

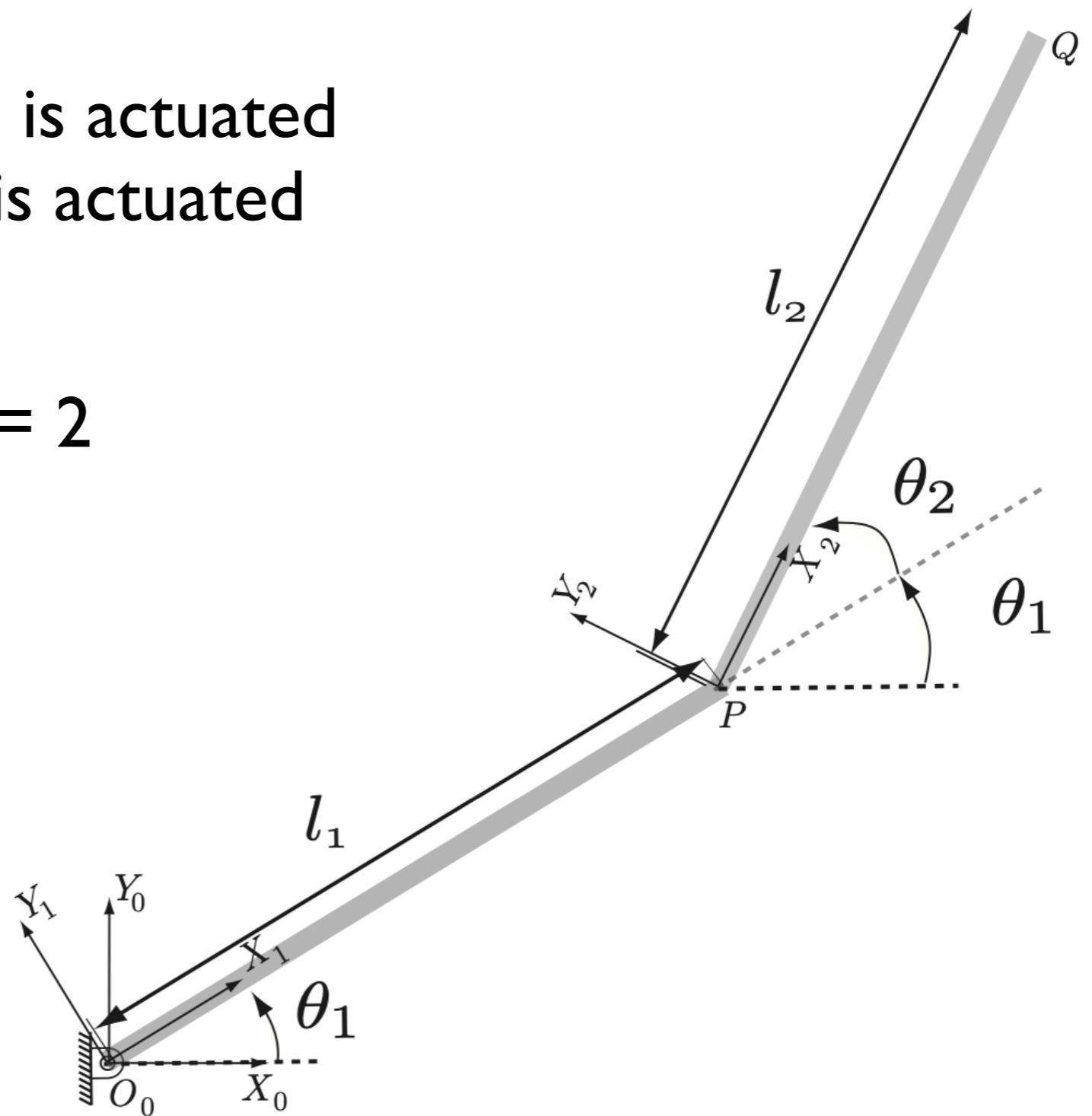
MuJoCo: Control underactuated systems (I)

- 1) Pendubot: Only link 1 is actuated
- 2) Acrobot: Only link 2 is actuated

Degrees of freedom (m) = 2

Actuators (n) = 1

Under-actuation, $m > n$

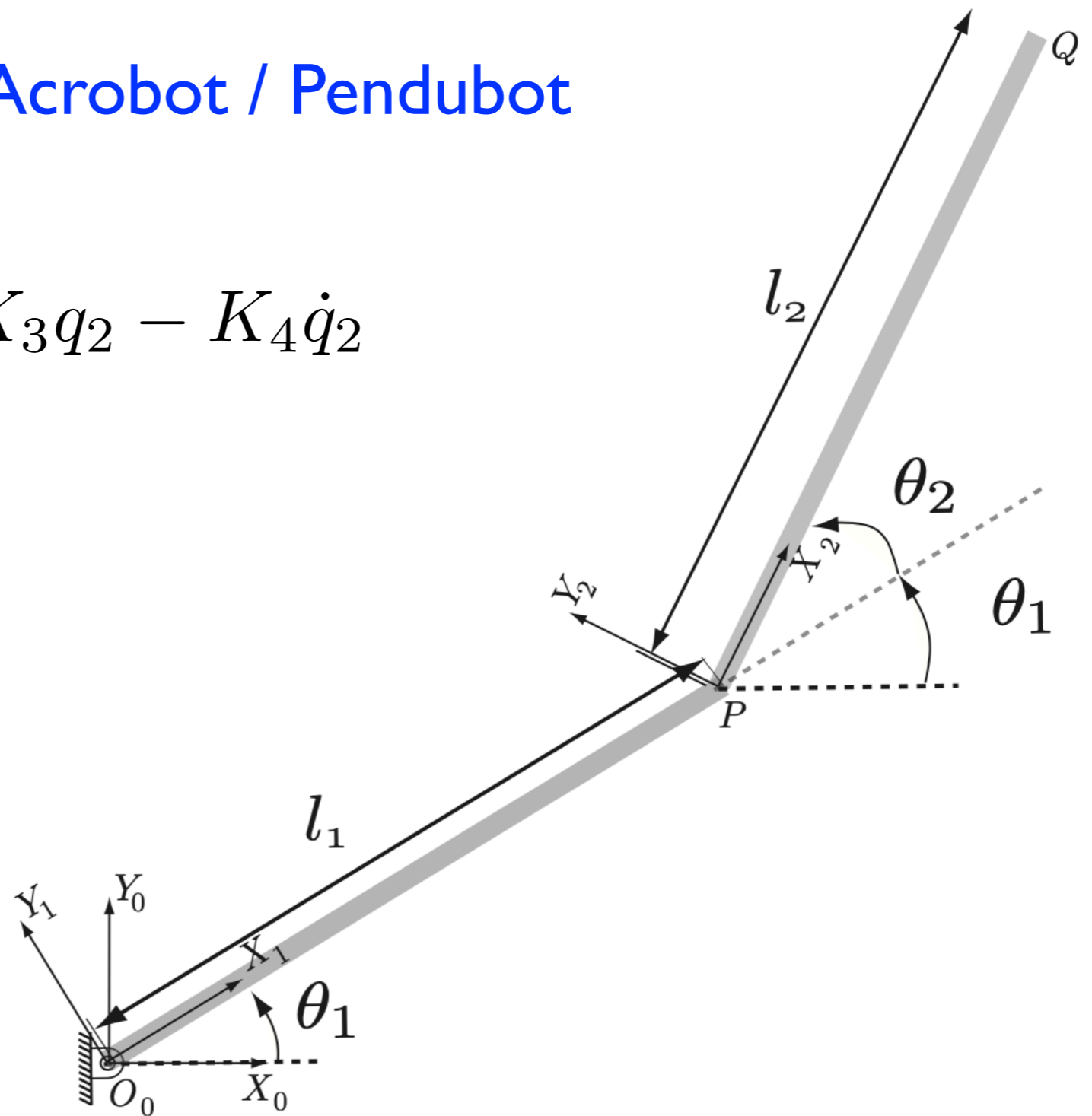


MuJoCo: Control underactuated systems (2)

Goal: Balance control of Acrobot / Pendubot

$$T = -K_1 q_1 - K_2 \dot{q}_1 - K_3 q_2 - K_4 \dot{q}_2$$

How to choose K's?



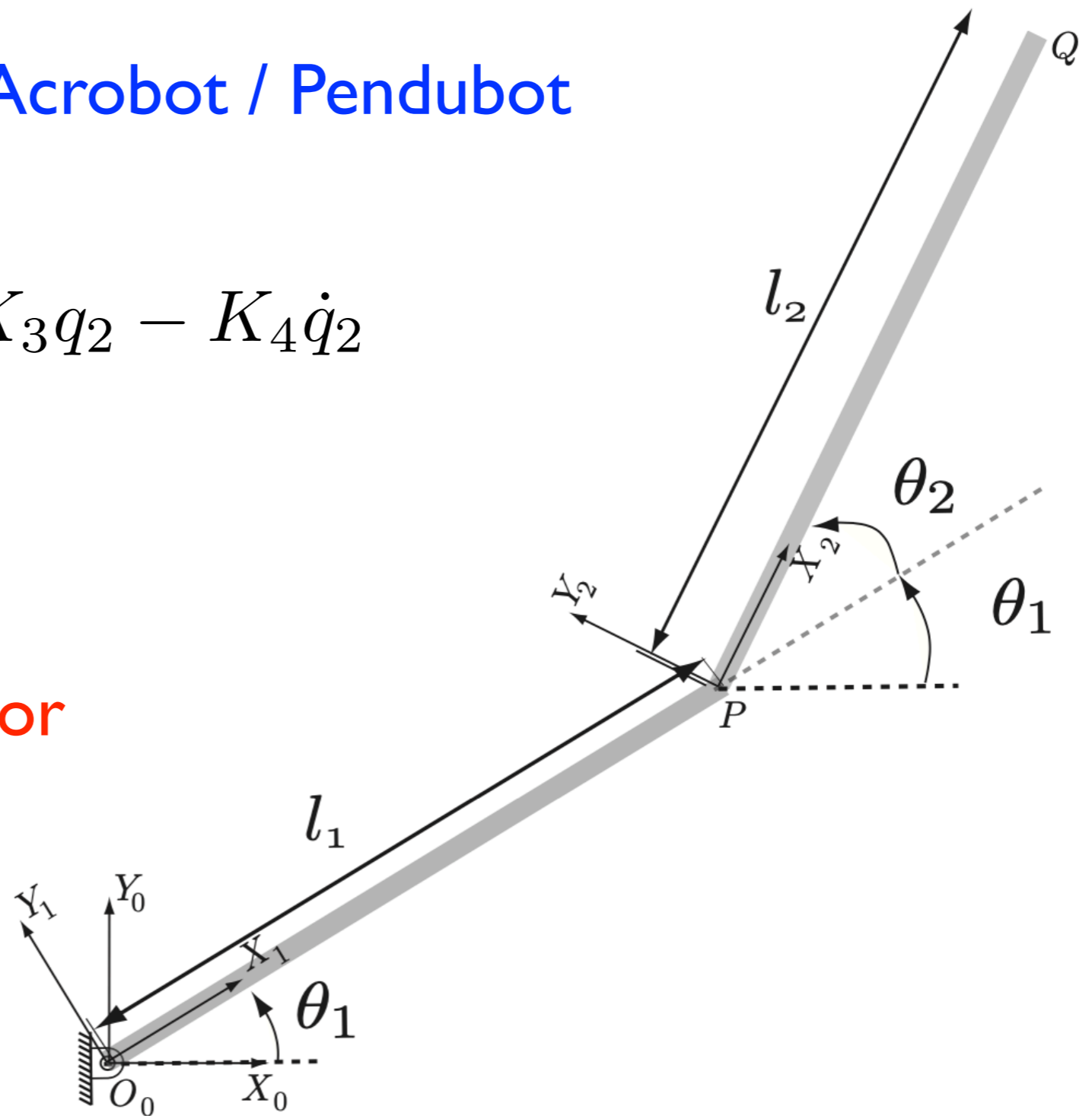
MuJoCo: Control underactuated systems (3)

Goal: Balance control of Acrobot / Pendubot

$$T = -K_1 q_1 - K_2 \dot{q}_1 - K_3 q_2 - K_4 \dot{q}_2$$

How to choose K's?

Linear Quadratic Regulator



We will control the acrobot.

Easy to extend to pendubot or cart-pendulum

MuJoCo: Linear Quadratic Regulator

Using [template_dbpendulum.zip](#) to get started

1. From tiny.cc/mujoco download `template_dbpendulum.zip` and unzip in `myproject`
2. Rename folder `template` to `dbpendulum_lqr`
3. Make these three changes
 1. `main.c` — line 28, change `template_writeData2/` to `dbpendulum_lqr/`
 2. `makefile` — change `ROOT = template_writeData` to `ROOT = dbpendulum_lqr` also UNCOMMENT (del #) appropriate to your OS
 3. `run_unix / run_win.bat` change `<template_writeData2>` to `<dbpendulum_lqr>`
4. In the *shell, navigate to `dbpendulum_lqr` and type `./run_unix` (unix) or `run_win` (windows); *shell = terminal for mac/linux / x64 for win

MuJoCo: Linear Quadratic Regulator (L)

$$\dot{x} = \mathbf{A}x + \mathbf{B}u$$

Compute input u such that $x(t) \rightarrow 0$

$$\text{minimize } J = \int_0^{\infty} (x^T Q x + u^T R u)$$

The minimization gives a gain matrix \mathbf{K} such that

$$u = -\mathbf{K}x$$

where \mathbf{K} is found using `lqr` in matlab: $\mathbf{K} = \text{lqr}(\mathbf{A}, \mathbf{B}, \mathbf{Q}, \mathbf{R})$

MuJoCo: Linear Quadratic Regulator (2)

$$\text{minimize } J = \int_0^{\infty} (x^T Q x + u^T R u)$$

- Q and R are user chosen matrices
- One way of choosing these matrices
 - $Q = I_{(n \times n)}$ and $R = \rho I_{(m \times m)}$;
 - I is the identity matrix, n is size of x, m is size of u, rho is a design parameter
- $\rho \ll 1$ will give large gains and vice versa for $\rho \gg 1$

MuJoCo: Linear Quadratic Regulator (3)

Most systems are nonlinear

$$\dot{x} = f(x, u)$$

This equation need to be linearized to

$$\dot{x} = \mathbf{A}x + \mathbf{B}u$$

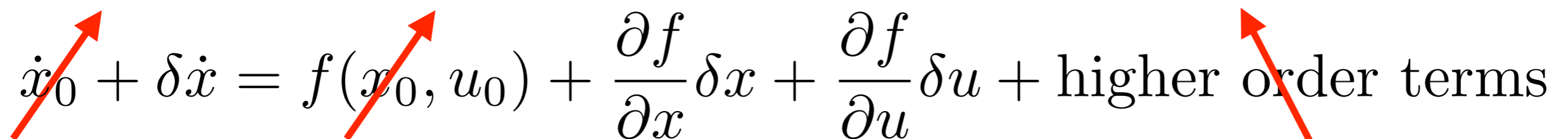
MuJoCo: Linear Quadratic Regulator (4)

Linearize about a reference set point, (x_0, u_0)

$$\dot{x}_0 = f(x_0, u_0)$$

Taylor series expansion, consider only first-order terms

$$\dot{x}_0 + \delta\dot{x} = f(x_0, u_0) + \frac{\partial f}{\partial x} \delta x + \frac{\partial f}{\partial u} \delta u + \text{higher order terms}$$


$$\cancel{\dot{x}_0} + \delta\dot{x} = \cancel{f(x_0, u_0)} + \frac{\partial f}{\partial x} \delta x + \frac{\partial f}{\partial u} \delta u + \cancel{\text{higher order terms}}$$

$$\delta\dot{x} = A\delta x + B\delta u \quad \text{where} \quad A = \frac{\partial f}{\partial x} \quad \text{and} \quad B = \frac{\partial f}{\partial u}$$

MuJoCo: Double pendulum (I)

Equations for double pendulum

$$M(q)\ddot{q} + N(q, \dot{q}) = T$$

- M is the mass matrix, It's dimension is 2×2
- N is gravity + coriolis forces, It's dimension is 2×1
- T is the external torque, It's dimension is 2×1

Pendubot $T = \begin{bmatrix} u \\ 0 \end{bmatrix}$

Acrobot $T = \begin{bmatrix} 0 \\ u \end{bmatrix}$

MuJoCo: Double pendulum (2)

We need to write equations in this fashion.

$$\dot{x} = f(x, u) \quad \text{Eq. 1}$$

From equations of pendulum:

$$\ddot{q} = M^{-1}(q)(T - N(q, \dot{q})) = \{\ddot{q}_1, \ddot{q}_2\}^T \quad \text{Eq. 2}$$

We can now write a function f (see Eq 1) as follows

$$\underbrace{(\dot{q}_1, \ddot{q}_1, \dot{q}_2, \ddot{q}_2)}_{\text{Outputs}} = (f_1, f_2, f_3, f_4) = f(\underbrace{q_1, \dot{q}_1, q_2, \dot{q}_2, u}_{\text{Inputs}}) \quad \text{Eq. 3}$$

From Eq. 2

Lets write code to compute f (Eq. 3)

MuJoCo: Double pendulum (3)

Computing linearization of f : $A = \frac{\partial f}{\partial x}$ $B = \frac{\partial f}{\partial u}$

where $f = \{\dot{q}_1, \ddot{q}_1, \dot{q}_2, \ddot{q}_2\}^T$ and $x = \{q_1, \dot{q}_1, q_2, \dot{q}_2\}^T$

A and B matrices look like this

$$A = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial q_1} & \frac{\partial f_1}{\partial \dot{q}_1} & \frac{\partial f_1}{\partial q_2} & \frac{\partial f_1}{\partial \dot{q}_2} \\ \frac{\partial f_2}{\partial q_1} & \frac{\partial f_2}{\partial \dot{q}_1} & \frac{\partial f_2}{\partial q_2} & \frac{\partial f_2}{\partial \dot{q}_2} \\ \frac{\partial f_3}{\partial q_1} & \frac{\partial f_3}{\partial \dot{q}_1} & \frac{\partial f_3}{\partial q_2} & \frac{\partial f_3}{\partial \dot{q}_2} \\ \frac{\partial f_4}{\partial q_1} & \frac{\partial f_4}{\partial \dot{q}_1} & \frac{\partial f_4}{\partial q_2} & \frac{\partial f_4}{\partial \dot{q}_2} \end{bmatrix} \quad B = \frac{\partial f}{\partial u} = \begin{bmatrix} \frac{\partial f_1}{\partial u} \\ \frac{\partial f_2}{\partial u} \\ \frac{\partial f_3}{\partial u} \\ \frac{\partial f_4}{\partial u} \end{bmatrix}$$

MuJoCo: Double pendulum (4)

Computing A and B using finite difference (pseudo-code)

1) Create a function that returns f : $f(q_1, \dot{q}_1, q_2, \dot{q}_2, u)$

2) Compute nominal f_0 (4x1): $f(q_1^0, \dot{q}_1^0, q_2^0, \dot{q}_2^0, u^0)$

3) Perturb first element $q_1^0 + \epsilon$

4) Compute new f (4x1): $f(q_1^0 + \epsilon, \dot{q}_1^0, q_2^0, \dot{q}_2^0, u^0)$

5) Compute first column of A:

$$A(:, 1) = \frac{f(q_1^0 + \epsilon, \dot{q}_1^0, q_2^0, \dot{q}_2^0, u^0) - f(q_1^0, \dot{q}_1^0, q_2^0, \dot{q}_2^0, u^0)}{\epsilon}$$

MuJoCo: Double pendulum (5)

Computing A and B using finite difference (pseudo-code)

6) Repeat to compute other rows of columns of A and so on

$$A(:, 2) = \frac{f(q_1^0, \dot{q}_1^0 + \epsilon, q_2^0, \dot{q}_2^0, u^0) - f(q_1^0, \dot{q}_1^0, q_2^0, \dot{q}_2^0, u^0)}{\epsilon}$$

7) Compute B

$$B = \frac{f(q_1^0, \dot{q}_1^0, q_2^0, \dot{q}_2^0, u^0 + \epsilon) - f(q_1^0, \dot{q}_1^0, q_2^0, \dot{q}_2^0, u^0)}{\epsilon}$$

MuJoCo: Double pendulum (6)

8) Computing gain matrix K using `lqr`

In MATLAB: $K = \text{lqr}(A,B,Q,R)$

9) Test the controller

- Initial perturbation: `d->qpos` and `d->qvel`
- Adding noise in `mycontroller`
 - `mju_standardNormal` (to generate noise)
 - `d->xfric_applied` (cartesian force)
 - `d->qfric_applied` (joint torque)