Feedback linearization and spline parametrizations

REPLAN team

March 18, 2025

Outline

Outline

Motivation - Integrator chain 1/2

Feedback linearization and/or flatness leads to the problem of controlling a system which is composed of several integrator chains decoupled from one another.

- How do we actually control a system? What is the relationship between input and output?
- Let's recall the state equation of a linear model:

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t) \end{cases}$$

- State representations are not unique. If the system is controllable we can bring it into a controllable canonical form[1]:

$$\begin{bmatrix} \dot{x}_1(t) \\ \vdots \\ \dot{x}_{n-1}(t) \\ \dot{x}_n(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \\ \gamma_1 & \gamma_2 & \dots & \gamma_n \end{bmatrix} \begin{bmatrix} x_1(t) \\ \vdots \\ x_{n-1}(t) \\ x_n(t) \end{bmatrix} + \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} u(t)$$

- We have a chain of integrations:

$$\dot{x}_1 = x_2, \quad \dot{x}_2 = x_3, \quad \dots \quad \dot{x}_{n-1} = x_n, \quad \dot{x}_n = \gamma_1 x_1 + \dots \gamma_n x_n + u$$

[1] The representation is valid for the particular case with one input. It becomes more complicated for the general case.

Motivation - Integrator chain 2/2

- The system's output is a combination of states; for simplicity, we choose C s.t. $y(t) = x_1(t)$
- Starting from the chain of integrations:

$$\dot{x}_1 = x_2, \quad \dot{x}_2 = x_3, \quad \dots \quad \dot{x}_{n-1} = x_n, \quad \dot{x}_n = \gamma_1 x_1 + \dots \gamma_n x_n + u$$

- We observe that we can reformulate the input u corresponding to a desired output y(t) as:

$$u(t) = y^{(n)}(t) - \gamma_1 y(t) - \dots - \gamma_n y^{(n-1)}(t)$$

- Working method for motion planning[2]:
  1. build a reference $\bar{y}(t)$ and its derivatives up to order n−1
  2. build the associated reference input $\bar{u}(t)$
  3. attach a "trajectory tracking" mechanism to penalize deviations: $u(t)$ depends on $\bar{u}(t)$ and the tracking error $y(t) - \bar{y}(t)$
- How do we generalize for the nonlinear case? - E.g. Feedback linearization, backstepping technique, flatness etc.

[2] but also in general with some small adjustments

Recall the classical industrial controllers – PID family

Proportional

$$u(t) = K_p e(t)$$

Proportional - Integral

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau$$

Proportional - Integral - Derivative

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de}{dt}$$

with K_p, K_i, K_d tuning parameters.

Example I: Double integrator

Let us consider a second-order system described by the differential equation

$$\ddot{y} = u$$

- Give a state representation for this system. The chosen state vector is given by $x = (y, \dot{y})$.

$$\dot{x}_1 = x_2, \quad \dot{x}_2 = u$$

- Choosing a setpoint $w(t)$ (a known function, e.g. a polynomial) control this system using a PD controller of the type:

$$u = k_1 (w - y) + k_2 (\dot{w} - \dot{y}) + \ddot{w}$$

To do & discuss

Give error dynamics for $e = w - y$ and compute k_1 and k_2, allowing to have an error e which converges towards 0 in an exponential manner.

Idea of feedback linearization

- consider the non-linear system described by:

$$\begin{cases} \dot{x} &= f(x,u) \\ y &= h(x), \end{cases} \Rightarrow \begin{cases} \dot{x} &= f(x) + g(x)u \\ y &= h(x), \end{cases}$$

- $u \in \mathbb{R}^m$ - inputs and $y \in \mathbb{R}^m$ -outputs.
- no. inputs $=$ no. outputs $= m$

Idea

"Linearize" the system using a controller of the type $u = \xi(x, v)$, where v is the virtual input of same dimension m.

Caution!

We assume that the state is **completely accessible**. If this is not the case, we need to build an observer in a non-linear context, which might be a DIFFICULT operation.

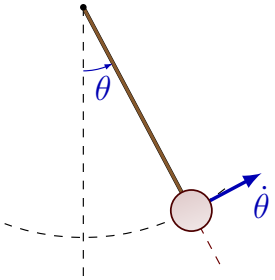
- since the state is assumed to be accessible, y should not be considered a sensor output, but rather as *setpoint variables*.

Example II: Pendulum

Consider the normalized model of the pendulum (i.e., $m = 1$, $g = 1$, $l = 1$).

- the input of this system is the torque u exerted on the pendulum.
- Objective: $x_1(t)$ to be equal to a time-varying setpoint $r(t)$
- Idea: a state feedback controller s. t. the error $e = r - x_1$ converges[a] towards 0
- differentiate y until the input u appears:

$$\begin{aligned} \dot{y} &= x_2 \\ \ddot{y} &= -\sin x_1 + u. \end{aligned}$$



- choose $u = \sin x_1 + v$, where v - *"intermediate/virtual input"* $\rightarrow \ddot{y} = v$
- choose the virtual input as: $v = (r - y) + 2(\dot{r} - \dot{y}) + \ddot{r}$
- the complete controller is expressed by: $u = \sin x_1 + (r - x_1) + 2(\dot{r} - x_2) + \ddot{r}$
- if $r(t) = sin(t)$ then $u = \sin x_1 + (\sin t - x_1) + 2(\cos t - x_2) - \sin t$

$$\begin{cases} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ -\sin x_1 + u \end{bmatrix} \\ y = x_1 \end{cases}$$

[a] as $e^{(-t)} \rightarrow$ *poles in* -1

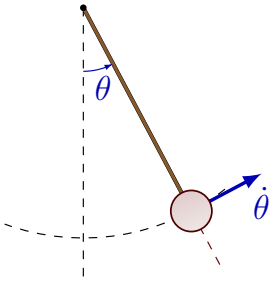
Example II: Pendulum

Consider the normalized model of the pendulum (i.e., $m = 1$, $g = 1$, $l = 1$).

- the input of this system is the torque u exerted on the pendulum.
- Objective: $x_1(t)$ to be equal to a time-varying setpoint $r(t)$
- Idea: a state feedback controller s. t. the error $e = r - x_1$ converges[a] towards 0
- differentiate y until the input u appears:

$$\begin{aligned} \dot{y} &= x_2 \\ \ddot{y} &= -\sin x_1 + u. \end{aligned}$$

- choose $u = \sin x_1 + v$, where v - *"intermediate/virtual input"* $\rightarrow \ddot{y} = v$
- choose the virtual input as: $v = (\dot{r} - \dot{y}) + 2(\dot{r} - \dot{y}) + \ddot{r}$
- the complete controller is expressed by: $u = \sin x_1 + (\dot{r} - x_1) + 2(\dot{r} - x_2) + \ddot{r}$
- if $r(t) = sin(t)$ then $u = \sin x_1 + (\sin t - x_1) + 2(\cos t - x_2) - \sin t$



No approximation arising from linearization has been performed.

[a] as $e^{(-t)} \rightarrow$ *poles in* -1

Example III: Dubins car with continuous velocity control action

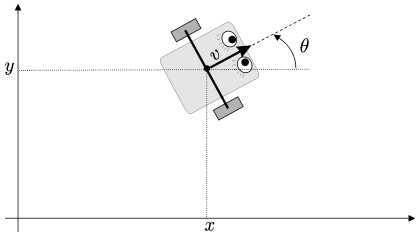
$$\begin{cases} \dot{x} &= \nu \cdot \cos\theta \\ \dot{y} &= \nu \cdot \sin\theta \\ \dot{\theta} &= u_1 \\ \dot{\nu} &= u_2 \end{cases}$$

with the state vector $\xi = \begin{bmatrix} x & y & \theta & \nu \end{bmatrix}^\top$

Objective Compute a controller that would allow us to describe a cycloid with the equation:

$$\begin{aligned} x_r(t) &= R\sin(f_1 t) + R\sin(f_2 t) \\ y_r(t) &= R\cos(f_1 t) + R\cos(f_2 t) \end{aligned}$$

with $R = 15$, $f_1 = 0.02$, $f_2 = 0.12$.



As in **Example II**, we can consider an input:

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} -\nu\sin(\theta) & \cos(\theta) \\ \nu\cos(\theta) & \sin(\theta) \end{bmatrix}^{-1} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

with the virtual input

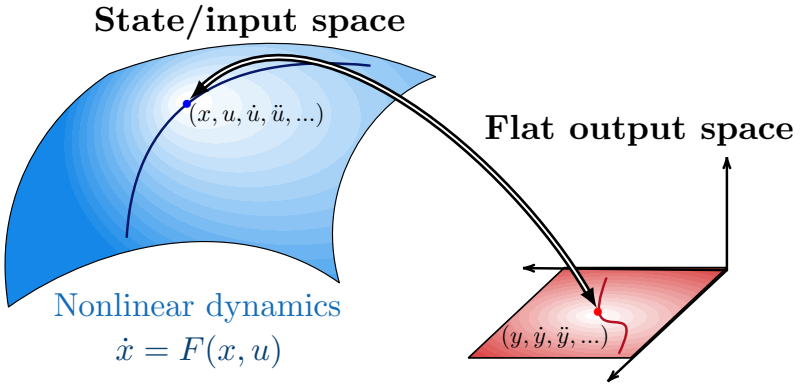
$$\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} x_r - x + 2(\dot{x}_r - \dot{x}) + \ddot{x}_r \\ y_r - y + 2(\dot{y}_r - \dot{y}) + \ddot{y}_r \end{bmatrix}$$

Outline

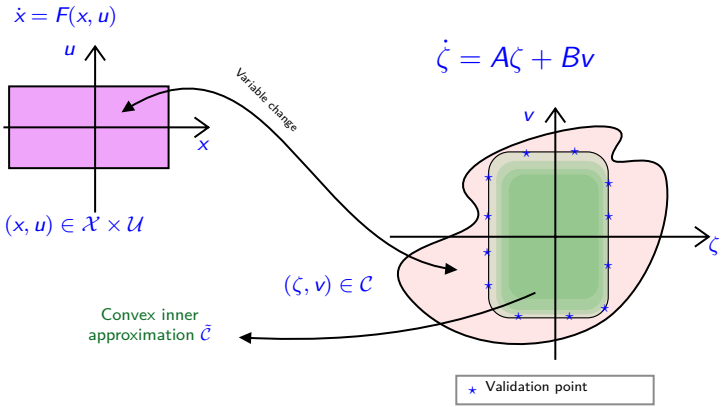
The flat representation of a nonlinear system

- The notion of flatness allows us to express the states / inputs of the system according to a flat output which, in turn, depends only on the state and derivatives of the input.
- For
$$\dot{x} = f(x, u), \quad x \in \mathbb{R}^n, u \in \mathbb{R}^m,$$
with flat output
$$y = h\left(x, u, \dot{u}, \ldots, u^{(r)}\right),$$
- system's trajectories (x, u) are written in function of the flat output:
$$x = \phi_x\left(y, \dot{y}, \ldots, y^{(q)}\right), \qquad u = \phi_u\left(y, \dot{y}, \ldots, y^{(q+1)}\right)$$
- Platitude has allowed us to "hide" the dynamic relationships between input and state: we can express everything according to a single quantity ($y(t)$ and its derivatives)

The flat representation of a nonlinear system (illustration)



There are problems (when considering constraints!)



Flat formulation for the Dubins car

- Recall the simplified Dubins car model:

$$\begin{cases} \dot{x} &= u_V \cos\theta \\ \dot{y} &= u_V \sin\theta \\ \dot{\theta} &= \frac{u_V}{L} \tan u_\theta \end{cases}$$

- We choose as flat output $z = \begin{bmatrix} z_1 & z_2 \end{bmatrix}^\top = \begin{bmatrix} x & y \end{bmatrix}^\top$
- We reformulate the remaining inputs (u_V, u_ϕ) and states (ϕ):

$$\begin{cases} u_V &= \sqrt{\dot{z}_1^2 + \dot{z}_2^2} \\ u_\phi &= \arctan\left(\frac{L\dot{\theta}}{u_V}\right) = \arctan\left(L\frac{\ddot{z}_2\dot{z}_1 - \dot{z}_2\ddot{z}_1}{\left(\dot{z}_1^2 + \dot{z}_2^2\right)^{\frac{3}{2}}}\right) \\ \theta &= \arctan\left(\frac{\dot{z}_2}{\dot{z}_1}\right) \end{cases}$$

Flat formulation for the UAV with 2 degrees of freedom

Consider a 2D 3-DOF model of a fixed-wing UAV in which the autopilot makes turns at a fixed altitude:

$$\begin{aligned}\dot{x} &= V_\mathrm{a}\cos\psi,\\ \dot{y} &= V_\mathrm{a}\sin\psi,\\ \dot{\psi} &= \frac{g\ \tan\phi}{V_\mathrm{a}}.\end{aligned}$$

$$\begin{aligned}\psi &= \arctan\left(\frac{\dot{z}_2}{\dot{z}_1}\right),\\ V_a &= \sqrt{\dot{z}_1^2+\dot{z}_2^2},\\ \phi &= \arctan\left(\frac{1}{g}\frac{\ddot{z}_2\dot{z}_1-\dot{z}_2\ddot{z}_1}{\sqrt{\dot{z}_1^2+\dot{z}_2^2}}\right).\end{aligned}$$

The flat output is taken after the position components of the state,

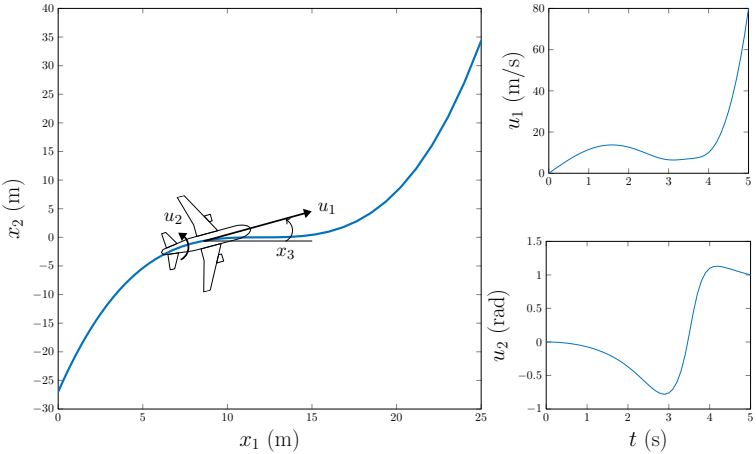
$$z = \begin{bmatrix} z_1 & z_2 \end{bmatrix}^\top = \begin{bmatrix} x & y \end{bmatrix}^\top.$$

- all states (and associated constraints / costs) are now expressed in terms of flat output and a finite number of its derivatives
- the problem is reduced to finding an output z that respects the restrictions

Example trajectory for the 2-DOF UAV

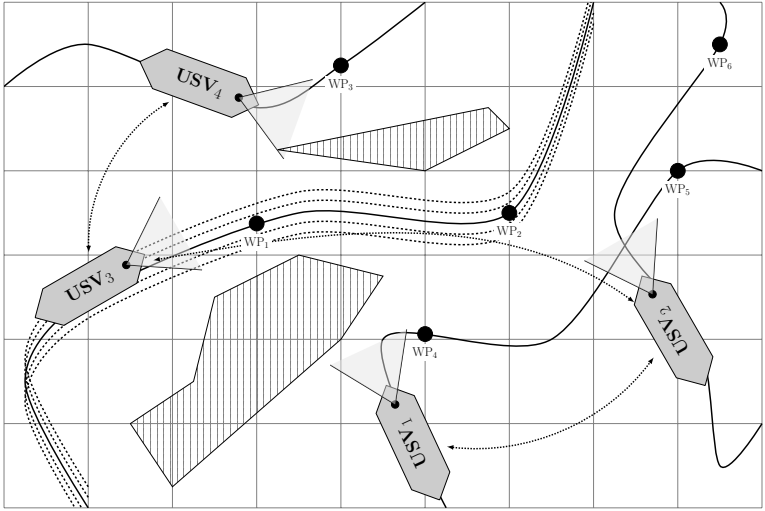
Consider a simple parameterization:

$$z_1(t) = t^2,\ z_2(t) = (0.25t^2 - 3)^3,$$



Outline

Constraints and costs – motion planning



Standard optimization problem

Let us consider a standard optimal control problem (OCP) of the form:

$$\min_u J(u) = \int_0^T L(x(t), u(t))\mathrm{d}t,$$
$$x(0) = x_0, x(T) = x_T,$$
$$u(0) = u_0, u(T) = u_T,$$
$$x(t) \in \mathcal{X}, u(t) \in \mathcal{U},$$
$$\dot{x}(t) = F(x(t), u(t)).$$

$$\min_{a_{ij}} J(a_{ij}) = \int_0^T L(\gamma(y, ..., y^{(s-1)}), \delta(y, \dots, y^{(s)}))\mathrm{d}t,$$
$$\gamma(y(0), ..., y^{(s-1)}(0)) = x_0, \gamma(y(T), ..., y^{(s-1)}(T)) = x_T,$$
$$\delta(y(0), ..., y^{(s)}(0)) = u_0, \delta(y(T), ..., y^{(s)}(T)) = u_T,$$
$$\gamma(y, ..., y^{(s-1)}) \in \mathcal{X}, \delta(y, \dots, y^{(s)})) \in \mathcal{U}$$

Generating the trajectory as an optimization problem

- The flat output $y(t)$ is parametrized thrugh a family of functions (a basis) $\{B_i(t)\}_{i=1\dots n}$:

$$z(t) = \sum_{i=1}^{n} P_i B_i(t)$$

- Polynomial, Bezier, B-spline, NURBS, etc. functions can be used as families of basis functions.

- A typical motion planning optimization problem (minimizes energy, respects dynamics, goes through intermediate points, avoids collisions, respects speed limits):

$$\min_{z(t)} \int_{t_i}^{t_f} \|\dot{z}(t)\|^2 dt \qquad \leftarrow \text{cost} \rightarrow \qquad \min_{P_i} \int_{t_i}^{t_f} \left\| P_i \dot{B}_i(t) \right\| dt$$

$$\text{s.t.} z(\tau_j) = w_j, \forall j \qquad \leftarrow \text{passing way-points} \rightarrow s.t. \qquad \sum_{i=1}^{n} P_i B_i(\tau_j) = w_j, \forall j$$

$$z(t) \notin O_k, \forall k \qquad \leftarrow \text{obstacle avoidance} \rightarrow \qquad \sum_{i=1}^{n} P_i B_i(t) \notin O_k, \forall k$$

Definitions and construction of Bezier functions

- Bezier functions are given by formula

$$B_{i,n}(t) = \binom{n}{i}(1-t)^{n-i}t^i, \quad \forall t \in [0,1]$$

  The binomial term: $\binom{n}{i} = \frac{n!}{i!(n-i)!}$

- A Bezier curve is obtained by weighting the basis functions $B_{i,n}(t)$ with control points P_i:

$$\begin{aligned} z(t) &= \sum_{i=0}^{n} B_{i,n}(t)P_i = \sum_{i=0}^{n} \binom{n}{i}(1-t)^{n-i}t^i P_i \\ &= (1-t)^n P_0 + \binom{n}{1}(1-t)^{n-1}tP_1 + \cdots + \binom{n}{n-1}(1-t)t^{n-1}P_{n-1} + t^n P_n \end{aligned}$$

- The derivatives of the flat output can also be expressed according to the control points:

$$\dot{z}(t) = n\sum_{i=0}^{n-1} B_{i,n-1}(t)(P_{i+1} - P_i)$$

- The Bezier curve $z(t)$ is found in the interior of the control region defined by $\{P_i\}$

B-spline functions – definition

We consider o sequence of time instants which form a *knot vector*[3]:

$$\boldsymbol{\zeta} = \{\tau_1 \leq \tau_2 \leq \cdots \leq \tau_m\}.$$

For $d \leq m-2$, any non-empty sequence (with $\tau_\ell \neq \tau_{\ell+d+1}$) $\{\tau_\ell \leq \cdots \leq \tau_{\ell+d+1}\} \subset \boldsymbol{\zeta}$ leads recursively at the B-spline function ℓ of order d:

$$B_{\ell,d,\boldsymbol{\zeta}}(t) = \frac{t-\tau_\ell}{\tau_{\ell+d}-\tau_\ell} B_{\ell,d-1,\boldsymbol{\zeta}}(t) + \frac{\tau_{\ell+d+1}-t}{\tau_{\ell+d+1}-\tau_{\ell+1}} B_{\ell+1,d-1,\boldsymbol{\zeta}}(t),$$

$$B_{\ell,0,\boldsymbol{\zeta}}(t) = \begin{cases} 1, & t \in [\tau_\ell, \tau_{\ell+1}), \\ 0, & \text{altfel.} \end{cases}$$

Taking $n = m-d-1$ B-spline functions $\{B_{1,d,\boldsymbol{\zeta}}, \ldots, B_{n,d,\boldsymbol{\zeta}}\}$ with the control points $\{P_1, \ldots, P_n\} \subset \mathbb{R}^p$ leads to the B-spline curve

$$z(t) = \sum_{i=1}^{n} P_i B_{i,d,\boldsymbol{\zeta}}(t), \ \forall t \in [\tau_{d+1}, \tau_{n+1}].$$

[3] T. Lyche, C. Manni, and H. Speleers (2017). *B-Splines and Spline Approximation.*

B-spline functions – properties

- Local partition of unity, for any $\ell = d+1 \dots n$:

$$\sum_{i=\ell-d}^{\ell} B_{i,d,\boldsymbol{\zeta}}(t) = 1,\ \forall t \in [\tau_\ell, \tau_{\ell+1})$$

- Local convexity, for any $\ell = d+1 \dots n$:

$$z(t) = \sum_{i=\ell-d}^{\ell} P_i B_{i,d,\boldsymbol{\zeta}}(t), \forall t \in [\tau_\ell, \tau_{\ell+1})$$

The global equivalent:

$z(t) = \sum_{i=1}^{n} P_i B_{i,d,\boldsymbol{\zeta}}(t), \forall t \in [\tau_{d+1}, \tau_{n+1})$.

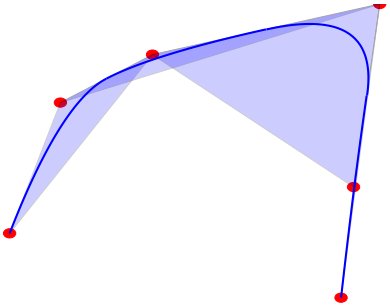
- Smoothness: $B_{i,d,\boldsymbol{\zeta}}(\tau_\ell) \in \mathcal{C}^{d-\mu_\ell}$ la $\tau_\ell \in \boldsymbol{\zeta}$ with multiplicity μ_ℓ și $\mathcal{C}^\infty$ otherwise.

- Unit partitioning, for any $\ell = d+1 \dots n$:

$$\sum_{i=\ell-d}^{\ell} B_{i,d,\boldsymbol{\zeta}}(t) = 1,\ \forall t \in [\tau_\ell, \tau_{\ell+1})$$

The global equivalent:

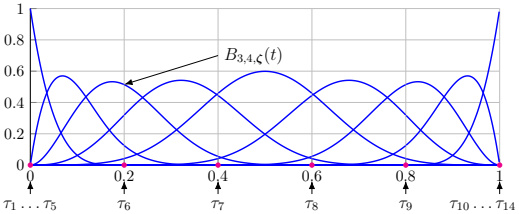
$\sum_{i=1}^{n} B_{i,d,\boldsymbol{\zeta}}(t) = 1,\ \forall t \in [\tau_{d+1}, \tau_{n+1})$.



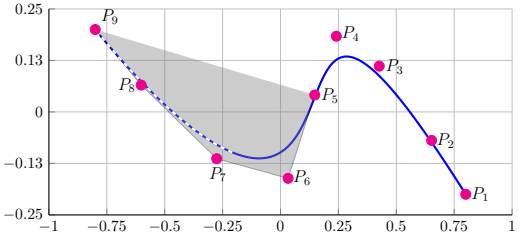
B-spline functions – example

- a knot-vector which partitions the interval $[0, 1]$ in $p = 5$ equidistant sub-intervals
- first and last point have multiplicity $d + 1 = 5$
- applying the definition for $d = 4$ leads to $n = 10$ B-spline basis functions
- adding the control points, we arrive at a B-Spline curve

B-spline basis functions



B-spline curve and its control points



Using B-splines to characterize constraints

- Define a region characterized by a finite collection of linear inequalities:

$$S = \{z \in \mathbb{R}^q : \ a_k^\top z \geq b_k, \ k = 1 \dots n_c\}$$

- Taking $z(t) = \sum_{i=1}^{n} P_i B_{i,d,\boldsymbol{\zeta}}(t)$, the k-th inequality from S, $[t_s, t_{s+1})$, becomes:

$$a_k^\top \sum_{\ell=s}^{s+d} P_\ell B_{\ell,d,\boldsymbol{\zeta}}(t) - b_k \geq 0,$$

$$a_k^\top \sum_{\ell=s}^{s+d} P_\ell B_{\ell,d,\boldsymbol{\zeta}}(t) - b_k \sum_{\ell=s}^{s+d} B_{\ell,d,\boldsymbol{\zeta}}(t) \geq 0,$$

$$\sum_{\ell=s}^{s+d} \left(a_k^\top P_\ell - b_k\right) B_{\ell,d,\boldsymbol{\zeta}}(t) \geq 0.$$

- The left-side of the inequality is itself a B-spline curve characterized by basis $\{B_{\ell,d,\boldsymbol{\zeta}}(t)\}$ and control points $\{a_k^\top P_\ell - b_k\}_{\ell=1\dots n_\nu}$:

$$\bar{z}_k(t) = \sum_{\ell=1}^{n} \left(a_k^\top P_\ell - b_k\right) B_{\ell,d,\boldsymbol{\zeta}}(t) = \sum_{\ell=1}^{n} \bar{P}_\ell B_{\ell,d,\boldsymbol{\zeta}}(t), \ \forall t \in [t_1, t_{n+1}).$$

The idea

Recall that the k-th inequality, $a_k^\top z \geq b_k$, is equivalent with:

$$\bar{z}_k(t) = \sum_{\ell=1}^{n} \left(a_k^\top P_\ell - b_k \right) B_{\ell,d,\boldsymbol{\zeta}}(t) = \sum_{\ell=1}^{n} \bar{P}_\ell B_{\ell,d,\boldsymbol{\zeta}}(t), \ \forall t \in [t_1, t_{n+1}).$$

The continuous-time validation of inclusions/exclusions becomes a positivity problem:

- set inclusion over the interval $[t_s, t_{s+1})$:

$$z(t) \in S \iff \bar{z}_k(t) \geq 0, \forall k$$

- set exclusion over the interval $[t_s, t_{s+1})$:

$$z(t) \notin S \iff \exists k \text{ s.t. } \bar{z}_k(t) \leq 0.$$

Note the locality property: $k(s)$ may change with each sub-interval! Relevant for the exclusion case where a constraint may not be always active.

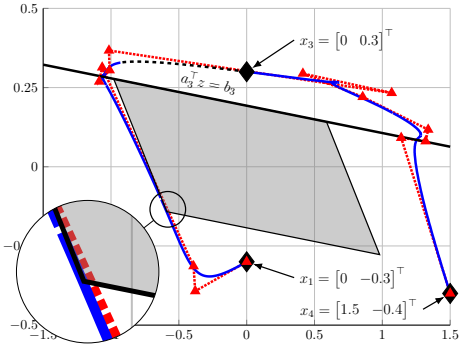
Illustrative example – I

Consider:

- a region S to be avoided
- way-points x_i to be passed-through
- minimize total path-length

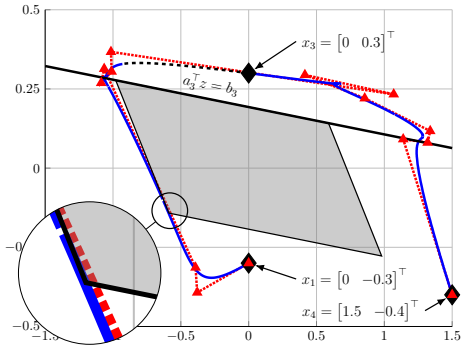
$$(P_1^* \dots P_{n_\nu}^*) = \arg \min_{P_1 \dots P_{n_\nu}} \int_{t_1}^{t_{n+1}} ||z'(t)||^2 dt$$

$$z(t) \notin S, \ \forall t \in [t_1, t_{n+1}]$$
$$z(t_1) = x_1, z(t_5) = x_3, z(t_{n+1}) = x_4.$$



Illustrative example – II

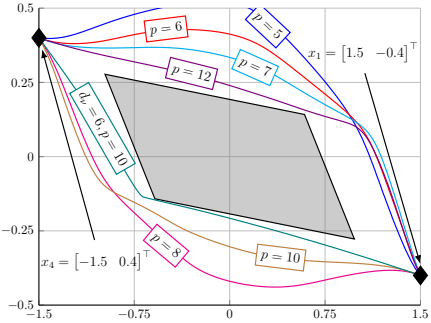
- the k-th inequality is active on the s-th interval iff $\alpha^{s,k} = 0$ and may change with each sub-interval
- during interval $[t_5, t_6)$ the curve lies opposite the obstacle wrt hyperplane $a_3^\top z = b_3$ (i.e., $\alpha^{s=5,k=3} = 0$)



s	1	2	3	4	5	6	7	8	9	10
$\alpha^{s,1}$	1	1	1	1	1	1	1	1	1	1
$\alpha^{s,2}$	0	0	1	1	1	1	1	1	1	1
$\alpha^{s,3}$	1	0	0	0	0	0	0	1	1	1
$\alpha^{s,4}$	1	1	1	1	1	1	1	0	0	0

Illustrative example – III

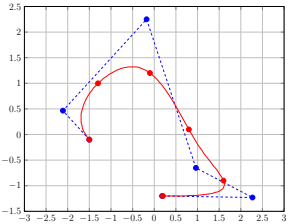
- increasing the number of sub-intervals p reduces the path length
- increasing the smoothness of the curve reduces the path length
- the influence of p appears to taper off when $p \gg n_c$

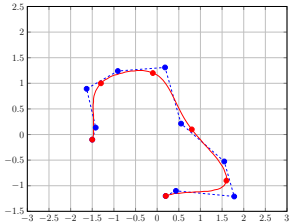


p	4	5	6	7	8
length	*	3.2889	3.2556	3.2420	3.2403
p	9	10	12	10 with $d_\nu = 7$	
length	3.2373	3.2131	3.2183	3.1588	

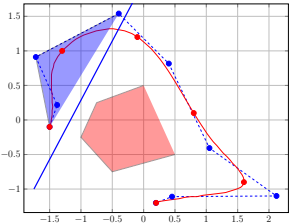
Illustrative example – IV

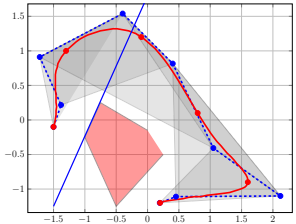
- We minimize the energy of the trajectory which has to pass through way-points:





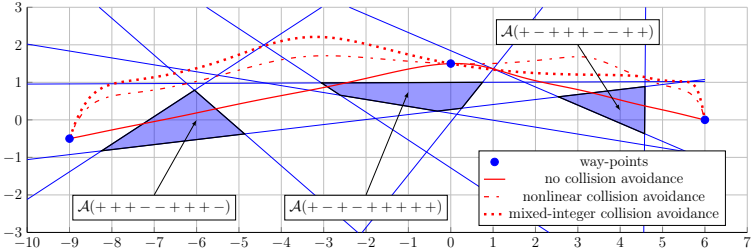
- We guarantee obstacle avoidance:



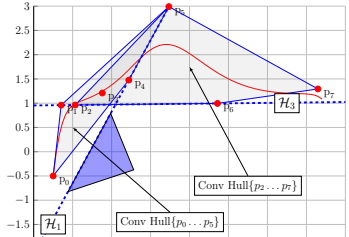


Illustrative example – V

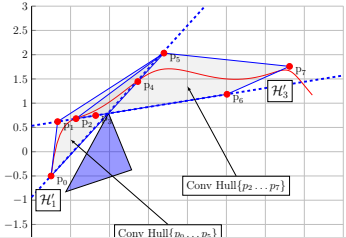
various collision avoidance methods



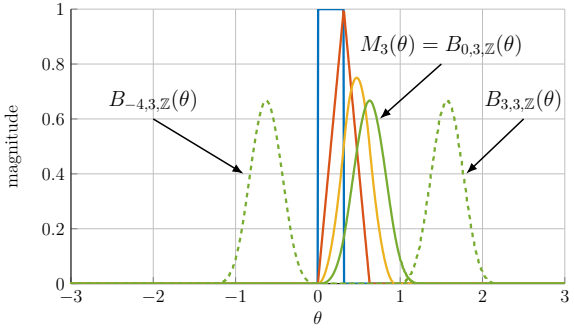
detail of the mixed-integer case



detail of the nonlinear case



Cardinal B-splines – definition



The cardinal splines Lyche, Manni, and Speleers 2017of degree $p \geq 1$:

$$B_{k,p,\mathbb{Z}}(t)$$

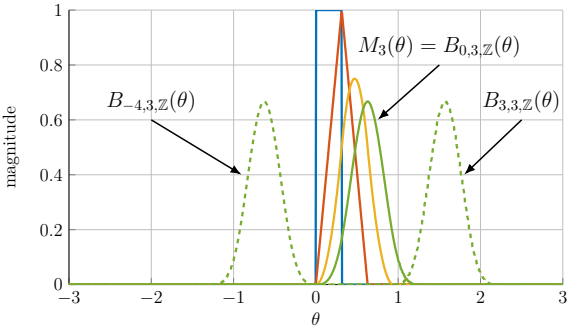
are obtained from:

$$M_0(\theta) = \begin{cases} 1, & \forall \theta \in [0,1), \\ 0, & \text{otherwise}, \end{cases}$$

$$M_p(\theta) = \frac{\theta}{p} M_{p-1}(\theta) + \frac{p+1-\theta}{p} M_{p-1}(\theta - 1),$$

$$B_{k,p,\mathbb{Z}}(\theta) = B_{0,p,\mathbb{Z}}(\theta - k) := M_p(\theta - k), \quad \forall k \in \mathbb{Z}.$$

Cardinal B-splines – properties



- Local support:

$$B_{k,p,\mathbb{Z}}(\theta) = 0, \forall \theta \notin [k, k+p+1);$$

- Partition of unity:

$$\sum_{k=m-p}^{m} B_{k,p,\mathbb{Z}}(\theta) = 1, \forall \theta \in [m, m+1);$$

- The Fourier transform is given by

$$\mathcal{F}\Big(M_p(\theta)\Big)(\omega) = \left(\frac{1 - e^{-j\omega}}{j\omega}\right)^{p+1}.$$

NURBS functions – I

The NURBS (Non-Uniform Rational B-Splines) functions of degree d are obtained by weighting the d-order B-splines with $\{\omega_1, \ldots, \omega_n\} \geq 0$:

$$R_{\ell,d,\boldsymbol{\zeta}}(t) = \frac{B_{\ell,d,\boldsymbol{\zeta}}(t)\omega_\ell}{\sum_{i=1}^{n} B_{i,d,\boldsymbol{\zeta}}(t)\omega_i}, \ \forall \ell = 1, \ldots, n.$$

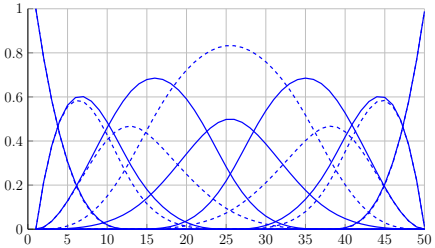
In addition to the B-splines properties they have also the property of homogeneous representation:

Homogeneous representation (projective invariance).

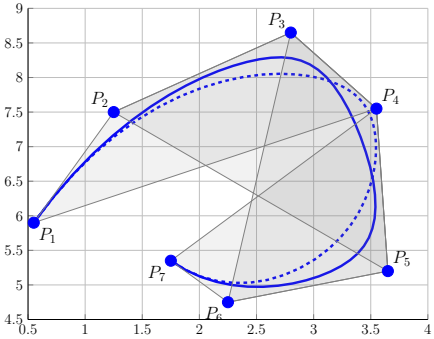
The rational curve $z(t) \in \mathbb{R}^p$ may be represented in homogeneous coordinates as a polynomial curve $z^\omega(t) \in \mathbb{R}^{p+1}$. That is, to each pair of control point and weight $(P_i, \omega_i) \in \mathbb{R}^p \times \mathbb{R}_+$ corresponds a control point $P_i^\omega = \begin{bmatrix} \omega_i P_i^\top & \omega_i \end{bmatrix}^\top$ which characterize the B-Spline curve $z^\omega(t) = \sum_{i=1}^{n} P_i^\omega B_{i,d,\boldsymbol{\zeta}}(t)$ and relation $z(t) = \begin{bmatrix} \mathbf{I}_p & \mathbf{0}_p \end{bmatrix} z^\omega(t) / \begin{bmatrix} \mathbf{0}_p^\top & 1 \end{bmatrix} z^\omega(t)$ holds.

NURBS functions – II

NURBS basis functions



NURBS curves and convex hull



Changing the weights we arrive at different curves which sit inside the same convex hull!

The knot insertion procedure

The NURBS curve describes a rational spline with polynomial components of degree d. Any NURBS basis with at least as many elements, defined over a properly chosen knot vector, will be able to describe the same curve. Choosing the new knot-vector is done through *knot insertion*, or its generalization, *knot refinement*.

Consider time instant σ with $1 \leq k < n+1$ taken such that $t_k < \sigma < t_{k+1}$ (or, equivalently stated, $[t_k, t_{k+1}) = [t_k, \sigma) \cup [\sigma, t_{k+1})$). Then:

$$z(t) = \sum_{i=1}^{n} P_i R_{i,d,\zeta}(t) = \sum_{i=1}^{n+1} Q_i \overline{R}_{i,d,\overline{\zeta}}(t), \ \forall t \in [\overline{\tau}_{d+1}, \overline{\tau}_{n+2}],$$

where,

$$Q_i = \bar{\alpha}_i P_i + (1 - \bar{\alpha}_i) P_{i-1}$$

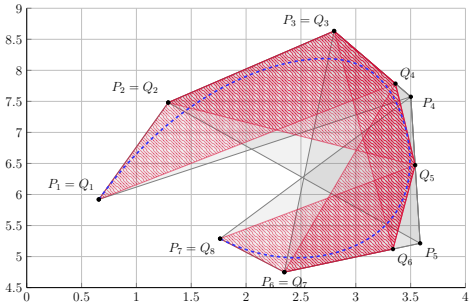
$$\bar{\alpha}_i = \frac{\alpha_i \omega_i}{\alpha_i \omega_i + (1 - \alpha_i)\omega_{i-1}}$$

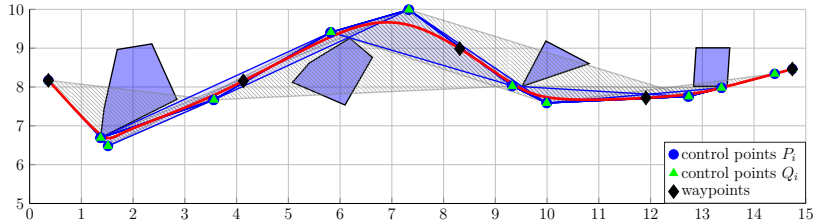
$$\alpha_i = \begin{cases} 1, & i \leq k - d \\ \frac{\sigma - \tau_i}{\tau_{i+d} - \tau_i}, & k - d + 1 \leq i \leq k \\ 0, & i \geq k + 1 \end{cases}$$

and with $\bar{\omega}_i = \alpha_i \omega_i + (1 - \alpha_i)\omega_{i-1}$.

Illustrative example

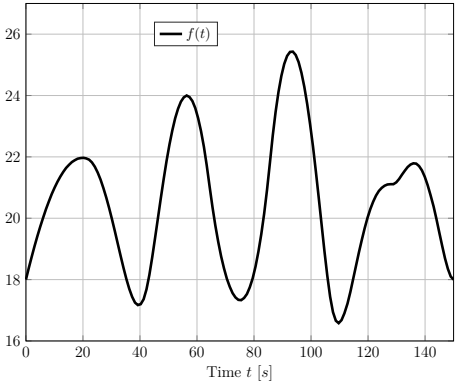
- use knot insertion/refinement ($[t_k, t_{k+1}) = [t_k, \sigma) \cup [\sigma, t_{k+1})$)
- the new control points depend linearly on the former ($Q_i = \alpha_i P_i + \alpha_{i-1} P_{i-1}$)
- the new convex hull union is contained in the former
- the number of variables remains constant, only the number of constraints increases





Schoenberg operator

How to approximate a bounded real-valued function f?

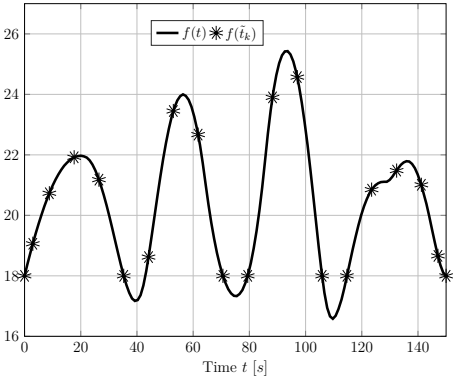


Schoenberg operator

How to approximate a bounded real-valued function f?

The Greville points are

$$\tilde{t}_k = \frac{\tilde{\tau}_{k+1} + \ldots + \tilde{\tau}_{k+\tilde{d}-1}}{\tilde{d}-1}.$$



Schoenberg operator

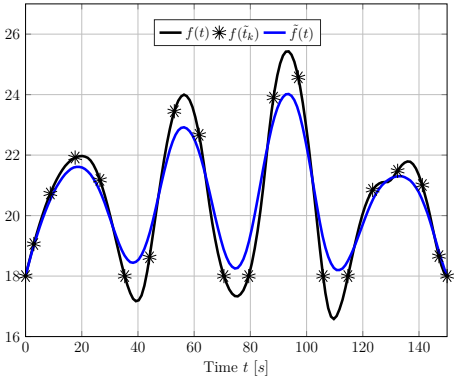
How to approximate a bounded real-valued function f?

The Schoenberg operator Beutel, Gonska, Kacsó, and Tachev 2002 approximates a function (call it $f(\cdot)$) by interpolating it as a weighted sum of B-splines basis functions, $\tilde{B}_{k,\tilde{d},\tilde{\xi}}(t)$, and the function's values in the Greville points, $\tilde{t}_k$:

$$\tilde{f}(t) = \sum_{k=0}^{\tilde{n}-1} f(\tilde{t}_k)\tilde{B}_{k,\tilde{d},\tilde{\xi}}(t),$$

where the Greville points are

$$\tilde{t}_k = \frac{\tilde{\tau}_{k+1} + \ldots + \tilde{\tau}_{k+\tilde{d}-1}}{\tilde{d}-1}.$$



Outline

Energy minimization of a quadcopter

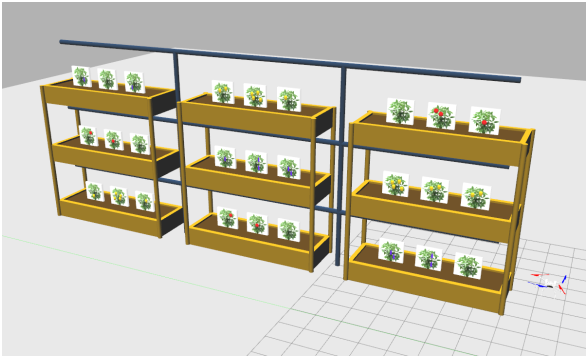


Figure: Arena of the ICUAS Competition 2024.

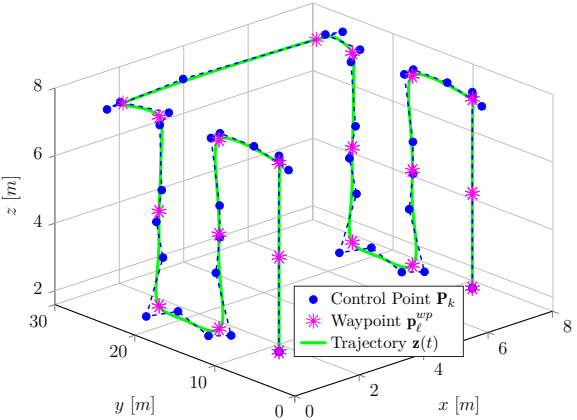


Figure: Non-aggressive B-spline trajectory.

Energy minimization of a quadcopter: Non-aggressive trajectory

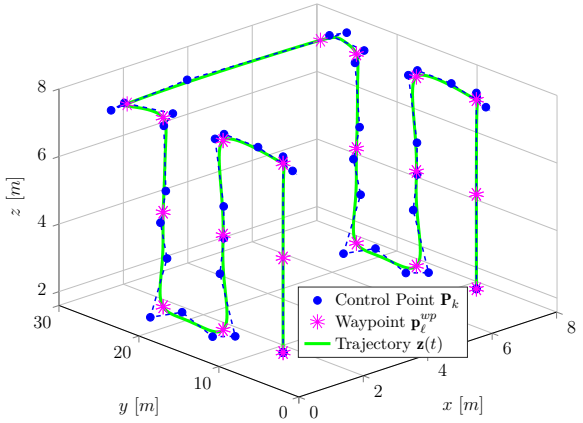


Figure: Non-aggressive B-spline trajectory.

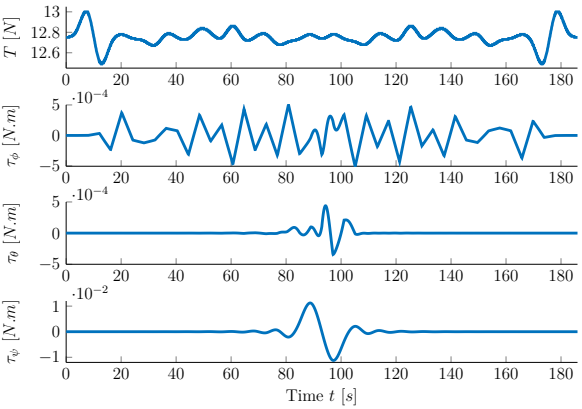


Figure: Evolution of thrust and torques.

[1] **V. Marguet**, F. Stoican, I. Prodan: *Energy-Efficient Trajectory Planning with B-Splines and the Schoenberg Quasi-Interpolant. The 63rd IEEE Conference on Decision and Control*, CDC'24, Milan, Italy, 16-19 December 2024.