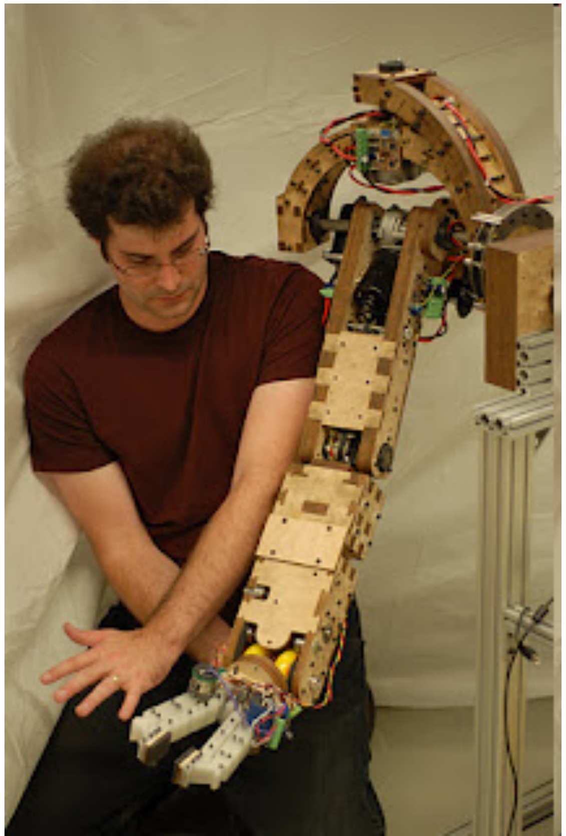
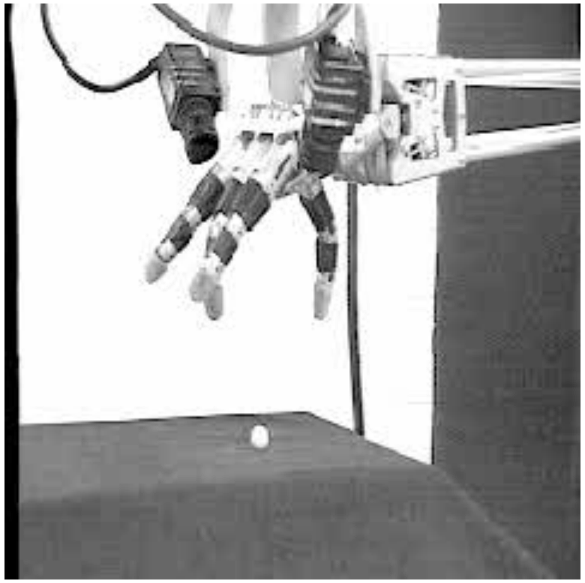
Robot Arms, Hands: Kinematics



With slides from Renata Melamud

Kinematics studies the motion of bodies

What a robot arm and hand can do



- Martin 1992-’97 PhD work

What a robot arm and hand can do

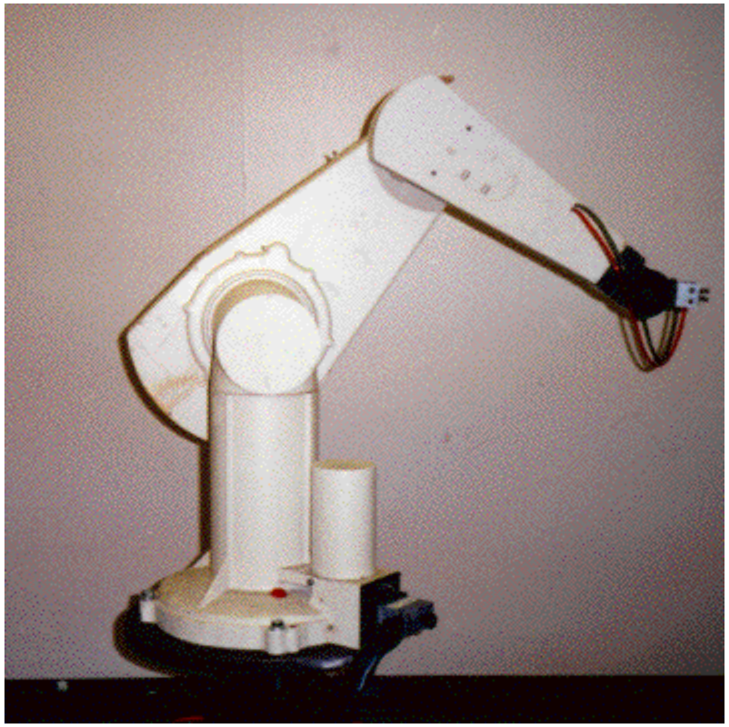
- Camilo 2011-? PhD work

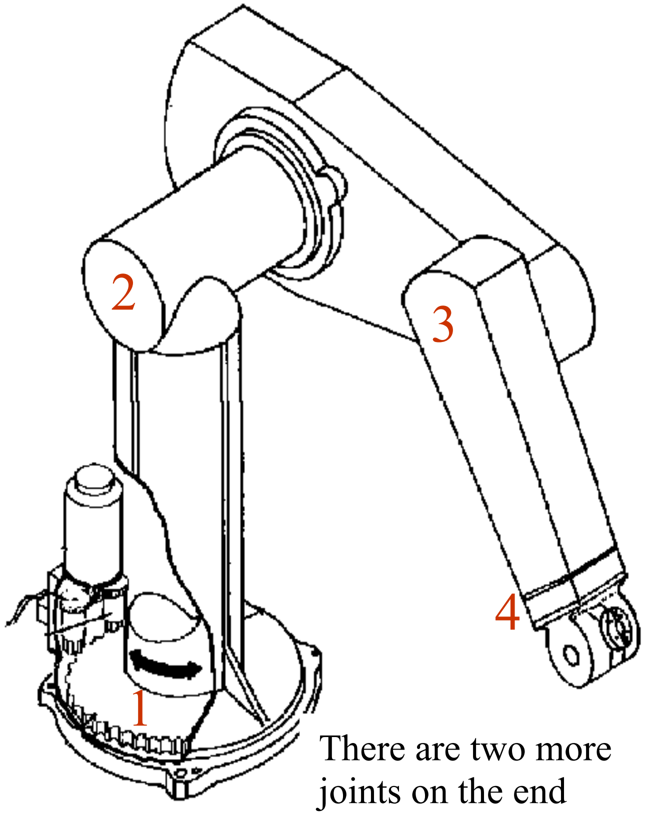
Robotics field

- 6 Million mobile robots
  - From $100 roomba to $millions Mars rovers
- 1 million robot arms
  - Usually $20,000-100,000, some millions
- Value of industrial robotics: $25 billion
- Arms crucial for these industries:
  - Automotive (Welding, painting, some assembly)
  - Electronics (Placing tiny components on PCB)
  - General: Pack boxes, move parts from conveyor to machines

http://www.youtube.com/watch?v=DG6A1Bsi-lg

An classic arm - The PUMA 560



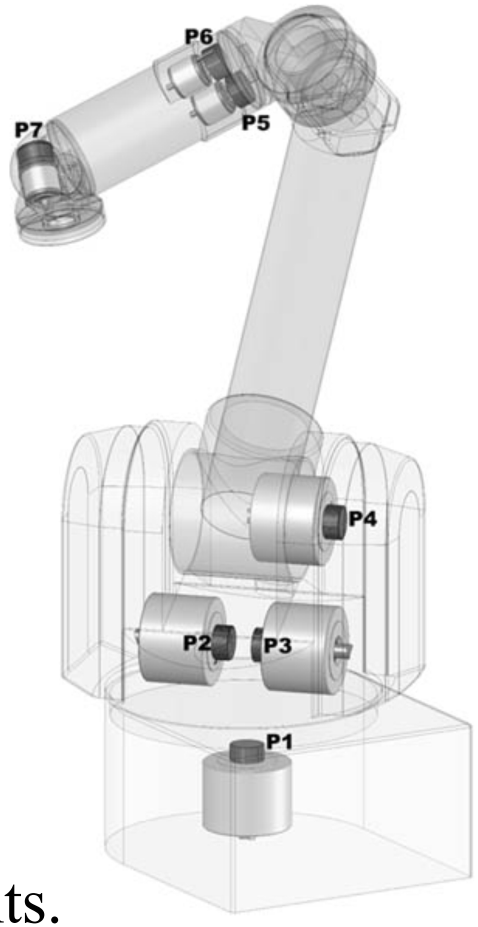


There are two more joints on the end effector (the gripper)

The PUMA 560 has SIX revolute joints

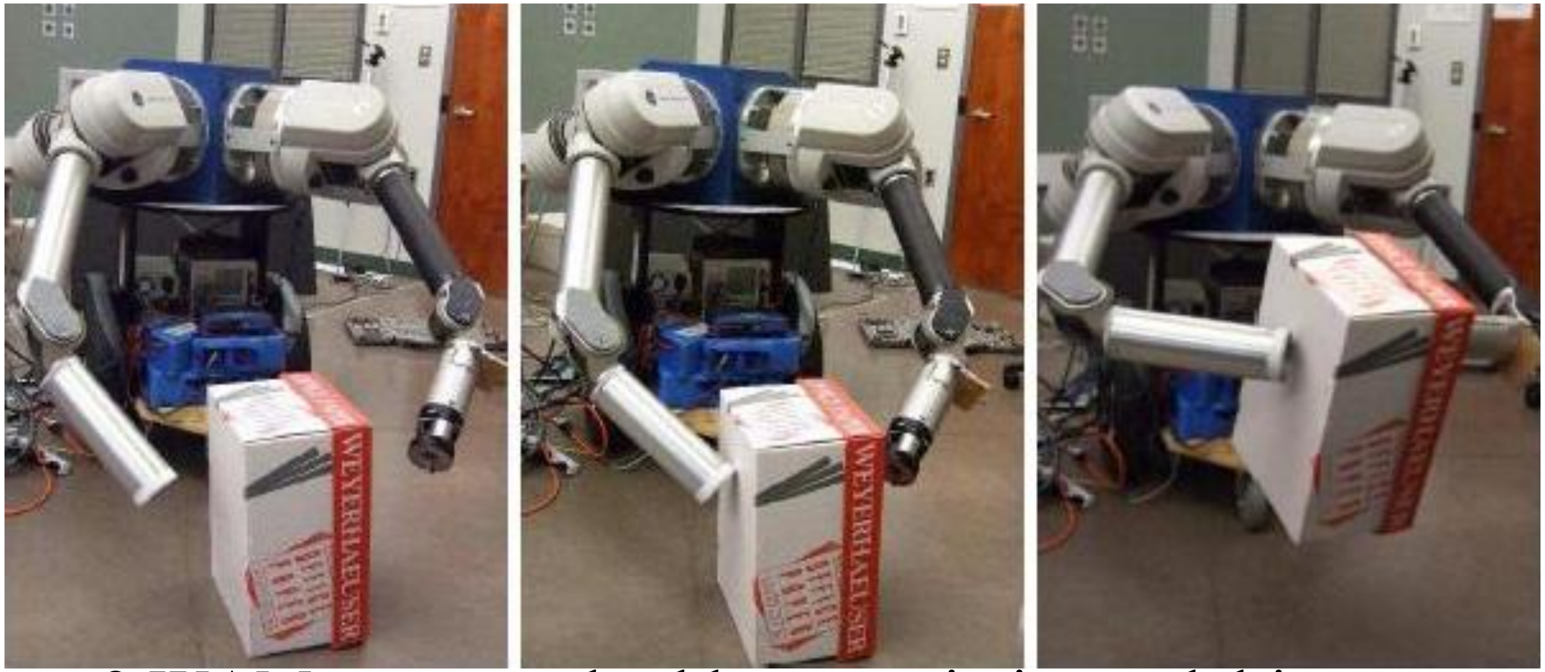
A revolute joint has ONE degree of freedom (1 DOF) that is defined by its angle

An modern arm - The Barrett WAM



- The WAM has SEVEN revolute joints.
- Similar motion (Kinematics) to human

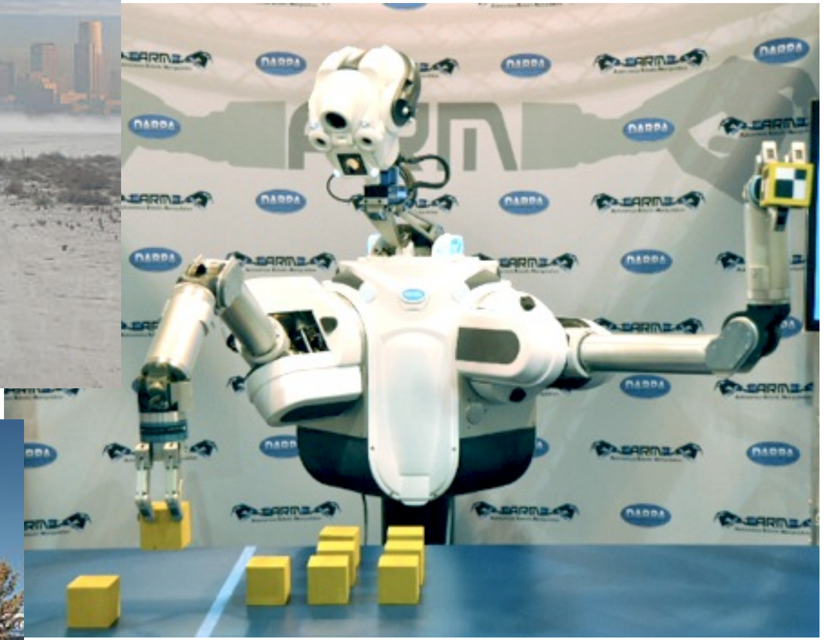
UA Robotics Lab platform
2 arm mobile manipulator

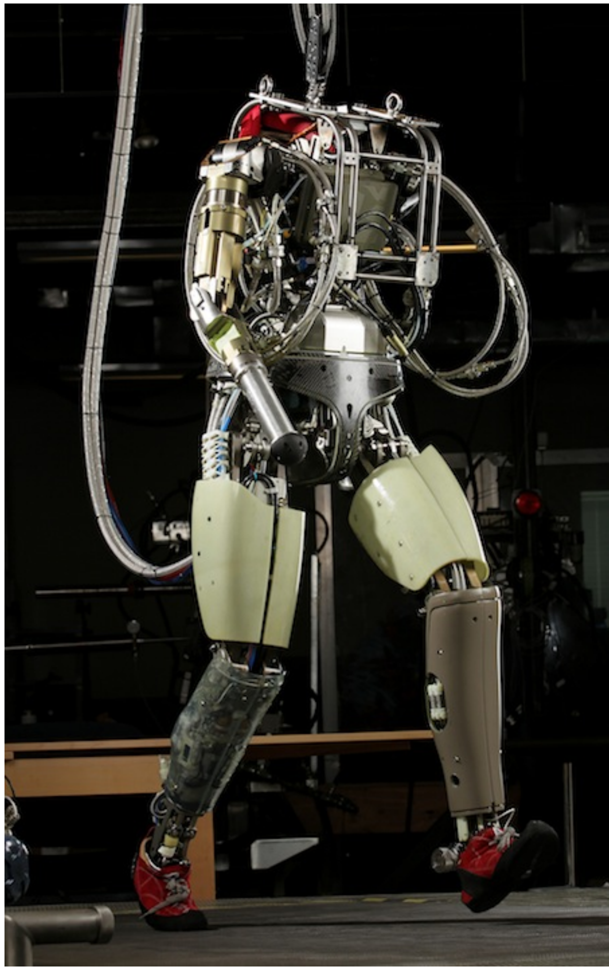


- 2 WAM arms, steel cable transmission and drive
- Segway mobile platform
- 2x Quad core computer platform.
- Battery powered, 4h run time.

Robotics challenges

•





Manipulation ‘11-14

Navigation ‘05

Humanoids ’12-

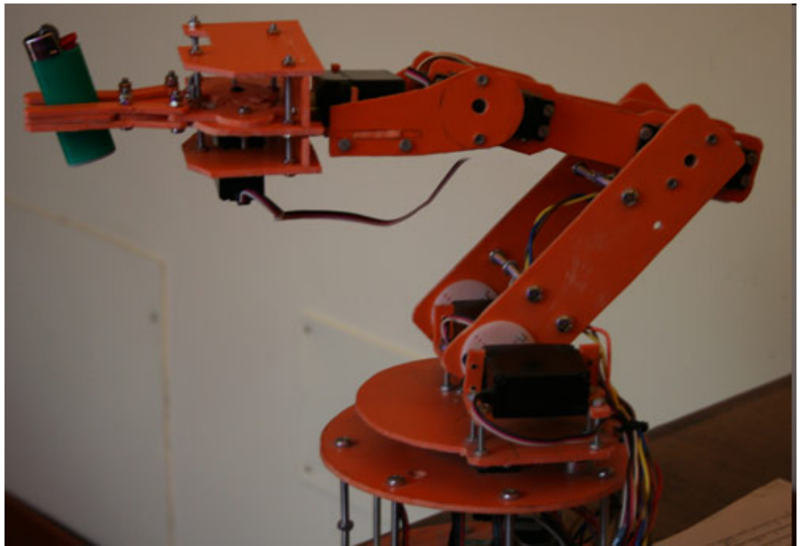
Build or buy?

- Off the shelf kits:

Lego

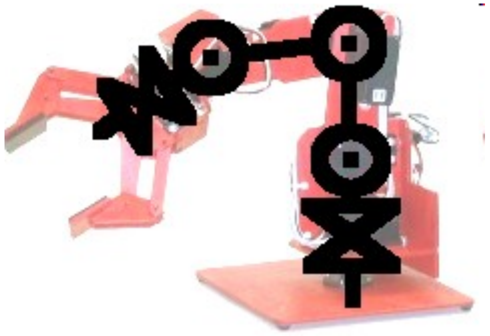
Lynxmotion

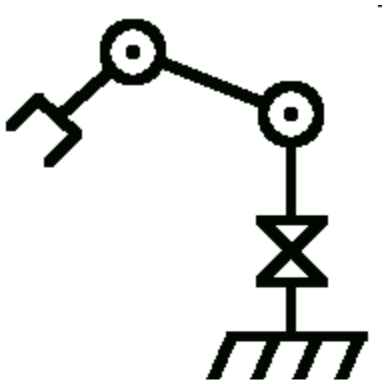
- Build your own:



Mathematical modeling

Robot





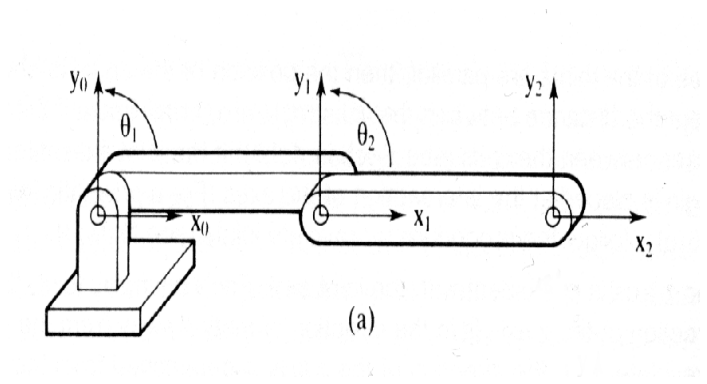
Abstract model

Strategy:

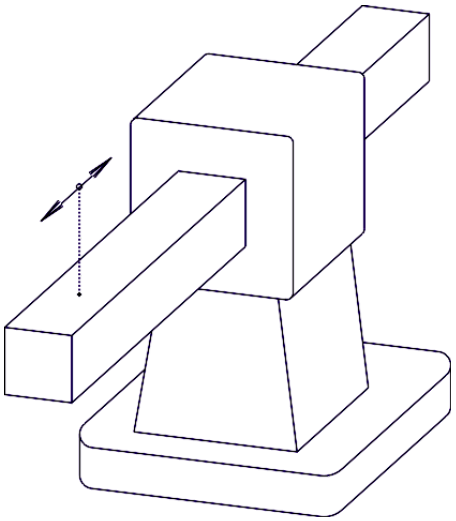
1. Model each joint separately
2. Combine joints and linkage lengths

http://www.societyofrobots.com/robot_arm_tutorial.shtml

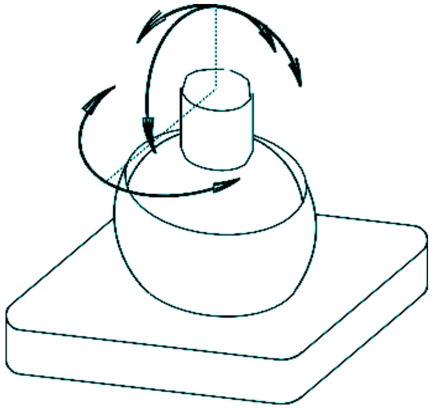
Other basic joints



Revolute Joint
1 DOF (Variable - ▯)



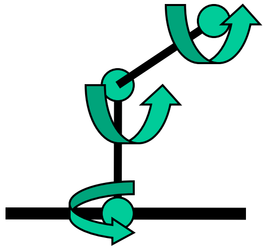
Prismatic Joint
1 DOF (linear) (Variables - d)



Spherical Joint
3 DOF (Variables - ▯$_1$, ▯$_2$, ▯$_3$)

Example
Matlab robot

Successive
translation and
rotation

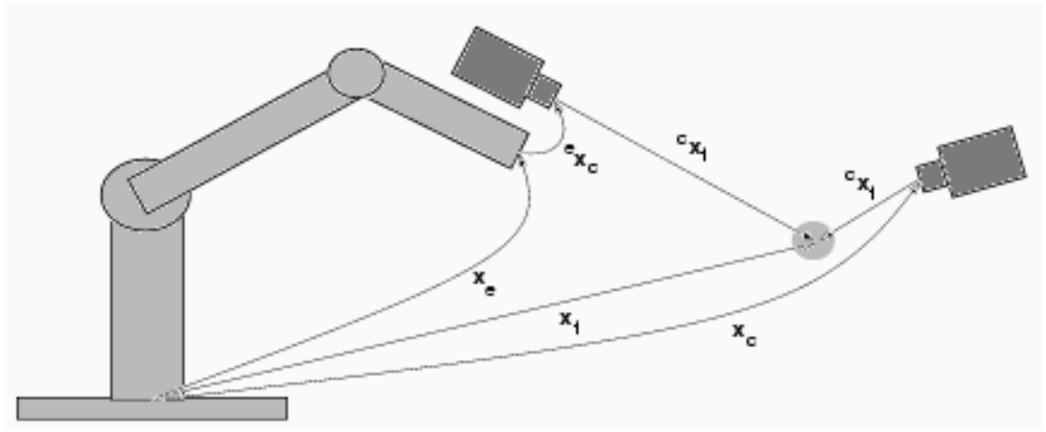


```
% robocop  Simulates a 3 joint robot

function Jpos =
    robocop(theta1,theta2,theta3,L1,L2,L3,P0)

Rxy1 = [cos(theta1) sin(theta1) 0
          -sin(theta1) cos(theta1) 0
          0                   0      1];

Rxz2 = [cos(theta2) 0 sin(theta2)
          0              1    0
          -sin(theta2) 0 cos(theta2)];

Rxz3 = [cos(theta3) 0 sin(theta3)
          0              1    0
          -sin(theta3) 0 cos(theta3)];

P1 = P0 + Rxy1*[L1 0 0]';
P2 = P1 + Rxy1*Rxz2*[L2 0 0]';
P3 = P2 + Rxy1*Rxz2*Rxz3*[L3 0 0]';
Jpos = [P0 P1 P2 P3];
```

Problem: Lots of coordinate frames to calibrate

Robot

- Base frame
- End-effector frame
- Object



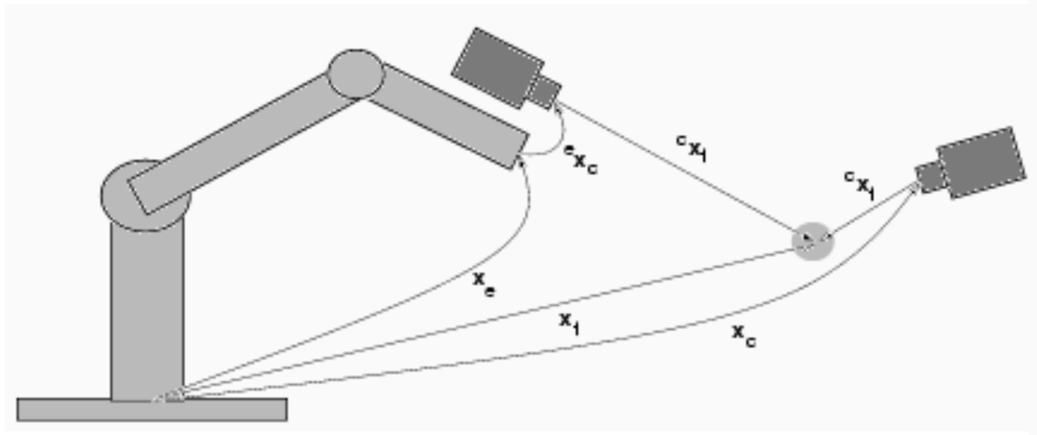
Problem: Lots of coordinate frames to calibrate

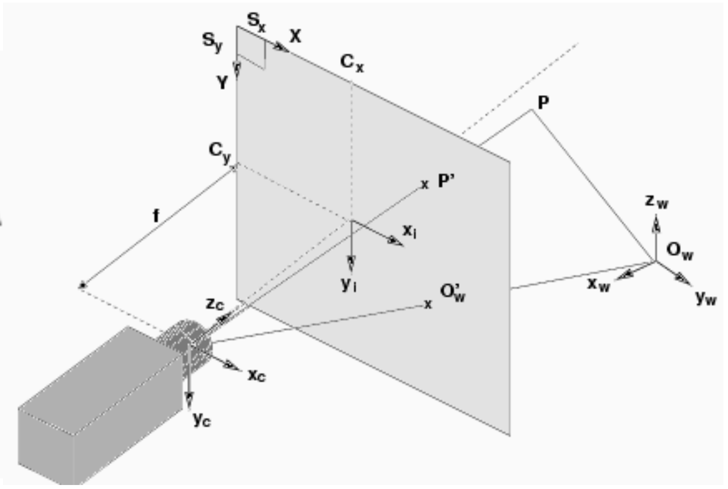
Robot

- Base frame
- End-effector frame
- Object

Camera

- Center of projection
- Different models





We are interested in two kinematics topics

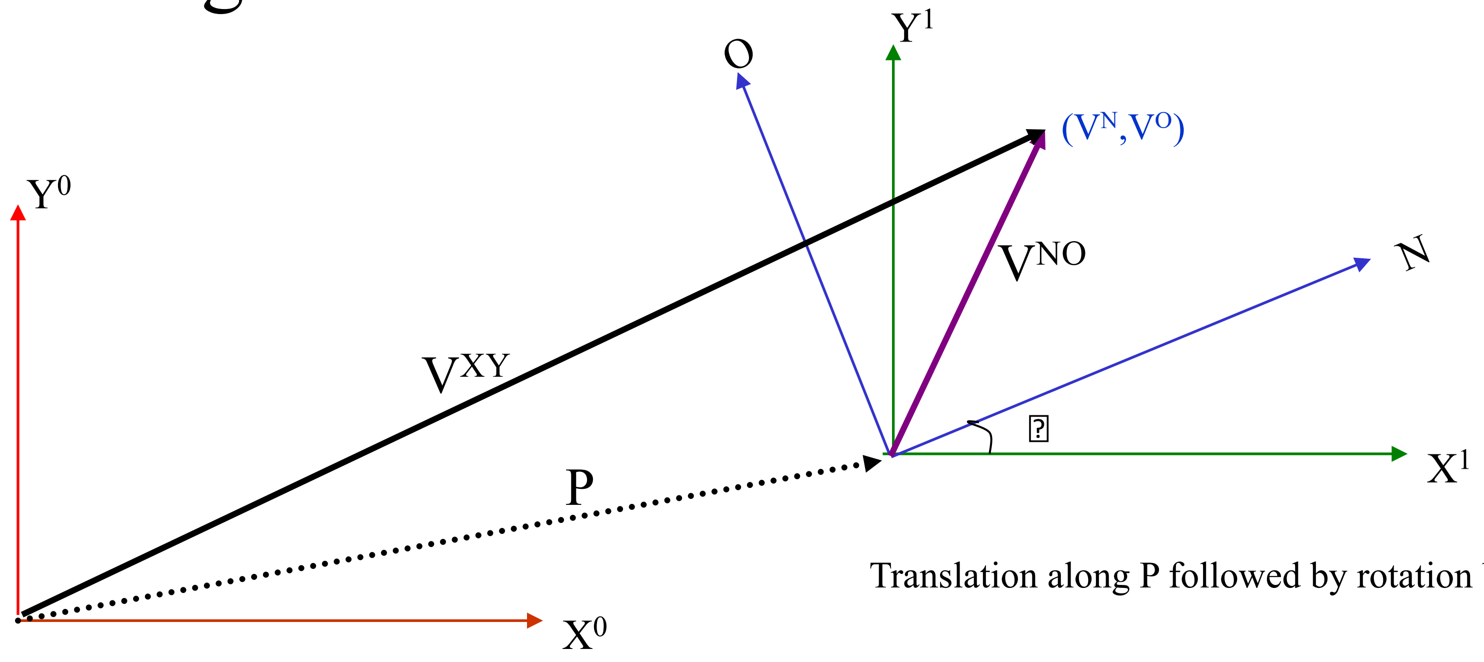
Forward Kinematics (angles to position)

What you are given:	The length of each link
	The angle of each joint
What you can find:	The position of any point (i.e. it's (x, y, z) coordinates

Inverse Kinematics (position to angles)

What you are given:	The length of each link
	The position of some point in the world (reachable)
What you can find:	The angles of each joint needed to obtain that position

Change Coordinate Frame



Translation along P followed by rotation by θ

$$V^{XY} = \begin{bmatrix} V^X \\ V^Y \end{bmatrix} = \begin{bmatrix} P_x \\ P_y \end{bmatrix} + \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} V^N \\ V^O \end{bmatrix}$$

(Note : P_x, P_y are relative to the original coordinate frame. Translation followed by rotation is different than rotation followed by translation.)

In other words, knowing the coordinates of a point (V^N,V^O) in some coordinate frame (NO) you can find the position of that point relative to your original coordinate frame (X^0Y^0).

HOMOGENEOUS REPRESENTATION

Putting it all into a Matrix

$$\mathbf{V}^{XY} = \begin{bmatrix} \mathbf{V}^X \\ \mathbf{V}^Y \end{bmatrix} = \begin{bmatrix} \mathbf{P}_x \\ \mathbf{P}_y \end{bmatrix} + \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \mathbf{V}^N \\ \mathbf{V}^O \end{bmatrix}$$

What we found by doing a translation and a rotation

$$= \begin{bmatrix} \mathbf{V}^X \\ \mathbf{V}^Y \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{P}_x \\ \mathbf{P}_y \\ 1 \end{bmatrix} + \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{V}^N \\ \mathbf{V}^O \\ 1 \end{bmatrix}$$

Padding with 0's and 1's

$$= \begin{bmatrix} \mathbf{V}^X \\ \mathbf{V}^Y \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & \mathbf{P}_x \\ \sin\theta & \cos\theta & \mathbf{P}_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{V}^N \\ \mathbf{V}^O \\ 1 \end{bmatrix}$$

Simplifying into a matrix form

$$\mathbf{H} = \begin{bmatrix} \cos\theta & -\sin\theta & \mathbf{P}_x \\ \sin\theta & \cos\theta & \mathbf{P}_y \\ 0 & 0 & 1 \end{bmatrix}$$

Homogenous Matrix for a Translation in XY plane, followed by a Rotation around the z-axis

Rotation Matrices in 3D – OK,lets return from homogenous repn

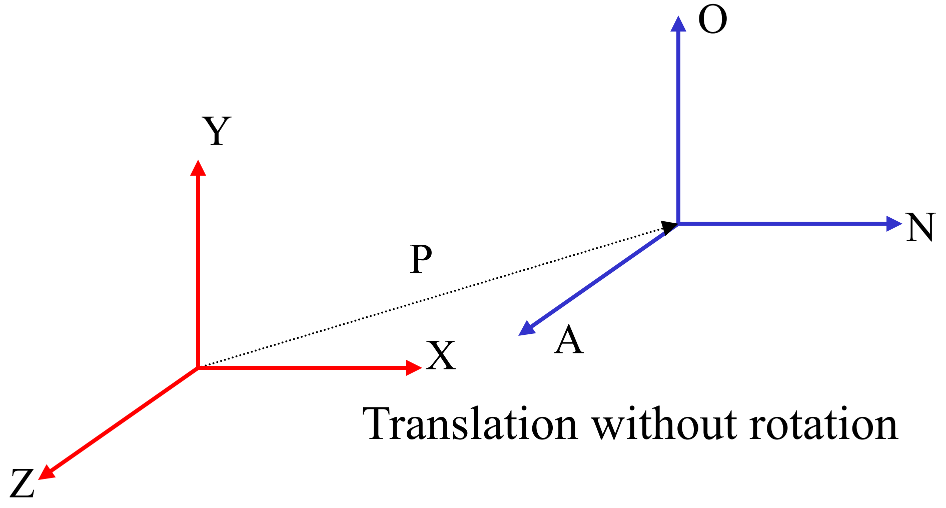
$$\mathbf{R_z} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \longleftarrow \text{Rotation around the Z-Axis}$$

$$\mathbf{R_y} = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \longleftarrow \text{Rotation around the Y-Axis}$$

$$\mathbf{R_z} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \longleftarrow \text{Rotation around the X-Axis}$$

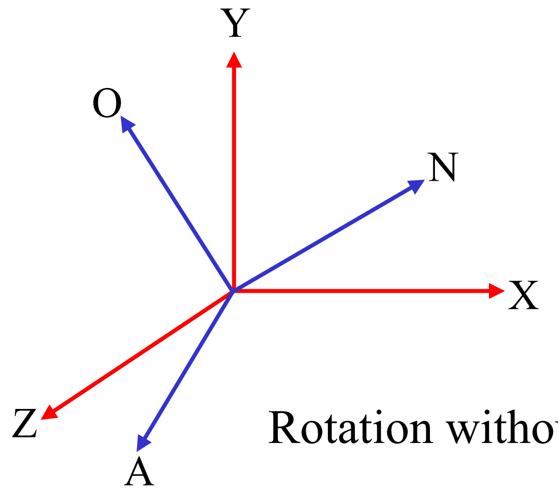
Homogeneous Matrices in 3D

H is a 4x4 matrix that can describe a translation, rotation, or both in one matrix



Translation without rotation

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & P_x \\ 0 & 1 & 0 & P_y \\ 0 & 0 & 1 & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Rotation without translation

$$\mathbf{H} = \begin{bmatrix} n_x & o_x & a_x & 0 \\ n_y & o_y & a_y & 0 \\ n_z & o_z & a_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation part:

Could be rotation around z-axis, x-axis, y-axis or a combination of the three.

Homogeneous Continued….

$$V^{XY} = H \begin{bmatrix} V^N \\ V^O \\ V^A \\ 1 \end{bmatrix}$$

The (n,o,a) position of a point relative to the current coordinate frame you are in.

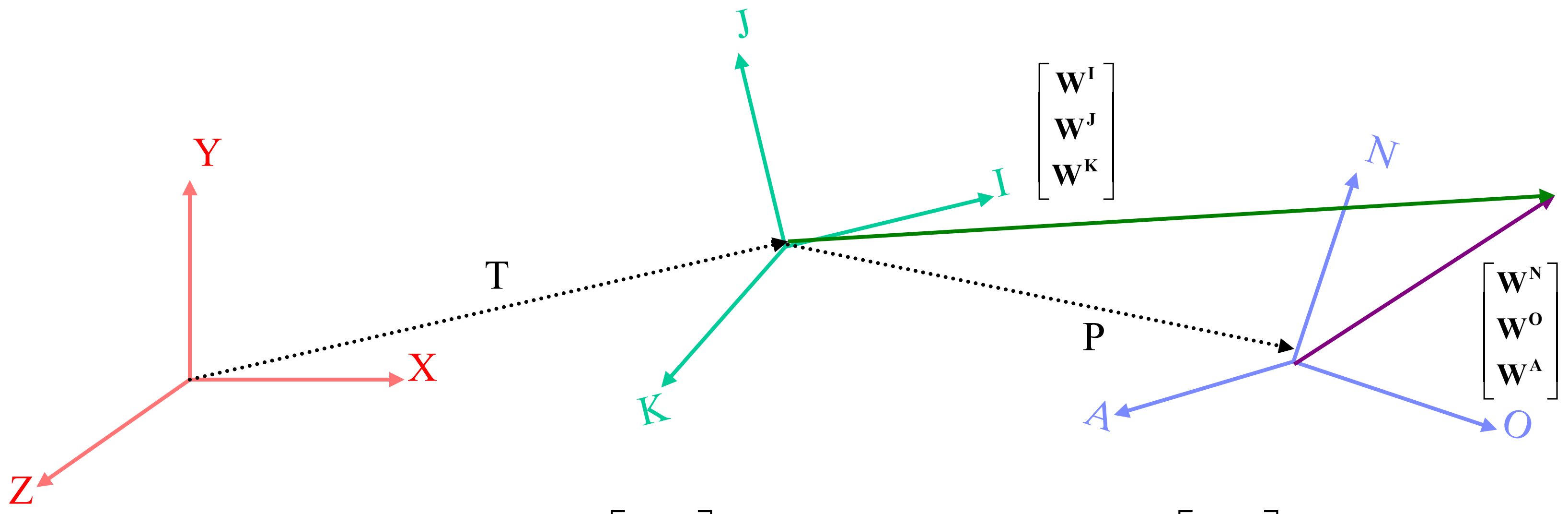
$$V^{XY} = \begin{bmatrix} n_x & o_x & a_x & P_x \\ n_y & o_y & a_y & P_y \\ n_z & o_z & a_z & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} V^N \\ V^O \\ V^A \\ 1 \end{bmatrix}$$

$$V^X = n_x V^N + o_x V^O + a_x V^A + P_x$$

The rotation and translation part can be combined into a single homogeneous matrix IF and ONLY IF both are relative to the same coordinate frame.

Finding the Homogeneous Matrix

EX.



J
I
Y
X
Z
T
K
P
N
A
O
$\begin{bmatrix} W^I \\ W^J \\ W^K \end{bmatrix}$
$\begin{bmatrix} W^N \\ W^O \\ W^A \end{bmatrix}$

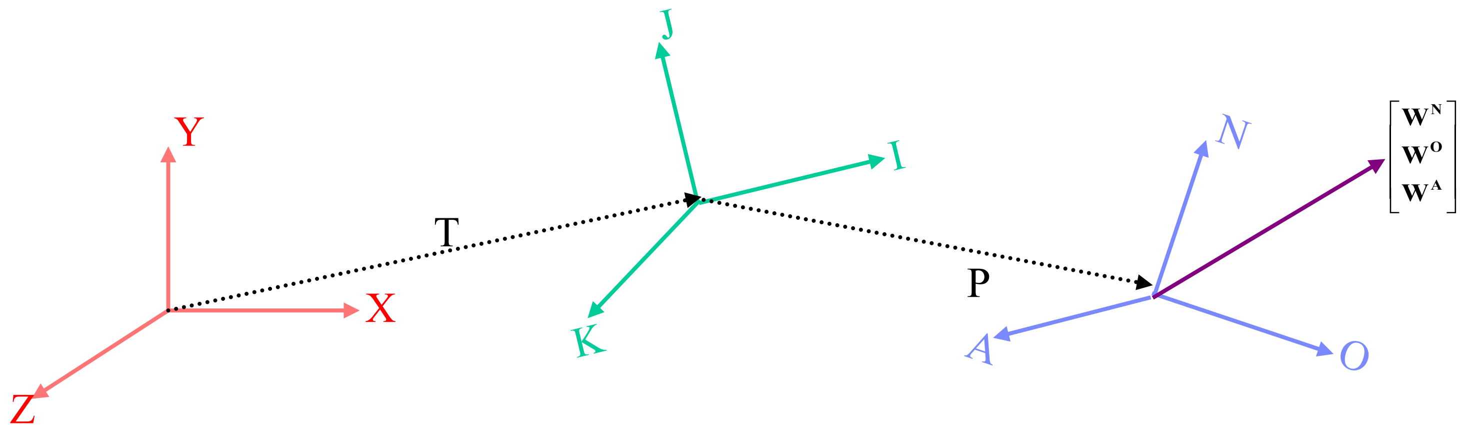
$\begin{bmatrix} W^X \\ W^Y \\ W^Z \end{bmatrix}$ Point relative to the X-Y-Z frame

$\begin{bmatrix} W^I \\ W^J \\ W^K \end{bmatrix}$ Point relative to the I-J-K frame

$\begin{bmatrix} W^N \\ W^O \\ W^A \end{bmatrix}$ Point relative to the N-O-A frame

$$\begin{bmatrix} W^I \\ W^J \\ W^K \end{bmatrix} = \begin{bmatrix} P_i \\ P_j \\ P_k \end{bmatrix} + \begin{bmatrix} n_i & o_i & a_i \\ n_j & o_j & a_j \\ n_k & o_k & a_k \end{bmatrix} \begin{bmatrix} W^N \\ W^O \\ W^A \end{bmatrix}$$

$$\begin{bmatrix} W^I \\ W^J \\ W^K \\ 1 \end{bmatrix} = \begin{bmatrix} n_i & o_i & a_i & P_i \\ n_j & o_j & a_j & P_j \\ n_k & o_k & a_k & P_k \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} W^N \\ W^O \\ W^A \\ 1 \end{bmatrix}$$



$$\begin{bmatrix} W^X \\ W^Y \\ W^Z \end{bmatrix} = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} + \begin{bmatrix} i_x & j_x & k_x \\ i_y & j_y & k_y \\ i_z & j_z & k_z \end{bmatrix} \begin{bmatrix} W^I \\ W^J \\ W^k \end{bmatrix} \longrightarrow \begin{bmatrix} W^X \\ W^Y \\ W^Z \\ 1 \end{bmatrix} = \begin{bmatrix} i_x & j_x & k_x & T_x \\ i_y & j_y & k_y & T_y \\ i_z & j_z & k_z & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} W^I \\ W^J \\ W^K \\ 1 \end{bmatrix}$$

Substituting for $\begin{bmatrix} W^I \\ W^J \\ W^K \end{bmatrix}$

$$\begin{bmatrix} W^X \\ W^Y \\ W^Z \\ 1 \end{bmatrix} = \begin{bmatrix} i_x & j_x & k_x & T_x \\ i_y & j_y & k_y & T_y \\ i_z & j_z & k_z & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} n_i & o_i & a_i & P_i \\ n_j & o_j & a_j & P_j \\ n_k & o_k & a_k & P_k \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} W^N \\ W^O \\ W^A \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} W^X \\ W^Y \\ W^Z \\ 1 \end{bmatrix} = H \begin{bmatrix} W^N \\ W^O \\ W^A \\ 1 \end{bmatrix}$$

$$H = \begin{bmatrix} i_x & j_x & k_x & T_x \\ i_y & j_y & k_y & T_y \\ i_z & j_z & k_z & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} n_i & o_i & a_i & P_i \\ n_j & o_j & a_j & P_j \\ n_k & o_k & a_k & P_k \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

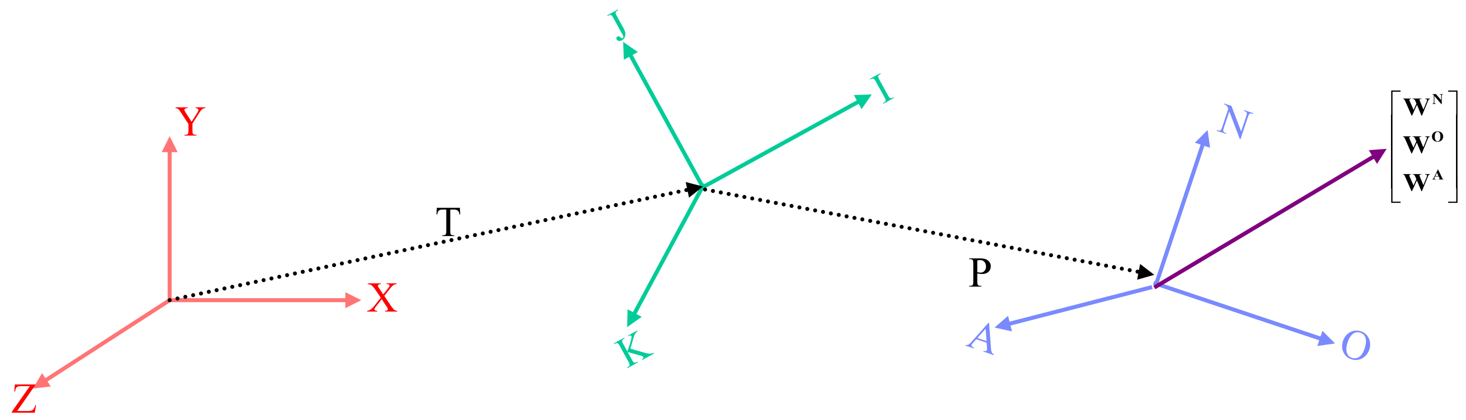
$\longrightarrow$ Product of the two matrices

Notice that H can also be written as:

$$H = \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} i_x & j_x & k_x & 0 \\ i_y & j_y & k_y & 0 \\ i_z & j_z & k_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & P_i \\ 0 & 1 & 0 & P_j \\ 0 & 0 & 1 & P_k \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} n_i & o_i & a_i & 0 \\ n_j & o_j & a_j & 0 \\ n_k & o_k & a_k & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

H = (Translation relative to the XYZ frame) * (Rotation relative to the XYZ frame)
* (Translation relative to the IJK frame) * (Rotation relative to the IJK frame)

The Homogeneous Matrix is a concatenation of numerous translations and rotations



One more variation on finding H:

H = (Rotate so that the X-axis is aligned with T)

* (Translate along the new t-axis by || T || (magnitude of T))

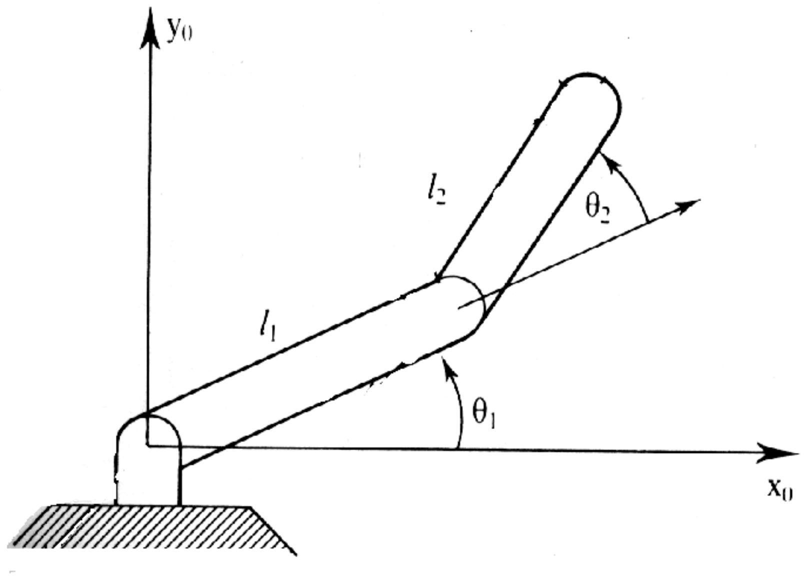
* (Rotate so that the t-axis is aligned with P)

* (Translate along the p-axis by || P ||)

* (Rotate so that the p-axis is aligned with the O-axis)

This method might seem a bit confusing, but it's actually an easier way to solve our problem given the information we have. Here is an example…

Forward Kinematics



The Situation:

You have a robotic arm that starts out aligned with the x_o-axis. You tell the first link to move by ⍰$_1$ and the second link to move by ⍰$_2$.

The Quest:

What is the position of the end of the robotic arm?

Solution:

1. Geometric Approach

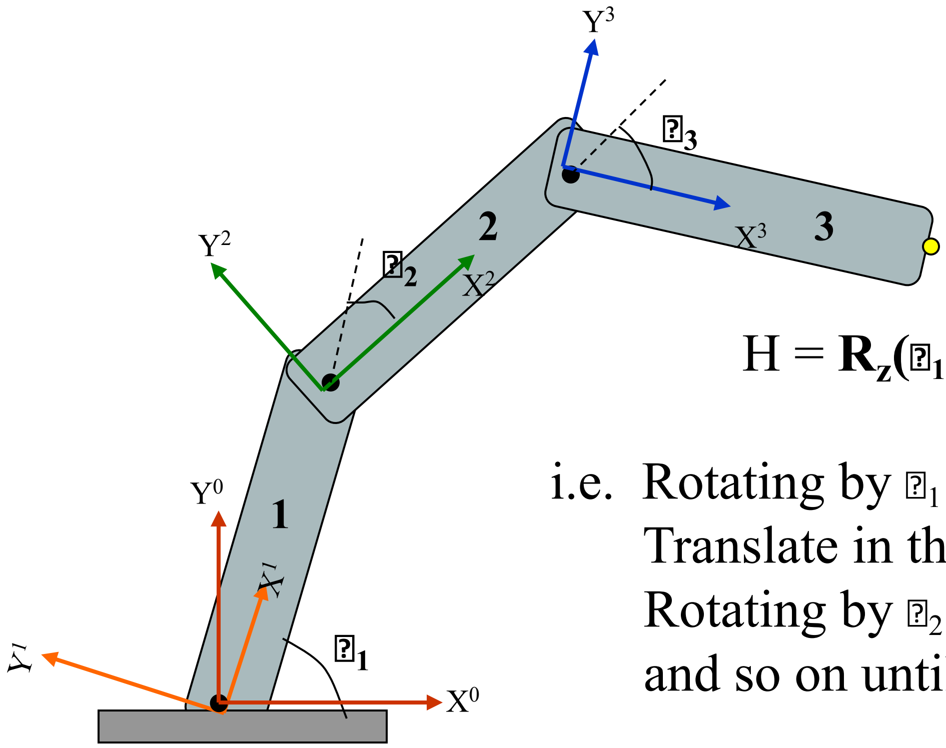
This might be the easiest solution for the simple situation. However, notice that the angles are measured relative to the direction of the previous link. (The first link is the exception. The angle is measured relative to it’s initial position.) For robots with more links and whose arm extends into 3 dimensions the geometry gets much more tedious.

2. Algebraic Approach

Involves coordinate transformations.

Example Problem:

You are have a three link arm that starts out aligned in the x-axis. Each link has lengths l_1, l_2, l_3, respectively. You tell the first one to move by θ_1, and so on as the diagram suggests. Find the Homogeneous matrix to get the position of the yellow dot in the X^0Y^0 frame.



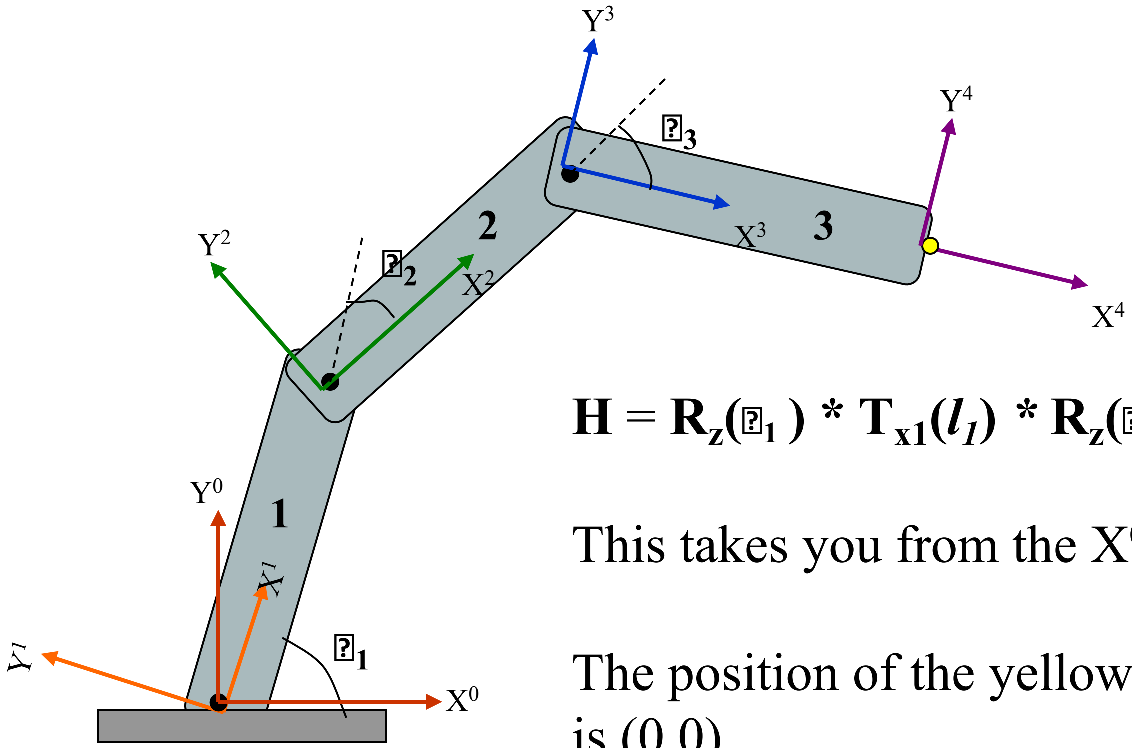
$$H = \mathbf{R_z(\theta_1)} * \mathbf{T_{x1}(\mathit{l_1})} * \mathbf{R_z(\theta_2)} * \mathbf{T_{x2}(\mathit{l_2})} * \mathbf{R_z(\theta_3)}$$

i.e. Rotating by θ_1 will put you in the X^1Y^1 frame.
Translate in the along the X^1 axis by l_1.
Rotating by θ_2 will put you in the X^2Y^2 frame.
and so on until you are in the X^3Y^3 frame.

The position of the yellow dot relative to the X^3Y^3 frame is $(\mathbf{\mathit{l_3}}, 0)$. Multiplying H by that position vector will give you the coordinates of the yellow point relative the the X^0Y^0 frame.

Slight variation on the last solution:

Make the yellow dot the origin of a new coordinate X^4Y^4 frame



$$\mathbf{H} = \mathbf{R_z(\theta_1) * T_{x1}(\mathit{l_1}) * R_z(\theta_2) * T_{x2}(\mathit{l_2}) * R_z(\theta_3) * T_{x3}(\mathit{l_3})}$$

This takes you from the X^0Y^0 frame to the X^4Y^4 frame.

The position of the yellow dot relative to the X^4Y^4 frame is (0,0).

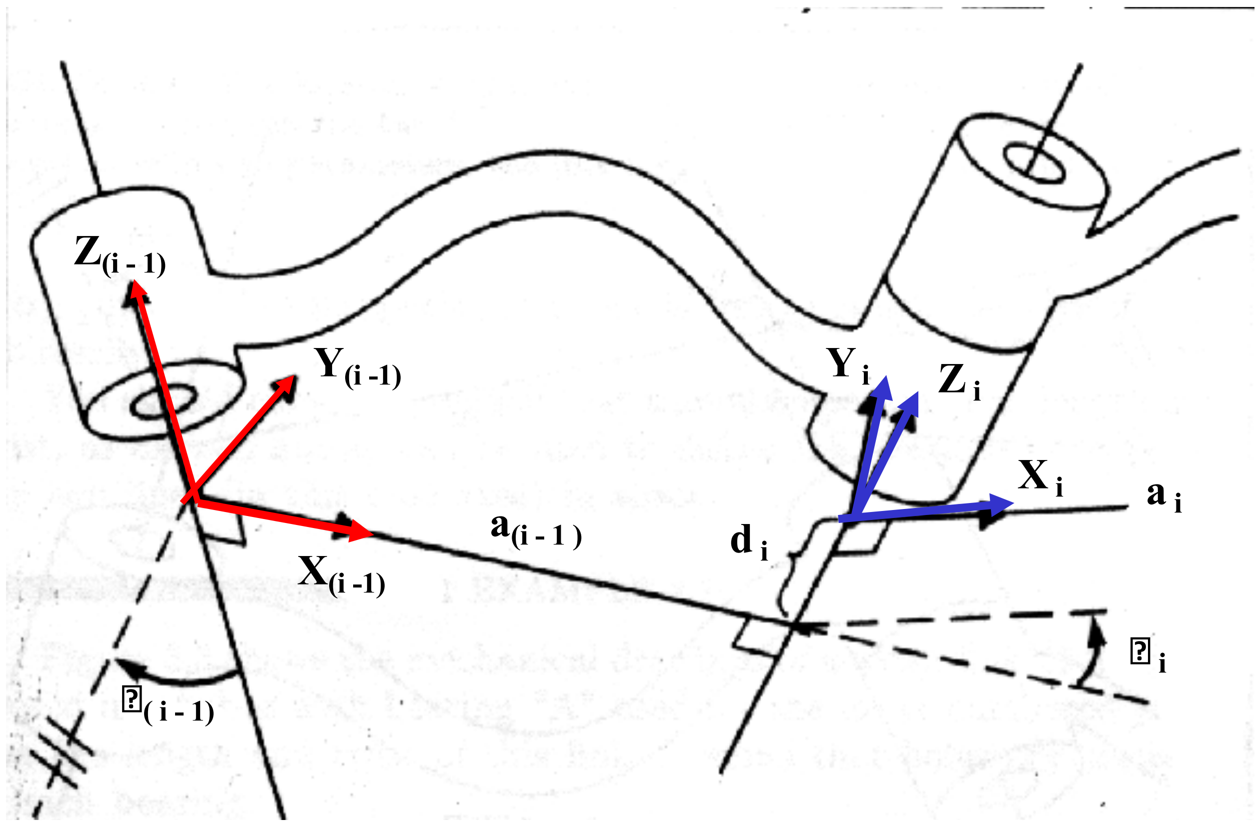
$$\begin{bmatrix} \mathbf{X} \\ \mathbf{Y} \\ \mathbf{Z} \\ \mathbf{1} \end{bmatrix} = \mathbf{H} \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{1} \end{bmatrix} \longleftarrow$$

Notice that multiplying by the (0,0,0,1) vector will equal the last column of the H matrix.

More on Forward Kinematics…

Denavit - Hartenberg Parameters

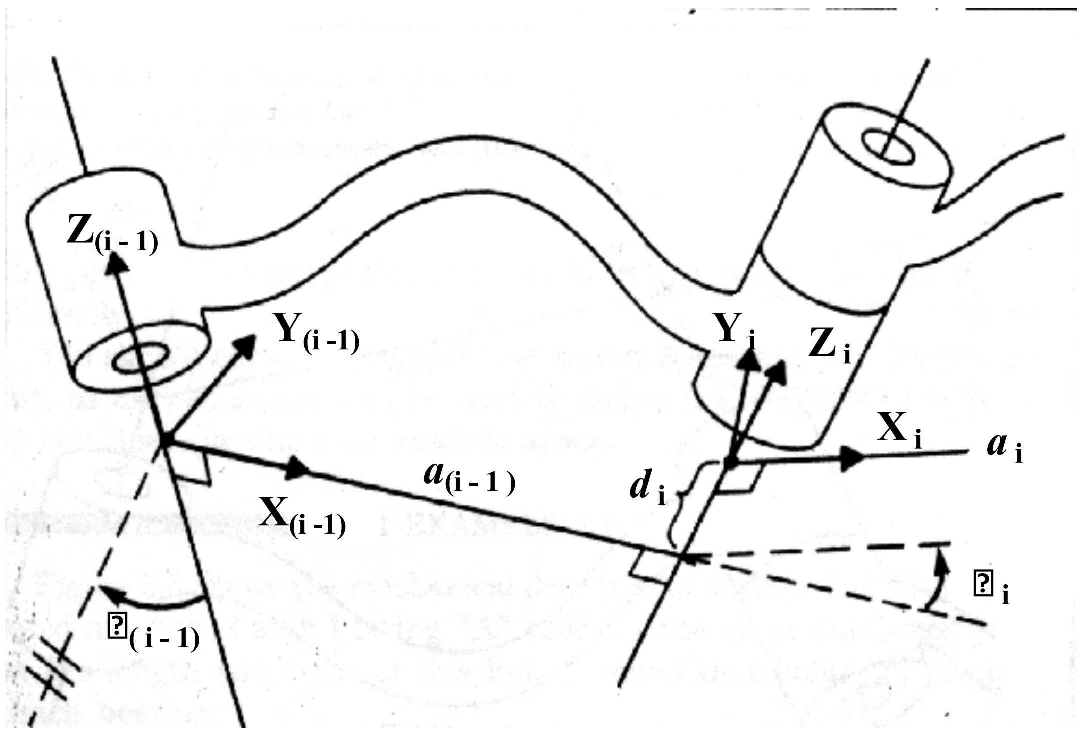
Denavit-Hartenberg Notation



IDEA: Each joint is assigned a coordinate frame. Using the Denavit-Hartenberg notation, you need 4 parameters to describe how a frame (*i*) relates to a previous frame (*i -1*).

THE PARAMETERS/VARIABLES: **⍰, *a* , *d*, ⍰**

The Parameters



You can align the two axis just using the 4 parameters

1) $a_{(i-1)}$

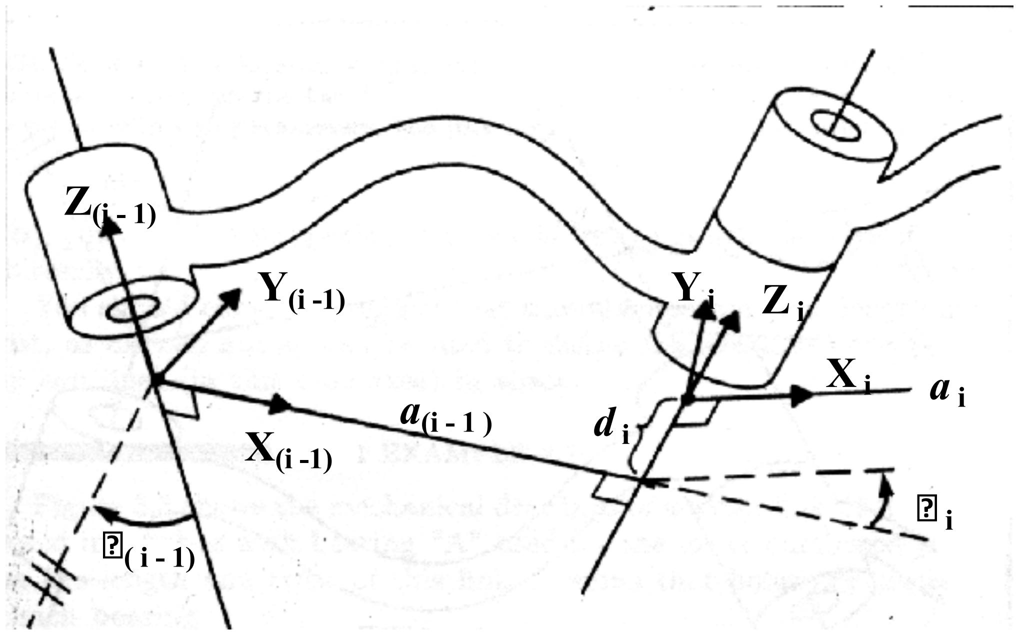
Technical Definition: $a_{(i-1)}$ is the length of the perpendicular between the joint axes. The joint axes is the axes around which revolution takes place which are the $Z_{(i-1)}$ and $Z_{(i)}$ axes. These two axes can be viewed as lines in space. The common perpendicular is the shortest line between the two axis-lines and is perpendicular to both axis-lines.

a ***(i-1) cont...***

Visual Approach - “A way to visualize the link parameter $\boldsymbol{a_{(i-1)}}$ is to imagine an expanding cylinder whose axis is the $Z_{(i-1)}$ axis - when the cylinder just touches the joint axis *i* the radius of the cylinder is equal to $\boldsymbol{a_{(i-1)}}$.” (Manipulator Kinematics)

It’s Usually on the Diagram Approach - If the diagram already specifies the various coordinate frames, then the common perpendicular is usually the $X_{(i-1)}$ axis. So $\boldsymbol{a_{(i-1)}}$ is just the displacement along the $X_{(i-1)}$ to move from the *(i-1)* frame to the *i* frame.

If the link is prismatic, then $\boldsymbol{a_{(i-1)}}$ is a variable, not a parameter.



2) ⍰$_{(i-1)}$

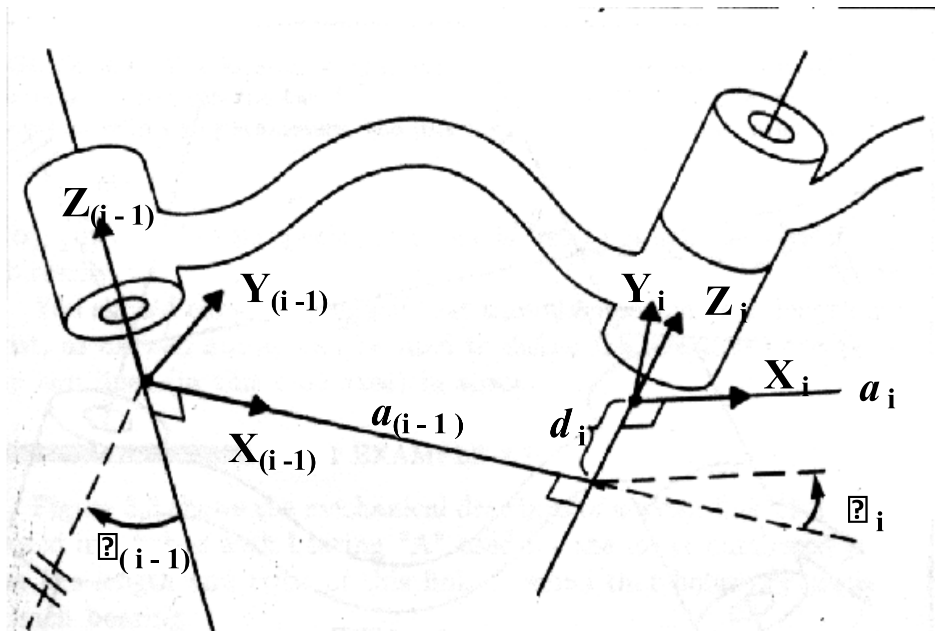
Technical Definition: Amount of rotation around the common perpendicular so that the joint axes are parallel.

i.e. How much you have to rotate around the $X_{(i-1)}$ axis so that the $Z_{(i-1)}$ is pointing in the same direction as the Z_i axis. Positive rotation follows the right hand rule.

3) $d_{(i-1)}$

Technical Definition: The displacement along the Z_i axis needed to align the $a_{(i-1)}$ common perpendicular to the a_i common perpendicular.

In other words, displacement along the Z_i to align the $X_{(i-1)}$ and X_i axes.



4) ⍰$_i$

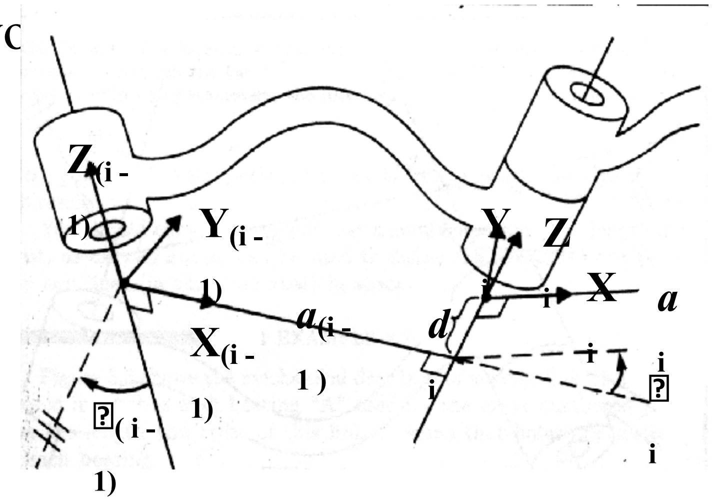
Amount of rotation around the Z_i axis needed to align the $X_{(i-1)}$ axis with the X_i axis.

The Denavit-Hartenberg Matrix

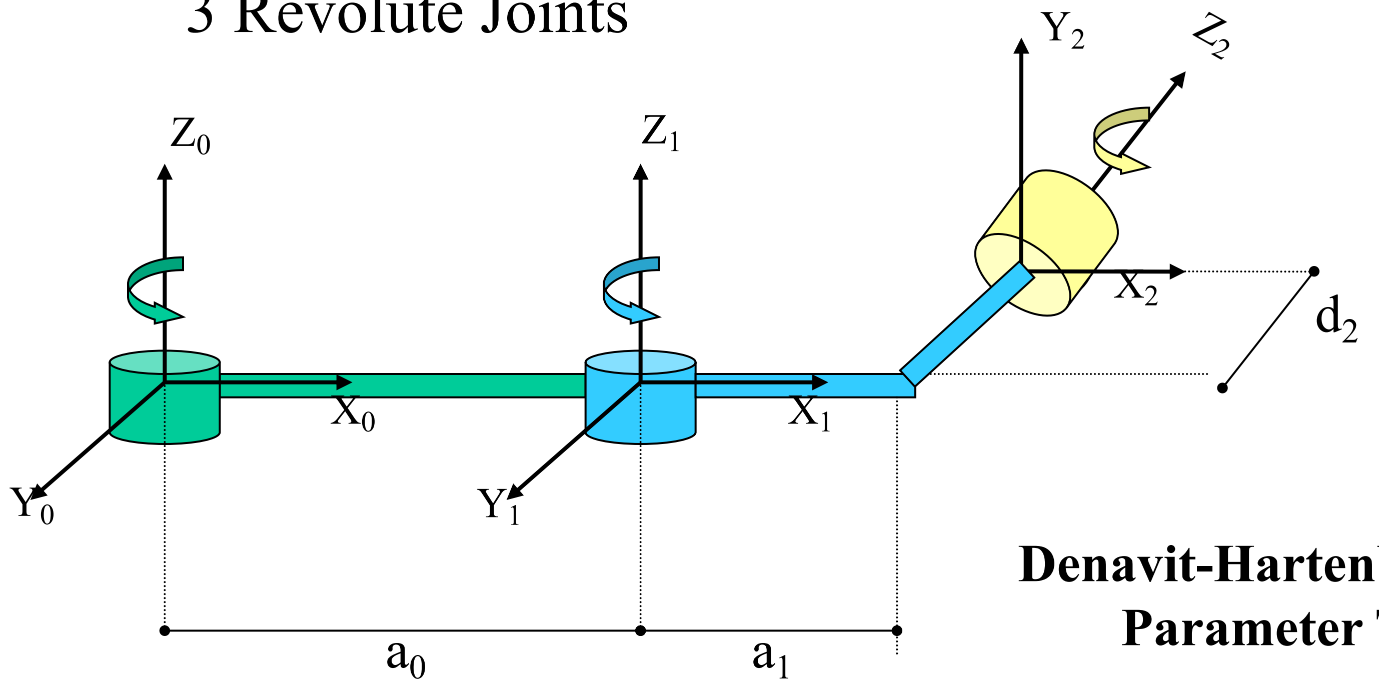
$$\begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & a_{(i-1)} \\ \sin\theta_i\cos\alpha_{(i-1)} & \cos\theta_i\cos\alpha_{(i-1)} & -\sin\alpha_{(i-1)} & -\sin\alpha_{(i-1)}d_i \\ \sin\theta_i\sin\alpha_{(i-1)} & \cos\theta_i\sin\alpha_{(i-1)} & \cos\alpha_{(i-1)} & \cos\alpha_{(i-1)}d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Just like the Homogeneous Matrix, the Denavit-Hartenberg Matrix is a transformation matrix from one coordinate frame to the next. Using a series of D-H Matrix multiplications and the D-H Parameter table, the final result is a transformation matrix from some frame to yc

Put the transformation here



3 Revolute Joints



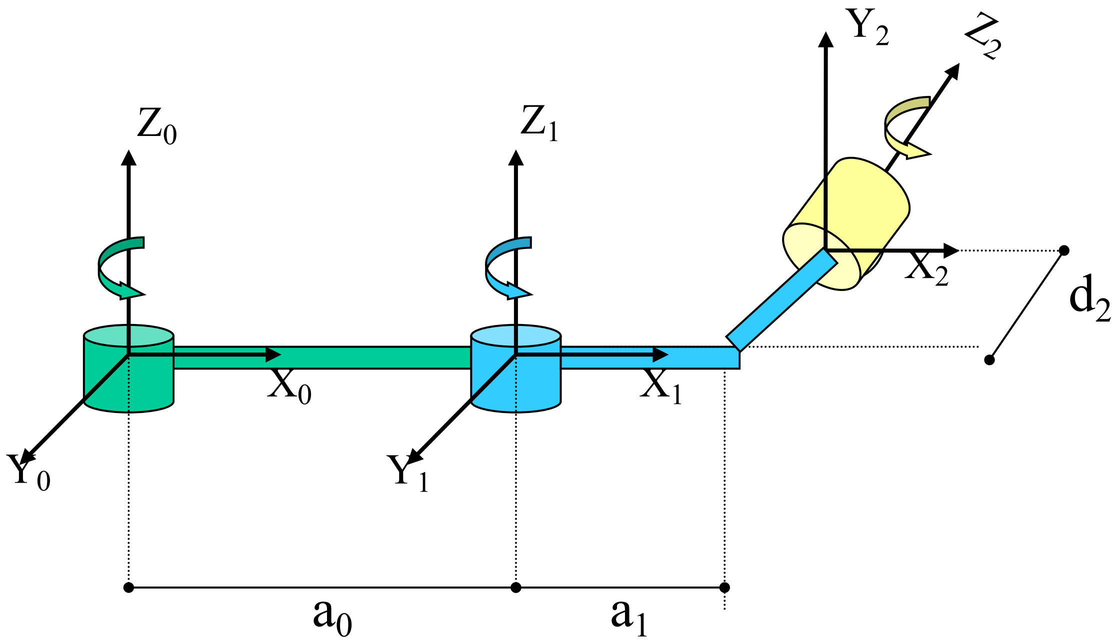
Denavit-Hartenberg Link Parameter Table

Notice that the table has two uses:

1) To describe the robot with its variables and parameters.

2) To describe some state of the robot by having a numerical values for the variables.

i	$\alpha_{(i-1)}$	$a_{(i-1)}$	d_i	θ_i
0	0	0	0	θ_0
1	0	a_0	0	θ_1
2	-90	a_1	d_2	θ_2



i	$\alpha_{(i-1)}$	$a_{(i-1)}$	d_i	θ_i
0	0	0	0	θ_0
1	0	a_0	0	θ_1
2	-90	a_1	d_2	θ_2

$$\mathbf{V}^{X_0Y_0Z_0} = \mathbf{T}\begin{bmatrix} \mathbf{V}^{X_2} \\ \mathbf{V}^{Y_2} \\ \mathbf{V}^{Z_2} \\ \mathbf{1} \end{bmatrix}$$

$$\mathbf{T} = ({}_{0}\mathbf{T})({}^{0}_{1}\mathbf{T})({}^{1}_{2}\mathbf{T})$$

Note: T is the D-H matrix with *(i-1)* = 0 and *i* = 1.

i	$\alpha_{(i-1)}$	$a_{(i-1)}$	d_i	θ_i
0	0	0	0	θ_0
1	0	a_0	0	θ_1
2	-90	a_1	d_2	θ_2

$$
{}_0\mathbf{T} = \begin{bmatrix} \cos\theta_0 & -\sin\theta_0 & 0 & 0 \\ \sin\theta_0 & \cos\theta_0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

This is just a rotation around the Z_0 axis

$$
{}^0_1\mathbf{T} = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & a_0 \\ \sin\theta_1 & \cos\theta_1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

This is a translation by a_0 followed by a rotation around the Z_1 axis

$$
{}^1_2\mathbf{T} = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & a_1 \\ 0 & 0 & 1 & d_2 \\ -\sin\theta_2 & -\cos\theta_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

This is a translation by a_1 and then d_2 followed by a rotation around the X_2 and Z_2 axis

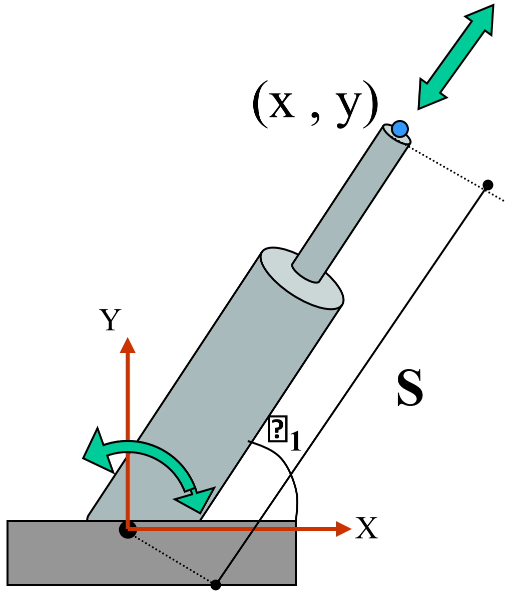
$$
\mathbf{T} = ({}_0\mathbf{T})({}^0_1\mathbf{T})({}^1_2\mathbf{T})
$$

Inverse Kinematics

From Position to Angles

A Simple Example

Revolute and Prismatic Joints Combined



Finding ?:

$$\theta = \arctan(\frac{y}{x})$$

More Specifically:

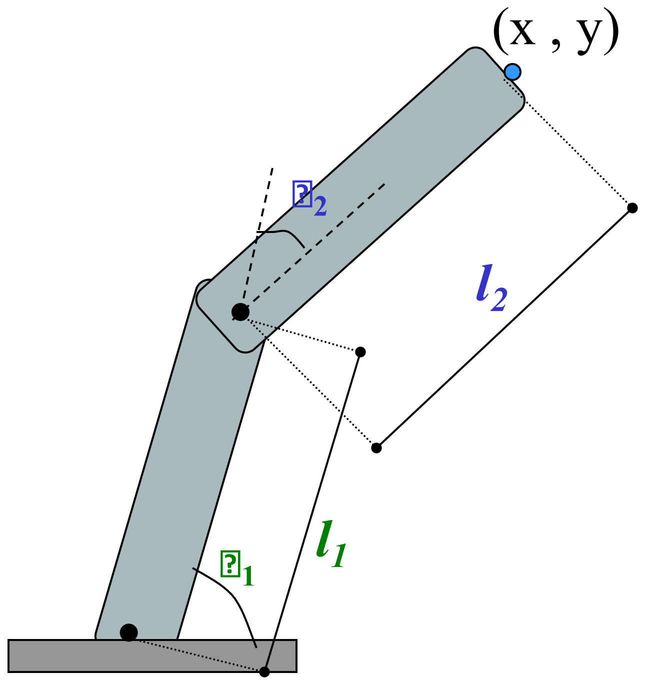
$$\theta = \arctan 2(\frac{y}{x})$$

arctan2() specifies that it's in the first quadrant

Finding **S**:

$$S = \sqrt{(x^2 + y^2)}$$

Inverse Kinematics of a Two Link Manipulator

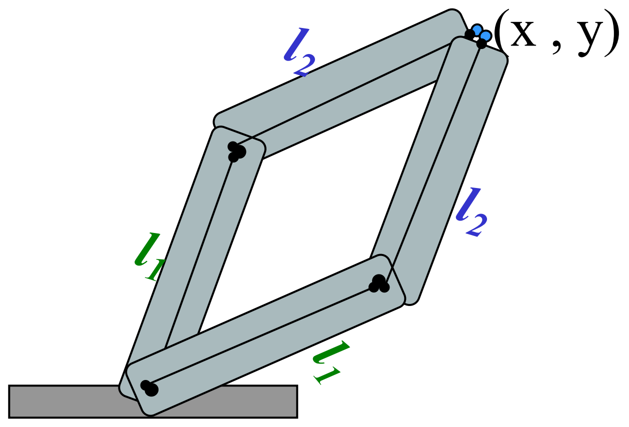


Given: l_1, l_2, x, y

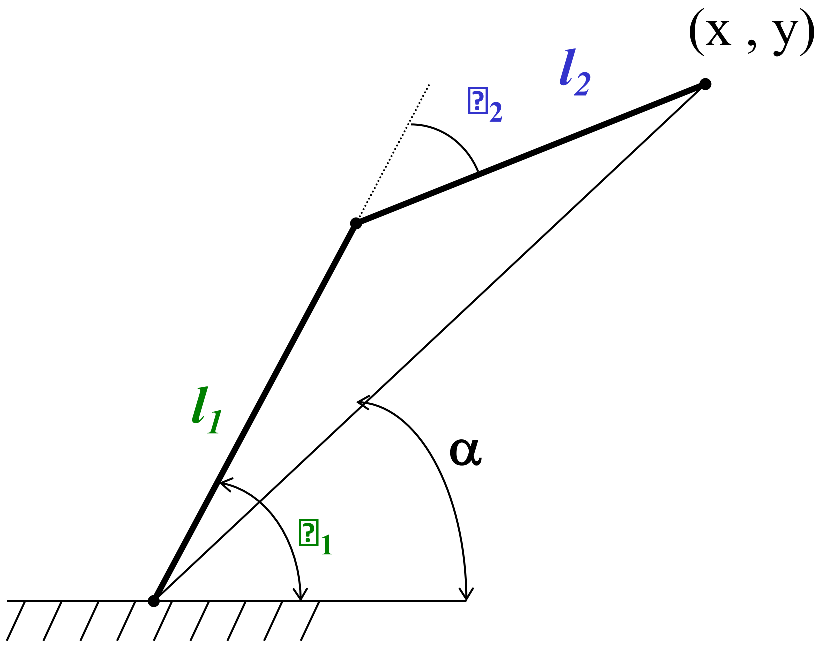
Find: $\boxed{?}_1$, $\boxed{?}_2$

Redundancy:

A unique solution to this problem does not exist. Notice, that using the "givens" two solutions are possible. Sometimes no solution is possible.



The Geometric Solution



Using the Law of Cosines:

$$c^2 = a^2 + b^2 - 2ab\cos C$$

$$(x^2 + y^2) = l_1^{\,2} + l_2^{\,2} - 2 l_1 l_2 \cos(180 - \theta_2)$$

$$\cos(180 - \theta_2) = -\cos(\theta_2)$$

$$\cos(\theta_2) = \frac{x^2 + y^2 - l_1^{\,2} - l_2^{\,2}}{2 l_1 l_2}$$

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

Redundant since θ_2 could be in the first or fourth quadrant.

Using the Law of Cosines:

$$\frac{\sin B}{b} = \frac{\sin C}{c}$$

$$\frac{\sin\bar{\theta}_1}{l_2} = \frac{\sin(180 - \theta_2)}{\sqrt{x^2 + y^2}} = \frac{\sin(\theta_2)}{\sqrt{x^2 + y^2}}$$

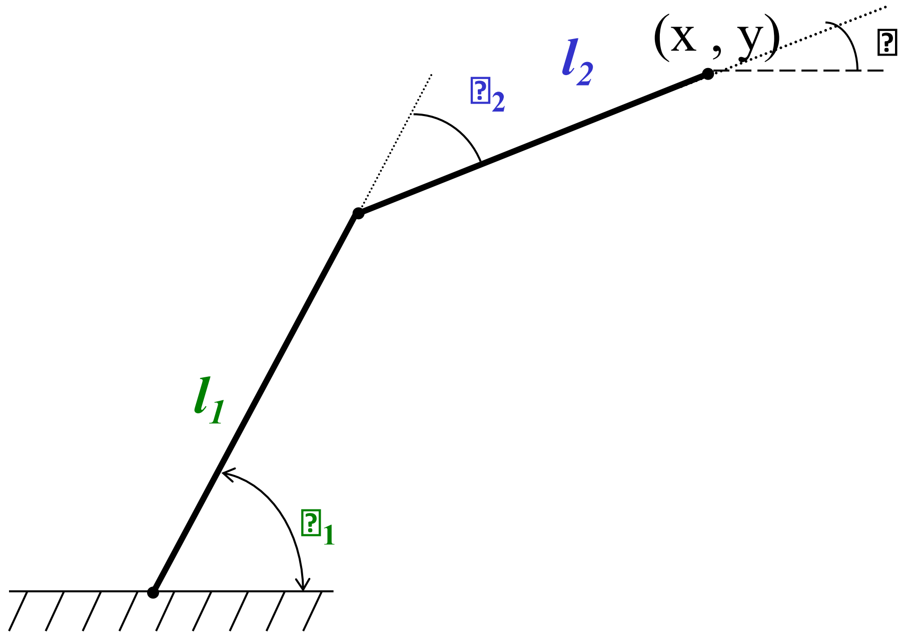
$$\theta_1 = \bar{\theta}_1 + \alpha$$

$$\alpha = \arctan 2\left(\frac{y}{x}\right)$$

Redundancy caused since θ_2 has two possible values

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

The Algebraic Solution



$$c_1 = \cos\theta_1$$

$$c_{1+2} = \cos(\theta_2 + \theta_1)$$

$$(1)\, x = l_1 c_1 + l_2 c_{1+2}$$

$$(2)\, y = l_1 s_1 + l_2 \sin_{1+2}$$

$$(3)\, \theta = \theta_1 + \theta_2$$

$$(1)^2 + (2)^2 = x^2 + y^2 =$$

$$= \left(l_1^2 c_1^2 + l_2^2 (c_{1+2})^2 + 2 l_1 l_2 c_1 (c_{1+2})\right) + \left(l_1^2 s_1^2 + l_2^2 (\sin_{1+2})^2 + 2 l_1 l_2 s_1 (\sin_{1+2})\right)$$

$$= l_1^2 + l_2^2 + 2 l_1 l_2 \left(c_1 (c_{1+2}) + s_1 (\sin_{1+2})\right)$$

$$= l_1^2 + l_2^2 + 2 l_1 l_2 c_2 \leftarrow \text{Only Unknown}$$

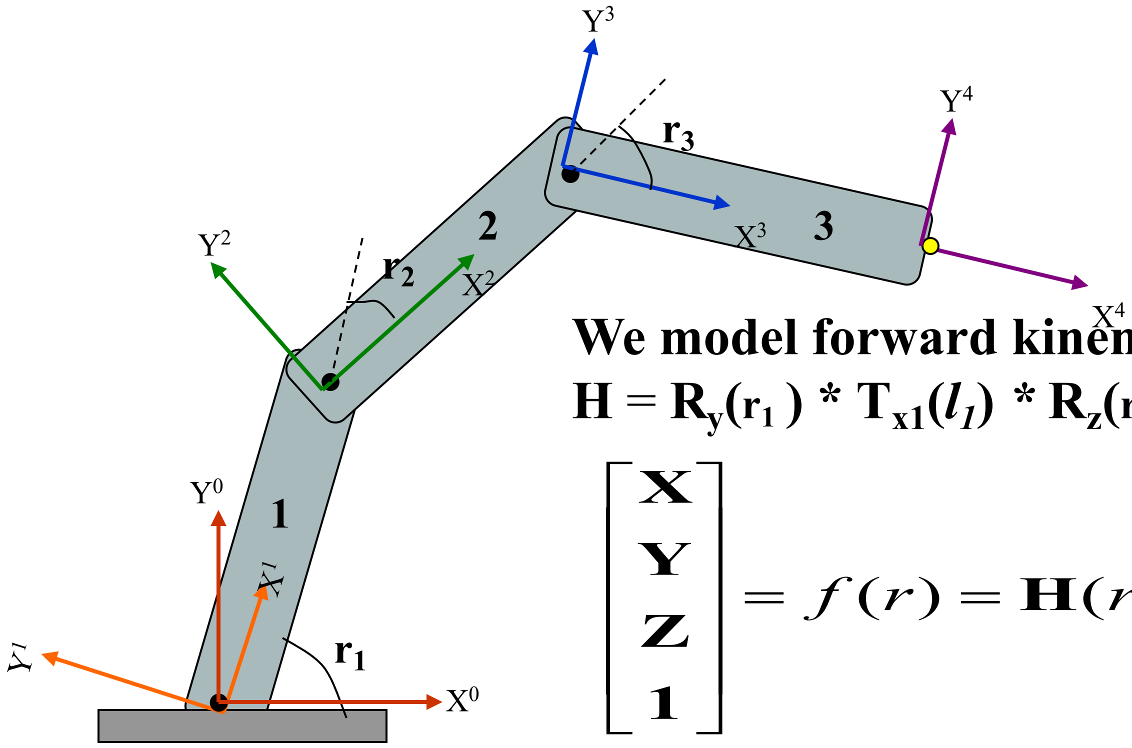
$$\therefore \theta_2 = \arccos\left(\frac{x^2 + y^2 - l_1^2 - l_2^2}{2 l_1 l_2}\right)$$

Note:

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

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

The Numeric solution



We model forward kinematics as

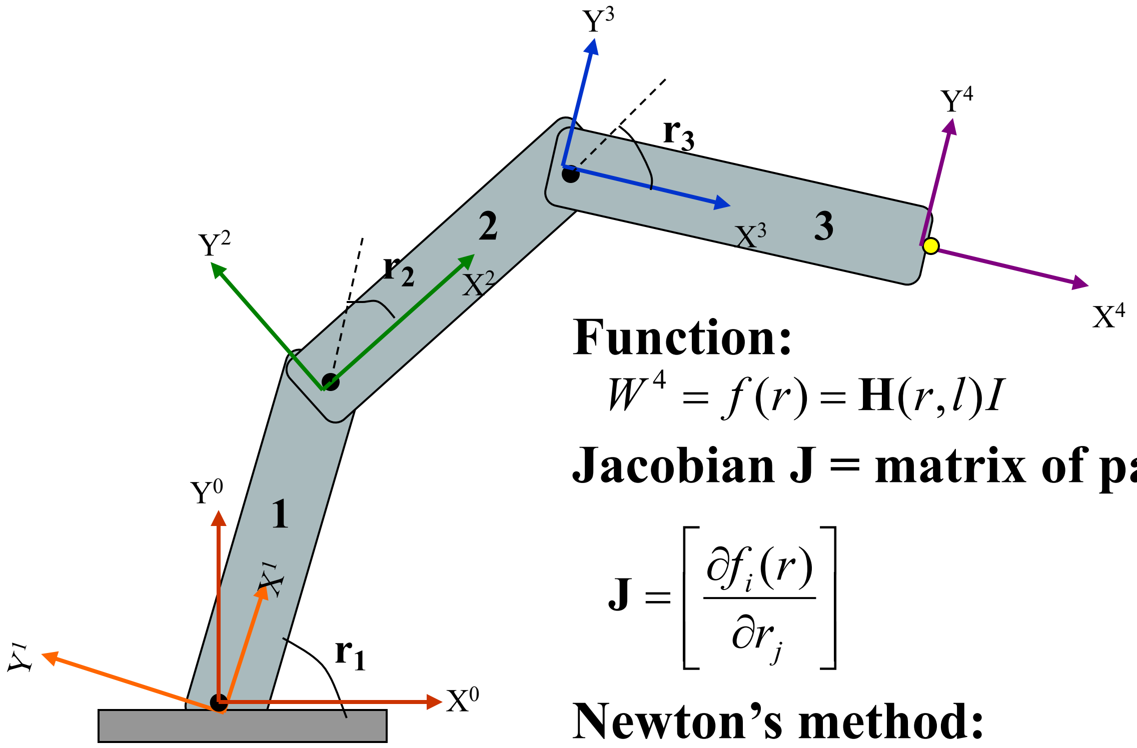
$\mathbf{H} = \mathbf{R_y(r_1)} * \mathbf{T_{x1}}(l_1) * \mathbf{R_z(r_2)} * \mathbf{T_{x2}}(l_2) * \mathbf{R_z(r_3)} * \mathbf{T_{x3}}(l_3)$

$$\begin{bmatrix} \mathbf{X} \\ \mathbf{Y} \\ \mathbf{Z} \\ \mathbf{1} \end{bmatrix} = f(r) = \mathbf{H}(r,l) \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{1} \end{bmatrix}$$

Now given desired Cartesian position [X,Y,Z] solve numerically for the corresponding joint angles [$\mathbf{r_1\, r_2\, r_3}$] :

$$\mathbf{0} = \begin{bmatrix} \mathbf{X} \\ \mathbf{Y} \\ \mathbf{Z} \\ \mathbf{1} \end{bmatrix} - f(r)$$

The Numeric solution: How to solve? Newton's Metod



Function:

$$W^4 = f(r) = \mathbf{H}(r,l)I$$

Jacobian J = matrix of partial derivatives:

$$\mathbf{J} = \left[\frac{\partial f_i(r)}{\partial r_j} \right]$$

Newton's method:

Guess initial joint angles r

Iterate

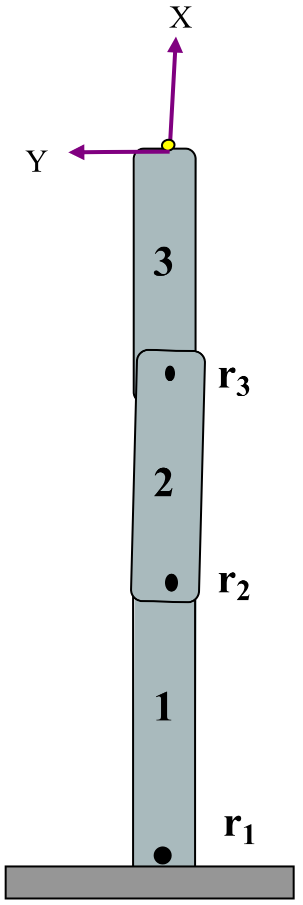
J*dr = W-f(r)

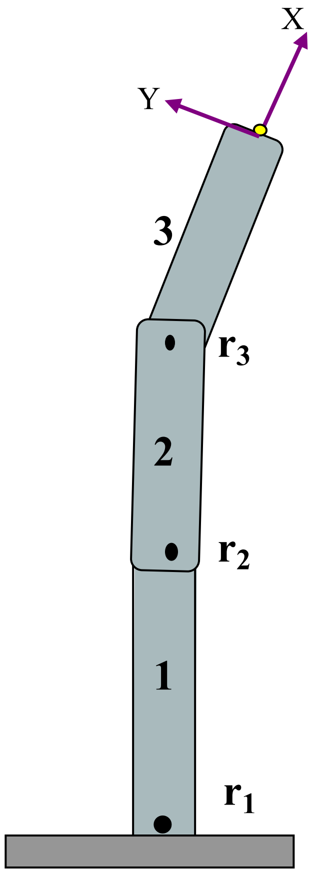
r = r+dr

If guess is close enough r converges to solution.

Otherwise may diverge.

Newton's Metod: Convergence issues

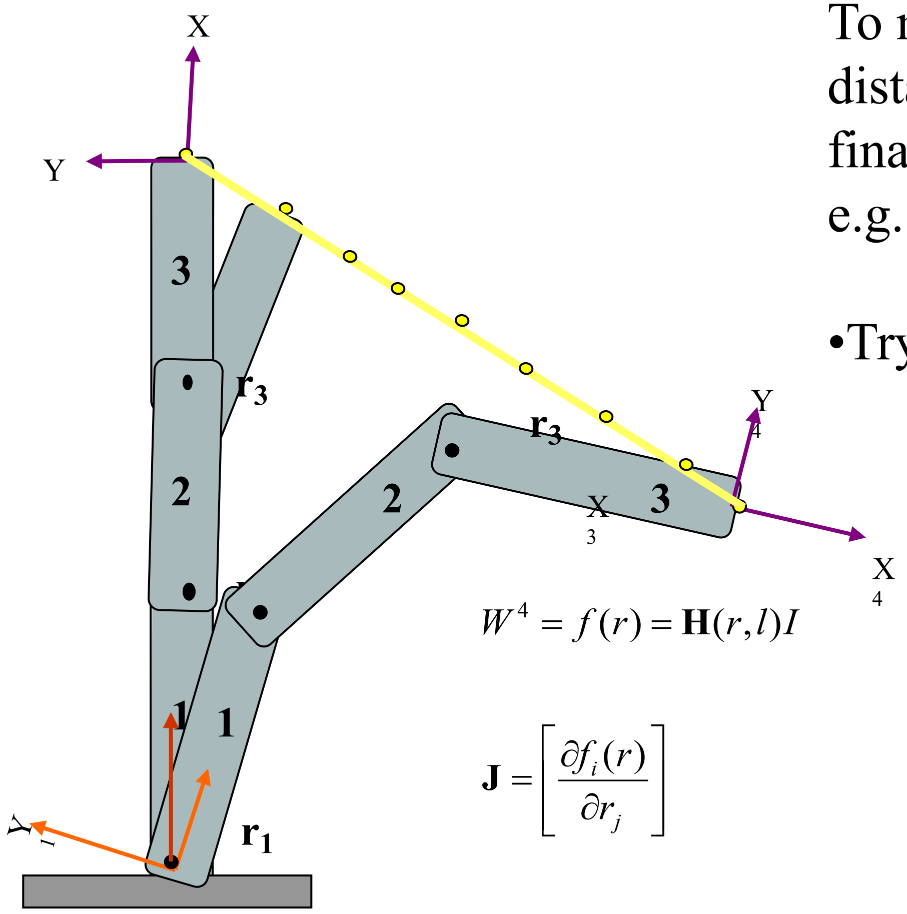




Use a start position with known W and r
(e.g. Wi = [0, $l_1+l_2+l_3$, 0, 1]^T for r = 0)

Let next Wk close to this initial.
Use r0 as initial guess for r1
Iterate
  J*dr = Wk-f(r)
  r = r+dr
r guess is close so r converges to solution.

Newton's Metod: Convergence issues



$$W^4 = f(r) = \mathbf{H}(r,l)I$$

$$\mathbf{J} = \left[\frac{\partial f_i(r)}{\partial r_j} \right]$$

To make a large movement, divide the total distance from (known) initial Wi to the new final Wf into small steps ○ Wk
e.g. on a line

•Try this in lab!

Resolved rate control

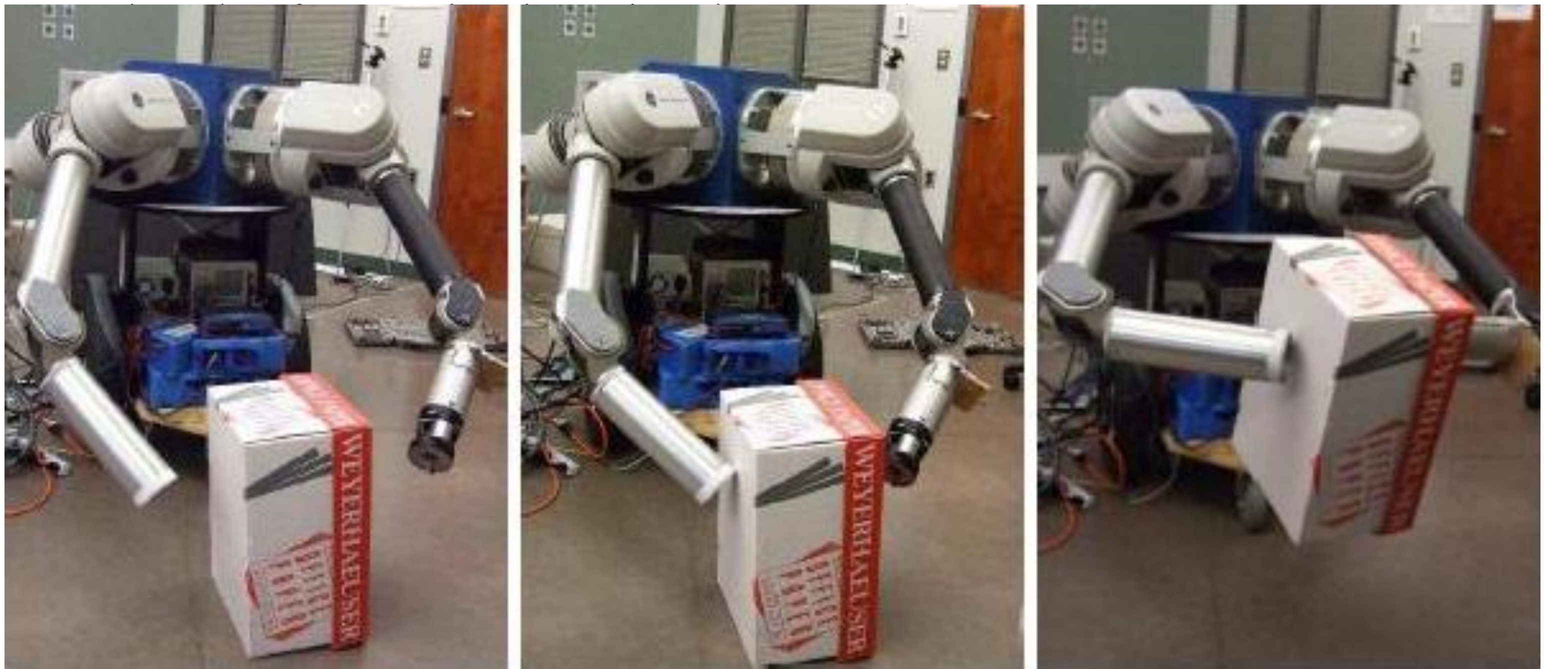
- Here instead of computing an inverse kinematics solution then move the robot to that point, we actually move the robot dr for every iteration in newtons method.
- Let dr → 0, then we can view this as velocity control:

$$\dot{r} = \mathbf{J}(r(t))^{-1}\dot{w}$$

$\dot{w} = v =$ Cartesian translation velovity

Conclusion

- Forward kinematics can be tedious for multilink arms
- Inverse kinematics can be solved algebraically or numerically. The latter is more common for complex arms or vision-guided control (later)
- Limitations: We avoided details of the various angular representations (Euler, quarternion or exponentials) and their detailed use in Kinematics. (this typically takes

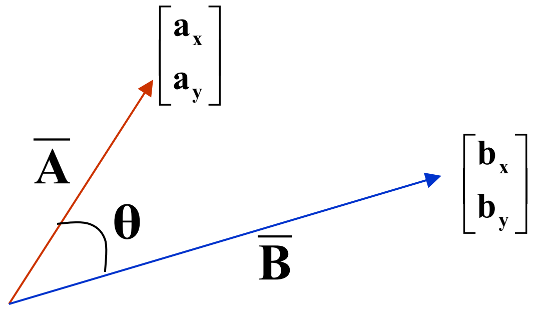


Quick Math Review

Dot Product:

Geometric Representation:

$$\overline{\mathbf{A}} \bullet \overline{\mathbf{B}} = \|\mathbf{A}\|\|\mathbf{B}\|\cos\theta$$



$\begin{bmatrix} a_x \\ a_y \end{bmatrix}$
$\overline{A}$
θ
$\begin{bmatrix} b_x \\ b_y \end{bmatrix}$
$\overline{B}$

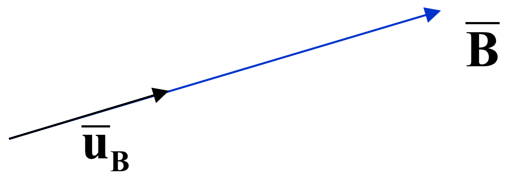
Matrix Representation:

$$\overline{\mathbf{A}} \bullet \overline{\mathbf{B}} = \begin{bmatrix} \mathbf{a}_x \\ \mathbf{a}_y \end{bmatrix} \bullet \begin{bmatrix} \mathbf{b}_x \\ \mathbf{b}_y \end{bmatrix} = \mathbf{a}_x\mathbf{b}_x + \mathbf{a}_y\mathbf{b}_y$$

Unit Vector

Vector in the direction of a chosen vector but whose magnitude is 1.

$$\overline{\mathbf{u}}_B = \frac{\overline{\mathbf{B}}}{\|\mathbf{B}\|}$$



Quick Matrix Review

Matrix Multiplication:

An (m x n) matrix A and an (n x p) matrix B, can be multiplied since the number of columns of A is equal to the number of rows of B.

Non-Commutative Multiplication

AB is NOT equal to BA

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} * \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} (ae+bg) & (af+bh) \\ (ce+dg) & (cf+dh) \end{bmatrix}$$

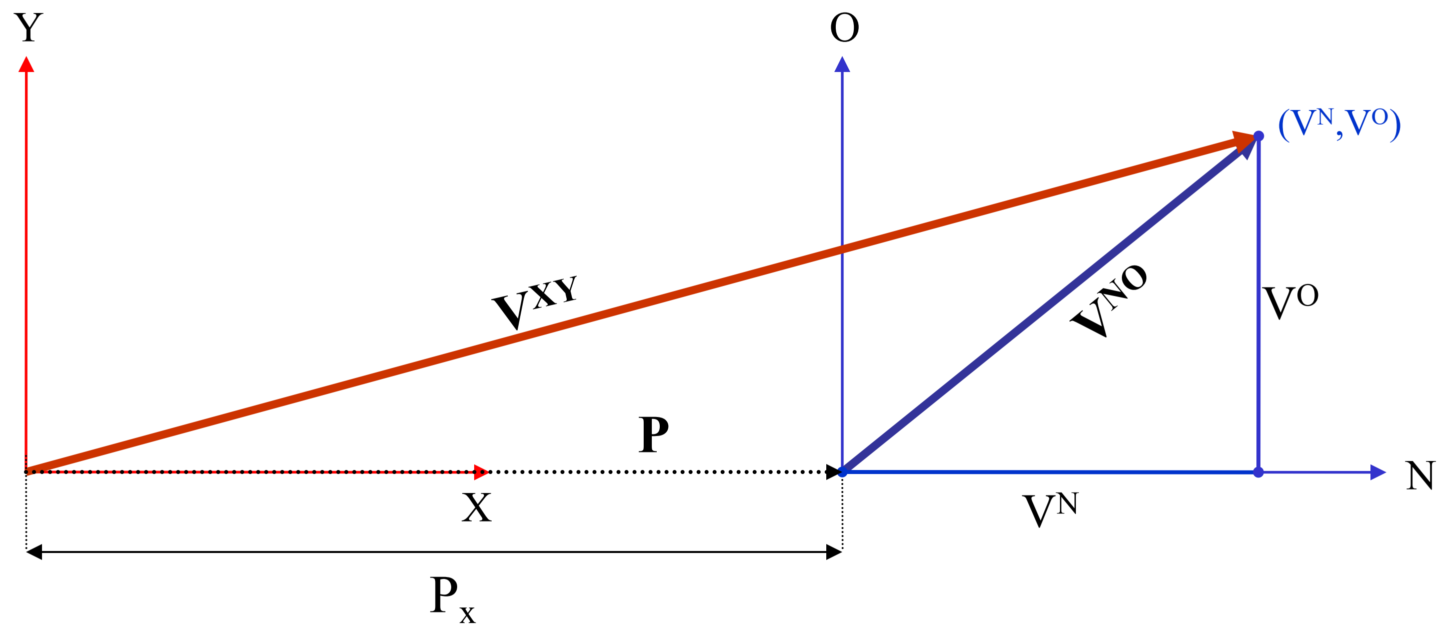
Matrix Addition:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} + \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} (a+e) & (b+f) \\ (c+g) & (d+h) \end{bmatrix}$$

Basic Transformations

Moving Between Coordinate Frames

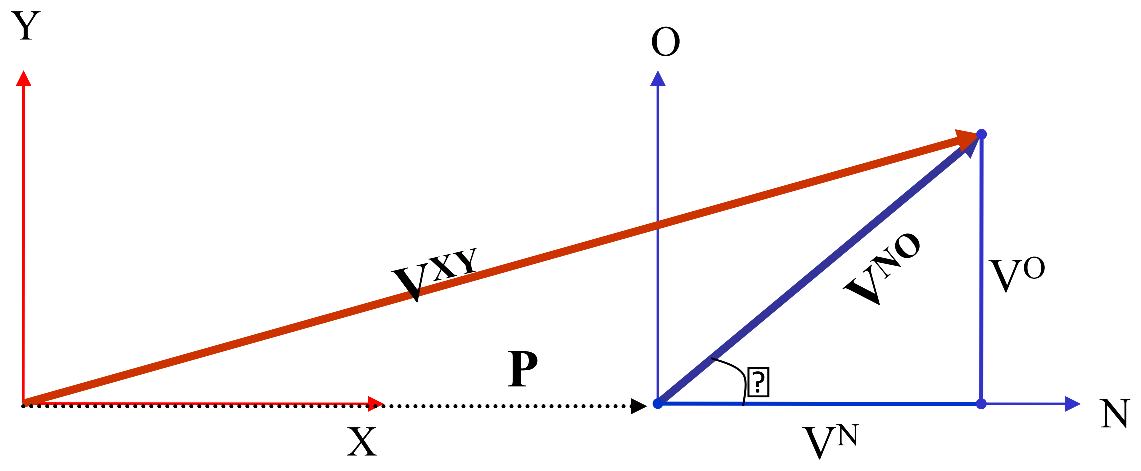
Translation Along the X-Axis



P_x = distance between the XY and NO coordinate planes

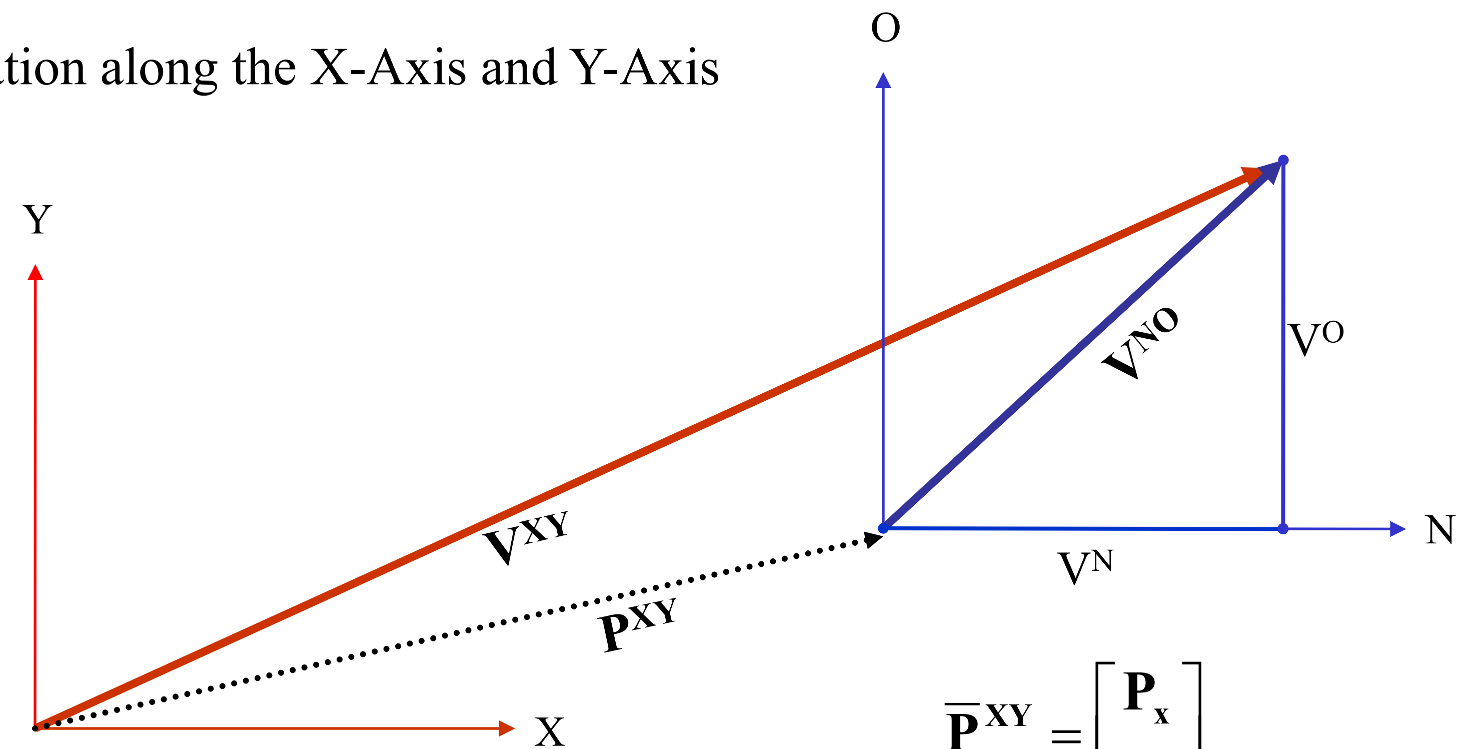
Notation:
$$\overline{\mathbf{V}}^{XY} = \begin{bmatrix} \mathbf{V}^X \\ \mathbf{V}^Y \end{bmatrix} \qquad \overline{\mathbf{V}}^{NO} = \begin{bmatrix} \mathbf{V}^N \\ \mathbf{V}^O \end{bmatrix} \qquad \overline{\mathbf{P}} = \begin{bmatrix} \mathbf{P}_x \\ \mathbf{0} \end{bmatrix}$$

Writing $\overline{\mathbf{V}}^{XY}$ in terms of $\overline{\mathbf{V}}^{NO}$



$$\overline{\mathbf{V}}^{XY} = \begin{bmatrix} \mathbf{P}_X + \mathbf{V}^N \\ \mathbf{V}^O \end{bmatrix} = \overline{\mathbf{P}} + \overline{\mathbf{V}}^{NO}$$

Translation along the X-Axis and Y-Axis



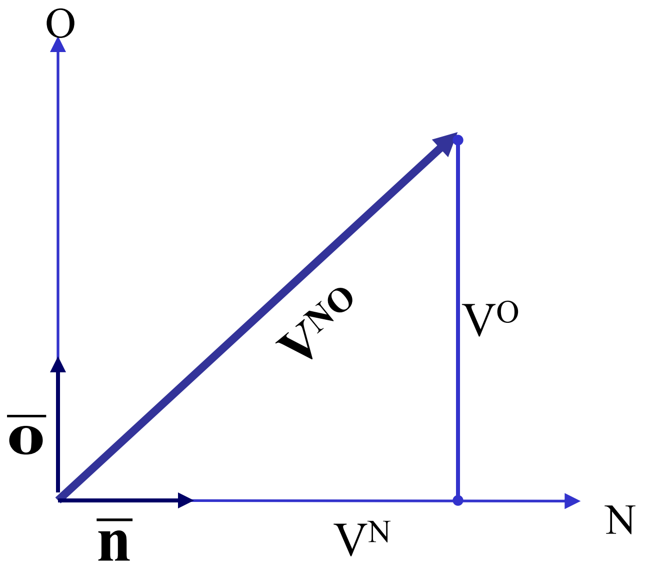
$$\overline{\mathbf{P}}^{XY} = \begin{bmatrix} \mathbf{P}_x \\ \mathbf{P}_Y \end{bmatrix}$$

$$\overline{\mathbf{V}}^{XY} = \overline{\mathbf{P}} + \overline{\mathbf{V}}^{NO} = \begin{bmatrix} \mathbf{P}_X + \mathbf{V}^N \\ \mathbf{P}_Y + \mathbf{V}^O \end{bmatrix}$$

Using Basis Vectors

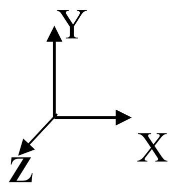
Basis vectors are unit vectors that point along a coordinate axis

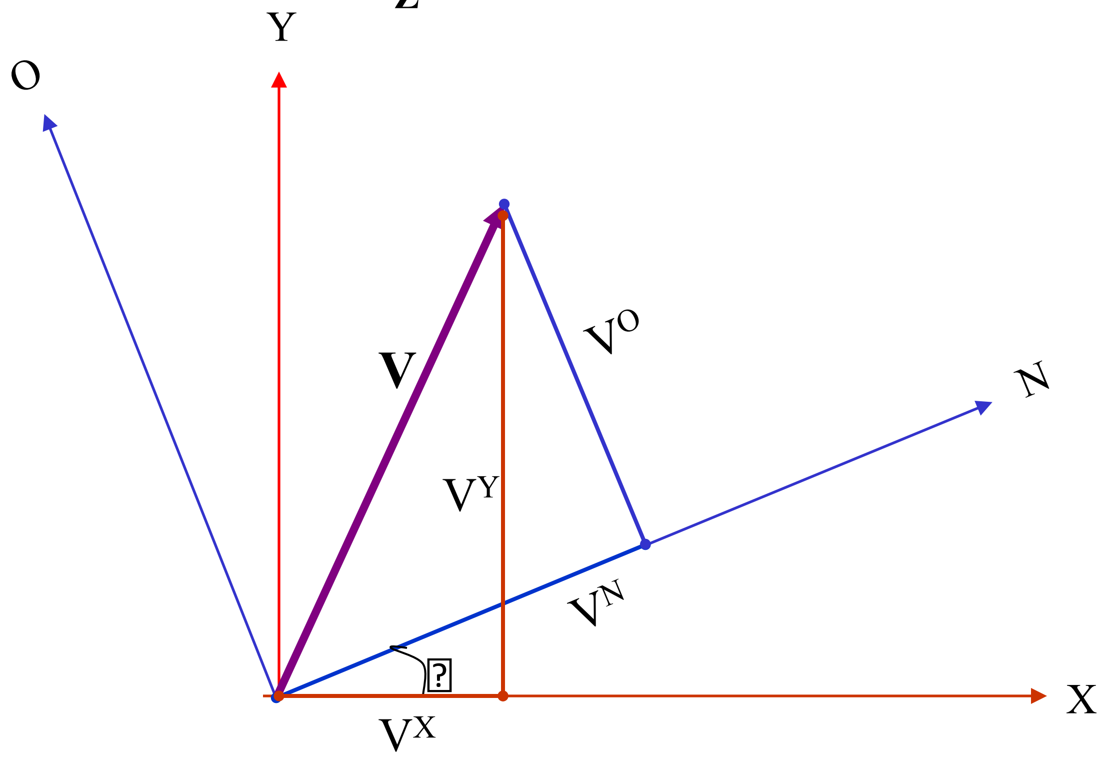
$\overline{\mathbf{n}}$	Unit vector along the N-Axis
$\overline{\mathbf{o}}$	Unit vector along the N-Axis
$\left\Vert \mathbf{V}^{\mathbf{NO}} \right\Vert$	Magnitude of the V^{NO} vector



$$\overline{\mathbf{V}}^{\mathbf{NO}} = \begin{bmatrix} \mathbf{V}^{\mathbf{N}} \\ \mathbf{V}^{\mathbf{O}} \end{bmatrix} = \begin{bmatrix} \left\Vert \mathbf{V}^{\mathbf{NO}} \right\Vert \mathbf{cos\theta} \\ \left\Vert \mathbf{V}^{\mathbf{NO}} \right\Vert \mathbf{sin\theta} \end{bmatrix} = \begin{bmatrix} \left\Vert \mathbf{V}^{\mathbf{NO}} \right\Vert \mathbf{cos\theta} \\ \left\Vert \mathbf{V}^{\mathbf{NO}} \right\Vert \mathbf{cos(90-\theta)} \end{bmatrix} = \begin{bmatrix} \overline{\mathbf{V}}^{\mathbf{NO}} \bullet \overline{\mathbf{n}} \\ \overline{\mathbf{V}}^{\mathbf{NO}} \bullet \overline{\mathbf{o}} \end{bmatrix}$$

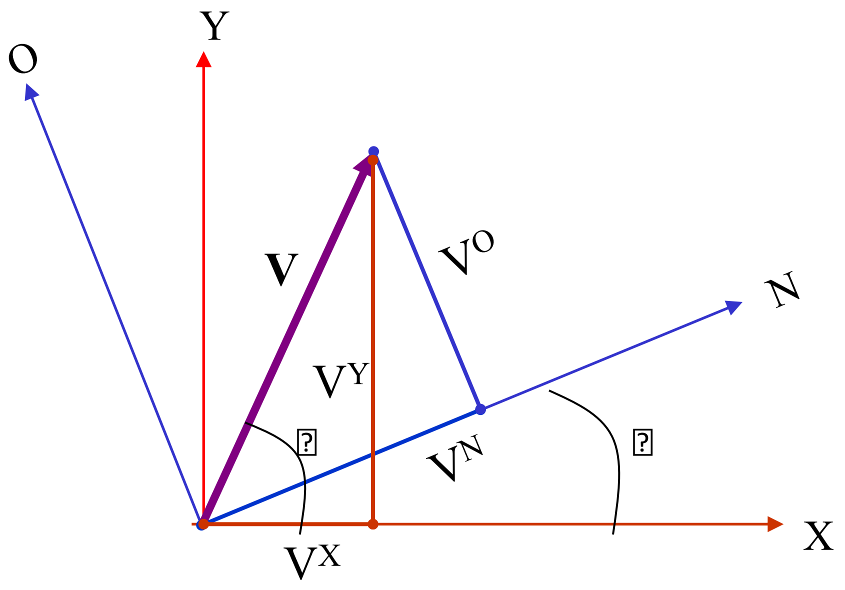
Rotation (around the Z-Axis)





⍰ = Angle of rotation between the XY and NO coordinate axis

$$\overline{\mathbf{V}}^{\mathbf{XY}} = \begin{bmatrix} \mathbf{V}^{\mathbf{X}} \\ \mathbf{V}^{\mathbf{Y}} \end{bmatrix} \qquad \overline{\mathbf{V}}^{\mathbf{NO}} = \begin{bmatrix} \mathbf{V}^{\mathbf{N}} \\ \mathbf{V}^{\mathbf{O}} \end{bmatrix}$$



$\underline{\mathbf{x}}$ Unit vector along X-Axis

$\underline{\wedge}$ Can be considered with respect to the XY coordinates or NO coordinates

$$\left\|\overline{\mathbf{V}}^{\mathbf{XY}}\right\| = \left\|\overline{\mathbf{V}}^{\mathbf{NO}}\right\|$$

$$\mathbf{V}^{\mathbf{X}} = \left\|\overline{\mathbf{V}}^{\mathbf{XY}}\right\|\mathbf{cos}\boldsymbol{\alpha} = \left\|\overline{\mathbf{V}}^{\mathbf{NO}}\right\|\mathbf{cos}\boldsymbol{\alpha} = \overline{\mathbf{V}}^{\mathbf{NO}} \bullet \overline{\mathbf{x}}$$

$$\mathbf{V}^{\mathbf{X}} = (\mathbf{V}^{\mathbf{N}} * \overline{\mathbf{n}} + \mathbf{V}^{\mathbf{O}} * \overline{\mathbf{o}}) \bullet \overline{\mathbf{x}}$$

(Substituting for V^{NO} using the N and O components of the vector)

$$\begin{aligned}\mathbf{V}^{\mathbf{X}} &= \mathbf{V}^{\mathbf{N}}(\overline{\mathbf{x}} \bullet \overline{\mathbf{n}}) + \mathbf{V}^{\mathbf{O}}(\overline{\mathbf{x}} \bullet \overline{\mathbf{o}}) \\ &= \mathbf{V}^{\mathbf{N}}(\mathbf{cos}\boldsymbol{\theta}) + \mathbf{V}^{\mathbf{O}}(\mathbf{cos}(\boldsymbol{\theta} + \mathbf{90})) \\ &= \mathbf{V}^{\mathbf{N}}(\mathbf{cos}\boldsymbol{\theta}) - \mathbf{V}^{\mathbf{O}}(\mathbf{sin}\boldsymbol{\theta})\end{aligned}$$

Similarly….

$$\mathbf{V}^{Y} = \left\|\overline{\mathbf{V}}^{NO}\right\| \sin\alpha = \left\|\overline{\mathbf{V}}^{NO}\right\| \cos(90-\alpha) = \overline{\mathbf{V}}^{NO} \bullet \overline{\mathbf{y}}$$

$$\mathbf{V}^{Y} = (\mathbf{V}^{N} * \overline{\mathbf{n}} + \mathbf{V}^{O} * \overline{\mathbf{o}}) \bullet \overline{\mathbf{y}}$$

$$\begin{aligned}\mathbf{V}^{Y} &= \mathbf{V}^{N}(\overline{\mathbf{y}} \bullet \overline{\mathbf{n}}) + \mathbf{V}^{O}(\overline{\mathbf{y}} \bullet \overline{\mathbf{o}}) \\ &= \mathbf{V}^{N}(\cos(90-\theta)) + \mathbf{V}^{O}(\cos\theta) \\ &= \mathbf{V}^{N}(\sin\theta) + \mathbf{V}^{O}(\cos\theta)\end{aligned}$$

So….

$$\mathbf{V}^{X} = \mathbf{V}^{N}(\cos\theta) - \mathbf{V}^{O}(\sin\theta)$$
$$\mathbf{V}^{Y} = \mathbf{V}^{N}(\sin\theta) + \mathbf{V}^{O}(\cos\theta)$$

$$\overline{\mathbf{V}}^{XY} = \begin{bmatrix} \mathbf{V}^{X} \\ \mathbf{V}^{Y} \end{bmatrix}$$

Written in Matrix Form

$$\overline{\mathbf{V}}^{XY} = \begin{bmatrix} \mathbf{V}^{X} \\ \mathbf{V}^{Y} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \mathbf{V}^{N} \\ \mathbf{V}^{O} \end{bmatrix}$$

Rotation Matrix about the z-axis

$$\begin{aligned} x &= l_1\, c_1 + l_2\, c_{1+2} \\ &= l_1\, c_1 + l_2\, c_1 c_2 - l_2 s_1 s_2 \\ &= c_1 (l_1 + l_2\, c_2) - s_1 (l_2 s_2) \end{aligned}$$

$$\boxed{\begin{aligned} &\textit{Note}: \\ &\cos(a \pm b) = (\cos a)(\cos b) \mp (\sin a)(\sin b) \\ &\sin(a \pm b) = (\cos a)(\sin b) \pm (\cos b)(\sin a) \end{aligned}}$$

$$\begin{aligned} y &= l_1\, s_1 + l_2\, \sin_{1+2} \\ &= l_1\, s_1 + l_2\, s_1 c_2 + l_2\, s_2 c_1 \\ &= c_1 (l_2\, s_2) + s_1 (l_1 + l_2\, c_2) \end{aligned}$$

We know what θ_2 is from the previous slide. We need to solve for θ_1 . Now we have two equations and two unknowns (sin θ_1 and cos θ_1)

$$c_1 = \frac{x + s_1 (l_2 s_2)}{(l_1 + l_2\, c_2)}$$

$$y = \frac{x + s_1 (l_2 s_2)}{(l_1 + l_2\, c_2)} (l_2\, s_2) + s_1 (l_1 + l_2\, c_2)$$

Substituting for c_1 and simplifying many times

$$= \frac{1}{(l_1 + l_2\, c_2)} \left(x\, l_2 s_2 + s_1 (l_1{}^2 + l_2{}^2 + 2 l_1 l_2\, c_2) \right)$$

Notice this is the law of cosines and can be replaced by $x^2 + y^2$

$$s_1 = \frac{y(l_1 + l_2\, c_2) - x\, l_2 s_2}{x^2 + y^2}$$

$$\theta_1 = \arcsin\left(\frac{y(l_1 + l_2\, c_2) - x\, l_2 s_2}{x^2 + y^2} \right)$$