Matlab II

Acknowledgement: many slides in this lecture were downloaded from various sources in the internet

Grunläggande Matlab operationer

```
>> % This is a comment, it starts with a “%”
>> y = 5*3 + 2^2;        % simple arithmetic
>> x = [1 2 4 5 6];      % create the vector “x”
>> x1 = x.^2;            % square each element in x
>> E = sum(abs(x).^2);   % Calculate signal energy
>> P = E/length(x);      % Calculate a signal power
>> x2 = x(1:3);          % Select first 3 elements in x
>> z = 1+i;              % Create a complex number
>> a = real(z);          % Pick off real part
>> b = imag(z);          % Pick off imaginary part
>> plot(x);              % Plot the vector as a signal
>> t = 0:0.1:100;        % Generate sampled time
>> x3=exp(-t).*cos(t);   % Generate a discrete signal
>> plot(t, x3, ‘x’);     % Plot points
```

Andra Matlab programmering strukturer

Loops

```
for i=1:100
    sum = sum+i;
end
```

Goes round the for loop 100 times, starting at i=1 and finishing at i=100

```
i=1;
while i<=100
    sum = sum+i;
    i = i+1;
end
```

Similar, but uses a while loop instead of a for loop

Decisions

```
if i==5
    a = i*2;
else
    a = i*4;
end
```

Executes whichever branch is appropriate depending on test

```
switch i
case 5
    a = i*2;
otherwise
    a = i*4;
end
```

Similar, but uses a switch

Matlab as a calculator

MATLAB can be used as a ‘clever’ calculator

- This has very limited value in engineering

Real value of MATLAB is in programming

- Want to store a set of instructions
- Want to run these instructions sequentially
- Want the ability to input data and output results
- Want to be able to plot results
- Want to be able to ‘make decisions’

Example revisited

$$\boxed{y = \sum_{i=1}^{n} \frac{1}{\sqrt{i}} = \frac{1}{\sqrt{1}} + \frac{1}{\sqrt{2}} + \frac{1}{\sqrt{3}} + \ldots}$$

Can do using MATLAB as a calculator

```
>> x = 1:10;
>> term = 1./sqrt(x);
>> y = sum(term);
```

Far easier to write as an M-file

How to write an m-file

File → New → m-file

Takes you into the file editor

Enter lines of code (nothing happens)

Save file (we will call ours L2Demo.m)

Exit file

Run file

Edit (*ie* modify) file if necessary

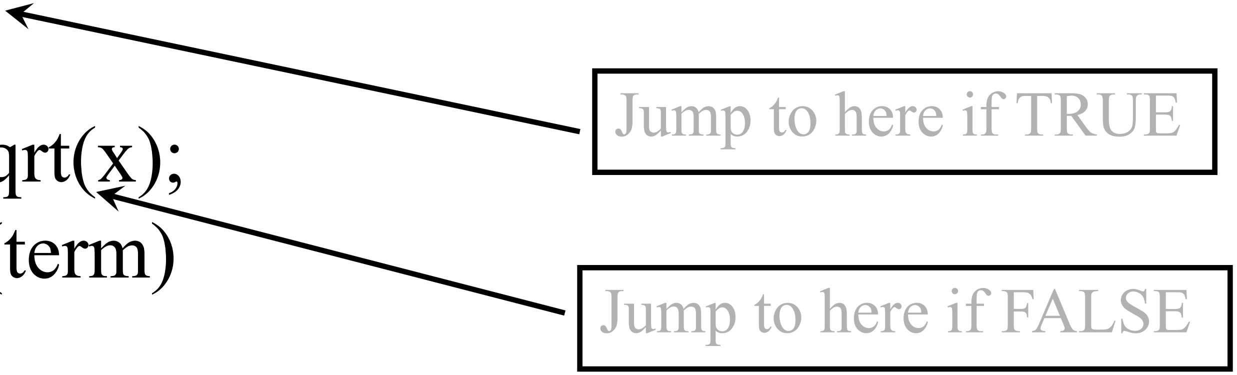
L2Demo version 1

```
n = input( ‘Enter the upper limit:  ‘);
x = 1:n;        % Matlab is case sensitive
term = sqrt(x);
y = sum(term)
```

What happens if n < 1 ?

L2Demo version 2

```
n = input( ‘Enter the upper limit:  ‘);
if n < 1
        disp ( ‘Your answer is meaningless!’ )
end
x = 1:n;
term = sqrt(x);
y = sum(term)
```



Jump to here if TRUE

Jump to here if FALSE

Decision making in Matlab

For ‘simple’ decisions?
IF … END (as in last example)
More complex decisions?
IF … ELSEIF … ELSE ... END

Example: Real roots of a quadratic equation

L3Demo: roots of $ax^2 + bx + c = 0$

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Roots set by discriminant

$\Delta < 0$ (no real roots)

$\Delta = 0$ (one real root)

$\Delta > 0$ (two real roots)

$$\Delta = b^2 - 4ac$$

MATLAB needs to make decisions (based on Δ)

L3Demo: roots of $ax^2 + bx + c = 0$

One possible m-file

```
Read in values of a, b, c
Calculate $\Delta$
IF $\Delta < 0$
Print message ‘ No real roots’→ Go END
ELSEIF $\Delta = 0$
Print message ‘One real root’→ Go END
ELSE
Print message ‘Two real roots’
END
```

```
%==================================================================
%   Demonstration of an m-file
%   Calculate the real roots of a quadratic equation
%==================================================================
```

Header

```
clear all;      % clear all variables
clc;            % clear screen

coeffts = input('Enter values for a,b,c (as a vector):  ');    % Read in equation coefficients
a = coeffts(1);
b = coeffts(2);
c = coeffts(3);
```

Initialisation

```
delta = b^2 - 4*a*c;    % Calculate discriminant
```

Calculate Δ

```
% Calculate number (and value) of real roots

if delta < 0
   fprintf('\nEquation has no real roots:\n\n')
   disp(['discriminant = ', num2str(delta)])
elseif delta == 0
   fprintf('\nEquation has one real root:\n')
   xone = -b/(2*a)
else
   fprintf('\nEquation has two real roots:\n')
   x(1) = (-b + sqrt(delta))/(2*a);
   x(2) = (-b – sqrt(delta))/(2*a);
   fprintf('\n First root = %10.2e\n\t Second root = %10.2f', x(1),x(2))
end
```

Make decisions
based on value of Δ

Flow Control

- if
- for
- while
- break
- ….

Operators (relational, logical)

- == Equal to
- ~= Not equal to
- < Strictly smaller
- > Strictly greater
- <= Smaller than or equal to
- >= Greater than equal to
- & And operator
- | Or operator

Control Structures

- If Statement Syntax

```
if (Condition_1)
        Matlab Commands
elseif (Condition_2)
        Matlab Commands
elseif (Condition_3)
        Matlab Commands
else
        Matlab Commands
end
```

Some Dummy Examples

```
if ((a>3) & (b==5))
     Some Matlab Commands;
end
```

```
if (a<3)
     Some Matlab Commands;
elseif (b~=5)
     Some Matlab Commands;
end
```

```
if (a<3)
     Some Matlab Commands;
else
     Some Matlab Commands;
end
```

Control Structures

- For loop syntax

```
for i=Index_Array
   Matlab Commands
end
```

Some Dummy Examples

```
for i=1:100
      Some Matlab Commands;
end
```

```
for j=1:3:200
      Some Matlab Commands;
end
```

```
for m=13:-0.2:-21
      Some Matlab Commands;
end
```

```
for k=[0.1 0.3 -13 12 7 -9.3]
      Some Matlab Commands;
end
```

Control Structures

- While Loop Syntax

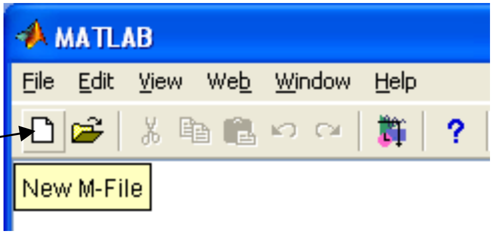
```
while (condition)
   Matlab Commands
end
```

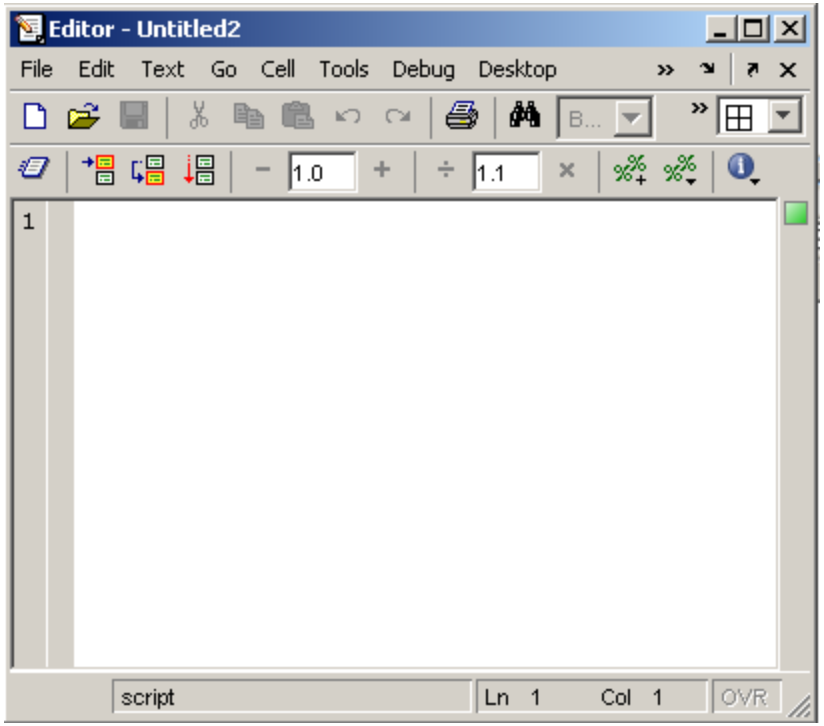
Dummy Example

```
while  ((a>3) & (b==5))
     Some Matlab Commands;
end
```

Use of M-File

Click to create a new M-File

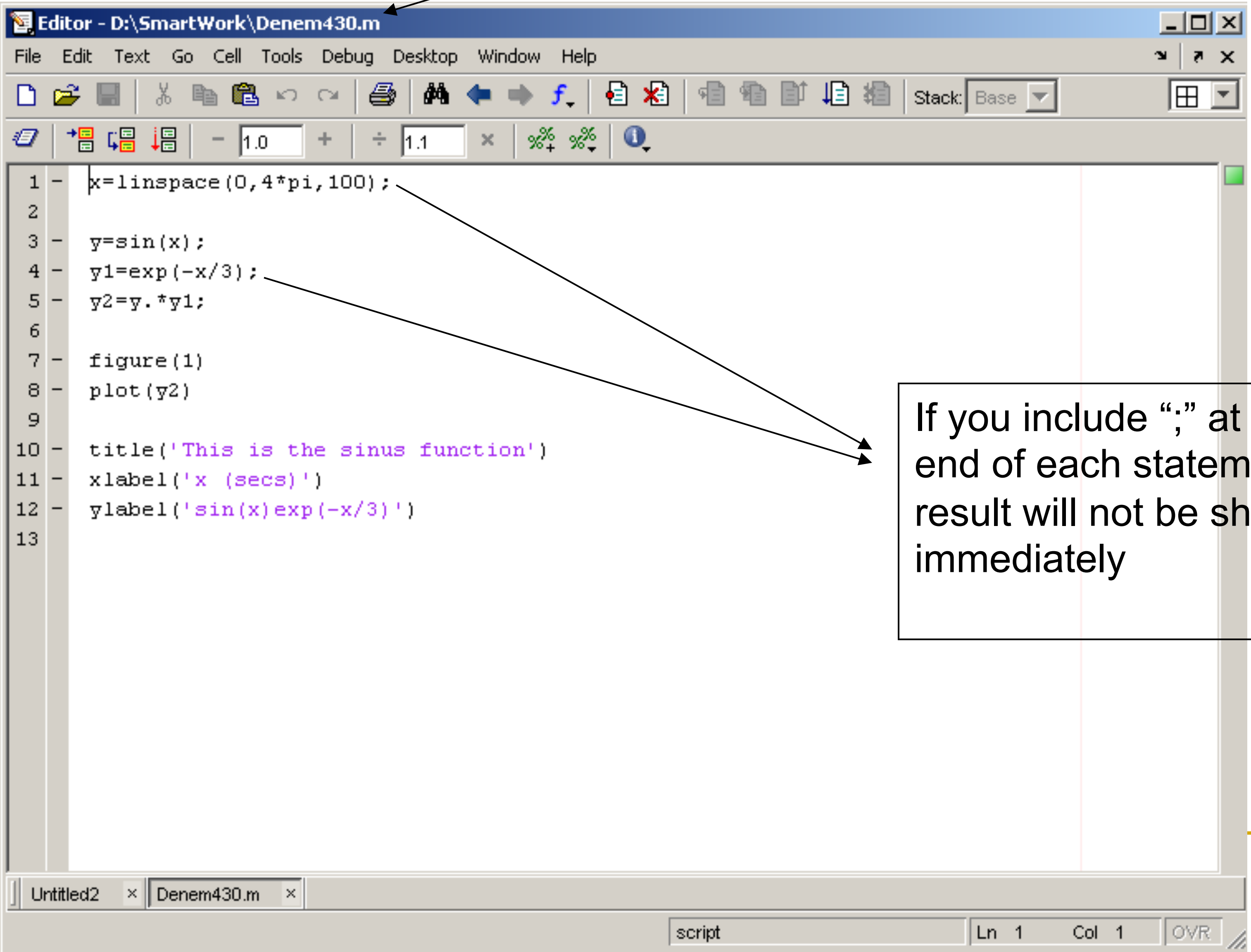




- Extension “.m”
- A text file containing script or function or program to run

Use of M-File

Save file as *Denem430*.m



Editor - D:\SmartWork\Denem430.m
File Edit Text Go Cell Tools Debug Desktop Window Help
Stack: Base
1.0
1.1
Untitled2
Denem430.m
script
Ln 1
Col 1
OVR

```
x=linspace(0,4*pi,100);

y=sin(x);
y1=exp(-x/3);
y2=y.*y1;

figure(1)
plot(y2)

title('This is the sinus function')
xlabel('x (secs)')
ylabel('sin(x)exp(-x/3)')

```

If you include ";" at the end of each statement, result will not be shown immediately

Writing User Defined Functions

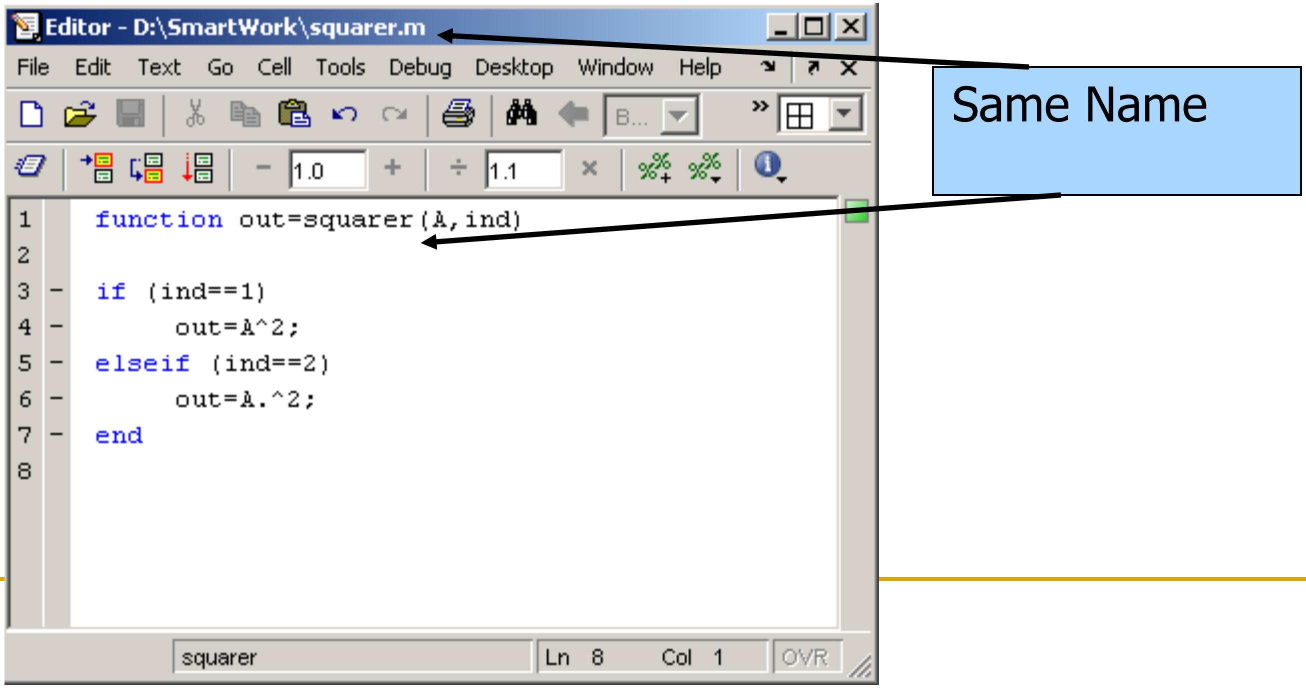
- Functions are m-files which can be executed by specifying some inputs and supply some desired outputs.
- The code telling the Matlab that an m-file is actually a function is

```
function out1=functionname(in1)
function out1=functionname(in1,in2,in3)
function [out1,out2]=functionname(in1,in2)
```

- You should write this command at the beginning of the m-file and you should save the m-file with a file name same as the function name

Writing User Defined Functions

- Examples
  - Write a function : out=squarer (A, ind)
    - Which takes the square of the input matrix if the input indicator is equal to 1
    - And takes the element by element square of the input matrix if the input indicator is equal to 2

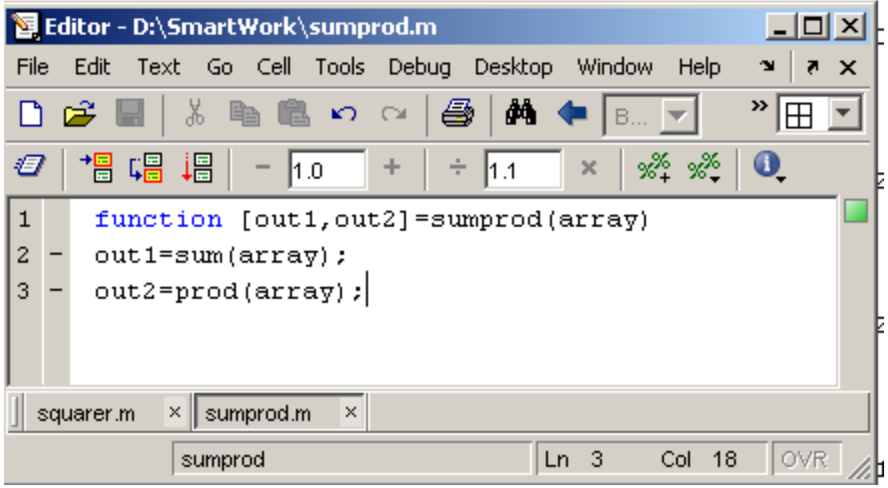

Editor - D:\SmartWork\squarer.m
File Edit Text Go Cell Tools Debug Desktop Window Help
B...
1.0
1.1
squarer
Ln 8 Col 1
OVR

```
function out=squarer(A,ind)

if (ind==1)
       out=A^2;
elseif (ind==2)
       out=A.^2;
end

```

Same Name

Writing User Defined Functions

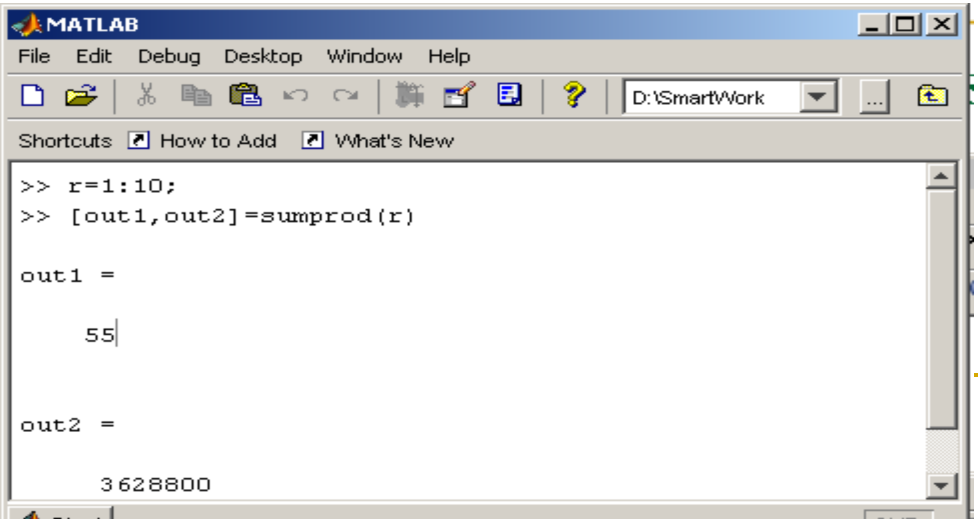
- Another function which takes an input array and returns the sum and product of its elements as outputs



Editor - D:\SmartWork\sumprod.m
File Edit Text Go Cell Tools Debug Desktop Window Help
1.0
1.1
squarer.m
sumprod.m
sumprod
Ln 3 Col 18
OVR

```
function [out1,out2]=sumprod(array)
out1=sum(array);
out2=prod(array);
```

- The function sumprod(.) can be called from command window or an m-file as



MATLAB
File Edit Debug Desktop Window Help
D:\SmartWork
Shortcuts How to Add What's New

```
>> r=1:10;
>> [out1,out2]=sumprod(r)

out1 =

    55


out2 =

     3628800
```

Notes:

- “%” is the neglect sign for Matlab (equaivalent of “//” in C). Anything after it on the same line is neglected by Matlab compiler.
- Sometimes slowing down the execution is done deliberately for observation purposes. You can use the command “pause” for this purpose

```
pause %wait until any key
pause(3) %wait 3 seconds
```

Useful Commands

- The two commands used most by Matlab users are

```
>>help functionname
```

```
>>lookfor keyword
```

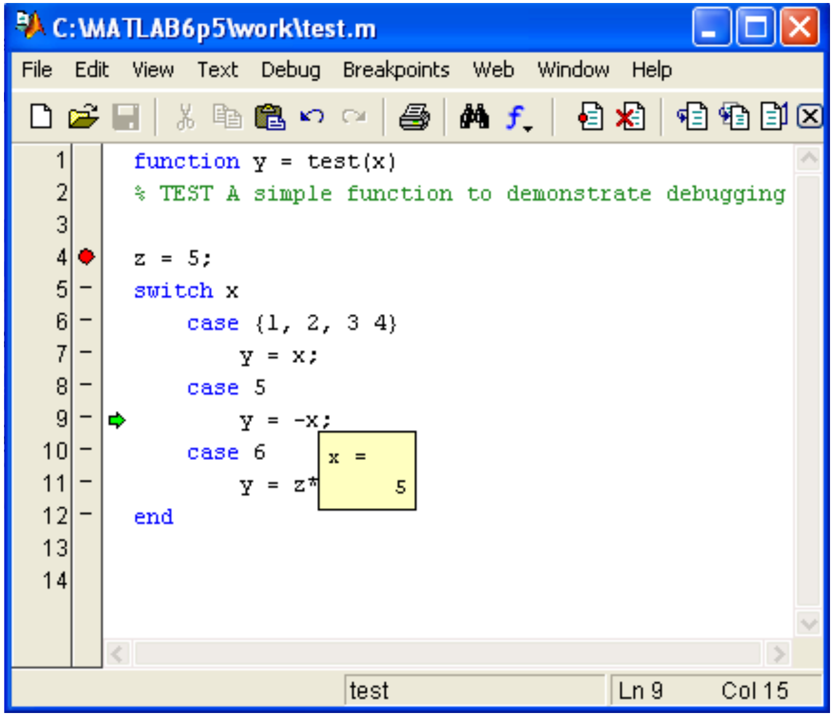
Matlab Debugger

Because Matlab is an interpreted language, there is no compile type syntax checking and the likelihood of a run-time error is higher

Run-time debugging can help

Use the debug and breakpoints pull-down menus to determine where to stop program and inspect variables

Step over lines/step into functions to evaluate what happens



```
function y = test(x)
% TEST A simple function to demonstrate debugging

z = 5;
switch x
    case {1, 2, 3 4}
        y = x;
    case 5
        y = -x;
    case 6
        y = z*
end
```

```
x =
      5
```

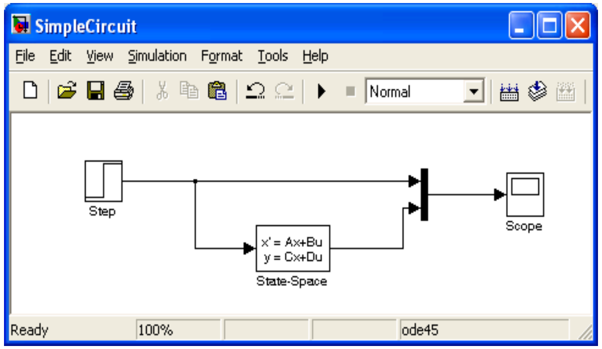
Simulink

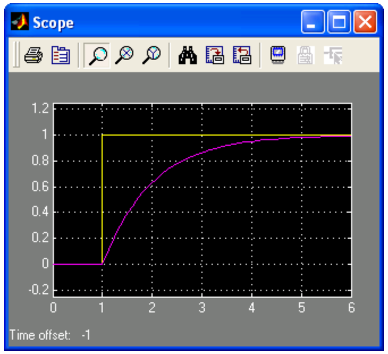
Simulink is a graphical, “drag and drop” environment for building simple and complex signal and system dynamic simulations.

It allows users to concentrate on the structure of the problem, rather than having to worry (too much) about a programming language.

The parameters of each signal and system block is configured by the user (right click on block)

Signals and systems are simulated over a particular time.

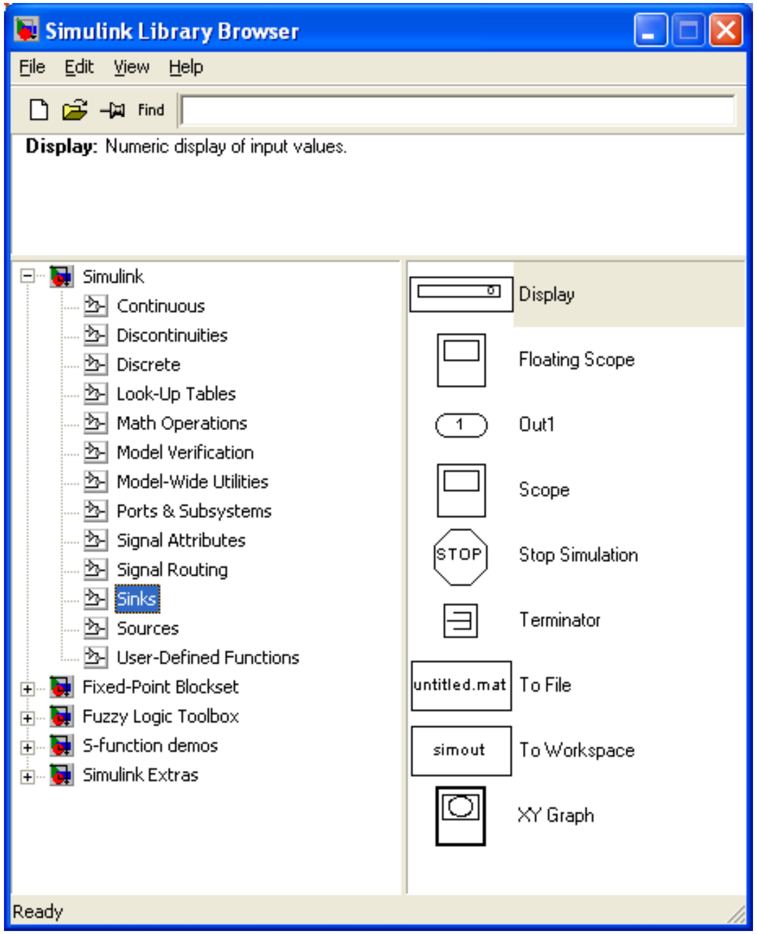




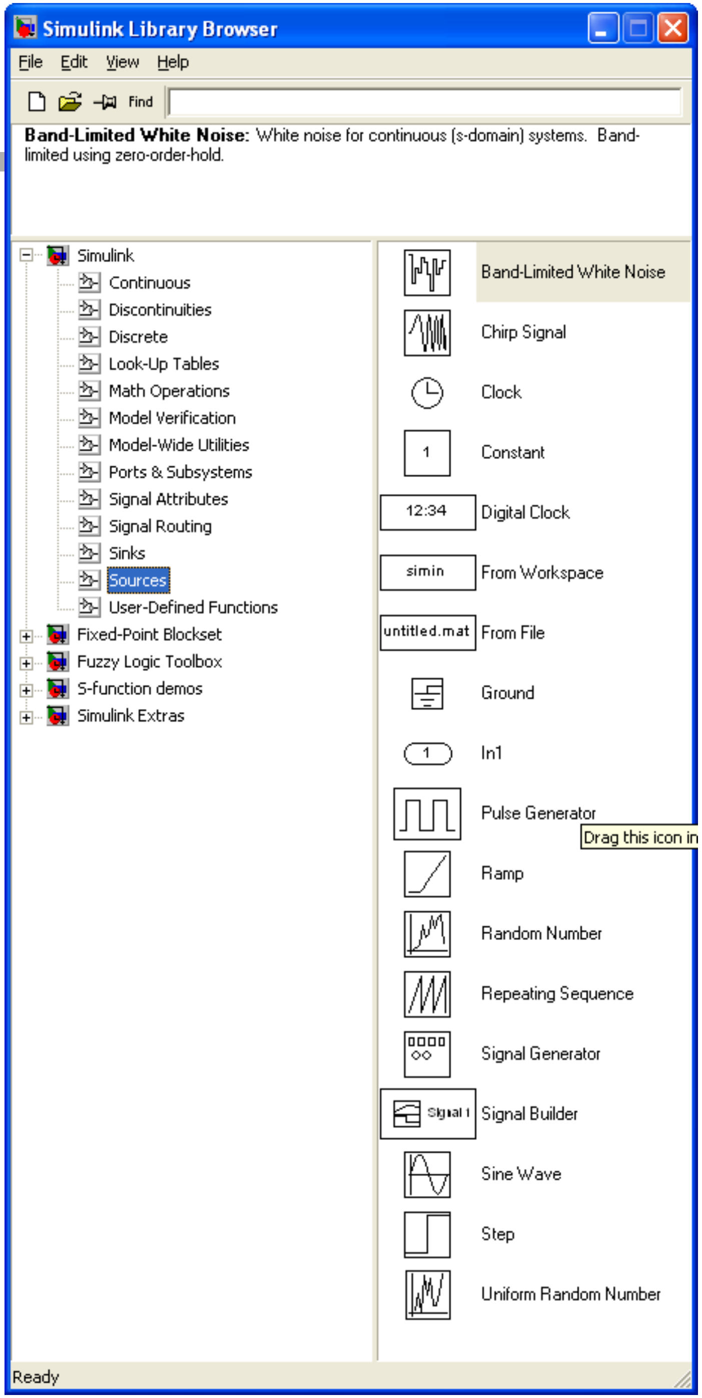
Simulink

Two main libraries for manipulating signals in Simulink:

- **Sources**: generate a signal
- **Sink**: display, read or store a signal



Display: Numeric display of input values.



Band-Limited White Noise: White noise for continuous (s-domain) systems. Band-limited using zero-order-hold.

Simulink

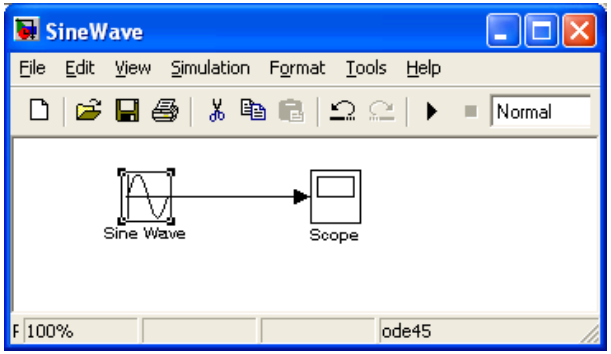
Copy "sine wave" source and "scope" sink onto a new Simulink work space and connect.

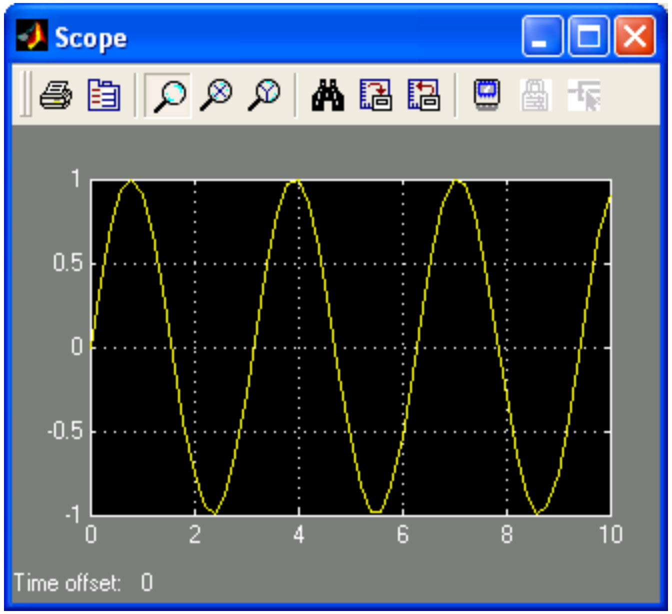
Set sine wave parameters modify to 2 rad/sec

Run the simulation:

Simulation - Start

Open the scope and leave open while you change parameters (sin or simulation parameters) and re-run





Bra referens…

http://web.cecs.pdx.edu/~mperkows/CLASS_479/MATLAB/matlab1.pdf

http://web.cecs.pdx.edu/~mperkows/CLASS_479/MATLAB/matlab2.pdf

http://web.cecs.pdx.edu/~mperkows/CLASS_479/MATLAB/matlab3.pdf

http://web.cecs.pdx.edu/%7Emperkows/CLASS_479/MATLAB/matlab4.pdf