University of BRISTOL

Introduction To Scientific Computing

Basics of MATLAB

Dr. Fintan Healy
Room 1.31, Queens Building
fintan.healy@bristol.ac.uk

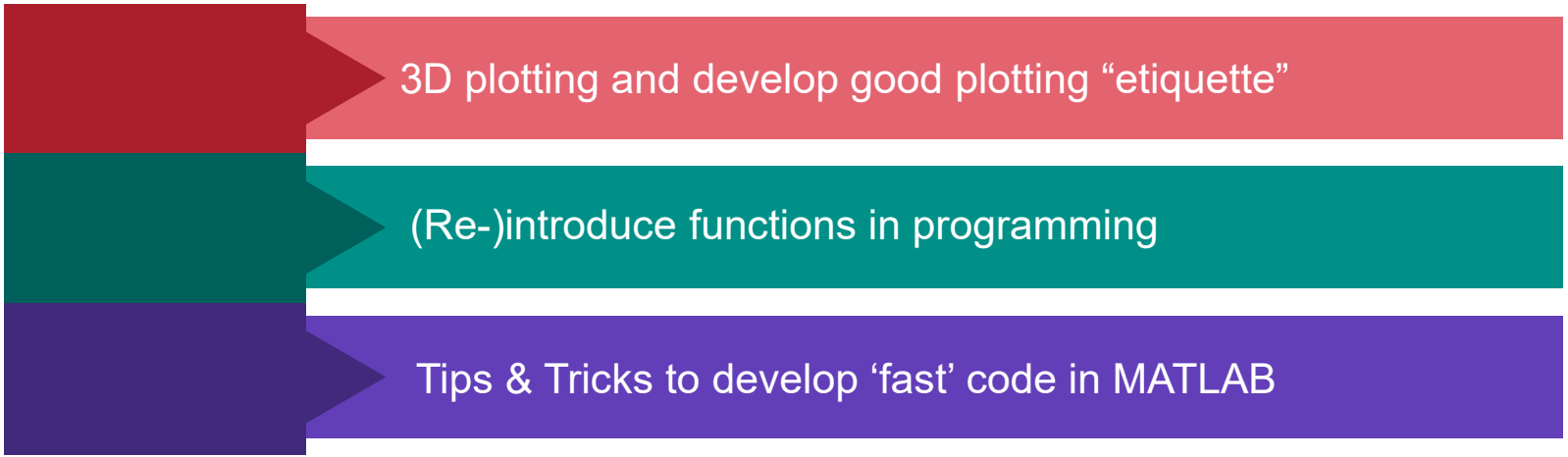
University of BRISTOL

Lecture 3

Syntax Basics 2

https://i2sc.fintanhealy.co.uk

Lecture Aims



MATLAB

- By now, you should feel relatively *"comfortable"* with MATLAB
- During your studies, you have learnt basic programming concepts through the lens of the Python language
  - but all these concepts translate to MATLAB!
- Remember which language you used depends on the task:

Capability	MATLAB	Python	C++
Scientific Programming	Good	Good	Good
Data Science	Good	Good	Poor
Dashboards	Poor	Excellent	Poor
Plotting	Good	Good-	Poor
Real-time Control-systems	Excellent	Poor	Good
Experiments	Good	Okay	Poor
Documentation	Excellent	Good-	Okay
Debugging	Excellent	Good*	Okay
3rd party Integration	Poor	Good	Okay
Deep learning	Okay	Excellent	Good
Execution time	Okay	Okay	Excellent
Community Contributions	Okay	Excellent	Good

Opinions based on programming requirements for "general engineering" in industry

University of BRISTOL

Plotting

bristol.ac.uk

Plotting Etiquette

- Last lecture, we learnt the basics of creating figures in MATLAB
  - Creating figures (figure;)
  - Plotting multiple lines (hold on)
  - Basic commands to alter appearance
- But this example figure will not get you many marks in your coursework!
- What's Missing?
  - Axes labels + units!
  - A legend (what is each line?)
  - A caption to describe what the figure is

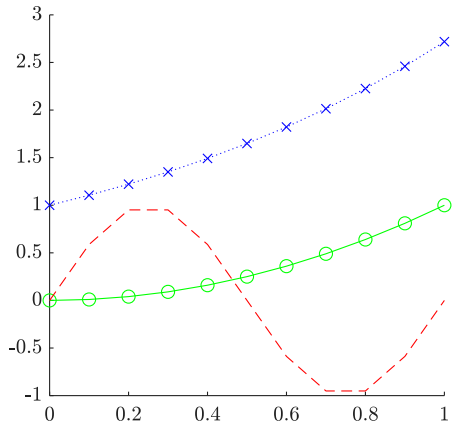
MATLAB

```
figure;
t = 0:0.1:1;
hold on;

y = t.^2;
plot(t,y,'go-');
%green line with circle markers

y2 = sin(2*pi*t);
plot(t,y2,'r--');
%red dashed line

y3 = exp(t);
plot(t,y3,'bx:');
%blue dotted line with x markers
```



Plotting Etiquette

Example Student Report

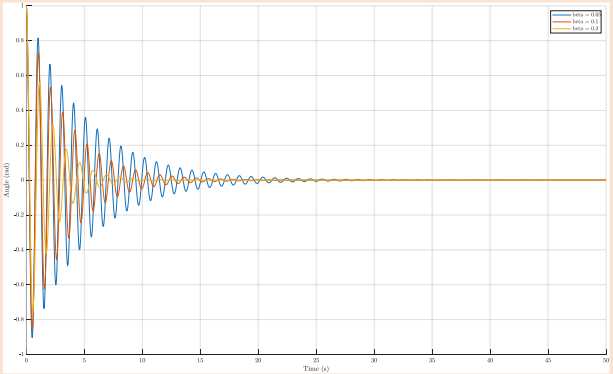


Figure 1 – Example response

Figure 1 shows the variation in the response of a damped pendulum when three damping coefficients are used. As the damping coefficient is increased, the rate of decay of the oscillations increases and the damped natural frequency is reduced…

- **Example problem**: plot the variation in the angle and angular velocity of a damped pendulum described by the equation

$$\theta = \theta_0 e^{-\beta t}\cos\left(t\sqrt{\omega_0^2 - \beta^2}\right)$$

where $\omega_0 = 1\text{rad s}^{-1}$, $\theta_0 = 1$ rad and β equals 0.2, 0.3, or 0.5. Describe the variation in the response…

What is wrong with this figure?

MATLAB

```
figure;
hold on;
t = 0:0.01:50;
for beta = [0.2,0.3,0.5]
theta = exp(-beta*t).*cos(2*pi*t*sqrt(1^2-beta^2));
plot(t, theta);
end
xlabel('Time (s)');
ylabel('Angle (rad)');
legend('beta = 0.2','beta = 0.3','beta = 0.5','Orientation','vertical');
grid on
```

Plotting Etiquette

Example Student Report

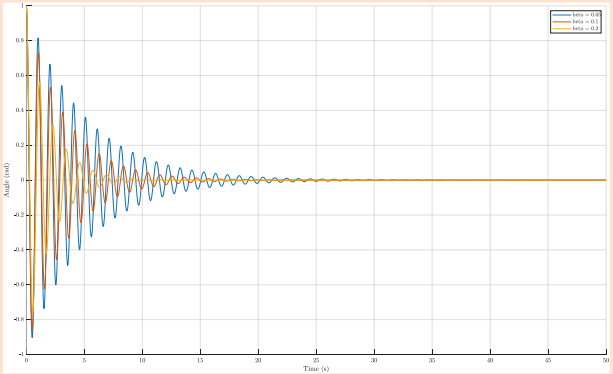


Figure 1 – Example response

Figure 1 shows the variation in the response of a damped pendulum when three damping coefficients are used. As the damping coefficient is increased, the rate of decay of the oscillations increases and the damped natural frequency is reduced…

MATLAB

```
xlim([0 5])
```

- **Lots of whitespace** - Content should take up >50% of the figure
- **Text is too small –** You can't read the axis labels or numbers
- **Poor figure caption –** A good figure caption makes the figure self-contained. E.g.
  - **No**: "Figure 1: Results of the experiment."
  - **Yes**: "Figure 1: Lift coefficient versus angle of attack for a NACA 0012 aerofoil at a Reynolds number of '*x*'."

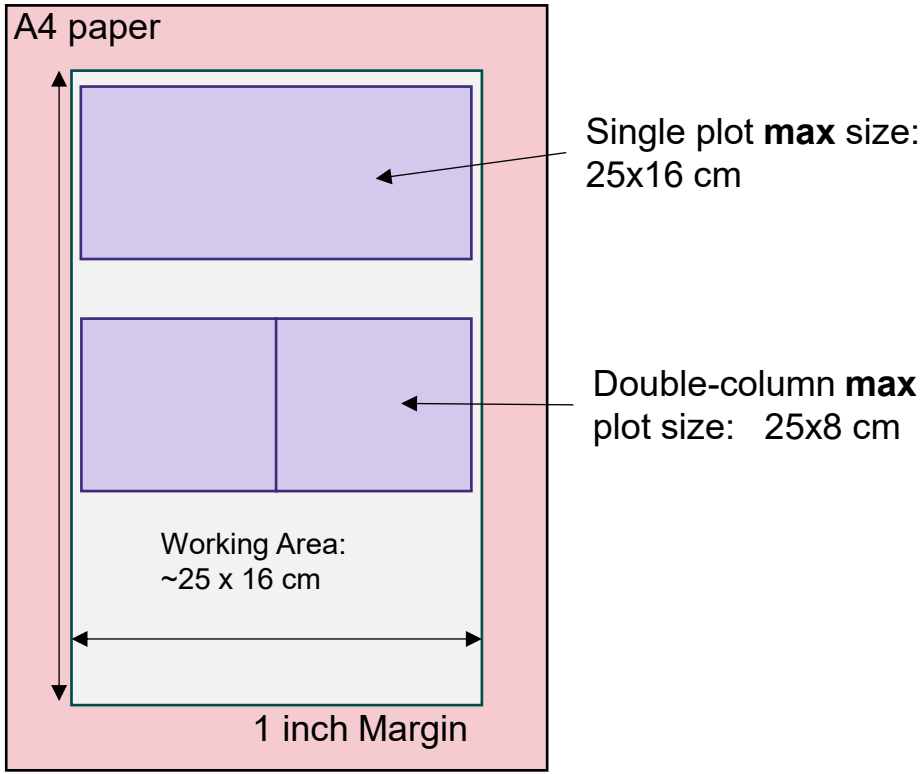
MATLAB

```
xlabel('Time [s]','FontSize',11)
```

Plotting Best Practice

- The font size in figures should be similar to the rest of the text
  - A4 paper has dimensions 29.7 x 21cm
  - Default margins 1 inch
- We can set the:
  - size of the figure
  - font size

  to ensure the figure text will be visible in a report



Plotting Best Practice

- 'figure' function can return a *handle* to the figure
- We can then set the size of the figure window
  - f.Position accepts a 1x4 array.
    - Position of the bottom left corner
    - Width
    - Height
- Multiple ways to set font size
  - Googling it is generally your best option!

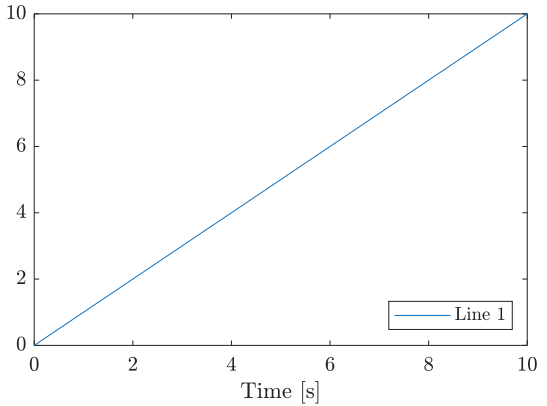
MATLAB

```
f = figure;
f.Units = "centimeters"; % set units to define size in
f.Position = [5 5 12 8]; % [xpos ypos width height]
% xpos and ypos are pos of bottom left corner of figure

plot(0:10,(0:10))
xlabel('Time [s]','FontSize',11) % set font size for xlabel

ax = gca; % get current axes
ax.FontSize = 11; % set default font size for axes

lg = legend('Line 1'); % return a legend object
lg.FontSize = 10; % set font size for legend
lg.Location = "southeast";
```



Plotting Best Practice

- Methods to save / export the figure
  - ‘file’ toolbar in the figure window
  - programmatically

MATLAB

```
copygraphics(gcf)
exportgraphics(gcf,'test.png')
exportgraphics(gcf,'test.png','Resolution',600)
exportgraphics(gcf,'test.pdf')
```

Note

‘gcf’ returns the currently active “figure handle”.
GCF – Get Current Figure…

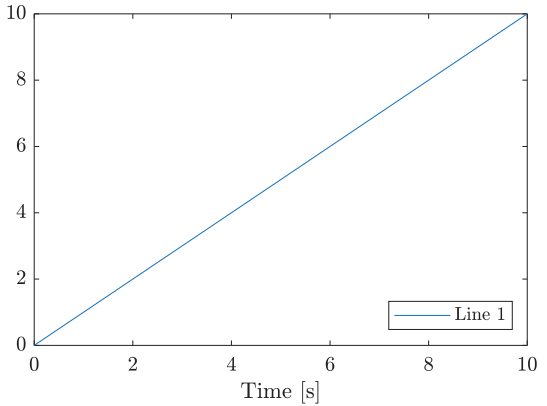
MATLAB

```
f = figure;
f.Units = "centimeters"; % set units to define size in
f.Position = [5 5 12 8]; % [xpos ypos width height]
% xpos and ypos are pos of bottom left corner of figure

plot(0:10,(0:10))
xlabel('Time [s]','FontSize',11) % set font size for xlabel

ax = gca; % get current axes
ax.FontSize = 11; % set default font size for axes

lg = legend('Line 1'); % return a legend object
lg.FontSize = 10; % set font size for legend
lg.Location = "southeast";
```



Plotting Etiquette

Example Student Report

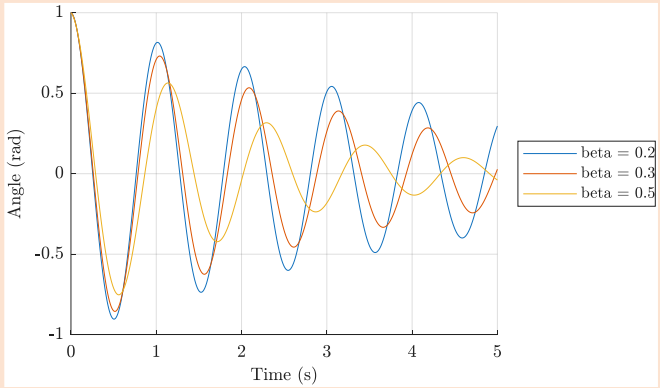


Figure 1 – Step response of a damped pendulum with different damping coefficients, β.

Figure 1 shows the variation in the response of a damped pendulum when three damping coefficients are used. As the damping coefficient is increased, the rate of decay of the oscillations increases and the damped natural frequency is reduced…

- **Example problem**: plot the variation in the angle and angular velocity of a damped pendulum described by the equation

$$\theta = \theta_0 e^{-\beta t}\cos\left(2\pi t\sqrt{\omega_0^2 - \beta^2}\right)$$

where $\omega_0 = 1\text{rad s}^{-1}$, $\theta_0 = 1$ rad and β equals 0.2, 0.3, or 0.5. Describe the variation in the response…

MATLAB

```
f = figure; hold on;
f.Units = "centimeters";
f.Position = [f.Position(1:2) 14 8];

t = 0:0.01:50;
for beta = [0.2,0.3,0.5]
theta = exp(-beta*t).*cos(2*pi*t*sqrt(1^2-beta^2));
plot(t, theta);
end

xlabel('Time (s)','FontSize',10);
ylabel('Angle (rad)','FontSize',10);
lg = legend('beta = 0.2','beta = 0.3','beta = 0.5');
lg.FontSize = 9;
lg.Location = "eastoutside";
lg.Orientation = "vertical";
grid on
xlim([0 5])
```

University of BRISTOL

3D Plotting

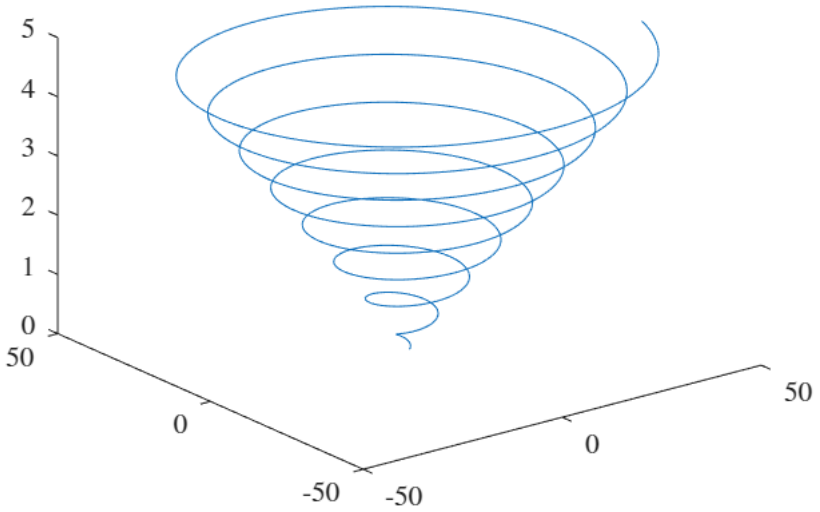
bristol.ac.uk

3D Line Plots

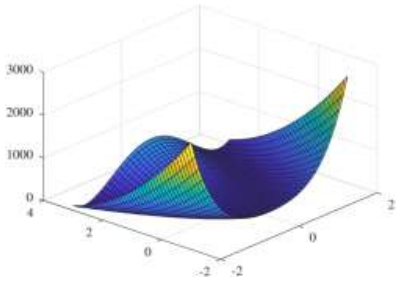
- MATLAB can plot line graphs in 3D.
- The syntax of the ‘plot3’ command is similar to the standard plot command.

MATLAB

```
figure;
t =[0:0.01:50];
x=cos(t).*t;
y=sin(t).*t;
z =0.1* t;
plot3(x,y,z)
```



3D Surface Plots



MATLAB

```
x = 1:5;
y = 1:5;
[X,Y] = meshgrid(x,y);
```

- If we want to visualise a scalar function of two variables, e.g.

$$z = (1-x)^2 + 100(y-x^2)^2$$

- We must first create a structured grid on which to plot the surface
- The built-in function `meshgrid` is ideal for this
  - It creates two arrays which define the XY coordinates of a uniform grid

$x =$

1	2	3	4

$y =$

1	2	3	4

$X =$

1	2	3	4
1	2	3	4
1	2	3	4
1	2	3	4

$Y =$

1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4

$(X, Y) =$

(1,1)	(2,1)	(3,1)	(4,1)
(1,2)	(2,2)	(3,2)	(4,2)
(1,3)	(2,3)	(3,3)	(4,3)
(1,4)	(2,4)	(3,4)	(4,4)

3D Surface Plots

- If we want to visualise a scalar function of two variables, e.g.

$$z = (1-x)^2 + 100(y-x^2)^2$$

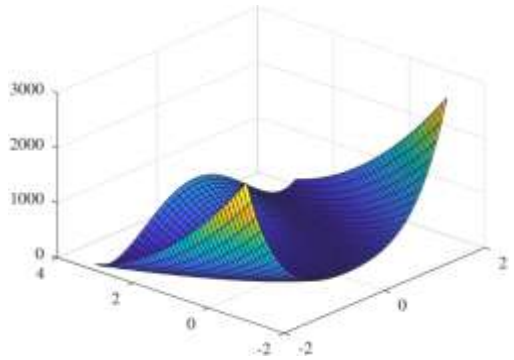
- We create a structured grid using `meshgrid`
- we can use 'surf' to plot a 3D surface
- we can use 'contour' to plot a 2D contour plot

MATLAB

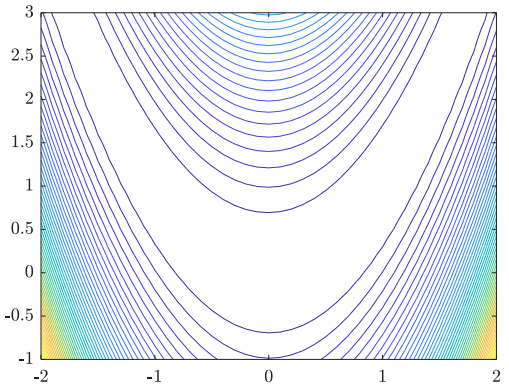
```
x = -2:0.1:2;
y = -1:0.1:3;
[X,Y]= meshgrid (x,y);
F =(1 -X ).^2+100*(Y-X .^2).^2;

figure;
surf (X,Y,F);

figure;
contour(X,Y,F,50);
```



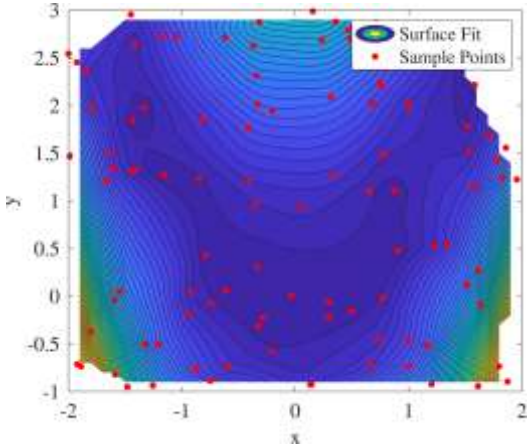
Surface plot



Contour plot

Plotting Irregular Data?

- What if our data is not regularly spaced?
- Interpolate onto a regularly spaced grid.
  - Can use `polyfit` to fit a polynomial curve
    - The underlying surface needs to be representable as a polynomial!
  - Can use `scatteredInterpolant` to smoothly interpolate between values.



MATLAB

```
% Monte Carlo sample of the Rosenbrock function
rng(1);                 % reproducible random numbers
N  = 100;
xs = -2 + 4*rand(N,1);                % x in [-2, 2]
ys = -1 + 4*rand(N,1);                % y in [-1, 3]
Fs = (1 - xs).^2 + 100*(ys - xs.^2).^2;

% Build the interpolant from the scattered points
interp = scatteredInterpolant(xs, ys,…
 Fs, 'natural', 'none');

% Evaluate it on a regular grid
x = -2:0.1:2;
y = -1:0.1:3;
[X,Y] = meshgrid(x,y);
F = interp(X,Y);

% Contour plot, with the sample points overlaid
figure;
contourf(X,Y,F,50);
hold on;
plot(xs, ys, 'r.', 'MarkerSize', 15);
xlabel('x'); ylabel('y');
```

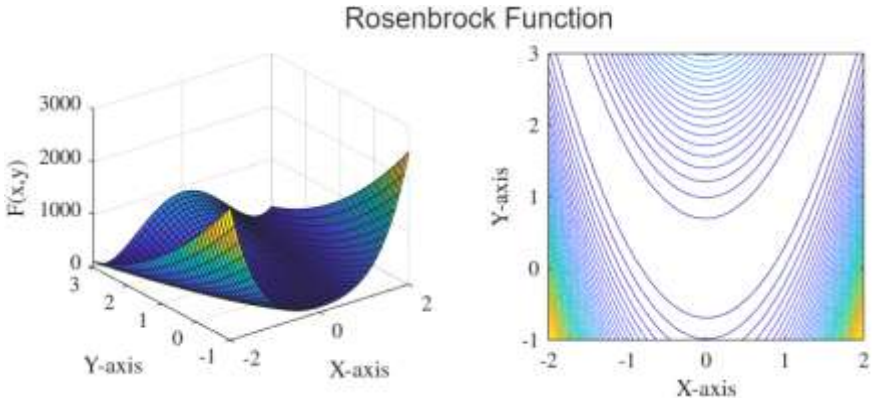
Multi-Plots

- You can include more than one figure in the same figure window by using the function ‘tiledlayout(m,n)’
  - m and n are the number of rows and columns, respectively
- You can then use ‘nexttile(x)’ to switch between tiles on your figure

MATLAB

```
x = -2:0.1:2;
y = -1:0.1:3;
[X,Y]= meshgrid (x,y);
F =(1 -X ).^2+100*(Y-X .^2).^2;
f=figure;
f.Units = "centimeters";
f.Position = [5,5,16,7];

t = tiledlayout(1,2);
nexttile(1);
surf (X,Y,F);
xlabel('X-axis'); ylabel('Y-axis'); zlabel('F(x,y)');
nexttile(2);
contour(X,Y,F,50);
xlabel('X-axis'); ylabel('Y-axis');
t.Title.String = 'Rosenbrock Function';
```



University of BRISTOL

Working With Functions In MATLAB

bristol.ac.uk

Why Functions?

- A function is a reusable block of code designed to perform a specific task.
- Key Features:
  - **Reusability**: Write once, use multiple times
  - **Modularity**: Break down a program into smaller, logical parts
  - **Abstraction**: Hide implementation details and focus on functionality.
  - **Maintainability**: Easier to debug and update code.
- `[<outputs>] = <func_name>(<inputs>)`
  - is called the function signature

bristol.ac.uk

- Functions in MATLAB have the following syntax

MATLAB

```
function [<outputs>] = <func_name>(<inputs>)
%FUNC_NAME Summary of this function goes here
%   Detailed explanation goes here

% Your code here
end
```

- Functions in MATLAB can have multiple inputs and outputs

MATLAB

```
function [L] = calcLift(rho ,vel ,area ,Cl)
   % LIFT Calculate the lift of a wing
   L =0.5* rho* vel.^2 * area *Cl;
end
```

Note

Using this Docstring means ‘help calcLift’ will show you documentation.

Example Functions

Task:

- Calculate lift and drag for an aircraft at different altitudes and velocities, with the following constants $S = 20m^2$, $C_L = 0.5$ and $C_D = 0.02$
- Assume density varies according to

$$\rho = 1.225\, e^{\left(\frac{-h}{8400}\right)}$$

where h is height in metres.

V [m/s]	Alt [km]
50	0
53.07	1
102.14	12

MATLAB

```
% Flight condition 1
alt = 0; V = 50; S = 20; CL = 0.5; CD = 0.02;
L1 = 0.5 * 1.225*exp(alt/-8400) * V^2 * S * CL;
D1 = 0.5 * 1.225*exp(alt/-8400) * V^2 * S * CD;

% Flight condition 2
alt = 1000; V = 53.07; S = 20; CL = 0.5; CD = 0.02;
L2 = 0.5 * 1.225*exp(alt/-8400) * V^2 * S * CL;
D2 = 0.5 * 1.225*exp(alt/-8400) * V^2 * S * CD;

% Flight condition 3
alt = 12000; V = 102.14; S = 20; CL = 0.5; CD = 0.02;
L3 = 0.5 * 1.225*exp(alt/-8400) * V^2 * S * CL;
D3 = 0.5 * 1.225*exp(alt/-8400) * V^2 * S * CD;
```

- **Attempt 1**: Naive code
  - Lots of repetition
- (**DRY**) don't repeat yourself!
  - Encapsulate repeated code into a function

Example Functions

Attempt 2

- Cleaner code, but we are still not respecting DRY!
- Also consider the SRP **“Single Responsibility Principle”**
- A function with a single responsibility:
  - Has one clear purpose
  - Performs a single transformation
  - Can be described without using "and“
- Respecting **DRY** and **SRP** leads to better code that is more maintainable and reusable

MATLAB

```
S = 20; CL = 0.5; CD = 0.02; % constant properties

alt = 0; V = 50; % Flight condition 1
[L1,D1] = calculate_forces(alt, V, S, CL, CD);

alt = 1000; V = 53.07; % Flight condition 2
[L2,D2] = calculate_forces(alt, V, S, CL, CD);

alt = 12000; V = 102.14; % Flight condition 3
[L3,D3] = calculate_forces(alt, V, S, CL, CD);

function [L,D] = calculate_forces(alt, v, S, CL, CD)
    % Calculate air density using an exponential atmosphere model
    rho = 1.225 * exp(alt / -8400);  % Simplified model
    q = 0.5 * rho * v^2;  % Dynamic pressure
    % Calculate lift and drag using modular functions
    L = q * S * CL;
    D = q * S * CD;
end
```

Example Functions

Attempt 3

- Respecting **DRY** and **SRP** leads to better code that is more maintainable and reusable
  - Function ‘air_density’ or ‘calcForce’ could easily be used in other code
- Be careful not to overuse the SRP
  - We could also have a function for dynamic pressure…
  - **Complexity Rule**: Hide complexity, don't create it



bristol.

MATLAB

```matlab
% constant properties
S = 20; CL = 0.5; CD = 0.02;
alts = [0, 1000, 12000]; % altitudes for flight conditions
Vs = [50, 53.07, 102.14]; % corresponding velocities

L = zeros(1,3); % Preallocate lift array
D = zeros(1,3); % Preallocate drag array

for i = 1:length(alts)
    rho = air_density(alts(i));
    L(i) = calcForce(rho, Vs(i), S, CL);
    D(i) = calcForce(rho, Vs(i), S, CD);
end

function [rho] = air_density(alt)
    % Calculate air density using an exponential atmosphere model
    rho = 1.225 * exp(alt / -8400);  % Simplified model
end

function [F] = calcForce(rho, v, S, coeff)
    % CALCFORCE Calculate lift or drag force
    F =  0.5 * rho * v^2 * S * coeff;
end
```

Inline or Separate File Functions

- Functions can be written within a script, but they can't be reused in other scripts
- Alternatively, you can write functions in a separate file
  - These can then be used in other scripts
- If in a dedicated file, the filename must match the function name



Flight_conditions.m

```
% constant properties
S = 20; CL = 0.5; CD = 0.02;
alts = [0, 1000, 12000]; % altitudes for flight conditions
Vs = [50, 53.07, 102.14]; % corresponding velocities

L = zeros(1,3); % Preallocate lift array
D = zeros(1,3); % Preallocate drag array

for i = 1:length(alts)
    rho = air_density(alts(i));
    L(i) = calcForce(rho, Vs(i), S, CL);
    D(i) = calcForce(rho, Vs(i), S, CD);
end

function [rho] = air_density(alt)
    % Calculate air density using an exponential atmosphere model
    rho = 1.225 * exp(alt / -8400);  % Simplified model
End
```

calcForce.m

```
function [F] = calcForce(rho, v, S, coeff)
    % CALCFORCE Calculate lift or drag force
    F =  0.5 * rho * v^2 * S * coeff;
end
```

Anonymous Functions

- Simple one-line maths function with a single output?
  - Consider anonymous functions

```
<handle> = @(<inputs>)<expression>;
```

  - It returns a function handle that you can use like a function. e.g.

```
z = @(x,y) x.^2 + y.^2;

z(3,3) % returns 18
```

  - However, they are difficult to reuse, so use them sparingly

This is functionally identical to before!

MATLAB

```
% constant properties
S = 20; CL = 0.5; CD = 0.02;
alts = [0, 1000, 12000]; % altitudes for flight conditions
Vs = [50, 53.07, 102.14]; % corresponding velocities

L = zeros(1,3); % Preallocate lift array
D = zeros(1,3); % Preallocate drag array

air_density = @(alt) 1.225 * exp(alt / -8400);
calcForce = @(rho, v, S, coeff) 0.5 * rho * v^2 * S * coeff;

for i = 1:length(alts)
    rho = air_density(alts(i));
    L(i) = calcForce(rho, Vs(i), S, CL);
    D(i) = calcForce(rho, Vs(i), S, CD);
End
```

University of BRISTOL

MATLAB Tips & Tricks

How to develop fast(er) code

bristol.ac.uk

Vectorisation

- We can take advantage of the knowledge that MATLAB is optimised for coding with vectors, and often avoid unnecessary looping (and get massive performance increases).
- An example is to plot
$$y = \sin(x), \qquad 0 \leq x \leq \pi$$
which requires lots of values of x to get a smooth curve
- **Pass arrays into functions rather than iterating over elements!**
- **If you need to loop, pre-allocating arrays is also important**

For-loop: 305 ms

```
x = 0:1e-6:pi;
for i =1:length(x)
   y(i)= sin(x(i));
end
```

For-loop 2: 62 ms

```
x = 0:1e-6:pi;
y = zeros(size(x));
for i =1:length(x)
   y(i)= sin(x(i));
end
```

Vectorised: 20 ms

```
x = 0:1e-6:pi;
y = sin(x);
```

Timing Your Code

- Use the functions ‘tic’ and ‘toc’ to quickly time sections of your code
  - ‘toc’ outputs the elapsed time to the command window
- Timing a single execution has significant variations
- Average multiple calls

Time individual runs

```
for i = 1:5
  tic;
  x = 0:1e-6:pi;
  y = zeros(size(x));
  for i =1:length(x)
    y(i)= sin(x(i));
  end
  toc;
end
```

→

68.7 ms
68.3 ms
70.2 ms
61.2 ms
61.8 ms

Average multiple runs

```
tic;
for i = 1:100
  x = 0:1e-6:pi;
  y = zeros(size(x));
  for i =1:length(x)
    y(i)= sin(x(i));
  end
end
t = toc/100;
```

→

62.2 ms

Timing Your Code

- Encapsulating your code in a function makes for cleaner source code
- You can also utilise the built-in function ‘timeit’ to time your function. This automatically calls the function multiple times.
- For line-by-line timing, look at ‘**code profiling**’
- Optimising code is good – but use your time wisely
  - Don’t spend 2 hours optimising code you’ll run twice!

MATLAB

```
tic;
for i = 1:100
  [x,y] = testFunc();
end
t = toc/100

timeit(@testFunc)

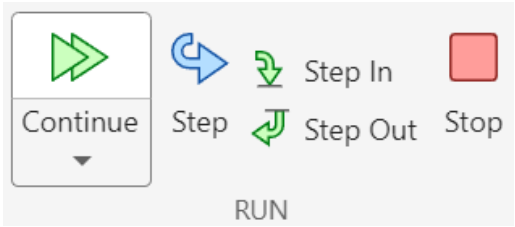
function [x,y] = testFunc()
   x = 0:1e-6:pi;
   y = zeros(size(x));
   for i =1:length(x)
     y(i)= sin(x(i));
   end
end
```

[Debug MATLAB Code Files - MATLAB & Simulink]

Debugging

- The Debugging tools built into the MATLAB IDE are powerful. Use them to your advantage
- By placing “Breakpoints”, you can pause the execution of your code on any line. You can then
  - Explore the value of variables in the workspace
  - Manually run lines in the command window to alter values
  - Step through your code line by line

- When paused at a breakpoint you will see the following options




```
1   n = 50;
2   r = rand(n,1)
3   plot(r)
4
5   m = mean(r);
6   hold on
7   plot([0,n],[m,m])
8   hold off
9   title("Mean of Random Uniform Data")
```

Name	Shortcut	Description
Continue	**F5**	Continue running file until the end or next breakpoint
Step	**F10**	Run the current line of code
Step in	**F11**	Run the current line of code, and, if the line contains a call to another function, step into that function.
Step out	**Shift+F11**	Run the rest of the called function, leave the called function, and then pause.
Stop	**Shift+F5**	End debugging session

Symbolic Math Toolbox Documentation

Symbolic Algebra

- MATLAB can also do symbolic algebra!
- What is the solution to this indefinite integral?

$$\int (2 + 0.05x^2 + \sin(x))\, dx$$

- What is the solution to this definite integral?

$$\int_0^{10} (2 + 0.05x^2 + \sin(x))\, dx$$

- What is the solution of this derivative?

$$\frac{d}{dx}(\sin^2 x\,(x^3 - \cos x\,))$$

MATLAB

```
syms x

y = 2 + (0.05 * x^2) + sin(x); % define symbolic equation
int_y = int(y, x); % calc indefinite integral
pretty(int_y) % display nicely

y = 2 + (0.05 * x^2) + sin(x); % define symbolic equation
int_y = int(y, x,0,10); % calc definite integral
double(int_y); % convert to double


y = sin(x)^2 * (x^3 - cos(x));
d1 = diff(y, x); % first derivative

pretty(simplify(d1)) % display nicely, simplify trig functions

pretty(cos(x)^2 + sin(x)^2)
pretty(simplify(cos(x)^2 + sin(x)^2)) % should simplify to 1
```

University of BRISTOL

SUMMARY

bristol.ac.uk

Summary

- Have shown you how to generate ‘pretty’ figures
- (Re-)Introduced you to functions
- Covered tip-and-tricks including vectorisation

https://i2sc.fintanhealy.co.uk/

Moving Forward

- **No more labs – but your journey is not over!**
  - Attempt the rest of the workbooks; they will save you time when it comes to your coursework!
  - You have until 30th October to submit parameters for the group design task
- **Don’t neglect Python**
  - Being ‘conversational’ in multiple programming languages is a good skill for the 21st century
- There is still more to learn! For example:
  - **object-oriented-programming**
  - **functional programming**
  - Collaborative coding – **version control & git repositories**