Exercise: $e \sim 2.7183$

- Write a program to estimate the Euler constant by Monte Carlo simulation.
- It can be done as follows.
- Let N be the number of iterations.
- For each iteration, find the minimal number n so that $\sum_{i=1}^{n} r_i > 1$ where r_i is the random variable following the standard uniform distribution (you can simply use **rand**).
- Then e is the average of n.

Special Issue: Sort

```
>> stocks = {"GOOG", 15;
             "TSMC", 12;
             "AAPL", 18};
>> [~, idx] = sort([stocks{:, 2}], "descend")

idx =

     3     1     2

>> stocks = stocks(idx, :)

stocks =

    "AAPL"    [18]
    "GOOG"    [15]
    "TSMC"    [12]
```

Programming Exercise: Sorting Algorithm[1]

- Let A be any array.
- Write a program which outputs the sorted array of A (in ascending order).
- For example, $A = [5, 4, 1, 2, 3]$.
- Then the sorted array is $[1, 2, 3, 4, 5]$.

[1] See `https://visualgo.net/sorting`.

Special Issue: Random Permutation

- Use **randperm** to generate an index array with a random order.

```
>> A = ["Matlab", "Python", "Java", "C++"];
>> idx = randperm(length(A))

idx =

     3     1     2     4

>> A(idx)

ans =

  1x4 string array

    "Java"    "Matlab"    "Python"    "C++"
```

"Exploring the unknown requires tolerating uncertainty."

– Brian Greene

"I can live with doubt, and uncertainty, and not knowing. I think it is much more interesting to live not knowing than have answers which might be wrong."

– Richard Feynman

Speedup: Vectorization (Revisited)[2]

- Vector in, vector out.

```
>> x = randi(100, 1, 5)

x =

    88    30    90    73    82

>> dx = diff(x)

dx =

   -58    60   -17     9
```

[2] More about vectorization.

Advantages from Vectorization

- Appearance: vectorized mathematical code appears more like the mathematical expressions found in textbooks, making the code easier to understand.
- Less error prone: without loops, vectorized code is often shorter.
    - Fewer lines of code mean fewer opportunities to introduce programming errors.
- Performance: vectorized code often runs much faster than the corresponding code containing loops.

Performance Analysis: Profiling

- Use a timer to measure your performance.[3]
  - In newer version, press the button *Run and Time*.
- Identify which functions are consuming the most time.
- Know why you are calling them and then look for alternatives to improve the overall performance.

[3] Note that the results may differ depending on the difference of run-time environments, so make sure that you benchmark the algorithms on the same conditions.

tic & toc

- The command **tic** makes a stopwatch timer start.
- The command **toc** returns the elapsed time from the stopwatch timer started by **tic**.

```
>> tic
>> toc
Elapsed time is 0.786635 seconds.
>> toc
Elapsed time is 1.609685 seconds.
>> toc
Elapsed time is 2.417677 seconds.
```

Selected Performance Suggestions[4]

- Preallocate arrays.
  - Instead of continuously resizing arrays, consider preallocating the maximum amount of space required for an array.
- Vectorize your code.
- Create new variables if data type changes.
- Use functions instead of scripts.
- Avoid overloading Matlab built-in functions.

[4] See Techniques for Improving Performance.

Programming Exercise: A Benchmark

- Let $N = 1e1, 1e2, 1e3, 1e4, 1e5$.
- Write a program which produces a benchmark for the following three cases:
  - Generate an array of $1 : N$ by dynamically resizing the array.
  - Generate an array of $1 : N$ by allocating an array of size N and filling up sequentially.
  - Generate an array of $1 : N$ by vectorization.

Analysis of Algorithms (Optional)

- For one problem, there exist various algorithms (solutions).
- We then compare these algorithms for various considerations and choose the most appropriate one.
- In general, we want efficient algorithms.
- Except for real-time performance analysis, could we predict before the program is completed?
- Definitely yes.

Growth Rate

- Now we use $f(n)$ to denote the growth rate of time cost as a function of n.
  - In general, n refers to the data size.
- For simplicity, assume that every instruction (e.g. $+ - \times \div$) takes 1 unit of computation time.
- Find $f(n)$ for the following problem.
  - Sum(n): ?
  - Triangle(n): ?

O-notation[5]

- In math, O-notation describes the limiting behavior of a function, usually in terms of simple functions.
- We say that

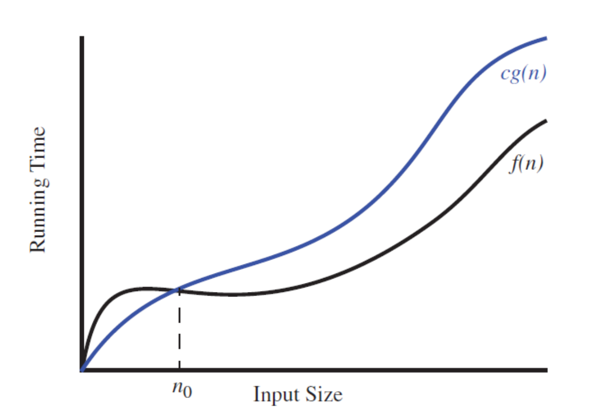
$$f(n) \in O(g(n)) \text{ as } n \to \infty$$

if and only if $\exists c > 0, n_0 > 0$ such that

$$|f(n)| \leq c|g(n)| \quad \forall n \geq n_0.$$

- So $O(g(n))$ is a collection featured by a simple function $g(n)$.
- We use $f(n) \in O(g(n))$ to denote that $f(n)$ is one instance of $O(g(n))$.

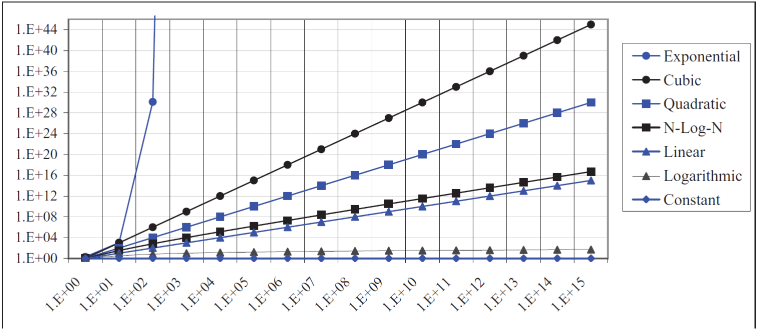
[5] See https://en.wikipedia.org/wiki/Big_O_notation.



- Big-O is used for the asymptotic upper bound of time complexity of algorithm.
- In layman's term, Big-O describes the worst case of this algorithm.

- For example, $8n^2 - 3n + 4 \in O(n^2)$.
  - For large n, you could ignore the last two terms. (Why?)
  - It is easy to find a constant $c > 0$ so that $cn^2 > 8n^2$, say $c = 9$.
  - Hence the statement is proved.
- Also, $8n^2 - 3n + 4 \in O(n^3)$ but we seldom say this. (Why?)
- However, $8n^2 - 3n + 4 \notin O(n)$. (Why?)
- What is this analysis related to the algorithm?
- Any insight?

Common Simple Functions[6]



constant	*logarithm*	*linear*	*n-log-n*	*quadratic*	*cubic*	*exponential*
1	$\log n$	n	$n \log n$	n^2	n^3	a^n

[6]See Table 4.1 and Figure 4.2 in Goodrich and etc, p. 161.

Remarks

- We often make a trade-off between time and space.
  - Unlike time, we can reuse memory.
  - Users are sensitive to time.
- Playing game well is hard.[7]
- Solve the problem P ?= NP, which is one of Millennium Prize Problems.[8]

[7] See https://en.wikipedia.org/wiki/Game_complexity.
[8] See https://en.wikipedia.org/wiki/P_versus_NP_problem.

"All roads lead to Rome."

– Anonymous

"但如你根本並無招式，敵人如何來破你的招式？"

– 風清揚。笑傲江湖。第十回。傳劍

```
>> Lecture 3
>>
>>              -- Graphics
>>
```

Introduction

- Engineers use graphic techniques to make the information easier to understand.
- With graphs, it is easy to identify trends, pick out highs and lows, and isolate data points that may be measurement or calculation errors.
- Graphs can also be used as a quick check to determine if a computer solution is yielding expected results.
- A set of ordered pairs is used to identify points on a 2D graph.

2D Line Plot

- **plot**(x, y) creates a 2D line plot for all (x, y) pairs in order.
- You may use more parameters for the plot as follows:

Data markers†		Line types		Colors	
Dot (·)	.	Solid line	-	Black	k
Asterisk (*)	*	Dashed line	- -	Blue	b
Cross (×)	×	Dash-dotted line	-.	Cyan	c
Circle (o)	o	Dotted line	:	Green	g
Plus sign (+)	+			Magenta	m
Square (□)	s			Red	r
Diamond (◇)	d			White	w
Five-pointed star (★)	p			Yellow	y

†Other data markers are available. Search for "markers" in MATLAB Help.

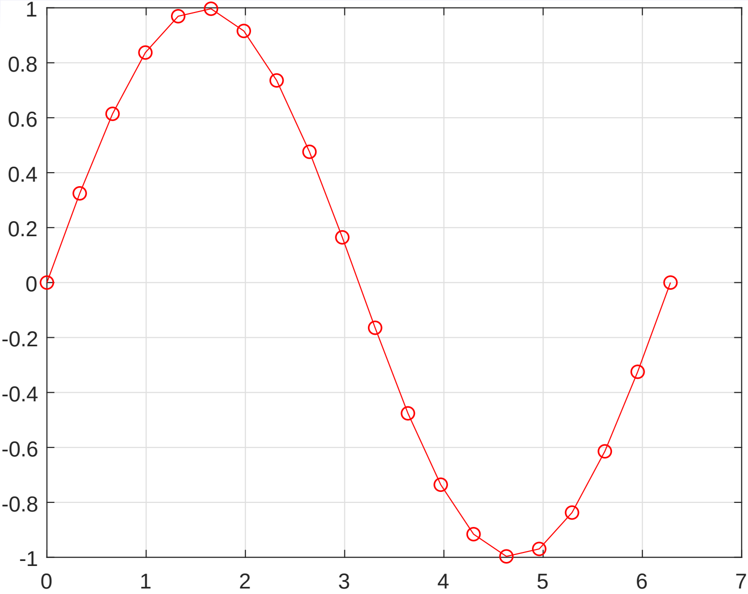
Example

```
clear; clc; close all;

x = linspace(0, 2 * pi, 20);
y = sin(x);

figure; plot(x, y, "r-o");
grid on;
```

- Call **figure** to create a figure.
- Use **close** to close all figures or specific one.
- Use **grid** to add the gray grid as the background.

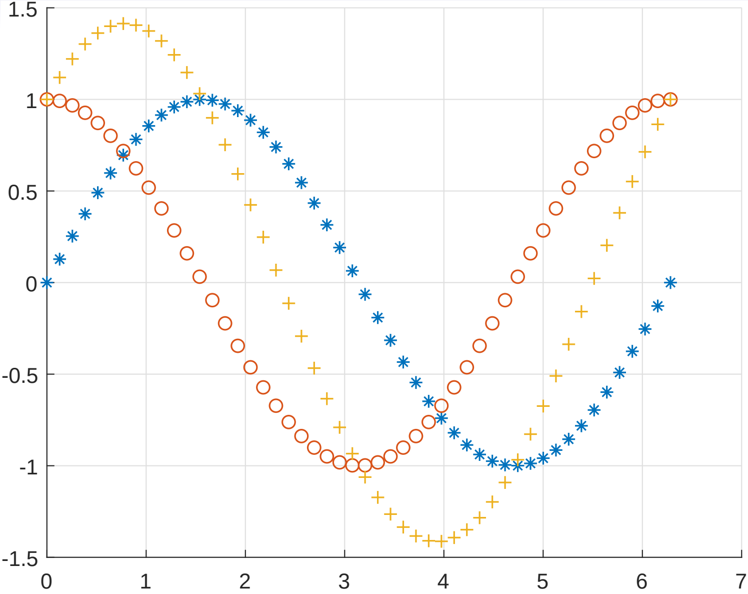

1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
1
2
3
4
5
6
7

Example: Multiple Curves

```
clear; clc; close all;

x = linspace(0, 2 * pi, 50);
figure; hold on; grid on;
plot(x, sin(x), '*');
plot(x, cos(x), 'o');
plot(x, sin(x) + cos(x), '+');
```

- Use **hold** to put multiple curves in the same figure.


1.5
1
0.5
0
-0.5
-1
-1.5
0
1
2
3
4
5
6
7

Selected Annotations

- Use **title** to add a title to the plot.
- Use **xlabel** to add a label to the x axis of the plot.
- Use **ylabel** to add a label to the y axis of the plot.
- Use **legend** to add legends for lines.
- More annotations can be created by **annotation**.[9]
- Note that you can always generate the codes associated with the plot you modified.

[9] See https://www.mathworks.com/help/matlab/examples/annotating-plots.html.

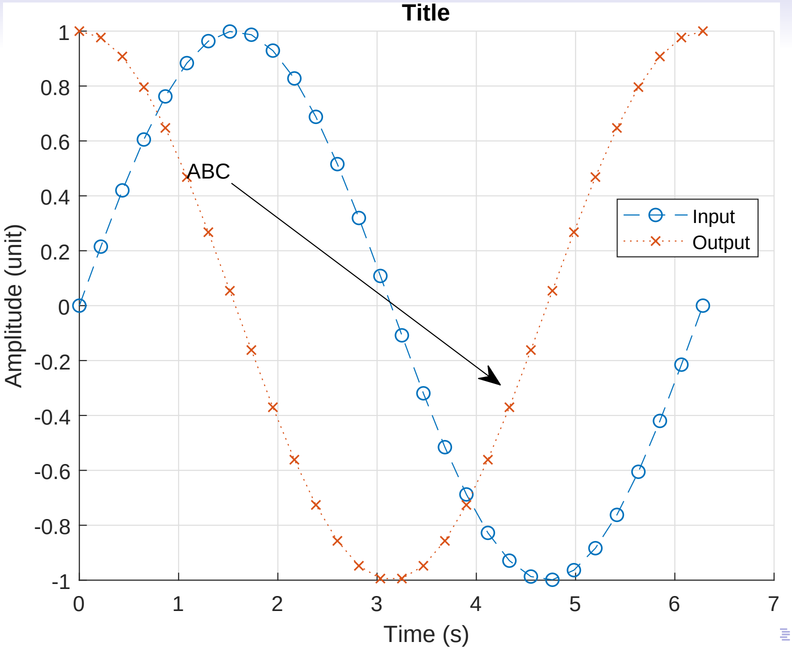
Example

```
clear; clc; close all;

x = linspace(0, 2 * pi, 30);
y = sin(x); z = cos(x);

figure; hold on; grid on;
plot(x, y, "o--");
plot(x, z, "x:");
legend("Input", "Output", "location", "best");

xlabel("Time (s)"); ylabel("Amplitude (unit)");
title("Title");
annotation("textarrow", [.3, .6], [.7, .4] , ...
            "String", "ABC");
```


Title
ABC
Input
Output
Amplitude (unit)
Time (s)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
1
2
3
4
5
6
7

Graphics Objects

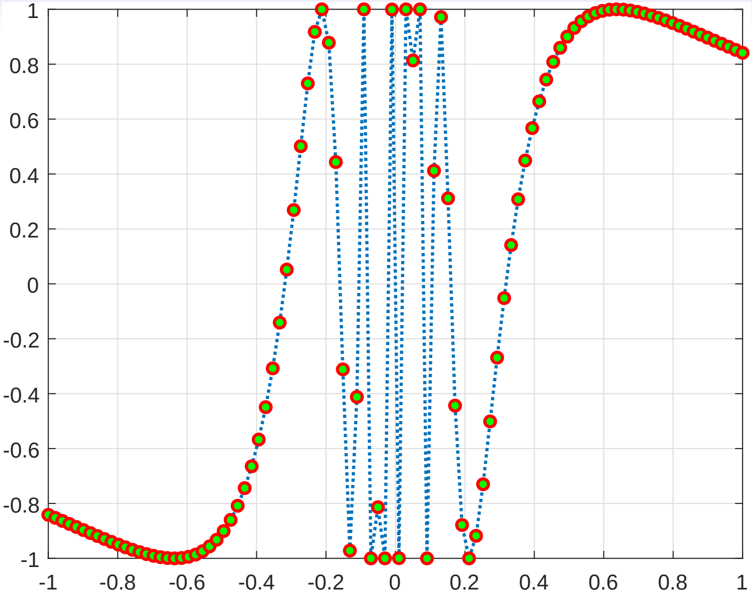
- You can use *plot tool* (in the figures) to change the properties.
- Graphics objects are the components for data visualization.
- Each object can be assigned to a unique identifier, called a graphics handle.
- Via graphics handles, you can manipulate their properties[10] by the following instructions:
  - **set**: set properties.
  - **get**: query properties.

[10] See `http://www.mathworks.com/help/matlab/graphics-object-properties.html`.

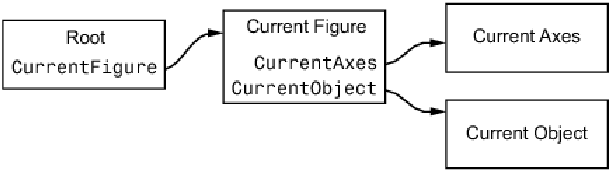
Example

```
clear; clc; close all;

x = linspace(-1, 1, 100);
h = plot(x, sin(1 ./ x));
grid on;
set(h, "Marker", "o");
set(h, "MarkerSize", 5);
set(h, "LineWidth", 1.5);
set(h, "LineStyle", ":");
set(h, "MarkerEdgeColor", "r");
set(h, "MarkerFaceColor", "g");
```


1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1

Graphics Object Identification[11]

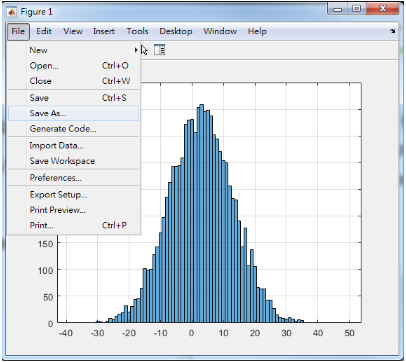


- **gcf**: **g**et **c**urrent **f**igure
- **gca**: **g**et **c**urrent **a**xis
- **gco**: **g**et **c**urrent **o**bject

[11]See http://www.mathworks.com/help/matlab/creating_plots/accessing-object-handles.html.

Output Figures

- You can save one figure as a specific image format.
  - For example, bmp, jpeg, and eps.
- Use the hot key ctrl + s.



- You can also use **print** to save the figures.[12]

```
clear; clc; close all;

x = linspace(0, 2 * pi, 20);
y = sin(x);

figure; plot(x, y, "r-o"); grid on;
print(gcf, "-djpeg", "sin.jpg", "-r300");
```

- Use **saveas** to save figure in a specific file format.[13]
- Use **savefig** to save figure and contents to fig-file.[14]

[12] See http://www.mathworks.com/help/matlab/ref/print.html.
[13] See https://www.mathworks.com/help/matlab/ref/saveas.html.
[14] See https://www.mathworks.com/help/matlab/ref/savefig.html.

Exercise: TWSE:IND

```
clear; clc; close all;

[~, ~, raw] = xlsread("y9999.xlsx");
prices = [raw{4 : end, 2}];
volumes = [raw{4 : end, 3}];
dates = datetime(raw(4 : end, 1), ...
                  "format", "yyyy/MM/dd");

fig1 = figure;
plot(dates, prices); grid on;
ylabel("TWSE:IND");
annotation(fig1, "arrow", [0.4 0.88], [0.28 0.65]);
```

- Use **datetime** to convert a date string to a datetime object.
- Note that you need to specify a date format, say `"yyyy/MM/dd"`.


TWSE:IND
13000
12000
11000
10000
9000
8000
7000
6000
5000
4000
2008
2010
2012
2014
2016
2018
2020

Bar Plot[15]

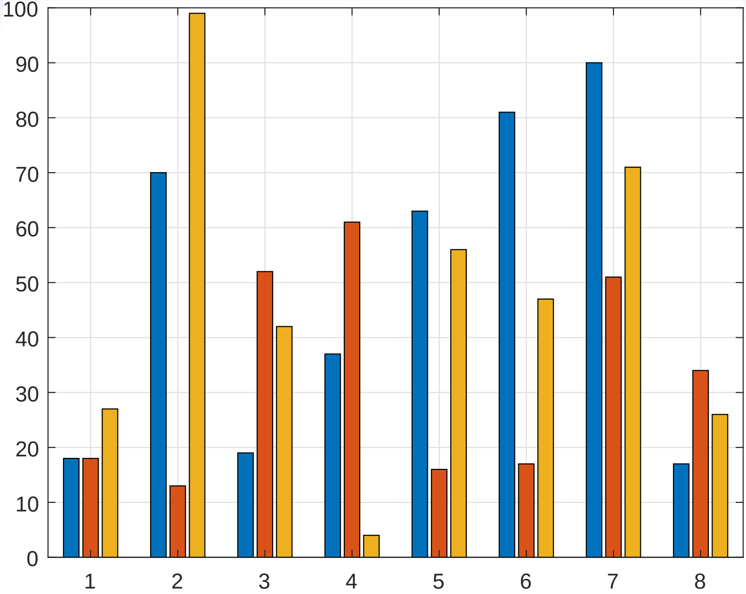
- Use **bar** to draws a bar chart, for example,

```
clear; clc; close all;

x = randi(100, 8, 3);
bar(x); grid on;
```

- Try **barh**.

[15] See http://www.mathworks.com/help/matlab/ref/bar.html and http://www.mathworks.com/help/matlab/creating_plots/overlay-bar-graphs.html.

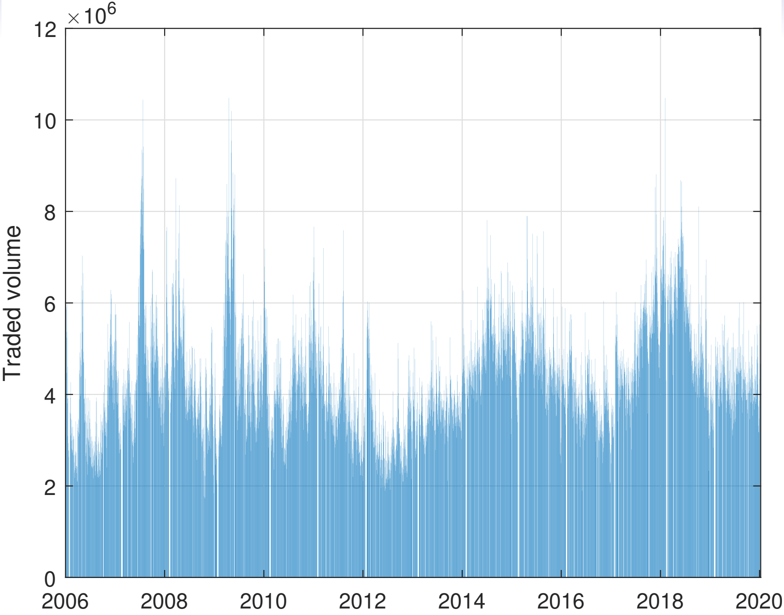

100
90
80
70
60
50
40
30
20
10
0
1
2
3
4
5
6
7
8

Exercise: Traded Volumes of TWSE:IND

```
clear; clc; close all;

[~, ~, raw] = xlsread("y9999.xlsx");
prices = [raw{4 : end, 2}];
volumes = [raw{4 : end, 3}];
dates = datetime(raw(4 : end, 1), ...
                 "format", "yyyy/MM/dd");

figure;
bar(dates, volumes); grid on;
ylabel("Traded volume");
```


×10^6
12
10
8
6
4
2
0
Traded volume
2006
2008
2010
2012
2014
2016
2018
2020

Dual y-Axes Plot

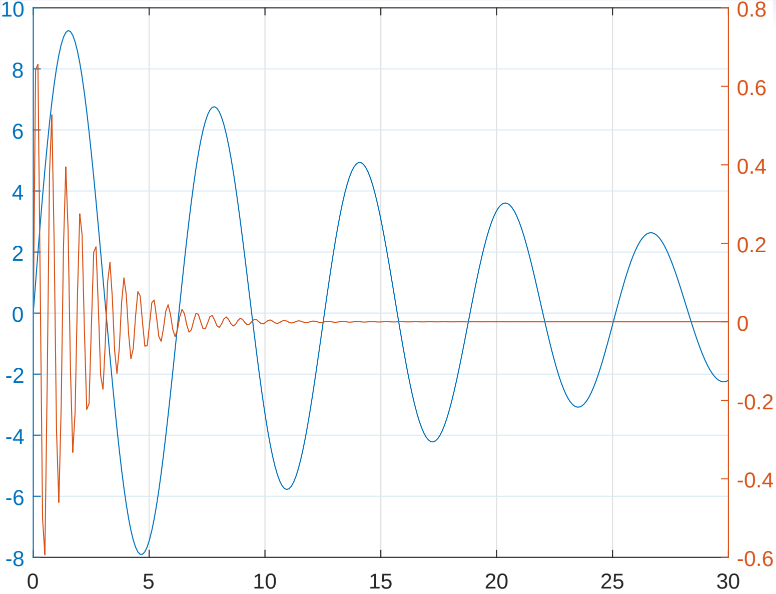
- Use **yyaxis** to specify the left/right y axis, for example,

```
clear; clc; close all;

x = linspace(0, 30, 300);
y1 = 10 * exp(-0.05 * x) .* sin(x);
y2 = 0.8 * exp(-0.5 * x) .* sin(10 * x);

figure;
yyaxis left;
plot(x, y1); grid on;
yyaxis right;
plot(x, y2);
```

- Use **plotyy** in old version.

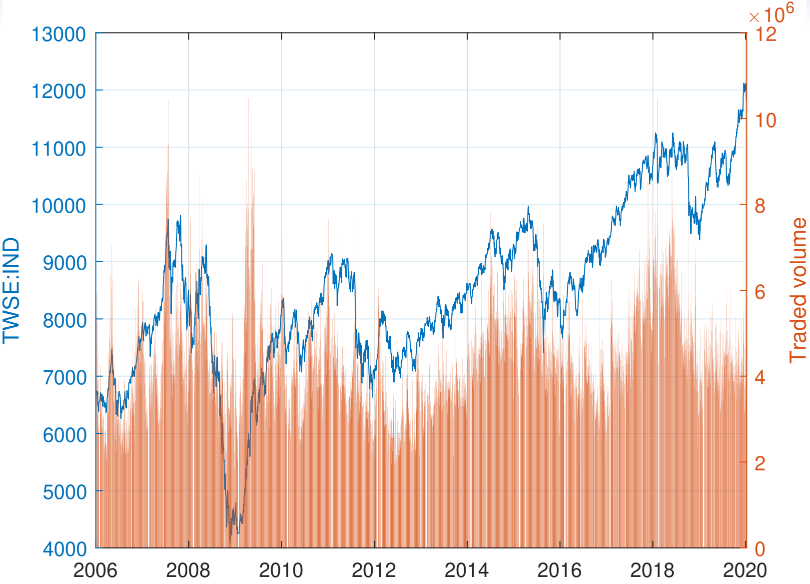

10
8
6
4
2
0
-2
-4
-6
-8
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
0
5
10
15
20
25
30

Exercise: Index feat. Volume in One Figure

```
clear; clc; close all;

[~, ~, raw] = xlsread("y9999.xlsx");
prices = [raw{4 : end, 2}];
volumes = [raw{4 : end, 3}];
dates = datetime(raw(4 : end, 1), ...
                 "format", "yyyy/MM/dd");

yyaxis left; plot(dates, prices);
ylabel("TWSE:IND"); grid on;
yyaxis right; bar(dates, volumes);
ylabel("Traded volume"); grid on;
```



TWSE:IND
Traded volume
$\times 10^6$
13000
12000
11000
10000
9000
8000
7000
6000
5000
4000
12
10
8
6
4
2
0
2006
2008
2010
2012
2014
2016
2018
2020

Histogram Plot[17]

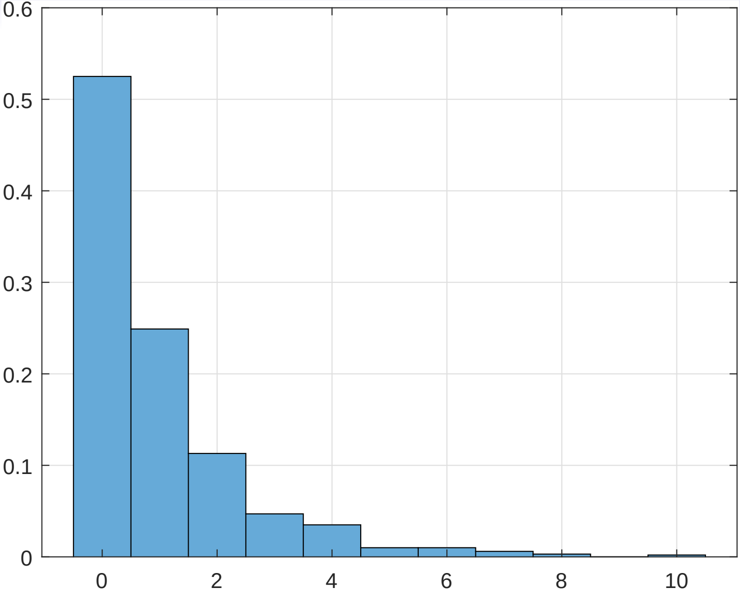
- Histograms group the numeric data into bins.
- Use **histogram** to create histogram plots.[16]

```
clear; clc; close all;

data = randn(1, 1e3) .^ 2;
figure;
histogram(data, ...
            "BinMethod", "integers", ...
            "Normalization", "probability");
grid on;
```

[16] If your version is before 2014, use **hist**.

[17] More details could be found in `https://www.mathworks.com/help/matlab/ref/matlab.graphics.chart.primitive.histogram.html`.

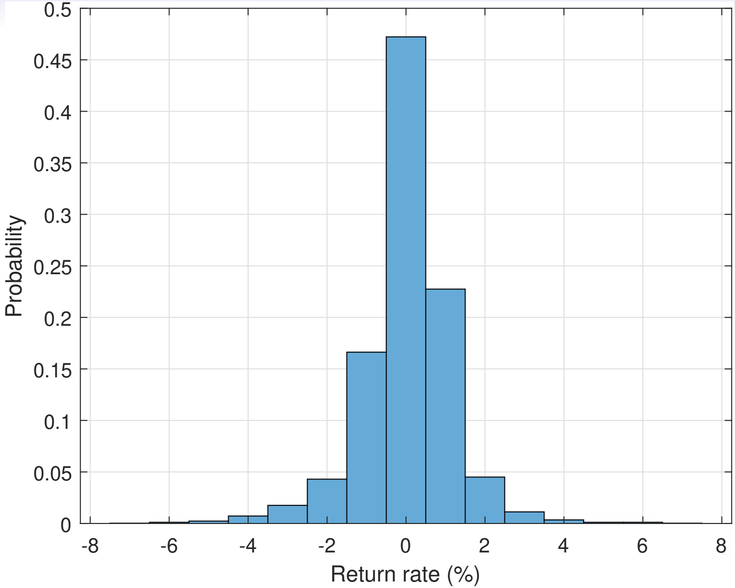

0.6
0.5
0.4
0.3
0.2
0.1
0
0
2
4
6
8
10

Exercise: Distribution of Return Rates of TWSE:IND

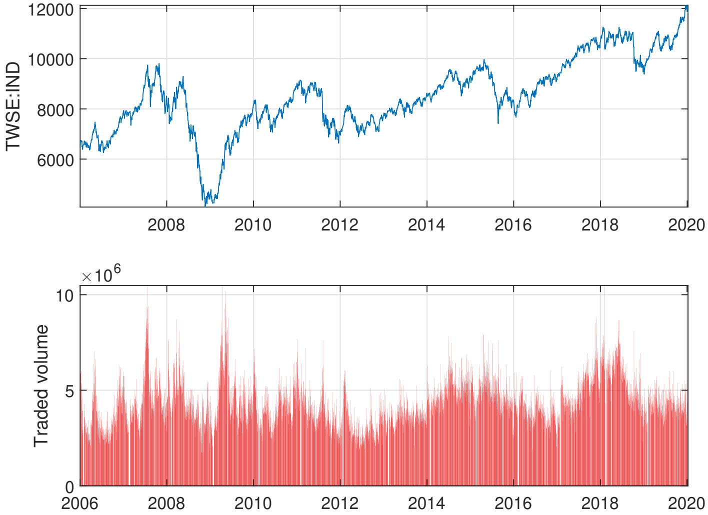
```
clear; clc; close all;

[~, ~, raw] = xlsread("y9999.xlsx");
prices = [raw{4 : end, 2}];
volumes = [raw{4 : end, 3}];
dates = datetime(raw(4 : end, 1), ...
                 "format", "yyyy/MM/dd");
return_rates = diff(prices) ./ prices(1 : end - 1);

figure;
histogram(return_rates * 100, ...
          "binmethod", "integer", ...
          "normalization", "probability");
xlabel("Return rate (%)");
ylabel("Probability"); grid on;
```


Probability
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
-8
-6
-4
-2
0
2
4
6
8
Return rate (%)

Grid Plot: **subplot**[18]



[18] See `https://www.mathworks.com/help/matlab/ref/subplot.html`.

```
clear; clc; close all;

[~, ~, raw] = xlsread("y9999.xlsx");
prices = [raw{4 : end, 2}];
volumes = [raw{4 : end, 3}];
dates = datetime(raw(4 : end, 1), ...
                 "format", "yyyy/MM/dd");

figure;
subplot(2, 1, 1); plot(dates, prices); grid on;
ylabel("TWSE:IND");
subplot(2, 1, 2); bar(dates, volumes, "r"); grid on;
ylabel("Traded volume");
```

- Use **subplot**(m, n, p) to divide the current figure into an m-by-n grid and use p to specify the certain subplot.

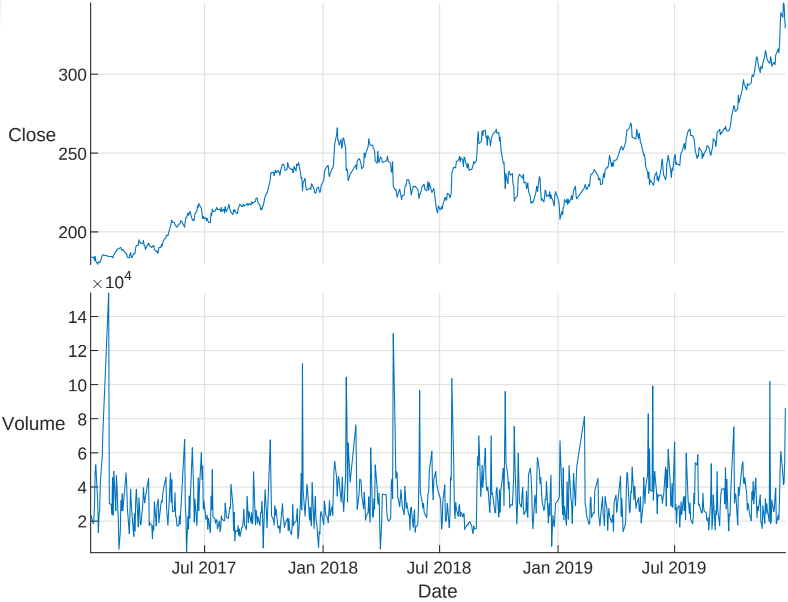
Digression: Table[19]

- Use **table** to create a table for column-oriented or tabular data that is often stored as columns in a spreadsheet.
- Use **detectImportOptions** to create import options based on the contents of a file (if **readtable** cannot read files correctly).
- Use **stackedplot** to draw a stacked plot of several variables with common x-axis.

[19] See https://www.mathworks.com/help/matlab/tables.html.

```
clear; clc; close all;

filename = "2330.xlsx";
s2330 = readtable(filename, ...
                  detectImportOptions(filename));
% Delete the first two rows.
s2330(1 : 2, :) = [];
% Assign the header name for each column.
s2330.Properties.VariableNames = ["Date", "Open", ...
    "High", "Low", "Close", "Volume"];
% Convert date strings to datetime objects.
s2330.Date = datetime(s2330.Date, ...
                      "format", "yyyy-MM-dd");
% Use stackedplot to draw an interactive plot!
stackedplot(s2330, {"Close", "Volume"}, ...
            "xvar", "Date"); grid on;
```


Close
300
250
200
×10^4
Volume
14
12
10
8
6
4
2
Jul 2017
Jan 2018
Jul 2018
Jan 2019
Jul 2019
Date

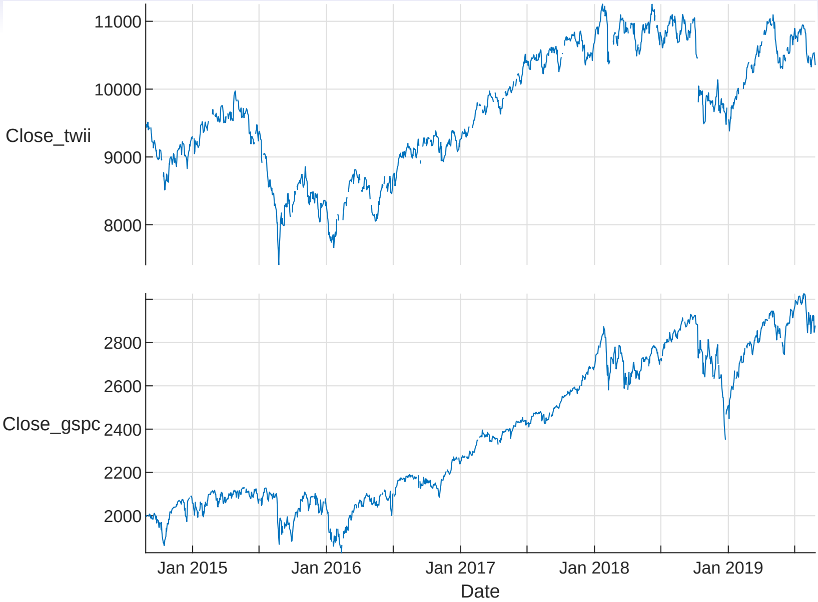
Selected Table Functions

- File I/O: **readtable**, **writetable**.
- Summary information: **head**, **tail**, **summary**, **stackedplot**.
- Sort, rearrange, and customize: **sortrows**, **unique**, **addvars**, **removevars**, **rows2vars**, **stack**, **unstack**, **inner2outer**.
- Join and set operations: **join**, **innerjoin**, **outerjoin**, **union**, **intersect**, **ismember**, **setdiff**, **setxor**.
- Apply functions to table contents: **varfun**, **rowfun**, **findgroups**, **splitapply**, **groupsummary**

Exercise: Merging Two Tables

```
clear; clc; close all;

gspc = readtable("^GSPC.csv");
twii = readtable("^TWII.csv", ...
                 detectImportOptions("^TWII.csv"));
twii.Date = datetime(twii.Date, ...
                     "format", gspc.Date.Format);
% Merge two time series by union of dates.
merged_table = outerjoin(twii, gspc, ...
                         "Keys", "Date", ...
                         "MergeKeys", 1);
stackedplot(merged_table, ...
            {"Close_twii", "Close_gspc"}, ...
            "xvariable", "Date"); grid on;
```



11000
10000
9000
8000
Close_twii
2800
2600
2400
2200
2000
Close_gspc
Jan 2015
Jan 2016
Jan 2017
Jan 2018
Jan 2019
Date

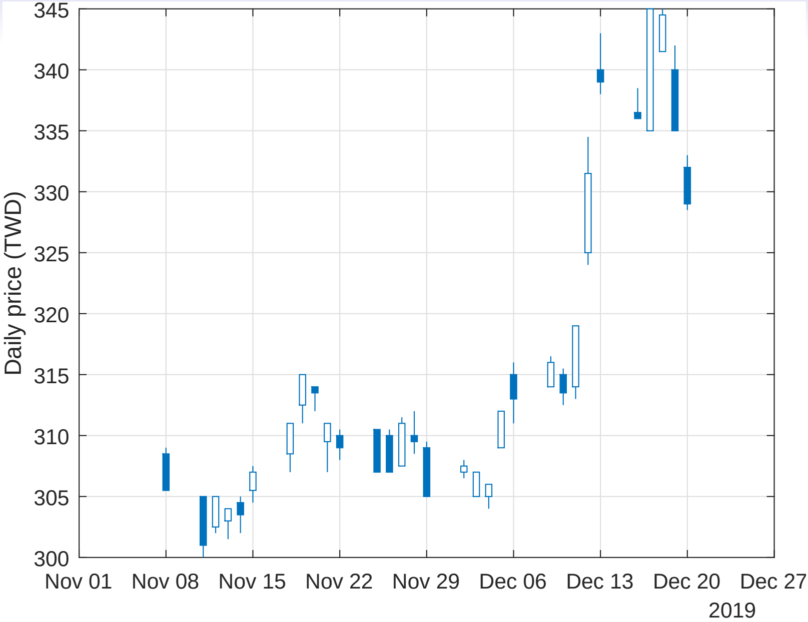
Candle Chart with Timetable[20]

```
% Ingore the part identical to the previous ...
    example of 2330.

s2330 = table2timetable(s2330, "RowTimes", "Date");
candle(s2330(end - 30 :  end, :)); % last 30 days
ylabel("Daily price (TWD)");
```

- Use **timetable** to convert the table (with variable names: `"Open"`, `"High"`, `"Low"`, `"Close"`) to a timetable by specifying the *RowTimes*.
- Try **priceandvol**.

[20] See https://www.mathworks.com/help/finance/candle.html and https://www.mathworks.com/help/matlab/timetables.html with https://www.mathworks.com/help/finance/examples/using-timetables-in-finance.html.


Daily price (TWD)
345
340
335
330
325
320
315
310
305
300
Nov 01
Nov 08
Nov 15
Nov 22
Nov 29
Dec 06
Dec 13
Dec 20
Dec 27
2019

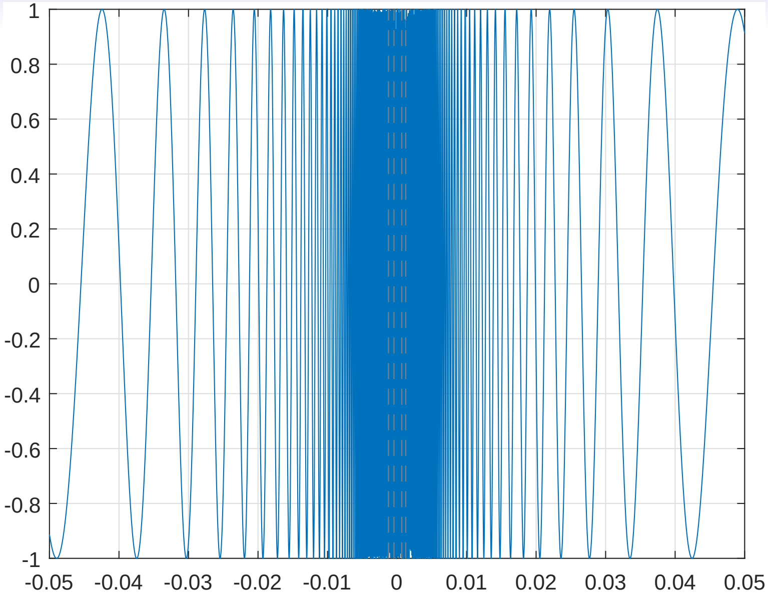
Smart Plot: **fplot**

- Use **fplot** to make a line plot over a specific range with adaptive steps.
- You may assign a function in a string form to **fplot**.[21]

```
clear; clc; close all;

fplot("sin(1 / x)", [-0.05, 0.05]); grid on;
```

[21] Warning: **fplot** will not accept character vector or string inputs in a future release.

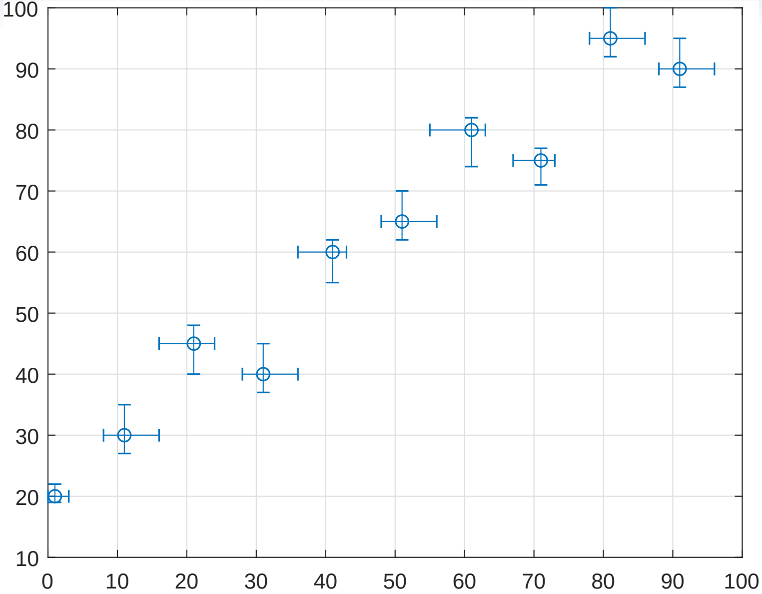

1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-0.05
-0.04
-0.03
-0.02
-0.01
0
0.01
0.02
0.03
0.04
0.05

errorbar[22]

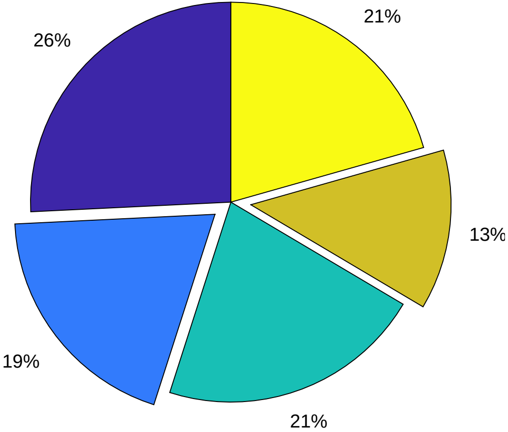
```
clear; clc; close all;

x = 0 : 10 : 100;
y = [20 30 45 40 60 65 80 75 95 90];
yneg = [1 3 5 3 5 3 6 4 3 3];
ypos = [2 5 3 5 2 5 2 2 5 5];
xneg = [1 3 5 3 5 3 6 4 3 3];
xpos = [2 5 3 5 2 5 2 2 5 5];
errorbar(x, y, yneg, ypos, xneg, xpos, "o");
grid on;
```

[22] See https://www.mathworks.com/help/matlab/ref/errorbar.html.


100
90
80
70
60
50
40
30
20
10
0
10
20
30
40
50
60
70
80
90
100

Pie Chart



```
clear; clc; close all;

X = rand(1, 5);
labels = {"A", "B", "C", "D", "E"};
explode = [0, 1, 0, 1, 0];
pie(X, explode, labels);
```

- Use **pie** to create a pie chart.[23]
- Note that the explode vector is used to offset slices for the nonzero elements.

[23]See https://www.mathworks.com/help/matlab/ref/pie.html.

Word Cloud

market_values

```matlab
clear; clc; close all;

[~, ~, raw] = xlsread("twse_mktValue.xlsx");

stock_ticks = string(raw(4 : end, 1));
idx = strcmp(raw(:, 3), "-"); % Find all "-"s.
raw(idx, 3) = {0};            % Replace them by 0.
market_values = [raw{4 : end, 3}]';

tbl = table(stock_ticks, market_values);
figure;
wordcloud(tbl, "stock_ticks", "market_values");
```

- Use **strcmp** to compare strings and return a boolean vector.
- Use **wordcloud** to create a word cloud chart from text data.[24]

[24] See https://www.mathworks.com/help/matlab/ref/wordcloud.html.

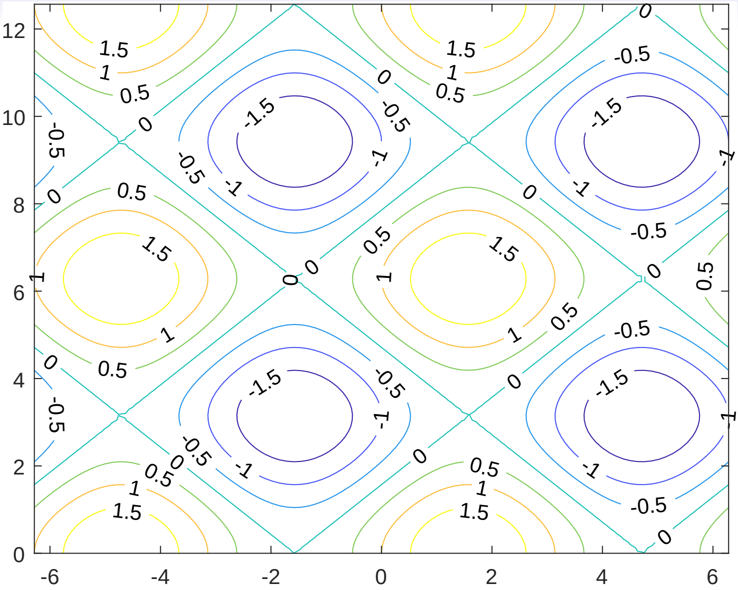
Contours[25]

- Use **meshgrid** to partition the specified range of x and y.
- Note that the return values are in form of matrices. (Why?)

```
clear; clc; close all;

x = linspace(-2 * pi, 2 * pi);
y = linspace(0, 4 * pi);
[X, Y] = meshgrid(x, y);
Z = sin(X) + cos(Y); % Using vectorization.
figure; contour(X, Y, Z, "showtext", "on");
```

[25] See https://www.mathworks.com/help/matlab/ref/contour.html. You may try **contourf**.


12
10
8
6
4
2
0
-6
-4
-2
0
2
4
6
1.5
1
0.5
0
-0.5
-1
-1.5

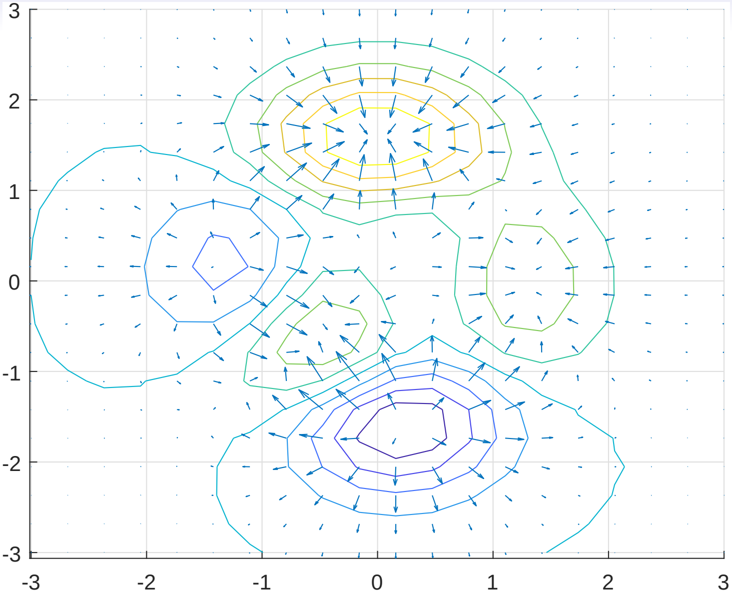
Quiver (Velocity) Plot

- Use **quiver**(x, y, u, v) to plot a vector (u, v) at the coordinate (x, y).
- Use **peaks** with a positive number as sample size to generate a set of 3d points.[26]

```
clear; clc; close all;

[x, y, z] = peaks(20);
[u, v] = gradient(z);
figure; hold on; grid on;
contour(x, y, z, 10);
quiver(x, y, u, v);
```

[26] See https://www.mathworks.com/help/matlab/ref/peaks.html.

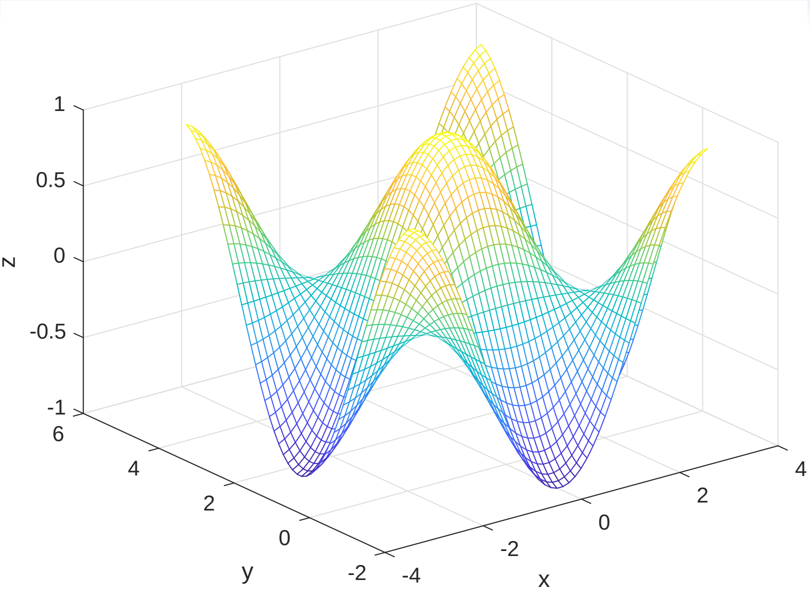

3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3

Mesh Plot

- Use **mesh** to draw a wireframe mesh.

```
clear; clc; close all;

x = linspace(-3, 3, 50);
y = x + pi / 2;
[X, Y] = meshgrid(x, y);
Z = cos(X) .* sin(Y);
figure; mesh(X, Y, Z); grid on;
xlabel("x"); ylabel("y"); zlabel("z");
```

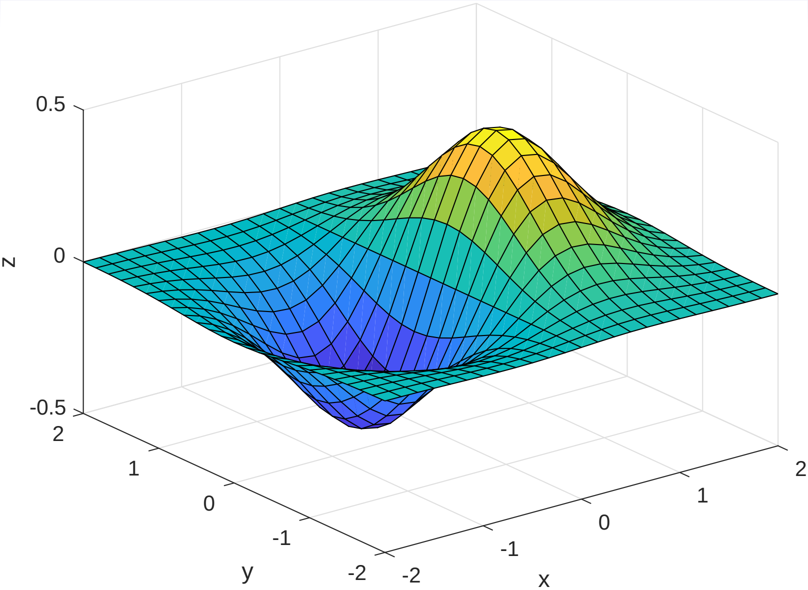

1
0.5
0
-0.5
-1
z
6
4
2
0
-2
y
-4
-2
0
2
4
x

Surface Plot

- Use **surf** to draw a colored surface.
- Try **meshz**, **meshc**, **surfc**, and **waterfall**.

```
clear; clc; close all;

x = linspace(-2, 2, 25);
y = linspace(-2, 2, 25);
[X, Y] = meshgrid(x, y); % form all x-y pairs
Z = X .* exp(-X .^ 2 - Y .^ 2);
surf(X, Y, Z);
xlabel("x"); ylabel("y"); zlabel("z");
```

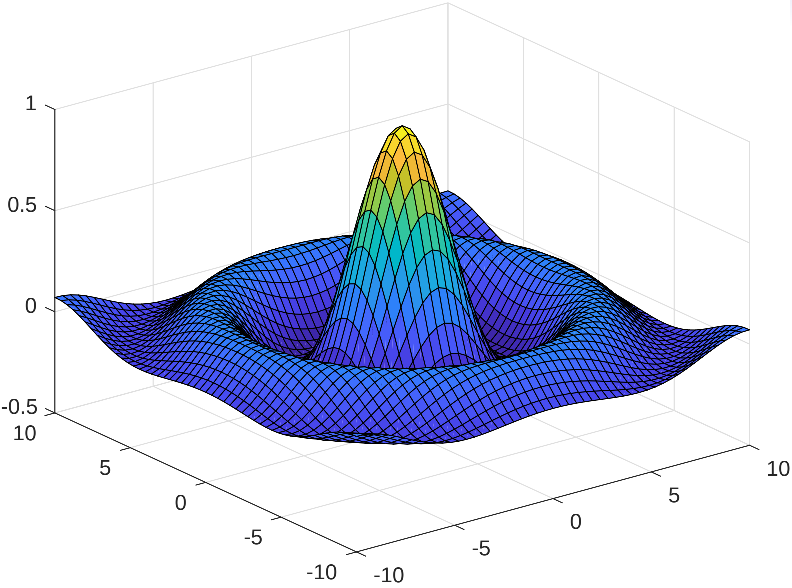

0.5
0
-0.5
z
2
1
0
-1
-2
y
-2
-1
0
1
2
x

Exercise

- Write a program to draw a surface plot for $sinc(R) = \frac{sin(R)}{R}$.
- Note that there exists a singularity at $R = 0$, which should be removed by replacing a zero with $eps = 2.2204 \times 10^{-16}$.

```
clear; clc; close all;

[X, Y] = meshgrid(linspace(-10, 10, 51));
R = sqrt(X .^ 2 + Y .^ 2);
R(R == 0) = eps; % Avoid the singularity.
Z = sin(R) ./ R;
surf(X, Y, Z);
```


1
0.5
0
-0.5
10
5
0
-5
-10
-10
-5
0
5
10

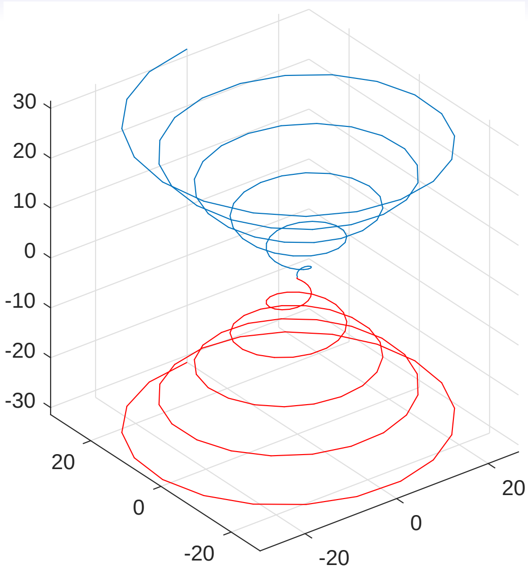
3D Line Plot

- Use **plot3** to draw a 3d curve.

```
clear; clc; close all;

t = linspace(0, 10 * pi, 100);
x = t .* sin(t); y = t .* cos(t);

figure;
plot3(x, y, t); hold on;
plot3(x, y, -t, "r");
axis equal; grid on;
```


30
20
10
0
-10
-20
-30
20
0
-20
-20
0
20

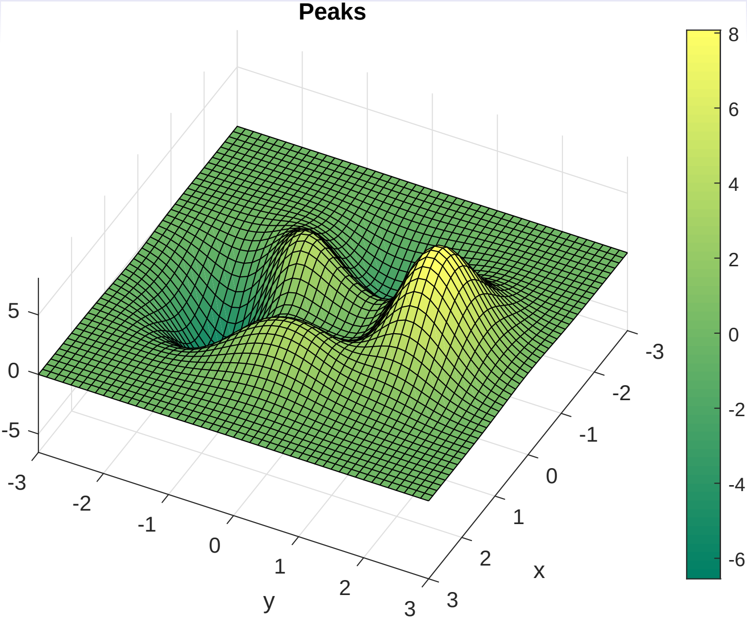
Misc[29]

- Use the button *Rotate 3D* to change the view angle.
- Use **view** to set the view angle.[27]
- Try **colorbar** and **colormap**.[28]

```
clear; clc; close all;

peaks;
view([117, 58]); % View angle in degree.
colorbar; % Appends a colorbar to the current axes.
colormap summer; % Change the colormap.
```

[27] az: azimuth (horizontal) rotation; el: vertical elevation.
[28] See `https://www.mathworks.com/help/matlab/ref/colormap.html`.
[29] See `https://www.mathworks.com/products/matlab/plot-gallery.html`.


Peaks
5
0
-5
-3
-2
-1
0
1
2
3
y
-3
-2
-1
0
1
2
3
x
8
6
4
2
0
-2
-4
-6