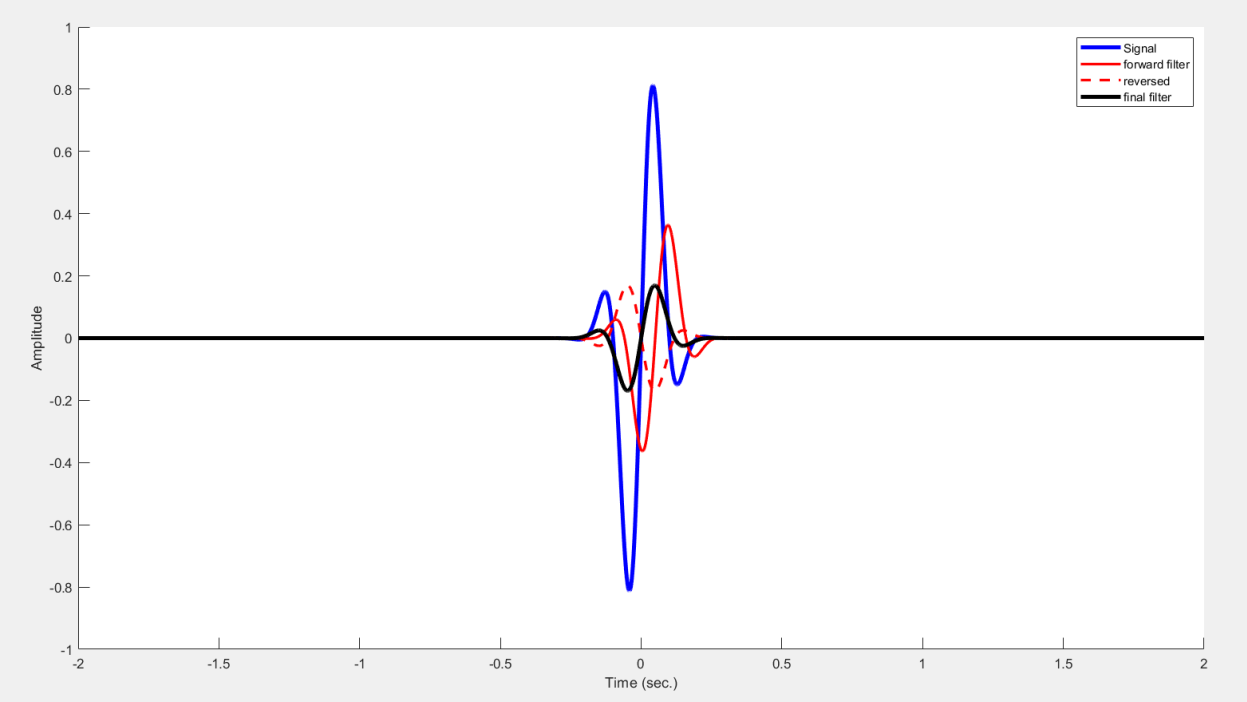
Generate a signal (Morlet wavelet) at 5 Hz. Apply a causal (forward) and acausal (zero-phase-shift) narrowband filter.

Look through the MATLAB *filtfilt* function to find where and how it implements the zero-phase-shift filtering.

Skills: filter, filtfilt, firls

## MATLAB

Code: MasterMATLAB_1800_causalZeroFilt.m




Code extracted from key segment of R2020a filtfilt function

```
function yout = ffOneChanCat(b,a,y,zi,nfact,L)

coder.varsize('yout');
yout = y;
for ii=1:L
   % Single channel, data explicitly concatenated into one vector
   ytemp = [2*yout(1,1)-yout(nfact(1,1)+1:-1:2,1); yout(:,1); 2*yout(end,1)-yout(end-1:-1:end-
nfact(1,1),1)];

   % filter, reverse data, filter again, and reverse data again
   ytemp = filter(b(:,ii),a(:,ii),ytemp(:,1),zi(:,ii)*ytemp(1,1));
   ytemp = ytemp(end:-1:1,1);
   ytemp = filter(b(:,ii),a(:,ii),ytemp(:,1),zi(:,ii)*ytemp(1,1));

   % retain reversed central section of y
   yout = ytemp(end-nfact(1,1):-1:nfact(1,1)+1,1);
end
```

```
end
```

## 111. LINE NOISE NOTCH FILTER

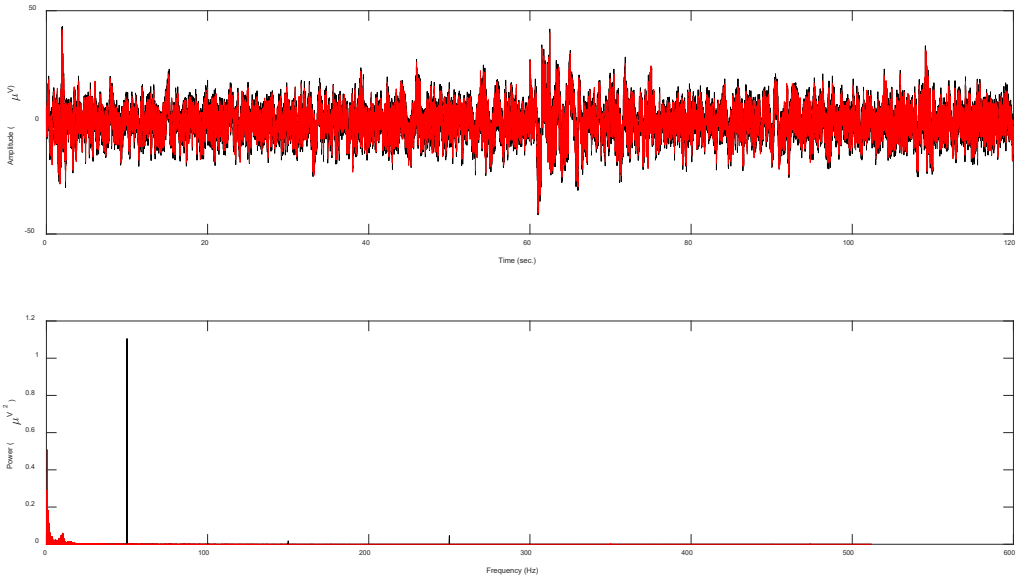
Generate a notch filter to remove 50 Hz line noise from human EEG (electrical brain recording) data.

Also remove harmonics of 50 Hz.

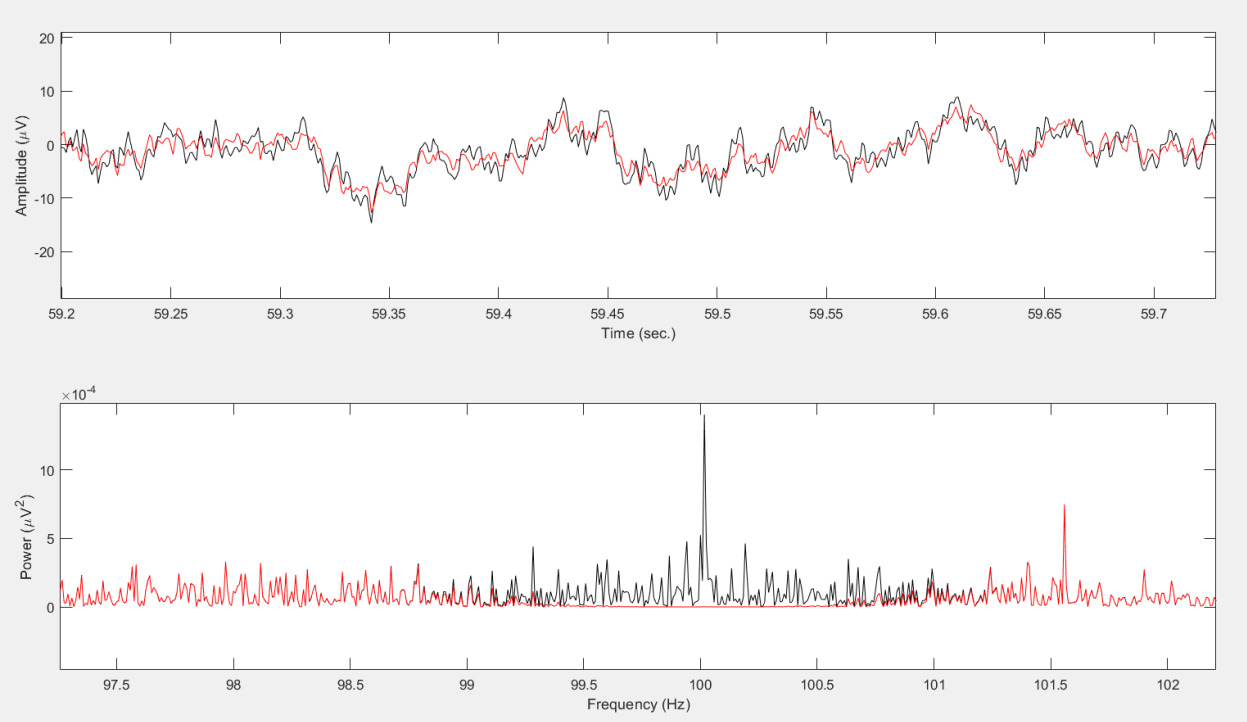
Skills: filter, filtfilt, fir1

## MATLAB

Code: MasterMATLAB_1820_notchFilter.m




Close-up at first harmonic (100 Hz)




## 112. COMPUTE ENVELOPE OVER PEAKY, NOISY SIGNAL

Add large-amplitude spikes to random noise. Determine an amplitude threshold and identify the spikes. Generate a new time series at the interpolated envelope over spikes.

Loop over possible thresholds to have a more data-driven approach to finding the best threshold.

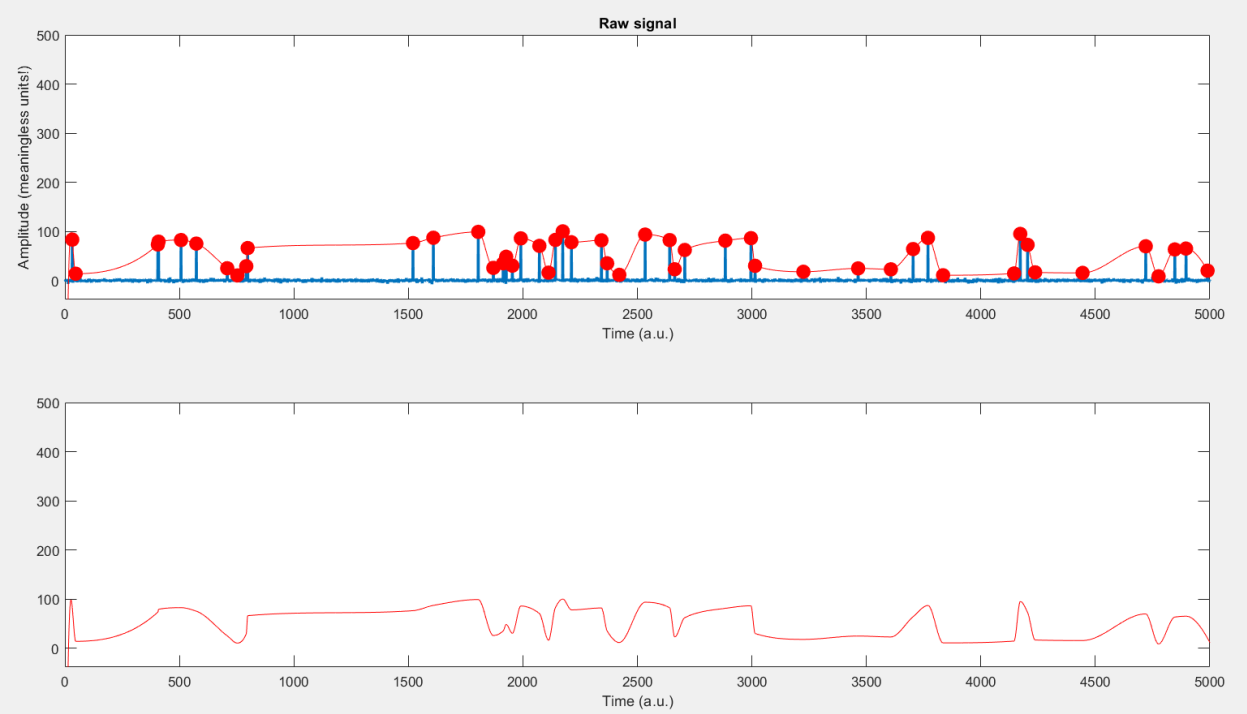
Skills: randperm, sum, find, interp1

Interpolate: estimate new values between data points.

Extrapolate: extend plot beyond known data points using trends in known data.

### MATLAB

Code: MasterMATLAB_1840_peakEnvelope.m




## 113. FREQUENCY-DOMAIN MEAN FILTER

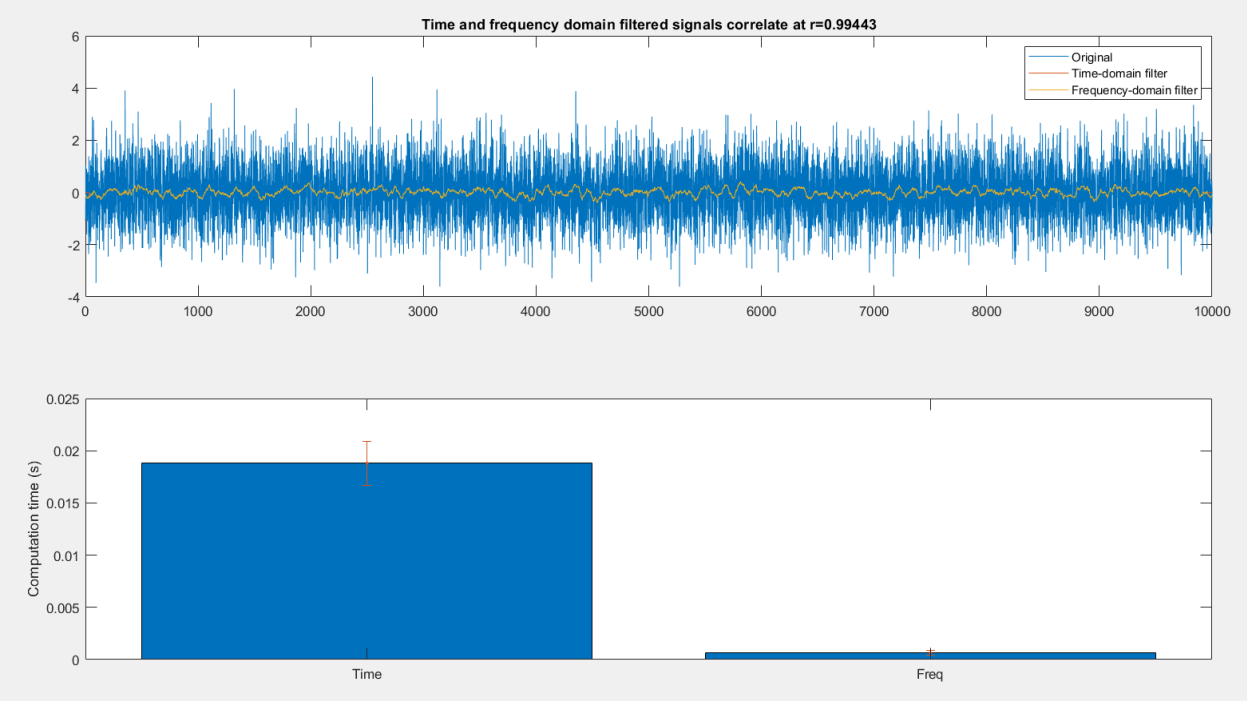
Implement the time-domain moving-average smoothing filter via frequency-domain multiplication (circular convolution). Plot computation times for both implementations.

Repeat the timer for 100 iterations and include error bars.

Skills: fft, ifft, tic/toc, plot

### MATLAB

Code: MasterMATLAB_1860_freqMeanFilt.m




## 114. CREATE A "CHIRP: (FM SIGNAL)

Create a sine wave with time-varying frequency. Show the time-domain signal and its power spectrum.

Compute a time series of instantaneous frequencies.

Skills: sin, fft, abs, hilbert (signal processing toolbox), angle

### MATLAB

Code: MasterMATLAB_1880_chirp.m

### PHASE TO FREQUENCY

Converting phase to frequency for FFT using hilbert and unwrap MATLAB functions

## Equation:

$$F(t,f) = \left(\varphi_{t,f} - \varphi_{t-1,f}\right)\frac{\delta}{2\pi}$$

## Where:

$F\ is\ instantaneous\ frequency$

$\varphi\ is\ phase$

$\delta\ is\ sample\ rate$

$t, f\ is\ signal\ time\ and\ frequency$

MATLAB:

```
% compute instantaneous frequency
phaseangles = angle(hilbert(signal));
instfreq = diff(unwrap(phaseangles)) / (2*pi/srate);
```

## LINEAR CHIRP

Frequency changes linearly over time.

Equation:

$$\text{signal} = \sin(2\pi y_t t_t)$$

where:

$y_t\ and\ t_t$ are both time series vectors.

And:

$$y = f_1 + (\frac{f_1 + f_2}{2} - f_1)\frac{1}{N}(a_k)_{k=0}^{k=N-1}$$

where:

$f_1$ is the start frequency

$f_2$ is the stop frequency

$a$ is a set of linear indices from $0\ to\ N-1$

$N$ is the total number of time points

## MATLAB:

```
% details, details
srate = 1000;
time = 0:1/srate:5;
pnts = length(time);

hz = linspace(0,srate/2,floor(pnts/2)-1);

% frequency range for dipolar chirp (linear)
f = [5 15]; % in Hz

% dipolar frequency formula
ff = f(1) + (mean(f)-f(1))*(0:pnts-1)/pnts;
signal = sin(2*pi*ff.*time);
```

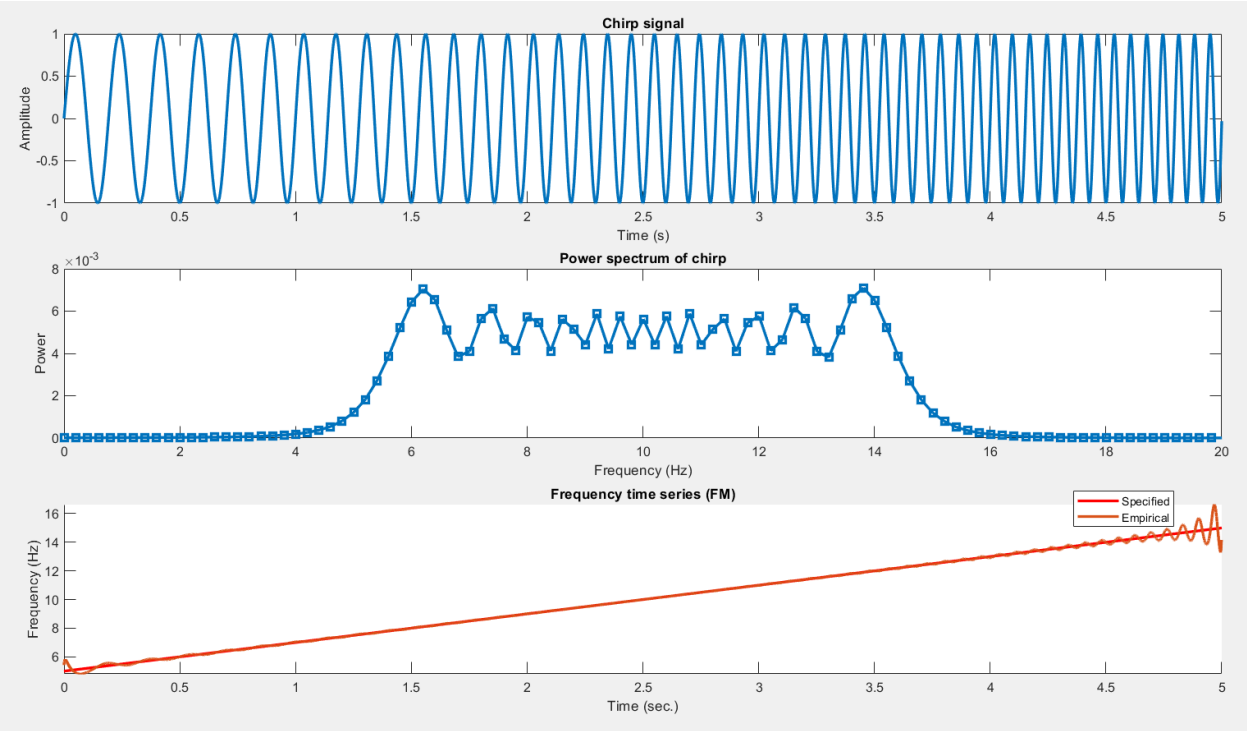



Figure 1: Linear Chirp

### MULTIPOLAR CHIRP

Frequency can charge arbitrarily over time.

## Equation:

$$signal = \sin(2\pi(y_t + t_t))$$

where:

$y_t \; and \; t_t$ are both time series vectors.

And:

$$y_t = \delta \sum_{k=1}^{t} f_k$$

where:

$\delta$ is the sampling rate

$f_k$ represents a series of frequencies that are fairly close to each other over time

MATLAB:

```
% details, details
srate = 1000;
time = 0:1/srate:5;
pnts = length(time);

hz = linspace(0,srate/2,floor(pnts/2)-1);

% multipolar frequency formula
freqmod = exp(-(time-mean(time)).^2)*10+10;
signal = sin( 2*pi * (time + cumsum(freqmod)/srate) );
```

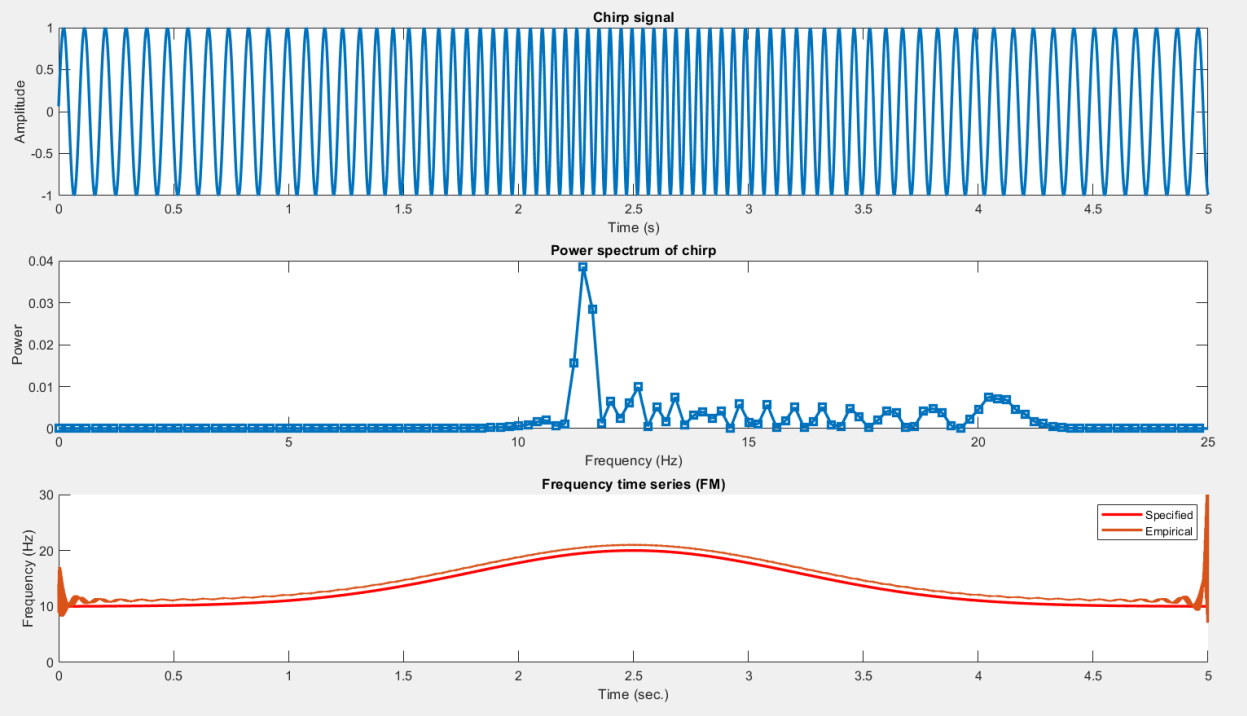



Figure 2: Multipolar Chirp

## 115. DETRENDED FLUCTUATION ANALYSIS

Create a “1/f” (a.k.a. pink noise) random time series, and a white-noise time series. Implement a detrended fluctuation analysis to determine the “fractalness” of both time series.

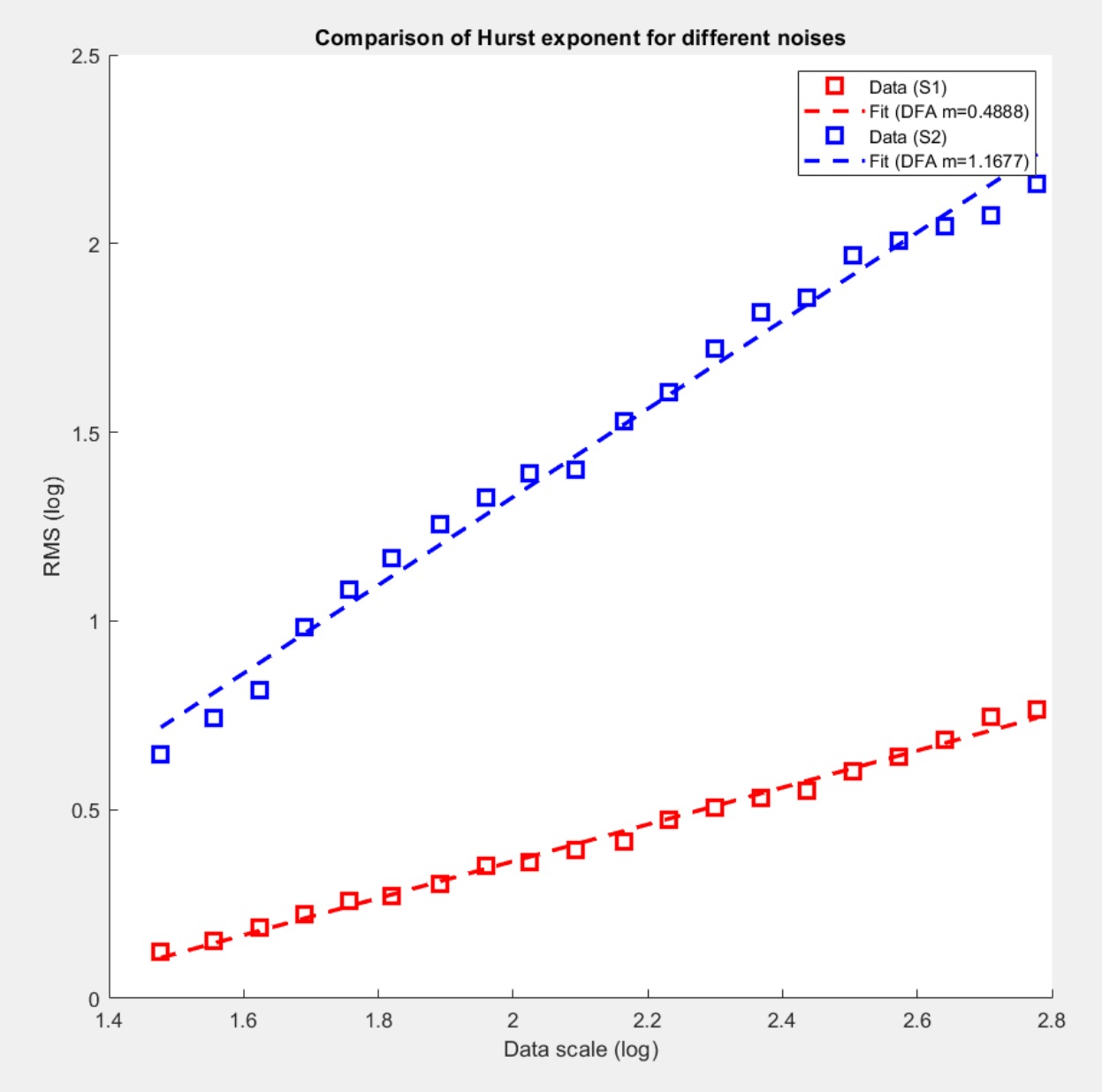
Skills: exp, ifft, log10, cumsum, reshape, detrend

### DETRENDED FLUCTUATION ANALYSIS:

1. Convert signal to mean-centered cumulative sum.
2. Define 20 logarithmically spaced scales between 1% and 20% of the length of the signal.
3. Loop over scales and:
    a. Split the time series into “trials” with length corresponding to scale.
    b. Detrend the time series in each trial.
    c. Compute the RMS for each trial, and take the average over trials.
4. Compute the linear fit between the log-scale factors and log-RMS.

### MATLAB

Code: MasterMATLAB_1900_fluctuation.m




## SECTION 19: SPECTRAL ANALYSIS

### 116. CREATE MULTISPECTRAL TIME SERIES

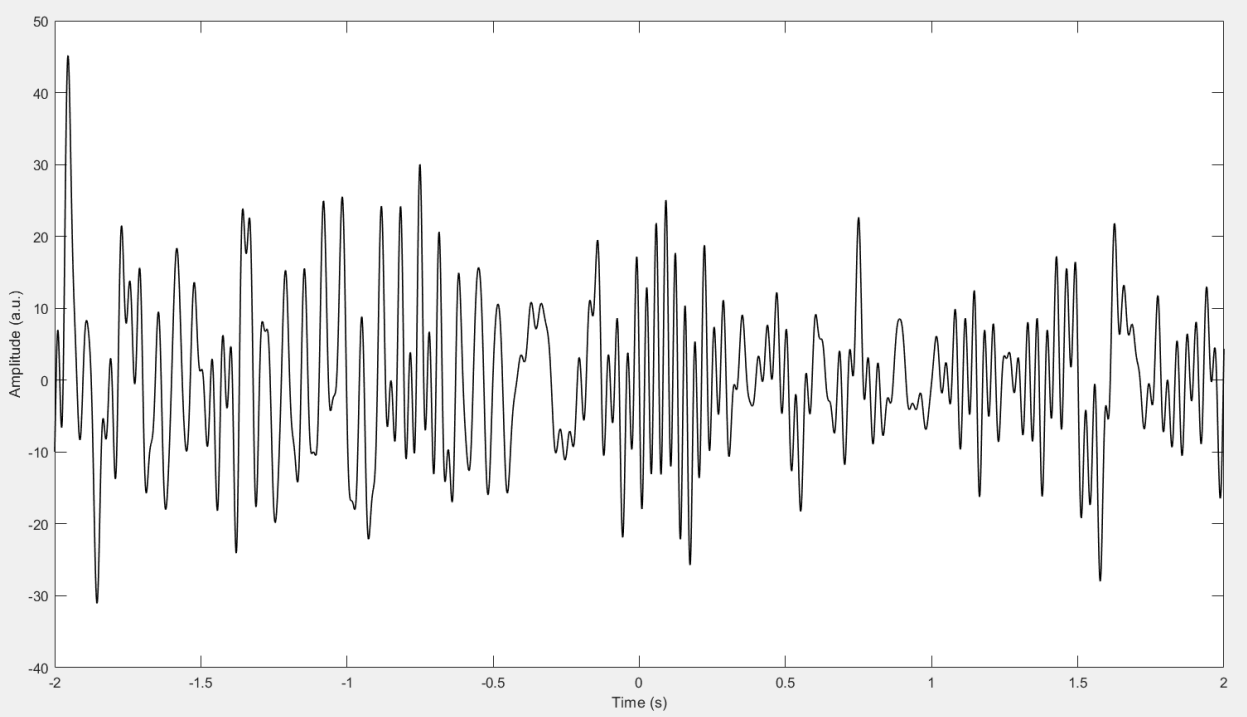
Generate a time series by summing 5 sine waves of different frequencies and amplitudes.

Use time-varying amplitudes instead of a fixed amplitude per frequency.

Skills: for, interp1, sin

### MATLAB

Code: MasterMATLAB_1920_multispectralTS.m




## 117. POWER SPECTRUM FROM FFT AND WELCH'S METHOD

Compute the power spectrum of a multispectral signal (see previous video) using FFT and Welch's method. Add dashed vertical lines to indicate simulated frequency components.

Add all dashed lines in one code-line.

Skills: fft

### LECTURELET

http://mikexcohen.com/lecturelets/fftstationarity/fftstationarity.html

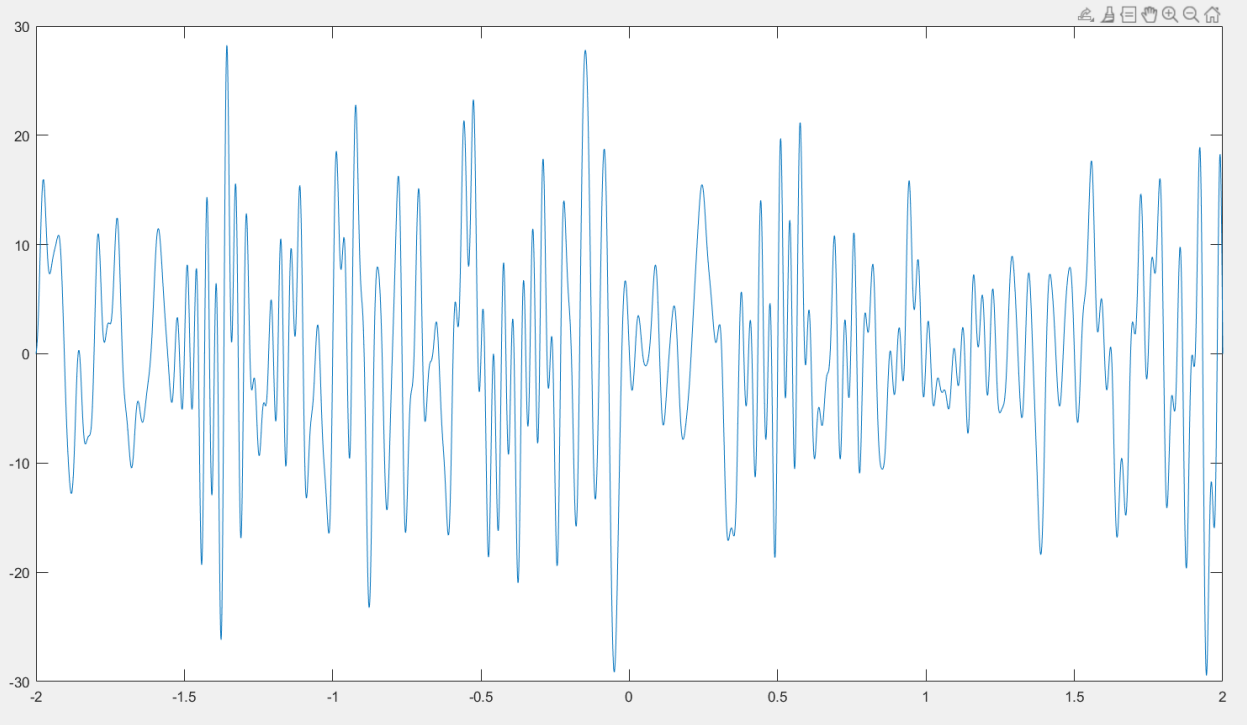
Topic: effect on frequency domain when non-stationarities exist in the time domain.

Code: fftstationarity.m

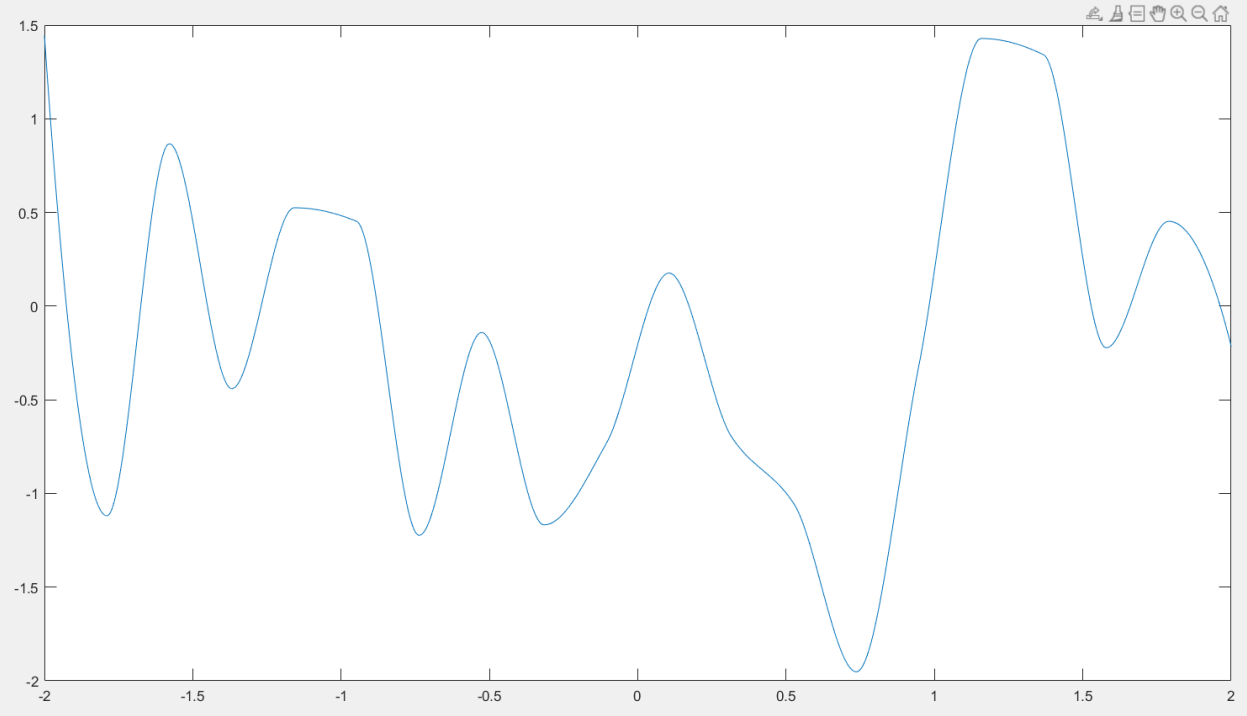
## MATLAB

Code: MasterMATLAB_1940_powerSpectrum.m

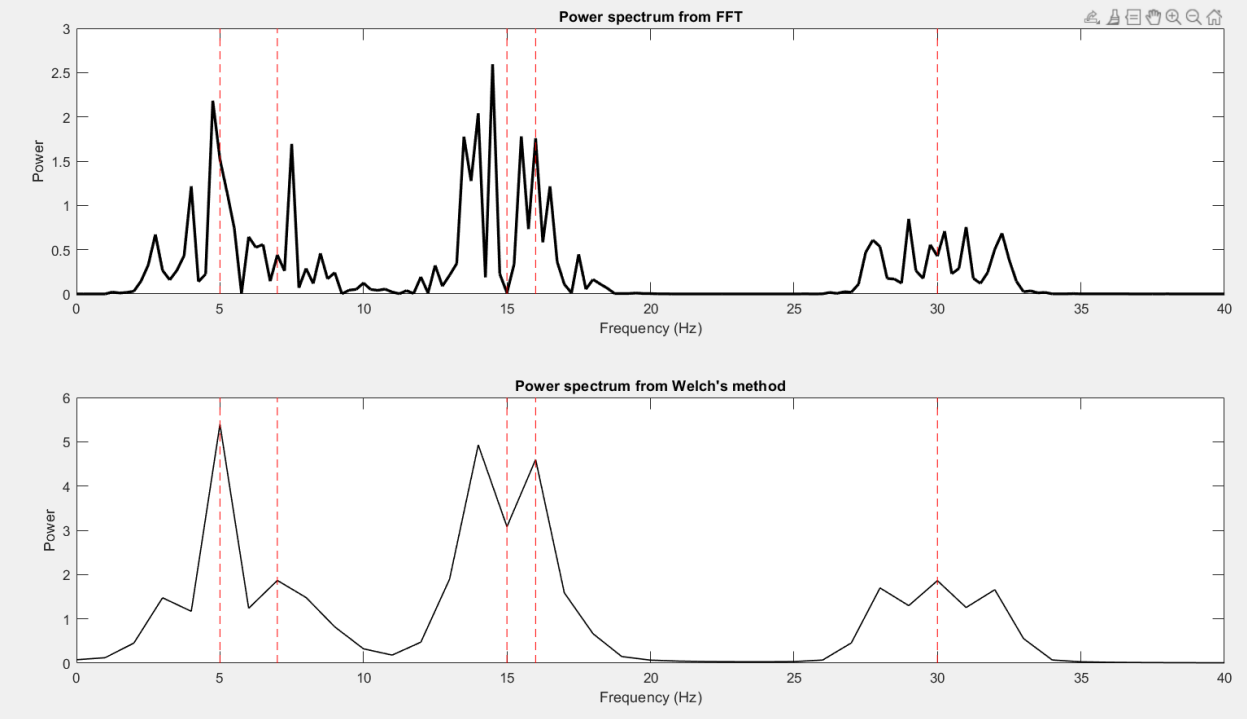
Here is a signal generated from the script showing 5 frequencies (5, 7, 15, 16 and 30 Hz) mixed together along with amplitude modulation (AM) as an example of a signal with non-singularity. The power spectrum from FFT appears with extra energy outside of the primary frequencies as a result.



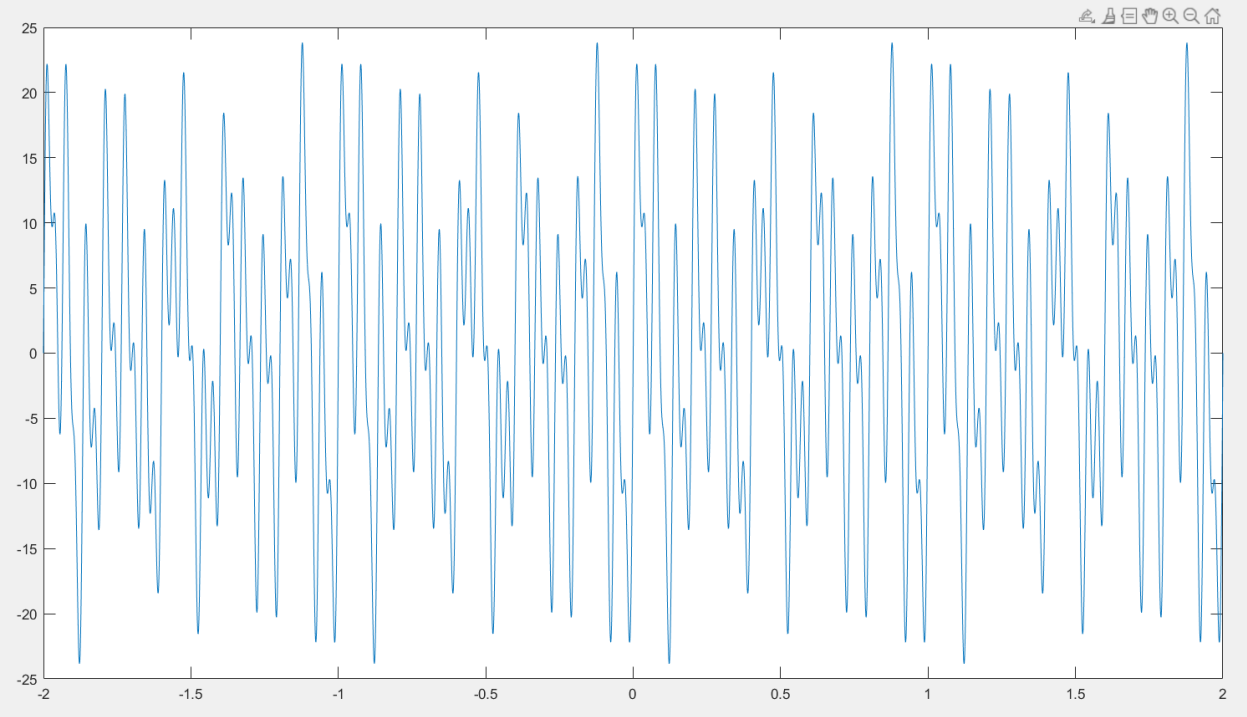

The amplitude modulation applied to the 5 frequencies.



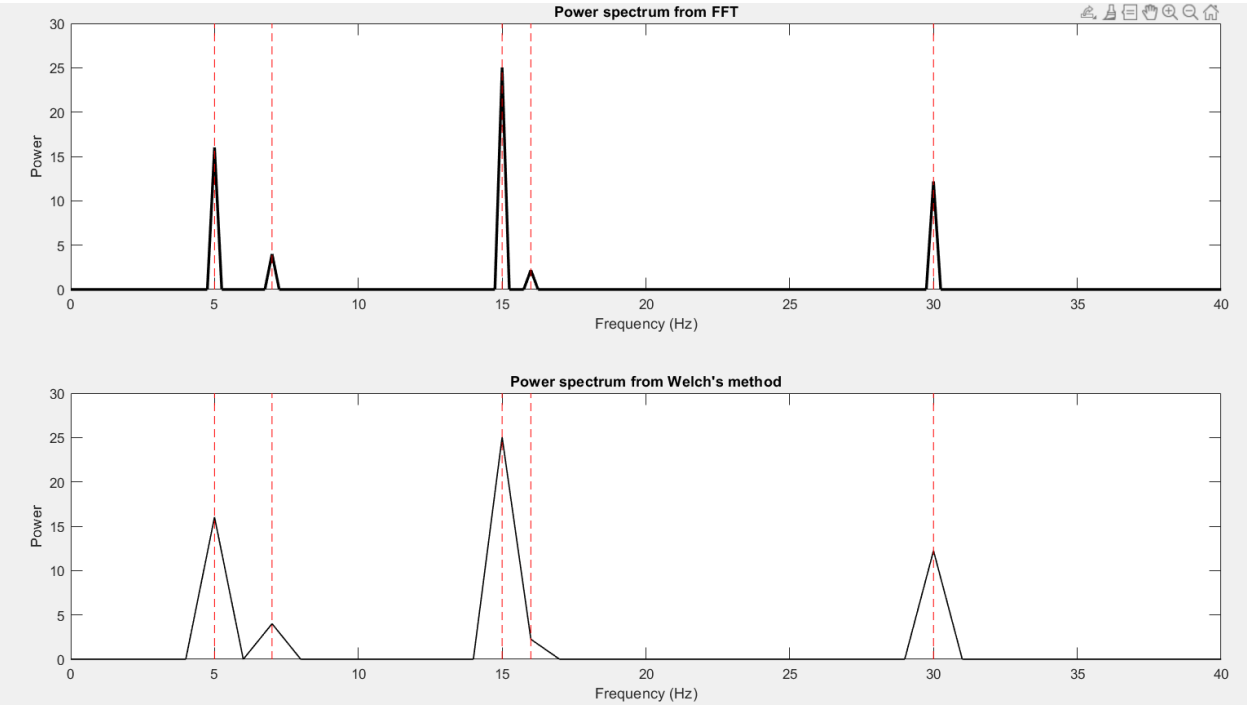

The static power spectrum and the spectrum using the Welch's method to reduce noise in exchange for reduced frequency resolution.



The same 5 frequencies mixed together without the AM.




The same power spectrums using the signal above but without AM. Without the non-singularities the static power spectrum shows very clean frequency peaks. While the Welch's method shows the lack of frequency resolution when two frequencies are close together.



## 118. SPECTROGRAM OF A BIRD CALL

Import an audio file of a bird call. Inspect the time- and frequency- domain representations, and display the time-frequency power using a spectrogram.

Sum the spectrogram over time and frequency, and compare that to the static FFT and RMS time series.

Skills: audioread, soundsc, spectrogram

### MATLAB

Code: MasterMATLAB_1960_birdCallSpect.m

Bird call file: XC403881.mp3

Bird call source: https://www.xeno-canto.org/403881

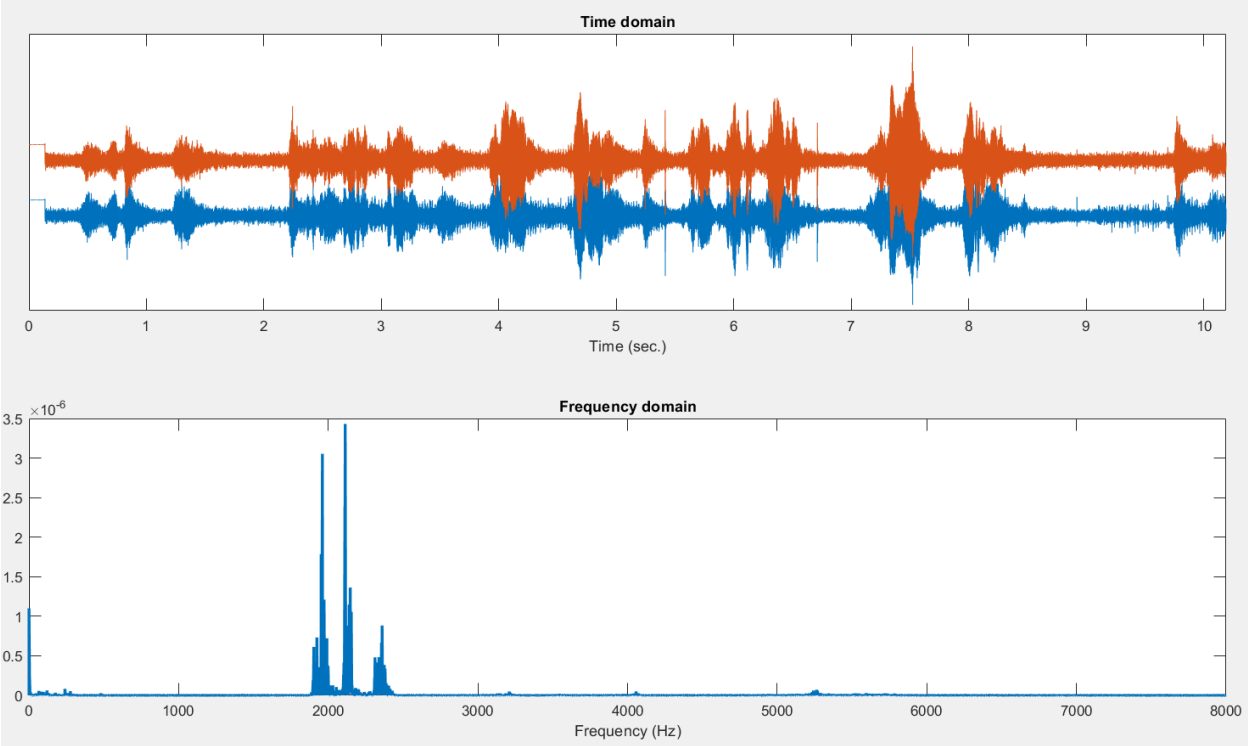



Figure 1: Time domain vs Frequency domain of bird call

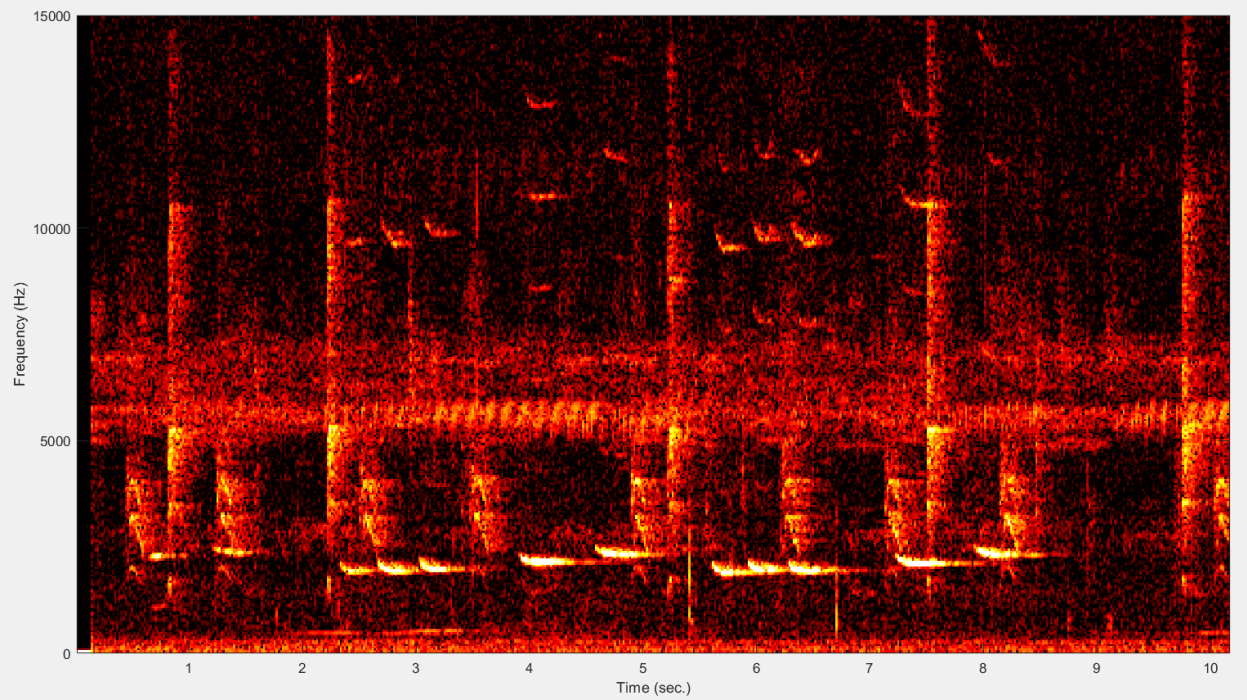



Figure 2: Spectrogram

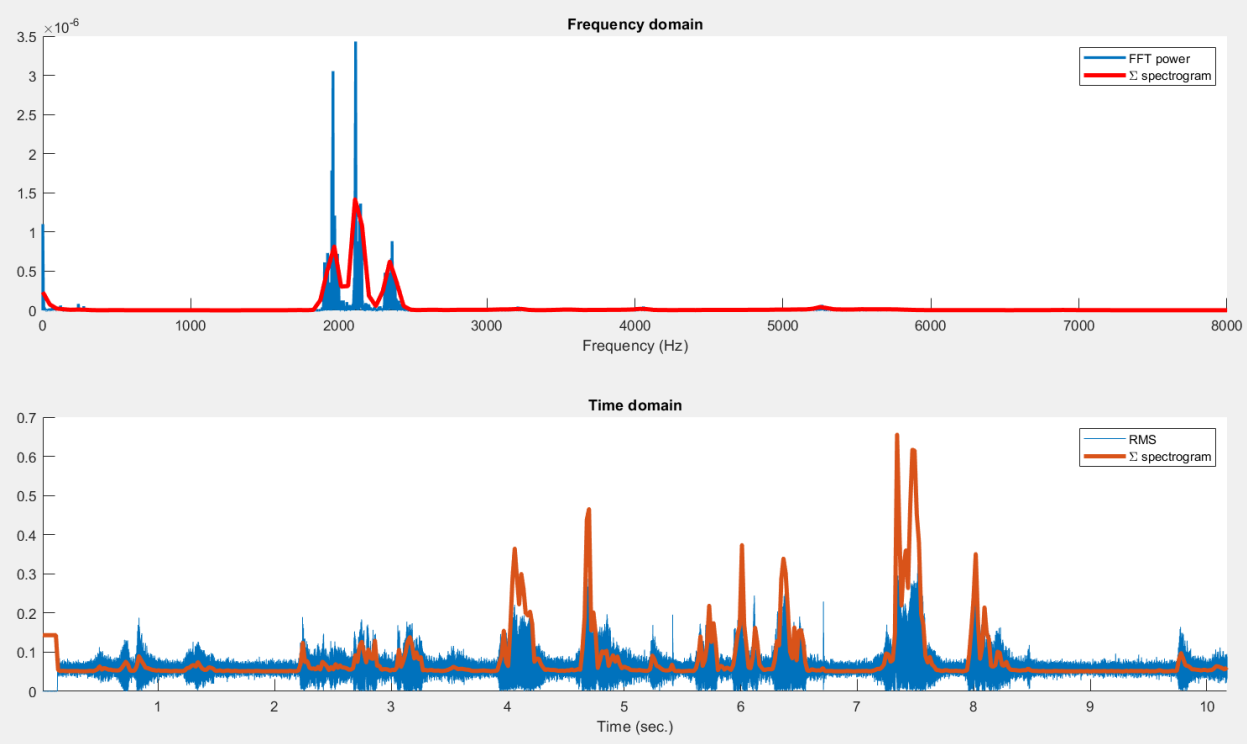



Figure 3: compare spectra from FFT vs. summed spectrogram (Welch's method)

## 119. PHASE-SCRAMBLE NARROWBAND TIME SERIES

Generate a narrowband time series by computing the inverse FFT of Gaussian-windowed nonnegative random numbers. Scramble the phases and reconstruct the time series.

Skills: ifft, rand

Equation for a Gaussian in the Frequency Domain (see lecture 109):

$$g = e^{-.5((h-p)/s)^2}$$

$$s = \frac{w(2\pi - 1)}{4\pi}$$

Where:

$h$: frequencies (Hz)

$p$: peak frequency (Hz)

$w$: FWHM (Hz)

FWHM: Full Width Half Maximum

MATLAB:

```
% parameters
srate = 600;  % sampling rate in Hz
npnts = srate*2; % number of time and frequency points
time = (0:npnts-1)/srate; % time vector

% create frequencies vector (using trick of going beyond Nyquist)
hz = linspace(0,srate,npnts);

% parameters for frequency-domain Gaussian
fwhm = 5;  % fwhm in Hz
peakf = 13; % peak frequency of Gaussian, in Hz

% create Gaussian
```

```
s = fwhm*(2*pi-1)/(4*pi); % normalized width
x = hz-peakf;          % shifted frequencies
fx = exp(-5*(x/s).^2);     % the Gaussian
```

Equation for reconstructed signal:

$$x = IFFT( ae^{i\varphi} )$$

Where:

$\varphi$: Phase

$a$: Real portion of frequency domain

MATLAB:

```
% signal is ifft of gaussian times random noise
signal = real( ifft( fx.*rand(1,npnts) ) );

% compute the Fourier transform of the signal
signalX = fft( signal );

% shuffle the phases
phases = angle( signalX );

% create a new Fourier series as ae^{ip}
shuffdata = abs(signalX) .* exp(1i*phases);
```

MATLAB

Code: MasterMATLAB_1980_phaseScramble.m

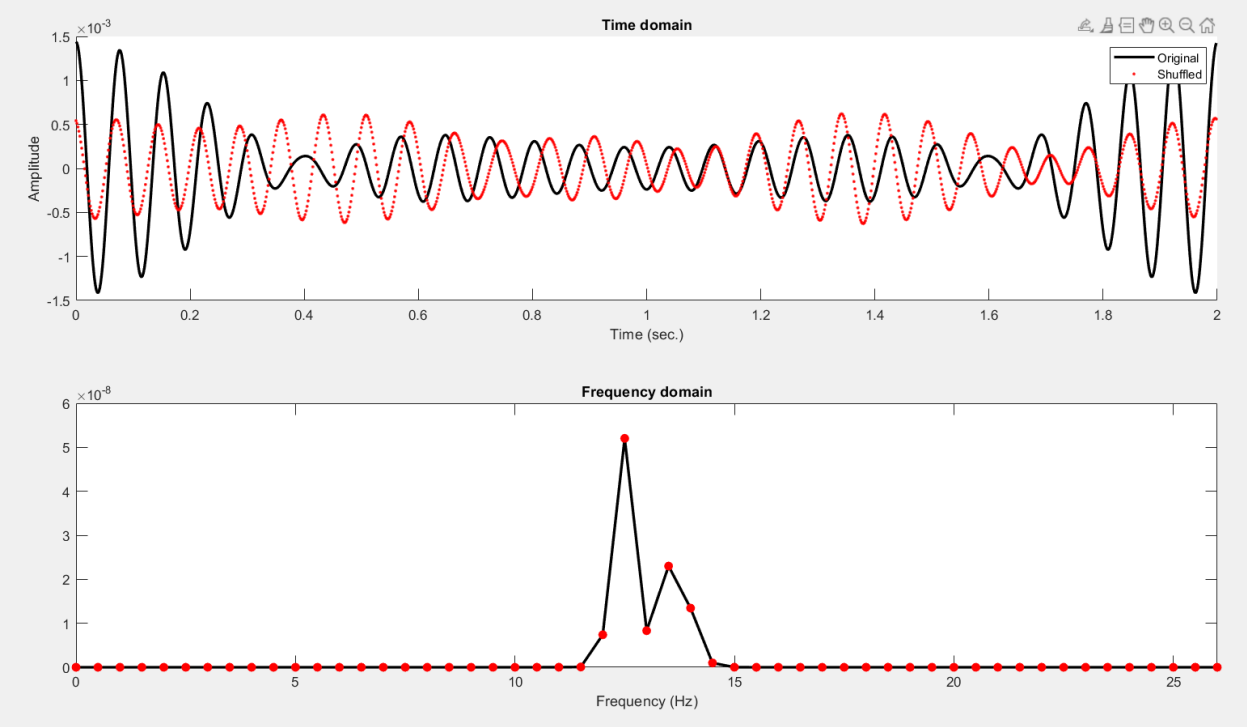



Figure 1: Time domain compares same frequencies but with different phases, while Frequency domain shows that the same frequencies are present even when phase has been changed.

## 120. HILBERT TRANSFORM

Implement the FFT-based algorithm to compute the Hilbert transform of a signal.

Skills: fft, complex, Hilbert, real, abs, angle

Steps to perform the Hilbert Transform:

1. Take the FFT of the signal.

```
% 1) take FFT
fc = fft(signal);
```

2. Create a copy of the Fourier coefficients multiplied by i.

```
% 2) create a copy that is multiplied by the complex operator
complexf = 1i*fc;
```

3. Identify the positive (>0 to < Nyquist) and negative (>Nyquist to end) frequencies.

```
% 3) find indices of positive and negative frequencies
posF = 2:floor(n/2)+mod(n,2);
negF = ceil(n/2)+1+~mod(n,2):n;
```

4. Add -i times the copy positive frequencies and add +i times the copy negative frequencies.

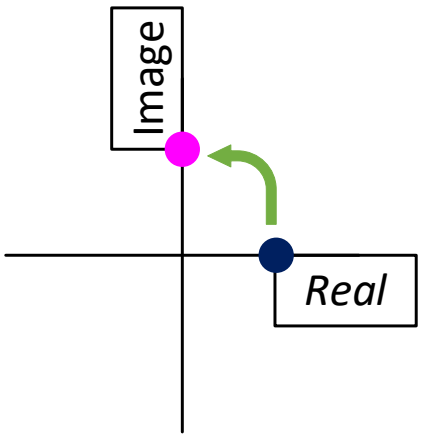
```
% 4) rotate Fourier coefficients
fc(posF) = fc(posF) + -1i*complexf(posF);
fc(negF) = fc(negF) + 1i*complexf(negF);
```

5. Take the inverse FFT.

```
% 5) take inverse FFT
hilbertx = ifft( fc );
```

Result of performing the Hilbert Transform is a complex signal in the time domain from which the following time series can be extracted:

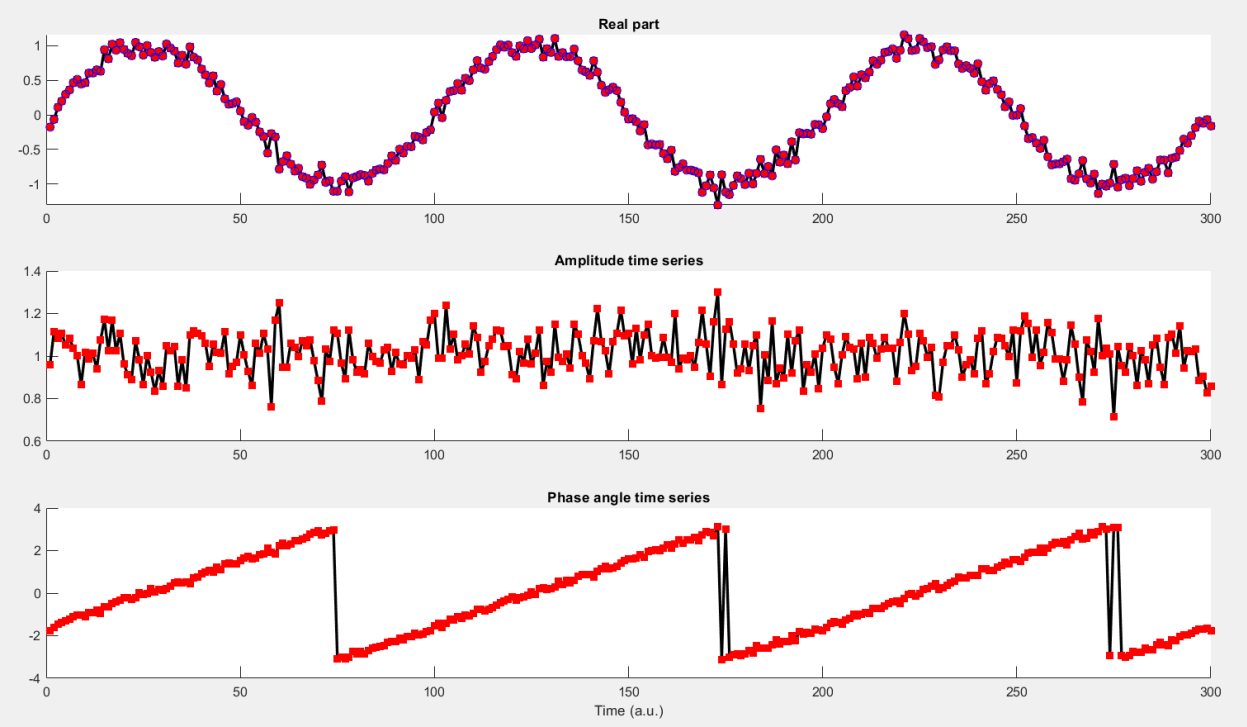
- Real part (same as original signal
- Amplitude
- Phase Angle




The geometric interpretation: put the real value signal into the complex plane, then rotate the coefficients 90 degrees (or π/2).

## MATLAB

Code: MasterMATLAB_2000_HilbertTrans.m




# 121. OSCILLATIONS IN HUMAN BRAIN RECORDINGS

Copy-and-adapt the code from the “Power spectrum from FFT and Welch’s method” video to work with the human EEG dataset.

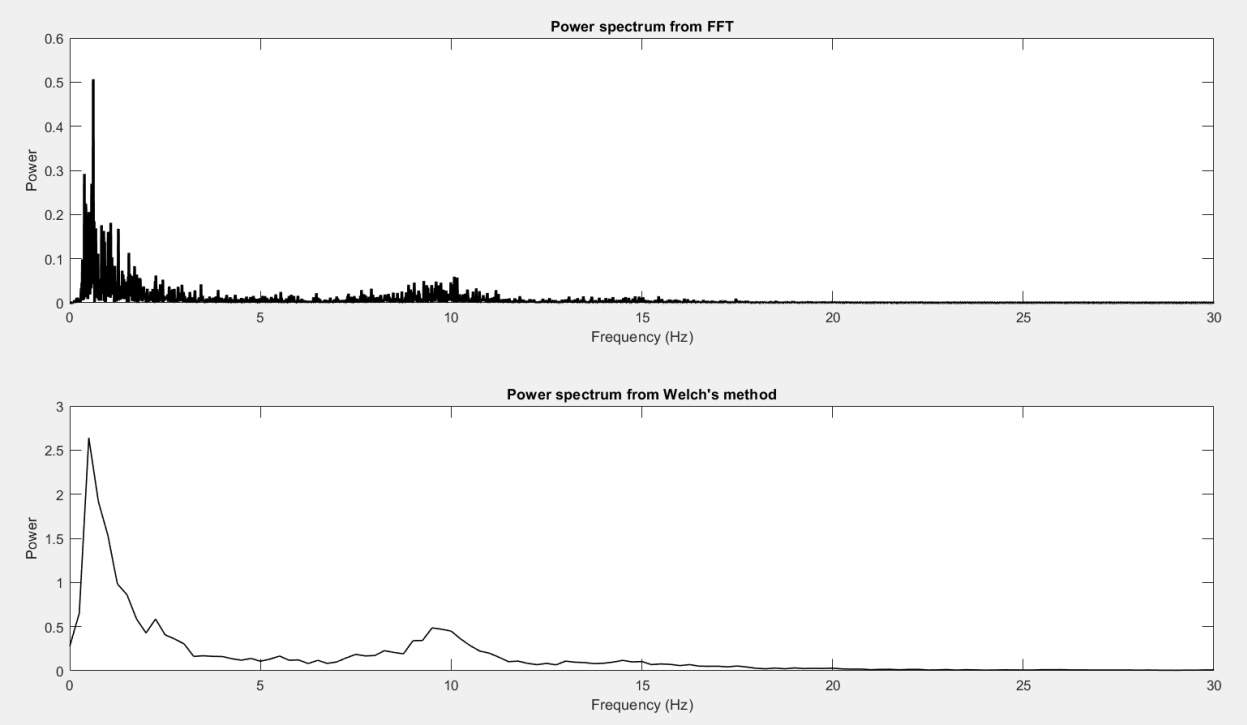
Skills: fft

## MATLAB

Code: MasterMATLAB_2020_humanBrain.m

Data from section 18: EEGrestingState.mat

Reuse code from section 19, ecture 117: MasterMATLAB_1940_powerSpectrum.m




## 122. TIME FREQUENCY ANALYSIS VIA WAVELET CONVOLUTION

Create a chirp comprising a Gaussian plus a linear trend. Produce a time-frequency power plot ("spectrogram') by generating a family of Morlet wavelets and convolving each wavelet with a chirp.

Implement convolution "manually" instead of using the MATLAB *conv* function.

Skills: linspace, fft, conv, floor, abs, contour

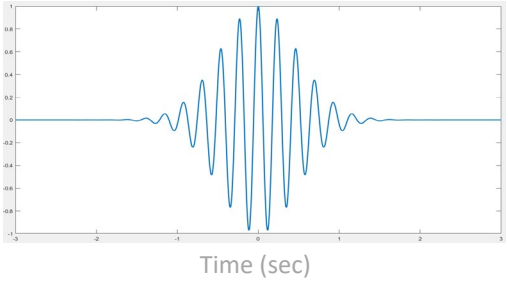
Equation for complex Morlet wavelet:

$$w_f = e^{i2\pi f t} e^{-t^2/2s^2}$$

Where width is:

$$s = \frac{13}{2\pi f}$$

Note: 13 in numerator above, is the number of cycles for the wavelet



## MATLAB

Code: MasterMATLAB_2040_waveletConv.m

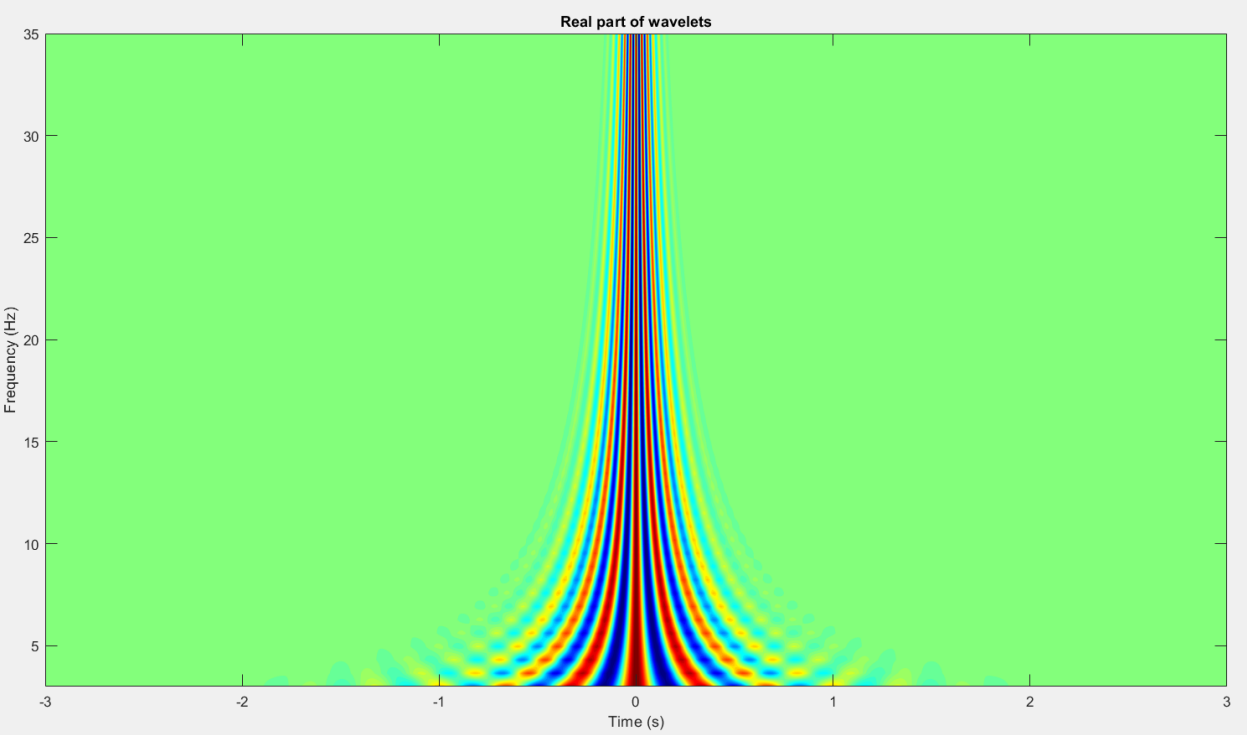



Figure 1

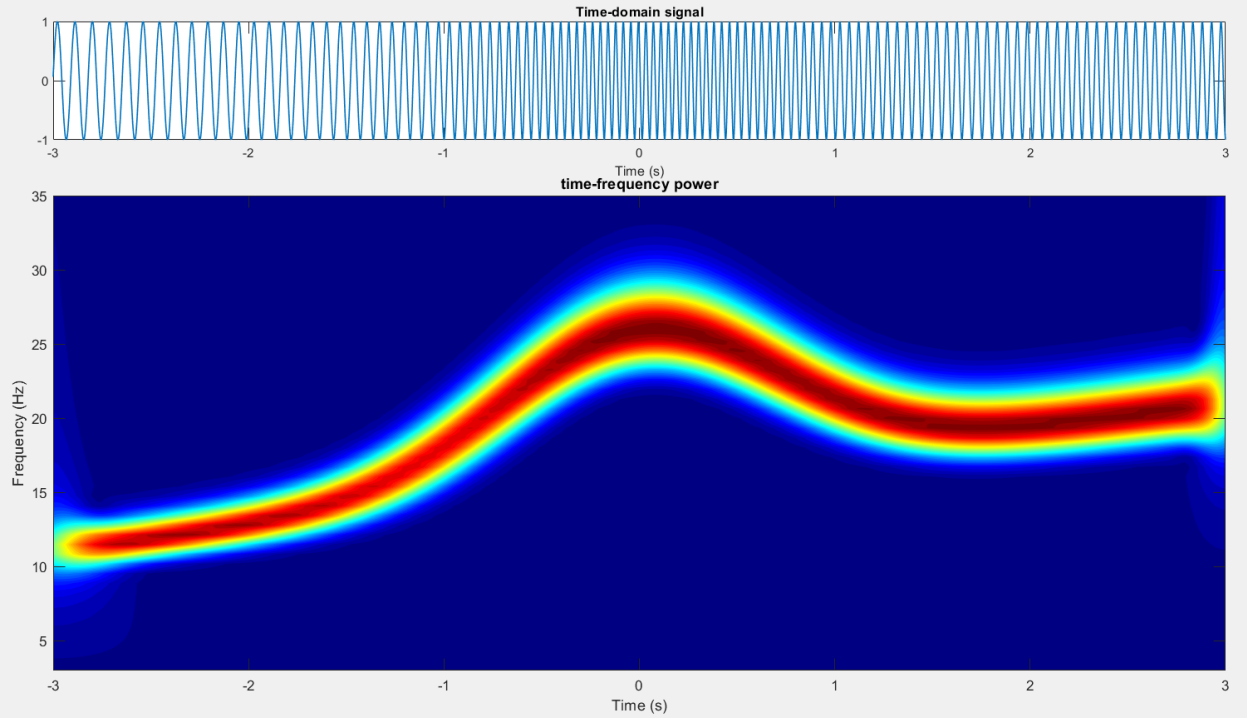



Figure 2: where time-frequency power is normalized using manual conv (not possible with MATLAB conv function)

## 123. UNSOLVED: TIME-FREQUENCY ANALYSIS OF INTERPOLATED SIGNAL

Wavelet convolution of spectrally interpolated signal

References: 104. spectral mixing interpolation lecture, and 122. Time Frequency Analysis via Wavelet Convolution

# SECTION 20: MATRIX ANALYSIS

## 124. FOUR WAYS TO COMPUTE THE DOT PRODUCT

Find four different ways to implement the “dot product” operation in MATLAB.

Use the geometric definition of the dot product to find a 5’th implementation!

Skills: dot, sum, for, acos

Dot product is the computational backbone for:

- Correlation
- Covariance
- Convolution
- Fourier transform
- And more…

## Four equations to compute the dot product:

$$\alpha = \boldsymbol{a}^T \boldsymbol{b} = \sum_{i=1}^{n} a_i \, b_i = \cos(\theta_{ab}) \; ||a|| \; ||b||$$

### MATLAB

Code: MasterMATLAB_2060_dotproduct.m

```
% two vectors (must be the same size!)
v = [ 1 2 3 4 ];
w = [ 0 5 1 9 ];

% method 1 (sum)
dp1 = sum( v.*w );

% method 2 (dot)
dp2 = dot( v,w );

% method 3 (multiplication and transpose)
dp3_opta = v(:)'*w(:); % convert both v and w into column vectors using "(:)"
dp3_optb = v*w';

% method 4 (loop)
dp4 = 0;
for i=1:length(v)
   dp4 = dp4 + v(i)*w(i);
end

% bonus: method 5 (geometry)
phs = acos( dot(v,w) / (norm(v)*norm(w)) );
dp5 = norm(v)*norm(w)*cos(phs);
```

## 125. SOLVED: PLOT VECTORS AND COMPUTE LENGTHS

Skills: plot, plot3, dot, ‘xlim’, ‘ylim’, axis

Equation for vector length:

$$\sqrt{\boldsymbol{v}^T\boldsymbol{v}}$$

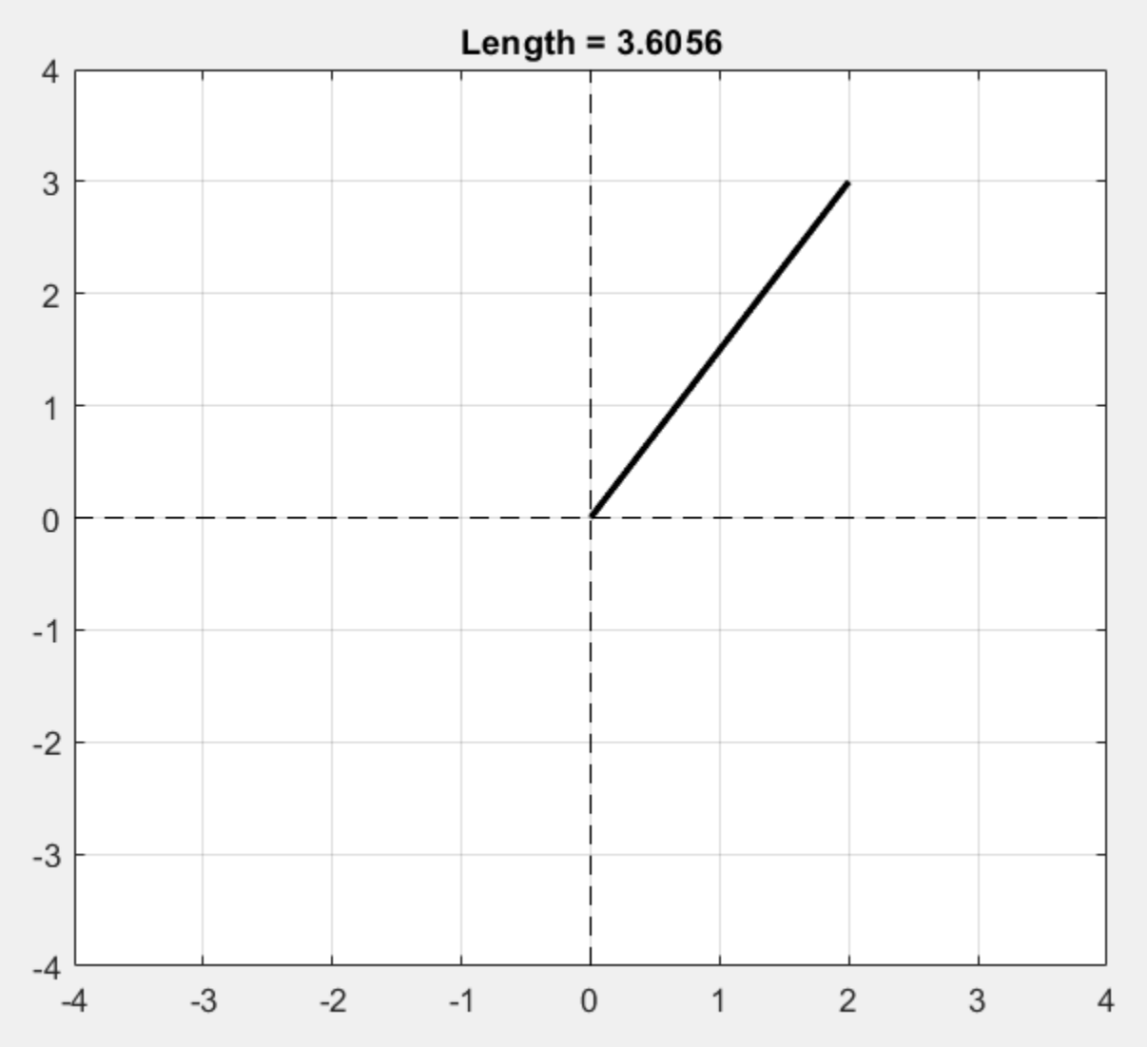
Where:

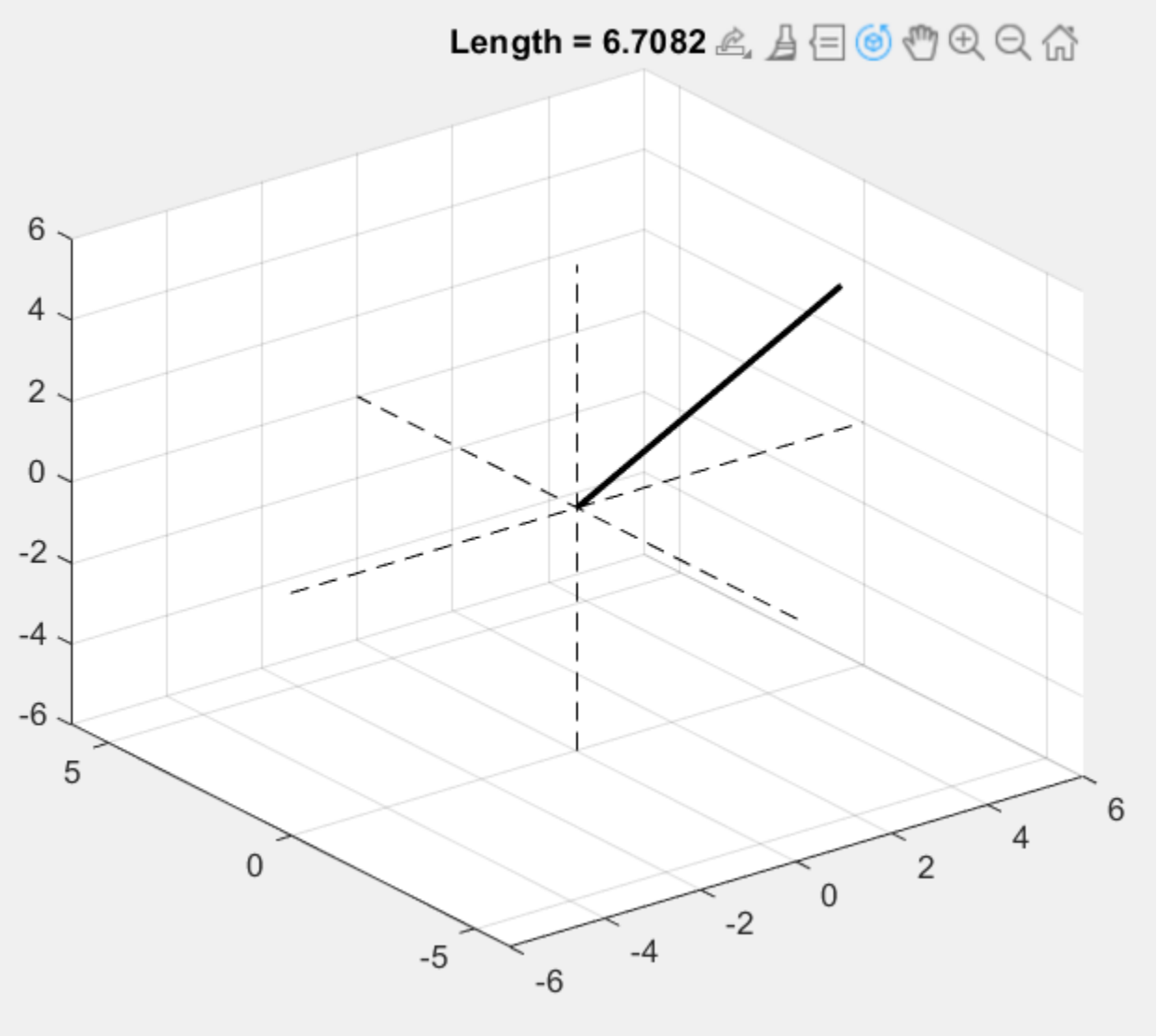
$$\boldsymbol{v}_{2d} = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$$

$$\boldsymbol{v}_{3d} = \begin{bmatrix} 4 \\ -2 \\ 5 \end{bmatrix}$$

### MATLAB

Code: PlotVectorsAndLengths_Lec125.m


Length = 3.6056
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4




## 126. HERMITIAN VS. REGULAR TRANSPOSE

Use MATLAB to understand what the “Hermitian transpose” means and how to implement the Hermitian vs. the regular transpose.

Use MATLAB to understand why the Hermitian transpose is necessary for complex vectors (hint: compute the length of [0 i]).

Skills: transpose, complex

## Transpose:

Exchange rows and columns of a matrix.

**Hermitian transpose:**

Transpose + complex conjugate

**Complex conjugate:**

[a+ ib] -> [a - ib]

MATLAB

Code: MasterMATLAB_2080_Hermitian.m

## 127. CREATE A SYMMETRIC MATRIX: ADDITIVE

Create a nonsymmetric square matrix **A**. Make it symmetric by computing $\mathbf{S=A+A^T}$. Demonstrate the symmetry by showing that $\mathbf{S==S^T}$

Use MATLAB to understand why $\mathbf{S=(A+A^T)}/2$ might be a better implementation.

Skills: transpose, reshape

MATLAB

Code: MasterMATLAB_2100_symmetricAdd.m

Original Nonsymmetric Matrix:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 2 & 0 \\ 9 & 2 & 5 \end{bmatrix}$$

Symmetric Matrix:

$$(A + A^T)/2 = \begin{bmatrix} 1 & 3 & 6 \\ 3 & 2 & 1 \\ 6 & 1 & 5 \end{bmatrix}$$

## 128. CREATE A SQUARE SYMMETRIC MATRIX: MULTIPLICATIVE

Create a rectangular matrix **A**. Make it symmetric by computing **S**=**AA**$^T$. Demonstrate the symmetry by showing that **S**==**S**$^T$

Make images of the matrices to determine whether **AA**$^T$ is the same thing as **A**$^T$**A**

Skills: transpose, mtimes, imagesc

### MATLAB

Code: MasterMATLAB_2120_symmetricMult.m

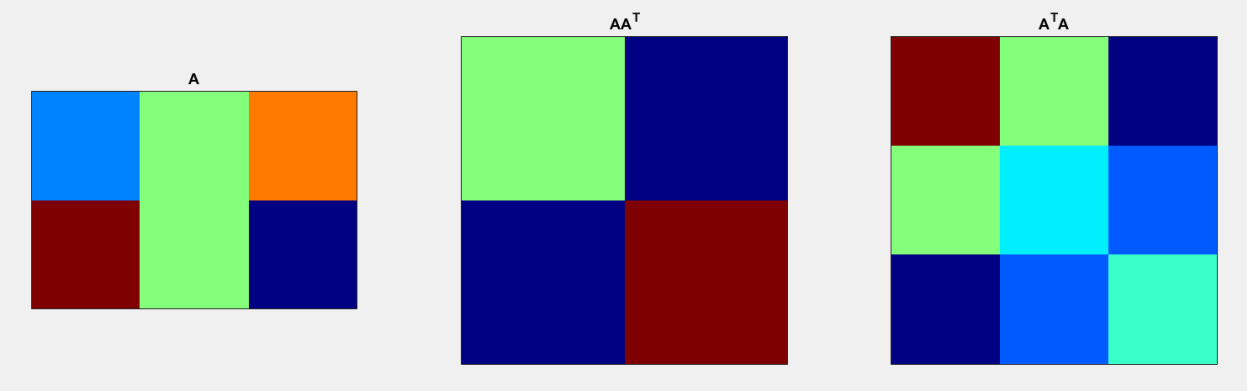



Figure 1: imagesc shows original matrix and both symmetric matrices. Colors help show symmetry.

## 129. MXM MATRIX WITH RANK M-1

Create an MxM matrix with rank=M. Replace one column with a linear combination of other columns to produce rank M-1.

Use bsxfun to generate and apply a random linear weighting of columns.

Skills: rank, randn, bsxfun, repmat

Rank: Number of independent columns in a matrix.

$\begin{bmatrix} 1 & 2 \\ 1 & 3 \end{bmatrix}$ $\begin{bmatrix} 1 & 2 \\ 1 & 2 \end{bmatrix}$ $\begin{bmatrix} 1 & 2 & 4 \\ 1 & 3 & 5 \end{bmatrix}$

Rank = 2 Rank = 1 Rank = 2

$\begin{bmatrix} a & \cdots & b \\ & \cdots & \\ c & \cdots & d \end{bmatrix}$ $\begin{bmatrix} a & \cdots & qa \\ & \cdots & \\ c & \cdots & qc \end{bmatrix}$

MxM MxM

Rank = M Rank = M - 1

MATLAB

Code: MasterMATLAB_2140_matrixRankM1.m

## 130. MXN MATRIX WITH RANK R VIA SVD

Create an MxN random-number matrix, and compute its Singular Value Decomposition (SVD).

Reconstruct a new MxN matrix with rank r<M,N from the SVD.

Skills: svd, norm

Overview of SVD process:

**A = U * S * $V^T$**

$\begin{bmatrix} \quad \end{bmatrix} = \begin{bmatrix} \quad \end{bmatrix} \begin{bmatrix} \quad & 0 \\ 0 & \quad \end{bmatrix} \begin{bmatrix} \quad \end{bmatrix}$

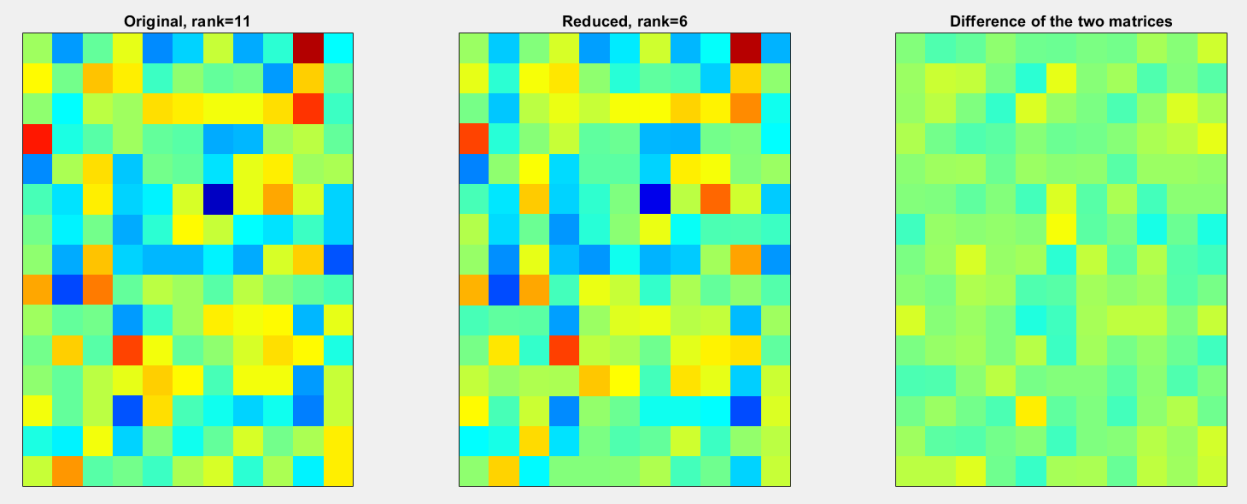
mxn = mxm * mxn * nxn

Where matrix S is diagonal and:

Rank: $r=\sum(\sigma>0)$

MATLAB

Code: MasterMATLAB_2160_matrixRankSVD.m




## 131. CREATE A RANDOM HANKEL MATRIX

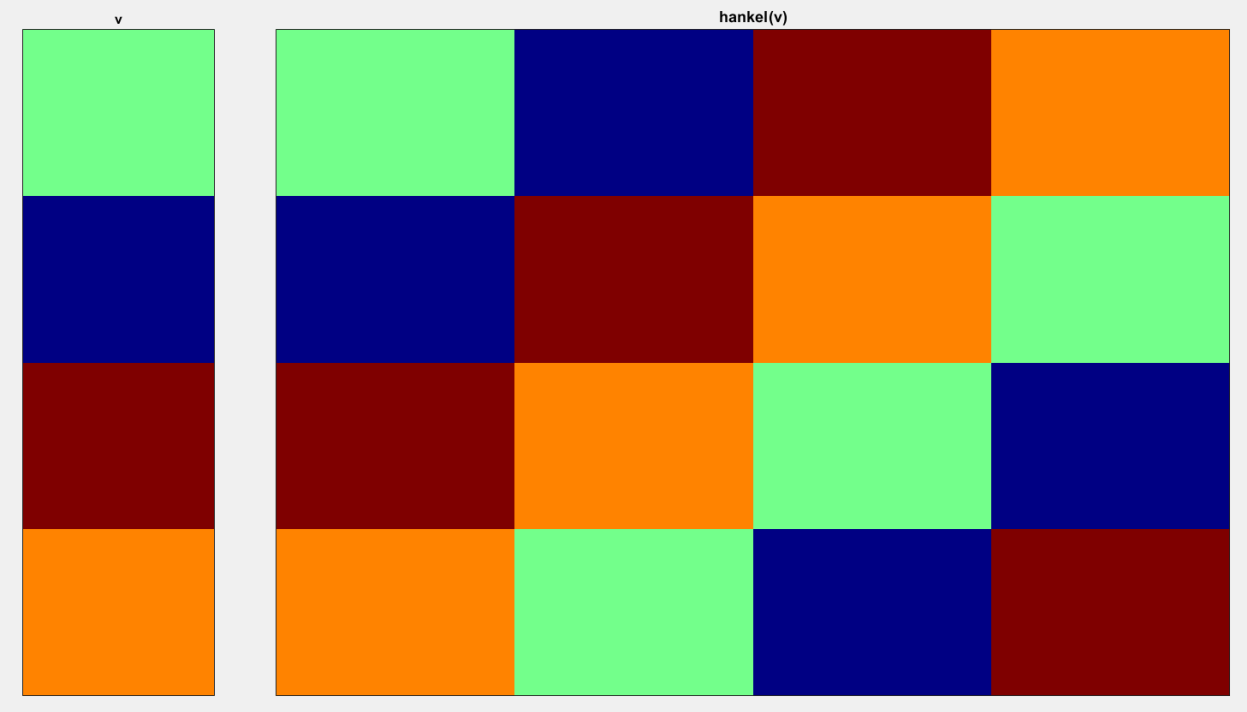
Create an MxM "Hankel" matrix based on a vector of random numbers. Use a loop and the MATLAB function *hankel*.

Implement the element-wise formula.

Skills: hankel

MATLAB

Code: MasterMATLAB_2180_HankelMatrix.m



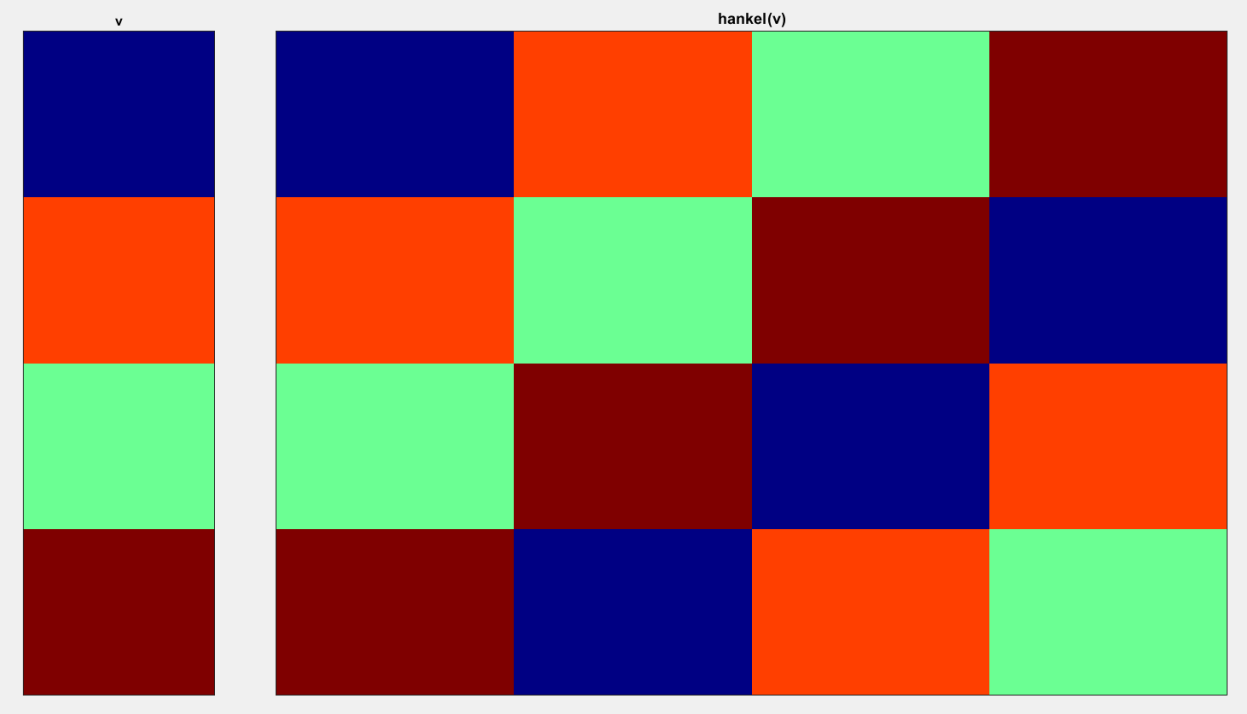

## 132. SOLVED: ELEMENT-WISE HANKEL MATRIX WITH MOD FUNCTION

Use the *mod* function to generate element-wise Hankel matrix

Skills: mod, interp1

### MATLAB

Code: HankelMatrix_ElementWise_Mod_132.m




## 133 SOLVED: CREATE A TOEPLITZ MATRIX

Create an element-wise Toeplitz matrix. Compare results with MATLAB *toeplitz* function.

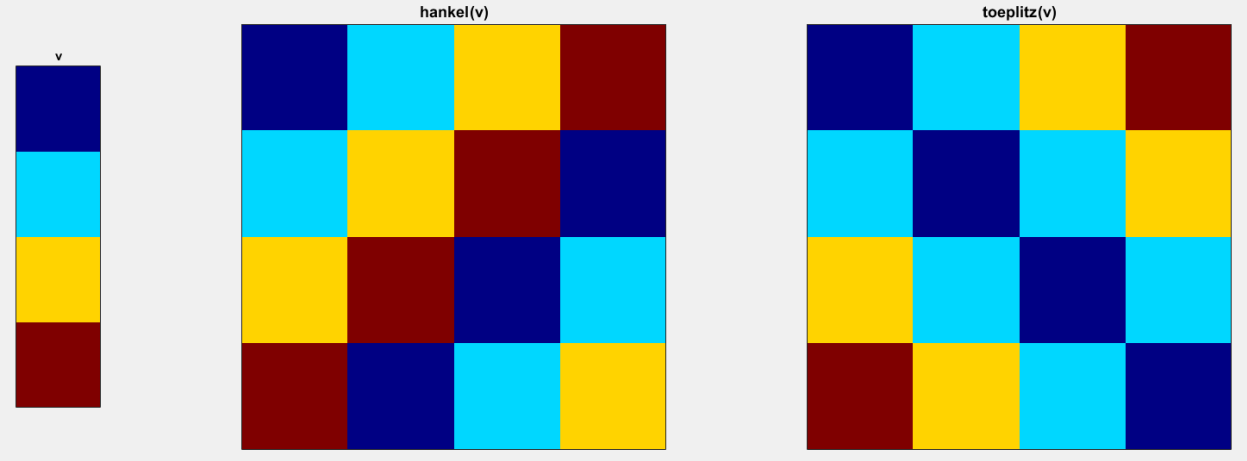
Skills: toeplitz, reshape, subplot, axis image, axis square

### MATLAB:

Code: ToeplitzMatrix_vs_hankel_133

One of two test results reported in command widow:

MATLAB toeplitz equal to calculated version




## 134. EIGENVECTORS OF A HANKEL MATRIX

Create an MxM Hankel matrix. Extract its eigenvectors and eigenvalues. Sort eigenvectors according to descending eigenvalues, and visualize.

Compute the frequency of the eigenvectors, plot the frequency spectrum, and image the frequency-sorted eigenvectors.

Skills: hankel, eig, sort, diff

### MATLAB

Code: MasterMATLAB_2200_HankelEig.m

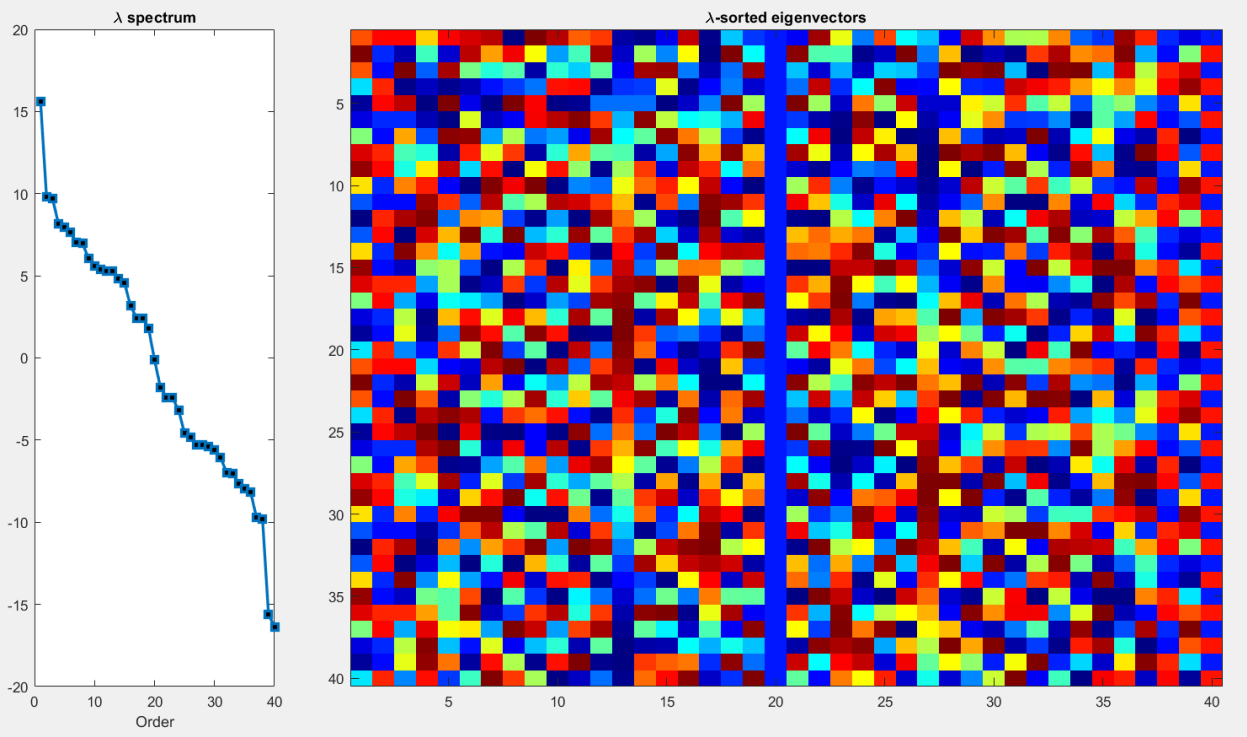



Figure 1

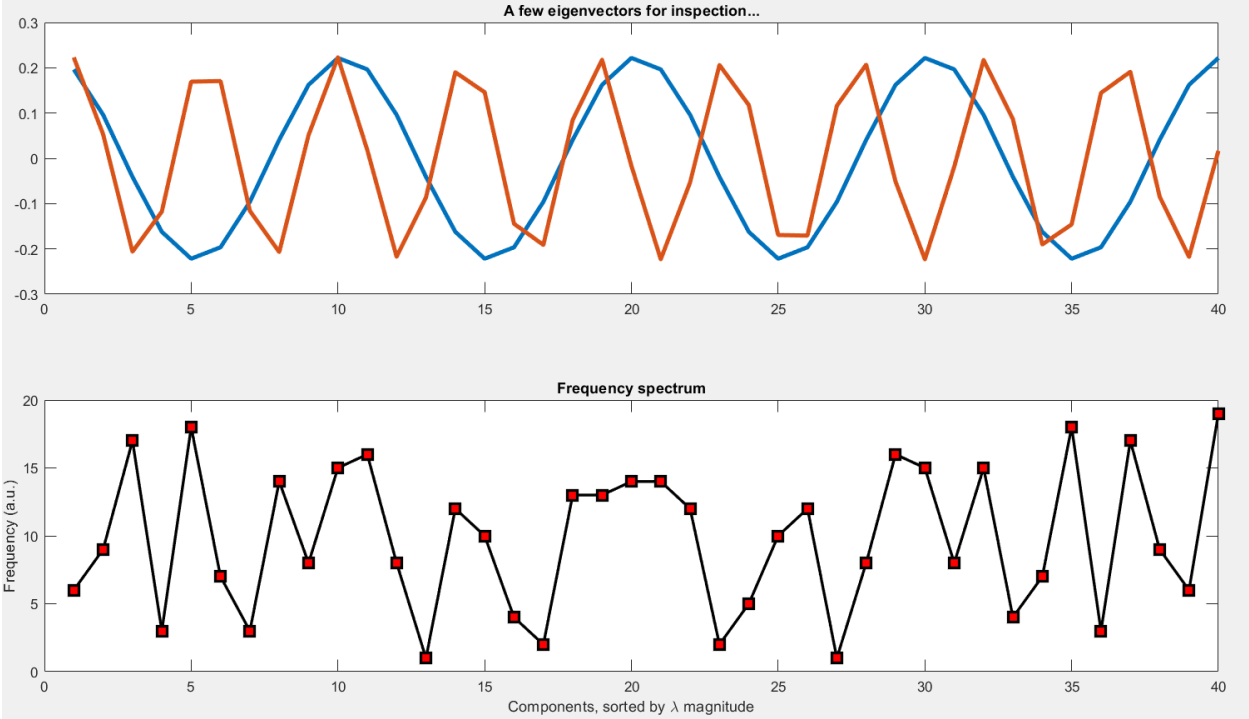



Figure 2

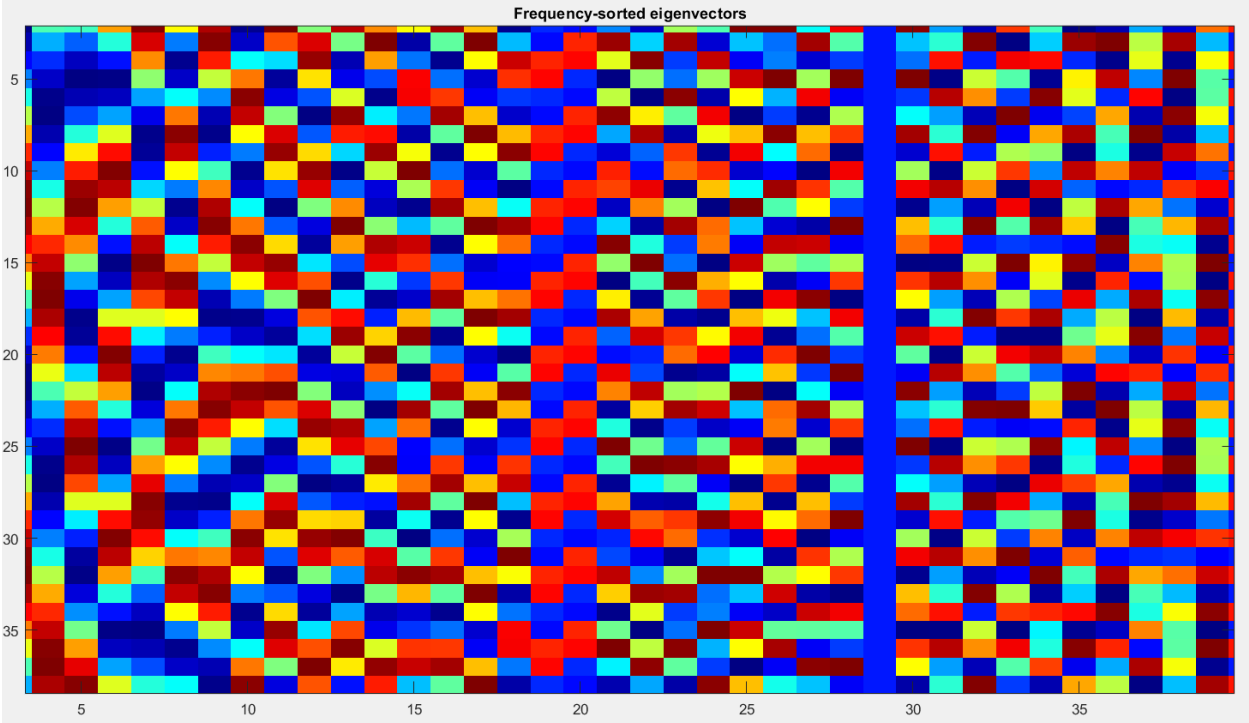



Figure 3

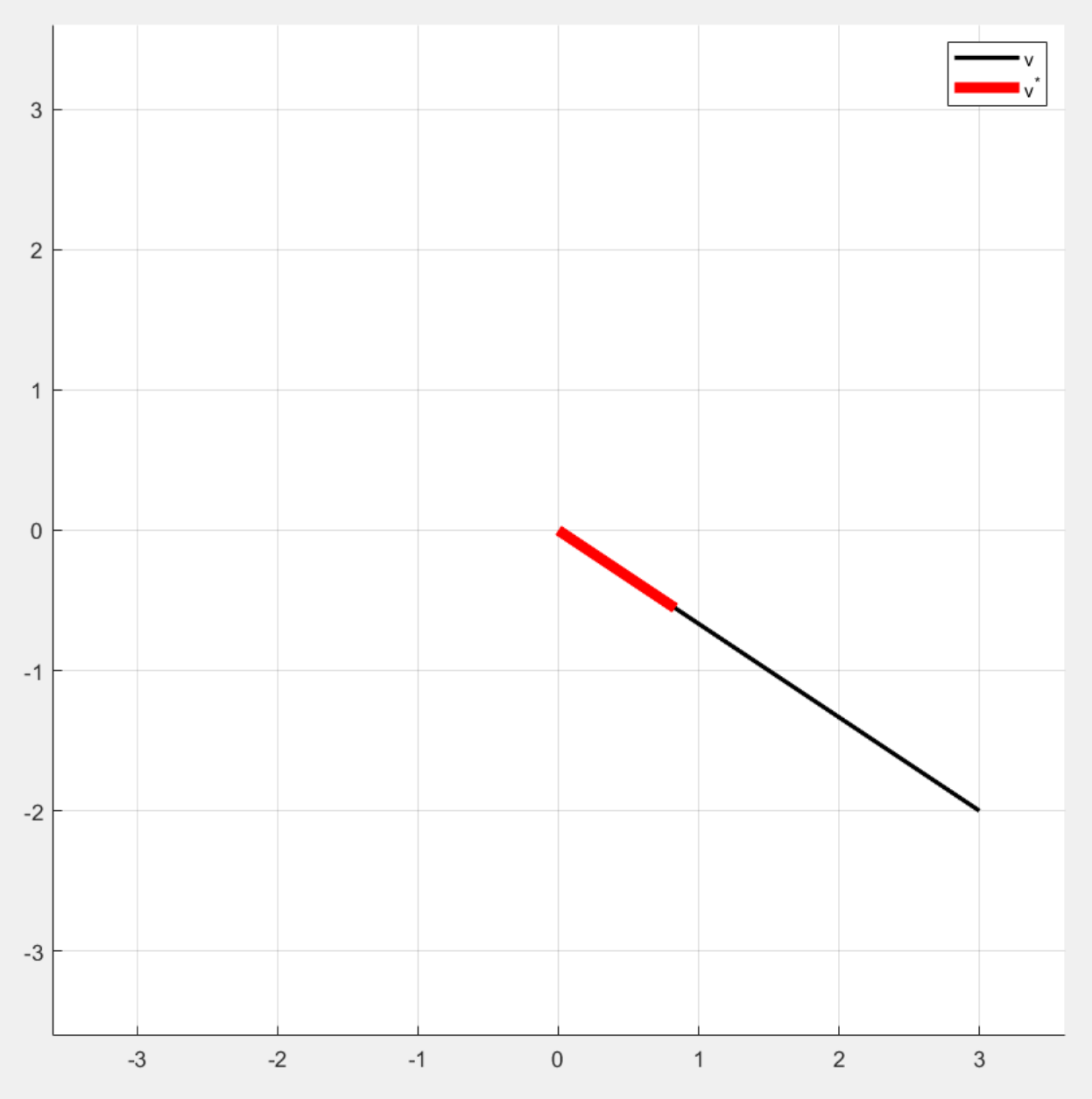
## 135. COMPUTE A UNIT VECTOR IN SOME DIRECTION

Create an 2x1 vector. Compute the magnitude (*norm*) of the vector, and create a unit vector in that same direction by scaling the vector by tis norm. Plot both vectors and confirm that the new vector has unit length.

Skills: norm, plot

### MATLAB

Code: MasterMATLAB_2220_unitVector.m




## 136. ORTHOGONALIZE A PAIR OF VECTORS

Decompose a vector **v** into an orthogonal component, and a parallel component, relative to vector **w**.

Plot the original and decomposed vectors.

Skills: norm, dot

Separating a vector into parts

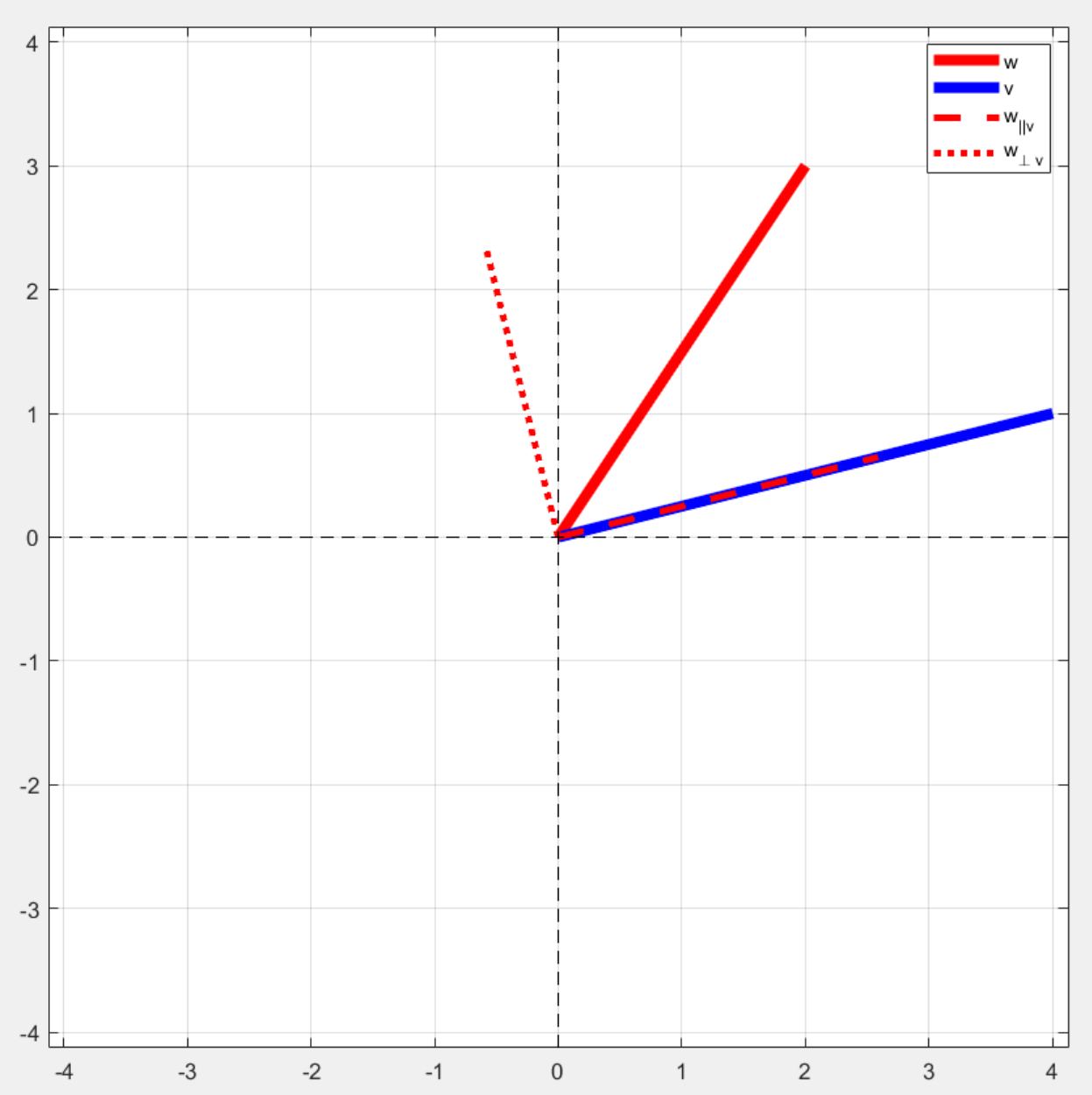
$$w = w_{\|v} + w_{\perp v}$$

$$w_{\|v} = proj_{\perp v} w = \frac{w^T v}{v^T v} v$$

$$w_{\perp v} = w - w_{\|v}$$

MATLAB

Code: MasterMATLAB_2221_orthogonalVects.m




## 127. GRAM-SCHMIDT ALGORITHM

Create an MxN matrix of random integers. Apply the Gram-Schmidt orthogonalization algorithm, and show that the result is an orthogonal matrix.

Obtain the same result using QR decomposition.

Skills: qr, norm, dot, round

Procedure for creating a set of orthonormal vectors:

1) Start with $\mathbf{v_1}$ and normalize to unit length: $\mathbf{v}_1^* = \frac{\mathbf{v_1}}{|\mathbf{v}_1|}$
2) For all remaining vectors:
    a. Orthogonalize $\mathbf{v}_n^*$ to all previous vectors
    b. Normalize $\mathbf{v}_n^*$ to unit length

## MATLAB

Code: MasterMATLAB_2240_GramSchmidt.m

Code results:

```
>> MasterMATLAB_2240_GramSchmidt

Gram Schmidt orthogonalization algorithm...

Test that dot product between columns are zero
sum( GS(:,1).*GS(:,2) ) = 3.4694e-17

Test that dot product between a column and itself is 1
sum( GS(:,3).*GS(:,3) ) = 1

Test that multiplication produces a diagonal matrix
GS'*GS =
   1.0000    0.0000   -0.0000    0.0000
   0.0000    1.0000   -0.0000   -0.0000
  -0.0000   -0.0000    1.0000   -0.0000
   0.0000   -0.0000   -0.0000    1.0000


QR decomposition algorithm...

Test that multiplication produces a diagonal matrix
GS'*GS =
   1.0000   -0.0000   -0.0000    0.0000
  -0.0000    1.0000   -0.0000   -0.0000
  -0.0000   -0.0000    1.0000    0.0000
   0.0000   -0.0000    0.0000    1.0000
```

## 138. MATRIX INVERSE VIA QR DECOMPOSITION

Invert a square matrix via QR. Visualize the matrix, its inverse (via QR and via *inv*), and their product.

Skills: qr, norm, dot, round, inv, subplot

## Properties of Q and R from MATLAB *qr* function:

$$\mathbf{A} = \mathbf{QR}$$

$$\mathbf{A}^{-1} = (\mathbf{QR})^{-1}$$

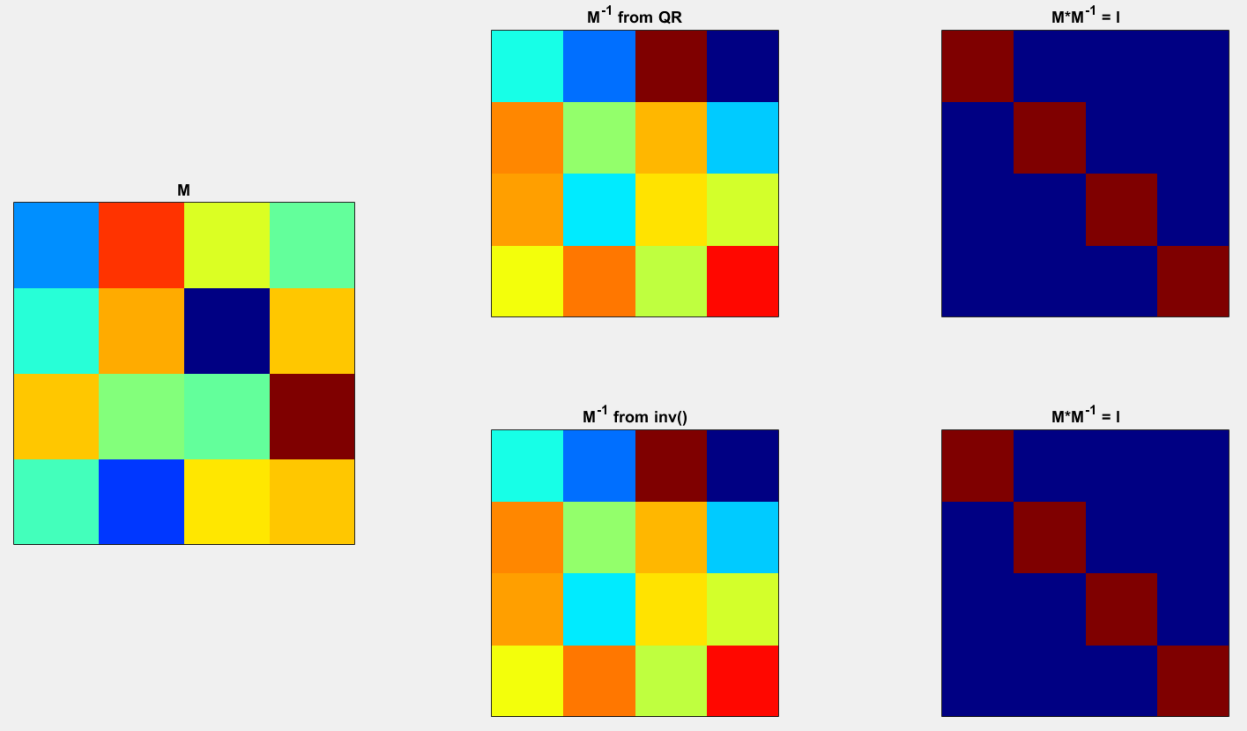
$$\mathbf{A}^{-1} = \mathbf{R}^{-1}\mathbf{Q}^{-1}$$

$$\mathbf{A}^{-1} = \mathbf{R}^{-1}\mathbf{Q}^{T}$$

Try to avoid using MATLAB *inv* as it can be unstable. *inv* is reliable only when matrix is square and of full rank.

### MATLAB

Code: MasterMATLAB_2260_QRinverse.m




## 139. VISUALIZE THE QUADRATIC FORM OF A 2X2 MATRIX

Create and display a surface of the quadratic form of a 2x2 matrix.

Normalize the quadratic form according to the magnitude of vector w used to compute the quadratic form.

Skills: surf, , shading interp

This project is at the intersection of:

- Mathematics
- MATLAB Programming
- Data Visualization
- Art

## Quadratic Form of a 2x2 Matrix:

$$Q = \mathbf{w}^T\mathbf{A}\mathbf{w}$$

Example:

$$Q_{1,2} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}^T \begin{bmatrix} 0 & 1 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} = 24$$

Normalized Quadratic Form:

$$Q = \frac{\mathbf{w}^T \mathbf{A} \mathbf{w}}{\mathbf{w}^T \mathbf{w}}$$

MATLAB

Code: MasterMATLAB_2280_quadForm.m

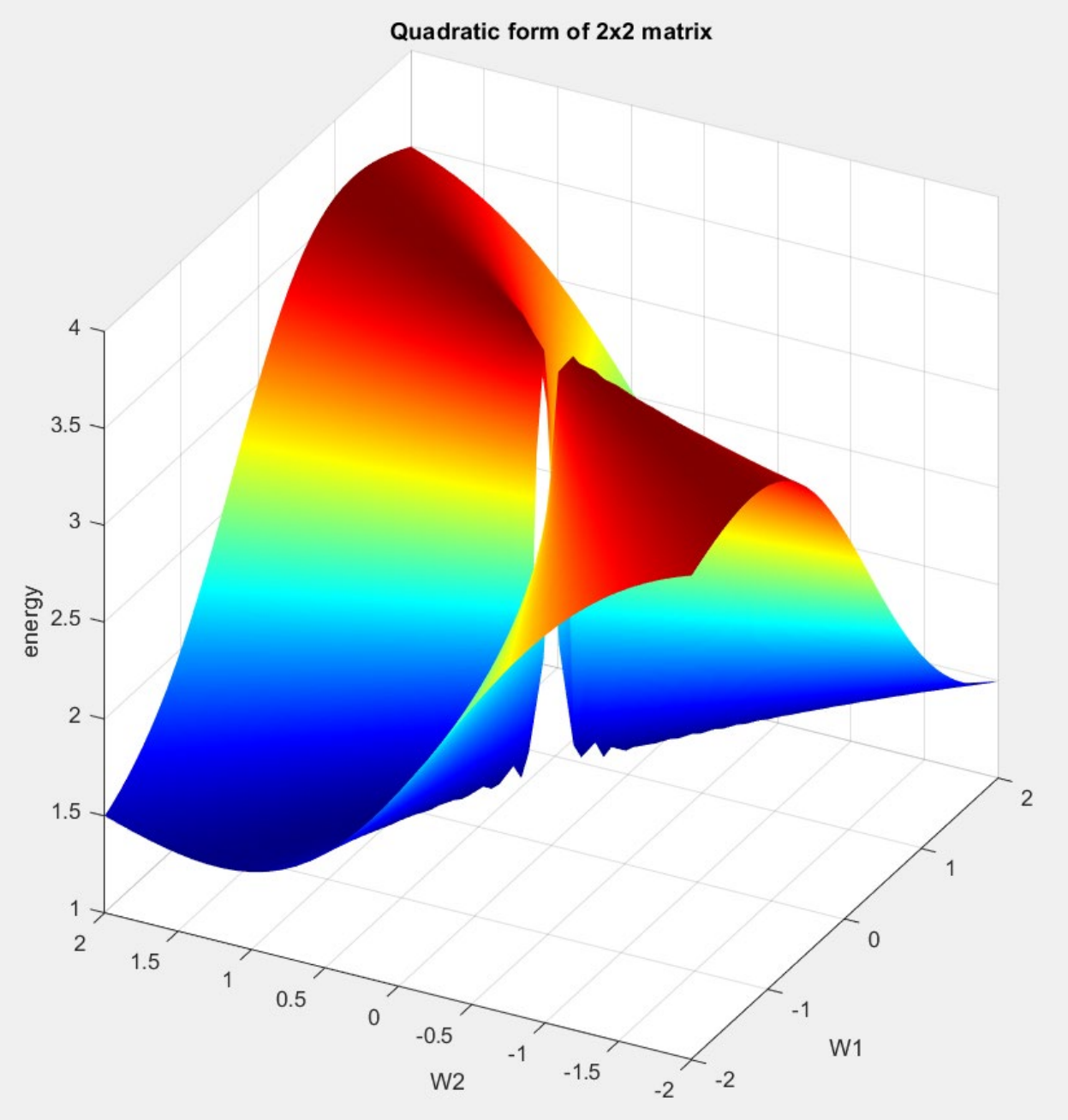



Figure 1: Normalized Quadratic Form of 2x2 matrix

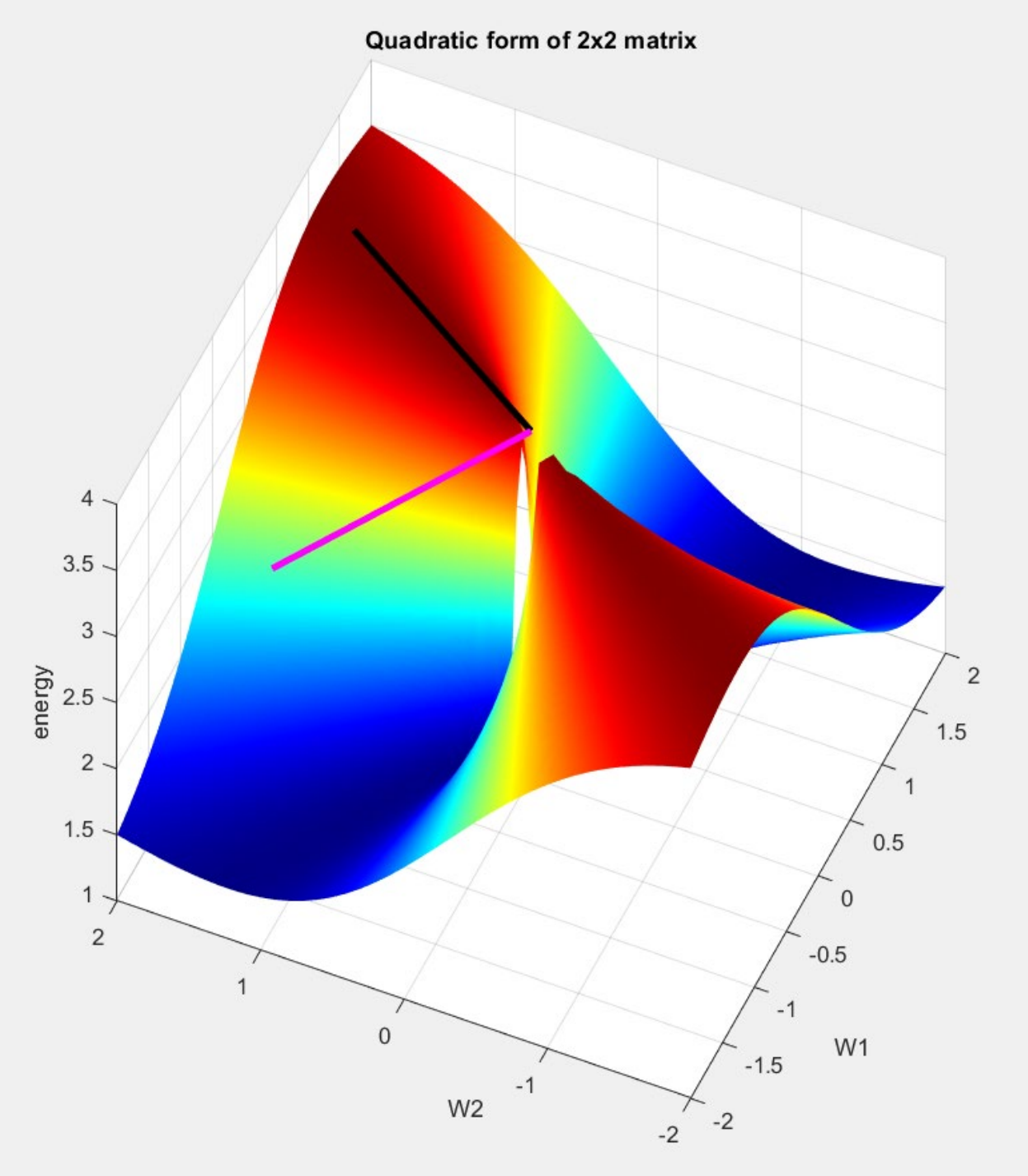
## 140 EIGENVECTORS AND QUADRATIC FORM

Compute the eigenvectors of a 2D matrix and display those vectors on top of the quadratic form.

Skills: surf, shading, eig

MATLAB

Code: MasterMATLAB_2281_quadFormEig.m



## 141. PCA OF LOW-RANK SPACE-TIME DATA

Create 3D time-space data as the combination of three 2 2D patterns (linear Horizontal, linear vertical, Gaussian) that fluctuate randomly over time. Perform Principle Component Analysis (PCA) and show the eigenspectrum and first three component maps.

Skills: eig, meshgrid

PCA is the eigen decomposition of the covariance matrix.

See lecture “105. SPATIAL SMOOTHING ON A GRID OF CHANNELS” for more space-time discussion. Add random noise to the space-time dataset. Mean-smooth the data using a 2D raised kernel. Show a movie of the noisy and smoothed datasets.

### MATLAB

Code: MasterMATLAB_2300_PCA_lowRank.m

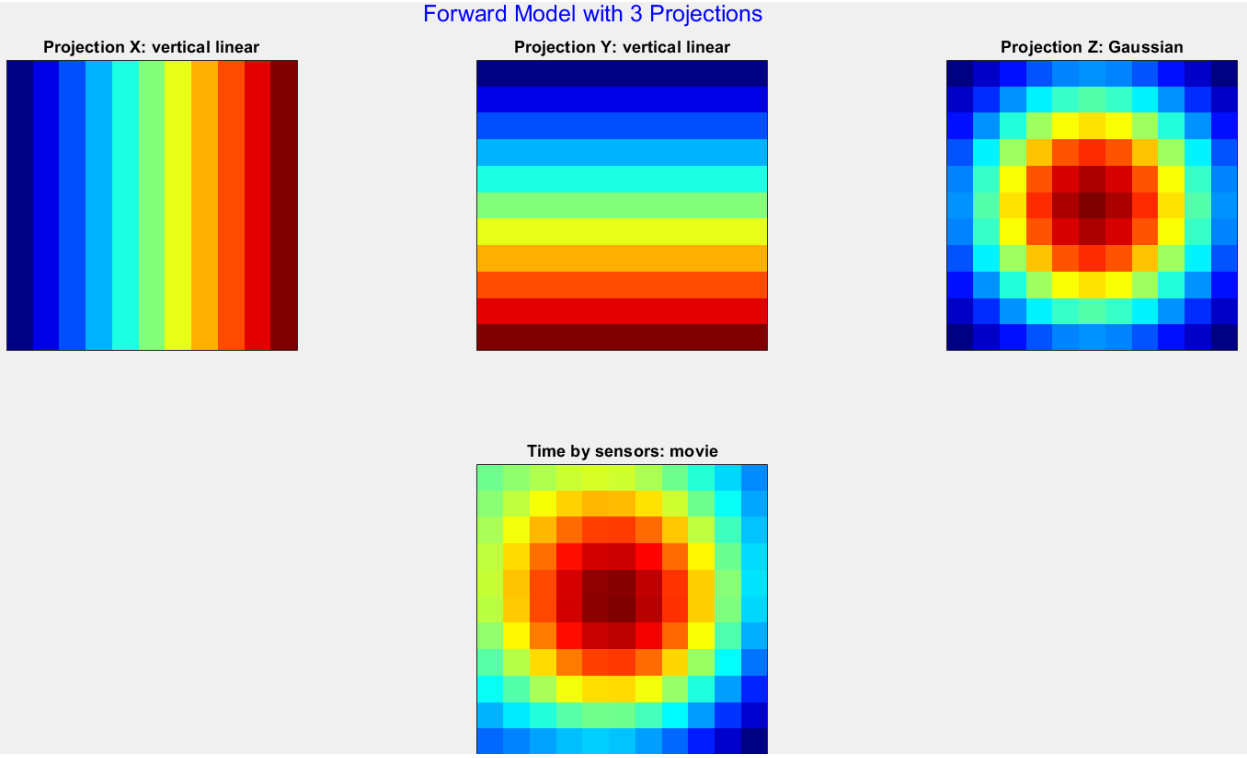



Figure 1

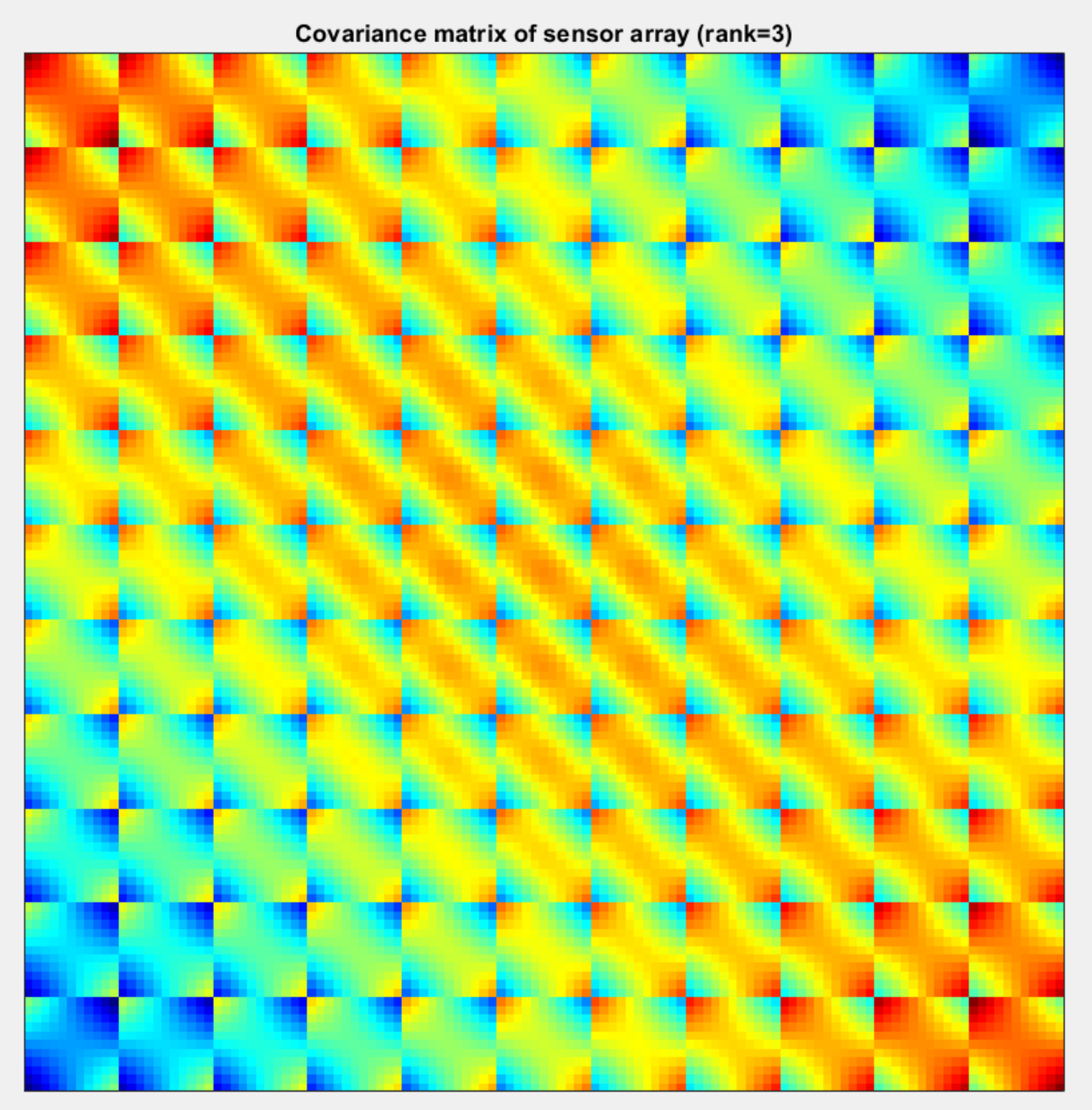



Figure 2

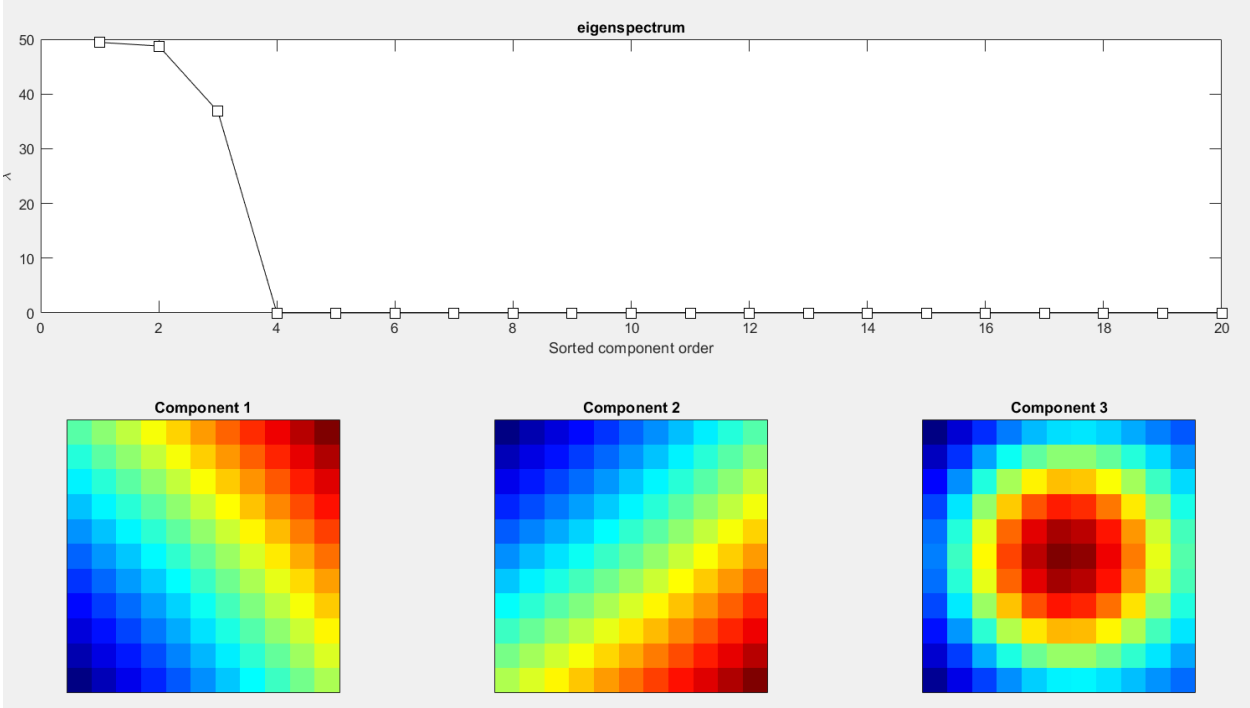



Figure 3

## 142. COVARIANCE SHRINKAGE REGULARIZATION IN PCA

Apply covariance “shrinkage” and re-implement PCA. Observe the effects of 0-10% shrinkage on eigenvalues and eigenvectors.

Skills: eig, meshgrid, sort, reshape

Shrinkage regularization is often used:

- Machine learning
- Artificial intelligence
- Classification analysis
- Dimensionality reduction
- Impose extra smoothing on a multi-variate problem

# Formula for Covariance shrinkage:

$$\mathbf{C} = \mathbf{X}^T\mathbf{X}$$

Where:

**C** is the Covariance matrix

**X** is the Data matrix (must be mean centered)

Alternate version of Covariance, $\tilde{\mathbf{C}}$:

$$\tilde{\mathbf{C}} = (1 - \gamma)\mathbf{C} + \alpha\mathbf{I}$$

Where:

$\mathbf{I}$ is the identity matrix

$\gamma$ is the amount of shrinkage

$\gamma \in [0, 1]$

$\alpha = \gamma\overline{\lambda}$

$\overline{\lambda}$ is the average of all eigenvalues of the original matrix: **C**

MATLAB

Code: MasterMATLAB_2320_shrinkPCA.m

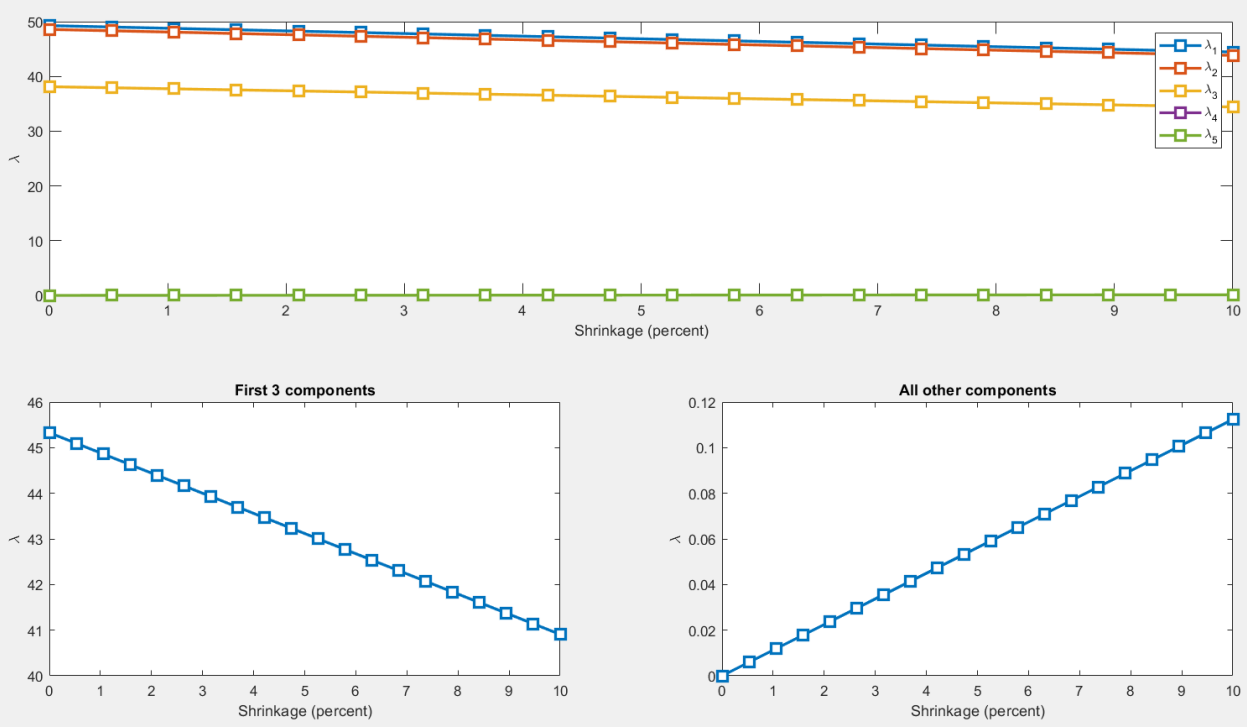

λ
Shrinkage (percent)
λ1
λ2
λ3
λ4
λ5
First 3 components
λ
Shrinkage (percent)
All other components
λ
Shrinkage (percent)

Figure 1

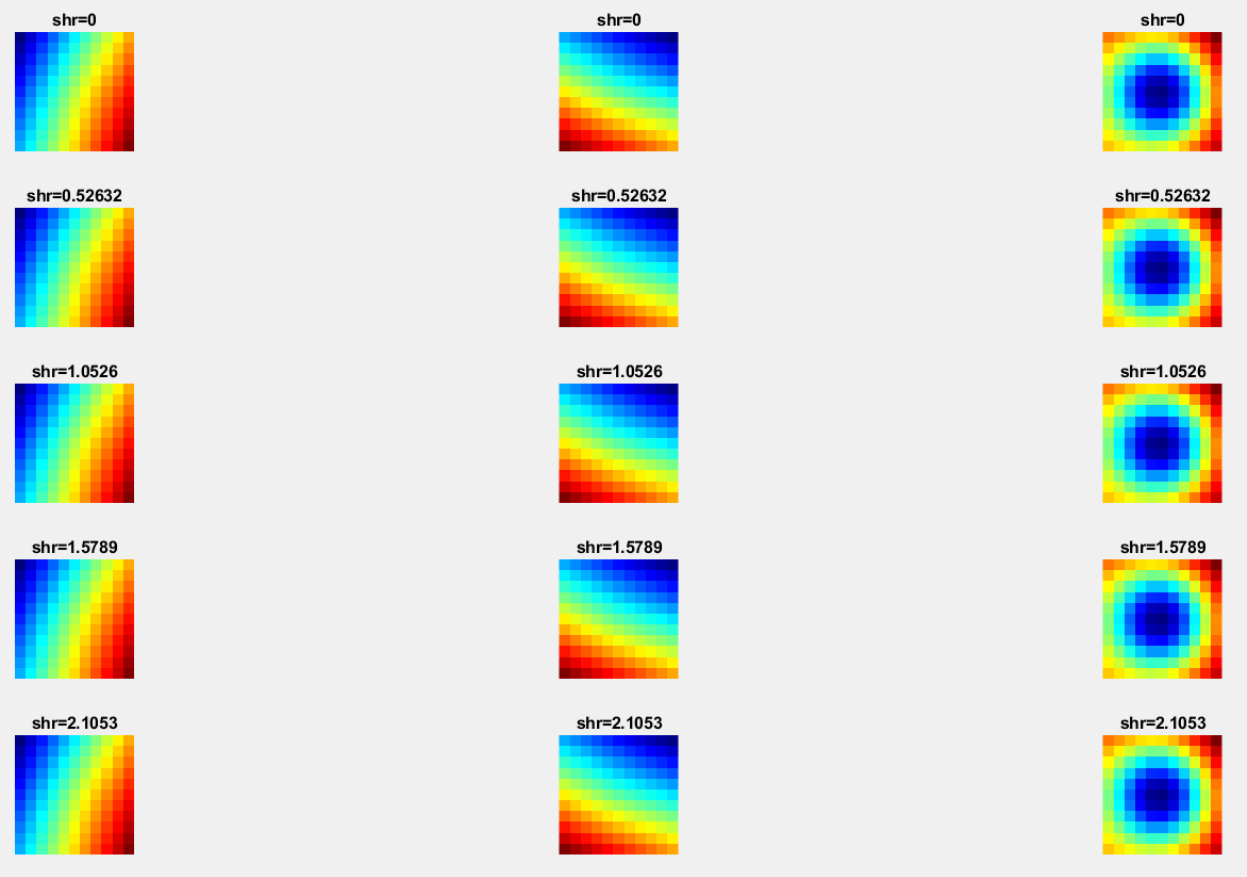



Figure 2

## SECTION 21: CIRCULAR DISTRIBUTIONS AND ANALYSES

### 143. DRAW A VECTOR IN POLAR PLANE

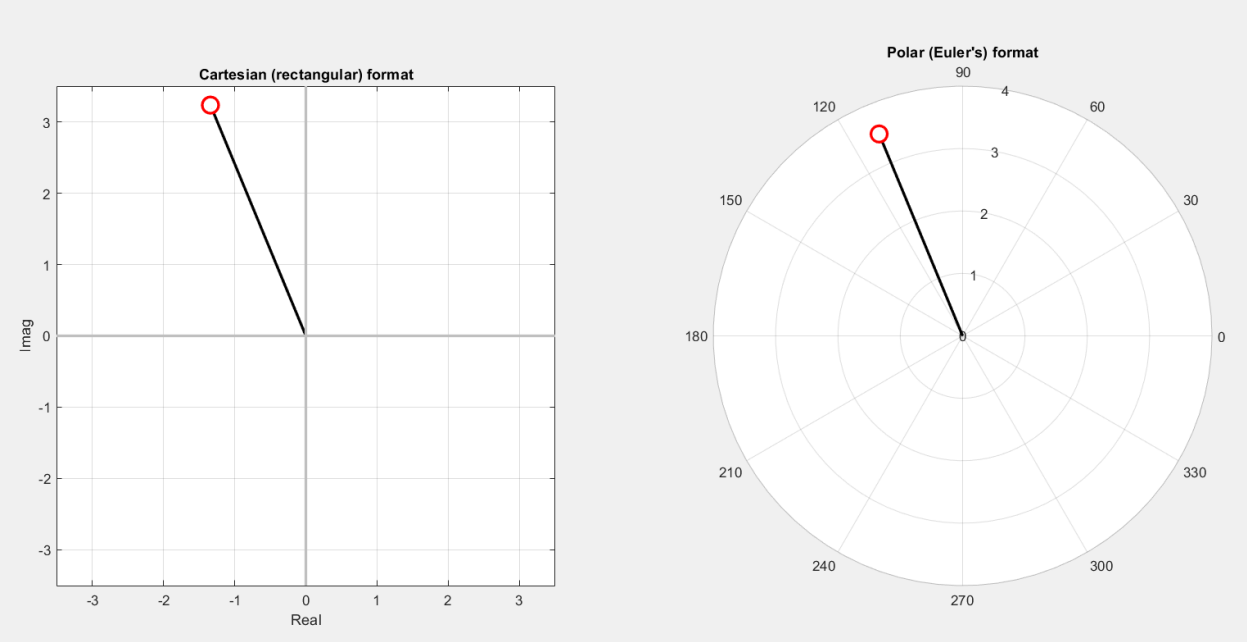
Draw a complex vector in a Cartesian and a polar plot.

Use *polarplot* and adjust the radial axis limit.

Skills: polar, polarplot, real, imag, angle, abs

#### MATLAB

Code: MasterMATLAB_2340_polarVector.m




## 144. CIRCULAR HISTOGRAM

Generate a sinusoidal and non-sinusoidal time series. Extract the phase time series and show in a circular histogram.

Skills: sin, Hilbert, angle, rose, polarplot

Phase uniform signal:

$$f = \sin(t)$$

Phase non-uniform signal:

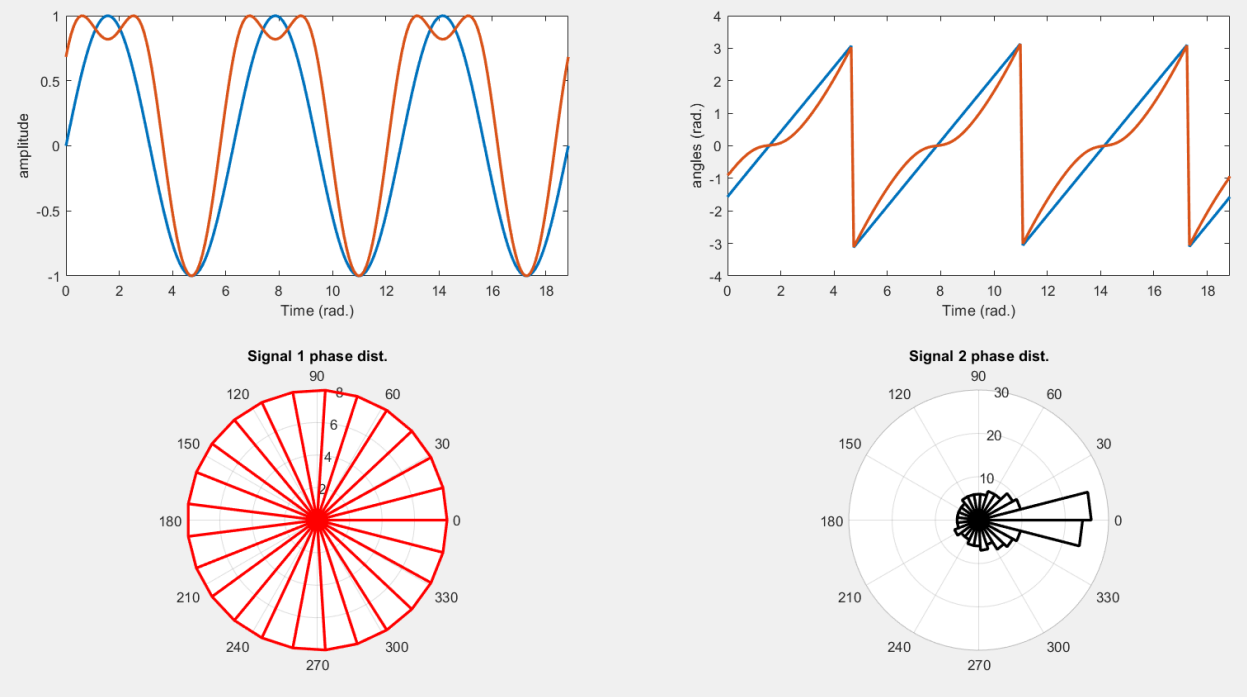
$$f = 2\sin(1 + sin(t)) - 1$$

Where time is defined as:

$$t \in [0, 6\pi]$$

## MATLAB

Code: MasterMATLAB_2360_circularHistogram.m




## 145. COMPUTE AND PLOT MEAN VECTOR LENGTH

Generate uniform and nonuniform phase angles. Plot the phase angles as unit vectors in a polar plot. Compute the average vector and plot it on top. In the title, list the average vector length.

Skills: angle, Hilbert, sin, polarplot, num2str

Compute the mean vector:

$$z = n^{-1} \sum_{t=1}^{n} e^{i\phi_t}$$

Phase uniform signal:

$$f_1 = \sin(t)$$

Phase non-uniform signal:

$$f_2 = \sin(.9 + sin(t))$$

Where time is defined as:

$$t \in [0, 6\pi]$$

### MATLAB

Code: MasterMATLAB_2380_meanVectLength.m


## 146. PHASE DIFFERENCE BETWEEN TWO DISTRIBUTIONS

Generate two signals and extract their phase angle time series. Compute the mean phase angle difference and plot.

Dynamically update the plot title using handles

Skills: angle, Hilbert, sin, polarplot, set, get

A measure of phase synchronization between two oscillators. A measure often used in physics and biology such as neuroscience.

How to generate a phase angle time series: angles of the Hilbert transform at each time point. Extra care is needed when the signals are not mean centered.

## Compute the mean vector:

$$z = n^{-1} \sum_{t=1}^{n} e^{i(\phi_t - \theta_t)}$$

## Two distributions offset by a phase difference (π/5):

$$f_1 = \frac{(1 + \sin(t))^2}{4}$$

$$f_2 = \frac{(1 + \sin(t + \pi/5))^2}{4}$$

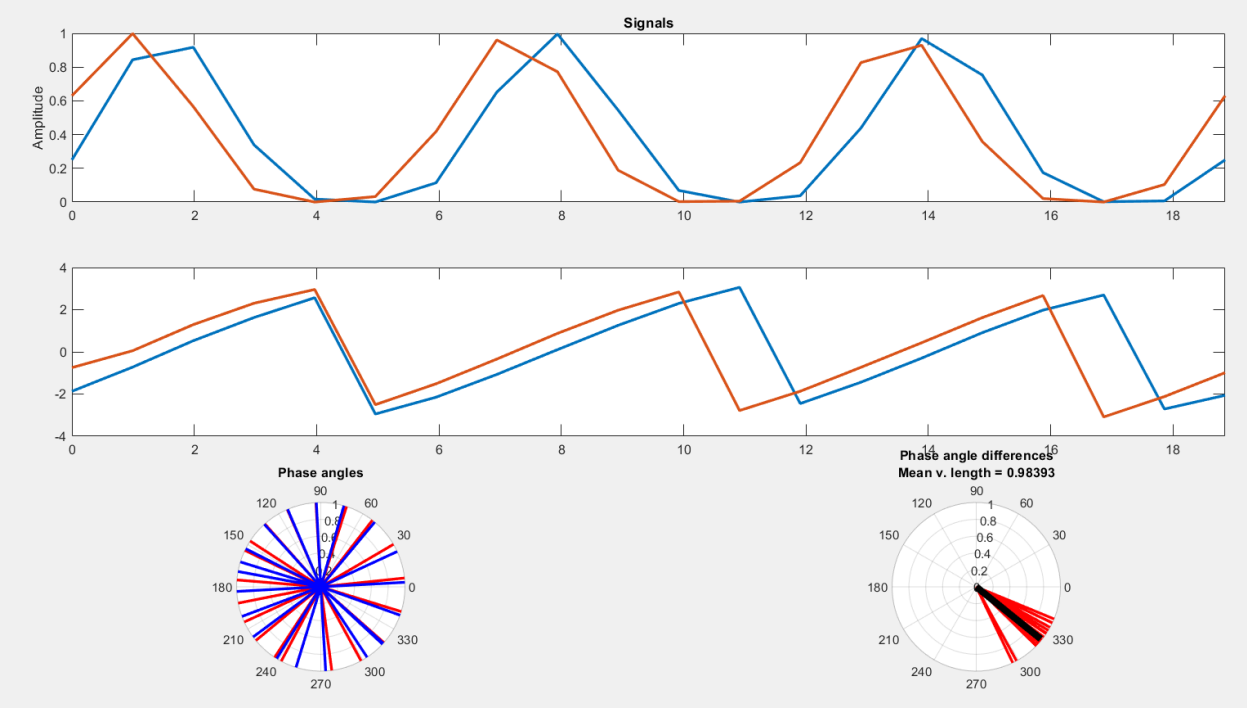
## Where time is defined as:

$$t \in [0, 6\pi]$$

---

MATLAB

Code: MasterMATLAB_2400_phaseDifferences.m




## SECTION 22: FRACTAL TIME SERIES AND IMAGES

### 147. THE SIERPINSKI TRIANGLE

Follow the formula to create a Sierpinski triangle.

Animate the dots. The triangle should be drawn from the bottom-up

Skills: Deal, sub2ind

A fractal is a scale free picture.

## Algorithm for Sierpinski Triangle:

for i in {2,3,…,N}

k ← random in {1,2,3}

$x_i \leftarrow .5x_{i-1} + k - 1$

$y_i \leftarrow .5y_{i-1}$

if k is 2

$y_i \leftarrow .5y_{i+1}$

## MATLAB

Code: MasterMATLAB_2420_Sierpinski.m

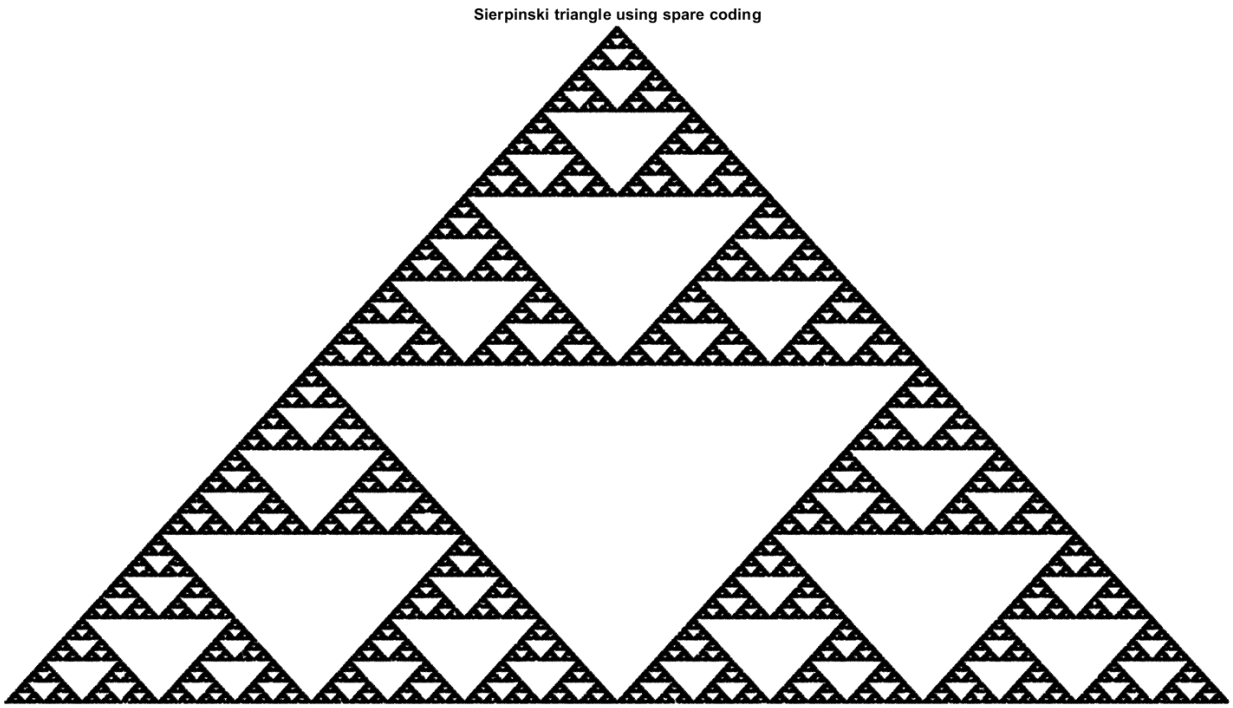



Figure 1

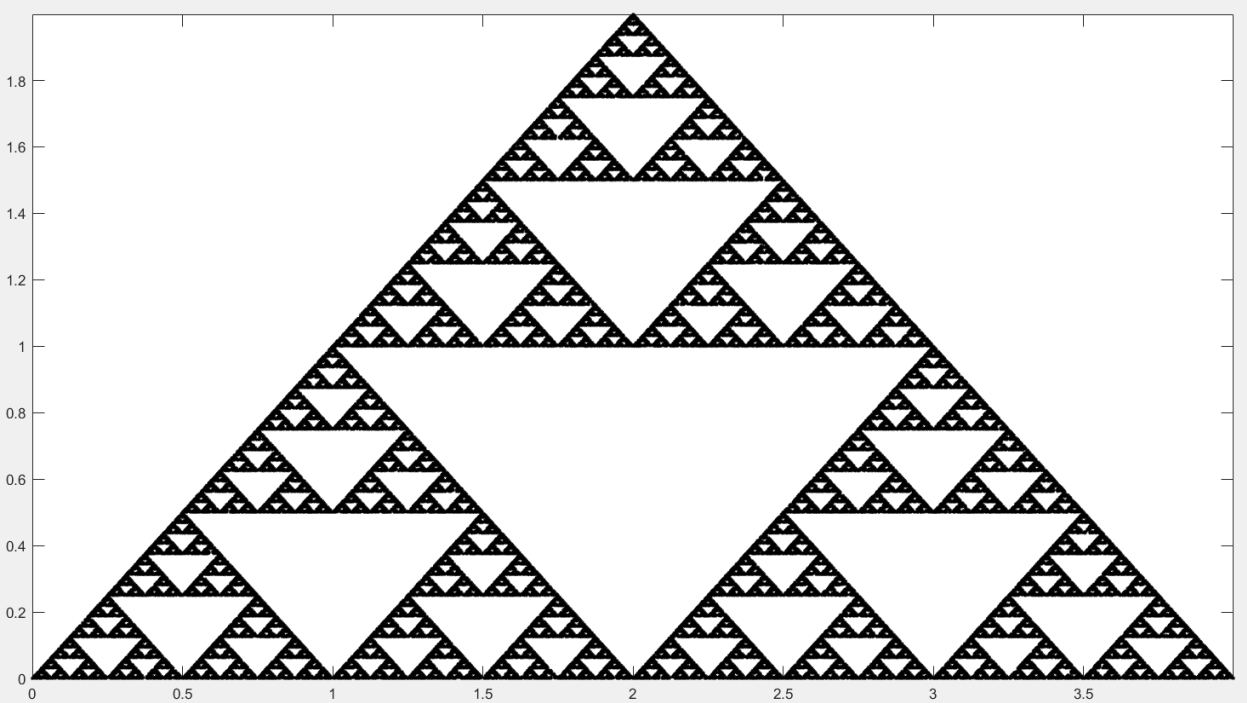



Figure 2

## 148. UNSOLVED: SIERPINSKI TRIANGLE AS DENSE MATRIX

Convert the sparse matrix (XY coordinates) into a full matrix.

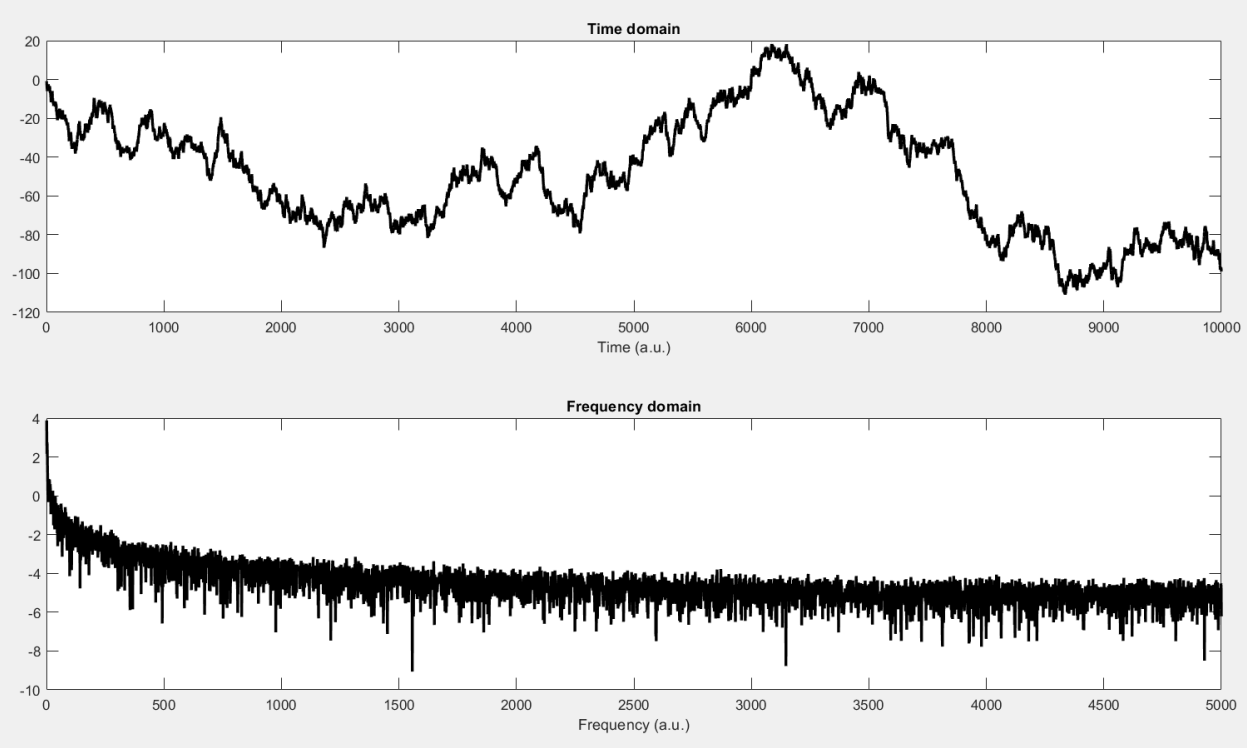
## 149. BROWNIAN MOTION

Create a Brownian motion time series by integrating (cumulative sum) normally distributed random noise. Plot in the time domain and frequency domain.

Skills: cumsum, randn, fft

### MATLAB

Code: MasterMATLAB_2440_BrownianMotion.m




Note how the log(fft) starts high and trends down over frequency. This is a characteristic of Brownian motion.

## 150. CANTOR SET AND DEVIL'S STAIRCASE

Create and plot Cantor's set and the associated devil's staircase.

From visual inspection, determine whether the derivative of the devil's staircase is also fractal.

Skills: repmat, reshape, plot

Note: each figure plotted is a fractal (aka scale free picture).

## Pseudo code to generate Cantor's Set and the Devil's Staircase:

$$x \leftarrow \{0, 1\}$$

Repeat k times:

$$x \leftarrow \{x/3, (x+2)/3\}$$

Where:

$k$: is the depth parameter

MATLAB

Code: MasterMATLAB_2460_devilStaircase.m

Figure 1: Cantor Set with depth k set to 11 (a fractal)

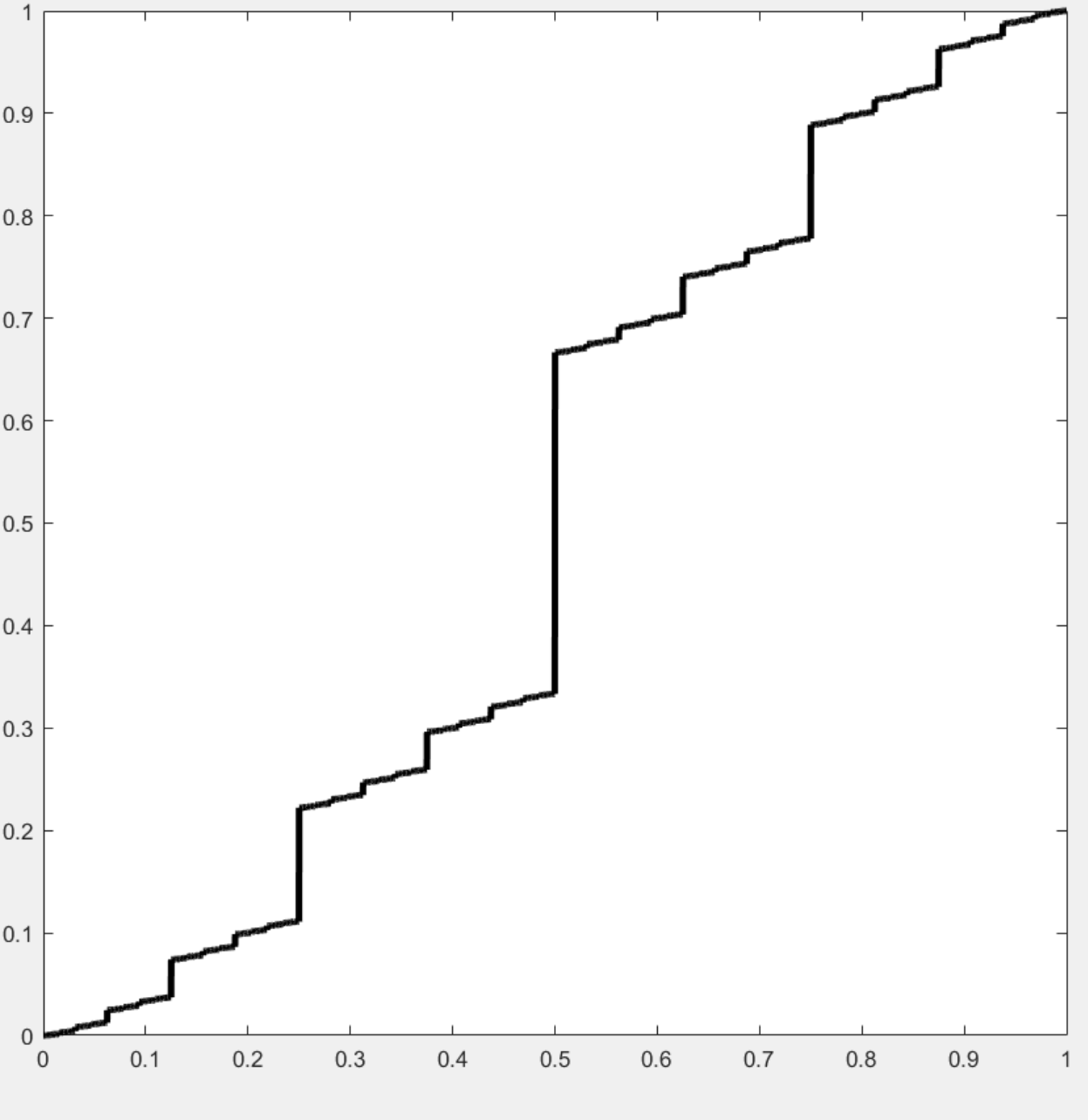



Figure 2: Devil's Staircase generated from Cantor's Set (also a fractal)

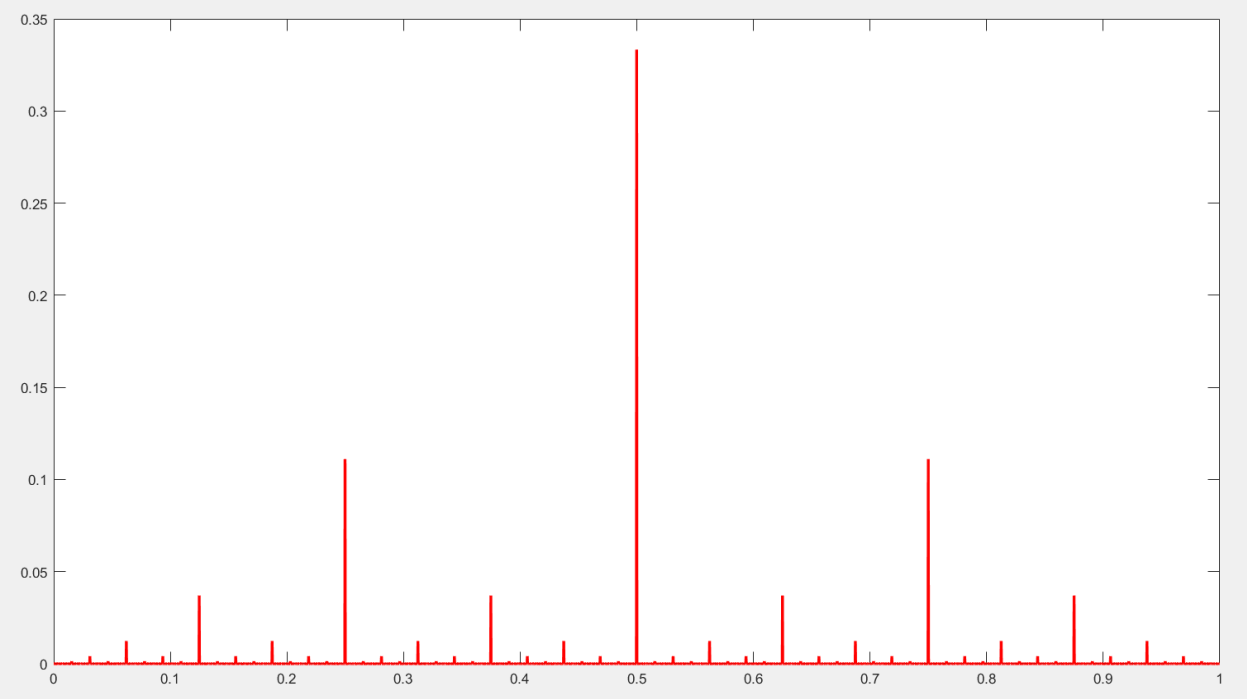



Figure 3: derivative of Devil's Staircase (also a fractal)

151. Unsolved: Initialize the Cantor Set

Find an algorithm for the length of the devil's staircase from recursive parameter k.

## 152. MANDELBROT SET

Compute the complex function and visualize the resulting Mandelbrot set.

Animate the set bring created over iterations.

Skills: complex, meshgrid

## Pseudo Code to Generate Mandelbrot Set:

Repeat these instructions k times.

At each iteration *i*:

Compute the “quadratic map”:

$\mathbf{Z} = \mathbf{Z}^2 + \mathbf{C}$

Find elements with $|\mathbf{Z}|>2$:

Label those elements in matrix **M** as *i*.

MATLAB

Code: MasterMATLAB_2480_Mandelbrot.m

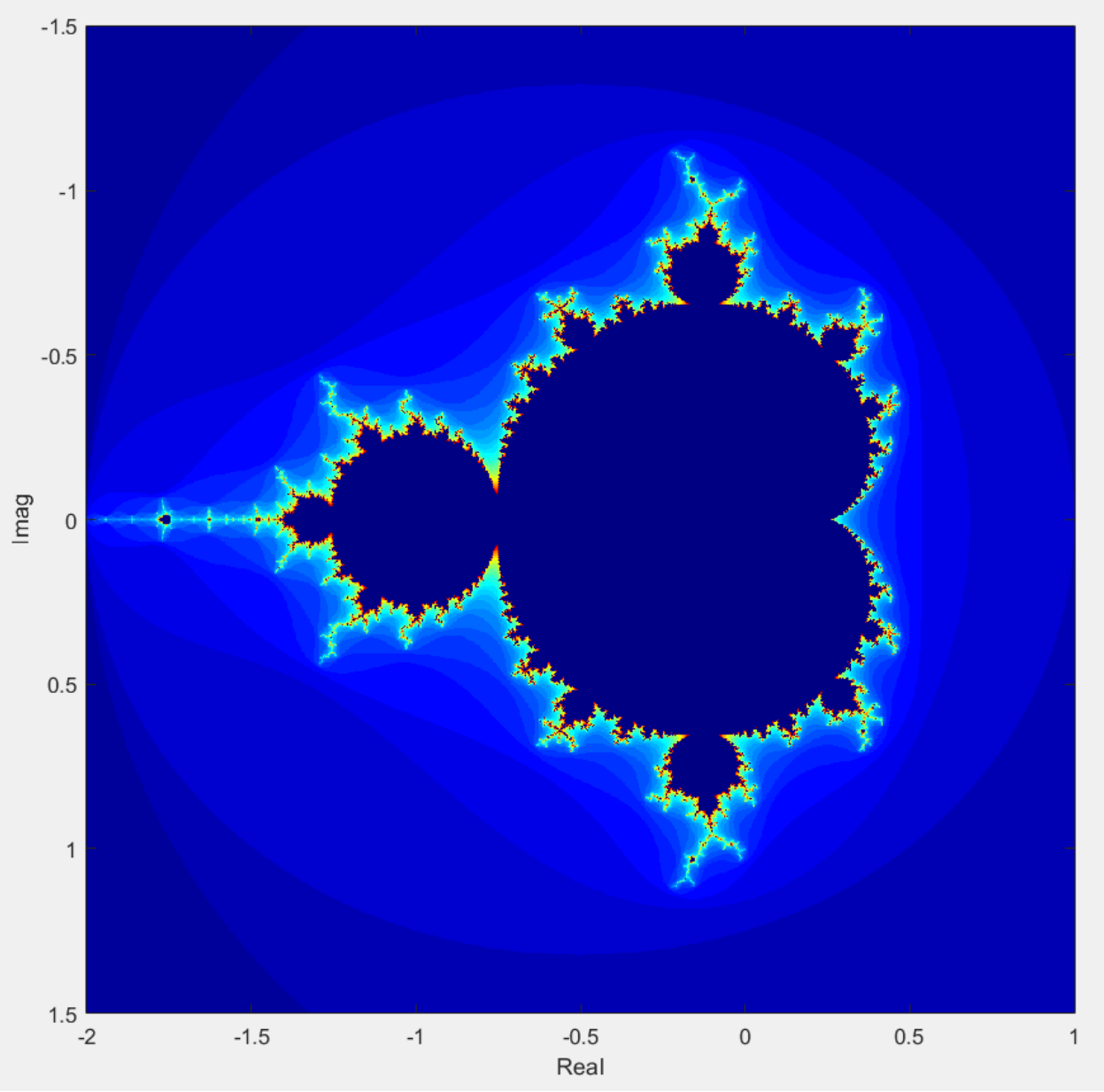



Figure 1: Last step in movie.

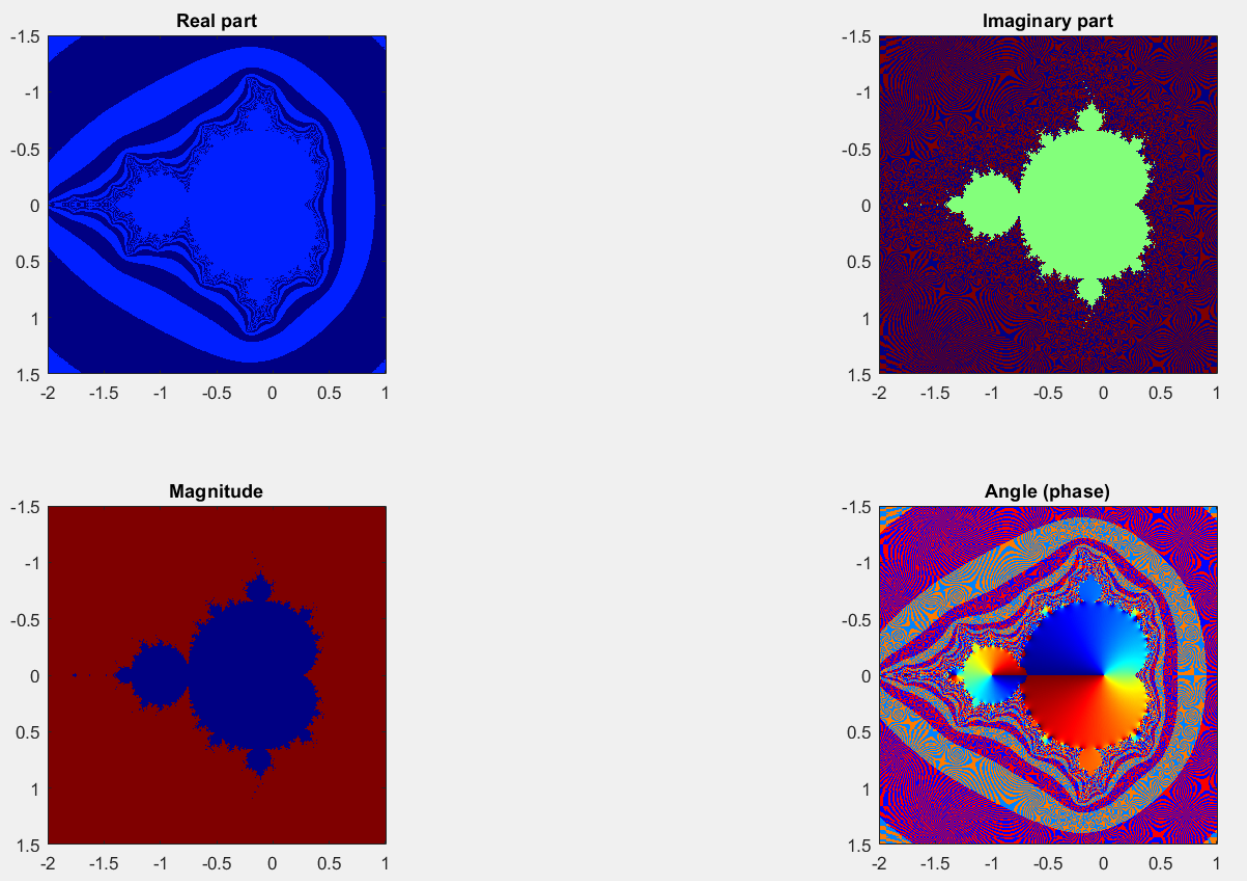



Figure 2: Other views of the Mandelbrot Set.

## 153. WEIERSTRASS FUNCTION

Compute a Weierstrass function and its power spectrum.

Determine the analytical and empirical relationship between the b parameter and the Weierstrass spectral power.

Skills: cos

Weierstrass Functions (one of many, a fractal time series):

$$f = f + 2^{-i} \cos(xb^i)$$

Where depth is defined as:

$$i = 1, 2, 3, \ldots, n$$

A constant:

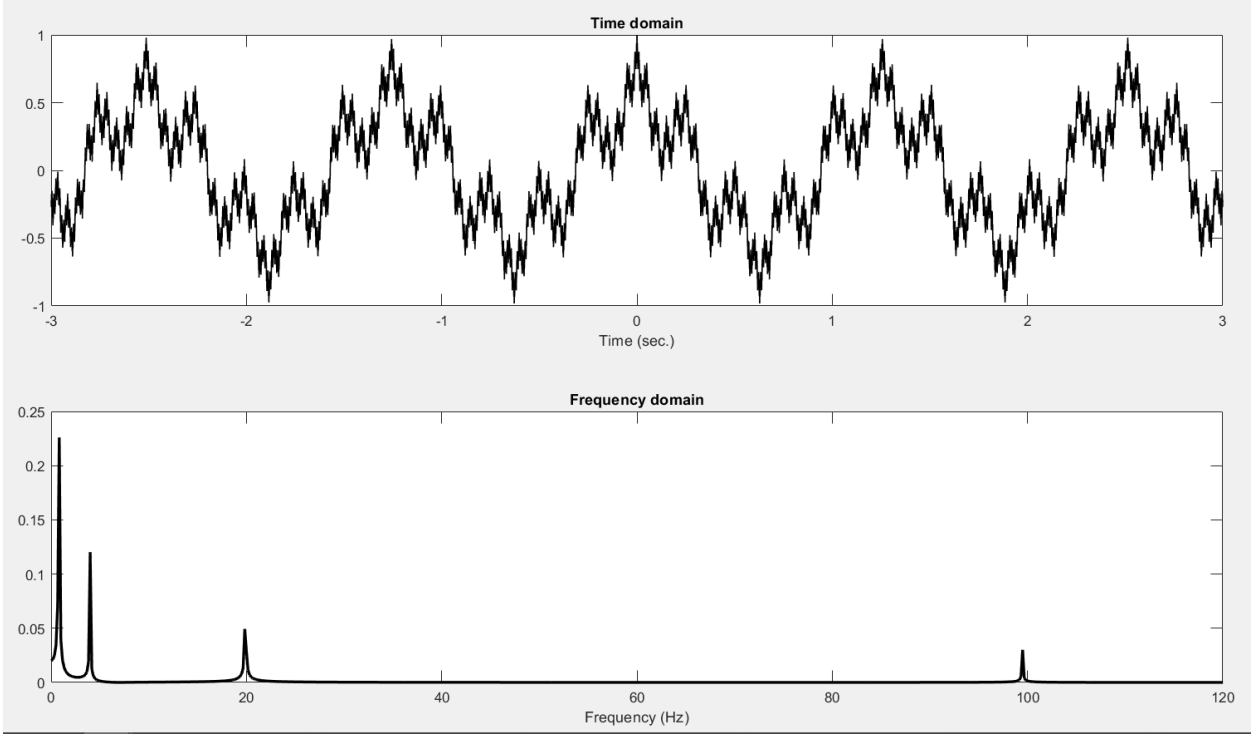
$$b = 5$$

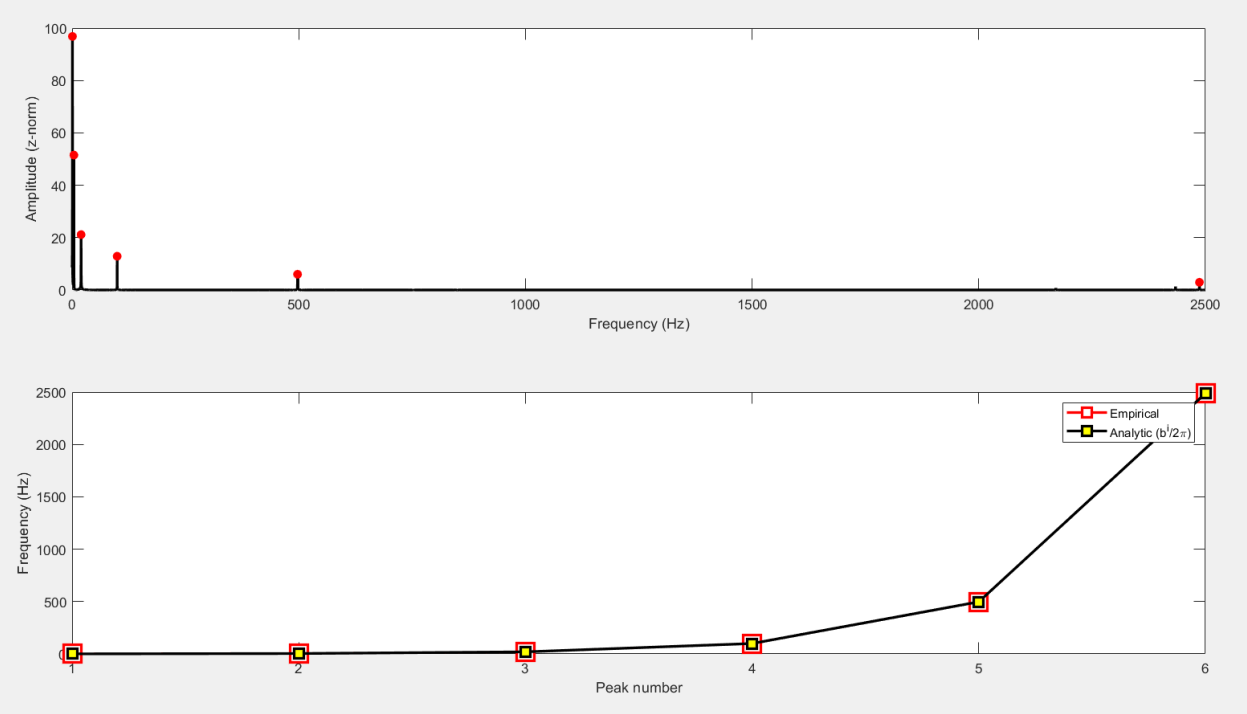
Where time is defined as:

$$x \in (-3, 3)$$

## MATLAB

Code: MasterMATLAB_2500_Weierstrass.m






## 154. FRACTAL CIRCLES AND BUBBLES

Plot lots of circles and spheres with radii drawn from a scale-free distribution.

Make larger bubbles red and smaller bubbles blue.

Skills: cos, sin, sphere, surf, histogram

## Draw circles according to:

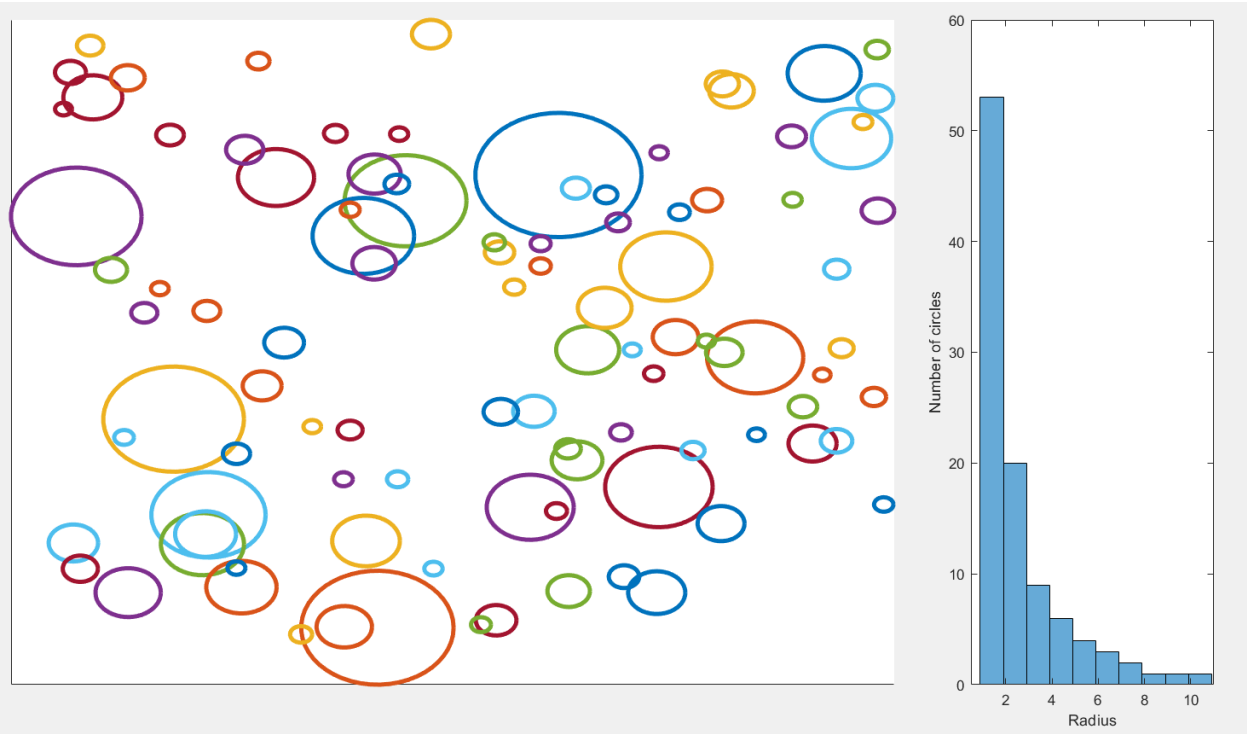
$$x = r\cos(\theta)$$

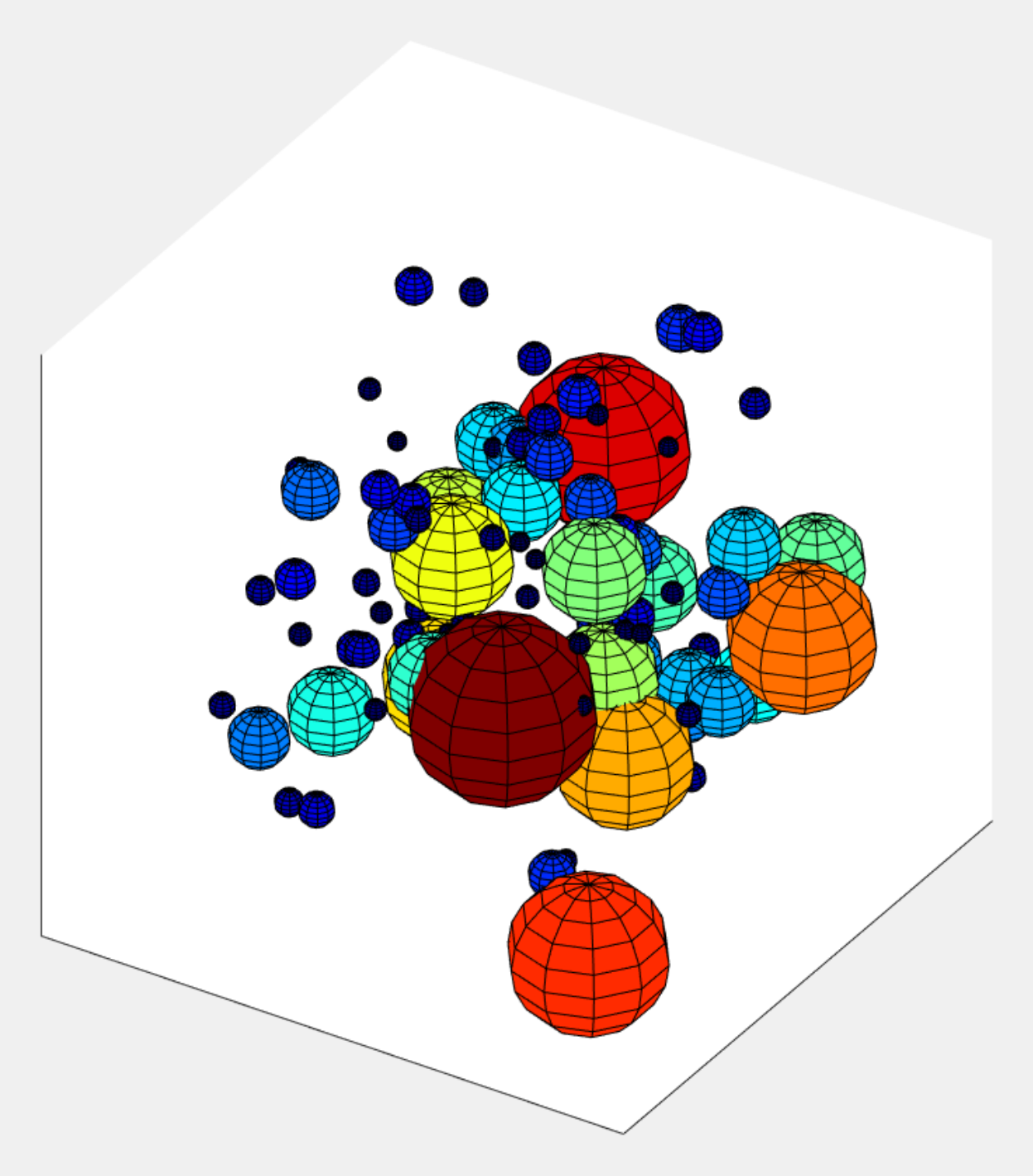
$$y = r\sin(\theta)$$

$r$: radius of circle

## MATLAB

Code: MasterMATLAB_2520_CirclesBubbles.m





SECTION 23: NONPARAMETRIC STATISTICS

LECTURE 155. WILCOXON RANK TEST

Generate two lognormal random distributions and use the Wilcoxon rank-sum test to determine whether they are statistically significantly different.

Use *repmat* to plot the two sets of data in one plot function without using hold on.

Skills: repmat, ranksum, structure

Wilcoxon t-test (aka Wilcoxon rank-sum test) is similar to the regular t-test except it works on rank transformed data, instead of the raw data themselves. Advantage of rank transformed data becomes totally immune to outliers.

## Two Lognormal Random Distributions:

$$f_1 = e^{u_1}\, e^{r_1/2}$$

$$f_2 = e^{u_2}\, e^{r_2/2}$$

## Where:

$$u_1 = 5.3$$

$$u_2 = 5$$

## Noise Characteristics (mean of 0, variance of 1):

$$r_1, r_2 = N(0, 1)$$

---

MATLAB

Code: MasterMATLAB_2540_Wilcoxon

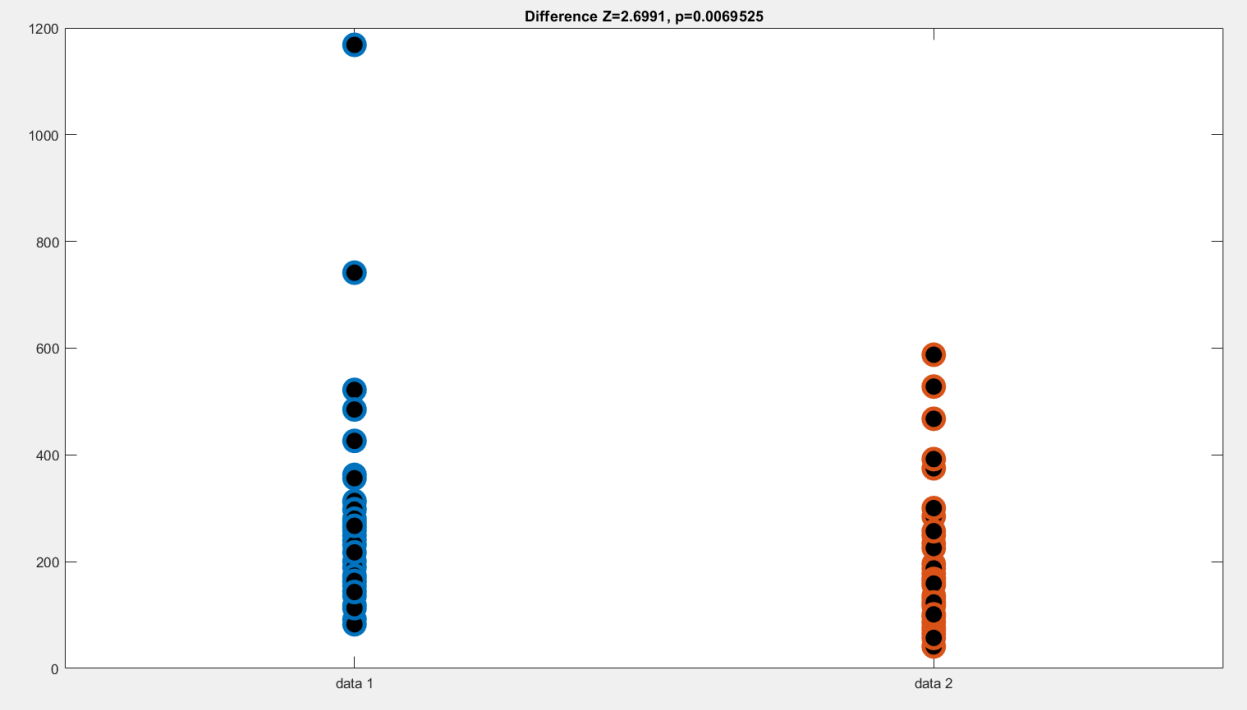



Figure: example of a low p value (1%)

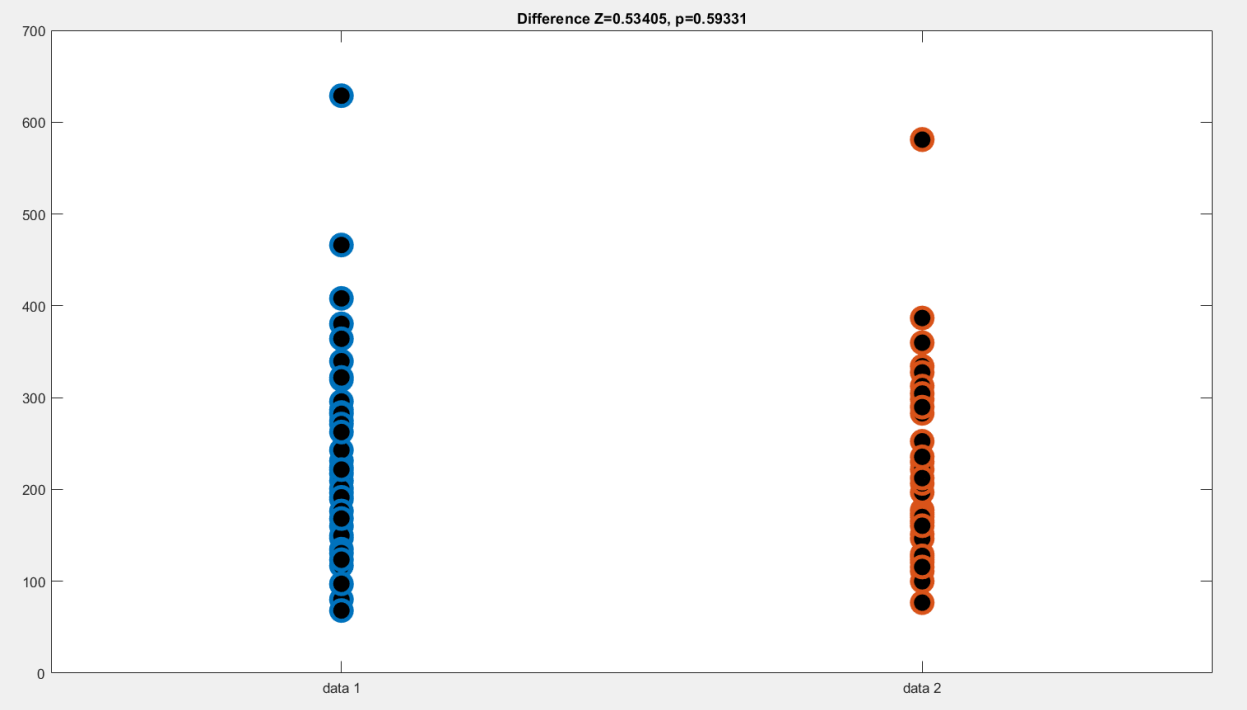



Figure: example of a high p value (59%)

## 156. 2D SPACE OF WILCOXON EFFECT SIZES

Simulate the effects of mean differences and standard deviation on the effect size. Average the results over 100 experiments.

Interpret the results.

Skills: ranksum, contour, colorbar

## Two Lognormal Random Distributions:

$$f_1 = e^{u_1}\, e^{r_1/2}$$

$$f_2 = e^{u_2}\, e^{r_2/2}$$

## Where:

$$u_1 = \in (5, 6)$$

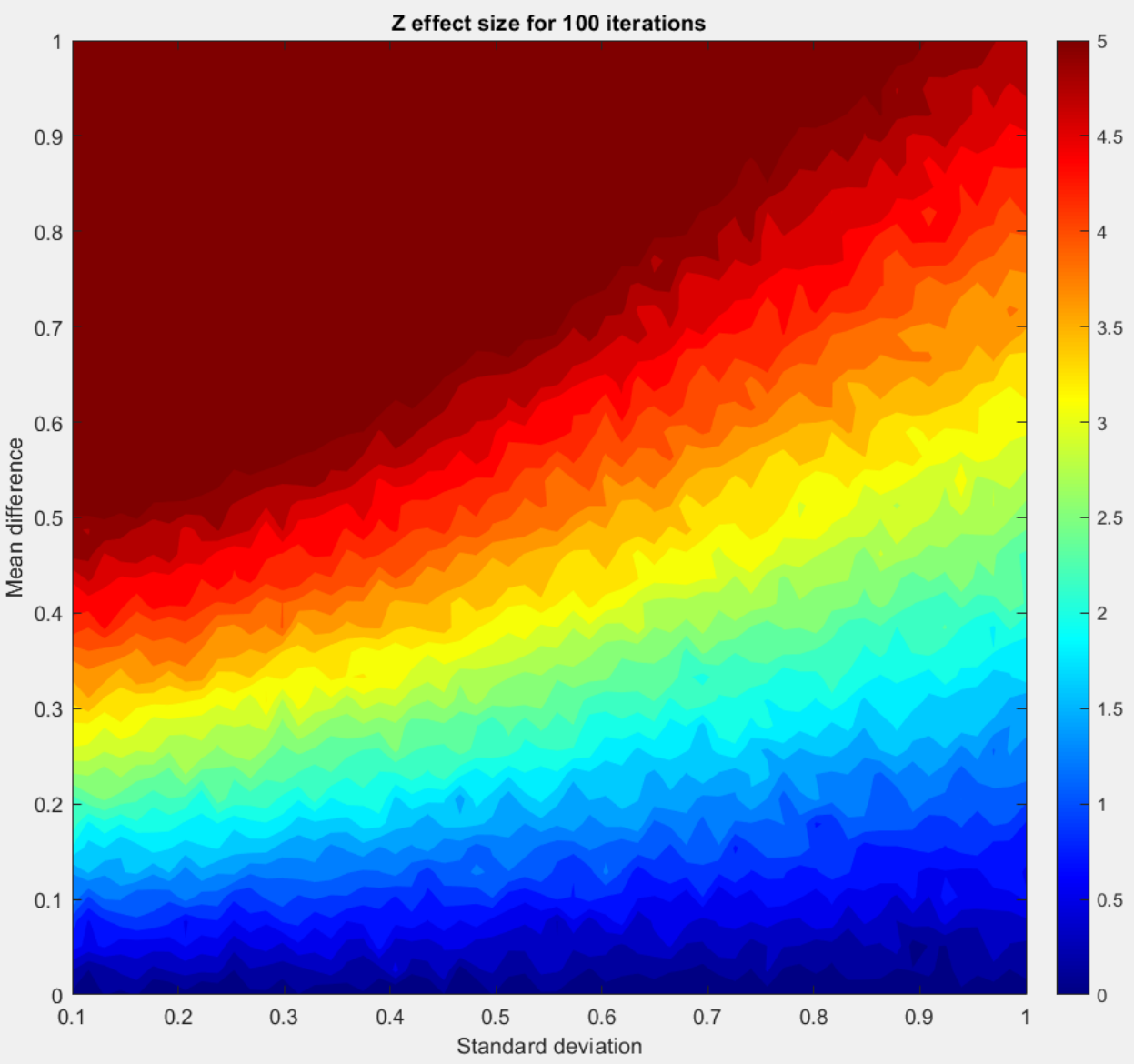
$$r_1 = N(0, \in (.1, 1))$$

$$u_2 = 5$$

$$r_2 = N(0, 1)$$

### MATLAB

Code: MasterMATLAB_2560_Wilcoxon2Dspace.m




## 157. KL DIVERGENCE OF TWO DISTRIBUTIONS

Compute the Kullback-Leibler divergence between lognormal and normal distributions.

Skills: linspace, histcounts, isfinite, log

# KL Divergence:

$$D_{KL}(P \parallel Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$

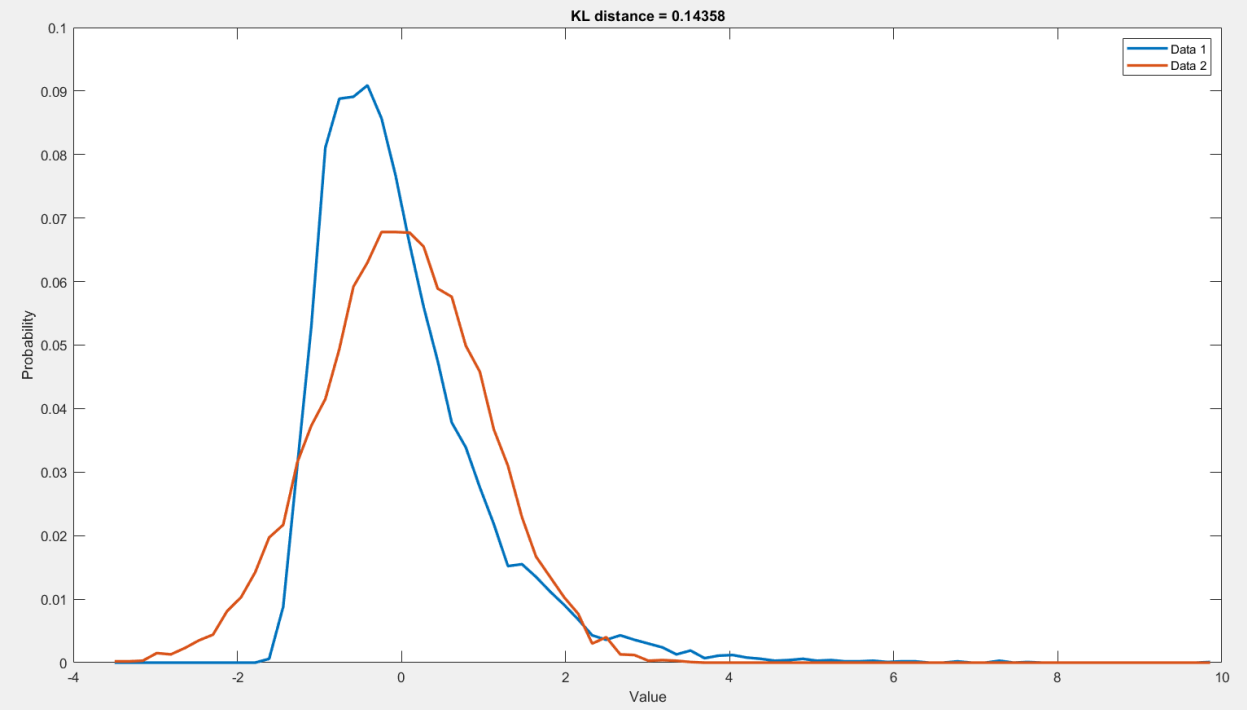
Where:

$P, Q$: Discrete probability distributions

Ref: https://en.wikipedia.org/wiki/Kullback%E2%80%93Leibler_divergence#Definition

### MATLAB

Code: MasterMATLAB_2580_KLdivergence.m




## 158. 2D SPACE OF KL DIVERGENCES

Manipulate standard deviation and mean difference between two distributions to generate a 2D parameter map of KL distances.

Average the results over 20 trials (repetitions).

Explain why “jet’ is not a good colormap for this project.

Skills: histcounts, colorbar, colormap

$$D_{KL}(P \parallel Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$

$d_1 = N(0,1)$

$d_2 = N(\mu, \sigma)$

$\mu \in (0, 4)$

$\sigma \in (.1, 3)$

## MATLAB

Code: MasterMATLAB_2600_KL2D.m

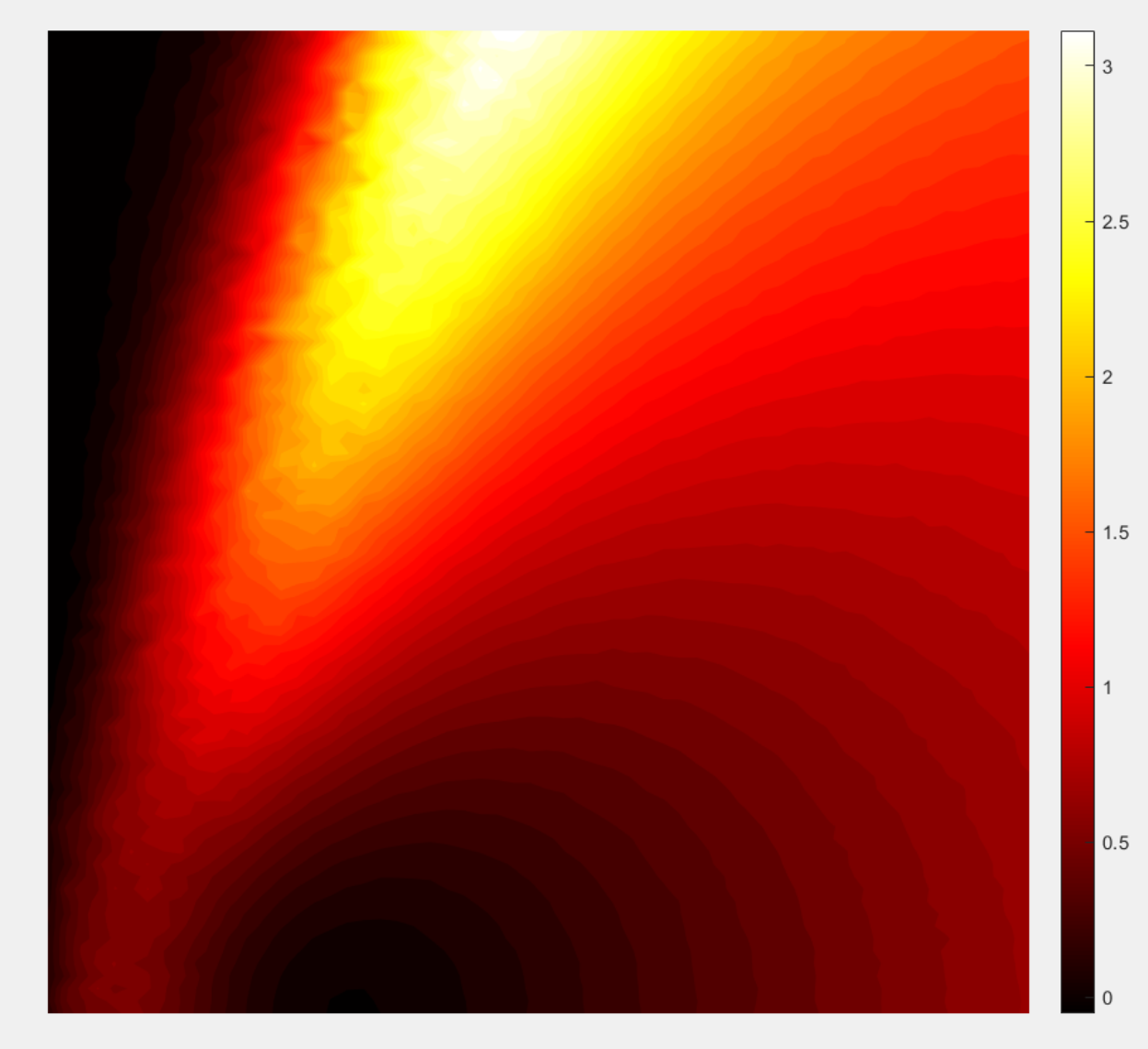



Figure 1: contourf

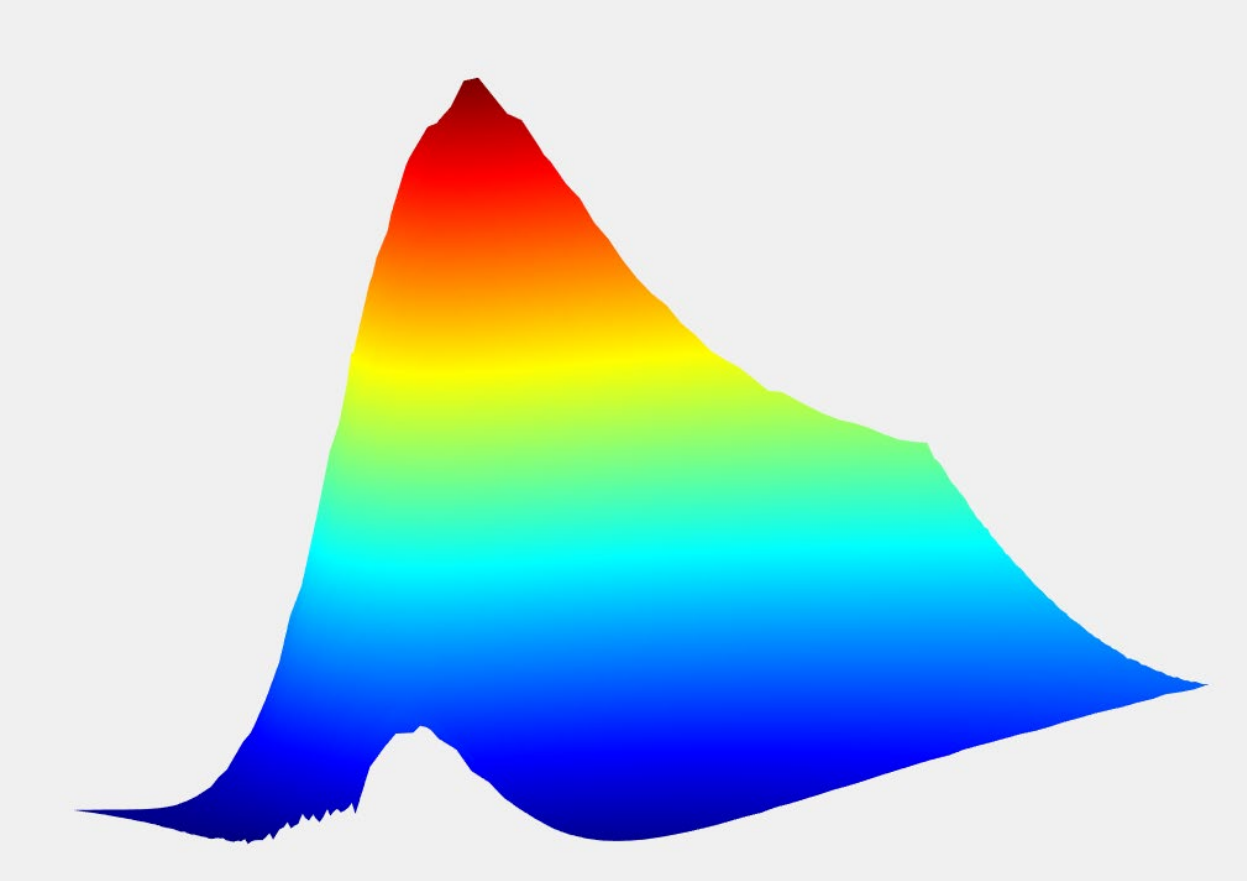


Figure 2: 3D surf

## 159. PERMUTATION TESTING (FOR EMPIRICAL Z VALUE)

Generate two lognormal distributions with different means, and use permutation testing to determine whether they are statistically significantly different.

Use subplots to arrange the figure like in the next slide.

Implement a “meta-permutation” test to reduce variability.

Skills: histogram, sqrt, var, ones, subplot

Test to determine if one distribution is greater than another using the difference in the mean between two distributions. The t test is a statistical procedure to test the differences in means.

t Test:

$$t = \frac{m_a - m_b}{\sqrt{\frac{\sigma_a^2}{N_a} + \frac{\sigma_b^2}{N_b}}}$$

$m_a, m_b$: are the means of dataset a and b;

$Numerator$: Sum of the variances for the two distributions;

$\sigma_a, \sigma_b$: Standard deviation for the two distributions,

$N_a, N_b$: Number of data points in datasets a and b.

**Goal of permutation testing:** The data is non-normally distributed so p-values can not be used to determine if t test result is significant. So a nonparametric approach will be used to determine if the t statistic result is significant.

Swap data points between the two distributions randomly.

If the distributions are different swapping data values will lower the t test.

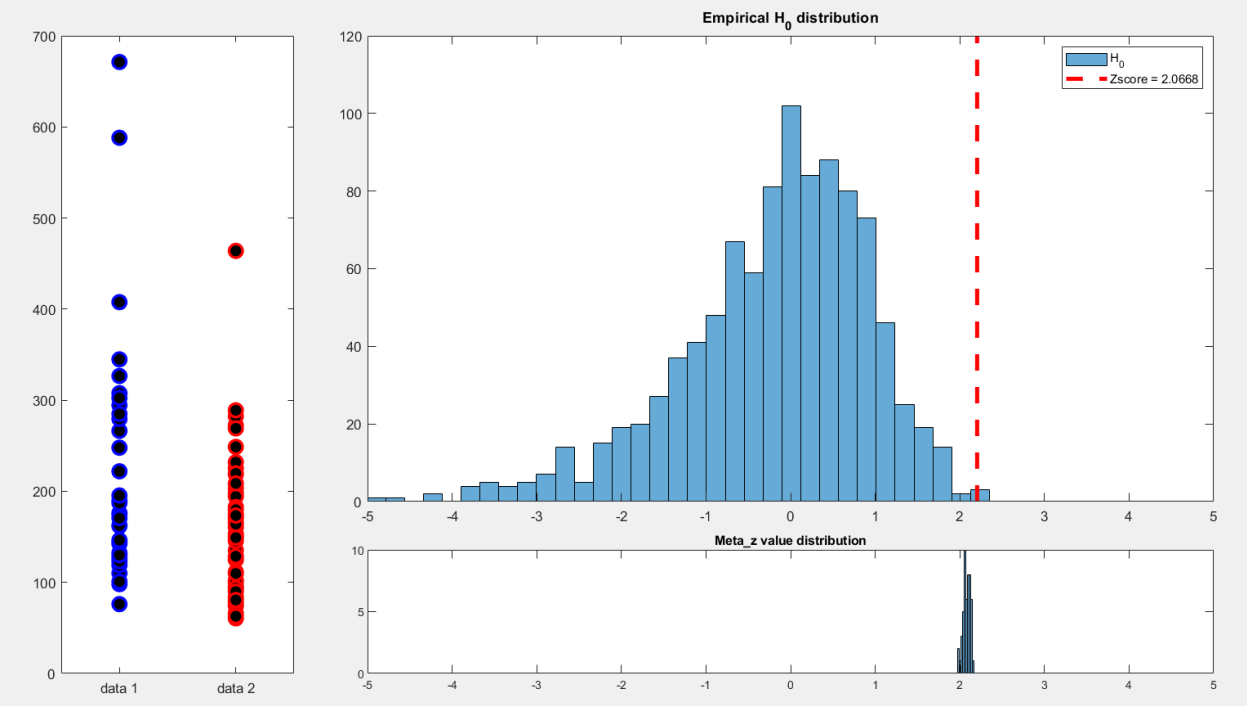
If the distributions are the same swapping the data values will not change the value of the t test,

Repeat by swapping 1000's of times

---

MATLAB

Code: MasterMATLAB_2620_permutationTesting.m




## 160. BOOTSTRAPPING FOR CONFIDENCE INTERVALS

Use “bootstrapping” to obtain 95% confidence intervals for the mean of a distribution.

Implement the project without using the statistics toolbox functions: *randsample* and *prctile*.

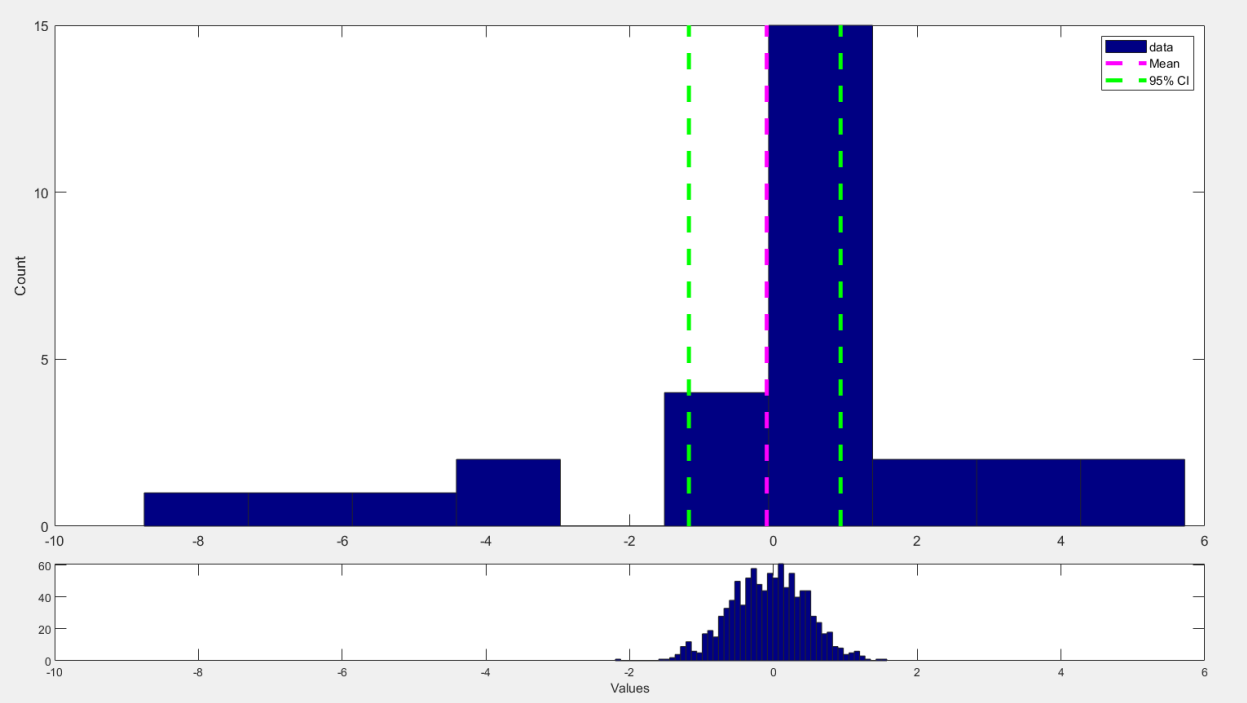
Skills: randsample, prctile, mean, hist

Bootstrapping:

- Select N random elements, with replacement
- Compute the mean
- Repeat k Times

### MATLAB

Code: MasterMATLAB_2640_bootstrapping.m




## 161: UNSOLVED: BOOTSTRAPPING MEDIANS

# SECTION 24: NONLINEAR MODEL FITTING

## 162. FIND A FUNCTION MINIMUM

Implement a function as an anonymous function and find its bounded and unbounded minimum.

Figure out what is “wrong” with *fminsearch*!

Skills: fminsearch, fminbnd, func2str

### Function #1 for this project:

$$f(x) = \sin(x^2) + x\cos^2(x)$$

$$x \in [-\pi, 5\pi/2]$$

Function #2 for this project:

$$f(x) = \int_{x_0}^{x} \sin(x^2)dx$$

MATLAB

Code: MasterMATLAB_2660_anonMin.m

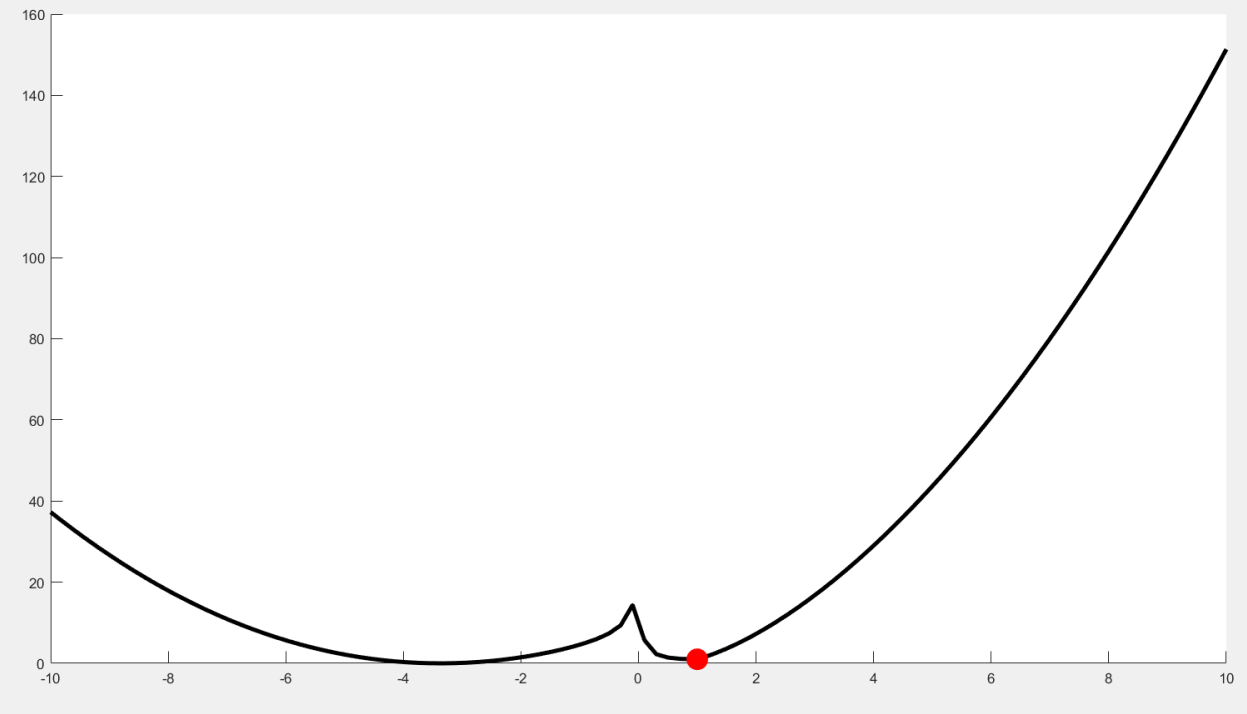



Figure 1: demonstrate anonymous functions and

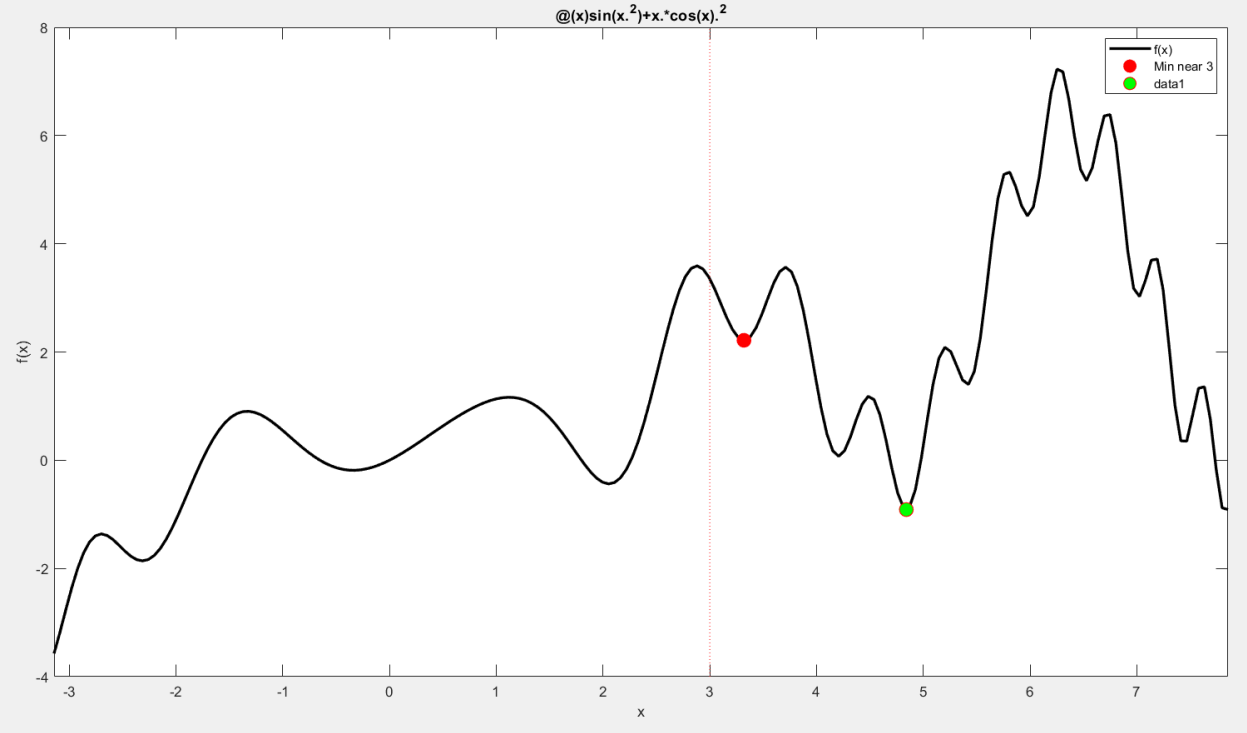



Figure 2: Demonstrate use of *fminsearch* (red dot) and *fminbnd* (green dot).

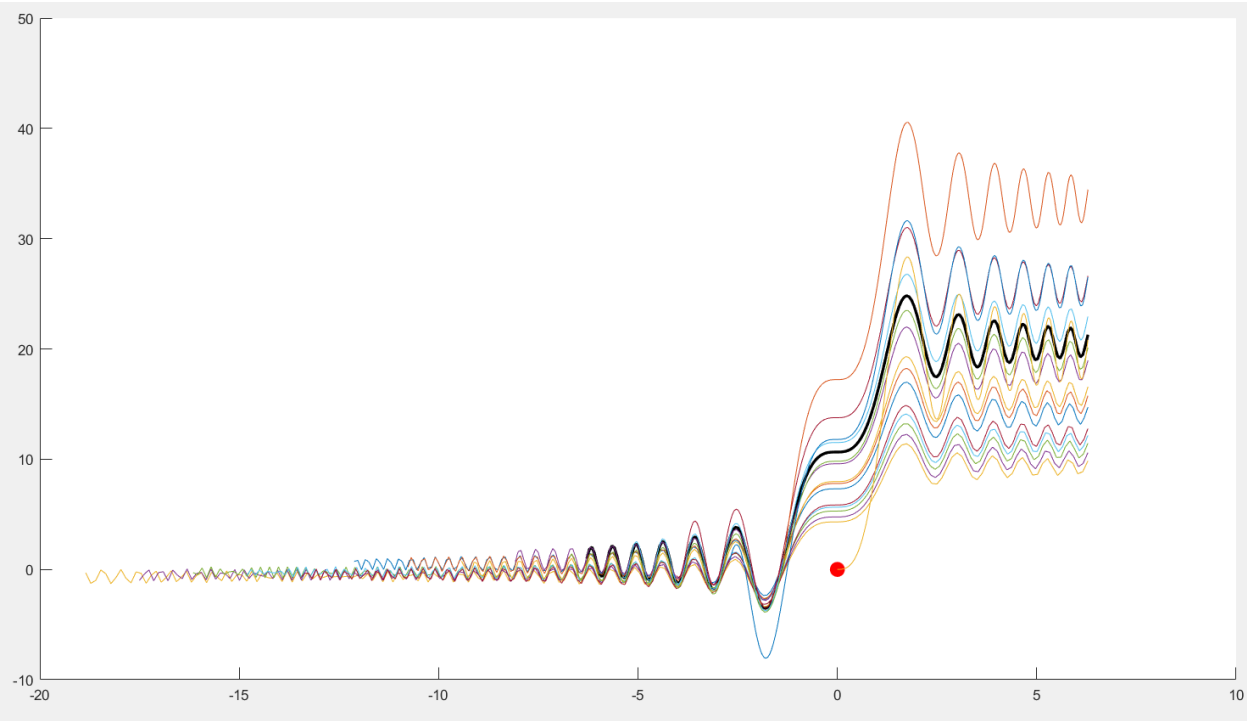


Figure 3: Demonstrate strange behavior of *fminsearch* for an integral function (see function #2 equation)

## 163. FIT A GAUSSIAN TO A DISTRIBUTION

Write a separate function that can be called to fit a Gaussian to a set of points.

Test a variety of starting parameters to determine their importance on the final result.

Skills: fminsearch, exp

## MATLAB

Code: MasterMATLAB_2680_fitGaussian.m

Function: fitGaussian.m

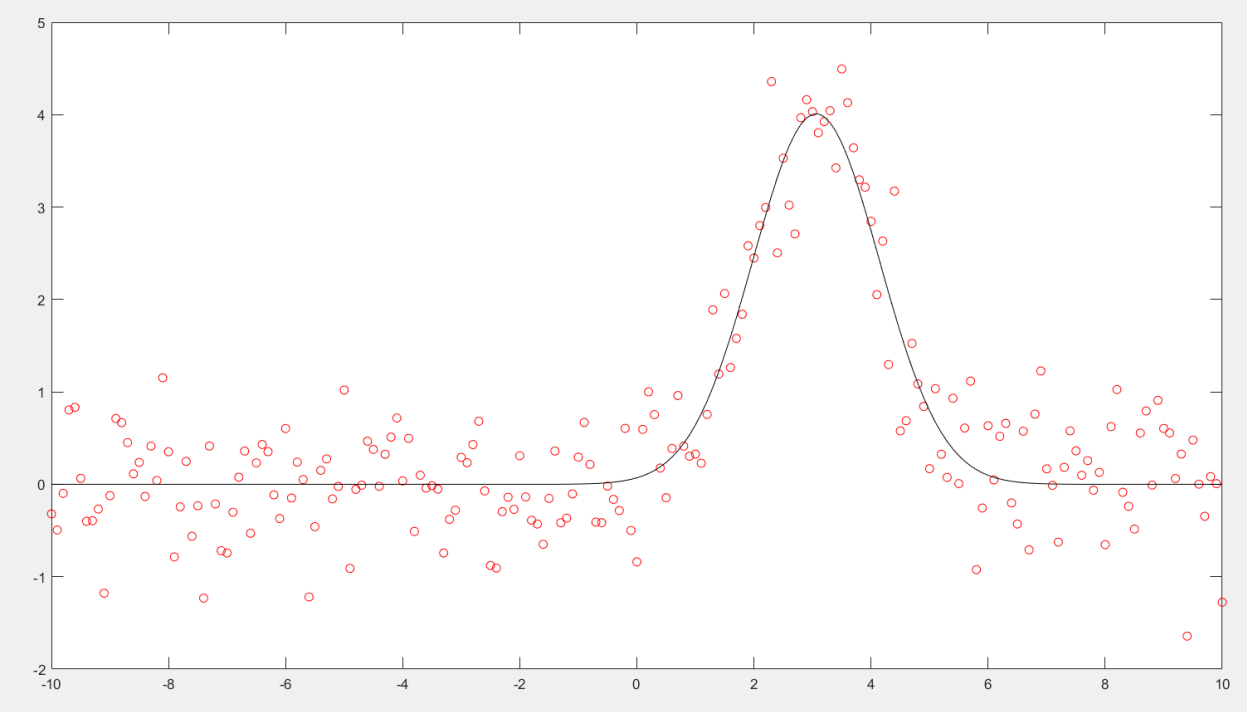



Figure 1: demonstrate how to use fminsearch with a search function to best-fit data

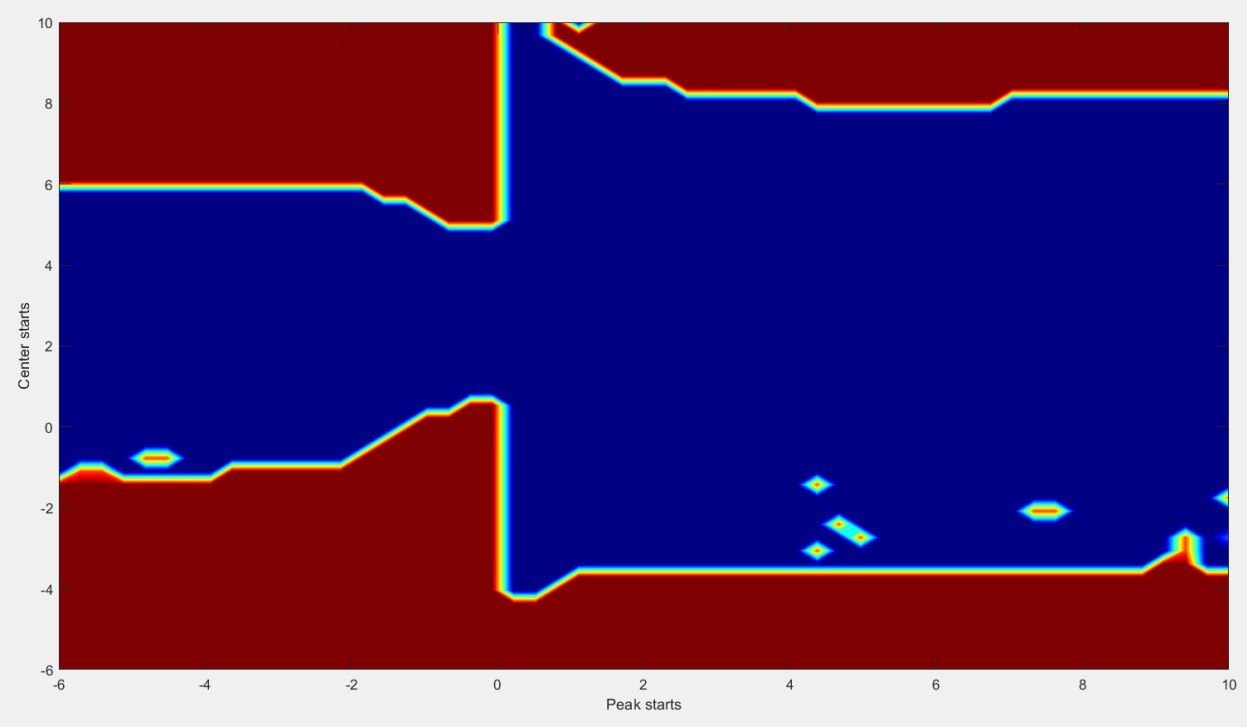



Figure 2: Find the range of initial enter and peak values within parameter list to solve the best-fit

## 164. TWO-PIECE LINEAR REGRESSION

Generate random data according to a triangle distribution.

Fit a piecewise linear function to noisy data.

Write an algorithm to find a good initial parameter.

Skills: fminsearch, sqrt

## Generating triangular-distributed random variates:

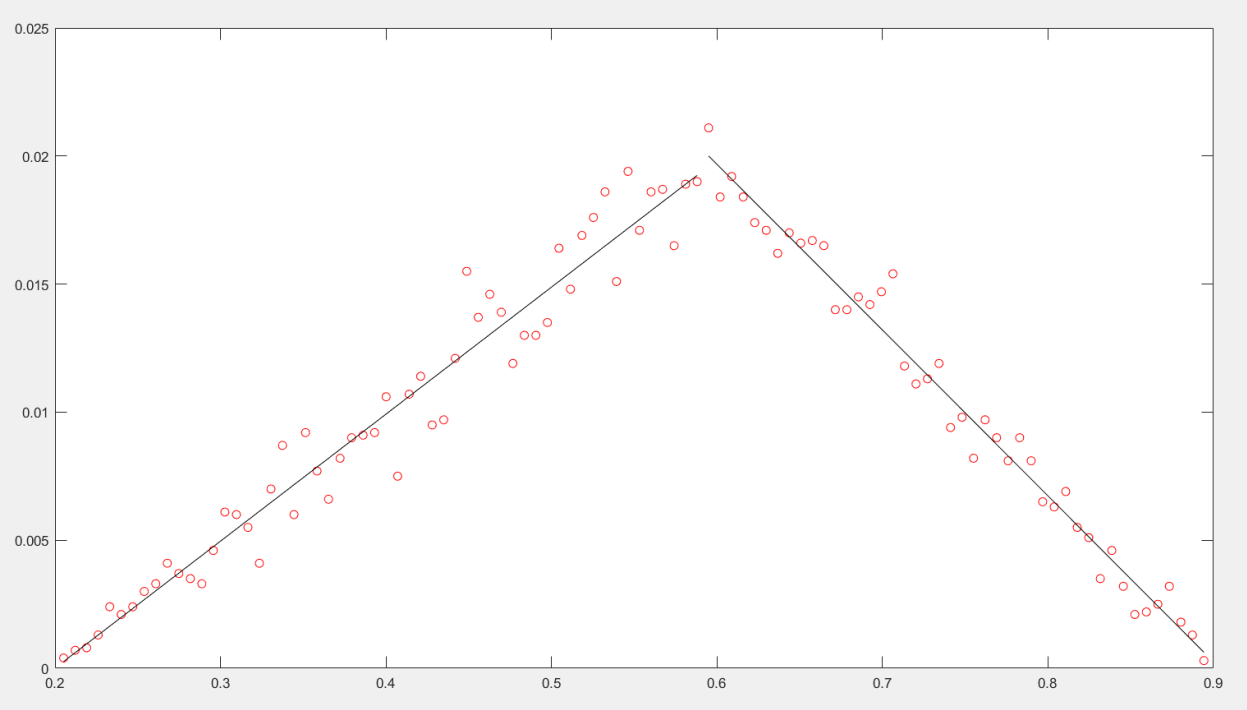
$$\begin{cases} X = a + \sqrt{U(b-a)(c-a)} & for\ 0 < U < F(c) \\ X = b - \sqrt{(1-U)(b-a)(b-c)} & for\ F(c) \leq U\ < 1 \end{cases}$$

$$F(c) = (c-a)(b-a)$$

### MATLAB

Code: MasterMATLAB_2700_fit2piece.m

Function: fit2segLinear.m




## 165. FIT A SINE WAVE

Use nonlinear search methods to fit a 3-parameter sine model to a sine wave.

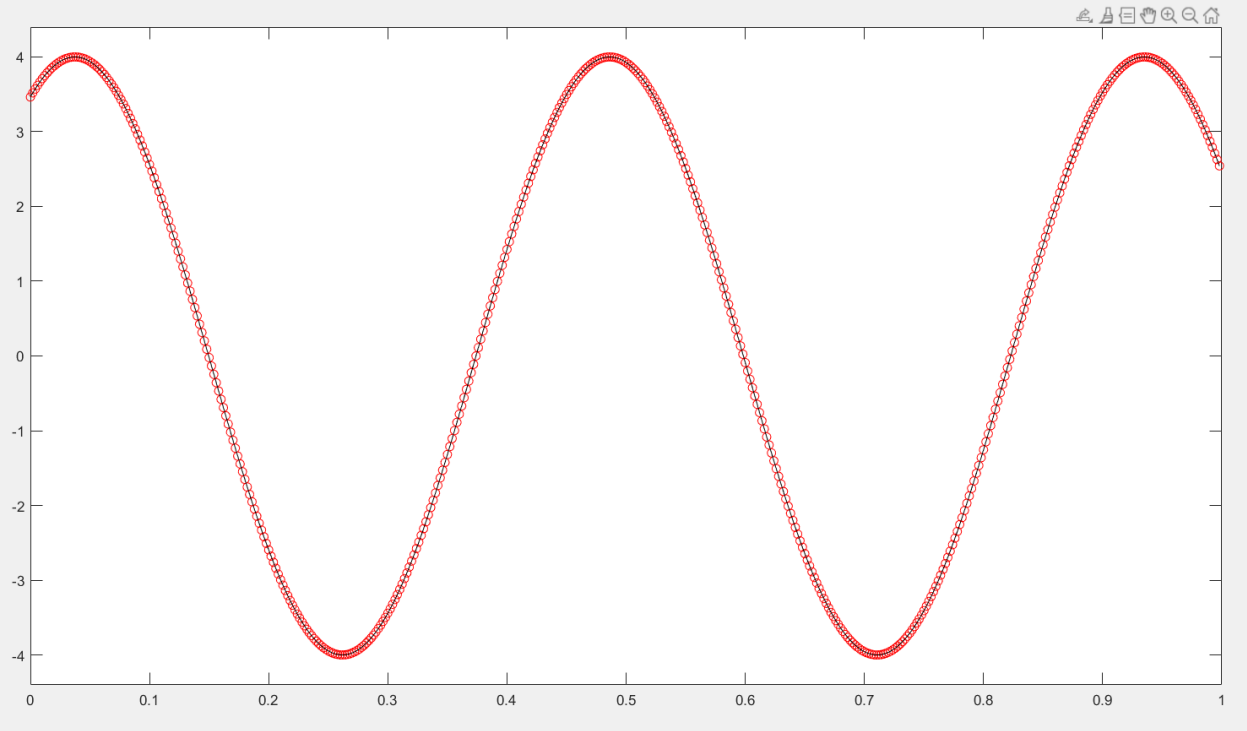
Time how long the minimization takes with and without the plotting.

Skills: fminsearch

### MATLAB

Code: MasterMATLAB_2720_fitSine.m

Function: fitSine.m




## 166. FIT A CIRCLE TO A NOISY RING

Generate a noisy ring and find the best-fit circle.

Compare computation times and fits for *fminsearch* and lsqnonlin

Make it an oval and see if it still works!

Skills: fminsearch, lsqnonlin, repmat

### MATLAB

Code: MasterMATLAB_2740_noisyRing.mron

Function: fitCirc.m

Function: fitOval.m

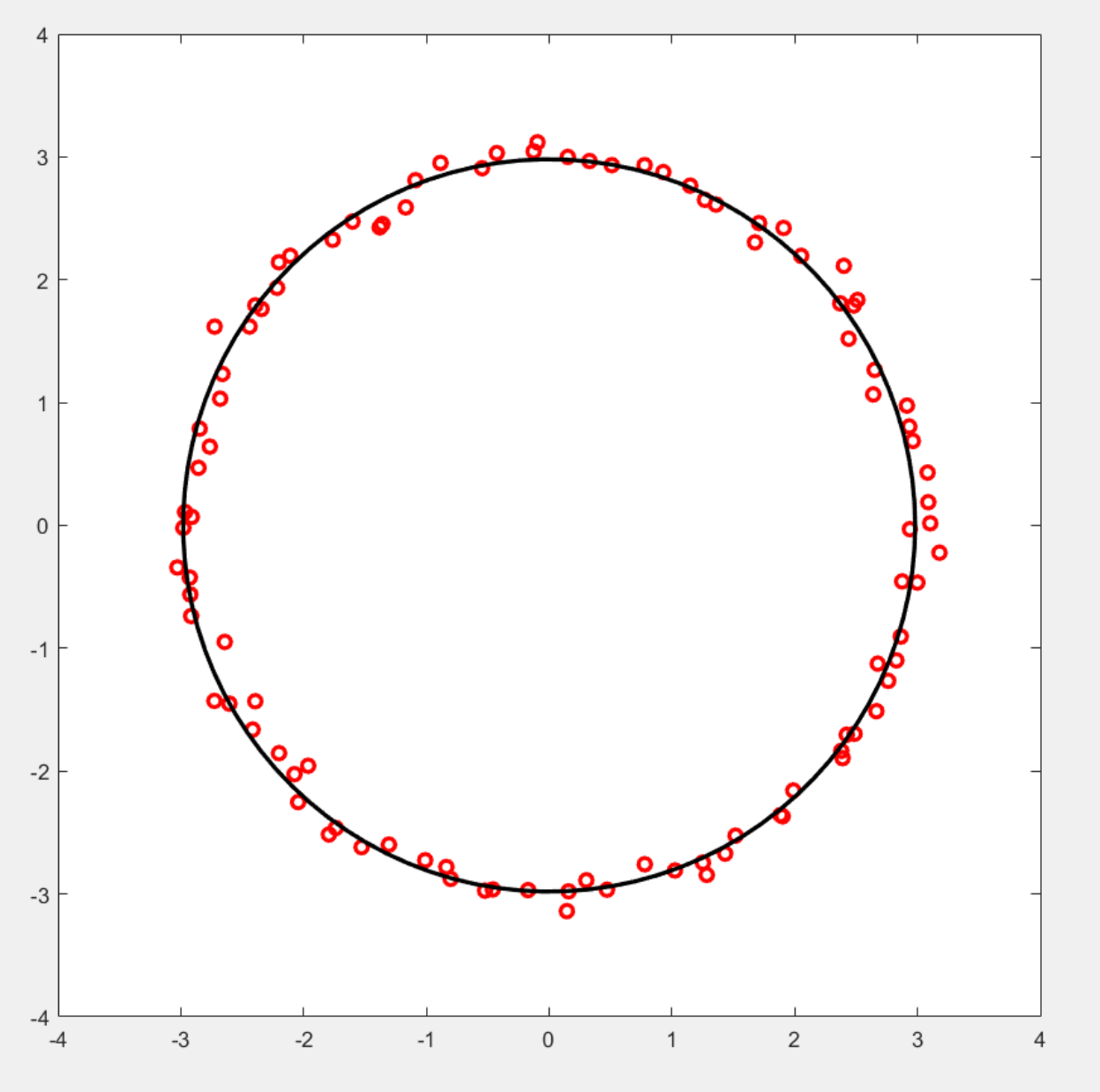



Figure 1: Use fminsearch to fit a noisy circle

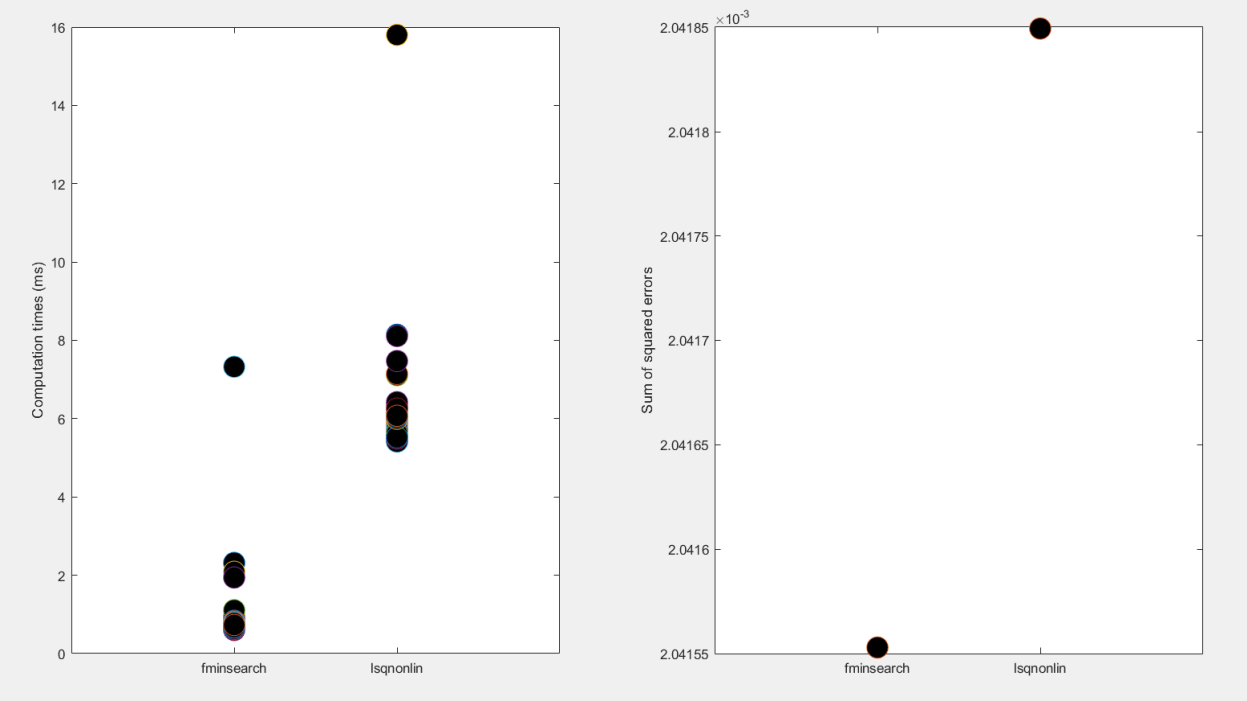



Figure 2: Performance comparison between fminsearch and lsqnonlin

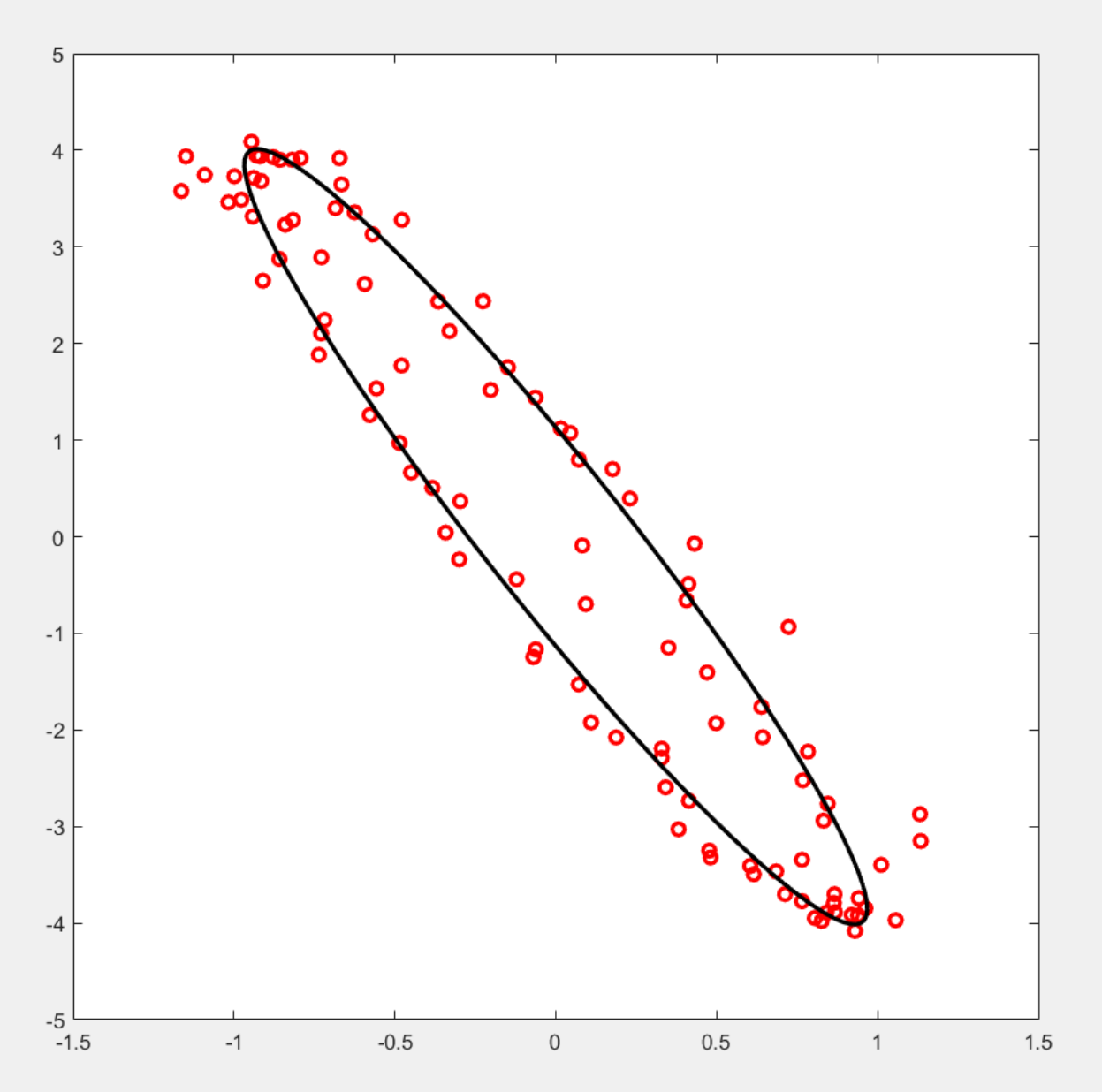



Figure 3: Use fminsearch to fit a noisy oval

## MISC. NOTES

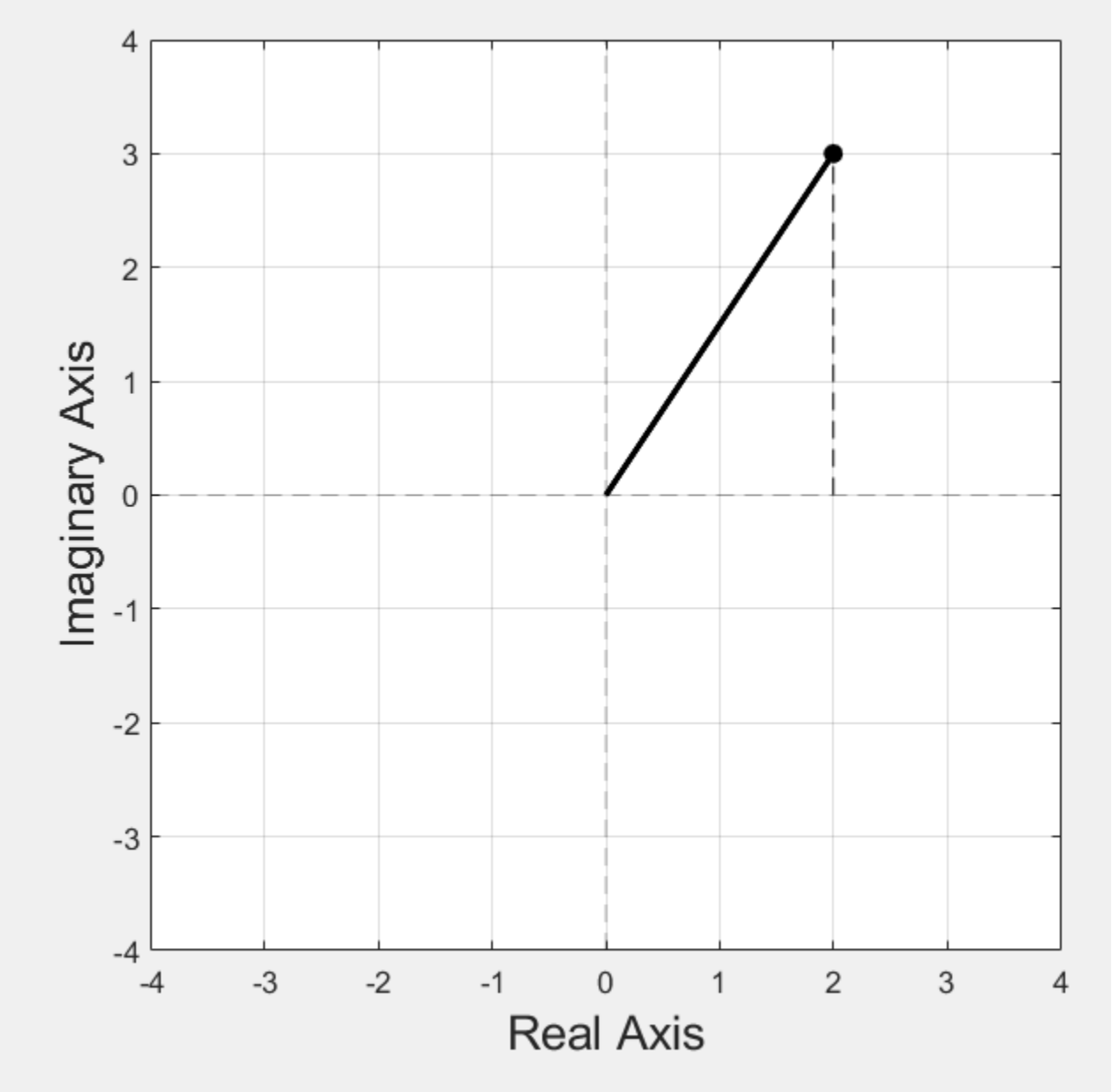
MATLAB default file location: D:\Ron\Documents\MATLAB

## FIND PARAMETERS OF A SINE WAVE

```
% create a sine wave using normalized time points
t = (0:n-1)/n;
x = a*sin( t*f + p);

% test code: estimate fitSine parameters
amp = (max(x)-min(x))/2; % estimate amplitude
[~,locs] = findpeaks(x,t);
fre = 2*pi*1/max(diff(locs)); % estimate frequency
ph = asin(x(1)/a); % estimate phase (in radians)
```

## CREATE A 4-QUADRENT GRAPH

```
%% Create a 4-quadrant graph with line from origin

gridSize = [-4 4];
gridAlpha = .3;
figure(1), clf

% grid
h1 = plot(zeros(2,1),gridSize,'k--'); hold on
h1.Color(4) = gridAlpha; % set alpha value
h2 = plot(gridSize,zeros(2,1),'k--');
h2.Color(4) = gridAlpha; % set alpha value
% line
plot([0 2],[0 3],'k','linew',2)
% dot at end of line
plot([2 2],[3 3],'ko','MarkerFaceColor','k')
% dashed line from x-axis to dot
plot([2 2],[0 3],'k--')
% details
xlabel('Real Axis', 'FontSize', 16)
ylabel('Imaginary Axis', 'FontSize', 16)
grid on
axis square
```




## CLC

clc

Clear command window

## CLEAR

clear

Clear the workspace

## WHOS

```
>> whos convres2 signal
  Name           Size          Bytes  Class     Attributes

  convres2     10200x1          81600  double
  signal       10000x1          80000  double
```

## WHICH

which

Determine if a variable is also a predefined function, and where the code is located

```
>> which timer
C:\Program Files\MATLAB\R2020a\toolbox\matlab\iofun\@timer\timer.m  % timer constructor
```

## PLOTTING

### BAR

bar

plot a bar chart

### CLA

cla

Clear plotting axis (the plots are cleared, not the axis titles)

### CLF

clf

clear the figure associated with a figure(x)

### DATACURSORMODE

datacursormode

Active data cursor within plot using mouse

### ERRORBAR

errorbar

plot with error bars

### GET

get(gca,'ylim') get y-axis limits using get current access option

### SET

### GCA

set(gca,’xlim’,[0 15])

set x axis limit to 0 and 15 units

## HANDLE

h = plot(time,signal);

set(h,'color',[1 1 1]*.7)

set the color of the plot line using a handle, h, in this case