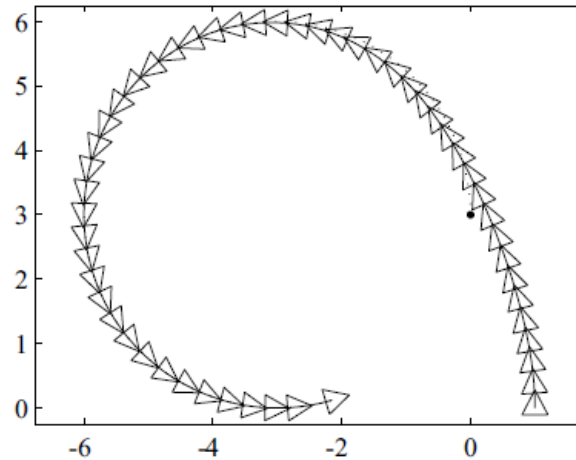


COMS W4733: Computational Aspects of Robotics

Lecture 13: Motion Control



Instructor: Tony Dear

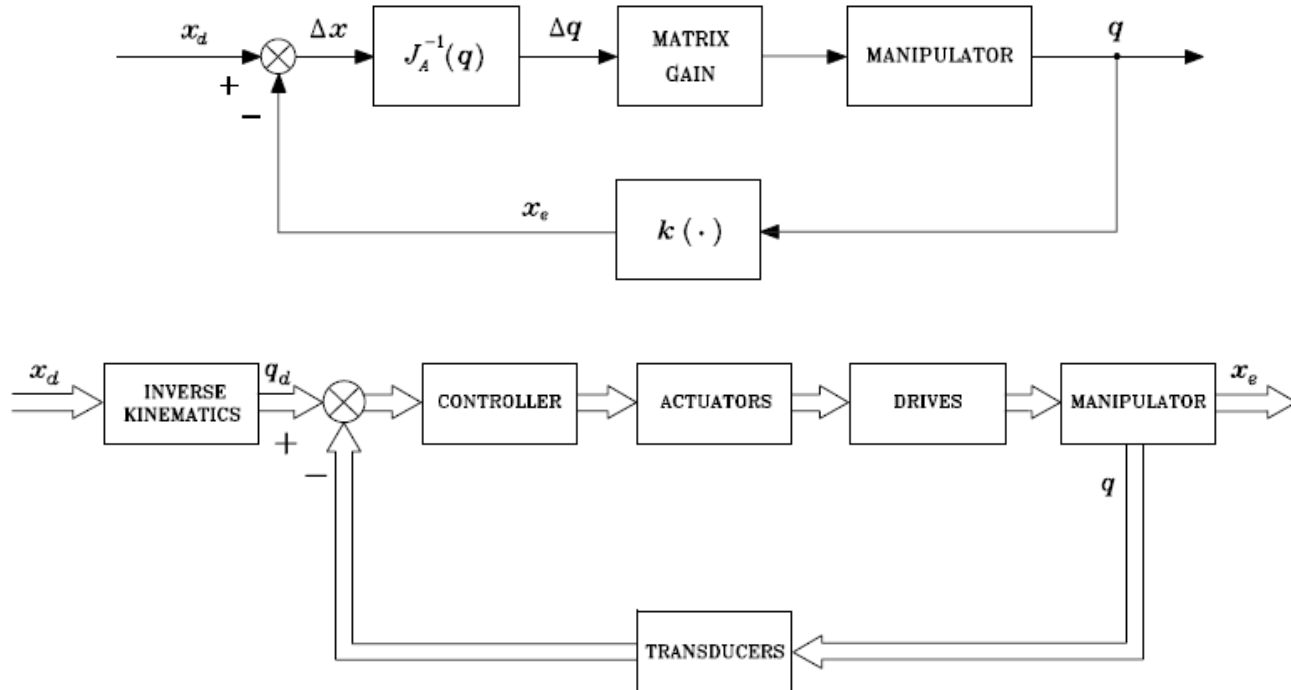
Motion Control

- Suppose we have a desired pose or trajectory that our robot should follow
- We can compute controlled input velocities using inverse kinematics
- Is that all we need to do?

- What if there is noise in what we see or do?
- What if we deviate from our path?

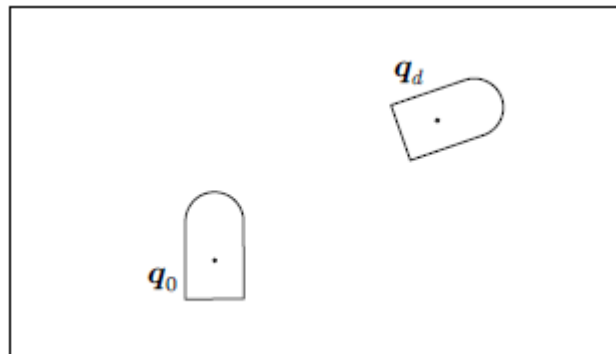
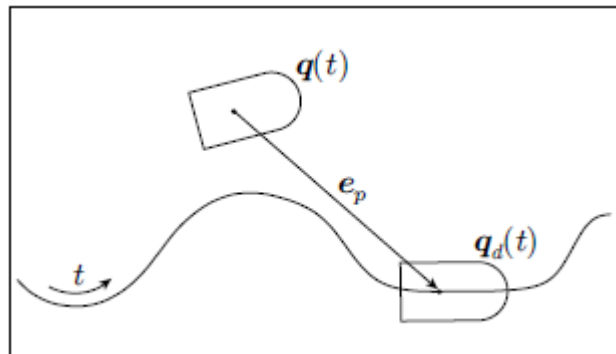
- Robot most likely has sensors—we need to complete the feedback loop!

Manipulator Control Scheme



Mobile Control Problems

- **Trajectory tracking:** *Asymptotically* follow a given trajectory $(x_d(t), y_d(t))$ from some initial configuration q_0
 - We may never reach it exactly, but we can at least minimize error over time
- **Posture regulation:** Move from some initial q_0 to a given q_d
 - Trajectory is not given, but we may be able to plan one



Unicycle Kinematics

- We derived the following:

$$\dot{x} = v \cos \theta$$

$$\dot{y} = v \sin \theta$$

$$\dot{\theta} = \omega$$

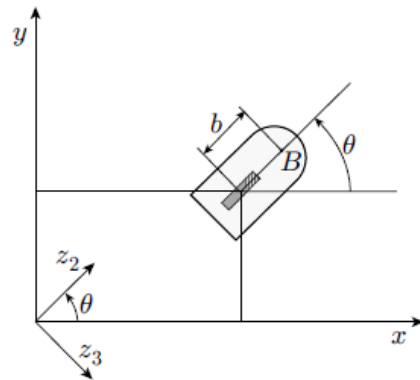


$$v_d(t) = \sqrt{\dot{x}_d(t)^2 + \dot{y}_d(t)^2}$$

$$\omega_d(t) = \frac{\dot{x}_d(t)\ddot{y}_d(t) - \ddot{x}_d(t)\dot{y}_d(t)}{\dot{x}_d(t)^2 + \dot{y}_d(t)^2}$$

- Suppose robot is currently at $\mathbf{q}(t) = (x, y, \theta)^T$ and wants to get to $\mathbf{q}_d(t) = (x_d, y_d, \theta_d)^T$
- Represent the error $\mathbf{q}_d - \mathbf{q}$ in robot's body frame:

$$\mathbf{e} = \begin{bmatrix} e_x \\ e_y \\ e_\theta \end{bmatrix} \begin{bmatrix} c_\theta & s_\theta & 0 \\ -s_\theta & c_\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_d - x \\ y_d - y \\ \theta_d - \theta \end{bmatrix}$$



Error Dynamics

- How does the error change over time?

$$\mathbf{e} = \begin{bmatrix} e_x \\ e_y \\ e_\theta \end{bmatrix} \begin{bmatrix} c_\theta & s_\theta & 0 \\ -s_\theta & c_\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_d - x \\ y_d - y \\ \theta_d - \theta \end{bmatrix} \quad \longrightarrow \quad \dot{\mathbf{e}} = \begin{bmatrix} \dot{e}_x \\ \dot{e}_y \\ \dot{e}_\theta \end{bmatrix} = \begin{bmatrix} v_d \cos e_\theta - v + e_y \omega \\ v_d \sin e_\theta - e_x \omega \\ \omega_d - \omega \end{bmatrix}$$

- This is a *nonlinear dynamical system* of the form $\dot{\mathbf{e}}(t) = \mathbf{f}(\mathbf{e}(t), \mathbf{u}(t))$
- We want to find controllers $\mathbf{u}(t) = (v(t), \omega(t))$ to drive $\mathbf{e}(t)$ to 0
- Unfortunately, each component has dependencies on other components

Linearization

- If this were a linear system it'd be easier to analyze
 - Idea: *Linearize* $\dot{\mathbf{e}}(t) = \mathbf{f}(\mathbf{e}(t), \mathbf{u}(t))$ about $\mathbf{e}_d, \mathbf{u}_d$
- $$\dot{\mathbf{e}} = \begin{bmatrix} \dot{e}_x \\ \dot{e}_y \\ \dot{e}_\theta \end{bmatrix} = \begin{bmatrix} v_d \cos e_\theta - v + e_y \omega \\ v_d \sin e_\theta - e_x \omega \\ \omega_d - \omega \end{bmatrix}$$

$$\dot{\mathbf{e}}(t) = \mathbf{f}(\mathbf{e}(t), \mathbf{u}(t))$$

$$\dot{\mathbf{e}}(t) = \mathbf{A}(t)\mathbf{e}(t) + \mathbf{u}(t)$$



$$\mathbf{A}(t) = \left[\frac{\partial \mathbf{f}}{\partial \mathbf{e}} \right] \bigg|_{\mathbf{e}=0, \mathbf{u}=\mathbf{u}_d}$$

$$\mathbf{B}(t) = \left[\frac{\partial \mathbf{f}}{\partial \mathbf{u}} \right] \bigg|_{\mathbf{e}=0, \mathbf{u}=\mathbf{u}_d}$$

$$\mathbf{B}(t) = \left[\begin{array}{cc} -1 & e_y \\ \sin e_\theta & -e_x \\ 0 & -1 \end{array} \right] \bigg|_{\mathbf{e}=0, \mathbf{u}=\mathbf{u}_d} = \left[\begin{array}{cc} -1 & 0 \\ 0 & 0 \\ 0 & -1 \end{array} \right]$$

$$\mathbf{A}(t) = \left[\begin{array}{ccc} 0 & \omega & -v_d \sin e_\theta \\ -\omega & 0 & v_d \cos e_\theta \\ 0 & 0 & 0 \end{array} \right] \bigg|_{\mathbf{e}=0, \mathbf{u}=\mathbf{u}_d} = \left[\begin{array}{ccc} 0 & \omega_d & 0 \\ -\omega_d & 0 & v_d \\ 0 & 0 & 0 \end{array} \right]$$

Closed-Loop Dynamics

$$\dot{\mathbf{e}} = \begin{bmatrix} 0 & \omega_d & 0 \\ -\omega_d & 0 & v_d \\ 0 & 0 & 0 \end{bmatrix} \mathbf{e} + \begin{bmatrix} -1 & 0 \\ 0 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$$

- This is an ODE system in the form $\dot{\mathbf{x}}(t) = \mathbf{A}(t)\mathbf{x}(t) + \mathbf{B}(t)\mathbf{u}(t)$
- $\mathbf{u}(t)$ are just the input velocities, so we can pick them arbitrarily
- We can choose a set of *linear feedback controls* s.t.

$$\begin{aligned} v(t) &= k_1 e_x(t) \\ \omega(t) &= k_2 e_y(t) + k_3 e_\theta(t) \end{aligned} \quad \Rightarrow \quad \dot{\mathbf{e}}(t) = \tilde{\mathbf{A}}(t)\mathbf{e}(t) = \begin{bmatrix} -k_1 & \omega_d & 0 \\ -\omega_d & 0 & v_d \\ 0 & -k_2 & -k_3 \end{bmatrix} \mathbf{e}(t)$$

When does this go to 0?

Linear Stability

$$\dot{\mathbf{e}}(t) = \tilde{\mathbf{A}}\mathbf{e}(t) = \begin{bmatrix} -k_1 & \omega_d & 0 \\ -\omega_d & 0 & v_d \\ 0 & -k_2 & -k_3 \end{bmatrix} \mathbf{e}(t)$$

- No general solution to the above *non-autonomous linear system* if $\tilde{\mathbf{A}}$ is function of t
- What if $\tilde{\mathbf{A}}$ is constant? Closed-form solution exists for the *autonomous linear system*

$$\mathbf{e}(t) = \exp(\tilde{\mathbf{A}}t)\mathbf{e}(t_0) = \sum_{i=1}^n c_i \exp(\lambda_i t) \mathbf{v}_i$$

λ_i are the eigenvalues of $\tilde{\mathbf{A}}$
 \mathbf{v}_i are the eigenvectors of $\tilde{\mathbf{A}}$

- $\mathbf{e}(t)$ goes to 0 (system is **stable**) if all eigenvalues λ_i have negative real parts

Linear Stability

$$\dot{\mathbf{e}}(t) = \tilde{\mathbf{A}}\mathbf{e}(t) = \begin{bmatrix} -k_1 & \omega_d & 0 \\ -\omega_d & 0 & v_d \\ 0 & -k_2 & -k_3 \end{bmatrix} \mathbf{e}(t)$$

- We select k_1 , k_2 , and k_3 (and thus v and ω) so that eigenvalues of $\tilde{\mathbf{A}}$ are negative
- Eigenvalues are given by roots of characteristic polynomial:

$$p(\lambda) = \det \begin{bmatrix} -k_1 - \lambda & \omega_d & 0 \\ -\omega_d & -\lambda & v_d \\ 0 & -k_2 & -k_3 - \lambda \end{bmatrix} = \lambda(\lambda + k_1)(\lambda + k_3) + \omega_d^2(\lambda + k_3) + v_d k_2 (\lambda + k_1)$$

- Suppose we pick $k_1 = k_3 = k, k > 0$
 $k_2 = \frac{a^2 - \omega_d^2}{v_d}, a > 2k$
 $p(\lambda) = (\lambda + k)(\lambda^2 + k\lambda + a^2)$ guaranteed to have roots with negative real part

General Linear System Control

- Many systems can be made linear into the form $\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t)$
- We can *close the loop* by choosing $\mathbf{u}(t)$ as a function of $\mathbf{x}(t)$: $\mathbf{u}(t) = -\mathbf{K}\mathbf{x}(t)$
- We now have a *homogeneous* linear system of the form $\dot{\mathbf{x}}(t) = (\mathbf{A} - \mathbf{K})\mathbf{x}(t) = \tilde{\mathbf{A}}\mathbf{x}(t)$
- To ensure stability, choose gains \mathbf{K} s.t. $\tilde{\mathbf{A}}$ has eigenvalues with negative real parts
- Many methods of linear control: root locus, Nyquist plots, etc.

Feedback Linearization

- If we're lucky it may be possible to transform a nonlinear system into a linear one using a direct change of coordinates instead of making a approximation
- Suppose we're interested in stabilizing the point B instead of $(x, y)^T$:

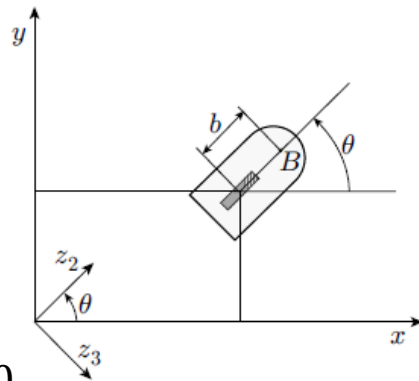
- The dynamics of B are
$$\begin{bmatrix} \dot{z}_1 \\ \dot{z}_2 \end{bmatrix} = \begin{bmatrix} c_\theta & -bs_\theta \\ s_\theta & bc_\theta \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$$

$$\begin{aligned} z_1 &= x + b \cos \theta \\ z_2 &= y + b \sin \theta \end{aligned}$$

- Thus if we transform the inputs
$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} c_\theta & -bs_\theta \\ s_\theta & bc_\theta \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$$

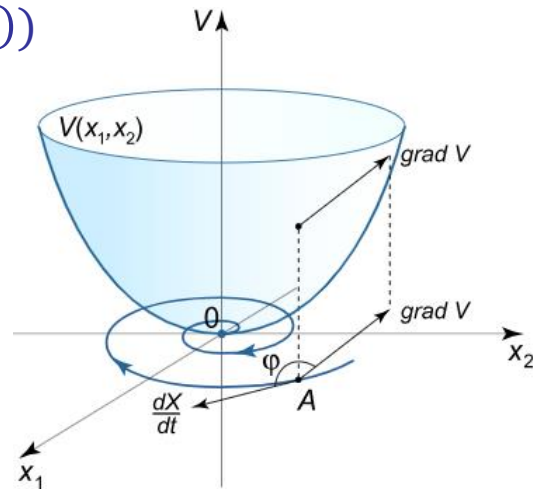
- We end up with a linear system
$$\begin{bmatrix} \dot{z}_1 \\ \dot{z}_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

- A simple linear controller:
$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} \dot{z}_{1d} + k_1(z_{1d} - z_1) \\ \dot{z}_{2d} + k_2(z_{2d} - z_2) \end{bmatrix} \quad k_1, k_2 > 0$$



Nonlinear Systems

- We *approximated* the unicycle to be a linear system
- The approximation is only good when the error is close to 0
- How can we find controllers to ensure stability of nonlinear systems $\dot{x}(t) = f(x(t))$?
- We need to satisfy the criteria of a **Lyapunov function** $V(x(t))$
 - Positive-definite away from 0: $V > 0, \forall x \neq 0$
 - Equal to 0 at 0: $V(0) = 0$
 - \dot{V} negative-definite away from 0: $\dot{V} < 0, \forall x \neq 0$
- Existence of such a function guarantees that $x(t) \rightarrow 0$



Nonlinear Unicycle Controller

- Recall the original, nonlinear dynamics of the unicycle prior to linearization:

$$\dot{\mathbf{e}} = \begin{bmatrix} \dot{e}_x \\ \dot{e}_y \\ \dot{e}_\theta \end{bmatrix} = \begin{bmatrix} v_d \cos e_\theta - v + e_y \omega \\ v_d \sin e_\theta - e_x \omega \\ \omega_d - \omega \end{bmatrix}$$

- If we use the nonlinear controller with $k_i > 0$:
$$v(t) = k_1 e_x + v_d \cos e_\theta$$
$$\omega(t) = k_2 v_d \frac{\sin e_\theta}{e_\theta} e_y + k_3 e_\theta + \omega_d$$

- The closed loop dynamics of $\mathbf{e}(t)$ become

$$\dot{\mathbf{e}} = \begin{bmatrix} e_y \omega - k_1 e_x \\ v_d \sin e_\theta - e_x \omega \\ -k_2 v_d \frac{\sin e_\theta}{e_\theta} e_y - k_3 e_\theta \end{bmatrix}$$

- We can show that this is stable!

Nonlinear Unicycle Controller

- Consider the Lyapunov function

$$V(\mathbf{e}) = \frac{1}{2}k_2(e_x^2 + e_y^2) + \frac{1}{2}e_\theta^2$$

$$\dot{\mathbf{e}} = \begin{bmatrix} e_y\omega - k_1e_x \\ v_d \sin e_\theta - e_x\omega \\ -k_2v_d \frac{\sin e_\theta}{e_\theta} e_y - k_3e_\theta \end{bmatrix}$$

- Note that $V > 0$ away from $\mathbf{e} = 0$ and $V(0) = 0$
- Check the time derivative:

$$\begin{aligned} \dot{V}(\mathbf{e}) &= k_2(e_x\dot{e}_x + e_y\dot{e}_y) + e_\theta\dot{e}_\theta \\ &= k_2(e_x(e_y\omega - k_1e_x) + e_y(v_d \sin e_\theta - e_x\omega)) + e_\theta \left(-k_2v_d \frac{\sin e_\theta}{e_\theta} e_y - k_3e_\theta \right) \\ &= -k_1k_2e_x^2 - k_3e_\theta^2 \end{aligned}$$

- $\dot{V}(\mathbf{e}) < 0$ away from $\mathbf{e} = 0$ and $\dot{V}(0) = 0$: $\mathbf{e}(t)$ will converge to 0!

Posture Regulation

- *Regulation* is the problem of driving the robot to a fixed configuration \mathbf{q}_d
- Contrast to consistently tracking the robot along a trajectory

- Regulation is actually a more difficult problem!
- Linear tracking controller required that velocity v_d be nonzero
- Feedback linearized controller could not control for orientation

- Universal controllers do not exist for nonholonomic robots!
- Contrast to manipulators, which have controllers based on inverse kinematics

Regulation for Unicycle

- Assume desired pose is the origin: $\mathbf{q}_d = (0,0,0)^T$
- Define the following polar coordinates and rewrite vehicle kinematics:

$$\rho = \sqrt{x^2 + y^2}$$

$$\gamma = \text{Atan2}(y, x) - \theta + \pi$$

$$\delta = \gamma + \theta$$



$$\dot{\rho} = -v \cos \gamma$$

$$\dot{\gamma} = \frac{v}{\rho} \sin \gamma - \omega$$

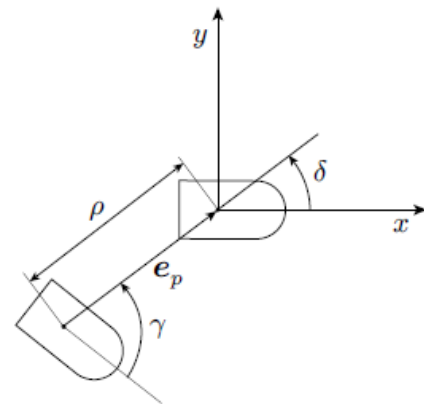
$$\dot{\delta} = \frac{v}{\rho} \sin \gamma$$

- Let's use the following feedback control with $k_i > 0$:

$$v = k_1 \rho \cos \gamma$$

$$\omega = k_2 \gamma + \frac{k_1}{\gamma} (\gamma + k_3 \delta) \sin \gamma \cos \gamma$$

- Again, we can show stability using a Lyapunov function



Regulation for Unicycle

- Consider the following Lyapunov function:

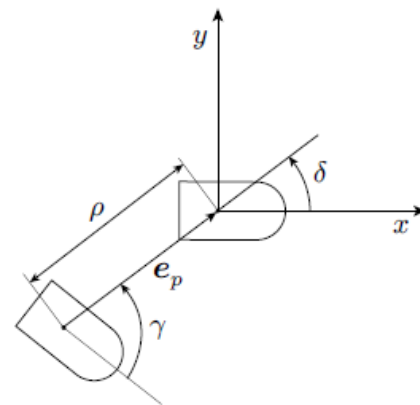
$$V = \frac{1}{2}(\rho^2 + \gamma^2 + k_3\delta^2)$$

- $V > 0$ everywhere but 0; $V(0) = 0$
- Look at its time derivative:

$$\begin{aligned}\dot{V} &= \rho\dot{\rho} + \gamma\dot{\gamma} + k_3\delta\dot{\delta} \\ &= -\rho v \cos \gamma + \gamma \left(\frac{v}{\rho} \sin \gamma - \omega \right) + k_3\delta \frac{v}{\rho} \sin \gamma \\ &= -k_1\rho^2 \cos^2 \gamma - k_2\gamma^2\end{aligned}$$

- $\dot{V} < 0$ everywhere but 0; $\dot{V}(0) = 0$

$$\begin{aligned}\dot{\rho} &= -v \cos \gamma \\ \dot{\gamma} &= \frac{v}{\rho} \sin \gamma - \omega \\ \dot{\delta} &= \frac{v}{\rho} \sin \gamma\end{aligned}$$



$$\begin{aligned}v &= k_1\rho \cos \gamma \\ \omega &= k_2\gamma + \frac{k_1}{\gamma}(\gamma + k_3\delta) \sin \gamma \cos \gamma\end{aligned}$$

Summary

- Motion control is used alongside motion planning to ensure that the robot actually ends up going where we want it to go
- Two important motion control problems: trajectory tracking, posture regulation
- Problems are inherently nonlinear, but we can linearize them near the desired control point to approximate their dynamics
- Linear stability amounts to finding negative eigenvalues
- Nonlinear stability is trickier; one way to show stability is to use a Lyapunov function