Multijoint Torque Control

EE366/CS380: Introduction to Robotics

Dr. Basit Memon

Electrical and Computer Engineering
Habib University

April 24, 2024

Table of Contents

Multi-joint control

- We need to obtain a model for the entire robot manipulator - all joints.
- Motion of one link affects the other links. The model is complex.
- Can we just control each joint independently as we did with velocity commands?

Table of Contents

- Entire manipulator can be expressed by following rigid body dynamics:

$$J(q)\ddot{q} + C(q,\dot{q})\dot{q} + B_v\dot{q} + g(q) = \tau$$

- q represents joint variable - θ for revolute joint and d for prismatic
- τ is any torque acting on manipulator – it could be desired, i.e. actuator, or undesired disturbance.

Example: Model of two-link manipulator

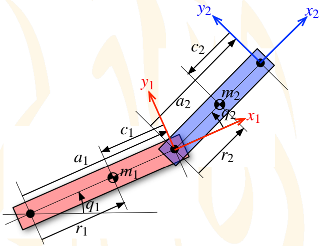


Figure: Two links planar arm

$$
\begin{aligned}
\tau_1 &= \boldsymbol{M}_{11}(q_2)\ddot{q}_1 + \underbrace{\boldsymbol{M}_{12}(q_2)\ddot{q}_2 + C_1(q_2)\dot{q}_1\dot{q}_2 + C_2(q_2)\dot{q}_2^2 + g(q_1, q_2)}_{\text{disturbance}} \\
\boldsymbol{M}_{11} &= m_1\left(a_1^2 + 2a_1c_1 + c_1^2\right) + m_2\left(a_1^2 + (a_2 + c_2)^2 + (2a_1a_2 + 2a_1c_2)\cos q_2\right) \\
\boldsymbol{M}_{12} &= m_2(a_2 + c_2)(a_2 + c_2 + a_1\cos q_2) \\
C_1 &= -2a_1m_2(a_2 + c_2)\sin q_2 \\
C_2 &= -a_1m_2(a_2 + c_2)\sin q_2 \\
g &= (a_1m_1 + a_1m_2 + c_1m_1)\cos q_1 + (a_2m_2 + c_2m_2)\cos(q_1 + q_2)
\end{aligned}
$$

Figure: Dynamics for net torque at joint 1

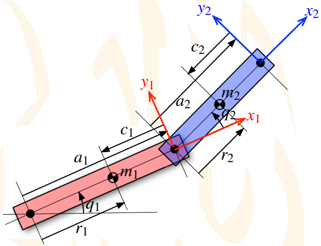


Figure: Two links planar arm

$$\tau_1 = \overset{J_l}{\boxed{M_{11}(q_2)}}\ddot{q}_1 + \underbrace{M_{12}(q_2)\ddot{q}_2 + \overset{\text{Coriolis}}{\boxed{C_1(q_2)\dot{q}_1\dot{q}_2 + C_2(q_2)\dot{q}_2^2}} + \overset{\text{Gravity}}{\boxed{g(q_1, q_2)}}}_{\text{disturbance}}$$

$$M_{11} = m_1\left(a_1^2 + 2a_1c_1 + c_1^2\right) + m_2\left(a_1^2 + (a_2 + c_2)^2 + (2a_1a_2 + 2a_1c_2)\cos q_2\right)$$

$$M_{12} = m_2(a_2 + c_2)(a_2 + c_2 + a_1\cos q_2)$$

$$C_1 = -2a_1m_2(a_2 + c_2)\sin q_2$$

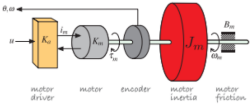
$$C_2 = -a_1m_2(a_2 + c_2)\sin q_2$$

$$g = (a_1m_1 + a_1m_2 + c_1m_1)\cos q_1 + (a_2m_2 + c_2m_2)\cos(q_1 + q_2)$$

Figure: Dynamics for net torque at joint 1

Joint Space – Model

$$\begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_n \end{bmatrix}$$

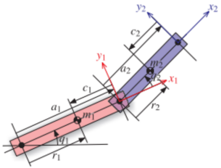


Motor

$$J_m \ddot{q}_m = \frac{K_m V_a}{R_a} - \frac{K_m^2 \dot{q}_m}{R_a} - B_m \dot{q}_m - \tau_l$$

Gear Assembly

$$\tau_j = K_r \tau_m$$

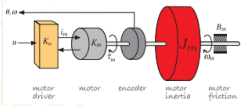


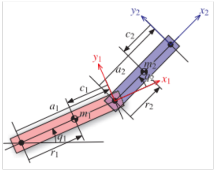
Robot Arm

$$\tau = J(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q)$$

$$\begin{bmatrix} q_1(t) \\ q_2(t) \\ \vdots \\ q_n(t) \end{bmatrix}, \begin{bmatrix} \dot{q}_1(t) \\ \dot{q}_2(t) \\ \vdots \\ \dot{q}_n(t) \end{bmatrix}, \begin{bmatrix} \ddot{q}_1(t) \\ \ddot{q}_2(t) \\ \vdots \\ \ddot{q}_n(t) \end{bmatrix}$$

Torque balancing equation could be written on arm side





$$\begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_n \end{bmatrix}$$

Motor

$$J_m \ddot{q}_m = \frac{K_m V_a}{R_a} - \frac{K_m^2 \dot{q}_m}{R_a} - B_m \dot{q}_m - \tau_l$$

Gear Assembly

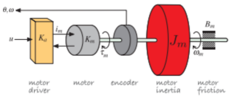
$$\tau_j = K_r \tau_m$$

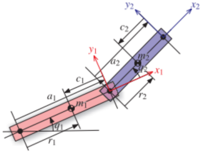
Robot Arm

$$\tau = J(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q)$$

$$\begin{bmatrix} q_1(t) \\ q_2(t) \\ \vdots \\ q_n(t) \end{bmatrix}, \begin{bmatrix} \dot{q}_1(t) \\ \dot{q}_2(t) \\ \vdots \\ \dot{q}_n(t) \end{bmatrix}, \begin{bmatrix} \ddot{q}_1(t) \\ \ddot{q}_2(t) \\ \vdots \\ \ddot{q}_n(t) \end{bmatrix}$$

Figure: Motor Reflected on the arm side





$$\begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_n \end{bmatrix}$$

Motor

$$J_m \ddot{q}_m = \frac{K_m V_a}{R_a} - \frac{K_m^2 \dot{q}_m}{R_a} - B_m \dot{q}_m - \tau_l$$

Robot Arm

$$\tau = J(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q)$$

Gear Assembly

$$\tau_j = K_r \tau_m$$

$$\begin{bmatrix} q_1(t) \\ q_2(t) \\ \vdots \\ q_n(t) \end{bmatrix}, \begin{bmatrix} \dot{q}_1(t) \\ \dot{q}_2(t) \\ \vdots \\ \dot{q}_n(t) \end{bmatrix}, \begin{bmatrix} \ddot{q}_1(t) \\ \ddot{q}_2(t) \\ \vdots \\ \ddot{q}_n(t) \end{bmatrix}$$

Figure: Arm Reflected on the motor side

On the motor side,

$$
\begin{aligned}
J_m \ddot{q}_m &= \frac{K_m V_a}{R_a} - \left(\frac{K_m^2}{R_a} + B_m\right)\dot{q}_m - \tau_l \\
&= \frac{K_m V_a}{R_a} - \left(\frac{K_m^2}{R_a} + B_m\right)\dot{q}_m - \frac{J(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q)}{K_r}
\end{aligned}
$$

Using the gear ratio, $q = \frac{q_m}{K_r}$:

$$
\left(J_m + \frac{J(q)}{K_r^2}\right)\ddot{q}_m = \frac{K_m V_a}{R_a} - \left(\frac{K_m^2}{R_a} + B_m + \frac{C(q,\dot{q})}{K_r^2}\right)\dot{q}_m - \frac{g(q)}{K_r}
$$

Manipulator is modeled by coupled non-linear ODE :(

$$J(q) = \begin{bmatrix} m_1 a_1^2 + m_2(a_1^2 + 2a_1 a_2 \cos q_2 + a_2^2) & m_2(a_1 a_2 \cos q_2 + a_2^2) \\ m_2(a_1 a_2 \cos q_2 + a_2^2) & m_2 a_2^2 \end{bmatrix}$$

$$C(q, \dot{q}) = \begin{bmatrix} -2m_2 a_1 a_2 \sin q_2 \, \dot{q}_2 & \dot{q}_2 \\ m_2 a_1 a_2 \dot{q}_1 \sin q_2 & 0 \end{bmatrix}$$

$$g(q) = \begin{bmatrix} (m_1 + m_2) a_1 g \cos q_1 + m_2 g a_2 \cos(q_1 + q_2) \\ m_2 g a_2 \cos(q_1 + q_2) \end{bmatrix}$$

- Previously J was constant matrix. Now all of these are dependent on q. A nonlinear model completely!
- Can we control each motor/joint separately?

Table of Contents

Is independent joint control possible?

- Doesn't seem so since the J matrix is coupled.
- Unless, we can decouple it.
- Note that this will be an approximate model!
- Might as well try for linear approximate model.

Large gear ratio results in diagonal decoupled J

- If the gear ratio is large, $\frac{J(q)}{K_r^2}$ can be approximated by a constant diagonal matrix.

$$\frac{J(q)}{K_r^2} = \frac{1}{K_r^2}\begin{bmatrix} m_1 a_1^2 + m_2(a_1^2 + 2a_1 a_2 \cos q_2 + a_2^2) & m_2(a_1 a_2 \cos q_2 + a_2^2) \\ m_2(a_1 a_2 \cos q_2 + a_2^2) & m_2 a_2^2 \end{bmatrix}$$

Terms are dependent on q, but choose maximum inertia possible over all q.

- Now we can treat each joint independently – Decentralized control.

$$\frac{J(q)}{K_r^2} \approx \begin{bmatrix} I_1 & 0 \\ 0 & I_2 \end{bmatrix}$$

Nonlinear terms treated as disturbances, given large gear ratio.

- If the gear ratio is large and $\dot{q}$ is small, $\frac{C(q,\dot{q})\dot{q}_m}{K_r^2}$ is also very small.

$$\frac{C(q,\dot{q})\dot{q}_m}{K_r^2} = \frac{1}{K_r}\begin{bmatrix} -2m_2 a_1 a_2 \sin q_2\, \dot{q}_2 & \dot{q}_2 \\ m_2 a_1 a_2 \dot{q}_1 \sin q_2 & 0 \end{bmatrix} \dot{q}_m$$

- We can model this and gravity term as disturbance for control problem:

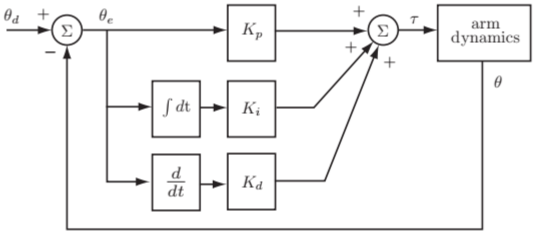
$$\left(J_m + \frac{J(q)}{K_r^2}\right)\ddot{q}_m = \frac{K_m V_a}{R_a} - \left(\frac{K_m^2}{R_a} + B_m + \frac{C(q,\dot{q})}{K_r^2}\right)\dot{q}_m - \frac{g(q)}{K_r}$$

Approximately, and only approximately

$$J_s \ddot{q}_m = u - B_s \dot{q}_m - D$$

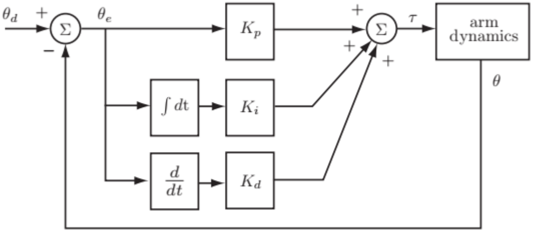
- In order to highlight the effect of K_r, the inverse is displayed as a reciprocal but don't be misled. K_r is a matrix in general and its inverse would be taken.

PID reduces error to zero for step trajectory and disturbances.



- It was shown that PID control reduces error to zero, if desired trajectory is ramp or step function.
- It was shown that PID control reduces effects of disturbance to zero, if disturbance is step function.

What is the equation for arm dynamics here?



$$J_s \ddot{q}_m = u - B_s \dot{q}_m - D$$

OR

$$\left(J_m + \frac{J(q)}{K_r^2}\right)\ddot{q}_m = u - \left(B_s + \frac{C(q,\dot{q})}{K_r^2}\right)\dot{q}_m - \frac{g(q)}{K_r}?$$

PID designed using approx. model, but reality doesn't change!

- When I'm designing:

$$J_s \ddot{q}_m = u - B_s \dot{q}_m - D$$

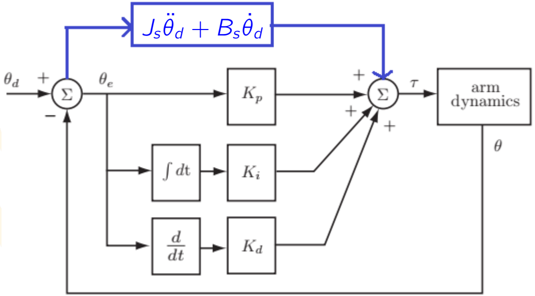
- $u = K_p \theta_e + K_i \int_0^t \theta_e dt + K_d \dot{\theta}_e$
- Find K_p, K_i, and K_d to minimize error and converge fast.

- Install sensors to measure θ and $\dot{\theta}$
- Program a controller to output $K_p \theta_e + K_i \int_0^t \theta_e dt + K_d \dot{\theta}_e$ to the motor and arm.
- This is equivalent to:

$$\left(J_m + \frac{J(q)}{K_r^2} \right) \ddot{q}_m =$$

$$K_p \theta_e + K_i \int_0^t \theta_e dt + K_d \dot{\theta}_e -$$

$$\left(B_s + \frac{C(q, \dot{q})}{K_r^2} \right) \dot{q}_m - \frac{g(q)}{K_r}$$



- The feedforward function is determined using approximate system model.
- The feedback controller can be used to reject disturbance (PD or PID).

Table of Contents

Strategies – Centralized Control

- When $J(q)$ cannot be made a diagonal matrix (e.g. motors are directly connected to links without gears), we'll have to design controller for all joints together.
- Coupling of dynamics cannot be avoided:

$$J(q)\ddot{q} = \tau$$

Still ignoring the other terms.

- We have a multi-input, multi-output problem.
- While PID controller is still a good controller for this problem, the approach discussed for designing PID is no longer suitable, and one typically uses state-space methods.

- Dynamics of an arm in all glory:

$$J(q)\ddot{q} + C(q,\dot{q})\dot{q} + B\dot{q} + g(q) = u.$$

- By using gear mechanisms, we can decouple inertia matrix and make it independent of q. By assuming that $\dot{q}$ and $\ddot{q}$ are small, we can ignore $C(q,\dot{q})$ term. But, we still have to treat $g(q)$ as disturbance.

- Find K_p and K_d for $u = K_p\, q_e + K_d\, \dot{q}_e$, assuming that system is modeled by $J_s\ddot{q}_m = u - B_s\dot{q}_m - D$

- Implementation:
  - Measure q and $\dot{q}$.
  - We know form of $g(q)$. Substitute q to find $g(q)$.
  - Let $u = K_p\, q_e + K_d\, \dot{q}_e + \frac{g(q)}{K_r}$

$u = K_P\, q_e + K_D \dot{q}_e + \frac{g(q)}{K_r}$

- Complete system dynamics are:

$$\left(J_m + \frac{J(q)}{K_r^2}\right)\ddot{q}_m = K_P(\theta_d - \theta) + K_D(\dot{\theta}_d - \dot{\theta}) + \frac{g(q)}{K_r} - \left(B_s + \frac{C(q,\dot{q})}{K_r^2}\right)\dot{q}_m - \frac{g(q)}{K_r}$$

- The dynamics used for design are now closer to actual dynamics

- Why stop at gravity term? Let's get rid of all annoying elements.
- $u =$ Controller for achieving desired trajectory $+$ Inverses of annoying terms in dynamics
- System dynamics are:

$$\left(J_m + \frac{J(q)}{K_r^2}\right)\ddot{q}_m = u - \left(B_s + \frac{C(q,\dot{q})}{K_r^2}\right)\dot{q}_m - \frac{g(q)}{K_r} + d$$

- Let $u =$ Controller for achieving desired trajectory $+ \left(B_s + \frac{C(q,\dot{q})}{K_r^2}\right)\dot{q}_m + \frac{g(q)}{K_r}$

Inverse Dynamics change dynamics exactly to linear system.

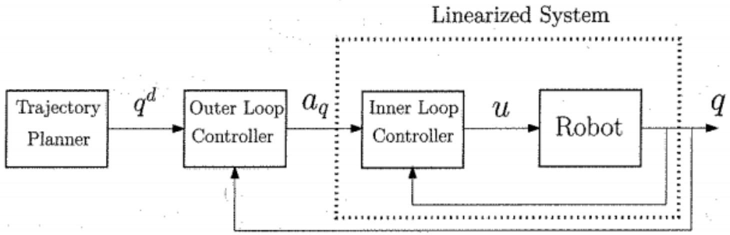
$$\left(J_m + \frac{J(q)}{K_r^2}\right)\ddot{q}_m = \text{Trajectory Controller} + \left(B_s + \frac{C(q,\dot{q})}{K_r^2}\right)\dot{q}_m + \frac{g(q)}{K_r} - \left(B_s + \frac{C(q,\dot{q})}{K_r^2}\right)\dot{q}_m - \frac{g(q)}{K_r} + d$$

- Can we get rid of inertia term too?
- Let $u = \left(J_m + \frac{J(q)}{K_r^2}\right) a_q + \left(B_s + \frac{C(q,\dot{q})}{K_r^2}\right)\dot{q}_m + \frac{g(q)}{K_r}$, where a_q is controller to achieve desired trajectory.
- System dynamics reduce to: $\ddot{q}_m = a_q$. System is now a linear decoupled one. This is exact (not an approximation).
- **Inverse Dynamics or Computed Torque Control**.

- Complete system dynamics are:

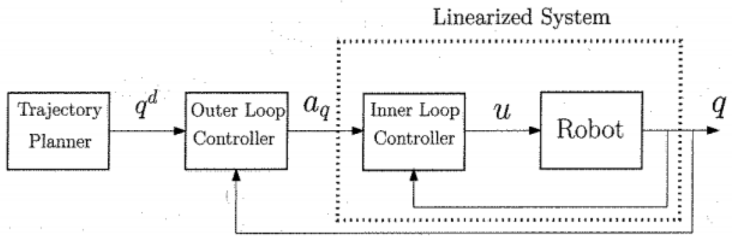
$$\left(J_m + \frac{J(q)}{K_r^2}\right)\ddot{q}_m = \left(J_m + \frac{J(q)}{K_r^2}\right) a_q + \left(B_s + \frac{C(q,\dot{q})}{K_r^2}\right)\dot{q}_m + \frac{g(q)}{K_r} - \left(B_s + \frac{C(q,\dot{q})}{K_r^2}\right)\dot{q}_m - \frac{g(q)}{K_r}$$

- **New Objective:** Find a_q, so that $\ddot{q}_m = a_q$ gives $q_m = q_d$.



What should be controller a_q?

- What if $a_q = \ddot{q}_d$?
- Let's be safe, and add feedback in case there are errors in the model or if there is a disturbance.
- Say $a_q = \ddot{q}_d(t) + K_p(q_d - q + K_d(\dot{q}_d - \dot{q})$.
- Since $\ddot{q}_m = a_q$, we have
$$\ddot{q}_e + K_d \dot{q}_e + K_p q_e = 0,$$
where $q_e = q_d - q$.
- We know that, we can choose K_p and K_d to make any error converge to zero at a suitable rate.



$$u = \left(J_m + \frac{J(q)}{K_r^2}\right)[\ddot{q}_d(t) + K_p(q_d - q) + K_d(\dot{q}_d - \dot{q})] + \left(B_s + \frac{C(q,\dot{q})}{K_r^2}\right)\dot{q}_m + \frac{g(q)}{K_r}$$

- Measure q and $\dot{q}$.
- Substitute q and $\dot{q}$ to find $g(q)$, $J(q)$, and $C(q,\dot{q})$.
- Substitute in expression for u

Inverse Dynamics requires parameters, but there is uncertainty

- The dynamics of an arm can in general be expressed as:

$$J(q)\ddot{q} + C(q,\dot{q})\dot{q} + B\dot{q} + g(q) = u.$$

- Let $u = \boxed{\hat{J}}(q)\, a_q + \boxed{\hat{C}}(q,\dot{q})\dot{q} + \boxed{\hat{B}}\dot{q} + \boxed{\hat{g}}(q)$
- Notice the hats on J, C, B, and g. These are our estimates for these quantities. What do we get if we now substitute u?

$$\ddot{q} = a_q + \eta(q,\dot{q},a_q),$$

where

$$\eta = J^{-1}\left(\tilde{J}a_q + \tilde{C}\dot{q} + \tilde{B}\dot{q} + \tilde{g}\right)$$

and

$$\tilde{(\cdot)} = (\cdot) - \hat{(\cdot)}.$$

Robust Control ensures error converges for bounded uncertainties

- Will our original $a_q = \ddot{q}_d + K_p(q_d - q) + K_d(\dot{q}_d - \dot{q})$ still work?
- Perhaps. But we would like some guarantees.

If we're certain that

$$||\eta|| \leq \rho(\tilde{q}, \dot{\tilde{q}}, t)$$

then we can design a controller

$$a_q = \ddot{q}_d - K_P(q - q_d) - K_D(\dot{q} - \dot{q}_d) + \delta a,$$

which guarantees global convergence of tracking error.

$$\delta a = \begin{cases} -\rho \frac{B^T Pe}{||B^T Pe||} & \text{if } ||B^T Pe|| \neq 0 \\ 0 & \text{if } ||B^T Pe|| = 0 \end{cases},$$

$$e = (\tilde{q}^T, \dot{\tilde{q}}^T)^T,$$

$$B = (0, I_n)^T,$$

$$Q = PA + A^T P,$$

$$A = \begin{bmatrix} 0 & I_n \\ -K_P & -K_D \end{bmatrix}$$

and $Q > 0$ and symmetric,

$$\rho = \frac{1}{1-\alpha}\left[\alpha\beta + ||K||||e|| + \lambda_H \phi(e,t)\right],$$

$$1 > \alpha \geq ||J^{-1}\hat{J} - I_n||$$

$$\beta > \sup_{t \in [0,\infty)} ||\ddot{q}_d(t)||$$

$$K = (K_P, K_D)$$

$$\lambda_H \geq ||J^{-1}||$$

$$\phi(e,t) \geq ||\tilde{C}\dot{q} + \tilde{B}\dot{q} + \tilde{g}||$$

- Adaptive controllers estimate controller parameters online and are adapt at dealing with imperfect knowledge of dynamical parameters.

- Recall the dynamics of an arm can be expressed as:

$$J(q)\ddot{q} + C(q,\dot{q})\dot{q} + B\dot{q} + g(q) = u.$$

- Let

$$\begin{aligned} u &= \boxed{\hat{J}}(q)\boxed{a_q} + \boxed{\hat{C}}(q,\dot{q})\dot{q} + \boxed{\hat{B}}\dot{q} + \boxed{\hat{g}}(q) \\ &= \hat{J}(q)\boxed{[\ddot{q}_d - K_P(q - q_d) - K_D(\dot{q} - \dot{q}_d)]} + \hat{C}(q,\dot{q})\dot{q} + \hat{B}\dot{q} + \hat{g}(q) \end{aligned}$$

- Adaptive control assumes that $\hat{J}$, $\hat{C}$, $\hat{B}$, $\hat{g}$ are not constants but time varying quantities now, which we'll update at each iteration.

- The entire idea of Adaptive control hinges on linear parameterization property of adaptive dynamics, according to which

$$J(q)\ddot{q} + C(q,\dot{q})\dot{q} + B\dot{q} + g(q) = Y(q,\dot{q},\ddot{q})\,a,$$

where Y, called regressor is a known matrix, and a is a vector of parameters.

  - Similarly,

$$\hat{J}(q)\ddot{q} + \hat{C}(q,\dot{q})\dot{q} + \hat{B}\dot{q} + \hat{g}(q) = Y(q,\dot{q},\ddot{q})\,\hat{a}$$

- How does this help us?

$$\begin{aligned} u &= \hat{J}(q)\left[\ddot{q}_d - K_P(q - q_d) - K_D(\dot{q} - \dot{q}_d)\right] + \hat{C}(q, \dot{q})\dot{q} + \hat{B}\dot{q} + \hat{g}(q) \\ &= \hat{J}(q)\left[\ddot{q}_d - K_P(q - q_d) - K_D(\dot{q} - \dot{q}_d)\right] + Y(q, \dot{q}, \ddot{q})\hat{a} - \hat{J}(q)\ddot{q} \\ &= \hat{J}(q)\left[(\ddot{q}_d - \ddot{q}) + K_P(q_d - q) + K_D(\dot{q}_d - \dot{q})\right] + Y(q, \dot{q}, \ddot{q})\hat{a} \end{aligned}$$

- We can find an update equation of the form:

$$\dot{\hat{a}} = -\Gamma^{-1} Y^T \hat{J}^{-1} B^T P (e_q^T, \dot{e}_q^T)^T$$

- We know that,

$$\begin{aligned}\dot{X} &= J(q)\dot{q}\\ \ddot{X} &= J(q)\ddot{q} + \dot{J}(q)\dot{q}\\ \tau &= J^T(q)F\end{aligned}$$

  where J is analytical Jacobian, X is vector of end-effector positions and orientations, τ is vector of joint torques, and F is vector of end-effector forces and torques.

- Substituting these into $M(q)\ddot{q} + C(q,\dot{q})\dot{q} + B\dot{q} + g(q) = u$, we can show that

$$M(q)\left[J^{-1}\ddot{X} - J^{-1}\dot{J}J^{-1}\dot{X}\right] + C(q,\dot{q})J^{-1}\dot{X} + BJ^{-1}\dot{X} + g(q) = J^T(q)F.$$

Task Space or Operational Space Control

$$\Lambda(q)\,\ddot{X} + \Gamma(q,\dot{q})\,\dot{X} + \eta(q) = F,$$

where

$$\begin{aligned}
\Lambda(q) &= J^{-T}(q)M(q)J^{-1}(q)\\
\Gamma(q,\dot{q}) &= J^{-T}(q)C(q,\dot{q})J^{-1}(q) + J^{-T}(q)BJ^{-1}(q) - \Lambda(q)\dot{J}(q)J^{-1}(q)\\
\eta(q) &= J^{-T}(q)g(q)
\end{aligned}$$

- Same controllers can be designed in task space using these dynamics.
- The end-effector position is seldom directly measured. So, even if we're working in task space we may have to use joint sensors and kinematics equations for X to be fed into control algorithm.