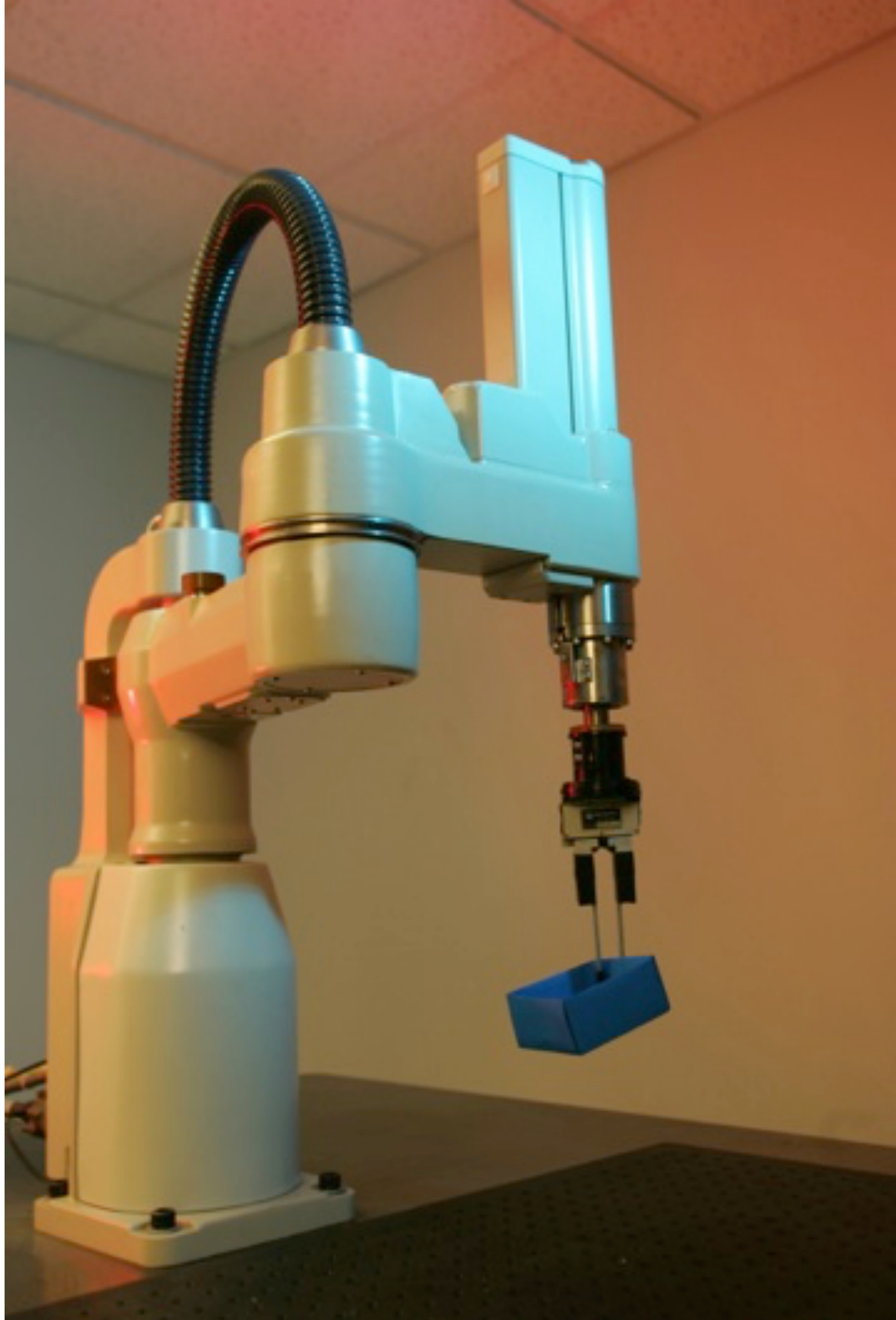
CS 3630!

Lecture 11:
Jacobians and
Trajectory Control

Motivation

- Robot arms are used extensively in industry
- They will become more prevalent in the future
  - Bottleneck is perception and grasping
  - Deep learning is revolutionizing both

- Hone your 2D geometry skills
- Introduce basic notion of control
- Use of velocity relationships in robotics

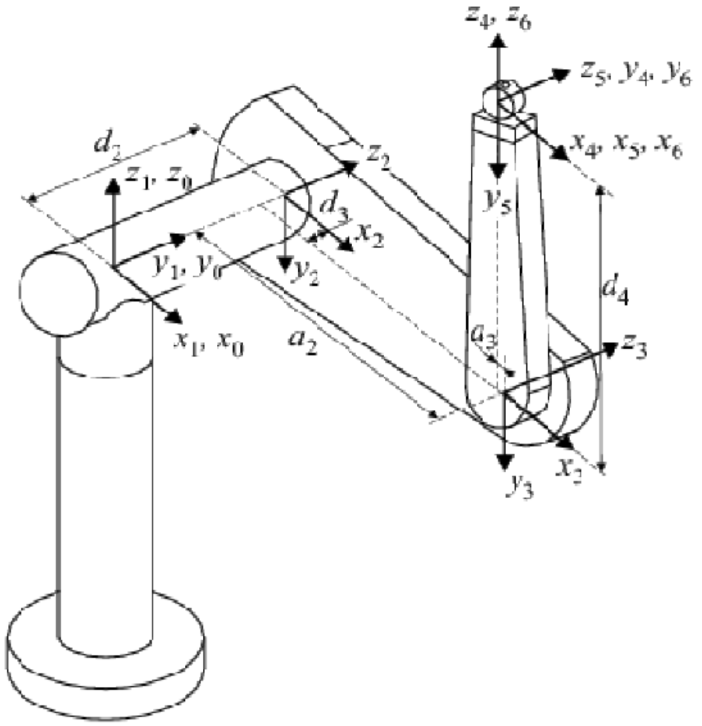


Topics

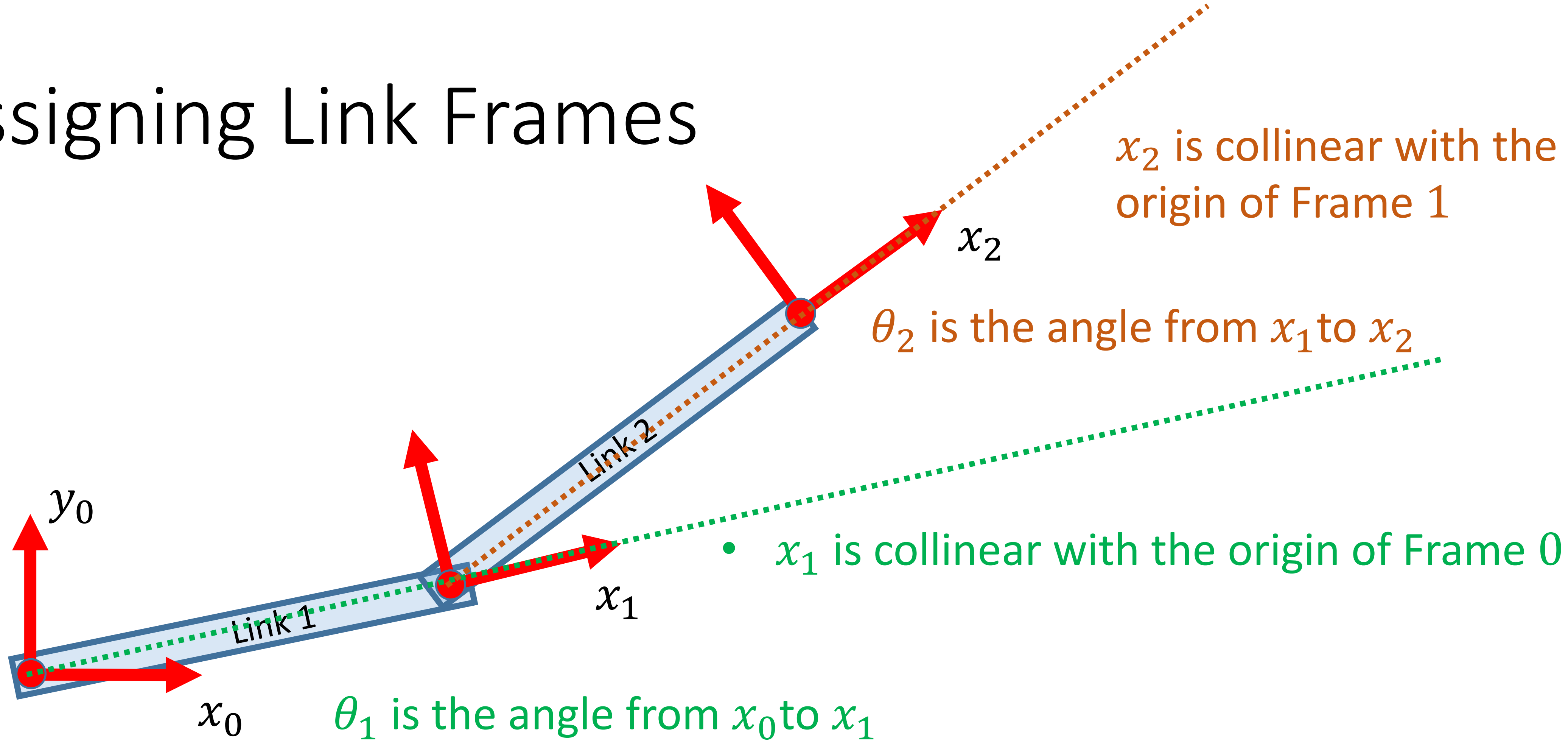
1. **Forward Kinematics Review**
2. **RRR Worked Example**
3. **Joint-Space Motion Control**
4. **The Manipulator Jacobian**
5. **Cartesian Motion Control**

1. Forward Kinematics Review

- Kinematics describes the position and motion of a robot, without considering the forces required to cause the motion.
- Forward Kinematics: *Given the value for each joint variable, q_i, determine the position and orientation of the end-effector (gripper, tool) frame.*



Assigning Link Frames



- End-effector frame T can be attached to link n in any manner that is convenient.
- In this case, $n = 2$, and we take Frame 2 to be the end-effector frame.

The Forward Kinematic Map

- The forward kinematic map gives the position and orientation of the end-effector frame as a function of the joint variables:

$$T_t^0(q) = T_1^0(q_1)\ldots T_i^{i-1}(q_i)\ldots T_n^{n-1}(q_n)T_t^n.$$

End-effector in frame n

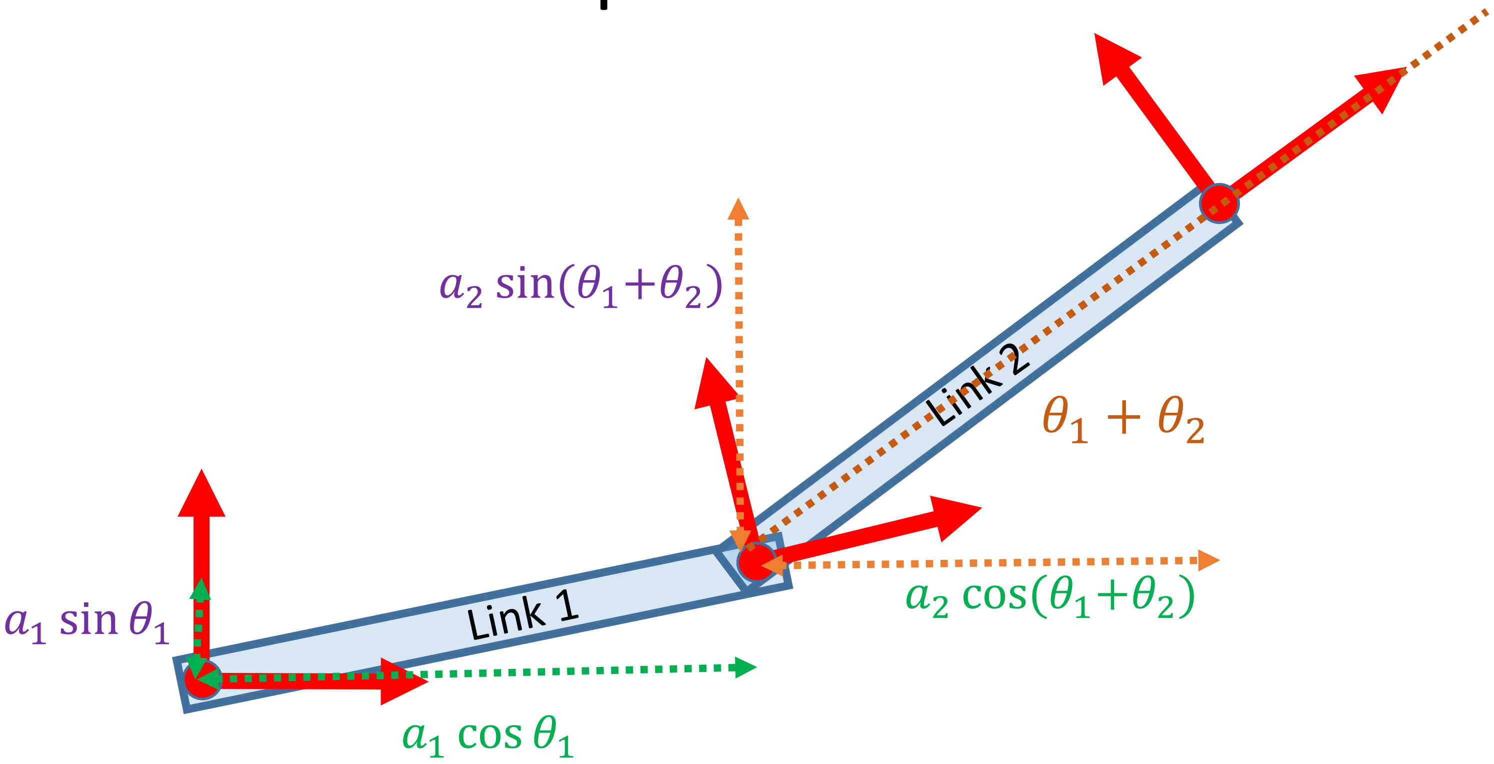
- For the two-link planar arm, we have

$$T_2^0 = T_1^0 T_2^1$$

End-effector == frame 2

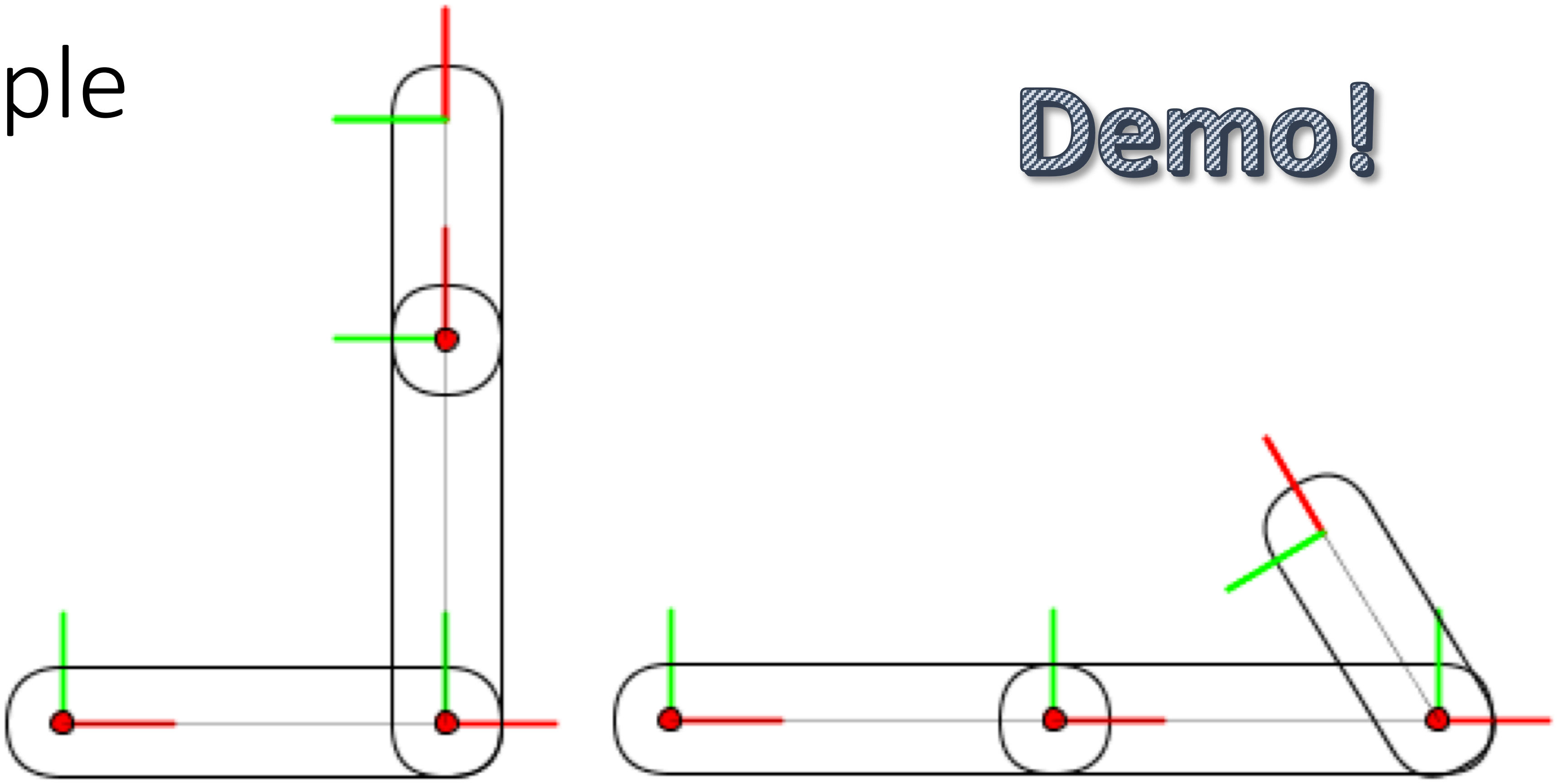
$$T_2^0 = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & a_1\cos\theta_1 \\ \sin\theta_1 & \cos\theta_1 & a_1\sin\theta_1 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & a_2\cos\theta_2 \\ \sin\theta_2 & \cos\theta_2 & a_2\sin\theta_2 \\ 0 & 0 & 1 \end{bmatrix}$$

2-link Example



$$T_2^0 = \begin{bmatrix} \cos(\theta_1+\theta_2) & -\sin(\theta_1+\theta_2) & a_1 \cos\theta_1 + a_2\cos(\theta_1+\theta_2) \\ \sin(\theta_1+\theta_2) & \cos(\theta_1+\theta_2) & a_1 \sin\theta_1 + a_2 \sin(\theta_1+\theta_2) \\ 0 & 0 & 1 \end{bmatrix}$$

2. RRR Example



Demo!

- Three revolute joints
- End-effector – Link 3 frame
- a_1=3.5, a_2=3.5, a_3=2

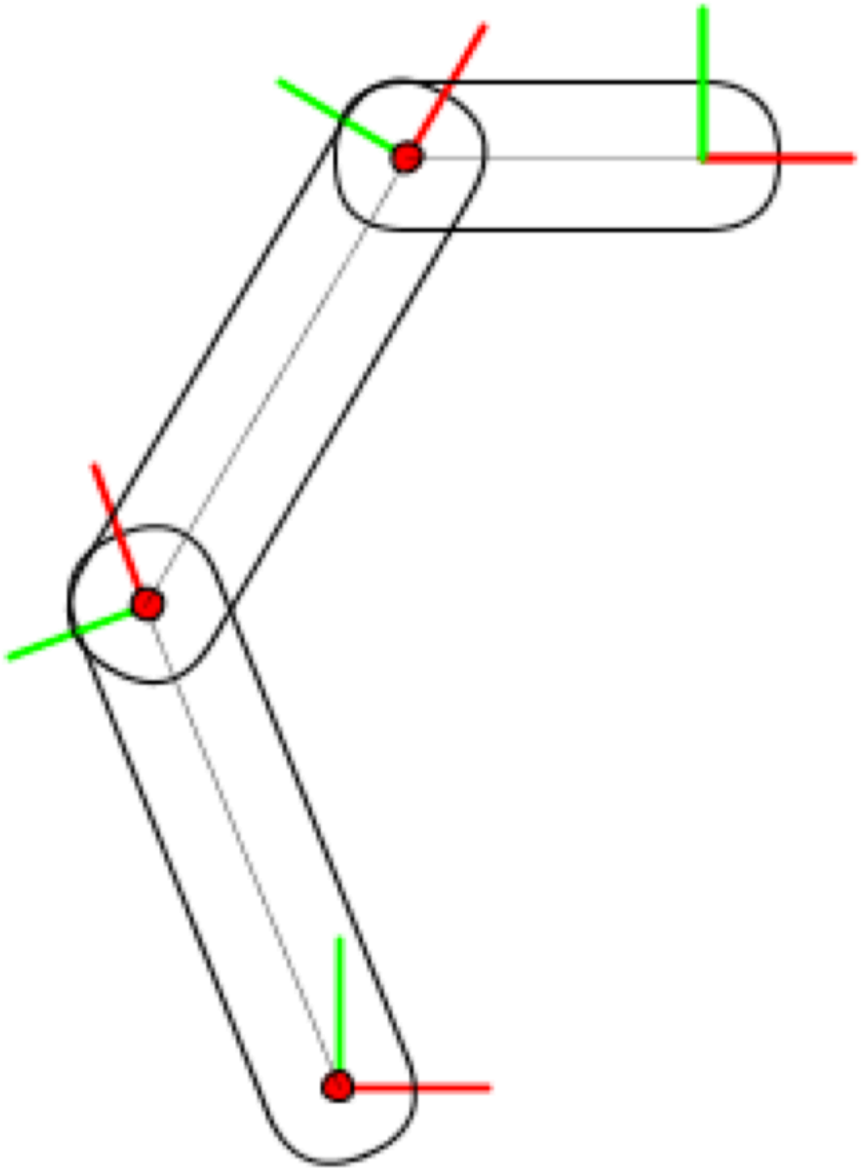
RRR example, cont'd

$$T_1^0 = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 3.5\cos\theta_1 \\ \sin\theta_1 & \cos\theta_1 & 3.5\sin\theta_1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$T_2^1 = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 3.5\cos\theta_2 \\ \sin\theta_2 & \cos\theta_2 & 3.5\sin\theta_2 \\ 0 & 0 & 1 \end{bmatrix}$$

$$T_3^2 = \begin{bmatrix} \cos\theta_3 & -\sin\theta_3 & 2\cos\theta_3 \\ \sin\theta_3 & \cos\theta_3 & 2\sin\theta_3 \\ 0 & 0 & 1 \end{bmatrix}$$

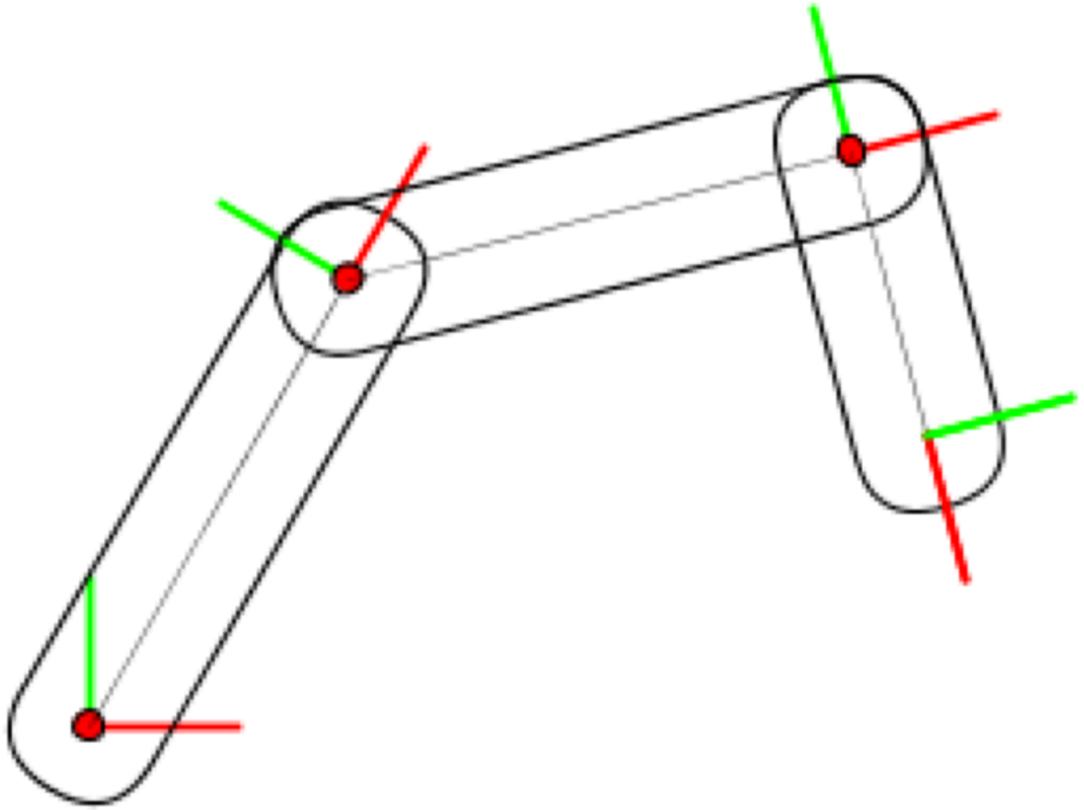
- End-effector == frame 3



(a) $\theta_1 = 112^\circ$, $\theta_2 = -52^\circ$, and $\theta_3 = -60^\circ$

RRR example, cont'd

- Multiply 3 matrices
- Note R in upper left
- Check orientation!



(b) $\theta_1 = 60^\circ$, $\theta_2 = -45^\circ$, and $\theta_3 = -90^\circ$

$$T_t^s(q) = \begin{pmatrix} \cos\beta & -\sin\beta & 3.5\cos\theta_1 + 3.5\cos\alpha + 2\cos\beta \\ \sin\beta & \cos\beta & 3.5\sin\theta_1 + 3.5\sin\alpha + 2\sin\beta \\ 0 & 0 & 1 \end{pmatrix}$$

with $\alpha = \theta_1 + \theta_2$ and $\beta = \theta_1 + \theta_2 + \theta_3$, the latter being the tool orientation.

3. Motion Control

- Trajectory following is important
  - Spray-painting
  - Sealing
  - Welding
- Three main approaches:
  - Trajectory replay
  - Joint-space Motion Control
  - Cartesian Motion Control

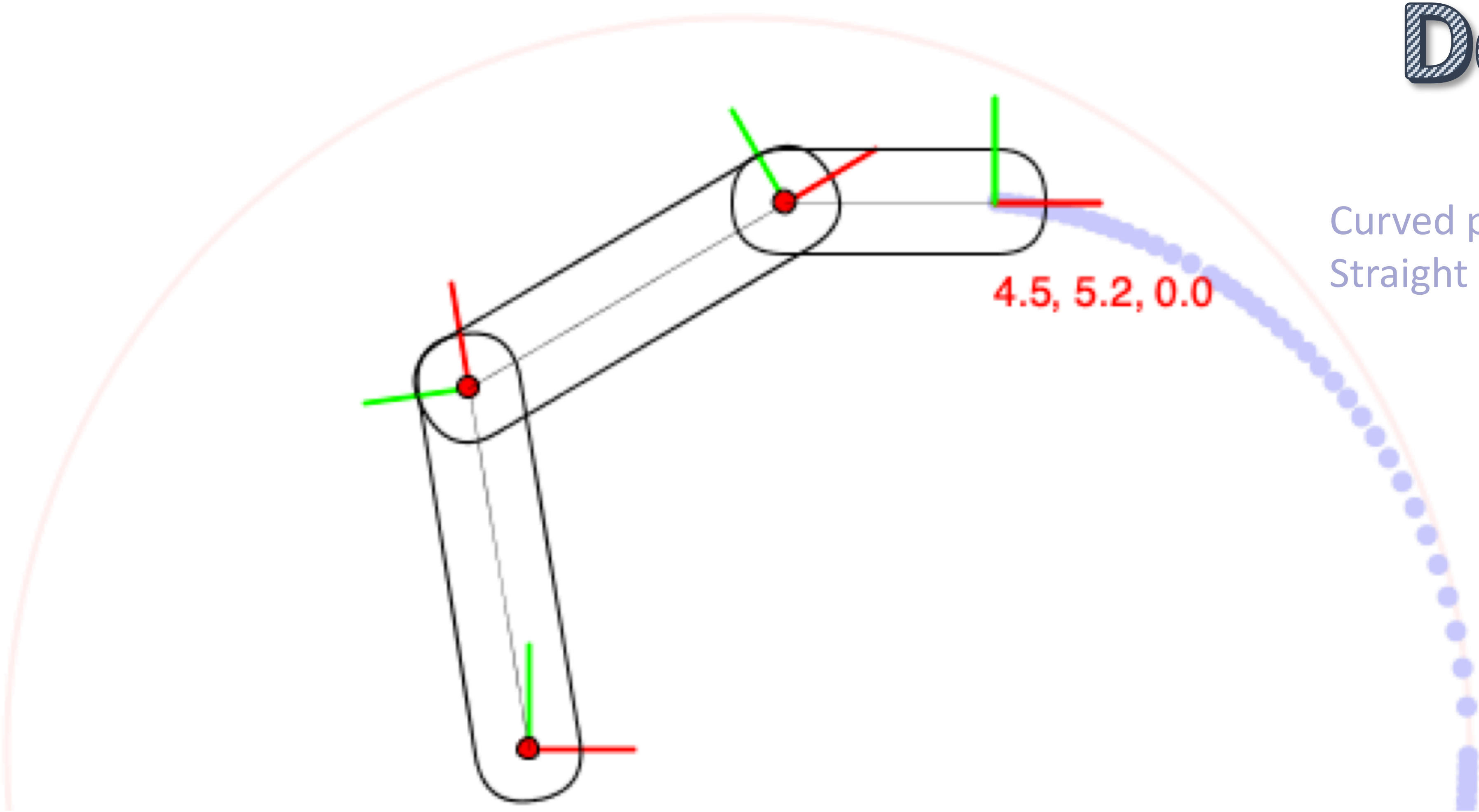
Image by Roboguru

Trajectory Replay


- Teaching by demonstration
- Define a set of **waypoints** by "showing" the robot
- Similar to keyframe animation in graphics
- Still need to interpolate between waypoints

Joint-Space Motion Control

Demo!

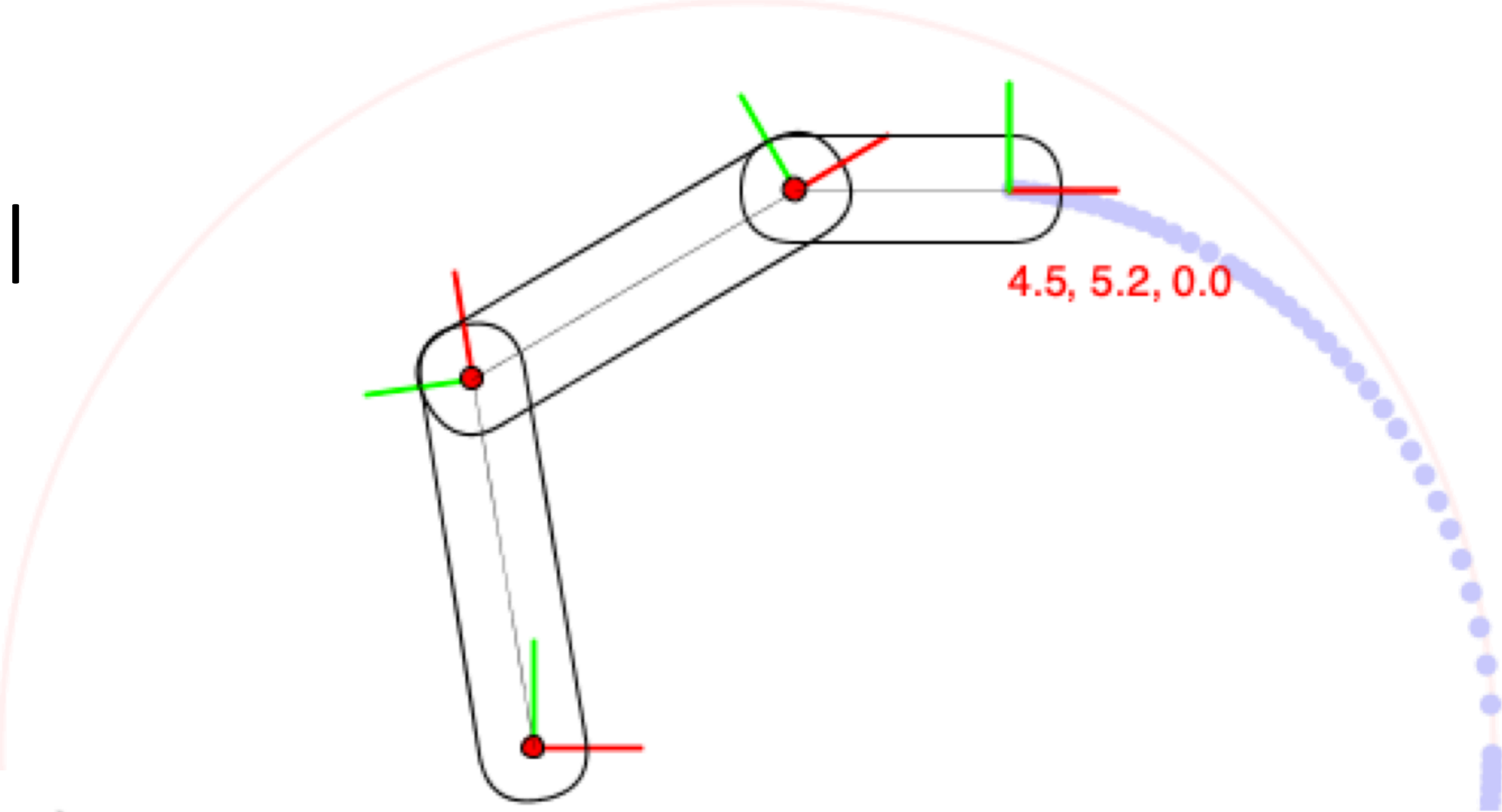


Curved path in cartesian space
Straight line in joint space

- Desired tool pose from waypoint or **inverse kinematics**[1].

[1] IK = next lecture!

Proportional Feedback Control



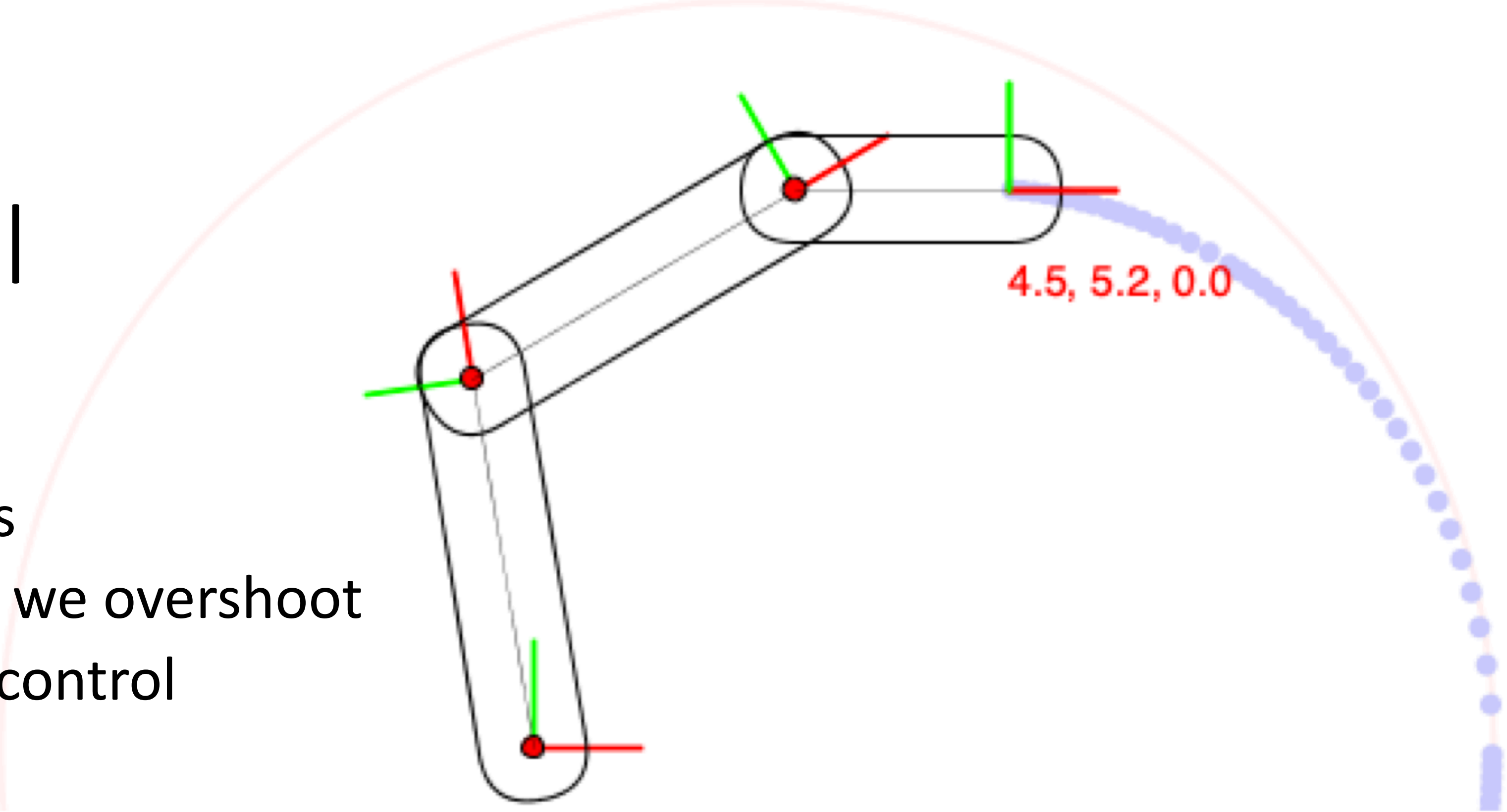
- Feedback law:

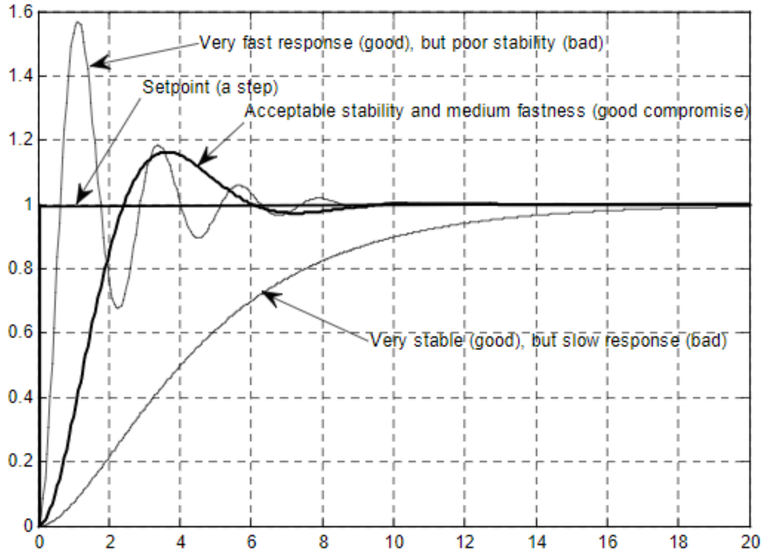
$$q_{t+1} = q_t + K_p(q_d - q_t)$$

- At every time step:
  - Calculate **joint space error** $e_t = q_d - q_t$
  - Increase of decrease proportional to e_t
  - K_p is proportional gain parameter

Proportional Feedback Control

- Properties:
  - Closer to goal -> smaller steps
  - Automatically reverses sign if we overshoot
  - Generalizes to vector-valued control
- Value of Kp really matters:
  - too high: overshoot
  - too low: slow convergence
- Special case of PID control



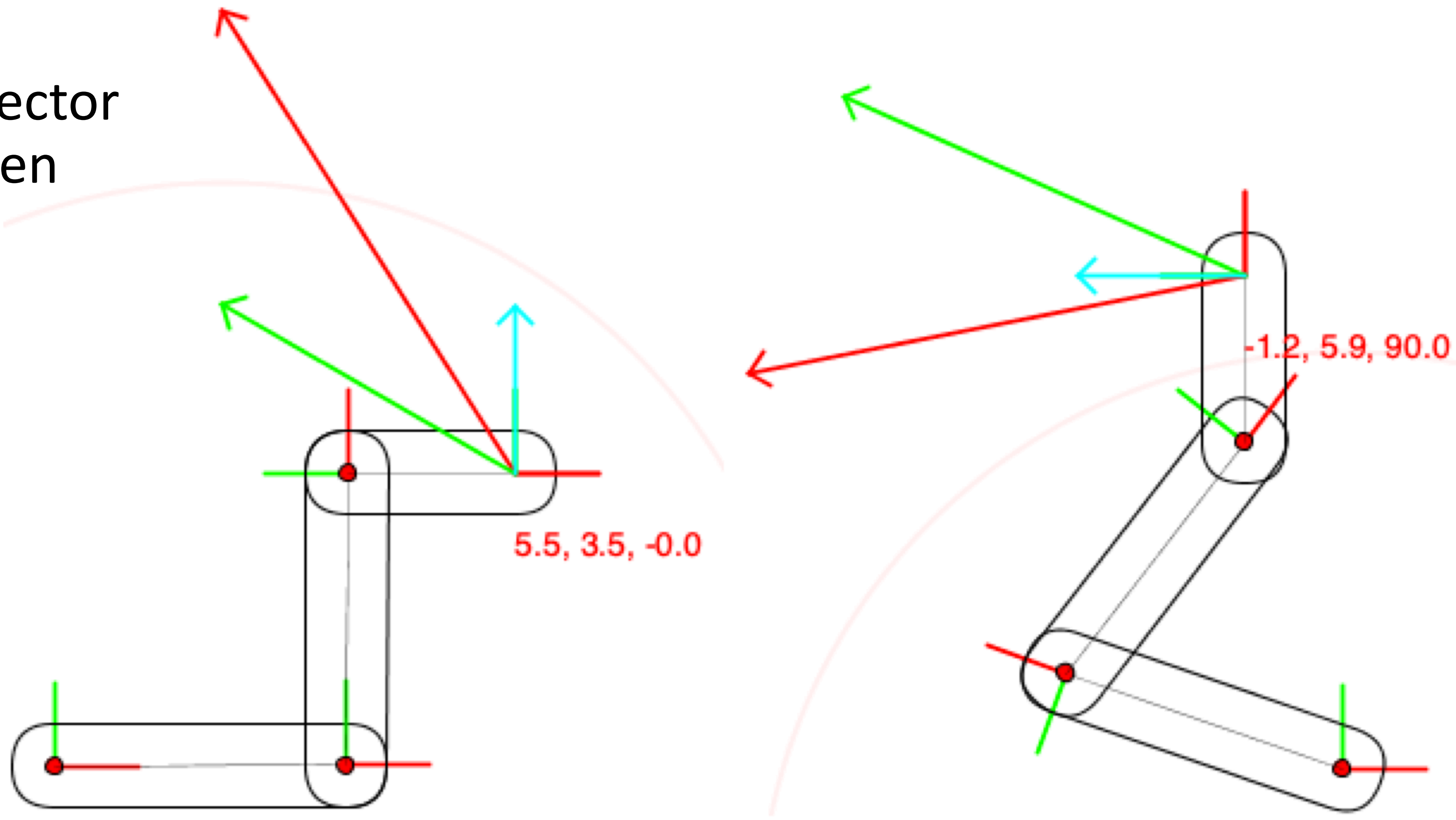


https://arduinoplusplus.wordpress.com/2017/06/21/pid-control-experiment-tuning-the-controller/

4. The Manipulator Jacobian

Demo!

- Velocity of end-effector if we move any given joint?
- Given by arrows:
- R=joint 1
- G=joint 2
- B=joint 3



Jacobian = linear map

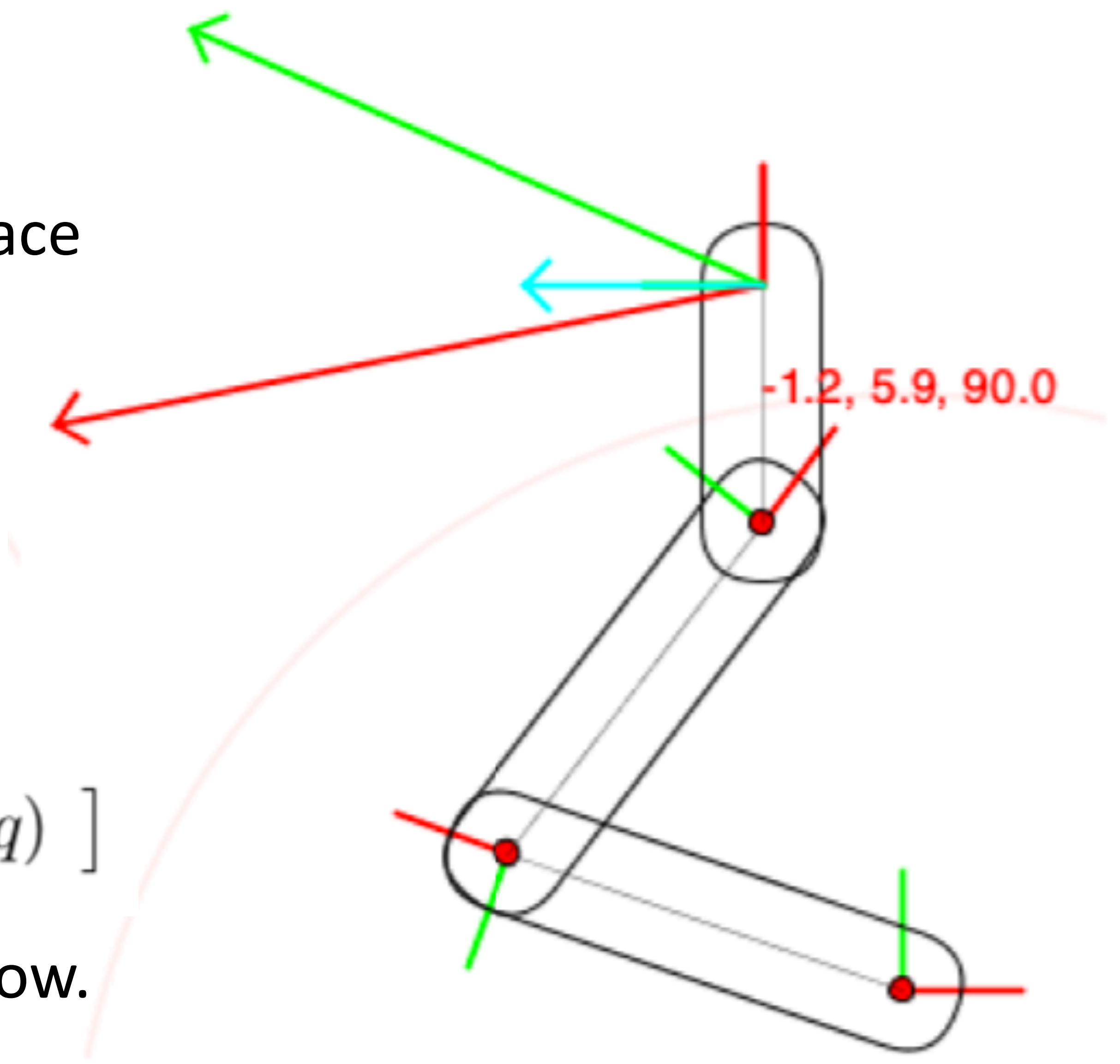
- Linear relationship between joint space velocity and cartesian velocity (pose space!)

$$[\dot{x}, \dot{y}, \dot{\theta}]^T = J(q)\dot{q}$$

- J is *3xn* matrix:

$$J(q) \triangleq \left[\begin{array}{cccc} J_1(q) & J_2(q) & \ldots & J_n(q) \end{array}\right]$$

- Each $J_i(q)$ column corresponds to arrow.
- Partial derivative of pose wrpt q_i



Worked Example: RRR manipulator

- Remember:

$$T_t^s(q) = \begin{pmatrix} \cos\beta & -\sin\beta & 3.5\cos\theta_1 + 3.5\cos\alpha + 2\cos\beta \\ \sin\beta & \cos\beta & 3.5\sin\theta_1 + 3.5\sin\alpha + 2\sin\beta \\ 0 & 0 & 1 \end{pmatrix}$$

- Extracting x, y, theta:

$$\begin{bmatrix} x(q) \\ y(q) \\ \theta(q) \end{bmatrix} = \begin{bmatrix} 3.5\cos\theta_1 + 3.5\cos\alpha + 2\cos\beta \\ 3.5\sin\theta_1 + 3.5\sin\alpha + 2\sin\beta \\ \beta \end{bmatrix}$$

- So what is Jacobian???

Worked Example: RRR manipulator

- x, y, theta:

$$\begin{bmatrix} x(q) \\ y(q) \\ \theta(q) \end{bmatrix} = \begin{bmatrix} 3.5\cos\theta_1 + 3.5\cos\alpha + 2\cos\beta \\ 3.5\sin\theta_1 + 3.5\sin\alpha + 2\sin\beta \\ \beta \end{bmatrix}$$

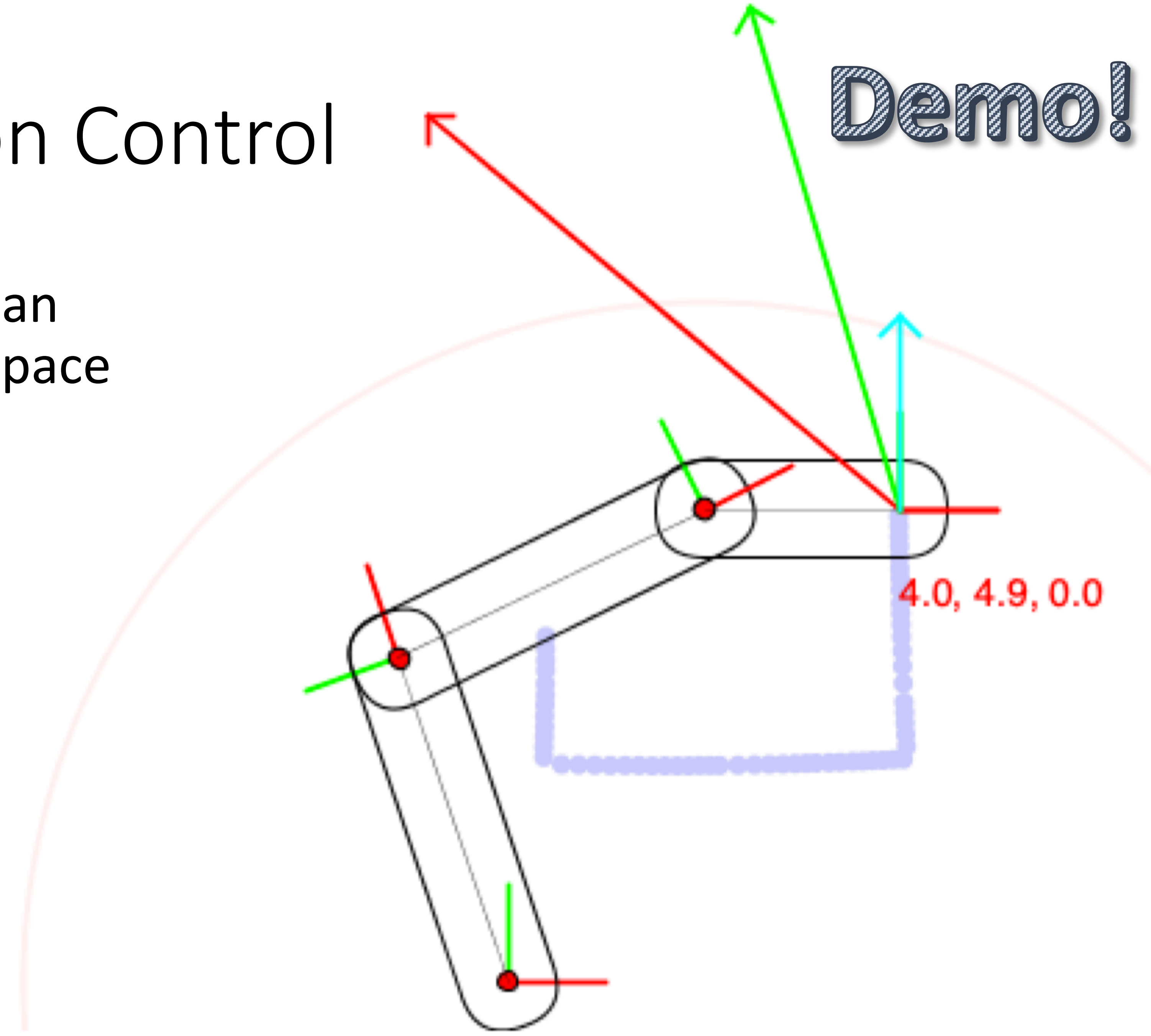
- Jacobian:

$$\begin{pmatrix} -3.5\sin\theta_1 - 3.5\sin\alpha - 2.5\sin\beta & -3.5\sin\alpha - 2.5\sin\beta & -2\sin\beta \\ 3.5\cos\theta_1 + 3.5\cos\alpha + 2.5\cos\beta & 3.5\cos\alpha + 2.5\cos\beta & 2\cos\beta \\ 1 & 1 & 1 \end{pmatrix}$$

5. Cartesian Motion Control

- Convert direction in cartesian space to direction in joint space
- Yields straight-line paths

Demo!



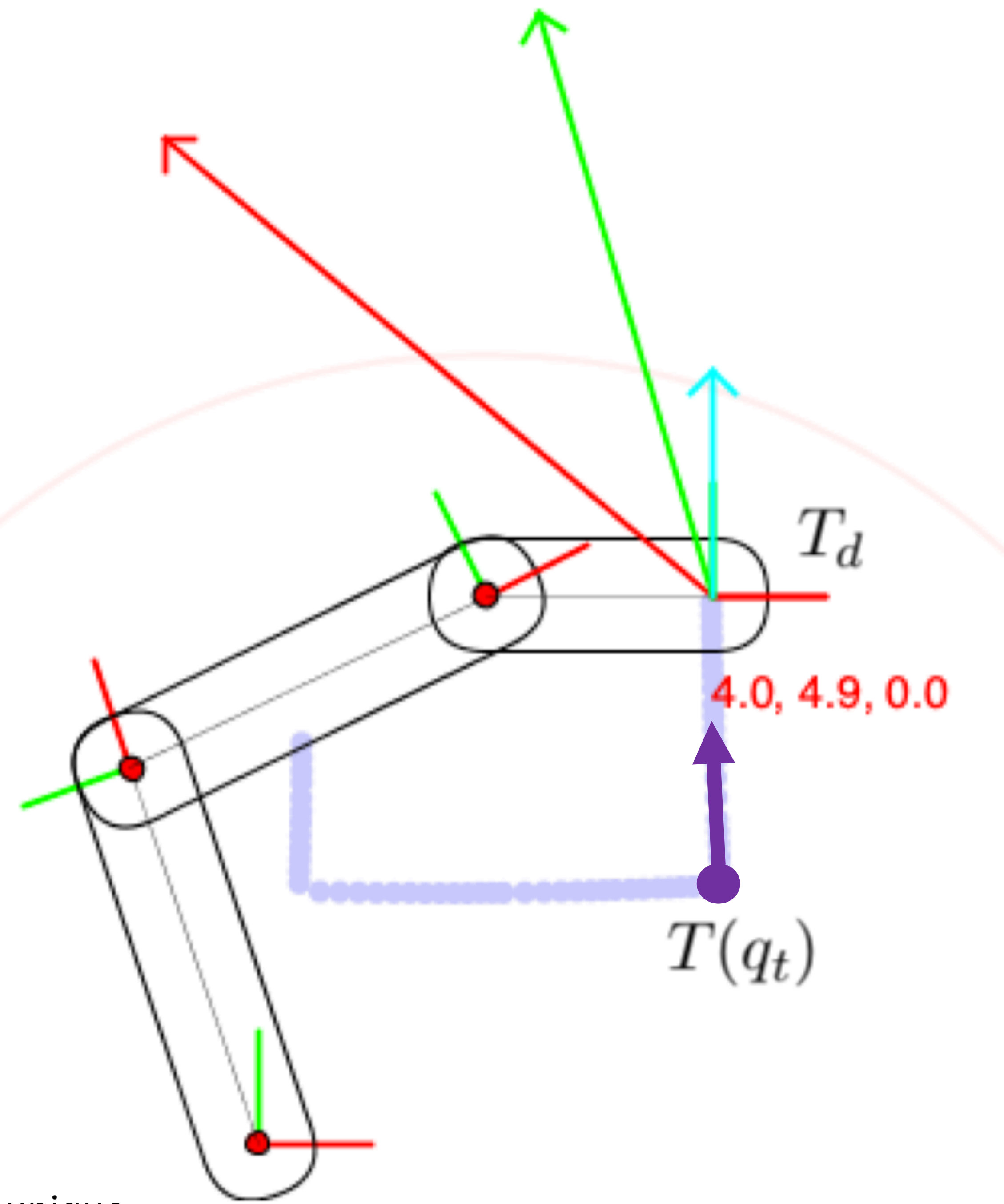
How do we convert?

- We want a straight line!
- Calculate (scaled) direction of the line
- Error in cartesian space:

$$E_t(q) = \begin{bmatrix} e_x \\ e_y \\ e_\theta \end{bmatrix} = \begin{bmatrix} x_d - x(q_t) \\ y_d - y(q_t) \\ \theta_d \ominus \theta(q_t) \end{bmatrix}$$

- Then, simple proportional control:

$$q_{t+1} = q_t + K_p J(q_t)^{-1} E_t(q)$$



Small print: we have to take when subtracting angles, as they are not unique

Summary

1. **Forward Kinematics** is just multiplying transforms
2. We went through an **RRR Worked Example**
3. **Joint-Space Motion Control** creates paths that minimize distance in joint space
4. The **Manipulator Jacobian** provides a relationship between cartesian and joint-space velocities/displacements
5. **Cartesian Motion Control** exploits this relationship to provide predictable paths in cartesian space