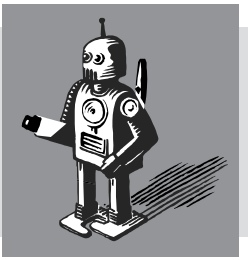
B

MATLAB with Allied Toolbox and Software



In this appendix, steps to use the commercial software MATLAB (WR: Matlab) and allied software like MuPAD, Simulink are explained. Two independent developments based on MATLAB software, e.g., Robotics Tool Box for MATLAB (Corke, 2011), and Recursive Dynamic Simulator or ReDySim (Shah et al., 2013) are also explained. A good account of how to use MATLAB is given in Pratap (2010).

B.1 USE OF MATLAB

MATLAB is a commercial software by MathWorks Inc., USA. It has large number of mathematical operators and commands that can perform a wide range of analysis, e.g., matrix operations, algebraic and differential equation solutions, optimizations, control experiments, etc.

B.1.1 Initialize MATLAB

It is expected that MATLAB is installed in a computer where the user will be performing computations. If an icon is available on the Desktop, the user has to double-click on it using the left button of the mouse connected to the computer. Alternatively, the user can left-click, in sequence, on the button/menus that pop up: Start -> Programs -> Matlab-> MATLAB.

B.1.2 How to use MATLAB

When MATLAB software starts, the MATLAB screen appears with the ">>" prompt. This window is called MATLAB command window. Some basic operations are shown here. For detailed description, one may refer to books available on MATLAB, e.g., Pratap (2010), and use the "Demos" and "Help" menus of the software.

B.1.3 Vectors in MATLAB

For any column vector or what is simply referred in this book as a vector, say, $\mathbf{a} \equiv [a_1 \quad a_2 \quad a_3]^T$, the following commands are used:

```
>> syms a1 a2 a3
>> av = [a1; a2; a3]
av =
[a1]
[a2]
[a3]
```

where "syms" is a MATLAB command to define the variables as *symbolic,* instead of numerical. Moreover, "av" is a user defined variable name for the vector **a**. In contrast, a row vector, say, $\mathbf{a} \equiv [a_1 \quad a_2 \quad a_3]$, can be generated as

```
>> syms a1 a2 a3
>> av = [a1 a2 a3]
av =
[a1, a2, a3]
```

If the elements of the row vector are numbers, say, $\mathbf{a} \equiv [2 \quad 1 \quad 5]$ then

```
>> av = [2 1 5]
av  =
     2     1     5
```

Addition of two vectors, say, $\mathbf{a} \equiv [2 \quad 1 \quad 5]^T$ and $\mathbf{b} \equiv [1 \quad 4 \quad 2]^T$, are obtained as

```
>> av = [2; 1; 3];
>> bv = [1; 4; 2];
>> cv = av+bv
cv  =
3
5
5
```

The scalar product of the above two vectors **a** and **b** is obtained as

```
>> av'*bv
ans  =
12
```

where the prime (') operator in MATLAB means *transpose*. Moreover, no variable is assigned on the left of the operation. Hence, it is given in MATLAB as "ans" meaning *answer*.

B.1.4 Matrices in MATLAB

Let two matrices, **A** and **B**, are given by

$$\mathbf{A} \equiv \begin{bmatrix} 2 & 3 & 1 \\ 3 & 5 & 2 \\ 5 & 1 & 6 \end{bmatrix}; \text{ and } \mathbf{B} \equiv \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 5 \\ 8 & 2 & 9 \end{bmatrix} \tag{B.1}$$

Their representations in MATLAB are as follows:

```
>> am = [2 3 1;3 5 2;5 1 6]
am  =
     2     3     1
     3     5     2
     5     1     6
>> bm = [1 2 3;2 3 5;8 2 9]
bm  =
     1     2     3
     2     3     5
     8     2     9
```

Addition of the above two matrices **A** + **B** is obtained as

```
>> am+bm
ans  =
     3     5     4
     5     8     7
    13     3    15
```

whereas the multiplication of **A** and **B**, denoted by **AB**, is given by

```
>> am*bm
ans  =
    16    15    30
    29    25    52
    55    25    74
```

Besides algebraic operations, one can find the properties of the matrices as well, e.g., determinant, eigenvalues, etc. For example, the determinant of **A** is calculated as

```
>> det(am)
ans  =
    10
```

whereas eigenvalues of **A** are obtained using the following command:

```
>> eig(am)
ans  =
    9.2085
    0.3121
    3.4794
```

Moreover, the inverse of **A** can be evaluated as

```
>> inv(am)
ans  =
    2.8000   -1.7000    0.1000
   -0.8000    0.7000   -0.1000
   -2.2000    1.3000    0.1000
```

Finally, the solution of a set of algebraic equations written in the form of $\mathbf{Ax} = \mathbf{b}$, i.e.,

$$\begin{aligned} 2x + 3y + z &= 1 \\ 3x + 5y + 2z &= 4 \\ 5x + y + 6z &= 2 \end{aligned} \tag{B.2}$$

is given by $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$. Note that the matrix **A** is given in Eq. (B.1), whereas the vector **b** has appeared in Section B.1.3. Vector **x** of Eq. (B.2) is defined as $\mathbf{x} \equiv [1 \quad 4 \quad 2]^T$. The solution **x** in MATLAB is obtained as

```
>> xv = am\bv
xv  =
   -3.8000
    1.8000
    3.2000
```

One can now check the multiplication of the matrix **A** with the vector **b** as

```
>> am*xv
ans  =
    1.0000
    4.0000
    2.0000
```

which is same as the vector **b**.

It is pointed out here that whatever has been done above in the MATLAB command window can also be typed in a .m file, say, "soln.m" which can then be run from the MATLAB window or clicking "Debug->Run" in the file editor. Here, the software may prompt for MATLAB path setting. The user should specify the one where "soln.m" is saved. In either case, the results will appear in the MATLAB command window.

B.1.5 Ordinary Differential Equations

A set of differential equations can be solved using the in-built commands of MATLAB, e.g., "ode45", etc. These commands require that the ODEs are written in the first-order state-space form shown in Eq. (8.126). It can be solved as follows:

```
>>%%% The following commands can be used in the command window also.
>> tspan = [0 10]; y0 = [pi/2; 0]; %Initial conditions of the state
variables.
>> [t,y] = ode45('ch8fdyn1',tspan,y0) % Calling "ode45" function
```

In the third line above, 'ch8fdyn1' is the name of a .m file containing the first-order state-space description of the equation of motion. Its contents are as follows:

```
>>%% Contents ch8fdyn1.m
>>%For one-link arm
>>function ydot  = ch8fdyn1(t,y); %Function defines the state-space
form.
>>m  =  1; a  =  2; g  =  9.81; tau = 0; % System parameters
>>iner  =  m*a*a/3; grav  =  m*g*a/2;
>>ydot = [y(2);(tau-grav*sin(y(1)))/iner]; %State-space equations
```

B.1.6 Plots

MATLAB has plot commands that provide the variation of, say, y data with time t of Section B.1.5. For example,

```
>> plot [t,y]
```

will give the plot shown in Fig. 8.20.

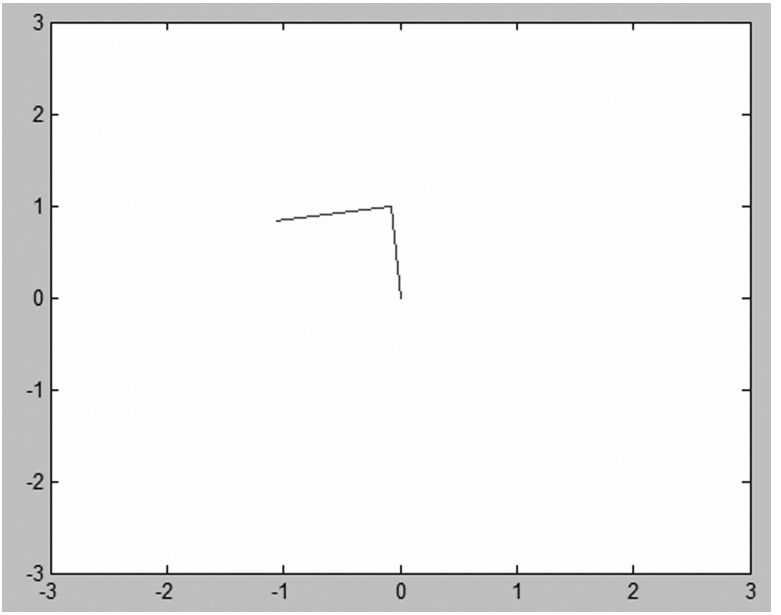
B.1.7 Animation

MATLAB provides a feature of animation also. A simple set of commands as shown in Fig. B.1(a) can generate an animation whose screenshot is shown in Fig. B.1(b).

```
>>% Animation of a 2-link robot arm
>>a1 = 1;       a2 = 1;       % Link lengths
>>t = 0:0.01:4/3;           % Time
>>xmin = -(a1+a2)*1.5;      xmax = (a1+a2)*1.5;
>>ymin = -(a1+a2)*1.5;      ymax = (a1+a2)*1.5;
>>for i = 1:1:length(t)
>>dth1 = 2*pi*45/60;   dth2 = 2*pi*45/60;     % Velocity
>>th1 = dth1*t(i);     th2 = dth2*t(i);       % Displacement
>>xy1 = [a1*cos(th1);  a1*sin(th1)];
>>xy2 = xy1+[a2*cos(th1+th2);  a2*sin(th1+th2)];
>>xy = [[0; 0] xy1 xy2];
>>plot(xy(1,:),xy(2,:))
>>axis([xmin xmax  ymin ymax]);
>>drawnow % Draws at every instant of the lines
>>end
>>%% MATLAB program ends
```

(a) MATLAB program for animation

(*Contd.*)



(b) Animation screenshot

Fig. B.1 Animation program of a two-link robot manipulator

B.2 MuPAD

MuPAD is a package for computer algebra. In the recent versions of MATLAB, MuPAD can be accessed with a simple command "mupad". A new window pops up which is the MuPAD notebook where formulas can be typed, as shown in Fig. 5.17. With the press of "Enter" button, results are displayed. Referring to Fig. B.2, some commands and syntax of MuPAD are explained below:

- Variable assignment is done with ": = " and not with " = " as in MATLAB. Thus, one must type "y: = a*x^2+b*x+c" or "a: = 10" in the MuPAD notebook. While the input command texts appear in red, the output that immediately follows the input is displayed in blue.
- Output suppression is done with ":" at the end of the command and by NOT ";" as in MATLAB. Thus, to suppress the output, one must type "y: = a*x^2+b*x+c:"
- Recall of the previous answer is done with "%" and not "ans" as done in MATLAB. Thus, typing "subs (%,x = a)" will substitute *a* for *x* in the previous answer.
- One can find the solution of the function "y" using the command "soln: = solve (y,x)."
- An expression, e.g., "f: = (a+b)^2", can be expanded using the command "g: = expand(f)."
- Using some numerical values for the function of "y" above, e.g., $a = 2$, $b = 3$, and $c = 4$, the function is plotted for the range of x values, say, $x = 1, \ldots, 10$. It is shown at the bottom of Fig. B.2.

```
y:=a*x^2+b*x+c
```

$$a\,x^2 + b\,x + c$$

```
subs(%,x=a)
```

$$a^3 + b\,a + c$$

```
soln:=solve(y,x)
```

$$\begin{cases} \left\{ -\frac{b+\sqrt{b^2-4\,a\,c}}{2\,a},\ -\frac{b-\sqrt{b^2-4\,a\,c}}{2\,a} \right\} & \text{if } a \neq 0 \\ \left\{ -\frac{c}{b} \right\} & \text{if } a = 0 \wedge b \neq 0 \\ \mathbb{C} & \text{if } a = 0 \wedge b = 0 \wedge c = 0 \\ \varnothing & \text{if } a = 0 \wedge b = 0 \wedge c \neq 0 \end{cases}$$

```
f:=(a+b)^2
```

$$(a+b)^2$$

```
g:=expand(f)
```

$$a^2 + 2\,a\,b + b^2$$

```
ynum:=subs(y,a=2,b=3,c=4)
```

$$2\,x^2 + 3\,x + 4$$

```
fplot:=plot(ynum,x=1..10)
```

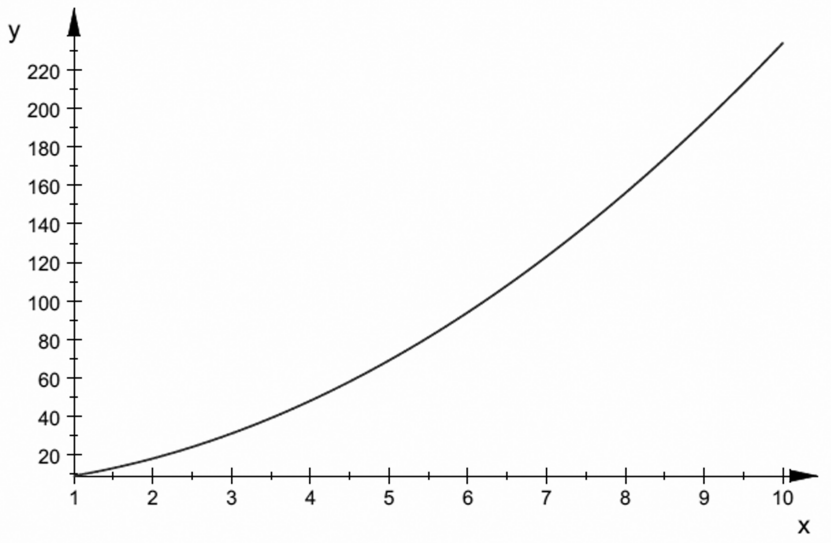


Fig. B.2 MuPAD screenshot

B.3 SIMULINK

It is a toolbox used to model a system in block-diagram form. Once the MATLAB window is open upon typing "simulink," a new window pops up with the name of a Simulink module and elementary blocks, as shown on the left of Fig. B.3. Once a new model is selected from the drop-down "file" menu (top-left), another window pops up where one can draw several boxes by choosing elementary blocks. This is indicated with "Two systems modeled." Clicking the run button on the top of this window, analysis is performed whose results can be seen by double-clicking the "scope1" and "scope2" appearing on the right of "Two systems modeled." The plots are then visible on the right. Any system parameter which needs a change can be done by double-clicking the blocks and changing the values appearing against some variables.

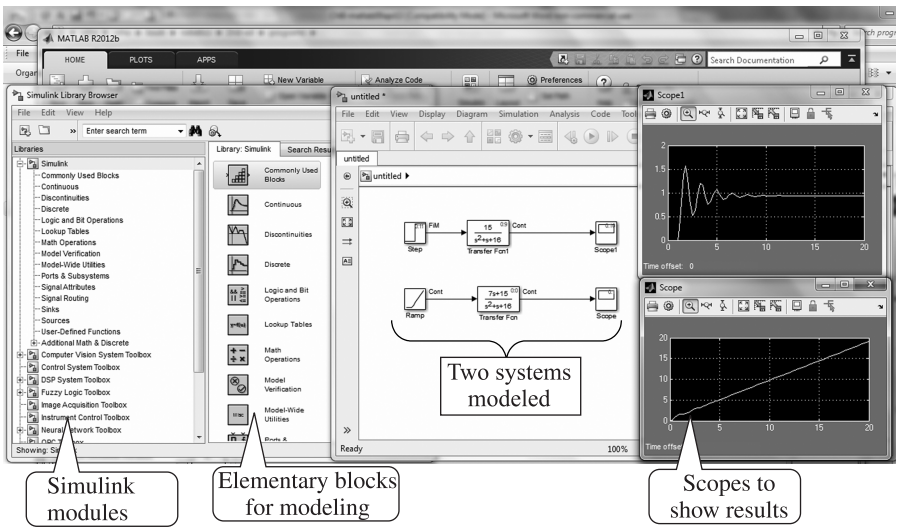


Fig. B.3 Simulink windows

B.4 ROBOTICS TOOLBOX

The Robotics Toolbox is a set of MATLAB functions for the analyses of robot kinematics, dynamics, vision, and other similar aspects (WR: Robot Toolbox). Once downloaded from the link http://www.petercorke.com/robot, one can run "start_rvc" by double-clicking on it. The window in MATLAB will be open and the toolbox is ready to use. Upon typing the following command:

```
>> rtbdemo
```

the window on the left of Fig. B.4 pops up, and different demos with their commands show up on the right. For example, a command with its answer appears as follows:

```
>> rotx(pi/2)
ans  =
     1.0000         0         0
           0    0.0000   -1.0000
           0    1.0000    0.0000
```

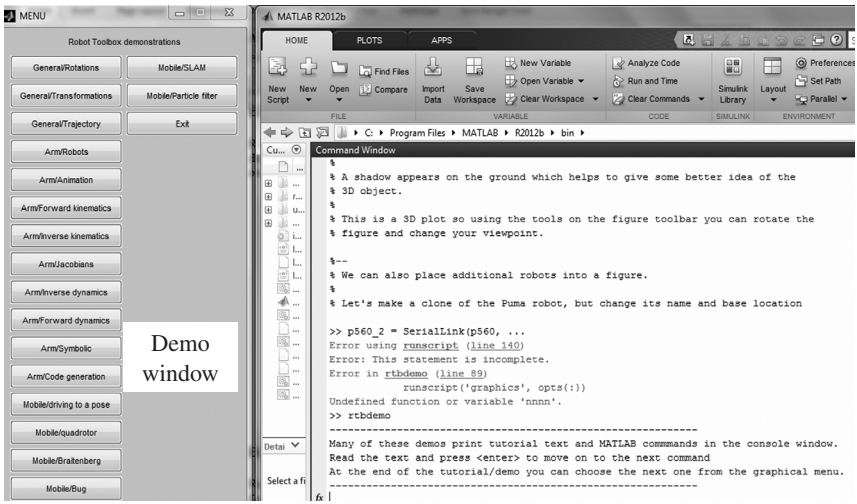


Fig. B.4 MATLAB window running Robotics Toolbox (rtbdemo)

For the forward kinematics of a PUMA robot, time and joint trajectory are generated as

```
>> mdl_puma560
>> t = [0:.05:2]';      % generate a time vector
>> q = jtraj(qz, qr, t); % generate joint coordinate trajectory
```

Upon generation of the results, the following command

```
>> p560.plot(q);
```

will show an animation as shown in Fig. B.5. Similarly, there are many more commands which can be explored and can be used to validate many of the formulas and results given in this book by putting numerical values in them.

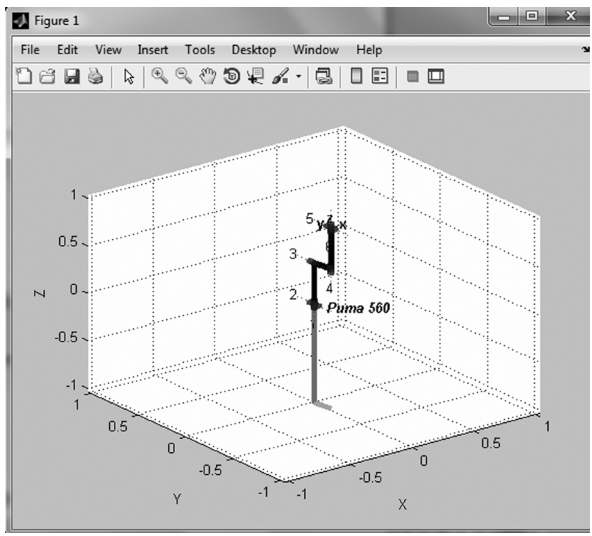


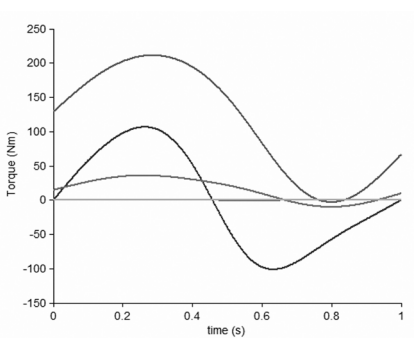
Fig. B.5 Animation screenshot generated by Robotics Toolbox

B.5 RECURSIVE DYNAMICS SIMULATOR (ReDySim)

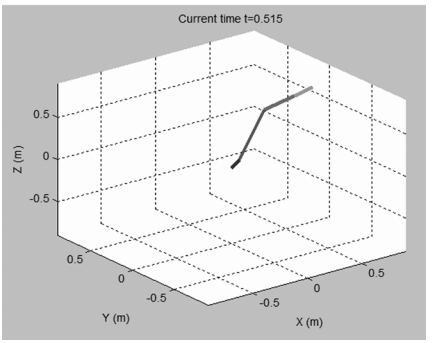
Recursive Dynamics Simulator (ReDySim) is a MATLAB-based recursive solver (Shah et al., 2013) for the dynamic analysis of robotic and multibody systems. ReDySim has the capability to incorporate any control algorithm and trajectory planner with utmost ease. This ability provides flexibility to the user/researcher in incorporating any customized algorithms. ReDySim showed considerable improvement over the commercial software and existing algorithms in terms of both computational time and numerical accuracy. ReDySim has three modules as shown in Table B.1, where the screenshot from simulation and animation results are shown in Fig. B.6.

Table B.1 Modules in ReDySim

1	Basic module: Fixed-base systems (Open- and closed-loop systems)	Three-link robot, gripper, KUKA robot, four-bar mechanism, biped, long chain, 3-RRR parallel robot and robotic leg
2	Specialized module: Floating-base Systems (a) Module for space robots (b) Module for legged robots	Three- and 7-link space robots, dual-arm space robot, biped, quadruped and hexapod legged robots
3	Symbolic module (a) Module for fixed-base systems (b) Module for floating-base systems	Fixed-base prismatic-revolute (PR)-robot, 2-link robot, KUKA robot, and floating-base 2-link space robot



(a) Joint torques (3 of them are close to zero)



(b) Animation schreenshot

Fig. B.6 Plots and animation screenshot of KULA KR-5 robot in ReDySim

SUMMARY

In this appendix, the use of MATLAB and its associated toolbox and software are explained. Very brief explanations of each of the items are outlined to get started with the software. This will help the readers use any or all of them to verify some of the examples and solve exercises given in this book.