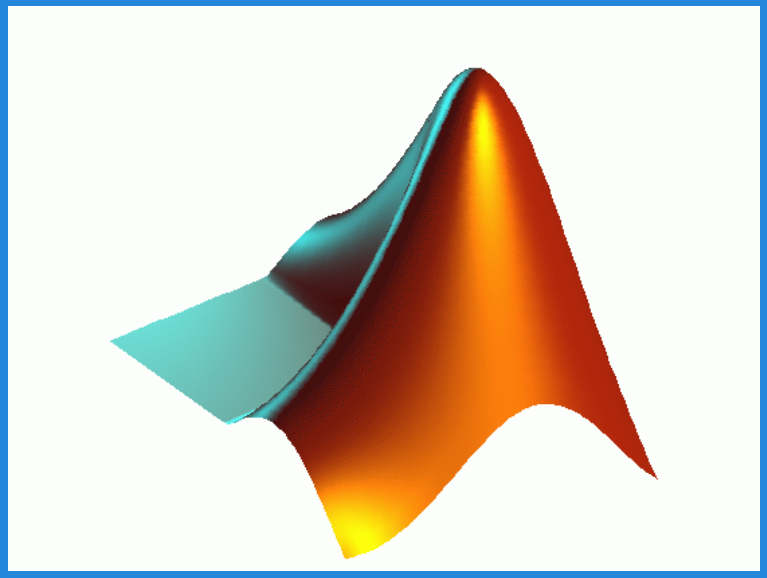

MATLAB

Devon Cormack and James Staley

MATrix LABoratory

- Originally developed in 1970s as a FORTRAN wrapper, later rewritten in C
- Designed for the purpose of high-level numerical computation, visualization, and programming.
- MATLAB is especially useful for these applications because it is implicitly parallel. It will automatically use the maximum number of threads available (# of cores on machine).
- Can perform matrix operations quickly

MATLAB environment and scripts

- MATLAB is written and run from scripts (.m files) in the MATLAB editor.
- The Workspace shows all currently allocated variables.
- Comments are done with a single '%', a double '%%' indicates a Cell within a script that will be run independently of other cells.
- A semi-colon will suppress the output of a line of code. If this is left off every single statement will be printed out.
- Whitespace does not matter but newlines do.

```
%% Basic matlab script
a = 'this string will be printed'

%% this cell will be run only when selected

b = 'this string will not';

%% so will this one

'all hail the semi-colon'
```

Types

- All data in MATLAB are stored by default as doubles. Any arithmetic operator can be applied to any value (except 64 bit integers).
- Values can be specified as floats, and either signed or unsigned 16, 32, or 64 bit integers.
- Strings are held as 1xN matrices of characters
- 'true' and 'false' resolve to 1 and 0

```
a = 'hello there';
b = a(1); %prints h
b = b + 1; %prints 105
b = char(b); %prints i, casting an illegal value to a character results in a blank space

c = 1.2345; %default double
d = single(9.876); % explicit 32-bit float

%prints the same value twice
sixteen = uint16(2^16);
sixteen = sixteen + 1;

%64 bit ints
useless = uint64(2^64);
%illegal statements, arithmetic operators are undefined
% a = useless + 1;
% a = uslesss / 100;

%abs, bitor, xor etc... are possible with 64 bit integers
```

Variables and Expressions

- All variables in MATLAB are polymorphic and stored as matrices.
- To assign a value to a variable, no type needs to be specified. The assignment looks like:

```
a = 7; %This will assign the matrix [7] to the variable a
```

- Matrices can be assigned by using whitespace or commas to indicate columns and semicolons to indicate rows

```
a = [1 2 3; 4 5 6];      %Assigns [ 1 2 3 ; 4 5 6 ] to variable a
```

- Expressions in MATLAB evaluate <,>,<=,>=, and == conditional expressions. They compare the array on the right of the expression to the array on the left of the expression element-by-element and return a boolean matrix

```
if a < b && b >= c
    a = c;
end
```

Common Programming Statements

For and While Loops

```
for i = 1:5
    %do something
end
```

```
while a<b
    %Do something
end
```

If, Else, Else if statements

```
if a < b
    %Do something
elseif a > 2*b
    %Do something
else
    %Do something
end
```

Element-wise Operations

```
a = [1 2 3];
b = [4 5 6];
c = a.*b;  %results in [4 10 18]
```

Functions

- Files must be named "<function_name>.m"
- Nested functions allowed.
- Functions can have multiple return values.
- Handles can be used to pass a parameter to a different workspace.

```
%Function newCounter(init_val) initializes the counter and returns the
%handle to increment the counter

function varargout = makeCounter(init_val)
if nargout < 2
  error('Not enough output arguments')
end
%Set the initial value of the counter
current = init_val;
%Return the handle to the incCounter nested function so that it can be
%called to increment the counter
varargout{1} = @incCounter;
varargout{2} = @getCounter;

    %Nested function incCounter increments the counter
    function incCounter
        current = current + 1;
    end

    %Nested function getCounter returns the counter's value.
    function out = getCounter
        out = current;
    end
end

%To return a handle to a function, use @function where function is the name
%of the function. This creates a dynamic copy of the function
```

Memory

- All variables are stored in the active workspace until manually cleared.
- A function has its own active workspace.
- Workspaces can be saved and reloaded.

```
pack
```

```
save(saved_data.mat)
```

- User has very little control over low-level memory management

Unique Features

- MATLAB is one-indexed

```
a = [1 2 3];
a(0)                    %Throws an error
a(1)                      %Prints out the first element of a
```

- MATLAB can create different types of matrices using commands:

```
a = zeros(7,7)       %Makes a 7x7 array of zeros
a = ones(7,7)          %Makes a 7x7 array of ones
a = eye(7,7)          % Makes a 7x7 identity matrix
```

- MATLAB is commercial software, and is backed by the development team at MathWorks. It also has a very active user database that shares code.
- MATLAB runs in its own window that allows you to see the complete workspace and all of the active variables. Commands are run from the MATLAB command prompt, rather than the Windows Command Prompt or the terminal

Interval Generation

- Easily create intervals between a start and an end value that are uniformly spaced by a designated value.
- A : B : C generates an array beginning with A and ending with C with entries spaced by B.

```
for i = 1:.1:10
    c = 100/i
end

a = 100:-1:1 %array from 100 downto 1
```

Why Use Matlab?

MATLAB is commonly used by:

- Engineers
- Mathematicians
- Biologists
- Physicists
- Computer Scientists
- Good for mathematical modeling, linear algebra, and vectorized operations

Graphing and Data Analysis

- MATLAB is an excellent data analysis tool with a huge library of graphing and analysis functions.

```
%%
a = 1:.1:10;
b = 10:-.1:1;

%plot two labelled graphs in one figure
figure(1)
subplot(2,1,1);
plot(a,b);
xlabel('time')
ylabel('Variable')
title('a plot')

subplot(2,1,2);
plot(a,a);
xlabel('time')
ylabel('Different Variable')
title('a plot')


% make a histogram of the distribution of 1000 random numbers
c = zeros(1,1000);
for i = 1:1000
    c(1,i) = rand;
end
figure(2)
hist(c)
legend('random number distribution')
```