Symbolic
Robotics
Toolbox

For use with MATLAB Release 2

Christian Paquette, Chris Harper, Roger Toogood
Dept. of Mechanical Engineering
University of Alberta
Canada

Contact: roger.toogood@ualberta.ca

June 1996

Introduction

This Toolbox contains a number of routines used in the study of robot kinematics. These routines utilize MATLAB's capability to perform symbolic computations. If you have done any kinematic analysis of robots, you will recall the drudgery and high chance of making errors in performing repetitive operations involving the 4X4 matrices that represent the transformations between robot link frames. This Toolbox is meant to act as an aide for you - it will perform all those algebraic operations, and leave you with time to think about what you are doing!

The Toolbox was inspired by the Robotics Toolbox of Peter Corke that performs numerical computations for robot kinematic and dynamic analysis. At the present time, only robot kinematics routines are implemented in this Toolbox.

The program is menu driven and largely self-explanatory. All you need to do is supply the Denavit-Hartenberg parameters for each robot link. The program has several sets of DH parameters built-in, or you can enter a set of DH parameters yourself and these can be saved to disk for later recall. Data files containing the DH parameters of several common 3 DOF robot positioning types are included in the directory containing the software. The Symbolic Toolbox will automatically form the A-matrices for the links (the notational conventions are described below). Then, via the menu system, you can direct the Toolbox to compute the forward kinematics, or portions of the inverse kinematics, or portions of the manipulator Jacobian. As each symbolic computation is carried out, you will be able to view the generated symbolic expressions on the screen (perhaps to be saved in a diary file), or substitute numerical values for the joint variables to calculate the various terms.

This write-up should provide you with the necessary information needed to operate this toolbox to its maximum capability. It contains an overall description of the Symbolic Toolbox script files. You will find a step-by-step worked and annotated example in the Appendix. You may find it useful to follow along with the example in order to fully understand the discussion of the routines involved.

Common Sense Disclaimer

Although we have made a considerable effort to verify these routines, the results should not automatically be assumed to be 100% correct. Be sure to perform some tests of your own to verify that

a) you are using the routines in the expected manner,

b) the routines are performing correctly,

and c) the routines are doing what you want (or what you think they should be doing!).

To help reduce the possibility of user-error, the menu system will guide you through steps for a typical robot kinematic solution. We have included some fail-safe checks in the code, but it is possible that you may use the software in a way that was not intended or anticipated, in which case it is likely for the program to fail.

Symbolic Toolbox Notation

As much as possible, we have tried to maintain a notational style that is consistent with common usage in the field. We generally follow the notation used in Richard Paul's book Robot Manipulators: Mathematics, Programming, and Control.

Link Parameters

Non-zero link parameters, or joint values, are represented using lower case letters and a link number :

Notation	Definition	Example
ai	length of link i	a1
di	offset of link i	d2
ti	joint angle of link i	t4

The twist of a link (alpha) is not restricted, but multiples of 90° will produce the cleanest symbolic expressions. Thus, you might see an entry similar to the following for the (1,4) element of a matrix:

```
a1 cos(t1+t2)+a2 cos(t2)
```

The program will handle both revolute and prismatic links. The nature of the joint is indicated by the parameter sigma (0 = revolute, 1 = prismatic). For prismatic joints, the value of the joint angle theta should be a multiple of 90° to produce the cleanest symbolic expressions. The actual numerical values of the link parameters are not necessary (unless you want the expressions calculated). Rather, the symbolic operations only need to know which parameters are non-zero.

Link A-matrices

The A-matrices are indicated using an upper case **A** and a link number (for example, **A1**). Links are numbered consequtively starting from the base (link 0). When you are prompted to identify an A-matrix, you only need to specify the link number.

Robot Transforms

Two types of computed robot transforms are used in the Toolbox. Transforms that arise from forward kinematic analysis are represented by Aij, where i and j are link numbers. For example, the transform **A23 = A2*A3**. When you are requested to identify one of these transforms, you use an input such as [2,3].

Transforms used in the inverse kinematics are represented by Tij. For example, for a 6 DOF robot we start with the matrix **T6** that is presumed known, but represented symbolically as

T6 = [**n o a p**]

where **n**, **o**, **a**, and **p** are 4X1 vectors. The forward solution is represented by

A16 = A1*A2*A3*A4*A5*A6

If we pre-multiply by **A1** we will get

T26 = A1^{-1}*T6

and **A26 = A2*A3*A4*A5*A6**

We have used the somewhat unusual notation for **T26** to emphasize its relation with the product **A26**, ie. these represent the same transform. If we set them equal in the inverse kinematic solution we should be able to obtain an expression involving only t1 as an unknown. Although this could be automated, we have not yet done so..

Toolbox Operation

We will now describe each of the buttons that appear on the menu. These will be roughly in the order they would appear when the program is run, so you might like to follow along with the example given in the Appendix as the various menu options are described.

To start the Toolbox, issue the command

```
robsym
```

at the Matlab command prompt. This assumes that the Toolbox is either in the default (current) directory, or on the MATLAB search path. You will be asked (via a menu) if you want to start a diary file. This file is always called `robsym.out`.

MAIN MENU BUTTONS

Here is a brief list of the commands available on the main menu. Further details are given below.

Create A Matrices	specification of DH parameters and setup of the link A-matrices. Parameters can be read from disk, entered by user, or selected from internal library of common robot geometries.
Forward Kinematics	multiplies all A-matrices of the current robot DH parameter set
Evaluate Matrices	allows substitution of numerical values in order to calculate elements in individual A-matrices, or forward kinematic transforms
Inverse Kinematics	step-by-step manipulation of A-matrices to assist with the inverse kinematic solution

List Current Variables	list all variable names in memory; this is useful to remind yourself of variable names prior to entry into some of the routines
View Elements of a Previous Matrix	view elements of any transformation matrix one element at a time or by columns; this is useful if the symbolic strings become long and scroll off the right of the screen or if the printed output in the diary file is too wide
Create or Evaluate Jacobian	evaluates the first three rows of the manipulator Jacobian (in world frame) that relate joint rates to the Cartesian velocity of the tip
QUIT	

Detailed Descriptions:

Main Menu: Create A Matrices

This button is will bring up another menu that will allow you to define the DH parameters and create the A matrices. You can do this in a number of ways:

Read DH Parms. From File	a file selection dialog box will come up to allow you to identify a previously created disk file containing the DH parameters. This file should be created using the "Enter DH Parameters" button.
Enter DH Parameters	This is used to enter your own DH parameters. The program will ask you to enter parameters in matrix form as follows: [alpha1, a1, theta1, d1, sigma1; alpha2, a2, theta2... and so on for each link ... ...] with one row for each robot link. Exact numerical values are not required for link length a or offset d. For revolute joints, although it must be given, the value entered for theta is ignored, as is d for prismatic joints. Angle constants (twist, or joint angle for prismatic joints only) should be multiples of 90° for best simplification of the symbolic expressions. The parameter sigma determines variable type (0 if revolute, 1 if prismatic). After entering all the parameters, you will be asked if you want to save the set of DH parameters in a file. If so, a file selection dialog box will open.

Mitsubishi...	predefined DH parameters for Mitsubishi RM101 robot. This is a 5 DOF all-revolute robot.
Revolute RRR	predefined parameters for a 3 DOF RRR robot
Spherical RRP	predefined dh parameters for a 3 DOF RRP robot
Cylindrical RPP	predefined dh parameters for a 3 DOF RRP robot
Cartesian PPP	predefined dh parameters for a 3 DOF PPP robot

Main Menu: Forward Kinematics

This button activates a routine that creates the forward kinematics transforms for the robot defined by the current DH parameters. If you have not previously entered the DH parameters, you will be requested to do so at this time. The program will then form all the A-matrices and compute the forward kinematic transforms Ain, i = n to 1 (ie. reverse order) where n = DOF of the robot. These will be displayed on the screen. The transform from base to tip will always be **A1n**.

Main Menu: Evaluate Matrices

This button calls a submenu with two buttons:

Evaluate A Matrix	This button is used to evaluate a previously created "A" matrix in order to test it by comparing orientation and position to the original robot. You must specify the desired matrix. (eg. for matrix **A2**, enter 2). Then enter theta (in degrees), offset, and link length when asked.
Evaluate Transform	This button is used to perform the same operation as above only it is performed on a previously created transform. For transform **A13 = A1*A2*A3**, enter [1,3]. The program automatically compensates for prismatic joints.

Main Menu: View Elements of Previous Matrices

Normally, you must perform the steps "Create A Matrices", "Forward Kinematics", (and "Inverse Kinematics") before using this function. If these have not been done, the program will call the routines for setting the DH parameters and performing the forward kinematics. View options are selected using the two buttons shown below:

View Element of A Matrix	Choose this button if you want to take a close look at a previous "A" matrix. The screen will prompt you to enter which one. Enter the number of the matrix. (eg. enter 2 to see matrix **A2**). Then see **FORMAT** below.

View Element of Transform	Choose this button if you want to look at a previous transform. You can look at either a forward kinematics (Aij) or inverse kinematics (Tij) transform using one of the two buttons
From Forward Kinematics	Enter desired transform (eg. for **A13** enter [1,3]). See **FORMAT** below.
From Inverse Kinematics	For example for **T23 = A^-1*T3** and **A23 = A2*A3**, enter the numbers [2,3]. See **FORMAT** below.

FORMAT After choosing a matrix as outlined above, a menu will appear with two buttons asking for format type.

Select Row and Column	Choose this button to look at just one element of a matrix or transform. Enter by specifying location (eg. for second element in the third row, enter [3,2]. The screen will display this element.
View All	Choose this button to look at each element of a matrix or transform one by one. Every time you press a key, the screen will display the next element until all have been shown.

Main Menu: Inverse Kinematics

Normally, you must have the DH parameters and forward solution done before you enter this command. The program will look for these and call up the appropriate routines if you haven't done this already. Then, the inverse kinematics menu is shown with a button called "Multiply by inverse A matrix". The program starts with two matrices **T** (containing symbolic **n**, **o**, **a**, and **p** vectors) and the forward kinematic solution **A1n = A1*A2*A3*...*An**. Every time you press the "Multiply..." button the program will bring one matrix over to the other side. (eg. displays matrices **T1n = A1^1*T** and **A2n = A2*A3*...*An**). This continues until no more matrices are left to move. These inverse steps can be used to determine a solution to the inverse kinematics problem by element-by-element comparison of these two products. This could conceivably be automated, but has not been implemented in the current version of the toolbox. The best way to view the results of the inverse kinematics routine is by using the "View Matrices" button on the main menu.

Main Menu: Create or Evaluate Jacobian

This calls a routine to generate the manipulator Jacobian with respect to the world coordinate frame. At this time, only the first three rows of the Jacobian are produced that relate the joint rates to the Cartesian velocity of the tip. Up to 6 DOF are allowed. Options available are the following:

Enter Equations	This is the only part of the toolbox that can be used independently. It can be is used if you know the equations for the position of the tip (px, py, and pz). You will be asked to enter these equations one at a time. Enter them using t's, (t1, t2, t3...etc), to represent joint angles, d's for prismatic lengths, (d1, d2...etc), and so on. Then enter the number of variables of the robot (D.O.F.). Next the program needs to know the joint types, so enter which joints are prismatic, (eg. [1,3] if joints 1 and 3 are prismatic). Enter [0] if all joints are revolute.
Enter Previous Transform	Use this button to find the Jacobian using the expressions for px, py, and pz obtained in a previously created forward kinematic transform. This will not work unless the current DH parameters have been entered in the "Creating A Matrices" section and the forward solution has been performed. Next enter the desired transform (eg. [1,3] for **A1*A2*A3**).
Evaluate or Test Jacobian	Used to calculate the values of terms in the Jacobian matrix, and test the matrix using either joint velocities or tip velocities
Tests Using Joint Rates	This button can be used to test the Jacobian by entering joint rates in rad/s.
Test Using Tip Velocities	Same as above only enter tip velocities and receive joint rates.

APPENDIX: AN ANNOTATED EXAMPLE

This is the diary file obtained using the Toolbox. Editorial comments are inserted in **{...}**

```
robsym

=================================================================
Welcome to the Symbolic Robotics Toolbox
User input and control is via the keyboard, and menus such as the
one above.  Keep your eye on both the MATLAB window and the menu
window for prompts for input.

At MAIN MENU
Creating A Matrices
Setting DH parameters for RRR Robot
```

{These were read from the internal database.
Columns are alpha, a, t, d, sigma for each link}

```
dh =
    90     0     0     1     0
     0     1     0     0     0
     0     1     0     0     0

Generating A matrices

A1 =
[cos(t1), 0,  sin(t1),  0]
[sin(t1), 0, -cos(t1),  0]
[      0, 1,        0, d1]
[      0, 0,        0,  1]

A2 =
[cos(t2), -sin(t2), 0, a2*cos(t2)]
[sin(t2),  cos(t2), 0, a2*sin(t2)]
[      0,        0, 1,          0]
[      0,        0, 0,          1]

A3 =
[cos(t3), -sin(t3), 0, a3*cos(t3)]
[sin(t3),  cos(t3), 0, a3*sin(t3)]
```

```
[      0,        0, 1,          0]
[      0,        0, 0,          1]

At MAIN MENU
Forward Kinematics

A33 =
[cos(t3), -sin(t3), 0, a3*cos(t3)]
[sin(t3),  cos(t3), 0, a3*sin(t3)]
[      0,        0, 1,          0]
[      0,        0, 0,          1]

A23 =
[cos(t2+t3), -sin(t2+t3), 0, a3*cos(t2+t3)+a2*cos(t2)]
[sin(t2+t3),  cos(t2+t3), 0, a3*sin(t2+t3)+a2*sin(t2)]
[         0,           0, 1,                        0]
[         0,           0, 0,                        1]
```

{This is the forward kinematics final result. The format is awkward here - the “View/Format” commands are illustrated below.}

```
A13 =
[cos(t1)*cos(t2+t3),     -cos(t1)*sin(t2+t3),          sin(t1),
cos(t1)*(a3*cos(t2+t3)+a2*cos(t2))]
[sin(t1)*cos(t2+t3),     -sin(t1)*sin(t2+t3),     -cos(t1),
sin(t1)*(a3*cos(t2+t3)+a2*cos(t2))]
[              sin(t2+t3),                 cos(t2+t3),              0,
a3*sin(t2+t3)+a2*sin(t2)+d1]
[                 0,                      0,         0,                1]

At MAIN MENU
Viewing Matrices
Currently defined variables:
Matrices and Transforms begin with A or T
Your variables are:
A1                  A33                 i
A13                 MENU_VARIABLE       menu1
A2                  dh                  menu2
A23                 dhrow               n
```

```
A3                    fkindone

Enter Transform number (e.g., for A23=A2*A3 or T23=A^-1*T3 enter [2,3]): [1,3]
Elements given by columns.  Press any key to continue.
A13(1,1)=
                              cos(t1) cos(t2 + t3)
A13(2,1)=
                              sin(t1) cos(t2 + t3)
A13(3,1)=
                                  sin(t2 + t3)
A13(4,1)=
                                       0
A13(1,2)=
                             - cos(t1) sin(t2 + t3)
A13(2,2)=
                             - sin(t1) sin(t2 + t3)
A13(3,2)=
                                  cos(t2 + t3)
A13(4,2)=
                                       0
A13(1,3)=
                                    sin(t1)
A13(2,3)=
                                   - cos(t1)
A13(3,3)=
                                       0
A13(4,3)=
                                       0
A13(1,4)=
                     cos(t1) (a3 cos(t2 + t3) + a2 cos(t2))
A13(2,4)=
                     sin(t1) (a3 cos(t2 + t3) + a2 cos(t2))
A13(3,4)=
                       a3 sin(t2 + t3) + a2 sin(t2) + d1
A13(4,4)=
                                       1
```

```
At MAIN MENU
Evaluating A Matrices
```

{Now, we’ll substitute some numbers to get a numerical result.}

```
Enter A matrix number (e.g., for A2 enter 2): 1
Enter Joint Angle, theta1 in degrees: 30
Enter Joint Offset, d1 in meters: .5
Requested A matrix is:

A =
    0.8660         0    0.5000         0
    0.5000         0   -0.8660         0
         0    1.0000         0    0.5000
         0         0         0    1.0000

At MAIN MENU
Evaluating A Matrices
```

{Forward solution - here is where we supply numerical values for all non-zero DH parameters.}

```
Enter Transform number (e.g., for A13=A1*A2*A3 enter [1,3]): [1,3]
Enter Joint Angle, theta1 in degrees: 30
Enter Joint Offset, d1 in meters: .5
Enter Joint Angle, theta2 in degrees: 60
Enter Link Length, a2 (meters): 1
Enter Joint Angle, theta3 in degrees: -30
Enter Link Length, a3 (meters): .75
Requested Transform is:

A =
    0.7500   -0.4330    0.5000    0.9955
    0.4330   -0.2500   -0.8660    0.5748
    0.5000    0.8660         0    1.7410
         0         0         0    1.0000

At MAIN MENU
Matrices for Inverse Kinematics
This will help you do the symbolic inverse kinematics of the robot selected
It will start in the form T=A1*A2*A3*A4*A5
 and every time you multiply by the inverse A matrix
 it will become     A1^-1*T=A2*A3*A4*A5 and so on..
```

```
T13 =
[nx,ox,ax,px]
[ny,oy,ay,py]
[nz,oz,az,pz]
[ 0, 0, 0, 1]

A13 =
[cos(t1)*cos(t2+t3),     -cos(t1)*sin(t2+t3),          sin(t1),
cos(t1)*(a3*cos(t2+t3)+a2*cos(t2))]
[sin(t1)*cos(t2+t3),     -sin(t1)*sin(t2+t3),     -cos(t1),
sin(t1)*(a3*cos(t2+t3)+a2*cos(t2))]
[             sin(t2+t3),                cos(t2+t3),             0,
a3*sin(t2+t3)+a2*sin(t2)+d1]
[                 0,                      0,         0,                 1]
```

{First A matrix moved over to left side.}

```
T23 =
[cos(t1)*nx+sin(t1)*ny,     cos(t1)*ox+sin(t1)*oy,     cos(t1)*ax+sin(t1)*ay,
cos(t1)*px+sin(t1)*py]
[                    nz,                        oz,                        az,
      pz-d1]
[sin(t1)*nx-cos(t1)*ny,     sin(t1)*ox-cos(t1)*oy,     sin(t1)*ax-cos(t1)*ay,
sin(t1)*px-cos(t1)*py]
[                     0,                        0,                        0,          1]



A23 =
[cos(t2+t3), -sin(t2+t3), 0, a3*cos(t2+t3)+a2*cos(t2)]
[sin(t2+t3),  cos(t2+t3), 0, a3*sin(t2+t3)+a2*sin(t2)]
[         0,           0, 1,                        0]
[         0,           0, 0,                        1]

At MAIN MENU
Viewing Matrices
Currently defined variables:
Matrices and Transforms begin with A or T

Your variables are:
```

```
A                   c                   menu3
A1                  check               menu4
A13                 d                   menu6
A2                  dh                  menu7
A23                 dhrow               n
A3                  element             r
A33                 fkindone            t
MENU_VARIABLE       i                   tran
T13                 link_num            trans_num
T23                 menu1
a                   menu2
```

{Let's have a better look at T23. Button selected for Inverse Transforms.}

```
Enter Transform number (e.g., for A23=A2*A3 or T23=A^-1*T3 enter [2,3]): [2,3]
Elements given by columns.  Press any key to continue.
T23(1,1)=
                            cos(t1) nx + sin(t1) ny
T23(2,1)=
                                       nz
T23(3,1)=
                            sin(t1) nx - cos(t1) ny
T23(4,1)=
                                       0
T23(1,2)=
                            cos(t1) ox + sin(t1) oy
T23(2,2)=
                                       oz
T23(3,2)=
                            sin(t1) ox - cos(t1) oy
T23(4,2)=
                                       0
T23(1,3)=
                            cos(t1) ax + sin(t1) ay

T23(2,3)=
                                       az
T23(3,3)=
                            sin(t1) ax - cos(t1) ay
```

```
T23(4,3)=
                                       0
T23(1,4)=
                            cos(t1) px + sin(t1) py
T23(2,4)=
                                    pz - d1
T23(3,4)=
                            sin(t1) px - cos(t1) py
T23(4,4)=
                                       1
```

{Let's have a better look at A23. Button selected for Forward Transforms.}

```
Enter Transform number (e.g., for A23=A2*A3 or T23=A^-1*T3 enter [2,3]): [2,3]
Elements given by columns.  Press any key to continue.
A23(1,1)=
                                  cos(t2 + t3)
A23(2,1)=
                                  sin(t2 + t3)
A23(3,1)=
                                       0
A23(4,1)=
                                       0
A23(1,2)=
                                 - sin(t2 + t3)
A23(2,2)=
                                  cos(t2 + t3)
A23(3,2)=
                                       0
A23(4,2)=
                                       0
A23(1,3)=
                                       0
A23(2,3)=
                                       0
A23(3,3)=
                                       1
A23(4,3)=
                                       0
```

```
A23(1,4)=
                          a3 cos(t2 + t3) + a2 cos(t2)
A23(2,4)=
                          a3 sin(t2 + t3) + a2 sin(t2)
A23(3,4)=
                                       0
A23(4,4)=
                                       1

At MAIN MENU
Setting up the Jacobian
*** NOTE:
This Jacobian routine computes only the derivatives of
 px, py, and pz.  Up to 6 joints can be handled.

Your variables are:
A                    check                menu4
A1                   d                    menu6
A13                  dh                   menu7
A2                   dhrow                mnu
A23                  element              n
A3                   fkindone             r
A33                  i                    t
MENU_VARIABLE        l                    tran
T13                  link_num             trans_num
T23                  menu1                v
a                    menu2                var
c                    menu3
```

{This routine only utilizes the Aij matrices. In the present case, the forward kinematics that contain px, py, and pz are in matrix A13.}

```
Enter Transform number (e.g., for A1*A2*A3 enter [1,3]): [1,3]
The Jacobian is:
Elements by column. Press any key for next element
jacobian(1,1)=
                    - sin(t1) (a3 cos(t2 + t3) + a2 cos(t2))
jacobian(2,1)=
                     cos(t1) (a3 cos(t2 + t3) + a2 cos(t2))
```

```
jacobian(3,1)=
                                        0
jacobian(1,2)=
                    cos(t1) (- a3 sin(t2 + t3) - a2 sin(t2))
jacobian(2,2)=
                    sin(t1) (- a3 sin(t2 + t3) - a2 sin(t2))
jacobian(3,2)=
                          a3 cos(t2 + t3) + a2 cos(t2)
jacobian(1,3)=
                           - cos(t1) a3 sin(t2 + t3)
jacobian(2,3)=
                           - sin(t1) a3 sin(t2 + t3)
jacobian(3,3)=
                                a3 cos(t2 + t3)
```

{Obtain the numerical value of the Jacobian.}

```
Evaluating the Jacobian
Enter Joint Angle, theta1 in degrees: 30
Enter Joint Offset, d1 in meters: .5
Enter Joint Angle, theta2 in degrees: 60
Enter Link Length, a2 (meters): 1
Enter Joint Angle, theta3 in degrees: -30
Enter Link Length, a3 (meters): .75
Numerical value of Jacobian is:
   -0.5748   -1.0748   -0.3248
    0.9955   -0.6205   -0.1875
         0    1.1495    0.6495

Testing the Jacobian (known joint rates)
Enter Joint Rate, For Joint1 (rad/s): .1
Enter Joint Rate, For Joint2 (rad/s): -.2
Enter Joint Rate, For Joint3 (rad/s): .2
Using the most recently evaluated Jacobian,

 Tip Velocities are: 0.092524   0.186154    -0.1  Meters/s

Testing the Jacobian (known tip speed)
enter x tip speed   0
```

```
enter y tip speed   .5
enter z tip speed   0
Using the most recently evaluated Jacobian,

 Joint rates are: 0.37669   -0.433013   0.766346  Rads/s
At MAIN MENU
PROGRAM EXIT

 Thank You for Using Our Program

 Please wipe your feet on the way out!
```