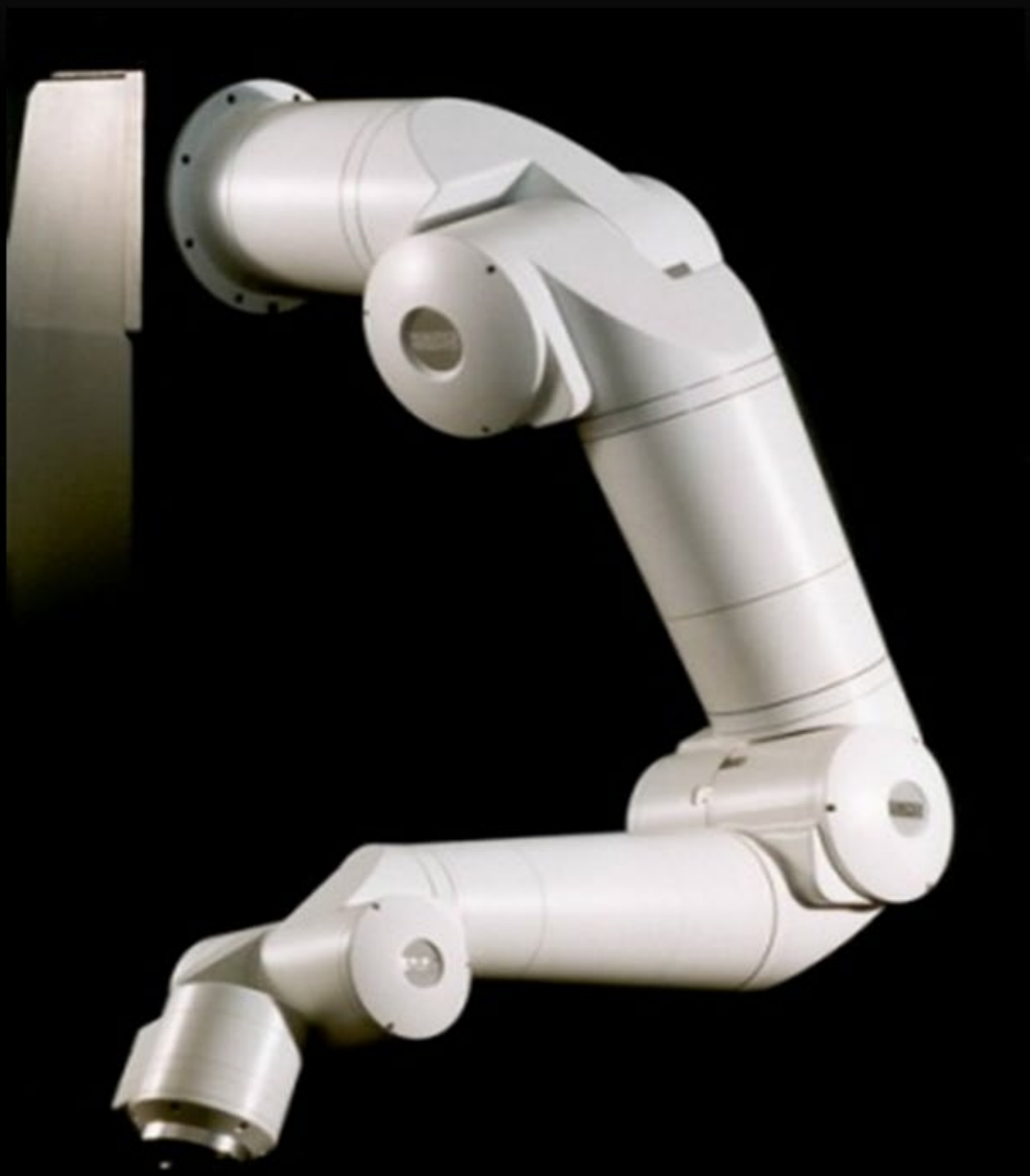
BOB WILLIAMS

Robot Mechanics



Robot Mechanics

Dr. Robert L. Williams II
Mechanical Engineering
Ohio University

NotesBook self-published for
EE/ME 4290/5290 Mechanics and Control of Robotic Manipulators

williar4@ohio.edu
people.ohio.edu/williams

Stewart Platform/PUMA Robot NASA LaRC

Robonaut NASA JSC

Resisting the unreasonable cost of textbooks since 2008

This EE/ME 4290/5290 NotesBook is augmented by the on-line EE/ME 4290/5290 Supplement.

Robot Mechanics

Author: **Robert L. Williams II, Ph.D.**
Mechanical Engineering
Ohio University
people.ohio.edu/williams

Chief Editor:	Dr. Bob
Sole Typist:	Dr. Bob
Marketing Director:	Dr. Bob
Production Manager:	Dr. Bob
Project Supervisor:	Dr. Bob
Technical Lead:	Dr. Bob
Bottle Washer:	Dr. Bob
Layout Expert:	Dr. Bob
Colour Consultant:	Dr. Bob
Cover Designer:	Dr. Bob
Best Boy:	Dr. Bob
MATLAB Guru:	Dr. Bob
Supplement Author:	Dr. Bob

Cover photo: Robotics Research Corporation modular, dexterous 7-dof 7R kinematically-redundant K-1207i robot.
http://www.robotics-research.com/

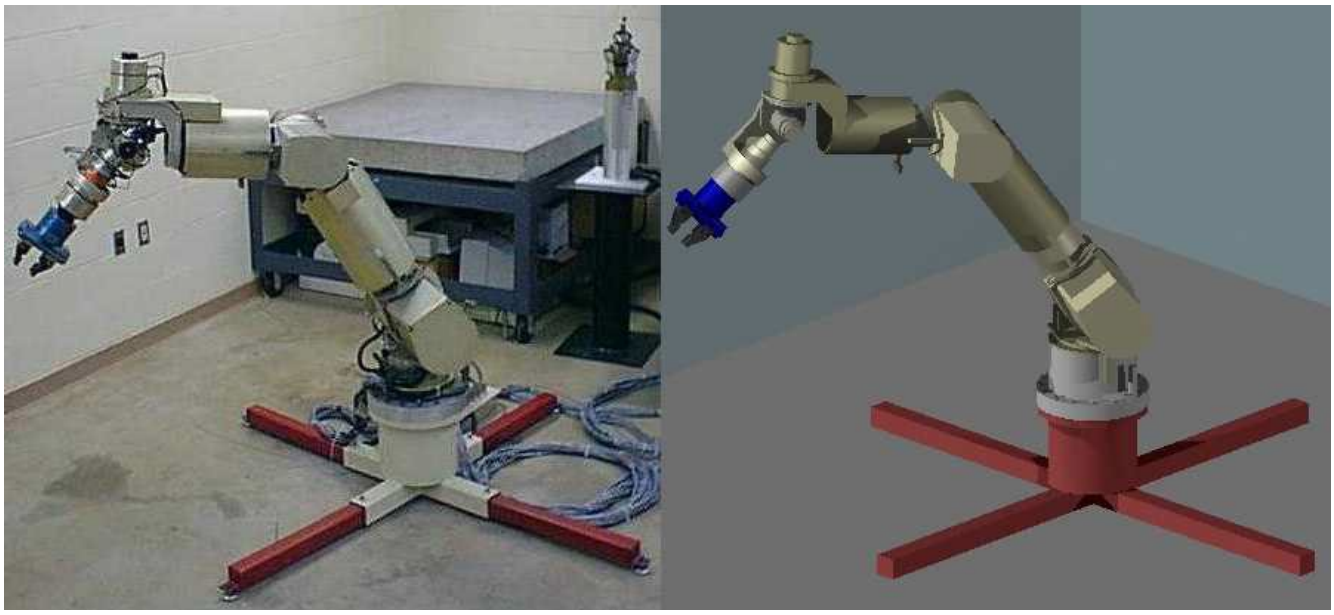
The body text is set in 12-pt Times New Roman, and the headings, sub-headings, and sub-sub-headings are set in 16-pt, 14-pt, and 12-pt Arial, respectively.

This NotesBook is intended for EE / ME 4290 / 5290 Mechanics and Control of Robotic Manipulators, a one-semester senior / graduate technical elective course in Electrical and Mechanical Engineering at Ohio University. Covered are mobility, motion description, orthonormal rotation matrices, Denavit-Hartenberg parameters, forward and inverse pose kinematics, trajectory generation, velocity kinematics, kinematic redundancy for serial-chain robotic manipulators, plus parallel robot mechanics. The on-line supplement further presents serial robot dynamics and control. MATLAB Software is used as a tool throughout for robot analyses and simulations. Warning: my NotesBook concept serves both as textbook and notebook – some equations, figures, and examples are blank and must be completed in class. Readers external to Ohio University are welcome with that caveat.

Keywords: robotics, manipulators, kinematics, dynamics, controls, kinematic redundancy, parallel robots, engineering, MATLAB

EE/ME 4290/5290 NotesBook Table of Contents Dr. Bob

1. Introduction



NASA 8-axis 8R Spatial Serial Manipulator, Hardware and CAD Model

1.1 Introduction to Robotics

Please see Dr. Bob's on-line Introduction to Robotics:
people.ohio.edu/williams/html/PDF/IntroRob.pdf

Notation

1) Uppercase variables are matrices or vectors. Lowercase variables are scalars.

2) Leading sub- and superscripts indicate which coordinate system the quantity is expressed in; e.g.:
^{A}P is a position vector P expressed in Cartesian coordinate frame $\{A\}$.
$^{A}P_{B}$ is a position vector from the origin of frame $\{A\}$ to the origin of frame $\{B\}$ expressed in the basis of frame $\{A\}$.
$^{A}_{B}R$ is an orthonormal rotation matrix giving the orientation of frame $\{B\}$ relative to frame $\{A\}$.
$^{A}_{B}T$ is a homogeneous transformation matrix giving the pose (position and orientation) of frame $\{B\}$ relative to frame $\{A\}$.

3) Trailing superscript indicates inverse or transpose of a matrix: R^{-1} or R^{T}.

4) Trailing subscripts indicate vector components (x,y,z) or are descriptive.

5) Trigonometric functions are often abbreviated:

$$\text{cosine}(\theta_i) = \cos\theta_i = c_i$$
$$\text{sine}(\theta_i) = \sin\theta_i = s_i$$
$$\text{tangent}(\theta_i) = \tan\theta_i = t_i$$

Required Mathematics

- Vectors, matrices, linear algebra
- Trigonometry, geometry, algebra
- Calculus
- Ordinary differential equations

Brief Review of Trigonometry and Calculus

All IV quadrants – degrees and radians

sine, cosine, tangent functions

inverse trigonometric functions – including **`atan2`**

Some trigonometric identities:

$$\sin(-a) = -\sin(a) \qquad \cos(a \pm b) = cacb \mp sasb \qquad \cos^2\theta_i + \sin^2\theta_i = 1$$

$$\cos(-a) = \cos(a) \qquad \sin(a \pm b) = sacb \pm casb$$

Law of Cosines: $C^2 = A^2 + B^2 - 2AB\cos c$ Law of Sines: $\frac{\sin a}{A} = \frac{\sin b}{B} = \frac{\sin c}{C}$

Some Calculus Relationships

product rule	**derivatives**	**integration**
$\frac{d(x(t)y(t))}{dt} = \frac{dx(t)}{dt}y(t) + x(t)\frac{dy(t)}{dt}$	$s = x(t)$	$a = \ddot{x}(t)$
chain rule	$v = \frac{ds}{dt} = \dot{x}(t)$	$v = \int a = \dot{x}(t)$
$\frac{df(x(t))}{dt} = \frac{df(x(t))}{dx(t)}\frac{dx(t)}{dt}$	$a = \frac{dv}{dt} = \frac{d^2s}{dt^2} = \ddot{x}(t)$	$s = \int v = \iint a = x(t)$

The same mathematics we learn for robot mechanics and control in this class apply equally well to aircraft and spacecraft mechanics and control, machine tools, vehicle motion, 3D engineering mechanics, biomechanics, and computer graphics, among other applications.

1.2 MATLAB Introduction

MATLAB is a general engineering analysis and simulation software. MATLAB stands for **MAT**rix **LAB**oratory. It was originally developed specifically for control systems simulation and design engineering, but it has grown over the years to cover many engineering and scientific fields. MATLAB is based on the C language, and its programming is vaguely C-like, but simpler. MATLAB is sold by Mathworks Inc. (www.mathworks.com) and Ohio University has a site license. For an extensive introduction to the MATLAB software, please see Dr. Bob's on-line MATLAB Primer:

people.ohio.edu/williams/html/PDF/MATLABPrimer.pdf

1.3 Mobility

Calculating the **mobility**, i.e. the number of **degrees-of-freedom (dof)**, for a robot is very important to our topic. This is based simply on counting the number of links and the number and types of joints.

Mobility M is the number of **degrees-of-freedom (dof)** which a robot, mechanism, or structure has.

Degrees-of-freedom (dof) – the minimum number of independent parameters required to fully specify the location of a device.

For planar and spatial devices, we have the following mobility classifications:

$M > 1$ robot	EE/ME 4290/5290 robotics
$M = 1$ mechanism	ME 3011 kinematics
$M = 0$ structure (statically-determinate)	ET 2200 statics
$M < 0$ structure (statically-indeterminate)	ET 2200 statics

For planar or spatial serial robots, to find the mobility, one may simply count the number of single degree-of-freedom joints. This is also equal to the number of independent motors to control the robot. More formally, we can also use Kutzbach's mobility equations for planar or spatial robots.

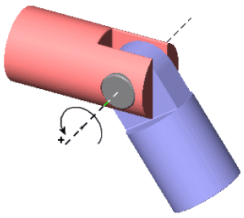
Kutzbach's Planar Mobility Equation

$$M = 3(N-1) - 2J_1 - 1J_2$$

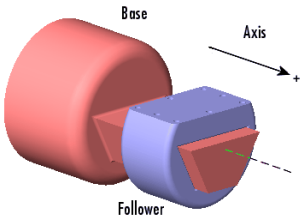
where

M is the mobility
N is the total number of links, including ground
J_1 is the number of one-dof joints
J_2 is the number of two-dof joints

J_1 – one-dof joints	revolute, prismatic
J_2 – two-dof joints	cam joint, gear joint, slotted-pin joint



revolute joint R – pin joint, turning pair



prismatic joint P – sliding pair

In general J_1 **R** and **P** joints are used much more extensively in practical robots than J_2 joints.

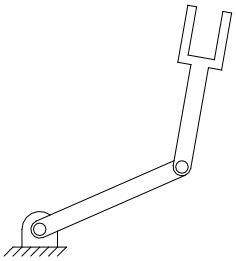
In Kutzbach's planar mobility equation, links give degrees of freedom and joints constrain or remove degrees of freedom. Each moving link has 3-dof in the plane – but the ground link is counted and hence must be subtracted in the equation. One-dof joints thus remove 2-dof in planar motion and two-dof joints thus remove 1-dof in planar motion.

A general planar *n*-link serial robot has *n*+1 links (including the fixed ground link), connected by *n* active 1-dof joints:

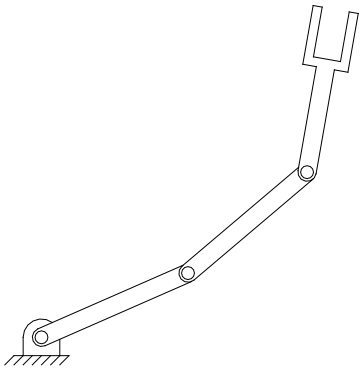
$$M = 3(n+1-1) - 2n - 1(0) = 3n - 2n = n \quad \text{dof}$$

This agrees with the previous statement, that we can count the number of active 1-dof joints to find the mobility. For parallel robots this does not apply, since some joints are active and others are passive.

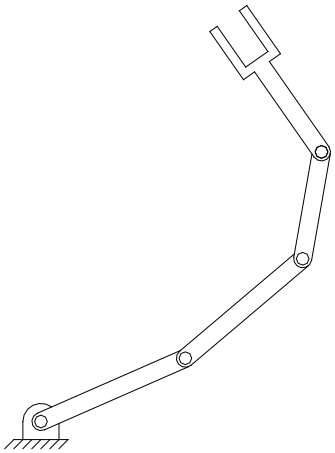
Planar robot mobility unsolved examples



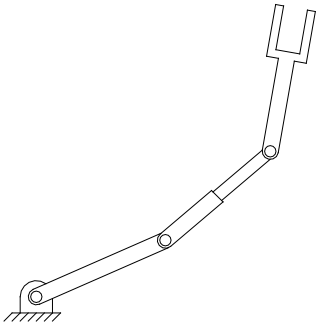
2R serial robot



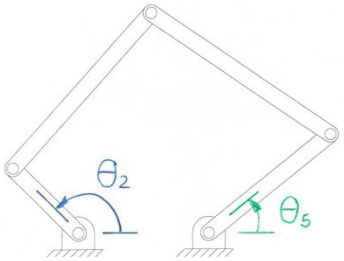
3R serial robot



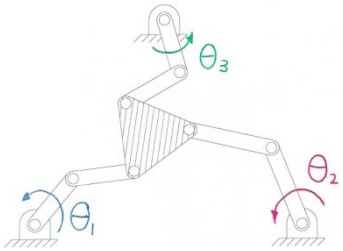
4R serial robot



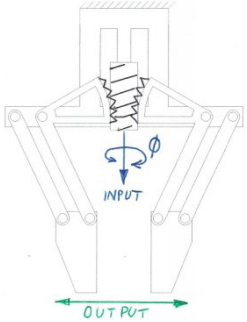
RRPR serial robot



5-bar RRRRR parallel robot



3-RRR parallel robot



Parallel jaw gripper mechanism

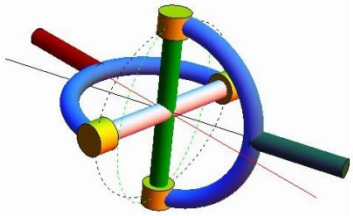
Kutzbach's Spatial Mobility Equation

$$M = 6(N-1) - 5J_1 - 4J_2 - 3J_3 - 2J_4 - 1J_5$$

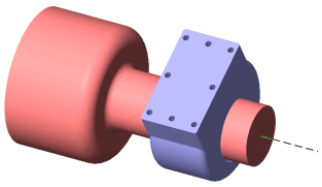
Where:

M is the mobility
N is the total number of links, including ground
J_1 is the number of one-dof joints
J_2 is the number of two-dof joints
J_3 is the number of three-dof joints
J_4 is the number of four-dof joints
J_5 is the number of five-dof joints

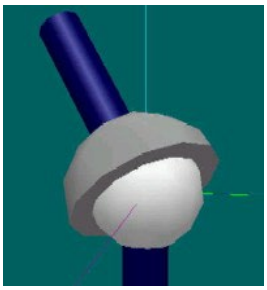
J_1 – one-dof joints:	revolute, prismatic, screw
J_2 – two-dof joints:	universal, cylindrical, gear joint
J_3 – three-dof joints:	spherical
J_4 – four-dof joints:	spherical in a slot
J_5 – five-dof joints:	spatial cam



2-dof universal joint U



2-dof cylindrical joint C



3-dof spherical joint S

In general J_1 **R** and **P**, J_2 **U**, and J_3 **S** joints are used much more extensively in practical robots than the other named joints.

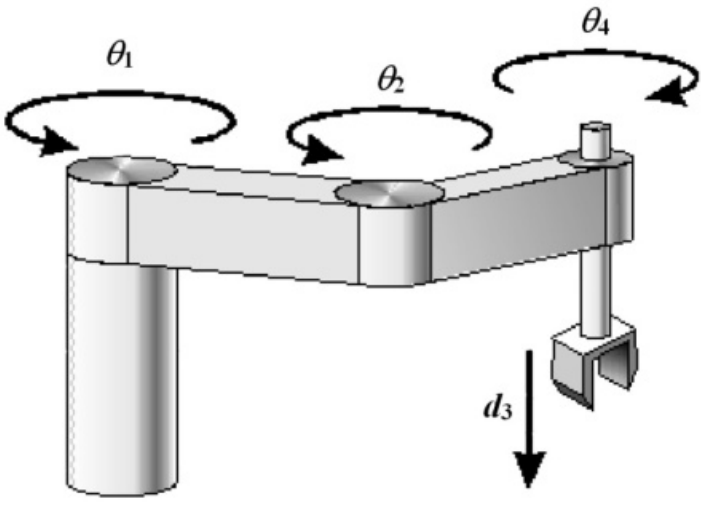
In Kutzbach's spatial mobility equation, links give degrees of freedom and joints constrain or remove degrees of freedom. Each moving link has 6-dof in 3D – but the ground link is counted and hence must be subtracted in the equation. One-dof joints remove 5-dof, two-dof joints remove 4-dof, three-dof joints remove 3-dof, and so on, in spatial motion.

A general spatial n-link serial robot has n+1 links (including the fixed ground link), connected by n active 1-dof joints:

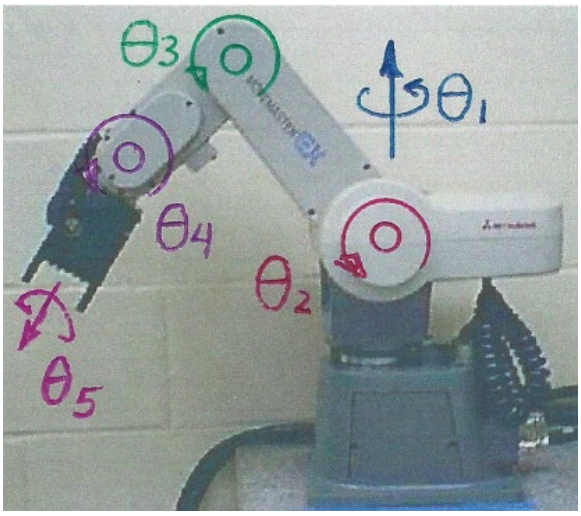
$$M = 6(n+1-1) - 5n - 4(0) - 3(0) - 2(0) - 1(0) = 6n - 5n = n \quad \text{dof}$$

This agrees with the previous statement, that we can count the number of active 1-dof joints to find the mobility. For parallel robots this does not apply. However, for a properly-designed parallel robot, counting the number of actuators will agree with the number of degrees-of-freedom, if the designer has done the proper job.

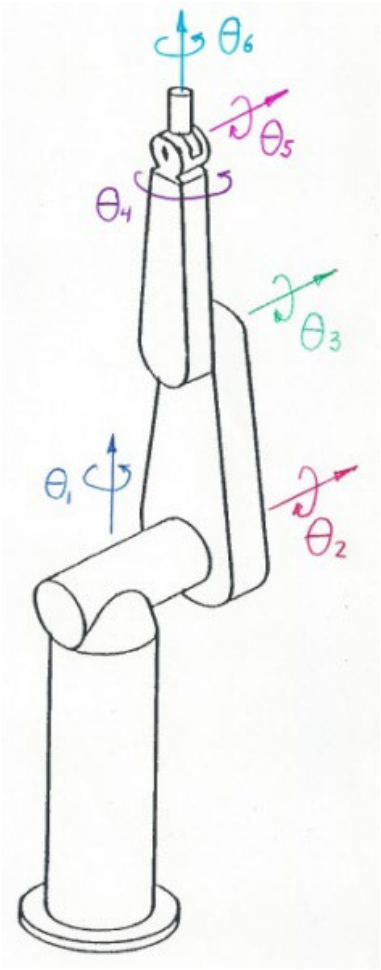
Spatial robot mobility unsolved examples



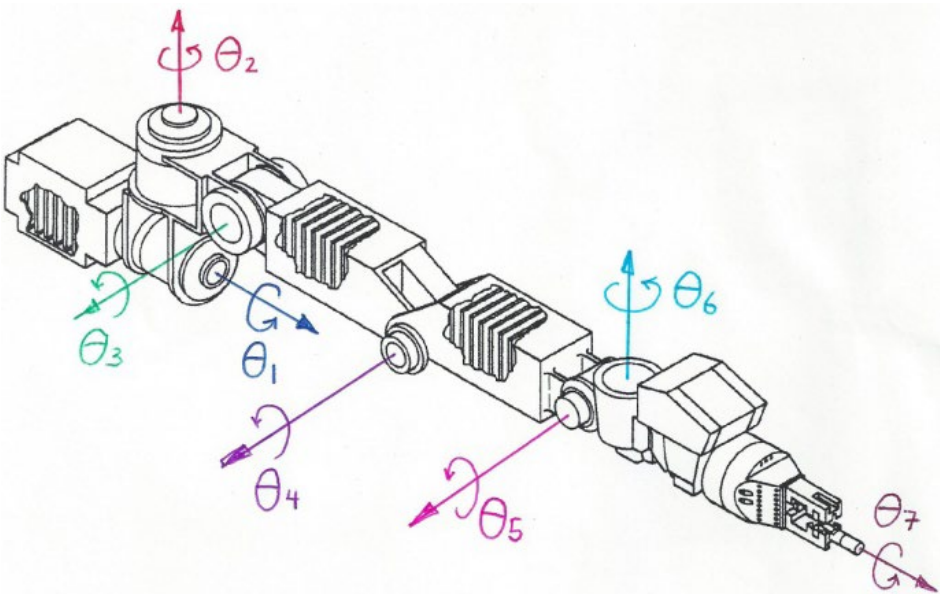
RRPR SCARA serial robot



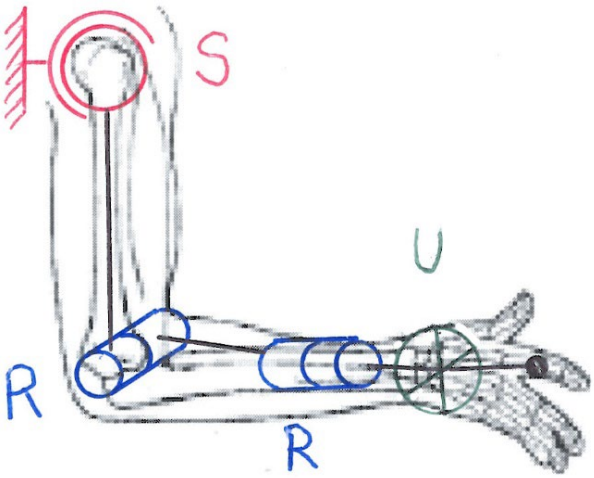
5R Mitsubishi serial robot



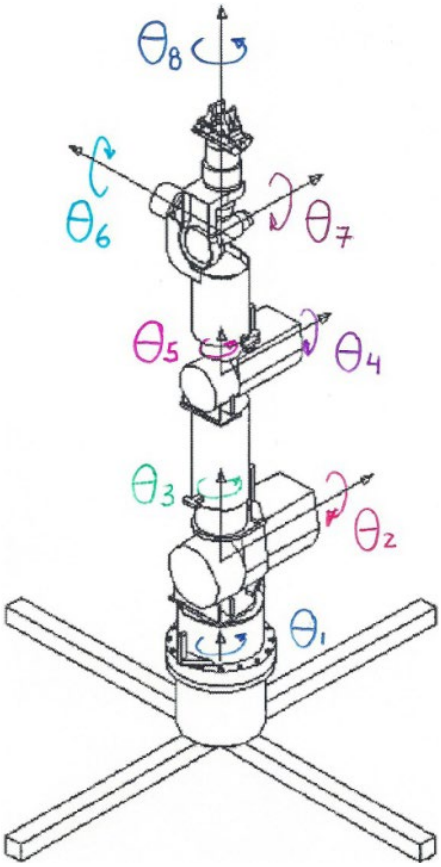
6R PUMA serial robot



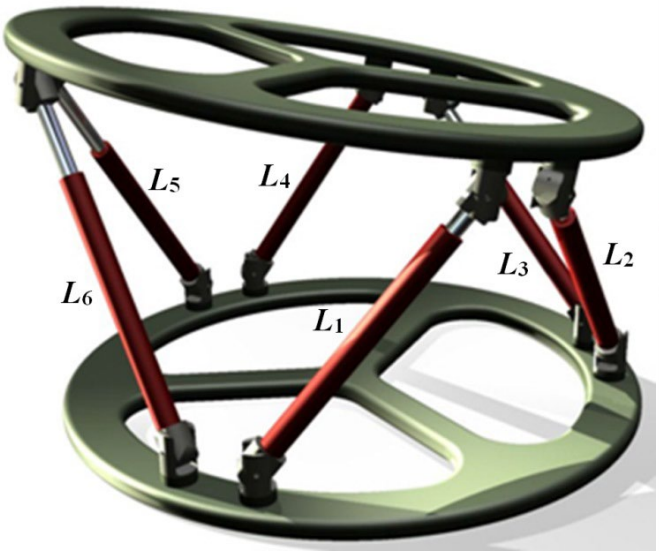
7R Flight Telerobotic Servicer (FTS) serial robot



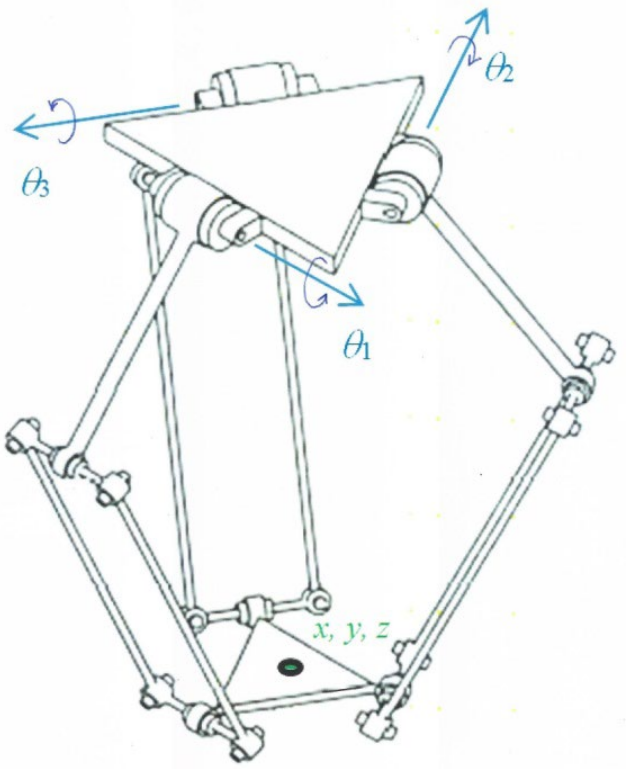
SRRU Human arm (ignore hand freedoms)



8R NASA ARMII serial robot



6-UPS Stewart Platform parallel robot



Translational 3-RUU Delta parallel robot

The **6-UPS** notation above on the left indicates there are six identical 'legs' connecting the moving platform to the base in the Stewart Platform. Quotes are shown around 'legs' since each 'leg' is actually two links connected by a prismatic joint. Each 'leg' is composed of a base-mounted universal joint **U**, prismatic joint **P**, and platform-mounted spherical joint **S**. The underbar on the prismatic joint indicates that this joint is actuated in each leg and the **U** and **S** joints are all passive.

Similarly, the **3-RUU** notation above on the right indicates there are three identical 'legs' connecting the moving platform to the base in the Delta Robot. Each 'leg' is composed of an active base-mounted revolute joint **R**, and two effective **U**-joints. The underbar on the revolute joint indicates that this joint is actuated in each leg and the two **U** joints are passive.

For some more unsolved mobility examples, please see Dr. Bob's on-line Atlas of Structures, Mechanisms, and Robots:

people.ohio.edu/williams/html/PDF/MechanismAtlas.pdf

MATLAB function to calculate mobility

```
% Function for calculating planar mobility
%   Dr. Bob, EE/ME 4290/5290

function M = dof(N,J1,J2)

M = 3*(N-1) - 2*J1 - 1*J2;
```

Usage:

```
mob = dof(4,4,0);    % for 4-bar and slider-crank mechanisms
```

Result:

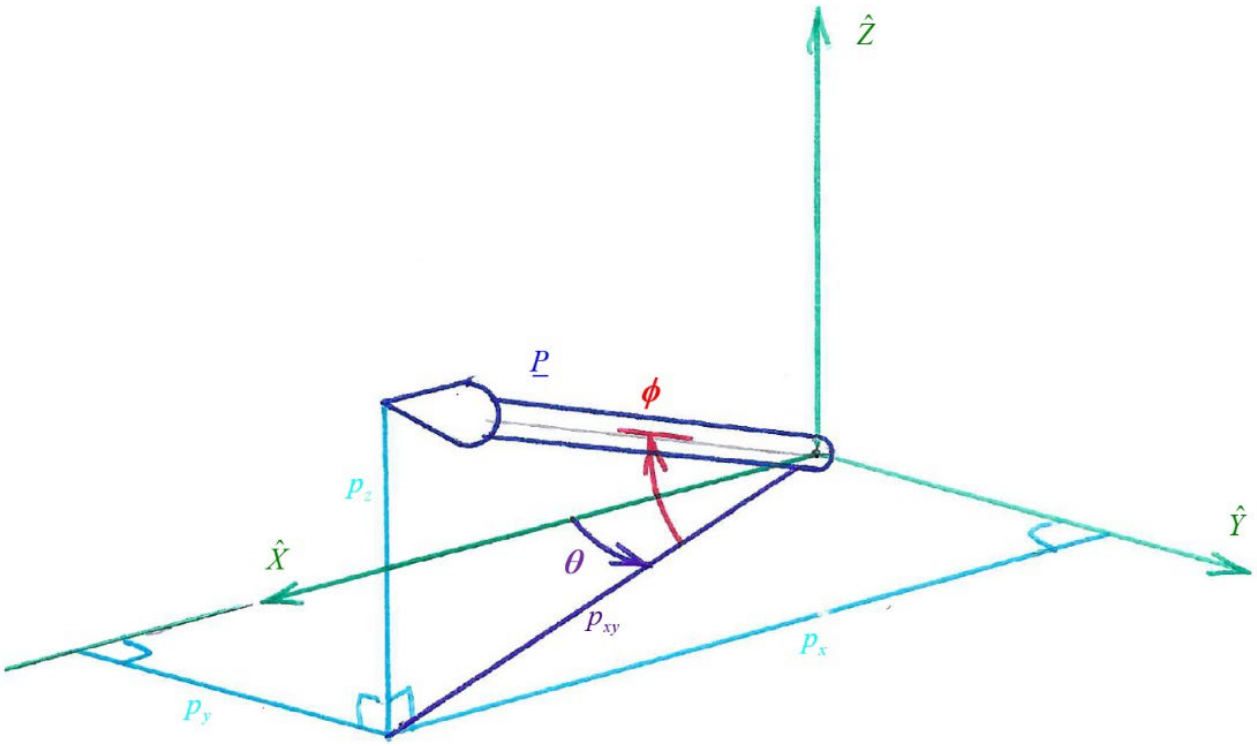
```
mob = 1
```

1.4 Matrices

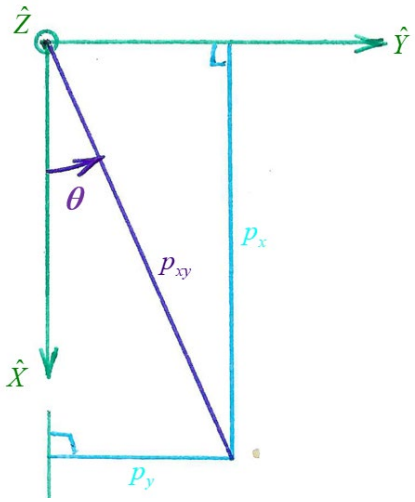
For a review of **matrices**, please see Dr. Bob's on-line review of Matrices and Linear Algebra (people.ohio.edu/williams/html/PDF/MatricesLinearAlgebra.pdf).

1.5 Vectors

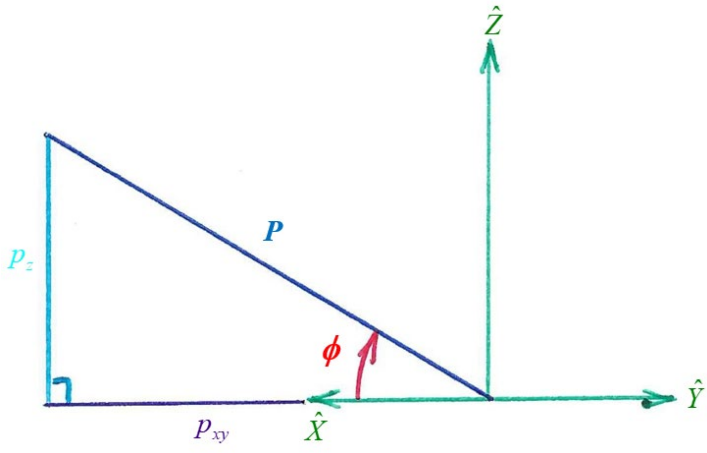
A vector is an arrow in 3D space with magnitude and direction. Vectors are used to represent translational position, velocity, acceleration, and force. Vectors are also used to represent angular velocity, angular acceleration, and torque (moment). Any vector has magnitude, direction (including sense), and point of application (which matters only for translational terms and force, but not for rotational terms and moment). Below is a figure for Cartesian/spherical representation of a 3D vector.



Top View



Pseudo Side View



Cartesian *XYZ* position vector representation:

$$\mathbf{P} = \underline{P} = p_x\hat{i} + p_y\hat{j} + p_z\hat{k} = \begin{Bmatrix} p_x \\ p_y \\ p_z \end{Bmatrix} = \begin{Bmatrix} p_x & p_y & p_z \end{Bmatrix}^T$$

Cartesian ↔ spherical representation

Below we present these transformations for vector description. These transformations also serve as our first forward and inverse kinematics robotics solutions, assuming a U-joint at the spherical vector origin, and a P-joint along the vector (i.e., the structure of a spatial **3-dof Spherical Robot**).

Forward Solution

Given the spherical coordinates $\{\theta \quad \phi \quad P\}$

Calculate the Cartesian coordinates $\{p_x \quad p_y \quad p_z\}^T$

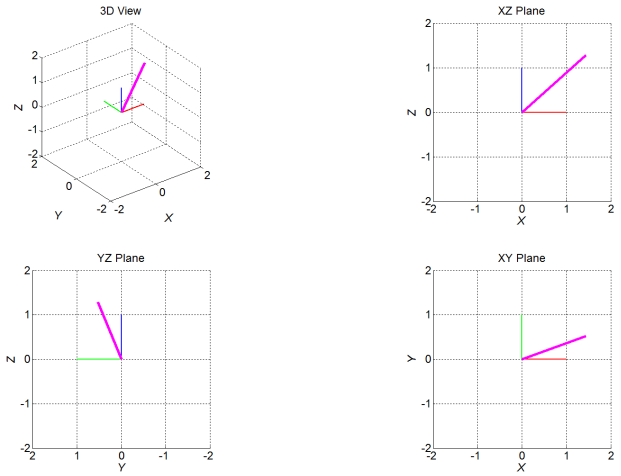
Inverse Solution: given the Cartesian coordinates $\{p_x \quad p_y \quad p_z\}^T$, calculate the spherical coordinates $\{\theta \quad \phi \quad P\}$.

There is a unique inverse solution (technically $\begin{Bmatrix}\theta+\pi\\ \phi+\pi\\ -P\end{Bmatrix}$ is also a solution).

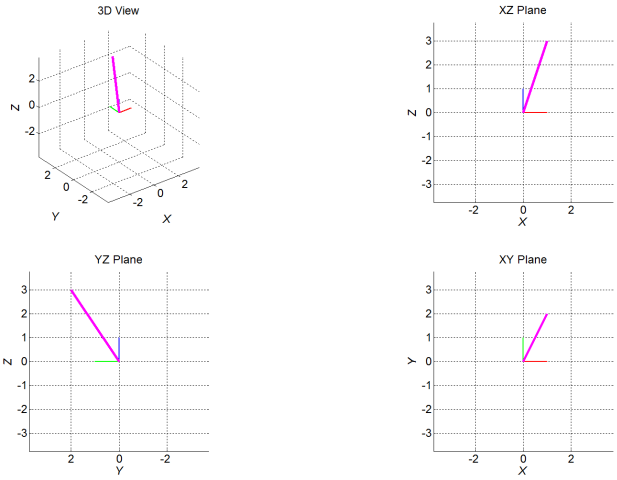
Forward example: $\begin{Bmatrix}\theta\\ \phi\\ P\end{Bmatrix}=\begin{Bmatrix}20^\circ\\ 40^\circ\\ 2\end{Bmatrix} \rightarrow \begin{Bmatrix}p_x\\ p_y\\ p_z\end{Bmatrix}=\begin{Bmatrix}1.44\\ 0.52\\ 1.29\end{Bmatrix}$

Inverse example: $\begin{Bmatrix}p_x\\ p_y\\ p_z\end{Bmatrix}=\begin{Bmatrix}1\\ 2\\ 3\end{Bmatrix} \rightarrow \begin{Bmatrix}\theta\\ \phi\\ P\end{Bmatrix}=\begin{Bmatrix}63.4^\circ\\ 53.3^\circ\\ 3.74\end{Bmatrix}$

Introduce the circular check.



Forward Example Graphic



Inverse Example Graphic

Vector Addition

Vectors add tail-to-head (or subtract head-to-tail); express components in the same coordinate frame. Vector addition yields a vector result.

$$\mathbf{A}+\mathbf{B}=\begin{Bmatrix}a_x\\a_y\\a_z\end{Bmatrix}+\begin{Bmatrix}b_x\\b_y\\b_z\end{Bmatrix}=\begin{Bmatrix}a_x+b_x\\a_y+b_y\\a_z+b_z\end{Bmatrix}$$

Vector addition is commutative.

$$\mathbf{A}+\mathbf{B}=\mathbf{B}+\mathbf{A}=\begin{Bmatrix}b_x+a_x\\b_y+a_y\\b_z+a_z\end{Bmatrix}=\begin{Bmatrix}a_x+b_x\\a_y+b_y\\a_z+b_z\end{Bmatrix}$$

Graphical interpretation of vector addition:

Vector Dot Product

The vector dot product is the projection of one vector onto another. The vector dot product yields a scalar result.

$$\mathbf{A}\bullet\mathbf{B}=\begin{Bmatrix}a_x\\a_y\\a_z\end{Bmatrix}\bullet\begin{Bmatrix}b_x\\b_y\\b_z\end{Bmatrix}=a_xb_x+a_yb_y+a_zb_z$$

The vector dot product is commutative:

$$\mathbf{A}\bullet\mathbf{B}=\mathbf{B}\bullet\mathbf{A}=\begin{Bmatrix}b_x\\b_y\\b_z\end{Bmatrix}\bullet\begin{Bmatrix}a_x\\a_y\\a_z\end{Bmatrix}=b_xa_x+b_ya_y+b_za_z$$

Here is another formula to calculate the vector dot product.

$$\mathbf{A}\bullet\mathbf{B}=\|\mathbf{A}\|\|\mathbf{B}\|\cos\phi$$

where ϕ is the angle between the two vectors (in a 3D plane containing both vectors **A** and **B**).

Yet another formula to calculate the vector dot product uses the transpose and matrix multiplication. This is similar to the first dot product formula above.

$$\mathbf{A}\bullet\mathbf{B}=\mathbf{A}^T\mathbf{B}=\begin{Bmatrix}a_x & a_y & a_z\end{Bmatrix}\begin{Bmatrix}b_x\\b_y\\b_z\end{Bmatrix}=a_xb_x+a_yb_y+a_zb_z$$

Graphical interpretation of the vector dot product:

Vector Cross Product

The vector cross product of two vectors gives a third vector mutually perpendicular to the original two vectors. The vector cross product yields a vector result.

$$\mathbf{A}\times\mathbf{B}=\begin{Bmatrix}a_x\\a_y\\a_z\end{Bmatrix}\times\begin{Bmatrix}b_x\\b_y\\b_z\end{Bmatrix}=\begin{vmatrix}\hat{i}&\hat{j}&\hat{k}\\a_x&a_y&a_z\\b_x&b_y&b_z\end{vmatrix}$$

$$=\hat{i}\left(a_yb_z-a_zb_y\right)-\hat{j}\left(a_xb_z-a_zb_x\right)+\hat{k}\left(a_xb_y-a_yb_x\right)$$

$$=\begin{Bmatrix}a_yb_z-a_zb_y\\a_zb_x-a_xb_z\\a_xb_y-a_yb_x\end{Bmatrix}$$

The vector cross product is **NOT** commutative. However, there is a close relationship ($\mathbf{B}\times\mathbf{A}$ is equal in magnitude and opposite direction to $\mathbf{A}\times\mathbf{B}$).

$$\mathbf{A}\times\mathbf{B}\neq\mathbf{B}\times\mathbf{A}$$
$$\mathbf{B}\times\mathbf{A}=-\mathbf{A}\times\mathbf{B}$$

Here is another formula to calculate the vector cross product.

$$\left\|\mathbf{A}\times\mathbf{B}\right\|=\left\|\mathbf{A}\right\|\left\|\mathbf{B}\right\|\sin\phi$$

where ϕ is the angle between the two vectors (in a 3D plane containing both vectors **A** and **B**). The resulting cross product direction is found via the right-hand-rule: Put your right hand fingers along the first vector **A**, rotate it into the second vector **B**; your right thumb, perpendicular to both vectors, gives the direction of $\mathbf{A}\times\mathbf{B}$.

Here is yet another formula to calculate the vector cross product that uses matrix multiplication.

$$\mathbf{A}\times\mathbf{B}=\begin{bmatrix}0&-a_z&a_y\\a_z&0&-a_x\\-a_y&a_x&0\end{bmatrix}\begin{Bmatrix}b_x\\b_y\\b_z\end{Bmatrix}=\begin{Bmatrix}a_yb_z-a_zb_y\\a_zb_x-a_xb_z\\a_xb_y-a_yb_x\end{Bmatrix}$$

where the skew-symmetric cross product operator matrix is:

$$\mathbf{A}\times=\begin{bmatrix}0&-a_z&a_y\\a_z&0&-a_x\\-a_y&a_x&0\end{bmatrix}$$

Graphical interpretation of the vector cross product:

Vector Examples $\mathbf{A} = \begin{Bmatrix} 1 \\ 2 \\ 3 \end{Bmatrix}$ $\mathbf{B} = \begin{Bmatrix} 4 \\ 5 \\ 6 \end{Bmatrix}$

Addition: $\mathbf{A} + \mathbf{B} = \mathbf{B} + \mathbf{A} = \begin{Bmatrix} 5 \\ 7 \\ 9 \end{Bmatrix}$ **Drawing:**

Dot Product: $\mathbf{A} \bullet \mathbf{B} = \mathbf{B} \bullet \mathbf{A} = 32$

Cross Product: $\mathbf{A} \times \mathbf{B} = \begin{Bmatrix} -3 \\ 6 \\ -3 \end{Bmatrix}$ $\mathbf{B} \times \mathbf{A} = \begin{Bmatrix} 3 \\ -6 \\ 3 \end{Bmatrix}$

The Same Vector Examples in MATLAB

```
%
% Vectors.m - vector examples
%          Dr. Bob, EE/ME 4290/5290

clear; clc;

A = [1;2;3];          % Define two 3x1 vectors
B = [4;5;6];
sum1 = A+B            % Vector addition
sum2 = B+A
dot1 = dot(A,B)       % Vector dot product
dot2 = dot(B,A)
cross1 = cross(A,B) % Vector cross product
cross2 = cross(B,A)
```

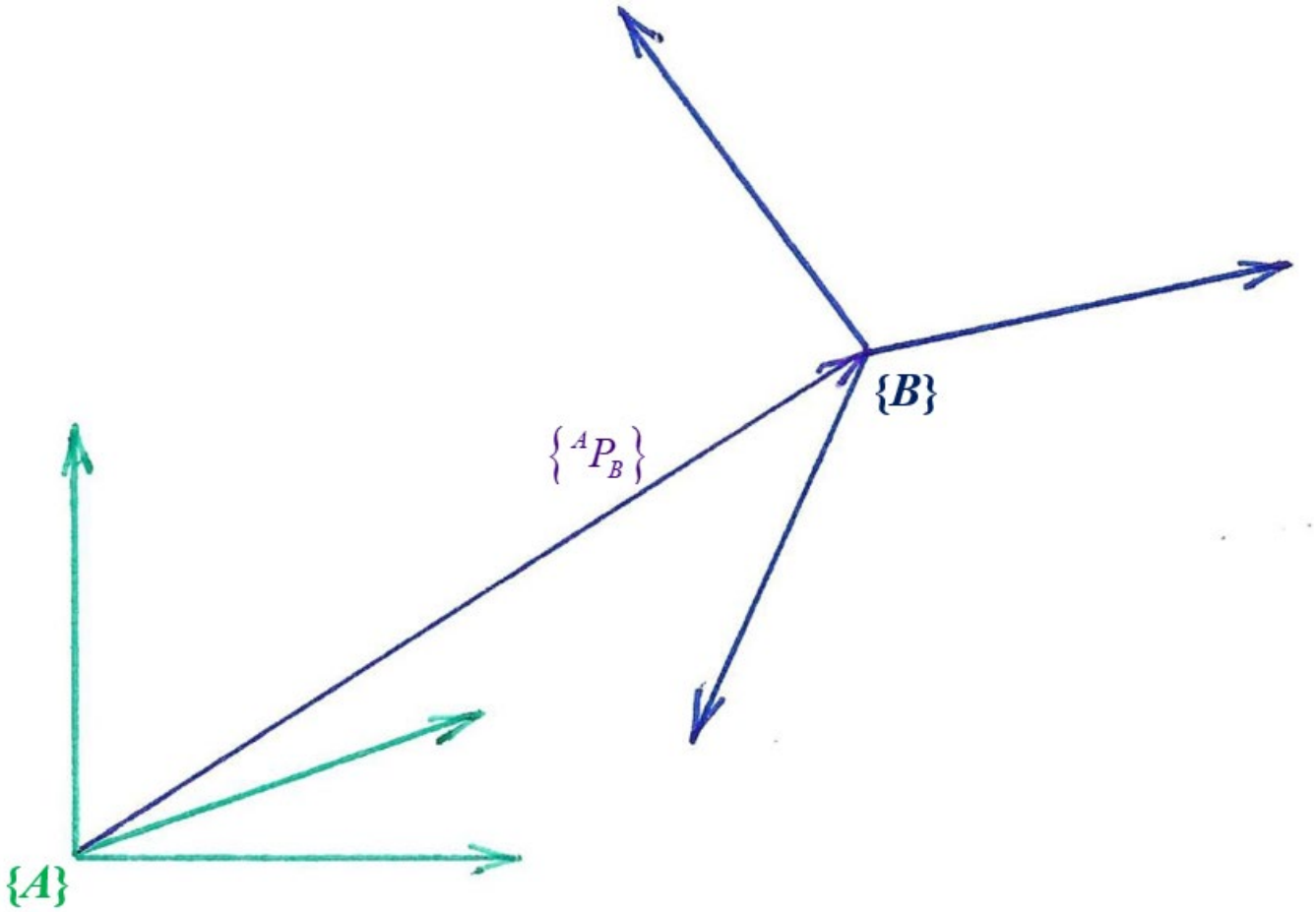
Output of Vectors.m

```
sum1 =     5
           7
           9

dot1 =    32

cross1 = -3
           6
          -3
```

```
sum2 =     5
           7
           9

dot2 =    32

cross2 =   3
          -6
           3
```

2. 3D Motion Descriptions and Transformations

In robotics analysis and control we need to describe the spatial position and orientation (**Pose**) of various links, tools, end-effectors, sensors, environment locations, and workpieces. We attach a separate, independent right-handed Cartesian coordinate frame to each moving body of interest. These frames are fixed in the moving body and are moved relative to each other via active robot joints. Here is a drawing of moving frame $\{B\}$ relative to reference frame $\{A\}$, representing any pair of such frames:



$\{{}^{A}P_{B}\}$ and $\left[{}^{A}_{B}R\right]$ are required to describe the position (translations) and orientation (rotations) of $\{B\}$ with respect to $\{A\}$ – together this is called the **Pose**.

Position $X = \{x \quad y \quad z \quad \theta_x \quad \theta_y \quad \theta_z\}^T$

Note: position is a vector but rotation (*pitch*, *yaw*, *roll* or $\theta_x, \theta_y, \theta_z$) IS NOT A VECTOR.

Velocity $\dot{X} = \{\dot{x} \quad \dot{y} \quad \dot{z} \quad \omega_x \quad \omega_y \quad \omega_z\}^T$

Acceleration $\ddot{X} = \{\ddot{x} \quad \ddot{y} \quad \ddot{z} \quad \alpha_x \quad \alpha_y \quad \alpha_z\}^T$

Velocity and acceleration are vectors, both translational and rotational.

The 3D representation of orientation (rotations) CANNOT BE EXPRESSED AS A VECTOR.

Position

A point has no orientation, just position that can be described by vectors in 3D space. Position vector addition is commutative, i.e. $\underline{P}_1 + \underline{P}_2 = \underline{P}_2 + \underline{P}_1$

$$\underline{P} = \begin{Bmatrix} x \\ y \\ z \end{Bmatrix}$$

Orientation

Description of 3D orientations (rotations, attitude) are much harder since there is no associated vector description. Orientation will be handled via 3x3 orthonormal rotation matrices $\left[{}^{A}_{B}R \right]$.

2.1 Description of 3D Orientation

2.1.1 Orthonormal Rotation Matrices

Orientation

For orientation, we need two frames; let us describe the orientation of $\{B\}$ with respect to $\{A\}$. Spatial (3D) orientations have 3 dof (e.g. roll, pitch, yaw – airplane coordinates).

For airplane orientation we have two major choices (three rotations about the fixed Cartesian coordinate system axes or three rotations about the moving Cartesian coordinate system axes).

Planar - easy, just one-dof θ pitch; this is a vector representation.

Spatial - hard, artificial, cannot be a vector representation. 3D rotation representation options:

- **3 Rotations about fixed frame**
- **3 Rotations about moving frame** (Euler Angles)
- **Axis – angle rotation** (Screw Theory)
- **Quaternions** (Similar to Axis – angle, but requires 4 parameters, not 3; see Section 2.1.2)

All descriptions lead to the Orthonormal Rotation Matrix. The descriptions listed above are rather artificial since many possibilities lead to the same unique Orthonormal Rotation Matrix describing the 3D, 3-dof orientation of $\{B\}$ with respect to $\{A\}$.

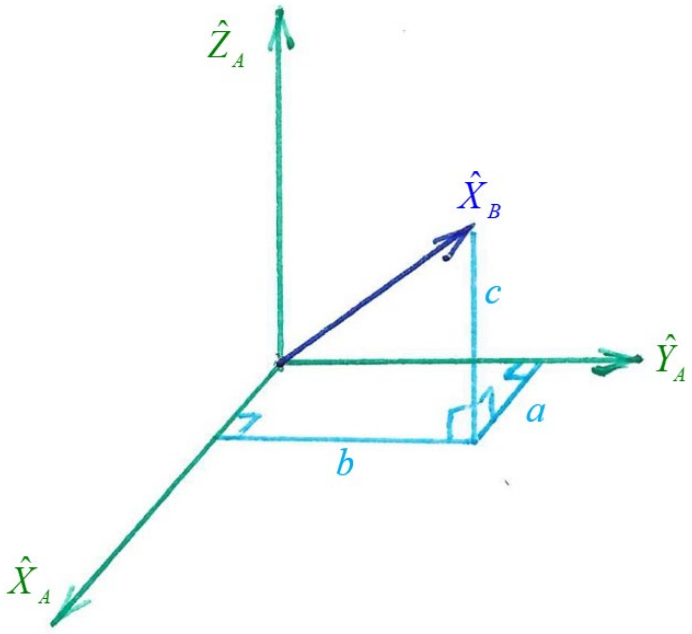
Dr. Bob's 12-Step Program for Orthonormal Rotation Matrices

1. Demonstrate that spatial rotations are not described by vectors (spatial rotations are not commutative); example:

2. Form of Orthonormal Rotation Matrices

$\left[{}^{A}_{B}R \right]$ is a 3x3 matrix describing the orientation of frame $\{B\}$ with respect to frame $\{A\}$. Remember 3D orientations have only 3 dof (e.g. roll, pitch, yaw); however, the rotation matrix has 9 elements – why? (Let's answer this later in Step 9, Constraints on $\left[{}^{A}_{B}R \right]$.)

Demonstrate projection of $\{B\}$ axes onto $\{A\}$ basis; e.g. $\left\{ {}^{A}\hat{X}_{B} \right\}$:



What vector operation is used for projecting one vector onto another?

$$\left[{}^{A}_{B}R \right] = \begin{bmatrix} \hat{X}_B \cdot \hat{X}_A & \hat{Y}_B \cdot \hat{X}_A & \hat{Z}_B \cdot \hat{X}_A \\ \hat{X}_B \cdot \hat{Y}_A & \hat{Y}_B \cdot \hat{Y}_A & \hat{Z}_B \cdot \hat{Y}_A \\ \hat{X}_B \cdot \hat{Z}_A & \hat{Y}_B \cdot \hat{Z}_A & \hat{Z}_B \cdot \hat{Z}_A \end{bmatrix}$$
$$= \begin{bmatrix} | & | & | \\ {}^{A}\hat{X}_B & {}^{A}\hat{Y}_B & {}^{A}\hat{Z}_B \\ | & | & | \end{bmatrix}$$

Recall a formula to calculate the vector dot product:

$$\hat{X}_B \cdot \hat{X}_A = \left\| \hat{X}_B \right\| \left\| \hat{X}_A \right\| \cos\gamma_1 = (1)(1)\cos\gamma_1 = \cos\gamma_1$$

where γ_1 is the angle between $\hat{X}_B$ and $\hat{X}_A$, which are both unit vectors (of length 1). So the Orthonormal Rotation Matrix is also called the Direction Cosine Matrix. Note the dot product is commutative, so another way to define it is row-wise:

$$\left[{}^{A}_{B}R \right] = \begin{bmatrix} \hat{X}_A \cdot \hat{X}_B & \hat{X}_A \cdot \hat{Y}_B & \hat{X}_A \cdot \hat{Z}_B \\ \hat{Y}_A \cdot \hat{X}_B & \hat{Y}_A \cdot \hat{Y}_B & \hat{Y}_A \cdot \hat{Z}_B \\ \hat{Z}_A \cdot \hat{X}_B & \hat{Z}_A \cdot \hat{Y}_B & \hat{Z}_A \cdot \hat{Z}_B \end{bmatrix} = \begin{bmatrix} - & {}^{B}\hat{X}_A & - \\ - & {}^{B}\hat{Y}_A & - \\ - & {}^{B}\hat{Z}_A & - \end{bmatrix}$$

3. Inverse Orthonormal Rotation Matrix

The inverse Orthonormal Rotation matrix may be found merely by swapping our reference and moving frames. $\{B\}$ was the moving frame and now becomes the reference frame; $\{A\}$ was the reference frame and now becomes the moving frame of interest. The inverse of $\left[{}^{A}_{B}R\right]$ describes the orientation of $\{A\}$ with respect to $\{B\}$:

$$\left[{}^{B}_{A}R\right]=\left[{}^{A}_{B}R\right]^{-1}$$

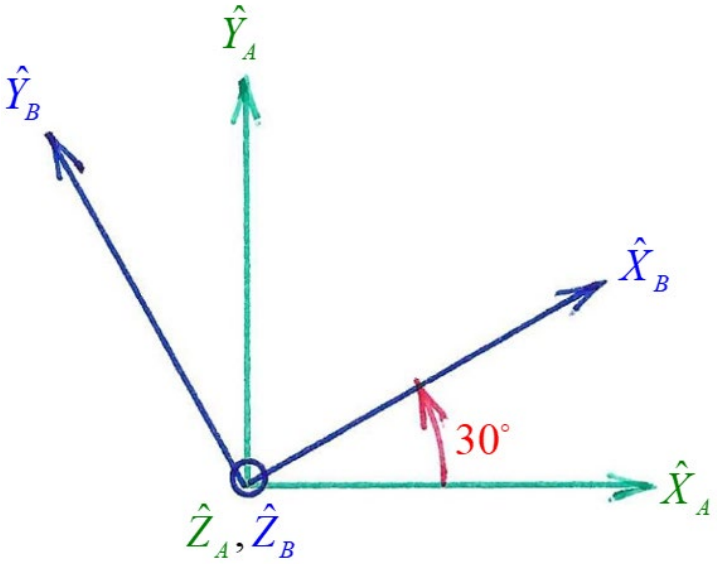
How do we calculate this? One could use the standard numerical matrix inversion function **`inv`** in MATLAB. However, this is computationally expensive, often subject to singularities (though not in this case), and fails to take advantage of the inherent structure of the Orthonormal Rotation Matrix. The form for the inverse rotation matrix is:

$$\left[{}^{A}_{B}R\right]^{-1}=\left[{}^{B}_{A}R\right]=\begin{bmatrix}\hat{X}_A\cdot\hat{X}_B & \hat{Y}_A\cdot\hat{X}_B & \hat{Z}_A\cdot\hat{X}_B\\ \hat{X}_A\cdot\hat{Y}_B & \hat{Y}_A\cdot\hat{Y}_B & \hat{Z}_A\cdot\hat{Y}_B\\ \hat{X}_A\cdot\hat{Z}_B & \hat{Y}_A\cdot\hat{Z}_B & \hat{Z}_A\cdot\hat{Z}_B\end{bmatrix}=\begin{bmatrix}| & | & |\\ {}^{B}\hat{X}_A & {}^{B}\hat{Y}_A & {}^{B}\hat{Z}_A\\ | & | & |\end{bmatrix}$$

Comparing this form with the relationship from the previous page, we see:

So, to find the inverse of an Orthonormal Rotation Matrix, we need only take the transpose, which requires no computation (only exchanging memory locations) and is never subject to singularities. This is a ***beautiful property*** and only holds for this very special orthonormal type of matrix.

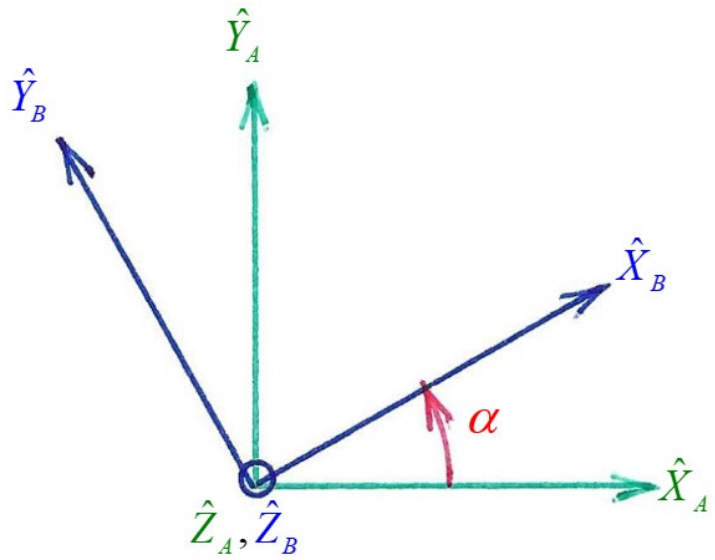
4. Simple planar example. For the following sketch, find $\left[{}^{A}_{B}R\right]$; also find $\left[{}^{B}_{A}R\right]=\left[{}^{A}_{B}R\right]^{-1}$:



5. Single axis, single angle rotations

To describe general 3D orientations we will use a series of three single axis, single angle rotations. Therefore, we need to be able to understand and derive the rotation matrix representing a single angle rotation about a single primary axis, for the **X**, **Y**, and **Z** axes.

We now derive the rotation matrix representing the orientation of $\{B\}$ with respect to $\{A\}$ when $\{B\}$ is rotated about the **Z** axis of $\{A\}$ by angle α. Here is the associated figure:



From the geometry of this situation, the $[R_Z(\alpha)]$ formula, which in itself is a valid Orthonormal Rotation Matrix, is:

The student is left to derive in a similar manner $[R_Y(\beta)]$ and $[R_X(\gamma)]$, which are also required.

$$[R_Y(\beta)] = \begin{bmatrix} c\beta & 0 & s\beta \\ 0 & 1 & 0 \\ -s\beta & 0 & c\beta \end{bmatrix}$$

$$[R_X(\gamma)] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\gamma & -s\gamma \\ 0 & s\gamma & c\gamma \end{bmatrix}$$

6. Description of general (compound) spatial orientation

Description of orientation is rather artificial since many methods will lead to the same numerical description for a given $\left[{}^{A}_{B}R \right]$. We will use a series of three single axis, single angle rotations to build an overall 3D orientation description. We can perform ensuing rotations either about the **fixed XYZ** axes or about the **moving XYZ** axes. There are twelve distinct ways to describe 3 general rotations about **fixed axes** and also twelve distinct ways to describe 3 general rotations about **moving axes** (called Euler angles). Here are the 12 distinct valid permutations for either fixed axes or Euler angle conventions.

X-Y-X	**Y-X-Y**	**Z-X-Y**
X-Y-Z	**Y-X-Z**	**Z-X-Z**
X-Z-X	**Y-Z-X**	**Z-Y-X**
X-Z-Y	**Y-Z-Y**	**Z-Y-Z**

Note: **X-Y-Y**, etc., are not allowed since a repeated axis rotation such as **Y-Y** is not independent, but is equivalent to one **Y** rotation (simply adding the two sequential **Y** rotations), only providing 2-dof when we need 3-dof.

Review of combinations and permutations.

The table above gives the 12 surviving permutations of all **X-Y-Z** possibilities when the order matters.

Permutations – all possible sets from a group where the order matters.

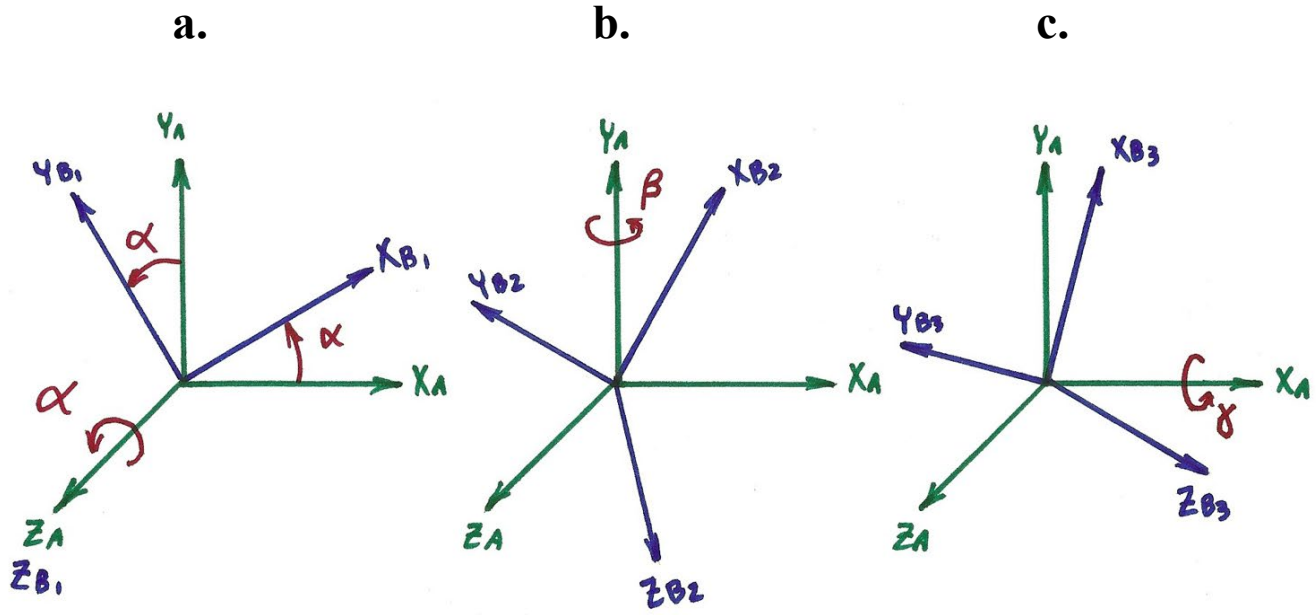
> Three group members taken three ways yields $3^3 = 27$ permutations. 15 of these permutations are not allowed, whenever two sequential axes are identical (e.g. the **X-Y-Y** case described above). This leaves 12 valid permutations as displayed in the table above.

Combinations – all possible sets from a group where the order does not matter.

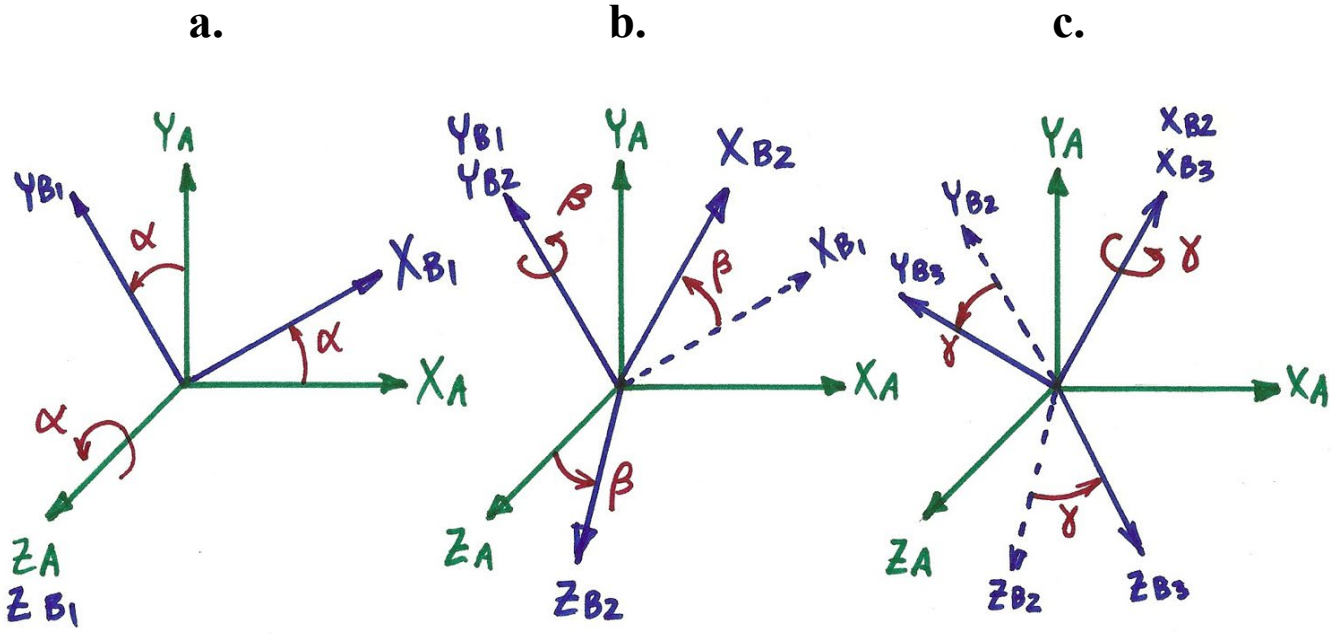
> In this case, there is only one combination, **X-Y-Z**, since order does not matter for combinations. If repeats are allowed as in the permutations above, then there are 9 total combinations (permutation is the applicable concept for 3D rotations since the order matters):

X-X-X	**X-Y-Y**	**Y-Z-Y**
X-X-Y	**X-Y-Z**	**Y-Z-Z**
X-X-Z	**Y-Y-Y**	**Z-Z-Z**

Now we will demonstrate fixed vs. moving axes (Euler) convention for **Z-Y-X** (α–β–γ) rotations. Which is better for the aircraft and robotics orientation description cases?



Z-Y-X (α–β–γ) Fixed Axes Convention



Z-Y-X (α–β–γ) Moving Axes (Euler Angles) Convention

For both cases, start with moving frame $\{B\}$ and reference frame $\{A\}$ initially aligned in orientation (and the same origin for all rotations).

7. Euler Angles. Since there are so many artificial methods to reach the same $\left[{}^{A}_{B}R \right]$ result, let us choose one convention and always use it:

Z-Y-X (α–β–γ) Euler angles (about moving axes) – demonstrate using frames $\{B\}$ and $\{A\}$:

Start with frames $\{B\}$ and $\{A\}$ aligned (what is $\left[{}^{A}_{B}R \right]$ for that special configuration?). Assume $\{B\}$ is the moving frame and $\{A\}$ is the reference frame.

a. First rotate moving frame $\{B\}$ by an angle α about the axis $\left\{ \hat{Z}_B \right\}$ (which is identical to $\left\{ \hat{Z}_A \right\}$ for this first rotation only).

b. Next rotate moving frame $\{B\}$ by an angle β about the axis $\left\{ \hat{Y}_B \right\}$ (which was moved away from $\left\{ \hat{Y}_A \right\}$ in the a. rotation).

c. Last rotate moving frame $\{B\}$ by an angle γ about the axis $\left\{ \hat{X}_B \right\}$ (which has been twice rotated away from $\left\{ \hat{X}_A \right\}$ in the a. and b. rotations).

The overall orientation description for this Euler rotation convention is derived as follows:

$$\left[{}^{A}_{B}R \right] = \left[R_Z(\alpha) \right] \left[R_Y(\beta) \right] \left[R_X(\gamma) \right]$$

$$\left[{}^{A}_{B}R \right] = \begin{bmatrix} c\alpha & -s\alpha & 0 \\ s\alpha & c\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\beta & 0 & s\beta \\ 0 & 1 & 0 \\ -s\beta & 0 & c\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\gamma & -s\gamma \\ 0 & s\gamma & c\gamma \end{bmatrix}$$

$$\left[{}^{A}_{B}R \right] = \begin{bmatrix} c\alpha c\beta & -s\alpha c\gamma + c\alpha s\beta s\gamma & s\alpha s\gamma + c\alpha s\beta c\gamma \\ s\alpha c\beta & c\alpha c\gamma + s\alpha s\beta s\gamma & -c\alpha s\gamma + s\alpha s\beta c\gamma \\ -s\beta & c\beta s\gamma & c\beta c\gamma \end{bmatrix}$$

Given α–β–γ, it is easy to find $\left[{}^{A}_{B}R \right]$. This is called the **Forward Solution** or the **Z-Y-X** (α–β–γ) Euler angles to Orthonormal Rotation Matrix calculation.

The Forward Solution for **Z-Y-X** (α–β–γ) Euler angles is mathematically identical to the **X-Y-Z** (γ–β–α) Fixed angles Forward Solution.

8. Inverse Euler angles solution. Given a valid $\left[{}^{A}_{B}R \right]$, find α–β–γ, a much harder problem.

The inverse problem is solved by forming three independent equations or combinations of equations from the symbolic expression of $\left[{}^{A}_{B}R \right]$:

$$\left[{}^{A}_{B}R \right] = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = \begin{bmatrix} c\alpha c\beta & -s\alpha c\gamma + c\alpha s\beta s\gamma & s\alpha s\gamma + c\alpha s\beta c\gamma \\ s\alpha c\beta & c\alpha c\gamma + s\alpha s\beta s\gamma & -c\alpha s\gamma + s\alpha s\beta c\gamma \\ -s\beta & c\beta s\gamma & c\beta c\gamma \end{bmatrix}$$

The r_{ij} components are 9 given, consistent (see constraint discussion in the next subsection) numbers of $\left[{}^{A}_{B}R \right]$. The Euler angles α–β–γ are the 3 unknowns.

Using the first column and the facts that $c^2\alpha + s^2\alpha = 1$ and $\tan\beta = \sin\beta / \cos\beta$:

$$\beta = \text{atan2}\left[-r_{31}, \pm\sqrt{r_{11}^2 + r_{21}^2} \right]$$

Using the 21 and 11 terms:

$$\alpha = \text{atan2}\left[\frac{r_{21}}{c\beta}, \frac{r_{11}}{c\beta} \right]$$

Using the 32 and 33 terms:

$$\gamma = \text{atan2}\left[\frac{r_{32}}{c\beta}, \frac{r_{33}}{c\beta} \right]$$

Where **`atan2`** is the quadrant-specific inverse tangent function in MATLAB. Note that dividing by $c\beta$ on the numerator and denominator of the last two solutions may seem silly since $c\beta$ cancels out, but the ***sign*** of $c\beta$ makes a difference. There are two overall inverse solution sets, making use of the $\pm$ in the β solution. We generally only need one solution set, using +. Both α–β–γ solution sets must yield the original $\left[{}^{A}_{B}R \right]$ when substituted into the forward rotation matrix expression (this is the circular check).

When $c\beta = 0$, the above solution fails (Does it fail? Try it in MATLAB). This is known as an algorithmic singularity because the math fails and the robot is not in a physical singularity (discussed later). Remember Euler angle convention is rather artificial – an algorithmic singularity is also called an artificial singularity. There is an alternate solution in the case of algorithmic singularity, where one can only calculate the sum or difference of α and γ. One method in this case is to set the value of α and calculate γ.

$$\beta = 90^\circ \qquad\qquad \beta = -90^\circ$$
$$\alpha = 0 \qquad\qquad \alpha = 0$$
$$\gamma = \text{atan2}(r_{12}, r_{13}) \qquad\qquad \gamma = \text{atan2}(-r_{12}, -r_{13})$$

9. Orthonormal Rotation Matrix Constraints

Remember spatial rotations have 3-dof (e.g. roll-pitch-yaw, or α–β–γ), but the Orthonormal Rotation Matrix has 9 numbers. Therefore, there must be 6 scalar constraints on these nine numbers. These six constraints come from the name ***Orthonormal***.

There are three ***Orthogonal*** constraints. All three columns (and rows) are mutually perpendicular to each other for all possible orientations. This makes sense since the **X-Y-Z** axes of frame $\{B\}$ are permanently mutually perpendicular for all motion. We can use one vector cross product or three vector dot products to express these three constraints.

There are three ***Normalized*** constraints. All three columns (and rows) are unit vectors. We express these three constraints with three scalar vector length equations.

10. Euler Angles Forward and Inverse Examples

This section presents examples for the forward and inverse problems for the **Z-Y-X** (α–β–γ) Euler angles convention.

a. **Forward calculation:** Given **Z-Y-X** Euler angles $\alpha = 50^\circ$, $\beta = 40^\circ$, and $\gamma = 30^\circ$, find

Answer:

$$\left[{}^{A}_{B}R \right] = \begin{bmatrix} 0.49 & -0.46 & 0.74 \\ 0.59 & 0.80 & 0.11 \\ -0.64 & 0.38 & 0.66 \end{bmatrix}$$

Be sure to check the six constraints (see 9. above) to ensure this is a valid Orthonormal Rotation Matrix result.

b. Inverse solution:

Given $\left[{}^{A}_{B}R \right] = \begin{bmatrix} 0.49 & -0.46 & 0.74 \\ 0.59 & 0.80 & 0.11 \\ -0.64 & 0.38 & 0.66 \end{bmatrix}$, solve for the **Z-Y-X** Euler angles α–β–γ (both solution sets).

Answer:

Solution	α	β	γ
1	50°	40°	30°
2	230°	140°	210°

The first solution is what we started with above. Be sure to check the second solution set by plugging angles α–β–γ into the forward solution above to ensure the same $\left[{}^{A}_{B}R \right]$ results. This is called a **circular check**.

The solution sets pattern is always as shown in the table below.

1	α	β	γ
2	$\alpha + 180^\circ$	$180^\circ - \beta$	$\gamma + 180^\circ$

Remember we only need one solution set due to the artificiality; the first set can be used in all cases.

11. Position vector rotational transformations

We now introduce a very useful matrix-vector equation for transforming the coordinates of a given position vector $\left\{ {}^{B}P \right\}$ known in the $\{B\}$ frame to the same vector $\left\{ {}^{A}P \right\}$ described in the $\{A\}$ frame coordinates (changing the basis of the vector from $\{B\}$ to $\{A\}$, that is, changing the coordinate frame into which the vector is projected to obtain its components). The vector length is unchanged in this case.

The **XYZ** components of a vector are the projections of that vector onto the **XYZ** axes of the frame of interest. Let's project a vector P onto the $\{A\}$ frame using the dot product.

$$
\begin{aligned}
{}^{A}p_X &= \left\{ \hat{X}_A \right\} \bullet \{P\} \\
{}^{A}p_Y &= \left\{ \hat{Y}_A \right\} \bullet \{P\} \\
{}^{A}p_Z &= \left\{ \hat{Z}_A \right\} \bullet \{P\}
\end{aligned}
$$

Now, let's change the basis (the Cartesian coordinate frame in which the coordinates of a vector are expressed) of vector P from $\{B\}$ to $\{A\}$. The right-hand side vectors of the above equations can be expressed in $\{B\}$ coordinates as follows.

$$
\begin{aligned}
{}^{A}p_X &= \left\{ {}^{B}\hat{X}_A \right\} \bullet \left\{ {}^{B}P \right\} \\
{}^{A}p_Y &= \left\{ {}^{B}\hat{Y}_A \right\} \bullet \left\{ {}^{B}P \right\} \\
{}^{A}p_Z &= \left\{ {}^{B}\hat{Z}_A \right\} \bullet \left\{ {}^{B}P \right\}
\end{aligned}
$$

Now, $\left\{ {}^{B}\hat{X}_A \right\}, \left\{ {}^{B}\hat{Y}_A \right\}, \left\{ {}^{B}\hat{Z}_A \right\}$ are the rows of $\left[{}^{A}_{B}R \right]$, so thinking of the relationship between the dot product and matrix multiplication, we have

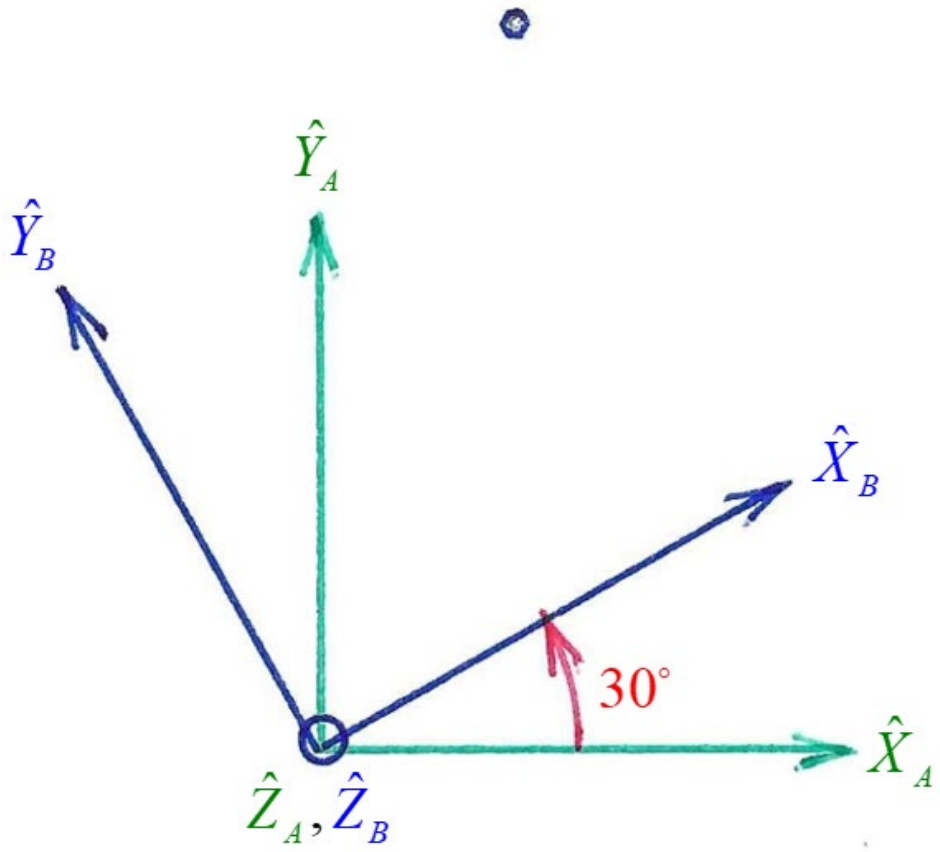
$$
\left\{ {}^{A}P \right\} = \left[{}^{A}_{B}R \right] \left\{ {}^{B}P \right\}
$$

Note that since the two vectors are identical in direction and length (since we are not yet considering position vector differences between the two coordinate frame origins), only the basis for expression of the **XYZ** components are different. So, we can use the Orthonormal Rotation Matrix to rotate the coordinates of a vector from one Cartesian coordinate frame to another.

Example for Step 11

Given $\left[{}^{A}_{B}R \right] = \left[R_Z(\alpha = 30^\circ) \right]$ and $\left\{ {}^{B}P \right\} = \begin{Bmatrix} 1 \\ 1 \\ 0 \end{Bmatrix}$

Find $\left\{ {}^{A}P \right\}$ using $\left\{ {}^{A}P \right\} = \left[{}^{A}_{B}R \right] \left\{ {}^{B}P \right\}$



12. Fixed and Euler Angles Convention Examples

We conclude this section on Orthonormal Rotation Matrices by presenting examples for fixed vs. Euler angle conventions with **Z-Y-X** (α–β–γ) notation.

a. **Euler Angles.** Given **Z-Y-X** Euler angles $\alpha = 50^\circ, \beta = 40^\circ, \gamma = 30^\circ$, find $\left[{}^{A}_{B}R \right]$. This is the same example from the previous section.

Answer:

$$\left[{}^{A}_{B}R \right] = \begin{bmatrix} 0.49 & -0.46 & 0.74 \\ 0.59 & 0.80 & 0.11 \\ -0.64 & 0.38 & 0.66 \end{bmatrix}$$

b. **Fixed Angles.** Given **Z-Y-X** Fixed-axis angles $\alpha = 50^\circ, \beta = 40^\circ, \gamma = 30^\circ$, find $\left[{}^{A}_{B}R \right]$. Relative to Euler angles, we must reverse the formula:

$$\left[{}^{A}_{B}R \right] = \left[R_X(\gamma) \right] \left[R_Y(\beta) \right] \left[R_Z(\alpha) \right]$$

Answer:

(quite different from Euler convention, see the figures below)

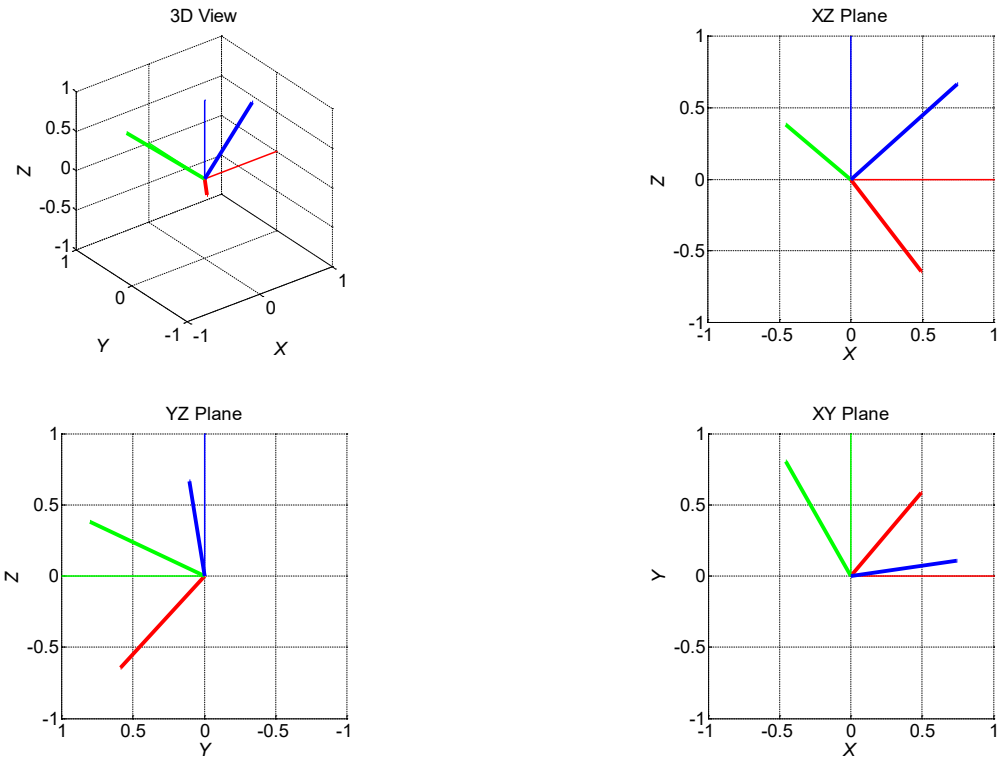
$$\left[{}^{A}_{B}R \right] = \begin{bmatrix} 0.49 & -0.59 & 0.64 \\ 0.87 & 0.31 & -0.38 \\ 0.03 & 0.75 & 0.66 \end{bmatrix}$$

c. **Euler Angles.** Now calculate the **Z-Y-X** Euler angles (α–β–γ) notation to yield the identical orthonormal rotation matrix as the fixed-axis convention (use the fixed-axis $\left[{}^{A}_{B}R \right]$ from b. above and the inverse Euler solution).

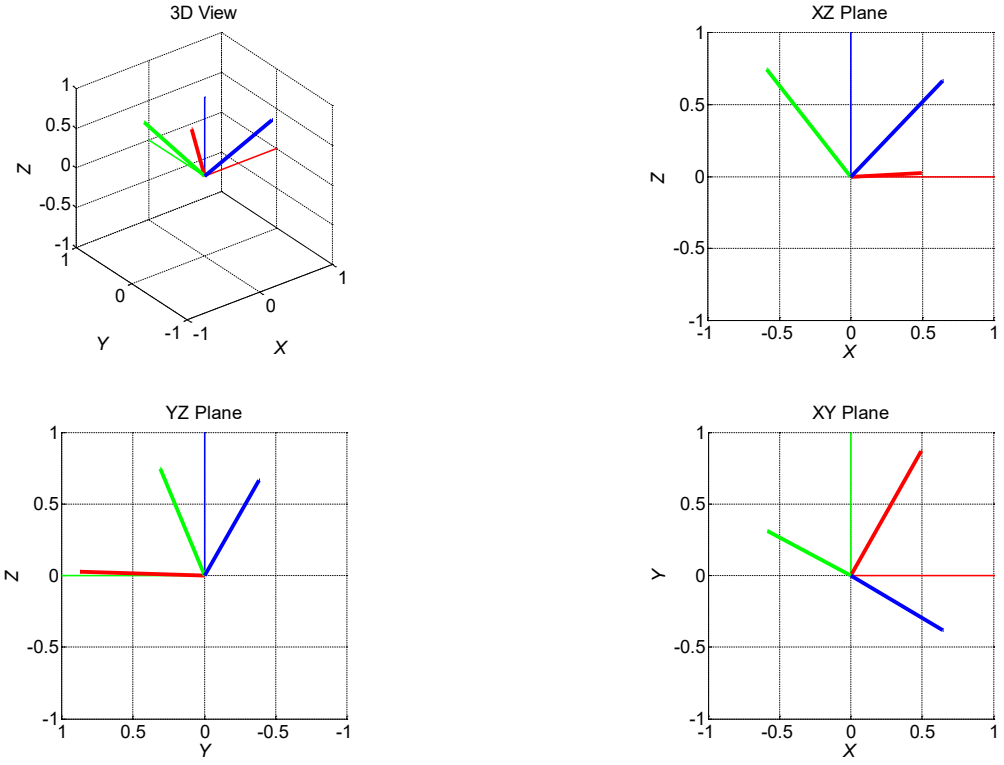
Answer $\alpha = 60.5^\circ$, $\beta = -1.4^\circ$, and $\gamma = 48.4^\circ$ (this is the first solution set only)

Check plug this set of Euler angles into the forward Euler solution to ensure you get the correct rotation matrix back (circular check). Also check the graphics – both solutions check out.

Graphical results for these examples are presented on the next page.



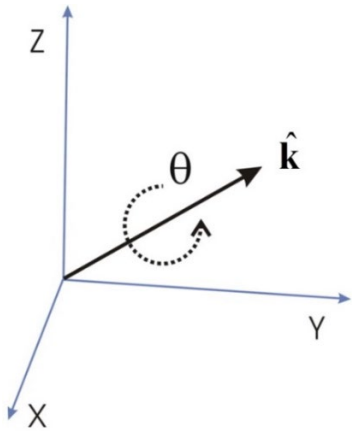
Z-Y-X Euler angles example $\alpha = 50^\circ$, $\beta = 40^\circ$, and $\gamma = 30^\circ$ ***X* axes, *Y* axes, *Z* axes**



Z-Y-X Fixed-axis angles example $\alpha = 50^\circ$, $\beta = 40^\circ$, and $\gamma = 30^\circ$ ***X* axes, *Y* axes, *Z* axes**

2.1.2 Quaternions

An alternative method to describe 3D orientations is the so-called **axis-angle method**. As in Euler angle description, start with the pose of moving frame $\{B\}$ identical to the pose of reference frame $\{A\}$ (that is, keep their origins coincident and initially the **XYZ** axes are aligned). Then perform a single rotation of angle θ, about a unit vector ${}^{A}\{\hat{\mathbf{k}}\} = \{k_x \quad k_y \quad k_z\}^T$. Any general roll-pitch-yaw 3D spatial rotation can be described in this manner (according to Euler's 3D rotations theorem). This represents 3-dof, the number of independent spatial rotations. The 3-dof are represented by θ, k_x, and k_y (since with a unit vector we have the constraint $k_z = \sqrt{1-k_x^2-k_y^2}$).



Now, if we no longer require the two coordinate frame origins to be identical for all motion, then we have screw motion. That is, a rotation of angle θ about a unit vector ${}^{A}\{\hat{\mathbf{k}}\}$ and a translational displacement along ${}^{A}\{\hat{\mathbf{k}}\}$, called the pitch of the screw (not to be confused with the rotational pitch motion). This is the basis of Screw Theory (Ball, 1800), wherein general 3D translational and rotational motions are represented by a rotation around the screw axis and displacement along it. This is beyond the scope of EE/ME 4290/5290.

Quaternions are related to the **axis-angle method**, but 4 parameters are used instead of 3, so there is one redundancy. This section briefly introduces Quaternions, also called Euler Parameters, as an alternative method to represent 3D orientation of frame $\{B\}$ relative to frame $\{A\}$. Like Euler angles (and rotations about Fixed axes), Quaternions lead to the orthonormal rotation matrix $\left[{}^{A}_{B}R\right]$. Benefits of Quaternions include:

- Quaternions have unique representation for $-180^\circ < \theta \le 180^\circ$. This is a major benefit since there are 12 ways to describe Euler angles and 12 ways to describe rotations about Fixed axes.
- Quaternions are not subject to the artificial singularities that affect Euler angles and rotations about Fixed axes in the inverse problem. This is another major benefit.

Axis-Angle $\leftrightarrow$ **Quaternions** $\leftrightarrow$ $\left[{}^{A}_{B}R\right]$

For the case where reference frame $\{A\}$ is fixed and moving frame $\{B\}$ is rotated about unit vector ${}^{A}\left\{\hat{\mathbf{k}}\right\} = \left\{k_x \quad k_y \quad k_z\right\}^T$ by angle θ, the Euler Parameters (Quaternion $\{Q\}$) are defined as follows.

$$\{Q\} = \begin{Bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \end{Bmatrix} = \begin{Bmatrix} k_x \sin\left(\frac{\theta}{2}\right) \\ k_y \sin\left(\frac{\theta}{2}\right) \\ k_z \sin\left(\frac{\theta}{2}\right) \\ \cos\left(\frac{\theta}{2}\right) \end{Bmatrix} = \begin{Bmatrix} \begin{Bmatrix} k_x \\ k_y \\ k_z \end{Bmatrix} \sin\left(\frac{\theta}{2}\right) \\ \cos\left(\frac{\theta}{2}\right) \end{Bmatrix}$$

The Quaternion $\{Q\}$ may be interpreted as a vector from the first three terms, plus a scalar, the fourth term. Since spatial orientations have 3-dof, Quaternion representation has one constraint (all four are not independent); the following constraint results from the four Quaternion terms.

$$\varepsilon_1^2 + \varepsilon_2^2 + \varepsilon_3^2 + \varepsilon_4^2 = 1$$

The **forward Axis-Angle/Quaternion solution** is stated: Given numbers for the unit vector axis ${}^{A}\left\{\hat{\mathbf{k}}\right\} = \left\{k_x \quad k_y \quad k_z\right\}^T$ and angle θ, calculate the 4x1 Quaternion $\{Q\}$ (four Euler Parameters ε_i, $i = 1,2,3,4$). This solution is presented above, in the definition for $\{Q\}$.

The **inverse Quaternion/Axis-Angle solution** is stated: Given valid numbers for the 4x1 Quaternion $\{Q\}$, calculate the unit vector axis ${}^{A}\left\{\hat{\mathbf{k}}\right\} = \left\{k_x \quad k_y \quad k_z\right\}^T$ and angle θ. This solution is presented below. Trigonometric uncertainty (i.e. the double-valued result for inverse cosine) is not an issue since we can just take the primary solution for θ (in quadrant I or IV) and the associated rotation axis ${}^{A}\left\{\hat{\mathbf{k}}\right\}$ will be appropriate for that angle.

$$\theta = 2\cos^{-1}(\varepsilon_4)$$

$$\begin{Bmatrix} k_x \\ k_y \\ k_z \end{Bmatrix} = \frac{1}{\sin\left(\frac{\theta}{2}\right)} \begin{Bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \end{Bmatrix}$$

The **forward Quaternion/Rotation Matrix solution** is stated: Given numbers for the Quaternion (four Euler Parameters), calculate the 3x3 orthonormal rotation matrix $\left[{}^{A}_{B}R\right]$. Given the four Euler parameters (i.e. given the Quaternion), the 3x3 orthonormal rotation matrix is calculated as follows.

$$\left[{}^{A}_{B}R\right] = \begin{bmatrix} 1-2(\varepsilon_2^2+\varepsilon_3^2) & 2(\varepsilon_1\varepsilon_2-\varepsilon_3\varepsilon_4) & 2(\varepsilon_1\varepsilon_3+\varepsilon_2\varepsilon_4) \\ 2(\varepsilon_1\varepsilon_2+\varepsilon_3\varepsilon_4) & 1-2(\varepsilon_1^2+\varepsilon_3^2) & 2(\varepsilon_2\varepsilon_3-\varepsilon_1\varepsilon_4) \\ 2(\varepsilon_1\varepsilon_3-\varepsilon_2\varepsilon_4) & 2(\varepsilon_2\varepsilon_3+\varepsilon_1\varepsilon_4) & 1-2(\varepsilon_1^2+\varepsilon_2^2) \end{bmatrix}$$

An equivalent formula for this same 3x3 orthonormal rotation matrix $\left[{}^{A}_{B}R \right]$ is:

$$\left[{}^{A}_{B}R \right] = (\varepsilon_4^2 - \{\hat{\varepsilon}\}^T \{\hat{\varepsilon}\})[I_3] + 2\{\hat{\varepsilon}\}\{\hat{\varepsilon}\}^T + 2\varepsilon_4 \left[\hat{\varepsilon}^{\times} \right]$$

where $\{\hat{\varepsilon}\} = \begin{Bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \end{Bmatrix}$ and $\left[\hat{\varepsilon}^{\times} \right] = \begin{bmatrix} 0 & -\varepsilon_3 & \varepsilon_2 \\ \varepsilon_3 & 0 & -\varepsilon_1 \\ -\varepsilon_2 & \varepsilon_1 & 0 \end{bmatrix}$

are the Euler parameter vector and skew-symmetric cross-product matrix, respectively, ε_4 is the Euler parameter scalar, and $[I_3]$ is the 3x3 identity matrix. This form for $\left[{}^{A}_{B}R \right]$ is convenient, though it looks complicated, since it relies on matrix-vector computations, to compare numerical results to the previous form for $\left[{}^{A}_{B}R \right]$.

The **inverse Rotation Matrix/Quaternion solution** is stated: Given valid numbers for the 3x3 orthonormal rotation matrix $\left[{}^{A}_{B}R \right]$, calculate the Quaternion.

$$\text{Given } \left[{}^{A}_{B}R \right] = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix},$$

this **inverse Quaternion solution** is presented below.

$$\varepsilon_4 = \frac{\sqrt{1 + r_{11} + r_{22} + r_{33}}}{2}$$

$$\varepsilon_1 = \frac{r_{32} - r_{23}}{4\varepsilon_4}$$

$$\varepsilon_2 = \frac{r_{13} - r_{31}}{4\varepsilon_4}$$

$$\varepsilon_3 = \frac{r_{21} - r_{12}}{4\varepsilon_4}$$

Many references claim this is a singularity-free solution (no artificial singularities like the inverse Euler angles solution has). However, for a rotation of 180° about any of the major axes **XYZ**, ε_4 goes to zero. There is an artificial singularity in this case. However, in the limit as ε_4 approaches zero, all four terms in the inverse solution above remain finite, which is a major improvement over the type of artificial singularity found in the inverse Euler angle and inverse rotations about Fixed axes solutions.

There are two solution sets if we allow $\pm$ in the square-root for ε_4, but the second solution set does not yield anything useful the first solution cannot, so let us just use the + square-root as shown above, so there is a unique solution. (The same could be said of the inverse Euler angle solution, so this is not a benefit of Quaternions.)

Quaternion and Axis-Angle Examples

Let us use the same example from the Euler angle section from Orthonormal Rotation matrices.

Given **Z-Y-X** Euler angles $\alpha = 50^\circ, \beta = 40^\circ, \gamma = 30^\circ$, we found $\left[{}^{A}_{B}R \right] = \begin{bmatrix} 0.49 & -0.46 & 0.74 \\ 0.59 & 0.80 & 0.11 \\ -0.64 & 0.38 & 0.66 \end{bmatrix}$.

Given this $\left[{}^{A}_{B}R \right]$, the **inverse Rotation Matrix/Quaternion solution** yields:

$$\{Q\} = \begin{Bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \end{Bmatrix} = \begin{Bmatrix} 0.08 \\ 0.40 \\ 0.30 \\ 0.86 \end{Bmatrix}$$

Then, using this Quaternion as the input to the **forward Quaternion/Rotation Matrix solution** (either alternative solution) yields:

$$\left[{}^{A}_{B}R \right] = \begin{bmatrix} 0.49 & -0.46 & 0.74 \\ 0.59 & 0.80 & 0.11 \\ -0.64 & 0.38 & 0.66 \end{bmatrix}$$

which matches, as expected and required (circular check). This Quaternion $\{Q\}$ is unique, a big benefit compared to the 12 artificial possibilities for Euler angles.

Given this unique Quaternion $\{Q\}$, the **inverse Quaternion/Axis-Angle solution** yields:

$$\theta = 61.4^\circ \qquad \text{about} \qquad {}^{A}\left\{ \hat{\mathbf{k}} \right\} = \begin{Bmatrix} k_x \\ k_y \\ k_z \end{Bmatrix} = \begin{Bmatrix} 0.16 \\ 0.79 \\ 0.59 \end{Bmatrix}$$

Then, using this axis-angle as the input to the **forward Axis-Angle/Quaternion solution** yields:

$$\{Q\} = \begin{Bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \end{Bmatrix} = \begin{Bmatrix} 0.08 \\ 0.40 \\ 0.30 \\ 0.86 \end{Bmatrix}$$

which matches, as expected and required (circular check).

To summarize, Quaternions are an appealing alternative to Euler angles for representation of 3D orientations. Benefits of Quaternions include unique representation and no artificial singularities over $-180^\circ < \theta < 180^\circ$. However, one basic fact remains unchanged: there is no vector representation for 3D orientations. Therefore, **Quaternions are not a vector representation for 3D orientations**.

Visualization of Axis-Angle convention

Euler's Theorem on 3D rotations teaches that any general 3D rotation can be represented by a single rotation angle θ about the appropriate rotational axis $\hat{\mathbf{k}}$. As we have seen, this is the basis for the definition of Quaternions, an appealing alternative to Euler angles or rotations about fixed axes.

Perhaps the Euler Angles and rotations about fixed axes are easier to visualize than the axis-angle convention – therefore it will be instructive to develop a MATLAB program to animate the axis-angle convention and compare it to known Euler Angles examples.

A convenient method to rotate a vector about a given axis $\hat{\mathbf{k}}$ by an angle θ was developed by Olinde Rodrigues (1795-1851, Jew of Portuguese descent living in France). Using his algorithm, not celebrated in his lifetime, but rediscovered in the 20th century, there is no need to use the rotation matrix convention to perform this axis-angle rotation.

Rodrigues Formula, given below, is based on decomposing the vector to rotate **P** into components parallel and perpendicular to the rotation axis $\hat{\mathbf{k}}$ and only rotating the perpendicular component by angle θ.

Rodrigues Formula:

$$\mathbf{P}_{\mathbf{rot}} = \mathbf{P}\cos\theta + (\hat{\mathbf{k}} \times \mathbf{P})\sin\theta + \hat{\mathbf{k}}(\hat{\mathbf{k}} \cdot \mathbf{P})(1 - \cos\theta)$$

Rodrigues Formula, Matrix Form:

$$\mathbf{P}_{\mathbf{rot}} = \mathbf{R}(\hat{\mathbf{k}}, \theta)\mathbf{P}$$

where:

$$\mathbf{R}(\hat{\mathbf{k}}, \theta) = [\mathbf{I}_3] + [\hat{\mathbf{k}}^{\times}]\sin\theta + [\hat{\mathbf{k}}^{\times}]^2 (1 - \cos\theta)$$

and:

$$[\hat{\mathbf{k}}^{\times}] = \begin{bmatrix} 0 & -k_z & k_y \\ k_z & 0 & -k_x \\ -k_y & k_x & 0 \end{bmatrix}$$

Like the other rotation conventions (Euler Angles, rotations about fixed axes), we start with reference frame $\{A\}$ and moving frame $\{B\}$ coincident, with the same origins and **XYZ** unit vector pointing directions. Then we apply Rodrigues' Formula three times, once each for the standard basis of $\{B\}$,

$$\mathbf{P} = \hat{\mathbf{X}}_B = \begin{Bmatrix} 1 \\ 0 \\ 0 \end{Bmatrix} \qquad \mathbf{P} = \hat{\mathbf{Y}}_B = \begin{Bmatrix} 0 \\ 1 \\ 0 \end{Bmatrix} \qquad \mathbf{P} = \hat{\mathbf{Z}}_B = \begin{Bmatrix} 0 \\ 0 \\ 1 \end{Bmatrix}$$

From before we have:

$$^A\left\{\hat{\mathbf{k}}\right\} = \begin{Bmatrix} k_x \\ k_y \\ k_z \end{Bmatrix} \qquad \text{and} \qquad k_z = \sqrt{1 - k_x^2 - k_y^2}$$

Applying this in MATLAB, animating given $\hat{\mathbf{k}}$ and θ, will generate the visualization we desire. Then in the final orientation of $\{B\}$ with respect to $\{A\}$, we can compare the resulting graphics to known Euler Angle examples.

Rodrigues Formula; Analytical Expressions for the Standard Basis Vectors:

$$\hat{\mathbf{X}}_{B_{\text{rot}}} = \begin{Bmatrix} k_x^2(1-\cos\theta) + \cos\theta \\ k_x k_y(1-\cos\theta) + k_z \sin\theta \\ k_z k_x(1-\cos\theta) - k_y \sin\theta \end{Bmatrix}$$

$$\hat{\mathbf{Y}}_{B_{\text{rot}}} = \begin{Bmatrix} k_x k_y(1-\cos\theta) - k_z \sin\theta \\ k_y^2(1-\cos\theta) + \cos\theta \\ k_y k_z(1-\cos\theta) + k_x \sin\theta \end{Bmatrix}$$

$$\hat{\mathbf{Z}}_{B_{\text{rot}}} = \begin{Bmatrix} k_z k_x(1-\cos\theta) + k_y \sin\theta \\ k_y k_z(1-\cos\theta) - k_x \sin\theta \\ k_z^2(1-\cos\theta) + \cos\theta \end{Bmatrix}$$

Rodrigues Formula; Analytical Expressions for the Standard Basis Vectors using matrix formulation (the results must agree):

$$\mathbf{P}_{\text{rot}} = \mathbf{R}(\hat{\mathbf{k}}, \theta)\mathbf{P}$$

where:

$$\left[\hat{\mathbf{k}}^{\times}\right] = \begin{bmatrix} 0 & -k_z & k_y \\ k_z & 0 & -k_x \\ -k_y & k_x & 0 \end{bmatrix} \qquad \left[\hat{\mathbf{k}}^{\times}\right]^2 = \begin{bmatrix} -k_y^2 - k_z^2 & k_x k_y & k_z k_x \\ k_x k_y & -k_z^2 - k_x^2 & k_y k_z \\ k_z k_x & k_y k_z & -k_x^2 - k_y^2 \end{bmatrix} = \begin{bmatrix} k_x^2 - 1 & k_x k_y & k_z k_x \\ k_x k_y & k_y^2 - 1 & k_y k_z \\ k_z k_x & k_y k_z & k_z^2 - 1 \end{bmatrix}$$

$$\mathbf{R}(\hat{\mathbf{k}},\theta)=\begin{bmatrix} 1+(k_x^2-1)(1-\cos\theta) & k_xk_y(1-\cos\theta)-k_z\sin\theta & k_zk_x(1-\cos\theta)+k_y\sin\theta \\ k_xk_y(1-\cos\theta)+k_z\sin\theta & 1+(k_y^2-1)(1-\cos\theta) & k_yk_z(1-\cos\theta)-k_x\sin\theta \\ k_zk_x(1-\cos\theta)-k_y\sin\theta & k_yk_z(1-\cos\theta)+k_x\sin\theta & 1+(k_z^2-1)(1-\cos\theta) \end{bmatrix}$$

$$\mathbf{R}(\hat{\mathbf{k}},\theta)=\begin{bmatrix} k_x^2(1-\cos\theta)+\cos\theta & k_xk_y(1-\cos\theta)-k_z\sin\theta & k_zk_x(1-\cos\theta)+k_y\sin\theta \\ k_xk_y(1-\cos\theta)+k_z\sin\theta & k_y^2(1-\cos\theta)+\cos\theta & k_yk_z(1-\cos\theta)-k_x\sin\theta \\ k_zk_x(1-\cos\theta)-k_y\sin\theta & k_yk_z(1-\cos\theta)+k_x\sin\theta & k_z^2(1-\cos\theta)+\cos\theta \end{bmatrix}=\left[{}^{A}_{B}R\right]$$

Matrix

$$\hat{\mathbf{X}}_{B_{\text{rot}}}=\begin{Bmatrix} k_x^2(1-\cos\theta)+\cos\theta \\ k_xk_y(1-\cos\theta)+k_z\sin\theta \\ k_zk_x(1-\cos\theta)-k_y\sin\theta \end{Bmatrix}$$

$$\hat{\mathbf{Y}}_{B_{\text{rot}}}=\begin{Bmatrix} k_xk_y(1-\cos\theta)-k_z\sin\theta \\ k_y^2(1-\cos\theta)+\cos\theta \\ k_yk_z(1-\cos\theta)+k_x\sin\theta \end{Bmatrix}$$

$$\hat{\mathbf{Z}}_{B_{\text{rot}}}=\begin{Bmatrix} k_zk_x(1-\cos\theta)+k_y\sin\theta \\ k_yk_z(1-\cos\theta)-k_x\sin\theta \\ k_z^2(1-\cos\theta)+\cos\theta \end{Bmatrix}$$

Original Approach

$$\hat{\mathbf{X}}_{B_{\text{rot}}}=\begin{Bmatrix} k_x^2(1-\cos\theta)+\cos\theta \\ k_xk_y(1-\cos\theta)+k_z\sin\theta \\ k_zk_x(1-\cos\theta)-k_y\sin\theta \end{Bmatrix}$$

$$\hat{\mathbf{Y}}_{B_{\text{rot}}}=\begin{Bmatrix} k_xk_y(1-\cos\theta)-k_z\sin\theta \\ k_y^2(1-\cos\theta)+\cos\theta \\ k_yk_z(1-\cos\theta)+k_x\sin\theta \end{Bmatrix}$$

$$\hat{\mathbf{Z}}_{B_{\text{rot}}}=\begin{Bmatrix} k_zk_x(1-\cos\theta)+k_y\sin\theta \\ k_yk_z(1-\cos\theta)-k_x\sin\theta \\ k_z^2(1-\cos\theta)+\cos\theta \end{Bmatrix}$$

The matrix equation approach yields identical results to the original approach, as required and expected.

Axis-Angle 3D Rotation Convention Visualization Example:

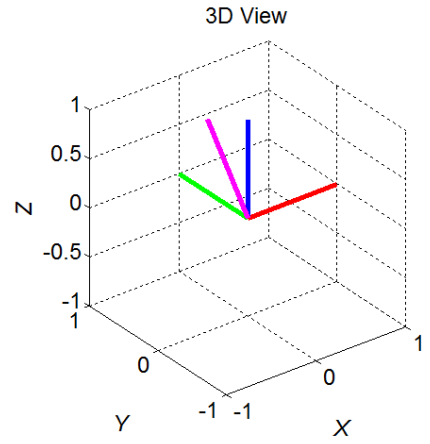
Let us use the same example from the Quaternion numerical example above.

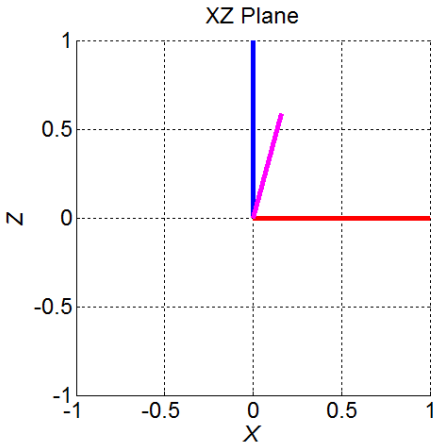
Given the Axis-Angle inputs:

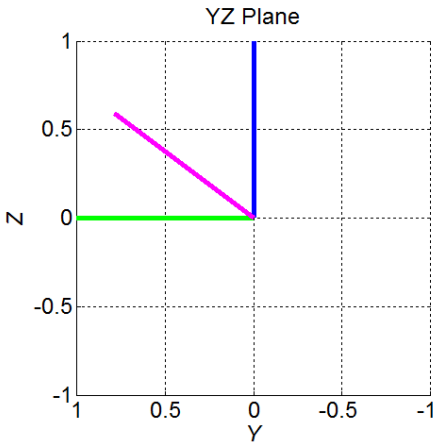
$$\theta = 61.4^\circ \qquad \text{about} \qquad {}^{A}\{\hat{\mathbf{k}}\} = \begin{Bmatrix} k_x \\ k_y \\ k_z \end{Bmatrix} = \begin{Bmatrix} 0.16 \\ 0.79 \\ 0.59 \end{Bmatrix}$$

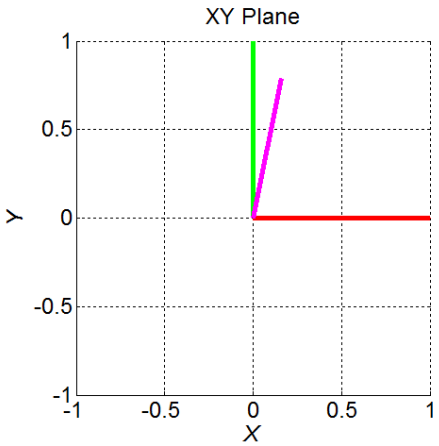
Animate the rotation of $\{B\}$ with respect to $\{A\}$. Start with reference frame $\{A\}$ and moving frame $\{B\}$ coincident, with the same origins and **XYZ** unit vector pointing directions. Then apply Rodrigues' Formula three times, once each for the standard basis of $\{B\}$, for 10 equal steps in the range $0 \le \theta \le 61.4^\circ$.

In the figures below, the ${}^{A}\{\hat{\mathbf{k}}\}$ axis is represented by the magenta line, and red-green-blue stand for the **X-Y-Z** axes, respectively. The reference frame $\{A\}$ has thin lines and the moving frame $\{B\}$ thick lines.

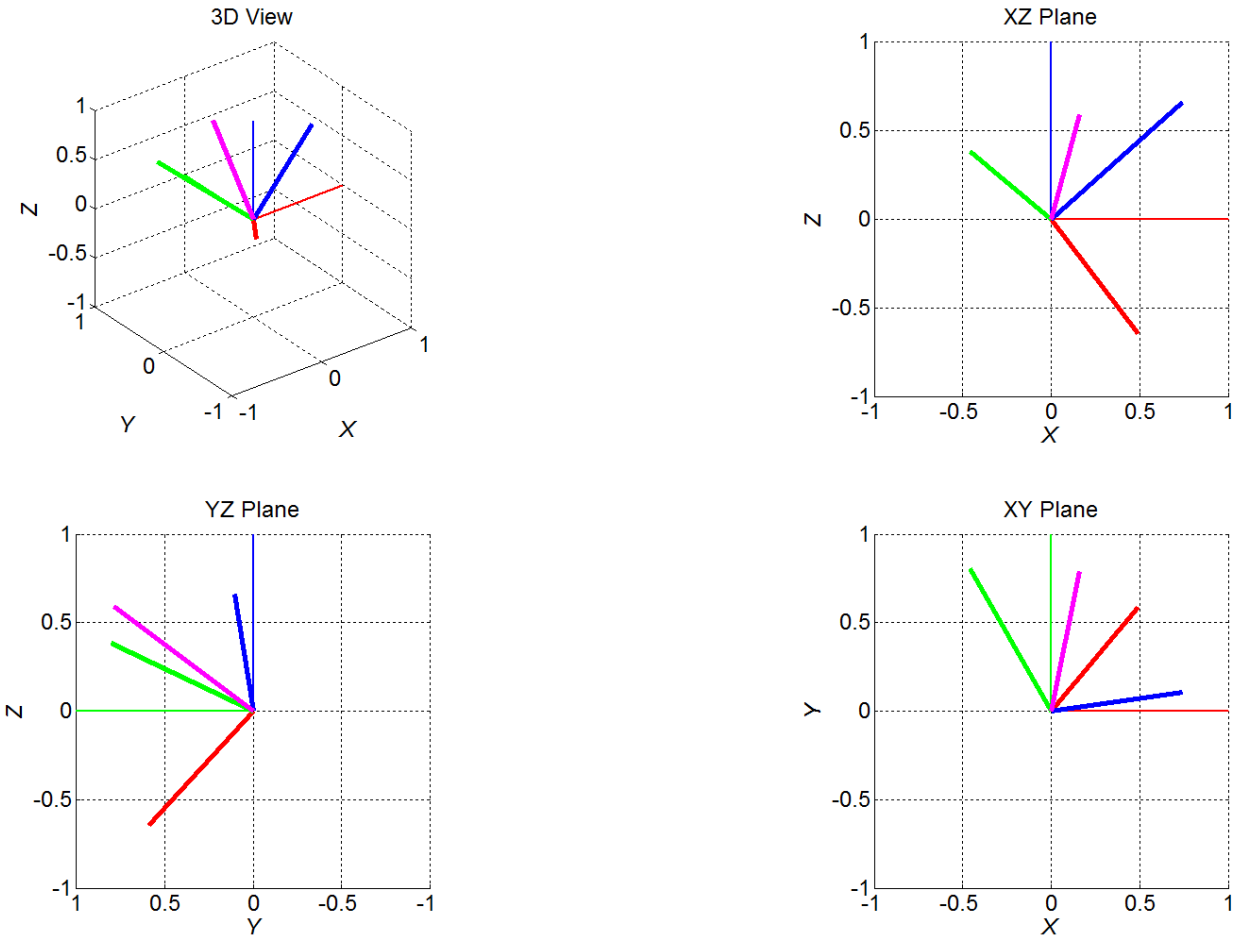








Initial Configuration



Configuration After Completed Rotational Motion

Associated with the final orientation in this Axis-Angle convention example are the following Quaternion, Orthonormal Rotation Matrix, and Euler Angles. The above axis-angle final configuration agrees perfectly with the **Z-Y-X** α, β, γ Euler Angle animation graphic (not shown) for the same orientation.

$$\{Q\} = \begin{Bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \end{Bmatrix} = \begin{Bmatrix} 0.08 \\ 0.40 \\ 0.30 \\ 0.86 \end{Bmatrix}$$

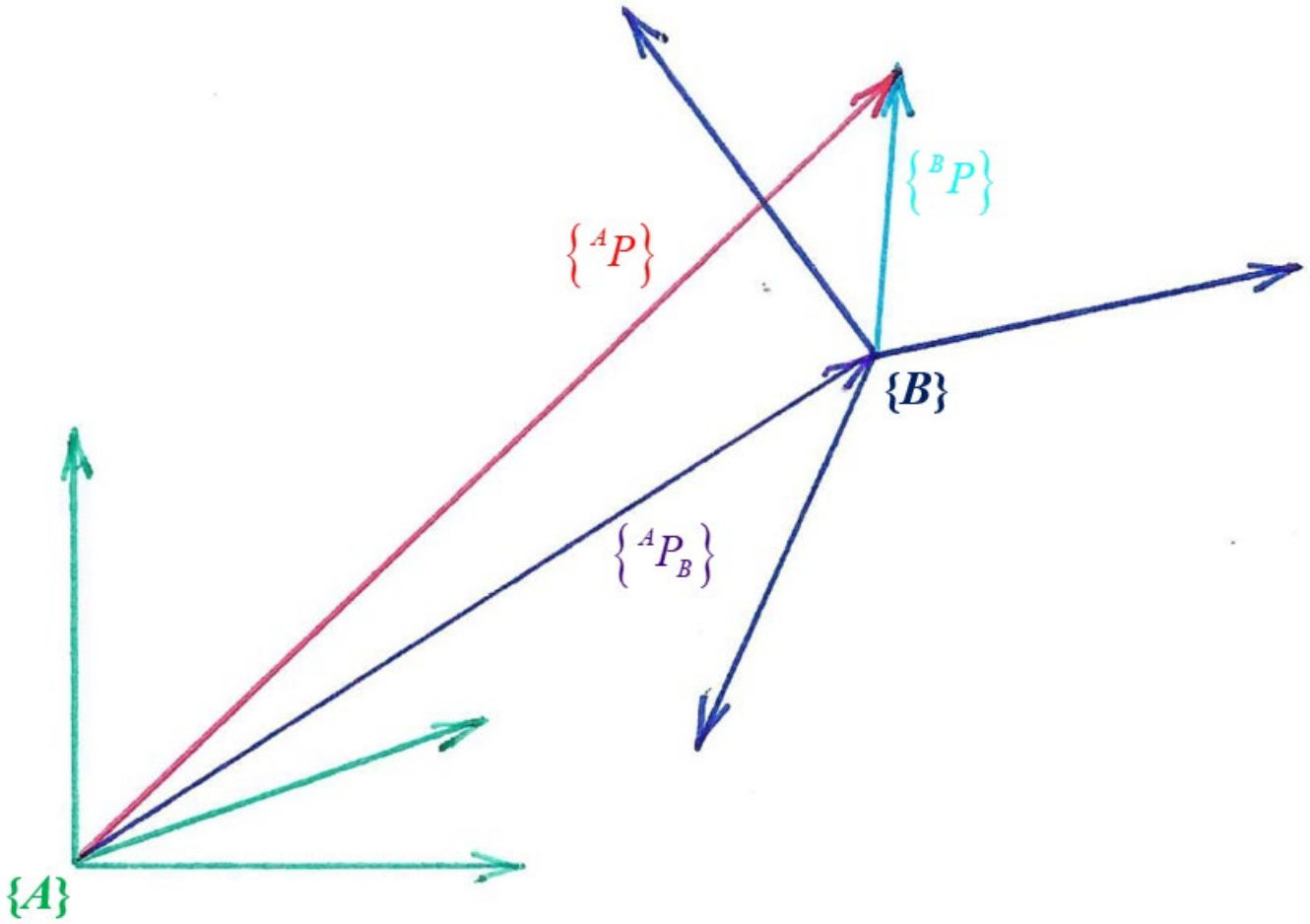
$$\left[{}^{A}_{B}R \right] = \begin{bmatrix} 0.49 & -0.46 & 0.74 \\ 0.59 & 0.80 & 0.11 \\ -0.64 & 0.38 & 0.66 \end{bmatrix}$$

$$\alpha = 50^\circ, \beta = 40^\circ, \gamma = 30^\circ$$

2.2 3D Transformations

2.2.1 Homogeneous Transformation Matrices

The Homogeneous Transformation Matrix is one convenient 4x4 matrix representation to give pose (position and orientation) of one moving Cartesian coordinate frame with respect to a reference frame. The figure to derive the homogeneous transformation matrix is given below.



The vector loop closure equation is:

But all vectors must be expressed in a common frame for addition, {A} in this case:

This is a change in basis (i.e. transformation of coordinate representation from $\{B\}$ to $\{A\}$ – remember $\left\{ {}^{A}P \right\} = \left[{}^{A}_{B}R \right] \left\{ {}^{B}P \right\}$ from earlier), plus a translation since now $\left\{ {}^{A}P_{B} \right\}$ is considered. The general form of a (4 x 4) Homogenous Transformation Matrix is:

There is perspective and scaling when the last row is not $\begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}$ – this is used in computer graphics. For robotics purposes, this last row ***never*** changes from $\begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}$ – we don't want to scale or skew rigid links moved by active joints.

We must append a 1 to all 3x1 position vectors to use with Homogeneous Transformation Matrices. This leads to a dummy equation of 1 = 1, but this is necessary for matrix multiplication of Homogeneous Transformation Matrices with position vectors.

3 Interpretations for Homogeneous Transformation Matrices

Let us use a common $\left[{}^{A}_{B}T \right]$ numerical example for all 3 interpretations.

$$\left[{}^{A}_{B}T \right] = \begin{bmatrix} 0.866 & -0.5 & 0 & 2 \\ 0.5 & 0.866 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

1) Description of a Frame

Homogeneous Transformation Matrix $\left[{}^{A}_{B}T\right]$ describes the pose (position and orientation) of a moving Cartesian coordinate frame $\{B\}$ with respect to a reference frame $\{A\}$. The Cartesian coordinate frame $\{A\}$ can be moving too, but we place an observer on $\{A\}$ to watch how $\{B\}$ is translating and rotating with respect to $\{A\}$.

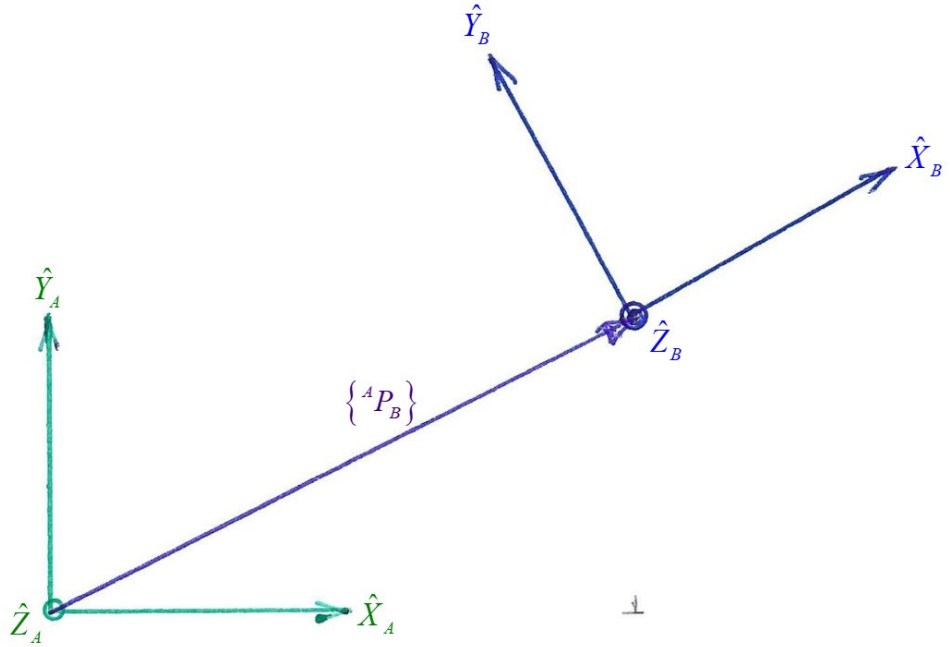
The problem statement is: Given $\left[{}^{A}_{B}T\right]$, describe the pose of moving frame $\{B\}$ with respect to reference frame $\{A\}$ (hint: the translational position is described by $\left\{{}^{A}P_{B}\right\}$ and the rotational orientation is described by orthonormal rotation matrix $\left[{}^{A}_{B}R\right]$).

$\left\{{}^{A}P_{B}\right\}$, the last column (top three terms) of $\left[{}^{A}_{B}T\right]$, is the position vector giving the location of the origin of $\{B\}$ with respect to the origin of $\{A\}$, expressed in the coordinates of $\{A\}$ (that is, $\{A\}$ is the basis, or coordinate frame of expression for all terms).

$\left[{}^{A}_{B}R\right]$, the upper-left 3x3 submatrix of $\left[{}^{A}_{B}T\right]$, is the rotation matrix giving the orientation of $\{B\}$ with respect to $\{A\}$. As we learned previously, the columns of $\left[{}^{A}_{B}R\right]$ are the **XYZ** unit vectors of $\{B\}$ projected onto the $\{A\}$ **XYZ** unit vector directions.

$$\left[{}^{A}_{B}R\right]=\begin{bmatrix} | & | & | \\ \left\{{}^{A}\hat{X}_{B}\right\} & \left\{{}^{A}\hat{Y}_{B}\right\} & \left\{{}^{A}\hat{Z}_{B}\right\} \\ | & | & | \end{bmatrix}$$

Here is the figure for the first interpretation, using our numerical example.



2) Transform Mapping

Homogeneous Transformation Matrix $\left[{}^{A}_{B}T \right]$ maps $\left\{ {}^{B}P \right\} \rightarrow \left\{ {}^{A}P \right\}$. This converts the description of a vector known in one Cartesian coordinate basis frame $\{B\}$ into another basis frame $\{A\}$. There are both position and orientation differences in general.

The problem statement is:

Given $\left\{ {}^{B}P \right\}$, find $\left\{ {}^{A}P \right\}$ using:

$$\begin{Bmatrix} {}^{A}P \\ 1 \end{Bmatrix} = \left[{}^{A}_{B}T \right] \begin{Bmatrix} {}^{B}P \\ 1 \end{Bmatrix}$$

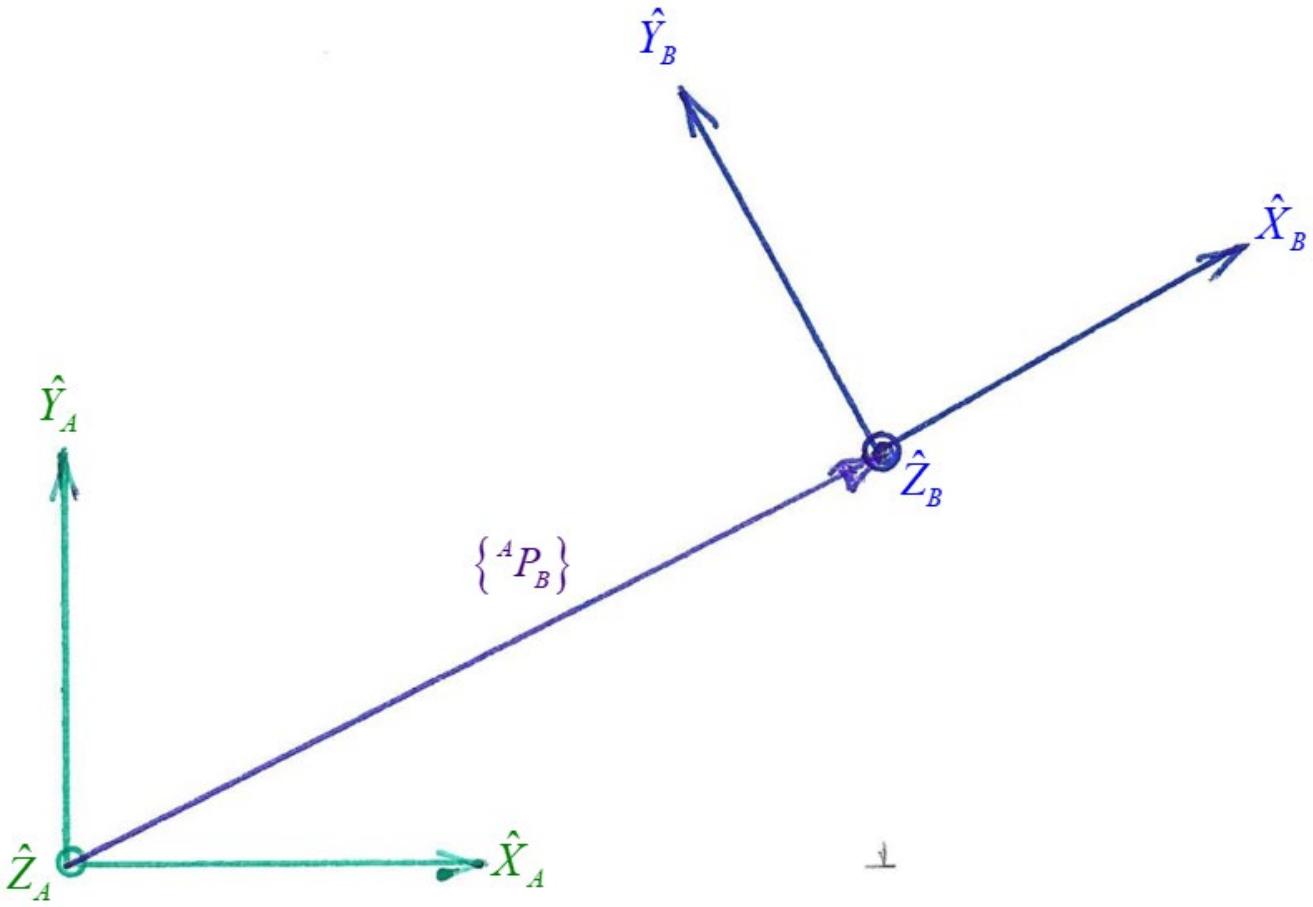
Validate the results using a sketch.

Example: Given the numerical $\left[{}^{A}_{B}T \right]$ from above and $\left\{ {}^{B}P \right\} = \begin{Bmatrix} 1 \\ \sqrt{3} \\ 0 \end{Bmatrix}$.

$$\begin{Bmatrix} {}^{A}P \\ 1 \end{Bmatrix} = \begin{bmatrix} 0.866 & -0.5 & 0 & 2 \\ 0.5 & 0.866 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} 1 \\ \sqrt{3} \\ 0 \\ 1 \end{Bmatrix} =$$

Note: $\left\{ {}^{A}P \right\} = \left\{ {}^{A}P_{B} \right\} + \left[{}^{A}_{B}R \right] \left\{ {}^{B}P \right\}$ and 1 = 1 is equivalent to $\begin{Bmatrix} {}^{A}P \\ 1 \end{Bmatrix} = \left[{}^{A}_{B}T \right] \begin{Bmatrix} {}^{B}P \\ 1 \end{Bmatrix}$.

Here is the figure for the second interpretation in this numerical example.



3) Transform Operator

Homogeneous Transformation Matrix [T] operates on $\left\{ {}^{A}P_1 \right\}$ to yield $\left\{ {}^{A}P_2 \right\}$. That is, [$T$] physically translates and rotates original vector $\left\{ {}^{A}P_1 \right\}$ to a new position and orientation, $\left\{ {}^{A}P_2 \right\}$. There is no frame $\{B\}$ for this interpretation, but we only use the Cartesian coordinate frame $\{A\}$. Using the Transform Operator, the original vector $\left\{ {}^{A}P_1 \right\}$ is translated and rotated to a new vector $\left\{ {}^{A}P_2 \right\}$. The amount and axes of the translation and rotation are found in the Transform Operator matrix [T]. The order of 3D translations and rotations does not matter if we assume rotations always occur about the vector tail. We still use **Z-Y-X (α–β–γ)** Euler angle convention, and that order still matters very much.

The problem statement is:

Given $\left\{ {}^{A}P_1 \right\}$, find $\left\{ {}^{A}P_2 \right\}$ using:

$$\begin{Bmatrix} {}^{A}P_2 \\ 1 \end{Bmatrix} = [T] \begin{Bmatrix} {}^{A}P_1 \\ 1 \end{Bmatrix}$$

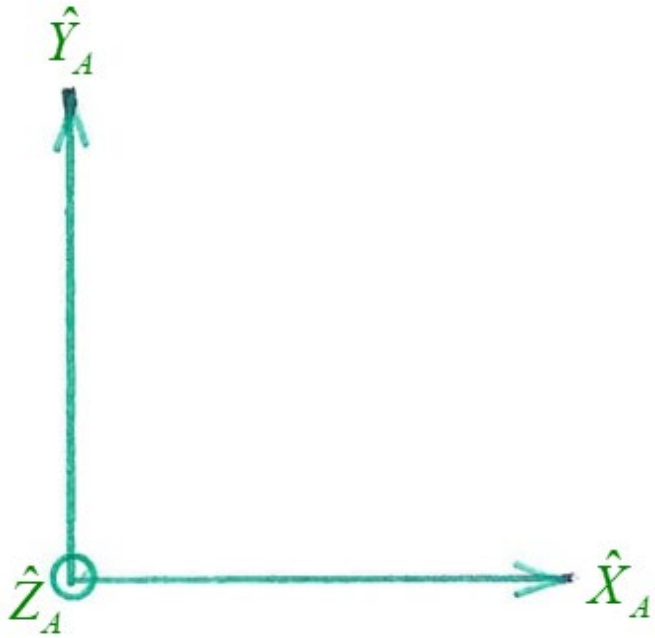
Validate the results using a sketch.

Example: Let [T] be the same numerical $\left[{}^{A}_{B}T \right]$ given above and let $\left\{ {}^{A}P_1 \right\} = \begin{Bmatrix} 1 \\ \sqrt{3} \\ 0 \end{Bmatrix}$.

$$\begin{Bmatrix} {}^{A}P \\ 1 \end{Bmatrix} = \begin{bmatrix} 0.866 & -0.5 & 0 & 2 \\ 0.5 & 0.866 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} 1 \\ \sqrt{3} \\ 0 \\ 1 \end{Bmatrix}$$

$$\begin{Bmatrix} {}^{A}P \\ 1 \end{Bmatrix} = \begin{bmatrix} \frac{\sqrt{3}}{2} & -\frac{1}{2} & 0 & 2 \\ \frac{1}{2} & \frac{\sqrt{3}}{2} & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} 1 \\ \sqrt{3} \\ 0 \\ 1 \end{Bmatrix} = \begin{Bmatrix} \frac{\sqrt{3}}{2}(1) - \frac{1}{2}(\sqrt{3}) + 0(0) + 2(1) \\ \frac{1}{2}(1) + \frac{\sqrt{3}}{2}(\sqrt{3}) + 0(0) + 1(1) \\ 0(1) + 0(\sqrt{3}) + 1(0) + 0(1) \\ 0(1) + 0(\sqrt{3}) + 0(0) + 1(1) \end{Bmatrix} = \begin{Bmatrix} 2 \\ 3 \\ 0 \\ 1 \end{Bmatrix}$$

Here is the figure for the third interpretation in this numerical example.



Homogeneous Transformation Matrix Inversion

Given $\left[{}^{A}_{B}T \right]$, find the opposite Homogeneous Transformation Matrix, i.e. the matrix representing the pose of Cartesian coordinate frame $\{A\}$ with respect to $\{B\}$. This is the matrix inverse of $\left[{}^{A}_{B}T \right]$.

$$\left[{}^{B}_{A}T \right] = \left[{}^{A}_{B}T \right]^{-1}$$

We can just use matrix inversion (MATLAB function **`inv`**), but this is computationally expensive, may be numerically unstable, and doesn't take advantage of the structure of Homogeneous Transformation Matrices. Alternatively, Gaussian elimination is less computationally expensive and more robust numerically, but it still doesn't take advantage of the structure of Homogeneous Transformation Matrices. So let us do this inversion by taking advantage of the Homogeneous Transformation Matrix structure.

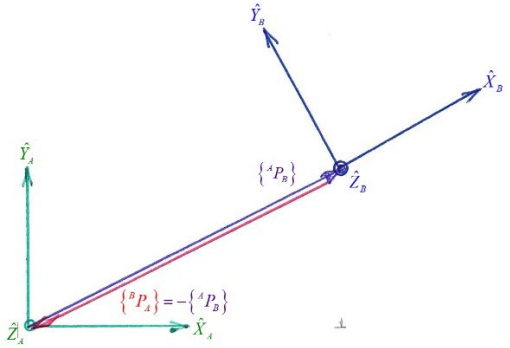
$$\left[{}^{A}_{B}T \right]^{-1} = \left[{}^{B}_{A}T \right] =$$

Recall the ***Beautiful Property*** from Orthonormal Rotation Matrices:

Translational part

Without regard for frame of expression:

With regard to frame of expression:



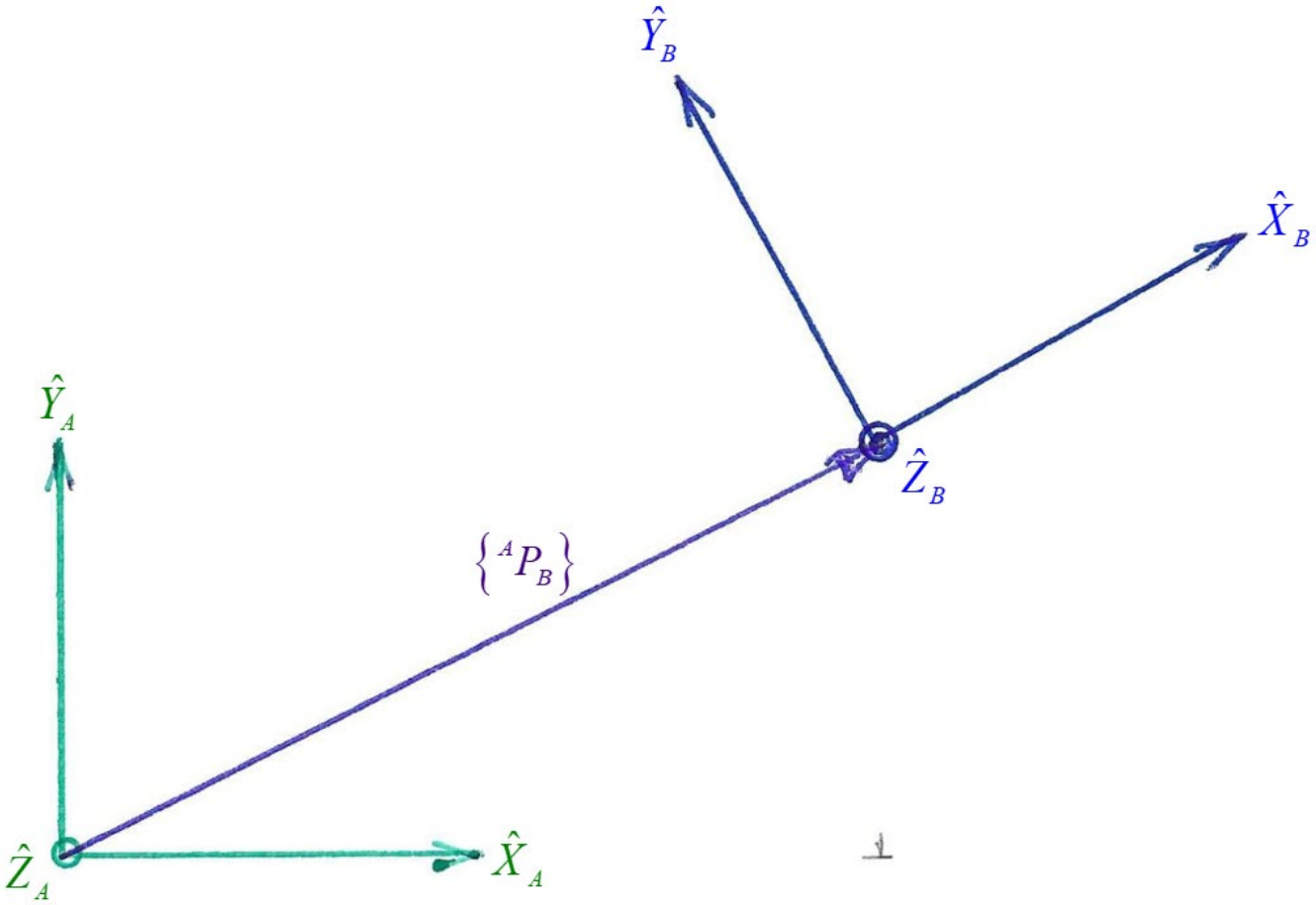
The formula is $\left[{}^{A}_{B}T \right]^{-1} =$

Example for Homogeneous Transformation Matrix Inversion.

Given $\left[{}^{A}_{B}T\right] = \begin{bmatrix} 0.866 & -0.5 & 0 & 2 \\ 0.5 & 0.866 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ find $\left[{}^{B}_{A}T\right] = \left[{}^{A}_{B}T\right]^{-1}$.

Answer: $\left[{}^{A}_{B}T\right]^{-1} = \begin{bmatrix} 0.866 & 0.5 & 0 & -2.232 \\ -0.5 & 0.866 & 0 & 0.134 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ (check via MATLAB function **`inv`**).

The figure for this numerical inverse Homogeneous Transformation Matrix example is given below.



2.2.2 Transform Equations

We will represent a serial robot by a series of coordinate frames, with one moving Cartesian coordinate frame attached to each moving rigid link at each active joint.

The pose of consecutive Cartesian coordinate frames $\{i\}$ with respect to $\{i–1\}$ is represented by $\left[{}^{i-1}_{i}T \right]$. A vector $\left\{ {}^{i}P \right\}$ is mapped back to $\left\{ {}^{i-1}P \right\}$ by $\left[{}^{i-1}_{i}T \right]$ (using the second Homogeneous Transformation Matrix interpretation, Transform Mapping):

$$\begin{Bmatrix} {}^{i-1}P \\ 1 \end{Bmatrix} = \left[{}^{i-1}_{i}T \right] \begin{Bmatrix} {}^{i}P \\ 1 \end{Bmatrix}$$

For a general robot, there are many such neighboring frame transformations (e.g. 7 for a 6-dof serial robot). Example: Frames $\{A\}$, $\{B\}$, $\{C\}$, and $\{D\}$ are neighboring frames that lie along a serial robot chain, attached to consecutive active joints.

Given $\left\{ {}^{D}P \right\}$ (e.g. the tool location known in the last frame $\{D\}$), find $\left\{ {}^{A}P \right\}$ (e.g. the tool location in the base frame $\{A\}$):

The associated transform graph is useful to understand the relationships amongst the various frames (for clarity, dots represent frames, arrows represent transforms; reverse arrow for inverse transform):

Now, given any three of these Homogeneous Transformation Matrices, we are able to calculate the fourth using linear algebra and matrix inversion. Here is an example:

Given $\left[{}^{A}_{D}T \right]$, $\left[{}^{A}_{B}T \right]$, and $\left[{}^{C}_{D}T \right]$, find $\left[{}^{B}_{C}T \right]$.

The associated transform graph shows this mathematical operation:

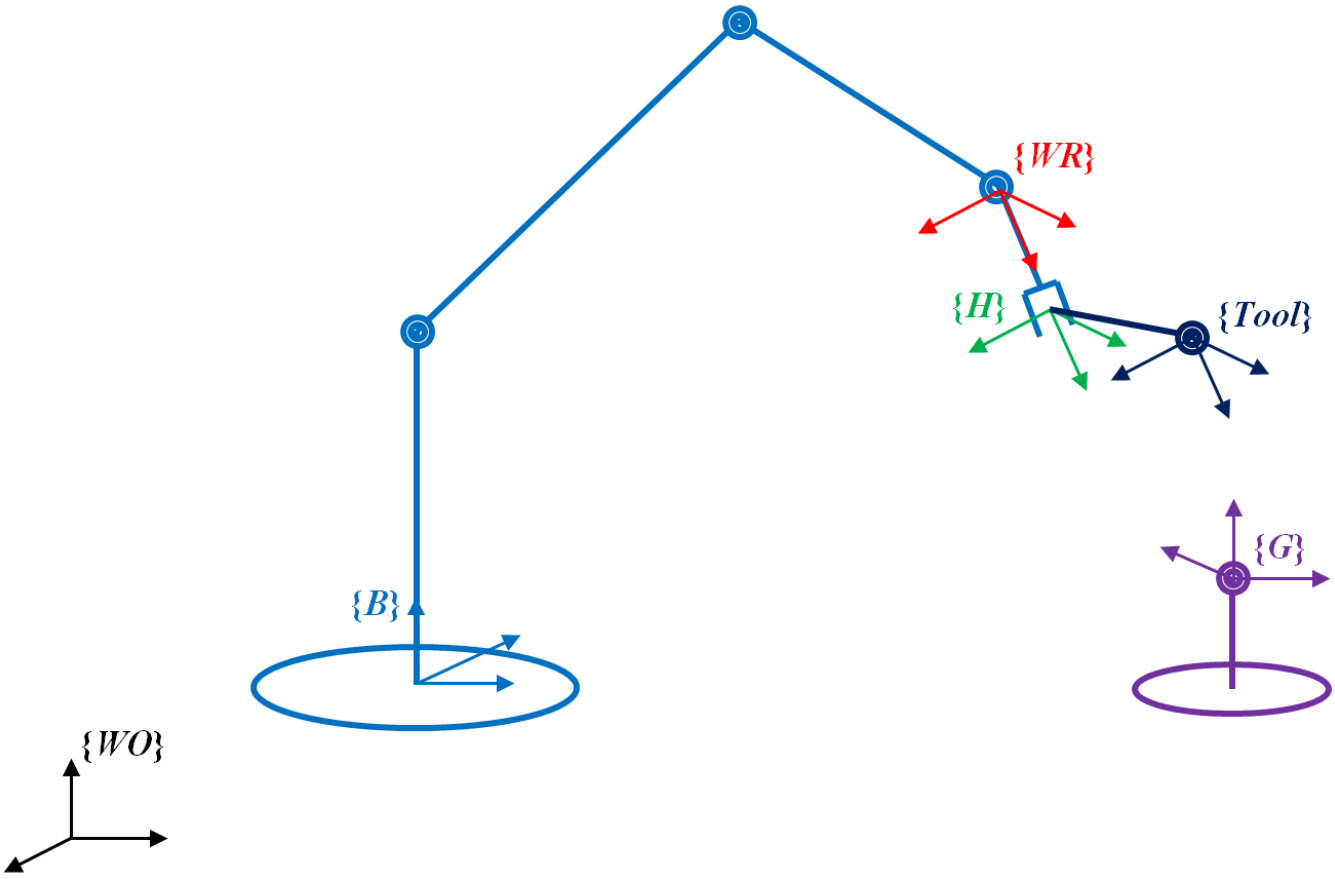
A robotics transform equations example starts on the next page.

Robotics Transform Equations Example

Frames:

{*WO*}	World	{*B*}	Base	{*WR*}	Wrist
{*H*}	Hand	{*Tool*}	Tool	{*G*}	Goal

Figure for Example:



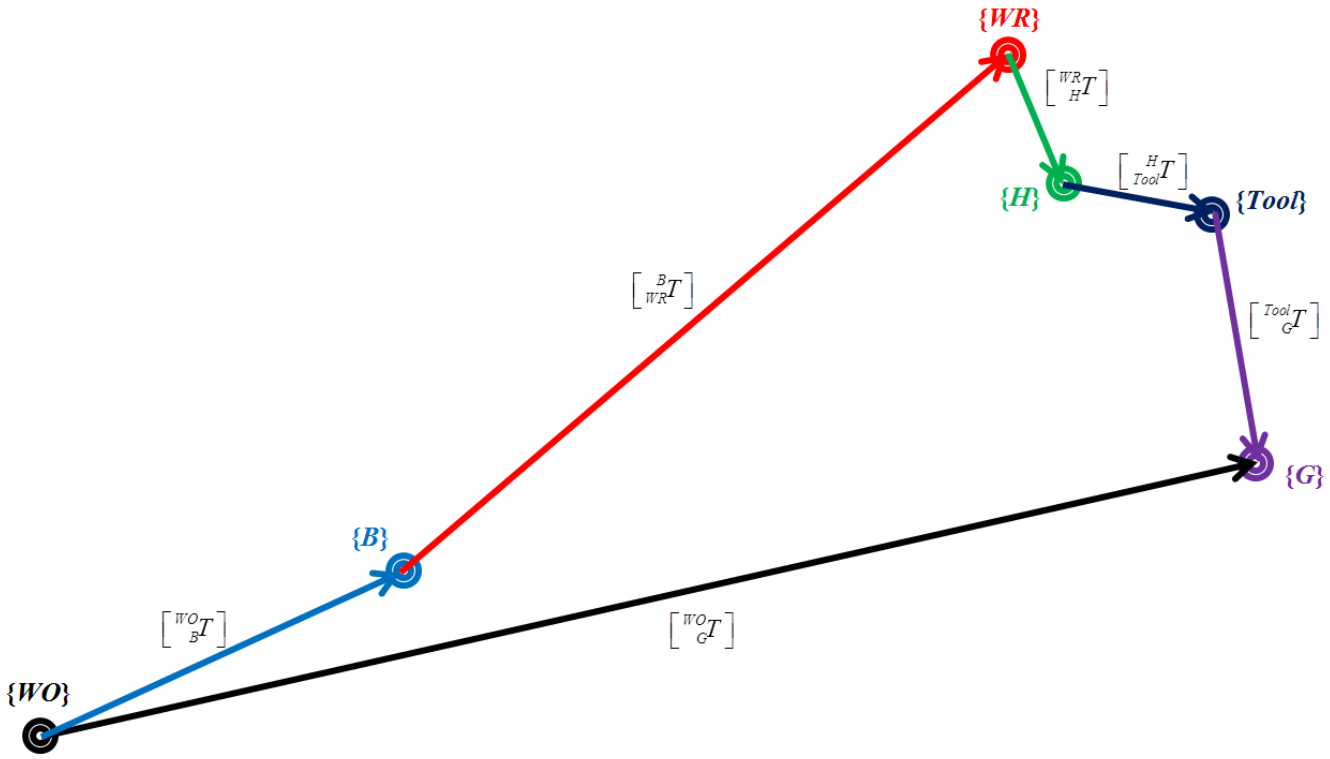
Given constant known matrices:

$$\left[{}^{WO}_{B}T\right],\left[{}^{WR}_{H}T\right],\left[{}^{H}_{Tool}T\right],\left[{}^{WO}_{G}T\right]$$

Note: we are assuming that the robot base is fixed and the robot goal is also fixed. If the robot base is moving on a linear track, then we will say $\left[{}^{WO}_{B}T\right]$ is known but not constant. If the robot goal is moving, such as on a conveyor belt, then we will say $\left[{}^{WO}_{G}T\right]$ is known but not constant. $\left[{}^{WR}_{H}T\right]$ is always constant; if not, something is broken in your robot (i.e. there are no active joints between the robot wrist and hand). Also, $\left[{}^{H}_{Tool}T\right]$ is a known constant matrix, otherwise the grip of the hand on the tool has slipped.

For this example robot scenario, the transform loop-closure equation and associated Transform Graph are given below:

$$\left[{}^{WO}_{G}T\right]=\left[{}^{WO}_{B}T\right]\left[{}^{B}_{WR}T\right]\left[{}^{WR}_{H}T\right]\left[{}^{H}_{Tool}T\right]\left[{}^{Tool}_{G}T\right]$$



1) Given $\left[{}^{Tool}_{G}T\right]$ from a machine vision algorithm. Calculate $\left[{}^{B}_{WR}T\right]$ resulting from machine vision – we could compare with a different $\left[{}^{B}_{WR}T\right]$, calculated by reading joint angle encoders and using Forward Pose Kinematics (later in Chapter 4).

Solution: We start with the above transform loop-closure equation. To solve for $\left[{}^{B}_{WR}T\right]$ we must multiply by four inverse matrices, on the correct side of each transform matrix and on both sides of the equation.

First we can remove $\left[{}^{WO}_{B}T\right]$ and $\left[{}^{Tool}_{G}T\right]$ from the RHS by pre-multiplying by $\left[{}^{WO}_{B}T\right]^{-1}$ on both sides of the transform equation and also by post-multiplying by $\left[{}^{Tool}_{G}T\right]^{-1}$ on both sides of the transform equation:

$$\left[{}^{WO}_{G}T\right]=\left[{}^{WO}_{B}T\right]\left[{}^{B}_{WR}T\right]\left[{}^{WR}_{H}T\right]\left[{}^{H}_{Tool}T\right]\left[{}^{Tool}_{G}T\right]$$

$$\left[{}^{WO}_{B}T\right]^{-1}\left[{}^{WO}_{G}T\right]\left[{}^{Tool}_{G}T\right]^{-1}=\left[{}^{WO}_{B}T\right]^{-1}\left[{}^{WO}_{B}T\right]\left[{}^{B}_{WR}T\right]\left[{}^{WR}_{H}T\right]\left[{}^{H}_{Tool}T\right]\left[{}^{Tool}_{G}T\right]\left[{}^{Tool}_{G}T\right]^{-1}$$

Simplifying by using $[A]^{-1}[A]=[I_4]$ and $[B][B]^{-1}=[I_4]$, plus $[I_4][C]=[C]$ and $[D][I_4]=[D]$ yields:

$$\left[{}^{WO}_{B}T\right]^{-1}\left[{}^{WO}_{G}T\right]\left[{}^{Tool}_{G}T\right]^{-1}=\left[{}^{B}_{WR}T\right]\left[{}^{WR}_{H}T\right]\left[{}^{H}_{Tool}T\right]$$

Now we can remove $\left[{}^{H}_{Tool}T\right]$ from the RHS by post-multiplying by $\left[{}^{H}_{Tool}T\right]^{-1}$ on both sides of the transform equation and simplifying using $[B][B]^{-1}=[I_4]$ and $[D][I_4]=[D]$:

$$\left[{}^{WO}_{B}T\right]^{-1}\left[{}^{WO}_{G}T\right]\left[{}^{Tool}_{G}T\right]^{-1}\left[{}^{H}_{Tool}T\right]^{-1}=\left[{}^{B}_{WR}T\right]\left[{}^{WR}_{H}T\right]\left[{}^{H}_{Tool}T\right]\left[{}^{H}_{Tool}T\right]^{-1}$$

$$\left[{}^{WO}_{B}T\right]^{-1}\left[{}^{WO}_{G}T\right]\left[{}^{Tool}_{G}T\right]^{-1}\left[{}^{H}_{Tool}T\right]^{-1}=\left[{}^{B}_{WR}T\right]\left[{}^{WR}_{H}T\right]$$

Now we can remove $\left[{}^{WR}_{H}T\right]$ from the RHS by post-multiplying by $\left[{}^{WR}_{H}T\right]^{-1}$ on both sides of the transform equation:

$$\left[{}^{WO}_{B}T\right]^{-1}\left[{}^{WO}_{G}T\right]\left[{}^{Tool}_{G}T\right]^{-1}\left[{}^{H}_{Tool}T\right]^{-1}\left[{}^{WR}_{H}T\right]^{-1}=\left[{}^{B}_{WR}T\right]\left[{}^{WR}_{H}T\right]\left[{}^{WR}_{H}T\right]^{-1}$$

Finally, again using $[B][B]^{-1}=[I_4]$ and $[D][I_4]=[D]$, and reversing sides of the equation yields the desired result:

$$\left[{}^{B}_{WR}T\right]=\left[{}^{WO}_{B}T\right]^{-1}\left[{}^{WO}_{G}T\right]\left[{}^{Tool}_{G}T\right]^{-1}\left[{}^{H}_{Tool}T\right]^{-1}\left[{}^{WR}_{H}T\right]^{-1}$$

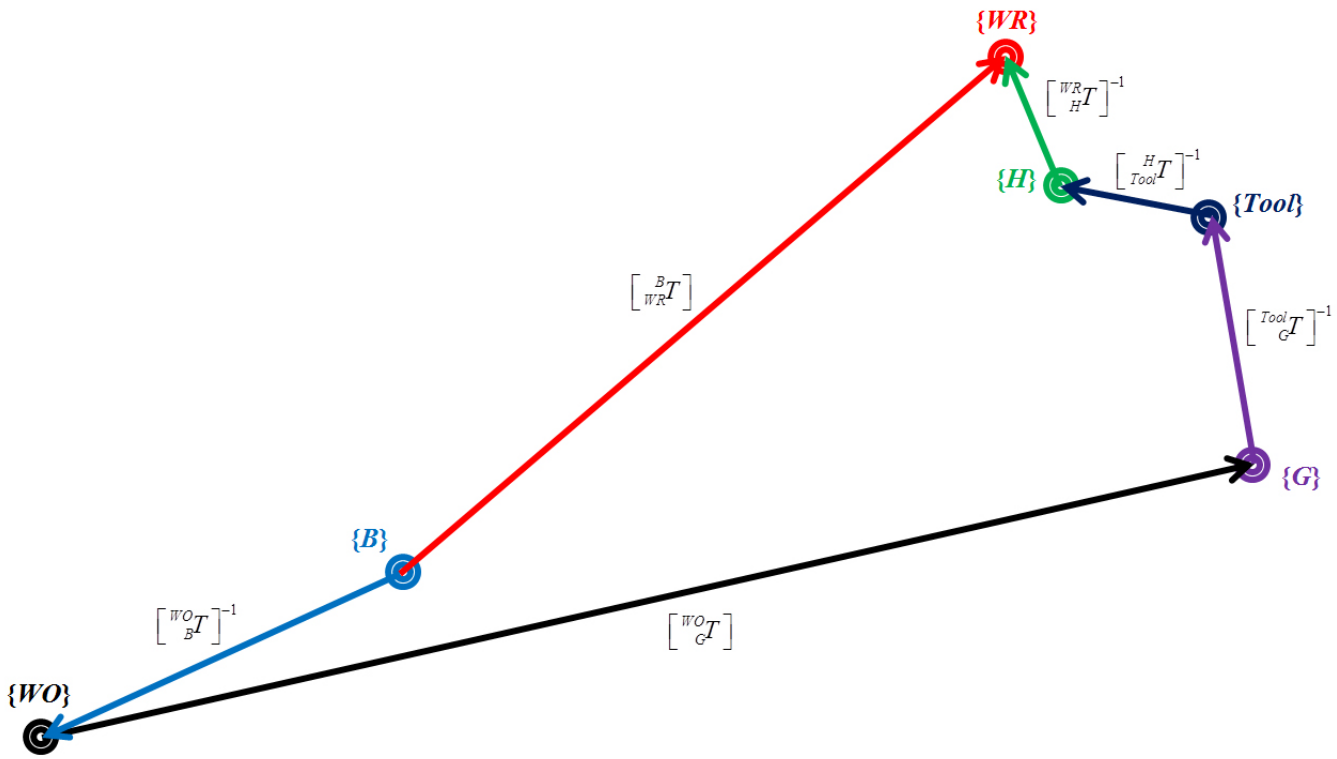
One way to validate this result is to replace the inverse matrices in the result using the following substitution, from inverse homogeneous transformation matrices:

$$\left[{}^{b}_{a}T\right]^{-1}=\left[{}^{a}_{b}T\right]$$

$$\left[{}^{B}_{WR}T\right]=\left[{}^{WO}_{B}T\right]^{-1}\left[{}^{WO}_{G}T\right]\left[{}^{Tool}_{G}T\right]^{-1}\left[{}^{H}_{Tool}T\right]^{-1}\left[{}^{WR}_{H}T\right]^{-1}$$

$$\left[{}^{B}_{WR}T\right]=\left[{}^{B}_{WO}T\right]\left[{}^{WO}_{G}T\right]\left[{}^{G}_{Tool}T\right]\left[{}^{Tool}_{H}T\right]\left[{}^{H}_{WR}T\right]$$

We see that this above result agrees with a proper transform equation, i.e. frame *WR* is the right-most subscript, frame *B* is the left-most superscript, and the *H*, *Tool*, *G* and *WO* frames 'cancel out' in the middle, hence validating our derived result. The associated Transform Graph is shown below.



Alternate Solution Form

The above solution for $\left[{}^{B}_{WR}T\right]$ in Example 1 can be expressed more simply as follows:

$$\left[{}^{B}_{WR}T\right]=\left[{}^{WO}_{B}T\right]^{-1}\left[{}^{WO}_{G}T\right]\left[{}^{Tool}_{G}T\right]^{-1}\left[{}^{H}_{Tool}T\right]^{-1}\left[{}^{WR}_{H}T\right]^{-1}$$

$$\left[{}^{B}_{WR}T\right]=\left[{}^{WO}_{B}T\right]^{-1}\left[{}^{WO}_{G}T\right]\left(\left[{}^{WR}_{H}T\right]\left[{}^{H}_{Tool}T\right]\left[{}^{Tool}_{G}T\right]\right)^{-1}$$

$$\left[{}^{B}_{WR}T\right]=\left[{}^{WO}_{B}T\right]^{-1}\left[{}^{WO}_{G}T\right]\left[{}^{WR}_{G}T\right]^{-1}$$

where:

$$\left[{}^{WR}_{G}T\right]^{-1}=\left(\left[{}^{WR}_{H}T\right]\left[{}^{H}_{Tool}T\right]\left[{}^{Tool}_{G}T\right]\right)^{-1}=\left[{}^{Tool}_{G}T\right]^{-1}\left[{}^{H}_{Tool}T\right]^{-1}\left[{}^{WR}_{H}T\right]^{-1}$$

and we have used the following fact from matrix algebra:

$$\left[[A][B][C]\right]^{-1}=[C]^{-1}[B]^{-1}[A]^{-1}$$

2) This second example is for the identical robot scenario as Example 1 above. Now assume we want the $\{Tool\}$ frame to be the same as the $\{G\}$ frame, i.e. we wish the robot to move so that the tool is aligned with the goal object. Now we calculate the 4x4 Homogeneous Transformation Matrix $\left[{}^{B}_{WR}T\right]$ to achieve that desired robot pose (position and orientation).

In this case:

$$\left[{}^{Tool}_{G}T\right]=\left[{}^{G}_{Tool}T\right]=\left[I_4\right]$$

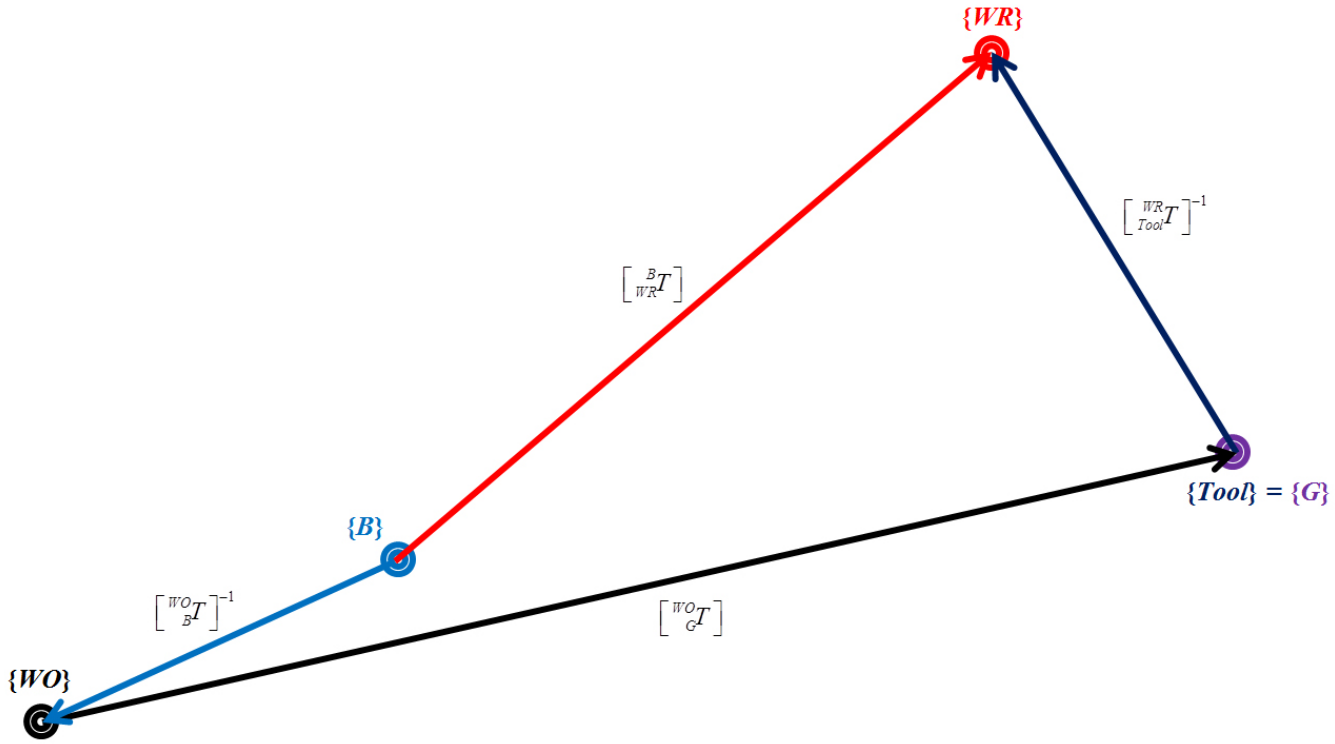
And so the desired answer for $\left[{}^{B}_{WR}T\right]$ is:

$$\left[{}^{B}_{WR}T\right]=\left[{}^{WO}_{B}T\right]^{-1}\left[{}^{WO}_{G}T\right]\left[{}^{WR}_{Tool}T\right]^{-1}$$

Validating this second result:

$$\begin{aligned}\left[{}^{B}_{WR}T\right]&=\left[{}^{WO}_{B}T\right]^{-1}\left[{}^{WO}_{G}T\right]\left[{}^{WR}_{Tool}T\right]^{-1}\\&=\left[{}^{B}_{WO}T\right]\left[{}^{WO}_{G}T\right]\left[{}^{Tool}_{WR}T\right]\end{aligned}$$

We see that this above result agrees with a proper transform equation, i.e. frame *WR* is the right-most subscript, frame *B* is the left-most superscript, and the *Tool* (identical to *G* in this example) and *WO* frames 'cancel out' in the middle, hence validating our derived result. The associated Transform Graph is shown below.



3. Denavit-Hartenberg (DH) Parameters

Denavit-Hartenberg (DH) Parameters[1] are a standard description of the geometric configuration of joints and links in a serial robot. Caution: The Paul[2] interpretation of DH Parameters is different from that of Craig[3]. Both are in use in robotics currently and we will use only Craig convention.

There are four parameters to describe completely, in general, the pose relationship between consecutive Cartesian coordinate frames $\{i\}$ with respect to $\{i–1\}$. Joints $i–1$ and i are connected by link $i–1$. Cartesian coordinate frame $\{i–1\}$ is rigidly attached to link $i–1$ and it moves with joint $i–1$.

The four DH parameters are two rotations and two translations. There is only one control variable out of the four DH parameters, so the other three parameters are constants. There are four transform operators to change from Cartesian frame $\{i–1\}$ to Cartesian frame $\{i\}$ (that statement was Interpretation #3 of the Homogeneous Transformation Matrix; or, to give the pose of frame $\{i\}$ with respect to frame $\{i–1\}$, which is Interpretation #1 of the Homogeneous Transformation Matrix).

Frame Attachment Conventions

1) The Z axis is **always** along the rotation direction for revolute (**R**) joints, and **always** along the translation direction for prismatic (**P**) joints.

2) The X axis is **always** directed along the common normal between consecutive Z axes; it can point either direction along the common normal. Specifically, $\hat{X}_{i-1}$ is always simultaneously perpendicular to $\hat{Z}_{i-1}$ and $\hat{Z}_i$, for all i. If consecutive Z axes intersect, X must still be perpendicular to both. If consecutive Z axes are along the same line, there are infinite choices in defining X (still perpendicular to both Z axes). The cross product defines the common normal of vectors $\hat{Z}_{i-1}$ and $\hat{Z}_i$.

3) The Y axis is formed by the right hand rule, with Z and X known. Showing only Z and X is sufficient, the drawing is made clearer by ***NOT*** showing the Y axes.

First Link

Choose $\hat{Z}_0$ along $\hat{Z}_1$, and ensure the origins and orientations of frames $\{0\}$ and $\{1\}$ are identical when the first variable is 0. This is optional but simplifies the resulting kinematics equations.

Simplify Kinematics

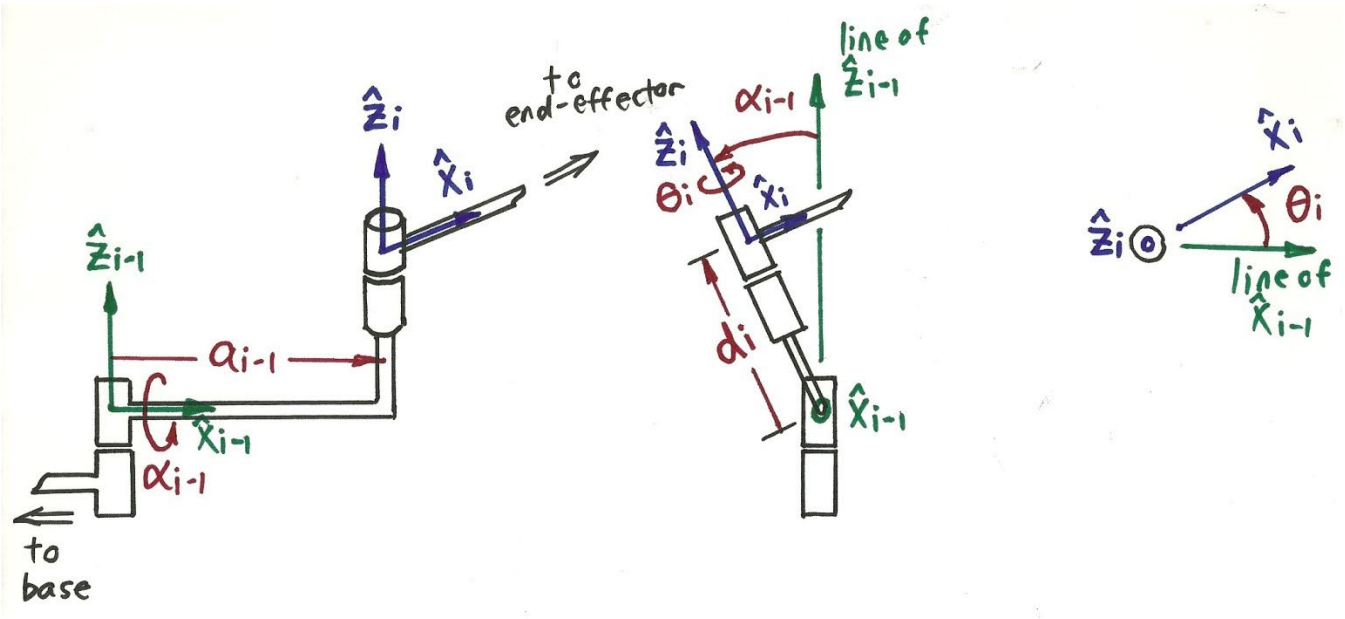
Place kinematic base $\{0\}$ coincident with $\{1\}$ rather than at the physical base of the robot.
Place final active frame $\{n\}$ at the wrist instead of at the hand or end-effector (tool) frame.

In deriving DH parameters, it is useful to imagine what frame and link moves with each active joint – do this along the entire serial chain. (The first joint moves all other links/joints, the second joint moves all links/joints beyond itself, but not the first, and so on.)

---

[1] J. Denavit and R.S. Hartenberg, 1955, A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices, Journal of Applied Mechanics.
[2] R.P. Paul, 1983, Robot Manipulators: Mathematics, Programming, and Control, MIT Press, Cambridge, MA.
[3] J.J. Craig, 2005, Introduction to Robotics: Mechanics and Control, Third edition, Pearson Prentice Hall, Upper Saddle River, NJ.

Drawing of General Link Connecting 2 R Joints (for P joint see below)



Front view (plane of $\hat{X}_{i-1}, \hat{Z}_{i-1}$) **Side view (down $\hat{X}_{i-1}$)** **Down $\hat{Z}_i$ view (pseudo top view)**

DH Parameters definitions

α_{i-1}	angle between	$\hat{Z}_{i-1}$	to $\hat{Z}_i$	measured about	$\hat{X}_{i-1}$	offset angle
a_{i-1}	distance from	$\hat{Z}_{i-1}$	to $\hat{Z}_i$	measured along	$\hat{X}_{i-1}$	link length
d_i	distance from	$\hat{X}_{i-1}$	to $\hat{X}_i$	measured along	$\hat{Z}_i$	link offset
θ_i	angle between	$\hat{X}_{i-1}$	to $\hat{X}_i$	measured about	$\hat{Z}_i$	joint angle

Screw Pairs

α_{i-1}, a_{i-1}	align Z axes about and along $\hat{X}_{i-1}$
d_i, θ_i	align X axes along and about $\hat{Z}_i$

DH Parameter Variables

For revolute (**R**) joints, the variable is θ_i, and for prismatic (**P**) joints, the variable is d_i.

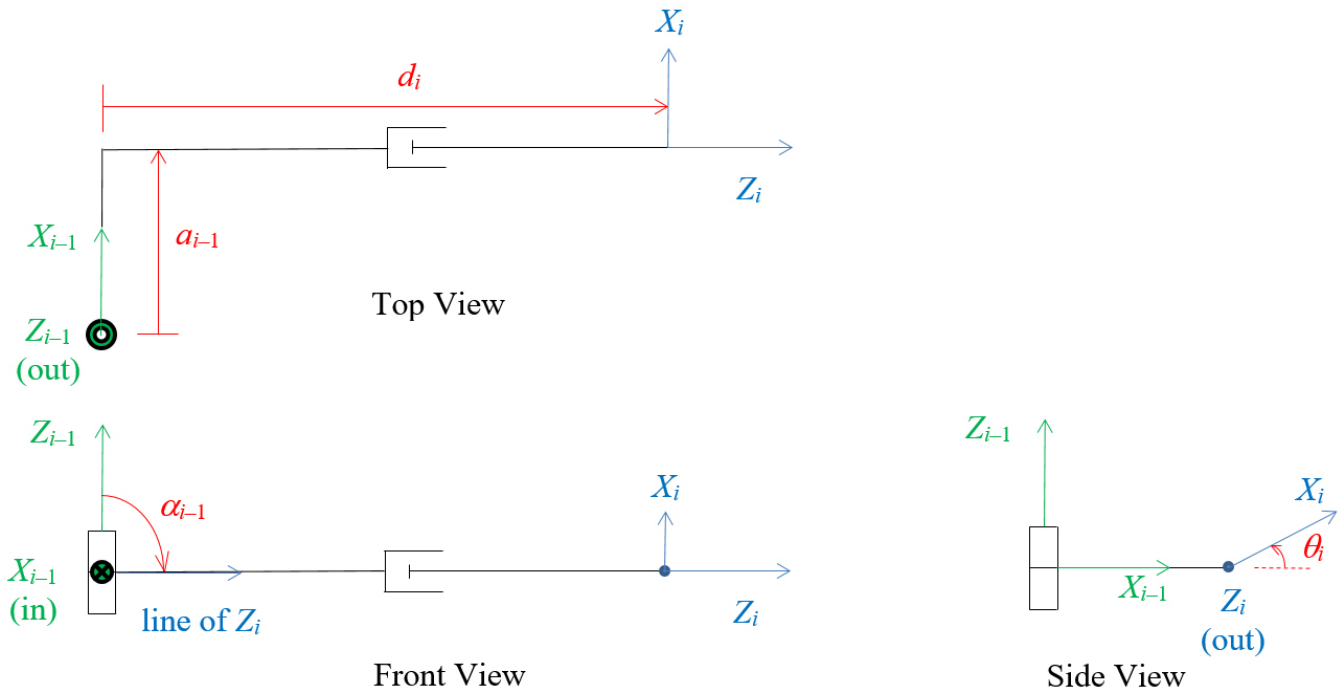
Table of DH Parameters

The columns of this table are the link/joint index *i* and the four DH parameters. The rows of this table are the four DH parameters for each active link/joint in the serial chain. A robot with *n* moving link/joint pairs has a total of 4*n* DH parameters to find and tabulate, with *n* variables, **one per row**.

i	α_{i-1}	a_{i-1}	d_i	θ_i
1				
2				
…	…	…	…	…
i				
…	…	…	…	…
n				

Drawing of General Link Connecting R and P Joints

Here is a figure showing the Denavit-Hartenberg (DH) parameters for a prismatic (**P**) joint:

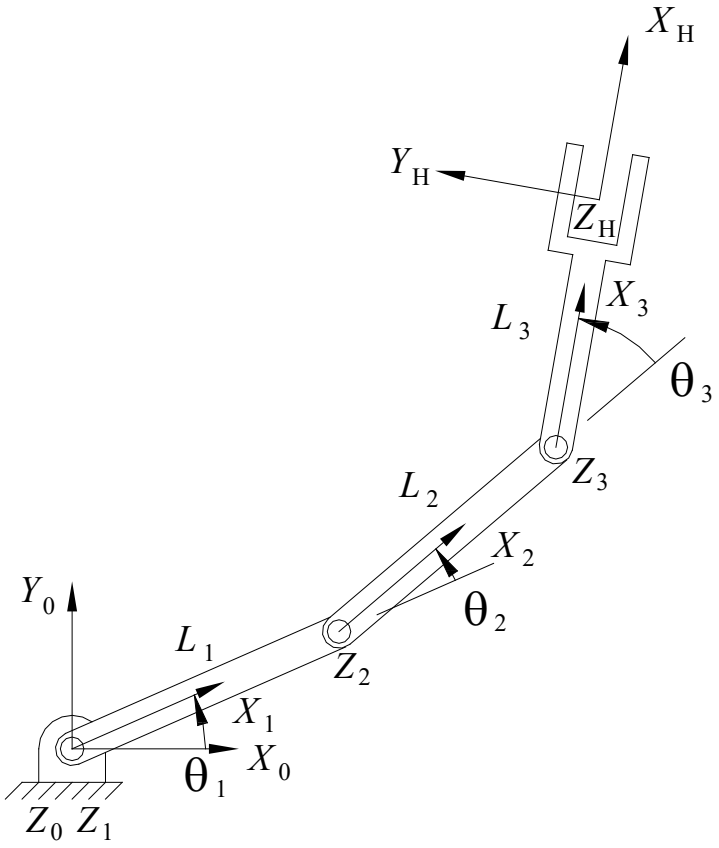


P-joint DH Parameters

Remember for each active link/joint combination, there must be 4 DH parameters, 3 constant and one variable. For the **P**-joint, the DH parameter d_i is **variable**, the length along the sliding axis Z_i. The remaining three DH parameters α_{i-1}, a_{i-1}, and θ_{i-1}, defined identically as in the previous page, are **constants** in this case.

DH Parameters Examples

1. 3-dof Planar 3R Robot



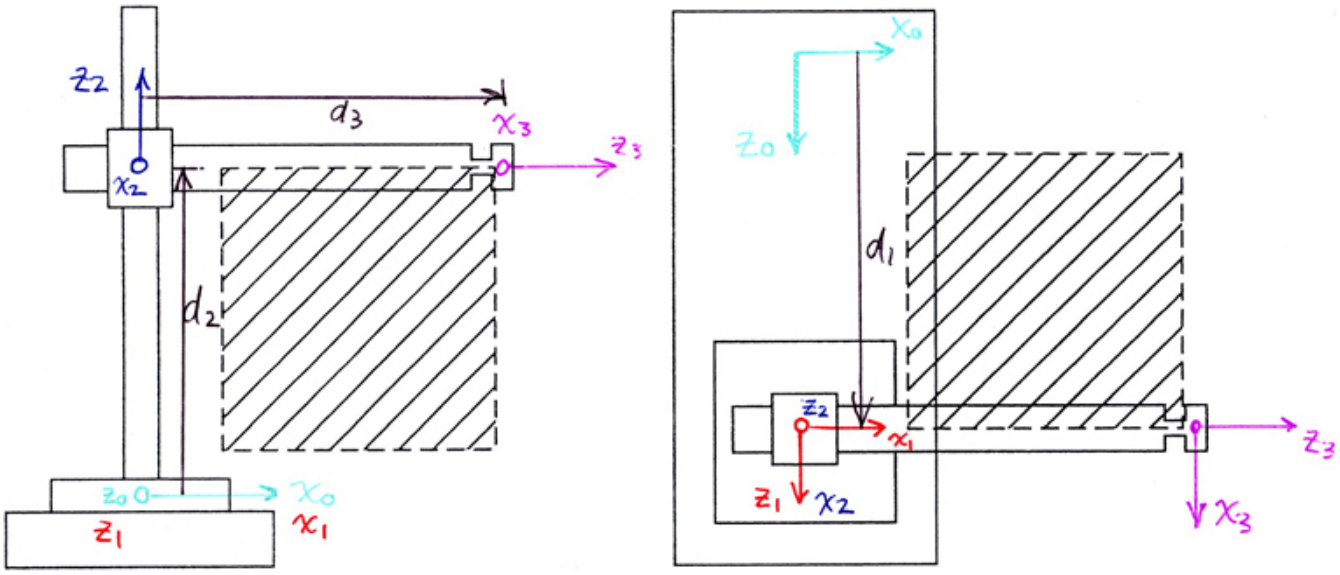
i	α_{i-1}	a_{i-1}	d_i	θ_i
1				
2				
3				

Issues

- The links are in different parallel planes, but we artificially shift all X axes to align in a single plane (so that all $d_i = 0$).
- What happened to L_3?
- In all DH Parameter Tables, there must be **one and only one variable per row**.

2. 3-dof Spatial 3P Cartesian Robot

Workspace shown in hatched lines.



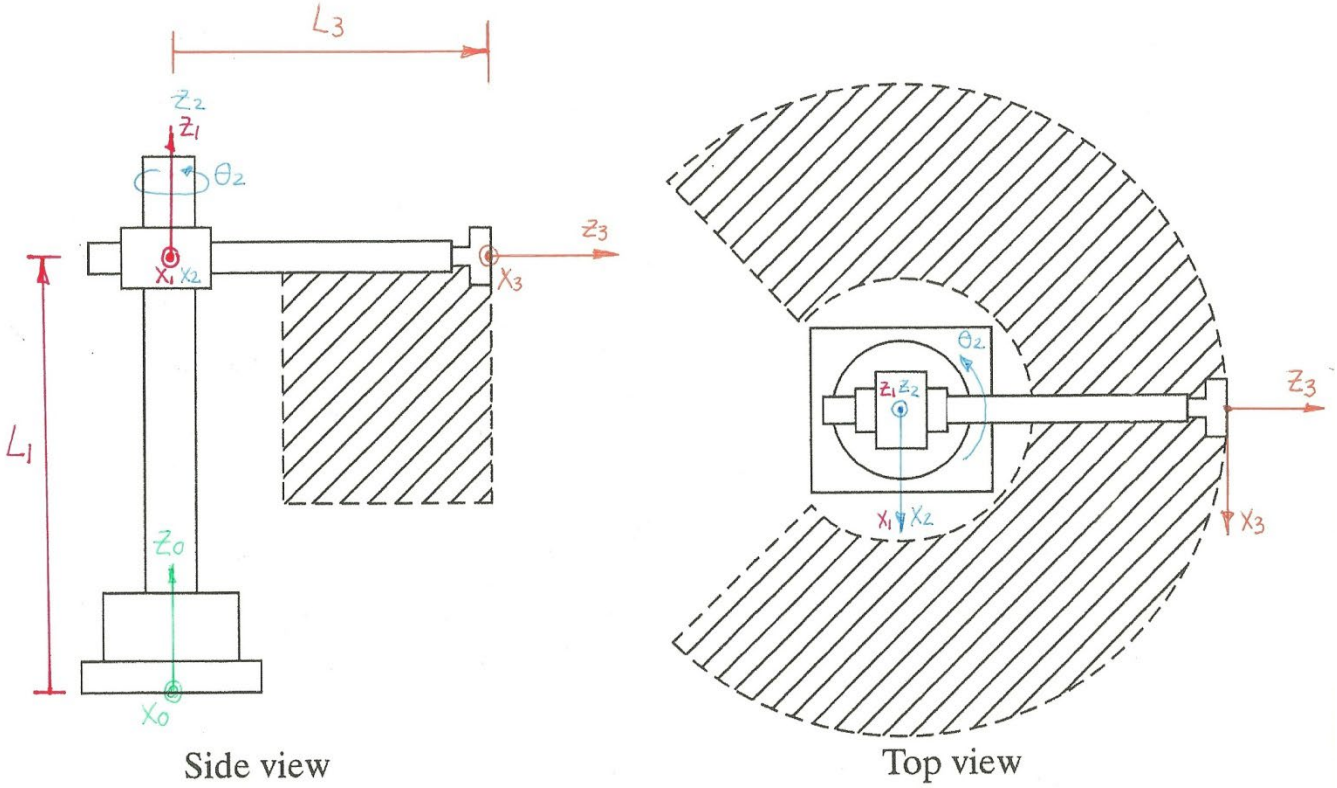
i	α_{i-1}	a_{i-1}	d_i	θ_i
1				
2				
3				

Issues

- Align each Z_i axis with each prismatic joint with length variable d_i.
- In all DH Parameter Tables, there must be **one and only one variable per row**.

3. 3-dof Spatial PRP Cylindrical Robot

Workspace shown in hatched lines.

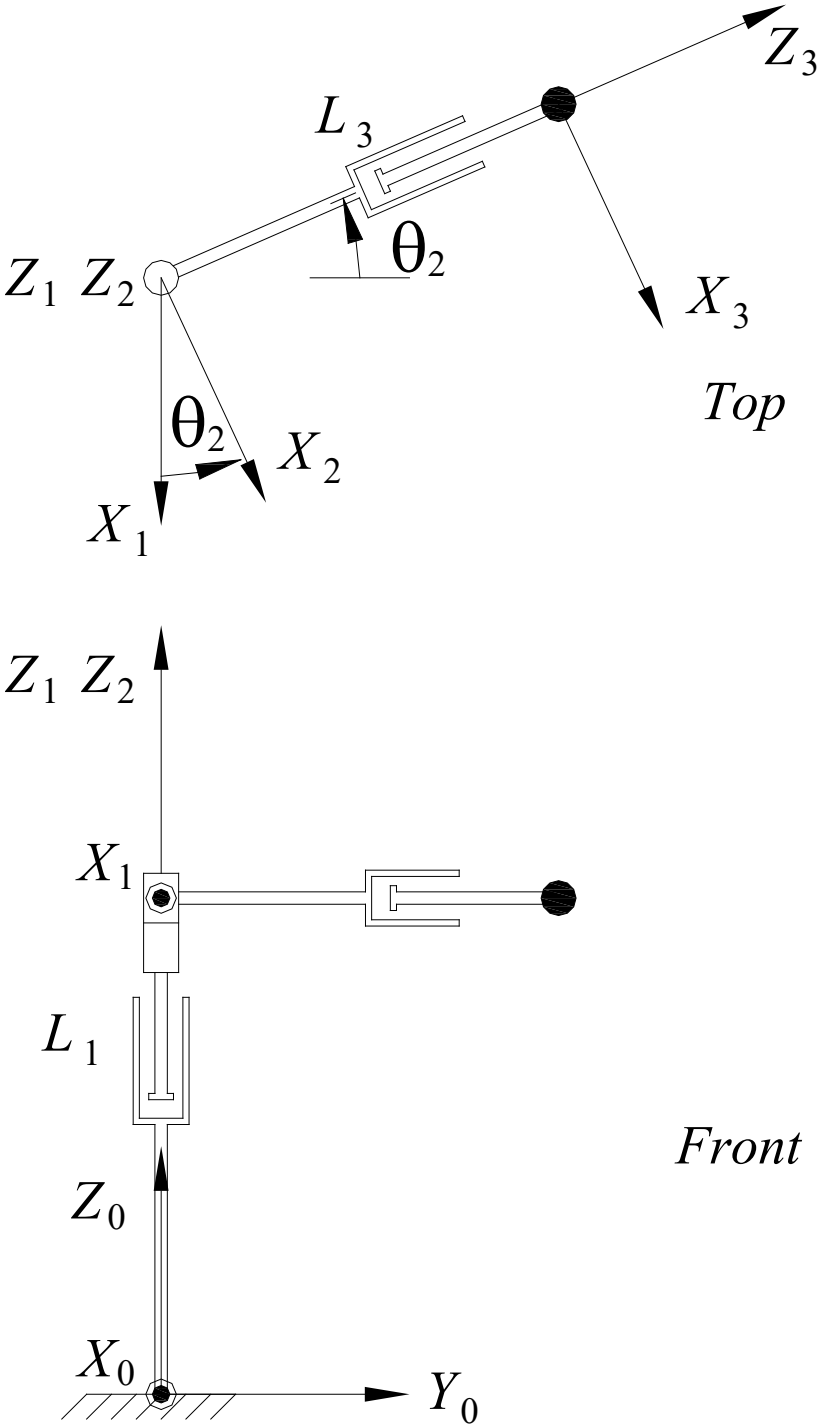


i	α_{i-1}	a_{i-1}	d_i	θ_i
1				
2				
3				

Issues

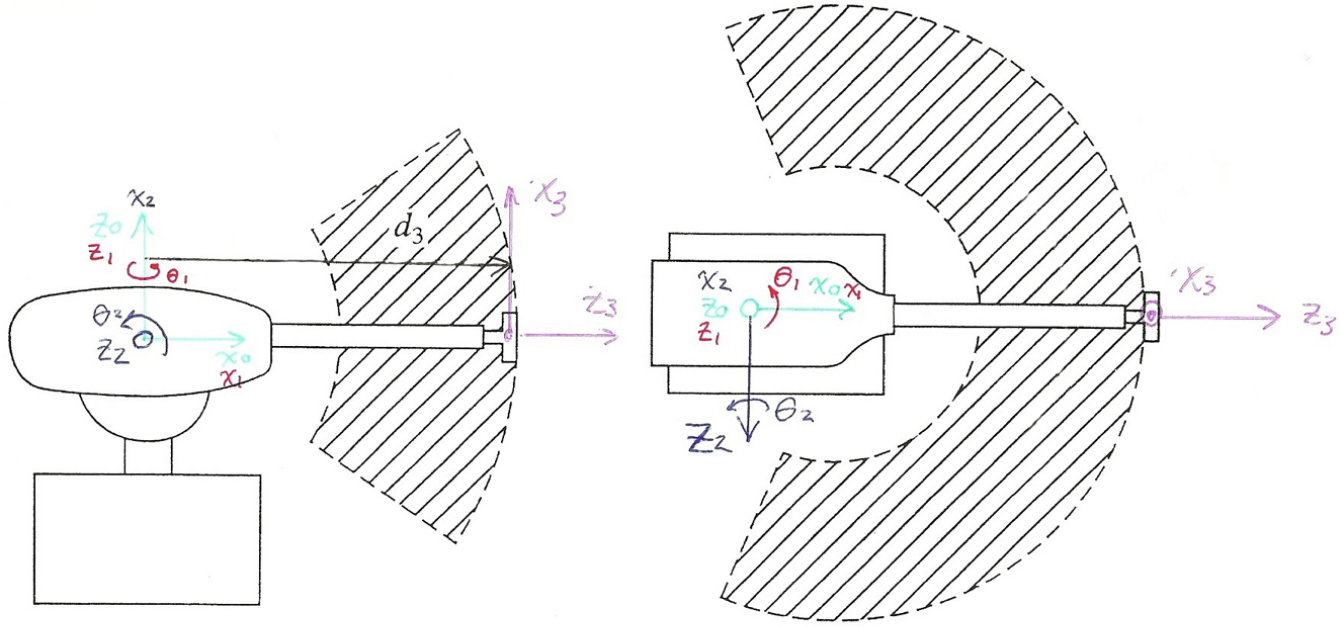
- Why is X_2 not along L_3? X_2 must be mutually perpendicular to Z_2 and Z_3 so it cannot be in the line of Z_3! X_0 and X_1 are pointing out of the page, to avoid an angle offset for θ_2.
- In all DH Parameter Tables, there must be **one and only one variable per row**.

3. 3-dof Spatial PRP Cylindrical Robot – additional images



4. 3-dof Spatial RRP Spherical Robot

Workspace shown in hatched lines.



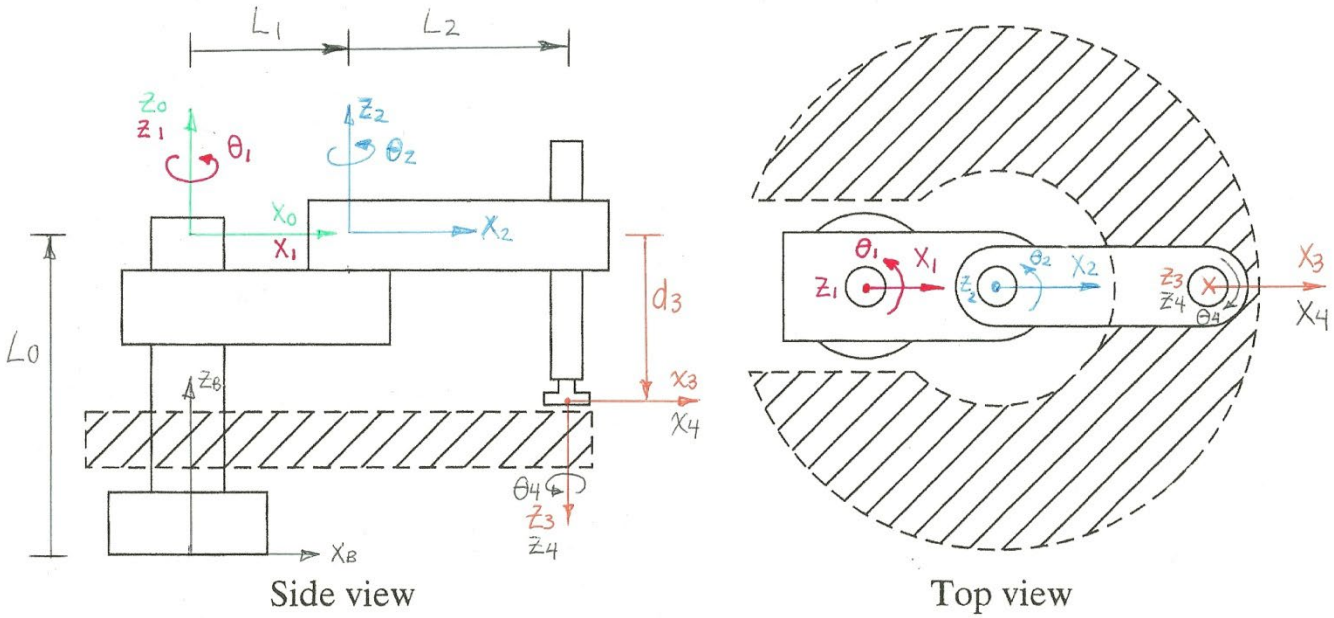
Side view

Top view

i	α_{i-1}	a_{i-1}	d_i	θ_i
1				
2				
3				

5. 4-dof Spatial RRPR SCARA Robot

Workspace shown in hatched lines.
Actually, this workspace top-view by Craig is wrong (see Section 4.4).



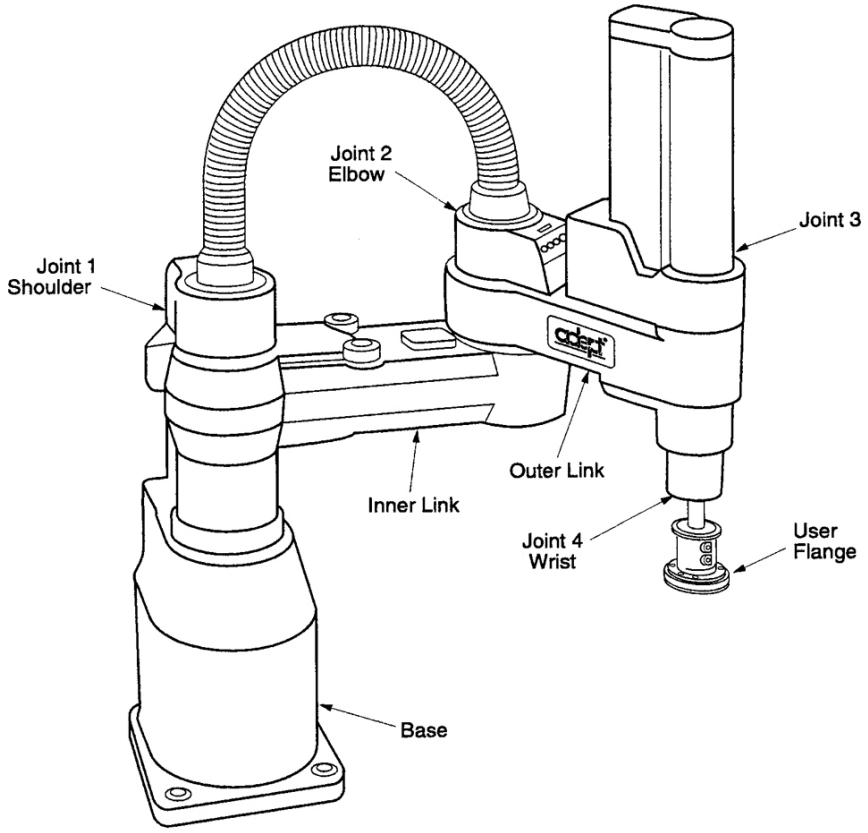
i	α_{i-1}	a_{i-1}	d_i	θ_i
1				
2				
3				
4				

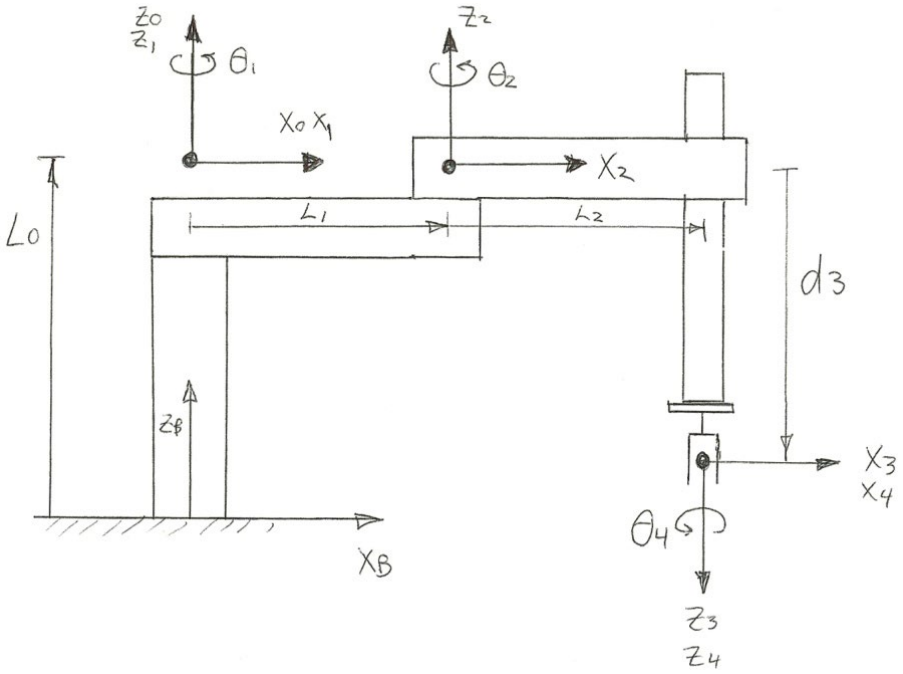
Issues

- Simplify: place $\{B\}$ on table and $\{0\}$ origin coincident with $\{1\}$, not even on the physical robot. Also place the $\{3\}$ origin coincident with the $\{4\}$ origin, at the gripper center.

- Be sure there is **one and only one variable per row**.

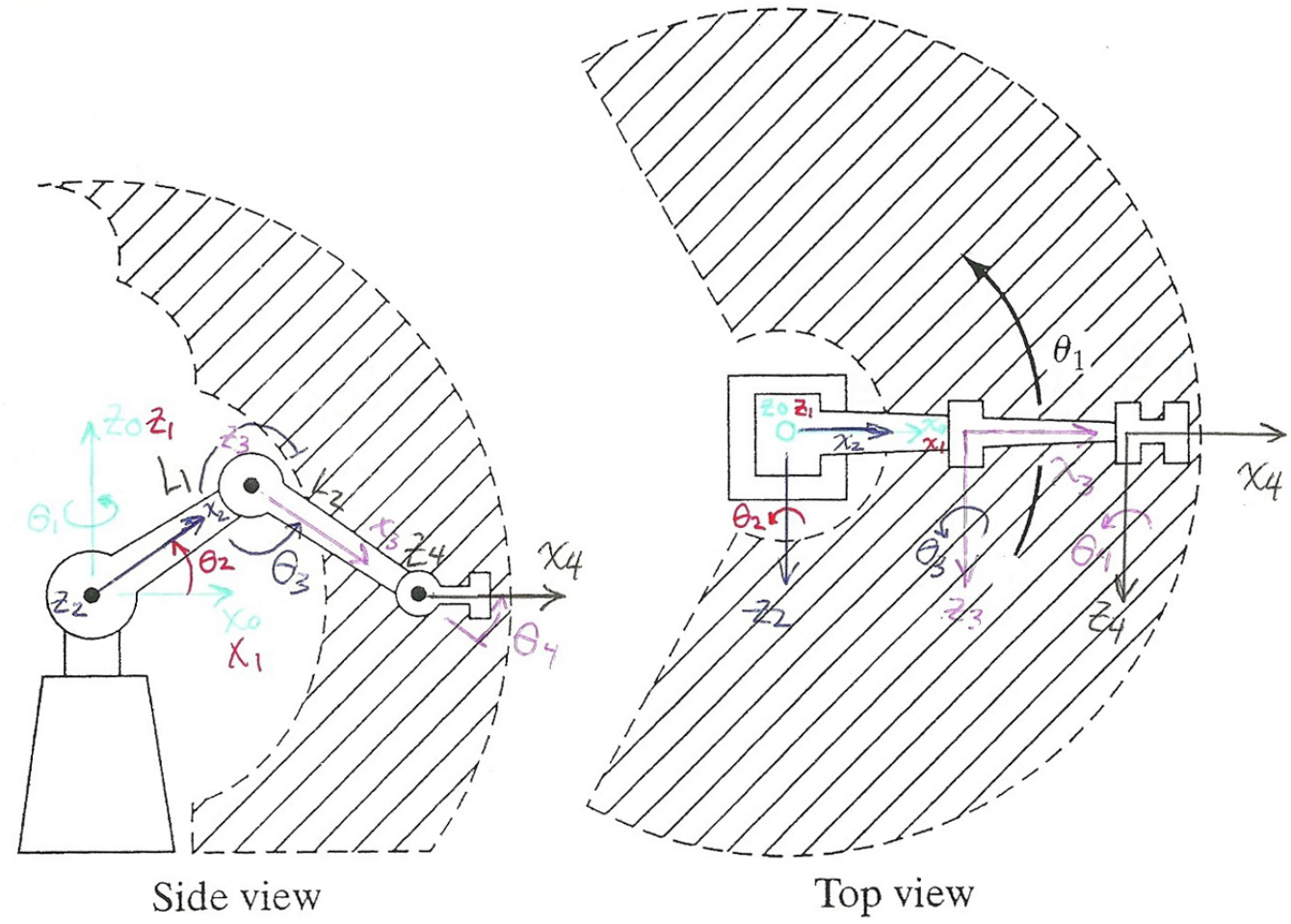
5. 4-dof Spatial RRPR SCARA Robot – additional images





6. 4-dof Spatial 4R Articulated Robot

Workspace shown in hatched lines.

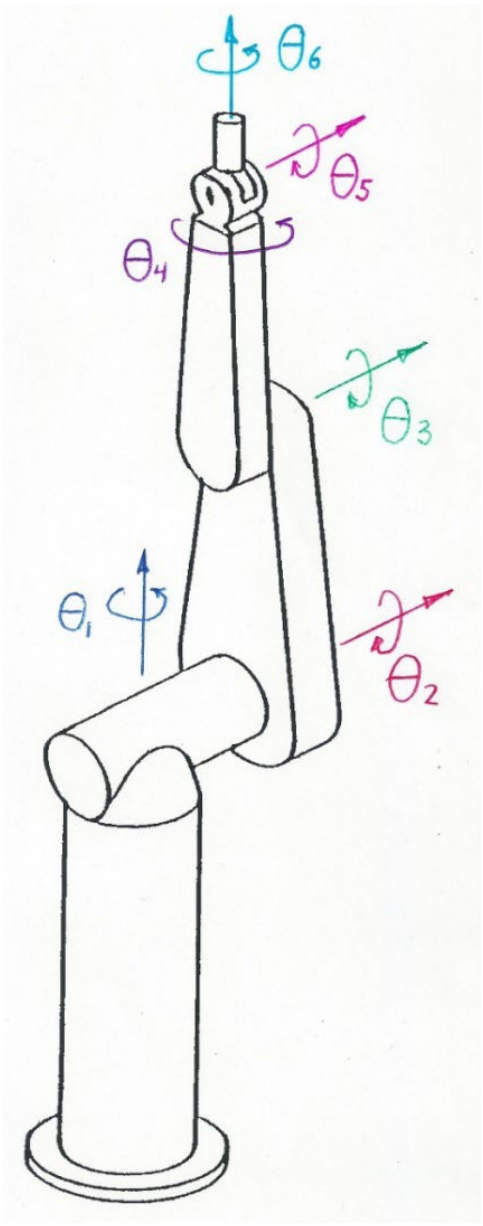


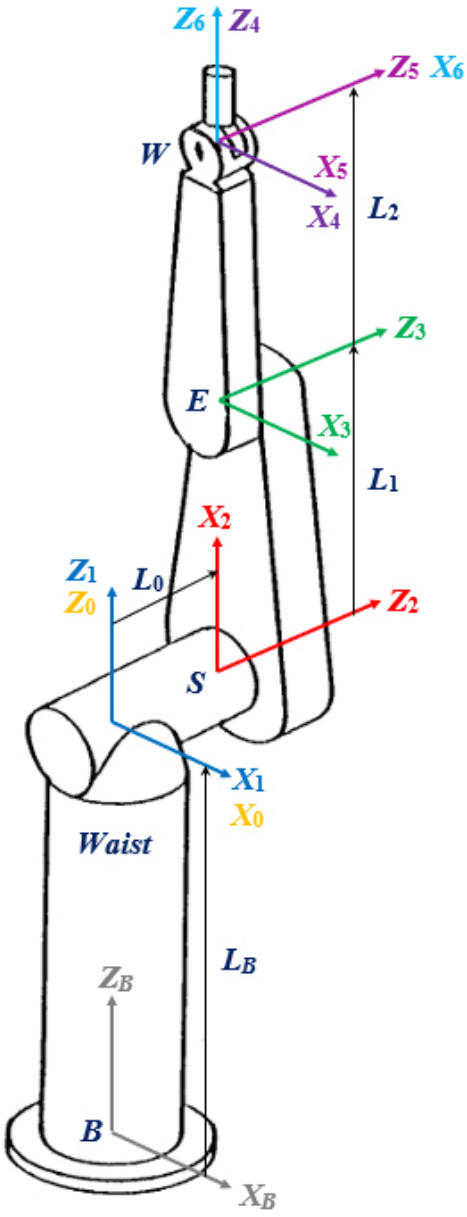
Side view

Top view

i	α_{i-1}	a_{i-1}	d_i	θ_i
1				
2				
3				
4				

7. 6-dof Spatial 6R Unimation PUMA Robot (articulated robot)





i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	θ_1
2	-90°	0	L_0	$\theta_2 - 90^\circ$
3	0	L_1	0	$\theta_3 + 90^\circ$
4	90°	0	L_2	θ_4
5	-90°	0	0	θ_5
6	90°	0	0	$\theta_6 + 90^\circ$

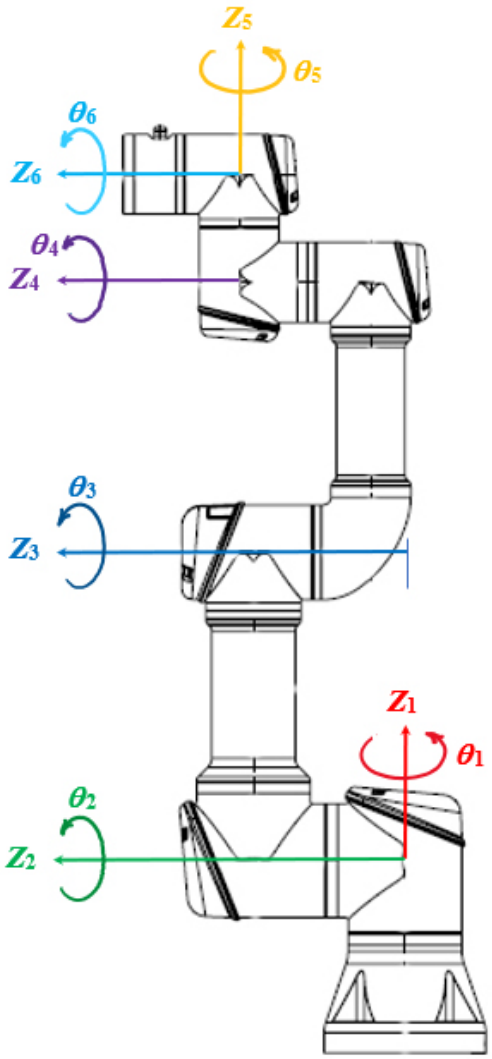
Joint angle offsets

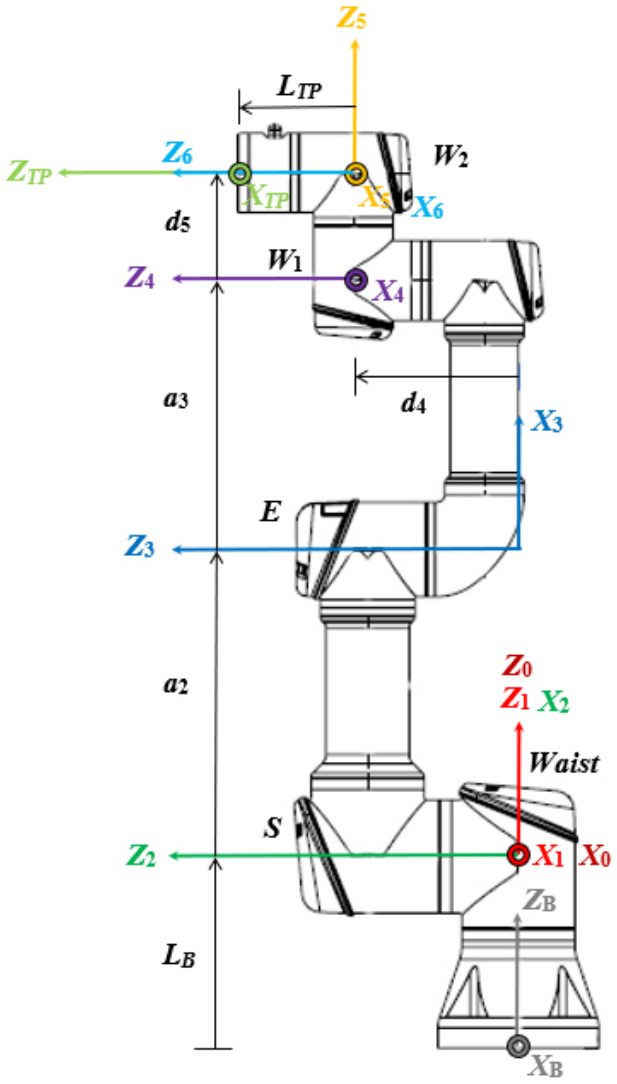
The DH active joint angle variable for a revolute joint is defined as:

θ_i angle between $\hat{X}_{i-1}$ to $\hat{X}_i$ measured about $\hat{Z}_i$

Serial robot kinematic diagrams generally show the zero value definition for each joint angle parameter. In the zero angle configuration, if the $\hat{X}_{i-1}$ and $\hat{X}_i$ axes are NOT identical when $\theta_i = 0$, then a (constant) joint angle offset is required to account for this. Examples of this may be seen in the second, third, and sixth rows for the PUMA Robot above.

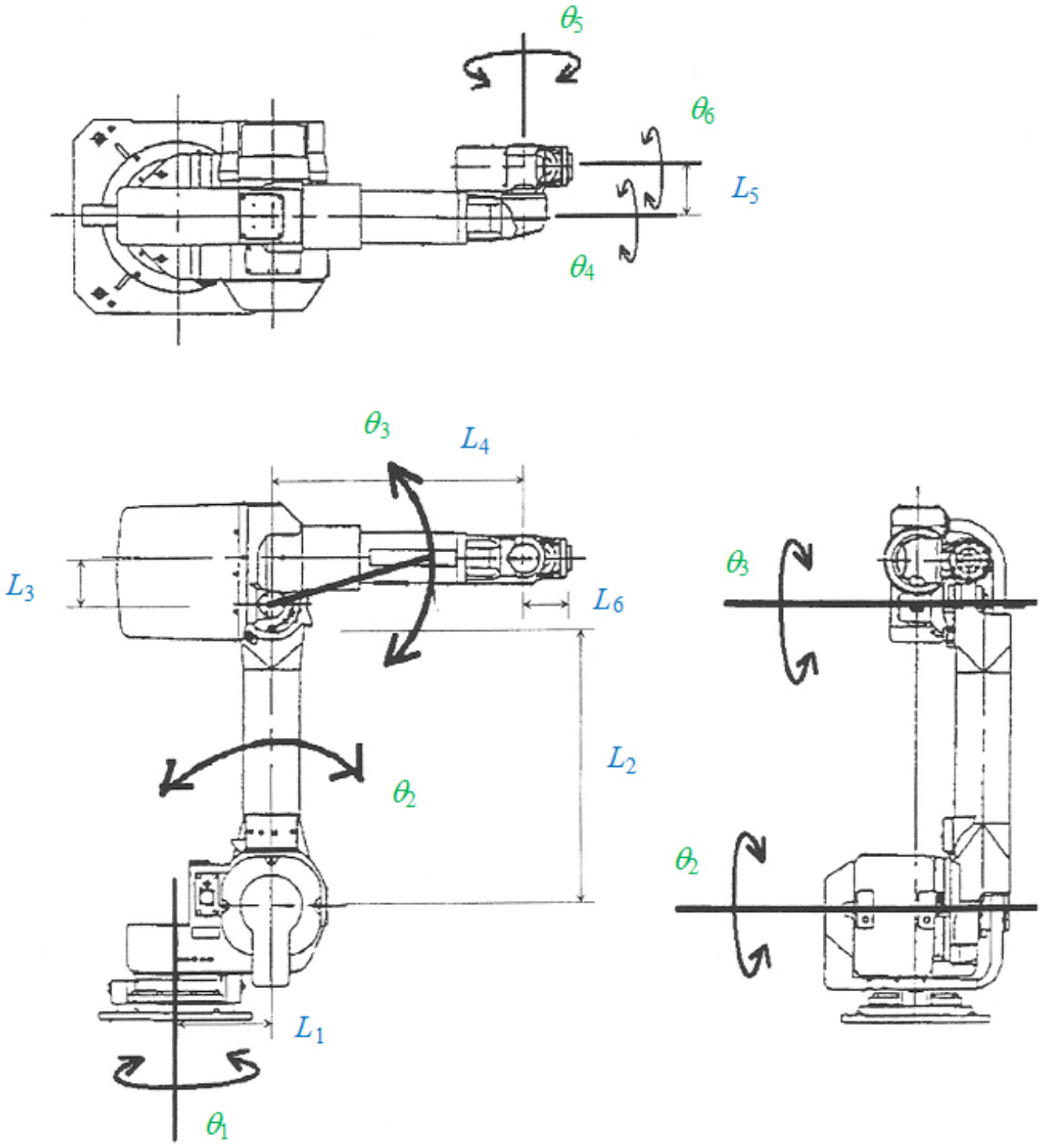
8. 6-dof Spatial 6R Universal UR3e Cobot (Collaborative robot)





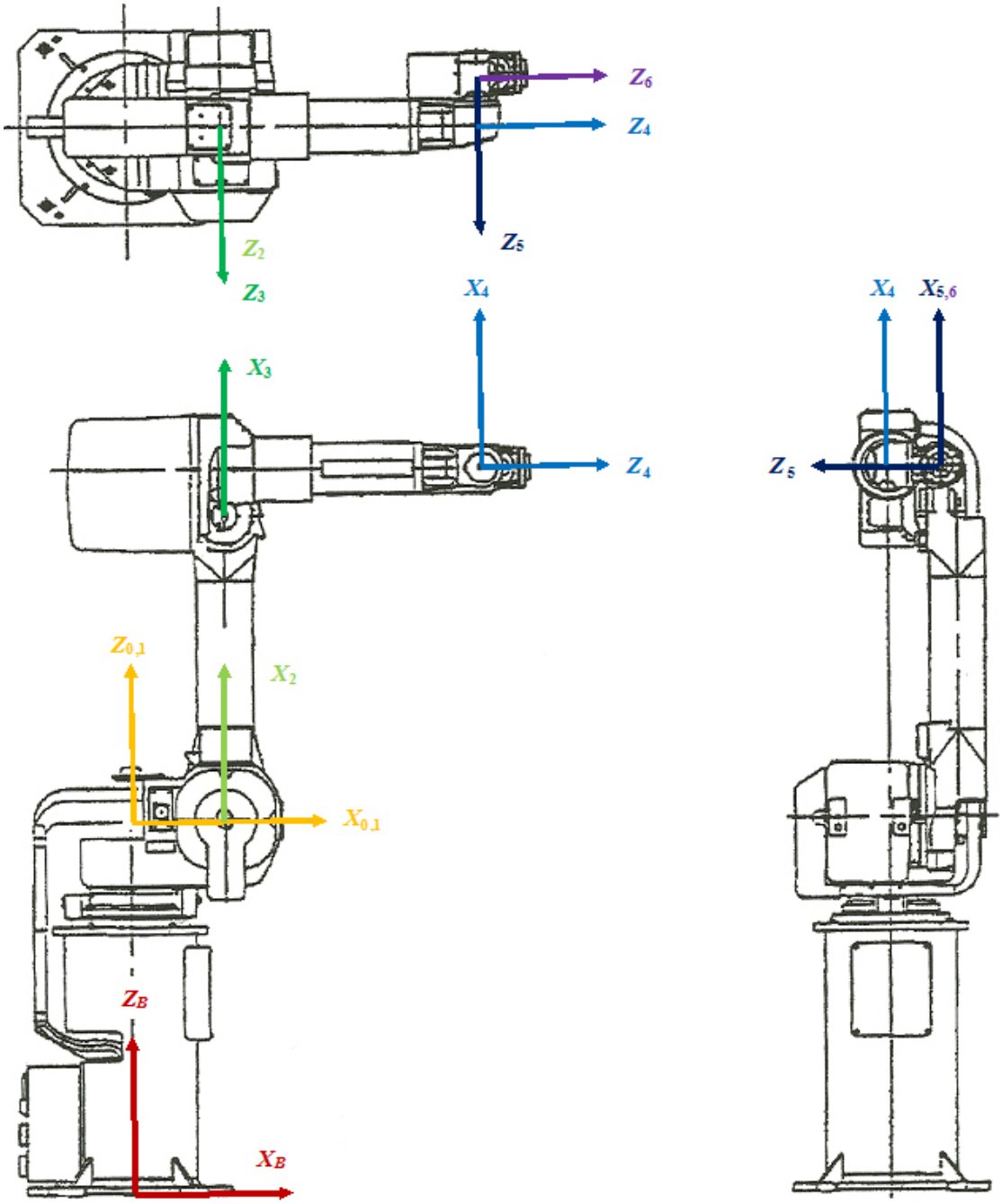
i	α_{i-1}	a_{i-1}	d_i	θ_i
1				
2				
3				
4				
5				
6				

9. 6-dof Spatial 6R Fanuc S10 Robot with offset wrist



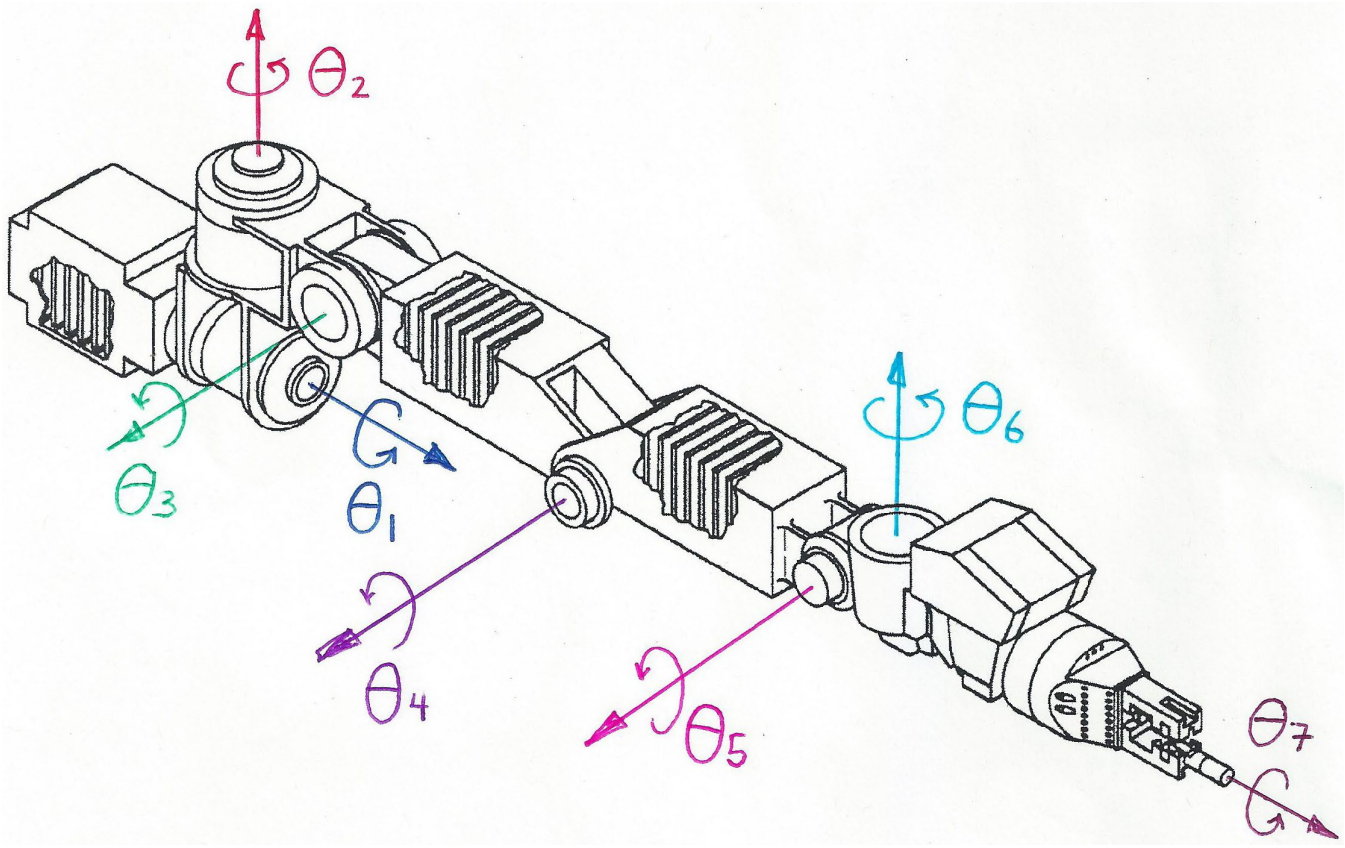
i	1	2	3	4	5	6
L_i (mm)	215	700	100	600	100	25

9. 6-dof 6R Fanuc S10 Robot (continued) – coordinate frames



i	α_{i-1}	a_{i-1}	d_i	θ_i
1				
2				
3				
4				
5				
6				

10. 7-dof Spatial 7R NASA Flight Telerobotic Servicer (FTS) Robot – you define coordinate frames

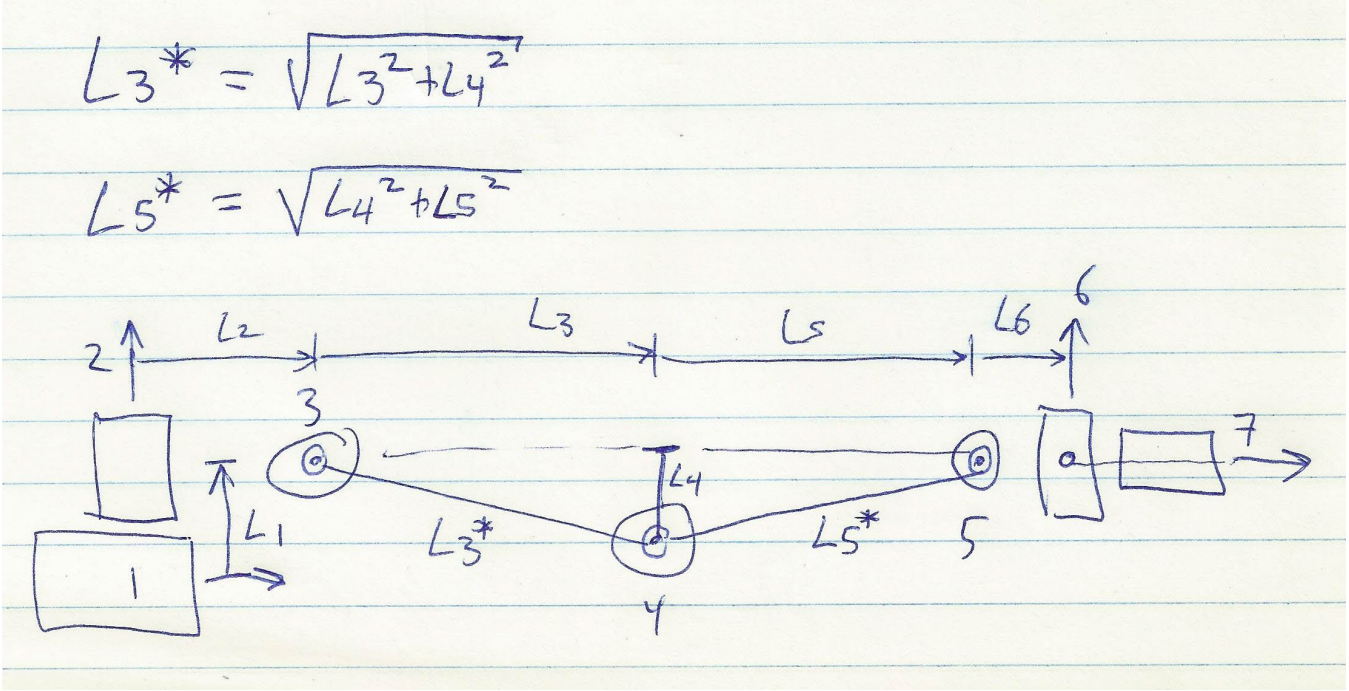


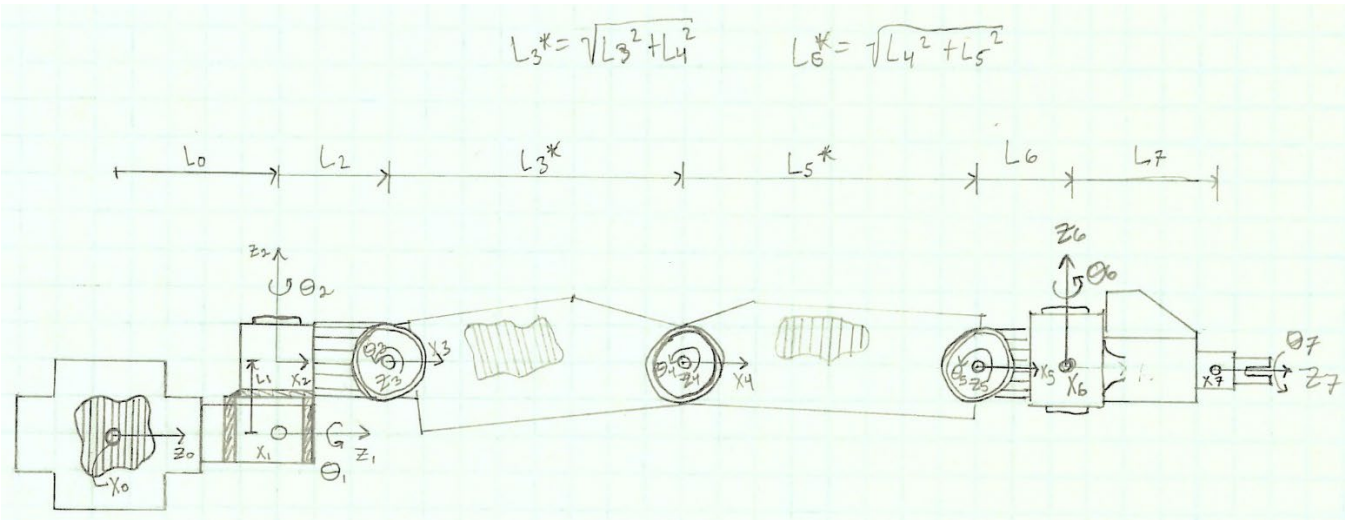
i	α_{i-1}	a_{i-1}	d_i	θ_i
1				
2				
3				
4				
5				
6				
7				

Kinematic lengths / offsets identified in the drawing on the next page.

10. 7-dof Spatial 7R NASA Flight Telerobotic Servicer (FTS) Robot (continued)

HINT: there is no need for joint angle offsets on joints 3, 4, or 5. Instead, the zero angle configuration can be drawn with joint 4 in-line with joints 3 and 5 (unlike the configurations the drawings above and below show). A messy side-view is show below to clarify. This shows the side view for the figure above, not in the configuration I suggest in this hint – this is re-drawn in the second side-view below.



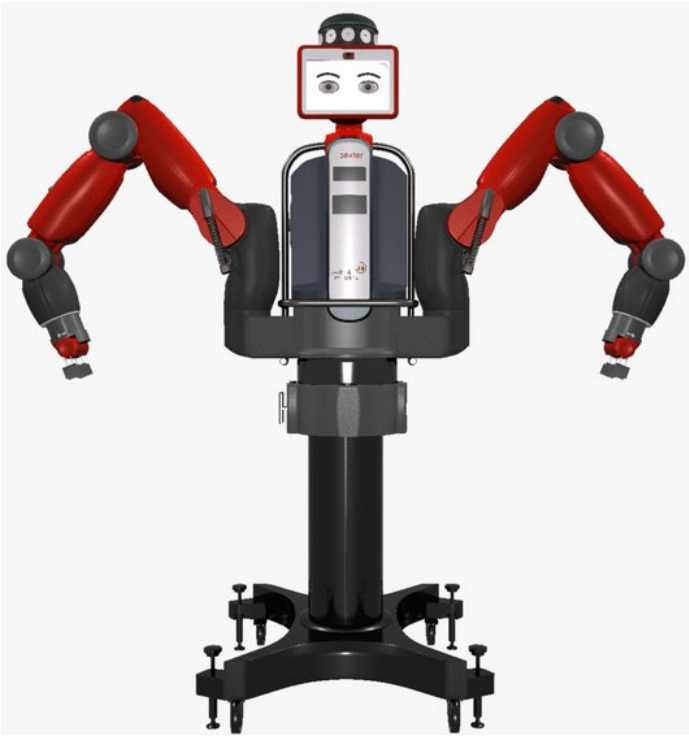


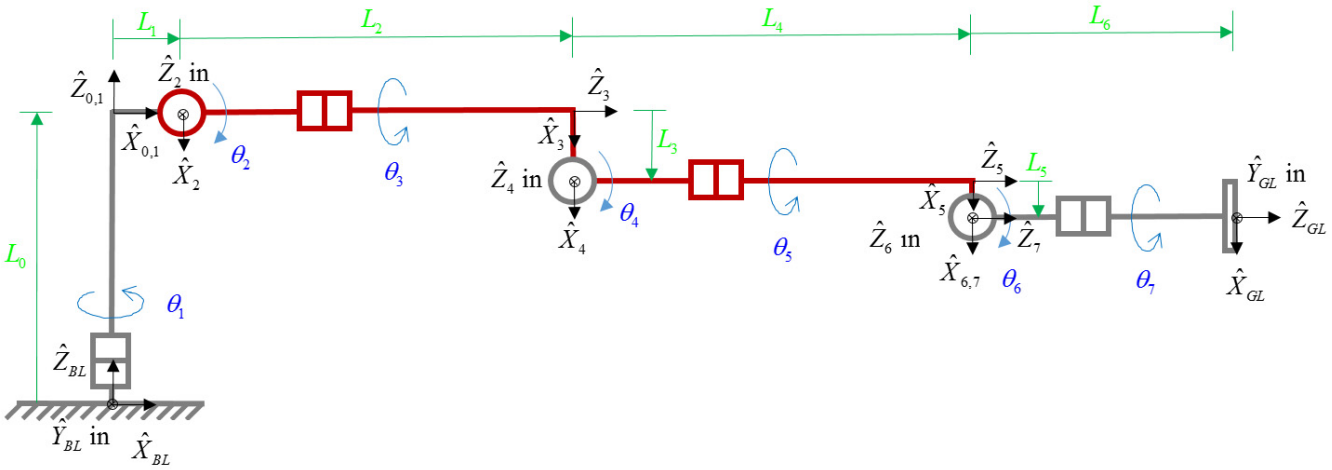
FTS Side View by Collier Fais

nota bene: for simplicity of the FPK equations:

- **make the origin of {0} identical to {1}**
- **make the origin of {7} identical to {6}**

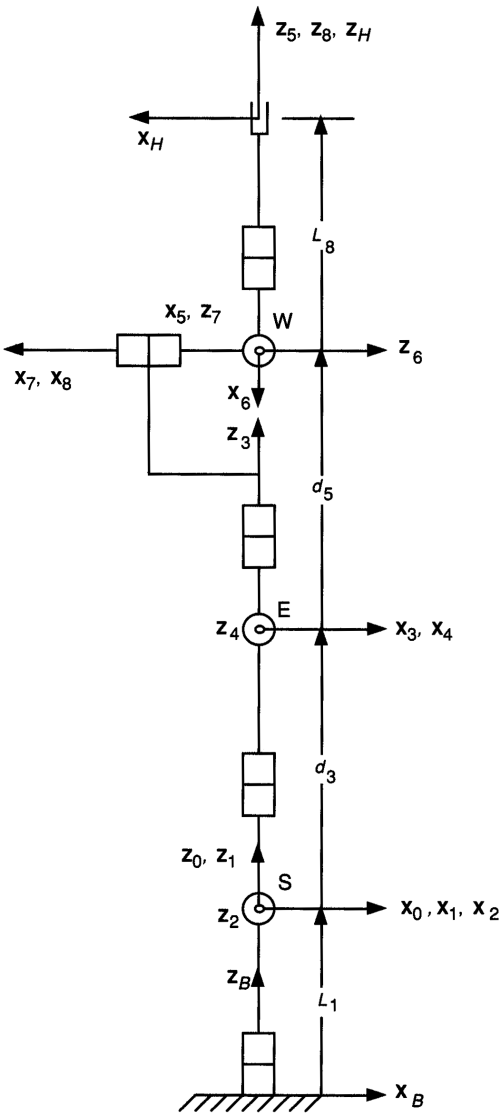
11. 7-dof Baxter Spatial 7R Robot (left arm)

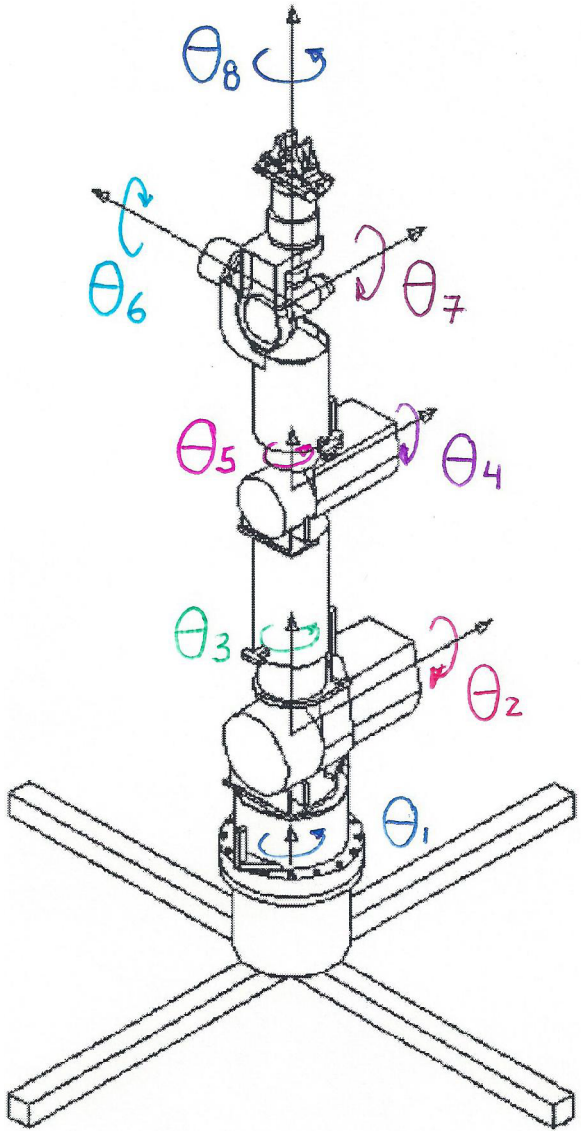




i	α_{i-1}	a_{i-1}	d_i	θ_i
1				
2				
3				
4				
5				
6				
7				

12. 8-dof Spatial 8R NASA AAI ARMII Robot





i	α_{i-1}	a_{i-1}	d_i	θ_i
1				
2				
3				
4				
5				
6				
7				
8				

13. 20-dof Spatial 20R DARwIn-OP Humanoid Walking/Soccer Robot[4]

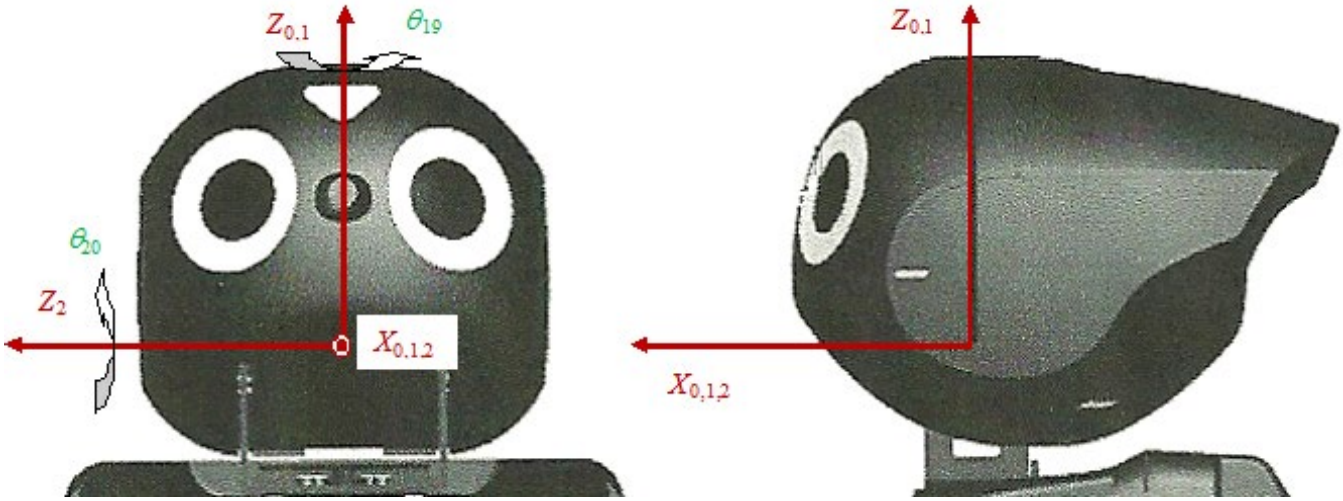
(Robotis, 2011)

Height: 455 mm (about 18 inches)
Mass: 2.8 kg (just over 6 pounds weight)

[4] R.L. Williams II, 2012, "DARwIn-OP Humanoid Robot Kinematics", Proceedings of the ASME IDETC/CIE, Chicago IL, August 12-15, DETC2012-70265.

DARwIn-OP Two-dof 2R Pan/Tilt Head

The Cartesian reference frame definitions for the two-dof pan/tilt head are shown in the figure below.



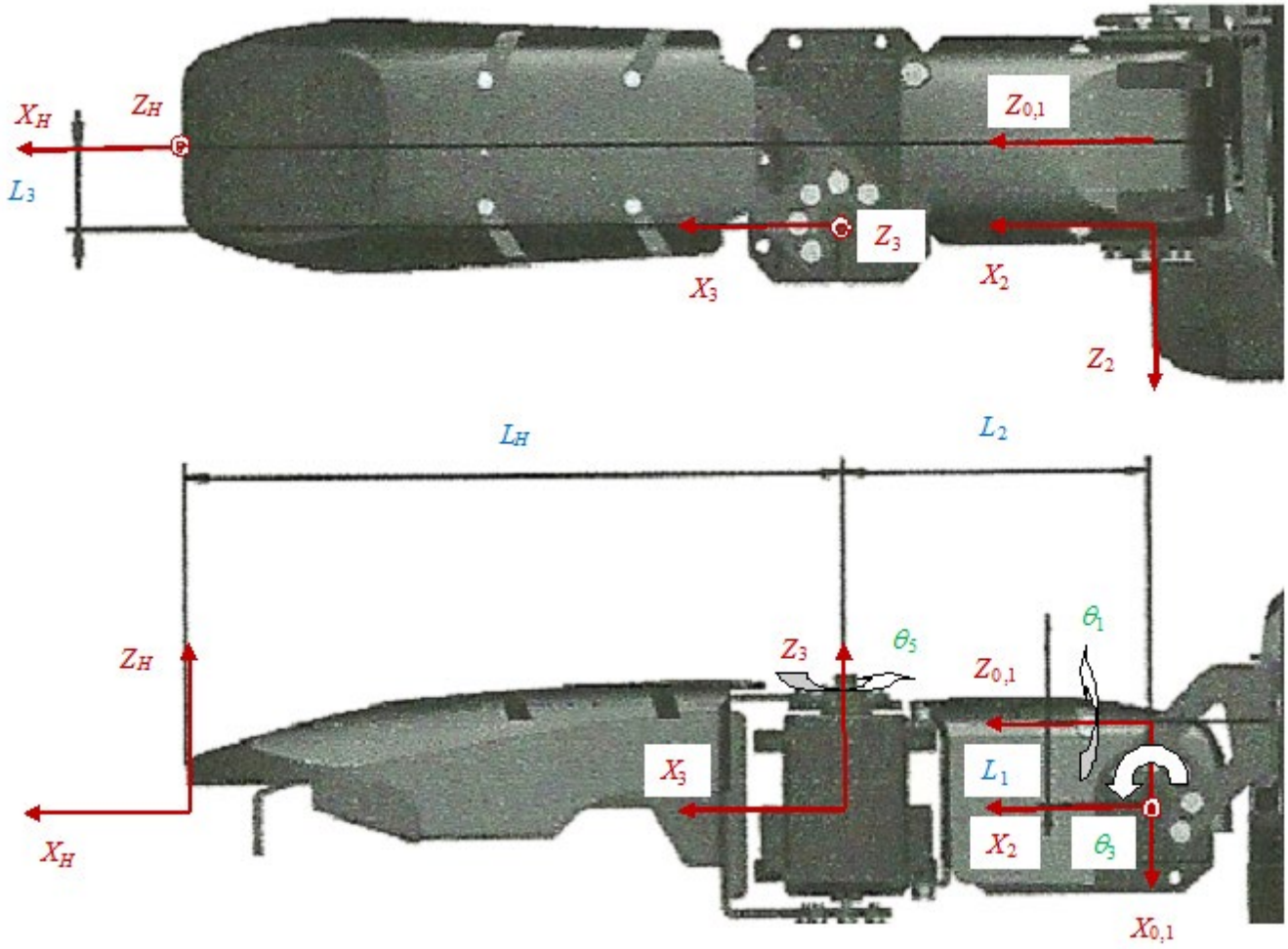
Two-dof Pan/Tilt Head Cartesian Coordinate Frame Definitions

Two-dof Pan/Tilt Head Denavit-Hartenberg Parameters

i	α_{i-1}	a_{i-1}	d_i	θ_i
1				
2				

DARwIn-OP Three-dof 3R Right Arm

The Cartesian reference frame definitions for the three-dof right arm are shown in the figure below.



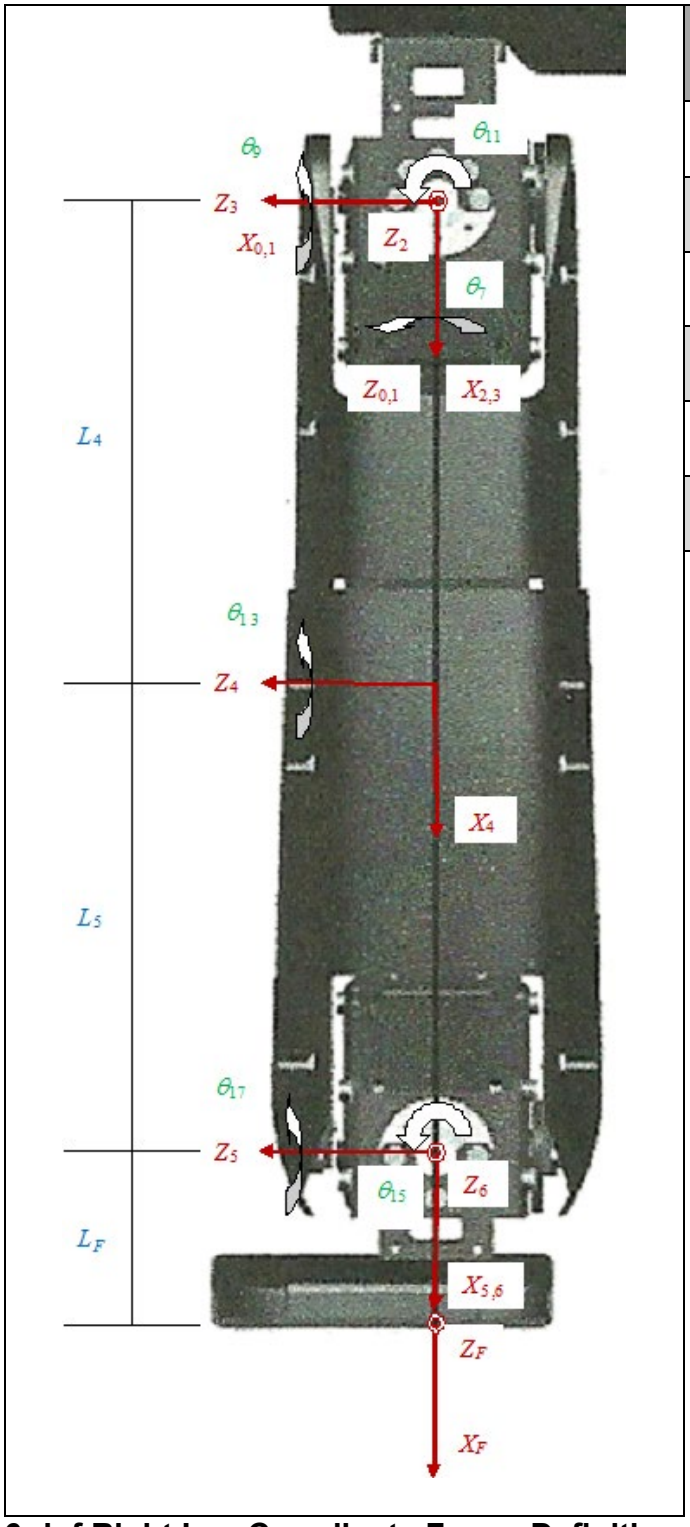
Three-dof Right Arm Cartesian Coordinate Frame Definitions, Top and Front Views

Three-dof Right Arm Denavit-Hartenberg Parameters

i	α_{i-1}	a_{i-1}	d_i	θ_i
1				
2				
3				

DARwIn-OP Six-dof 6R Right Leg

The Cartesian reference frame definitions for the six-dof right leg are shown in the figure below.



6-dof Right Leg Coordinate Frame Definitions

Six-dof Right Leg DH Parameters

i	α_{i-1}	a_{i-1}	d_i	θ_i
1				
2				
3				
4				
5				
6				

4. Forward Pose Kinematics (FPK)

Given the serial robot geometry (DH Parameters) and numerical values for all n 1-dof joint variables (which are part of the DH parameters), find the pose (position and orientation) of the end-effector Cartesian coordinate frame (or other frame of interest) with respect to the base Cartesian coordinate frame.

Given:

Calculate: (4x4 homogeneous transformation matrix)

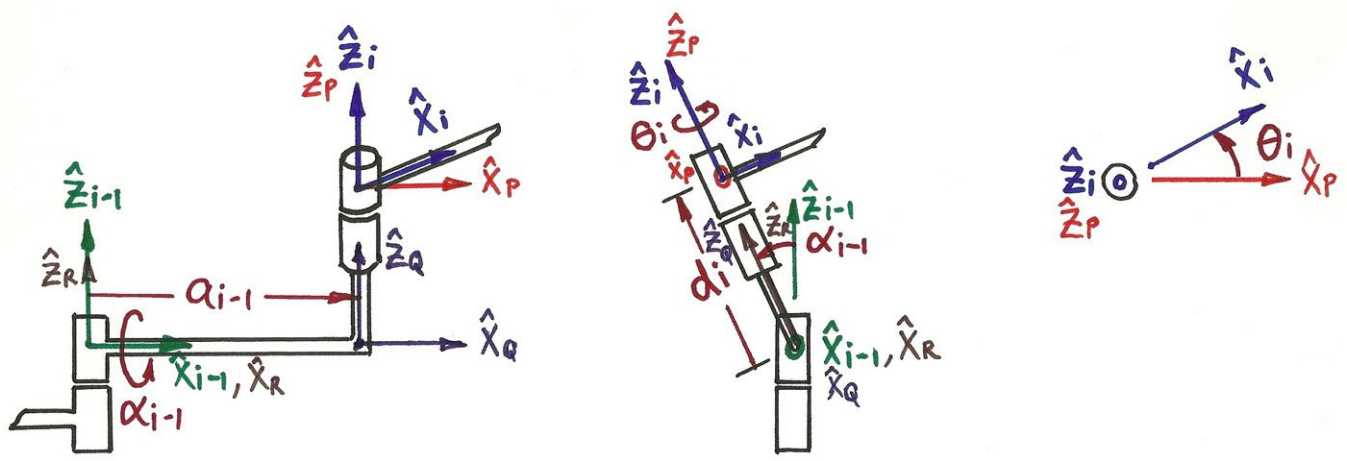
The above assumes all joints are revolute joints with given joint **angles** θ_i. If there are prismatic joints, a joint **length** value d_i must instead be given for those joints (since those θ_i are known and constant).

Non-linear (transcendental) expressions result, but the FPK solution is straight-forward since all joint angles inside the *sines* and *cosines* are given. We will use matrix multiplication of consecutive link transformation matrices to find the FPK result. Results can be evaluated numerically or symbolically. The preferred result is symbolic expressions of the Forward Pose Kinematics result that can be implemented numerically in MATLAB.

Forward Pose Kinematics is useful for robot simulation and sensor-based control. The problem is made tractable by isolating the problem to the pose of one frame with respect to its previous neighbor along the serial chain (use one row of the DH parameters table to determine this pose transform matrix). Then repeat this for all moving links/joints with respect to their previous neighbor and multiply all pose transform matrices together in the correct order.

4.1 Derivation of Consecutive Link Transformations

Determine the pose of frame $\{i\}$ with respect to frame $\{i-1\}$, represented by the 4x4 matrix $\left[{}^{i-1}_{i}T\right]$. For convenience, attach 3 intermediate frames $\{R\}$, $\{Q\}$, and $\{P\}$ to the DH Parameters figure.



DH Parameters Figure with Intermediate Frames {*R*}, {*Q*}, {*P*}

To move frame $\{i-1\}$ into frame $\{i\}$, the following 4 steps are made.

rotate	α_{i-1}	from $\hat{Z}_{i-1}$ to $\hat{Z}_R$	about $\hat{X}_{i-1}$
translate	a_{i-1}	from $\hat{Z}_R$ to $\hat{Z}_Q$	along $\hat{X}_R$
translate	d_i	from $\hat{X}_Q$ to $\hat{X}_P$	along $\hat{Z}_Q$
rotate	θ_i	from $\hat{X}_P$ to $\hat{X}_i$	about $\hat{Z}_P$

These steps are four uses of Interpretation #3 of the homogeneous transformation matrix (transform operator). Interpretation #1 could be used instead (pose of frame $\{i\}$ with respect to frame $\{i-1\}$), reversing the above steps and reversing the direction of motion for each DH parameter. The equation below is identical in either case.

$$\left[{}^{i-1}_{i}T\right] = \left[{}^{i-1}_{R}T(\alpha_{i-1})\right]\left[{}^{R}_{Q}T(a_{i-1})\right]\left[{}^{Q}_{P}T(d_i)\right]\left[{}^{P}_{i}T(\theta_i)\right] = \left[R_X(\alpha_{i-1})\right]\left[D_X(a_{i-1})\right]\left[D_Z(d_i)\right]\left[R_Z(\theta_i)\right]$$

$$\left[{}^{i-1}_{i}T\right] = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -d_i s\alpha_{i-1} \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & d_i c\alpha_{i-1} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Physical interpretation: $\left[{}^{i-1}_{i}R\right], \left\{{}^{i-1}P_i\right\}$

Screw Matrices:

$$\left[{}^{i-1}_{i}T\right] = \left[\text{Screw}_X(a_{i-1},\alpha_{i-1})\right]\left[\text{Screw}_Z(d_i,\theta_i)\right]$$

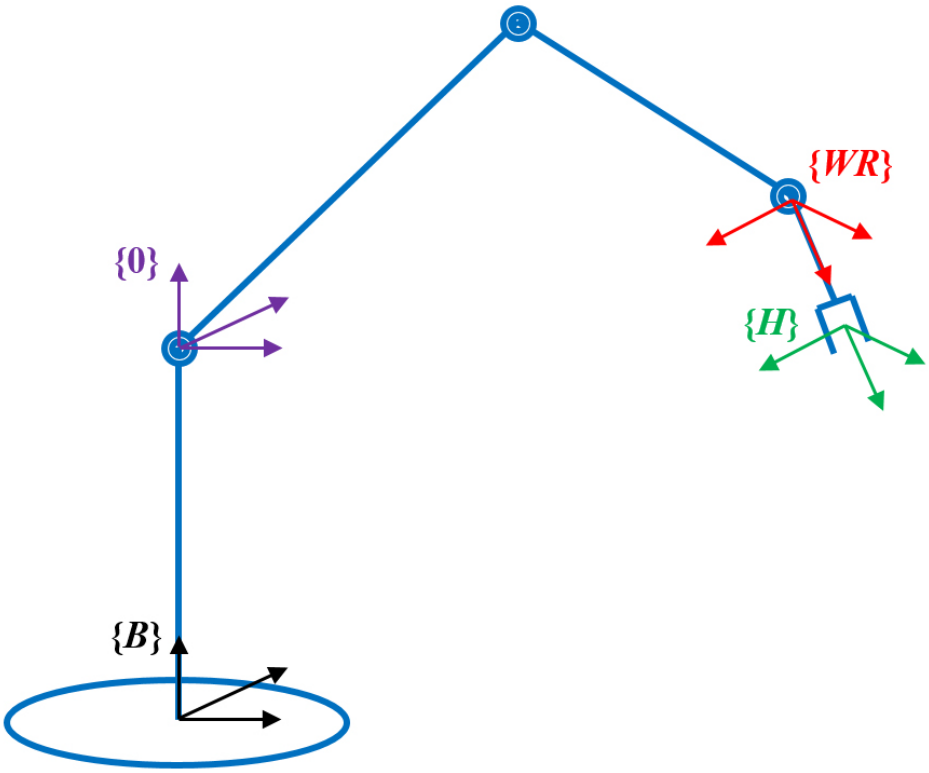
$$\left[\text{Screw}_X(a_{i-1},\alpha_{i-1})\right] = \left[R_X(\alpha_{i-1})\right]\left[D_X(a_{i-1})\right] = \left[D_X(a_{i-1})\right]\left[R_X(\alpha_{i-1})\right] = \begin{bmatrix} 1 & 0 & 0 & a_{i-1} \\ 0 & c\alpha_{i-1} & -s\alpha_{i-1} & 0 \\ 0 & s\alpha_{i-1} & c\alpha_{i-1} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\left[\text{Screw}_Z(d_i,\theta_i)\right] = \left[R_Z(\theta_i)\right]\left[D_Z(d_i)\right] = \left[D_Z(d_i)\right]\left[R_Z(\theta_i)\right] = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & 0 \\ s\theta_i & c\theta_i & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Caution screws commute (the order of translation and rotation doesn't matter), but matrix multiplication **DOES NOT COMMUTE** in general. Also, you cannot commute the order of the X and Z screws, they must appear as given in the equations on this page.

4.2 General FPK Solution for Serial Robots

Figure for PUMA-like serial robot – now we need the DH frames for each active joint/link:



Here is the generic serial robot Forward Pose Kinematics Solution:

Here is the specific Forward Pose Kinematics Solution for the six-dof 6R PUMA serial robot:

The PUMA Forward Pose Kinematics Solution can be extended to include the {B} and {H} frames:

Note: The MATLAB Symbolic Toolbox is particularly useful to generate and simplify the FPK solution. See Dr. Bob's on-line MATLAB Primer for a sample symbolic math m-file:
people.ohio.edu/williams/html/PDF/MATLABPrimer.pdf

4.3 FPK Examples

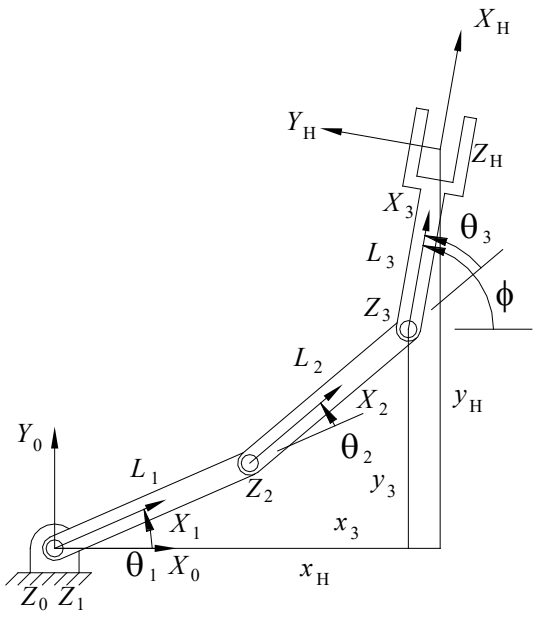
4.3.1 Planar 3R Serial Robot FPK Solution

Forward Pose Kinematics Symbolic Derivations

Given the robot (the 9 constant DH Parameters) and $\theta_1, \theta_2, \theta_3$, Calculate $\left[{}^{0}_{3}T\right]$ and $\left[{}^{0}_{H}T\right]$

.

Note $\left[{}^{0}_{3}T\right]$ is represented by $\{x_3 \quad y_3 \quad \phi\}^T$ and $\left[{}^{0}_{H}T\right]$ by $\{x_H \quad y_H \quad \phi\}^T$.



i	α_{i-1}	a_{i-1}	d_i	θ_i		
1	0	0	0	θ_1	$\Rightarrow$	$\left[{}^{0}_{1}T\right]$
2	0	L_1	0	θ_2	$\Rightarrow$	$\left[{}^{1}_{2}T\right]$
3	0	L_2	0	θ_3	$\Rightarrow$	$\left[{}^{2}_{3}T\right]$

Substitute each row of the DH parameters table into the equation for $\left[{}^{i-1}_{i}T\right]$.

$$\left[{}^{i-1}_{i}T\right] = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -d_i s\alpha_{i-1} \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & d_i c\alpha_{i-1} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\left[{}^{0}_{1}T(\theta_1)\right]=\begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \left[{}^{1}_{2}T(\theta_2)\right]=\begin{bmatrix} c_2 & -s_2 & 0 & L_1 \\ s_2 & c_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \left[{}^{2}_{3}T(\theta_3)\right]=\begin{bmatrix} c_3 & -s_3 & 0 & L_2 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\left[{}^{0}_{3}T\right]=\left[{}^{0}_{1}T(\theta_1)\right]\left[{}^{1}_{2}T(\theta_2)\right]\left[{}^{2}_{3}T(\theta_3)\right]=\begin{bmatrix} c_{123} & -s_{123} & 0 & L_1c_1+L_2c_{12} \\ s_{123} & c_{123} & 0 & L_1s_1+L_2s_{12} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(interpret geometrically)

In this planar robot there is a significant simplification of the rotation matrix terms by using the trigonometric sum-of-angles formulas.

$$\cos(a \pm b) = cacb \mp sasb$$
$$\sin(a \pm b) = sacb \pm casb$$

This is possible since all 3 active *Z* axes are always parallel. An example for the [1,1] term of $\left[{}^{0}_{3}T\right]$ is given below.

Original multiplication: $(c_1c_2 - s_1s_2)c_3 - (c_1s_2 + s_1c_2)s_3$

First simplification: $c_{12}c_3 - s_{12}s_3$

Second simplification: c_{123}

$$c_{12} = \cos(\theta_1 + \theta_2) \qquad c_{123} = \cos(\theta_1 + \theta_2 + \theta_3)$$
$$s_{12} = \sin(\theta_1 + \theta_2) \qquad s_{123} = \sin(\theta_1 + \theta_2 + \theta_3)$$

What happened to L_3? It appears in an additional, fixed transform.

$$\left[{}^{3}_{H}T\right]=\begin{bmatrix} 1 & 0 & 0 & L_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(by inspection from the planar 3R robot figure)

For this robot we need to control the {*H*} frame instead of the {3} frame. The overall Forward Pose Kinematics Homogeneous Transformation Matrix is given below. L_3 is a constant. $\left[{}^{3}_{H}T\right]$ is found by inspection, not from the DH Parameter process, since there is no variable involved.

$$\left[{}^{0}_{H}T\right]=\left[{}^{0}_{3}T(\theta_1,\theta_2,\theta_3)\right]\left[{}^{3}_{H}T(L_3)\right]$$

$$\left[{}^{0}_{H}T\right]=\begin{bmatrix} c_{123} & -s_{123} & 0 & L_1c_1+L_2c_{12} \\ s_{123} & c_{123} & 0 & L_1s_1+L_2s_{12} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & L_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}=\begin{bmatrix} c_{123} & -s_{123} & 0 & L_1c_1+L_2c_{12}+L_3c_{123} \\ s_{123} & c_{123} & 0 & L_1s_1+L_2s_{12}+L_3s_{123} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The position vector of $\left[{}^{0}_{H}T\right]$ is significantly more complicated than that of $\left[{}^{0}_{3}T\right]$; the orientation (rotation matrix) of $\left[{}^{0}_{H}T\right]$ is identical to that of $\left[{}^{0}_{3}T\right]$.

Planar 3R Robot FPK Snapshot Examples Given fixed robot lengths $L_1=3, L_2=2, L_3=1$ (m).

1) Given $\theta_1=15^\circ, \theta_2=25^\circ, \theta_3=35^\circ$, calculate $\left[{}^{0}_{H}T\right]$.

$$\left[{}^{0}_{H}T\right]=\left[{}^{0}_{3}T(\theta_1,\theta_2,\theta_3)\right]\left[{}^{3}_{H}T(L_3)\right]$$

$$\left[{}^{0}_{3}T\right]=\begin{bmatrix} 0.26 & -0.97 & 0 & 4.43 \\ 0.97 & 0.26 & 0 & 2.06 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \left\{{}^{0}X_3\right\}=\left\{x_3 \quad y_3 \quad \phi\right\}^T=\left\{4.43 \quad 2.06 \quad 75^\circ\right\}^T$$

$$\left[{}^{3}_{H}T\right]=\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

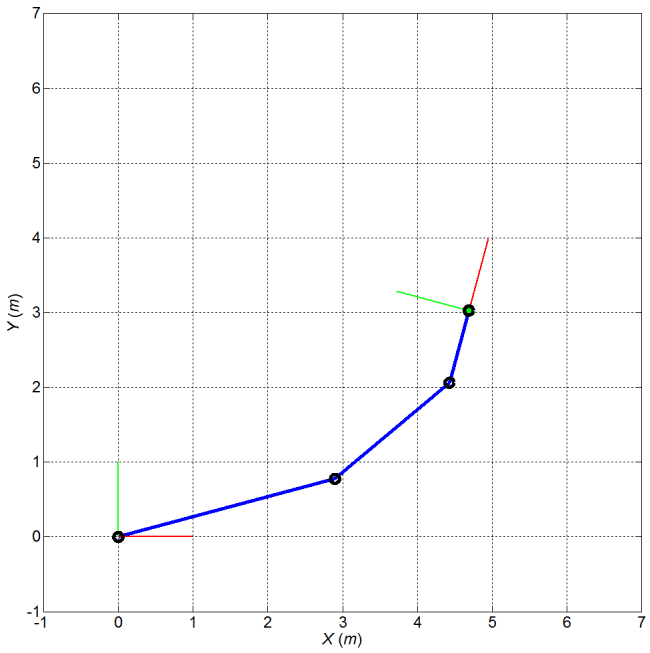
$$\left[{}^{0}_{H}T\right]=\begin{bmatrix} 0.26 & -0.97 & 0 & 4.69 \\ 0.97 & 0.26 & 0 & 3.03 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \left\{{}^{0}X_H\right\}=\left\{x_H \quad y_H \quad \phi\right\}^T=\left\{4.69 \quad 3.03 \quad 75^\circ\right\}^T$$

2) Given $\theta_1=90^\circ, \theta_2=\theta_3=0^\circ$, calculate $\left[{}^{0}_{H}T\right]$. Since $\left[{}^{3}_{H}T\right]$ is constant, it is unchanged.

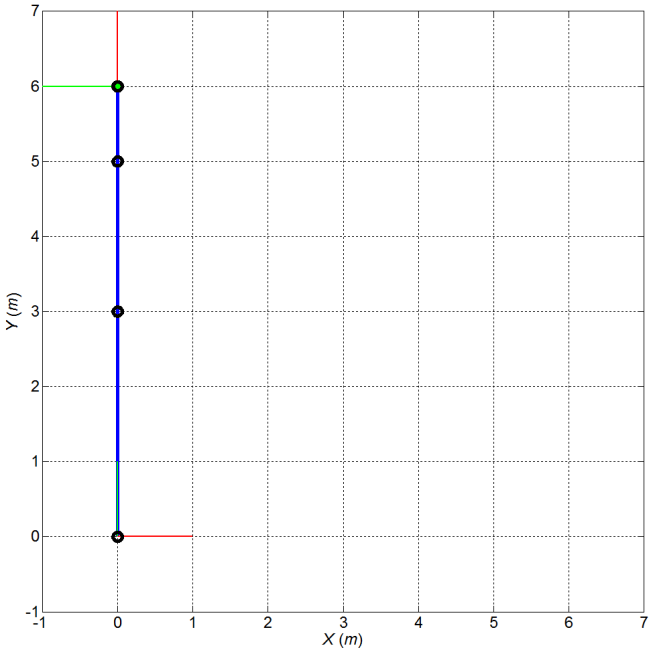
$$\left[{}^{0}_{3}T\right]=\begin{bmatrix}0 & -1 & 0 & 0\\ 1 & 0 & 0 & 5\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \quad \left\{{}^{0}X_{3}\right\}=\left\{x_3 \quad y_3 \quad \phi\right\}^{T}=\left\{0 \quad 5 \quad 90^{\circ}\right\}^{T}$$

$$\left[{}^{0}_{H}T\right]=\begin{bmatrix}0 & -1 & 0 & 0\\ 1 & 0 & 0 & 6\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \quad \left\{{}^{0}X_{H}\right\}=\left\{x_H \quad y_H \quad \phi\right\}^{T}=\left\{0 \quad 6 \quad 90^{\circ}\right\}^{T}$$

Check both planar 3R FPK results using a sketch (MATLAB graphics below). Be sure to include the $\{H\}$ and $\{0\}$ axes to check the orientation $\left[{}^{0}_{H}R\right]$ in addition to the position vector $\left\{{}^{0}P_{H}\right\}$.



Planar 3R FPK Example 1



Planar 3R FPK Example 2

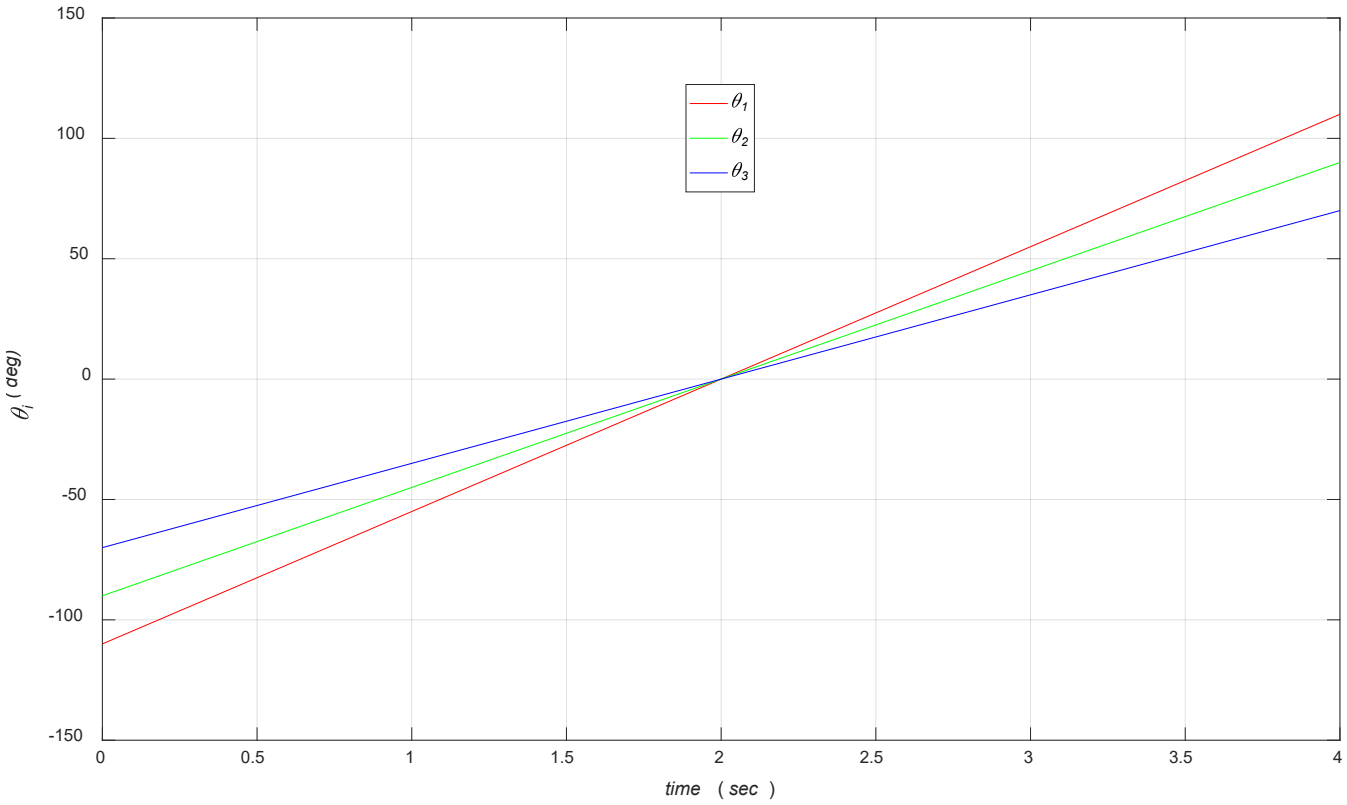
(*X* red, *Y* green, *Z* out of page)

Planar 3R Robot FPK Trajectory Example

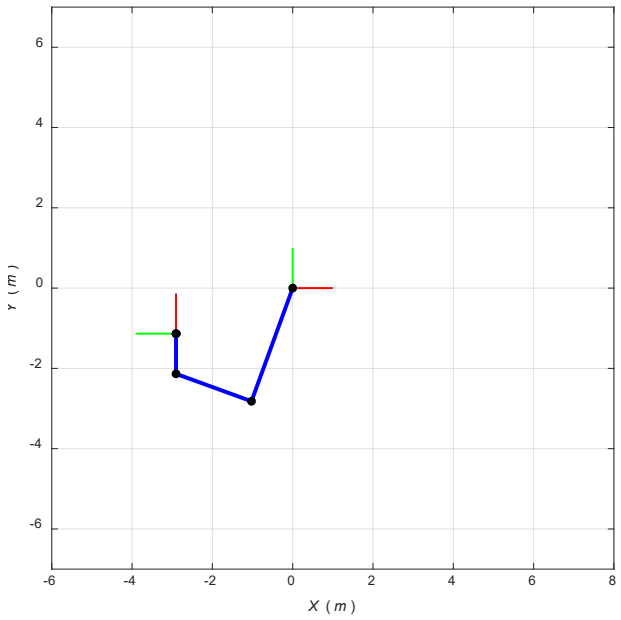
We simulate FPK for the entire range of motion from the minimum to maximum angle on each joint simultaneously. For this Planar 3R Robot FPK simulation we assume the following joint angle limits. Then the commanded motion is to move each of the joints simultaneously from min to max joint limits in equal steps (different size for each joint).

Angle	Min	Max
θ_1	-110°	110°
θ_2	-90°	90°
θ_3	-70°	70°

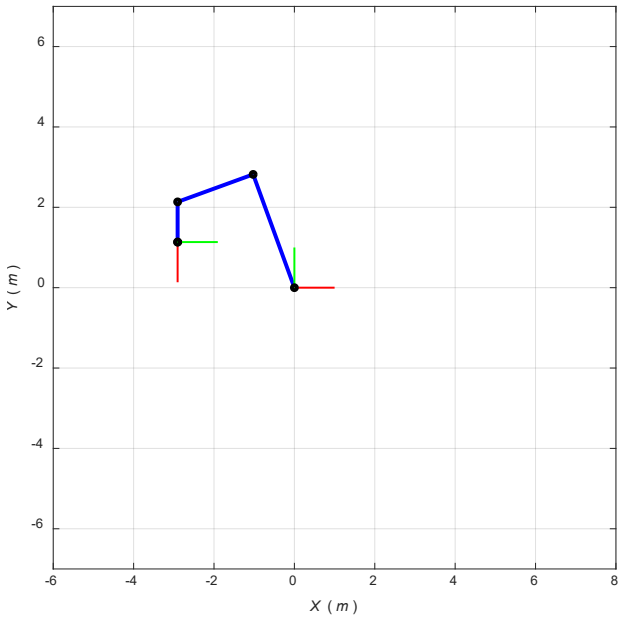
Again, given fixed robot lengths $L_1 = 3, L_2 = 2, L_3 = 1$ (m). The results of this Planar 3R Robot FPK trajectory motion simulation are given in the following plots. This motion is not very useful, i.e. it is just demonstrating some steady robot joint motions. Inverse Pose Kinematics (IPK, Chapter 5) is more useful for practical robot control.



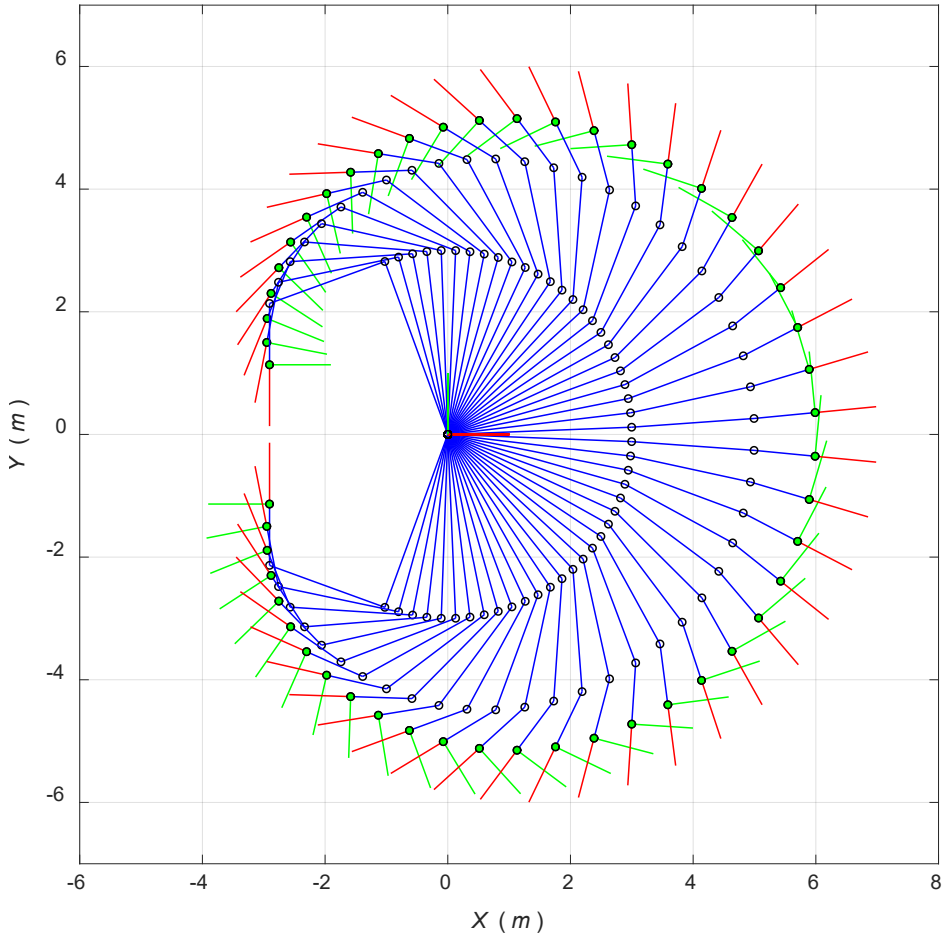
FPK Input Joint Angles



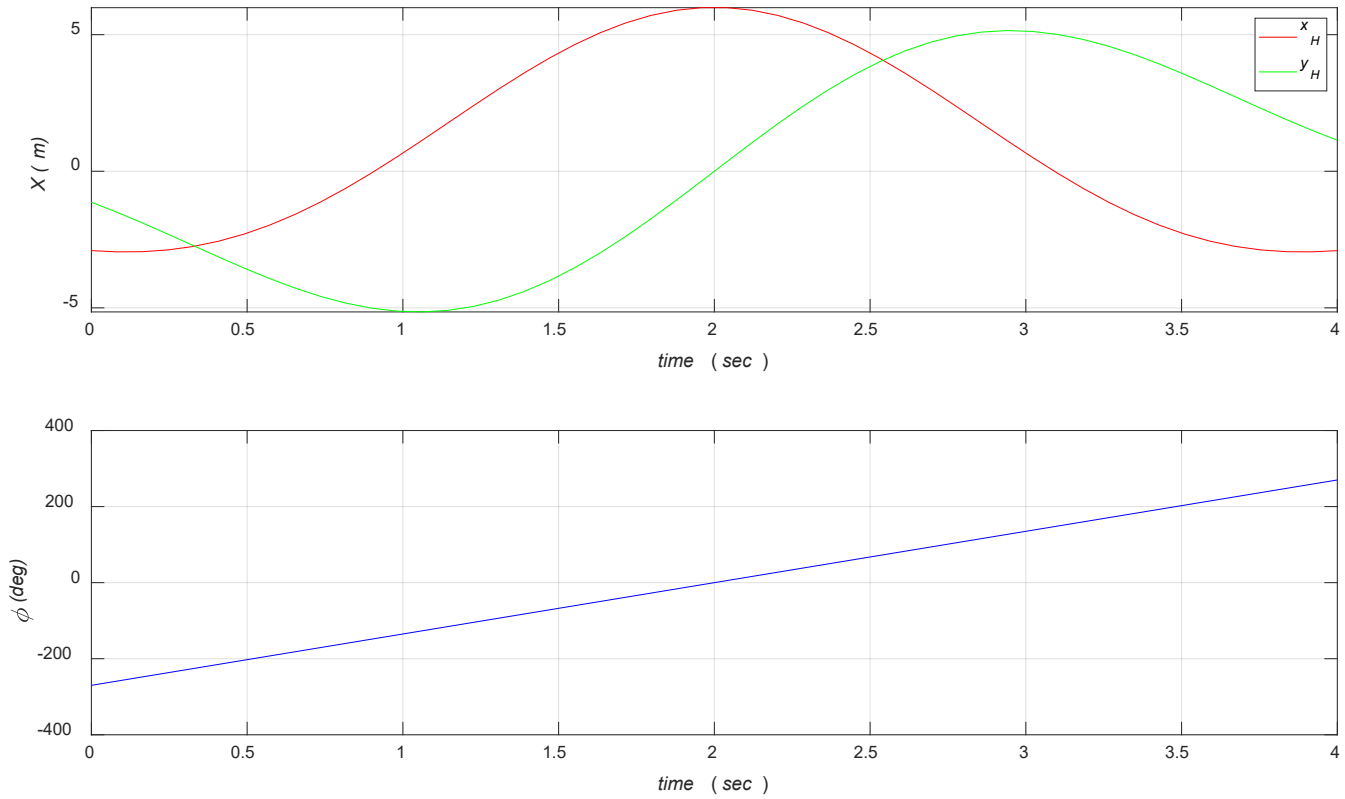
Initial Pose



Final Pose



Spirograph Image of all Poses



FPK Output Cartesian Values

4.3.2 Spatial 4-dof SCARA Serial Robot FPK Solution by Inspection

This section presents validation of the SCARA FPK solution by inspection, in detail.

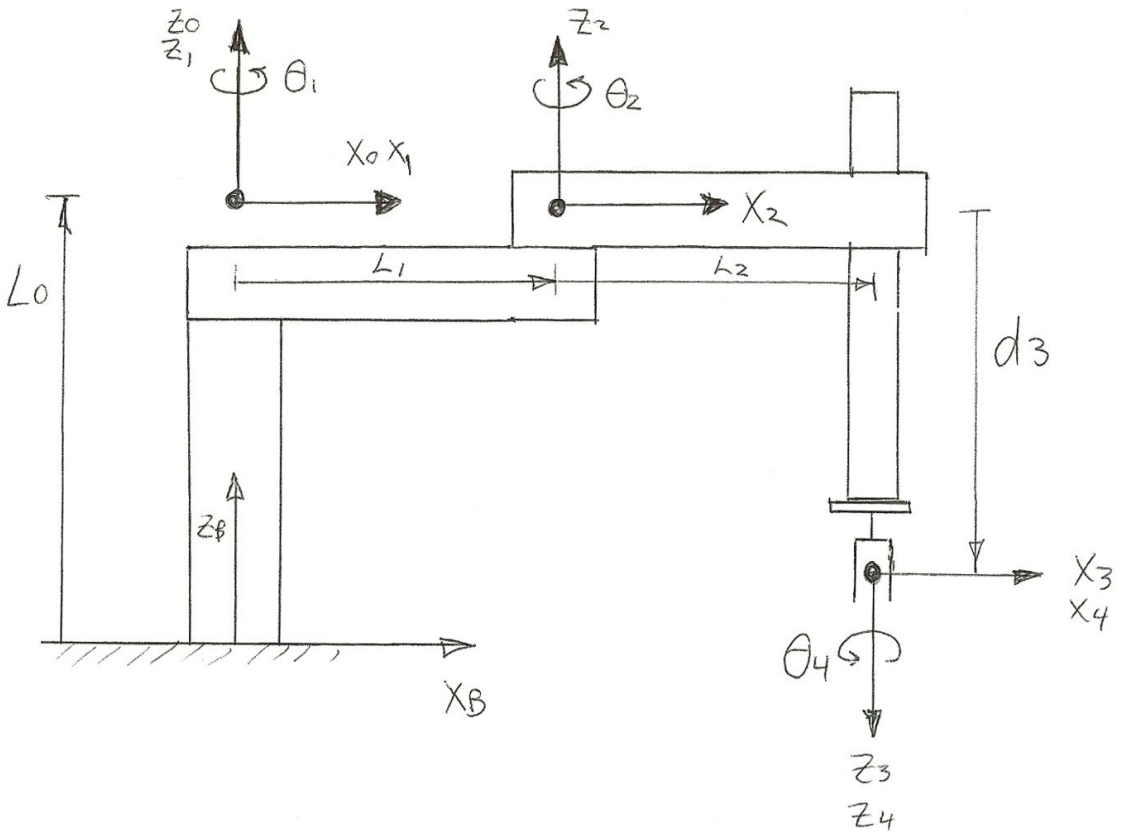
Example: Given $\begin{bmatrix}\theta_1 & \theta_2 & d_3 & \theta_4\end{bmatrix} = \begin{bmatrix}-90^\circ & -90^\circ & 0.15 & 90^\circ\end{bmatrix}$ (deg and m)

$L_0 = 0.552$, $L_1 = 0.300$, $L_2 = 0.250$ (m)

Numerical FPK solution via MATLAB:

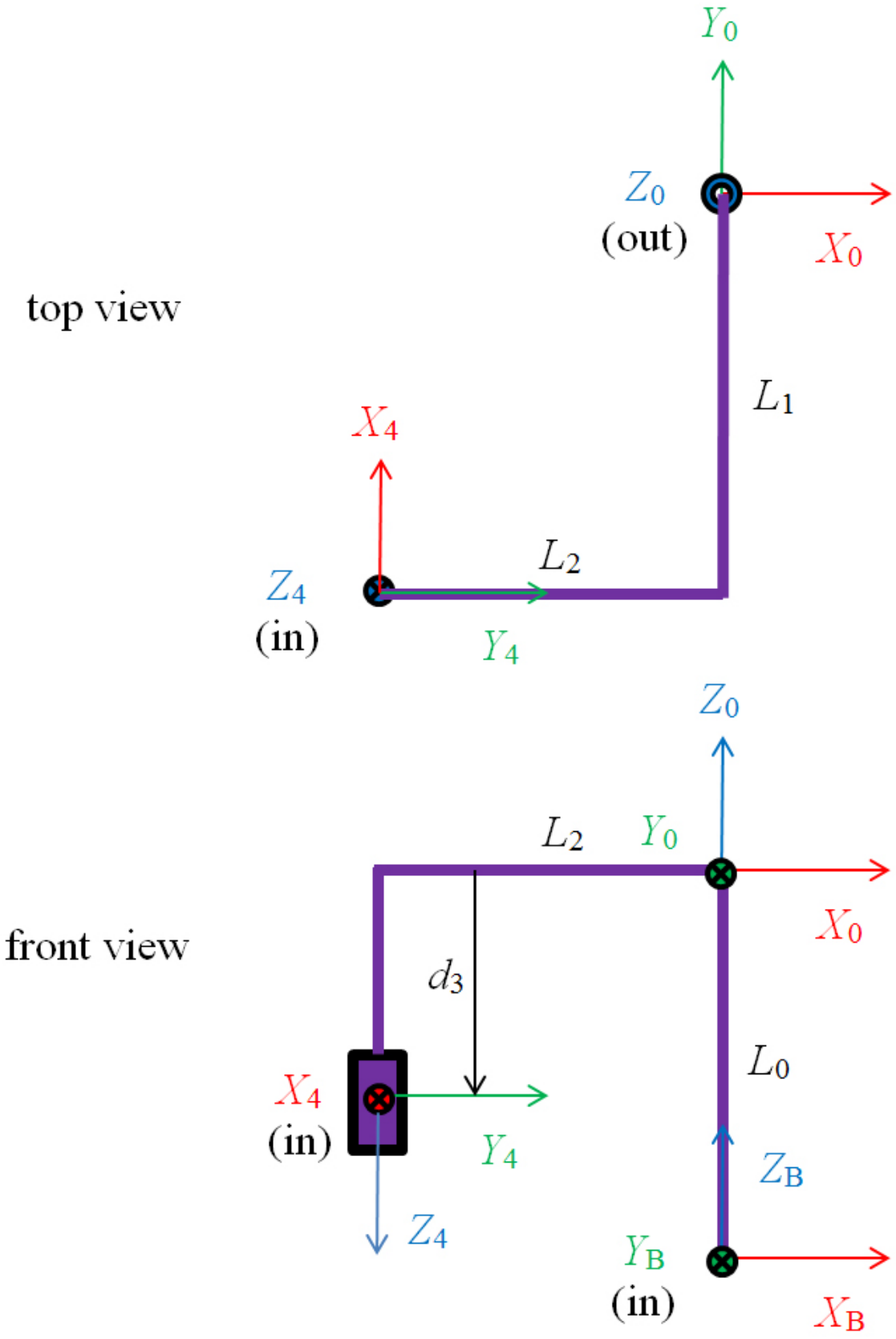
```
T40 =    0.0000    1.0000         0   -0.2500
         1.0000    0.0000         0   -0.3000
              0         0   -1.0000   -0.1500
              0         0         0    1.0000

T4B =    0.0000    1.0000         0   -0.2500
         1.0000    0.0000         0   -0.3000
              0         0   -1.0000    0.4020
              0         0         0    1.0000
```

See the next pages for SCARA FPK validation sketches.



Adept 550 SCARA Serial Robot Kinematic Diagram

SCARA Robot Top and Front View Sketches for validation in this example:



SCARA Robot FPK Solution Validation by Inspection

First we need to recall how the 4x4 homogeneous transformation matrix is composed:

$$\left[{}^{0}_{4}T\right]=\begin{bmatrix} & \left[{}^{0}_{4}R\right] & & \left\{{}^{0}P_{4}\right\} \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \left[{}^{B}_{4}T\right]=\begin{bmatrix} & \left[{}^{B}_{4}R\right] & & \left\{{}^{B}P_{4}\right\} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Further recall that the 3x3 orthonormal rotation matrix is composed column-wise as follows:

$$\left[{}^{0}_{4}R\right]=\begin{bmatrix} | & | & | \\ \left\{{}^{0}\hat{X}_{4}\right\} & \left\{{}^{0}\hat{Y}_{4}\right\} & \left\{{}^{0}\hat{Z}_{4}\right\} \\ | & | & | \end{bmatrix} \qquad \left[{}^{B}_{4}R\right]=\begin{bmatrix} | & | & | \\ \left\{{}^{0}\hat{X}_{4}\right\} & \left\{{}^{0}\hat{Y}_{4}\right\} & \left\{{}^{0}\hat{Z}_{4}\right\} \\ | & | & | \end{bmatrix}$$

Now, how does $\hat{X}_4$ project onto the *XYZ* basis of {0}? Look at the sketches on the previous page. $\hat{X}_4$ is in the same direction as $\hat{Y}_0$. Similarly, how do $\hat{Y}_4$ and $\hat{Z}_4$ project onto the *XYZ* basis of {0}? Again, refer to the sketches. $\hat{Y}_4$ is in the same direction as $\hat{X}_0$ and $\hat{Z}_4$ is in the opposite direction of $\hat{Z}_0$. Therefore:

$$\left\{{}^{0}\hat{X}_{4}\right\}=\begin{Bmatrix}0\\1\\0\end{Bmatrix} \qquad \left\{{}^{0}\hat{Y}_{4}\right\}=\begin{Bmatrix}1\\0\\0\end{Bmatrix} \qquad \left\{{}^{0}\hat{Z}_{4}\right\}=\begin{Bmatrix}0\\0\\-1\end{Bmatrix} \qquad \left[{}^{0}_{4}R\right]=\begin{bmatrix}0&1&0\\1&0&0\\0&0&-1\end{bmatrix}$$

$\{B\}$ has the same orientation as Cartesian frame {0} so:

$$\left[{}^{B}_{4}R\right]=\left[{}^{0}_{4}R\right]=\begin{bmatrix}0&1&0\\1&0&0\\0&0&-1\end{bmatrix}$$

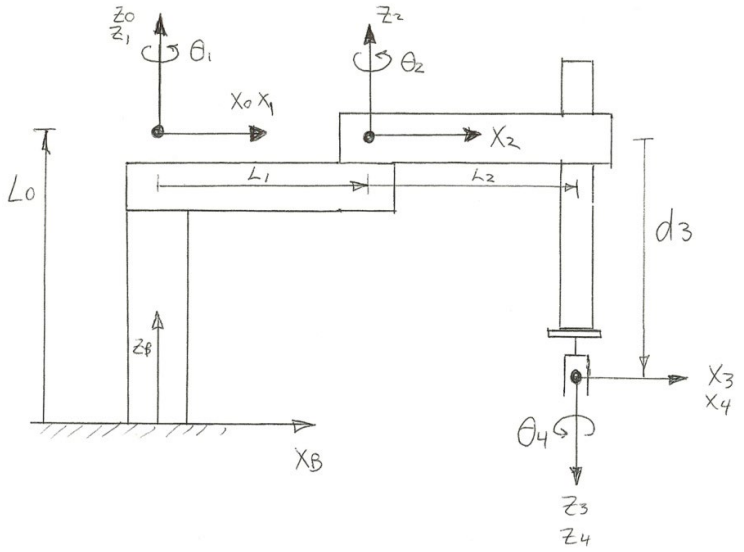
which agrees with the MATLAB numerical results given earlier, thus validating the rotation matrices part of the FPK solution. For the position vector part of the FPK solution, again refer to the sketches given above to see that:

$$\left\{{}^{0}P_{4}\right\}=\begin{Bmatrix}-L_2\\-L_1\\-d_3\end{Bmatrix}=\begin{Bmatrix}-0.250\\-0.300\\-0.150\end{Bmatrix} \qquad \text{and} \qquad \left\{{}^{B}P_{4}\right\}=\begin{Bmatrix}-L_2\\-L_1\\L_0-d_3\end{Bmatrix}=\begin{Bmatrix}-0.250\\-0.300\\0.402\end{Bmatrix}$$

which agrees with the MATLAB numerical results given earlier, thus validating the position vector part of the FPK solution.

4.3.3 3D MATLAB Graphics for Serial Robot Animation

This section presents some basic MATLAB code to draw a 3D robot to the screen, for either snapshots or motion animation following Forward Pose Kinematics, or other robotics calculations (presented later). Specifically we will show how to draw a 3D stick-figure representation for the laboratory SCARA robot below.



Adept 550 SCARA 4-dof 3D Robot

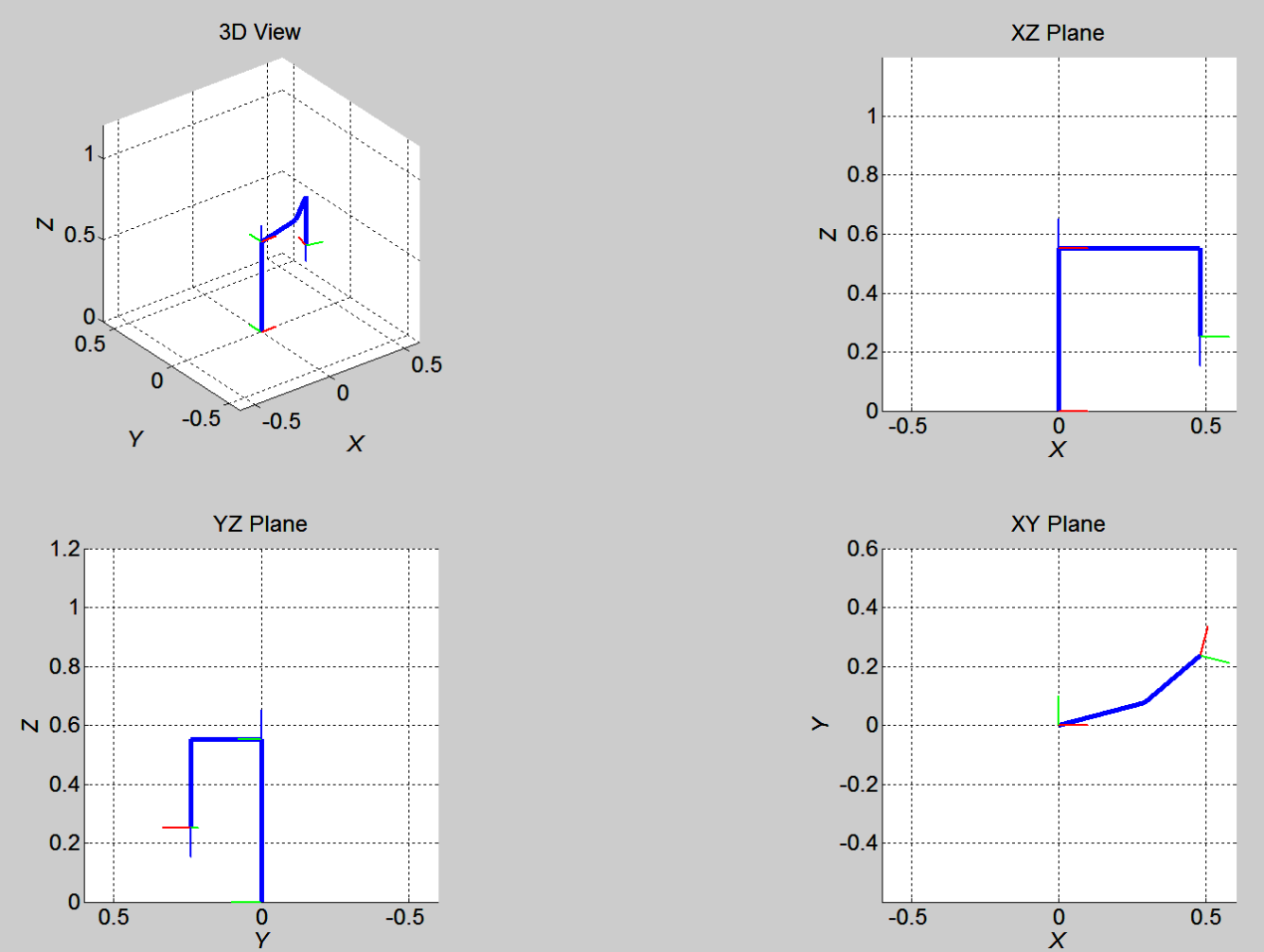
MATLAB Code

```
xx1 = [0                          0];      % link 0 from {B} to {0}
yy1 = [0                          0];
zz1 = [0                         L0];
xx2 = [0                      L1*c1];      % link 1 from {0} to {2}
yy2 = [0                      L1*s1];
zz2 = [L0                        L0];
xx3 = [L1*c1           L1*c1+L2*c12];      % link 2 from {2} to end of L2
yy3 = [L1*s1           L1*s1+L2*s12];
zz3 = [L0                        L0];
xx4 = [L1*c1+L2*c12    L1*c1+L2*c12];      % link 3 from end of L2 to {4}
yy4 = [L1*s1+L2*s12    L1*s1+L2*s12];
zz4 = [L0                   L0-d3];
figure;
plot3(xx1,yy1,zz1,'b',xx2,yy2,zz2,'b',xx3,yy3,zz3,'b',xx4,yy4,zz4,'b');
```

This will generate the basic stick-figure on the screen; here are some add-ons.

- Be sure to use **`axis('equal')`** or **`axis('square')`** with appropriate axis limits.
- Use **`grid`** to clarify the resulting image.
- Use **`'linewidth',4`** inside the **`plot3`** command to make thicker lines.
- Draw thinner unit vectors to represent the origins and directions of {*B*}, {0}, and {4}.
- Use **`subplot`** for 4 views (3 planar projections and orthographic view) – see **`view(AZ,EL)`**.
- Use *X Y Z* axis labels.
- Use **`plot`** instead of **`plot3`** for 2D planar robots; this does not require the *Z* components.

Here is what the simplified 3D 4-dof Adept 550 SCARA robot could look like in MATLAB.



This was generated using snapshot FPK for the inputs $\begin{bmatrix}\theta_1 & \theta_2 & d_3 & \theta_4\end{bmatrix} = \begin{bmatrix}15^\circ & 25^\circ & 0.300 & -35^\circ\end{bmatrix}$. The thicker robot lines are drawn in blue (`'b'` in the code) and the $\{B\}$, $\{0\}$, and $\{4\}$ axes use the convention r g b for the X Y Z unit vectors.

If you want to use this SCARA graphics in a motion simulation, you must use **`pause(dt)`** within your motion loop. Otherwise the animation will flash by so fast you cannot see it. **`dt`** can be calculated to approximately obtain real-time animations, though normal MATLAB and certainly MS Windows are not real-time.

Use of on-screen robot graphics is very helpful in validating that your simulated snapshots and motion trajectories are working correctly.

4.4 Robot Workspace

The **workspace** (also called **work envelope**) of a robot manipulator is defined as the volume in space that a robot's end-effector can reach. The workspace limits are due to the total reach of the combined robot links (both extended and folded upon each other). The workspace limits are also due to the specific joint limits of each joint.

Workspace is defined for both position (**translational workspace**) and orientation (**rotational workspace**). The **translational workspace** is easier to determine and display, and is the focus of this section. The figure below shows the **translational workspace** of a Cybotech P15 articulated robot, in side (left) and top (right) views.

Generally large translational and rotational workspaces are desired for a more capable robot. Generally serial robots have a big advantage over parallel robots in terms of larger workspace (exception: parallel cable-suspended robots).

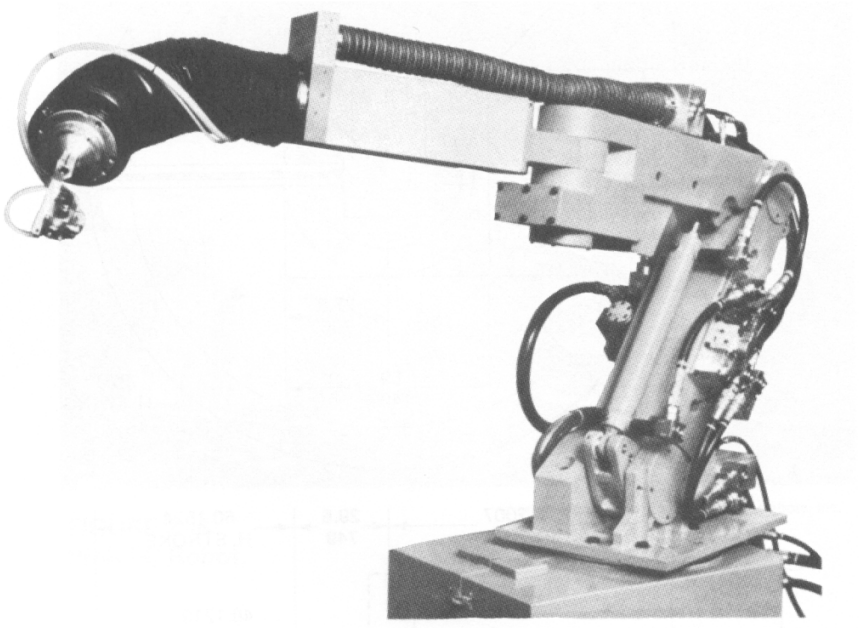
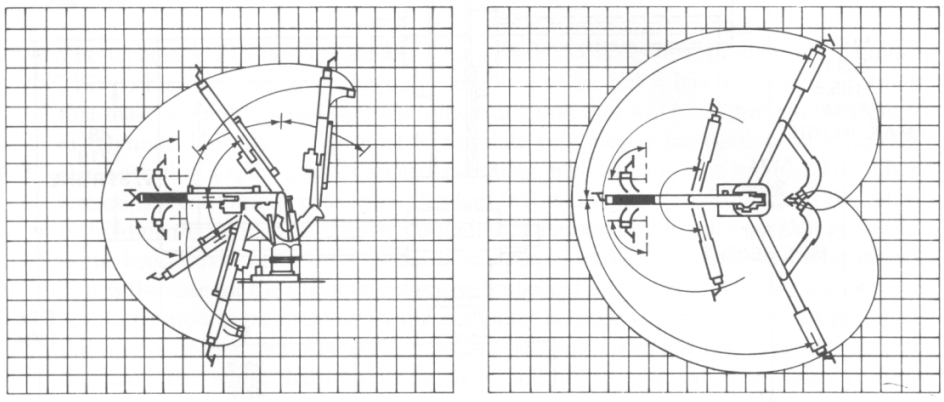


Photo used with permission of Cybotech Corporation.

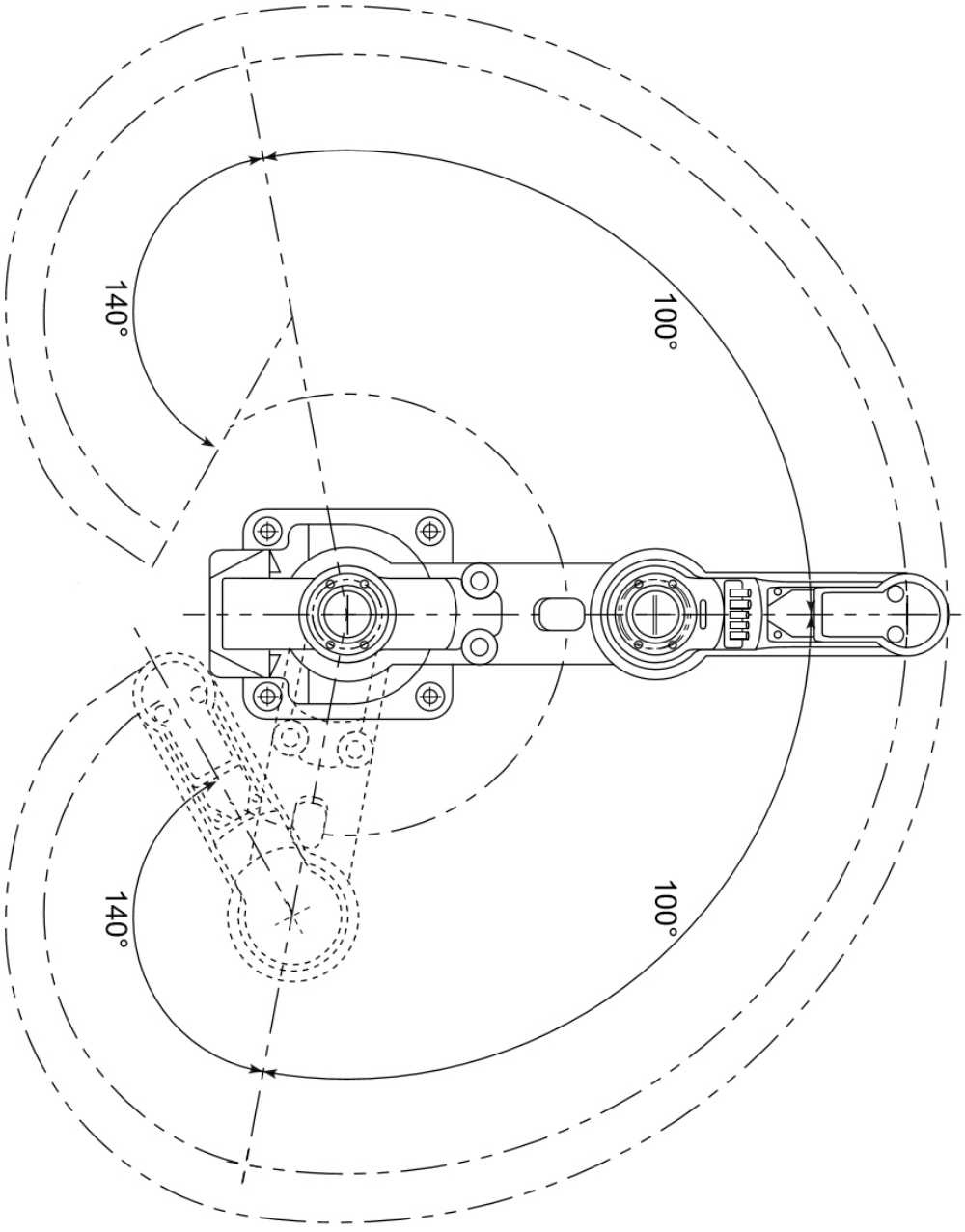


1 SQUARE = 1 FOOT

Cybotech P15 Manipulator Translational Workspace

The robot workspace attainable at **<u>any</u>** orientation is called the **reachable workspace**. The robot workspace attainable at **<u>all</u>** orientations is called the **dexterous workspace**. The **dexterous workspace** is a subset of the **reachable workspace**. For parallel manipulators the **dexterous workspace** is often null; therefore a limited dexterous workspace is generally defined (such as all orientations within $\pm 30^{\circ}$ rotation about all three axes).

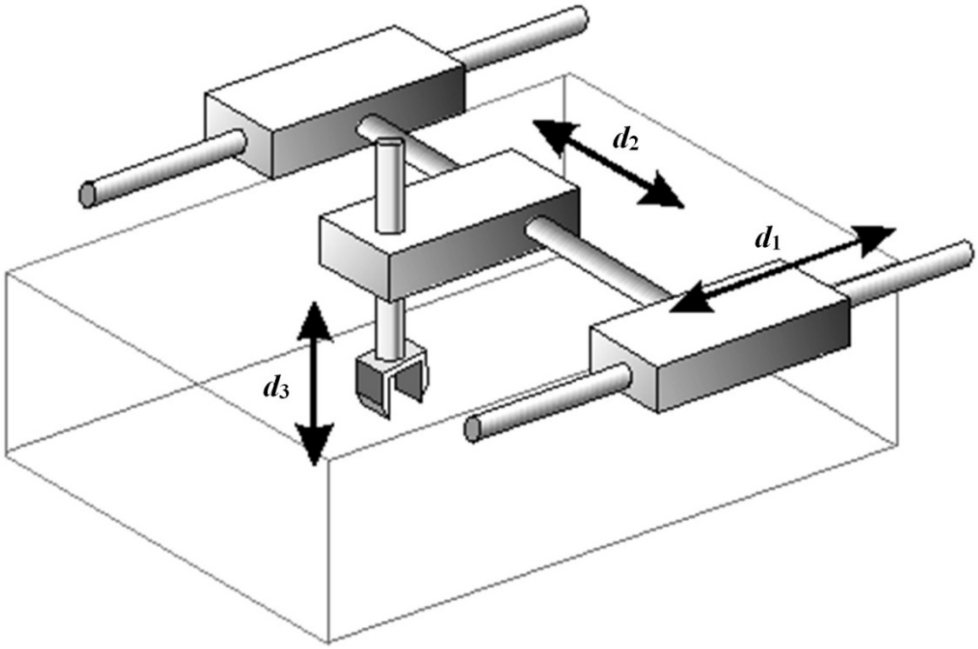
The reachable workspace of the Adept 550 SCARA robot is shown below. This top-view planar workspace shape is extended 200 mm in the vertical direction by the prismatic joint limits on d_3 (into the paper), to form a volume.



Adept 550 SCARA Robot Reachable Workspace

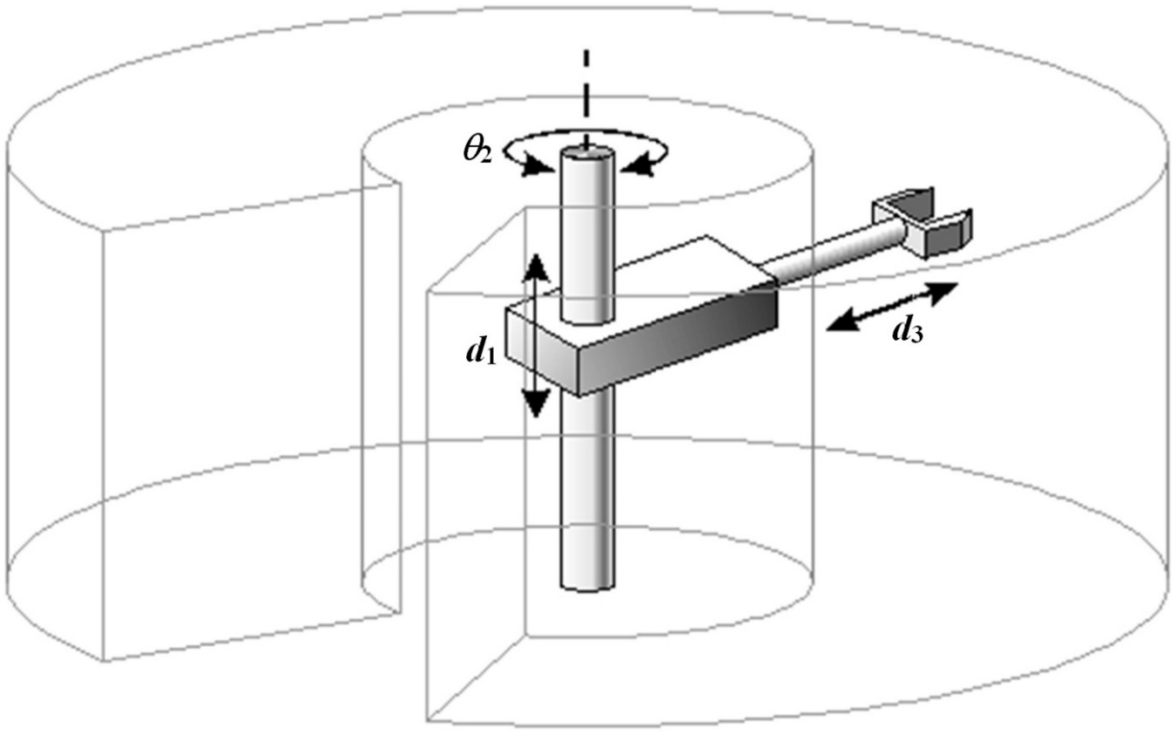
Adept 550 Table-Top Robot User's Guide, 1995

The next five images represent the reachable, or translational, workspaces of five common serial manipulator arrangements (Cartesian, cylindrical, spherical, SCARA, and articulated).



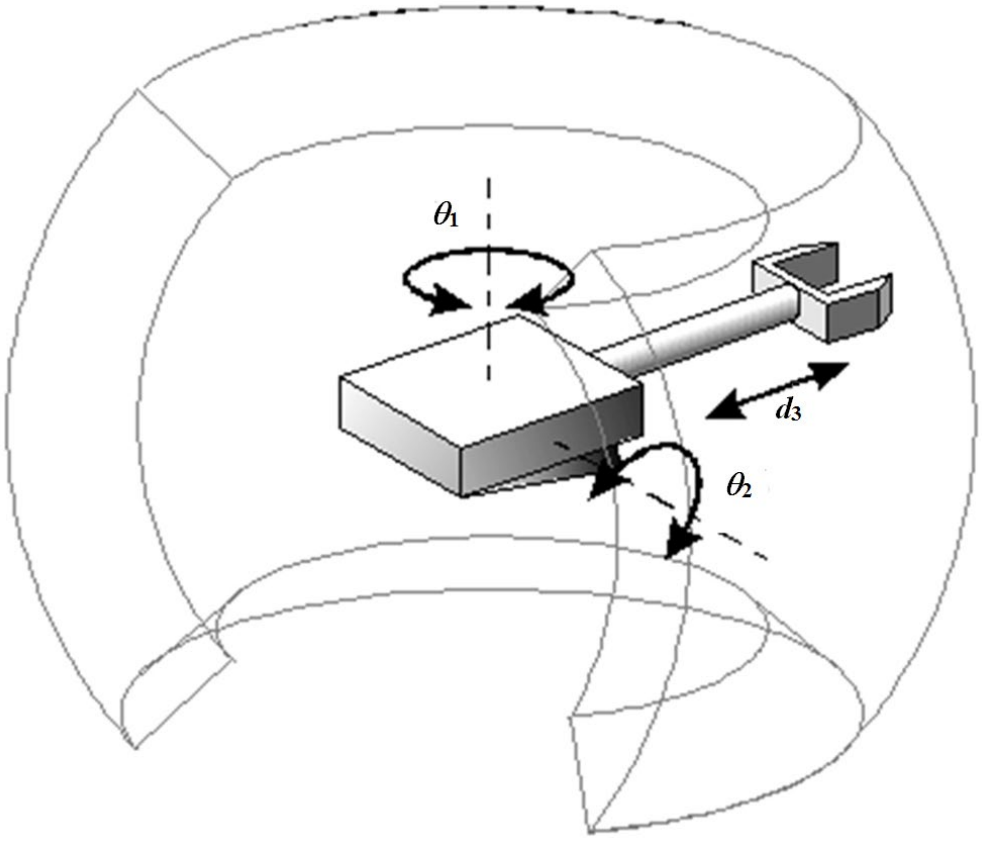
Cartesian Manipulator Translational Workspace

prime.jsc.nasa.gov/ROV/types.html



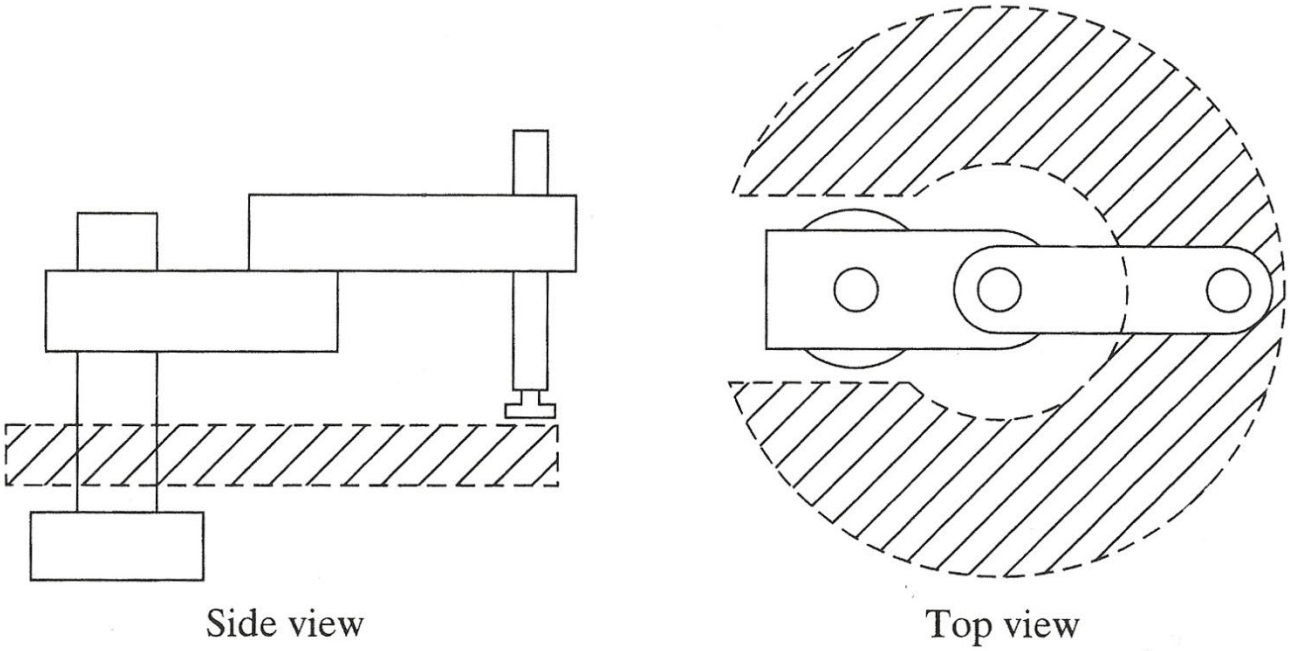
Cylindrical Manipulator Translational Workspace

prime.jsc.nasa.gov/ROV/types.html



Spherical Manipulator Translational Workspace

prime.jsc.nasa.gov/ROV/types.html



SCARA Manipulator Translational Workspace

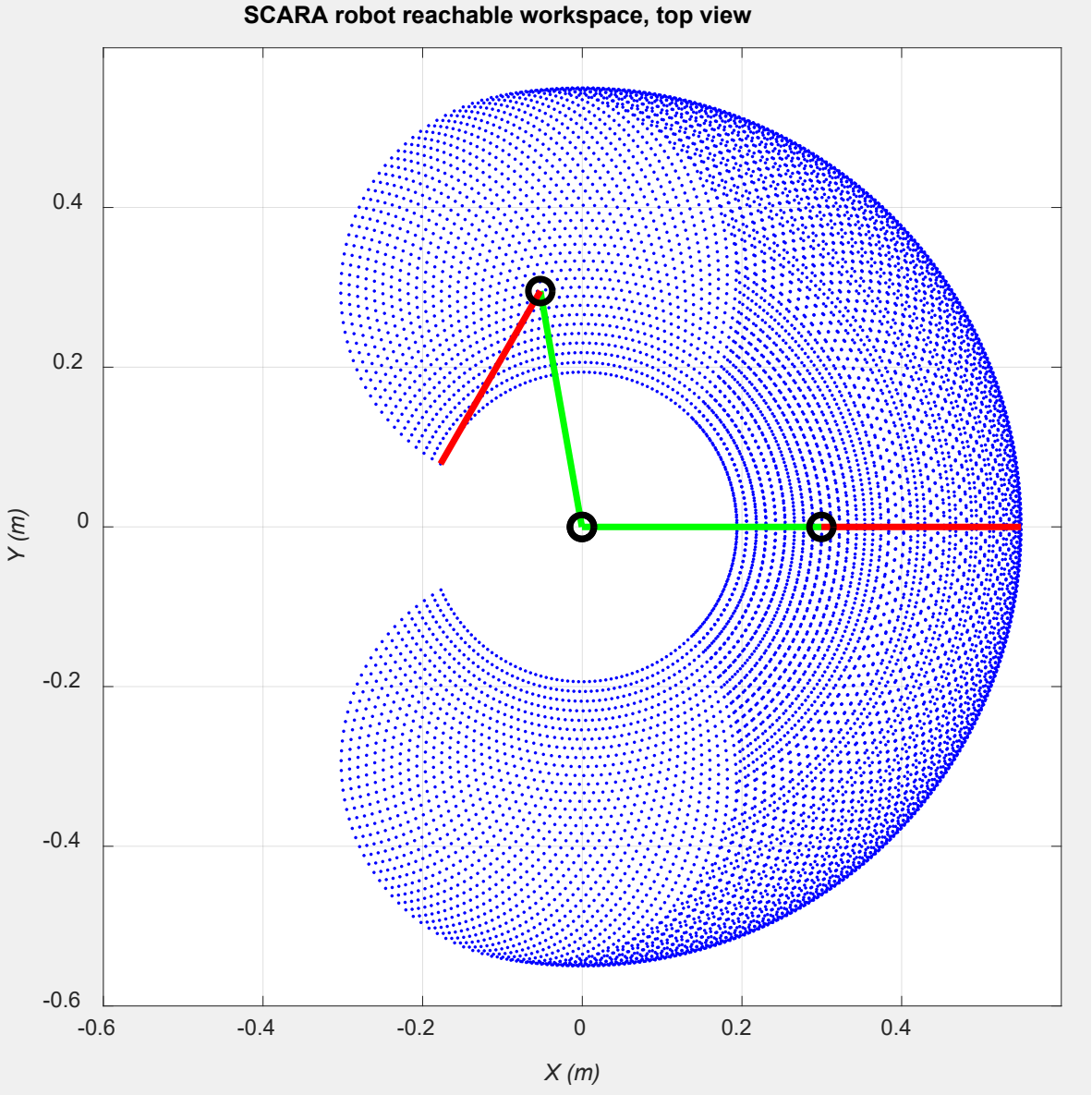
Craig (2005)

Actually, Craig's drawing of the SCARA reachable workspace Top View is quite wrong:

1. The outer workspace boundary extends beyond the hand, i.e. the wrist joint;
2. The internal workspace boundaries to the left cannot be straight as shown; and
3. The concentric circles must be centered about the shoulder joint, not the elbow joint.

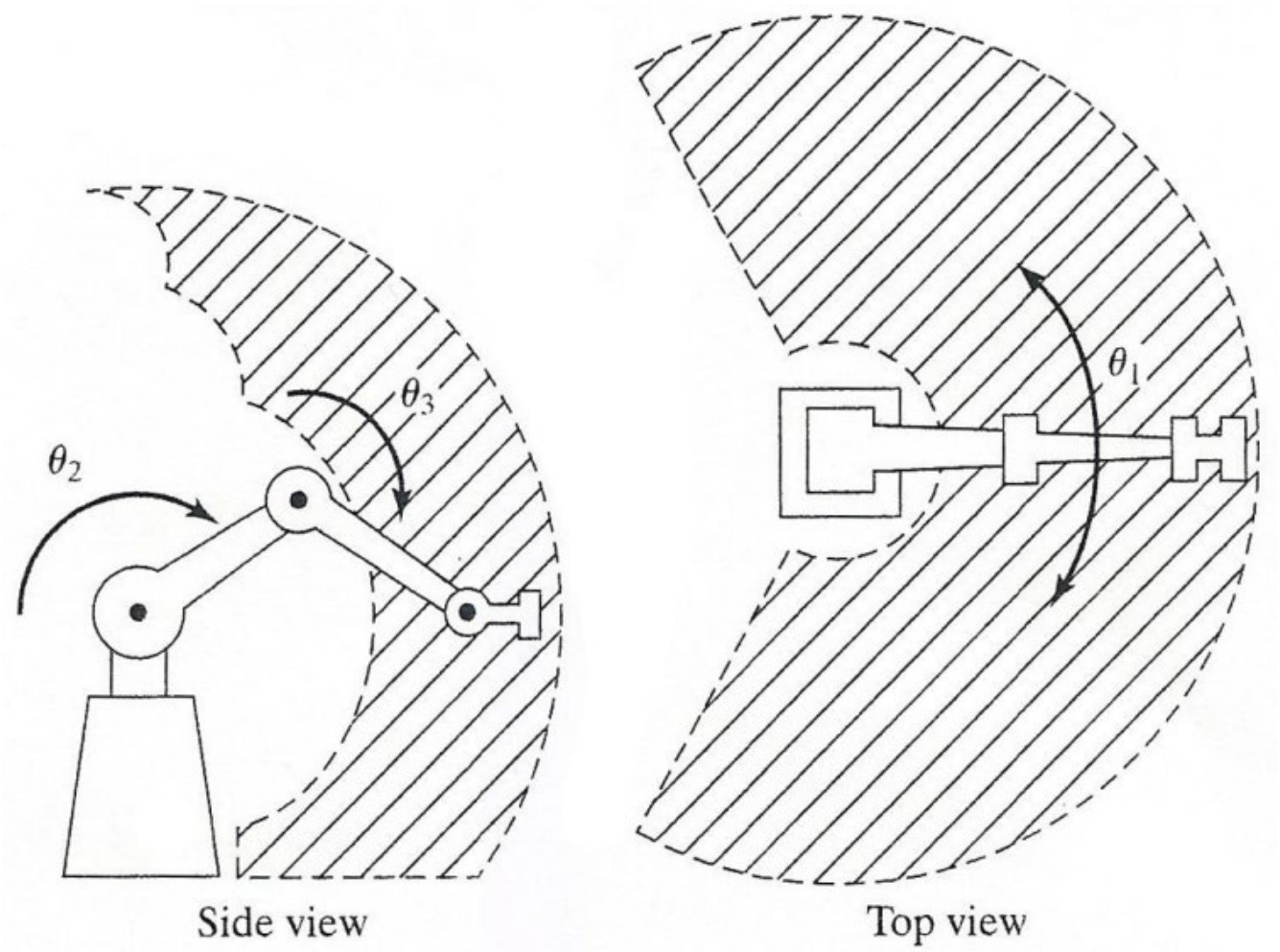
Craig (2005) tried to improve this situation, but only addressed 2 above (only partly corrected); 1 and 3 are still wrong.

The top-view SCARA workspace drawing from the Adept Manual, given earlier is more accurate. How can one validate this workspace? Let us use MATLAB and Forward Pose Kinematics to draw the top-view workspace of the Adept 550 SCARA robot. The shoulder joint is at (0,0).



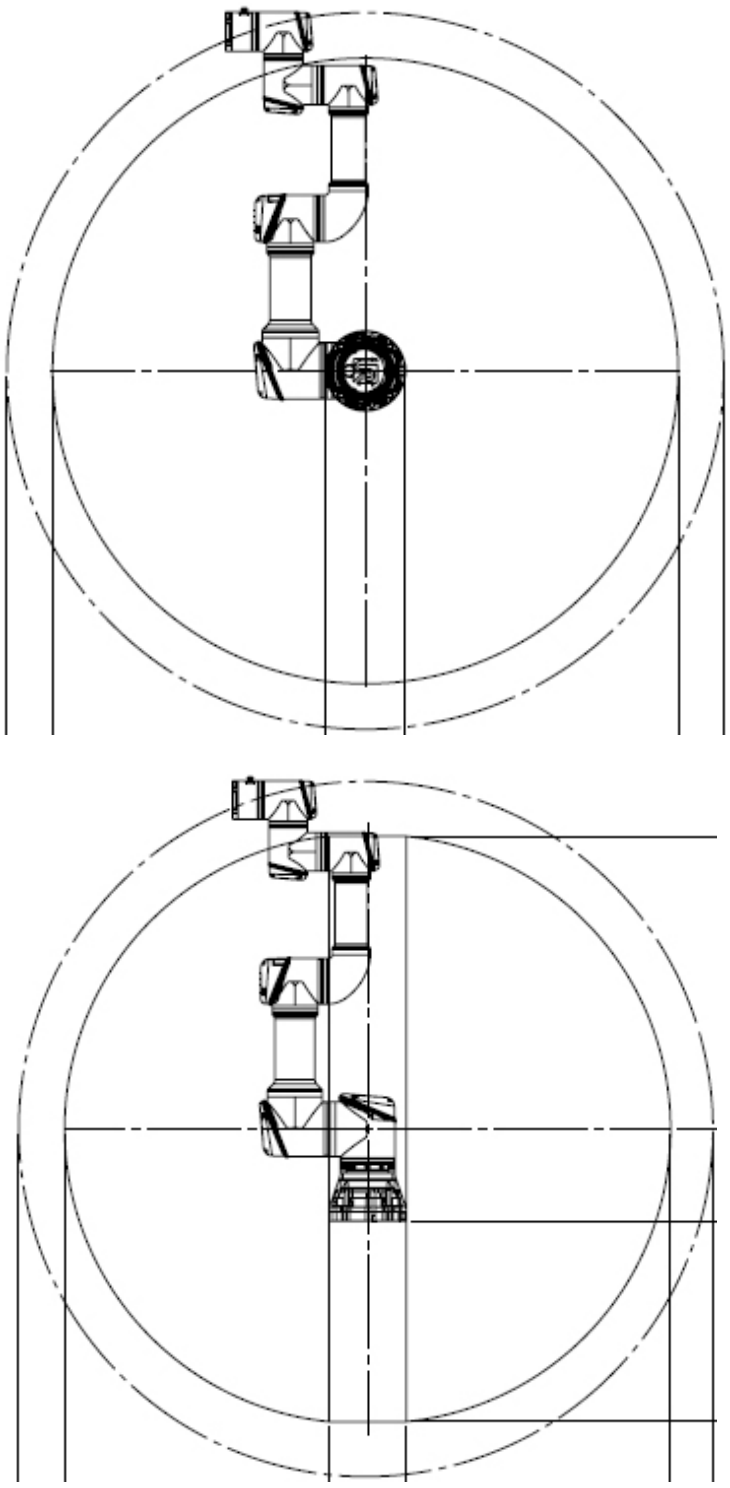
Note the following parameters were used in this workspace determination, from the Adept 550 SCARA:

$L_1 = 0.300$ $L_2 = 0.250$ (m) $\theta_1 = \pm 100^\circ$ $\theta_2 = \pm 140^\circ$



Articulated Manipulator Translational Workspace

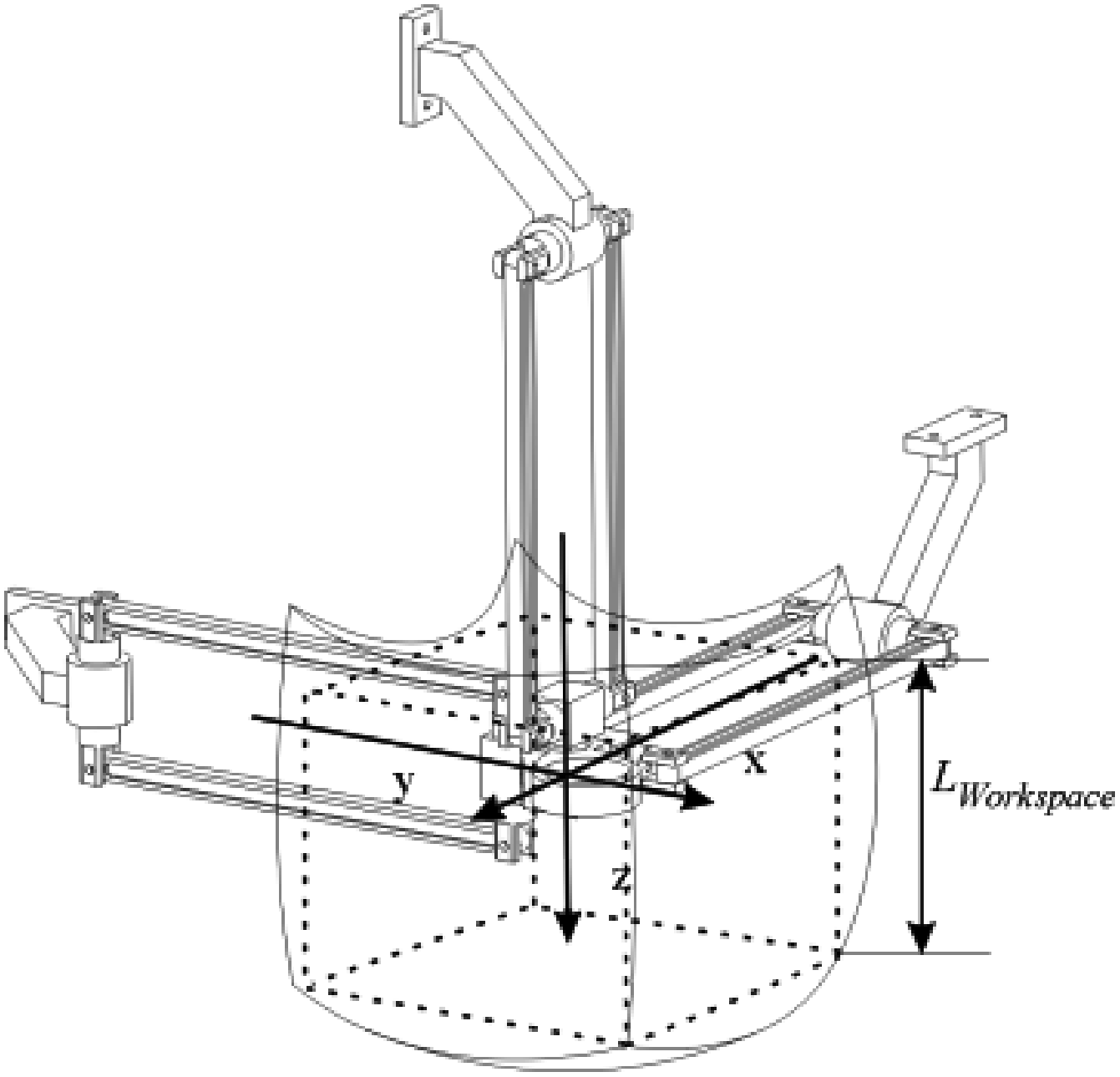
Craig (2005)



Universal UR3e 6-dof Cobot Reachable Workspace, top view and front view

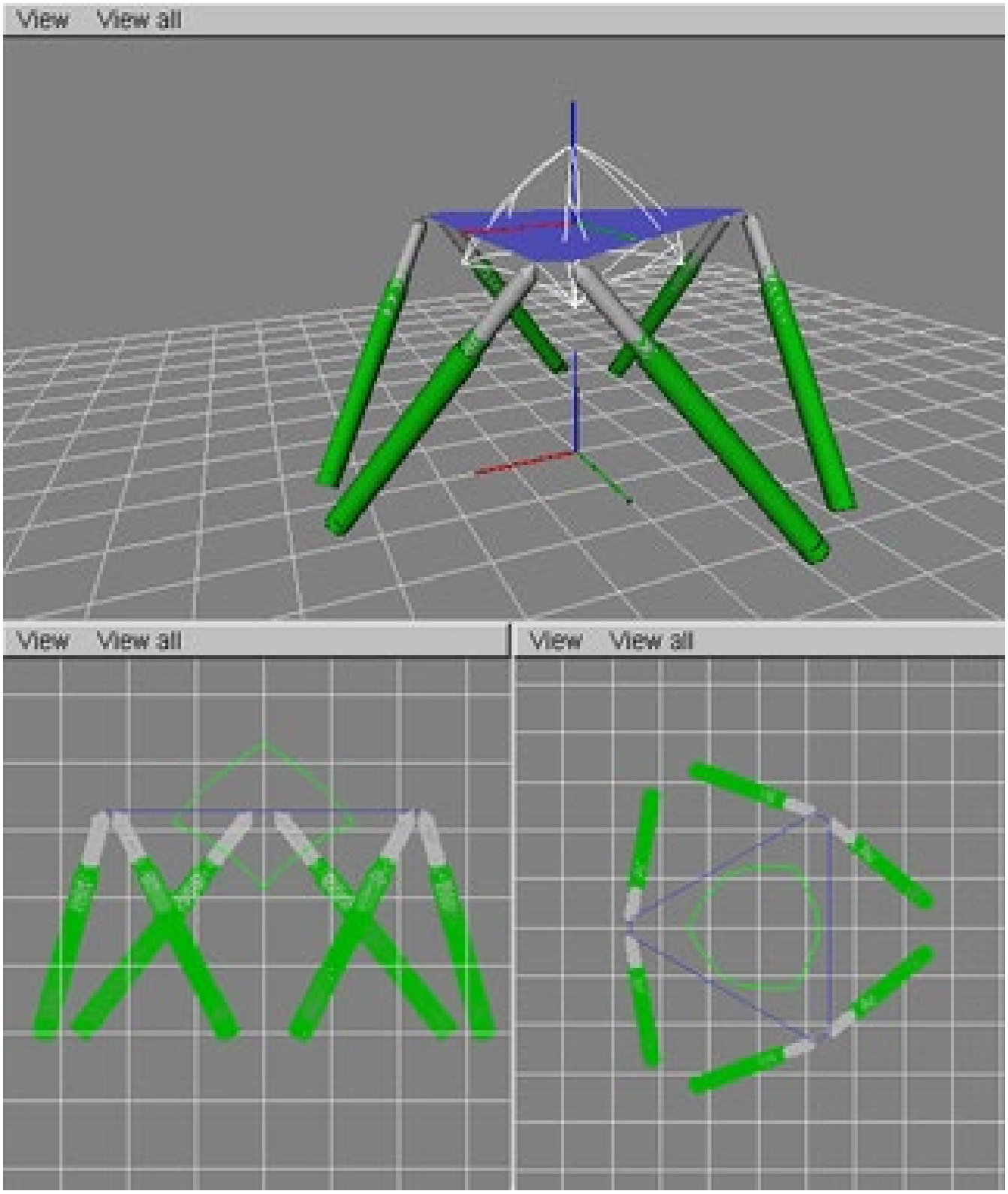
This robot has an amazing reachable workspace, even for a serial robot, since its first five joints have a generous $\pm 360^{\circ}$ (and the sixth joint is unlimited in rotation, both directions). As the figures show, this is a full spherical reachable workspace! Of course, mounted on a table, this would reduce to a hemisphere. With its base suspended from a pedestal, the spherical workspace is approached.

The figure below presents the translational workspace for a parallel robot (the three-axis Orthoglide). Imagine how much greater the translational workspace would be if there were only one supporting serial chain to the end-effector. However, the tradeoff is that the parallel robot achieves much greater accelerations, stiffness, and accuracy, with lower moving mass, than an equivalent serial robot with a larger workspace.



Orthoglide Robot Reachable Workspace

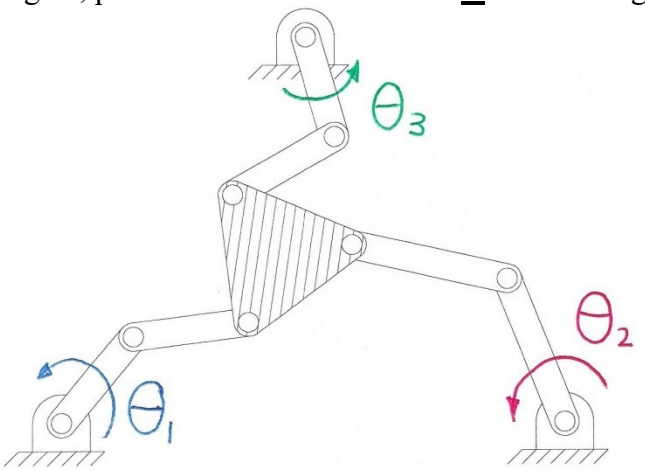
parallemic.org/Reviews/Review011.html



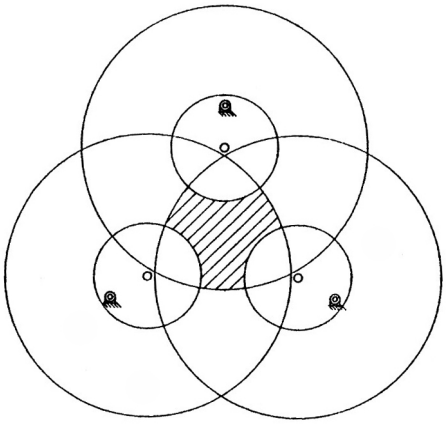
6-UPS Stewart Platform Parallel Robot Workspace

robot.gmc.ulaval.ca/en/research/theme105P.html

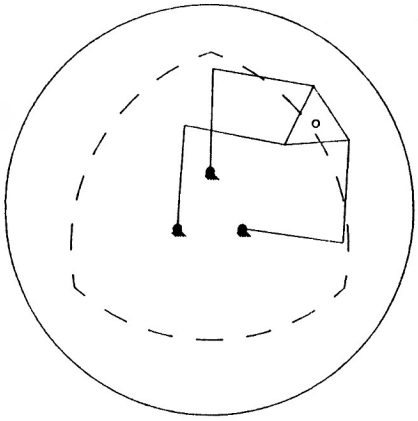
Again, parallel robots such as the **3-RRR** shown generally have small workspaces.



3-RRR Parallel Robot

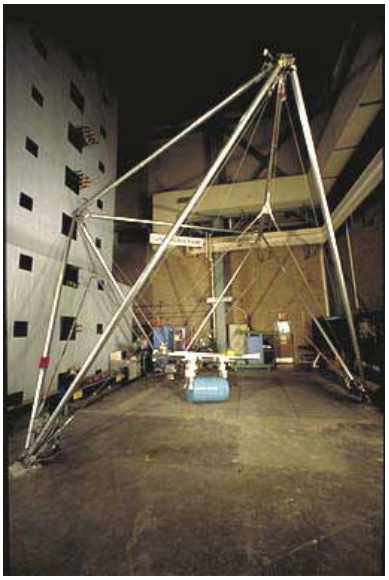


Sample Limited Workspace[5]



Optimized 3-RRR Parallel Robot Workspace[4]

A notable exception to the small workspace rule of parallel robots is the cable-suspended robots. These devices, such as the **NIST RoboCrane** (right, developed at NIST by Dr. James Albus), can be designed with arbitrarily-large workspaces. Cable-suspended robots with workspaces on the order of kilometers have been proposed and built.



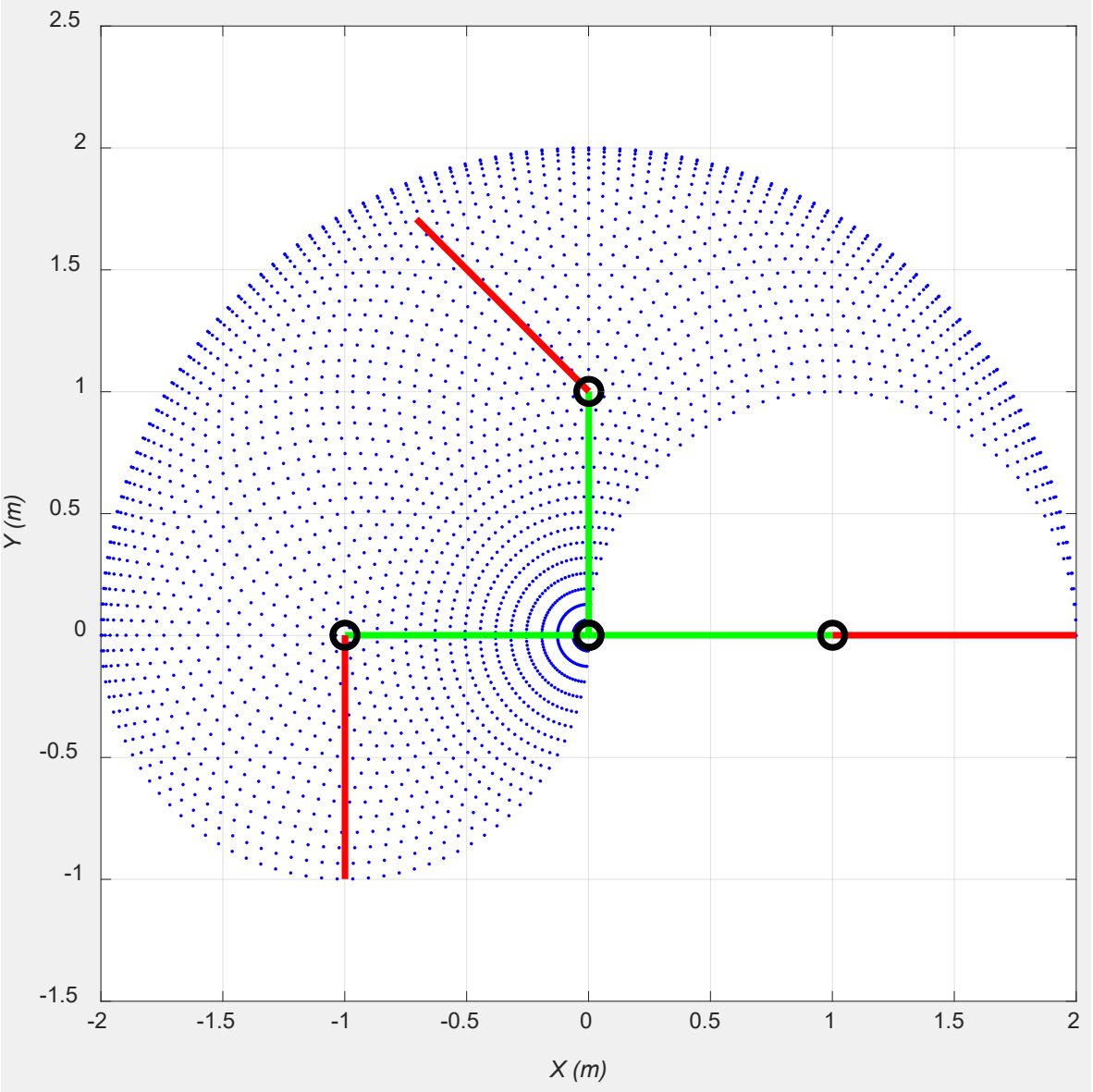
[5] R.L. Williams II, 1988, Planar Robotic Mechanisms: Analysis and Configuration Comparison, Ph.D. Dissertation, Department of Mechanical Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA, August.

Workspace Determination Example

Robot workspace may be determined by geometric methods, numerical methods, or combinations of these methods.

Considering link lengths and practical joint limits, the Forward Pose Kinematics solution may be simulated over the entire joints ranges of motion to establish the workspace. Assigning a dot to each valid reachable point, the reachable workspace may be generated numerically and graphically for planar and even spatial robots.

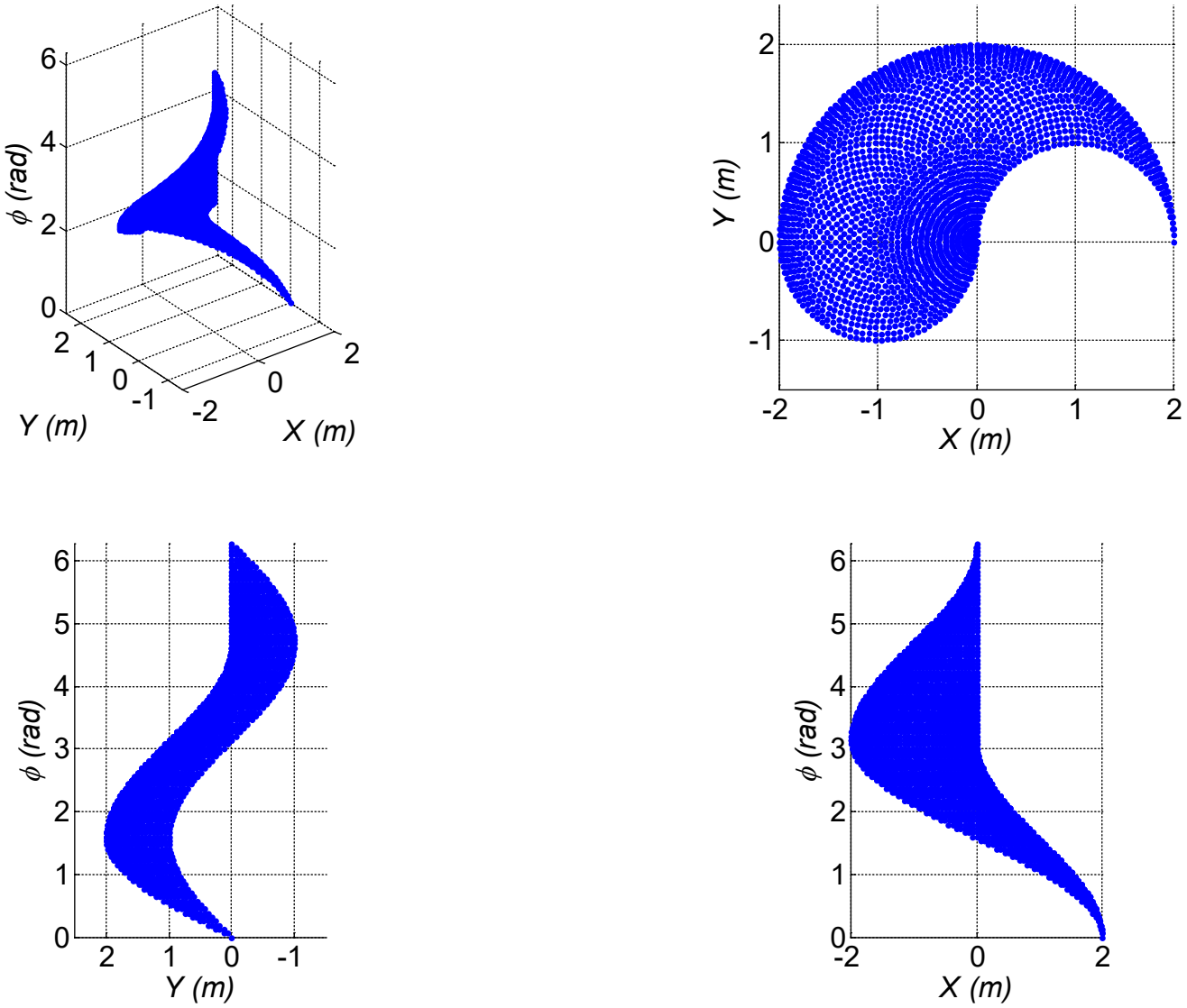
For a planar 2R serial robot with link lengths $L_1 = 1$ and $L_2 = 1$ (m), plus joint angle limits $0 \le \theta_1 \le 180^\circ$ and $0 \le \theta_2 \le 180^\circ$, with the grounded R joint at (0,0), the MATLAB graphic below displays the reachable workspace.



Planar 2R Serial Robot Reachable Workspace

$L_1 = 1$, $L_2 = 1$ (m), $0 \le \theta_1 \le 180^\circ$, $0 \le \theta_2 \le 180^\circ$

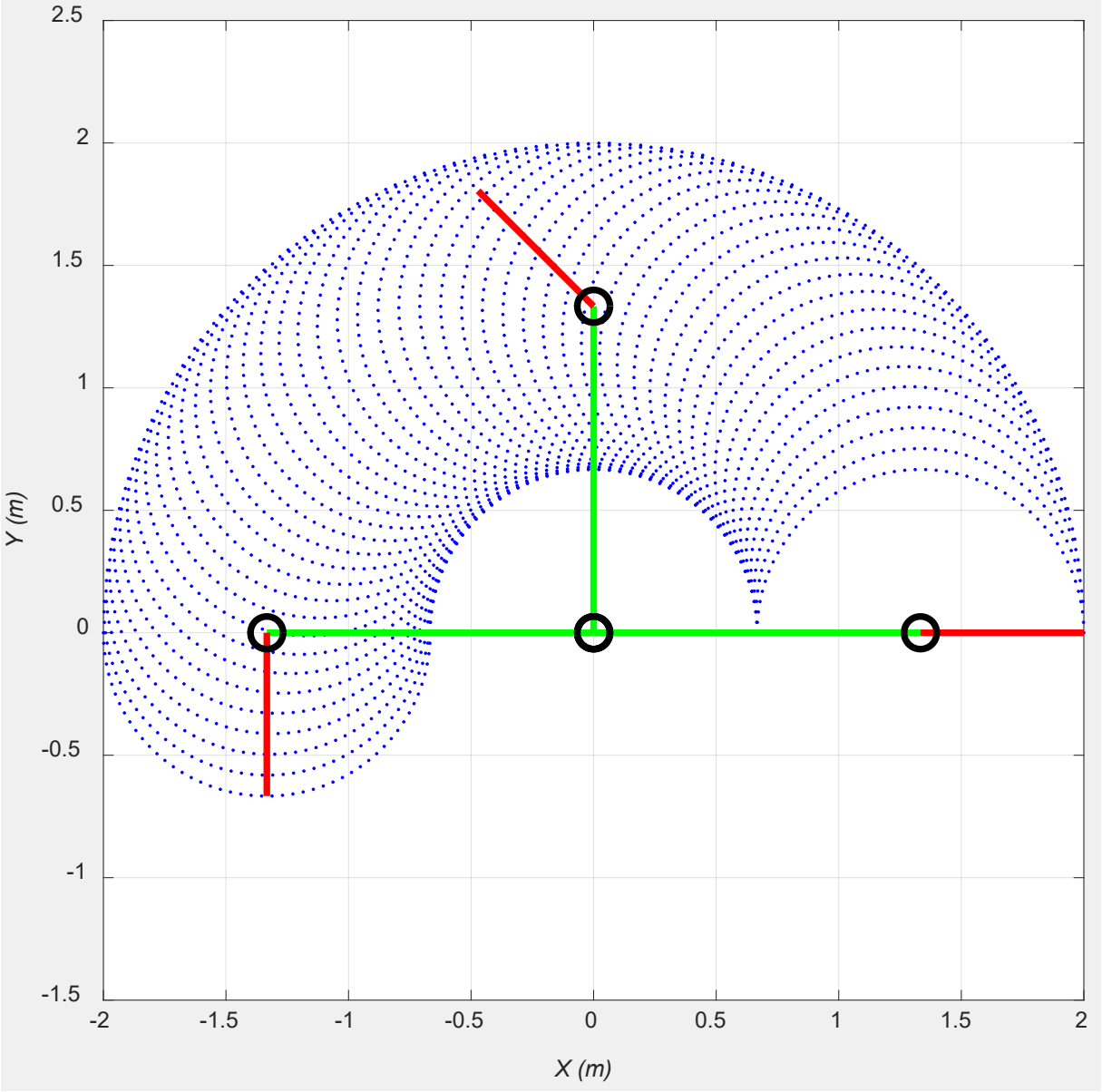
The next graphic shows the same robot workspace, considering the $\phi = \theta_1 + \theta_2$ rotational workspace in addition to the above reachable workspace (called the dexterous workspace).



Planar 2R Serial Robot Dexterous Workspace

$L_1 = 1$, $L_2 = 1$ (m), $0 \le \theta_1 \le 180^\circ$, $0 \le \theta_2 \le 180^\circ$

Finally, for this example, robot reachable workspace is clearly strongly related to the link lengths. For a planar 2R serial robot with link lengths $L_1 = 4/3$ and $L_2 = 2/3$ (m), plus joint angle limits $0 \le \theta_1 \le 180^\circ$ and $0 \le \theta_2 \le 180^\circ$, with the grounded R joint at (0,0), the MATLAB graphic below displays the reachable workspace. Note that in this example, $L_1 + L_2 = 2$ still, as in the above graphics, for a fair comparison, but now $L_1 = 2L_2$.



Planar 2R Serial Robot Reachable Workspace

$L_1 = 4/3$, $L_2 = 2/3$ (m), $0 \le \theta_1 \le 180^\circ$, $0 \le \theta_2 \le 180^\circ$

5. Inverse Pose Kinematics (IPK)

Given the serial robot geometry and numerical values for the desired pose of the end-effector Cartesian coordinate frame with respect to the base frame, calculate the required joint variables to achieve this pose. Recall the serial robot geometry is described by the DH Parameters, three constants and one variable per DH table row.

Given:

Find:

The IPK problem involves non-linear (transcendental), coupled equations to solve for the joint unknowns. The IPK solution is generally difficult and multiple solutions generally exist. Analytical solutions do not exist for some manipulator structures. Inverse Pose Kinematics is required for robot pose control. The required equations to solve are from the Forward Pose Kinematics problem.

Forward Pose Kinematics (FPK) of Serial Robots

The FPK solution involves multiplying consecutive link homogeneous transformation matrices, regardless of the number of joints. There is a unique solution which can be found numerically or symbolically, that always exists with no singularity problems. This problem is straight-forward for serial-chain robots.

Inverse Pose Kinematics of Serial Robots

Task space (Cartesian) degrees-of-freedom m vs. Joint space degrees-of freedom n

a) $m > n$**; overdetermined**, no IPK solution exists (since you are trying to operate with a limited robot that can only cover a subspace of the desired task space).

b) $m = n$**; determined**, finite IPK solutions exist (such a robot spans the task space one-to-one). This is the case we will consider in class.

c) $m < n$**; underdetermined**, infinite IPK solutions exist (this is the kinematic redundancy case). We can optimize the robot performance in addition to following desired pose trajectories.

Number of solutions

For $m = n$, there are generally (finite) multiple IPK solution sets, for example elbow up or elbow down. There are more than one manipulator configuration to achieve a given Cartesian pose. Demonstrate with the PUMA model.

Existence of solutions

For $m = n$ or $m < n$, no real IPK solution exists when the given Cartesian pose is out of the manipulator's workspace. The solutions are complex (imaginary) in this case.

For $m > n$, an IPK solution exists if the pose is within the workspace, and ***ONLY IF*** the commanded pose is within the limited task space spanned by the manipulator. For example, a 5-dof ($n = 5$) industrial robot cannot turn a ratchet in general, since it lacks the last wrist roll joint, but is sufficient for axis-symmetric applications (such as spray painting).

This is still the $m = n$ case because m is reduced to $m = 5$ for axis-symmetric Cartesian tasks not requiring the robot roll freedom.

5.1 Inverse Pose Kinematics Solution Methods

Numerical – e.g. Newton-Raphson iteration. Requires a good initial guess and only finds one solution, closest to the initial guess.

Graphical (draw and measure the solution)

Analytical (closed-form solution)

Algebraic vs. Geometric, or a combination of the two

Tan-half angle substitution – useful in solving inverse pose kinematics

General Analytical Solution Procedure for Inverse Pose Kinematics of Serial Robots

The solution of the Inverse Pose Kinematics problem starts with the same expressions from Forward Pose Kinematics. Write 16 equations (equate two Homogeneous Transformation Matrices). The LHS matrix is composed of known, given numbers from the given pose. The RHS matrix elements are functions of the unknown DH Parameter variables (from the Forward Pose Kinematics Homogeneous Transformation Matrix). For IPK the Cartesian pose $\left[{}^{B}_{H}T\right]$ is known (commanded) and the joint variables $\{\theta_1 \quad \theta_2 \quad \theta_3 \quad \cdots \quad \theta_n\}^T$ are unknown.

Spatial Serial Robots

$$\left[{}^{B}_{H}T\right] = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} f_{11}(\theta_1,\theta_2,\cdots\theta_n) & f_{12}(\theta_1,\theta_2,\cdots\theta_n) & f_{13}(\theta_1,\theta_2,\cdots\theta_n) & f_{14}(\theta_1,\theta_2,\cdots\theta_n) \\ f_{21}(\theta_1,\theta_2,\cdots\theta_n) & f_{22}(\theta_1,\theta_2,\cdots\theta_n) & f_{23}(\theta_1,\theta_2,\cdots\theta_n) & f_{24}(\theta_1,\theta_2,\cdots\theta_n) \\ f_{31}(\theta_1,\theta_2,\cdots\theta_n) & f_{32}(\theta_1,\theta_2,\cdots\theta_n) & f_{33}(\theta_1,\theta_2,\cdots\theta_n) & f_{34}(\theta_1,\theta_2,\cdots\theta_n) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{aligned} r_{11} &= f_{11}(\theta_1,\theta_2,\cdots\theta_n) & r_{12} &= f_{12}(\theta_1,\theta_2,\cdots\theta_n) & r_{13} &= f_{13}(\theta_1,\theta_2,\cdots\theta_n) & p_x &= f_{14}(\theta_1,\theta_2,\cdots\theta_n) \\ r_{21} &= f_{21}(\theta_1,\theta_2,\cdots\theta_n) & r_{22} &= f_{22}(\theta_1,\theta_2,\cdots\theta_n) & r_{23} &= f_{23}(\theta_1,\theta_2,\cdots\theta_n) & p_y &= f_{24}(\theta_1,\theta_2,\cdots\theta_n) \\ r_{31} &= f_{31}(\theta_1,\theta_2,\cdots\theta_n) & r_{32} &= f_{32}(\theta_1,\theta_2,\cdots\theta_n) & r_{33} &= f_{33}(\theta_1,\theta_2,\cdots\theta_n) & p_z &= f_{34}(\theta_1,\theta_2,\cdots\theta_n) \\ 0 &= 0 & 0 &= 0 & 0 &= 0 & 1 &= 1 \end{aligned}$$

16 equations, 4 trivial (row 4 [0 0 0 1]) – down to 12 equations.

3 *XYZ* translational equations (column 4); all 3 are independent – still at 12 equations.

9 rotational equations; only 3 are independent – down to 6 equations.

Result: 6 independent equations so we can handle up to 6 joint unknowns

Planar Serial Robots (*XY* plane)

$$\left[{}^{B}_{H}T\right] = \begin{bmatrix} r_{11} & r_{12} & 0 & p_x \\ r_{21} & r_{22} & 0 & p_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} f_{11}(\theta_1,\theta_2,\cdots\theta_n) & f_{12}(\theta_1,\theta_2,\cdots\theta_n) & 0 & f_{14}(\theta_1,\theta_2,\cdots\theta_n) \\ f_{21}(\theta_1,\theta_2,\cdots\theta_n) & f_{22}(\theta_1,\theta_2,\cdots\theta_n) & 0 & f_{24}(\theta_1,\theta_2,\cdots\theta_n) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{array}{llll} r_{11} = f_{11}(\theta_1,\theta_2,\cdots\theta_n) & r_{12} = f_{12}(\theta_1,\theta_2,\cdots\theta_n) & 0 = 0 & p_x = f_{14}(\theta_1,\theta_2,\cdots\theta_n) \\ r_{21} = f_{21}(\theta_1,\theta_2,\cdots\theta_n) & r_{22} = f_{22}(\theta_1,\theta_2,\cdots\theta_n) & 0 = 0 & p_y = f_{24}(\theta_1,\theta_2,\cdots\theta_n) \\ 0 = 0 & 0 = 0 & 1 = 1 & 0 = 0 \\ 0 = 0 & 0 = 0 & 0 = 0 & 1 = 1 \end{array}$$

16 equations, 4 trivial (row 4 [0 0 0 1]) – down to 12 equations.

6 more equations are trivial (row 3, plus column 3) – down to 6 equations.

2 *XY* translational equations (column 4); both are independent – still at 6 equations.

4 rotational equations; only 1 is independent – down to 3 equations.

Result: 3 independent equations so we can handle up to 3 joint unknowns

If ***m*** = ***n*** (the number of Cartesian and joint degrees-of-freedom match) we can solve the problem. Perhaps it cannot be solved in analytical closed-form; sometimes the solution has to be numerical.

m = ***n*** **Examples**

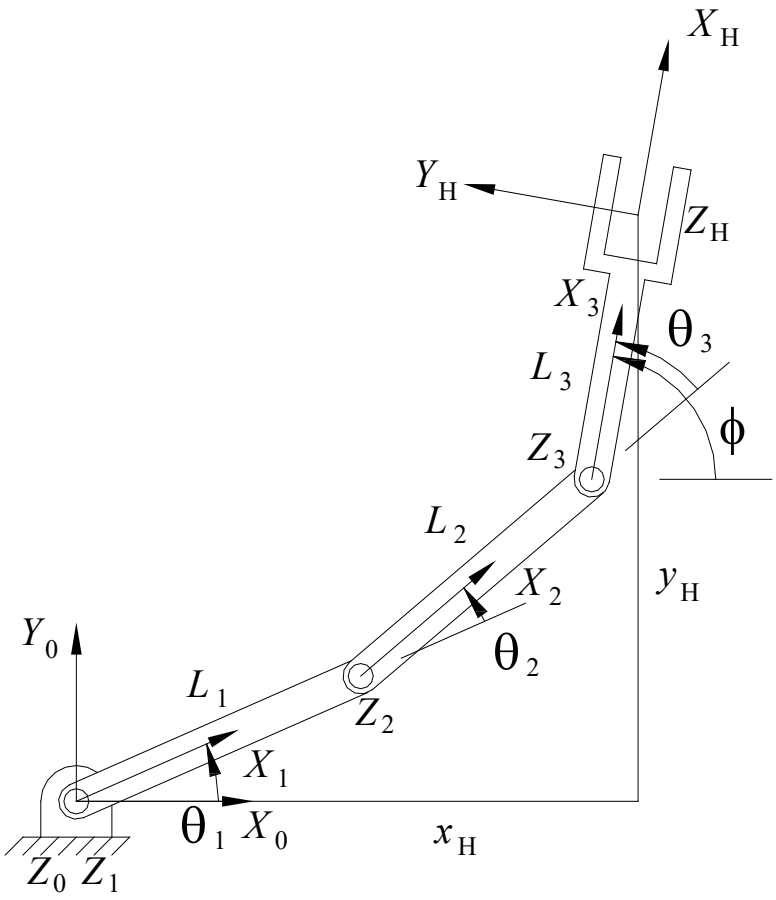
Spatial			**Planar**		
$m = n = 4$	SCARA	$\{x \quad y \quad z \quad \theta_z\}$	$m = n = 2$	Planar 2R	$\{x \quad y\}$
$m = n = 5$	Mitsubishi	$\{x \quad y \quad z \quad \theta_x \quad \theta_y\}$	$m = n = 3$	Planar 3R	$\{x \quad y \quad \phi_z\}$
$m = n = 6$	PUMA	$\{x \quad y \quad z \quad \theta_x \quad \theta_y \quad \theta_z\}$			

Pieper's solution – if at least three consecutive axes are intersecting in a serial robot (at least three consecutive coordinate frames share a common origin point), an analytical inverse pose solution is guaranteed to exist. So a 6-dof spatial robot with a spherical wrist is guaranteed to have an analytical IPK solution.

Pose Kinematics of Parallel Robots

There is an interesting duality with serial robots. For parallel robots, the forward kinematics solution is generally coupled and non-linear, while the inverse kinematics solution is generally more straight-forward. This is reversed for serial robots.

5.2 Planar 3R Robot IPK Solution



Inverse Pose Kinematics Symbolic Solution

Given: the robot (the 9 constant DH Parameters) and $\left[{}^{0}_{H}T\right]$ (the pose to reach)

Find: the three active joint angles $\theta_1,\theta_2,\theta_3$ (the 3 variable DH Parameters)

Forward Kinematics Expressions:

$$\left[{}^{0}_{H}T\right]=\begin{bmatrix} r_{11} & r_{12} & 0 & p_x \\ r_{21} & r_{22} & 0 & p_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}=\begin{bmatrix} c\phi & -s\phi & 0 & x_H \\ s\phi & c\phi & 0 & y_H \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}=\begin{bmatrix} c_{123} & -s_{123} & 0 & L_1c_1+L_2c_{12}+L_3c_{123} \\ s_{123} & c_{123} & 0 & L_1s_1+L_2s_{12}+L_3s_{123} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where:

$$c_{12}=\cos(\theta_1+\theta_2) \qquad c_{123}=\cos(\theta_1+\theta_2+\theta_3)$$
$$s_{12}=\sin(\theta_1+\theta_2) \qquad s_{123}=\sin(\theta_1+\theta_2+\theta_3)$$

Subspace of general 3D pose space

Note that the Forward Pose Kinematics result can be represented by $\left[{}^{0}_{H}T\right]$ or by $\left\{{}^{0}X_{H}\right\}=\left\{x_H \quad y_H \quad \phi\right\}^T$. The given numerical matrix $\left[{}^{0}_{H}T\right]$ cannot be a general 3D pose since this is a planar robot. The expression of the subspace of general 6-dof poses spanned by the *XY* plane is given above, described by $\left[{}^{0}_{H}T\right]$.

Inverse Pose Kinematics Solution Methods

1) Graphical
2) Geometric
3) Algebraic/trigonometric – we will now follow this method to solve the problem.

First, take advantage of the fixed L_3 transform to eliminate θ_3 from the translational equations:

$$\left[{}^{0}_{H}T\right]=\left[{}^{0}_{3}T\right]\left[{}^{3}_{H}T(L_3)\right]$$

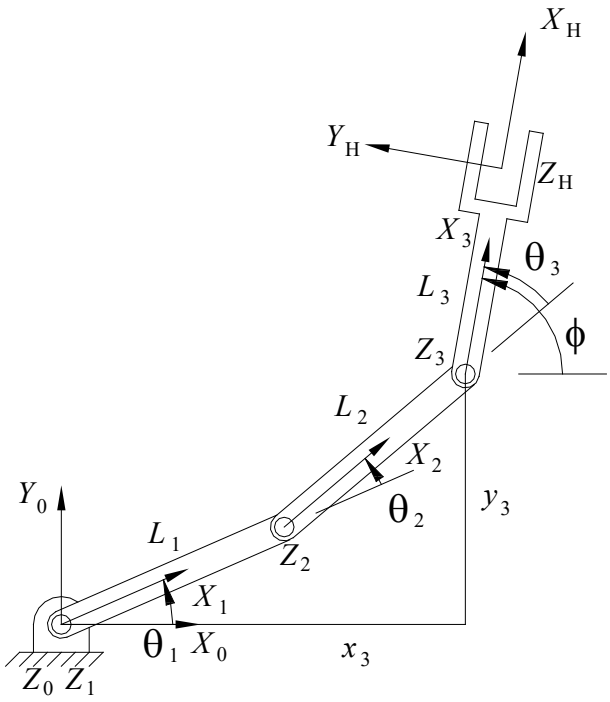
$$\left[{}^{0}_{3}T\right]=\left[{}^{0}_{H}T\right]\left[{}^{3}_{H}T(L_3)\right]^{-1}$$

$$\left[{}^{0}_{3}T\right]=\begin{bmatrix} c\phi & -s\phi & 0 & x_3 \\ s\phi & c\phi & 0 & y_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}=\begin{bmatrix} c_{123} & -s_{123} & 0 & L_1c_1+L_2c_{12} \\ s_{123} & c_{123} & 0 & L_1s_1+L_2s_{12} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where, by inspection:

$$\left[{}^{3}_{H}T(L_3)\right]=\begin{bmatrix} 1 & 0 & 0 & L_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and L_3 is a constant.



Inverse Pose Equations to Solve

Write the 2 independent translational equations:

Out of the 4 rotational equations, only 1 is independent; we use a ratio of 2 equations:

Alternatively, the simplest rotational equation to use is:

We have three equations to solve for the three unknowns. Thanks to the fixed transform simplification, the two translational equations involve only θ_1, θ_2 (fully coupled and nonlinear in these two unknowns); the rotational equation involves all three unknowns. Therefore, let us solve for θ_1, θ_2 first from the translational equations and find θ_3 from the rotational equation second. Rearrange the translational equations so the θ_2 terms are isolated:

Square and add these rearranged translational equations to eliminate θ_2:

The result is one equation in one unknown θ_1:

$$E = -2L_1 x_3$$
$$F = -2L_1 y_3$$
$$G = x_3^2 + y_3^2 + L_1^2 - L_2^2$$

We can solve this equation for θ_1 by using the **Tangent Half-Angle Substitution** (this is derived in the on-line EE/ME 4290/5290 Supplement).

If we define $t = \tan\left(\frac{\theta_1}{2}\right)$

then $\cos\theta_1 = \frac{1-t^2}{1+t^2}$ and $\sin\theta_1 = \frac{2t}{1+t^2}$

Substitute this **Tangent Half-Angle Substitution** into the *EFG* equation:

$$E\left(\frac{1-t^2}{1+t^2}\right) + F\left(\frac{2t}{1+t^2}\right) + G = 0$$

$$E(1-t^2) + F(2t) + G(1+t^2) = 0$$

$$(G-E)t^2 + (2F)t + (G+E) = 0$$

using the quadratic formula and simplifying:

$$t_{1,2} = \frac{-F \pm \sqrt{E^2 + F^2 - G^2}}{G-E}$$

Solve for θ_1 by inverting the original Tangent Half-Angle Substitution definition:

Two θ_1 solutions result from the $\pm$ in the quadratic formula. Both are correct since there are two valid solutions – elbow up and elbow down.

Recall that **Trigonometric Uncertainty** says the plain inverse tangent function **`atan`** yields a double-value (the primary angle is in Quadrant I or IV of the trigonometric unit circle; the second solution is then the primary angle plus 180°). So why did we use the plain inverse tangent function **`atan`**, which takes a single ratio input rather than the **`atan2`** function, which takes **`(num,den)`** as the input and uses logic to select the one correct Quadrant for our answer? (Especially since we do have a numerator and denominator in $t_{1,2}$.)

Answer.

From above, the answer for θ_1 is:

$$\theta_1 = 2\tan^{-1}(t)$$

There are two branches, one for each *t* value; only one is shown above. Recall we have a numerator and denominator for each *t*, but we can ignore them and form a single ratio for the **`atan`** function. With the 2 multiplying the inverse tangent result, it doesn't matter whether we use **`atan`** or **`atan2`** since the final answer will come to the same angle. Example:

For $\frac{\theta_1}{2} = \tan^{-1}(0.5774)$, we don't know if the solution is:

$$\frac{\theta_1}{2} = 30^\circ \qquad \text{or} \qquad \frac{\theta_1}{2} = 210^\circ$$

However, the multiplier 2 takes care of this uncertainty, since angles are identical every 360°:

$$\theta_1 = 60^\circ \qquad \text{or} \qquad \theta_1 = 420^\circ = 60^\circ$$

Returning to the IPK solution, to find θ_2, again use the *XY* translational equations.

Find the unique θ_2 for each θ_1 value (this time we must use the quadrant-specific **`atan2`** function in MATLAB).

Now that we have two solution sets for (θ_1,θ_2), return to the rotational equation to solve for θ_3.

There are two overall solution sets to the inverse pose kinematics problem for the planar 3R robot. Be sure to keep the solutions together, i.e. $(\theta_1,\theta_2,\theta_3)_1$ and $(\theta_1,\theta_2,\theta_3)_2$. The only pattern evident between the two solution sets is $\theta_{2\,\text{Elbow Up}} = -\theta_{2\,\text{Elbow Down}}$. Make a clear table of your multiple-solution results and be sure to use the circular check to prove both sets of solution angles satisfy the originally-given Cartesian pose.

i	***t*** $\pm$	**branch**	θ_1	θ_2	θ_3
1	−	elbow down	θ_{1_1}	θ_{2_1}	θ_{3_1}
2	+	elbow up	θ_{1_2}	$\theta_{2_2} = -\theta_{2_1}$	θ_{3_2}

The on-line EE/ME 4290/5290 Supplement presents an alternate to the tangent half-angle substitution method and another planar 3R IPK alternate solution method.

Planar 3R Robot IPK Graphical Solution

The planar 3R articulated robot Inverse Pose Kinematics solution may be performed **graphically**, by drawing the robot, determining the robot closure, and measuring the joint angle answers for both solution branches. This is an excellent method to validate your computer IPK results at a given snapshot. Here are the graphical IPK solution steps for the planar 3R robot.

- Draw the known ground link (establish a point for the origin of the {0} frame). Show the fixed X_0 and Y_0 unit vectors. Establish a reasonable length scale.

- Draw the given hand link, length L_3, from given hand point x_H, y_H, opposite to given hand angle ϕ. The proximal endpoint of L_3 opposite the given ϕ (towards the ground link for an acute angle ϕ) is the origin of {3}. Show the moving X_H and Y_H unit vectors at the distal end of L_3.

- Draw a circle of radius L_1, centered at the origin of {0}.

- Draw a circle of radius L_2, centered at the origin of {3}.

- These two circles intersect in general in two points. Connect the two solution branches, elbow up and elbow down. Measure the unknown values θ_1, θ_2, θ_3. Each angle is measured relative to the end of the previous link in each solution branch.

- There are two special cases.
    1. If the two circles do not intersect then the desired pose is outside of the reachable workspace for the robot. The analytical solutions for the angles in this case are imaginary.

    2. If the two circles intersect in a single point then the desired pose lies on the edge of the reachable workspace. In this case both solution sets (elbow up and elbow down) have degenerated to one real solution set (elbow straight out for $\theta_2 = 0$ and elbow folded back upon itself for $\theta_2 = 180^\circ$). This is a singular configuration (as presented later in the velocity section).

<u>Graphical Solution Figure</u>

Planar 3R Robot Inverse Pose Kinematics Examples

1) Given $\left[{}^{0}_{H}T\right]$ and $L_1 = 3, L_2 = 2, L_3 = 1$, calculate $(\theta_1, \theta_2, \theta_3)_i, i = 1, 2$.

$$\left[{}^{0}_{H}T\right] = \begin{bmatrix} 0.26 & -0.97 & 0 & 4.69 \\ 0.97 & 0.26 & 0 & 3.03 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{or, equivalently } \left\{{}^{0}X_{H}\right\} = \begin{Bmatrix} 4.69 \\ 3.03 \\ 75^\circ \end{Bmatrix}$$

$$\left[{}^{0}_{3}T\right] = \left[{}^{0}_{H}T\right]\left[{}^{H}_{3}T\right] = \left[{}^{0}_{H}T\right]\left[{}^{3}_{H}T\right]^{-1}$$

$$\left[{}^{0}_{3}T\right] = \begin{bmatrix} 0.26 & -0.97 & 0 & 4.69 \\ 0.97 & 0.26 & 0 & 3.03 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.26 & -0.97 & 0 & 4.43 \\ 0.97 & 0.26 & 0 & 2.06 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

That is, $\left\{{}^{0}X_{3}\right\} = \{x_3 \quad y_3 \quad \phi\}^T = \{4.43 \quad 2.06 \quad 75^\circ\}$ is the IPK input. The IPK answers (two $(\theta_1, \theta_2, \theta_3)$ solution sets) are given in the table below.

Answers (*deg*)

i	***t*** $\pm$	**branch**	θ_1	θ_2	θ_3
1	–	elbow down	15	25	35
2	+	elbow up	35	–25	65

The first line is expected since this is the same pose as the planar 3R robot Forward Pose Kinematics Example 1. Now we must check the second solution set to ensure it is correct (substitute the second solution set into Forward Pose Kinematics and ensure the same $\left[{}^{0}_{H}T\right]$ is obtained – this is the so-called **circular check**).

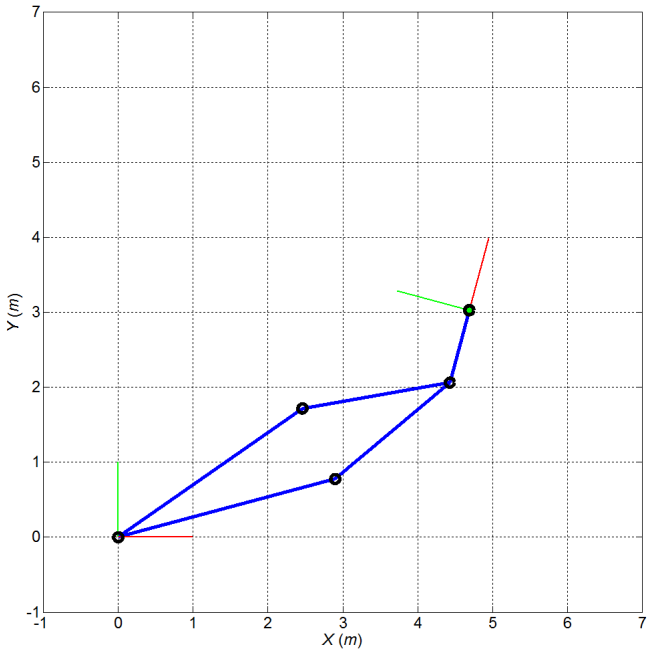
2) Given $\left[{}^{0}_{H}T\right]$ and $L_1 = 3, L_2 = 2, L_3 = 1$, calculate $(\theta_1, \theta_2, \theta_3)_i, i = 1, 2$.

$$\left[{}^{0}_{H}T\right] = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 6 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \left\{{}^{0}X_{H}\right\} = \begin{Bmatrix} 0 \\ 6 \\ 90^\circ \end{Bmatrix} \qquad \left[{}^{0}_{3}T\right] = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 5 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \left\{{}^{0}X_{3}\right\} = \begin{Bmatrix} 0 \\ 5 \\ 90^\circ \end{Bmatrix}$$

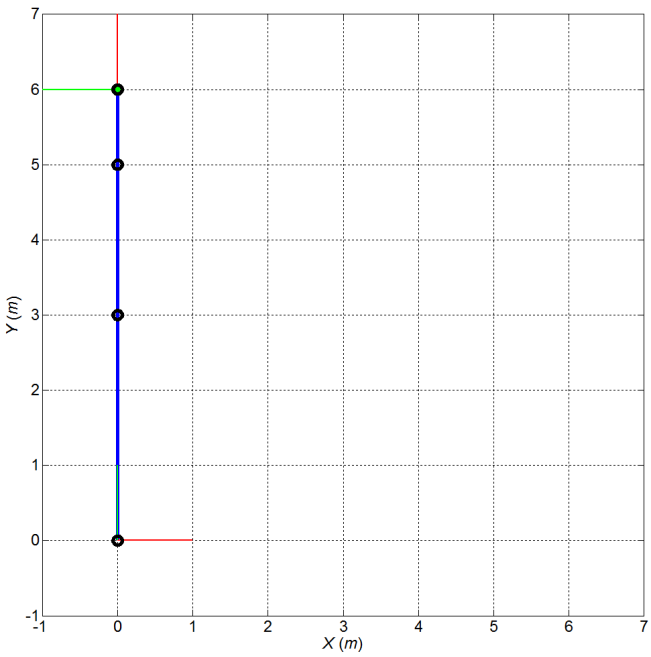
Answers (*deg*)

i	***t*** $\pm$	**branch**	θ_1	θ_2	θ_3
1	$\pm$	same	90	0	0

There is only one solution branch (neither elbow down nor elbow up – why?).



Planar 3R IPK Example 1
(*X* red, *Y* green, *Z* out of page)



Planar 3R IPK Example 2
(elbow up and down are the same)

3) Given $\left[{}^{0}_{H}T\right]$ and $L_1 = 3, L_2 = 2, L_3 = 1$, calculate $(\theta_1, \theta_2, \theta_3)_i, i = 1, 2$.

$$\left[{}^{0}_{H}T\right] = \begin{bmatrix} 0.87 & -0.50 & 0 & 4.00 \\ 0.50 & 0.87 & 0 & 6.93 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \left\{{}^{0}X_{H}\right\} = \begin{Bmatrix} 4.00 \\ 6.93 \\ 30^{\circ} \end{Bmatrix}$$

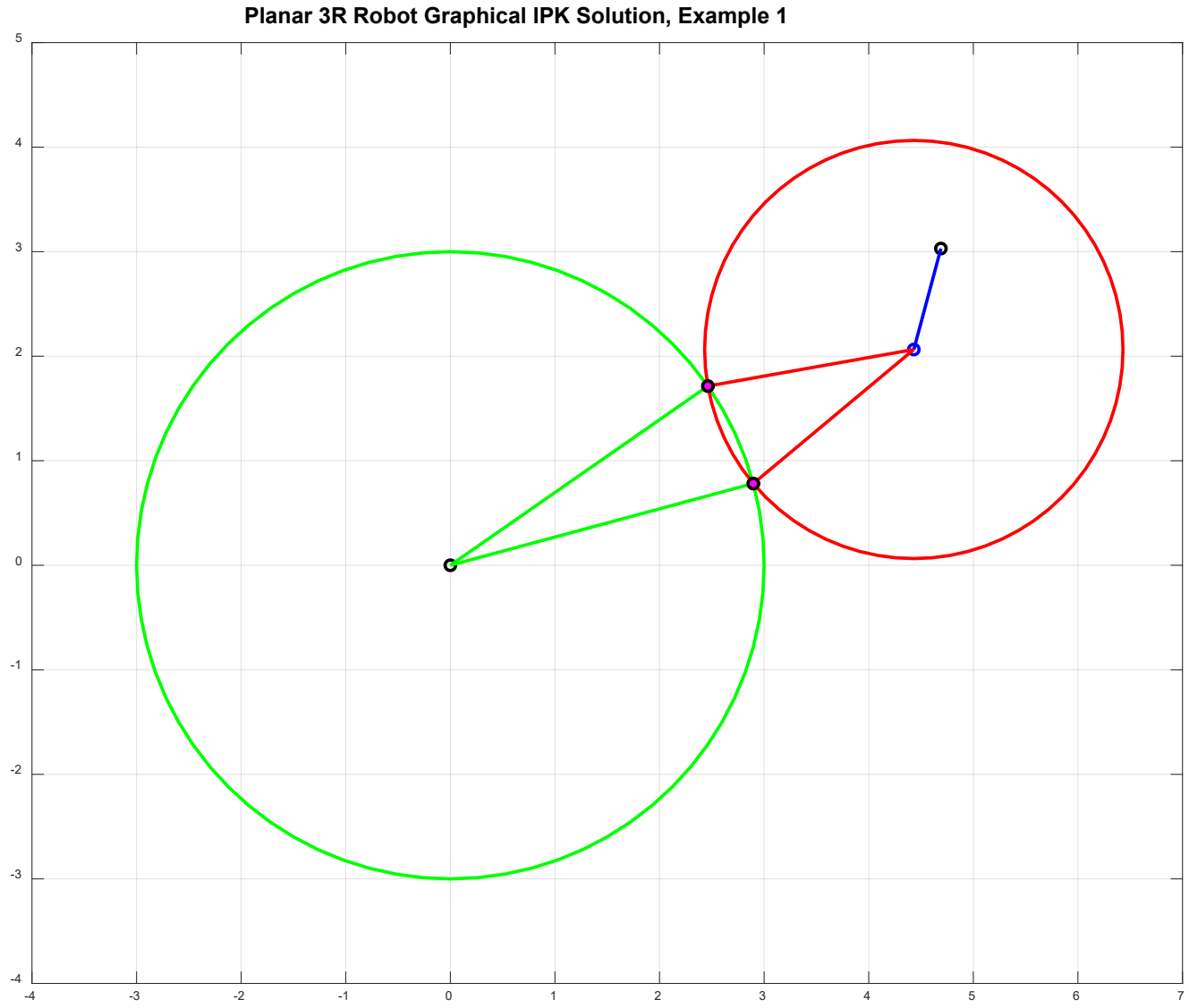
$$\left[{}^{0}_{3}T\right] = \begin{bmatrix} 0.87 & -0.50 & 0 & 3.13 \\ 0.50 & 0.87 & 0 & 6.43 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \left\{{}^{0}X_{3}\right\} = \begin{Bmatrix} 3.13 \\ 6.43 \\ 30^{\circ} \end{Bmatrix}$$

This case is impossible (no solution exists). The answers (not shown) are imaginary (complex numbers) which means the desired pose is outside of the manipulator reachable workspace. That is, the vector length of the *XY* components of $\left\{{}^{0}X_{H}\right\}$, 8.00, exceeds the straight-out ($[\theta_1 \quad \theta_2 = 0 \quad \theta_3 = 0]$) sum $L_1 + L_2 + L_3 = 6$.

Inverse-Pose-Based Robot Controller Block Diagram:

Here is the planar 3R IPK graphical solution for the following given inputs:

$$L_1 = 3, L_2 = 2, L_3 = 1 \qquad \left[{}^{0}_{H}T\right] = \begin{bmatrix} 0.26 & -0.97 & 0 & 4.69 \\ 0.97 & 0.26 & 0 & 3.03 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{i.e. } \left\{{}^{0}X_{H}\right\} = \begin{Bmatrix} 4.69 \\ 3.03 \\ 75^{\circ} \end{Bmatrix}$$



This is the same numerical example from above.

Plan3RIPKGraphical.m

Planar 3R Robot Inverse Pose Kinematics Trajectory Example

More interesting than snapshot examples is making a virtual robot motion trajectory. In this example, initial and final required poses are given for the Planar 3R Robot, and IPK will be calculated at many steps in-between, yielding the required joint angles for control of the robot to achieve the trajectory. Note this trajectory is such that the Cartesian motions (x_H, y_H, ϕ) will all be chopped into equal segments, providing straight-line motions in the Cartesian space.

Given: $L_1 = 3, \quad L_2 = 2, \quad L_3 = 1$ m

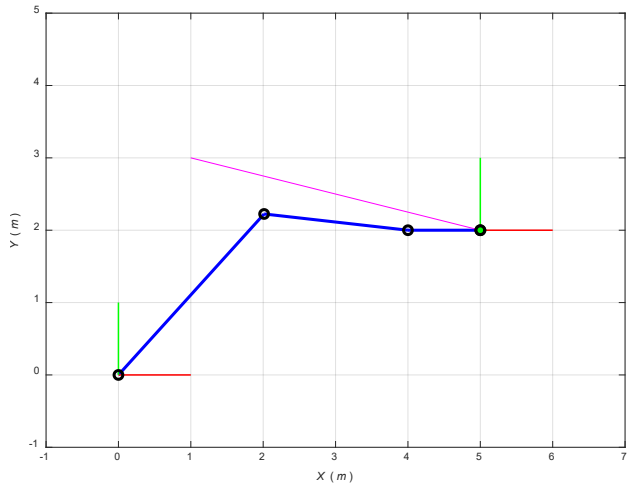
initial pose: $\{{}^0X_H\} = \{5 \quad 2 \quad 0^\circ\}$ m and deg

final pose: $\{{}^0X_H\} = \{1 \quad 3 \quad 90^\circ\}$

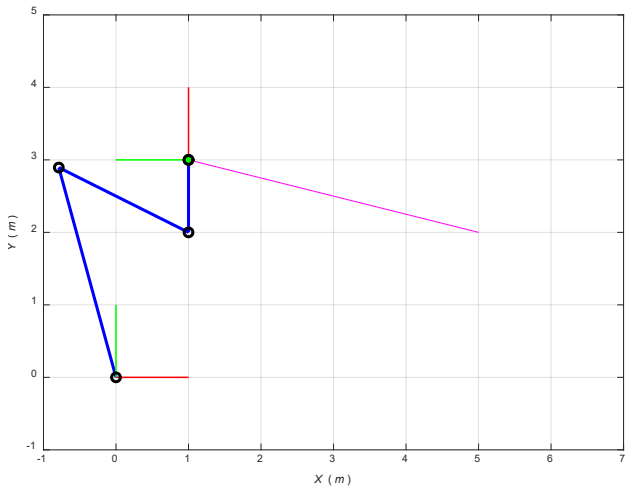
Calculate:

the required time histories of $(\theta_1, \theta_2, \theta_3)$, for the elbow-up branch.

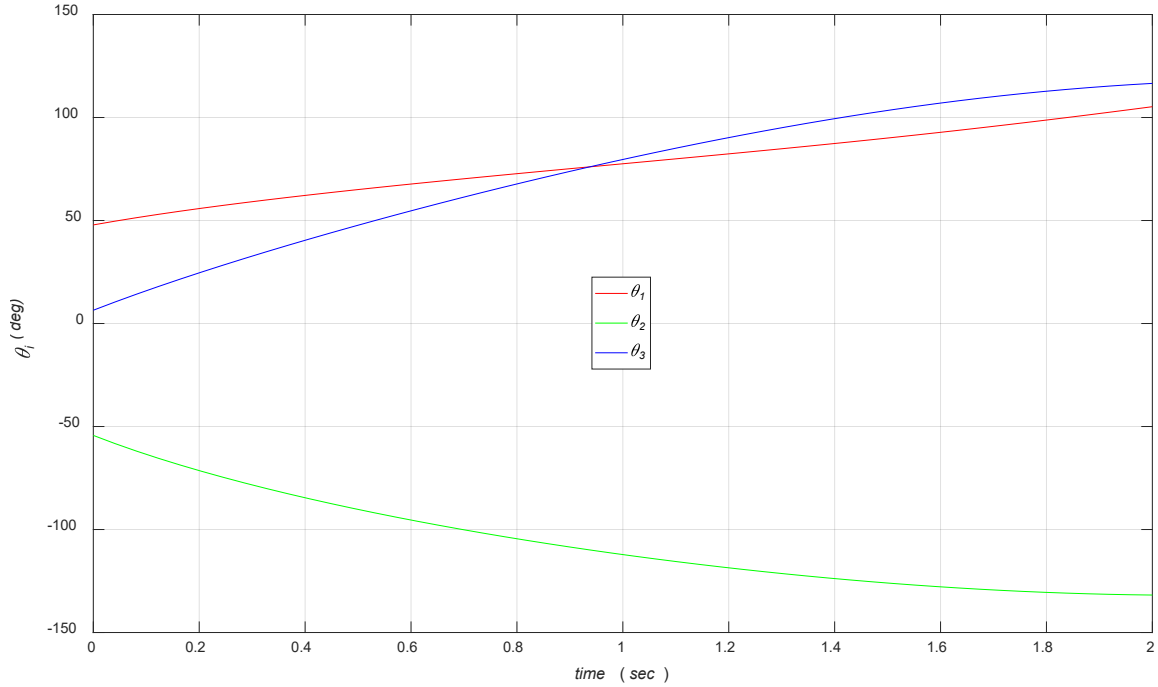
Associating a 2 sec time range with this motion, and using $N = 50$ steps, the graphics below show the simulated results.



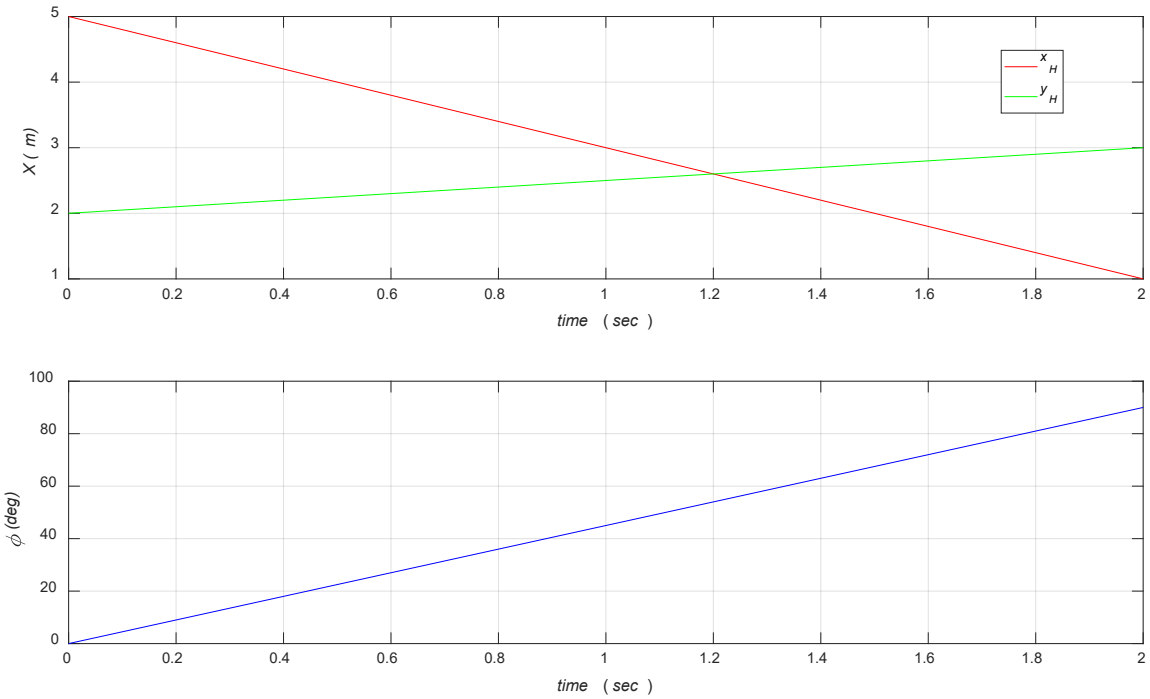
Initial Pose, elbow up



Final Pose, elbow up

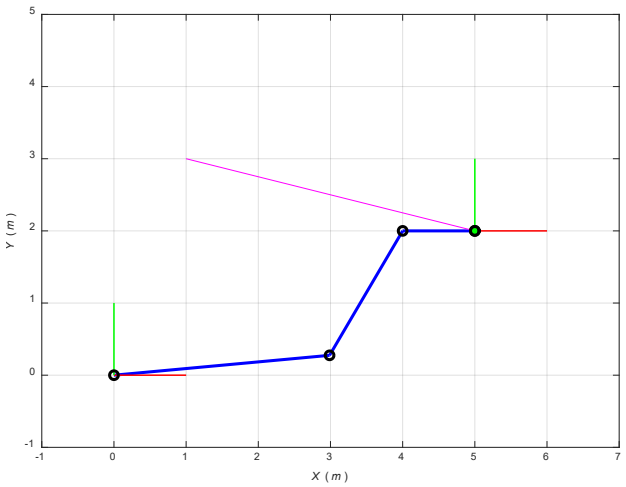


Required Joint Angles, elbow up

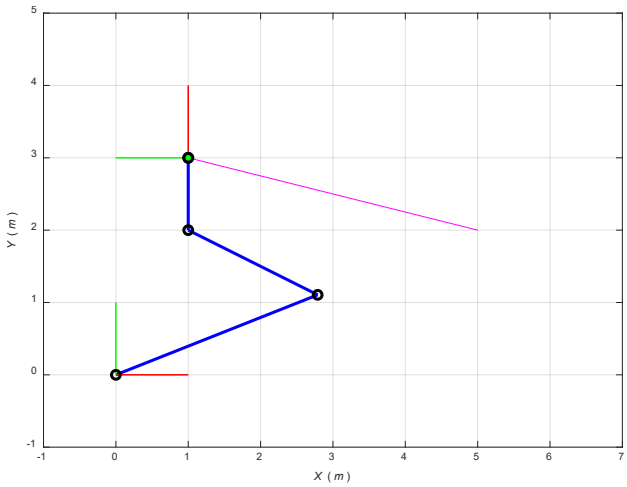


Commanded Cartesian Motion

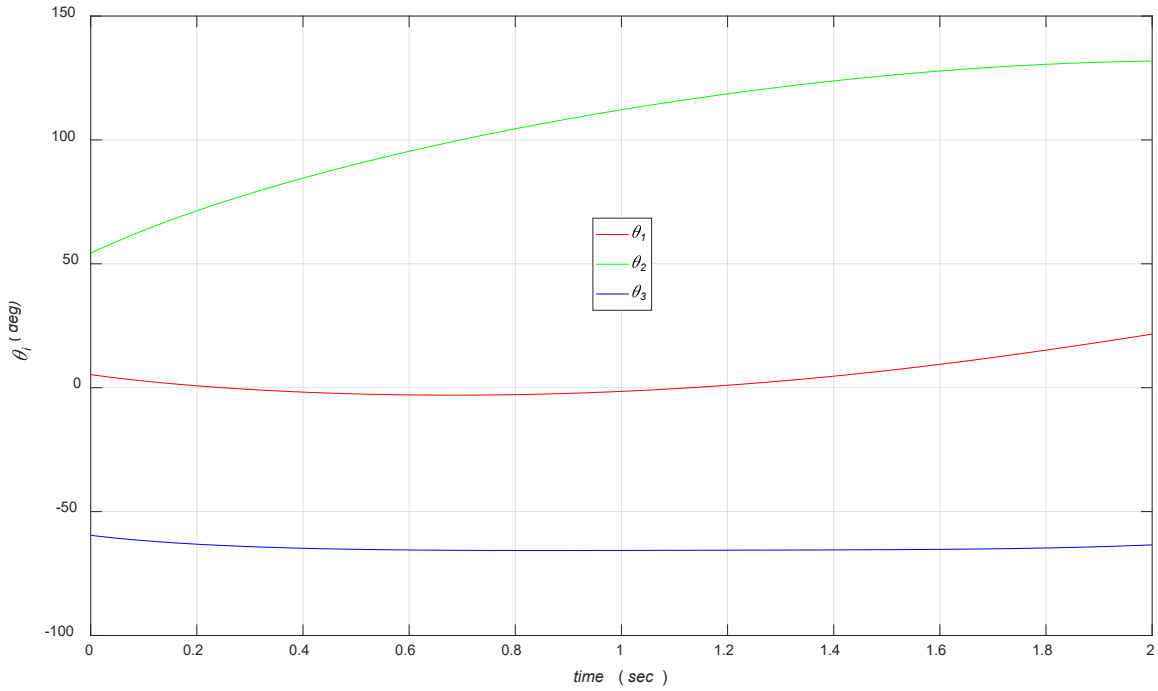
Now let us repeat this example exactly, but for the elbow-down solution branch.



Initial Pose, elbow down



Final Pose, elbow down



Required Joint Angles, elbow down

Note the commanded Cartesian motion is identical to the case show above. What is the symmetry of the required elbow joint angles of the elbow-down solution branch vs. the elbow-up solution branch?

6. Trajectory Generation

A **robot trajectory** is the commanded and/or actual time history of robot *position*, *velocity*, and *acceleration* motion. Trajectories are at the joint and/or Cartesian levels (robot kinematics relates these two levels). One can command and control Cartesian trajectories of $X = \{x \quad y \quad z \quad \gamma \quad \beta \quad \alpha\}^T$ via joint trajectories of $\Theta = \{\theta_1 \quad \theta_2 \quad \theta_3 \quad \theta_4 \quad \theta_5 \quad \theta_6\}^T$, for a general 6-dof 6R spatial robot.

Many robotics applications require motion only at discrete points.

- spot welding
- assembly
- pick and place operations

How can the robot be moved smoothly, safely, and efficiently between the required discrete poses? Other robotics applications require commanded trajectory motion.

- arc welding
- spray painting
- parts handling from moving conveyors

How can trajectories be commanded to satisfy motion requirements smoothly, safely, and efficiently? To both questions, there are ***infinite*** possible answers.

- straight-line motion
- constant velocity motion (resolved-rate control will be presented later)
- avoid obstacles
- shortest distance (same as straight-line motion? – Cartesian only)
- shortest time

Even with one specific choice (e.g. avoid obstacles), there are still ***infinite*** possible trajectories. In **Trajectory Generation**, we will focus on a subset of the possible trajectory generation methods: We will assume the robot control architecture is Inverse Pose Kinematics Control and we will perform **smooth** Joint-Space Trajectory Generation to move between discrete commanded Cartesian poses.

Inverse Pose Kinematics

Calculate the sets of *n* joint angles (for R, or length for P) required to place the robot end-effector in a given set of required Cartesian poses. We will consider just an initial and final pose; more can be included by making the final pose the new initial and adding a new final pose. The required poses are denoted as **path points** (zero velocity), while intermediate poses to pass through are denoted **via points**.

For the initial and final poses, use the Inverse Pose Kinematics solution to determine valid sets of joint angles to satisfy these Cartesian pose commands. Now, how do we move **smoothly** between the initial and final joint angle sets? We could just define a lot of intermediate Cartesian poses and calculate IPK many more times, but this is computationally inefficient and there is a better method.

Joint-Space Trajectory Generation

We now present five related methods for joint-space trajectory generation. We assume that we need to move *n* joint angles independently (but simultaneously) from an initial set to a final set, corresponding to required pose commands from the IPK solution. So, we will focus on joint angle (or length) trajectory generation for one joint only – the results are readily extendable to all *n* joint angles.

6.1 Third-Order Polynomial

Using a **smooth motion** criterion, we must ensure the joint angle position and velocity functions are continuous. Further, the initial and final joint angles must match the given angles and the initial and final joint rates must be zero for starting and stopping at rest. There are many ways to provide continuous position and velocity functions but polynomials are a good choice. In the figure below it can be shown that there are infinite paths from the initial joint angle to the final joint angle for the i^{th} joint:

The four smooth motion joint constraints are:

Let us make a polynomial fit – with four constraints, what order polynomial is required? The general functions for joint *i* angle and joint *i* angular velocity are:

We obtain four linear equations in the four unknown polynomial coefficients a_i, $i = 0,1,2,3$. The 2x2 matrix/vector equation to solve for the remaining two coupled unknowns is:

Single Third-Order Polynomial Solution

$$\theta(t) = a_3 t^3 + a_2 t^2 + a_1 t + a_0$$
$$\dot{\theta}(t) = 3a_3 t^2 + 2a_2 t + a_1$$
$$\ddot{\theta}(t) = 6a_3 t + 2a_2$$
$$\dddot{\theta}(t) = 6a_3$$

$$a_0 = \theta_S$$
$$a_1 = 0$$
$$a_2 = \frac{3}{t_F^2}(\theta_F - \theta_S)$$
$$a_3 = -\frac{2}{t_F^3}(\theta_F - \theta_S)$$

The 2x2 coefficient matrix determinant is t_F^4, so as long as $t_F \neq 0$, this solution always exists.

Example – Third-order Polynomial

Use a third-order polynomial fit for smooth joint space trajectory generation, for one joint only.

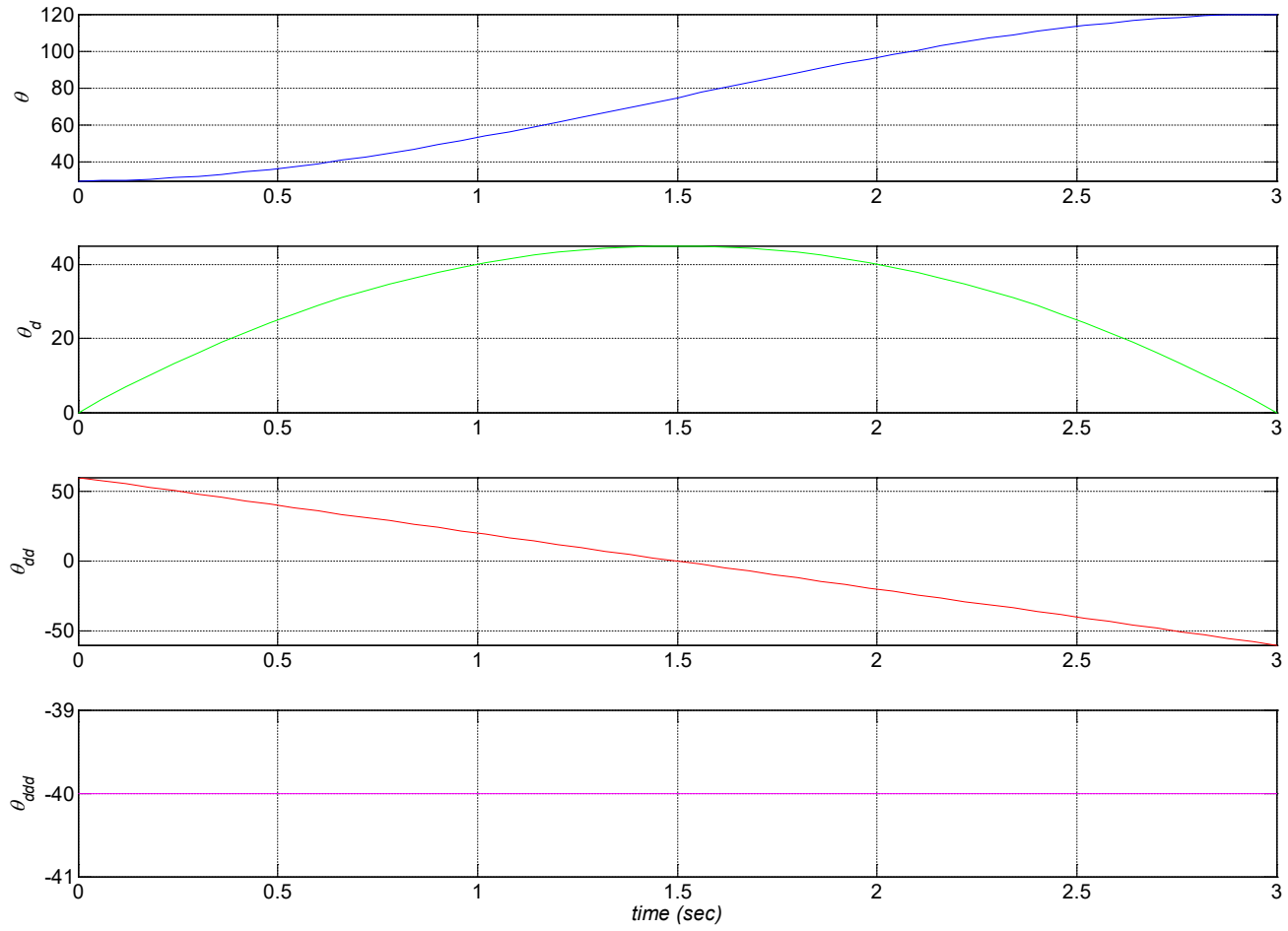
Given: $\theta_S = 30^\circ$, $\theta_F = 120^\circ$, $t_f = 3$ *sec*

Find: $\theta(t)$ for smooth trajectory generation. Plot $\theta(t), \dot{\theta}(t), \ddot{\theta}(t), \dddot{\theta}(t)$

Result

$$\theta(t) = -6.67t^3 + 30t^2 + 30$$
$$\dot{\theta}(t) = -20t^2 + 60t$$
$$\ddot{\theta}(t) = -40t + 60$$
$$\dddot{\theta}(t) = -40$$

(Note *deg* units are used throughout)



The *jerk* looks fine at first, but there are actually infinite spikes in *jerk* at the start and end. Why?

Once $\theta(t)$ is known, what happens next? One must discretize the $\theta(t)$ function at the controller time step size and send these numerical values to the single robot joint controller at the proper times. Again, this is done *n* times independently but simultaneously for an *n*-dof serial robot.

6.2 Fifth-Order Polynomial

Now, the third-order polynomial will provide smooth motion with zero velocity at the start and end. But the resulting *jerk* has infinite spikes at the start and end (because there is an impossible finite jump in acceleration at the start and finish times). How can we avoid this?

Higher-order polynomial; Use the same four smooth motion joint constraints, plus two more constraints to avoid infinite *jerk* spikes.

Let us make a polynomial fit – with six constraints, what order polynomial is required? The general functions for joint *i* angle, joint *i* angular velocity, and joint *i* acceleration are:

We obtain six linear equations in the six unknown polynomial coefficients a_i, $i = 0,1,2,3,4,5$.

The remaining coupled unknown polynomial coefficients' linear equations may be expressed by the following 3x3 matrix/vector equation.

$$\begin{bmatrix} t_F^3 & t_F^4 & t_F^5 \\ 3t_F^2 & 4t_F^3 & 5t_F^4 \\ 6t_F & 12t_F^2 & 20t_F^3 \end{bmatrix} \begin{Bmatrix} a_3 \\ a_4 \\ a_5 \end{Bmatrix} = \begin{Bmatrix} \theta_F - \theta_S \\ 0 \\ 0 \end{Bmatrix} \qquad \begin{bmatrix} 1 & t_F & t_F^2 \\ 3 & 4t_F & 5t_F^2 \\ 6 & 12t_F & 20t_F^2 \end{bmatrix} \begin{Bmatrix} a_3 \\ a_4 \\ a_5 \end{Bmatrix} = \begin{Bmatrix} (\theta_F - \theta_S)/t_F^3 \\ 0 \\ 0 \end{Bmatrix}$$

Single Fifth-Order Polynomial Solution

$$\begin{aligned} a_0 &= \theta_S \\ a_1 &= 0 \\ a_2 &= 0 \\ a_3 &= \frac{10}{t_F^3}(\theta_F - \theta_S) \\ a_4 &= -\frac{15}{t_F^4}(\theta_F - \theta_S) \\ a_5 &= \frac{6}{t_F^5}(\theta_F - \theta_S) \end{aligned}$$

$$\begin{aligned} \theta(t) &= a_5t^5 + a_4t^4 + a_3t^3 + a_2t^2 + a_1t + a_0 \\ \dot{\theta}(t) &= 5a_5t^4 + 4a_4t^3 + 3a_3t^2 + 2a_2t + a_1 \\ \ddot{\theta}(t) &= 20a_5t^3 + 12a_4t^2 + 6a_3t + 2a_2 \\ \dddot{\theta}(t) &= 60a_5t^2 + 24a_4t + 6a_3 \end{aligned}$$

$$\begin{aligned} \theta(t) &= a_5t^5 + a_4t^4 + a_3t^3 + \theta_S \\ \dot{\theta}(t) &= 5a_5t^4 + 4a_4t^3 + 3a_3t^2 \\ \ddot{\theta}(t) &= 20a_5t^3 + 12a_4t^2 + 6a_3t \\ \dddot{\theta}(t) &= 60a_5t^2 + 24a_4t + 6a_3 \end{aligned}$$

The 3x3 coefficient matrix determinant is $2t_F^9$ ($2t_F^3$), so as long as $t_F \neq 0$, this solution always exists.

Example – Fifth-Order Polynomial

Use a fifth-order polynomial fit for smooth joint space trajectory generation, plus finite *jerk*, for one joint only.

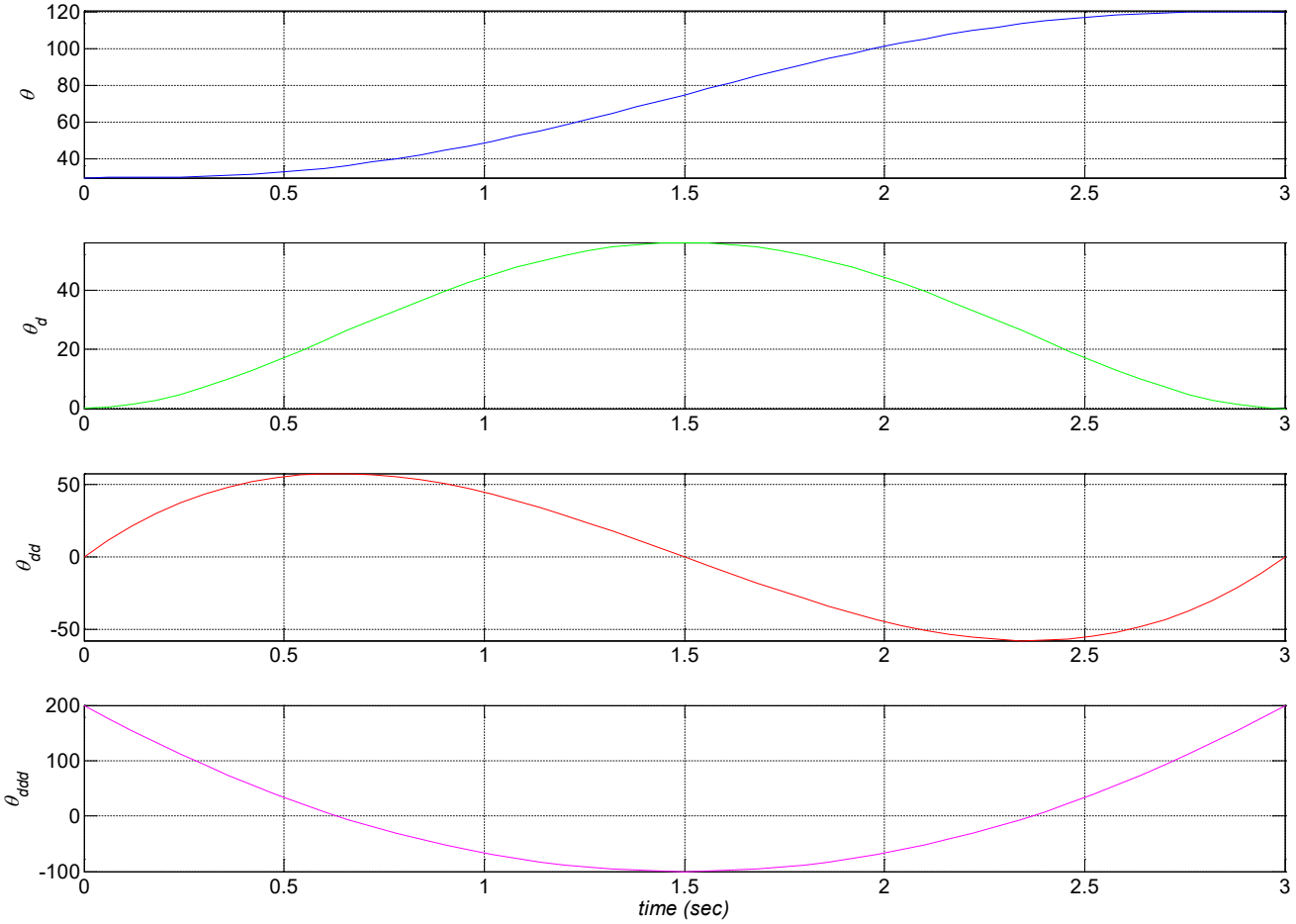
Given: $\theta_S = 30^\circ$, $\theta_F = 120^\circ$, $t_f = 3$ *sec*

Find: $\theta(t)$ for smooth trajectory generation and finite *jerk*. Plot $\theta(t), \dot{\theta}(t), \ddot{\theta}(t), \dddot{\theta}(t)$

Result

$$\theta(t) = 2.22t^5 - 16.67t^4 + 33.33t^3 + 30$$
$$\dot{\theta}(t) = 11.11t^4 - 66.67t^3 + 100t^2$$
$$\ddot{\theta}(t) = 44.44t^3 - 200t^2 + 200t$$
$$\dddot{\theta}(t) = 133.33t^2 - 400t + 200$$

(Note *deg* units are used throughout)



Now the *jerk* has a discontinuous jump at the start and end, but it is a finite discontinuity. This fifth-order polynomial is thus superior to the previous third-order polynomial. The joint acceleration starts and ends at zero, so there is no impossible jump in acceleration anywhere during the motion.

What is the cost of the 5th-order polynomial compared to the 3rd-order? (slightly-higher velocity and significantly-higher jerk.) But this is well worth the cost since the original jerk had infinite spikes.

6.3 Two Third-Order Polynomials with a Via Point

Now, many motion requirements for real-world tasks may require the robot to pass through (or near) a pose without stopping. Let us consider a case with fixed initial and final **path points** (where the robot must start and stop with zero velocity) and one intermediate **via point** where the robot needn't stop. This approach is often used for obstacle avoidance. The Cartesian and joint space figures are shown below.

Two third-order polynomials will provide smooth motion with continuous position and velocity and zero velocity at the start and end. The same four smooth motion joint constraints from earlier apply, but now for two different third-order polynomials.

$$\theta_1(t) = a_{13}t^3 + a_{12}t^2 + a_{11}t + a_{10} \qquad \theta_2(t) = a_{23}t^3 + a_{22}t^2 + a_{21}t + a_{20}$$
$$\dot{\theta}_1(t) = 3a_{13}t^2 + 2a_{12}t + a_{11} \qquad \dot{\theta}_2(t) = 3a_{23}t^2 + 2a_{22}t + a_{21}$$
$$\ddot{\theta}_1(t) = 6a_{13}t + 2a_{12} \qquad \ddot{\theta}_2(t) = 6a_{23}t + 2a_{22}$$
$$\dddot{\theta}_1(t) = 6a_{13} \qquad \dddot{\theta}_2(t) = 6a_{23}$$

$$\theta_1(0) = \theta_S \qquad \theta_2(t_2) = \theta_F$$
$$\dot{\theta}_1(0) = 0 \qquad \dot{\theta}_2(t_2) = 0$$

It is convenient to shift the time axis so that the $\theta_2(t)$ polynomial has zero time starting at the via time. t_1 is the end time of the first range ($t_1 = t_V$) and t_2 is the relative end time of the second range ($t_2 = t_F - t_V$). So for polynomial 1 t goes from $0 \to t_1$ and for polynomial 2 t goes from $0 \to t_2$. To make one continuous plot over time for these two polynomials, the student must take special care.

We need four more constraints since we now have two third-order polynomials. We force the first polynomial to end at the via angle θ_V and the second polynomial to start at the via angle θ_V (two constraints). We also ensure the velocities match at the via point (they need not go to zero, they must only match – this is one constraint). We also ensure the accelerations match at the via point (they need not go to zero, only match – this is also one constraint). This will ensure that the *jerk* stays finite during this via point transition between the two polynomials. These four additional constraints are:

$$\theta_1(t_1) = \theta_V \qquad \dot{\theta}_1(t_1) = \dot{\theta}_2(0)$$
$$\theta_2(0) = \theta_V \qquad \ddot{\theta}_1(t_1) = \ddot{\theta}_2(0)$$

There are eight linear equations in eight unknowns (the two sets of polynomial coefficients a_{ij}, i = 1,2 and j = 0,1,2,3).

$$\theta_1(0) = \theta_S = a_{10}$$
$$\dot{\theta}_1(0) = 0 = a_{11}$$
$$\theta_2(t_2) = \theta_F = a_{23}t_2^3 + a_{22}t_2^2 + a_{21}t_2 + a_{20}$$
$$\dot{\theta}_2(t_2) = 0 = 3a_{23}t_2^2 + 2a_{22}t_2 + a_{21}$$
$$\theta_1(t_1) = \theta_V = a_{13}t_1^3 + a_{12}t_1^2 + a_{10}$$
$$\theta_2(0) = \theta_V = a_{20}$$
$$3a_{13}t_1^2 + 2a_{12}t_1 = a_{21}$$
$$6a_{13}t_1 + 2a_{12} = 2a_{22}$$

The remaining coupled unknown polynomial coefficients' linear equations may be expressed by the following 5x5 matrix/vector equation.

$$\begin{bmatrix} t_1^2 & t_1^3 & 0 & 0 & 0 \\ 0 & 0 & t_2 & t_2^2 & t_2^3 \\ 0 & 0 & 1 & 2t_2 & 3t_2^2 \\ 2t_1 & 3t_1^2 & -1 & 0 & 0 \\ 2 & 6t_1 & 0 & -2 & 0 \end{bmatrix} \begin{Bmatrix} a_{12} \\ a_{13} \\ a_{21} \\ a_{22} \\ a_{23} \end{Bmatrix} = \begin{Bmatrix} \theta_V - \theta_S \\ \theta_F - \theta_V \\ 0 \\ 0 \\ 0 \end{Bmatrix} \qquad \begin{bmatrix} 1 & t_1 & 0 & 0 & 0 \\ 0 & 0 & 1 & t_2 & t_2^2 \\ 0 & 0 & 1 & 2t_2 & 3t_2^2 \\ 2t_1 & 3t_1^2 & -1 & 0 & 0 \\ 2 & 6t_1 & 0 & -2 & 0 \end{bmatrix} \begin{Bmatrix} a_{12} \\ a_{13} \\ a_{21} \\ a_{22} \\ a_{23} \end{Bmatrix} = \begin{Bmatrix} (\theta_V - \theta_S)/t_1^2 \\ (\theta_F - \theta_V)/t_2 \\ 0 \\ 0 \\ 0 \end{Bmatrix}$$

The 5x5 coefficient matrix determinant is $4t_1^3t_2^3(t_1 + t_2)$ ($4t_1t_2^2(t_1 + t_2)$), so as long as neither $t_1 = 0$ nor $t_2 = 0$, this solution always exists. Of course, $t_1 + t_2 \neq 0$ for positive time.

Two Third-Order Polynomials with Via Point Solution

Three of the coefficients appear immediately as seen above. The remaining five unknowns are solved from the above 5x5 matrix/vector system of equations.

For the special case of the via time being half the final time ($t_1 = t_2 = t_F/2 = t_V = T$), the analytical solution is:

$$a_{10} = \theta_S$$
$$a_{11} = 0$$
$$a_{20} = \theta_V$$
$$a_{12} = \frac{-3(\theta_F - \theta_V) + 9(\theta_V - \theta_S)}{4T^2}$$
$$a_{13} = \frac{3(\theta_F - \theta_V) - 5(\theta_V - \theta_S)}{4T^3}$$
$$a_{21} = \frac{3(\theta_F - \theta_S)}{4T}$$
$$a_{22} = \frac{3(\theta_F - \theta_V) - 3(\theta_V - \theta_S)}{2T^2}$$
$$a_{23} = \frac{-5(\theta_F - \theta_V) + 3(\theta_V - \theta_S)}{4T^3}$$

Example – Two Third-Order Polynomials with Via Point

Use two third-order polynomials for smooth joint space trajectory generation, plus motion through an additional via point (no need to stop at the via point, just ensure smooth motion), for one joint only.

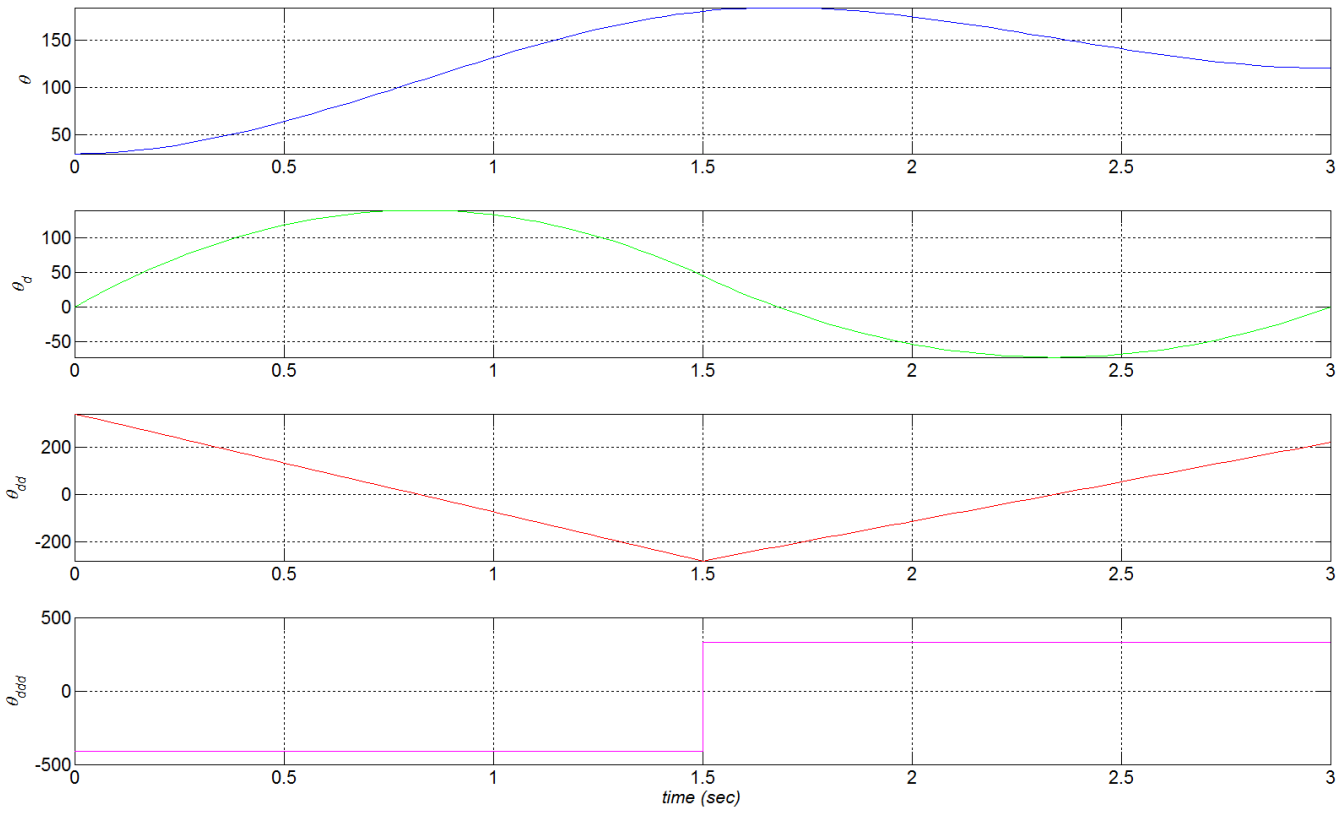
Given: $\theta_S = 30^\circ$, $\theta_V = 180^\circ$, $\theta_F = 120^\circ$, $t_V = 1.5, t_F = 3$ $(t_1 = t_2 = 1.5)$ *sec*

Find: $\theta_1(t)$ and $\theta_2(t)$ for smooth trajectory generation and continuous motion through a via point.
Plot $\theta(t), \dot{\theta}(t), \ddot{\theta}(t), \dddot{\theta}(t)$.

Result

$$\theta_1(t) = -68.89t^3 + 170t^2 + 30 \qquad \theta_2(t) = 55.56t^3 - 140t^2 + 45t + 180$$
$$\dot{\theta}_1(t) = -206.67t^2 + 340t \qquad \dot{\theta}_2(t) = 166.67t^2 - 280t + 45$$
$$\ddot{\theta}_1(t) = -413.33t + 340 \qquad \ddot{\theta}_2(t) = 333.33t - 280$$
$$\dddot{\theta}_1(t) = -413.33 \qquad \dddot{\theta}_2(t) = 333.33$$

(Note *deg* units are used throughout)



Note that the joint angle passes through the via point $\theta_V = 180^\circ$ and keeps going for a brief time – this is because the velocity is still positive beyond the first 1.5 *sec*. The maximum angle is $\theta_{MAX} = 183.9^\circ$, occurring at t = 1.68 sec. At θ_{MAX} the angular velocity goes to zero (the slope of the angle is zero at that point since the angle is changing direction). Also note that the velocity term in $\theta_2(t)$, the t coefficient a_{21}, is non-zero since the velocity does not need to go to zero at this via point. Further note that the velocity and acceleration have been successfully matched at the 1.5 *sec* via point transition as required. Note that the *jerk* is not matched at the transition (the slope of the two polynomial accelerations are the same magnitude but different signs), but it remains finite since the acceleration is matched.

Also, the *jerk* still has an infinite spike at the start and end as we saw in the single third-order polynomial example, which is **unacceptable**.

The matching of two separate polynomials at the via point is a lot of work. How can we improve upon this with only one polynomial? See the next two examples – this work is original, developed at Ohio University by the author and published at the 2011 ASME IDETC in Washington DC.

6.4 Single Fourth-Order Polynomial with a Via Point

We can achieve the same goals as the two third-order polynomials meeting at a via point much more simply. We use a single polynomial, forced to go through a via point. Here are the constraints for meeting the required angles with smooth motion (since we use a single continuous polynomial, the velocity and accelerations are guaranteed to match at the via point). This is original work presented at the 2011 ASME IDETC in Washington DC[6].

Note in this case since there is only one time range, it is convenient to treat all times as absolute, rather than relative as when we did matching of two third-order polynomials. With five constraints we can use a single fourth-order polynomial.

$$\theta(t) = a_4 t^4 + a_3 t^3 + a_2 t^2 + a_1 t + a_0$$
$$\dot{\theta}(t) = 4a_4 t^3 + 3a_3 t^2 + 2a_2 t + a_1$$
$$\ddot{\theta}(t) = 12a_4 t^2 + 6a_3 t + 2a_2$$
$$\dddot{\theta}(t) = 24a_4 t + 6a_3$$

Similar to previous derivations, from two constraints we immediately have:

$$a_0 = \theta_S \qquad\qquad a_1 = 0$$

The remaining coupled unknown polynomial coefficients' linear equations may be expressed by the following 3x3 matrix/vector equation.

$$\begin{bmatrix} t_V^2 & t_V^3 & t_V^4 \\ t_F^2 & t_F^3 & t_F^4 \\ 2t_F & 3t_F^2 & 4t_F^3 \end{bmatrix} \begin{Bmatrix} a_2 \\ a_3 \\ a_4 \end{Bmatrix} = \begin{Bmatrix} \theta_V - \theta_S \\ \theta_F - \theta_S \\ 0 \end{Bmatrix} \qquad \begin{bmatrix} 1 & t_V & t_V^2 \\ 1 & t_F & t_F^2 \\ 2 & 3t_F & 4t_F^2 \end{bmatrix} \begin{Bmatrix} a_2 \\ a_3 \\ a_4 \end{Bmatrix} = \begin{Bmatrix} (\theta_V - \theta_S)/t_V^2 \\ (\theta_F - \theta_S)/t_F^2 \\ 0 \end{Bmatrix}$$

[6] R.L. Williams II, 2011, "Improved Robotics Joint-Space Trajectory Generation with Via Point", Proceedings of the ASME IDETC/CIE, Washington DC, August 28-31, 2011, DETC2011-47592.

The 3x3 coefficient matrix determinant is $t_V^2 t_F^4 (t_F - t_V)^2$ (the right one is $t_F (t_F - t_V)^2$), so as long as neither $t_V = 0$ nor $t_F = 0$, this solution always exists. $t_F > t_V$ is required in this scenario, so that term cannot be zero.

Single Fourth-Order Polynomial with Via Point Analytical Solution

$$
\begin{aligned}
a_0 &= \theta_S \\
a_1 &= 0 \\
a_2 &= \frac{1}{(t_F - t_V)^2}\left[\frac{(\theta_V - \theta_S)t_F^2}{t_V^2} - \frac{(\theta_F - \theta_S)(4t_F - 3t_V)t_V}{t_F^2}\right] \\
a_3 &= \frac{2}{(t_F - t_V)^2}\left[-\frac{(\theta_V - \theta_S)t_F}{t_V^2} + \frac{(\theta_F - \theta_S)(2t_F^2 - t_V^2)}{t_F^3}\right] \\
a_4 &= \frac{1}{(t_F - t_V)^2}\left[\frac{(\theta_V - \theta_S)}{t_V^2} - \frac{(\theta_F - \theta_S)(3t_F - 2t_V)}{t_F^3}\right]
\end{aligned}
$$

For the special case of the via time being half the final time ($t_V = t_F/2$), the analytical solution is:

$$
\begin{aligned}
a_0 &= \theta_S \\
a_1 &= 0 \\
a_2 &= \frac{1}{t_F^2}\left[16(\theta_V - \theta_S) - 5(\theta_F - \theta_S)\right] \\
a_3 &= -\frac{2}{t_F^3}\left[16(\theta_V - \theta_S) - 7(\theta_F - \theta_S)\right] \\
a_4 &= \frac{8}{t_F^4}\left[2(\theta_V - \theta_S) - (\theta_F - \theta_S)\right]
\end{aligned}
$$

Example – Single Fourth-Order Polynomial with Via Point

Use a single fourth-order polynomial for smooth joint space trajectory generation, plus motion through an additional via point (no need to stop at the via point, only ensure smooth motion), for one joint only.

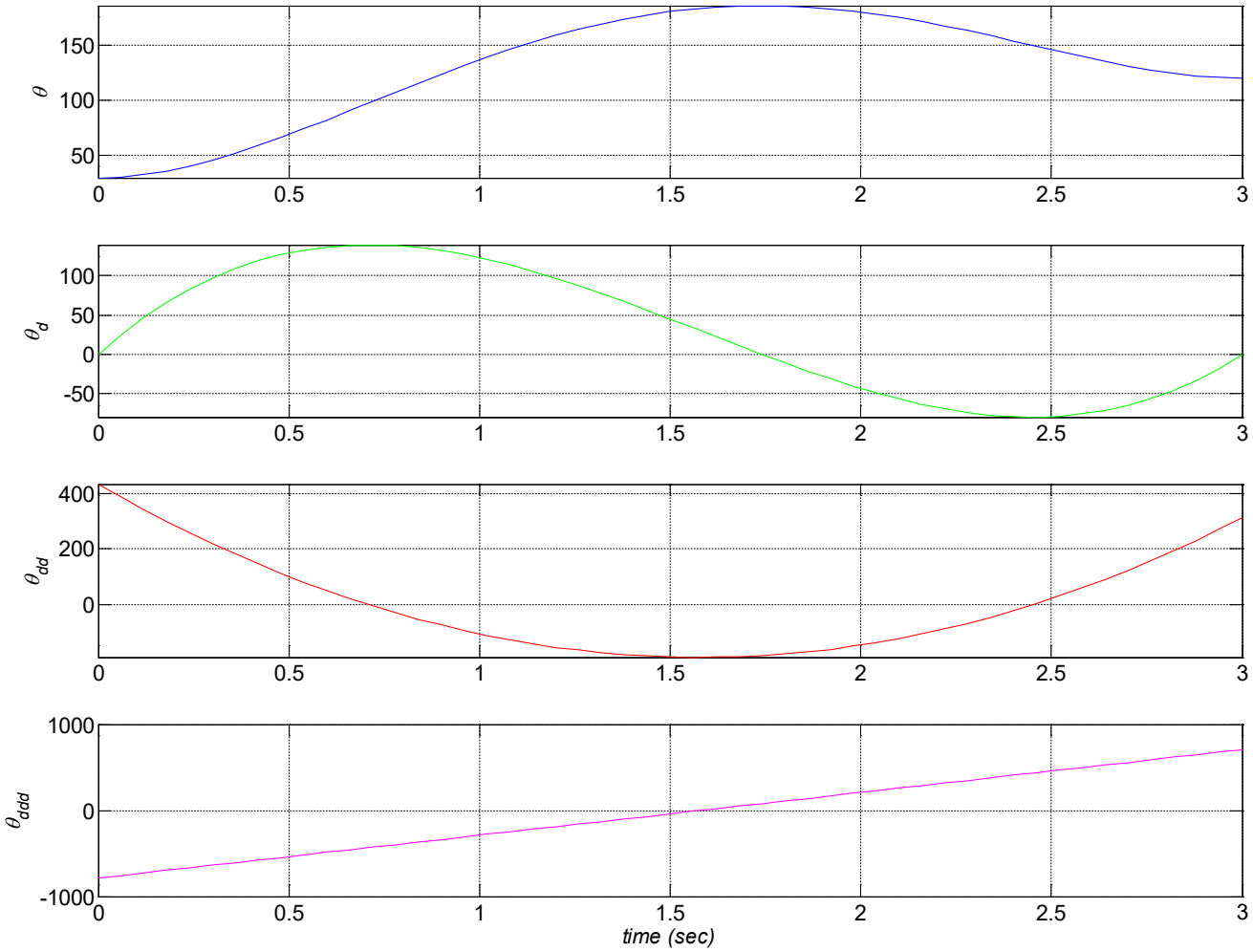
Given: $\theta_S = 30^\circ$, $\theta_V = 180^\circ$, $\theta_F = 120^\circ$, $t_V = 1.5, t_F = 3$ *sec*

Find: $\theta(t)$ for smooth trajectory generation and continuous motion through a via point. Plot $\theta(t), \dot{\theta}(t), \ddot{\theta}(t), \dddot{\theta}(t)$.

Result

$$\theta(t) = 20.74t^4 - 131.11t^3 + 216.67t^2 + 30$$
$$\dot{\theta}(t) = 82.96t^3 - 393.33t^2 + 433.33t$$
$$\ddot{\theta}(t) = 248.89t^2 - 786.67t + 433.33$$
$$\dddot{\theta}(t) = 497.78t - 786.67$$

(Note *deg* units are used throughout)



The $\theta(t)$ shape for the single fourth-order polynomial is very similar to that for the two third-order polynomials. Again the joint angle passes through the via point $\theta_V = 180^\circ$ and keeps going briefly, due to the fact that the velocity is still positive beyond the first 1.5 *sec*. The peak for this fourth-order case is slightly greater and occurs in time slightly after the peak for the two third-order polynomials case. The maximum angle is $\theta_{MAX} = 185.4^\circ$, occurring at $t = 1.74$ sec. For the fourth-order polynomial, a side benefit has arisen. The *jerk* is now continuous at the via time, where it was discontinuous for the two matched third-order polynomials.

The *jerk* still has an infinite spike at the start and end as we saw with both previous third-order polynomial examples, which is **unacceptable**. We correct this with a single sixth-order polynomial as presented in the following Section 6.5. Again, that work is original.

6.5 Single Sixth-Order Polynomial with a Via Point

We can achieve the same goals as the two third-order polynomials meeting at a via point much more simply.[5] We use a single polynomial, forced to go through a via point. Here are the seven constraints for meeting the required angles with smooth motion, starting and ending with zero angular velocities and accelerations. Since we use a single continuous polynomial, the velocity and accelerations are guaranteed to match at the via point.

Note in this case since there is only one time range, it is convenient to treat all times as absolute, rather than relative as when we did matching of two third-order polynomials. With seven constraints a single six-order polynomial is required.

$$\theta(t) = a_6 t^6 + a_5 t^5 + a_4 t^4 + a_3 t^3 + a_2 t^2 + a_1 t + a_0$$
$$\dot{\theta}(t) = 6a_6 t^5 + 5a_5 t^4 + 4a_4 t^3 + 3a_3 t^2 + 2a_2 t + a_1$$
$$\ddot{\theta}(t) = 30a_6 t^4 + 20a_5 t^3 + 12a_4 t^2 + 6a_3 t + 2a_2$$
$$\dddot{\theta}(t) = 120a_6 t^3 + 60a_5 t^2 + 24a_4 t + 6a_3$$

Similar to previous derivations, from three constraints we immediately have:

$$a_0 = \theta_S \qquad a_1 = 0 \qquad a_2 = 0$$

The remaining coupled unknown polynomial coefficients' linear equations may be expressed by the following 4x4 matrix/vector equation.

$$\begin{bmatrix} t_V^3 & t_V^4 & t_V^5 & t_V^6 \\ t_F^3 & t_F^4 & t_F^5 & t_F^6 \\ 3t_F^2 & 4t_F^3 & 5t_F^4 & 6t_F^5 \\ 6t_F & 12t_F^2 & 20t_F^3 & 30t_F^4 \end{bmatrix} \begin{Bmatrix} a_3 \\ a_4 \\ a_5 \\ a_6 \end{Bmatrix} = \begin{Bmatrix} \theta_V - \theta_S \\ \theta_F - \theta_S \\ 0 \\ 0 \end{Bmatrix} \qquad \begin{bmatrix} 1 & t_V & t_V^2 & t_V^3 \\ 1 & t_F & t_F^2 & t_F^3 \\ 3 & 4t_F & 5t_F^2 & 6t_F^3 \\ 6 & 12t_F & 20t_F^2 & 30t_F^3 \end{bmatrix} \begin{Bmatrix} a_3 \\ a_4 \\ a_5 \\ a_6 \end{Bmatrix} = \begin{Bmatrix} (\theta_V - \theta_S)/t_V^3 \\ (\theta_F - \theta_S)/t_F^3 \\ 0 \\ 0 \end{Bmatrix}$$

The 4x4 coefficient matrix determinant is $2t_F^9 t_V^3 (t_F - t_V)^3$ ($2t_F^3 (t_F - t_V)^3$), so as long as neither $t_V = 0$ nor $t_F = 0$, this solution always exists. $t_F > t_V$ is required in this scenario, so that term cannot be zero.

Single Sixth-Order Polynomial with Via Point Analytical Solution

$$a_0 = \theta_S$$
$$a_1 = 0$$
$$a_2 = 0$$

$$a_3 = \frac{1}{(t_F - t_V)^3}\left[\frac{(\theta_V - \theta_S)t_F^3}{t_V^3} - \frac{(\theta_F - \theta_S)(15t_F^2 - 24t_F t_V + 10t_V^2)t_V}{t_F^3}\right]$$

$$a_4 = \frac{3}{(t_F - t_V)^3}\left[-\frac{(\theta_V - \theta_S)t_F^2}{t_V^3} + \frac{(\theta_F - \theta_S)(5t_F^3 - 9t_F t_V^2 + 5t_V^3)}{t_F^4}\right]$$

$$a_5 = \frac{3}{(t_F - t_V)^3}\left[\frac{(\theta_V - \theta_S)t_F}{t_V^3} - \frac{(\theta_F - \theta_S)(8t_F^3 - 9t_F^2 t_V + 2t_V^3)}{t_F^5}\right]$$

$$a_6 = \frac{1}{(t_F - t_V)^3}\left[-\frac{(\theta_V - \theta_S)}{t_V^3} + \frac{(\theta_F - \theta_S)(10t_F^2 - 15t_F t_V + 6t_V^2)}{t_F^5}\right]$$

For the special case of the via time being half the final time ($t_V = t_F/2$), the analytical solution is:

$$a_0 = \theta_S$$
$$a_1 = 0$$
$$a_2 = 0$$

$$a_3 = \frac{2}{t_F^3}\left[32(\theta_V - \theta_S) - 11(\theta_F - \theta_S)\right]$$

$$a_4 = -\frac{3}{t_F^4}\left[64(\theta_V - \theta_S) - 27(\theta_F - \theta_S)\right]$$

$$a_5 = \frac{3}{t_F^5}\left[64(\theta_V - \theta_S) - 30(\theta_F - \theta_S)\right]$$

$$a_6 = -\frac{32}{t_F^6}\left[2(\theta_V - \theta_S) - (\theta_F - \theta_S)\right]$$

Example – Single 6th-Order Polynomial with Via Point

Use a single sixth-order polynomial for smooth joint space trajectory generation, plus motion through an additional via point (no need to stop at the via point, only ensure smooth motion), for one joint only.

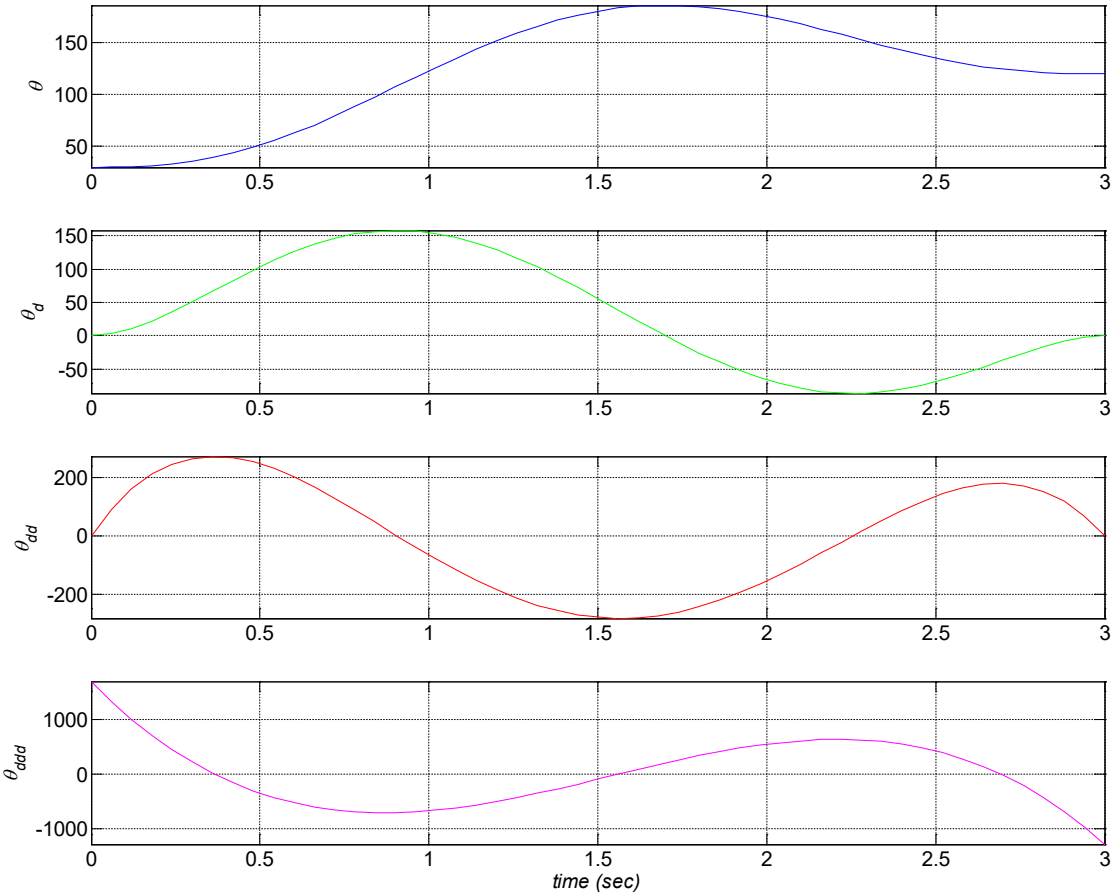
Given: $\theta_S = 30^\circ$, $\theta_V = 180^\circ$, $\theta_F = 120^\circ$, $t_V = 1.5, t_F = 3$ *sec*

Find: $\theta(t)$ for smooth trajectory generation and continuous motion through a via point. Plot $\theta(t), \dot{\theta}(t), \ddot{\theta}(t), \dddot{\theta}(t)$.

Result

$$\theta(t) = -9.22t^6 + 85.19t^5 - 265.56t^4 + 282.22t^3 + 30$$
$$\dot{\theta}(t) = -55.3t^5 + 425.9t^4 - 1062.6t^3 + 846.7t^2$$
$$\ddot{\theta}(t) = -276.5t^4 + 1703.7t^3 - 3186.7t^2 + 1693.3t$$
$$\dddot{\theta}(t) = -1106.2t^3 + 5111.1t^2 - 6373.3t + 1693.3$$

(Note *deg* units are used throughout)



Again the $\theta(t)$ shape is similar to the previous two cases with via point. The maximum angle is $\theta_{MAX} = 185.6^\circ$, occurring at $t = 1.68$ sec. Note the magnitude of the angular velocity, acceleration, and jerk are not greatly different for the three cases we have presented with via point. Generally they are the lowest for the two third-order polynomials. This single sixth-order polynomial approach has eliminated the problem of infinite spikes in *jerk* at the start and end of each motion that occurred in Section 6.4. Here the *jerk* is discontinuous at the start and finish, but it remains finite, which obeys the rule of thumb for mechanical design/motion.

7. Velocity Kinematics

7.1 Velocity Introduction

Up to now, our robotics mathematics has been in the pose domain. Velocity is the first time derivative of position (translational and rotational velocity are both 3x1 vectors in the 3D world). Velocity analysis requires linearized equations and the inverse velocity solution is more straight-forward than inverse pose solutions.

Velocity is useful for many topics in robotics

- Resolved-rate (inverse velocity) control of robots
- Velocity of any point on manipulator
- Moving objects in the robot workspace
- Manipulator dynamics

Pose

Translational

Position **is** a vector

Rotational

e.g. *Z-Y-X* 3-2-1 Euler angles α, β, γ

Orientation **is not** a vector.

Velocity

is the vector time rate of change of position and orientation.

$\{\omega\} \neq \{\dot{\alpha} \quad \dot{\beta} \quad \dot{\gamma}\}^T$ (non-holonomic)

Both translational and rotational velocities are vectors.

7.1.1 Translational Velocity

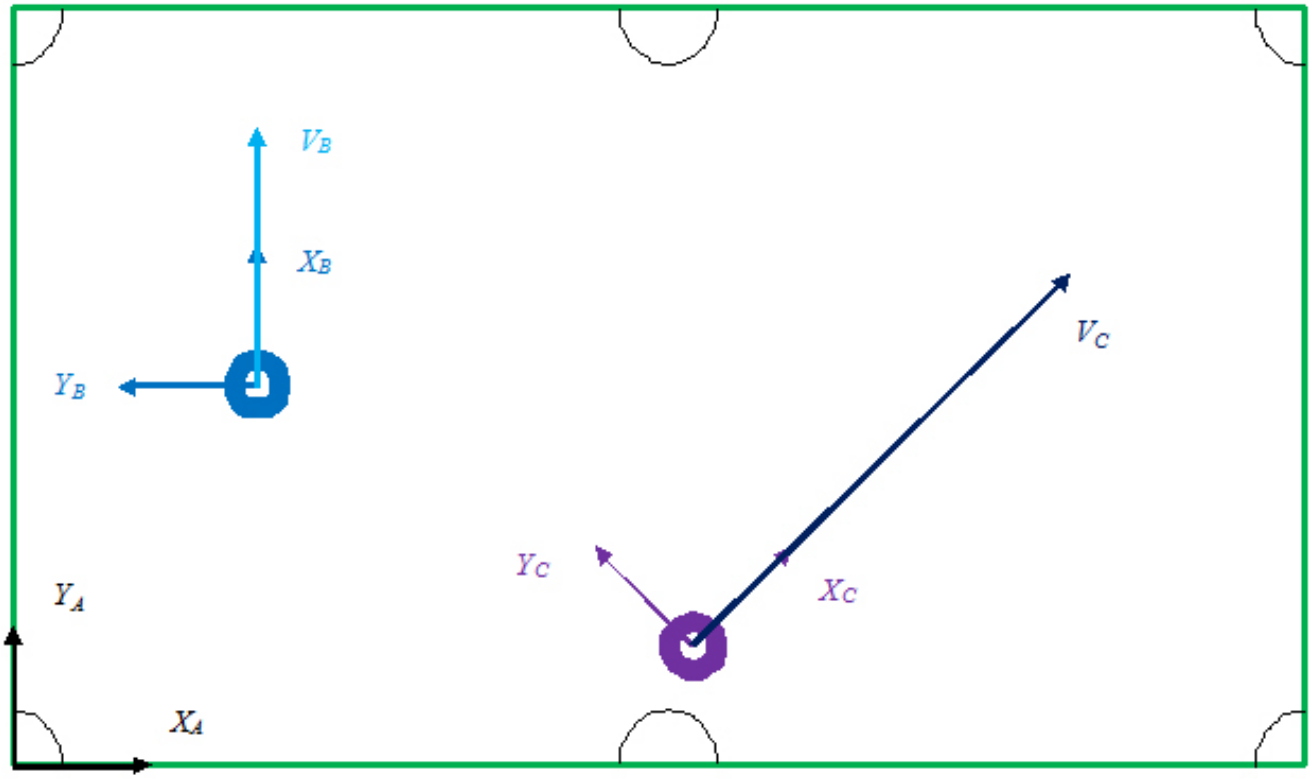
Three coordinate frames are involved in general (two origin points and a frame). ${}^k\{{}^iV_j\}$ is the translational velocity of the origin of $\{j\}$ moving with respect to the origin of $\{i\}$, expressed in the basis (coordinates) of $\{k\}$.

For position vectors, three frames were also involved, just one was usually implied. $\{{}^iP_j\}$ is the position vector from the origin of $\{i\}$ to the origin of $\{j\}$, expressed in $\{i\}$ coordinates. So, this is more properly ${}^k\{{}^iP_j\}$, where $\{k\}$ is implicitly assumed to be $\{i\}$, unless otherwise stated.

Translational Velocity Example

This example demonstrates the relative velocity equation and the various indices in ${}^k\{{}^iV_j\}$. Two balls *B* and *C*, with the attached coordinate frames shown, are moving in the plane with instantaneous translational velocity vectors V_B and V_C. $\{A\}$ is a fixed reference frame.

Given: the absolute velocity of B V_B is $2\frac{in}{s}@90^\circ$ the absolute velocity of C V_C is $4\frac{in}{s}@45^\circ$



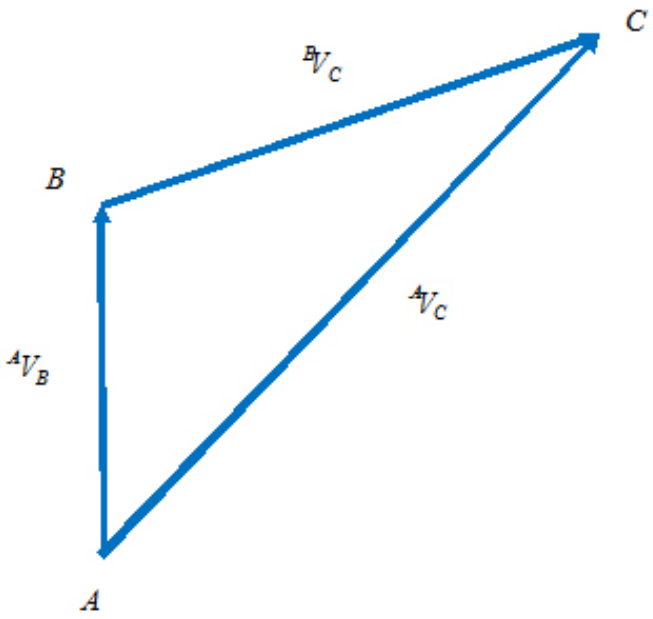
From the given information:

$$^A\{^AV_B\} = \begin{Bmatrix} 0 \\ 2 \\ 0 \end{Bmatrix}; \qquad ^A\{^AV_C\} = \begin{Bmatrix} 2.83 \\ 2.83 \\ 0 \end{Bmatrix}$$

From the relative velocity equation:

$$^A\{^AV_C\} = {}^A\{^AV_B\} + {}^A\{^BV_C\} \qquad ^A\{^BV_C\} = {}^A\{^AV_C\} - {}^A\{^AV_B\} = \begin{Bmatrix} 2.83 \\ 0.83 \\ 0 \end{Bmatrix}; \qquad 2.95\frac{in}{s}@16^\circ$$

The relative translational velocity diagram is shown below.



Since velocity is a free vector, these same velocity vectors ($\{{}^{A}V_B\}, \{{}^{A}V_C\}, \{{}^{B}V_C\}$) can be expressed in frames $\{B\}$ and $\{C\}$ by using the rotation matrix (or just by looking at the components).

$$ {}^{k}\{{}^{i}V_j\} = \left[{}^{k}_{A}R\right] {}^{A}\{{}^{i}V_j\} $$

$$ {}^{B}\{{}^{A}V_B\} = \begin{Bmatrix} 2 \\ 0 \\ 0 \end{Bmatrix} \qquad {}^{B}\{{}^{A}V_C\} = \begin{Bmatrix} 2.83 \\ -2.83 \\ 0 \end{Bmatrix} \qquad {}^{B}\{{}^{B}V_C\} = \begin{Bmatrix} 0.83 \\ -2.83 \\ 0 \end{Bmatrix} $$

$$ {}^{C}\{{}^{A}V_B\} = \begin{Bmatrix} 1.41 \\ 1.41 \\ 0 \end{Bmatrix} \qquad {}^{C}\{{}^{A}V_C\} = \begin{Bmatrix} 4 \\ 0 \\ 0 \end{Bmatrix} \qquad {}^{C}\{{}^{B}V_C\} = \begin{Bmatrix} 2.58 \\ -1.43 \\ 0 \end{Bmatrix} $$

There are 27 permutations for A, B, C. So far, we have given 9 combinations. There are two more types.

1) ${}^{k}\{{}^{j}V_i\} = -{}^{k}\{{}^{i}V_j\}$ e.g. ${}^{A}\{{}^{B}V_A\} = \begin{Bmatrix} 0 \\ -2 \\ 0 \end{Bmatrix}$ (9 of these)

2) ${}^{k}\{{}^{i}V_i\} = \{0\}$ e.g. ${}^{A}\{{}^{B}V_B\} = \begin{Bmatrix} 0 \\ 0 \\ 0 \end{Bmatrix}$ (9 of these)

General three-part velocity equation

From the homogeneous transformation matrix development, in order to derive the relationship $\left\{{}^{A}P\right\} = \left[{}^{A}_{B}T\right]\left\{{}^{B}P\right\}$, we started with the two-part pose equation repeated below.

$$\left\{{}^{A}P\right\} = \left\{{}^{A}P_{B}\right\} + \left[{}^{A}_{B}R\right]\left\{{}^{B}P\right\}$$

A single time derivative of this two-part pose equation yields the so-called three-part velocity equation (origin velocity, sliding velocity, and tangential velocity).

$$\left\{{}^{A}V\right\} = \left\{{}^{A}\dot{P}_{B}\right\} + \left[{}^{A}_{B}R\right]\left\{{}^{B}\dot{P}\right\} + \left[{}^{A}_{B}\dot{R}\right]\left\{{}^{B}P\right\}$$

$$\left\{V\right\} = \left\{V_0\right\} + \left\{V_P\right\} + \left\{\omega\right\} \times \left\{P\right\}$$

Adding descriptive indices for moving frames, reference frames, and basis frame $\{A\}$ completes this translational velocity equation.

$${}^{A}\left\{{}^{A}V_{P}\right\} = {}^{A}\left\{{}^{A}V_{B}\right\} + \left[{}^{A}_{B}R\right]\left\{{}^{B}V_{P}\right\} + {}^{A}\left\{{}^{A}\omega_{B}\right\} \times \left[{}^{A}_{B}R\right]\left\{{}^{B}P\right\}$$

A rotating rigid body has different translational velocities at different points. However, it has the same rotational velocity over the whole rigid body.

7.1.2 Rotational Velocity

Angular Velocity Vector

${}^{A}\left\{{}^{A}\omega_{B}\right\}$ is the angular velocity of Cartesian coordinate frame $\{B\}$ moving with respect to $\{A\}$, expressed in the basis of $\{A\}$, i.e. in $\{A\}$ coordinates. ${}^{A}\left\{{}^{A}\omega_{B}\right\}$ is a unique vector description, as opposed to the 12 artificial descriptions for the same 3D orientation when using Euler angles.

Rotational velocity is non-holonomic: we cannot integrate ${}^{A}\left\{{}^{A}\omega_{B}\right\}$ to get an orientation vector; in fact, there is no such thing as an orientation vector.

$$\left\{\omega\right\} = \left\{\omega_x \quad \omega_y \quad \omega_z\right\}^{T} \neq \left\{\dot{\theta}_x \quad \dot{\theta}_y \quad \dot{\theta}_z\right\}^{T}$$ but there is a relationship.

Kinematic Differential Equations in terms of Orientation Angles

e.g. *Z-Y-X* (α–β–γ) Euler angle convention

$$\left[{}^{A}_{B}R\right] = \begin{bmatrix} c\alpha c\beta & -s\alpha c\gamma + c\alpha s\beta s\gamma & s\alpha s\gamma + c\alpha s\beta c\gamma \\ s\alpha c\beta & c\alpha c\gamma + s\alpha s\beta s\gamma & -c\alpha s\gamma + s\alpha s\beta c\gamma \\ -s\beta & c\beta s\gamma & c\beta c\gamma \end{bmatrix}$$

From **Spacecraft Dynamics** by Kane, Likins, and Levinson.

$${}^{A}\left\{{}^{A}\omega_{B}\right\} = \begin{Bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{Bmatrix} = \begin{bmatrix} -s\beta & 0 & 1 \\ c\beta s\gamma & c\gamma & 0 \\ c\beta c\gamma & -s\gamma & 0 \end{bmatrix} \begin{Bmatrix} \dot{\alpha} \\ \dot{\beta} \\ \dot{\gamma} \end{Bmatrix}$$

where $\{\dot{\alpha} \quad \dot{\beta} \quad \dot{\gamma}\}^T$ are the Euler angle rates. Remember description of 3D rotations is artificial (there are 23 other descriptions using both Euler and Fixed angles descriptions, not counting axis-angle convention or quaternions. The inverse rate relationship for *Z-Y-X* (α–β–γ) Euler angle convention is:

$$\begin{Bmatrix} \dot{\alpha} \\ \dot{\beta} \\ \dot{\gamma} \end{Bmatrix} = \begin{bmatrix} 0 & \frac{s\gamma}{c\beta} & \frac{c\gamma}{c\beta} \\ 0 & c\gamma & -s\gamma \\ 1 & s\gamma t\beta & c\gamma t\beta \end{bmatrix} \begin{Bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{Bmatrix} \qquad \text{where: } t\beta = \tan\beta = \frac{\sin\beta}{\cos\beta}$$

This solution above has the artificial (algorithmic) singularities $\beta = \pm 90^\circ$. A rotating rigid body has the same rotational velocity at different points.

Angular velocity vector equation for multiple frames:

Though no 3D orientation description can be differentiated to obtain a 3D angular velocity vector, there is a simple method to determine 3D angular velocity vectors for robotics velocity analysis, using the following relative angular velocity vector equation (example for 3 revolute joints).

$$\left\{{}^0\omega_3\right\} = \left\{{}^0\omega_1\right\} + \left\{{}^1\omega_2\right\} + \left\{{}^2\omega_3\right\}$$

All of these component angular velocity vectors must be expressed in the same frame.

$${}^0\left\{{}^0\omega_3\right\} = {}^0\left\{{}^0\omega_1\right\} + {}^0\left\{{}^1\omega_2\right\} + {}^0\left\{{}^2\omega_3\right\} = {}^0\left\{{}^0\omega_1\right\} + \left[{}^0_1R\right]\left\{{}^1\omega_2\right\} + \left[{}^0_2R\right]\left\{{}^2\omega_3\right\}$$

Often it is more convenient to express all vectors in the local moving frames (as in robot joints, where R joints must rotate about the local *Z* axis; what if one or more joints are prismatic joints P?).

$${}^0\left\{{}^0\omega_3\right\} = \left[{}^0_1R\right]{}^1\left\{{}^0\omega_1\right\} + \left[{}^0_2R\right]{}^2\left\{{}^1\omega_2\right\} + \left[{}^0_3R\right]{}^3\left\{{}^2\omega_3\right\}$$

7.1.3 Combined Translational and Rotational Velocity

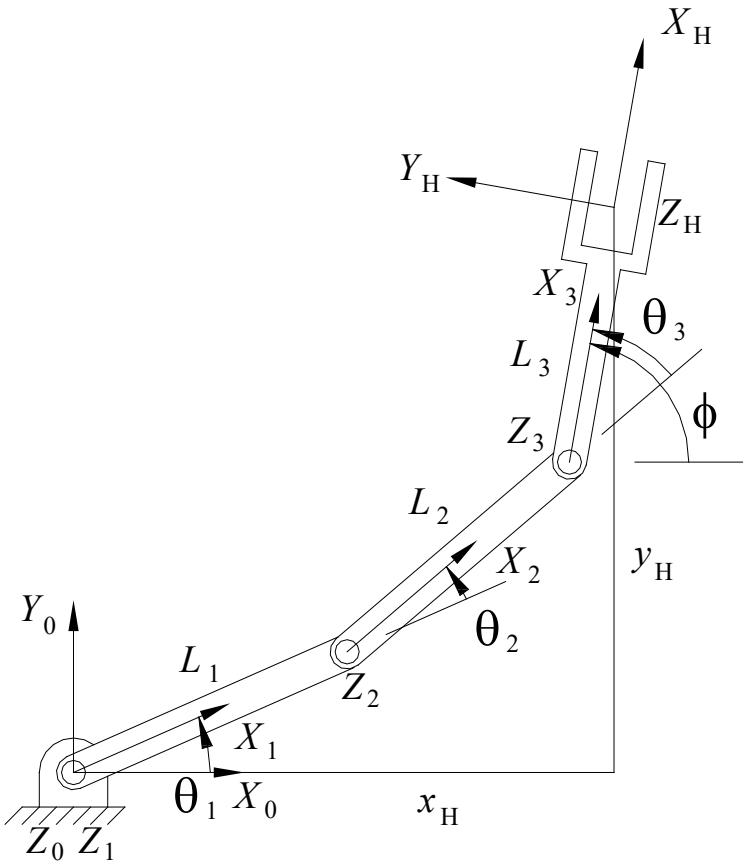
We can artificially combine Cartesian translational and rotational velocity vectors into one convenient combined velocity vector, for example giving the motion of the hand frame $\{H\}$ moving with respect to the base frame $\{B\}$.

$${}^k\left\{{}^B\dot{X}_H\right\} = {}^k\begin{Bmatrix} \left\{{}^BV_H\right\} \\ \left\{{}^B\Omega_H\right\} \end{Bmatrix} = \begin{Bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \omega_x \\ \omega_y \\ \omega_z \end{Bmatrix}$$

Where *k* is the frame of coordinates expression, i.e. the velocity vector is projected into the basis of frame $\{k\}$. If *k* is not stated it is assumed to be *B*. In the general 3D velocity case, both ${}^k\left\{{}^BV_H\right\}$ and ${}^k\left\{{}^B\Omega_H\right\}$ are 3x1 vectors, with units *m*/*s* and *rad*/*s*, respectively.

7.2 Velocity Equations Derivations

7.2.1 Planar 3R 3-dof Robot



Recall the forward pose kinematics result:

$$\left[{}^{0}_{H}T\right] = \begin{bmatrix} c_{123} & -s_{123} & 0 & L_1c_1 + L_2c_{12} + L_3c_{123} \\ s_{123} & c_{123} & 0 & L_1s_1 + L_2s_{12} + L_3s_{123} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c\phi & -s\phi & 0 & x_H \\ s\phi & c\phi & 0 & y_H \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The three independent pose kinematics equations are:

$$\begin{aligned} {}^{0}x_H &= L_1c_1 + L_2c_{12} + L_3c_{123} \\ {}^{0}y_H &= L_1s_1 + L_2s_{12} + L_3s_{123} \\ \phi &= \theta_1 + \theta_2 + \theta_3 \end{aligned}$$

The translational velocity equations are found from the first time derivative of the *XY* position equations:

Use the relative angular velocity equation to find the rotational velocity equation (be sure to express all angular velocities in a common frame, {0} here).

$$
\begin{aligned}
{}^0\{{}^0\omega_H\} &= \left[{}^0_1R\right]{}^1\{{}^0\omega_1\} + \left[{}^0_2R\right]{}^2\{{}^1\omega_2\} + \left[{}^0_3R\right]{}^3\{{}^2\omega_3\} + \left[{}^0_HR\right]{}^H\{{}^3\omega_H\} \\
&= \begin{bmatrix} c_1 & -s_1 & 0 \\ s_1 & c_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{Bmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{Bmatrix} + \begin{bmatrix} c_{12} & -s_{12} & 0 \\ s_{12} & c_{12} & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{Bmatrix} 0 \\ 0 \\ \dot{\theta}_2 \end{Bmatrix} + \begin{bmatrix} c_{123} & -s_{123} & 0 \\ s_{123} & c_{123} & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{Bmatrix} 0 \\ 0 \\ \dot{\theta}_3 \end{Bmatrix} + \begin{bmatrix} c_{123} & -s_{123} & 0 \\ s_{123} & c_{123} & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{Bmatrix} 0 \\ 0 \\ 0 \end{Bmatrix} \\
&= \begin{Bmatrix} 0 \\ 0 \\ \dot{\theta}_1 + \dot{\theta}_2 + \dot{\theta}_3 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ {}^0\omega_{Hz} \end{Bmatrix}
\end{aligned}
$$

For the planar case only, this rotational velocity equation could have also been found from the time derivative of ϕ (this ONLY applies to the planar case where all active Z axes are parallel).

Write the translational and rotational velocity equations into one matrix-vector system of equations.

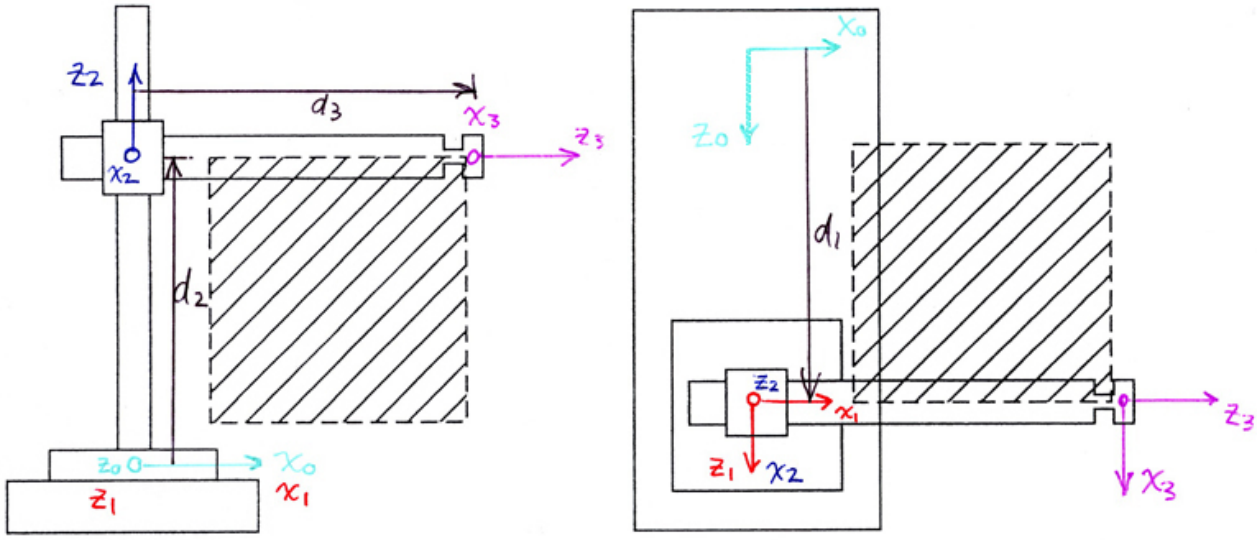
where $\left[{}^0J_H\right]$ is the **Jacobian matrix** mapping the relative joint rates $\left\{\dot{\Theta}\right\}$ to the absolute Cartesian velocities $\left\{{}^0\dot{X}\right\}$. $\left\{{}^0\dot{X}\right\}$ are the Cartesian translational and rotational rates of frame $\{H\}$ moving with respect to $\{0\}$, expressed in the coordinates of $\{0\}$.

$$
\left[{}^0J_H\right] = \begin{bmatrix} -L_1s_1 - L_2s_{12} - L_3s_{123} & -L_2s_{12} - L_3s_{123} & -L_3s_{123} \\ L_1c_1 + L_2c_{12} + L_3c_{123} & L_2c_{12} + L_3c_{123} & L_3c_{123} \\ 1 & 1 & 1 \end{bmatrix}
$$

Velocity equations derivations examples for additional robots (3-dof Cartesian and 3-dof translational PUMA) are presented in the following two subsections.

7.2.2 Spatial 3P 3-dof Cartesian manipulator

$$\{{}^0\dot{X}\} = [{}^0J]\{\dot{L}\} \qquad \{L\} = \{d_1 \quad d_2 \quad d_3\}^T$$



Derive ${}^0J$ via Partial derivatives definition

$$[J] = \left[\frac{\partial f_i}{\partial L_j}\right] \qquad f = {}^0\{{}^0P_3\} = \begin{Bmatrix} p_x \\ p_y \\ p_z \end{Bmatrix} = \begin{Bmatrix} d_3 \\ d_2 \\ d_1 \end{Bmatrix}$$

$$\frac{\partial p_x}{\partial d_1} = 0 \qquad \frac{\partial p_x}{\partial d_2} = 0 \qquad \frac{\partial p_x}{\partial d_3} = 1$$

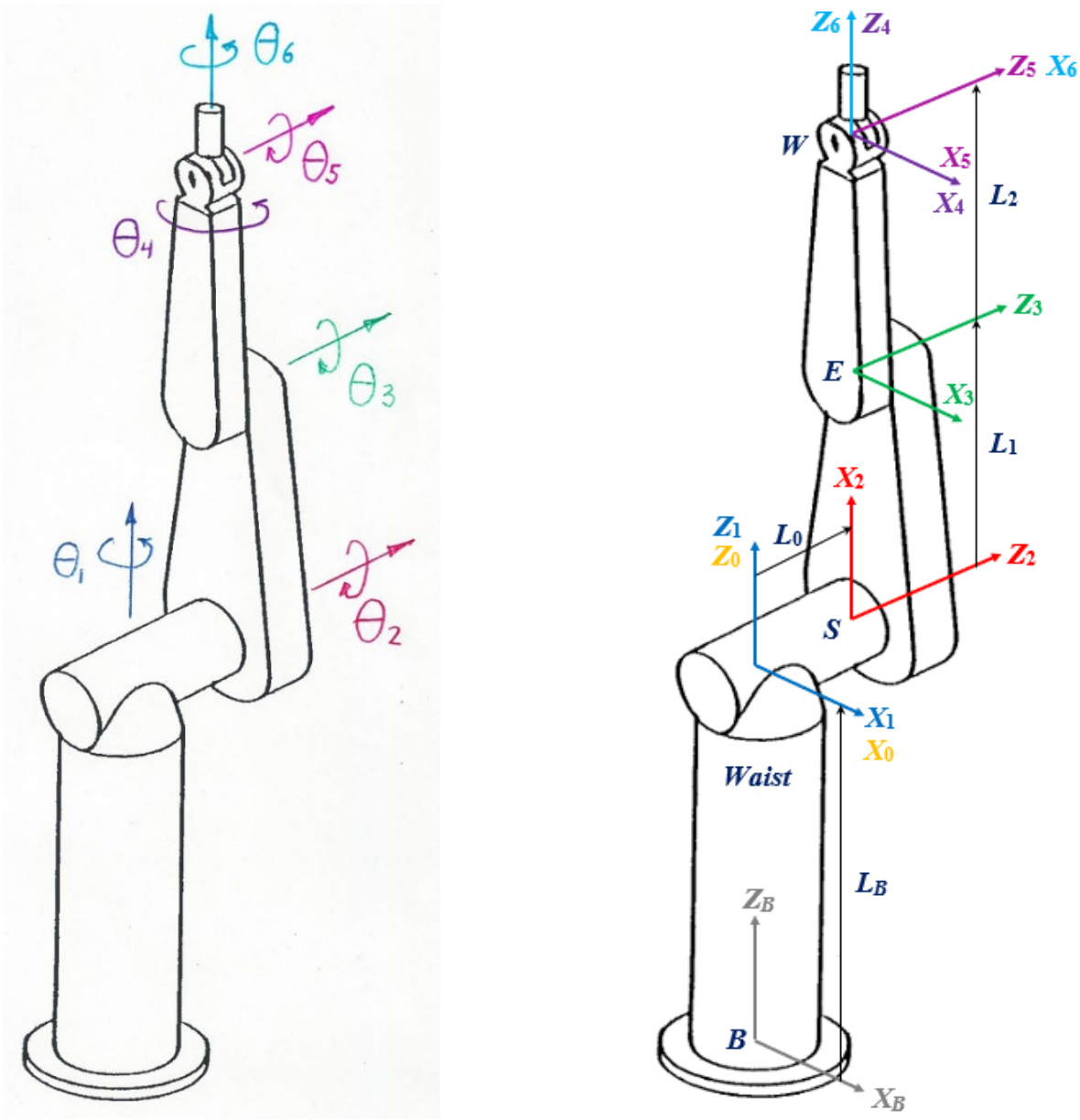
$$\frac{\partial p_y}{\partial d_1} = 0 \qquad \frac{\partial p_y}{\partial d_2} = 1 \qquad \frac{\partial p_y}{\partial d_3} = 0$$

$$\frac{\partial p_z}{\partial d_1} = 1 \qquad \frac{\partial p_z}{\partial d_2} = 0 \qquad \frac{\partial p_z}{\partial d_3} = 0$$

$$[{}^0J] = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \qquad {}^0\begin{Bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{Bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{Bmatrix} \dot{d}_1 \\ \dot{d}_2 \\ \dot{d}_3 \end{Bmatrix}$$

$${}^0\{{}^0\omega_3\} = \begin{Bmatrix} 0 \\ 0 \\ 0 \end{Bmatrix}$$

7.2.3 Spatial 6R 6-dof PUMA robot (3R 3-dof translational part only)



The PUMA translational velocity equations are derived as follows, where W indicates the origin of the wrist frame $\{W\}$ (recall the position of the wrist frame origin is a function of only the first three joint angles $(\theta_1,\theta_2,\theta_3)$. We must take the first time derivative of the position vector $\left\{{}^0P_6(\theta_1,\theta_2,\theta_3)\right\}$:

$$\left\{{}^0P_6(\theta_1,\theta_2,\theta_3)\right\} = {}^0\begin{Bmatrix} x \\ y \\ z \end{Bmatrix}_W = \begin{Bmatrix} -L_0 s_1 + L_1 c_1 s_2 + L_2 c_1 s_{23} \\ L_0 c_1 + L_1 s_1 s_2 + L_2 s_1 s_{23} \\ L_1 c_2 + L_2 c_{23} \end{Bmatrix}$$

Here are the PUMA translational velocity equations:

$$
\begin{aligned}
{}^0\dot{x}_W &= -L_0 c_1 \dot{\theta}_1 - L_1 s_1 s_2 \dot{\theta}_1 + L_1 c_1 c_2 \dot{\theta}_2 - L_2 s_1 s_{23} \dot{\theta}_1 + L_2 c_1 c_{23} (\dot{\theta}_2 + \dot{\theta}_3) \\
{}^0\dot{y}_W &= -L_0 s_1 \dot{\theta}_1 + L_1 c_1 s_2 \dot{\theta}_1 + L_1 s_1 c_2 \dot{\theta}_2 + L_2 c_1 s_{23} \dot{\theta}_1 + L_2 s_1 c_{23} (\dot{\theta}_2 + \dot{\theta}_3) \\
{}^0\dot{z}_W &= -L_1 s_2 \dot{\theta}_2 - L_2 s_{23} (\dot{\theta}_2 + \dot{\theta}_3)
\end{aligned}
$$

written in matrix-vector form:

$$
{}^0 \begin{Bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{Bmatrix}_W = \begin{bmatrix} -L_0 c_1 - L_1 s_1 s_2 - L_2 s_1 s_{23} & L_1 c_1 c_2 + L_2 c_1 c_{23} & L_2 c_1 c_{23} \\ -L_0 s_1 + L_1 c_1 s_2 + L_2 c_1 s_{23} & L_1 s_1 c_2 + L_2 s_1 c_{23} & L_2 s_1 c_{23} \\ 0 & -L_1 s_2 - L_2 s_{23} & -L_2 s_{23} \end{bmatrix} \begin{Bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{Bmatrix}
$$

$$
{}^0 \left\{ \dot{X} \right\}_W = \left[{}^0 J \right] \left\{ \dot{\Theta} \right\}
$$

where

$$
\left[{}^0 J \right] = \begin{bmatrix} -L_0 c_1 - L_1 s_1 s_2 - L_2 s_1 s_{23} & L_1 c_1 c_2 + L_2 c_1 c_{23} & L_2 c_1 c_{23} \\ -L_0 s_1 + L_1 c_1 s_2 + L_2 c_1 s_{23} & L_1 s_1 c_2 + L_2 s_1 c_{23} & L_2 s_1 c_{23} \\ 0 & -L_1 s_2 - L_2 s_{23} & -L_2 s_{23} \end{bmatrix}
$$

is the PUMA robot translational velocity Jacobian matrix.

Why is the (3,1) term of the Jacobian matrix zero? ($\dot{\theta}_1$ cannot affect ${}^0\dot{z}_W$)

7.3 Jacobian Matrices

The Jacobian matrix is a linear transformation mapping joint rates to Cartesian velocities:

m = dimension of the Cartesian (task) space n = dimension of the joint space

We can express the resulting Cartesian velocities in any frame $\{k\}$; $\left\{\dot{\Theta}\right\}$ are the relative joint angle rates and hence are expressed about the n different local Z axes.

The Jacobian matrix is a multi-dimensional form of the derivative:

For a general n-joint spatial robot operating in the standard m=6-dimensional Cartesian space:

$$\left\{{}^{k}\dot{X}\right\} = \begin{Bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \omega_x \\ \omega_y \\ \omega_z \end{Bmatrix} \qquad \left\{\dot{\Theta}\right\} = \begin{Bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \vdots \\ \dot{\theta}_n \end{Bmatrix} \text{ (use } \dot{d}_i \text{ for any prismatic joints)}$$

There are (at least) 4 methods to derive the Jacobian matrix for serial robot velocity; these are presented below.

1) Time derivatives and relative angular velocity equation

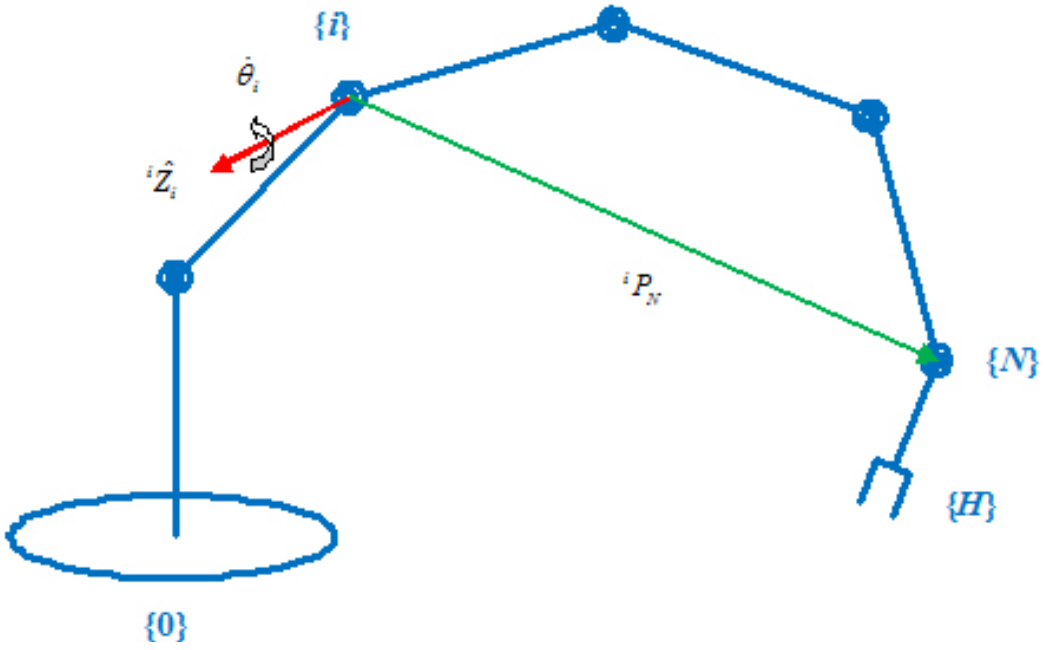
This method has been used for the Planar 3R and PUMA robots above.

2) Jacobian matrix column as the end-effector translational and rotational velocity due to joint *i*

Physical interpretation of the Jacobian matrix. Each column i is the absolute Cartesian velocity vector of the last active joint frame $\{N\}$ with respect to the base frame, due to joint i only, and with the variable joint rate ($\dot{\theta}_1$ or $\dot{d}_i$ for revolute or prismatic joint, respectively) factored out.

$$ {}^{k}[J] = {}^{k}\begin{bmatrix} | & | & \cdots & | \\ \left\{{}^{0}\dot{X}_N\right\}_1 & \left\{{}^{0}\dot{X}_N\right\}_2 & \cdots & \left\{{}^{0}\dot{X}_N\right\}_N \\ | & | & \cdots & | \end{bmatrix} \qquad {}^{k}\left\{{}^{0}\dot{X}_N\right\}_i = \begin{Bmatrix} {}^{k}\left\{{}^{0}V_N\right\}_i \\ {}^{k}\left\{{}^{0}\omega_N\right\}_i \end{Bmatrix} $$

a) Revolute Joint Columns



Here is Jacobian matrix column *i*, for a revolute joint.

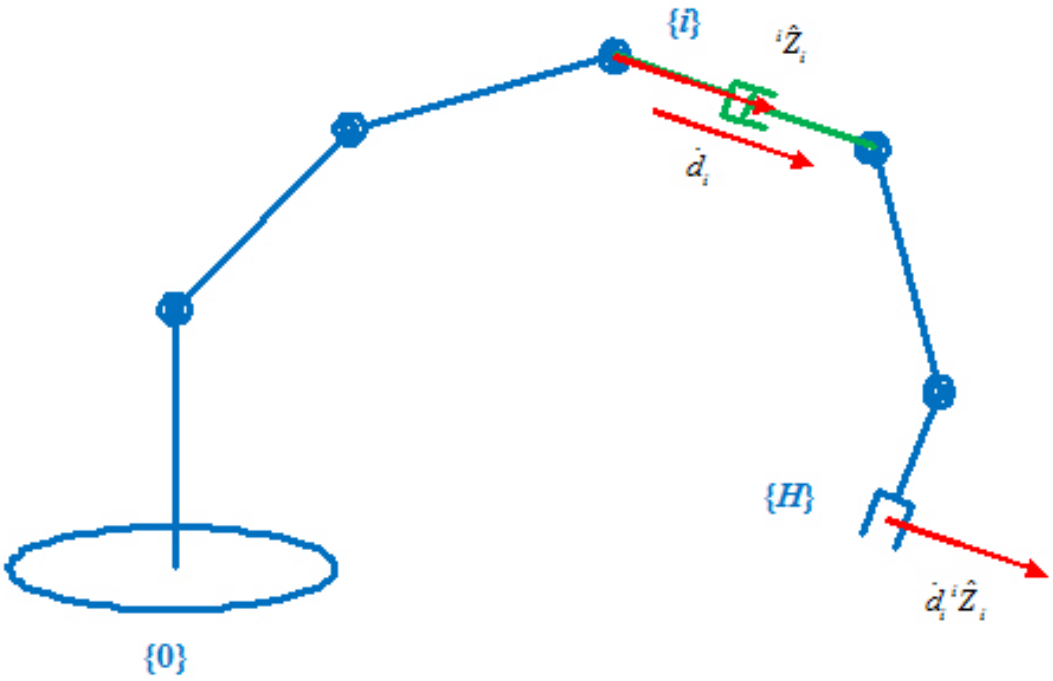
$$ {}^k\{J\}_i = \begin{Bmatrix} \{{}^k\hat{Z}_i\} \times {}^k\{{}^iP_N\} \\ \{{}^k\hat{Z}_i\} \end{Bmatrix} = \begin{Bmatrix} [{}^k_iR]\{{}^i\hat{Z}_i \times {}^iP_N\} \\ [{}^k_iR]\{{}^i\hat{Z}_i\} \end{Bmatrix} $$

where $\{{}^k\hat{Z}_i\} = [{}^k_iR]\{{}^i\hat{Z}_i\}$ is the third column of $[{}^k_iR]$ and ${}^k\{{}^iP_N\} = [{}^k_iR]\,{}^i\{{}^iP_N\}$ where ${}^i\{{}^iP_N\}$ is the translational part of $[{}^i_NT]$. The end-effector Cartesian velocity is found using the velocity transformation of Section 7.9.

Here is the Jacobian matrix for an all-revolute manipulator.

$$ {}^k[J] = {}^k\left[\begin{Bmatrix} \{{}^k\hat{Z}_1\} \times {}^k\{{}^1P_N\} \\ \{{}^k\hat{Z}_1\} \end{Bmatrix} \cdots \begin{Bmatrix} \{{}^k\hat{Z}_i\} \times {}^k\{{}^iP_N\} \\ \{{}^k\hat{Z}_i\} \end{Bmatrix} \cdots \begin{Bmatrix} \{{}^k\hat{Z}_N\} \times {}^k\{{}^NP_N\} \\ \{{}^k\hat{Z}_N\} \end{Bmatrix} \right] $$

b) Prismatic Joint Columns



Here is Jacobian matrix column *i*, for a prismatic joint.

$$ {}^{k}\{J\}_i = \begin{Bmatrix} \{ {}^{k}\hat{Z}_i \} \\ 0 \end{Bmatrix} = \begin{Bmatrix} [{}^{k}_{i}R] \{ {}^{i}\hat{Z}_i \} \\ 0 \end{Bmatrix} $$

where $\{ {}^{k}\hat{Z}_i \} = [{}^{k}_{i}R] \{ {}^{i}\hat{Z}_i \}$ is the third column of $[{}^{k}_{i}R]$.

3) Derive ${}^0[J]$ via Partial derivatives definition

Example for the planar 3R robot.

${}^0[J] =$

$$f = {}^0\{{}^0P_3\} = \begin{Bmatrix} p_x \\ p_y \end{Bmatrix} =$$

$$\frac{\partial p_i}{\partial t} = \sum_{j=1}^{3} \frac{\partial p_i}{\partial \theta_j}\frac{d\theta_j}{dt} = \sum_{j=1}^{3} \frac{\partial p_i}{\partial \theta_j}\dot{\theta}_j \qquad i = x, y$$

$$\frac{\partial p_x}{\partial \theta_1} = \qquad \frac{\partial p_x}{\partial \theta_2} = \qquad \frac{\partial p_x}{\partial \theta_3} =$$

$$\frac{\partial p_y}{\partial \theta_1} = \qquad \frac{\partial p_y}{\partial \theta_2} = \qquad \frac{\partial p_y}{\partial \theta_3} =$$

The angular velocity vector is found from the relative angular velocity equation.

$$\begin{aligned} {}^0\{{}^0\omega_H\} &= \left[{}^0_1R\right]^1\{{}^0\omega_1\} + \left[{}^0_2R\right]^2\{{}^1\omega_2\} + \left[{}^0_3R\right]^3\{{}^2\omega_3\} + \left[{}^0_H R\right]^H\{{}^3\omega_H\} \\ &= \begin{bmatrix} c_1 & -s_1 & 0 \\ s_1 & c_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{Bmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{Bmatrix} + \begin{bmatrix} c_{12} & -s_{12} & 0 \\ s_{12} & c_{12} & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{Bmatrix} 0 \\ 0 \\ \dot{\theta}_2 \end{Bmatrix} + \begin{bmatrix} c_{123} & -s_{123} & 0 \\ s_{123} & c_{123} & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{Bmatrix} 0 \\ 0 \\ \dot{\theta}_3 \end{Bmatrix} + \begin{bmatrix} c_{123} & -s_{123} & 0 \\ s_{123} & c_{123} & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{Bmatrix} 0 \\ 0 \\ 0 \end{Bmatrix} \\ &= \begin{Bmatrix} 0 \\ 0 \\ \dot{\theta}_1 + \dot{\theta}_2 + \dot{\theta}_3 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ {}^0\omega_{Hz} \end{Bmatrix} \end{aligned}$$

The overall Jacobian matrix in {0} coordinates is:

$$^{0}[J] =$$

f – vector of m Cartesian functions. This relationship holds only for translational part. That is, no Cartesian orientation vector exists that we can differentiate to get the angular velocity vector.

4) Velocity recursion a 'la Craig[7]

Add translational and rotational velocities serially link-by-link from the base to the end-effector link.

Revolute joint

$$\left\{ {}^{i+1}\omega_{i+1} \right\} = \left[{}^{i+1}_{i}R \right] \left\{ {}^{i}\omega_{i} \right\} + \left\{ {}^{i+1}\hat{Z}_{i+1} \right\} \dot{\theta}_{i+1}$$

$$\left\{ {}^{i+1}v_{i+1} \right\} = \left[{}^{i+1}_{i}R \right] \left(\left\{ {}^{i}v_{i} \right\} + \left\{ {}^{i}\omega_{i} \right\} \times \left\{ {}^{i}P_{i+1} \right\} \right)$$

Prismatic joint

$$\left\{ {}^{i+1}\omega_{i+1} \right\} = \left[{}^{i+1}_{i}R \right] \left\{ {}^{i}\omega_{i} \right\}$$

$$\left\{ {}^{i+1}v_{i+1} \right\} = \left[{}^{i+1}_{i}R \right] \left(\left\{ {}^{i}v_{i} \right\} + \left\{ {}^{i}\omega_{i} \right\} \times \left\{ {}^{i}P_{i+1} \right\} \right) + \left\{ {}^{i+1}\hat{Z}_{i+1} \right\} \dot{d}_{i+1}$$

Start with $\left\{ {}^{0}\omega_{0} \right\} = \left\{ {}^{0}v_{0} \right\} = \left\{ 0 \quad 0 \quad 0 \right\}^{T}$ (if the robot base frame {0} is fixed).

Use the velocity recursion equations, $i = 0, 1, 2, \cdots, N-1$. Then, factor out $\left\{ \dot{\Theta} \right\}$ from $\left\{ {}^{N}v_{N} \right\}, \left\{ {}^{N}\omega_{N} \right\}$ to get the Jacobian matrix $^{N}[J]$.

[7] J.J. Craig, 2005, Introduction to Robotics: Mechanics and Control, Third edition, Pearson Prentice Hall, Upper Saddle River, NJ.

7.4 Forward Velocity Problem

There are many useful applications for the Jacobian matrix in robot velocity and related topics. One major Jacobian matrix application is the solution to the **Forward Velocity Problem**. This problem is stated, in general:

Given: the robot (including all constant DH Parameters)

values for all of the joint variables $\{\Theta\}$
(angles θ_i for **R** joints and lengths d_i for **P** joints)

values for all of the relative joint rates $\left\{\dot{\Theta}\right\}$
(joint rates $\dot{\theta}_i$ for **R** joints and $\dot{d}_i$ for **P** joints)

Find: the end-effector Cartesian velocity $\left\{\dot{X}\right\}$ (translational and rotational)

The Forward Velocity Solution is straight-forward, using the equation:

we can calculate the resulting Cartesian velocities $\left\{{}^{k}\dot{X}\right\}$ given the Jacobian matrix $\left[{}^{k}J(\Theta)\right]$ and the known relative joint rates $\left\{\dot{\Theta}\right\}$. Note the Jacobian matrix is a function of the known joint values Θ (angles θ_i for **R** joints and lengths d_i for **P** joints).

7.5 Resolved-Rate Control Algorithm

Another useful application for the Jacobian matrix is the **Inverse Velocity Problem**, whose solution is the basis for the **Resolved-Rate Control Algorithm**[8]. The Inverse Velocity Problem is stated, in general:

Given: the robot (including all constant DH Parameters)

values for all of the joint variables $\{\Theta\}$
(angles θ_i for **R** joints and lengths d_i for **P** joints)

the end-effector Cartesian velocity $\{\dot{X}\}$ (translational and rotational)

Find: all of the relative joint rates $\{\dot{\Theta}\}$
(joint rates $\dot{\theta}_i$ for **R** joints and $\dot{d}_i$ for **P** joints)

The Inverse Velocity Solution is:

where we calculate the required relative joint rates $\{\dot{\Theta}\}$ to achieve the given desired Cartesian velocities $\{{}^k\dot{X}\}$, using the inverse of the configuration-dependent Jacobian matrix $[{}^kJ(\Theta)]$. This works for an $m = n$ square matrix, assuming full rank, i.e. the Jacobian matrix determinant is not zero. Here are two options to solve the Inverse Velocity problem in MATLAB.

- Numerical or symbolical Jacobian matrix inversion

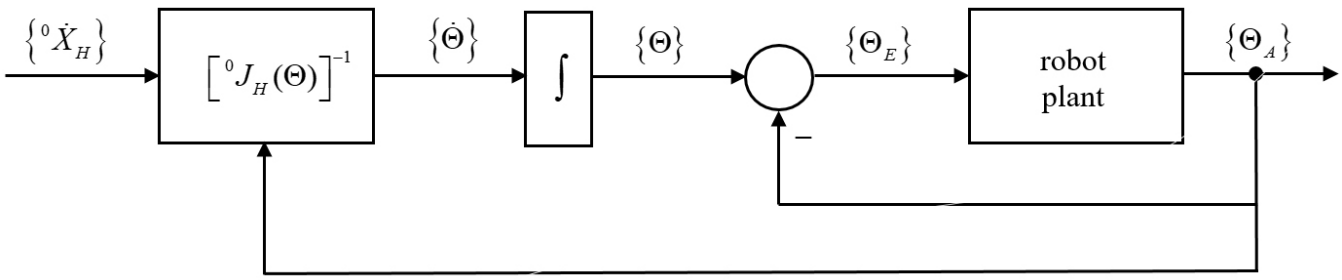
```
thd = inv(J)*Xd      % Solution via matrix inverse
```

- Numerical Gaussian elimination is more efficient and robust.

```
thd = J\Xd           % Solution via Gaussian elimination
```

[8] D.E. Whitney, 1969, Resolved Motion Rate Control of Manipulators and Human Prostheses, IEEE Trans on Man-Machine Systems.

Resolved-Rate Control Block Diagram



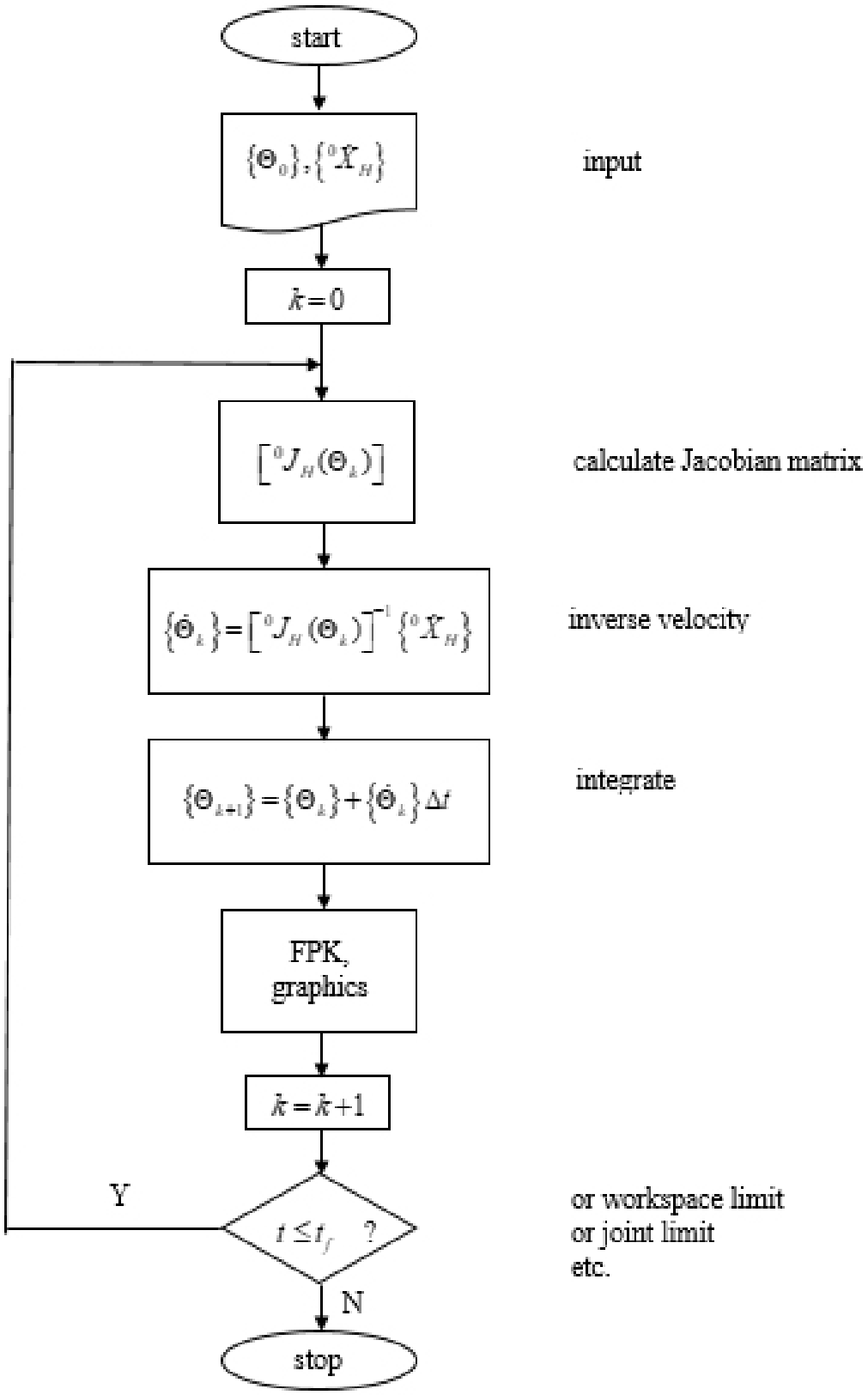
Resolved-Rate Control Block Diagram – joint control details

Crucial - Units

The translational rows of the Jacobian matrix have length units (m). The rotational rows of the Jacobian matrix are unitless. Therefore, since Cartesian velocity units of $\{\dot{X}\}$ are m/sec and rad/sec, in the overall velocity equation $\{\dot{X}\} = [J]\{\dot{\Theta}\}$, one MUST use units of rad/sec for $\{\dot{\Theta}\}$, not deg/sec!

Further, the units and size mismatch between translational and rotational rows of the Jacobian matrix can cause numerical troubles, which is well-known for serial robots.

Resolved-Rate Algorithm Flowchart



7.6 Robot Singularities

Robot singularities occur when the Jacobian matrix loses full rank. At a singularity the **Inverse Velocity Problem** (see the **Resolved-Rate Algorithm** section above) cannot be solved reliably since it requires division by zero, yielding infinite joint rates.

At a singular configuration, the robot has lost freedom to move in one or more Cartesian directions (translations and/or rotations).

Singular configurations in the velocity domain also result in a singular solution in the inverse pose domain.

Singularities are classified into 2 types. 1) Workspace boundary singularities, such as when the elbow is straight out (or folded onto itself), and 2) Workspace interior singularities, such as when two wrist axes align during robot motion.

There is a velocity / force duality for singularities. At a singularity, a freedom to move has been lost in a certain direction, but in the same direction, a force (or moment) may be resisted with zero actuator torque.

Another way to put it, at a singularity, infinite joint rates are theoretically required to produce finite Cartesian motion of the end-effector. Also, zero joint torque is required to resist a force or moment (instead the manipulator structure resists the force/moment, not the actuators).

Singular configurations arise when $|J| = 0$ for square Jacobian matrices. As the Jacobian matrix determinant approaches zero, the Jacobian matrix becomes ill-conditioned. One method to calculate the inverse of a matrix is shown below; $[J]^{-1}$ does not exist at $|J| = 0$, and it is unreliable near $|J| = 0$.

$$[J]^{-1} = \frac{\text{adjoint}[J]}{|J|}$$

where $|J|$ is the determinant of [*J*].

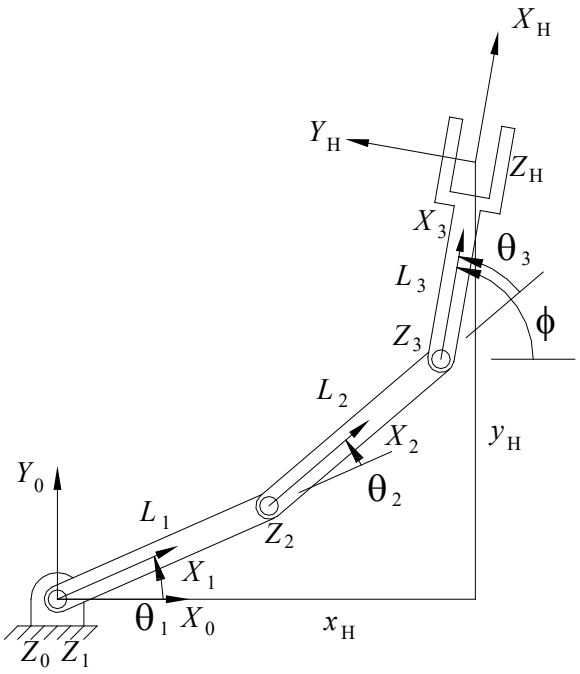
$$[\text{adjoint}(J)] = [\text{cofactor}(J)]^T$$

cofactor(*J*) $c_{ij} = (-1)^{i+j} |J_{ij}|$

minor minor $|J_{ij}|$ is the determinant of the submatrix of [*J*] with row *i* and column *j* removed.

Singularity Analysis Example

Recall that for the planar 3R robot, the Jacobian matrix derived earlier is:



$$\left[{}^0J_H\right] = \begin{bmatrix} -L_1s_1 - L_2s_{12} - L_3s_{123} & -L_2s_{12} - L_3s_{123} & -L_3s_{123} \\ L_1c_1 + L_2c_{12} + L_3c_{123} & L_2c_{12} + L_3c_{123} & L_3c_{123} \\ 1 & 1 & 1 \end{bmatrix}$$

For this somewhat complicated expression, the Jacobian matrix determinant is amazingly simple:

$$\left|{}^0J_H\right| = L_1L_2\sin\theta_2$$

The singularity conditions for this robot are derived from:

$$\left|{}^0J_H\right| = L_1L_2\sin\theta_2 = 0$$

$L_1 = 0$
$L_2 = 0$ (not possible)

$\sin\theta_2 = 0$; $\theta_2 = 0, 180^\circ, \ldots$

It is meaningful to plot the Jacobian determinant in such a simple case:

Where are the min and max for this function? Where is the determinant zero (already stated above)?

7.7 Cartesian Wrench / Joint Torques Statics Transformation

In this section we derive the relationship between static Cartesian wrenches (forces / moments) applied to the environment by the robot end-effector and the required robot joint torques to do this.

Principle of virtual work. If a body is in static equilibrium, the total virtual work of the forces acting on the body is zero for any virtual displacement of the body. Recall that work is a scalar term found via dot product of force and displacement vectors, with *Nm* units.

work virtual work

where $\{\delta s\}$ is an infinitesimal virtual vector displacement.

In particular, we can equate a virtual work expression in the **joint space** with a virtual work expression in the **Cartesian space**, since they are zero.

Where $\{\tau\} \equiv n \times 1$ is the vector of n joint torques, $\{\delta\theta\} \equiv n \times 1$ is the vector of n infinitesimal joint angle displacements, $\{W\} \equiv m \times 1$ is the Cartesian wrench (m forces and moments) vector, $\{\delta X\} \equiv m \times 1$ is the vector of m infinitesimal Cartesian displacements (translational and rotational – due to the non-holonomic nature of 3D rotations, this only works for infinitesimal virtual angular displacements), n is the joint space dimension, and m is the Cartesian space dimension.

Express the dot product using matrix multiplication and the transpose:

Multiplying both sides of the **forward velocity solution** $\{\dot{X}\} = [J]\{\dot{\theta}\}$ by δt yields:

Substituting into the above equation yields:

Which must hold for all infinitesimal virtual angular displacements $\{\delta\theta\}$ and so:

Finally, transposing both sides yields:

The Jacobian matrix J and Cartesian wrench W must be expressed in the basis coordinates of the same frame $\{k\}$. Just like the joint rates, $\{\tau\}$ has no dependence on frame, since these are relative joint torques about the n Z axes. $\{\tau\} = [J]^T \{W\}$ calculates the joint torque directions to apply $\{W\}$, not resist $\{W\}$. That is, $\{\tau\} = [J]^T \{W\}$ causes the end-effector to exert force/torque on the environment with Cartesian wrench W (the environment exerts back onto the robot end-effector with an equal-and-opposite wrench W).

The Jacobian matrix $[J]$ for static torque calculations is the same as that for velocity analysis. This statics transformation is a mapping from Cartesian space to joint space which does not require an inverse. That is indeed a rare and beautiful property. It can never be singular and any number of joints is allowed. Very little computation is required compared to matrix inversion, since only matrix transposition and multiplication is required.

7.8 Velocity Kinematics Example

Planar 3R Robot

Given $L_1 = 3, L_2 = 2, L_3 = 1$, $\theta_1 = 15^\circ, \theta_2 = 25^\circ, \theta_3 = 35^\circ$, and $\dot{\theta}_1 = 1, \dot{\theta}_2 = 2, \dot{\theta}_3 = 3$

a) Forward Velocity Kinematics

The Jacobian matrix giving the velocity of {3} moving with respect to {0} has simpler analytical terms than the Jacobian matrix of {*H*} moving with respect to {0}.

$$\left\{ {}^0\dot{X}_3 \right\} = \left[{}^0J_3 \right] \left\{ \dot{\Theta} \right\} = \begin{bmatrix} -2.062 & -1.286 & 0 \\ 4.430 & 1.532 & 0 \\ 1 & 1 & 1 \end{bmatrix} \begin{Bmatrix} 1 \\ 2 \\ 3 \end{Bmatrix} = \begin{Bmatrix} -4.634 \\ 7.494 \\ 6 \end{Bmatrix}$$

Now we can find the translational and rotational velocities at the {*H*} frame (see Section 7.9).

$${}^0\left\{ {}^0V_H \right\} = {}^0\left\{ {}^0V_3 \right\} + {}^0\left\{ {}^0\omega_3 \right\} \times L_3 \left\{ {}^0\hat{X}_3 \right\} = \begin{Bmatrix} -4.634 \\ 7.494 \end{Bmatrix} + \begin{Bmatrix} -5.796 \\ 1.553 \end{Bmatrix} = \begin{Bmatrix} -10.430 \\ 9.047 \end{Bmatrix}$$

$${}^0\left\{ {}^0\omega_H \right\} = {}^0\left\{ {}^0\omega_3 \right\} = 6$$

b) Inverse Velocity Kinematics (analytical result)

$$\left[{}^0J_3 \right] = \begin{bmatrix} -L_1 s_1 - L_2 s_{12} & -L_2 s_{12} & 0 \\ L_1 c_1 + L_2 c_{12} & L_2 c_{12} & 0 \\ 1 & 1 & 1 \end{bmatrix} \qquad \left[{}^0J_3 \right]^{-1} = \begin{bmatrix} \dfrac{c_{12}}{L_1 s_2} & \dfrac{s_{12}}{L_1 s_2} & 0 \\ \dfrac{-L_1 c_1 - L_2 c_{12}}{L_1 L_2 s_2} & \dfrac{-L_1 s_1 - L_2 s_{12}}{L_1 L_2 s_2} & 0 \\ \dfrac{c_1}{L_2 s_2} & \dfrac{s_1}{L_2 s_2} & 1 \end{bmatrix}$$

$$\left\{ \dot{\Theta} \right\} = \left[{}^0J_3 \right]^{-1} \left\{ {}^0\dot{X}_3 \right\} = \begin{bmatrix} 0.604 & 0.507 & 0 \\ -1.747 & -0.813 & 0 \\ 1.143 & 0.306 & 1 \end{bmatrix} \begin{Bmatrix} -4.634 \\ 7.494 \\ 6 \end{Bmatrix} = \begin{Bmatrix} 1 \\ 2 \\ 3 \end{Bmatrix}$$

c) Singularity Analysis

The Jacobian matrix determinant is invariant under coordinate transformations.

$$\left|{}^0J\right| = \left|{}^1J\right| = \left|{}^2J\right| = \left|{}^3J\right| = L_1 L_2 \sin\theta_2$$

Singularity conditions

$L_1 = 0$
$L_2 = 0$ (not possible) $\qquad \sin\theta_2 = 0; \qquad \theta_2 = 0, 180^\circ, \ldots$

Example

$\left|{}^0J\right| = 3 \cdot 2 \cdot \sin(25^\circ) = 2.54$. It is more meaningful to plot the Jacobian determinant.

Two interesting cases

min	$\left\|J\right\| = 0$	$\theta_2 = 0$
max	$\left\|J\right\| = 6$	$\theta_2 = 90^\circ$

d) Static Wrench Analysis

Given $L_1 = 3, L_2 = 2, L_3 = 1$, $\theta_1 = 15^\circ, \theta_2 = 25^\circ, \theta_3 = 35^\circ$, and $\left\{{}^0W\right\}$, calculate $\left\{\tau\right\}$.

$$\left\{\tau\right\} = \left[{}^0J\right]^T \left\{{}^0W\right\} \qquad \begin{Bmatrix}\tau_1\\ \tau_2\\ \tau_3\end{Bmatrix} = \left[{}^0J\right]^T \begin{Bmatrix}f_x\\ f_y\\ m_z\end{Bmatrix}$$

a) Force $\qquad \left\{\tau\right\} = \begin{bmatrix}-2.062 & 4.430 & 1\\ -1.286 & 1.532 & 1\\ 0 & 0 & 1\end{bmatrix}\begin{Bmatrix}1\\ 1\\ 0\end{Bmatrix} = \begin{Bmatrix}2.368\\ 0.246\\ 0\end{Bmatrix}$

b) Moment $\qquad \left\{\tau\right\} = \begin{bmatrix}-2.062 & 4.430 & 1\\ -1.286 & 1.532 & 1\\ 0 & 0 & 1\end{bmatrix}\begin{Bmatrix}0\\ 0\\ 1\end{Bmatrix} = \begin{Bmatrix}1\\ 1\\ 1\end{Bmatrix}$

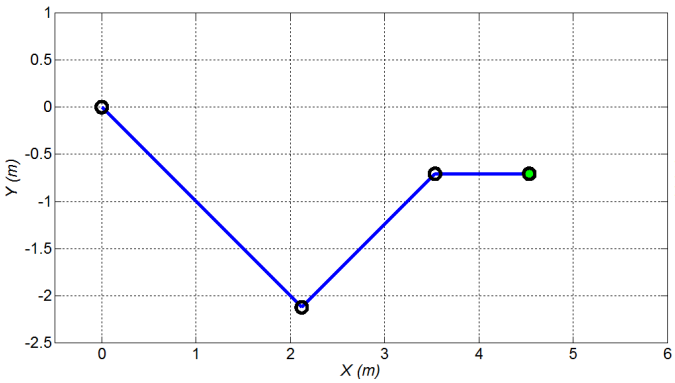
Why are all three b joint torques 1 *Nm*? All joint torques must resist against the previous neighbor link, which can be seen by drawing the link free-body diagrams.

c) Force and moment

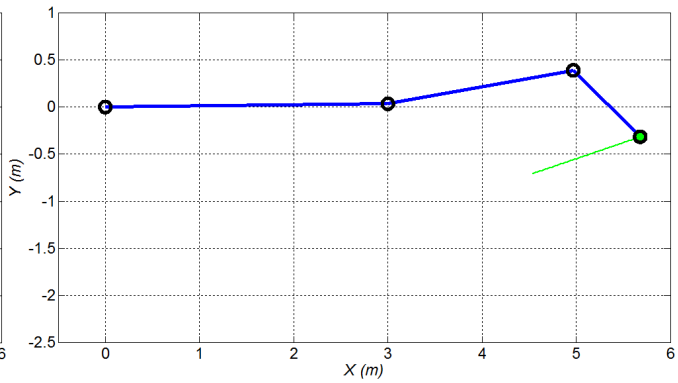
$$\{\tau\} = \begin{bmatrix} -2.062 & 4.430 & 1 \\ -1.286 & 1.532 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} 1 \\ 1 \\ 1 \end{Bmatrix} = \begin{Bmatrix} 3.368 \\ 1.246 \\ 1 \end{Bmatrix}$$

Resolved-Rate Control Examples

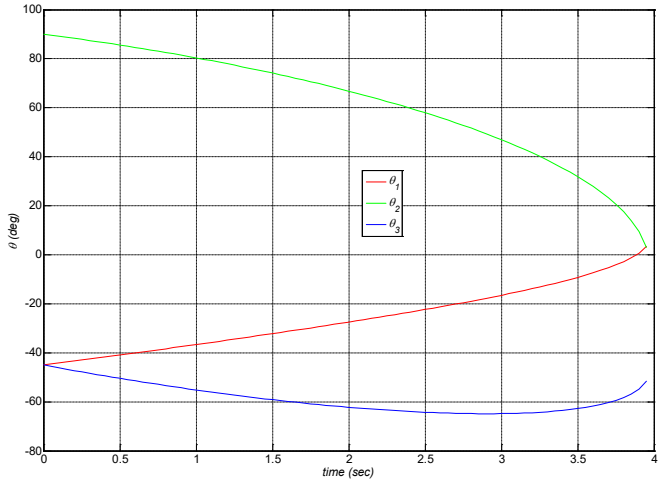
1. Simulate resolved-rate control for the planar 3R 3-dof robot, given robot lengths $L_1 = 3$, $L_2 = 2$, $L_3 = 1$ (m), initial joint angles $\{\Theta\} = \{\theta_1 \quad \theta_2 \quad \theta_3\}^T = \{-45^\circ \quad 90^\circ \quad -45^\circ\}^T$, and the constant commanded Cartesian rates $\{{}^0\dot{X}\} = \{\dot{x}_H \quad \dot{y}_H \quad \omega_z\}^T = \{0.3 \quad 0.1 \quad -0.2\}^T$ $(m/s$ and $rad/s)$. Simulate for 3.9 *sec*, using time steps of $dt = 0.05$ *sec*. Also simulate the inverse statics solution, i.e. calculate the joint torques $\{\mathrm{T}\} = \{\tau_1 \quad \tau_2 \quad \tau_3\}^T$ (Nm) given the Cartesian wrench $\{{}^0W\} = \{f_x \quad f_y \quad m_z\}^T = \{3 \quad 2 \quad 1\}^T$ (N and Nm, constant).



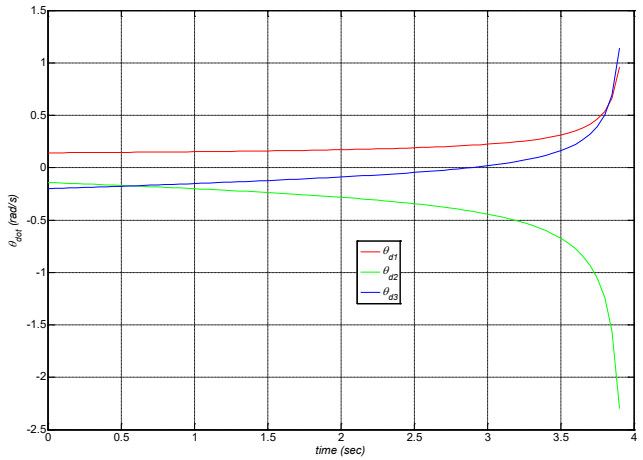
Initial Robot Pose



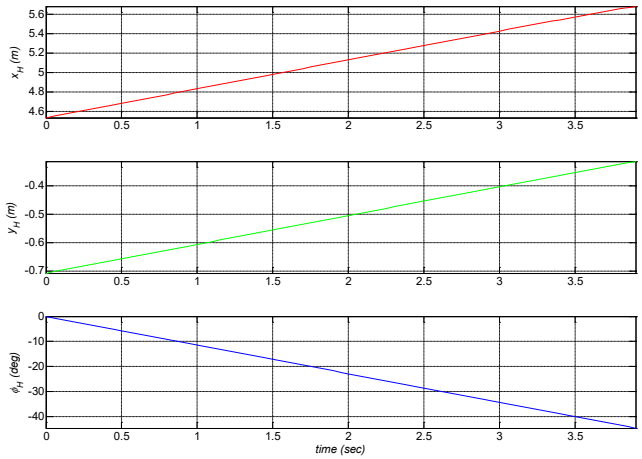
Final Robot Pose



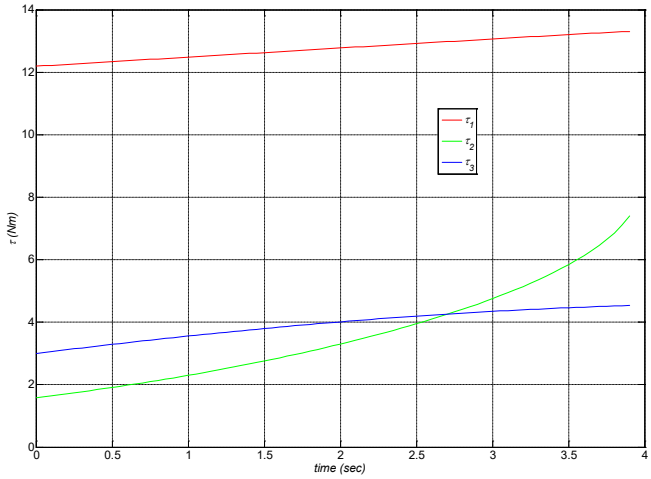
Robot Joint Angles



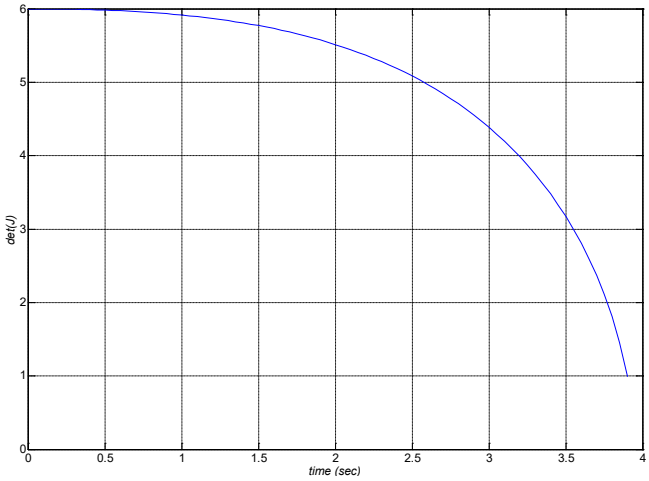
Robot Joint Rates



Robot Cartesian Pose



Robot Joint Torques



Jacobian Matrix Determinant

We see that the planar 3R robot is rapidly approaching the $\theta_2 = 0$ elbow-out singularity at the end of this simulated motion (see the final robot pose figure and the Jacobian matrix determinant plot above). The Jacobian matrix determinant plot is plunging towards zero at the end of this simulation time and the slopes of the joint angles and joint angle rates plots are increasing significantly, which present significant problems for a real robot.

The inverse velocity solution yields infinite joint rates at singularities and unreliably high joint rates in the neighborhood of singularities.

Planar 3R Robot Resolved-Rate Simulation Example 2

In the Planar 3R Robot IPK Section 5.2 we presented a Planar 3R Robot example where the Cartesian motions x_H, y_H, ϕ were all be chopped into equal segments, providing straight-line motions in the Cartesian space. Let us now repeat this same example, but using the resolved-rate method rather than IPK.

Given: $L_1 = 3, \quad L_2 = 2, \quad L_3 = 1$ m

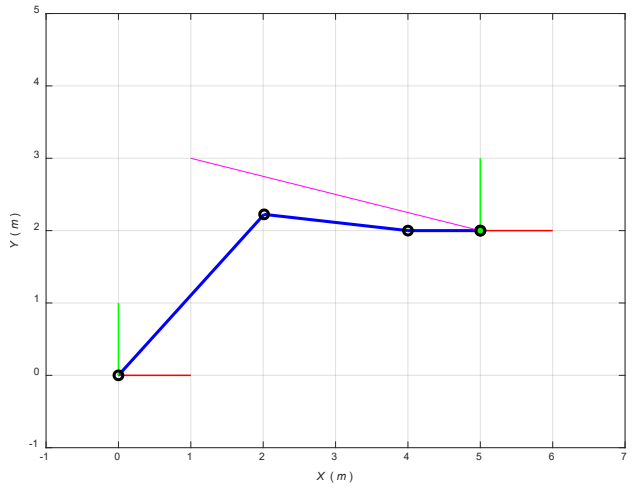
initial pose: $\{{}^0X_H\} = \{5 \quad 2 \quad 0^\circ\}$ m and deg

final pose: $\{{}^0X_H\} = \{1 \quad 3 \quad 90^\circ\}$

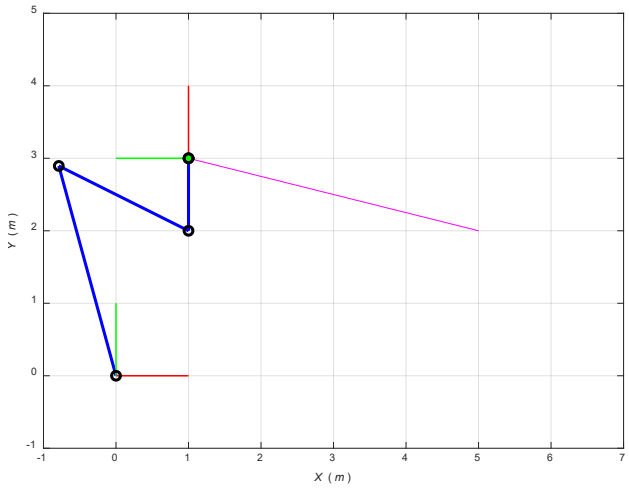
Calculate:

the required time histories of $(\theta_1, \theta_2, \theta_3)$, using resolved-rate, not IPK.

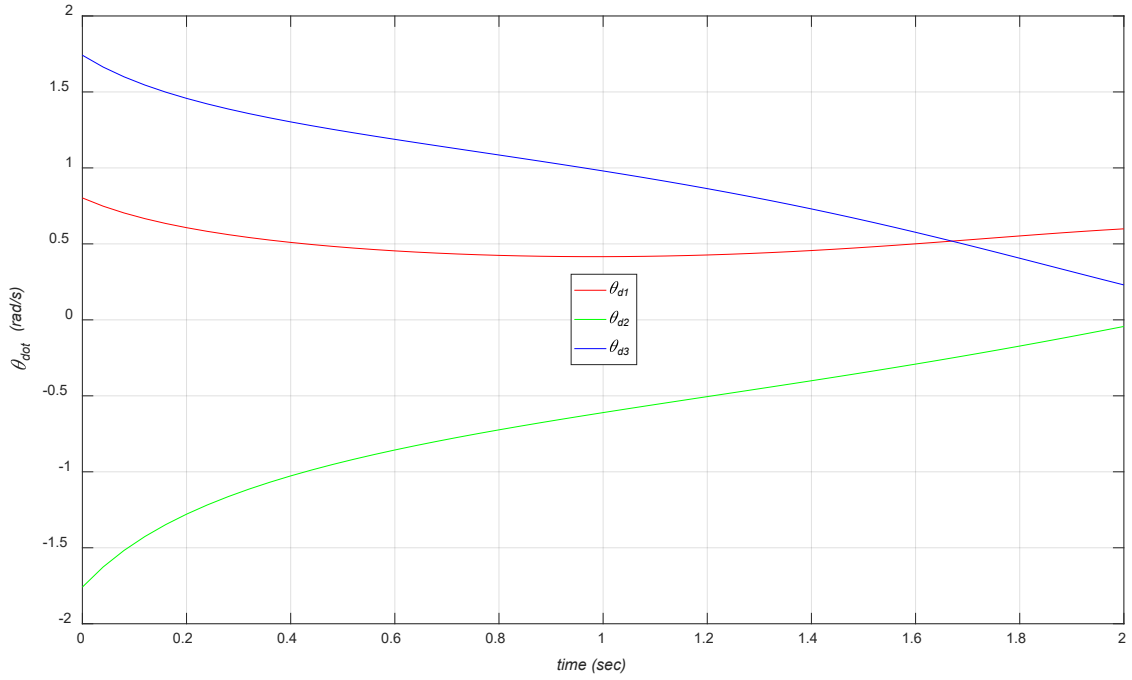
The initial joint angles, from IPK, are $\theta_1 = 47.8645, \theta_2 = -54.3147, \theta_3 = 6.4502$, corresponding to the elbow-up branch. Subtracting the initial pose from the final pose and dividing by the 2 sec time range yields $\{{}^0\dot{X}_H\} = \{-2 \quad 0.5 \quad 0.7854\}$ m/s and rad/s. **CAUTION** – for the rotational portion, this may only be done for planar motion, NOT 3D motion with Euler Angles. Using $N = 50$ steps, we find $\Delta t = 0.04$ sec. The graphics below show the simulated results.



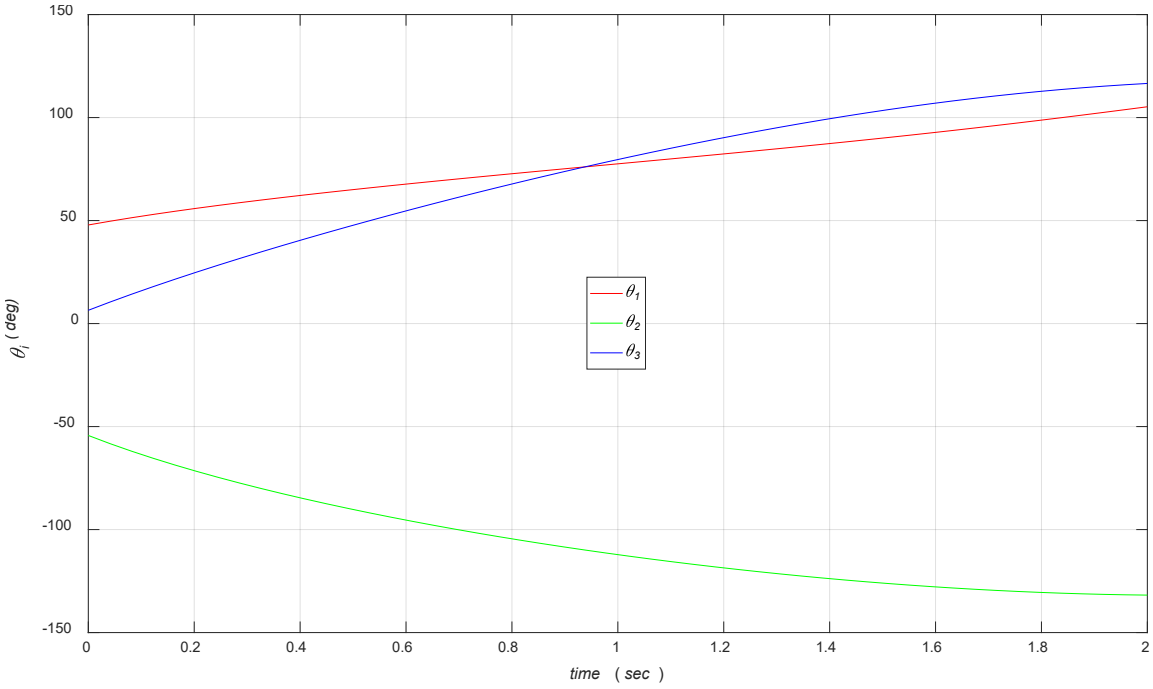
Initial Pose



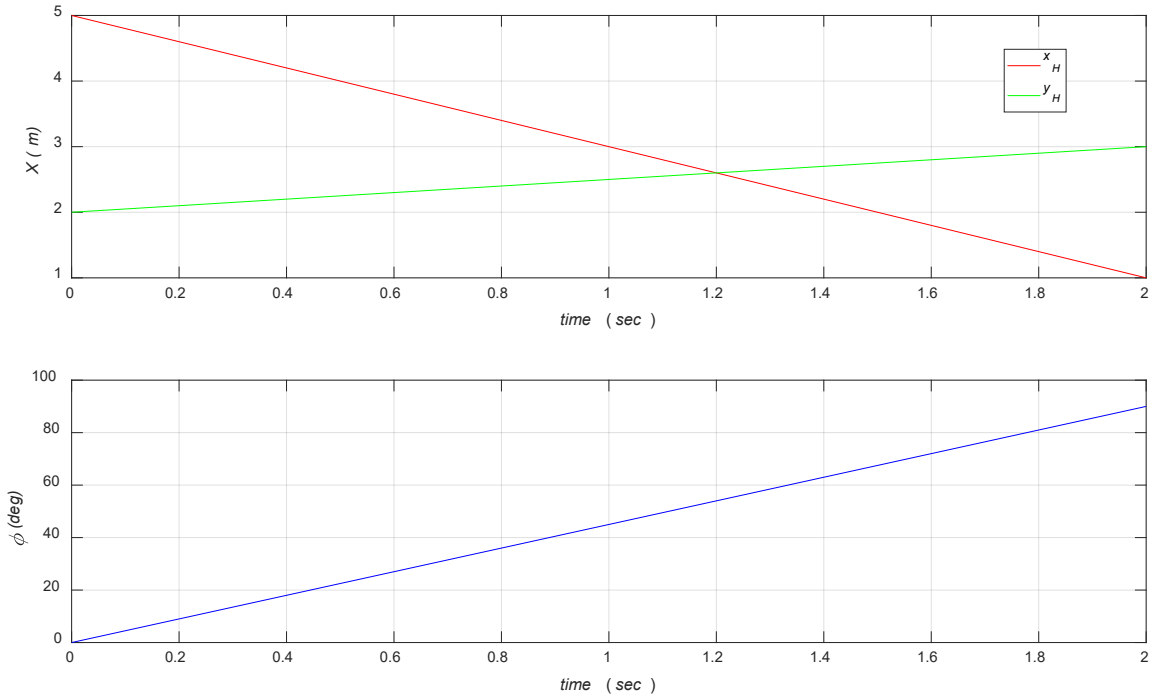
Final Pose



Required Joint Rates



Required Joint Angles



Commanded Cartesian Motion

Assuming Δt is sufficiently small, the required joint angles results in this resolved-rate example are IDENTICAL to those from the IPK simulation presented in Section 5.2, because the commanded Cartesian motion was identical, found through $\left\{ {}^{0}\dot{X}_{H} \right\}$ in the current example.

8. Kinematically-Redundant Robots (KRRs)

A serial kinematically-redundant robot (KRR) is one that has more active joints than is minimally-required to accomplish the Cartesian task. **Redundancy** indicates extra, more than required.

8.1 Introduction

3 types of electromechanical redundancy

Fail-safe	two motors at a joint; if one fails, use the second (e.g. NASA FTS)
Actuation	more motors than *dof*; antagonistic loading to increase stiffness
Kinematic	more *dof* than required for the task; optimization possible

From **biomechanics**, the human displays all three types of robot redundancy.

Kinematic Redundancy

In this class we will focus only on kinematic redundancy. With goals of increased **dexterity and autonomy**, the next generation of robots should be kinematically-redundant for increased capability.

Wider application with KRRs. Industrial robot workcells are currently set up carefully to avoid obstacles, joint limits, robot singularities, etc. This is costly, time-consuming, and limiting. More rigorous applications in industrial robotics, plus remote robot operations in space, undersea, and nuclear power plant environments, plus service industry robots (maid, helper, delivery, hospital, etc.) would all benefit from kinematically-redundant robots (KRRs).

Performance Optimization with KRRs. The engineer can optimize robot performance in addition to satisfying required motion trajectories with the extra freedom available in a KRR. Here is a partial list of possible optimization tasks (combinations of these are also possible).

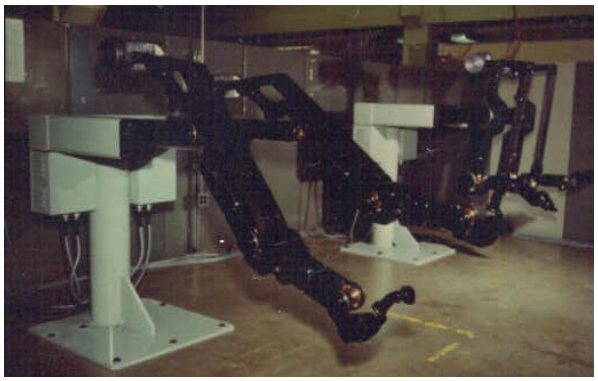
- avoid obstacles
- avoid joint limits
- avoid singularities
- minimize robot energy
- minimize joint torques
- minimize trajectory time
- minimize base shaking force (important in space robots)

local optimization – optimization in the neighborhood of the current robot configuration
global optimization – optimization over the entire robot motion range (min or max)

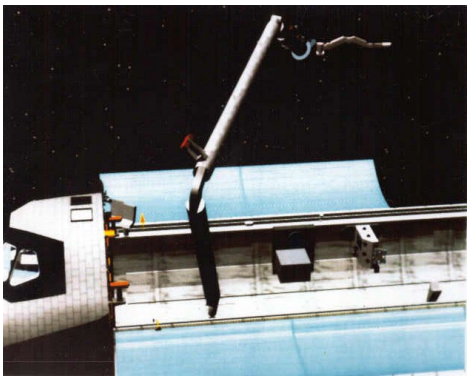
Kinematically-Redundant Robot System Examples (see the following page)

- redundant arm – 7-dof (human, LTM, FTS, RRC), 8-dof (ARMII)
- humanoid – Special Purpose Dexterous Manipulator (SPDM) Canada; two 7-dof arms on a 5-dof trunk including waist and neck joints, comprising a 19-dof system, Robonaut
- compound robot – Space Station Remote Manipulator System (SSRMS) with SPDM or Robonaut
- arm on track – industrial robot on a linear track significantly extends the workspace
- arm with moving workcell – 5-dof arm (axisymmetric) with 2-dof table
- cooperating serial robots forming a closed-chain parallel robot by gripping the same workpiece

SSRMS with SPDM on International Space Station



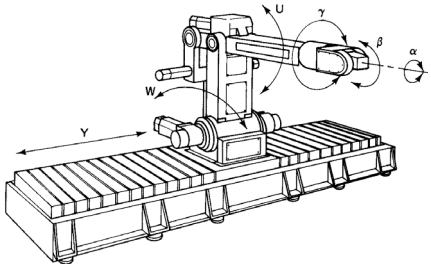
Laboratory Telerobotic Manipulator (LTM)



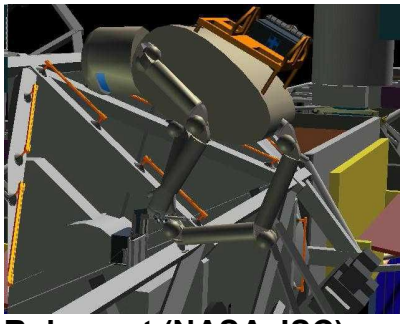
SRMS with Flight Telerobotic Servicer (FTS)

7-dof RRC

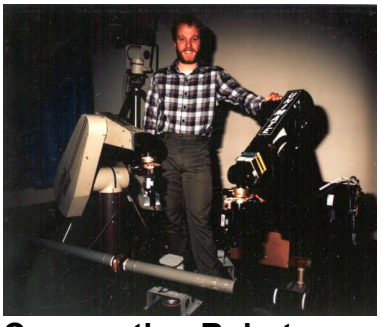
8-dof ARMII (NASA LaRC)



Industrial Robot on Track



Robonaut (NASA JSC)



Cooperating Robots

Artificial Muscles

Kinematically-Redundant Robot A KRR has more joint freedoms than required for the Cartesian task. That is, the dimension of the joint space *n* is greater than the dimension of Cartesian task space *m*. The general velocity relationship is:

$m < n$ degree of redundancy = $n - m$

The inverse velocity problem is ***underconstrained***, that is, infinite valid solutions exist (actually, $n - m$ infinities of solutions exist).

Self-motion (human arm) – optimization takes place in joint rate combinations such that $\{\dot{X}\} = \{0\}$.

Primary Task	Particular Solution	achieves required trajectory $\{\dot{X}\}$
Secondary Task	Homogeneous Solution	optimization of some performance criteria in null space (self-motion) $\{\dot{X}\} = \{0\}$

8.2 Inverse Velocity (Resolved-Rate) Solution

8.2.1 Pseudoinverse-based

Particular Solution

The particular solution satisfies the **Primary Task**, i.e. the commanded Cartesian trajectory $\{\dot{X}\}$. The particular solution is obtained from the minimum joint rate problem, stated as follows. That is, from the infinite possible solutions we will choose the unique solution with the minimum joint rates.

Minimize the scalar cost function $f = \frac{1}{2}\{\dot{\Theta}\}^T \{\dot{\Theta}\}$

Subject to the rate equations $\{\dot{X}\} = [J]\{\dot{\Theta}\}$

The **particular solution** is:

where k_P is a scalar gain and $[J]^*$ is the underconstrained **Moore-Penrose pseudoinverse** of the Jacobian matrix:

$[J]^*$ can be calculated by the MATLAB function **`pinv`**.

Note that:

$$[J][J]^* = [I_m] \qquad \text{but} \qquad [J]^*[J] \neq [I_n]$$

so the underconstrained Moore-Penrose pseudoinverse is called the right inverse of $[J]$.

Note that the left inverse of an overconstrained matrix may be used to find the least-squares solution, where there is no solution, but a solution with the smallest possible error is found (for data fitting). This is opposite to the KRR problem we are considering.

Homogeneous Solution

The homogeneous solution satisfies the **Secondary Task**, which can be used for optimization to avoid obstacles, avoid joint limits, and avoid singularities, among other secondary tasks. The homogeneous solution doesn't affect the Cartesian trajectory: $\{\dot{\Theta}_H\}$ causes $\{\dot{X}\} = \{0\}$. The homogeneous solution projects an arbitrary vector *z* into the null-space of the Jacobian matrix.

The **homogeneous solution** is:

where k_H is a scalar gain.

Total Solution

The total solution to the underconstrained KRR inverse velocity problem is the sum of the **particular** and **homogeneous solutions** (obtained via linear superposition):

KRR Statics Transformation

For KRRs, the statics transformation derivation is unchanged. That is, the Jacobian matrix transpose still works to transform the desired Cartesian wrench into the required joint torques, an inverse solution without inverting a matrix (check the dimensions below for a KRR).

$$\{\tau\} = [J]^T \{W\} = [J]^T \begin{Bmatrix} F \\ M \end{Bmatrix}$$

8.2.2 Kinematically-Redundant Robot Singularities

The Moore-Penrose pseudoinverse is still subject to singularities. **Singular Value Decomposition (SVD)**, presented below, can be used in the vicinity of singularities to obtain a reliable numerical pseudoinverse to move through singularities, but it will not change the physical robot singularity (loss of motion).

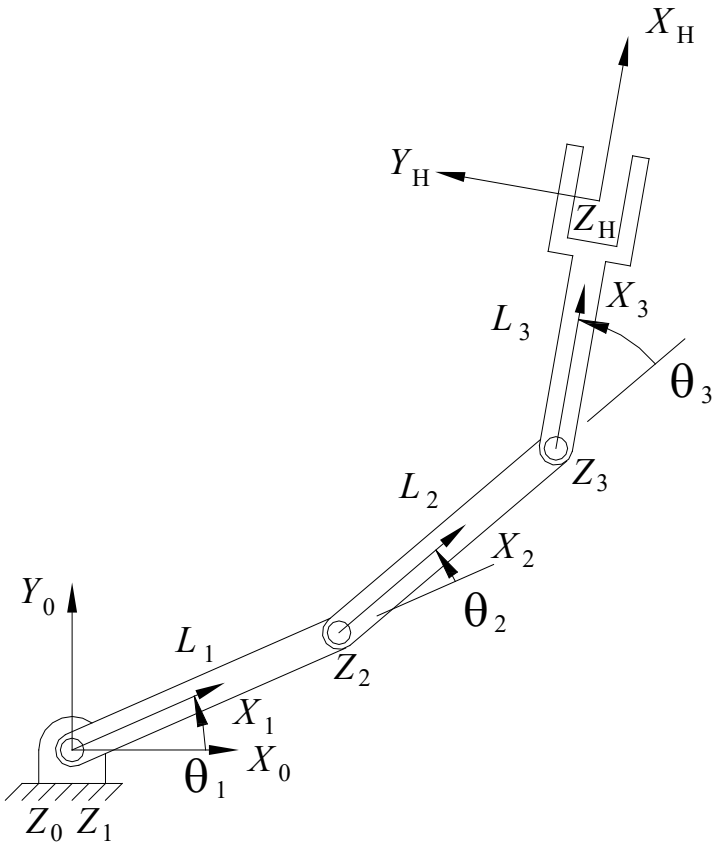
A kinematically-redundant robot is singular when the following determinant is zero.

$$\left| [J][J]^T \right| = 0$$

due to the previously-given definition of the underconstrained Moore-Penrose pseudoinverse $[J]^*$.

8.3 Kinematically-Redundant Robot Examples

Example 1. Planar Kinematically-Redundant 3-dof 3R Robot with *m* = 2



This KRR example uses the resolved-rate algorithm: solve $\{\dot{\Theta}\}$ from $\{\dot{X}\} = [J]\{\dot{\Theta}\}$.

For this **snapshot example**, a three-link 3R 3-dof planar robot is required to satisfy only 2-dof translations (no 1-dof rotations). The underconstrained Jacobian matrix for this case is 2x3 (giving the translational velocity of the hand frame {*H*} origin with respect to the base frame {0} origin, expressed in {0} coordinates; *m*=2 and *n*=3).

$$\left[{}^0J\right] = \begin{bmatrix} -L_1 s_1 - L_2 s_{12} - L_3 s_{123} & -L_2 s_{12} - L_3 s_{123} & -L_3 s_{123} \\ L_1 c_1 + L_2 c_{12} + L_3 c_{123} & L_2 c_{12} + L_3 c_{123} & L_3 c_{123} \end{bmatrix}$$

For a **numerical example**, let $L_1 = L_2 = L_3 = 1$ and $\theta_1 = 60^\circ, \theta_2 = -60^\circ, \theta_3 = 30^\circ$. The required trajectory (instantaneous Cartesian rates for this snapshot example) is $\{\dot{X}\} = \{1 \quad 1\}^T$.

$$\left[{}^0J\right] = \begin{bmatrix} -1.366 & -0.500 & -0.500 \\ 2.366 & 1.866 & 0.866 \end{bmatrix}$$

Particular Solution

$$\left\{\dot{\Theta}_P\right\} = k_P \left[J\right]^* \left\{\dot{X}\right\} = k_P \left[J\right]^T \left[\left[J\right]\left[J\right]^T\right]^{-1} \left\{\dot{X}\right\}$$

$$\left[J\right]\left[J\right]^T = \begin{bmatrix} 2.366 & -4.598 \\ -4.598 & 9.830 \end{bmatrix} \qquad \left[\left[J\right]\left[J\right]^T\right]^{-1} = \begin{bmatrix} 4.646 & 2.173 \\ 2.173 & 1.118 \end{bmatrix} \qquad \text{both are symmetric}$$

$$\left[J\right]^* = \begin{bmatrix} -1.205 & -0.323 \\ 1.732 & 1.000 \\ -0.441 & -0.118 \end{bmatrix} \qquad k_P = 1$$

$$\left\{\dot{\Theta}_P\right\} = \begin{bmatrix} -1.205 & -0.323 \\ 1.732 & 1.000 \\ -0.441 & -0.118 \end{bmatrix} \begin{Bmatrix} 1 \\ 1 \end{Bmatrix} = \begin{Bmatrix} -1.527 \\ 2.732 \\ -0.559 \end{Bmatrix}$$

check: $\left[J\right]\left\{\dot{\Theta}_P\right\} = \begin{Bmatrix} 1 \\ 1 \end{Bmatrix}$ as required.

This $\left\{\dot{\Theta}_P\right\}$ is the least-norm solution for joint rates. To check, find another solution and see if the norm is greater, as it must be. The simplest way to find another solution (out of the infinity of solutions) is to lock a joint and solve for the other two joint rates. Locking joint 2 arbitrarily yields a singular reduced Jacobian matrix for all motion (elbow joint is particular-only). So instead choose to lock joint 1.

$\dot{\theta}_1 = 0$, remove column 1 from the Jacobian matrix; this reduced problem is no longer redundant.

$$\begin{Bmatrix} 1 \\ 1 \end{Bmatrix} = \begin{bmatrix} -0.500 & -0.500 \\ 1.866 & 0.866 \end{bmatrix} \begin{Bmatrix} \dot{\theta}_2 \\ \dot{\theta}_3 \end{Bmatrix}$$

$$\begin{Bmatrix} \dot{\theta}_2 \\ \dot{\theta}_3 \end{Bmatrix} = \begin{bmatrix} 1.732 & 1.000 \\ -3.732 & -1.000 \end{bmatrix} \begin{Bmatrix} 1 \\ 1 \end{Bmatrix}$$

$$\left\{\dot{\Theta}_P\right\} = \begin{Bmatrix} 0 \\ 2.732 \\ -4.732 \end{Bmatrix}$$

This particular solution has a larger Euclidean norm than that associated with the pseudoinverse solution: 5.464 > 3.179. ***Q.E.D.*** Note the elbow joint angular rate $\dot{\theta}_2$ is unchanged.

Homogeneous Solution

$$\left\{\dot{\Theta}_H\right\} = k_H\left[\left[I_n\right]-\left[J\right]^*\left[J\right]\right]\{z\}$$

The Moore-Penrose pseudoinverse is a right-sided inverse, that is:

$$\left[J\right]\left[J\right]^* = \left[I_m\right], \text{ but } \left[J\right]^*\left[J\right] \neq \left[I_n\right]$$

$$\left[J\right]^*\left[J\right] = \begin{bmatrix} 0.882 & 0 & 0.323 \\ 0 & 1 & 0 \\ 0.323 & 0 & 0.118 \end{bmatrix}$$

for this example.

The symmetric null-space-projection matrix is:

$$\left[\left[I_3\right]-\left[J\right]^*\left[J\right]\right] = \begin{bmatrix} 0.118 & 0 & -0.323 \\ 0 & 0 & 0 \\ -0.323 & 0 & 0.882 \end{bmatrix}.$$

This matrix projects an arbitrary vector z into the null-space of the Jacobian matrix, meaning the resulting homogeneous solution $\left\{\dot{\Theta}_H\right\}$ maps through [J] to the zero vector (i.e. causing zero end-effector Cartesian motion). Using an arbitrary $\{z\}$ (not attempting to optimize anything), with $k_H = 0.5$:

$$\left\{\dot{\Theta}_H\right\} = 0.5\begin{bmatrix} 0.118 & 0 & -0.323 \\ 0 & 0 & 0 \\ -0.323 & 0 & 0.882 \end{bmatrix}\begin{Bmatrix} 1 \\ 1 \\ 1 \end{Bmatrix} = \begin{Bmatrix} -0.102 \\ 0 \\ 0.280 \end{Bmatrix}$$

check: $\left[J\right]\left\{\dot{\Theta}_H\right\} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}$ as required.

The homogeneous term for the elbow joint will always be zero. A secondary solution cannot be added because only the elbow controls the distance from the shoulder to the wrist. The physical interpretation of the null space for this simple example is like a planar 4-bar mechanism: If we fix the position x,y of the hand, this is just like having a second fixed pivot. But angle ϕ of the hand is not constrained, so the self-motion of the 3-link planar robot is just the motion of the 4-bar mechanism through its range of motion. Infinite different combinations of joint angles $\theta_1, \theta_2, \theta_3$ yield the same x,y.

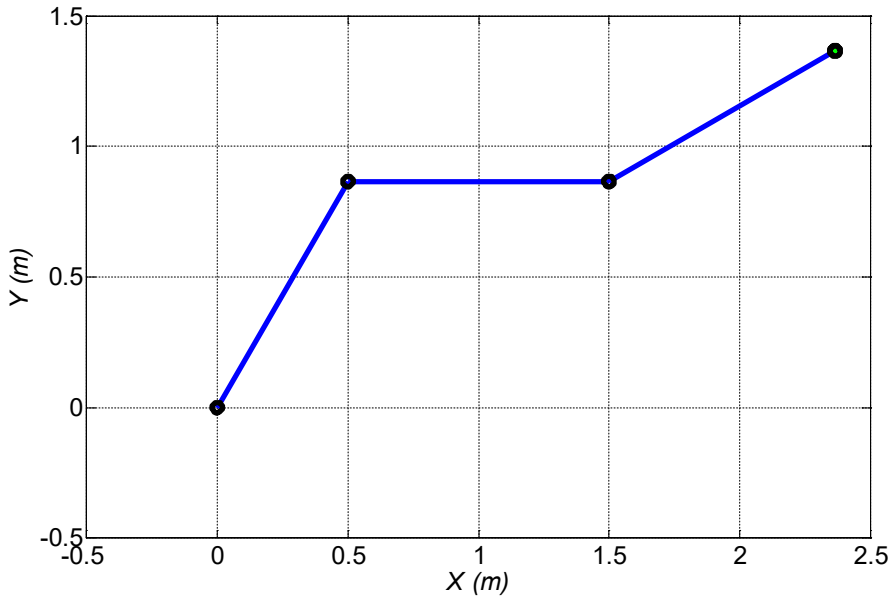
Total Solution

$$\left\{\dot{\Theta}\right\} = \left\{\dot{\Theta}_P\right\} + \left\{\dot{\Theta}_H\right\} = k_P \left[J\right]^* \left\{\dot{X}\right\} + k_H \left[\left[I_n\right] - \left[J\right]^* \left[J\right]\right] \left\{z\right\}$$

$$\left\{\dot{\Theta}\right\} = \begin{Bmatrix} -1.630 \\ 2.732 \\ -0.280 \end{Bmatrix}$$

check:

$$\left[J\right]\left\{\dot{\Theta}\right\} = \begin{bmatrix} -1.366 & -0.500 & -0.500 \\ 2.366 & 1.866 & 0.866 \end{bmatrix} \begin{Bmatrix} -1.630 \\ 2.732 \\ -0.280 \end{Bmatrix} = \begin{Bmatrix} 1 \\ 1 \end{Bmatrix}$$

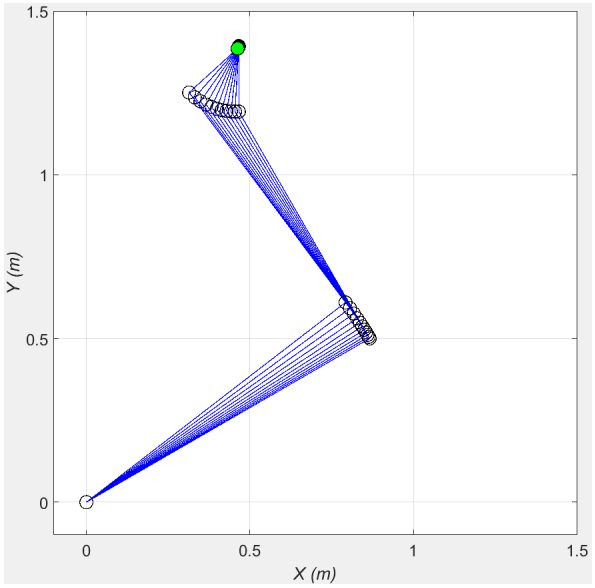


Planar 3R robot snapshot

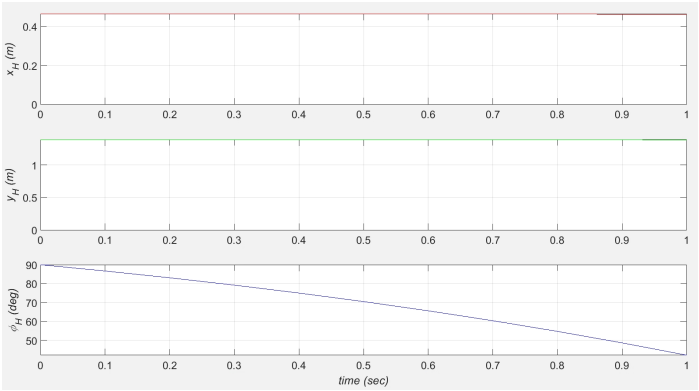
`Plan3RKRRrr.m` $n - m = 3 - 2 = 1$ translational rates only $\left\{\dot{x}_H \quad \dot{y}_H\right\}^T$

Now let us demonstrate self-motion for the Example 1 robot, by using the Homogeneous Solution only, in a null-space trajectory (that ensures $\{\dot{X}\} = \{0\}$). This new example is for the same arrangement, i.e. Planar 3R robot with $n - m = 3 - 2 = 1$ degree of kinematic redundancy, and no control of Cartesian angle ϕ. Also, there are different robots lengths and initial simulation joint angles:

$L_1 = 1.0, L_2 = 0.8, L_3 = 0.2$ m $\qquad$ $\theta_{10} = 30^\circ, \theta_{20} = 90^\circ, \theta_{30} = -30^\circ$ $\qquad$ `t = [0:0.05:1]` sec



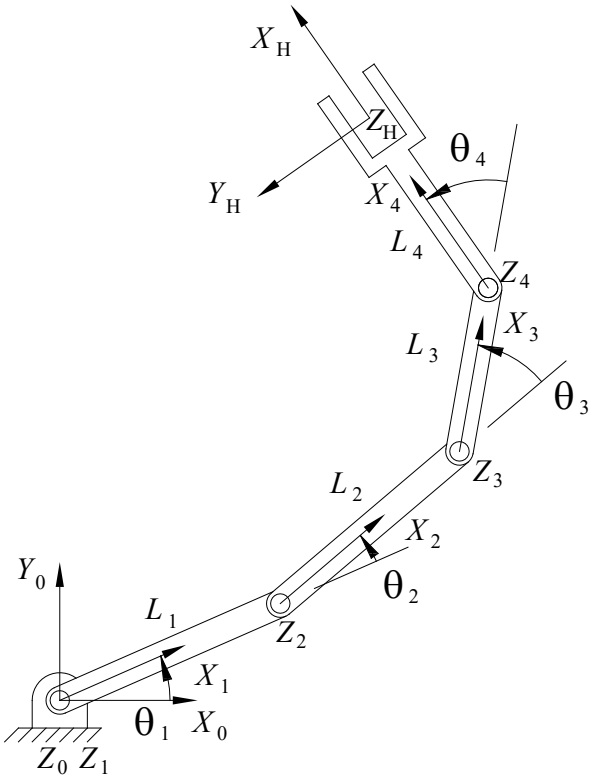
As seen in the Spirograph figure above, which was run for this new example robot with $k_H = 1$, the nature of the self-motion for this robot is like a four-bar mechanism. This is because the Cartesian rotation ϕ is not controlled and the end-effector point acts as a fixed R joint point (though it is not in reality, but achieving $\{\dot{X}\} = \{0\}$ at each simulation step). As seen in the resulting Cartesian variables plot below, found using FPK at each stage in the resolved-rate simulation, x_H = 0.466 and y_H = 1.393 are constant, and ϕ varies because we do not control it (ϕ is driving the 'four-bar mechanism'). Actually, x_H and y_H are not quite constant, but vary by 3 and 6 mm, respectively. The reason for this is that the resolved-rate method is not mathematically-exact. Since it relies on time derivatives, the smaller the Δt, the more accurate the results will be (smaller than the 0.05 sec chosen here).



Example 2. Planar Kinematically-Redundant 4-dof 4R Robot with *m* = 2

Now, let us repeat this example so that the elbow joint is involved in the homogeneous solution – the simplest way to do this is to use a robot with two elbow joints, so the distance from the shoulder to the wrist is a function of both elbow joint angles, rather than a single elbow joint angle.

To accomplish this for a **snapshot example**, a four-link 4R 4-dof planar robot is required to satisfy only 2-dof translations (no 1-dof rotations). The underconstrained Jacobian matrix for this case is 2x4 (giving the translational velocity of the hand frame {*H*} origin with respect to the base frame {0} origin, expressed in {0} coordinates; *m*=2 and *n*=4, so the degree of kinematic redundancy is 4 – 2 = 2). A four-link 4R 4-dof planar robot required to satisfy 2-dof translations and a 1-dof rotation (see Example 3 below) will not work since that would take us back to the one-elbow case.



DH Parameters Table

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	θ_1
2	0	L_1	0	θ_2
3	0	L_2	0	θ_3
4	0	L_3	0	θ_4

Forward Kinematics Expressions

$$\left[{}^{0}_{H}T\right]=\begin{bmatrix} c_{1234} & -s_{1234} & 0 & L_1c_1+L_2c_{12}+L_3c_{123}+L_4c_{1234} \\ s_{1234} & c_{1234} & 0 & L_1s_1+L_2s_{12}+L_3s_{123}+L_3s_{1234} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}=\begin{bmatrix} c\phi & -s\phi & 0 & x_H \\ s\phi & c\phi & 0 & y_H \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The three independent kinematics equations are:

$$\begin{aligned} {}^{0}x_H &= L_1c_1+L_2c_{12}+L_3c_{123}+L_4c_{1234} \\ {}^{0}y_H &= L_1s_1+L_2s_{12}+L_3s_{123}+L_3s_{1234} \\ \phi &= \theta_1+\theta_2+\theta_3+\theta_4 \end{aligned}$$

This is a four-link, 4-dof, 4R planar kinematically-redundant robot for general planar Cartesian trajectories x_H, y_H, ϕ. The **Jacobian matrix** is found by taking the first time derivative of the translational equations and using the relative angular velocity equation. The underconstrained Jacobian matrix for this case is of order 3x4 (giving translational and rotational velocity of the hand frame {H} origin with respect to the base frame {0} origin, expressed in {0} coordinates):

$$\left[{}^{0}J\right]=\begin{bmatrix} -L_1s_1-L_2s_{12}-L_3s_{123}-L_4s_{1234} & -L_2s_{12}-L_3s_{123}-L_4s_{1234} & -L_3s_{123}-L_4s_{1234} & -L_4s_{1234} \\ L_1c_1+L_2c_{12}+L_3c_{123}+L_4c_{1234} & L_2c_{12}+L_3c_{123}+L_4c_{1234} & L_3c_{123}+L_4c_{1234} & L_4c_{1234} \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

- The degree of kinematic redundancy is $n - m = 4 - 3 = 1$ for translational and rotational rates (see Example 3).

- The degree of kinematic redundancy is $n - m = 4 - 2 = 2$ for translational rates only – in this case cut the last row of the Jacobian matrix to make a doubly-unconstrained 2x4 Jacobian matrix (as in this problem, Example 2).

The required 2x4 Jacobian matrix for Example 2 is:

$$\left[{}^{0}J\right]=\begin{bmatrix} -L_1s_1-L_2s_{12}-L_3s_{123}-L_4s_{1234} & -L_2s_{12}-L_3s_{123}-L_4s_{1234} & -L_3s_{123}-L_4s_{1234} & -L_4s_{1234} \\ L_1c_1+L_2c_{12}+L_3c_{123}+L_4c_{1234} & L_2c_{12}+L_3c_{123}+L_4c_{1234} & L_3c_{123}+L_4c_{1234} & L_4c_{1234} \end{bmatrix}$$

Numerical example: let $L_1=L_2=L_3=L_4=1$ and $\theta_1=60^\circ,\theta_2=-60^\circ,\theta_3=30^\circ,\theta_4=30^\circ$. The required translational trajectory (instantaneous Cartesian rates for this snapshot example) is $\{\dot{X}\}=\{1 \quad 1\}^T$. The numerical Jacobian matrix at this snapshot is:

$$\left[{}^{0}J\right]=\begin{bmatrix} -2.232 & -1.366 & -1.366 & -0.866 \\ 2.866 & 2.366 & 1.366 & 0.500 \end{bmatrix}$$

Particular Solution

$$\left\{\dot{\Theta}_P\right\} = k_P \left[J\right]^* \left\{\dot{X}\right\} = k_P \left[J\right]^T \left[\left[J\right]\left[J\right]^T\right]^{-1} \left\{\dot{X}\right\}$$

$$\left[J\right]^* = \begin{bmatrix} -0.161 & 0.059 \\ 0.764 & 0.721 \\ -0.646 & -0.398 \\ -0.925 & -0.661 \end{bmatrix}$$

$$\left\{\dot{\Theta}_P\right\} = (1)\begin{bmatrix} -0.161 & 0.059 \\ 0.764 & 0.721 \\ -0.646 & -0.398 \\ -0.925 & -0.661 \end{bmatrix}\begin{Bmatrix} 1 \\ 1 \end{Bmatrix} = \begin{Bmatrix} -0.102 \\ 1.484 \\ -1.043 \\ -1.587 \end{Bmatrix}$$

This $\left\{\dot{\Theta}_P\right\}$ is the least-norm solution for joint rates.

check: $\left[J\right]\left\{\dot{\Theta}_P\right\} = \begin{Bmatrix} 1 \\ 1 \end{Bmatrix}$ as required.

Homogeneous Solution

$$\left\{\dot{\Theta}_H\right\} = k_H \left[\left[I_n\right] - \left[J\right]^* \left[J\right]\right]\left\{z\right\}$$

The symmetric null-space-projection matrix is:

$$\left[\left[I_4\right] - \left[J\right]^* \left[J\right]\right] = \begin{bmatrix} 0.471 & -0.360 & -0.301 & -0.169 \\ -0.360 & 0.339 & 0.059 & 0.301 \\ -0.301 & 0.059 & 0.661 & -0.360 \\ -0.169 & 0.301 & -0.360 & 0.530 \end{bmatrix}.$$

This matrix projects an arbitrary vector *z* into the null-space of the Jacobian matrix, meaning the resulting homogeneous solution $\left\{\dot{\Theta}_H\right\}$ maps through [*J*] to the zero vector (i.e. causing zero end-effector Cartesian motion). Using an arbitrary {*z*} (not attempting to optimize anything), with $k_H = 0.5$:

$$\left\{\dot{\Theta}_H\right\} = (0.5)\begin{bmatrix} 0.471 & -0.360 & -0.301 & -0.169 \\ -0.360 & 0.339 & 0.059 & 0.301 \\ -0.301 & 0.059 & 0.661 & -0.360 \\ -0.169 & 0.301 & -0.360 & 0.530 \end{bmatrix}\begin{Bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{Bmatrix} = \begin{Bmatrix} -0.180 \\ 0.169 \\ 0.030 \\ 0.151 \end{Bmatrix}$$

check: $\left[J\right]\left\{\dot{\Theta}_H\right\} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}$ as required.

Note this null-space projection matrix proves that both elbow angles are now involved in the homogeneous solution (since there are no rows nor columns of all zeros).

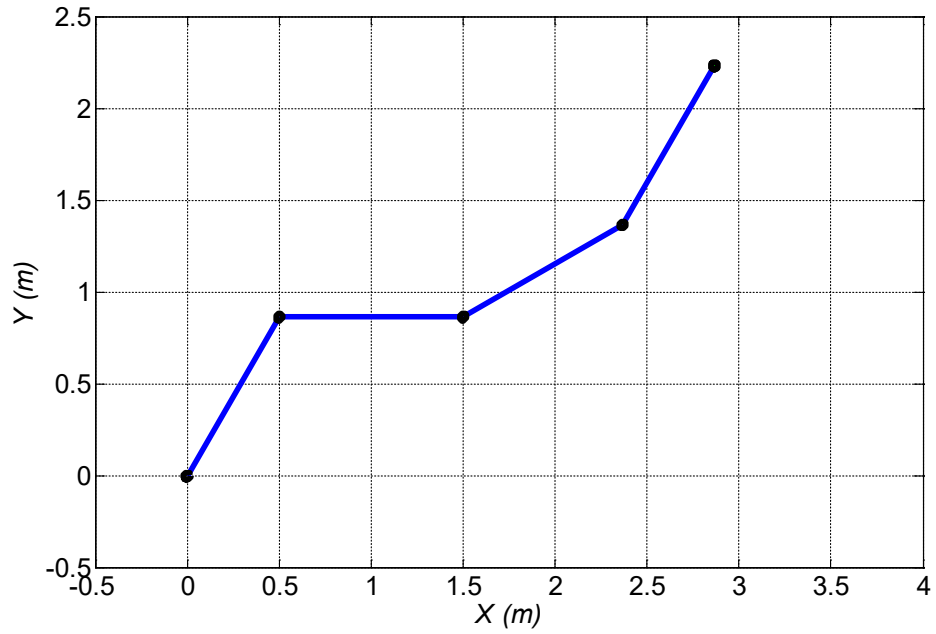
Total Solution

$$\left\{\dot{\Theta}\right\} = \left\{\dot{\Theta}_P\right\} + \left\{\dot{\Theta}_H\right\} = k_P \left[J\right]^* \left\{\dot{X}\right\} + k_H \left[\left[I_n\right] - \left[J\right]^* \left[J\right]\right] \left\{z\right\}$$

$$\left\{\dot{\Theta}\right\} = \begin{Bmatrix} -0.282 \\ 1.654 \\ -1.014 \\ -1.436 \end{Bmatrix}$$

check:

$$\left[J\right]\left\{\dot{\Theta}\right\} = \begin{bmatrix} -2.232 & -1.366 & -1.366 & -0.866 \\ 2.866 & 2.366 & 1.366 & 0.500 \end{bmatrix} \begin{Bmatrix} -0.282 \\ 1.654 \\ -1.014 \\ -1.436 \end{Bmatrix} = \begin{Bmatrix} 1 \\ 1 \end{Bmatrix}$$



Planar 4R robot snapshot

`Plan4Rrr2.m` $n - m = 4 - 2 = 2$ translational rates only $\left\{\dot{x}_H \quad \dot{y}_H\right\}^T$

If we were to run this case for a homogeneous null-space resolved-rate simulation trajectory, it would have a 2-dof 5-bar robot-type of self-motion.

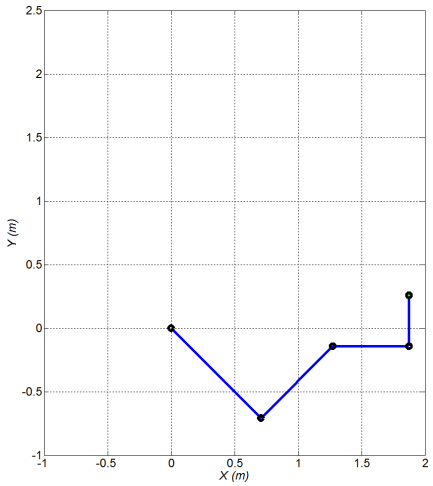
Example 3. Planar Kinematically-Redundant 4-dof 4R Robot Example with *m* = 3

Now we control the rotational motion also, so the degree of kinematic redundancy is $n-m=\ 4\ -\ 3\ =\ 1$ for translational and rotational rates. The required 3x4 Jacobian matrix for Example 3 was derived in Example 2 above:

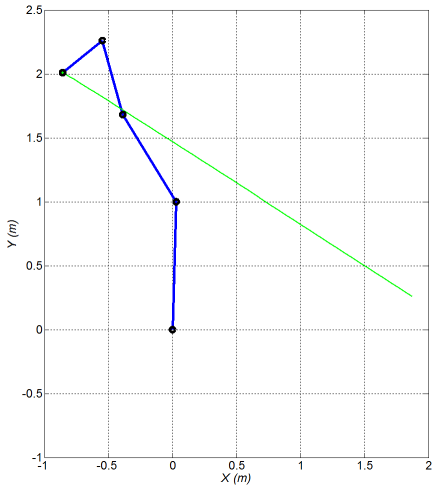
$$\left[{}^{0}J\right]=\begin{bmatrix} -L_1s_1-L_2s_{12}-L_3s_{123}-L_4s_{1234} & -L_2s_{12}-L_3s_{123}-L_4s_{1234} & -L_3s_{123}-L_4s_{1234} & -L_4s_{1234} \\ L_1c_1+L_2c_{12}+L_3c_{123}+L_4c_{1234} & L_2c_{12}+L_3c_{123}+L_4c_{1234} & L_3c_{123}+L_4c_{1234} & L_4c_{1234} \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Planar 4R Resolved-Rate Control Trajectory Simulation Numerical Example

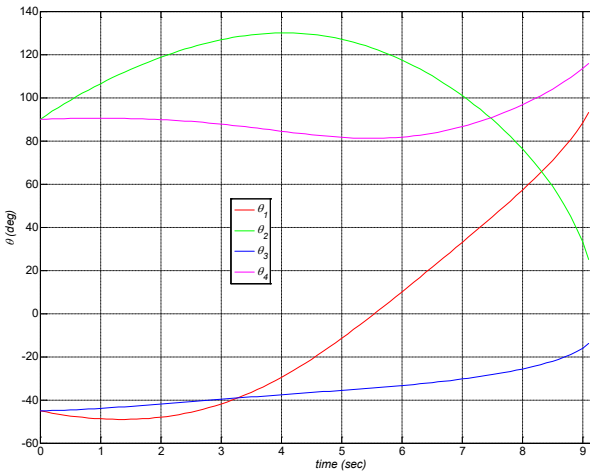
Simulate resolved-rate control for the planar 4R robot, the particular solution only ($k_P=1$ and $k_H=0$), given the robot lengths L_1 = 1, L_2 = 0.8, L_3 = 0.6 , L_4 = 0.4 (m), initial joint angles $\{\Theta\}=\{\theta_1\ \ \theta_2\ \ \theta_3\ \ \theta_4\}^T=\{-45^\circ\ \ 90^\circ\ \ -45^\circ\ \ 90^\circ\}^T$, and the constant commanded Cartesian rates ${}^{0}\{\dot{X}\}=\{\dot{x}_H\ \ \dot{y}_H\ \ \omega_z\}^T=\{-0.3\ \ 0.2\ \ 0.25\}^T$ (m/s and rad/s). Simulate for 9 *sec*, using time steps of *dt* = 0.1 *sec*. Also simulate the inverse statics solution, i.e. calculate the joint torques $\{\mathrm{T}\}=\{\tau_1\ \ \tau_2\ \ \tau_3\ \ \tau_4\}^T$ (Nm) given the constant Cartesian wrench ${}^{0}\{W\}=\{f_x\ \ f_y\ \ m_z\}^T=\{3\ \ 2\ \ 1\}^T$ (N and Nm).



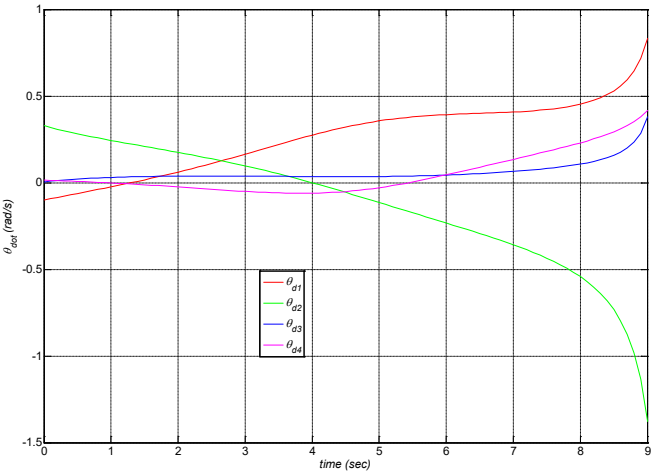
Initial Robot Pose



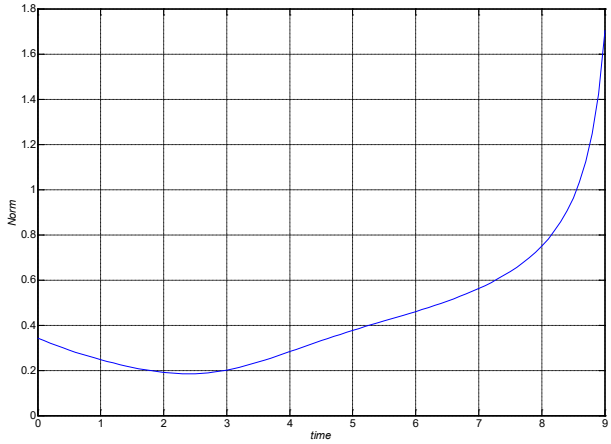
Final Robot Pose



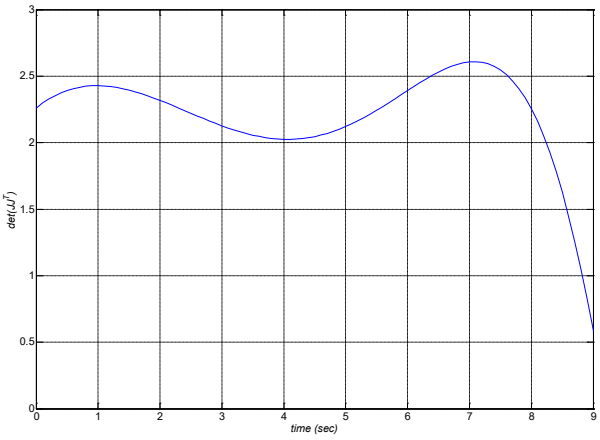
Robot Joint Angles



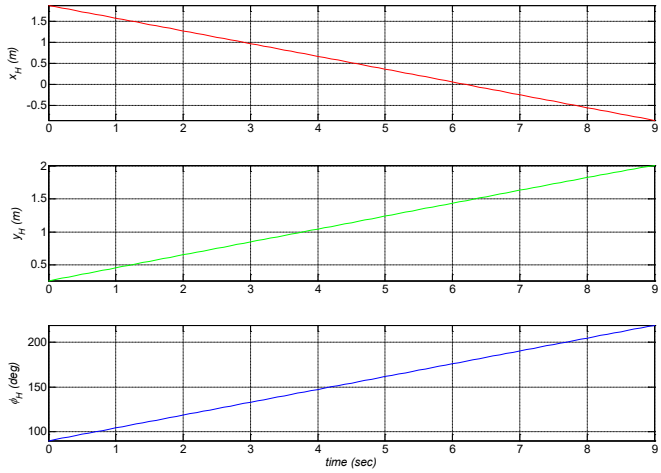
Robot Joint Rates



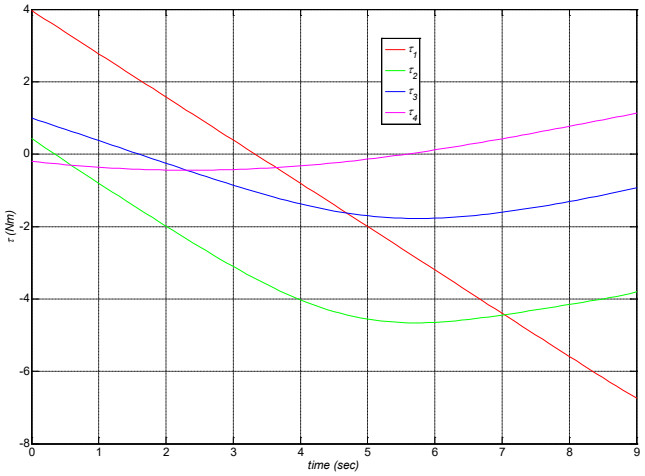
Norm of Joint Rates



Determinant of [*J*][*J*]^T



Robot Cartesian Pose

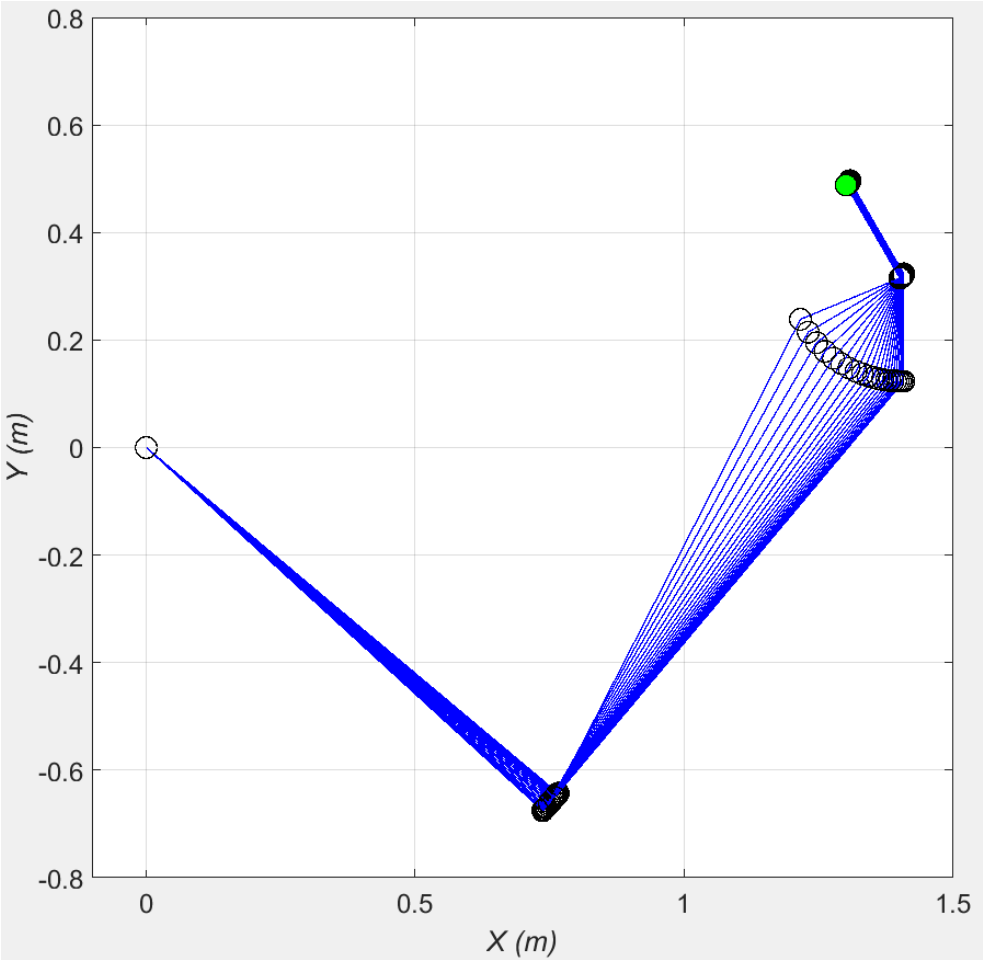


Robot Joint Torques

`Plan4Rrr.m` $n - m = 4 - 3 = 1$ translational and rotational rates $\{\dot{x}_H \quad \dot{y}_H \quad \omega_z\}^T$

Now let us demonstrate self-motion for the Example 3 robot, by using the Homogeneous Solution only, in a null-space trajectory (that ensures $\{\dot{X}\} = \{0\}$). This new example is for the same arrangement, i.e. Planar 3R robot with $n - m = 4 - 3 = 1$ degree of kinematic redundancy, with control of Cartesian angle ϕ to stay constant. Also, there are different robots lengths and initial simulation joint angles:

$L_1 = 1.0, L_2 = 1.0, L_3 = 0.2, L_4 = 0.2$ m $\quad \theta_{10} = -40^\circ, \theta_{20} = 90^\circ, \theta_{30} = 40^\circ, \theta_{40} = 30^\circ$ `t = [0:0.05:1]` sec



As seen in the Spirograph figure above, which was run for this new example robot with $k_H = 3$, the nature of the self-motion for this robot is again like a four-bar mechanism (as in Example 1). This is because, though now the Cartesian rotation ϕ is controlled, there is a fourth link, so the entire the end-effector link pose is constant, and the frame {4} original point acts as a fixed R joint point (though it is not in reality, but achieving $\{\dot{X}\} = \{0\}$ at each simulation step). Again, the resulting Cartesian variables plot are essentially constant (not shown), found using FPK at each stage in the resolved-rate simulation. x_H and y_H vary by 6 mm each from their initial positions. The reason for this is again that the resolved-rate method is not mathematically-exact. Since it relies on time derivatives, the smaller the Δt, the more accurate the results will be (smaller than the 0.05 sec chosen here). However, the FPK Cartesian angle results are rock-steady at $\phi = 120^\circ$, with no error even out to the 10th decimal place.

9. Parallel Robots

9.1 Introduction

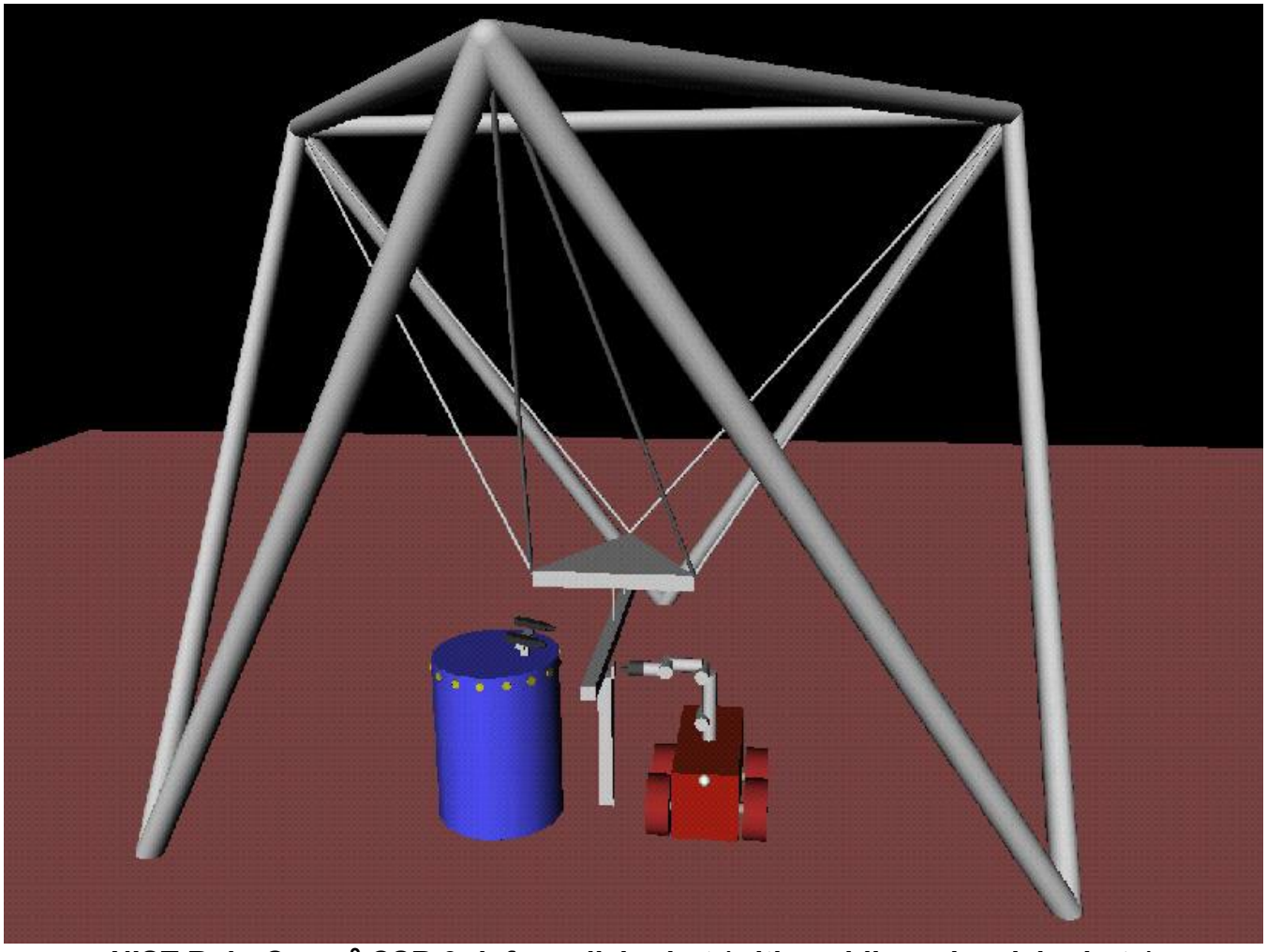
Serial vs. Parallel Robot – demonstrate with wooden model kit. Example parallel manipulators include the Stewart Platform, the NIST RoboCrane, the Delta Robot, and the 2-dof 5-bar robot (NASA Hyper-redundant Serpentine Arm and backhoes are hybrid serial/parallel).

 **KUKA LWR 7R 7-dof serial robot** autsys.aalto.fi	 **Stewart Platform 6-UPS 6-dof parallel robot** mrmoco.com
Serial Robot (articulated arm). The joints and links extend from the base in a serial fashion to the end-effector. The overall structure is cantilevered which is not good for resisting end-effector loads. All joints must be active.	**Parallel Robot (Stewart Platform).** The joints and links are arranged in parallel from the base to the end-effector. There are multiple load paths from the end-effector to the base. There is a combination of active and passive joints.

Comparison of Parallel vs. Serial Robots

Advantages of parallel robots	**Disadvantages of parallel robots**
• Higher loads • Higher accelerations • Lower mass • Higher stiffness • Better accuracy and repeatability • Ground-mounted actuators • Direct-drive • Open structure for cabling	• SMALLER WORKSPACE • Generally kinematics and dynamics analyses are more difficult

Cable-Suspended Robots (CSRs) have garnered an immense amount of research interest internationally in the past two decades. CSRs are a type of parallel robot wherein the entire motive structure of the robot consists of lightweight, stiff, strong cables moving an end-effector platform.



NIST RoboCrane[9] CSR 6-dof parallel robot (with mobile and serial robots)

frc.ri.cmu.edu

Generally CSRs share the same Advantages and Disadvantages of Parallel vs. Serial Robots, as presented on the previous page, with many benefits even stronger for CSRs. However, the Translational Workspaces of CSRs can be HUGE; up to 1 km^2 footprint CSRs have been successfully designed, built, and controlled.

What is the primary disadvantage of Cable-Suspended Robots, compared to other Parallel Robots and Serial Robots?

[9] J. Albus, R. Bostelman, and N. Dagalakis, 1993, "The NIST RoboCrane", Journal of National Institute of Standards and Technology, 10(5): 709-724.

A more comprehensive qualitative comparison of serial vs. parallel robots was done by Pandilov and Dukovski[10], whose main results are presented below. Dr. Bob added the comparison column for Cable-Suspended Robots (CSRs) and the entire last row.

Pandilov and Dukovski[9] Comparison of Serial vs. Parallel Robots and Dr. Bob Comparison of CSRs

Characteristic	Serial Robot[9]	Parallel Robot[9]	Cable-Suspended Robot (Dr. Bob)
Workspace	Large	Small and Complex	HUGE!
Solving FPK	Easy	Very difficult	Difficult
Solving IPK	Difficult	Easy	Very Easy
Position Error	Accumulates	Averages	Averages
Force Error	Averages	Accumulates	Accumulates
Maximum Force	Min actuator force	Sum all actuator forces	Sum all actuator forces
Stiffness	Low	High	Excellent
Dynamics	Poor, especially with increasing size	Very good	Excellent
Modeling and Solving Dynamics	Relatively simple	Very complex	Very simple
Inertia	Large	Small	Very Small
Current Application Areas	Very wide	Limited	Very limited
Payload-to-Weight Ratio	Low	High	Excellent
Speed and Acceleration	Low	High	Excellent
Accuracy	Low	High	Excellent
Uniformity of Components	Low	High	Excellent
Calibration	Relatively simple	Complicated	Complicated
Workspace-to-Robot Size Ratio	High	Low	Excellent
Dual-Directional Actuators (Dr. Bob)	Yes	Yes	No

[10] Z. Pandilov and V. Dukovski, 2014, "Comparison of the Characteristics between Serial and Parallel Robots", *Acta Tehnica Corviniensis – Bulletin of Engineering,* VII: 144-160.

Mobility – Kutzbach's planar and spatial mobility equations (see Section 1.3) for calculating overall manipulator dof are very important for parallel manipulators to ensure the correct number and type of active and passive joints and links are used to obtained the desired dof. There are infinite variations and possible designs for parallel robots.

Kinematics problems for parallel robots

- forward and inverse pose solutions
- workspace
- forward and inverse velocity solutions

Both inverse pose and rate problems have been presented for all possible planar parallel robots with 3 identical 3-dof legs[11].

There exists an interesting duality between serial and parallel robots: for parallel robots, the inverse pose and velocity solutions are generally straight-forward and the forward pose and velocity problems are harder. We know the opposite case is true for serial robots in general.

We could also do parallel robot **dynamics** but this is of less concern for a parallel robot than for a serial robot due to the parallel nature and its inherent advantages.

Serial robot kinematics – start with DH parameters, homogeneous transformation matrices

Parallel robot kinematics – start with vector loop-closure equations

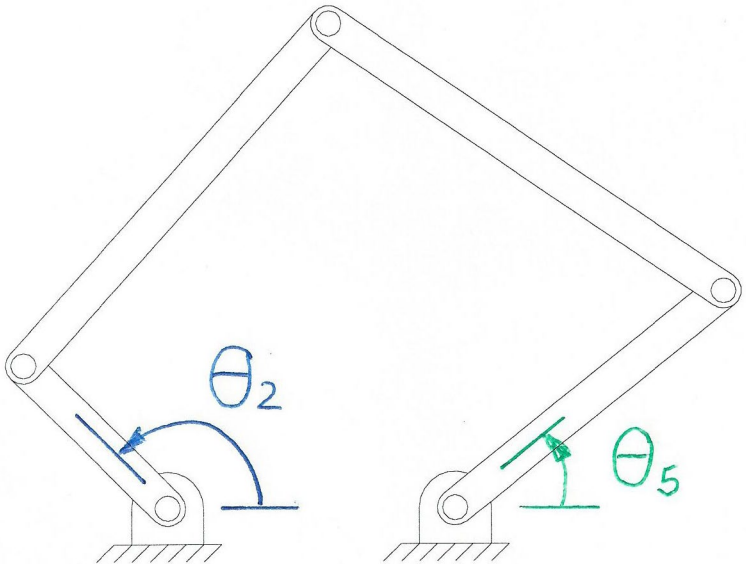
Airbus 320 Full Flight Simulator
(Stewart Platform parallel robot)
6-UPS
en.wikipedia.org

Adept Quattro Parallel Robot
(Delta parallel robot)
4-RUU
adept.com/products/robots/parallel

[11] R.L. Williams II and B.H. Shelley, Inverse Kinematics for Planar Parallel Manipulators, CD Proceedings of the 1997 ASME Design Technical Conferences, 23rd Design Automation Conference, DETC97/DAC-3851, Sacramento, CA, September 14-17, 1997.

9.2 Planar 2-dof Five-Bar Parallel Robot

The Planar 2-dof Five-Bar Parallel Robot is similar to the Planar 1-dof Four-Bar Mechanism. There is an extra moving link and another revolute joint, providing 2-dof instead of 1-dof. Therefore, there must be two motors (ground-mounted), providing input angles θ_2 and θ_5 independently as shown in the diagram below.



Planar 2-dof Five-Bar RRRRR Parallel Robot

The **RRRRR** designation means that there are five revolute joints in this parallel robot, and the first and last (the two grounded revolute joints) are actuated via independent motors. The three intermediate moving revolute joints are passive.

The planar five-bar has five total rigid links (four moving and one ground link) and five revolute joints, each 1-dof joints (J_1). Therefore the total number of degrees-of-freedom (dof), or Mobility M, is:

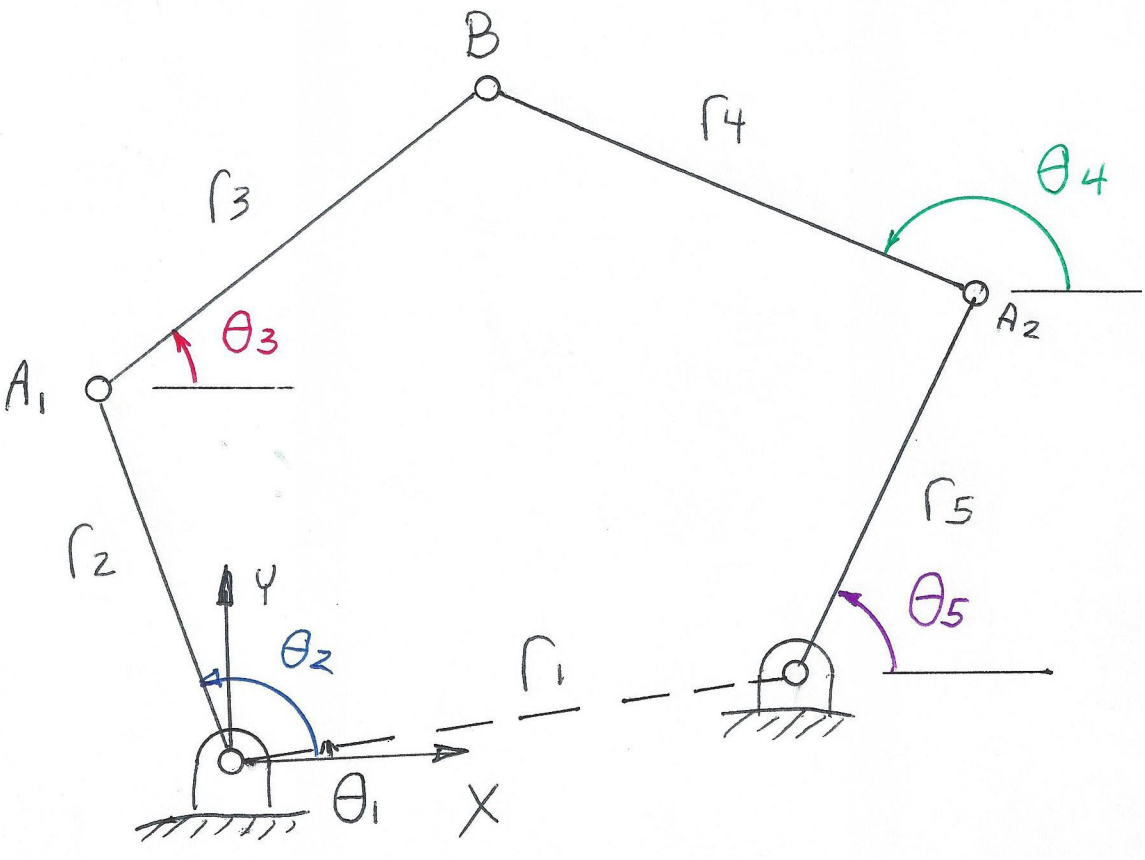
$$M = 3(N-1) - 2J_1 - 1J_2$$
$$M = 3(5-1) - 2(5) - 1(0)$$
$$M = 2 \quad \text{dof}$$

Since $M > 1$, the planar five-bar qualifies as a robot, and it is a parallel robot since there are two paths from the ground link to the end-effector.

The end-effector could be located on link 3 or link 4 (see kinematic diagram below). For generality, we will place the end-effector point to coincide with Point B, the point where links 3 and 4 intersect with a revolute joint.

9.2.1 Forward Position Kinematics

Step 1. Draw the **Kinematic Diagram**



r_1	constant ground link length	θ_1	constant ground link angle
r_2	constant input link length	θ_2	variable input angle
r_3	constant passive link length	θ_3	variable passive angle
r_4	constant passive link length	θ_4	variable passive angle
r_5	constant second input link length	θ_5	variable second input angle

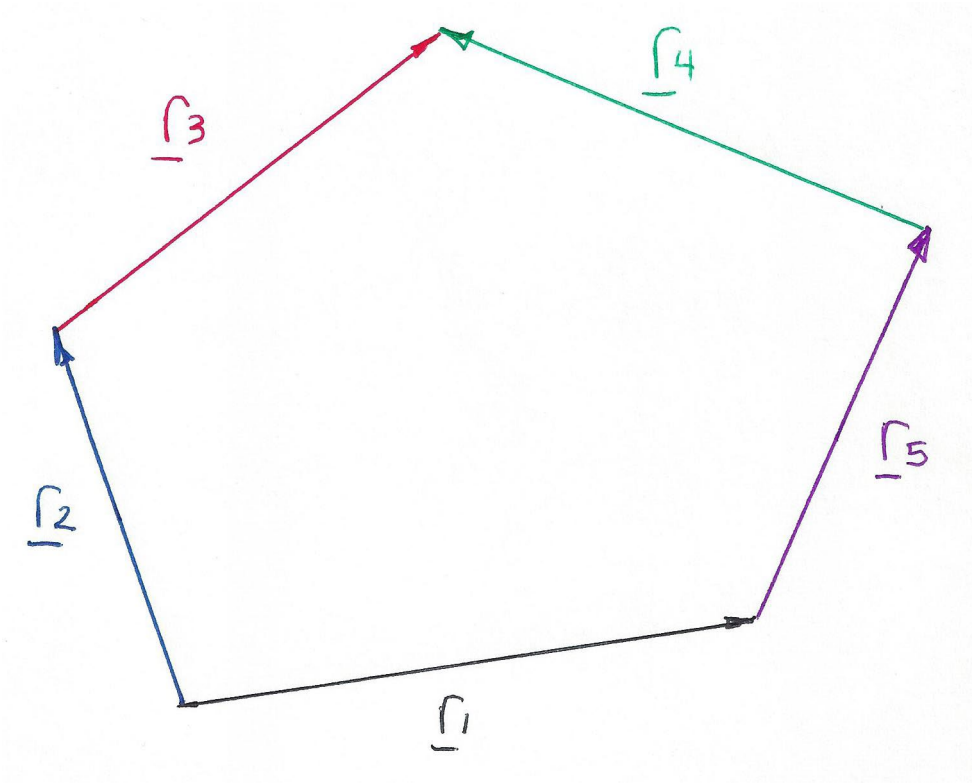
Link 1 is the fixed ground link. All angles θ_1 through θ_5 are absolute, measured in a right-hand sense from the horizontal X direction to each link.

Step 2. **State the Problem**

Given: r_1, θ_1, r_2, r_3, r_4, r_5; plus 2 independent input angles θ_2 and θ_5

Find: end-effector point $\underline{B} = \{B_x \quad B_y\}^T$ plus θ_3 and θ_4

Step 3. Draw the **Vector Diagram**. Define all angles in a positive sense, measured with the right hand from the right horizontal to the link vector (tail-to-head; your right-hand thumb is located at the vector tail and your right-hand fingers point in the direction of the vector arrow).



Step 4. Derive the **Vector-Loop-Closure Equation**. Starting at one point, add vectors tail-to-head until you reach a second point. Write the VLCE by starting and ending at the same points, but choosing a different path.

$$\underline{r}_2 + \underline{r}_3 = \underline{r}_1 + \underline{r}_5 + \underline{r}_4$$

Step 5. Write the ***XY* Components** for the Vector-Loop-Closure Equation. Separate the one vector equation into its two *X* and *Y* scalar components.

$$r_2 c_2 + r_3 c_3 = r_1 c_1 + r_5 c_5 + r_4 c_4$$
$$r_2 s_2 + r_3 s_3 = r_1 s_1 + r_5 s_5 + r_4 s_4$$

where the following abbreviations have been used:

$$c_i = \cos\theta_i \qquad i = 1,2,3,4,5$$
$$s_i = \sin\theta_i$$

Step 6. Solve for the Unknowns from the *XY* equations. Step 5 yielded two coupled nonlinear transcendental equations in the two unknowns θ_3, θ_4. Though difficult, we can solve this problem analytically. It is very similar to the planar 1-dof four-bar mechanism position analysis solution, since it involves the intersection of two known circles.

Rewrite the above *XY* equations to isolate the θ_3 unknown on one side of both equations.

$$r_3 c_3 = r_1 c_1 + r_5 c_5 + r_4 c_4 - r_2 c_2 = a + r_4 c_4$$
$$r_3 s_3 = r_1 s_1 + r_5 s_5 + r_4 s_4 - r_2 s_2 = b + r_4 s_4$$

where:

$$a = r_1 c_1 + r_5 c_5 - r_2 c_2$$
$$b = r_1 s_1 + r_5 s_5 - r_2 s_2$$

Then square and add the two *XY* position kinematics equations and eliminate θ_3 using the trigonometric identity $\cos^2\theta_3 + \sin^2\theta_3 = 1$. This results in a single nonlinear transcendental equation in the single unknown θ_4:

$$E\cos\theta_4 + F\sin\theta_4 + G = 0$$

where:

$$E = 2ar_4$$
$$F = 2br_4$$
$$G = a^2 + b^2 - r_3^2 + r_4^2$$

Or, unsubstituting terms *a* and *b* and simplifying:

$$E = 2r_4(r_1 c_1 + r_5 c_5 - r_2 c_2)$$
$$F = 2r_4(r_1 s_1 + r_5 s_5 - r_2 s_2)$$
$$G = r_1^2 + r_2^2 - r_3^2 + r_4^2 + r_5^2 + 2r_1 r_5\cos(\theta_5 - \theta_1) - 2r_1 r_2\cos(\theta_2 - \theta_1) - 2r_2 r_5\cos(\theta_5 - \theta_2)$$

We can solve this equation for θ_4 by using the **Tangent Half-Angle Substitution** (used earlier in EE/ME 4290/5290 to help solve the Planar 3R IPK).

If we define $t = \tan\frac{\theta_4}{2}$ then $\cos\theta_4 = \frac{1-t^2}{1+t^2}$ and $\sin\theta_4 = \frac{2t}{1+t^2}$

Substitute the **Tangent Half-Angle Substitution** into the *EFG* equation:

$$E\left(\frac{1-t^2}{1+t^2}\right) + F\left(\frac{2t}{1+t^2}\right) + G = 0$$

$$E(1-t^2)+F(2t)+G(1+t^2)=0$$

$$(G-E)t^2+(2F)t+(G+E)=0$$

This equation is easily solved using the quadratic formula:

$$t_{1,2}=\frac{-F\pm\sqrt{E^2+F^2-G^2}}{G-E}$$

Solve for θ_4 by inverting the original Tangent Half-Angle Substitution definition:

$$\theta_{4_{1,2}}=2\tan^{-1}(t_{1,2})$$

The **atan2** function need not be used in the above equation for θ_4, due to multiplying by 2.

Two θ_4 solutions result, from the $\pm$ in the quadratic formula. Both are correct since there are two valid solution branches – end-effector up and end-effector down. To find θ_3, return to the translational position kinematics equations, with θ_3 terms isolated.

$$\theta_{3_{1,2}}=\text{atan2}(r_1s_1+r_5s_5+r_4s_{4_{1,2}}-r_2s_2\ ,\ r_1c_1+r_5c_5+r_4c_{4_{1,2}}-r_2c_2)$$

This equation calculates the unique $\theta_{3_{1,2}}$ for each $\theta_{4_{1,2}}$ value. In this case we must use the quadrant-specific **atan2** function in MATLAB.

Thus, there are two solution sets for the forward position kinematics problem for the planar **RRRRR** five-bar robot. Be sure to keep the solutions together, i.e. $(\theta_3,\theta_4)_1$ and $(\theta_3,\theta_4)_2$.

The ultimate forward position kinematics solution, point *B*, may now be found since the passive angles for links 3 and 4 are now known. There will be two possible point *B* branch values, again corresponding to end-effector up (using – from the quadratic formula) and end-effector down (using +). Both left/right alternatives below will yield the same results for both branches of the point *B* values, one using the left path and the other using the right path.

point *B*, two branches, from θ_2 and $\theta_{3_{1,2}}$	**point *B*, same two branches, from θ_5 and $\theta_{4_{1,2}}$**
$\underline{B}=\underline{r}_2+\underline{r}_3$	$\underline{B}=\underline{r}_1+\underline{r}_5+\underline{r}_4$
$\begin{Bmatrix}B_x\\B_y\end{Bmatrix}_{1,2}=\begin{Bmatrix}r_2c_2+r_3c_{3_{1,2}}\\r_2s_2+r_3s_{3_{1,2}}\end{Bmatrix}$	$\begin{Bmatrix}B_x\\B_y\end{Bmatrix}_{1,2}=\begin{Bmatrix}r_1c_1+r_5c_5+r_4c_{4_{1,2}}\\r_1s_1+r_5s_5+r_4s_{4_{1,2}}\end{Bmatrix}$

Five-Bar Parallel Robot Forward Position Kinematics Snapshot Example

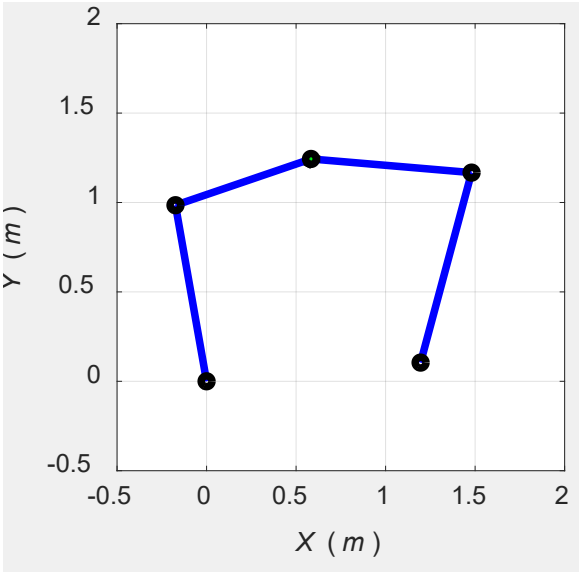
Constant robot dimensions: $r_1 = 1.2$, $\theta_1 = 5^\circ$, $r_2 = 1.0$, $r_3 = 0.8$, $r_4 = 0.9$, $r_5 = 1.1$ (m)

Given: input angles $\theta_2 = 100^\circ$ and $\theta_5 = 75^\circ$

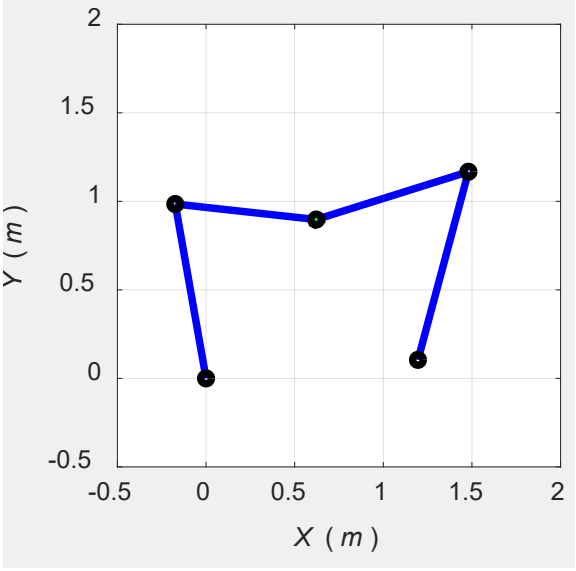
Find: end-effector point $\underline{B} = \{B_x \quad B_y\}^T$ plus θ_3 and θ_4

Solution:

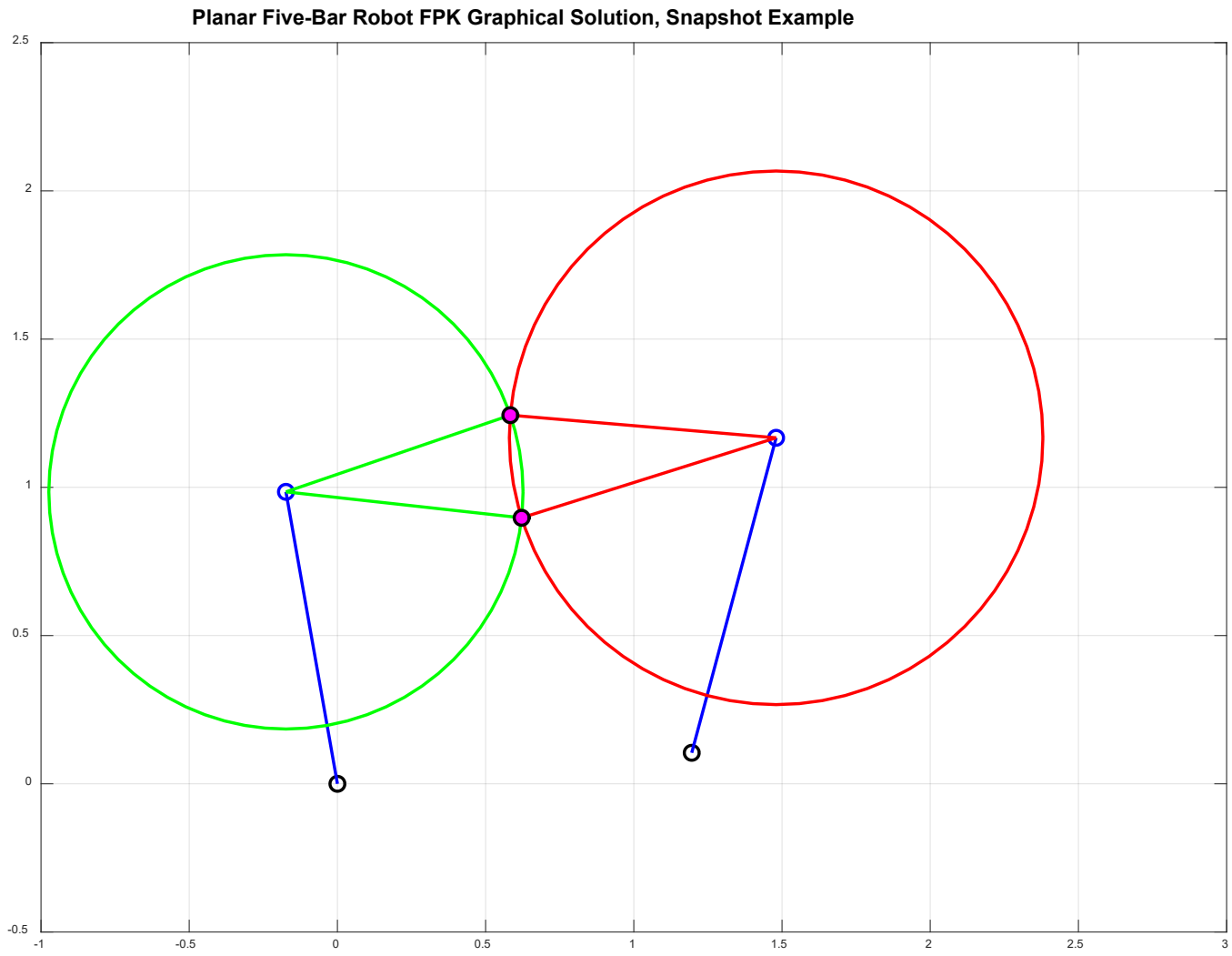
branch	*x* (m)	*y* (m)	θ_3 (deg)	θ_4 (deg)
end-effector up	0.5834	1.2435	18.9	175.1
end-effector down	0.6215	0.8972	–6.3	–162.6



End-Effector Up



End-Effector Down



FiveBarGraphical.m

9.2.2 Inverse Position Kinematics

Given: r_1, θ_1, r_2, r_3, r_4, r_5; plus desired end-effector point $\underline{B} = \{B_x \quad B_y\}^T$

Find: actuator angles θ_2 and θ_5, plus passive angles θ_3 and θ_4

The actuator unknowns θ_2 and θ_5 can be found independently from the two vector loop-closure equations presented at in the Forward Position Kinematics Section 9.2.1. Passive unknown angle θ_3 can be found along with θ_2, and passive unknown angle θ_4 can be found along with θ_5.

$$\underline{B} = \underline{r}_2 + \underline{r}_3 \qquad\qquad \underline{B} = \underline{r}_1 + \underline{r}_5 + \underline{r}_4$$

$$\begin{Bmatrix} B_x \\ B_y \end{Bmatrix} = \begin{Bmatrix} r_2c_2 + r_3c_3 \\ r_2s_2 + r_3s_3 \end{Bmatrix} \qquad \begin{Bmatrix} B_x \\ B_y \end{Bmatrix} = \begin{Bmatrix} r_1c_1 + r_5c_5 + r_4c_4 \\ r_1s_1 + r_5s_5 + r_4s_4 \end{Bmatrix}$$

Isolate the passive unknown angle in each of the above *XY* scalar equations:

$$r_3c_3 = B_x - r_2c_2 \qquad\qquad r_4c_4 = B_x - r_1c_1 - r_5c_5$$
$$r_3s_3 = B_y - r_2s_2 \qquad\qquad r_4s_4 = B_y - r_1s_1 - r_5s_5$$

Squaring and adding both sets separately will eliminate the passive unknowns for now:

$$E_2\cos\theta_2 + F_2\sin\theta_2 + G_2 = 0 \qquad\qquad E_5\cos\theta_5 + F_5\sin\theta_5 + G_5 = 0$$
$$E_2 = -2r_2B_x \qquad\qquad E_5 = 2r_5(-B_x + r_1c_1)$$
$$F_2 = -2r_2B_y \qquad\qquad F_5 = 2r_5(-B_y + r_1s_1)$$
$$G_2 = r_2^2 - r_3^2 + B_x^2 + B_y^2 \qquad\qquad G_5 = r_1^2 + r_5^2 - r_4^2 + B_x^2 + B_y^2 - 2r_1(B_xc_1 + B_ys_1)$$

Solving independently for the two actuator unknowns using the tangent half-angle substitution twice:

$$t_{2_{1,2}} = \frac{-F_2 \pm \sqrt{E_2^2 + F_2^2 - G_2^2}}{G_2 - E_2} \qquad\qquad t_{5_{1,2}} = \frac{-F_5 \pm \sqrt{E_5^2 + F_5^2 - G_5^2}}{G_5 - E_5}$$

$$\theta_{2_{1,2}} = 2\tan^{-1}(t_{2_{1,2}}) \qquad\qquad \theta_{5_{1,2}} = 2\tan^{-1}(t_{5_{1,2}})$$

Now return to the original *XY* scalar equations to solve for the passive unknowns:

$$\theta_{3_{1,2}} = \text{atan2}(B_y - r_2s_{2_{1,2}}\,,\, B_x - r_2c_{2_{1,2}}) \qquad\qquad \theta_{4_{1,2}} = \text{atan2}(B_y - r_1s_1 - r_5s_{5_{1,2}}\,,\, B_x - r_1c_1 - r_5c_{5_{1,2}})$$

There are two possible values (left and right branches) for unknowns θ_2 and θ_3; independently there are two possible values (left and right branches) for unknowns θ_5 and θ_4. Therefore, there are four overall valid solutions to the Inverse Position Kinematics problem for the planar parallel five-bar robot.

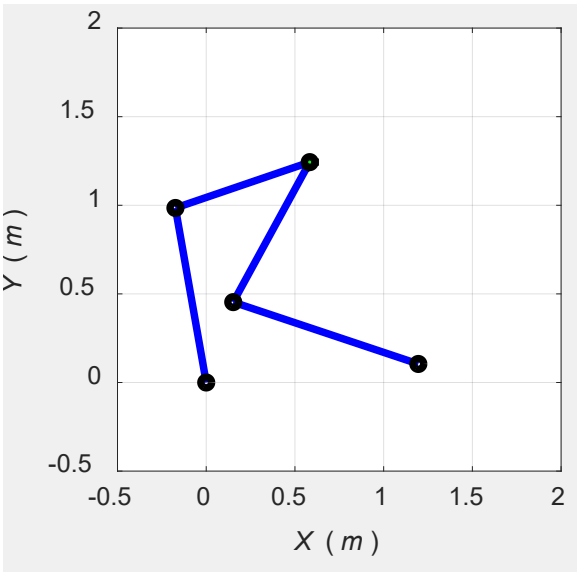
Five-Bar Parallel Robot Inverse Position Kinematics Snapshot Example (same robot)

Given: end-effector point $\underline{B} = \{B_x \quad B_y\}^T = \{0.5834 \quad 1.2435\}^T$

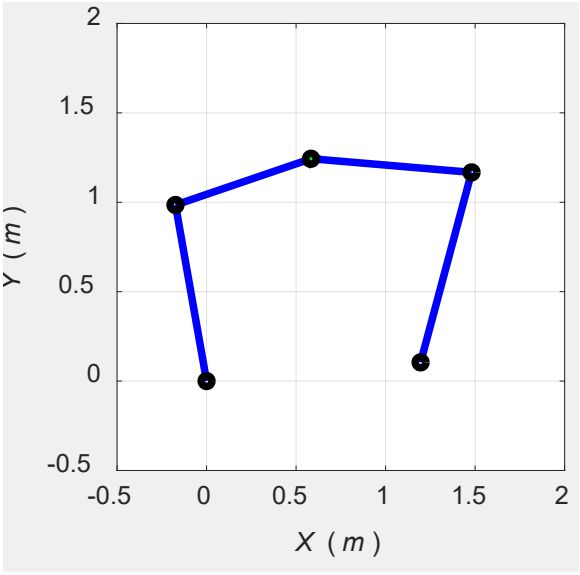
Find: actuator angles θ_2 and θ_5, plus passive angles plus θ_3 and θ_4

Solution:

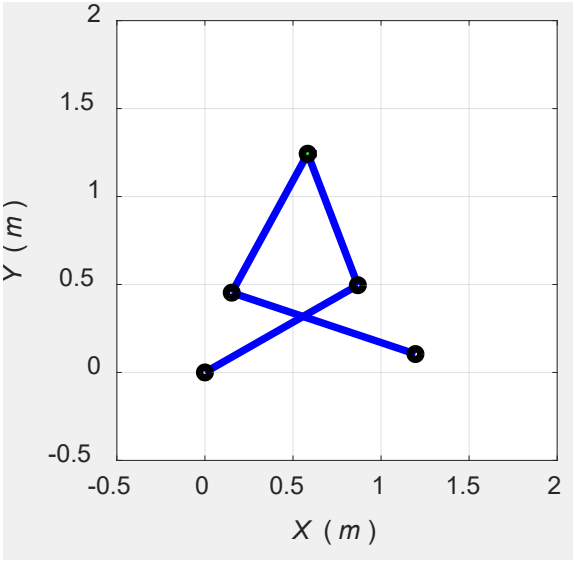
solution branch	θ_2 (deg)	θ_5 (deg)	θ_3 (deg)	θ_4 (deg)
left/left	100	161.5	18.9	61.4
left/right	100	75	18.9	175.1
right/left	29.7	161.5	110.9	61.4
right/right	29.7	75	110.9	175.1



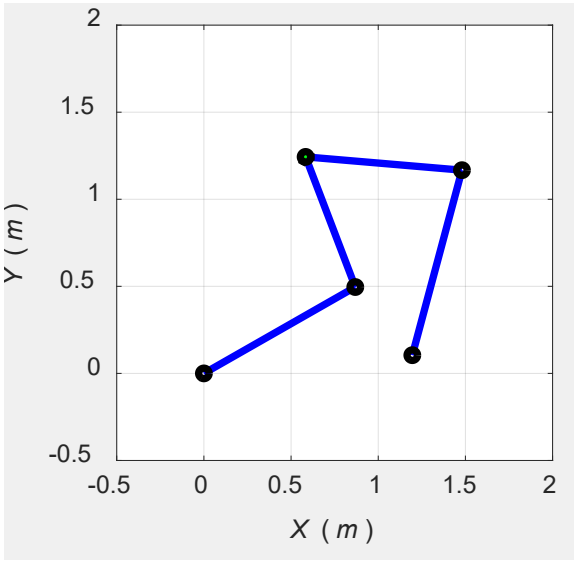
Left / Left



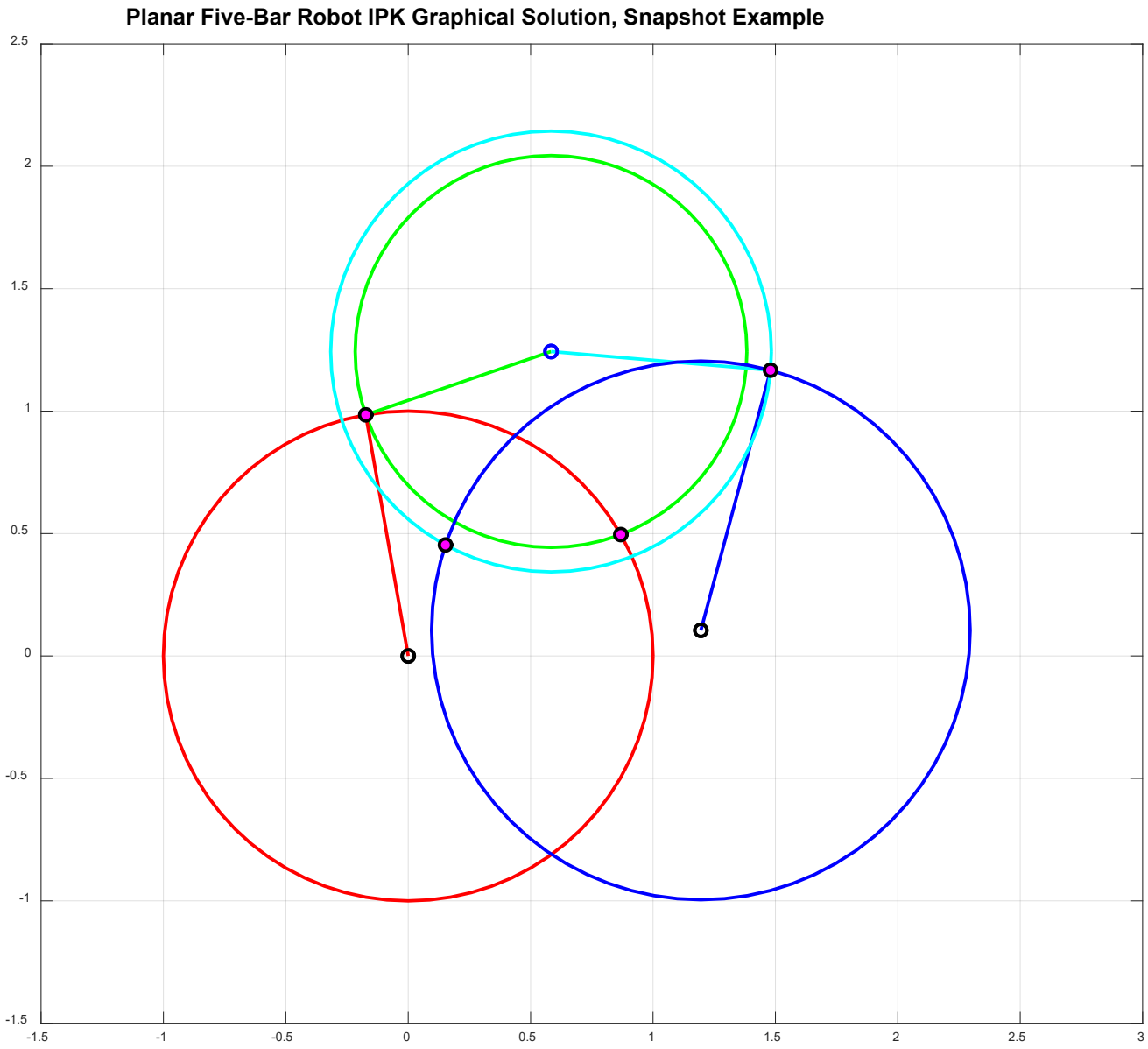
Left / Right



Right / Left



Right / Right



FiveBarGraphical.m

Five-Bar Parallel Robot Workspace

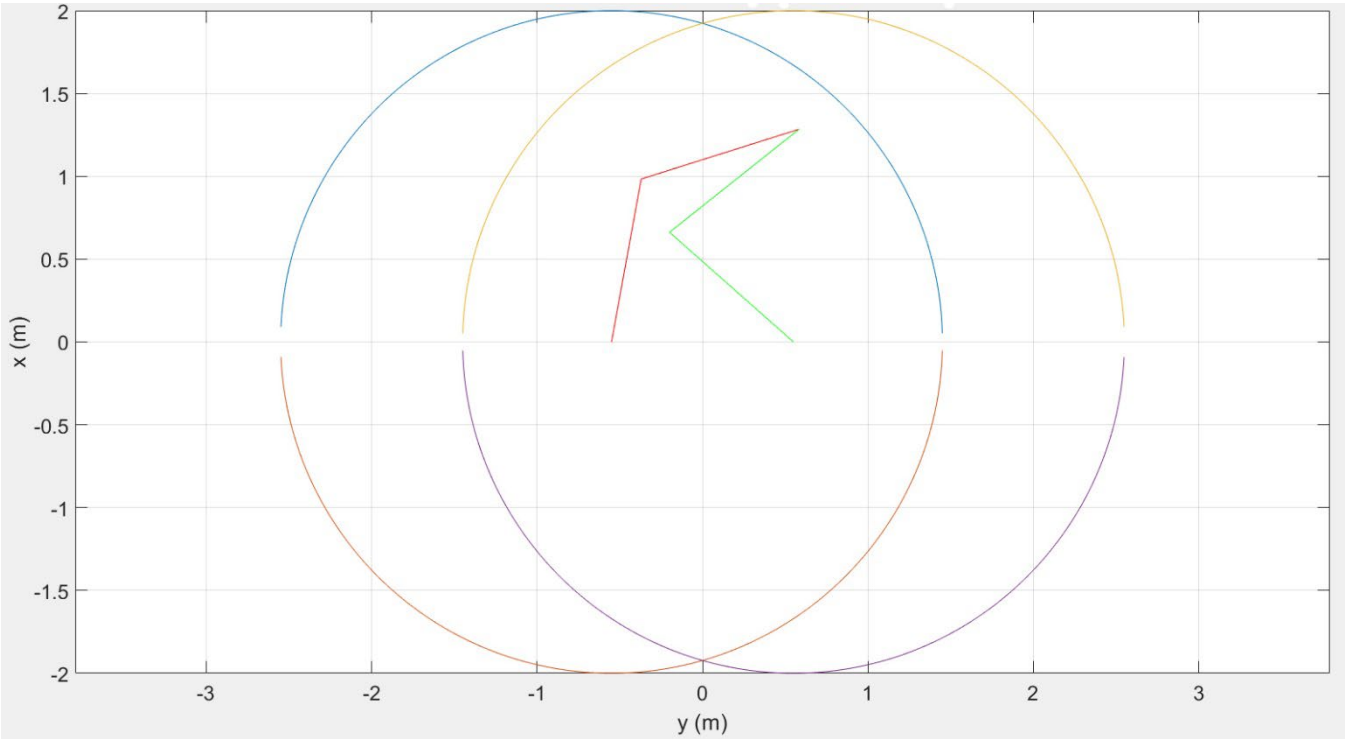
Constant robot dimensions: $r_1 = 2,\ \theta_1 = 0^\circ,\ r_2 = 1,\ r_3 = 1,\ r_4 = 1,\ r_5 = 1$ (m)

The MATLAB figure below, created by OU student Ryan Radecki, shows the workspace of the 2-dof Five-Bar Parallel robot, in an inverse position simulation context. The reachable workspace is the intersection of the two circles, the shape of an American football.

Five-Bar Parallel Robot Workspace

We see that when $r_2 = r_3 = r_4 = r_5$, the workspace is optimal since there are no holes (annuli) in either left or right circles.

However, the $r_1 = 2$ case shown above is non-optimal; r_1 should be minimally greater than the $r_2 = r_3 = r_4 = r_5$ value, to ensure links 2 and 5 can pass through the ground link to reach the lower workspace. The figure below shows a larger workspace, with $r_1 = 1.1$.



Larger Five-Bar Parallel Robot Workspace

9.2.3 Velocity Kinematics

Step 1. The five-bar robot **Position Analysis** must first be complete.

Step 2. Identify the five-bar robot velocity parameters.

$\dot{\underline{\theta}}_i = \underline{\omega}_i$ (i = 2,3,4,5) is the absolute angular velocity of link i. $\dot{\underline{\theta}}_1 = \underline{\omega}_1 = \underline{0}$ since link 1, the fixed ground link, cannot rotate.

The vector loop-closure equations developed in the Position Kinematics section are the starting point for deriving the velocity equations.

Forward Velocity Kinematics

Step 3. State the Problem

Given: r_1, θ_1, r_2, r_3, r_4, r_5; angles θ_2, θ_3, θ_4, and θ_5; plus $\dot{\theta}_2$ and $\dot{\theta}_5$

Find: end-effector translational velocity $\dot{\underline{B}} = \{\dot{x} \quad \dot{y}\}^T$ plus $\dot{\theta}_3$ and $\dot{\theta}_4$

Step 4. Derive the velocity equations. Take the first time derivative of the vector loop closure equations from position analysis, in *XY* component form, repeated below.

$$r_2c_2 + r_3c_3 = r_1c_1 + r_5c_5 + r_4c_4$$
$$r_2s_2 + r_3s_3 = r_1s_1 + r_5s_5 + r_4s_4$$

The first time derivative of the position equations is given below:

$$-r_2\dot{\theta}_2s_2 - r_3\dot{\theta}_3s_3 = -r_5\dot{\theta}_5s_5 - r_4\dot{\theta}_4s_4$$
$$r_2\dot{\theta}_2c_2 + r_3\dot{\theta}_3c_3 = \quad r_5\dot{\theta}_5c_5 + r_4\dot{\theta}_4c_4$$

Since all links are rigid (i.e. no links are changing lengths), all $\dot{r}_i = 0$ and thus the four *XY* pairs of terms above represent the tangential velocities of the endpoint of each link. Gathering unknowns on the LHS, and substituting the following terms yields the following matrix-vector equations, two linear equations in the two unknowns $\dot{\theta}_3$ and $\dot{\theta}_4$:

$$\begin{bmatrix} a & b \\ d & e \end{bmatrix} \begin{Bmatrix} \dot{\theta}_3 \\ \dot{\theta}_4 \end{Bmatrix} = \begin{Bmatrix} c \\ f \end{Bmatrix}$$

$$a = \quad r_3s_3 \qquad d = -r_3c_3$$
$$b = -r_4s_4 \qquad e = \quad r_4c_4$$
$$c = -r_2\dot{\theta}_2s_2 + r_5\dot{\theta}_5s_5 \qquad f = \quad r_2\dot{\theta}_2c_2 - r_5\dot{\theta}_5c_5$$

Step 5. Solve the velocity equations for the unknowns $\dot{\theta}_3$ and $\dot{\theta}_4$.

The solution of the above matrix-vector set of equations is:

$$\dot{\theta}_3 = \frac{ce - bf}{ae - bd} \qquad\qquad \dot{\theta}_4 = \frac{af - cd}{ae - bd}$$

With $\dot{\theta}_3$ and $\dot{\theta}_4$ known, the end-effector translational velocity $\underline{\dot{B}} = \{\dot{x} \quad \dot{y}\}^T$ can be found either from the link2/link3 dyad or the link5/link4 dyad; both yield identical results.

$$\underline{\dot{B}} = \begin{Bmatrix} \dot{x} \\ \dot{y} \end{Bmatrix} = \begin{Bmatrix} -r_2\dot{\theta}_2 s_2 - r_3\dot{\theta}_3 s_3 \\ r_2\dot{\theta}_2 c_2 + r_3\dot{\theta}_3 c_3 \end{Bmatrix} \qquad\qquad \underline{\dot{B}} = \begin{Bmatrix} \dot{x} \\ \dot{y} \end{Bmatrix} = \begin{Bmatrix} -r_5\dot{\theta}_5 s_5 - r_4\dot{\theta}_4 s_4 \\ r_5\dot{\theta}_5 c_5 + r_4\dot{\theta}_4 c_4 \end{Bmatrix}$$

Planar Five-Bar Parallel Robot Forward Velocity singularity condition

When does the above forward velocity solution fail? At a robot singularity, when the determinant of the coefficient matrix *A* goes to zero, the forward velocity solution fails. This case would require division by zero, yielding infinite $\dot{\theta}_3$ and $\dot{\theta}_4$. Let's see what this means physically.

$$|A| = ae - bd = (r_3 s_3)(r_4 c_4) - (-r_4 s_4)(-r_3 c_3) = r_3 r_4 s_3 c_4 - r_3 r_4 c_3 s_4 = -r_3 r_4 \sin(\theta_4 - \theta_3)$$

Where we used the trigonometric identity $\sin(a-b) = \sin a \cos b - \cos a \sin b$.

$|A| = 0$ when $\sin(\theta_4 - \theta_3) = 0$, i.e. when:

$$\theta_4 - \theta_3 = 0^\circ, 180^\circ, \cdots$$

Physically, this happens when links 3 and 4 are collinear (straight out or folded on top of each other). This corresponds to the boundary between the two Forward Position Kinematics solution branches.

Inverse Velocity Kinematics

Given: r_1, θ_1, r_2, r_3, r_4, r_5; angles θ_2, θ_3, θ_4, and θ_5; plus the desired end-effector translational velocity $\underline{\dot{B}} = \{\dot{x} \quad \dot{y}\}^T$

Find: required actuator angular rates $\dot{\theta}_2$ and $\dot{\theta}_5$, plus passive angular rates $\dot{\theta}_3$ and $\dot{\theta}_4$

The actuator unknowns $\dot{\theta}_2$ and $\dot{\theta}_5$ can be found independently from the two end-effector velocity $\underline{\dot{B}} = \{\dot{x} \quad \dot{y}\}^T$ equations from Step 5 in the Forward Velocity Kinematics subsection above. These equations are repeated below, written in matrix-vector form. Passive unknown angular rate $\dot{\theta}_3$ can be found along with $\dot{\theta}_2$, and passive unknown angular rate $\dot{\theta}_4$ can be found along with $\dot{\theta}_5$.

$$\begin{bmatrix} -r_2 s_2 & -r_3 s_3 \\ r_2 c_2 & r_3 c_3 \end{bmatrix} \begin{Bmatrix} \dot{\theta}_2 \\ \dot{\theta}_3 \end{Bmatrix} = \begin{Bmatrix} \dot{x} \\ \dot{y} \end{Bmatrix} \qquad \begin{bmatrix} -r_4 s_4 & -r_5 s_5 \\ r_4 c_4 & r_5 c_5 \end{bmatrix} \begin{Bmatrix} \dot{\theta}_4 \\ \dot{\theta}_5 \end{Bmatrix} = \begin{Bmatrix} \dot{x} \\ \dot{y} \end{Bmatrix}$$

and the inverse velocity solutions are:

$$\dot{\theta}_2 = \frac{\dot{x}c_3 + \dot{y}s_3}{r_2 \sin(\theta_3 - \theta_2)} \qquad \dot{\theta}_4 = \frac{\dot{x}c_5 + \dot{y}s_5}{r_4 \sin(\theta_5 - \theta_4)}$$

$$\dot{\theta}_3 = \frac{-\dot{x}c_2 - \dot{y}s_2}{r_3 \sin(\theta_3 - \theta_2)} \qquad \dot{\theta}_5 = \frac{-\dot{x}c_4 - \dot{y}s_4}{r_5 \sin(\theta_5 - \theta_4)}$$

Planar Five-Bar Parallel Robot Inverse Velocity singularity condition

When does the above inverse velocity solution fail? At an inverse velocity robot singularity (different from the forward velocity singularity condition presented earlier), when the determinant of the coefficient matrix A_{23} goes to zero, the inverse velocity solution fails. This case would require division by zero, yielding infinite $\dot{\theta}_2$ and $\dot{\theta}_3$. Also, when the determinant of the coefficient matrix A_{45} goes to zero, the inverse velocity solution fails. This case would require division by zero, yielding infinite $\dot{\theta}_4$ and $\dot{\theta}_5$. Let's see what this means physically.

$$|A_{23}| = r_2 r_3 \sin(\theta_3 - \theta_2) \qquad |A_{45}| = r_4 r_5 \sin(\theta_5 - \theta_4)$$

$$|A_{23}| = 0 \quad \text{when} \quad \sin(\theta_3 - \theta_2) = 0 \qquad |A_{45}| = 0 \quad \text{when} \quad \sin(\theta_5 - \theta_4) = 0$$

$$\text{i.e. when } \theta_3 - \theta_2 = 0^\circ, 180^\circ, \cdots \qquad \text{i.e. when } \theta_5 - \theta_4 = 0^\circ, 180^\circ, \cdots$$

Physically, this happens when links 2 and 3 are collinear (straight out or folded on top of each other). This corresponds to the boundary between the two Inverse Position Kinematics solution branches for the link2/link3 dyad. This also occurs when links 4 and 5 are collinear (straight out or folded on top of each other). This corresponds to the boundary between the two Inverse Position Kinematics solution branches for the link5/link4 dyad. Either case will yield singularity, i.e. the two conditions need not occur together.

Planar Five-Bar Parallel Robot Velocity Kinematics for active joints only

Often it will be convenient to solve the Forward and Inverse Velocity Kinematics problem for the planar five-bar parallel robot, ignoring the passive joints velocity variables $\dot{\theta}_3$ and $\dot{\theta}_4$. The solution to the Inverse Velocity Kinematics problem for active joints only is immediately available from the above expressions, i.e.:

$$\dot{\theta}_2 = \frac{\dot{x}c_3 + \dot{y}s_3}{r_2 \sin(\theta_3 - \theta_2)}$$

$$\dot{\theta}_5 = \frac{-\dot{x}c_4 - \dot{y}s_4}{r_5 \sin(\theta_5 - \theta_4)}$$

This result is written in matrix-vector form below:

$$\begin{Bmatrix} \dot{\theta}_2 \\ \dot{\theta}_5 \end{Bmatrix} = \begin{bmatrix} \dfrac{c_3}{r_2 \sin(\theta_3 - \theta_2)} & \dfrac{s_3}{r_2 \sin(\theta_3 - \theta_2)} \\ \dfrac{-c_4}{r_5 \sin(\theta_5 - \theta_4)} & \dfrac{-s_4}{r_5 \sin(\theta_5 - \theta_4)} \end{bmatrix} \begin{Bmatrix} \dot{x} \\ \dot{y} \end{Bmatrix}$$

Inverse velocity solution

$$\{\dot{\boldsymbol{\Theta}}\} = [\mathbf{M}]\{\dot{\mathbf{X}}\}$$

where **[M]** is the inverse of the Jacobian matrix **[J]**, $[\mathbf{M}] = [\mathbf{J}]^{-1}$. Ironically, to solve the Forward Velocity Problem ignoring the passive joint rates, we must invert the above set of equations:

$$\{\dot{\mathbf{X}}\} = [\mathbf{M}]^{-1}\{\dot{\boldsymbol{\Theta}}\} = [\mathbf{J}]\{\dot{\boldsymbol{\Theta}}\}$$

Forward velocity solution

$$\begin{Bmatrix} \dot{x} \\ \dot{y} \end{Bmatrix} = \begin{bmatrix} \dfrac{-r_2 s_4 \sin(\theta_3 - \theta_2)}{\sin(\theta_3 - \theta_4)} & \dfrac{-r_5 s_3 \sin(\theta_5 - \theta_4)}{\sin(\theta_3 - \theta_4)} \\ \dfrac{r_2 c_4 \sin(\theta_3 - \theta_2)}{\sin(\theta_3 - \theta_4)} & \dfrac{r_5 c_3 \sin(\theta_5 - \theta_4)}{\sin(\theta_3 - \theta_4)} \end{bmatrix} \begin{Bmatrix} \dot{\theta}_2 \\ \dot{\theta}_5 \end{Bmatrix}$$

From an inspection of the above alternate Forward and Inverse Velocity solutions for the five-bar robot (which necessarily must be equivalent to those solutions presented earlier), we see that the same exact singularity conditions exist that were identified earlier.

Planar Five-Bar Parallel Robot further analyses

Further analysis is presented in the EE/ME 4290/5290 Supplement for the planar five-bar parallel robot: Acceleration analysis and Inverse Dynamics analysis.

9.3 Intersection of Two Circles

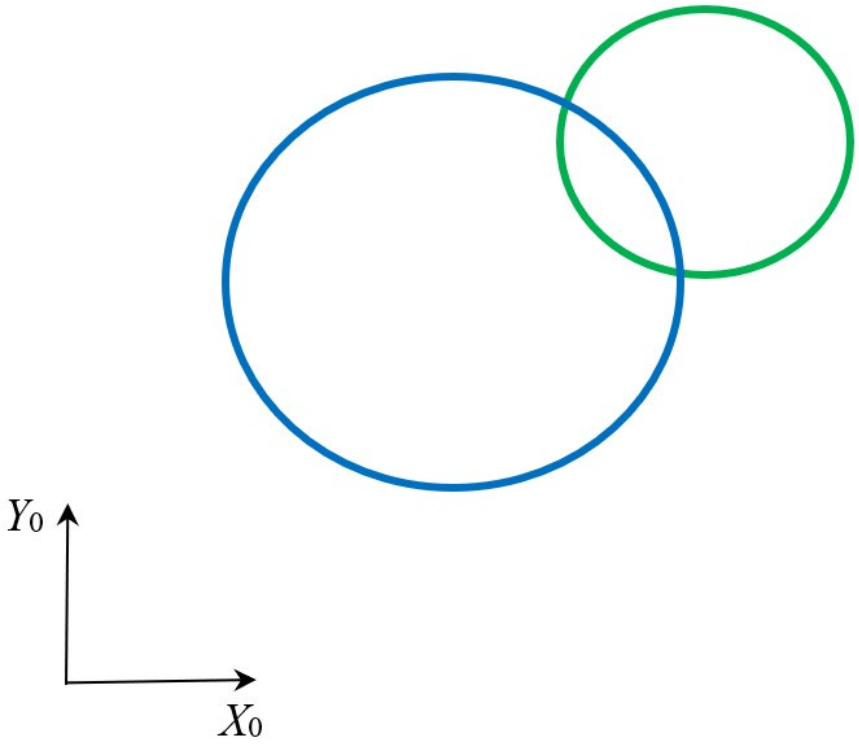
Derive the analytical solution for the intersection points of two planar circles.

This solution is very useful for many planar mechanisms kinematics problems, the IPK solution of the planar 3R robot, plus for many planar parallel robot kinematics problems.

The two circles must be totally **general** (i.e. any centers and any radii). The use of a different coordinate frame, coordinate transformations, and homogeneous transformation matrix concepts from robotics can be used to simplify this problem dramatically! The following notation is used for the two general circles:

circle 1	center (a_x, a_y)	radius r_1
circle 2	center (b_x, b_y)	radius r_2

Original problem figure:

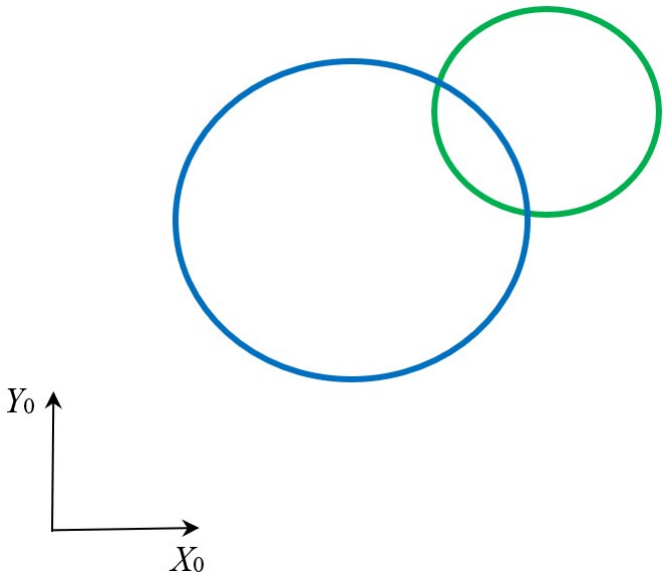


The two circle equations are:

$$(x-a_x)^2+(y-a_y)^2=r_1^2$$
$$(x-b_x)^2+(y-b_y)^2=r_2^2$$

It is possible to expand these equations, subtract one from the other to eliminate the squared unknowns, solve for y as a function of x, substitute into one equation to solve for x and then y. However, this is quite complicated (it is presented in the EE/ME 4290/5290 Supplement) and a much simpler method is presented here.

These two circles are defined in the {0} Cartesian coordinate frame, with general center locations. Define the {1} Cartesian coordinate frame whose origin is at the center point for the first circle and whose $\hat{X}_1$ axis is pointing to the other circle center point. Simplified problem figure:



In this case the two circles are of center (0,0) radius r_1 and center (L,0) radius r_2, with respect to {1} coordinates. The two circle equations in {1} are:

$$x^2 + y^2 = r_1^2$$
$$(x-L)^2 + y^2 = r_2^2$$

where L is the known distance between the two given circle centers in {0}, $L = \sqrt{(b_x - a_x)^2 + (b_y - a_y)^2}$. Expanding the second the equation and subtracting the second equation from the first yields:

$$x^2 + y^2 = r_1^2$$
$$x^2 - 2Lx + L^2 + y^2 = r_2^2$$

$$2Lx - L^2 = r_1^2 - r_2^2$$

From which the x coordinate of the two solutions can be easily found, in {1} coordinates:

$${}^1x = \frac{r_1^2 - r_2^2 + L^2}{2L}$$

Using the first equation, we easily find the two possible y solutions, also in {1} coordinates:

$${}^1y_{1,2} = \pm\sqrt{r_1^2 - {}^1x^2}$$

The two solutions in {1} coordinates are $({}^1x, {}^1y_1)$ and $({}^1x, {}^1y_2)$.

Now we need to transform these solutions to the {0} coordinate frame using our standard methods from robotics:

$$\left\{{}^0P\right\} = \left[{}^0_1T\right]\left\{{}^1P\right\} \qquad \begin{Bmatrix} {}^0x_{1,2} \\ {}^0y_{1,2} \\ 0 \\ 1 \end{Bmatrix} = \begin{bmatrix} c\phi & -s\phi & 0 & a_x \\ s\phi & c\phi & 0 & a_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} {}^1x \\ {}^1y_{1,2} \\ 0 \\ 1 \end{Bmatrix} \qquad \phi = \tan^{-1}\left[\frac{b_y - a_y}{b_x - a_x}\right]$$

One must use the quadrant-specific **`atan2`** function for calculating ϕ above.

In general there are two solutions to this problem, indicated by the 1,2 subscripts above. In the special case when $L = r_1 + r_2$, the two solutions degenerate to only one, and the circles are tangent. In this special case ${}^1y_{1,2} = 0$. Further, when $L > r_1 + r_2$, there are no solutions since the two circles do not intersect. In that case ${}^1y_{1,2} = \pm\sqrt{r_1^2 - {}^1x^2}$ will be imaginary. Special cases figures:

Parallel Robots Analysis

The on-line EE/ME 4290/5290 Supplement presents the kinematics of some classic parallel robots in detail: 3-dof planar **3-RPR** robot, 2-dof planar 5-bar **RRRRR** robot acceleration and inverse dynamics, 6-dof NIST RoboCrane cable-suspended robot, 3-dof spatial translational-only Delta Robot (**3-RUU** and **3-PUU** versions), and the 6-dof **6-UPS** Stewart Platform.