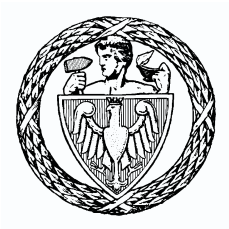
Mobile Robots

Motion Control of Wheeled Mobile Robots



Lecture 6

Holonomic vs Nonholobomic Mobile Robots

Holonomic
omnidirectional robot

Nonholonomic
differential-drive robot

Holonomic Robots

The first type of holonomic robot:

- The constraints do exist but are all holonomic kinematic constraints.
- A holonomic kinematic constraint can be expressed as an explicit function of pose variables only.
- This is the case for all holonomic robots with $\delta_M < 3$.

The second type of holonomic robot:

- A robot that has zero nonholonomic kinematic constraints, i.e., $N_f = 0$ and $N_s = 0$.
- This is the case for all holonomic robots with $\delta_M = 3$.

A robot is holonomic iff $DOF = DDOF$.

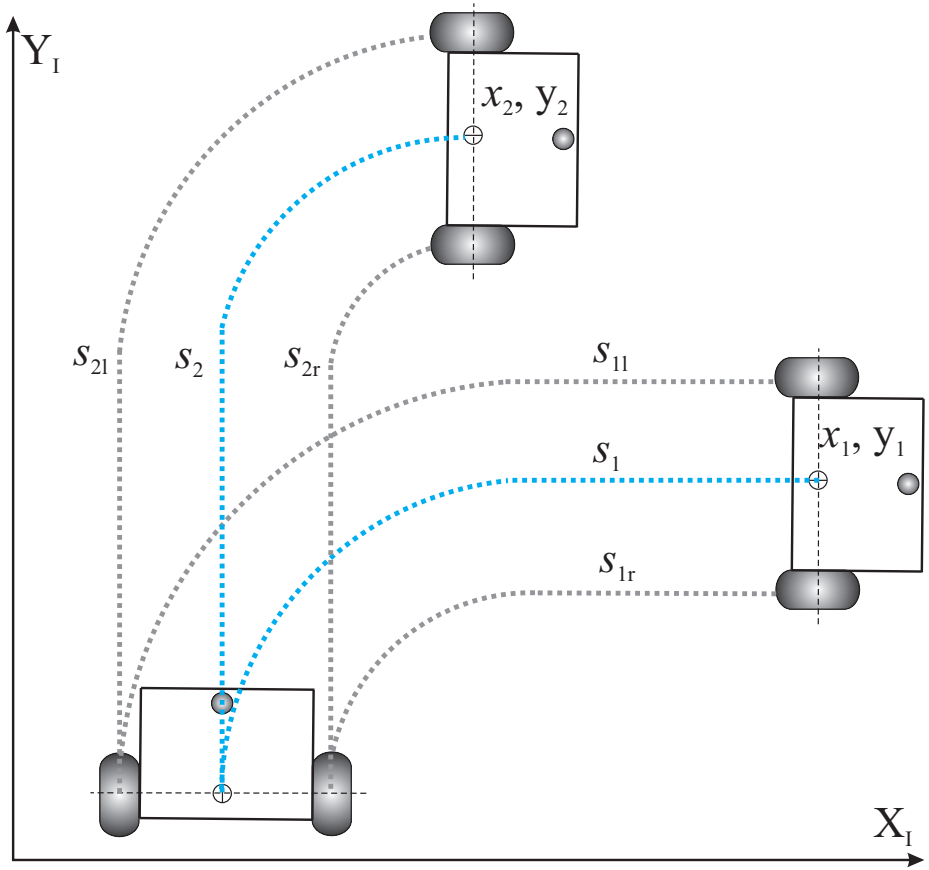
- DOF defines the robot's ability to achieve various poses, and
- $DDOF$ (differential degrees of freedom) ($DDOF = \delta_m$) governs the robot's ability to achieve various paths.
- An omnidirectional robot is a holonomic robot with $DDOF = 3$.

Nonholonomic Robot

Nonholonomic robot:

- A robot with one or more nonholonomic kinematic constraints.
- A nonholonomic constraint requires a differential relationship, such as the derivative of a pose variable.
- It cannot be integrated to provide a constraint in terms of the pose variables only.
- Nonholonomic constraints restrict only the velocity, they allow to reach any pose.
- The constraints defined in terms of velocities show which directions of motion at each time instant are not feasible.
- Fixed and steered standard wheels impose nonholonomic constraints.

Mobile Robot Kinematics: Nonholonomic Systems



but

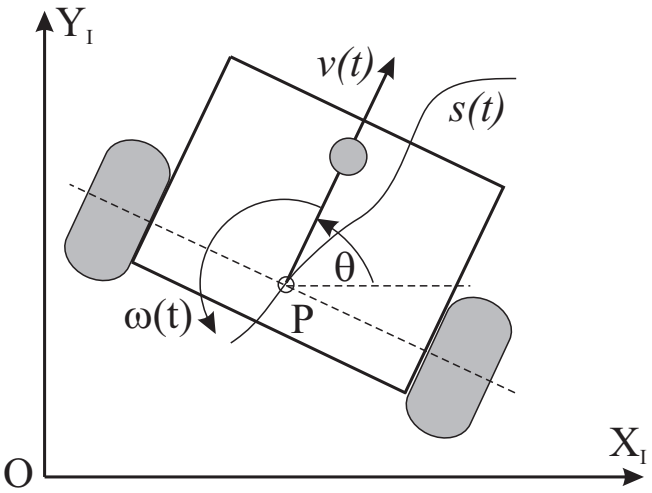
$$s_1 = s_2; \quad s_{1r} = s_{2r}; \quad s_{1l} = s_{2l}$$

$$x_1 \neq x_2; \quad y_1 \neq y_2$$

Nonholonomic systems:

- differential equations are not integrable to the final pose,
- the measure of the traveled distance of each wheel is not sufficient to calculate the final position of the robot, one has also to know how this movement was executed as a function of time.

Nonholonomic Systems: Mathematical Interpretation



A mobile robot is moving along a trajectory $s(t)$. At every instant of the movement its velocity $v(t)$ is:

$$v(t) = \frac{\partial s}{\partial t} = \frac{\partial x}{\partial t}\cos\theta + \frac{\partial y}{\partial t}\sin\theta$$

$$ds = dx\cos\theta + dy\sin\theta$$

- Function $v(t)$ is said to be integrable (holonomic) if there exists a trajectory function $s(t)$ that can be described by the values x, y, and θ only: $s = s(x, y, \theta)$
- This is the case if (Condition for an integrable function)

$$\frac{\partial^2 s}{\partial x \partial y} = \frac{\partial^2 s}{\partial y \partial x}; \quad \frac{\partial^2 s}{\partial x \partial \theta} = \frac{\partial^2 s}{\partial \theta \partial x}; \quad \frac{\partial^2 s}{\partial y \partial \theta} = \frac{\partial^2 s}{\partial \theta \partial y}$$

- With $s = s(x, y, \theta)$ we get: $ds = \frac{\partial s}{\partial x}dx + dy\frac{\partial s}{\partial y} + d\theta\frac{\partial s}{\partial \theta}$

Nonholonomic Systems: The Mobile Robot Example

- In the case of a mobile robot where

$$ds = dx\cos\theta + dy\sin\theta$$

- and by comparing the equation above with

$$ds = \frac{\partial s}{\partial x}dx + dy\frac{\partial s}{\partial y} + d\theta\frac{\partial s}{\partial \theta}$$

- we find

$$\frac{\partial s}{\partial x} = \cos\theta; \quad \frac{\partial s}{\partial y} = \sin\theta; \quad \frac{\partial s}{\partial \theta} = 0$$

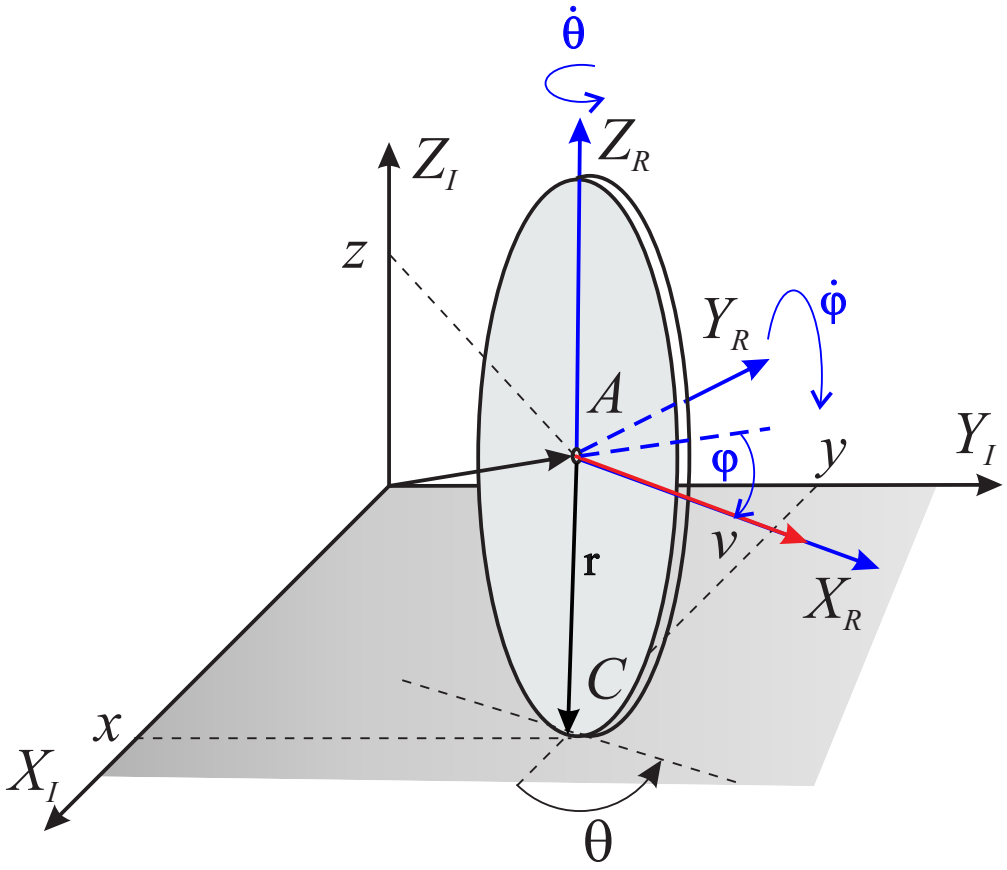
- Condition for an integrable (holonomic) function:

$$\frac{\partial^2 s}{\partial x \partial y} = \frac{\partial^2 s}{\partial y \partial x}; \quad \frac{\partial^2 s}{\partial x \partial \theta} = \frac{\partial^2 s}{\partial \theta \partial x}; \quad \frac{\partial^2 s}{\partial y \partial \theta} = \frac{\partial^2 s}{\partial \theta \partial y}$$

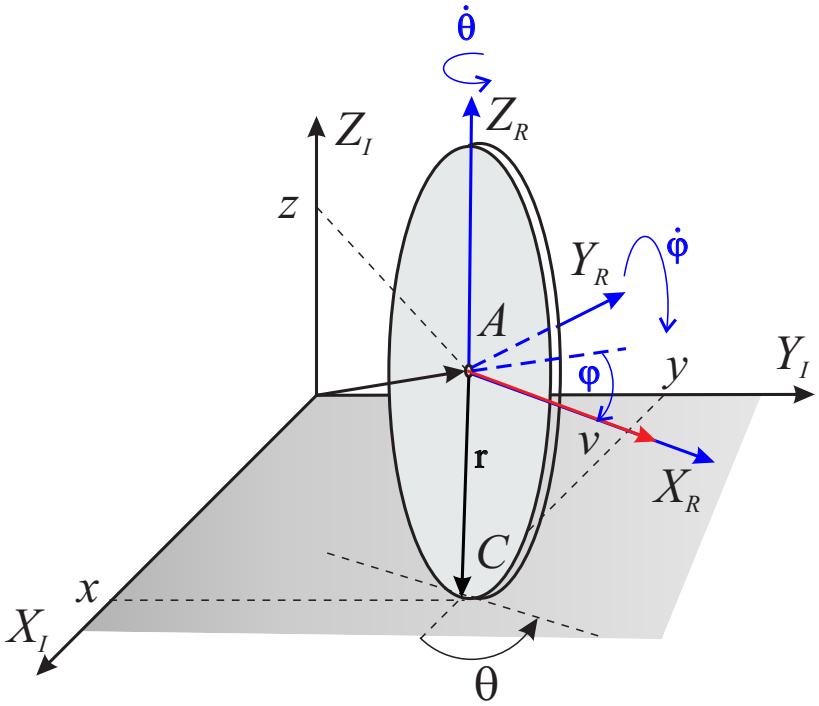
It is easy to check that the second ($-\sin\theta = 0$) and third ($\cos\theta = 0$) term in the above equation do not hold!

The Unicycle – Nonholonomic System

- The linear (translational) and rotational (angular) velocities of a rolling wheel are not independent if the wheel rolls without slipping.
- Nonholonomic constraints arise in rolling without slipping constraints.



The Unicycle – Nonholonomic System



- The configuration of the unicycle is defined by the variables:

$$\boldsymbol{q} = [x, y, \theta, \varphi]^{\mathsf{T}} \in \mathbb{R}^2 \times \mathbb{S}^1 \times \mathbb{S}^1,$$

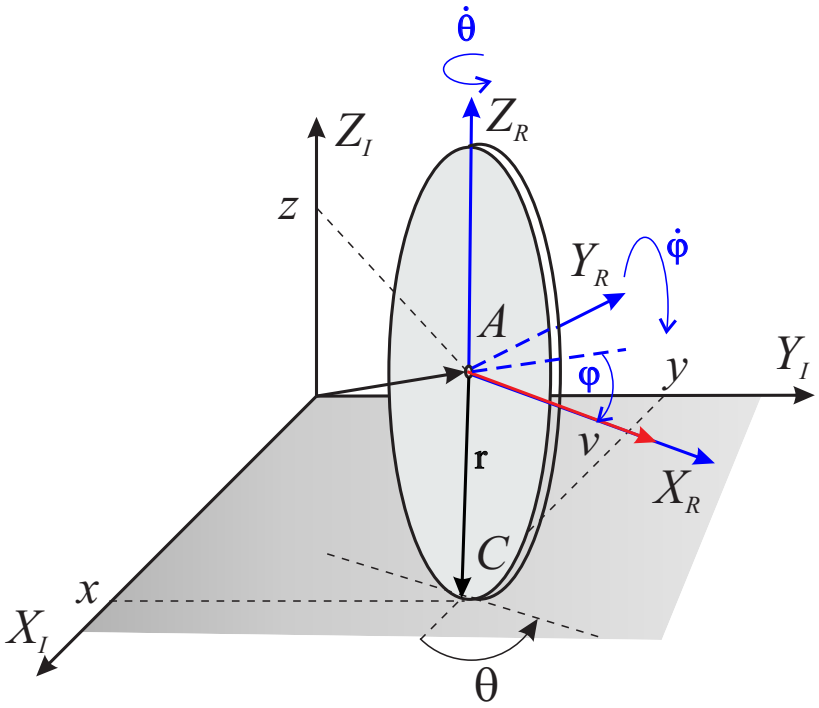
where x, y denote the Cartesian position (z=r=const.), θ the heading angle, and φ the rolling angle.

- Generalized velocity:

$$\dot{\boldsymbol{q}} = [\dot{x}, \dot{y}, \dot{\theta}, \dot{\varphi}]^{\mathsf{T}} \in \mathbb{R}^4,$$

where $\dot{x}, \dot{y}$ denote coordinates of linear velocity (in the inertial coordinate frame), while $\dot{\theta}, \dot{\varphi}$ are angular velocities.

The Unicycle – Nonholonomic System



- Linear velocity described in the local frame:

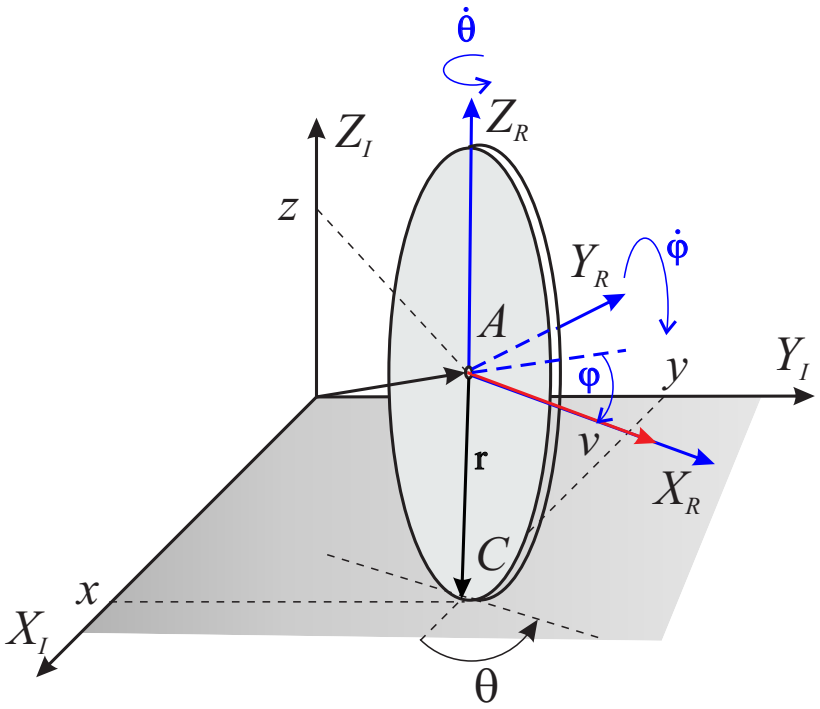
$$\boldsymbol{v} = [v_x, v_y]^{\mathsf{T}} \in \mathbb{R}^2$$

- Mapping of the linear velocity from the local frame to the inertial frame:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = R(\theta)\boldsymbol{v}, \qquad (1)$$

where $R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \in SO(2)$ is a rotation matrix (an element of the special orthogonal group SO(2)).

The Unicycle – Nonholonomic System



- Obtaining $\boldsymbol{v}$ from (1) we have:

$$\boldsymbol{v} = R^{\mathsf{T}}(\theta)[\dot{x}, \dot{y}]^{\mathsf{T}}$$

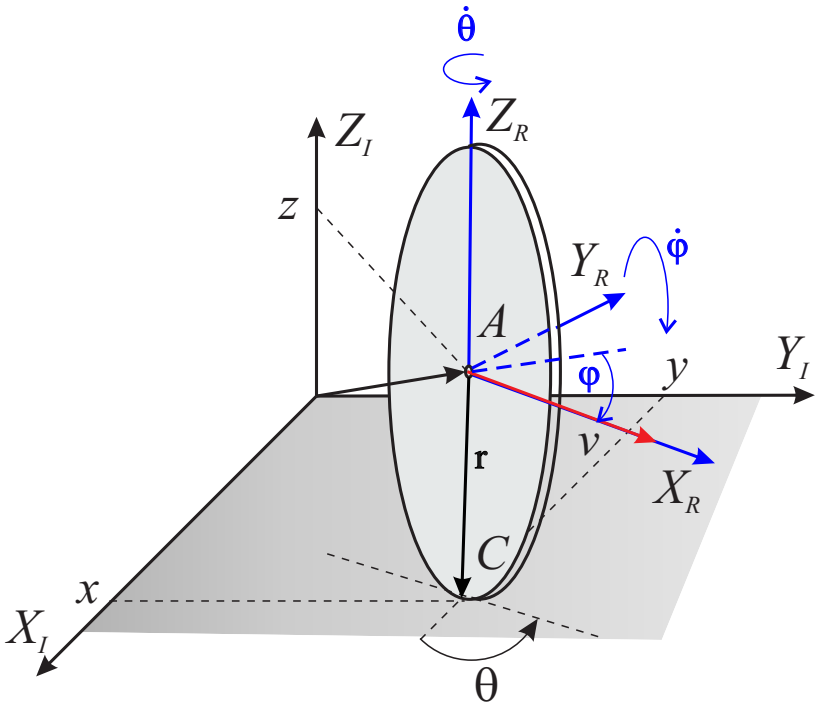
where components of $\boldsymbol{v}$ are

$$v_x = \dot{x}\cos\theta + \dot{y}\sin\theta \qquad (2)$$

$$v_y = -\dot{x}\sin\theta + \dot{y}\cos\theta \qquad (3)$$

- Assuming an ideal motion of the wheel, without slipping in the longitudinal (along axis X_R) and lateral (along axis Y_R) directions, velocities v_x and v_y depend on the components of generalized velocity $\dot{\boldsymbol{q}}$.

The Unicycle – Kinematic Constraints



- In the longitudinal direction we can consider the condition of pure rolling:

$$v_x - r\dot{\varphi} = 0 \tag{4}$$

- Substituting (2) to (4) we obtain

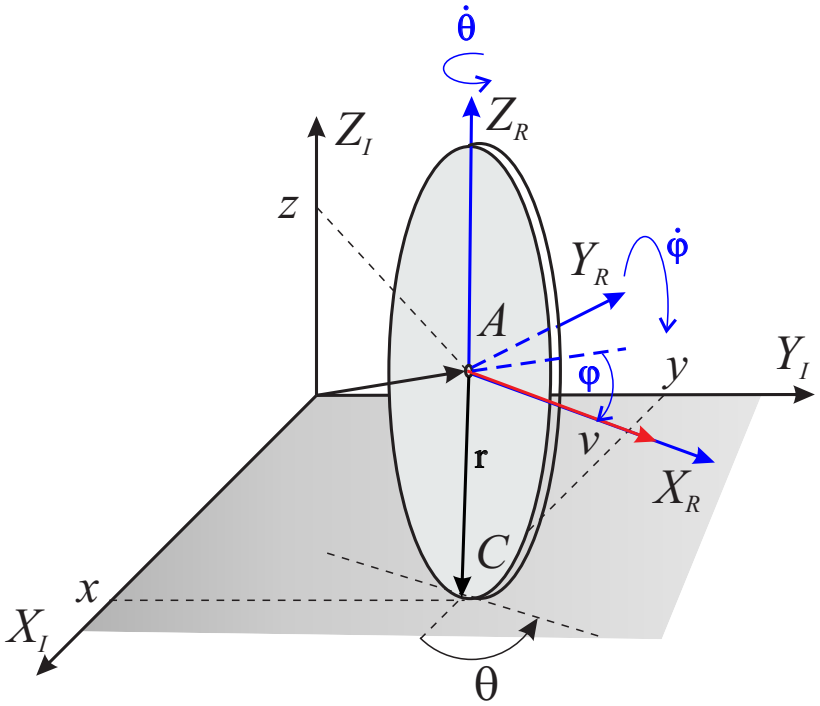
$$\dot{x}\cos\theta + \dot{y}\sin\theta - r\dot{\varphi} = 0, \tag{5}$$

Alternatively, it can be written in the matrix form:

$$\underbrace{\begin{bmatrix}\cos\theta & \sin\theta & 0 & -r\end{bmatrix}}_{w_1(\boldsymbol{q})}\dot{\boldsymbol{q}} = 0, \tag{6}$$

where $w_1(\boldsymbol{q}) \in \mathbb{R}^{1\times 4}$ denotes the constraint matrix.

The Unicycle – Kinematic Constraints



- In the lateral direction any motion is not permitted:

$$v_y = 0 \tag{7}$$

- Substituting from (3) for v_y yields:

$$-\dot{x}\sin\theta + \dot{y}\cos\theta = 0 \tag{8}$$

- Again (8) can be expressed in the matrix form:

$$\underbrace{\begin{bmatrix} -\sin\theta & \cos\theta & 0 & 0 \end{bmatrix}}_{w_2(\boldsymbol{q})} \dot{\boldsymbol{q}} = 0, \tag{9}$$

where $w_2(\boldsymbol{q}) \in \mathbb{R}^{1\times 4}$ is the constraint matrix.

- The constraints (6) and (9) are linear and they are defined in the Pfaffian form:

$$\underbrace{\begin{bmatrix} \cos\theta & \sin\theta & 0 & -r \\ -\sin\theta & \cos\theta & 0 & 0 \end{bmatrix}}_{W(\boldsymbol{q})} \dot{\boldsymbol{q}} = \mathbf{0}, \tag{10}$$

  where

$$W(\boldsymbol{q}) = \begin{bmatrix} w_1(\boldsymbol{q}) \\ w_2(\boldsymbol{q}) \end{bmatrix} \in \mathbb{R}^{2\times 4}$$

- Formally we can state that velocity $\dot{\boldsymbol{q}}$ lies in the null space of matrix $W(\boldsymbol{q})$ or alternatively, that it belongs to the kernel of matrix $W(\boldsymbol{q})$:

$$\dot{\boldsymbol{q}} \in \mathrm{Ker}(W(\boldsymbol{q}))$$

The Unicycle – Kinematic Constraints

- Since the dimension of the unicycle configuration space is $n = 4$ and there are two constraint equations, we need to find two basis vectors g_1, g_2 orthogonal to w_1, w_2.
  It is easy to verify that

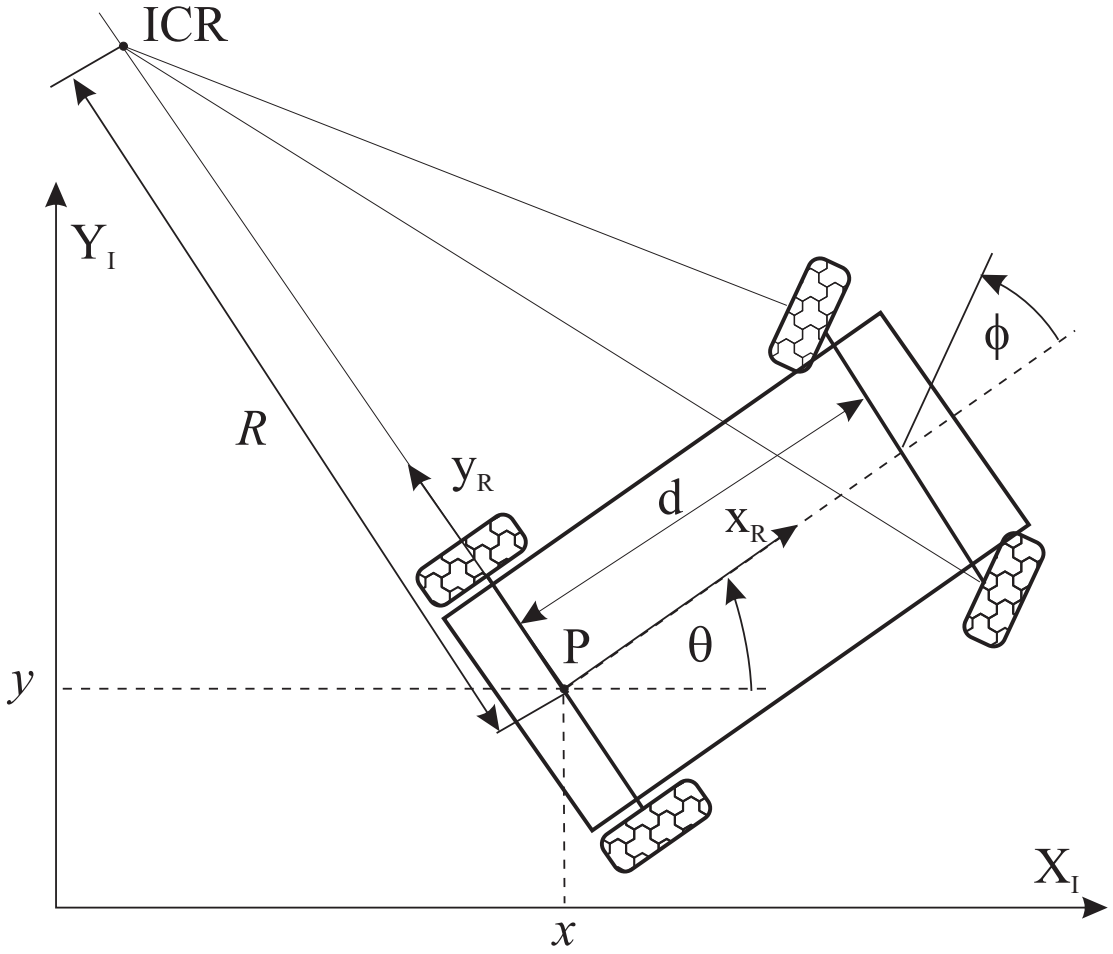
$$g_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}; \qquad g_2 = \begin{bmatrix} r\cos\theta \\ r\sin\theta \\ 0 \\ 1 \end{bmatrix}$$

- The velocity vector $\dot{\boldsymbol{q}}$ can be expressed as a linear combination of the basis vectors:

$$\dot{\boldsymbol{q}} = g_1(q)u_1 + g_2(q)u_2,$$

  where the coefficient u_1 is the turning rate and u_2 is the rate of rolling.

Nonholonomic Systems: The Kinematic Car I



The configuration of the car-like robot:

$$\boldsymbol{q} = \begin{bmatrix} x \\ y \\ \theta \\ \phi \end{bmatrix}$$

Nonholonomic Systems: The Kinematic Car II

The constraints are obtained by setting the sideways velocity of the front an rear wheels to zero:

$$\sin\theta\,\dot{x} - \cos\theta\,\dot{y} = 0$$
$$\sin(\theta+\phi)\,\dot{x} - \cos(\theta+\phi)\,\dot{y} - d\cos\phi\,\dot{\theta} = 0$$

The constraints can be written as

$$[\sin\theta \;\; -\cos\theta \; 0 \; 0]\,\dot{\boldsymbol{q}} = \langle w_1, \dot{\boldsymbol{q}}\rangle = 0$$
$$[\sin(\theta+\phi) \;\; -\cos(\theta+\phi) \;\; -d\cos\phi \; 0]\,\dot{\boldsymbol{q}} = \langle w_2, \dot{\boldsymbol{q}}\rangle = 0$$

It is easy to find vectors

$$g_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}; \quad g_2 = \begin{bmatrix} \cos\theta \\ \sin\theta \\ \frac{1}{d}\tan\phi \\ 0 \end{bmatrix}$$

Integrability of kinematic constraints

How to examine integrability of the velocity constraints given by $W(\boldsymbol{q})\dot{\boldsymbol{q}} = 0$?

In order to answer this we will consider the following lemma.

Lemma

Assuming that kinematic constraint $W(\boldsymbol{q})\dot{\boldsymbol{q}} = 0$ *is integrable it follows that there exists function* $\boldsymbol{k}(\boldsymbol{q}) = 0$ *such that the following relationship is satisfied*:

$$\frac{\partial}{\partial \boldsymbol{q}} \boldsymbol{k}(\boldsymbol{q}) = W(\boldsymbol{q})$$

- Kinematic constraint is integrable – holonomic – it means that configuration space is constrained and the chosen coordinate variables are redundant.
- Kinematic constraint is nonintegrable – nonholonomic – the constraint do not reduce the dimension of the configuration space.

Stabilizability I

- Any arbitrary configuration

$$\boldsymbol{q} = \left[\boldsymbol{p}^{\mathsf{T}},\ \beta_s^{\mathsf{T}}\right]^{\mathsf{T}}, \quad \text{where } \boldsymbol{p} = [x, y, \theta]^{\mathsf{T}}$$

  may be an equilibrium point for the posture kinematic model.

- Equilibrium means that the robot is at rest:
  - with a given constant posture $\overline{\boldsymbol{p}}$ and a given constant orientation $\overline{\beta_s}$ of the steering wheel, and
  - zero controls, i.e., $\overline{\boldsymbol{u}} = 0$.

Stabilizability II

Consider the question of the existence of a feedback control $\boldsymbol{u}(\boldsymbol{q})$ able to stabilize a mobile robot at a particular configuration $\overline{\boldsymbol{q}}$.

- For holonomic robots (for which $\boldsymbol{q}$ reduces to $\boldsymbol{p}$) the problem is trivial, because $\dot{\boldsymbol{p}}$ can be assigned arbitrarily from the input $\boldsymbol{u}$.

- For nonholonomic robots (restricted mobility robots) the problem is more complex.
  - The (nonholonomic) posture kinematic model is not stabilizable by a *continuous static time-invariant* state feedback.
  - Such a model can be stabilized using *smooth time-varying state feedback* or *noncontinuous feedback*.

Static State-Feedback Linearizability

We analyze the existence of state feedback achieving full or partial linearization of the posture kinematic model:

- The largest subsystem of the posture kinematic model linearizable by a smooth static feedback has dimension $(\delta_m + \delta_s)$.
- The largest linearizable subsystem of the kinematic posture model is obtained by selecting $(\delta_m + \delta_s)$ adequate linearizing output functions depending on $\boldsymbol{q}$.
- A vector of $(3 - \delta_m)$ components remains non-linearized.
- For robots with holonomic posture models, the models are fully linearizable by static state feedback.
- Restricted mobility robots (nonholonomic) posture models are only partially linearizable.

Control Systems

- From the control point of view, most of dynamical systems are *control affine* nonlinear control systems

$$\dot{x} = g_0(x) + \sum_{i=1}^{m} g_i(x)u_i, \quad u \in \mathcal{U} \subset \mathbb{R}^m$$

- The vector field g_0 is called the drift vector field, defining unforced motion of the system.
- If $g_0 = 0$, the system is called *drift-free* or *driftless*.
- The $g_i, i = 1, \ldots, m$ are linearly independent *control vector fields*.
- The control vector $u(t)$ is piecewise continuous.
- We will study driftless control system

$$\dot{x} = \sum_{i=1}^{m} g_i(x)u_i, \quad u \in \mathcal{U} \subset \mathbb{R}^m$$

- For example the unicycle is a driftless control system

$$\dot{x} = g_1(x)u_1 + g_2(x)u_2,$$

where u_1 is the driving speed and u_2 is the steering velocity.

Motion Control

Consider motion control problem for nonholonomic wheeled mobile robots moving on the plane:

- Motion control is not straight forward because mobile robots are mostly nonholonomic systems.
- Most controllers are not considering the dynamics of the system.
- However, in the case of high-speed mobile robots dynamic constraints must be taken into account in addition to kinematic constraints.
- Many mobile platforms such as tank-type vehicles and four-wheel slip/skid systems violate the kinematic constraints e.g. rolling-without-slipping constraints.

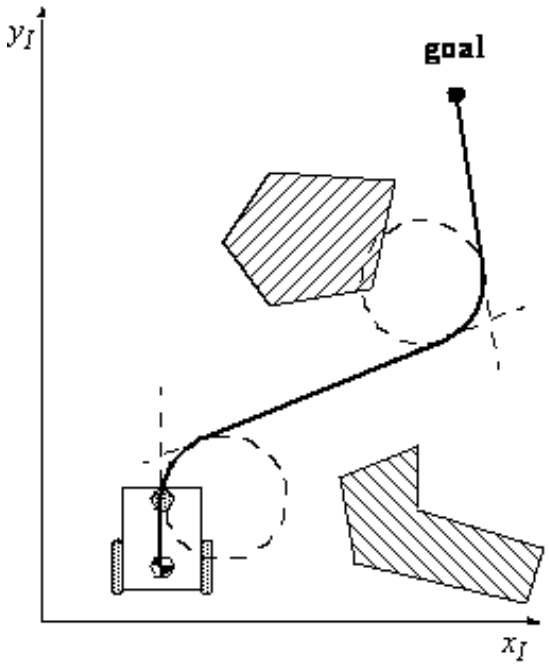
Two broad classes of motion control strategies:

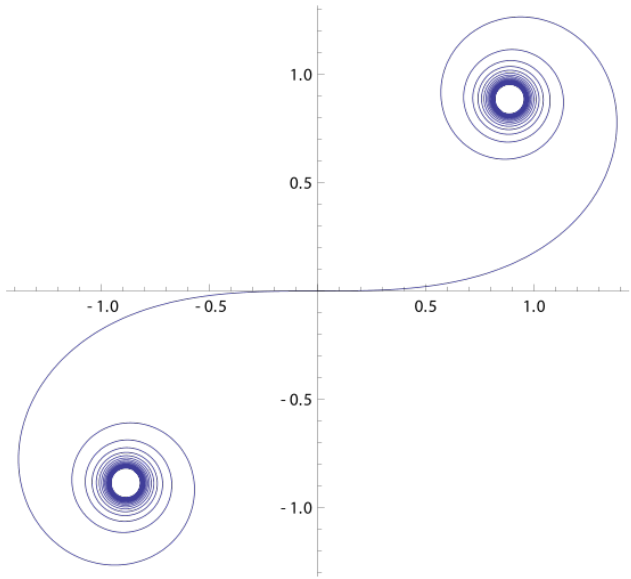
- *open-loop* control
- *feedback* control

Motion Control: Open Loop Control I

- The objective of a controller is to follow a desired trajectory.
- Typically, the trajectory/path is decomposed into a sequence of elementary motion segments of clearly defined shapes:
  - straight lines and segments of a circle (as done on American roads),
  - segments of clothoids in which the curvature changes linearly with time (as done on European roads),

  which any robot with $\delta_M \geq 2$ can execute.





Motion Control: Open Loop Control II

- The control problem is to pre-compute a smooth trajectory based on a combination of line and circle segments.
- Disadvantages:
  - It is not an easy task to pre-compute a feasible trajectory if all limitations and constraints have to be considered.
  - The control system is not robust to modeling errors (the sources of which are numerous) and that it cannot guarantee that the mobile robot will move along the desired trajectory as planned.
  - The controller does not adapt or correct the trajectory if dynamical changes of the environment occur.
  - The resulting trajectories are usually not smooth.
- A more appropriate strategy in motion control is to use a feedback controller that uses the path specification as “control” points to drive the robot system.

Exercise – Open loop control I

For a differential-drive robot compute robot base poses (expressed in an inertial coordinate frame) at times $t = 0.5$, $t = 1$ and $t = 2$ assuming, that at $t = 0$ the robot starts in pose $\boldsymbol{p}(0) = [0, 0, 0]^{\mathsf{T}}$. The wheel radius is $r = 1$, and the distance between wheels is $d = 0.5$. The angular speeds of the left $\omega_L = \dot{\varphi}_L$ and right $\omega_R = \dot{\varphi}_R$ wheel are shown in the Figure.

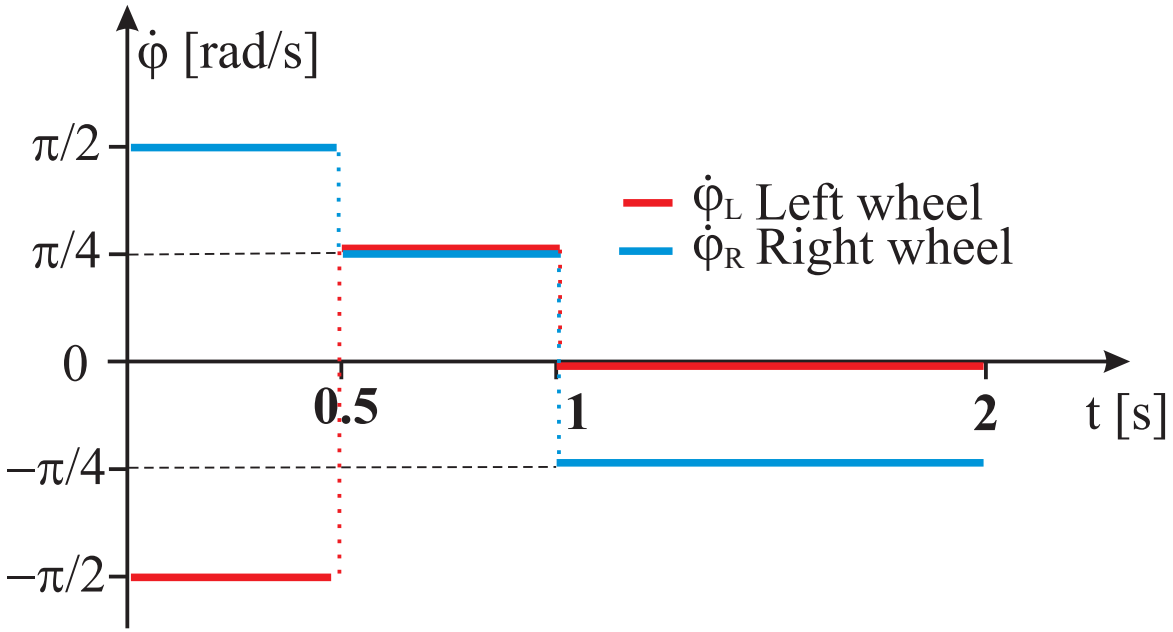
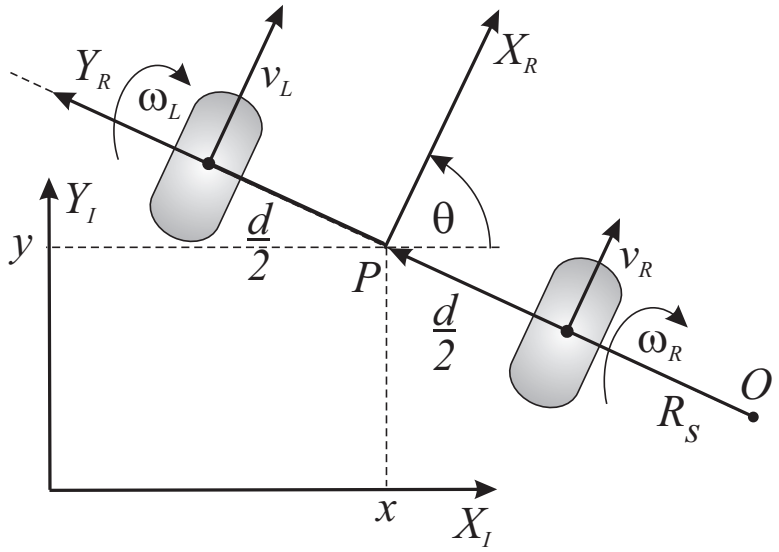


Fig. Wheel angular velocities

Exercise – Open loop control II

Solution:



ICR expressed in the robot coordinate frame (X_R, Y_R):

$$\begin{bmatrix} O_x \\ O_y \end{bmatrix}_R = \begin{bmatrix} 0 \\ R_s \end{bmatrix}$$

and in the inertial coordinate frame (X_I, Y_I):

$$\begin{bmatrix} O_x \\ O_y \\ 1 \end{bmatrix}_I = \begin{bmatrix} \cos\theta & -\sin\theta & x \\ \sin\theta & \cos\theta & y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ R_s \\ 1 \end{bmatrix} = \begin{bmatrix} x - R_s sin\theta \\ y + R_s \cos\theta \\ 1 \end{bmatrix}$$

t=0.5: The first movement is a rotation in place with

$$\omega = \frac{r(\omega_R - \omega_L)}{d} = \frac{r(\dot{\varphi}_R - \dot{\varphi}_L)}{d} = \frac{\frac{\pi}{2} - \left(-\frac{\pi}{2}\right)}{0.5} = 2\pi,$$

the duration of the in place rotation is $\Delta t = 0.5$, thus the final orientation is equal to

$$\theta(0.5) = \theta(0) + \omega \cdot \Delta t = 0 + 2\pi \cdot 0.5 = \pi,$$

and the robot pose at $t = 0.5$ is $\boldsymbol{p}(0.5) = [0, 0, \pi]^\mathsf{T}$.

t=1: The second movement is a straight line motion with

$$v = \frac{r(\dot{\varphi}_L + \dot{\varphi}_R)}{2} = \frac{\frac{\pi}{4} + \frac{\pi}{4}}{2} = \frac{\pi}{4}$$

The robot starts from $\boldsymbol{p}(0.5) = [0, 0, \pi]^\mathsf{T}$ facing left and moves for $\Delta t = 0.5$ along the X axis (in the negative direction) thus moving to

$$x(1) = x(0.5) - v \cdot \Delta t = 0 - \frac{\pi}{4} \cdot 0.5 = -\frac{\pi}{8}$$

the final pose at $t = 1$ is thus $\boldsymbol{p}(1) = [-\frac{\pi}{8}, 0, \pi]^\mathsf{T}$.

t=2: Finally, the robot rotates around the left wheel and performs an arc of circumference at the angular speed of

$$\omega = \frac{r(\omega_R - \omega_L)}{d} = \frac{\frac{-\pi}{4} - 0}{0.5} = -\frac{\pi}{2},$$

The ICR $O = [O_x, O_y]^\mathsf{T} = [-\frac{\pi}{8}, -0.25]^\mathsf{T}$ is on the left wheel.

The final pose $\boldsymbol{p}(2) = [x(2), y(2), \theta(2)]^{\mathsf{T}}$ of the robot at $t = 2$ we can compute applying analytical formula

$$\begin{bmatrix} x(2) \\ y(2) \\ \theta(2) \end{bmatrix} = \begin{bmatrix} \cos(\omega \cdot \Delta t) & -\sin(\omega \cdot \Delta t) & 0 \\ \sin(\omega \cdot \Delta t) & \cos(\omega \cdot \Delta t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x - O_x \\ y - O_y \\ \theta \end{bmatrix} + \begin{bmatrix} O_x \\ O_y \\ \omega \cdot \Delta t \end{bmatrix}$$

and by plugging values we obtain

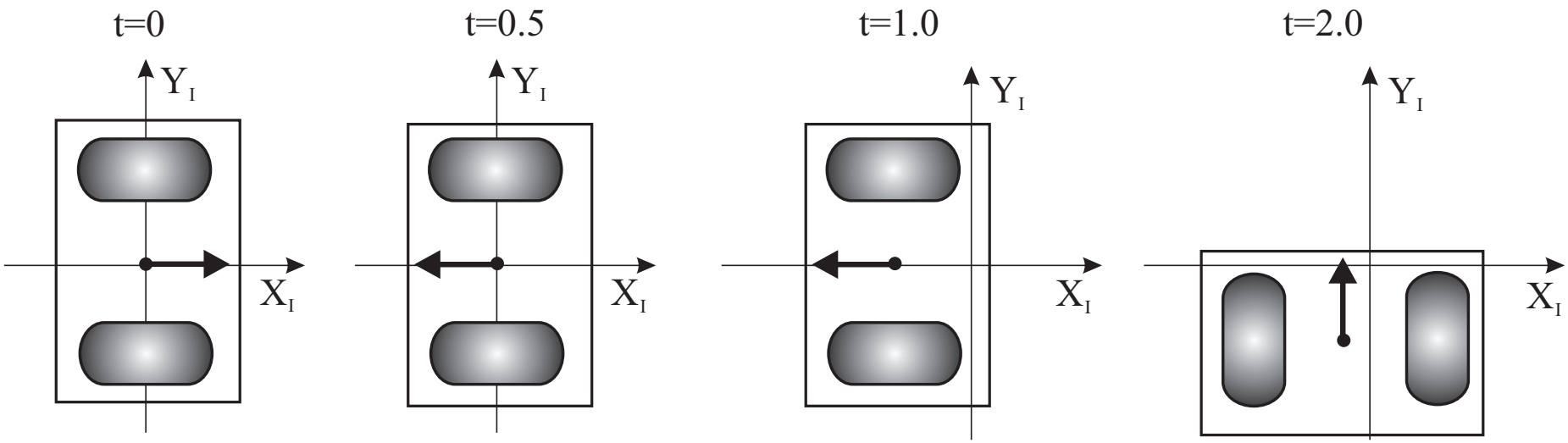
$$\begin{bmatrix} x(2) \\ y(2) \\ \theta(2) \end{bmatrix} = \begin{bmatrix} \cos(-\frac{\pi}{2}) & -\sin(-\frac{\pi}{2}) & 0 \\ \sin(-\frac{\pi}{2}) & \cos(-\frac{\pi}{2}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -\frac{\pi}{8} - (-\frac{\pi}{8}) \\ 0 - (-0.25) \\ \pi \end{bmatrix} + \begin{bmatrix} -\frac{\pi}{8} \\ -0.25 \\ -\frac{\pi}{2} \end{bmatrix}$$

Finally the robot pose at $t = 2$ is

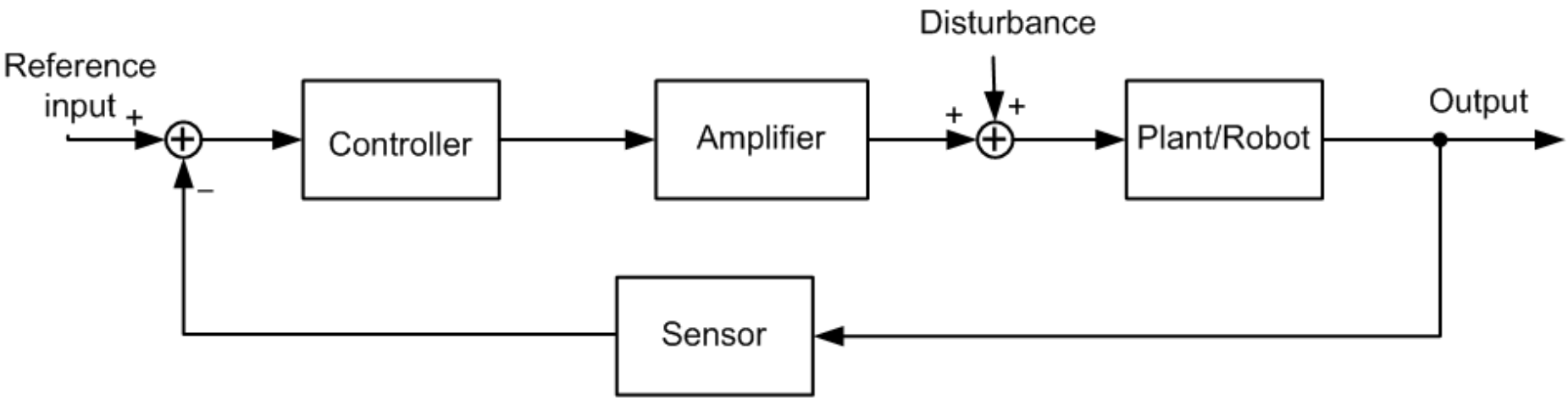
$$\boldsymbol{p}(2) = \begin{bmatrix} x(2) \\ y(2) \\ \theta(2) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0.25 \\ \pi \end{bmatrix} + \begin{bmatrix} -\frac{\pi}{8} \\ -0.25 \\ -\frac{\pi}{2} \end{bmatrix} = \begin{bmatrix} 0.25 - \frac{\pi}{8} \\ -0.25 \\ \frac{\pi}{2} \end{bmatrix}$$

Exercise – Open loop control VI

Robot poses



Feedback Control



Advantages of closed-loop vs. open-loop:

- disturbance rejection
- reduced effect of an imperfect knowledge of the system
- improved performance
- stabilization

Risks of closed-loop control:

– a stable system can be made unstable!

Proportional Control

Control law:

$$u = k \cdot (y - r) = k \cdot e,$$

where
u – control; y – output (measured); r – reference; k – proportional gain.

Properties:

- More action when the measured output is far from the desired value (i.e. bigger error).
- Can produce a smooth behavior (smooth path).
- The gain k has an effect on the closed-loop performance.

Drawbacks:

- Too large k may render the system unstable.
- In some cases, the output does not converge to the desired value. (When?)

Proportional-Derivative Control

Control law:

$$u = k\left((y-r) + T_d \frac{d}{dt}(y-r)\right) = k\left(e + T_d \frac{de}{dt}\right),$$

where k – proportional gain, T_d – derivative time
Properties:

- control u anticipates the changes by extrapolating $e = y - r$

$$u \approx k\ (y(t+T_d) - r(t+T_d)) = k\, e(t+T_d)$$

- prevents or reduces the anti-stabilizing effect of feedback
- parameters k and T_d have an effect on the closed-loop performance
- PD is usually sufficient if the system itself already has an integrator (such as the motion control of a mobile robot)

Drawbacks:

– derivating quickly-changing signals (discontinuities in desired value, noisy measure) amplifies them and may produce the saturation of u.

Feedback Control: Problem Statement

Simple proportional-gain controller:

- In the robot reference frame the error is defined as

$$e = [x,\ y,\ \theta]^{\mathsf{T}}_R$$

- The task is now to design a control matrix K, if it exists

$$K = \begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \end{bmatrix} \qquad \text{with } k_{ij} = k(t, e)$$

- Such that the control of $v(t)$ and $\omega(t)$

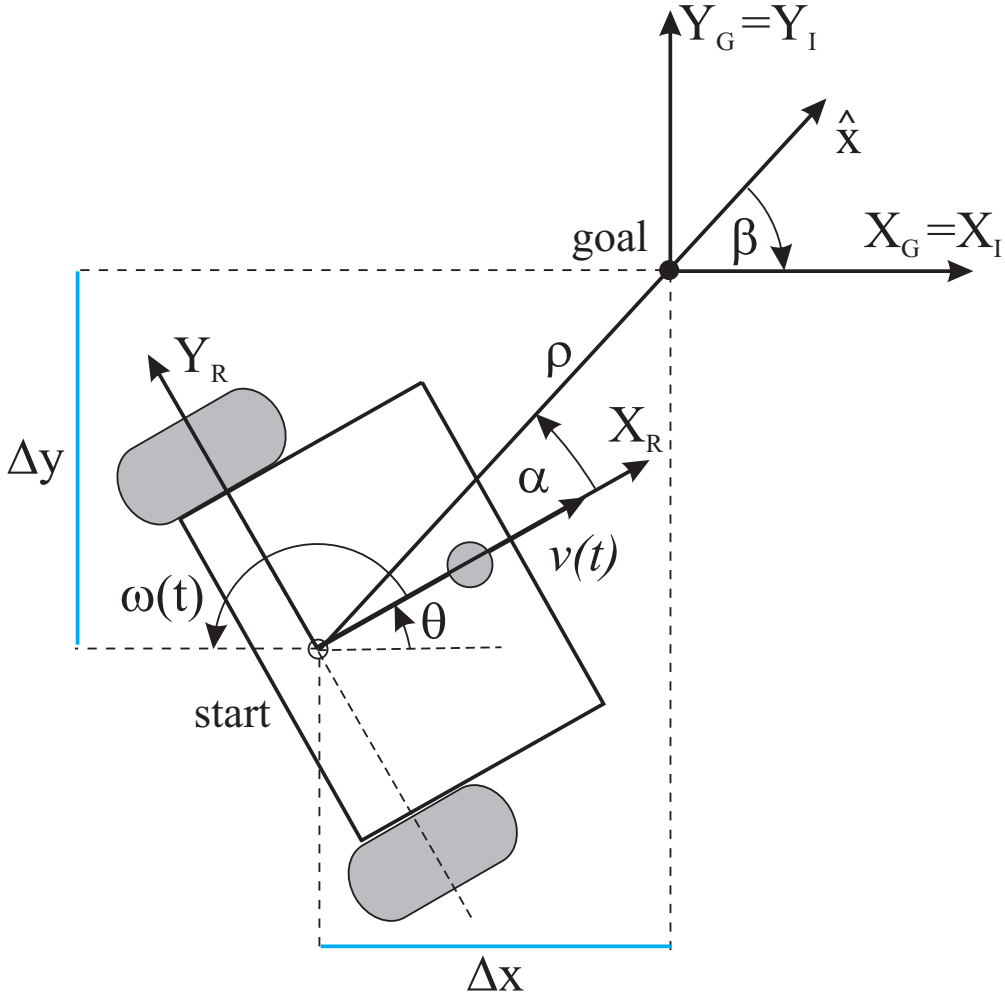
$$\begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} = K \cdot e = K \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}_R$$

  drives the error $e(t)$ to zero $\lim_{t\to\infty} e(t) = 0$.
- Vector $v(t)$ is always heading in the X_R direction of the robot's reference frame due to the non-holonomic constraint.

Kinematic Position Control I

The kinematics of a differential drive robot described in the inertial frame:

$$\dot{\boldsymbol{p}}_I = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$$



Kinematic Position Control II

- Let α denote the angle between the X_R axis of the robot's reference frame and the vector connecting the center of the axle of the wheels with the final position.
- Transformation into polar coordinates with its origin at goal position:

$$\rho = \sqrt{\Delta x^2 + \Delta y^2}$$
$$\alpha = -\theta + \text{atan2}(\Delta y, \Delta x)$$
$$\beta = -\theta - \alpha$$

- In the polar coordinates the kinematics is as follows:

$$\begin{bmatrix} \dot{\rho} \\ \dot{\alpha} \\ \dot{\beta} \end{bmatrix} = \begin{bmatrix} -\cos\alpha & 0 \\ \frac{\sin\alpha}{\rho} & -1 \\ -\frac{\sin\alpha}{\rho} & 0 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$$

Kinematic Position Control III

Remarks:

- The coordinates transformation is not defined at $x = y = 0$; as in such a point the determinant of the Jacobian matrix of the transformation is not defined, i.e. it is unbounded.
- For $\alpha \in I_1 = (-\frac{\pi}{2}, \frac{\pi}{2}]$ the forward direction of the robot points toward the goal, for $\alpha \in I_2 = (-\pi, -\frac{\pi}{2}] \cup (\frac{\pi}{2}, \pi]$ it is the backward direction.
- By properly defining the forward direction of the robot at its initial configuration, it is always possible to have $\alpha \in I_1$ at t=0. However this does not mean that α remains in I_1 for all time t.

Kinematic Position Control: The Control Law

It can be shown, that with linear control law

$$v = k_\rho \rho$$
$$\omega = k_\alpha \alpha + k_\beta \beta$$

the feedback controlled system (closed-loop system)

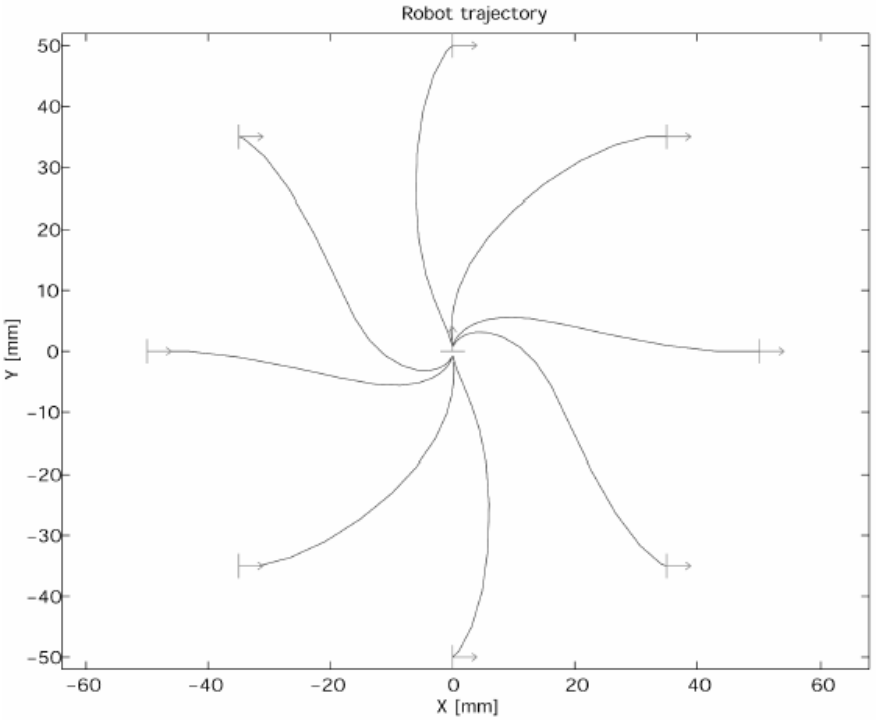
$$\begin{bmatrix} \dot{\rho} \\ \dot{\alpha} \\ \dot{\beta} \end{bmatrix} = \begin{bmatrix} -k_\rho \rho \cos \alpha \\ k_\rho \sin \alpha - k_\alpha \alpha - k_\beta \beta \\ -k_\rho \sin \alpha \end{bmatrix}$$

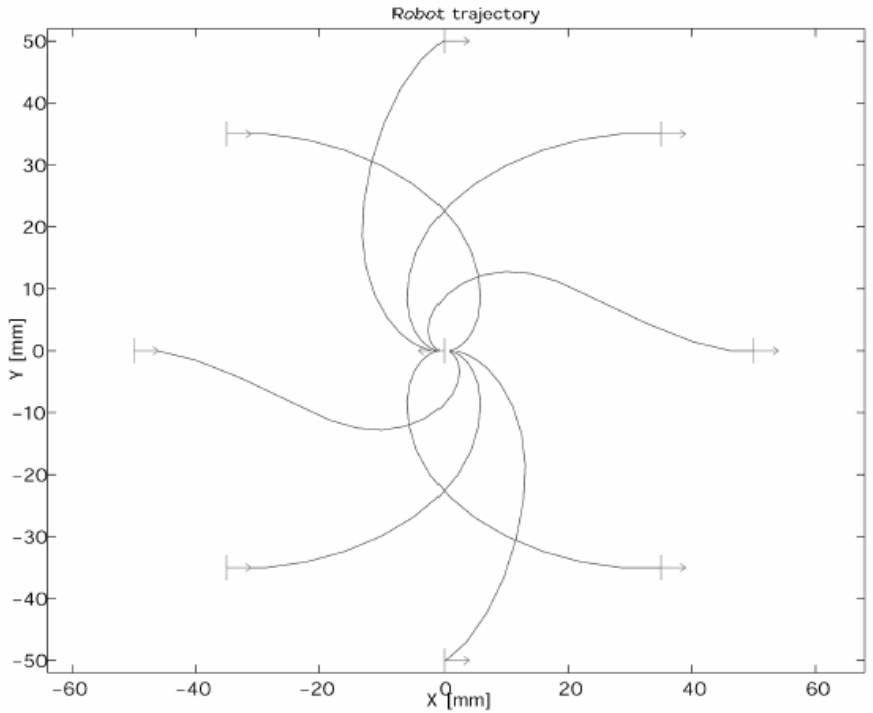
will drive the robot to $(\rho, \alpha, \beta) = (0, 0, 0)$, which is a unique equilibrium point and the goal pose.

Kinematic Position Control: Resulting Paths

The goal is in the center and the control parameters are
$k = (k_\rho, k_\alpha, k_\beta) = (3, 8, -1.5)$

Resulting path when the robot is initially on the circle in the x, y plane:





Kinematic Position Control: Stability Issue

It can be shown, that the closed loop control system is **locally** *exponentially stable* if

$$k_\rho > 0; \quad k_\beta < 0; \quad k_\alpha - k_\rho > 0; \qquad k = (k_\rho, k_\alpha, k_\beta) = (3, 8, -1.5)$$

Proof:
Linearized around the equilibrium (for small $x \to 0 : \cos x = 1, \sin x = x$) we have

$$\begin{bmatrix} \dot{\rho} \\ \dot{\alpha} \\ \dot{\beta} \end{bmatrix} = \begin{bmatrix} -k_\rho & 0 & 0 \\ 0 & -(k_\alpha - k_\rho) & -k_\beta \\ 0 & -k_\rho & 0 \end{bmatrix} \begin{bmatrix} \rho \\ \alpha \\ \beta \end{bmatrix}; A = \begin{bmatrix} -k_\rho & 0 & 0 \\ 0 & -(k_\alpha - k_\rho) & -k_\beta \\ 0 & -k_\rho & 0 \end{bmatrix}$$

and the characteristic polynomial of the matrix A of all roots

$$p_A(\lambda) = \det(\lambda I - A) = (\lambda + k_\rho)(\lambda^2 + \lambda(k_\alpha - k_\rho) - k_\rho k_\beta)$$

have negative real parts.

Wall Follower: Controller Design Example I

- Again we consider differential drive posture kinematic model:

$$\dot{\boldsymbol{p}}_I = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix},$$

and

$$u = [v,\ \omega]^{\mathsf{T}}; \quad \boldsymbol{y} = [y,\ \theta]^{\mathsf{T}}; \quad \theta \approx 0$$

- Desired distance from the wall defines the error:

$$e = y - r, \qquad \text{so } \dot{e} = \dot{y} \text{ and } \ddot{e} = \ddot{y},$$

where $r = const.$ – set-value for y coordinate.

- Assume constant forward linear velocity $v = v_0$ approximately parallel to the wall: $\theta \approx 0$.
- We want an error e to act like a ”damped spring”

$$\ddot{e} + k_1\dot{e} + k_2 e = 0$$

Wall Follower: Controller Design Example II

- For small values of θ

$$\dot{e} = \dot{y} = v \sin\theta \approx v\,\theta$$
$$\ddot{e} = \ddot{y} = v \cos\theta\,\dot{\theta} \approx v\,\omega$$

- Assuming $v = v_0$ and solving for ω we have

$$u = \begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} v_0 \\ -k_1\theta - \frac{k_2}{v_0} e \end{bmatrix}$$

  in this way we derive the PD controller.
- System $\ddot{e} + k_1\dot{e} + k_2 e = 0$ is critically damped if $k_1^2 - 4\,k_2 = 0$, i.e.,

$$k_1 = \sqrt{4\,k_2}$$

- Slightly under-damped system performs better (why?)
  - k_2 can be tuned experimentally (e.g. Ziegler-Nichols PD tuning)
  - k_1 should be a bit less than $\sqrt{4\,k_2}$

Wall Follower: Controller Design Example III

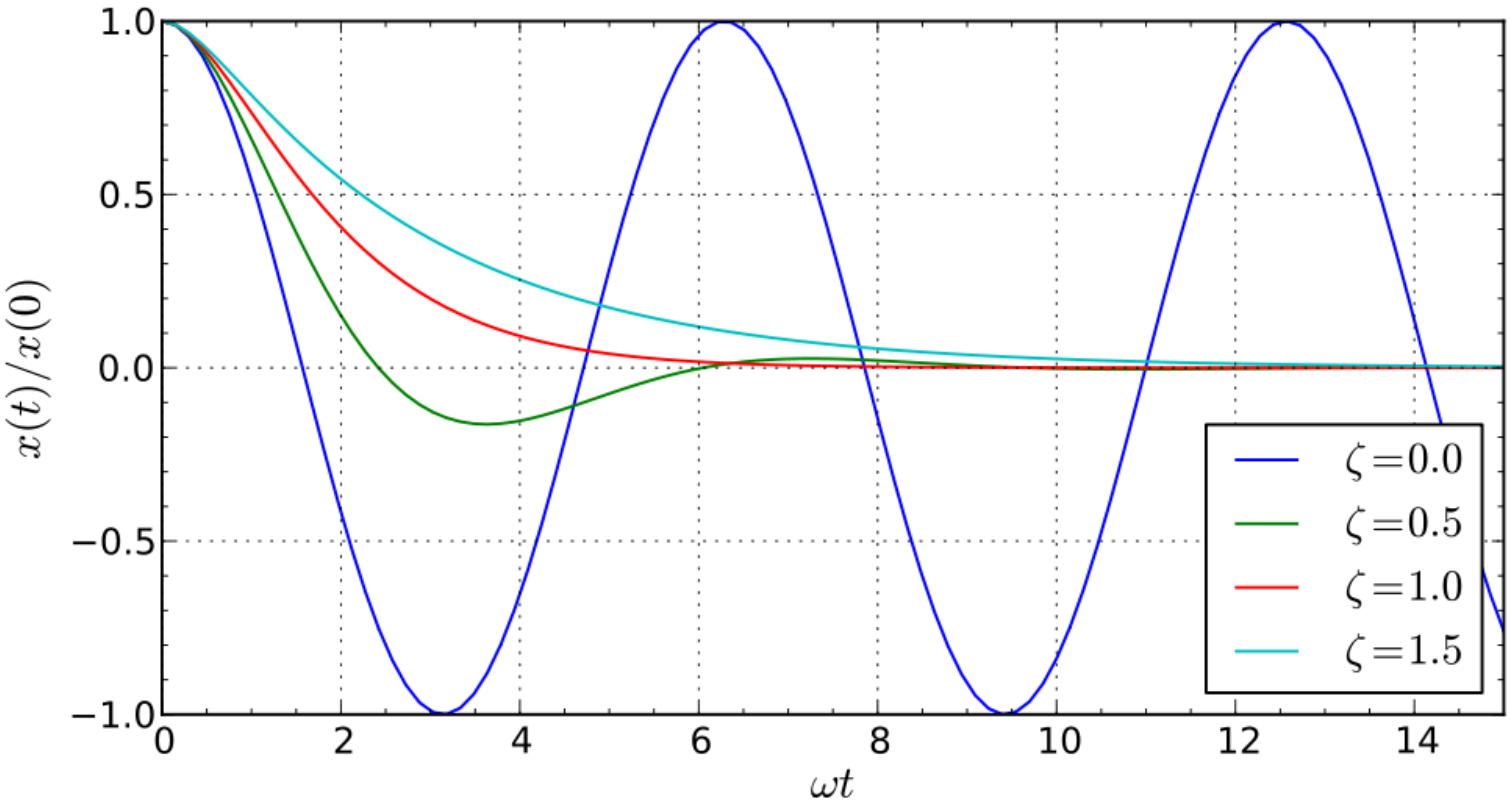
- Second order linear oscillatory systems can be described by

$$\ddot{x} + 2\zeta\omega_0\dot{x} + \omega_0^2 x = 0,$$

where ω_0 is the *undamped angular frequency* of the oscillator and ζ is a constant called the *damping ratio*.
The value of ζ determines the behavior of the system. A damped harmonic oscillator can be:

  - *Overdamped* ($\zeta > 1$): The system returns (exponentially decays) to equilibrium without oscillating. Larger values of the damping ratio ζ return to equilibrium more slowly.
  - *Critically damped* ($\zeta = 1$): The system returns to equilibrium as quickly as possible without oscillating.
  - *Underdamped* ($0 < \zeta < 1$): The system oscillates (at reduced frequency compared to the undamped case) with the amplitude gradually decreasing to zero.
  - *Undamped* ($\zeta = 0$): The system oscillates at its natural resonant frequency (ω_0).

Wall Follower: Controller Design Example IV



Time dependence of the system behavior on the value of the damping ratio $\zeta = 0$ for zero-velocity initial condition.

Motion Control Problems I

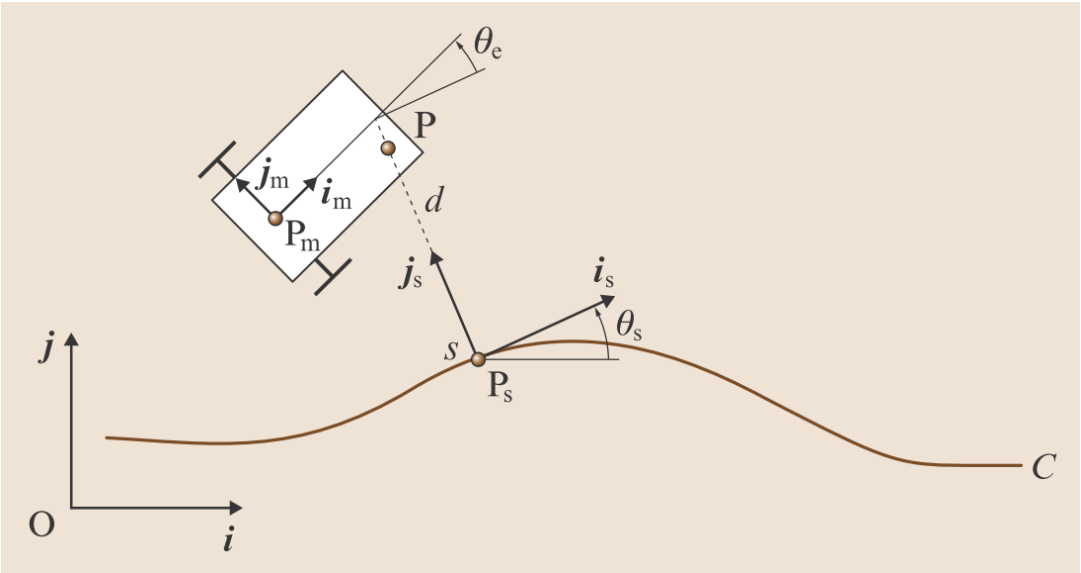
Three generic motion control problems can be distinguished:

- Path following
- Stabilization of trajectories
- Stabilization of fixed postures

Motion Control Problems II

✓ **Path following**:

- Given a curve $\mathcal{C}$ on the plane, a pre-specified (nonzero) linear velocity v_0 for the robot chassis, and a point P attached to the chassis.
- The goal is to have the point P follow the curve $\mathcal{C}$ when the robot moves with the velocity v_0.
- The variable that one has to stabilize at zero is thus the distance between the point P and the curve (i.e., the distance between P and the closest point M on $\mathcal{C}$).



Motion Control Problems III

✓ **Stabilization of trajectories**:

- The goal is to monitor the distance gone along the curve $\mathcal{C}$.
- This problem differs from the previous one in that the vehicle's linear velocity is no longer predetermined.
- The geometric curve $\mathcal{C}$ is is parameterized with the time t, $t \mapsto (x_r(t), y_r(t))$ with respect to a global reference frame $\mathcal{F}_I = (X_I, Y_I)$.
- The goal is to stabilize the position error vector $(x(t) - x_r(t), y(t) - y_r(t))$, where $(x(t), y(t))$ coordinates of point P.
- Perfect tracking is achievable only if the reference trajectory is feasible for the physical vehicle, and that a trajectory which is feasible for a unicycle-type vehicle is not necessarily feasible for a car-like vehicle.
- Sometimes the chassis orientation $\theta(t)$ has to be controlled at a desired reference value $\theta_r(t)$.
- Typically, most trajectories produced by an omnidirectional vehicle are not feasible for a non-holonomic mobile robot.

✓ **Stabilization of fixed postures**:

- The objective is to stabilize at zero the posture $\xi = [x(t), y(t), \theta(t)]^{\mathsf{T}}$.
- Although a fixed desired (or reference) posture is obviously a particular case of a feasible trajectory, this problem cannot be solved by classical control methods for linear systems or based on linearization.
- This problem is very different from the problems of path following and trajectory tracking with nonzero linear velocity, much in the same way as a human driver knows, from experience, that parking a car at a precise location involves techniques and skills different from those exercised when cruising on a road.

Asymptotic Stabilization of Fixed Postures I

Asymptotic stabilization of equilibria (or fixed points) cannot be achieved by using continuous feedbacks which depend on the state only (i.e., continuous pure-state feedbacks).

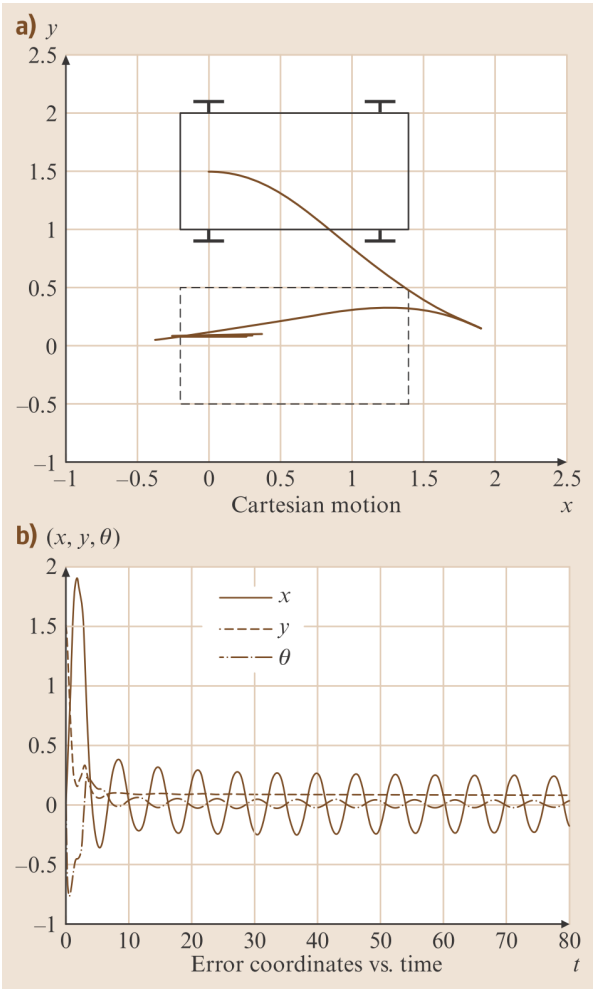
To solve this problem three major types of controls have been considered:

1. *Continuous time-varying feedbacks*, which, besides from depending on the state ξ, depend also on the exogenous time variable (i.e., $u(\xi, t)$, instead of instead of $u(\xi)$ for classical feedbacks).

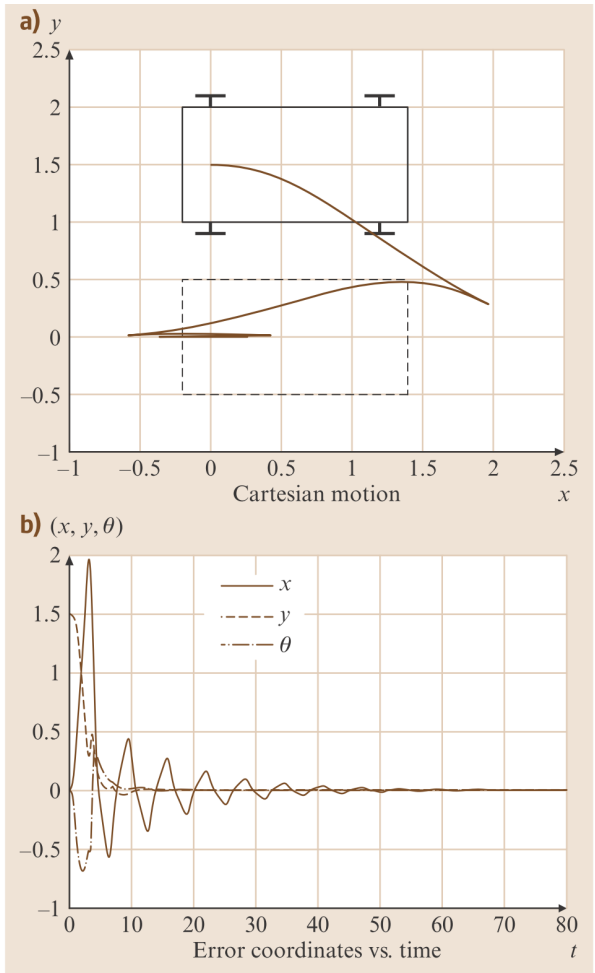
2. *Discontinuous feedbacks*, in the classical form $u(\xi)$, the function u is not continuous at the equilibrium that one wishes to stabilize.

3. *Hybrid discrete/continuous feedbacks*, it is mostly composed of time-varying feedbacks, either continuous or discontinuous (this class of feedbacks is not defined as precisely as the other two sets of controls).

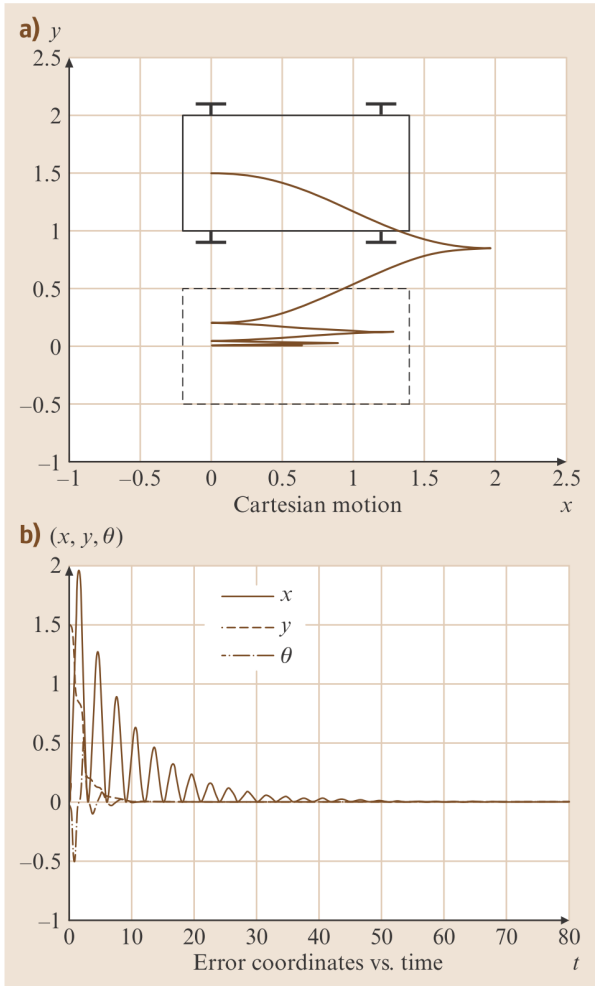
Asymptotic Stabilization of Fixed Postures II



Lipschitz-continuous controller



Continuous time-varying feedback



Hybrid discrete/continuous controller

Source: Springer Handbook of Robotics, Springer, 2016.

Asymptotic Stabilization of Fixed Postures III

Disadvantages of the feedback control of nonholonomic robots:

- A shortcoming of a Lipschitz-continuous controller, is that the system state converges to zero quite slowly.
- The nonlinear time-varying is not robust with respect to modeling errors.
- Stability of the origin and convergence to this point can be jeopardized by arbitrarily small modeling errors, even in the absence of measurement noise.
- The system solutions end up oscillating in the neighborhood of the origin, instead of converging to the origin.
- Robustness of the stability property against modeling errors, and control discretization and delays could only be achieved with Lipschitz-continuous feedbacks which, yield slow convergence.