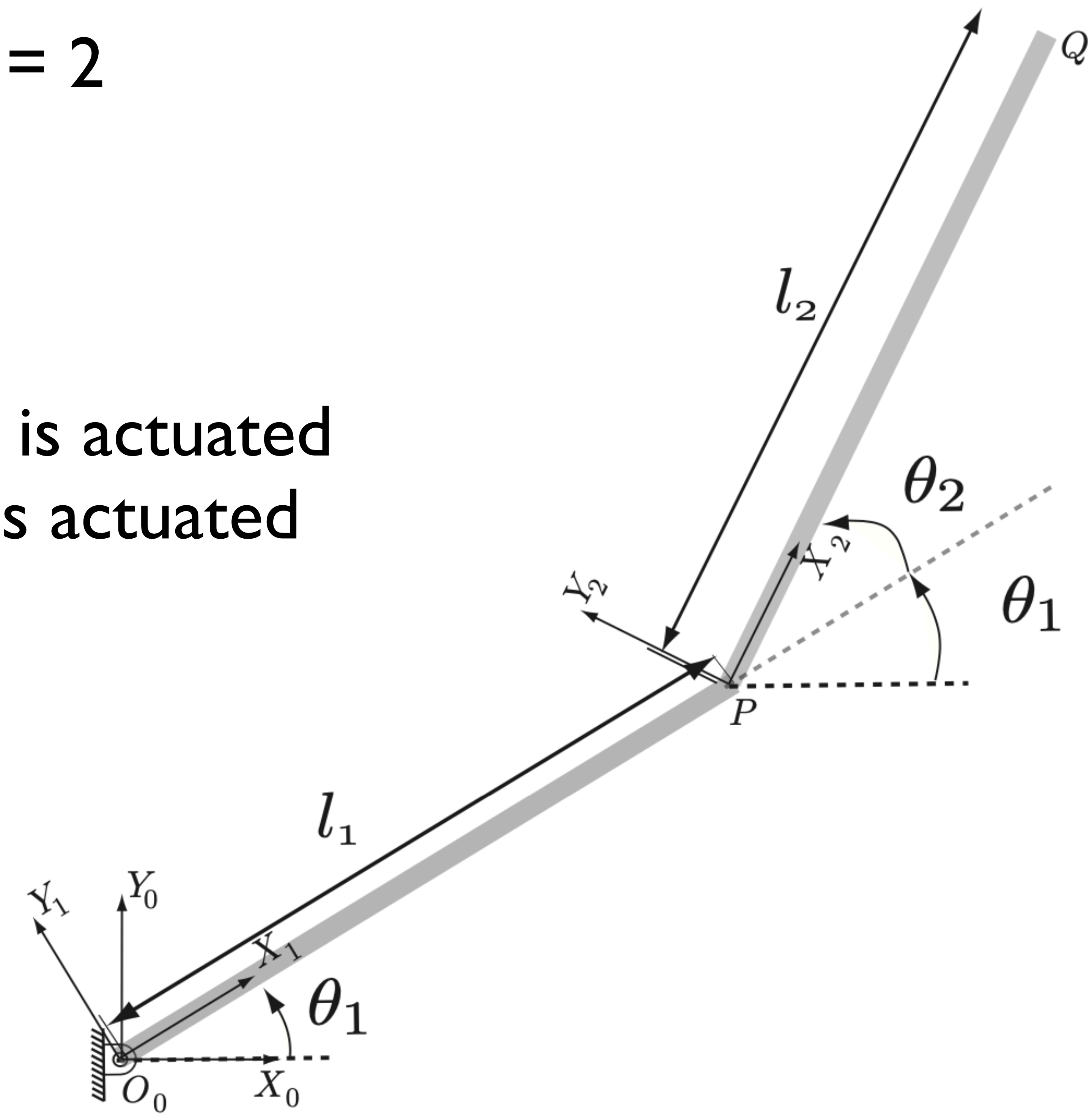
Control an underactuated systems (1)

Degrees of freedom (m) = 2
Actuators (n) = 1
Under-actuation, m > n

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

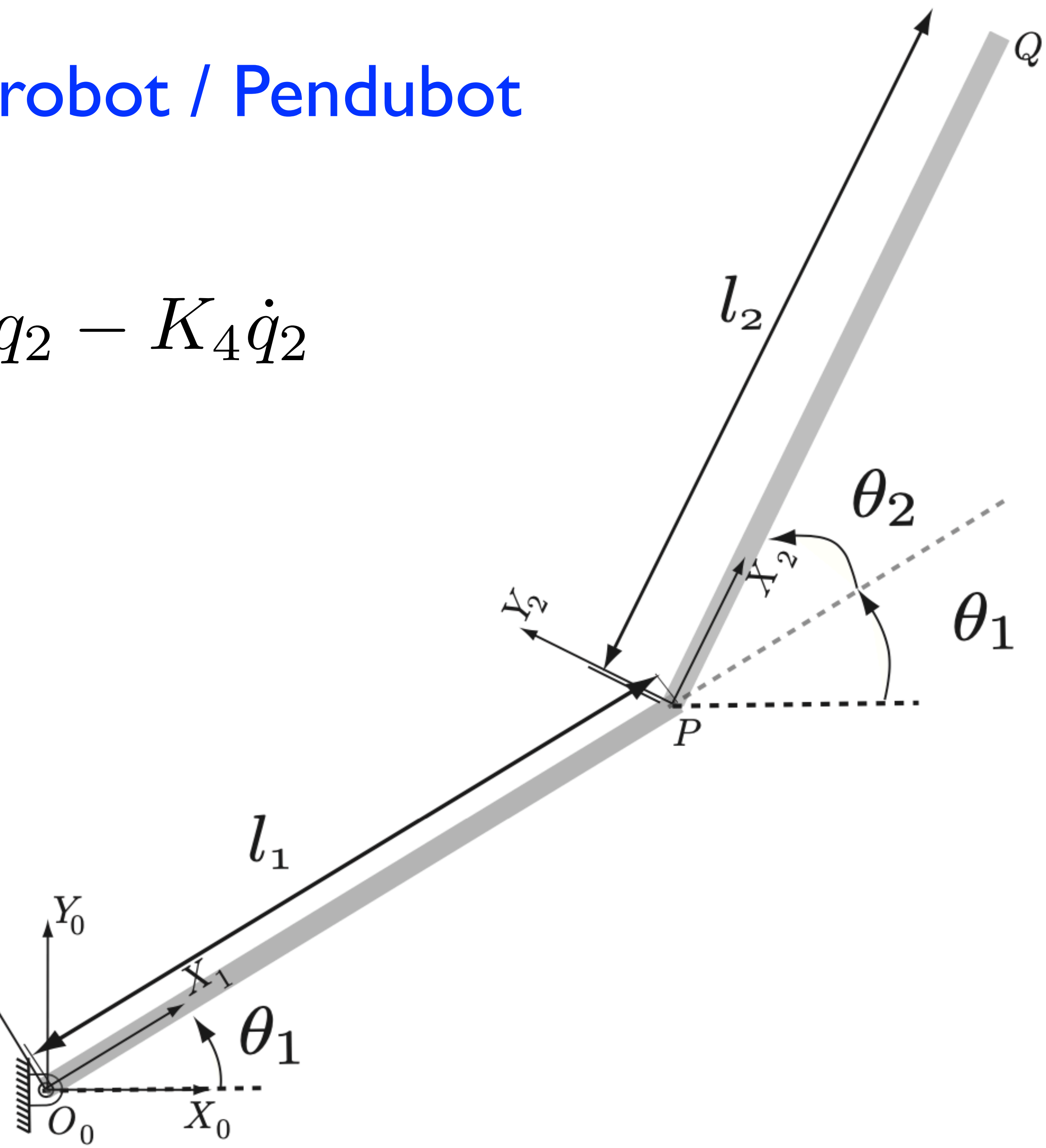


Control an underactuated systems (2)

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?



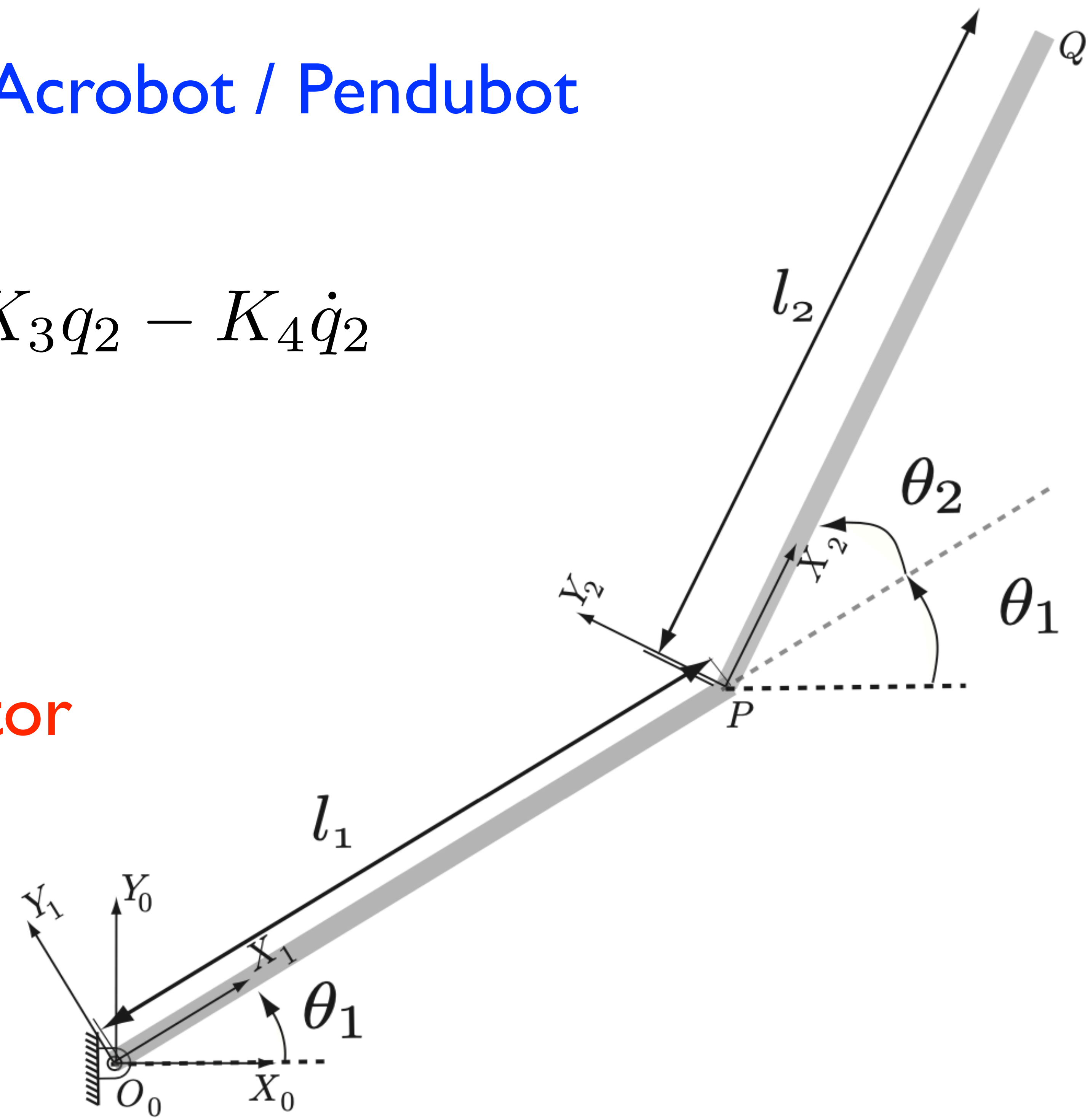
Control an underactuated systems (3)

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 pendubot (link 1 is actuated)
Easy to extend to acrobat (link 2 is actuated)

Linear Quadratic Regulator (1)

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

Compute input u such that x(t) ->0

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

The minimization gives a gain matrix K such that

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

K is found using lqr in python: K = control.lqr(A,B,Q,R)

Install control; in terminal: pip install control

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 matrics
  - Q = I_(nxn) and R = rho I_(mxm);
  - I is the identity matrix, n is size of x, m is size of u, rho is a design parameter
- rho << 1 will give large gains and vice versa for rho >> 1

Linear Quadratic Regulator (3)

Most systems are nonlinear

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

This equation need to be linearized to

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

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

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

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

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

Double pendulum (1)

Equations for double pendulum

M qddot + frc_bias = tau

- M is the mass matrix, It’s dimension is 2x2
- frc_bias is gravity + coriolis forces, It’s dimension is 2x1
- tau is the external torque, It’s dimension is 2x1

Pendubot tau = [ctrl 0]’

Acrobot tau = [0 ctrl]’

Double pendulum (2)

We need to write equations in this fashion.

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

From equations of pendulum:

qddot = f(q,qdot,ctrl)=inverse(M)(tau-frc_bias) Eq. 2

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

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

From Eq. 2

Lets write code to compute xdot = f(x,u) (Eq. 3)

Double pendulum (3)

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

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

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

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 f0 (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:

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

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

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

Double pendulum (6)

8) Computing gain matrix K using lqr

In python: K = control.lqr(A,B,Q,R)

9) Test the controller

- Adding disturbance torque in mycontroller using np.random.normal and qfrc_applied