TAČR Centrum kompetence CIDAM

SW Tools and methodology for optimization of mechatronic systems in terms of structure, parameters and control

SW knihovna funkcí a funkčních bloků pro modelování robotů

Výzkumná zpráva V011

Martin Švejda

13. prosince 2017

T A

Č R

CIDAM

Center for Intelligent Drives and Advanced Machine Control

Abstrakt

Předložená zpráva se zabývá popisem knihovny SW nástrojů implementovaných v prostředí Matlab, Simulink/SimMechanics a zahrnuje základní funkce pro řešení kinematických a dynamických úloh především sériových manipulátorů. Knihovna je určena zejména pro efektivní a rychlé možnosti tvorby virtuálních simulačních modelů manipulátorů, jejich analýzu a jako podpůrné SW prerekvizita pro výzkum a vývoj pokročilých algoritmů návrhu, optimalizace a řízení pohybu robotických architektur.

Obsah

1 Úvod

Předložená zpráva zahrnuje uživatelský popis knihovny předeimplementovaných funkcí v prostředí Matlab a funkčních bloků v prostředí Matlab/Simulink/SimMechanics, které jsou určeny k efektivnímu vytváření virtuálních simulačních modelů robotických manipulátorů. V předložené knihovně jsou implementovány elementární i pokročilé funkce z oblasti kinematického a dynamického modelování manipulátorů. Knihovna funkcí a funkčních bloků je využívána v současnosti především v následujících oblastech:

- Výzkum a vývoj nových robotických architektur (modelování, simulace a analýza)
- Optimalizace robotických architektur (optimalizační algoritmy mohou ve svých výpočtech efektivně využívat elementárních předimplementovaných algoritmů, např. vyčíslení hodnoty účelové funkce)
- Podpora pro vývoj pokročilých metod řízení pohybu robotů (předimplementované funkce tvoří základní stavební prvky pro vyvíjené algoritmy řízení pohybu, např. řízení síly, impedanční řízení, optimální řízení pohybu redundantních manipulátorů, atd.)

Poznamenejme, že předložená knihovna funkcí a funkčních bloků byla intenzivně využívána při návrhu a analýze vyvíjených robotů typu „Zakladač“ [5, 9, 10] a „Vodník“ [8, 7].

2 Standardní funkce a funkční bloky pro modelování sériových manipulátorů

Implementace funkcí a funkčních bloků vychází z metod a algoritmů popsaných v dostupné literatuře, především pak v [2, 3], detailní rozbor algoritmů spadající do knihovny „robotLib“ lze nalézt v [6]. Každé rameno uvažovaného sériového manipulátoru je aktuováno právě jedním rotačním (typ **R**) či translačním (typ **P**) aktuátorem.

Předimplementované funkce - Matlab (m-files)

- `T = DH(DHpar)`:
  Vytvoří homogenní transformační matici.

  Vstupy:

  `DHpar` Vektor Denavit-Hartenbergových (D-H) parametrů, ve formátu: $[d, \theta, a, \alpha]$

  Výstupy:

  `T` Odpovídající homogenní transformační matici $\boldsymbol{T}(d, \theta, a, \alpha)$, dle D-H úmluvy.

- `[R,dR,ddR] = EulerAnglesZYX2rotMatrix([EA,dEA,ddEA])`[1]:
  Přepočet z Eulerových úhlů (v uspořádání postupných rotací ZYX) na matici rotace.

[1] Analogicky implementované i funkce pro uspořádání postupných rotací XYZ: `EulerAnglesXYZ2rotMatrix`, `rotMatrix2EulerAnglesXYZ`

Vstupy:

`EA, dEA, ddEA` Vektor Eulerových úhlů (polohy, rychlosti, zrychlení), ve formátu: $EA = [\gamma, \beta, \alpha]$, dEA resp. $ddEA$ odpovídají rychlostem resp. zrychlením.

Výstupy:

`R, dR, ddR` Odpovídající matice rotace a její časové derivace.

- `[EA,dEA,ddEA] = rotMatrix2EulerAnglesZYX([R,dR,ddR])`[1]:
  Přepočet z matice rotace na Eulerovy úhly (v uspořádání postupných rotací ZYX). Vyřešena singularity v reprezentaci (fixováním hodnoty úhlu γ).

  Vstupy:

  `R, dR, ddR` Matice rotace a její časové derivace.

  Výstupy:

  `EA, dEA, ddEA` Vektor Eulerových úhlů (polohy, rychlosti, zrychlení), ve formátu: $EA = [\gamma, \beta, \alpha]$, dEA resp. $ddEA$ odpovídají rychlostem resp. zrychlením.

- `[omega,domega] = rotMatrix2angularVelAccel([R,dR])`:
  Přepočet z matice rotace $\boldsymbol{R}$ a její časové derivace $\dot{\boldsymbol{R}}$ na vektor úhlové rychlosti $\boldsymbol{\omega}$ a zrychlení $\dot{\boldsymbol{\omega}}$.

  Vstupy:

  `R, dR` Matice rotace a její časové derivace.

  Výstupy:

  `omega,domega` Odpovídající vektor úhlové rychlosti a zrychlení.

- `[dR,ddR] = angularVelAccel2rotMatrix([R,omega,domega])`:
  Přepočet z matice rotace $\boldsymbol{R}$ a vektoru úhlové rychlosti $\boldsymbol{\omega}$ a úhlového zrychlení $\dot{\boldsymbol{\omega}}$ na odpovídající první $\dot{\boldsymbol{R}}$ a druhou $\ddot{\boldsymbol{R}}$ časovou derivaci matice rotace.

  Vstupy:

  `R,omega,domega` Matice rotace, vektor úhlové rychlosti a zrychlení.

  Výstupy:

  `dR,ddR` Odpovídající první a druhá časová derivace matice rotace.

- `Links = robotSetup(kinPar,dynPar)`:
  Vytvoří strukturu kinematických a dynamických parametrů pro i-tý sériový kinematický řetězec - využíváno dalšími funkčními bloky.

Vstupy:

`kinPar{i}.`	Struktura kinematických parametrů manipulátoru (pro i-tý sériový kinematický řetězec)
`.DHpar`	Matice D-H parametrů: $[d_1, \theta_1, a_1, \alpha_1; d_2, \theta_2, a_2, \alpha_2; \ldots; d_n, \theta_n, a_n, \alpha_n]$
`.jointType`	Typy příslušných kloubů (1 **P** či **R** kloub na 1 rameno - jeden řádek D-H parametrů): např.: $[\boldsymbol{P}; \boldsymbol{R}; \ldots; \boldsymbol{R}]$
`.Qhome`	Počáteční (domovská) hodnota polohy kloubových souřadnic (příslušné D-H parametry jsou nahrazeny odpovídajícími hodnotami `Qhome`)
`.endEffComp`	Kompenzace polohy koncového efektoru, ve formátu: $[\boldsymbol{O}_e^n, \boldsymbol{R}_e^n]$
`.baseComp`	Kompenzace polohy základny, ve formátu: $[\boldsymbol{O}_0^b, \boldsymbol{R}_0^b]$
`dynPar{i}.`	Struktura dynamických parametrů manipulátoru (pro i-tý sériový kinematický řetězec)
`.mass`	Vektor hmotností ramen, ve formátu: $[m_1; m_2 \ldots; m_n]$
`.inertiaTensor`	Vektor tensorů setrvačnosti ramen v těžišti vzhledem k s.s. příslušného ramene, ve formátu: $[\boldsymbol{I}_1, \boldsymbol{I}_2, \ldots, \boldsymbol{I}_n]$
`.gravityCenter`	Matice vektorů těžišť ramen vzhledem k příslušnému s.s. ramene, ve formátu: $[\boldsymbol{T}_1, \boldsymbol{T}_2, \ldots, \boldsymbol{T}_N]$
`.gravityVector`	Vektor gravitačního zrychlení vzhledem k s.s. F_b, ve formátu: $[g_x; g_y; g_z]$

Výstupy:

`Links{i}{j+1}.`	Struktura obsahující informace o j-tém ramenu, i-tého kinematického řetězce ($j = 1, 2 \ldots, n$). Struktura je dále používána funkčními bloky.
`.A`	Počátek j-tého ramene, tzn. počátek s.s. F_{j-1}
`.B`	Konec j-tého ramene, tzn. počátek s.s. j-tého ramene F_j
`.R`	Matice rotace s.s. j-tého ramene F_j, vzhledem k s.s. F_b
`.qhome`	Hodnota domovské polohy kloubové souřadnice j-tého ramene
`.mass`	Hmotnost j-tého ramene
`.inertiaTensor`	(Konstantní) tensor setrvačnosti j-tého ramene, vzhledem k s.s. ramene F_j
`.gravityCenter`	(Konstantní) umístěné těžiště j-tého ramene, vzhledem k s.s. ramene F_j
`.jointAxis`	Směrový vektor osy rotace/translace j-tého kloubu, vzhledem k s.s. F_b
`.jointType_sigma`	Identifikátor typu j-tého kloubu, $\sigma = 0$ pro **R** kloub, $\sigma = 1$ pro **P** kloub
`.R0b`	(jen pro $j = 0$) Kompenzace polohy základny, matice rotace $\boldsymbol{R}_0^b$ s.s. F_0, vzhledem k s.s. F_b
`.O0b`	(jen pro $j = 0$) Kompenzace polohy základny, počátek $\boldsymbol{O}_0^b$ s.s. F_0, vzhledem k s.s. F_b
`.gravityVector`	(jen pro $j = 0$) Vektor gravitačního zrychlení vzhledem k s.s. F_b
`.numOfLinks`	(jen pro $j = 0$) Počet ramen manipulátoru n (bez kompenzace polohy základny a koncového efektoru)
`.Ren`	(jen pro $j = n+1$) Kompenzace polohy konc. efektoru, matice rotace $\boldsymbol{R}_e^n$ s.s. F_e, vzhledem k s.s. F_n
`.Oen`	(jen pro $j = n+1$) Kompenzace polohy konc. efektoru, počátek $\boldsymbol{O}_e^n$ s.s. F_e, vzhledem k s.s. F_n

- `[J,dJ,Jcomp,Jadd] = kinJacobian(jointCoords,DHparam,jointType,...`
  `...baseComp,endEffComp)`:
  Výpočet kinematického jakobiánu $\boldsymbol{J}_n$ a jeho časové derivace $\dot{\boldsymbol{J}}_n$, včetně kompenzace $\boldsymbol{J}_{\text{comp}}$ a aditivní konstanty $\boldsymbol{J}_{\text{add}}$ (kompenzace polohy základny a koncového efektoru).

  Vstupy:

`jointCoords`	Hodnoty poloh kloubových souřadnic a odpovídající rychlosti, ve formátu: $[\boldsymbol{Q}, \dot{\boldsymbol{Q}}]$
`DHparam`	Matice D-H parametrů, viz výše
`joinType`	Typy příslušných kloubů, viz výše
`endEffComp`	Kompenzace polohy koncového efektoru, viz výše
`baseComp`	Kompenzace polohy základny, viz výše

  Výstupy:

`J`	Kinematický jakobián $\boldsymbol{J}_n \in \mathbb{R}^{6\times n}$
`dJ`	Časová derivace kinematického jakobiánu $\dot{\boldsymbol{J}}_n \in \mathbb{R}^{6\times n}$
`Jcomp`	Kompenzační matice $\boldsymbol{J}_{\text{comp}} \in \mathbb{R}^{6\times 6}$
`Jadd`	Aditivní vektor (zrychlení) $\boldsymbol{J}_{\text{add}} \in \mathbb{R}^{6\times 1}$

- `[genCoords] = forwardKinematics(jointCoords,kinPar,N)`:[2]
  Výpočet dopředné geometrické úlohy a dopředné okamžité kinematické úlohy (rychlosti, zrychlení). Obecně platná pro sériové manipulátory.

  Vstupy:

`jointCoords`	Hodnoty poloh kloubových souřadnic, odpovídajících rychlostí a zrychlení, ve formátu: $[\boldsymbol{Q}, \dot{\boldsymbol{Q}}, \ddot{\boldsymbol{Q}}]$
`kinPar`	Kinematické parametry manipulátoru, viz funkce `robotSetup`
`N`	Pro $N = j+1$ bude výstupem funkce poloha, rychlost a zrychlení j-tého ramena, tedy s.s. F_j, $j = 1, 2, \ldots, n$, pro $N = 1$ resp. $N = n+2$ bude výstupem poloha, rychlost, zrychlení 0-tého s.s. F_0 resp. s.s. konc. efektoru F_e. Pokud N nezadáno, vracena poloha, rychlost a zrychlení s.s. konc. efektoru F_e

  Výstupy:

`genCoords`	Polohy, rychlosti a zrychlení zobecněných souřadnic (závisí na vstupu `N`) odpovídajícího s.s. vzhledem k s.s. F_b, např. pro N nezadáno je výstup ve formátu $\left[[\boldsymbol{O}_e^b, \boldsymbol{R}_e^b], [\dot{\boldsymbol{O}}_e^b, \boldsymbol{\omega}_e^b], [\ddot{\boldsymbol{O}}_e^b, \dot{\boldsymbol{\omega}}_e^b]\right]$

- `[tau] = inverseDynamicModel(jointCoords,endEff_force_moment,kinPar,dynPar)`:
  Výpočet inverzní dynamické úlohy pro sériový manipulátor, tzn. požadovaných sil/silových momentů kloubových souřadnic $\boldsymbol{\tau}$ potřebných k realizaci pohybu manipulátoru odpovídající poloze $\boldsymbol{Q}$, rychlosti $\dot{\boldsymbol{Q}}$ a zrychlení $\ddot{\boldsymbol{Q}}$ kloubových souřadnic při uvažování externí síly a silového momentu $\boldsymbol{F}$ působící na koncový efektor. Řešení odpovídá dynamické rovnici:

$$\boldsymbol{\tau} = \boldsymbol{M}(\boldsymbol{Q}) \cdot \ddot{\boldsymbol{Q}} + \boldsymbol{C}(\boldsymbol{Q}, \dot{\boldsymbol{Q}}) \cdot \dot{\boldsymbol{Q}} + \boldsymbol{G}(\boldsymbol{Q}) + \boldsymbol{J}^T(\boldsymbol{Q}) \cdot \boldsymbol{F}$$

[2] Poznamenejme, že funkce `inverseKinematics` není předimplementována, neboť nemá analytické (a algoritmizovatelné) řešení pro obecný sériový manipulátor. Pro účely inverzní okamžitou kinematickou úlohu lze využít funkce `kinJacobian`.

Vstupy:

`jointCoords`	Hodnoty poloh kloubových souřadnic, odpovídajících rychlostí a zrychlení, ve formátu: $[\boldsymbol{Q}, \dot{\boldsymbol{Q}}, \ddot{\boldsymbol{Q}}]$
`endEff_force_moment`	Externí síla a silový moment $\boldsymbol{F}$ působící na koncový efektor, vzhledem k s.s. F_b, ve formátu: $[\boldsymbol{f}; \boldsymbol{\mu}]$
`kinPar`	Kinematické parametry manipulátoru, viz funkce `robotSetup`
`dynPar`	Dynamické parametry manipulátoru, viz funkce `robotSetup`

Výstupy:

`tau`	Vektor sil/silových momentů $\boldsymbol{\tau}$ příslušných kloubových souřadnic, ve formátu: $[\tau_1; \tau_2; \dots \tau_n]$

- `[ddq] = forwardDynamicModel(jointCoords,joint_force_moment,...`
  `...endEff_force_moment,kinPar,dynPar)`:
  Výpočet dopředné dynamické úlohy pro sériový manipulátor, tzn. kloubových zrychleních $\ddot{\boldsymbol{Q}}$ na základě známých poloh $\boldsymbol{Q}$ a rychlostí $\dot{\boldsymbol{Q}}$ kloubových souřadnic a externího silového/momentového působení $\boldsymbol{F}$ na koncový efektor. Řešení odpovídá dynamické rovnici:

$$\ddot{\boldsymbol{Q}} = \boldsymbol{M}^{-1}(\boldsymbol{Q}) \cdot \left(\boldsymbol{\tau} - \boldsymbol{C}(\boldsymbol{Q}, \dot{\boldsymbol{Q}}) \cdot \dot{\boldsymbol{Q}} - \boldsymbol{G}(\boldsymbol{Q}) - \boldsymbol{J}^T(\boldsymbol{Q}) \cdot \boldsymbol{F} \right)$$

Vstupy:

`jointCoords`	Hodnoty poloh kloubových souřadnic a odpovídajících rychlostí, ve formátu: $[\boldsymbol{Q}, \dot{\boldsymbol{Q}}]$
`joint_force_moment`	Vektor sil/silových momentů $\boldsymbol{\tau}$ příslušných kloubových souřadnic, ve formátu: $[\tau_1; \tau_2; \dots \tau_n]$
`endEff_force_moment`	Externí síla a silový moment $\boldsymbol{F}$ působící na koncový efektor, vzhledem k s.s. F_b, ve formátu: $[\boldsymbol{f}; \boldsymbol{\mu}]$
`kinPar`	Kinematické parametry manipulátoru, viz funkce `robotSetup`
`dynPar`	Dynamické parametry manipulátoru, viz funkce `robotSetup`

Výstupy:

`ddq`	Vektor zrychleních kloubových souřadnic $\ddot{\boldsymbol{Q}}$, ve formátu $[\ddot{q}_1; \ddot{q}_2; \dots; \ddot{q}_n]$

Implementované funkční bloky - Matlab/Simulink/SimMechanics

Viz Obrázek 1.

- `Manipulator Link`:
  Blok reprezentující j-té rameno manipulátoru umístěné v i-tém kinematickém řetězci[3].

  Vstupy/Výstupy:

`CS Fi-1,CS Fi`	Přípojné body ramena tvoří s.s. ramene F_j a s.s. ramene předcházejícího F_{j-1}

  Parametry:

`Chain order`	Pořadí kinematického řetězce, kde je rameno umístěno.
`Link order`	Pořadí umístění ramene v daném kinematickém řetězci.

[3] Poznamenejme, že možnost realizovat různé kinematické řetězce vznikla s ohledem na možnosti realizace paralelních kinematických architektur.

- `Manipulator Joint (Actuation: Motion)`:
  Blok reprezentující j-tý kloub umístěný v i-tém kinematickém řetězci. Kloub je aktuován pohybem (tzn. příslušnou polohou, rychlostí a zrychlením své kloubové souřadnice)[4].

  Vstupy/Výstupy:

`B,F`	*Base*, *Follower* - standardní přípojné body kloubů v SimMechanicsu
`[pos,vel,accel]`	Poloha, rychlost a zrychlení kloubové souřadnice kloubu.
`Com. force/torque`	Požadované síla/silový moment (skalár) potřebný k definovanému pohybu kloubu (interní řešení inverzní dynamické úlohy solverem v SimMechanicsu)
`Reaction torque`	Požadovaný silový moment (vektor) reprezentující rotační reakci kloubu do závěsu
`Reaction force`	Požadovaná síla (vektor) reprezentující translační reakci kloubu do závěsu

  Parametry:

`Chain order`	Pořadí kinematického řetězce, kde je kloub umístěn.
`Joint order`	Pořadí umístění kloubu v daném kinematickém řetězci.

- `joint home position`:
  Blok vrací hodnotu domovské polohy kloubové souřadnice j-tého kloubu umístěného v i-tém kinematickém řetězci.

  Vstupy/Výstupy:

-	Výstupní hodnota domovské polohy kloubové souřadnice (nulová rychlost a zrychlení), ve formátu $[qhome; 0; 0]$

  Parametry:

`Chain order`	Pořadí kinematického řetězce, kde je kloub umístěn.
`Joint order`	Pořadí umístění kloubu v daném kinematickém řetězci.

- `Base Compensation`:
  Blok reprezentující kompenzaci polohy základny (nehmotné rameno pevně připojeno k nultému s.s.).

  Vstupy/Výstupy:

`CS Fb,CS F0`	Přípojné body reprezentované s.s. základny F_b a s.s. F_0

  Parametry:

`Chain order`	Pořadí kinematického řetězce, kde je kompenzace umístěna.

- `End Eff. Compensation`:
  Blok reprezentující kompenzaci polohy koncového efektoru (nehmotné rameno pevně připojeno k poslednímu s.s. ramene).

  Vstupy/Výstupy:

`CS Fe,CS Fn`	Přípojné body reprezentované s.s. koncového efektoru F_e a s.s. posledního ramene F_n

  Parametry:

`Chain order`	Pořadí kinematického řetězce, kde je kompenzace umístěna.

[4] V souvislosti s výstupem bloku `Com. force/torque` vyvstává otázka, zda-li je nutné řešit inverzní dynamickou úlohu (potenciálně dopřednou dynamickou úlohu) externí funkcí (`inverseDynamicModel`, `forwardDynamicModel`). Odpověď jednoznačně spočívá v předpokládanému použití implementovaných funkcí k řešení optimalizačních úloh, neboť výpočetní čas potřebný k výpočtu inverzní/dopředné dynamické úlohy prostřednictvím implementovaných funkcí je ve srovnání s možným spouštěním simulace v prostředí SimMechanics (a využití získaných výsledků) nesrovnatelně kratší.

- `Manipulator P Joint (Actuation: Passive)`:
  Blok reprezentující j-tý kloub typu **P** umístěný v i-tém kinematickém řetězci. Kloub není aktuován (pasivní kloub). Pasivní kloub je používán při modelování paralelních robotických architektur.

  Vstupy/Výstupy:

`B,F`	*Base*, *Follower* - standardní přípojné body kloubů v SimMechanicsu
`[pos,vel,accel]`	Poloha, rychlost a zrychlení kloubové souřadnice kloubu.
`Com. force/torque`	Požadované síla/silový moment (skalár) potřebný k definovanému pohybu kloubu (interní řešení inverzní dynamické úlohy solverem v SimMechanicsu)
`Reaction torque`	Požadovaný silový moment (vektor) reprezentující rotační reakci kloubu do závěsu
`Reaction force`	Požadovaná síla (vektor) reprezentující translační reakci kloubu do závěsu

  Parametry:

`Chain order`	Pořadí kinematického řetězce, kde je kloub umístěn.
`Joint order`	Pořadí umístění kloubu v daném kinematickém řetězci.

- `Manipulator R Joint (Actuation: Passive)`:
  Blok reprezentující j-tý kloub typu **R** umístěný v i-tém kinematickém řetězci. Kloub není aktuován (pasivní kloub). Pasivní kloub je používán při modelování paralelních robotických architektur.

  Vstupy/Výstupy:

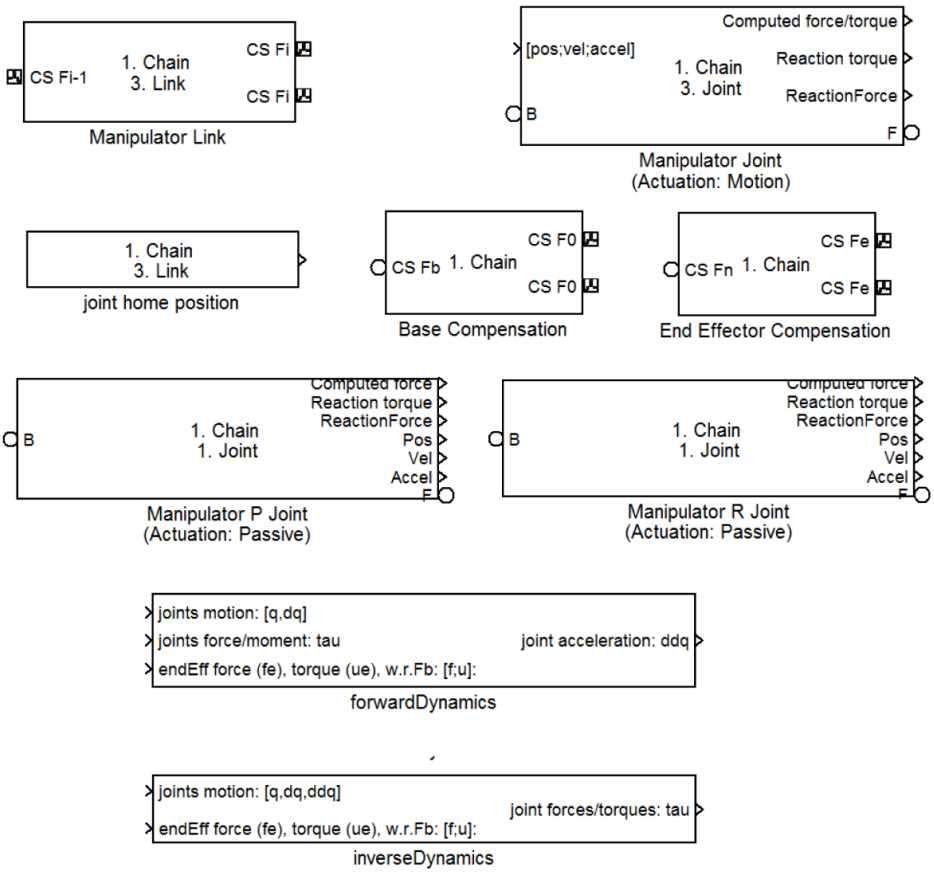
`B,F`	*Base*, *Follower* - standardní přípojné body kloubů v SimMechanicsu
`[pos,vel,accel]`	Poloha, rychlost a zrychlení kloubové souřadnice kloubu.
`Com. force/torque`	Požadované síla/silový moment (skalár) potřebný k definovanému pohybu kloubu (interní řešení inverzní dynamické úlohy solverem v SimMechanicsu)
`Reaction torque`	Požadovaný silový moment (vektor) reprezentující rotační reakci kloubu do závěsu
`Reaction force`	Požadovaná síla (vektor) reprezentující translační reakci kloubu do závěsu

  Parametry:

`Chain order`	Pořadí kinematického řetězce, kde je kloub umístěn.
`Joint order`	Pořadí umístění kloubu v daném kinematickém řetězci.

- `forwardDynamics`:
  Blok reprezentující řešení dopředné dynamické úlohy založený na funkci `forwardDynamicModel` s odpovídajícími vstupy/výstupy. Bez parametrů. Řešení dopředné dynamické úlohy je implementováno **pouze pro sériové manipulátory.**

- `inverseDynamics`:
  Blok reprezentující řešení inverzní dynamické úlohy založený na funkci `inverseDynamicModel` s odpovídajícími vstupy/výstupy. Bez parametrů. Řešení inverzní dynamické úlohy je implementováno **pouze pro sériové manipulátory.**



Obrázek 1: Standardní bloky knihovny do prostředí SimMechanics

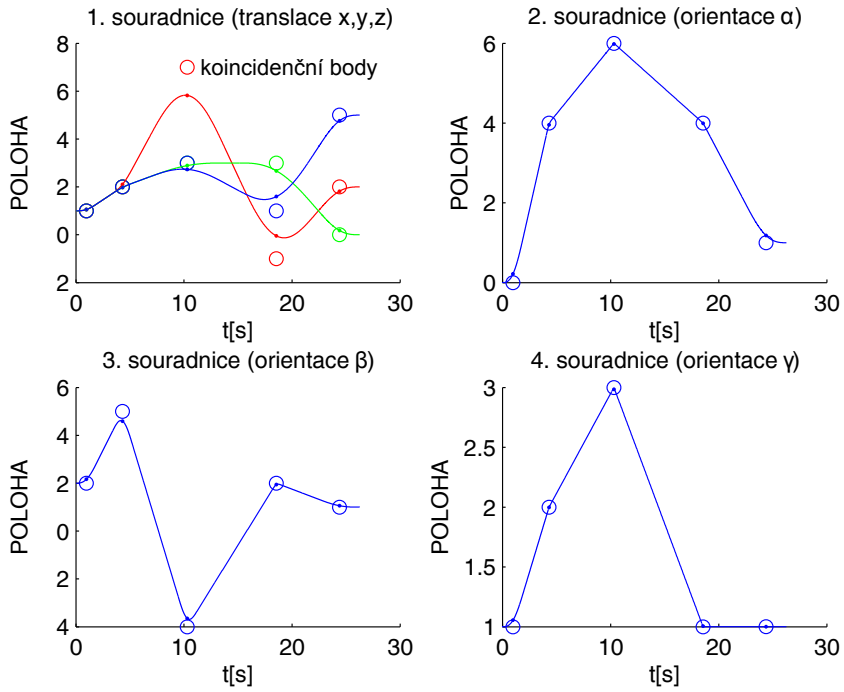
3 Funkce pro plánování trajektorie koncového efektoru

Poznamenejme, že za účelem možností simulačního ověření chování vyvíjených architektur manipulátorů byly vyvinuty a implementovány některé algoritmy plánování trajektorie formou níže uvedených funkcí v Matlabu. Shrňme pouze jejich základní vlastnosti, detailní informace lze nalézt v [4].

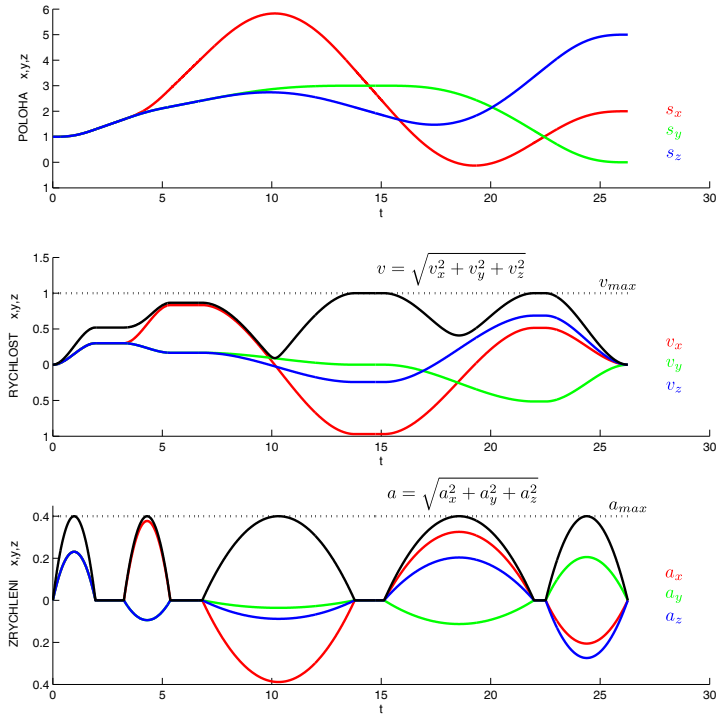
Předimplementované funkce - Matlab (m-files)

- `[pos,alpha,beta,gamma]=lineInterpolationWithPolynomialBlending(points,...`
  `...axesConstraints,timeRes)`:
  Jedná se o aproximační metodu. Zadané koincidenční body `points` zadané pozicí a orientací (Eulerovy úhly) jsou interpolovány přímkovými segmenty s polynomiálním napojováním, tzn. v definované vzdálenosti od interpolovaného bodu spojitě přechází přímkový segment v polynomiální (kubický), tedy interpolovaná trajektorie přesně prochází pouze prvním a posledním koincidenčním bodem. Omezení `axesConstraints` obsahují požadavky na maximální rychlost (tečnou) a zrychlení (obecné, tzn. tečné a normálové), které nesmí být porušeny. Právě omezení udávají tvar polynomiálního napojení. Omezení na maximální zrychlení lze interpretovat v aplikacích „pick and place“ jako omezení na zrychlení v zatáčkách trajektorie s ohledem na limity udržení neseného břemene uchopovačem na koncovém efektoru manipulátoru. Příklad trajektorie je znázorněn na Obrázku 2, odpovídající průběhy rychlosti a zrychlení potom na Obrázku 3. Poznamenejme, že z přirozeného důvodu

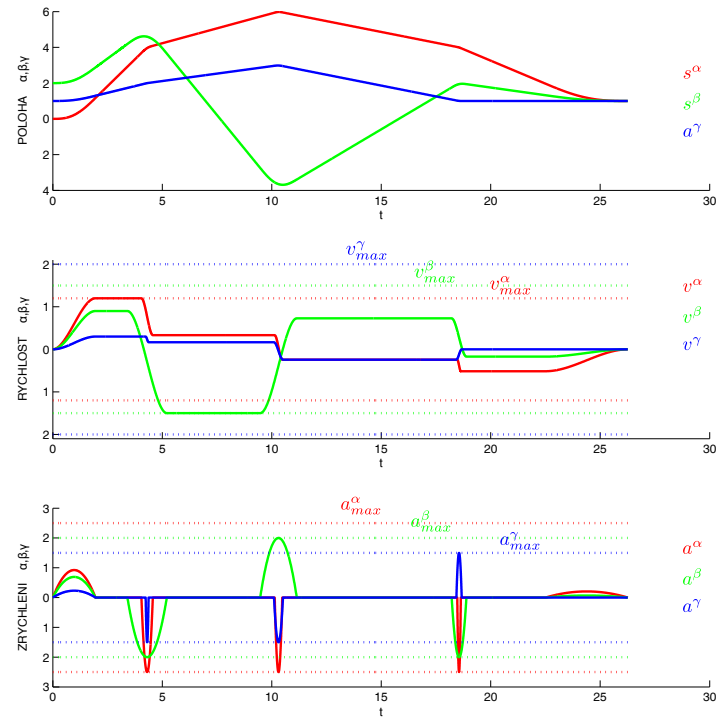
je uvažována rychlost resp. zrychlení translace jako norma složek rychlosti resp. zrychlení (omezení v_{max}, a_{max}) zatímco u orientace je omezována každá složka (Eulerův úhel) zvlášť ($v^{\star}_{\max}$, $a^{\star}_{\max}$, $\star = \{\alpha, \beta, \gamma\}$).



Obrázek 2: Příklad aproximace zadaných koincidenčních bodů (přímková interpolace s polynomiálním napojováním)



(a) Pro translaci



(b) Pro orientaci

Obrázek 3: Průběh polohy, rychlosti a zrychlení podél uvažované trajektorie

Vstupy:

`points` Matice m koincidenčních bodů (translace $\boldsymbol{O}$, rotace α, β, γ), ve formátu:

$$\left[\begin{array}{c|c|c|c} \boldsymbol{O}_1 & \boldsymbol{O}_2 & \dots & \boldsymbol{O}_m \\ \alpha_1 & \alpha_2 & \dots & \alpha_m \\ \beta_1 & \beta_2 & \dots & \beta_m \\ \gamma_1 & \gamma_2 & \dots & \gamma_m \end{array}\right]$$

`axesConstraints` Matice omezení pohybu, ve formátu:

$$\left[\begin{array}{c|c|c|c} v_{max} & v^{\alpha}_{max} & v^{\beta}_{max} & v^{\gamma}_{max} \\ a_{max} & a^{\alpha}_{max} & a^{\beta}_{max} & a^{\gamma}_{max} \end{array}\right]$$

`timeRes` Časová diference, se kterou jsou generovány body výstupních vektorů

Výstupy:

`pos` Datová struktura s časem (*Structure with time* v interpretaci v prostředí Matlab), ve formátu:

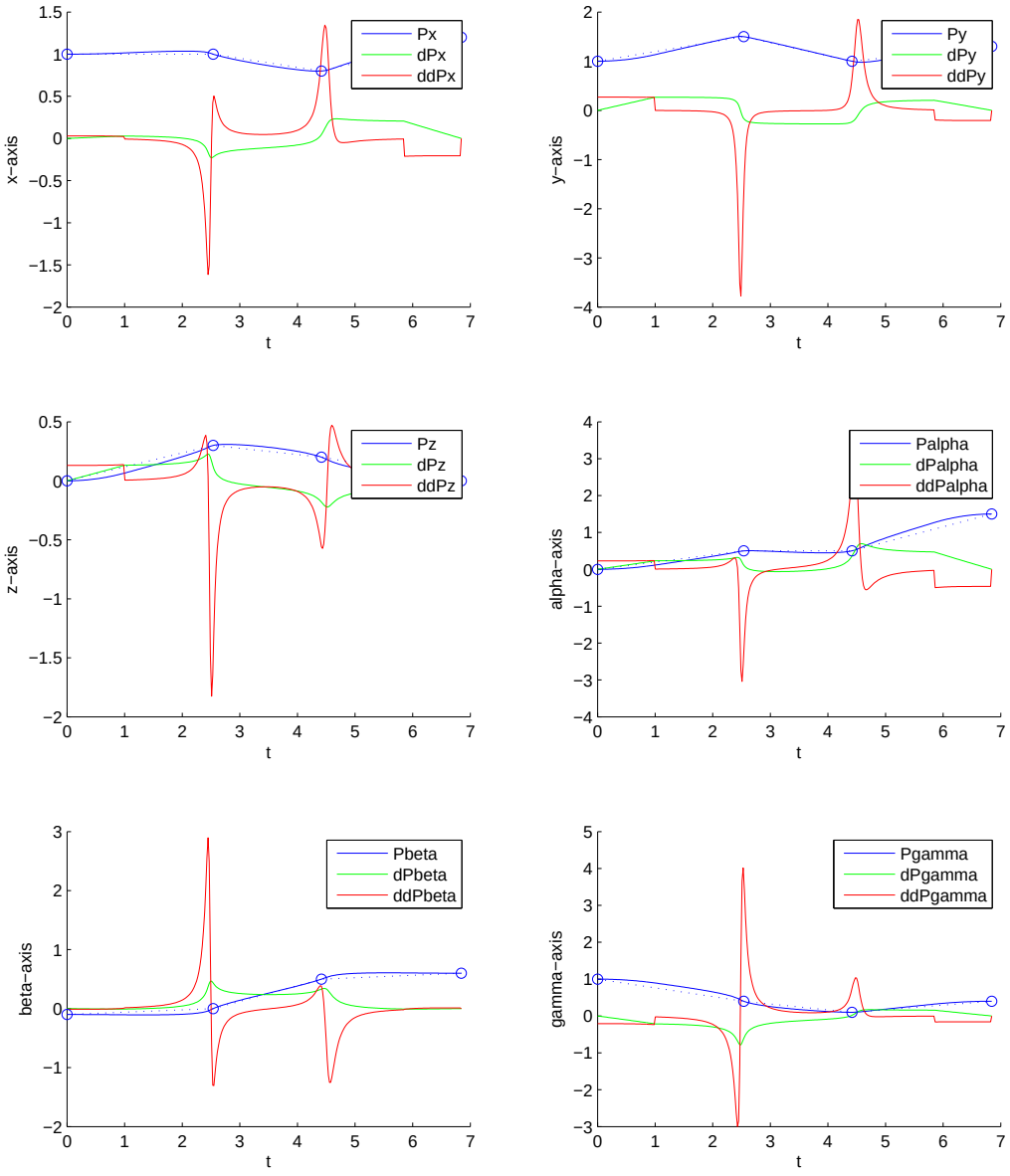
$$\texttt{pos.signals.values} = \begin{bmatrix} t_0: & x & y & z & \dot{x} & \dot{y} & \dot{z} & \ddot{x} & \ddot{y} & \ddot{z} \\ t_0 + \texttt{timeRes}: & x & y & z & \dot{x} & \dot{y} & \dot{z} & \ddot{x} & \ddot{y} & \ddot{z} \\ \vdots & & & & & & & & & \end{bmatrix}$$

`alpha, beta, gamma` Datová struktura s časem (*Structure with time* v interpretaci v prostředí Matlab), ve formátu:

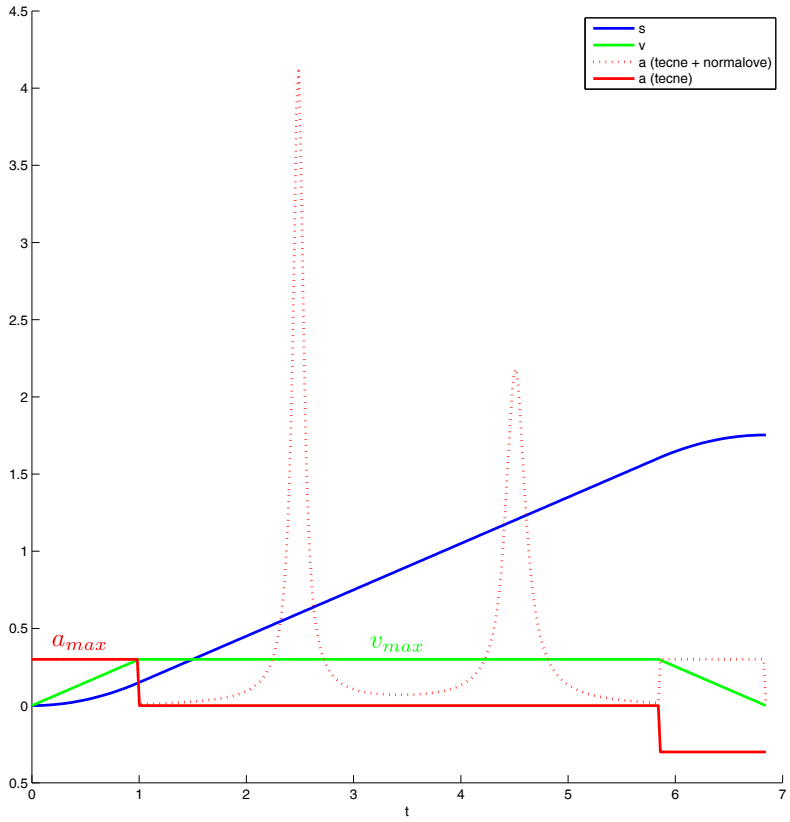
$$\star\texttt{.signals.values} = \begin{bmatrix} t_0: & \star & \dot{\star} & \ddot{\star} \\ t_0 + \texttt{timeRes}: & \star & \dot{\star} & \ddot{\star} \\ \vdots & & & \end{bmatrix}$$

kde $\star = \{\alpha, \beta, \gamma\}$

- `[trajectory]=interpolator_cubic_time(points,...`
  `...axesConstraints,timeRes,feedType)`:
  Metoda generování trajektorie pohybu koncového efektoru, která je od předchozí odlišná zejména ve dvou zásadních aspektech. Za prvé se jedná o interpolační metodu, která prokládá (přesně) koincidenční body spline křivkou (po částech kubickým polynomem, který zajišťuje pouze nativní předpoklady na spojitost polohy, rychlosti a zrychlení v koincidenčních bodech). Druhou zásadní změnou je, že profil ujeté dráhy s, rychlosti v a tečného zrychlení a podél křivky je předepsán (tzv. korekce feedrate při generování výsledných interpolovaných dat) jako časově optimálním průběh s omezeními kladenými na maximální rychlost v_{max} a zrychlení a_{max} (typický lichoběžníkový profil rychlosti, viz např. [2, 1]). Příklad trajektorie je znázorněn na Obrázku 4, odpovídající průběhy ujeté dráhy s, rychlosti v a zrychlení a jsou znázorněny na Obrázku 5. Poznamenejme, že v uvedeném případě se jedná o omezení translačního pohybu (průběhy vektoru Eulerových úhlů jsou dopočítávány s ohledem na synchronizaci výsledného pohybu). V případě uvažovaných omezení v orientaci, tzn. ujetá dráha, rychlost a tečné zrychlení vektoru Eulerových úhlů, lze režim přepnout parametrem `feedType`. Uvedená inovativní metoda generování trajektorie včetně numerických algoritmů řešení byla vyvinuta v rámci řešených úloh optimalizace, viz [4].



Obrázek 4: Příklad interpolace zadaných koincidenčních bodů (kubický spline s korekcí feedrate)



Obrázek 5: Průběh polohy, rychlosti a zrychlení podél uvažované trajektorie (omezení v translaci)

Vstupy:

`points` Matice m koincidenčních bodů (translace $\boldsymbol{O}$, rotace α, β, γ), ve formátu:

$$\left[\begin{array}{c|c|c|c} \boldsymbol{O}_1 & \boldsymbol{O}_2 & \dots & \boldsymbol{O}_m \\ \alpha_1 & \alpha_2 & \dots & \alpha_m \\ \beta_1 & \beta_2 & \dots & \beta_m \\ \gamma_1 & \gamma_2 & \dots & \gamma_m \end{array}\right]$$

`axesConstraints` Omezení translačního/rotačního pohybu, ve formátu $[v_{max}, a_{max}]$

`timeRes` Časová diference, se kterou jsou generovány body výstupních vektorů

`feedType` Omezení na max. rychlost a zrychlení pohybu podél interpolované křivky s ohledem na translaci (`feefType='pos'`) či orientaci (`feedType='orient'`)

Výstupy:

`trajectory` Datová struktura s časem (*Structure with time* v interpretaci v prostředí Matlab), ve formátu:

$$\texttt{trajectory.signals.values} = \begin{bmatrix} t_0: & \boldsymbol{pos} & \boldsymbol{vel} & \boldsymbol{accel} \\ t_0 + \texttt{timeRes}: & \boldsymbol{pos} & \boldsymbol{vel} & \boldsymbol{accel} \\ \vdots & & & \end{bmatrix}$$

kde $\boldsymbol{pos} = [x\ y\ z\ \alpha\ \beta\ \gamma]^T$, $\boldsymbol{vel} = [\dot{x}\ \dot{y}\ \dot{z}\ \dot{\alpha}\ \dot{\beta}\ \dot{\gamma}]^T$, $\boldsymbol{accel} = [\ddot{x}\ \ddot{y}\ \ddot{z}\ \ddot{\alpha}\ \ddot{\beta}\ \ddot{\gamma}]^T$.

4 Závěr

Knihovny předimplementovaných funkcí a funkčních bloků mohou být volně staženy zde:

Standardní funkce a funkční bloky pro modelování sériových manipulátorů

`http://home.zcu.cz/~msvejda/PhD_disertace/Algoritmy/robotLib/General/`

Funkce pro plánování trajektorie koncového efektoru

`http://home.zcu.cz/~msvejda/PhD_disertace/Algoritmy/robotLib/TrajectoryPlanning/`

Poděkování

Tento výzkum byl podpořen Technologickou agenturou České republiky z prostředků projektu Centra Kompetence CIDAM TE02000103.

Reference

[1] Bláha, L.: *Plánování pohybu podél specifikované cesty s uvažováním všech pohybových omezení*. Dizertační práce, Západočeská univerzita v Plzni, Katedra kybernetiky, 2014.

[2] Khalil, W.; Dombre, E.: *Modeling, Identification and Control of Robots*. Kogan Page Science paper edition, Elsevier Science, 2004, ISBN 9780080536613.
URL `http://books.google.cz/books?id=nyrY0Pu5kl0C`

[3] Sciavicco, L.; Siciliano, B.: *Modelling and Control of Robot Manipulators*. Advanced Textbooks in Control and Signal Processing, Springer London, 2000, ISBN 9781852332211.
URL `http://books.google.fr/books?id=v9PLbcYd9aUC`

[4] Svejda, M.; Cechura, T.: Interpolation method for robot trajectory planning. In *Process Control (PC), 2015 20th International Conference on*, June 2015, s. 406–411, doi:10.1109/PC.2015.7169997.

[5] Švejda, M.: Kinematical, kinetostatical and dynamical analysis of 4DoF manipulator, parametric optimization of mechanical construction (WP5-DV026). Technická zpráva, NTIS, ZČU v Plzni, 2015.
URL `http://home.zcu.cz/~msvejda/_publications/2015/8_zakladace_navrh_optimalizace.pdf`

[6] Švejda, M.: *Optimalizace robotických architektur*. Dizertační práce, Západočeská univerzita v Plzni, 2016.
URL `http://home.zcu.cz/~msvejda/_publications/2016/4_SvejdaMartin_thesis_2016_06_14.pdf`

[7] Švejda, M.: Dynamická analýza manipulátoru „Vodník“. Technická zpráva, Západočeská univerzita v Plzni, 2017.
URL `http://home.zcu.cz/~msvejda/_publications/2017/7_dynAnalyza.pdf`

[8] Švejda, M.: Návrh, řízení a optimalizace manipulátoru do průmyslových myček („vodník“). Technická zpráva, Západočeská univerzita v Plzni, 2017.
URL `http://home.zcu.cz/~msvejda/_publications/2017/1_vodnik_optimalizace_rizeni.pdf`

[9] Švejda, M.: OPTIMALIZACE PARAMETRŮ ROBOTU TYPU ZAKLADAČ (Verze: 1.2). Technická zpráva, Západočeská univerzita v Plzni, 2017.
URL `http://home.zcu.cz/~msvejda/_publications/2017/10_Zakladac_optimalizace_V1_2.pdf`

[10] Švejda, M.; Jáger, A.: FINÁLNÍ VERZE ROBOTU TYPU ZAKLADAČ (Analýza finálního CAD modelu) (Verze: 1.1). Technická zpráva, Západočeská univerzita v Plzni, 2017.
URL `http://home.zcu.cz/~msvejda/_publications/2017/9_Zakladac_finalniVerze_V1_1.pdf`