Mașina Dubins. Generarea traiectoriilor în plan
curs 3 - opțional SPER

Florin Stoican

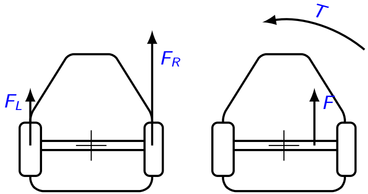
1 martie 2022

Cuprins

Cuprins

Model matematic – "robot diferențial"

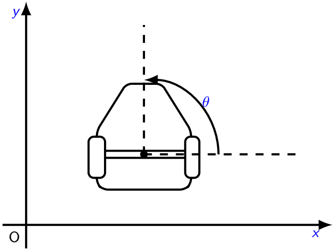
- Modelul matematic cel mai simplu
- Cele două roți se controlează independent



- Mișcarea este definită de poziție (x, y) în plan și orientare față de axa orizontală (θ)

$$\begin{cases} \dot{x} &= \frac{r}{2}(u_l + u_r)\cos\theta \\ \dot{y} &= \frac{r}{2}(u_l + u_r)\sin\theta \\ \dot{\phi} &= \frac{r}{L}(u_r - u_l) \end{cases} \Rightarrow \begin{cases} \dot{x} &= u_V\cos\theta \\ \dot{y} &= u_V\sin\theta \\ \dot{\phi} &= u_\phi \end{cases}$$

Mașina este condusă prin intrările u_V și u_ϕ.



Model matematic – "mașina simplă"

- Modelul matematic "robot diferențial" nu este suficient de realist
- Mai realist: "mașina simplă" ce nu poate aluneca lateral[1]
- Schimbarea orientării (θ) se face prin schimbarea direcției roților în tandem

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

Mașina este condusă prin intrările u_V și u_θ.

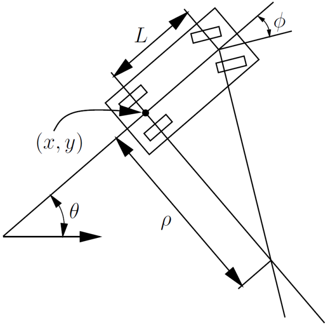


Fig. 13.1 din LaValle 2006

[1] LaValle, S. M. *Planning algorithms*. 2006

Variante ale modelului standard

- Mașina nu se poate întoarce "pe loc"; are un un unghi $\phi_{\max} \leq \pi/2$, sau, echivalent spus, o rază de curbură minimă $\rho_{\min} = L/\tan\phi_{\max}$

- Este natural să considerăm constrângeri pe intrări (u_V și u_ϕ):

  mașina simplă: $u_V \in [-1, 1]$ și $u_\phi \in (-\phi_{\max}, -\phi_{\max})$, îi corespunde o rază de curbură minimă $\rho_{\min} = L/\tan\theta_{\max}$

  mașina Reeds-Shepp: $u_V \in \{-1, 0, 1\}$, mașina funcționează în modurile "înapoi", "parcare" și "înainte"

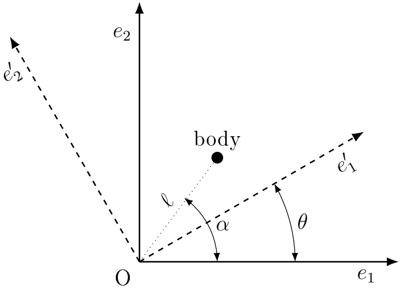
  mașina Dubins: $u_V \in \{0, 1\}$, mașina funcționează doar în modurile "parcare" și "înainte".

- Aceste modele matematice sunt considerate nonholonomice deoarece apar constrângeri diferențiale ce nu pot fi integrate, în cazul nostru:

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

  Echivalent spus, nu este posibilă reformularea acestei constrângeri într-o nouă formă, în care nu apar derivate.

Rotații în 2D



- Rotația este echivalentă cu înmulțirea cu o matrice de forma:

$$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

- Determinantul este $det(R) = 1 \Rightarrow$ transformarea nu modifică volumul corpului
- În plus, cei doi vectori ai lui R sunt ortogonali:

$$\cos\theta \times (-\sin\theta) + \sin\theta \times \cos\theta = 0$$

Traiectoria de timp minim pentru mașina Dubins

- Considerăm mașina Dubins simplificată:

$$\begin{cases} \dot{x} = & \cos\theta \\ \dot{y} = & \sin\theta \\ \dot{\theta} = & u \end{cases}$$

- În cel mai simplu caz, suntem interesați de trasarea unei traiectorii între două configurații arbitrare (configurație = combinație de poziție și orientare a vehiculului)

$$(x(t_i), y(t_i), \theta(t_i)) \qquad \longrightarrow \qquad (x(t_f), y(t_f), \theta(t_f))$$

- Se arată că orice traiectorie de timp minim urmează o combinație de pași de tipul:
  - rotație la maxim într-o direcție (sens trigonometric sau anti-trigonometric)
  - deplasare în linie dreaptă
  - rotație la maxim încă o data (sens trigonometric sau anti-trigonometric)
- Pentru orice pereche de configurații sunt doar 6 posibilități optime de trasare a traiectoriei Dubins:

$$\{LRL, RLR, LSL, LSR, RSL, RSR\}$$

unde, L – left, R – right și S – straight.

Traiectorie Dubins – ilustrații

- Fiecărei componente îi asociem o mărime (unghi arc sau lungime segment):

$$\{L_\alpha R_\beta L_\gamma, R_\alpha L_\beta R_\gamma, L_\alpha S_d L_\gamma, L_\alpha S_d R_\gamma, R_\alpha S_d L_\gamma, R_\alpha S_d R_\gamma\},$$

unde $\alpha, \gamma \in [0, 2\pi)$, $\beta \in (\pi, 2\pi)$ și $d \geq 0$.

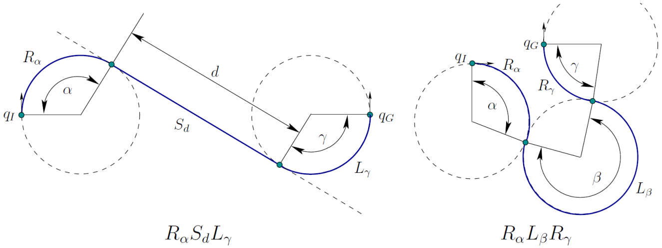


Fig. 15.4 din LaValle 2006

- Dintre toate posibilitățile trebuie selectată cea cu durata mai scurtă.

Traiectoriile Reeds-Shepp

- La mașina Reeds-Shepp, spre deosebire de mașina Dubins, se permite și mersul înapoi. Modelul simplificat este:

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

  unde $u_1 \in \{-1, 1\}$ și $u_2 \in [-\tan\phi_{\max}, \tan\phi_{\max}]$.

- Acum avem 48 de primitive posibile, compuse din până la 5 mișcări distincte
- Ilustrație traiectorie:

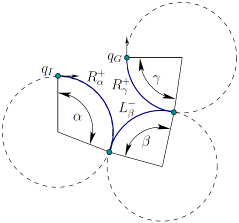


Fig. 15.9 din LaValle 2006

Cazuri similare

- Problema de atingere a unei configurații în timp minim se poate pune pentru diverse modele și cu constrângeri impuse acestora
- Pentru robotul diferențial (cel cu două roți ce se pot mișca independent), dacă penalizăm efortul de rotație în cost

$$\int_{t_i}^{t_f} \sqrt{\dot{x}(t)^2 + \dot{y}(t)^2} + |\dot{\theta}(t)| dt,$$

  obținem traiectoriile Balkom-Mason, compuse din cel mult 5 componente ce iau valori dintr-un set de 4 mișcări distincte $\Rightarrow$ 40 traiectorii distincte
- Traiectoriile Dubins se pot extinde pentru cazul 3D (dinamici UAV cu două grade de libertate)

Cuprins

Motivație și cazul liniar – I

- Cum controlăm de fapt un sistem? Care este legătura între intrare și ieșire?
- Să ne aducem aminte de ecuația de stare a unui model liniar:

$$\dot{x}(t) = Ax(t) + Bu(t)$$

- Reprezentările pe stare nu sunt unice. Dacă sistemul este controlabil putem să îl aducem într-o formă canonic controlabilă[2]:

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

- Avem un lanț de integrări:

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

[2] Reprezentarea este valabilă pentru cazul particular cu o intrare. Devine mai complicat pentru cazul general.

Motivație și cazul liniar – II

- Ieșirea sistemului este o combinație de stări, pentru simplitate alegem $y(t) = x_1(t)$
- Pornind de la lanțul de integrări:

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

- Observăm că putem reformula intrarea u ce corespunde unei ieșiri dorite $y(t)$ ca fiind

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

- Mod de lucru pentru planificarea mișcării:
  1. construim o referință $\bar{y}(t)$ și derivatele acesteia până la ordinul $n-1$
  2. construim intrarea de referință asociată $\bar{u}(t)$
  3. atașăm un mecanism de “trajectory tracking” pentru a penaliza abaterile: $u(t)$ depinde de $\bar{u}(t)$ și de eroarea de urmărire $y(t) - \bar{y}(t)$

- Cum generalizăm pentru cazul neliniar?

Reprezentarea plată a unui sistem neliniar

- Noțiunea de platitudine ne permite să exprimăm stările/intrările sistemului în funcție de o ieșire plată ce, la rândul său, depinde doar de stare și derivate ale intrării.

- Pentru

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

  cu ieșirea plată

$$y = h\left(x, u, \dot{u}, \ldots, u^{(r)}\right),$$

- traiectoriile sistemului (x, u) se scriu în funcție de ieșirea plată și derivatele sale:

$$x = \phi_x\left(y, \dot{y}, \ldots, y^{(q)}\right), \qquad u = \phi_u\left(y, \dot{y}, \ldots, y^{(q+1)}\right)$$

- Platitudinea ne-a permis să "ascundem" relațiile dinamice dintre intrare și stare: putem exprima totul în funcție de o singură mărime ($y(t)$ și derivatele sale)

Reformularea plată pentru mașina Dubins

- Dinamica simplificată pentru mașina Dubins:

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

- Alegem ca ieșire plată $z = \begin{bmatrix} z_1 & z_2 \end{bmatrix}^\top = \begin{bmatrix} x & y \end{bmatrix}^\top$
- Reformulare a intrărilor (u_V, u_ϕ) și a stărilor rămase (ϕ):

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

Alte implementări – UAV cu 2 grade de libertate

Considerăm un model 2D 3-DOF al unui UAV cu aripă fixă în care autopilotul execută întoarceri la altitudine fixă:

$$\dot{x}(t) = V_{\mathrm{a}}(t)\cos\psi(t),$$
$$\dot{y}(t) = V_{\mathrm{a}}(t)\sin\psi(t),$$
$$\dot{\psi}(t) = \frac{g\ \tan\phi(t)}{V_{\mathrm{a}}(t)}.$$

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

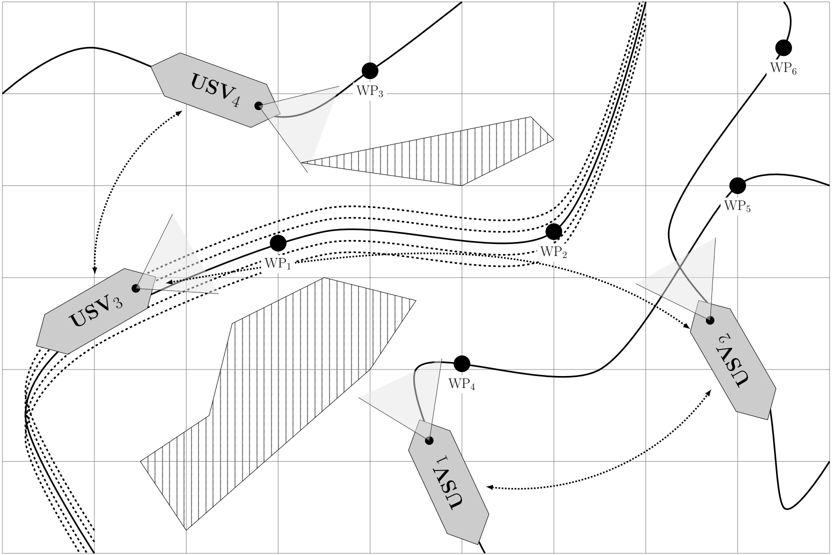
Ieșirea plată este luată după componentele de poziție ale stării,

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

- toate stările (și constrângeri/costuri asociate) sunt acum exprimate în funcție de ieșirea plată și un număr finit de derivate ale acesteia
- problema se reduce la găsirea unei ieșiri $z(t)$ ce respectă restricțiile

Cuprins

Constrângeri și costuri – planificarea mișcării



Generarea traiectoriei ca o problemă de optimizare

- Ieșirea plată $y(t)$ se parametrizează pe o familie de funcții $\{B_i(t)\}_{i=1\dots n}$:

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

- Ca familii de funcții bază se pot folosi funcții polinom, Bezier, B-spline, NURBS, etc.

- O problemă tipică de optimizare a planificării mișcării (minimizează energia, respectă dinamica, trece prin puncte intermediare, evită coliziuni, respectă limite de viteză):

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

$$\text{s.t.} z(\tau_j) = w_j, \forall j \qquad \leftarrow \text{trecere prin puncte} \rightarrow s.t. \qquad \sum_{i=1}^{n} P_i B_i(\tau_j) = w_j, \forall j$$

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

Definiție și construcție funcții Bezier

- Funcțiile Bezier sunt date de formula

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

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

- O curbă Bezier se obține ponderând funcțiile bază $B_{i,n}(t)$ cu puncte de control P_i:

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

- Derivatele ieșirii plate se pot și ele exprima în funcție de punctele de control:

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

- Curba Bezier $z(t)$ se găsește în interiorul regiunii de control definite de $\{P_i\}$

Exemplu utilizare – I

- O problemă tipică de optimizare a planificării mișcării (minimizează energia, respectă dinamica, trece prin puncte intermediare, evită coliziuni, respectă limite de viteză):

$$\min_{u(\cdot)} \int_{t_i}^{t_f} \|\dot{z}(t)\|^2 dt$$

s.t. dinamica se respectă,

$$z(t_j^c) = c_j,\ \forall j,$$
$$z(t) \notin \mathcal{O}_i,\ \forall i, \forall t \in [t_i, t_f]$$
$$\underline{V}^2 \leq \dot{z}_1^2(t) + \dot{z}_2^2(t) \leq \overline{V}^2.$$

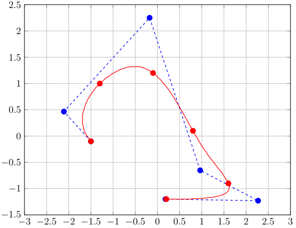
$$\min_{P_1 \dots P_n} \sum_{i=1}^{n-1} \sum_{j=1}^{n-1} \tilde{P}_i^\top \tilde{P}_j \int_{t_i}^{t_f} B_{i,d-1,\zeta}(t) B_{j,d-1,\zeta}(t) dt$$

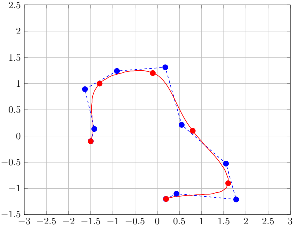
$$\text{s.t.} \sum_{i=1}^{n} B_{i,d,\zeta}(t_j^c) P_i = c_j,\ \forall j,$$
$$\left(\bigcup_{\ell=d+1}^{n} \operatorname{ConvHull}_{i=\ell-d}^{\ell}\{P_i\}\right) \cap \mathcal{O}_i = \emptyset,\ \forall i,$$
$$\left(\bigcup_{\ell=d}^{n-1} \operatorname{ConvHull}_{i=\ell-d+1}^{\ell}\{\tilde{P}_i\}\right) \cap \mathcal{C}\left(\underline{V}\right) = \emptyset,\ \tilde{P}_i \in \mathcal{C}\left(\overline{V}\right).$$

- Luând $z(t) = \sum_{\ell=1}^{n} P_\ell B_{\ell,d,\zeta}(t)$ avem:
  - o formulare suficientă față de forma inițială
  - o reformulare în funcție de punctele de control (ponderile asociate sistemului)
  - permite verificarea constrângerilor continue

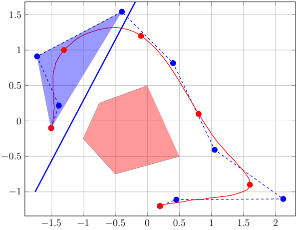
Exemplu utilizare – II

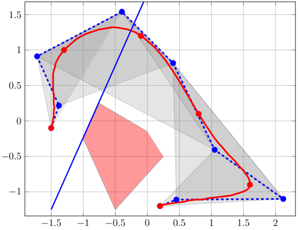
- Minimizăm energia traiectoriei forțând trecerea prin puncte intermediare:





- Garantăm ocolirea obstacolelor:





Variații - funcții B-spline (I)

Considerăm o secvență de instanțe de timp ce formează un *knot vector*[3]:

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

Pentru $d \leq m - 2$, fiecare secvență nenulă (cu $\tau_\ell \neq \tau_{\ell+d+1}$) $\{\tau_\ell \leq \cdots \leq \tau_{\ell+d+1}\} \subset \zeta$ conduce recursiv la funcția B-spline ℓ de ordin d:

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

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

Luând $n = m - d - 1$ funcții B-spline $\{B_{1,d,\zeta}, \ldots, B_{n,d,\zeta}\}$ cu punctele de control $\{P_1, \ldots, P_n\} \subset \mathbb{R}^p$ conduce la curba B-spline

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

[3] Lyche, T., C. Manni **and** H. Speleers. "Foundations of spline theory: B-splines, spline approximation, and hierarchical refinement". **in** *Splines and PDEs: From Approximation Theory to Numerical Linear Algebra*: 2018, pp. 1–76, 2018.

.

Variații - funcții B-spline (II)

- Cu suport local, pentru orice $\ell = 1 \dots n$:

$$B_{\ell,d,\zeta}(t) = 0, \forall t \notin [\tau_\ell, \tau_{\ell+d+1}]$$

- Convexitate locală, pentru orice $\ell = d+1 \dots n$:

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

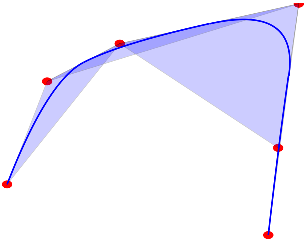
Echivalentul global: $z(t) = \sum_{i=1}^{n} P_i B_{i,d,\zeta}(t), \forall t \in [\tau_{d+1}, \tau_{n+1})$.

- Netezime: $B_{i,d,\zeta}(\tau_\ell) \in \mathcal{C}^{d-\mu_\ell}$ la $\tau_\ell \in \zeta$ cu multiplicitate μ_ℓ și $\mathcal{C}^\infty$ altfel.

- Partiționarea unității, pentru orice $\ell = d+1 \dots n$:

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

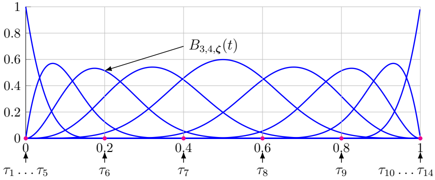
Echivalentul global: $\sum_{i=1}^{n} B_{i,d,\zeta}(t) = 1, \ \forall t \in [\tau_{d+1}, \tau_{n+1})$.



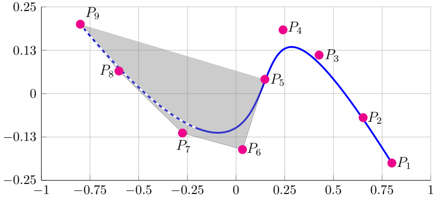
Variații - funcții B-spline (III)

- un knot-vector ce partiționează intervalul $[0, 1]$ în $p = 5$ sub-intervale echidistante
- primul și ultimul punct au multiplicitate $d + 1 = 5$
- aplicând definiția pentru $d = 4$ conduce la $n = 10$ funcții B-spline bază
- adăugând puncte de control, conduce la o curbă B-Spline

funcții bază B-spline



curba B-spline și punctele ei de control



Cuprins

Concluzii

- Mașina Dubins și variațiile sale acoperă o largă clasă de platforme robotice pentru deplasarea în plan.
- Traiectoriile Dubins sunt relativ conservative dar se pot construi relativ ușor/analitic.
- Reformulările cu platitudine permit să exprimăm intrarea în funcție de ieșirea plată.
- Ieșirea plată se parametrizează cu funcții bază (Bezier sau B-spline) ceea ce permite reformularea problemei de optimizare asociate unei probleme de planificare a mișcării.

Referințe

[1] LaValle, S. M.*Planning algorithms*. 2006.

[2] Lyche, T., C. Manni **and** H. Speleers.“Foundations of spline theory: B-splines, spline approximation, and hierarchical refinement”.**in***Splines and PDEs: From Approximation Theory to Numerical Linear Algebra*: 2018,pp. 1–76, 2018.