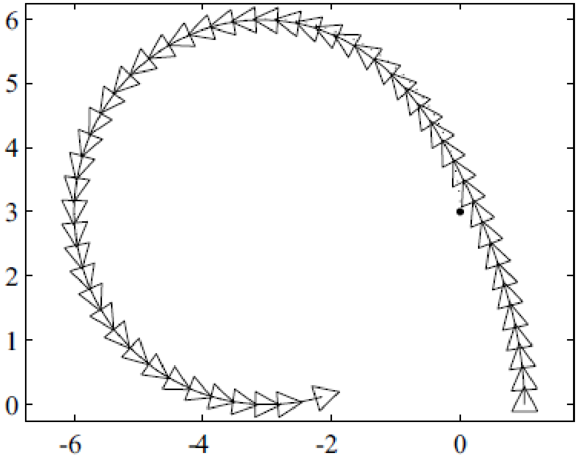
# COMS W4733: Computational Aspects of Robotics

## Lecture 13: Motion Control




Instructor: Tony Dear

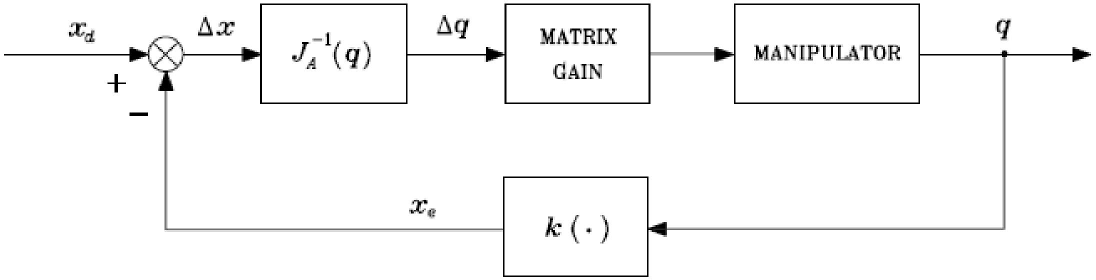
# Motion Control

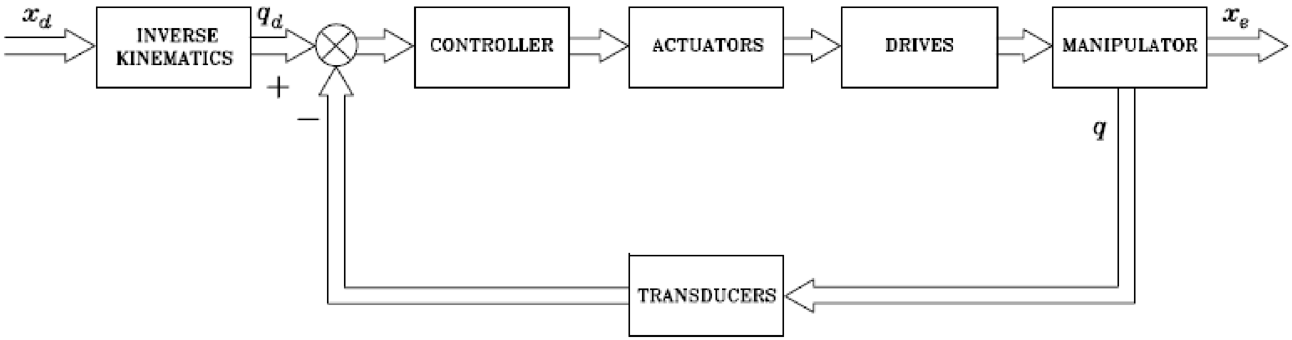
- Suppose we have a desired pose or trajectory that our robot should follow
- We can compute controlled input velocities using inverse kinematics
- Is that all we need to do?

- 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!

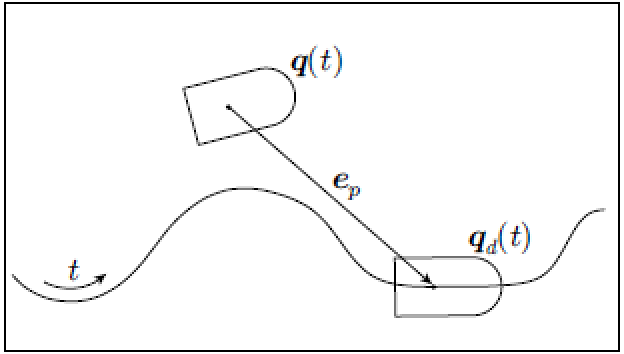
# Manipulator Control Scheme



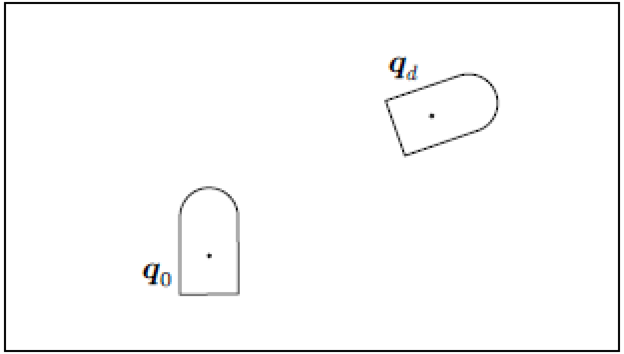



# Mobile Control Problems

- **Trajectory tracking**: *Asymptotically* follow a given trajectory $(x_d(t), y_d(t))$ from some initial configuration $\boldsymbol{q}_0$
  - We may never reach it exactly, but we can at least minimize error over time




- **Posture regulation**: Move from some initial $\boldsymbol{q}_0$ to a given $\boldsymbol{q}_d$
  - Trajectory is not given, but we may be able to plan one



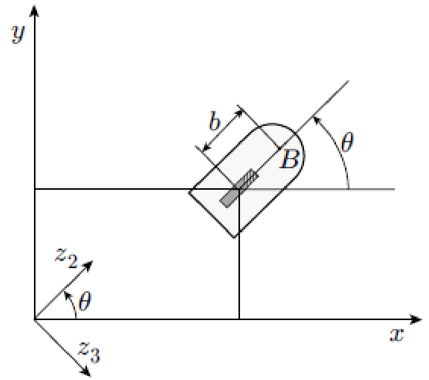
# Unicycle Kinematics

- We derived the following:

$$\dot{x} = v\cos\theta \qquad\qquad v_d(t) = \sqrt{\dot{x}_d(t)^2 + \dot{y}_d(t)^2}$$
$$\dot{y} = v\sin\theta \quad \Longrightarrow$$
$$\dot{\theta} = \omega \qquad\qquad \omega_d(t) = \frac{\dot{x}_d(t)\ddot{y}_d(t) - \ddot{x}_d(t)\dot{y}_d(t)}{\dot{x}_d(t)^2 + \dot{y}_d(t)^2}$$

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

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



# Error Dynamics

- How does the error change over time?

$$\boldsymbol{e} = \begin{bmatrix} e_x \\ e_y \\ e_\theta \end{bmatrix} \begin{bmatrix} c_\theta & s_\theta & 0 \\ -s_\theta & c_\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_d - x \\ y_d - y \\ \theta_d - \theta \end{bmatrix} \Rightarrow \dot{\boldsymbol{e}} = \begin{bmatrix} \dot{e}_x \\ \dot{e}_y \\ \dot{e}_\theta \end{bmatrix} = \begin{bmatrix} \omega_d e_y + u_1 - e_y u_2 \\ -\omega_d e_x + v_d \sin e_\theta + e_x u_2 \\ u_2 \end{bmatrix}$$

$$u_1 = v_d \cos e_\theta - v$$
$$u_2 = \omega_d - \omega$$

- This is a *nonlinear dynamical system* of the form $\dot{\boldsymbol{e}}(t) = \boldsymbol{f}(\boldsymbol{e}(t), \boldsymbol{u}(t))$
- We want to find controllers $\boldsymbol{u}(t)$ to drive $\boldsymbol{e}(t)$ 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* $\dot{\boldsymbol{e}}(t) = \boldsymbol{f}(\boldsymbol{e}(t), \boldsymbol{u}(t))$ about $\boldsymbol{e}_d, \boldsymbol{u}_d$

$$\dot{\boldsymbol{e}} = \begin{bmatrix} \omega_d e_y + u_1 - e_y u_2 \\ -\omega_d e_x + v_d \sin e_\theta + e_x u_2 \\ u_2 \end{bmatrix}$$

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

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

$$\boldsymbol{A}(t) = \left[\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{e}}\right]\Bigg|_{\boldsymbol{e}=0,\boldsymbol{u}=\boldsymbol{u}_d} \qquad \boldsymbol{B}(t) = \left[\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\right]\Bigg|_{\boldsymbol{e}=0,\boldsymbol{u}=\boldsymbol{u}_d}$$

$$\boldsymbol{B}(t) = \begin{bmatrix} 1 & -e_y \\ 0 & e_x \\ 0 & 1 \end{bmatrix}\Bigg|_{\boldsymbol{e}=0,\boldsymbol{u}=0} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\boldsymbol{A}(t) = \begin{bmatrix} 0 & \omega_d - u_2 & 0 \\ -\omega_d + u_2 & 0 & v_d \cos e_\theta \\ 0 & 0 & 0 \end{bmatrix}\Bigg|_{\boldsymbol{e}=0,\boldsymbol{u}=0} = \begin{bmatrix} 0 & \omega_d & 0 \\ -\omega_d & 0 & v_d \\ 0 & 0 & 0 \end{bmatrix}$$

# Closed-Loop Dynamics

$$\dot{\boldsymbol{e}} = \begin{bmatrix} 0 & \omega_d & 0 \\ -\omega_d & 0 & v_d \\ 0 & 0 & 0 \end{bmatrix} \boldsymbol{e} + \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

- This is an ODE system in the form $\dot{\boldsymbol{x}}(t) = \boldsymbol{A}(t)\boldsymbol{x}(t) + \boldsymbol{B}(t)\boldsymbol{u}(t)$
- $\boldsymbol{u}(t)$ are just the input velocities, so we can pick them arbitrarily

- We can choose a set of *linear feedback controls* s.t.

$$u_1(t) = -k_1 e_x(t)$$
$$u_2(t) = -k_2 e_y(t) - k_3 e_\theta(t)$$

$$\Rightarrow$$

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

When does this go to 0?

# Linear Stability

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

- No general solution to the above *non-autonomous linear system* if $\widetilde{\boldsymbol{A}}$ is function of $t$

- What if $\widetilde{\boldsymbol{A}}$ is constant? Closed-form solution exists for the *autonomous linear system*

$$\boldsymbol{e}(t) = \exp\left(\widetilde{\boldsymbol{A}}t\right)\boldsymbol{e}(t_0) = \sum_{i=1}^{n} c_i \exp(\lambda_i t)\, \boldsymbol{v}_i$$

$\lambda_i$ are the eigenvalues of $\widetilde{\boldsymbol{A}}$
$\boldsymbol{v}_i$ are the eigenvectors of $\widetilde{\boldsymbol{A}}$

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

# Linear Stability

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

- We select $k_1$, $k_2$, and $k_3$ (and thus $v$ and $\omega$) so that eigenvalues of $\widetilde{\boldsymbol{A}}$ 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 $\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t)$

- We can *close the loop* by choosing $\boldsymbol{u}(t)$ as a function of $\boldsymbol{x}(t)$: $\boldsymbol{u}(t) = -\boldsymbol{K}\boldsymbol{x}(t)$

- We now have a *homogeneous* linear system of the form $\dot{\boldsymbol{x}}(t) = (\boldsymbol{A} - \boldsymbol{K})\boldsymbol{x}(t) = \widetilde{\boldsymbol{A}}\boldsymbol{x}(t)$

- To ensure stability, choose gains $\boldsymbol{K}$ s.t. $\widetilde{\boldsymbol{A}}$ has eigenvalues with negative real parts

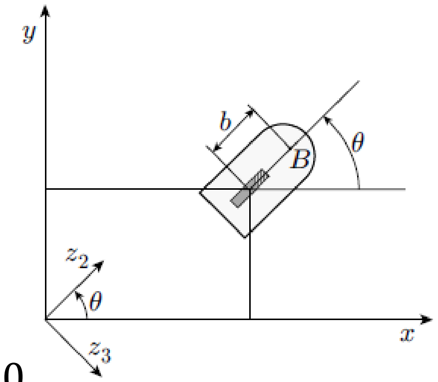
- Many methods of linear control: root locus, Nyquist plots, etc.

# 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 $(x, y)^T$:

$$z_1 = x + b\cos\theta$$
$$z_2 = y + b\sin\theta$$

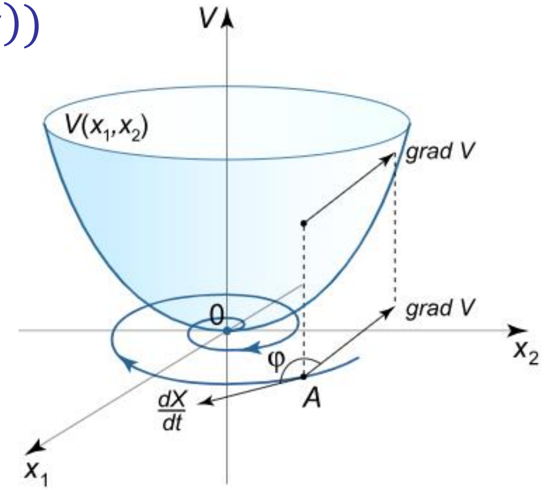
- The dynamics of $B$ are $\begin{bmatrix} \dot{z}_1 \\ \dot{z}_2 \end{bmatrix} = \begin{bmatrix} c_\theta & -bs_\theta \\ s_\theta & bc_\theta \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$
- Thus if we transform the inputs $\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} c_\theta & -bs_\theta \\ s_\theta & bc_\theta \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$
- 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 $\dot{\boldsymbol{x}}(t) = \boldsymbol{f}\big(\boldsymbol{x}(t)\big)$?
- We need to satisfy the criteria of a **Lyapunov function** $V(\boldsymbol{x}(t))$
  - Positive-definite away from 0: $V > 0, \forall \boldsymbol{x} \neq 0$
  - Equal to 0 at 0: $V(0) = 0$
  - $\dot{V}$ negative-definite away from 0: $\dot{V} < 0, \forall \boldsymbol{x} \neq 0$




- Existence of such a function guarantees that $\boldsymbol{x}(t) \to 0$

# Nonlinear Unicycle Controller

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

$$\dot{\boldsymbol{e}} = \begin{bmatrix} \dot{e}_x \\ \dot{e}_y \\ \dot{e}_\theta \end{bmatrix} = \begin{bmatrix} u_1 + e_y\omega \\ v_d \sin e_\theta - e_x\omega \\ \omega_d - \omega \end{bmatrix}$$

- 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 $\boldsymbol{e}(t)$ become

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

- We can show that this is stable!

# Nonlinear Unicycle Controller

- Consider the Lyapunov function

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

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

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

$$\begin{aligned} \dot{V}(\boldsymbol{e}) &= k_2\left(e_x\dot{e}_x + e_y\dot{e}_y\right) + e_\theta\dot{e}_\theta \\ &= k_2\left(e_x(e_y\omega - k_1 e_x) + e_y(v_d \sin e_\theta - e_x\omega)\right) + e_\theta\left(-k_2 v_d \frac{\sin e_\theta}{e_\theta} e_y - k_3 e_\theta\right) \\ &= -k_1 k_2 e_x^2 - k_3 e_\theta^2 \end{aligned}$$

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

# Posture Regulation

- *Regulation* is the problem of driving the robot to a fixed configuration $\boldsymbol{q}_d$
- 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

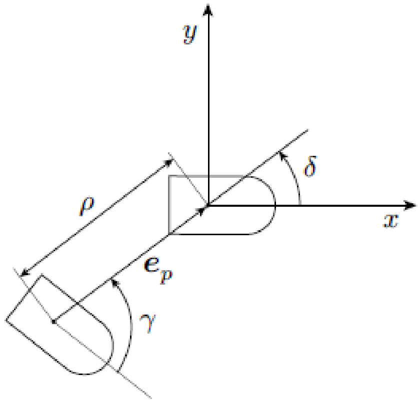
# Regulation for Unicycle

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

$$\rho = \sqrt{x^2 + y^2}$$
$$\gamma = \mathrm{Atan2}(y, x) - \theta + \pi$$
$$\delta = \gamma + \theta$$

$$\Rightarrow$$

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




- 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:

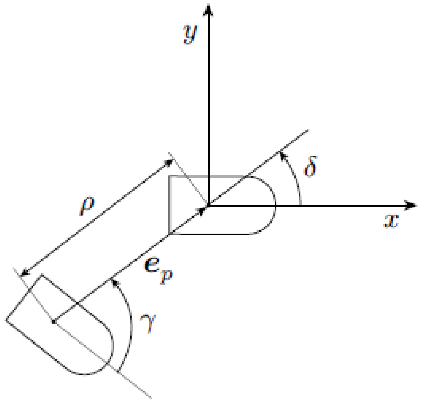
$$V = \frac{1}{2}(\rho^2 + \gamma^2 + k_3\delta^2)$$

- $V > 0$ 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