16-311-Q Introduction to Robotics

Lecture 12: Feedback-Based Control IV

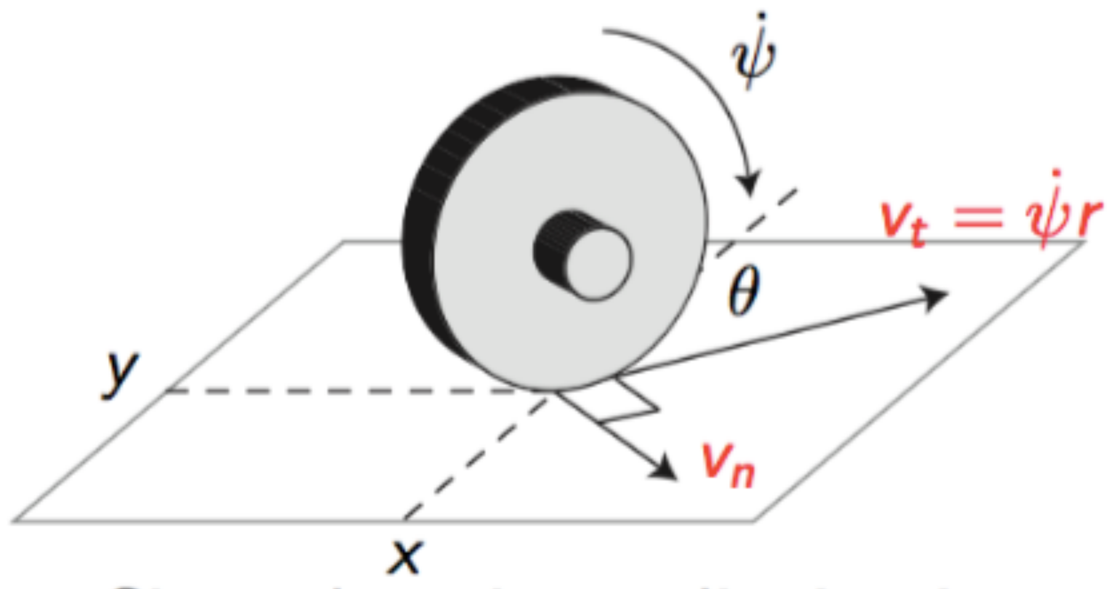
Instructor:

Gianni A. Di Caro

جامعة كارنيجي ميلون في قطر
Carnegie Mellon University Qatar

BACK TO NON HOLONOMIC CONSTRAINTS

- Each wheel introduces in the system a non-holonomic constraint since it does not allow normal translations to the rolling direction.
- The wheel constraints the instant robot mobility, without typically reducing the configuration space (eg: parking parallel).



Without constraints:

$$\begin{cases} \dot{x} = v_t \cos\theta + v_n \cos\left(\theta + \frac{\pi}{2}\right) \\ \dot{y} = v_t \sin\theta + v_n \sin\left(\theta + \frac{\pi}{2}\right) \end{cases}$$

Since there is no slipping in normal direction $v_n = 0$:

$$\begin{cases} \dot{x} = v_t \cos\theta \\ \dot{y} = v_t \sin\theta \end{cases} \Leftrightarrow \tan\theta = \frac{\dot{y}}{\dot{x}} \Leftrightarrow \left[\begin{array}{l} \dot{x}\sin\theta - \dot{y}\cos\theta = 0 \\ \text{Mobility contraint} \end{array} \right]$$

PFAFFIAN FORM OF CONSTRAINTS

- Constraint vector equation: $a(q)\dot{q} = 0$ (1 wheel).
- Constraints matrix equation: $A(q)\dot{q} = 0$ (N wheels).
- A constraint that can be written as $A(q)\dot{q} = 0$ is said *Pfaffian Constraint*

Non-holonomic Contraint

- It cannot be fully integrated
- It cannot be written in the configuration space
- They do not restrict the space of configurations but the instant robot mobility.

Allowable speeds

They may be generated by a matrix $G(q)$ such that:

$$\begin{aligned} \mathrm{Im}(G(q)) &= \mathrm{Ker}(A(q)), \quad \forall q \in \mathbb{C} \\ \mathbb{C} &= \mathbb{R}^N \quad \text{config. space} \end{aligned}$$

General formulation

$$\dot{q} = G(q)v$$

- It represents the allowable directions of motion in the configuration space (allowable velocities)
- It binds speeds in the operational space with speeds in the configuration space

INTEGRABILITY CONDITIONS

Integrability condition

Given the kinematic contraint in Pfaffian Form:

$$\mathbf{a}^{\mathsf{T}}(\mathbf{q})\dot{\mathbf{q}} = \sum_{j=1}^{n} a_j(\mathbf{q})\dot{q}_j = 0$$

In order for it to be holonomic there must be a scalar function $h(\mathbf{q})$ and an integration factor $\gamma(\mathbf{q}) \neq 0$ such that:

$$\gamma(\mathbf{q})a_j(\mathbf{q}) = \frac{\partial h(\mathbf{q})}{\partial q_j}$$

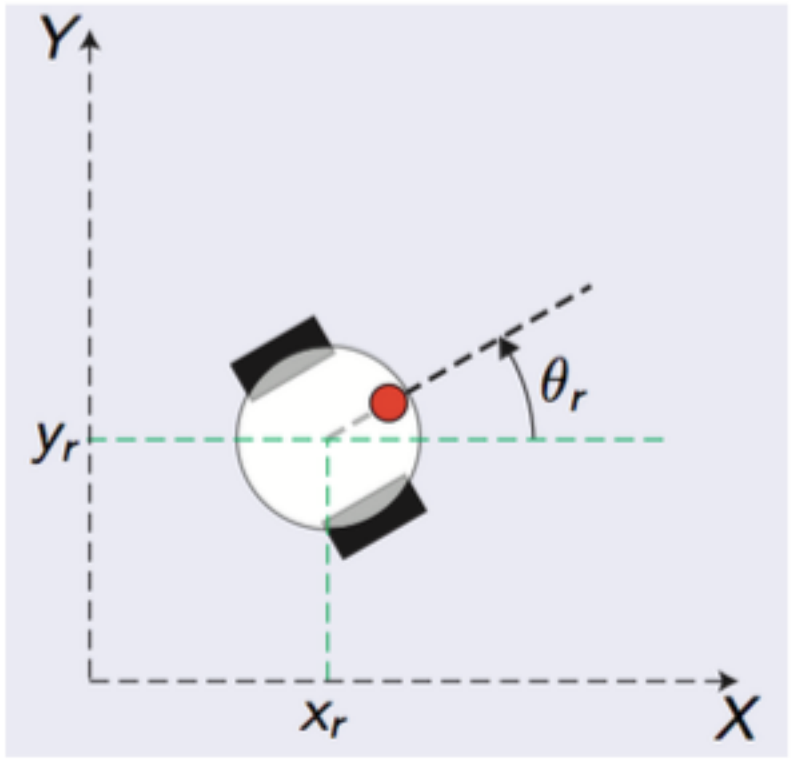
Schwarz **Theorem: more useful in practice because it does not appear $h(\mathbf{q})$**

$$\frac{\partial(\gamma a_k)}{\partial q_j} = \frac{\partial(\gamma a_j)}{\partial q_k} \qquad j, k = 1, ..., n, \quad j \neq k \tag{1}$$

PFAFFIAN CONSTRAINTS FOR THE UNICYCLE

- The configuration is described by $q = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$
- Constraint: $\dot{x}\sin\theta - \dot{y}\cos\theta = 0$
- Pfaffian Form: $A(q)\dot{q} = 0$ with:

$$\begin{cases} A(q) &= [\sin\theta, -\cos\theta, 0] \\ q &= [x, y, \theta]^T \end{cases}$$

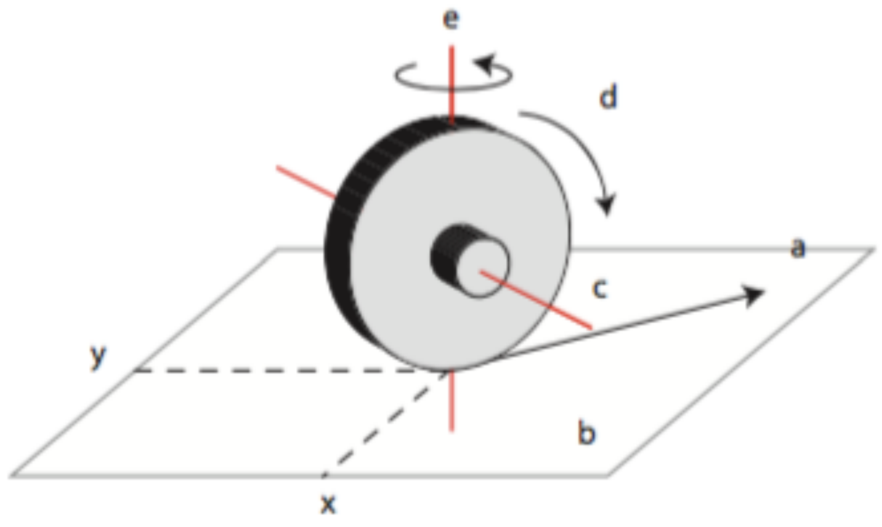


$$\mathrm{Ker}(A(q)) = span\left(\begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}\right) = \mathrm{Im}(G(q))$$

UNICYCLE KINEMATIC MODEL

Unicycle Kinematic Model

$$\dot{q} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix} v + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \omega = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$$



- v: the linear velocity of the contact point between the wheel and the ground and is equal to the product between angular velocity of the wheel around its horizontal axis and the radius of the wheel
- ω: angular velocity of the robot, equals to the angular velocity of the wheel around the vertical axis

Control inputs

By acting on v and ω it is possible to modify the robot configuration

PLANNING (A PATH, A TRAJECTORY)

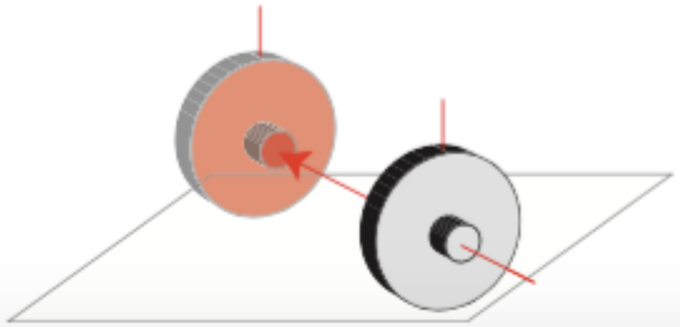
It's an inverse kinematic problem + Require motion control

Planning for a WMR

- Problem: determining a trajectory in the configuration space to take the robot from a certain initial configuration to a final configuration, both feasible
- The initial and final configurations (boundary conditions) and *any* point of the trajectory must be compatible with the kinematic constraints of the robot

Definition

A trajectory is not feasible if it requires the robot to perform motion incompatible with its kinematic constraints.

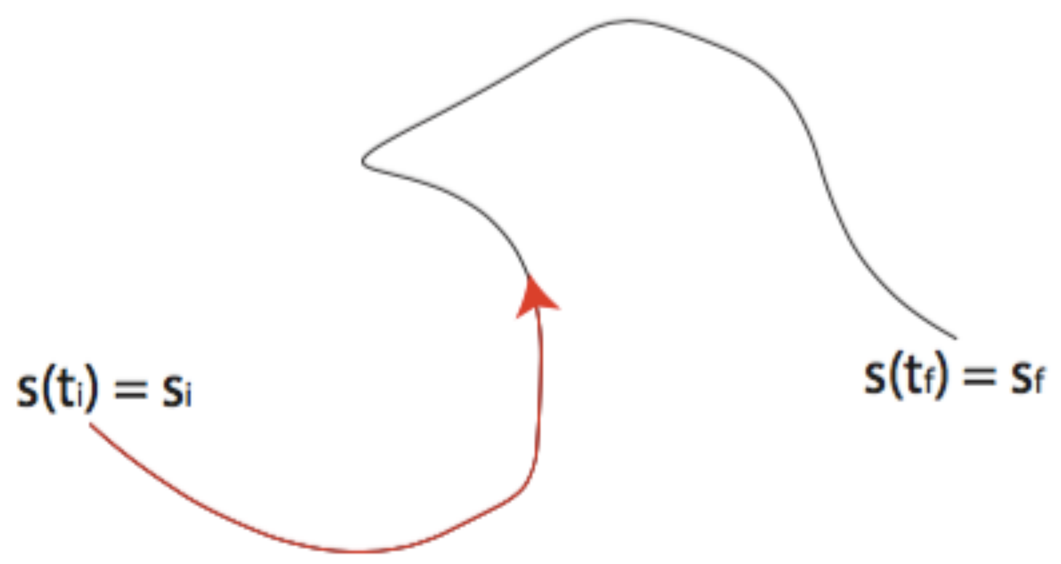


Example: unicycle can not have lateral translational trajectories.

PLANNING: SPACE-TIME COMPONENTS

Space-time separation of the trajectory

- We want to plan a trajectory $q(t)$ for t belongs to $[t_i, t_f]$ that take the robot from an initial configuration $q(t_i) = q_i$ to a final configuration $q(t_f) = q_f$
- We assume no obstacles



The trajectory $q(t)$ can be decomposed in:

- a path $q(s)$, with $\dfrac{dq(s)}{ds} \neq 0, \forall s$
- a motion law $s = s(t)$, with $s_i \leq s \leq s_f$, with $\begin{cases} s(t_i) & = & s_i \\ s(t_f) & = & s_f \end{cases}$
  s monotonic, i.e. $\dot{s}(t) \geq 0$
- Typical choice for s is the *curvilinear abscissa* along the path: $\begin{cases} s_i & = & 0 \\ s_f & = & L \end{cases}$

FEASIBILITY OF A PATH

Space-time separation of the trajectory $\dot{q} = \frac{dq}{dt} = \frac{dq}{ds}\dot{s} = q'\dot{s}$

- q' has the direction of the tangent to the path in the configurations space oriented for growing s
- $\dot{s}$ is a scalar which modulates the intensity

Form the Pfaffian form of the nonholonomic constraints we get the **feasability condition** of the geometric path:

$$\begin{cases} A(q)\dot{q} & = & A(q)q'\dot{s} = 0 \\ \dot{s} & > & 0, \forall t \in [t_i, t_f] \end{cases}$$
$$\Downarrow$$
$$A(q)q'$$

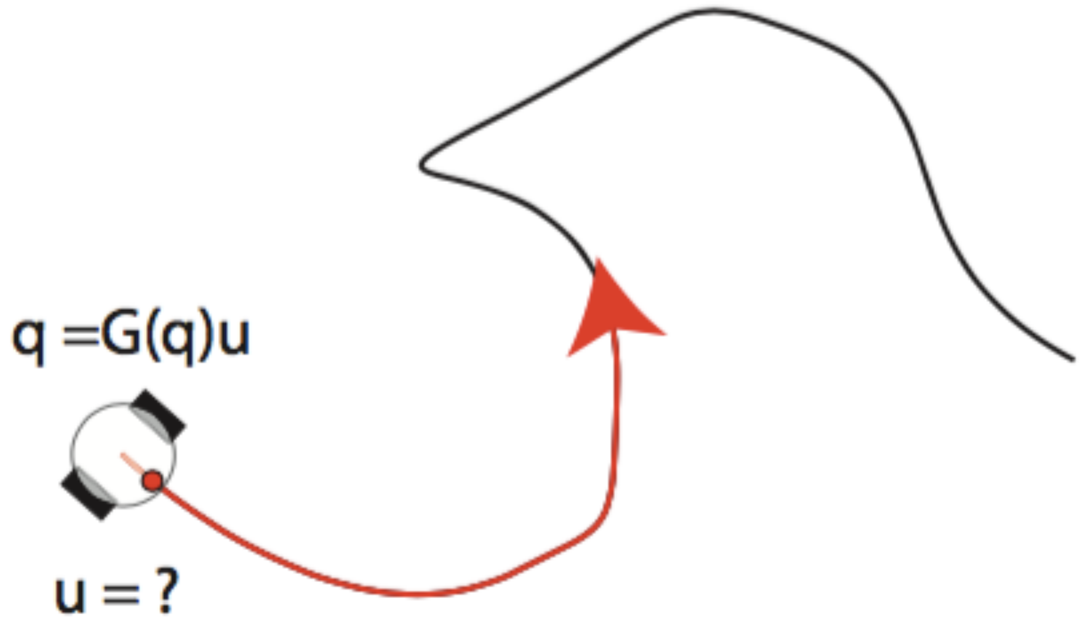
A feasable path is given by: $q' = G(q)\tilde{u}$

FROM GEOMETRIC TO VELOCITY INPUTS

- *Geometric inputs* $\tilde{u}$: determine the geometry path
- Chosen $\tilde{u}$ feasable, we define the *motion law* $s(t)$ to define how fast the robot run across the path

Problem

How to combine the geometric path with known inputs $\tilde{u}$ and the motion law in order to obtain the control inputs for the robot?



$$q' = G(q)\tilde{u}(s)$$
$$\Downarrow$$
$$\frac{dq}{ds}\dot{s} = G(q)\tilde{u}(s)\dot{s}$$
$$\Downarrow$$
$$\begin{cases} \dot{q} & = & G(q)\tilde{u}(s)\dot{s} \\ \dot{q} & = & G(q)u(t) \end{cases}$$
$$\Downarrow$$
$$\tilde{u}(s)\dot{s} = u(t)$$

FEASIBLE GEOMETRIC PATH FOR UNICYCLE

Example: unicycle

For the unicycle, the wheel's nonholonomic constraints imply the following feasibility condition for the geometric path:

$$[\sin\theta, -\cos\theta, 0]\, q' = x' \sin\theta - y' \cos\theta = 0$$

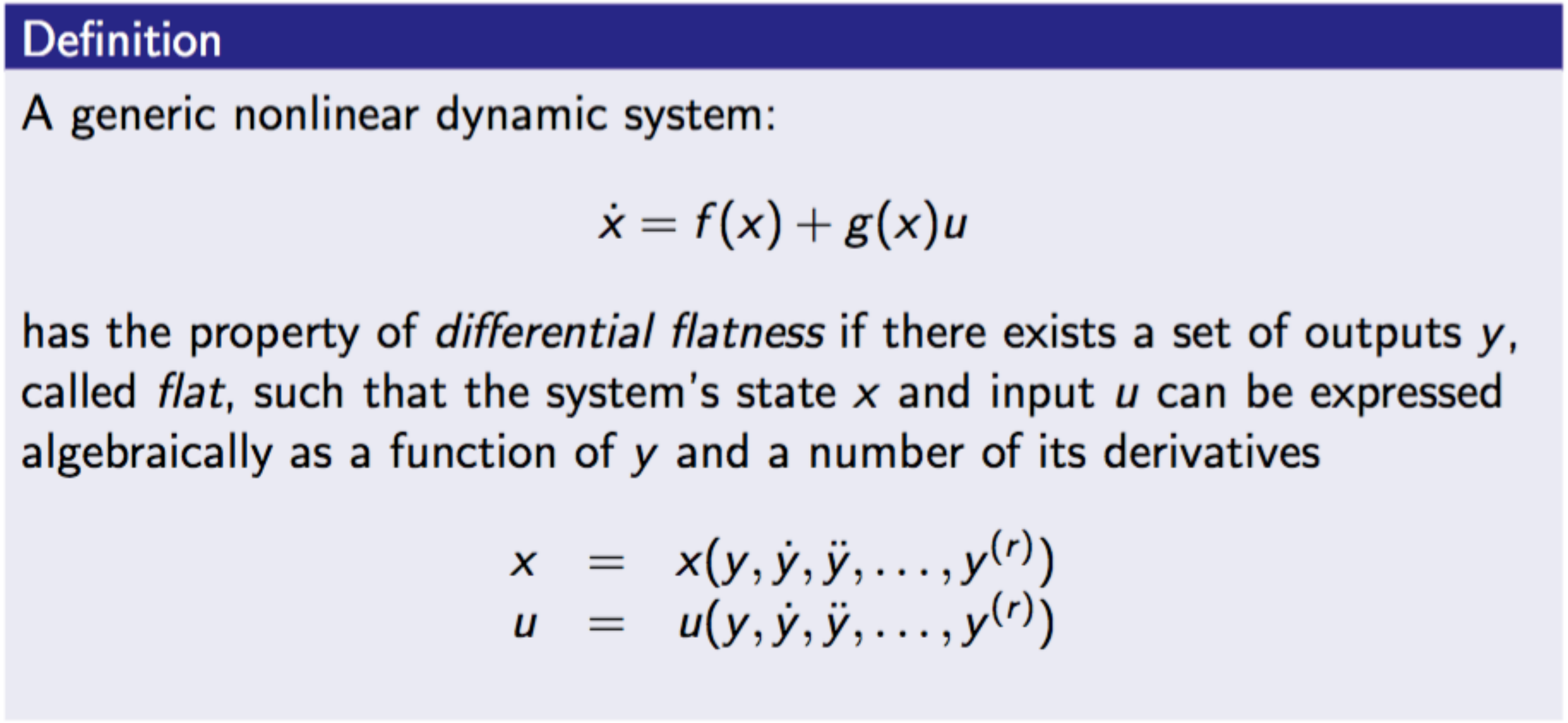
- The condition highlights the fact that the Cartesian speed must be oriented along the direction of motion (no lateral slip)
- The feasible paths for the unicycle are given by:

$$\begin{cases} x' &= \tilde{v}\cos\theta \\ y' &= \tilde{v}\sin\theta \\ \theta' &= \tilde{\omega} \end{cases}$$

- The kinematic inputs are obtained from the geometric ones:

$$\begin{cases} v(t) &= \tilde{v}\dot{s} \\ \omega(t) &= \tilde{\omega}\dot{s} \end{cases}$$

PATH PLANNING EXPLOITING DIFFERENTIAL FLATNESS



Definition

A generic nonlinear dynamic system:

$$\dot{x} = f(x) + g(x)u$$

has the property of *differential flatness* if there exists a set of outputs y, called *flat*, such that the system's state x and input u can be expressed algebraically as a function of y and a number of its derivatives

$$\begin{aligned} x &= x(y, \dot{y}, \ddot{y}, \ldots, y^{(r)}) \\ u &= u(y, \dot{y}, \ddot{y}, \ldots, y^{(r)}) \end{aligned}$$

- Assigning a trajectory for a dynamic system equals to assign its outputs, in space and/or time. In particular, for a mobile robot, an output is a pose in the environment, while a trajectory is a sequence of poses, possibly specified through a time parameter.

- Therefore, in the case of differential flatness, once an output trajectory is assigned in terms of some variable y, the associated trajectory of the state x and history of control inputs u are uniquely determined.

DIFFERENTIAL FLATNESS FOR UNICYCLE

- For the unicycle/differential and bicycle cases the Cartesian coordinates of the robot reference point are *flat outputs*. This can be established both for *trajectory following* (depending on s) and *trajectory tracking* (depending on t).

- In the case of trajectory tracking, a Cartesian trajectory is provided as *outputs* to be followed in time: $(x_d(t), y_d(t))$.

- The associated *state* trajectory is: $\mathbf{q}_d(t) = \left[x_d(t) \;\; y_d(t) \;\; \theta_d(t)\right]^T$, where the *orientation* is $\theta_d(t) = atan2(\dot{y}_d(t), \dot{x}_d(t)) + k\pi$

- The *kinematic inputs* (the controls given to the wheels) that drive the robot along the Cartesian trajectory are obtained from the kinematic equations:

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

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

It is apparent that both state and control inputs are functions of the outputs and of their derivatives, up to order $r = 2$

DIFFERENTIAL FLATNESS FOR UNICYCLE

For the unicycle (and bicycle) models, the **Cartesian coordinates of the robot's reference point are flat outputs,** both for path (depending on s) and trajectory following (depending on t)

- In case of path following, a path is provided as a set of points to be followed in space:

$$(x(s), y(s)), \quad s \in [0, 1]$$

- The associate state path in configuration space:

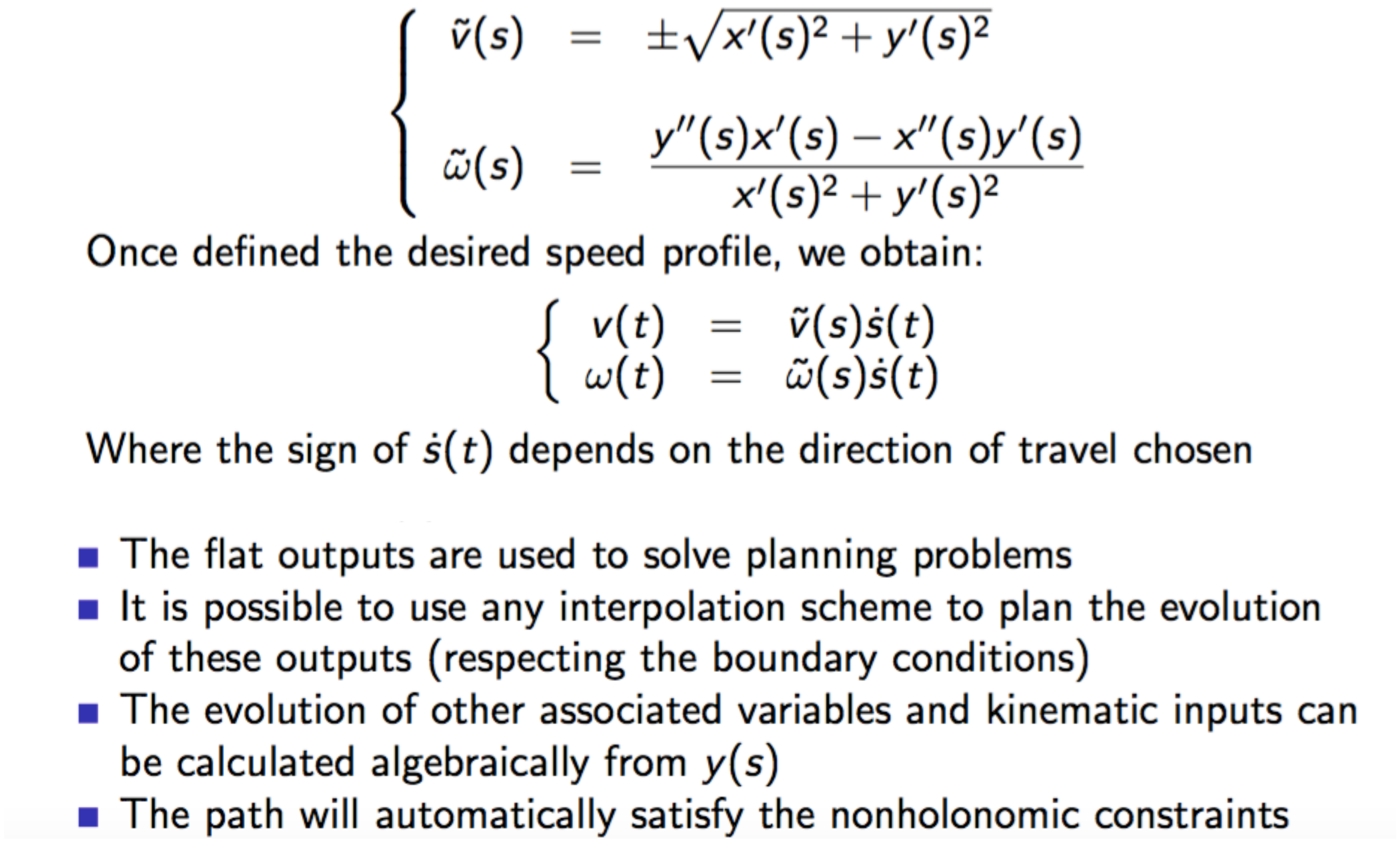
$$\boldsymbol{q}(s) = [x(s), y(s), \theta(s)]^T, \quad s \in [0, 1]$$

- The orientation of the robot is derived using the flat outputs based on the constraint equations (that guarantee the kinematic feasibility)

$$\theta(s) = \text{atan2}(\dot{y}(s), \dot{x}(s)) + k\pi \quad k = 0 \text{ forward motion}, \;\; k = 1 \text{ backward motion}$$

DIFFERENTIAL FLATNESS FOR UNICYCLE

The kinematic inputs (velocity controls to wheels) that would move the robot along the path ($x(s)$, $y(s)$) are also obtained from kinematic equations using flat outputs:



$$\begin{cases} \tilde{v}(s) & = & \pm\sqrt{x'(s)^2 + y'(s)^2} \\ \tilde{\omega}(s) & = & \dfrac{y''(s)x'(s) - x''(s)y'(s)}{x'(s)^2 + y'(s)^2} \end{cases}$$

Once defined the desired speed profile, we obtain:

$$\begin{cases} v(t) & = & \tilde{v}(s)\dot{s}(t) \\ \omega(t) & = & \tilde{\omega}(s)\dot{s}(t) \end{cases}$$

Where the sign of $\dot{s}(t)$ depends on the direction of travel chosen

- The flat outputs are used to solve planning problems
- It is possible to use any interpolation scheme to plan the evolution of these outputs (respecting the boundary conditions)
- The evolution of other associated variables and kinematic inputs can be calculated algebraically from $y(s)$
- The path will automatically satisfy the nonholonomic constraints

DIFFERENTIAL FLATNESS PLANNING

Problem: unicycle

Plan the trajectory to take the robot from an initial configuration $q_i = [x_i, y_i, \theta_i]$ to a final configuration $q_f = [x_f, y_f, \theta_f]$

Solution

Since x, y are *flat* outputs, they can be used to plan the trajectory

- I suppose to parameterize the curve with a parameter $s = [0, 1]$
- Let's use a *cubic polynomial*:

$$\begin{aligned} x(s) &= s^3 x_f - (s-1)^3 x_i + \alpha_x s^2 (s-1) + \beta_x s (s-1)^2 \\ y(s) &= s^3 y_f - (s-1)^3 y_i + \alpha_y s^2 (s-1) + \beta_y s (s-1)^2 \end{aligned}$$

that automatically satisfy the boundary conditions on x, y

$$\begin{aligned} x(0) &= x_i \qquad & x(1) &= x_f \\ y(0) &= y_i \qquad & y(1) &= y_f \end{aligned}$$

DIFFERENTIAL FLATNESS PLANNING

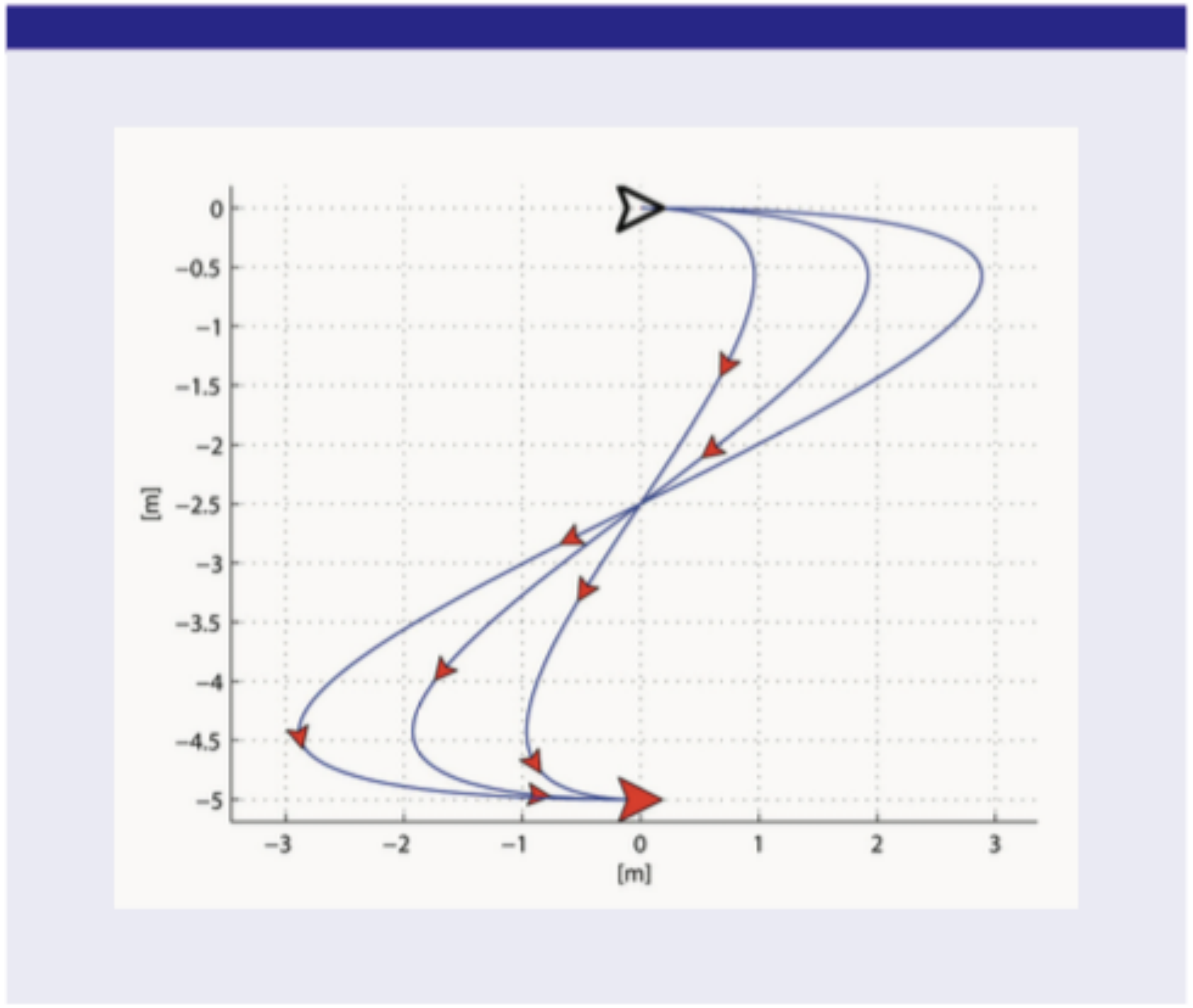
Since the orientation depends on each point on the values of x', y', also the boundary conditions on θ must be met:

$$\begin{aligned} x'(0) &= k_i \cos\theta_i & \quad x'(1) &= k_f \cos\theta_f \\ y'(0) &= k_i \sin\theta_i & \quad y'(1) &= k_f \sin\theta_f \end{aligned}$$

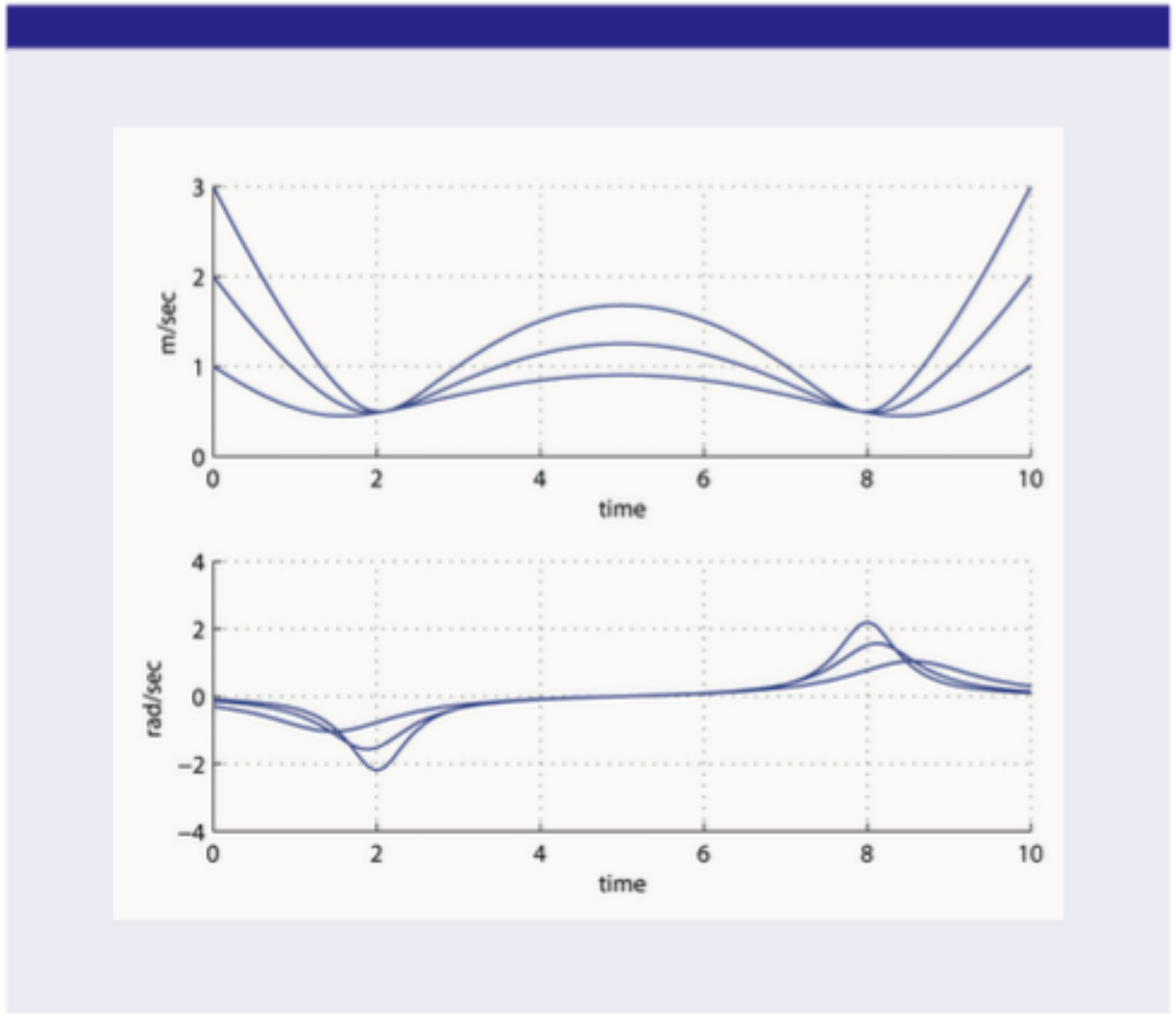
- $k_i > 0$ e $k_f > 0$ are free parameters of the same sign
- They represent the initial and final geometric speed
- They influence the type of path obtained
- They are used to compute the values α_x, α_y, β_x, β_y. Let's choose $k_i = k_f = k$

$$\begin{bmatrix} \alpha_x \\ \alpha_y \end{bmatrix} = \begin{bmatrix} k\cos\theta_f - 3x_f \\ k\sin\theta_f - 3y_f \end{bmatrix} \quad \begin{bmatrix} \beta_x \\ \beta_y \end{bmatrix} = \begin{bmatrix} k\cos\theta_i + 3x_i \\ k\sin\theta_i + 3y_i \end{bmatrix}$$

DIFFERENTIAL FLATNESS PLANNING



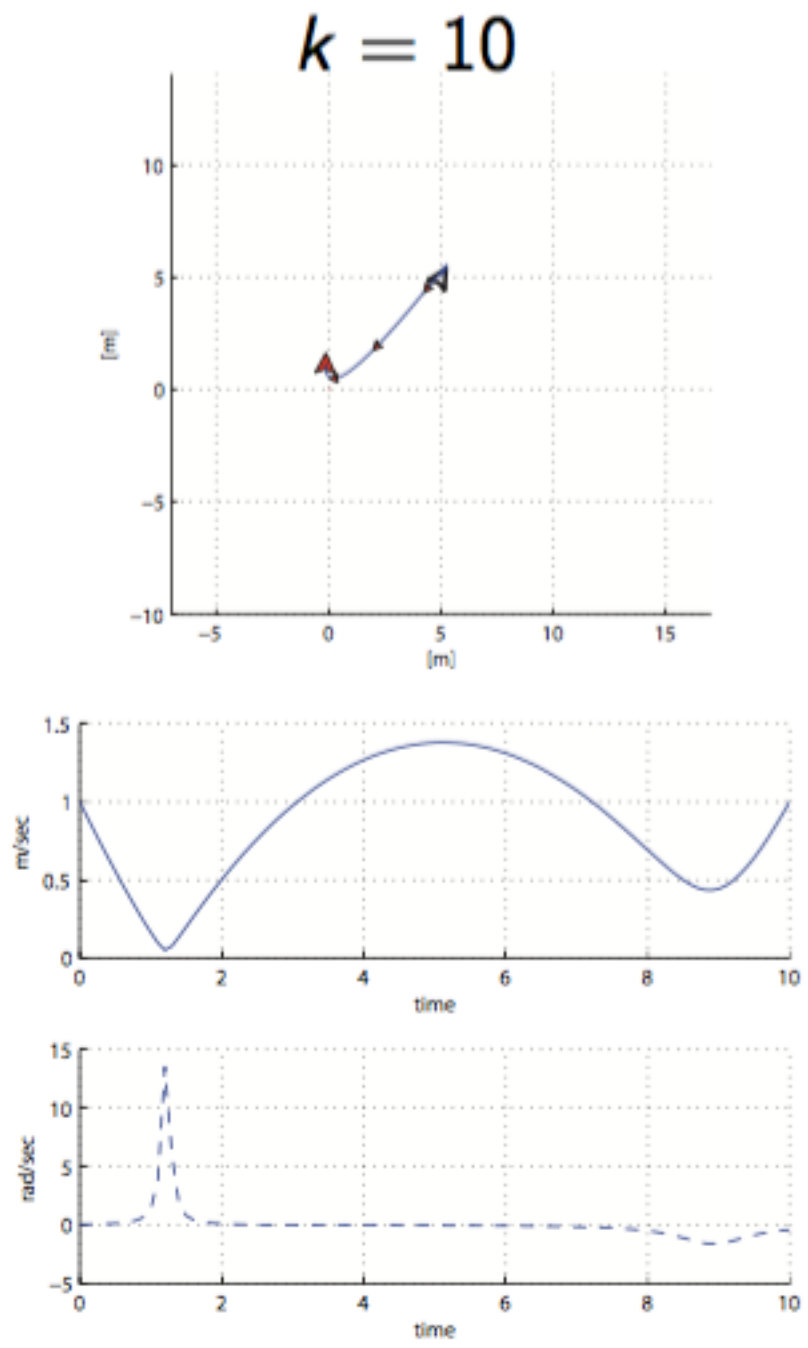
$$\begin{aligned} q_i &= [0, 0, 0] \\ q_f &= [0, -5, 0] \end{aligned}$$

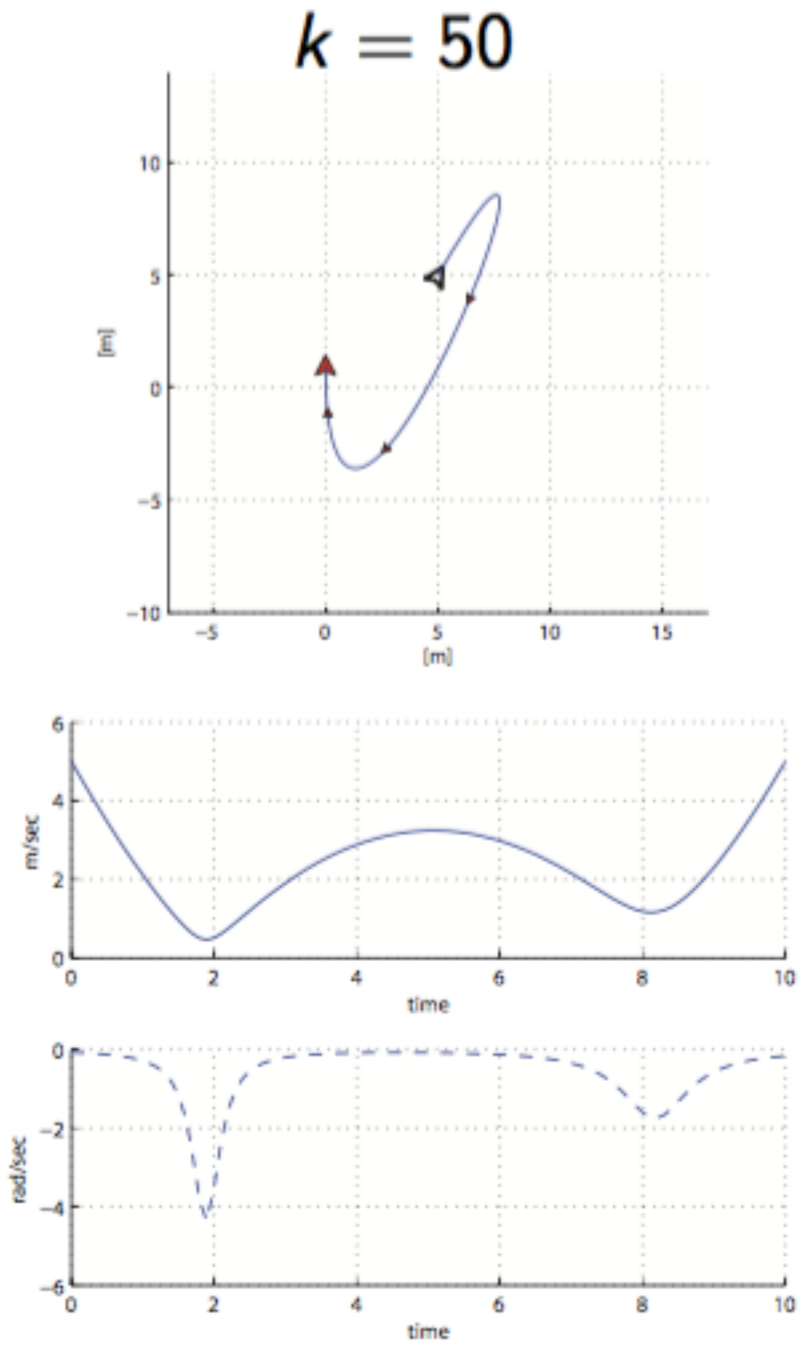


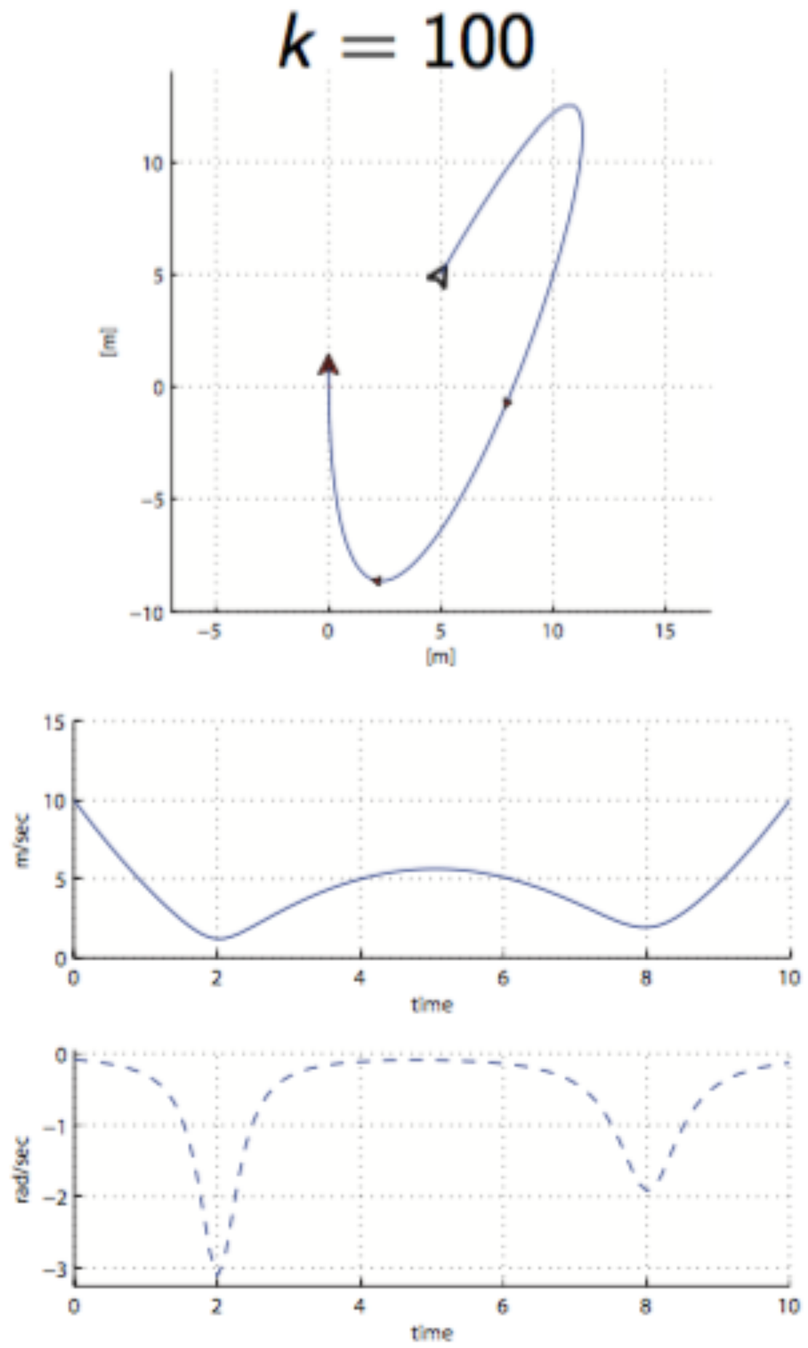
$$k = 10,\ k = 20,\ k = 30$$

DIFFERENTIAL FLATNESS PLANNING

Parking: $q_i = \left[5, 5, \frac{\pi}{3}\right]^T \Rightarrow q_f = \left[0, 1, \frac{\pi}{2}\right]^T$







TRAJECTORY TRACKING

- For the trajectory tracking problem be feasible, the trajectory $(x_d(t), y_d(t))$ must be defined to be compliant with the holonomic and non-holonomic constraints of the robot
- Let's focus on a *unicycle / differential drive* $\rightarrow$ the trajectory must be of the following form, for some specified inputs for v_d and ω_d, to be admissible:

$$\dot{x}_d = v_d \cos(\theta_d)$$
$$\dot{y}_d = v_d \sin(\theta_d)$$
$$\dot{\theta}_d = \omega_d$$

- Based the property of *differential flatness*, these equations can be solved for θ_d and for the reference inputs:

$$\theta_d(t) = atan2(\dot{y}_d(t), \dot{x}_d(t)) + k\pi \quad (k = 0 \text{ for forward motion, } k = 1 \text{ for backward motion})$$
$$v_d(t) = \pm\sqrt{\dot{x}_d^2(t) + \dot{y}_d^2(t)}$$
$$\omega_d(t) \frac{\ddot{y}_d(t)\dot{x}_d(t) - \ddot{x}_d(t)\dot{y}_d(t)}{\dot{x}_d^2(t) + \dot{y}_d^2(t)}$$

- A feedback-based controller, closed-loop, is needed: using an open-loop approach based on the above equations would not provide satisfactory results in practice.
- The feedback signal can be an error measure e based on the difference between the desired and the current state: $e(t) = \mathbf{q}_d(t) - \mathbf{q}(t)$, $\mathbf{q}_d(t) = \begin{bmatrix} x_d(t) & y_d(t) & \theta_d(t) \end{bmatrix}^T$, $\mathbf{q}(t) = \begin{bmatrix} x(t) & y(t) & \theta(t) \end{bmatrix}^T$

USE OF DIFFERENTIAL FLATNESS

- In the case of trajectory tracking, a Cartesian trajectory is provided as *outputs* to be followed in time: $(x_d(t), y_d(t))$.

- The associated *state* trajectory is: $\boldsymbol{q}_d(t) = \left[x_d(t) \;\; y_d(t) \;\; \theta_d(t)\right]^T$, where the *orientation* is $\theta_d(t) = atan2(\dot{y}_d(t), \dot{x}_d(t)) + k\pi$

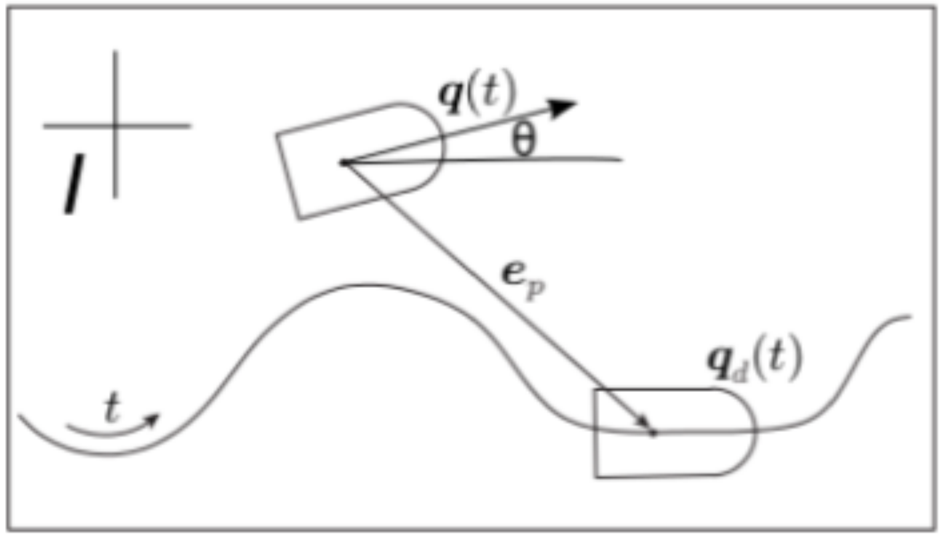
- The *kinematic inputs* (the controls given to the wheels) that drive the robot along the Cartesian trajectory are obtained from the kinematic equations:

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

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

It is apparent that both state and control inputs are functions of the outputs and of their derivatives, up to order $r = 2$

Error representation



- The tracking error vector e is conveniently expressed in terms of its *projections on the rotated reference frame of the robot wrt to the inertial frame*. In this way the positional part of the error is the Cartesian component of the error expressed in a reference frame aligned with the current orientation of the robot:

$$\boldsymbol{e} = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_d - x \\ y_d - y \\ \theta_d - \theta \end{bmatrix} = \begin{bmatrix} (x_d - x)\cos(\theta) + (y_d - y)\sin(\theta) \\ -(x_d - x)\sin(\theta) + (y_d - y)\cos(\theta) \\ \theta_d - \theta \end{bmatrix}$$

ERROR DYNAMICS

- Differentiating wrt time and using kinematic equations for expressing $x(t), y(t), \theta(t), x_d(t), y_d(t), \theta_d(t)$, the error dynamics becomes:

$$\dot{e}_1 = v_d \cos(e_3) - v + e_2\omega$$
$$\dot{e}_2 = v_d \sin(e_3) - e_1\omega$$
$$\dot{e}_3 = \omega_d - \omega$$

- The following (invertible) input transformation (where u_1 and u_2 are be the actual input control signals) is aimed at setting robot's velocities bringing to zero the error:

$$v = v_d \cos(e_3) - u_1$$

$$\omega = \omega_d - u_2$$

- Resulting tracking-error dynamics:

$$\dot{\boldsymbol{e}} = \begin{bmatrix} 0 & \omega_d & 0 \\ -\omega_d & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \boldsymbol{e} + \begin{bmatrix} 0 \\ \sin(e_3) \\ 0 \end{bmatrix} v_d + \begin{bmatrix} 1 & -e_2 \\ 0 & e_1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

- Non-linear, time-varying dynamics: The first term is linear (in $\boldsymbol{e}$), the second is not linear because of $sin(e_3)$, and the third is not linear because of the multiplication between variables $\boldsymbol{u}$ and $\boldsymbol{e}$. The first and the second term are, in general, time-varying because of the presence of the reference inputs $v_d(t)$ and $\omega_d(t)$.

LINEARIZATION

- Approximate linearization of the error dynamics e around the reference trajectory, around which $\boldsymbol{e} \approx 0$. The linearization sets $\sin(e_3) = e_3$, and the input signals u_1 and u_2 are evaluated on the trajectory (such that the coefficients of their components vanish):

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

- The linear feedback is set as follows (it reduces linear and angular velocities proportionally to the value of the corresponding error components):

$$u_1 = -k_1 e_1$$

$$u_2 = -k_2 e_2 - k_3 e_3$$

- The resulting closed-loop linearized dynamics:

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

STABILITY

- letting

$$k_1 = k_3 = 2\zeta a \qquad k_2 = \frac{a^2 - \omega_d^2}{v_d}$$

with $a > 0$, $\zeta \in (0,1)$, the characteristic polynomial of $\boldsymbol{A}(t)$ becomes time-invariant

$$p(\lambda) = \underbrace{(\lambda + 2\zeta a)}_{\text{real negative eigenvalue}}\underbrace{(\lambda^2 + 2\zeta a \lambda + a^2)}_{\text{pair of complex eigenvalues with negative real part}}$$

- caveat: this does not guarantee asymptotic stability, unless v_d and ω_d are constant (as on circles and lines); even in this case, asymptotic stability of the unicycle is not global (indirect Lyapunov method)

NON-LINEAR IS STABLE

- the actual velocity inputs v, ω are obtained plugging the feedbacks u_1, u_2 in the input transformation
- note:$(v,\ \omega) \to (v_d,\ \omega_d)$ as $\boldsymbol{e} \to \mathbf{0}$ (pure feedforward)
- note: $k_2 \to \infty$ as $v_d \to 0$, hence this controller can only be used with persistent cartesian trajectories (stops are not allowed)
- global stability is guaranteed by a nonlinear version

$$
\begin{aligned}
u_1 &= -k_1(v_d, \omega_d)\, e_1 \\
u_2 &= -k_2\, v_d \frac{\sin e_3}{e_3}\, e_2 - k_3(v_d, \omega_d)\, e_3
\end{aligned}
$$

if k_1, k_3 bounded, positive, with bounded derivatives

Extensive mathematical details can be found in the reference book: *Robotics: Modelling, Planning and Control*, Siciliano, B., Sciavicco, L., Villani, L., Oriolo, G., Springer Series in Advanced Textbooks in Control and Signal Processing, pp. 632, 2009.