PATH AND TRAJECTORY PLANNING

viernes, 11 de octubre de 2019 11:11 a. m.

- Definitions:
  - Path in C
    - $q = q(s); s \in [s_i, s_f]$
      - s: path parameter (e.g. arc length)
  - Trajectory in C
    - $q = q(t); t \in [t_i, t_f]$
      - t: time
    - A trajectory can also be defined as a path and a timing law
      - $q = q(s)$
      - $s = s(t)$
        - $s(t_i) = s_i$
        - $s(t_f) = s_f$
    - This allows the problem to be divided into the **geometry** (path) of the motion and the **speed** (timing law) of the motion.
  - **Path Planning**: given q_i, q_f in C, find a **feasible** path (a path that the robot can follow) that goes from the initial configuration to the final one.
    - $q(s_i) = q_i$
    - $q(s_f) = q_f$
  - **Trajectory Planning**: given q_i, q_f in C, find a **feasible** trajectory that goes from the initial configuration to the final one over a certain time interval.
    - $q(t_i) = q_i$
    - $q(t_f) = q_f$
  - Both path and trajectory planning occur in the absense of obstacles. In the presence of obstacles the problem is called **motion planning**.
    - **Direct approach**: plan directly a trajectory
    - **Decoupled approach**: plan first a path and then a timing law. Convenient because the speed limits on the actuators do not limit the path, only the timing law.
  - **Feasible:** WMRs are constrained by $A^T(q)\dot{q} = 0$
    - Trajectories: $q(t)$ such that:
      - $\dot{q}(t) \in \mathcal{N}(A^T(q))\ \forall t$
    - Paths:
      - $\dot{q} = \frac{dq}{dt} = \frac{\partial q}{\partial s}\frac{ds}{dt}$
        - $\frac{\partial q}{\partial s} = q'$: geometric tangent. If the s parameter is the arc length, then q' is a unit vector.
        - $\frac{ds}{dt}$: speed that modulates $\dot{q}$ along the direction of q'
      - $A^T(q)\dot{q} = 0 \rightarrow A^T(q)q'\dot{s} = 0$
        - Since $\dot{s} \neq 0$, then:
        - $A^T(q)q' = 0$
    - Then the geometric tangents q' are constrained exactly like the $\dot{q}$. In general, paths that have a component along the zero motion line of a robot are not feasible.
    - Analogously to what we did with the kinematic constraints, we can represent the constraint on the geometric tangents as a linear combination of the basis of the nullspace of $A^T(q)$
      - **Geometric version of the kinematic model:**

        - $q' = \sum_{i=1}^{m} g_i(q)\tilde{u}_i$
    - Relation between u_i and $\tilde{u}_i$. Multiply by $\dot{s}$
        - $q'\dot{s} = \sum_{i=1}^{m} g_i(q)\tilde{u}_i\dot{s}$
        - $u_i = \tilde{u}_i\dot{s}$
            - u_i: velocity inputs
            - $\tilde{u}_i$: geometric inputs
            - $\dot{s}$: speed
    - **Trajectory planning**:
        - Decoupled approach:
            - Choose the geometric inputs in the model
            - Integrate the model to get a path
            - Choose a timing law
        - Direct approach:
            - Choose the velocity inputs in the model
            - Integrate to get a trajectory directly

- **Differential Flatness**:
    - Consider a nonlinear, driftless system
        - $\dot{q} = \sum_{i=1}^{m} g_i(q)u_i$
    - The system is differentialy flat (DF) if there exists a set of outputs $w = h(q)$, called **flat outputs**, such that q and u can be written as algebraic functions of $(w, \dot{w}, \ddot{w}, \dots w^{(r)})$:
        - $\exists w = h(q): q = \alpha(w, \dot{w}, \ddot{w}, \dots w^{(r)}), u = \beta(w, \dot{w}, \ddot{w}, \dots w^{(r)})$
    - In DF systems, knowing the evolution of the FOs allows to reconstruct the evolution of the whole state q and the history of control inputs u. **Also applies to geometric models**. So, knowing the evolution of the FOs over s allows to reconstruct the evolution of q over s (the path) and the history of u over s (the geometric inputs).
    - Example: the unicycle
        - Kinematic model:
            - $\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \cos\theta \\ \sin\theta \\ 0 \end{pmatrix} v + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \omega$
        - Geometric model
            - $\begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix} = \begin{pmatrix} \cos\theta \\ \sin\theta \\ 0 \end{pmatrix} \tilde{v} + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tilde{\omega}$
        - Flat outputs are:
            - $w = \begin{pmatrix} x \\ y \end{pmatrix}$
            - x, y are known directly.
        - Can we reconstruct θ?
            - $\theta = ATAN2(\dot{y}, \dot{x}) + k\pi; k = 0,1$
            - The two choices for k account for the fact that the same cartesian path may be followed moving forward ($k = 0$) or backward ($k = 1$). If the initial orientation is assigned only one k is correct.
        - Now that we have reconstructed all the states, we need to reconstruct the inputs.
            - $v = \pm\sqrt{\dot{x}^2 + \dot{y}^2}$
            - $\omega = \dot{\theta} = \frac{1}{1 + \frac{\dot{y}^2}{\dot{x}^2}} \frac{\ddot{y}\dot{x} - \dot{y}\ddot{x}}{\dot{x}^2} = \frac{(\ddot{y}\dot{x} - \dot{y}\ddot{x})}{\dot{x}^2 + \dot{y}^2}$

- If $\dot{x} = \dot{y} = 0$ then we cannot reconstruct the ω or θ.
    - Interpretation: We have a movie of the unicycle motion from the ceiling, so we see its position and its motion but θ is hidden. By knowing the motion of the center we can reconstruct the whole state and inputs. If the robot is not moving we cannot reconstruct θ and ω
  - Exercise: Bicycle.
    - Kinematic model
      - $\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi} \end{pmatrix} = \begin{pmatrix} \cos\theta \\ \sin\theta \\ \frac{\tan\phi}{l} \\ 0 \end{pmatrix} v + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \omega$
    - Geometric model
      - $\begin{pmatrix} x' \\ y' \\ \theta' \\ \phi' \end{pmatrix} = \begin{pmatrix} \cos\theta \\ \sin\theta \\ \frac{\tan\phi}{l} \\ 0 \end{pmatrix} \tilde{v} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \widetilde{\omega}$
    - Flat outputs
      - $w = \begin{pmatrix} x \\ y \end{pmatrix}$
    - Reconstruct θ and ϕ
      - $\theta = ATAN2(\dot{y}, \dot{x}) + k\pi; k = 0,1$
      - $\phi = ATAN2(l\omega, v)$
    - But
      - $v = \pm\sqrt{\dot{x}^2 + \dot{y}^2}$
      - $\omega = \dot{\theta} = \frac{(\ddot{y}\dot{x} - \dot{y}\ddot{x})}{\dot{x}^2 + \dot{y}^2}$
    - It is important to notice that the value of ϕ changes with the choice of the velocity input v
  - Example: the $(2, n)$ chained form
    - Flat outputs:
      - $w = \begin{pmatrix} z_1 \\ z_n \end{pmatrix}$
    - Reconstruct state:
      - $z_{k-1} = \dot{z}_k / \dot{z}_1$
      - For $k \in [2, n]$
    - Reconstruct inputs:
      - $v_1 = \dot{z}_1$
      - $v_2 = \dot{z}_2$
      - $\dot{z}_{k-1} = \frac{\ddot{z}_k \dot{z}_1 - \dot{z}_k \ddot{z}_1}{\ddot{z}_1^2}$
      - For $k \in [2, n]$
  - If a system is DF then it can be expressed in chained form and viceversa.
- **Path Planninng**:
  - For path planning we consider the geometric version of the kinematic model:
    - $q' = \sum_{i=1}^{m} g_i(q)\tilde{u}_i$
      - $q' = \frac{\partial q}{\partial s}$: geometric tangent
      - $\tilde{u}_i(s) = \frac{u_i(t)}{s}$: geometric inputs
  - **Scheme 1**: With **Differential Flatness** (and parametrized paths): **If the system is flat then we can describe the whole evolution of the state and the inputs from the FOs.**

a. Compute boundary conditions
   - $w_i = h(q_i)$
   - $w_f = h(q_f)$

b. Generate a path for the FOs $w \in [w_i, w_f]$: using a geometric path in s.

c. Reconstruct the path for the whole state $q(s)$ and for the geometric inputs $\tilde{u}_i(s)$ using the reconstruction formulas of flatness.

- Example: Unicycle
  - $q_i = \begin{pmatrix} x_i \\ y_i \\ \theta_i \end{pmatrix} \rightarrow q_f = \begin{pmatrix} x_f \\ y_f \\ \theta_f \end{pmatrix}$

  i. Boundary conditions $s \in [0,1]$:
     - $w_i = \begin{pmatrix} x_i \\ y_i \end{pmatrix} w_f = \begin{pmatrix} x_f \\ y_f \end{pmatrix}$
     - $x'(0) = \tilde{v}(0) \cos\theta_i$
     - $y'(0) = \tilde{v}(0) \sin\theta_i$
     - $x'(1) = \tilde{v}(1) \cos\theta_f$
     - $y'(1) = \tilde{v}(1) \sin\theta_f$
     - The initial and final velocities are **free** parameters.

  ii. Generate a path for x and y
     - Cubic polinomials (we can choose any interpolating basis that satisfies the constraints)
       - $x(s) = a_x s^3 + b_x s^2 + c_x s + d_x$
       - $y(s) = a_y s^3 + b_y s^2 + c_y s + d_y$
     - We get:
       - $x(s) = s^3 x_f - (s-1)^3 x_i + \alpha_x s^2 (s-1) + \beta_x s(s-1)$
       - $y(s) = s^3 y_f - (s-1)^3 y_i + \alpha_y s^2 (s-1) + \beta_y s(s-1)$
       - $\begin{pmatrix} \alpha_x \\ \alpha_y \end{pmatrix} = \begin{pmatrix} K\cos\theta_f - 3x_f \\ K\sin\theta_f - 3y_f \end{pmatrix}$
       - $\begin{pmatrix} \beta_x \\ \beta_y \end{pmatrix} = \begin{pmatrix} K\cos\theta_i - 3x_i \\ K\sin\theta_i - 3y_i \end{pmatrix}$
       - $v(0) = v(1) = K$: this is a free choice, they do not have to be equal.

  iii. Reconstruct the path:
     - $\theta(s) = ATAN2(y'(s), x'(s))$
     - $\tilde{v}(s) = \pm\sqrt{x'^2 + y'^2}$
     - $\tilde{\omega}(s) = \dfrac{(y''x' - y'x'')}{x'^2 + y'^2}$

- **Scheme 2**: With **chained form** (and **parametrized inputs**): **if the robot can be transformed in chained form.**
  - This technique can be easily generalized to any kinematic model.
  - This approach consists of parametrizing the inputs instead of the path.
  - Geometric version of the chained form:
    - $\dot{z}_1' = \tilde{v}_1$
    - $\dot{z}_2' = \tilde{v}_2$
    - $\dot{z}_3' = z_2 \tilde{v}_1$
    - …
    - $\dot{z}_n' = z_{n-1} \tilde{v}_1$
  - First we have to convert to the chained form the initial and desiered configurations, So we need to transform the coordinates from q_i, q_f to z_i, z_f. With the hypothesis:
    - $z_{1,f} \neq z_{1,i}$
    - $\Delta = z_{1,f} - z_{1,i}$

  a. $\tilde{v}_1 = sign(\Delta)$
     - $sign(\Delta) = 1; \Delta > 0$

- $sign(\Delta) = 0; \Delta = 0$
- $sign(\Delta) = -1; \Delta < 0$

b. $\tilde{v}_2 = c_0 + c_1 s + \cdots + c_{n-2} s^{n-2}$
  - $n-1$ free parameters to impose the $n-1$ conditions
  - With $s \in [s_i, s_f]$ = $[0, |\Delta|]$
  - $z_2(|\Delta|) = z_{2,f}$
  - $z_n(|\Delta|) = z_{n,f}$
  - Exploiting the fact that the system is linear, $\tilde{v}_1$ is constant.
  - This results in a linear system of equations:
    - $D(z_i, z_f, \Delta) \begin{pmatrix} c_0 \\ c_1 \\ \cdots \\ c_{n-2} \end{pmatrix} = d(z_i, z_f, \Delta)$
      - D: always non singular if $\Delta \neq 0$
      - c_i: unknowns

c. Knowing $\tilde{v}_1, \tilde{v}_2$, integrate the system to obtain z_i. Then use inverse transformation to get original coordinates q and inputs u

- Remarks:
  - Both schemes are guaranteed to produce **feasible** paths.
    - Scheme 1: uses reconstruction formulas that are based on the model (geometric)
    - Scheme 2: paths are obtained by direct integration the model (geometric)
    - All generated paths will automatically satisfy the constraints (encoded in the model)
  - All systems that are flat can be represented in chained form and viceversa.
    - Scheme 1: requires differential flatness
    - Scheme 2: requires chained form transformability

- **Trajectory Planning**:
  - **Decoupled approach**: we have a path $q(s), s \in [s_i, s_f]$. We want to add a timing law $s = s(t), t \in [t_i, t_f]$, with $s(t_i) = s_i, s(t_f) = s_f$.

    a. We need to choose any $s(s)$ so that:
      - $u_i = \tilde{u}_i \dot{s}$

    b. Check if the velocity bounds are satisfied with the chosen $s(t)$
      - $|u_i(t)| \leq u_{i,max}$

    c. If not, redefine the time:
      - $\tau = \frac{t}{\mathrm{T}}$
      - $\dot{s} = \frac{\mathrm{ds}}{\mathrm{dt}} = \frac{\mathrm{ds}}{\mathrm{d\tau}} \frac{\mathrm{d\tau}}{\mathrm{dt}} = \frac{\mathrm{ds}}{\mathrm{d\tau}} \frac{1}{\mathrm{T}}$
  - **Direct approach**: use scheme 1 or 2 with t in place of s using the kinematic model in place of the geometric model. Applying the velocity bounds on this approach not only changes the timing law while keeping the path, **it will change the whole trajectory**, i.e. changing both the path and the timing law.

Chained Form

A "canonical" form for kinematic models of WMR

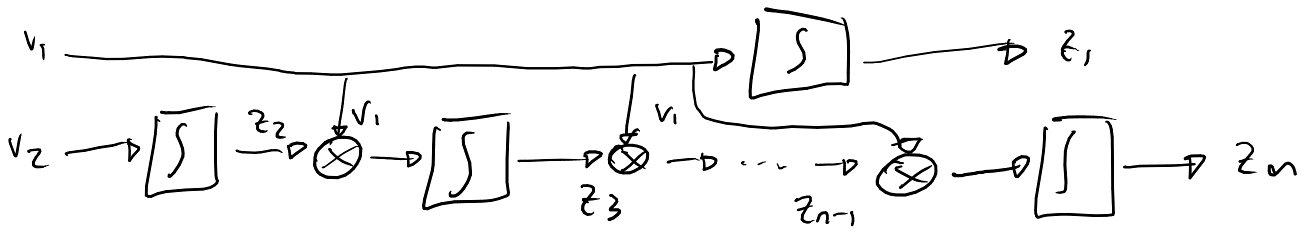
$$\dot{z} = \begin{pmatrix} \dot{z}_1 \\ \vdots \\ \dot{z}_n \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \\ z_2 v_1 \\ \vdots \\ z_{n-1} v_1 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 \\ 0 \\ z_2 \\ \vdots \\ z_{n-1} \end{pmatrix}}_{\gamma_1} v_1 + \underbrace{\begin{pmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}}_{\gamma_2} v_2$$

A particular driftless 2-input dynamical system

$z \in \mathbb{R}^n$ STATE

$v \in \mathbb{R}^2$ inputs

Block scheme



Controllable

$[\gamma_1, \gamma_2], [\gamma_1, [\gamma_1, \gamma_2]], \ldots \to \dim \Delta_A = n$ (verify!)

There exist N&S conditions for transforming

$\dot{q} = g_1(q) u_1 + g_2(q) u_2 \quad (*)$

into $\quad \dot{z} = \gamma_1(z) v_1 + \gamma_2(z) v_2$

all systems like $(*)$ with $n \le 4$ satisfy these conditions

(unicycle bicycle can be put in chained form)

$*$ tricycle may not!

How to transform?

via

(1) coord transformation $z = \alpha(q)$

(2) input transformation $v = \beta(q)\,u$

↳ a feedback transformation (it needs the state of the robot)

unicycle

$$\begin{cases}\dot{x} = v\cos\theta \\ \dot{y} = v\sin\theta \\ \dot{\theta} = \omega\end{cases} \quad (1) \qquad \text{vs.} \qquad \begin{cases}\dot{z}_1 = v_1 \\ \dot{z}_2 = v_2 \\ \dot{z}_3 = z_2 v_1\end{cases}$$

input of the chained form

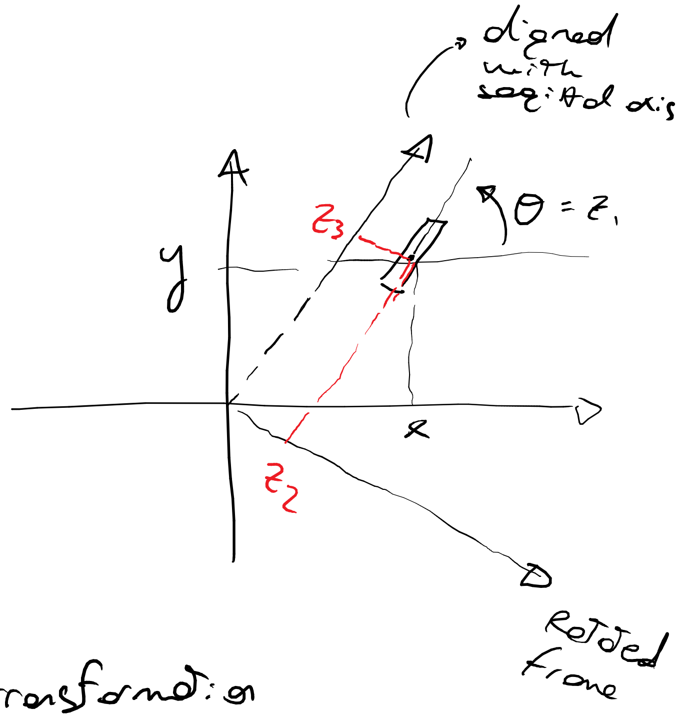
chained form (2)

$$q = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} \qquad u = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} v \\ \omega \end{pmatrix}$$

drift vel.

(1) becomes (2) letting

$$\begin{cases} z_1 = \theta \\ z_2 = x\cos\theta + y\sin\theta \\ z_3 = x\sin\theta - y\cos\theta \end{cases}$$



⇓

(2,3) chained form

↓ inputs

↳ states

$$\boxed{\dot{z}_1 = \dot{\theta} = \omega = v_1}$$

1st input transformation

$$\dot{z}_2 = \dot{x}\cos\theta - x\sin\theta\,\dot{\theta} + \dot{y}\sin\theta - y\cos\theta\,\dot{\theta}$$
$$= v\cos^2\theta + v\sin^2\theta - (x\sin\theta + y\cos\theta)\,\dot{\theta}$$
$$= v - z_2\,\omega = v_2$$

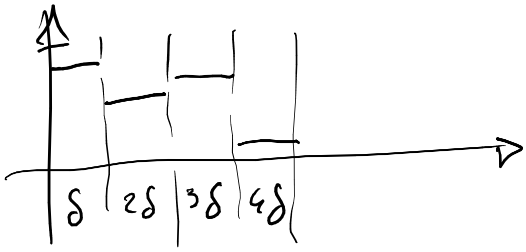
2nd input transformation

$\dot{z}_3$ to check using coord. & input transf. $= z_2 v_1$

Use of closed form

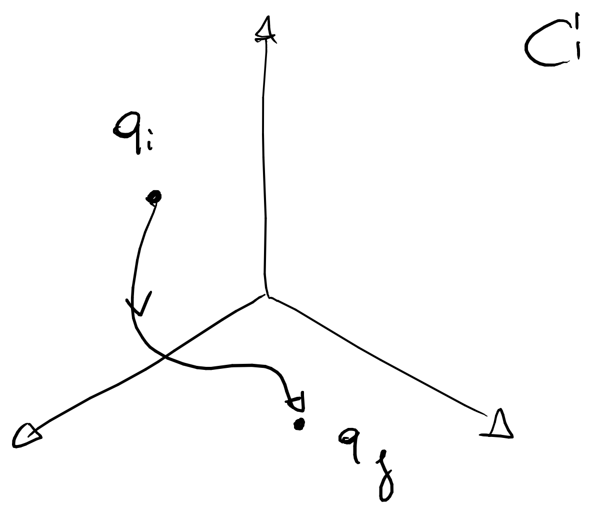
good for planning & control

- It applies to a large number of WMR
- It can be easily integrated under appropriate inputs (e.g. piecewise constant)



Path & Trajectory planning → for WMR

- Path, trajectory
- Timing law
- Differential flatness
- Path planning
- Trajectory planning



→ path parameter (typically the arc length)

path $q(s), \; s \in [s_i, s_f]$

$q(s_i) = q_i \;, \; q(s_f) = q_f$ — I can go back & forth with s on the path

trajectory $q(t), \; t \in [t_i, t_f]$

$q(t_i) = q_i \;, \; q(t_f) = q_f$ → Particular path with time as parameter, I can't go back

Path: Geometry & motion

Trajectory: Geometry + speed motion

$q = q(t), \; t \in [t_i, t_f]$ can be separated in

① $q = q(s) \;, \; s \in [s_i, s_f]$
$q(s_i) = q_i \;, \; q(s_f) = q_f$

② $s = s(t), \; t \in [t_i, t_f]$
$s(t_i) = s_i \;, \; s(t_f) = s_f$

path + timing law

Path Planning

given $q_i, q_f \in C_i$ find a path $q = q(s)$, $s \in [s_i, s_f]$
such that $q(s_i) = q_i$
$q(s_f) = q_f$ $\rightarrow$ A geometric Problem

Trajectory planning

Given $q_i, q_f \in C_i$, find a trajectory $q(t)$, $t \in [t_i, t_f]$
such that $q(t_i) = q_i$
$q(t_f) = q_f$ $\rightarrow$ A kinematic Problem

Usually path & trajectory are defined with the absence of obstacles (treated in motion planning)

---

2 approaches

1. all-in-one : <u>directly</u> plan a trajectory $q(t)$
2. decoupled (two phases) : first plan a path then a timing law (better than 1 because we separate the plan part from the timing law)

Non holonomic constraints

→ local mobility is restricted:

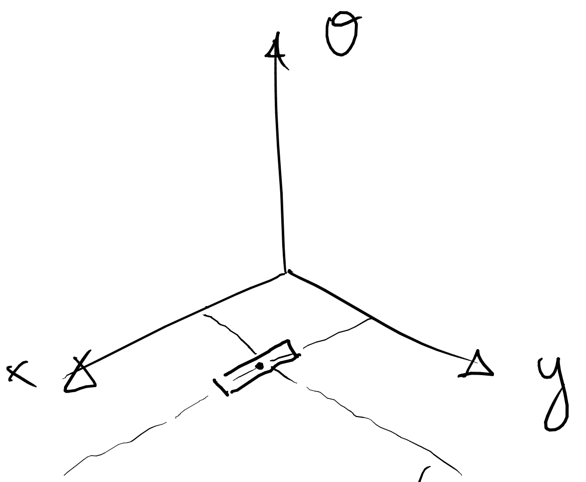
$\dot{q} \in N(A^T(q))$

↳ not all generalized velocities are admissible
therefore
not all trajectories in C are admissible!

• What about paths?

Also paths are restricted cause it's a matter of geometry

<u>example</u>: unicycle



→ not a possible trajectory (ZMCL)

⇓

the path along the ZMCL is actually <u>not admissible</u>

Analytically

$\dot{q}$ what are these generalized velocities?

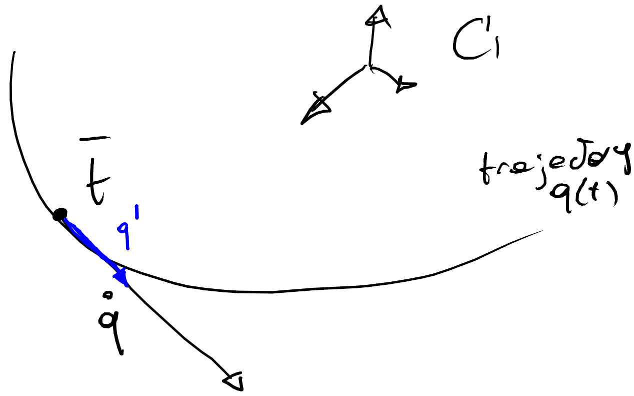
$\dot{q} = \frac{\partial q}{\partial t}$ $\begin{pmatrix} q_1 \\ \vdots \\ q_n \end{pmatrix}$

$q = q(t)$ a trajectory seen as
$q = q(s(t))$
therefore
→ $\frac{\partial q}{\partial t} = \underbrace{\frac{\partial q}{\partial s}}_{q'} \underbrace{\frac{\partial s}{\partial t}}_{\dot{s}} = q' \cdot \dot{s}$

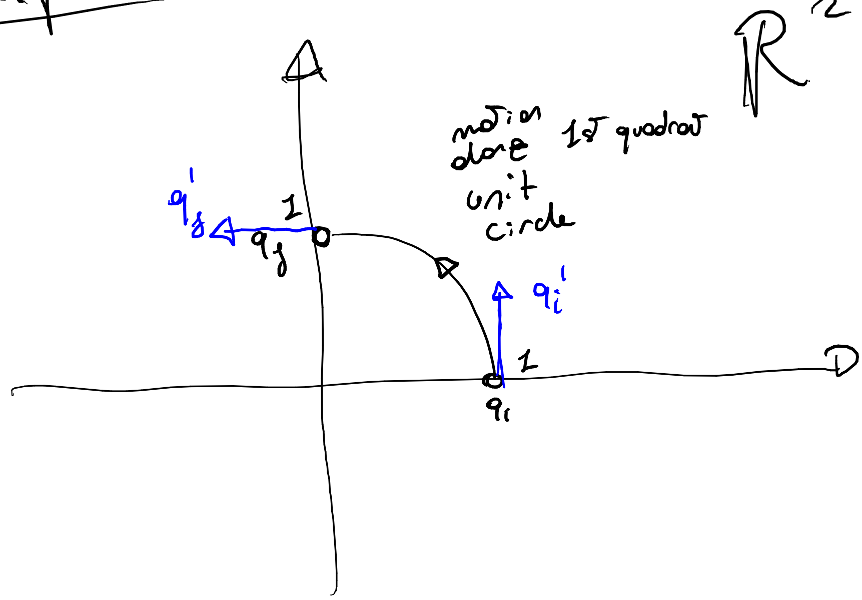
$q' \cdot \dot{s} \rightsquigarrow$ a scalar

$$\downarrow$$

$$\begin{pmatrix} \frac{\partial q_1}{\partial s} \\ \vdots \\ \frac{\partial q_n}{\partial s} \end{pmatrix}$$ the tangent vector is the path at s



<u>example</u>



$q(s) = \begin{pmatrix} \cos s \\ \sin s \end{pmatrix}$ $\quad s \in [\underbrace{0}_{s_i}, \underbrace{\pi/2}_{s_g}]$ $\quad$ s: arc length

$q' = \begin{pmatrix} -\sin s \\ \cos s \end{pmatrix}$ the tangent vector

$\|q'\| = 1$ because s is the arc length!

<u>exercise</u> : find a different parametrization $\begin{pmatrix} \text{e.g.} \\ x = s \end{pmatrix}$
compute q'
verify it's a tangent vector, $\|q'\| \neq 1$
↳ because we are not using arc length

$\dot{q} = q' \cdot \dot{s}$ then

$A^T(q)\dot{q} = 0$ Pfaffian constraints can be written as

$A^T(q) q' \underbrace{\dot{s}}$ $= 0$

↳ must be true for any $\dot{s}$

$\Downarrow$

$$\boxed{A^T(q) q' = 0}$$

kinematic constraints actually are constraints on tangent vectors to the path

let's exploit this part. →

$\{\varepsilon_1(q), \dots, \varepsilon_n(q)\}$ a basis of the null space $N(A^T(q))$
same as before

(*)
$$\boxed{q' = \sum_{j=1}^{n} \varepsilon_j(q)\, \tilde{u}_j}$$

same vfs as before

$\tilde{u}_j$ different coefficients → different inputs

time derivatives replaced by derivatives wrt s

→ Geometric version of the kinematic model

Relationship $u_j / \tilde{u}_j$

u_j → velocity inputs

$\tilde{u}_j$ → geometric inputs

multiply (*) by $\dot{s}$

$$q' \cdot \dot{s} = \sum_{j=1}^{n} \varepsilon_j(q)\, \tilde{u}_j \cdot \dot{s}$$

$$= \dot{q} = \sum_{j=1}^{n} \varepsilon_j(q)\, u_j$$

⟹ $$\boxed{\underbrace{u_j}_{\text{velocity inputs}} = \underbrace{\tilde{u}_j}_{\text{geometric inputs}} \cdot \dot{s}}$$

→ Kinematic model: Good for trajectory planning

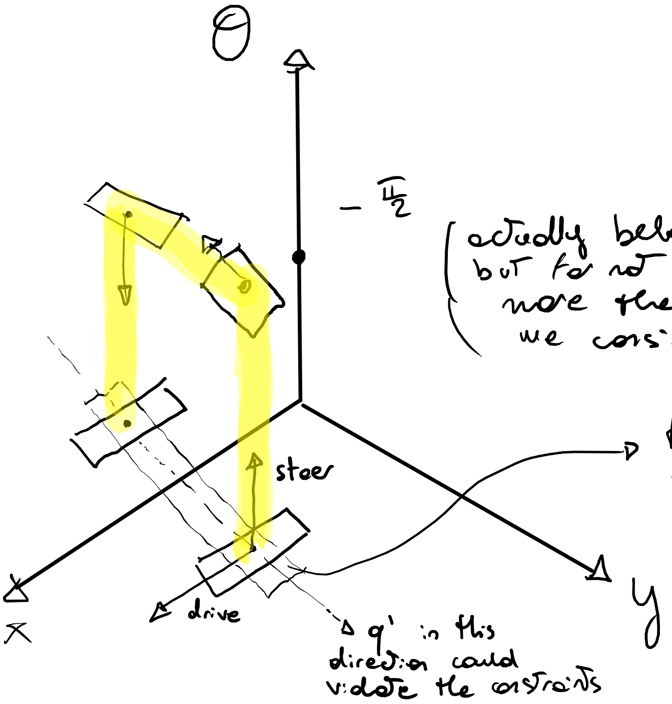
Interpretation of (*)

Choose any $\tilde{u}_j$ $(j = 1 .. n)$, the model will predict the resulting path

→ good for path planning

Unicycle

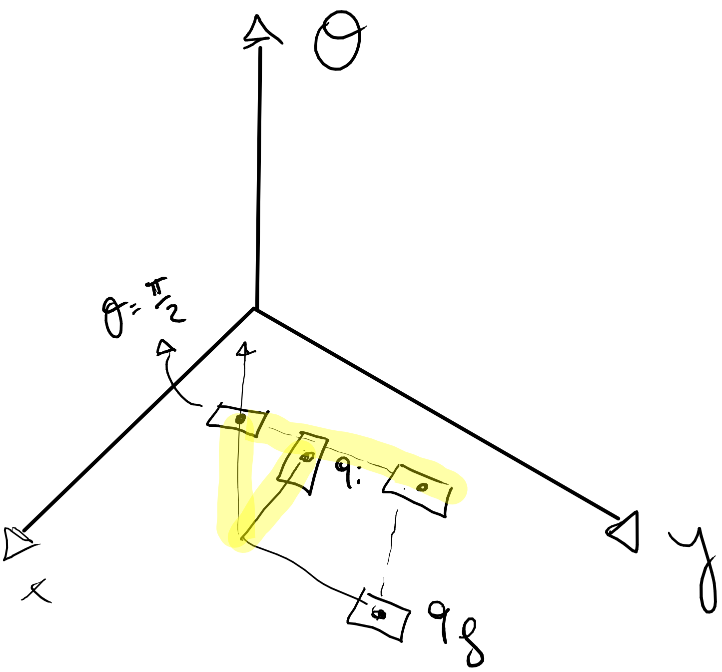
$$q' = \begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix} = \underbrace{\begin{pmatrix} \cos\theta \\ \sin\theta \\ 0 \end{pmatrix} \tilde{v}}_{\text{drive}} + \underbrace{\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tilde{\omega}}_{\text{steer}}$$



(actually below the axis but for not complicating more the graph we consider the opposite)

The cartesian footprint of the motion is a segment along ZRL

↳ This, however, is NOT the 'C-space' motion!



Differential flatness

A system is differentially flat if states and inputs can be RECONSTRUCTED from the evolution of some outputs (FLAT outputs)

definition

$$\dot{q} = \sum_{j=1}^{n} g_j(q) u_j = G(q)u \quad \left(\text{or } q' = \sum_{j=1}^{n} g_j(q)\tilde{u}_j = G(q)\tilde{u}\right)$$

is differentially flat if there exist a set of outputs

$W = h(q)$ such that $\qquad W$ = flat outputs

$$\begin{cases} q = \eta(w, \dot{w}, \dots, w^{(r)}) \\ u = \mu(w, \dot{w}, \dots, w^{(r)}) \end{cases} \Rightarrow$$

these two allow an ALGEBRAIC RECONSTRUCTION OF STATE / INPUTS

we don't need an integral then we don't need initial conditions.

$$\eta(w, w', \dots, w^{(r)'})$$
$$\mu(w, w', \dots, w^{(r)'})$$

$\rightarrow$ if we are considering a geometric model

Unicycle

$$\begin{cases} \dot{x} = v\cos\theta \\ \dot{y} = v\sin\theta \\ \dot{\theta} = \omega \end{cases}$$

is <u>differentially flat</u>

flat outputs $W = \begin{pmatrix} x \\ y \end{pmatrix}$

a fact: • state reconstruction $(w \rightarrow q)$

$$\begin{cases} x = w_1 = x \\ y = w_2 = y \\ \theta = \arctan \dfrac{\dot{y}}{\dot{x}} \end{cases}$$

better

$= \mathrm{ATAN2}(\dot{y}, \dot{x}) + k\pi$

$k = 0$ if $v > 0$

$k = 1$ if $v < 0$

• input reconstruction $(w \rightarrow v, \omega)$

$$\begin{cases} v = \pm\sqrt{\dot{x}^2 + \dot{y}^2} \\ \omega = \dfrac{1}{1 + \left(\dfrac{\dot{y}}{\dot{x}}\right)^2} \dfrac{\ddot{y}\dot{x} - \ddot{x}\dot{y}}{\dot{x}^2} = \dfrac{\ddot{y}\dot{x} - \ddot{x}\dot{y}}{\dot{x}^2 + \dot{y}^2} \end{cases}$$

+ (forward motion)

− (backward motion)

$(r = 2)$

For the <u>cartesian trajectory</u>

$x(t), y(t)$

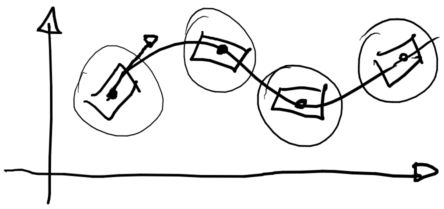
$\longrightarrow$ state trajectory $q(t)$

input history $u(t)$

The possibility of reconstruction is lost when $\dot{x} = \dot{y} = 0$

(motion degenerates to a point)

<u>interpretation</u>



top view

$\odot$ snapshot

need a move for reconstruction

$\longrightarrow$ cannot tell θ (state reconstruction is lost)

cannot tell ω (input reconstruction is lost)

Path & Trajectory Planning

differential flatness

$w = h(q)$ flat outputs
↓
function of the state

$$\begin{cases} q = q(w, \dot{w}, \dots, w^{(r)}) \\ u = u(w, \dot{w}, \dots, w^{(r)}) \end{cases}$$ ALGEBRAIC reconstruction

Unicycle : flat outputs are x, y
Bicycle : flat outputs are x, y
Tricycle with Trailer : //

Chained form also differentially flat

$\dot{z}_1 = v_1$
$\dot{z}_2 = v_2$
$\dot{z}_3 = z_2 v_1$
(2,3) case

flat outputs $\underline{z_1}$, $\underline{z_3}$

$$z_2 = \frac{\dot{z}_3}{v_1} = \frac{\dot{z}_3}{\dot{z}_1}$$ → state reconstruction: algebraic function of the 1-st order derivative

$$v_1 = \dot{z}_1$$
$$v_2 = \dot{z}_2 = \frac{\ddot{z}_3 \dot{z}_1 - \dot{z}_3 \ddot{z}_1}{\dot{z}_1^2}$$

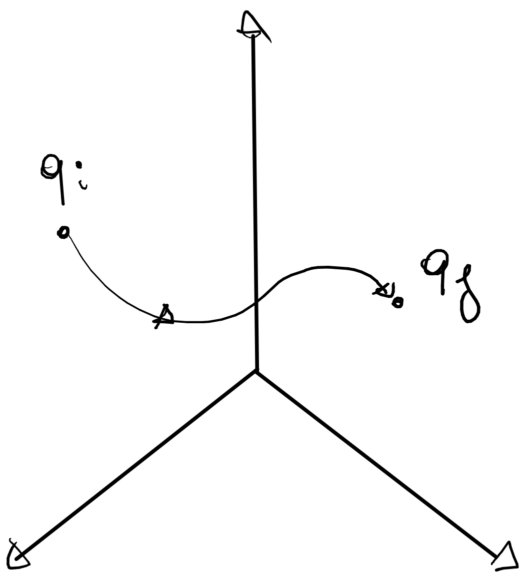
(2, n) chained form → z_1, z_n are FLAT outputs

Use of flatness for planning

flat outputs can evolve arbitrarily

$\Rightarrow$ Then the reconstruction formulas will provide the associated state trajectory, which will be automatically feasible

Generic algorithm for path planning / trajectory plan. using flatness



hyp robot is flat

$w = h(q)$

flat outputs

1. Compute $w_i = h(q_i)$, $w_f = h(q_f)$
2. Generate a path (in s) or a trajectory (in t) from w_i to w_f with the appropriate boundary conditions (interpolation problem)
3. Use the reconstruction formulas to compute the state and input

$\downarrow$

path / trajectory

example : unicycle

$$q_i = \begin{pmatrix} \boxed{\begin{matrix} x_i \\ y_i \end{matrix}} \\ \theta_i \end{pmatrix} \longrightarrow q_f = \begin{pmatrix} x_f \\ y_f \\ \theta_f \end{pmatrix}$$

$$w_i = \begin{pmatrix} x_i \\ y_i \end{pmatrix} \qquad w_f = \begin{pmatrix} x_f \\ y_f \end{pmatrix}$$

$$\begin{cases} x' = \tilde{v} \cos \theta \\ y' = \tilde{v} \sin \theta \\ \theta' = \tilde{\omega} \end{cases} \Rightarrow \text{Geometric model} \rightarrow \text{path planning (not trajectory)}$$

boundary conditions

$$x'(s_i) = \tilde{v}(s_i) \cos\theta_i \rightarrow k_i$$
$$y'(s_i) = \tilde{v}(s_i) \sin\theta_i$$
$$x'(s_f) = \tilde{v}(s_f) \cos\theta_f$$
$$y'(s_f) = \tilde{v}(s_f) \sin\theta_f \rightarrow k_f$$

x must go from x_i to x_f satisfying $x'(s_i) = \ldots$ (#)
and the same for $y \ldots$ $x'(s_f) = \ldots$

$\Rightarrow$ a cubic polynomial

for $s \in [0, 1] = [s_i, s_f]$

$$x(s) = s^3 x_f - (s-1)^3 x_i + \alpha_x s^2(s-1) + \beta_x s(s-1)$$

$$x(0) = x_i$$
$$x(1) = x_f$$

α_x, β_x so as to satisfy (#)

$$\alpha_x = k \cos \theta_f - 3x_f$$
$$\beta_x = k \cos \theta_i + 3x_i \qquad (k = k_i = k_f)$$

And the same for y

$$\begin{matrix} x(s), & x'(s), & x''(s) \\ y(s), & y'(s), & y''(s) \end{matrix} \xRightarrow{\text{reconstr.}} \theta(s), v(s), \omega(s)$$

example : chained form

(2,3)

$$\begin{cases} z_1' = \tilde{v}_1 \\ z_2' = \tilde{v}_2 \\ z_3' = z_2 \tilde{v}_1 \end{cases} \quad \left(\text{geometric model}\right)$$

$$z_i = \begin{pmatrix} z_{1i} \\ z_{2i} \\ z_{3i} \end{pmatrix} \rightarrow z_f = \begin{pmatrix} z_{1f} \\ z_{2f} \\ z_{3f} \end{pmatrix}$$

$$w_i = \begin{pmatrix} z_{1i} \\ z_{3i} \end{pmatrix} \rightarrow w_f = \begin{pmatrix} z_{1f} \\ z_{3f} \end{pmatrix}$$

flat outputs : z_1, z_3

(see book for interpolation)

boundary conditions

$$z_2 = \frac{z_3'}{z_1'} \rightarrow \frac{z_3'(s_f)}{z_1'(s_f)} = z_{2f} \text{ and } \frac{z_3'(s_i)}{z_1'(s_i)} = z_{2i}$$

An alternative approach (not based on differential flatness) works for robots that can be transformed into chained form

idea: easily

path planning: easily integrable with the appropriate inputs

$z_1' = \tilde{v}_1$

$z_2' = \tilde{v}_2$

$z_3' = z_2 \tilde{v}_1$

$\vdots$

$z_n' = z_{n-1} \tilde{v}_1$

define $\Delta = z_{1,g} - z_{1,i}$, assume $\Delta \neq 0$

choose $\tilde{v}_1 = \mathrm{sgn}(\Delta) = (\pm 1)$

$\downarrow$

$\tilde{v}_2 = \underline{c_0} + \underline{\underline{c_1}} s + \dots + \underline{\underline{c_{n-2}}} s^{n-2}$ $\quad$ $n-1$ params (to "place" $n-1$ variables $z_2, \dots, z_{n-1}$)

$s \in [\underset{s_i}{0}, \underset{s_g}{|\Delta|}]$

Integrate (can be done because $\tilde{v}_1 = \pm 1$) the chained form equations and impose that

$$\begin{cases} z_2(|\Delta|) \overset{!}{=} z_{2,g} \\ \vdots \\ n-1 \text{ eqs in } n-1 \text{ unknowns} \\ \vdots \\ z_n(|\Delta|) \overset{!}{=} z_{n,g} \end{cases} \longrightarrow \text{a linear system}$$

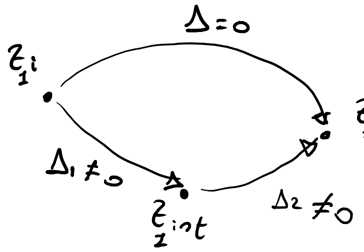
$$\underset{\text{non sing iff } \Delta \neq 0}{D(\Delta)} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-2} \end{pmatrix} = b(z_i, z_g, \Delta)$$

e.g. in the (2,3) case

$$D(\Delta) = \begin{pmatrix} |\Delta| & \Delta^2/2 \\ \mathrm{sgn}(\Delta)\frac{\Delta^2}{2} & \frac{\Delta^3}{6} \end{pmatrix} \qquad b(\dots) = \begin{pmatrix} z_{2g} - z_{2i} \\ z_{3g} - z_{3i} - z_{2i}\Delta \end{pmatrix}$$

if $\underline{\Delta = 0}$ $\to$ $\Delta = z_{1g} - z_{1i}$

1. add a "via point"



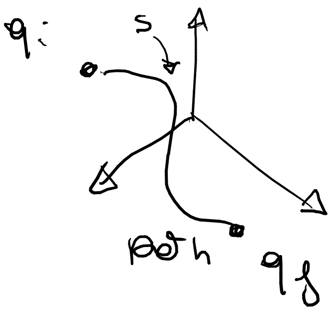
2. if z_1 is an angle: (as for the unicycle)
let $z_{1,g}^{new} = z_{1,g}^{old} + 2\pi$

Choice of timing law

We have a path $q(s) \in [s_i, s_g]$

$q(s_i) = q_i$

$q(s_g) = q_g$



how do we choose a timing law $s = s(t)$?

A trajectory is needed to actually control the robot in time

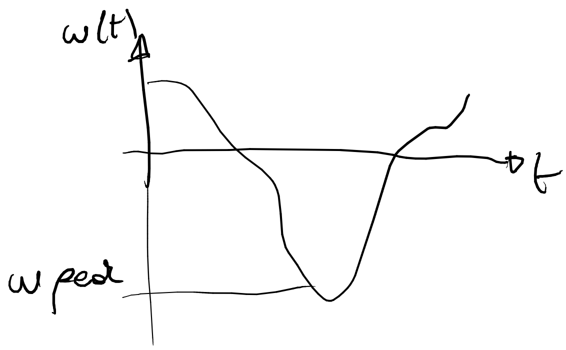
Velocity bounds

unicycle $|v(t)| \le v_{max}$
$|\omega(t)| \le \omega_{max}$

consider these bounds when choosing a timing law

let $t = \alpha s$ $\left(s = \frac{t}{\alpha}\right)$

1. $\alpha = 1$, compute $v(t)$, $\omega(t)$



2. if $|v_{peak}| \le v_{max}$
$|\omega_{peak}| \le \omega_{max}$

then $\alpha = 1$ is ok $\Rightarrow s = t$

3. else, let $\eta = \max\left(\frac{|v_{peak}|}{v_{max}}, \frac{|\omega_{peak}|}{\omega_{max}}\right)$ and let $\alpha = \frac{1}{\eta}$

$\Rightarrow$ now $|v| \le v_{max}$ & $|\omega| \le \omega_{max}$ with (at least) one velocity saturating the bound at peak value

Autonomous and Mobile Robotics
Prof. Giuseppe Oriolo

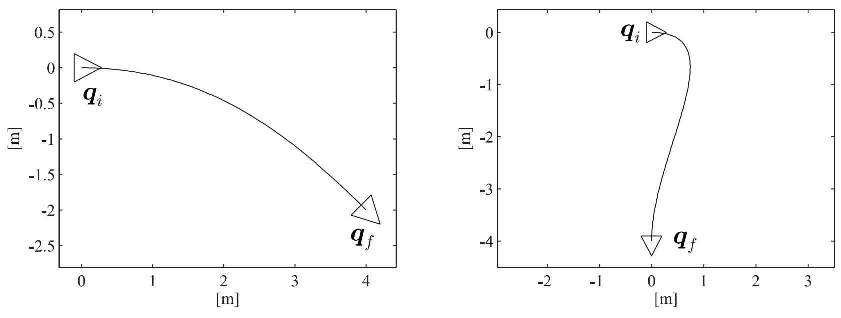
Wheeled Mobile Robots 3
Path/Trajectory Planning
companion slides for the blackboard lecture

Dipartimento di Ingegneria Informatica
Automatica e Gestionale Antonio Ruberti

Sapienza
Università di Roma

parking a unicycle: numerical results

1. forward parking
cubic polynomials for cartesian coords x,y (flat outputs)

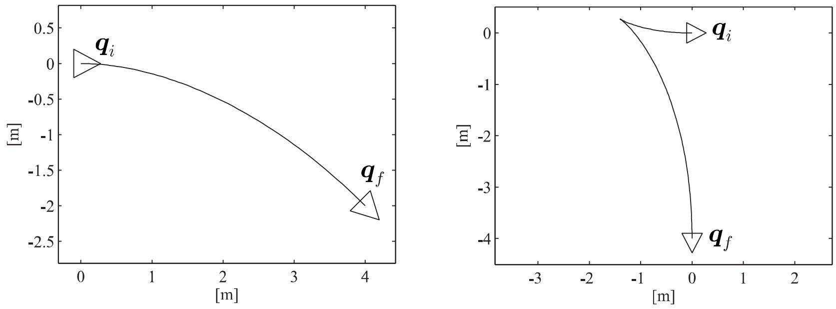


- $k=5>0$, hence forward motion
- no motion inversions

parking a unicycle: numerical results

1. forward parking

parameterized inputs on chained form

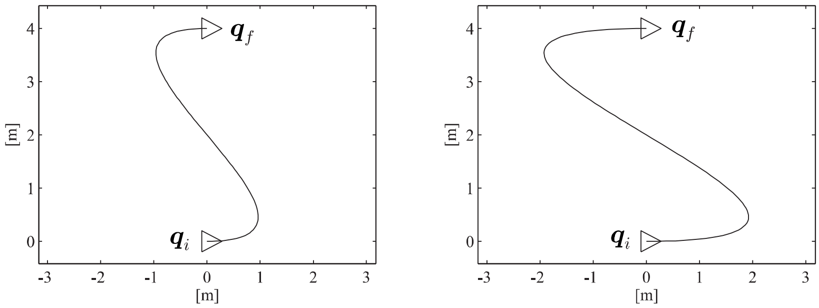


- first maneuver is similar
- a motion inversion (cusp) in the second

parking a unicycle: numerical results

2. parallel parking

cubic polynomials for cartesian coords x,y (flat outputs)

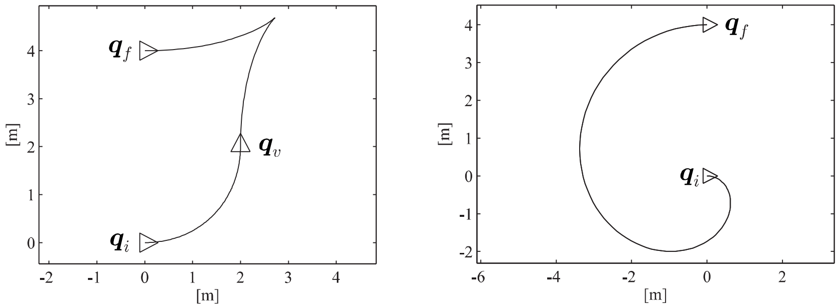


- left: k=10, right: k=20
- no motion inversions

parking a unicycle: numerical results

2. parallel parking

parameterized inputs on chained form

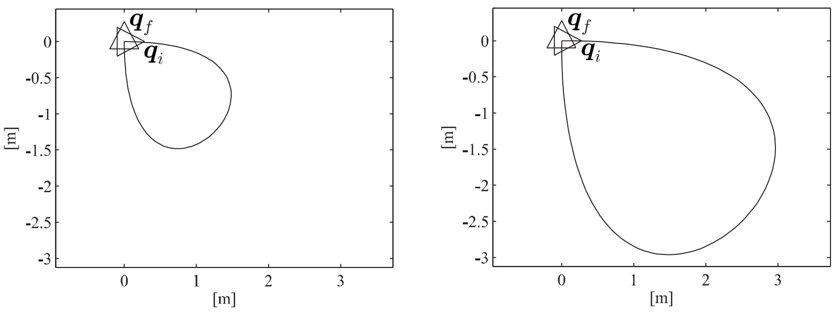


- left: with a via point
- right: requiring a full rotation

parking a unicycle: numerical results

3. pure reorientation

cubic polynomials for cartesian coords x,y (flat outputs)

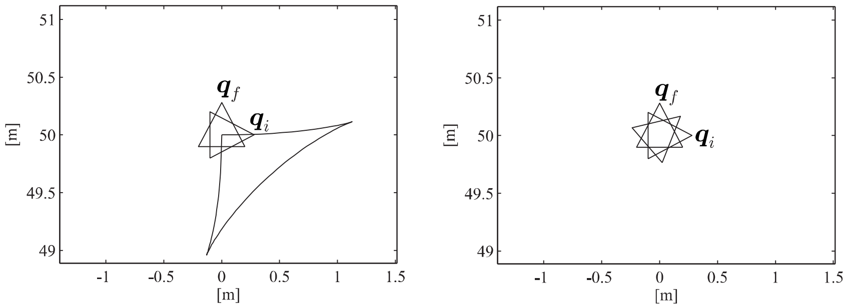


- left: k=10, right: k=20
- need to move the cartesian coordinates!

parking a unicycle: numerical results

3. pure reorientation

parameterized inputs on chained form



- left: straightforward
- right: placing the origin of z_2, z_3 at $\boldsymbol{q}_i$