Robotics: Path planning

(option ISD-master SMART)

W. Perruquetti

Ecole Centrale de Lille,
Cité Scientifique, BP 48,
F-59651 Villeneuve d'Ascq Cedex - FRANCE.
tel : +33 3 20 33 54 50 fax : +33 3 20 33 54 18
e-mail : wilfrid.perruquetti@ec-lille.fr

Septembre 2010 / Chapter 4

Objectives

Pre-requisits:

- Basic physics and maths,
- previous chapters (Robotics Chapter 1 to 3): modeling,

Objectives : Path planning and tracking

- What is Path planning,
- Path planning, controllability and flatness,
- Path planning as a polynomial interpolation problem with constraints: application to manipulators, mobile robots and networked mobile robots,
- Tracking (stabilization around a trajectory).

Objectives

Pre-requisits:

- Basic physics and maths,
- previous chapters (Robotics Chapter 1 to 3): modeling,

Objectives : Path planning and tracking

- What is Path planning,
- Path planning, controllability and flatness,
- Path planning as a polynomial interpolation problem with constraints: application to manipulators, mobile robots and networked mobile robots,
- Tracking (stabilization around a trajectory).

Objectives

Pre-requisits:

- Basic physics and maths,
- previous chapters (Robotics Chapter 1 to 3): modeling,

Objectives : Path planning and tracking

- What is Path planning,
- Path planning, controllability and flatness,
- Path planning as a polynomial interpolation problem with constraints: application to manipulators, mobile robots and networked mobile robots,
- Tracking (stabilization around a trajectory).

Objectives

Pre-requisits:

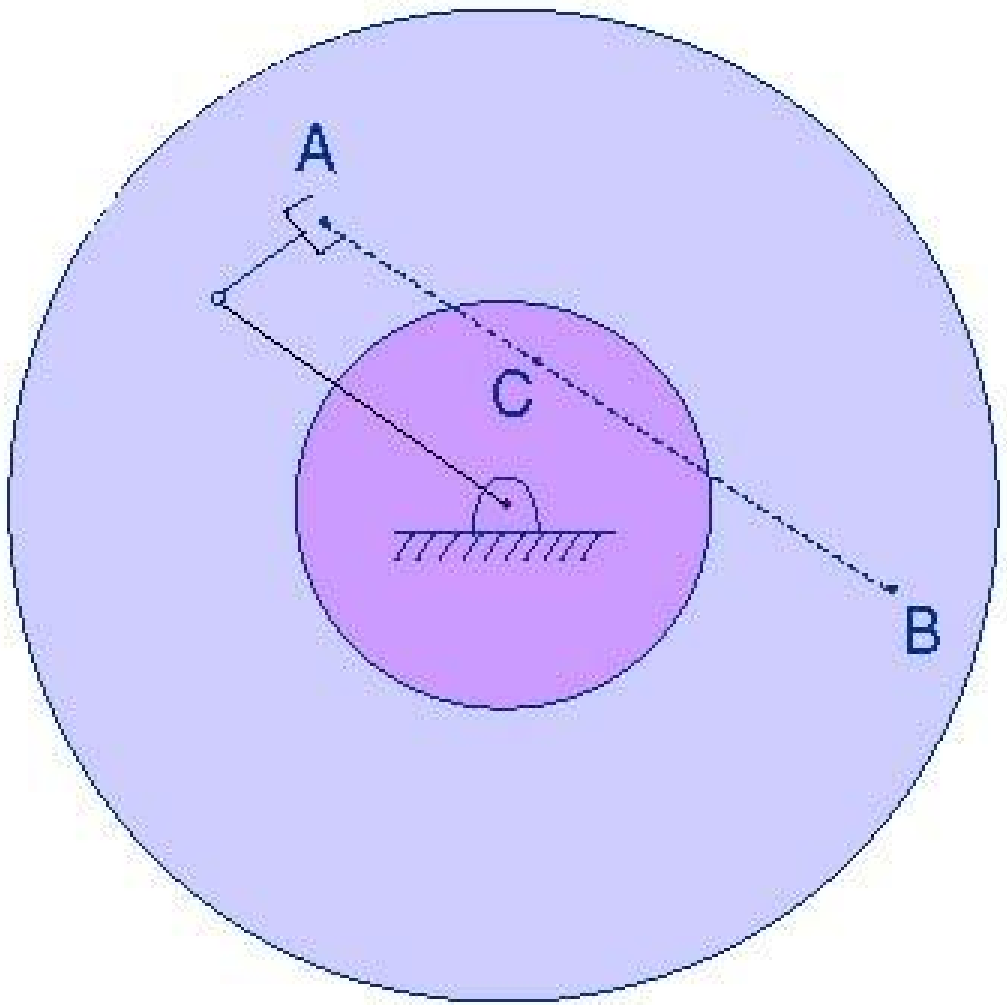
- Basic physics and maths,
- previous chapters (Robotics Chapter 1 to 3): modeling,

Objectives : Path planning and tracking

- What is Path planning,
- Path planning, controllability and flatness,
- Path planning as a polynomial interpolation problem with constraints: application to manipulators, mobile robots and networked mobile robots,
- Tracking (stabilization around a trajectory).

Table of Contents

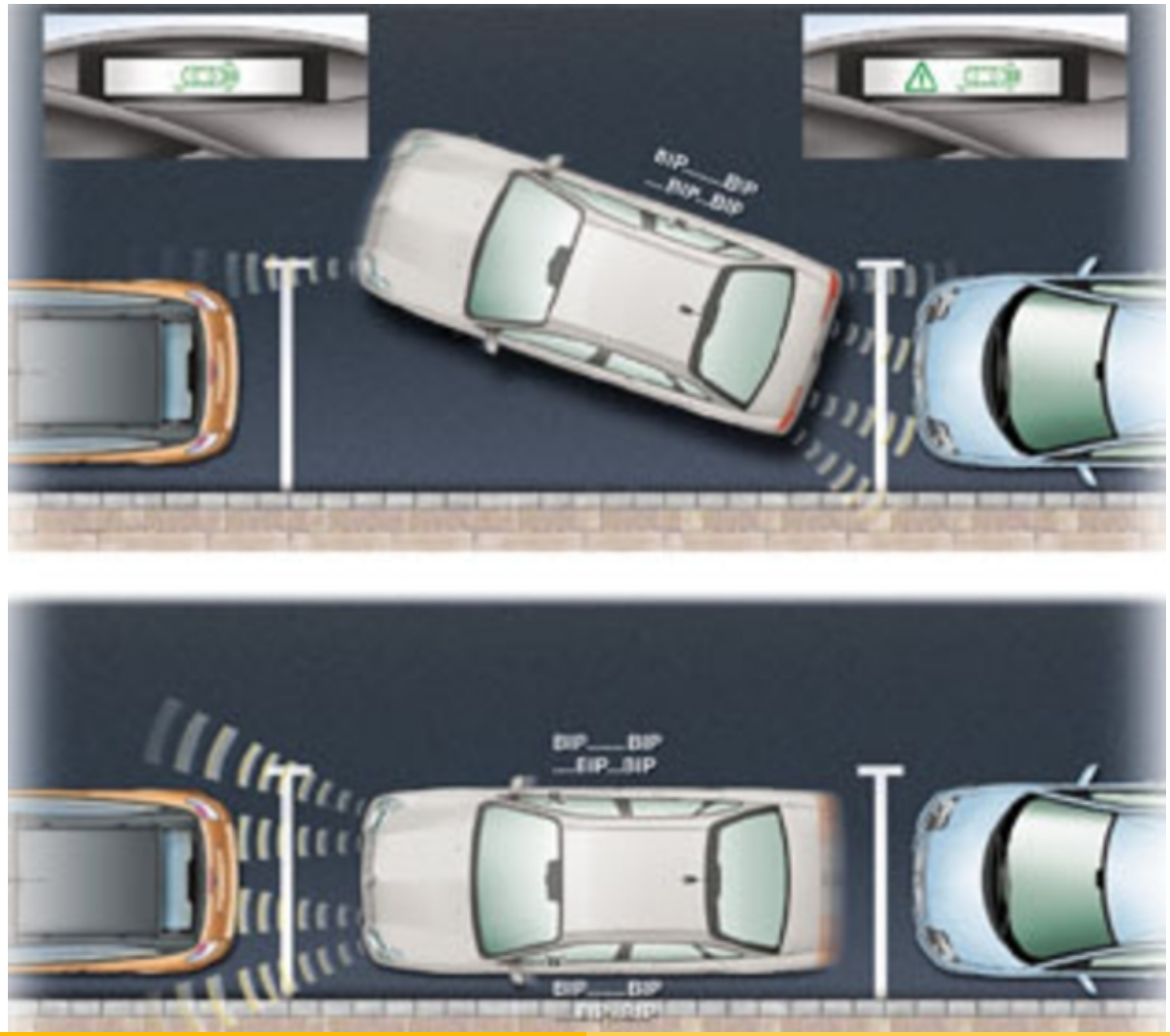
Manipulator from one configuration to another



Car parking

Car parking

Car parking

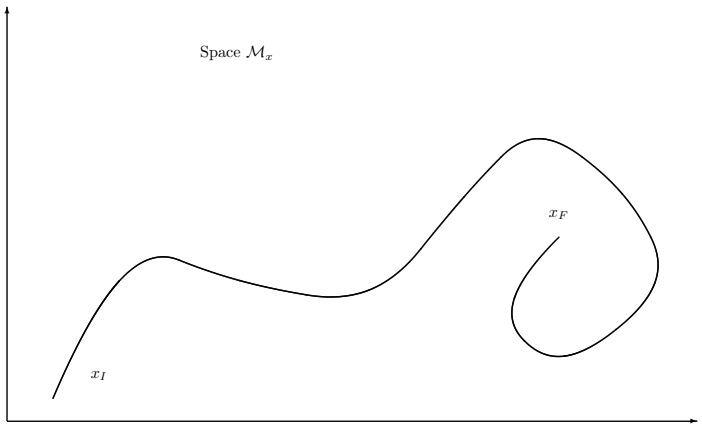


Video

Posture:

$$x = f(q)$$

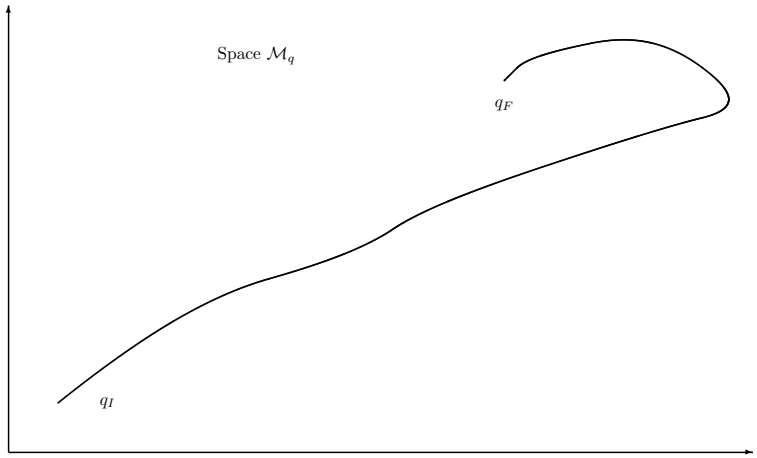
Direct geometric model (manipulator) or Posture model (mobile robot)



Posture

Configuration (state):

$$q = (q_1, \ldots, q_n)$$



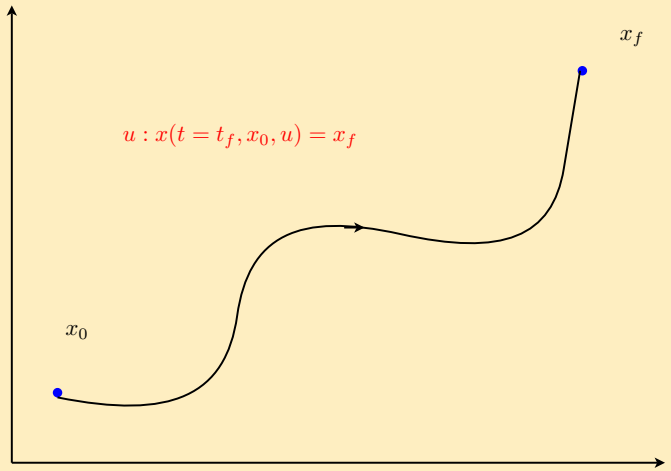
State/Configuration

path planning for state in automatic control it is a reachability problem (controllability)

Controllability

What is ctrb ?

Does there exists a time function (control) $u(t)$ s.t. starting from x_0 the trajectory will reach x_f in a pre-given time t_f?



Controllability

$u(t)$ is the **open loop control** (nominal control)

Path planning: everything is fine?? More or less ...

Experimental results on the open loop system

Path planning: everything is fine?? More or less ...

Video

Path planning: Oups

A unicycle type model . . . with pies

What happens when skidding or sliding??(which may appears if the contact between the road and the tire is not maintained . . .) or just when we have uncertainties in the actuator dynamics or in the dynamical model

For example when skidding or sliding:

$$\left\{ \begin{array}{rcl} \dot{x}_i & = & v_i \cos\theta_i + \pi_{1,i} \\ \dot{y}_i & = & v_i \sin\theta_i + \pi_{2,i} \\ \dot{\theta}_i & = & w_i + \pi_{3,i} \end{array} \right.$$

$\pi_{1,i}, \pi_{2,i}, \pi_{3,i}$ are unknown additive uncertainties.

Question

Path planning needs Tracking !!

$$\dot{x} = f(x, u), y_{\text{flat}}$$

Path planning $\Rightarrow$ nominal trajectory for $y_{\text{flat}}^{\text{nominal}}(t)$ leads to nominal trajectory for $x^{\text{nominal}}(t)$ and open loop control $u^{\text{nominal}}(t)$.

Error Dynamics: $(e = x - x^{\text{nominal}}(t))$

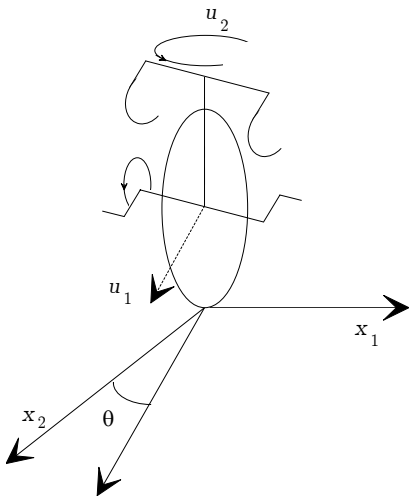
$$\dot{e} = f(e + x^{\text{nominal}}(t), u)$$

Set $u = u^{\text{nominal}}(t) + v$ and find good v s.t.

$$\lim_{t \to \infty} e(t) = 0$$

Table of Contents

Lie bracket



$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \sin\theta \\ \cos\theta \\ 0 \end{pmatrix} u_1 + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} u_2.$$

The linearized model around any point is $\dot{x} = Bu$ thus the Kalman rank criterium $\mathrm{rank}C(A,B) = n$ is not satisfied

$$\mathrm{rank}C(A,B) = \mathrm{rank}B = 2 \neq 3 = n$$

where $C(A,B) = [B, AB, \ldots, A^{n-1}B] = B$!!. Thus the linearized model is NOT controllable but everyday life shows us that we are able to park car.

Lie bracket

Let us apply a piecewise constant input

$$u(t) = \begin{cases} (1,0), t \in [0, \varepsilon[\\ (0,1), t \in [\varepsilon, 2\varepsilon[\\ (-1,0), t \in [2\varepsilon, 3\varepsilon[\\ (0,-1), t \in [3\varepsilon, 4\varepsilon[\end{cases}$$

Using a Taylor serie expansion and using the following notations $\theta = x_3, g_1(x) = (\sin(x_3), \cos(x_3), 0)^T, g_2 = (0,0,1)^T$: every things during such short time acts as $\dot{x} = [g_1, g_2](x)$ where

$$[g_1, g_2] = \left(\frac{\partial g_2}{\partial x} g_1 - \frac{\partial g_1}{\partial x} g_2 \right),$$

Lie bracket

$$
\begin{aligned}
x(\varepsilon) &= x_0 + \varepsilon g_1(x_0) + \frac{1}{2}\varepsilon^2 \left(\frac{\partial g_1}{\partial x}\right)_{x=x_0} g_1(x_0) + o(\varepsilon^2), \\
x(2\varepsilon) &= x(\varepsilon) + \varepsilon g_2(x(\varepsilon)) + \frac{1}{2}\varepsilon^2 \left(\frac{\partial g_2}{\partial x}\right)_{x=x(\varepsilon)} g_2(x(\varepsilon)) + o(\varepsilon^2) \\
&= x_0 + \varepsilon\left(g_1(x_0) + g_2(x_0)\right) + \varepsilon^2 \left(\frac{\partial g_2}{\partial x}\right)_{x=x_0} g_1(x_0) + \frac{1}{2}\varepsilon^2 \left(\frac{\partial g_1}{\partial x}\right)_{x=x_0} g_1(x_0) \\
&\quad + \frac{1}{2}\varepsilon^2 \left(\frac{\partial g_2}{\partial x}\right)_{x=x_0} g_2(x_0) + o(\varepsilon^2)
\end{aligned}
$$

Lie bracket

$$
\begin{aligned}
x(3\varepsilon) &= x(2\varepsilon) - \varepsilon g_1(x(2\varepsilon)) + \frac{1}{2}\varepsilon^2 \left(\frac{\partial g_1}{\partial x}\right)_{x=x(2\varepsilon)} g_1(x(2\varepsilon)) + o(\varepsilon^2) \\
&= x_0 + \varepsilon\left(g_1(x_0) + g_2(x_0)\right) + \varepsilon^2 \left(\frac{\partial g_2}{\partial x}\right)_{x=x_0} g_1(x_0) + \varepsilon^2 \left(\frac{\partial g_1}{\partial x}\right)_{x=x_0} g_1(x_0) \\
&\quad + \frac{1}{2}\varepsilon^2 \left(\frac{\partial g_2}{\partial x}\right)_{x=x_0} g_2(x_0) - \varepsilon\left(g_1(x_0) + \varepsilon \left(\frac{\partial g_1}{\partial x}\right)_{x=x_0} \left(g_1(x_0) + g_2(x_0)\right)\right) + o(\varepsilon^2) \\
&= x_0 + \varepsilon g_2(x_0) + \varepsilon^2 \left(\frac{1}{2}\left(\frac{\partial g_2}{\partial x}\right)_{x=x_0} g_2(x_0) + \left(\frac{\partial g_2}{\partial x}\right)_{x=x_0} g_1(x_0) - \left(\frac{\partial g_1}{\partial x}\right)_{x=x_0} g_2(x_0)\right) \\
&= x_0 + \varepsilon g_2(x_0) + \varepsilon^2 \left(\frac{1}{2}\left(\frac{\partial g_2}{\partial x}\right)_{x=x_0} g_2(x_0) + [g_1, g_2](x_0)\right) + o(\varepsilon^2) \\
x(4\varepsilon) &= x(3\varepsilon) - \varepsilon g_2(x(3\varepsilon)) + \frac{1}{2}\varepsilon^2 \left(\frac{\partial g_2}{\partial x}\right)_{x=x(3\varepsilon)} g_2(x(3\varepsilon)) + o(\varepsilon^2) \\
&= x_0 + \varepsilon g_2(x_0) + \varepsilon^2 \left(\left(\frac{\partial g_2}{\partial x}\right)_{x=x_0} g_2(x_0) + [g_1, g_2](x_0)\right) - \varepsilon\left(g_2(x_0) + \varepsilon \left(\frac{\partial g_2}{\partial x}\right)_{x=x_0} g_2(x\right. \\
&= \left(Id + \varepsilon^2 [g_1, g_2]\right)(x_0) + o(\varepsilon^2)
\end{aligned}
$$

Lie bracket

Le **crochet de Lie** (ou commutateur) défini par :

$$[g_1, g_2] = \left(\frac{\partial g_2}{\partial x} g_1 - \frac{\partial g_1}{\partial x} g_2 \right),$$

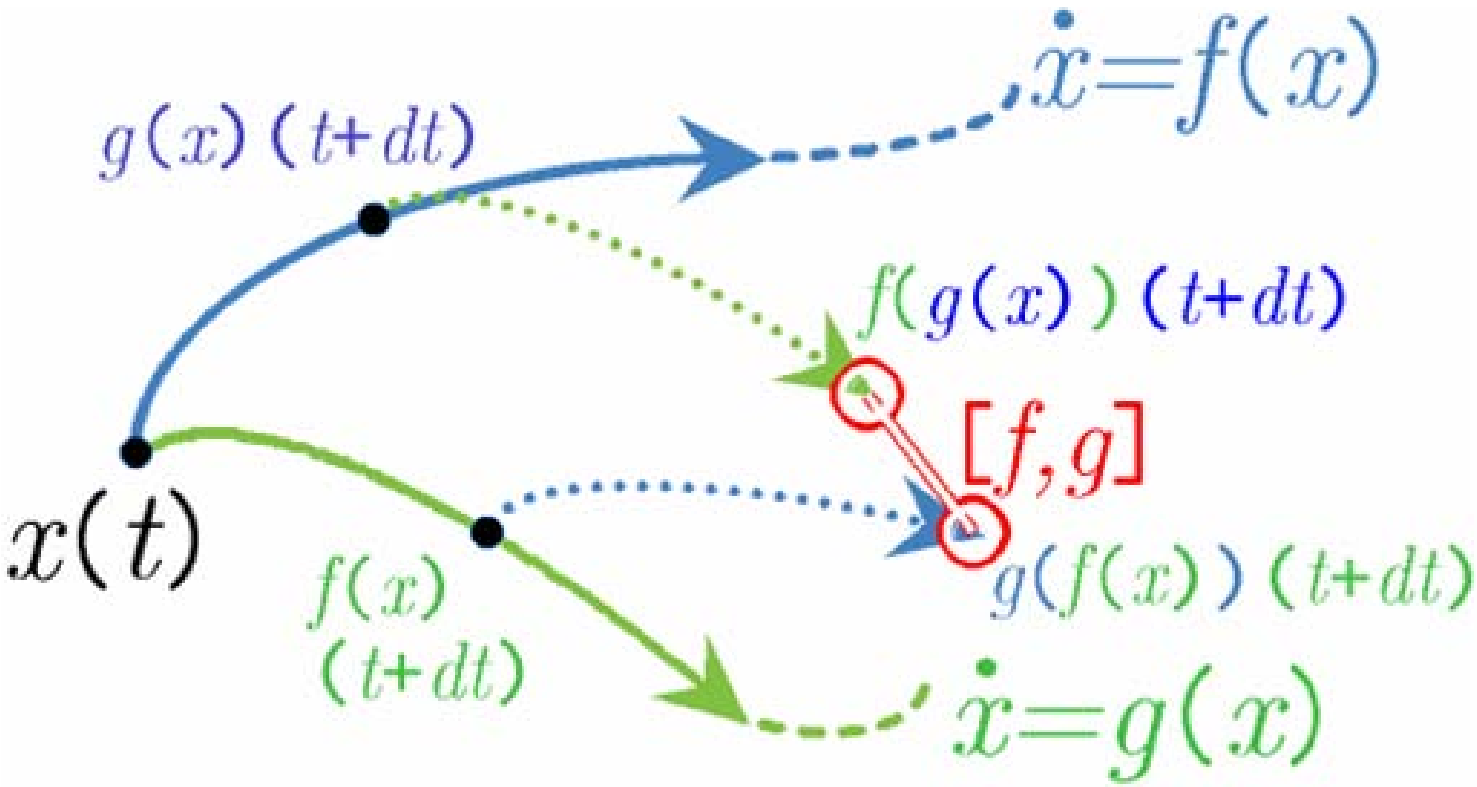
permet de calculer la condition de commutativité de deux flots $\Phi^t_{g_1}$ et $\Phi^s_{g_2}$.

Theorem

Soient g_1 et g_2 des champs de vecteurs $\mathcal{C}^\infty$ complets, définis sur $\mathcal{X}$ (par exemple $\mathbb{R}^n$). Alors :

$$\forall t, \forall s, \quad \Phi^t_{g_1} \circ \Phi^s_{g_2} = \Phi^s_{g_2} \circ \Phi^t_{g_1} \Leftrightarrow [g_1, g_2] = 0.$$

Lie bracket



Chochet de Lie.

Lie bracket

Proof.

Soient $x_0 \in \mathbb{R}^n$ et $t, s > 0$ donnés. Pour un champ de vecteurs analytique X, on a $\Phi_X^t(y) = y + tX(y) + R(t,y)$, où $R(t,y)$ représente un reste s'annulant pour $t \to 0$. On obtient donc :

$$\Phi_{g_1}^t \circ \Phi_{g_2}^s(x_0) = x_0 + (sg_2(x_0) + tg_1(x_0)) + st\frac{\partial g_1}{\partial x} g_2(x_0) + R_1(t,s,x_0),$$
$$\Phi_{g_2}^s \circ \Phi_{g_1}^t(x_0) = x_0 + (sg_2(x_0) + tg_1(x_0)) + st\frac{\partial g_2}{\partial x} g_1(x_0) + R_2(t,s,x_0),$$

et donc :

$$\Phi_{g_1}^t \circ \Phi_{g_2}^s(x_0) - \Phi_{g_2}^s \circ \Phi_{g_1}^t(x_0) = st[g_2, g_1](x_0) + R_3(t,s,x_0).$$

Prenons $t = s$, alors l'implication découle immédiatement. Pour la réciproque, $[g_1, g_2] = 0 \Rightarrow \forall x_0 \in \mathbb{R}^n : \lim_{t\to 0}(\frac{\Phi_{g_2}^{-t} \circ \Phi_{g_1}^s \circ \Phi_{g_2}^t(x_0) - \Phi_{g_1}^s(x_0)}{t}) = 0$. Soit la trajectoire $x(t) = \Phi_{g_2}^{-t} \circ \Phi_{g_1}^s \circ \Phi_{g_2}^t(x_0)$, alors $\dot{x}(t) = 0$, donc $\Phi_{g_2}^{-t} \circ \Phi_{g_1}^s \circ \Phi_{g_2}^t = \Phi_{g_1}^s$. □

Lie bracket : Propriétés

1. Bilinéaire sur $\mathbb{R}$:

$$[\alpha_1 g_1 + g_2, g] = \alpha_1 [g_1, g] + [g_2, g]$$
$$[g, \alpha_1 g_1 + g_2] = \alpha_1 [g, g_1] + [g, g_2]$$

2. anticommutativité :

$$[f, g] = -[g, f]$$

3. identité de Jacobi :

$$[f, [g, h]] + [g, [h, f]] + [h, [f, g]] = 0$$

4. flot (commutativité) :

$$\forall t, \forall s, \quad \Phi^t_{g_1} \circ \Phi^s_{g_2} = \Phi^s_{g_2} \circ \Phi^t_{g_1} \Leftrightarrow [g_1, g_2] = 0.$$

Lie bracket

En automatique, la non-commutativité des champs a une application très importante puisqu'elle permet de caractériser l'atteignabilité (version locale de la commandabilité) d'un système commandé du type

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

Lie bracket

Et de façon générale pour des systèmes de la forme

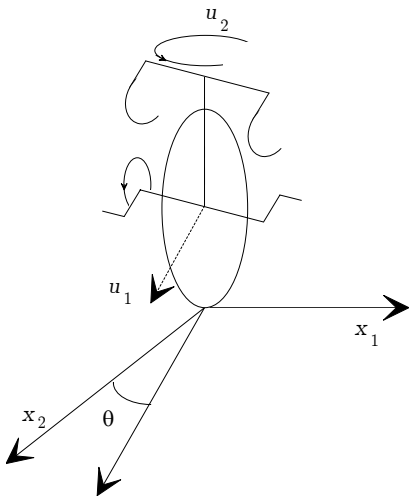
$$\dot{x} = f(x) + \sum_{i=1}^{p} g_i(x)u_i, x \in \mathbb{R}^n \quad (1)$$

pour cela il faut que

$$\text{rang}(\mathcal{A}\{f, g_1, g_2, \ldots, g_p\}) = n,$$

où $\mathcal{A}\{f, g_1, g_2, \ldots, g_p\}$ est l'algèbre de Lie engendrée par les champs de vecteurs $\{f, g_1, g_2, \ldots, g_p\}$.
f est le champs de dérive: il est à noter que les modèles cinématiques que l'on va rencontrer ici sont sans dérive, c'est-à-dire que $f = 0$.

Lie bracket



$$\begin{pmatrix}\dot{x}_1\\ \dot{x}_2\\ \dot{\theta}\end{pmatrix} = \begin{pmatrix}\sin\theta\\ \cos\theta\\ 0\end{pmatrix} u_1 + \begin{pmatrix}0\\ 0\\ 1\end{pmatrix} u_2.$$

$$g_1(x) = (\sin(\theta), \cos(\theta), 0)^T, \quad \Phi^t_{g_1} : \begin{pmatrix} x_{10}\\ x_{20}\\ \theta_0\end{pmatrix} \mapsto \begin{pmatrix} x_{10} + \sin(\theta_0)t\\ x_{20} + \cos(\theta_0)t\\ \theta_0\end{pmatrix},$$

$$g_2 = (0, 0, 1)^T, \quad \Phi^t_{g_2} : \begin{pmatrix} x_{10}\\ x_{20}\\ \theta_0\end{pmatrix} \mapsto \begin{pmatrix} x_{10}\\ x_{20}\\ \theta_0 + t\end{pmatrix},$$

$$\Rightarrow \Phi^t \circ \Phi^s \neq \Phi^s \circ \Phi^t$$

Crochet de Lie

$$([g_1, g_2], g_1, g_2) = \begin{pmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

$$\dim\ (\mathrm{vect}\{[g_1, g_2], g_1, g_2\}) = 3,$$

Definition

Soient $g_1, g_2, \ldots, g_p$ des champs de vecteurs que l'on suppose libres (idépendant les uns des autres). L'algèbre de Lie engendrée par les champs de vecteurs g_i est la distribution construite à partir de $\mathrm{vect}\{g_1, g_2, \ldots, g_p\}$ à laquelle on ajoute tous les crochets de Lie successifs formés des g_i à condition qu'ils augmentent la dimension de l'algèbre (c'est-à-dire qui ne sont pas déjà engendrés par la distribution que l'on est en train de construire !). On la note $\mathcal{A}\{g_1, g_2, \ldots, g_p\}$.

$\mathcal{A}$ = accéssibilité (controllabilité). Si $\mathrm{rang}(\mathcal{A}\{g_1, g_2, \ldots, g_p\}) = n$ alors le système $\dot{x} = G(x)u, x \in \mathbb{R}^n, G = (g_1, g_2, \ldots, g_p)$ (robot mobile) ou (1) est accessible (localement commandable).

Le crochet de Lie de deux champs de vecteurs g_1, g_2 est un nouveau champs de vecteur $[g_1, g_2]$, défini par

$$[g_1,g_2](x) = \left(\frac{\partial g_2}{\partial x} g_1 - \frac{\partial g_1}{\partial x} g_2\right)(x). \tag{2}$$

Table of Contents

Flatness is the key point

Flatness (see works from M. Fliess, J.Lévine, Ph.Martin, et P.Rouchon details in [**?**, **?**, **?**, **?**, **?**])

☞ For linear systems

$$\dot{x} = Ax + Bu$$

the following notions are equivalent :

1. controllability,
2. Brunovsky normal form,
3. the parametrization of the state variables and the inputs using m outputs (Brunovsky outputs = flat outputs).

Flatness is the key point

Theorem

Let us consider the following linear system $\dot{x} = Ax + Bu$, *if controllable, and* B *is full rank then there exist a regular static feedback* ($u = Pz + Qv, Q$ *invertible*) *and a change of coordinatee* ($z = Rx, R$ *invertible*) *such that*

$$\begin{aligned} y_1^{(\alpha_1)} &= v_1, \\ &\dots \\ y_m^{(\alpha_m)} &= v_m, \end{aligned}$$

with $z = (y_1, \dot{y}_1, \dots, y_1^{(\alpha_1 - 1)}, \dots, y_m, \dot{y}_m, \dots, y_m^{(\alpha_m - 1)})$ *and* α_i *positive integers.*

Flatness is the key point

Definition

System

$$\dot{x} = f(x, u), x \in \mathbb{R}^n, u \in \mathbb{R}^m,$$

is **flat** if there exist m functions of the state, the inputs and their derivatives up to order $r \leq n$ (**flat outputs**) such that the state variables and the outputs can be expressed in terms of the flat outputs.

☞ This is that there exist three mapping

Flatness is the key point

Definition (continued)

$$\varphi_y : \mathbb{R}^n \times \underbrace{\mathbb{R}^m \times \ldots \times \mathbb{R}^m}_{r+1} \mapsto \mathbb{R}^m, \varphi_x : \underbrace{\mathbb{R}^m \times \ldots \times \mathbb{R}^m}_{r} \mapsto \mathbb{R}^n,$$

$$\varphi_u : \underbrace{\mathbb{R}^m \times \ldots \times \mathbb{R}^m}_{r+1} \mapsto \mathbb{R}^m$$

such that

$$\begin{aligned} y &= \varphi_y(x, u, \dot{u}, \ddot{u}, \ldots, u^{(r)}), \\ x &= \varphi_x(y, \dot{y}, \ldots, y^{(r-1)}), \\ u &= \varphi_u(y, \dot{y}, \ldots, y^{(r)}). \end{aligned}$$

Flatness: double integrator

Example

$$\begin{aligned} \dot{x}_1 &= x_2 && (3)\\ \dot{x}_2 &= f(x) + u && (4)\\ y &= h(x) && (5) \end{aligned}$$

Flat output is $y_{\text{flat}} = x_1$

$$\begin{aligned} x_1 &= y_{\text{flat}},\\ x_2 &= \dot{x}_1 = \dot{y}_{\text{flat}},\\ y &= h(x_1, x_2) = h(y_{\text{flat}}, \dot{y}_{\text{flat}}),\\ u &= \dot{x}_2 - f(x) = \ddot{y}_{\text{flat}} - f(y_{\text{flat}}, \dot{y}_{\text{flat}}). \end{aligned}$$

Flatness: simple pendulum

Example

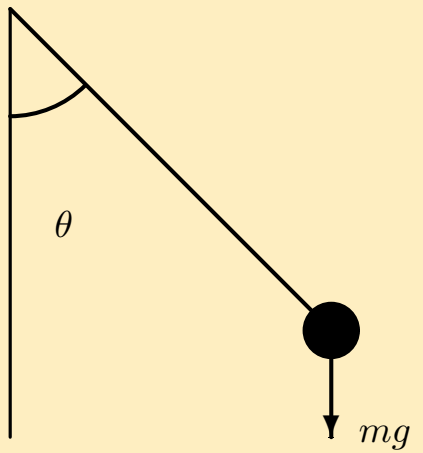
Model with friction:

$$\ddot{\theta} + \frac{\delta}{ml^2}\dot{\theta} + \frac{g}{l}\sin(\theta) = \frac{u}{ml^2}.$$

$y = \theta$ (it is also a flat output). Let $z_1 = \theta, z_2 = \dot{\theta}$ we get:

$$\begin{aligned} \dot{z}_1 &= z_2, \\ \dot{z}_2 &= -\frac{\delta}{ml^2} z_2 - \frac{g}{l}\sin(z_1) + \frac{u}{ml^2} \end{aligned}$$

Pendulum m, l with applied torque u.



Pendulum.

Flatness: simple pendulum

Example

$$\begin{aligned} \dot{z}_1 &= z_2, \\ \dot{z}_2 &= -\frac{\delta}{ml^2} z_2 - \frac{g}{l}\sin(z_1) + \frac{u}{ml^2} \end{aligned}$$

Flat output is $y_{\text{flat}} = z_1$

$$\begin{aligned} z_1 &= y_{\text{flat}}, \\ z_2 &= \dot{x}_1 = \dot{y}_{\text{flat}}, \\ y &= z_1 = y_{\text{flat}}, \\ u &= ml^2\ddot{y}_{\text{flat}} + \delta\dot{y}_{\text{flat}} + mgl\sin(y_{\text{flat}}). \end{aligned}$$

Flatness: SISO linear ctrb system

Any linear controllable SISO system is flat. Indeed, due to controllability

$$\frac{Y(p)}{U(p)} = \frac{n(p)}{d(p)}, n(p) \wedge d(p) = 1$$

Thus there exist a, b solution of the Bezout equation

$$a(p)n(p) + b(p)d(p) = 1$$

Flatness: SISO linear ctrb system

Flat output is $Y_{\text{flat}}(p) = \frac{U(p)}{d(p)}$ (sol. of Lin. ODE with input u).

$$a(p)n(p)Y_{\text{flat}}(p) + b(p)d(p)Y_{\text{flat}}(p) = Y_{\text{flat}}(p)$$

$$n(p)Y_{\text{flat}}(p) = \frac{n(p)}{d(p)}U(p) = Y(p)$$

$$d(p)Y_{\text{flat}}(p) = d(p)\frac{U(p)}{d(p)} = U(p)$$

$$a(p)Y(p) + b(p)U(p) = Y_{\text{flat}}(p)$$

Flatness: SISO linear ctrb system

$$\begin{aligned} y &= \varphi_x(y, \dot{y}, \ldots, y^{(\deg d)}), \\ u &= \varphi_u(y, \dot{y}, \ldots, y^{(\deg n)}). \end{aligned}$$

Ctrb : $\mathrm{rank} C(A,B) = n, C(A,B) = [B, AB, \ldots, A^{n-1}B]$.
Change of var. $z = C(A,B)x$

$$\begin{aligned} \dot{z}_1 &= -a_0 z_n + u, \\ \dot{z}_n &= z_1 - a_1 z_n, \\ &\ldots \\ \dot{z}_1 &= z_{n-1} - a_{n-1} z_n, \\ y_{\text{flat}} &= z_n \end{aligned}$$

Flatness: Fully actuated manipulator

Dynamics: Let $q = (q_1, \ldots, q_n), q \in \mathcal{M}_q$ (state) where $\mathcal{M}_{\text{II}}$ is the configuration space (state space). For each d.o.f (en français: d.d.l) there is an actuator u has dimension n.
Using Euler Lagrange $\mathcal{L} = \mathcal{E}_c - \mathcal{E}_p$, $\mathcal{E}_c(q, \dot{q}), \mathcal{E}_p(q)$,

$$\frac{d}{dt}(\frac{\partial \mathcal{L}}{\partial \dot{q}_i}) - \frac{\partial \mathcal{L}}{\partial q_i} + \frac{\partial D}{\partial \dot{q}_i} = D_i.$$

$$J(q)\ddot{q} + C(q, \dot{q}) + G(q) = u$$

Flat output is $y_{\text{flat}} = q$
State $x = (q, \dot{q}) = (y_{\text{flat}}, \dot{y}_{\text{flat}})$
Input $u = J(q)\ddot{q} + C(q, \dot{q}) + G(q) = F(y_{\text{flat}}, \dot{y}_{\text{flat}}, \ddot{y}_{\text{flat}})$ If and ONLY if n fully actuated d.o.f

Kinematic model: flatness is the key point

Theorem (P. Martin et P. Rouchon [?] (see also [?, ?]))

Any driftless non linear system

$$\dot{x} = B(x)u$$

(which is the case for non holonomic mobile robots with m inputs and **at most** *$m+2$ states is flat.*

$\exists$ 3 functions: one defining m *flat outputs* (thus differentially independent) in terms of q, $u, \dot{u}, \ldots, u^{(a)}$) and two other functions one for q the other for u allowing to express them in terms of the output and its time derivatives (in finite number).

Kinematic model: flatness is the key point

- ☞ Thus the PKM and PDM are flat.
- ☞ Thus it implies that they are controllable.
- ☞ But from Brockett's theorem (see [**?**]) they are not stabilizable by a continuous static time-invariant state feedback.

Kinematic model: flatness is the key point

1. Unicycle mobile robot (type (2,0))

$$\begin{aligned} \dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta \\ \dot{\theta} &= w \end{aligned} \tag{6}$$

2. Car-like mobile robot (type (1,1))

$$\begin{aligned} \dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta \\ \dot{\theta} &= v\frac{\tan(\phi)}{l} \\ \dot{\phi} &= w \end{aligned} \tag{7}$$

Kinematic model: flatness is the key point

Flat Outputs: (x, y).

Indeed:

1. for (82): $\theta = \arctan\left(\frac{\dot{y}}{\dot{x}}\right), v = \pm\sqrt{\dot{x}^2 + \dot{y}^2}, w = \frac{\dot{x}\ddot{y} - \dot{y}\ddot{x}}{\dot{x}^2 + \dot{y}^2}$
2. for (7): $\theta = \arctan\left(\frac{\dot{y}}{\dot{x}}\right), v = \pm\sqrt{\dot{x}^2 + \dot{y}^2}, \phi = \arctan\left(l\frac{\dot{x}\ddot{y} - \dot{y}\ddot{x}}{(\dot{x}^2 + \dot{y}^2)^{3/2}}\right), w = \dot{\phi}.$

Flatness: what else ?

Advantages of this point of view:

- There exists a bijective mapping between admissible trajectories of the dynamical system and the trajectories of the flat outputs,
- One can completly parameterized a nominal trajectory using interpolating polynomial Bésier, Hermite, B-spline . . . (it may be an equilibrium point),
- Usefull to get open loop control,
- Usefull for sizing actuators (dimensionnement en français),
- Usefull for closed-loop control design (flatness $\Leftrightarrow$ dynamic feedback liberalization),
- There exist extensions for other classes of dynamical systems (delays, PDE, etc . . .)

Flatness: Hot air balloon

Hot air balloon

"Altitude"

$$\begin{aligned} \dot{z}_1 &= z_2, \\ \dot{z}_2 &= -\frac{1}{\tau_2} z_2 + \alpha(\theta - \theta_{\text{ambiant}}(z_1)) + p_{\text{wind}} \\ \dot{\theta} &= -\frac{1}{\tau_1(z_1)}(\theta - \theta_{\text{ambiant}}(z_1)) + u \end{aligned} \tag{8}$$

Measured variable $y = z_1$. p_{wind} wind action (speed), z_1 altitude $\tau_1(z_1) > 0$, u heater action.
A.N. : $\alpha = 0.5$, $\tau_2 = 10^{-1}$,
$\theta_{\text{ambiant}}(z_1) = 25 - \frac{90}{10000} z_1$,
$\tau_1(z_1) = 3600 \left(1 - \frac{z_1}{20000}\right)$

Flatness: Hot air balloon

☞ No wind action $p = 0$: Flat output is $y_{\text{flat}} = z_1$

$$
\begin{aligned}
z_1 &= y_{\text{flat}}, \\
z_2 &= \dot{x}_1 = \dot{y}_{\text{flat}}, \\
\theta &= \theta_{\text{ambiant}}(y_{\text{flat}}) + \frac{1}{\alpha}\left(\ddot{y}_{\text{flat}} + \frac{1}{\tau_2}\dot{y}_{\text{flat}}\right), \\
y &= z_1 = y_{\text{flat}}, \\
u &= \dot{\theta} + \frac{1}{\tau_1(z_1)}(\theta - \theta_{\text{ambiant}}(z_1)) \\
&= \frac{\partial \theta_{\text{ambiant}}}{\partial z_1}(y_{\text{flat}})\dot{y}_{\text{flat}} + \frac{1}{\alpha}\left(y^{(3)}_{\text{flat}} + \frac{1}{\tau_2}\ddot{y}_{\text{flat}}\right) \\
&\quad + \frac{1}{\alpha\tau_1(z_1)}(\ddot{y}_{\text{flat}} + \frac{1}{\tau_2}\dot{y}_{\text{flat}})
\end{aligned}
$$

Flatness: Hot air balloon

☞ Design a heating process and control such that the operating range of the hot air balloon is : $z_1 \in [0, 1000]m$.

☞ 1rst STEP (trajectory planning): Find $y^N_{\text{flat}}(t)$ such that

$$y^N_{\text{flat}}(t_i) = y_i \quad (9)$$
$$y^N_{\text{flat}}(t_f) = y_f \quad (10)$$
$$\dot{y}^N_{\text{flat}}(t_i) = 0 \quad (11)$$
$$\dot{y}^N_{\text{flat}}(t_f) = 0 \quad (12)$$

Smooth trajectory. Bézier interpolation

$$y^N_{\text{flat}}(t) = a_0 + a_1\tau + a_2\tau^2 + a_3\tau^3, \tau = \frac{t - t_i}{t_f - t_i}$$

$$\dot{y}^N_{\text{flat}}(t) = \dot{\tau}\left(a_1 + 2a_2\tau + 3a_3\tau^2\right), \dot{\tau} = \frac{1}{t_f - t_i}$$

Flatness: Hot air balloon

$$\begin{pmatrix} y_i \\ 0 \\ y_f \\ 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \dot{\tau} & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & \dot{\tau} & 2\dot{\tau} & 3\dot{\tau} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix}$$

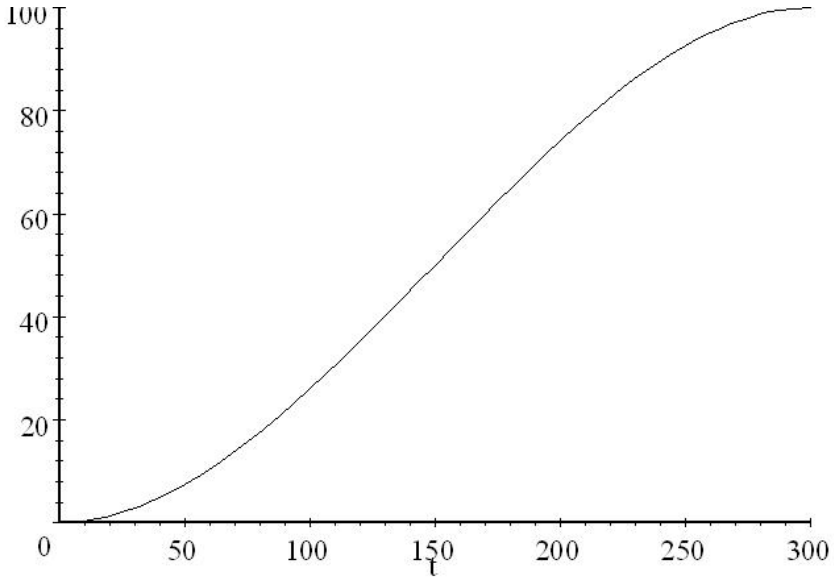
$$\begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1/\dot{\tau} & 0 & 0 \\ -3 & -2/\dot{\tau} & 3 & -1/\dot{\tau} \\ 2 & 1/\dot{\tau} & -2 & 1/\dot{\tau} \end{pmatrix} \begin{pmatrix} y_i \\ 0 \\ y_f \\ 0 \end{pmatrix} = \begin{pmatrix} y_i \\ 0 \\ 3(y_f - y_i) \\ -2(y_f - y_i) \end{pmatrix}$$

$$y^N_{\text{flat}}(t) = y_i + (y_f - y_i)\tau^2(3 - 2\tau), \tau = \frac{t - t_i}{t_f - t_i}$$

Flatness: Hot air balloon

☞ Trajectory: $y_i = 0, y_f = 100, t_f - t_i = 300, t_i = 0$

$$y^N_{\text{flat}}(t) = 100\tau^2(3 - 2\tau), \tau = \frac{t}{300}$$



Path planning Hot air balloon

Flatness: Hot air balloon

☞ 2nd STEP (Open loop control and simulation):

$$u^N = \frac{\partial \theta_{\text{ambiant}}}{\partial z_1}(y^N_{\text{flat}})\dot{y}^N_{\text{flat}} + \frac{1}{\alpha}\left(y^{N(3)}_{\text{flat}} + \frac{1}{\tau_2}\ddot{y}^N_{\text{flat}}\right) + \frac{1}{\alpha\tau_1(y^N_{\text{flat}})}(\ddot{y}^N_{\text{flat}} + \frac{1}{\tau_2}\dot{y}^N_{\text{flat}}), \quad (13)$$

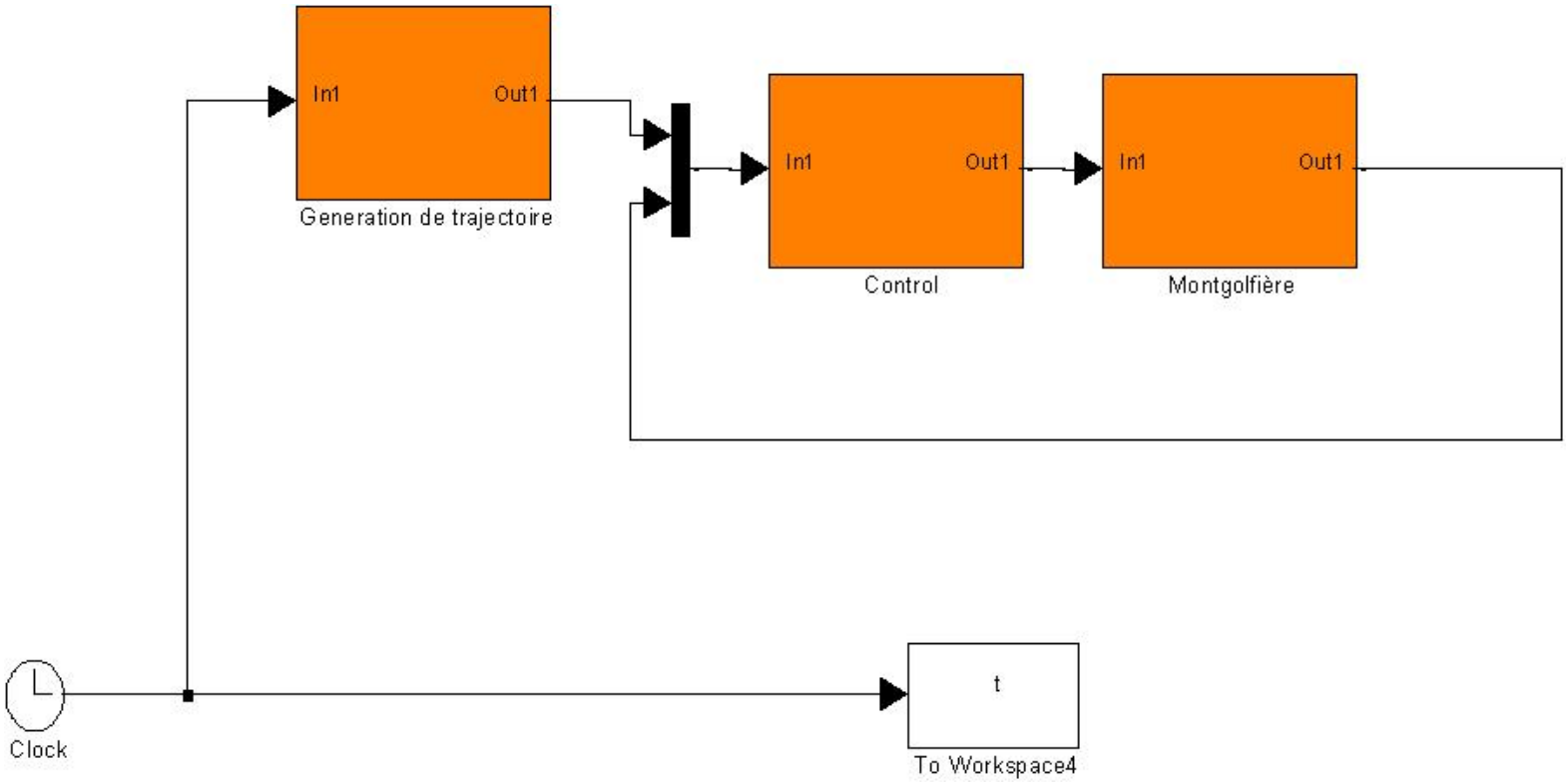
$$y^N_{\text{flat}}(t) = y_i + (y_f - y_i)\frac{(t-t_i)^2}{(t_f-t_i)^2}\left(3 - 2\frac{t-t_i}{t_f-t_i}\right), \quad (14)$$

$$\dot{y}^N_{\text{flat}}(t) = \frac{(y_f - y_i)(t-t_i)}{(t_f-t_i)^2}\left(6 - 2\frac{t-t_i}{t_f-t_i}\right), \quad (15)$$

$$\ddot{y}^N_{\text{flat}}(t) = -12\frac{(y_f - y_i)}{(t_f-t_i)^3}, \quad (16)$$

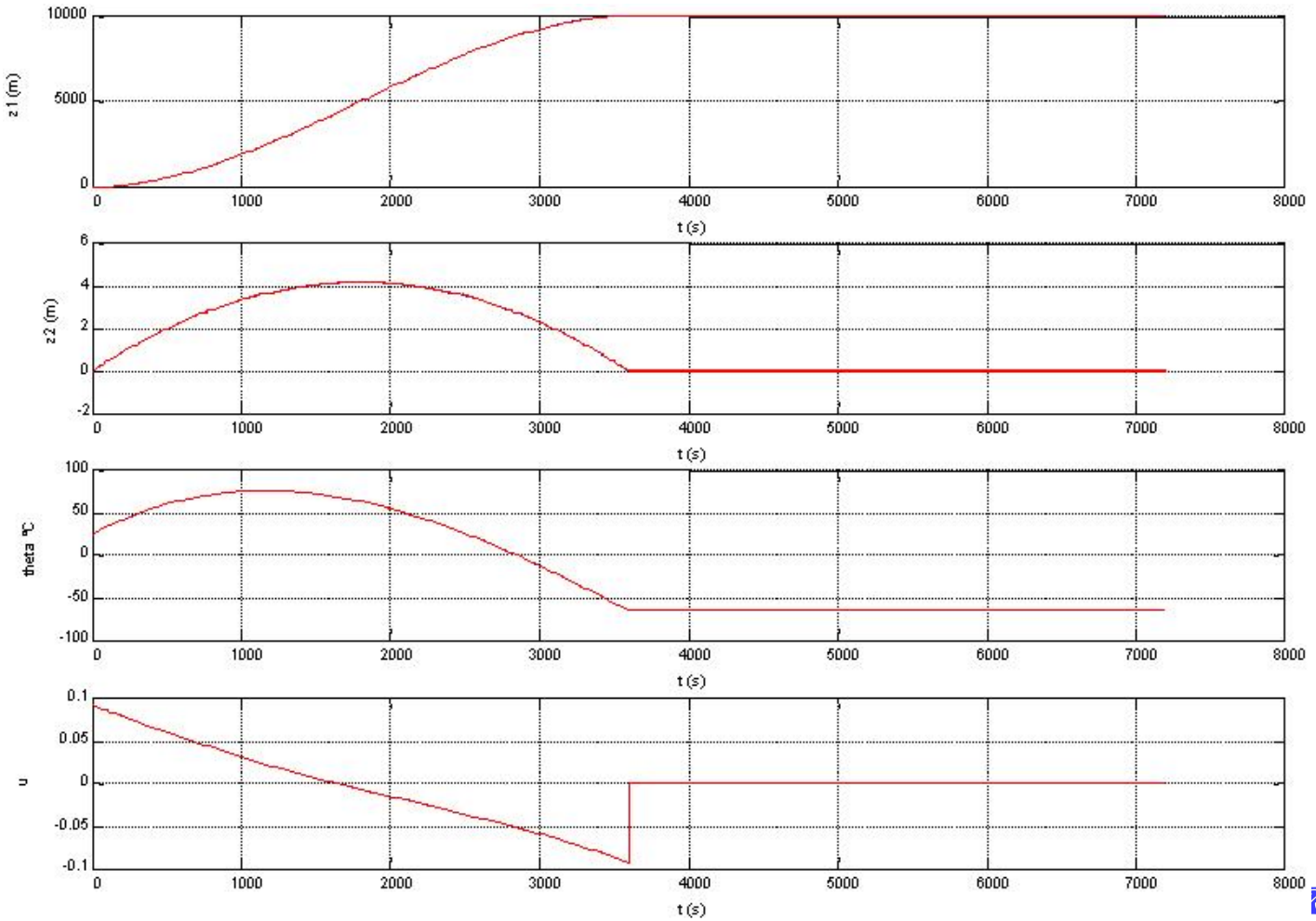
$$y^{N(3)}_{\text{flat}}(t) = 0. \quad (17)$$

Flatness: Hot air balloon



Simulink

Flatness: Hot air balloon



Open loop control

Flatness: Hot air balloon

☞ 3rd STEP: Heating process (sizing) If $|v| \leq v_{\max}$ and $|a| \leq a_{\max}$ then

$$\left| \frac{(y_f - y_i)(t - t_i)}{(t_f - t_i)^2} \left(6 - 2\frac{t - t_i}{t_f - t_i} \right) \right| \leq v_{\max} \quad (18)$$

$$\left| 12\frac{(y_f - y_i)}{(t_f - t_i)^3} \right| \leq a_{\max} \quad (19)$$

A.N: $v_{\max} = 5\ m/s$, $a_{\max} = 0.05\ m/s^2$, $\delta y = 1000\ m$, $\delta t = 3\ h$

$$4.4 \leq 5m/s \quad (20)$$

$$3.215 \times 10^{-7} \leq 5 \times 10^{-2} m/s^2 \quad (21)$$

Flatness: Hot air balloon

$$
\begin{aligned}
u^N &= \frac{\partial \theta_{\text{ambiant}}}{\partial z_1}(y_{\text{flat}})\dot{y}^N_{\text{flat}} + \frac{1}{\alpha}\left(y^{(3)}_{\text{flat}} + \frac{1}{\tau_2}\ddot{y}^N_{\text{flat}}\right) \\
&\quad + \frac{1}{\alpha\tau_1(y^N_{\text{flat}})}(\ddot{y}^N_{\text{flat}} + \frac{1}{\tau_2}\dot{y}^N_{\text{flat}}) \qquad (22) \\
&\max \quad (\text{each term}) \\
u^N_{\max} &= \frac{90}{10000}(3.7) + \frac{1}{0.05}\left(2.0576 \times 10^{-5}\right) \qquad (23) \\
u^N_{\max} &= 5.5 \times 10^{-2} \qquad (24)
\end{aligned}
$$

Be careful computation done with $p = 0$ otherwise more power !!

Flatness: Hot air balloon

☞ 4th STEP: Closed Loop system (and simulation)

$$e = z_1 - z_{1c}(t), z_{1c}(t) = y^N_{\text{flat}}(t). \quad (25)$$

$$u = \frac{\partial \theta_{\text{ambiant}}}{\partial z_1}(y_{\text{flat}})\dot{y}_{\text{flat}} + \frac{1}{\alpha}\left(y^{(3)}_{\text{flat}} + \frac{1}{\tau_2}\ddot{y}_{\text{flat}}\right) + \frac{1}{\alpha}v + \frac{1}{\alpha\tau_1(y_{\text{flat}})}(\ddot{y}_{\text{flat}} + \frac{1}{\tau_2}\dot{y}_{\text{flat}}) \quad (26)$$

$$y^{(3)}_{\text{flat}} = v + \dot{p}, \sup|\dot{p}| \leq \pi \quad (27)$$

$$y^{N(3)}_{\text{flat}} = 0 \quad (28)$$

$$v = -K\text{sign}(e) - k_0 e - k_1 \dot{e} - k_2 \ddot{e} \quad (29)$$

$$e^{(3)} + k_2\ddot{e} + k_1\dot{e} + k_0 e = \dot{p} - K\text{sign}(e)$$

Flatness: Hot air balloon

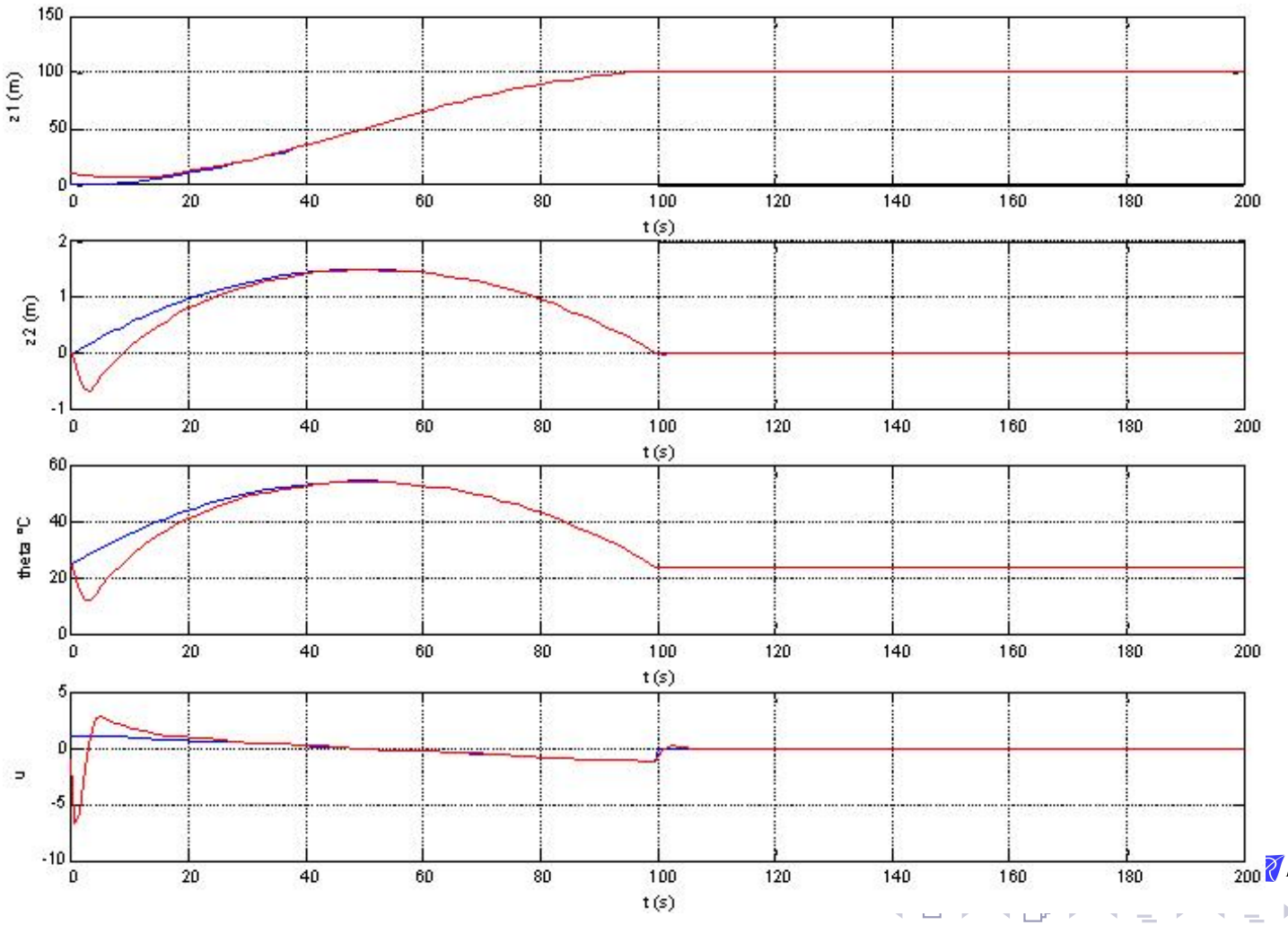


Table of Contents

Path planning: Problem formulation

Manipulator

Configuration: $q = (q_1, \ldots, q_n), q \in \mathcal{M}_q$ (state), $\mathcal{M}$ is the configuration space (state space).
Dynamics: using Euler Lagrange $\mathcal{L} = \mathcal{E}_c - \mathcal{E}_p$,
$\frac{d}{dt}(\frac{\partial \mathcal{L}}{\partial \dot{q}_i}) - \frac{\partial \mathcal{L}}{\partial q_i} + \frac{\partial D}{\partial \dot{q}_i} = D_i$.

$$J(q)\ddot{q} + C(q, \dot{q}) + G(q) = \tau$$

Posture (EF position and orientation):
$x = (x_1, \ldots, x_6), x \in \mathcal{M}_x$, where $\mathcal{M}_x$ is the work space or operational space of $\dim = 6$; Position $(x_1, x_2, x_3) =$ coordinates of $\mathcal{O}_{n+1}$ and orientation (x_4, x_5, x_6) is given by the rotation matrix

$$T_0^n(q) = T_0^1(q_1)T_1^2(q_2)\ldots T_{n-1}^n(q_n)$$

Direct geometric model (manipulator):

Mobile Robot

Configuration: $q = (q_1, \ldots, q_n), q \in \mathcal{M}_q$ (state), $\mathcal{M}$ is the configuration space (state space).
Kinematic model: Non holonomic constraint $H(q)\dot{q} = 0$, where H is a full rank $(n - m \times n)-$matrix, leading to state equation (Kinematic model is a driftless system):

$$\dot{q} = G(q)u, q \in \mathbb{R}^n, u \in \mathbb{R}^m$$

Dynamic model: using Euler Lagrange
$J(q)\dot{u} + C(q, \dot{q}, u) + G(q) = \tau$
Posture ($(x, y)-$position and orientation θ):
$x = (x, y, \theta), x \in \mathcal{M}_x$, where $\mathcal{M}_x$ is the work space or operational space of $\dim = 3$
Direct geometric model (mobile robot):

$$f : \begin{array}{ccc} \mathcal{M}_q & \rightarrow & \mathcal{M}_x \\ q & \mapsto & x = f(q) \end{array} \quad (30)$$

(frequently x is a part of q).

☞ Constraints from physics : $|v| < v_{\max}, |a| < a_{\max}, \ldots$,
☞ Controllability, flatness (full parametrization using flat outputs)

Path planning: Problem formulation

Manipulator

Path planning of the Posture (EF position and orientation):
$x = (x_1, \ldots, x_6), x \in \mathcal{M}_x$ be careful $\mathcal{M}_x$ is a subspace of $\mathbb{R}^6$ (not all positions are reachable!)

$$\dot{x} = J_f \dot{q} \quad (31)$$
$$J(q)\ddot{q} + C(q, \dot{q}) + G(q) = u \quad (32)$$

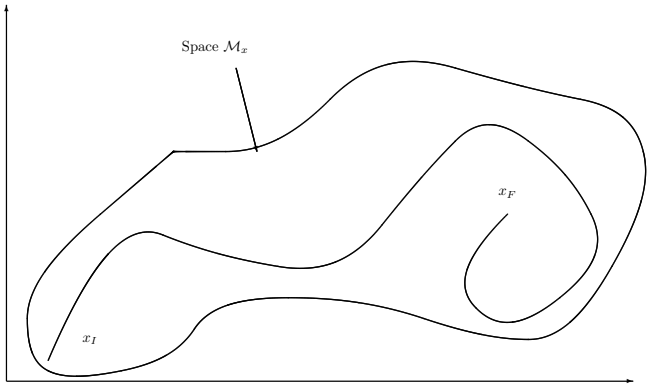
Main problem is to transform an admissible path for x into a trajectory in q (flat outputs).

Mobile Robot

Path planning of the Posture : $X = (x, y, \theta), X \in \mathcal{M}_x$,

$$\dot{q} = B(q)u \quad (33)$$
$$J(q)\ddot{q} + C(q, \dot{q}) + G(q) = u \quad (34)$$

Main problem is to transform an admissible path for x into a trajectory in terms of flat outputs



Constraints from physics : $|v| < v_{\max}, |a| < a_{\max}, \ldots$

Path planning: Problem formulation

$$\dot{z} = f(z) + g(z)u \quad (35)$$

$$y_{\text{flat}} = h(z) \quad (36)$$

z is the state $(q, \dot{q})$ Posture : $x = P(y_{\text{flat}})$ (see the difference between manipulator and mobile robots)

$$Path : [0,1] \rightarrow \mathcal{M}_x \quad (37)$$

$$s \mapsto Path(s) \in \mathcal{M}_x \quad (38)$$

Path planning: Problem formulation

Problem: Let us define $\mathcal{M}_{y_{\text{flat}}}$ such that

$$y_{\text{flat}} \in \mathcal{M}_{y_{\text{flat}}} \Rightarrow x \in \mathcal{M}_x.$$

Find

$$\begin{aligned} y_{\text{flat}}^N : [0,1] &\rightarrow \mathcal{M}_{y_{\text{flat}}} && (39) \\ s &\mapsto y_{\text{flat}}^N(s) && (40) \end{aligned}$$

such that

$$Path([0,1]) = (P \circ y_{\text{flat}}^N)([0,1])$$

Main obstruction is that P is not one-to-one !!

Path planning: Problem formulation

Once the y_{flat}^N is obtained from the flatness property one gets:

- the nominal state trajectory (reference trajectory for tracking)

$$x^N = \varphi_x(y_{\text{flat}}^N, \dot{y}_{\text{flat}}^N, \ldots, y_{\text{flat}}^{N(r-1)})$$

- the nominal control (open loop control)

$$u^N = \varphi_u(y_{\text{flat}}^N, \dot{y}_{\text{flat}}^N, \ldots, y_{\text{flat}}^{N(r)}).$$

Path planning: a simple example

Table of Contents

Path planning: Polynomial interpolation

☞ Let $P_i = (x_i, y_i), i = 1, \ldots, r$ be given points in the plan $\mathbb{R}^2$, one would like to find a polynomial P such that:

$$P(x_i) = y_i, i = 1, \ldots, r \tag{41}$$

Theorem

A necessary and sufficient condition such that there exists a unique polynomial L with degree at most equal to r satisfying (41) is that $x_i \neq x_j, \forall i \neq j$.

Path planning: Polynomial interpolation

Lagrange polynomial interpolation:

$$L(x) = \sum_{i=1}^{r} L_i(x) y_i, \tag{42}$$

$$L_i(x) = \prod_{i=1, i \neq j}^{r} \frac{x - x_i}{x_j - x_i} \tag{43}$$

If $y_i = f(x_i)$ for a given function f there exist results about the interpolation error, the choice of the control points x_i and the convergence rate.

Path planning: Polynomial interpolation

☞ If we look at an interpolating polynomial which satisfy (41) and

$$P'(x_i) = y'_i, i = 1, \ldots, r \tag{44}$$

Theorem

A necessary and sufficient condition such that there exists a unique polynomial H with degree at most equal to $2r + 1$ satisfying (41) and (44) is that $x_i \neq x_j, \forall i \neq j$.

Path planning: Polynomial interpolation

Hermite polynomial interpolation:

$$H(x) = \sum_{i=1}^{r} H_i(x) y_i + \sum_{i=1}^{r} V_i(x) y_i', \quad (45)$$

$$H_i(x) = (1 - 2(x - x_i) L_i'(x_i)) L_i^2(x) \quad (46)$$

$$V_i(x) = (x - x_i)^2 L_i^2(x) \quad (47)$$

$$L_i(x) = \prod_{i=1, i \neq j}^{r} \frac{x - x_i}{x_j - x_i} \quad (48)$$

If $y_i = f(x_i)$ for a given function f there exist results about the interpolation error, the choice of the control points x_i and the convergence rate.

Path planning: Splines functions

Bernstein Polynomial:

$$B_k^n(x) = \mathbf{C}_n^k (1-x)^{n-k} x^k, 0 \leq k \leq n, \mathbf{C}_n^k = \frac{n!}{k!(n-k)!} \quad (49)$$

We have the following properties:

$$B_0^{n+1}(x) = (1-x)B_0^n(x) \quad (50)$$
$$B_k^{n+1}(x) = xB_{k-1}^n(x) + (1-x)B_k^n(x) \quad (51)$$
$$B_{n+1}^{n+1}(x) = xB_n^n(x) \quad (52)$$
$$(53)$$

Roots : 0 (k), 1 $(n-k)$
Optimum at $x = \frac{k}{n}$.

Path planning: Splines functions

$$\sum_{k=0}^{n} B_k^n(x) = 1 \quad (54)$$

$$\sum_{k=0}^{n} \frac{k}{n} B_k^n(x) = x \quad (55)$$

$$\sum_{k=0}^{n} \frac{k(k-1)}{n(n-1)} B_k^n(x) = x^2 \quad (56)$$

$B_k^n(x)$ basis: any polynomial $P(x) = \sum_{k=0}^{n} a_k x^k$ is a linear combination of the $B_k^n(x)$

$$P(x) = \sum_{k=0}^{n} b_k B_k^n(x)$$

Path planning: Splines functions

Plan $\mathbb{R}^2$:

- the Bézier points (or control points) are given by $(\frac{i}{n}, b_i)$, for $i = 0, \ldots, n$,
- the polygonal line joinning the Bézier points is called the Bézier polygone,
- the Bézier polygone contains the graphe of P which contains itself the points $(0, b_0), (1, b_n)$
- in order to modify the form just modify the Bézier points.

Path planning: Splines functions

One can get recursively the values of P and its successive derivatives using the De Casteljau Algorithm from the Bézier coefficients:

$$b_{rs}(x) = \sum_{i=r}^{s} b_i B_{i-r}^{s-r}(x), 0 \leq r \leq s \leq n \quad (57)$$

$$b_{rr}(x) = b_r \quad (58)$$

$$b_{rs}(x) = (1-x)b_{r,s-1} + x b_{r+1,s} \quad (59)$$

$$P(x) = b_{0n}(x) \quad (60)$$

$$P^{(k)}(x) = \frac{n!}{(n-k)!}\Delta^k b_{0n}(x) \quad (61)$$

(62)

Path planning: Splines functions

☞ Let $P_i = (x_i, y_i), i = 1, \ldots, n$ be given points in the plan $\mathbb{R}^2$, let us define a sequence of mapping from $[0, 1]$:

$$B_0(P_i)(t) = P_i, \forall i = 1, \ldots, r \quad (63)$$

$$B_k(P_i, \ldots, P_{i+k})(t) = (1-t)B_{k-1}(P_0, \ldots, P_{i+k-1})(t) \quad (64)$$

$$+tB_{k-1}(P_{i+1}, \ldots, P_{i+k}) \quad (65)$$

$$(66)$$

Path planning: Splines functions

☞ Bézier curve associated to the P_i is the parameterized arc defined by:

$$B_n(P_0, \ldots, P_n) : t \mapsto B_n(P_0, \ldots, P_n)(t)$$

Denoting by $B_{rs} : t \mapsto B_{s-r}(P_r, \ldots, P_s)(t)$ for $r \leq s \in \mathbb{N}$ we have

$$B_{rs}(t) = \sum_{i=r}^{s} B_{i-r}^{s-r}(t) P_i \tag{67}$$

$$B_{0n}(t) = \sum_{i=0}^{n} B_i^n(t) P_i \tag{68}$$

(69)

Path planning: Splines functions

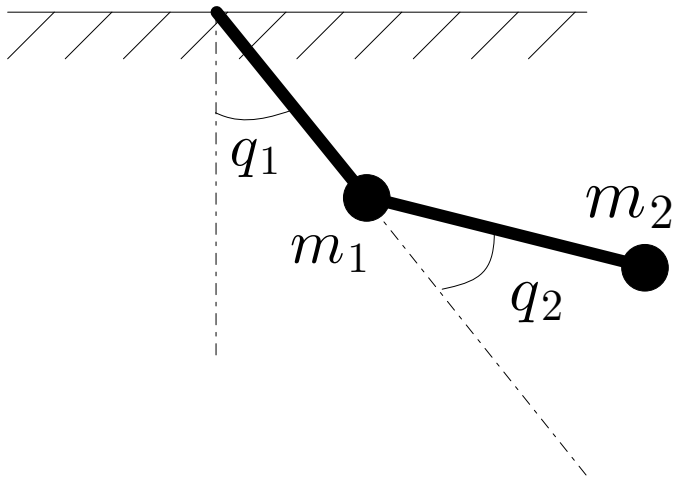
Path planning: Splines functions

Path planning: Splines functions

Path planning: Splines functions

Table of Contents

Path planning: 2 d.d.l manipulator



$$J(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau + \tau_{\text{ext}} \tag{70}$$

where τ is the control and $\tau_{\text{ext}} = 0$ is the load (actually there is no load).

Path planning: 2 d.d.l manipulator

The x-axis is vertical and y-axis is horizontal: the end effector has position

$$(x, y) = (l_1 \cos(q_1) + l_2 \cos(q_1 + q_2), l_1 \sin(q_1) + l_2 \sin(q_1 + q_2)),$$

$$X = f(q),$$

Note that we have a one to one map. And using the classical notation $c_{i_1,i_2,\ldots,i_n} = \cos(q_{i_1} + q_{i_2} + \ldots + a_{i_n})$, $s_{i_1,i_2,\ldots,i_n} = \sin(q_{i_1} + q_{i_2} + \ldots + a_{i_n})$ the posture is given by

$$(x, y) = (l_1 c_1 + l_2 c_{12}, l_1 s_1 + l_2 s_{12})$$

Path planning: 2 d.d.l manipulator

One can also use polar coordinates

$$
\begin{aligned}
z &= \rho \exp(i\theta) = l_1 \exp(iq_1) + l_2 \exp(i(q_1 + q_2)) && (71)\\
&= \exp(iq_1)(l_1 + l_2 \exp(iq_2)), && (72)\\
\rho &= \sqrt{(l_1 + l_2 c_2)^2 + l_2^2 s_2^2}, && (73)\\
\theta &= q_1 + \arctan\left(\frac{l_2 s_2}{l_1 + l_2 c_2}\right) && (74)
\end{aligned}
$$

Path planning: 2 d.d.l manipulator

$$
\begin{aligned}
J(q) &= \begin{pmatrix} a_{11}(q_2) & a_{12}(q_2) \\ a_{12}(q_2) & a_{22} \end{pmatrix}, \\
a_{11}(q_2) &= m_1 l_1^2 + m_2 l_1^2 + m_2 l_2^2 + 2 m_2 l_1 l_2 \cos(q_2), \\
a_{12}(q_2) &= m_2 l_2^2 + m_2 l_1 l_2 \cos(q_2), \\
a_{22} &= m_2 l_2^2, \\
C(q,\dot{q}) &= m_2 l_2 l_1 \sin(q_2) \begin{pmatrix} -\dot{q}_2^2 - 2\dot{q}_1\dot{q}_2 \\ \dot{q}_1^2 \end{pmatrix}, \\
G &= -g \begin{pmatrix} m_2 l_2 \sin(q_1 + q_1) + (m_1 + m_2) l_1 \sin(q_1) \\ m_2 l_2 \sin(q_1 + q_1) \end{pmatrix}, \\
\tau &= \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}.
\end{aligned}
\tag{75}
$$

Path planning: 2 d.d.l manipulator

Numerical values: $l_1 = 1$ (m), $l_2 = 0.5$ m, $m_1 = 1$ kg, $m_2 = 0.4$ kg,$g = 9.81$ m.s^{-2}.
Considering the masses of the links and some external load, the terms $\Delta M(q), \Delta C(q, \dot{q}), \Delta G(q)$ and τ_{ext} must be introduced.
Assuming that the linear masses of the links are respectively 0.2 kg/m and 0.1 kg/m

Path planning: 2 d.d.l manipulator

Flat output : $y_{\text{flat}} = (q_1, q_2)$, $M_{y_{\text{flat}}} = \mathbb{S}^2$
Posture: $z = l_1 \exp(iq_1) + l_2 \exp(i(q_1 + q_2)) = P(y_{\text{flat}})$
Path: chosen as a circle of radius $\frac{l_1+l_2}{2}$:

$$Path : [0,1] \rightarrow \mathcal{M}_x \quad (76)$$

$$s \mapsto Path(s) = \frac{l_1 + l_2}{2} \exp(2i\pi s) \in \mathcal{M}_x \quad (77)$$

Path planning: 2 d.d.l manipulator

$$y_{\text{flat}}^N : [0,1] \rightarrow \mathcal{M}_{y_{\text{flat}}} \quad (78)$$

$$s \mapsto y_{\text{flat}}^N(s) = (q_1(s), q_2(s))^T \quad (79)$$

such that

$$\frac{l_1 + l_2}{2} \exp(is) = (l_1 \exp(iq_1(s)) + l_2 \exp(i(q_1(s) + q_2(s)))$$

Path planning: 2 d.d.l manipulator

$$4\left[(l_1 + l_2 c_2)^2 + l_2^2 s_2^2\right] = (l_1 + l_2)^2 \quad (80)$$

$$2\pi s = q_1 + \arctan\left(\frac{l_2 s_2}{l_1 + l_2 c_2}\right) \quad (81)$$

One can chose a constant angle for q_2 such that 80 hold which ends the path planning!! But in that case we did not take into account velocities (initial and final), constraints and dynamics...

Table of Contents

Path planning: (2,0)-mobile robot

$$\begin{aligned}\dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta \\ \dot{\theta} &= w\end{aligned} \tag{82}$$

Flat Outputs: (x, y). Indeed for (82):

$$\theta = \arctan\left(\frac{\dot{y}}{\dot{x}}\right) \tag{83}$$

$$v = \pm\sqrt{\dot{x}^2 + \dot{y}^2} \tag{84}$$

$$w = \frac{\dot{x}\ddot{y} - \dot{y}\ddot{x}}{\dot{x}^2 + \dot{y}^2} \tag{85}$$

Path planning: (2,0)-mobile robot

Path planning: find $y = f(x)$ such that

$$f(x_i) = y_i \quad (86)$$
$$f(x_f) = y_f \quad (87)$$
$$f'(x_i) = \tan(\theta_i) \quad (88)$$
$$f'(x_f) = 0 \quad (89)$$

Polynomial interpolation:

$$f(x) = a_0 + a_1 d + a_2 d^2 + a_3 d^3, d = \frac{x - x_i}{x_f - x_i}$$

$$f'(x) = d'(a_1 + 2a_2 d + 3a_3 d^2), d'(x) = \frac{1}{x_f - x_i}$$

Path planning: (2,0)-mobile robot

$$\begin{pmatrix} y_i \\ \tan(\theta_i) \\ y_f \\ 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & d' & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & d' & 2d' & 3d' \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix}$$

$$\begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1/d' & 0 & 0 \\ -3 & -2/d' & 3 & -1/d' \\ 2 & 1/d' & -2 & 1/d' \end{pmatrix} \begin{pmatrix} y_i \\ \tan(\theta_i) \\ y_f \\ 0 \end{pmatrix}$$

Path planning: (2,0)-mobile robot

$$\begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} y_i \\ \frac{\tan(\theta_i)}{d'} \\ 3(y_f - y_i) - 2\frac{\tan(\theta_i)}{d'} \\ -2(y_f - y_i) + \frac{\tan(\theta_i)}{d'} \end{pmatrix}$$

$$\alpha = \frac{\tan(\theta_i)}{d'} = \tan(\theta_i)(x_f - x_i)$$

$$d = \frac{x - x_i}{x_f - x_i}$$

$$y = f(x) = y_i + \alpha d + [3(y_f - y_i) - 2\alpha]\, d^2 + [-2(y_f - y_i) + \alpha]\, d^3.$$

Path planning: (2,0)-mobile robot

Needs to find a time parametrization of the flat outputs:

$$x = x^N(t)$$

satisfying the following conditions

$$x^N(t_i) = x_i \quad (90)$$
$$x^N(t_f) = x_f \quad (91)$$
$$\dot{x}^N(t_i) = 0 \quad (92)$$
$$\dot{x}^N(t_f) = 0 \quad (93)$$

Polynomial interpolation:

$$x^N(\tau) = b_0 + b_1\tau + b_2\tau^2 + b_3\tau^3, \tau = \frac{t - t_i}{t_f - t_i}$$

$$\dot{x}^N(\tau) = \dot{\tau}\left(b_1 + 2b_2\tau + 3b_3\tau^2\right), \dot{\tau} = \frac{1}{t_f - t_i}$$

Path planning: (2,0)-mobile robot

$$\begin{pmatrix} x_i \\ 0 \\ x_f \\ 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \dot{\tau} & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & \dot{\tau} & 2\dot{\tau} & 3\dot{\tau} \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

$$\begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1/\dot{\tau} & 0 & 0 \\ -3 & -2/\dot{\tau} & 3 & -1/\dot{\tau} \\ 2 & 1/\dot{\tau} & -2 & 1/\dot{\tau} \end{pmatrix} \begin{pmatrix} x_i \\ 0 \\ x_f \\ 0 \end{pmatrix} = \begin{pmatrix} x_i \\ 0 \\ 3(x_f - x_i) \\ -2(x_f - x_i) \end{pmatrix}$$

$$x^N(t) = x_i + (x_f - x_i)\tau^2(3 - 2\tau), \tau = \frac{t - t_i}{t_f - t_i}$$

Path planning: (2,0)-mobile robot

Open loop control:

$$v = \pm\sqrt{\dot{x}^2 + \dot{y}^2} \quad (94)$$

$$w = \frac{\dot{x}\ddot{y} - \dot{y}\ddot{x}}{\dot{x}^2 + \dot{y}^2} \quad (95)$$

$$y^N(t) = f(x^N(t)), \quad (96)$$

$$\dot{y}^N(t) = \dot{x}^N(t) f'(x^N(t)) \quad (97)$$

$$\ddot{y}^N(t) = \ddot{x}^N(t) f'(x^N(t)) + \dot{x}^{N2}(t) f''(x^N(t)) \quad (98)$$

$$v^N(t) = \dot{x}^N(t)\sqrt{1 + f'^2(x^N(t))} \quad (99)$$

$$w^N(t) = \frac{f''(x^N(t))}{1 + f'^2(x^N(t))} \quad (100)$$

Path planning: (2,0)-mobile robot

Path planning: (2,0)-mobile robot

Path planning: (2,0)-mobile robot

Path planning: (2,0)-mobile robot

Path planning: (2,0)-mobile robot

Path planning: (2,0)-mobile robot

Table of Contents

On the way to integration

Philosophy integrated layer :
"Strategic layer" (goal planning) + "Tactical layer" (guidance, navigation) + "Reflexive layer" (obstacle avoidance)

Solution

☞ Generate and execute a (sub)-optimal path planning which satisfy:

- geometric formation and communications constraints,
- obstacle avoidance constraints,
- given boundary conditions,
- other constraints: time constraints (rescue missions), energy constraints (batteries duration, ...)

On the way to integration

Philosophy integrated layer :
"Strategic layer" (goal planning) + "Tactical layer" (guidance, navigation) + "Reflexive layer" (obstacle avoidance)

Solution

☞ Generate and execute a (sub)-optimal path planning which satisfy:

- geometric formation and communications constraints,
- obstacle avoidance constraints,
- given boundary conditions,
- other constraints: time constraints (rescue missions), energy constraints (batteries duration, . . .)

On the way to integration

Philosophy integrated layer :
"Strategic layer" (goal planning) + "Tactical layer" (guidance, navigation) + "Reflexive layer" (obstacle avoidance)

Solution

☞ Generate and execute a (sub)-optimal path planning which satisfy:

- geometric formation and communications constraints,
- obstacle avoidance constraints,
- given boundary conditions,
- other constraints: time constraints (rescue missions), energy constraints (batteries duration, . . .)

On the way to integration

Philosophy integrated layer :
"Strategic layer" (goal planning) + "Tactical layer" (guidance, navigation) + "Reflexive layer" (obstacle avoidance)

Solution

☞ Generate and execute a (sub)-optimal path planning which satisfy:

- geometric formation and communications constraints,
- obstacle avoidance constraints,
- given boundary conditions,
- other constraints: time constraints (rescue missions), energy constraints (batteries duration, ...)

On the way to integration

Philosophy integrated layer :
"Strategic layer" (goal planning) + "Tactical layer" (guidance, navigation) + "Reflexive layer" (obstacle avoidance)

Solution

☞ Generate and execute a (sub)-optimal path planning which satisfy:

- geometric formation and communications constraints,
- obstacle avoidance constraints,
- given boundary conditions,
- other constraints: time constraints (rescue missions), energy constraints (batteries duration, ...)

Sub-graph path planning and constraints propagation

In both cases with or without leaders the overall group will be divided into small sub-groups.

- ☞ with leader recursive tree (look at soon and father nodes) + constraints propagation
- ☞ without leader associate to each nodes a weight, the bigger means that this node has more informations on its neighbors, ...

Sub-graph path planning and constraints propagation

In both cases with or without leaders the overall group will be divided into small sub-groups.

- ☞ with leader recursive tree (look at soon and father nodes) + constraints propagation
- ☞ without leader associate to each nodes a weight, the bigger means that this node has more informations on its neighbors, ...

Sub-graph path planning and constraints propagation

In both cases with or without leaders the overall group will be divided into small sub-groups.

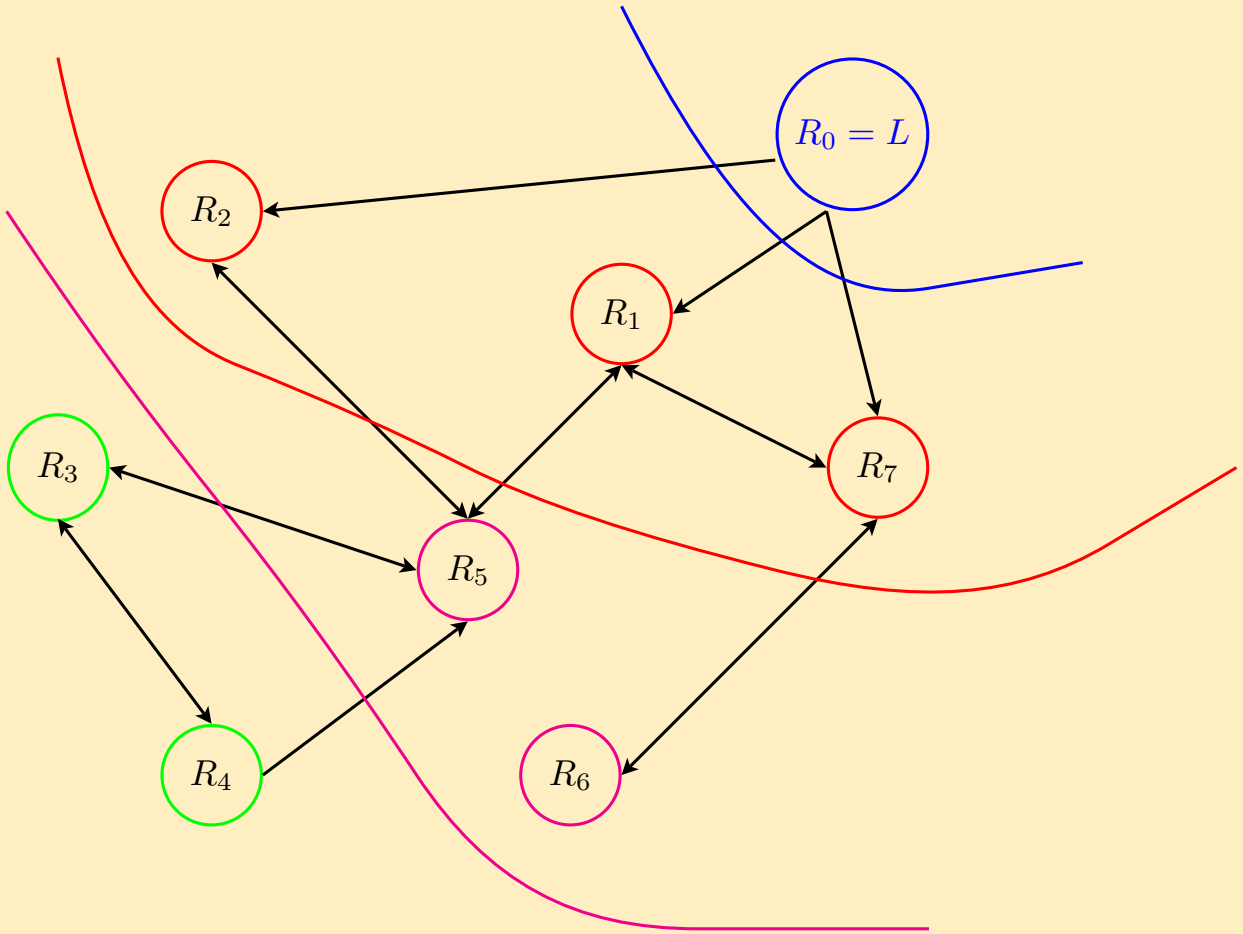
- with leader recursive tree (look at soon and father nodes) + constraints propagation
- without leader associate to each nodes a weight, the bigger means that this node has more informations on its neighbors, ...

Sub-graph path planning and constraints propagation

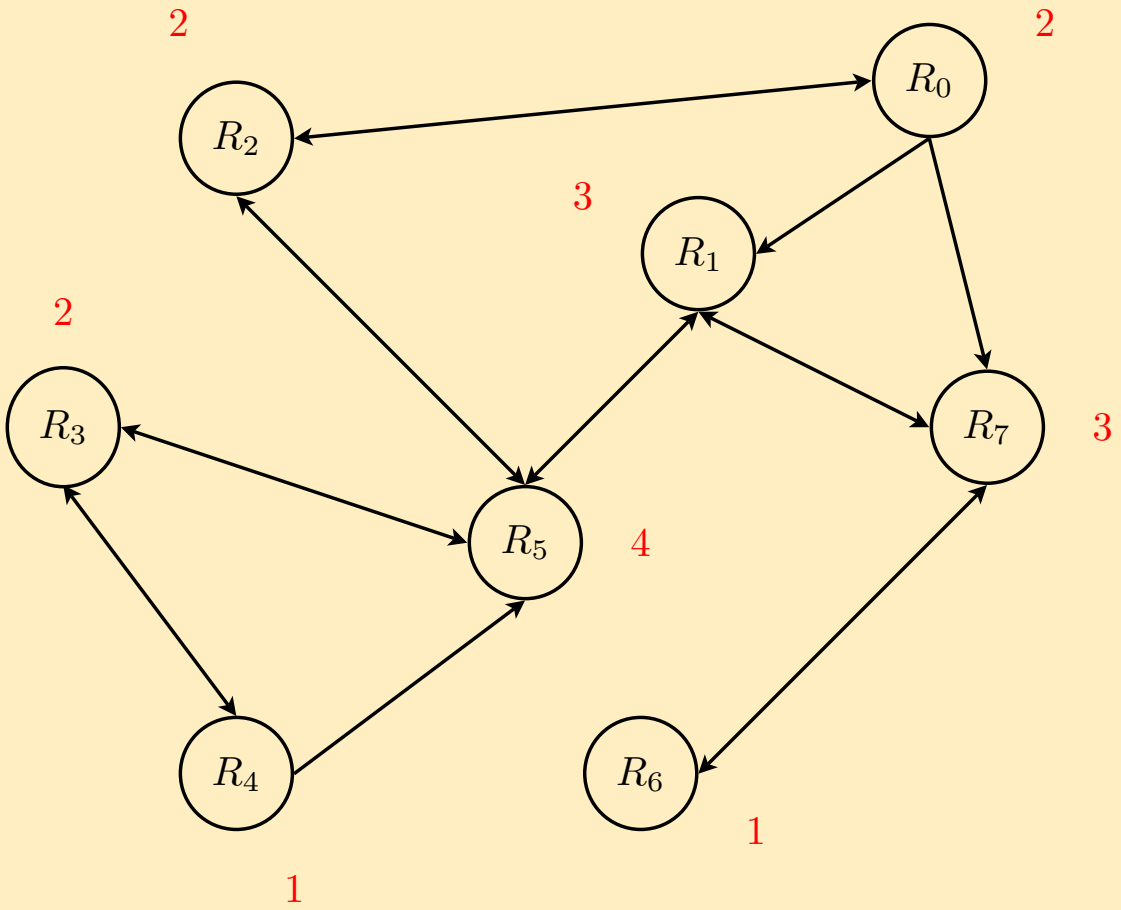
In both cases with or without leaders the overall group will be divided into small sub-groups.

- ☞ with leader recursive tree (look at soon and father nodes) + constraints propagation
- ☞ without leader associate to each nodes a weight, the bigger means that this node has more informations on its neighbors, ...

Sub-graph path planning and constraints propagation



Sub-graph path planning and constraints propagation



Sub-graph path planning and constraints propagation

In order to obtain a feasible path for the i^{th} robot (knowing its position and the ones of its neighbors which are feeding the i^{th} robot with some informations):
☞use the i^{th} row of $\mathcal{G}^c$ or $\mathcal{G}^f$.

General constraints

We would like to include in the path planning the following constraints :

1. some constraints due to physics (energy limitation, maximal velocity and acceleration of the robots)
2. obstacle avoidance,
3. collision avoidance with the robots and other mobile objects,
4. distances between robots (communications),
5. geometry of the formation
6. ...

Somme settings

Group of N_i[a] mobile robots related to the i^{th} mobile robot evolving in a partially known space with some obstacles (N_o obstacles).

- each obstacle O_m ($m \in \{1, \ldots, N_o\}$) is covered by a disc centered at (x_m^o, y_m^o) with radius r_m^o (if complex geometry use a covering of discs).
- I_N the set $\{1, \ldots, N\}$,
- the $n^{\text{th}} \in I_N$ mobile robot denoted by A_n and located at (x_n, y_n) occupies a space modeled by a disc of radius r_n centered at (x_n, y_n).
- robot A_n: X_n and U_n denotes respectively the state variables and the control variables.

[a] Index i (dropped sometimes) will refer to some properties or known objects linked to the i^{th} mobile robot. $N = N_i$!

Path planning for networked mobile robots

☞ For our group we shall find the trajectories and the control inputs which minimize the functional

$$J = \int_{t_0}^{t_f} C(X_1, \ldots, X_N, U_1, \ldots, U_N) dt,$$

where t_0 is the fixed initial time (assumed to be equal to zero), t_f is the fixed or unknown [a] final time and C is the cost function. The trajectories must join the known terminal states $X_n(0)$ and $X_n(t_f)$ with $n \in I_N$ and satisfy the following constraints:

[a] if time optimal path planning !

Path planning for networked mobile robots

$\forall t \in [0, t_f], \forall (n, n') \in I_N \times I_N,\ n \neq n',$

$C1$ the control bounds:

$$\|U_n\| \leq U_n^{max},$$
$$S_n^1 = U_n^{max} - \|U_n\| \geq 0$$

$C2$ the collision avoidance between robot A_n and the N_o obstacles. Each obstacle O_m $(m \in I_{N_o})$ is assumed to be a disc of center located at (x_m^o, y_m^o) and of radius r_m^o. These constraints are:

$$S_{nm}^2 = d(A_n, O_m)^2 - (r_n + r_m^o)^2 \geq 0$$
$$d(A_n, O_m) = \sqrt{(x_n - x_m^o)^2 + (y_n - y_m^o)^2}$$

Path planning for networked mobile robots

$C3$ the collision avoidance between robots A_n and $A_{n'}$:

$$S^3_{nn'} = d(A_n, A_{n'})^2 - (r_n + r_{n'})^2 \geq 0$$

$$d(A_n, A_{n'}) = \sqrt{(x_n - x_{n'})^2 + (y_n - y_{n'})^2}$$

$C4$ the distance between each robot has to be bounded so that the distance of communication is satisfied:

$$S^4_{nn'} = d^2_{max} - d(A_n, A_{n'})^2 \geq 0.$$

Penalty

The constrained optimization problem is transformed into an equivalent unconstrained problem by the construction of the following penalized function Ω and weights μ:

$$J_p = \int_{t_0}^{t_f} C(X_1, \ldots, X_N, U_1, \ldots, U_N)dt + \int_{t_0}^{t_f} \sum_{n=1}^{N} \left\{\mu_n^1 \Omega\left(S_n^1\right)\right\} dt$$
$$+ \int_0^{t_f} \sum_{n=1}^{N} \sum_{m=1}^{N_o} \left\{\mu_{nm}^2 \Omega(S_{nm}^2)\right\} dt + \int_0^{t_f} \sum_{n=1}^{N} \sum_{\substack{n'=1 \\ n' \neq n}}^{N} \left\{\mu_{nn'}^3 \Omega\left(S_{nn'}^3\right)\right\} dt$$
$$+ \int_0^{t_f} \sum_{n=1}^{N} \sum_{\substack{n'=1 \\ n' \neq n}}^{N} \left\{\mu_{nn'}^4 \Omega(S_{nn'}^4)\right\} dt. \qquad (101)$$

Penalty

The penalty function Ω and the weight μ penalize the cost J when the constraints are violated. The choice of this function determines the convergence of the proposed algorithm. Here, the chosen penalty may be:

$$\Omega : M \mapsto \begin{cases} 0 & \text{if } M \geq 0 \\ M^2 & \text{otherwise.} \end{cases}$$

Flatness is back

- ☞ Many physical systems are flat [**?**],
- ☞ (82) and (7) are flat: $z_n = [z_{1n}, z_{2n}]^T \in \mathbb{R}^2$ are the flat outputs.
- ☞ Let $Z = (z_1, \ldots, z_N)$ be the flat outputs of the global system made of the N robots.

Flatness is back

The problem of finding curves that take the system from the initial to the final condition is reduced to find sufficiently smooth curves that satisfies terminal constraints. Once the trajectory constraints are mapped into the flat output space, optimal trajectories are planned in this space. Therefore, the flatness property enables to eliminate the dynamic constraints.

☞ (101) can be expressed in function of the flat outputs and a finite number r of their time derivatives:

$$J_p = \int_0^{t_f} L(Z(t), \dot{Z}(t), \ldots, Z^{(r)}(t))dt.$$

On the way to NPP

Thus, the optimal control problem is:

$$\min_{Z} \int_0^{t_f} L(Z(t), \dot{Z}(t), \ldots, Z^{(r)}(t))dt \tag{102}$$

subject to terminal constraints: $\forall n \in I_N$: $X_n(0)$ $X_n(t_f)$ given. In order to numerically solve (102), the flat outputs are parameterized by using a basis of functions. That is to say, $\forall i \in \{1, 2\}$, $\forall n \in I_N$,

$$z_{in}(t) = \sum_{j=1}^{P} a_{in,j} h_j(t),$$

where $a_{in,j} \in \mathbb{R}$ and P is the dimension of basis functions $H = \{h_1, \ldots, h_P\}$.

On the way to NPP

Problem (102) becomes:

$$\min_{a_{in,j}} \int_0^{t_f} L(Z(t), \dot{Z}(t), \ldots, Z^{(r)}(t))dt. \tag{103}$$

Then, the time domain is truncated into smaller intervals by quadratic laws. Let N_{sample} be the number of sampled time, problem (103) is approximated by the nonlinear programming:

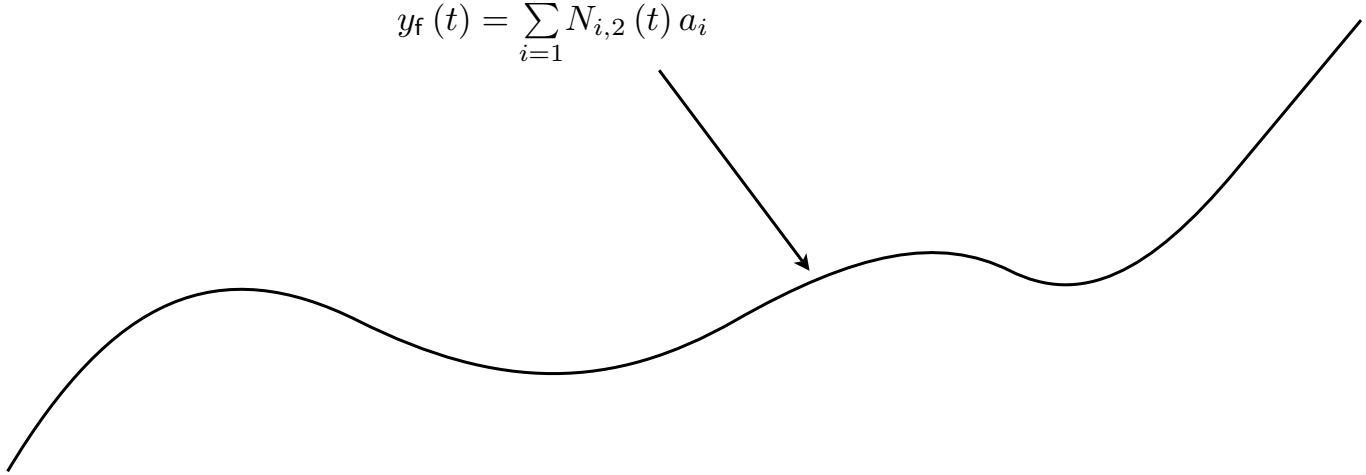
$$\min_{a_{in,j}} \sum_{k=1}^{N_{sample}} \sigma_k L(Z(t_k), \dot{Z}(t_k), \ldots, Z^{(r)}(t_k)) \tag{104}$$

subject to boundary constraints, where the weight σ_k and the time t_k ($\forall k \in I_{N_{sample}}$, $0 \leq t_k \leq t_f$) are chosen by different methods (trapezoidal, rectangle, Gauss-Legendre, ...).

Functions basis H: first used in Milam et al. 2001 [?]

☞ The function basis H governs the convergence speed of the algorithm: B-spline functions (numerically feasible).
Let $\sigma = \{t_0 < t_1 < ... < t_P = t_f\}$ be a subdivision of $[t_0, t_f]$ where P is a non zero fixed integer. Using σ let the knot vector $Q = \{q_1, \ldots, q_T\}$ be a set of T non-decreasing numbers called knots with $q_1 = t_0 = 0$ and $q_T = t_f$. To define B-spline basis functions, we need to introduce degree d of these basis functions.

$$y_f(t) = \sum_{i=1}^{T} N_{i,2}(t)\, a_i$$



$$Q = \{q_1 = \ldots = q_k = t_0 \quad q_{k+1} = t_1 \qquad\qquad q_{k+p} = \ldots = q_T = t_P = t_f.\}$$

$$\sigma = \{t_0 \qquad t_1 \qquad \cdots \qquad \cdots \qquad t_{P-1} \qquad \cdots \qquad t_P = t_f\}$$

Definition

The i^{th} B-spline basis function of degree $d = T - P - 1$, denoted by $N_{i,d}(t)$, $\forall i \in I_P$, is recursively defined as follows [**?**]:

$$N_{i,0}(t) = \begin{cases} 1 \text{ if } q_i \leq t < q_{i+1} \\ 0 \text{ otherwise} \end{cases}$$

$$N_{i,d}(t) = \frac{t - q_i}{q_{i+d} - q_i} N_{i,d-1}(t) + \frac{q_{i+d+1} - t}{q_{i+d+1} - q_{i+1}} N_{i+1,d-1}(t)$$

B-splines properties

B-splines properties are [**?**]:

1. compact support: $N_{i,d}$ are zeroing outside $[q_i, q_{i+d+1}] \Rightarrow$ impact near the considered point (good for numerical implementation). Indeed the gradient descent method for (104) leads to a sparse Hessian matrix (most of entries are zeros).
2. $N_{i,d}(t)$is a polynomial function in the variable t with degree d.
3. $N_{i,d}(t) \geq 0, \ \forall d, i, t.$
4. Easily differentiable with a strong smoothness degree : B-splines are $C^{s_i - 1}$. The $j^{th}-$derivative can be obtained from

$$N_{i,d}^{(j)}(t) = \frac{d-1}{d-i-j}\left[\frac{(t-q_i)}{(q_{i+d}-q_i)}N_{i,d-1}^{(j)}(t) + \frac{q_{i+d+1}-t}{q_{i+d+1}-q_{i+1}}N_{i+1,d-1}^{(j)}(t)\right] \quad (105)$$

with $j = 0, ..., k-1$.

STEP 1

Ce qui conduit à une procédure d'optimisation directe décomposée en deux étapes:

- ⋆ initialisation : planification d'une trajectoire initiale pour le centre de masse de la flottille en ne gardant que les obstacles de tailles trop importantes pour maintenir les distances inter-robots : ces obstacles doivent être contournés par la flottille. Translation de la trajectoire obtenue pour obtenir des trajectoires initiales pour chacun des robots respectant les conditions initiales et finales $(x_{n,\text{initial}}, y_{n,\text{initial}})$.

- ⋆ Calcul : paramétrisation de sorties plates de chaque robot par

des B-splines de degrés $d = 2$:

$$x_n(t_k) = x_{n,\text{initial}}(t_k) + \sum_{j=1}^{P} a_{1n,j} B_{j,2}(t_k)$$

$$y_n(t_k) = y_{n,\text{initial}}(t_k) + \sum_{j=1}^{P} a_{2n,j} B_{j,2}(t_k)$$

avec $t_k \in [t_0, t_f]$, $k \in I_{N_{\text{échantillons}}}$. Obtention des coefficients $a_{1n,j}$ et $a_{2n,j}$ assurant (104).:

- détermination du gradient de (104) divisé en cinq contributions: coût initial (C), et les quatre contraintes C1–C4.
- itération de l'algorithme de descente du gradient jusqu'à obtention de trajectoire qui satisfassent les contraintes C1–C4..
- en utilisant la platitude on en déduit le contrôle nominal en boucle ouverte pour chaque robot.

☞ Algorithm NP + B-spline + gradient descent method (follows the negative gradient of function (104)).
☞ Flat outputs are $z_n = (x_n, y_n) \Rightarrow$ find the optimal flat outputs.

Two steps

Two main phases:

- ⋆ initialization: motion planning for the gravity center of the formation.
- ⋆ computation: design of the flat outputs for each robot in the full map.

STEP 1 zoom of initialization procedure

In order to initialize our algorithm, the procedure is to find:

- ⋆ the terminal configurations of the formation gravity center (x_g, y_g, θ_g).
- ⋆ a final time guess and its associated trajectory (the curve that satisfies the terminal constraints with the appropriate initial and final orientation).
- ⋆ obstacles set O^1 for which the strategy applied is a bypass. The decision criterion depends on the size of the obstacle and distance d_{max}.
- ⋆ the optimal trajectory for the gravity center by gradient descent when the O^1 obstacles are only considered. It satisfies the collision avoidance with set O^1 and the control bounds.
- ⋆ to deduce trajectories $x_{n,initial}$ and $y_{n,initial}$ for each robot.

STEP 2 zoom of computation

For each robot $n \in I_N$, the terminal configurations are:

$$\begin{cases} x_n(0) = x_{n,0} \\ y_n(0) = y_{n,0} \\ \theta_n(0) = \theta_{n,0} \\ v_n(0) = 0 \end{cases} \quad \begin{cases} x_n(t_f) = x_{n,1} \\ y_n(t_f) = y_{n,1} \\ \theta_n(t_f) = \theta_{n,1} \\ v_n(t_f) = 0 \end{cases} .$$

The terms of finite dimensional B-spline curves of degree $r = 2$ are added:

$$x_n(t_k) = x_{n,initial}(t_k) + \sum_{j=1}^{P} a_{1n,j} N_{j,2}(t_k)$$

$$y_n(t_k) = y_{n,initial}(t_k) + \sum_{j=1}^{P} a_{2n,j} N_{j,2}(t_k)$$

STEP 2 zoom of computation

In order to find the robots trajectories that respect all the constraints (C1)-(C4), the computation phase is

- ⋆ to determine the gradient of function (104) which is divided into five contributions: the time minimization, the control bounds, the obstacle avoidance, the collision avoidance and the respect of distances between robots.
- ⋆ to apply iteratively the gradient descent until trajectories that satisfy the constraints are found.
- ⋆ to deduce the control inputs for each robot.

Animation

Video

Animation

Video

Animation

Video

Animation

Video