Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra kybernetiky

DIPLOMOVÁ PRÁCE

Plzeň, 2017 **Vojtěch STRÁNSKÝ**

Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra kybernetiky

BALANCOVÁNÍ MÍČE NA PRŮMYSLOVÉM ROBOTU

Plzeň, 2017

Vojtěch STRÁNSKÝ

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: **2016/2017**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Vojtěch STRÁNSKÝ**
Osobní číslo:	**A15N0111P**
Studijní program:	**N3918 Aplikované vědy a informatika**
Studijní obor:	**Kybernetika a řídicí technika**
Název tématu:	**Balancování míče na průmyslovém robotu**
Zadávající katedra:	**Katedra kybernetiky**

Zásady pro vypracování:

1. Vytvořte matematický model soustavy složené z průmyslového robotu Stäubli a nestabilně uloženého míče na hrotu efektoru robotu.
2. Navrhněte regulátor stabilizující míč v nestabilní rovnovážné poloze.
3. Navrhněte regulátor zajišťující stabilní pohyb míče po zvolené periodické trajektorii.
4. Navržené regulátory ověřte simulačně užitím matematického modelu soustavy v programovém systému Matlab-SimMechanics.
5. V případě možnosti ověřte navržené řízení též na reálném robotu Stäubli.

Rozsah grafických prací: **dle potřeby**

Rozsah kvalifikační práce: **40-50 stránek A4**

Forma zpracování diplomové práce: **tištěná**

Seznam odborné literatury:

Mark W. Spong, Seth Hutchinson a M. Vidyasagar: Robot Modeling and Control, First Edition, John Wiley and Sons, Inc.

Vedoucí diplomové práce: **Prof. Ing. Miloš Schlegel, CSc.**
Katedra kybernetiky

Datum zadání diplomové práce: **3. října 2016**

Termín odevzdání diplomové práce: **21. května 2017**

Doc. RNDr. Miroslav Lávička, Ph.D.
děkan

L.S.

Prof. Ing. Josef Psutka, CSc.
vedoucí katedry

V Plzni dne 3. října 2016

PROHLÁŠENÍ

Předkládám tímto k posouzení a obhajobě diplomovou práci zpracovanou na závěr studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí.

V Plzni dne .....

.....
Vojtěch Stránský

Poděkování

Tímto bych chtěl poděkovat vedoucímu diplomové práce Prof. Ing. Miloši Schlegelovi, CSc. za jeho rady, obětovaný čas a trpělivost. Dále bych rád poděkoval každému, kdo se podílel na řešení problémů obsažených v této práci.

ABSTRAKT

Tato práce se zabývá úlohou balancování míče na průmyslovém robotu. Naším cílem je demonstrovat možnosti použití manipulátoru pro precizní úkony, které jsou reprezentovány stabilizací míče. Pro zjednodušení je míč modelován jako kyvadlo. Nejprve je sestaven matematický model kyvadla na hrotu. Poté je pro tento model pomocí dvoubodové okrajové úlohy vypočtena trajektorie pohybu splňující určité požadavky. Podél této požadované trajektorie je systém linearizován a pro řízení je navržen lineárně-kvadratický regulátor na konečném horizontu. Následně je v prostředí SimMechanics sestaven model sériového manipulátoru, na kterém je na závěr simulačně ověřena funkčnost navrženého řízení.

KLÍČOVÁ SLOVA

Stabilizace, inverzní kyvadlo, LQR, průmyslový robot, sériový manipulátor, plánování trajektorie

ABSTRACT

This thesis deals with the issue of balancing a ball via industrial robot. Our goal is to demonstrate the possibilities of manipulator usage for precise operations, which are represented by ball stabilization. For simplicity the ball is modelled as a pendulum. First the mathematical model of a pendulum on a thin tip is made. Then we compute desired trajectory of motion via solving two-point boundary value problem. We then linearise the system around this trajectory and we design finite-horizon linear quadratic regulator. Then we make up a model of serial manipulator in SimMechanics. Lastly we verify the function of the designed controller by running a simulation with this model.

KEYWORDS

Stabilization, inverse pendulum, LQR, industrial robot, serial manipulator, trajectory planning

Obsah

1 Úvod

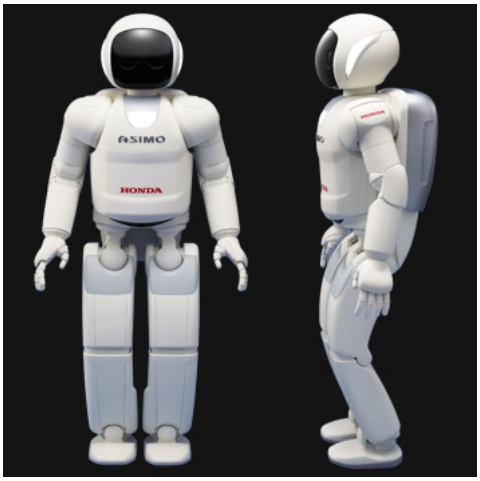
Průmysloví roboti dnes již běžně nahrazují lidskou práci u výrobních linek, kdy například v automobilkách přemísťují těžké díly, kompletují karoserie, svařují nebo lakují. Vzhledem ke snaze o modernizaci výrobních procesů spolu s těmito aplikacemi dnes narůstá poptávka i po automatizaci mnohem preciznějších úkonů, při kterých jsou kladeny větší nároky na přesnost pohybu, ať už z hlediska polohy, nebo z hlediska rychlosti. Pro výrobce manipulátorů z toho plyne jak nutnost případného zlepšení konstrukce robotů, čímž se zlepší jejich dynamické vlastnosti, tak především vývoj přesnějších řídicích algoritmů. Takové řízení je poté samozřejmě nutné ozkoušet a demonstrovat na reálném systému. V této práci pro předvedení možností průmyslového manipulátoru využijeme úlohu balancování míče na hrotu, který bude připevněn na koncový efektor manipulátoru. Aby se povedlo míč ustabilizovat, je zapotřebí přesných a rychlých zásahů, schopnosti manipulátoru tedy dobře prověříme. Balancování míče je poté zobecněním populárního problému stabilizace inverzního kyvadla. Ve své nejzákladnější podobě je systém inverzního kyvadla tvořen kyvadlem otočným kolem jedné osy, které je umístěno na vozíku pohybujícím se v ose kolmé na osu otáčení kyvadla. Kyvadlo se poté snažíme stabilizovat v horní poloze. Jedná se tedy o nestabilní systém s nelineární dynamikou, který je však stabilizovatelný, a právě proto se jedná o jakýsi testovací příklad a populární demonstrační ukázku pro různé strategie řízení.

1.1 Podaktuované systémy

V případě inverzního kyvadla na vozíku se navíc jedná o takzvaně „podaktuovaný“ systém (underactuated system). Tyto systémy se vyznačují tím, že jejich počet ovladatelných stupňů volnosti je menší než jejich celkový počet stupňů volnosti. U plně aktuovaných systémů (fully-actuated systems) jsou přímo aktivně ovládány všechny stupně volnosti. Výše zmíněný systém inverzního kyvadla na vozíku má dva stupně volnosti, jeden pro horizontální pohyb vozíku a druhý pro otáčení kyvadla kolem pivotu, ale ovládat můžeme pouze polohu vozíku a úhel natočení kyvadla je poté ovládán nepřímo. V praxi se podaktuované systémy vyskytují kvůli konstrukčním omezením, kdy není fyzikálně možné systém osadit pohonem pro každý stupeň volnosti, nebo kvůli ekonomickým důvodům, ať už jde o snížení nákladů na akční prvky a konstrukci celého systému nebo o snížení energie potřebné k provozu.

Obecně však nemusí být podaktuované systémy pouze řešením „z donucení“, v některých případech je výhodné využít přirozenou dynamiku konstrukce. Pro srovnání plně aktuovaného a podaktuovaného systému uvedeme následující příklad, ukázaný v [2]. Jako zástupce plně aktuovaného systému byl vybrán obecně známý humanoidní robot ASIMO od firmy Honda (obrázek 1.1.1), který má v každém kloubu dolních končetin umístěn motor. Při chůzi je jeho pohyb plynulý, nepůsobí však úplně přirozeně. Při řízení tohoto robota je totiž použito zpětnovazební řízení s velkým zesílením za účelem potlačení přirozené dynamiky a striktního dodržování požadované trajektorie. Následkem tohoto způsobu řízení je však vysoká spotřeba energie a také omezení stavového prostoru na chůzi po známém rovném povrchu.

Proti tomu byl jako příklad podaktuovaného systému uveden pasivní dynamický chodící „robot“ představený v [4] (obrázek 1.1.2). Jedná se o vhodně sestavenou kon-



Obrázek 1.1.1: Robot ASIMO od firmy Honda [3]

strukci bez jediného akčního prvku, přesto je tento robot schopen velmi přirozené chůze dolů po mírně nakloněné rovině. V některých případech je tedy výhodné s přirozenou dynamikou pracovat a ne se ji snažit potlačit. Mnohem více o podaktuovaných systémech zejména z oblasti robotiky lze najít v [2].

Obrázek 1.1.2: Pasivní dynamický chodící robot představený v [4]

1.2 Balancování míče na průmyslovém robotu

Vraťme se zpět k systému inverzního kyvadla na vozíku. Tento systém byl na Katedře kybernetiky ZČU v Plzni již dříve rozšířen do dvourozměrného prostoru přidáním dalšího stupně volnosti a sestrojením robota balancujícího míčem na hrotu [5]. Myšlenku

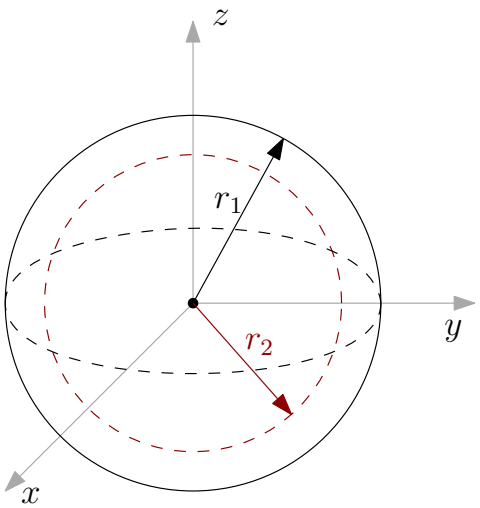
tohoto „robotického lachtana“ nyní dále rozšíříme do třírozměrného prostoru na balancování míče či kyvadla na hrotu, který bude připevněn ke koncovému efektoru malého antropomorfního průmyslového manipulátoru značky Stäubli. V našem případě se tedy sice bude jednat o v podstatě „školní“ úlohu vytvářenou za účelem marketingové prezentace manipulátorů pro značku Stäubli, tato úloha je však velmi podobná praktickým problémům jako jsou například řízení letu rakety, již zmíněná chůze dvounohých robotů nebo samostatné balancování dvoukolového dopravního prostředku Segway PT.

2 Matematický model míče na hrotu

Pro návrh strategie řízení je nejprve zapotřebí vytvořit matematický model řízeného systému, začneme tedy s tvorbou matematického modelu míče nestabilně uloženého na hrotu. Pro určení momentu setrvačnosti budeme míč uvažovat jako kulovitou slupku o určité tloušťce s dutým středem (obrázek 2.0.1). Z odvození, které je k nahlédnutí v [6], obdržíme pro moment setrvačnosti následující vztah:

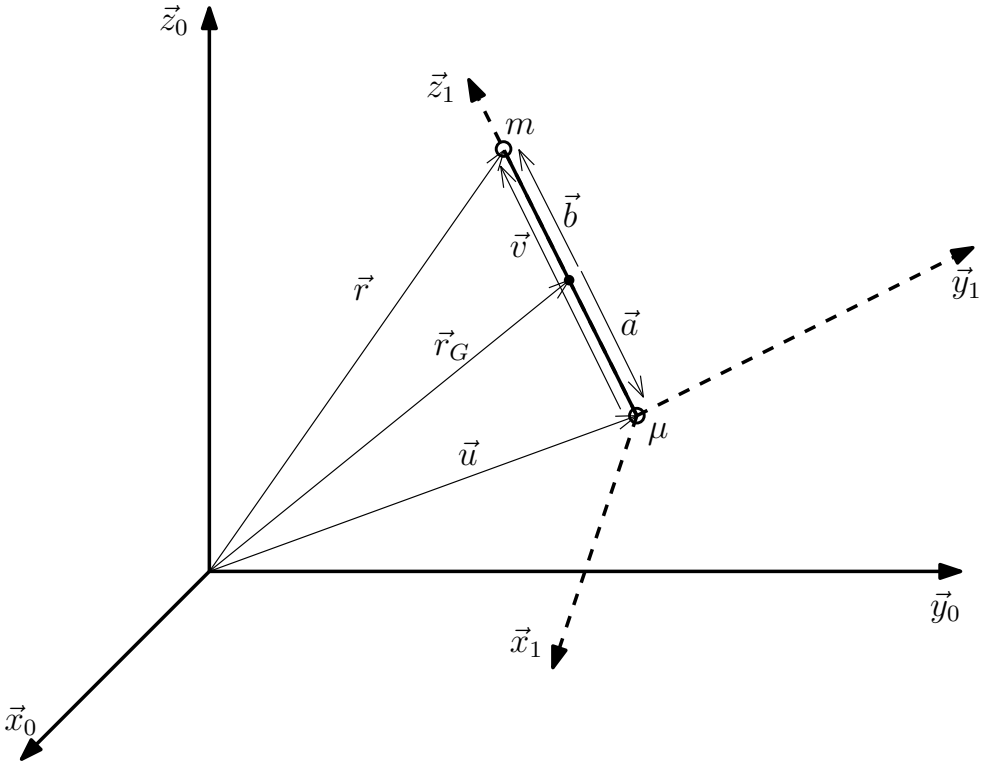
$$J = \frac{2}{5} m \frac{r_2^5 - r_1^5}{r_2^3 - r_1^3} \tag{2.0.1}$$

- J ... moment setrvačnosti $[kg \cdot m^2]$
- m ... hmotnost míče $[kg]$
- r_1 ... vnitřní poloměr slupky míče $[m]$
- r_2 ... vnější poloměr slupky míče $[m]$



Obrázek 2.0.1: Schéma míče

Pro zjednodušení budeme míč modelovat jako matematické kyvadlo se sférickým kloubem. Matematické kyvadlo je tvořeno hmotným bodem, který je zavěšen na tuhém vlákně se zanedbatelnou hmotností. Odpor vzduchu a tření v pivotu je zanedbáváno. Pro odvození pohybových rovnic použijeme Newton-Eulerovu metodu. Systém budeme modelovat jako soustavu dvou hmotných bodů podle následujícího schématu (obrázek 2.0.2).



Obrázek 2.0.2: Schéma kyvadla modelovaného soustavou dvou hmotných bodů

Jednotlivé vektory a proměnné ve schématu mají následující význam:

- $\vec{x}_0,\ \vec{y}_0,\ \vec{z}_0$ ... inerciální soustava souřadnic pevně spojená se zemí
- $\vec{x}_1,\ \vec{y}_1,\ \vec{z}_1$ ... pohyblivá soustava souřadnic pevně spojená s kyvadlem
- m ... hmotnost hmotného bodu na konci kyvadla
- μ ... hmotnost hmotného bodu v pivotu
- $\vec{u}$ ... poloha pivotu vzhledem k inerciální soustavě souřadnic
- $\vec{r}$ ... poloha hmotného bodu na konci kyvadla vzhledem k inerciální soustavě souřadnic
- $\vec{r}_G$ ... poloha těžiště kyvadla vzhledem k inerciální soustavě souřadnic
- $\vec{v}$ ... vzdálenost hmotného bodu na konci kyvadla od pivotu vzhledem k inerciální soustavě souřadnic
- $\vec{a}$ ... vzdálenost pivotu od těžiště kyvadla vzhledem k inerciální soustavě souřadnic
- $\vec{b}$ ... vzdálenost hmotného bodu na konci kyvadla od těžiště kyvadla vzhledem k inerciální soustavě souřadnic

Kyvadlo je tedy tvořeno dvěma hmotnými body, jedním v pivotu a druhým na konci kyvadla. S kyvadlem je pevně spojena soustava souřadnic $\vec{x}_1,\ \vec{y}_1,\ \vec{z}_1$, jejíž počátek je shodný s polohou pivotu a směr její osy $\vec{z}_1$ je shodný se směřováním kyvadla. Tato

soustava souřadnic vznikne rotační transformací inerciální soustavy souřadnic $\vec{x}_0$, $\vec{y}_0$, $\vec{z}_0$, neboli soustavy souřadnic pevně spojené se zemí. Soustava souřadnic pevně spojená s tělesem tedy vyjadřuje odklon kyvadla od vzpřímené polohy. Tento odklon je vyjádřen rotací kolem osy $\vec{x}_0$ o úhel α a rotací kolem osy $\vec{y}_0$ o úhel β (obrázky 2.0.3 a 2.0.4). Vzhledem k uspořádání systému nejsme schopni působit na rotaci míče kolem osy $\vec{z}_0$. Pokud však budeme začínat z klidové polohy, nemělo by k ní teoreticky docházet. V praxi nejspíš k této rotaci dojde kvůli tření mezi hrotem a míčem, nicméně pouze v malé míře a neměla by tedy ovlivňovat chování systému, proto jí nebudeme uvažovat. Zbylé dvě rotace kolem os $\vec{x}_0$ a $\vec{y}_0$ můžeme reprezentovat následujícími rotačními maticemi.

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix} \tag{2.0.2a}$$

$$R_y(\beta) = \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix} \tag{2.0.2b}$$

Spojením těchto dvou rotací poté obdržíme výsledné natočení kyvadla. Výslednou rotační matici získáme vynásobením matic elementárních rotací.

$$R = R_x(\alpha)R_y(\beta) = \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ \sin(\alpha)\sin(\beta) & \cos(\alpha) & -\sin(\alpha)\cos(\beta) \\ -\cos(\alpha)\sin(\beta) & \sin(\alpha) & \cos(\alpha)\cos(\beta) \end{bmatrix} \tag{2.0.3}$$

Pomocí této matice dále můžeme přecházet mezi inerciální a pohyblivou soustavou souřadnic následujícím způsobem:

$$\vec{x}_0 = R\vec{x}_1 \tag{2.0.4a}$$

$$\vec{x}_1 = R^{-1}\vec{x}_0 = R^T\vec{x}_0 \tag{2.0.4b}$$

Nyní se vrátíme k modelu kyvadla z obrázku 2.0.2. Pro jednotlivé vektory platí následující vztahy:

$$\vec{r}_G = \vec{u} - \vec{a} \tag{2.0.5a}$$

$$\vec{v} = \vec{b} - \vec{a} \tag{2.0.5b}$$

$$\vec{F}_\mu = \mu\vec{g} + \vec{f} \tag{2.0.5c}$$

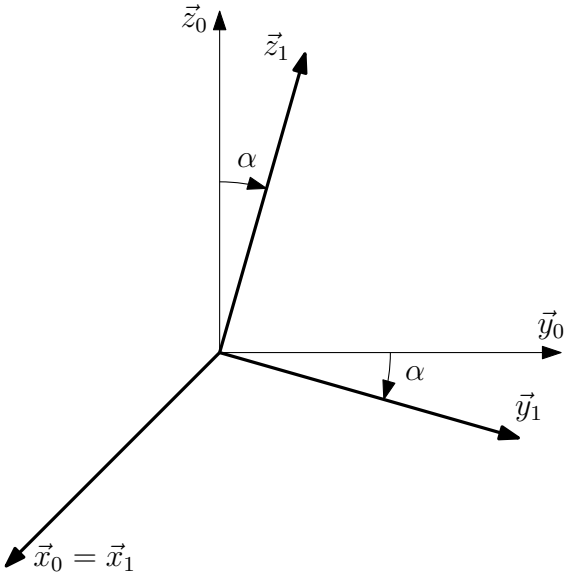
$$\vec{F}_m = m\vec{g} \tag{2.0.5d}$$

$$\vec{a} = -\frac{m}{m+\mu}\vec{v} \tag{2.0.5e}$$

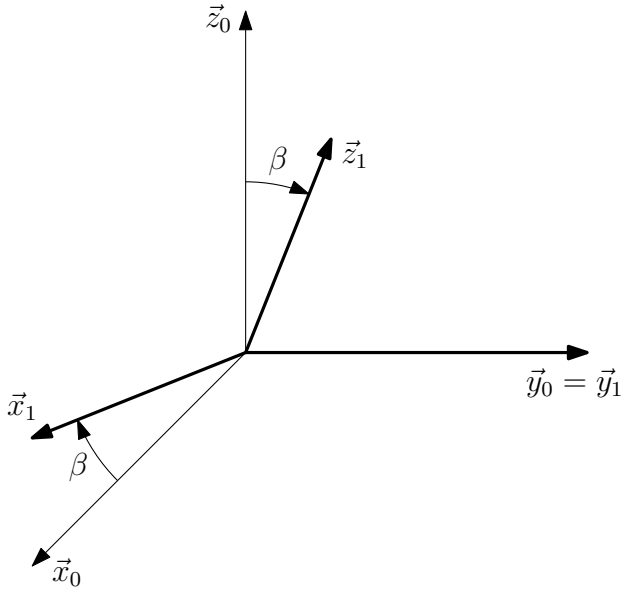
$$\vec{b} = \frac{\mu}{m+\mu}\vec{v} \tag{2.0.5f}$$

Vyjádříme vektor polohy těžiště vzhledem k inerciální soustavě souřadnic $\vec{r}_G$.

$$\vec{r}_G = \vec{u} + \frac{m}{m+\mu}\vec{v} \tag{2.0.6}$$



Obrázek 2.0.3: Rotace kolem osy $\vec{x}_0$ o úhel α



Obrázek 2.0.4: Rotace kolem osy $\vec{y}_0$ o úhel β

Podle Newton-Eulerovy metody pro soustavu hmotných bodů platí:

$$\vec{F} \triangleq \sum_i \vec{F}_i = m_\Sigma \ddot{\vec{r}}_G \tag{2.0.7a}$$

$$\vec{M}_G \triangleq \sum_i \vec{M}_{iG} = \sum_i \vec{\rho}_i \times \vec{F}_i = \frac{d}{dt}\left[\sum_i \left(\vec{\rho}_i \times m_i \dot{\vec{\rho}}_i\right)\right] \tag{2.0.7b}$$

První rovnice popisuje pohyb těžiště soustavy. Druhá rovnice poté popisuje rotační pohyb soustavy kolem těžiště. Nyní do těchto rovnic dosadíme z předchozích vztahů pro jednotlivé vektory a upravíme.

$$\sum_i \vec{F}_i = m_\Sigma \ddot{\vec{r}}_G \tag{2.0.8a}$$

$$\vec{F}_\mu + \vec{F}_m = (m+\mu)\vec{g} + \vec{f} = (m+\mu)\frac{d^2}{dt^2}\left(\vec{u} + \frac{m}{m+\mu}\vec{v}\right) \tag{2.0.8b}$$

$$\vec{M}_G = \frac{d}{dt}\vec{L}_G \tag{2.0.9a}$$

$$\vec{a} \times \vec{F}_\mu + \vec{b} \times \vec{F}_m = \frac{d}{dt}\left(\vec{a} \times \mu\dot{\vec{a}} + \vec{b} \times m\dot{\vec{b}}\right) \tag{2.0.9b}$$

$$\vec{a} \times \left(\mu\vec{g} + \vec{f}\right) + \vec{b} \times m\vec{g} = \vec{a} \times \mu\ddot{\vec{a}} + \vec{b} \times m\ddot{\vec{b}} \tag{2.0.9c}$$

Nyní provedeme limitní přechod, kdy budeme uvažovat, že hmotnost hmotného bodu v pivotu je zanedbatelná, tedy $\mu = 0$. Vztahy pro jednotlivé vektory přejdou na následující tvar. Pro $\mu = 0$:

$$\vec{F}_\mu = \vec{f} \tag{2.0.10a}$$

$$\vec{a} = -\vec{v} \tag{2.0.10b}$$

$$\vec{b} = 0 \tag{2.0.10c}$$

$$\vec{r}_G = \vec{u} + \vec{v} \tag{2.0.10d}$$

Dosadíme do rovnic (2.0.8) a (2.0.9).

$$m\vec{g} + \vec{f} = m\frac{d^2}{dt^2}\left(\vec{u} + \vec{v}\right) \tag{2.0.11a}$$

$$-\vec{v} \times \vec{f} = 0 \tag{2.0.11b}$$

Vektorově přenásobíme rovnici (2.0.11a) z levé strany vektorem $\vec{v}$.

$$\vec{v} \times m\vec{g} + \vec{v} \times \vec{f} = \vec{v} \times m\frac{d^2}{dt^2}\left(\vec{u} + \vec{v}\right) \tag{2.0.12}$$

Dosadíme ze vztahu (2.0.11b) a vyjádříme vektor $\vec{v}$ v transformované soustavě souřadnic jako $R\vec{v}_1$.

$$R\vec{v}_1 \times m\vec{g} = R\vec{v}_1 \times m\frac{d^2}{dt^2}\left(\vec{u} + R\vec{v}_1\right) \tag{2.0.13}$$

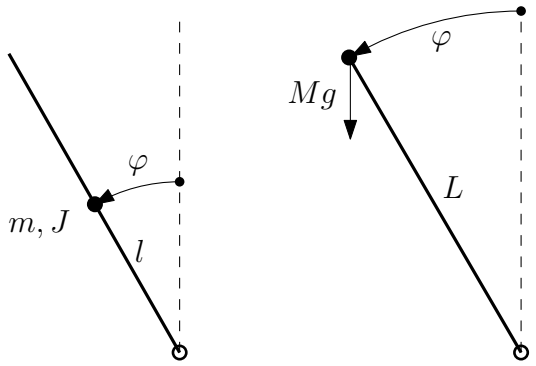
Vzhledem k tomu, že vektor $\vec{v}_1$ vyjadřuje polohu hmotného bodu na konci kyvadla vzhledem k pohyblivé soustavě souřadnic, bude vypadat následovně:

$$\vec{v}_1 = \begin{bmatrix} 0 & 0 & l \end{bmatrix}^T \tag{2.0.14}$$

Parametr l zde vyjadřuje délku kyvadla. Tato délka se s časem samozřejmě nemění, rovnici (2.0.13) tedy můžeme vyjádřit takto.

$$R\vec{v}_1 \times m\vec{g} = R\vec{v}_1 \times m\left(\ddot{\vec{u}} + \ddot{R}\vec{v}_1\right) \tag{2.0.15}$$

Tímto jsme tedy obdrželi popis systému, který je řízen zrychlením pivotu $\ddot{\vec{u}}$. Při odvozování tohoto předpisu jsme uvažovali matematické kyvadlo, které však plně neodpovídá realitě. Pro přechod k fyzickému kyvadlu musíme určit ekvivalentní délku matematického kyvadla. Budeme uvažovat fyzické kyvadlo se vzdáleností těžiště od pivotu l, hmotností m a momentem setrvačnosti kolem těžiště J. Dále uvažujme matematické kyvadlo o délce L a hmotnosti hmotného bodu M a stejné naklonění obou kyvadel φ, což je znázorněno na následujícím obrázku (obrázek 2.0.5).



Obrázek 2.0.5: Schéma fyzického (vlevo) a matematického (vpravo) kyvadla

Moment setrvačnosti matematického kyvadla kolem pivotu spočteme jako

$$J_0 = ML^2 \tag{2.0.16}$$

Pro fyzické kyvadlo známe moment setrvačnosti kolem těžiště J a vzdálenost těžiště od pivotu l. Odtud můžeme získat moment setrvačnosti kyvadla kolem pivotu podle Steinerovy věty:

$$J_0 = J + ml^2 \tag{2.0.17}$$

Pokud požadujeme, aby byla obě kyvadla ekvivalentní, musí se jejich moment setrvačnosti okolo pivotu shodovat, tedy

$$J_0 = J + ml^2 = ML^2 \tag{2.0.18}$$

Z druhého Newtonova pohybového zákona pro matematické kyvadlo plyne

$$ML^2\ddot{\varphi} = MLg \cdot \sin(\varphi) \tag{2.0.19a}$$
$$L\ddot{\varphi} = g \cdot \sin(\varphi) \tag{2.0.19b}$$

Obdobně pro fyzické kyvadlo platí

$$(J + ml^2)\ddot{\varphi} = mlg \cdot \sin(\varphi) \tag{2.0.20a}$$
$$\frac{J + ml^2}{lm}\ddot{\varphi} = g \cdot \sin(\varphi) \tag{2.0.20b}$$

Je vidět, že pravé strany rovnic obou kyvadel jsou si rovny a v obou rovnicích se na levé straně vyskytuje člen $\ddot{\varphi}$. Pro platnost rovnice (2.0.18) se tedy musí rovnat zbylé členy na levých stranách a redukovaná délka matematického kyvadla je tím pádem dána následujícím předpisem.

$$L = \frac{J + ml^2}{ml} \tag{2.0.21}$$

Takto spočtenou délku matematického kyvadla nyní dosadíme do vektoru $\vec{v}_1$ z rovnice (2.0.14).

$$\vec{v}_1 = \begin{bmatrix} 0 & 0 & \frac{J+ml^2}{ml} \end{bmatrix}^T \tag{2.0.22}$$

Dosazením tohoto upraveného vektoru do rovnice (2.0.15) získáme pohybovou rovnici pro fyzické kyvadlo. Dále zatím budeme počítat s tímto fyzickým kyvadlem a s pro něj odvozenými vztahy. Pokud bychom chtěli přejít k míči, nahradili bychom moment setrvačnosti J vyskytující se v předchozích vztazích momentem setrvačnosti míče (duté kulovité slupky o určité tloušťce), který je uveden v rovnici (2.0.1).

Do výsledné rovnice (2.0.15) nyní dosadíme již dříve určenou matici rotace (2.0.3), vektor $\vec{v}_1$ určený pro fyzické kyvadlo (2.0.22) a dále vektor gravitačního zrychlení

$$\vec{g} = \begin{bmatrix} 0 & 0 & -g \end{bmatrix}^T \tag{2.0.23}$$

a vektor řízení. Řízení budeme uvažovat ve formě zrychlení pivotu ve směru jednotlivých souřadnic, vypustíme tedy druhou derivaci v označení a přejdeme na označení $\ddot{\vec{u}} \Rightarrow \vec{u}$. Vektor řízení je poté

$$\vec{u} = \begin{bmatrix} u_x & u_y & u_z \end{bmatrix}^T \tag{2.0.24}$$

kde jednotlivé složky jsou zrychlení pivotu ve směrech jednotlivých souřadnic. Vektorový součin, který se v rovnici (2.0.15) vyskytuje, můžeme vypočítat následujícím způsobem:

$$a \times b = \hat{a}b \tag{2.0.25a}$$

$$a = \begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix}^T \tag{2.0.25b}$$

$$\hat{a} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \tag{2.0.25c}$$

Dosazením do rovnice a následným roznásobením jsme obdrželi tři rovnice pro dvě neznámé, jedná se tedy o přeurčenou soustavu rovnic. Takováto soustava rovnic nemá řešení, pokud některé z rovnic nejsou lineární kombinací ostatních, což je přesně náš případ, neboť při výpočtu řešení z kterékoliv kombinace dvou rovnic jsme obdrželi stejný výsledek. Jako stav systému jsme zvolili úhel natočení kyvadla kolem osy $\vec{x}_0$ (úhel α), úhel natočení kyvadla kolem osy $\vec{y}_0$ (úhel β) a poté rychlosti těchto rotací ($\dot{\alpha}$ a $\dot{\beta}$).

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} \alpha \\ \beta \\ \dot{\alpha} \\ \dot{\beta} \end{bmatrix} \tag{2.0.26}$$

Výsledkem je poté následující nelineární t-variantní systém:

$$\dot{x}_1 = x_3 \tag{2.0.27a}$$
$$\dot{x}_2 = x_4 \tag{2.0.27b}$$
$$\dot{x}_3 = f_3(x, u, t) \tag{2.0.27c}$$
$$\dot{x}_4 = f_4(x, u, t) \tag{2.0.27d}$$

Z hlediska úspory místa zavedeme následující značení.

$$\cos(x) = c_x \tag{2.0.28a}$$
$$\sin(x) = s_x \tag{2.0.28b}$$

Stavové rovnice systému nám vyšly následovně.

$$\dot{x}_1 = \dot{\alpha} \tag{2.0.29a}$$
$$\dot{x}_2 = \dot{\beta} \tag{2.0.29b}$$
$$\dot{x}_3 = \frac{2s_\beta\dot{\alpha}\dot{\beta}ml^2 + 2Js_\beta\dot{\alpha}\dot{\beta} + c_\alpha mlu_y + s_\alpha gml + s_\alpha mlu_z}{c_\beta(J + ml^2)} \tag{2.0.29c}$$
$$\dot{x}_4 = \frac{-s_\beta c_\beta\dot{\alpha}^2ml^2 - Js_\beta c_\beta\dot{\alpha}^2 + c_\alpha s_\beta glm + c_\alpha s_\beta mlu_z - s_\beta s_\alpha mlu_y - c_\beta mlu_x}{J + ml^2} \tag{2.0.29d}$$

Výstup budeme uvažovat shodný se stavem systému, přímý vliv vstupu na výstup je nulový. Pokud nyní provedeme linearizaci v ustáleném stavu, tedy ve vzpřímené poloze s nulovými vstupy ($x_0 = [0, 0, 0, 0]^T$ a $u_0 = [0, 0, 0]^T$) získáme následující stavový model.

$$\dot{x} = Ax + Bu \tag{2.0.30a}$$
$$y = Cx \tag{2.0.30b}$$
$$\begin{bmatrix}\dot{\alpha}\\ \dot{\beta}\\ \ddot{\alpha}\\ \ddot{\beta}\end{bmatrix} = \begin{bmatrix}0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\\ \frac{gml}{J+ml^2} & 0 & 0 & 0\\ 0 & \frac{gml}{J+ml^2} & 0 & 0\end{bmatrix}\begin{bmatrix}\alpha\\ \beta\\ \dot{\alpha}\\ \dot{\beta}\end{bmatrix} + \begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 0 & \frac{ml}{J+ml^2} & 0\\ \frac{ml}{J+ml^2} & 0 & 0\end{bmatrix}\begin{bmatrix}u_x\\ u_y\\ u_z\end{bmatrix} \tag{2.0.30c}$$
$$\begin{bmatrix}y_1\\ y_2\\ y_3\\ y_4\end{bmatrix} = \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\alpha\\ \beta\\ \dot{\alpha}\\ \dot{\beta}\end{bmatrix} \tag{2.0.30d}$$

Pokud se podíváme na výsledné rovnice tohoto stavového modelu, můžeme pozorovat, že odpovídají linearizovaným vztahům pro jednoduché inverzní kyvadlo.

3 Plánování trajektorie

Matematický model kyvadla na hrotu je odvozený, nyní můžeme přistoupit k naplánování trajektorie pohybu. Naším cílem je, aby manipulátor s míčem, případně kyvadlem, na hrotu objel nějakou předem definovanou trajektorii. Pokud bychom pozici koncového efektoru manipulátoru a tím pádem i hrotu s míčem měnili jen pomalu, mohli bychom nejspíše uvažovat, že míč bude po celou dobu pohybu ve vzpřímené poloze a stav bychom se tedy po celou dobu snažili držet nulový. Pokud však budeme chtít dosáhnout nějakého rychlejšího pohybu, je nutné míč naklonit „po směru“ pohybu. Malý moment před započetím pohybu je míč nejprve nutné naklonit ve směru tohoto budoucího pohybu, pokud je poté trajektorie ve tvaru oblouku, je nutné aby byl míč nakloněn dovnitř tohoto oblouku a tím bojoval proti odstředivé síle. Před zastavením je naopak nutné, aby míč byl nakloněn opačně, neboť po zastavení hrotu se míč setrvačnou silou dostane do klidové vzpřímené polohy. Naším úkolem je tedy předpočítat vyhovující trajektorii koncového efektoru, kdy jeho rychlost a zrychlení budou v počátečním a koncovém bodu trajektorie nulové. K takto určenému pohybu koncového efektoru a jemu odpovídajícímu zrychlení, neboli řízení systému, je poté pro každý časový okamžik pohybu podle periody vzorkování nutné spočítat naklonění a okamžitou úhlovou rychlost kyvadla. Toto naklonění a rychlost musí být také na počátku i konci pohybu po trajektorii rovné nule. Musíme tedy pro náš systém vyřešit takzvanou dvoubodovou okrajovou úlohu.

3.1 Dvoubodová okrajová úloha

Pojem okrajová úloha (boundary value problem) se vyskytuje v matematice v oboru diferenciálních rovnic. Obyčejné diferenciální rovnice se vyskytují v matematice, fyzice i inženýrství a popisují děje spojitě se měnící v čase. Sama o sobě má soustava obyčejných diferenciálních rovnic mnoho řešení. Obyčejně je hledané řešení dáno určením hodnot všech jeho komponent v jednom bodě $x = a$. V tomto případě hovoříme o takzvané počáteční úloze. Oproti tomu v okrajové úloze řešíme soustavu obyčejných diferenciálních rovnic spolu s dalšími omezeními, kterým říkáme okrajové podmínky. Okrajová úloha tedy specifikuje hodnoty nebo rovnice pro komponenty řešení ve více bodech x. Okrajová úloha nemusí mít žádné řešení, nebo jich naopak může mít více nebo nekonečně mnoho. Z tohoto důvodu vyžadují programy pro řešení okrajového problému po uživateli, aby zadal odhad pro požadované řešení. Často je součástí řešení také hodnota parametrů, pro které má okrajový problém řešení. Pro tyto parametry je také nutné zadat počáteční odhad. V našem konkrétním případě hledáme řešení soustavy obyčejných diferenciálních rovnic, které bude vyhovovat požadavku na vzpřímenou polohu kyvadla (nulový stav) na začátku i na konci pohybu. Máme tedy dva body x na krajích časového intervalu, na kterém hledáme řešení soustavy obyčejných diferenciálních rovnic. Odtud název dvoubodová okrajová úloha. Obecně můžeme tento problém zapsat jako

$$\dot{y}(x) = f(x, y(x)) \tag{3.1.1a}$$

$$x \in \langle a; b \rangle \tag{3.1.1b}$$

$$g(y(a), y(b)) = 0 \tag{3.1.1c}$$

Funkce g definuje okrajové podmínky. V praxi je někdy třeba řešit i vícebodové okrajové úlohy. Tento problém je poté řešitelný převedením na několik dvoubodových okrajových úloh.

3.1.1 Numerické metody řešení okrajových úloh

Pokud je funkce f hladká na intervalu $\langle a; b\rangle$, daná počáteční úloha $\dot{y} = f(x, y),\ y(a)$ má právě jedno řešení. Jako příklad dvoubodové okrajové úlohy je uvedena rovnice

$$\ddot{y} + y = 0 \tag{3.1.2}$$

s okrajovými podmínkami

$$y(a) = A \tag{3.1.3a}$$
$$y(b) = B \tag{3.1.3b}$$

Důležitým přístupem k analýze takovéto úlohy je uvažovat soubor řešení počátečních úloh. Nechť

$$y(x, s) \tag{3.1.4}$$

je řešením uvedeného příkladu s počátečními hodnotami

$$y(a) = A \tag{3.1.5a}$$
$$\dot{y}(a) = s \tag{3.1.5b}$$

Každé $y(x, s)$ postupuje k $x = b$ a otázkou je, pro které hodnoty s platí

$$y(b, s) = B \tag{3.1.6}$$

Pokud existuje řešení s této algebraické rovnice, poté odpovídající $y(x, s)$ poskytuje řešení diferenciální rovnice, které splňuje okrajové podmínky. Tento teoretický postup je založen na řešení souboru počátečních úloh pro obyčejné diferenciální rovnice a na řešení nelineárních algebraických rovnic. Protože existují efektivní programy pro řešení obou těchto problémů, nabízí se zkombinovat je do programu řešícího okrajovou úlohu. Na této myšlence je založena takzvaná metoda střelby. Pokud jsou počáteční úlohy využívané k řešení stabilní, jedná se o efektivní metodu, pro některé okrajové úlohy však tento fakt neplatí. Potom dochází k situacím, že zatímco řešení okrajové úlohy není citlivé na změny v okrajových hodnotách, řešení počátečních úloh metody střelby jsou naopak na stejné změny velmi citlivá. Nestabilní řešení počátečních úloh poté mohou způsobit neschopnost nalezení přesného řešení, pro obecné použití je tedy tato metoda nevhodná.

Tímto neduhem netrpí kolokační metoda. Mějme dvoubodový okrajový problém.

$$\dot{y} = f(x, y, p) \tag{3.1.7a}$$
$$x \in \langle a; b\rangle \tag{3.1.7b}$$
$$g(y(a), y(b), p) = 0 \tag{3.1.7c}$$

Proměnná p je zde vektor neznámých parametrů, který z dalších rovnic pro zjednodušení vypustíme. Přibližné řešení $S(x)$ je spojitá funkce, která je kubickým polynomem na každém podintervalu $\langle x_n; x_{n+1}\rangle$ mřížky $a = x_0 < x_1 < \cdots < x_N = b$. Tato funkce splňuje okrajovou podmínku

$$g(S(a), S(b)) = 0 \tag{3.1.8}$$

a zároveň splňuje diferenciální rovnice na obou krajích a uprostřed intervalu.

$$\dot{S}(x_n) = f(x_n, S(x_n)) \tag{3.1.9a}$$
$$\dot{S}((x_n + x_{n+1})/2) = f((x_n + x_{n+1})/2, S((x_n + x_{n+1})/2)) \tag{3.1.9b}$$
$$\dot{S}(x_{n+1}) = f(x_{n+1}, S(x_{n+1})) \tag{3.1.9c}$$

Z těchto podmínek dostáváme soustavu nelineárních algebraických rovnic pro koeficienty definující $S(x)$. Na rozdíl od metody střelby je řešení $y(x)$ aproximováno na celém intervalu $\langle a; b\rangle$ a okrajové podmínky jsou uvažovány ve všech časech. Nelineární algebraické rovnice jsou řešeny iterativně pomocí linearizace, používají se tedy funkce pro řešení lineárních rovnic namísto kódů pro řešení počátečních úloh. Okrajové úlohy mohou mít obecně více než jedno řešení, je tedy nutné poskytnout programům pro jejich výpočet předběžný odhad požadovaného řešení, včetně počáteční mřížky, na které ho chceme najít. Řešení okrajových úloh také často zahrnuje nalezení parametrů, pro které existuje řešení a pro které musíme také poskytnout počáteční odhad. Poskytnutí dobrého odhadu patří k nejtěžším úkolům při řešení okrajové úlohy. Pro některé odhady může dojít k singularitám, jindy pro změnu můžeme dostat řešení, které nevyhovuje námi specifikovaným požadavkům. V druhém případě může být vhodné výsledek použít jako počáteční odhad pro další běh programu.

V našem případě použijeme pro výpočet dvoubodové okrajové úlohy program MATLAB a jeho funkci *bvp4c()*. Tato funkce je založena na kolokační metodě, konkrétně na od ní odvozené Simpsonově metodě. Více o okrajových úlohách a implementaci funkce *bvp4c()* pro jejich řešení v MATLABu se lze dočíst například v [7], odkud jsme si vypůjčili uvedenou teorii k tomuto tématu.

3.2 Výpočet trajektorie - řešení dvoubodové okrajové úlohy

Chtěli bychom najít takové řízení, jehož výsledkem by byl pohyb koncového efektoru manipulátoru, potažmo hrotu s míčem/kyvadlem, po uzavřené křivce z klidu do klidu. Na začátku i konci pohybu musí být kyvadlo ve vzpřímené klidové poloze, oba úhly natočení i jejich rychlosti tedy musí být nulové. Požadujeme, aby se koncový efektor na úplném začátku a na úplném konci trajektorie nehýbal, jeho rychlost i zrychlení tedy musí být v těchto okamžicích nulové. Zrychlení odpovídá řízení systému. Uvažujme čas pro vykonání celého pohybu po trajektorii T. Musí platit

$$v(0) = 0 \tag{3.2.1a}$$
$$a(0) = u(0) = 0 \tag{3.2.1b}$$

pro počátek pohybu a

$$v(T) = 0 \tag{3.2.2a}$$
$$a(T) = u(T) = 0 \tag{3.2.2b}$$

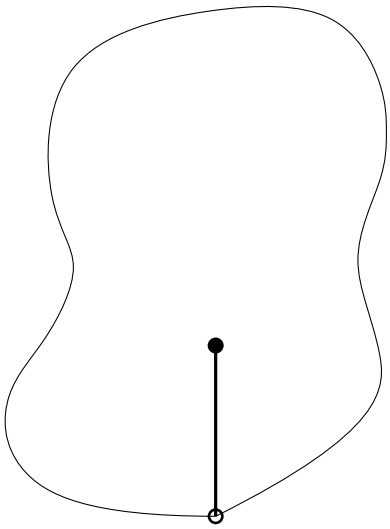
pro konečný časový okamžik pohybu. Dále bychom chtěli, aby trajektorie, kterou bude pivot opisovat, byla uzavřená křivka. Nulová tedy musí být v obou časech i poloha koncového efektoru.

$$s(0) = 0 \tag{3.2.3a}$$
$$s(T) = 0 \tag{3.2.3b}$$

Požadavky jsme uvedli pro vektory pohybu, rychlosti a řízení, které se skládají ze tří složek, jedna složka pro každou osu.

$$s = \begin{bmatrix} s_x & s_y & s_z \end{bmatrix}^T \tag{3.2.4a}$$
$$v = \begin{bmatrix} v_x & v_y & v_z \end{bmatrix}^T \tag{3.2.4b}$$
$$u = \begin{bmatrix} u_x & u_y & u_z \end{bmatrix}^T \tag{3.2.4c}$$

Obecné schéma trajektorie je znázorněno na obrázku 3.2.1.



$s(0) = s(T) = 0$
$v(0) = v(T) = 0$
$u(0) = u(T) = 0$

Obrázek 3.2.1: Obecné schéma požadované trajektorie koncového efektoru s kyvadlem

Naším úkolem je najít funkci, která bude splňovat všechny podmínky. Tato funkce musí být nulová v časech 0 i T a to samé platí i pro její první a druhou derivaci. Uvedené požadavky budou splněny například při použití funkce

$$s_k(t) = 1 - \cos(k\omega t) \tag{3.2.5}$$

Frekvenci ω spočteme jako

$$\omega = \frac{2\pi}{T} \tag{3.2.6}$$

První derivace této funkce je

$$v_k(t) = k\omega \sin(k\omega t) \tag{3.2.7}$$

a pro její druhou derivaci platí

$$a_k(t) = u_k(t) = (k\omega)^2 \cos(k\omega t) \tag{3.2.8}$$

Pro druhou derivaci nejsou podmínky splněny. Můžeme však sečíst několik harmonických složek s vhodnými koeficienty tak, aby požadavky platili. Tímto zároveň neporušíme platnost požadavků pro polohu $s(k)$ a rychlost $v(k)$.

$$u(t) = \sum_{k=1}^{N} c_k (k\omega)^2 \cos(k\omega t) \tag{3.2.9a}$$

$$u(t)|_{t=0} = \sum_{k=1}^{N} c_k (k\omega)^2 = 0 \tag{3.2.9b}$$

$$u(t)|_{t=T} = \sum_{k=1}^{N} c_k (k\omega)^2 = 0 \tag{3.2.9c}$$

Aby se jednotlivé harmonické složky vhodně odečetly, stačí koeficient poslední harmonické vypočíst z ostatních koeficientů.

$$\sum_{k=1}^{N-1} c_k (k\omega)^2 = -c_N (N\omega)^2 \tag{3.2.10a}$$

$$c_N = -\frac{\sum_{k=1}^{N-1} c_k k^2 \omega^2}{N^2 \omega^2} \tag{3.2.10b}$$

$$c_N = -\frac{\sum_{k=1}^{N-1} c_k k^2}{N^2} \tag{3.2.10c}$$

Konkrétně můžeme tento součet harmonických složek provést zvlášť pro řízení v každé ose.

$$u_x(t) = \sum_{k=1}^{N} c_{xk}(k\omega)^2 \cos(k\omega t) \tag{3.2.11a}$$

$$u_y(t) = \sum_{k=1}^{N} c_{yk}(k\omega)^2 \cos(k\omega t) \tag{3.2.11b}$$

$$u_z(t) = \sum_{k=1}^{N} c_{zk}(k\omega)^2 \cos(k\omega t) \tag{3.2.11c}$$

Naším problémem je nyní nalezení koeficientů $c_{xk},\ c_{yk},\ c_{zk};\quad k = 1, \ldots, N$ tak, aby kyvadlo na začátku stálo vzpřímeně

$$x(0) \stackrel{!}{=} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.2.12}$$

a to samé musí platit i na konci pohybu.

$$x(T) \stackrel{!}{=} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.2.13}$$

Musíme tedy vyřešit dvoubodovou okrajovou úlohu. Pro tento účel použijeme funkci *bvp4c()* v programu MATLAB, která nám jako výsledek poskytne hodnotu některých koeficientů funkce řízení (zbytek musíme předem vhodně zvolit) a průběh stavu kyvadla při vykonávání pohybu. Budeme tedy mít k dispozici řídicí funkci a pro každý časový okamžik podle periody vzorkování tomu odpovídající náklon a úhlovou rychlost kyvadla tak, že budou splněny okrajové podmínky.

Při psaní skriptu pro spuštění funkce *bvp4c()* budeme postupovat podle rad a příkladů uvedených v [7]. Prvním krokem je zapsat soustavu obyčejných diferenciálních rovnic jako soustavu obyčejných diferenciálních rovnic prvního řádu. V našem případě můžeme tento krok vynechat, neboť vzhledem k volbě stavových proměnných již máme model v odpovídajícím tvaru (2.0.29). Protože máme čtyři stavové proměnné a určené všechny počáteční podmínky $x(0)$, musíme mít také čtyři volné parametry pro funkci řízení, abychom se byli schopni trefit do požadovaného bodu na konci trajektorie $x(T)$. Ve všech složkách vektoru řízení se tedy musí dohromady vyskytovat čtyři volné parametry a případné koeficienty u ostatních harmonických složek zvolíme předem. Funkci *bvp4c()* musíme poskytnout soustavu obyčejných diferenciálních rovnic, soubor okrajových podmínek a dále počáteční odhad mřížky, na které chceme úlohu řešit, počáteční odhad řešení v bodech této mřížky a odhad volných parametrů. Nakonec můžeme ještě specifikovat další možnosti, jako je například požadovaná relativní tolerance.

Pro simulaci jsme neznali reálné parametry systému, jejich hodnoty jsme se tedy pokusili vhodně zvolit.

$$l = 0.1\ [m] \tag{3.2.14a}$$
$$m = 0.1\ [kg] \tag{3.2.14b}$$
$$g = 9.81\ [m/s^2] \tag{3.2.14c}$$
$$T = 2\ [s] \tag{3.2.14d}$$
$$T_S = 0.01\ [s] \tag{3.2.14e}$$

T_S označuje periodu vzorkování. Moment setrvačnosti vypočítáme podle (2.0.1). Vnější poloměr míče r_2 bude odpovídat vzdálenosti těžiště kyvadla a pivotu l. Tloušťku pláště míče jsme odhadli na $2mm$, tomu tedy bude odpovídat vnitřní poloměr míče r_1.

$$r_2 = 0.1\ [m] \tag{3.2.15a}$$
$$r_1 = 0.098\ [m] \tag{3.2.15b}$$
$$J \doteq 0.00065351\ [kg \cdot m^2] \tag{3.2.15c}$$

Hodnotu frekvence omega vypočteme ze vzorce (3.2.6).

Pro simulaci jsme zkoušeli různě měnit jednotlivé složky vektoru vstupu, dobu pohybu T a s ní související mřížku, odhady řešení i odhady parametrů. Bohužel i při velmi malé změně jednoho čísla můžeme obdržet naprosto rozdílné řešení a nelze vysledovat, jak je řešení ovlivněno změnou konkrétního parametru. Pro některé kombinace parametrů může dojít k singularitě a řešení nenalezneme vůbec. Mohlo by se také stát, že bychom našli řešení, které sice vyhovuje okrajovým podmínkám, ale někde v průběhu by kyvadlo spadlo dolů a následně by se opět vyhouplo. Nezbylo nám tedy nic jiného než metodou pokus-omyl zkoušet měnit různé parametry, dokud jsme neobdrželi nějaké rozumné řešení, které by nám mohlo vyhovovat pro testování na reálném systému. V dokumentaci pro náš manipulátor Stäubli TX40 (**Dodatek 1**) jsou uvedené pracovní prostory a maximální úhlové rychlosti jednotlivých kloubů. U výsledné trajektorie tak můžeme ověřit alespoň to, jestli není naprosto mimo rozsah manipulátoru. Pro další důkaz realizovatelnosti trajektorie však bude potřeba alespoň simulace na modelu robotu. V tomto okamžiku jsme tedy schopni vyloučit jen naprosto nevyhovující trajektorie. Momentálně nám však toto nemusí vadit a až se dostaneme k simulaci, případně navrhneme jinou trajektorii. Pro potřeby simulace jsme tedy zvolili následující funkci řízení se čtyřmi neznámými parametry.

$$u_x = 0.1\omega^2 \cos(\omega t) + p_1(2\omega)^2 \cos(2\omega t) + p_2(3\omega)^2 \cos(3\omega t) - \frac{0.1 + p_1 2^2 + p_2 3^2}{4^2}(4\omega)^2 \cos(4\omega t) \tag{3.2.16a}$$
$$u_y = p_3\omega^2 \cos(\omega t) + p_4(2\omega)^2 \cos(2\omega t) - \frac{p_3 + p_4 2^2}{3^2}(3\omega)^2 \cos 3\omega t \tag{3.2.16b}$$
$$u_z = 0.1\omega^2 \cos(\omega t) - \frac{0.1}{2^2}(2\omega)^2 \cos(2\omega t) \tag{3.2.16c}$$

Proměnné $p_1,\ p_2,\ p_3,\ p_4$ jsou neznámé parametry. Další koeficienty jsme zvolili shodně jako 0.1 a koeficient nejvyšší harmonické pro každou složku vektoru řízení byl vypočten

podle vzorce (3.2.10c). Mřížka pro výpočet je dána časem pohybu T a periodou vzorkování T_S. Odhad řešení jsme zvolili konstantní pro všechny body mřížky.

$$\hat{y} = \begin{bmatrix} 0.015 & 0.015 & 0 & 0 \end{bmatrix}^T \tag{3.2.17}$$

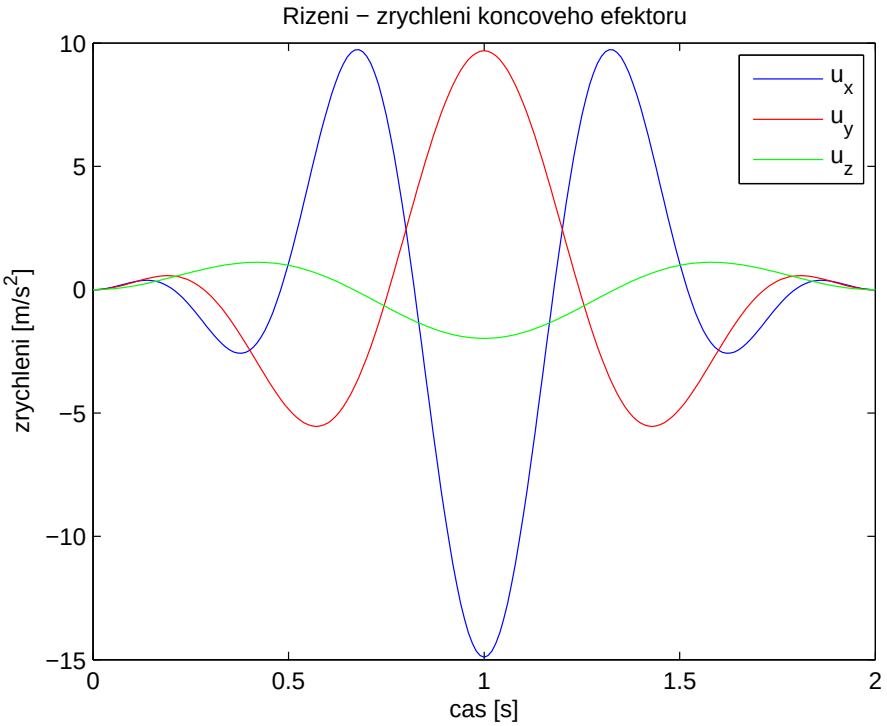
Odhad parametrů jsme nastavili jako

$$\hat{p} = \begin{bmatrix} 0.4 & 0.4 & 0.4 & 0.4 \end{bmatrix}^T \tag{3.2.18}$$

Relativní toleranci pro výpočet jsme nastavili na $1e^{-10}$, požadujeme tedy vysokou přesnost. Ukázka programu je k vidění v **Dodatku 2**. Při psaní skriptu jsme se drželi značení používaného v [7] i v nápovědě programu MATLAB k funkci *bvp4c()*. Pokud spustíme simulaci s tímto nastavením, obdržíme řešení dané dvoubodové okrajové úlohy. Z výsledných volných parametrů

$$p = \begin{bmatrix} -0.1080 & 0.0728 & -0.2202 & 0.1227 \end{bmatrix}^T \tag{3.2.19}$$

můžeme po dosazení do rovnic (3.2.16) určit přesný průběh požadovaného řízení. Dále budeme výsledky reprezentovat graficky. Průběh řízení je vykreslen v obrázku 3.2.2.



Obrázek 3.2.2: Řízení systému ve formě zrychlení koncového efektoru manipulátoru

Z grafu je vidět, že podmínka na nulové zrychlení pivotu na začátku i konci pohybu je splněna. Ověříme ještě splnění podmínek na rychlost a polohu pivotu. Zrychlení je derivací rychlosti a druhou derivací polohy, tyto funkce tedy budou ve tvaru

$$v_x = 0.1\omega \sin(\omega t) + p_1(2\omega)\sin(2\omega t) + p_2(3\omega)\sin(3\omega t) - \frac{0.1 + p_1 2^2 + p_2 3^2}{4^2}(4\omega)\sin(4\omega t) \tag{3.2.20a}$$

$$v_y = p_3\omega \sin(\omega t) + p_4(2\omega)\sin(2\omega t) - \frac{p_3 + p_4 2^2}{3^2}(3\omega)\sin 3\omega t \tag{3.2.20b}$$

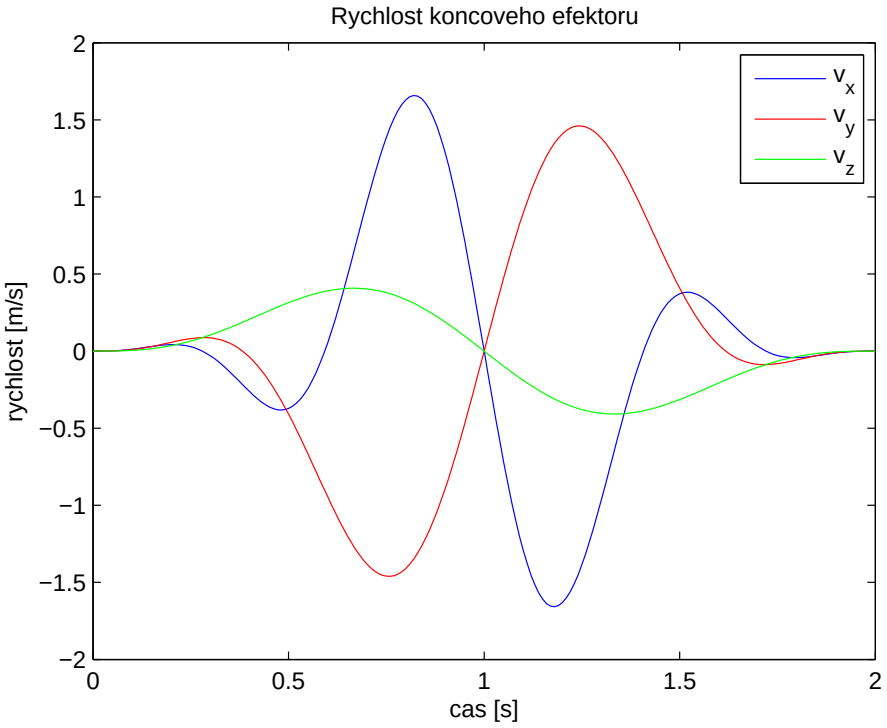
$$v_z = 0.1\omega \sin(\omega t) - \frac{0.1}{2^2}2\omega \sin(2\omega t) \tag{3.2.20c}$$

$$s_x = 0.1(1 - \cos(\omega t)) + p_1(1 - \cos(2\omega t)) + p_2(1 - \cos(3\omega t)) - \frac{0.1 + p_1 2^2 + p_2 3^2}{4^2}(1 - \cos(4\omega t)) \tag{3.2.21a}$$

$$s_y = p_3(1 - \cos(\omega t)) + p_4(1 - \cos(2\omega t)) - \frac{p_3 + p_4 2^2}{3^2}(1 - \cos 3\omega t) \tag{3.2.21b}$$

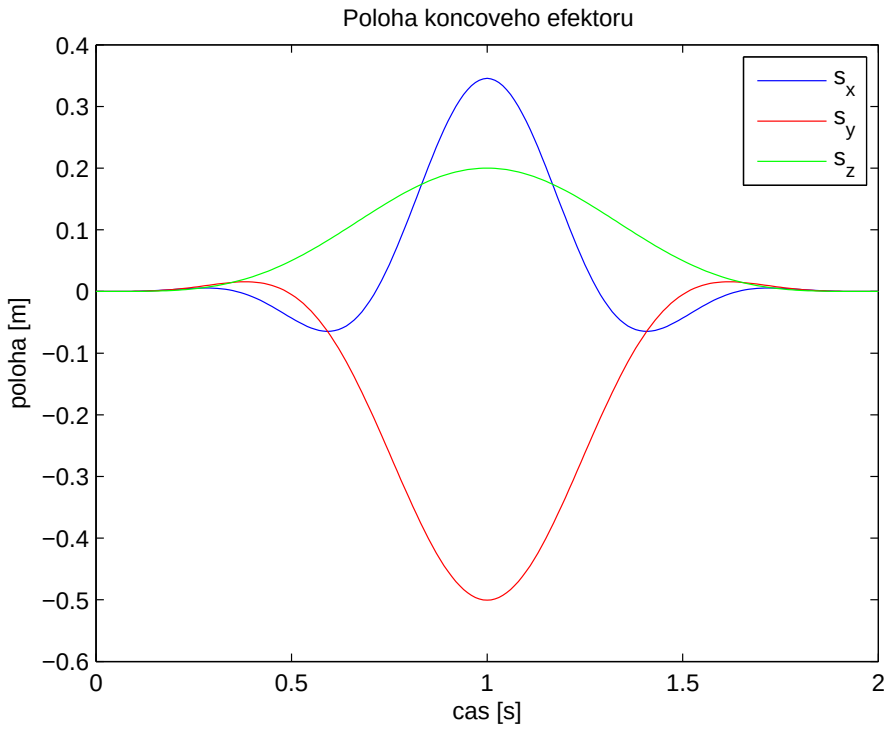
$$s_z = 0.1(1 - \cos(\omega t)) - \frac{0.1}{2^2}(1 - \cos(2\omega t)) \tag{3.2.21c}$$

Rychlost a polohu vykreslíme do grafů (obrázky 3.2.3 a 3.2.4) a ověříme splnění požadavků.

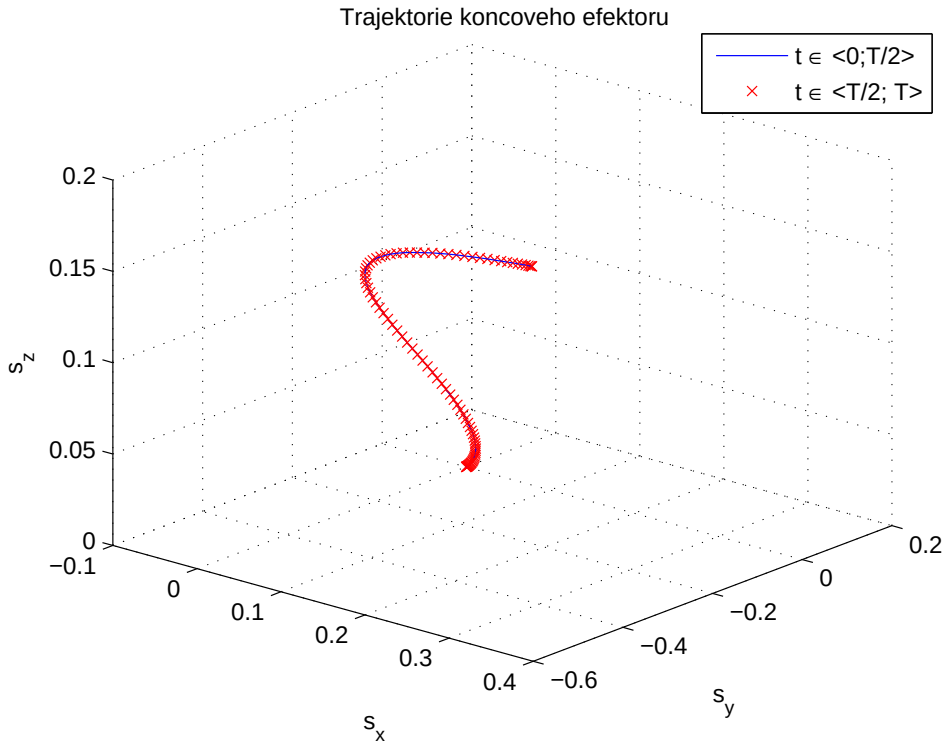


Obrázek 3.2.3: Vývoj rychlosti koncového efektoru

Můžeme pozorovat, že podmínky na počátku a konci pohybu opravdu platí. Z takto vykresleného grafu polohy si nejspíš obrázek o trajektorii manipulátoru neuděláme, pokusíme se tedy pohyb v jednotlivých osách vykreslit do prostoru (obrázek 3.2.5).

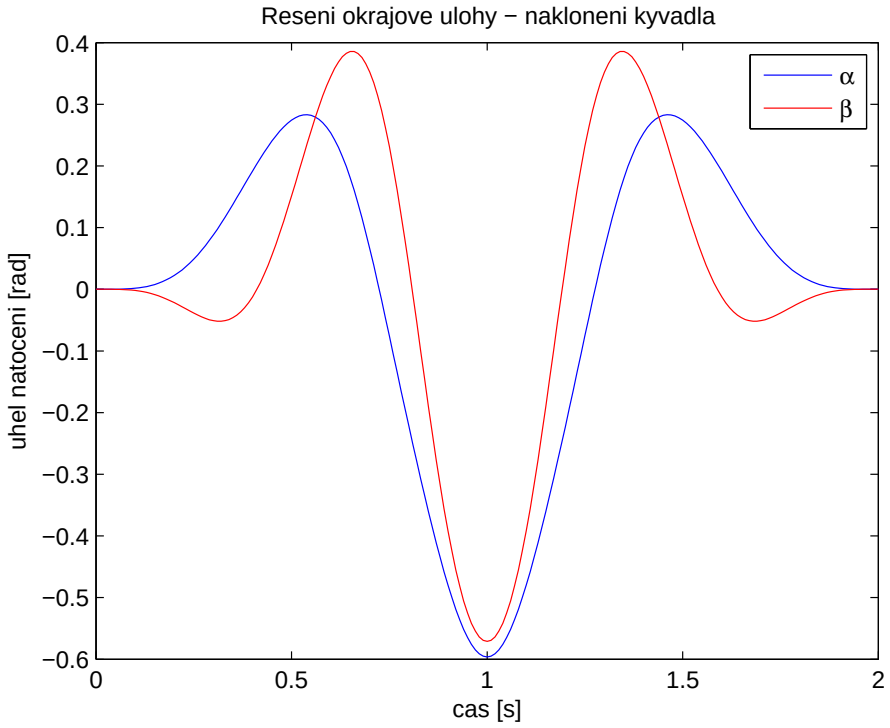


Obrázek 3.2.4: Vývoj polohy koncového efektoru



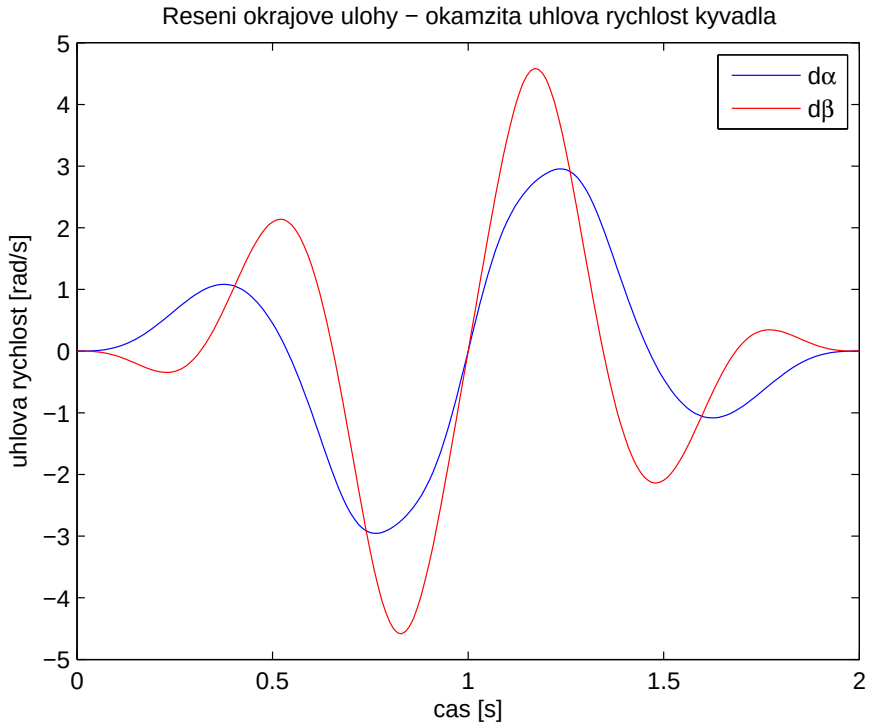
Obrázek 3.2.5: Trajektorie koncového efektoru

Bohužel, na papír se prostorová křivka vykresluje špatně, nicméně graf nám dává představu o podobě trajektorie. Při definování dvoubodové okrajové úlohy jsme požadovali uzavřenou trajektorii. Tu jsme sice obdrželi, nicméně ne v podobě, kterou jsme si představovali. Doufali jsme, že bychom ideálně mohli získat trajektorii ve tvaru jakési „zprohýbané“ kružnice (jako v obrázku 3.2.1). Funkce pro výpočet dvoubodové okrajové úlohy nám však vrátila uzavřenou trajektorii ve formě pohybu po křivce a následném pohybu po stejné křivce opačným směrem. Tento průběh trajektorie je vidět i z grafu (obrázek 3.2.5), kde první půlka času je znázorněna modrou čarou a druhá půlka je znázorněna červenými body. Jak je vidět, tyto dvě části se překrývají. Zmíněné chování se dá odhadnout i ze souměrnosti předchozích grafů. Získané řešení není ideální, nicméně naše požadavky splňuje a výrazně lepší se nám nalézt nepodařilo. Co se týče realizovatelnosti trajektorie, v materiálech k robotu (**Dodatek 1**)je uveden maximální dosah $515mm$. Ten však platí pro koncový efektor, který bude v našem případě neustále směřovat přímo vzhůru vzhledem k pevné soustavě souřadnic, zajímá nás tedy údaj pro předcházející kloub $450mm$. To je maximální poloměr kružnice, kterou je robot schopen opsat. V grafu polohy (obrázek 3.2.4) největší rozdíl polohy v jedné ose necelých $0.6m$, při vhodném nastavení počáteční polohy by se tedy tato trajektorie mohla do pracovního prostoru vejít i s rezervou pro odchylky. Zároveň se kvůli konstrukčním omezením manipulátor nemůže při tomto pohybu dostat do pozice blízko středu kružnice (poloměr $< 151mm$). Pokud k tomuto problému dojde, zjistíme až při simulaci s modelem manipulátoru. Zbývá vykreslit vypočtené průběhy stavu kyvadla (obrázky 3.2.6 a 3.2.7).



Obrázek 3.2.6: Průběh náklonu kyvadla

Opět vidíme, že požadavky na vzpřímenou polohu kyvadla na začátku i konci pohybu jsou splněny.



Obrázek 3.2.7: Průběh úhlové rychlosti kyvadla

4 Návrh regulátoru pro sledování trajektorie

V tomto okamžiku již máme k dispozici požadované řízení a požadovanou trajektorii stavu, které budeme chtít sledovat. Inverzní kyvadlo je ve vzpřímené poloze nestabilní. Kyvadlo tedy bude držet nahoře pouze pro nulové počáteční podmínky a nulový vstup. Toto navíc platí pouze pro model, protože jsme zanedbali vnější vlivy. To znamená, že pokud bychom do systému pustili požadované řízení, kyvadlo by po chvíli spadlo. Tím se dostáváme k návrhu regulátoru, který udrží kyvadlo ve vzpřímené poloze a zajistí sledování požadované trajektorie. Konkrétně budeme používat lineárně-kvadratický regulátor (LQR).

4.1 LQ regulátor

Lineárně-kvadratický regulátor je optimálním řešením takzvané lineárně-kvadratické úlohy. Jedná se o úlohu, kdy pro lineární systém

$$\dot{x} = Ax + Bu \tag{4.1.1}$$

se známou počáteční podmínkou x_0 hledáme optimální řízení definované v čase $t \in \langle t_1; t_2 \rangle$, které minimalizuje index kvality v podobě kvadratické ztrátové funkce

$$J = x^T(t_1)Fx(t_1) + \int_{t_0}^{t_1} \left(x^T Q x + u^T R u\right) dt \tag{4.1.2}$$

kde matice Q, R a F jsou návrhové parametry.

- $Q = Q^T \succeq 0$ ... pozitivně semidefinitní matice penalizující stav
- $R = R^T \succ 0$ ... pozitivně definitní matice penalizující řízení
- $F \succeq 0$ ... pozitivně semidefinitní matice penalizující koncový stav

Výsledné řešení je optimální ve smyslu kompromisu mezi kvalitou řízení (Q) a cenou (R). Toto optimální řízení je časově proměnná stavová zpětná vazba

$$u = K(t)x \tag{4.1.3a}$$
$$K(t) = -R^{-1}B^T P(t) \tag{4.1.3b}$$

kde matice $P(t)$ je řešením Riccatiho diferenciální rovnice

$$A^T P(t) + P(t)A - P(t)BR^{-1}B^T P(t) + Q = \dot{P}(t) \tag{4.1.4}$$

s okrajovou podmínkou

$$P(t_1) = F \tag{4.1.5}$$

V praktických úlohách návrhu regulačních obvodů můžeme chtít systém řídit na nekonečném časovém horizontu a nechceme penalizovat konkrétní konečný stav, matice F poté bude nulová. Kvadratická ztrátová funkce přejde na tvar

$$J = \int_0^\infty \left(x^T Q x + u^T R u\right) dt \tag{4.1.6}$$

Výsledným optimálním řízením je poté časově invariantní stavová zpětná vazba

$$u = Kx \tag{4.1.7a}$$
$$K = -R^{-1}B^T P \tag{4.1.7b}$$

kde matice P je řešení algebraické Riccatiho rovnice

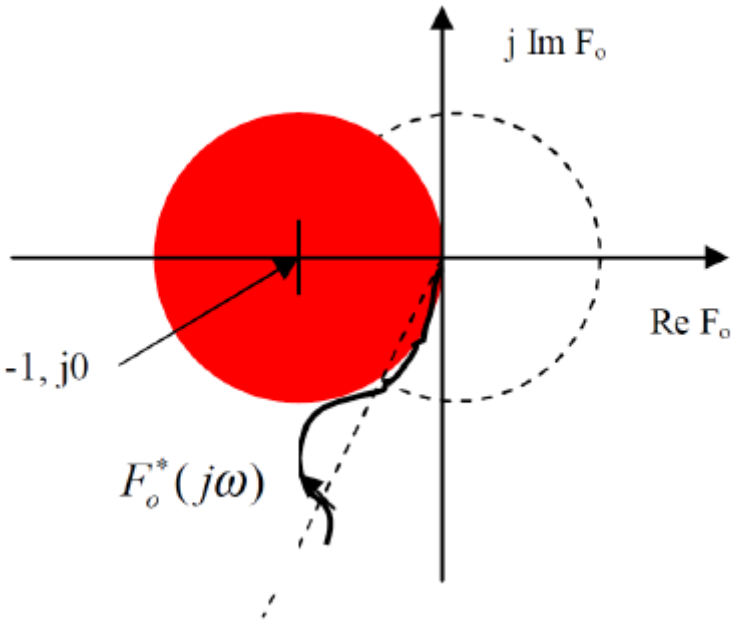
$$A^T P + PA - PBR^{-1}B^T P + Q \tag{4.1.8}$$

Obě tyto formy LQ regulátoru existují v obdobné podobě také pro diskrétní systémy.

Jednou z výhodných vlastností LQ regulátoru je jednoznačné řešení ve formě stavové zpětné vazby. Regulátor poté nezvyšuje řád systému a jeho implementace není složitá. Výsledný regulátor také dosahuje bezpečnosti v zesílení $g_m = \langle \frac{1}{2}, \infty \rangle$ a bezpečnosti ve fázi $\gamma = 60°$, což splňuje požadavky na robustnost regulátoru. Grafické znázornění je na obrázku 4.1.1.

Jak již bylo řečeno, LQ regulátor zajišťuje rozumný kompromis mezi velikostí akčních zásahů a velikostí regulační odchylky. Pro limitní případy dochází k následujícímu chování. Uvažujme matici R jako diagonální matici.

$$R = \rho^2 I \tag{4.1.9}$$



Obrázek 4.1.1: Grafická interpretace robustnosti LQR

V případě $\rho \to \infty$, tedy pokud je řízeni velmi nákladné (expensive control), je optimální řízení z hlediska ceny takové, které pouze „překlopí“ nestabilní póly systému podle imaginární osy do levé poloroviny a stabilní póly nechá být. Pokud naopak $\rho \to 0$, znamená to, že řízení nás nic nestojí (cheap control) a část pólů poté konverguje k nulám podle metody geometrického místa kořenů (GMK) a zbylé póly se blíží nekonečnu po asymptotách odpovídajících Butterworthovo přímkám v komplexní rovině.

Mezi nevýhody LQ regulátoru naopak patří nutnost řešení Riccatiho rovnice a problém volby matic Q a R. Obvykle matice volíme diagonální, čímž redukujeme počet parametrů. Vyšší hodnota prvků v matici Q znamená vyšší penalizaci odpovídajících stavů. To samé platí u matice R v souvislosti s penalizací řízení. U jednotlivých prvků nezáleží na jejich absolutní velikosti, ale na jejich poměru. Pokud tedy obě matice přenásobíme stejným číslem, regulátor se nezmění. Při snaze o splnění návrhových požadavků se poté často jedná o iterativní proces, kdy návrhář zkontroluje výsledky dosažené s vypočteným regulátorem a případně upraví jednotlivé váhy a vypočte nový regulátor.

Informace o této problematice jsme čerpali z [8] a [9]. V těchto zdrojích lze také najít další informace.

4.2 Linearizace systému podél zvolené trajektorie

Jak již bylo uvedeno v předchozí kapitole a jak už i název LQR napovídá, tento regulátor je určen pro lineární systémy. V našem případě stabilizace inverzního kyvadla však chceme řídit nelineární systém. Pokud bychom chtěli míč pouze stabilizovat ve vzpřímené poloze, mohli bychom použít linearizaci v tomto pracovním bodě (viz (2.0.30)). Pokud však chceme sledovat trajektorii získanou řešením dvoubodového okrajového problému, takto linearizovaný model by nám nejspíše nestačil, neboť kyvadlo se při pohybu po této

trajektorii dostane dál mimo vzpřímenou polohu a při těchto větších náklonech by již zmíněný linearizovaný model nefungoval. Proto budeme muset přistoupit k linearizaci podél požadované trajektorie systému. Máme nelineární funkci popisující systém

$$\dot{x} = f\left(x(t), u(t)\right) \tag{4.2.1}$$

a požadované stavy systému a řízení ve všech časových okamžicích trajektorie.

$$\bar{x}(t),\ \bar{u}(t) \tag{4.2.2}$$

Pro tyto stavy a vstupy platí:

$$\dot{\bar{x}}(t) = f\left(\bar{x}(t), \bar{u}(t)\right) \tag{4.2.3}$$

Nyní provedeme Taylorův rozvoj 1. řádu funkce $f(x(t), u(t))$ a všechny vyšší členy zanedbáme.

$$f\left(x(t), u(t)\right) \approx f\left(\bar{x}(t), \bar{u}(t)\right) + \frac{\partial f\left(\bar{x}(t), \bar{u}(t)\right)}{\partial x(t)}\left(x(t) - \bar{x}(t)\right) + \frac{\partial f\left(\bar{x}(t), \bar{u}(t)\right)}{\partial u(t)}\left(u(t) - \bar{u}(t)\right) + \ldots \tag{4.2.4}$$

Zavedeme nové označení členů v této funkci a rovnici přepíšeme.

$$A(t) = \frac{\partial f\left(\bar{x}(t), \bar{u}(t)\right)}{\partial x(t)} \tag{4.2.5a}$$

$$B(t) = \frac{\partial f\left(\bar{x}(t), \bar{u}(t)\right)}{\partial u(t)} \tag{4.2.5b}$$

$$z(t) = x(t) - \bar{x}(t) \tag{4.2.5c}$$

$$v(t) = u(t) - \bar{u}(t) \tag{4.2.5d}$$

$$f\left(x(t), u(t)\right) = f\left(\bar{x}(t), \bar{u}(t)\right) + A(t)z(t) + B(t)v(t) \tag{4.2.5e}$$

Na závěr zapíšeme ve tvaru stavového modelu pro stav $z(t)$.

$$z(t) = x(t) - \bar{x}(t) \tag{4.2.6a}$$

$$\dot{z}(t) = \dot{x}(t) - \dot{\bar{x}}(t) \tag{4.2.6b}$$

$$= f(x(t), u(t)) - f(\bar{x}(t), \bar{u}(t)) \tag{4.2.6c}$$

$$= f(\bar{x}(t), \bar{u}(t)) + A(t)z(t) + B(t)v(t) - f(\bar{x}(t), \bar{u}(t)) \tag{4.2.6d}$$

$$= A(t)z(t) + B(t)v(t) \tag{4.2.6e}$$

Máme tedy hotový linearizovaný t-variantní spojitý model. Pro použití na reálném systému je třeba tuto stavovou rovnici zdiskretizovat. Pro diskretizaci budeme používat vzorkovací periodu T_S. Vstup se poté bude měnit jen v momentech vzorkování a mezi nimi bude konstantní.

$$v(t) = v(kT_S) \triangleq v_k; \quad t \in \langle kT_S; (k+1)T_S) \tag{4.2.7}$$

Obecně platí

$$z(T) = e^{AT}z_0 + \int_0^T e^{A(T-\tau)}Bu(\tau)d\tau \tag{4.2.8a}$$

$$= e^{AT}z_0 + \int_0^T e^{AT}e^{-A\tau}Bv(\tau)d\tau \tag{4.2.8b}$$

$$= e^{AT}z_0 + e^{AT}\int_0^T e^{-A\tau}d\tau Bv_0 \tag{4.2.8c}$$

Pro matice našeho diskrétního t-variantního systému poté budou platit vztahy

$$A_k = e^{A(kT_S)T_S} \tag{4.2.9a}$$

$$B_k = e^{A(kT_S)}\int_{(k-1)T_S}^{kT_S} -e^{-A(\tau)\tau}B(\tau)d\tau \tag{4.2.9b}$$

Tyto vztahy vyjadřují exaktní diskretizaci, my však využijeme jednodušší aproximativní diskretizaci. Derivaci v rovnici (4.2.6e) nahradíme dopřednou diferencí.

$$\dot{z}(t) \approx \frac{z((k+1)T_S) - z(kT_S)}{T_S} \tag{4.2.10a}$$

$$\frac{z((k+1)T_S) - z(kT_S)}{T_S} = A(kT_S)z(kT_S) + B(kT_S)v(kT_S) \tag{4.2.10b}$$

$$z((k+1)T_S) = z(kT_S) + A(kT_S)z(kT_S)T_S + B(kT_S)v(kT_S)T_S \tag{4.2.10c}$$

Tuto rovnici můžeme přepsat do následující podoby

$$z_{k+1} = A_k z_k + B_k v_k \tag{4.2.11}$$

přičemž

$$A_k = I + A(kT_S)T_S \tag{4.2.12a}$$

$$B_k = B(kT_S)T_S \tag{4.2.12b}$$

Nyní máme diskretizovaný systém a můžeme se pustit do návrhu řízení. Protože chceme sledovat trajektorii s přesně daným časem vykonání, použijeme diskrétní verzi LQ regulátoru na konečném horizontu.

4.3 LQR na konečném horizontu pro diskrétní systémy

Budeme se držet postupu uvedeného v [8]. Vyjdeme z rovnice diskretizovaného lineárního systému (4.2.11).

$$V\left(z(t_0), v_{t_0}^{N-1}, t_0\right) = z^T(N)Q(N)z(N) + \sum_{t_0}^{N-1}\begin{bmatrix} z^T(t) & v^T(t)\end{bmatrix}\begin{bmatrix} Q(t) & S(t) \\ S^T(t) & R(t)\end{bmatrix}\begin{bmatrix} z(t) \\ v(t)\end{bmatrix} \tag{4.3.1}$$

$V(z(t), v_t^{N-1}, t)$ je koncová část kriteria optimality. Pokud $t = 0$, zahrnuje koncová část celý časový horizont a je tedy rovna kvadratické ztrátové funkci LQ regulátoru. Tuto hodnotu chceme samozřejmě minimalizovat.

$$J = V\left(z(0), v_0^{N-1}, 0\right) \to \min \tag{4.3.2}$$

Optimální řešení v každém časovém okamžiku t a odpovídajícím stavu $z(t)$ je dáno minimalizací odpovídající koncové části kritéria přes všechny přípustné posloupnosti řízení. Zajímá nás pouze koncová část kritéria, neboť minulost již změnit nemůžeme a hodnota kritéria již může být ovlivněna pouze budoucím řízením.

$$V^*(z(t), t) = \min_{v(.)\in\Omega} V\left(z(t), v_t^{N-1}, t\right) \tag{4.3.3}$$

Optimální řešení budeme hledat pomocí metody dynamického programovaní. Tato metoda je založena na principu Bellmanovy optimalizační rekurze.

$$V^*(z(t), t) = \min_{v(.)\in\Omega} \left\{ \begin{bmatrix} z^T(t) & v^T(t) \end{bmatrix} \begin{bmatrix} Q(t) & S(t) \\ S^T(t) & R(t) \end{bmatrix} \begin{bmatrix} z(t) \\ v(t) \end{bmatrix} + V^*(z(t+1), t+1) \right\} \tag{4.3.4a}$$

$$= \min_{v(.)\in\Omega} \left\{ \begin{bmatrix} z^T(t) & v^T(t) \end{bmatrix} \begin{bmatrix} Q(t) & S(t) \\ S^T(t) & R(t) \end{bmatrix} \begin{bmatrix} z(t) \\ v(t) \end{bmatrix} + V^*(A(t)z(t) + B(t)v(t), t+1) \right\} \tag{4.3.4b}$$

V čase $t = N$ bude platit

$$V^*(z(N), N) = z^T(N)Q(N)z(N) \tag{4.3.5}$$

Nyní dokážeme, že

$$V^*(z(t), t) = z^T(t)P(t)z(t) \tag{4.3.6}$$

Pro $t = N$ tvrzení platí, pokud

$$P(N) = Q(N) \tag{4.3.7}$$

Nyní nechť platí

$$V^*(z(t+1), t+1) = z^T(t+1)P(t+1)z(t+1) \tag{4.3.8}$$

potom podle (4.3.4b) bude platit

$$V^*(z(t),t) = \tag{4.3.9a}$$

$$= \min_{v(.)\in\Omega} \Bigg\{ \begin{bmatrix} z^T(t) & v^T(t) \end{bmatrix} \begin{bmatrix} Q(t) & S(t) \\ S^T(t) & R(t) \end{bmatrix} \begin{bmatrix} z(t) \\ v(t) \end{bmatrix} + \tag{4.3.9b}$$

$$+ (A(t)z(t) + B(t)v(t))^T P(t+1)(A(t)z(t) + B(t)v(t)) \Bigg\} \tag{4.3.9c}$$

$$= \min_{v(.)\in\Omega} \left\{ \begin{bmatrix} z^T(t) & v^T(t) \end{bmatrix} \begin{bmatrix} Q(t) + A^T(t)P(t+1)A(t) & S(t) + A^T(t)P(t+1)B(t) \\ S^T(t) + B^T(t)P(t+1)A(t) & R(t) + B^T(t)P(t+1)B(t) \end{bmatrix} \begin{bmatrix} z(t) \\ v(t) \end{bmatrix} \right\} \tag{4.3.9d}$$

Optimální řízení $u^*(t)$ lze tedy získat v každém kroku minimalizací kvadratické funkce.

$$u^*(t) = -\left[R(t) + B^T(t)P(t+1)B(t)\right]^{-1}\left[S^T(t) + B^T(t)P(t+1)A(t)\right] z(t) \tag{4.3.10}$$

Zisková matice stavového regulátoru je tedy

$$K(t) = -\left[R(t) + B^T(t)P(t+1)B(t)\right]^{-1}\left[S^T(t) + B^T(t)P(t+1)A(t)\right] \tag{4.3.11}$$

Matici P počítáme pomocí zpětné rekurze.

$$P(t) = A^T(t)P(t+1)A(t) + Q(t) + \left[S(t) + A^T(t)P(t+1)B(t)\right] K(t) \tag{4.3.12a}$$

$$P(N) = Q(N) \tag{4.3.12b}$$

Nyní máme k dispozici vztahy pro výpočet LQ regulátoru, stačí nám tedy již vhodně nastavit návrhové parametry, jinými slovy vhodně zvolit matice Q a R. Matici S nebudeme používat, bude tedy nulová. Nastavení vhodných vah patří ke stěžejním problémům návrhu LQ regulátoru a existují pro něj jen velmi obecné poučky. Připomínáme, že v našem případě řešíme problém sledování trajektorie pro systém (4.2.11), kdy stav je $z = x - \bar{x}$ a řízení je $v = u - \bar{u}$. Určitě budeme chtít dělat co nejvíce pro to, abychom ustabilizovali kyvadlo. Řízení nás poté nejspíš bude zajímat jen do té míry, aby bylo na reálném systému fyzikálně realizovatelné. Toto nám však stále neříká mnoho o nastavení jednotlivých vah v maticích LQ regulátoru a pro každý systém může tento předpoklad vyústit v naprosto rozdílné poměry jednotlivých vah. Pokud budeme požadovat velkou přesnost stavu kyvadla, mohlo by se stát, že se koncový efektor manipulátoru bude hodně odchylovat od požadované trajektorie a v rámci snahy o stabilizaci kyvadla pojede po úplně jiné dráze, v krajním případě pak narazí na limity pracovního prostoru. Z tohoto důvodu rozšíříme stav systému o polohu koncového manipulátoru a zavedeme tak i zpětnou vazbu od polohy. Regulátor se poté bude stále soustředit zejména na stabilizaci kyvadla, pokud se však pivot dostane daleko od požadované polohy, začne se tato složka projevovat a regulátor se bude snažit dostat koncový efektor blíže k jeho ideální pozici. Model systému (2.0.29) budeme muset rozšířit o model pohybu pivotu.

$$s = \begin{bmatrix} s_x & s_y & s_z \end{bmatrix}^T \tag{4.3.13a}$$
$$v = \dot{s} \tag{4.3.13b}$$
$$a = \dot{v} = u \tag{4.3.13c}$$

Zrychlení pivotu již v modelu máme ve formě řízení, musíme ho tedy ještě rozšířit o polohu a rychlost pivotu ve všech třech osách.

$$x = \begin{bmatrix} \alpha \\ \beta \\ \dot{\alpha} \\ \dot{\beta} \\ s_x \\ s_y \\ s_z \\ v_x \\ v_y \\ v_z \end{bmatrix} \tag{4.3.14}$$

Model celého systému tedy bude desátého řádu a jeho rovnice budou následující:

$$\dot{x}_1 = \dot{\alpha} \tag{4.3.15a}$$
$$\dot{x}_2 = \dot{\beta} \tag{4.3.15b}$$
$$\dot{x}_3 = \frac{2s_\beta\dot{\alpha}\dot{\beta}ml^2 + 2Js_\beta\dot{\alpha}\dot{\beta} + c_\alpha mlu_y + s_\alpha gml + s_\alpha mlu_z}{c_\beta(J + ml^2)} \tag{4.3.15c}$$
$$\dot{x}_4 = \frac{-s_\beta c_\beta\dot{\alpha}^2ml^2 - Js_\beta c_\beta\dot{\alpha}^2 + c_\alpha s_\beta glm + c_\alpha s_\beta mlu_z - s_\beta s_\alpha mlu_y - c_\beta mlu_x}{J + ml^2} \tag{4.3.15d}$$
$$\dot{x}_5 = v_x \tag{4.3.15e}$$
$$\dot{x}_6 = v_y \tag{4.3.15f}$$
$$\dot{x}_7 = v_z \tag{4.3.15g}$$
$$\dot{x}_8 = u_x \tag{4.3.15h}$$
$$\dot{x}_9 = u_y \tag{4.3.15i}$$
$$\dot{x}_{10} = u_z \tag{4.3.15j}$$

Skript pro výpočet časově proměnné stavové zpětné vazby LQ regulátoru napsaný v programu MATLAB, který navazuje na skript pro řešení dvoubodové okrajové úlohy, je k nahlédnutí v **Dodatku 3**.

4.4 Simulace

V tomto okamžiku máme k dispozici požadovanou trajektorii a umíme k ní vypočítat LQ regulátor, je tedy čas ověřit funkčnost regulátoru simulačně. Jak jsme již psali, pro problém nastavení návrhových parametrů LQ regulátoru neexistuje obecný návod.

Nezbývá nám tedy nic jiného, než použít oblíbenou metodu pokus - omyl. Pro účely simulace jsme tedy po několika pokusech nastavili následující hodnoty.

$$Q = \begin{bmatrix} 100 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 20 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.4.1a}$$

$$R = \begin{bmatrix} 100 & 0 & 0 \\ 0 & 200 & 0 \\ 0 & 0 & 700 \end{bmatrix} \tag{4.4.1b}$$

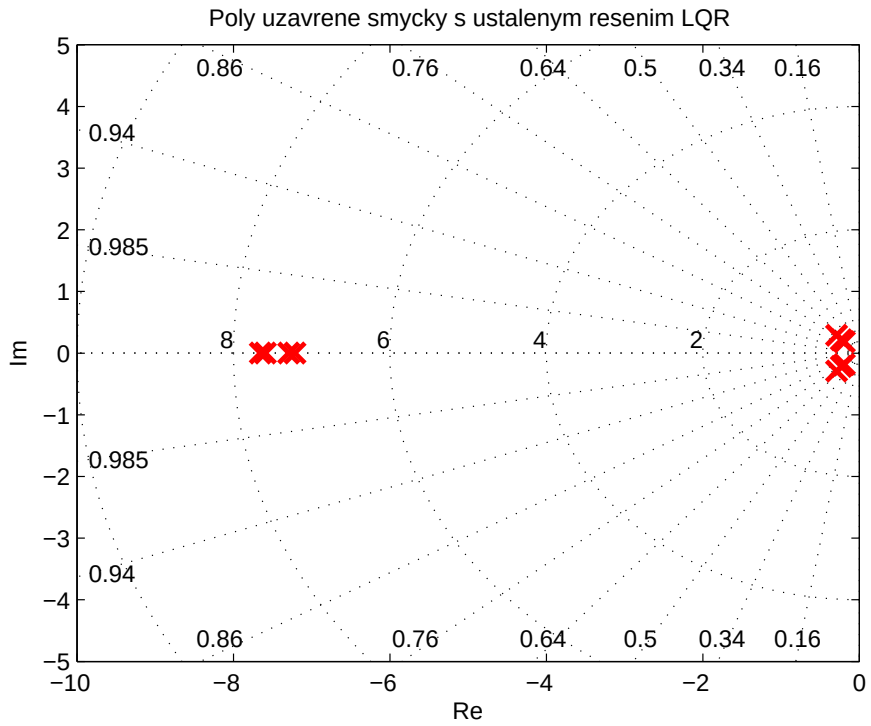
Matice jsme volili diagonální za účelem omezení počtu parametrů. Dále se také dá předpokládat, že pro náklony a úhlové rychlosti kyvadla v obou osách budeme požadovat stejnou penalizaci. Jako výsledek řešení LQ úlohy obdržíme časově proměnnou stavovou zpětnou vazbu, pro každý moment trajektorie podle periody vzorkování máme tedy odpovídající matici zesílení. Po vykonání trajektorie se kyvadlo vrátí do vzpřímené polohy, zpětná vazba poté odpovídá linearizaci v tomto bodě. Tato „poslední“ matice zesílení je tedy ustáleným řešením, které budeme používat k další stabilizaci kyvadla v blízkém okolí vzpřímené polohy po skončení sledování trajektorie. Póly uzavřené smyčky s ustáleným řešením jsou

$$p_1 = -7.2129 + 0.0000i \tag{4.4.2a}$$
$$p_2 = -7.6563 + 0.0000i \tag{4.4.2b}$$
$$p_3 = -7.2917 + 0.0000i \tag{4.4.2c}$$
$$p_4 = -7.5925 + 0.0000i \tag{4.4.2d}$$
$$p_5 = -0.1911 + 0.1841i \tag{4.4.2e}$$
$$p_6 = -0.1911 - 0.1841i \tag{4.4.2f}$$
$$p_7 = -0.2285 + 0.2169i \tag{4.4.2g}$$
$$p_8 = -0.2285 - 0.2169i \tag{4.4.2h}$$
$$p_9 = -0.2913 + 0.2893i \tag{4.4.2i}$$
$$p_{10} = -0.2913 - 0.2893i \tag{4.4.2j}$$

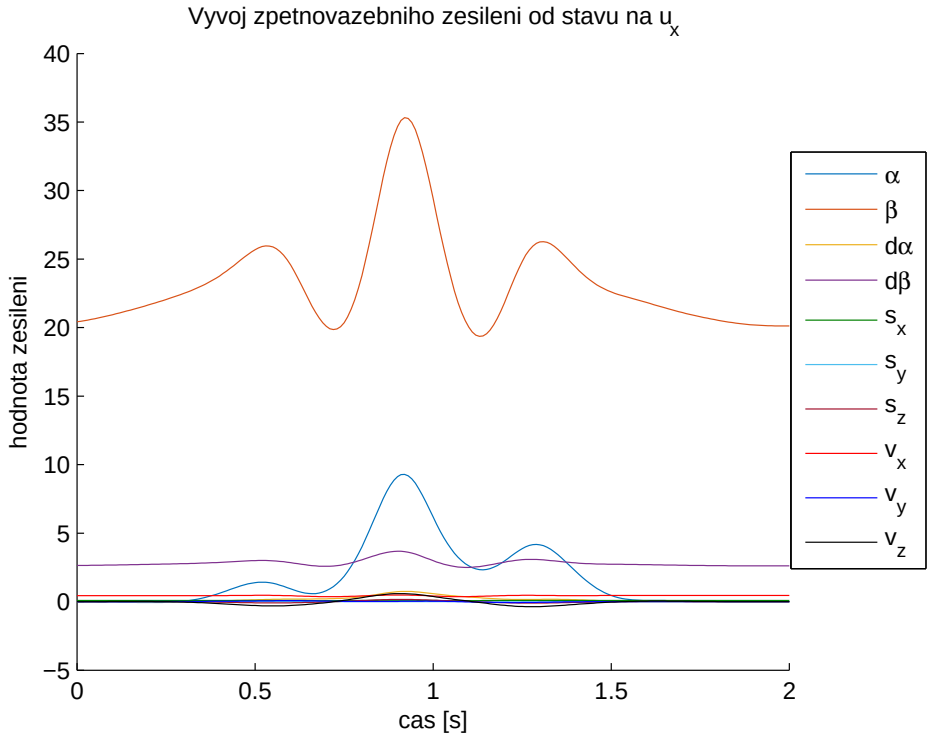
Všechny póly mají zápornou reálnou část, tím pádem jsou stabilní. Zároveň je reálná složka všech pólů větší než jejich imaginární část, všechny póly jsou tedy i dobře zatlumené. Póly můžeme zobrazit v imaginární rovině (obrázek 4.4.1).

Vykreslíme také vývoj hodnot zpětnovazebního zesílení od všech položek stavu na jednotlivé výstupy (obrázky 4.4.2, 4.4.3 a 4.4.4).

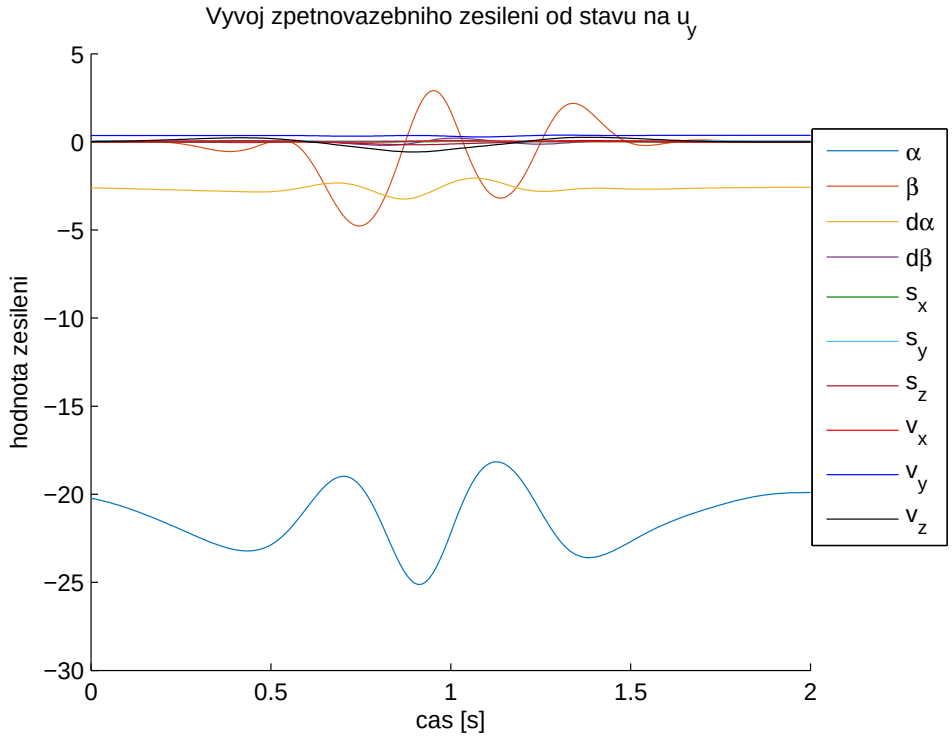
Z grafů je vidět, že zejména zesílení od úhlů náklonu kyvadla se v průběhu trajektorie poměrně výrazně mění. Je to způsobeno velkými náklony kyvadla v průběhu požadované



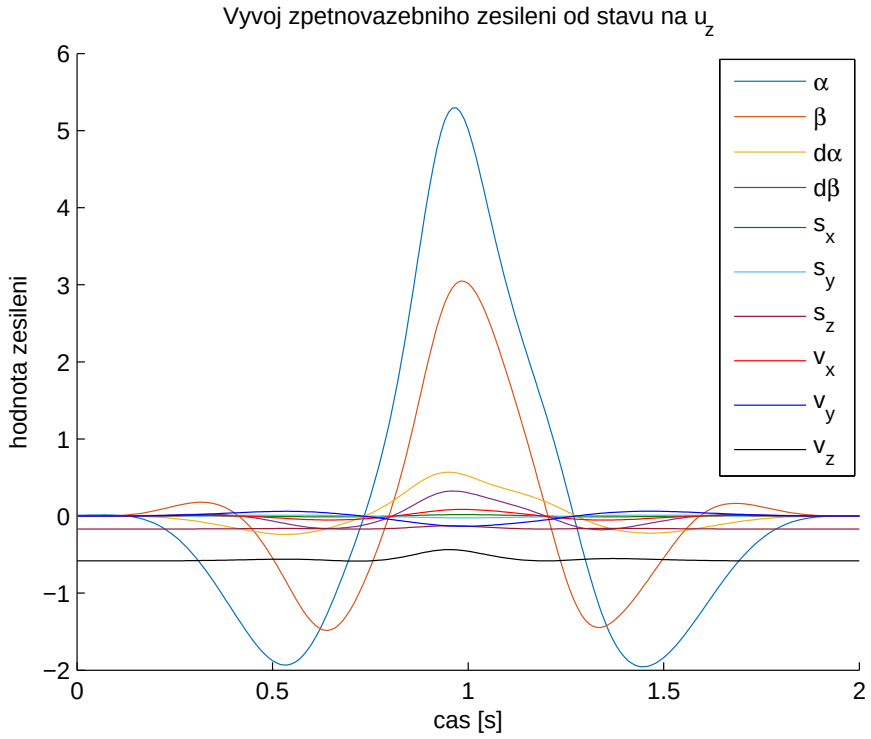
Obrázek 4.4.1: Póly uzavřené smyčky s ustáleným řešením LQ úlohy



Obrázek 4.4.2: Vývoj zesílení od jednotlivých složek stavu na řízení u_x



Obrázek 4.4.3: Vývoj zesílení od jednotlivých složek stavu na řízení u_y



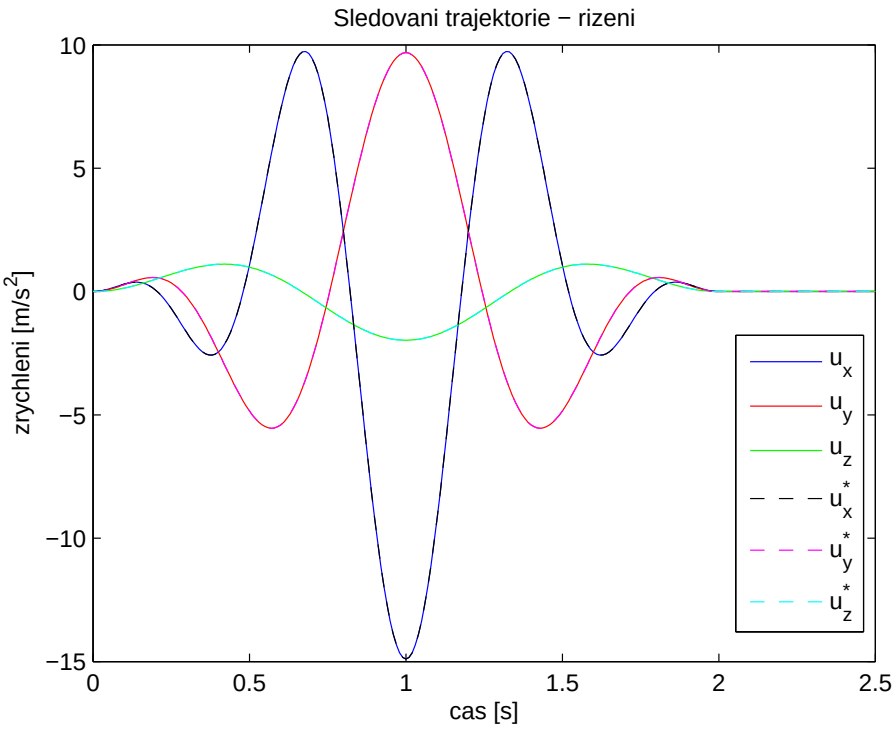
Obrázek 4.4.4: Vývoj zesílení od jednotlivých složek stavu na řízení u_z

trajektorie, projeví se tedy fakt, že systém byl postupně linearizován kolem různých pracovních bodů, které již nelze považovat za blízké okolí vzpřímené polohy. Kyvadlo by při pohybu po této trajektorii s největší pravděpodobností nebylo možné ustabilizovat pomocí konstantní zpětné vazby, proto jsme počítali časově proměnnou variantu. Z ostatních stavů jsou změny zesílení znatelné pro úhlové rychlosti kyvadla, zesílení od zbylých složek se příliš nemění a možná by pro ně bylo použitelné ustálené řešení.

4.4.1 Simulace v programu Simulink

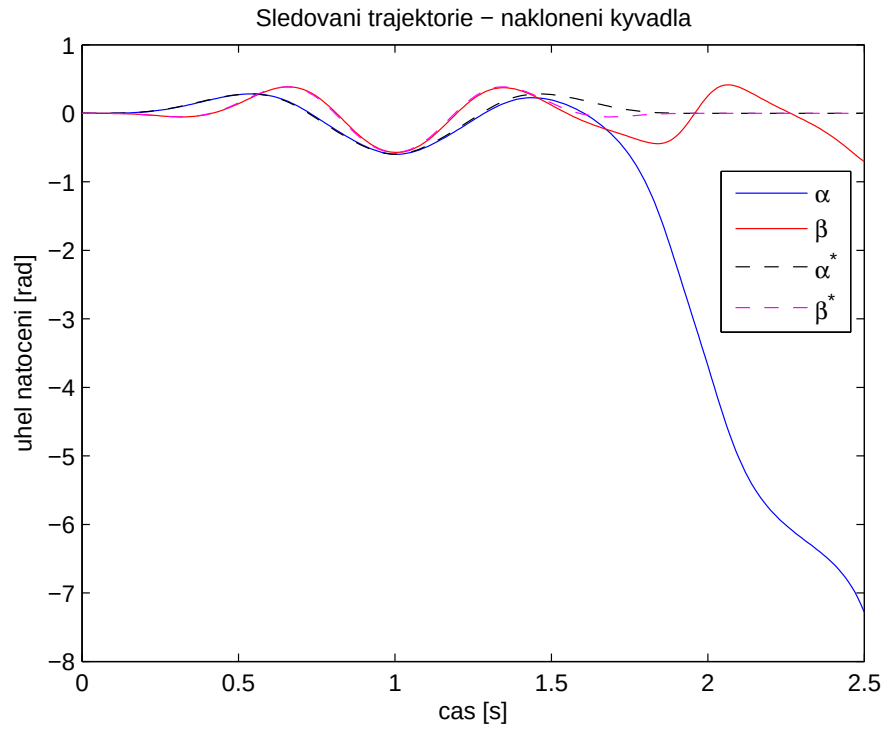
Nejprve jsme sestavili model v programovém systému MATLAB/Simulink. Simulační schéma je uvedeno v **Dodatku 6**. Subsystém ve schématu obsahuje namodelované stavové rovnice kyvadla na hrotu (2.0.29). Průběh signálů odpovídajících požadované trajektorii a časově proměnná zpětná vazba jsou importovány z pracovního prostředí MATLABu (Workspace) ve formě časové řady (timeseries).

Nejprve ověříme funkčnost sestaveného modelu tím, že rozpojíme zpětnou vazbu. Systém tedy bude bez regulátoru a protože kyvadlo na hrotu je nestabilní, mělo by po vykonání požadované řídící sekvence bez dalších zásahů spadnout.

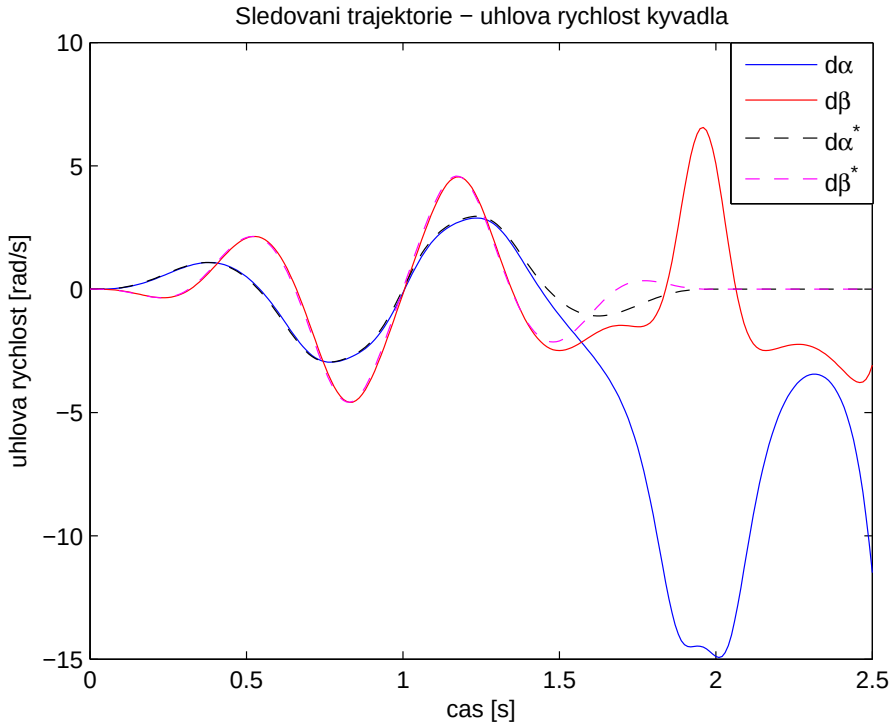


Obrázek 4.4.5: Průběh řízení při rozpojení zpětné vazby - Simulink

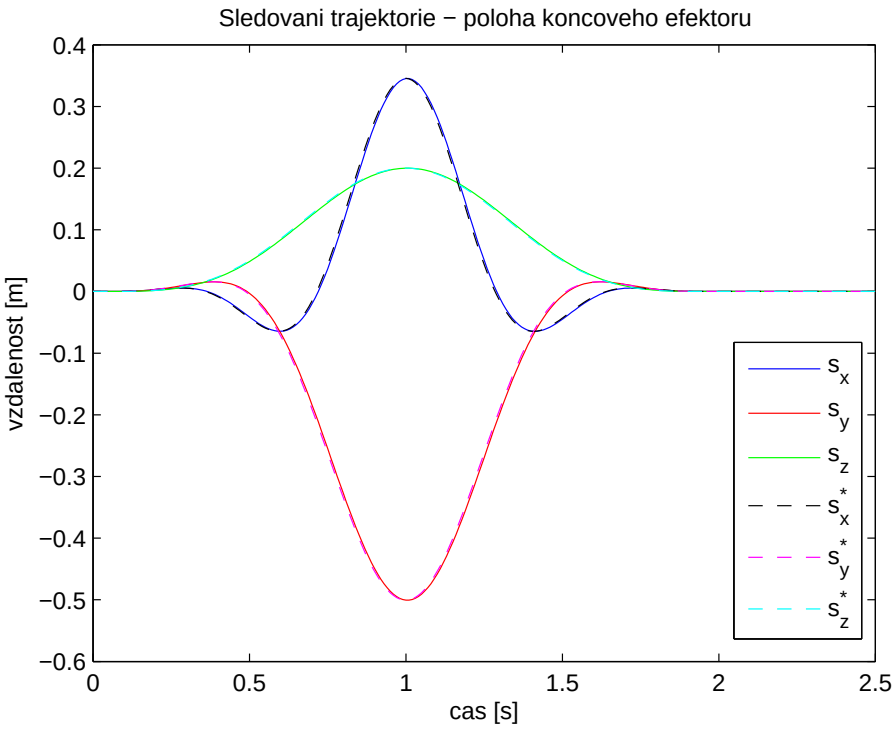
Graf řízení (obrázek 4.4.5) ukazuje, že po vykonání požadovaného řízení už opravdu nedochází k žádným dalším korekcím. Tomu odpovídá i poloha koncového efektoru (obrázek 4.4.8), která se shoduje s požadovanou. Další dva grafy vyobrazují průběhy náklonu (obrázek 4.4.6) a úhlové rychlosti (obrázek 4.4.7) kyvadla. Z velikosti úhlu náklonu v radiánech je vidět, že kyvadlo skutečně spadne. Protože v modelu uvažujeme pevné spojení pivotu a kyvadla a jelikož zanedbáváme vnější síly, jako jsou tření v ložisku pivotu



Obrázek 4.4.6: Průběh náklonu kyvadla při rozpojení zpětné vazby - Simulink



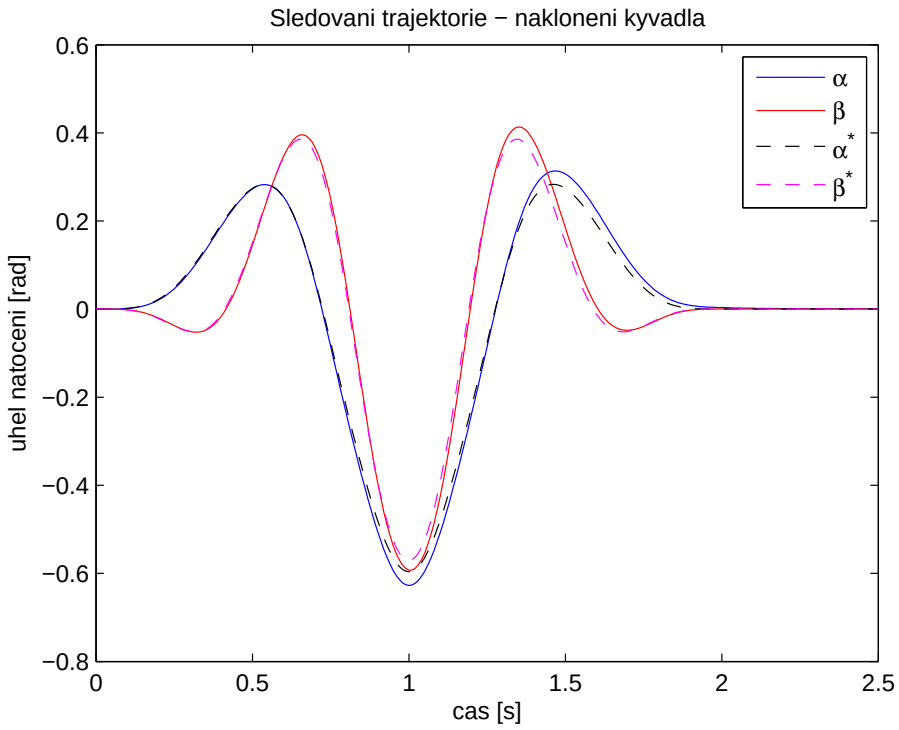
Obrázek 4.4.7: Průběh úhlové rychlosti kyvadla při rozpojení zpětné vazby - Simulink



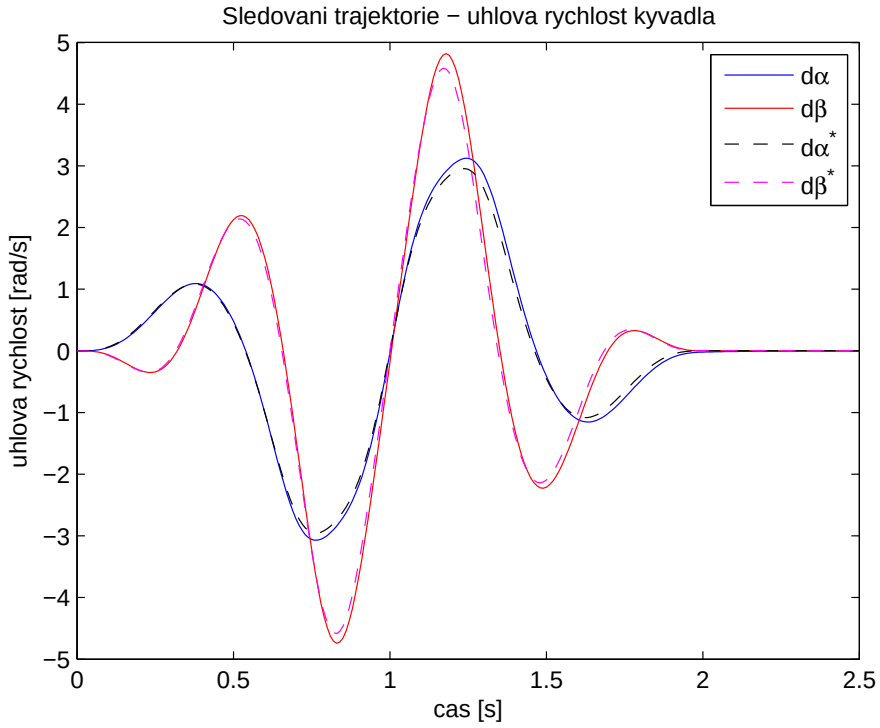
Obrázek 4.4.8: Průběh polohy pivotu při rozpojení zpětné vazby - Simulink

a odpor vzduchu, kyvadlo se otočí kolem dokola a opět se dostane do vzpřímené polohy (-2π), ve které však opět nevydrží. Toto chování nám napovídá, že model by měl být sestaven správně. Nyní tedy zapojíme zpětnou vazbu a simulačně ověříme funkci navrženého regulátoru. V následujících grafech vykreslíme náklon kyvadla (obrázek 4.4.9), jeho úhlovou rychlost (obrázek 4.4.10), řízení (obrázek 4.4.11) a také odchylku koncového efektoru od požadované pozice (obrázek 4.4.12).

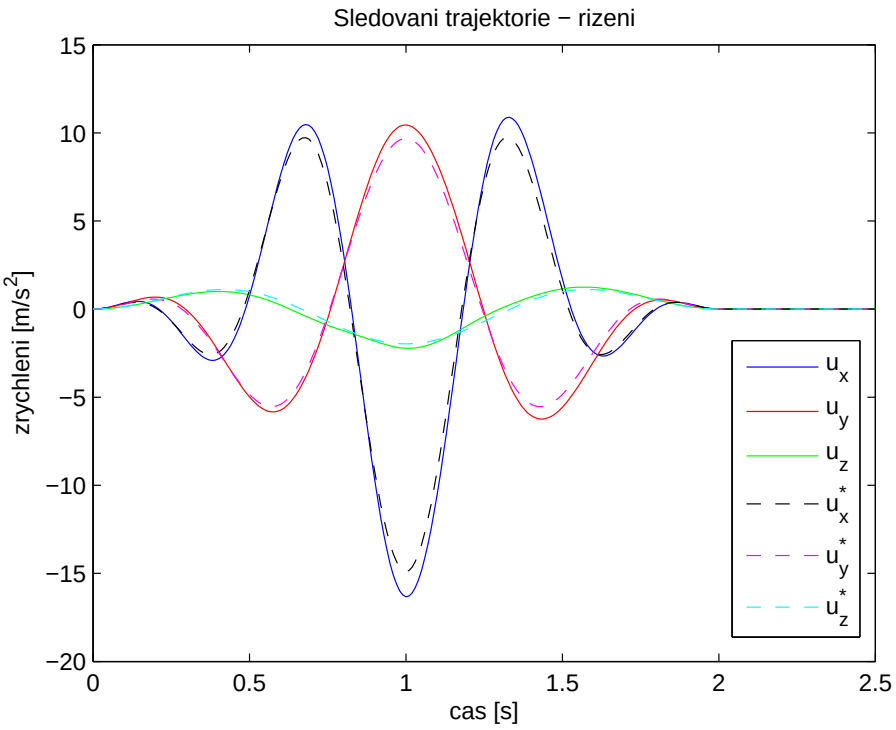
Z grafů je vidět, že regulátor poměrně přesně, a jen s mírnými odchylkami ke konci, sleduje požadované průběhy jednotlivých signálů. Kyvadlo nyní po vykonání požadované trajektorie zůstane ve vzpřímené poloze, ve které ho regulátor udržuje již jen velmi malými zásahy, které jsou patrné až při velkém přiblížení grafu. Můžeme si všimnout, že pozice pivotu přesně neodpovídá požadované poloze a na konci trajektorie je odchylka v osách y a z přibližně $10cm$, což není málo, ale pokud se tímto nedostaneme mimo pracovní prostor manipulátoru, nemusí nám odchylka příliš vadit. Tato nepřesnost je dána nastavením regulátoru, kdy jsme se soustředili především na stabilizaci kyvadla. Navíc pokud průběh polohy vykreslíme na delším časovém horizontu (obrázek 4.4.13), vidíme, že koncový efektor se časem postupně stahuje k požadované pozici. Přesně z tohoto důvodu jsme do systému zaváděli zpětnou vazbu i od polohy pivotu. Díky tomuto řešení by se při působení poruchy v podobě pulzu (lehké strčení do míče) měl manipulátor pomalu vracet zpět do původní pozice a neměl by „utéct“ mimo pracovní prostor.



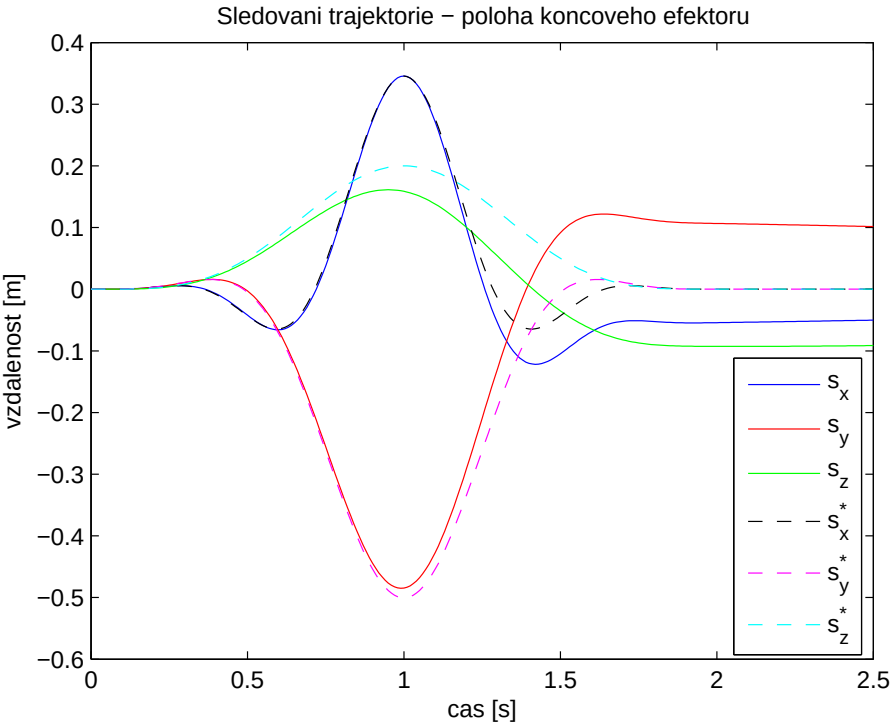
Obrázek 4.4.9: Průběh náklonu kyvadla - Simulink



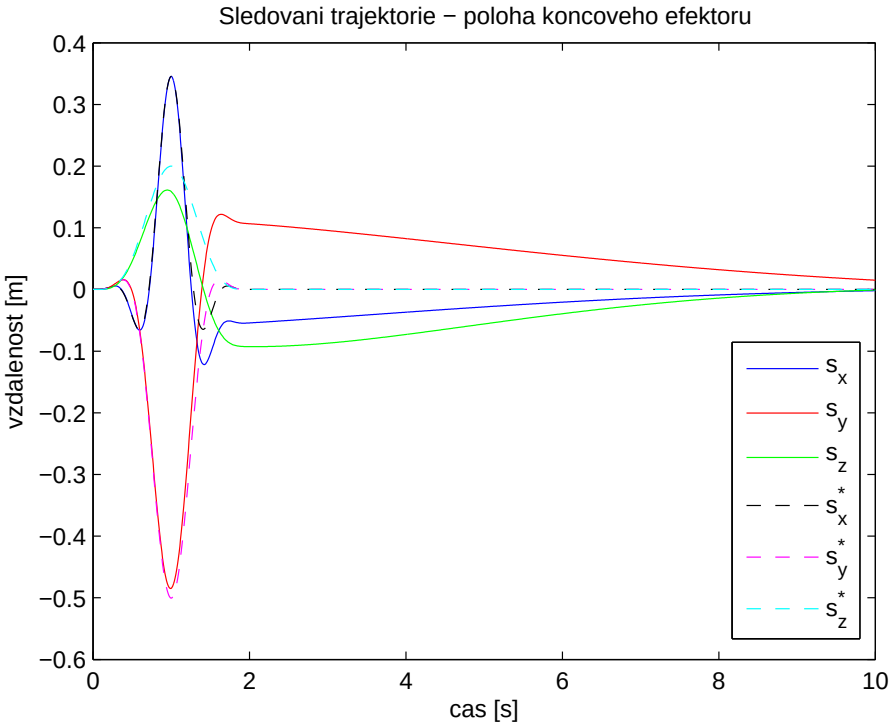
Obrázek 4.4.10: Průběh úhlové rychlosti kyvadla - Simulink



Obrázek 4.4.11: Průběh řízení systému s LQ regulátorem - Simulink



Obrázek 4.4.12: Průběh polohy pivotu - Simulink



Obrázek 4.4.13: Průběh polohy pivotu na delším časovém úseku - Simulink

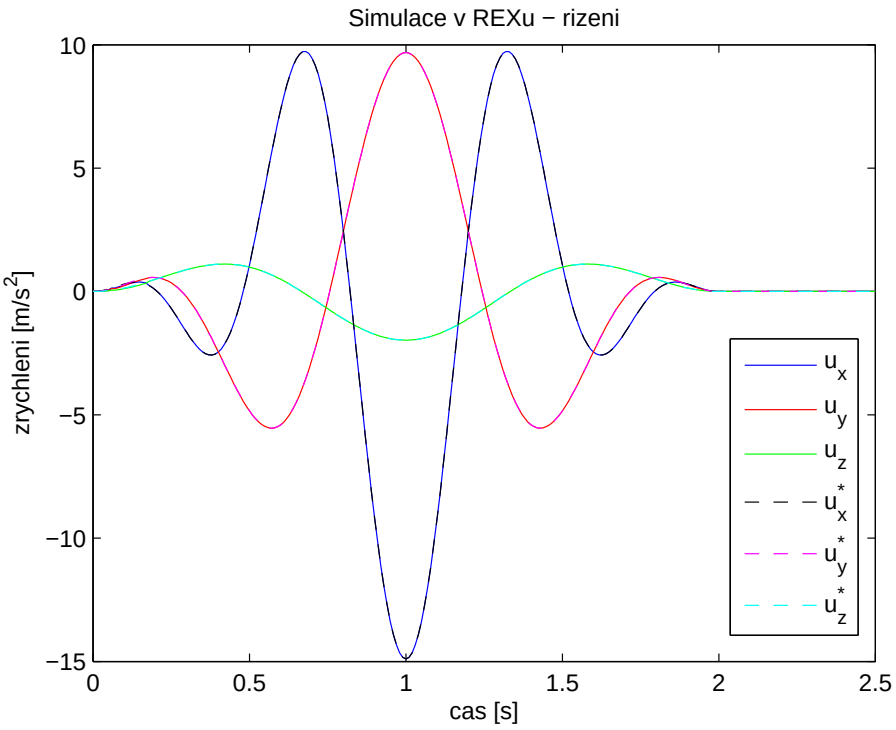
4.4.2 Simulace v programu REX

Systém zkusíme odsimulovat také v programu REX. Jedná se o řídicí systém vyvíjený na katedře kybernetiky Západočeské univerzity v Plzni. V tomto případě jsme předpočítané signály a časově proměnnou stavovou zpětnou vazbu z MATLABu uložili do souborů s příponou *.csv*. Tyto soubory jsme poté v REXu načetli pomocí bloků CNA a data v nich obsažená jsme poté podle potřeby zpracovali pomocí bloků REXLANG, které umožňují vytvoření uživatelem definovaných funkcí v kódu podobném programovacímu jazyku C. Funkce na zpracování signálů požadované trajektorie a na načtení časově proměnné zpětné vazby jsou k nahlédnutí v **Dodatku 4**, respektive v **Dodatku 5**. Bloková schémata simulační smyčky a subsystému, který opět obsahuje namodelované dynamické rovnice kyvadla na hrotu jsou k nahlédnutí v **Dodatku 7**.

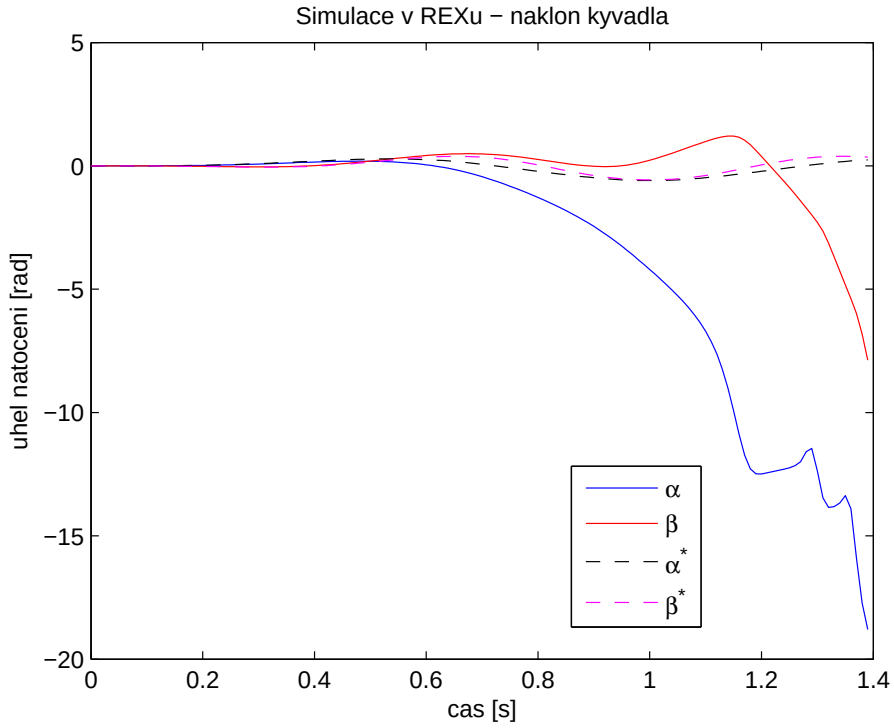
Stejně jako v Simulinku nejdříve otestujeme chování systému s rozpojenou zpětnou vazbou. Opět očekáváme, že kyvadlo bez řízení spadne.

Opět správně vidíme, že po vykonání řídicí sekvence pro požadovanou trajektorii již do systému žádné řízení nezasahuje (obrázek 4.4.14). Kyvadlo poté samozřejmě spadne (obrázek 4.4.15). Tomu odpovídají také úhlové rychlosti (obrázek 4.4.16). Můžeme si všimnout, že oproti Simulinku nyní kyvadlo spadne téměř ihned. To je způsobeno rozdílným solverem obou programů a jedná se tedy o vlastnost, ne o chybu. Ostré skoky viditelné ke konci zobrazeného časového úseku jsou způsobeny naražením na saturační meze integrátoru. V grafu polohy pivotu (obrázek 4.4.17) lze pozorovat, že poloha se přesně neustálí na nule. To může být způsobeno odlišnou implementací integračního bloku v REXu oproti Simulinku.

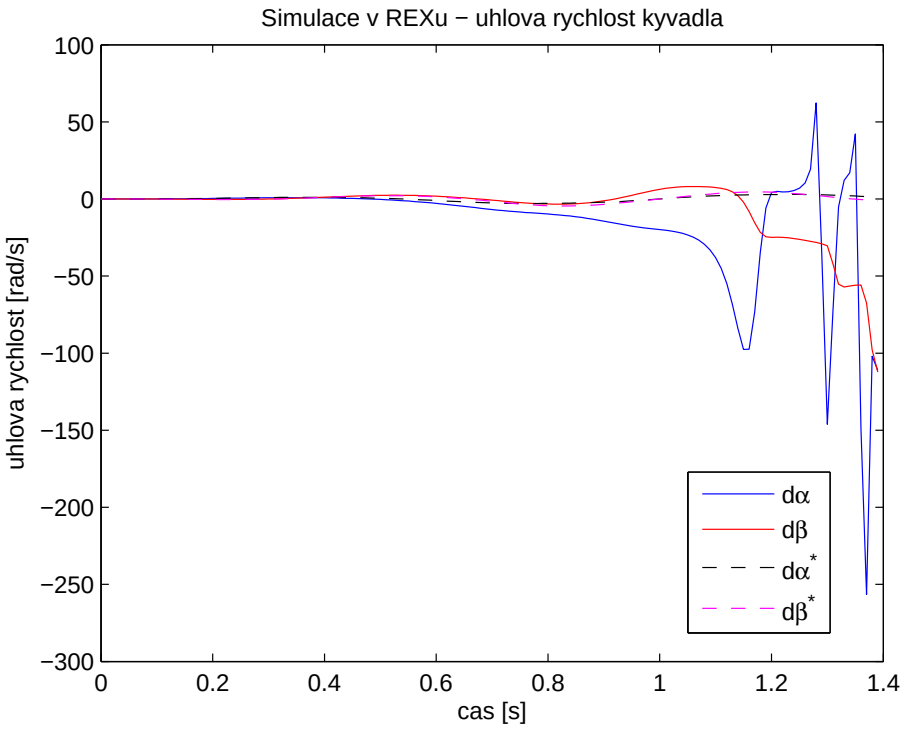
Pro další simulaci již zpětnou vazbu zavedeme. Z výsledných grafů je vidět, že re-



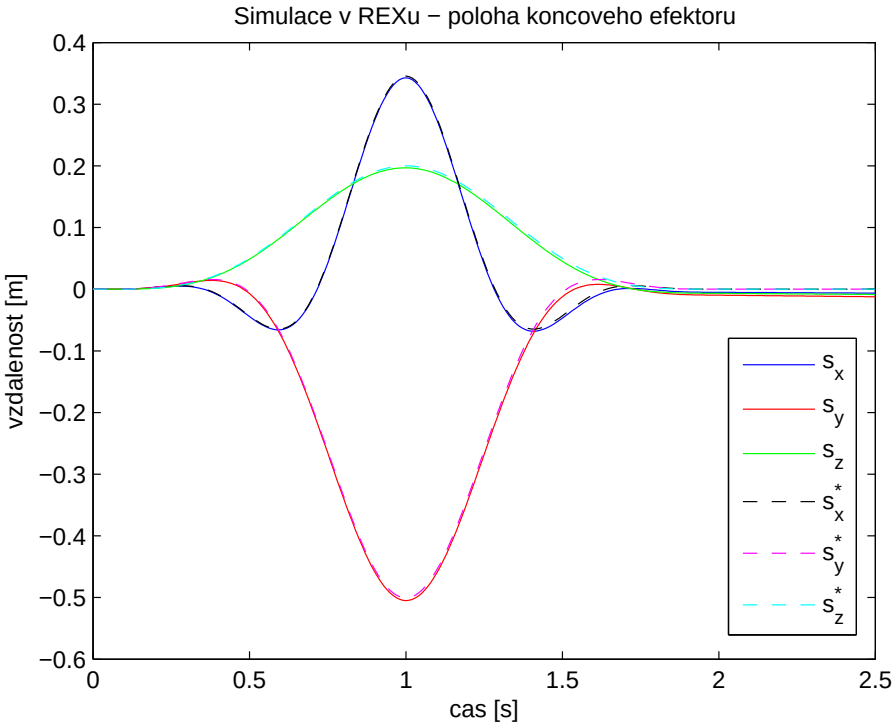
Obrázek 4.4.14: Průběh řízení při rozpojení zpětné vazby - REX



Obrázek 4.4.15: Průběh náklonu kyvadla při rozpojení zpětné vazby - REX

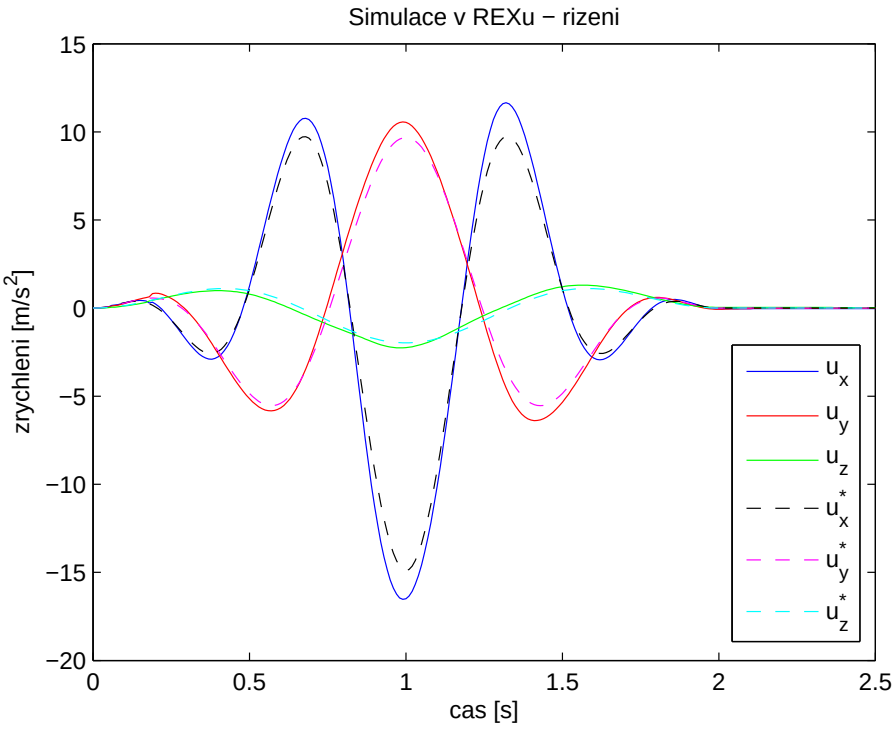


Obrázek 4.4.16: Průběh úhlové rychlosti kyvadla při rozpojení zpětné vazby - REX

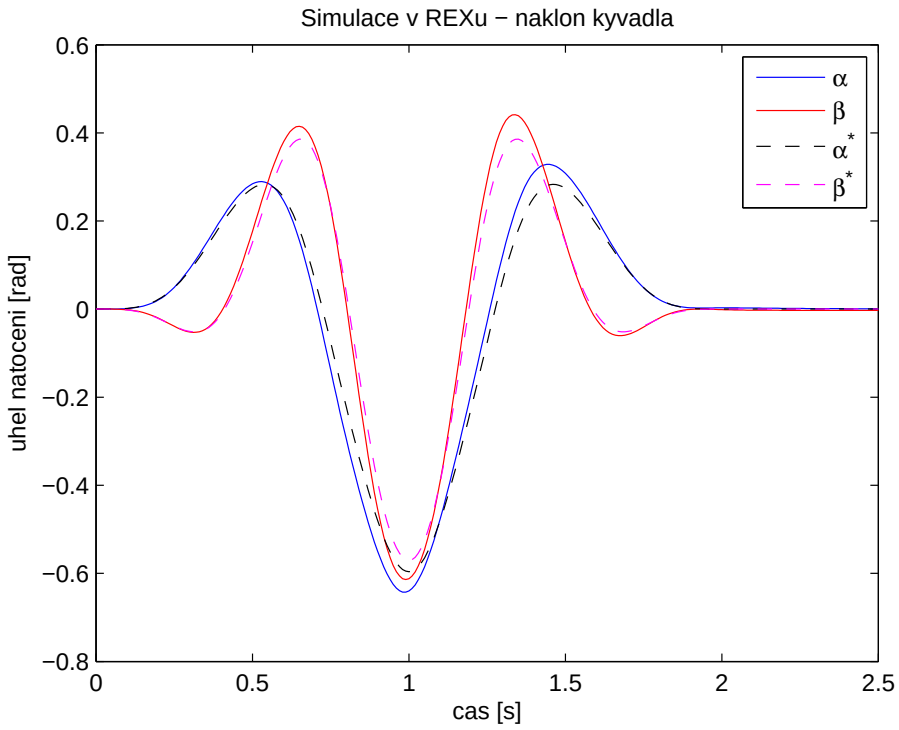


Obrázek 4.4.17: Průběh polohy pivotu při rozpojení zpětné vazby - REX

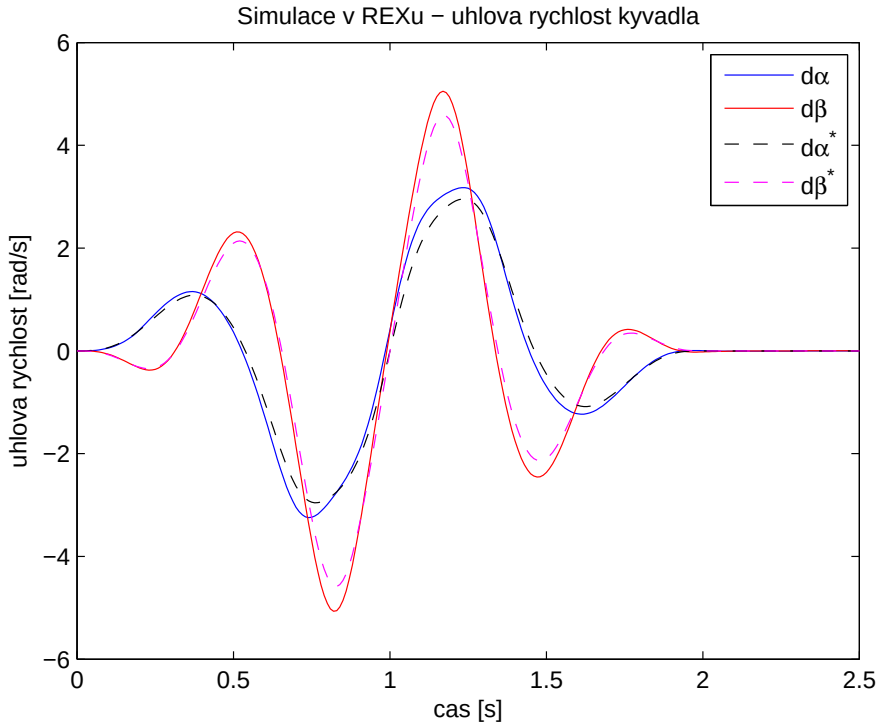
gulátor svými zásahy (obrázek 4.4.18) ustabilizuje kyvadlo ve vzpřímené poloze (obrázky 4.4.19 a 4.4.20). Opět můžeme pozorovat, že pivot se vlivem nastavení regulátoru odchyluje od požadované pozice (obrázek 4.4.21), ale postupně se stahuje zpět do výchozí pozice (obrázek 4.4.22). Celkově je z grafů vidět sledování požadovaných signálů odpovídajících vypočtené trajektorii bez výrazných odchylek. Výsledky simulace v programu REX se kvůli rozdílnému solveru od předchozích výsledků ze Simulinku mírně liší.



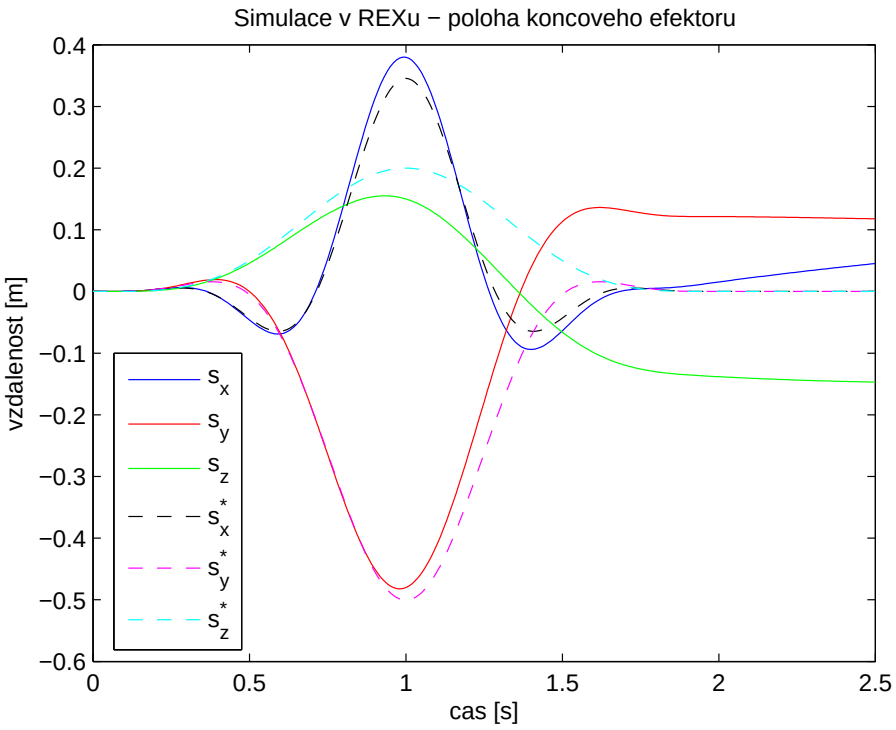
Obrázek 4.4.18: Průběh řízení systému s LQ regulátorem - REX



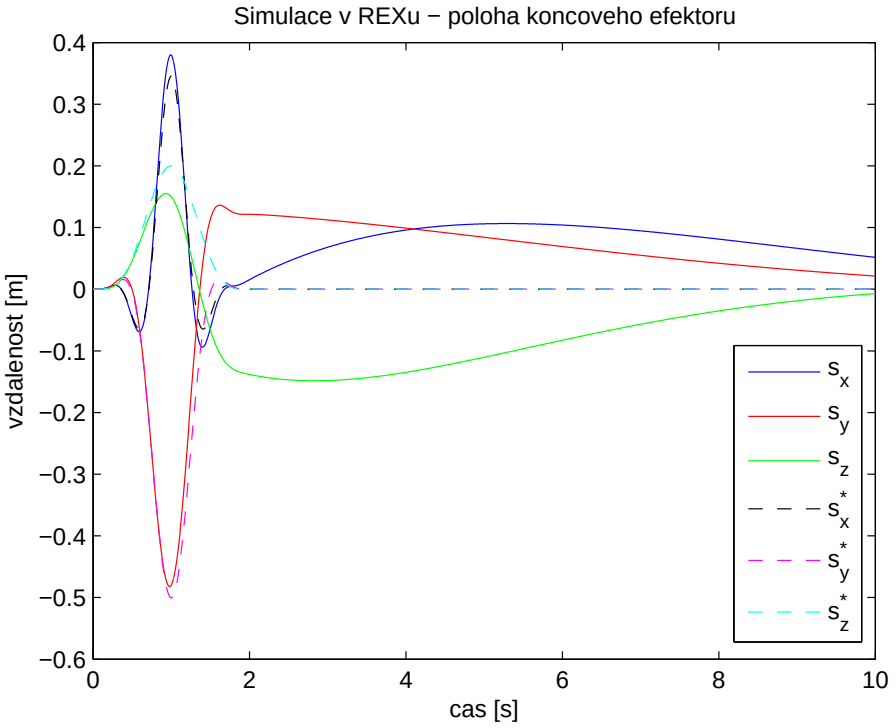
Obrázek 4.4.19: Průběh náklonu kyvadla - REX



Obrázek 4.4.20: Průběh úhlové rychlosti kyvadla - REX



Obrázek 4.4.21: Průběh polohy pivotu - REX



Obrázek 4.4.22: Průběh polohy pivotu na delším časovém úseku - REX

5 Průmyslový manipulátor

Robotika obecně je jedním z významných odvětví oboru mechatroniky. Mechatronika podle definice [10] sjednocuje principy mechaniky, elektroniky, optiky, informatiky a automatického řízení za účelem konstruování jednodušších, účinnějších a spolehlivějších systémů. Význačné postavení zde má právě obor automatického řízení, který zařízením dodává „skrytý důmysl“ pomocí vhodných senzorů, aktuátorů a vloženého řízení, čímž je umožněno lokálně ovlivňovat fyzikální vlastnosti systému. Robotika samotná se poté zabývá návrhem konstrukce robotů, návrhem vhodných algoritmů řízení, simulacemi a testováním a finálním uvedením do provozu. Pokud jde o samotné roboty, ti se mohou vyskytovat v různých podobách. Obecně je lze rozdělit do dvou základních kategorií (opět podle [10]).

1. **Manipulátory**
   Manipulátory byly původně mechanická, později elektromechanická zařízení, jejichž hlavním účelem bylo zesilovat, případně zpřesňovat práci člověka. S rozvojem technologií začaly manipulátory obsazovat řadu průmyslových aplikací a v mnohých případech zcela nahradili lidskou práci, zejména díky své přesnosti, bezchybnosti a neúnavnosti. Moderní manipulátory se používají na výrobních linkách pro přemísťování částí výrobků, svařování, lakování atd. Často se také používají k práci v omezených nebo nebezpečných prostorech a díky své přesnosti také v lékařství, kde manipulátor ulehčuje práci chirurga. V této práci budeme pracovat právě s robotem z této kategorie. Konkrétně půjde o sériový šesti-osý antropomorfní manipulátor se sférickým zápěstím.

2. **Humanoidní roboti**
   Tato skupina robotů se snaží přiblížit člověku a to nejen vzhledem, ale i projevy inteligence. U humanoidních robotů je hlavní důraz kladen na autonomii a interakci s prostředím. Jejich vývoj se tak prolíná s disciplínami umělé inteligence, jako jsou například počítačové vnímání a rozhodování. Tito roboti jsou budoucností tohoto vědního oboru, tato práce se jimi však přímo nezabývá. Nicméně, jak již bylo psáno v úvodu, některé poznatky z úlohy stabilizace a řízení podaktuovaných systémů lze použít například pro zlepšení chůze humanoidních robotů.

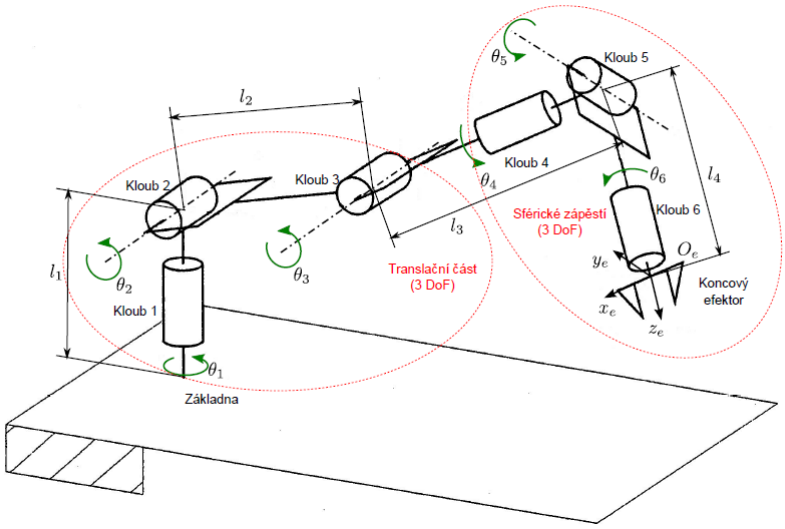
V našem případě tedy budeme pracovat s robotem spadajícím do kategorie manipulátorů. Manipulátory lze obecně rozdělit do dvou skupin podle mechanické konstrukce.

1. **Sériové manipulátory**
   Do této skupiny spadá náš manipulátor. Pro sériové manipulátory platí, že každé rameno je pomocí kloubů spojeno s dalšími dvěma rameny, přičemž základna a koncový efektor (první a poslední rameno) jsou spojeny pouze s jedním dalším ramenem. Tyto manipulátory jsou tedy tvořeny otevřeným kinematickým řetězcem a je možné je popsat acyklickým grafem, ve kterém uzly zastupují klouby a hrany zastupují ramena manipulátoru. V praxi se jedná o nejčastější typ používaných robotů, především z důvodu jednoduché mechanické architektury a poměrně velkého pracovního prostoru. Nevýhodou je, že každý kloub musí být osazen vlastním aktuátorem, což klade větší nároky na pevnost konstrukce, skrz kterou musí být po celé délce

vedena kabeláž. Každé rameno je navíc zatěžováno výhradně na ohyb, čemuž musí opět odpovídat robustnost konstrukce. To všechno se neblaze podepisuje na hmotnosti a tím pádem i na dynamických vlastnostech manipulátoru.

2. **Paralelní manipulátory**
   Koncový efektor těchto manipulátorů je se základnou spojen dvěma nebo více otevřenými kinematickými řetězci. Celkově je tedy základ těchto manipulátorů tvořen uzavřeným řetězcem a lze je popsat cyklickým grafem. Paralelní manipulátory nejsou prozatím tolik rozšířené kvůli složitější mechanické architektuře a omezenějšímu pracovnímu prostoru. Velkou výhodou tohoto typu naopak je, že hmotnost břemene je rozložena mezi jednotlivé kinematické řetězce a ty tedy nemusí být tak robustní. Při vhodně vymyšlené konstrukci lze všechny aktuátory umístit na základnu manipulátoru a tím dále snížit váhu jednotlivých ramen, což se odráží v lepších dynamických vlastnostech. Pří práci v nebezpečném prostředí lze takto také lépe chránit aktuátory.

Námi používaný manipulátor Stäubli TX 40 je sériový antropomorfní manipulátor se sférickým zápěstím, který se skládá ze šesti rotačních kloubů a napodobuje funkci lidské paže. V současnosti se jedná o nejčastěji používaný typ manipulátorů v průmyslu. Jednoduché schéma je k vidění na obrázku 5.0.23.

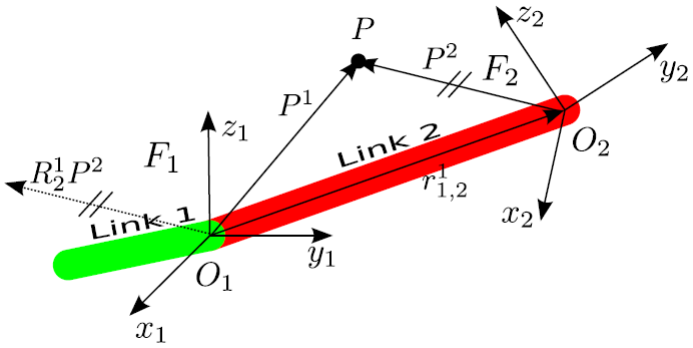


Obrázek 5.0.23: Sériový šesti-osý antropomorfní manipulátor se sférickým zápěstím

S tímto prostorovým uspořádáním jednotlivých kloubů a ramen má manipulátor šest stupňů volnosti, což je počet potřebný k popsání polohy koncového efektoru v prostoru (3 pro polohu a 3 pro natočení). Obecně je každé rameno pevně spojeno se svým souřadným systémem a pohyb manipulátoru je v podstatě transformací mezi souřadnými systémy jeho ramen. To je základní myšlenka úmluv pro popis kinematiky manipulátorů, které systematickou cestou popisují pohyb mechanické konstrukce manipulátoru. Při uvádění následující teorie budeme čerpat z [11] a budeme se zároveň držet zde uvedených postupů a značení. Teorii lze nastudovat také z [1].

5.1 Reprezentace polohy, rychlosti a zrychlení v robotice

Budeme předpokládat dvě ramena *Link 1* a *Link 2*. S každým z těchto ramen je pevně spojen souřadný systém (s.s.) $F_1 = \{\boldsymbol{O}_1 - \boldsymbol{x}_1\boldsymbol{y}_1\boldsymbol{z}_1\}$, respektive $F_2 = \{\boldsymbol{O}_2 - \boldsymbol{x}_2\boldsymbol{y}_2\boldsymbol{z}_2\}$. Budeme uvažovat výhradně pravotočivé s.s. Situace je znázorněna na obrázku 5.1.1.



Obrázek 5.1.1: Transformace souřadných systémů

Vzájemná translace s.s. je dána vektorem translace $\boldsymbol{r}_{1,2}^1 = \boldsymbol{O}_2^1 - \boldsymbol{O}_1^1 = \boldsymbol{O}_2^1$. Vzájemná rotace s.s. může být vyjádřena maticí rotace R_2^1. Horní index v těchto vztazích vyjadřuje vztažný s.s., dolní index pak označuje konkrétní bod či vektor. Matice rotace je reálná matice $[3 \times 3]$, jejíž sloupce reprezentují jednotkové směrové vektory s.s. F_2 vzhledem k s.s. F_1.

$$R_2^1 = [\boldsymbol{x}_2^1\boldsymbol{y}_2^1\boldsymbol{z}_2^1] \tag{5.1.1}$$

Zároveň se jedná o ortogonální matici, její sloupce jsou tedy kolmé.

$$\left(R_2^1\right)^T R_2^1 = I \qquad \Rightarrow \qquad \left(R_2^1\right)^T = \left(R_2^1\right)^{-1} = R_1^2 \tag{5.1.2}$$

Prvky matice rotace jsou vzájemně závislé, nezávislé jsou 3 parametry. Pro transformaci souřadnic bodu $\boldsymbol{P}$ mezi jednotlivými s.s. platí následující vztahy.

$$\boldsymbol{P}^1 = \boldsymbol{r}_{1,2}^1 + R_2^1\boldsymbol{P}^2 \tag{5.1.3a}$$

$$\boldsymbol{P}^2 = -R_1^2\boldsymbol{r}_{1,2}^1 + R_1^2\boldsymbol{P}^1 \tag{5.1.3b}$$

Existují tři základní elementární rotace dvou s.s. Jedná se o rotace o úhel α kolem osy $\boldsymbol{x}$, o úhel β kolem osy $\boldsymbol{y}$ a o úhel γ kolem osy $\boldsymbol{z}$.

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix} \tag{5.1.4a}$$

$$R_y(\beta) = \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix} \tag{5.1.4b}$$

$$R_z(\gamma) = \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.1.4c}$$

Tyto úhly se nazývají *Eulerovské úhly* a reprezentují jednu z možností popisu rotace těles v prostoru. Obecná rotace lze následně vyjádřit skládáním elementárních rotací, kdy tyto rotace provedeme buď postupně podle aktuálních os, nebo všechny vzhledem k fixované s.s. Postupnou rotací podle schématu XYZ máme na mysli otočení s.s. F_1 kolem osy $\boldsymbol{x}_1$ o úhel α, čímž vznikne nový s.s. F_1'. Následuje rotace tohoto s.s. okolo osy $\boldsymbol{y}_1'$ o úhel β a vzniká s.s. F_1''. Na závěr provedeme rotaci F_1'' okolo osy $\boldsymbol{z}_1''$ o úhel γ. Výsledná rotace lze zapsat jako

$$R_2^1 = R_x(\alpha)R_y(\beta)R_z(\gamma) \tag{5.1.5}$$

V druhém případě začneme úplně stejně, výsledný s.s. F_1' však poté nebudeme rotovat kolem nově vzniklé osy $\boldsymbol{y}_1'$, ale kolem původní osy $\boldsymbol{y}_1$. Vzniklý s.s. F_1'' poté otočíme opět podle původní osy $\boldsymbol{z}_1$. Tato rotace se zapíše následovně

$$R_2^1 = R_z(\gamma)R_y(\beta)R_x(\alpha) \tag{5.1.6}$$

Při uvažování postupné rotace vypadá výsledná rotační matice následovně. Využijeme zkrácené označení (2.0.28).

$$R_2^1(\alpha,\beta,\gamma) = \begin{bmatrix} c_\beta c_\gamma & -c_\beta s_\gamma & s_\beta \\ s_\alpha s_\beta c_\gamma + c_\alpha s_\gamma & -s_\alpha s_\beta s_\gamma + c_\alpha c_\gamma & -s_\alpha c_\beta \\ -c_\alpha s_\beta c_\gamma + s_\alpha s_\gamma & c_\alpha s_\beta s_\gamma + s_\alpha c_\gamma & c_\alpha c_\beta \end{bmatrix} \tag{5.1.7}$$

Z této matice lze takzvanou zpětnou transformací získat zpět Eulerovy úhly.

$$\begin{array}{ll} \text{Pro } \beta \in \left(-\dfrac{\pi}{2}, \dfrac{pi}{2}\right): & \text{Pro } \beta \in \left(\dfrac{\pi}{2}, \dfrac{3pi}{2}\right): \\ \alpha = atan2(-r_{23}, r_{33}) & \alpha = atan2(r_{23}, -r_{33}) \\ \beta = atan2\left(r_{13}, \sqrt{r_{23}^2 + r_{33}^2}\right) & \beta = atan2\left(r_{13}, -\sqrt{r_{23}^2 + r_{33}^2}\right) \\ \gamma = atan2(-r_{12}, r_{11}) & \gamma = atan2(r_{12}, -r_{11}) \end{array} \tag{5.1.8a}$$

Pro úhel $\beta = \pm\frac{\pi}{2}$ dochází k singularitě v reprezentaci, neboť osy $\boldsymbol{x}_1$ a $\boldsymbol{z}_1''$ jsou rovnoběžné a nelze tak určit úhly α a γ, pouze jejich součet či rozdíl.

Pro popis rotace lze využít i jiné reprezentace, například reprezentaci pomocí obecné osy rotace nebo reprezentaci pomocí jednotkového kvaternionu, pro kterou je odstraněn problém singulárních poloh při zpětné transformaci. Více například v uvedeném zdroji.

K celkovému popisu polohy s.s. lze využít homogenní transformační matici. Polohu s.s. F_2 vzhledem k F_1 můžeme pomocí homogenní transformační matice vyjádřit jako

$$T_2^1 = \left[\begin{array}{ccc:c} & R_2^1 & & \boldsymbol{r}_{1,2}^1 \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right] \tag{5.1.9}$$

Tato matice tedy dává zároveň informace o rotaci i translaci. Poslední řádek vyjadřuje perspektivu a měřítko a používá se v počítačové grafice, v robotice nemá význam. Homogenní souřadnice bodu $\boldsymbol{P}$ v s.s. F_1 mohou být vyjádřeny pomocí homogenních souřadnic téhož bodu v s.s. F_2.

$$\left[\begin{array}{c} \boldsymbol{P}^1 \\ \hdashline 1 \end{array}\right] = T_2^1 \left[\begin{array}{c} \boldsymbol{P}^2 \\ \hdashline 1 \end{array}\right] = \left[\begin{array}{c} \boldsymbol{r}_{1,2}^1 + R_2^1 \boldsymbol{P}^2 \\ \hdashline 1 \end{array}\right] \tag{5.1.10}$$

Při transformaci vektorů nezáleží na posunu, jen na vzájemné poloze bodů, poté platí vztahy uvedené v (2.0.4). Transformace můžeme také skládat. Uvažujme s.s. F_1, F_2 a F_3 a odpovídající homogenní transformační matice T_2^1 a T_3^2. Potom homogenní transformační matice mezi F_1 a F_3 je

$$\begin{aligned} T_3^1 &= \left[\begin{array}{ccc:c} & R_3^1 & & \boldsymbol{r}_{2,3}^1 \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right] = T_2^1 T_3^2 = \\ &= \left[\begin{array}{ccc:c} & R_2^1 & & \boldsymbol{r}_{1,2}^1 \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right] \left[\begin{array}{ccc:c} & R_3^2 & & \boldsymbol{r}_{2,3}^2 \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right] = \left[\begin{array}{ccc:c} & R_2^1 R_3^2 & & R_2^1 \boldsymbol{r}_{2,3}^2 + \boldsymbol{r}_{1,2}^1 \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right] \end{aligned} \tag{5.1.11}$$

Dostáváme se k reprezentaci polohy a zrychlení. Pro bod $\boldsymbol{P}$ v s.s. F_1 je můžeme v závislosti na jeho rychlosti a zrychlení v s.s. F_2 získat derivací vztahu 5.1.10.

$$\left[\begin{array}{c} \dot{\boldsymbol{P}}^1 \\ \hdashline 0 \end{array}\right] = \dot{T}_2^1 \left[\begin{array}{c} \boldsymbol{P}^2 \\ \hdashline 1 \end{array}\right] + T_2^1 \left[\begin{array}{c} \dot{\boldsymbol{P}}^2 \\ \hdashline 0 \end{array}\right] \tag{5.1.12a}$$

$$\left[\begin{array}{c} \ddot{\boldsymbol{P}}^1 \\ \hdashline 0 \end{array}\right] = \ddot{T}_2^1 \left[\begin{array}{c} \boldsymbol{P}^2 \\ \hdashline 1 \end{array}\right] + 2\dot{T}_2^1 \left[\begin{array}{c} \dot{\boldsymbol{P}}^2 \\ \hdashline 0 \end{array}\right] + T_2^1 \left[\begin{array}{c} \ddot{\boldsymbol{P}}^2 \\ \hdashline 0 \end{array}\right] \tag{5.1.12b}$$

kde

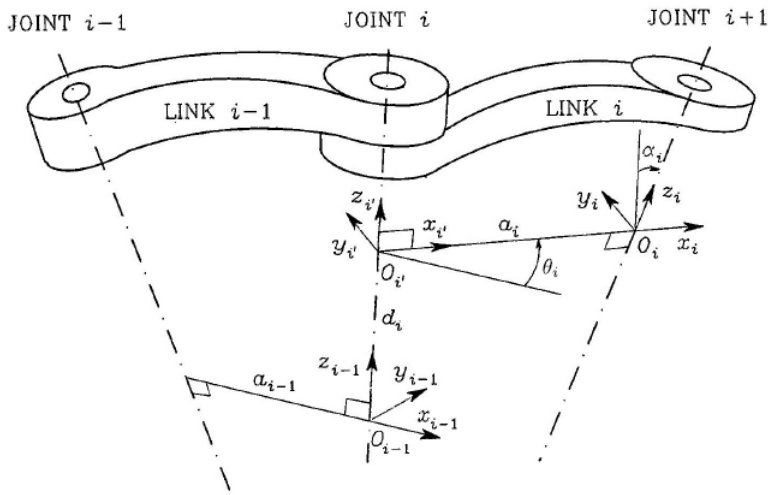
$$\dot{T}_2^1 = \left[\begin{array}{ccc:c} & \dot{R}_2^1 & & \dot{\boldsymbol{r}}_{1,2}^1 \\ \hdashline 0 & 0 & 0 & 0 \end{array}\right] \qquad \ddot{T}_2^1 = \left[\begin{array}{ccc:c} & \ddot{R}_2^1 & & \ddot{\boldsymbol{r}}_{1,2}^1 \\ \hdashline 0 & 0 & 0 & 0 \end{array}\right] \tag{5.1.13}$$

Časové derivace vektoru translace $\boldsymbol{r}_{1,2}^1$ vyjadřují translační rychlost a zrychlení s.s. F_2 v s.s. F_1. Časové derivace matice rotace R_2^1 poté souvisejí s úhlovou rychlostí a úhlovým zrychlením. Více opět v [11].

5.2 Denavit-Hartenbergova úmluva

Účelem úmluv pro popis kinematiky manipulátorů je jednoduchou, systematickou a rekurzivní cestou definovat souřadné systémy jednotlivých ramen a jejich vzájemné transformace a tím popsat pohyb mechanické konstrukce manipulátoru. Pohyb manipulátoru je ovlivněn geometrickými parametry ξ, které zahrnují geometrický tvar ramen, kloubů a jejich vzájemnou konfiguraci, a kloubovými souřadnicemi Q, které obsahují aktuální polohu kloubů manipulátoru. Denavit-Hartenbergova úmluva je nejrozšířenější z úmluv. Při jejím popisu budeme nadále vycházet z [11].

Budeme uvažovat dvojici ramen *Link i* a *Link i-1* spojenou kloubem *Joint i* s jedním stupněm volnosti, viz obrázek 5.2.1.



Obrázek 5.2.1: Schéma Denavit-Hartenbergovy úmluvy

Dle D-H úmluvy lze s.s. $F_i = \{\boldsymbol{O}_i - \boldsymbol{x}_i\boldsymbol{y}_i\boldsymbol{z}_i\}$ za předpokladu znalosti $F_{i-1} = \{\boldsymbol{O}_{i-1} - \boldsymbol{x}_{i-1}\boldsymbol{y}_{i-1}\boldsymbol{z}_{i-1}\}$ definovat následovně:

- Zvol osu $\boldsymbol{z}_i$ podél osy rotace (případně translace) kloubu *Joint i+1* a osu $\boldsymbol{z}_i'$ podél osy rotace (translace) kloubu *Joint i*.
- Umísti počátek $\boldsymbol{O}_i$ s.s. F_i do průsečíku osy $\boldsymbol{z}_i$ a normály os $\boldsymbol{z}_{i-1}$ a $\boldsymbol{z}_i$. Umísti počátek $\boldsymbol{O}_i'$ s.s. $F_i' = \boldsymbol{O}_i' - \boldsymbol{x}_i'\boldsymbol{y}_i'\boldsymbol{z}_i'$ do průsečíku osy $\boldsymbol{z}_{i-1}$ a téže normály.

- Zvol osy $\boldsymbol{x}_i$ a $\boldsymbol{x}'_i$ podél normály ve směru od kloubu *Joint i* do kloubu *Joint i+1*.
- Zvol osy $\boldsymbol{y}_i$ a $\boldsymbol{y}'_i$ tak, aby výsledné s.s. byly pravotočivé.

D-H úmluva poté jednoznačně nedefinuje umístění s.s. v následujících případech

- Pro $F_0 = \{\boldsymbol{O}_0 - \boldsymbol{x}_0\boldsymbol{y}_0\boldsymbol{z}_0\}$ je jednoznačně určena pouze osa $\boldsymbol{z}_0$ podle osy rotace (translace) prvního kloubu *Joint 1*. Osu $\boldsymbol{x}_0$ a počátek $\boldsymbol{O}_0$ lze volit libovolně, osa $\boldsymbol{y}_0$ poté dotváří pravotočivý systém.
- Pro $F_n = \{\boldsymbol{O}_n - \boldsymbol{x}_n\boldsymbol{y}_n\boldsymbol{z}_n\}$, kde n je počet kloubů s jedním stupněm volnosti uvažovaného manipulátoru, není jednoznačně určena osa $\boldsymbol{z}_n$, protože kloub *Joint n+1* již neexistuje. Osa $\boldsymbol{x}_n$ musí zůstat kolmá na osu $\boldsymbol{z}_{n-1}$.
- Pokud jsou osy $\boldsymbol{z}_{i-1}$ a $\boldsymbol{z}_i$ dvou po sobě jdoucích kloubů rovnoběžné, není jejich normála jednoznačně definována a může tak být libovolně posunuta v jejich směru.
- Pokud se osy $\boldsymbol{z}_{i-1}$ a $\boldsymbol{z}_i$ dvou po sobě jdoucích kloubů protínají, je normála nulové délky a osa $\boldsymbol{x}_i$ je volena tak, aby byla kolmá k rovině definované těmito osami.

Poté může být vzájemná poloha s.s. F_{i-1} a F_i popsána pouze pomocí čtyř D-H parametrů:

- a_i ... vzdálenost mezi počátky $\boldsymbol{O}_i$ a $\boldsymbol{O}'_i$
- d_i ... vzdálenost mezi počátky $\boldsymbol{O}_{i-1}$ a $\boldsymbol{O}'_i$
- α_i ... úhel mezi osami $\boldsymbol{z}_{i-1}$ a $\boldsymbol{z}_i$ daný pootočením s.s. F'_i kolem osy $\boldsymbol{x}'_i$
- θ_i ... úhel mezi osami $\boldsymbol{x}_{i-1}$ a $\boldsymbol{x}_i$ daný pootočením s.s. F_{i-1} kolem osy $\boldsymbol{z}_{i-1}$

Pokud je *Joint i* rotační, je proměnná definující pohyb kloubu θ_i a ostatní parametry jsou konstanty definující geometrické uspořádání ramene *Link i*. Pro prizmatický kloub je proměnnou definující jeho pohyb parametr d_i.

Transformační vztah mezi s.s. F_{i-1} a F_i vyjádřený pomocí homogenní transformační matice je dán jako

$$\begin{aligned} T_i^{i-1} &= T_{i'}^{i-1}T_i^{i'} = Trans(\boldsymbol{z}, d_i)Rot(\boldsymbol{z}, \theta_i)Trans(\boldsymbol{x}, a_i)Rot(\boldsymbol{x}, \alpha_i) = \\ &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{\alpha_i} & -s_{\alpha_i} & 0 \\ 0 & s_{\alpha_i} & c_{\alpha_i} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \\ &= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i}c_{\alpha_i} & s_{\theta_i}s_{\alpha_i} & a_ic_{\theta_i} \\ s_{\theta_i} & c_{\theta_i}c_{\alpha_i} & -c_{\theta_i}s_{\alpha_i} & a_is_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{5.2.1}$$

5.3 Přímý geometrický model

V praxi se obvykle dají měřit jen polohy jednotlivých aktuátoru v kloubech (pomocí enkodérů) a ne přímo poloha koncového efektoru. Přímý geometrický model řeší otázku, jaká bude poloha koncového efektoru manipulátoru, pokud známe polohu jeho kloubů. Chceme tak získat zobecněné souřadnice X z kloubových souřadnic Q. Matematicky:

$$X = G(Q, \xi) \tag{5.3.1}$$

kde ξ jsou návrhové parametry manipulátoru. Podle D-H úmluvy platí:

$$Q = \begin{bmatrix} q_1 & q_2 & \dots & q_n \end{bmatrix}, \qquad q_i = \begin{cases} \theta \text{ pro } \textit{Joint } i \text{ rotačního typu} \\ d_i \text{ pro } \textit{Joint } i \text{ prizmatického typu} \end{cases} \tag{5.3.2}$$

Přímý geometrický model pro n kloubů sériového manipulátoru lze formulovat ve tvaru

$$T_n^0(Q) = \prod_{i=1}^{n} T_i^{i-1}(q_i) \tag{5.3.3}$$

Pro sériové manipulátory má přímý geometrický model vždy analytické řešení. Z praktického hlediska je někdy výhodné definovat ještě další dva s.s., které zajišťují kompenzaci koncového efektoru a základny manipulátoru.

$$T_e^b(Q) = T_0^b(Q) T_n^0(Q) T_e^b(Q) \tag{5.3.4}$$

Více detailů opět v uvedeném zdroji.

5.4 Inverzní geometrický model

Inverzní geometrický model řeší opačnou úlohu, tedy jaká bude poloha aktuátoru manipulátoru, pokud známe polohu koncového efektoru. Nyní tedy chceme určit kloubové souřadnice Q ze zobecněných souřadnic X. Matematicky

$$Q = G^{-1}(X, \xi) \tag{5.4.1}$$

Jedná se tedy o primární úlohu při plánování pohybu robota, která umožňuje takzvaný koordinovaný pohyb aktuátorů manipulátoru. Nalezení inverze nelineární vektorové transformační funkce G^{-1} však není obecně triviální problém, neboť ve funkci G se vinou transformační matice T_e^b vyskytují součty násobků a mocnin členů $\cos q_i$ a $\sin q_i$. Dochází tak k výskytu nejednoznačných řešení a singulárních poloh. Pro jednoduché manipulátory lze nalézt přímé analytické řešení, dále existují specializované metody pro konkrétní varianty architektur, metody pro řešení obecných architektur a numerické metody. Další informace včetně příkladů lze nalézt v [12].

5.5 Přímá a inverzní okamžitá kinematická úloha

V přímé a inverzní geometrické úloze byly uvažovány pouze polohové závislosti mezi kloubovými souřadnicemi Q a zobecněnými souřadnicemi X. Kompletní kinematický popis však požaduje i znalost závislostí mezi rychlostmi a zrychleními souřadnic. Konkrétně přímá kinematická úloha řeší problém nalezení závislosti rychlosti nebo zrychlení zobecněných souřadnic X na rychlosti nebo zrychlení kloubových souřadnic Q.

$$\dot{X} = \frac{\partial G(Q)}{\partial Q}\dot{Q} = J(Q)\dot{Q} \tag{5.5.1a}$$

$$\ddot{X} = \dot{J}(Q,\dot{Q})\dot{Q} + J(Q)\ddot{Q} \tag{5.5.1b}$$

kde $J(Q)$ je jakobián a $\dot{J}(Q,\dot{Q})$ je časová derivace jakobiánu. Jakobián může být analytický nebo kinematický, přičemž tyto reprezentace se liší pouze v reprezentaci orientace. Pokud je orientace vyjádřena Eulerovými úhly $\alpha,\ \beta,\ \gamma$, mluvíme o analytickém jakobiánu, poté

$$\dot{X} = \begin{bmatrix}\dot{x} & \dot{y} & \dot{z} & \dot{\alpha} & \dot{\beta} & \dot{\gamma}\end{bmatrix}^T \tag{5.5.2}$$

O kinematickém jakobiánu mluvíme v případě, že rychlost orientace koncového efektoru je realizována prostřednictvím vektoru úhlové rychlosti $\boldsymbol{\omega} = \begin{bmatrix}\omega_x & \omega_y & \omega_z\end{bmatrix}^T$ a tak platí

$$\dot{X} = \begin{bmatrix}\dot{x} & \dot{y} & \dot{z} & \omega_x & \omega_y & \omega_z\end{bmatrix}^T \tag{5.5.3}$$

Inverzní okamžitá kinematická úloha poté řeší problém nalezení závislosti rychlosti, respektive zrychlení kloubových souřadnic Q na rychlosti, respektive zrychlení zobecněných souřadnic.

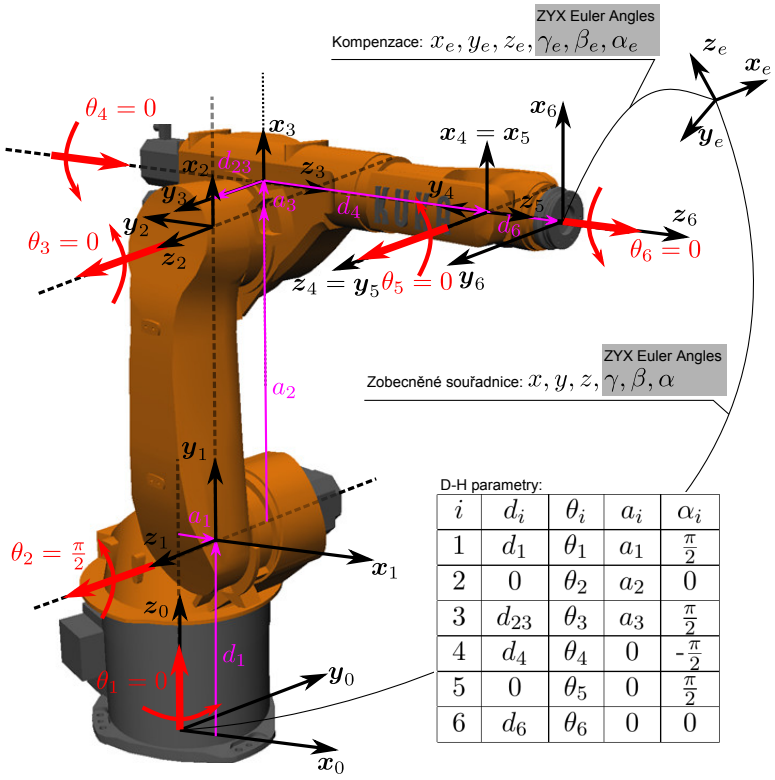
$$\dot{Q} = J^{-1}(Q)\dot{X} \tag{5.5.4a}$$

$$\ddot{Q} = J^{-1}(\ddot{X} - J(Q,\dot{Q})\dot{Q}) \tag{5.5.4b}$$

Více o této problematice v [11] a [13].

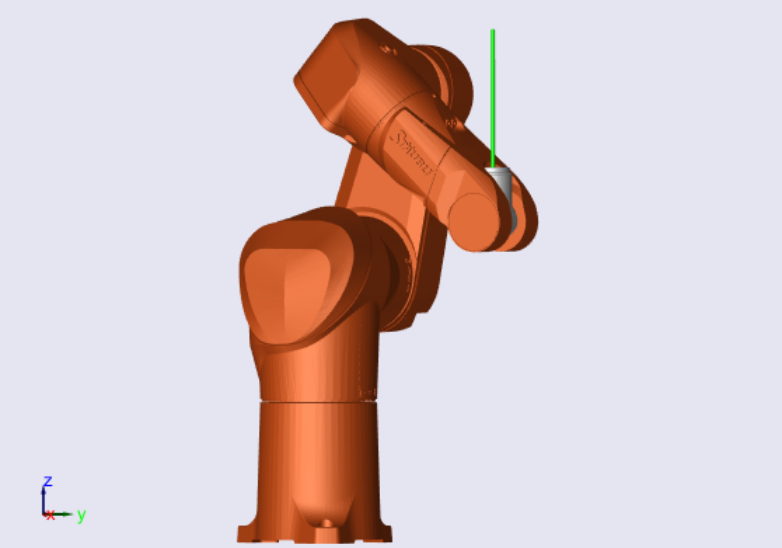
5.6 Model manipulátoru Stäubli TX 40

Model manipulátoru byl sestaven pomocí knihovny *robotLib* v základu popsané v [11], která obsahuje funkční bloky a funkce vytvořené za účelem vytváření virtuálních simulačních modelů sériových manipulátorů. Mimo jiné tato knihovna obsahuje funkce pro definování Denavit-Hartenbergových parametrů nebo pro výpočet dopředné a inverzní kinematiky. Určení D-H parametrů je znázorněno v obrázku 5.6.1. Za jednotlivé parametry bylo poté dosazeno podle dokumentace použitého manipulátoru. Následně bylo sestaveno simulační schéma robota v programovém prostředí SimMechanics (**Dodatek 8**, zde jako součást regulační smyčky). Při simulaci jsme poté vykreslovali animaci, obrázek z ní je k vidění v obrázku 5.6.2. Do manipulátoru jsou pouštěny signály ve formě polohy, odpovídajících rychlostí a zrychlení. Tyto signály jsou pomocí inverzní okamžité kinematické úlohy přepočteny na signály vyjadřující požadovaný pohyb jednotlivých kloubů.



i	d_i	θ_i	a_i	α_i
1	d_1	θ_1	a_1	$\frac{\pi}{2}$
2	0	θ_2	a_2	0
3	d_{23}	θ_3	a_3	$\frac{\pi}{2}$
4	d_4	θ_4	0	$-\frac{\pi}{2}$
5	0	θ_5	0	$\frac{\pi}{2}$
6	d_6	θ_6	0	0

Obrázek 5.6.1: Schéma určení D-H parametrů antropomorfního manipulátoru se sférickým zápěstím



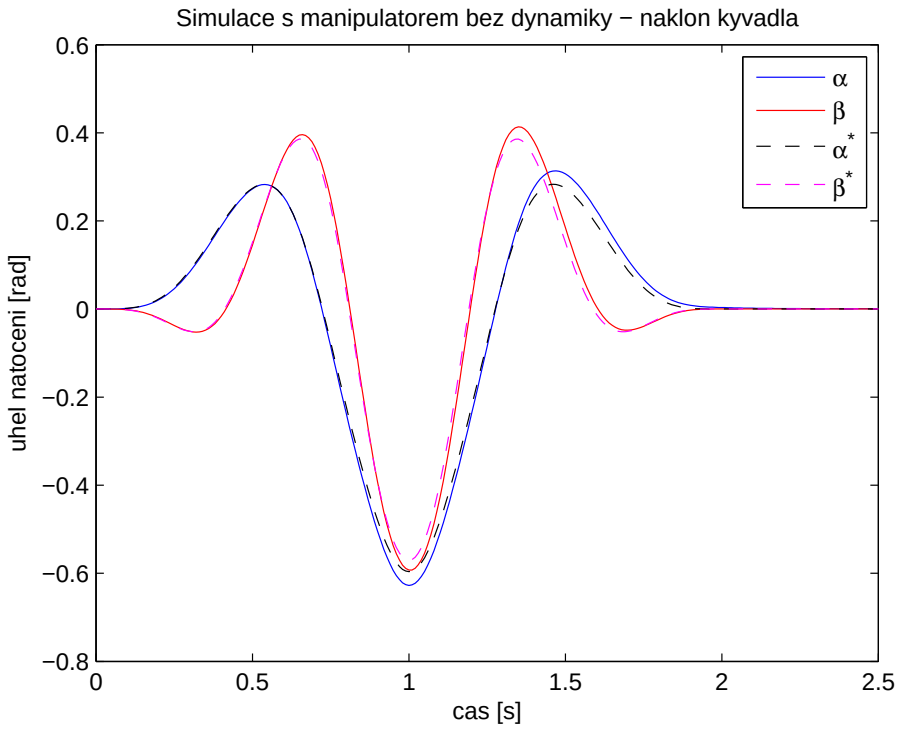
Obrázek 5.6.2: Animace manipulátoru v SimMechanicsu

6 Simulace s modelem manipulátoru

6.1 Simulace bez uvažování dynamiky manipulátoru

Nejprve budeme robota uvažovat jako ideální manipulátor. Celý robot by tedy byl dokonale tuhý, v kloubech by nebyla žádná vůle a motory by byly také ideální. Ma-

nipulátor by nám tak do systému nevnášel nemodelovanou dynamiku a odezva by tak měla být shodná s odezvou samotného systému kyvadla na hrotu bez manipulátoru. Robot Stäubli namodelovaný v SimMechanicsu umístíme mimo zpětnovazební smyčku a v tomto případě tedy bude sloužit jen pro účely ilustrace.

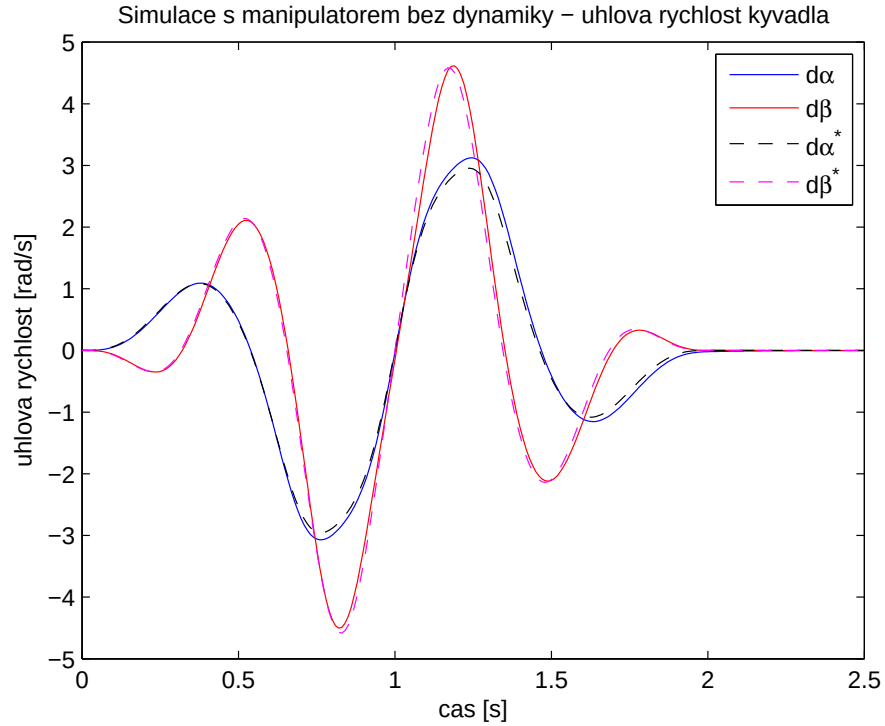


Obrázek 6.1.1: Simulace s manipulátorem bez uvažování jeho dynamiky - průběh náklonu kyvadla

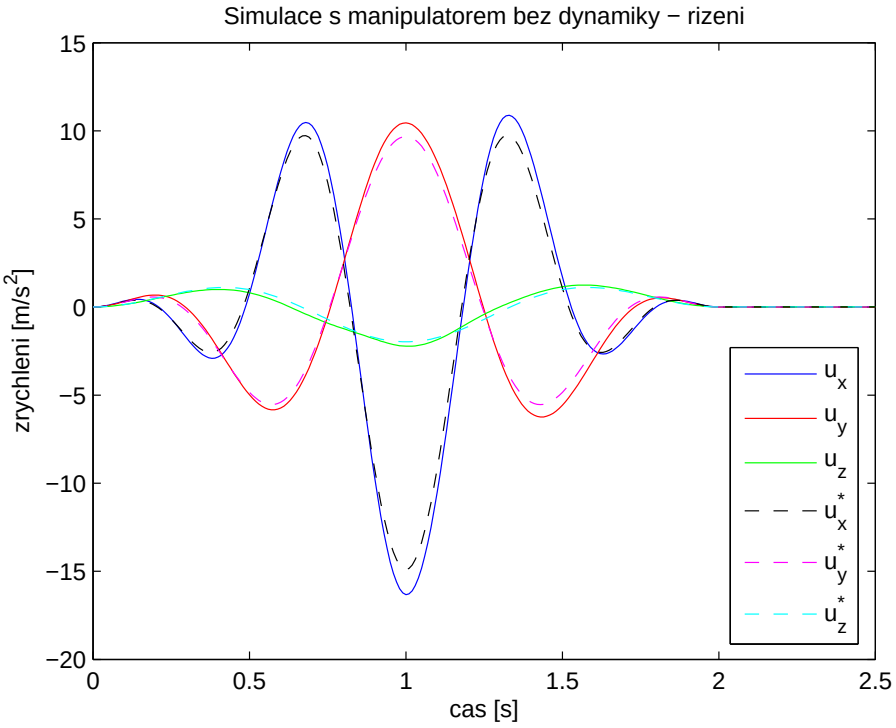
Můžeme pozorovat, že grafy 6.1.1 - 6.1.5 jsou opravdu shodné s výsledky simulace bez manipulátoru (kapitola 4.4.1).

6.2 Simulace s uvažováním dynamiky manipulátoru

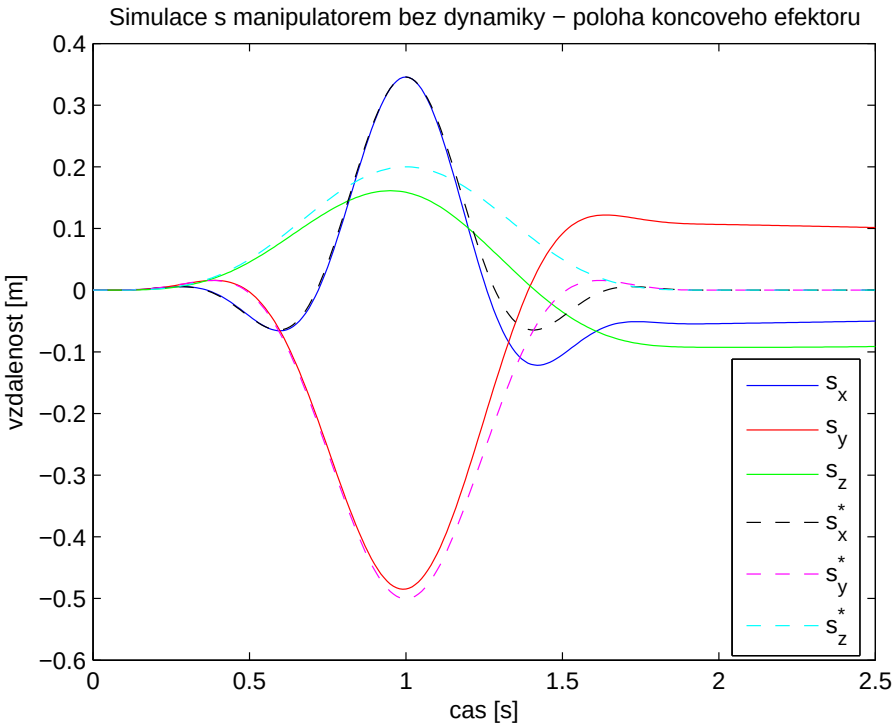
V reálném světě se však takovýto ideální manipulátor samozřejmě nevyskytuje a pokaždé nám do systému vnese nějakou další, dříve nemodelovanou dynamiku. S tím je tedy třeba počítat a pokud možno zahrnout dynamiku už do simulace. Toho dosáhneme tak, že model manipulátoru s kyvadlem zařadíme do zpětnovazební regulační smyčky. Řízení poté bude počítáno z výstupů senzorů umístěných na tomto modelu. Při měření jsme se museli vypořádat s drobným zádrhelem, neboť na kyvadle nelze snímat přímo úhel natočení, pouze jeho úhlovou rychlost. Jednoduchým řešením je úhlovou rychlost jednoduše zintegrovat. Při integraci ale dochází k nepřesnosti a po vykonání trajektorie se úhly kyvadla neustálily na nulové hodnotě. Manipulátor poté při snaze vyrovnat tuto odchylku narazil na svá konstrukční omezení a dostal by se mimo pracovní prostor. Ze senzoru na kyvadle lze však vyvést i rotační matici, ze které je následně možné vypočíst úhly rotace α a β kolem os x a y. Pokud náklon kyvadla určujeme tímto způsobem, k



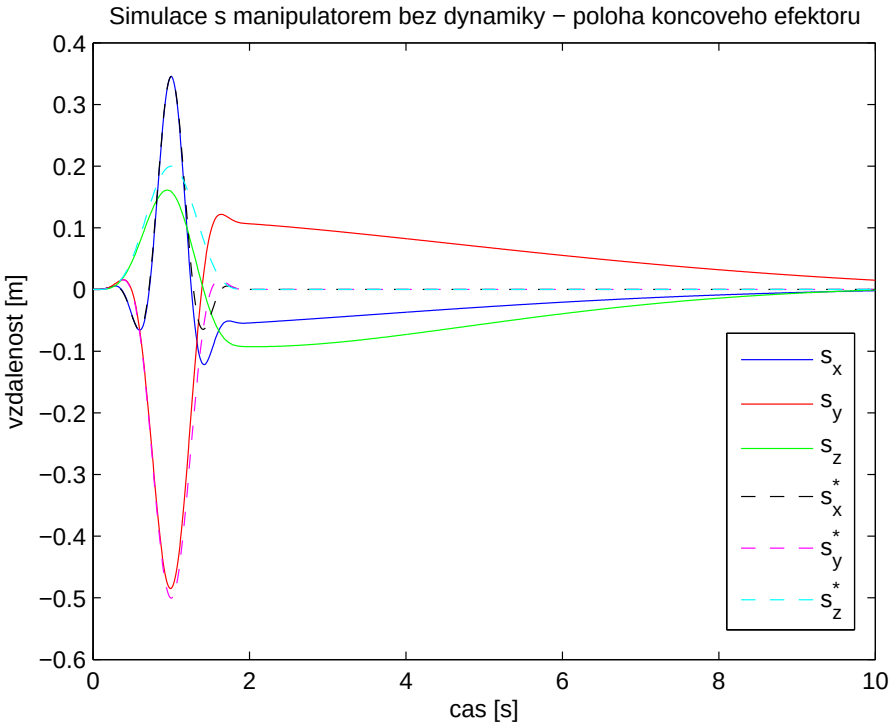
Obrázek 6.1.2: Simulace s manipulátorem bez uvažování jeho dynamiky - průběh úhlové rychlosti kyvadla



Obrázek 6.1.3: Simulace s manipulátorem bez uvažování jeho dynamiky - průběh řízení



Obrázek 6.1.4: Simulace s manipulátorem bez uvažování jeho dynamiky - průběh polohy koncového efektoru



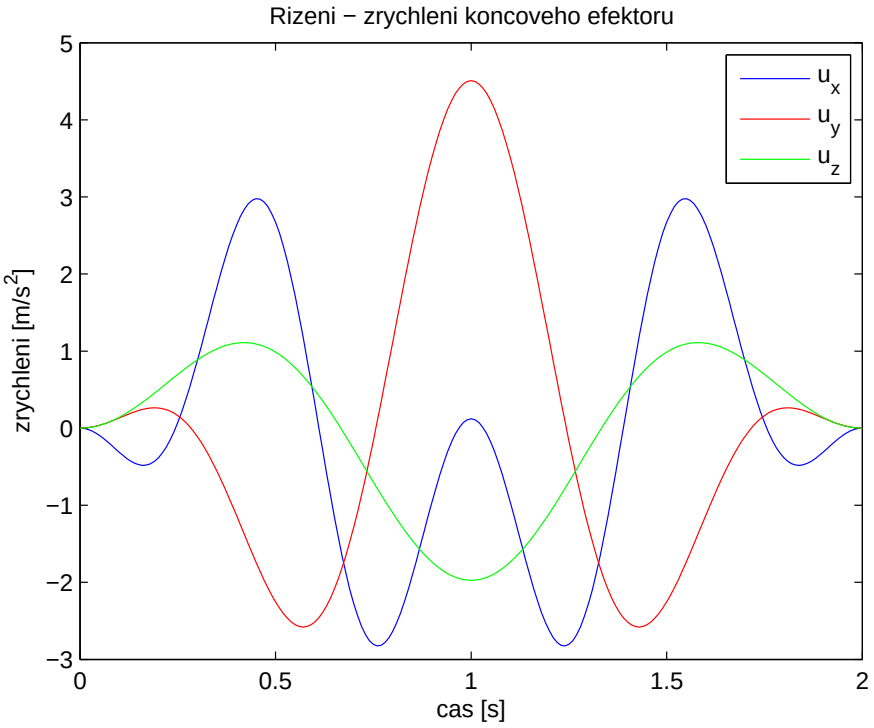
Obrázek 6.1.5: Simulace s manipulátorem bez uvažování jeho dynamiky - průběh polohy koncového efektoru na delším časovém úseku

problémům již nedochází. Polohu a rychlost pivotu poté můžeme přímo odměřit z koncového efektoru.

Při spuštění simulace jsme narazili na další problém. Vinou dynamiky manipulátoru dochází k větší odchylce polohy koncového efektoru od požadované trajektorie a nejsme schopni se vejít do pracovního prostoru robota, přičemž nepomohla ani změna počáteční polohy.

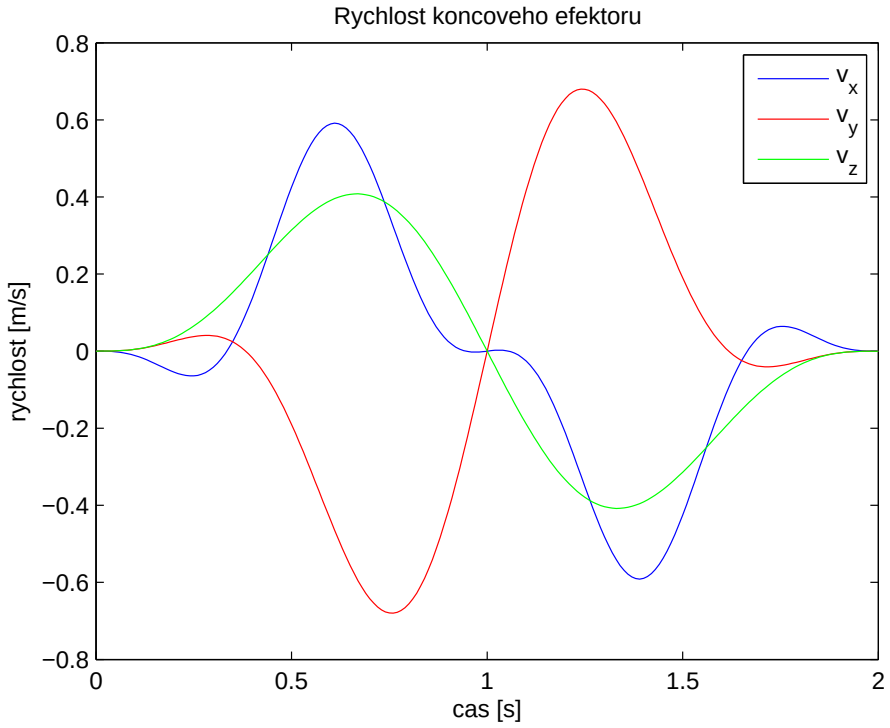
6.2.1 Výpočet nové trajektorie

Nezbývá nám nic jiného, než vypočíst jinou trajektorii. Požadavky zůstávají stejné jako v případě první trajektorie (kapitola 3.2), tentokrát se ale budeme snažit najít prostorově úspornější trajektorii, se kterou se vejdeme do pracovního rozsahu manipulátoru i při větších odchylkách polohy. Podobu funkce řízení a čas ponecháme beze změny a budeme hýbat počátečními odhady řešení a hodnot parametrů. Protože hledáme polohově kratší trajektorii na stejném časovém horizontu, dá se předpokládat, že náklon kyvadla bude v jejím průběhu menší. Postupným zkoušením jsme našli následující trajektorii.

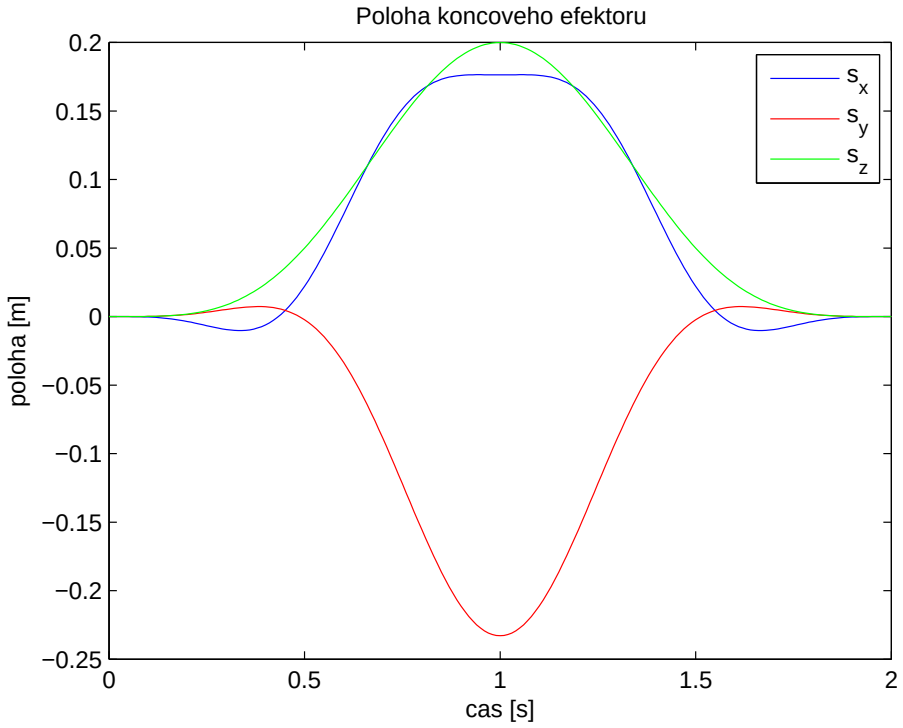


Obrázek 6.2.1: Nová trajektorie - zrychlení koncového efektoru (řízení)

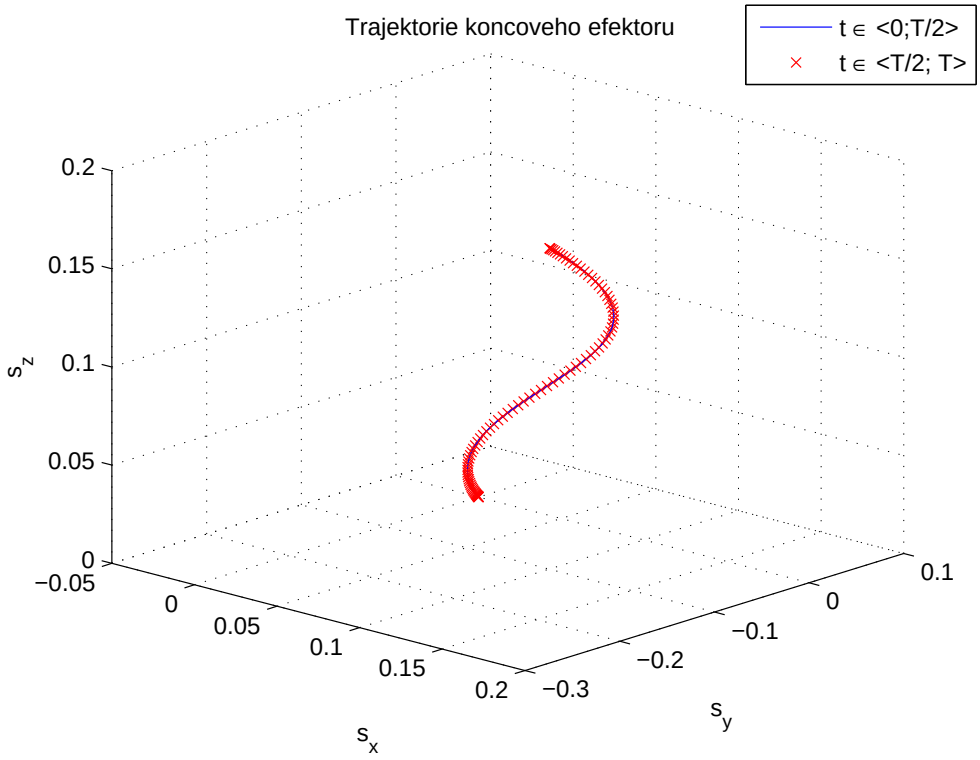
Zrychlení (obrázek 6.2.1), rychlost (obrázek 6.2.2) i poloha koncového efektoru (obrázek 6.2.3) vyhovují požadavku na nulovou hodnotu na počátku i konci pohybu. Největší rozdíl polohy v jedné ose je nyní oproti předchozí trajektorii poloviční, s udržením se v pracovním prostoru manipulátoru by tedy neměl být problém. Z prostorového grafu (obrázek 6.2.4) je vidět, že pohyb koncového efektoru je podobného charakteru jako předtím. Grafy náklonu a úhlové rychlosti kyvadla (obrázky 6.2.5 a 6.2.6) potvrzují domněnku, že náklon kyvadla bude dosahovat menších úhlů. V porovnání s první trajektorií je úhel maximálního náklonu v jedné ose přibližně poloviční.



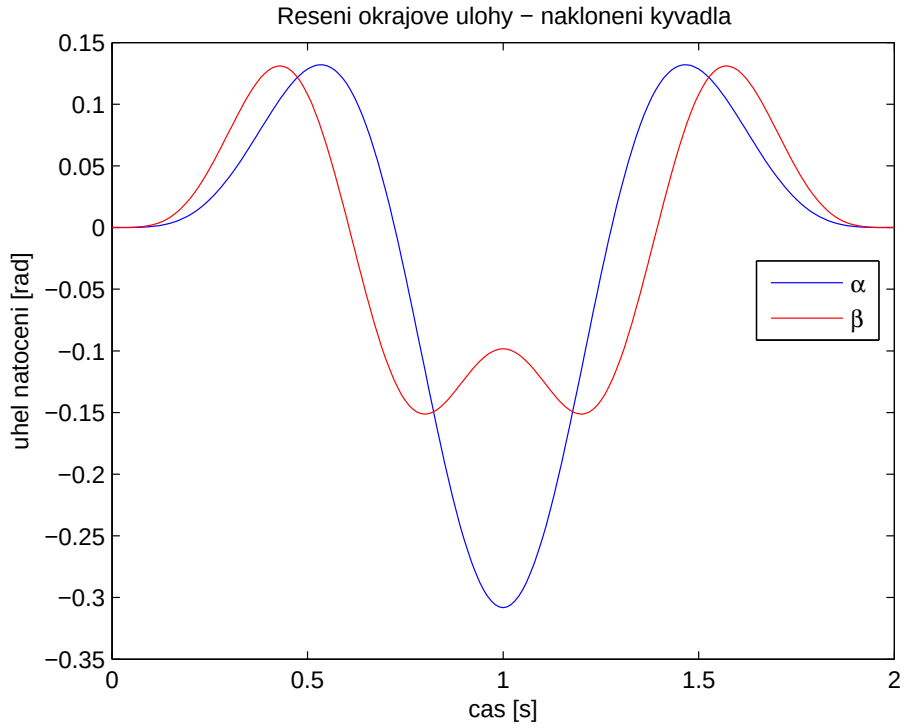
Obrázek 6.2.2: Nová trajektorie - rychlost koncového efektoru



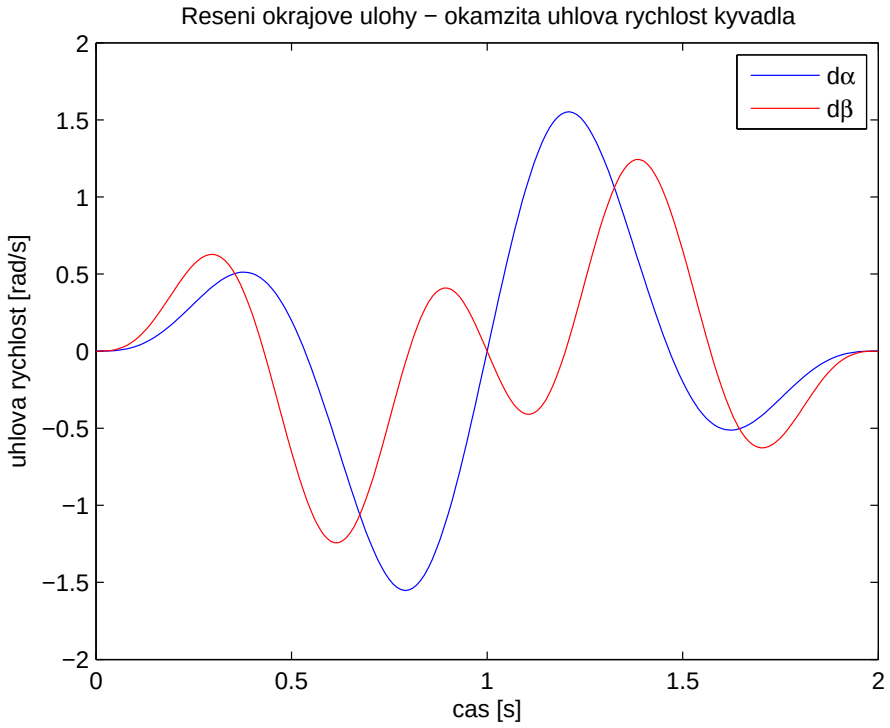
Obrázek 6.2.3: Nová trajektorie - poloha koncového efektoru



Obrázek 6.2.4: Nová trajektorie - trajektorie koncového efektoru



Obrázek 6.2.5: Nová trajektorie - průběh náklonu kyvadla



Obrázek 6.2.6: Nová trajektorie - průběh úhlové rychlosti kyvadla

6.2.2 Návrh LQR k nové trajektorii

Obdobně jako v případě první trajektorie, i nyní musíme provést linearizaci systému podél nově vypočítané trajektorie a poté vypočítat LQ regulátor na konečném horizontu. Při návrhu řízení jsme se opět soustředili zejména na ustabilizování kyvadla. Postupně jsme došli k nastavení parametrů regulátoru

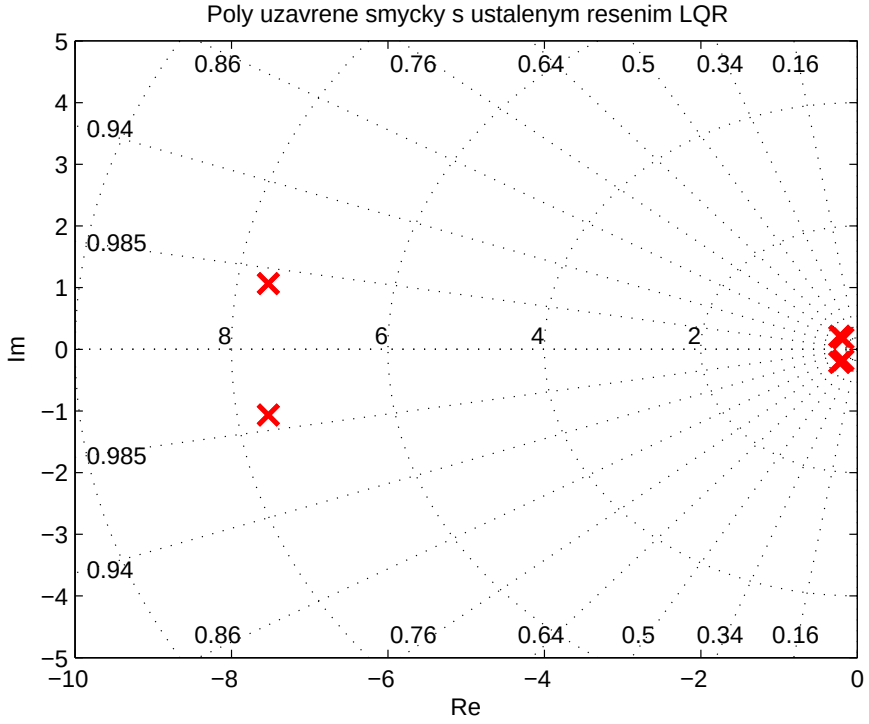
$$Q = \begin{bmatrix} 1000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.2.1a}$$

$$R = \begin{bmatrix} 100 & 0 & 0 \\ 0 & 100 & 0 \\ 0 & 0 & 500 \end{bmatrix} \tag{6.2.1b}$$

Póly uzavřené smyčky s ustáleným řešením jsou nyní

$$p_1 = -7.5289 + 1.0645i \tag{6.2.2a}$$
$$p_2 = -7.5289 - 1.0645i \tag{6.2.2b}$$
$$p_3 = -7.5289 + 1.0645i \tag{6.2.2c}$$
$$p_4 = -7.5289 - 1.0645i \tag{6.2.2d}$$
$$p_5 = -0.2233 + 0.2124i \tag{6.2.2e}$$
$$p_6 = -0.2233 - 0.2124i \tag{6.2.2f}$$
$$p_7 = -0.2233 + 0.2124i \tag{6.2.2g}$$
$$p_8 = -0.2233 - 0.2124i \tag{6.2.2h}$$
$$p_9 = -0.1792 + 0.1761i \tag{6.2.2i}$$
$$p_{10} = -0.1792 - 0.1761i \tag{6.2.2j}$$

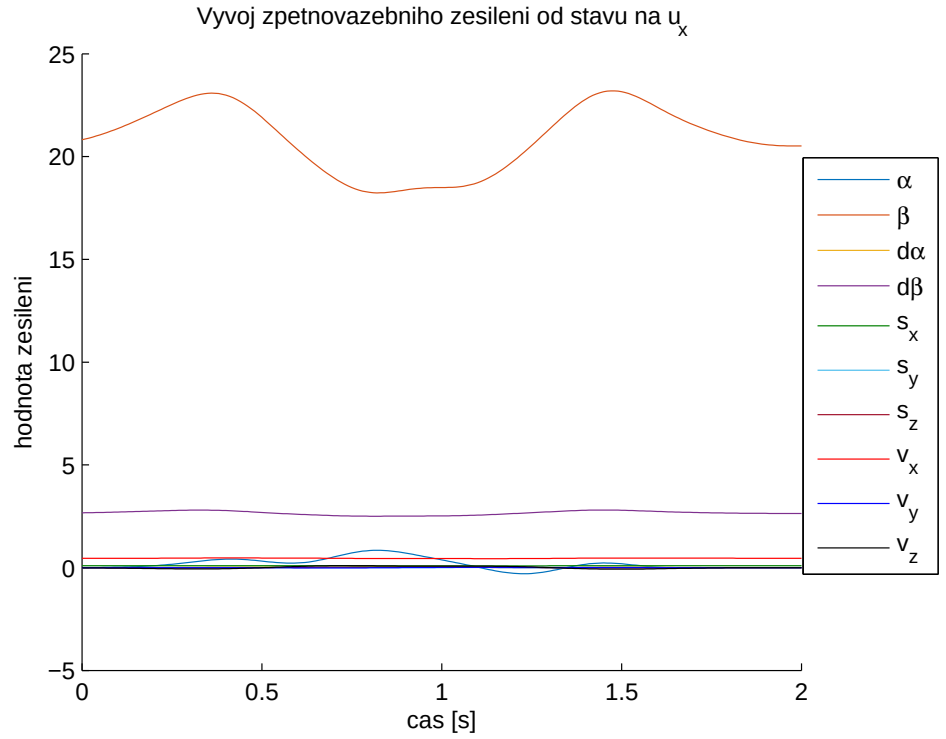
Všechny póly jsou opět stabilní a dostatečně zatlumené. Grafické znázornění je na obrázku 6.2.7.



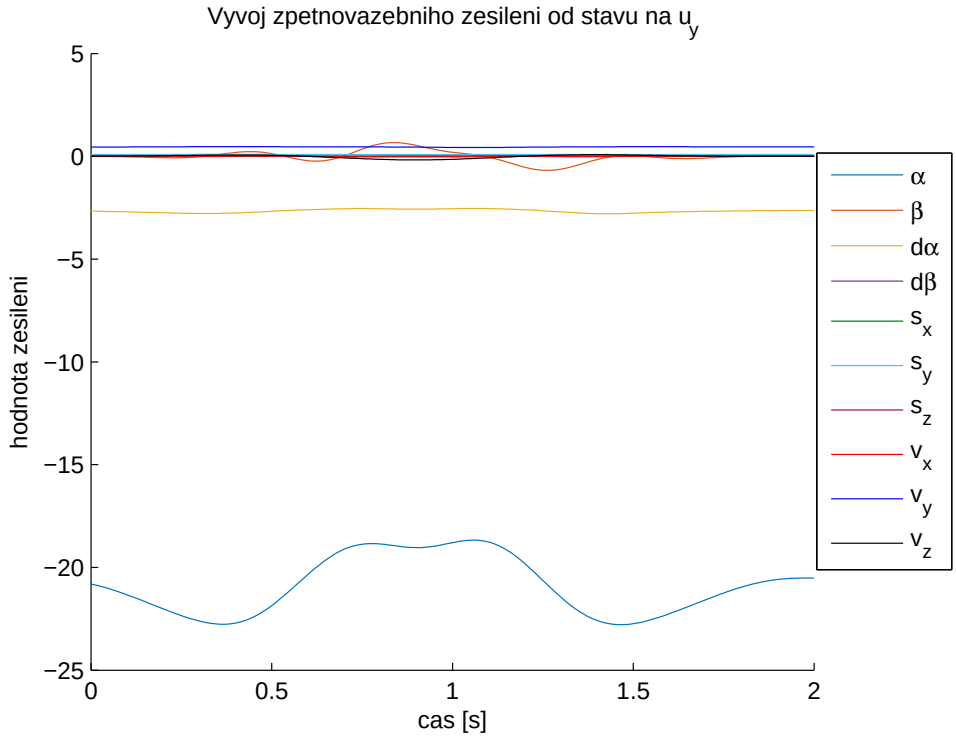
Obrázek 6.2.7: Póly uzavřené smyčky s ustáleným řešením LQ úlohy pro novou trajektorii

I pro novou trajektorii vykreslíme vývoj zesílení (obrázky 6.2.8, 6.2.9 a 6.2.10).

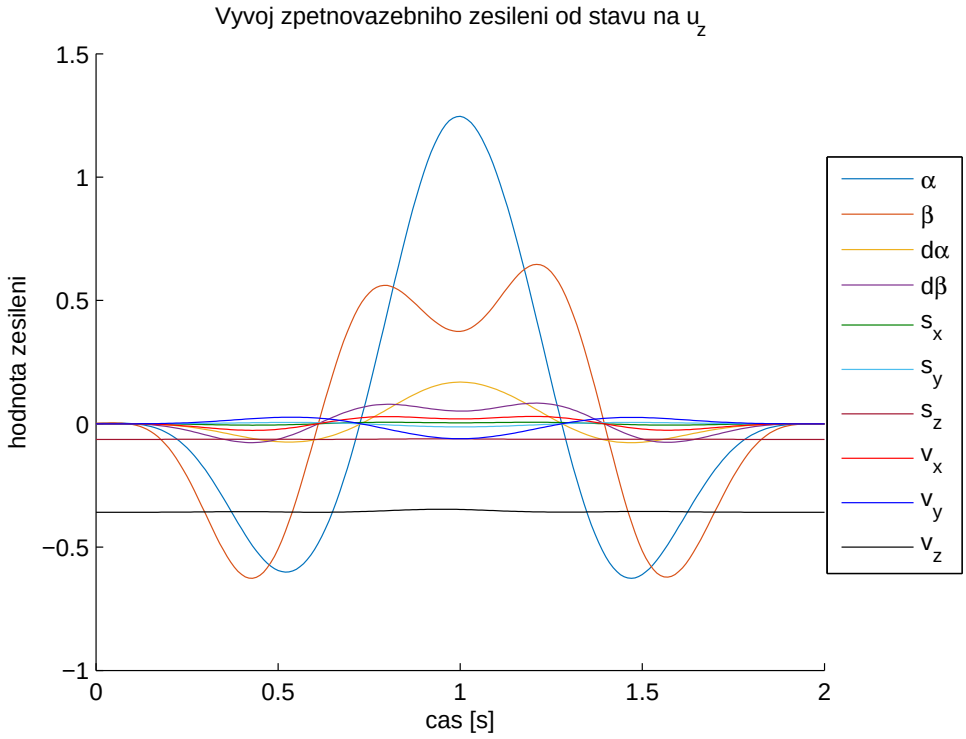
Tak jako v případě první trajektorie, i nyní vidíme změny zesílení především od úhlů náklonu kyvadla a v menší míře také od úhlových rychlostí. Vzhledem k menším náklonům kyvadla v průběhu trajektorie se daly očekávat menší změny zesílení oproti původní trajektorii, což se nám potvrdilo. Stále však platí, že řízení s konstantní zpětnou vazbou pro všechny složky stavu by nejspíš nefungovalo. Chování řízeného systému otestujeme v Simulinku.



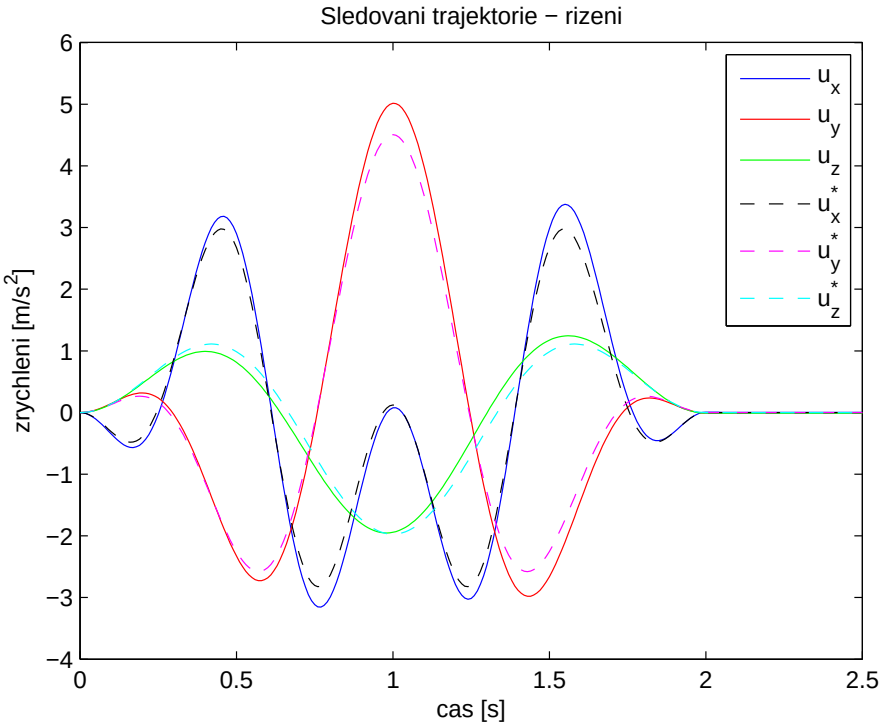
Obrázek 6.2.8: Vývoj zesílení od jednotlivých složek stavu na řízení u_x



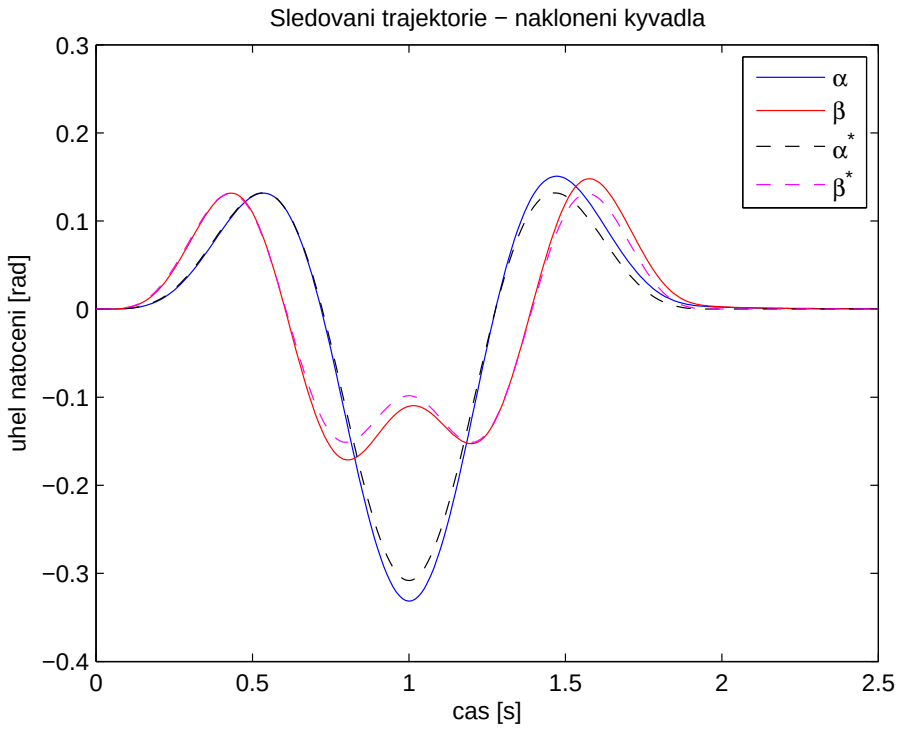
Obrázek 6.2.9: Vývoj zesílení od jednotlivých složek stavu na řízení u_y



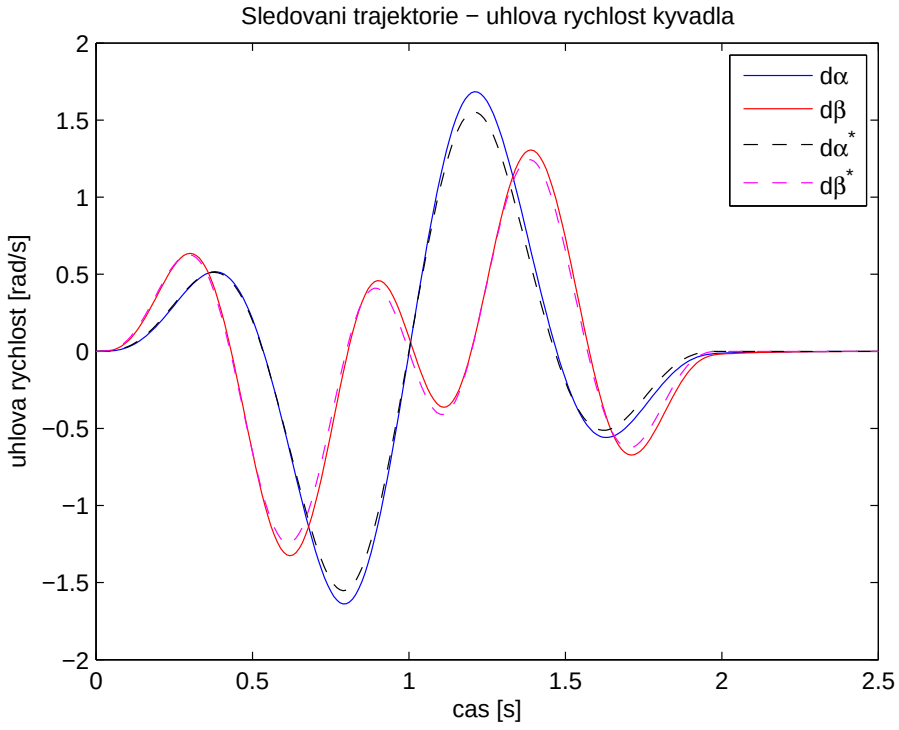
Obrázek 6.2.10: Vývoj zesílení od jednotlivých složek stavu na řízení u_z



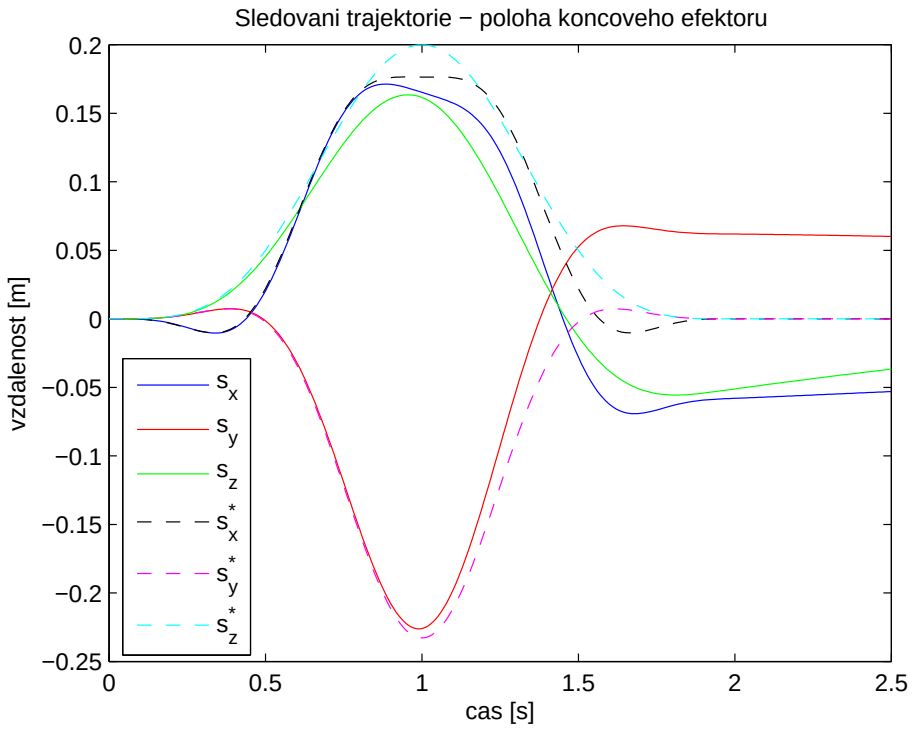
Obrázek 6.2.11: Nová trajektorie - průběh řízení s LQ regulátorem



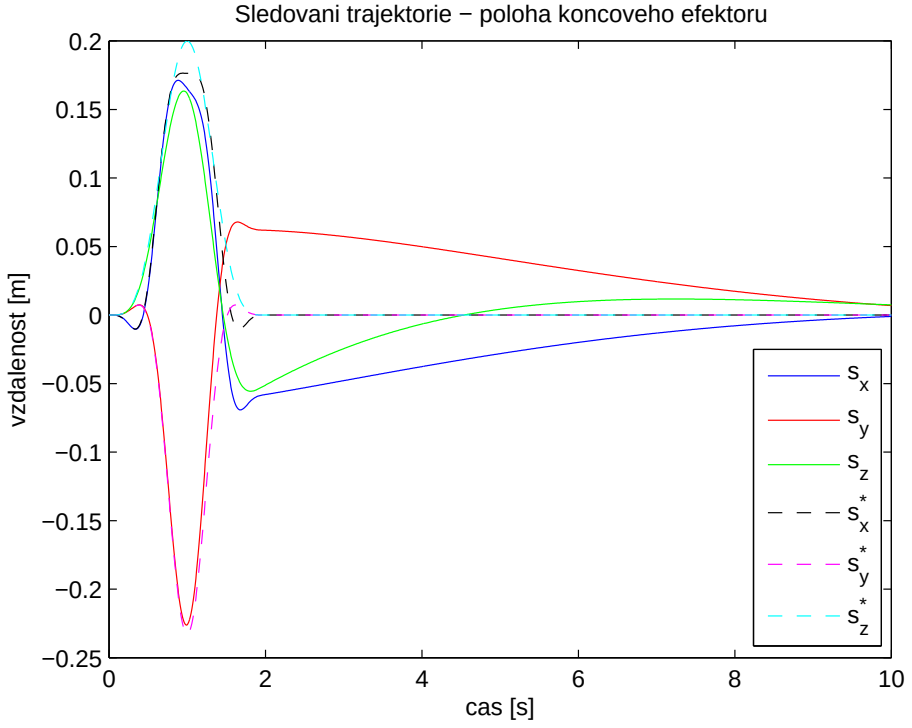
Obrázek 6.2.12: Nová trajektorie - vývoj náklonu kyvadla při řízení s LQ regulátorem



Obrázek 6.2.13: Nová trajektorie - vývoj úhlové rychlosti kyvadla při řízení s LQ regulátorem



Obrázek 6.2.14: Nová trajektorie - poloha pivotu při řízení s LQ regulátorem

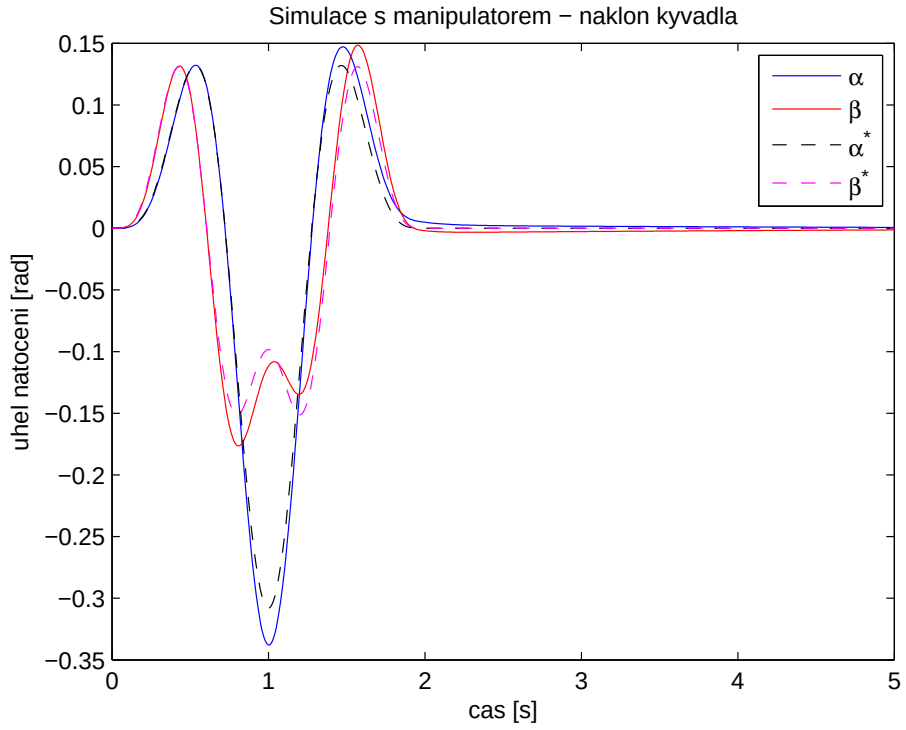


Obrázek 6.2.15: Nová trajektorie - poloha pivotu při řízení s LQ regulátorem na delším časovém úseku

Chování je obdobné jako u první trajektorie. Vidíme, že řízení (obrázek 6.2.11), náklon (obrázek 6.2.12) i úhlová rychlost kyvadla (obrázek 6.2.13) sledují požadovanou trajektorii bez výrazných odchylek a po jejím skončení jsou v blízkém okolí nuly, což odpovídá stabilizaci kyvadla ve vzpřímené poloze. V případě polohy pivotu (obrázek 6.2.14) je situace o něco horší, díky zavedení zpětné vazby od polohy se však koncový efektor postupně vrací zpět do výchozí pozice (obrázek 6.2.15).

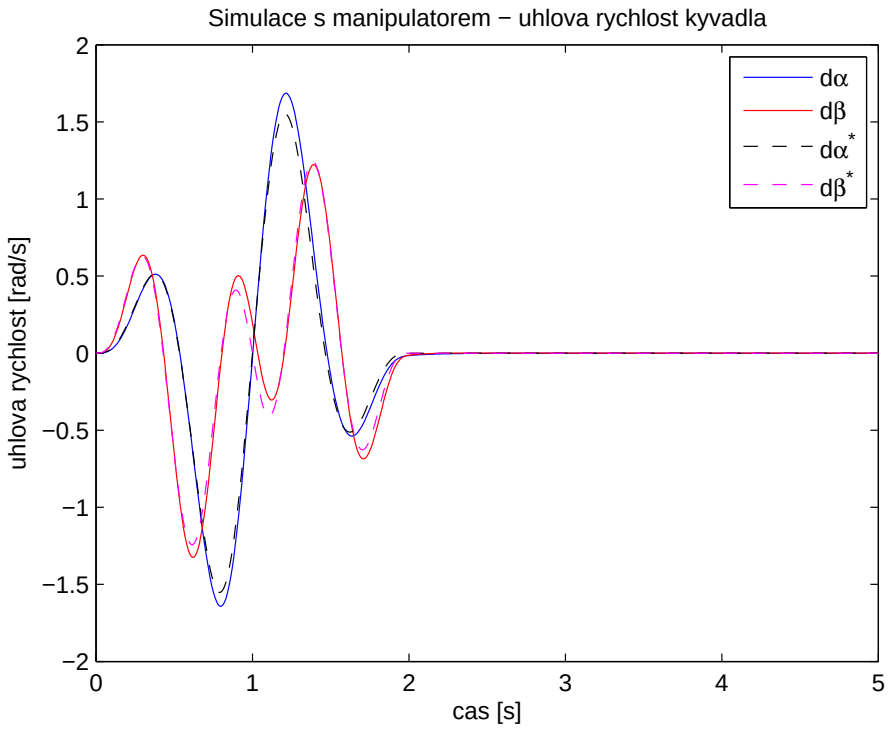
6.2.3 Simulace s novou trajektorií

Máme novou trajektorii a k ní vypočtený LQ regulátor, zbývá zopakovat simulační experiment s modelem manipulátoru v regulační smyčce. S touto trajektorií již simulace proběhla úspěšně, robot neopustil svůj pracovní prostor a kyvadlo ustabilizoval. Celý průběh lze v SimMechanicsu pozorovat v trojrozměrné animaci, zde vykreslíme alespoň záznamy jednotlivých signálů.

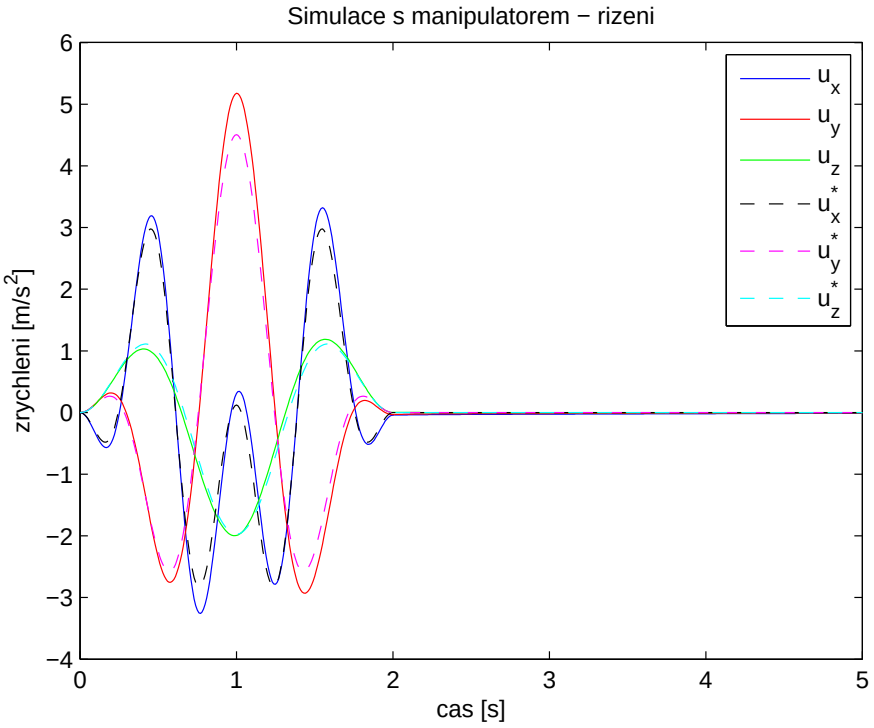


Obrázek 6.2.16: Simulace s manipulátorem - průběh náklonu kyvadla

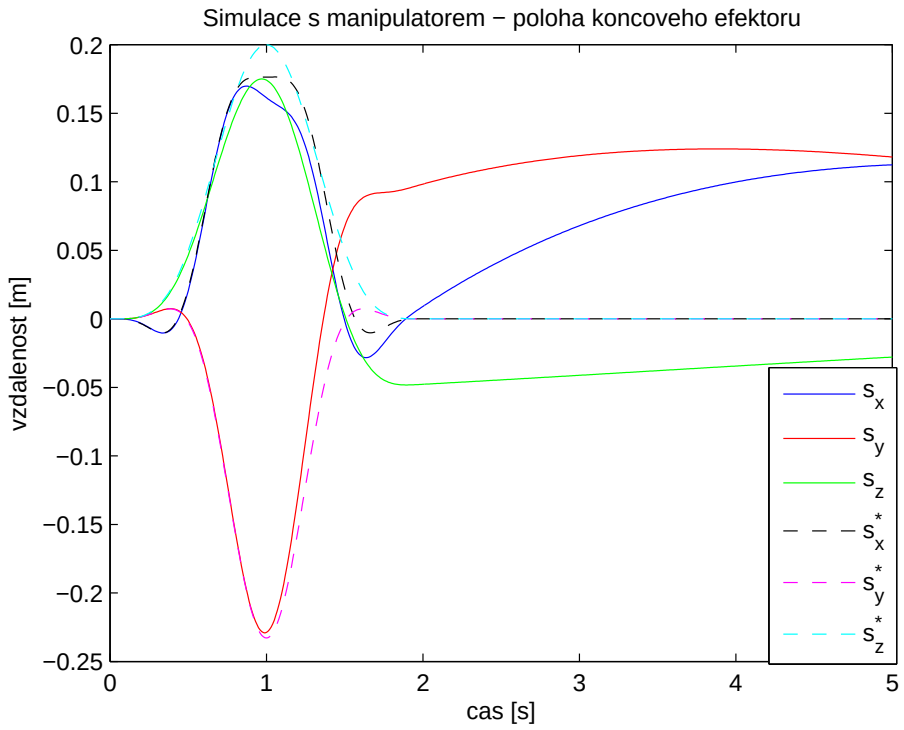
Při porovnání grafů s výsledky simulace pro novou trajektorii bez manipulátoru zjišťujeme, že vliv dynamiky robota rozhodně není zanedbatelný. Nejdůležitější je, že kyvadlo se povedlo ustabilizovat. Náklon kyvadla (obrázek 6.2.16) poměrně přesně kopíruje požadovaný průběh s největšími odchylkami ve špičkách grafu. Po vykonání trajektorie je náklon velmi malý, ne však úplně nulový a trvá zhruba další tři vteřiny, než se úhel zmenší natolik, že není z nepřiblíženého grafu zřetelná odchylka od vzpřímené pozice. Bez výraznějších odchylek je sledována i úhlová rychlost kyvadla (obrázek 6.2.17) a tomu odpovídá i řízení (obrázek 6.2.18), které po skončení trajektorie už jen malými zásahy vyrovná kyvadlo. Naopak mnohem větší odchylky jsou vidět v grafu polohy koncového



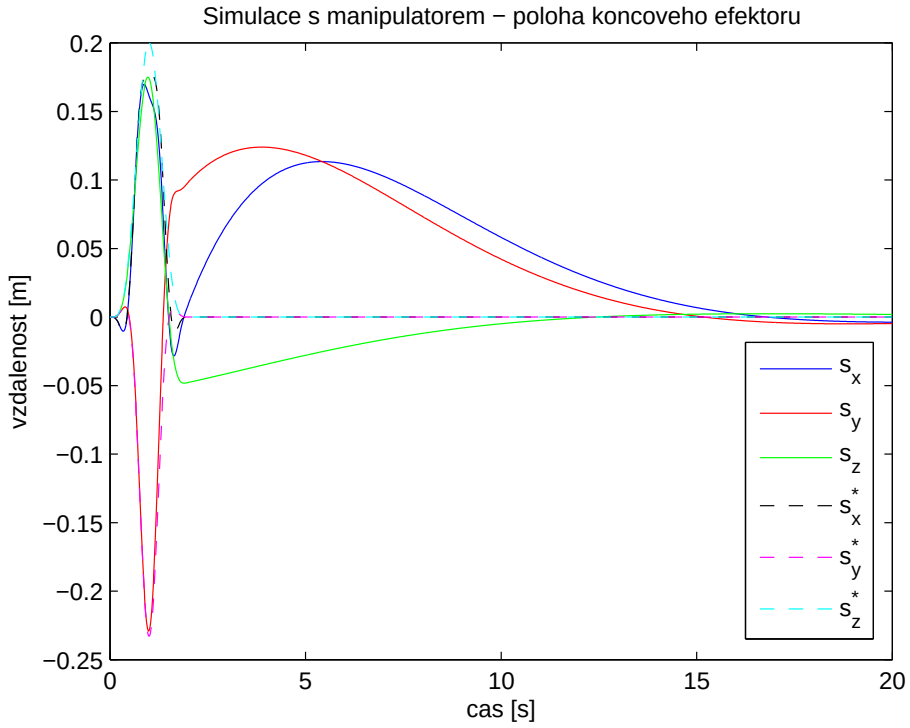
Obrázek 6.2.17: Simulace s manipulátorem - průběh úhlové rychlosti kyvadla



Obrázek 6.2.18: Simulace s manipulátorem - průběh řízení



Obrázek 6.2.19: Simulace s manipulátorem - průběh polohy koncového efektoru



Obrázek 6.2.20: Simulace s manipulátorem - průběh polohy koncového efektoru na delším časovém úseku

efektoru (obrázek 6.2.19), kde se naplno projevila dynamika manipulátoru. Pozice pivotu přesně neodpovídala ani při simulaci bez manipulátoru, po jeho přidání do regulační smyčky je ale situace ještě o něco horší. I zde však zafunguje zpětná vazba od polohy a regulátor pomalu vrátí koncový efektor do počáteční pozice (obrázek 6.2.20), samozřejmě tak, aby při tom nevyvedl kyvadlo ze vzpřímené polohy. Jak je z grafu vidět, celá tato operace trvá téměř dvacet vteřin. Chování by nejspíš šlo zlepšit lepším nastavením regulátoru, což ale, jak jsme již psali, není triviální problém a ve výsledku by se jednalo o zdlouhavé nastavování metodou pokus/omyl. Teoreticky by také mohlo pomoci dále rozšířit stav o integraci odchylky polohy pivotu. Další možností by bylo, při znalosti dynamiky manipulátoru, její zahrnutí do návrhu regulátoru.

7 Ověření na reálném systému

Měli jsme v plánu ověřit funkčnost navrženého řízení i na reálném systému. Na koncový efektor manipulátoru Stäubli TX 40 by byl připevněn hrot, na kterém by poté robot balancoval míč, případně kyvadlo. Řízení mělo probíhat s využitím řídicího systému REX, proto jsme simulační schéma sestavovali i v něm. Problém je však ve snímání polohy, respektive náklonu balancovaného objektu. Jednou možností měření by byla soustava ultrazvukových senzorů vzdálenosti, která byla použita na systému „robotického lachtana“ v [5], případně by místo ultrazvukových senzorů šly použít laserové snímače. V našem případě jsme však chtěli polohu snímat pomocí kamery. S jednou kamerou, která by snímala náklon míče shora (případně zespoda), by bylo možné řídit celý systém alespoň v rovině. Bohužel v okamžiku zpracovávání této práce nebyla kamera k dispozici, musíme se tedy smířit s výsledky pouze ve formě simulace. Tímto zůstává prostor k případné další práci na tomto problému.

8 Závěr

V této práci jsme se zabývali balancováním míče na průmyslovém robotu. Nejprve bylo nutné sestavit matematický model míče na tenkém hrotu. Pohybové rovnice systému jsme odvodili pomocí Newton-Eulerovy metody, přičemž jsme míč modelovali jako matematické kyvadlo na sférickém kloubu. Odtud bylo později možné přejít k fyzickému kyvadlu a případně dále k míči. Pro zvolený stav v podobě úhlů otočení kyvadla kolem os x, respektive y a jim odpovídajícím úhlovým rychlostem a řízení v podobě zrychlení pivotu kyvadla jsme získali nelineární t-variantní systém.

Kyvadlo jsme nechtěli pouze stabilizovat ve vzpřímené poloze, ale požadovali jsme sledování nějaké trajektorie. Protože jsme po této trajektorii chtěli jezdit rychle, bylo nutné vypočítat, jak má být kyvadlo při tomto pohybu nakloněno, aby nespadlo. Tím jsme se dostali k řešení dvoubodové okrajové úlohy. V ní jsme požadovali, aby kyvadlo bylo na začátku i konci trajektorie ve vzpřímené poloze. Z toho plynuly požadavky na rychlost a zrychlení pivotu, které také musely být na začátku a na konci nulové. Dále jsme požadovali pohyb po uzavřené křivce, stejné okrajové podmínky tedy platily i pro polohu pivotu. Na základě těchto požadavků jsme vhodným součtem harmonických složek sestavili funkci řízení a řešením dvoubodové okrajové úlohy (v MATLABu pomocí funkce

bvp4c()) jsme získali vhodnou trajektorii pivotu a k pohybu po ní odpovídající průběhy náklonu a úhlové rychlosti kyvadla.

K řízení jsme chtěli použít lineárně-kvadratický regulátor, systém tedy bylo nutné linearizovat. Protože jsme požadovali stabilizaci kyvadla při určitém pohybu, bylo nutné provést linearizaci systému podél zvolené trajektorie. Výsledný linearizovaný t-variantní spojitý model bylo pro zamýšlené použití na reálném systému třeba zdiskretizovat. Poté jsme mohli navrhnout LQ regulátor na konečném horizontu pro diskrétní systémy. Řešením LQ úlohy jsme obdrželi časově proměnnou stavovou zpětnou vazbu. Použití ustáleného řešení by nebylo možné, neboť kyvadlo se při pohybu po trajektorii poměrně výrazně odchyluje od vzpřímené polohy, linearizace v tomto pracovním bodě by tedy neodpovídala skutečnosti. Hlavním požadavkem na řízení byla stabilizace kyvadla. V takovém případě by se však pivot mohl dostat daleko od zamýšlené polohy. Z tohoto důvodu jsme systém rozšířili o model pohybu pivotu a stav se tedy rozšířil o polohy a rychlosti pivotu v jednotlivých osách. Zavedením zpětné vazby od polohy poté bylo zajištěno, že regulátor bude mít alespoň nějakou snahu vrátit pivot do požadované pozice. Navržené řízení jsme následně ozkoušeli simulačně na modelech sestavených v programech Simulink a REX.

Dále jsme se dostali k robotické části práce. Nejprve jsme probrali teoretické základy popisu geometrického uspořádání a kinematiky manipulátorů. Poté byl sestaven model robota Stäubli TX 40 v programu SimMechanics. Následovala simulace s manipulátorem v regulační smyčce. Při ní jsme zjistili, že vlivem dynamiky manipulátoru dochází k větší odchylce polohy koncového efektoru od požadované trajektorie a nevešli jsme se tak do pracovního prostoru robota. Museli jsme se tedy vrátit k řešení dvoubodové okrajové úlohy a vypočítat novou, prostorově úspornější trajektorii. Podél této trajektorie jsme poté opět linearizovali systém a vypočetli jsme zpětnovazební řízení. S novou trajektorií již simulace s modelem manipulátoru v regulační smyčce proběhla bez problémů. Hlavním výsledkem je, že regulátor zajistil stabilní pohyb kyvadla po požadované trajektorii a i po jejím skončení kyvadlo ustabilizoval ve vzpřímené poloze. Poloha koncového efektoru se kvůli snaze o stabilizaci kyvadla znatelně odchyluje od požadované pozice, zejména v druhé polovině trajektorie a po jejím skončení. Díky zavedení zpětné vazby od polohy se ale koncový efektor postupně stahuje ke své požadované pozici. Nastavení regulátoru není naprosto ideální, v simulaci ale regulátor kyvadlo ustabilizuje a k dalšímu vylepšování by docházelo až při práci na reálném systému, neboť ten se od modelu bude stejně lišit. Bohužel, k ověření řízení na reálném systému nedošlo, protože jsme neměli k dispozici kameru, pomocí které jsme chtěli snímat stav kyvadla, respektive míče. Alespoň prozatím jsme se tedy museli spokojit s výsledkem ve formě úspěšné simulace.

Reference

[1] Mark W. Spong, Seth Hutchinson a M. Vidyasagar: Robot Modeling and Control, First Edition, John Wiley and Sons, Inc., 2005

[2] Russ Tedrake: Underactuated Robotics: Algorithms for Walking, Running, Swimming, Flying, and Manipulation (Course Notes for MIT 6.832) [vid. 26.4.2017]. Dostupné z: http://underactuated.mit.edu/

[3] http://www.honda.co.jp/ASIMO/about/ [vid. 26.4.17]

[4] Steven H. Collins, Martijn Wisse a Andy Ruina: A three-dimensional passive-dynamic walking robot with two legs and knees, International Journal of Robotics Research, July 2001, str. 607-615.

[5] Martin Goubej, Miloš Schlegel a Radek Škarda: Mechatronic model for education - Robotic sea lion

[6] http://web.mit.edu/8.01t/www/materials/modules/old_guide/guide16Appendix.pdf [vid. 27.5.2017]

[7] Lawrence F. Shampine, Jacek Kierzenka a Mark W. Reichelt: Solving Boundary Value Problems for Ordinary Diferential Equations in Matlab with bvp4c, October 26. 2000

[8] Miloš Schlegel: LQ regulátor, Doplňující text k předmětu Průmyslové systémy (KKY/PS), 2004

[9] Martin Goubej: LQR a LQG řízení, Přednášky k předmětu Robustní řízení lineárních systémů (KKY/RLS), 2016

[10] Matin Švejda: Mechatronika a robotika jako vědní disciplína, Přednáška č.1 k předmětu Úvod do robotiky a mechatroniky (KKY/URM), 2014

[11] Martin Švejda: Optimalizace robotických architektur, Disertační práce, 2016

[12] Martin Švejda: Inverzní geometrický model SM - metody řešení, příklady, Přednáška č.4 k předmětu Úvod do robotiky a mechatroniky (KKY/URM), 2013

[13] Martin Švejda: Přímý a inverzní kinematický model sériových manipulátorů, singulární polohy, Přednáška č.4 k předmětu Úvod do robotiky a mechatroniky (KKY/URM), 2013

Seznam obrázků

DODATKY

Dodatek 1 - Dokumentace k robotu Stäubli TX40 - konstrukční omezení

2.5. PERFORMANCE

See figures **2.6**, **2.7** (1) Brake release access area

	Standard arm
Work envelope	
R.M max. reach between axis 1 and 5	450 mm
R.m1 min. reach between axis 1 and 5	151 mm
R.m2 min. reach between axis 2 and 5	162 mm
R.b reach between axis 3 and 5	225 mm
Maximum speed at load center of gravity	8.2 m/s
Placing **Repeatability** in accordance with ISO 9283	± 0.02 mm

2.5.1. RANGE, SPEED AND RESOLUTION

Axis	1	2	3	4	5	6
Range (°)	360	250	276	540	253.5	540 (1)
Working range distribution (°)	A ± 180	B ± 125	C ± 138	D ± 270	E + 133.5 - 120	F ± 270
Nominal speed (°/s)	287	287	430	410	320	700
Maximum speed (°/s) (2)	555	475	585	1035	1135	1575
Angular resolution (°.10^{-3})	0.057	0.057	0.122	0.114	0.122	0.172

(1) Can be configured by software up to ± 18 000°. See the "Software configuration" chapter in the "Controller" documentation. Can be configured as a "continuous" axis under the continuousAxis license.
(2) Maximum speed for reduced conditions of load and inertia.

Low speed in manual mode: 250 mm/s at tool centre point and 45 °/s on each axis.

CAUTION:
In some arm configurations, the maximum joint speeds can be reached only if payloads and inertias are reduced.

Dodatek 2 - Skript na výpočet dvoubodové okrajové úlohy v MATLABu

```
%funkce pro vypocet trajektorie
global l m g J T w t

l = 0.1;
m = 0.2;
g = -9.81;
r2 = 0.1;   %vnejsi polomer mice = l
r1 = r2-0.002;  %vnitrni polomer mice
J = (2/5) * m * ((r2^5-r1^5)/(r2^3-r1^3))  %moment kolem stredu
T = 2;
w = 2*pi/T;
t = 0 : 0.01 : T;

options = bvpset('RelTol', 1e-10);  %nastaveni tolerance

% inicializace - mesh, odhad reseni, odhad parametru
% solinit = bvpinit(t,[0.015 0.015 0 0],0.4*ones(1,4)); %prvni trajektorie
solinit = bvpinit(t,[0.01 0.01 0 0],0.699*ones(1,4));   %nova trajektorie

sol = bvp4c(@kyvadlo_ode,@kyvadlo_bc,solinit,options);

%vykresleni vysledku
y = deval(sol,t);
plot(t,y(1,:),'b')
hold on
plot(t,y(2,:),'r')

%rizeni
ux = 0.1*w^2*cos(w*t) + sol.parameters(1)*(2*w)^2*cos(2*w*t) +...
    sol.parameters(2)*(3*w)^2*cos(3*w*t) -...
    ((0.1+sol.parameters(1)*2^2+sol.parameters(2)*3^2)/4^2)*...
    (4*w)^2*cos(4*w*t);
uy = sol.parameters(3)*w^2*cos(w*t) + sol.parameters(4)*(2*w)^2*cos(2*w*t)...
    - ((sol.parameters(3)+sol.parameters(4)*2^2)/3^2)*(3*w)^2*cos(3*w*t);
uz = 0.1*w^2*cos(w*t) - (0.1/2^2)*(2*w)^2*cos(2*w*t);

%poloha efektoru
sx = 0.1*(1-cos(w*t)) + sol.parameters(1)*(1-cos(2*w*t)) +...
    sol.parameters(2)*(1-cos(3*w*t)) -...
    ((0.1+sol.parameters(1)*2^2+sol.parameters(2)*3^2)/4^2)*...
    (1-cos(4*w*t));
sy = sol.parameters(3)*(1-cos(w*t)) + sol.parameters(4)*(1-cos(2*w*t)) -...
    ((sol.parameters(3)+sol.parameters(4)*2^2)/3^2)*(1-cos(3*w*t));
sz = 0.1*(1-cos(w*t)) - (0.1/2^2)*(1-cos(2*w*t));

%rychlost efektoru
vx = 0.1*w*sin(w*t) + sol.parameters(1)*(2*w)*sin(2*w*t) +...
```

```
    sol.parameters(2)*(3*w)*sin(3*w*t) -...
    ((0.1+sol.parameters(1)*2^2+sol.parameters(2)*3^2)/4^2)*(4*w)*sin(4*w*t);
vy = sol.parameters(3)*w*sin(w*t) + sol.parameters(4)*(2*w)*sin(2*w*t) -...
    ((sol.parameters(3)+sol.parameters(4)*2^2)/3^2)*(3*w)*sin(3*w*t);
vz = 0.1*w*sin(w*t) - (0.1/2^2)*(2*w)*sin(2*w*t);


% ------------------------------------------------------

function dydx = kyvadlo_ode(x,y,par)
%ODE function - model kyvadla (mice)

global l m g J T w t

%uzavrena trajektorie
ux = 0.1*w^2*cos(w*x) + par(1)*(2*w)^2*cos(2*w*x) +...
    par(2)*(3*w)^2*cos(3*w*x) -...
    ((0.1+par(1)*2^2+par(2)*3^2)/4^2)*(4*w)^2*cos(4*w*x);
uy = par(3)*w^2*cos(w*x) + par(4)*(2*w)^2*cos(2*w*x) -...
    ((par(3)+par(4)*2^2)/3^2)*(3*w)^2*cos(3*w*x);
uz = 0.1*w^2*cos(w*x) - (0.1/2^2)*(2*w)^2*cos(2*w*x);
u = [ux; uy; uz];

dydx = [y(3);
        y(4);
        (2*sin(y(2))*y(3)*y(4)*l^2*m+2*J*sin(y(2))*y(3)*y(4)+...
        cos(y(1))*l*m*u(2)-sin(y(1))*g*m*l+sin(y(1))*l*m*u(3))/...
        (cos(y(2))*(l^2*m+J));
        (-sin(y(2))*cos(y(2))*y(3)^2*l^2*m-...
        J*sin(y(2))*cos(y(2))*y(3)^2-...
        cos(y(1))*sin(y(2))*g*l*m+cos(y(1))*sin(y(2))*l*m*u(3)-...
        sin(y(2))*sin(y(1))*l*m*u(2)-cos(y(2))*l*m*u(1))/(l^2*m+J)];


% ------------------------------------------------------

function res = kyvadlo_bc(ya,yb,par)
%okrajove podminky

res = [ya(1);
        ya(2);
        ya(3);
        ya(4);
        yb(1);
        yb(2);
        yb(3);
        yb(4)];
```

Dodatek 3 - Skript na výpočet LQR na konečném horizontu pro diskrétní systémy v MATLABu

```
%time variant state feedback for ball balancing
% !!! predem spustit funkci pro vypocet trajektorie

Ts=0.01;    % sampling period of control
t = t';

x1s = y(1,:)'; %stav kyvadla/mice
x2s = y(2,:)';
x3s = y(3,:)';
x4s = y(4,:)';
x5s = sx';   %poloha koncoveho efektoru
x6s = sy';
x7s = sz';
x8s = vx';  %rychlost koncoveho efektoru
x9s = vy';
x10s = vz';

nn=length(t);
N=nn-1;
Ks=[];

Q1 = 100;
Q2 = 100;
Q3 = 1;
Q4 = 1;
Q5 = 1;
Q6 = 1;
Q7 = 20;
Q8 = 1;
Q9 = 1;
Q10 = 1;
R1 = 100;
R2 = 200;
R3 = 700;
Q = diag([Q1,Q2,Q3,Q4,Q5,Q6,Q7,Q8,Q9,Q10]);
RO = diag([R1,R2,R3]);

tt=t(end);
ttp = 0;
for k=N+1:-1:1
    ttp=ttp+Ts;
    tt=tt-Ts;

    switch k
        case N+1,
                q1 = 0;
```

```
        q2 = 0;
        q3 = 0;
        q4 = 0;
        q5 = 0;
        q6 = 0;
        q7 = 0;
        q8 = 0;
        q9 = 0;
        q10 = 0;
        u1 = 0;
        u2 = 0;
        u3 = 0;

        Aue = [0 0 1 0 0 0 0 0 0 0;
            0 0 0 1 0 0 0 0 0 0;
            (-g*m*l)/(l^2*m+J), 0, 0, 0, 0, 0, 0, 0, 0, 0;
            0, (-g*l*m)/(l^2*m+J), 0, 0, 0, 0, 0, 0, 0, 0;
            0 0 0 0 0 0 0 1 0 0;
            0 0 0 0 0 0 0 0 1 0;
            0 0 0 0 0 0 0 0 0 1;
            0 0 0 0 0 0 0 0 0 0;
            0 0 0 0 0 0 0 0 0 0;
            0 0 0 0 0 0 0 0 0 0];
        Bue = [0 0 0;
            0 0 0;
            0, l*m/(l^2*m+J), 0;
            -l*m/(l^2*m+J), 0, 0;
            0 0 0;
            0 0 0;
            0 0 0;
            1 0 0;
            0 1 0;
            0 0 1];

        F = eye(size(Aue))+(Ts*Aue);
        G = Ts*Bue;
        [Pkp1,L,Kde] = dare(F,G,Q,RO);
        Ks=[-Kde,Ks];
        eig(Aue-Bue*Kde)
otherwise,
        q1 = x1s(k);
        q2 = x2s(k);
        q3 = x3s(k);
        q4 = x4s(k);
        q5 = x5s(k);
        q6 = x6s(k);
        q7 = x7s(k);
        q8 = x8s(k);
        q9 = x9s(k);
```

```
q10 = x10s(k);
u1 = ux(k);
u2 = uy(k);
u3 = uz(k);

Ak = [0 0 1 0 0 0 0 0 0 0;
    0 0 0 1 0 0 0 0 0 0;
    (-sin(q1)*l*m*u2-cos(q1)*g*m*l+cos(q1)*l*m*u3)/...
    (cos(q2)*(l^2*m+J)), ...
    (2*sin(q2)*q3*q4*l^2*m+2*J*sin(q2)*q3*q4+...
    cos(q1)*l*m*u2-sin(q1)*g*m*l+sin(q1)*l*m*u3)*...
    sin(q2)/(cos(q2)^2*(l^2*m+J))+...
    (2*cos(q2)*q3*q4*l^2*m+2*J*cos(q2)*q3*q4)/...
    (cos(q2)*(l^2*m+J)), ...
    (2*sin(q2)*q4*l^2*m+2*J*sin(q2)*q4)/...
    (cos(q2)*(l^2*m+J)), ...
    (2*sin(q2)*q3*l^2*m+2*J*sin(q2)*q3)/...
    (cos(q2)*(l^2*m+J)), 0, 0, 0, 0, 0, 0;
    (sin(q1)*sin(q2)*g*l*m-sin(q1)*sin(q2)*l*m*u3-...
    sin(q2)*cos(q1)*l*m*u2)/(l^2*m+J), ...
    (-cos(q2)^2*q3^2*l^2*m+sin(q2)^2*q3^2*l^2*m...
    -J*cos(q2)^2*q3^2+J*sin(q2)^2*q3^2-...
    cos(q1)*cos(q2)*g*l*m+cos(q1)*cos(q2)*l*m*u3-...
    cos(q2)*sin(q1)*l*m*u2+sin(q2)*l*m*u1)/(l^2*m+J), ...
    (-2*sin(q2)*cos(q2)*q3*l^2*m-2*J*sin(q2)*...
    cos(q2)*q3)/(l^2*m+J), 0, 0, 0, 0, 0, 0, 0;
    0 0 0 0 0 0 0 1 0 0;
    0 0 0 0 0 0 0 0 1 0;
    0 0 0 0 0 0 0 0 0 1;
    0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0];
Bk = [0 0 0;
    0 0 0;
    0, cos(q1)*l*m/(cos(q2)*(l^2*m+J)), ...
    sin(q1)*l*m/(cos(q2)*(l^2*m+J));
    -cos(q2)*l*m/(l^2*m+J), ...
    -sin(q2)*sin(q1)*l*m/(l^2*m+J), ...
     cos(q1)*sin(q2)*l*m/(l^2*m+J);
    0 0 0;
    0 0 0;
    0 0 0;
    1 0 0;
    0 1 0;
    0 0 1];

Fk =eye(size(Ak))+(Ts*Ak);
Gk = Ts*Bk;
KkT=-inv(RO+Gk'*Pkp1*Gk)*Gk'*Pkp1*Fk;
```

```
                Pk=(Q+Fk'*Pkp1*Fk)+(Gk'*Pkp1*Fk)'*KkT;
                Pkp1=Pk;
                Ks=[KkT;Ks];
    end;
end;

%vytvoreni casove rady pro simulaci v Simulinku
Ks_sim = Ks(1:3,:);
for k = 1:1:N
    Ks_sim(:,:,k+1) = Ks(3*k+1:3*k+3,:);
end
Ks_sim = timeseries(Ks_sim,t);

S = [sx;sy;sz];
S = timeseries(S,t);

%pridani casove znacky ke kazdemu radku K pro pouziti v RexLangu
for k = 0:1:N
    tKs(3*k+1,1) = t(k+1);
    tKs(3*k+2,1) = t(k+1);
    tKs(3*k+3,1) = t(k+1);
end

%ulozeni vysledku pro simulaci v REXu
trajectory_data = [t,ux',uy',uz',x1s,x2s,x3s,x4s,x5s,x6s,x7s,x8s,x9s,x10s];
reg_data = [tKs,Ks];
save('trajectoryData.csv','-ASCII', '-DOUBLE','-TABS','trajectory_data');
save('kyvadlo_reg.csv','-ASCII', '-DOUBLE','-TABS','reg_data');
```

Dodatek 4 - Kód pro blok REXLANG pro načtení signálů požadované trajektorie (jazyk podobný C)

```
//assigning inputs to variables, these variables are READ-ONLY
bool input(0) RUN; //value from u0 input
//double input(1) secondinput; //value from u1 input
//add inputs as needed

//assigning variables to outputs, these variables are WRITE-ONLY
double output(0) ux;    //value to send to y0 output
double output(1) uy;    //value to send to y1 output
double output(2) uz;    //value to send to y2 output
double output(3) a;     //value to send to y3 output
double output(4) b;     //value to send to y4 output
double output(5) da;    //value to send to y5 output
double output(6) db;    //value to send to y6 output
double output(7) sx;    //value to send to y7 output
double output(8) sy;    //value to send to y8 output
double output(9) sz;    //value to send to y9 output
//add output signals as needed

#define N 14    //pocet sloupcu

double tRun;    //cas od zacatku behu
int  status=0;  //stav bloku (faze algoritmu)
int  idx;       //aktualni radka v souboru s daty
int nmax=201;   //aktualni delka souboru s daty
double data[201][N+1];  //pole, kam se kopiruji data trajektorie
//!!!parseovanim souboru z MATLABu blokem CNA vznikne parasitne
//1 sloupec navic (proto N+1)

#define data_t 0
#define data_ux 1
#define data_uy 2
#define data_uz 3
#define data_a 4
#define data_b 5
#define data_da 6
#define data_db 7
#define data_sx 8
#define data_sy 9
#define data_sz 10
#define data_vx 11
#define data_vy 12
#define data_vz 13

void SetData(void)
{
    int i, j;
```

```
    char sData[30];

    sData=".CNA_data1:acn[";
    for(i=0;i<nmax;++i)
    {
        for(j=0;j<N+1;++j)  //N+1 kvuli parazitnimu sloupci
        {
            data[i][j]=GetExtDouble(sData+long2str((N+1)*i+j)+"]");
        }
        Suspend(0.01);  //aby nedoslo k prekroceni periody pri nacitani dat
    }
}

int parchange(void)
{
    return 0;
}

//the init procedure is executed once when the REXLANG function block initializes
long init(void)
{
    parchange();
    SetData();
    return 0;
}

//the main procedure is executed repeatedly (once in each sampling period)
long main(void)
{
    switch(status)
    {
    case 0:  //cekani na zahajeni
        tRun = 0.0;
        idx = 0;
        if(RUN==0)
            break;
        status = 1;
    case 1:  //nacteni vygenerovane trajetorie
        while(idx<nmax-1 && tRun>data[idx+1][data_t])
            ++idx;
        if(tRun<=data[nmax-1][data_t])
        {
            ux = data[idx][data_ux];
            uy = data[idx][data_uy];
            uz = data[idx][data_uz];
            a = data[idx][data_a];
            b = data[idx][data_b];
            da = data[idx][data_da];
            db = data[idx][data_db];
```

```
            sx = data[idx][data_sx];
            sy = data[idx][data_sy];
            sz = data[idx][data_sz];
        }
        else
        {
            ux = 0;
            uy = 0;
            uz = 0;
            a = 0;
            b = 0;
            da = 0;
            db = 0;
            sx = 0;
            sy = 0;
            sz = 0;
        }
        tRun += GetPeriod();
        break;
    }
    return 0;
}

//the exit procedure is executed once when the task is correctly terminated
// (system shutdown, downloading new control algorithm, etc.)
long exit(void)
{
    /* PUT YOUR CODE HERE */
    return 0;
}
```

Dodatek 5 - Kód pro blok REXLANG pro načtení časově proměnné matice stavové zpětné vazby (jazyk podobný C)

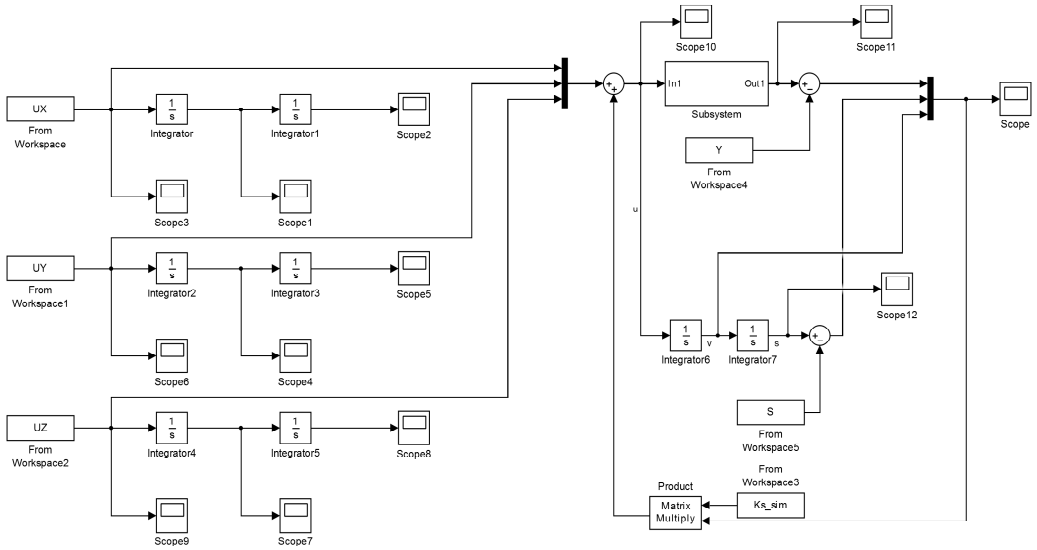
```
//assigning inputs to variables, these variables are READ-ONLY
double input(0) a;     //value from u0 input
double input(1) b;     //value from u1 input
double input(2) da;    //value from u2 input
double input(3) db;    //value from u3 input
double input(4) sx;    //value from u4 input
double input(5) sy;    //value from u5 input
double input(6) sz;    //value from u6 input
double input(7) vx;    //value from u7 input
double input(8) vy;    //value from u8 input
double input(9) vz;    //value from u9 input

bool input(10) RUN;    //value from u10 input
//add inputs as needed

//assigning variables to outputs, these variables are WRITE-ONLY
double output(0) ux;    //value to send to y0 output
double output(1) uy;    //value to send to y1 output
double output(2) uz;    //value to send to y2 output
//add output signals as needed

#define N 11    //pocet sloupcu

double tRun;    //cas od zacatku behu
int  status=0;  //stav bloku (faze algoritmu)
int  idx;       //aktualni radka v souboru s daty
int nmax=603;  //delka souboru s daty (pocet radek)
double KS[603][N+1];
//!!!parseovanim souboru z MATLABu blokem CNA
//vznikne parasitne 1 sloupec navic

#define KS_t 0
#define KS_a 1
#define KS_b 2
#define KS_da 3
#define KS_db 4
#define KS_sx 5
#define KS_sy 6
#define KS_sz 7
#define KS_vx 8
#define KS_vy 9
#define KS_vz 10

void SetData(void)
{
    int i, j;
```

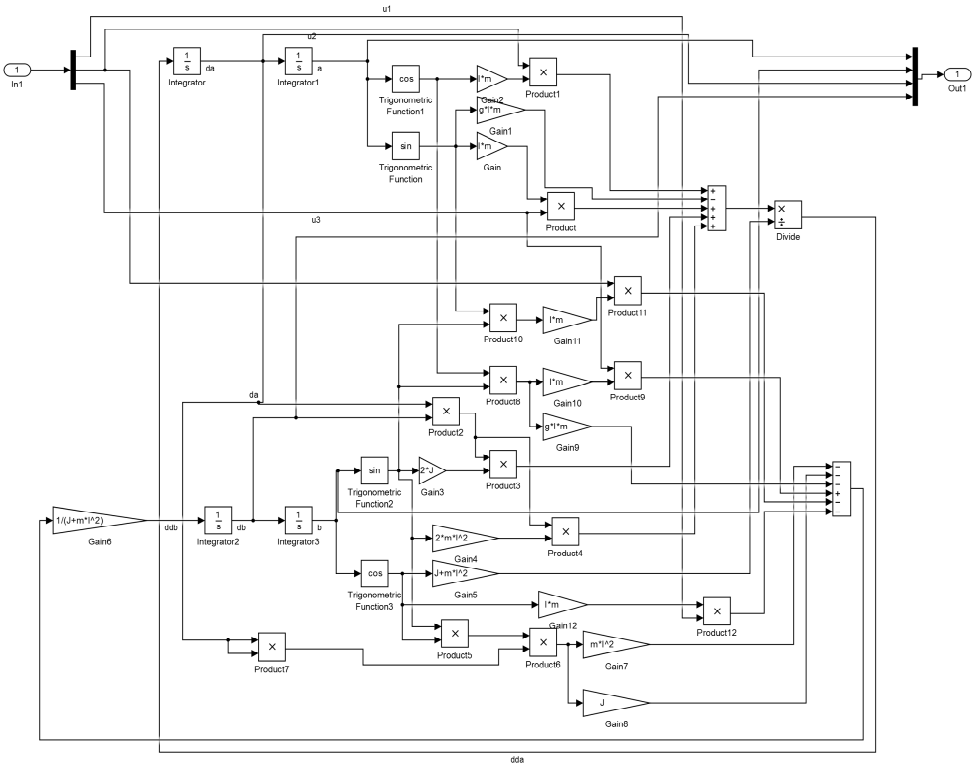
```
    char sData[30];

    sData=".CNA_reg1:acn[";
    for(i=0;i<nmax;++i)
    {
        for(j=0;j<N+1;++j)  //N+1 kvuli parazitnimu sloupci
        {
            KS[i][j]=GetExtDouble(sData+long2str((N+1)*i+j)+"]");
        }
        Suspend(0.01);  //aby nedoslo k prekroceni periody pri nacitani dat
    }
}

int parchange(void)
{
    return 0;
}

//the init procedure is executed once when the REXLANG function block initializes
long init(void)
{
    parchange();
    SetData();
    return 0;
}

//the main procedure is executed repeatedly (once in each sampling period)
long main(void)
{
    switch(status)
    {
    case 0:  //ustaleny stav, cekani na zahajeni
        tRun = 0.0;
        idx = 0;
        if(RUN==0)
        {
            ux = KS[nmax-3][KS_a]*a + KS[nmax-3][KS_b]*b +
                 KS[nmax-3][KS_da]*da + KS[nmax-3][KS_db]*db +
                 KS[nmax-3][KS_sx]*sx + KS[nmax-3][KS_sy]*sy +
                 KS[nmax-3][KS_sz]*sz + KS[nmax-3][KS_vx]*vx +
                 KS[nmax-3][KS_vy]*vy + KS[nmax-3][KS_vz]*vz;

            uy = KS[nmax-2][KS_a]*a + KS[nmax-2][KS_b]*b +
                 KS[nmax-2][KS_da]*da + KS[nmax-2][KS_db]*db +
                 KS[nmax-2][KS_sx]*sx + KS[nmax-2][KS_sy]*sy +
                 KS[nmax-2][KS_sz]*sz + KS[nmax-2][KS_vx]*vx +
                 KS[nmax-2][KS_vy]*vy + KS[nmax-2][KS_vz]*vz;

            uz = KS[nmax-1][KS_a]*a + KS[nmax-1][KS_b]*b +
```

```
            KS[nmax-1][KS_da]*da + KS[nmax-1][KS_db]*db +
            KS[nmax-1][KS_sx]*sx + KS[nmax-1][KS_sy]*sy +
            KS[nmax-1][KS_sz]*sz + KS[nmax-1][KS_vx]*vx +
            KS[nmax-1][KS_vy]*vy + KS[nmax-1][KS_vz]*vz;
        break;
    }
    status = 1;
case 1:  //sledovani trajektorie
    while(idx<nmax-3 && tRun>KS[idx+3][KS_t])
        idx = idx+3;
    if(tRun<=KS[nmax-1][KS_t])
    {
        ux = KS[idx][KS_a]*a + KS[idx][KS_b]*b +
            KS[idx][KS_da]*da + KS[idx][KS_db]*db +
            KS[idx][KS_sx]*sx + KS[idx][KS_sy]*sy +
            KS[idx][KS_sz]*sz + KS[idx][KS_vx]*vx +
            KS[idx][KS_vy]*vy + KS[idx][KS_vz]*vz;

        uy = KS[idx+1][KS_a]*a + KS[idx+1][KS_b]*b +
            KS[idx+1][KS_da]*da + KS[idx+1][KS_db]*db +
            KS[idx+1][KS_sx]*sx + KS[idx+1][KS_sy]*sy +
            KS[idx+1][KS_sz]*sz + KS[idx+1][KS_vx]*vx +
            KS[idx+1][KS_vy]*vy + KS[idx+1][KS_vz]*vz;

        uz = KS[idx+2][KS_a]*a + KS[idx+2][KS_b]*b +
            KS[idx+2][KS_da]*da + KS[idx+2][KS_db]*db +
            KS[idx+2][KS_sx]*sx + KS[idx+2][KS_sy]*sy +
            KS[idx+2][KS_sz]*sz + KS[idx+2][KS_vx]*vx +
            KS[idx+2][KS_vy]*vy + KS[idx+2][KS_vz]*vz;

    }
    else
    {
        ux = KS[nmax-3][KS_a]*a + KS[nmax-3][KS_b]*b +
            KS[nmax-3][KS_da]*da + KS[nmax-3][KS_db]*db +
            KS[nmax-3][KS_sx]*sx + KS[nmax-3][KS_sy]*sy +
            KS[nmax-3][KS_sz]*sz + KS[nmax-3][KS_vx]*vx +
            KS[nmax-3][KS_vy]*vy + KS[nmax-3][KS_vz]*vz;

        uy = KS[nmax-2][KS_a]*a + KS[nmax-2][KS_b]*b +
            KS[nmax-2][KS_da]*da + KS[nmax-2][KS_db]*db +
            KS[nmax-2][KS_sx]*sx + KS[nmax-2][KS_sy]*sy +
            KS[nmax-2][KS_sz]*sz + KS[nmax-2][KS_vx]*vx +
            KS[nmax-2][KS_vy]*vy + KS[nmax-2][KS_vz]*vz;

        uz = KS[nmax-1][KS_a]*a + KS[nmax-1][KS_b]*b +
            KS[nmax-1][KS_da]*da + KS[nmax-1][KS_db]*db +
            KS[nmax-1][KS_sx]*sx + KS[nmax-1][KS_sy]*sy +
            KS[nmax-1][KS_sz]*sz + KS[nmax-1][KS_vx]*vx +
```

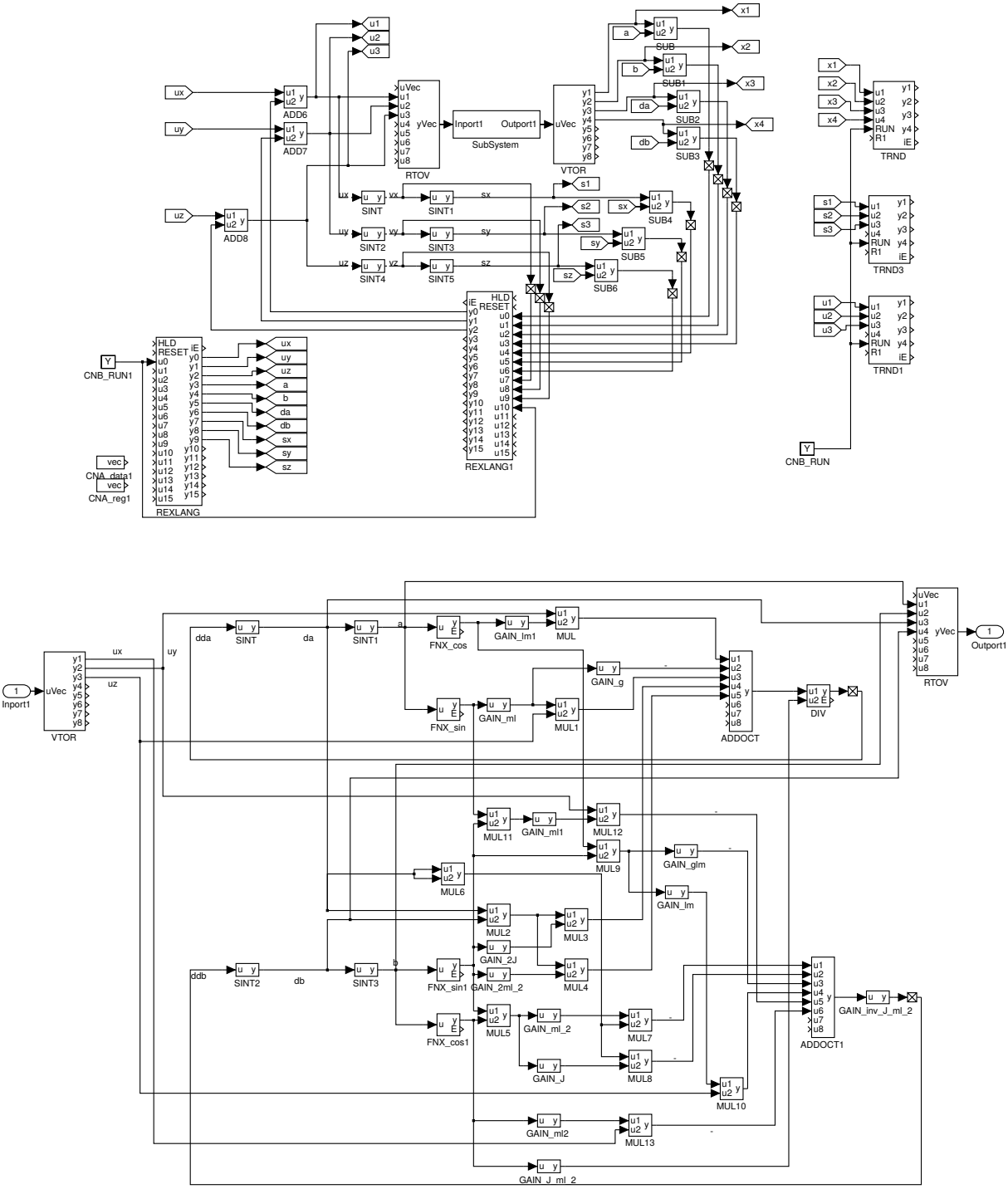
```
                KS[nmax-1][KS_vy]*vy + KS[nmax-1][KS_vz]*vz;
        }
        tRun += GetPeriod();
        break;
    }
    return 0;
}

//the exit procedure is executed once when the task is correctly terminated
// (system shutdown, downloading new control algorithm, etc.)
long exit(void)
{
    /* PUT YOUR CODE HERE */
    return 0;
}
```

Dodatek 6 - Simulační schéma regulační smyčky s LQ regulátorem a subsystému kyvadla na hrotu sestavené v programu Simulink





Dodatek 7 - Simulační schéma regulační smyčky s LQ regulátorem a subsystému kyvadla na hrotu sestavené v programu REX



Dodatek 8 - Simulační schéma regulační smyčky s modelem robotu Stäubli sestaveným v SimMechanicsu s využitím knihovny *robotLib*