

# Remainder topics

## Intro to Robotics

### 1 Inverse Manipulator Kinematics

Kinematics is the science of motion that treats the subject without regard to the forces that cause it. Within the science of kinematics, one studies the position, the velocity, the acceleration, and all higher order derivatives of the position variables (with respect to time or any other variable(s)). Hence, the study of the kinematics of manipulators refers to all the geometrical and time-based properties of the motion.

Given the desired position and orientation of the tool relative to the station, how do we compute the set of joint angles which will achieve this desired result?

Solving the problem of finding the required joint angles to place the tool frame, {T}, relative to the station frame, {S}, is split into two parts. First, frame transformations are performed to find the wrist frame, {W}, relative to the base frame, {B}, and then the inverse kinematics are used to solve for the joint angles.

### 2 Jacobian Matrix and Singularities

Definition of Jacobian - multidimensional matrix of derivatives ( $J \rightarrow$  velocity differentiate from positions), Jacobians can relate the joint velocities to cartesian velocities in a robot manipulator.

$$\text{Cartesian: } v = \begin{bmatrix} \dot{P}_x \\ \dot{P}_y \\ \dot{P}_z \end{bmatrix} = \text{Joint: } \begin{bmatrix} {}^0v \\ {}^0w \end{bmatrix} = J * \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}$$

${}^0v$  is linear

${}^0w$  is angular

You always need the jacobian in the base frame  ${}^0J$ , when you don't have it; you can solve for it. Use formula:

$${}^0J = {}^0R_x * {}^xJ$$

Singularities- is a point where the robot loses one or more degrees of freedom. Here, it is impossible to move the end effector (tool) in a particular direction,

regardless of the joint rates. two types:

- Workspace-bound singularities- fully stretched out , or tool folded back on itself. Tool is at boundary of workspace.
- Workspace interior singularities - 2 or more joint axes line up with each other.
- At singularities robots lose their degrees of freedom. if the  $\det[J(\theta)] = 0$  singularity exist, if  $\det[J(\theta)] \neq 0$  no singularities. Remember  $\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \dots \\ \theta_n \end{bmatrix}$

### 3 Jacobian Matrix - Partial Differentiation Method

Jacobian involves finding velocities. Steps:

- DH table  $\rightarrow$  Transformation matrix
- Get  ${}^0T = {}^0T_1 {}^1T_2 \dots {}^nT_e$  and extract translation column to get  $P = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix}$  (position of end effector relative to base frame)
- When we differentiate P, we get:  $\begin{bmatrix} \dot{P}_x \\ \dot{P}_y \\ \dot{P}_z \end{bmatrix} = {}^0J \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix}$

**General formula:**

Given position of joints:

- $y_1 = f_1(x_1, x_2, \dots)$
- $y_2 = f_2(x_1, x_2, \dots)$
- ...
- $y_n = f_n(x_1, x_2, \dots)$

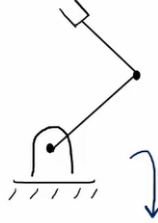
We can then differentiate to get:

- $\partial y_1 = f_1(\frac{\partial f_1}{\partial x_1} \partial x_1, \frac{\partial f_1}{\partial x_2} \partial x_2, \dots)$
- $\partial y_2 = f_2(\frac{\partial f_2}{\partial x_1} \partial x_1, \frac{\partial f_2}{\partial x_2} \partial x_2, \dots)$
- ...
- $\partial y_n = f_n(\frac{\partial f_n}{\partial x_1} \partial x_1, \frac{\partial f_n}{\partial x_2} \partial x_2, \dots)$

In Matrix form:  $\dot{Y} = J\dot{X} \rightarrow \begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \\ \dots \end{bmatrix} = \left[ \frac{\partial F}{\partial X} = J \right] \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dots \end{bmatrix}$

### 3.1 Example

#### Example



**Q:** Velocity of the tip of the arm as a function of joint rates

- (i) Give the answer in terms of frame 0
- (ii) Give the answer in terms of frame 2

Axis (i)	$\alpha_{i-1}$	$a_{i-1}$	$d_i$	$\theta_i$
1	0	L1	0	$\theta_1$
2	0	L2	0	$\theta_2$

**Part 1** using kinematics formula to get:

$${}^0_1T = \begin{bmatrix} c\theta_1 & -s\theta_1 & 0 & l_1 \\ s\theta_1 & c\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^1_2T = \begin{bmatrix} c\theta_2 & -s\theta_2 & 0 & l_2 \\ s\theta_2 & c\theta_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Now we can solve for:

$${}^0_2T = \begin{bmatrix} c_{12} & -s_{12} & 0 & l_1 + l_2 c\theta_1 \\ s_{12} & c_{12} & 0 & l_2 s\theta_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Giving us :  $P = \begin{bmatrix} l_1 + l_2 c\theta_1 \\ l_2 s\theta_1 \\ 0 \end{bmatrix}$  thus, :  $\dot{P} = \begin{bmatrix} -l_2 s\theta_1 \dot{\theta}_1 \\ l_2 c\theta_1 \dot{\theta}_1 \\ 0 \end{bmatrix}$  We can then write:

$$\dot{P} = \begin{bmatrix} -l_2 s\theta_1 \\ l_2 c\theta_1 \\ 0 \end{bmatrix} [\dot{\theta}_1] \text{ where this is equivalent to } \dot{Y} = J\dot{X} \text{ and } {}^0V = {}^0J[\dot{\theta}_1]$$

**ANSWERS QUESTION VELOCITY OF THE TOOL(FRAME 2) RELATIVE TO FRAME 0**

**Part 2**

Solve  ${}^2V = {}^2J[\dot{\theta}_1]$ . Using formula:

$${}^2J = {}^2_0R {}^0J$$

From previous question, where we solved for  ${}^0_2T$  we can extract  ${}^0_2R$  and take

transpose to get:

$${}^2_0R = \begin{bmatrix} c_{12} & s_{12} & 0 \\ -s_{12} & c_{12} & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ so, we get:}$$

$${}^2J = \begin{bmatrix} c_{12} & s_{12} & 0 \\ -s_{12} & c_{12} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -l_2 s \theta_1 \\ l_2 c \theta_1 \\ 0 \end{bmatrix} = \begin{bmatrix} -l_2 c_{12} s_1 + l_2 s_{12} c_1 \\ l_2 s_{12} s_1 + l_2 c_{12} c_1 \\ 0 \end{bmatrix}$$

and finally solving for  ${}^2v$ , we get:

$${}^2v = \begin{bmatrix} -l_2 c_{12} s_1 + l_2 s_{12} c_1 \\ l_2 s_{12} s_1 + l_2 c_{12} c_1 \\ 0 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \end{bmatrix}$$

## 4 Jacobian Matrix - Velocity Propagation Method

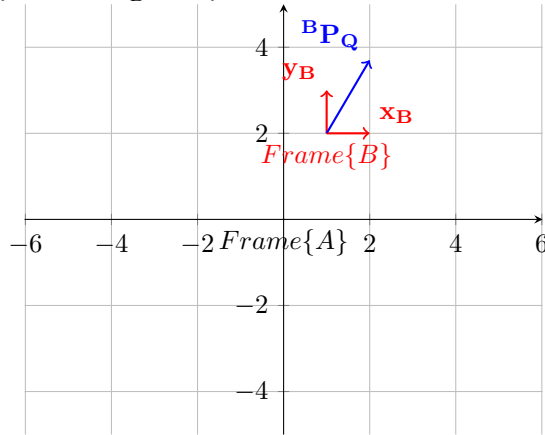
**Intro Steps:**

- Get linear velocity ( $v$ )
  - Get angular velocity ( $\omega$ )
  - Start from first frame and solve eqns till you get to end effector frame  
 $0 \rightarrow 1 \rightarrow 2 \rightarrow \dots \rightarrow E$
  - ${}^E V_E = {}^E J \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}$
- $${}^0 J = {}^0_E R * {}^E J$$

### Linear Velocity

Consider two frames  $\{A\}, \{B\}$ ; where  $\{A\}$  is fixed and  $\{B\}$  is moving away from  $\{A\}$ . Orientation of  $\{B\}$  does not change. Let Frame  $\{B\}$  have a point  $Q({}^B P_Q)$ . We can compute the linear velocity of  ${}^B P_Q$  with respect to frame  $\{A\}$  using the formula:

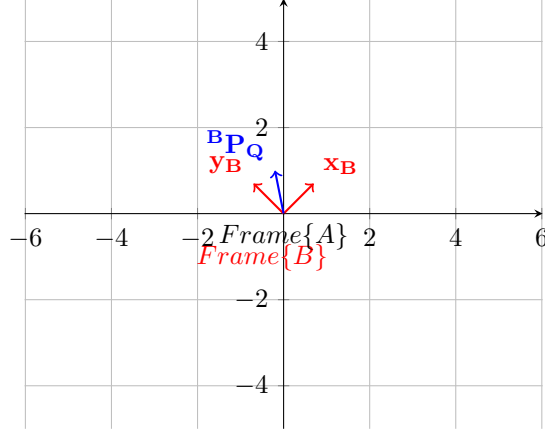
$${}^A V_Q = {}^A V_B + {}^A_B R {}^B V_Q$$



### Angular Velocity

Consider two frames  $\{A\}, \{B\}$ ; where  $\{A\}$  is fixed and  $\{B\}$  is moving away from  $\{A\}$ . No linear motion for  $\{B\}$ . Let Frame  $\{B\}$  have a point  $Q$  ( ${}^B P_Q$ ). We can compute the angular velocity of  ${}^B P_Q$  with respect to frame  $\{A\}$  using the formula:

$${}^A V_Q = {}^A R^B V_Q + {}^A \Omega \times {}^A R^B P_Q$$



### 4.1 Velocity Propagation equations(iterative)

1.  ${}^{i+1}\omega_{i+1} = {}^i R({}^i \omega_i) + \dot{\theta}_{i+1} {}^{i+1}\hat{z}_{i+1}$
2.  ${}^{i+1}v_{i+1} = {}^i R({}^i v_i + {}^i \omega_i \times {}^i P_{i+1}) + \dot{d}_{i+1} {}^{i+1}\hat{z}_{i+1}$

- $i$  is joint/frame
- $i+1$  next frame
- $\omega$  angular velocity
- $v$  linear velocity

- $\dot{\theta}_{i+1} \hat{z}$  is for revolute joints:  $\begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_{i+1} \end{bmatrix}$

- $\dot{d}_{i+1} \hat{z}$  is for prismatic joints:  $\begin{bmatrix} 0 \\ 0 \\ \dot{d}_{i+1} \end{bmatrix}$

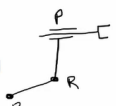
- The main idea is starting from frame  $\{0\}$  iteratively solve the solution for  $n$  joints.

## 4.2 Example

### Example

(a) Find the linear and angular velocity at the last frame  
 (b) Find the Jacobian at the last frame  
 (c) Find the Jacobian at the base frame  
 (d) Find an equation for the singularities of the robot

1) DH Table  
 2) Transformation matrices  
 3) Velocity Propagation  $\rightarrow$  base  $\rightarrow$  final  
 4)  ${}^3J$   
 5)  ${}^0J$   
 6) Singularities



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

$\leftarrow \cos(\theta_1)$

$${}^1_2T = \begin{bmatrix} -s_2 & -c_2 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ c_2 & -s_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$\leftarrow \sin(\theta_2)$

$${}^2_3T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & -d_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^0_3T = \begin{bmatrix} -c_1s_2 & s_1 & c_1c_2 & c_1c_2d_3 \\ -s_1s_2 & c_1 & s_1c_2 & s_1c_2d_3 \\ c_2 & 0 & s_2 & s_2d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

#### Part a:

Base frame usually never moves so we can say: **frame{0}**

$${}^0\omega_0 = {}^0v_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

**Frame {1}** iteration(i=0):

$$1. {}^1\omega_1 = {}^0R({}^0\omega_0) + \dot{\theta}_1 \hat{z} = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{bmatrix}$$

$$2. {}^1v_1 = {}^0R({}^0v_0 + {}^0\omega_0 \times {}^0P_1) + d_1 {}^1\hat{z}_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad d_1 {}^1\hat{z}_1 \text{ goes to zero cause revolute joint here.}$$

**Frame {2}** iteration(i=1):

$$1. {}^2\omega_2 = {}^1R({}^1\omega_1) + \dot{\theta}_2 \hat{z} = \begin{bmatrix} -s_2 & 0 & c_2 \\ -c_2 & 0 & -s_2 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_2 \end{bmatrix} = \begin{bmatrix} c_2\dot{\theta}_1 \\ -s_2\dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}$$

$$2. {}^2v_2 = {}^1R({}^1v_1 + {}^1\omega_1 \times {}^1P_2) + d_2 {}^2\hat{z}_2 = \begin{bmatrix} -s_2 & 0 & c_2 \\ -c_2 & 0 & -s_2 \\ 0 & -1 & 0 \end{bmatrix} \left( \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$d_2 {}^2\hat{z}_2$  goes to zero cause revolute joint here.

**Frame {3}** iteration(i=2):

$$1. {}^3\omega_3 = {}^3R({}^2\omega_2) + \dot{\theta}_3\hat{z} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} c_2\dot{\theta}_1 \\ -s_2\dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} = \begin{bmatrix} c_2\dot{\theta}_1 \\ \dot{\theta}_2 \\ s_2\dot{\theta}_1 \end{bmatrix} \quad \dot{\theta}_2^2\hat{z}_2 \text{ goes to zero}$$

cause prismatic joint here.

$$2. {}^3v_3 = {}^3R({}^2v_2 + {}^2\omega_2 \times {}^2P_3) + \dot{d}_3{}^3\hat{z}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} \left( \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} c_2\dot{\theta}_1 \\ -s_2\dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} \times \begin{bmatrix} 0 \\ \dot{d}_3 \\ 0 \end{bmatrix} \right) +$$

$$\begin{bmatrix} 0 \\ 0 \\ \dot{d}_3 \end{bmatrix} = \begin{bmatrix} d_3\dot{\theta}_2 \\ -d_3c_2\dot{\theta}_1 \\ \dot{d}_3 \end{bmatrix}$$

**Part b:**

Find  ${}^3J$ . Note when solving for jacobian, you only need to use linear velocity.

$${}^3v = \begin{bmatrix} d_3\dot{\theta}_2 \\ -d_3c_2\dot{\theta}_1 \\ \dot{d}_3 \end{bmatrix} = \begin{bmatrix} 0 & d_3 & 0 \\ -d_3c_2 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{d}_3 \end{bmatrix} \rightarrow \dot{Y} = J\dot{X}$$

**Part c:**

Find  ${}^0J$ .

$${}^0J = {}^0R^3J = \begin{bmatrix} -c_{12} & s_1 & c_1c_2 \\ -s_{12} & c_1 & s_1c_2 \\ c_2 & 0 & s_2 \end{bmatrix} \begin{bmatrix} 0 & d_3 & 0 \\ -d_3c_2 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -s_1d_3c_2 & -c_1s_2d_3 & c_1c_2 \\ -c_1d_3c_2 & -s_1s_2d_3 & s_1c_2 \\ 0 & c_2d_3 & s_2 \end{bmatrix}$$

**Part D**

Find singularities by taking determinant: You must solve the unknowns in  $\det[{}^0J]$ . You can use levitas determinant or some software. The unknowns will be  $\theta_1, \theta_2, d_3$  and these are values for parameters that reduce DOF of robot(aka robot must avoid these values).

## 5 Jacobian Matrix - Torques and Forces on Joints

Two type of joints -revolute and prismatic.

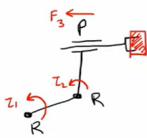
Revolute produces torque and prismatic produces force.

**Equation:**  $\tau = {}^0J\hat{F}$  where,;

$\tau$  : is the vector of torques and forces for each joint

$\hat{F}$  : the unit forces

Example



$${}^0J = \begin{bmatrix} -s_1 d_3 c_2 & -c_1 s_2 d_3 & c_1 c_2 \\ -c_1 d_3 c_2 & -s_1 s_2 d_3 & s_1 c_2 \\ 0 & c_2 d_3 & s_2 \end{bmatrix}$$

$\tau = ({}^0J)^T \cdot \hat{F}$

$$\begin{bmatrix} \tau_1 \\ \tau_2 \\ F_3 \end{bmatrix} = \begin{bmatrix} -s_1 d_3 c_2 & -c_1 s_2 d_3 & c_1 c_2 \\ -c_1 d_3 c_2 & -s_1 s_2 d_3 & s_1 c_2 \\ 0 & c_2 d_3 & s_2 \end{bmatrix}^T \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix}$$

## 6 Lagrangian Mechanics (Torques and Forces)

**Goal:** Find the torques or forces acting on a joint  
 $\mathcal{L} = \text{kinetic(KE)} - \text{potential(PE)}$

- $\text{KE} = \frac{1}{2}mv^2$
- $\text{PE} = mgh$
- $\theta = \text{position}$
- $\dot{\theta} = \text{velocity}$
- $R \rightarrow \text{torque (revolute)}$
- $P \rightarrow \text{force (prismatic)}$

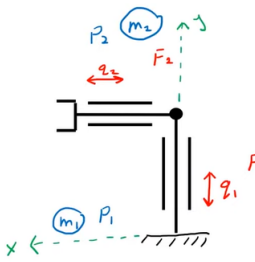
### 6.1 Algorithm

- Robot manipulator  $\rightarrow$  Position vector ( $\theta$ )
- Differentiate  $\rightarrow$  velocity vector ( $\dot{\theta}$ )
- $\text{KE} = \frac{1}{2}mv^2$ ,  $\text{PE} = mgh$
- $\mathcal{L} = \text{KE} - \text{PE}$
- $\tau = \frac{d}{dt} \left( \frac{\partial \mathcal{L}}{\partial \dot{\theta}} \right) - \frac{\partial \mathcal{L}}{\partial \theta} - [\text{external forces/torques}] \theta$  or  $\mathbf{F}$



## 6.2 Example

### Worked Example

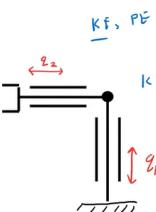


- Two prismatic joints - P1 and P2
- Masses of links - m1 and m2
- Displacement of joints - q1 and q2
- Input forces produced by actuators - F1 and F2

Position vectors

$$P_{m_1} = \begin{bmatrix} 0 \\ q_1 \end{bmatrix} \quad P_{m_2} = \begin{bmatrix} q_2 \\ q_1 \end{bmatrix}$$

Velocity vectors

$$V_{m_1} = \frac{dP_{m_1}}{dt} = \begin{bmatrix} 0 \\ \dot{q}_1 \end{bmatrix} \quad V_{m_2} = \begin{bmatrix} \dot{q}_2 \\ \dot{q}_1 \end{bmatrix}$$


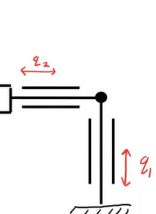
KF, PE

$$KE = \frac{1}{2} m v^2 = \frac{1}{2} m v_x^2 + \frac{1}{2} m v_y^2$$

$$KE = \frac{1}{2} m (v_x^2 + v_y^2)$$

$$KE_{m_1} = \frac{1}{2} m_1 (\dot{q}_1^2) \quad ; \quad KE_{m_2} = \frac{1}{2} m_2 (\dot{q}_2^2 + \dot{q}_1^2)$$

PE = mgh ← height (vertical component)

$$PE_{m_1} = m_1 g q_1 \quad , \quad PE_{m_2} = m_2 g q_1$$


Worked Example

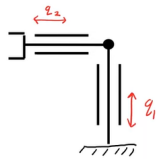
$$\mathcal{L} = KE - PE$$

$$\mathcal{L} = KE_{m_1} + KE_{m_2} - PE_{m_1} - PE_{m_2}$$

$$\mathcal{L} = \frac{1}{2} m_1 \dot{q}_1^2 + \frac{1}{2} m_2 (\dot{q}_2^2 + \dot{q}_1^2) - m_1 g q_1 - m_2 g q_1$$

$$\mathcal{L} = m_1 \left( \frac{1}{2} \dot{q}_1^2 - g q_1 \right) + m_2 \left( \frac{1}{2} \dot{q}_2^2 + \frac{1}{2} \dot{q}_1^2 - g q_1 \right)$$

## Worked Example



Prismatic  
 $F_{m_1}, F_{m_2}$

$$\tau = \frac{d}{dt} \left( \frac{\partial L}{\partial \dot{\theta}} \right) - \frac{\partial L}{\partial \theta} - \left[ \text{External Forces / Torques} \right]_{\theta}$$

$$F_{m_1} = \frac{d}{dt} \left( \frac{\partial L}{\partial \dot{z}_1} \right) - \frac{\partial L}{\partial z_1} - F_1(z_1)$$

$$F_{m_2} = \frac{d}{dt} \left( \frac{\partial L}{\partial \dot{z}_2} \right) - \frac{\partial L}{\partial z_2} - F_2(z_2)$$

$$F_{m_1} = \frac{d}{dt} (m_1 \dot{z}_1 + m_2 \dot{z}_2) + m_1 g + m_2 g - F_1(z_1)$$

$$F_{m_2} = \frac{d}{dt} (m_2 \dot{z}_2) - 0 - F_2(z_2) \quad m_2 \ddot{z}_2 \leftarrow \text{acc}$$

$\frac{\partial L}{\partial \dot{z}_1} = m_1 \dot{z}_1 + m_2 \dot{z}_2$   
 $\frac{\partial L}{\partial \dot{z}_1} = -m_1 g - m_2 g$   
 $\frac{\partial L}{\partial \dot{z}_2} = m_2 \dot{z}_2$   
 $\frac{\partial L}{\partial z_2} = 0$

## 7 Trajectory Planning and Generation

**Trajectory:** Time history of position, velocity and acceleration for each DOF.

- Run time
- Path update rate (60-2000hz)
- You also want smooth motion(minimize wear and tear)
- **Via points:** auxiliary/intermediate points between initial and final positions
- Optimization problem because there are many ways to potentially go from initial to final

### Path Generation Methods

- Joint Space Schemes  $\rightarrow$  using function of joint angles(in terms of  $\theta$ ), each via point(including final and initial) is converted to a set of joint angles using inverse kinematics
- Cartesian Schemes  $\rightarrow$  directly uses the position and orientation of the robot frames along with the path points as specified by the user to generate the trajectory.

In general, joint space schemes are easier to use and compute. This is because cartesian schemes require inverse kinematics to be used at every single point along the path, in the path update. So higher computational burden on the robot.

## 7.1 Joint Space Schemes

Is split into two sub methods:

- Cubic Polynomials (can be even higher order)
- Parabolic Blends

## 7.2 Cubic Polynomials

The idea here is to find an equation (3rd degree polynomial) that satisfies the following constraints:

- $\theta(t_0) = \theta_0$  aka the initial position at time 0 is the initial position of the robot
- $\dot{\theta}(0) = 0$  aka the derivative of the initial position has velocity 0
- $\theta(t_f) = \theta_f$  the final position at time f, is at the intended final position

To model a polynomial with 4 constraints you need at least a 3rd degree polynomial. If you add via points, then you can increase the degree of the polynomial by adding additional constraints.

**Form of functions:**

- $\theta(t) = a_0 + a_1t + a_2t^2 + a_3t^3$  Position
- $\dot{\theta}(t) = a_1 + 2a_2t + 3a_3t^2$  Velocity
- $\ddot{\theta}(t) = 2a_2 + 6a_3t$  Acceleration

**Example:**

**Cubic Polynomials - Example**

From A  $\rightarrow$  B

$t=0$

$\theta(0) = \theta_0 = a_0 + a_1(0) + a_2(0^2) + a_3(0^3) = 2 \rightarrow a_0 = 2$

$\dot{\theta}(0) = 0 = a_1 + 2a_2(0) + 3a_3(0^2) \rightarrow a_1 = 0$

$t=2$

$\theta(2) = 2 + a_2(2^2) + a_3(2^3) = 15 \rightarrow 4a_2 + 8a_3 = 13$

$\dot{\theta}(2) = 0 = 2a_2 + 3a_3(2^2) \rightarrow 4a_2 + 12a_3 = 0$

$\rightarrow \textcircled{2}$

$\textcircled{1} \text{ \& } \textcircled{2} \Rightarrow a_2 = 9.75$   
 $a_3 = -3.25$

**Position**

$\theta(t) = a_0 + a_1t + a_2t^2 + a_3t^3$

$\dot{\theta}(t) = a_1 + 2a_2t + 3a_3t^2$

$\ddot{\theta}(t) = 2a_2 + 6a_3t$

$\theta(t) = 2 + 9.75t^2 - 3.25t^3$  (A  $\rightarrow$  B)

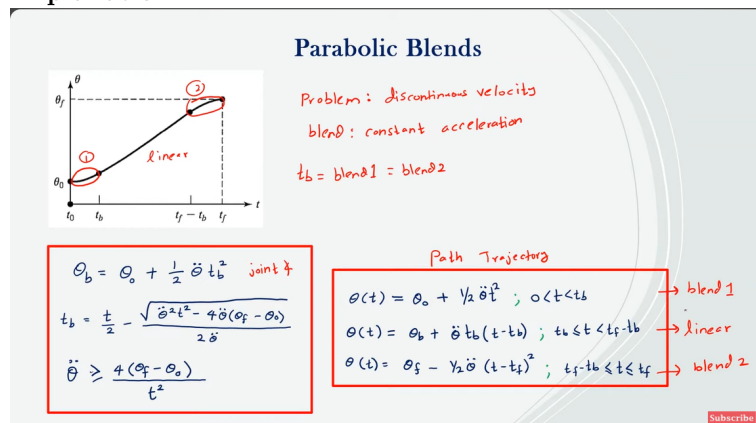
$\theta(t) = 15 + 3.75t^2 - 1.25t^3$  (B  $\rightarrow$  C)

Subscript

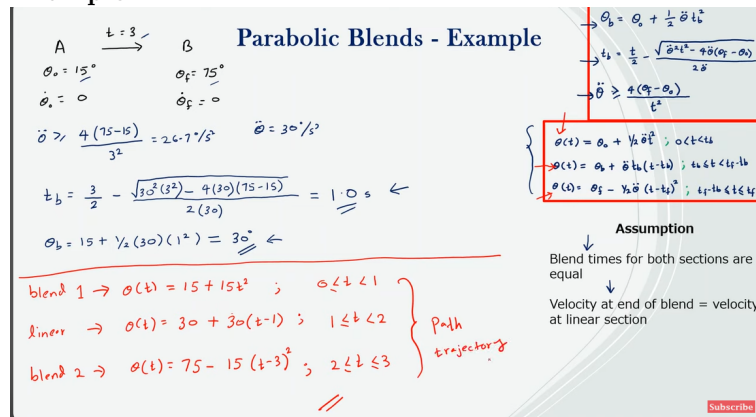
## 7.3 Parabolic Blends

To solve discontinuous velocities, we use parabolic blends.

**Explanation:**



**Example:**



## 8 Credit

These are my notes from the robotics playlist youtube channel @thatsengineering5235, all of the external images used are examples from their videos.