2

Kinematics of AdeptThree Robot Arm

Adelhard Beni Rehiara
University of Papua
Indonesia

1. Introduction

Robots are very powerful elements of today's industry. They are capable of performing many different tasks and operations precisely and do not require common safety and comfort elements humans need. However, it takes much effort and many resources to make a robot function properly. Most companies that made industrial robots can be found in the market such as Adept Robotics, Staubli Robotics and Fanuc Robotics. As a result, there are many thousands of robots in industry.
An AdeptThree robot arm is a selectively compliant assembly robot arm (SCARA) manufactured by the Adept Company. In general, traditional SCARA's are 4-axis robot arms within their work envelope. They have the jointed two-link arm layout similar to our human arms and commonly used in pick-and-place, assembly, and packaging applications. As a SCARA robot, an AdeptThree robot has 4 joints which denote that it has 4 degree of freedom (DOF). The robot has been designed with completed components including operating system and programming language namely V+ (Rehiara and Smit, 2010).
In robotic, there are two important studies which are kinematics and dynamics studies. Robot kinematics is the study of robot motion without regards to the forces that result it. On the other hand, the relationship between motion, and the associated forces and torques is studied in robot dynamics. In this chapter, kinematics problem for an AdeptThree robot will be explained in detail.

2. AdeptThree robot system

An AdeptThree robot is a 4-axis SCARA robot which is designed for assembling and part-handling tasks. The body of the robot is too big compared to the most SCARAs but it has strength and rigidity to carry a load about 25 kg (55 lb) as its maximum payload. For the working envelope, it has a 1067 mm maximum radial that can make more than two meters in diameter and also it has 305 mm Z-axis stroke. Fig. 1 shows the physical system of an AdepthThree robot arm. All of the figures in this section are provided by Adept company (1991).
As manufactured by Adept Company, AdeptThree robot is designed to be compatible with the other Adept products either the Adept MC or the Adept CC controller interface. All of the control and operation of the AdeptThree robot are programmed through the selected controller. In this case, the robot is using the Adept MC controller.

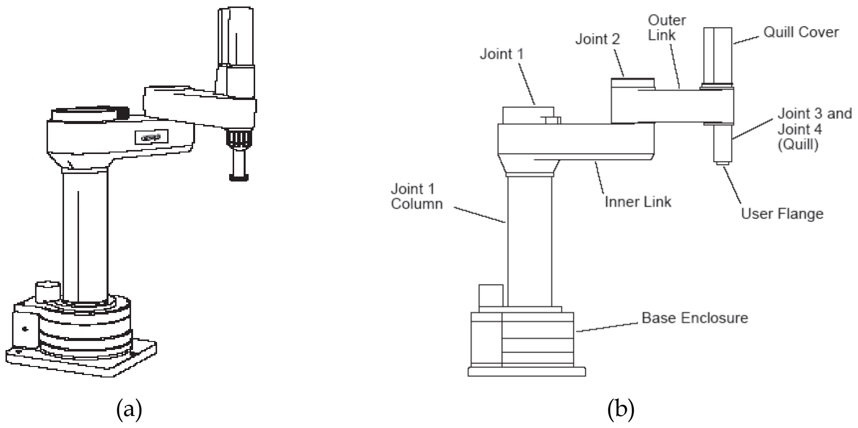


Fig. 1. (a) Physics of an AdeptThree Robot Arm and (b) Joints and links names

2.1 Joints motion

An AdeptThree robot has 4 joints which are linked to the robot. Joint 3 is a translational joint which can move along Z-axis while joint 1, 2, and 4 are rotational joints. Working envelope of the robot is shown in fig.2 (b).

First joint is the base joint and it is also called"the shoulder" as its function looks like a human shoulder. In this joint, the rotational movement of the inner link and the column will be provided. The joint has a maximum movement of about 300^0 that can be separated in 150^0 to the left and 150^0 to the right as in fig.2 (a).

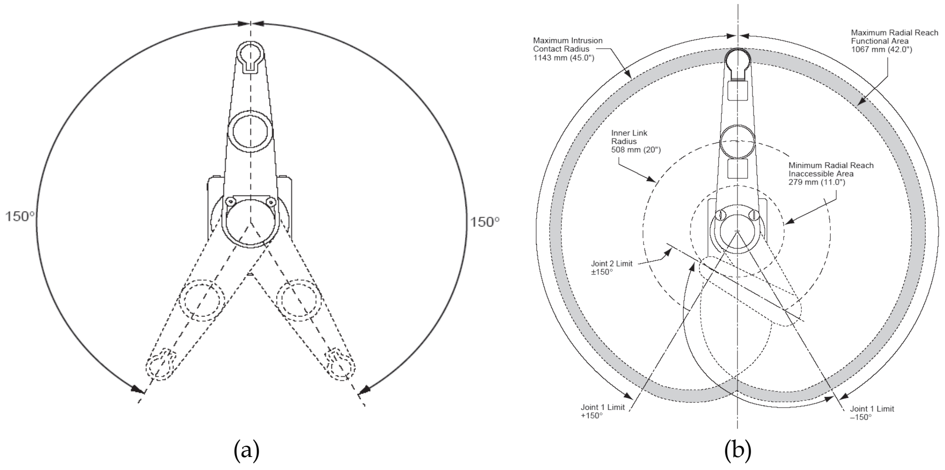
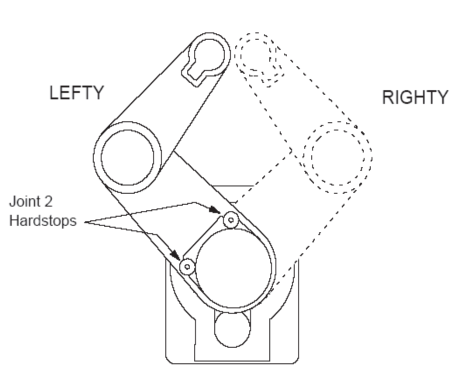
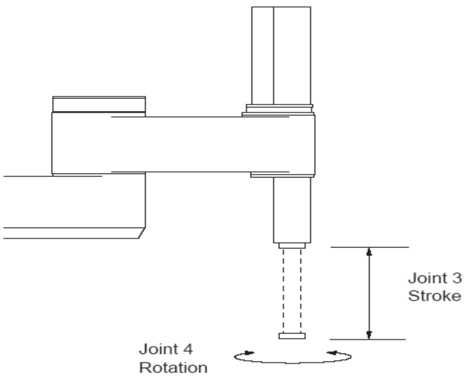

150°
150°
Maximum Intrusion Contact Radius 1143 mm (45.0")
Maximum Radial Reach Functional Area 1067 mm (42.0")
Inner Link Radius 508 mm (20")
Minimum Radial Reach Inaccessible Area 279 mm (11.0")
Joint 2 Limit ±150°
Joint 1 Limit +150°
Joint 1 Limit –150°
(a)
(b)

Fig. 2. (a) 1st joint motion and (b) working envelope

Second joint is called "the elbow" as its function looks like a human elbow. In this joint, both the outer link and inner link are linked. Furthermore this joint is similar to 1st joint, the maximum movement of the joint is also about 300^0.





(a) (b)

Fig. 3. (a) 2nd joint, (b) 3rd and 4th joint movements

Figure 3(a) shows the movement of the 2nd joint. In order to avoid any ambiguity to program the robot, the robot can be programmed to move like a human left or right arm by using the syntax “LEFTY” or “RIGHTY”.
Third joint is placed at the end of the outer link. It has a maximum stroke of about 12 inches or 30.5 cm. Fig. 3(b) shows the 3rd joint and also 4th joint.
Fourth joint is also called the "the wrist”. The joint can be moved over a range of 540^0. Its function is similar to a human wrist and it can be rotated as a human hand to tighten a bolt or unscrew a screw.
Although the AdeptThree robot has the widest working envelope, it still has a limitation. The limitation is about the travelling of each joint and it was built to avoid the damage of the robot. The maximum joint travel is confined by soft-stop and hard-stop. Soft-stop and hard stop occur when the joint is expected to pass the limit angle. While both stops happen, robot power will be turned off.
Soft-stop can be a programmed cancellation and it requires the robot arm to be moved manually into its working envelope. After the arm into the working envelope, the robot arm can be used directly without any other setting. On the other hand, the action of the hard-stop is to cancel all of the robot operations and it requires to move the robot manually by using the manual control pendant (MCP) to its working area.

2.2 Operating system

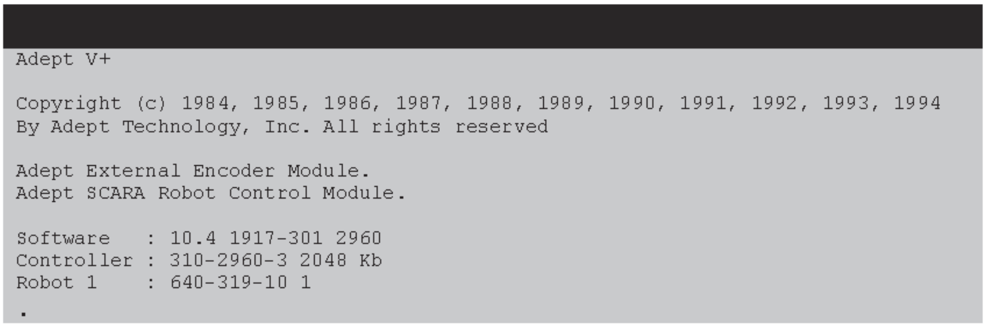
The AdeptThree robot has its own operating system called V+ that also can recognize some syntaxes in programming the robot. As an operating system, the V+ can handle all of the system operations. The programming language in the robot operating system is a high level programming language. It is similar to C or Pascal programming and it can transfer syntaxes to machine language.
The V+ real-time and multi-tasking operating system manages all system level operations, such as input/output (I/O), program execution, task management, memory management and disk file operations. As a programming language, V+ has a rich history and has evolved into the most powerful, safe and predictable, robot programming language available today. V+ is the only language to provide an integrated solution to all of the programming needs in a robotic work cell, including safety, robot motion, vision operations, force sensing and I/O.

In general, the syntaxes using by V+ can be categorized into 4 parts:

- Monitor command, it can be used directly by typing it one by one.
- Program command, it will be run if it is used in a program lines.
- Real-time command, it only can be run in a program.
- String command, it is used to handle all operations with string variable. It can be used in monitor and program command.

2.3 Robot setup

Before using the robot, it is needed to be booted by using its operation system V+. The booting screen of the Adept + is placed in fig. 4. Dot (.) command in the last line means that the robot is ready to be commanded by applying the V+ syntaxes.



```
Adept V+

Copyright (c) 1984, 1985, 1986, 1987, 1988, 1989, 1990, 1991, 1992, 1993, 1994
By Adept Technology, Inc. All rights reserved

Adept External Encoder Module.
Adept SCARA Robot Control Module.

Software   : 10.4 1917-301 2960
Controller : 310-2960-3 2048 Kb
Robot 1    : 640-319-10 1
.
```

Fig. 4. Adept V+ booting screen

As shown in fig.4, the robot consists of some modules which are software (V+ version 10.4), controller module and a robot arm. Unlike most computers, the controller does not have BIOS (basic input output system) memory; therefore the robot time needs to be changed with the actual time every time after tuned on.

3. Kinematics

Kinematics in robotics is a statement form about geometrical description of a robot structure. From the geometrical equation we can get relationship among joints spatial geometry concept on a robot with ordinary co-ordinate concept which is used to determine the position of an object. In other word, kinematics is the relationships between the positions, velocities, and accelerations of the links of a robot arm. The aim of kinematics is to define position relative of a frame to its original coordinates.

Using kinematics model, a programmer can determine the configuration of input reference that should be fed to every actuator so that the robot can do coincide movements of all joint to reach the desired position. On the other hand, with information of position that is shown by every joint while robot is doing a movement, the programmer by means of kinematics analysis can determine where is arm tip position or which parts of the robot should be moved in spatial coordinate. Kinematics problem consists of forward and inverse kinematics and each type of the kinematics has its own function as illustrated in fig.5.

From fig. 5, forward kinematics is used for transferring joint variable to get end-effector position. On the other hand, inverse kinematics will be applied to find joint variable from end-effector position.

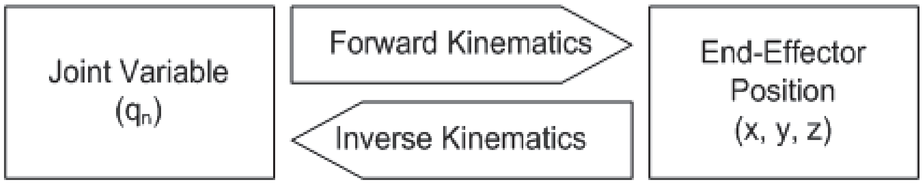


Fig. 5. Forward and inverse kinematics diagram

3.1 Forward kinematics

Forward kinematics problem is deal with finding the position and orientation of a robot end-effector as a function of its joint angles. Forward kinematics problem is relatively simple and it is easy to be implemented. There are two methods for building forward kinematics provided in this section.

3.1.1 Graphical method

A simple forward kinematics can be derived from its space using graphical solution. With a three link planar robot in fig.6, the graphical method for solving forward kinematics will be described in this section.

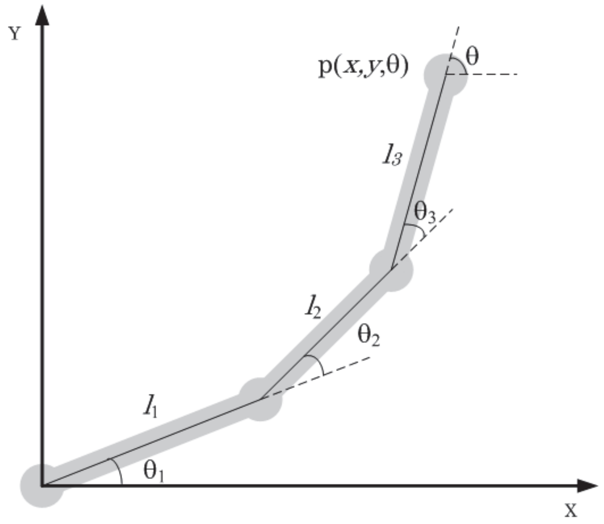


Fig. 6. Geometric of three link planar robot

Using the vector algebra solution to analyse the graph, the coordinate of the robot end-effector can be solved as follows.

$$\begin{aligned} x &= l_1\cos(\theta_1) + l_2\cos(\theta_1+\theta_2) + l_3\cos(\theta_1+\theta_2+\theta_3) \\ y &= l_1\sin(\theta_1) + l_2\sin(\theta_1+\theta_2) + l_3\sin(\theta_1+\theta_2+\theta_3) \\ \theta &= \theta_1+\theta_2+\theta_3 \end{aligned} \tag{1}$$

Maple is mathematical software which is widely used in computation, modelling and simulation. In each section of the kinematics and Jacobean, the script of the software is provided. The Maple script for building forward kinematics using the graphical method is listed as follows.

```
> restart:
> n:=3:y:=0:c:=0:
> for i from 1 to n do
> for j from i to n do
> c:=c+theta[j];
> end do;
> y:=y+l[n-i+1]*cos(c):c:=0:
> end do;
  ...
```

3.1.2 D-H convention

The steps to get the position in using D-H convention are finding the Denavid-Hartenberg (D-H) parameters, building A matrices, and calculating T matrix with the coordinate position which is desired.

D-H Parameters

D-H notation is a method of assigning coordinate frames to the different joints of a robotic manipulator. The method involves determining four parameters to build a complete homogeneous transformation matrix. These parameters are the twist angle *ai* , link length *ai*, link offset *di* , and joint angle *θi* (Jaydev, 2005). Based on the manipulator geometry, two of the parameters which are αi and *a*i have constant values, while the di and θi parameters can be variable depending on whether the joint is prismatic or revolute.
Jaydev (2005) has provided 10 steps to denote the systematic derivation of the D-H parameters as :

1. Label each axis in the manipulator with a number starting from 1 as the base to n as the end-effector. Every joint must have an axis assigned to it.
2. Set up a coordinate frame for each joint. Starting with the base joint, set up a right handed coordinate frame for each joint. For a rotational joint, the axis of rotation for axis *i* is always along Z_{i-1}. If the joint is a prismatic joint, Z_{i-1} should point in the direction of translation.
3. The X_i axis should always point away from the Z_{i-1}axis.
4. Y_i should be directed such that a right-handed orthonormal coordinate frame is created.
5. For the next joint, if it is not the end-effector frame, steps 2–4 should be repeated.
6. For the end-effector, the Z_n axis should point in the direction of the end-effector approach.
7. Joint angle θ_i is the rotation about Z_{i-1} to make X_{i-1} parallel to X_i.
8. Twist angle α_i is the rotation about Xi axis to make Z_{i-1}parallel to Z_i.
9. Link length a_i is the perpendicular distance between axis *i* and axis *i + 1*.
10. Link offset d_i is the offset along the Z_{i-1} axis.

A Matrix

The A matrix is a homogenous 4x4 transformation matrix which describe the position of a point on an object and the orientation of the object in a three dimensional space. The homogeneous transformation matrix from one frame to the next frame can be derived by the determining D-H parameters. The homogenous rotation matrix along an axis is given by

$$Rot = \left[\begin{array}{ccc|c} \cos\theta & -\cos\alpha\ \sin\theta & \sin\alpha\ \sin\theta & 0 \\ \sin\theta & \cos\alpha\ \cos\theta & -\sin\alpha\ \sin\theta & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \quad (2)$$

and the homogeneous translation matrix transforming coordinates from a frame to the next frame is given by

$$Trans = \left[\begin{array}{ccc|c} 1 & 0 & 0 & a \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \quad (3)$$

Where the four quantities θ_i, a_i, d_i, α_i are the names joint angle, link length, link offset, and twist angle respectively. These names derive from specific aspects of the geometric relationship between two coordinate frames. The four parameters are associated with link *i* and joint *i*.

In Denavit-Hartenberg convention, each homogeneous transformation matrix A_i is represented as a product of four basic transformations as follows.

$$A_i = Rot(z,\theta_i)\, Trans\,(z,d_i)\, Trans\,(x,a_i)\, Rot(x,\alpha_i) \quad (4)$$

or in completed form as

$$A_i = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & 0 \\ \sin(\theta_i) & \cos(\theta_i) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
$$\begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha_i) & -\sin(\alpha_i) & 0 \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5)$$

By simplifying equation 5, the matrix A_i which is known as D-H convention matrix is given in equation 6.

$$A_i = \left[\begin{array}{ccc|c} \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \sin\theta_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\alpha_i \cos\theta_i & -\sin\alpha_i \sin\theta_i & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \quad (6)$$

In the matrix A_i, about three of the four quantities are constant for a given link. While the other parameter which is θ_i for a revolute joint and d_i for a prismatic joint is variable for a joint. The A_i matrix contains a 3x3 rotation matrix, a 3x1 translation vector, a 1x3 perspective vector and a scaling factor. The A_i matrix can be simplified as follows.

$$A_i = \begin{bmatrix} R_{i3x3} & P_{i3x1} \\ 0_{i1x3} & 1 \end{bmatrix} \tag{7}$$

T Matrix

The T matrix is a kinematics chain of transformation. The matrix can be used to obtain coordinates of an end-effector in terms of the base link. The matrix can be built from 2 or more A matrices depending on the number of manipulator joint(s). The T matrix can be formulated as

$$T \equiv T_n = A_1 A_2, \ldots, A_n \tag{8}$$

Inside the T matrix, the direct kinematics can be found in the translation matrix P_i while the X, Y and Z positions are P_1, P_2 and P_3 respectively.

Solution for the robot

An AdeptThree robot arm with four joints is figured in fig. 7. The AdeptThree robot joint motions are revolution, revolution, prismatic and revolution (RRPR) respectively from joint 1 to 4. So the robot has four degrees of freedom.

From fig. 7, joints 1, 2, and 4 are revolute joints; then the values of θ_i are variable. Since there is no rotation about prismatic joint in joint 3, the θ_i values for joint 3 is zero while d_i is variable.

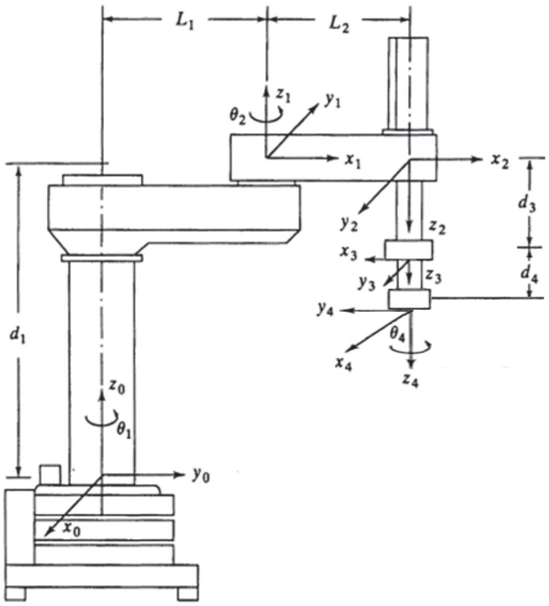


Fig. 7. Links and joints parameters of an AdeptThree robot arm

Each axis of the AdeptThree robot was numbered from 1 to 4 based on the algorithm explained before. After established coordinate frames, the next step is to determine the D-H parameters by first determining α_i. The α_i is the rotation about X_i to make Z_{i-1} parallel with Z_i. Starting from axis 1, α_1 is 0 because Z_0 and Z_1 are parallel. For axis 2, the α_2 is π or 180° because Z_2 is opposite of Z_0 which is pointing down along the translation of the prismatic joint. α_3 and α_4 values are zero because Z_3 is parallel with Z_2 and Z_4 is also parallel with Z_3.
The next step is to determine a_i and d_i. For axis 1, there is an offset d_1 between axes 1 and 2 in the Z_0 direction. There is also a distance a_1 between both axes. For axis 2, there is a distance a_2 between axes 2 and 3 away from the Z_1 axis. No offset is found in this axis so d_2 is zero. In axis 3, due to prismatic joint, the offset d_3 is variable. Between axes 3 and 4, there is an offset d_4 which is equal to this distance, while a_3 and a_4 are zero. The completed D-H parameters are listed in table 1.

Axis Number	*Joint Angle* θ_i	*Link Offset* d_i	*Link Length* a_i	*Twist Angle* α_i
1	θ_1	d_1	l_1	*0*
2	θ_2	*0*	l_2	π
3	*0*	d_3	*0*	*0*
4	θ_4	d_4	*0*	*0*

Table 1. D-H Parameters of an AdeptThree Robot

The transformation matrix A_i can now be computed. Using the expression in equation 6 the A matrices of each joint can be build as

$$A_1^0 = \begin{bmatrix} c_1 & -s_1 & 0 & l_1c_1 \\ s_1 & c_1 & 0 & l_1s_1 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{9}$$

$$A_2^1 = \begin{bmatrix} c_2 & s_2 & 0 & l_2c_2 \\ s_2 & -c_2 & 0 & l_2s_2 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{10}$$

$$A_3^2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{11}$$

$$A_4^3 = \begin{bmatrix} c_4 & -s_4 & 0 & 0 \\ s_4 & c_4 & 0 & 0 \\ 0 & 0 & 1 & d4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{12}$$

T matrix is created by multiplying each A matrix defined using equation 9 to 12 and the result is as follows.

$$T = \begin{bmatrix} s_4 s_{1+2} + c_4 c_{1+2} & s_4 c_{1+2} + c_4 s_{1+2} & 0 & l_2 c_{1+2} + l_1 c_1 \\ s_4 c_{1+2} + c_4 s_{1+2} & -s_4 s_{1+2} + c_4 c_{1+2} & 0 & l_2 s_{1+2} + l_1 s_1 \\ 0 & 0 & -1 & -d_4 - d_3 + d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{13}$$

Where c_i and s_i are the cosines and sinus of θ_i, c_{1+2} and $s_{1+2}$are $\cos(\theta_1+\theta_2)$ and $\sin(\theta_1+\theta_2)$, l_i is the length of link i and d_i is the offset of link i.
By using the T matrix, it is possible to calculate the values of (P_x, P_y, P_z) with respect to the fixed coordinate system. Then the P_x, P_y, P_z which are obtained with direct kinematics are equations which are listed below:

$$\begin{aligned} P_x &= l_2 c_{1+2} + l_1 c_1 \\ P_y &= l_2 s_{1+2} + l_1 s_1 \\ P_z &= -d_4 - d_3 + d_1 \end{aligned} \tag{14}$$

Where constant parameters l_1=559 mm, l_2=508 mm, and d_1=876.3 mm. The direct kinematics can be used to find the end-effector coordinate of the robot movement by substituting the constant parameter values to the above equation.
Maple script for the D-H convention of forward kinematics is listed as follows.

```
> restart:
> DH:=Matrix(<<theta[1],theta[2],0,theta[4]>|<d[1],0,d[3],d[4]>|<l[1],l[2],
  0, 0>|<0,pi,0,0>>):
> for i from 1 to 4 do
> A[i]:=Matrix(<<cos(DH[i,1]),sin(DH[i,1]),0,0>|<-cos(DH[i,4])*sin(DH[i,1]),
  cos(DH[i,4])*cos(DH[i,1]),sin(DH[i,4]),0>|<sin(DH[i,4])*sin(DH[i,1]),-
  sin(DH[i,4])*sin(DH[i,1]),cos(DH[i,4]),0>|<DH[i,3]*cos(DH[i,1]),DH[i,3]*si
  n(DH[i,1]),DH[i,2],1>>);
> end do:
> T:=simplify(A1.A2.A3.A4);
  ...
```

3.2 Inverse kinematics

Inverse kinematics deals with the problem of finding the appropriated joint angles to get a certain desired position and orientation of the end-effector. Finding the inverse kinematics solution for a general manipulator can be a very tricky task. In general, inverse kinematics solutions are non linear. To find those equations can be complicated and sometimes there is no solution for the problem. Geometric and algebraic methods are provided in this section for solving inverse kinematics of a robot arm.

3.2.1 Geometric method

One of the simple ways to solve the inverse kinematics problem is by using geometric solution. With this method, cosines law can be used. A two planar manipulator will be used to review this kinematics problem as in following figure.

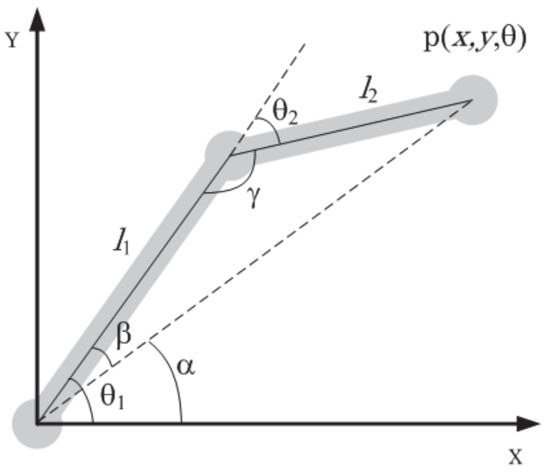


Fig. 8. Geometric of two link planar robot

With cosines law, we get

$$\left(x^2+y^2\right)=l_1^2+l_2^2-2l_1l_2\cos(180-\theta_2) \tag{15}$$

Since cos(180-θ2) = -cos(θ2) then the equation 15 will become

$$\left(x^2+y^2\right)=l_1^2+l_2^2+2l_1l_2\cos(\theta_2) \tag{16}$$

By solving the equation 16 for getting the cos(θ2),

$$\cos\theta_2=\frac{x^2+y^2-l_1^2-l_2^2}{2l_1l_2} \tag{17}$$

Therefore the θ2 will be determined by taking inverse cosines as

$$\theta_2=\arccos\left(\frac{x^2+y^2-l_1^2-l_2^2}{2l_1l_2}\right) \tag{18}$$

Again looking the fig. 8, we get

$$\frac{\sin(\beta)}{l_2}=\frac{\sin(\gamma)}{\sqrt{x^2+y^2}};\quad \alpha=\arctan\left(\frac{y}{x}\right) \tag{19}$$

Where sin(γ) = sin(180-θ2) = sin(θ2). By replacing sin(γ) with sin(θ2), the equation 19 will become

$$\beta = \arcsin\left(\frac{l_2 \sin(\theta_2)}{\sqrt{x^2+y^2}}\right) \tag{20}$$

Since $\theta_1 = \beta + \alpha$, the θ_1 can be solved as

$$\theta_1 = \arcsin\left(\frac{l_2 \sin(\theta_2)}{\sqrt{x^2+y^2}}\right) + \arctan\left(\frac{y}{x}\right) \tag{21}$$

Maple script for the geometric method of inverse kinematics is listed as follows.

```
> restart:
> beta:=solve(sin(beta)/l2=sin(theta2)/sqrt(x^2+y^2), beta):
> alpha:=arctan(y,x):
> theta1:=beta+alpha;
  ...
> theta2:=solve(y^2+x^2=l1^2+l2^2+(2*l1*l2*cos(theta)),theta);
  ...
```

3.2.2 Algebraic method

The other simple ways to solve the inverse kinematics problem is by using algebraic solution. This method is used to make an invert of forward kinematics. Rewriting the end-effector coordinate from forward kinematics:

$$\begin{aligned} x &= l_1 c_1 + l_2 c_{1+2} \\ y &= l_1 s_1 + l_2 s_{1+2} \end{aligned} \tag{22}$$

Using the square of the coordinate, we get

$$\begin{aligned} x^2 + y^2 = l_1^2 c_1^2 + l_2^2 (c_{1+2})^2 + 2 l_1 l_2 c_1 (c_{1+2}) + \\ l_1^2 s_1^2 + l_2^2 (s_{1+2})^2 + 2 l_1 l_2 s_1 (s_{1+2}) \end{aligned} \tag{23}$$

Since $\cos(a)^2+\sin(a)^2 = 1$ and also $\cos(a+b)^2+\sin(a+b)^2 = 1$, the equation 23 can be simplify as

$$x^2 + y^2 = l_1^2 + l_2^2 + 2 l_1 l_2 \left[c_1(c_{1+2}) + s_1(s_{1+2})\right] \tag{24}$$

Note that

$$\begin{aligned} \cos(a \pm b) &= \cos(a)\cos(b) \mp \sin(a)\sin(b) \\ \sin(a \pm b) &= \cos(a)\sin(b) \pm \sin(a)\cos(b) \end{aligned} \tag{25}$$

By simplifying the formulation inside the parenthesis in equation 24 with the rule in equation 25, the only left parameter is $\cos(\theta_2)$; so the equation 24 will become

$$x^2 + y^2 = l_1^2 + l_2^2 + 2 l_1 l_2 c_2 \tag{26}$$

Now the θ_2 can be formulated as the function of inverse cosines

$$\theta_2 = \arccos\left(\frac{x^2 + y^2 - l_1^2 - l_2^2}{2l_1l_2}\right) \tag{27}$$

Using the rule of sinus and cosines in equation 25, the end-effector coordinate can be rewritten as

$$\begin{aligned} x &= l_1c_1 + l_2c_1c_2 - l_2s_1s_2 \\ y &= l_1s_1 + l_2s_1c_2 + l_2c_1s_2 \end{aligned} \tag{28}$$

There are two unknown parameters inside the equation which are cos(θ1) and sin(θ1). The cos(θ1) can be defined from the rewritten x as

$$c_1 = \frac{x + l_2s_1s_2}{l_1 + l_2c_2} \tag{29}$$

The sin(θ1) is still a missing parameter and it is need to be solved. Substituting c_1 to y in equation 28, we get

$$y = \frac{x + l_2s_1s_2}{l_1 + l_2c_2}\left(l_2s_2\right) + l_1s_1 + l_2s_1c_2 \tag{30}$$

The equation 28 will become,

$$\begin{aligned} y = {} & \frac{xl_2s_2 + l_2^2s_1s_2^2}{l_1 + l_2c_2} + \frac{l_1^2s_1 + l_1l_2s_1c_2}{l_1 + l_2c_2} + \\ & \frac{l_1l_2s_1c_2 + l_2^2s_1c_2^2}{l_1 + l_2c_2} \end{aligned} \tag{31}$$

Simplifying the equation 31 we get

$$\mathrm{y} = \frac{\mathrm{xl_2s_2} + \mathrm{s_1}\left(\mathrm{l_1^2} + \mathrm{l_2^2} + 2\mathrm{l_1l_2c_2}\right)}{\mathrm{l_1} + \mathrm{l_2c_2}} \tag{32}$$

The parenthesis in equation 32 can be replaced using cosines law with $x^2 + y^2$. Therefore the sinus of θ1 can derived from the above equation as

$$y = \frac{xl_2s_2 + s_1\left(x^2 + y^2\right)}{l_1 + l_2c_2} \tag{33}$$

Now the θ1 will be got as the function of inverse sinus as

$$\theta_1 = \arcsin\left(\frac{\mathrm{y}(\mathrm{l_1} + \mathrm{l_2c_2}) - \mathrm{xl_2s_2}}{\mathrm{x^2} + \mathrm{y^2}}\right) \tag{34}$$

Until now we had defined both θ1 and θ2 of a two planar robot that is similar to the AdeptThree robot. The joint angles can be used by applying link length of the robot to the equation of those angles.

Maple script for the algebraic method of inverse kinematics is listed below.

```
> restart:
> theta2:=solve(x^2+y^2=l1^2+l2^2+(2*l1*l2*cos(theta2)),theta2);
  ...
> restart:
> cos(theta1):=solve(x=l1*cos(theta1)+l2*cos(theta1)*cos(theta2)-
  l2*sin(theta1)*sin(theta2),cos(theta1)):
> theta1:=simplify(solve(y=l1*sin(theta1)+l2*sin(theta1)*cos(theta2)+
  l2*cos(theta1)*sin(theta2),theta1));
  ...
```

4. Jacobean

The Jacobean defines the transformation between the robot hand velocity and the joint velocity. Knowing the joint velocity, the joint angles and the parameters of the arm, the Jacobean can be computed and the hand velocity calculated in terms of the hand Cartesian coordinates. The Jacobean is an important component in many robot control algorithms. Normally, a control system receives sensory information about the robot's environment, most naturally implemented using Cartesian coordinates, yet robots operate in the joint or world coordinates. Transforms are needed between Cartesian coordinates and joint coordinates and vice versa. The transformation between the velocity of the arm, in terms of its joint speeds, and the velocity of the arm in Cartesian coordinates, in a particular frame of reference, is very important. Solving the inverse kinematics can provide a transform, but this would be a difficult task to perform in real-time and in most cases no unique solutions exist for the inverse kinematics. An alternative is to use the Jacobean (Zomaya et al. , 1999).
Many ways to design a Jacobean matrix of a robot arm were provided. Zomaya et al. (1999) had presented three kinds of algorithms to perform a Jacobean matrix. First algorithm is the simple way. Without using matrix calculation, the Jacobean can be built from T matrix. Second algorithm was found to perform very well using a sequential processing method. Third algorithm is also provided to sequential machine, but it would be interesting to study how well it maps onto the mesh with multiple buses. The other algorithm was provided by Manjunath (2007) and Frank (2006). It uses tool configuration vector to perform the Jacobean. The last algorithm will be used and explained in this paper (Rehiara, 2011).
Given joint variable coordinate of the end effectors:

$$q = [q_1\ q_2\ \ldots\ q_n]^T \tag{35}$$

Where $q_i = \theta_i$ for a rotary joint and $q_i = d_i$ for a prismatic joint. Nonlinear transformation from joint variable *q(t)* to *y(t)* is defined as *y=h(q)*, then the velocities of joint axes is given by

$$\dot{y} = \frac{\partial h}{\partial q}\dot{q} = J\dot{q} \tag{36}$$

Where J is the Jacobean of manipulator. Inverse of the Jacobean J^{-1} relates the change in the end-effector to the change in axis displacements,

$$\dot{q} = J^{-1}\dot{y} \tag{37}$$

The Jacobean is not always invertible, in certain positions it will happen. These positions are called geometric singularities of the mechanism.
A rotation matrix in a T matrix is formed by three 3x1 vector. In simple, the T matrix can be rewriting as

$$T = \begin{bmatrix} n & o & a & p \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{38}$$

Where a is the approach vector of the end-effector, o is the orientation vector which is the direction specifying the orientation of the hand, from fingertip to fingertip while n is the normal vector which is chosen to complete the definition of a right-handed coordinate system (Frank, 2006).
The T matrix can be used to design the Jacobean by first defining the tool configuration vector w as follows.

$$\omega(q) = \begin{bmatrix} \dfrac{p_i}{a_i e^{(qn/\pi)}} \end{bmatrix} \tag{39}$$

Rewriting p and a vector from equation 13, we get the tool configuration vector as

$$\omega(q) = \begin{bmatrix} l_2 c_{1+2} + l_1 c_1 \\ l_2 s_{1+2} + l_1 s_1 \\ \dfrac{-d_4 - d_3 + d_1}{0} \\ 0 \\ e^{\left(\frac{q4}{\pi}\right)} \end{bmatrix} \tag{40}$$

Then the Jacobean matrix is the differential of the tool configuration vector ω as

$$J(q) = \frac{\partial w}{\partial q_i} \tag{41}$$

By taking a differentiation of the eq. 40, the Jacobean for the AdeptThree robot is defines as

$$J(q) = \begin{bmatrix} -l_1 s_1 - l_2 s_{1+2} & -l_2 s_{1+2} & 0 & 0 \\ l_1 c_1 + l_2 c_{1+2} & l_2 c_{1+2} & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \dfrac{e^{\left(q4/\pi\right)}}{\pi} \end{bmatrix} \tag{42}$$

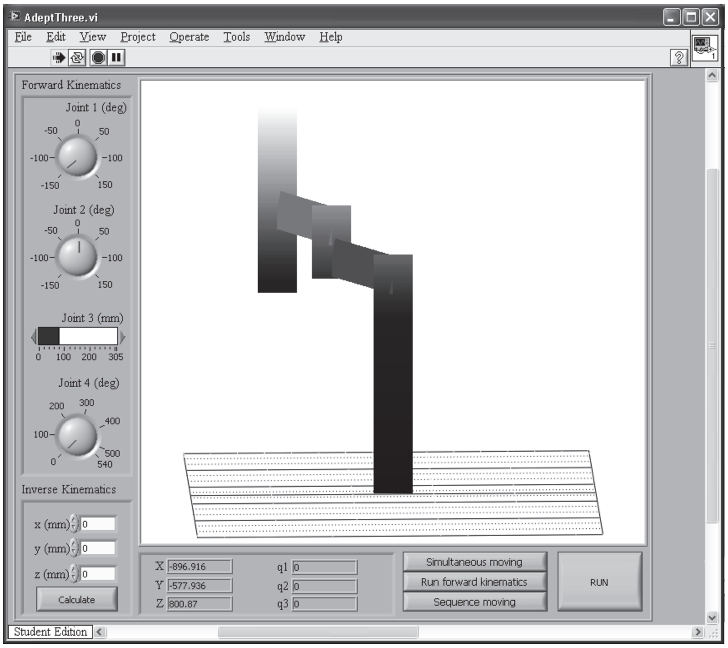
The first 3x3 matrix in the Jacobean is also called direct Jacobean. Because the Jacobean in eq. 42 is not a square matrix, it is not invertible. In this condition, the direct Jacobean can be useful since it is a square and invertible matrix.

Maple script for forming the Jacobean is listed below.

```
> restart: with(LinearAlgebra):
> q:=vector(4,[phi1,phi2,d3,phi4]):
> J:=matrix(6,4):
> w[1]:=l1*cos(phi1)+l2*cos(phi1+phi2):
> w[2]:=l1*sin(phi1)+l2*sin(phi1+phi2):
> w[3]:=-d4-d3+d1;
> w[4]:=0;
> w[5]:=0;
> w[6]:=exp(q[4]/pi);
> for i from 1 to 4 do
> for j from 1 to 6 do
> J[j,i]:=diff(w[j],q[i]);
> end do;
> end do;
> print(J);
  ...
```

5. Kinematics simulation

A Virtual Instrumentation (VI) was built to the section of kinematics simulation for supporting the manual calculation of a four DOF SCARA robot. The VI is a product of



x (mm) 0
y (mm) 0
z (mm) 0

X -896.916
Y -577.936
Z 800.87

q1 0
q2 0
q3 0

Fig. 9. SCARA robot simulation

graphical programming in LabView which is produced by National Instrumentation. The designed VI can simulate visual movement of the SCARA robot. The advantage of utilizing LabView is that the graphical programming language is easy and simple to be used. A user only needs to set each property to program the VI.
As shown in fig.9, the VI can be used to move the robot by applying the method of forward and inverse kinematics. To support the visual joint tracking, the VI is provided with simultaneous moving and sequence moving buttons. In simultaneous moving mode, each joint move together in same time. On the other hand sequence moving mode provides the motion of each joint one by one. Started from 1st joint to 4th joint, each joint will move after the other finished its task. The position of the end-effector is given in X, Y and Z boxes, while the joint variables are shown in q_1, q_2 and q_3 boxes.

6. Conclusion

This paper formulates and solves the kinematics problem for an AdeptThree robot arm. The forward kinematics of an AdeptThree robot was explained utilizing D-H convention while inverse kinematics of the robot was design using the principal cosines. Jacobean for the robot was design by using tool configuration vectors and direct Jacobean. Some script to design forward and inverse kinematics and also Jacobean matrix were provided using Maple. A graphical solution for simulating and calculating the robot kinematics was implemented in a virtual instrumentation (VI) of LabView. Using the VI, forward kinematics for a four dof SCARA robot can be simulated. Inverse kinematics for the robot can also be calculated with this VI.

7. References

[1] Jaydev P. Desai (2005). *D-H Convention*, Robot and Automation Handbook, CRC Press, USA, ISBN 0-8493-1804-1.
[2]Zomaya A.Y., Smitha H., Olariub S., Computing robot Jacobians on meshes with multiple buses, *Microprocessors and Microsystems*, no. 23, (1999), pp 309–324.
[3] Frank L.Lewis, Darren M.Dawson, Chaouki T.Abdallah (2006), *Robot Manipulators Control*, Marcel Dekker, Inc., New York.
[4] Bulent Ozkan, Kemal Ozgoren, Invalid Joint Arrangements and Actuator Related Singular Configuration of a System of two Cooperating SCARA Manipulator, *Journal of Mechatronics*, Vol.11, (2001), pp 491-507.
[5] Taylan Das M., L. Canan Dulger, Mathematical Modeling, Simulation and Experimental Verification of a SCARA Robot, *Journal of Simulation Modelling Practice and Theory*, Vol.13, (2005), pp 257-271.
[6] Mete Kalyoncu, Mustafa Tinkir (2006), Mathematical Modeling for Simulation and Control of Nonlinear Vibration of a Single Flexible Link, *Procedings of Intelligent Manufacturing Systems Symposium*, Sakarya University Turkey, May 29-31, 2006.
[7] Mustafa Nil, Ugur Yuzgec, Murat Sonmez, Bekir Cakir (2006),, Fuzzy Neural Network Based Intelligent Controller for 3-DOF Robot Manipulator *Procedings of Intelligent Manufacturing Systems Symposium*, Sakarya University Turkey, May 29-31, 2006.
[8] Rasit Koker, Cemil Oz, Tarik Cakar, Huseyin Ekiz, A Study of Neural Network Based Inverse Kinematics Solution for a Three-Joint Robot, *Journal of Robotics and Autonomous System*, Vol.49, (2004), pp 227-234

[9] Adept, (1991), *AdeptThree Robot: User's Guide*, Adept Technology, USA.

[10] Manjunath T.C., Ardil C., Development of a Jacobean Model for 4-Axes indigenously developed SCARA System, *International Journal of Computer and Information Science and Engineering*, Vol. 1 No 3, (2007), pp 152-158.

[11] John Faber Archila Diaz, Max Suell Dutra, Claudia Johana Diaz (2007), Design and Construction of a Manipulator Type Scara, Implementing a Control System, *Proceedings of COBEM, 19th International Congress of Mechanical Engineering*, November 5-9, 2007, Brasília.

[12] Rehiara Adelhard Beni, Smit Wim (2010), Controller Design of a Modeled AdeptThree Robot Arm, *Proceedings of the 2010 International Conference on Modelling, Identification and Control*, Japan, July 17-19, 2010, pp 854-858.

[13] Rehiara Adelhard Beni, System Identification Solution for Developing an AdeptThree Robot Arm Model, *Journal of Selected Areas in Robotics and Control*, February Edition, (2011), pp. 1-5 available at http://www.cyberjournals.com/Papers/Feb2011/06.pdf.