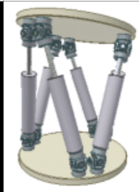
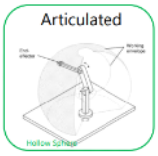
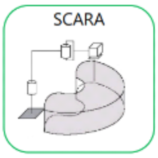
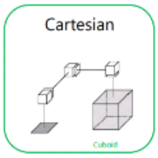
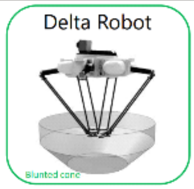
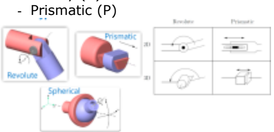
1: Robot Classification

Monday, 26 June 2023   10:33 AM

	Serial robot (general)	Articulated	SCARA	Cartesian	Cylindrical	Parallel (general)	Parallel - Stewart platform	Parallel - Delta
Image	[redacted]					[redacted]		
Description	[redacted]					[redacted]	- base and platform - 6 actuated struts - spherical joints	- base and platform - three parallelograms **maintains end-effector orientation**
Joint configuration	[redacted]	RRRRRR (5-6 DOF typically)	RRP (3-4 DOF)	PPP	RPP	[redacted]	6 prismatic links	- different configurations
Work envelope	[redacted]	hollow sphere 		cuboid 		[redacted]	N/A	
Pros	- easy forward kinematics - large workspace	- flexible - versatile	- fast - simple kinematics - cheap - high duty cycle (fast) - rigid in z-axis	- simplest kinematics - can be very stiff - can be very large		- greater structural rigidity -> greater accuracy - easy inverse kinematics - small position error (averages out) - max force = sum of all actuator forces - high rigidity		
Cons	- difficult inverse kinematics - low rigidity - accumulative position error	- complex kinematics - low rigidity -> links and joints must be made excessively strong to eliminate deflections - slow, heavy, massive - max force is limited by minimum actuator force		- typically slower		- small workspace - difficult forward kinematics		
Applications		- e.g. car spray painting	- pick and place	- e.g. 3D printers - e.g. storage and sorting systems			- radio telescope	- high speed pick and place

Joint types:
- Rotary (R)
- Prismatic (P)



E.g.

Compliance = "the extent to which a robot end effector moves as a result of an applied force"

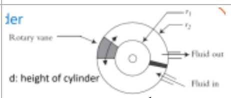
1: safety

Monday, 26 June 2023 10:54 AM

I. 4024.33**01**-2017 – Robots
    a. Aimed at **robot designer**
II. 4024.33**02**-2017 – Robot systems and integration
    a. Aimed at robot systems engineers, **integrators and installers**
III. 4024.33**03**-2017 – Collaborative robots
    a. (robots where the human interacts directly with the robot)
    b. Aimed at **integrators of collaborative systems**

2: Actuators

Saturday, 24 June 2023 7:27 pm

	Details	Pros	Cons
Motors	Dc, stepper, ac		
Gears	Output torque $T_l = N T_m$ Output speed $\dot{\theta}_l = \frac{1}{N}\dot{\theta}_m$	- Better control, precision - higher compliance than hydraulics - clean - reliable - low maintenance - can be spark free - gears can reduce inertia on motors	- low stiffness - needs reduction gear -> backlash, cost, weight - needs brakes (for when no power)
Hydraulics (linear)	Output force: $F = p * A$ (linear) Flow rate: $Q = \frac{\Delta V}{t}$ $\Delta V = v * t * A$, A = area of cylinder piston	- highest power to weight ratio - stiff, high accuracy, good response - no gear reduction needed - wide range of speeds	- May leak, dirty - requires pump + reservoir - expensive, noisy - susceptible to dirt in the oil - low compliance
Hydraulics (Rotary)	 Output torque: $T = \frac{1}{2}pd\left(r_2^2 - r_1^2\right)$ Flow rate: $Q = \frac{\Delta V}{t} = \omega * \left(r_2^2 - r_1^2\right) * d/2$ $\Delta V = \theta * \left(r_2^2 - r_1^2\right) * d/2$ $\theta = \omega * t$	^	^
Pneumatics	- Lower power - Better compliance (deformation) - Environmentally friendly - Useful for soft robots	- reliable components - no leaks or sparks - simple, inexpensive - low pressure compared to hydraulics - good for pick and place - compliant	- noisy - low stiffness - requires pressurised air + filter - deforms constantly under load - inaccurate response
Piezoelectric	Controlled via voltage		
Magnetic	Controlled via external magnetic field		

Actuator Comparison

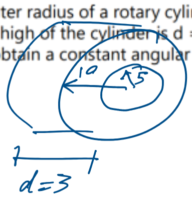
Hydraulic	Electric	Pneumatic
+ Highest power/weight ratio	+ Better control, precision	+ Reliable components
+ Stiff system, high accuracy, better response	+ Higher compliance than hydraulics	+ No leaks or sparks
+ no reduction needed	+ Reduction gears reduce inertia on motor	+ Simple and inexpensive
+ Wide range of speeds	+ Clean	+ Low pressure compared with hydraulics
- May leak, dirty	+ Reliable, low maintenance	+ Good for pick and place
- Requires pump, reservoir etc.	+ Can be spark-free for explosive environments	+ Compliant
- Expensive and noisy	- Low stiffness	- Noisy, low stiffness
- Susceptible to dirt in oil	- Needs reduction gears -backlash, cost, weight	- Require pressurised air, filter
- Low compliance	- Need brakes for when power removed	- Deform constantly under load, and inaccurate response

Example 1: Hydraulic Rotary Cylinder

$\omega = 1$ rad/s

The inner and outer radius of a rotary cylinder are 5cm and 10cm, respectively. The high of the cylinder is d = 3cm. Calculate the Pump flow rate (Q) to obtain a constant angular velocity of w = 1 rad/s.

$Q = \Delta V / t$



$\Delta V = \omega t \left((10 \times 10^{-2})^2 - (5 \times 10^{-2})^2 \right) \cdot \frac{d}{2}$

$\therefore Q = 112 \times 10^{-6}\ m^3/s$

$= 0.112\ L/s$

2: Sensors

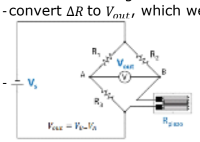
Saturday, 24 June 2023 8:20 PM

<table>
<tr><td>Position sensors:</td><td></td><td>Application</td></tr>
<tr><td>Potentiometer</td><td>
$\frac{\theta_{wiper}}{\theta_{max}} = \frac{V_{out}}{V_{max}} = \frac{V_{32}}{V_{31}} = \frac{R_{wiper}}{R_{max}}$</td><td>- non-continuous rotation motors
- small, inexpensive</td></tr>
<tr><td>Encoders - Grey Code Disk</td><td>

<table><tr><th>Position</th><th>C</th><th>B</th><th>A</th></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>2</td><td>0</td><td>1</td><td>1</td></tr><tr><td>3</td><td>0</td><td>1</td><td>0</td></tr><tr><td>4</td><td>1</td><td>1</td><td>0</td></tr><tr><td>5</td><td>1</td><td>1</td><td>1</td></tr><tr><td>6</td><td>1</td><td>0</td><td>1</td></tr><tr><td>7</td><td>1</td><td>0</td><td>0</td></tr></table></td><td>- motors
- gives position without a reference position</td></tr>
<tr><td>Encoders - rotary</td><td>- counts pulses to get distance
- differentiate to get velocity (number of pulses / time)</td><td>- requires a reference (original position)</td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td>Acceleration sensors:</td><td></td><td></td></tr>
<tr><td>Capacitive sensing</td><td>- e.g. accelerometers
- $C = \frac{\epsilon A}{d}, \epsilon = permittivity\ constant, A = area, d = dist.\ between\ plates$
- force causes plates to move -> capacitance change is detected and distance change is calculated -> force can be calculated -> acceleration can be calculated since we know the mass of the moving plate (F=ma)

Amplifier circuit:
$V_{out} = \frac{V_p(C_{Sens} - C_{Ref})}{C_F}$

see example</td><td>- acceleration sensing
- tactile force sensing</td></tr>
<tr><td>Piezoresistive sensing</td><td>- when material bends, resistance changes
- $R = \frac{\rho L}{A}$
- measure ΔR

Wheatstone bridge circuit:
- convert ΔR to V_{out}, which we can measure


- $R_1 = R_2 = R_3 = R_s = R$
- $V_{out} = \frac{1}{4} * \frac{\Delta R}{R} * V_s$

Amplifier circuit:
- to amplify ΔV_{out}
- $V_{out-opAmp} = V_{out} * \left(1 + \frac{R_{internal}}{R_{gain}}\right)$
• R_int is given by manufacturer, R_gain is chosen
see example</td><td>- tactile force sensing</td></tr>
<tr><td>(extension) optical sensing</td><td>- force/pressure deflects the light</td><td>3D force measurement, an array of sensors</td></tr>
</table>

decimal	binary	graycode
0	0	0
1	1	1
2	10	11
3	11	10
4	100	110
5	101	111
6	110	101
7	111	100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

Example 4 – Capacitive tactile sensors

The distance between the two parallel electrodes of a capacitive tactile sensors is d = 1mm. Under a touching force of 1N, this distance changes to d1 =0.99mm. Assuming that F and ΔC has a linear relationship, calculate the touching force if ΔC/C = 0.5%.

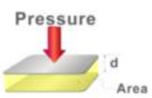
Answer:

$C = \frac{\varepsilon A}{d}$ $\Delta C_1 = \varepsilon A(\frac{1}{d_1} - \frac{1}{d_0})$

Under 1N force

$\frac{\Delta C_1}{C} = \frac{(\frac{1}{0.99} - 1/1)}{1/1} \approx 1\%$

As the relationship between applied force and capacitance change is linear,



$\frac{F_2}{F_1} = \frac{\Delta C_2}{\Delta C_1} = \frac{0.5\%}{1\%} = 0.5$

$F_2 = 0.5 \times F_1 = 0.5$ (N)

Under 1N Force, $\frac{\Delta C}{C} = \frac{\frac{\varepsilon A}{d1} - \frac{\varepsilon A}{d_0}}{\frac{\varepsilon A}{d_0}} = \frac{\cancel{\varepsilon A}\left(\frac{1}{0.99 \times 10^{-3}} - \frac{1}{1 \times 10^{-3}}\right)}{\cancel{\varepsilon A}\left(\frac{1}{1 \times 10^{-3}}\right)} \doteq 0.010101 \ldots \doteq 1\%$

For F=?, $\frac{\Delta C}{C} = 0.5\%$. Assume $F = a\,\Delta C$ => $\frac{F_1}{F_2} = \frac{\Delta C_1}{\Delta C_2} = \frac{\Delta C_1 / C}{\Delta C_2 / C}$

=> $\frac{F}{1N} = \frac{0.5\%}{1\%}$ => $F = 0.5N$

Example 5 – Piezoresistive tactile sensors

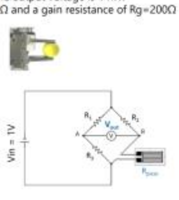
A robotic tactile sensor is connected to a Wheatstone bridge, where R1 = R2= R3 = Rs = 500 Ω. Applying a force of 1N changes the sensor resistance (Rs) to 501 Ω. Assume that the relationship between the applied forces and the output voltages is linear, calculate a gripping force if the output voltage is 1 mV.

If an instrumental amplifier AD623 with an internal resistance of 100kΩ and a gain resistance of Rg=200Ω is used, calculated the output voltage.

Answer:

At F1 = 1N, $V_{out,1N} = \frac{1}{4}\frac{\Delta R}{R} V_S = \frac{1}{4} \times \frac{501 - 500}{500} \times 1$

$V_{out,1N} = 0.0005\ (V) = 0.5(mV)$



As the relationship between F and Vout is linear, the gripping force is

$F = 1(N) \times \frac{V_{out,F}}{Vout, 1N} = 1(N) \times \frac{1\ (mV)}{0.5(mV)} = 2(N)$

Gain = 1+R/Rg = 1+100,000/200= 501

Output voltage from the Opamp = Gain × $1(mV) = 501(mV)$

Case 1: $F = 1N, \Delta R = 1\Omega$ $V_{out,1N} = \frac{1}{4} \cdot \frac{1}{500} \cdot (1V) = \frac{1}{2000} V$

Case 2:
$V_{out} = 1mV = \frac{1}{4} \cdot \frac{\Delta R}{R} V_s$

$F = a \cdot V_{out}$

=> $\frac{F}{1N} = \frac{1mV}{V_{out(1N)}}$

$F = 2\ N$

$$V_{out,\,opAmp} = V_{out}\left(1 + \frac{R_{int}}{R_{gain}}\right)$$
$$= 1mV\left(1 + \frac{100k}{200}\right)$$
$$= 0.501\ V \quad \text{or} \quad 501\,mV$$

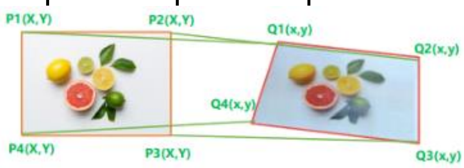
2: Computer Vision

Saturday, 24 June 2023 9:58 PM

Object detection via Colour masking:

```
% Read in the image from file
image = imread("TestImage.jpg");

% Colour mask: (select the third element of an image to specify R,G or B)
mask = (image(:,:,2) > MIN_GREEN) & (image(:,:,1) < MAX_RED);

% filtering: (remove small artifacts)
mask = bwareaopen(mask,MIN_PIXELS_FOR_AN_OBJECT);

% Fill gaps:
se = strel('disk', DISK_SIZE);

% Detect circles, and get their centers + radii
[centers,radii] = imfindcircles(mask,[MIN_RADIUS, MAX_RADIUS]);
```

Planar Homography & Perspective Transforms:

- requires 4 pairs of points to define the homography matrix
- 

- $$\begin{pmatrix} P1_X \\ P1_Y \\ 1 \end{pmatrix} = \boldsymbol{g} \cdot \begin{pmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & 1 \end{pmatrix} \begin{pmatrix} Q1_x \\ Q1_y \\ 1 \end{pmatrix}$$

Here ***g*** *is a gain matrix and is a function of* H_{31} *and* H_{32}.

In matlab:

```
%% Transform the image:

% Real world coordinates:
width = 650;
height = 450;
p1 = [0,0]; p2 = [0,height]; p3 = [width,0]; p4 = [width,height];
% (centers = location of the corresponding points on the image)

% Get homography matrix
tform = fitgeotrans(centers, [p3;p1;p2;p4],'projective');
H_matrix= tform.T;

% Warp image and crop as required
image_flat = imwarp(image,tform,OutputView=imref2d([height, width]));
figure(4);
imshow(image_flat);

% (optional) Calculate a flat image coordinate from an original image point
p_image_orig = [10, 20]
p_image_flat = transformPointsForward(tform, p_image_orig)

% (optional) Calculate an original image coordinate from a flat image point
p_image_flat = [10, 20]
p_image_real = transformPointsInverse(tform, p_image_flat)
```

3: Coordinate frame transforms

Saturday, 24 June 2023 3:01 PM

To convert from one coordinate frame to another

	Description	Example	Notes	Matlab
				Matlab https://www.petercorke.com/RTB/r9/html/index.html
${}^{A}\xi_{B}$	Pose of frame B with respect to frame A			
Translation vector	d			
Rotation matrix	R $R_Y(\theta) = \begin{bmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{bmatrix}$	2D: ${}^{A}R_B = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}$ 3D: $R_{zyx}(\phi,\theta,\psi) = R_z(\phi).R_y(\theta).R_x(\psi)$ $= \begin{bmatrix} c_\phi & -s_\phi & 0 \\ s_\phi & c_\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ -s_\theta & 0 & c_\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\psi & -s_\psi \\ 0 & s_\psi & c_\psi \end{bmatrix}$ $= \begin{bmatrix} c_\phi c_\theta & -s_\phi c_\psi + c_\phi s_\theta s_\psi & s_\phi s_\psi + c_\phi s_\theta c_\psi \\ s_\phi c_\theta & c_\phi c_\psi + s_\phi s_\theta s_\psi & -c_\phi s_\psi + s_\phi s_\theta c_\psi \\ -s_\theta & c_\theta s_\psi & c_\theta c_\psi \end{bmatrix}$	- - Forms a special orthogonal group: $\boldsymbol{SO(n)}$	- rotx(angle) - roty(angle) - rotz(angle) 2D: - rot2
Homogeneous Transform	- $T = \begin{bmatrix} R & d \\ 0\,0\,0 & 1 \end{bmatrix}$ - ${}^{A}T_C = {}^{A}T_B\,{}^{B}T_C$ - ${}^{A}T_C = {}^{A}T_B\,{}^{B}T_C$ pose of frame C with respect to frame A	$H = Rot_{x,\alpha} Trans_{x,b} Trans_{z,d} Rot_{z,\theta}$ $= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\alpha & -s\alpha & 0 \\ 0 & s\alpha & c\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & b \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\theta & -s\theta & 0 & 0 \\ s\theta & c\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ $\begin{bmatrix} {}^{A}\boldsymbol{p} \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} {}^{A}R_B & {}^{A}\boldsymbol{p}_{Bo} \\ 0\,0\,0 & 1 \end{bmatrix}}_{{}^{A}T_B} \begin{bmatrix} {}^{B}\boldsymbol{p} \\ 1 \end{bmatrix}$ **Homogeneous Transformation**	- Forms a special euclidean group: $\boldsymbol{SE(n)}$	- TR = rt2tr(R, d)
Pure rotation, $R(\theta)$	$Rot_{x,\alpha} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\alpha & -sin\alpha & 0 \\ 0 & sin\alpha & cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ $Rot_{z,\theta} = \begin{bmatrix} cos\theta & -sin\theta & 0 & 0 \\ sin\theta & cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	(D) ${}^{A}T_B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ - rotate about x axis 180 degrees		`-trotx(theta)` `-troty(theta)` `-trotz(theta)`
Pure translation, $Q(d)$	$Trans_{z,d} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d \\ 0 & 0 & 0 & 1 \end{bmatrix}$	(A) ${}^{A}T_B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 5 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ - translate 5 units along the y axis		`-transl(x, y, z)` `-transl([x, y, z])`

Special Orthogonal Group: $\boldsymbol{SO(n)}$
- if a matrix is square and its transpose is equal to its inverse matrix, it is orthogonal
  - Hence, R is orthogonal
- to be part of the special group ($R \in SO(n)$), R must have
  - The columns (and rows) of R are mutually orthogonal
  - Each column (and row) of R is a unit vector
  - $\det(R) = 1$, hence the length of the vector is unchanged

Special Euclidean group: $\boldsymbol{SE(n)}$

$T = \begin{bmatrix} R & d \\ 0\,0\,0 & 1 \end{bmatrix}$

where
- $\boldsymbol{d} \in \mathbb{R}^3$ is translation operator
- $\boldsymbol{R} \in SO(3)$ is rotation operator

so, (d,R) forms the Special Euclidean Group SE(3)

$$\boxed{SE(3) = \mathbb{R}^3 \times SO(3)}$$

Compounding Rotation Matrices:

$$R_{zyx}(\phi,\theta,\psi) = R_z(\phi).R_y(\theta).R_x(\psi)$$

$$= \begin{bmatrix} c_\phi & -s_\phi & 0 \\ s_\phi & c_\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ -s_\theta & 0 & c_\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\psi & -s_\psi \\ 0 & s_\psi & c_\psi \end{bmatrix}$$

$$= \begin{bmatrix} c_\phi c_\theta & -s_\phi c_\psi + c_\phi s_\theta s_\psi & s_\phi s_\psi + c_\phi s_\theta c_\psi \\ s_\phi c_\theta & c_\phi c_\psi + s_\phi s_\theta s_\psi & -c_\phi s_\psi + s_\phi s_\theta c_\psi \\ -s_\theta & c_\theta s_\psi & c_\theta c_\psi \end{bmatrix}$$

- ("c" represents cos, "s" represents sin)
- Compounding: (rotate about z, then y, then x)

- if rotating about current axes, add subsequent rotations to the right
  - rotation order will read left to right
  - $${}^0R_3 = \underbrace{R_{x\alpha}}_{\text{Step 1}}.\underbrace{R_{y\beta}}_{\text{Step 2}}.\underbrace{R_{z\gamma}}_{\text{Step 3}}$$
  - if rotating about fixed axes, add subsequent rotations to the left
    - rotation order will read right to left
    - $${}^0R_3 = \overset{\text{Rotation3}}{R_z(\psi)}.\ \overset{\text{Rotation 2}}{R_y(\theta)}.\ \overset{\text{Rotation 1}}{R_x(\phi)}$$

Compounding using T:

- ${}^AT_C = {}^AT_B{}^BT_C$
-

3: Angle transform representations

Monday, 26 June 2023 3:02 PM

Euler angles:

Roll-pitch-yaw representation:
- ZYX and XYZ conventions
- (rotations about current axes)

ZYZ: Classic euler angles

$$R_{ZYZ} \quad = \quad R_{z,\phi}R_{y,\theta}R_{z,\psi}$$

- rotations are about "current" axes

Axis angle representation:
- k = axis of rotation
- theta = angle of rotation about axis k

given $R \in SO(3)$,

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

$$\theta = \quad \cos^{-1}\left(\frac{r_{11} + r_{22} + r_{33} - 1}{2}\right)$$

$$k \quad = \quad \frac{1}{2\sin\theta}\begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix}$$

e.g.

4: DH convention

Saturday, 24 June 2023 12:28 PM

DH Table:

i-1 is a revolute joint

Link	Joint angle (degree)	Link offset (cm)	Link length (cm)	Link twist (degree)
i	$\theta_i + \theta_i^*$	d_i^*	a_i	α_i

i-1 is a prismatic joint

Link	Joint angle (degree)	Link offset (cm)	Link length (cm)	Link twist (degree)
i	θ_i^*	$d_i + d_i^*$	a_i	α_i

- for each link (joint)
- (prismatic joint is a linear actuator)

transform from frame {i-1} to {i} has four steps:

1. rotate by joint angle (θ_i) about joint axle axis (z_{i-1})
2. translate by link offset (d_i) along joint axle axis (z_{i-1})
3. translate by link length (a_i) along x_i (towards new joint's origin)
4. rotate by link twist (α_i) about x_i (to align the z_i axis)

$$^{i-1}T_i = R_{(i-1)}(\theta_i).Q_{(i-1)}(d_i).Q_i(a_i).R_i(\alpha_i)$$

$$^{i-1}T_i = \begin{pmatrix} cos\theta_i & -sin\theta_i\, cos\alpha_i & sin\theta_i sin\alpha_i & a_i cos\theta_i \\ sin\theta_i & cos\theta_i cos\alpha_i & -cos\theta_i sin\alpha_i & a_i sin\theta_i \\ 0 & sin\alpha_i & cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Example in MATLAB:

```
theta = deg2rad([10 50 65]); % radians
d =       10^-2*[32 0   0];  % length unit (e.g. m)
a =       10^-2*[0  11 16];  % length unit (e.g. m)
alpha = deg2rad([90 0  0 ]); % radians
L(1) = Link('revolute', 'd', d(1), 'a', a(1), 'alpha', alpha(1), 'offset', 0);
L(2) = Link('revolute', 'd', d(2), 'a', a(2), 'alpha', alpha(2), 'offset', 0);
L(3) = Link('revolute', 'd', d(3), 'a', a(3), 'alpha', alpha(3), 'offset', 0);

robot = SerialLink(L, 'name', 'ArticulatedRobot'); % Will print DH table

% Calculate combined transformation matrix:
T_0_to_n = robot.fkine(theta')

% Transform vector from end-effector frame to base frame:
p_n = [1; 1; 1];
p_0 = T_0_to_n*p_n

% Calculate an individual transformation matrix:
i = 1;
T_ind = trotz(theta(i))*transl(0, 0, d(i))*transl(a(i), 0, 0)*trotx(alpha(i))

% Method 2:
frameTransformationMatrix(theta(i), d(i), a(i), alpha(i))


% Function to calculate the frame transformation matrix
 function T = frameTransformationMatrix(theta,d,a,alpha)

T = SE3([[cos(theta),  -sin(theta)*cos(alpha),  sin(theta)*sin(alpha), a*cos(theta)];
        [ sin(theta),   cos(theta)*cos(alpha), -cos(theta)*sin(alpha), a*sin(theta)];
        [ 0,            sin(alpha),             cos(alpha),            d];
        [ 0,            0,                     0,                      1]]);
end
```

5: The Jacobian

Sunday, 2 July 2023 3:10 PM

$$J_i = \begin{pmatrix} {}^0\mathbf{z}_{i-1} \times ({}^0\mathbf{o}_n - {}^0\mathbf{o}_{i-1}) \\ {}^0\mathbf{z}_{i-1} \end{pmatrix} \text{ if joint } i \text{ is revolute;}$$

Or

$$J_i = \begin{pmatrix} {}^0\mathbf{z}_{i-1} \\ 0 \end{pmatrix} \text{ if joint } i \text{ is prismatic.}$$

where

$$\begin{array}{cccc} & & {}^0\boldsymbol{z}_i & {}^0\boldsymbol{O}_i \\ \end{array}$$

$$ {}^0T_i = \begin{pmatrix} {}^0R_i & {}^0\mathbf{o}_i \\ 0 & 1 \end{pmatrix} = \begin{array}{cccc} & & {}^0\boldsymbol{z}_i & {}^0\boldsymbol{O}_i \\ \end{array}\begin{pmatrix} r_{1,1} & r_{1,2} & r_{1,3} & o_{1,4} \\ r_{2,1} & r_{2,2} & r_{2,3} & o_{2,4} \\ r_{3,1} & r_{3,2} & r_{3,3} & o_{3,4} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

and

$$ {}^{i-1}T_i = \begin{pmatrix} cos\theta_i & -sin\theta_i cos\alpha_i & sin\theta_i sin\alpha_i & a_i cos\theta_i \\ sin\theta_i & cos\theta_i cos\alpha_i & -cos\theta_i sin\alpha_i & a_i sin\theta_i \\ 0 & sin\alpha_i & cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\dot{d} = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\theta} \\ \dot{\phi} \\ \dot{\varphi} \end{pmatrix} = \begin{pmatrix} \frac{\delta f_1}{\delta q_1} & \frac{\delta f_1}{\delta q_2} & \cdots & \frac{\delta f_1}{\delta q_n} \\ \frac{\delta f_2}{\delta q_1} & \frac{\delta f_2}{\delta q_2} & \cdots & \frac{\delta f_2}{\delta q_n} \\ \frac{\delta f_3}{\delta q_1} & \frac{\delta f_3}{\delta q_2} & \cdots & \frac{\delta f_3}{\delta q_n} \\ \frac{\delta f_4}{\delta q_1} & \frac{\delta f_4}{\delta q_2} & \cdots & \frac{\delta f_4}{\delta q_n} \\ \frac{\delta f_5}{\delta q_1} & \frac{\delta f_5}{\delta q_2} & & \frac{\delta f_5}{\delta q_n} \\ \frac{\delta f_6}{\delta q_1} & \frac{\delta f_6}{\delta q_2} & & \frac{\delta f_6}{\delta q_n} \end{pmatrix} \begin{pmatrix} q_1 \\ q_2 \\ \cdot \\ \cdot \\ \cdot \\ q_n \end{pmatrix} \Rightarrow \boxed{\begin{pmatrix} v \\ \omega \end{pmatrix} = \begin{bmatrix} J_v \\ J_\omega \end{bmatrix} \dot{q}}$$

Find $\boldsymbol{J_v}$ and $\boldsymbol{J_\omega}$ from DH parameters

- end_effector_velocity = J * joint_velocities;

5: Inverse Kinematics

Tuesday, 18 July 2023 5:44 PM

3 Methods:

1. kinematic decoupling
   a. requires 6 joints, and last 3 joints intersect at a single point (wrist centre)
   b. first solve q1,q2,q3 for the position of wrist center
   c. then solve q4,q5,q6 for orientation of wrist
2. Algebraic
   a. e.g. 2 link manipulator
3. Numerical methods

Given a transform matrix

$$^{0}T_n = \begin{pmatrix} R & \boldsymbol{p} \\ 0 & 1 \end{pmatrix},$$

Find the joint variables.

Method 1:

Kinematic Decoupling

- Step 1: find q_1, q_2, q_3 such that $\boldsymbol{p}_c = \begin{pmatrix} x_c \\ y_c \\ z_c \end{pmatrix}$
- Step 2: use q_1, q_2, q_3 to evaluate $^{0}R_3$

$$^{0}T_3(q_1, q_2, q_3) = {}^{0}T_1 {}^{1}T_2 {}^{2}T_3 = \begin{pmatrix} ^{0}R_3 & ^{0}P_3 \\ 0 & 1 \end{pmatrix}$$

- Step3: The orientation of the wrist center is a function of q_4, q_5 and q_6.
  Find a set of Euler angles corresponding to the rotation matrix:
  - $^{3}R_6 = (^{0}R_3)^{-1} R = (^{0}R_3)^{T} R$

Inverse Jacobian:

$$\boxed{\dot{q} = {}^{0}J_n^{-1} \, {}^{0}\dot{d}_n}$$

- convert joint linear velocity (d_dot) to joint angular velocity (q_dot)

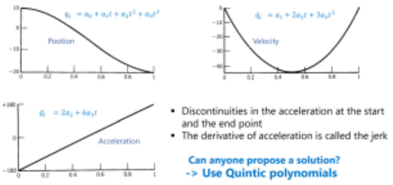
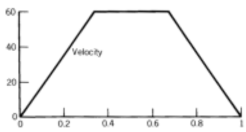
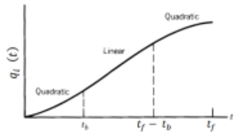
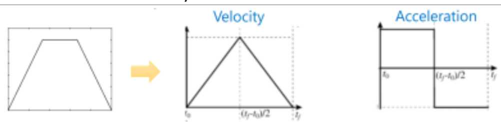
7: Manipulability

Saturday, 29 July 2023 1:58 PM

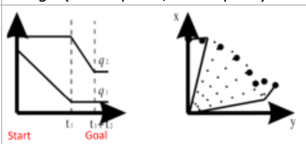
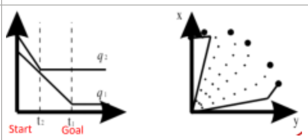
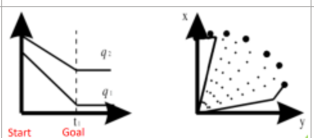
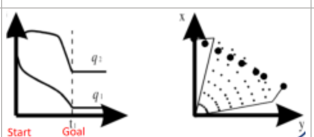
- $m = |\det(J(\mathbf{q}))|$

7: Trajectory Generation

Saturday, 29 July 2023 3:38 PM

Point to point trajectories:		
Cubic Polynomial		syms a0 a1 a2 a3 pos_0 pos_f vel_f vel_0 t_f eqn1 = a0 + a1*power(0, 1) + a2*power(0, 2) + a3*power(0, 3) == pos_0; eqn2 = a0 + a1*power(t_f, 1) + a2*power(t_f, 2) + a3*power(t_f, 3) == pos_f; eqn3 = a1 + 2*a2*0 + 3*a3*power(0, 2) == vel_0; eqn4 = a1 + 2*a2*t_f + 3*a3*power(t_f, 2) == vel_f; solve(eqn1, eqn2, eqn3, eqn4, a0, a1, a2, a3) ❑ Assume $t_0 = 0$ results in, $a_0 = q_0;$ $a_1 = \dot{q}_0;$ $a_2 = \frac{3(q_1 - q_0) - (2\dot{q}_0 + \dot{q}_1)t_f}{t_f^2};$ $a_3 = \frac{2(q_0 - q_1) - (\dot{q}_0 + \dot{q}_1)t_f}{t_f^3}$
Quintic Polynomial	$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5$ $q_0 = a_0 + a_1 t_0 + a_2 t_0^2 + a_3 t_0^3 + a_4 t_0^4 + a_5 t_0^5$ $v_0 = a_1 + 2a_2 t_0 + 3a_3 t_0^2 + 4a_4 t_0^3 + 5a_5 t_0^4$ $\alpha_0 = 2a_2 + 6a_3 t_0 + 12a_4 t_0^2 + 20a_5 t_0^3$ $q_f = a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3 + a_4 t_f^4 + a_5 t_f^5$ $v_f = a_1 + 2a_2 t_f + 3a_3 t_f^2 + 4a_4 t_f^3 + 5a_5 t_f^4$ $\alpha_f = 2a_2 + 6a_3 t_f + 12a_4 t_f^2 + 20a_5 t_f^3$	time = 0:0.001:5; % 5 seconds, sampling every 0.001 second q0 = 0; % initial pos qf = 1; % final pos v0 = 5; % initial vel vf = 0; % final vel [S,SD,SDD] = tpoly(q0, qf, time,v0,vf); % save data tpoly(q0, qf, time,v0,vf); % plot % Also, for multiple joints: q0 = [0 2]; qf = [1 -1]; % v_max = 50; figure mtraj(@tpoly, q0, qf, time)
Trapezoidal / Linear segments with parabolic blends	 blend time is given by: $\boxed{t_b = \frac{q_0 - q_f + V t_f}{V}}$ and $0 < t_b \leq t_f/2$ Rearranging: $\boxed{\frac{q_f - q_0}{t_f} < V \leq \frac{2(q_f - q_0)}{t_f}}$ where V = max velocity	% Trapezoidal: lspb(start, goal, time, V) q_0 = 0; % initial pose q_f = 10; % final pose t_f = 18; % seconds if 1 % Specify max velocity: V = 0.6; % Max velocity lspb(q_0, q_f, 0:0.1:t_f, V) t_blend = (q_0 - q_f + V*t_f)/V else % Or specify blend time: t_blend = t_f/2; V = (q_0 - q_f) / (t_blend - t_f) lspb(q_0, q_f, 0:0.1:t_f, V) end (can do [s sd sdd] = lspb(...))
Bang-Bang		same as LSPB, except -t_blend = tf/2

Types:

Name	Image (Joint space / workspace)	Explanation
Axis to Axis		One link at a time
Simultaneous movement on axes		Both links at max speed
coordinated path		Both links move such that motion finishes at the same time. (one link is at max speed, others are slowed to match)
continuous straight path		Linear workspace movement -e.g. moveL

8: Path Planning

Saturday, 29 July 2023 4:03 PM

Artificial Potential Field	$$F_{\mathrm{att},i}(q) = \begin{cases} -\zeta_i(o_i(q) - o_i(q_f)) & : \; \lVert o_i(q) - o_i(q_f) \rVert \le d \\ -d\zeta_i \dfrac{(o_i(q) - o_i(q_f))}{\lVert o_i(q) - o_i(q_f) \rVert} & : \; \lVert o_i(q) - o_i(q_f) \rVert > d \end{cases}$$ - (use parabolic well at short distances, conic well at large distances) $$F_{rep,i}(q) = \eta_i \left(\frac{1}{\rho(o_i(q))} - \frac{1}{\rho_0} \right) \frac{1}{\rho^2(o_i(q))} \nabla \rho(o_i(q))$$ - only for $\rho\left(o_i(q)\right) \le \rho_0$ - else $= 0$ $$U_{\mathrm{rep},i}(q) = \begin{cases} \frac{1}{2}\eta_i \left(\frac{1}{\rho(o_i(q))} - \frac{1}{\rho_0} \right)^2 & ; \; \rho(o_i(q)) \le \rho_0 \\ 0 & ; \; \rho(o_i(q)) > \rho_0 \end{cases}$$ • η_i is a scaling factor and $\rho\left(o_i(q)\right) = \text{distance to obstacle}$, $\Delta\rho\left(o_i(q)\right) = \frac{o_i - o_{obst}}{\lVert o_i - o_{obst} \rVert}$	see quiz 2 prep
Bug 2		bug = Bug2(config_space_binary'); % Note use of the transposed cost map, due to Matlab's ordering of indices % in the cost map array. % Generate the plan for every free point to the goal goal = [goal_cell_i, goal_cell_j]; start= [start_cell_i, start_cell_j]; % Generate the path bug_path=bug.query(start,goal); axis xy bug.plot hold on plot(bug_path(:, 1), bug_path(:, 2),'LineWidth', 4, 'Color', 'g'); see lecture 8 examples
D*		ds = Dstar(config_space_binary'); % Note use of the transposed cost map, due to Matlab's ordering of indices % in the cost map array. % Generate the plan for every free point to the goal ds.plan(goal_cell); % Generate the path ds_path = ds.query(start_cell); % Plot the costmap and the path in joint space c = ds.costmap(); f2 = figure(2) %set(gcf, 'OuterPosition',[100 100 scrsz(3)/2-100 scrsz(4)/2]); imagesc(c) axis xy %ds.plot hold on plot(ds_path(:, 1), ds_path(:, 2)); ds.modify_cost([50,55; 80,99], 100); ds.plan(); ds_path1 = ds.query(start_cell); f3 = figure(3) %set(gcf, 'OuterPosition',[100 100 scrsz(3)/2-100 scrsz(4)/2]); imagesc(c) axis xy %ds.plot hold on plot(ds_path1(:, 1), ds_path1(:, 2));
Probabilistic Road Map (PRM)		% convert start and goal position to cell positions prm = PRM(config_space_binary', 'npoints', 200); %%% number of sampling points prm.plan(); prm_path = prm.query(start_cell, goal_cell); prm.plot(); hold on plot(prm_path(:, 1), prm_path(:, 2), 'LineWidth', 4, 'Color', 'g'); % Convert path back to joint angles (from cells) prm_path_angles = zeros(size(prm_path)); for i = 1:length(prm_path) angle_1 = cell2angle(prm_path(i, 1), min1, max1, n1); angle_2 = cell2angle(prm_path(i, 2), min2, max2, n2); prm_path_angles(i, :) = [angle_1, angle_2]; end

Control with Artificial Potential Fields: (converting force to torque)

- τ: Vector of joint torques | $\tau = J_v^T F$ | F: workspace force at end effector
- $\boldsymbol{\tau}(q) = \sum_i J_{O_i}^T * \boldsymbol{F}_i(q)$
- F = total artificial force (attraction + repulsion)
  - different F and J for each joint
- NOTE: the jacobians are transposed (truncate then transpose)
  - F is a column vector for each force
  - this causes the final torque vector to be a column vector of the torques of the joints

e.g.

```
% jacobian calculation:
startup_rvc;
L(1) = Link([0 0 1 0]);
L(2) = Link([0 0 0 0]);
robot = SerialLink(L)
J1 = jacob0(robot, [0, 0])

L(1) = Link([0 0 1 0]);
L(2) = Link([0 0 1 0]);
robot = SerialLink(L)
J2 = jacob0(robot, [0 pi/2])

J_O1_transpose = J1(1:2,:)'
J_O2_transpose = J2(1:2,:)'
% Torque calculation:
tau = J_O1_transpose * F_total(:,1) + J_O2_transpose * F_total(:,2) % in the form [tau_q1; tau_q2]
% (counter clockwise positive)
```

9: Euler-Lagrange

Sunday, 30 July 2023 10:40 AM

1 Dimension:

The Lagrangian:

- $L = K - P$

Force:

We define the Lagrangian $\mathcal{L}$ as,

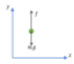
$$\mathcal{L} = K - P$$

$$L = K - P = \frac{1}{2}m\dot{y}^2 - mgy$$

Since,

$$\frac{\partial L}{\partial \dot{y}} = m\dot{y} = \frac{\partial K}{\partial \dot{y}}$$

$$\frac{\partial L}{\partial y} = -mg = \frac{\partial P}{\partial y}$$



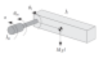
Force acting on the system is,

$$\boxed{f = m\ddot{y} + mg = \frac{d}{dt}\frac{\partial L}{\partial \dot{y}} - \frac{\partial L}{\partial y}}$$

$$f = \frac{d}{dt}\frac{\delta L}{\delta \dot{q}} - \frac{\delta L}{\delta q}$$

Similar for torques:

Let θ_l and θ_m denote the angles of the link and motor shaft, respectively. Then $\theta_m = r\theta_l$ where r : 1 is the gear ratio.



The kinetic energy of the system

$$K = \frac{1}{2}J_m\dot{\theta}_m^2 + \frac{1}{2}J_l\dot{\theta}_l^2 = \frac{1}{2}(r^2 J_m + J_l)\dot{\theta}_l^2$$

The potential energy

$$P = Mgl(1 - \cos\theta_l)$$

Defining $J = r^2 J_m + J_l$ The Lagrangian $\mathcal{L} = \frac{1}{2}J\dot{\theta}_l^2 - Mgl(1 - \cos\theta_l)$

3 Dimensions, with N links:

Kinetic Energy of a single link:

Kinetic energy of the i[th] link:

$$K_i = \frac{1}{2}(m_i \boldsymbol{v}_{ci}^T \boldsymbol{v}_{ci} + \boldsymbol{\omega}_i^T I_i \boldsymbol{\omega}_i)$$

Apply the Jacobian

$$\boldsymbol{v}_{ci} = J_{vci}(\boldsymbol{q})\,\dot{\boldsymbol{q}}$$

$$\boldsymbol{\omega}_i = R_i^T(\boldsymbol{q})J_{\omega i}(\boldsymbol{q})\dot{\boldsymbol{q}}$$

- J_{vci} – Jacobian matrix for velocity of the center of mass of link i
- $J_{\omega i}$ – Jacobian matrix for angular velocity of link i
- $R_i^T(\boldsymbol{q})$ – transformation matrix that transforms the angular velocity vector from the object frame to the inertial frame

- need to get the Jacobian matrix for each link's center of mass
- keep velocity and rotational velocity Jacobians separate

For n links:

$$K = \frac{1}{2}\sum_{i=1}^{n}(m_i \boldsymbol{v}_{ci}^T \boldsymbol{v}_{ci} + \boldsymbol{\omega}_i^T I_i \boldsymbol{\omega}_i)$$

$$K = \frac{1}{2}\dot{\boldsymbol{q}}^T \underbrace{\sum_{i=1}^{n}\left(m_i J_{vci}(\boldsymbol{q})^T J_{vci}(\boldsymbol{q}) + J_{\omega i}(\boldsymbol{q})^T R_i(\boldsymbol{q}) I_i R_i^T(\boldsymbol{q}) J_{\omega i}(\boldsymbol{q})\right)}_{D(\boldsymbol{q})\text{ – inertia matrix with } n \times n \text{ terms.}}\dot{\boldsymbol{q}}$$

$$\boxed{K = \frac{1}{2}\dot{\boldsymbol{q}}^T D(\boldsymbol{q})\dot{\boldsymbol{q}}}$$

- includes both linear and rotational kinetic energy
- later, the D matrix is described in terms of its elements as "d_{ij}"

Potential energy:

$$P(q) = \sum_{i=1}^{n} m_i \boldsymbol{g}^T \boldsymbol{r}_{ci}$$

Now to find euler-lagrange:

- $L = \frac{1}{2}\dot{\boldsymbol{q}}^T D(\boldsymbol{q})\dot{\boldsymbol{q}} \quad - P(\boldsymbol{q})$
- $\boxed{L = \frac{1}{2}\sum_{i,j}^{n} d_{ij}(\boldsymbol{q})\dot{q}_i\dot{q}_j - P(\boldsymbol{q})}$
- $\frac{\partial L}{\partial \dot{q}_k} = \sum_j d_{kj}(\boldsymbol{q})\dot{q}_j$
  - take derivative wrt $\dot{q}_i$ (k == i)
  - one of them would have been squared and so 1/2 is lost
- $\boxed{\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_k} = \sum_j d_{kj}(\boldsymbol{q})\ddot{q}_j + \sum_j \frac{\partial d_{kj}(\boldsymbol{q})}{\partial q_i}\dot{q}_i\dot{q}_j}$

  $k = 1, 2, \ldots\ldots, n$
  - since P doesn't depend on $\dot{q}$
- $\boxed{\frac{\partial L}{\partial q_k} = \frac{1}{2}\sum_{i,j}^{n} \frac{\partial d_{ij}(\boldsymbol{q})}{\partial q_k}\dot{q}_i\dot{q}_j - \frac{\partial P(\boldsymbol{q})}{\partial q_k}}$

hence,

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_k} - \frac{\partial L}{\partial q_k} = \tau_k$$

$$\boxed{\sum_j d_{kj}(\boldsymbol{q})\ddot{q}_j + \sum_{ij}\left\{\frac{\partial d_{kj}}{\partial q_i} - \frac{1}{2}\frac{\partial d_{ij}}{\partial q_k}\right\}\dot{q}_i\dot{q}_j + \frac{\partial P}{\partial q_k} = \tau_k}$$

This can be simplified by splitting it into three terms:

$$g_k = \frac{\partial P}{\partial q_k}$$

$$c_{ijk} = \frac{1}{2}\left\{\frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{ki}}{\partial q_j} - \frac{\partial d_{ij}}{\partial q_k}\right\}$$

$$\sum_j d_{kj}(\boldsymbol{q})\ddot{q}_j + \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ijk}(\boldsymbol{q})\dot{q}_i\dot{q}_j + g_k(q) = \tau_k;\ k = 1, 2, \ldots, n$$

$$\underbrace{M(\boldsymbol{q})\ddot{\boldsymbol{q}}}_{\text{Joint-space inertia matrix}} + \underbrace{C(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}}}_{\text{Coriolis and centripetal coupling matrix}} + \underbrace{g(\boldsymbol{q})}_{\text{Gravity loading}} = \boldsymbol{\tau};$$

- D(q) == M(q)

Summary:

$$g_k = \frac{\partial P}{\partial q_k}$$

$$c_{ijk} = \frac{1}{2}\left\{\frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{ki}}{\partial q_j} - \frac{\partial d_{ij}}{\partial q_k}\right\}$$

$$\sum_j d_{kj}(\boldsymbol{q})\ddot{q}_j + \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ijk}(\boldsymbol{q})\dot{q}_i\dot{q}_j + g_k(q) = \tau_k;\ k = 1, 2, \ldots, n$$

$$\underbrace{M(\boldsymbol{q})\ddot{\boldsymbol{q}}}_{\text{Joint-space inertia matrix}} + \underbrace{C(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}}}_{\text{Coriolis and centripetal coupling matrix}} + \underbrace{g(\boldsymbol{q})}_{\text{Gravity loading}} = \boldsymbol{\tau};$$

- D(q) == M(q)

$$K = \frac{1}{2}\dot{\boldsymbol{q}}^T \underbrace{\sum_{i=1}^{n}\left(m_i J_{vci}(\boldsymbol{q})^T J_{vci}(\boldsymbol{q}) + J_{\omega i}(\boldsymbol{q})^T R_i(\boldsymbol{q}) I_i R_i^T(\boldsymbol{q}) J_{\omega i}(\boldsymbol{q})\right)}_{D(\boldsymbol{q})\text{ – inertia matrix with } n \times n \text{ terms.}}\dot{\boldsymbol{q}}$$

$$\boxed{K = \frac{1}{2}\dot{\boldsymbol{q}}^T D(\boldsymbol{q})\dot{\boldsymbol{q}}}$$

where

$$P(q) = \sum_{i=1}^{n} m_i \boldsymbol{g}^T \boldsymbol{r}_{ci}$$

- equals the amount of work required to displace the center of mass of link i from the origin of base frame to position $\boldsymbol{r}_{ci}$
- i.e. $P(q_i) = m_i g h_i$

1D example summary:

- Step 1: Find the kinetic energy
- Step 2: Find potential energy
- Step 3: Obtain terms in the euler lagrange equation
- Step 4: sum together

n-dimension summary:

1. Get P(q) (1x1) (potential energy equation in terms of q)
   a. Get vectors to the center of gravity for each link, from frame {0}
   b. $P_i = m_i[0\ 0\ 9.81]\begin{bmatrix} o_{ci}(1) \\ o_{ci}(2) \\ o_{ci}(3) \end{bmatrix}$ or in matlab "P_i = m_i*g'*o_ci"
   c. OR you can just sum them up frame by frame (i.e. for o_{ci} w.r.t frame {i-1})
2. Get D matrix (nxn)
   a. translational component:
      i. find the linear velocity Jacobians of the center of mass of each link
   b. rotational component: (not required for prismatic)
      i. find the angular velocity Jacobian (the same for c.o.m / joint)
      ii. find the moment of inertia of each link, about the center of mass
         1) $I_i = I_{zz} = \frac{1}{12}m(a^2 + b^2)$ assuming other axes are negligible
3. Get c_{ijk} elements
   a. (C matrix is (nxnxn) but can't easily sum elements to get torque)
   b. calculate each element individually
4. Get g vector (nx1)
   a. $g_k(q) = \frac{\partial P}{\partial q_k}$
5. Sum everything together

- $$K = \frac{1}{2}\,\dot{\boldsymbol{q}}^T \underbrace{\sum_{i=1}^{n}\left(m_i J_{vci}(\boldsymbol{q})^T J_{vci}(\boldsymbol{q}) + J_{\omega i}(\boldsymbol{q})^T R_i(\boldsymbol{q}) I_i R_i^T(\boldsymbol{q}) J_{\omega i}(\boldsymbol{q})\right)}_{D(\boldsymbol{q}) \text{ – inertia matrix with } n \times n \text{ terms.}}\,\dot{\boldsymbol{q}}$$

$$\boxed{K = \frac{1}{2}\,\dot{\boldsymbol{q}}^T D(\boldsymbol{q})\dot{\boldsymbol{q}}}$$