




MATHCAD  Mathcad 8

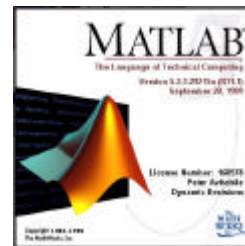
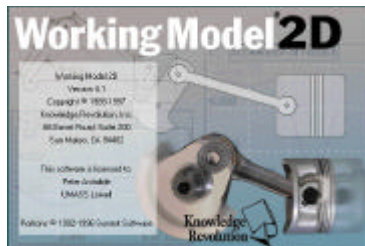
- General write up on MATHCAD usage
- Example of functions
- Example of finding roots
- Example of matrix manipulation
- Example of programming tools
- AVI file: *simple\_make\_sine.avi* (external file)

MATLAB  Matlab

- General write up on MATHLAB usage
- Example of functions as external files

Working Model  Working Model 2D Homework Edition 4.1

- General write up on Working Model usage
- AVI file showing the general steps to make a four bar linkage (external file):  
*WM-part1.avi*, *WM-part2.avi*, *WM-part3.avi*





## 22.321 MATHCAD Software Usage

MATHCAD is a general purpose mathematical software package commonly used in engineering analyses. Some basic familiarization using MATHCAD is presented in this document to acquaint the student with MATHCAD and its use. (Only use of MATHCAD as it pertains to its use in ME 22.321 is presented in this document.)

The first step is to start up MATHCAD

In order to run MATHCAD software :

From the START menu, click



Only a handful of commands will be presented here for the simple utilization of MATHCAD to generate some simple functions and plots of those functions. This will be followed by an example.

MATHCAD is a graphical program where *What You See Is What You Get (WYSIWUG)*. Since it is graphically based there are many commands that can be given using either the pull down menus or by using special keystrokes (basically, *secret handshakes*). As you step through this tutorial, these will be explained. Once MATHCAD has been started, you will see a clean page.

Now as you work, you may want to either save an existing session or reload a previous session that had been saved earlier. To do this, you can use the **File** command to **Open**, **Close** and **Save** (as in any typical windows based programs).

In MATHCAD, if you need assistance with a particular command, you can use the **HELP** function. This operates as any other windows based help function. The **Resource Center** is very useful in providing additional information to help in using MATHCAD. It is *strongly recommended* that you look at the **Getting Started** section to get more familiar using MATHCAD.



Rather than list out all of the MATHCAD commands that might be needed, let's start by trying to do some simple operations in MATHCAD to do some simple manipulations and plotting. Let's start with a simple function to plot.

First let's get a series of values for the variable  $x$ . Do this by clicking in the window (a red cross will appear) and typing

$x : 1 ; 5$                       but  $x := 1 .. 5$  appears

The first of the secret handshakes occurs. What happened here is that an assignment of  $x$  equal to the values 1,2,3,4,5 was made. The way this is done is by using the assignment “:=” statement but this is obtained by typing “:” at the keyboard.

The second secret handshake is that we need an array of numbers from 1 to 5 in increments of 1; this is obtained by typing “;” at the keyboard which produces “..” (which indicates a series of numbers from 1 to 5)

Now to see the values of  $x$ , then we would type

$x =$                                       and an array of numbers appears

Now just to allow for some formatting of this simple array, let's move the output of  $x$  towards the right side of the screen. To do this, move the cursor somewhere outside the equations generated thus far and use the normal windows “click and drag” to highlight the  $x$  output. (Notice that the regions become activated as dotted boxes appear around them, but change to solid lines as the mouse is released.) Now move the cursor to the exterior of the highlighted region and a hand will appear indicating that the mouse (with a click and hold) can be used to move the selected item on the page.

We can move these items anywhere on the screen, but the actual location is very important. It turns out that while MATHCAD is a very powerful tool, it is also very fussy! The items must appear in order from left to right and top to bottom. Basically, what that means is that you can't print out the values of  $x$  until they are defined. But that makes sense.

Now let's define  $x$  differently by using a subscripted array.

First, let's define an array with  $n$  equal to sequential values from 1 to 5. Type

$n : 1 ; 5$  which produces  $n := 1 .. 5$  on the screen

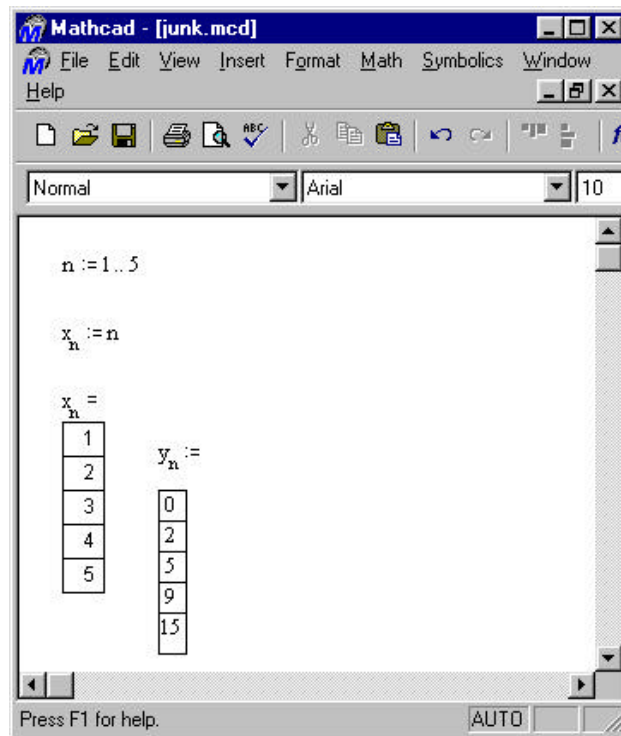
This now sets up an array of numbers that can be used as a subscript for variables.

Now let's enter in data for  $x$  and  $y$ . This will be done using a subscripted approach. First let's enter the two lines of information in MATHCAD and then explain what the input means. The variable  $x$  will be input by using the variable  $n$  directly and the values of  $y$  will be entered individually.

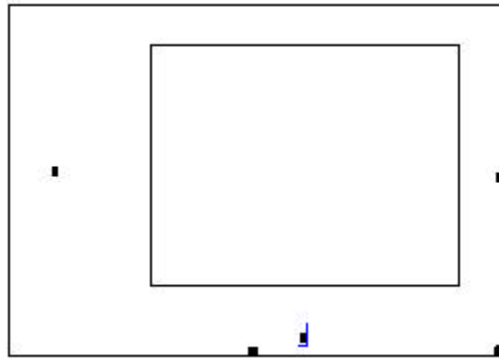
$x [ n : n$  which produces  $x_n := n$  on the screen  
 $y [ n : 0, 2, 5, 9, 15$  which produces  $y_n$  with the values shown in a column

The  $[$  indicates a subscripted variable. The  $x$  variable was subscripted with 'n' values which were assigned to be the values of 'n' and the results were shown in a column on the screen. The  $y$  values were subscripted with 'n' values which were typed in at the keyboard (separated by commas).

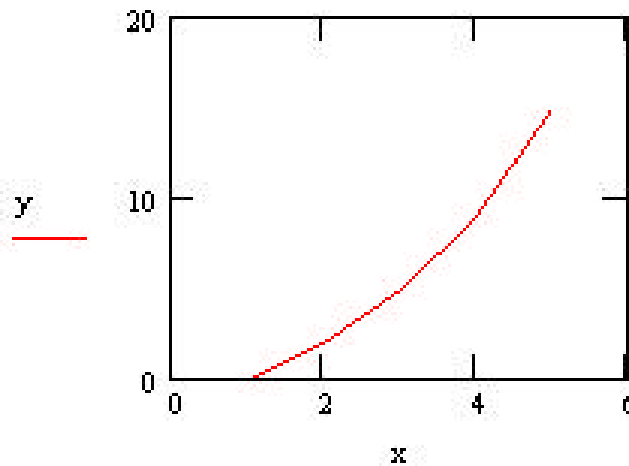
Right now the data on your screen should look something like the window below.



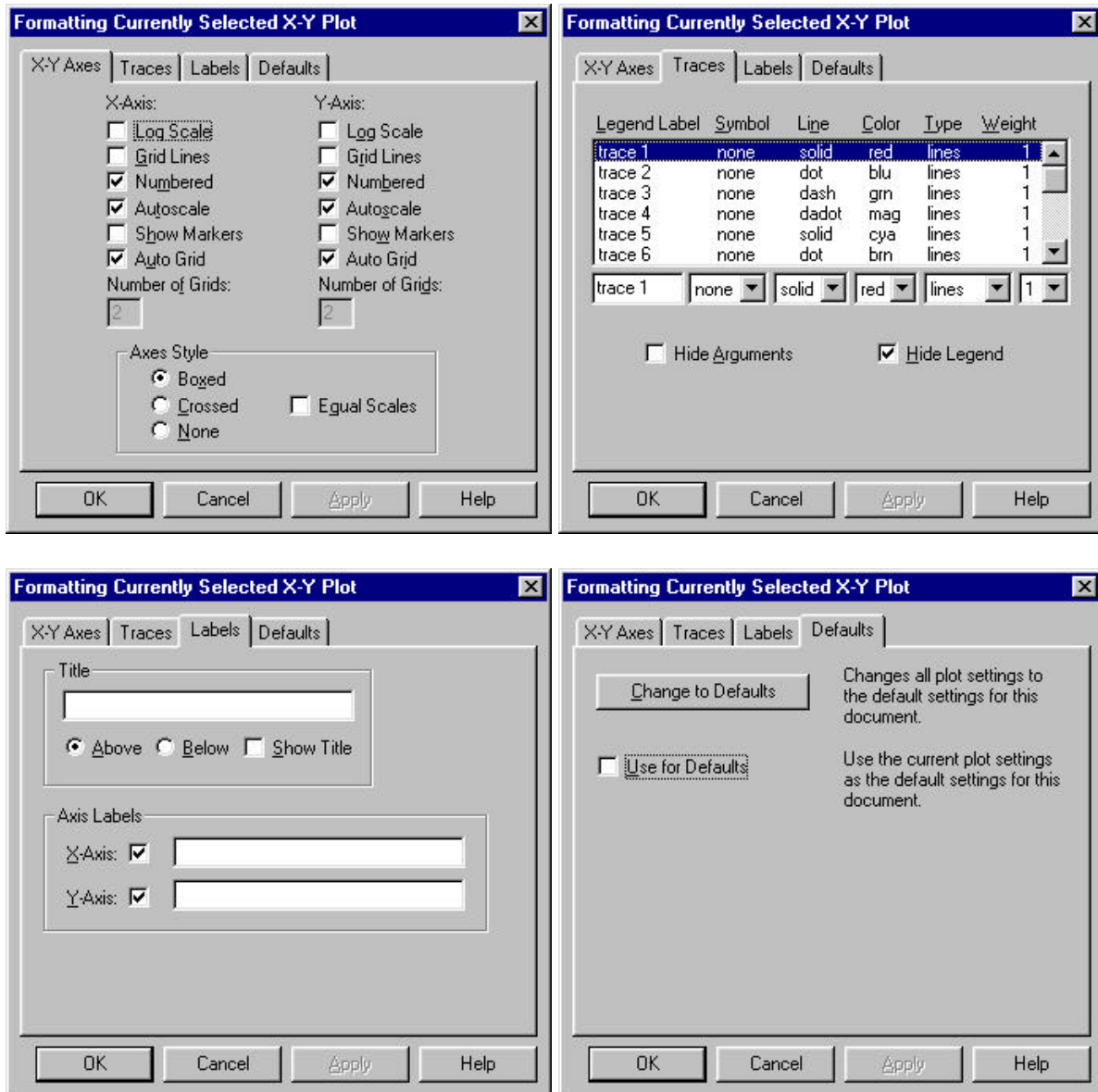
Now let's try to plot  $x$  and  $y$  and show the use of the plotting commands. At the keyboard, type @ which is a shortcut key to generate a plot. The plot window will open with no data or information to plot as shown below.



Click on the dark box along the  $x$  axis of the plot and enter  $x$ . Do the same for  $y$  along the  $y$  axis. Then click outside the plot box. The plot window will have data as shown below. Notice that MATHCAD did select some reasonable plot limits based on the values of  $x$  and  $y$  contained in those arrays.

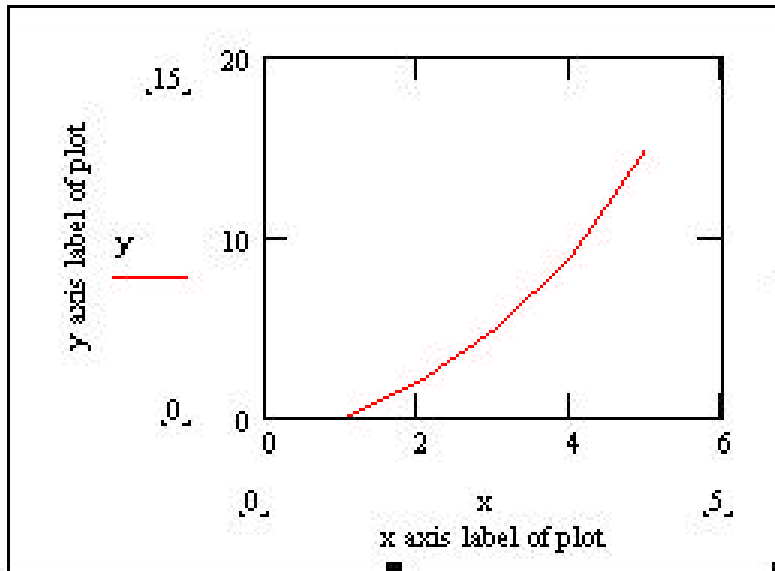
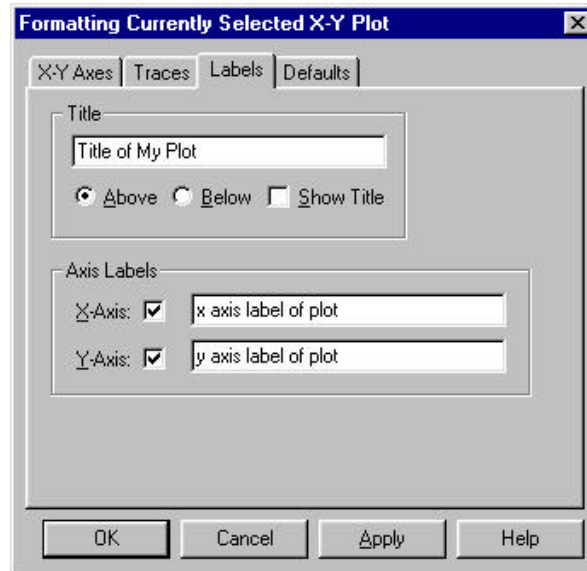


Double click on the plot window in MATHCAD and there are several plotting attributes that can be modified to change the “look and feel” of the plot. Explore some of these plotting options.



Now let's add some labels to this data. Double click on the plot to open the formatting window. Select the Labels tab and enter in:

Title:	Title of My Plot
X-Axis	x axis label of plot
Y-axis	y axis label of plot



Try other features to customize your plots

Now let's skip to something more interesting like plotting a function. First let's do a simple sine wave and then move on to an exponentially decaying sine wave.

Let's define the variable  $x$  to have values from 0 to  $4\pi$  and then plot the sine of it. First, define an index counter 'n' that has one degree increments from 0 to 720 degrees. Then, define the variable  $x_n$  to be the variable n (as was done in the previous example).

**n : 0 , 1 : 360\*2**  
**x [ n : n**

which produces **n := 0 , 1 .. 360\*2** on the screen  
which produces  **$x_n := n$**  on the screen

This variable needs to be converted to radians so that the sine of the function can be obtained.

**xrad [ n :  $\pi * x [ n / 180$**

which produces  **$xrad_n := \pi x_n / 180$**  on the screen

**y [ n : sin (xrad<sub>n</sub>)**

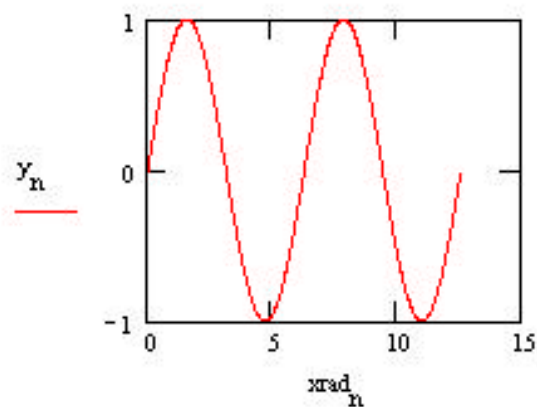
which produces  **$y_n := \sin(xrad_n)$**  on the screen

Now plot the function just as was done earlier. Note that the plot variables are typed in as

**xrad [ n**  
**y [ n**

which produces  **$xrad_n$**  on the screen  
which produces  **$y_n$**  on the screen

Then click outside the plot box to see a figure like the one shown below

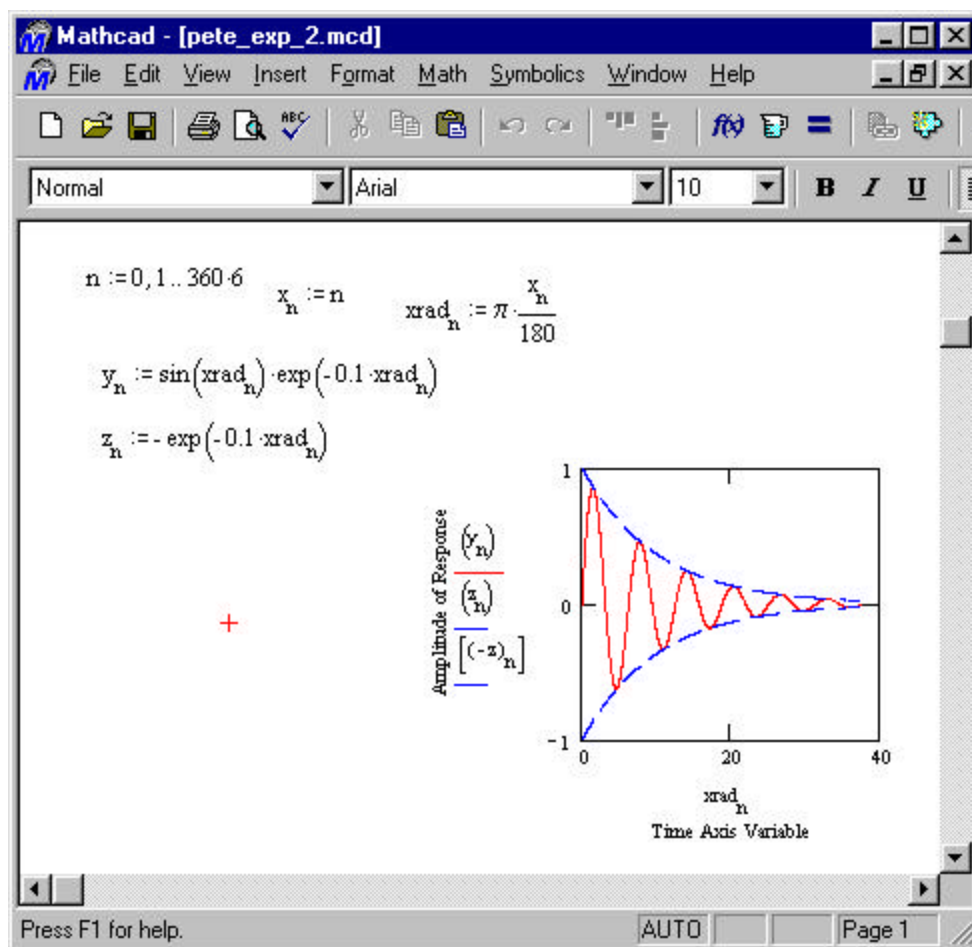




Now let's put a more difficult expression together and put together some more elaborate plotting commands with multiple plots. Only some brief explanations of the commands are given here.

Let's plot a damped exponential sine wave here (with 6 cycles) and overlay the exponential envelop on the plot – we will also include colors and line types and dump the plot out to the clipboard in WINDOWS so we can import the picture into WORD for instance.

Type the commands shown in the window below. Use the plotting Format tools to change the line type and colors of the envelop of the damped exponential sine wave. (In MATHCAD, multiple plots can be obtained for the y axis by separating all the variables with a comma)



As the MATHCAD file is modified, the file can be saved to the original file name or new file name. It is **strongly recommended** that you save the file periodically as you develop your equations and plots.

This is the generation of a sine using non-subscripted variables

$$x := 0, \frac{\pi}{4} .. 2 \cdot \pi$$

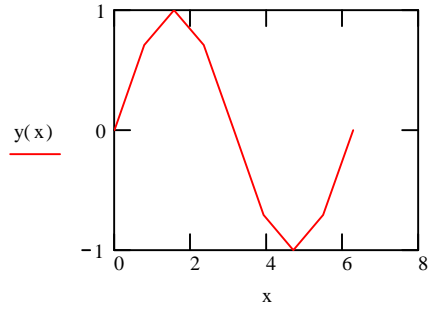
$$y(x) := \sin(x)$$

x =

0
0.785
1.571
2.356
3.142
3.927
4.712
5.498
6.283

y(x) =

0
0.707
1
0.707
0
-0.707
-1
-0.707
0



This is the same sine generated using subscripted variables

$$n := 1 .. 9$$

$$x_n := (n - 1) \cdot \frac{\pi}{4}$$

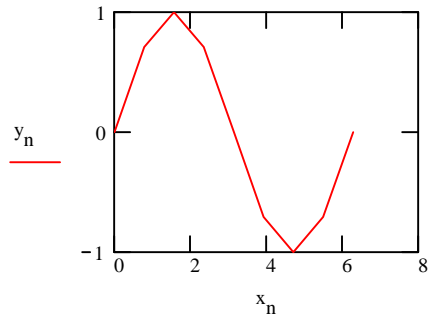
$$y_n := \sin(x_n)$$

x<sub>n</sub> =

0
0.785
1.571
2.356
3.142
3.927
4.712
5.498
6.283

y<sub>n</sub> =

0
0.707
1
0.707
0
-0.707
-1
-0.707
0



### Example of GIVEN and FIND MATHCAD Commands

$$x := 1$$

Given

The **GIVEN** command is typed on the screen  
It is NOT a Text Block for annotation

$$x^2 - 4 = 0$$

The equation to be evaluated is typed and the  
**CTRL/=** is used to complete the equation assignment

$$\text{Find}(x) = 2$$

**FIND** is then used to solve for the roots

---

$$x := -1$$

Given

$$x^2 - 4 = 0$$

$$\text{Find}(x) = -2$$

---

---

### Example of POLYROOTS Command

$$w := \begin{pmatrix} -4 \\ 0 \\ 1 \end{pmatrix}$$

Evaluate the same function using **POLYROOTS**

Enter a vector using the MATRIX (**CTRL/M**) command  
inserting the coefficients of the polynomial to be evaluated  
starting with the lowest order term in the first vector slot

$$\text{polyroots}(w) = \begin{pmatrix} -2 \\ 2 \end{pmatrix}$$

**POLYROOTS** finds the roots of the equation.

**Remember** to enter the polynomial coefficients  
from lowest to highest and all terms must be  
included even if one is not present as in this case

In MATLAB, similar commands exist. The coefficients of the polynomial are input to a vector array in descending order (opposite to MATHCAD) and then evaluated using the **ROOTS** command in MATLAB

```
>> C = [ 1 , 0 , -4 ];
```

```
>> ROOTS (C)
```

## Numerical Evaluation for Root Finding

The roots of the function below are 2,-2

On either side of zero there will be a switch between each of the roots

What happens close to zero?

$i := 1..10$

$x_i := -1 \cdot 10^{-i}$

Given

$(x_i)^2 - 4 = 0$

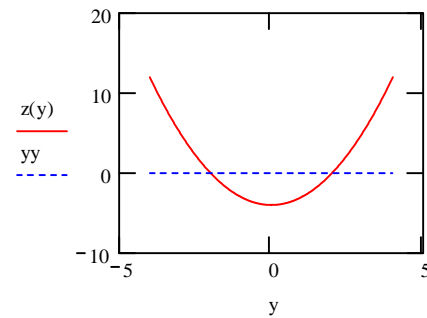
Find( $x_i$ ) =

	0
0	-2
1	-2
2	-2
3	-2
4	-2
5	-2
6	-2
7	2
8	2
9	2
10	

$y := -4, -3.99..4$

$yy := 0$

$z(y) := y^2 - 4$



The root extraction process uses a numerical scheme to find the roots. Care should be exercised in extracting roots. Most times the root finder will work fine but this simple example shows that the two different roots will be found with very small changes in the starting values

### Simultaneous Equations Solution - Matrix Inversion Approach

Let's define 3 equations with x, y and z given and step through a solution

Let x=1, y=2 and z=3. And the three equations are :

$$2x - y = 0 \quad ; \quad -x + 2y - z = 0 \quad ; \quad -y + z = 1$$

One way to solve these are through the use of matrices as follows:

$$\begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix} \begin{Bmatrix} x \\ y \\ z \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 1 \end{Bmatrix}$$

The 3x3 matrix needs to be inverted and post-multiplied by the vector on the right hand side of the equation to solve for x,y,z.

$$\begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix}^{-1} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

In MATHCAD use the **CTRL/M** to create a matrix or vector by specifying the size of the matrix. The matrix can be inverted by using the menu **Math -> Matrix -> Invert** or use the **Matrix Toolbar**

In MATLAB, similar commands exist. The matrix is input as a 3x3 matrix along with the 3x1 vector of the right hand side. The a solution is obtained for  $A x = B$  as shown below

```
>> A = [ 2 , -1 , 0 ; -1 , 2 , -1 ; 0 , -1 , 1 ];% The 3x3 matrix called A
>> B = [ 0 ; 0 ; 1 ]; % Vector B describing right hand side
>> C = A \ B; % Solution is found in C
```

### Simultaneous Equations Solution - Given/Find Approach

$$x := 0 \quad y := 0 \quad z := 0$$

Given

$$2 \cdot x - 1 \cdot y + 0 \cdot z = 0$$

$$-x + 2 \cdot y - z = 0$$

$$-y + z = 1$$

$$\text{Find}(x, y, z) = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

Initial variable values are assumed

The **GIVEN** command is typed on the screen  
It is NOT a Text Block for annotation

The equations are entered with **CTRL/=** to  
assign values for the equations

The **FIND** command is then used to solve  
for the variables in question

## Programming Toolbars

---

In **MATHCAD**, use the **Math Toolbar** to select the **Programming Icon**  
- a variety of tools appear with the most important being the program operator

$x := -2, -1 \dots 2$

$$y(x) := \begin{cases} -x & \text{if } x < 0 \\ x & \text{otherwise} \end{cases}$$

x =

-2
-1
0
1
2

y(x) =

2
1
0
1
2

In defining a program, the first item is use the **Add Line** from the **Programming Tool**. Then expressions can be specified as needed

In this example, the values of x which were less than zero are changed to positive values otherwise they were left unchanged

Note that the value of (-x) could also have been written as absolute |x|

---

In **MATLAB**, similar commands exist. **MATLAB** uses the **if else end** command shown

```
>> A = -2; % input arbitrary value for A
>> if A < 0
    A = -A % notice that there are NO command
        % prompts >> until end typed
    else A = A
    end
>> (answer appears here)
```

This needs to be typed every time this evaluation is needed.

A function can also be generated which makes this much easier and is the better approach.

In **MATHCAD**, another simple example using the program operator

---

$$x := 0, \frac{\pi}{6} .. 3 \frac{\pi}{2}$$

Let's consider the quadrants for a sine wave and find the absolute amplitude in each quadrant

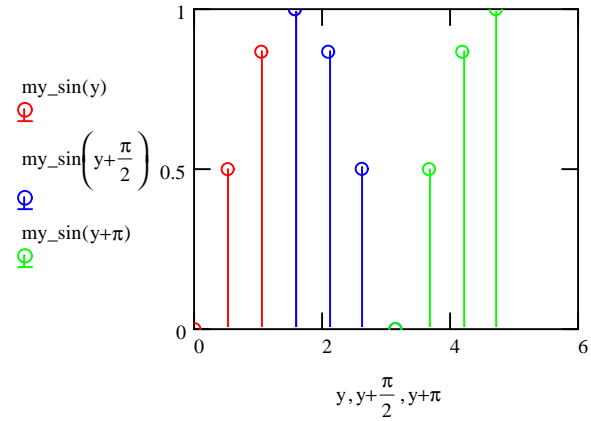
$$\text{my\_sin}(x) := \begin{cases} \sin(x) & \text{if } 0 \leq x \leq \frac{\pi}{2} \\ (\sin(\pi - x)) & \text{if } \frac{\pi}{2} < x < \pi \\ -\sin(\pi - x) & \text{if } \pi \leq x \leq 3 \cdot \frac{\pi}{2} \\ \left( \sin\left(x + \frac{\pi}{2}\right) \right) & \text{if } 3 \cdot \frac{\pi}{2} < x \leq 2 \cdot \pi \end{cases}$$

Consider the quadrants for a sine wave and find the absolute amplitude in each quadrant

The logic for this could be written many different ways

$$y := 0, \frac{\pi}{6} .. \frac{\pi}{2}$$

x =	sin(x) =	my_sin(x) =
0	0	0
0.524	0.5	0.5
1.047	0.866	0.866
1.571	1	1
2.094	0.866	0.866
2.618	0.5	0.5
3.142	0	0
3.665	-0.5	0.5
4.189	-0.866	0.866
4.712	-1	1



A **MATHCAD** hint for getting the **atan(y/x)** into the proper quadrant (handed out on last year's take home exam)

Using the programming toolbar will make creating this function a snap.

Once this function is defined within the program, simply use atan2(x,y) like any other function built-in function in MATHCAD. The function needs to be defined near the top of the worksheet so that it can be referenced further down in the worksheet. (J.Sherwood 16 Dec 1999)

---

$$\text{atan2}(x, y) := \begin{cases} (0.5 \cdot \pi) & \text{if } x = 0 \\ \text{atan}\left(\frac{y}{x}\right) & \text{if } x > 0 \\ \left(\text{atan}\left(\frac{y}{x}\right) + \pi\right) & \text{otherwise} \end{cases}$$





## 22.321 MATLAB Software Usage

MATLAB is a general purpose software package commonly used in engineering analyses. Some basic familiarization using MATLAB is presented in this document to refresh or re-acquaint the student with MATLAB and its use. The student is expected to have seen some use of MATLAB in previous engineering courses. (Therefore, only use of MATLAB as it pertains to its use in ME 22.321 is presented in this document.)

The first step is to start up MATLAB  
(assuming that a Windows based version is being used)

In order to run MATLAB software :

From the START menu, click



Only a handful of commands will be presented here for the simple utilization of MATLAB to generate some simple functions and plots of those functions. This will be followed by an example.

When starting MATLAB, the commands that are entered at the MATLAB prompt can be saved to a log file called a diary file using

**diary filename** (the filename is your choice)

The log file can be turned on and off using **diary on** and **diary off** as desired.

Upon starting MATLAB, typically you will be working in one of the MATLAB directories such as **C:\MATLABR11\bin**. In order to determine this you can use

**pwd** (which identifies the current working directory)



In order to move to a different directory to do your work, you can change using

**cd c:\temp** (this does a change directory to c:\temp for instance)

Now as you work, you may want to either save an existing session or reload a previous session that had been saved earlier. To do this, you can

<b>save</b>	(save the session to default matlab.mat)
<b>save new_session.mat</b>	(save to a particular file with .MAT)
<b>load previous_session.mat</b>	(reload a previous saved file)

In MATLAB, if you need assistance with a particular command, you can type

<b>help</b>	(lists all MATLAB help info)
<b>help "command"</b>	(provides help on particular command)

Another important feature of MATLAB is that you can write very simple script files that can be saved as text files and used to run MATLAB. Later in this session, we will discuss script files further.

Rather than list out all of the MATLAB commands that might be needed, let's start by trying to do some simple operations in MATLAB to do some simple manipulations and plotting. Let's start with a simple function to plot.



First let's get a series of values for the variable x. We will do this by typing

`x = [ 1 : 1 : 5 ]`                      Array x starts at 1, increments by 1, to 5  
`y = [0 2 5 9 15]`                      Array y has values shown

You will notice that as you typed this, the values appeared on the screen. In order to prevent this, the commands can be ended with a semi-colon ";" and then the output will not be shown on the screen. But if you wanted to check what got entered in as your data, then you could type

`x = [ 1 : 1 : 5 ];`  
`y = [0 2 5 9 15];`  
`x`    (to list the values in x)  
`y`    (to list the values in y)

(One point to make is that MATLAB really thinks in terms of linear algebra as its main purpose in life. This means that it really expects to see data as vectors and matrices. The data we have entered above is actually a row vector - not a column vector)

Before we plot, we might want to put these values in a table for printing

`table( : , 1 ) = x';`                      (puts all data of x in column 1 of table)  
`table( : , 2 ) = y';`                      (puts all data of y in column 2 of table)  
`disp(table);`                              (prints out table of values in columns)

Now let's plot this data and look at some plotting alternatives.

`plot(x,y)`                                      (shows data in plot window)

Now let's add some labels to this data.

`title('title of my plot')`                      (plot title)  
`xlabel('x axis label of plot')`                      (x axis label)  
`ylabel('y axis label of plot')`                      (y axis label)

You can do various customizations of your plots – see additional info with **help plot**



Now let's skip to something more interesting like plotting a function. First let's plot a simple sine wave and then move on to an exponentially decaying sine wave.

Let's define the variable  $x$  to have values from 0 to  $4\pi$  and then plot the sine of it.

<b>x = [ 0 : pi/180 : 4*pi];</b>	(values incremented by 1 degree)
<b>y = sin (x);</b>	(sine computed using radians)
<b>plot(x,y)</b>	(shows data in plot window)
<b>title('2 cycles of sine')</b>	(plot title)
<b>xlabel('radians')</b>	(x axis label)
<b>ylabel('amplitude')</b>	(y axis label)

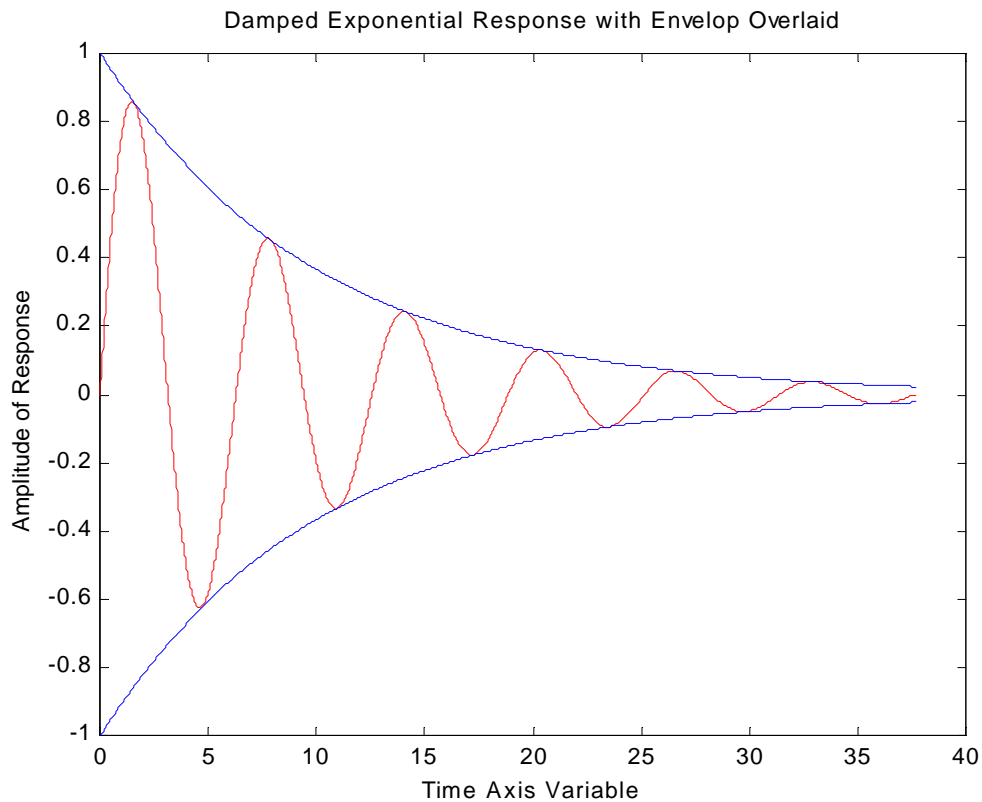
Now let's put a more difficult expression together and put together some more elaborate plotting commands with multiple plots. Only some brief explanations of the commands are given here.

Let's plot a damped exponential sine wave here and overlay the exponential envelop on the plot – we will also include colors and line types and dump the plot out to the clipboard in WINDOWS so we can import the picture into WORD for instance.

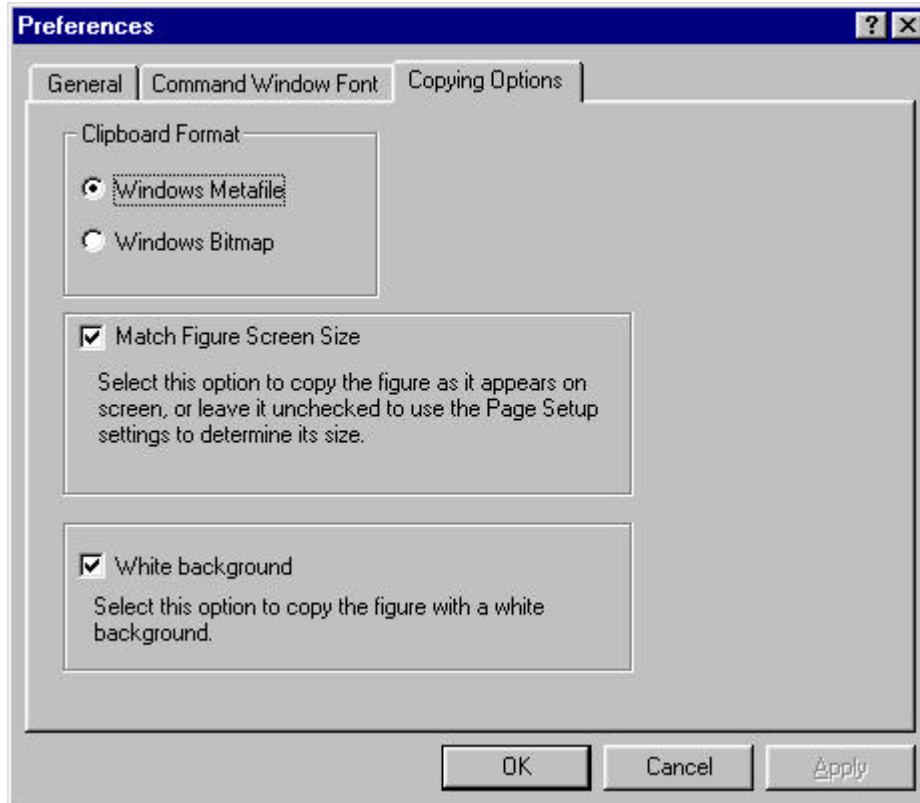
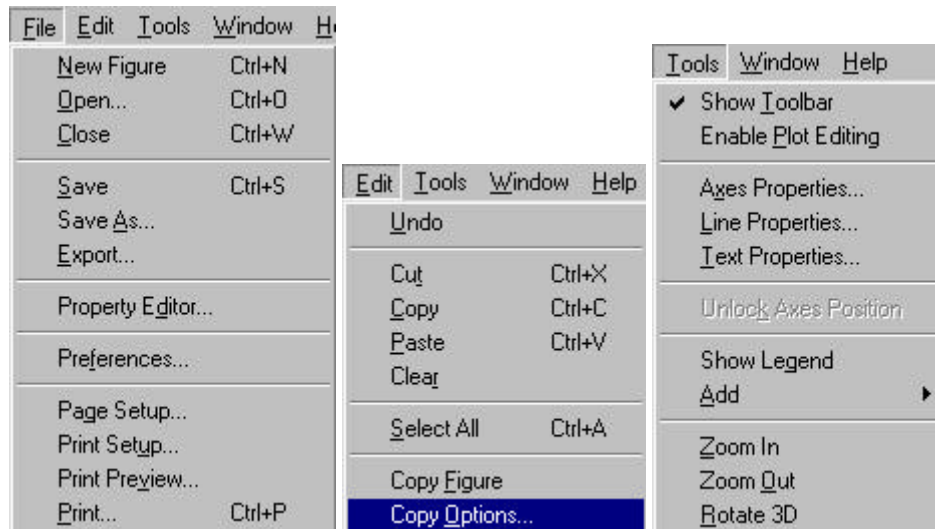


(In MATLAB, when we want to multiply terms of an array together one at a time and not perform a matrix/vector multiplication, then we will need to precede the operator symbol with the “.” command – take note of it in the commands below.)

```
x=[ 0 : pi/180 : 12*pi ];  
y = sin(x) .* exp(-x/10);  
z = exp(-x/10);  
plot( x, y, 'r', x, z, 'b--', x, (-z), 'b--' )  
title('Damped Exponential Response with Envelop Overlaid')  
xlabel('Time Axis Variable')  
ylabel('Amplitude of Response')
```

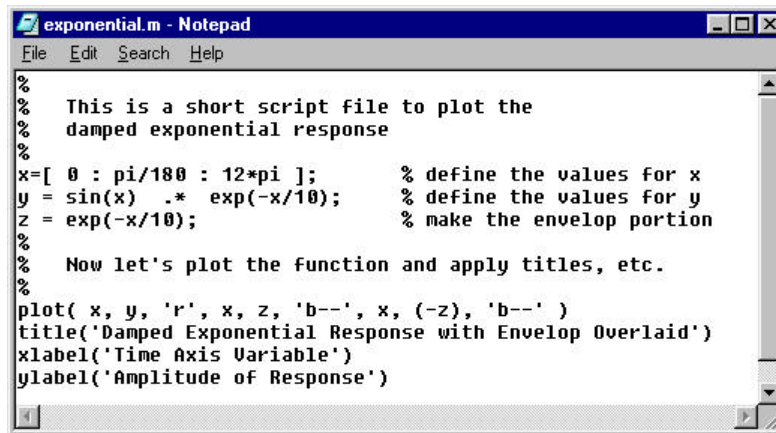


There are a variety of features that are available in the MATLAB plot window to customize the plots. It is advised that you explore some of these capabilities.



One other useful feature in MATLAB is to use a script file, commonly called an “m” file since the name of the file always has an extension of ‘.m’

Basically, the text editor can be used to generate a file that can be used in MATLAB. In the example below, a file was created in Notepad in Windows and saved to the file named “exponential.m”. This file was then executed by typing the file name without the extension at the end. As long as there are valid MATLAB commands in this file then the resulting output is exactly the same as typing all the commands at the command prompt in MATLAB. The file is shown below for reference.



```

%
% This is a short script file to plot the
% damped exponential response
%
x=[ 0 : pi/180 : 12*pi ];      % define the values for x
y = sin(x) .* exp(-x/10);    % define the values for y
z = exp(-x/10);              % make the envelop portion
%
% Now let's plot the function and apply titles, etc.
%
plot( x, y, 'r', x, z, 'b--', x, (-z), 'b--' )
title('Damped Exponential Response with Envelop Overlaid')
xlabel('Time Axis Variable')
ylabel('Amplitude of Response')

```

The commands typed are shown below. Note that the ‘%’ allows for comment lines to be added to the script file to allow for easier reading. This file can be run by typing its name (without its extension) at the MATLAB command prompt (providing you are in the current working directory)

```

%
% This is a short script file to plot the
% damped exponential response
%
x=[ 0 : pi/180 : 12*pi ];      % define the values for x
y = sin(x) .* exp(-x/10);    % define the values for y
z = exp(-x/10);              % make the envelop portion
%
% Now let's plot the function and apply titles, etc.
%
plot( x, y, 'r', x, z, 'b--', x, (-z), 'b--' )
title('Damped Exponential Response with Envelop Overlaid')
xlabel('Time Axis Variable')
ylabel('Amplitude of Response')

```

```

%
% Script file to illustrate the use of a function script
% (NOTE: Function below must be in a separate file)
%
x = -5;                % pick a negative value for x
%
make_pmz(x);          % this will return the value +5
%
y = make_pmz(x)       % this returns the value to variable y
%
%
x = 19;               % pick a positive value for x
%
y = make_pmz(x)       % this returns the positive value to variable y
%
%
zero = 0               % now try a value of zero
%
wow = make_pmz(zero)   % WOW that's a big value
%
%-----
%
% The lines after this point need to be put in a separate file
% that must be named 'make_pm.m' which is then a function file
% The first line of the script must contain 'function'
%
% This function can then be called from any MATLAB script
% (note that function must be in the 'path' or 'pwd'
%
%-----FUNCTION STARTS ON THE NEXT LINE-----
function [a]=make_pmz(a)
%
% return positive value for plus or minus value
% and when zero return value 1234567 !!! (as an example)
%
if a == 0
    a = 1234567;
elseif a < 0
    a = -a;
else a = a;
end
%-----FUNCTION ENDS HERE-----
%
%
%      /\ /\ /\ /\      SEPARATE FILE BELOW      /\ /\ /\ /\
%
%
%----- ANOTHER SCRIPT FILE FOR ATAN QUADRANT DETERMINATION -----
function [my_atan]=check_atan(x,y)
%
% Get the ATAN into the correct quadrant
% (adapted from J.Sherwood's MATHCAD file)
%
if x == 0
    my_atan = pi/2;
elseif x > 0
    my_atan = atan(y/x);
elseif x < 0
    my_atan = (atan(y/x)+pi);
end
%-----

```



## 22.321 WORKING MODEL Software Usage

Working Model is a commercially available motion simulation software package. Some basic familiarization using Working Model is presented in this document; a separate more detailed and complete tutorial is available and handed out separately. (Only the use of Working Model as it pertains to its use in ME 22.321 is presented in this document and only to help acquaint the student with some of its basic features.)

In order to run Working Model software :

From the START menu, click



A program startup screen similar to that shown in Figure 1 will appear.

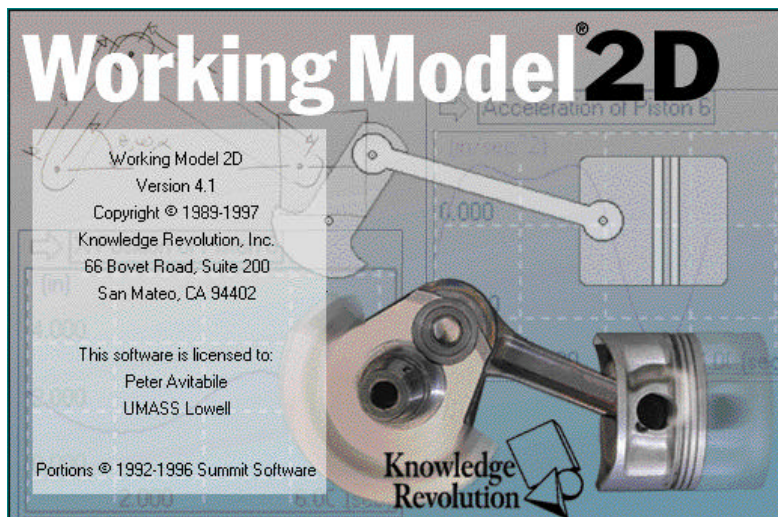


Fig 1 – Working Model Startup Screen

This tutorial will expose the student to some of the features of Working Model through the use of a simple example problem. Various aspects of Working Model will be highlighted through the use of this example problem (rather than provide a detailed description of each command as a separate item).

The main window of Working Model is shown in Figure 2. There are several categories of tools that are identified in the figure. Some of these tools will be used in this tutorial in order to acquaint the student with the use of Working Model.

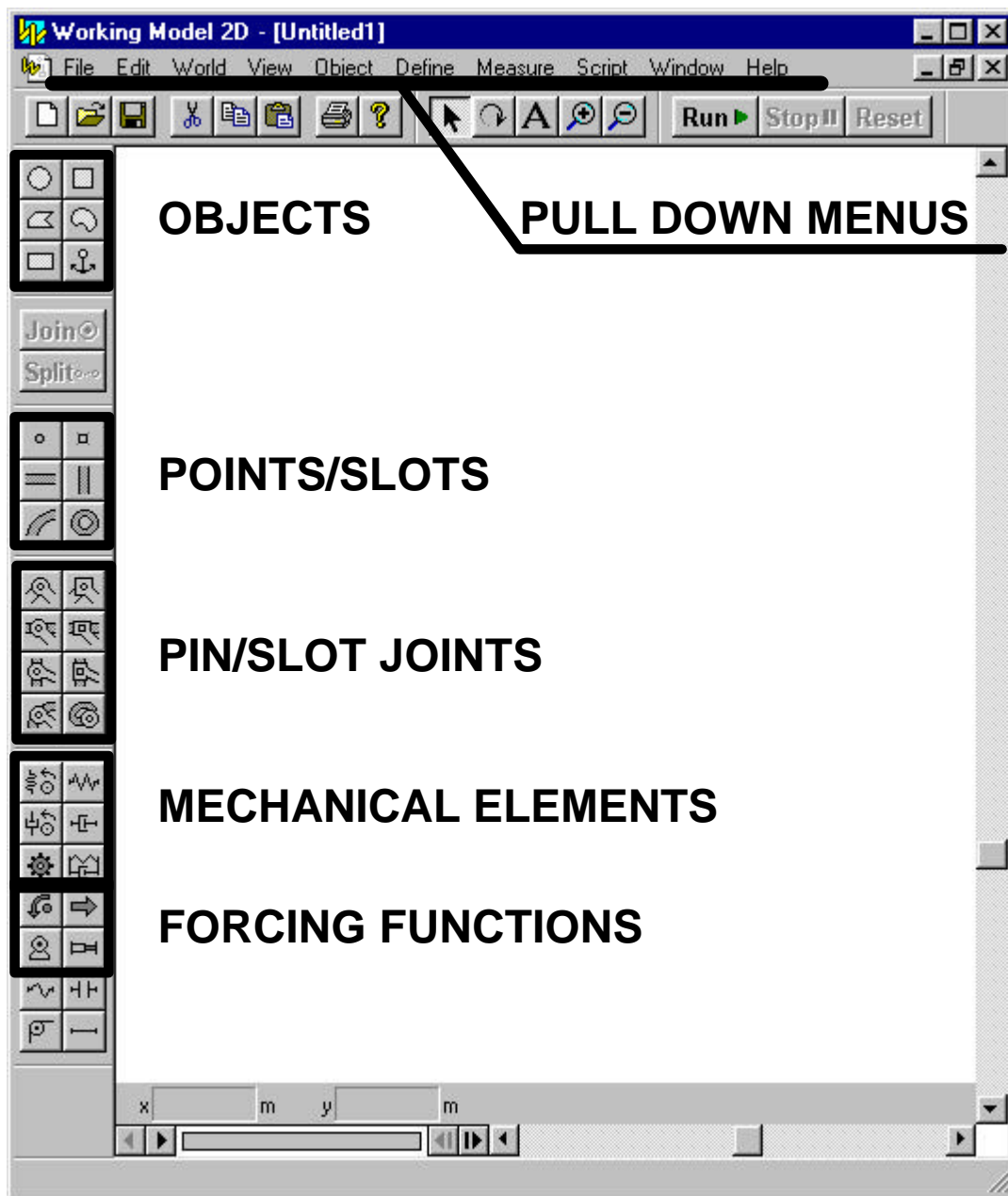
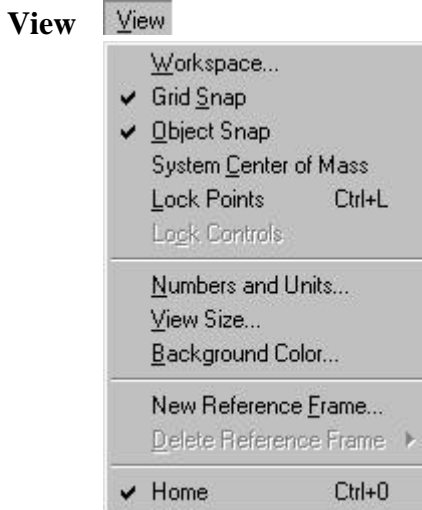


Fig 2 – Working Model Screen Regions

Let's start by trying a simple four bar linkage. With Working Model started, first let's enable some features in the **Workspace**.



**Workspace** (turn on **Rulers** and **Grids** to provide better workspace description)  
**Numbers and Units** (to select the desired units **English(pounds)** for this example)  
**View Size** (to set the screen size at 10 inches for this example)  
 Figure 3 shows these screen windows.

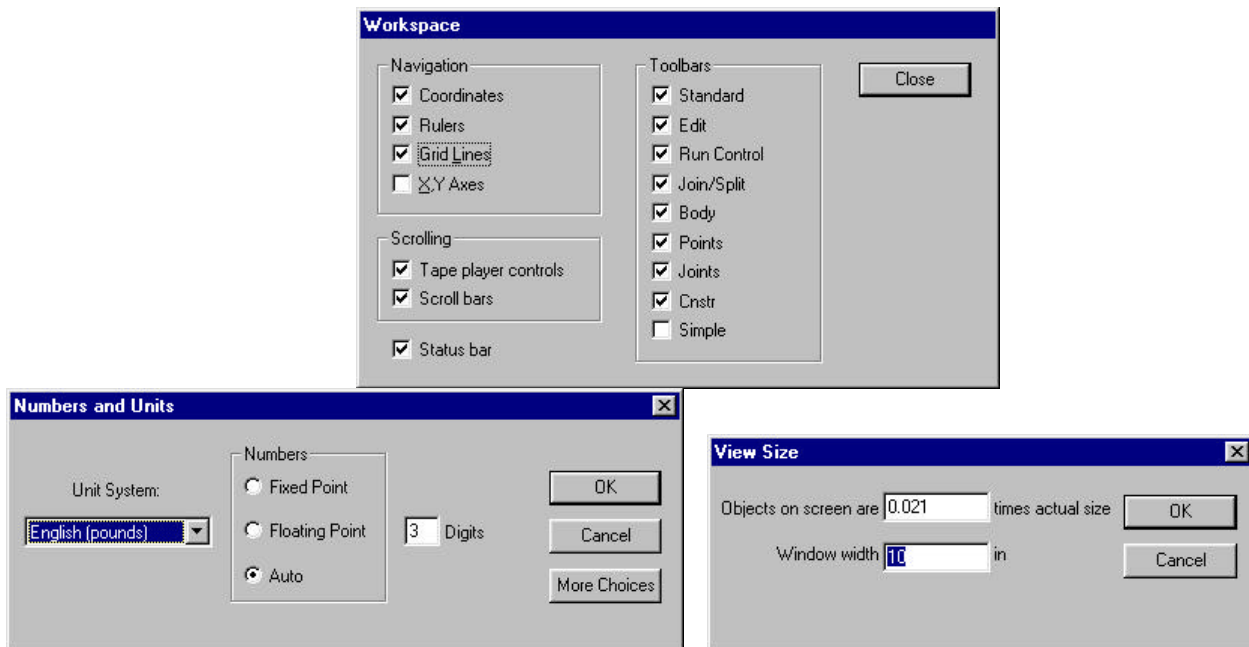


Fig 3 – Working Model Screens – Workspace, Numbers and Units, and View Size



Now select the rectangular object tool and draw a link with dimensions 2 inch long (height in vertical direction) by 0.25 inch wide; locate the link so that the lower left corner of the link is at (0.0,0.0). As the link is drawn, the geometric x,y coordinate locations are shown at the bottom of the screen as shown in Figure 4.

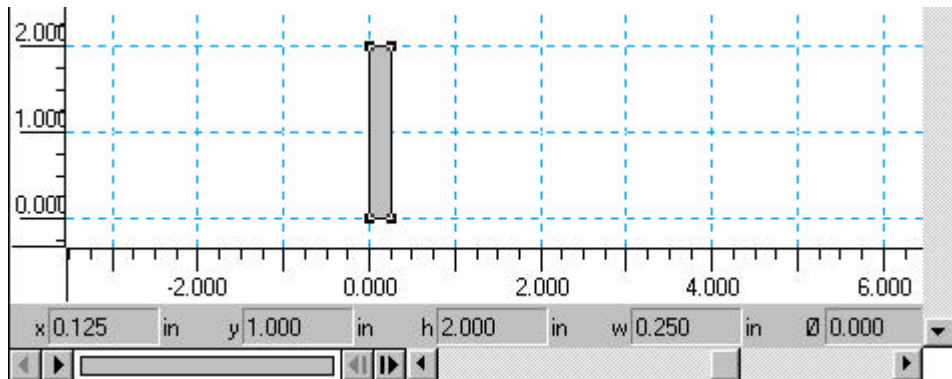


Fig 4 – Working Model Geometry Information Region

Draw another link with dimensions 1 inch long (height in vertical direction) by 0.25 inch wide; locate the link so that the lower left corner of the link is at (2.5,0.0).



Now draw a triangular link using the polygon tool. Set the vertices at (0,3) for the first point, (2.5,3) for the second point, and (1,5) for the third point; close the polygon by clicking on the first point.

The current workspace will look like that shown in Figure 5. The **Properties**, **Appearance** and **Geometry** can now be viewed. These can be used to modify the characteristics of the links draw (as well as any additional Working Model components that are included in the model). Each of the components can be selected in the pull down menus in these individual windows.

**Properties** (properties can be specified here)

**Appearance** (parameters can be specified here – note the **Track ...** selections)

**Geometry** (coordinates defining each component are defined here)

Figure 6 shows these screens.

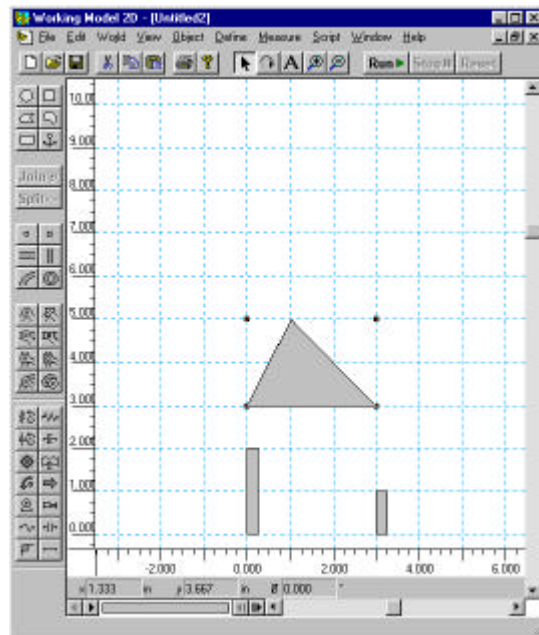


Fig 5 – Working Model with Unconnected Linkage Parts

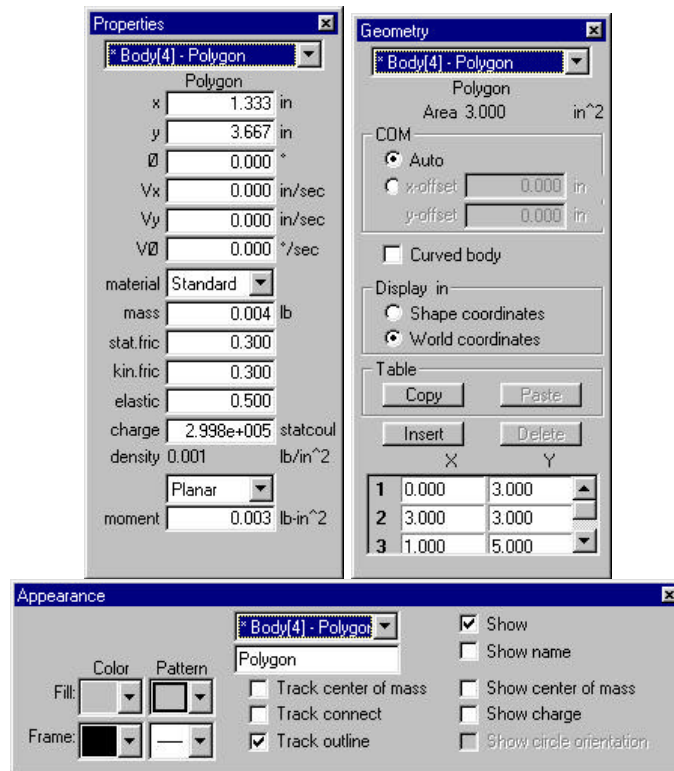
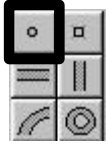
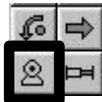


Fig 6 – Working Model Screens – Properties, Geometry and Appearance

Now some points need to be identified on the links created so that the links can be joined.



Use the point tool to specify points on the links. Select points at the end of each link and on the ternary link. (Make sure that the points on the links are at the mid-point of the link and at the corners of the ternary link).



Select the motor tool and place a motor at the lower end of the shorter link. The rotation speed of the motor can be specified in the **Properties** window. Now depending how the motor was placed at the end of the link, the motor may already be attached to the end of the link.



Press the Run button and observe what happens. If all the links fall to the bottom of the screen, then the motor is not hooked to the end of the link. However, if the shorter link rotates about the center of the motor, then the motor and link are hooked together. At this point, the links need to be **Joined** together or **Pinned** to ground to form an assembled system.

Select the point that was placed at the lower end of the longer link. This can be done by clicking on the point or using the mouse to draw a selection box around the point at the end of the link. (Make sure that the point doesn't move from the mid-side of the link.) This approach can be used to select one or several points simultaneously for joining purposes.



Now let's select the pin tool and select the lower end of the long link to be hooked to ground with a simple pinned joint.

Now select the point at the upper end of the longer link and the lower left corner of the ternary link so that these two points are the only points selected in the window.



Use the join tool to connect these two points. Also select the lower right corner of the ternary link with the point at the upper end of the shorter link and join these points. Your Working Model should now look like Figure 7. (The components can also be named as shown and **Tracking** turned on as desired). Figure 8 shows the associated appearance screens.



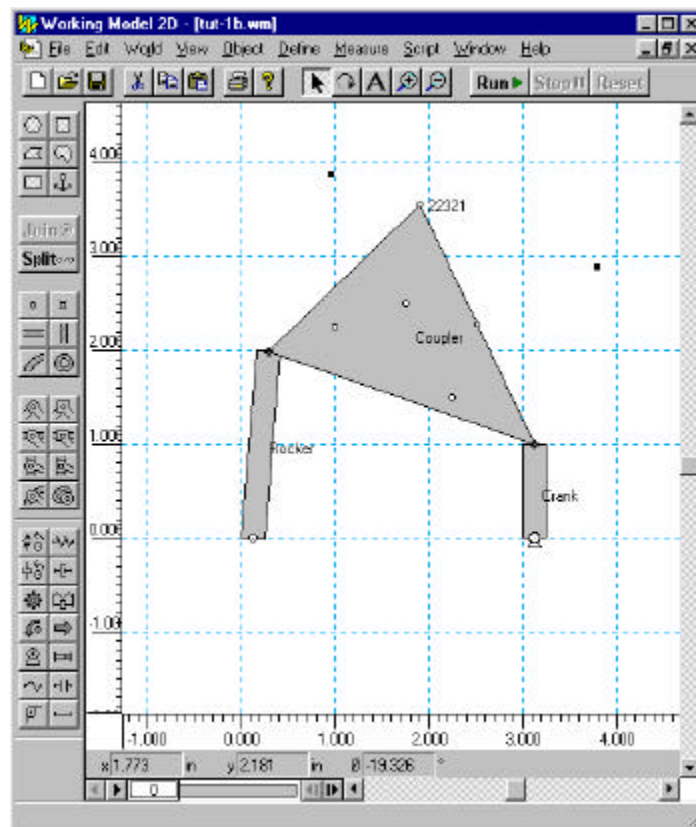


Fig 7 – Working Model with Linkage Connected and Components Identified

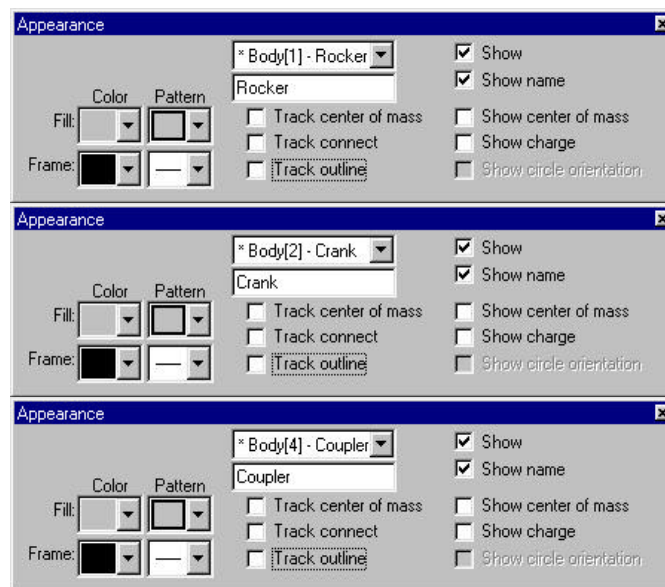



Fig 8 – Working Model Appearance Information Associated with Figure 7

Now  the assembled system and observe what happens. Watch the link motion.

Now go select several points on the coupler and place points at those locations. Set the properties of these points so that the motion can be tracked.

Now select the coupler and change its **Appearance** so that the fill color is white and there is no pattern selected (which allows the coupler to appear transparent) as shown I Figure 9.

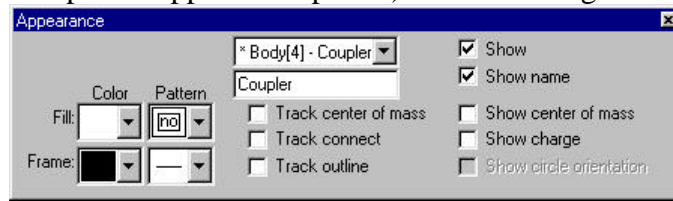
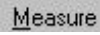




Fig 9 – Working Model Appearance Screen for Transparent Definition for Body 4

Select the point at the top of the coupler that it not attached to a link. Once the point is selected, then



select measure  and position  to open up a window for plotting the displacements of the selected points. The plot box is shown in Figure 10. Click on the arrow  to change the style of the plot.

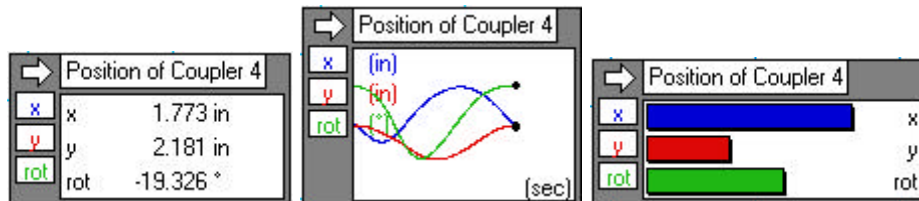


Fig 10 – Working Model Plot Screen Types

In order to erase the screen to start with a fresh animation,

select  and then 



Figure 11 shows the final design with the coupler curves for the points selected on the coupler and the position of the selected point on the coupler.

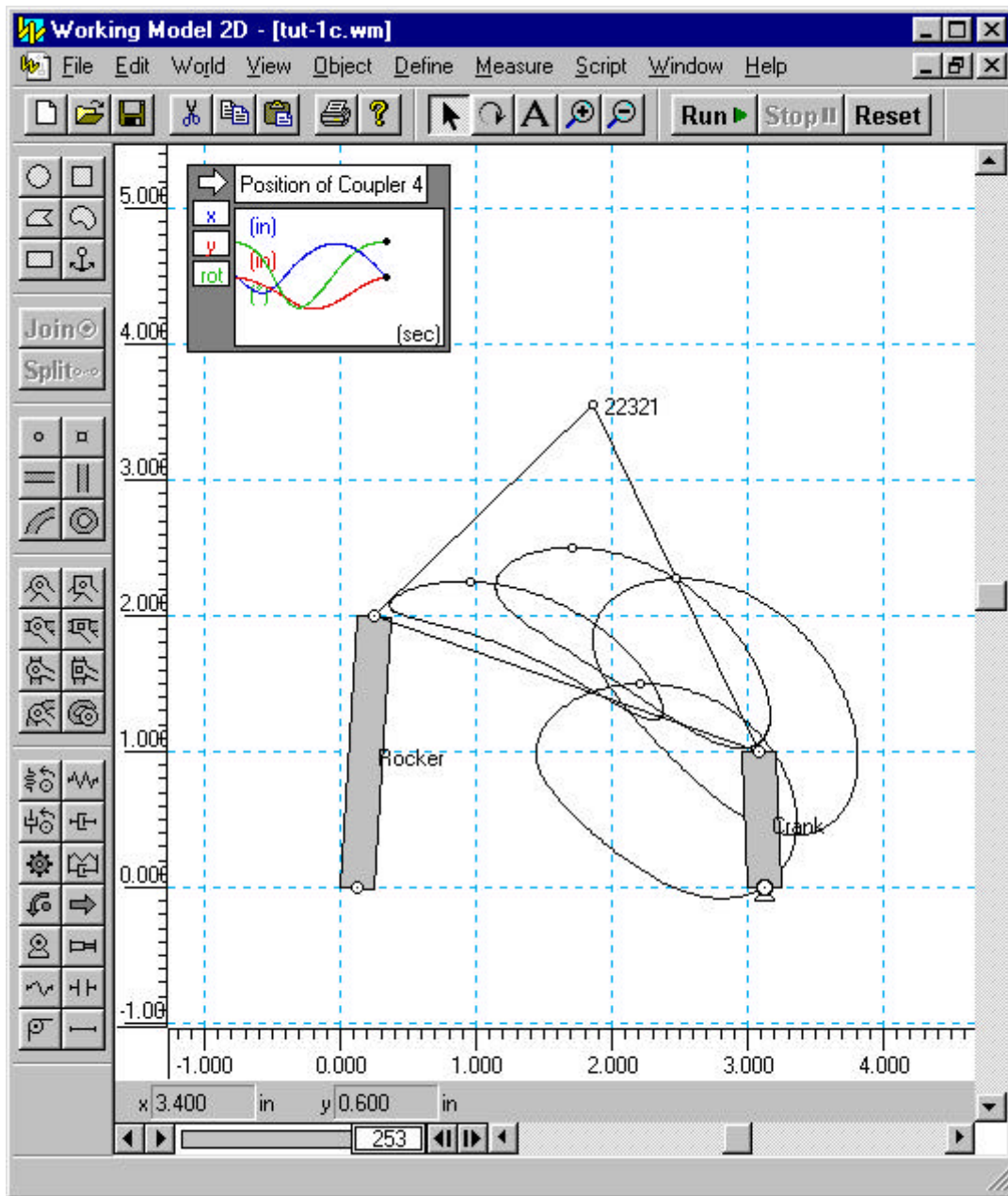


Fig 11 – Working Model Screen Showing Coupler Curves and Plot Window

## CREATING LINKS IN A CAD PACKAGE AND IMPORTING INTO WORKING MODEL

The links for your design may come from a CAD drawing package such as AutoCAD, CADKey or Pro-Engineer. When designing links in this manner, the links should all be drawn separate from each other. The resulting CAD file should be exported (from the CAD software) to create a .DXF file which can be read into Working Model.

To import the CAD database,



And then select the file that contains the linkages.

Once the linkages have been imported into Working Model, the lines connecting the linkages may be seen by Working Model as separate entities. If this is the case, then it may be necessary to group lines together to form a link or polygon. To do this, use

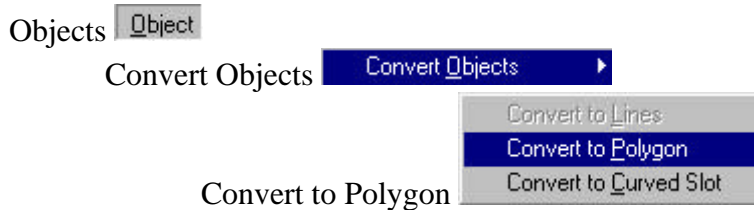


Figure 12 shows the ternary link represented only by three lines defining the profile of the coupler. Once the points are selected and a polygon identified, the properties can be changed to select color, etc. as shown in the Figure 13.

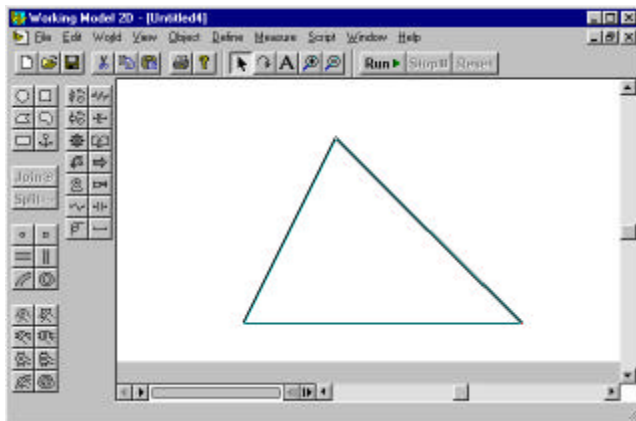


Fig 12 – Working Model Linkage Showing Lines

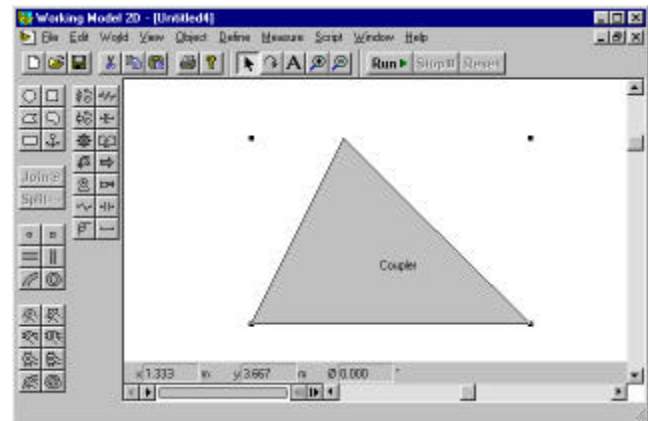


Fig 13 – Working Model Startup Screen