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

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

## 2. SERIAL MANIPULATOR KINEMATICS

2.7 **Inverse Pose Kinematics** ................................................................. 4  
2.7.2 Planar 3R Robot IPK Solution ....................................................... 4  
2.7.4 Pieper's Solution Method ............................................................... 12  
2.7.5 Spatial 8R NASA AAI ARMII Robot IPK Solution ....................... 14

## 4. PARALLEL ROBOTS ........................................................................ 19

4.2 **Planar 2-DOF Five-Bar Parallel Robot** ......................................... 19  
4.2.3 Acceleration Kinematics ................................................................. 19  
4.2.4 Inverse Dynamics .................................................................... 22  
4.3 **Intersection of Two Circles** ......................................................... 29  
4.4 **Planar 3-RPR Manipulator** ............................................................ 30  
4.4.1 Planar 3-RPR Manipulator Inverse Pose Kinematics ................. 30  
4.4.2 Planar 3-RPR Manipulator Forward Pose Kinematics .............. 32  
4.4.3 Planar 3-RPR Manipulator Velocity Kinematics ......................... 34  
4.5 **Newton-Raphson Method** ............................................................ 38  
4.6 **Parallel Manipulator Workspace** .................................................. 42  
4.7 **NIST RoboCrane Cable-Suspended Parallel Robot** .................... 43  
4.8 **Delta Parallel Robot Inverse and Forward Kinematics** ............. 53  
4.9 **Stewart Platform** ..................................................................... 54

## 5. MANIPULATOR DYNAMICS ............................................................. 55

5.1 **Inertia Tensor (Mass Distribution)** ............................................. 56  
5.2 **Newton-Euler Recursive Algorithm** ............................................ 57  
5.3 **Lagrange-Euler Energy Method** .................................................. 59  
5.4 **Simple Dynamics Example** ......................................................... 60  
5.5 **Structure of Manipulator Dynamics Equations** ......................... 65  
5.6 **Robot Dynamics Example** .......................................................... 66

## 6. ROBOT CONTROL ARCHITECTURES ............................................. 70

6.1 **Inverse Pose Control Architecture** ............................................ 70  
6.2 **Inverse Velocity (Resolved-Rate) Control Architecture** ............ 70  
6.3 **Inverse Dynamics Control Architecture** .................................... 71  
6.4 **NASA Langley Telerobotics Resolved-Rate Control Architecture** 72  
6.5 **Naturally-Transitioning Rate-to-Force Controller Architecture** ..... 74  
6.6 **Single Joint Control** ................................................................. 75
2. Serial Manipulator Kinematics

2.7 Inverse Pose Kinematics

2.7.2 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 \( \phi \), 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:

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

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

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:

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

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 \( \theta_1 \) solution of the Inverse Pose Kinematics solution for the planar 3R robot in the 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:

\[ \cos \psi = \frac{E}{\sqrt{E^2 + F^2}} \quad \sin \psi = \frac{F}{\sqrt{E^2 + F^2}} \]

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 $\theta$ 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 $\text{atan2}$ must be used in the above expression for $\psi$.

There are two solutions for $\theta$, 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 $\theta$ to exist, we must have

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

If this condition is violated for the planar 3R robot, this means that the given input pose $x_3, y_3, \phi$ 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 $\theta_2$ terms first. Here are the original $XY$ equations:

\[
\begin{align*}
  x_3 &= L_1 \cos \theta_1 + L_2 \cos(\theta_1 + \theta_2) \\
  y_3 &= L_1 \sin \theta_1 + L_2 \sin(\theta_1 + \theta_2)
\end{align*}
\]

Squaring and adding these equations as-is yields:

\[
\begin{align*}
  x_3^2 &= L_1^2 \cos^2 \theta_1 + 2L_1L_2 \cos \theta_1 \cos(\theta_1 + \theta_2) + L_2^2 \cos^2(\theta_1 + \theta_2) \\
  y_3^2 &= L_1^2 \sin^2 \theta_1 + 2L_1L_2 \sin \theta_1 \sin(\theta_1 + \theta_2) + L_2^2 \sin^2(\theta_1 + \theta_2)
\end{align*}
\]

\[
\begin{align*}
  x_3^2 + y_3^2 &= L_1^2 + L_2^2 + 2L_1L_2 [\cos \theta_1 \cos(\theta_1 + \theta_2) + \sin \theta_1 \sin(\theta_1 + \theta_2)]
\end{align*}
\]

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

\[
\begin{align*}
  x_3^2 + y_3^2 &= L_1^2 + L_2^2 + 2L_1L_2 \cos(\theta_1 - (\theta_1 + \theta_2)) \\
  &= L_1^2 + L_2^2 + 2L_1L_2 \cos(-\theta_2) \\
  &= L_1^2 + L_2^2 + 2L_1L_2 \cos \theta_2
\end{align*}
\]

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 $\theta_2$ 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 ± square root in the $\sin \theta_2$ term.
Knowing $\theta_2$ in this second alternative IPK solution method, we must return to the original $XY$ equations and solve for $\theta_1$.

\[
\begin{align*}
x_3 &= L_1 \cos \theta_1 + L_2 \cos(\theta_1 + \theta_2) = L_1 \cos \theta_1 + L_2 (\cos \theta_1 \cos \theta_2 - \sin \theta_1 \sin \theta_2) \\
y_3 &= L_1 \sin \theta_1 + L_2 \sin(\theta_1 + \theta_2) = L_1 \sin \theta_1 + L_2 (\sin \theta_1 \cos \theta_2 + \cos \theta_1 \sin \theta_2)
\end{align*}
\]

\[
\begin{align*}
x_3 &= (L_1 + L_2 \cos \theta_2) \cos \theta_1 - (L_2 \sin \theta_2) \sin \theta_1 \\
y_3 &= (L_1 + L_2 \cos \theta_2) \sin \theta_1 + (L_2 \sin \theta_2) \cos \theta_1
\end{align*}
\]

\[
\begin{align*}
x_3 &= k_1 \cos \theta_1 - k_2 \sin \theta_1 & \text{where} & \quad k_1 &= L_1 + L_2 \cos \theta_2 \\
y_3 &= k_1 \sin \theta_1 + k_2 \cos \theta_1 & \quad k_2 &= L_2 \sin \theta_2
\end{align*}
\]

Now, with $\theta_2$ 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 $\theta_1$ and will yield identical results, one unique $\theta_1$ for each $\theta_2$ 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).

\[
\begin{align*}
k_1 &= r \cos \gamma \\
k_2 &= r \sin \gamma & \text{where} & \quad r &= +\sqrt{k_1^2 + k_2^2} \\
\gamma &= \tan^{-1}\left[\frac{k_2}{k_1}\right]
\end{align*}
\]

We only use the positive square root term in $r$ and the quadrant-specific inverse tangent function $\text{atan2}$ must be used in the above expression for $\gamma$. 
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 $\theta_1$, one for each $\theta_2$ value.

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

$$
\theta_1 = \arctan \left( \frac{y_3}{x_3} \right) - \gamma
$$

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 $\theta_1$ (and leads $\theta_1$ 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 $\theta_1$ was able to cancel out of the equations and leave only $\theta_2$ 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 $\theta_2$ and does not depend on $\theta_1$ or $\theta_3$. 
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 $0^6T$, calculate $(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)$.

$0^6T = f(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)$

$0^6T = \begin{bmatrix}
0^6R(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6) & \{0^6P(\theta_1, \theta_2, \theta_3)\} \\
0 & 0 & 0 & 0 & 1
\end{bmatrix}$

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

1) Translational equations: Given $\{0^6P\}$, calculate $(\theta_1, \theta_2, \theta_3)$ values (4 sets).
   3 independent equations, 3 unknowns.

2) Rotational equations: Given $0^6R$, 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.

$0^6R(\theta_4, \theta_5, \theta_6) = [0^6R(\theta_1, \theta_2, \theta_3)]^T [0^6R]$

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.

\[
\begin{bmatrix}
0^T_6
\end{bmatrix} = \begin{bmatrix}
0^T_1 \\
0^T_2 \\
0^T_3 \\
0^T_4 \\
0^T_5 \\
0^T_6
\end{bmatrix} = \begin{bmatrix}
1^T_2(\theta_1) \\
2^T_3(\theta_2) \\
3^T_4(\theta_3) \\
4^T_5(\theta_4) \\
5^T_6(\theta_5) \\
6^T_6(\theta_6)
\end{bmatrix}
\]

The left-hand-side \(0^T_6\) is a set of valid given numbers, the input to the IPK problem. Inverting the first matrix on the left yields:

\[
\begin{bmatrix}
0^T_1
\end{bmatrix}^{-1} = \begin{bmatrix}
1^T_2(\theta_1) \\
2^T_3(\theta_2) \\
3^T_4(\theta_3) \\
4^T_5(\theta_4) \\
5^T_6(\theta_5) \\
6^T_6(\theta_6)
\end{bmatrix}
\]

from which we can solve for \(\theta_1\) and \(\theta_3\). Now we invert the homogeneous transformation matrix that combines the first three matrices in the original equation.

\[
\begin{bmatrix}
0^T_2
\end{bmatrix}^{-1} = \begin{bmatrix}
1^T_2(\theta_1) \\
2^T_3(\theta_2) \\
3^T_4(\theta_3) \\
4^T_5(\theta_4) \\
5^T_6(\theta_5) \\
6^T_6(\theta_6)
\end{bmatrix}
\]

We can solve for \(\theta_2\) with \(\theta_1\) and \(\theta_3\) already known. This completes the solution for the arm angle joints \((\theta_1, \theta_2, \theta_3)\), possible because the position vector \(\{0^TP_6\}\) 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.

\[
\begin{bmatrix}
0^T_4
\end{bmatrix}^{-1} = \begin{bmatrix}
1^T_2(\theta_1) \\
2^T_3(\theta_2) \\
3^T_4(\theta_3) \\
4^T_5(\theta_4) \\
5^T_6(\theta_5) \\
6^T_6(\theta_6)
\end{bmatrix}
\]

No we can isolate and solve for \(\theta_4\), \(\theta_5\), and \(\theta_6\) in turn.
2.7.5 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 $[^0S]T$ to achieve the commanded $[^H]T$.

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 Williams¹.

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

**Reduced problem statement**

Given $[^B]P = \{x_5, y_5, z_5\}^T$, calculate $(\theta_1, \theta_2, \theta_4)$, 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 $[^B]P = [^B]P_5$ 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)*:

$$[^B]P_5 = \begin{bmatrix} x_5 \\ y_5 \\ z_5 \end{bmatrix} = \begin{bmatrix} -c_1(d_3s_2 + d_5s_2s_{24}) \\ -s_1(d_3s_2 + d_5s_2s_{24}) \\ d_2 + d_5c_2 + d_6c_{24} \end{bmatrix},$$

where

$$c_{24} = \cos(\theta_2 + \theta_4)$$

$$s_{24} = \sin(\theta_2 + \theta_4)$$

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

Solution Process

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

\[ \frac{\theta_1}{180^\circ} \text{ is also a valid solution} \]

2. Since $\theta_1$ is now known (two values), we can modify the $Y$ and $Z$ equations.

\[ Y = \frac{-y_5}{s_1} \]

where:

\[ Z = z_5 - d_8 \]

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

Square and add to eliminate $\theta_4$:

The result is one equation in one unknown $\theta_2$.

\[ E \cos \theta_2 + F \sin \theta_2 + G = 0 \]

where

\[ E = 2Zd_3 \]

\[ F = 2Yd_3 \]

\[ G = -Y^2 - Z^2 - d_3^2 + d_5^2 \]

We can solve this equation for $\theta_2$ by using the **Tangent Half-Angle Substitution**. We presented this back in the Inverse Pose Solution of the planar 3R robot; we solve for $\theta_2$ (in that section, it was for $\theta_1$).

Solve for $\theta_2$ by inverting the original Tangent Half-Angle definition.
Two $\theta_2$ solutions result from the $\pm$ in the quadratic formula; both are correct (there are multiple solutions – elbow up and elbow down). To find $\theta_4$, return to original (arranged) translational equations.

\[
d_s s_{24} = Y - d_s s_2
\]
\[
d_s c_{24} = Z - d_s c_2
\]

Find the unique $\theta_4$ for each $\theta_2$ value (use the quadrant-specific \texttt{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 $\theta_1$. For each $\theta_1$, there are two values for $\theta_2$; for each valid $(\theta_1, \theta_2)$, there is a unique $\theta_4$. 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.

<table>
<thead>
<tr>
<th>$i$</th>
<th>$\theta_1$</th>
<th>$\theta_2$</th>
<th>$\theta_3$</th>
<th>$\theta_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\theta_1$</td>
<td>$\theta_{21}$</td>
<td>0</td>
<td>$\theta_4$</td>
</tr>
<tr>
<td>2</td>
<td>$\theta_1$</td>
<td>$\theta_{22}$</td>
<td>0</td>
<td>$-\theta_4$</td>
</tr>
<tr>
<td>3</td>
<td>$\theta_1 + 180^\circ$</td>
<td>$-\theta_{22}$</td>
<td>0</td>
<td>$\theta_4$</td>
</tr>
<tr>
<td>4</td>
<td>$\theta_1 + 180^\circ$</td>
<td>$-\theta_{21}$</td>
<td>0</td>
<td>$-\theta_4$</td>
</tr>
</tbody>
</table>

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 $\{^b P_5\}$. You can also calculate the associated $\left[^b R_4\right]$. All of these resulting rotation matrices should be different (why?).
8R ARMII Robot Translational Inverse Pose Kinematics Example

Given \( \{^8P_5\} \), calculate \((\theta_1, \theta_2, \theta_4)_i \) for \( i = 1, 2, 3, 4 \).

\[
\{^8P_5\} = \begin{Bmatrix}
x_5 \\
y_5 \\
z_5 \\
\end{Bmatrix} = \begin{Bmatrix}
-0.6572 \\
-0.1159 \\
1.6952 \\
\end{Bmatrix}
\]

**Answers (deg)**

<table>
<thead>
<tr>
<th>( i )</th>
<th>( \theta_1 )</th>
<th>( \theta_2 )</th>
<th>( \theta_3 )</th>
<th>( \theta_4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>20</td>
<td>0</td>
<td>30</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>46.6</td>
<td>0</td>
<td>-30</td>
</tr>
<tr>
<td>3</td>
<td>190</td>
<td>-46.6</td>
<td>0</td>
<td>30</td>
</tr>
<tr>
<td>4</td>
<td>190</td>
<td>-20</td>
<td>0</td>
<td>-30</td>
</tr>
</tbody>
</table>

**Check** all solution sets via Forward Pose Kinematics to ensure all yield the correct, commanded \( \{^8P_5\} \).
4. Parallel Robots

4.2 Planar 2-dof Five-Bar Parallel Robot

4.2.3 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.

\( \dot{\alpha}_{i} = \alpha_{i} \) (i = 2,3,4,5) is the absolute angular acceleration of link \( i \). \( \dot{\alpha}_{1} = 0 \) since link 1, the fixed ground link, cannot rotate.

The velocity equations developed in the Velocity Kinematics section are the starting point for deriving the acceleration equations.

4.2.3.1 Forward Acceleration Kinematics

Given \( r_{1}, \theta_{1}, r_{2}, r_{3}, r_{4}, r_{5} \); angles \( \theta_{2}, \theta_{3}, \theta_{4}, \theta_{5} \); angular rates \( \dot{\theta}_{2}, \dot{\theta}_{3}, \dot{\theta}_{4}, \dot{\theta}_{5} \); plus actuator angular accelerations \( \ddot{\theta}_{2}, \ddot{\theta}_{3} \)

Find desired end-effector translational acceleration \( \dot{B} = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} \), 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:

\[
\begin{align*}
-r_{2} \dot{\theta}_{2} s_{2} - r_{3} \dot{\theta}_{3} s_{3} &= -r_{5} \dot{\theta}_{5} s_{5} - r_{4} \dot{\theta}_{4} s_{4} \\
r_{2} \dot{\theta}_{2} c_{2} + r_{3} \dot{\theta}_{3} c_{3} &= r_{5} \dot{\theta}_{5} c_{5} + r_{4} \dot{\theta}_{4} c_{4}
\end{align*}
\]

\[
\begin{align*}
-r_{2} \ddot{\theta}_{2} c_{2} - r_{3} \ddot{\theta}_{3} c_{3} - r_{5} \dot{\theta}_{5}^{2} s_{5} - r_{4} \dot{\theta}_{4}^{2} s_{4} &= -r_{5} \dot{\theta}_{5} c_{5} - r_{4} \dot{\theta}_{4} c_{4} \\
r_{2} \ddot{\theta}_{2} c_{2} - r_{3} \ddot{\theta}_{3} c_{3} - r_{5} \dot{\theta}_{5}^{2} s_{5} - r_{4} \dot{\theta}_{4}^{2} s_{4} &= r_{5} \ddot{\theta}_{5} c_{5} - r_{4} \ddot{\theta}_{4} c_{4} - 2 r_{5} \dot{\theta}_{5} \dot{\theta}_{5} c_{5} - 2 r_{4} \dot{\theta}_{4} \dot{\theta}_{4} c_{4}
\end{align*}
\]

Since all links are rigid (i.e. no links are changing lengths), all \( r_{i} = 0 \) 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 \( \ddot{\theta}_{4} \):
The solution of the above matrix-vector set of equations is:

\[
\begin{pmatrix}
    a & b \\
    d & e
\end{pmatrix}
\begin{pmatrix}
    \dot{\theta}_3 \\
    \dot{\theta}_4
\end{pmatrix}
= \begin{pmatrix}
    C \\
    F
\end{pmatrix}
\]

\[
a = r_3s_3 \\
b = -r_4s_4 \\
c = -r_2\dot{s}_2 - r_2\dot{\theta}_2c_2 - r_3\dot{\theta}_3c_3 + r_5\dot{\theta}_5s_5 + r_5\dot{\theta}_5^2c_5 + r_4\dot{\theta}_4^2c_4 \\
d = -r_5c_5 \\
e = r_4c_4 \\
F = r_2\ddot{s}_2 - r_2\dot{\theta}_2^2s_2 - r_3\dot{\theta}_3^2s_3 - r_5\dot{\theta}_5c_5 + r_2\dot{\theta}_2^2s_2 + r_4\dot{\theta}_4^2c_4 + r_4\dot{\theta}_4^2s_4
\]

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

\[
\ddot{B} = \begin{pmatrix}
    \ddot{x} \\
    \ddot{y}
\end{pmatrix} = \begin{pmatrix}
    -r_2\dot{\theta}_2s_2 - r_2\dot{\theta}_2^2c_2 - r_3\dot{\theta}_3s_3 - r_5\dot{\theta}_5^2c_3 \\
    r_2\dot{\theta}_2c_2 - r_2\dot{\theta}_2^2s_2 + r_3\dot{\theta}_3c_3 - r_5\dot{\theta}_5^2s_3
\end{pmatrix}
\]

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).

4.2.3.2 Inverse Acceleration Kinematics

Given \( r_1, \theta_1, r_2, r_3, r_4, r_5; \) angles \( \theta_2, \theta_3, \theta_4, \) and \( \theta_5; \) angular rates \( \dot{\theta}_2, \dot{\theta}_3, \dot{\theta}_4, \dot{\theta}_5; \) plus desired end-effector translational acceleration \( \ddot{B} = \begin{pmatrix} \ddot{x} \\ \ddot{y} \end{pmatrix} \)

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 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 \( \ddot{B} = \begin{pmatrix} \ddot{x} \\ \ddot{y} \end{pmatrix} \) are repeated below, written in matrix-vector form:
\[
\begin{bmatrix}
-r_2 s_2 & -r_3 s_3 \\
r_2 c_2 & r_3 c_3
\end{bmatrix}
\begin{bmatrix}
\dot{\theta}_2 \\
\dot{\theta}_3
\end{bmatrix}
= \begin{bmatrix}
\dot{x} + r_2 \dot{\theta}_2^2 c_2 + r_3 \dot{\theta}_3^2 c_3 \\
\dot{y} + r_2 \dot{\theta}_2^2 s_2 + r_3 \dot{\theta}_3^2 s_3
\end{bmatrix}
\]
\[
\begin{bmatrix}
-r_2 s_4 & -r_3 s_5 \\
r_2 c_4 & r_3 c_5
\end{bmatrix}
\begin{bmatrix}
\dot{\theta}_4 \\
\dot{\theta}_5
\end{bmatrix}
= \begin{bmatrix}
\dot{x} + r_4 \dot{\theta}_4^2 c_4 + r_5 \dot{\theta}_5^2 c_4 \\
\dot{y} + r_4 \dot{\theta}_4^2 s_4 + r_5 \dot{\theta}_5^2 s_4
\end{bmatrix}
\]

And the inverse acceleration solutions are:

\[
\dot{\theta}_2 = \frac{\dot{x} c_2 + \dot{y} s_2 + r_2 \dot{\theta}_2^2 \cos(\theta_2 - \theta_3) + r_3 \dot{\theta}_3^2}{r_2 \sin(\theta_3 - \theta_2)}
\]

\[
\dot{\theta}_3 = \frac{-\dot{x} c_2 - \dot{y} s_2 - r_2 \dot{\theta}_2^2 \cos(\theta_3 - \theta_4) - r_3 \dot{\theta}_3^2}{r_3 \sin(\theta_3 - \theta_2)}
\]

\[
\dot{\theta}_4 = \frac{\dot{x} c_4 + \dot{y} s_4 + r_4 \dot{\theta}_4^2 \cos(\theta_2 - \theta_4) + r_5 \dot{\theta}_5^2}{r_4 \sin(\theta_3 - \theta_4)}
\]

\[
\dot{\theta}_5 = \frac{-\dot{x} c_4 - \dot{y} s_4 - r_4 \dot{\theta}_4^2 \cos(\theta_3 - \theta_4) - r_5 \dot{\theta}_5^2}{r_5 \sin(\theta_3 - \theta_4)}
\]

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 $\dot{\theta}_3$ and $\dot{\theta}_4$. This is similar to that subsection presented in the Velocity section and is left to the interested reader.
4.2.4 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)](image)

- $F_{ij}$ unknown vector internal joint force of link $i$ acting on link $j$.
- $\ell_{ij}$ 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 $r_1, \theta_1, r_2, \theta_2, r_3, \theta_3, r_4, \theta_4, r_5$, 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 $A_{GZ2}, A_{GZ3}, A_{GZ4}, A_{GZ5}$, and external end-effector force $F_E$

Find:

The driving actuator torques $\tau_2$ and $\tau_5$, plus internal joint forces $F_{21}, F_{32}, F_{43}, F_{54}, F_{15}$

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 $3 \times 4 = 12$ 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 $1+1+2 \times 5 = 12$ scalar unknowns.

Therefore, this problem can be solved.
Step 4. Derive the Newton-Euler Dynamics Equations

Newton's Second Law

**Link 2**
\[ \sum F_2 = F_{32} - F_{21} = m_2 a_{G2} \]

**Link 3**
\[ \sum F_3 = F_{43} - F_{32} + F_E = m_3 a_{G3} \]

**Link 4**
\[ \sum F_4 = F_{54} - F_{43} = m_4 a_{G4} \]

**Link 5**
\[ \sum F_5 = F_{15} - F_{54} = m_5 a_{G5} \]

Euler's Rotational Dynamics Equation

**Link 2**
\[ \sum M_{G2} = \tau_2 + r_{32} \times F_{32} - r_{12} \times F_{21} = I_{G2Z} \alpha_2 \]

**Link 3**
\[ \sum M_{G3} = r_{43} \times F_{43} - r_{23} \times F_{32} + r_{43} \times F_E = I_{G3Z} \alpha_3 \]

**Link 4**
\[ \sum M_{G4} = r_{54} \times F_{54} - r_{34} \times F_{43} = I_{G4Z} \alpha_4 \]

**Link 5**
\[ \sum M_{G5} = \tau_5 + r_{15} \times F_{15} - r_{45} \times F_{54} = I_{G5Z} \alpha_5 \]
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**

\[ F_{32X} - F_{21X} = m_2 A_{G2X} \]

\[ F_{32Y} - F_{21Y} = m_2 A_{G2Y} \]

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

**Link 3**

\[ F_{43X} - F_{32X} = m_3 A_{G3X} - F_{EX} \]

\[ F_{43Y} - F_{32Y} = m_3 A_{G3Y} - F_{EY} \]

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

**Link 4**

\[ F_{14X} - F_{43X} = m_4 A_{G4X} \]

\[ F_{14Y} - F_{43Y} = m_4 A_{G4Y} \]

\[ (r_{14X} F_{14Y} - r_{14Y} F_{14X}) - (r_{43X} F_{43Y} - r_{43Y} F_{43X}) = I_{G4Z} \alpha_4 \]

**Link 5**

\[ F_{15X} - F_{54X} = m_5 A_{G5X} \]

\[ F_{15Y} - F_{54Y} = m_5 A_{G5Y} \]

\[ \tau_5 + (r_{15X} F_{15Y} - r_{15Y} F_{15X}) - (r_{54X} F_{54Y} - r_{54Y} F_{54X}) = I_{G5Z} \alpha_5 \]
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**

\[
\begin{bmatrix}
-1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & r_{23Y} & -r_{23X} & -r_{34Y} & r_{34X} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & r_{34Y} & -r_{34X} & -r_{54Y} & r_{54X} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & r_{45Y} & -r_{45X} & -r_{15Y} & r_{15X} & 0 & 1 & 0 \\
\end{bmatrix}
\begin{bmatrix}
F_{21X} \\
F_{21Y} \\
F_{32X} \\
F_{32Y} \\
F_{43X} \\
F_{43Y} \\
F_{44X} \\
F_{44Y} \\
F_{45X} \\
F_{45Y} \\
r_2 \\
r_5 \\
\end{bmatrix}
= 
\begin{bmatrix}
m_2 A_{G2X} \\
m_2 A_{G2Y} \\
I_{G2Z} \alpha_2 \\
m_3 A_{G3X} - F_{EX} \\
m_3 A_{G3Y} - F_{EY} \\
I_{G3Z} \alpha_3 - r_{43X} F_{EY} + r_{43Y} F_{EX} \\
m_4 A_{G4X} \\
m_4 A_{G4Y} \\
I_{G4Z} \alpha_4 \\
m_5 A_{G5X} \\
m_5 A_{G5Y} \\
I_{G5Z} \alpha_5 \\
\end{bmatrix}
\]

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

\[
v = \text{inv}(A) * b; \quad \% \text{Solution via matrix inverse}
\]

Using Gaussian elimination is more efficient and robust to solve for \(v\).

MATLAB

\[
v = A \backslash b; \quad \% \text{Solution via Gaussian elimination}
\]

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 \texttt{for i} loop.

\[
\begin{align*}
F21x(i) &= v(1); \\
F21y(i) &= v(2); \\
&\vdots \\
F15y(i) &= v(10); \\
\tau2(i) &= v(11); \\
\tau5(i) &= v(12);
\end{align*}
\]

Terms for the matrix-vector equation

Absolute translational center-of-gravity accelerations

\[
\mathbf{A}_{G2} = \begin{bmatrix}
A_{G2x} \\
A_{G2y}
\end{bmatrix} = \begin{bmatrix}
-R_{G2}\alpha_2 \sin \theta_2 - R_{G2}\omega_2^2 \cos \theta_2 \\
R_{G2}\alpha_2 \cos \theta_2 - R_{G2}\omega_2^2 \sin \theta_2
\end{bmatrix}
\]

\[
\mathbf{A}_{G3} = \begin{bmatrix}
A_{G3x} \\
A_{G3y}
\end{bmatrix} = \begin{bmatrix}
-r_2 \alpha_2 \sin \theta_2 - r_2 \omega_2^2 \cos \theta_2 - R_{G2}\alpha_3 \sin \theta_2 - R_{G2}\omega_3^2 \cos \theta_3 \\
r_2 \alpha_2 \cos \theta_2 - r_2 \omega_2^2 \sin \theta_2 + R_{G3}\alpha_3 \cos \theta_2 - R_{G3}\omega_3^2 \sin \theta_3
\end{bmatrix}
\]

\[
\mathbf{A}_{G4} = \begin{bmatrix}
A_{G4x} \\
A_{G4y}
\end{bmatrix} = \begin{bmatrix}
-r_3 \alpha_3 \sin \theta_3 - r_3 \omega_3^2 \cos \theta_3 - R_{G3}\alpha_4 \sin \theta_3 - R_{G3}\omega_4^2 \cos \theta_4 \\
r_3 \alpha_3 \cos \theta_3 - r_3 \omega_3^2 \sin \theta_3 + R_{G4}\alpha_4 \cos \theta_3 - R_{G4}\omega_4^2 \sin \theta_4
\end{bmatrix}
\]

\[
\mathbf{A}_{G5} = \begin{bmatrix}
A_{G5x} \\
A_{G5y}
\end{bmatrix} = \begin{bmatrix}
-R_{G5}\alpha_5 \sin \theta_5 - R_{G5}\omega_5^2 \cos \theta_5 \\
R_{G5}\alpha_5 \cos \theta_5 - R_{G5}\omega_5^2 \sin \theta_5
\end{bmatrix}
\]

where \(R_{Gi} = r_i/2\) for uniformly-distributed homogeneous material with regular geometry.
Moment arm position vectors

\[ \mathbf{r}_{12} = \begin{pmatrix} r_{12x} \\ r_{12y} \end{pmatrix} = \begin{pmatrix} -\frac{r_2}{2} \cos \theta_2 \\ -\frac{r_2}{2} \sin \theta_2 \end{pmatrix} \]

\[ \mathbf{r}_{34} = \begin{pmatrix} r_{34x} \\ r_{34y} \end{pmatrix} = \begin{pmatrix} \frac{r_4}{2} \cos \theta_4 \\ \frac{r_4}{2} \sin \theta_4 \end{pmatrix} \]

\[ \mathbf{r}_{32} = \begin{pmatrix} r_{32x} \\ r_{32y} \end{pmatrix} = \begin{pmatrix} \frac{r_2}{2} \cos \theta_2 \\ \frac{r_2}{2} \sin \theta_2 \end{pmatrix} \]

\[ \mathbf{r}_{54} = \begin{pmatrix} r_{54x} \\ r_{54y} \end{pmatrix} = \begin{pmatrix} -\frac{r_4}{2} \cos \theta_4 \\ -\frac{r_4}{2} \sin \theta_4 \end{pmatrix} \]

\[ \mathbf{r}_{23} = \begin{pmatrix} r_{23x} \\ r_{23y} \end{pmatrix} = \begin{pmatrix} \frac{r_3}{2} \cos \theta_3 \\ \frac{r_3}{2} \sin \theta_3 \end{pmatrix} \]

\[ \mathbf{r}_{15} = \begin{pmatrix} r_{15x} \\ r_{15y} \end{pmatrix} = \begin{pmatrix} \frac{r_5}{2} \cos \theta_5 \\ -\frac{r_5}{2} \sin \theta_5 \end{pmatrix} \]

\[ \mathbf{r}_{43} = \begin{pmatrix} r_{43x} \\ r_{43y} \end{pmatrix} = \begin{pmatrix} \frac{r_3}{2} \cos \theta_3 \\ \frac{r_3}{2} \sin \theta_3 \end{pmatrix} \]

\[ \mathbf{r}_{45} = \begin{pmatrix} r_{45x} \\ r_{45y} \end{pmatrix} = \begin{pmatrix} \frac{r_5}{2} \cos \theta_5 \\ \frac{r_5}{2} \sin \theta_5 \end{pmatrix} \]
4.3 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 4290/5290 NotesBook. Here we solve the above equations directly, without utilizing the 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:

\[
d = -\frac{(b_x - a_x)}{(b_y - a_y)}
\]

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

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 \(\{0\}\).
4.4 Planar 3-RPR Manipulator

4.4.1 Planar 3-RPR Manipulator Inverse Pose Kinematics

Inverse Pose Kinematics (for pose control)

**Given:**
\( x, y, \phi \)

**Find:**
\( L_1, L_2, L_3 \)

Vector loop-closure equations:

\[
\left\{ \begin{array}{c}
0 \\
0 \\
0
\end{array} \right\}_H^i = \left\{ \begin{array}{c}
0 \\
0 \\
0
\end{array} \right\}_{A_i} + \left\{ \begin{array}{c}
L_i
\end{array} \right\}
\]

The vector loop-closure equations are rewritten below:

\[
\left\{ \begin{array}{c}
L_i
\end{array} \right\} = \left\{ \begin{array}{c}
P_i
\end{array} \right\}_H + \left[ \begin{array}{c}
R
\end{array} \right]_{C_i} \left\{ \begin{array}{c}
P_i
\end{array} \right\}_C - \left\{ \begin{array}{c}
A_i
\end{array} \right\}
\]

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

\[
L_i = \left\| \left\{ \begin{array}{c}
L_i
\end{array} \right\} \right\| = \left\| \left\{ \begin{array}{c}
P_i
\end{array} \right\}_H + \left[ \begin{array}{c}
R
\end{array} \right]_{C_i} \left\{ \begin{array}{c}
P_i
\end{array} \right\}_C - \left\{ \begin{array}{c}
A_i
\end{array} \right\} \right\|
\]

\( i = 1, 2, 3 \)
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.

$$\theta_i = \tan^{-1} \left[ \frac{C_{ix} - A_{iy}}{C_{iy} - A_{ix}} \right] \quad i = 1, 2, 3$$
4.4.2 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: \( x, y, \phi \)

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:

\[
\left\{ ^0 \mathbf{L}_i \right\} = \left\{ ^0 \mathbf{P}_R \right\} + \left[ ^0 \mathbf{R} \right] \left\{ ^H \mathbf{P}_C \right\} - \left\{ ^0 \mathbf{A}_i \right\} \quad i = 1, 2, 3
\]

\[
\begin{align*}
L_c \phi_i &= \left[ \begin{array}{c} x \\ y \\ \phi \end{array} \right] + \left[ \begin{array}{ccc} c\phi & -s\phi & -s\phi \\ s\phi & c\phi & -c\phi \\ 0 & 0 & 1 \end{array} \right] \left[ ^H \mathbf{C}_{ix} \right] - \left\{ ^0 \mathbf{A}_{ix} \right\} \\
L_s \phi_i &= \left[ \begin{array}{c} x \\ y \\ \phi \end{array} \right] + \left[ \begin{array}{ccc} c\phi & -s\phi & -s\phi \\ s\phi & c\phi & -c\phi \\ 0 & 0 & 1 \end{array} \right] \left[ ^H \mathbf{C}_{iy} \right] - \left\{ ^0 \mathbf{A}_{iy} \right\}
\end{align*}
\]

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.

\[
\begin{align*}
L_c \phi_i &= x + c\phi^H \mathbf{C}_{ix} - s\phi^H \mathbf{C}_{iy} - ^0 \mathbf{A}_{ix} \\
L_s \phi_i &= y + s\phi^H \mathbf{C}_{ix} + c\phi^H \mathbf{C}_{iy} - ^0 \mathbf{A}_{iy} \\
&\quad i = 1, 2, 3
\end{align*}
\]

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 \( x, y, \phi \). This problem is solved via the Newton-Raphson iterative numerical method in Section 4.4.

\[
x^2 + y^2 + A_{ix}^2 + A_{iy}^2 + C_{ix}^2 + C_{iy}^2 - L_i^2 - 2(xA_{ix} + yA_{iy}) + 2c\phi(xC_{ix} + yC_{iy} - A_{ix}C_{ix} - A_{iy}C_{iy}) \\
+ 2s\phi(-xC_{ix} + yC_{iy} + A_{ix}C_{ix} - A_{iy}C_{ix}) = 0 \\
\quad i = 1, 2, 3
\]

where known constants \( ^0 \mathbf{A}_{ix}, ^0 \mathbf{A}_{iy}, ^H \mathbf{C}_{ix}, ^H \mathbf{C}_{iy} \) 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 $A_3C_3$ constrains the mechanism to a statically-determinate structure of 0-dof. Point $C_3$ 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 $A_3$ (Hunt, 1990).

Figure:

Branch $A_1C_1C_2A_2$ is a 4-bar mechanism with input angle $\theta_1$ and output angle $\theta_2$ (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.
4.4.3 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](image)

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:

\[
\{^0P_H\} = \{^0A_i\} + \{^0L_i\} + \{^0L_{Hi}\} \quad i = 1, 2, 3
\]
The $XY$ component equations are:

\[
\begin{align*}
x &= x_i + L_c \theta_i + L_H i \cos(\theta_i + \beta_i) \\
y &= y_i + L_s \theta_i + L_H i \sin(\theta_i + \beta_i)
\end{align*}
\]

and the angle equation is:

\[
\phi + \gamma_i = \theta_i + \beta_i 
\]

$\beta_i$ for use in velocity equations:

\[
\begin{align*}
\theta_1 + \beta_1 &= \phi + \gamma_1 \\
\theta_2 + \beta_2 &= \phi + \gamma_2 + 120^\circ \\
\theta_3 + \beta_3 &= \phi + 270^\circ
\end{align*}
\]

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

The velocity equations for one $\text{RPR}$ leg are obtained from the first time derivatives of the $XY$ and angle equations.

\[
\begin{align*}
\dot{x} &= -L_s \theta_i \dot{\theta}_i + \dot{L}_c \theta_i - L_H i \sin(\theta_i + \beta_i)(\dot{\theta}_i + \dot{\beta}_i) \\
\dot{y} &= L_c \theta_i \dot{\theta}_i + \dot{L}_s \theta_i + L_H i \cos(\theta_i + \beta_i)(\dot{\theta}_i + \dot{\beta}_i) \\
\dot{\phi} &= \dot{\theta}_i + \dot{\beta}_i
\end{align*}
\]

These equations are written in matrix-vector form to yield the $\text{RPR}$ leg Jacobian matrix.

\[
\begin{align*}
\begin{bmatrix}
\dot{x} \\
\dot{y} \\
\omega_z
\end{bmatrix}
&= 
\begin{bmatrix}
-L_s \theta_i - L_H i \sin(\theta_i + \beta_i) & c\theta_i & -L_H i \sin(\theta_i + \beta_i) \\
L_c \theta_i + L_H i \cos(\theta_i + \beta_i) & s\theta_i & L_H i \cos(\theta_i + \beta_i) \\
1 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
\dot{\theta}_i \\
\dot{\beta}_i \\
\dot{L}_\phi
\end{bmatrix}
\end{align*}
\]

Written in compact notation:

\[
\begin{align*}
\{\dot{X}\} &= [J_i]\{\dot{\beta}_i\} \\
n & = 1, 2, 3
\end{align*}
\]
\[ \{\dot{X}\} = \{x\ y\ \omega_z\}^T \] is the same for all three legs (the Cartesian end-effector rates). \( \{\dot{\rho}_i\} \) 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:

\[
\begin{bmatrix}
-\sin \theta_i \\
\cos \theta_i \\
\sin \theta_i \\
\frac{L_i}{L_i} \\
\frac{L_i}{L_i} \\
\frac{L_i}{L_i}
\end{bmatrix}
\begin{bmatrix}
\dot{c}\theta_i \\
\dot{c}\theta_i \\
\dot{s}\theta_i \\
\frac{L_{ii}}{L_i} \\
\frac{L_{ii}}{L_i} \\
\frac{L_{ii}}{L_i}
\end{bmatrix}
\begin{bmatrix}
\dot{x} \\
\dot{y} \\
\omega_z
\end{bmatrix}
\]

\( i = 1,2,3 \)

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:

\[
\begin{bmatrix}
\dot{L}_1 \\
\dot{L}_2 \\
\dot{L}_3
\end{bmatrix}
= \begin{bmatrix}
\frac{L_{ii}}{L_i} & \frac{L_{ii}}{L_i} & \frac{L_{ii}}{L_i}
\end{bmatrix}
\begin{bmatrix}
\dot{x} \\
\dot{y} \\
\omega_z
\end{bmatrix}
\]

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

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 \([J_{NR}]\) from the numerical FPK solution.

**Forward Velocity Problem**

This problem is obtained by inverting the Inverse Velocity Solution:

\[ \{\dot{X}\} = [M^{-1}]\{L\} \]

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.

```matlab
% 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:

```matlab
syms L LH t1 b1;
```
4.5 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.

\[
\{ F(\mathbf{X}) \} = \{ 0 \}
\]

where the \( n \) functions are

\[
\{ F(\mathbf{X}) \} = \{ F_1(\mathbf{X}) \ F_2(\mathbf{X}) \ \cdots \ F_n(\mathbf{X}) \}^T
\]

and the \( n \) variables are

\[
\{ \mathbf{X} \} = \{ x_1 \ x_2 \ \cdots \ x_n \}^T
\]

Perform a Taylor Series Expansion of \( \{ F \} \) about \( \{ \mathbf{X} \} \):

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

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 \( \{ \mathbf{X} \} \). If \( \{ \delta \mathbf{X} \} \) is small, the higher-order terms \( O(\{ \delta \mathbf{X}^2 \}) \) from the expansion are negligible. For solution, we require:

\[
F_i(\{ \mathbf{X} \} + \{ \delta \mathbf{X} \}) = 0 \quad i = 1, 2, \cdots , n
\]

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

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

So to calculate the required correction factor \( \delta \mathbf{X} \) at each solution iteration step, we must solve

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

Solution via Gaussian elimination on \( [J_{NR}] \{ \delta \mathbf{X} \} = -\{ F(\{ \mathbf{X} \}) \} \) 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: \( \{ F(\{ X \}) \} = \{ 0 \} \)

1) Make an initial guess to the solution: \( \{ X_0 \} \)

2) Solve \( [J_{NR}]\{ \{ X_k \} \}\{ \delta X_k \} = -\{ F(\{ X_k \}) \} \) for \( \{ \delta X_k \} \), where \( k \) is the iteration counter.

3) Update the current best guess for the solution: \( \{ X_{k+1} \} = \{ X_k \} + \{ \delta X_k \} \)

4) Iterate until \( \| \{ \delta X_k \} \| < \epsilon \), where we use the Euclidean norm and \( \epsilon \) 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 \( \{ X_0 \} \) 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:

\[ \{ X \} = \{ x \ y \ \phi \}^T \]

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.

\[ J_{NR}(\{X\}) = \begin{bmatrix} \frac{\partial F_i}{\partial x_j} \end{bmatrix} \]

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

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

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

where \( \{^0 A_i\} = \begin{bmatrix} A_{ix} \\ A_{iy} \end{bmatrix} \) and \( \{^i C_i\} = \begin{bmatrix} C_{ix} \\ C_{iy} \end{bmatrix} \)

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 $i = 1, 2, \ldots, 6$. 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 $\{X\} = \{x, y, \phi, \theta_1, \theta_2, \theta_3\}^T$.

$$
\begin{bmatrix}
J_{NR}(\{X\})
\end{bmatrix} = 
\begin{bmatrix}
1 & 0 & -P_1s\phi - P_1c\phi & L_1s\theta_1 & 0 & 0 \\
0 & 1 & P_1c\phi - P_1s\phi & -L_1c\theta_1 & 0 & 0 \\
1 & 0 & -P_2s\phi - P_2c\phi & 0 & L_2s\theta_2 & 0 \\
0 & 1 & P_2c\phi - P_2s\phi & 0 & -L_2c\theta_2 & 0 \\
1 & 0 & -P_3s\phi - P_3c\phi & 0 & 0 & L_3s\theta_3 \\
0 & 1 & P_3c\phi - P_3s\phi & 0 & 0 & -L_3c\theta_3 \\
\end{bmatrix}
$$
4.6 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 2.6 of the 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](image)

For determination of the **dexterous workspace**, it is most convenient to numerically or geometrically generate in MATLAB the reachable workspace for different $\phi$ 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 $\pm \phi_{LIMIT}$. 
4.7 NIST RoboCrane Cable-Suspended Parallel Robot

The NIST RoboCrane\textsuperscript{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 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 $\{P\}$:

$$
{^0B_i} = \begin{bmatrix}
\frac{s_B}{2} \\
-h_B \\
H
\end{bmatrix} \quad
{^0B_2} = \begin{bmatrix}
0 \\
l_B \\
H
\end{bmatrix} \quad
{^0B_3} = \begin{bmatrix}
\frac{-s_B}{2} \\
-h_B \\
H
\end{bmatrix}
$$

$$
{^P\mathbf{p}_1} = \begin{bmatrix}
0 \\
-l_p \\
0
\end{bmatrix} \quad
{^P\mathbf{p}_2} = \begin{bmatrix}
\frac{s_p}{2} \\
h_p \\
0
\end{bmatrix} \quad
{^P\mathbf{p}_3} = \begin{bmatrix}
\frac{-s_p}{2} \\
h_p \\
0
\end{bmatrix}
$$

where:

$$
h_B = \frac{\sqrt{3}}{6} s_B \quad l_B = \frac{\sqrt{3}}{3} s_B \quad h_p = \frac{\sqrt{3}}{6} s_p \quad l_p = \frac{\sqrt{3}}{3} s_p
$$
RoboCrane Kinematics Diagrams (Top and Front Views)
<table>
<thead>
<tr>
<th>name</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_B$</td>
<td>base equilateral triangle side</td>
</tr>
<tr>
<td>$s_P$</td>
<td>platform equilateral triangle side</td>
</tr>
<tr>
<td>$H$</td>
<td>height from ground to base triangle</td>
</tr>
<tr>
<td>$h_B$</td>
<td>planar distance from ${0}$ to near base side</td>
</tr>
<tr>
<td>$l_B$</td>
<td>planar distance from ${0}$ to a base vertex</td>
</tr>
<tr>
<td>$h_P$</td>
<td>planar distance from ${P}$ to near platform side</td>
</tr>
<tr>
<td>$l_P$</td>
<td>planar distance from ${P}$ to a platform vertex</td>
</tr>
<tr>
<td>$m$</td>
<td>platform mass</td>
</tr>
</tbody>
</table>

**Inverse Pose Kinematics (IPK) Solution**

The 6-cable RoboCrane robot inverse pose kinematics (IPK) problem is stated: Given the desired moving platform pose $[^0 PT]$, 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 $[^0 PT]$ may be specified in terms of the desired vector location $[^0 P]$ 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 $\alpha$-$\beta$-$\gamma$, Z-Y-X Euler Angles (Craig, 2005), the associated orthonormal rotation matrix is:

$$[^0 P R] = \begin{bmatrix} cac\beta & -sac\gamma + cas\beta s\gamma & sas\gamma + cas\beta c\gamma \\ sac\beta & cac\gamma + sas\beta s\gamma & -cas\gamma + sas\beta c\gamma \\ -s\beta & c\beta s\gamma & c\beta c\gamma \end{bmatrix}$$

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

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

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 $[^0 PT]$, 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:
\[ \{^0P_j\} = \begin{bmatrix} ^0T \end{bmatrix} \{^P_j\} \quad j = 1, 2, 3 \]

where the \( \{^P_j\} \) 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:

\[
\begin{align*}
L_1 &= \left\| ^0B_1 - ^0P_1 \right\| \\
L_2 &= \left\| ^0B_1 - ^0P_2 \right\| \\
L_3 &= \left\| ^0B_2 - ^0P_2 \right\| \\
L_4 &= \left\| ^0B_2 - ^0P_3 \right\| \\
L_5 &= \left\| ^0B_3 - ^0P_3 \right\| \\
L_6 &= \left\| ^0B_3 - ^0P_1 \right\| 
\end{align*}
\]

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 \( L_i \), \( i = 1, 2, \ldots, 6 \), calculate the resulting moving platform pose \[ ^0T \]. 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 \( B_3P_1B_4 \). 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 \( s_j \) simultaneously. This solution was presented by Dr. Bob\(^3\). The solution boils down to an 8\(^{th}\)-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.
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 $t_i$, $i = 1, 2, \cdots, 6$. 
The vector force and moment equations of static equilibrium are:

\[
\sum_{i=1}^{6} t_i + mg + F_{\text{EXT}} = 0 \\
\sum_{i=1}^{6} m_i + \begin{bmatrix} 0 & R \end{bmatrix} \dot{p} \begin{pmatrix} p_{CG} \end{pmatrix} \times mg + M_{\text{EXT}} = 0
\]

where \( t_i = \dot{t}_i \dot{L}_i \) is the vector cable tension applied to the moving platform by the \( i^{th} \) active cable (in the positive cable length direction \( \dot{L}_i \) as established in IPK); \( m \) is the moving platform mass; \( g = \begin{pmatrix} 0 & 0 & -g \end{pmatrix}^T \) is the gravity vector; \( F_{\text{EXT}} \) is the external vector force exerted on the moving platform by the environment; \( m_i = \begin{bmatrix} 0 & R \end{bmatrix} \begin{pmatrix} p_{CG} \end{pmatrix} \times \begin{pmatrix} t_i \end{pmatrix} \) is the moment due to the \( i^{th} \) active cable tension (\( \dot{p}_i \) is the moment arm from the moving platform control point \( P \) to the \( i^{th} \) active cable connection point, expressed in \( \{P\} \) coordinates); \( \begin{pmatrix} p_{CG} \end{pmatrix} \) is the position vector to the moving platform \( CG \) from the moving platform control point \( P \) (the origin of \( \{P\} \)); and \( M_{\text{EXT}} \) 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:

\[
[S] \begin{pmatrix} t \end{pmatrix} = - \begin{pmatrix} W_{\text{EXT}} + G \end{pmatrix}
\]
where $\{t\} = \{t_1, t_2, \ldots, t_6\}^T$ is the vector of active cable tensions, $\{G\} = \{mg \ 0^\mathbf{r} \mathbf{R}_{CG} \times mg\}^T$ is the gravity wrench vector, $\{W_{\text{EXT}}\} = \{F_{\text{EXT}} \ M_{\text{EXT}}\}^T$ is the external wrench vector, and the statics Jacobian matrix $[S]$ is:

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

where $\mathbf{P}_j = \{0 \mathbf{P}_j\} = \{0^\mathbf{r} \mathbf{R}_{\mathbf{P}_j}\} \{\mathbf{P}_j\}, \ j = 1, 2, 3$.

### RoboCrane Inverse Pseudostatics Solution

The statics equations can be used in two ways. Given the active cable tensions $\{t\}$ and the six cable unit vectors $\hat{L}_i$ 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 $\{t\}$ given the focus cabin mass and pose, plus all $\hat{L}_i$. It is solved by inverting the static equilibrium equations:

$$\{t\} = -[S]^{-1} \{W_{\text{EXT}} + G\}$$

The statics Jacobian Matrix $[S]$ is a square 6x6 matrix and hence the standard matrix inverse applies in (11). A unique $\{t\}$ 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 $\tau_i$ 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 $^p\mathbf{P}_{CG} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}^T$. Also there is zero external wrench $\{\mathbf{W}_{EXT}\}$.

**Snapshot Example**

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

$$
^0\mathbf{L} = \begin{bmatrix} 7.080 \\ 8.313 \\ 6.203 \\ 5.777 \\ 8.494 \\ 8.711 \end{bmatrix}^T \text{ m}
$$

$$
^0\mathbf{t} = \begin{bmatrix} 325.1 \\ 125.2 \\ 318.6 \\ 352.4 \\ 76.1 \\ 123.8 \end{bmatrix}^T \text{ N}
$$
**Trajectory Example**

The moving platform control point \( \{P\} \) traces a circle of center \( \{0 \ 0 \ 4\}^T \) and radius 2 m, with zero orientations \( \{\alpha \ \beta \ \gamma\} = \{0 \ 0 \ 0\} \). 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](image)

**Cable Lengths**

![Cable Lengths](image)

**Cable Tensions**

![Cable Tensions](image)
4.8 Delta Parallel Robot Inverse and Forward Kinematics

Clavel’s 3-RUU Delta Robot⁴ is arguably the most successful commercial parallel robot to date. The left image below shows the original design from Clavel’s U.S. patent⁵, and the right photograph below shows one commercial instantiation of the Delta Robot.

![Delta Robot Design](Image)

![ABB FlexPicker Delta Robot](Image)

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:


---

4.9 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.

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

www.ohio.edu/people/williard/html/PDF/StewartPlatform.pdf
5. Manipulator 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 \( N \times 1 \) 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 \( N \times 1 \) 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 \( \{ a_i \}, \{ \omega_i \}, \{ \alpha_i \} \) 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
5.1 Inertia Tensor (Mass Distribution)

The inertia tensor is a spatial generalization of the planar scalar moment of inertia. Its units are mass times distance$^2$ ($kg\cdot m^2$). 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

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

Mass products of inertia

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

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). $P_C = \{x_C \ y_C \ z_C\}^T$ 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.

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

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

$$[^A I] = [^C I] + m \left[ \{P_C\}^T \{P_C\} [I_3] - \{P_C\} \{P_C\}^T \right]$$
5.2 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 \( i \)**

\[
\{ f_i \} \quad \text{internal force exerted on link} \; i \; \text{by link} \; i-1.
\]

\[
\{ n_i \} \quad \text{internal moment exerted on link} \; i \; \text{by link} \; i-1.
\]

**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 method 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)

\[
\begin{align*}
\{\omega_{i+1}\} &= \{\omega_i\} + \{\ddot{Z}_{i+1}\} \dot{\theta}_{i+1} \\
\{\alpha_{i+1}\} &= \{\alpha_i\} + \{\omega_i\} \times \{\ddot{Z}_{i+1}\} \dot{\theta}_{i+1} + \{\dddot{Z}_{i+1}\} \ddot{\theta}_{i+1} \\
\{a_{i+1}\} &= \{a_i\} + \{\alpha_i\} \times \{i P_{i+1}\} + \{\omega_i\} \times \{\omega_i\} \times \{i P_{i+1}\} \\
\{a_{Ci+1}\} &= \{a_{i+1}\} + \{\alpha_{i+1}\} \times \{i P_{Ci+1}\} + \{\omega_{i+1}\} \times \{\omega_{i+1}\} \times \{i P_{Ci+1}\}
\end{align*}
\]

Inertial loading (kinetics)

\[
\begin{align*}
\{F_{i+1}\} &= m_{i+1} \{a_{Ci+1}\} \\
\{N_{i+1}\} &= \begin{bmatrix} cI \end{bmatrix} \{\alpha_{i+1}\} + \{\omega_{i+1}\} \times \begin{bmatrix} cI \end{bmatrix} \{\omega_{i+1}\}
\end{align*}
\]

**Inward iteration for kinetics**  \( i : N \to 1 \)
(without regard for frames of expression, for clarity)

Internal forces and moments

\[
\begin{align*}
\{f_i\} &= \{f_{i+1}\} + \{F_{i}\} \\
\{n_i\} &= \{n_{i+1}\} + \{i P_{Ci}\} \times \{F_{i}\} + \{i P_{i+1}\} \times \{f_{i+1}\} + \{N_{i}\}
\end{align*}
\]

Externally applied joint torques

\[
\tau_i = \{n_i\} \cdot \{Z_i\}
\]

**Inclusion of gravity forces**

\[
\begin{bmatrix} 0 \omega_0 \end{bmatrix} = \{g\}
\]

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).
5.3 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.

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

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

$$ u_i = u_{REF} - m_i \left\{ 0 \right\} \cdot \left\{ 0 P_{Ci} \right\} $$

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

$$ u(\Theta) = \sum_{i=1}^{N} u_i $$

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

$$ \frac{d}{dt} \left( \frac{\partial k}{\partial \dot{\theta}_i} \right) - \frac{\partial k}{\partial \theta_i} + \frac{\partial u}{\partial \theta_i} = \tau_i $$
5.4 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.)

\[ \sum F_x = m a_{cx} \]

a) \[ ^0 F_x = m \left( -\frac{L}{2} \alpha s \theta - \frac{L}{2} \omega^2 c \theta \right) \]
\[ ^0 F_x = -\frac{mL}{2} (\ddot{s} \theta + \dot{c} \dot{s} \theta) \]

\[ \sum F_y = m a_{cy} \]

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

\[ \sum M_{z_0} = ^0 I \ddot{\theta} = \tau - \frac{mL}{2} g \dot{c} \theta \]
\[ \tau = ^0 I \ddot{\theta} + \frac{mL}{2} g \dot{c} \theta \]

where \[ ^0 I = c I + m d^2 = c \frac{I_z}{z} + \frac{mL^2}{4} \]
2) Newton-Euler recursion

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

$$\begin{align*}
\{^0P_i\} &= \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \\
\{^1P_{ci}\} &= \begin{bmatrix} L/2 \\ 0 \\ 0 \end{bmatrix}, \\
\{^0R\} &= \begin{bmatrix} c\theta & s\theta & 0 \\ -s\theta & c\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}, \\
\{^cI_1\} &= \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}
\end{align*}$$

$$\begin{align*}
\{^0\omega_b\} &= \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \\
\{^0\alpha_0\} &= \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \\
\{^0a_0\} &= \begin{bmatrix} 0 \\ 0 \end{bmatrix}
\end{align*}$$

to account for gravity

$$\begin{align*}
\{^1a_i\} &= \begin{bmatrix} c\theta & s\theta & 0 \\ -s\theta & c\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} g\theta \\ g\theta \\ 0 \end{bmatrix}
\end{align*}$$

$$\begin{align*}
\{^1a_{ci}\} &= \{^1a_i\} \times \{^1P_{ci}\} + \{^1\omega_i\} \times \{^1P_{ci}\} + \{^1\alpha_i\} = \begin{bmatrix} g\theta \frac{L}{2} \dot{\theta}^2 \\ g\theta \dot{\theta} \\ 0 \end{bmatrix}
\end{align*}$$

$$\begin{align*}
\{^1F_i\} &= m \begin{bmatrix} g\theta \frac{L}{2} \dot{\theta}^2 \\ g\theta \dot{\theta} \\ 0 \end{bmatrix}, \\
\{^1N_i\} &= [^cI_1] \{^1\alpha_i\} + [^cI_1] \{^1\omega_i\} = \begin{bmatrix} 0 \\ 0 \\ I_{zz} \end{bmatrix}
\end{align*}$$
Inward iteration \( i = 1 \) (the only iteration)

\[
\begin{align*}
\{^2f_2\} &= \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad \{^2n_2\} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \\
\{^1f_1\} &= \begin{bmatrix} 1R \end{bmatrix}\{^2f_2\} + \{^1F_1\} = m \begin{bmatrix} gs\theta - \frac{L}{2}\dot{\theta}^2 \\ gc\theta + \frac{L}{2}\ddot{\theta} \\ 0 \end{bmatrix}, \\
\{^1n_1\} &= \begin{bmatrix} 1R \end{bmatrix}\{^2n_2\} + \{^1P_1\} \times \{^1F_1\} + \{^1P_2\} \times \begin{bmatrix} 1R \end{bmatrix}\{^2f_2\} + \{^1N_1\} \\
\{^1n_1\} &= \begin{bmatrix} 0 \\ 0 \\ mL \frac{gc\theta + \frac{L}{2}\ddot{\theta}}{2} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ I \ddot{\theta} \end{bmatrix} \\
\tau &= \{^1n_1\} \times \{^1f_1\} \\
\tau &= \left( I + \frac{mL^2}{4} \right) \ddot{\theta} + \frac{mL}{2} gc\theta
\end{align*}
\]

Check with the sophomore dynamics method above.

\[
\begin{align*}
\{^0f_1\} &= \begin{bmatrix} c\theta & -s\theta & 0 \\ s\theta & c\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} gc\theta + \frac{L}{2}\ddot{\theta} \end{bmatrix} = m \begin{bmatrix} -\frac{L}{2} \ddot{\theta}^2 c\theta - \frac{L}{2} \ddot{\theta} s\theta \\ -\frac{L}{2} \ddot{\theta}^2 s\theta + \frac{L}{2} \ddot{\theta} c\theta + g \\ 0 \end{bmatrix}
\end{align*}
\]
3) Lagrange-Euler formulation

\[ \{1, \omega_1\} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad [c_i I] = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \]

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

\[ \{1, v_c\} = \begin{bmatrix} \frac{L}{2} \dot{\theta} \\ 0 \\ 0 \end{bmatrix}, \quad \{1, \omega_1\} [c_i I] \{1, \omega_1\} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} \frac{L}{2} \dot{\theta} \\ 0 \\ I_{zz} \dot{\theta} \end{bmatrix} \]

\[ k = \frac{1}{2} \frac{mL^2}{4} \dot{\theta}^2 + \frac{1}{2} I_{zz} \dot{\theta}^2 \]

\[ u = 0 - m \begin{bmatrix} 0 \\ -g \end{bmatrix} \begin{bmatrix} \frac{L}{2} \dot{\theta} \\ 0 \\ 0 \end{bmatrix} = mg \frac{L}{2} s \dot{\theta} \]

\[ L = k - u = \frac{1}{2} \frac{mL^2}{4} \dot{\theta}^2 - mg \frac{L}{2} s \dot{\theta} \]

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

\[ \frac{\partial L}{\partial \dot{\theta}} = \dot{\theta} \left( I_{zz} + \frac{mL^2}{4} \right) \]

\[ \frac{d}{dt} \left( \frac{\partial L}{\partial \dot{\theta}} \right) = \ddot{\theta} \left( I_{zz} + \frac{mL^2}{4} \right) \]

\[ \frac{\partial L}{\partial \theta} = -mg \frac{L}{2} c \dot{\theta} \]

\[ \tau = \left( I_{zz} + \frac{mL^2}{4} \right) \ddot{\theta} + \frac{mL}{2} g c \dot{\theta} \]

This result agrees with the sophomore and Newton-Euler recursion dynamics methods above to solve the same problem.
5.5 Structure of Manipulator Dynamics Equations

**State Space Equation**

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

- \([M(\Theta)]\) \(N \times N\) 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**

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

- \([M(\Theta)]\) \(N \times N\) mass matrix; symmetric and positive definite
- \([B(\Theta)]\) \(N \times \frac{N(N-1)}{2}\) Coriolis matrix
- \(\{\dot{\Theta}^2\}\) \(N \times 1\) \(\{\dot{\Theta}_1^2, \dot{\Theta}_2^2, \ldots, \dot{\Theta}_N^2\}\)^T
- \([C(\Theta)]\) \(N \times N\) centripetal matrix
- \(\{G(\Theta)\}\) \(N \times 1\) vector of gravity terms

**Cartesian State Space Equation**

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

- \([M_x(\Theta)]\) \(N \times N\) 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

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

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

Note
5.6 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.

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

\[
\tau_1 = \left( I_{zz1} + I_{zz2} + \frac{m_1 L_1^2}{4} + m_2 L_1^2 + m_2 L_1 L_2 c_2 + \frac{m_1 L_2^2}{4} \right) \dot{\theta}_1 + \left( I_{zz2} + \frac{m_2 L_1 L_2 c_2}{2} + \frac{m_2 L_2^2}{4} \right) \dot{\theta}_2
\]

\[+ \left( -\frac{m_2 L_1 L_2 c_2}{2} \right) \ddot{\theta}_2 + (-m_2 L_1 L_2 s_2) \dot{\theta}_1 \dot{\theta}_2 + \left( \frac{m_1 L_1 c_1}{2} + m_2 L_1 c_1 + \frac{m_2 L_2 c_{12}}{2} \right) g\]

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:

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

\[ \tau_1 = \tau_2 \]

\[ \ddot{\Theta} = \frac{I_{zz} + I_{xz} + \frac{m_1 L_2^2}{4}}{I_{zz} + \frac{m_1 L_2 c_2}{2} + \frac{m_2 L_2^2}{4}} \]

\[ \{ V(\Theta, \dot{\Theta}) \} = \begin{bmatrix} -\frac{m_2 L_1 s_2}{2} \dot{\theta}_2^2 + \left(-m_2 L_2 s_2 \right) \dot{\theta}_1 \dot{\theta}_2 \\ \frac{m_2 L_1 s_2}{2} \dot{\theta}_1^2 \end{bmatrix} \]

\[ \{ G(\Theta) \} = \begin{bmatrix} \frac{m_1 c_1}{2} + m_2 L_1 c_1 + \frac{m_2 L_1 c_{12}}{2} \\ \frac{m_2 L_1 c_{12}}{2} \end{bmatrix} \]

Configuration Space Representation

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

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

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

\[ [B(\Theta)] \{ \dot{\Theta} \} = \begin{bmatrix} -\frac{m_2 L_1 L_2 s_2}{2} \\ 0 \end{bmatrix} \dot{\theta}_1 \dot{\theta}_2 \]

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

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

The constant commanded Cartesian velocity is

$$0 \{\dot{X}\} = \begin{cases} \dot{x} \\ \dot{y} \end{cases} = \begin{bmatrix} 0 \\ 0.5 \end{bmatrix} \ (m/s)$$

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

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 $X$; $\phi$ must move to compensate for the pure $Y$ motion, but we cannot control $\phi$ 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

Cartesian Pose

Joint Angles

Joint Rates

Joint Accelerations

Joint Torques

Dynamics Results Ignoring Gravity

Dynamics Results Including Gravity
6. 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.

6.1 Inverse Pose Control Architecture

6.2 Inverse Velocity (Resolved-Rate) Control Architecture
6.3 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.

\[
\begin{align*}
\{\tau\} &= [M(\Theta)] \{\dot{\theta}\} + \{V(\Theta, \dot{\theta})\} + \{F(\Theta, \dot{\theta})\} + \{G(\Theta)\} \\
\{\tau\} & \quad \text{vector of applied joint torques} \\
[M(\Theta)] & \quad N \times N \text{ mass matrix; symmetric and positive definite} \\
\{V(\Theta, \dot{\theta})\} & \quad N \times 1 \text{ vector of Coriolis and centripetal terms} \\
\{F(\Theta, \dot{\theta})\} & \quad N \times 1 \text{ vector of friction torques} \\
\{G(\Theta)\} & \quad N \times 1 \text{ vector of gravity terms} \\
\{\Theta\}, \{\dot{\Theta}\}, \{\ddot{\Theta}\} & \quad N \times 1 \text{ joint angles, rates, and accelerations vectors}
\end{align*}
\]

**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.
6.4 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](image1)
![Dual PUMA Robots with Space Station Task](image2)

**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](image3)
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.
6.5 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

---

6.6 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
- \( n \gg 1 \) (the large gear ratio \( n \) reduces speed and increases torque)

Real-world vs. model characteristics

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

Motor

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

**Gear ratio** $n$

Substituting (6) into (5) yields (7)
Reflect load inertia and damping to motor shaft

Substitute (8) into (4)

Substitute (8) into (7) to eliminate $\tau_L$

Combine terms

Define

$$J_E = J_M + J_L \frac{n^2}{n^2}$$

effective inertia, total reflected to motor shaft

$$C_E = C_M + C_L \frac{n^2}{n^2}$$

effective damping, total reflected to motor shaft
Final Mechanical Model (ODE)

This is a second-order ODE in $\theta_M$. Can also be written as first-order ODE in $\omega_M$.

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