BOB WILLIAMS

Robot Mechanics

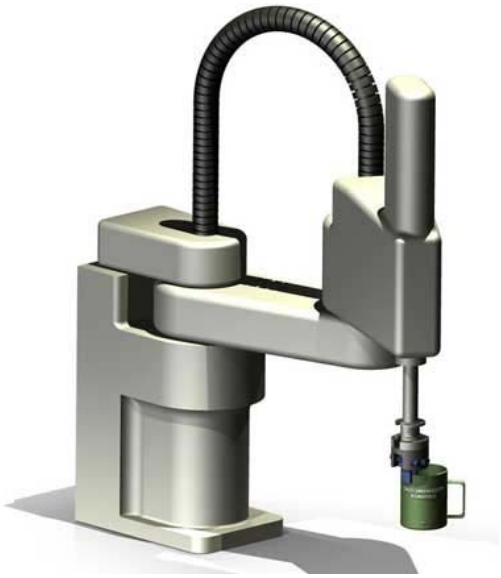
on-line NotesBook Supplement

Robot Mechanics

Dr. Robert L. Williams II
Mechanical Engineering
Ohio University

NotesBook Supplement for
EE/ME 4290/5290 Mechanics and Control of Robotic Manipulators

williar4@ohio.edu
ohio.edu/mechanical-faculty/williams



4-dof SCARA Serial Robot

6-dof Stewart Platform Parallel Robot

These notes supplement the EE/ME 4290/5290 NotesBook by Dr. Bob

This document presents supplemental notes to accompany the EE/ME 4290/5290 NotesBook. The outline given in the Table of Contents dovetails with and augments the EE/ME 4290/5290 NotesBook outline and thus is incomplete here.

EE/ME 4290/5290 Supplement Table of Contents

4. Forward Pose Kinematics (FPK)

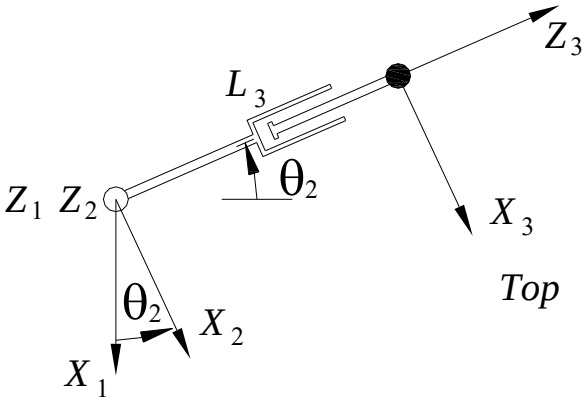
4.3 FPK Examples

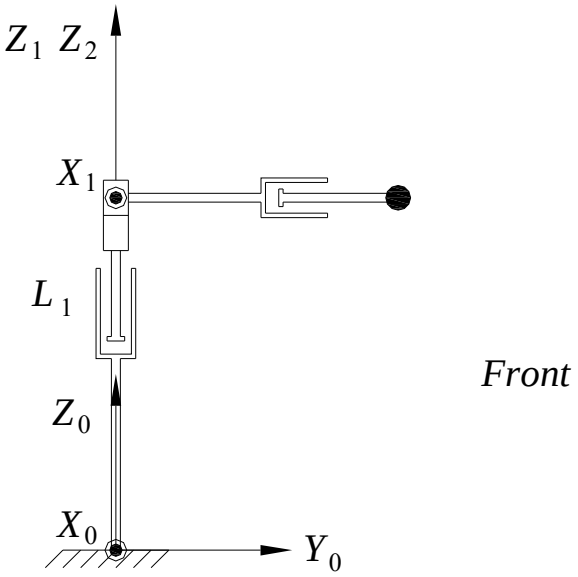
4.3.4 Spatial PRP Cylindrical Serial Robot FPK Solution

Forward Pose Kinematics Symbolic Derivations

Given the cylindrical robot (the 9 constant DH Parameters) and L_1, θ_2, L_3, Calculate $\left[{}^0_3T\right]$.

Note $\left[{}^0_3T\right]$ is represented by $\left\{{}^0X_3\right\} = \left\{x_3 \quad y_3 \quad z_3 \quad \phi\right\}^T$.





i	α_{i-1}	a_{i-1}	d_i	θ_i		
1	0	0	L_1	0	$\Rightarrow$	$\left[{}^0_1T\right]$
2	0	0	0	θ_2	$\Rightarrow$	$\left[{}^1_2T\right]$
3	-90°	0	L_3	0	$\Rightarrow$	$\left[{}^2_3T\right]$

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

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

$$\left[{}^{0}_{1}T(L_1)\right]=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & L_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \left[{}^{1}_{2}T(\theta_2)\right]=\begin{bmatrix} c\theta_2 & -s\theta_2 & 0 & 0 \\ s\theta_2 & c\theta_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \left[{}^{2}_{3}T(L_3)\right]=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & L_3 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\left[{}^{0}_{3}T\right]=\left[{}^{0}_{1}T(L_1)\right]\left[{}^{1}_{2}T(\theta_2)\right]\left[{}^{2}_{3}T(L_3)\right]=\begin{bmatrix} c\theta_2 & 0 & -s\theta_2 & -L_3 s\theta_2 \\ s\theta_2 & 0 & c\theta_2 & L_3 c\theta_2 \\ 0 & -1 & 0 & L_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\left\{{}^{0}X_3\right\}=\left\{x_3 \quad y_3 \quad z_3 \quad \phi\right\}^T$$
$$=\left\{-L_3 s\theta_2 \quad L_3 c\theta_2 \quad L_1 \quad \theta_2\right\}^T$$

(interpret geometrically)

In this robot the point of interest is the origin of {3} so no hand frame {*H*} is required. Also, the kinematic base frame {0} is already on the floor, so there is no need for another base frame {*B*}.

Spatial Cylindrical Robot FPK Examples

1) Given $L_1=3, \theta_2=30^\circ, L_3=2$, calculate $\left[{}^{0}_{3}T\right]$.

$$\left[{}^{0}_{3}T\right]=\begin{bmatrix} 0.866 & 0 & -0.5 & -1 \\ 0.5 & 0 & 0.866 & 1.732 \\ 0 & -1 & 0 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

or

$$\left\{{}^{0}X_3\right\}=\left\{x_3 \quad y_3 \quad z_3 \quad \phi\right\}^T$$
$$=\left\{-1 \quad 1.732 \quad 3 \quad 30^\circ\right\}^T$$

2) Given $L_1=2, \theta_2=-90^\circ, L_3=1$, calculate $\left[{}^{0}_{3}T\right]$.

$$\left[{}^{0}_{3}T\right]=\begin{bmatrix} 0 & 0 & 1 & 1 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

or

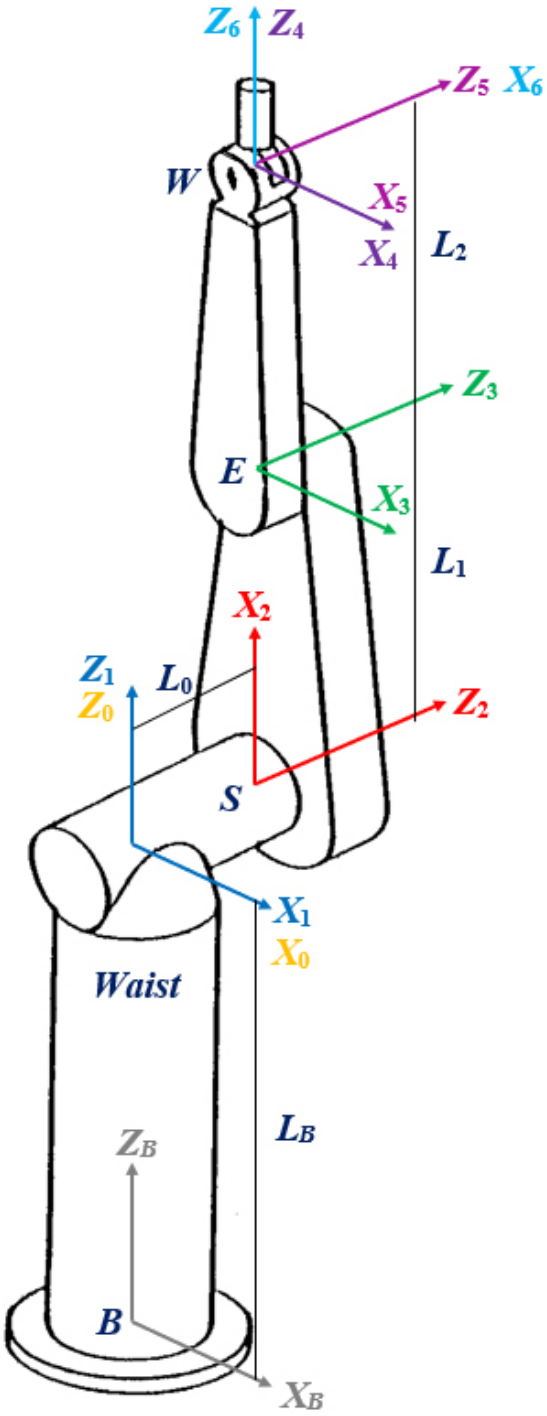
$$\left\{{}^{0}X_3\right\}=\left\{x_3 \quad y_3 \quad z_3 \quad \phi\right\}^T$$
$$=\left\{1 \quad 0 \quad 2 \quad -90^\circ\right\}^T$$

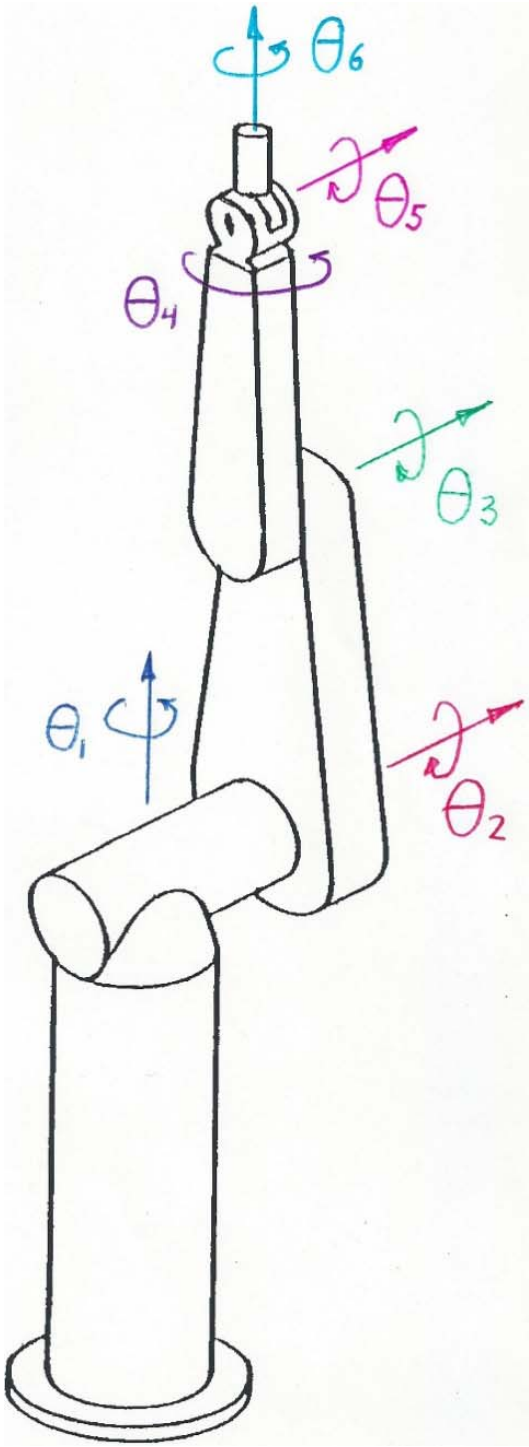
Check both results with sketches (top and front views). Be sure to include the {3} and {0} *XYZ* axes to check the orientation $\left[{}^{0}_{3}R\right]$ in addition to the position vector $\left\{{}^{0}P_3\right\}$.

4.3.5 Spatial 6R PUMA Serial Robot FPK Solution

Spatial 6R PUMA Robot ***Forward Pose Kinematics Symbolic Derivations***

Given the robot (the 18 constant DH Parameters) and $(\theta_1,\theta_2,\theta_3,\theta_4,\theta_5,\theta_6)$, Calculate $\left[{}^{0}_{6}T\right]$ and $\left[{}^{B}_{H}T\right]$.





Step 1. Derive all n neighboring $\left[{}^{i-1}_{i}T\right]$ matrices by substituting each row of the DH parameters table into the formula we derived for $\left[{}^{i-1}_{i}T\right]$ on the previous page. Row i yields the homogeneous transformation matrix $\left[{}^{i-1}_{i}T\right]$. Do this for all n active dof, i.e. all n rows of the DH parameters table.

Example: Spatial 6-dof 6R PUMA Robot DH Parameters Table (figure given earlier)

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

Substitute each row of the DH parameters table into the equation below for $\left[{}^{i-1}_{i}T\right]$. Evaluate the angle offsets also, i.e. use the trigonometric sum-of-angle formulas (given on the next page).

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

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

$$\left[{}^{0}_{1}T\right] = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \left[{}^{1}_{2}T\right] = \begin{bmatrix} s_2 & c_2 & 0 & 0 \\ 0 & 0 & 1 & L_0 \\ c_2 & -s_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \left[{}^{2}_{3}T\right] = \begin{bmatrix} -s_3 & -c_3 & 0 & L_1 \\ c_3 & -s_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\left[{}^{3}_{4}T\right] = \begin{bmatrix} c_4 & -s_4 & 0 & 0 \\ 0 & 0 & -1 & -L_2 \\ s_4 & c_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \left[{}^{4}_{5}T\right] = \begin{bmatrix} c_5 & -s_5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s_5 & -c_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \left[{}^{5}_{6}T\right] = \begin{bmatrix} -s_6 & -c_6 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ c_6 & -s_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Step 2. Use matrix multiplication with consecutive homogeneous transformation matrices in the correct order to yield the forward pose kinematics solution.

$\left[{}^{0}_{6}T\right]$ gives the pose (position and orientation) of the last active joint/link Cartesian coordinate frame {6} with respect to the kinematic base Cartesian coordinate frame {0}. It is a function of all $n = 6$ joint variables.

PUMA FPK Solution

$$\left[{}^{0}_{6}T\right]=\left[{}^{0}_{1}T(\theta_1)\right]\left[{}^{1}_{2}T(\theta_2)\right]\left[{}^{2}_{3}T(\theta_3)\right]\left[{}^{3}_{4}T(\theta_4)\right]\left[{}^{4}_{5}T(\theta_5)\right]\left[{}^{5}_{6}T(\theta_6)\right]$$

There are 2 different representations for the FPK solution: $\left[{}^{0}_{6}T\right]\leftrightarrow\left\{{}^{0}X_{6}\right\}$, where $\left\{{}^{0}X_{6}\right\}=\left\{x \quad y \quad z \quad \gamma \quad \beta \quad \alpha\right\}^{T}$, and we use *Z-Y-X* ($\alpha$–$\beta$–$\gamma$) Euler angles convention.

This FPK solution for the spatial 6-dof 6R PUMA robot can be simplified by deriving the analytical results separately for the arm joints $(\theta_1,\theta_2,\theta_3)$, primarily responsible for positioning in 3D, and the wrist joints $(\theta_4,\theta_5,\theta_6)$, primarily responsible for orienting in 3D.

$$\left[{}^{0}_{6}T\right]=\left[{}^{0}_{3}T(\theta_1,\theta_2,\theta_3)\right]\left[{}^{3}_{6}T(\theta_4,\theta_5,\theta_6)\right]$$

PUMA Arm Angles FPK Solution

Take advantage of parallel *Z* axes in the consecutive arm frames {2} and {3} – use trigonometric sum-of-angles formulas (given above).

$$\left[{}^{0}_{3}T(\theta_1,\theta_2,\theta_3)\right]=\left[{}^{0}_{1}T(\theta_1)\right]\left[{}^{1}_{3}T(\theta_2,\theta_3)\right]$$

$$\left[{}^{0}_{3}T(\theta_1,\theta_2,\theta_3)\right]=\begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} c_{23} & -s_{23} & 0 & L_1 s_2 \\ 0 & 0 & 1 & L_0 \\ -s_{23} & -c_{23} & 0 & L_1 c_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\left[{}^{0}_{3}T(\theta_1,\theta_2,\theta_3)\right]=\begin{bmatrix} c_1 c_{23} & -c_1 s_{23} & -s_1 & -L_0 s_1 + L_1 c_1 s_2 \\ s_1 c_{23} & -s_1 s_{23} & c_1 & L_0 c_1 + L_1 s_1 s_2 \\ -s_{23} & -c_{23} & 0 & L_1 c_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where the following abbreviations were used:

$$c_{23}=\cos(\theta_2+\theta_3)$$
$$s_{23}=\sin(\theta_2+\theta_3)$$

PUMA Wrist Angles FPK Solution

There are no consecutive parallel *Z* axes in the consecutive wrist frames {4}, {5}, and {6}. Therefore, no sum-of-angles formula simplification is possible.

$$\left[{}^{3}_{6}T(\theta_4,\theta_5,\theta_6)\right]=\left[{}^{3}_{4}T(\theta_4)\right]\left[{}^{4}_{5}T(\theta_5)\right]\left[{}^{5}_{6}T(\theta_6)\right]$$

$$\left[{}^{3}_{6}T(\theta_4,\theta_5,\theta_6)\right]=\begin{bmatrix} -s_4c_6-c_4c_5s_6 & s_4s_6-c_4c_5c_6 & c_4s_5 & 0 \\ -s_5s_6 & -s_5c_6 & -c_5 & -L_2 \\ c_4c_6-s_4c_5s_6 & -c_4s_6-s_4c_5c_6 & s_4s_5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

It is left to the reader to perform the symbolic multiplication $\left[{}^{0}_{6}T\right]=\left[{}^{0}_{3}T\right]\left[{}^{3}_{6}T\right]$; the result is very complicated and can be found by using symbolic math on the computer (e.g. the MATLAB Symbolic Toolbox). Alternatively, these two matrices can be evaluated and multiplied numerically, using $\left[{}^{0}_{6}T\right]=\left[{}^{0}_{3}T\right]\left[{}^{3}_{6}T\right]$.

Here we will present one aspect of this multiplication. Due to the spherical wrist (wrist frames {4}, {5}, and {6} share a common origin), the position vector $\left\{{}^{0}P_6\right\}$ is a function only of the first three joint angles.

$$\left\{{}^{0}P_6(\theta_1,\theta_2,\theta_3)\right\}=\left\{{}^{0}P_4\right\}=\left[{}^{0}_{3}T\right]\left\{{}^{3}P_4\right\}=\begin{Bmatrix} -L_0s_1+L_1c_1s_2+L_2c_1s_{23} \\ L_0c_1+L_1s_1s_2+L_2s_1s_{23} \\ L_1c_2+L_2c_{23} \end{Bmatrix}$$

Where constant vector $\left\{{}^{3}P_4\right\}=\begin{Bmatrix}0 & -L_2 & 0\end{Bmatrix}^T$. In this way the PUMA FPK symbolic expressions are partially decoupled. The position vector $\left\{{}^{0}P_6\right\}$ is only a function of $(\theta_1,\theta_2,\theta_3)$, but the rotation matrix $\left[{}^{0}_{6}R\right]$ is a complicated function of all six joint angles. This fact is used to simplify the Inverse Pose Kinematics solution.

Additional, fixed transforms

The above $\left[{}^{0}_{6}T\right]$ result is for the active joints only – often we need to expand this result to include additional transformations that are constant. For example, the kinematic base frame {0} may be located at the shoulder of the robot, while another base frame {*B*} may be mounted on the floor. Also, the Forward Pose Kinematics expressions will be simplest if {6} is located at the last active wrist joint; if a tool, gripper, or other end-effector is attached we need another frame of interest (say {*H*} for hand) attached; {*H*} is rigidly connected to {6} (i.e. no more joints in between) but offset by some distance.

The overall Forward Pose Kinematics Homogeneous Transformation Matrix is given in generic form below. Note that the fixed matrices $\left[{}^{B}_{0}T\right]$ and $\left[{}^{6}_{H}T\right]$ are not determined by DH parameters since there is no active joint involved in those two transformations. Instead, we simply determine these matrices by inspection. Make the orientation identical if possible.

$$\left[{}^{B}_{H}T\right]=\left[{}^{B}_{0}T(L_B)\right]\left[{}^{0}_{6}T(\theta_1,\theta_2,\theta_3,\theta_4,\theta_5,\theta_6)\right]\left[{}^{6}_{H}T(L_H)\right]$$

$$\left[{}^{B}_{0}T(L_B)\right]=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&L_B\\0&0&0&1\end{bmatrix} \qquad \left[{}^{6}_{H}T(L_H)\right]=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&L_H\\0&0&0&1\end{bmatrix}$$

Sum of Angles Simplification

The first use of the sum-of-angles formulas below is in simplification of $\left[{}^{i-1}_{i}T\right]$ when there are angle offsets for one or more of the θ_i (i = 2,3,6 in the PUMA robot).

If any two (or more) consecutive Z axes are parallel (i.e. consecutive θ_i rotate about parallel Z axes), we can simplify the resulting symbolic Forward Pose Kinematics expressions by using sum of angle formulas.

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

Many common industrial robots have this parallel axes characteristic for at least one pair. First multiply together any two individual Homogeneous Transformation Matrices that represent consecutive parallel axes (shown for joints 2 and 3 of the PUMA above). Take care to keep the proper matrix multiplication order; i.e. use the associative property of matrix multiplication, ***DO NOT*** commute the order of matrices. Then use the above trigonometric formulas to simplify all possible terms to sums of angles before completing the other matrix multiplications.

PUMA Robot FPK Examples

Given fixed robot lengths $L_B=1.0, L_0=0.3, L_1=1.5, L_2=1.2, L_H=0.5$ (m).

1) Given $\{\Theta\}=(10^\circ,20^\circ,30^\circ,40^\circ,50^\circ,60^\circ)$, calculate $\left[{}^{0}_{6}T\right]$ and $\left[{}^{B}_{H}T\right]$.

$$\left[{}^{0}_{6}T\right]=\begin{bmatrix}0.023&0.637&0.771&1.358\\0.030&-0.771&0.636&0.544\\0.999&0.008&-0.036&2.181\\0&0&0&1\end{bmatrix} \qquad \left[{}^{B}_{H}T\right]=\begin{bmatrix}0.023&0.637&0.771&1.744\\0.030&-0.771&0.636&0.862\\0.999&0.008&-0.036&3.163\\0&0&0&1\end{bmatrix}$$

2) Given $\{\Theta\}=(-60^\circ,-50^\circ,-40^\circ,-30^\circ,-20^\circ,-10^\circ)$, calculate $\left[{}^{0}_{6}T\right]$ and $\left[{}^{B}_{H}T\right]$.

$$\left[{}^{0}_{6}T\right]=\begin{bmatrix}0.638&0.699&-0.322&-0.915\\0.437&0.015&0.899&2.184\\0.634&-0.715&-0.296&0.964\\0&0&0&1\end{bmatrix} \qquad \left[{}^{B}_{H}T\right]=\begin{bmatrix}0.638&0.699&-0.322&-1.076\\0.437&0.015&0.899&2.634\\0.634&-0.715&-0.296&1.816\\0&0&0&1\end{bmatrix}$$

5. Inverse Pose Kinematics (IPK)

5.2 Planar 3R Robot IPK Solution

Tangent half-angle substitution derivation

In this subsection we first derive the tangent half-angle substitution using an analytical/trigonometric method. Defining parameter *t* to be:

$$t = \tan\left(\frac{\phi}{2}\right)$$

i.e. the tangent of half of the unknown angle ϕ, we need to derive $\cos\phi$ and $\sin\phi$ as functions of parameter *t*. This derivation requires the trigonometric sum of angles formulae.

$$\cos(a \pm b) = \cos a \cos b \mp \sin a \sin b$$
$$\sin(a \pm b) = \sin a \cos b \pm \cos a \sin b$$

To derive the $\cos\phi$ term as a function of *t*, we start with:

$$\cos\phi = \cos\left(\frac{\phi}{2} + \frac{\phi}{2}\right)$$

The cosine sum of angles formula yields:

$$\cos\phi = \cos^2\left(\frac{\phi}{2}\right) - \sin^2\left(\frac{\phi}{2}\right)$$

Multiplying by a '1', i.e. $\cos^2\left(\frac{\phi}{2}\right)$ over itself yields:

$$\cos\phi = \frac{\cos^2\left(\frac{\phi}{2}\right) - \sin^2\left(\frac{\phi}{2}\right)}{\cos^2\left(\frac{\phi}{2}\right)} \cos^2\left(\frac{\phi}{2}\right) = \left[1 - \tan^2\left(\frac{\phi}{2}\right)\right] \cos^2\left(\frac{\phi}{2}\right)$$

The cosine squared term can be divided by another '1', i.e. $\cos^2\left(\frac{\phi}{2}\right) + \sin^2\left(\frac{\phi}{2}\right) = 1$.

$$\cos\phi = \left[1 - \tan^2\left(\frac{\phi}{2}\right)\right] \left[\frac{\cos^2\left(\frac{\phi}{2}\right)}{\cos^2\left(\frac{\phi}{2}\right) + \sin^2\left(\frac{\phi}{2}\right)}\right]$$

Dividing top and bottom by $\cos^2\left(\frac{\phi}{2}\right)$ yields:

$$\cos\phi = \left[1-\tan^2\left(\frac{\phi}{2}\right)\right]\left[\frac{1}{1+\tan^2\left(\frac{\phi}{2}\right)}\right]$$

Remembering the earlier definition for *t*, this result is the first derivation we need, i.e.:

$$\cos\phi = \frac{1-t^2}{1+t^2}$$

To derive the $\sin\phi$ term as a function of *t*, we start with:

$$\sin\phi = \sin\left(\frac{\phi}{2}+\frac{\phi}{2}\right)$$

The sine sum of angles formula yields:

$$\sin\phi = \sin\left(\frac{\phi}{2}\right)\cos\left(\frac{\phi}{2}\right)+\cos\left(\frac{\phi}{2}\right)\sin\left(\frac{\phi}{2}\right) = 2\sin\left(\frac{\phi}{2}\right)\cos\left(\frac{\phi}{2}\right)$$

Multiplying top and bottom by cosine yields:

$$\sin\phi = 2\frac{\sin\left(\frac{\phi}{2}\right)}{\cos\left(\frac{\phi}{2}\right)}\cos^2\left(\frac{\phi}{2}\right) = 2\tan\left(\frac{\phi}{2}\right)\cos^2\left(\frac{\phi}{2}\right)$$

From the first derivation we learned:

$$\cos^2\left(\frac{\phi}{2}\right) = \frac{1}{1+\tan^2\left(\frac{\phi}{2}\right)}$$

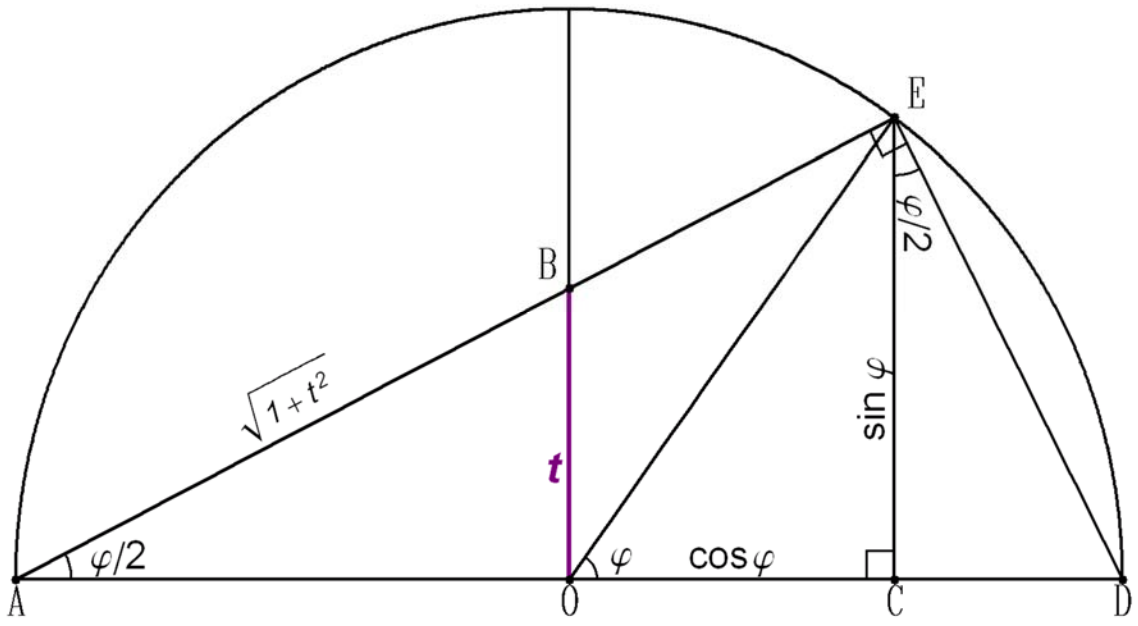
Substituting this term yields:

$$\sin\phi = 2\tan\left(\frac{\phi}{2}\right)\left[\frac{1}{1+\tan^2\left(\frac{\phi}{2}\right)}\right]$$

Remembering the earlier definition for *t*, this result is the second derivation we need, i.e.:

$$\sin\phi = \frac{2t}{1+t^2}$$

The tangent half-angle substitution can also be derived using a graphical method as in the figure below.



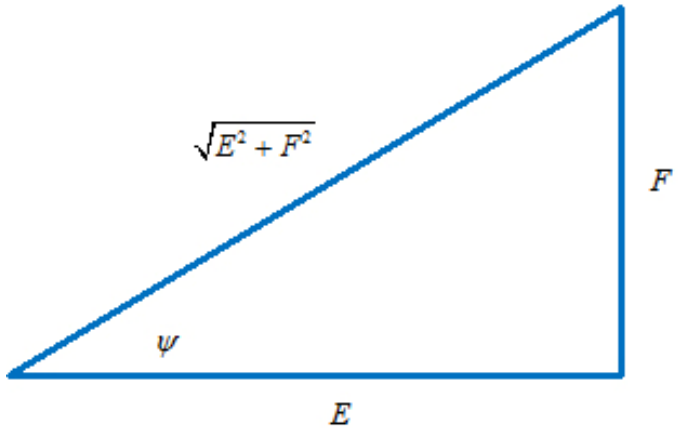
Alternate Planar 3R IPK solution method

The equation form

$$E\cos\theta+F\sin\theta+G=0$$

arises often in the position solutions for mechanisms and robots. It appeared in the θ_1 solution of the Inverse Pose Kinematics solution for the planar 3R robot in the EE/ME 4290/5290 NotesBook and was solved using the tangent half-angle substitution.

Next we present an alternative and simpler solution to this equation. We make two simple trigonometric substitutions based on the figure below.



Clearly from this figure we have:

$$\cos\psi=\frac{E}{\sqrt{E^2+F^2}} \qquad \sin\psi=\frac{F}{\sqrt{E^2+F^2}}$$

In the original equation we divide by $\sqrt{E^2+F^2}$ and rearrange:

$$\frac{E}{\sqrt{E^2+F^2}}\cos\theta+\frac{F}{\sqrt{E^2+F^2}}\sin\theta=\frac{-G}{\sqrt{E^2+F^2}}$$

The two simple trigonometric substitutions yield:

$$\cos\theta\cos\psi+\sin\theta\sin\psi=\frac{-G}{\sqrt{E^2+F^2}}$$

Applying the sum-of-angles formula $\cos(a\pm b)=\cos a\cos b\mp\sin a\sin b$ yields:

$$\cos(\theta - \psi) = \frac{-G}{\sqrt{E^2 + F^2}}$$

And so the solution for θ is:

$$\theta_{1,2} = \psi \pm \cos^{-1}\left[\frac{-G}{\sqrt{E^2 + F^2}}\right]$$

where

$$\psi = \tan^{-1}\left[\frac{F}{E}\right]$$

and the quadrant-specific inverse tangent function **`atan2`** must be used in the above expression for ψ.

There are two solutions for θ, indicated by the subscripts 1,2, since the inverse cosine function is double-valued. Both solutions are correct. We expected these two solutions from the tangent-half-angle substitution approach. They correspond to the elbow-up and elbow-down solutions (the engineer must determine which is which) for the planar 3R robot IPK solution.

For real solutions for θ to exist, we must have

$$-1 \;\leq\; \frac{-G}{\sqrt{E^2 + F^2}} \;\leq\; 1 \qquad \text{or} \qquad 1 \;\geq\; \frac{G}{\sqrt{E^2 + F^2}} \;\geq\; -1$$

If this condition is violated for the planar 3R robot, this means that the given input pose x_3, y_3, ϕ is beyond the robot workspace limits.

Second alternate solution method

For the planar 3R robot IPK problem we now present a second alternate solution method that does not initially result in the equation form $E\cos\theta+F\sin\theta+G=0$. We still square and add the *XY* position equations, but without isolating the θ_2 terms first. Here are the original *XY* equations:

$$x_3 = L_1\cos\theta_1 + L_2\cos(\theta_1+\theta_2)$$
$$y_3 = L_1\sin\theta_1 + L_2\sin(\theta_1+\theta_2)$$

Squaring and adding these equations as-is yields:

$$x_3^2 = L_1^2\cos^2\theta_1 + 2L_1L_2\cos\theta_1\cos(\theta_1+\theta_2) + L_2^2\cos^2(\theta_1+\theta_2)$$
$$y_3^2 = L_1^2\sin^2\theta_1 + 2L_1L_2\sin\theta_1\sin(\theta_1+\theta_2) + L_2^2\sin^2(\theta_1+\theta_2)$$

$$x_3^2+y_3^2 = L_1^2+L_2^2+2L_1L_2\left[\cos\theta_1\cos(\theta_1+\theta_2)+\sin\theta_1\sin(\theta_1+\theta_2)\right]$$

Applying the sum-of-angles formula $\cos(a\pm b)=\cos a\cos b\mp\sin a\sin b$ yields:

$$\begin{aligned} x_3^2+y_3^2 &= L_1^2+L_2^2+2L_1L_2\cos(\theta_1-(\theta_1+\theta_2)) \\ &= L_1^2+L_2^2+2L_1L_2\cos(-\theta_2) \\ &= L_1^2+L_2^2+2L_1L_2\cos\theta_2 \end{aligned}$$

Where we have also used the trigonometric identity $\cos(-a)=\cos a$.

We must deal with trigonometric uncertainty (double-valued inverse trigonometric functions) by using **`cos`** and **`sin`** together, rather than just **`cos`**. The above equation yields:

$$\cos\theta_2 = \frac{x_3^2+y_3^2-L_1^2-L_2^2}{2L_1L_2}$$

From the trigonometric identity $\cos^2\theta_2+\sin^2\theta_2=1$ we obtain:

$$\sin\theta_2 = \pm\sqrt{1-\cos^2\theta_2}$$

The solution for θ_2 is then:

$$\theta_2 = \text{atan2}(\pm\sin\theta_2, \cos\theta_2)$$

The expected two solution sets (elbow-up and elbow-down) come from the on the $\pm$ square root in the $\sin\theta_2$ term.

Knowing θ_2 in this second alternative IPK solution method, we must return to the original *XY* equations and solve for θ_1.

$$x_3 = L_1\cos\theta_1 + L_2\cos(\theta_1+\theta_2) = L_1\cos\theta_1 + L_2(\cos\theta_1\cos\theta_2 - \sin\theta_1\sin\theta_2)$$
$$y_3 = L_1\sin\theta_1 + L_2\sin(\theta_1+\theta_2) = L_1\sin\theta_1 + L_2(\sin\theta_1\cos\theta_2 + \cos\theta_1\sin\theta_2)$$

$$x_3 = (L_1 + L_2\cos\theta_2)\cos\theta_1 - (L_2\sin\theta_2)\sin\theta_1$$
$$y_3 = (L_1 + L_2\cos\theta_2)\sin\theta_1 + (L_2\sin\theta_2)\cos\theta_1$$

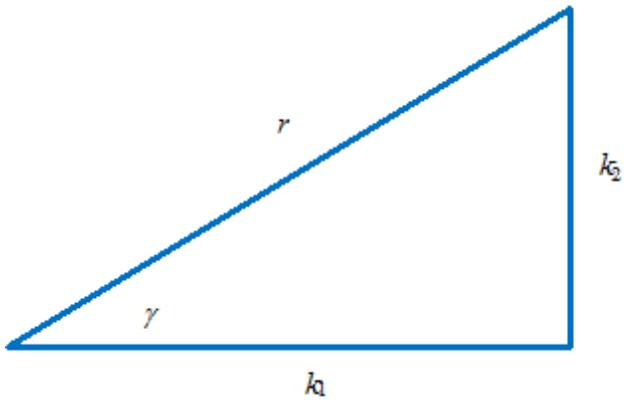
$$x_3 = k_1\cos\theta_1 - k_2\sin\theta_1$$
$$y_3 = k_1\sin\theta_1 + k_2\cos\theta_1$$

where

$$k_1 = L_1 + L_2\cos\theta_2$$
$$k_2 = L_2\sin\theta_2$$

Now, with θ_2 known, these two *XY* equations are not independent. Notice both have the form $E\cos\theta_1 + F\sin\theta_1 + G = 0$. We know two methods to solve this type of equation (tangent half-angle substitution, plus the $\cos\psi$ and $\sin\psi$ simple trigonometric substitution). Either equation can be solved for θ_1 and will yield identical results, one unique θ_1 for each θ_2 branch (elbow-up and elbow-down).

However, if we continue to use both equations, an interesting geometric interpretation will appear. Let us make the following polar substitution, very similar to the previous $\cos\psi$ and $\sin\psi$ simple trigonometric substitution (see the figure below).



$$k_1 = r\cos\gamma$$
$$k_2 = r\sin\gamma$$

where

$$r = +\sqrt{k_1^2 + k_2^2}$$
$$\gamma = \tan^{-1}\left[\frac{k_2}{k_1}\right]$$

We only use the positive square root term in *r* and the quadrant-specific inverse tangent function **atan2** must be used in the above expression for γ.

Applying this substitution to the *XY* equations yields:

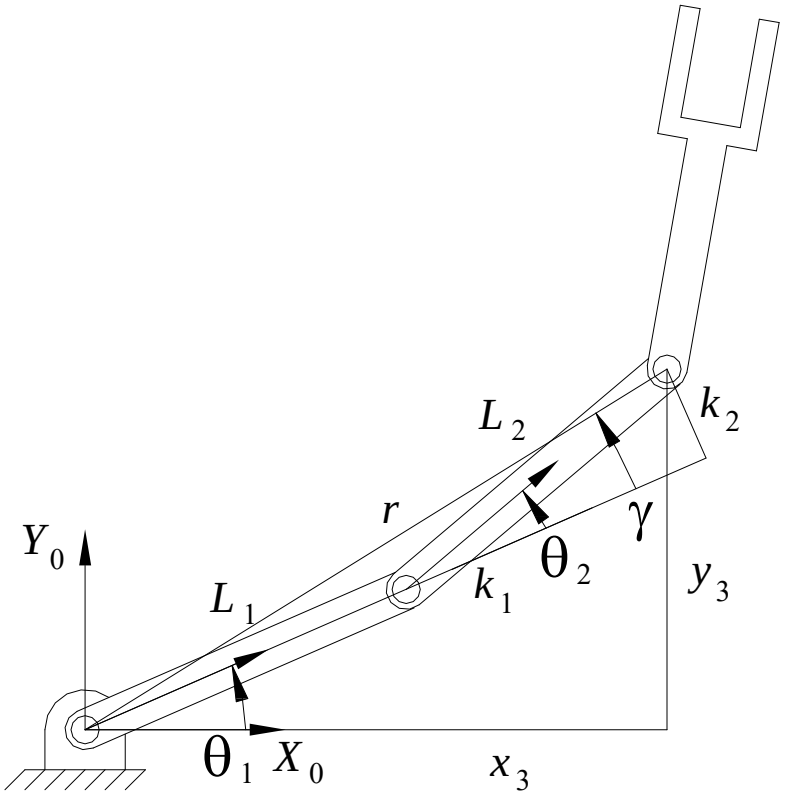
$$x_3 = r\cos\theta_1\cos\gamma - r\sin\theta_1\sin\gamma = r\cos(\theta_1+\gamma)$$
$$y_3 = r\sin\theta_1\cos\gamma + r\cos\theta_1\sin\gamma = r\sin(\theta_1+\gamma)$$

We can form a ratio of the *Y* to the *X* equation to solve for θ_1, one for each θ_2 value.

$$\frac{r\sin(\theta_1+\gamma)}{r\cos(\theta_1+\gamma)} = \frac{y_3}{x_3}$$

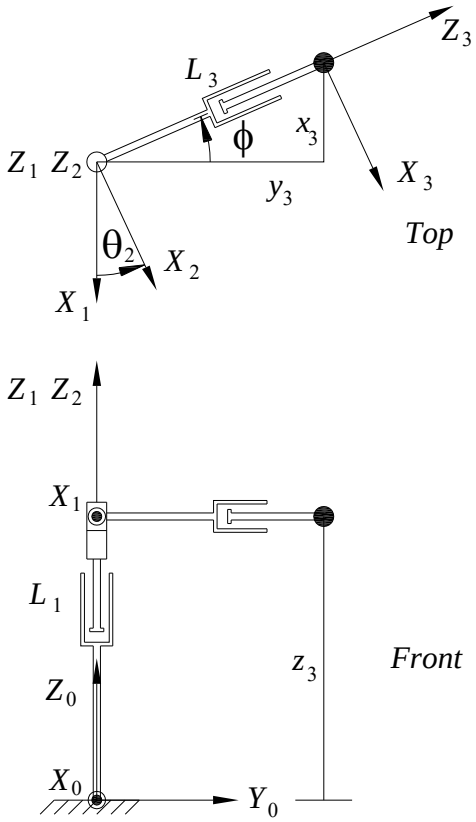
$$\theta_1 = \text{atan2}(y_3, x_3) - \gamma$$

The interesting geometric interpretation in the planar 3R IPK solution via this method is shown in the figure below. The entire k_1–k_2–r triangle shown above is rotated with θ_1 (and leads θ_1 for the elbow-down solution). r is the distance from the origin of fixed frame {0} to the origin of moving frame {3}.



This is why θ_1 was able to cancel out of the equations and leave only θ_2 for solving first in this solution method. The length r from the origin of {0} to the origin of {3} is only a function of the elbow joint angle θ_2 and does not depend on θ_1 or θ_3.

5.3 Spatial PRP Cylindrical Robot IPK Solution



Inverse Pose Kinematics Problem

Given the robot (the 9 constant DH Parameters) and $\left[{}^{0}_{3}T\right]$, or $\left\{{}^{0}X_{3}\right\}=\left\{x_3 \quad y_3 \quad z_3 \quad \phi\right\}$, Calculate the required joint values L_1, θ_2, L_3. The inverse pose equations come from the forward pose expressions:

$$\left[{}^{0}_{3}T\right]=\begin{bmatrix} r_{11} & 0 & r_{13} & p_x \\ r_{21} & 0 & r_{23} & p_y \\ 0 & -1 & 0 & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}=\begin{bmatrix} c\phi & 0 & -s\phi & x_3 \\ s\phi & 0 & c\phi & y_3 \\ 0 & -1 & 0 & z_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}=\begin{bmatrix} c\theta_2 & 0 & -s\theta_2 & -L_3 s\theta_2 \\ s\theta_2 & 0 & c\theta_2 & L_3 c\theta_2 \\ 0 & -1 & 0 & L_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

This robot has spatial translational motion, but its rotation is limited to the top-view plane. There is a subspace of the general 6-dof pose, represented by $\left[{}^{0}_{3}T\right]$ or by $\left\{{}^{0}X_{3}\right\}=\left\{x_3 \quad y_3 \quad z_3 \quad \phi\right\}^{T}$.

<u>We have a problem</u> – there are only 3 joints (***n* = 3**) but there are ***m* = 4** Cartesian values. This is an overconstrained problem and no solution exists in general. A dependency exists among $\left\{{}^{0}X_{3}\right\}=\left\{x_3 \quad y_3 \quad z_3 \quad \phi\right\}^{T}$ and thus all four cannot be commanded independently. Therefore, let us command only 3 Cartesian values $\left\{{}^{0}X_{3\text{RED}}\right\}=\left\{{}^{0}P_{3}\right\}=\left\{x_3 \quad y_3 \quad z_3\right\}^{T}$; we will treat this robot as a translational freedom robot. ϕ is not independent but is related to x_3 and y_3. The three equations to solve for the three unknowns are then taken only from the translational equations.

Reduced (translation only) Cylindrical Robot Inverse Pose Kinematics Solution, *m* = *n* = 3:

$$\begin{aligned} x_3 &= -L_3 s\theta_2 \\ y_3 &= \ L_3 c\theta_2 \\ z_3 &= \ L_1 \end{aligned} \qquad \Rightarrow \qquad \begin{aligned} L_1 &= z_3 \\ \theta_2 &= \text{atan2}(-x_3, y_3) \\ L_3 &= +\sqrt{x_3^2 + y_3^2} \end{aligned}$$

Mathematically, there are two solution sets. However, the $-L_3$ solution is not a practical choice, so we only show one solution set above. If $-L_3$ is allowed, then $\theta_2 + 180^\circ$ is the angle solution. This is not a general spatial manipulator, i.e. $\left\{ {}^0P_3 \right\}$ and $\left[{}^0_3R \right]$ cannot be specified independently.

Cylindrical Robot Inverse Pose Kinematics Examples

1) Given $\left\{ {}^0P_3 \right\}$, calculate L_1, θ_2, L_3.

$$\left\{ {}^0P_3 \right\} = \begin{Bmatrix} x_3 \\ y_3 \\ z_3 \end{Bmatrix} = \begin{Bmatrix} -1 \\ 1.732 \\ 3 \end{Bmatrix}$$

Answers (*m* and *deg*)

i	**branch**	L_1	θ_2	L_3
1	practical	3	30	2
2	impractical	3	210	–2

The first line is expected since this is the same position as the spatial PRP cylindrical robot Forward Pose Kinematics Example 1. Now we must check the second solution set to ensure it is correct (substitute the second, impractical, solution set into Forward Pose Kinematics and ensure the same $\left\{ {}^0P_3 \right\}$ is obtained). Also, for both cases you can calculate $\left[{}^0_3R \right]$ – they will be different (why?).

2) Given $\left\{ {}^0P_3 \right\}$, calculate L_1, θ_2, L_3.

$$\left\{ {}^0P_3 \right\} = \begin{Bmatrix} x_3 \\ y_3 \\ z_3 \end{Bmatrix} = \begin{Bmatrix} 1 \\ 0 \\ 2 \end{Bmatrix}$$

Answers (*m* and *deg*)

i	**branch**	L_1	θ_2	L_3
1	practical	2	–90	1
2	impractical	2	90	–1

As mentioned earlier there is only one practical solution set (1) for the cylindrical robot. Solution set (2), with the negative L_3, is given in the above examples only for completeness.

5.4 Pieper's Solution Method

Decoupled Inverse Pose Kinematics Solution

Pieper proved that if a 6-dof robot has any three consecutive joint axes intersecting, there exists a closed-form (analytical) solution to the inverse position kinematics. The majority of industrial robots are in this category.

In particular, many robots have a spherical wrist, i.e. three wrist actuators that rotate about a axes intersecting in a common point. In this case, the position and orientation sub-problems may be decoupled. We solve for the first three joints first using the position vector input. Then we solve for the second three joints next using the given orientation, based on the orientation caused by the first three joints. The given position vector must point to the wrist point (the shared origin point of three consecutive wrist frames). We can always transform any end-effector or other tool vector to this origin point, since there are no more active joints beyond the last wrist joint.

PUMA Example (without details):

Given $\left[{}^{0}_{6}T \right]$, calculate $(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)$.

$$\left[{}^{0}_{6}T \right] = f(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)$$

$$\left[{}^{0}_{6}T \right] = \begin{bmatrix} \left[{}^{0}_{6}R(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6) \right] & \left\{ {}^{0}P_6(\theta_1, \theta_2, \theta_3) \right\} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix}$$

Joints 4, 5, and 6 cannot affect the translation of the wrist origin point.

1) Translational equations: Given $\left\{ {}^{0}P_6 \right\}$, calculate $(\theta_1, \theta_2, \theta_3)$ values (4 sets).
3 independent equations, 3 unknowns.

2) Rotational equations: Given $\left[{}^{0}_{6}R \right]$, and knowing $(\theta_1, \theta_2, \theta_3)$, calculate $(\theta_4, \theta_5, \theta_6)$ values (2 sets).
3 independent equations, 6 dependent equations, 3 unknowns.

$$\left[{}^{3}_{6}R(\theta_4, \theta_5, \theta_6) \right] = \left[{}^{0}_{3}R(\theta_1, \theta_2, \theta_3) \right]^T \left[{}^{0}_{6}R \right]$$

4 sets of $(\theta_1, \theta_2, \theta_3)$; 2 sets of $(\theta_4, \theta_5, \theta_6)$ for each. Therefore, there are 8 overall solutions to the inverse position problem for the PUMA. Some solution sets may lie outside joint ranges. Generally one would choose the closest solution to the previous position.

We can use homogeneous transformation equations to isolate and solve for the various unknowns in turn. I call this 'peeling-off' homogeneous transformations matrices with unknowns to separate variables. The approach is to multiply by the inverse component homogeneous transformation matrices as a function of one unknown joint variable each. Below the details are not shown so you must try this for yourself. The original FPK transform equation is the starting point for IPK, as always.

$$\left[{}^{0}_{6}T \right] = \left[{}^{0}_{1}T(\theta_1) \right] \left[{}^{1}_{2}T(\theta_2) \right] \left[{}^{2}_{3}T(\theta_3) \right] \left[{}^{3}_{4}T(\theta_4) \right] \left[{}^{4}_{5}T(\theta_5) \right] \left[{}^{5}_{6}T(\theta_6) \right]$$

The left-hand-side $\left[{}^{0}_{6}T \right]$ is a set of valid given numbers, the input to the IPK problem. Inverting the first matrix on the left yields:

$$\left[{}^{0}_{1}T(\theta_1) \right]^{-1} \left[{}^{0}_{6}T \right] = \left[{}^{1}_{2}T(\theta_2) \right] \left[{}^{2}_{3}T(\theta_3) \right] \left[{}^{3}_{4}T(\theta_4) \right] \left[{}^{4}_{5}T(\theta_5) \right] \left[{}^{5}_{6}T(\theta_6) \right]$$

from which we can solve for θ_1 and θ_3. Now we invert the homogeneous transformation matrix that combines the first three matrices in the original equation.

$$\left[{}^{0}_{3}T(\theta_2) \right]^{-1} \left[{}^{0}_{6}T \right] = \left[{}^{3}_{4}T(\theta_4) \right] \left[{}^{4}_{5}T(\theta_5) \right] \left[{}^{5}_{6}T(\theta_6) \right]$$

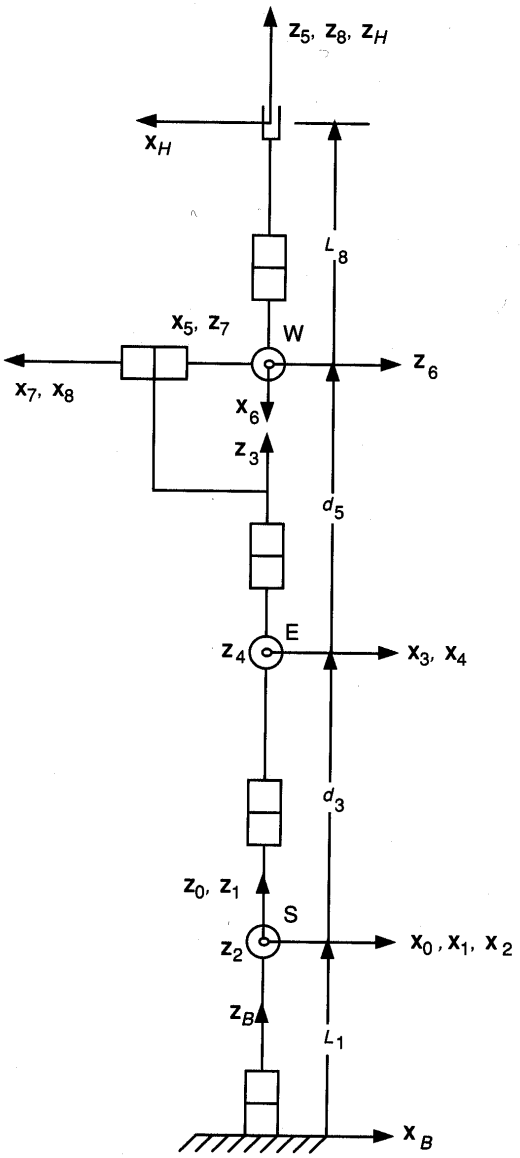
We can solve for θ_2 with θ_1 and θ_3 already known. This completes the solution for the arm angle joints $(\theta_1, \theta_2, \theta_3)$, possible because the position vector $\left\{ {}^{0}P_{6} \right\}$ is only a function of $(\theta_1, \theta_2, \theta_3)$, as stated earlier.

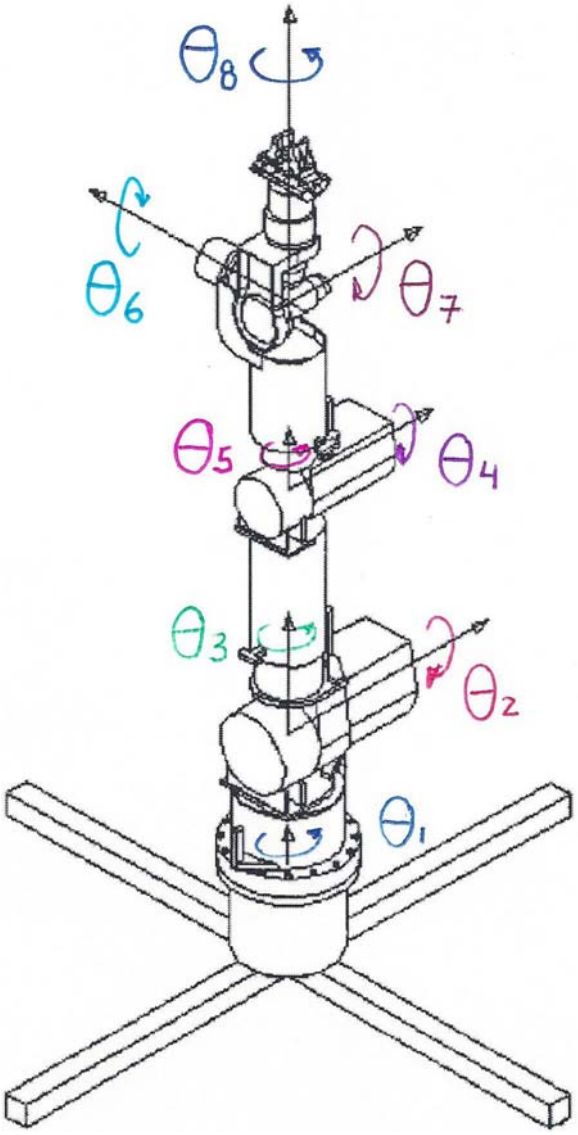
Now we must solve for the wrist joint angles $(\theta_4, \theta_5, \theta_6)$ knowing $(\theta_1, \theta_2, \theta_3)$. Performing one more matrix inversion on the left will separate these unknowns sufficiently.

$$\left[{}^{0}_{4}T(\theta_4) \right]^{-1} \left[{}^{0}_{6}T \right] = \left[{}^{4}_{5}T(\theta_5) \right] \left[{}^{5}_{6}T(\theta_6) \right]$$

No we can isolate and solve for θ_4, θ_5, and θ_6 in turn.

5.5 Spatial 8R NASA AAI ARMII Robot IPK Solution





Inverse Pose Kinematics General Statement

Given: Calculate:

Since m = 6 (Cartesian dof) and n = 8 (joint dof) we have the underconstrained kinematically-redundant case. There are infinite solutions (multiple as well). There are some great ways to make use of this redundancy for performance optimization in addition to following commanded Cartesian translational and rotational velocity trajectories. For inverse pose purposes we will here simplify instead and lock out the redundancy so that $m = n = 6$; let us choose $\theta_3 = \theta_5 = 0$ for all motion to accomplish this. Then we have a determined Inverse Pose Kinematics problem with finite solutions, still with multiple joint angle solution sets.

The Forward Pose Kinematics relationship is:

So, the first step should be to simplify the equations as much as possible by calculating the required $\left[{}^{0}_{8}T \right]$ to achieve the commanded $\left[{}^{B}_{H}T \right]$.

The problem can be decoupled between the arm joints 1-4 and the wrist joints 5-8 since the ARMII has a spherical wrist (all 4 wrist joint Cartesian coordinate frames share the same origin). See the previous section that explained the Pieper results for the 6-axis PUMA robot.

Now, we will further simplify by ignoring the wrist joints 6-8 (5 is already locked to zero) and solve the Inverse Pose Kinematics problem only for the arm joints 1,2, and 4. The full inverse solution is given in Williams[1].

Inverse Pose Kinematics Symbolic Solution for Arm Joints Only, with $\theta_3 = 0$

Reduced problem statement

Given $\left\{ {}^{B}P_5 \right\} = \left\{ x_5 \quad y_5 \quad z_5 \right\}^T$, calculate $(\theta_1, \theta_2, \theta_4)_i$, for all possible solution sets *i*.

That is, with only three active joints, we can only specify three Cartesian objectives, in this case the *XYZ* location of the origin of {5} with respect to origin of {*B*} (and expressed in the basis of {*B*}). Note that $\left\{ {}^{B}P_5 \right\} = \left\{ {}^{B}P_8 \right\}$ due to the spherical wrist.

The equations to solve for this problem come from the Forward Pose Kinematics relationships for the ARMII robot, the translational portion only (further, with $\theta_3 = 0$).

This equation yields (the derivation is left to student, use symbolic *Forward Pose Kinematics*):

$$\left\{ {}^{B}P_5 \right\} = \begin{Bmatrix} x_5 \\ y_5 \\ z_5 \end{Bmatrix} = \begin{Bmatrix} -c_1(d_3 s_2 + d_5 s_{24}) \\ -s_1(d_3 s_2 + d_5 s_{24}) \\ d_B + d_3 c_2 + d_5 c_{24} \end{Bmatrix} \qquad \text{where} \qquad \begin{aligned} c_{24} &= \cos(\theta_2 + \theta_4) \\ s_{24} &= \sin(\theta_2 + \theta_4) \end{aligned}$$

(Since $\theta_3 = 0$ always, the *Z* axes of 2 and 4 are always parallel and we used the sum-of-angles trig formulas.)

[1] R.L. Williams II, Kinematic Equations for Control of the Redundant Eight-Degree-of-Freedom Advanced Research Manipulator II, NASA Technical Memorandum 4377, NASA Langley Research Center, Hampton, VA, July 1992.

Solution Process

1. A ratio of the *Y* to *X* equations yields:

$\theta_1 + 180^\circ$ is also a valid solution

2. Since θ_1 is now known (two values), we can modify the *Y* and *Z* equations.

$$Y = \frac{-y_5}{s_1}$$

where:

$$Z = z_5 - d_B$$

Isolate the $(\theta_2 + \theta_4)$ terms:

Square and add to eliminate θ_4:

The result is one equation in one unknown θ_2.

$$E\cos\theta_2 + F\sin\theta_2 + G = 0$$

where

$$E = 2Zd_3$$
$$F = 2Yd_3$$
$$G = -Y^2 - Z^2 - d_3^2 + d_5^2$$

We can solve this equation for θ_2 by using the ***Tangent Half-Angle Substitution***. We presented this back in the Inverse Pose Solution of the planar 3R robot; we solve for θ_2 (in that section, it was for θ_1).

Solve for θ_2 by inverting the original Tangent Half-Angle definition.

Two θ_2 solutions result from the $\pm$ in the quadratic formula; both are correct (there are multiple solutions – elbow up and elbow down). To find θ_4, return to original (arranged) translational equations.

$$d_5 s_{24} = Y - d_3 s_2$$
$$d_5 c_{24} = Z - d_3 c_2$$

Find the unique θ_4 for each θ_2 value (use the quadrant-specific **`atan2`** function in MATLAB).

Solutions Summary

The solution is now complete for the ARMII robot reduced inverse pose problem (translational joints only, plus $\theta_3 = 0$).

There are multiple solutions since there are two values for θ_1. For each θ_1, there are two values for θ_2; for each valid (θ_1, θ_2), there is a unique θ_4. So there are a total of four $(\theta_1, \theta_2, \theta_4)$ solution sets for this reduced problem. We can show this with the PUMA model (it's not the same robot, but it has similar joints when $\theta_3 = 0$).

These four solution sets occur in a very special arrangement pattern, summarized in the table below.

i	θ_1	θ_2	θ_3	θ_4
1	θ_1	θ_{2_1}	0	θ_4
2	θ_1	θ_{2_2}	0	$-\theta_4$
3	$\theta_1 + 180^\circ$	$-\theta_{2_2}$	0	θ_4
4	$\theta_1 + 180^\circ$	$-\theta_{2_1}$	0	$-\theta_4$

In all numerical examples, you can check the results; plug all solution sets $(\theta_1, \theta_2, \theta_4)$ one at a time into the Forward Pose solution and verify that all sets yield the same, commanded $\left\{ {}^B P_5 \right\}$. You can also calculate the associated $\left[{}^B_4 R \right]$. All of these resulting rotation matrices should be different (why?).

8R ARMII Robot Translational Inverse Pose Kinematics Example

Given $\left\{ {}^{B}P_5 \right\}$, calculate $(\theta_1, \theta_2, \theta_4)_i \quad i = 1,2,3,4$.

$$\left\{ {}^{B}P_5 \right\} = \begin{Bmatrix} x_5 \\ y_5 \\ z_5 \end{Bmatrix} = \begin{Bmatrix} -0.6572 \\ -0.1159 \\ 1.6952 \end{Bmatrix}$$

Answers (*deg*)

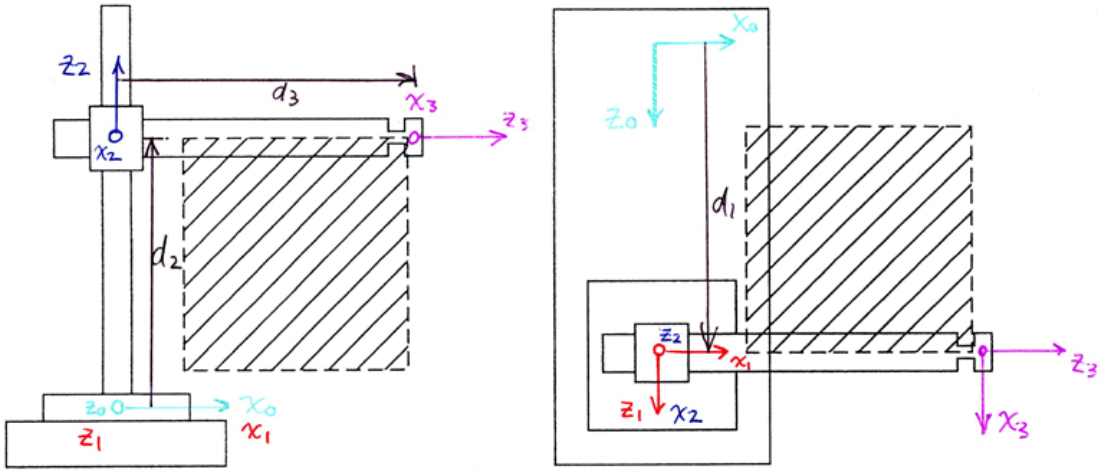
i	θ_1	θ_2	θ_3	θ_4
1	10	20	0	30
2	10	46.6	0	-30
3	190	-46.6	0	30
4	190	-20	0	-30

Check all solution sets via Forward Pose Kinematics to ensure all yield the correct, commanded $\left\{ {}^{B}P_5 \right\}$.

7. Velocity Kinematics

7.8 Velocity Kinematics Example

Spatial 3P Cartesian Manipulator Velocity Example



Side view Top view

a) Forward Velocity Kinematics

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

b) Inverse Velocity Kinematics (analytical result)

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

c) Singularity Analysis

$$ \left|{}^0J\right| = \left|{}^1J\right| = \left|{}^2J\right| = \left|{}^3J\right| = 1 $$

no possible singularities

d) Static Force Analysis

$$ \{f\} = \left[{}^0J\right]^T\{{}^0F\} \qquad \begin{Bmatrix}f_1\\ f_2\\ f_3\end{Bmatrix} = \begin{bmatrix}0 & 0 & 1\\ 0 & 1 & 0\\ 1 & 0 & 0\end{bmatrix}{}^0\begin{Bmatrix}F_x\\ F_y\\ F_z\end{Bmatrix} = {}^0\begin{Bmatrix}F_z\\ F_y\\ F_x\end{Bmatrix} $$

7.9 Jacobian Matrix Expressed in Another Frame

Here the Jacobian matrix still relates the end-effector frame velocity with respect to the base frame. But simpler analytical expressions are possible for the Jacobian matrix, by choosing an intermediate frame to express the coordinates of the velocity vectors (different basis of expression).

$$ {}^{k}\begin{Bmatrix}\{v\}\\ \{\omega\}\end{Bmatrix} = \begin{bmatrix}\left[{}^{k}_{0}R\right] & [0]\\ [0] & \left[{}^{k}_{0}R\right]\end{bmatrix} {}^{0}\begin{Bmatrix}\{v\}\\ \{\omega\}\end{Bmatrix} $$

$$ {}^{k}\begin{Bmatrix}\{v\}\\ \{\omega\}\end{Bmatrix} = \left[{}^{k}J\right]\left\{\dot{\Theta}\right\} \qquad\qquad {}^{0}\begin{Bmatrix}\{v\}\\ \{\omega\}\end{Bmatrix} = \left[{}^{0}J\right]\left\{\dot{\Theta}\right\} $$

$$ \left[{}^{k}J\right]\left\{\dot{\Theta}\right\} = \begin{bmatrix}\left[{}^{k}_{0}R\right] & [0]\\ [0] & \left[{}^{k}_{0}R\right]\end{bmatrix}\left[{}^{0}J\right]\left\{\dot{\Theta}\right\} $$

$\dot{\Theta}$ is not dependent on a frame (relative joint rates)

$$ \left[{}^{k}J\right] = \begin{bmatrix}\left[{}^{k}_{0}R\right] & [0]\\ [0] & \left[{}^{k}_{0}R\right]\end{bmatrix}\left[{}^{0}J\right] $$

Planar 3R Robot

The Jacobian was derived in frame {0} (here we go to {3} instead of {*H*}).

$$ \left[{}^{0}J\right] = \begin{bmatrix} -L_1 s_1 - L_2 s_{12} & -L_2 s_{12} & 0 \\ L_1 c_1 + L_2 c_{12} & L_2 c_{12} & 0 \\ 1 & 1 & 1 \end{bmatrix} $$

$$ \left[{}^{1}J\right] = \begin{bmatrix} -L_2 s_2 & -L_2 s_2 & 0 \\ L_1 + L_2 c_2 & L_2 c_2 & 0 \\ 1 & 1 & 1 \end{bmatrix} \qquad \left[{}^{1}_{0}R\right] = \begin{bmatrix} c_1 & s_1 & 0 \\ -s_1 & c_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} $$

$$ \left[{}^{2}J\right] = \begin{bmatrix} L_1 s_2 & 0 & 0 \\ L_1 c_2 + L_2 & L_2 & 0 \\ 1 & 1 & 1 \end{bmatrix} \qquad \left[{}^{2}_{0}R\right] = \begin{bmatrix} c_{12} & s_{12} & 0 \\ -s_{12} & c_{12} & 0 \\ 0 & 0 & 1 \end{bmatrix} $$

$$ \left[{}^{3}J\right] = \begin{bmatrix} L_1 s_{23} + L_2 s_3 & L_2 s_3 & 0 \\ L_1 c_{23} + L_2 c_3 & L_2 c_3 & 0 \\ 1 & 1 & 1 \end{bmatrix} \qquad \left[{}^{3}_{0}R\right] = \begin{bmatrix} c_{123} & s_{123} & 0 \\ -s_{123} & c_{123} & 0 \\ 0 & 0 & 1 \end{bmatrix} $$

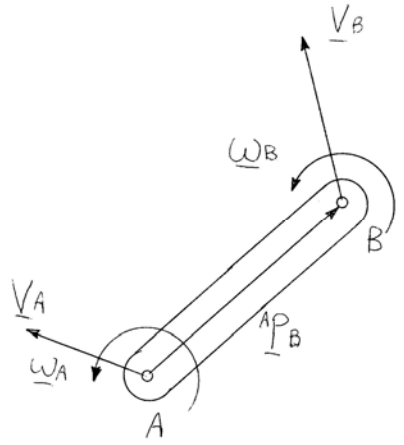
$\left[{}^{3}J\right]$ agrees with that derived in the third method above.

7.10 Cartesian Transformation of Velocities and Wrenches

Here we show how to move velocity and wrench vectors (both translational and rotational) from one point to another on a rotating rigid body. We can replace one vector with an equivalent vector acting at a different point. For example, to calculate the velocity (or wrench) at the wrist to produce a desired velocity (or wrench) at the hand.

Velocity Transformation

Here we find the equivalent velocity at $\{A\}$ corresponding to a given desired motion of $\{B\}$. For instance, $\{B\}$ could be the hand frame $\{H\}$ where we want motion and $\{A\}$ would then be the last wrist frame $\{n\}$ for which the Jacobian matrix is derived.



Basic equations

$$\{v_B\} = \{v_A\} + \{\omega_A\} \times \left\{ {}^A P_B \right\} \qquad \{\omega_B\} = \{\omega_A\}$$

reversing

$$\{v_A\} = \{v_B\} - \{\omega_B\} \times \left\{ {}^A P_B \right\} = \{v_B\} + \left\{ {}^A P_B \right\} \times \{\omega_B\} \qquad \{\omega_A\} = \{\omega_B\}$$

All vectors must be expressed in same frame, choose $\{A\}$.

$$ {}^A \begin{Bmatrix} \{ {}^0 V_A \} \\ \{ {}^0 \omega_A \} \end{Bmatrix} = \begin{bmatrix} \left[{}^A_B R \right] & \left[{}^A P_B \times \right] \left[{}^A_B R \right] \\ [0] & \left[{}^A_B R \right] \end{bmatrix} {}^B \begin{Bmatrix} \{ {}^0 V_B \} \\ \{ {}^0 \omega_B \} \end{Bmatrix}$$

$$ {}^A \{ \dot{X} \} = \left[{}^A_B T_V \right] {}^B \{ \dot{X} \}$$

$$ \left[{}^A_B T_V \right] = \begin{bmatrix} \left[{}^A_B R \right] & \left[{}^A P_B \times \right] \left[{}^A_B R \right] \\ [0] & \left[{}^A_B R \right] \end{bmatrix}$$

Where $$ \left[{}^A P_B \times \right] = \begin{bmatrix} 0 & -p_z & p_y \\ p_z & 0 & -p_x \\ -p_y & p_x & 0 \end{bmatrix} $$ is the cross product matrix.

Velocity transformation example

Find the equivalent velocity at *A* corresponding to a given motion of *B*.

Given: $${}^{B}\left\{{}^{0}V_{B}\right\}=\begin{Bmatrix}1\\1\\0\end{Bmatrix} \qquad {}^{B}\left\{{}^{0}\omega_{B}\right\}=\begin{Bmatrix}0\\0\\2\end{Bmatrix}$$

$$\left\{{}^{A}P_{B}\right\}=\begin{Bmatrix}2.6\\1.5\\0\end{Bmatrix} \qquad \left[{}^{A}_{B}R\right]=\begin{bmatrix}0.866 & -0.5 & 0\\0.5 & 0.866 & 0\\0 & 0 & 1\end{bmatrix}$$

$$\left[{}^{A}P_{B}\times\right]\left[{}^{A}_{B}R\right]=\begin{bmatrix}0 & 0 & 1.5\\0 & 0 & -2.6\\-1.5 & 2.6 & 0\end{bmatrix}\begin{bmatrix}0.866 & -0.5 & 0\\0.5 & 0.866 & 0\\0 & 0 & 1\end{bmatrix}=\begin{bmatrix}0 & 0 & 1.5\\0 & 0 & -2.6\\0 & 3 & 0\end{bmatrix}$$

$${}^{A}\left\{\dot{X}\right\}=\left[{}^{A}_{B}T_{V}\right]{}^{B}\left\{\dot{X}\right\}$$

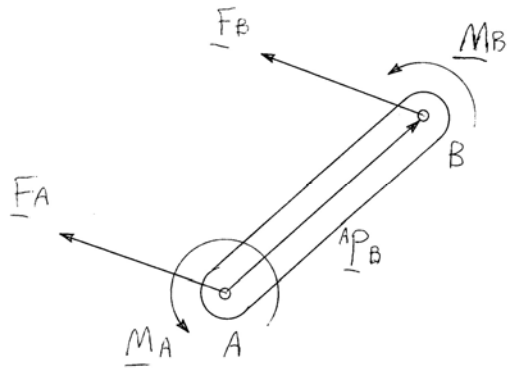
$${}^{A}\begin{Bmatrix}\left\{{}^{0}V_{A}\right\}\\\left\{{}^{0}\omega_{A}\right\}\end{Bmatrix}=\begin{bmatrix}0.866 & -0.5 & 0 & 0 & 0 & 1.5\\0.5 & 0.866 & 0 & 0 & 0 & -2.6\\0 & 0 & 1 & 0 & 3 & 0\\0 & 0 & 0 & 0.866 & -0.5 & 0\\0 & 0 & 0 & 0.5 & 0.866 & 0\\0 & 0 & 0 & 0 & 0 & 1\end{bmatrix}\begin{Bmatrix}1\\1\\0\\0\\0\\2\end{Bmatrix}=\begin{Bmatrix}3.366\\-3.834\\0\\0\\0\\2\end{Bmatrix}$$

Check $$\left\{\omega_{A}\right\}=\left\{\omega_{B}\right\} \qquad \left\{V_{A}\right\}=\left\{V_{B}\right\}-\left\{\omega\right\}\times\left\{r\right\}$$

$${}^{B}\left\{{}^{0}V_{A}\right\}=\begin{Bmatrix}1\\1\\0\end{Bmatrix}-\begin{vmatrix}\hat{i} & \hat{j} & \hat{k}\\0 & 0 & 2\\3 & 0 & 0\end{vmatrix}=\begin{Bmatrix}1\\1\\0\end{Bmatrix}-\begin{Bmatrix}0\\6\\0\end{Bmatrix}=\begin{Bmatrix}1\\-5\\0\end{Bmatrix}$$

Pseudostatic Wrench Transformation

Here we find the equivalent wrench at $\{A\}$ corresponding to a given wrench at $\{B\}$. For instance, $\{B\}$ could be the hand frame $\{H\}$ where we want the wrench to be exerted and $\{A\}$ would then be the last wrist frame $\{n\}$ for which the Jacobian matrix is derived.



Basic equations

$$\{F_A\} = \{F_B\}$$
$$\{M_A\} = \{M_B\} + \{{}^A P_B\} \times \{F_B\}$$

more properly, $\{{}^k F_B\}$

All vectors must be expressed in the same frame, choose $\{A\}$.

$$\begin{Bmatrix} \{F_A\} \\ \{M_A\} \end{Bmatrix} = \begin{bmatrix} \begin{bmatrix} {}^A_B R \end{bmatrix} & [0] \\ \begin{bmatrix} {}^A P_B \times \end{bmatrix} \begin{bmatrix} {}^A_B R \end{bmatrix} & \begin{bmatrix} {}^A_B R \end{bmatrix} \end{bmatrix} \begin{Bmatrix} \{F_B\} \\ \{M_B\} \end{Bmatrix}$$

$$\{W\} = \begin{bmatrix} {}^A_B T_W \end{bmatrix} \{W\}$$

$$\begin{bmatrix} {}^A_B T_W \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} {}^A_B R \end{bmatrix} & [0] \\ \begin{bmatrix} {}^A P_B \times \end{bmatrix} \begin{bmatrix} {}^A_B R \end{bmatrix} & \begin{bmatrix} {}^A_B R \end{bmatrix} \end{bmatrix}$$

Note $\begin{bmatrix} {}^A_B T_W \end{bmatrix}$ is a block-transpose of $\begin{bmatrix} {}^A_B T_V \end{bmatrix}$; i.e. $\begin{bmatrix} {}^A P_B \times \end{bmatrix} \begin{bmatrix} {}^A_B R \end{bmatrix}$ is switched with [0].

There is a duality here. In the velocity transformation, the rotational term is unchanged, while in the wrench transformation, the translational term is unchanged.

Wrench transformation example

Find the equivalent wrench at A corresponding to a given wrench at B.

Given: $${}^{B}\{F_B\} = \begin{Bmatrix} 1 \\ 1 \\ 0 \end{Bmatrix} \qquad {}^{B}\{M_B\} = \begin{Bmatrix} 0 \\ 0 \\ 0 \end{Bmatrix}$$

$\left\{ {}^{A}P_B \right\}, \left[{}^{A}_{B}R \right], \left[{}^{A}P_B \times \right] \left[{}^{A}_{B}R \right]$ are the same as in the velocity example above.

$$ {}^{A}\{W\} = \left[{}^{A}_{B}T_W \right] {}^{B}\{W\}$$

$$ {}^{A}\begin{Bmatrix} \{F_A\} \\ \{M_A\} \end{Bmatrix} = \begin{bmatrix} 0.866 & -0.5 & 0 & 0 & 0 & 0 \\ 0.5 & 0.866 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1.5 & 0.866 & -0.5 & 0 \\ 0 & 0 & -2.6 & 0.5 & 0.866 & 0 \\ 0 & 3 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix} = \begin{Bmatrix} 0.366 \\ 1.366 \\ 0 \\ 0 \\ 0 \\ 3 \end{Bmatrix}$$

Check $$\{F_A\} = \{F_B\} \qquad \{M_A\} = \{r\} \times \{F_B\}$$

The derivations and examples for both velocity and wrench transformations were for the inverse case, i.e. given the end vector values $\{B\}$, calculate the vector values on the same rigid link, but inward toward $\{A\}$.

We could easily adapt the derivations and transformations to perform the forward calculation, i.e. given the inward vector values $\{A\}$, calculate the end vector values on the same rigid link, outward towards $\{B\}$.

8. Kinematically-Redundant Robots (KRRs)

8.2 Inverse Velocity (Resolved-Rate) Solution

8.2.1 Pseudoinverse-based

Alternate Particular Solution – satisfies primary task (satisfies Cartesian trajectory)

Based on Minimum Manipulator Kinetic Energy $f = \frac{1}{2}\left\{\dot{\Theta}\right\}^T [W] \left\{\dot{\Theta}\right\}$

$[W] = \left[M(\Theta)\right]$; The weighting matrix is the manipulator inertia tensor from robot dynamics

$$\left\{\dot{\Theta}_P\right\} = [J]^* \left\{\dot{X}\right\}$$

where $$[J]^* = [M]^{-1}[J]^T \left([J][M]^{-1}[J]^T\right)^{-1}$$

This Moore-Penrose pseudoinverse form is subject to singularities. Singular Value Decomposition (SVD) would ameliorate this problem.

Homogeneous Solution

For optimization, choose $z = \left\{\nabla H(\Theta)\right\}$, where $\left\{H(\Theta)\right\}$ is an objective function of joint angles to be minimized or maximized. Use $k_H > 0$ for maximization and $k_H < 0$ for minimization.

Joint Limit Avoidance $$H_J(\Theta) = \sum_{i=1}^{n} \left(\frac{\theta_i - \theta_{ci}}{\Delta\theta_i} \right)^2$$

Singularity Avoidance (Manipulability Maximization) $H_M(\Theta) = \sqrt{\left|[J][J]^T\right|}$

8.2.3 Klein and Huang's Algorithm

This methods yields the same results as the pseudoinverse with gradient projection into the null-space, but it is more efficient (less computations). Klein and Huang's algorithm accomplishes the particular and homogeneous solutions at the same time.

$$\left(\left[J\right]\left[J\right]^T\right)\left\{w\right\} = \left\{\dot{X}\right\} - k\left[J\right]\left\{\nabla H\left(\Theta\right)\right\};$$

solve $\{w\}$ using Gaussian elimination, then:

$$\left\{\dot{\Theta}\right\} = \left[J\right]^T \left\{w\right\} + k\left\{\nabla H\left(\Theta\right)\right\}.$$

So much of the existing kinematically-redundant robot literature is dedicated to more efficient redundancy resolution, but I think with today's processors, this is no longer a problem.

8.2.4 Singular Value Decomposition

Singular Value Decomposition (SVD) yields the same results as the pseudoinverse with gradient projection into the null-space, but with singularity robustness. If [*J*] is less than full rank, the solution cannot track arbitrary Cartesian trajectories, but SVD results are bounded so the motion can drive through singularities, as opposed to yielding infinite joint rates at singularities.

$$[J] = [U][W][V]^T$$

[*J*]	*m* x *n*	Jacobian matrix
[*U*]	*m* x *n*	column orthogonal matrix
[*W*]	*n* x *n*	positive semi-definite diagonal matrix
[*V*]	*n* x *n*	orthogonal matrix

$$[J]^* = [V]\left[diag\left(\frac{1}{w_j} \right) \right][U]^T$$

w_j are the singular values of [*J*]. For underconstrained systems of equations, there will always be $n - m$ zero singular values, where $n - m$ is the degree of redundancy. Any additional zero singular values correspond to degeneracies in *J*. In the above expression, $\frac{1}{w_j}$ is set to zero for $w_j = 0$ (ain't math fun?!?)

Both [*U*] and [*V*] are column-orthonormal (ignoring the last $n - m$ columns of [*U*]). Matrix [*V*] is also row-orthonormal, i.e.:

$$[V][V]^T = [V]^T[V] = [I_n]$$

$$[U][U]^T = [I_m]; \qquad \text{however, } [U]^T[U] \neq [I_n] \text{ (see the SVD example)}$$

Columns of *U* corresponding to nonzero w_j are an orthonormal basis which spans the range of [*J*]. Columns of *V* corresponding to zero w_j are an orthonormal basis for the null-space of [*J*].

Singular Value Decomposition (SVD) Example

$$[J] = \begin{bmatrix} -1.366 & -0.500 & -0.500 \\ 2.366 & 1.866 & 0.866 \end{bmatrix}$$

$$[J] = [U][W][V]^T$$

$$[J] = \begin{bmatrix} -0.430 & 0.903 & 0 \\ 0.903 & 0.430 & 0 \end{bmatrix} \begin{bmatrix} 3.467 & 0 & 0 \\ 0 & 0.420 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0.786 & -0.514 & -0.344 \\ 0.548 & 0.836 & 0 \\ 0.288 & -0.188 & 0.939 \end{bmatrix}^T$$

$$[J]^* = [V]\left[diag\left(\frac{1}{w_j} \right) \right][U]^T$$

$$[J]^* = \begin{bmatrix} 0.786 & -0.514 & -0.344 \\ 0.548 & 0.836 & 0 \\ 0.288 & -0.188 & 0.939 \end{bmatrix} \begin{bmatrix} 0.288 & 0 & 0 \\ 0 & 2.381 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} -0.430 & 0.903 \\ 0.903 & 0.430 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} -1.205 & -0.323 \\ 1.732 & 1.000 \\ -0.441 & -0.118 \end{bmatrix}$$

In the above example, the third singular value is zero, so $\frac{1}{w_j}$ is set to zero ($n - m$ singular values will always be zero). This result agrees with $[J]^*$ from the Moore-Penrose pseudoinverse formula given above.

Note:

$$[U][U]^T = [I_2] \qquad [U]^T[U] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \neq I_3 \qquad [V][V]^T = [V]^T[V] = [I_3]$$

8.2.5 Generalized Inverses

A generalized inverse of a matrix gives an answer to the linear problem even when a true matrix inverse does not exist (underconstrained, overconstrained, or row-rank-deficient). Mathematically, [*G*] is a generalized inverse of matrix [*A*] if

$$[A][G][A] = [G]$$

The Moore-Penrose pseudoinverse is just one possible generalized inverse of a matrix. It is the one applied most often to redundancy resolution of manipulators. In addition to the above relationship, the following relationships hold for the Moore-Penrose pseudoinverse.

$$[G][A][G] = [A]$$

$$([G][A])^* = [G][A]$$

$$([A][G])^* = [A][G]$$

where $(\,)^*$ indicates the complex conjugate transpose.

9. Parallel Robots

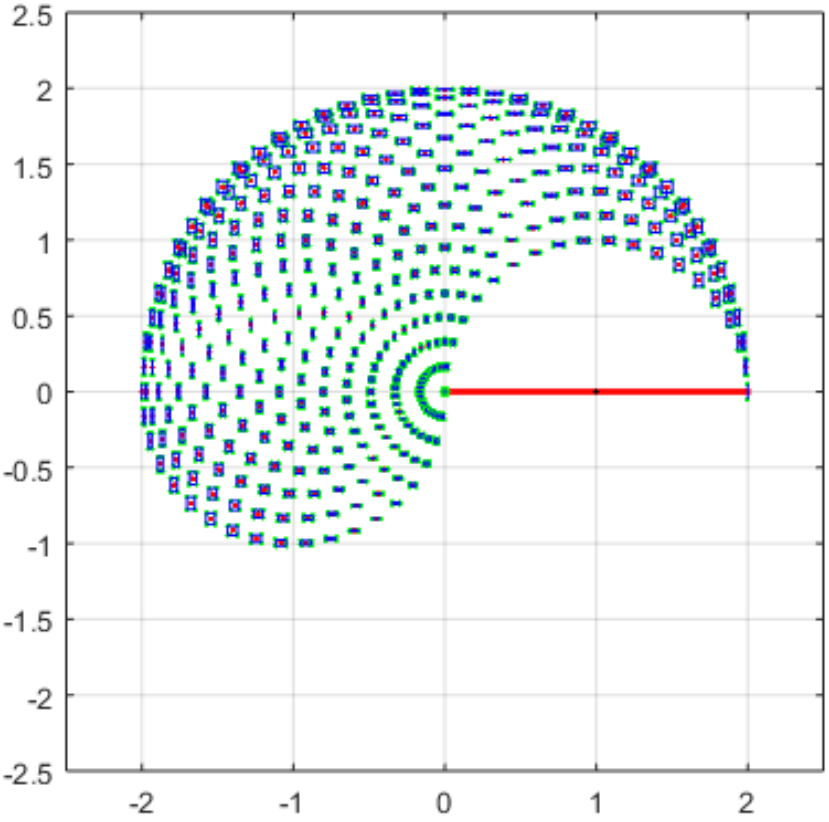
9.1 Introduction

Serial vs. Parallel Cartesian Sensitivity

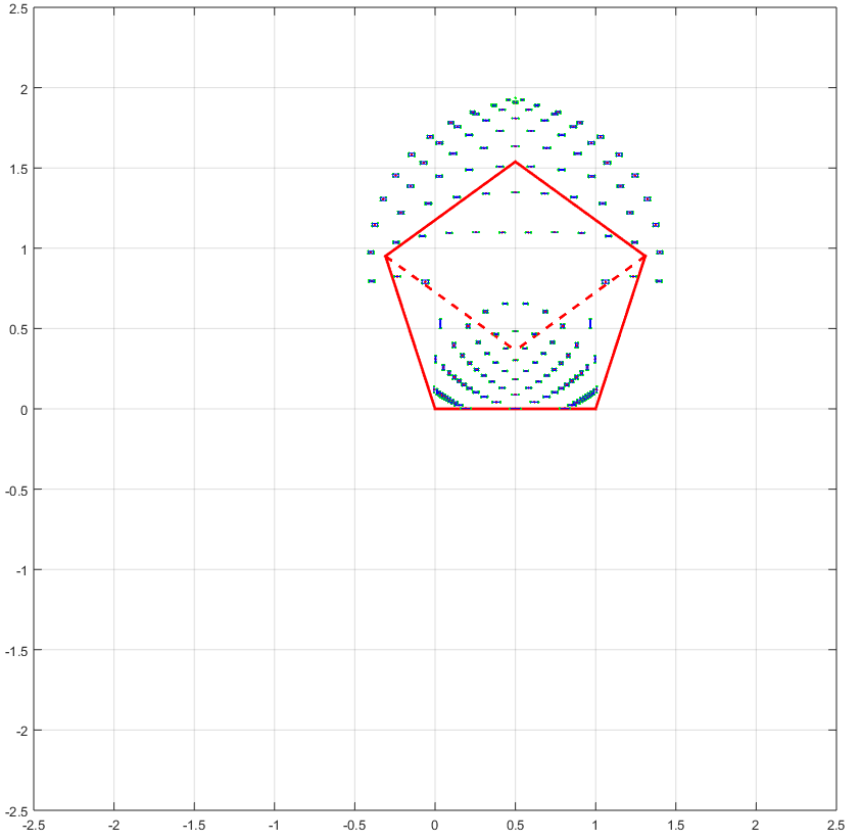
A planar 2-dof 2R serial robot is compared with a planar 2-dof 5R five-bar parallel robot of equal link lengths, with regards to Cartesian sensitivity. i.e. assuming a joint sensing uncertainty of $\pm 1^\circ$ for all active and passive joints, the following plots show the Cartesian uncertainty rectangles, but using Forward Pose Kinematics (FPK) in each case.

The following two plots show that the workspace for the serial robot is much larger than that of the parallel robot, when equal link lengths is the standard. However, these plots also show that the Cartesian sensitivity is much less for the parallel robot than the serial robot.

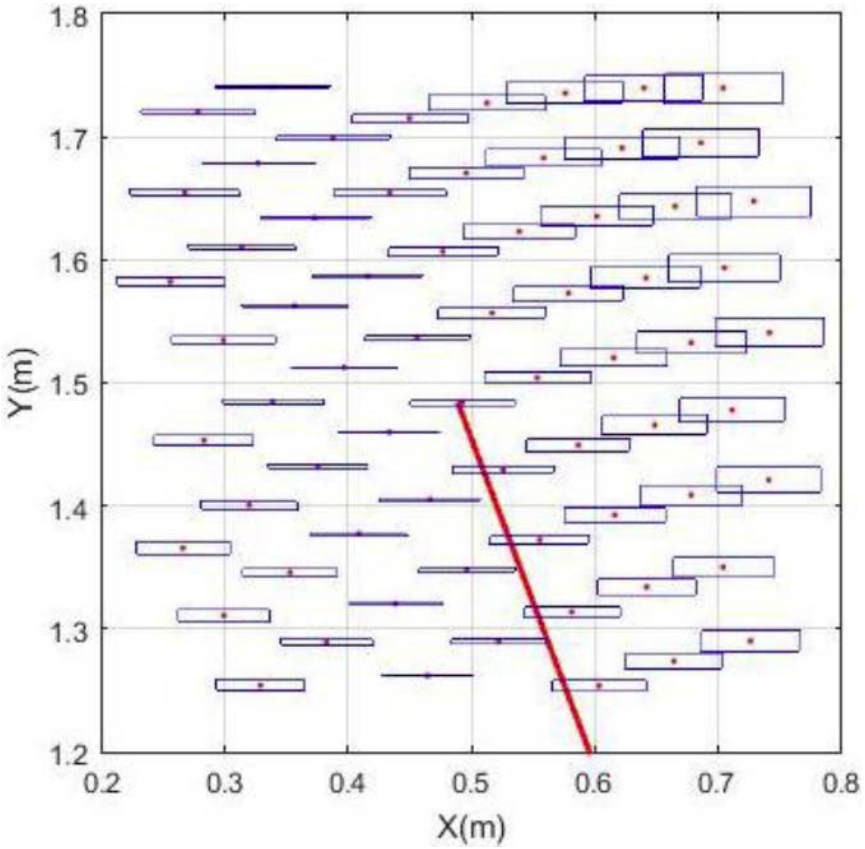
For the same two robots, the next pair of plots show the same Cartesian sensitivity information, zoomed into the same workspace subset. These plots show even more clearly the large advantage of the parallel robot over the serial robot regarding Cartesian sensitivity due to $\pm 1^\circ$ joint sensing uncertainty in each case.



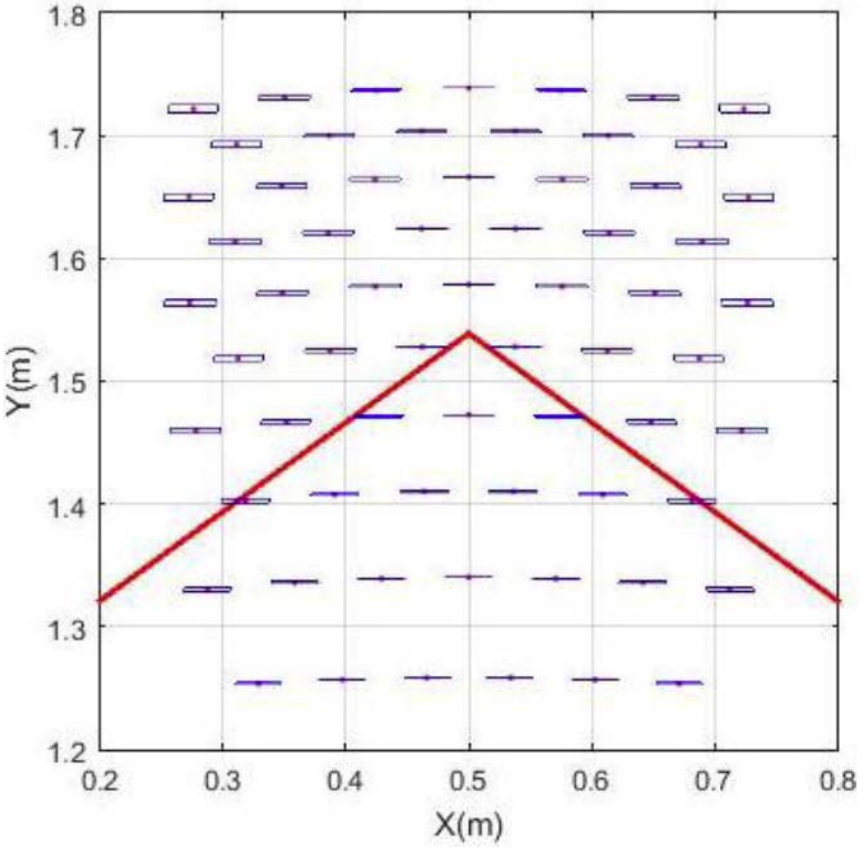
Serial 2R Robot Workspace and Cartesian Sensitivity



Parallel 5R Robot Workspace and Cartesian Sensitivity



Serial 2R Robot Workspace and Cartesian Sensitivity, subset



Parallel 5R Robot Workspace and Cartesian Sensitivity, subset

9.2 Planar 2-dof Five-Bar Parallel Robot

9.2.4 Acceleration Kinematics

Step 1. The five-bar robot **Position** and **Velocity Analyses** must first be complete.

Step 2. Identify the five-bar robot acceleration parameters.

$\underline{\ddot{\theta}}_i = \underline{\alpha}_i$ (i = 2,3,4,5) is the absolute angular acceleration of link i. $\underline{\ddot{\theta}}_1 = \underline{\alpha}_1 = \underline{0}$ since link 1, the fixed ground link, cannot rotate.

The velocity equations developed in the Planar 2-dof Five-Bar Parallel Robot Velocity Kinematics section are the starting point for deriving the acceleration equations.

Forward Acceleration Kinematics

Given r_1, θ_1, r_2, r_3, r_4, r_5; angles θ_2, θ_3, θ_4, and θ_5; angular rates $\dot{\theta}_2, \dot{\theta}_3, \dot{\theta}_4, \dot{\theta}_5$; plus actuator angular accelerations $\ddot{\theta}_2, \ddot{\theta}_5$

Find desired end-effector translational acceleration $\underline{\ddot{B}} = \begin{Bmatrix} \ddot{x} & \ddot{y} \end{Bmatrix}^T$, plus passive angular accelerations $\ddot{\theta}_3, \ddot{\theta}_4$

The acceleration equations come from the first time derivative of the velocity equations as shown below:

$$\begin{aligned} -r_2\dot{\theta}_2 s_2 - r_3\dot{\theta}_3 s_3 &= -r_5\dot{\theta}_5 s_5 - r_4\dot{\theta}_4 s_4 \\ r_2\dot{\theta}_2 c_2 + r_3\dot{\theta}_3 c_3 &= r_5\dot{\theta}_5 c_5 + r_4\dot{\theta}_4 c_4 \end{aligned}$$

$$\begin{aligned} -r_2\ddot{\theta}_2 s_2 - r_2\dot{\theta}_2^2 c_2 - r_3\ddot{\theta}_3 s_3 - r_3\dot{\theta}_3^2 c_3 &= -r_5\ddot{\theta}_5 s_5 - r_5\dot{\theta}_5^2 c_5 - r_4\ddot{\theta}_4 s_4 - r_4\dot{\theta}_4^2 c_4 \\ r_2\ddot{\theta}_2 c_2 - r_2\dot{\theta}_2^2 s_2 + r_3\ddot{\theta}_3 c_3 - r_3\dot{\theta}_3^2 s_3 &= r_5\ddot{\theta}_5 c_5 - r_5\dot{\theta}_5^2 s_5 + r_4\ddot{\theta}_4 c_4 - r_4\dot{\theta}_4^2 s_4 \end{aligned}$$

Since all links are rigid (i.e. no links are changing lengths), all $\dot{r}_i = 0$ and $\ddot{r}_i = 0$, thus the eight XY pairs of terms above represent the absolute tangential accelerations and absolute centripetal accelerations at 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 two unknowns $\ddot{\theta}_3$ and $\ddot{\theta}_4$:

$$\begin{bmatrix} a & b \\ d & e \end{bmatrix} \begin{Bmatrix} \ddot{\theta}_3 \\ \ddot{\theta}_4 \end{Bmatrix} = \begin{Bmatrix} C \\ F \end{Bmatrix}$$

$$\begin{aligned} a &= r_3 s_3 \\ b &= -r_4 s_4 \\ C &= -r_2 \ddot{\theta}_2 s_2 - r_2 \dot{\theta}_2^2 c_2 - r_3 \dot{\theta}_3^2 c_3 + r_5 \ddot{\theta}_5 s_5 + r_5 \dot{\theta}_5^2 c_5 + r_4 \dot{\theta}_4^2 c_4 \\ d &= -r_3 c_3 \\ e &= r_4 c_4 \\ F &= r_2 \ddot{\theta}_2 c_2 - r_2 \dot{\theta}_2^2 s_2 - r_3 \dot{\theta}_3^2 s_3 - r_5 \ddot{\theta}_5 c_5 + r_5 \dot{\theta}_5^2 s_5 + r_4 \dot{\theta}_4^2 s_4 \end{aligned}$$

The solution of the above matrix-vector set of equations is:

$$\ddot{\theta}_3 = \frac{Ce - bF}{ae - bd} \qquad\qquad \ddot{\theta}_4 = \frac{aF - Cd}{ae - bd}$$

With $\ddot{\theta}_3$ and $\ddot{\theta}_4$ now known, the end-effector translational acceleration $\underline{\ddot{B}} = \{\ddot{x} \quad \ddot{y}\}^T$ can be found either from the link2/link3 dyad or the link5/link4 dyad; both yield identical results.

$$\underline{\ddot{B}} = \begin{Bmatrix} \ddot{x} \\ \ddot{y} \end{Bmatrix} = \begin{Bmatrix} -r_2 \ddot{\theta}_2 s_2 - r_2 \dot{\theta}_2^2 c_2 - r_3 \ddot{\theta}_3 s_3 - r_3 \dot{\theta}_3^2 c_3 \\ r_2 \ddot{\theta}_2 c_2 - r_2 \dot{\theta}_2^2 s_2 + r_3 \ddot{\theta}_3 c_3 - r_3 \dot{\theta}_3^2 s_3 \end{Bmatrix} \qquad \underline{\ddot{B}} = \begin{Bmatrix} \ddot{x} \\ \ddot{y} \end{Bmatrix} = \begin{Bmatrix} -r_5 \ddot{\theta}_5 s_5 - r_5 \dot{\theta}_5^2 c_5 - r_4 \ddot{\theta}_4 s_4 - r_4 \dot{\theta}_4^2 c_4 \\ r_5 \ddot{\theta}_5 c_5 - r_5 \dot{\theta}_5^2 s_5 + r_4 \ddot{\theta}_4 c_4 - r_4 \dot{\theta}_4^2 s_4 \end{Bmatrix}$$

Planar Five-Bar Parallel Robot Forward Acceleration singularity condition

When does the above forward acceleration solution fail? The forward acceleration coefficient matrix is identical to the forward velocity coefficient matrix, which means the singularity conditions are identical (i.e. when links 3 and 4 are collinear, corresponding to the branch boundary between the two Forward Pose Kinematics solutions).

Inverse Acceleration Kinematics

Given r_1, θ_1, r_2, r_3, r_4, r_5; angles θ_2, θ_3, θ_4, and θ_5; angular rates $\dot{\theta}_2, \dot{\theta}_3, \dot{\theta}_4, \dot{\theta}_5$; plus desired end-effector translational acceleration $\underline{\ddot{B}} = \{\ddot{x} \quad \ddot{y}\}^T$

Find required actuator angular accelerations $\ddot{\theta}_2, \ddot{\theta}_5$, plus passive angular accelerations $\ddot{\theta}_3, \ddot{\theta}_4$

The actuator unknowns $\ddot{\theta}_2, \ddot{\theta}_5$ can be found independently from the two vector loop-closure equations presented at the end of the Planar 2-dof Five-Bar Parallel Robot Forward Acceleration Kinematics section. Passive unknown angular acceleration $\ddot{\theta}_3$ can be found along with $\ddot{\theta}_2$, and passive unknown angular acceleration $\ddot{\theta}_4$ can be found along with $\ddot{\theta}_5$. The two independent equations for end-effector acceleration $\underline{\ddot{B}} = \{\ddot{x} \quad \ddot{y}\}^T$ are repeated below, written in matrix-vector form:

$$\begin{bmatrix} -r_2 s_2 & -r_3 s_3 \\ r_2 c_2 & r_3 c_3 \end{bmatrix} \begin{Bmatrix} \ddot{\theta}_2 \\ \ddot{\theta}_3 \end{Bmatrix} = \begin{Bmatrix} \ddot{x} + r_2 \dot{\theta}_2^2 c_2 + r_3 \dot{\theta}_3^2 c_3 \\ \ddot{y} + r_2 \dot{\theta}_2^2 s_2 + r_3 \dot{\theta}_3^2 s_3 \end{Bmatrix} \qquad \begin{bmatrix} -r_4 s_4 & -r_5 s_5 \\ r_4 c_4 & r_5 c_5 \end{bmatrix} \begin{Bmatrix} \ddot{\theta}_4 \\ \ddot{\theta}_5 \end{Bmatrix} = \begin{Bmatrix} \ddot{x} + r_5 \dot{\theta}_5^2 c_5 + r_4 \dot{\theta}_4^2 c_4 \\ \ddot{y} + r_5 \dot{\theta}_5^2 s_5 + r_4 \dot{\theta}_4^2 s_4 \end{Bmatrix}$$

And the inverse acceleration solutions are:

$$\ddot{\theta}_2 = \frac{\ddot{x}c_3 + \ddot{y}s_3 + r_2\dot{\theta}_2^2 \cos(\theta_3 - \theta_2) + r_3\dot{\theta}_3^2}{r_2 \sin(\theta_3 - \theta_2)} \qquad \ddot{\theta}_4 = \frac{\ddot{x}c_5 + \ddot{y}s_5 + r_4\dot{\theta}_4^2 \cos(\theta_5 - \theta_4) + r_5\dot{\theta}_5^2}{r_4 \sin(\theta_5 - \theta_4)}$$

$$\ddot{\theta}_3 = \frac{-\ddot{x}c_2 - \ddot{y}s_2 - r_3\dot{\theta}_3^2 \cos(\theta_3 - \theta_2) - r_2\dot{\theta}_2^2}{r_3 \sin(\theta_3 - \theta_2)} \qquad \ddot{\theta}_5 = \frac{-\ddot{x}c_4 - \ddot{y}s_4 - r_5\dot{\theta}_5^2 \cos(\theta_5 - \theta_4) - r_4\dot{\theta}_4^2}{r_5 \sin(\theta_5 - \theta_4)}$$

Planar Five-Bar Parallel Robot Inverse Acceleration singularity condition

When does the above inverse acceleration solution fail? The inverse acceleration coefficient matrices are identical to the inverse velocity coefficient matrices, which means the singularity conditions are identical (i.e. when links 2 and 3 are collinear or when links 4 and 5 are collinear, corresponding to the branch boundaries between the two Inverse Pose Kinematics solution branches for the link2/link3 dyad and also for the link5/link4 dyad).

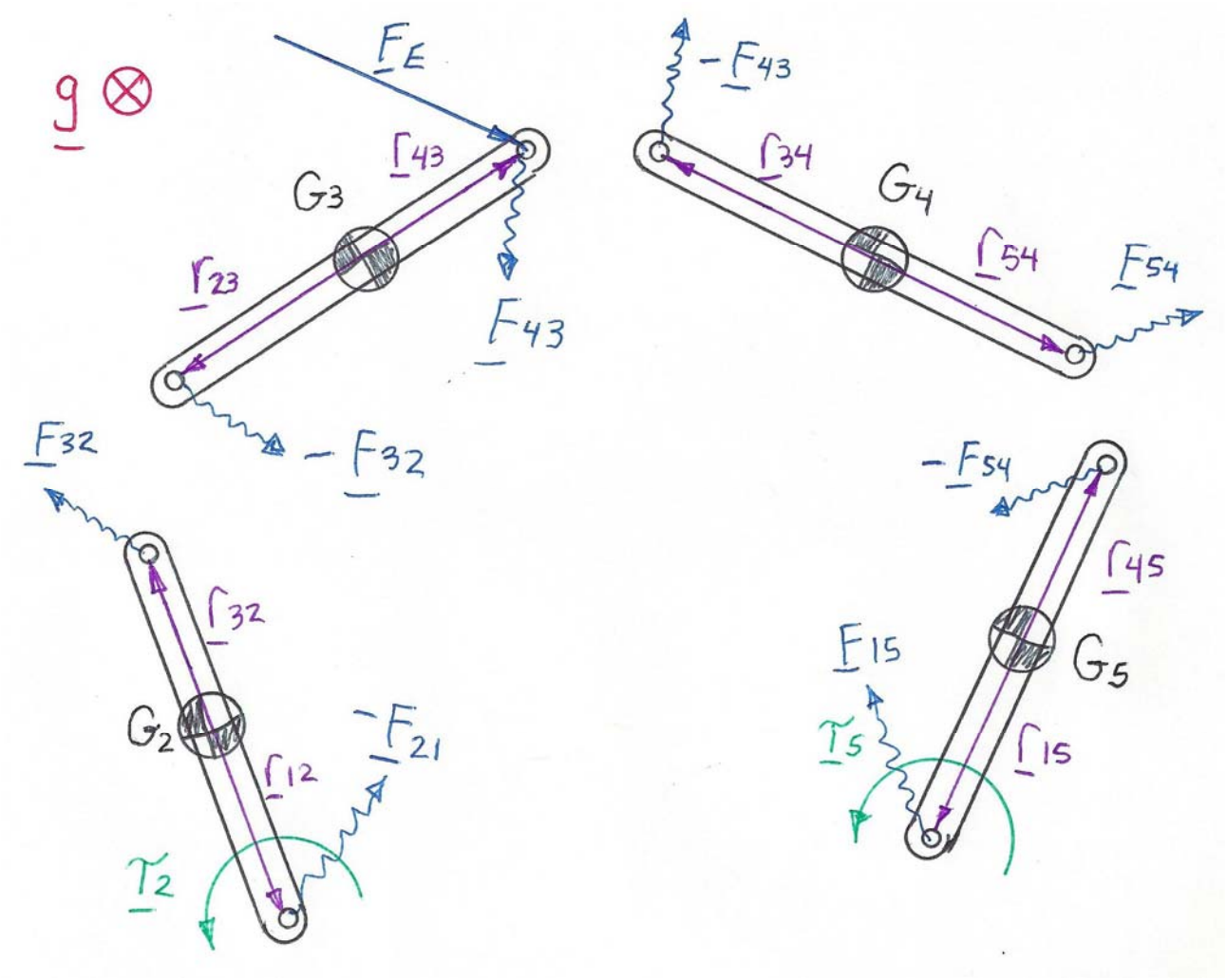
Planar Five-Bar Parallel Robot Acceleration Kinematics for active joints only

Again, it is possible to solve the Forward and Inverse Acceleration Kinematics problem for the planar five-bar parallel robot, ignoring the passive joints velocity variables $\ddot{\theta}_3$ and $\ddot{\theta}_4$. This is similar to that subsection presented in the Planar 2-dof Five-Bar Parallel Robot Velocity Section and is left to the interested reader.

9.2.5 Inverse Dynamics

Step 1. The five-bar robot **Position, Velocity, and Acceleration Analyses** must first be complete.

Step 2. Draw the five-bar robot free-body diagrams (FBDs)



Five-Bar Parallel Robot Free-Body Diagrams (FBDs)

$\underline{F}_{ij}$ unknown vector internal joint force of link *i* acting on link *j*.

$\underline{r}_{ij}$ known moment arm vector pointing to the joint connection with link *i* from the *CG* of link *j*.

Step 3. State the Problem

Given:

The robot (kinematic parameters $r_1, \theta_1, r_2, r_3, r_4, r_5$, masses m_2, m_3, m_4, m_5, center-of-mass vectors CG_2, CG_3, CG_4, CG_5, mass moments of inertia $I_{GZ2}, I_{GZ3}, I_{GZ4}, I_{GZ5}$), kinematic motion angles $\theta_2, \theta_3, \theta_4, \theta_5$, angular velocities $\dot{\theta}_2, \dot{\theta}_3, \dot{\theta}_4, \dot{\theta}_5$, angular accelerations $\ddot{\theta}_2, \ddot{\theta}_3, \ddot{\theta}_4, \ddot{\theta}_5$, translational *CG* accelerations $\underline{A}_{G2}, \underline{A}_{G3}, \underline{A}_{G4}, \underline{A}_{G5}$, and external end-effector force $\underline{F}_E$

Find:

The driving actuator torques $\underline{\tau}_2$ and $\underline{\tau}_5$, plus internal joint forces $\underline{F}_{21}, \underline{F}_{32}, \underline{F}_{43}, \underline{F}_{54}, \underline{F}_{15}$

Count the number of scalar unknowns and the number of scalar equations:

- Since this is planar problem there are three scalar dynamics equations per moving link (two forces *XY* from Newton's Second Law and one moment *Z* from Euler's Rotational Dynamics Equation) and there are four moving links, for a total of 3x4 = 12 scalar equations.

- Two vector torques (of one component each) and ten vector internal joint forces (of two components each) are identified above, for a total of 1+1+2x5 = 12 scalar unknowns.

Therefore, this problem can be solved.

Step 4. Derive the Newton-Euler Dynamics Equations

Newton's Second Law

Link 2

$$\sum \underline{F}_2 = \underline{F}_{32} - \underline{F}_{21} = m_2 \underline{A}_{G2}$$

Link 3

$$\sum \underline{F}_3 = \underline{F}_{43} - \underline{F}_{32} + \underline{F}_E = m_3 \underline{A}_{G3}$$

Link 4

$$\sum \underline{F}_4 = \underline{F}_{54} - \underline{F}_{43} = m_4 \underline{A}_{G4}$$

Link 5

$$\sum \underline{F}_5 = \underline{F}_{15} - \underline{F}_{54} = m_5 \underline{A}_{G5}$$

Euler's Rotational Dynamics Equation

Link 2

$$\sum \underline{M}_{G2} = \underline{\tau}_2 + \underline{r}_{32} \times \underline{F}_{32} - \underline{r}_{12} \times \underline{F}_{21} = I_{G2Z} \underline{\alpha}_2$$

Link 3

$$\sum \underline{M}_{G3} = \underline{r}_{43} \times \underline{F}_{43} - \underline{r}_{23} \times \underline{F}_{32} + \underline{r}_{43} \times \underline{F}_E = I_{G3Z} \underline{\alpha}_3$$

Link 4

$$\sum \underline{M}_{G4} = \underline{r}_{54} \times \underline{F}_{54} - \underline{r}_{34} \times \underline{F}_{43} = I_{G4Z} \underline{\alpha}_4$$

Link 5

$$\sum \underline{M}_{G2} = \underline{\tau}_5 + \underline{r}_{15} \times \underline{F}_{15} - \underline{r}_{45} \times \underline{F}_{54} = I_{G5Z} \underline{\alpha}_5$$

Step 5. **Derive the *XYZ* scalar dynamics equations** from the vector dynamics equations.

For each moving link we obtain

- Two *XY* force component equations from Newton's Second Law
- One *Z* moment equation from Euler's Rotational Dynamics Equation

Link 2

$$F_{32X} - F_{21X} = m_2 A_{G2X}$$

$$F_{32Y} - F_{21Y} = m_2 A_{G2Y}$$

$$\tau_2 + (r_{32X} F_{32Y} - r_{32Y} F_{32X}) - (r_{12X} F_{21Y} - r_{12Y} F_{21X}) = I_{G2Z} \alpha_2$$

Link 3

$$F_{43X} - F_{32X} = m_3 A_{G3X} - F_{EX}$$

$$F_{43Y} - F_{32Y} = m_3 A_{G3Y} - F_{EY}$$

$$(r_{43X} F_{43Y} - r_{43Y} F_{43X}) - (r_{23X} F_{32Y} - r_{23Y} F_{32X}) = I_{G3Z} \alpha_3 - r_{43X} F_{EY} + r_{43Y} F_{EX}$$

Link 4

$$F_{14X} - F_{43X} = m_4 A_{G4X}$$

$$F_{14Y} - F_{43Y} = m_4 A_{G4Y}$$

$$(r_{54X} F_{54Y} - r_{54Y} F_{54X}) - (r_{34X} F_{43Y} - r_{34Y} F_{43X}) = I_{G4Z} \alpha_4$$

Link 5

$$F_{15X} - F_{54X} = m_5 A_{G5X}$$

$$F_{15Y} - F_{54Y} = m_5 A_{G5Y}$$

$$\tau_5 + (r_{15X} F_{15Y} - r_{15Y} F_{15X}) - (r_{45X} F_{54Y} - r_{45Y} F_{54X}) = I_{G5Z} \alpha_5$$

Step 5. Derive the *XYZ* scalar dynamics equations (cont.)

Write these *XYZ* scalar equations in matrix/vector form.

Five-bar robot inverse dynamics 12x12 matrix-vector equation

$$\begin{bmatrix} -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ r_{12Y} & -r_{12X} & -r_{32Y} & r_{32X} & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & r_{23Y} & -r_{23X} & -r_{43Y} & r_{43X} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & r_{34Y} & -r_{34X} & -r_{54Y} & r_{54X} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & r_{45Y} & -r_{45X} & -r_{15Y} & r_{15X} & 0 & 1 \end{bmatrix} \begin{Bmatrix} F_{21X} \\ F_{21Y} \\ F_{32X} \\ F_{32Y} \\ F_{43X} \\ F_{43Y} \\ F_{54X} \\ F_{54Y} \\ F_{15X} \\ F_{15Y} \\ \tau_2 \\ \tau_5 \end{Bmatrix} = \begin{Bmatrix} m_2 A_{G2X} \\ m_2 A_{G2Y} \\ I_{G2Z}\alpha_2 \\ m_3 A_{G3X} - F_{EX} \\ m_3 A_{G3Y} - F_{EY} \\ I_{G3Z}\alpha_3 - r_{43X}F_{EY} + r_{43Y}F_{EX} \\ m_4 A_{G4X} \\ m_4 A_{G4Y} \\ I_{G4Z}\alpha_4 \\ m_5 A_{G5X} \\ m_5 A_{G5Y} \\ I_{G5Z}\alpha_5 \end{Bmatrix}$$

$$[A]\{v\} = \{b\}$$

Step 6. Solve for the unknowns

The coefficient matrix [*A*] is dependent on geometry (through the moment arms, which are dependent on the angles from kinematics solutions). The known vector $\{b\}$ is dependent on inertial terms, gravity, and the given external forces and moments. $\{v\}$ is the vector of unknowns.

Solution by matrix inversion $\{v\} = [A]^{-1}\{b\}$

MATLAB
```
v = inv(A)*b;  % Solution via matrix inverse
```

Using Gaussian elimination is more efficient and robust to solve for **v**.

MATLAB
```
v = A\b;       % Solution via Gaussian elimination
```

The solution to the unknown internal forces and input torque are contained in the components of **v**. To save these values for plotting later, use the following MATLAB code, inside the **for i** loop.

```
F21x(i) = v(1);
F21y(i) = v(2);
:         :
:         :
F15y(i) = v(10);
tau2(i) = v(11);
tau5(i) = v(12);
```

Terms for the matrix-vector equation

Absolute translational center-of-gravity accelerations

$$\underline{A}_{G2} = \begin{Bmatrix} A_{G2X} \\ A_{G2Y} \end{Bmatrix} = \begin{Bmatrix} -R_{G2}\alpha_2 \sin\theta_2 - R_{G2}\omega_2^2 \cos\theta_2 \\ R_{G2}\alpha_2 \cos\theta_2 - R_{G2}\omega_2^2 \sin\theta_2 \end{Bmatrix}$$

$$\underline{A}_{G3} = \begin{Bmatrix} A_{G3X} \\ A_{G3Y} \end{Bmatrix} = \begin{Bmatrix} -r_2\alpha_2 \sin\theta_2 - r_2\omega_2^2 \cos\theta_2 - R_{G3}\alpha_3 \sin\theta_3 - R_{G3}\omega_3^2 \cos\theta_3 \\ r_2\alpha_2 \cos\theta_2 - r_2\omega_2^2 \sin\theta_2 + R_{G3}\alpha_3 \cos\theta_3 - R_{G3}\omega_3^2 \sin\theta_3 \end{Bmatrix}$$

$$\underline{A}_{G4} = \begin{Bmatrix} A_{G4X} \\ A_{G4Y} \end{Bmatrix} = \begin{Bmatrix} -r_5\alpha_5 \sin\theta_5 - r_5\omega_5^2 \cos\theta_5 - R_{G4}\alpha_4 \sin\theta_4 - R_{G4}\omega_4^2 \cos\theta_4 \\ r_5\alpha_5 \cos\theta_5 - r_5\omega_5^2 \sin\theta_5 + R_{G4}\alpha_4 \cos\theta_4 - R_{G4}\omega_4^2 \sin\theta_4 \end{Bmatrix}$$

$$\underline{A}_{G5} = \begin{Bmatrix} A_{G5X} \\ A_{G5Y} \end{Bmatrix} = \begin{Bmatrix} -R_{G5}\alpha_5 \sin\theta_5 - R_{G5}\omega_5^2 \cos\theta_5 \\ R_{G5}\alpha_5 \cos\theta_5 - R_{G5}\omega_5^2 \sin\theta_5 \end{Bmatrix}$$

where $R_{Gi} = r_i/2$ for uniformly-distributed homogeneous material with regular geometry.

Moment arm position vectors

$$\underline{r}_{12} = \begin{Bmatrix} r_{12X} \\ r_{12Y} \end{Bmatrix} = \begin{Bmatrix} -\frac{r_2}{2}\cos\theta_2 \\ -\frac{r_2}{2}\sin\theta_2 \end{Bmatrix}$$

$$\underline{r}_{34} = \begin{Bmatrix} r_{34X} \\ r_{34Y} \end{Bmatrix} = \begin{Bmatrix} \frac{r_4}{2}\cos\theta_4 \\ \frac{r_4}{2}\sin\theta_4 \end{Bmatrix}$$

$$\underline{r}_{32} = \begin{Bmatrix} r_{32X} \\ r_{32Y} \end{Bmatrix} = \begin{Bmatrix} \frac{r_2}{2}\cos\theta_2 \\ \frac{r_2}{2}\sin\theta_2 \end{Bmatrix}$$

$$\underline{r}_{54} = \begin{Bmatrix} r_{54X} \\ r_{54Y} \end{Bmatrix} = \begin{Bmatrix} -\frac{r_4}{2}\cos\theta_4 \\ -\frac{r_4}{2}\sin\theta_4 \end{Bmatrix}$$

$$\underline{r}_{23} = \begin{Bmatrix} r_{23X} \\ r_{23Y} \end{Bmatrix} = \begin{Bmatrix} -\frac{r_3}{2}\cos\theta_3 \\ -\frac{r_3}{2}\sin\theta_3 \end{Bmatrix}$$

$$\underline{r}_{15} = \begin{Bmatrix} r_{15X} \\ r_{15Y} \end{Bmatrix} = \begin{Bmatrix} -\frac{r_5}{2}\cos\theta_5 \\ -\frac{r_5}{2}\sin\theta_5 \end{Bmatrix}$$

$$\underline{r}_{43} = \begin{Bmatrix} r_{43X} \\ r_{43Y} \end{Bmatrix} = \begin{Bmatrix} \frac{r_3}{2}\cos\theta_3 \\ \frac{r_3}{2}\sin\theta_3 \end{Bmatrix}$$

$$\underline{r}_{45} = \begin{Bmatrix} r_{45X} \\ r_{45Y} \end{Bmatrix} = \begin{Bmatrix} \frac{r_5}{2}\cos\theta_5 \\ \frac{r_5}{2}\sin\theta_5 \end{Bmatrix}$$

9.3 Intersection of Two 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

The two circle equations are:

$$(x-a_x)^2+(y-a_y)^2=r_1^2$$
$$(x-b_x)^2+(y-b_y)^2=r_2^2$$

A simpler method is presented in the EE/ME 4290/5290 NotesBook. Here we solve the above equations directly, without utilizing that transformation approach.

Expanding these two equations, and subtracting the second one from the first yields (note the squared unknowns have been eliminated): *y* as a function of *x*:

$$y = dx + e$$

where:

$$d = -\frac{(b_x - a_x)}{(b_y - a_y)}$$
$$e = \frac{r_1^2 - r_2^2 + b_x^2 - a_x^2 + b_y^2 - a_y^2}{2(b_y - a_y)}$$

Substituting this function for *y* into the first circle equation allows us to solve for *x*:

$$ax^2 + bx + c = 0$$

where:

$$a = 1 + d^2$$
$$b = 2(de - a_x - a_y d)$$
$$c = a_x^2 + a_y^2 + e^2 - 2a_y e - r_1^2$$

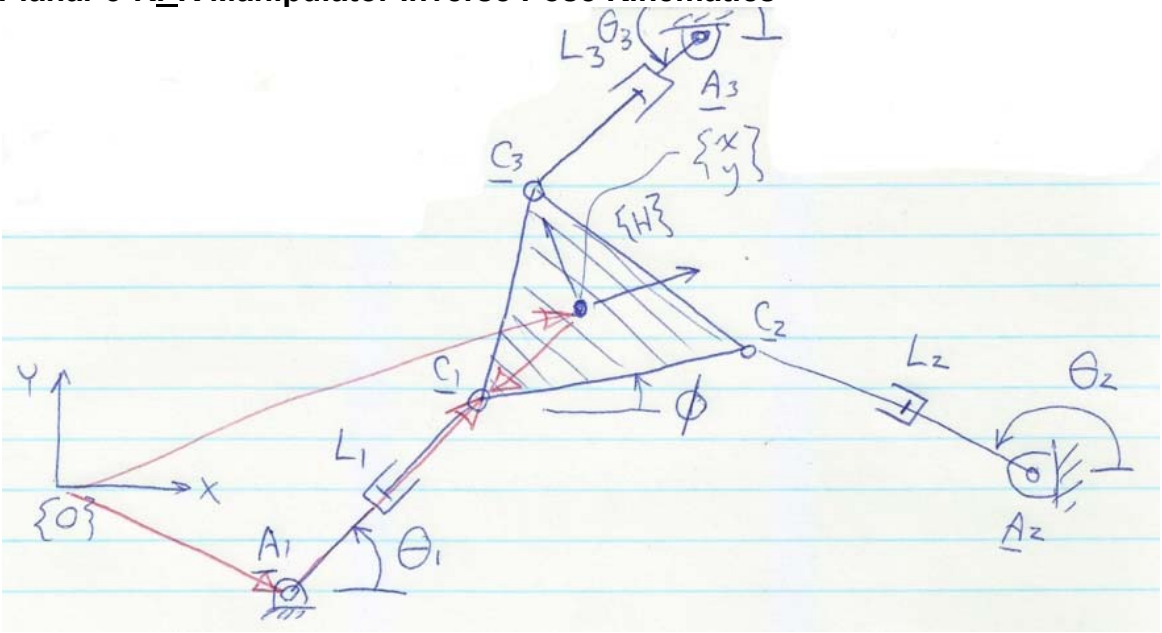
And so the solution is:

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$
$$y_{1,2} = dx_{1,2} + e$$

Note that these two (*x*,*y*) solution points are expressed in the coordinates of the one reference frame {0}.

9.4 Planar 3-RPR Manipulator

9.4.1 Planar 3-RPR Manipulator Inverse Pose Kinematics



Planar 3-RPR Manipulator Kinematic Diagram

Inverse Pose Kinematics (for pose control)

Given: x, y, ϕ

Find: L_1, L_2, L_3

Vector loop-closure equations:

$$\left\{{}^0\mathbf{P}_H\right\} + \left[{}^0_H\mathbf{R}\right]\left\{{}^H\mathbf{P}_{C_i}\right\} = \left\{{}^0\mathbf{A}_i\right\} + \left\{{}^0\mathbf{L}_i\right\} \qquad i = 1,2,3$$

The vector loop-closure equations are rewritten below:

$$\left\{{}^0\mathbf{L}_i\right\} = \left\{{}^0\mathbf{P}_H\right\} + \left[{}^0_H\mathbf{R}\right]\left\{{}^H\mathbf{P}_{C_i}\right\} - \left\{{}^0\mathbf{A}_i\right\} \qquad i = 1,2,3$$

The inverse pose solution is straight-forward, found independently for each of the three legs. The Euclidean norm is used in the equations below.

$$L_i = \left\|\left\{{}^0\mathbf{L}_i\right\}\right\| = \left\|\left\{{}^0\mathbf{P}_H\right\} + \left[{}^0_H\mathbf{R}\right]\left\{{}^H\mathbf{P}_{C_i}\right\} - \left\{{}^0\mathbf{A}_i\right\}\right\| \qquad i = 1,2,3$$

We can also calculate the intermediate passive joint variables $\theta_1, \theta_2, \theta_3$, (independently for each of the three legs) for use in velocity and dynamics analyses. The quadrant-specific atan2 function must be used in the equations below.

$$\theta_i = \tan^{-1}\left[\frac{C_{iy} - A_{iy}}{C_{ix} - A_{ix}}\right] \qquad i = 1,2,3$$

9.4.2 Planar 3-RPR Manipulator Forward Pose Kinematics

3-RPR parallel robot Forward Pose Kinematics (for simulation and sensor-based control)

Given: L_1, L_2, L_3

Find: x, y, ϕ

This is a coupled, nonlinear problem to solve – it is difficult to solve and multiple solutions generally exist, like the **Inverse Pose Kinematics** problem for serial robots. We use the same vector loop-closure equations from Inverse Pose Kinematics, repeated and with details filled in below:

$$\left\{ {}^0\mathbf{L}_i \right\} = \left\{ {}^0\mathbf{P}_H \right\} + \left[{}^0_H\mathbf{R} \right] \left\{ {}^H\mathbf{P}_{C_i} \right\} - \left\{ {}^0\mathbf{A}_i \right\} \qquad i = 1, 2, 3$$

$$\begin{Bmatrix} L_i c\theta_i \\ L_i s\theta_i \end{Bmatrix} = \begin{Bmatrix} x \\ y \end{Bmatrix} + \begin{bmatrix} c\phi & -s\phi \\ s\phi & c\phi \end{bmatrix} \begin{Bmatrix} {}^H C_{ix} \\ {}^H C_{iy} \end{Bmatrix} - \begin{Bmatrix} {}^0 A_{ix} \\ {}^0 A_{iy} \end{Bmatrix}$$

Considering all three legs simultaneously (the problem is coupled and nonlinear), these represent six scalar equations in the six unknowns $x, y, \phi, \theta_1, \theta_2, \theta_3$.

We can use the Newton-Raphson numerical iteration technique to solve this Forward Pose Kinematics Problem. We can directly solve the above six equations for the six unknowns.

$$\begin{aligned} L_i c\theta_i &= x + c\phi\, {}^H C_{ix} - s\phi\, {}^H C_{iy} - {}^0 A_{ix} \\ L_i s\theta_i &= y + s\phi\, {}^H C_{ix} + c\phi\, {}^H C_{iy} - {}^0 A_{iy} \end{aligned} \qquad i = 1, 2, 3$$

However, we don't always need the intermediate variables $\theta_1, \theta_2, \theta_3$ (also, we can calculate these angles later, using Inverse Pose Kinematics, if required for velocity, dynamics, or computer simulation). So, to simplify the Forward Pose Kinematics Problem, square and add each *XY* equation pair (for all three legs) to eliminate the intermediate variables $\theta_1, \theta_2, \theta_3$. Then we will have three equations to solve for the three primary unknowns x, y, ϕ. This problem is solved via the Newton-Raphson iterative numerical method in Section 9.5.

$$\begin{aligned} x^2 + y^2 + A_{ix}^2 + A_{iy}^2 + C_{ix}^2 + C_{iy}^2 - L_i^2 - 2(xA_{ix} + yA_{iy}) + 2c\phi(xC_{ix} + yC_{iy} - A_{ix}C_{ix} - A_{iy}C_{iy}) \\ +2s\phi(-xC_{iy} + yC_{ix} + A_{ix}C_{iy} - A_{iy}C_{ix}) = 0 \end{aligned} \qquad i = 1, 2, 3$$

where known constants ${}^0A_{ix}, {}^0A_{iy}, {}^HC_{ix}, {}^HC_{iy}$ were shortened to $A_{ix}, A_{iy}, C_{ix}, C_{iy}$ for clarity.

Alternate analytical 3-RPR manipulator forward pose solution

This **3-RPR** robot forward pose problem is equivalent to finding the assembly configurations of a four-bar mechanism with known input/output link lengths L_1, L_2 and an *RR* constraining dyad of known length L_3. By itself the four-bar mechanism has infinite assembly configurations because it has one-dof. *RR* dyad $\underline{A}_3\underline{C}_3$ constrains the mechanism to a statically-determinate structure of 0-dof. Point $\underline{C}_3$ defines a four-bar coupler curve which is a tricircular sextic (sixth-degree algebraic curve) that has a maximum of six intersections with the circle of radius L_3 centered at $\underline{A}_3$ (Hunt, 1990).

Figure:

Branch $A_1C_1C_2A_2$ is a 4-bar mechanism with input angle θ_1 and output angle θ_2 (both unknowns). With given lengths L_1 and L_2, this four-bar mechanism has 1-dof, and it can trace out a coupler curve for point C_3 in the plane. In general, this coupler curve is a ***tricircular sextic***. The forward pose kinematics solution may be found by intersecting leg A_3C_3 (a circle of given radius L_3, centered at known centerpoint A_3) with the coupler curve. There are at most six intersections between a circle and *tricircular sextic* and so there may be up to six real multiple solutions to the 3-RPR parallel robot forward pose kinematics problem. There are always six solutions, but 0, 2, 4, or 6 of these will be real, depending on the commanded configuration and robot geometry.

9.4.3 Planar 3-RPR Manipulator Velocity Kinematics

First the pose configuration variables must all be known. Then we can define and solve two problems.

Forward velocity kinematics (for simulation)

Given: $\dot{L}_1, \dot{L}_2, \dot{L}_3$

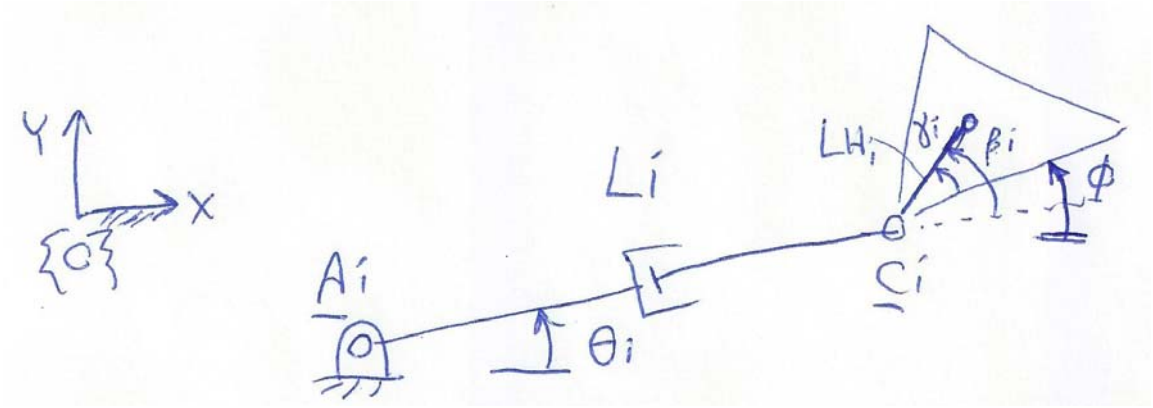
Find: $\dot{x}, \dot{y}, \omega_z$ where $\omega_z = \dot{\phi}$

Inverse velocity kinematics (for resolved-rate control)

Given: $\dot{x}, \dot{y}, \omega_z$

Find: $\dot{L}_1, \dot{L}_2, \dot{L}_3$

In both cases intermediate passive joint rate unknowns $\dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3$ are involved. Both velocity kinematics problems use the same rate equations; we will derive these from looking at the three single **RPR** legs separately (meeting at the end-effector). Here is the figure for the i^{th} leg:



Planar 3-RPR Manipulator Velocity Diagram, leg *i*

As usual, the velocity equations will be obtained by a time derivative of the applicable pose equations. The vector loop-closure equation for leg *i* is:

$$\left\{{}^0\mathbf{P}_H\right\} = \left\{{}^0\mathbf{A}_i\right\} + \left\{{}^0\mathbf{L}_i\right\} + \left\{{}^0\mathbf{L}_{Hi}\right\} \qquad i = 1, 2, 3$$

The *XY* component equations are:

$$\begin{aligned} x &= {}^{0}A_{ix} + L_i c\theta_i + L_{Hi}\cos(\theta_i + \beta_i) \\ y &= {}^{0}A_{iy} + L_i s\theta_i + L_{Hi}\sin(\theta_i + \beta_i) \end{aligned} \qquad i = 1,2,3$$

and the angle equation is:

$$\phi + \gamma_i = \theta_i + \beta_i \qquad i = 1,2,3$$

β_i for use in velocity equations:

$$\begin{aligned} \theta_1 + \beta_1 &= \phi + \gamma_1 \\ \theta_2 + \beta_2 &= \phi + \gamma_2 + 120^\circ \\ \theta_3 + \beta_3 &= \phi + 270^\circ \end{aligned}$$

These β_i relationships assume symmetry, with an equilateral end-effector triangle having $\gamma_1 = \gamma_2 = \gamma_3 = 30^\circ$.

The velocity equations for one **R**<u>**P**</u>**R** leg are obtained from the first time derivatives of the *XY* and angle equations.

$$\begin{aligned} \dot{x} &= -L_i s\theta_i\dot{\theta}_i + \dot{L}_i c\theta_i - L_{Hi}\sin(\theta_i + \beta_i)(\dot{\theta}_i + \dot{\beta}_i) \\ \dot{y} &= \quad L_i c\theta_i\dot{\theta}_i + \dot{L}_i s\theta_i + L_{Hi}\cos(\theta_i + \beta_i)(\dot{\theta}_i + \dot{\beta}_i) \qquad i = 1,2,3 \\ \dot{\phi} &= \dot{\theta}_i + \dot{\beta}_i \end{aligned}$$

These equations are written in matrix-vector form to yield the **R**<u>**P**</u>**R** leg Jacobian matrix.

$$\begin{Bmatrix} \dot{x} \\ \dot{y} \\ \omega_z \end{Bmatrix} = \begin{bmatrix} -L_i s\theta_i - L_{Hi}\sin(\theta_i + \beta_i) & c\theta_i & -L_{Hi}\sin(\theta_i + \beta_i) \\ L_i c\theta_i + L_{Hi}\cos(\theta_i + \beta_i) & s\theta_i & L_{Hi}\cos(\theta_i + \beta_i) \\ 1 & 0 & 1 \end{bmatrix} \begin{Bmatrix} \dot{\theta}_i \\ \dot{L}_i \\ \dot{\beta}_i \end{Bmatrix} \qquad i = 1,2,3$$

Written in compact notation:

$$\{\dot{\mathbf{X}}\} = [\mathbf{J}_i]\{\dot{\boldsymbol{\rho}}_i\} \qquad i = 1,2,3$$

$\{\dot{\mathbf{X}}\} = \{\dot{x} \quad \dot{y} \quad \omega_z\}^T$ is the same for all three legs (the Cartesian end-effector rates). $\{\dot{\boldsymbol{\rho}}_i\}$ includes one active and two passive joint rates for each of the three **RPR** legs, $i = 1, 2, 3$. Let us now find the overall robot Jacobian matrix, using only active rates and ignoring the passive rates. Invert the leg Jacobian matrix symbolically:

$$\begin{Bmatrix} \dot{\theta}_i \\ \dot{L}_i \\ \dot{\beta}_i \end{Bmatrix} = \begin{bmatrix} \dfrac{-s\theta_i}{L_i} & \dfrac{c\theta_i}{L_i} & \dfrac{-L_{Hi}c\beta_i}{L_i} \\ c\theta_i & s\theta_i & L_{Hi}s\beta_i \\ \dfrac{s\theta_i}{L_i} & \dfrac{-c\theta_i}{L_i} & 1+\dfrac{L_{Hi}c\beta_i}{L_i} \end{bmatrix} \begin{Bmatrix} \dot{x} \\ \dot{y} \\ \omega_z \end{Bmatrix} \qquad i = 1, 2, 3$$

Clearly the only singularity condition for this operation is when $L_i = 0$, which is generally impossible by design for the 3-RPR parallel robot.

Extract only the active joint row of this result and assemble all three active joint rows into the overall robot Jacobian matrix:

$$\begin{Bmatrix} \dot{L}_1 \\ \dot{L}_2 \\ \dot{L}_3 \end{Bmatrix} = \begin{bmatrix} c\theta_1 & s\theta_1 & L_{H1}s\beta_1 \\ c\theta_2 & s\theta_2 & L_{H2}s\beta_2 \\ c\theta_3 & s\theta_3 & L_{H3}s\beta_3 \end{bmatrix} \begin{Bmatrix} \dot{x} \\ \dot{y} \\ \omega_z \end{Bmatrix}$$

$$\{\dot{\mathbf{L}}\} = [\mathbf{M}]\{\dot{\mathbf{X}}\}$$

This above equation solves the **Inverse Velocity Problem**. No inversion is required and so the Inverse Velocity problem is **never** singular. Above it is expressed in compact notation. Note inverse velocity Jacobian matrix [**M**] is closely related to the Newton-Raphson Jacobian matrix [$\mathbf{J}_{NR}$] from the numerical FPK solution.

Forward Velocity Problem

This problem is obtained by inverting the Inverse Velocity Solution:

$$\{\dot{\mathbf{X}}\} = \left[\mathbf{M}^{-1}\right]\{\dot{\mathbf{L}}\}$$

Ironically, it is the Forward Velocity Problem that is subject to singularities for parallel robots.

Example Symbolic MATLAB Code

Here is the MATLAB code that was used to symbolically invert the **RPR** leg Jacobian matrix as presented above. Symbolic computing has a lot of power in robot kinematics, dynamics, and control.

```
%
%    Symbolic MATLAB code to invert the RPR leg Jacobian
%

clear; clc;

%    Declare symbolic variables
L   = sym('L');
LH  = sym('LH');
t1  = sym('t1');
b1  = sym('b1');
c1  = sym('cos(t1)');
s1  = sym('sin(t1)');
cp  = sym('cos(t1+b1)');
sp  = sym('sin(t1+b1)');

%    Jacobian elements
j11 = -L*s1-LH*sp;
j21 =  L*c1+LH*cp;
j13 = -LH*sp;
j23 =  LH*cp;

%    Jacobian matrix
J = sym([j11 c1 j13;j21 s1 j23;1 0 1]);

%    Invert
Jinv = inv(J);

%    Simplify
Jinvsimp = simple(Jinv);

%    Check
Ident1 = simple(Jinvsimp * J);
Ident2 = simple(J * Jinvsimp);
```

Note: the first four lines of the declaration statements of the m-code above may be replaced succinctly with:

```
syms L LH t1 b1;
```

9.5 Newton-Raphson Method

The Newton-Raphson Method involves numerical iteration to solve coupled sets of *n* nonlinear equations (algebraic/transcendental – not ODEs) in *n* unknowns. It requires a good initial guess of the solution to get started and it only yields one of the possible multiple solutions. The Newton-Raphson method is an extension of Newton's single function/single variable root-finding technique to *n* functions and *n* variables. The following is the form of the given functions to solve.

$$\{\mathbf{F}(\mathbf{X})\} = \{\mathbf{0}\}$$

where the *n* functions are $\{\mathbf{F}(\mathbf{X})\} = \{F_1(\mathbf{X}) \quad F_2(\mathbf{X}) \quad \cdots \quad F_n(\mathbf{X})\}^T$

and the *n* variables are $\{\mathbf{X}\} = \{x_1 \quad x_2 \quad \cdots \quad x_n\}^T$

Perform a Taylor Series Expansion of {**F**} about {**X**}:

$$F_i(\{\mathbf{X}\} + \{\delta\mathbf{X}\}) = F_i(\{\mathbf{X}\}) + \sum_{j=1}^{n} \frac{\partial F_i}{\partial x_j}\delta x_j + O(\{\delta\mathbf{X}^2\}) \qquad i = 1, 2, \cdots, n$$

Where $[J_{NR}] = [J_{NR}(\mathbf{X})] = \left[\frac{\partial F_i}{\partial x_j}\right]$ is the Newton-Raphson Jacobian Matrix, a multi-dimensional form of the derivative and a function of {**X**}. If $\{\delta\mathbf{X}\}$ is small, the higher-order terms $O(\{\delta\mathbf{X}^2\})$ from the expansion are negligible. For solution, we require:

$$F_i(\{\mathbf{X}\} + \{\delta\mathbf{X}\}) = 0 \qquad i = 1, 2, \cdots, n$$

Now with $O(\{\delta\mathbf{X}^2\}) \to 0$ we have:

$$F_i(\{\mathbf{X}\} + \{\delta\mathbf{X}\}) = F_i(\{\mathbf{X}\}) + \sum_{j=1}^{n} \frac{\partial F_i}{\partial x_j}\delta x_j = F_i(\{\mathbf{X}\}) + [J_{NR}]\{\delta\mathbf{X}\} = \{0\} \qquad i = 1, 2, \cdots, n$$

So to calculate the required correction factor $\delta\mathbf{X}$ at each solution iteration step, we must solve $\{\mathbf{F}(\{\mathbf{X}\})\} + [J_{NR}]\{\delta\mathbf{X}\} = \{0\}$.

$$\{\delta\mathbf{X}\} = -[J_{NR}]^{-1}\{\mathbf{F}(\{\mathbf{X}\})\}$$

Solution via Gaussian elimination on $[J_{NR}]\{\delta\mathbf{X}\} = -\{\mathbf{F}(\{\mathbf{X}\})\}$ is preferable numerically, since this is more efficient and more robust than matrix inversion.

Newton-Raphson Method Algorithm Summary

0) Establish the functions and variables to solve for: $\left\{\mathbf{F}\left(\{\mathbf{X}\}\right)\right\} = \{\mathbf{0}\}$

1) Make an initial guess to the solution: $\left\{\mathbf{X}_0\right\}$

2) Solve $[J_{NR}]\left(\{\mathbf{X}_k\}\right)\{\delta\mathbf{X}_k\} = -\left\{\mathbf{F}\left(\{\mathbf{X}_k\}\right)\right\}$ for $\left\{\delta\mathbf{X}_k\right\}$, where *k* is the iteration counter.

3) Update the current best guess for the solution: $\left\{\mathbf{X}_{k+1}\right\} = \left\{\mathbf{X}_k\right\} + \left\{\delta\mathbf{X}_k\right\}$

4) Iterate until $\left\|\{\delta\mathbf{X}_k\}\right\| < \varepsilon$, where we use the Euclidean norm and ε is a small, user-defined scalar solution tolerance. Also halt the iteration if the number of steps becomes too high (which means the solution is diverging). Generally less than 10 iterations is required for even very tight solution tolerances.

If the initial guess to the solution $\left\{\mathbf{X}_0\right\}$ is sufficiently close to an actual solution, the Newton-Raphson technique guarantees quadratic convergence.

Now, for manipulator forward pose problems, the Newton-Raphson technique requires a good initial guess to ensure convergence and yields only one of the multiple solutions. However, this does not present any difficulty since the existing known pose configuration makes an excellent initial guess for the next solution step (if the control rate is high, many cycles per second, the robot cannot move too far from this known initial guess). Also, except in the case of singularities where the multiple solution branches converge, the one resulting solution is generally the one you want, closest to the initial guess, most likely the actual configuration of the real robot.

There is a very interesting and ***beautiful*** relationship between numerical pose solution and the velocity problem for parallel robots. The Newton-Raphson Jacobian Matrix is nearly identical to the Inverse Velocity Jacobian Matrix for parallel robots. (In the planar case it is identical, in the spatial it is related very closely.) This reduces computation if you need both forward pose computation and inverse-velocity-based resolved-rate control.

3-RPR manipulator forward pose kinematics solution

Use the Newton-Raphson numerical iteration method for solution, with:

$$\{\mathbf{X}\} = \{x \quad y \quad \phi\}^T$$

The three coupled, nonlinear, transcendental functions F_i are the squared and added equations for each **RPR** leg, with L_i^2 brought to the other side to equate the functions to zero.

Derive the required Newton-Raphson Jacobian Matrix.

$$\left[J_{NR}\left(\{\mathbf{X}\}\right)\right] = \left[\frac{\partial F_i}{\partial x_j} \right]$$

$$\frac{\partial F_i}{\partial x} = 2x + 2(C_{ix}c\phi - C_{iy}s\phi) - 2A_{ix} \qquad i = 1,2,3$$

$$\frac{\partial F_i}{\partial y} = 2y + 2(C_{ix}s\phi + C_{iy}c\phi) - 2A_{iy} \qquad i = 1,2,3$$

$$\frac{\partial F_i}{\partial \phi} = -2s\phi(xC_{ix} + yC_{iy} - A_{ix}C_{ix} - A_{iy}C_{iy}) + 2c\phi(-xC_{iy} + yC_{ix} + A_{ix}C_{iy} - A_{iy}C_{ix}) \qquad i = 1,2,3$$

where $\left\{ {}^0\mathbf{A}_i \right\} = \begin{Bmatrix} A_{ix} \\ A_{iy} \end{Bmatrix}$ and $\left\{ {}^H\mathbf{C}_i \right\} = \begin{Bmatrix} C_{ix} \\ C_{iy} \end{Bmatrix}$

Use $i = 1,2,3$ in the above definitions in the proper places in the overall Newton-Raphson Jacobian Matrix. After the forward pose problem is solved at each motion step, we can calculate the intermediate variables $\theta_1, \theta_2, \theta_3$ as in inverse pose kinematics solution above.

Alternate 3-RPR manipulator forward pose Newton-Raphson solution

As we said, we could have solved the original six equations in the six unknowns including the three intermediate variables $\theta_1, \theta_2, \theta_3$. The functions are simpler (no squaring and adding) but the size of the problem is doubled to $i = 1,2,\cdots,6$. Below is the required Jacobian Matrix for this case, where the odd functions are the *X* equations and the even functions are the *Y* equations; also the variable ordering is $\{\mathbf{X}\} = \{x \quad y \quad \phi \quad \theta_1 \quad \theta_2 \quad \theta_3\}^T$.

$$\left[J_{NR}\left(\{\mathbf{X}\}\right)\right] = \begin{bmatrix} 1 & 0 & -P_{1x}s\phi - P_{1y}c\phi & L_1 s\theta_1 & 0 & 0 \\ 0 & 1 & P_{1x}c\phi - P_{1y}s\phi & -L_1 c\theta_1 & 0 & 0 \\ 1 & 0 & -P_{2x}s\phi - P_{2y}c\phi & 0 & L_2 s\theta_2 & 0 \\ 0 & 1 & P_{2x}c\phi - P_{2y}s\phi & 0 & -L_2 c\theta_2 & 0 \\ 1 & 0 & -P_{3x}s\phi - P_{3y}c\phi & 0 & 0 & L_3 s\theta_3 \\ 0 & 1 & P_{3x}c\phi - P_{3y}s\phi & 0 & 0 & -L_3 c\theta_3 \end{bmatrix}$$

9.6 Parallel Manipulator Workspace

Since reduced workspace of parallel robots (when compared to serial robots) is their chief disadvantage, it becomes very important to determine the workspace of parallel robots and maximize it through design.

There are two workspaces to consider (the same as for serial robots in Section 4.4 of the EE/ME 4290/5290 NotesBook).

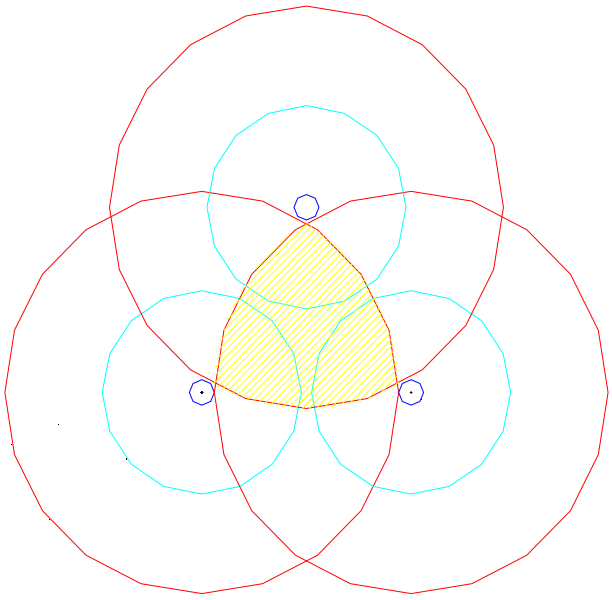
1) **reachable workspace** is the volume in 3D space reachable by the end-effector in **any** orientation

2) **dexterous workspace** is a subset of the **reachable workspace** because it is the volume in 3D space reachable by the end-effector in **all** orientations.

For parallel robots, the **dexterous workspace** is almost always null since the rotation capability is never full for all three Euler angles; therefore we usually define a **reduced dexterous workspace** wherein all Euler angles can reach $\pm 30^{\circ}$ or some other user-definable range.

3-RPR Example

For planar parallel robots we can generally find the **reachable workspace** using a geometric method, figuring out what the end-effector can reach guided by each leg on its own, and then intersecting the results.



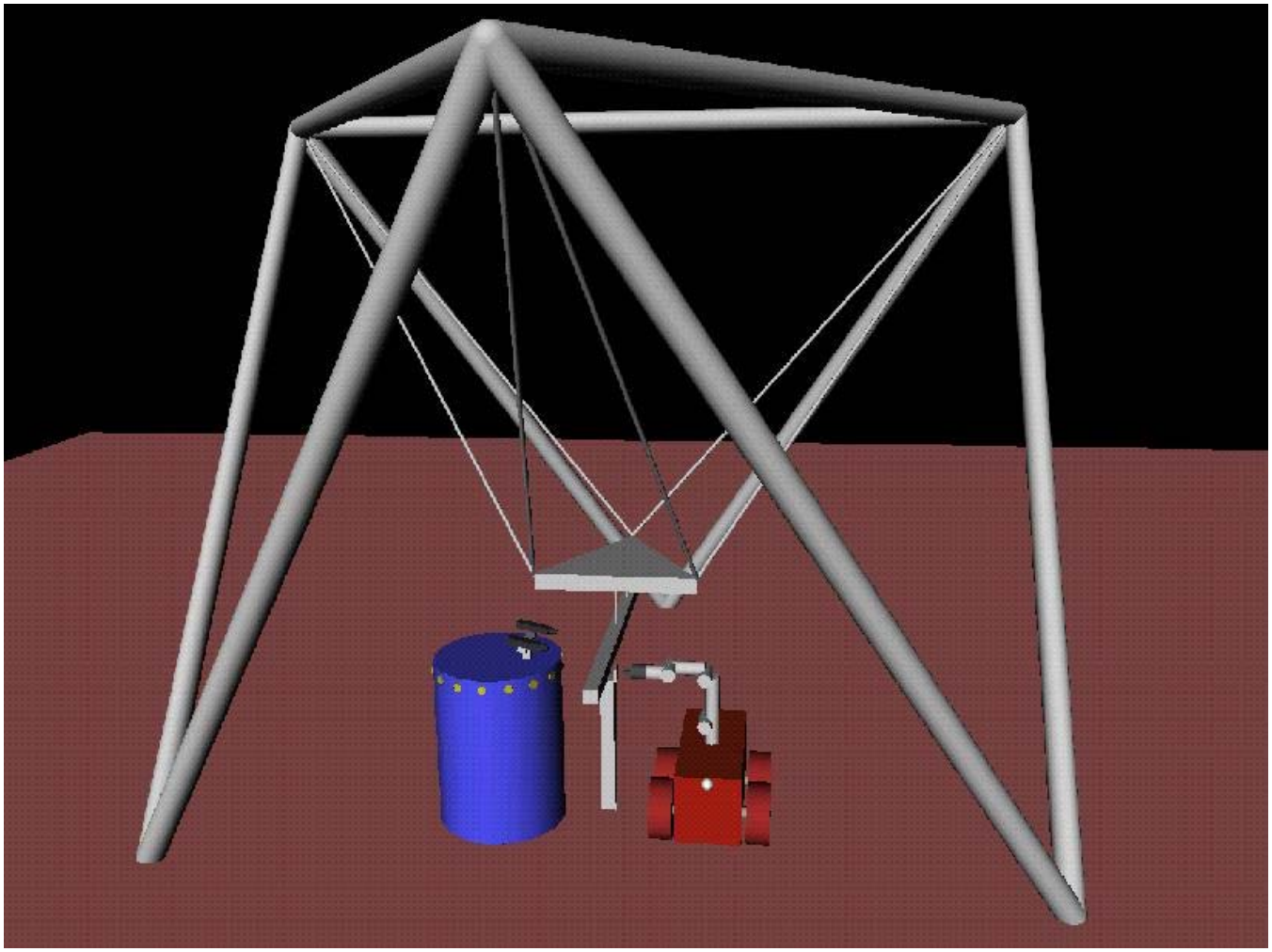
Example 3-RPR Reachable Workspace

For determination of the **dexterous workspace**, it is most convenient to numerically or geometrically generate in MATLAB the reachable workspace for different ϕ values (end-effector orientation) within the desired limits. Then stack these up and intersect them to find the **dexterous workspace**, defined for a reduced desired rotational range $\pm\phi_{LIMIT}$.

9.7 NIST RoboCrane Cable-Suspended Parallel Robot

The NIST RoboCrane[2] is a 6-dof, 6-cable-suspended robot that can position and orient its platform with 6 Cartesian dof. The RoboCrane is classified as an underconstrained cable-suspended robot since gravity is required in addition to the six active cables in order to try to maintain tension in all cables.

Like other cable-suspended robots, the RoboCrane shares the advantages of parallel robots vs. serial robots; in addition, the parallel robot disadvantage of small workspace is wiped out by the RoboCrane, which can have an arbitrarily-large translation workspace (though a very limited rotational workspace like other parallel robots). Cable-suspended robots can be lighter, stiffer, and simpler than other rigid-linked parallel robots. The main disadvantage of the RoboCrane and other cable-suspended robots is that their cable tensions can only be actuated unidirectionally; in plain English, you can't push a rope. This poses a significant controls problem plus the ever-present danger of losing robot tension and stiffness with one or more slack cables.



RoboCrane CAD Model with Serial Arm and Mobile Robots

frc.ri.cmu.edu

[2] J. Albus, R. Bostelman, and N. Dagalakis, 1993, "The NIST RoboCrane", Journal of National Institute of Standards and Technology, 10(5): 709-724.

RoboCrane Description

The six-dof RoboCrane is capable of *XYZ* translational and roll-pitch-yaw rotational control of its moving platform within its workspace. As shown in the kinematics diagrams below, the RoboCrane is essentially an inverted Stewart Platform parallel robot controlled by six active tensioning cables in place of the six hydraulic actuators. The side length of the base equilateral triangle is s_B and the side length of the moving platform equilateral triangle is s_P. At zero orientations, the moving platform equilateral triangle is inverted with respect to the base equilateral triangle as shown.

The fixed base Cartesian reference frame is {0}, located on the ground, whose origin is the center of the base equilateral triangle (from the top view). The base equilateral triangle is located at a height H above the ground. The three ground-fixed cable connection points are $B_i,\ i=1,2,3$ and the three moving-platform-fixed cable connection points are $P_i,\ i=1,2,3$. The six cables are connected between the ground-fixed and platform-fixed cable connection points as shown. The six tensioning motors for RoboCrane can be mounted on the ground (cables routed via pulleys at the top base frame), on the top base frame, or on the moving platform itself. The moving platform Cartesian reference frame is {P}, whose origin is located in the center of the platform equilateral triangle. At zero Euler angles, the orientation of {P} is identical to that of {0}.

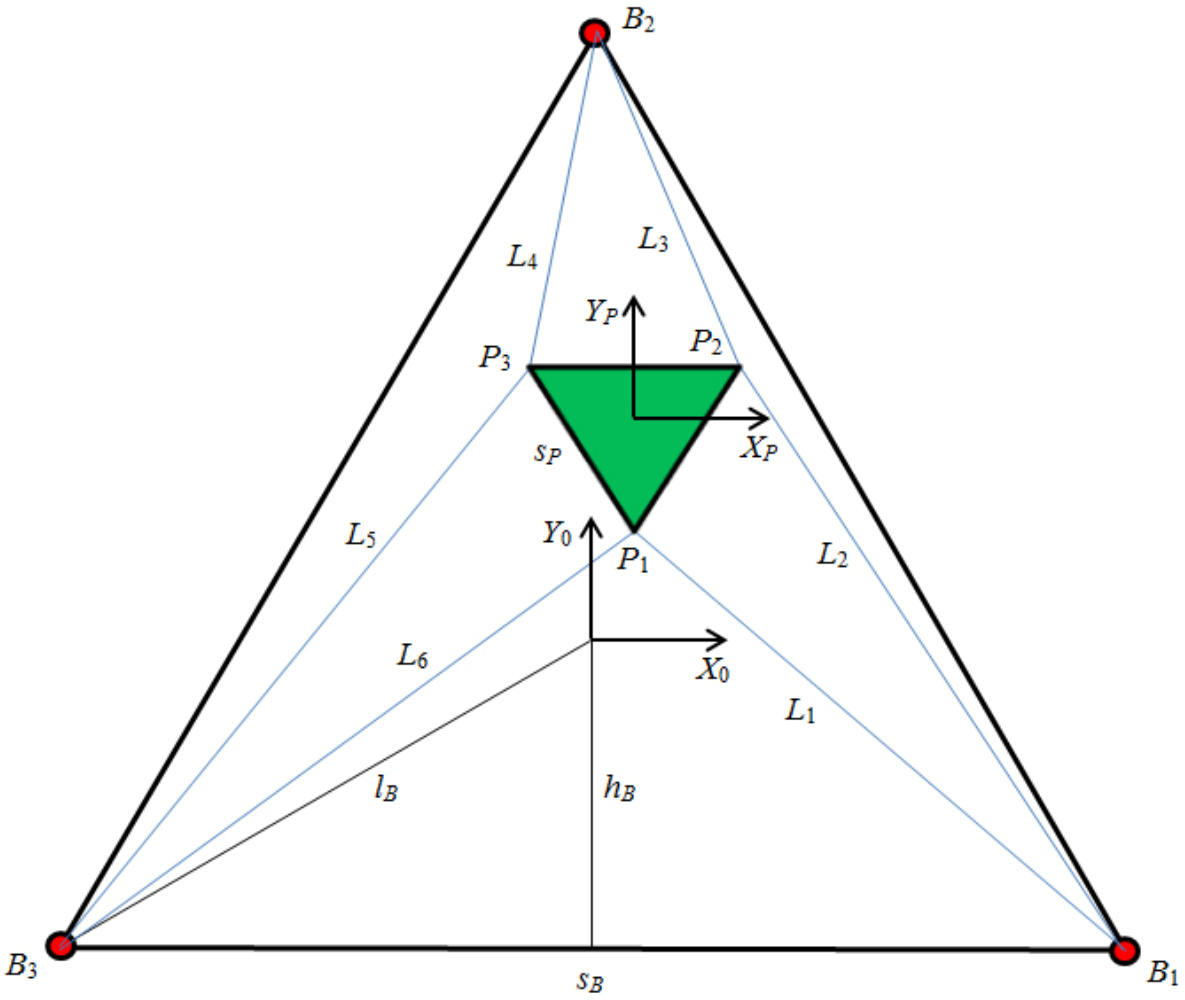
The fixed-base cable connection points B_i are constant in the base frame {0} and the platform-fixed cable connection points P_i are constant in the base frame {P}:

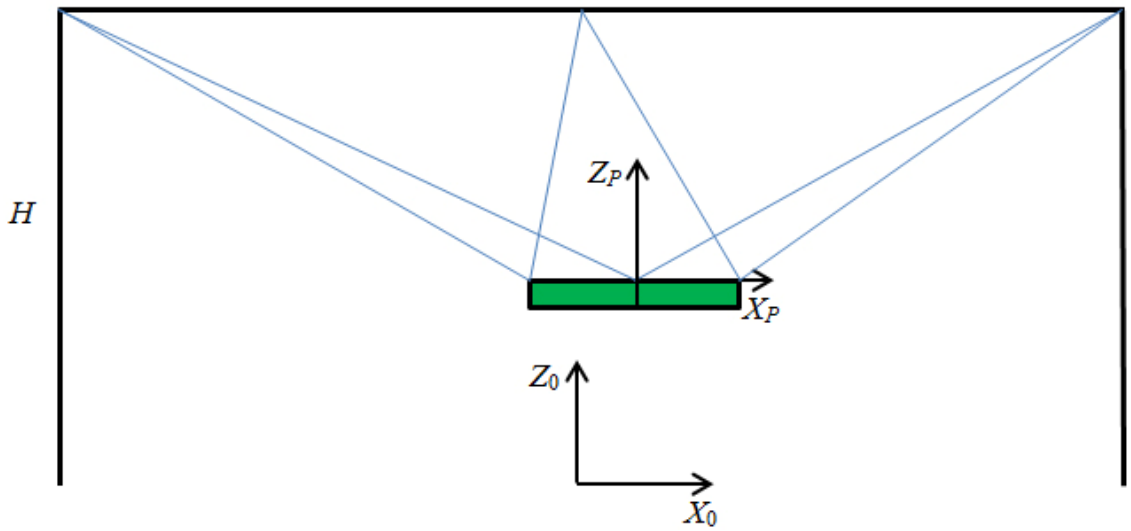
$$ {}^0\mathbf{B}_1 = \begin{Bmatrix} \frac{s_B}{2} \\ -h_B \\ H \end{Bmatrix} \qquad {}^0\mathbf{B}_2 = \begin{Bmatrix} 0 \\ l_B \\ H \end{Bmatrix} \qquad {}^0\mathbf{B}_3 = \begin{Bmatrix} -\frac{s_B}{2} \\ -h_B \\ H \end{Bmatrix} $$

$$ {}^P\mathbf{P}_1 = \begin{Bmatrix} 0 \\ -l_P \\ 0 \end{Bmatrix} \qquad {}^P\mathbf{P}_2 = \begin{Bmatrix} \frac{s_P}{2} \\ h_P \\ 0 \end{Bmatrix} \qquad {}^P\mathbf{P}_3 = \begin{Bmatrix} -\frac{s_P}{2} \\ h_P \\ 0 \end{Bmatrix} $$

where:

$$ h_B = \frac{\sqrt{3}}{6} s_B \qquad l_B = \frac{\sqrt{3}}{3} s_B \qquad h_P = \frac{\sqrt{3}}{6} s_P \qquad l_P = \frac{\sqrt{3}}{3} s_P $$





RoboCrane Kinematics Diagrams (Top and Front Views)

name	meaning
s_B	base equilateral triangle side
s_P	platform equilateral triangle side
H	height from ground to base triangle
h_B	planar distance from {0} to near base side
l_B	planar distance from {0} to a base vertex
h_P	planar distance from {P} to near platform side
l_P	planar distance from {P} to a platform vertex
m	platform mass

Inverse Pose Kinematics (IPK) Solution

The 6-cable RoboCrane robot inverse pose kinematics (IPK) problem is stated: Given the desired moving platform pose $\left[{}^{0}_{P}\mathbf{T} \right]$, calculate the six active cable lengths L_i, $i = 1, 2, \cdots 6$. This IPK model assumes straight cables (no sag) that are always in tension, and ignores cable mass and elasticity.

The IPK input $\left[{}^{0}_{P}\mathbf{T} \right]$ may be specified in terms of the desired vector location $\left\{ {}^{0}\mathbf{P}_{P} \right\}$ of the origin of moving frame {P} with respect to {0}, plus three angles representing the orientation of moving frame {P} with respect to {0}. Choosing α-β-γ, *Z-Y-X* Euler Angles (Craig, 2005), the associated orthonormal rotation matrix is:

$$\left[{}^{0}_{P}\mathbf{R} \right] = \begin{bmatrix} c\alpha c\beta & -s\alpha c\gamma + c\alpha s\beta s\gamma & s\alpha s\gamma + c\alpha s\beta c\gamma \\ s\alpha c\beta & c\alpha c\gamma + s\alpha s\beta s\gamma & -c\alpha s\gamma + s\alpha s\beta c\gamma \\ -s\beta & c\beta s\gamma & c\beta c\gamma \end{bmatrix}$$

Then the 4x4 homogeneous transformation matrix description of pose is (Craig, 2005):

$$\left[{}^{0}_{P}\mathbf{T} \right] = \begin{bmatrix} & \left[{}^{0}_{P}\mathbf{R} \right] & & \left\{ {}^{0}\mathbf{P}_{P} \right\} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The solution to the RoboCrane IPK problem may be used as the basis for a pose control scheme, executing pre-planned trajectories and other motions within its workspace. Like most cable-suspended robots and many parallel robots in general, the RoboCrane IPK solution is straight-forward and poses no computational challenge for real-time implementation. Given the desired moving cabin pose $\left[{}^{0}_{P}\mathbf{T} \right]$, we find the moving platform cable connection points P_i. Then the inverse pose solution consists simply of calculating the cable lengths using the Euclidean norm of the appropriate vector differences between the various moving and fixed cable connection points. The IPK solution yields a unique closed-form solution. The moving cable connection points P_1, P_2, and P_3 with respect to the fixed base frame {0} are:

$$\left\{ {}^{0}\mathbf{P}_{j} \right\} = \left[{}^{0}_{P}\mathbf{T} \right] \left\{ {}^{P}\mathbf{P}_{j} \right\} \qquad j = 1, 2, 3$$

where the $\left\{ {}^{P}\mathbf{P}_j \right\}$ vectors were given previously. Note we must augment each position vector above with a '1' in the fourth row to make the 4x4 matrix multiplication valid. The RoboCrane straight-cable IPK solution is then:

$$L_1 = \left\| {}^{0}\mathbf{B}_1 - {}^{0}\mathbf{P}_1 \right\| \qquad L_2 = \left\| {}^{0}\mathbf{B}_1 - {}^{0}\mathbf{P}_2 \right\|$$

$$L_3 = \left\| {}^{0}\mathbf{B}_2 - {}^{0}\mathbf{P}_2 \right\| \qquad L_4 = \left\| {}^{0}\mathbf{B}_2 - {}^{0}\mathbf{P}_3 \right\|$$

$$L_5 = \left\| {}^{0}\mathbf{B}_3 - {}^{0}\mathbf{P}_3 \right\| \qquad L_6 = \left\| {}^{0}\mathbf{B}_3 - {}^{0}\mathbf{P}_1 \right\|$$

Note the direction of the six cable length vectors above were chosen in the direction of positive cable tensions.

RoboCrane Forward Pose Kinematics (FPK) Solution

The 6-cable RoboCrane forward pose kinematics (FPK) problem is stated: Given the six active cable lengths L_i, $i = 1, 2, \cdots 6$, calculate the resulting moving platform pose $\left[{}^{0}_{P}\mathbf{T} \right]$. The FPK solution for cable-suspended robots and other parallel robots is generally very difficult. It requires the solution of multiple coupled nonlinear (transcendental) algebraic equations, from the vector loop-closure equations. Multiple valid solutions generally result.

Referring to the kinematics diagrams, we see that the RoboCrane FPK problem is identical to the FPK solution for an upside-down 3-3 Stewart Platform (assuming straight cables always under positive tension). The FPK solution is based on identifying 3 known triangles, $B_1P_2B_2$, $B_2P_3B_3$, and $B_3P_1B_1$. Construct a virtual link to P_j, perpendicular to base line B_iB_k for each of these three triangles. Imagine rotating each triangle (each virtual link) about B_iB_k. The FPK solution exists where all three P_j rotate until P_1P_2, P_2P_3, and P_3P_1 are each of the correct, known lengths s_P simultaneously. This solution was presented by Dr. Bob[3]. The solution boils down to an 8th-order polynomial, meaning that there are potentially 8 multiple solutions. Even pairs of some of these solutions may be imaginary.

[3] R.L. Williams II, 1992, "Kinematics of an In-Parallel Actuated Manipulator Based on the Stewart Platform Mechanism", NASA Technical Memorandum 107585, NASA Langley Research Center, Hampton, VA.

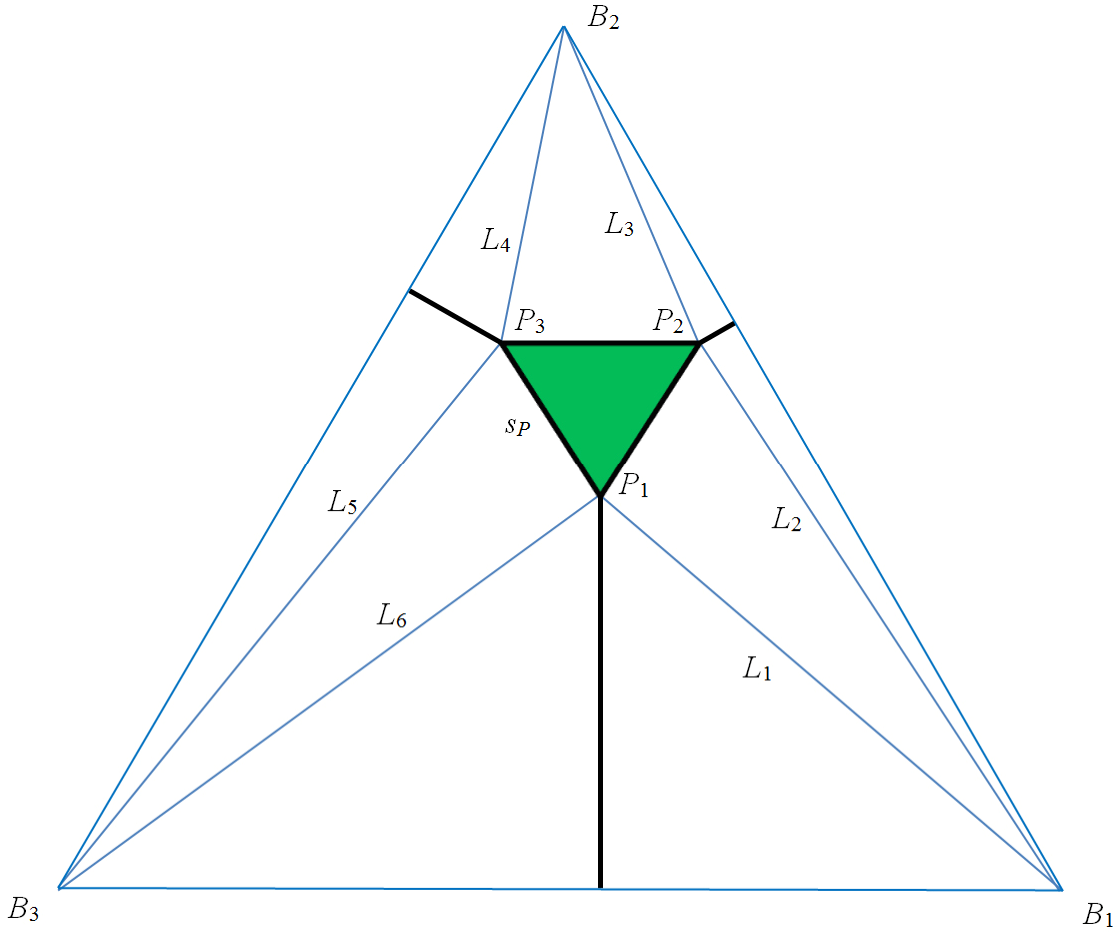


Figure 8. RoboCrane FPK Diagram

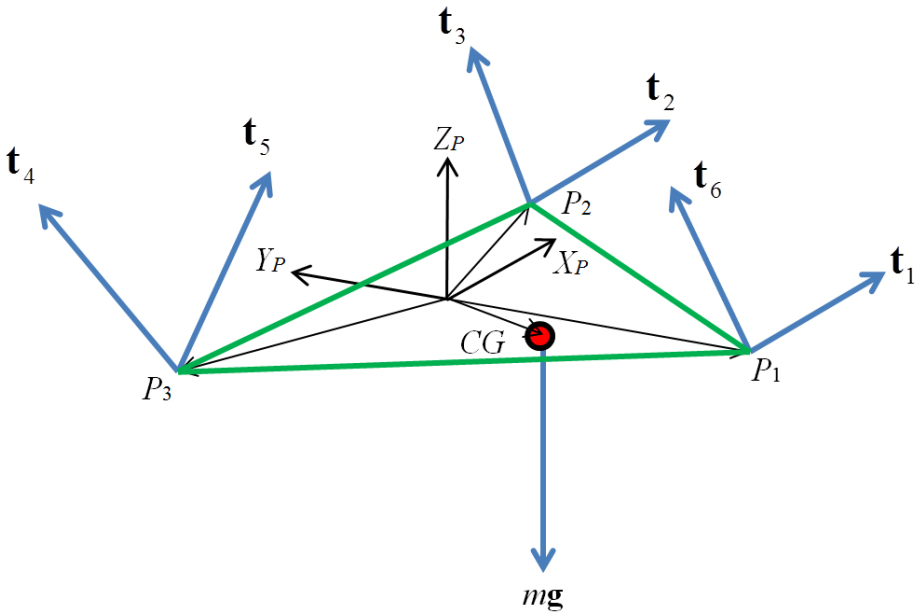
RoboCrane Pseudostatic Analysis

To maintain safe and stable control in all motions, all cable tensions must remain positive at all times. Gravity acting on the moving platform is required to ensure that the six active cables remain in tension, as long as the rotations are not too far from the nominal horizontal orientation. A pseudostatic model is developed in this section and applied to the RoboCrane inverse statics solution.

Equations for Static Equilibrium

This section presents statics modeling for the six-cable-suspended RoboCrane robot. All six active cables connect in parallel from the fixed base to the moving platform, as shown in the kinematics diagrams. Again we assume straight cables (no sag) that are always in tension, and ignore cable mass and elasticity. In pseudostatics it is assumed that the focus cabin accelerations and velocities are low enough to justify ignoring inertial dynamic effects and use statics equations of equilibrium.

For static equilibrium the vector force sum and vector moment sum of the six active cable tensions plus gravitational loading and external wrench acting on the focus cabin must balance to zero. The statics free-body diagram for the moving platform is shown below, where *CG* indicates the center of gravity location. The six active cable tension vectors are $\mathbf{t}_i$, $i = 1, 2, \cdots 6$.



RoboCrane Statics Free-Body Diagram

The vector force and moment equations of static equilibrium are:

$$\sum_{i=1}^{6}\mathbf{t}_i + m\mathbf{g} + \mathbf{F}_{EXT} = \mathbf{0}$$

$$\sum_{i=1}^{6}\mathbf{m}_i + {}^{0}_{P}\mathbf{R}\,{}^{P}\mathbf{P}_{CG} \times m\mathbf{g} + \mathbf{M}_{EXT} = \mathbf{0}$$

where $\mathbf{t}_i = t_i\hat{\mathbf{L}}_i$ is the vector cable tension applied to the moving platform by the i^{th} active cable (in the positive cable length direction $\hat{\mathbf{L}}_i$ as established in IPK); m is the moving platform mass; $\mathbf{g} = \{0 \quad 0 \quad -g\}^T$ is the gravity vector; $\mathbf{F}_{EXT}$ is the external vector force exerted on the moving platform by the environment; $\mathbf{m}_i = \left[{}^{0}_{P}\mathbf{R}\right]\left\{{}^{P}\mathbf{P}_j\right\} \times \left\{\mathbf{t}_i\right\}$ is the moment due to the i^{th} active cable tension ($^{P}\mathbf{P}_j$ is the moment arm from the moving platform control point P to the j^{th} active cable connection point, expressed in $\{P\}$ coordinates); ${}^{P}\mathbf{P}_{CG}$ is the position vector to the moving platform CG from the moving platform control point P (the origin of $\{P\}$); and $\mathbf{M}_{EXT}$ is the external vector moment exerted on the moving platform by the environment. Moments are summed about the moving platform control point P and all vectors must be expressed in a common frame; $\{0\}$ is chosen.

Now we derive the pseudostatics Jacobian matrix based on the vector force and moment statics equations. Substituting the above details into the static equilibrium equations yields:

$$[\mathbf{S}]\{\mathbf{t}\} = -\{\mathbf{W}_{EXT} + \mathbf{G}\}$$

where $\{\mathbf{t}\} = \{t_1 \quad t_2 \quad \cdots \quad t_6\}^T$ is the vector of active cable tensions, $\{\mathbf{G}\} = \{m\mathbf{g} \quad {}^0_P\mathbf{R}\,{}^P\mathbf{P}_{CG} \times m\mathbf{g}\}^T$ is the gravity wrench vector, $\{\mathbf{W}_{EXT}\} = \{\mathbf{F}_{EXT} \quad \mathbf{M}_{EXT}\}^T$ is the external wrench vector, and the statics Jacobian matrix $[\mathbf{S}]$ is:

$$[\mathbf{S}] = \begin{bmatrix} \hat{\mathbf{L}}_1 & \hat{\mathbf{L}}_2 & \hat{\mathbf{L}}_3 & \hat{\mathbf{L}}_4 & \hat{\mathbf{L}}_5 & \hat{\mathbf{L}}_6 \\ \mathbf{P}_1 \times \hat{\mathbf{L}}_1 & \mathbf{P}_2 \times \hat{\mathbf{L}}_2 & \mathbf{P}_2 \times \hat{\mathbf{L}}_3 & \mathbf{P}_3 \times \hat{\mathbf{L}}_4 & \mathbf{P}_3 \times \hat{\mathbf{L}}_5 & \mathbf{P}_1 \times \hat{\mathbf{L}}_6 \end{bmatrix}$$

where $\mathbf{P}_j = \{{}^0\mathbf{P}_j\} = [{}^0_P\mathbf{R}]\{{}^P\mathbf{P}_j\}$, $j = 1, 2, 3$.

RoboCrane Inverse Pseudostatics Solution

The statics equations can be used in two ways. Given the active cable tensions $\{\mathbf{t}\}$ and the six cable unit vectors $\hat{\mathbf{L}}_i$ from kinematics analysis, forward statics analysis uses the equations of static equilibrium directly to verify statics equilibrium. For control, simulation, and valid-tension workspace determination, the more useful problem is inverse statics analysis. This problem is stated: calculate the required active cable tensions $\{\mathbf{t}\}$ given the focus cabin mass and pose, plus all $\hat{\mathbf{L}}_i$. It is solved by inverting the static equilibrium equations:

$$\{\mathbf{t}\} = -[\mathbf{S}]^{-1}\{\mathbf{W}_{EXT} + \mathbf{G}\}$$

The statics Jacobian Matrix $[\mathbf{S}]$ is a square 6x6 matrix and hence the standard matrix inverse applies in (11). A unique $\{\mathbf{t}\}$ solution is guaranteed if the RoboCrane robot is not in a singular pose.

Assuming pseudostatic motion, the inertia of the actuator shafts do not enter into the analysis and the statics torque/tension relationship for each of the 6 actuators is $\tau_i = r_i t_i$, $i = 1, 2, \cdots 6$, where τ_i is the i^{th} actuator torque, r_i is the i^{th} cable reel radius, and t_i is the i^{th} cable tension.

RoboCrane Examples

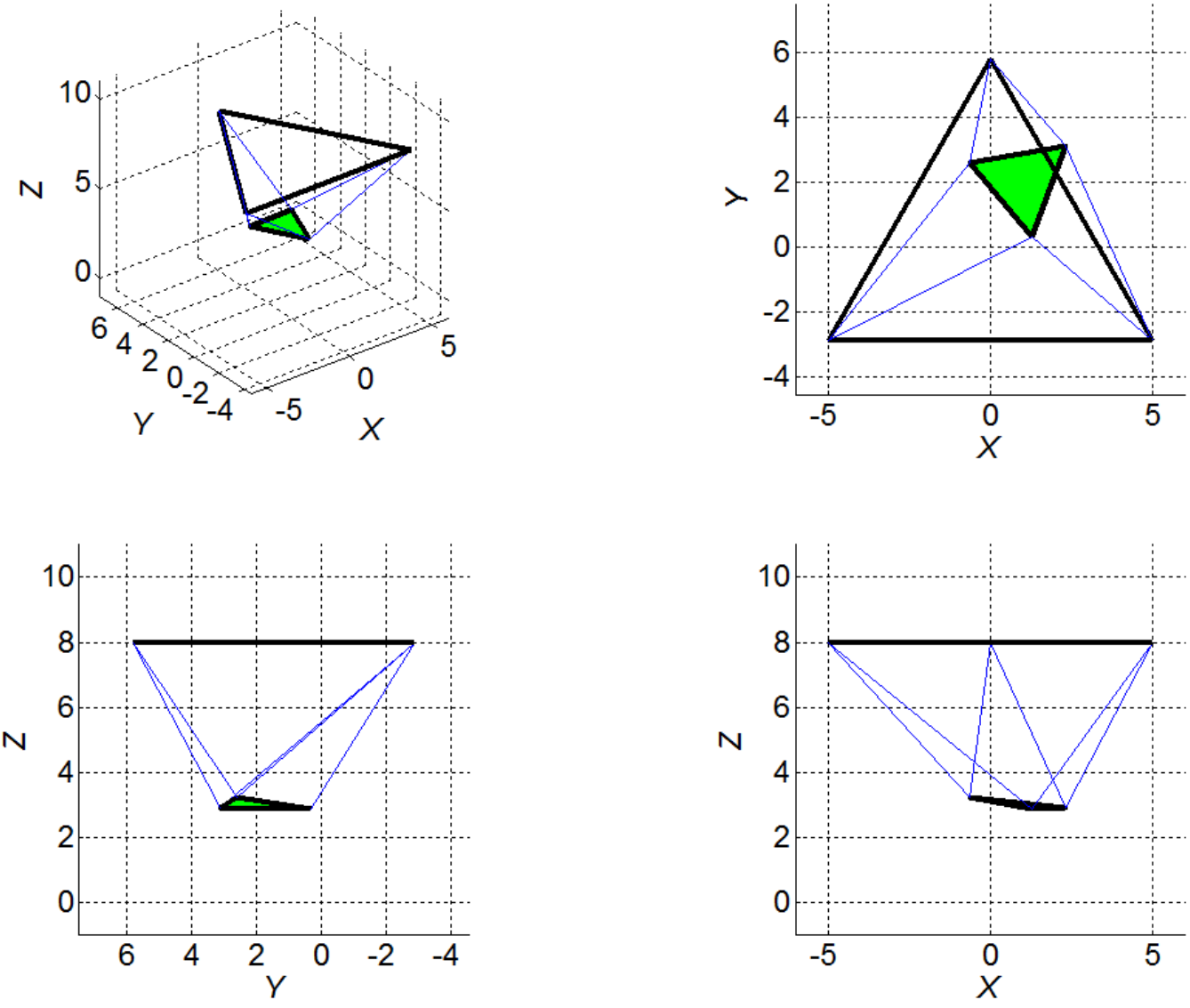
For these examples, the RoboCrane dimensions are s_B = 10 m, s_P = 3 m, H = 8 m, and m_P = 100 kg. All examples assume the platform *CG* is located at the origin of {*P*} so that ${}^{P}\mathbf{P}_{CG} = \{0 \quad 0 \quad 0\}^T$. Also there is zero external wrench $\{\mathbf{W}_{EXT}\}$.

Snapshot Example

Given $\{{}^{0}\mathbf{P}_{P}\} = \{1 \quad 2 \quad 3\}^T$ m and $\{\alpha \quad \beta \quad \gamma\} = \{10^\circ \quad 6^\circ \quad 4^\circ\}$, the calculated IPK and inverse statics results are:

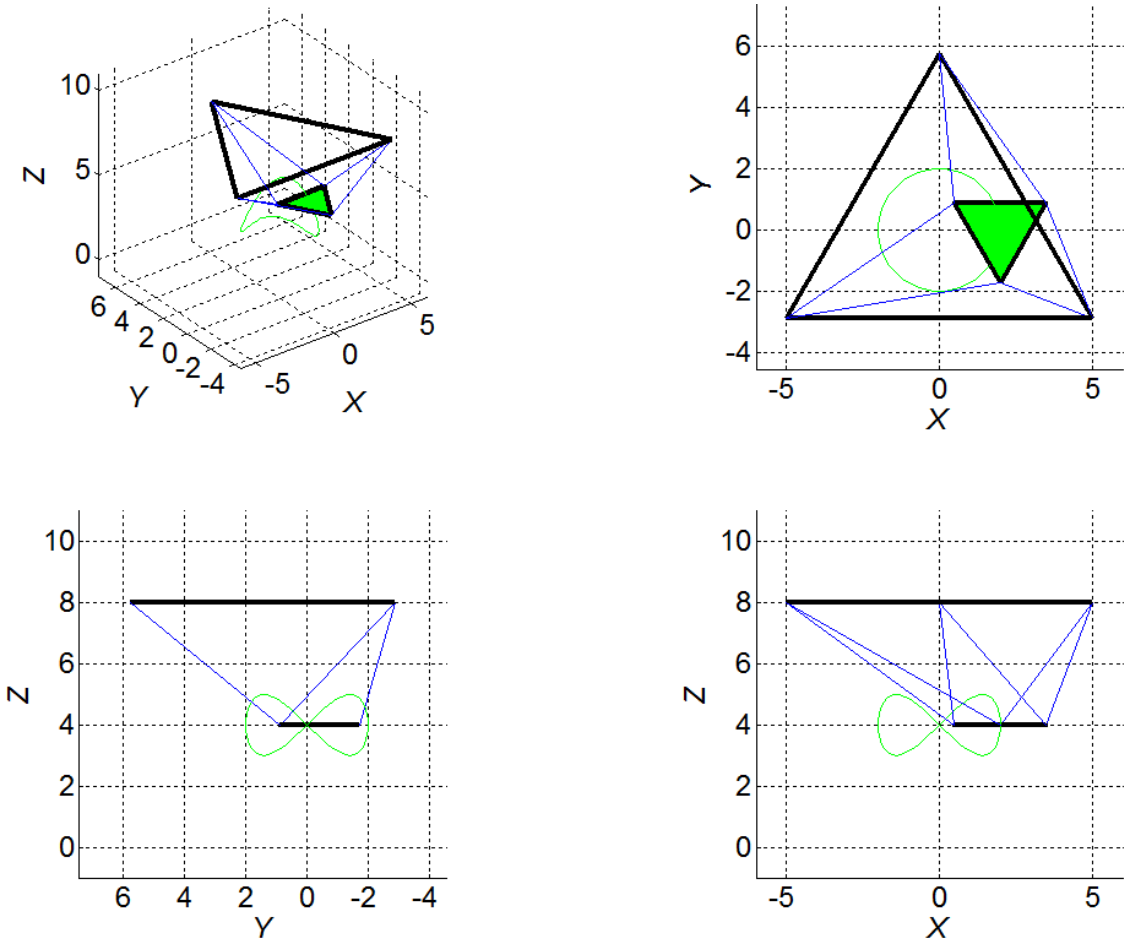
$$\mathbf{L} = \{7.080 \quad 8.313 \quad 6.203 \quad 5.777 \quad 8.494 \quad 8.711\}^T \text{ m}$$

$$\{\mathbf{t}\} = \{325.1 \quad 125.2 \quad 318.6 \quad 352.4 \quad 76.1 \quad 123.8\}^T \text{ N}$$

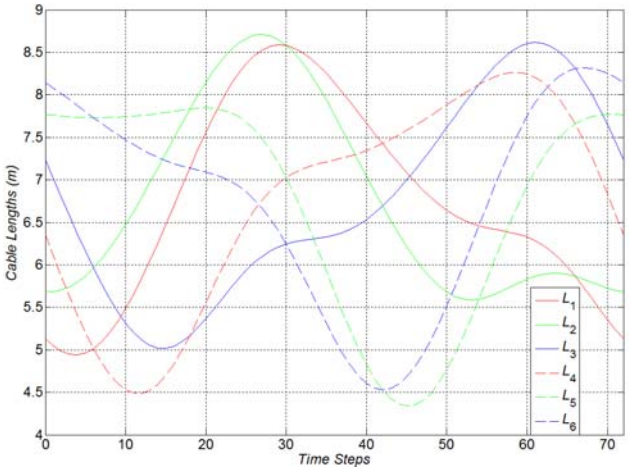


Trajectory Example

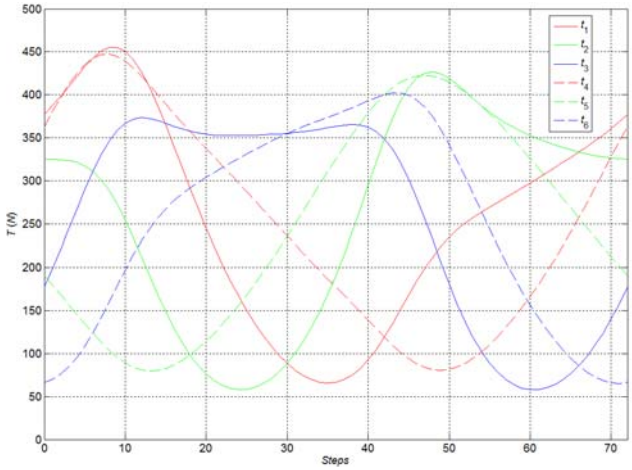
The moving platform control point $\{P\}$ traces a circle of center $\{0 \quad 0 \quad 4\}^T$ and radius 2 m, with zero orientations $\{\alpha \quad \beta \quad \gamma\} = \{0 \quad 0 \quad 0\}$. At the same time, the Z displacement goes through 2 entire sine wave motions centered on $Z = 4$ m with a 1 m amplitude.



Circular Trajectory



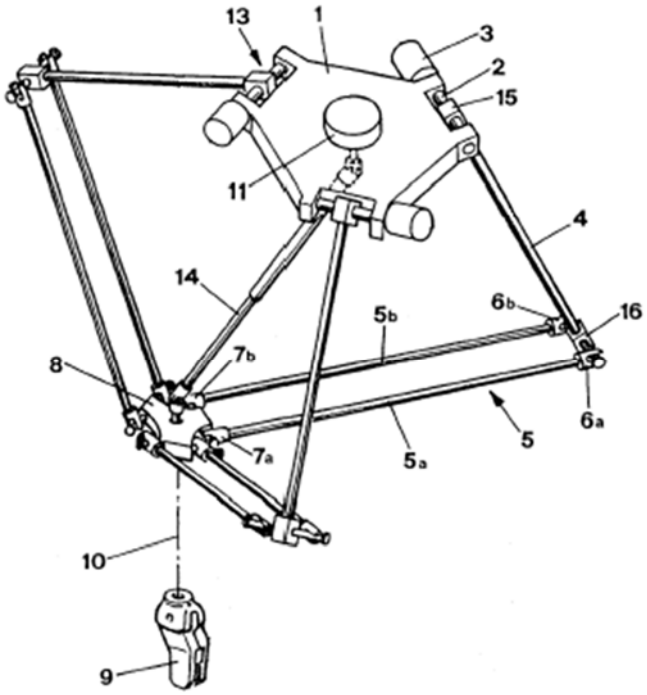
Cable Lengths



Cable Tensions

9.8 Delta Parallel Robot Inverse and Forward Kinematics

Clavel's **3-RUU** Delta Robot[4] is arguably the most successful commercial parallel robot to date. The left image below shows the original design from Clavel's U.S. patent[5], and the right photograph below shows one commercial instantiation of the Delta Robot.



Delta Robot Design[1]

ABB FlexPicker Delta Robot
www.abb.com

The Delta Robot has 4-degrees-of-freedom (dof), 3-dof for *XYZ* translation, plus a fourth inner leg to control a single rotational freedom at the end-effector platform (about the axis perpendicular to the platform). The remainder of this document will focus only on the 3-dof *XYZ* translation-only Delta Robot since that is being widely applied by 3D printers and Arduino hobbyists.

Presented is a description of the 3-dof Delta Robot, followed by kinematics analysis including analytical solutions for the inverse position kinematics problem and the forward position kinematics problem, and then examples for both, snapshots and trajectories. The revolute-inputs **3-RUU** and the prismatic inputs **3-PUU** Delta Robots are both covered.

This section is presented on-line:

ohio.edu/mechanical-faculty/williams/html/pdf/DeltaKin.pdf

[4] R. Clavel, 1991, "Conception d'un robot parallèle rapide à 4 degrés de liberté", Ph.D. Thesis, EPFL, Lausanne, Switzerland.
[5] R. Clavel, 1990, "Device for the Movement and Positioning of an Element in Space", U.S. Patent No. 4,976,582.

9.9 Stewart Platform

The **6-UPS** Stewart-Gough Platform, shown below, has seen significant practical applications from flight simulators and entertainment, to haptic interfaces and general parallel robots.

6-dof Stewart Platform Parallel Robot

Please see the following link for Dr. Bob's NASA Technical Memorandum presenting the kinematics of the Stewart Platform:

ohio.edu/mechanical-faculty/williams/html/PDF/StewartPlatform.pdf

10. Serial Robot Acceleration Kinematics

The acceleration vector is the first time derivative of the velocity vector (and the second time derivative of the position vector). It is linear in acceleration terms but nonlinear in rate components. Both Cartesian acceleration and velocity are vectors, with translational and rotational parts.

Acceleration Kinematics Analysis is useful for

- Resolved Acceleration Control
- Acceleration of any point on manipulator
- Moving objects in workspace – smooth trajectory generation
- Required for Dynamics

Translational Acceleration

$^{k}\left\{ {}^{i}A_{j} \right\}$ is the translational acceleration of the origin of frame $\{j\}$ with respect to reference frame $\{i\}$, expressed in the basis (coordinates) of $\{k\}$.

Transport Theorem (Craig, 2005)

$$\frac{d}{dt}\left(\left[{}^{A}_{B}R \right]\left\{ {}^{B}Q \right\}\right)=\left[{}^{A}_{B}R \right]\left\{ {}^{B}V_{Q} \right\}+\left\{ {}^{A}\omega_{B} \right\}\times\left[{}^{A}_{B}R \right]\left\{ {}^{B}Q \right\}$$

General five-part acceleration equation

Position

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

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

Velocity

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

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

Acceleration

$$\left\{ {}^{A}A \right\}=\frac{d}{dt}\left(\left\{ {}^{A}V_{B} \right\}+\left[{}^{A}_{B}R \right]\left\{ {}^{B}V_{P} \right\}+\left\{ {}^{A}\omega_{B} \right\}\times\left[{}^{A}_{B}R \right]\left\{ {}^{B}P_{P} \right\}\right)$$

$$\left\{ {}^{A}A \right\}=\left\{ {}^{A}A_{B} \right\}+\left[{}^{A}_{B}R \right]\left\{ {}^{B}A_{P} \right\}+2\left\{ {}^{A}\omega_{B} \right\}\times\left[{}^{A}_{B}R \right]\left\{ {}^{B}V_{P} \right\}+\left\{ {}^{A}\alpha_{B} \right\}\times\left[{}^{A}_{B}R \right]\left\{ {}^{B}P_{P} \right\}+\left\{ {}^{A}\omega_{B} \right\}\times\left(\left\{ {}^{A}\omega_{B} \right\}\times\left[{}^{A}_{B}R \right]\left\{ {}^{B}P_{P} \right\}\right)$$

$$\{A\}=\{A_0\}+\{A_P\}+2\{\omega\}\times\{V\}+\{\alpha\}\times\{P\}+\{\omega\}\times\left(\{\omega\}\times\{P\}\right)$$

Rotational Acceleration

${}^{k}\{{}^{i}\alpha_{j}\}$ is the rotational acceleration of frame $\{j\}$ with respect to reference frame $\{i\}$, expressed in the basis (coordinates) of $\{k\}$. No angular orientation exists for which we can take the time derivative to find the angular velocity. Instead we use relative velocity equations.

Rotational Velocity

$$\{{}^{A}\omega_{C}\} = \{{}^{A}\omega_{B}\} + [{}^{A}_{B}R]\{{}^{B}\omega_{C}\}$$

Rotational Acceleration

$$\{{}^{A}\alpha_{C}\} = \frac{d}{dt}\left(\{{}^{A}\omega_{B}\} + [{}^{A}_{B}R]\{{}^{B}\omega_{C}\}\right) = \{{}^{A}\alpha_{B}\} + \frac{d}{dt}\left([{}^{A}_{B}R]\{{}^{B}\omega_{C}\}\right)$$

$$\{{}^{A}\alpha_{C}\} = \{{}^{A}\alpha_{B}\} + [{}^{A}_{B}R]\{{}^{B}\alpha_{C}\} + \{{}^{A}\omega_{B}\} \times [{}^{A}_{B}R]\{{}^{B}\omega_{C}\}$$

For vectors expressed in the local frame

$${}^{A}\{{}^{A}\alpha_{C}\} = [{}^{A}_{B}R]\,{}^{B}\{{}^{A}\alpha_{B}\} + [{}^{A}_{C}R]\,{}^{C}\{{}^{B}\alpha_{C}\} + [{}^{A}_{B}R]\,{}^{B}\{{}^{A}\omega_{B}\} \times [{}^{A}_{C}R]\,{}^{C}\{{}^{B}\omega_{C}\}$$

Combined Translational and Rotational Acceleration

$$\{\ddot{X}\} = \begin{Bmatrix}\{a\}\\ \{\alpha\}\end{Bmatrix} \quad \text{where} \quad \{a\} = {}^{0}\{{}^{0}A_{N}\}, \quad \{\alpha\} = {}^{0}\{{}^{0}\alpha_{N}\} \quad \text{both } a \text{ and } \alpha \text{ are (3x1) vectors}$$

Acceleration Example

Planar 2R robot – acceleration of end-effector frame with respect to {0}, also expressed in {0}.

1) Craig (2005) acceleration recursion

$$\{{}^{1}P_{2}\} = \begin{Bmatrix}L_1\\0\\0\end{Bmatrix} \qquad \{{}^{2}P_{3}\} = \begin{Bmatrix}L_2\\0\\0\end{Bmatrix} \qquad [{}^{2}_{1}R] = \begin{bmatrix}c_2 & s_2 & 0\\ -s_2 & c_2 & 0\\ 0 & 0 & 1\end{bmatrix} \qquad [{}^{3}_{2}R] = [I_3]$$

$$\{{}^{0}\omega_{0}\} = \begin{Bmatrix}0\\0\\0\end{Bmatrix} \qquad \{{}^{0}\alpha_{0}\} = \begin{Bmatrix}0\\0\\0\end{Bmatrix} \qquad \{{}^{0}a_{0}\} = \begin{Bmatrix}0\\0\\0\end{Bmatrix}$$

$$\{{}^{1}\omega_{1}\} = \begin{Bmatrix}0\\0\\\dot{\theta}_1\end{Bmatrix} \qquad \{{}^{1}\alpha_{1}\} = \begin{Bmatrix}0\\0\\\ddot{\theta}_1\end{Bmatrix} \qquad \{{}^{1}a_{1}\} = \begin{Bmatrix}0\\0\\0\end{Bmatrix}$$

$$\left\{{}^2\omega_2\right\}=\begin{Bmatrix}0\\0\\\dot\theta_1+\dot\theta_2\end{Bmatrix}\qquad \left\{{}^2\alpha_2\right\}=\begin{Bmatrix}0\\0\\\ddot\theta_1+\ddot\theta_2\end{Bmatrix}$$

$$\left\{{}^2a_2\right\}=\left[{}^2_1R\right]\left(\left\{{}^1\alpha_1\right\}\times\left\{{}^1P_2\right\}+\left\{{}^1\omega_1\right\}\times\left(\left\{{}^1\omega_1\right\}\times\left\{{}^1P_2\right\}\right)+\left\{{}^1a_1\right\}\right)$$

$$\left\{{}^2a_2\right\}=\begin{bmatrix}c_2 & s_2 & 0\\ -s_2 & c_2 & 0\\ 0 & 0 & 1\end{bmatrix}\begin{Bmatrix}-L_1\dot\theta_1^2\\ L_1\ddot\theta_1\\ 0\end{Bmatrix}=\begin{Bmatrix}-L_1c_2\dot\theta_1^2+L_1s_2\ddot\theta_1\\ L_1s_2\dot\theta_1^2+L_1c_2\ddot\theta_1\\ 0\end{Bmatrix}$$

$$\left\{{}^3\alpha_3\right\}=\left\{{}^2\alpha_2\right\}=\begin{Bmatrix}0\\0\\\ddot\theta_1+\ddot\theta_2\end{Bmatrix}$$

$$\left\{{}^3a_3\right\}=\left[{}^3_2R\right]\left(\left\{{}^2\alpha_2\right\}\times\left\{{}^2P_3\right\}+\left\{{}^2\omega_2\right\}\times\left(\left\{{}^2\omega_2\right\}\times\left\{{}^2P_3\right\}\right)+\left\{{}^2a_2\right\}\right)$$

$$\left\{{}^3a_3\right\}=\begin{Bmatrix}-L_2(\dot\theta_1+\dot\theta_2)^2-L_1c_2\dot\theta_1^2+L_1s_2\ddot\theta_1\\ L_2(\ddot\theta_1+\ddot\theta_2)+L_1s_2\dot\theta_1^2+L_1c_2\ddot\theta_1\\ 0\end{Bmatrix}$$

$$\left\{{}^0a_3\right\}=\left[{}^0_3R\right]\left\{{}^3a_3\right\}$$

$$\left\{{}^0a_3\right\}=\begin{Bmatrix}-L_1(s_1\ddot\theta_1+c_1\dot\theta_1^2)-L_2\left(s_{12}(\ddot\theta_1+\ddot\theta_2)+c_{12}(\dot\theta_1+\dot\theta_2)^2\right)\\ L_1(c_1\ddot\theta_1-s_1\dot\theta_1^2)+L_2\left(c_{12}(\ddot\theta_1+\ddot\theta_2)-s_{12}(\dot\theta_1+\dot\theta_2)^2\right)\\ 0\end{Bmatrix}$$

Rewrite for comparison to results in the next section.

$$\left\{{}^0a_3\right\}=\begin{bmatrix}-L_1s_1-L_2s_{12} & -L_2s_{12}\\ L_1c_1+L_2c_{12} & L_2c_{12}\end{bmatrix}\begin{Bmatrix}\ddot\theta_1\\ \ddot\theta_2\end{Bmatrix}+\begin{bmatrix}-L_1c_1\dot\theta_1 & -L_2c_{12}(\dot\theta_1+\dot\theta_2)\\ -L_1s_1\dot\theta_1 & -L_2s_{12}(\dot\theta_1+\dot\theta_2)\end{bmatrix}\begin{Bmatrix}\dot\theta_1\\ (\dot\theta_1+\dot\theta_2)\end{Bmatrix}$$

The second term above can be written as

$$\begin{bmatrix}-L_1c_1\dot\theta_1-L_2c_{12}(\dot\theta_1+\dot\theta_2) & -L_2c_{12}(\dot\theta_1+\dot\theta_2)\\ -L_1s_1\dot\theta_1-L_2s_{12}(\dot\theta_1+\dot\theta_2) & -L_2s_{12}(\dot\theta_1+\dot\theta_2)\end{bmatrix}\begin{Bmatrix}\dot\theta_1\\ \dot\theta_2\end{Bmatrix}$$

Acceleration Example (cont.)

Planar 2R robot – acceleration of the end-effector frame with respect to {0}, also expressed in {0}. An alternative method is presented here.

2) Differentiation of rate equation

$$\{\dot{X}\} = [J]\{\dot{\theta}\} \qquad\qquad \{\ddot{X}\} = [J]\{\ddot{\theta}\} + [\dot{J}]\{\dot{\theta}\}$$

Do in frame {0} – if in frame {2} or {3}, one must account for $\omega \times r$.

$$[{}^0J] = \begin{bmatrix} -L_1 s_1 - L_2 s_{12} & -L_2 s_{12} \\ L_1 c_1 + L_2 c_{12} & L_2 c_{12} \\ 1 & 1 \end{bmatrix}$$

$$[\dot{J}] = \frac{d[J]}{dt} = \left[\frac{dJ_{ij}}{dt}\right]$$

$$\frac{dJ_{ij}}{dt} = \frac{\partial J_{ij}}{\partial \theta_1}\frac{d\theta_1}{dt} + \frac{\partial J_{ij}}{\partial \theta_2}\frac{d\theta_2}{dt} + \cdots + \frac{\partial J_{ij}}{\partial \theta_N}\frac{d\theta_N}{dt} \qquad\qquad \frac{dJ_{ij}}{dt} = \sum_{k=1}^{N}\frac{\partial J_{ij}}{\partial \theta_k}\dot{\theta}_k$$

$$[{}^0\dot{J}] = \begin{bmatrix} -(L_1 c_1 + L_2 c_{12})\dot{\theta}_1 - L_2 c_{12}\dot{\theta}_2 & -L_2 c_{12}\dot{\theta}_1 - L_2 c_{12}\dot{\theta}_2 \\ -(L_1 s_1 + L_2 s_{12})\dot{\theta}_1 - L_2 s_{12}\dot{\theta}_2 & -L_2 s_{12}\dot{\theta}_1 - L_2 s_{12}\dot{\theta}_2 \\ 0 & 0 \end{bmatrix}$$

$$[{}^0\dot{J}] = \begin{bmatrix} -L_1 c_1\dot{\theta}_1 - L_2 c_{12}(\dot{\theta}_1 + \dot{\theta}_2) & -L_2 c_{12}(\dot{\theta}_1 + \dot{\theta}_2) \\ -L_1 s_1\dot{\theta}_1 - L_2 s_{12}(\dot{\theta}_1 + \dot{\theta}_2) & -L_2 s_{12}(\dot{\theta}_1 + \dot{\theta}_2) \\ 0 & 0 \end{bmatrix}$$

$$\{{}^0\ddot{X}\} = [{}^0J]\{\ddot{\theta}\} + [{}^0\dot{J}]\{\dot{\theta}\}$$

$$\{{}^0\ddot{X}\} = \begin{bmatrix} -L_1 s_1 - L_2 s_{12} & -L_2 s_{12} \\ L_1 c_1 + L_2 c_{12} & L_2 c_{12} \\ 1 & 1 \end{bmatrix}\begin{Bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{Bmatrix} + \begin{bmatrix} -L_1 c_1\dot{\theta}_1 - L_2 c_{12}(\dot{\theta}_1 + \dot{\theta}_2) & -L_2 c_{12}(\dot{\theta}_1 + \dot{\theta}_2) \\ -L_1 s_1\dot{\theta}_1 - L_2 s_{12}(\dot{\theta}_1 + \dot{\theta}_2) & -L_2 s_{12}(\dot{\theta}_1 + \dot{\theta}_2) \\ 0 & 0 \end{bmatrix}\begin{Bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{Bmatrix}$$

This yields the same result as before.

Uses for general acceleration equation

$$\{\dot{X}\} = [J]\{\dot{\theta}\}$$
$$\{\ddot{X}\} = [J]\{\ddot{\theta}\} + [\dot{J}]\{\dot{\theta}\}$$
$$\{\ddot{\theta}\} = [J]^{-1}\left(\{\ddot{X}\} - [\dot{J}]\{\dot{\theta}\}\right)$$

i. Forward Acceleration Kinematics Analysis – $\{\ddot{X}\} = [J]\{\ddot{\theta}\} + [\dot{J}]\{\dot{\theta}\}$ predicts the Cartesian accelerations $\{\ddot{X}\}$ given the joint rates and accelerations.

ii. Resolved Acceleration Control – like resolved rate, but acceleration is commanded instead of velocity. Solve $\{\ddot{\theta}\} = [J]^{-1}\left(\{\ddot{X}\} - [\dot{J}]\{\dot{\theta}\}\right)$ and double integrate to get the commanded joint angles.

iii. Dynamics equations – $\{\ddot{\theta}\}$ is required for the Newton/Euler dynamics recursion in the EE/ME 4290/5290 Supplement, Chapter 5. If acceleration is calculated via numerical differentiation, numerical instability can result, so the analytical approach $\{\ddot{\theta}\} = [J]^{-1}\left(\{\ddot{X}\} - [\dot{J}]\{\dot{\theta}\}\right)$ is better.

Now, if *inverse dynamics control* is being used in the *resolved-rate control algorithm* framework, assume $\{\dot{X}\}$ is constant and so $\{\ddot{X}\} = \{0\}$. In this case:

$$\{\ddot{\theta}\} = -[J]^{-1}[\dot{J}]\{\dot{\theta}\}$$

How can we find the time rate of change of the Jacobian matrix $[\dot{J}]$? See the previous page for a specific example.

11. Serial Robot Dynamics

Kinematics is the study of motion ***without*** regard to forces.

Dynamics is the study of motion ***with*** regard to forces. It is the study of the relationship between forces/torques and motion. Dynamics is composed of kinematics and kinetics.

a) Forward Dynamics (simulation) – given the actuator forces and torques, compute the resulting motion (this requires the solution of highly coupled, nonlinear ODEs): Given $\{\tau\}$, calculate $\{\theta\},\{\dot{\theta}\},\{\ddot{\theta}\}$ (all are N x 1 vectors).

b) Inverse Dynamics (control) – given the desired motion, calculate the actuator forces and torques (this linear algebraic solution is much more straight-forward than Forward Dynamics): Given $\{\theta\},\{\dot{\theta}\},\{\ddot{\theta}\}$, calculate $\{\tau\}$ (all N x 1 vectors).

Both problems require the N ***dynamic equations of motion***, one for each link, which are highly coupled and nonlinear. There are two basic methods for deriving the ***dynamic equations of motion***.

- Newton-Euler recursion (force balance, including inertial forces with D'Alembert's principle).
- Lagrange-Euler formulation (energy method).

Kinetics

Translational **Newton's Second Law**

Inertial force at center of mass

Rotational **Euler's Equation**

Inertial moment anywhere on body

The **kinematics** terms $\{a_{Ci}\},\{\omega_i\},\{\alpha_i\}$ must be moving with respect to an inertially-fixed frame. The frame of expression $\{k\}$ needn't be an inertially-fixed frame.

Assumptions

- serial robot
- rigid links
- ignore actuator dynamics
- no friction
- no joint or link flexibility

11.1 Inertia Tensor (Mass Distribution)

The inertia tensor is a spatial generalization of the planar scalar moment of inertia. Its units are mass times distance^2 (kg-m^2). The symmetric inertia tensor expressed at a given point A in the rigid body, relative to frame $\{A\}$ is:

$$ {}^{A}I = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{xz} & I_{yz} & I_{zz} \end{bmatrix} $$

Mass moments of inertia

$$ I_{xx} = \iiint_V \left(y^2 + z^2\right)\rho dv \quad I_{yy} = \iiint_V \left(x^2 + z^2\right)\rho dv \qquad I_{zz} = \iiint_V \left(x^2 + y^2\right)\rho dv $$

Mass products of inertia

$$ I_{xy} = \iiint_V xy\rho dv \qquad I_{xz} = \iiint_V xz\rho dv \qquad I_{yz} = \iiint_V yz\rho dv $$

Principal moments of inertia

A certain orientation of the reference frame $\{A\}$, the principal axes, yields zero products of inertia. The invariant eigenvalues of a general $[{}^{A}I]$ are the principal moments of inertia, and the eigenvectors are the principal axes.

More interesting facts regarding inertia tensors

1) If two axes of the reference frame form a plane of symmetry for the mass distribution, the products of inertia normal to this plane are zero.
2) Moments of inertia must be positive, products of inertia may be either sign.
3) The sum of the three moments of inertia are invariant under rotation transformations.

Parallel-axis theorem

We can obtain the mass moment of inertia tensor at any point $\{A\}$ if we know the inertia tensor at the center of mass $\{C\}$ (assuming these two frames have the same orientation). $P_C = \begin{Bmatrix} x_C & y_C & z_C \end{Bmatrix}^T$ is the vector giving the location of the center of mass $\{C\}$ from the origin of $\{A\}$.

Here are some example inertia tensor components using the parallel axis theorem.

$$ {}^{A}I_{zz} = {}^{C}I_{zz} + m\left(x_C^2 + y_C^2\right) $$

$$ {}^{A}I_{xy} = {}^{C}I_{xy} - mx_C y_C $$

Here is the entire inertia tensor expressed in vector-matrix form using the parallel axis theorem.

$$ \left[{}^{A}I\right] = \left[{}^{C}I\right] + m\left[\left\{P_C\right\}^T\left\{P_C\right\}\left[I_3\right] - \left\{P_C\right\}\left\{P_C\right\}^T\right] $$

11.2 Newton-Euler Recursive Algorithm

This is a recursive approach (Craig, 2005) based on free-body diagrams (FBDs) to determine the dynamics relationships link-by-link. Used numerically it can calculate the inverse dynamics solution efficiently.

Free-body diagram of link *i*

$\{f_i\}$ internal force exerted on link *i* by link *i*-1.

$\{n_i\}$ internal moment exerted on link *i* by link *i*-1.

Inertial loads Newton and Euler translational and rotational dynamics equations

Force balance

Moment balance (about CG_i) (using D'Alembert's principle, the inertial force is $-m\{a\}$).

Newton-Euler Recursive Algorithm Summary

This methods can be used to find the robot dynamics equations of motion. It can also be used to directly solve the inverse dynamics problem numerically. The summary of equations below, from Craig (2005), assume an all revolute-joint manipulator (prismatic joint dynamics have different equations).

Outward iteration for kinematics $i: 0 \to N-1$

(without regard for frames of expression, for clarity)

Velocities and accelerations (kinematics)

$$\{\omega_{i+1}\} = \{\omega_i\} + \{\hat{Z}_{i+1}\}\dot{\theta}_{i+1}$$
$$\{\alpha_{i+1}\} = \{\alpha_i\} + \{\omega_i\} \times \{\hat{Z}_{i+1}\}\dot{\theta}_{i+1} + \{\hat{Z}_{i+1}\}\ddot{\theta}_{i+1}$$
$$\{a_{i+1}\} = \{a_i\} + \{\alpha_i\} \times \{{}^{i}P_{i+1}\} + \{\omega_i\} \times \left(\{\omega_i\} \times \{{}^{i}P_{i+1}\}\right)$$
$$\{a_{Ci+1}\} = \{a_{i+1}\} + \{\alpha_{i+1}\} \times \{{}^{i+1}P_{Ci+1}\} + \{\omega_{i+1}\} \times \left(\{\omega_{i+1}\} \times \{{}^{i+1}P_{Ci+1}\}\right)$$

Inertial loading (kinetics)

$$\{F_{i+1}\} = m_{i+1}\{a_{Ci+1}\}$$
$$\{N_{i+1}\} = \left[{}^{C}I\right]\{\alpha_{i+1}\} + \{\omega_{i+1}\} \times \left[{}^{C}I\right]\{\omega_{i+1}\}$$

Inward iteration for kinetics $i: N \to 1$

(without regard for frames of expression, for clarity)

Internal forces and moments

$$\{f_i\} = \{f_{i+1}\} + \{F_i\}$$
$$\{n_i\} = \{n_{i+1}\} + \{{}^{i}P_{Ci}\} \times \{F_i\} + \{{}^{i}P_{i+1}\} \times \{f_{i+1}\} + \{N_i\}$$

Externally applied joint torques

$$\tau_i = \{n_i\} \cdot \{Z_i\}$$

Inclusion of gravity forces

$$\{{}^{0}a_0\} = \{g\}$$

This is equivalent to a fictitious upward acceleration of $1g$ of the robot base, which accounts for the downward acceleration due to gravity (i.e. this conveniently includes the weight of all links).

11.3 Lagrange-Euler Energy Method

This is an alternative method to find the robot dynamics equations of motion. It requires only translational and rotational link velocities, not accelerations. The ***Lagrangian*** is formed from the **kinetic energy *k*** and **potential energy *u*** of the robot system.

$$L = k - u = k(\Theta, \dot{\Theta}) - u(\Theta)$$

$$k_i = \frac{1}{2} m_i \{V_{Ci}\}^T \{V_{Ci}\} + \frac{1}{2} \{\omega_i\}^T \left[{}^{Ci}I_i \right] \{\omega_i\}$$

$$u_i = u_{REF} - m_i \left\{ {}^0 g \right\} \bullet \left\{ {}^0 P_{Ci} \right\}$$

$$k(\Theta, \dot{\Theta}) = \sum_{i=1}^{N} k_i \qquad\qquad u(\Theta) = \sum_{i=1}^{N} u_i$$

Note

$$k(\Theta, \dot{\Theta}) = \frac{1}{2} \left\{ \dot{\Theta} \right\}^T \left[M(\Theta) \right] \left\{ \dot{\Theta} \right\}$$ where $\left[M(\Theta) \right]$ is the manipulator mass matrix.

Dynamic equations of motion

These are found for each active joint from the following expression involving the Lagrangian, joint variable, and actuator torque. Perform this equation *N* times, once for each joint variable *i*, to yield *N* independent dynamics equations of motion.

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_i} \right) - \frac{\partial L}{\partial \theta_i} = \tau_i$$

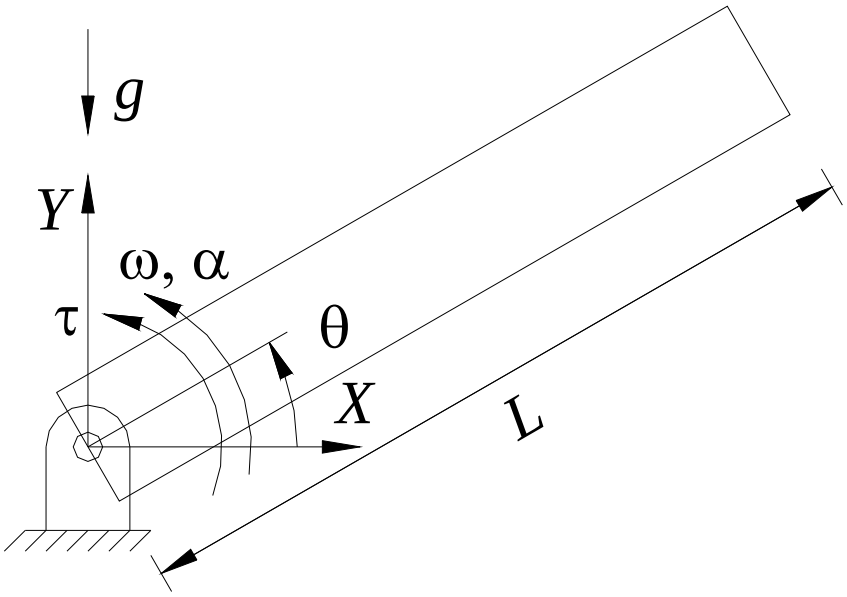
This expression may be rewritten using $L = k(\Theta, \dot{\Theta}) - u(\Theta)$.

$$\frac{d}{dt}\left(\frac{\partial k}{\partial \dot{\theta}_i} \right) - \frac{\partial k}{\partial \theta_i} + \frac{\partial u}{\partial \theta_i} = \tau_i$$

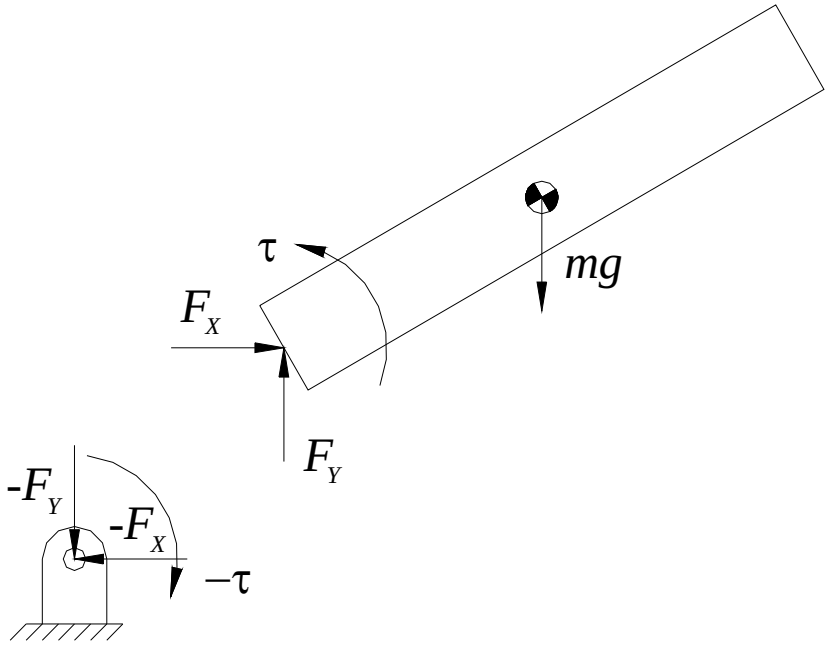
11.4 Simple Dynamics Example

Derive the ***dynamic equation of motion*** for a planar one-link 1R mechanism by three methods:

1) Sophomore dynamics method – FBD, force and moment dynamics balance.
2) Newton-Euler recursion.
3) Lagrange-Euler formulation.



1) Sophomore methods - FBD, force balance *Free-Body Diagram*



1) Sophomore methods – FBD, force balance (cont.)

$$\sum F_x = ma_{Cx}$$

a) $${}^0F_x = m\left(-\frac{L}{2}\alpha s\theta - \frac{L}{2}\omega^2 c\theta\right)$$

$${}^0F_x = -\frac{mL}{2}(\ddot{\theta}s\theta + \dot{\theta}^2 c\theta)$$

$$\sum F_y = ma_{Cy}$$

b) $${}^0F_y - mg = m\left(\frac{L}{2}\alpha c\theta - \frac{L}{2}\omega^2 s\theta\right)$$

$${}^0F_y = mg + m\frac{L}{2}(\ddot{\theta}c\theta - \dot{\theta}^2 s\theta)$$

c) $$\sum M_{z_0} = {}^0I\ddot{\theta} = \tau - \frac{mL}{2}gc\theta$$

$$\tau = {}^0I\ddot{\theta} + \frac{mL}{2}gc\theta$$

where $${}^0I = {}^CI + md^2 = {}^CI_{zz} + \frac{mL^2}{4}$$

Simple Dynamics Example (cont.)

2) Newton-Euler recursion

Outward iteration $i = 0$ (the only iteration)

$$\left\{{}^0P_1\right\} = \begin{Bmatrix} 0 \\ 0 \\ 0 \end{Bmatrix} \qquad \left\{{}^1P_{C1}\right\} = \begin{Bmatrix} \frac{L}{2} \\ 0 \\ 0 \end{Bmatrix} \qquad \left\{{}^1_0R\right\} = \begin{bmatrix} c\theta & s\theta & 0 \\ -s\theta & c\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \left\{{}^{C_1}I_1\right\} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}$$

$$\left\{{}^0\omega_0\right\} = \begin{Bmatrix} 0 \\ 0 \\ 0 \end{Bmatrix} \qquad \left\{{}^0\alpha_0\right\} = \begin{Bmatrix} 0 \\ 0 \\ 0 \end{Bmatrix} \qquad \left\{{}^0a_0\right\} = \begin{Bmatrix} 0 \\ g \\ 0 \end{Bmatrix} \text{ to account for gravity}$$

$$\left\{{}^1a_1\right\} = \left[{}^1_0R\right]\left(\left\{{}^0\alpha_0\right\} \times \left\{{}^0P_1\right\} + \left\{{}^0\omega_0\right\} \times \left(\left\{{}^0\omega_0\right\} \times \left\{{}^0P_1\right\}\right) + \left\{{}^0a_0\right\}\right)$$

$$\left\{{}^1a_1\right\} = \begin{bmatrix} c\theta & s\theta & 0 \\ -s\theta & c\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} 0 \\ g \\ 0 \end{Bmatrix} = \begin{Bmatrix} gs\theta \\ gc\theta \\ 0 \end{Bmatrix}$$

$$\left\{{}^1a_{C1}\right\} = \left\{{}^1\alpha_1\right\} \times \left\{{}^1P_{C1}\right\} + \left\{{}^1\omega_1\right\} \times \left(\left\{{}^1\omega_1\right\} \times \left\{{}^1P_{C1}\right\}\right) + \left\{{}^1a_1\right\} = \begin{Bmatrix} gs\theta - \frac{L}{2}\dot{\theta}^2 \\ gc\theta + \frac{L}{2}\ddot{\theta} \\ 0 \end{Bmatrix}$$

$$\left\{{}^1F_1\right\} = m\left\{{}^1a_{C1}\right\} = m\begin{Bmatrix} gs\theta - \frac{L}{2}\dot{\theta}^2 \\ gc\theta + \frac{L}{2}\ddot{\theta} \\ 0 \end{Bmatrix} \qquad \left\{{}^1N_1\right\} = \left[{}^{C_1}I_1\right]\left\{{}^1\alpha_1\right\} + \left\{{}^1\omega_1\right\} \times \left[{}^{C_1}I_1\right]\left\{{}^1\omega_1\right\} = \begin{Bmatrix} 0 \\ 0 \\ I_{zz}\ddot{\theta} \end{Bmatrix}$$

Inward iteration $i = 1$ (the only iteration)

$$\left\{{}^2 f_2\right\} = \begin{Bmatrix} 0 \\ 0 \\ 0 \end{Bmatrix} \qquad \left\{{}^2 n_2\right\} = \begin{Bmatrix} 0 \\ 0 \\ 0 \end{Bmatrix}$$

$$\left\{{}^1 f_1\right\} = \left[{}^1_2 R\right]\left\{{}^2 f_2\right\} + \left\{{}^1 F_1\right\} = m\begin{Bmatrix} gs\theta - \frac{L}{2}\dot{\theta}^2 \\ gc\theta + \frac{L}{2}\ddot{\theta} \\ 0 \end{Bmatrix}$$

$$\left\{{}^1 n_1\right\} = \left[{}^1_2 R\right]\left\{{}^2 n_2\right\} + \left\{{}^1 P_{C_1}\right\} \times \left\{{}^1 F_1\right\} + \left\{{}^1 P_2\right\} \times \left[{}^1_2 R\right]\left\{{}^2 f_2\right\} + \left\{{}^1 N_1\right\}$$

$$\left\{{}^1 n_1\right\} = \begin{Bmatrix} 0 \\ 0 \\ \frac{mL}{2}\left(gc\theta + \frac{L}{2}\ddot{\theta}\right) \end{Bmatrix} + \begin{Bmatrix} 0 \\ 0 \\ I_{zz}\ddot{\theta} \end{Bmatrix}$$

$$\tau = \left\{{}^1 n_1\right\} \bullet \left\{{}^1 \hat{Z}_1\right\}$$

$$\tau = \left(I_{zz} + \frac{mL^2}{4}\right)\ddot{\theta} + \frac{mL}{2} gc\theta$$

Check with the sophomore dynamics method above.

$$\left\{{}^0 f_1\right\} = \left[{}^0_1 R\right]\left\{{}^1 f_1\right\} = \begin{bmatrix} c\theta & -s\theta & 0 \\ s\theta & c\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} m\begin{Bmatrix} gs\theta - \frac{L}{2}\dot{\theta}^2 \\ gc\theta + \frac{L}{2}\ddot{\theta} \\ 0 \end{Bmatrix} = m\begin{Bmatrix} -\frac{L}{2}\dot{\theta}^2 c\theta - \frac{L}{2}\ddot{\theta} s\theta \\ -\frac{L}{2}\dot{\theta}^2 s\theta + \frac{L}{2}\ddot{\theta} c\theta + g \\ 0 \end{Bmatrix}$$

3) Lagrange-Euler formulation

$$\left\{{}^{1}\omega_{1}\right\}=\begin{Bmatrix}0\\0\\\dot{\theta}\end{Bmatrix} \qquad \left[{}^{C_1}I_{1}\right]=\begin{bmatrix}I_{xx}&0&0\\0&I_{yy}&0\\0&0&I_{zz}\end{bmatrix}$$

Kinetic and potential energy are scalars invariant with frame of expression – write *k* in {1} and *u* in {0}.

$$\left\{{}^{1}v_{C_1}\right\}=\begin{Bmatrix}0\\\frac{L}{2}\dot{\theta}\\0\end{Bmatrix} \qquad \left\{{}^{1}\omega_{1}\right\}\bullet\left[{}^{C_1}I_{1}\right]\left\{{}^{1}\omega_{1}\right\}=\begin{Bmatrix}0\\0\\\dot{\theta}\end{Bmatrix}\bullet\begin{Bmatrix}0\\0\\I_{zz}\dot{\theta}\end{Bmatrix}$$

$$k=\frac{1}{2}\frac{mL^2}{4}\dot{\theta}^2+\frac{1}{2}I_{zz}\dot{\theta}^2 \qquad u=0-m\begin{Bmatrix}0\\-g\\0\end{Bmatrix}\bullet\begin{Bmatrix}\frac{L}{2}c\theta\\\frac{L}{2}s\theta\\0\end{Bmatrix}=mg\frac{L}{2}s\theta$$

$$L=k-u=\frac{1}{2}\dot{\theta}^2\left(I_{zz}+\frac{mL^2}{4}\right)-mg\frac{L}{2}s\theta$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial\dot{\theta}}\right)-\frac{\partial L}{\partial\theta}=\tau$$

$$\frac{\partial L}{\partial\dot{\theta}}=\dot{\theta}\left(I_{zz}+\frac{mL^2}{4}\right) \qquad \frac{d}{dt}\left(\frac{\partial L}{\partial\dot{\theta}}\right)=\ddot{\theta}\left(I_{zz}+\frac{mL^2}{4}\right) \qquad \frac{\partial L}{\partial\theta}=-mg\frac{L}{2}c\theta$$

$$\tau=\left(I_{zz}+\frac{mL^2}{4}\right)\ddot{\theta}+\frac{mL}{2}gc\theta$$

This result agrees with the sophomore and Newton-Euler recursion dynamics methods above to solve the same problem.

11.5 Structure of Manipulator Dynamics Equations

State Space Equation

$$\{\tau\} = [M(\Theta)]\{\ddot{\Theta}\} + \{V(\Theta,\dot{\Theta})\} + \{G(\Theta)\}$$

$[M(\Theta)]$ $N \times N$ mass matrix; symmetric and positive definite

$\{V(\Theta,\dot{\Theta})\}$ $N \times 1$ vector of Coriolis and centripetal terms

$\{G(\Theta)\}$ $N \times 1$ vector of gravity terms

Configuration Space Equation

$$\{\tau\} = [M(\Theta)]\{\ddot{\Theta}\} + [B(\Theta)]\{\dot{\Theta}\dot{\Theta}\} + [C(\Theta)]\{\dot{\Theta}^2\} + \{G(\Theta)\}$$

$[M(\Theta)]$ $N \times N$ mass matrix; symmetric and positive definite

$[B(\Theta)]$ $N \times \frac{N(N-1)}{2}$ Coriolis matrix

$\{\dot{\Theta}\dot{\Theta}\}$ $\frac{N(N-1)}{2} \times 1$ $\{\dot{\theta}_1\dot{\theta}_2 \quad \dot{\theta}_1\dot{\theta}_3 \quad \cdots \quad \dot{\theta}_{N-1}\dot{\theta}_N\}^T$

$[C(\Theta)]$ $N \times N$ centripetal matrix

$\{\dot{\Theta}^2\}$ $N \times 1$ $\{\dot{\theta}_1^2 \quad \dot{\theta}_2^2 \quad \cdots \quad \dot{\theta}_N^2\}^T$

$\{G(\Theta)\}$ $N \times 1$ vector of gravity terms

Cartesian State Space Equation

$$\{F\} = [M_x(\Theta)]\{\ddot{X}\} + \{V_x(\Theta,\dot{\Theta})\} + \{G_x(\Theta)\}$$

$[M_x(\Theta)]$ $N \times N$ Cartesian mass matrix; symmetric and positive definite

$\{V_x(\Theta,\dot{\Theta})\}$ $N \times 1$ vector of Cartesian Coriolis and centripetal terms

$\{G_x(\Theta)\}$ $N \times 1$ vector of gravity terms in Cartesian space

where

$$[M_x(\Theta)] = [J^{-T}][M(\Theta)][J^{-1}]$$

$$\{V_x(\Theta,\dot{\Theta})\} = [J^{-T}]\left(\{V(\Theta,\dot{\Theta})\} - [M(\Theta)][J^{-1}][\dot{J}]\{\dot{\theta}\}\right)$$

$$\{G_x(\Theta)\} = [J^{-T}]\{G(\Theta)\}$$

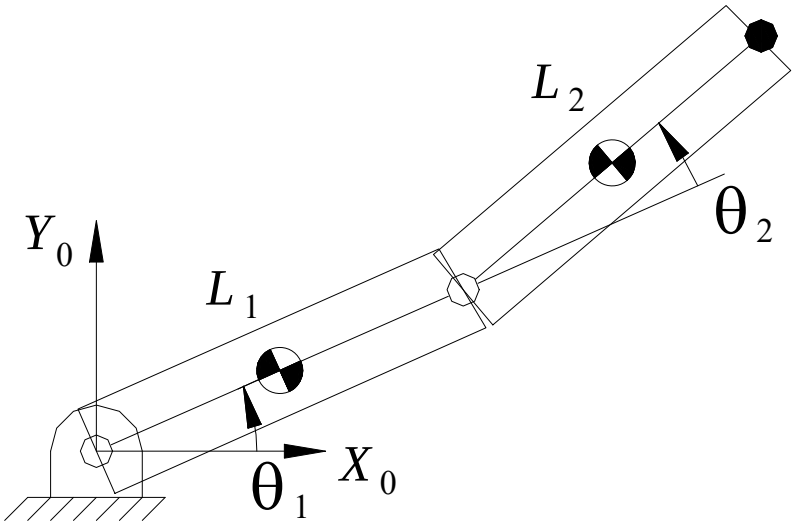
note

$$\{\dot{X}\} = [J]\{\dot{\Theta}\}$$

$$\{\ddot{X}\} = [\dot{J}]\{\dot{\Theta}\} + [J]\{\ddot{\Theta}\}$$

11.6 Robot Dynamics Example

Here we summarize the **dynamics equations of motion** for the two-link planar 2R robot when each link is modeled as a homogeneous rectangular solid of dimensions L_i, h_i, w_i of mass m_i.



Newton-Euler recursion with outward kinematics and inertial calculations, followed by inward kinetics balances yields.

$$\tau_2 = \left(I_{zz2} + \frac{m_2 L_1 L_2 c_2}{2} + \frac{m_2 L_2^2}{4} \right) \ddot{\theta}_1 + \left(I_{zz2} + \frac{m_2 L_2^2}{4} \right) \ddot{\theta}_2 + \left(\frac{m_2 L_1 L_2 s_2}{2} \right) \dot{\theta}_1^2 + \left(\frac{m_2 L_2 c_{12}}{2} \right) g$$

$$\tau_1 = \left(I_{zz1} + I_{zz2} + \frac{m_1 L_1^2}{4} + m_2 L_1^2 + m_2 L_1 L_2 c_2 + \frac{m_2 L_2^2}{4} \right) \ddot{\theta}_1 + \left(I_{zz2} + \frac{m_2 L_1 L_2 c_2}{2} + \frac{m_2 L_2^2}{4} \right) \ddot{\theta}_2$$
$$+ \left(-\frac{m_2 L_1 L_2 s_2}{2} \right) \dot{\theta}_2^2 + \left(-m_2 L_1 L_2 s_2 \right) \dot{\theta}_1 \dot{\theta}_2 + \left(\frac{m_1 L_1 c_1}{2} + m_2 L_1 c_1 + \frac{m_2 L_2 c_{12}}{2} \right) g$$

where $I_{zzi} = \frac{m_i}{12}(L_i^2 + h_i^2)$.

These dynamics equations of motion are very complicated – imagine how much worse these equations will be for a spatial 6-axis robot such as the PUMA industrial robot.

State Space Representation

For the planar 2R robot, the dynamics equations of motion from the previous page are expressed in state-space form below:

$$\{\tau\} = [M(\Theta)]\{\ddot{\Theta}\} + \{V(\Theta,\dot{\Theta})\} + \{G(\Theta)\} \qquad \{\tau\} = \begin{Bmatrix} \tau_1 \\ \tau_2 \end{Bmatrix} \qquad \{\ddot{\Theta}\} = \begin{Bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{Bmatrix}$$

$$[M(\Theta)] = \begin{bmatrix} I_{zz1} + I_{zz2} + \dfrac{m_1 L_1^2}{4} + m_2 L_1^2 + m_2 L_1 L_2 c_2 + \dfrac{m_2 L_2^2}{4} & I_{zz2} + \dfrac{m_2 L_1 L_2 c_2}{2} + \dfrac{m_2 L_2^2}{4} \\ I_{zz2} + \dfrac{m_2 L_1 L_2 c_2}{2} + \dfrac{m_2 L_2^2}{4} & I_{zz2} + \dfrac{m_2 L_2^2}{4} \end{bmatrix}$$

$$\{V(\Theta,\dot{\Theta})\} = \begin{Bmatrix} \left(-\dfrac{m_2 L_1 L_2 s_2}{2}\right)\dot{\theta}_2^2 + \left(-m_2 L_1 L_2 s_2\right)\dot{\theta}_1\dot{\theta}_2 \\ \left(\dfrac{m_2 L_1 L_2 s_2}{2}\right)\dot{\theta}_1^2 \end{Bmatrix}$$

$$\{G(\Theta)\} = \begin{Bmatrix} \dfrac{m_1 L_1 c_1}{2} + m_2 L_1 c_1 + \dfrac{m_2 L_2 c_{12}}{2} \\ \dfrac{m_2 L_2 c_{12}}{2} \end{Bmatrix} g$$

Configuration Space Representation

For the planar 2R robot, the dynamics equations of motion from the previous page are expressed in configuration-space form below.

$$\{\tau\} = [M(\Theta)]\{\ddot{\Theta}\} + [B(\Theta)]\{\dot{\Theta}\dot{\Theta}\} + [C(\Theta)]\{\dot{\Theta}^2\} + \{G(\Theta)\}$$

$[M(\Theta)], \{G(\Theta)\}$ same as above

$$[B(\Theta)]\{\dot{\Theta}\dot{\Theta}\} = \begin{Bmatrix} -m_2 L_1 L_2 s_2 \\ 0 \end{Bmatrix}\dot{\theta}_1\dot{\theta}_2$$

$$[C(\Theta)]\{\dot{\Theta}^2\} = \begin{bmatrix} 0 & -\dfrac{m_2 L_1 L_2 s_2}{2} \\ \dfrac{m_2 L_1 L_2 s_2}{2} & 0 \end{bmatrix}\begin{Bmatrix} \dot{\theta}_1^2 \\ \dot{\theta}_2^2 \end{Bmatrix}$$

Numerical Dynamics Example

For the planar 2R robot whose **dynamics equations of motion** were presented analytically above, here we calculate the required torques (i.e. solve the inverse dynamics problem) to provide the commanded motion at every time step in a resolved-rate control scheme. Identical results are obtained by both analytical equations and numerical Newton/Euler recursion.

Given

$L_1 = 1.0\ m$
$L_2 = 0.5\ m$

Both links are solid steel with mass density $\rho = 7806\ kg/m^3$ and both have width and thickness dimensions $w = t = 5\ cm$. The revolute joints are assumed to be perfect, connecting the links at their very edges (not physically possible, just a simplified model).

The initial robot configuration is

$$\{\Theta\} = \begin{Bmatrix} \theta_1 \\ \theta_2 \end{Bmatrix} = \begin{Bmatrix} 10^\circ \\ 90^\circ \end{Bmatrix}$$

The constant commanded Cartesian velocity is

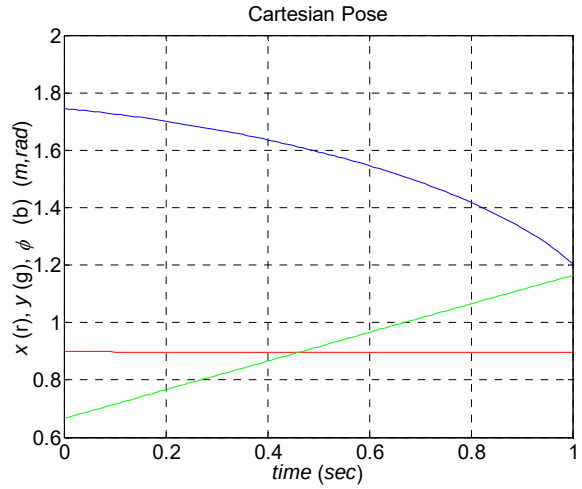
$$^0\{\dot{X}\} = {}^0\begin{Bmatrix} \dot{x} \\ \dot{y} \end{Bmatrix} = {}^0\begin{Bmatrix} 0 \\ 0.5 \end{Bmatrix} \qquad (m/s)$$

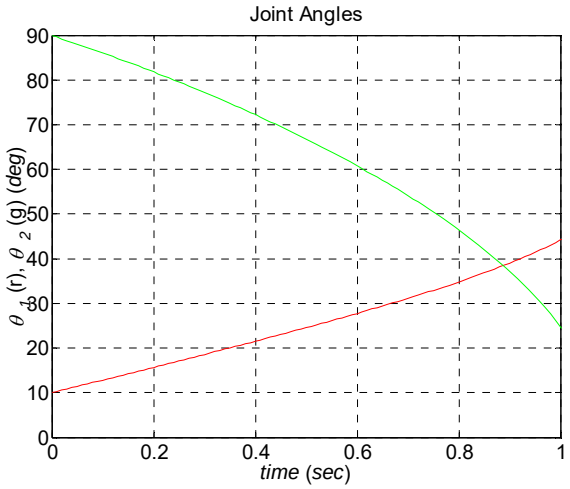
The dynamics equations require the relative joint accelerations $\ddot{\theta}$; how do we find these? (See the last page of the Acceleration Kinematics Section 10 in this EE/ME 4290/5290 NotesBook Supplement: $\{\ddot{\theta}\} = [J^{-1}]\left(\{\ddot{X}\} - [\dot{J}]\{\dot{\theta}\}\right)$. In this example, $\{\ddot{X}\} = \{0 \quad 0\}^T$.)

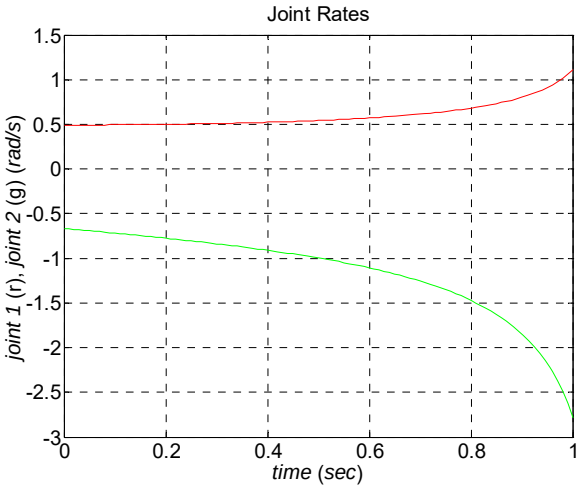
Simulate motion for 1 *sec*, with a control time step of 0.01 *sec*. The plots for various variables of interest (joint angles, joint rates, joint accelerations, joint torques, and Cartesian pose) for this problem are given on the following page.

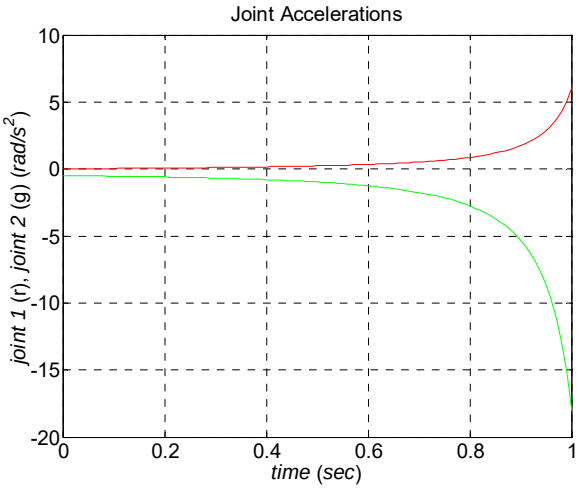
In the last plot, note that the robot travels 0.5 *m* in the Y_0 direction in 1 *sec* (which agrees with the constant commanded rate of 0.5 *m*/s). The robot does not move in *X*; ϕ must move to compensate for the pure *Y* motion, but we cannot control ϕ independently with only two-dof. The first three plots are kinematics terms related to the resolved-rate control scheme; they are inputs to the inverse dynamics problem. The joint torques are calculated by the numerical recursive Newton-Euler inverse dynamics algorithm. These are the joint torques necessary to move this robot's inertia in the commanded manner. Notice from the joint angles, joint rates, joint accelerations, and joint torques plots that the robot is approaching the $\theta_2 = 0$ singularity towards the end of this simulated motion.

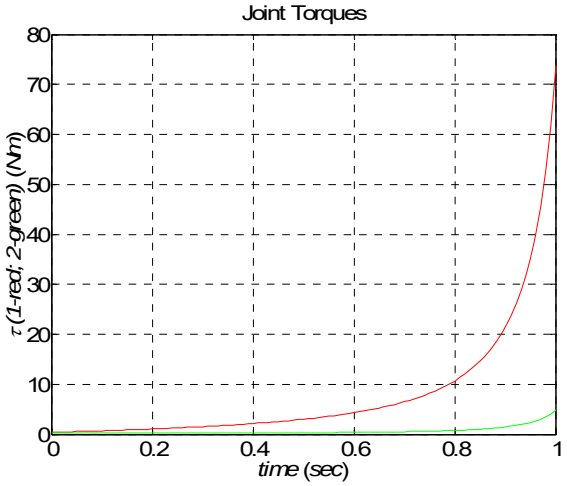
Numerical Dynamics Example: Plots



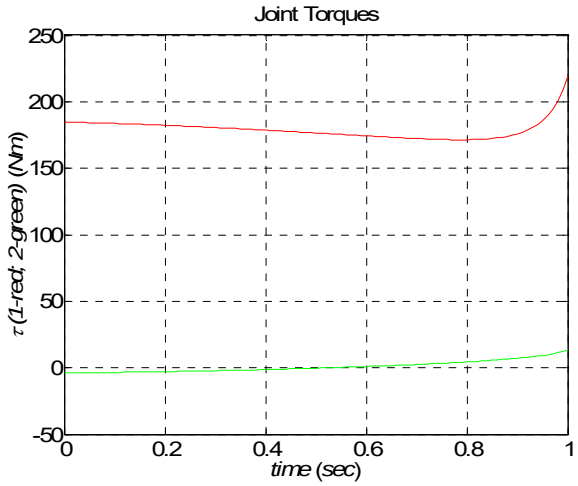








Dynamics Results Ignoring Gravity

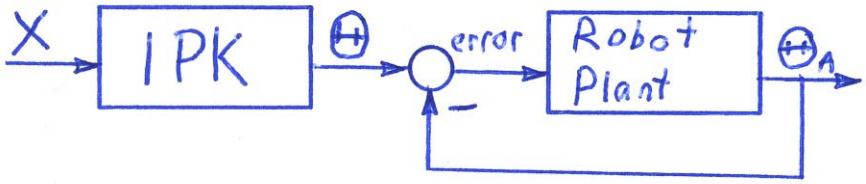


Dynamics Results Including Gravity

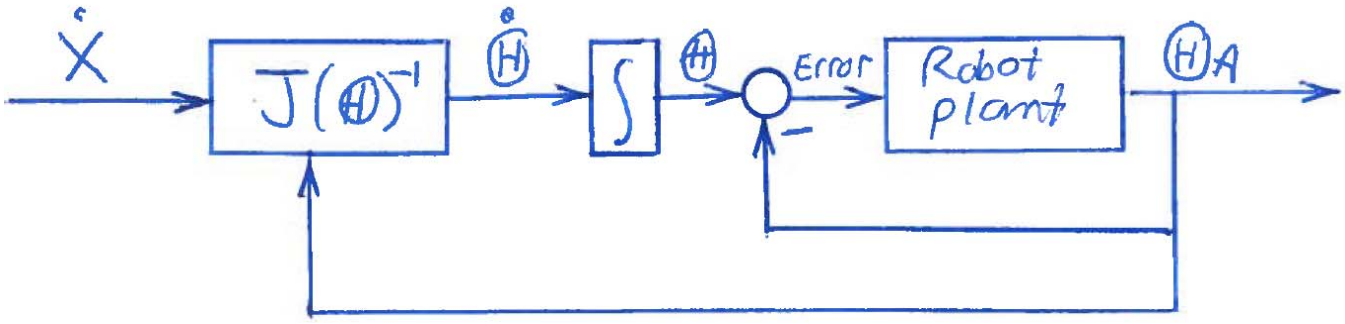
12. Robot Control Architectures

With the robotic kinematics and dynamics we have learned and simulated, we can simulate robot control using various popular robot control architectures.

12.1 Inverse Pose Control Architecture



12.2 Inverse Velocity (Resolved-Rate) Control Architecture



12.3 Inverse Dynamics Control Architecture

The inverse dynamics method is also called the computed torque control method, and it is also called the feedback linearization control method.

In Chapter 4 we learned that the robot dynamics equations of motion are highly coupled and nonlinear. Here they are in matrix/vector form.

$$\{\tau\} = [M(\Theta)]\{\ddot{\Theta}\} + \{V(\Theta,\dot{\Theta})\} + \{F(\Theta,\dot{\Theta})\} + \{G(\Theta)\}$$

$\{\tau\}$ vector of applied joint torques

$[M(\Theta)]$ $N \times N$ mass matrix; symmetric and positive definite

$\{V(\Theta,\dot{\Theta})\}$ $N \times 1$ vector of Coriolis and centripetal terms

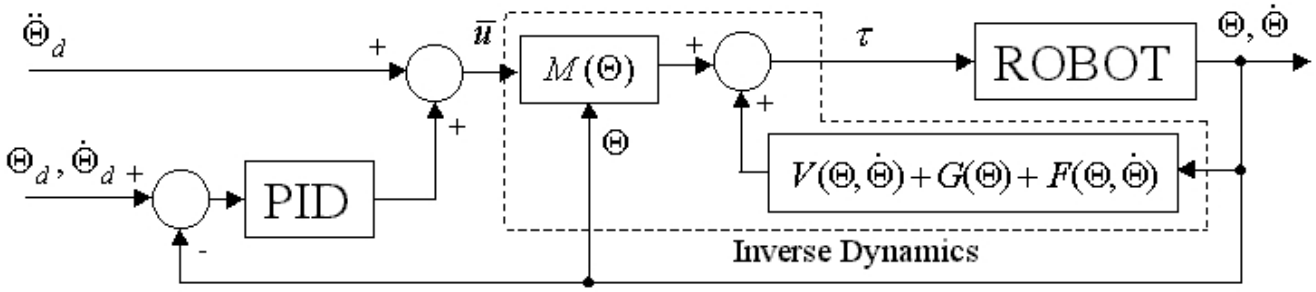
$\{F(\Theta,\dot{\Theta})\}$ $N \times 1$ vector of friction torques

$\{G(\Theta)\}$ $N \times 1$ vector of gravity terms

$\{\Theta\},\{\dot{\Theta}\},\{\ddot{\Theta}\}$ $N \times 1$ joint angles, rates, and accelerations vectors

Computed Torque Control Architecture

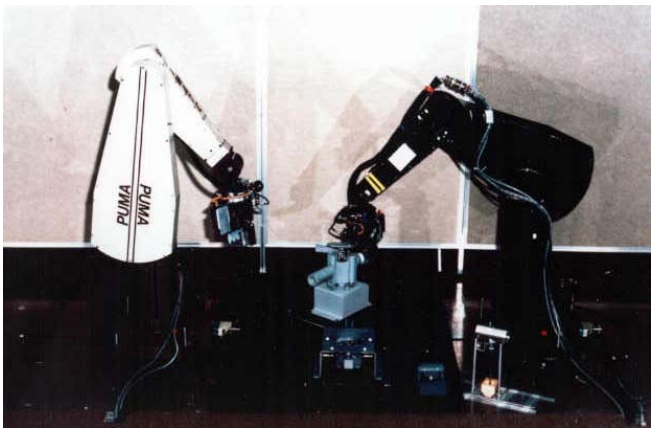
The computed-torque control method is based on canceling the dynamics effects by using the inverse dynamics method, in order to linearize the robot system for standard controller methods, such as PID control.



12.4 NASA Langley Telerobotics Resolved-Rate Control Architecture

Below is pictured the Intelligent Systems Research Laboratory (ISRL) at NASA Langley Research Center, circa 1992.

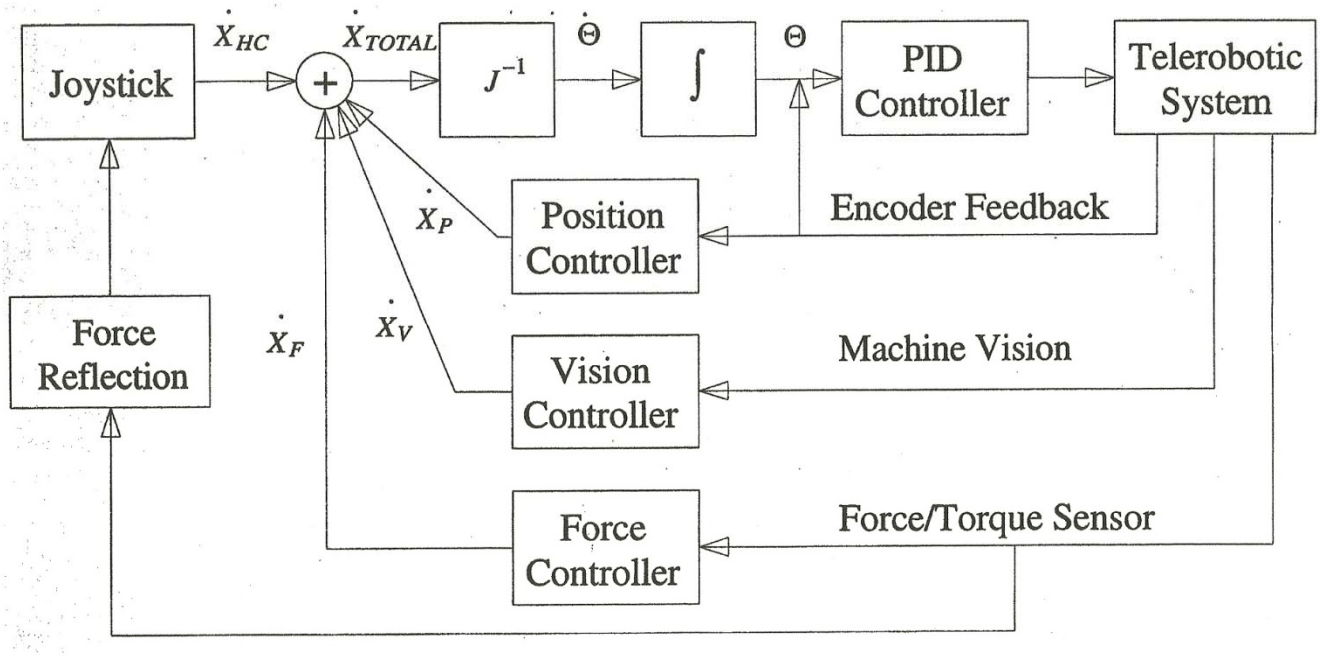
Force-Reflecting Operator Station



Dual PUMA Robots with Space Station Task

NASA Langley Research Center Telerobotics Laboratory (ISRL)

Below is shown the overall resolved-rate-based robot control architecture developed at NASA Langley Research Center in the Automation Technology Branch, with Dr. Bob on the team, in the early 1990s. This controller architecture was novel and is still unique in robot control.



NASA Langley Research Center Telerobotic Controller Architecture

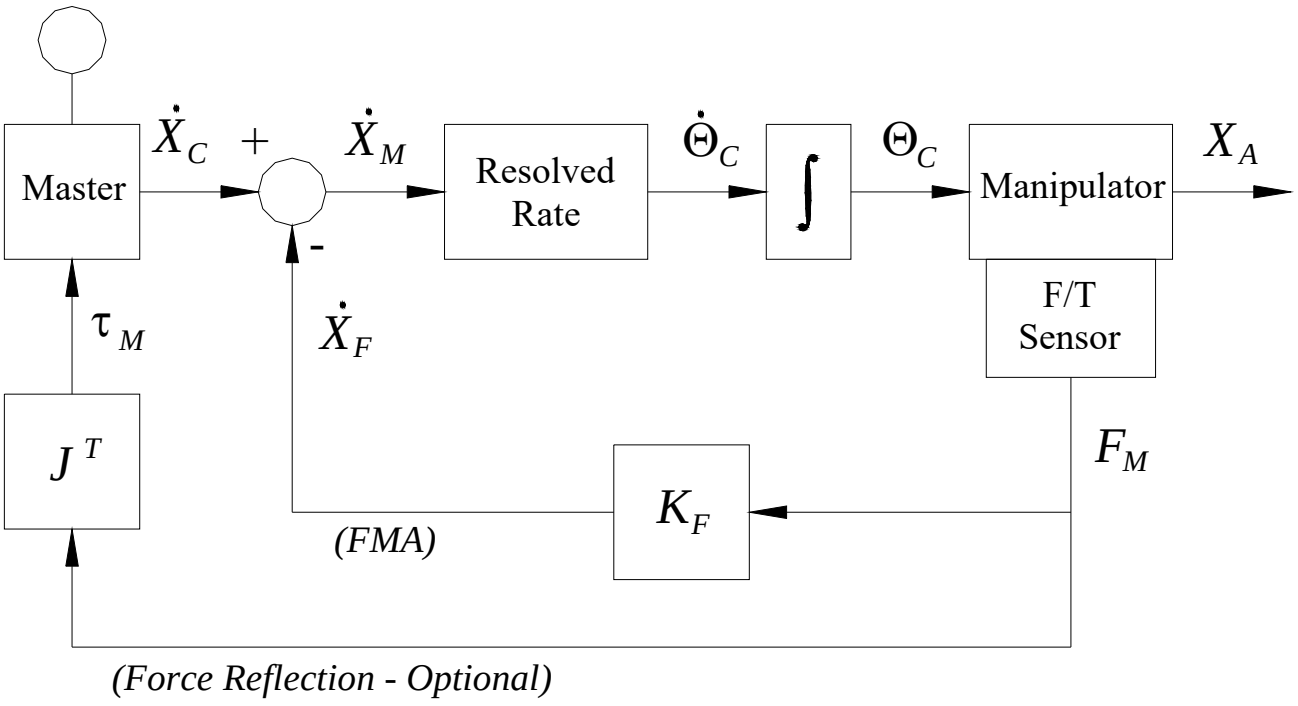
In the top, forward control path above we see that resolved-rate (inverse velocity) is the basic control method, as opposed to inverse pose control. Being a linear problem, the resolved-rate method has several attractive benefits over IPK control including the ability to add velocity control inputs as vectors, which is not possible with IPK (due to orientation).

The total Cartesian velocity vector (translational and rotational) input determining the robot motion is the sum of four inputs, from the hand controller, position controller, vision controller, and force/torque controller. The displacement of the hand controller is interpreted as a velocity input, for both the translational and rotational modes. Often the joystick will have a hardware or software spring return-to-center, including deadbands, to more easily turn off velocities after certain directions are finished for the time being. The position, vision, and force controllers (details hidden) yield a Cartesian velocity that will move the robot in the direction that the control algorithm determines in each case.

Not all control modes need be enabled at all times, but all can be combined is desired. This could lead to controller contention. If the joystick enables force reflection, this provides an extra sensory feedback back to the human operator which has been proven very effective in many experiments in different research labs. Today we call this force-reflection aspect of telerobotics haptics.

12.5 Naturally-Transitioning Rate-to-Force Controller Architecture

Resolved-rate control automatically (naturally) changes to force control when the robot end-effector enters into contact with the environment. A force/torque (F/T) sensor and force/moment accommodation (FMA) algorithm is required to accomplish this. After contact, the constant velocity command becomes a constant force command. In addition, if there is a human teleoperator with haptic interface, the force/moment that the human applies to the interface becomes proportional to the force/moment that the robot exerts on the environment at contact. This works and feels great in real-world applications, providing *telepresence*.



This control architecture was implemented by Dr. Bob and teams at NASA Langley Research Center and Wright-Patterson AFB[6].

[6] R.L. Williams II, J.M. Henry, M.A. Murphy, and D.W. Repperger, 1999, Naturally-Transitioning Rate-to-Force Control in Free and Constrained Motion, ASME Journal of Dynamic Systems, Measurement, and Control, Trans. ASME, 121(3): 425-432.

12.6 Single Joint Control

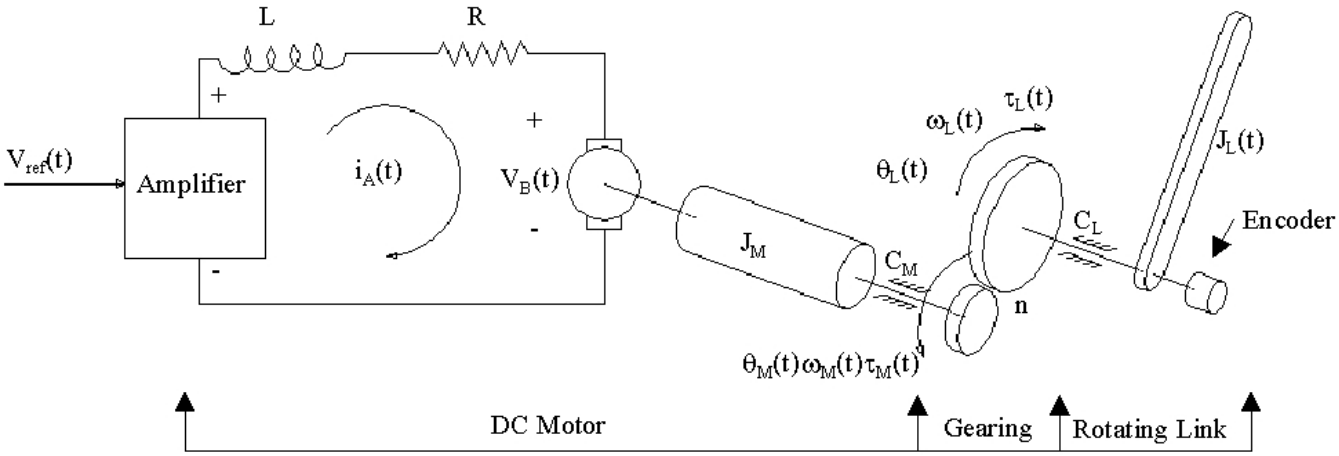
Every controller architecture we've considered requires linearized independent (but simultaneous) single joint angle control, presented in this section.

Manipulator dynamics is extremely complicated considering the number of terms. We can easily use symbolic MATLAB to crank out the terms but they go on for pages and their structure is difficult to understand. In this case, numerical Newton-Euler recursion a'la Craig (2005, Chapter 6) is useful to get around the need for analytical expressions.

How is manipulator dynamics done in industry for control purposes? In almost all cases it is **ignored**. How is this possible? (Large gear ratios tend to decouple the dynamic coupling in motion of one link on its neighbors – we will see this in modeling soon.) The vast majority of multi-axis industrial robots are controlled via *linearized, independent single joint control*. So, robot control in industry is generally accomplished by using N independent (but simultaneous) linearized joint controllers, where N is the number of joint space freedoms of the robot.

We now will briefly discuss single joint control. We will focus on a common system, the armature-controlled DC servomotor driving a single robot joint / gear train / link combination, shown below. This requires dynamics, but it is not coupled nor nonlinear.

Open-Loop System Diagram



$V(t)$	armature voltage	$\tau_M(t)$	generated motor torque	$\tau_L(t)$	load torque
L	armature inductance	$\theta_M(t)$	motor shaft angle	$\theta_L(t)$	load shaft angle
R	armature resistance	$\omega_M(t)$	motor shaft velocity	$\omega_L(t)$	load shaft velocity
$i_A(t)$	armature current	J_M	lumped motor inertia	$J_L(t)$	total load inertia
$v_B(t)$	back emf voltage	C_M	motor viscous damping	C_L	load viscous damping
		n	gear ratio		

Closed-Loop Feedback High-level Control Diagram

High-level Computer Control Diagram

Simplifying Assumptions

- rigid motor, load shafts
- $n >> 1$ (the large gear ratio n reduces speed and increases torque)

Real-world vs. model characteristics

Real World	Our Model (simplified)
nonlinear	linearized
multiple-input, multiple output (MIMO)	single-input, single output (SISO)
coupled	decoupled
time-varying load inertia	treat as a disturbance; the large gear ratio diminishes this problem
hysteresis, backlash, stiction, Coulomb friction	ignore
discrete and continuous	continuous for control design

Single Joint/Link System Modeling

We must derive all linear ordinary differential equations (ODEs) describing the dynamics of an armature-controlled DC servomotor driving a single robot joint / link. This is an electromechanical system, including a gear train. We must perform modeling, simulation, and control. The system diagram was shown earlier in this section.

Armature Circuit Diagram

Electrical Model Kirchoff's voltage law

(1)

Electromechanical coupling

- The generated motor torque is proportional to the armature current

(2)

- The back-emf voltage is proportional to the motor shaft angular velocity

(3)

($K_T = K_B$ can be shown)

Mechanical Model

Euler's Equation (the rotational form of Newton's Second Law):

Free-body diagrams

Load

(4)

Motor

(5)

where $\tau_{LR}(t)$ is the load torque reflected to the motor shaft.

Gear ratio ***n***

(6)

Substituting (6) into (5) yields (7)

(7)

Reflect load inertia and damping to motor shaft

Substitute into (4)

(8)

Substitute (8) into (7) to eliminate τ_L

(9)

Combine terms

(10)

define

$J_E = J_M + \frac{J_L}{n^2}$ effective inertia, total reflected to motor shaft

$C_E = C_M + \frac{C_L}{n^2}$ effective damping, total reflected to motor shaft

Final Mechanical Model (ODE)

(11)

This is a second-order ODE in θ_M. Can also be written as first-order ODE in ω_M.

(12)

For common industrial robots, $n >> 1$ so $\frac{1}{n^2}$ is small; therefore we can choose a nominal J_L and assume it is constant without much error in control.

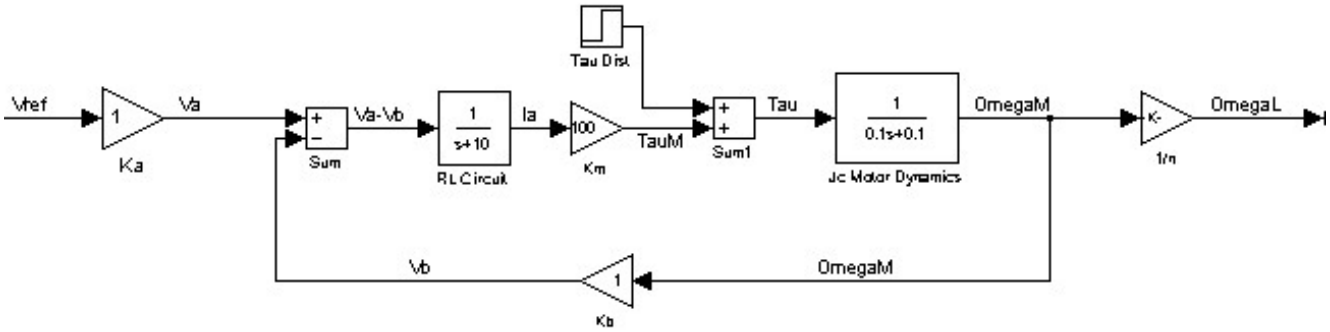
For example, the NASA 8-axis ARMII robot downstairs has $n = 200$ with harmonic gearing. This is why we can ignore the time-varying robot load inertia and design decoupled independent linear joint controllers. The gear-reduced load inertia variation is then treated as a disturbance to the single joint controller.

An alternative is to reflect the motor inertia and damping to the load shaft as shown below, but we will use the first case above.

$$(J_L + n^2 J_M)\ddot{\theta}_L + (C_L + n^2 C_M)\dot{\theta}_L = \tau_L$$

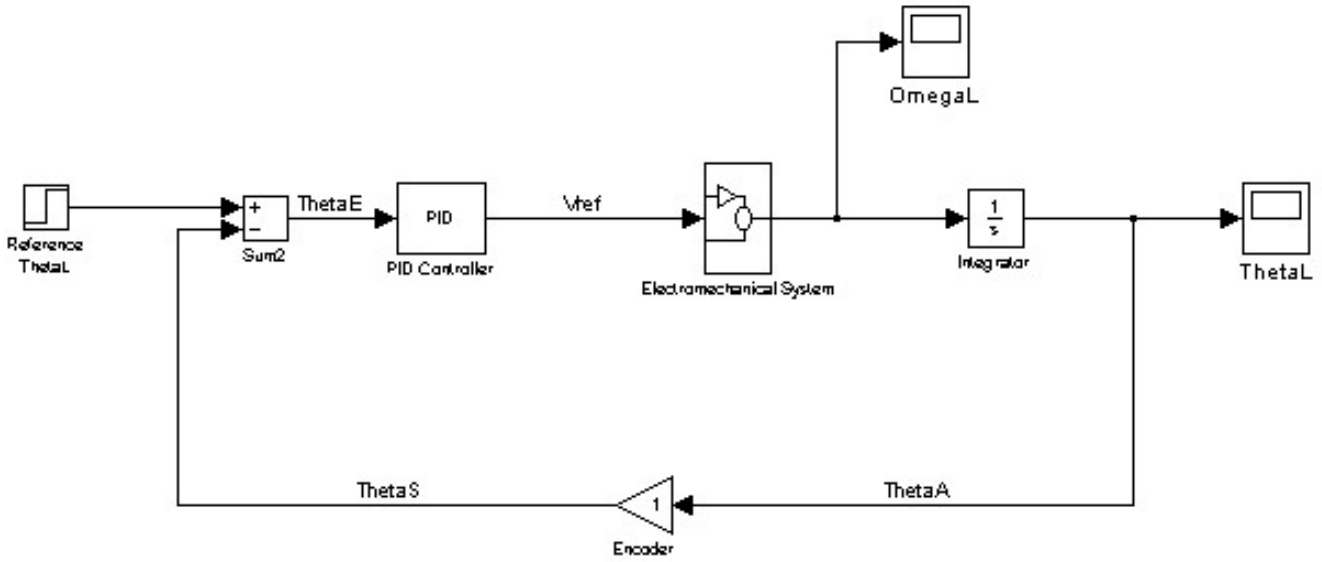
Open-Loop Block Diagram

Use Laplace transforms on different ODE pieces of the system model and then connect the components together in blocks connected by arrows representing variables. A transfer function goes inside each block representing component dynamics. This block diagram is shown below, in MATLAB/Simulink implementation, including a disturbance torque.



Closed-Loop Feedback Control Diagram

Assuming perfect encoder sensor for angular position feedback, we include block for the single-joint PID controller. The MATLAB/Simulink implementation is shown below.



PID Controller Design

PID Controller characteristics

PID stands for:

Controller transfer function:

Each term does this:

Use PID w/ approximate derivate in **Simulink** – numerical differentiation can lead to errors and numerical instability and thus it is to be avoided if possible.

Trial and Error PID Design

Start with the P gain value – start low and increase P to get the desired time nature of response. Then add the D gain value term for stability. Add the I gain value to reduce steady-state error. Always use Simulink to simulate control and dynamics response of the single joint/link system for different PID choices; compare various cases and select a suitable controller in simulation.

A better approach is to perform analytical design for PID controllers using classical control theory, such as in ME 3012.

Controller Performance Criteria

- Stability
- Rise Time
- Peak Time
- Percent Overshoot
- Settling Time
- Steady-state error

These performance criteria provide a rational basis for choosing a suitable controller, at least in theory and simulation. The real world always provides some additional challenges (noise, modeling errors, nonlinearities, calibration, backlash, etc.).

Include Gravity as a Disturbance Torque

Unmodeled, unexpected disturbances are modeled as disturbances in control systems. It is convenient to do so at the motor torque level in the block diagram.

Gravity is known and expected. However, in single joint control we can include the effect of gravity as a disturbance. First let us model the gravity effect for a single joint. Lump all outboard links, joints, and motors as a single rigid body.

- test with original PID gains
- redesign new PID gains with gravity disturbance considered