Independent Joint Control

- Treat each joint as an independent system and design separate controllers

Employed in most industrial robots

- Coupling effects are treated as disturbances

- performs satisfactorily if motions are not fast and gear ratios are large

  (large gear reduction → less effect of disturbance)

For a n-link robot,

Let $q_k = \frac{\theta_{m_k}}{r_k}$ $\quad k = 1, \ldots, n$

(θ_{m_k}: motor angle; r_k: gear ratio)

For joint k,

$$\sum_{j=1}^{n} d_{kj}\ddot{q}_j + \sum_{i,j=1}^{n} c_{ijk}(q)\dot{q}_i\dot{q}_j + g_k(q) = \tau$$

(d_{kj}: from the inertia matrix; $c_{ijk}(q)\dot{q}_i\dot{q}_j$: from the centrifugal and coriolis matrix; $g_k(q)$: from the gravity vector; left side: robot dynamics)

and

$$J_{m_k}\ddot{\theta}_{m_k} + \left(B_{m_k} + \frac{K_{bk}K_{mk}}{R_k}\right)\dot{\theta}_{m_k} = \frac{K_{mk}}{R_k}v_k - \frac{\tau_k}{r_k}$$

(J_{m_k}: motor inertia; B_{m_k}: motor damping; $\frac{K_{bk}K_{mk}}{R_k}$: actuator dynamics; K_{mk}: motor constants; v_k: applied voltage)

Combining,

$$\left(J_{m_k} + \frac{1}{r_k^2} d_{kk}(q) \right) \ddot{\theta}_{m_k} + \left(B_{m_k} + \frac{K_{bk}\, K_{mk}}{R_k} \right) \dot{\theta}_{m_k} = \frac{K_{mk}}{R_k} V_k - d_k$$

J_{eff_k} (effective inertia)

B_{eff_k} effective damping

u_k

$$d_k = \frac{1}{r_k} \left(\sum_{j \neq k} d_{kj} \ddot{q}_j + \sum_{i,j} c_{ijk} \dot{q}_i \dot{q}_j + g_k(q) \right)$$

or, $J_{eff} \ddot{\theta}_{m_k} + B_{eff_k} \dot{\theta}_m = u_k - d_k$

↳ similar to single DOF robot!

PD compensator : set point tracking

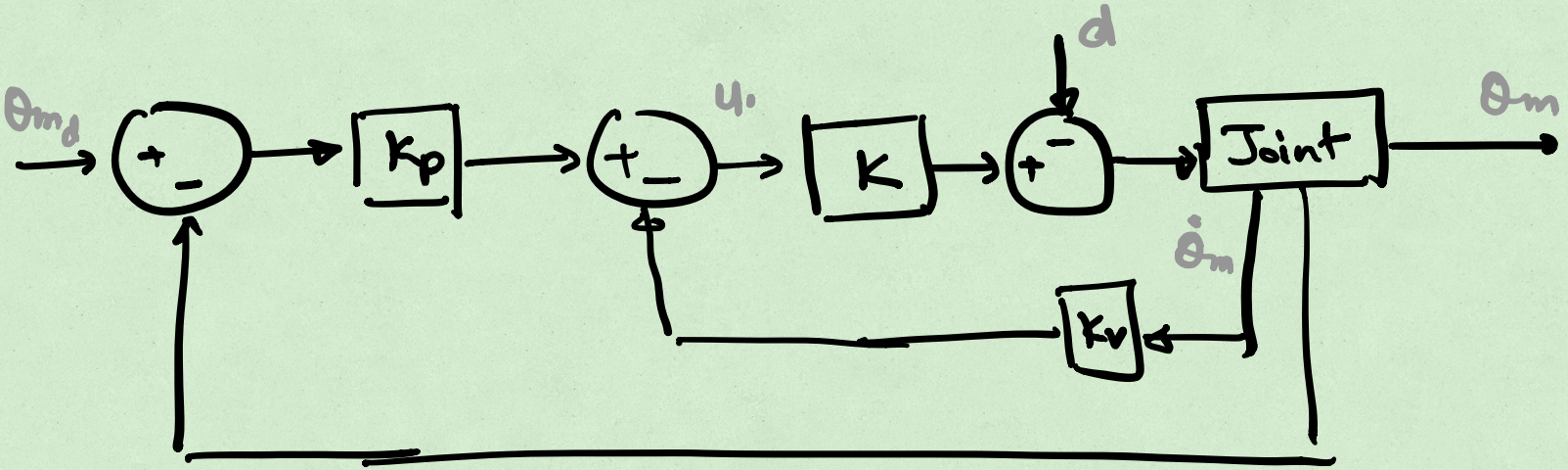
System : $J_{eff_k} \ddot{\theta}_{m_k} + B_{eff_k} \dot{\theta}_{m_k} = u_k - d_k$

Note : J_{eff} is a function of q, but large r_k reduces the influence and typically a constant average value is used

- Design PD compensator as earlier,

$$u_k = K_p(\theta_{m_{dk}} - \theta_{m_k}) - K_v \dot{\theta}_{m_k}$$

- implementation is simple and requires no information from other joints



Closed loop system:

$$\ddot{\theta}_{m_k} + \frac{B_{eff} + K K_v}{J_{eff}} \dot{\theta}_{m_k} + \frac{K K_p}{J_{eff}} (\theta_{m_k} - \theta_{m_d}) = -d_k$$

ω^2

$2\xi\omega$

steady state error, $e_{ss} = \frac{d_k}{K K_p}$ { use integral action to eliminate

choose $\xi = 1$, ω

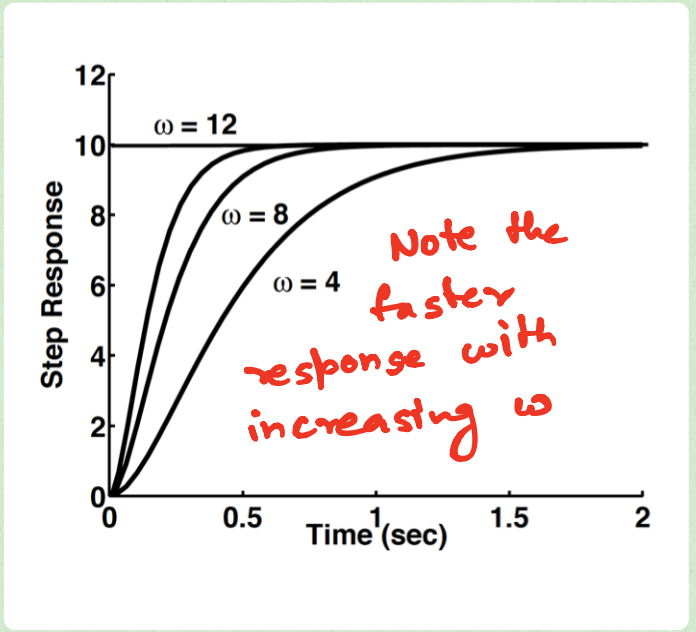
note if $K = J_{eff}$ $K_p = K_p'$

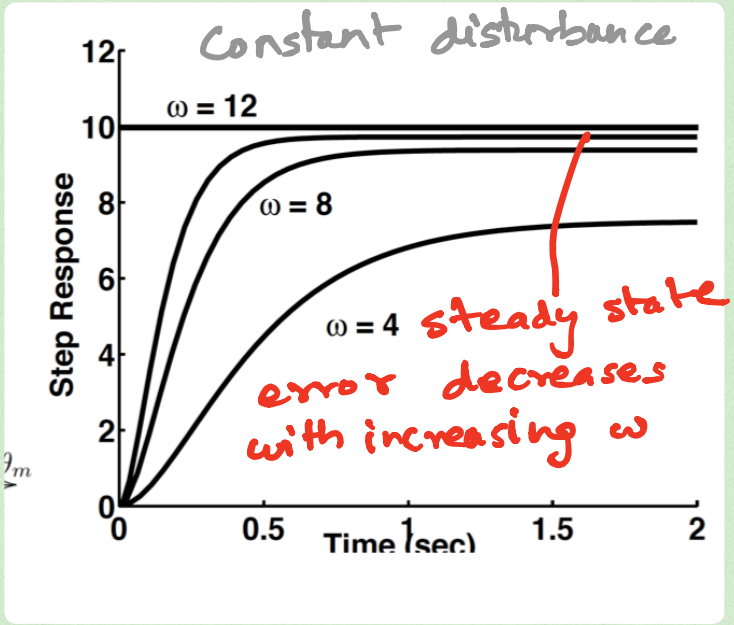
$$K_p = \frac{\omega^2 J_{eff}}{K} \qquad K_v = \frac{2\omega J_{eff} - B_{eff}}{K}$$

Second order system example

System: $\ddot{x} + \dot{x} = F$

Natural Frequency (ω)	Proportional Gain K_P	Derivative Gain K_D
4	16	7
8	64	15
12	144	23

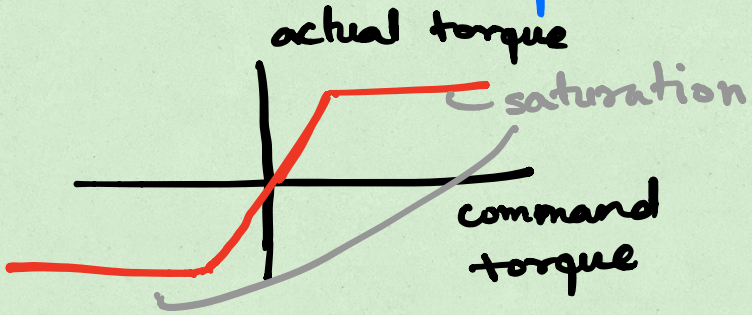




Actuator saturation

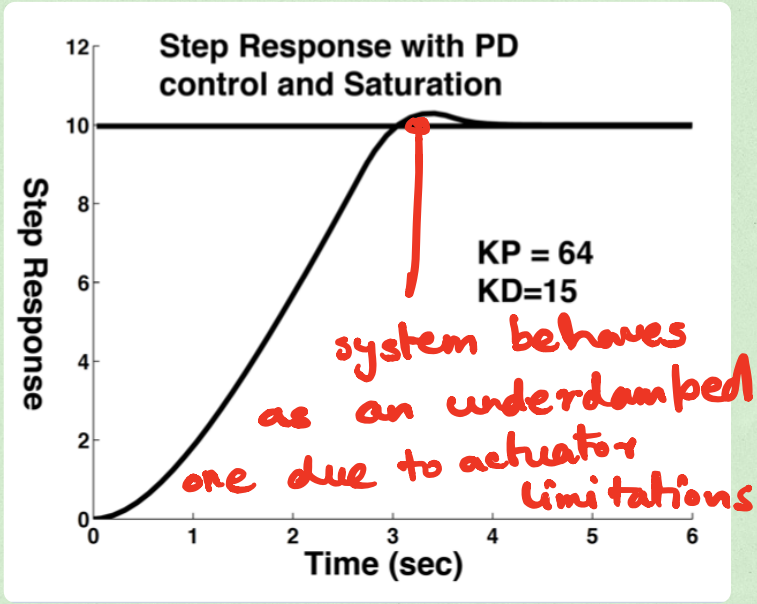
- Actuators have a peak torque generation capability

- If commanded torque > peak actuator torque, there is actuator saturation and only the peak actuator torque acts on the robot.



- Nonlinear phenomenon, not easy to handle

- degrades performance



Trajectory tracking

use a feedforward term like before.

System:

$$J_{eff}\ddot{\theta}_m + B_{eff}\dot{\theta}_m = KV - d$$

define desired trajectory for θ_m

$$V_K = \underbrace{\frac{J_{eff}\ddot{\theta}_{m_d} + B_{eff}\dot{\theta}_{m_d}}{K}}_{\text{feedforward term}} + K_p(\theta_{m_d} - \theta_m) + K_v(\dot{\theta}_{m_d} - \dot{\theta}_m)$$

tracking error, $e(t) = \theta_{m_d} - \theta_m$

closed-loop system:

$$J_{eff}\ddot{e} + (B_{eff} + KK_v)\dot{e} + KK_p e(t) = -d$$

Computed torque control

Disturbance rejection: use feedforward control based on expected value of disturbance d.

$$d_d = \frac{1}{r}\left(\sum d_{jk}(q_d)\,\ddot{q}_{jd} + \sum c_{ijk}(q_d)\,\dot{q}_{id}\,\dot{q}_{jd} + g_k(q_d) \right)$$

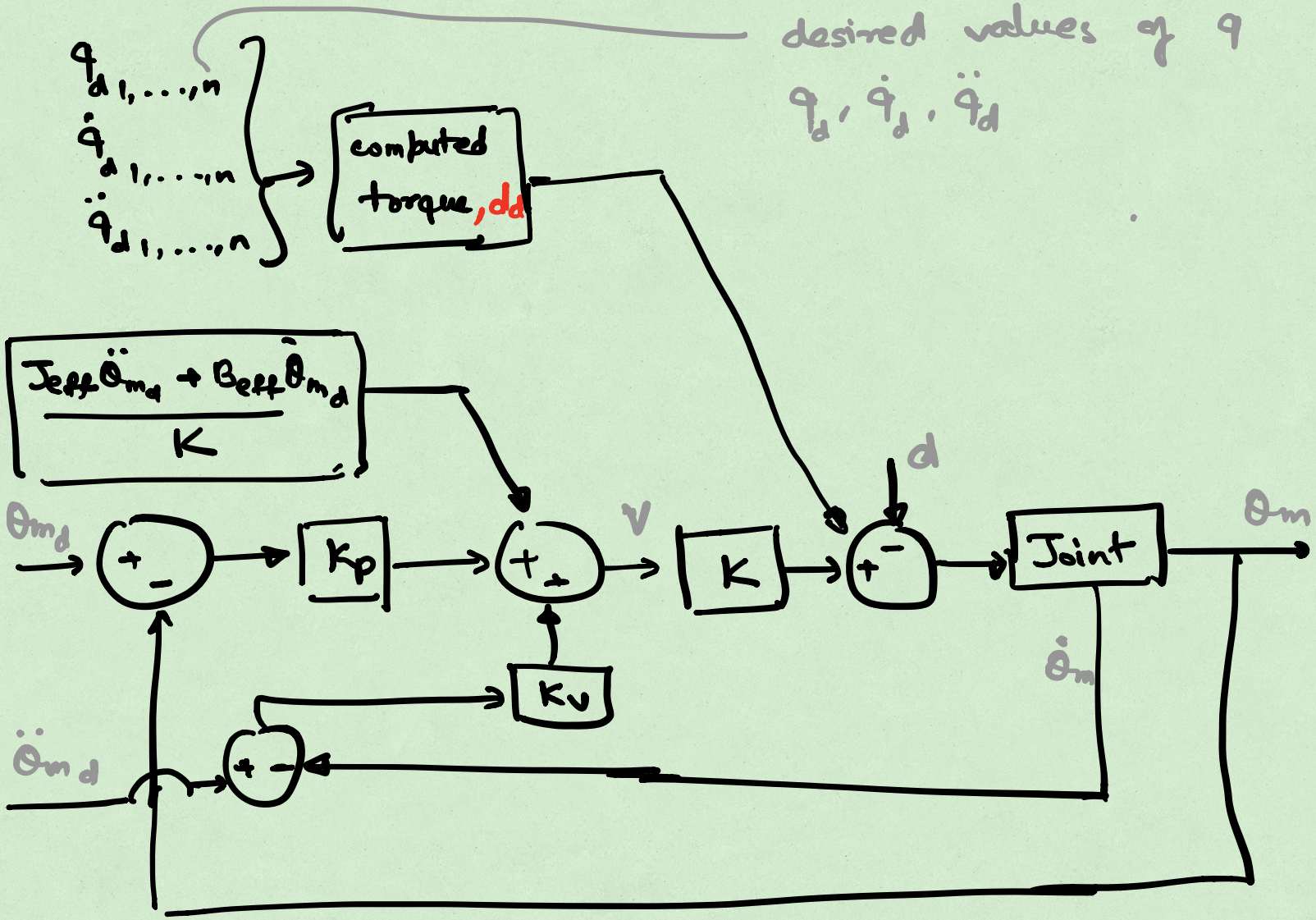
d_d: expected value of disturbance torque if desired trajectory is followed

Subscript 'd' indicates that we use desired values of states in this equation

$d - d_d = 0$, if tracking is perfect and there are no modeling or computational errors

- Terms can be pre-computed and stored offline

Computed torque control (continued...)



Performance : choice of ω

High K_p → better disturbance rejection

gains are limited by -

- unmodeled structural flexibilities (↳ coupling terms can be large)
- time-delays (associated w/ computations, actuations)
- sampling-rate (rate at which actuation command is updated)

$\omega \leq \frac{\omega_n}{2}$, ω_n : lowest frequency associated with the dynamic modes of the system

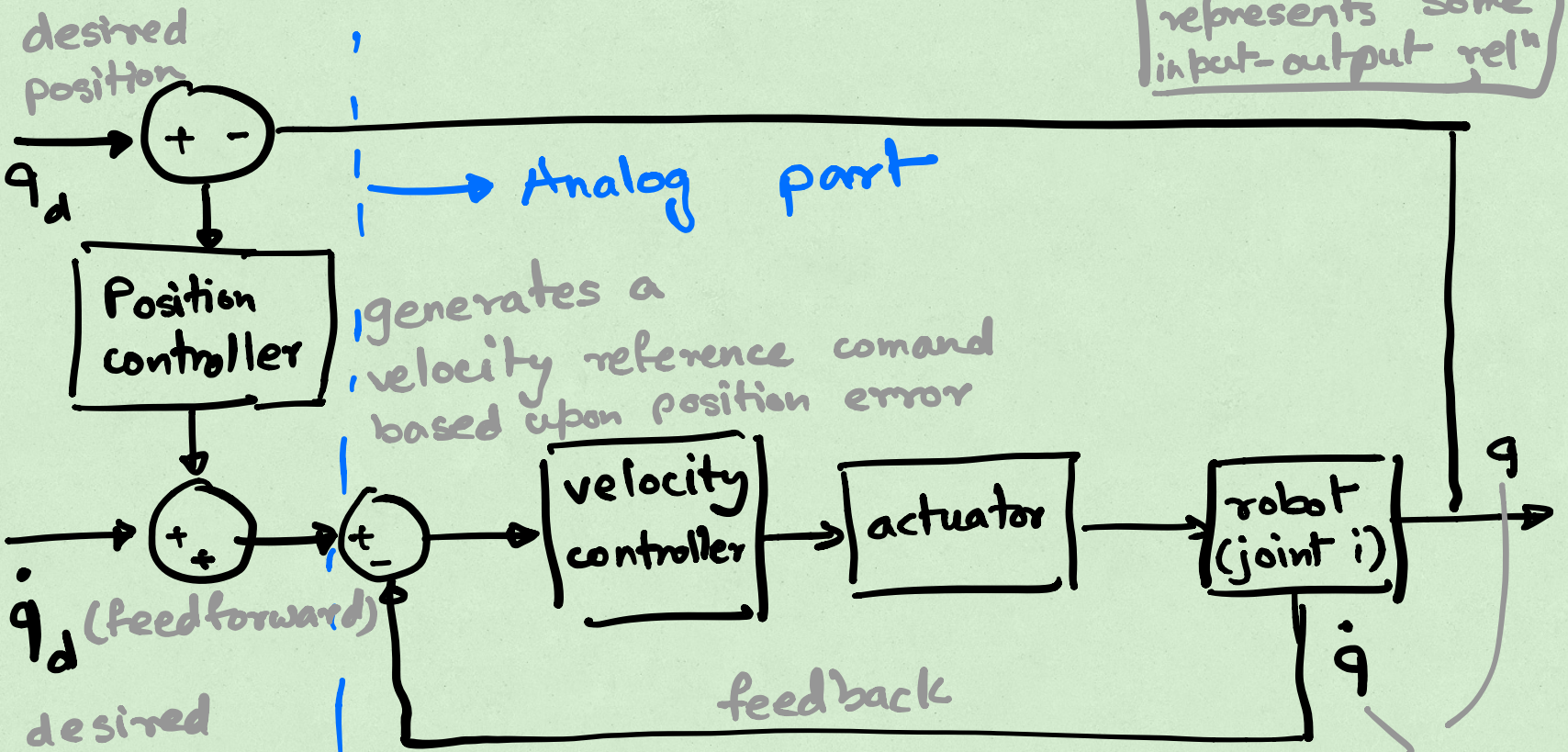
typically $\omega < 5$ Hz

Kinematic control of robots

- Robot is driven by torques produced by motors
- Possible to consider a Kinematic command, such as velocity, as an input
  - using low-level feedback control at joints that allow imposing commanded reference velocities.
  - many industrial robots have such feedback loops, and only allow Kinematic reference commands.
- Performance is satisfactory, provided desired motion is not too fast and accelerations are small.

Control Loop in industrial robots

Each block represents some input-output rel^n



Digital part

- separate controller for each joint
- Analog loop for velocity control of motor
- need to generate Kinematic commands to control

MULTIVARIABLE CONTROL

Independent joint control does not perform well
- if fast motions are required
- if we have a direct-drive robot (no gearbox)

Recall: Gearboxes introduce friction, backlash and increase link side inertia

These issues are eliminated in a direct drive robot!

But now we need to deal with the coupling effects due to inertia and centrifugal/Coriolis terms

↳ complex control schemes!

Robot Dynamics

System equations (for k-th joint)

$$\sum_{j=1}^{n} d_{jk}\ddot{q}_j + \sum_{i,j} c_{ijk}\dot{q}_i\dot{q}_j + \phi_k = \tau_k$$

ϕ_k ← from $g(q)$

$$J_{m_k}\ddot{\theta}_{m_k} + B_k\dot{\theta}_{m_k} = \frac{K_{m_k}V_k}{R_k} - \frac{\tau_k}{r_k}$$

$$B_k = B_{m_k} + \frac{K_{b_k}K_{m_k}}{R_k}$$

using $\theta_{m_k} = r_k q_k$,

$$r_k^2 J_{m_k}\ddot{q}_k + \sum_{j=1}^{n} d_{jk}\ddot{q}_j + \sum_{i,j} c_{ijk}\dot{q}_i\dot{q}_j + r_k^2 B_k\dot{q}_k + \phi_k = r_k\frac{K_{m_k}V_k}{R_k}$$

In matrix form,

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

$$M(q) = D(q) + J$$

$$J = \begin{bmatrix} r_1^2 J_{m_1} & & 0 \\ & \ddots & \\ 0 & & r_n^2 J_{m_n} \end{bmatrix}$$

$$B = \begin{bmatrix} r_1^2 B_1 & & 0 \\ & \ddots & \\ 0 & & r_n^2 B_n \end{bmatrix}$$

$$u_k = r_k \frac{K_{m_k}}{R_k} V_k$$

We can use similar concepts as with the single DOF system!

B tends to stabilize the system, hence we are going to assume $B = 0$ for simplicity.

PD control with gravity compensation

to eliminate steady state error

Choose $u = g(q) + K_p(q_d - q) - K_v\dot{q}$

$= g(q) + K_p e - K_v\dot{q}$ diagonal matrices now

Stability

$K = \frac{1}{2}\dot{q}^T M(q)\dot{q}$ $\qquad$ $V = \frac{1}{2}e^T K_p e$

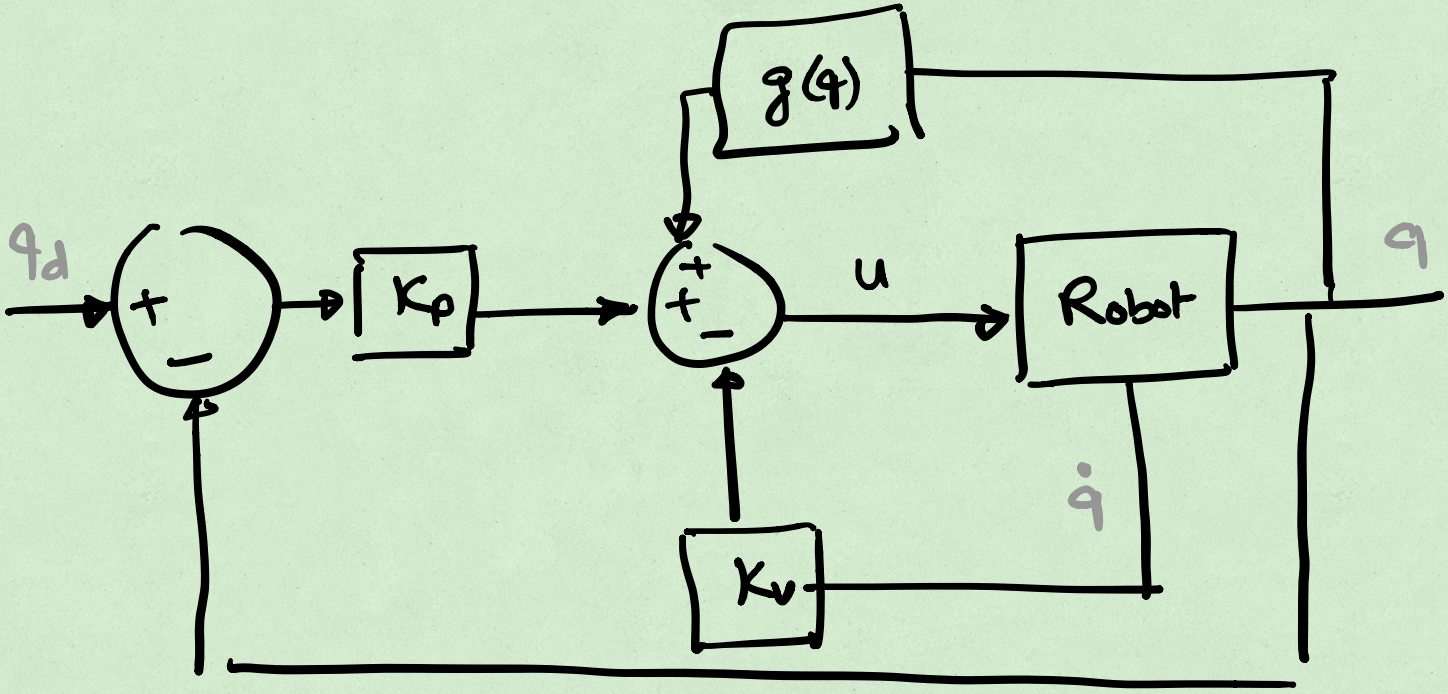
just as in the case of single DOF case

rate of change of total energy

$= -\dot{q}^T K_v \dot{q} < 0$, if $\dot{q} \neq 0$

System is stable!

It's easy to verify that at steady state $q = q_d$!



- $g(q)$ is computed, at every time step, using the measured value of q!

Control partitioning (inverse dynamics control)

Nonlinear decoupling: we want to choose u such that the system behaves as n unit masses.

- Again we can borrow from the SDOF case,

$$u = M(q)\,a_q + C(q,\dot{q})\dot{q} + g(q)$$

a_q → to be chosen.

$$\Rightarrow \ddot{q} = a_q$$

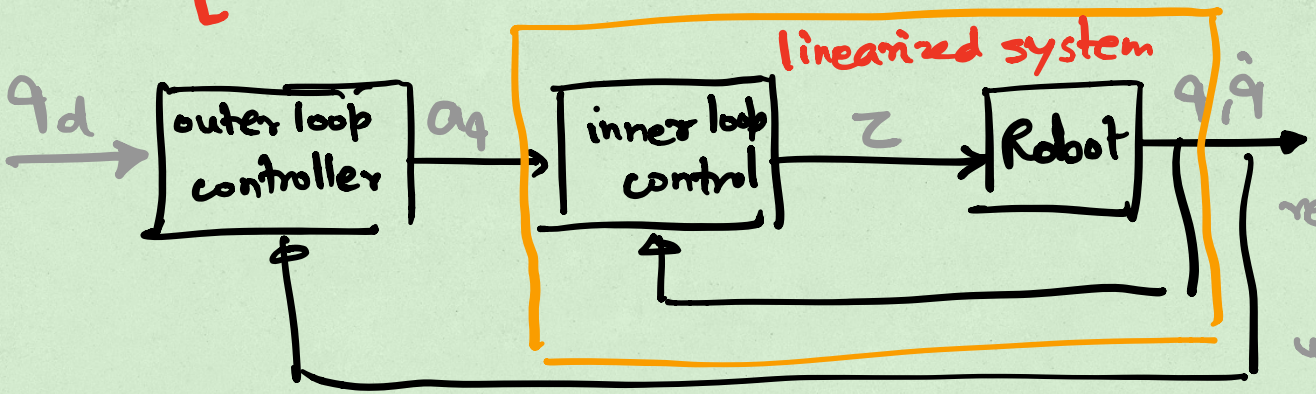
if a_q is a PD trajectory tracking controller,

Let, $a_q = \ddot{q}_d - K_p e - K_v \dot{e}$, $e = q - q_d$

then, $\ddot{e} + K_p e + K_v \dot{e} = 0$

Choose,

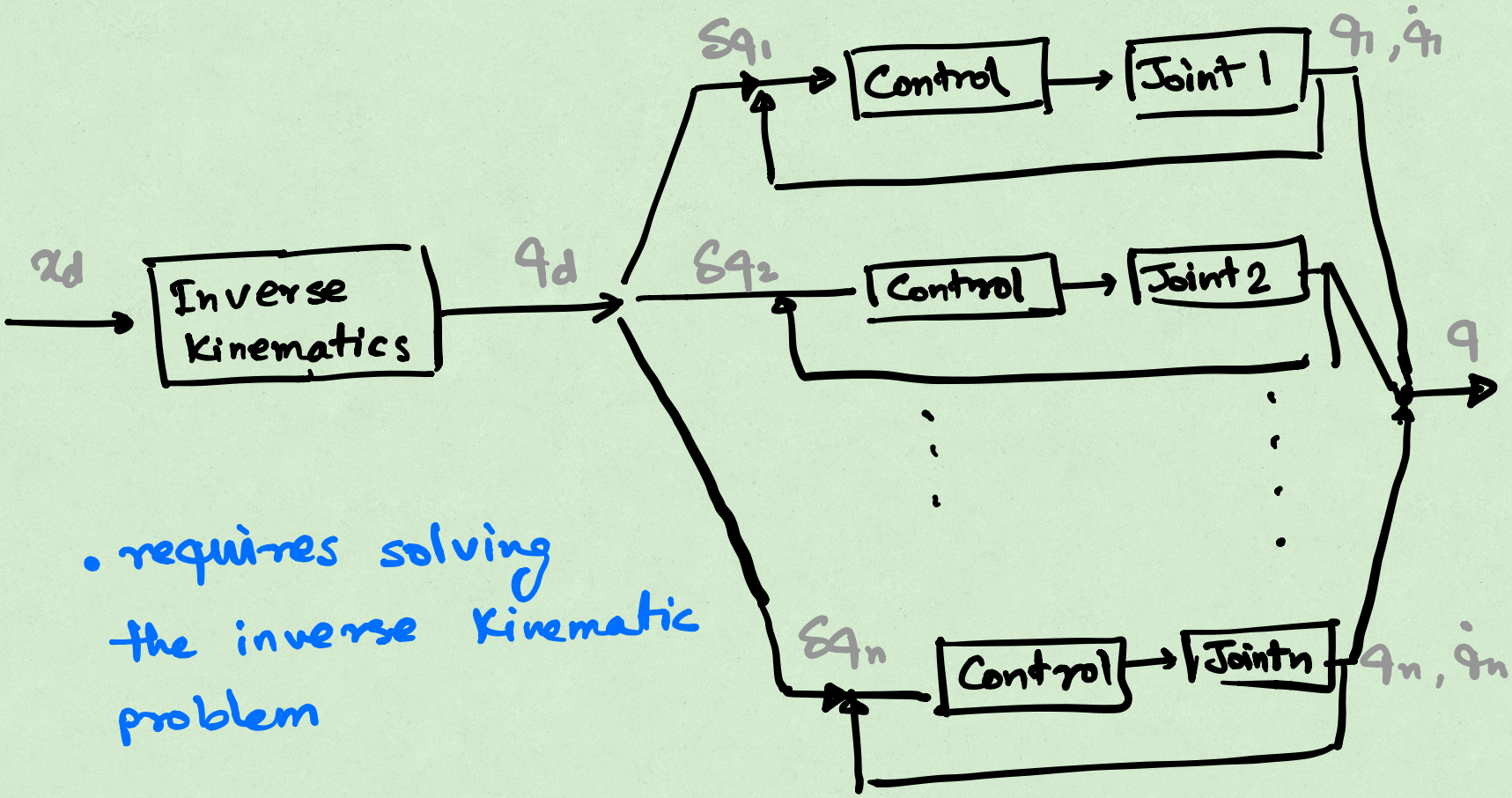
$$K_p = \begin{bmatrix} \omega_1^2 & & \\ & \ddots & \\ & & \omega_n^2 \end{bmatrix} \qquad K_v = \begin{bmatrix} 2\omega_1 & & \\ & \ddots & \\ & & 2\omega_n \end{bmatrix} \quad (\zeta = 1)$$



remember that we are really using estimates based on the model!

implementation requires computation of $M(q)$, $C(q,\dot{q})$ and $g(q)$ at each time step.

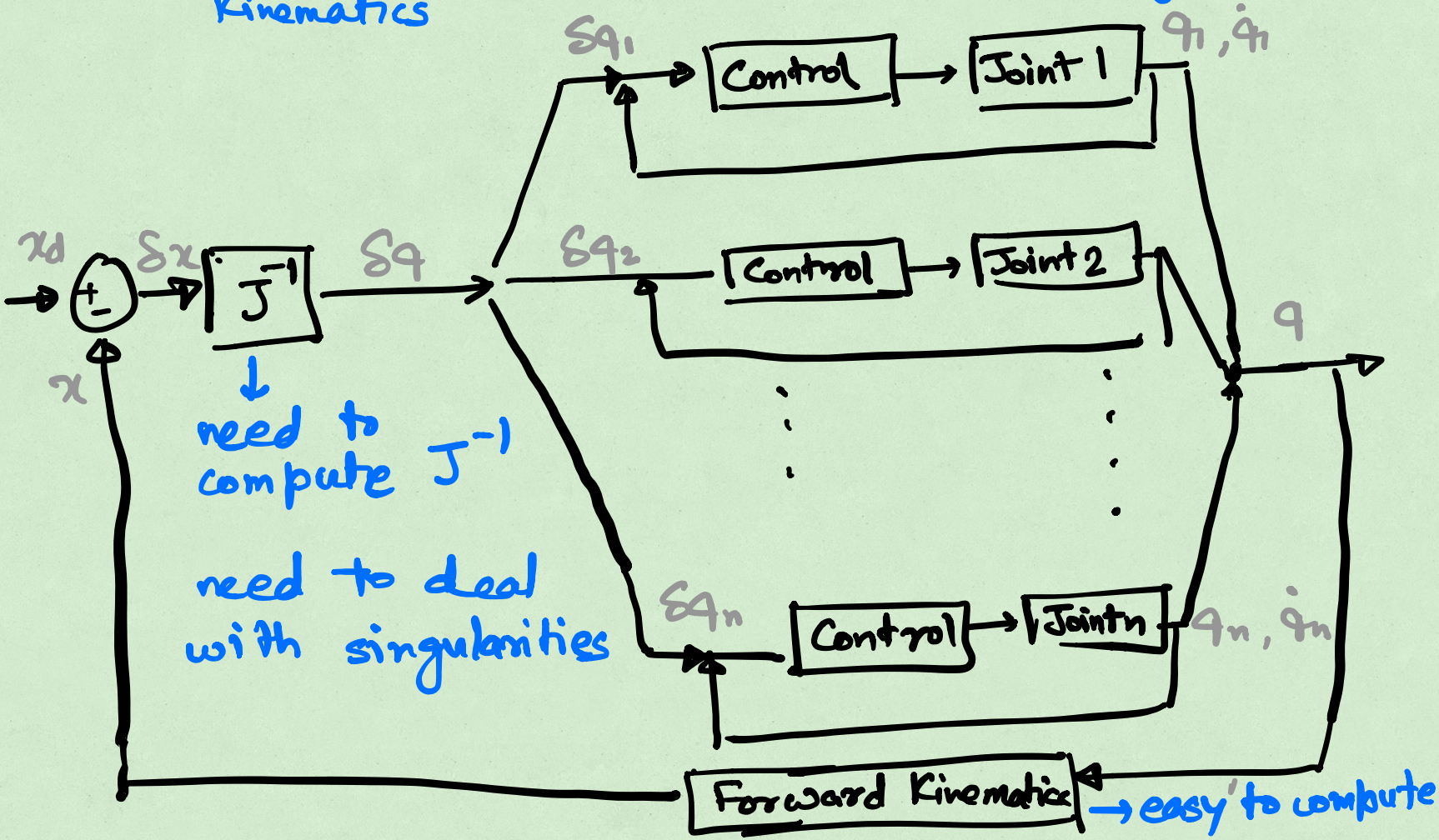
Joint space control



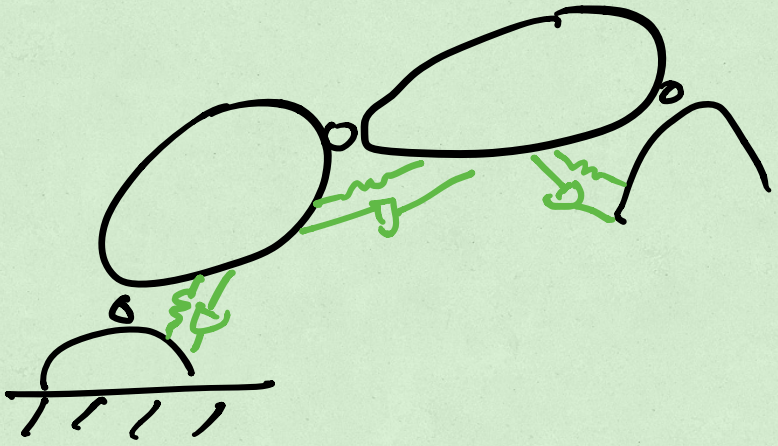
- requires solving the inverse kinematic problem

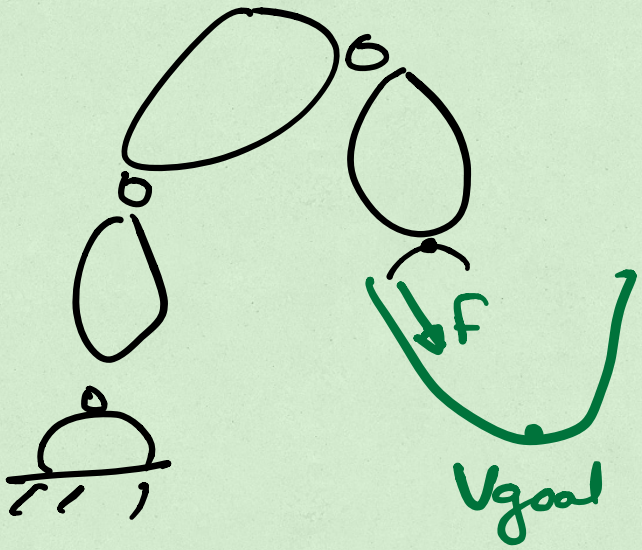
Task-space control

- use the Jacobian to avoid solving inverse Kinematics



Task space control





Joint space control

- like having springs at each joint
- need to compute inverse kinematics
- no need to worry about singularities

Task space control

- Define potential field in task space
- $f = -\frac{\partial V}{\partial x}$

$\tau = J^T f$

- easier to deal with objects

Task-space inverse dynamics

- a_q can be modified to follow a task space trajectory $X(t)$ — specifies position & orientation

We know, $\dot{X} = J\dot{q}$ $\quad\quad$ $\ddot{X} = \dot{J}\dot{q} + J\ddot{q}$

J: analytical Jacobian

$\ddot{q} = a_q$ after decoupling

Let, $a_q = J^{-1}(a_x - \dot{J}\dot{q})$,

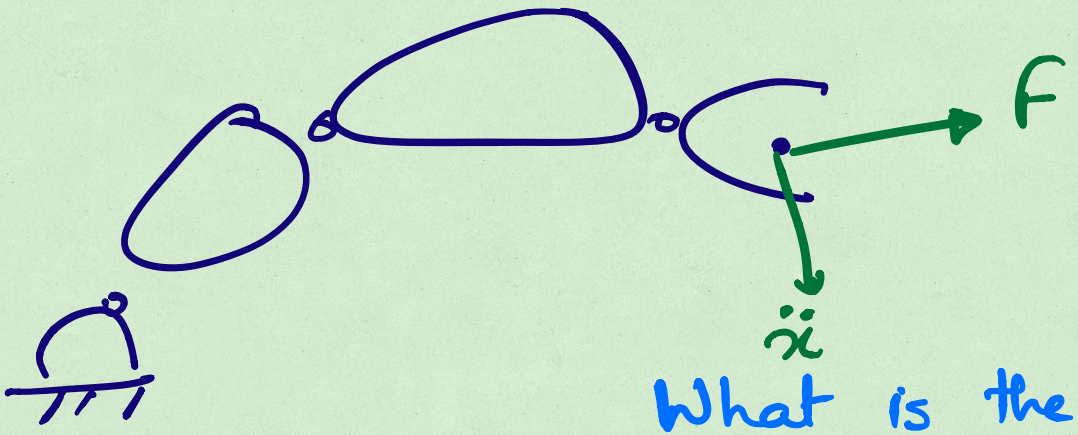
then, $\ddot{X} = a_x$ (similar to joint space inverse dynamics)

$a_x = \ddot{X}_d + K_p(X_d - X) + K_v(\dot{X}_d - \dot{X})$

$\ddot{X}$ + $K_v \dot{e}$ + $K_p e = 0$

$\dot{X}$ forward kin. needs to be computed

Task space dynamics



What is the relationship between F and $\ddot{x}$?

Note that due to coupling of links F and $\ddot{x}$ will be in different directions.

Behavior will also be direction dependent!

For example, if we push the end-effector, inertia felt would be different in diff. directions

Task space equations

end-effector generalized forces

Equations of motion

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right) - \frac{\partial L}{\partial x} = f$$

$$x = \begin{bmatrix} x \\ y \\ z \\ \alpha \\ \beta \\ \gamma \end{bmatrix}$$

$$M_x(x)\ddot{x} + V_x(x,\dot{x}) + G_x(x) = f$$

- x: end-effector position and orientation
- $M_x(x)$: end-effector kinetic energy matrix
- $V_x(x,\dot{x})$: centrifugal and Coriolis terms
- $G_x(x)$: gravity forces

Kinetic energy,

$$\frac{1}{2}\dot{q}^T D(q)\,\dot{q} = \frac{1}{2}\dot{x}^T M_x(x)\,\dot{x}$$

$$= \frac{1}{2}\dot{q}^T J^T M_x J\,\dot{q}$$

$$\downarrow$$

$$\frac{1}{2}\dot{q}^T D\,\dot{q} = \frac{1}{2}\dot{q}^T (J^T M_x J)\,\dot{q}$$

$$M_x = J^{-T}(q)\, D(q)\, J^{-1}$$

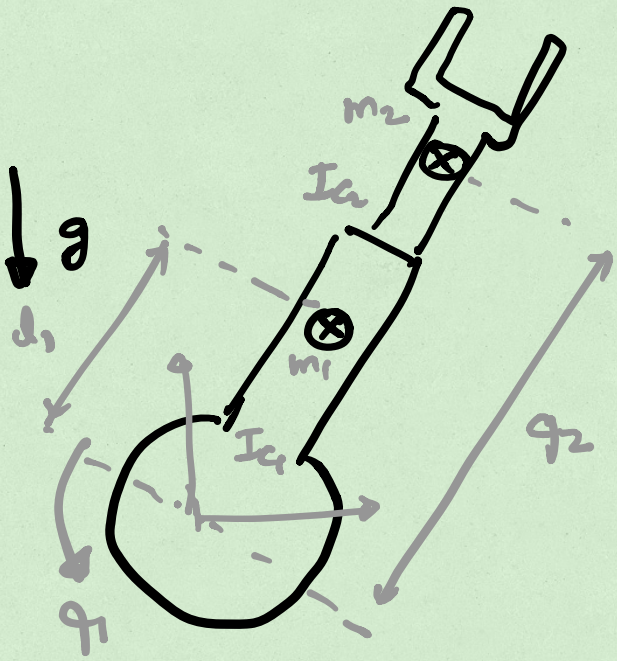
Task space equations (example)

$$q_2 = d_2$$

$$x = d_2 c_1$$

$$y = d_2 s_1$$

Jacobian, ${}^0J = \begin{bmatrix} -d_2 s_1 & c_1 \\ d_2 c_1 & s_1 \end{bmatrix}$



$${}^1J = \begin{bmatrix} 0 & 1 \\ d_2 & 0 \end{bmatrix}$$

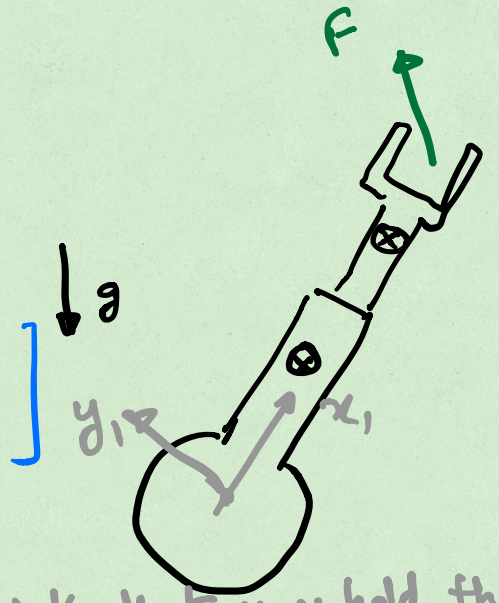
$${}^1J^{-1} = \begin{bmatrix} 0 & 1/d_2 \\ 1 & 0 \end{bmatrix}$$

Jacobian in frame 1

not m_1, m_2!

$$ {}^{1}M_x = \begin{bmatrix} 0 & 1 \\ 1/d_2 & 0 \end{bmatrix} \begin{bmatrix} m'_{11} & 0 \\ 0 & m_{22} \end{bmatrix} \begin{bmatrix} 0 & 1/d_2 \\ 1 & 0 \end{bmatrix} $$

$$ = \begin{bmatrix} m_2 & 0 \\ 0 & m_2 + m'_2 \end{bmatrix} $$



think that you hold the end-effector and pull. M_x gives information about the mass you will 'feel'.

to the end-effector

- If force F is applied in x_1-dir, the robot inertia felt is m_2: (only Link 2 moves!)
- If F is in y_1-dir, inertia felt is m_2 and a contribution due to Link 1 (will depend mass, inertia lengths etc.)

m'_2 is this effective mass felt at the end-effector

$$ {}^{0}M_{2} = \begin{bmatrix} c_1 & -s_1 \\ s_1 & c_1 \end{bmatrix} \begin{bmatrix} m_2 & 0 \\ 0 & m_2 + m_2' \end{bmatrix} \begin{bmatrix} c_1 & s_1 \\ -s_1 & c_1 \end{bmatrix} $$

rotation matrices to map J^{-T} and J^{-1}

$$ = \begin{bmatrix} m_2 + m_2' s_1^2 & -m_2' s_1 c_1 \\ -m_2' s_1 c_1 & m_2 + m_2' c_1^2 \end{bmatrix} $$

Not diagonal

direction of application of force and end-effector motion are not same!

Some properties may be decoupled in task space (depends on robot structure)

Similarly,

$$V_x(x,\dot{x}) = J^{-T} V(q,\dot{q}) - M_x(q)\, h(q,\dot{q})$$

$$h = \dot{J}\dot{q}$$

$$G_x(x) = J^{-T} G(q)$$

We can now apply earlier ideas for control:

PD control w/ gravity compensation

$$F = -K_p (x - x_{goal}) - K_v \dot{x} + \hat{G}(x)$$

or, $F = \hat{M}(x) F' + \hat{V}_x(x,\dot{x}) + \hat{G}(x)$, $\tau = J^T F$

non-linear decoupling

Stability can also be demonstrated as before.