Feedback for Stabilization & Control of a Two-Link Arm

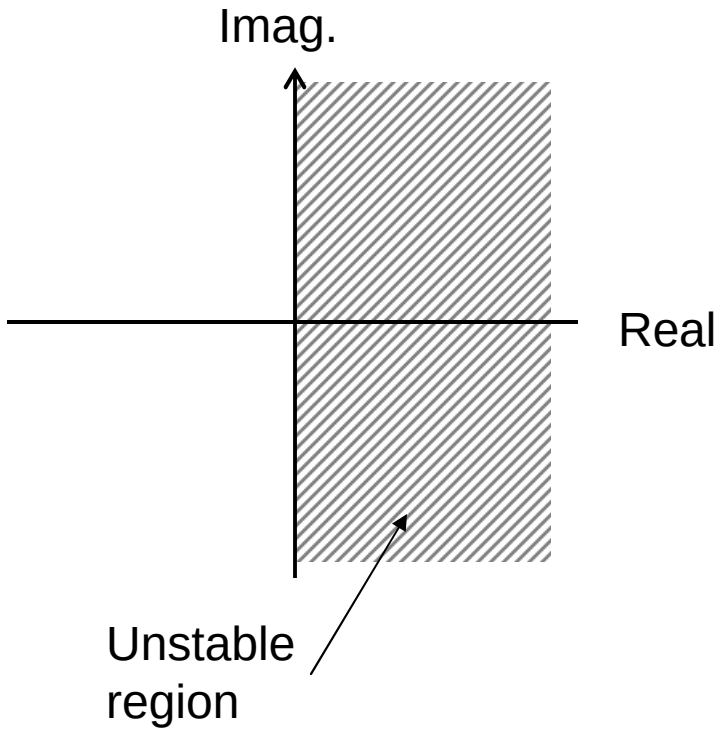
Andrew Stein
Dave Ferguson

11/7/02

Stabilization via Feedback

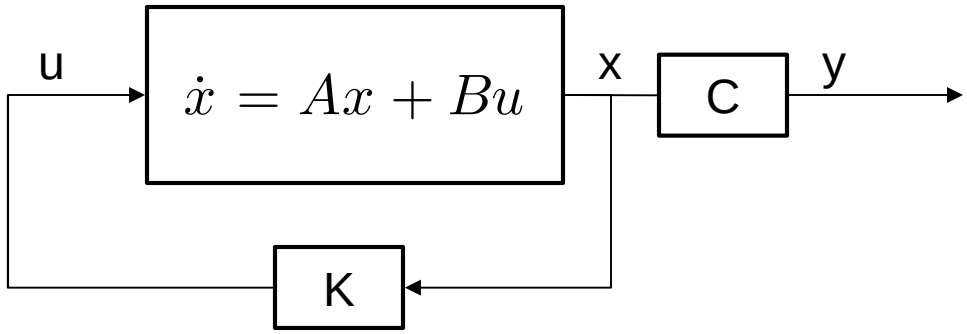
- Given an autonomous system (no external control), we have learned how to determine instability:
  - Poles lie in the right half plane, or, equivalently:
  - Eigenvalues of A lie in the right half plane
- If A is unstable, we can try to stabilize this system via feedback
  - State feedback
  - Output feedback

$$\dot{x} = Ax$$



State Feedback Design

- If states are directly measurable, can create the following system:



- Substituting u=Kx, this system is equivalent to:

$$\dot{x} = \underbrace{(A + BK)}_{M} x$$

- If we choose K such that the real parts of the eigenvalues of M are negative, we have stabilized the system. (This is not always possible.)

State Feedback Example

$$\dot{x} = Ax + Bu$$
$$u = -Kx$$

$$\dot{x} = (A + BK)u = Mx$$

This A is unstable (just trust us!)

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 2 & -1 & 4 & 3 \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$K = [\, k_1 \quad k_2 \quad k_3 \quad k_4 \,]$$

$$M = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 2 & -1 & 4 & 3 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} [\, k_1 \quad k_2 \quad k_3 \quad k_4 \,] = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 2-k_1 & -1-k_2 & 4-k_3 & 3-k_4 \end{bmatrix}$$

We use the characteristic polynomial to find the eigenvalues of M:

$$\det(\lambda I - M) = \lambda^4 - (3 - k_4)\lambda^3 - (4 - k_3)\lambda^2 - (-1 - k_2)\lambda - (2 - k_1) = 0$$

One possible stable M would have all eigenvalues = -1. That would yield:

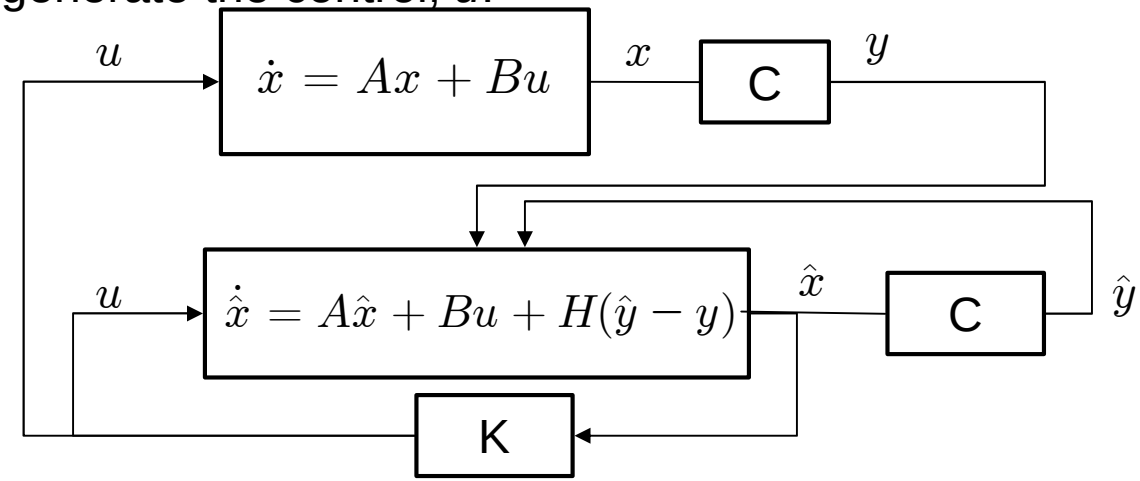
$$(\lambda + 1)^4 = \lambda^4 + 4\lambda^3 + 6\lambda^2 + 4\lambda + 1 = 0$$

Matching coefficients of ? and solving for the k_i:

$$K = [\, 3 \quad 3 \quad 10 \quad 7 \,]$$

Output Feedback Design

- Usually direct state measurements are not available. Then we must use the output as feedback.
- This involves using a *closed-loop observer*, which simulates the actual system to generate the control, *u*:



- The resulting system has the following state dynamics:

$$\begin{bmatrix} \dot{x} \\ \dot{\hat{x}} \end{bmatrix} = \underbrace{\begin{bmatrix} A & BK \\ -HC & A + BK + HC \end{bmatrix}}_{M} \begin{bmatrix} x \\ \hat{x} \end{bmatrix}$$

- We need to choose K and H such that M will be stable. (Not always possible.)
- There is a separation principle that allows us to find K and H separately for the matrices A+BK and A+HC. (It decouples the problem.)

Control Example: Two-link Robot Arm

- Kinetic Energy:

$$v_i = \begin{bmatrix} \dot{x}_i \\ \dot{y}_i \end{bmatrix} = \begin{bmatrix} -l_i \sin(q_i)\dot{q}_i \\ l_i \cos(q_i)\dot{q}_i \end{bmatrix}$$

$$K = \tfrac{1}{2} m_1 \|v_1\|^2 + \tfrac{1}{2} m_2 \|v_2\|^2$$

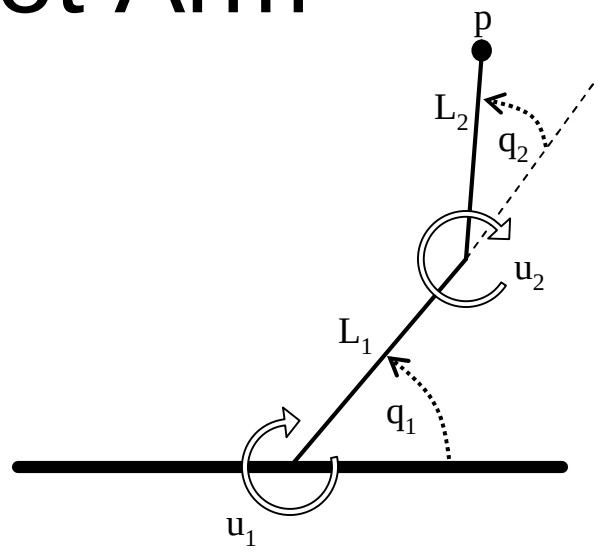
- Potential Energy:

$$P(q) = m_1 g l_1 \sin(q_1) + m_2 g (l_2 \sin(q_1) + l_2 \sin(q_1 + q_2))$$

- Lagrangian:

$$L = K - P$$

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



Notation:

- q_i = Joint Angles
- $\dot{q}_i$ = Joint [angular] velocities
- p = end effector coordinates
- L_i = Link lengths
- m_i = Link masses
- u_i = Joint torques (controls)
- g = gravity

Deriving Arm Dynamics

- D(q) is the *mass matrix* defined (via ugly trigonometry) such that:

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

- Using the Lagrangian, we can obtain the dynamics equation:

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

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right) = D(q)q + \dot{D}(q)\dot{q} \qquad \frac{\partial L}{\partial q} = \frac{\partial K}{\partial q} - \frac{\partial P}{\partial q}$$

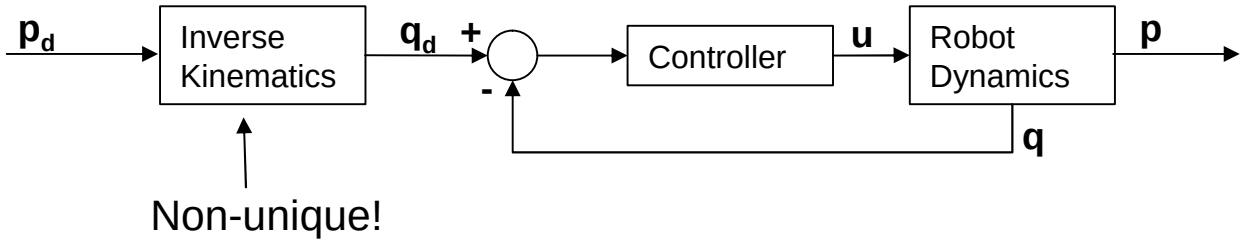
$$\frac{\partial L}{\partial \dot{q}} = D(q)\dot{q}$$

$$\underbrace{D(q)q} + \underbrace{\dot{D}(q)\dot{q} - \frac{\partial K}{\partial q}} + \underbrace{\frac{\partial P}{\partial q}} = u$$

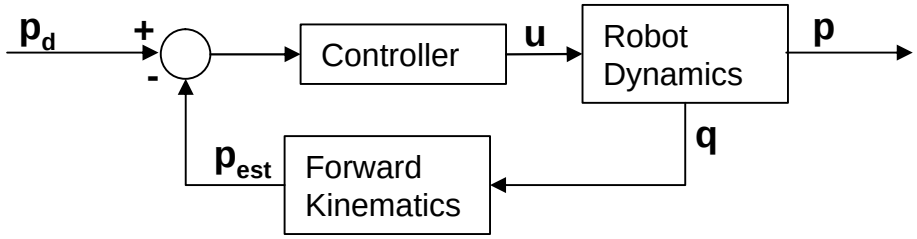
- Final form: $\boxed{D(q)q + V(q,\dot{q}) + G(q) = u}$

Control Strategies

- Joint Space Control

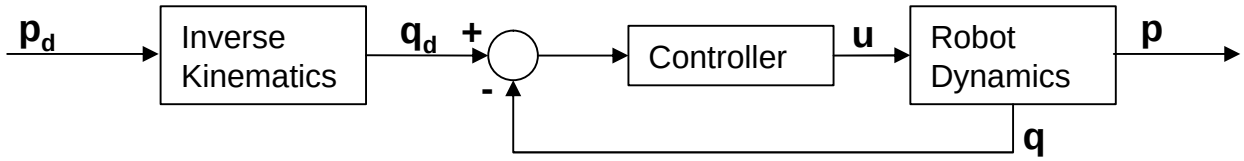


- Task Space Control



- In general, forward kinematics are “easier,” but possible singularities can occur in the Task Space Controller box. So there is an inherent tradeoff between the two approaches.

Classical [Linear] Joint Control

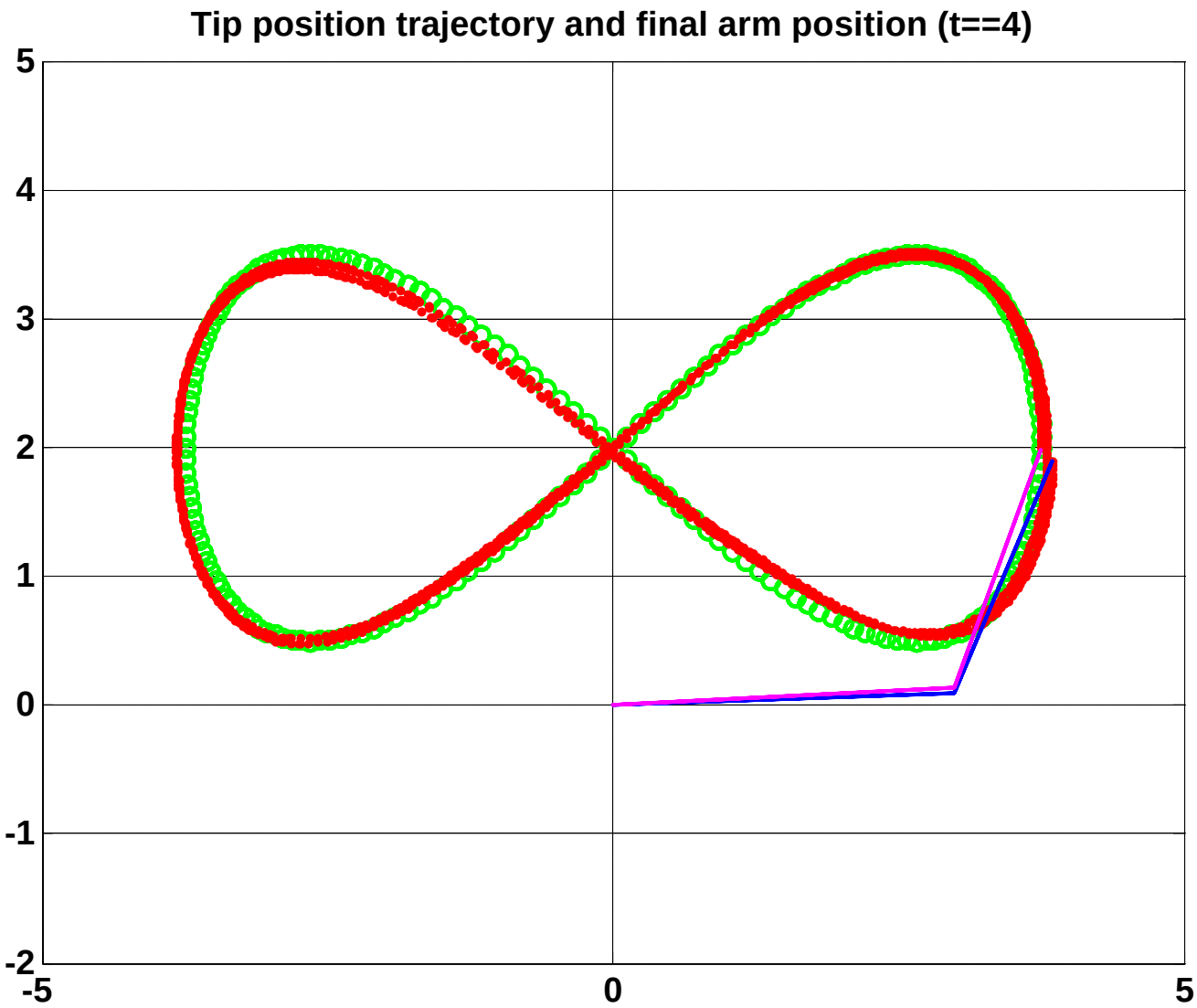


- What's in the Controller box?

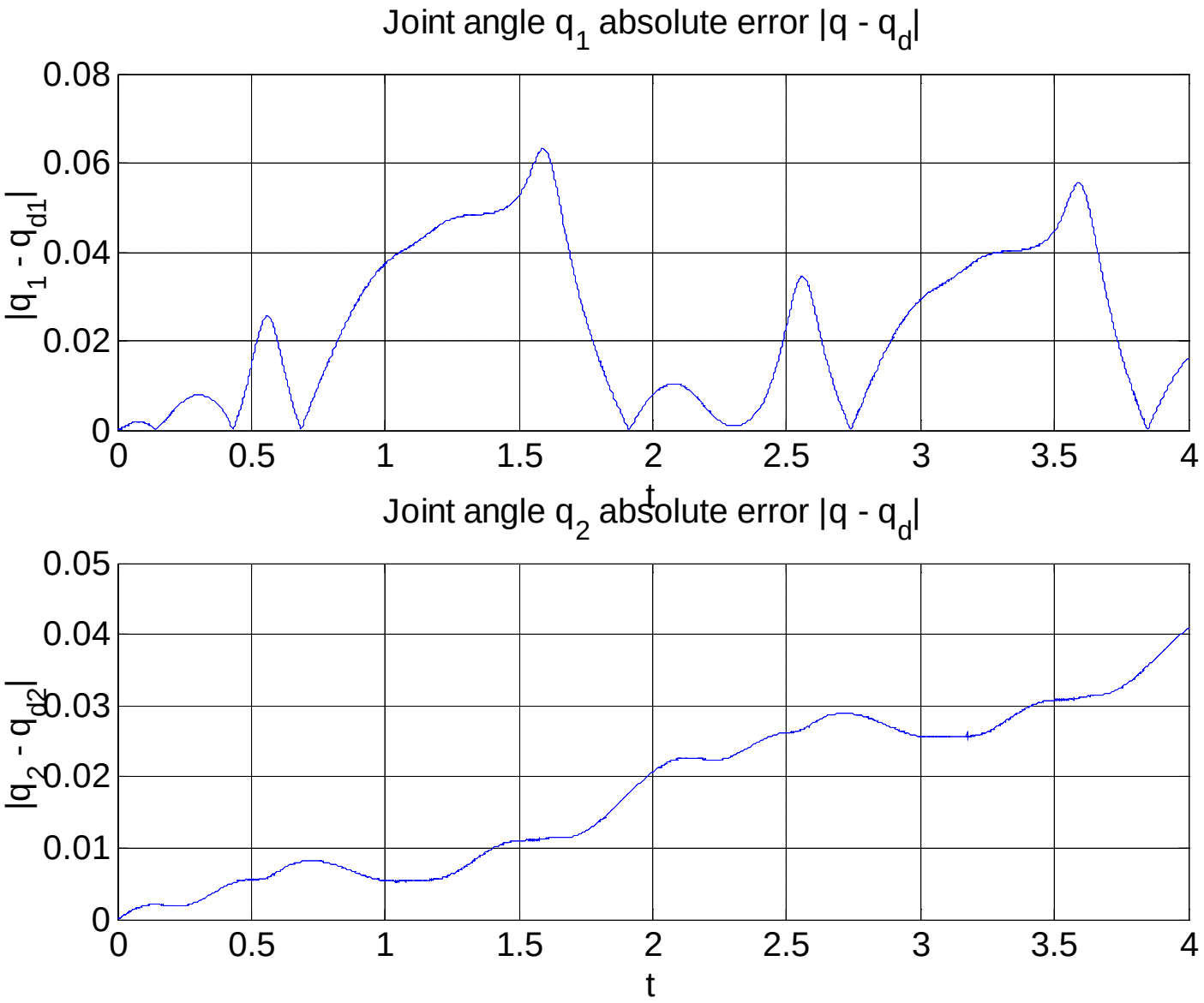
$$u = -K_p(q - q_d) - K_d(\dot{q} - \dot{q}_d) + q_d$$

- Problem: The nonlinearities of the system have not been modeled in the controller. We therefore have no guarantee of *asymptotic* tracking of desired trajectories.
- Answer: Use Nonlinear control methods (a bit nasty)
  - Computed Torque Control
  - Passivity Based Control
  - Adaptive Passivity Based Control - here, we separate out the parameters (e.g. masses, lengths) from the robot dynamics and "learn" them as we go with an *additional* controller.

Demo – Linear Joint Control:

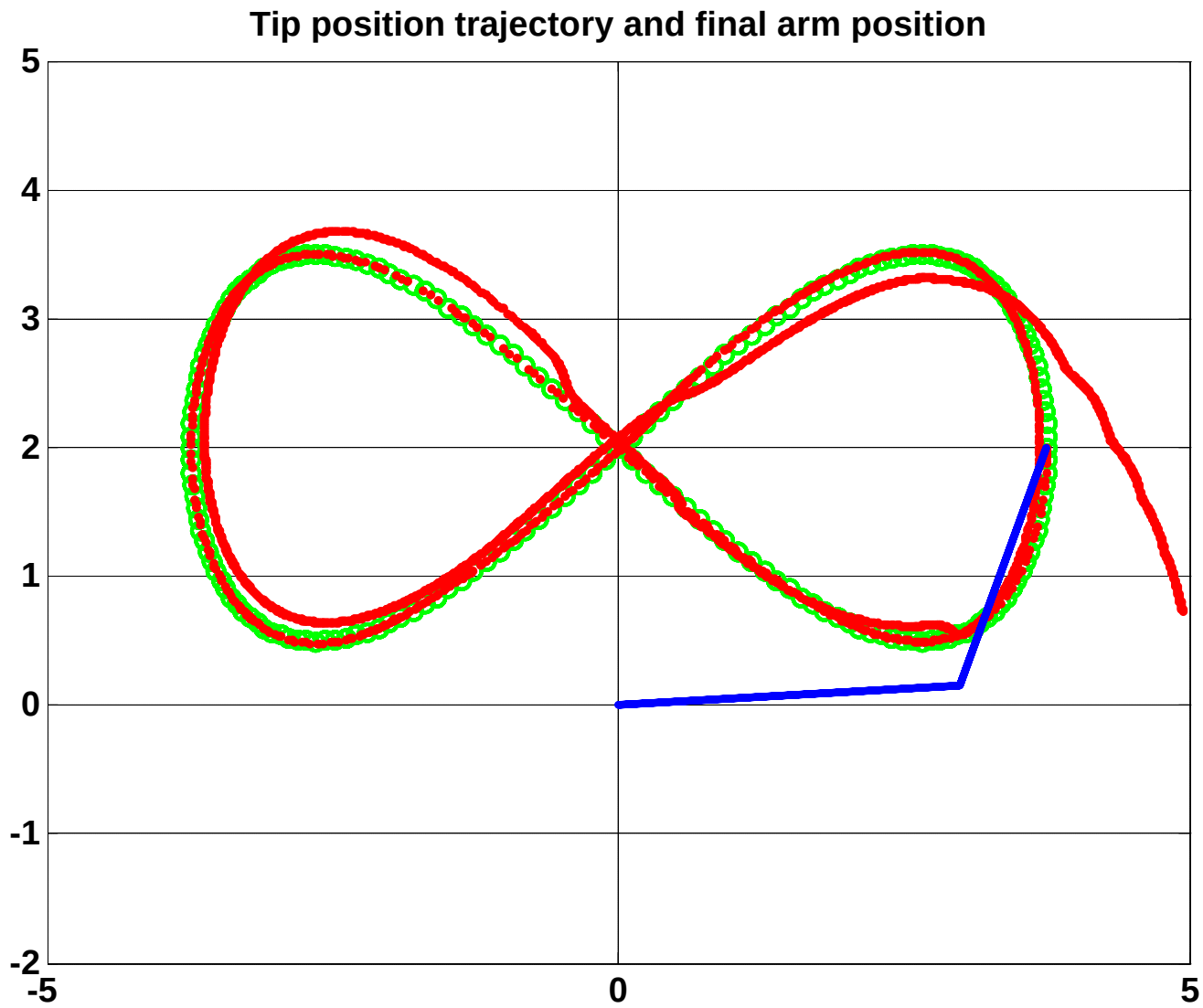


Seems good, but a closer look reveals non-decreasing error:

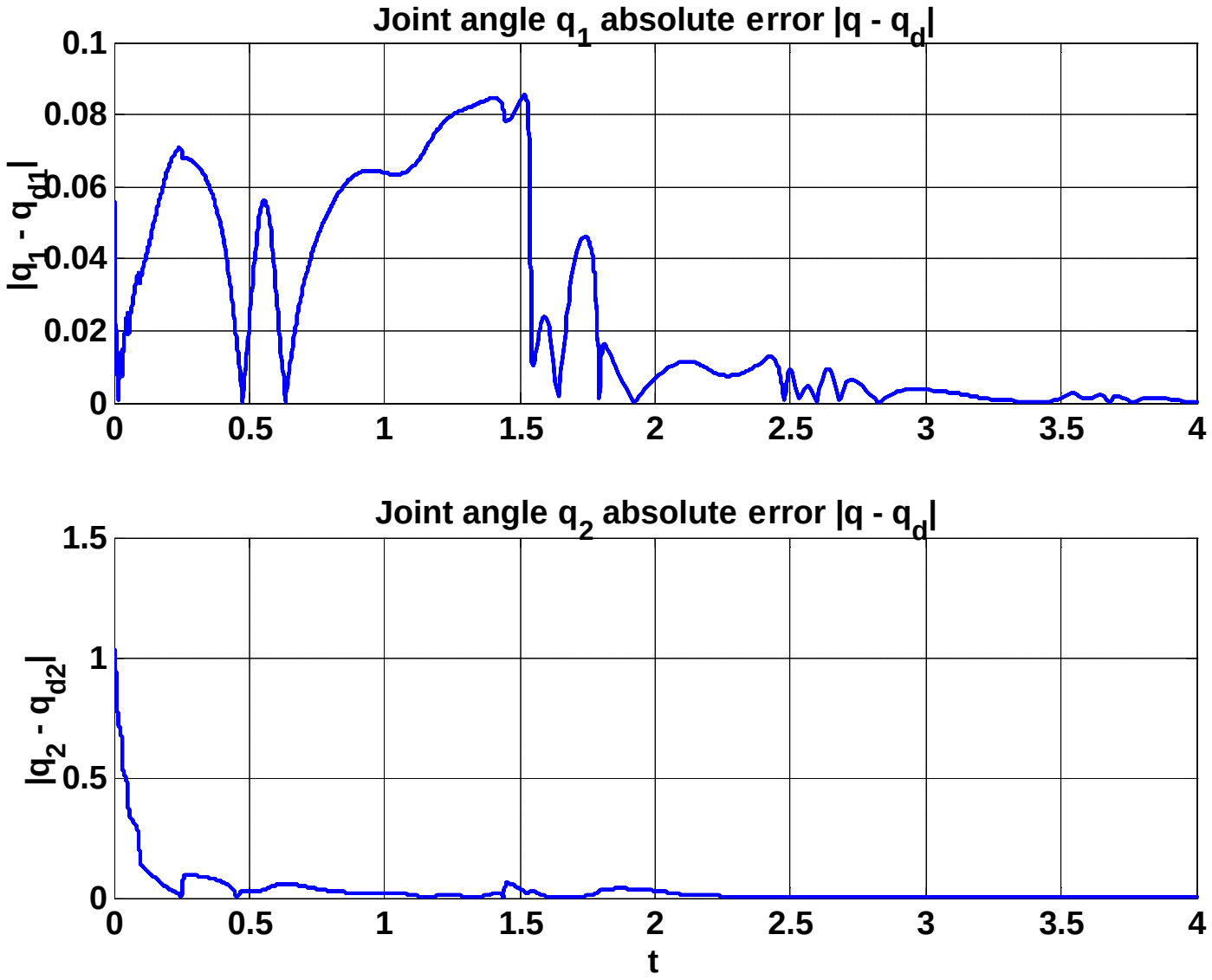


Note that the arm was even started in exactly the right state!

Demo – Adaptive Nonlinear Joint Control:



This is the kind of performance we want:



“Learning” via Adaptive Control

We see the correct mass for each link is “learned” over time:

(they were initialized at the incorrect values of 2 and 4, but converge to the true masses 1 and 1.)