1 Introduction to MATLAB

- MATLAB (*short for Matrix Laboratory*) is a package designed for both practical computations and theoretical investigations of numerical methods and computation.
- MATLAB is actually based on a complex suite of C programs that implement certain "primitive" subroutine operations, plus an interface driver routine which converts input "English-like" and mathematical expressions into the correct subroutine calls.

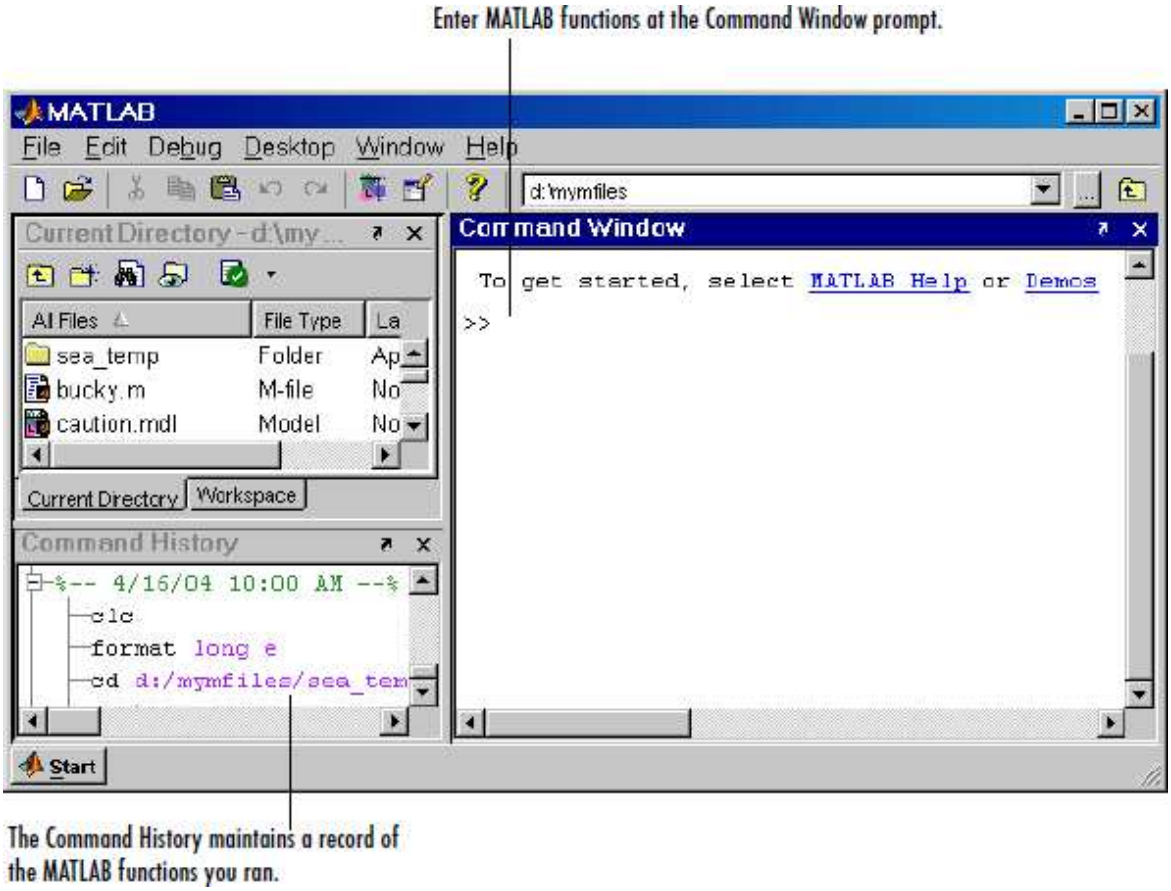


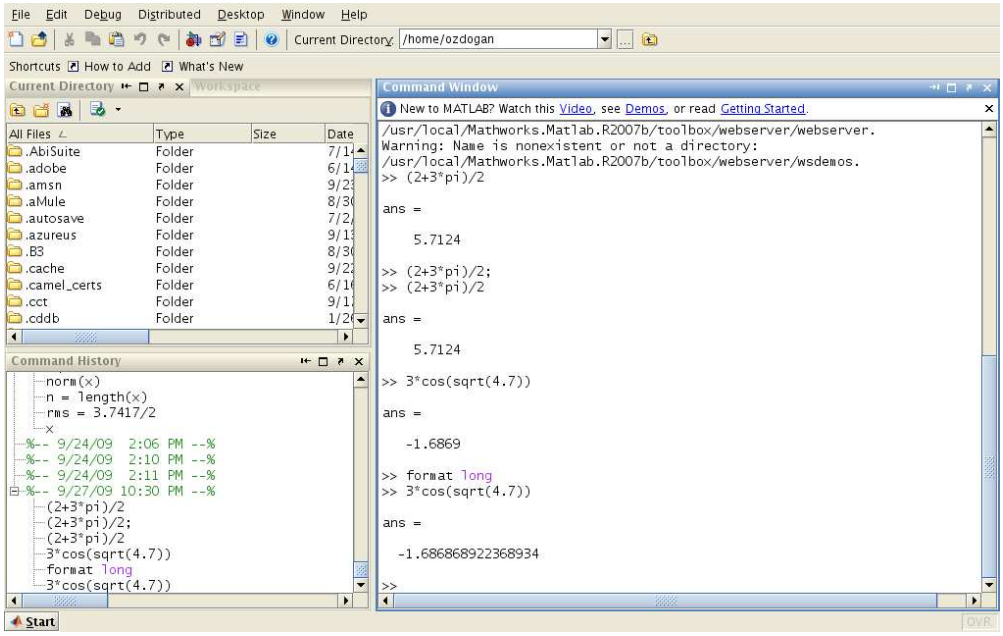
Figure 1: MATLAB Desktop.

- Most MATLAB toolbox commands and functions are really just text files containing sequences of other MATLAB commands and functions. Such files are commonly called **m-files**.

- The commands and functions stored in an m-file are then executed whenever the user appropriately enters the corresponding file name.
- Enter the following commands in the MATLAB Command Window prompt

```
>>(2+3*pi)/2
>>3*cos(sqrt(4.7))
>>format long
>>3*cos(sqrt(4.7))
```

You should have a dialogue as in the following figure



1.1 Help

- Information about specific MATLAB commands may obtained with the **help** command.
  - Simply typing help by itself actually produces a listing of the available MATLAB "toolboxes" (groupings of related commands and functions).

– If you have any question about MATLAB commands, **ask to MATLAB first** as

```
>>help
>>help cos
>>help sqrt
>>help format
>>help det
>>help sym/det.m
```

- Give the command **help chop** and examine the output. Next, give, in sequence, the commands

```
chop( 27.321592 , 5 )
chop( 27.321592 , 4 )
chop( 27.321592 , 3 )
```

What is the use of **chop** (built-in) function. Are the answers what you expected?

1.2 Assignment Statements

- Enter the following commands

```
>>a=3-floor(exp(2.9)) %What is floor? Try "help floor"
                      %on the prompt
>>b=sin(a); %What is the function of ";" at the end?
>>2*b-2
```

1.3 Relational Operators

== Equal to
∼= Not equal to
< Less than
> Greater than
<= Less than or equal to
>= Greater than or equal to
Logical Operators
∼ Not (complement)
& And (True if both operands true)
| Or (True if either/both operands true)

Boolean Values
1 True
0 false

1.4 Loops & Conditionals

- Example; type in the MATLAB command window prompt

```
for k=1:100
x=sqrt(k);
if ((k>10)&(x-floor(x)==0))
break
end
end
k
```

- Example;

```
n=10; k=0;
while k<=n
x=k/3;
disp([x x-2 x-3])
k=k+1;
end
```

Try

```
>>help for
>>help break
>>help while
>>help disp
```

1.5 Functions

- NOT built-in functions. User-defined functions.
- An efficient way to construct programs is to use user-defined functions, saved as m-files.
- These programs allow the user to specify the input and output parameters.
- They are easily called as subroutines in other programs.

- Construct an m-file in the m-file Editor/Debugger (by selecting **New** under **File** menu).
- **Example:** Place the function $f(x) = 1 + x - x^2/4$ in the m-file fun.m.
  - In the Editor/Debugger:
    **********Write into m-file**********

```
function y=fun(x)
y=1+x-x.^2/4; %(Why . ?)
```

    ****

  - Once this function is saved as an m-file named fun.m, try the following commands;

```
>>fun(3)
>>cos(fun(3))
>>feval('fun',4)
```

- Functions can be defined recursively!
- Follow the above procedure for the function below and save under the name of fact.m,

```
function y = fact(n)
y = 1;
if n > 1
y = n*fact(n-1);
end
```

```
>>fact(333)
```

1.6 Matrices & Arrays

- All variables in MATLAB are treated as matrices.
- Matrices can be entered directly or can be generated by other functions.

```
>>help zeros
>>help ones
>>help tan
```

```
>>A=[1 2 3; 4 5 6; 7 8 9] %direct entering
>>Z=zeros(3,5) %(a 3 x 5 matrix of zeroes)
>>X=ones(3,5)  %(a 3 x 5 matrix of ones)
>>Y=0:0.5:2
>>cos(Y)
>>A(2,3)

>>A(1:2,2:3)
>>A([1 3],[1 3])  %(another submatrix)
>>A(2,2)=tan(7.8)
>>B=[1 2;3 4]
>>C=B';
>>C %(C is the transpose of B)
>>3*(B*C)^3  %3(BC)3
```

- IMPORTANT: For any square matrix x (as the matrix A above), compare the difference between the results of
  $y = 1 + x - x.^2/4$
  and
  $y = 1 + x - x^2/4$
  by modifying your previous fun.m.

- Array Operations.

```
>>A=[1 2; 3 4]
>>A^2 %(matrix power A2)
>>A.^2 %(squares each entry of A)
>>cos(A./2) %cos (aij/2)
```

1.7 Graphics

- 2- and 3-D plots of curves and surfaces.

- Use plot for 2-D functions.

- For $y = cosx$ and $y = cos^2x$ over $[0, \pi]$:

```
>>x=0:0.1:pi; %(step-size = 0.1)
>>y=cos(x);
>>z=cos(x).^2;
>>plot(x,y,x,z,'o') %(o's at (x_k, z_k))
```

- Function plot. fplot('name', [a,b],n) plots function **name** with **n** points (default 25) in interval of **[a, b]**.

```
>>fplot('tanh',[-2,2]) %(tanh x, x  [-2, 2])
```

- plot and plot3 for parametric curves in 2- and 3-D space.
  - Ellipse $c(t) = (2cost, 3sint), t \in [0, 2\pi]$:

```
>>t=0:0.2:2*pi;
>>plot(2*cos(t),3*sin(t))
```

  - Curve $c(t) = (2cost, t^2, 1/t), t \in [0.1, 4\pi]$:

```
>>t=0.1:0.1:4*pi;
>>plot3(2*cos(t),t.^2,1./t)
```

- 3-D surface plots, specify a rectangular subset of the domain of a function with *meshgrid*, *mesh* or **surf**.

```
>>x=-pi:0.1:pi;
>>y=x;
>>[x,y]=meshgrid(x,y);
>>z=sin(cos(x+y));
>>mesh(z)

>>help plot
>>help fplot
>>help plot3
>>help mesh
>>help meshgrid
```

1.8 Numeric vs Symbolic Computation

- One drawback of MATLAB is that, by and large, it can only perform **numeric computation**, i.e. every MATLAB command causes a number of floating point computations to be made and the resulting number(s) displayed.

- Another area of computation that is beginning to come into its own is **symbolic computation**, which involves the manipulation of algebraic expressions, and not just numbers.

- The difference between these two types of computation is that, for example, a *numeric* computation package will only be able to tell you that

$$\int_0^1 xe^{-x}dx = 0.26424\ldots$$

while a *symbolic* computation program (i.e., Mathematica, MAPLE) will also tell you that

$$\int_0^1 xe^{-x}dx = 1 - 2e^{-1}$$

- Solving the integral given above both numerically and symbolically by using MATLAB.

```
>> syms x
>> int(x*exp(-x),0,1)
ans = -2*exp(-1)+1
>> eval(int(x*exp(-x),0,1))
ans =  0.2642
>> int(x*exp(-x))
ans = -x*exp(-x)-exp(-x)
```