MatLab: a humble primer...

- **Introduction.**
  - Data types.
  - Operators.
  - Useful Functions.

- **Programming.**
  - Scripts.
  - Functions.
Introduction: Data types

- **Matrix**: $M = \text{zeros}(2,3)$
  - arrays: column $M(:,j)$, row $M(i,:)$
  - multidimensional matrix: $MD = \text{zeros}(2,2,2,3)$
- **Strings**: $s = 'embodied intelligence'$
- **Structure**: $ailab = \text{struct}('people',[],'projects',[])$
- **Cell**: $c = \text{cell}(1,3)$;
  $c{1} = \text{zeros}(2); c{2} = 'chunk'; c{3} = \text{ones}(1,3);$
Intro: Matrix

- creating a matrix / array

```matlab
>> A=[]
A =
    []
>> A=[1 2 3; 4 5 6; 7 8 9]
A =
    1   2   3
    4   5   6
    7   8   9
>> A=eye(2)
A =
    1   0
    0   1
>> A=[1 2 3; ones(1,3); 7 8 9]
A =
    1   2   3
    1   1   1
    7   8   9
```

`>> help elmat`
## Intro: Matrix

### Accessing a matrix / array

```matlab
>> A=[1 2 3; 4 5 6; 7 8 9];
>> A(1,2)
an =
   2
>> A(2,:)  
an =
   4   5   6
>> A(:,3)  
an =
   3
   6
   9
>> A([1 3],:)
an =
   1   2   3
   7   8   9
>> A([1 2],[3 2])
an =
   3   2
   6   5
>> ans(2,:)

What is the output?
```

- Index can be logical.
- Index can be a function's output.
- Index can be variables.

```matlab
>> i=3; j=[1 2];
>> A(i,j)
an =
   7   8
```
Intro: Everyday's operators

- "::"

```
>> 1:5
ans =
    1   2   3   4   5
>> 1:2:10
ans =
    1   3   5   7   9
```

- ",," or " "

```
>> 1:5, [1,2]; [1 2]
ans =
    1   2   3   4   5
ans =
    1   2
ans =
    1   2
```

- "'" and ".'"

```
>> c=[1+2i; 3-2i];
>> c'
ans =
    1   2i   3   2i
>> c.'
ans =
    1   2i   3   2i
```

- ";"

```
>> 1:5;
>> p=[1;2;3]
p =
    1
    2
    3
```

© JP Carbajal 05. 2008
Intro: Everyday's operators

- “.” means element wise

```matlab
>> x=[1 2 3]; y=[1 0.5 1/3]; y.*x
ans =
    1    1    1
>> A=[1 1; 1 -1]; X=[1 1; 2 2]; A*X, A.*X
ans =
    3    3
    -1   -1
ans =
    1    1
    2   -2
>> A=[2 1; 0 2]; A^2, A.^2
ans =
    4    4
    0    4
ans =
    4    1
    0    4
```
Intro: Everyday's operators

- logical operators

```
>> xs=dec2bin(33); ys=dec2bin(45);
>> for i=1:length(xs),...
   x(i)=str2num(xs(i)); y(i)=str2num(ys(i));...
end
>>[x;y], x&y
ans =
   1     0     0     0     0     0     1
   1     0     1     1     0     1
ans =
   1     0     1     1     0     1

& (and), | (or), ~ (not), xor, any, all
```
Intro: Everyday's operators

- relational operators

> x=1:6; y=randperm(6);
> [x;y], x<=y

ans =
1  2  3  4  5  6  6
3  2  1  5  4  6
ans =
1  1  0  1  0  1

== (eq), ~= (ne), < (lt),
> (gt), <= (le), >= (ge)
Intro: Everyday's functions

- zeros, ones, eye, diag, blkdiag
- rand, randn, randperm, permute
- exp, sqrt, log, sin, cos,…
- det, inv, eig, norm
- sum, cumsum, diff, mean, std, fft
- reshape, repmat, flipud/lr, circshift
- linspace, logspace, meshgrid
- polyfit, polyval, interp

>> help elfun
>> help datafun
>> help specfun
Intro: Everyday's functions

diag, blkdiag

I need to build the matrix:

\[
\begin{pmatrix}
1 & 1 & 0 \\
0 & 1 & 1 \\
0 & 0 & 1 \\
\end{pmatrix}
\]

\[x + y = a\]

\[
\begin{pmatrix}
0 & 0 & 1 \\
\end{pmatrix}
\]

\[y + z = b\]

\[
\begin{pmatrix}
0 & 0 & 1 \\
\end{pmatrix}
\]

\[z = c\]

what about this one:

\[
\begin{pmatrix}
1 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 \\
0 & 0 & 1 & 1 \\
\end{pmatrix}
\]

Easy dude...

\[A=\text{eye}(3)+\text{diag}([1 \ 1],1)\]

...mmm, easier

\[B=\text{blkdiag}([1 \ 1],[1 \ 1])\]
Intro: Everyday's functions

rand, randperm

100 random numbers in [-1, 1]?
for i=1:100
    x(i)=2 * rand - 1;
end

JA!

Nein!!

x=2 * rand(1,100) - 1;

No loops...when possible

I have to choose the letters of the string x='destiny' at random without repeating...gosh!...

n=length(x);
ind=randperm(n); x(ind)

voila!
Intro: Everyday's functions

inv, “\”

```
3x + y - 5z = 6
x - 2z = 1
-3x + y - z = -8
```

```
>> A = [ 3 1 -5; ...
1 0 -2;-3 1 -1];
>> x = [6 1 -8]';
inv(A) * x
ans =
3
2
1
```

```
>> A(4,:) = ones(1,3);
>> x(4) = 10; y = A \ x
y =
3.5714
4.4034
1.7563
```

```
>> [A*y round(A*y)]
ans =
6.3361   6.0000
0.0588    0
-8.0672  -8.0000
9.7311   10.0000
```

“\” tries to minimize norm(A*x – y) with max number of non-zero elements. To minimize use pinv(A).
Intro: Everyday's functions

sum, cumsum, diff

```matlab
>> x=[ones(5,1) ([1:5]').^2], sx=sum(x), ...
cx=cumsum(x), dx=diff(cx)
sx =
    5   55
cx =
     1   1
     2   5
     3  14
     4  30
     5  55
dx =
     1   4
     1   9
     1  16
     1  25
```
Intro: Everyday's functions

flipud/lr, circshift, reshape, repmat

```matlab
>> x=[1 2 3]; fliplr(x), fliud(x')
ans =
     3  2  1
ans =
     3
     2
     1
>> x=eye(3); circshift(x,[1 0]), circshift(x,[0 1])
ans =
     0  0  1
     1  0  0
     0  1  0
ans =
     0  1  0
     0  0  1
     1  0  0
```
## Intro: Everyday's functions

### flipud/lr, circshift, reshape, repmat

```matlab
>> x=[1 1 1 2 2 2 3 3 3]; A=reshape(x,3,3)
A =
   1   2   3
   1   2   3
   1   2   3
>> repmat(A,2), repmat(A,1,2)
ans =
   1   2   3   1   2   3
   1   2   3   1   2   3
   1   2   3   1   2   3
   1   2   3   1   2   3
   1   2   3   1   2   3
   1   2   3   1   2   3
>> x2
x2
x2
x1
```

```
ans =
   1   2   3   1   2   3
   1   2   3   1   2   3
   1   2   3   1   2   3
   1   2   3   1   2   3
   1   2   3   1   2   3
   1   2   3   1   2   3
```
Intro: Everyday's functions

linspace, meshgrid

```
>> x = linspace(-1,1,7)
x =
   -1  -0.667  -0.333   0   0.333   0.667   1
>> x = 1:3; y=10:15; [X Y] = meshgrid(x,y)
X =
   1     2     3
   1     2     3
   1     2     3
   1     2     3
   1     2     3
   1     2     3
Y =
   10    10    10
   11    11    11
   12    12    12
   13    13    13
   14    14    14
   15    15    15
```
Intro: Everyday's functions

polyval

```matlab
>> p=[-3 2 1]; x=linspace(0,1.5,10); y=polyval(p,x)
y =
   1 1.25 1.33 1.25 1.0 0.58 0 -0.75 -1.67 -2.75
>> plot(x,y,'r-o'); legend('y=-3x^2+2x+1'); ...
axis([0 2 -3 1.5])
```
Intro: Everyday's functions

polyfit

```
>> noise=0.1*randn(1,10), yn=y+noise;
>> pf=polyfit(x,yn,2)
pf =
    3.0034  2.0332  0.9560
>>plot(x,y,'r-',x,yn,'r*',x,polyval(pf,x),'b-')
>>legen('original data','noisy data','fit')
```
Intro: Everyday's functions

polyfit

```matlab
>> x=0:0.1:2*pi; [X Y]=meshgrid(x,x); z=cos(X).*sin(Y);
>> surf(X,Y,z)
>> xlabel('x'); ylabel('y'); zlabel('z')
```
MatLab: a humble primer...

- Introduction.
  - Data types.
  - Operators.
  - Useful Functions.
- Programming.
  - Scripts.
  - Functions.
A script file is an external file that contains a sequence of statements. You can call a script by typing the filename.

Scripts are the simplest kind of M-file. They are useful for automating blocks of MATLAB commands.

Scripts share the base workspace with your interactive MATLAB session and with other scripts. They operate on existing data in the workspace, or they can create new data. Any variables that scripts create remain in the workspace after the script finishes.

Write a script that generates strings numbered in sequence: file1, file2, file3,...

Hints?
- `num2str`, `[string1 string2]`
Programming: M - file structure

function f = fact(n)
  % Compute a factorial value.
  % FACT(N) returns the factorial of N,
  % usually denoted by N!

  % Put simply, FACT(N) is PROD(1:N).
  f = prod(1:n);

Function definition line
H1 line
Help text
Comment
Function body

Function definition line
Defines the function name, and the number and order of input and output arguments

H1 line
A one line summary description of the program, displayed when you request help on the function

Help text
A more detailed description of the program, displayed together with the H1 line when you request help on the function

Function body
Program code that performs the actual computations and assigns values to any output arguments

Comments
Text in the body of the program that explains the internal workings of the program
• The first function in any M-file is called the **primary function**. Following the primary function can be any number of **subfunctions**, which can serve as subroutines to the primary function.

• You invoke this function using the **name of the M-file** in which it is defined. Under most circumstances, the primary function is the only function in an M-file that you can call from the command line or from another M-file function.

• It is customary to **give the primary function the same name as the M-file** in which it resides. If the function name differs from the filename, then you must use the filename to invoke the function.

---

**average.m:**

```matlab
function y = average(x)
% AVERAGE Mean of vector elements.
y = sum(x)/length(x); % Actual computation
```

```
>> average([12 60 42])
an =
   38
```
Programming: Primary function

- Write a function that takes a matrix and displays a message that says how many negative elements are in the matrix.

- Hints?
  - find or relational operator >0
  - length or sum
  - num2str
  - disp
Programming: Subfunction

- M-files can contain code for more than one function. Additional functions within the file are called subfunctions, and these are only visible to the primary function or to other subfunctions in the same file.
- Each subfunction begins with its own function definition line. The functions immediately follow each other. The various subfunctions can occur in any order, as long as the primary function appears first.
- Subfunctions cannot access variables used by other subfunctions, even within the same M-file, or variables used by the primary function of that M-file, unless you declare them as global within the pertinent functions, or pass them as arguments.
newstas.m:

function [avg, med] = newstats(u) % Primary function
% NEWSTATS Find mean and median with internal functions.
n = length(u);
avg = mean(u, n);
med = median(u, n);

function a = mean(v, n) % Subfunction
% Calculate average.
a = sum(v)/n;

function m = median(v, n) % Subfunction
% Calculate median.
w = sort(v);
if rem(n, 2) == 1
    m = w((n+1) / 2);
else
    m = (w(n/2) + w(n/2+1)) / 2;
end
Programming: Nested function

• A nested function is a function inside another function.
• To write a nested function, simply define one function within the body of another function in an M-file. You must always terminate a nested function with an end statement.

```
function x = A(p1, p2)
...
  function y = B(p3)
  ...
    function z = C(p4)
    ...
      end
  end
end
function w = D(p5)
...
  end
...
end
```
Programming: Nested function

- You can call a nested function: From the level immediately above it. From a function nested at the same level within the same parent function. From a function at any lower level. You can also call a subfunction from any nested function in the same M-file.

    function A(x, y)                   % Primary function
    B(x, y);
    D(y);
    function B(x, y)                  % Nested in A
    C(x);
    D(y);
    function C(x)                     % Nested in B
    D(x);
    end
    end
    function D(x)                     % Nested in A
    E(x);
    function E(x)                     % Nested in D
    ...
    end
    end

- Function A can call B or D, but not C or E.
- Function B can call D, and D can call B.
- Function C can call B or D, but not E.
Programming: Function handle

• A function handle is a callable association to a function. It contains an association to that function that enables you to invoke the function regardless of where you call it from. This means that, even if you are outside the normal scope of a function, you can still call it if you use its handle.
• With function handles, you can pass a function to another function.

```plaintext
a = 0; b = 5;
quad(@log, a, b)
ans =
   3.0472
quad(@sin, a, b)
ans =
   0.7163
quad(@humps, a, b)
ans =
  12.3566
```
Programming: Tips

• **Planning the Program**: Take the problem you are trying to solve and break it down into a series of smaller tasks. Implement each task as a separate function, each having a single purpose.

• **Using Pseudo-Code**: Write the initial draft of your program in a structured format using your own natural language. This pseudo-code is easier to think through than using a formal programming language.

• **Selecting the Right Data Structures**: Look at what classes and data structures are available to you and determine which of those best fit your needs in storing and passing your data.

• Descriptive function and variable names makes your code easier to understand.

• Precede each subfunction with a block of help text describing what that subfunction does.

• **Be sure to document your programs well** to make it easier for you or someone else to maintain them. Explaining each major section and any smaller segments of code that are not obvious.

%---------------------------------------------
% This function computes the ... <and so on>
%---------------------------------------------
Programming: Tips

Coding in Steps
• Do not try to write the entire program all at once. Write a portion of it, and then test that piece out. When you have that part working, then write the next piece, and so on.

Making Modifications in Steps
• When making modifications to a working program, do not make widespread changes all at one time. It's better to make a few small changes, test and debug, make a few more changes, and so on.

Functions with One Calling Function
• If you have a function that is called by only one other function, put it in the same M-file as the calling function, making it a subfunction.
MatLab: a humble primer...

Don't miss the next tutorial:

Dynamical Systems with Matlab