Report (Version# 3) on the project work on,

STUDY AND SIMULATION OF SEEKUR Jr – TERABOT S MANIPULATOR (5 DOF) KINEMATICS

Under the guidance of

Prof. Dr. Teresa Zielinska

Faculty of Power and Aeronautical Engineering

Done by,
EMARO 2013-15 Group,

Ashesh Vasalya,
Chandrakanth Reddy Kancharla,
David Jebaraj,
Mary Srujana,
Priti Chavan,
Ramamoorthy Luxman, &
TInu Varkey

Table of contents

S.No	Content	Pg.No
1	Cover page	0
2	Table of contents	1
3	Task A&B – Instroduction	2
4	Terabot S Manipulator drawing and Schematic	2
5	Direct/ Forward Kinematics	3
	a. Frame Assignments	3
	b. DH Parameter Table	4
	c. DKP Transformation matrix	6
6	Inverse Kinematics	7
	a. Intermediate matrices	7
7	Limiting the degree of freedom	12
	a. Choosing convenient joints for limiting the DOF	12
	b. Geometric approach	13
	c. Explanation note on the fixed orientation	17
	d. Euler's angles/ method	18
8	Verification of IKP solution by trajectory plotting	19
	a. Case 1: Linearly varying the joint angles	19
	b. Case 2: Curvelinearly varying the joint angles (Parabolic)	23
9	Matlab Simulations	28
10	VRML 3D Graphics	28
	a. VRML 3D Graphics Window	28
	b. Simulink Schematic Layout for 3D Graphics simulation	29
11	GUI	32
	a. Workspace	33
	b. Correctness menu	34
	c. DKP/ IKP menu	35
12	Annexure (Matlab Codes)	36
	a. VRML graphics code	36
	b. DKP/ IKP switch function	42
	c. Translation adjustment function	42
	d. IKP function	43
	e. GUI	46
	f. Verification in MATLAB by trajectory plotting	72
	g. DKP function	72
	h. IKP Function	72
	i. Linear trajectory	73
	k. Parabolic trajectory	75
13	Acknowledgement	78

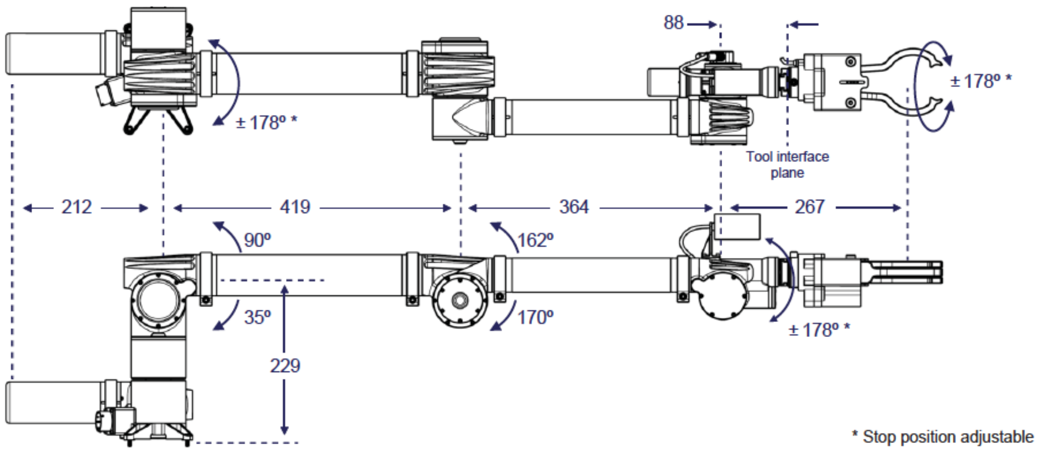
TASK: A&B

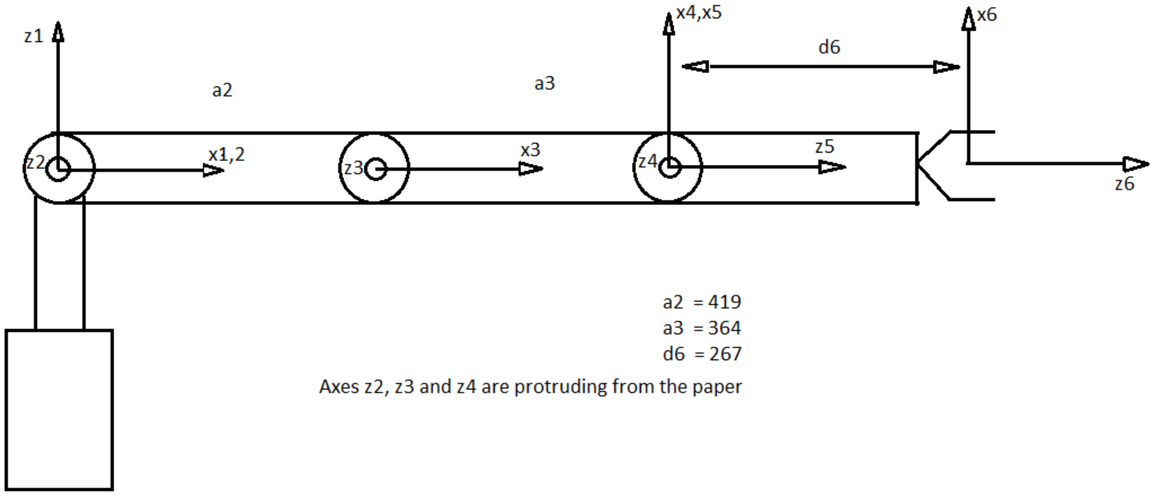
Introduction:

Kinematics is the description of motion without regard to the forces that cause it. It deals with the study of position, velocity, acceleration, and higher derivatives of the position variables.

The kinematics solutions of any robot manipulator are divided into two solutions, the first one is the solution of Forward kinematics, and the second one is the inverse kinematics solution. Forward kinematics will determine where the robot's manipulator hand will be if all joints are known. Where the inverse kinematics will calculate what each joint variable must be if the desired position and orientation of end-effector is determined. Hence, Forward kinematics is defined as transformation from joint space to Cartesian space where as Inverse kinematics is defined as transformation from Cartesian space to joint space.

TERABOT S Manipulator Drawing





Direct/Forward Kinematics :

The objective of the direct kinematics is to determine the accumulative effect that comes from the set of variables of each link, that is, to determine the position and orientation of the end-effector.

The analysis of the Direct Kinematics was made using the Denavit Hartembrerg convention. A coordinate frame is attached to each joint to determine DH parameters. Zi axis of the coordinate frame is pointing along the rotary or sliding direction of the joints. DH parameters are assigned by the following rules.

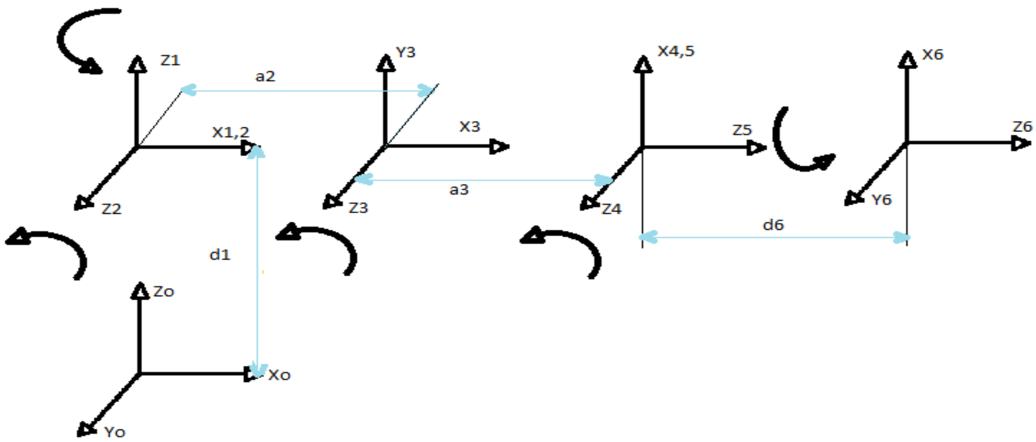
The distance from Zi-1 to Zi measured along Xi-1 is assigned as ai-1.

The angle between Zi-1 and Zi measured along Xi is assigned as αi-1.

The distance from Xi-1 to Xi measured along Zi is assigned as di.

The angle between Xi-1 to Xi measured about Zi is assigned as θi.

Frame assignment and D-H parameters for the Terabot_S Robot arm is given below:



Where,

d1 = 229; d6 = 267;

a2 =419; a3 = 364;

D-H Parameters:

	Link twist(α)	Link length(d)	Link offset(a)	Joint angle(θ)
1	0	d1	0	Θ1
2	π/2	0	0	Θ2
3	0	0	a2	Θ3
4	0	0	a3	Θ4
5	π/2	0	0	Θ5
6	0	d6	0	0

Transformation Matrices:

General transformation form for a single link from i-1 to i is as follows

$$ {}^{i-1}_{i}T = R_x(\propto_{i-1})D_x(a_{i-1})R_z(\theta_i)Q_i(d_i) $$

$$ {}^{i-1}_{i}T = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & a_{i-1} \\ \sin\theta_i\cos\alpha_{i-1} & \cos\theta_i\cos\alpha_{i-1} & -\sin\alpha_{i-1} & -d_i\sin\alpha_{i-1} \\ \sin\theta_i\sin\alpha_{i-1} & \cos\theta_i\sin\alpha_{i-1} & \sin\alpha_{i-1} & d_i\cos\alpha_{i-1} \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Where Rx and Rz represent rotation, Dx and Qi denote translation. The forward kinematics of the end-effector with respect to the base frame is determined by multiplying all of the i-1 to i matrices.

$$ {}^{base}_{end\_effector}T = {}^{0}_{1}T\,{}^{1}_{2}T \ldots {}^{n-1}_{n}T $$

Accordingly, the following matrices express the transformations matrices for our manipulator links. Notations s equivalent to sine & c equivalent to cosine are used to reduce the size of the equations.

$$ {}^{0}_{1}T = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{1}_{2}T = \begin{bmatrix} c_2 & -s_2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ S_2 & C_2 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{2}_{3}T = \begin{bmatrix} C_3 & -S_3 & 0 & a_2 \\ S_3 & C_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{3}_{4}T = \begin{bmatrix} C_4 & -S_4 & 0 & a_3 \\ S_4 & C_4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{4}_{5}T = \begin{bmatrix} C_5 & -S_5 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ S_5 & C_5 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{2}_{3}T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d6 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{0}_{n}T = \; {}^{0}_{n-1}T \; X \; {}^{n-1}_{n}T $$

Hence,

$$ {}^{0}_{2}T = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} X \begin{bmatrix} c_2 & -s_2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ s_2 & c_2 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c_1c_2 & -c_1s_2 & s_1 & 0 \\ s_1c_2 & -s_1s_2 & -c_1 & 0 \\ s_2 & c_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Similarly,

$$ {}^{0}_{3}T = \begin{bmatrix} c_1c_{23} & -c_1s_{23} & s_1 & a_2c_1c_2 \\ s_1c_{23} & -s_1s_{23} & -c_1 & a_2s_1c_2 \\ s_{23} & c_{23} & 0 & a_2s_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{0}_{4}T = \begin{bmatrix} c_1c_{234} & -c_1s_{234} & s_1 & c_1(a_3c_{23} + a_2c_2) \\ s_1c_{234} & -s_1s_{234} & -c_1 & s_1(a_3c_{23} + a_2c_2) \\ s_{234} & c_{234} & 0 & a_3s_{23} + a_2s_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{0}_{5}T = \begin{bmatrix} c_1c_{234}c_5 + s_1s_5 & -c_1s_{234} + s_1c_5 & c_1s_{234} & c_1(d_6s_{234} + a_3c_{23} + a_2c_2) \\ s_1c_{234}c_5 - c_1s_5 & -s_1c_{234}s_5 - c_1c_5 & -s_1S_{234} & s_1(a_3c_{23} + a_2c_2) \\ s_{234}c_5 & -s_{234}s_5 & -c_{234} & a_3s_{23} + a_2s_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{0}_{6}T = \begin{bmatrix} c_1c_{234}c_5 + s_1s_5 & -c_1c_{234}s_5 + s_1c_5 & c_1s_{234} & c_1(d_6s_{234} + a_3c_{23} + a_2c_2) \\ s_1c_{234}c_5 - c_1s_5 & -s_1c_{234}s_5 - c_1c_5 & -s_1s_{234} & s_1(d_6s_{234} + a_3c_{23} + a_2c_2) \\ s_{234}c_5 & -s_{234}s_5 & -c_{234} & -d_6s_{234} + a_3s_{23} + a_2s_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

This transformation matrix is the required DKP matrix of our manipulator.

From this transformation it's possible to calculate the position for a given set of joint angles.

$$ \boxed{X = c_1(d_6s_{234} + a_3c_{23} + a_2c_2)} $$

$$ \boxed{Y = s_1(d_6s_{234} + a_3c_{23} + a_2c_2)} $$

$$ \boxed{Z = -d_6s_{234} + a_3s_{23} + a_2s_2} $$

INVERSE KINEMATICS:

Tasks to be performed by a manipulator are in the Cartesian space, whereas actuators work in joint space. Cartesian space includes orientation matrix and position vector. However, joint space is represented by joint angles. The conversion of the position and orientation of a manipulator end-effector from Cartesian space to joint space is called as inverse kinematics problem. The simplest and the natural way to arrive at inverse kinematic solution is Geometric approach. We will see both geometrical approach and mathematical approach in the following.

The following process illustrates the mathematical approach for estimating the appropriate joint angles to reaching a given position in the 3 Dimensional spaces with a global reference frame.

To begin solving the inverse kinematics problem for our manipulator, we assume that the orientation and position of the end effector is known (for the desired position). Knowing the orientation and position of the end effector for the desired position, the transformation matrix ${}^6_0T$ for a given point is calculated and is expressed as

$$ {}^0_6T_d = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Intermediate matrices:

By analysis and trials, we found that the useful intermediate matrices for solving the IKP problem are ${}^1_6T$ and ${}^1_5T$. Now, we obtain these transformation matrices by the following operations.

From the assumed transformation matric T_d

$$ {}^1_0T = {}^0_1T^{-1} = \begin{bmatrix} c_1 & s_1 & 0 & 0 \\ -s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^1_6T_d = {}^1_0T \ X \ {}^0_6T $$

$$ {}^1_6T_d = \begin{bmatrix} c_1 & s_1 & 0 & 0 \\ -s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} r_{11}c_1 + r_{21}s_1 & r_{12}c_1 + r_{22}s_1 & r_{13}c_1 + r_{23}s_1 & p_xc_1 + p_y \\ -r_{11}s_1 + r_{21}c_1 & -r_{12}s_1 + r_{22}c_1 & -r_{13}s_1 + r_{23}c_1 & -p_xs_1 + p \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

.... Eq# 1

From DKP intermediate matrices

$$ {}^{1}_{3}T = \begin{bmatrix} c_{23} & -s_{23} & 0 & a_2\, c_2 \\ 0 & 0 & -1 & 0 \\ s_{23} & c_{23} & 0 & a_2\, s_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{1}_{4}T = \begin{bmatrix} c_{23} & -s_{23} & 0 & a_2\, c_2 \\ 0 & 0 & -1 & 0 \\ s_{23} & c_{23} & 0 & a_2\, s_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_4 & -s_4 & 0 & a_3 \\ s_4 & c_4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ = \begin{bmatrix} c_{234} & -s_{234} & 0 & a_3 c_{23} + a_2\, c_2 \\ 0 & 0 & -1 & 0 \\ s_{234} & c_{234} & 0 & a_3 s_{23} + a_2\, s_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{1}_{5}T = \begin{bmatrix} c_{234} & -s_{234} & 0 & a_3 c_{23} + a_2\, c_2 \\ 0 & 0 & -1 & 0 \\ s_{234} & c_{234} & 0 & a_3 s_{23} + a_2\, s_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_5 & -s_5 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s_5 & c_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ = \begin{bmatrix} c_{234} c_5 & -c_{234} s_5 & s_{234} & a_3 c_{23} + a_2\, c_2 \\ -s_5 & -c_5 & 0 & 0 \\ s_{234} c_5 & -c_{234} s_5 & -c_{234} & a_3 s_{23} + a_2\, s_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

……….Eq# 2

Now,

$$ {}^{1}_{5}T_d = {}^{1}_{6}T_d\, {}^{6}_{5}T $$

$$ {}^{6}_{5}T \;=\; {}^{5}_{6}T^{-1} \;= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

From Eq# 1

$$ {}^{1}_{5}T_d \;=\; \begin{bmatrix} r_{11}c_1 + r_{21}s_1 & r_{12}c_1 + r_{22}s_1 & r_{13}c_1 + r_{23}s_1 & p_x c_1 + p_y s_1 \\ -r_{11}s_1 + r_{21}c_1 & -r_{12}s_1 + r_{22}c_1 & -r_{13}s_1 + r_{23}c_1 & -p_x s_1 + p_y s_1 \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ = \begin{bmatrix} r_{11}c_1 + r_{21}s_1 & r_{12}c_1 + r_{22}s_1 & r_{13}c_1 + r_{23}s_1 & (p_x - d_6 r_{13})c_1 + (p_y - d_6 r_{23})s_1 \\ -r_{11}s_1 + r_{21}c_1 & -r_{12}s_1 + r_{22}c_1 & -r_{13}s_1 + r_{23}c_1 & -(p_x - d_6 r_{13})s_1 + (p_y - d_6 r_{23})c_1 \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

……..Eq# 3

Now,

Comparing elements (2,4) of Eq# 3 and Eq# 2:

$-(p_x - d_6 r_{13})s_1 + (p_y - d_6 r_{23})c_1 = 0$

$\frac{s_1}{c_1} = \frac{(p_y - d_6 r_{23})}{(p_x - d_6 r_{13})}$ hence $\Theta_1 = \arctan\left(\frac{p_y - d_6 r_{23}}{p_x - d_6 r_{13}}\right)$

Similarly comparing elements (2,3)

$-r_{13}s_1 + r_{23}c_1 = 0$

$\frac{s_1}{c_1} = \frac{r_{23}}{r_{13}}$

Hence,

$$\boxed{\mathbf{\Theta 1 = \arctan(}\frac{r_{23}}{r_{13}}\mathbf{)}}$$

Comparing elements (2,1) and (2,2)

$s_5 = r_{11}s_1 - r_{21}c_1$

$c_5 = r_{12}s_1 - r_{22}c_1$

Hence,

$\boxed{\Theta 5 = \text{Atan2}\,(r_{11}s_1 - r_{21}c_1; r_{12}s_1 - r_{22}c_1)}$

Comparing elements (1,4) and (3,4)

$a_3c_{23} + a_2\,c_2 = (p_x - d_6r_{13})s_1 + (p_y - d_6r_{23})c_1$

$a_3s_{23} + a_2\,s_2 = p_z - d_6r_{33}$

Let A = $p_x - d_6r_{13}$

B = $p_y - d_6r_{23}$

C = $p_z - d_6r_{33}$

$a_3c_{23} + a_2\,c_2 = \mathrm{A}c_1 + \mathrm{B}s_1$

$a_3s_{23} + a_2\,s_2 = \mathrm{C}$

$a_3^2c_{23}^2 + 2a_3c_{23}a_2c_2 + a_2^2c_2^2 = A^2c_1^2 + 2ABc_1s_1 + B^2s_1^2$

$a_3^2s_{23}^2 + 2a_3s_{23}a_2s_2 + a_2^2c_2^2 = \mathrm{C}^2$

$a_3^2(s_{23}^2 + c_{23}^2) + 2a_3a_2(c_{23}c_2 - s_{23}s_2) + a_2^2(c_2^2 + s_2^2) = A^2c_1^2 + 2ABc_1s_1 + B^2s_1^2 + \mathrm{C}^2$

$\mathrm{Cos}\Theta 3 = (A^2c_1^2 + 2ABc_1s_1 + B^2s_1^2 + \mathrm{C}^2 - a_3^2 - a_2^2).\frac{1}{2a_3a_2} = \mathrm{D}$

$\mathrm{Sin}\Theta 3 = \sqrt{1 - D^2}$

Hence,

$\boxed{\Theta_3 = \text{Atan2}\,(\sqrt{1 - D^2}, \mathrm{D})}$

Now to get Θ_2, polar co-ordinates will be introduced.

$a_3 s_{23} + a_2\, s_2 = p_z - d_6 r_{33}$

After expression of s_{23}

$(a_3 c_3 + a_2) s_2 + a_3 s_3 c_2 = p_z - d_6 r_{33} = C$

E $= (a_3 c_3 + a_2) s_2 = \beta c\rho$ $\qquad \beta = \sqrt{E^2 + F^2}$

F $= a_3 s_3 = \beta s\rho$ $\qquad \rho = Atan2(F, E)$

$Bc\rho \sin\Theta_2 + \beta s\rho \cos\Theta_2 = C$

$\sin(\rho + \Theta_2) = \frac{c}{\beta}$ hence, $\cos(\rho+\Theta_2) = \sqrt{1 - (\frac{c}{\beta})^2}$

$$\boxed{\boldsymbol{\Theta_2 = \mathrm{Atan2}\left(\frac{c}{\beta}, \sqrt{1 - (\frac{c}{\beta})^2}\right)}}$$

Finally, comparing elements (1,3) and (3,3)

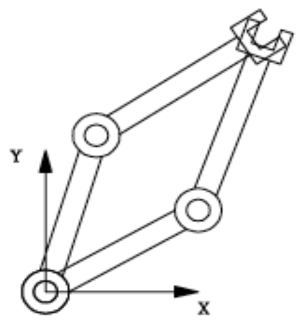
$\mathrm{Sin}(\Theta_2 + \Theta_3 + \Theta_4) = r_{13} \cos\Theta_1 + r_{23} \sin\Theta_1$

$\cos(\Theta_2 + \Theta_3 + \Theta_4) = -r_{33}$

$$\boxed{\boldsymbol{\Theta_4 = \mathrm{Atan2}(r_{13} \cos\Theta_1 + r_{23} \sin\Theta_1, -r_{33}) - \Theta_2 - \Theta_3}}$$

* Limiting the degree of freedom of Manipulator:

It is practically not possible to determine/ fix an appropriate orientation of the end effector for a given position, as there are infinite possibilities to reach a given point in a 3D space.

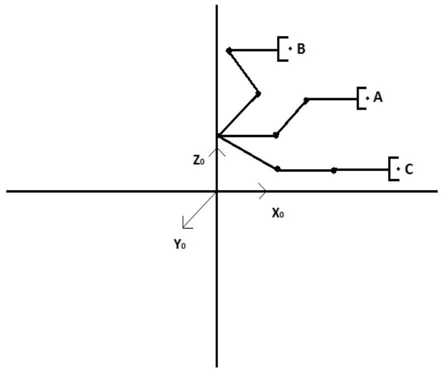


Obtaining optimal orientation is not simple

Due to this, it is not possible to determine the most appropriate orientation of the end effector without knowing each of the joint angles. In order to overcome this we propose to limit certain degrees of freedom of the manipulator, fix the orientation of the end effector and obtain the corresponding joint angles of the remaining joints.

Convenient joints – Chosen for limiting the degree of freedom:

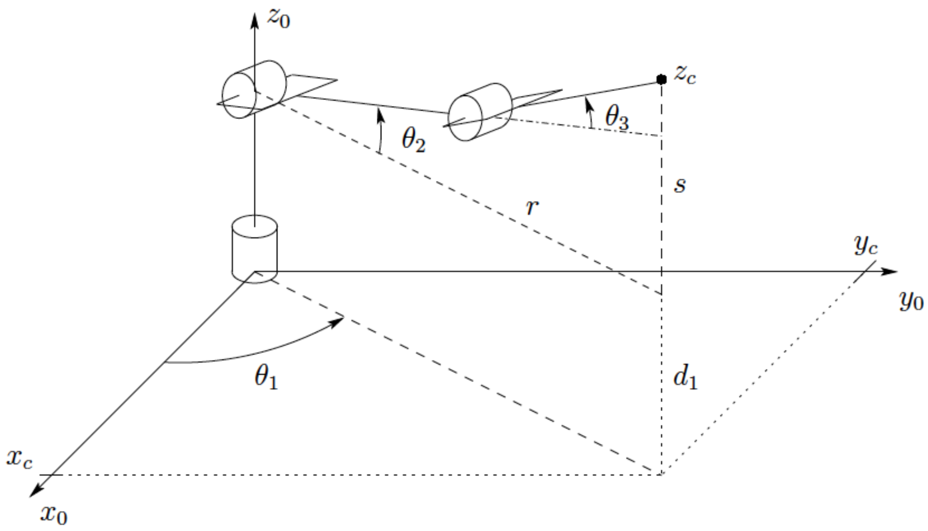
1. 5^{th} degree of freedom – By freezing the rotation of end effector to 0 radians. 5^{th} degree of freedom is removed.
2. 4^{th} degree of freedom – 4^{th} degree of freedom is not fully removed – it means that Joint# 4 is made to be not fully independent to move. Orientation of joint# 4 is made completely dependent on joint# 2 and joint# 3. The following diagram illustrates the advantage of limiting this degree of freedom.



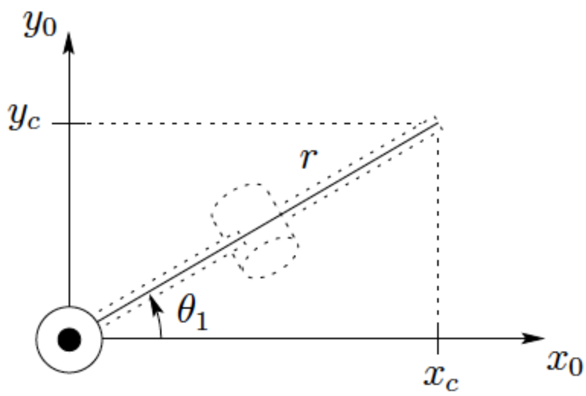
Note: Orientation of wrist for all point A, B, C remains same with respect to Joint# 1

Geometric Approach:

The general idea of the geometric approach is to solve for joint variables by projecting the manipulator's End-effector on to the planes of the base frame and solving a simple trigonometry problem. To solve for θ1, we project the arm onto the XY plane of Base frame and use trigonometry to find θ1. With the components of the End effector position being denoted by (Px,Py,Pz). We project it onto XY plane as shown in the Figure. We see from this projection that



Theta 1:



Projecting the given point on X0 - Y0 plane

$$\boldsymbol{\Theta 1 = \arctan\left(\frac{Yc}{Xc}\right)}$$

arctan(Yc, Xc) is defined for all (Xc,Yc) ≠ (0, 0) and equals the unique angle Θ such that

$$\cos\theta = \frac{x}{\sqrt{x^2+y^2}} \quad \sin\theta = \frac{y}{\sqrt{x^2+y^2}}$$

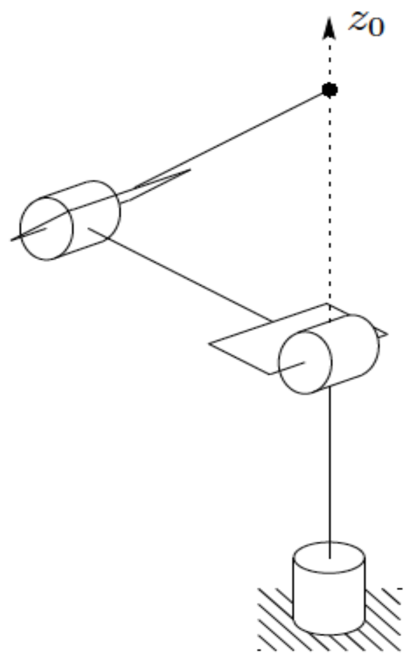
However there is always a second valid solution for the Θ1

$$\boldsymbol{\Theta 1 = \pi + \arctan\left(\frac{Yc}{Xc}\right)}$$

This will in turn, lead to different solutions for Θ2 and Θ3.

Impossibility of performing inverse kinematics for the points lying on X = 0, Y = 0 linen (Singular Configuration):

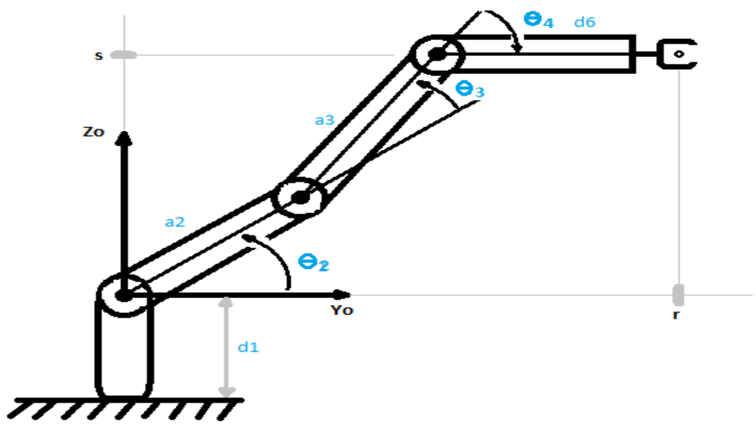
The two solutions for theta 1 are valid unless (Xc,Yc) = (0, 0). When Xc and Yc are 0, theta1 is undefined and manipulator is in a singular configuration as given below.



In this position, the given point intersects Z0, hence any value of theta1 leaves the position of the end effector fixed. Hence, our inverse kinematic model will not work for points with X = Y = 0.

Theta 2 and Theta 3:

To find the angles θ2, θ3 for the Manipulator given θ1, we consider the plane formed by the second, third and fourth links as shown in Figure below. We can apply Law of cosines to obtain the angle between link two and three.



$$\cos(\Theta_3) = \frac{t^2 + S^2 - a_2{}^2 - a_3{}^2}{2a_2a_3}$$

$$= \frac{(\sqrt{x^2+y^2} - d_6)^2 + (z-d_1)^2 - a_2{}^2a_3{}^2}{2a_2a_3} = D$$

Since, $t = r - d_6;\ r^2 = x^2 + y^2\ and\ S = z - d_1$, Θ_3 is given by

$$\boldsymbol{\Theta_3 = Atan2(D, \pm\sqrt{1 - D^2}}$$

The two solutions for θ3 correspond to the elbow-down position and elbow-up position, respectively. Similarly θ2 is given as

$$\boldsymbol{\Theta_2 = Atan2(t, S) - Atan2(a_2 + a_3c_3, a_3s_3)}$$

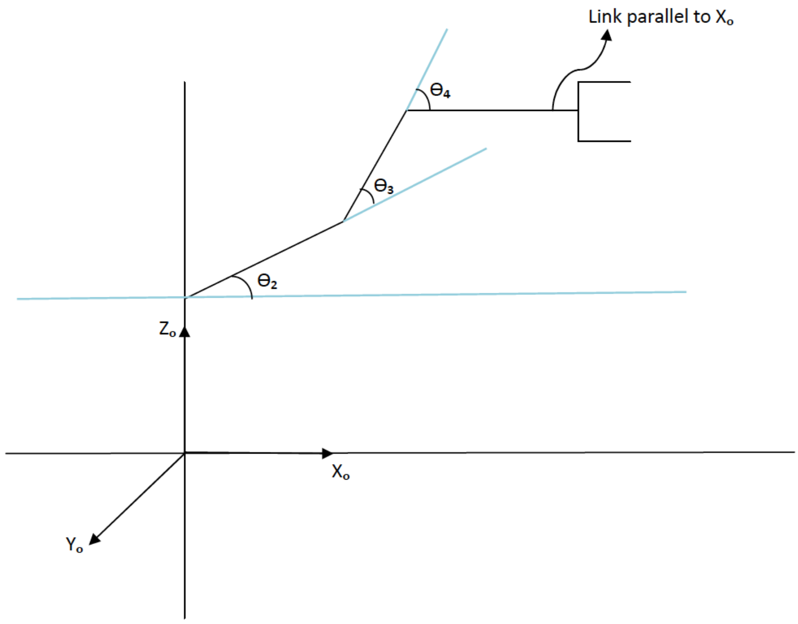
$$\boldsymbol{= Atan2\left(\sqrt{x^2+y^2}, z - d_1\right) - Atan2(a_2 + a_3c_3, a_3s_3)}$$

Theta 4 and theta 5:

As mentioned earlier, in order to simplify the inverse kinematic solution for our 5 DOF manipulator, orientation of the end effector with respect to frame 1 is kept fixed. i.e for any given point the orientation of the end effector is same with respect to frame 1.

Angle Theta 4 is chosen appropriately depending on Theta 2 and Theta 3. This means that for every set of Theta 2 and Theta 3, Theta 4 has a unique solution, which only can satisfy our proposed condition. The below given diagram illustrates the simple geometry using which Theta 4 is calculated.

This expression for Theta 4 derived below ensures the orientation of joint# 4 to be parallel with the reference frame X0 axis (i.e ground) always.



$$\Theta 4 = 0 - \Theta 2 - \Theta 3$$

Angle Theta 5 – Rotation of the wrist (Joint# 5) is frozen to 0.

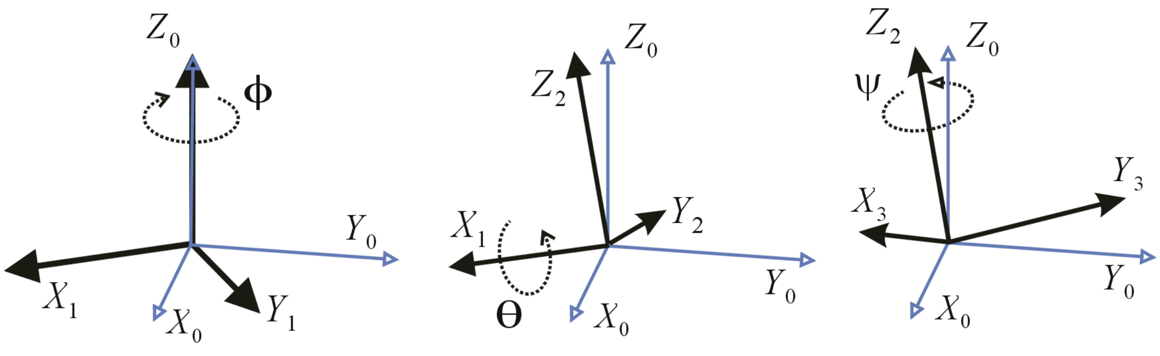
By these two conditions, it can be proven both geometrically and analytically that orientation of the end effector is always the same with respect to frame# 1.

This will now enhance us to easily determine the orientation part of the transformation matrix ${}^{1}_{6}\boldsymbol{T}$ for any given point, as it is a constant matrix – always the same for any given point. This constant orientation part of the transformation matrix ${}^{1}_{6}\boldsymbol{T}$ is given below.

$$\begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} & \boldsymbol{Px} \\ \mathbf{0} & \mathbf{-1} & \mathbf{0} & \boldsymbol{Py} \\ \mathbf{0} & \mathbf{0} & \mathbf{-1} & \boldsymbol{Pz} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}$$

Explanation note on the fixed orientation matrix:

The above matrix can be obtained by mathematically calculating the transformation matrix for any chosen arbitrary set of angles Theta 1, Theta 2 and Theta 3. However, it can also be obtained theoretically by Euler's method (ZXZ rotation) on Rotation about an arbitrary axis.



Euler's Angles for ZXZ rotation

For our constraint, i.e Link# 4 to be parallel to the reference X0 axis, it is obvious that, $\boldsymbol{\Phi}$ is $\frac{\pi}{2}$ radians, $\boldsymbol{\Theta}$ is $\boldsymbol{\pi}$ radians, and $\boldsymbol{\Psi}$ is -$\frac{\pi}{2}$ radians. For a given Yaw, Pitch and Roll angles, the rotation matrix is given by the Euler's expression given below.

$$\begin{bmatrix} \cos(\Phi)\cos(\Theta)\cos(\Psi) - \sin(\Phi)\sin(\Psi) & -\cos(\Psi)\sin(\Phi) - \cos(\Phi)\cos(\Theta)\sin(\Psi) & \cos(\Phi)\sin(\Theta \\ \cos(\Phi)\sin(\Psi) + \cos(\Theta)\cos(\Psi)\sin(\Phi) & \cos(\Phi)\cos(\Psi) - \cos(\Theta)\sin(\Phi)\sin(\Psi) & \sin(\Phi)\sin(\Theta \\ -\cos(\Psi)\sin(\Theta) & \sin(\Theta)\sin(\Psi) & \cos(\Theta) \end{bmatrix}$$

Substituting $\boldsymbol{\Phi}$, $\boldsymbol{\Phi}$, $\boldsymbol{\Phi}$ we get the transformation matrix ${}^{1}_{6}\mathbf{T}$ same as mentioned earlier.

<u>What next?</u>

Now, knowing the transformation matrix ${}^{1}_{6}\mathbf{T}$, it is easy to obtain the DKP matrix ${}^{0}_{6}\mathbf{T_d}$

$${}^{0}_{6}\mathbf{T_d} = {}^{0}_{1}\mathbf{T} \; \mathbf{X} \; {}^{1}_{6}\mathbf{T}$$

For a given point in the 3-D space, Px, Py, Pz, from the earlier expression for Θ1, is calculated by arc tan of Py/Px. Knowing the value of Theta 1 and Px, Py, Pz, we can now evaluate ${}^{0}_{1}\mathbf{T}$ by,

$${}^{0}_{1}\mathbf{T} = \begin{bmatrix} \mathbf{c_1} & -\mathbf{s_1} & \mathbf{0} & \mathbf{Px} \\ \mathbf{s_1} & \mathbf{c_1} & \mathbf{0} & \mathbf{Py} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{Pz} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}$$

Hence using this, ${}^{0}_{6}\mathbf{T_d}$ is calculated. By this, we have succeeded to obtain the ${}^{0}_{6}\mathbf{T_d}$ is the most difficult task for performing the inverse kinematics calculations. Knowing this DKP transformation matrix ${}^{0}_{6}\mathbf{T_d}$ the elements $\mathbf{r_{11}, r_{12}, r_{13}, r_{21}, r_{22}, r_{23}, r_{31}, r_{32}, r_{33}}$ are extracted.

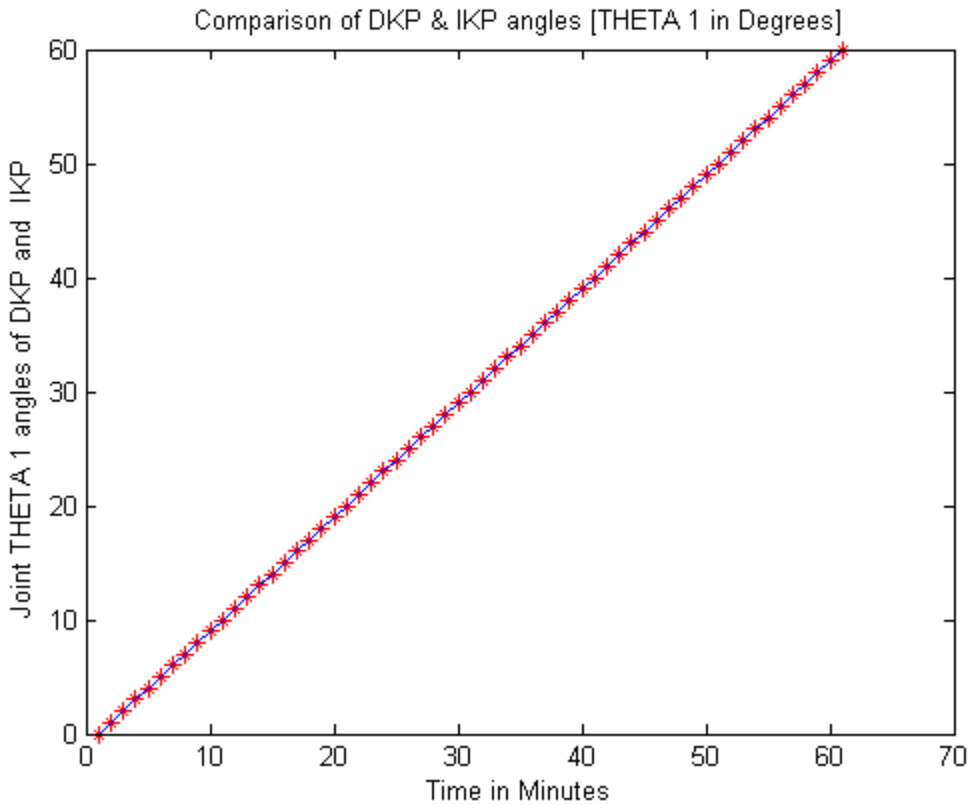
Ultimately knowing these, elements the inverse kinematic angles are calculated by the expressions as obtained earlier.

Verification of the IKP solution by trajectory plotting:

Having given the linear incremental values of Theta(1-3) from 0 to 60 Degrees for the test procedure, we Found the Direct kinematics for the position of the End-effector. And also the positions obtained from the Direct kinematics are given as the inputs to the Inverse kinematics problem at every step to obtain the Inverse kinematic angles from the formulated equations. And different plots based on the results of the DKP of any Theta(1-3) values and IKP of the corresponding DKP outputs are shown and explained below.

Case 1: Linearly varying Thetas.

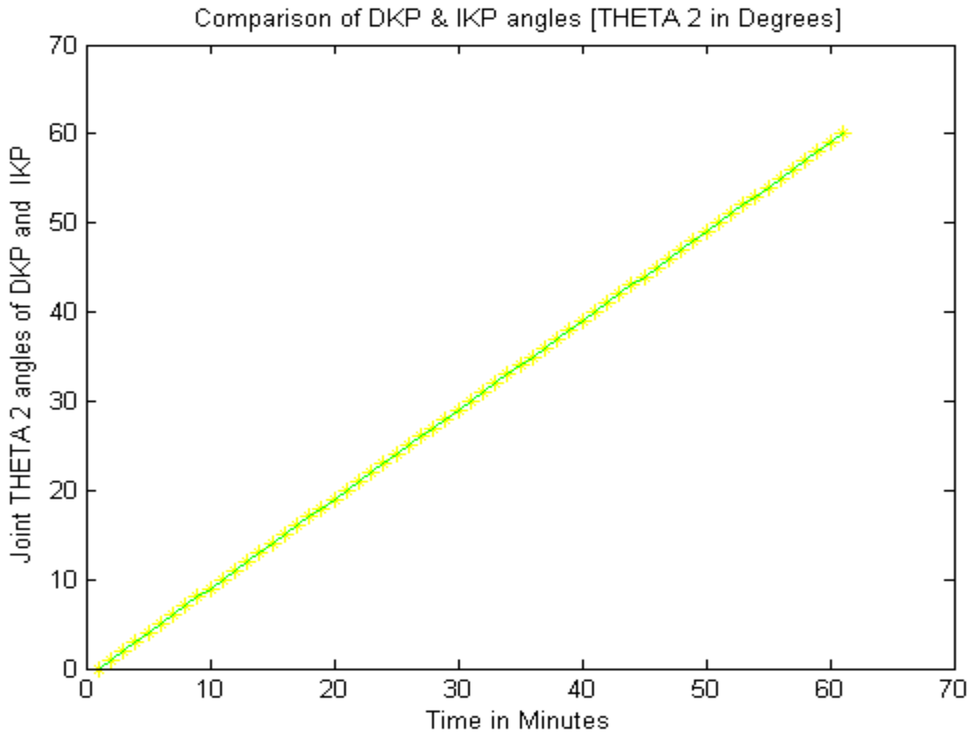
Angular difference of THETA-1:

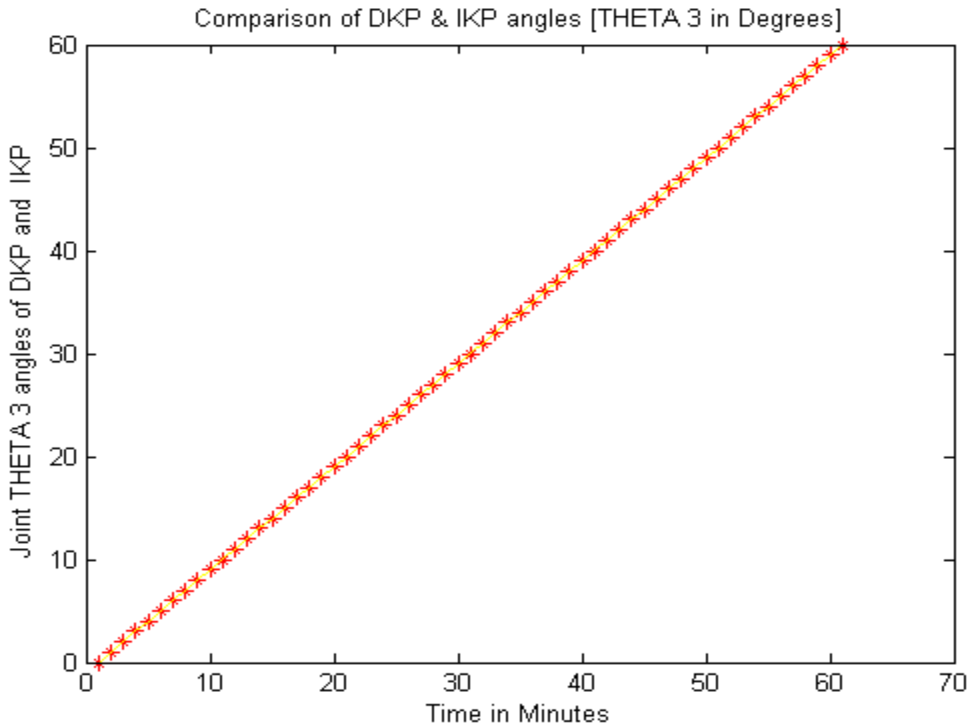


Above is the resultant plot of Both DKP and the IKP Theta-1 angles, Red stars depict DKP thetas and the Blue line IKP thetas. Thus from the plot we can see that there are no angular difference in DKP and IKP solution.

Angular difference of THETA-2,3:

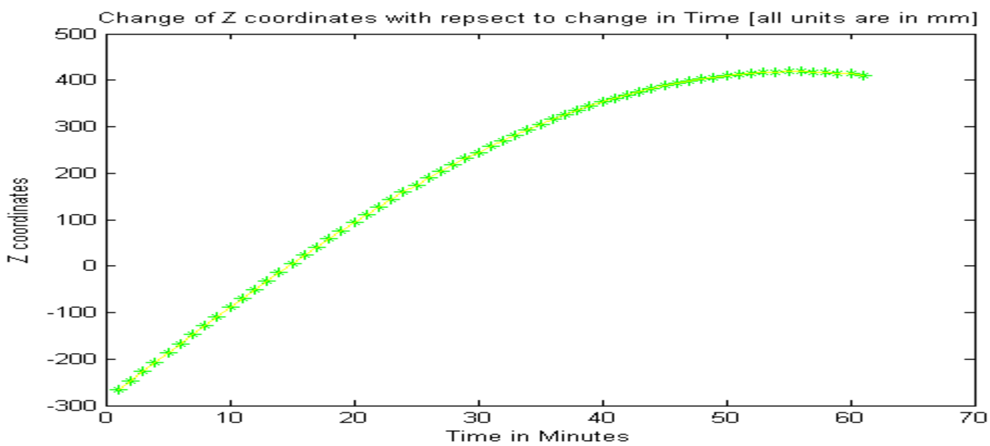
Similarly we can analyze the Theta-2 and Thet-3 for checking the error in their values and the following are the plots obtained by plotting the DKP and IKP Thetas in time domain.

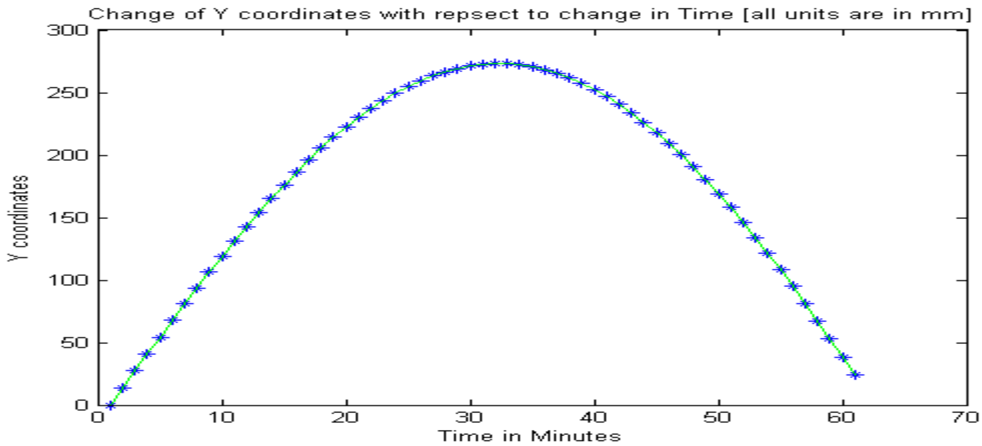


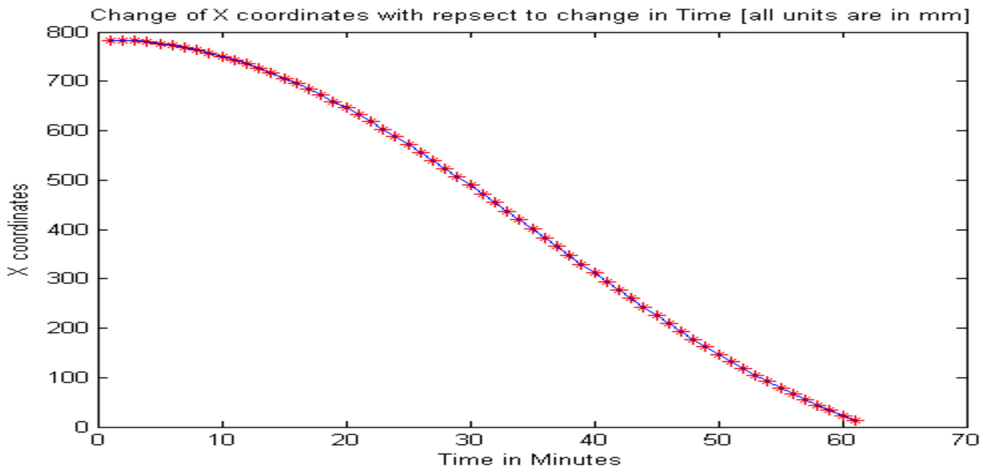


Positional Error along X,Y,Z:

We can even check the correctness of the solution by checking the position of the End-effecter when the original linearly changing Theta values are fed into DKP and the position of the End-effecter by giving the corresponding IKP Theta values obtained from the position values from DKP before into the DKP. The plots obtained from those computations are given below in Separate plots Axis wise

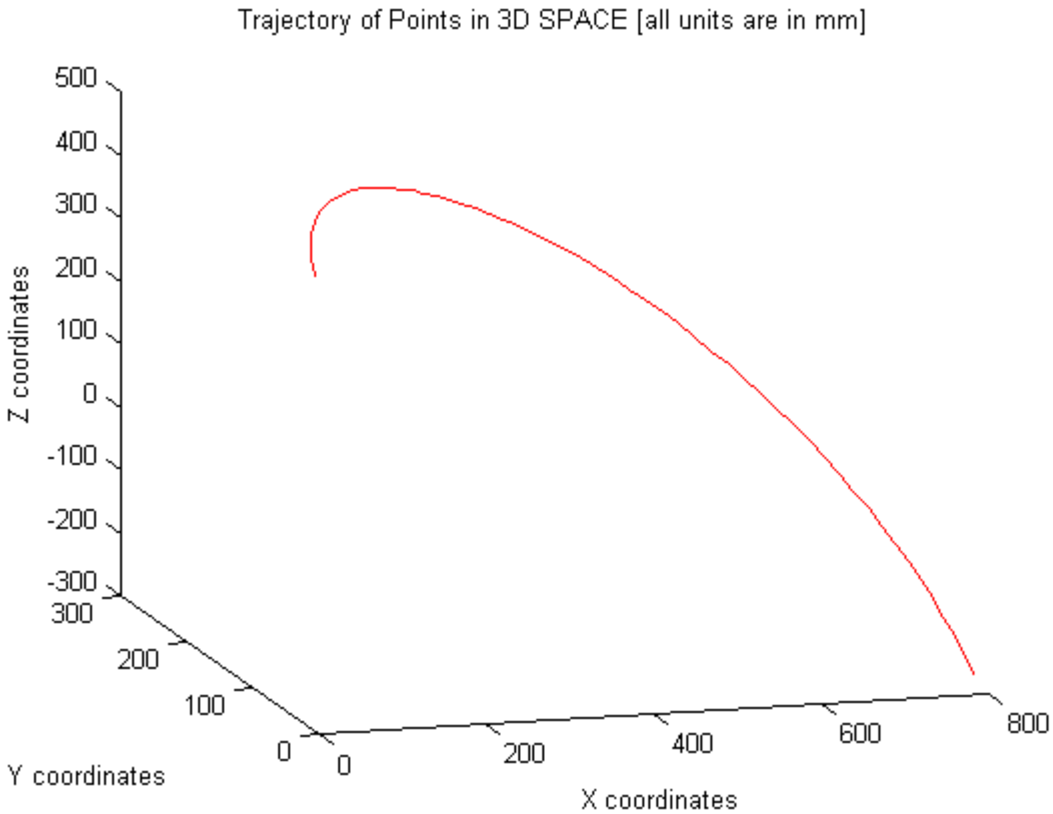






Position of the End-effecter in 3-D space:

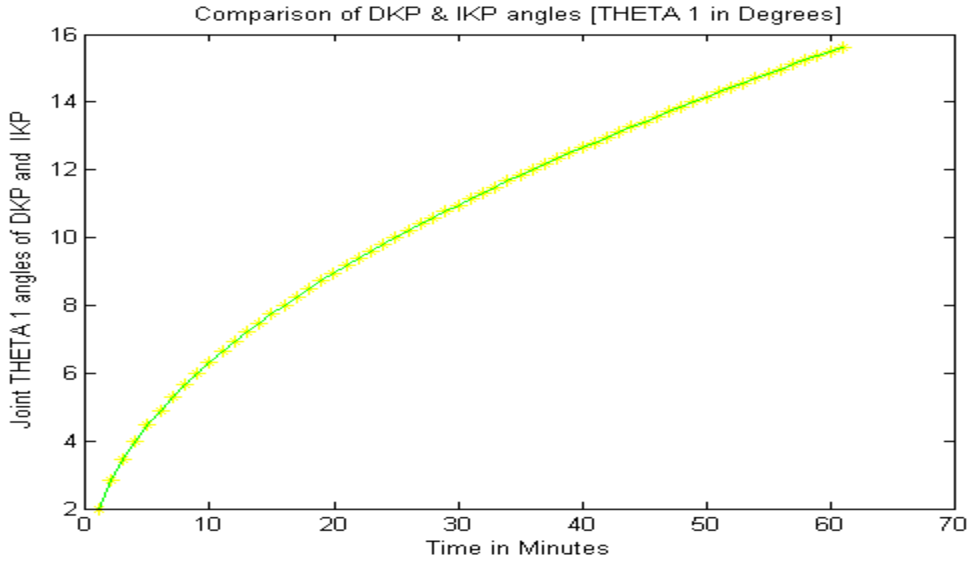
With the changing Thetas in the time domain we have different postion along X,Y,Z those different positions of the End-effecter are represented in a 3-D graph shown below. The graph describes the path which the End-effecter will travel while the Theta angels are linearly changing in time space.

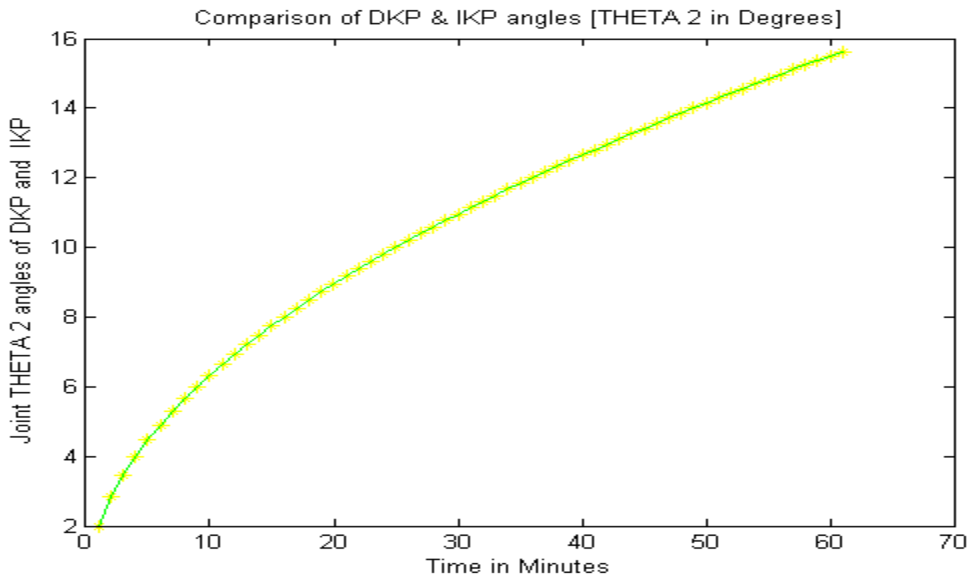


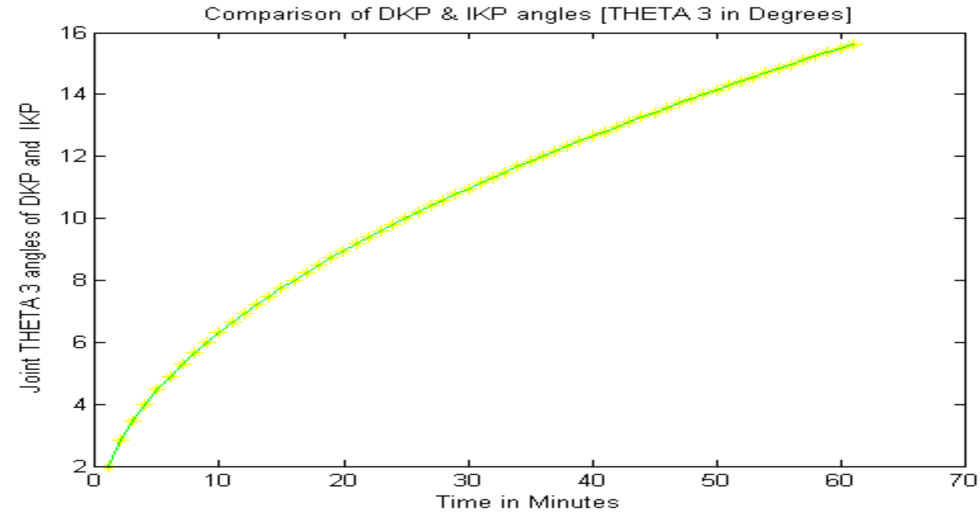
Case 2: curvelinearly varying Thetas (Parabolic).

Angular difference of THETA-1:

As mentioned in the Case-1 the plot is done between original Thetas and the Thetas obtained from the IKP equations and the resultant plot are shown below.

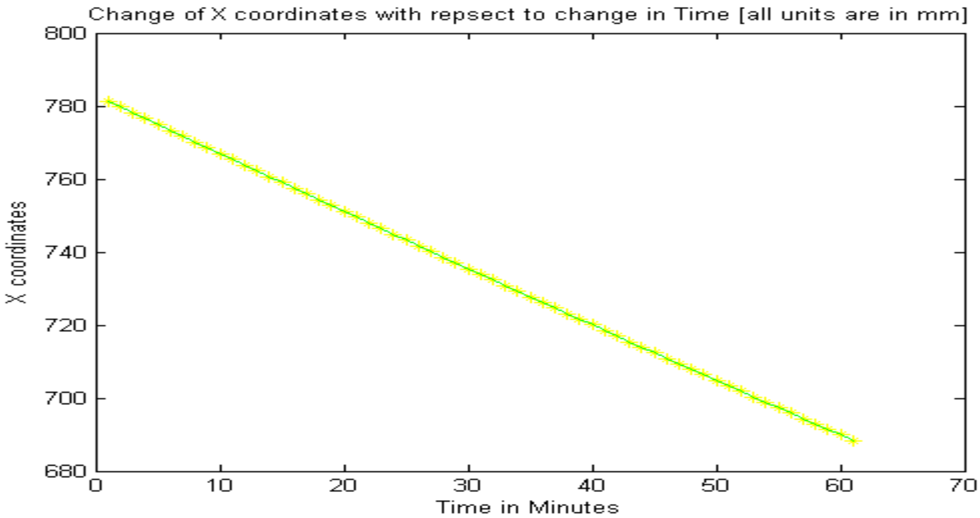

Comparison of DKP & IKP angles [THETA 1 in Degrees]
Joint THETA 1 angles of DKP and IKP
Time in Minutes
16
14
12
10
8
6
4
2
0
10
20
30
40
50
60
70

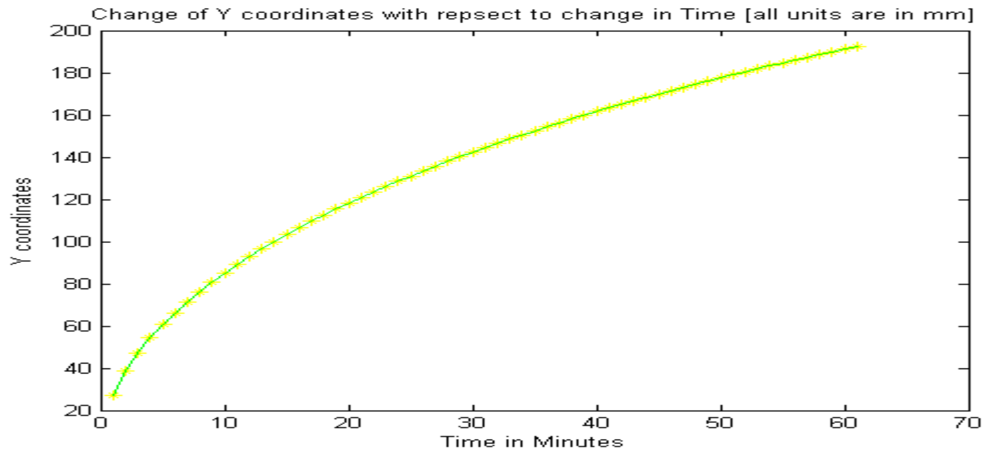

Comparison of DKP & IKP angles [THETA 2 in Degrees]
Joint THETA 2 angles of DKP and IKP
Time in Minutes
16
14
12
10
8
6
4
2
0
10
20
30
40
50
60
70

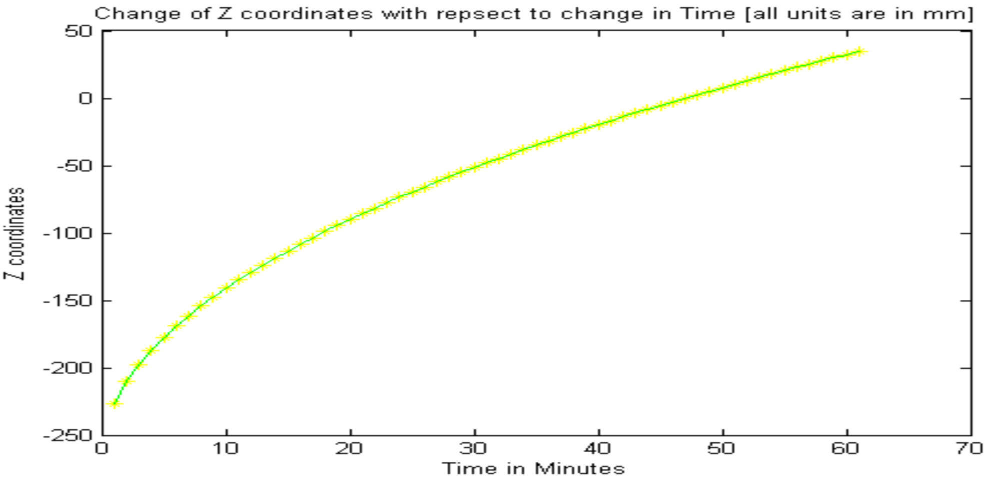


Positional Error along X,Y,Z:

As same as the case 1 we can check the correctness by both ways one is comparing the Thetas and other is comparing the X,Y,Z values of the DKP and the corresponding IKP. The following plots are the difference in the X,Y,Z positions.

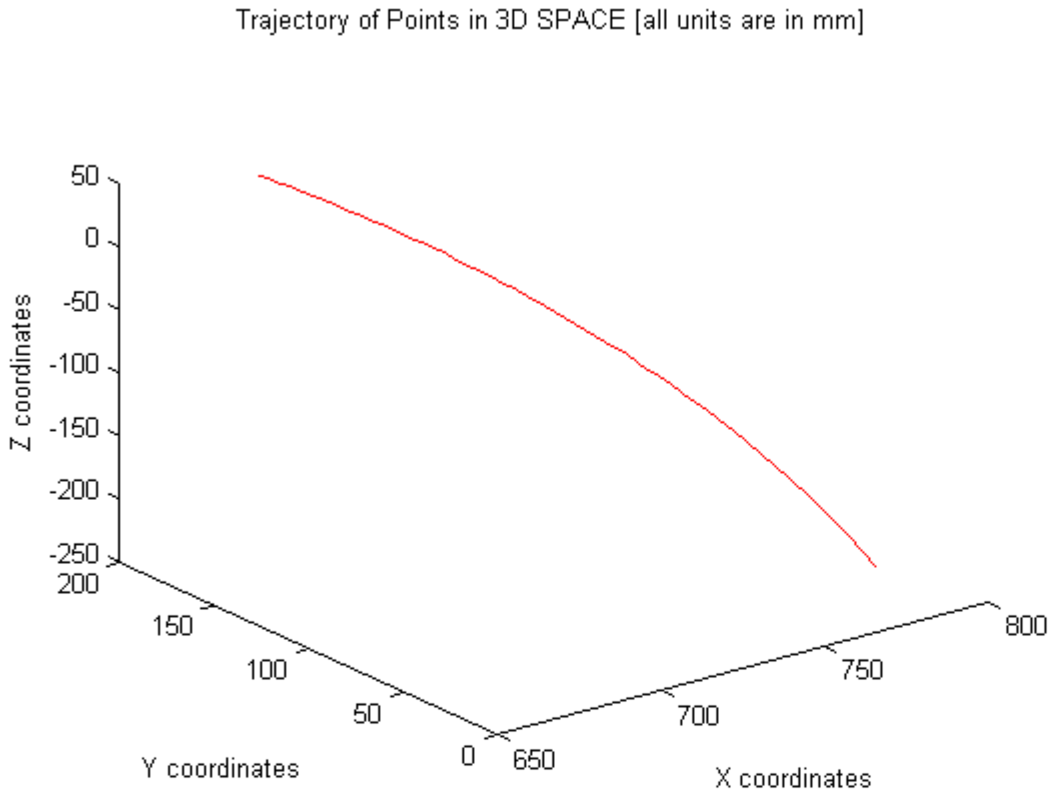






Position of the End-effecter in 3-D space:

With the changing Thetas in the time domain we have different postion along X,Y,Z those different positions of the End-effecter are represented in a 3-D graph shown below. The graph describes the path, which the End-effecter will travel while the Theta angels are curve linearly changing in time space.

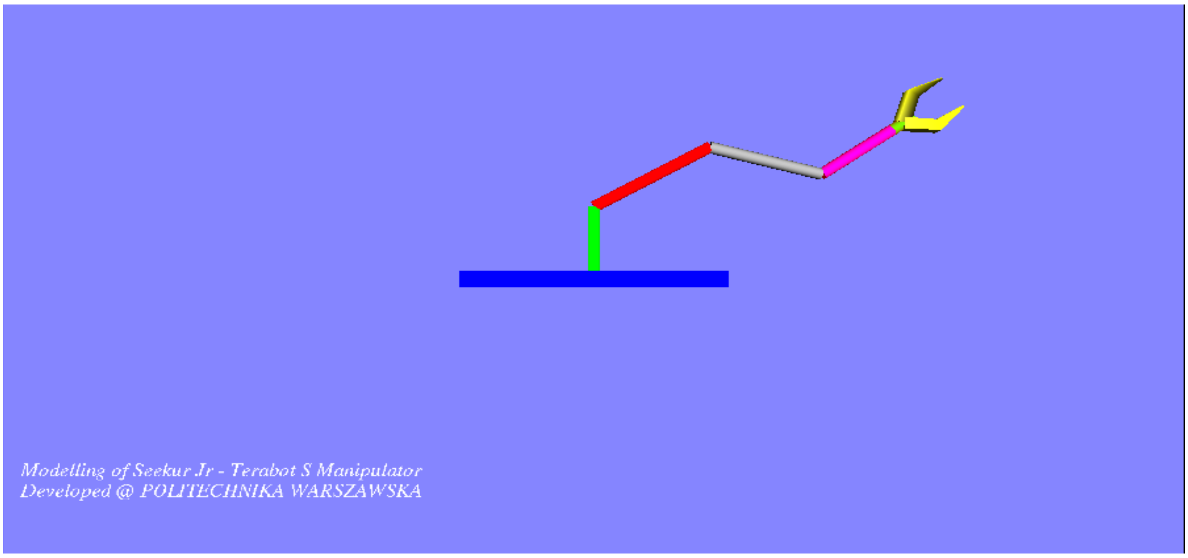

Trajectory of Points in 3D SPACE [all units are in mm]
50
0
-50
-100
-150
-200
-250
Z coordinates
200
150
100
50
0
Y coordinates
650
700
750
800
X coordinates

MATLAB Simulations

For our manipulator, the following simulation models were created using MATLAB:

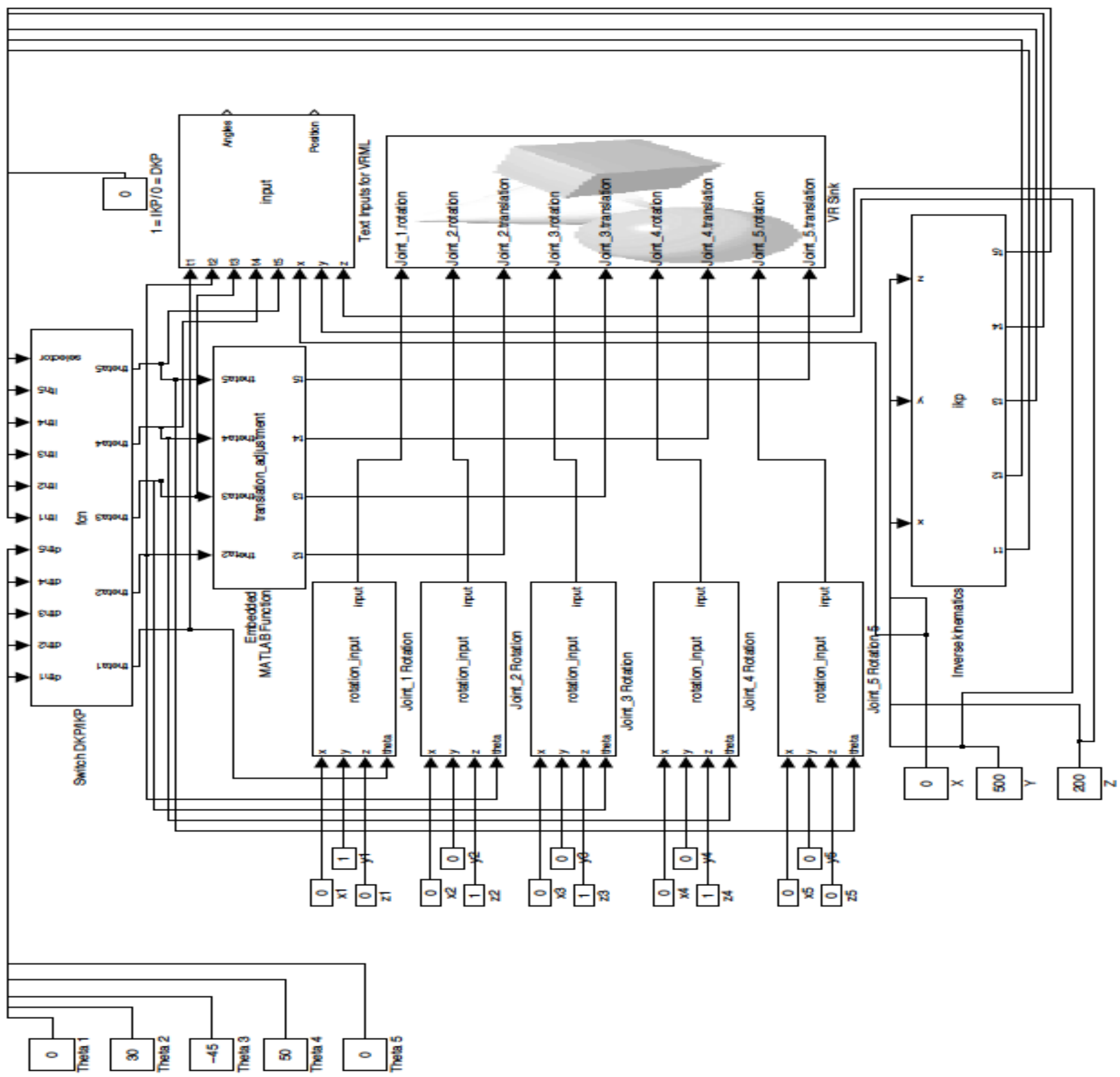
1. VRML 3-D Graphics depicting and illustrating the kinematics of the manipulator.
2. A single window GUI having options to demonstrate the following objectives
   a. Illustrating the 3 dimensional workspace of the manipulator.
   b. A tool to verify the correctness of the inverse kinematics solutions obtained theoretically in the 1st part of this report. This tool enhances us to cross verify the IKP calculated angles and the DKP angles by giving any arbitrary point as input.
   c. Study the trajectory of the manipulator for a set of points with comparison of DKP angles and IKP calculated angles. Angles comparisons are being made by plotting curves of thetas.

VRML 3-D Graphics:



VRML graphics developed for simulation of the manipulator kinematics.

The above screen shot image given above is the manipulator arm graphics made in VRML language. As seen in the image, the arm has a base in the bottom (Blue), Link# 1 vertically perpendicular to the base, which is capable of rotating along its axis, and the remaining links of the manipulator arm (Link# 2 – Green, Link# 3 – red, Link# 4 – Grey, Link# 5 – pink, End effector - wrist – green and yellow). The dimensions of these links are proportional to the physical dimensions.



Over all schematic Layout of the Simulink Model for the Kinematics Simulation.

In order to understand the above Simulink schematic, brief explanation on objective of this model, explanations on each component or block are made. They are as follows:

Objectives of the model:

1. Firstly, it provides the graphics depicting the exact replica of the physical manipulator, which enables us to visualize the kinematics of the robot manipulator. In short, it is a mechanical simulation for performing the kinematic operation of the manipulator.
2. Secondly, it enables us to check the validity and correctness of DKP and IKP solution arrived in our theoretical phase of study. Giving in the Joint angles, we can visualize the manipulator arm in the 3 D space of the VRML graphics. Like wise, we can also visualize the manipulator arm in the 3 D space with each joint angles by giving any desired position with respect to the reference frame – this involves the IKP operations.

Components/ Blocks:

1. Input components:

It can be inferred from the schematic lay out given above that our model broadly has two input blocks – Joint angles and Position. Joint angles input block in turn has two blocks – Thetas and rotation [X Y Z] vectors. For example if we intend to rotate joint# 1 by 90 Deg about Z axis, Theta 1 is fed as 90 and Joint_1 Rotation vector is fed as [0 0 1]. It is to be noted that as all of our theoretical workouts are made by assigning the frames such a way that all the joint rotations are made about its Z axis. Hence by default, the rotation vector is by default [0 0 1].

2. IKP Block:

In IKP block, the function for the performing the IKP operation is embedded. This function is a direct implementation of computing the joint angles from the equations arrived in the theoretical phase of the study.

3. IKP/ DKP Switch:

This switch enables us to choose between DKP and IKP visualization. In this model, we have assigned 1 for IKP and 0 for DKP. This means that, in order to visualize (Simulate) the DKP operation 0 is to be fed in the switch and similarly 1 for the IKP operation. IKP/ DKP switch block channels the translation and rotational matrices to the VRML block either from DKP block or IKP block depending on the input we give to the switch (0/1).

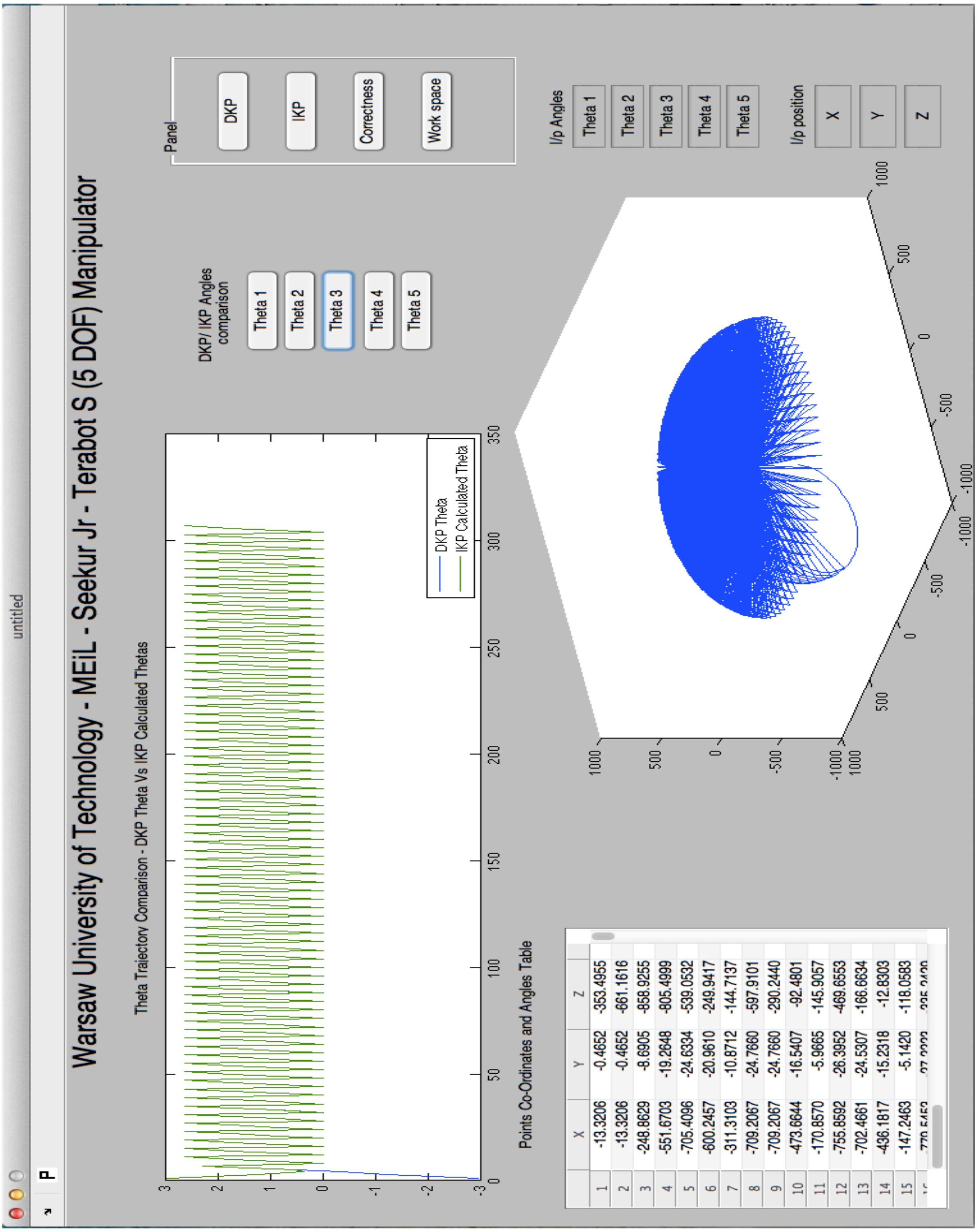
4. Translation adjustment block:

It is to be noted that the joint frames, which we considered during our theoretical derivation, are not same as the joint frames that exists by default in the VRML graphics. Hence the entire set of joint parameters (translation/ rotation) is to be modified accordingly before using it in the VRML for building the manipulator arm graphics. This modification or rather the adjustments are performed in "translation adjustment" block.

5. VRML block:

VRML block has the code for building the graphics of our manipulator in the 3 D space using the translational and rotational parameters obtained from the input components.

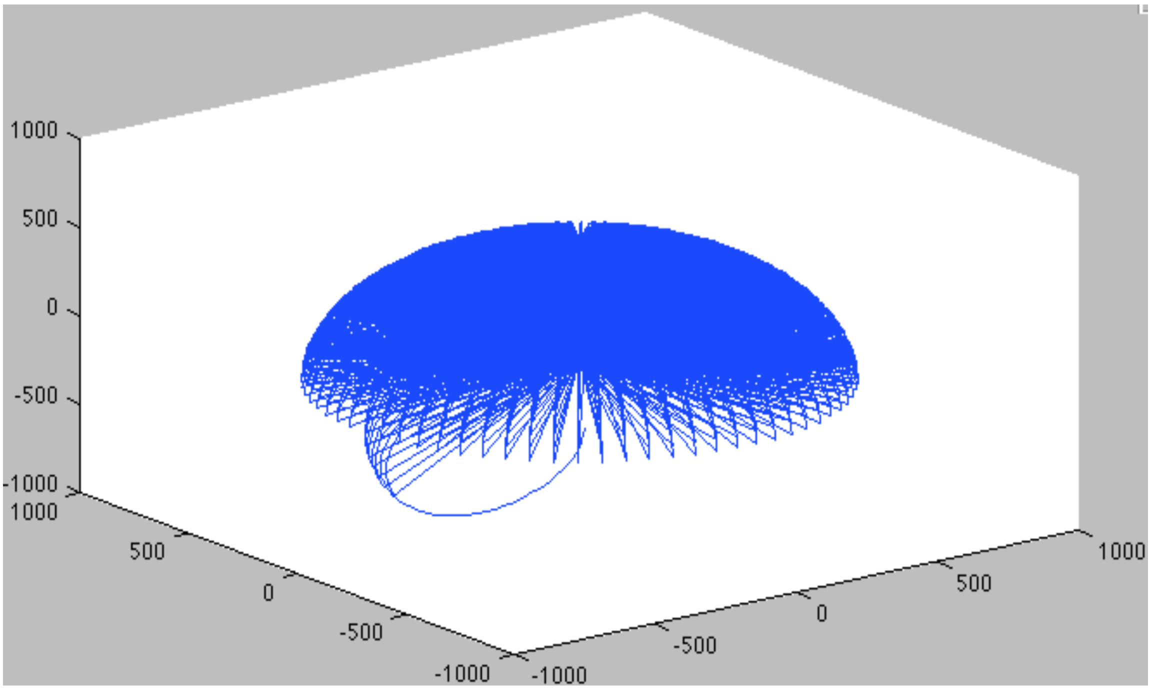
GUI:


untitled
Warsaw University of Technology - MEiL - Seekur Jr - Terabot S (5 DOF) Manipulator
Panel
DKP
IKP
Correctness
Work space
DKP/ IKP Angles comparison
Theta 1
Theta 2
Theta 3
Theta 4
Theta 5
I/p Angles
I/p position
X
Y
Z
Theta Trajectory Comparison - DKP Theta Vs IKP Calculated Thetas
DKP Theta
IKP Calculated Theta
Points Co-Ordinates and Angles Table
X Y Z
1 -13.3206 -0.4652 -353.4955
2 -13.3206 -0.4652 -661.1616
3 -248.8629 -8.6905 -858.9255
4 -551.6703 -19.2648 -805.4999
5 -705.4096 -24.6334 -539.0532
6 -600.2457 -20.9610 -249.9417
7 -311.3103 -10.8712 -144.7137
8 -709.2067 -24.7660 -597.9101
9 -709.2067 -24.7660 -290.2440
10 -473.6644 -16.5407 -92.4801
11 -170.8570 -5.9665 -145.9057
12 -755.8592 -26.3952 -469.6553
13 -702.4661 -24.5307 -166.6634
14 -436.1817 -15.2318 -12.8303
15 -147.2463 -5.1420 -118.0583

Again, in order to understand our GUI, we briefly describe each components of the GUI. They are as follows:

Workspace:

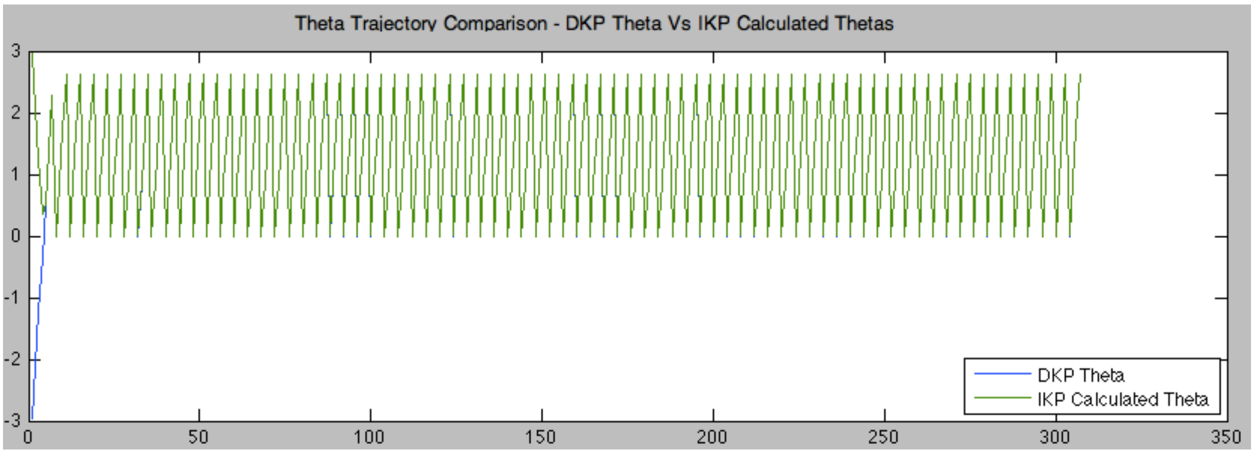
1. Workspace menu in our GUI, provides a visualization of the workspace of the manipulator on. It can show all the reachable points of the manipulator in the 3-D space.
2. The reachable points i.e the workspace of the manipulator is obtained by computing the DKP positions constraining the joint angles to with in the limits as given in the data sheet of the Terabot S manipulator.
3. It is to be noted that the deterministic position nature of the points along the X = 0, Y = 0 line is clearly visible in the 3D workspace plot.
4. All the reachable points are made available in the form of a table in our GUI. Reachable points along with their joint angles are available.



Correctness:

1. This correctness menu in our GUI enables us to verify the correctness of our IKP solution and existence of unique IKP solution for a given point.

2. GUI also provides the trajectory of the manipulator arm with varying points in the space. The GUI provides the computed IKP joint angles for a collection of points and its comparison with the DKP joint angles.
3. This comparison is made by plotting the joint angles obtained by IKP solution and the direct joint angles given to the DKP function for the particular point. Options in the form of Push buttons are made available for selecting the Joint Angle number for comparison (Theta 1, 2 , 3, 4, 5)
4. GUI also provides IKP and DKP joint angles and their corresponding positions in a tabular form.



	X	Y	Z
1	-13.3206	-0.4652	-353.4955
2	-13.3206	-0.4652	-661.1616
3	-248.8629	-8.6905	-858.9255
4	-551.6703	-19.2648	-805.4999
5	-705.4096	-24.6334	-539.0532
6	-600.2457	-20.9610	-249.9417
7	-311.3103	-10.8712	-144.7137
8	-709.2067	-24.7660	-597.9101
9	-709.2067	-24.7660	-290.2440
10	-473.6644	-16.5407	-92.4801
11	-170.8570	-5.9665	-145.9057
12	-755.8592	-26.3952	-469.6553
13	-702.4661	-24.5307	-166.6634
14	-436.1817	-15.2318	-12.8303
15	-147.2463	-5.1420	-118.0583

(Table has X,Y,Z, Corresponding DKP & IKP joint angles – with scroll option)

DKP/ IKP MENU:

1. DKP and IKP Menus in our GUI enable us to compute the DKP and IKP operation.
2. GUI has input text boxes – For angles and position. Inputting angles in the input angles text box will enable us to perform DKP operation for obtaining the position.

Inputting position (X, Y, Z) in the input position text box will enable us to perform the IKP angles computation. The results are then displayed in the table.

ANNEXURES (MATLAB Codes)

3D Graphics:

```
#VRML V2.0 utf8

#Seekur Jr. TERABOT - S Manipulator workspace illustration. (5 DOF manipulator)
#Warsaw University of Technology, Poland. Faculty of Power and Aeronautical Engineering.
#A project under the guidance of Prof. Dr. Teresa Zelienski
#Author: EMARO Team.


WorldInfo {
  title "TERABOT - S Manipulator"
  info [
    "$Revision: 1.1",
    "$Date: 2014/04/01 $",
    "$Author: EMARO Team $"
  ]
}



NavigationInfo {
  headlight TRUE
  type "EXAMINE"
}

DEF View1 Viewpoint {
    fieldOfView 0.9
    position    0 0 20
    description "Front view"
}

DEF View2 Viewpoint {
    fieldOfView 0.9
    position    20 0 0
    orientation 0 1 0 1.57
    description "Side view"
}

DEF View3 Viewpoint {
    fieldOfView 0.9
    position    0 20 0
```

```
   orientation 1 0 0 -1.57
   description "Top view"
}

DEF View4 Viewpoint {
   fieldOfView 0.9
   position    0 20 0
   orientation 1 1 0 -1.57
   description "Customized View"
}

Background  {
   groundColor [0.8 0.2 0.8]
   skyColor [0.52 0.52 1]
}


Transform {
  translation -18 -7 0
  children [
   Shape {
     appearance Appearance {
      material Material {
       diffuseColor 1 1 1
      }
     }
     geometry Text {
      string ["Modelling of Seekur Jr - Terabot S Manipulator"
              ,"Developed @ POLITECHNIKA WARSZAWSKA"]

       fontStyle FontStyle {
       family      "Wide Latin"
       style "ITALIC"
       size 0.722

      }
     }
    }
  ]
}

DEF Portal Transform {
```

```
children [
 Shape {
    appearance    Appearance {
       material  Material {
          diffuseColor  0 0 0.8
          emissiveColor 0 0 0.5
       }
    }
    geometry  Box {
       size  7 0.5 7
    }
 }
 DEF Base Transform {
    children [

       DEF Joint_1 Transform {
          translation   0 1.3950 0
          rotation 0 0 0 0
          center    0 5 0
          children [
            Shape {
               appearance  Appearance {
                  material    Material {
                     diffuseColor    0 0.8 0
                     emissiveColor   0 0.5 0
                  }
               }
               geometry    Cylinder {
                  height  2.29
                  radius  0.2
               }
            }
            Transform {
            translation 0 1.1450 0
            rotation    0 0 1  1.5708
               children DEF Joint_2 Transform {
            rotation    0 0 1  1.7453
            translation 2.0632 0.3637 0
                  children [
                     Shape {
                        appearance  Appearance {
                           material    Material {
                              diffuseColor    0.8 0 0
```

```
          emissiveColor   0.5 0 0
        }
      }
      geometry    Cylinder {
        height  4.19
        radius 0.2
      }

    }
    Transform {

      translation 0 -2.095 0
      rotation    0 0 0  0
      children DEF Joint_3 Transform {
      rotation    0 0 1  0.5236
      translation 0.9100 -1.5762 0

        children [
          Shape {
            appearance    Appearance {
              material  Material {

              }
            }
            geometry  Cylinder {
              height    3.64
              radius    0.2
            }
          }

    Transform {

      translation 0 -1.82 0
      rotation    0 0 0  0
      children DEF Joint_4 Transform {
      rotation 0 0 1 0
      translation 0 -1.335 0

        children [
          Shape {
            appearance    Appearance {
              material  Material {
```

```
                diffuseColor    0.8 0 1
                emissiveColor   0.5 0 0
                  }
               }
               geometry  Cylinder {
                  height    2.67
                  radius    0.2
               }
            }

      Transform {

         translation 0 -1.7350 0
         rotation    0 0 0  0
         children DEF Joint_5 Transform {

         translation 0 0.8 0
         rotation    0 0 0  0
            children [
             Shape {
               appearance    Appearance {
                  material  Material {
                diffuseColor    0 1 0
                emissiveColor   0.5 0 0
                  }
               }
               geometry  Cylinder {
                  height    0.8
                  radius    0.2
               }
            }


DEF Fingers Transform {
               translation   0 0 0
               children [

   DEF Finger1 Transform {
    translation   0 0 0
    rotation 0 0 1 0.3
                  children
    Shape {
                     appearance    Appearance {
```

```
                    material  Material {
           diffuseColor .8 .8 0
           specularColor .5 .5 .5
           emissiveColor .15 .15 0
           ambientIntensity 0
                    }
                  }
                  geometry  Extrusion {
crossSection [0.2 0.2, 0.2 -0.2, -0.2 -0.2, -0.2 0.2, 0.2 0.2 ]
          spine [0 0 0, 0.4 -1 0, 0 -2 0]
          scale [1 1, 1 1, 0.2 0.4]
                  }
      }
     },
     DEF Finger2 Transform {
      translation   0 0 0
      rotation 0 0 1 -0.3
               children

      Shape {
                  appearance    Appearance {
                    material  Material {
           diffuseColor .8 .8 0
           specularColor .5 .5 .5
           emissiveColor .15 .15 0
           ambientIntensity 0
                    }
                  }
                  geometry  Extrusion {
                  crossSection [0.2 0.2, 0.2 -0.2, -0.2 -0.2, -0.2 0.2, 0.2 0.2 ]
          spine [0 0 0, -0.4 -1 0, 0 -2 0]
          scale [1 1, 1 1, 0.2 0.4]
                  }
      }}]}]}}]
     }
    }
    ]
    }
    }
    ]
    }
  }
```

```
            ]
          }
        ]
      }
    ]
}
```

DKP/ IKP Switch Function:

```
function [theta1,theta2,theta3,theta4,theta5] =
fcn(dth1,dth2,dth3,dth4,dth5,ith1,ith2,ith3,ith4,ith5,selector)
%#eml

if selector >= 1
   theta1 = ith1;
   theta2 = ith2;
   theta3 = ith3;
   theta4 = ith4;
   theta5 = ith5;
else
   theta1 = (dth1)*pi/180;
   theta2 = (dth2)*pi/180;
   theta3 = (dth3)*pi/180;
   theta4 = (dth4)*pi/180;
   theta5 = (dth5)*pi/180;
end
```

Translation Adjustment Function:

```
function [t2,t3,t4,t5] = translation_adjustment(theta2,theta3,theta4,theta5)
%#eml

t2 = [2.095*sin(theta2) -2.09520*cos(theta2) 0];
t3 = [1.820*sin(theta3) -1.82000*cos(theta3) 0];
t4 = [1.335*sin(theta4) -1.33500*cos(theta4) 0];
t5 = [0 0 0];
```

IKP Function:

```
function [t1,t2,t3,t4,t5] = ikp(x,y,z)

z = z-229;

th1 = rand;
th2 = rand;
th3 = rand;
th4 =  -th3 - th2;
th5 = 0;

a2 = 419;
a3 = 364;
d6 = 267;

T_0_1 = [
   cos(th1),  -sin(th1),  0,  0;
   sin(th1),   cos(th1),  0,  0;
          0,          0,  1,  0;
          0,          0,  0,  1
];

T_1_2 = [
   cos(th2),  -sin(th2),   0,  0;
          0,          0,  -1,  0;
   sin(th2),   cos(th2),   0,  0;
          0,          0,   0,  1
];

T_2_3 = [
   cos(th3),  -sin(th3),  0,  a2;
   sin(th3),   cos(th3),  0,   0;
          0,          0,  1,   0;
          0,          0,  0,   1
];

T_3_4 = [
   cos(th4),  -sin(th4),  0,  a3;
   sin(th4),   cos(th4),  0,   0;
          0,          0,  1,   0;
          0,          0,  0,   1
```

```
];

T_4_5 = [
  cos(th5),  -sin(th5),   0,  0;
        0,         0,  -1,  0;
  sin(th5),   cos(th5),   0,  0;
        0,         0,   0,  1
];

T_5_6 = [
  1,  0,  0,   0;
  0,  1,  0,   0;
  0,  0,  1,  d6;
  0,  0,  0,   1
];

T_0_2 = T_0_1 * T_1_2;
T_0_3 = T_0_1 * T_1_2 * T_2_3;
T_0_4 = T_0_1 * T_1_2 * T_2_3 * T_3_4;
T_0_5 = T_0_1 * T_1_2 * T_2_3 * T_3_4 * T_4_5;
T_0_6 = T_0_1 * T_1_2 * T_2_3 * T_3_4 * T_4_5 * T_5_6;


calc_T_0_2 = [
  cos(th1)*cos(th2),  -cos(th1)*sin(th2),   sin(th1),  0;
  sin(th1)*cos(th2),  -sin(th1)*sin(th2),  -cos(th1),  0;
          sin(th2),           cos(th2),         0,  0;
                0,                  0,         0,  1;
];

calc_T_0_4 = [
  cos(th1)*cos(th2+th3+th4),  -cos(th1)*sin(th2+th3+th4),   sin(th1),  cos(th1)*( a3*cos(th2+th3)
+ a2*cos(th2) );
  sin(th1)*cos(th2+th3+th4),  -sin(th1)*sin(th2+th3+th4),  -cos(th1),  sin(th1)*( a3*cos(th2+th3)
+ a2*cos(th2) );
          sin(th2+th3+th4),           cos(th2+th3+th4),         0,              a3*sin(th2+th3) +
a2*sin(th2);
                        0,                          0,         0,                                   1;
];

calc_T_0_5 = [
  cos(th1)*cos(th2+th3+th4)*cos(th5) + sin(th1)*sin(th5),  -cos(th1)*cos(th2+th3+th4)*sin(th5)
+ sin(th1)*cos(th5),  cos(th1)*sin(th2+th3+th4),  cos(th1)*( a3*cos(th2+th3) + a2*cos(th2) );
```

```
   sin(th1)*cos(th2+th3+th4)*cos(th5) - cos(th1)*sin(th5),  -sin(th1)*cos(th2+th3+th4)*sin(th5) - cos(th1)*cos(th5),  sin(th1)*sin(th2+th3+th4),  sin(th1)*( a3*cos(th2+th3) + a2*cos(th2) );
                           sin(th2+th3+th4)*cos(th5),                           -sin(th2+th3+th4)*sin(th5),  -cos(th2+th3+th4),              a3*sin(th2+th3) + a2*sin(th2);
                                                    0,                                                   0,                  0,  1;
];

calc_T_0_6 = [
   cos(th1)*cos(th2+th3+th4)*cos(th5) + sin(th1)*sin(th5),  -cos(th1)*cos(th2+th3+th4)*sin(th5) + sin(th1)*cos(th5),  cos(th1)*sin(th2+th3+th4),  d6*cos(th1)*sin(th2+th3+th4) + cos(th1)*( a3*cos(th2+th3) + a2*cos(th2) );
   sin(th1)*cos(th2+th3+th4)*cos(th5) - cos(th1)*sin(th5),  -sin(th1)*cos(th2+th3+th4)*sin(th5) - cos(th1)*cos(th5),  sin(th1)*sin(th2+th3+th4),  d6*sin(th1)*sin(th2+th3+th4) + sin(th1)*( a3*cos(th2+th3) + a2*cos(th2) );
                           sin(th2+th3+th4)*cos(th5),                           -sin(th2+th3+th4)*sin(th5),  -cos(th2+th3+th4),                   -d6*cos(th2+th3+th4) + a3*sin(th2+th3) + a2*sin(th2);
                                                    0,                                                   0,                  0,  1;
];


% Inverse Kinematics:

t1 = atan2(-y,x);
T61 = inv(T_0_1)*(calc_T_0_6);
T61(1,4) = x;
T61(2,4) = y;
T61(3,4) = z;
calcT60 = [cos(t1) -sin(t1) 0 0;sin(t1) cos(t1) 0 0;0 0 1 0;0 0 0 1] * T61;


r11 =  calcT60(1,1);
r12 =  calcT60(1,2);
r13 =  calcT60(1,3);
px  =  x;
r21 =  calcT60(2,1);
r22 =  calcT60(2,2)
r23 =  calcT60(2,3)
py  =  y;
r31 =  calcT60(3,1)
r32 =  calcT60(3,2)
```

```
r33 =  calcT60(3,3)
pz  = z;



calc_T_0_6_d = [
   r11, r12, r13, px;
   r21, r22, r23, py;
   r31, r32, r33, pz;
     0,   0,   0,  1;
];



A = px - d6*r13;
B = py - d6*r23;
C = pz - d6*r33;
D = ( (A*cos(t1))^2 + 2*A*B*cos(t1)*sin(t1) + (B*sin(t1))^2 + C^2 - a3^2 - a2^2 ) / (2*a2*a3);

temp = real(sqrt(complex(1 - D^2)));
t3 = (atan2(temp, D ));

E = a3*cos(t3) + a2;
F = a3*sin(t3);
ksi = sqrt( E^2 + F^2 );
phi = atan2( F, E );
K = C;

temp = real(sqrt(complex(1 - (K/ksi)^2 )));

t2 = (atan2( (K/ksi), temp ) - phi);


t4 = -t2 -t3;

t5 = 0;
```

GUI

```
function varargout = untitled(varargin)
% UNTITLED M-file for untitled.fig
%      UNTITLED, by itself, creates a new UNTITLED or raises the existing
%      singleton*.
```

```
%
%      H = UNTITLED returns the handle to a new UNTITLED or the handle to
%      the existing singleton*.
%
%      UNTITLED('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in UNTITLED.M with the given input arguments.
%
%      UNTITLED('Property','Value',...) creates a new UNTITLED or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before untitled_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to untitled_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help untitled

% Last Modified by GUIDE v2.5 03-May-2014 03:24:07

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @untitled_OpeningFcn, ...
                   'gui_OutputFcn',  @untitled_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

```
% --- Executes just before untitled is made visible.
function untitled_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to untitled (see VARARGIN)

% Choose default command line output for untitled
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes untitled wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = untitled_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function ip_angle1_Callback(hObject, eventdata, handles)
% hObject    handle to ip_angle1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ip_angle1 as text
%        str2double(get(hObject,'String')) returns contents of ip_angle1 as a double


% --- Executes during object creation, after setting all properties.
function ip_angle1_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to ip_angle1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
Angles_input = str2double(get(hObject,'String'));
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ip_positionX_Callback(hObject, eventdata, handles)
% hObject    handle to ip_positionX (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ip_positionX as text
%        str2double(get(hObject,'String')) returns contents of ip_positionX as a double


% --- Executes during object creation, after setting all properties.
function ip_positionX_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ip_positionX (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in DKP_Push.
function DKP_Push_Callback(hObject, eventdata, handles)
% hObject    handle to DKP_Push (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
a2 = 419;
a3 = 364;
```

```
d6 = 267;
t1 = str2double(get(handles.ip_angle1,'String'));
t2 = str2double(get(handles.ip_angle2,'String'));
t3 = str2double(get(handles.ip_angle3,'String'));
t4 = str2double(get(handles.ip_angle4,'String'));
t5 = str2double(get(handles.ip_angle5,'String'));

th1 = deg2rad(t1);
th2 = deg2rad(t2);
th3 = deg2rad(t3);
th4 = deg2rad(t4);
th5 = deg2rad(t5);

x  =  d6*cos(th1)*sin(th2+th3+th4) + cos(th1)*( a3*cos(th2+th3) + a2*cos(th2) );
y  =  d6*sin(th1)*sin(th2+th3+th4) + sin(th1)*( a3*cos(th2+th3) + a2*cos(th2) );
z  = -d6*cos(th2+th3+th4) + a3*sin(th2+th3) + a2*sin(th2);
dat = [t1;t2;t3;t4;t5;x;y;z];
columnname =   {'Theta1','Theta2','Theta3','Theta4','Theta5','X','Y','Z'};
set(uitable,'data',dat','ColumnName',columnname);

% --- Executes on button press in IKP_Push.
function IKP_Push_Callback(hObject, eventdata, handles)
% hObject    handle to IKP_Push (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
a2 = 419;
a3 = 364;
d6 = 267;
x = str2double(get(handles.ip_positionX,'String'));
y = str2double(get(handles.ip_positionY,'String'));
z = str2double(get(handles.ip_positionZ,'String'));

th1 = rand;
th2 = rand;
th3 = rand;
th4 =  -th3 - th2;
th5 = 0;

a2 = 419;
a3 = 364;
d6 = 267;

T_0_1 = [
```

```
  cos(th1),  -sin(th1),  0,  0;
  sin(th1),   cos(th1),  0,  0;
        0,          0,  1,  0;
        0,          0,  0,  1
];

T_1_2 = [
  cos(th2),  -sin(th2),   0,  0;
        0,          0,  -1,  0;
  sin(th2),   cos(th2),   0,  0;
        0,          0,   0,  1
];

T_2_3 = [
  cos(th3),  -sin(th3),  0,  a2;
  sin(th3),   cos(th3),  0,   0;
        0,          0,  1,   0;
        0,          0,  0,   1
];

T_3_4 = [
  cos(th4),  -sin(th4),  0,  a3;
  sin(th4),   cos(th4),  0,   0;
        0,          0,  1,   0;
        0,          0,  0,   1
];

T_4_5 = [
  cos(th5),  -sin(th5),   0,  0;
        0,          0,  -1,  0;
  sin(th5),   cos(th5),   0,  0;
        0,          0,   0,  1
];

T_5_6 = [
  1,  0,  0,   0;
  0,  1,  0,   0;
  0,  0,  1,  d6;
  0,  0,  0,   1
];

T_0_2 = T_0_1 * T_1_2;
```

```
T_0_3 = T_0_1 * T_1_2 * T_2_3;
T_0_4 = T_0_1 * T_1_2 * T_2_3 * T_3_4;
T_0_5 = T_0_1 * T_1_2 * T_2_3 * T_3_4 * T_4_5;
T_0_6 = T_0_1 * T_1_2 * T_2_3 * T_3_4 * T_4_5 * T_5_6;


calc_T_0_2 = [
  cos(th1)*cos(th2),  -cos(th1)*sin(th2),   sin(th1),  0;
  sin(th1)*cos(th2),  -sin(th1)*sin(th2),  -cos(th1),  0;
          sin(th2),           cos(th2),          0,  0;
               0,                  0,          0,  1;
];

calc_T_0_4 = [
  cos(th1)*cos(th2+th3+th4),  -cos(th1)*sin(th2+th3+th4),   sin(th1),  cos(th1)*( a3*cos(th2+th3)
+ a2*cos(th2) );
  sin(th1)*cos(th2+th3+th4),  -sin(th1)*sin(th2+th3+th4),  -cos(th1),  sin(th1)*( a3*cos(th2+th3)
+ a2*cos(th2) );
          sin(th2+th3+th4),           cos(th2+th3+th4),          0,              a3*sin(th2+th3) +
a2*sin(th2);
                      0,                        0,          0,                                   1;
];

calc_T_0_5 = [
  cos(th1)*cos(th2+th3+th4)*cos(th5) + sin(th1)*sin(th5),  -cos(th1)*cos(th2+th3+th4)*sin(th5)
+ sin(th1)*cos(th5),  cos(th1)*sin(th2+th3+th4),  cos(th1)*( a3*cos(th2+th3) + a2*cos(th2) );
  sin(th1)*cos(th2+th3+th4)*cos(th5) - cos(th1)*sin(th5),  -sin(th1)*cos(th2+th3+th4)*sin(th5) -
cos(th1)*cos(th5),  sin(th1)*sin(th2+th3+th4),  sin(th1)*( a3*cos(th2+th3) + a2*cos(th2) );
                      sin(th2+th3+th4)*cos(th5),                          -sin(th2+th3+th4)*sin(th5),
-cos(th2+th3+th4),              a3*sin(th2+th3) + a2*sin(th2);
                                         0,                                          0,                      0,
1;
];

calc_T_0_6 = [
  cos(th1)*cos(th2+th3+th4)*cos(th5) + sin(th1)*sin(th5),  -cos(th1)*cos(th2+th3+th4)*sin(th5)
+ sin(th1)*cos(th5),  cos(th1)*sin(th2+th3+th4),  d6*cos(th1)*sin(th2+th3+th4) +
cos(th1)*( a3*cos(th2+th3) + a2*cos(th2) );
  sin(th1)*cos(th2+th3+th4)*cos(th5) - cos(th1)*sin(th5),  -sin(th1)*cos(th2+th3+th4)*sin(th5) -
cos(th1)*cos(th5),  sin(th1)*sin(th2+th3+th4),  d6*sin(th1)*sin(th2+th3+th4) +
sin(th1)*( a3*cos(th2+th3) + a2*cos(th2) );
                      sin(th2+th3+th4)*cos(th5),                          -sin(th2+th3+th4)*sin(th5),
-cos(th2+th3+th4),                    -d6*cos(th2+th3+th4) + a3*sin(th2+th3) + a2*sin(th2);
```

```
                                    0,                              0,              0,
1;
];

t1 = atan2(y,x);
T61 = inv(T_0_1)*(calc_T_0_6)
T61(1,4) = x;
T61(2,4) = y;
T61(3,4) = z;
calcT60 = [cos(t1) -sin(t1) 0 0;sin(t1) cos(t1) 0 0;0 0 1 0;0 0 0 1] * T61;


r11 =  calcT60(1,1);
r12 =  calcT60(1,2);
r13 =  calcT60(1,3);
px  =  x;
r21 =  calcT60(2,1);
r22 =  calcT60(2,2)
r23 =  calcT60(2,3)
py  =  y;
r31 =  calcT60(3,1)
r32 =  calcT60(3,2)
r33 =  calcT60(3,3)
pz  = z;



calc_T_0_6_d = [
  r11, r12, r13, px;
  r21, r22, r23, py;
  r31, r32, r33, pz;
    0,   0,   0,  1;
];



A = px - d6*r13;
B = py - d6*r23;
C = pz - d6*r33;
D = ( (A*cos(t1))^2 + 2*A*B*cos(t1)*sin(t1) + (B*sin(t1))^2 + C^2 - a3^2 - a2^2 ) / (2*a2*a3);

temp = real(sqrt(complex(1 - D^2)));
```

```
t3 = (atan2(temp, D ));

E = a3*cos(t3) + a2;
F = a3*sin(t3);
ksi = sqrt( E^2 + F^2 );
phi = atan2( F, E );
K = C;

temp = real(sqrt(complex(1 - (K/ksi)^2 )));

t2 = (atan2( (K/ksi), temp ) - phi);


t4 = -t2 -t3;

t5 = 0;

t1 = rad2deg(t1);
t2 = rad2deg(t2);
t3 = rad2deg(t3);
t4 = rad2deg(t4);
t5 = rad2deg(t5);
dat = [t1;t2;t3;t4;t5;x;y;z];
columnname =   {'Theta1','Theta2','Theta3','Theta4','Theta5','X','Y','Z'};
set(uitable,'data',dat','ColumnName',columnname);


% --- Executes on button press in Correctness_Push.
function Correctness_Push_Callback(hObject, eventdata, handles)
% hObject    handle to Correctness_Push (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
a2 = 419;
a3 = 364;
d6 = 267;
t1 = -178;
t2 = -35;
t3 = -170;
t4 = -178;
n = 1;
k=0;
while t1 < 178
   while t2 < 90
```

```
    while t3 < 162

        th5(n) =0;
        th1(n) = deg2rad(t1);
        th2(n) = deg2rad(t2);
        th3(n) = deg2rad(t3);
        th4(n) = -th2(n) - th3(n);
        r11 =  cos(th1(n))*cos(th2(n)+th3(n)+th4(n))*cos(th5(n)) + sin(th1(n))*sin(th5(n));
        r12 = -cos(th1(n))*cos(th2(n)+th3(n)+th4(n))*sin(th5(n)) + sin(th1(n))*cos(th5(n));
        r13 =  cos(th1(n))*sin(th2(n)+th3(n)+th4(n));
        px(n)  =  d6*cos(th1(n))*sin(th2(n)+th3(n)+th4(n)) +
cos(th1(n))*( a3*cos(th2(n)+th3(n)) + a2*cos(th2(n)) );
        r21 =  sin(th1(n))*cos(th2(n)+th3(n)+th4(n))*cos(th5(n)) - cos(th1(n))*sin(th5(n));
        r22 = -sin(th1(n))*cos(th2(n)+th3(n)+th4(n))*sin(th5(n)) - cos(th1(n))*cos(th5(n));
        r23 =  sin(th1(n))*sin(th2(n)+th3(n)+th4(n));
        py(n)  =  d6*sin(th1(n))*sin(th2(n)+th3(n)+th4(n)) +
sin(th1(n))*( a3*cos(th2(n)+th3(n)) + a2*cos(th2(n)) );
        r31 =  sin(th2(n)+th3(n)+th4(n))*cos(th5(n));
        r32 = -sin(th2(n)+th3(n)+th4(n))*sin(th5(n));
        r33 = -cos(th2(n)+th3(n)+th4(n));
        pz(n)  = -d6*cos(th2(n)+th3(n)+th4(n)) + a3*sin(th2(n)+th3(n)) + a2*sin(th2(n));

        calc_T_0_6_d = [
          r11, r12, r13, px(n);
          r21, r22, r23, py(n);
          r31, r32, r33, pz(n);
            0,   0,   0,  1;
        ];

        calc_th1(n) = atan( py(n) / px(n) );

        A = px(n) - d6*r13;
        B = py(n) - d6*r23;
        C = pz(n) - d6*r33;
        D = ( (A*cos(calc_th1(n)))^2 + 2*A*B*cos(calc_th1(n))*sin(calc_th1(n)) +
(B*sin(calc_th1(n)))^2 + C^2 - a3^2 - a2^2 ) / (2*a2*a3);

        calc_th3(n) = atan2( sqrt(1 - D^2), D );

        E = a3*cos(calc_th3(n)) + a2;
        F = a3*sin(calc_th3(n));
        ksi = sqrt( E^2 + F^2 );
```

```
                phi = atan2( F, E );
                K = C;

                calc_th2(n) = atan2( (K/ksi), sqrt( 1 - (K/ksi)^2 ) ) - phi;

                calc_th4(n) = -calc_th2(n) - calc_th3(n);

                calc_th5(n) = 0;
                index(n) = n;
                        n = n+1;




                    t3 = t3+50;
        end

                  t3 = 0;
                  t2 = t2+10;
      end

                t3 = 0;
                t2 = 0;
                t1 = t1+50;
end

plot(handles.Correctness_plot,index,(th4),index,(calc_th4));
h = legend(handles.Correctness_plot,'DKP Theta','IKP Calculated Theta',2);
%set(handles.Correctness_plot,title,'Comparison plot - DKP Angles Vs IKP Calculated Angles');
dat = [px;py;pz;th1;th2;th3;th4;th1;calc_th2;calc_th3;calc_th4];
columnname =
{'X','Y','Z','Theta1','Theta2','Theta3','Theta4','calc_th1','calc_th2','calc_th43','calc_th4'};
set(uitable,'data',dat','ColumnName',columnname);


% --- Executes on button press in Work_Space_Push.
function Work_Space_Push_Callback(hObject, eventdata, handles)
% hObject    handle to Work_Space_Push (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
a2 = 419;
a3 = 364;
d6 = 267;
```

```
A1 = -178;
A2 = -35;
A3 = -170;
n = 1;
while A1 < 178
   while A2 < 90
      while A3 < 162
         if (A2+A3)<178 && (A2+A3)>-178
            An1(n) = deg2rad(A1);
            An2(n) = deg2rad(A2);
            An3(n) = deg2rad(A3);
            An4(n) = -(An2(n) + An3(n));
            point_x(n)  =  d6*cos(An1(n))*sin(An2(n)+An3(n)+An4(n)) +
cos(An1(n))*( a3*cos(An2(n)+An3(n)) + a2*cos(An2(n)) );
            point_y(n)  =  d6*sin(An1(n))*sin(An2(n)+An3(n)+An4(n)) +
sin(An1(n))*( a3*cos(An2(n)+An3(n)) + a2*cos(An2(n)) );
            point_z(n)  = -d6*cos(An2(n)+An3(n)+An4(n)) + a3*sin(An2(n)+An3(n)) +
a2*sin(An2(n));
            index(n) = n;

                        n = n+1;
                     end
                     A3 = A3+5;
                  end
                  A3 = 0;
                  A2 = A2+5;
               end
               A3 = 0;
               A2 = 0;
               A1 = A1+5;
end

plot3(handles.Work_Space_Plot,point_x,point_y,point_z);
dat = [point_x;point_y;point_z;An1;An2;An3;An4];
columnname =   {'X','Y','Z','Theta1','Theta2','Theta3','Theta4'};
set(uitable,'data',dat','ColumnName',columnname);

% --- Executes on button press in Theta_1.
function Theta_1_Callback(hObject, eventdata, handles)
% hObject    handle to Theta_1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
a2 = 419;
a3 = 364;
d6 = 267;
t1 = -178;
t2 = -35;
t3 = -170;
t4 = -178;
n = 1;
k=0;
while t1 < 178
    while t2 < 90
        while t3 < 162

                th5(n) =0;
                th1(n) = deg2rad(t1);
                th2(n) = deg2rad(t2);
                th3(n) = deg2rad(t3);
                th4(n) = -th2(n) - th3(n);
                r11 =  cos(th1(n))*cos(th2(n)+th3(n)+th4(n))*cos(th5(n)) + sin(th1(n))*sin(th5(n));
                r12 = -cos(th1(n))*cos(th2(n)+th3(n)+th4(n))*sin(th5(n)) + sin(th1(n))*cos(th5(n));
                r13 =  cos(th1(n))*sin(th2(n)+th3(n)+th4(n));
                px(n)  =  d6*cos(th1(n))*sin(th2(n)+th3(n)+th4(n)) +
cos(th1(n))*( a3*cos(th2(n)+th3(n)) + a2*cos(th2(n)) );
                r21 =  sin(th1(n))*cos(th2(n)+th3(n)+th4(n))*cos(th5(n)) - cos(th1(n))*sin(th5(n));
                r22 = -sin(th1(n))*cos(th2(n)+th3(n)+th4(n))*sin(th5(n)) - cos(th1(n))*cos(th5(n));
                r23 =  sin(th1(n))*sin(th2(n)+th3(n)+th4(n));
                py(n)  =  d6*sin(th1(n))*sin(th2(n)+th3(n)+th4(n)) +
sin(th1(n))*( a3*cos(th2(n)+th3(n)) + a2*cos(th2(n)) );
                r31 =  sin(th2(n)+th3(n)+th4(n))*cos(th5(n));
                r32 = -sin(th2(n)+th3(n)+th4(n))*sin(th5(n));
                r33 = -cos(th2(n)+th3(n)+th4(n));
                pz(n)  = -d6*cos(th2(n)+th3(n)+th4(n)) + a3*sin(th2(n)+th3(n)) + a2*sin(th2(n));

                calc_T_0_6_d = [
                  r11, r12, r13, px(n);
                  r21, r22, r23, py(n);
                  r31, r32, r33, pz(n);
                    0,   0,   0,  1;
                ];

                calc_th1(n) = th1(n);

                A = px(n) - d6*r13;
```

```
            B = py(n) - d6*r23;
            C = pz(n) - d6*r33;
            D = ( (A*cos(calc_th1(n)))^2 + 2*A*B*cos(calc_th1(n))*sin(calc_th1(n)) +
(B*sin(calc_th1(n)))^2 + C^2 - a3^2 - a2^2 ) / (2*a2*a3);

            calc_th3(n) = atan2( sqrt(1 - D^2), D );

            E = a3*cos(calc_th3(n)) + a2;
            F = a3*sin(calc_th3(n));
            ksi = sqrt( E^2 + F^2 );
            phi = atan2( F, E );
            K = C;

            calc_th2(n) = atan2( (K/ksi), sqrt( 1 - (K/ksi)^2 ) ) - phi;

            calc_th4(n) = -calc_th2(n) - calc_th3(n);

            calc_th5(n) = 0;
            index(n) = n;
                        n = n+1;




                  t3 = t3+50;
      end

               t3 = 0;
               t2 = t2+10;
   end

            t3 = 0;
            t2 = 0;
            t1 = t1+50;
end

plot(handles.Correctness_plot,index,(th1),index,(calc_th1));
h = legend(handles.Correctness_plot,'DKP Theta','IKP Calculated Theta',4);
dat = [px;py;pz;th1;th2;th3;th4;th1;calc_th2;calc_th3;calc_th4];
columnname =
{'X','Y','Z','Theta1','Theta2','Theta3','Theta4','Calc_theta1','Calc_Theta2','Calc_Theta3','Calc_The
ta4'};
```

```
set(uitable,'data',dat','ColumnName',columnname);


% --- Executes on button press in Theta_3.
function Theta_3_Callback(hObject, eventdata, handles)
% hObject    handle to Theta_3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
a2 = 419;
a3 = 364;
d6 = 267;
t1 = -178;
t2 = -35;
t3 = -170;
t4 = -178;
n = 1;
k=0;
while t1 < 178
   while t2 < 90
      while t3 < 162

            th5(n) =0;
            th1(n) = deg2rad(t1);
            th2(n) = deg2rad(t2);
            th3(n) = deg2rad(t3);
            th4(n) = -th2(n) - th3(n);
            r11 =  cos(th1(n))*cos(th2(n)+th3(n)+th4(n))*cos(th5(n)) + sin(th1(n))*sin(th5(n));
            r12 = -cos(th1(n))*cos(th2(n)+th3(n)+th4(n))*sin(th5(n)) + sin(th1(n))*cos(th5(n));
            r13 =  cos(th1(n))*sin(th2(n)+th3(n)+th4(n));
            px(n)  =  d6*cos(th1(n))*sin(th2(n)+th3(n)+th4(n)) +
cos(th1(n))*( a3*cos(th2(n)+th3(n)) + a2*cos(th2(n)) );
            r21 =  sin(th1(n))*cos(th2(n)+th3(n)+th4(n))*cos(th5(n)) - cos(th1(n))*sin(th5(n));
            r22 = -sin(th1(n))*cos(th2(n)+th3(n)+th4(n))*sin(th5(n)) - cos(th1(n))*cos(th5(n));
            r23 =  sin(th1(n))*sin(th2(n)+th3(n)+th4(n));
            py(n)  =  d6*sin(th1(n))*sin(th2(n)+th3(n)+th4(n)) +
sin(th1(n))*( a3*cos(th2(n)+th3(n)) + a2*cos(th2(n)) );
            r31 =  sin(th2(n)+th3(n)+th4(n))*cos(th5(n));
            r32 = -sin(th2(n)+th3(n)+th4(n))*sin(th5(n));
            r33 = -cos(th2(n)+th3(n)+th4(n));
            pz(n)  = -d6*cos(th2(n)+th3(n)+th4(n)) + a3*sin(th2(n)+th3(n)) + a2*sin(th2(n));

            calc_T_0_6_d = [
              r11, r12, r13, px(n);
```

```
  r21, r22, r23, py(n);
  r31, r32, r33, pz(n);
    0,   0,   0,  1;
];

calc_th1(n) = atan( py(n) / px(n) );

A = px(n) - d6*r13;
B = py(n) - d6*r23;
C = pz(n) - d6*r33;
D = ( (A*cos(calc_th1(n)))^2 + 2*A*B*cos(calc_th1(n))*sin(calc_th1(n)) +
(B*sin(calc_th1(n)))^2 + C^2 - a3^2 - a2^2 ) / (2*a2*a3);

calc_th3(n) = atan2( sqrt(1 - D^2), D );

E = a3*cos(calc_th3(n)) + a2;
F = a3*sin(calc_th3(n));
ksi = sqrt( E^2 + F^2 );
phi = atan2( F, E );
K = C;

calc_th2(n) = atan2( (K/ksi), sqrt( 1 - (K/ksi)^2 ) ) - phi;

calc_th4(n) = -calc_th2(n) - calc_th3(n);

calc_th5(n) = 0;
index(n) = n;
        n = n+1;



      t3 = t3+50;
    end

    t3 = 0;
    t2 = t2+10;
  end

  t3 = 0;
  t2 = 0;
  t1 = t1+50;
```

```
end

plot(handles.Correctness_plot,index,(th3),index,(calc_th3));
h = legend(handles.Correctness_plot,'DKP Theta','IKP Calculated Theta',4);
dat = [px;py;pz;th1;th2;th3;th4;th1;calc_th2;calc_th3;calc_th4];
columnname =
{'X','Y','Z','Theta1','Theta2','Theta3','Theta4','Calc_theta1','Calc_Theta2','Calc_Theta3','Calc_The
ta4'};
set(uitable,'data',dat','ColumnName',columnname);


% --- Executes on button press in Theta_4.
function Theta_4_Callback(hObject, eventdata, handles)
% hObject    handle to Theta_4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
a2 = 419;
a3 = 364;
d6 = 267;
t1 = -178;
t2 = -35;
t3 = -170;
t4 = -178;
n = 1;
k=0;
while t1 < 178
   while t2 < 90
      while t3 < 162

            th5(n) =0;
            th1(n) = deg2rad(t1);
            th2(n) = deg2rad(t2);
            th3(n) = deg2rad(t3);
            th4(n) = -th2(n) - th3(n);
            r11 =  cos(th1(n))*cos(th2(n)+th3(n)+th4(n))*cos(th5(n)) + sin(th1(n))*sin(th5(n));
            r12 = -cos(th1(n))*cos(th2(n)+th3(n)+th4(n))*sin(th5(n)) + sin(th1(n))*cos(th5(n));
            r13 =  cos(th1(n))*sin(th2(n)+th3(n)+th4(n));
            px(n)  =  d6*cos(th1(n))*sin(th2(n)+th3(n)+th4(n)) +
cos(th1(n))*( a3*cos(th2(n)+th3(n)) + a2*cos(th2(n)) );
            r21 =  sin(th1(n))*cos(th2(n)+th3(n)+th4(n))*cos(th5(n)) - cos(th1(n))*sin(th5(n));
            r22 = -sin(th1(n))*cos(th2(n)+th3(n)+th4(n))*sin(th5(n)) - cos(th1(n))*cos(th5(n));
            r23 =  sin(th1(n))*sin(th2(n)+th3(n)+th4(n));
            py(n)  =  d6*sin(th1(n))*sin(th2(n)+th3(n)+th4(n)) +
```

```
sin(th1(n))*( a3*cos(th2(n)+th3(n)) + a2*cos(th2(n)) );
            r31 =  sin(th2(n)+th3(n)+th4(n))*cos(th5(n));
            r32 = -sin(th2(n)+th3(n)+th4(n))*sin(th5(n));
            r33 = -cos(th2(n)+th3(n)+th4(n));
            pz(n)  = -d6*cos(th2(n)+th3(n)+th4(n)) + a3*sin(th2(n)+th3(n)) + a2*sin(th2(n));

            calc_T_0_6_d = [
              r11, r12, r13, px(n);
              r21, r22, r23, py(n);
              r31, r32, r33, pz(n);
                0,   0,   0,  1;
            ];

            calc_th1(n) = atan( py(n) / px(n) );

            A = px(n) - d6*r13;
            B = py(n) - d6*r23;
            C = pz(n) - d6*r33;
            D = ( (A*cos(calc_th1(n)))^2 + 2*A*B*cos(calc_th1(n))*sin(calc_th1(n)) +
(B*sin(calc_th1(n)))^2 + C^2 - a3^2 - a2^2 ) / (2*a2*a3);

            calc_th3(n) = atan2( sqrt(1 - D^2), D );

            E = a3*cos(calc_th3(n)) + a2;
            F = a3*sin(calc_th3(n));
            ksi = sqrt( E^2 + F^2 );
            phi = atan2( F, E );
            K = C;

            calc_th2(n) = atan2( (K/ksi), sqrt( 1 - (K/ksi)^2 ) ) - phi;

            calc_th4(n) = -calc_th2(n) - calc_th3(n);

            calc_th5(n) = 0;
            index(n) = n;
                        n = n+1;




                    t3 = t3+50;
      end
```

```
                t3 = 0;
                t2 = t2+10;
    end

            t3 = 0;
            t2 = 0;
            t1 = t1+50;
end

plot(handles.Correctness_plot,index,(th4),index,(calc_th4));
h = legend(handles.Correctness_plot,'DKP Theta','IKP Calculated Theta',2);
dat = [px;py;pz;th1;th2;th3;th4;th1;calc_th2;calc_th3;calc_th4];
columnname =
{'X','Y','Z','Theta1','Theta2','Theta3','Theta4','Calc_theta1','Calc_Theta2','Calc_Theta3','Calc_The
ta4'};
set(uitable,'data',dat','ColumnName',columnname);


% --- Executes on button press in Theta_2.
function Theta_2_Callback(hObject, eventdata, handles)
% hObject    handle to Theta_2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
a2 = 419;
a3 = 364;
d6 = 267;
t1 = -178;
t2 = -35;
t3 = -170;
t4 = -178;
n = 1;
k=0;
while t1 < 178
    while t2 < 90
        while t3 < 162

            th5(n) =0;
            th1(n) = deg2rad(t1);
            th2(n) = deg2rad(t2);
            th3(n) = deg2rad(t3);
            th4(n) = -th2(n) - th3(n);
            r11 =  cos(th1(n))*cos(th2(n)+th3(n)+th4(n))*cos(th5(n)) + sin(th1(n))*sin(th5(n));
```

```
        r12 = -cos(th1(n))*cos(th2(n)+th3(n)+th4(n))*sin(th5(n)) + sin(th1(n))*cos(th5(n));
        r13 =  cos(th1(n))*sin(th2(n)+th3(n)+th4(n));
        px(n)  =  d6*cos(th1(n))*sin(th2(n)+th3(n)+th4(n)) +
cos(th1(n))*( a3*cos(th2(n)+th3(n)) + a2*cos(th2(n)) );
        r21 =  sin(th1(n))*cos(th2(n)+th3(n)+th4(n))*cos(th5(n)) - cos(th1(n))*sin(th5(n));
        r22 = -sin(th1(n))*cos(th2(n)+th3(n)+th4(n))*sin(th5(n)) - cos(th1(n))*cos(th5(n));
        r23 =  sin(th1(n))*sin(th2(n)+th3(n)+th4(n));
        py(n)  =  d6*sin(th1(n))*sin(th2(n)+th3(n)+th4(n)) +
sin(th1(n))*( a3*cos(th2(n)+th3(n)) + a2*cos(th2(n)) );
        r31 =  sin(th2(n)+th3(n)+th4(n))*cos(th5(n));
        r32 = -sin(th2(n)+th3(n)+th4(n))*sin(th5(n));
        r33 = -cos(th2(n)+th3(n)+th4(n));
        pz(n)  = -d6*cos(th2(n)+th3(n)+th4(n)) + a3*sin(th2(n)+th3(n)) + a2*sin(th2(n));

        calc_T_0_6_d = [
          r11, r12, r13, px(n);
          r21, r22, r23, py(n);
          r31, r32, r33, pz(n);
            0,   0,   0,  1;
        ];

        calc_th1(n) = atan( py(n) / px(n) );

        A = px(n) - d6*r13;
        B = py(n) - d6*r23;
        C = pz(n) - d6*r33;
        D = ( (A*cos(calc_th1(n)))^2 + 2*A*B*cos(calc_th1(n))*sin(calc_th1(n)) +
(B*sin(calc_th1(n)))^2 + C^2 - a3^2 - a2^2 ) / (2*a2*a3);

        calc_th3(n) = atan2( sqrt(1 - D^2), D );

        E = a3*cos(calc_th3(n)) + a2;
        F = a3*sin(calc_th3(n));
        ksi = sqrt( E^2 + F^2 );
        phi = atan2( F, E );
        K = C;

        calc_th2(n) = atan2( (K/ksi), sqrt( 1 - (K/ksi)^2 ) ) - phi;

        calc_th4(n) = -calc_th2(n) - calc_th3(n);

        calc_th5(n) = 0;
```

```
            index(n) = n;
                  n = n+1;



                t3 = t3+10;
        end

                t3 = 0;
                t2 = t2+10;
    end

            t3 = 0;
            t2 = 0;
            t1 = t1+10;
end


plot(handles.Correctness_plot,index,(th2),index,(calc_th2));
h = legend(handles.Correctness_plot,'DKP Theta','IKP Calculated Theta',3);
dat = [px;py;pz;th1;th2;th3;th4;th1;calc_th2;calc_th3;calc_th4];
columnname =
{'X','Y','Z','Theta1','Theta2','Theta3','Theta4','Calc_theta1','Calc_Theta2','Calc_Theta3','Calc_The
ta4'};
set(uitable,'data',dat','ColumnName',columnname);


% --- Executes on button press in Theta_5.
function Theta_5_Callback(hObject, eventdata, handles)
% hObject    handle to Theta_5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
a2 = 419;
a3 = 364;
d6 = 267;
t1 = -178;
t2 = -35;
t3 = -170;
t4 = -178;
n = 1;
k=0;
while t1 < 178
```

```
    while t2 < 90
        while t3 < 162

            th5(n) =0;
            th1(n) = deg2rad(t1);
            th2(n) = deg2rad(t2);
            th3(n) = deg2rad(t3);
            th4(n) = -th2(n) - th3(n);
            r11 =  cos(th1(n))*cos(th2(n)+th3(n)+th4(n))*cos(th5(n)) + sin(th1(n))*sin(th5(n));
            r12 = -cos(th1(n))*cos(th2(n)+th3(n)+th4(n))*sin(th5(n)) + sin(th1(n))*cos(th5(n));
            r13 =  cos(th1(n))*sin(th2(n)+th3(n)+th4(n));
            px(n)  =  d6*cos(th1(n))*sin(th2(n)+th3(n)+th4(n)) +
cos(th1(n))*( a3*cos(th2(n)+th3(n)) + a2*cos(th2(n)) );
            r21 =  sin(th1(n))*cos(th2(n)+th3(n)+th4(n))*cos(th5(n)) - cos(th1(n))*sin(th5(n));
            r22 = -sin(th1(n))*cos(th2(n)+th3(n)+th4(n))*sin(th5(n)) - cos(th1(n))*cos(th5(n));
            r23 =  sin(th1(n))*sin(th2(n)+th3(n)+th4(n));
            py(n)  =  d6*sin(th1(n))*sin(th2(n)+th3(n)+th4(n)) +
sin(th1(n))*( a3*cos(th2(n)+th3(n)) + a2*cos(th2(n)) );
            r31 =  sin(th2(n)+th3(n)+th4(n))*cos(th5(n));
            r32 = -sin(th2(n)+th3(n)+th4(n))*sin(th5(n));
            r33 = -cos(th2(n)+th3(n)+th4(n));
            pz(n)  = -d6*cos(th2(n)+th3(n)+th4(n)) + a3*sin(th2(n)+th3(n)) + a2*sin(th2(n));

            calc_T_0_6_d = [
              r11, r12, r13, px(n);
              r21, r22, r23, py(n);
              r31, r32, r33, pz(n);
                0,   0,   0,  1;
            ];

            calc_th1(n) = atan( py(n) / px(n) );

            A = px(n) - d6*r13;
            B = py(n) - d6*r23;
            C = pz(n) - d6*r33;
            D = ( (A*cos(calc_th1(n)))^2 + 2*A*B*cos(calc_th1(n))*sin(calc_th1(n)) +
(B*sin(calc_th1(n)))^2 + C^2 - a3^2 - a2^2 ) / (2*a2*a3);

            calc_th3(n) = atan2( sqrt(1 - D^2), D );

            E = a3*cos(calc_th3(n)) + a2;
            F = a3*sin(calc_th3(n));
```

```
            ksi = sqrt( E^2 + F^2 );
            phi = atan2( F, E );
            K = C;

            calc_th2(n) = atan2( (K/ksi), sqrt( 1 - (K/ksi)^2 ) ) - phi;

            calc_th4(n) = -calc_th2(n) - calc_th3(n);

            calc_th5(n) = 0;
            index(n) = n;
                        n = n+1;



                t3 = t3+50;
        end

              t3 = 0;
              t2 = t2+10;
    end

            t3 = 0;
            t2 = 0;
            t1 = t1+50;
end

plot(handles.Correctness_plot,index,(th5),index,(calc_th5));
h = legend(handles.Correctness_plot,'DKP Theta','IKP Calculated Theta',3);
dat = [px;py;pz;th1;th2;th3;th4;th1;calc_th2;calc_th3;calc_th4];
columnname =
{'X','Y','Z','Theta1','Theta2','Theta3','Theta4','Calc_theta1','Calc_Theta2','Calc_Theta3','Calc_Theta4'};
set(uitable,'data',dat','ColumnName',columnname);



function ip_angle2_Callback(hObject, eventdata, handles)
% hObject    handle to ip_angle2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ip_angle2 as text
```

```
%       str2double(get(hObject,'String')) returns contents of ip_angle2 as a double


% --- Executes during object creation, after setting all properties.
function ip_angle2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ip_angle2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ip_angle3_Callback(hObject, eventdata, handles)
% hObject    handle to ip_angle3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ip_angle3 as text
%       str2double(get(hObject,'String')) returns contents of ip_angle3 as a double


% --- Executes during object creation, after setting all properties.
function ip_angle3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ip_angle3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ip_angle4_Callback(hObject, eventdata, handles)
% hObject    handle to ip_angle4 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ip_angle4 as text
%        str2double(get(hObject,'String')) returns contents of ip_angle4 as a double


% --- Executes during object creation, after setting all properties.
function ip_angle4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ip_angle4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ip_angle5_Callback(hObject, eventdata, handles)
% hObject    handle to ip_angle5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ip_angle5 as text
%        str2double(get(hObject,'String')) returns contents of ip_angle5 as a double


% --- Executes during object creation, after setting all properties.
function ip_angle5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ip_angle5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function ip_positionY_Callback(hObject, eventdata, handles)
% hObject    handle to ip_positionY (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ip_positionY as text
%        str2double(get(hObject,'String')) returns contents of ip_positionY as a double


% --- Executes during object creation, after setting all properties.
function ip_positionY_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ip_positionY (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ip_positionZ_Callback(hObject, eventdata, handles)
% hObject    handle to ip_positionZ (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ip_positionZ as text
%        str2double(get(hObject,'String')) returns contents of ip_positionZ as a double


% --- Executes during object creation, after setting all properties.
function ip_positionZ_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ip_positionZ (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
```

```
    set(hObject,'BackgroundColor','white');
end
```

Verification in Matlab by Trajectory plotting:

DKP Function:

```
function a = DKPT(th1,th2,th3)

a2 = 419;
a3 = 364;
d6 = 267;

th1 = deg2rad(th1);
th2 = deg2rad(th2);
th3 = deg2rad(th3);
th4 = -th2-th3;
th5 = 0;

x = d6*cos(th1)*sin(th2+th3+th4) + cos(th1)*( a3*cos(th2+th3) +
a2*cos(th2) );
y = d6*sin(th1)*sin(th2+th3+th4) + sin(th1)*( a3*cos(th2+th3) +
a2*cos(th2) );
z = -d6*cos(th2+th3+th4) + a3*sin(th2+th3) + a2*sin(th2);

a = [x y z];
```

IKP Function:

```
function a = IKPT(x,y,z)

a2 = 419;
a3 = 364;
d6 = 267;

r11 =  1;
r12 =  0;
r13 =  0;
px  =  x;
r21 =  0;
r22 =  -1;
r23 =  0;
py  =  y;
r31 =  0;
r32 =  0;
```

```
r33 =  -1;
pz  =  z;

calc_T_0_6_d = [
   r11, r12, r13, px;
   r21, r22, r23, py;
   r31, r32, r33, pz;
     0,   0,   0,  1;
];

calc_th1 = atan2(y,x);

A = px - d6*r13;
B = py - d6*r23;
C = pz - d6*r33;
D = ( (A*cos(calc_th1))^2 + 2*A*B*cos(calc_th1)*sin(calc_th1) +
(B*sin(calc_th1))^2 + C^2 - a3^2 - a2^2 ) / (2*a2*a3);

calc_th3 = atan2( sqrt(1 - D^2), D );

E = a3*cos(calc_th3) + a2;
F = a3*sin(calc_th3);
ksi = sqrt( E^2 + F^2 );
phi = atan2( F, E );
K = C;

calc_th2 = atan2( (K/ksi), sqrt( 1 - (K/ksi)^2 ) ) - phi;

calc_th4 = -calc_th2 -calc_th3;

calc_th5 = 0;

calc_th1 = 360 * ( calc_th1/(2*pi) );
calc_th2 = 360 * ( calc_th2/(2*pi) );
calc_th3 = 360 * ( calc_th3/(2*pi) );
calc_th4 = -calc_th2 - calc_th3;
calc_th5 = 0;

a = [calc_th1 calc_th2 calc_th3 calc_th4 calc_th5];
```

Linear Trajectory:

```
th1 = [0:60];
th2 = [0:60];
```

```
th3 = [0:60];
n =1;

for n=1:61
    p(n,:) = DKPT(th1(n),th2(n),th3(n));
    a(n,:) = IKPT(p(n,1),p(n,2),p(n,3));
    t(n) = n
end

length(t)
length(th1)
length(a(:,1))
%DKP th1 and IKP th1 comparison
figure(1)
plot(t,th1,'r*',t,a(:,1),'b-');
xlabel('Time in Minutes');
ylabel('Joint THETA 1 angles of DKP and  IKP');
title('Comparison of DKP & IKP angles [THETA 1 in Degrees]');

%DKP th2 and IKP th2 comparison
figure(2)
plot(t,th2,t,a(:,2));
plot(t,th1,'y*',t,a(:,1),'g-');
xlabel('Time in Minutes');
ylabel('Joint THETA 2 angles of DKP and  IKP');
title('Comparison of DKP & IKP angles [THETA 2 in Degrees]');


figure(3)
plot(t,th3,'r*',t,a(:,3),'yo');
xlabel('Time in Minutes');
ylabel('Joint THETA 3 angles of DKP and  IKP');
title('Comparison of DKP & IKP angles [THETA 3 in Degrees]');


%3D plot of position trajectory
figure(4)
plot3(p(:,1),p(:,2),p(:,3),'r');
xlabel('X coordinates');
ylabel('Y coordinates');
zlabel('Z coordinates');
title('Trajectory of Points in 3D SPACE [all units are in mm]');


%time Vs Coordinates
```

```
figure(5)
plot(t,p(:,1),'r*');
xlabel('Time in Minutes');
ylabel('X coordinates');
title('Change of X coordinates with repsect to change in Time
[all units are in mm]');


figure(6)
plot(t,p(:,2),'b*');
xlabel('Time in Minutes');
ylabel('Y coordinates');
title('Change of Y coordinates with repsect to change in Time
[all units are in mm]');

figure(7)
plot(t,p(:,3),'go');
xlabel('Time in Minutes');
ylabel('Z coordinates');
title('Change of Z coordinates with repsect to change in Time
[all units are in mm]');
```

Parabolic Trajectory:

```
for n=1:61
    th1(n) = sqrt(4*n);
    th2(n) = sqrt(4*n);
    th3(n) = sqrt(4*n);
    p(n,:) = DKPT(th1(n),th2(n),th3(n));
    a(n,:) = IKPT(p(n,1),p(n,2),p(n,3));
    t(n) = n
end

length(t)
length(th1)
length(a(:,1))
%DKP th1 and IKP th1 comparison
figure(1)
plot(t,th1,'r*',t,a(:,1),'b-');
xlabel('Time in Minutes');
ylabel('Joint THETA 1 angles of DKP and  IKP');
title('Comparison of DKP & IKP angles [THETA 1 in Degrees]');

%DKP th2 and IKP th2 comparison
```

```
figure(2)
plot(t,th2,t,a(:,2));
plot(t,th1,'y*',t,a(:,1),'g-');
xlabel('Time in Minutes');
ylabel('Joint THETA 2 angles of DKP and  IKP');
title('Comparison of DKP & IKP angles [THETA 2 in Degrees]');


figure(3)
plot(t,th3,'r*',t,a(:,3),'yo');
xlabel('Time in Minutes');
ylabel('Joint THETA 3 angles of DKP and  IKP');
title('Comparison of DKP & IKP angles [THETA 3 in Degrees]');


%3D plot of position trajectory
figure(4)
plot3(p(:,1),p(:,2),p(:,3),'r');
xlabel('X coordinates');
ylabel('Y coordinates');
zlabel('Z coordinates');
title('Trajectory of Points in 3D SPACE [all units are in mm]');


%time Vs Coordinates
figure(5)
plot(t,p(:,1),'r*');
xlabel('Time in Minutes');
ylabel('X coordinates');
title('Change of X coordinates with repsect to change in Time
[all units are in mm]');


figure(6)
plot(t,p(:,2),'b*');
xlabel('Time in Minutes');
ylabel('Y coordinates');
title('Change of Y coordinates with repsect to change in Time
[all units are in mm]');

figure(7)
plot(t,p(:,3),'go');
xlabel('Time in Minutes');
ylabel('Z coordinates');
title('Change of Z coordinates with repsect to change in Time
```

```
[all units are in mm]');
```

Acknowledgement
(to be edited after completing task 3,4 and 5)

We would like to sincerely thank Prof. Dr. Teresa Zeilinska for providing us an opportunity to learn the Kinematics of the Seekur Jr Terabot S manipulator and for guiding our project.

We express our gratitude to Mr. Alex and Mr. Vibekananda Dutta for providing the required insights on the Seekur Jr Robot.