FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

ZÁPADOČESKÁ
UNIVERZITA
V PLZNI

Západočeská univerzita v Plzni
Fakulta aplikovaných věd

Přímovazební řízení pro kompenzaci nežádoucích vibrací kabiny výtahu

Bakalářská práce

Jakub Tvrz

KKY/Ing. Martin Goubej Ph.D.

Datum zpracování: 14. května 2023

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2022/2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Jakub TVRZ**
Osobní číslo:	**A20B0357P**
Studijní program:	**B0714A150005 Kybernetika a řídicí technika**
Specializace:	**Automatické řízení a robotika**
Téma práce:	**Přímovazební řízení pro kompenzaci nežádoucích vibrací kabiny výtahu**
Zadávající katedra:	**Katedra kybernetiky**

Zásady pro vypracování

1. Seznamte se s problematikou řízení pohybu výtahů. Sestavte matematický model dynamiky systému.
2. Navrhněte vhodnou strategii přímovazebního řízení pro kompenzaci nežádoucích vibrací kabiny v důsledku pružnosti závěsu. Zaměřte se na použití technik tvarování referenční veličiny a použití dopředné vazby.
3. Otestujte navržené techniky řízení na simulačním modelu a diskutujte potenciální zlepšení v dosažené kvalitě řízení oproti čistě zpětnovazebnímu řešení.
4. V případě možnosti ověřte navržené řešení na vhodném mechatronickém systému emulujícím dynamiku výtahu.

Rozsah bakalářské práce: **30-40 stránek A4**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná**

Seznam doporučené literatury:

K.J. Astrom, R. M. Murray, Feedback systems, Princeton 2009
G. C. Goodwin, S. F. Graebe, M. E. Salgado, Control System Design, Pearson 2000
Materiály ke kurzům KKY/LS1,LS2, SM

Vedoucí bakalářské práce: **Ing. Martin Goubej, Ph.D.**
Katedra kybernetiky

Datum zadání bakalářské práce: **17. října 2022**
Termín odevzdání bakalářské práce: **22. května 2023**

Západočeská univerzita v Plzni
Fakulta aplikovaných věd
L.S.

Doc. Ing. Miloš Železný, Ph.D.
děkan

Prof. Ing. Josef Psutka, CSc.
vedoucí katedry

V Plzni dne 17. října 2022

Prohlášení

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

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

V Plzni dne 14. května 2023:

_____

Poděkování

Rád bych poděkoval vedoucímu mé bakalářské práce Ing. Martinu Goubejovi, Ph.D. za provedení problematikou, užitečné rady, obohacení o nové vědomosti a čas, který mi věnoval. Rovněž bych chtěl poděkovat profesoru Ing. Miloši Schlegelovi, CSc., který mi během práce často zodpovídal dotazy.

Abstrakt

Cílem této práce je navrhnout přímovazební řízení s co nejlepším potlačením nežádoucích kmitů na straně kabiny výtahu a přitom využít možnosti dopředných filtrů přesně sledovat referenční trajektorii. Práce vychází z již zpracovaného projektu [1], který si kladl za cíl vytvořit vhodný matematický model systému a navrhnout robustní řízení včetně generátoru trajektorie pomocí klasického PI regulátoru.

Dále se v práci soustředím na vhodné navržení struktury přímovazebních filtrů problematické zejména z toho důvodu, že zpětnovazební smyčka je uzavřena přes jinou dynamiku, než která je skutečně řízena. Rovněž je řešen problém přepínání filtrů při nasazení na nelineární řídicí systém a jejich vysledování s účelem bezrázového přepínání. V návaznosti na to je použita linearizace podél trajektorie a časově proměnné přímovazební filtry. Oba tyto přístupy jsou porovnávány s robustním FIR filtrem.

Další částí je experimentální identifikace reálného systému a aplikování teoretických výsledků a postupů pro navržení řídicího systému za pomoci REXYGENu. Poslední nadstavbou nad zadáním této práce je vytvoření řídicího algoritmu výtahu ve smyslu logiky přejezdu mezi jednotlivými patry a rovněž naprogramování responzivní aplikace pro komunikaci s REXYGENem pomocí REST API, skrz kterou lze model ovládat a sledovat vizualizaci v reálném čase.

Klíčová slova

výtah, přímovazební řízení, robustní regulace kmitavých systémů, gain-scheduling, tvarování reference, FIR filtr, linearizace podél trajektorie, bezrázové přepínání filtrů, tříhmotový systém, vizualizace, REXYGEN

Abstract

The goal of this work is to design a feedforward control with the best possible suppression of unwanted vibrations on the side of the elevator cabin, while using the possibility of forward filters to accurately follow the reference trajectory. The work is based on the already completed project [1], which aimed to create a suitable mathematical plants of the system and design a robust control including a trajectory generator using a classic PI controller.

Furthermore, in my work, I focus on the appropriate design of the structure of feedforward filters, which is particularly problematic due to the fact that the feedback loop is closed through a different dynamic than the one that is actually controlled. There is also solved the problem of switching filters when it is deployed on a nonlinear control system and tracking them for smooth switching. Following this, linearization along the trajectory and time-varying feedforward filters are used. Both of these approaches are compared with a robust FIR filter.

The next part is the experimental identification of the real system and the application of theoretical results and procedures for the design of the control system with the help of REXYGEN. The last extension to the assignment of this work is the creation of an elevator control algorithm in the sense of the logic of crossing between individual floors, as well as the programming of a responsive application for communication with REXYGEN using the REST API, through which the model can be controlled and the visualization monitored in real time.

Keywords

elevator, feedforward control, robust regulation of oscillating systems, gain-scheduling, input-shaping, FIR filter, linearization around trajectory, shock-free switching of filters, three-mass system, visualization, REXYGEN

Obsah

Seznam obrázků

Seznam vztahů

Úvod

Výtahy jsou obecně již velmi dlouho v povědomí lidí, a tak se může zdát, že je to již všední a překonaná záležitost. Opak je ale pravdou – z hlediska vývoje řídících systémů představují takřka nekonečné možnosti pro zlepšení. Důkazem mohou být vysokorychlostní výtahy v mrakodrapech, kde rychlost jejich kabiny občas přesahuje i 70 km/h.

Možnosti a při takovýchto rychlostech i nezbytnosti tkví nejen v systému řízení, ale rovněž v konstrukci. Například se řeší aerodynamický tvar kabiny, přidávají se mechanické tlumiče, které pomáhají vyrušit kmitání kabiny na dlouhých závěsech, experimentuje se i s pohony založenými na magnetické levitaci, které známe z proslulých vysokorychlostních vlaků Maglev. Standardní výtahy v bytových budovách jezdí rychlostmi kolem $0.6 - 1\ m/s$ [5]. V této práci budu uvažovat rychlost nadstandardní, zhruba $2\ m/s$.

Já se však nebudu zabývat ničím z výše zmíněných vymožeností, ale naopak se pokusím klasický výtah s normální kabinou zavěšenou na obyčejném laně pomocí vhodného řízení přimět k rychlé avšak pohodlné a bezpečné jízdě. Největším zádrhelem jsou vibrace kabiny, které vznikají zejména při rozjezdu a při brzdění – jsou vyvolané *škubnutím* za lano. Můžeme si to představit asi tak, jako když máme kyblík s vodou zavěšený na provázku – Jestliže s provázkem trhneme, kyblík se rozskáče a voda se vylije. Musíme za provázek tahat pomalu a opatrně. Pokud chceme kyblík zdvihnout rychleji, musíme za provázek tahat velice chytře – a přesně toto *chytré tahání za provázek* je rozebírané v následující práci.

Cílem této práce je zamezit kmitání kabin vysokorychlostních výtahů na dlouhých závěsech, které zanáší do systému pružný mód. Tkví v nich i nelinearita systému – čím delší závěs je, tím více pruží. Zadáním je dostat se k metodě řízení, která by dovolila co největší rychlost a utlumila kmity způsobené změnou rychlosti. Zároveň je třeba myslet na praktickou stránku a lidi, kteří budou výtahem jezdit – proto je nutno vhodným způsobem tvarovat například i křivku zrychlení, aby byla jízda komfortní.

Přímovazební řízení nelze použít jen tak, a proto se první část této práce bude věnovat vytvoření matematického modelu výtahu, seznámení se s ním, nalezení vhodných fyzikálních parametrů (například pomocí převodů) tak, aby byl systém co nejlépe řiditelný. Další část bude věnována právě navržení vhodného robustního regulátoru a vyzkoušení zařazení různých filtrů pro zlepšení výsledků řízení. Toto vše je práce, kterou jsme se zabývali společně s kolegou Kubešem [1].

Po vytvoření prvního nástřelu řešení a seznámení se s modelem, jsme se každý věnovali své vlastní části práce a tou mou je přímovazební řízení, které je zevrubněji popsáno právě v této práci.

Nakonec jsem připravil kompletní řídicí systém, který zahrnuje i logiku přejezdů kabiny mezi jednotlivými patry a uživatelské ovládání včetně vizualizace. Pro reálnou aplikaci řízení byl použit software REXYGEN, pro vývoj regulátorů Matlab a Simulink a pro výpočty týkající se sestavení matematického modelu Maple. Simulace byla napsána pomocí knihovny JavaSimulation, vizualizace využívá Qt.

1 Základní komponenty

Úvod

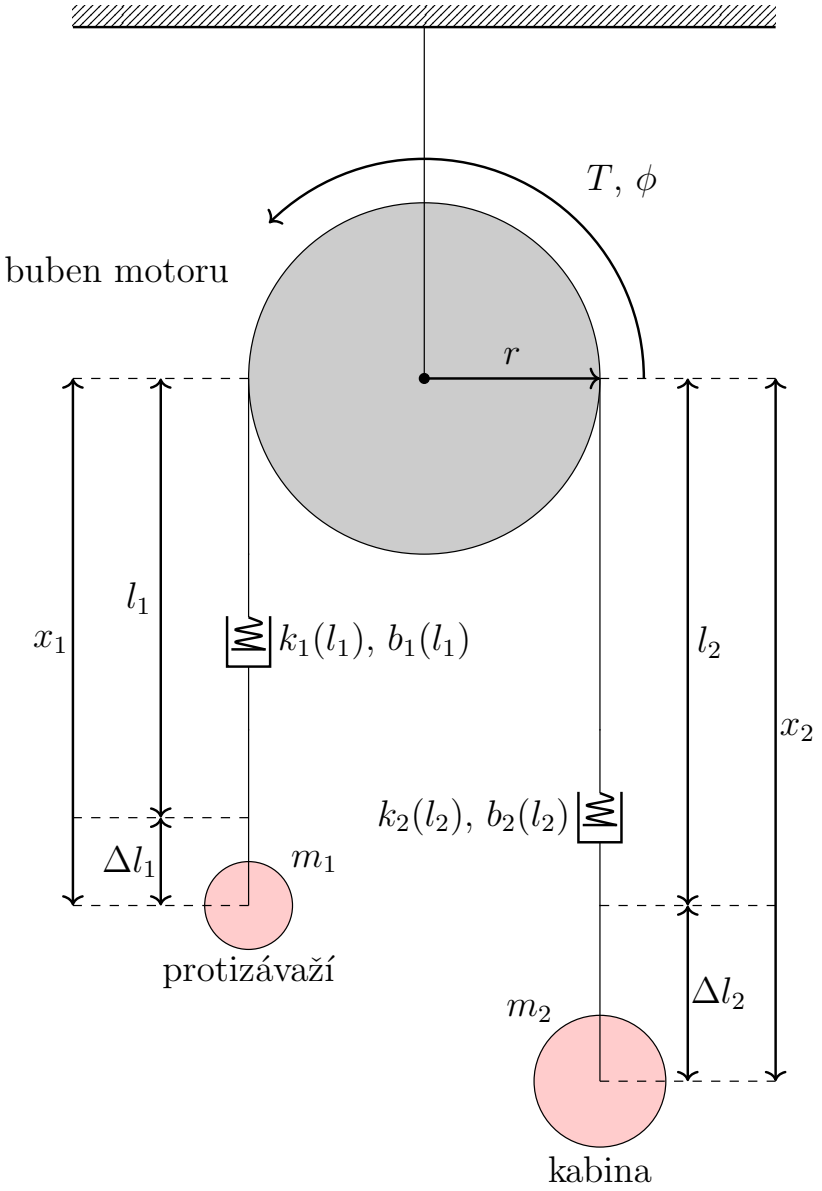
Tato část z velké části vychází již z předchozího zkoumání modelu, se kterým budu dále pracovat. Je to práce [1], kterou jsme již během studia vytvářeli s kolegou Kubešem. Díky tomu jsme získali zevrubnější vhled do problematiky a mohli si připravit podklady pro další práci.

Zde se tedy zabýváme vytvořením matematického modelu a jeho rozšířením o převodovku, postupně nalézáme úskalí práce s tříhmotovým systémem a učíme se principy, kterými se dají tyto problémy řešit. Nakonec se dobereme až k prvnímu výsledku a to robustnímu regulátoru, jehož chování se budu v této práci snažit vylepšit přímou vazbou. Před tím ale je třeba ještě použít filtr pro zatlumení nežádoucích frekvencí, na kterých jsou vybuzeny kmity. Ani nalezení PI regulátoru není tak snadné, protože výtah je model velmi proměnný v různých svých stavech.

Ze všech těchto problémů jsme však složili kompletní model s řídícím systémem a generátorem trajektorie, tudíž použitelný základ máme k dispozici. Následuje tedy shrnutí výsledků a postupů tohoto snažení.

1.1 Matematický model

Ze všeho nejdříve je třeba popsat model (viz obrázek 1) pomocí rovnic a následně provést simulace, abychom zjistili, zda jsou výpočty správné a systém se chová rozumně (tedy podle očekávání plynoucích z reálných zkušeností a znalostí fyzikálních zákonů). Těmito pokusy jsme získali i dobrý vhled do dynamiky systému.



Obrázek 1: Modelovaný systém

1.1.1 Pohybové rovnice

Při vytváření rovnic je třeba zavést určitá zjednodušení už jen proto, abychom dostali model, se kterým budeme moci nějak rozumně pracovat. Proto tedy lana považujeme za systém pružina-tlumič a neuvažujeme, že by se mohly "rozhoupávat"i v jiných směrech než nahoru a dolů. To by stejně u výtahů nemělo nastávat, jelikož jezdí v šachtě upevněných kolejnicích, ale samozřejmě tam malé výkmity budou a budou ovlivňovat například okamžitou velikost tření atp. Beztak by to nebyla příliš predikovatelná dynamika a působila by vlivem poruch, stejně tak je s ní tedy zacházeno a bude úkolem regulátoru ji potlačit.

Nyní můžeme začít postupně odvozovat vztahy z obrázku 1, kde máme délky závěsných lan $l_{1/2}$, které uvažujeme v neprodlouženém stavu, a výšky zavěšených hmot $x_{1/2}$ (resp. to, jak moc se odvinuly, tedy větší x znamená, že zátěž je níž), které reprezentují skutečnou polohu zátěží s již uvažovaným prodloužením dle Hookova zákona.

Vyjádříme si tedy délky lana na každé straně pomocí známých parametrů:

$$l_2 = l_0 – \phi \cdot r \;\; ; \;\; l_1 = l_0 - l_2 - \pi r = \phi r - \pi r = (\phi – \pi) r, \tag{1}$$

kde l_0 je klidová délka celého lana.

Dále musíme vyjádřit silové působení F_t na jednotlivé zátěže, kdy sílu lana působící na hmoty modelujeme jako systém pružina-tlumič ($F_{(s/d)i}$ – spring/damper force):

$$F_{t1} = m_1 g - F_{(s/d)1} - F_{f1} = m_1 g - k_1(x_1 - l_1) - b_1(\dot{x}_1 - \dot{l}_1) - F_{f1} \overset{!}{=} m_1 \ddot{x}_1 \tag{2}$$

$$F_{t2} = m_2 g - F_{(s/d)2} - F_{f2} = m_2 g - k_2(x_2 - l_2) - b_2(\dot{x}_2 - \dot{l}_2) - F_{f2} \overset{!}{=} m_2 \ddot{x}_2, \tag{3}$$

kde F_{fi} značí třecí sílu působící na hmoty v šachtě, ve které jsou usazené.

Rovněž potřebujeme vyjádřit celkové silové působení na buben motoru – tedy moment T_t, přičemž dojde k pomyslnému svázání obou hmot:

$$\begin{aligned} T_t &= T + F_{(s/d)1} \cdot r - F_{(s/d)2} \cdot r - T_f = \\ &= T + k_1(x_1 - l_1)r + b_1(\dot{x}_1 - \dot{l}_1)r - k_2(x_2 - l_2)r - b_2(\dot{x}_2 - \dot{l}_2)r - T_f \overset{!}{=} J \cdot \ddot{\phi} \end{aligned} \tag{4}$$

Zde vycházíme z toho, že točivý moment $T \triangleq F \cdot r$. T_f reprezentuje tření uvnitř motoru na bubnu: $T_f = b_f \cdot \omega = b_f \cdot \dot{\phi}$, T je pak vstupní točivý moment udělovaný motorem.

Dále budeme uvažovat, že hodnoty k (pružnost) a b (tlumení) v systému pružina-tlumič (lano) jsou proměnné a mění se spolu s délkou odvinutého lana, přičemž k_0 a b_0 jsou jednotková tuhost pružiny a jednotkový součinitel útlumu pro daný materiál

$$k_1(l_1) = \frac{k_0}{l_1} \qquad b_1(l_1) = \frac{b_0}{l_1}$$
$$k_2(l_2) = \frac{k_0}{l_2} \qquad b_2(l_2) = \frac{b_0}{l_2} \tag{5}$$

To má za důsledek fakt, že módy systému, frekvence vybuzení kmitů atd. budou proměnné rovněž spolu s délkou odvinutého lana a budou se tedy v čase měnit. To bude dále způsobovat problémy při návrhu regulátorů.

Nyní můžeme osamostatnit nejvyšší derivace (zrychlení) a získat model dynamiky kmitavého systému v klasickém tvaru, kde pružnost je svázána s polohou a tlumič s rychlostí. První dvě rovnice vydělíme příslušnou hmotností (pro vyjádření nejvyšší derivace) a dosadíme za l_1 a l_2 ze vztahu 1 (pozor na derivace konstant – např. πr – jsou nulové, proto v následujících rovnicích již nevystupují) a také za k_1, b_1, k_2, b_2 vztahy odvozené výše.

Dostaneme tedy následující rovnice:

$$\ddot{x}_1 = g - \frac{k_0}{m_1(\phi r - \pi r)}(x_1 - \phi r + \pi r) - \frac{b_0}{m_1(\phi r - \pi r)}(\dot{x}_1 - \dot{\phi} r) - \frac{F_{f1}}{m_1} \tag{6}$$

$$\ddot{x}_2 = g - \frac{k_0}{m_2(l_0 - \phi r)}(x_2 - l_0 + \phi r) - \frac{b_0}{m_2(l_0 - \phi r)}(\dot{x}_2 + \dot{\phi} r) - \frac{F_{f2}}{m_2} \tag{7}$$

$$\ddot{\phi} = \frac{T}{J} + \frac{k_0 r}{J(\phi r - \pi r)}(x_1 - \phi r + \pi r) + \frac{b_0 r}{J(\phi r - \pi r)}(\dot{x}_1 - \dot{\phi} r) + \ldots$$
$$\ldots - \frac{k_0 r}{J(l_0 - \phi r)}(x_2 - l_0 + \phi r) - \frac{b_0 r}{J(l_0 - \phi r)}(\dot{x}_2 + \dot{\phi} r) - b_f \frac{\dot{\phi}}{J} \tag{8}$$

1.1.2 Linearizace

Model budeme linearizovat samozřejmě v rovnovážném bodě. Jestliže je bod rovnovážný, je derivace stavu nulová, což pro nás znamená, že všechny nejvyšší derivace $(\ddot{x}_1, \ddot{x}_2, \ddot{\phi})$ nabývají nulové hodnoty. Dosadíme za ně tedy nulu a vyjádříme si jednotlivé proměnné ($x_1,\ x_2,\ \phi$), čímž dostaneme požadované počáteční podmínky a rovněž moment síly, který potřebujeme k udržení soustavy v tomto rovnovážném stavu. Z toho plyne, že stav není podle přesné terminologie rovnovážný nýbrž ustálený, protože pro udržení se v něm potřebujeme daný konstantní vstup.

Jelikož chceme získat obecný lineární model pro obecnou délku lana l, vyjádříme nakonec výrazy pomocí proměnných l_1 a l_2 (viz vztah 1). Dále se zde mluví o *rovnicích* – tím jsou myšleny výchozí pohybové rovnice systému (viz vztah 8).

První rovnice (vztah 6)
Derivace vynulujeme a vyjádříme x_1, čímž dostaneme první počáteční podmínku:

$$x_{10} = \frac{l_1(gm_1 - F_{f1} + k_0)}{k_0} \tag{9}$$

Druhá rovnice (vztah 7)
Derivace vynulujeme a vyjádříme x_2, čímž dostaneme druhou počáteční podmínku:

$$x_{20} = \frac{l_2(gm_2 - F_{F2} + k_0)}{k_0} \tag{10}$$

Třetí rovnice (vztah 8)
Derivace vynulujeme a dosadíme již vyjádřené x_2 a x_1. Získáme tak moment potřebný pro udržení rovnovážného stavu, který celkem logicky vyvžuje dvě hzavěšené hmoty (kompenzuje jejich váhový rozdíl) a v našem případě uvažuje i tření na ně působící.

$$T = gr(m_2 - m_1) + r(F_{F1} - F_{F2}) \tag{11}$$

Počáteční hodnota ϕ je natočení bubnu ve zvolené délce lana l_2. Budeme předpokládat, že volíme právě l_2, protože to je délka lana na straně kabiny. Pak pro l_1 bude platit následující vztah:

$$l_1 = l_0 - l_2 - \pi r \tag{12}$$

A z toho již dopočteme ϕ_0 v radiánech jako:

$$\phi_0 = \frac{l_1 + \pi r}{r} \tag{13}$$

V okamžiku, kdy máme rovnovážný stav, můžeme začít linearizovat v okolí tohoto bodu (víme, že v rovnicích můžeme za T dosadit výše vypočtený vztah 11).

Nyní tedy zbývá jen klasicky derivovat podle příslušných proměnných, abychom naplnili matice stavového modelu. Linearizace spočívá v tom, že provedeme rozvoj do Taylorovy řady a použijeme pouze první člen, zbytek zanedbáme. To znamená, že ve statické charakteristice systému se promítne tento nový systém jako tečna k původní křivce v bodě, kde byla linearizace provedena Ještě si zavedeme následující stavové proměnné:

$$
\begin{aligned}
[y_1, y_2, y_3, y_4, y_5, y_6] &= [x_1, x_2, \phi, \dot{x}_1, \dot{x}_2, \dot{\phi}] \\
\dot{y}_1 &= y_4 \\
\dot{y}_2 &= y_5 \\
\dot{y}_3 &= y_6 \\
\dot{y}_4 &= g - \frac{k_0}{m_1(y_3 r - \pi r)}(y_1 - y_3 r + \pi r) - \frac{b_0}{m_1(y_3 r - \pi r)}(y_4 - y_6 r) - \frac{F_{f1}}{m_1} \\
\dot{y}_5 &= g - \frac{k_0}{m_2(l_0 - y_3 r)}(y_2 - l_0 + y_3 r) - \frac{b_0}{m_2(l_0 - y_3 r)}(y_5 + y_6 r) - \frac{F_{f2}}{m_2} \\
\dot{y}_6 &= \frac{T}{J} + \frac{k_0 r}{J(y_3 r - \pi r)}(y_1 - y_3 r + \pi r) + \frac{b_0 r}{J(y_3 r - \pi r)}(y_4 - y_6 r) + \dots \\
&\dots - \frac{k_0 r}{J(l_0 - y_3 r)}(y_2 - l_0 + y_3 r) - \frac{b_0 r}{J(l_0 - y_3 r)}(y_5 + y_6 r) - b_f \frac{y_6}{J}
\end{aligned}
\tag{14}
$$

Parciální derivace prvních tří rovnic jsou poměrně zřejmé (získám samé nuly, jen na pozici daného stavu bude jednička viz vztah 19). Výpočty parciálních derivací zbylých rovnic vyžadují již složitější výpočty, ke kterým jsme využili software Maple:

Parciální derivace $\dot{y}_4$; uvažujeme $y_3 r - \pi r = l_1$

$$
\begin{aligned}
\frac{\partial \dot{y}_4}{\partial y_1} &= \frac{-k_0}{m_1 l_1} \\
\frac{\partial \dot{y}_4}{\partial y_2} &= 0 \\
\frac{\partial \dot{y}_4}{\partial y_3} &= \frac{k_0 r y_1}{m_1 l_1^2} \\
\frac{\partial \dot{y}_4}{\partial y_4} &= \frac{-b_0}{m_1 l_1} \\
\frac{\partial \dot{y}_4}{\partial y_5} &= 0 \\
\frac{\partial \dot{y}_4}{\partial y_6} &= \frac{b_0 r}{m_1 l_1}
\end{aligned}
\tag{15}
$$

Parciální derivace $\dot{y}_5$; uvažujeme $l_0 - y_3 r = l_2$

$$\begin{aligned}
\frac{\partial \dot{y}_5}{\partial y_1} &= 0 \\
\frac{\partial \dot{y}_5}{\partial y_2} &= \frac{-k_0}{m_2 l_2} \\
\frac{\partial \dot{y}_5}{\partial y_3} &= \frac{-k_0 r z_3}{m_2 l_2^2} \\
\frac{\partial \dot{y}_5}{\partial y_4} &= 0 \\
\frac{\partial \dot{y}_5}{\partial y_5} &= \frac{-b_0}{m_2 l_2} \\
\frac{\partial \dot{y}_5}{\partial y_6} &= \frac{-b_0 r}{m_2 l_2}
\end{aligned} \tag{16}$$

Parciální derivace $\dot{y}_6$

$$\begin{aligned}
\frac{\partial \dot{y}_6}{\partial y_1} &= \frac{k_0 r}{J l_1} \\
\frac{\partial \dot{y}_6}{\partial y_2} &= \frac{-k_0 r}{J l_2} \\
\frac{\partial \dot{y}_6}{\partial y_3} &= \frac{-k_0 r^2 (y_2 l_1^2 + l_1 l_2^2 - (l_1 - y_1) l_2^2)}{J l_1^2 l_2^2} \\
\frac{\partial \dot{y}_6}{\partial y_4} &= \frac{b_0 r}{J l_1} \\
\frac{\partial \dot{y}_6}{\partial y_5} &= \frac{-b_0 r}{J l_2} \\
\frac{\partial \dot{y}_6}{\partial y_6} &= \frac{-r^2 (l_1 + l_2 - b_f l_1 l_2) b_0}{J l_1 l_2}
\end{aligned} \tag{17}$$

Ve výrazech se stále vyskytují stavové proměnné y_i. To je v pořádku, jelikož linearizujeme pro obecnou délku závěsu l_2. V moment, kdy si zvolíme konkrétní délku l_2 (resp. budeme volit spíše reálnou výšku po prodloužení x_2 a z ní l_2 vypočteme), ve které budeme linearizovat, tak tyto hodnoty do výrazů dosadíme a získáme výrazy již bez stavových proměnných.

Matici B budou tvořit parciální derivace vstupu (tedy momentu), který se vykytuje jen v poslední rovnici:

$$\frac{\partial \dot{y}_6}{\partial T} = \frac{1}{J} \tag{18}$$

Matici C naplníme jedničkami na pozicích stavů, které chceme pozorovat, všude jinde budou nuly. Například tak, jak je to v následujícím případě, kde nás zajímají polohy zátěže a kabiny.

$$\boldsymbol{A} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ \frac{-k_0}{m_1 l_1} & 0 & \frac{k_0 r z_1}{m_1 l_1^2} & \frac{-b_0}{m_1 l_1} & 0 & \frac{b_0 r}{m_1 l_1} \\ 0 & \frac{-k_0}{m_2 l_2} & \frac{-k_0 r z_2}{m_2 l_2^2} & 0 & \frac{-b_0}{m_2 l_2} & \frac{-b_0 r}{m_2 l_2} \\ \frac{k_0 r}{J l_1} & \frac{-k_0 r}{J l_2} & \frac{-k_0 r^2 (y_2 l_1^2 + l_1 l_2^2 - (l_1 - z_1) l_2^2)}{J l_1^2 l_2^2} & \frac{b_0 r}{J l_1} & \frac{-b_0 r}{J l_2} & \frac{-r^2 (l_1 + l_2 - b_f l_1 l_2) b_0}{J l_1 l_2} \end{bmatrix}$$

$$\boldsymbol{B} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \frac{1}{J} \end{bmatrix}$$

$$\boldsymbol{C} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{D} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\begin{aligned} \dot{\boldsymbol{z}}(t) &= \boldsymbol{A}\boldsymbol{z}(t) + \boldsymbol{B}\boldsymbol{u}(t) \\ \boldsymbol{y}(t) &= \boldsymbol{C}\boldsymbol{z}(t) + \boldsymbol{D}\boldsymbol{u}(t) \end{aligned} \tag{19}$$

Správnost linearizace jsme ověřili pomocí experimentu s odchylkami od daného rovnovážného bodu. To znamená, že jsme nejprve pustili do systému čistě moment (viz vztah 11), který má udržovat rovnováhu, čímž byla ověřena správnost vypočtení počátečních podmínek (samozřejmě pokud systém nevykazoval žádnou změnu stavů). Pak jsme tento moment vychylovali a zjišťovali, jak moc se lineární model od nelineárního odchýlí v různě velkém okolí. Ukázalo se, že linearizace po jednom patře by měla být pro další práci dostačující.

1.1.3 Linearizace v různých výškách

Jelikož máme vyjádřený obecný lineární model, není problém dosadit hodnotu x_{20} – tedy skutečné počáteční polohy kabiny – v místech, kde předpokládáme jednotlivá patra. Vztahy pro počáteční podmínky pak vypadají následovně:

$$x_{20} = \text{zvolíme}$$

$$l_{20} = \frac{x_{20} k_0}{-g \cdot m_2 + F_{f1} - k_0} \tag{20}$$

$$l_{10} = l_0 - \pi r - l_{20} \tag{21}$$

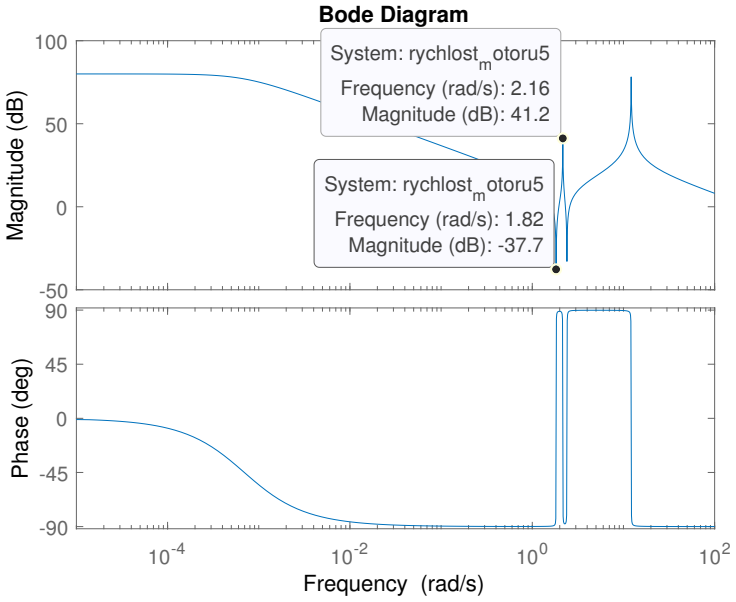
$$x_{10} = \frac{l_{10}(g \cdot m_1 - F_{f1} + k_0)}{k_0} \tag{22}$$

$$\phi_0 = \frac{l_{10} + \pi r}{r} \tag{23}$$

Tyto hodnoty dosadíme do obecného linearizovaného modelu (viz vztah 19) a dostaneme stavový model linearizovaný v konkrétním patře. Jelikož uvažujeme do začátku 10 patrovou budovu, získáme tímto způsobem 10 stavových modelů. Z těch není těžké vyjádřit přenosy, které potřebujeme, a to přenos na rychlost motoru a na rychlost kabiny. Rychlost motoru potřebujeme z toho důvodu, že se u systémů tohoto typu ukazuje, že nejvhodnější je řídit polohu nepřímo právě pomocí rychlosti motoru (to je dáno tím, že když si vykreslím Bodeho charakteristiky přenosů na rychlost motoru a rychlost zátěže, uvidím, že průběh fáze v přenosu na rychlost motoru zůstává stále mezi 0^o až -90^o. Oproti tomu průběh fáze u přenosu na rychlost kabiny padá v okolí rezonanční frekvence až k -270^o. Z toho vyplývá velké omezení na zesílení, které si mohu dovolit u regulátoru, a výsledná šířka pásma je mnohem menší, než při řízení se zpětnou vazbou na straně motoru. Proto je standardem u takovýchto systémů volit pro rychlostní smyčku motorovou zpětnou vazbu).

Po vykreslení Bodeho charakteristiky přenosu na rychlost motoru (například v pátém patře) však dojdeme k nepříjemnému zjištění. Tedy k tomu zjištění jsme došli dříve, když jsme se snažili navrhnout regulátor a až později se dozvěděli, čím je to dané. Ukazuje se totiž, že aby byl systém dobře řiditelný, chtělo by to, aby poměr frekvencí první antirezonance a rezonance byl roven cca 2 [6], což (viz obrázek 2) skutečně neodpovídá.

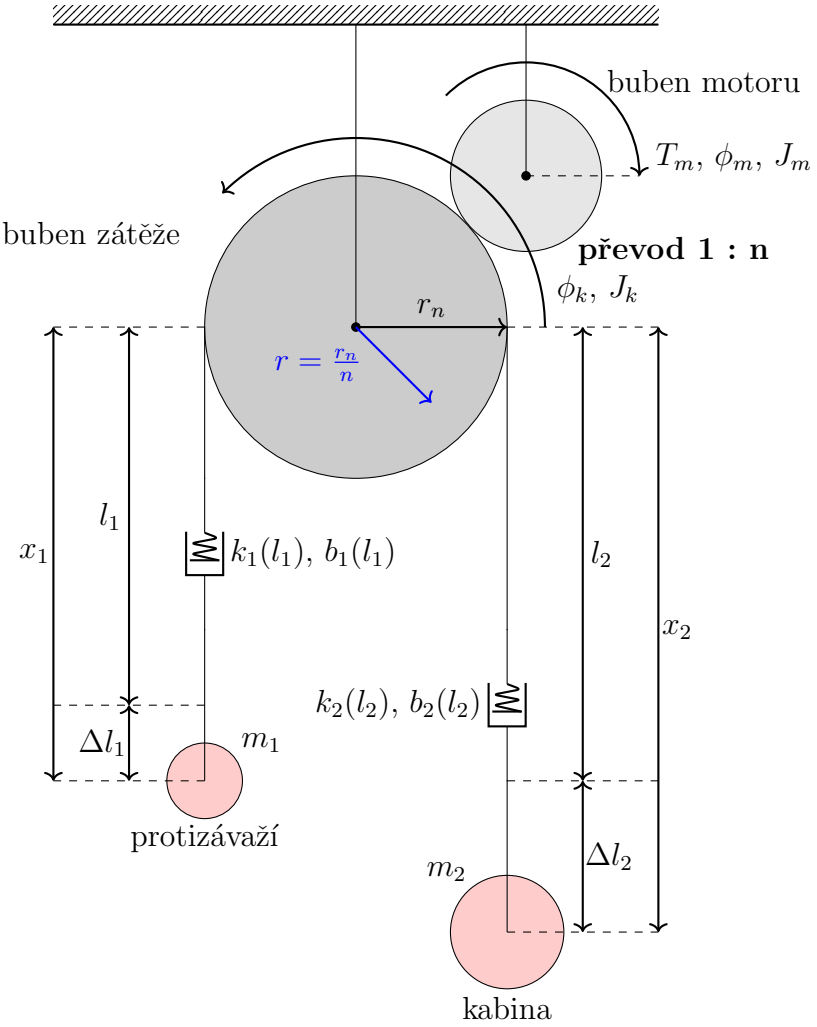
Z toho důvodu jsme se rozhodli do modelu zařadit ještě převodovku – stejně by tam byla v praxi potřeba, protože výtah se pohybuje relativně pomalu, ovšem pro elektrický motor jsou ideální mnohem vyšší pracovní otáčky, ve smyslu efektivity i opotřebení. Zavedením převodu rovněž ovlivníme fyzikální parametry (například virtuálně změníme poloměr špulky motoru a moment setrvačnosti), což by mohlo dopomoci k lepší řiditelnosti.



Obrázek 2: Špatný poměr rezonance a antirezonance

1.2 Převodovka

Z důvodů zmíněných v závěru předchozí části dochází k rozšíření matematického modelu o převodovku, která je v praxi rovněž součástí pohonu, aby bylo možné dosáhnout klidného otáčení špulky. Při tomto rozšíření (viz vztah 3) dojde především k virtuálnímu přeškálování poloměru a momentů sil.



Obrázek 3: Zadaný systém

Pokud budeme chtít odvodit pohybové rovnice tohoto systému, nebude to již nic těžkého, jen dojde k virtuální změně momentu setrvačnosti a poloměru, a to takovýmto způsobem:

$$J = J_m + \frac{J_k}{n^2}$$
$$r = \frac{r_n}{n} \tag{24}$$

přičemž J_m je moment setrvačnosti motoru a J_k je moment setrvačnosti kladky.

Kladku modelujeme zjednodušeně pomocí válce, tudíž dostáváme

$$J_k = \frac{1}{2} m_k r_n^2 \tag{25}$$

A moment setrvačnosti motoru definujeme na základě přibližně reálných parametrů [7] následovně:

$$J_m = \frac{1}{2} \cdot 0.5 \cdot 0.02^2 \tag{26}$$

Pokud tyto vztahy dosadíme do původních rovnic (viz vztah 8), získáme rovnice nové včetně převodu, které nám dovolí modelovat systém s libovolným převodem na straně motoru, což je v praxi velmi důležité.

Zvolené parametry by měly relativně dobře odpovídat reálu (viz vztah 27). Znovu vykreslená Bodeho charakteristika již vypadá vhodněji, minimálně zde již nejsou tak ostré píky rezonance a antirezonance (viz obrázek 4, kde jsou ukázané průběhy v různých polohách kabiny – v 1., 5. a 10. patře).

převodový poměr: $n = 15$

tíhové zrychlení: $g = 9.81$

hmotnost zátěže: $m_1 = 1000$

hmotnost kabiny: $m_2 = 600–1400$

hmotnost kladky: $m_k = 50$

celková délka lana: $l_0 = 33$

konstanta pružnosti: $k_0 = 5000000$

konstanta tlumení: $b_0 = 60000$

skutečný poloměr špulky motoru: $r_n = 0.3$

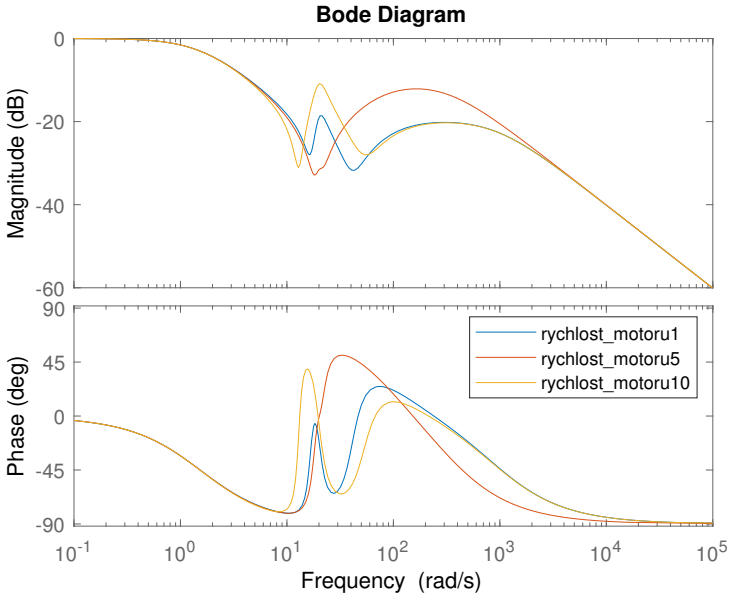
moment setrvačnosti motoru: $J_m = 1/2 \cdot 0.5 \cdot 0.02^2$

moment setrvačnosti kladky: $J_k = 1/2 \cdot m_k \cdot r_n^2$

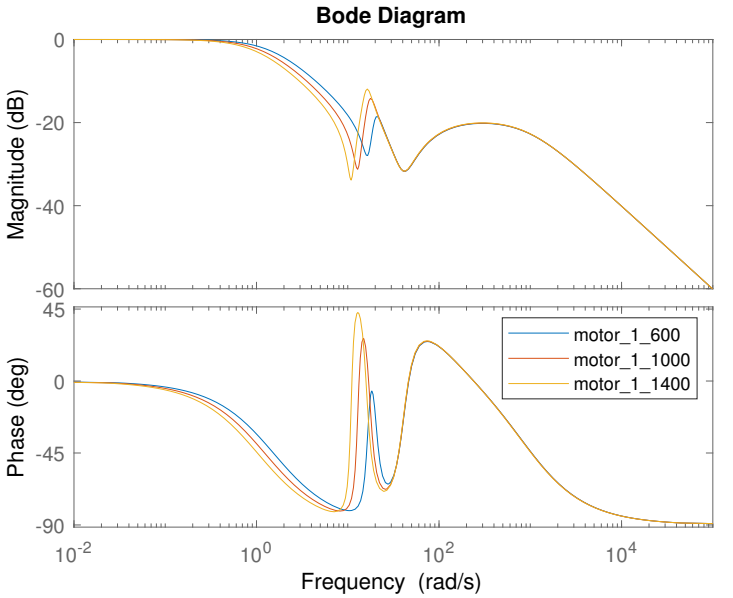
virtuální přeškálovaný poloměr: $r = r_n/n$

virtuální moment setrvačnosti: $J = J_m + J_k/n^2$ (27)

Ani přes to všechno se však stále nedařilo navrhnout robustní regulátor. Velký problém dělala především proměnlivá hmotnost kabiny (viz obrázek 5), která přináší posun frekvencí první rezonance. Když to zkombinujeme s různými patry, které mění zase velikost rezonance (amplitudu), tak dostáváme příliš širokou škálu možností, kde a jaká rezonance bude, což nám zabraňuje v navržení robustního regulátoru. Proto je další část věnována návrhu notch filtru.



Obrázek 4: Bodeho charakteristiky s převodovkou a vhodně naladěnými fyzikálními parametry pro kabinu v 1., 5. a 10. patře



Obrázek 5: Bodeho charakteristiky s převodovkou s kabinou v 1. patře, ale o různé hmotnosti

1.3 Robustní regulace

1.3.1 Notch filtr

Základní myšlenka vedoucí k návrhu notch filtru byla podnícena potřebou zbavit se první rezonance, o které jsme mluvili v předchozí části a vysvětlovali její negativní vlivy na řiditelnost systému. Notch filtr se má stát vlastně součástí regulátoru, což povede k jisté nevýhodě – jestliže určité frekvence takovýmto způsobem zatlumíme, bude horší reakce regulátoru na poruchy na těchto frekvencích.

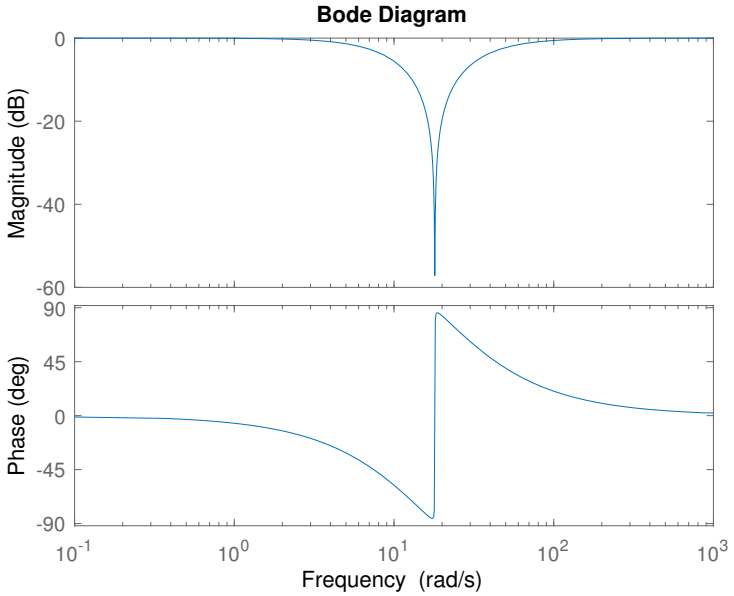
Ale zpět k filtru samotnému – je to vlastně pásmová zádrž. Notch filtr je možné reprezentovat přenosovou funkcí ve tvaru

$$Notch(s) = \frac{s^2 + 2 \cdot \xi_1 \omega_n + \omega_n^2}{s^2 + 2 \cdot \xi_2 \omega_n + \omega_n^2} \tag{28}$$

kde parametr ω_n určuje frekvenci, na které se bude nacházet pík filtru, a parametr ξ_2 říká, jak široký pík bude. Parametr ξ_1 pak určuje hloubku píku. Tento filtr (obrázek 6) bylo třeba navrhnout tak, aby dokázal odfiltrovat první rezonanci u všech přenosů – tudíž s dostatečnou (ale ne přílišnou) šířkou, která určité frekvence odfiltruje.

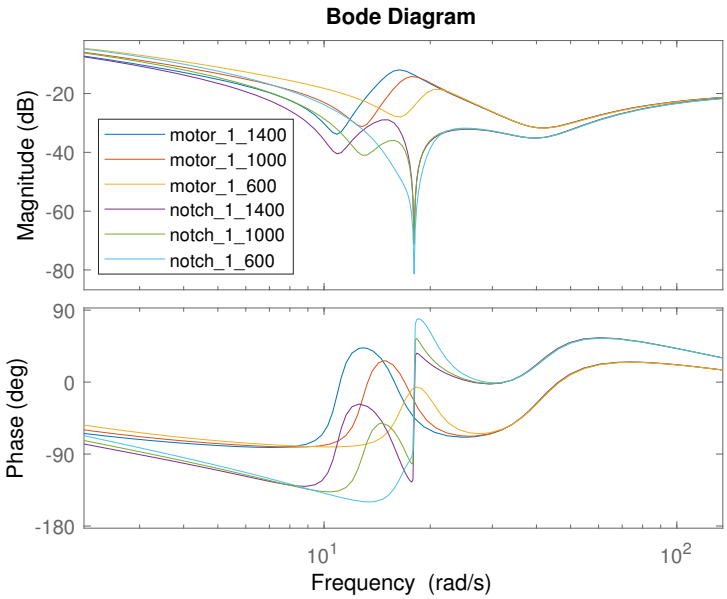
Parametry takovýmto způsobem naladěného notch filtru vypadají následovně (obrázek 29):

$$\begin{aligned} \omega_n &= 18 \\ \xi_1 &= 1 \\ \xi_2 &= \frac{1}{720} \end{aligned} \tag{29}$$



Obrázek 6: Bodeho charakteristika navrženého notch filtru

Na (obrázku 7) lze vidět výsledek , když se filtr pokusím aplikovat na přenosy jako takové – tedy když se pokusím zatlumit systém na frekvencích přibližně odpovídajících oblasti první rezonance:

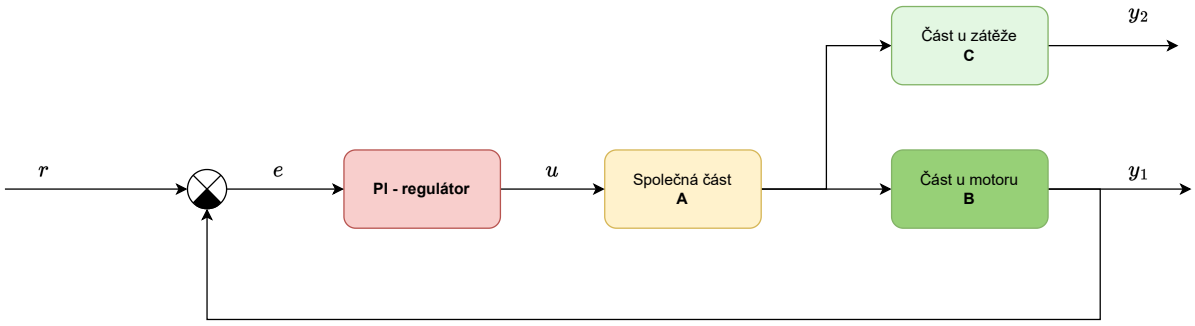


Obrázek 7: Porovnání Bodeho charakteristik přenosů s notch filtrem a bez

Skutečně jsme se rezonancí poměrně vhodně zbavili. Dále bude potřeba zahrnout tento filtr do regulátoru a navrhnout vhodně zbytek (zřejmě PI regulátor) tak, aby fungoval pro všechny přenosy – tedy pro kabinu v různých patrech a ještě různě naloženou.

1.3.2 PI regulátor

Pro vytvoření regulátoru nám byla doporučena rozšiřující aplikace pro Matlab vytvořená na KKY FAV ZČU jejímž autorem je Michal Špirk [8], jelikož je navržená přesně pro tento typ úloh – klasická dvouhmotová problematika, kdy chceme řídit systém zpětnou vazbou určité části smyčky, abychom dosáhli požadovaného chování v jiné části smyčky.[2]



Obrázek 8: Schéma systému vhodného pro H_∞ [2, p. 7]

Jak jde vidět na ilustrativním obrázku 8, systém má i společnou část **A** – tou je pro nás notch filtr (viz sekce 1.3.1) a dalo by se říct, že i regulátor. Samostatnými prvky **B** jsou přenos z momentu motoru na rychlost otáčení bubnu zátěže $T_m \rightarrow \dot{\phi}_k$, kde nesmíme zapomenout na zmiňovanou převodovku (viz obrázek 3), a přenos z momentu motoru na rychlost kabiny **C** $T_m \rightarrow \dot{x}_2$.

Jelikož aplikace umožňuje na rozdíl od jiných nástrojů pracovat s více přenosy zároveň, můžeme jí předložit rovnou všech 18 přenosů – 3 různá patra (1., 5. a 10.) v každém 3 různá zatížení (600, 1000 a 1400 kg) a to samé pro přenos nejen na motor ale i na kabinu. Samozřejmě se zde objeví právě pro navržení regulátoru i velmi důležitý notch filtr. Aplikace by nám tedy měla pomoci nalézt takový regulátor, který bude stabilizovat všechny tyto možnosti (přenosy) a nemusíme to složitě ručně zkoušet pro každý zvlášť.

Nedostaneme však jen jediný regulátor, ale celou množinu, která závisí na tom, jaký zvolíme požadovaný počet regulátorů a region, na kterém chceme hledat (regionem rozumíme prostor definovaný integračním a proporcionálním zesílením). Tyto nalezené regulátory jsou následně i graficky uspořádány podle určitého kritéria, kde použijeme ITAE pro kabinu, což je to, co nás nakonec zajímá nejvíce (chceme za co nejkratší čas dosáhnout ideálně minimálních kmitů na straně kabiny). A zároveň jsme toto kritérium vybrali pro přenos ve spodním patře s nejmenším zatížením, protože to je okamžik, kdy je lano ke kabině nejdelší a kabina je nejlehčí, tudíž kmitá nejvíce.

Ještě by bylo vhodné zmínit, co je to vlastně ta metoda H_∞. Zřejmě bude vhodné odkázat se na práci profesora Schlegela [3], ve které je tato metoda dopodrobna rozebrána a ukázána včetně odvození. Popsána je rovněž v již zmiňované bakalářské práci Michala Špirka [8]. Já zde opět naznačím jen základní myšlenku podloženou těmito pracemi. Nejprve se odkáži na obrázek 9, aby bylo zřejmé, o čem se v jednotlivých vztazích mluví.

Metoda H_∞ je spíše druhem matematické optimalizace než přímo stylem řízení, ačkoliv se o ní tak občas mluví. Jedna se zejména o optimalizaci ve zvoleném kritériu (v našem případě právě ITAE), kde se snažíme nalézt maximální / minimální hodnotu podle typu kritéria. Konkrétní tvar je uvedený ve vztahu 32.

$$\text{snaha minimalizovat} \|H_{W\to Z}(P,C)\|_\infty \tag{30}$$

$$\|H\|_\infty \triangleq \sup_{w\neq 0} \frac{\|H_w\|_2}{\|w\|_2} = \sup_{w\neq 0} \frac{\|w\|_2}{\|z\|_2} = \max \overline{\sigma}(H(j\omega)) \tag{31}$$

$$\|z\|_2 = \left(\frac{1}{2\pi}\int_{-\infty}^{\infty} TrH(j\omega)H^T(j\omega)d\omega\right)^{\frac{1}{2}} \tag{32}$$

Běžně se však uvádí na pohled jednodušší obecný postup v maticovém tvaru (viz vztah 32), který je poměrně snadno pochopitelný a aplikovatelný.

$$\begin{bmatrix} z \\ v \end{bmatrix} \begin{bmatrix} P_{11}(s) & P_{12}(s) \\ P_{21}(s) & P_{22}(s) \end{bmatrix} \begin{bmatrix} w \\ u \end{bmatrix} \tag{33}$$

$$u = C(s)\cdot v$$

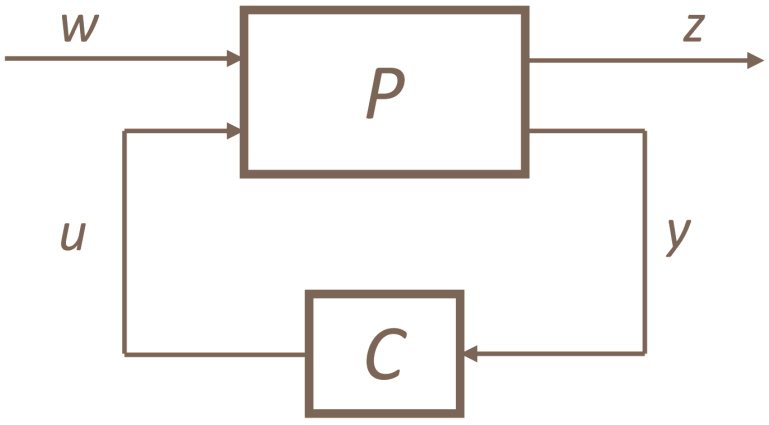
Z toho vyplývá, že z bude záviset na w následovně:

$$z = H_w(P,C)\cdot w, \tag{34}$$

kde H_w je definované jako

$$H_w = P_{11} + P_{12}C(I + P_{22}C)^{-1}P_{21} \tag{35}$$

Pak je nalezeno optimální zesílení regulátoru C tak, aby byla minimalizována funkce $H_w(P,C)$.



Obrázek 9: Schéma systému pro odvození metody H_∞ [3]

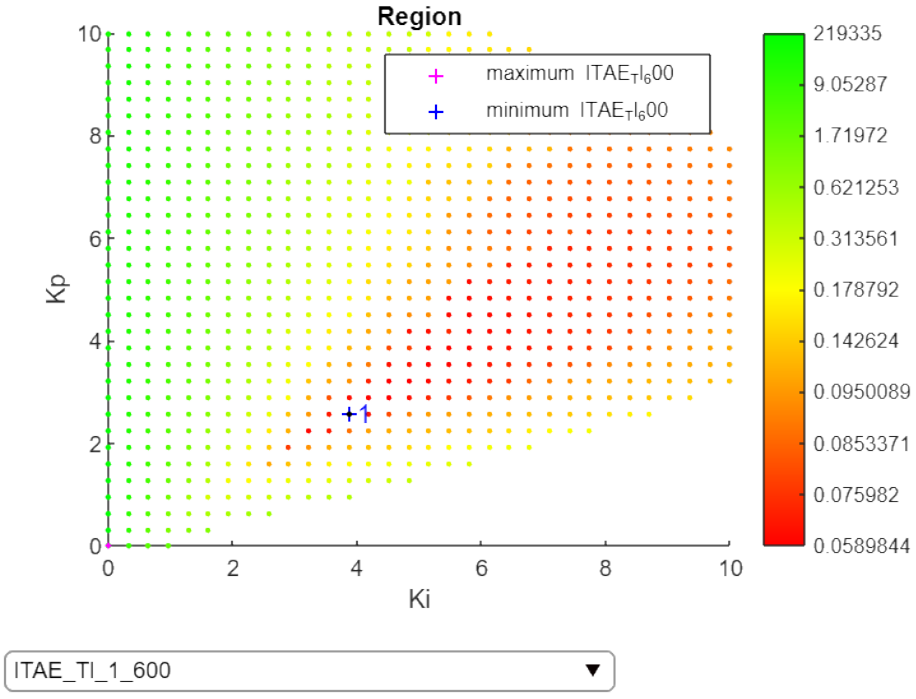
V tuto chvíli jsme ale narazili na problém v aplikaci, která nám určitou část regionu nevykreslila. Rozhodl jsem se tedy využít informace o tom, jakým směrem se parametry regulátoru vyvíjejí. Sice už nás tam software nepustil, ale snadno jsem mohl odhadnout, že vhodný regulátor bude někde v relativně malých zesíleních K_P a K_I, protože tímto směrem se kvalita regulace postupně zlepšovala

S touto informací se dal již použít i softwareový nástroj Matlabu Sisotool, který funguje na principu geometrického místa kořenů a umožňuje sledovat například bodeho charakteristiky nebo odezvu na poruchy a tak dále. Zde se mi ručně povedlo navrhnout relativně slušný PI regulátor:

$$F_R = \frac{1.803s + 4.024}{s} \tag{36}$$

Tento regulátor dobře fungoval pro všechny přenosy, ale říkal jsem si, že zde určitě bude ještě prostor pro zlepšení. Je to dané už jen tím, že pomocí nástroje Sisotool je možné sice regulátor navrhnout relativně snadno, ale o něco hůře se zde již pracuje s lehkými nuancemi regulace. Získal jsem ale přesnější informaci o ideální poloze integračního a proporcionálního zesílení PI regulátoru.

S touto informací jsme se můžeme vrátit zpět do aplikace H_∞ a pokusit se opětovně omezit region. Proč jej nebylo předtím možné zobrazit, na to jsme nepřišli, ale každopádně pokud jsme ho omezili jen na malá zesílení (od 0 do 10 viz obrázek 10) a vybrali dostatečně hustý rastr (vysoký počet regulátorů), tak se povedlo nalézt ideální nastavení K_P a K_I vzhledem k optimalizace ITAE kritéria, což bychom ručně hledali velmi dlouho, ale počítač to dokáže určit relativně snadno.



Obrázek 10: Region všech stabilizujících reg. v aplikaci H_∞, zbarvení podle ITAE kritéria

Zbývalo tedy jen vybrat regulátor s nejmenším ITAE (viz vztah 37):

$$F_R = \frac{2.581s + 3.871}{s} \tag{37}$$

Tedy dostaneme PI regulátor s $K_i = 3.871$ a $K_p = 2.581$

a získali jsme výsledky znázorněné na následujících obrázcích – vyhovující robustní regulátor, který dokáže uřídit model výtahu ve všech patrech a všech možných přípustných zatíženích s vlastnostmi, které jsou pro nás důležité (relativně nízký překmit, vyvážená doba na odregulování chyby, čehož jsme se kvůli použití notch filtru trochu obávali, a nízká kmitavost).

V následující kapitole budeme vyvíjet generátor trajektorie, který bude pro použití regulátoru velmi důležitý. Jde především o to určitým způsobem usměrnit průběh polohy, rychlosti a dalších derivací, ale má to ještě jeden důsledek, který je v tomto případě velmi důležitý – regulátor bude k regulaci dostávat vlastně minimální skoky, tudíž překmit, který lze vidět na obrázku 11, nebude nijak důležitý. Teď sice tvoří zhruba 15%, ale pokud nebude skok $1m$, ale třeba $1cm$, bude to již zanedbatelná hodnota. Ke všemu se tyto skokové hodntoy budou měnit tak rychle, že se překmity ani nestačí projevit.

Na následujících obrázcích, které jsou popsány níže, jsou znázorněny všechny důležité výstupy, které by nás mohly zajímat, abychom si udělali dobrou představu o chování systému.

Na obrázku 11 můžeme vidět přechodové charakteristiky všech devíti systémů (na straně kabiny) s tímto regulátorem (vztah 37).

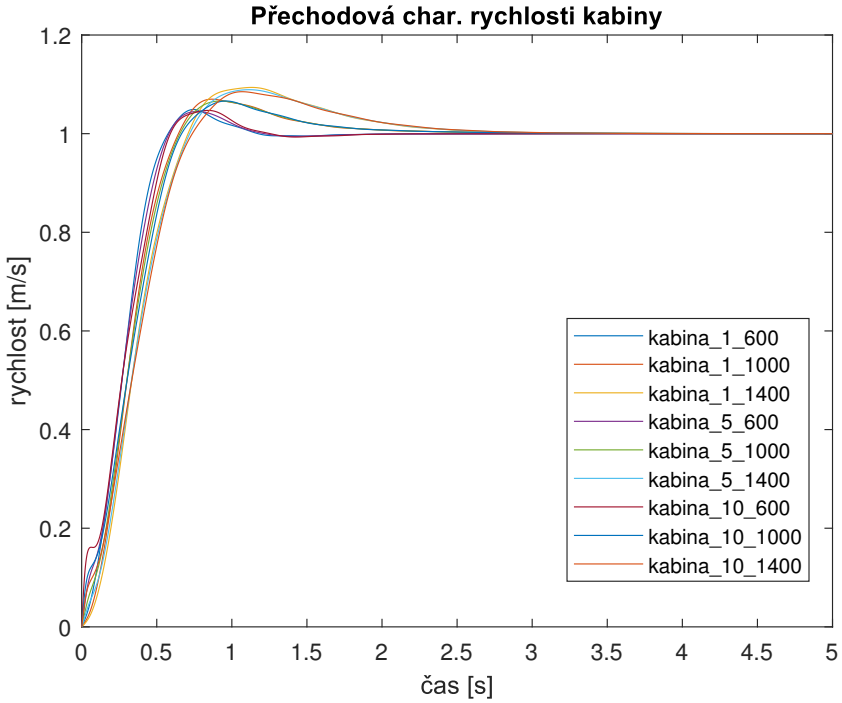
Na obrázku 12 pak vidíme odezvu na vstupní poruchu na straně kabiny, která je o něco pomalejší a má ne příliš pěkný průběh, což je dáno právě tím, že použitý notch filtr kazí právě na daných frekvencích odregulování poruch.

Na obrázku 13 jsou znázorněné přechodové charakteristiky všech devíti systémů (na straně motoru)

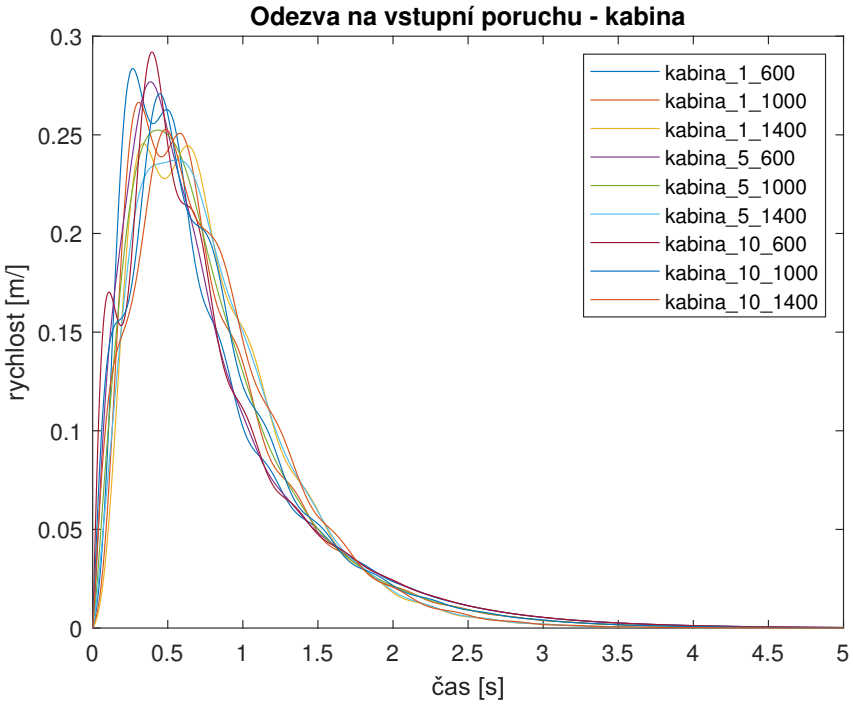
Na obrázku 14 je ještě odezva na vstupní poruchu na straně motoru

Na obrázku 15 jsou pak vyobrazeny Bodeho charakteristiky všech devíti systémů (na straně zátěže) s tímtéž regulátorem.

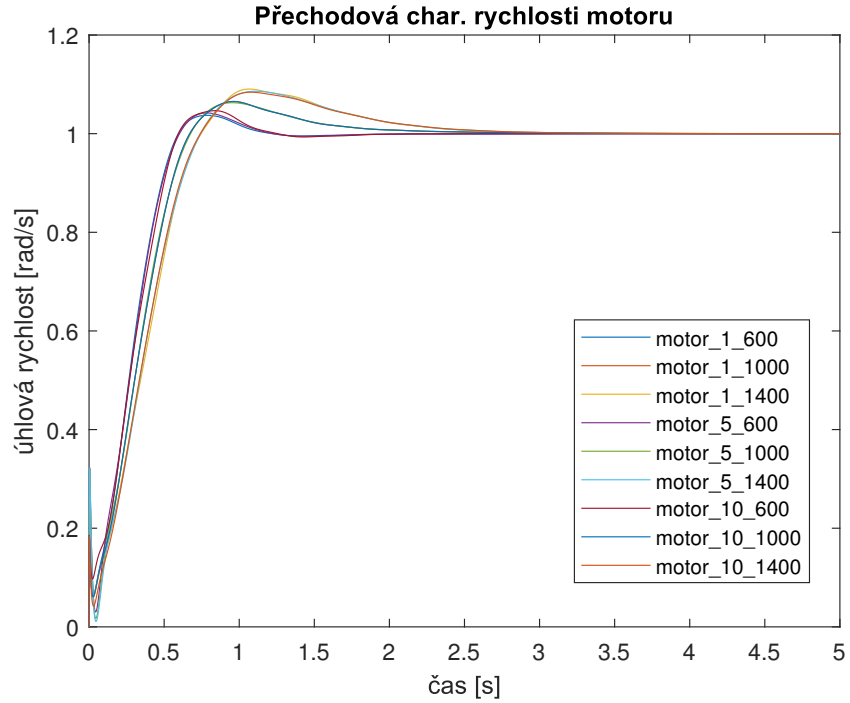
Bohužel zmiňovaná aplikace [2] neumožňuje vyexportovat vytvořené grafy, a tak přikládám jen snímek obrazovky, což neblaze ovlivnilo kvalitu. Domnívám se však, že základní vypovídací hodnota grafů je zachována, jen nejsou na pohled tak pěkné.



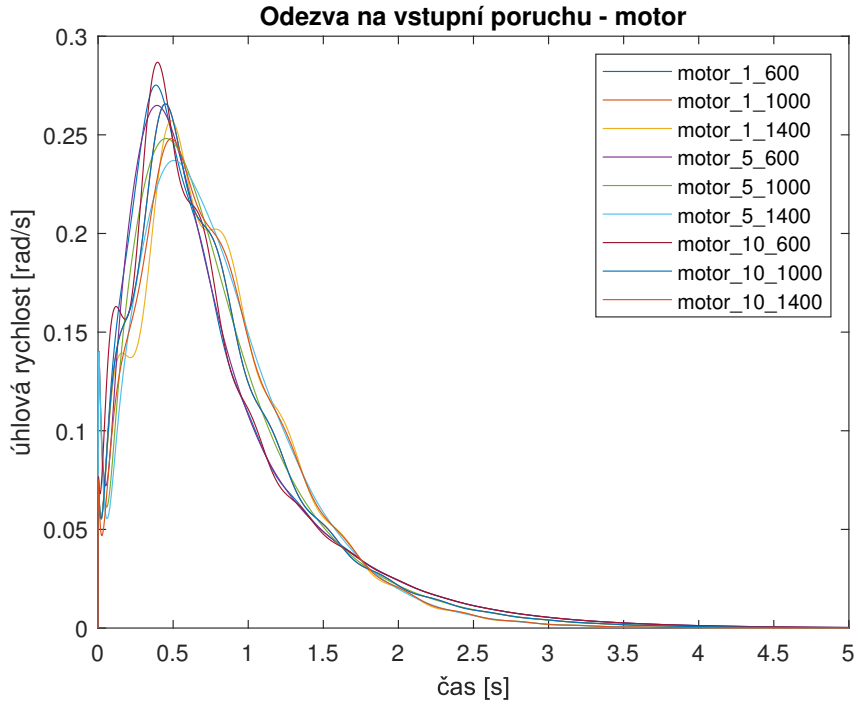
Obrázek 11: Přechodová charakteristika na straně kabiny, všech 9 systémů



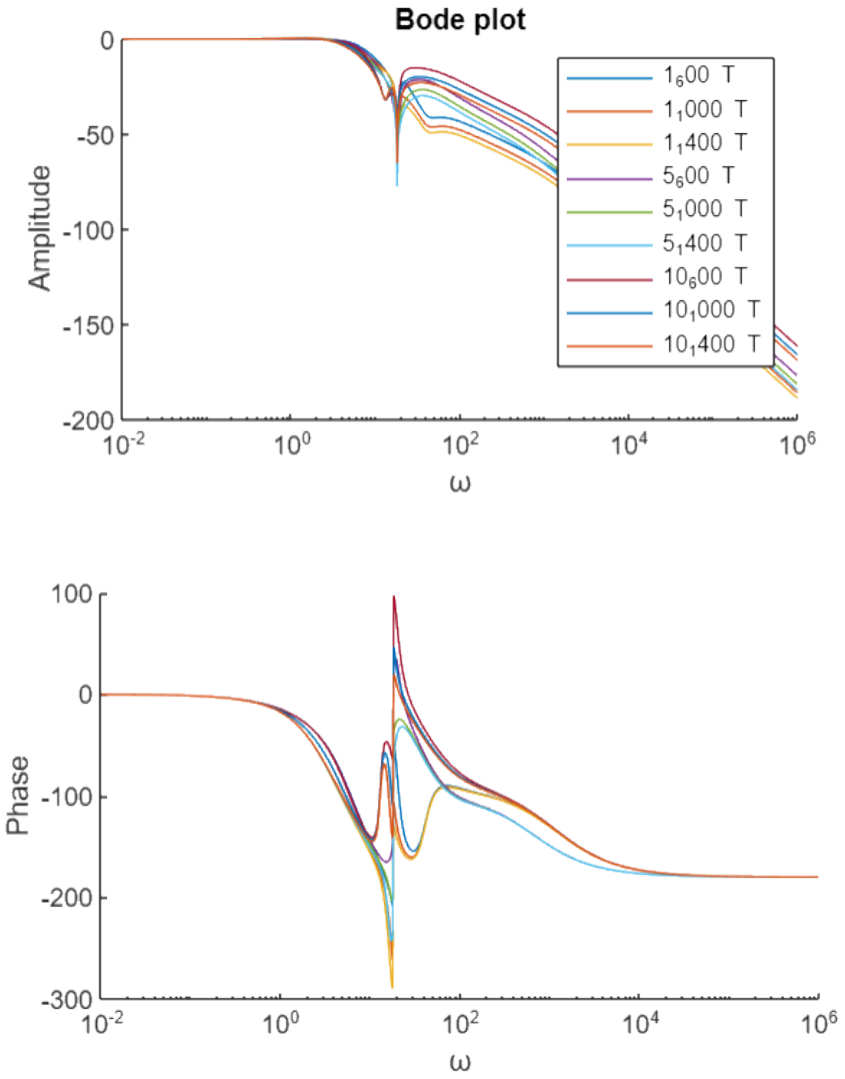
Obrázek 12: Odezva na vstupní poruchu na straně kabiny, všech 9 systémů



Obrázek 13: Přechodová charakteristika na straně motoru, všech 9 systémů



Obrázek 14: Odezva na vstupní poruchu na straně motoru, všech 9 systémů



Obrázek 15: Bodeho charakteristika na straně zátěže v aplikaci H_∞, všech 9 systémů

1.4 Generátor trajektorie

Pro regulaci je samozřejmě důležité vědět, kam regulovat. A k tomu slouží právě generátor trajektorie, který vygeneruje regulátoru určitou posloupnost požadovaných hodnot, které systém dovedou po vhodné křivce do kýženého stavu. Stejně tak nám generátor dává možnost definovat omezení kladená na chování uzavřené smyčky. Zohledňujeme zde například maximální požadovanou rychlost, zrychlení a definujeme tak parametry křivek, po kterých se systém může bezpečně pohybovat. Parametry, ze kterých trajektorii generujeme, jsou tedy (maximální) hodnota jerku (změna zrychlení), maximální povolené zrychlení, maximální rychlost a poloha, do které se chceme dostat (případně ještě počáteční poloha, ve které se nacházíme).

Ve výtazích je nutné neměnit zrychlení skokově kvůli komfortu pasažéra (někdy se jde ještě o jednu derivaci dál, tedy na změnu jerku). My se ale při generování pohybu dostáváme na úroveň jerku. Tím pádem může nastat 6 možností způsobu generování trajektorie [4, p. 15]. To, že některé případy vypadají zdánlivě stejně je dáno tím, že jednou je omezení zapříčiněno dosažením maximální rychlosti a jindy dosažením maximálního zrychlení.

Jelikož toto téma bylo již mnohokrát zpracováno, nebudeme zde předvádět celé odvození. Našli jsme však práci [4], kterou jsme při programování použili. Je v ní vše výborně a názorně vysvětleno, odvozen význam jednotlivých vzorců atd. V některých případech se tedy do práce odkazujeme či si vypůjčujeme obrázky a zde shrneme jen jakousi základní myšlenku algoritmu, který vygeneruje požadovanou trajektorii.

Prvním krokem je tedy rozhodnout podle zadaných parametrů, o jaký tvar zrychlení se bude jednat. To určíme na základě parametrů označených v citovaném článku jako v_a, s_a, s_v, které v podstatě vyjadřují obsah plochy pod křivkou zrychlení / rychlosti potřebný k dosažení požadované polohy (dáno tím, že mezi polohou, rychlostí atd. je vztah derivace – zpětně integrálu). Určujeme je na základě zadaných limitních hodnot.

Pokud určíme typ trajektorie, je možné spočítat analyticky již přepínací časy (čas, ve kterém dojde k přepnutí hodnoty jerku – změní se tedy průběh zrychlení, posléze rychlosti, tím se tvaruje křivka polohy, viz obrázek 16). Těchto přepínacích časů bude vždy 7, jen v některých případech některé z nich splynou a tudíž interval mezi nimi zanikne.

Pak tedy dochází k vyhodnocení v jednotlivých intervalech – výsledkem jsou snadno spočitatelné polynomy (lineární funkce, kvadratická... v podstatě vždy integrujeme jerk na zrychlení, zrychlení na rychlost atd.). Získáme tím funkce spojité v čase, tím pádem není problém při diskretizaci jen vybrat jejich hodnoty vždy v daném vzorku.

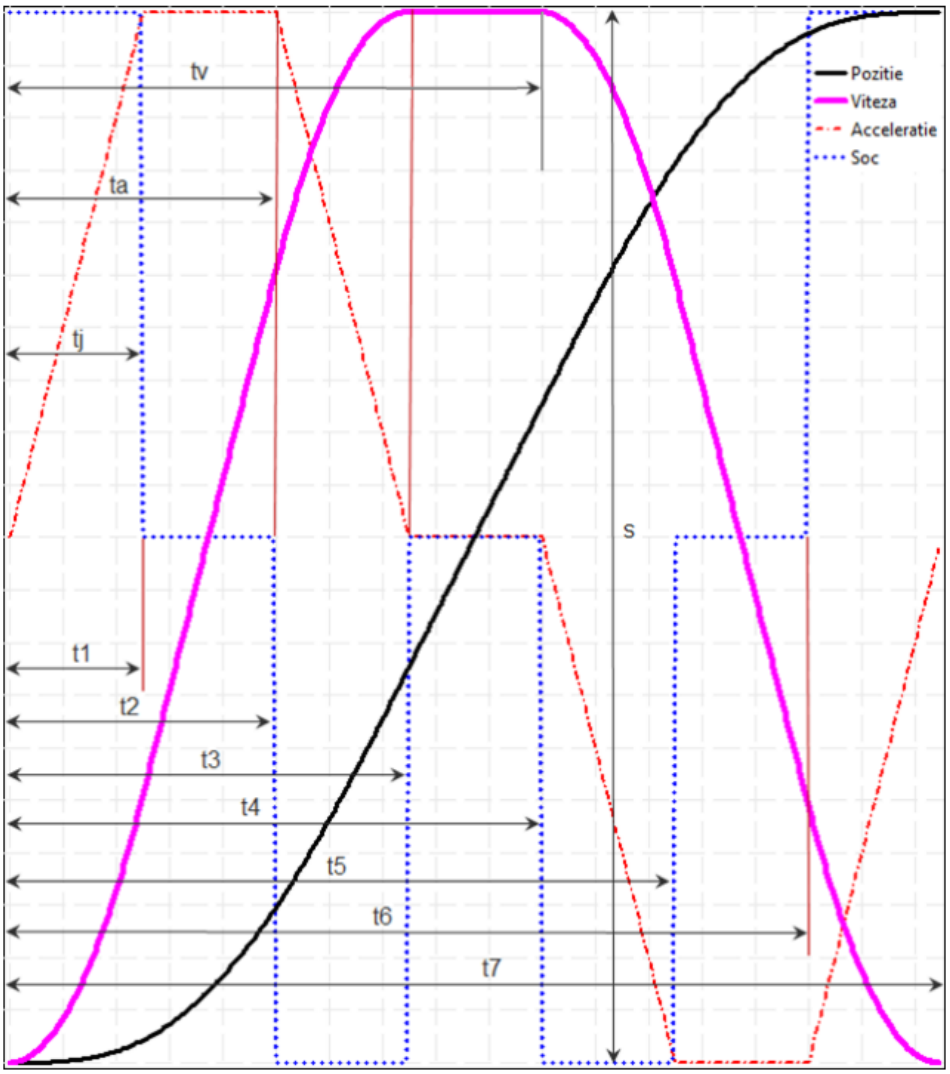
Problém je ale jinde. Výše zmíněná práce nepočítala s diskretizací, ale ta bude potřeba při reálném nasazení vždy, protože počítače ve spojitém čase zkrátka pracovat nedokážou. Čas poběží vždy po určitých kvantech. Pokud ale spočteme přepínací časy a ty se netrefí na hodnoty času generované počítačem, nedojde k přepnutí v přesný moment a tím pádem nedosáhneme v závěru přesně zadané polohy.

Uvažoval jsem tedy následovně: Jestliže přepínací čas není dělitelný periodou vzorkování, je třeba upravit zadané parametry (požadavky na maximální rychlost, zrychlení...). Nikdy nemůžeme přesáhnout původně zadané parametry. Přepínací čas tedy zaokrouhlíme tak, aby seděl s periodou vzorkování a zpětně přepočteme parametry (čas musíme zaokrouhlit na správnou stranu, abychom parametry vždy snížili, ne zvýšili).

Toto se pokusím demonstrovat na jednoduchém příkladu:

```
t = s/v_max;
t = t { mod(t,vzorek) + vzorek;
v_max = s/t;
```

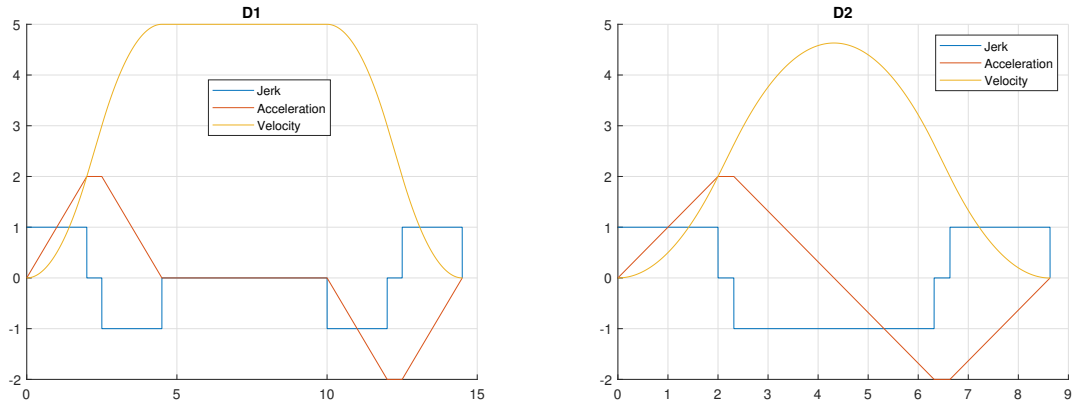
Od přepínacího času odečteme zbytek po dělení periodou vzorkování a ač to na první pohled vypadá nesmyslně, přičteme jeden vzorek, protože v_{max} je ve jmenovateli, když ji budeme chtít upravit, budeme s (celkovou dráhu, kterou plánuji urazit) dělit tímto časem, takže pokud bude větší, v_{max} tím zmenšíme, jak jsme chtěli.



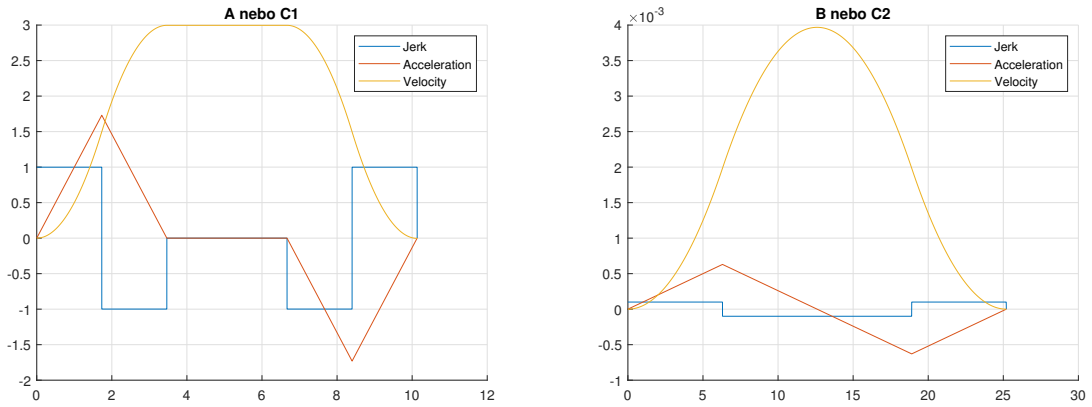
Obrázek 16: Znázornění přepínacích časů [4, p. 8]

Možností pro přepínací časy je ve skutečnosti nakonec méně, protože je nám vlastně jedno, jestli omezení vzniklo kvůli maximální rychlosti nebo zrychlení, oddělené jsou tyto možnosti spíše formálně. Typy, které mohou nastat, a 4 programově odlišené varianty, lze vidět na následujících obrázcích (viz obrázek 17 a obrázek 18). Označené jsou pomocí stejných písmen, jako v [4, p. 15]. Tyto výsledky jsou již vygenerované i se zohledněním daného vzorku diskretizace.

Značení **A, B, C1, C2, D1, D2** je použité v analogii se zmiňovanou prací, na kterou se v popisu odkazuji [4]. V podstatě odpovídá jednotlivým variantám. Varianta **A** značí případ, kdy nedochází ke konstantnímu držení zrychlení na jiné než nulové hodnotě, ale rychlost chvíli konstantní zůstává, varianta **B** oproti předchozí neobsahuje ani konstantní rychlost. Varianta **C1/2** je analogická s dvěma předchozími možnostmi, ale zalomení nastává vlivem omezení maximální hodnoty zrychlení a nikoliv rychlosti. Varianta **D1/2** pak popisuje obdobné případy jen s konstantním zrychlením na nenulové hodnotě. Analyticky tak může nastat těchto 6 možností s ohledem na tvary průběhu zrychlení a na *zaražení* vlivem maximální rychlosti nebo zrychlení.



Obrázek 17: Možnost D1 a D2



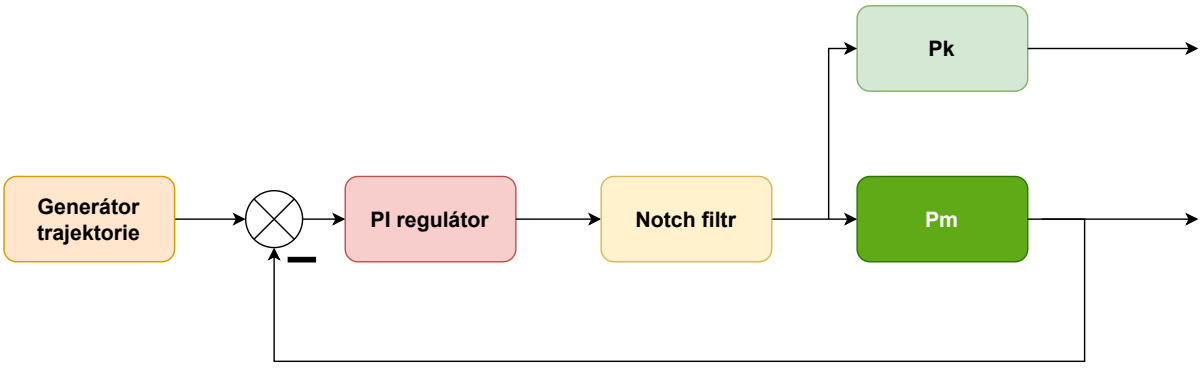
Obrázek 18: Možnost A nebo C1 a B nebo C2

1.5 Výsledky robustní regulace na nelineárním modelu s generátorem trajektorie

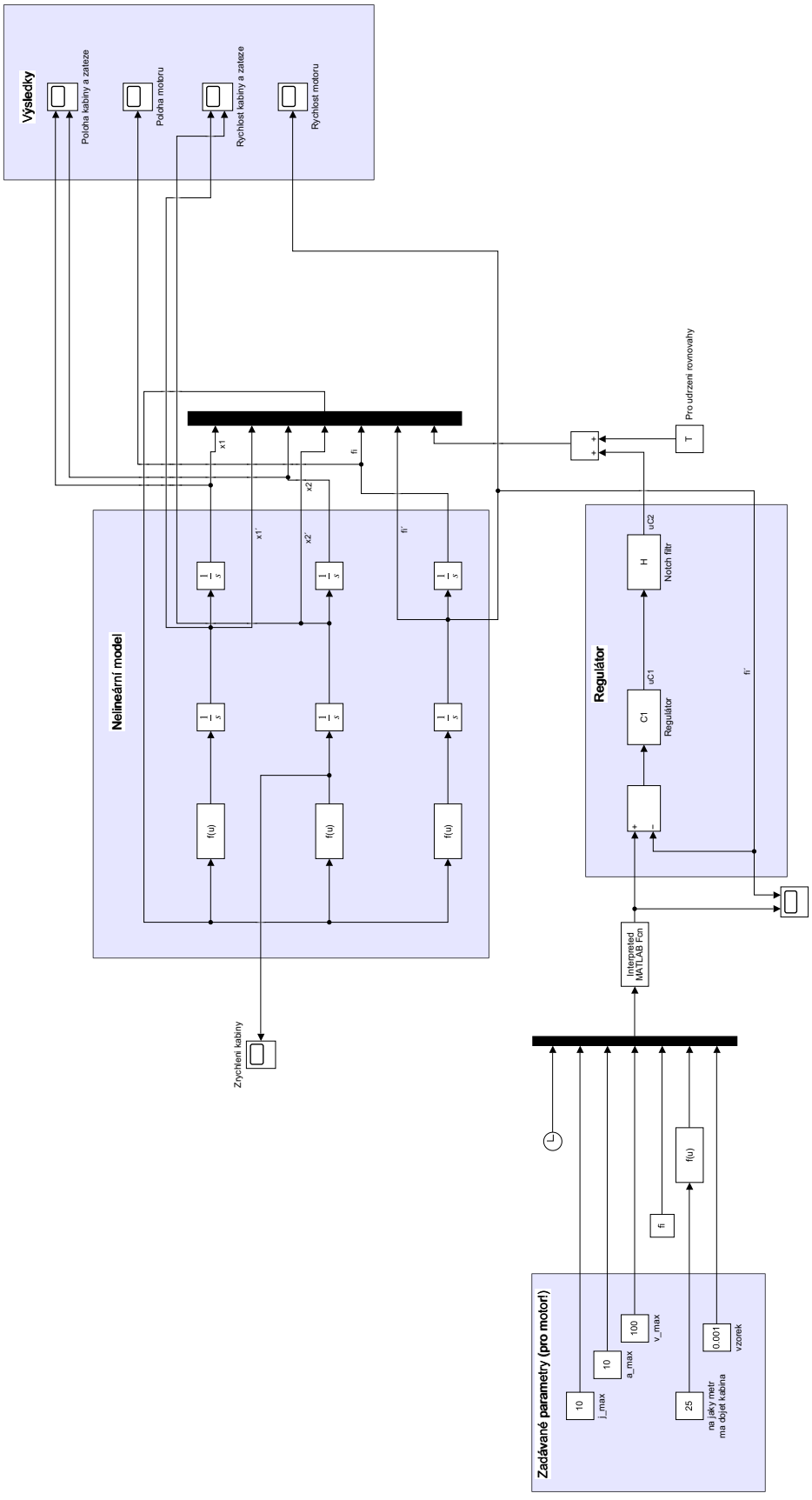
Poslední záležitostí této části bylo všechny dílčí úkoly složit do jednoho celku. Je to již kompletní použitelné řešení, ze kterého se bude odvíjet další práce. Na obrázku 19 je vidět celkové schéma, jak jsou za sebou jednotlivé komponenty seskládané. Naopak na obrázku 20 je kompletní schéma v Simulinku, kde je jasněji vidět, jak model funguje, je zde nelineární model (popsaný pohybovými rovnicemi, viz vztah 8), generátor trajektorie a výše popsaný PI regulátor. Generátor trajektorie je zde předělaný do funkce, která vrací aktuální hodnotu rychlosti v čase generovaném Simulinkem (je potřeba pohlídat, aby vzorkování v Simulinku bylo shodné se zadanou velikostí vzorku pro generátor trajektorie). Jelikož výtah řídíme pomocí motoru, jsou parametry zadávané do generátoru pro motor (tedy maximální rychlost otáčení motoru, maximální zrychlení atd., což dává to smysl i z toho hlediska, že motor jako takový má tyto parametry dané již z výroby a nalezneme je v manuálu). Jen zadávanou dojezdovou polohu jsme pomocí přepočtu nastavili tak, aby se dala zadávat jako poloha kabiny (dojde ale k přepočtu na polohu motoru).

V celé této části provádíme simulace s následujícími parametry:

$$\begin{gathered}
\text{Počáteční poloha kabiny: } x_{20} = 6 \\
\text{(z té již vyplývají ostatní počáteční podmínky)} \\
\text{Maximální rychlost motoru: } 100\ rad/s \\
\text{Maximální zrychlení motoru: } 10\ rad/s^2 \\
\text{Maximální jerk motoru: } 10\ rad/s^3 \\
\text{Cílová poloha kabiny: } x_{20} = 25 \\
\text{Vzorkování: } 0.001\ s
\end{gathered} \tag{38}$$

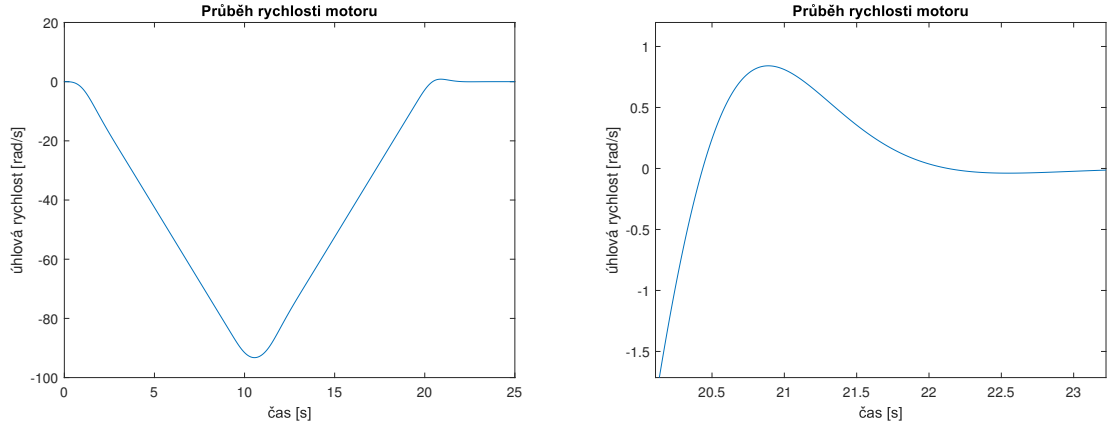


Obrázek 19: Schéma modelu s generátorem trajektorie



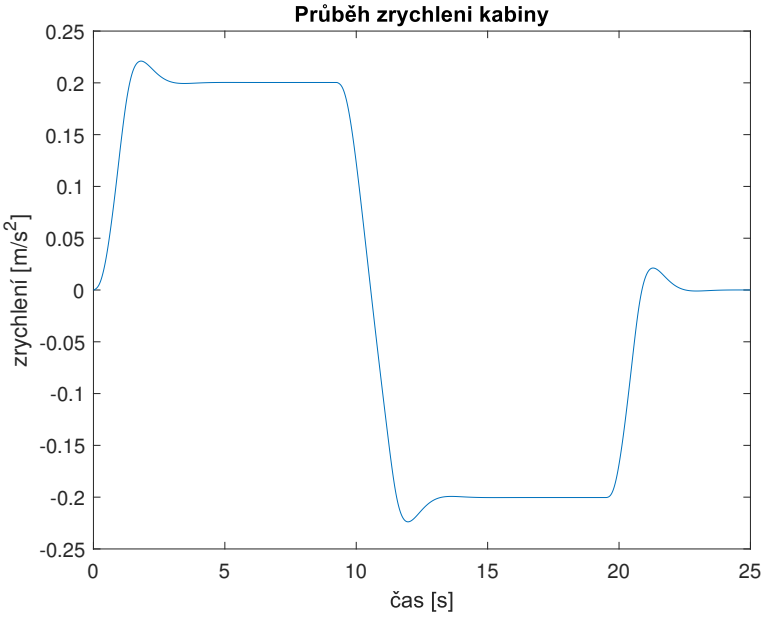
Obrázek 20: Schéma modelu s generátorem trajektorie v Simulinku

Prvním důležitým výsledkem je samotná regulace rychlosti motoru (viz obrázek 21) spojená právě s generátorem trajektorie, díky kterému je překmit výrazně snížen, protože regulátoru předáváme postupné "velmi nízké skoky".



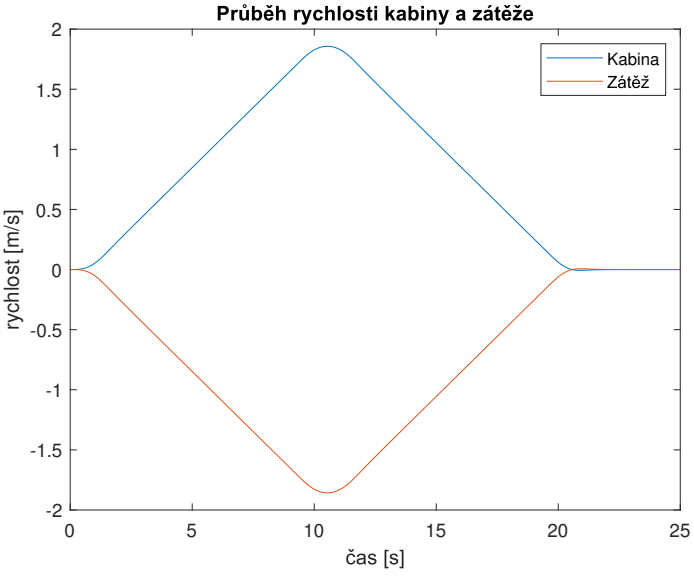
Obrázek 21: Průběh rychlosti motoru s vyobrazeným detailem (vpravo)

Dalším důležitým aspektem, který bychom měli sledovat, je zrychlení kabiny výtahu (viz obrázek 22). To je z toho důvodu, že u výtahů je průběh zrychlení velice důležitý z hlediska komfortu posádky. Jsou přímo dané normy pro průběhy zrychlení, při kterých se lidem nedělá špatně. Nejde čistě jen o zrychlení jako takové, ale i o jeho sklon (tedy vlastně velikost jerku). Obecně čím dále se v těchto derivacích polohy dostáváme, tím hladší můžeme získat křivku zrychlení a dosáhneme většího komfortu uživatelů a můžeme si tak dovolit i větší rychlosti.



Obrázek 22: Průběh zrychlení kabiny během regulace

Další věci, které nás zajímají, je samozřejmě i průběh rychlostí a poloh kabiny a zátěže (protizávaží). Nejsou tak důležité, jako předchozí grafy, ale rozhodně nás zajímá, jak hladce a plynule kabina dojede a zastaví. Nejprve tedy rychlosti (viz obrázek 23).

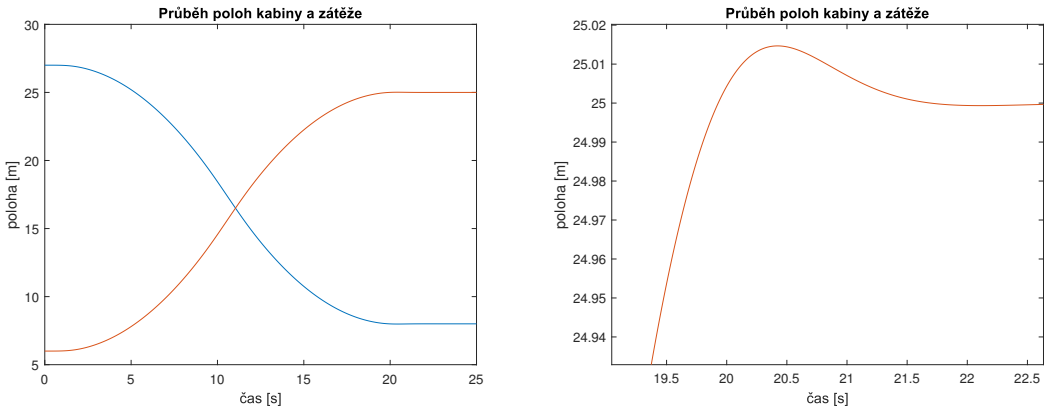


Obrázek 23: Průběh rychlostí kabiny a zátěže během regulace

U polohy můžeme vidět skutečně hezký plynulý nájezd způsobený především generátorem trajektorie a tím, že jsme se dostali až na úroveň jerku. Samozřejmě je zde zase nějaký překmit, v tomto případně konkrétně asi 5 mm (viz obrázek 24), ale to se ve skutečnosti vyřeší jinak.

V praxi by to takhle jen pomocí regulace přes rychlost fungovat nemohlo, protože jakmile někde bude tření větší/menší, než jsme počítali, nebo lidé v kabině trochu poskočí, ovlivní se tím rychlost, která již svým trochu jiným průběhem zanese poměrně velkou odchylku do polohy a za chvíli se celý systém rozjede. Proto by bylo vhodné zavést ještě kaskádní regulaci pomocí polohové smyčky.

Ani to však není nutné – ve skutečnosti se to často řeší tak, že kabinu doregulujeme do přibližné hodnoty, kterou požadujeme. A pak již necháme výtah jet velmi nízkou rychlostí, dokud nenarazí na koncový spínač a nezastaví. Tím se systém vlastně opět při každé jízdě zkalibruje.



Obrázek 24: Průběh polohy kabiny a zátěže během regulace i s detailem (vpravo)

Shrnutí

S pomocí matematicko-fyzikálního modelování, metod řízení lineárních systémů a chytré aplikace H_∞ [2] jsme získali vhodný robustní regulátor, doplněný notch filtrem a generátorem trajektorie, který zvládne uřídit nejen linearizovaný model ale i nelineární simulaci výtahu, a to pro různá zatížení a pro různé scénáře pohybu mezi rozličnými patry (tudíž je použitelný v praxi).

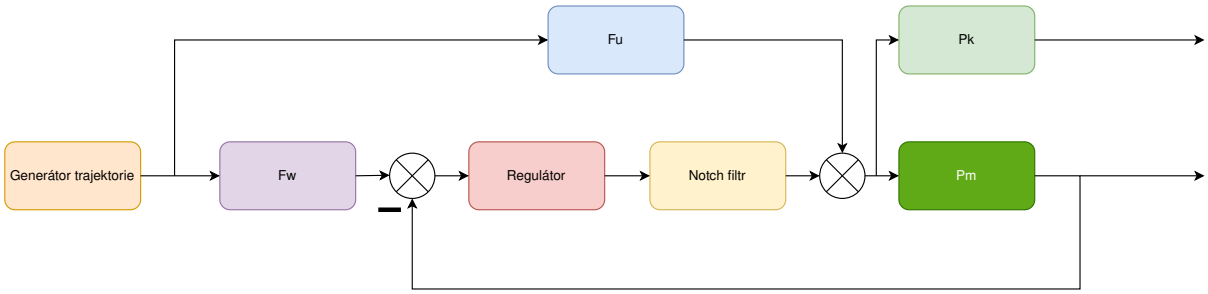
Tento regulátor následně využijeme jako výchozí v našich následujících pracích, kde se zaměříme na vylepšení vlastností regulace pomocí speciálních metod zpětnovazebního (Kubeš) a přímovazebního (Tvrz) řízení.

2 Přímovazební řízení

Úvod

Dopředná vazba nebo přímovazbení řízení obecně se používá zejména tehdy, když máme k dispozici nějaký prediktor. Pokud známe požadované chování určitým způsobem dopředu, je to vhodná strategie. Zde tedy mohu navázat na generátor trajektorie, o kterém byla řeč v jedné z předchozích kapitol. Jelikož reguluji rychlost, mám k dispozici ještě dvě derivace vstupu – zrychlení a jerk. Tyto derivace mohu dále využít při návrhu vstupního tvarovacího filtru F_W a filtru dopředné vazby na řízení F_u, které jsou znázorněné na obrázku 25.

Zde je tedy již kompletní schéma, dále v této kapitole bych však rád vysvětlil proč a zejména jak dojdeme k tvarům těchto filtrů, jak zužitkujeme generátor trajektorie a že díky takovéto predikci můžeme simulačně dosáhnout lokálně dokonalých výsledků.



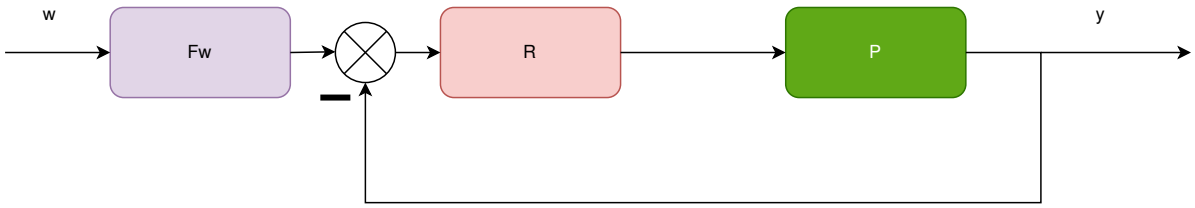
Obrázek 25: Schéma struktury přímovazebního řízení

2.1 Dopředná vazba od referenčního signálu

Obecně máme k dispozici dvě možnosti / varianty, jak toto udělat. V úvodu jsem ukázal schéma, kde jsou již spojené a použité zároveň, pro odvození si to však ukážeme na úplně jednoduchém obecném modelu regulační smyčky.

2.1.1 Návrh vstupního tvarovacího filtru F_W

V tomto případě se snažíme tvarovat referenční veličinu takovým způsobem, aby ji náš kompletní systém (tedy včetně regulátoru) mohl ideálně sledovat. Tento případ znázorněný na obrázku 26 je v praxi často používán už jen proto, že kolikrát nemáme možnost přímo zasahovat do vstupu, a tak jen vhodně tvarujeme referenci. Na druhou stranu je tento filtr mnohem složitější, jak vyplyne z matematického odvození.



Obrázek 26: Schéma vstupního tvarovacího filtru F_W

Nejprve si zapíšeme, jak bude vypadat přenos z reference až na výstup:

$$T_F = \frac{Y}{W} = F_W \frac{RP}{1+RP} = F_W T \tag{39}$$

Vidíme tedy, že tento přenos lze zapsat jako náš filtr vynásobený komplementární citlivostní funkcí, z čehož můžeme vyvodit požadavek na tvar F_W pro dokonalé sledování.

Pro ideální sledování musí platit, že

$$y = w \;\rightarrow\; T_F \stackrel{!}{=} 1$$

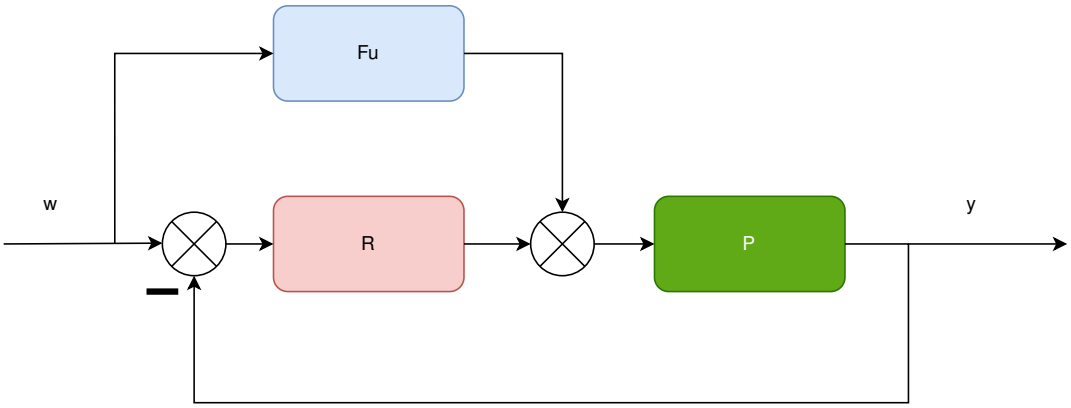
$$1 \stackrel{!}{=} F_W T \;\rightarrow\; F_W = T^{-1} \tag{40}$$

Toto však není v praxi vůbec snadné, protože jakmile je relativní řád $r(T) > 0$, tak je získaná inverze nekauzální. Stejně tak je problém při invertování neminimálně fázového systému s nestabilními nulami, jelikož pak vznikne filtr s nestabilními póly, tudíž nestabilní a nepoužitelný. Právě teď se hodí generátor trajektorie, který mi dává možnost *předpovědět* derivace dopředu, tím pádem možnost použít nekauzální filtr a získat alespoň lokálně při zanedbání poruch, šumů a podobných věcí ideální vysledování referenční veličiny.

2.1.2 Návrh filtru dopředné vazby na řízení F_U

V tomto případě jdeme o něco jednodušší cestou a to tak, že se snažíme dát systému jako takovému vstup, který jej správně provede po požadované trajektorii. V ideálním případě by tak měla být odchylka e nulová, ale vzhledem k tomu, že v praxi to tak rozhodně nechodí (a to nemluvím o tom, že můj systém je nelineární a já mám k dispozici linearizaci v 10 bodech), je nutné zde zachovat stále zpětnovazební regulační smyčku. Ale i tak zachováváme důležitou vlastnost – velmi zjednodušujeme práci regulátoru, který v tuto chvíli má za úkol již jen dorovnávat to, co nezvládne dopředná vazba.

Tato varianta znázorněná na obrázku 27 je však v praxi, jak jsem již říkal, hůře použitelná, protože ne vždy si můžeme dovolit zasahovat přímo do vstupu do systému, tedy sahat mezi regulátor a systém jako takový.



Obrázek 27: Schéma vstupního tvarovacího filtru F_W

Opět se pokusím vše ukázat a vysvětli za pomoci vzorců. Zde to bude o něco jednodušší, protože, jak jsem řekl, chceme dosáhnout nulové regulační odchylky, tím pádem nás zajímá přenos z reference na regulační odchylku:

$$F_{WE} = \frac{E}{W} = \frac{1}{1+RP} - \frac{F_U P}{1+RP} = \frac{1 - F_U P}{1+RP} \tag{41}$$

Zde se sice nedostaneme k žádnému popisu pomocí citlivostní funkce, ale vidíme, že nás bude zajímat vlastně jen čitatel tohoto přenosu:

Pro ideální sledování musí platit, že

$$e = 0 \;\rightarrow\; 1 - F_U P \stackrel{!}{=} 0$$

$$1 \stackrel{!}{=} F_U P \;\rightarrow\; F_U = P^{-1} \tag{42}$$

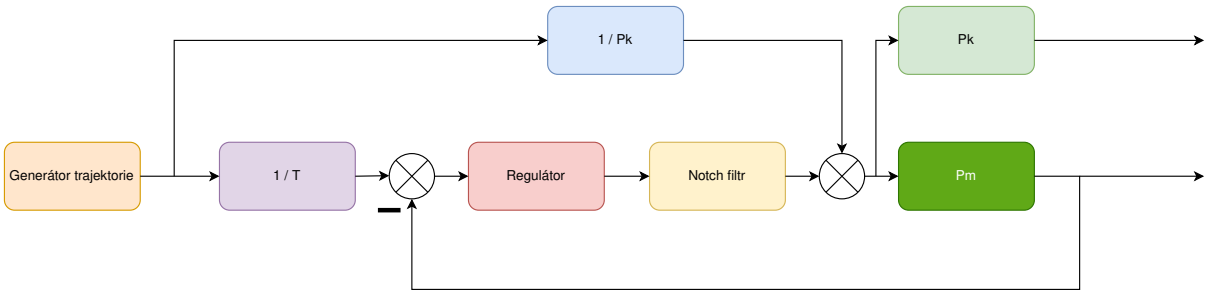
Opět je zde samozřejmě problém s ideální inverzí systému a jeho relativním řádem, ale to vše stále mám podchycené díky generátoru trajektorie. Pokud tedy mohu použít tuto variantu, mám zde další výhodu – vidím, že stačí invertovat jen systém samotný a ne komplementární citlivostní funkci, která vlastně popisuje chování celé uzavřené smyčky, což je v mém případě kromě systému a regulátoru rovněž notch filtr.

2.2 Aplikace dopředného řízení na tříhmotový systém

Pokud bych chtěl jedno z výše zmíněných schémat (viz obrázek 27 a obrázek 26) aplikovat na výtah s využitím navrženého robustního regulátoru, neuspěl bych. Je to tím, že z důvodů, které jsem již několikrát zmiňoval, řídím polohu kabiny pomocí přenosu na rychlost motoru. V okamžiku, kdy bych se pokusil tvarovat referenční veličinu pomocí inverze komplementární citlivostní funkce, využíval bych jen znalosti o přenosu na motor a nemohl bych vůbec zajistit ani nějak předpovědět, jak se bude chovat kabina. V opačném případě při použití dopředné vazby na řízení bych sice uvažoval kabinu, protože mě samozřejmě zajímá její chování, ale zase bych opomenul přenos na motor, který je pro mě neméně důležitý, protože je skrze něj uzavřená zpětná vazba, tudíž by nedošlo k požadovanému vynulování regulační odchylky. Generoval bych vstup takový, jako by byl ve zpětné vazbě přenos na kabinu a ne na motor.

Vidíme tedy, že se krásná myšlenka poněkud rozpadá. Teď jsem však mluvil o tom, že každý přístup by vytvořil dokonalé podmínky pro jeden z přenosů, ale zanedbal by ten druhý. Co je tedy spojit, využít oba přístupy a získat tak dokonalé řízení pro kabinu zohledňující přenos na motor v uzavřené smyčce?

Touto úvahou se již konečně dostáváme ke schématu uvedenému v úvodu, které však nyní můžeme doplnit o tvar jednotlivých filtrů, který jsem odvodil před chvílí. Toto schéma (viz obrázek 28) dává již přímou nápovědu, jak vytvořit jednotlivé filtry a implementovat je pro pokus v Simulinku.

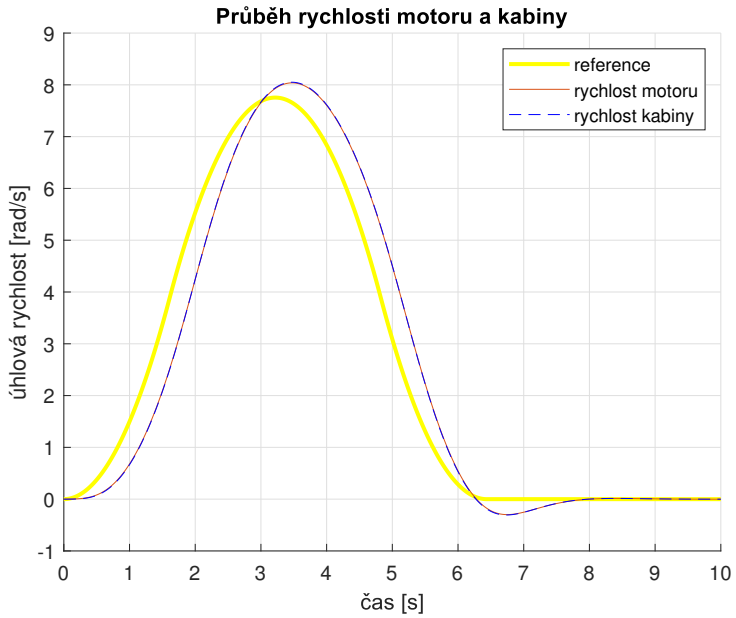


Obrázek 28: Finální schéma struktury přímovazebního řízení

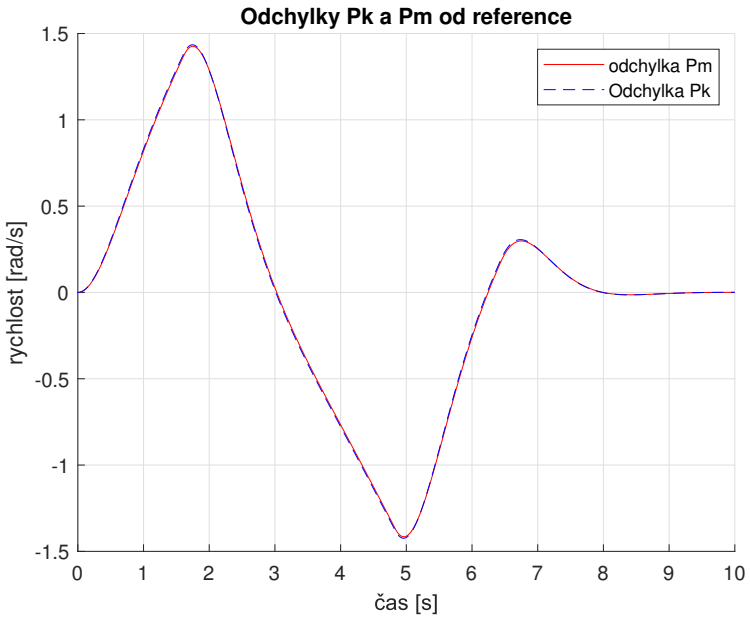
2.2.1 Robustní regulátor v daném patře pro porovnání

Abychom mohli následující výsledky přímovazebního řízení s něčím porovnávat, ukážu zde ještě na začátek chování čistě samotného robustního regulátoru ve stejném patře, v jakém se budu zabývat experimentováním s tvarovacím a dopředným filtrem.

Na obrázcích 29 a 30 si můžeme všimnout nedokonalostí, jako je poměrně znatelný překmit v regulaci rychlosti, který se pak samozřejmě projeví i na poloze. Na odchylce lze vidět velké zpoždění regulace, které se přímou vazbou právě snažím odstranit.



Obrázek 29: Finální schéma struktury přímovazebního řízení



Obrázek 30: Finální schéma struktury přímovazebního řízení

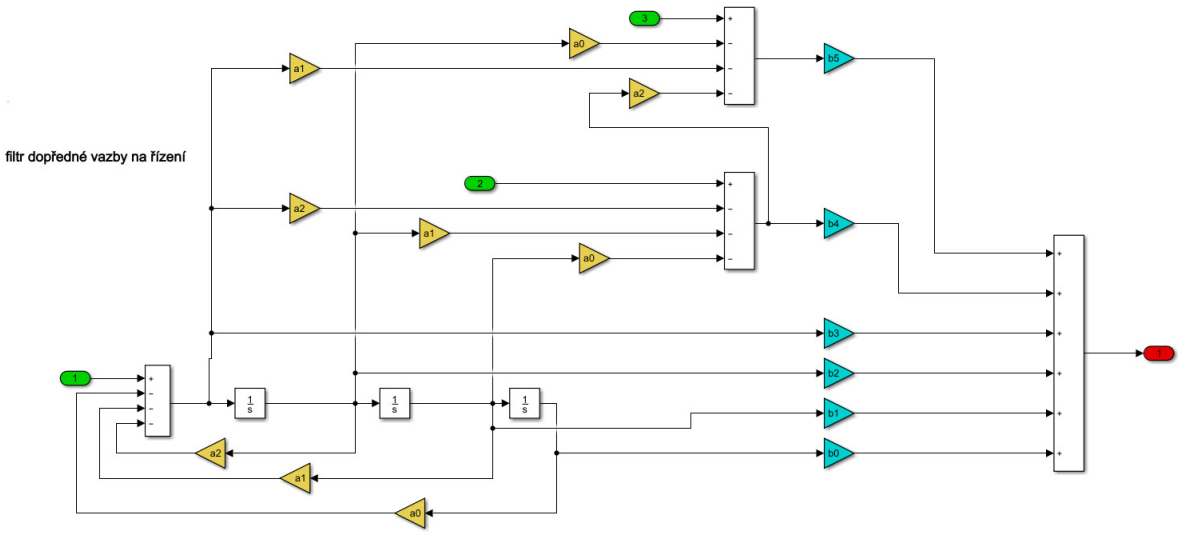
2.2.2 Implementace v Simulinku pro jedno patro

Pro první pokusy jsem si zvolil páté (prostřední) patro. Je to z toho důvodu, že zřejmě nejlépe zachycuje postupnou změnu dynamiky systému, ale jinak je to vcelku jedno. Důležité je, že znám přesně přenosy na rychlost motoru a kabiny, se kterými budu pracovat. Ideálně totiž bude toto řízení logicky fungovat jen okolo takto linearizovaného modelu (tedy dokud bude model odpovídat zvoleným přenosům). To znamená, že prozatím předpokládám pohyb na velmi malém úseku, který lze ještě aproximovat pomocí linearizovaného modelu v pátém patře.

Nejprve začnu s ukázkou rozložení obecné přenosové funkce. Konkrétně jsem si tedy zvolil invertovanou přenosovou funkci na rychlost kabiny, kterou ostatně používám. Inverzi celé komplementární citlivostní funkce ukazovat nebudu, protože je to stejný princip, jen má mnohem vyšší řád a ukázka by tudíž byla nepřehledná.

Uvažujme tedy přenos v takovémto tvaru (viz vztah 43), který odpovídá přenosu na rychlost kabiny v každém patře (samozřejmě s odlišnými koeficienty). Abychom mohli využít apriorní znalosti derivací vstupního signálu a použít tento nekauzální přenos, je nutné rozložit přenosovou funkci a namodelovat ji ze základních bločků. Na obrázku 31 je vše pro lepší orientaci znázorněno barevně – zeleně vstupy (tedy rychlost, zrychlení a jerk naší trajektorie), červeně výstup, žlutě koeficienty jmenovatele, modře koeficienty jmenovatele. Zde ukázaným způsobem je možné namodelovat libovolný nekauzální přenos za předpokladu znalosti dostatečného počtu derivací.

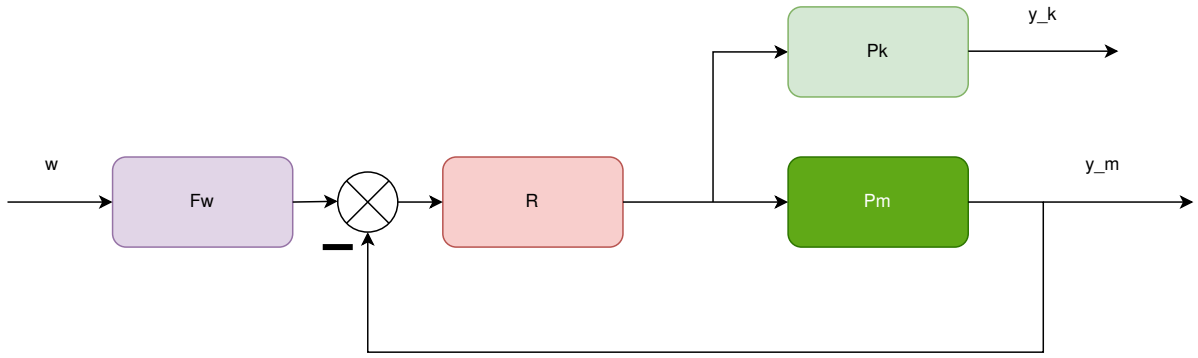
$$P_k^{-1} = \frac{b_5 s^5 + b_4 s^4 + b_3 s^3 + b_2 s^2 + b_1 s + b_0}{s^3 + a_2 s^2 + a_1 s + a_0} \tag{43}$$



Obrázek 31: Ukázka rozložení přenosové funkce

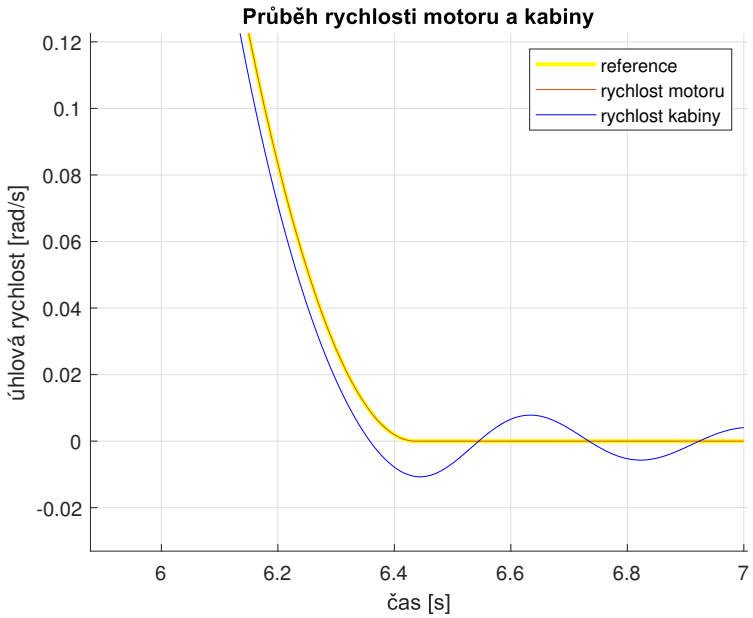
Již jsem ukázal, jakým způsobem je vytvořena a použita inverze systému spolu s generátorem trajektorie. V tuto chvíli se tedy můžeme vrátit zpět ke schématu kompletního řízení (viz obrázek 28) a to celé sestavit v Simulinku. Ještě je možná dobré podotknout, že do generátoru nastavuji parametry pro motor (tedy maximální rychlost, zrychlení a jerk motoru), protože právě přes motor je vedena regulace. V následujících pokusech tedy uvidíte, že hodnoty pro rychlosti, polohy atd. kabiny i motoru jsou shodné. To je dáno tím, že pokusy dělám na přenosech, které v sobě nemají zahrnutou konverzi mezi rychlostí kabiny a motoru. Říkal jsem si, že to není důležité a z hlediska vizualizace výsledků možná dokonce lepší, protože je zde lépe vidět porovnání chování právě na straně motoru a zátěže. Tento nedostatek je však samozřejmě odstraněn v okamžiku, kdy regulační smyčku připojím ke *skutečnému* nelineárnímu modelu, který má právě tuto konverzi již zahrnutou ve své dynamice.

Nyní můžeme konečně přikročit k výsledkům. Rád bych začal u toho, co jsem popisoval v části o teorii dopředné vazby a to k ověření tvrzení, že jen jeden nebo druhý filtr sám o sobě nebude vhodný. Jedná se tedy o takovéto schéma (viz obrázek 32), kdy se snažíme tvarováním referenční veličiny docílit vhodného chování.



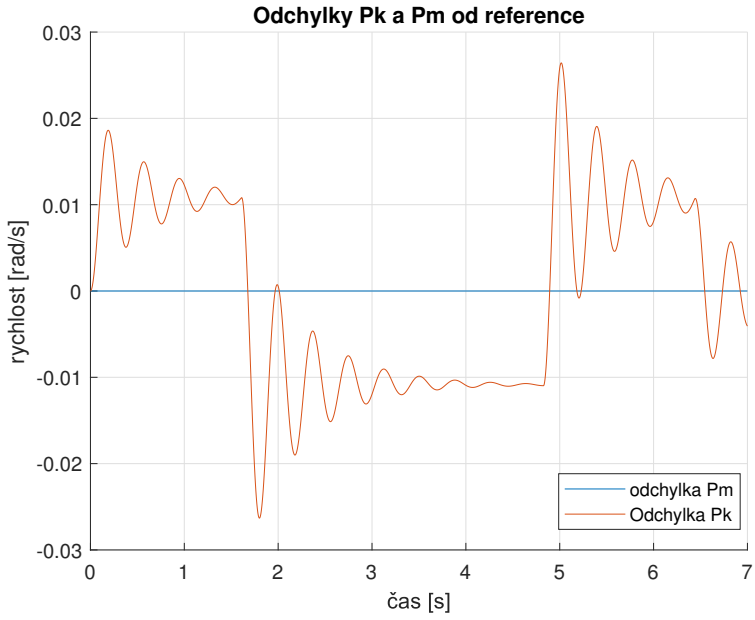
Obrázek 32: Schéma experimentu s filtrem F_W

Zde (viz obrázek 33) je detail zachycený v okamžiku, kdy dochází k brzdění na konci, tedy rychlost se dostává zpět na nulu. Můžeme si všimnout, že motor skutečně sleduje referenci velmi dobře, ale kabina kmitá (respektive rychlost kabiny, ale to se samozřejmě promítne i do polohy).



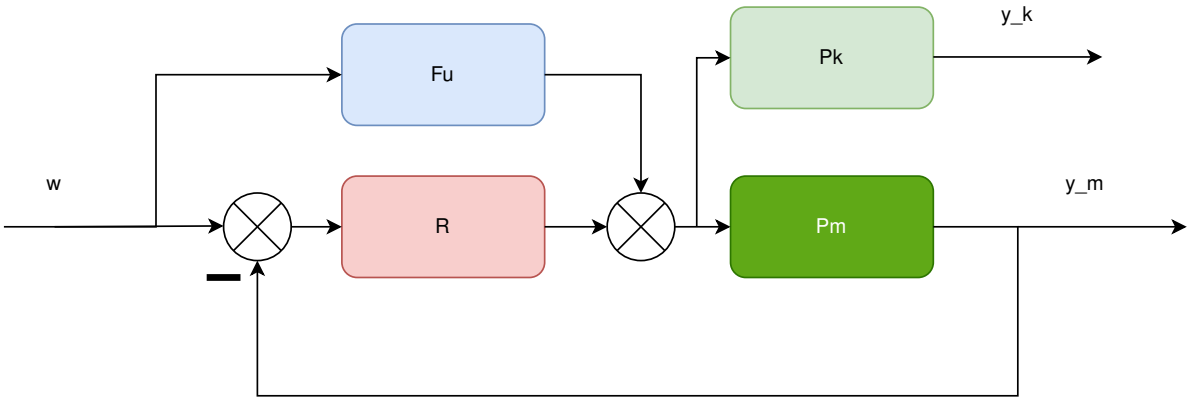
Obrázek 33: Výsledky experimentu s filtrem F_W

Dále je to ještě lépe vidět na grafu s vyobrazenou odchylkou od požadované reference (viz obrázek 34). Rychlost motoru skutečně sleduje referenci bezchybně, ale u rychlosti kabiny je problém, jak jsme očekávali, protože zkrátka nebereme v potaz dynamiku přenosu na rychlost kabiny.



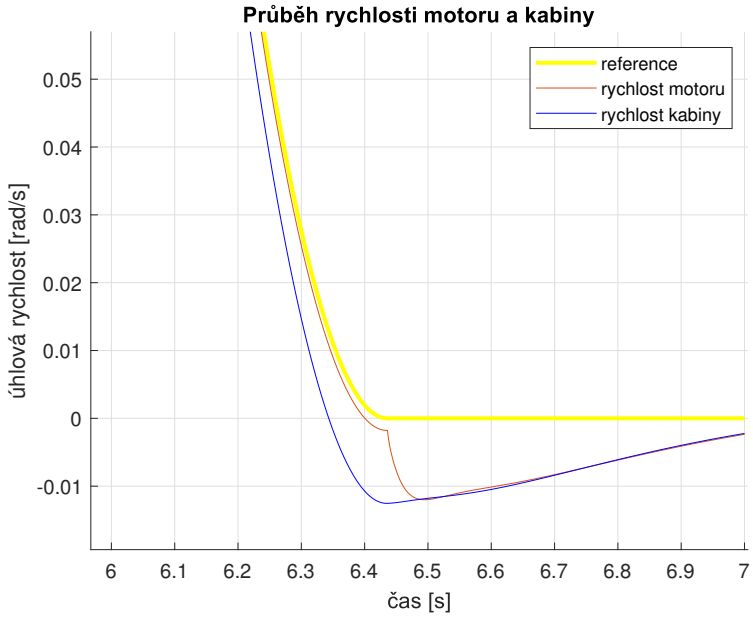
Obrázek 34: Odchylky u experimentu s filtrem F_W

Dalším případem je tedy pouze dopředná vazba, která bere v potaz jen dynamiku kabiny (viz obrázek 35)

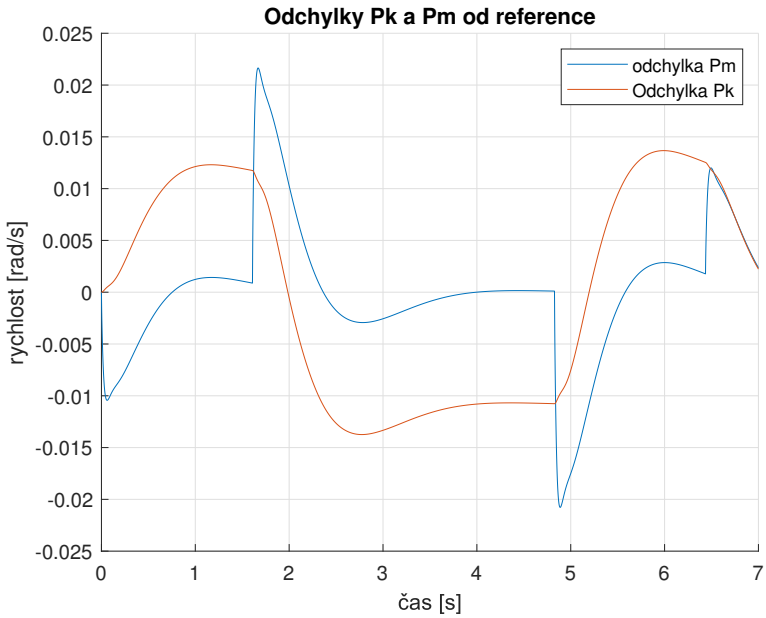


Obrázek 35: Schéma experimentu s filtrem F_u

Opět si můžeme všimnout, že se model nechová, jak by měl (viz obrázek 36). Zde dochází k úplným nesmyslům a rozhodně si takové chování nepřejeme. Znovu se můžeme podívat na průběh odchylek od reference (viz obrázek 37). Toto nepředvídatelné chování je způsobeno tím, že ovlivňujeme sice přenos na kabinu, ale ve smyčce je přenos na motor, tím pádem vlastně děláme něco úplně jiného, než bychom chtěli – to se logicky promítne dál.

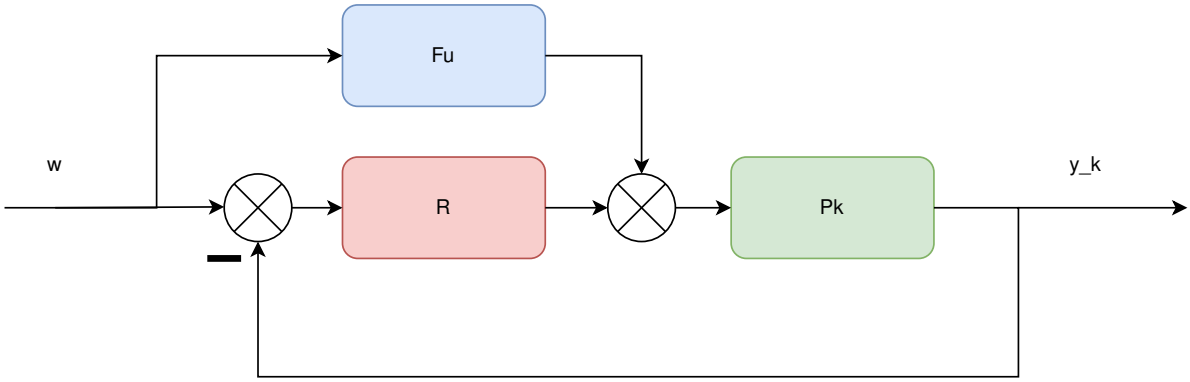


Obrázek 36: Výsledky experimentu s filtrem F_u



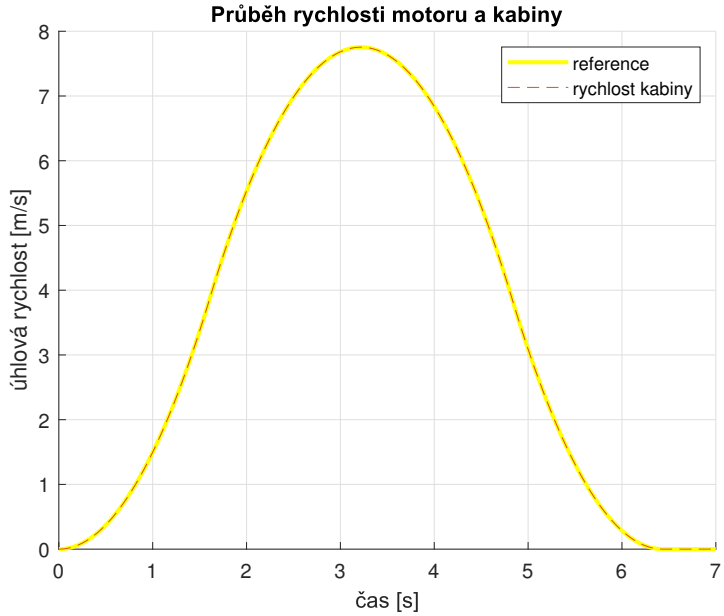
Obrázek 37: Odchylky u experimentu s filtrem F_u

Ještě bychom se měli ujistit, že jsem neudělal něco špatně v návrhu tohoto filtru, a tak udělám ještě jeden pokus. Uzavřu smyčku rovnou přes přenos na kabinu, jak je znázorněno na obrázku 38).



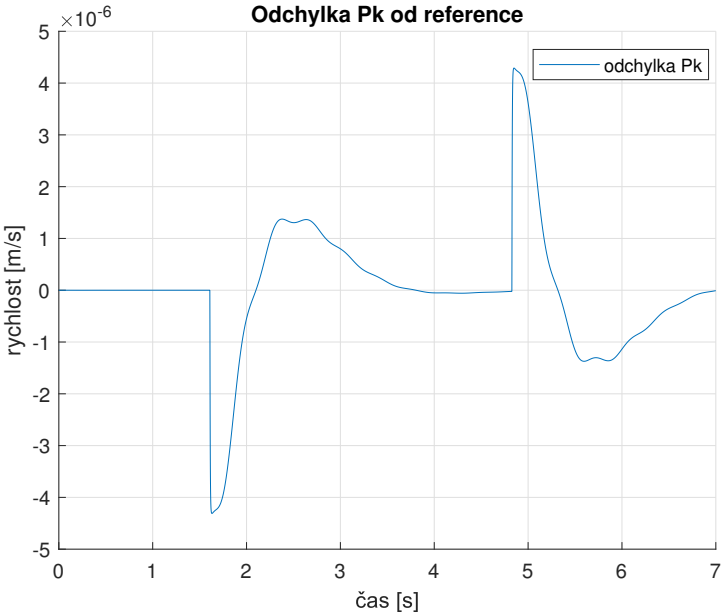
Obrázek 38: Odchylky u experimentu s filtrem F_u

Pokud se podíváme na průběh (viz obrázek 39), vypadá skutečně dokonale, ideálně sleduje referenci a tak dále. Proč tedy tuto regulaci nepoužít? O tom jsme již také mluvili, jedná se o průběh fáze a z něj vyplývající omezení na zesílení regulátoru. Výsledná šířka pásma je tak mnohem menší. Ještě to mohu objasnit z jiné strany – sice regulátoru hodně pomohu tím, že mu předpřipravím vhodný očekávatelný vstup, ale pokud přijde nějaká porucha, malá šířka pásma zabrání jejímu odregulování. Proto je lepší zůstat u verze se zpětnou vazbou uzavřenou přes motor, i když podle následujících výsledků vypadá tato možnost podmanivě.



Obrázek 39: Výsledek experimentu s kabinou ve zpětné vazbě

Ještě bychom měli zkontrolovat odchylku (viz obrázek 40). Naprosto nulová není, jsou tam nějaké odchylky od reference, ale neparné – v řádu 10^{-6}. To přisuzuji nepřesnostem (zaokrouhlení, numerická metoda...) při výpočtech Matlabu.

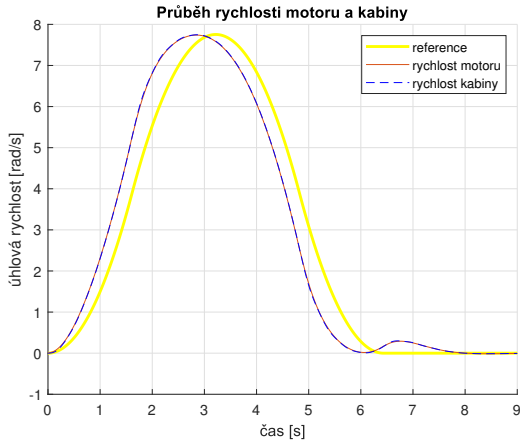


Obrázek 40: Odchylka u experimentu s kabinou ve zpětné vazbě

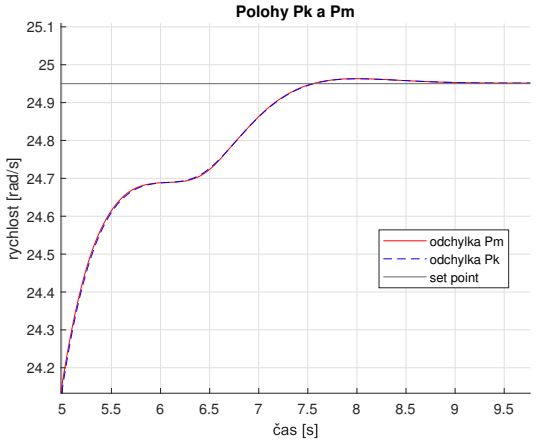
Už jsme si ukázali všechny jednodušší varianty a postupně je zavrhli z různých důvodů. Zbývá tedy spojit tyto části dohromady, jak jsme si to ukázali již na začátku této kapitoly (viz obrázek 28).

V tomto případě nám příliš nepomůže sledování regulační odchylky e, protože na vstup jsou generovány v podstatě nějaké kmity, které jdou proti přirozeným kmitům systému, takže je vyruší a zajistí hladký průběh. Zajímá nás odchylka již výsledného pohybu motoru nebo kabiny vůči požadované hodnotě, ale ani ta nám v tuto chvíli neřekne tolik protože dochází k jevu, který je zřetelný na průběhu rychlostí (viz obrázek 41). Bohužel mé předpoklady byly mylné a ani takovýto způsob, ke kterému jsem celou dobu směřoval, není správným řešením. Ačkoli to na první pohled působí, že rychlost se již dostala na správnou hodnotu, poloha (viz obrázek 42) ještě požadované polohy nenabyla a proto je zde ještě tato boule, která to dorovná. To ale není chování, které by nám vyhovovalo, proto se nad tím zkusíme ještě jednou zamyslet.

Dochází zde právě k problému způsobeným dvěma odlišnými dynamikami, které se navzájem tlučou. A nedá se tomu zamezit ani výše zmíněným způsobem, protože se vlastně stále snažíme natvrdo zkombinovat dvě dynamiky, ale v podstatě odděleně, každou jiným způsobem. Proto je potřeba na to jít ještě jinak.



Obrázek 41: Problematický průběh rychlostí



Obrázek 42: Vysvětlení průběhu rychlosti pomocí průběhu polohy

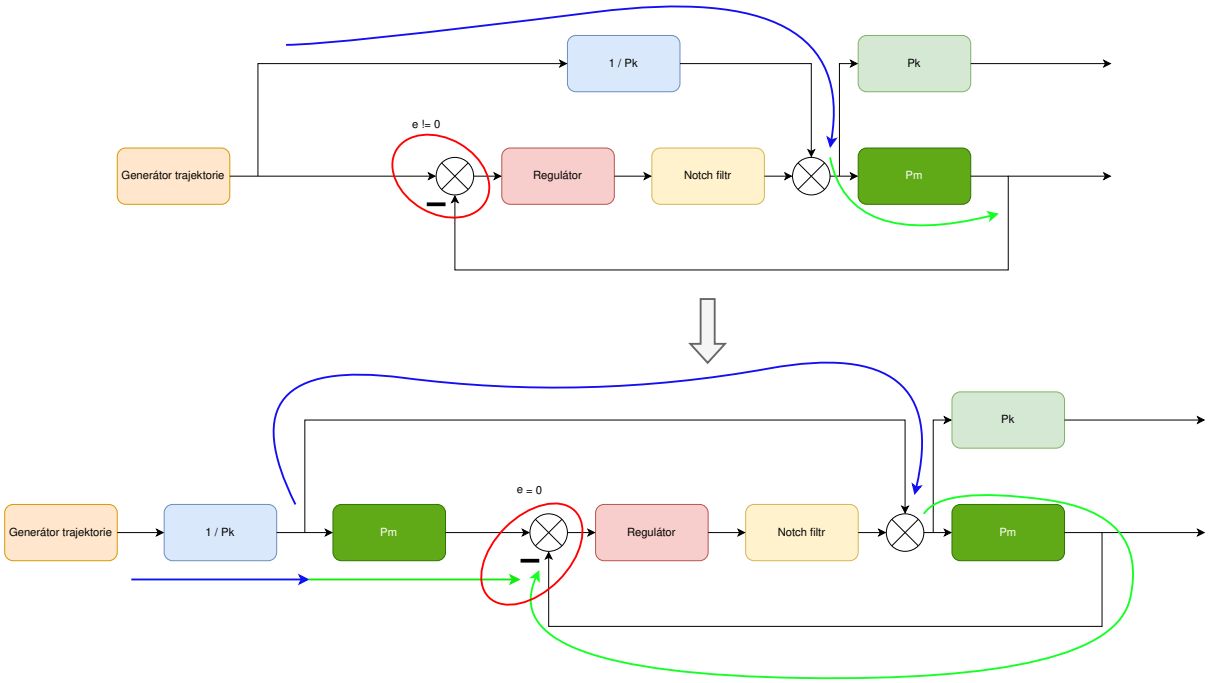
2.3 Nulová odchylka tvarovacího filtru

2.3.1 Odvození vhodného přímovazebního řízení pro dvouhmotový systém

Abych dokázal lépe vysvětlit své úvahy, budu se odkazovat na tento obrázek 43.

Jak je znázorněno v první části obrázku, snažím se dosáhnout nulové regulační odchylky – tedy, že v ideálním případě nebude muset regulátor dělat vůbec nic. To znamená, že cesta přicházející do červeně zakroužkovaného odčítacího uzlu jakožto požadovaná hodnota musí být stejná jako cesta, která se zde odečítá (tedy záporná zpětná vazba). V tom okamžiku už nebude mít regulátor s notch filtrem žádný vliv na chování (protože odchylka je nulová). To velmi zjednodušuje právě odvození toho, co bychom chtěli použít jako tvarovací filtr reference, aby se signály odečetly a my dostali odchylku $e = 0$.

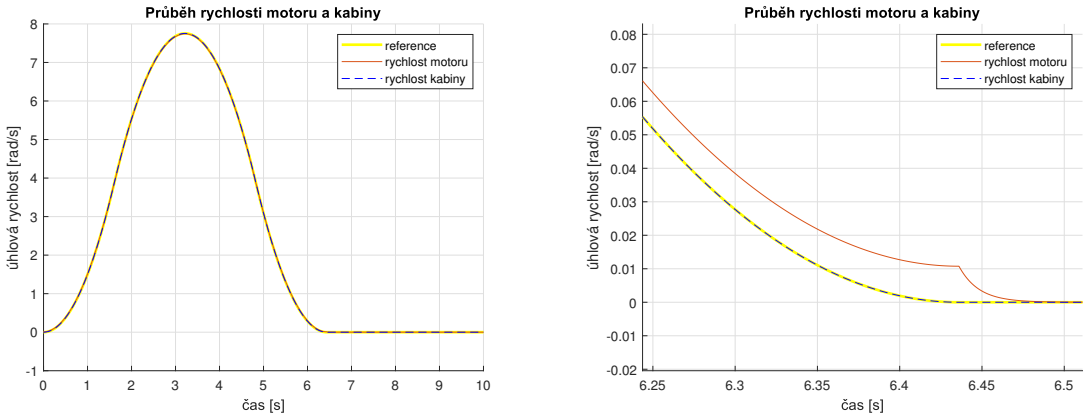
V první části obrázku je vyznačen signál, který se bude odečítat v případě, že odchylka bude nulová a regulátor s notch filtrem nebudou mít vliv. Je rozdělen na dvě části (označeno modrou a zelenou šipkou). Na druhé části je již vidět upravený tvarovací filtr, který vede na nulovou regulační odchylku. Modrá šipka značí část dopředné vazby na řízení odvozené již dříve jako inverze přenosu na rychlost kabiny. Na dorovnání signálů stačí tedy jen přenásobit přenosem na motor, což je naznačeno pomocí zelené šipky. Tímto způsobem mohu získat ideální řízení pouze pomocí přímé vazby, ale samozřejmě jen v případě, že pracuji s konkrétním linearizovaným systémem, protože potřebuji přesně znát přenosy na rychlost motoru a kabiny.



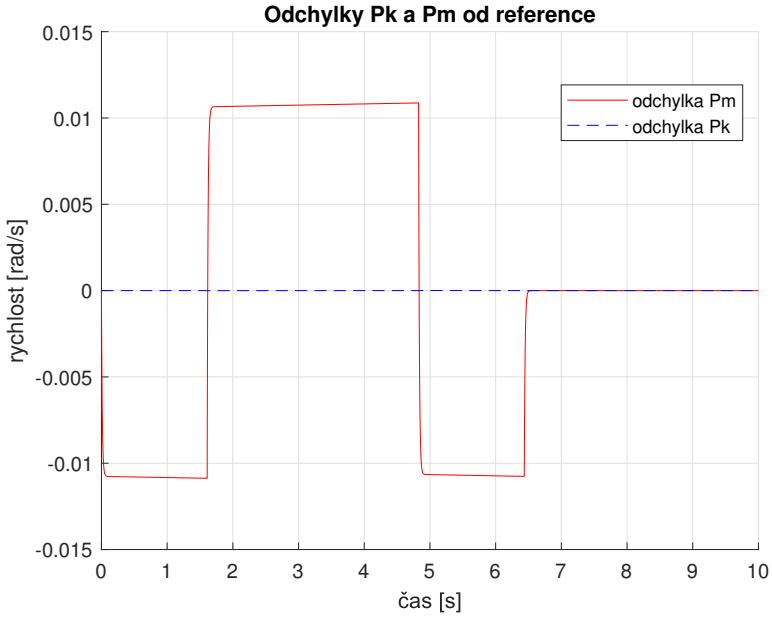
Obrázek 43: Schéma odvození přímé vazby

2.3.2 Simulace pro 5. patro

Na obrázku 44 je vidět průběh rychlostí kabiny a motoru (rychlostí proto, že řídím výtah právě skrz rychlost motoru). Na detailu si můžeme všimnout, že motor na konci trochu cuká, ale to je v pořádku – dokonce je to žádoucí, protože právě tímto cuknutím dorovná kmity vznikající na straně kabiny. Na obrázku 45 to je vidět asi ještě lépe – motor se od reference odchyluje, ale kabina ji sleduje přesně – a to je to, čeho jsme chtěli docílit.



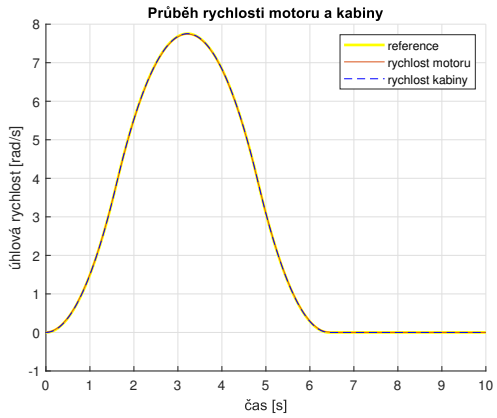
Obrázek 44: Průběh rychlostí s dopřednou vazbou a detail dojezdové části



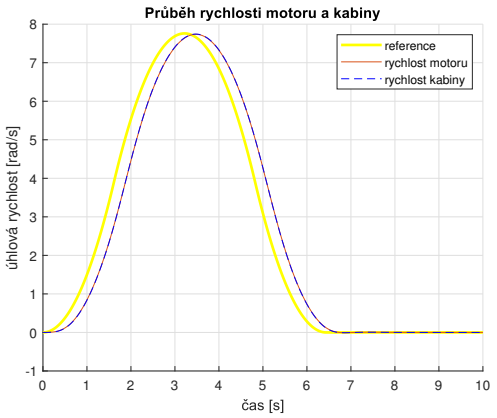
Obrázek 45: Zobrazení odchylek rychlostí kabiny a motoru od požadované hodnoty

Na těchto obrázcích je i dobře znatelné vylepšení, které dopředná vazba přinesla oproti použití obyčejného robustního regulátoru – sice nezlepšila chování, odezvu, šířku pásma ani nic podobného, ale docílili jsme okamžitého vysledování reference tím, že máme možnost *předpovídat* ideální vstup. Tento rozdíl je viditelný na obrázcích 46 a 47, kde jsou porovnány průběhy rychlostí kabiny a motoru a jejich odchylky od reference při použití navržené dopředné vazby a bez ní. Teď zcela jasně vidíme, že ačkoliv reakce regulátoru

jsou stále stejně rychlé, dokázali jsme vysledovat referenční veličinu přesně okamžitě a to na straně kabiny, o kterou ve skutečnosti jde především.

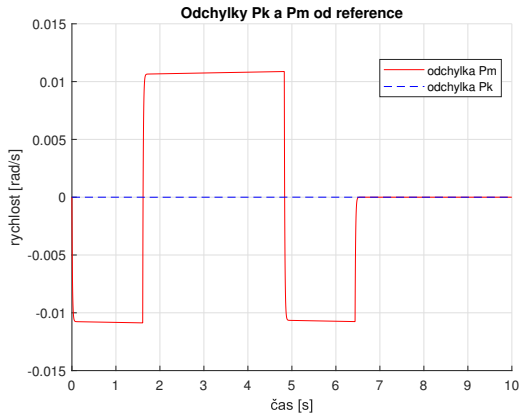


(a) S dopřednou vazbou

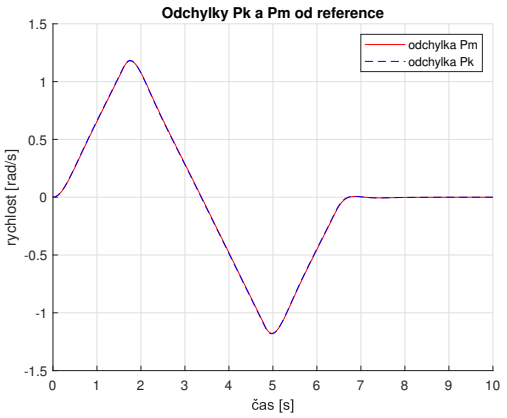


(b) Bez dopředné vazby

Obrázek 46: Porovnání rychlostí vůči referenci s dopřednou vazbou a bez



(a) S dopřednou vazbou



(b) Bez dopředné vazby

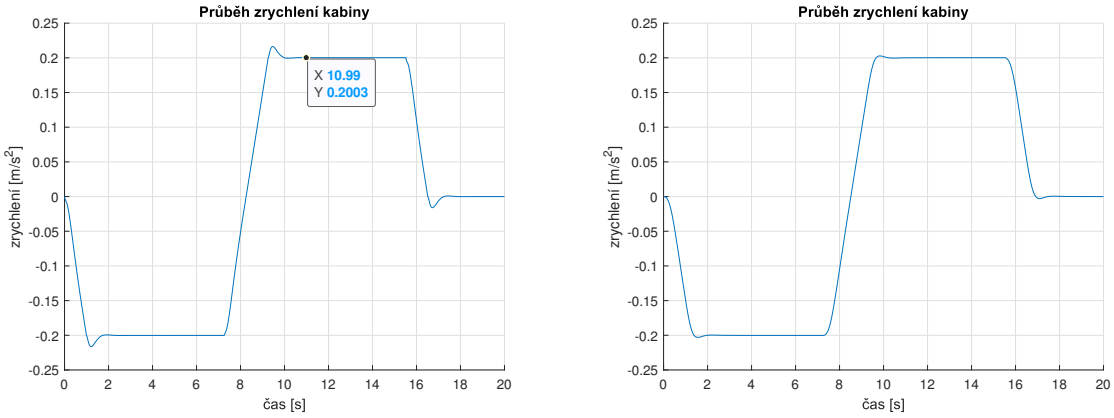
Obrázek 47: Porovnání odchylek od reference s dopřednou vazbou a bez

Je třeba si ale uvědomit, že všechny tyto testy, které mají tak dokonalé výsledky, byly prováděny na linearizovaném modelu v jednom konkrétním patře. Pokud budeme chtít použít tento druh řízení v praxi a nejprve samozřejmě simulačně testovat na nelineárním modelu, bude potřeba dopřednou vazbu měnit, protože je závislá na konkrétních parametrech přenosů na motor a kabinu, které samozřejmě platí jen v malém okolí.

Shrnutí

Co je třeba zmínit, je mírné zhoršení průběhu zrychlení kabiny oproti samotnému robustnímu regulátoru znázorněné na obrázku 48. Je to určitá daň za to, že dokážeme sledovat referenční veličinu přesně, okamžitě. Dále se ještě budu zabývat tím, jak omezit negativní důsledky tohoto zhoršení, ale prvním problémem by měla být aplikace na reálný tedy nelineární systém. Tento problém ale také způsobuje neznalost hmotnosti kabiny výtahu, která velmi ovlivňuje její chování a dynamiku. Ačkoliv regulátor je navržený robustní, tedy aby pokryl i tuto neurčitost, přímovazební filtry takto udělat nelze, jelikož používám přenosy s konkrétními hodnotami. Volil jsem proto přenosy pro prázdnou kabinu (tedy kabina je lehčí než protizávaží), protože tento stav je nejproblematičtější z hlediska regulace. Že jsem volil dobře se následně potvrdilo i v experimentech, protože ačkoliv by se zdálo logické použít přenos uprostřed, kdy kabina je stejně těžká jako závaží, tento filtr nedokáže dobře pokrýt možnost, kdy je kabina lehčí. Když je těžší, není to problém. Pokud bychom to chtěli objasnit pomocí pevných základů, můžeme si uvědomit, že lehčí kabina má větší tendenci kmitat a kmitá na vyšších frekvencích (kde tedy frekvence kmitů je ovlivněna také délkou odvinutého lana). Když je ale zátěž těžší než kabina, má systém tendenci *mávat* více s kabinou, která se rozkmitá každým protichůdným pohybem závaží. Proto je složitější při takovémto rozložení hmotností systém řídit. Z toho důvodu je pro účely filtru volena linearizace při nejnižší možné hmotnosti kabiny, nebo-li v nejhorším možném případě.

Výsledkem této části je nicméně úspěšné vyvinutí filtrů, které pomohou zajistit ideální vstup takový, který dopomůže přesnému vysledování reference. Další výhodou je, že regulátor je mnohem méně namáhaný, regulační odchylka je v tomto ideálním případě nulová, v případě nelineárního systému to tak nebude, ale bude mnohem menší, což znamená pro regulátor mnohem menší práci, menší akční zásahy.



(a) S dopřednou vazbou (b) Bez dopředné vazby

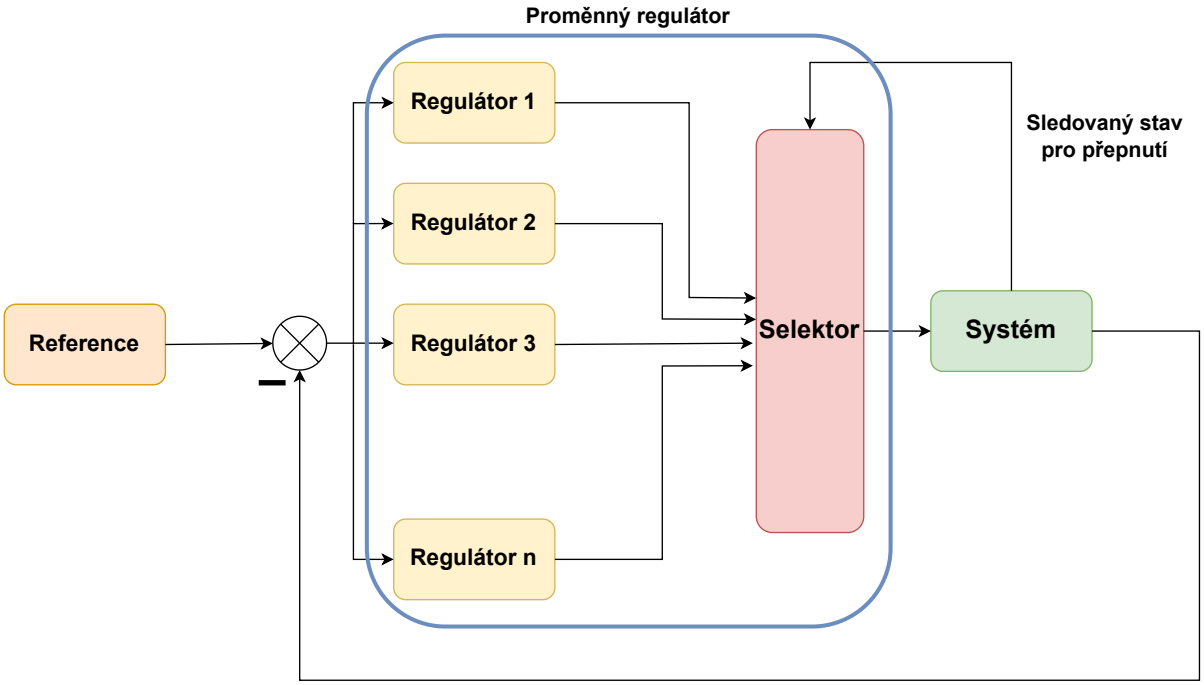
Obrázek 48: Porovnání zrychlení při přejezdu z 1. do 5. patra

3 Gain-scheduling přímovazebního řízení

3.1 princip gain-schedulingu

Aby bylo možné nasadit strategii výše vytvořeného přímovazebního řízení, je potřeba vyvinout určitý systém přepínání přímovazebních filtrů podle toho, v jakém místě se nacházíme a v jakém místě filtr přibližně odpovídá realitě. Tímto se zabývá právě takzvaný gain-scheduling, což je metoda řízení, která využívá sledovatelného stavu k přepínání regulátorů [9] případně v tomto trochu netradičním použití filtrů. Tím sledovatelným stavem je v našem případě poloha kabiny, jíž se nepřímo přes motor snažíme řídit. Na základě tohoto stavu byly již v předchozím projektu [1] vytvořeny linearizace v jednotlivých patrech a z nich následně separovány přenosy na rychlost kabiny a motoru, které jsou používány při návrhu přímovazebního řízení. To znamená, že můžeme na základě polohy kabiny přepínat tyto filtry vždy tak, aby byly co nejblíže skutečné poloze.

Standardní přístup je takový, že se navrhne n regulátorů (které nutno dodat mohou mít lepší lokální vlastnosti než jeden robustní) a ty se pak podle potřeby přepínají (viz obrázek 46). U problematiky výtahů je možné využít toho, že nepotřebujeme jiný regulátor v okamžiku, kdy je rychlost konstantní, jde nám jen o rozjezdy a brzdění. Tím pádem si můžeme dovolit použít jen dva regulátory – jeden pro rozjezd a druhý pro brzdění. Zároveň je zde ještě problematika vysledování integrační složky pro potlačení rázů při přepnutí regulátoru. V podstatě jde o odečet výstupů regulátorů a naladění integrační složky na stejnou hodnotu, aby regulátor začal generovat stejný výstup jako generoval regulátor před ním.



Obrázek 49: Gain-scheduling princip

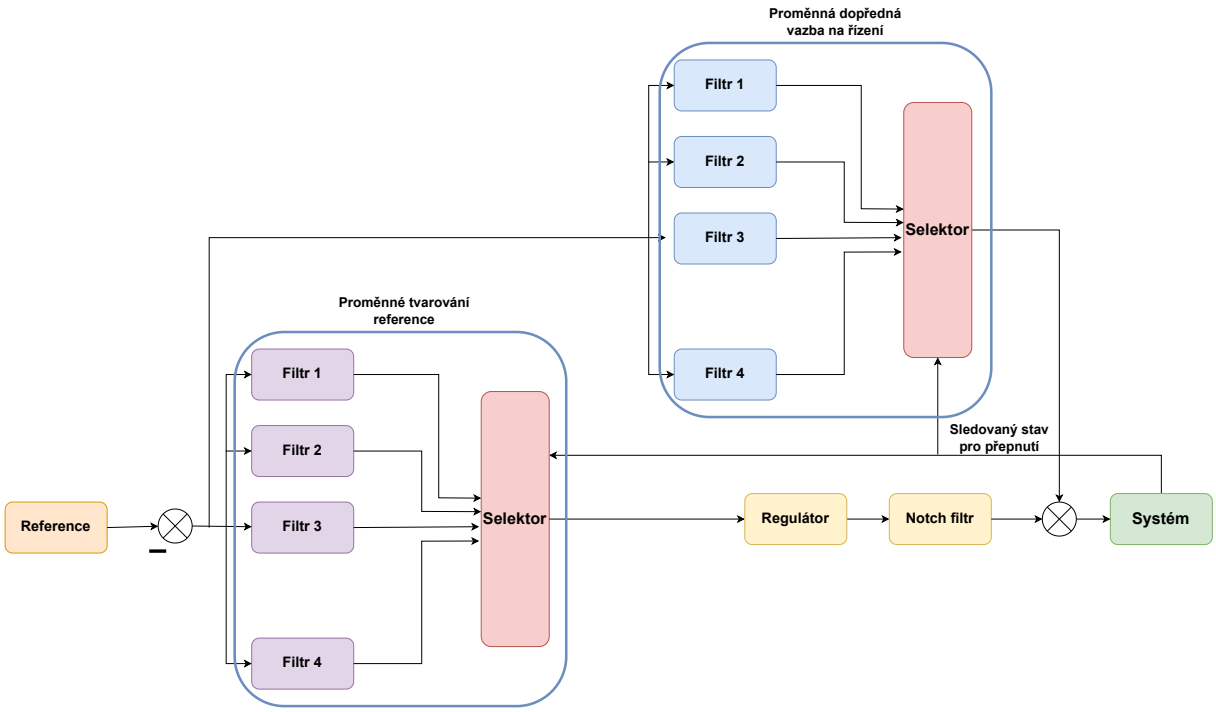
3.2 Gain-scheduling přímovazebního řízení s bezrázovým přepínáním

V našem případě to bude o něco složitější, protože nepracuji s regulátory, ale s filtry, které potřebuji měnit. Přibližné schéma je znázorněno na obrázku 46. Využity jsou filtry, které byly podrobně rozebrány v předchozí kapitole, pro všechna patra jsou automaticky vytvářeny skriptem z linearizovaných přenosů. Na první pohled na tom není nic moc složitějšího, dvojité selektování není v zásadě o nic složitější, jen je nutné přenastavit více parametrů. Problém však nastává v okamžiku, kdy se pokusíme regulátory přepínat. Při přepnutí se skokově přenastaví parametry, tudíž celkem logicky dojde ke skoku i v odezvě. Výhodou je, že řídíme rychlost, tudíž poloha se díky integraci již opět zahladí, ale například ve zrychlení je tento ráz dosti znatelný a to je určitě nežádoucí – s pasažéry by to nepříjemně trhalo nehledě na namáhání materiálů zejména pak lana. Vysledování zde však není tak jednoduché, jelikož i struktura filtru je mnohem složitější, než struktura PI případně PID regulátoru – pro představu filtry pro první patro vypadají následovně:

$$F_{U,1} = \frac{0.00303s^5 + 3.035s^4 + 248.5s^3 + 2801s^2 + 9.478 \times 10^4 s + 1.404 \times 10^5}{s^3 + 103.6s^2 + 3368s + 1.404 \times 10^5}$$

$$F_{W,1} = \frac{0.3s^9+307.5s^8+3.228\times10^4 s^7+1.469\times10^6 s^6+6.933\times10^7 s^5+1.214\times10^9 s^4+3.399\times10^{10} s^3+2.634\times10^{11} s^2+4.548\times10^{12} s+6.505\times10^{12}}{s^8+1105s^7+1.892\times10^5 s^6+1.294\times10^7 s^5+5.439\times10^8 s^4+1.792\times10^{10} s^3+2.399\times10^{11} s^2+4.548\times10^{12} s+6.505\times10^{12}}$$

přičemž v simulinku jsou modelovány ručně takovýmto způsobem (viz obrázek 31), protože jsou nekauzální a je potřeba ručně zavést jednotlivé derivace z generátoru trajektorie, jak bylo vysvětleno v příslušné kapitole.

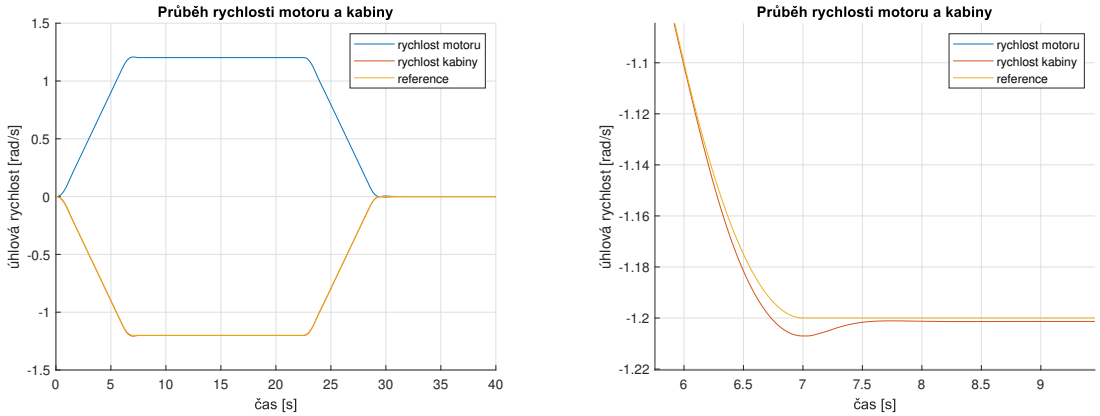


Obrázek 50: Gain-scheduling přímovazebních filtrů

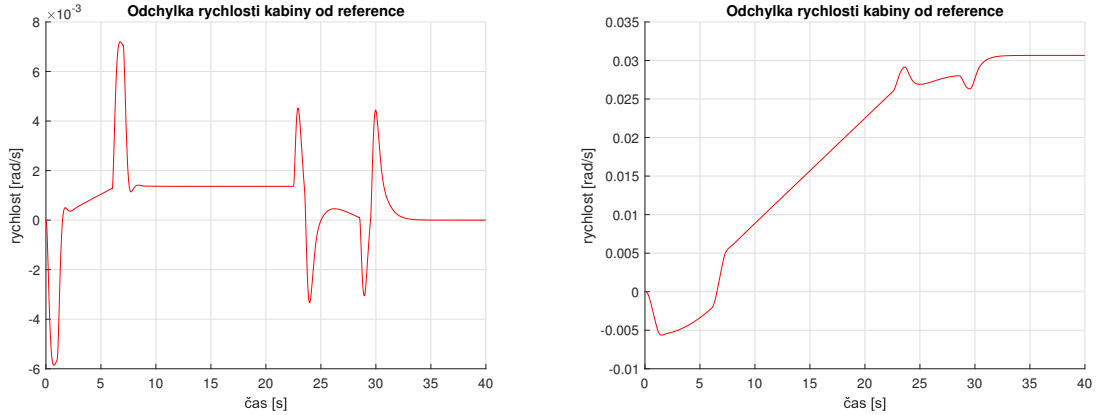
3.2.1 Přepínání v konstantní rychlosti

Jednou z možností, jak se vysledování vyhnout a přepínat i přesto bezrázově, je přepínat skutečně jen v konstantní rychlosti. Pokud je regulovaná veličina konstantní, přepnutí filtru na ní nemá vliv, protože na konstantní hodnotu dokáže každý filtr / regulátor regulovat dobře a je jedno, pro jaké patro je dělaný. Ve chvíli, kdy je hodnota konstantní, musí dávat všechny filtry konstantní a vlastně stejnou hodnotu. Nějaký malý ráz vznikne samozřejmě vlivem diskrétní simulace, ale ten je zanedbatelný.

Pokud ale řekneme, že budeme přepínat jen v konstantní rychlosti, přinášíme si tak velká omezení na parametry a zároveň kazíme dokonalé vysledování. Abych to vysvětlil, použiji simulaci, kdy výtah jel z prvního do desátého patra a přesně uprostřed došlo k přepnutí filtrů. Na obrázku 51 jde zejména pak v detailu vidět, že je tu problém při změně z lineárního průběhu rychlosti na konstantní, což můžeme interpretovat možná spíše tak, že problém nastává v okamžiku, kdy se pokoušíme přejít z konstantního zrychlení na nulové. Zde by byl zapotřebí opět nějaký jiný filtr, protože ten první již není dostatečný vzhledem k poloze, ve které je linearizovaný. Přepnutí tak sice proběhne bezrázově, ale zaplatíme za to překmitem v jiné části křivky. A to, že zde nabereme nějakou nepřesnost, ale přitom pak přepneme naprosto v klidu, má za důsledek ztrátu celkové přesnosti, což je vidět na obrázku 52 znázorňujícím odchylky poloh a rychlostí.



Obrázek 51: Ukázka přepínání bez vysledování v konstantní rychlosti – rychlosti



(a) Odchylka od požadované rychlosti (b) Odchylka od požadované polohy

Obrázek 52: Ukázka přepínání bez vysledování v konstantní rychlosti – rychlosti

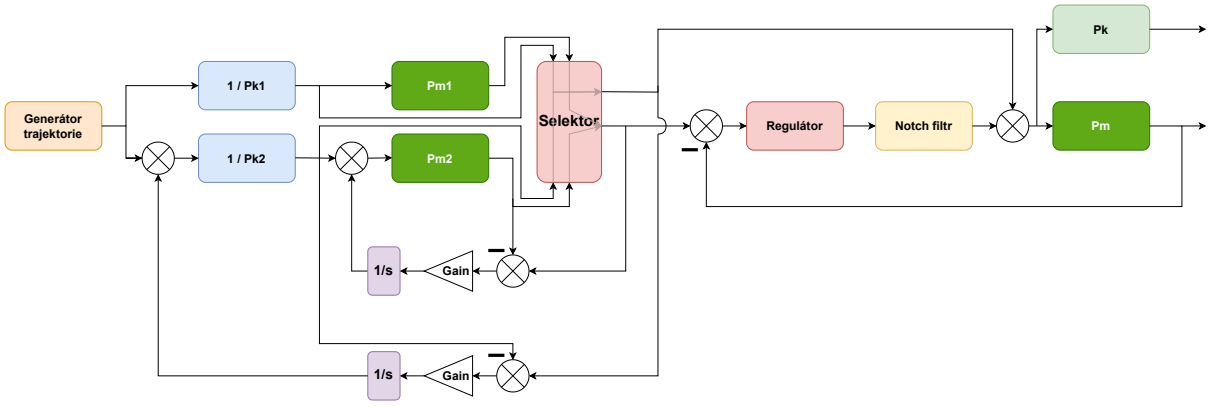
Problém se zdá být v tom, že ačkoliv rychlost doregulujeme na správnou požadovanou hodnotu, poloha *neskončí tam, kde by měla.* To je jednoduše způsobené právě tím posuvem kvůli brždění a zrychlování s nesprávně generovanou požadovanou hodnotou kvůli nevhodnému filtru. Můžeme si všimnout, že odchylka od požadované koncové polohy je malá, zhruba 3 *cm* v tom nejhorším případě – přejezdu mezi všemi deseti patry. To by v reálu až tak nevadilo, protože stejně dochází k jistým poruchám v šachtě s výtahem, takže takhle ideálně jako v simulaci by nikdy kabina nezastavila. Proto je možné přidat kaskádní regulaci na polohu kabiny v každém patře s tím, že by se vnější polohová smyčka spouštěla až v okamžiku, kdy kabina je opravdu poblíž dojezdové hodnoty, protože měřit polohu podél celé trajektorie je jak finančně tak technicky náročné. Proto se to většinou řeší spíš tak, že kabina zabrzdí do velmi nízké rychlosti ještě o chvíli dříve, než dojede na požadovanou hodnotu a touto malou rychlostí se pohybuje dál, dokud nenarazí na koncový spínač, který je instalován v každém patře. Tím je, kromě snadného řešení přesného dojezdu, zajištěna rovněž jakási *rekalibrace* kabiny.

Hlavní potíž tkví ale v tom, že v okamžiku, kdy přecházíme mezi lineární a konstantní částí trajektorie rychlosti, máme nastavené filtry, které již dávno neodpovídají realitě, protože kabina mezitím přejela do úplně jiné polohy. Takže buď by musely být parametry nastavené tak, aby kabina zrychlila na maximální rychlost velmi blízko počátečnímu patru, nebo to bude chtít přepnout vícekrát, konkrétně před každým zrychlováním či brzděním, tedy čtyřikrát. To ale již nebude přepnutí za konstantní rychlosti, takže budeme potřebovat vyvinout nějaké vysledování pro bezrázové přepínání během lineárního průběhu rychlosti.

3.2.2 Vysledování pomocí integrace odchylky

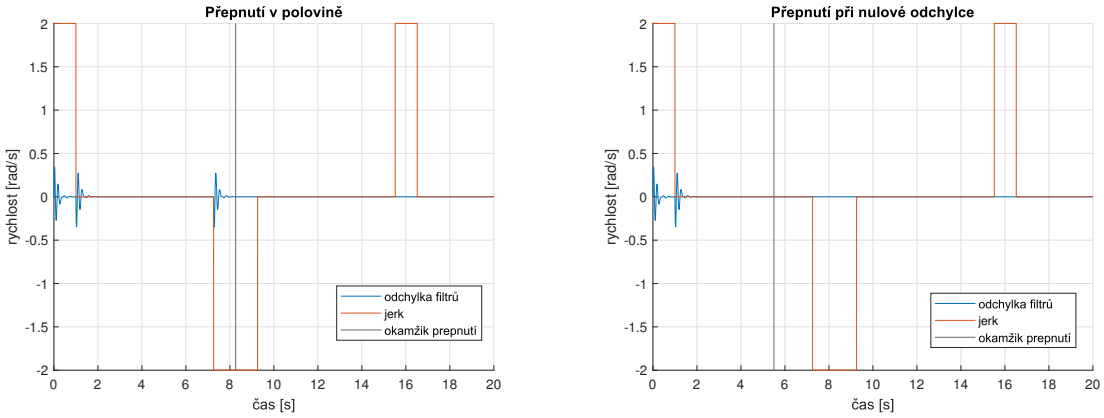
Došli jsme k závěru, že to bude chtít nějakým způsobem vysledovat stav předchozího filtru takovým způsobem, aby druhý filtr navázal ze stejné hodnoty, avšak aby se nezměnil jeho průběh. To je jasný úkol pro integrátor.

Budu uvažovat, že máme jen dva filtry – pro rozjezd a dojezd. Mezi nimi se přepne přesně v polovině dráhy (za chvíli vysvětlím, že možná bude lepší jiná varianta). To znamená, že stačí vysledovávat první filtr tím druhým, v okamžiku, kdy dojde k přepnutí, chceme, aby byla odchylka nulová. To zajistím vytažením signálu až za selektorem (viz schéma 53). Vůbec bude lepší toto všechno vysvětlit na obrázku:



Obrázek 53: Vysledování pomocí integrace odchylky

Integrátory dopomohou tomu, aby došlo k nenásilnému a postupnému přeladění filtru tak, aby po chvíli sám generoval stejnou hodnotu, jako první filtr. V podstatě jsem vytvořil ještě vnořenou lokální zpětnovazební smyčku s integračním regulátorem, která má za úkol *regulovat odchylku od předchozího filtru na nulu*. V tom okamžiku je možné provést přepnutí, při kterém nedojde k žádným rázům. Je ale dobré přepnutí provést hned, jak je odchylka dvou filtrů $e = 0 \pm \epsilon$, protože tato odchylka se zvětší pokaždé, co dojde ke změně jerku a bude nutné ji znovu nechat naintegrovat, než dojde k přepnutí. To je vidět na následujícím obrázku 54:



Obrázek 54: Ukázka přepínání v různých okamžicích

To že je odchylka nenulová na začátku nás vůbec netrápí, protože ještě nechceme přepínat. Problémy s tímto řešením, které na první pohled působí velmi dobře, jsou však následující:

- Naladit správné zesílení tak, aby fungovalo dobře pro všechny kombinace filtrů nemusí být snadné
- Potřebujeme vhodné parametry, aby se jerk neměnil příliš často a stačilo dojít k ustálení odchylky na nule
- Nutnost zkombinovat vhodný čas k přepnutí u obou filtrů (každého odchylka se pravděpodobně ustálí jinak rychle)
- Velmi složitá logika přepínacích okamžiků

I přes všechna tato negativa mě zajímalo, jak se bude toto přepínání chovat, pokud je použiji na nelineární systém, který už alespoň o něco více odpovídá realitě. A musím přiznat, že jsem byl velmi mile překvapený. Tento způsob velmi dobře zajistí bezrázové přepnutí filtrů, jen je třeba dát si pozor na pár věcí a jednou z nich jsou rozhodně přepínací časy generátoru trajektorie – tedy okamžiky, kdy se mění jerk. V těchto okamžicích, jak bylo ukázáno před chvílí na obrázku 54, dochází k opětovnému vybuzení přenosů filtrů (jednoduše řečeno při změně v trajektorii musí filtry začít něco dělat), tím pádem se opět zvětší jejich vzájemná odchylka vlivem odlišné dynamiky a je třeba čekat na vysledování, které zajišťuje integrátor. Co se týče nastavení vysledovací konstanty, musí být určena pro každý přenos – tedy pro každé patro – jiná, právě kvůli odlišné dynamice. To ale není takový problém, je to stejné jako při obyčejném gain-schedulingu, kde je potřeba naladit

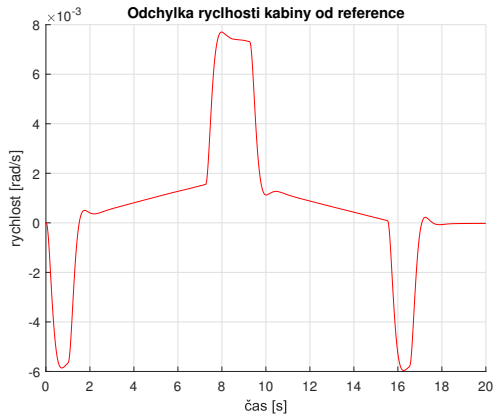
stejný počet regulátorů. Vlastně je to tedy ještě jednodušší, protože zde je parametr k ladění jen jeden a jde tedy snadno určit experimentálně.

3.2.3 Aplikace na nelineární model

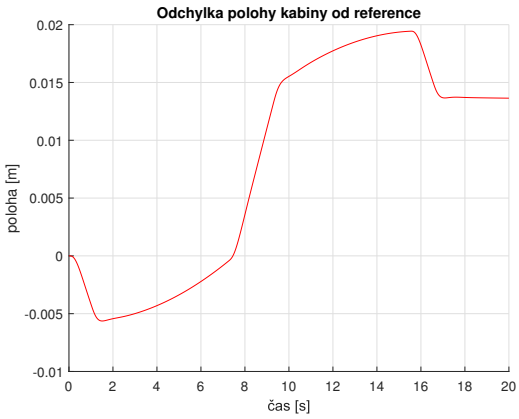
Jak bylo zmíněno v závěru předchozí části, prvním problémem je vhodné určení času přepnutí. Velký vliv na to má průběh jerku z důvodů, které již byly párkrát řečeny (ve zkratce při každé změně jerku dojde k nějaké změně trajektorie a tudíž je třeba znovu vysledovat druhý filtr). Jak vypadá průběh jerku je vidět na obrázku 18. Může být trochu odlišný v závislosti na zadaných parametrech, ale důležité je, aby alespoň v nějakém momentu byl dostatečně dlouho konstantní, abychom mohli provést přepnutí. To samozřejmě záleží i na kvalitě nastavení vysledovacích konstant.

Tento úkon nalezení vhodného okamžiku je buďto možné nějak logicky naprogramovat, vymyslet způsob ověřování, že odchylka je již dostatečně nízká a tak podobně, ale má to jeden háček. Když jsem vytvořil takovýto program, který v sobě zahrnoval i informaci o trvání konstantního jerku, simulaci to velmi zpomalilo. Je to tím, že se v každém časovém okamžiku musí vykonat poměrně dost výpočtů navíc, ověřit několik podmínek, zpracovat více dat z modelu... Zkrátka efektivita řízení se velmi snížila a bojím se, že by tomu bylo tak i u reálného řídícího systému. Je ale možné, že vzorkovací perioda by byla dostatečně velká, aby se v ní všechny výpočty provedly a že je toto spíše problémem simulací v Simulinku. Nakonec jsem to vyřešil tak, že přepínám podle typu trajektorie (což je zmiňováno až dále) kousek před změnou jerku, která vede na lineární průběh zrychlení. Jelikož jsou k tomu potřeba myšlenky, které jsou uvedeny až dále v práci, tento způsob přepínání vysvětlím později.

Nejprve jsem provedl simulaci pro přejezd z 1. do 5. patra. Zde se zdá, že vše funguje poměrně dobře, odchylka je skutečně malá, takže vidíme, že referenční veličinu sledujeme takřka okamžitě (viz obrázek 55). Je tam patrná nepřesnost v dojezdu polohy, ale ta, jak jsem zjistil, je dána numerickými výpočty během převodu trajektorie generované pro motor na odpovídající trajektorii pro kabinu. Potřebuji totiž generovat trajektorii motoru, protože skrz něj probíhá řízení, konverze (aby kabina měla správnou pozici) je daná samotným modelem a je tedy obsažená v něm (dalo by se říci, že proběhne samovolně, jako ve skutečnosti). Dalším faktorem, který toto ovlivňuje, je regulace rychlosti s absencí polohové smyčky, která by dorovnávala odchylky. Na obrázku 56 je vidět porovnání průběhu zrychlení kabiny při tomto řízení a jen za použití robustního regulátoru. Překmit se, jak vidíme, zvýšil, ale zároveň odpovídá cca jen 8%, což není příliš. Neměl by ani vadit cestujícím, protože je hladký. Je způsoben tím, že předgenerovávám referenci a filtry podle přenosů odpovídajícím 1. / 5. patru, ale přitom se pohybuji v mnohem větším okolí.

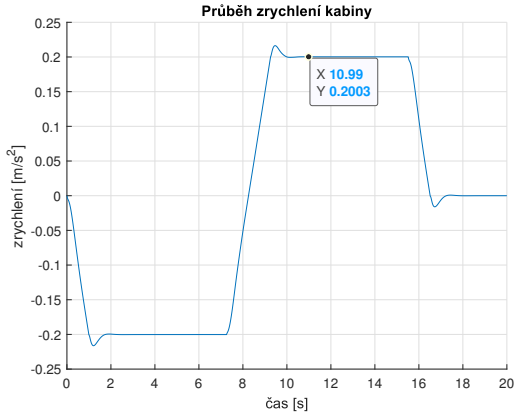


(a) Odchylka od požadované rychlosti

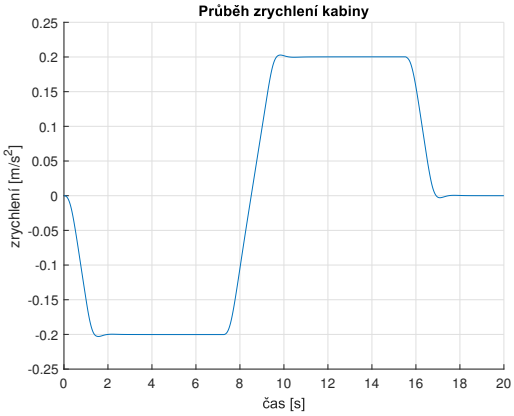


(b) Odchylka od požadované polohy

Obrázek 55: Odchylky od reference při přejezdu z 1. do 5. patra



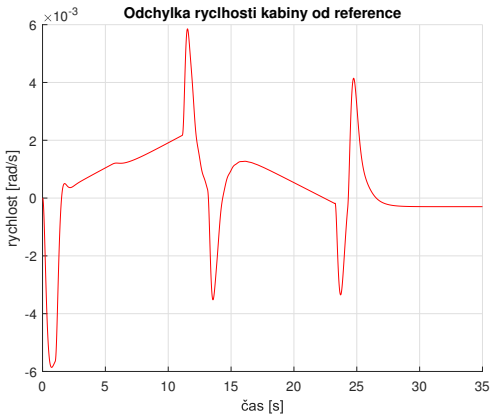
(a) S dopřednou vazbou



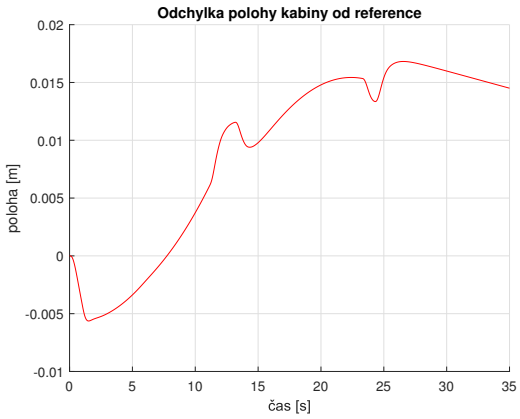
(b) Bez dopředné vazby

Obrázek 56: Porovnání zrychlení při přejezdu z 1. do 5. patra

Další simulaci jsem provedl po celém rozsahu výtahové šachty, tedy mezi 1. a 10. patrem. Zatím stále nechávám parametry takové, aby trajektorie rychlosti uprostřed nenabývala konstantní rychlosti. Na obrázku 57 jsou opět znázorněné odchylky, které vypadají velmi podobně a stále se skutečně držíme poměrně přesně trajektorie. Problém je vidět, když se podíváme na křivku zrychlení. Opět je samozřejmě horší než u samotného robustního regulátoru (viz obrázek 76), ale to je vlastně pořád víceméně stejné. Ten zásadní problém je právě v průběhu těchto překmitů. Znázorněno je to na obrázku 59. Když se podíváme na první a třetí překmit (tedy ty blízko rozjezdu a dojezdu) vidíme, že jsou hladké. Ale ten druhý, uprostřed, má již velmi nepěkný průběh způsobený – oproti předchozí simulaci pouze mezi pěti patry (viz obrázek 56) – tím, že probíhá někde kolem pátého patra a ideální vstup je generovaný filtrem určeným pro 1. / 10. patro, což je už přílišný rozdíl. Vidíme tedy, že bude zřejmě nutné přepínat mezi více filtry. Je to bohužel trochu jiné, než u klasického přepínání zpětnovazebních regulátorů, tam by to stačilo, zde ale upravujeme vstup, upravujeme trajektorii tak, aby byla ideální, což znamená, že každá změna v trajektorii se sem propíše. A protože regulujeme pomocí rychlosti, propíše se sem každá změna rychlosti mezi lineárním a konstantním průběhem.

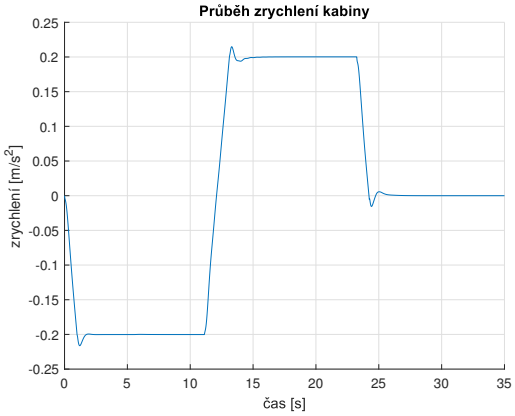


(a) Odchylka od požadované rychlosti

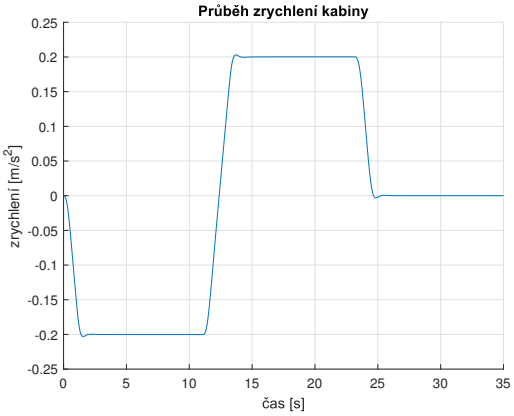


(b) Odchylka od požadované polohy

Obrázek 57: Odchylky od reference při přejezdu z 1. do 10. patra

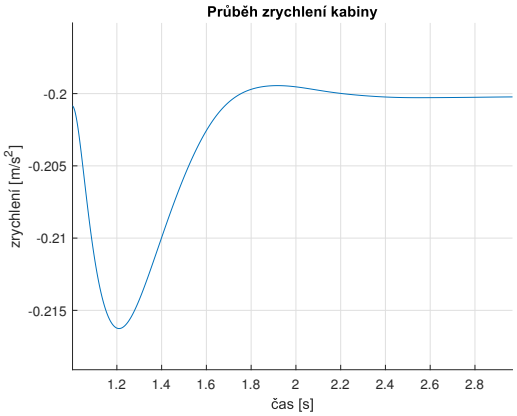


(a) S dopřednou vazbou

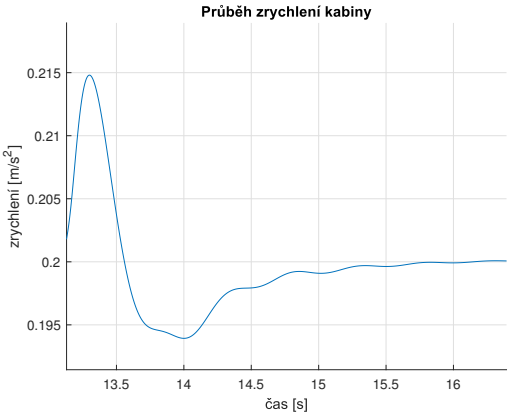


(b) Bez dopředné vazby

Obrázek 58: Porovnání zrychlení při přejezdu z 1. do 10. patra



(a) Blízko přenosu pro filtry



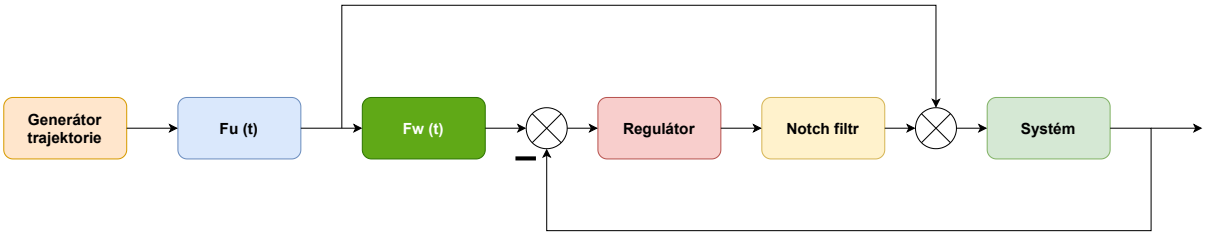
(b) Daleko od přenosu pro filtry

Obrázek 59: Porovnání překmitů ve zrychlení

3.2.4 Interpolace přepínaných parametrů

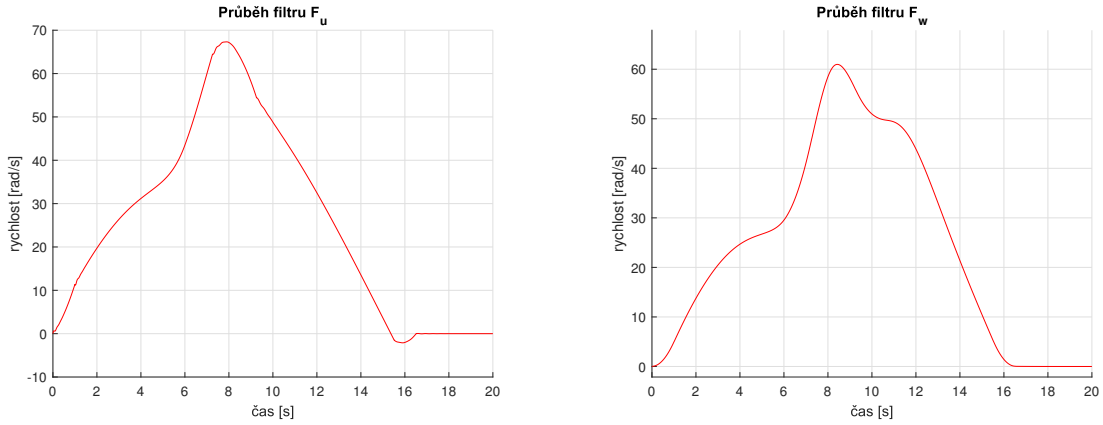
V úvodu této kapitoly byla zmíněna práce [9], ve které je kromě podstaty gain- schedulingu rozebírána metoda proložení parametrů, které přepínáme, nějakou spojitou křivkou. Podrobnosti toho, jak to lze provést, jsou uvedeny v práci, v Simulinku ale existuje bloček *LookUp Table*, který něco podobného dělá. Zadávají se mu přepínací body a hodnoty parametrů v těchto bodech a on je schopný je proložit křivkou (interpolovat) pomocí zadané metody (lineárně, kubicky atd.). Rozhodl jsem se tento přístup vyzkoušet ještě před tím, než půjdu dělat vícenásobné vysledování filtrů, abych znal porovnání a mohl případně včas vybrat vhodnější metodu.

Pokusil jsem se tedy mezi filtry nepřepínat přímo, ale měnit jejich hodnoty spojitě, což mě vrací opět k takovémuto schématu (viz obrázek 60) s tím, že filtry jsou vlastně spojitými funkcemi. Tím se vyřeší problém rázů.



Obrázek 60: Časově proměnné filtry

Když se ale podíváme na to, jak jednotlivé filtry vypadají, zjistíme, že problém tu je. Jak jde vidět na obrázku 61, filtry jsou vlivem proložení křivkou velmi zvlněné. U filtru F_U by to ani nevadilo, ten zkrátka generuje ideální vstup systému, to by klidně mohlo být, ale tvarování reference je poměrně nesmyslné. Když to takto zkusím spustit, zjistím, že systém skutečně (nutno říct velmi přesně) kopíruje křivku filtru F_W, což je naprosto logické, když mu ji dáváme jako referenci.



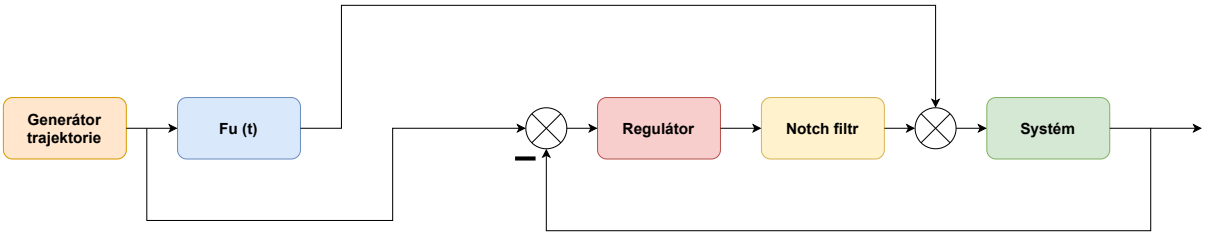
(a) Filtr na řízení

(b) Tvarování reference

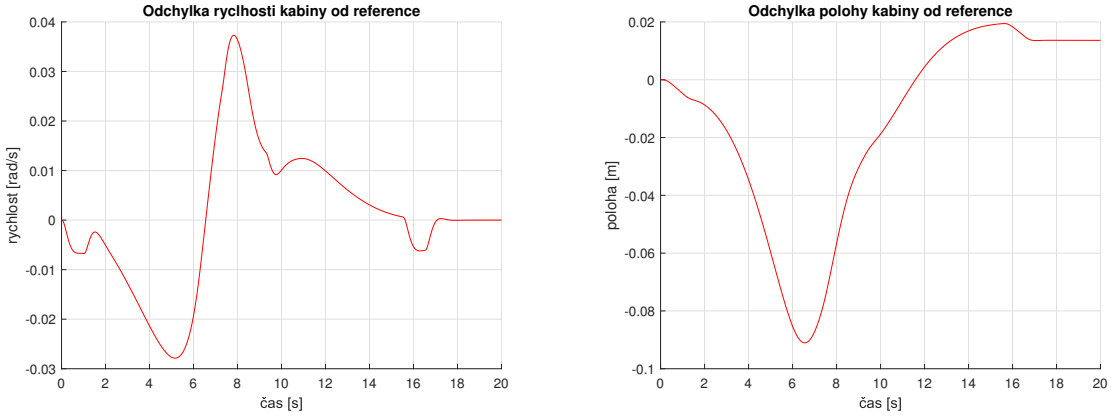
Obrázek 61: Porovnání překmitů ve zrychlení

To mě vedlo k nápadu dát takovémuto systému dokonalou referenci, tu, kterou chci, aby sledoval (viz obrázek 62), což je ale na jednu stranu nesmysl, protože tím vyřazuji problematiku, kterou jsem se zabýval v kapitole o přímovazebním řízení a to problém

s dvěma dynamikami motoru a kabiny, které musím obě zohlednit. Nicméně jsem to vyzkoušel a dopředná vazba na řízení je sama o sobě poměrně dobrá, protože výsledky (viz obrázek 63) jsou relativně uspokojivé a podobné těm, kterých jsem dosáhl selektováním dvou filtrů a jejich vysledováním, místy horší (viz obrázek 64). Rozhodně to ale není vhodná volba a interpolaci parametrů tímto zavrhuji (zkoušel jsem různé varianty využití bločku *LookUp Table*, proložení kubickou křivkou i lineární, různé numerické metody, ale nic není uspokojivé. Ke všemu při využití tohoto bločku by byl problém s manuálním vysledováním, takže jej dále již používat nebudu).



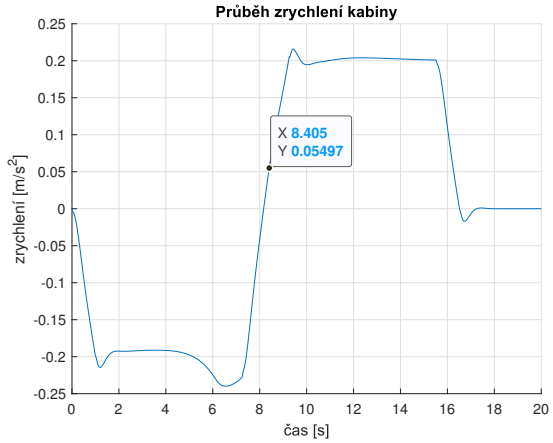
Obrázek 62: Časově proměnný filtr



(a) Odchylka rychlosti od reference

(b) Odchylka polohy od reference

Obrázek 63: Výsledky při použití interpolace parametrů filtrů

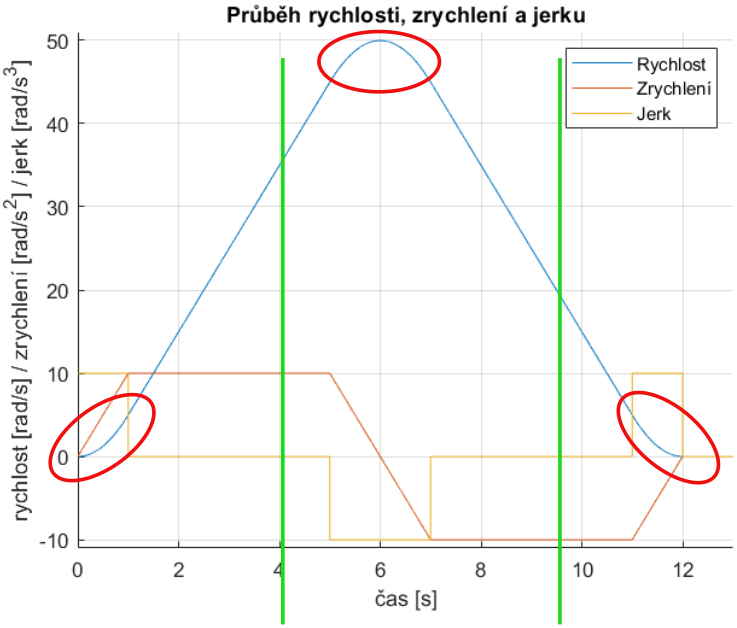


Obrázek 64: Průběh zrychlení při použití interpolace

3.2.5 Proč vícenásobné přepínání

Nejprve je třeba si rozmyslet, kolikrát skutečně budeme chtít filtr přepnout. Určitě to nebude v každém patře, protože to je zbytečné a ani by nedošlo tak rychle k vysledování. Takže by s tím bylo spíše více problémů než úspěchu. Přepínání ale bude určitě záviset na průběhu křivky rychlosti. Pokud celou dobu bude docházet jen ke zrychlování či zpomalování (viz obrázek 65), budou přepínací časy pouze 2 (vyznačeno zeleně) a tím pádem tři filtry (vyznačeno červeně) pro počáteční, koncové patro a patro mezi nimi. Pokud ale budou parametry generátoru trajektorie a zároveň rozdíl pater vycházet tak, že bude uprostřed nějaká konstantní část průběhu rychlosti, tak budou přepínací časy 4. Tady bych se vrátil opět ke kapitole, kde jsem řešil přepínání v konstantní rychlosti (obrázek 51), které je samo o sobě bezrázové. Problém zde byl však v tom, že *nájezd* a *odjezd* z konstantní rychlosti – tedy časy, kdy má rychlost průběh polynomiální a potřebuje vhodnou regulaci – byly již příliš vzdálené patru, ze kterého se bral přenos pro generování přímovazebních filtrů. Docházelo tak k odchylkám a ne příliš vhodnému chování. V tomto případě tedy budeme chtít přepnout před tímto *nájezdem*, *odjezdem* v konstantní rychlosti a za *odjezdem* před zabrzděním v koncové poloze (viz obrázek 66).

Je třeba si uvědomit, že typ trajektorie nezáleží jen na tom, jaké zvolíme parametry, ale také na tom, kolik pater má výtah ujet. Zkrátka když nejede dostatečně daleko, nestačí se na maximální rychlost vůbec rozjet. Záleží to tedy na poměrně mnoha faktorech na to, abychom vynášeli nějaké konkrétní soudy. Opět v praxi je to trochu jiné – parametry pro generátor trajektorie přesně známe, jedinou neznámou je pro nás počáteční a koncová výška patra.

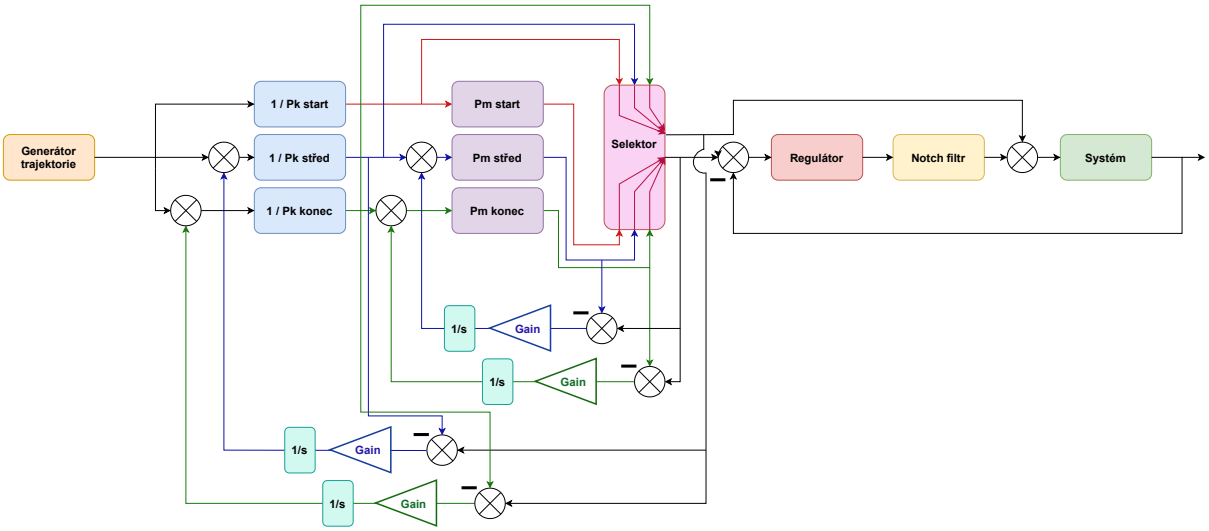


Obrázek 65: 1. průběh rychlosti s přepínacími časy a problematickými částmi pro filtry

Obrázek 66: 2. průběh rychlosti s přepínacími časy a problematickými částmi pro filtry

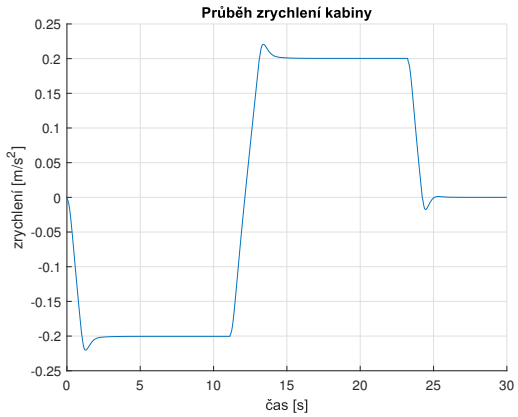
3.2.6 Dvojité přepínání pro specifický typ trajektorie

Jak již bylo uvedeno, tento způsob přepínání bude použit pro specifický typ trajektorie rychlosti (viz obrázek 65). Schéma provedení tohoto vysledování je znázorněno na obrázku 67 Dochází tedy nejprve k vysledování a přepnutí filtru ve středním patře a následně k vysledování a přepnutí na filtr odpovídající koncovému patru.

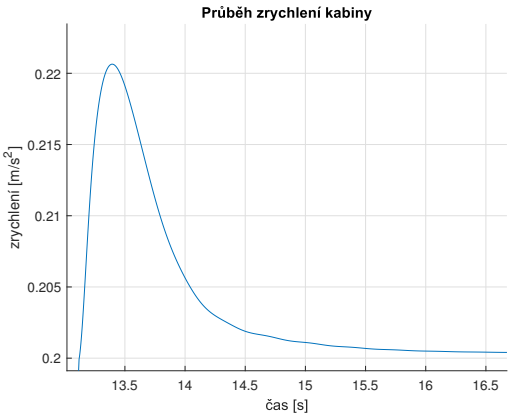


Obrázek 67: Schéma pro vysledování 3 filtrů

Výsledky jsou již uspokojivé a můžeme si všimnout zlepšení v porovnání s použitím pouze dvou filtrů (viz obrázek 59). Průběh překmitu ve zrychlení v poloze cca odpovídající střednímu patru je již rovněž hladký, což jde vidět na detailu obrázku 68.



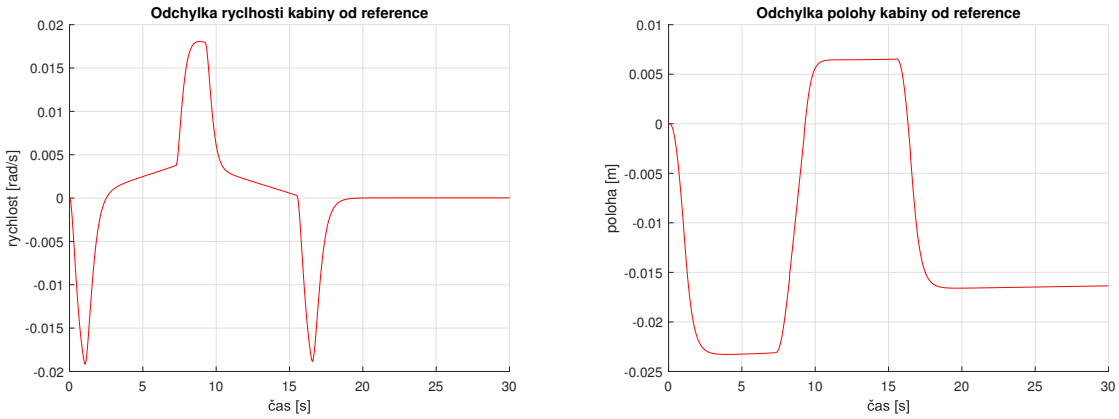
(a) Kompletní záznam



(b) Detail

Obrázek 68: Výsledky při použití tří filtrů

Dále si můžeme všimnout i zlepšení odchylky od reference (viz obrázek 69) oproti předchozím variantám, protože přidání filtru pomohlo přesnějšímu vysledování rychlosti a tím pádem je i odchylka polohy kabiny od referenční hodnoty nižší než předtím.

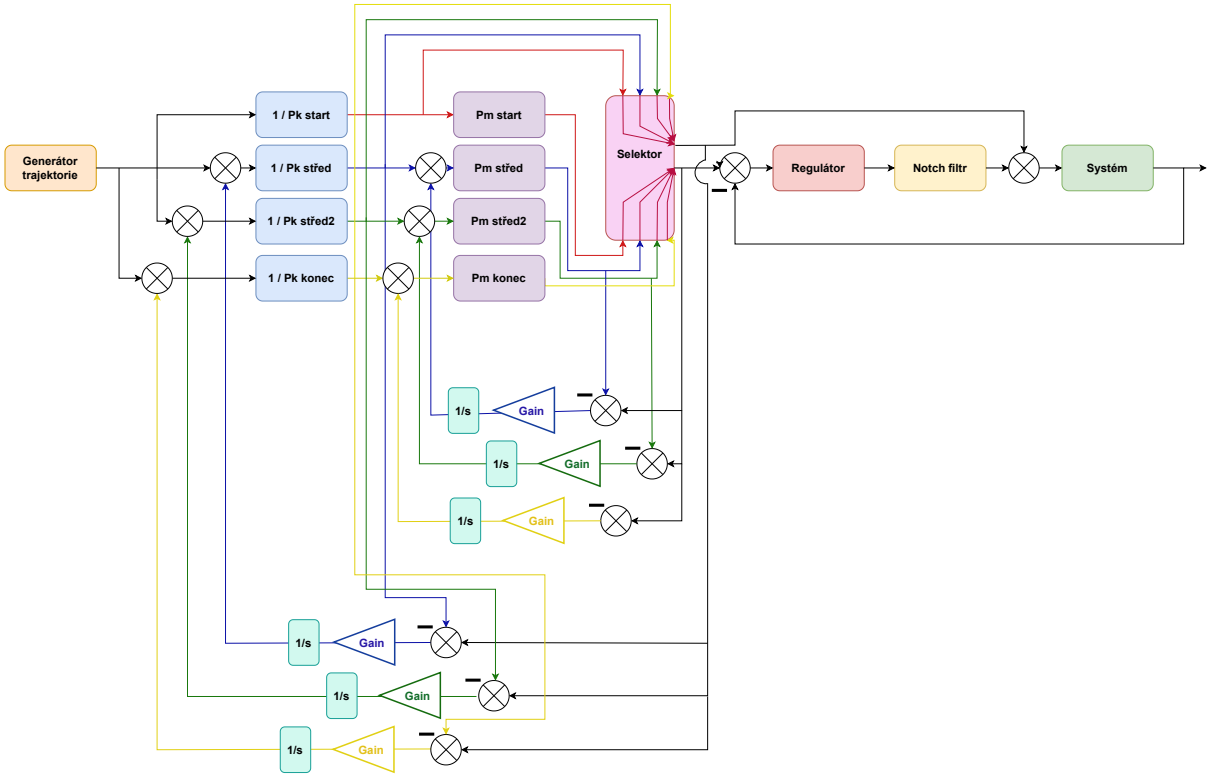


(a) Odchylka rychlosti (b) Odchylka polohy

Obrázek 69: Výsledky při použití tří filtrů

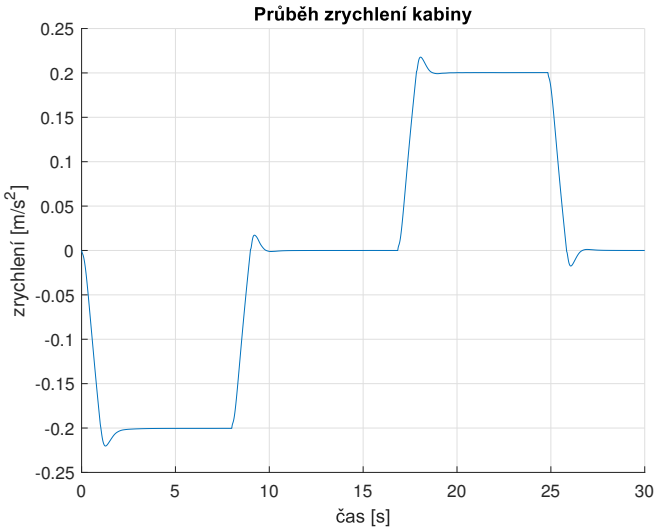
3.2.7 Trojité přepínání pro specifický typ trajektorie

Tento případ je takřka ekvivalentní s předchozím, jen platím pro jiný typ trajektorie (viz obrázek 66).



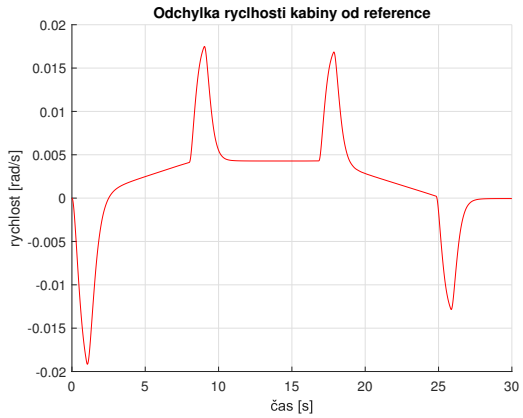
Obrázek 70: Schéma pro vysledování 4 filtrů

Na obrázku 71 je vidět průběh zrychlení kabiny jedoucí z 1. patra do 10., kde se filtry přepínají ještě pro mezipatra 3 a 5 s tomu odpovídajícími časy 7, 15 a 22 s. Křivka zrychlení je stále hladká, jediný problém, který stále setrvává v porovnání s robustním regulátorem, je vlastně ten překmit sám o sobě (spíše jeho zvětšení).

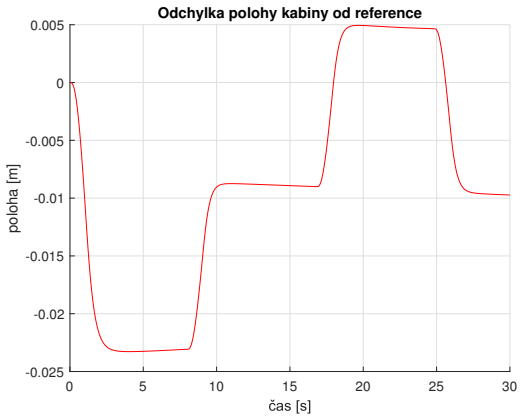


Obrázek 71: Výsledky při použití čtyř filtrů

Zde bych rovněž rád poukázal na odchylky (viz obrázek 72), které dokazují zlepšení i svým celkově přirozenějším a hladčím průběhem, než tomu bylo při nesprávném nastavení filtrů, kdy pak průběh polohy kabiny okolo požadované hodnoty trochu *nesourodě poletoval*.



(a) Odchylka rychlosti

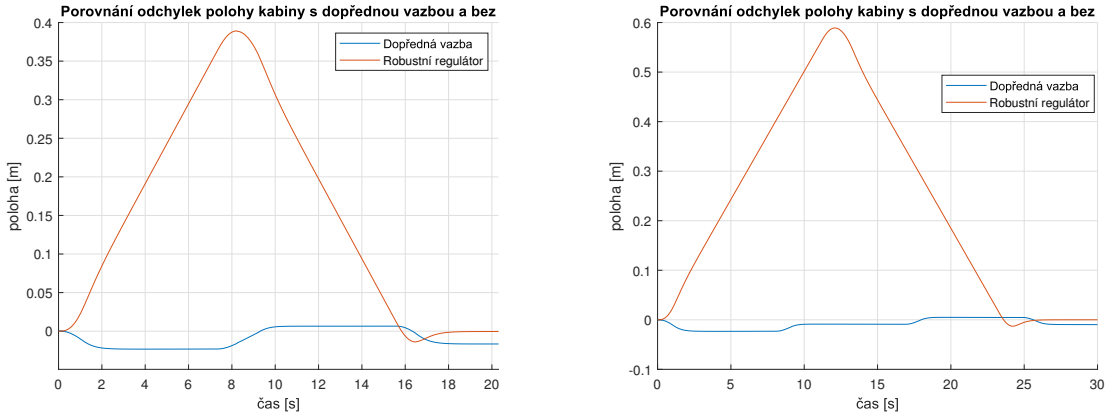


(b) Odchylka polohy

Obrázek 72: Výsledky při použití tří filtrů

Shrnutí

V tento okamžik by bylo vhodné shrnout dosažené výsledky a ukázat, že dopředná vazba skutečně k něčemu byla, že výsledky jsou uspokojivé, a že nedošlo jen ke zhoršení. Nejdůležitějším aspektem, o kterém jsme zde již mluvili, je určitě dosažení takřka přesného sledování reference. Slovem přesný myslím zejména okamžité vysledování bez prodlevy. K odstranění prodlevy oproti robustnímu regulátoru dochází díky předpovídání ideálního vstupu za pomoci generátoru trajektorie a právě dopředných filtrů. Toho bylo alespoň v simulačním světě dosaženo takřka dokonale, jak jde vidět na obrázku 73. Dokonce ani se zvyšující se délkou přejezdu se odchylka nezvětšuje, jak tomu je u robustního regulátoru.



(a) Přejezd z 1. do 5. patra

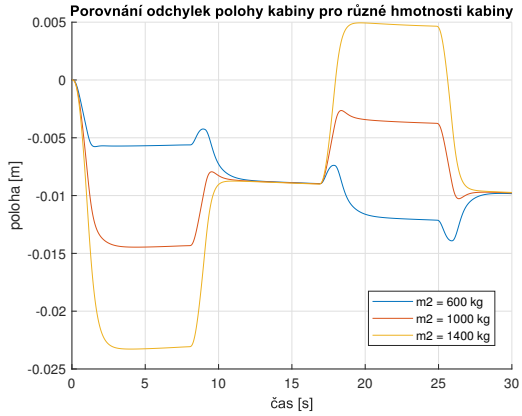
(b) Přejezd z 1. do 10. patra

Obrázek 73: Ukázka zlepšení pomocí přímé vazby

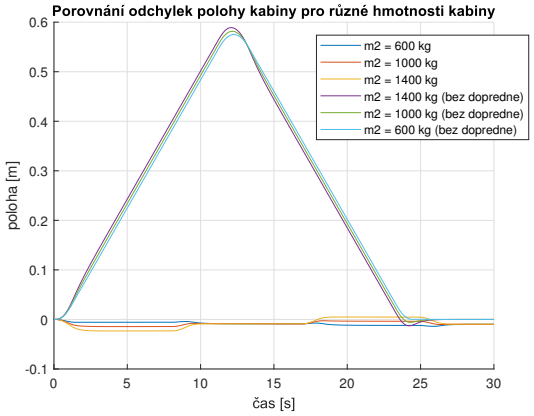
Dále by stálo za to vysvětlit jev, který je viditelný zejména na detailních průbězích odchylek v předchozích kapitolách.Vidíme, že vlastně nikdy poloha nedosáhne té naší kýžené polohy. To je dáno tím, že regulujeme rychlost a není zde žádná smyčka na polohu. Z důvodu ne zcela přesného sledování trajektorie rychlosti dochází logicky i k odchylce ve sledování polohy. Ta se v závěru promítne právě v odchylce od koncové polohy jako takové. Určitě by bylo možné to vyřešit kaskádní regulací na polohu, ale nemyslím si, že je to nutné v tomhle případě. I v reálu se to řeší, jak už jsem také zmiňoval, právě pomocí koncových spínačů a zpomalení těsně před koncovou polohou. Díky tomu dojde při každém zastavení i k opětovnému rekalibrování systému. V tomto ohledu má výhodu robustní regulátor, který je přesný z podstaty své struktury, i když v reálném použití za působení poruch by bylo rovněž třeba nějakých ošetření (kaskáda na polohu, koncový spínač...)

Tato nepřesnost zde vzniká zejména z toho důvodu, že dopředné filtry jsou naprosto přesnou inverzí přenosů systému v daných patrech. U nelineárního modelu samozřejmě platí tedy jen v jednom okamžiku, případně můžeme říct, že přibližně platí v malém okolí. Jakmile se kabina z této polohy odchýlí, inverze již není přesná, tudíž není přesné generování ideálního vstupu, tím pádem zde dochází k nějaké odchylce. To by bylo ve skutečnosti možné řešit například pomocí více linearizací, což není vzhledem k implementaci podstatný rozdíl. Kdyby se provedlo více linearizací, mohli bychom lépe vybírat použitelný filtr právě v daném problematickém polynomiálním průběhu rychlosti. Přesnost dopředné vazby lze tedy tímto směrem takřka donekonečna vylepšovat – záleží na našich možnostech.

Co bych naopak chtěl vyzdvihnout, je robustnost v jiném směru. Je samozřejmě potřeba, aby byl systém robustní vzhledem k neurčitostem. Takovou zřejmě nejdůležitější neurčitostí je v našem případě hmotnost kabiny. Hmotnost zátěže je předem daná a z matematických důvodů, vzhledem k řízení i k tomu, jak je to praktické zejména vůči úspoře energie, jsme zvolili zátěž o něco těžší než kabinu. To je vysvětlováno na začátku práce. V podstatě tedy mohou nastat 3 možnosti: buď je kabina lehčí než zátěž, stejně těžká, nebo naopak těžší. Ukazuje se, že pro řízení je nejhorší varianta, kdy je kabina lehčí než protizávaží. Z tohoto důvodu by bylo možná lepší navrhnout kabinu v základě stejně těžkou, jako je závaží. Zjistil jsem však, že v praxi se to tak takřka nikdy nedělá, protože jde také logicky o peníze a úsporu energie. A přece si nebudeme jen teoreticky ulehčovat práci, když by se to tak skutečně nepoužívalo. Dopředná vazba sice tedy nefunguje úplně spolehlivě ve větším okolí okolo dané linearizace (kterou lze libovolně zjemnit), ale zato je robustní vzhledem k proměnlivé hmotnosti. Právě tato robustnost vůči neurčitosti váhy kabiny je znázorněna na obrázcích 74 a 75 opět i s porovnáním při nevyužití dopředné vazby.

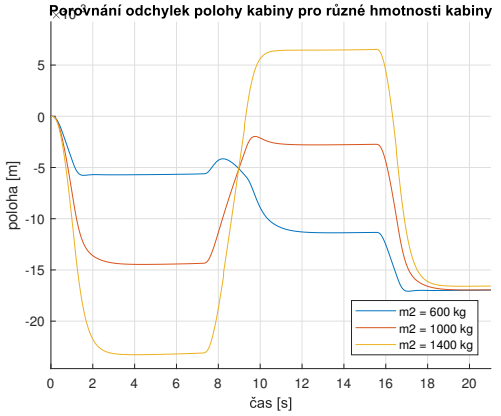


(a) S dopřednou vazbou

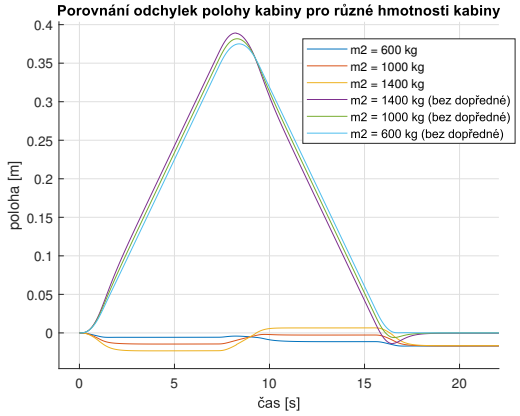


(b) Porovnání s robustním regulátorem

Obrázek 74: Porovnání v robustnosti vůči hmotnosti kabiny při přejezdu z 1. do 10. patra



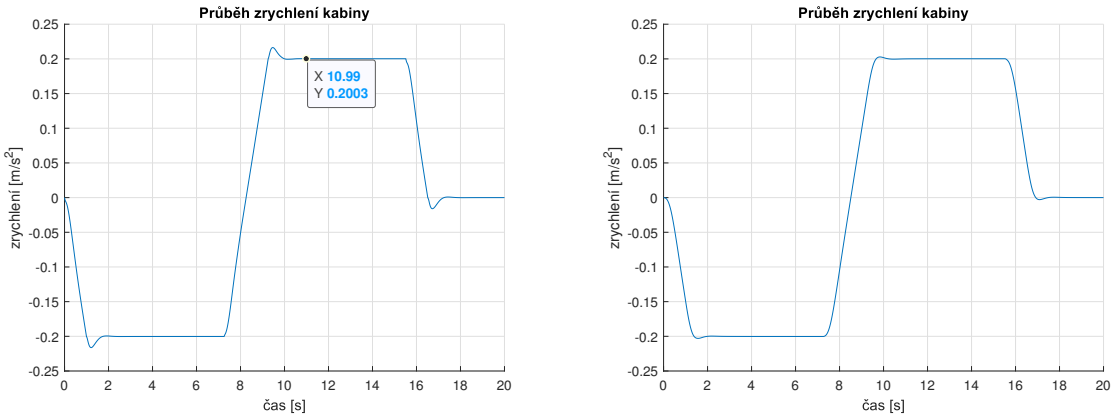
(a) S dopřednou vazbou



(b) Porovnání s robustním regulátorem

Obrázek 75: Porovnání v robustnosti vůči hmotnosti kabiny při přejezdu z 1. do 5. patra

Naopak zhoršení přichází v případě, kdy se podíváme na průběhy zrychlení (viz obrázek 76). Zkrátka a dopbře, je to vlastně pořád jeden a ten samý problém dokola – dopředná vazba počítá s příliš přesnou znalostí modelu (což při praktickém využití bude další oříšek) a regulaci jako takovou touto nepřesností zhoršuje. Ale je to jako se vším v automatickém řízení – jedno vylepšení je na úkor jiného a my musíme dělat kompromisy. Troufám si tvrdit, že zde se ten kompromis povedl nalézt poměrně dobře, protože zhoršení regulace není až tak zlé, a na oplátku jsme získali okamžité vysledování reference, což může mít velkou cenu.



(a) S dopřednou vazbou

(b) Bez dopředné vazby

Obrázek 76: Porovnání zrychlení při přejezdu z 1. do 5. patra

Poznámky k softwareovému řešení

Jelikož zde je uzavření určité části práce, rád bych objasnil fungování přiložených skriptů a simulačních schémat vztahující se k dokončenému řešení přímovazebních filtrů a jejich bezrázovému přepínání včetně rozvrhování těchto přepínacích časů.

Celou simulaci je z uživatelského hlediska možné spustit přímo z jednoho skriptu `primovazebni_reg.m`, kterému postačí zadat počáteční a koncové patro:

```
1     pocPatro = 1;
2     koncPatro = 5;
3
4     x2=(33-pocPatro*3);
5     x2f=(33-koncPatro*3);
6
7     %Integracni konstanty pro vysledovani
8     Ku = [0.3, 0.37, 0.6, 0.8, 6, 6, 6, 4, 4, 4];
9     Kw = [-700, -700, -200, 0.08, -400, -350, -700, -200,
    -300, -400];
```

Další možností je volit parametry modelu, ale v tom případě by bylo třeba upravit výše uvedené integrační konstanty pro vysledování přímovazebních filtrů, které se změnou parametrů modelu logicky také změní. Pokud by byly měněny fyzikální parametry modelu, bude třeba je změnit i ve skriptu `linearizace_10_pater.m` a vygenerovat nové přenosy na rychlost kabiny a motoru, které jsou následně používány k vytváření přímovazebních filtrů. Pokud se parametry změní výrazně, dost možná nebude správně fungovat robustní zpětnovazební regulátor.

S čím je možné bez větších důsledků hýbat, je hmotnost kabiny $m_2 \in < 600; 1400 >$ a logicky také parametry trajektorie jako je maximální rychlost, zrychlení, jerk, ale je třeba mít na zřeteli, že jsou to *parametry motoru*.

Typ trajektorie lze snadno zjistit podle parametrů, což je podrobněji ukázáno v kapitole o generátoru trajektorie a přímo odvozeno v této práci [4]. Tím rozliším dva typy, které potřebuji pro rozhodnutí o počtu přepínacích časů (viz obrázek 65). Zároveň dostanu i přepínací časy v rámci generátoru trajekotrie, tedy časy, ve kterých dochází ke změně hodnoty jerku. Ty mohu využít pro rozhodnutí, v jakých časech se bude přepínat – pro první případ (obrázek 65) budu chtít přepínat těsně před 2. a 6. změnou jerku (protože 3. a 4. čas je shodný a nedojde tudíž ke změně) a pro druhý případ (obrázek 66) chci přepínat před 2., 4. a 6. změnou jerku. Vždy tedy přepínám za konstantního zrychlení, proto ty sudé násobky změny jerku. Zároveň pokud je trajektorie druhého typu, ale čas konstantní rychlosti je jen velmi krátký, použiji jen tři filtry jako v prvním případě. Je to kvůli tomu, že by nestačilo dojít k vysledování a vznikl by ráz, a ani to není potřeba, protože kabina je pořád skoro stejně vysoko. Lépe to jde asi vidět na následující ukázce kódu:

```
1     if typ == 1
2         a = casy(2)-0.5;
3         b = casy(6)-0.5;
4         c = 1000000;
5
6         elseif typ == 2
7         if casy(4)-casy(3) < 3
8             a = casy(2)-0.5;
```

```
9               b = casy(6)-0.5;
10              c = 1000000;
11          else
12              a = casy(2)-0.5;
13              b = casy(4)-0.5;
14              c = casy(6)-0.5;
15          end
16      end
```

Následně jsou nalezeny přenosy odpovídající potřebným časům přepnutí podle polohy kabiny v tom okamžiku a spočteny jejich inverze. Z nich se pak uloží koeficienty použité v modelu v Simulinku `simulace_s_primou_vazbou.slx`. Simulace je rovněž provedena automaticky a vykresleny jsou základní grafy – zrychlení kabiny, odchylka polohy a rychlosti kabiny od reference. Pokud chcete vidět víc, v simulinku jsou vytažené bločky `scope` ve všech možných zajímavých místech (regulační odchylka, rychlost a přesnost vysledování jednotlivých filtrů atd.). Pro přehlednost modelu jsou použity subsystémy, pomocí kterých se můžeme ponořit až do vrstvy inverzního nekauzálního přenosu použitého v přímovazebním filtru. Struktura je zhruba takováto: `Přímovazební filtr` → `Čtveřice filtrů a systém jejich vysledování` (lokální zpětné vazby s integračním regulátorem) → `Rozdělení dvojic filtrů` F_U a F_W → `Model inverze nekauzálního přenosu na rychlost kabiny`.

4 Časově proměnné přímovazební filtry

4.1 Linearizace podél trajektorie

Po vytvoření struktury s bezrázovým přepínáním přímovazbeních filtrů vznikl poměrně komplikovaný model se složitým algoritmem rozhodování podle několika kritérií. Celý tento systém je velmi náročný na praktickou implementaci, kde máme často možnost vytvořit pouze regulátor s klasickou PID strukturou plus nějaký filtr pro tvarování reference. K tomu by mohlo být vhodné využít linearizaci podél trajektorie, což znamená, že bychom dostali matici dynamiky A časově proměnnou v závislosti na trajektorii, kterou bychom jí dali. Více dopodrobna se tím zabývá Břetislav Kubeš ve své bakalářské práci [10], já bych zde znázornil jen hlavní myšlenku provedení této linearizace.

V podstatě je to nadřazený případ linearizace v rovnovážném bodě, protože zde uvažujeme celou pracovní trajektorii a pohybujeme se v odchylkách kolem ní (viz vztah 45)

$$\begin{aligned} \dot{x_p}(t) &= f\left(x_p(t), u_p(t)\right) \\ y_p(t) &= h\left(x_p(t), u_p(t)\right) \end{aligned} \tag{44}$$

$$\Downarrow$$

$$\begin{aligned} \dot{x_p}(t) + \Delta\dot{x}(t) &= f\left(x_p(t) + \Delta x(t), u_p(t) + \Delta u(t)\right) \\ y_p(t) + \Delta y(t) &= h\left(x_p(t) + \Delta x(t), u_p(t) + \Delta u(t)\right) \end{aligned} \tag{45}$$

Následně se provede klasický rozvoj do Taylorovy řady a členy vyšších řádů zanedbáme (tím získáme právě tu linearizaci – myšleno vzhledem ke koeficientům). Dostáváme tedy lineární odchylkovou aproximaci (viz vztah 47)

$$y_p(t) + \Delta y(t) = h\left(x_p(t) + \Delta x(t), u_p(t) + \Delta u(t)\right) \tag{46}$$

$$y_p(t) + \Delta y(t) = h\left(x_p(t), u_p(t)\right) + \frac{\partial h}{\partial x}|_{(x_p,u_p)}\Delta x(t) + \frac{\partial h}{\partial u}|_{(x_p,u_p)}\Delta u(t) \tag{47}$$

Po dosazení ze vztahu 44 se můžeme dostat až na tvar čisté odchylky:

$$y_p(t) + \Delta y(t) = y_p(t) + \frac{\partial h}{\partial x}|_{(x_p,u_p)}\Delta x(t) + \frac{\partial h}{\partial u}|_{(x_p,u_p)}\Delta u(t) \tag{48}$$

$$\Delta y(t) = \frac{\partial h}{\partial x}|_{(x_p,u_p)}\Delta x(t) + \frac{\partial h}{\partial u}|_{(x_p,u_p)}\Delta u(t) \tag{49}$$

a celý model nyní můžeme vyjádřit v tomto tvaru, jak je patrné ve vztahu 63

$$\Delta\dot{x}(t) = \frac{\partial f}{\partial x}|_{(x_p,u_p)}\Delta x(t) + \frac{\partial f}{\partial u}|_{(x_p,u_p)}\Delta u(t)$$

$$\Delta y(t) = \frac{\partial h}{\partial x}|_{(x_p,u_p)}\Delta x(t) + \frac{\partial h}{\partial u}|_{(x_p,u_p)}\Delta u(t) \tag{50}$$

Důležité je si uvědomit, že stavy jsou časově proměnné, tedy že se skutečně pohybujeme podél trajektorie a odchylka určuje vzdálenost od této trajektorie. S takto vytvořeným lineárním modelem se náš problém vlastně přetransformuje na regulaci do nuly, protože naše nula teď pomyslně leží na trajektorii, kolem které byla linearizace provedena.

Aby to však nebylo jednoduché, trajektorie, jak již bylo zmíneno (viz obrázek 16), není zcela hladká a spojitá, je rozdělená na sedm časových intervalů. A tím pádem nelze popsat jedním polynomem, ale je třeba ji definovat právě podle těchto časů odlišnými polynomy. K tomu potřebujeme vygenerovat ekvivalentní trajektorie na všech časově proměnných nenámých v rovnicích – tedy pro délky lana na obou stranách (l_1 a l_2), pro polohu kabiny i protizávaží (x_1 a x_2) a pro polohu motoru (φ). Díky těmto znalostem je možné naplnit matici A časově proměnnými polynomy. Celý výpočetní postup pak vypadá následovně:

- Zadání pater, mezi kterými chci přejet
- Na tomto základě se spočte trajektorie s různým přepočtem pro různé proměnné
- Z toho mohu spočíst polynomy jež obsahuje matice A
- V každém časovém okamžiku simulace dosadím aktuální čas, vytvořím state-space a z něj si vezmu potřebné přenosy (přenos na rychlost kabiny, který invertuji, a přenos na rychlost motoru)
- Z těchto přenosů vytvořím přímovazební filtry, jak bylo rozebíráno v předchozí kapitole (viz obrázek 43)

Velkou výhodou tohoto řešení je, že filtr mám v podstatě jeden, ale časově proměnný, takže není potřeba provádět vysledování. V podstatě je spíše filtrů tolik, co vzorků simulace, ale znamená to, že se mění *nejvíce spojitě, jak mohou.*

Také je třeba podotknout, že dosazování do matice a vytváření přenosů v každém časovém okamžiku simulace je poměrně náročné a proto je lepší nechat předpočítat několik tisíc matic A (pro každý časový okamžik diskrétní simulace) a přenosů dopředu a následně je již jen používat.

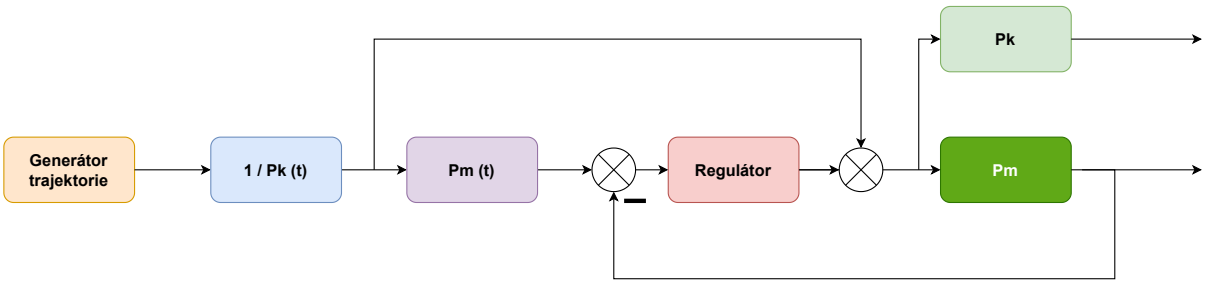
4.2 Implementace a výsledky

Díky tomu, že časový filtr sedí přesně v každém okamžiku simulace, jsme schopni regulovat velmi přesně a nedochází zde ani k odchylce od cílové polohy (viz obrázek 78). To bylo u předešlých variant způsobeno tím, že filtr v mnoha místech trajketorie vlastně neseděl a byl jen velmi vzdálenou aproximací, tudíž docházelo ke generování *trochu jiné* reference a tím pádem k *trochu jinému* průběhu křivky rychlosti. Z toho logicky vyplývá výsledný posun polohy. To například nenastávalo u obyčejného robustního zpětnovazebního PI regulátoru, protože ten sice sledoval trajektorii se zpožděním, ale za to ji kopíroval přesně. Zde je vlastně filtr v každém okamžiku přesný, tím pádem je konečná poloha rovněž správná.

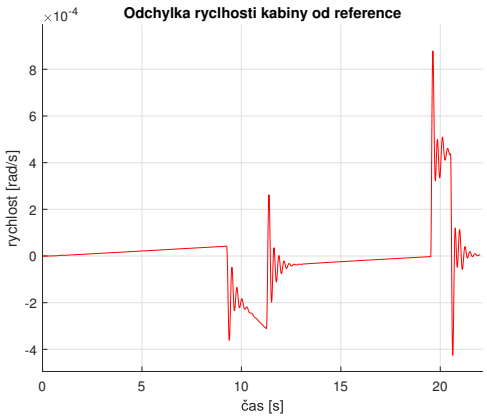
Když porovnáme odchylku polohy kabiny od reference se stejnou odchylkou vznikající při přepínání a vysledování jen pár filtrů, vidíme docela markantní rozdíl. Natož když použijeme srovnání s pouhým zpětnovazebním řízením (viz obrázek 79).

Rovněž je třeba zmínit, že díky přesným filtrům se změnšili potřebné zásahy regulátoru natolik, že bylo možné odstranit notch filtr a tím pádem opět získat zpět ztracené vlastnosti regulátoru odregulovat poruchu na zatlumené frekvenci (viz schéma 77). I regulátor jako takový bylo možné ještě dále upravit – nyní je agresivnější, lépe odreguluje poruchy a jeho zásahy v případě potřeby rychleji vrátí systém na požadovanou trajektorii (viz vztah 51). To se velmi projevilo na průběhu zrychlení, který je oproti předchozím případům výborný (viz obrázek 80). První zlom sleduje referenci dokonale, u druhých dvou vidíme trochu kmitavý přechodový děj, který však poměrně rychle odezní. Troufám si říci, že pohodlí cestování se tímto posunulo na komfortnější úroveň. Působí to, že jsme konečně nalezli skutečně dobrý způsob řízení, který (ač je výpočetně velmi náročný) nejlépe sleduje požadovanou trajektorii i odstraňuje nepříjemné překmity ve zrychlení.

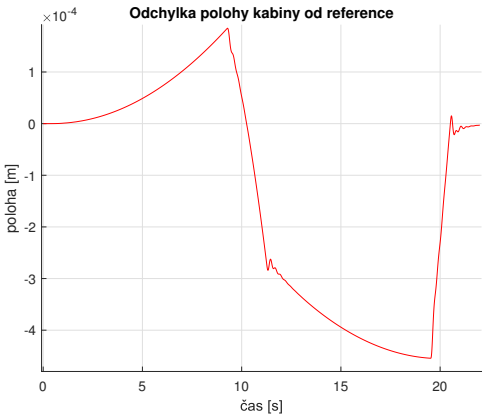
$$F_R(s) = \frac{8.415s + 10.68}{s} \tag{51}$$



Obrázek 77: Schéma časově proměnných filtrů s využitím linearizace podél trajektorie

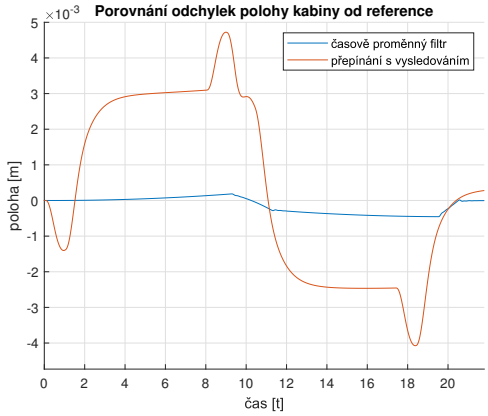


(a) Odchylka rychlosti od reference

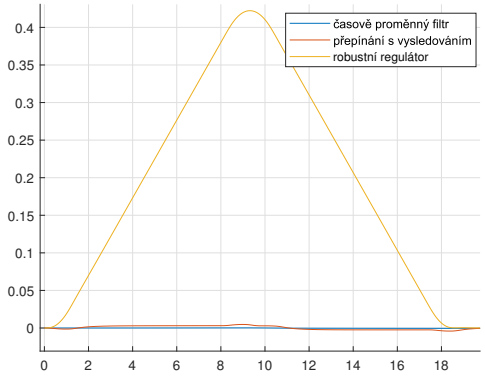


(b) Odchylka polohy od reference

Obrázek 78: Odchylky při přejezdu ze 3. do 8. patra za použití linearizace podél trajekotrie



(a) Porovnání s přepínáním a vysledováním

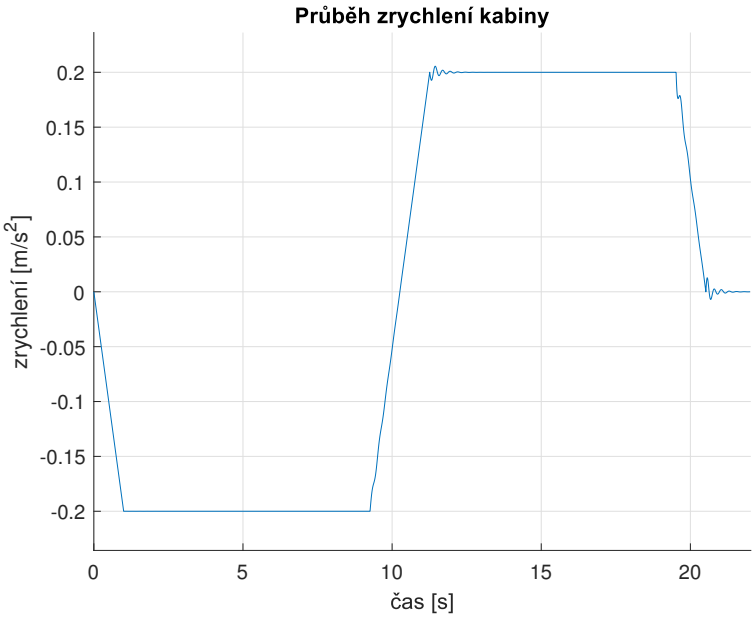


(b) Obecné porovnání

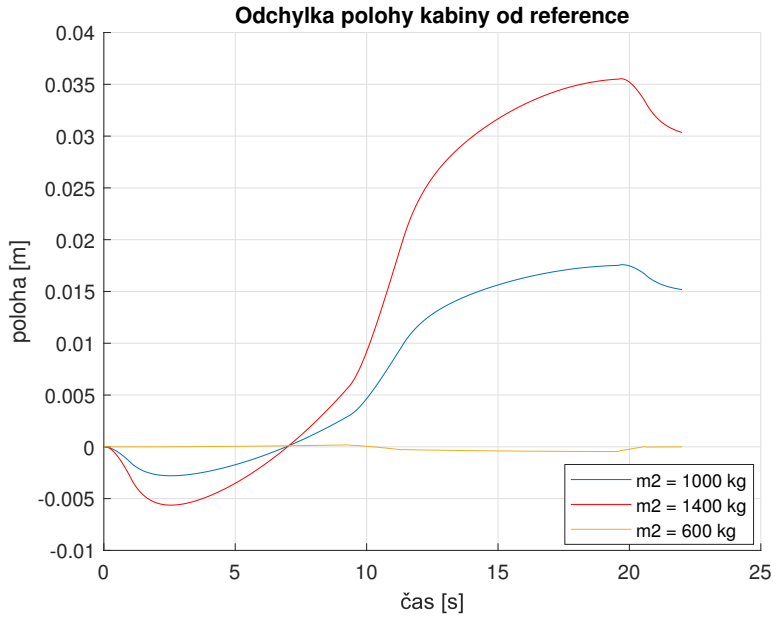
Obrázek 79: Porovnání odchylek při přejezdu ze 3. do 8. patra za použití linearizace podél trajekotrie

Vyskytl se ale opět nějaký jiný problém, jako vždy, když něco funguje podezřele dobře. Jak si můžeme všimnout na obrázku 81, vše se kazí, jakmile zadáme kabině jinou hmotnost, než pro kterou byly spočteny filtry. A rozdíl odchylek se zde pak projevuje velmi výrazně. To se dá opět logicky vysvětlit – tím, že je regulace závislá zejména na dopředných filtrech, které výrazně tvarují referenci přesně tak, aby byl výstup ideální, je tedy i mnohem více závislá na jejich správném a přesném určení a odchylka ve hmotnosti má tím pádem na odezvu větší vliv. Tudíž tento způsob nalezne využití spíše v okamžiku, kdy známe dobře model řízeného systému, nebo alespoň potřebné přenosy, protože není tak robustní. Například zde bychom mohli říct, že by se hodil u výtahů, které vnitřně váží před odjezdem svůj náklad. Pak by se teprve dosadilo do linearizace s daným doplňujícím parametrem a spočetly potřebné filtry. Hodilo by se nad tím uvažovat v okamžiku, kdy bychom potřebovali velmi přesně a rychle vysledovat danou trajektorii, což si zrovna v případě výtahu neumím zcela představit takovouto kritickou aplikaci. Pokud by to ale třeba bylo, tento přístup je dobré zvážit. Nemusíme se však omezovat jen na výtahy, obecně se dá tato problematika velmi snadno vztáhnout na libovolný tříhmotový systém,

který chceme řídit, a například u nějakých dopravníků už může být tato dochvilnost vyžadována.



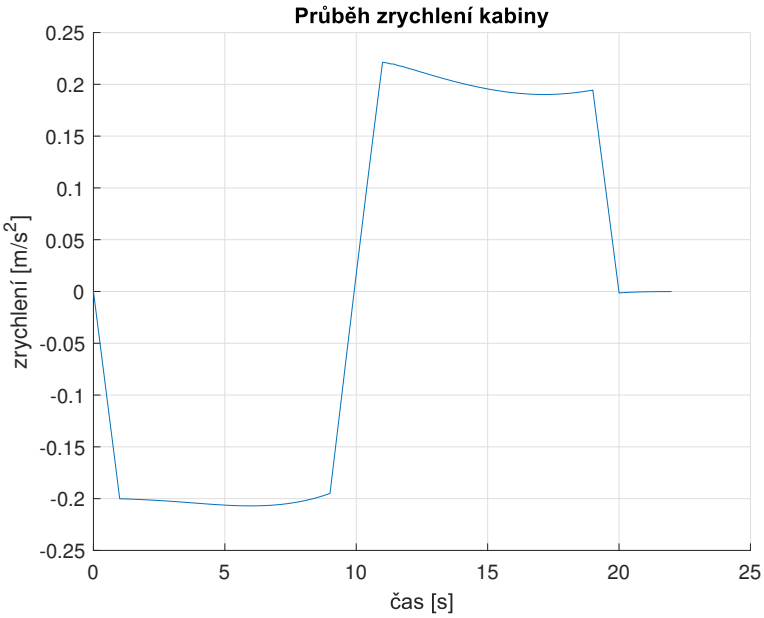
Obrázek 80: Průběh zrychlení za použití linearizace podél trajekotrie



Obrázek 81: Porovnání průběhu odchylek polohy se změnou hmotnosti kabiny

Kmitání v průběhu zrychlení však jde ještě dále upravit a vylepšit dalším zagresivněním regulátoru – konkrétně zvýšením integrační složky. Průběh je pak mnohem hladší a plynulejší bez jakýchkoliv kmitů (viz obrázku 82), na druhou stranu dochází k trochu zvláštnímu jevu. Tím je jakési plynulé zvlnění kolem požadované hodnoty zrychlení, které se ani nedá považovat za kmit a je určitě přínosem pro komfort jízdy v kabině. Záleží na našich preferencích. Ale můžeme si všimnout ještě jedné výhody tohoto přístupu – dojezd

do koncové polohy je ve zrychlení naprosto hladký a přesný a tím pádem i poloha najede klidně a správně tam, kam má (bez uvažování poruch).



Obrázek 82: Průběh zrychlení za použití linearizace podél trajekotrie

5 Robustní FIR filtr

5.1 Teorie návrhu

FIR filtr je v podstatě způsob tvarování reference, která se modifikuje takovým způsobem, aby eliminovala vybuzení kmitů v následujícím kmitavém systému (viz schéma 83). Otázkou samozřejmě je, jak jej spolehlivě navrhnout tak, aby skutečně byl robustní nejen vůči neurčitosti ve hmotnosti, ale právě i vůči neurčitosti v délce odvinutého lana. Tímto se zabývá Ing. Martin Goubej, Ph.D. ve své práci [11] pojednávající o vytváření diskrétních filtrů pomocí elementárních dopravních zpoždění s daným zesílením. Právě tím *jak* naladit tato zesílení se práce zabývá. Tato metoda byla použita k naladění FIR filtru pro náš model výtahu s danými fyzikálními parametry, aby bylo možné provést porovnání této implementačně snazší a robustní metody s mým návrhem gain-schedulingu přímovazebních filtrů a s dopřednými filtry proměnnými v čase na základě linearizace podél trajektorie.

Tento typ filtrů má oproti klasickým velké výhody. A i nyní díky němu bude možné odstranit náš notch filtr, což znamená, že získáme základní robustní regulátor s větší šířkou pásma a tedy schopností lépe odregulovat případné poruchy. Zároveň může být regulátor agresivnější a to, že by normálně měl příliš kmitavou odezvu, pokryje právě zmiňovaný FIR filtr. Níže následuje stručné vysvětlení principu a odvození tvaru filtru a jeho návrhu.

Tento tvarovací filtr může být popsán v časové oblasti následující impulsní funkcí:

$$h_s(t) = \sum_{i=1}^{n} A_i \delta(t - t_i), 0 \leqslant t_1 < t_{i+1}, A_i \neq 0, \tag{52}$$

kde A_i značí amplitudy i-tého pulsu a δ je diracův puls posunutý o čas t_i. Z toho je možné pomocí konvoluce určit odezvu tvarovacího filtru $v(t)$:

$$v(t) = \int_{-\infty}^{\infty} h_s(\tau) u(t-\tau) d\tau = \int_{-\infty}^{\infty} \left(\sum_{i=1}^{n} A_i \delta(t - t_i) \right) u(t-\tau) d\tau = \sum_{i=1}^{n} A_i u(t - t_i) \tag{53}$$

Při návrhu klasického FIR filtru je snahou zvolit vhodná zesílení k dopravním zpožděním, aby došlo k potlačení kmitavých módů následujícího systému v konečném čase. Začíná se zjednodušením systému až na takovou míru, že se z něj vytáhne pouze kmitavá část, kterou chceme tlumit, tedy přenos ve tvaru:

$$P_k(s) = \frac{\omega_k^2}{s^2 + 2\xi_k \omega_k s + \omega_k^2}; \quad k = 1 \dots m, \tag{54}$$

s tím, že je možné toto provádět pro více přenosů zároveň, tedy v našem případě pro linearizace v každém patře. To nese tu výhodu, že výsledný filtr bude robustní. V časové oblasti je možné přepsat propojení filtru s řízeným systémem pomocí celkové impulsní funkce:

$$h(t) = \sum_{i=1}^{n} A_i h_k(t - t_i) = \frac{\omega_k}{\sqrt{1-\xi_k^2}} e^{-\xi_k \omega_k t} \sqrt{C(\omega_k, \xi_k)^2 + S(\omega_k, \xi_k)^2} sin(\omega_d t – \psi)$$

$$C(\omega_k, \xi_k) = \sum_{i=1}^{n} A_i e^{\xi_k \omega_k t_i} cos(\omega_d t_i), \;\; S(\omega_k, \xi_k) = \sum_{i=1}^{n} A_i e^{\xi_k \omega t_i} sin(\omega_d t_i), \;\; \psi = arctan \frac{S}{C} \tag{55}$$

Pak můžeme vyjádřit maximální vybuzené kmity po odeznění přechodové doby filtru jako:

$$A^s_{max}(\omega_k, \xi_k) = \frac{\omega_k}{\sqrt{1-\xi_k^2}} e^{-\xi_k \omega_k t} \sqrt{C(\omega_k, \xi_k)^2 + S(\omega_k, \xi_k)^2} \tag{56}$$

kde nejhorší případ nastává v okamžiku, kdy sečteme amplitudy všech harmonických:

$$A_y^{max} = \sum^{\forall k} \frac{\omega_k}{\sqrt{1-\xi_k^2}} e^{-\xi_k \omega_k t} K_k \sqrt{C_k^2 + S_k^2} \triangleq R_k \sqrt{C_k^2 + S_k^2} \tag{57}$$

Nakonec již můžeme zapsat minimalizaci ztrátové funkce, pomocí které nastavíme nejlepší sadu parametrů zesílení k dopravním zpožděním tvořícím FIR filtr:

$$\{A_i\}^* = \underset{\forall A_i}{\operatorname{argmin}} \left\{ J = \sum_{\forall A_i} R_i (C_i^2 + S_i^2) \right\} \tag{58}$$

V případě této metody je návrh klasického FIR filtru, který má řadu praktických omezení, trochu pozměněn. Klíčové je nejprve model převést pomocí modální transformace do tvaru vyjadřující statickou aa kmitavou část systému:

$$P(s) = \frac{\Phi_{11}^2}{s^2} + \sum_{i=2}^{n} \frac{\omega_{1i}^2}{s^2 + 2\xi_i \omega_i s + \omega_i^2}$$

Při odvození se zaměříme pouze na proměnnou kmitavou část systému:

$$P_f(s) = \sum_{i=2}^{n} \frac{\Phi_{1i}^2}{s^2 + 2\xi_i \omega_i s + \omega_i^2}, \tag{59}$$

přičemž nejdůležitější jsou zesílení Φ_{1i} v čitateli, která udávají dílčí příspěvky jednotlivých módů do celkové výstupní impuslní odezvy $h_y(t)$, ve které se minimalizuje celková úroveň vibrací:

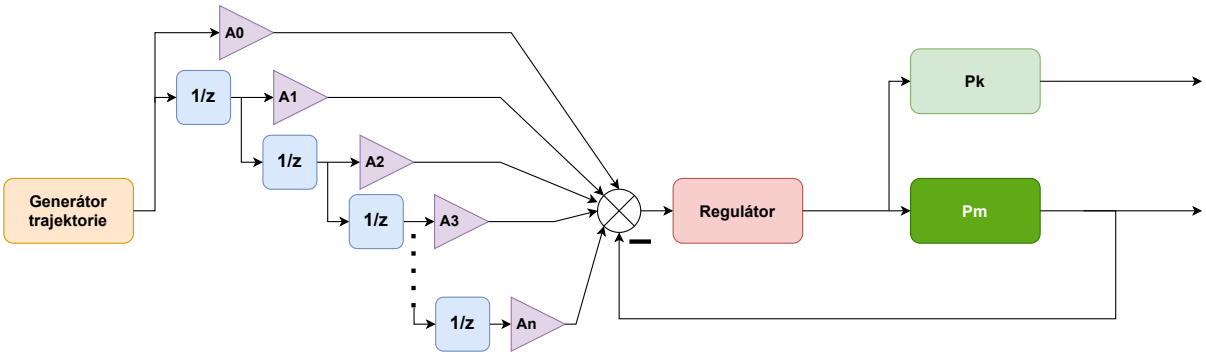
$$h_y(t) = \sum_{k=2}^{n} \frac{\omega_k}{\sqrt{1-\xi_k}} e^{-\xi_k \omega_k t} \sqrt{C_k^2 + S_k^2} sin(\omega_d t - Psi) K_k \tag{60}$$

a opět se zaměříme na nejhorší případ při součtu amplitud jednotlivých harmonických:

$$A_y^{max} = \sum_{\forall k} \frac{\omega_k}{\sqrt{1-\xi_k^2}} e^{-\xi_k \omega_k t} K_k \sqrt{C_k^2 + S_k^2} \triangleq R_k \sqrt{C_k^2 + S_k^2} \tag{61}$$

Zde jsou uvedeny jen nejdůležitější vztahy, kompletní vysvětlení a odvození je uvedeno v již zmiňované práci [11]. Nicméně byl tímto způsobem navržen robustní filtr s uvážením všech 60 přenosů (na rychlost kabiny v 10 patrech pro 3 možná naložení, stejně tak na rychlost motoru). Byl použit i nový agresivnější PI regulátor (viz vztah 62), schéma tohoto způsobu regulace je pak znázorněné na schématu 83. Koeficientů filtru $\{A_i\}^*$ je 100, ty tady nebudu vypisovat.

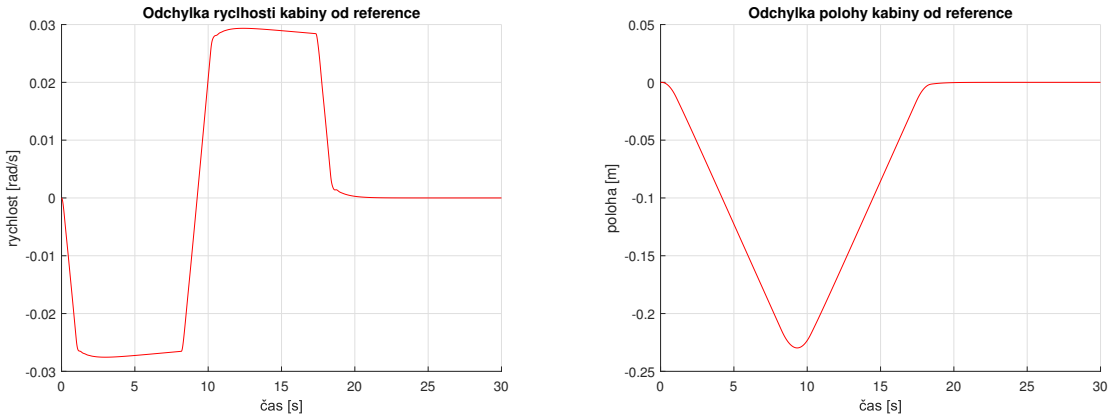
$$F_R(s) = \frac{8.125s + 10.5}{s} \tag{62}$$



Obrázek 83: Schéma navrženého FIR filtru

5.2 Výsledky a simulace

Na obrázku 84 a zejména pak v celkovém porovnání odchylek polohy kabiny od reference na obrázku 87 si můžeme všimnout, že odchylka se opět oproti předchozím přímovazebním filtrů zvětšila. To je dané právě jednak robustností tohoto filtru, ale ještě více jeho charakterem – tím, že je to ve své podstatě jen poskládání různě přenásobených dopravních zpoždění. Z toho důvodu zavádíme do systému celkové dopravní zpoždění o hodnotě *počet zpoždění * jejich hodnota*. Vlivem tohoto zpoždění už rovnou vidíme, že není možné sledovat zadanou trajektorii přesně, ale rovněž se zpožděním. To je vidět právě i na těchto odchylkách.



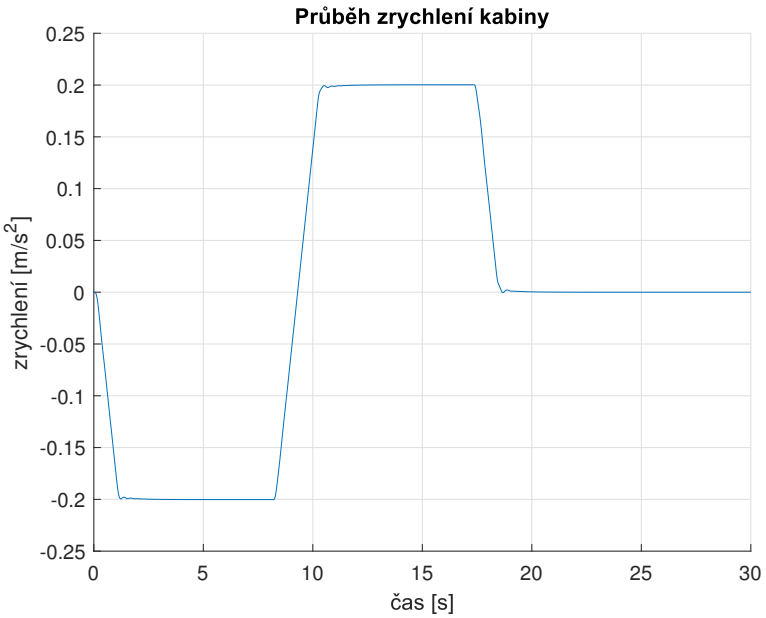
(a) Odchylka rychlosti od reference

(b) Odchylka polohy od reference

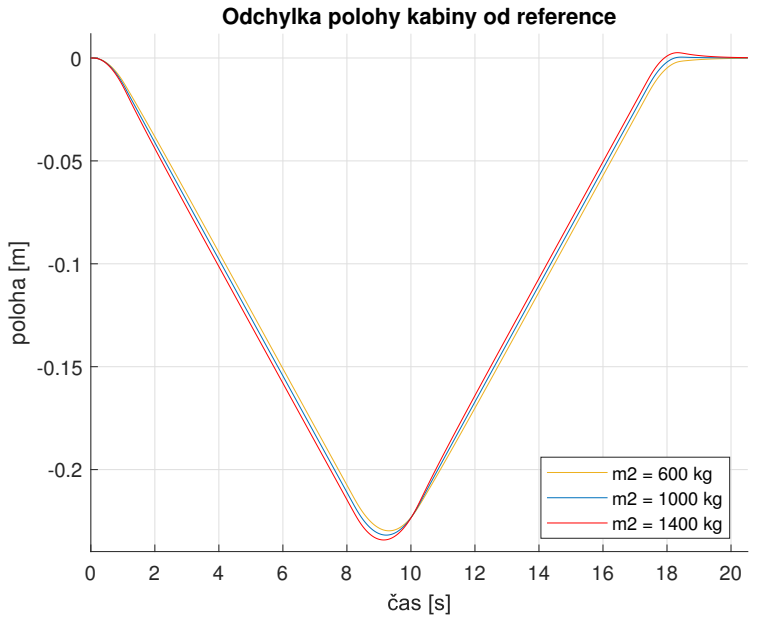
Obrázek 84: Porovnání odchylek při přejezdu ze 3. do 8. patra za použití linearizace podél trajekotrie

Naopak průběh zrychlení je oproti všem předchozím případům krásný (viz obrázek 85). Při přechodech sice lehce kmitá, ale překmit je nulový a průběh hladký. Co se týče zlepšení průběhu zrychlení, tak tento přístup rozhodně pomohl nejlépe (i když jsem se na tuto problematiku po celou dobu zas až tolik nesoustředil). Musíme také vzít v úvahu, že nyní je z regulátoru vypuštěn notch filtr, takže má systém schopnost lépe reagovat na poruchy (oproti samotnému robustnímu regulátoru), a přesto je i průběh zryhlení lepší.

Ještě je dobré zdůraznit, že je filtr skutečně robustní vůči neurčitosti v systému, kterou je v našem případě hmotnost kabiny, což jde vidět na obrázku obrázku 86. To je právě díky návrhu koeficientů na základě všech přenosů, jak jsem zmiňoval na začátku této kapitoly, a dokazuje to vhodnost použití FIR filtru pro potřeby výtahu, protože zde nám určité jím zavedené dopravní zpoždění nevadí.



Obrázek 85: Průběh zrychlení za použití robustního FIR filtru



Obrázek 86: Robustnost FIR filtru vůči hmotnosti

6 Shrnutí a porovnání přímovazebních přístupů

Ačkoliv jsem výsledky v průběhu práce porovnával a snažil se popsat výhody a nevýhody jednotlivých vyzkoušených přístupů, bylo by vhodné ještě nyní po vyzkoušení všeho, co jsem chtěl, porovnat vše dohromady. Celou dobu jsem se zabýval především odchylkou od požadované reference, okamžitým vysledováním, což je pěkná a zjímavá vlastnost, ale u výtahů vlastně ne zas tak důležitá. Nezajímá nás, jestli pojedeme s malým zpožděním, ale zda je jízdá hladká a bezpečná. Proto bych mohl začít porovnáním idexů kvality řízení, které jsem určil následujícně:

Jelikož je u výtahu nejdůležitější hladký průběh zrychlení (kvůli pohodlí cestujících), zaměříme se na derivaci zrychlení kabiny. Jedním kritériem tak bude celková energie tohoto signálu jakožto integrál z kvadrátu jerku kabiny, druhým maximální hodnota jerku kabiny:

$$\int_{-\infty}^{\infty} j_{kab}^2$$

$$max\left(|j_{kab}|\right) \tag{63}$$

Tabulka 1: Kritéria hodnocení kvality regulace

	$\int_{-\infty}^{\infty} j_{kab}^2$	$max\left(\lvert j_{kab}\rvert\right)$	$\int_{-\infty}^{\infty} a_{kab}^2$
Robustní regulátor	0.143	0.210	0.865
Přepínání filtrů s vysledováním	0.184	0.337	0.868
Linearizace podél trajektorie	0.188	0.774	0.707
FIR filtr	0.287	0.223	0.870

Když se podíváme na maximální absolutní hodnotu jerku u linearizace podél trajekotrie, vidíme, že dalece převyšuje ostatní hodnoty. Dost možná je to způsobené jen tím, že pro toto zjištění byla použita numerická derivace a došlo k nějaké chybě, jinak se maximální hodnoty v tomto signálu pohybují spíše kolem 0.310. Z tohoto kritéria jinak nejlépe vychází robustní integrátor, což je dané tím, že jeho přechodový děj neobsahoval žádné *vlnky*, ale pouze jediný (za to docela velký, ale hladký) překmit. Nejhorší průběh z hlediska hladkosti je (když pomineme chybu linearizace podél trajektorie) při přepínání filtrů s vysledováním. U té linearizace podél trajektorie je to však docela podobné, což jde vidět i v grafech, kde je přechodový děj podobně zvlněný (jen při použití přístupu s linearizací podél trajektorie je tento jev menší a rychleji odezní).

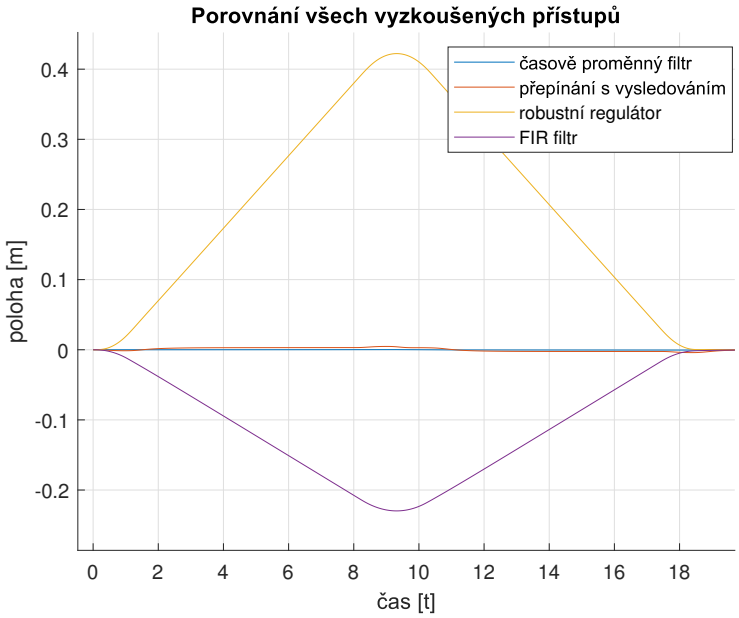
Dále je v tabulce zachycena celková energie signálu kvadrátu zrychlení a jerku. U zrychlení je rozdíl jen velmi malý, což je dané jeho průběhem, kde obsah pod křivkou je příliš velký na to, aby se zde projevil vliv kmitů, které oproti tomu zabírají jen velmi malou část.

Obecně z těchto hodnocení vychází velmi dobře obyčejný robustní regulátor – to je ale dané tím, že jeho průběh je skutečně hladký. Zde se v hodnocení promítla hladkost signálu, což je z hlediska komfortu určitě velmi důležité, ale nejsou zde zohledněné ostatní

diskutované vlastnosti jako je nízký překmit ve zrychlení, přesné sledování reference atp.

Další záležitostí, která nás může zajímat, je, jak přesně je sledována referenční veličina. A vlastně spíše než ta (rychlost) nás zajímá poloha kabiny. To je znázorněno na obrázku 87. Z tohoto hlediska funkci nejlépe plní přepínaní přímovazebních filtrů ať už v malém množství za pomoci vysledování, tak ve velkém množství (diskrétně časově proměnných). Použití velkého množství filtrů (využití linearizace podél trajekotrie) vede k nejlepším výsledkům, zároveň je řešení velmi náchylné na přesnost matematického modelu, jak bylo v příslušné kapitole diskutováno. Přepínání 3 až 4 filtrů s navrženým systémem vysledování je robustnější vůči neurčitosti (zejména testováno na proměnlivé hmotnosti kabiny). Na druhou stranu použití časově proměnných filtrů umožňuje odstranění notch filtru zavedeného k regulátoru, který zmenšuje šířku pásma a schopnost odregulovávat poruchy. Tento přístup je tak robustnější zas v jiném směru.

FIR filtr z tohoto hlediska vychází poměrně špatně, ale jde o to určit priority. Zrovna v tomto případě obyčejného výtahu to zřejmě okamžité sledování trajektorie nebude, tudíž by mohl být vhodný zejména díky hladkému průběhu křivky zrychlení kabiny. Rovněž i tento přístup umožnil odstranění notch filtru, tudíž zabírá větší šířku pásma. Jeví se tak jako nejlepší varianta pro klasické použití díky snadné implementaci, jednoduché struktuře a robustnosti v neurčitosti parametrů i v odregulování poruch. Jedinou nevýhodou je ne příliš přesné vysledování trajektorie.



Obrázek 87: Porovnání přesnosti sledování reference všech vyzkoušených přístupů

Ještě bych rád provedl testy související s rychlostí jízdy. Standardně jezdí výtahy v bytových domech rychlostí 0.6 m/s až 1 m/s [5]. Zde celou dobu uvažujeme parametry takové, že rychlost jízdy dosahuje cca až 2 m/s. Jestliže je však regulace kvalitní, mělo by být možné dosáhnout i vyšších rychlostí. Rád bych se pokusil dostat na nějakých 8 m/s, což odpovídá zhruba 30 km/h. To není žádná velká rychlost, v úvodu jsem zmiňoval i výtahy, které jezdí 70 km/h, ale na druhou stranu je potřeba dosáhnout poměrně velkého zrychlení, abychom se na tuto rychlost dostali.

Rychost jako taková totiž problém není, pokud budeme počítat s dostatečně dlouhým horizontem jízdy a ponecháme parametry maximálního jerku a zrychlení stejné a jen zvýšíme maximální možnou rychlost, kabina bude postupně (ale plynule a klidně) zrychlovat a možnosti, parametry a kvalita regulace nebudou změněny. Systém se na danou rychlost dostane a vše proběhne stejně jako v příkladech doposud. Problém však je, že chceme vysoké rychlosti dosáhnout pořád na úseku 10 pater. To znamená, že požadujeme strmější náběh zrychlení, tudíž mnohem vyšší pulsní zásah jerku. Jak už se ale ukázalo (například při přepínání 2 – 3 filtrů) právě tato skoková změna jerku má velký vliv na vybuzení vibrací systému, takže logicky s větším skokem dojde k vybuzení větších kmitů. Rovněž pasažér bude touto změnou postižen, protože právě i jerk je veličinou, kterou vnímají naše *biologické senzory*. Právě na jerk se kladou omezení, která mají zpříjemnit cestu cestujícím, takže jednou z dalších možností by bylo postoupit ještě o derivaci dále, aby průběh jerku nebyl skokový nýbrž lineární.

Níže na obrázcích 88 a 89 jsou znázorněné průběhy zrychlení pro jednotlivé navržené přístupy při pokusech s vysokou rychlostí. Obecně vypadá průběh dobře, při bližším pozorování si můžeme všimnou několika věcí, které mají své odůvodnění. První z nich je trochu "hrbolatý"průběh u přístupu s přepínáním filtrů. To je způsobeno právě příliš prudkou změnou jerku, která vybudí mnohem větší kmity při pokusu odregulovat rozdíl filtrů při vysledování. Rovněž se velmi sníží čas potřebný k vysledování. Nicméně průběh je přijatelný. Dá se i vylepšit *zagresivněním* integračních regulátorů (vhodnou úpravou vysledovacích konstant) sloužících k vysledování výstupů filtrů.

Linearizace trajektorie se změna rychlosti takřka nedotkla, což je logické vzhledem k tomu, že poskytujeme takřka ideální vstup v každém časovém okamžiku. Velmi dobře se zachoval i robustní regulátor, jelikož zde není nic navrch, žádné filtry, které by mohly vybuzovat vibrace a tak dále. V tomto směru je jeden robustní regulátor kvalitním a bezpečným řešením, problém pak ale přichází v jiných aspektech, o kterých jsme mluvili během celé práce a které jsem se snažil odstranit pomocí působení přímé vazby, predikcí, tvarováním reference a podobně.

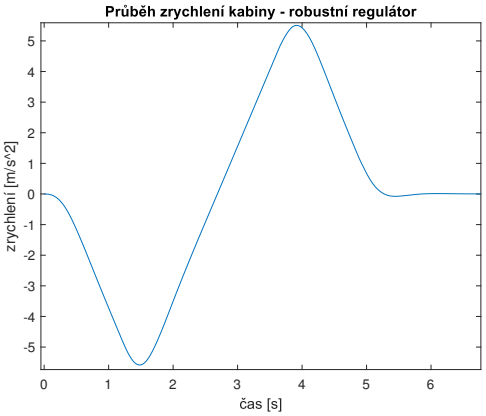
Obecně u výtahů nás tato vlastnost poměrně zajímá, chceme se dostat z jednoho patra do druhého co nejrychleji. Pro to je nejvhodnější použít přístup, který způsobuje nejmenší kmity, které jsou vybuzeny vyšší rychlostí (tedy vyššími skoky dávanými generátorem trajektorie). Z tohoto hlediska by byl nejvhodnější zřejmě FIR filtr případně linearizace podél traektorie, která je ale výpočetně skutečně velmi náročná.

Také nás ale zajímá, jak rychle skutečně kabina dojede do místa, ve kterém zastaví. To je myšleno tak, že generátor vygeneruje trajektorii a čas, kdy by měla být kabina v cílové poloze. Reálná kabina je pak ale vůči této trajektorii určitým způsobem zpožděná, takže v podstatě hledáme rozdíl ideálního a skutečného dojezdu. Také by se to dalo interpretovat jako doba, která uplyne od *stisknutí knoflíku* do skutečného zastavení kabiny. Budeme uvažovat, že kabina je v *cílové pozici* v okamžiku, kdy je její odchylka menší než 1 mm. Výsledky jsou znázorněny v tabulce 2. Ale vzhledem k tomu, že jsem se věnoval především přímovazebním přístupům, které si za cíl kladou minimalizovat odchylku od reference,

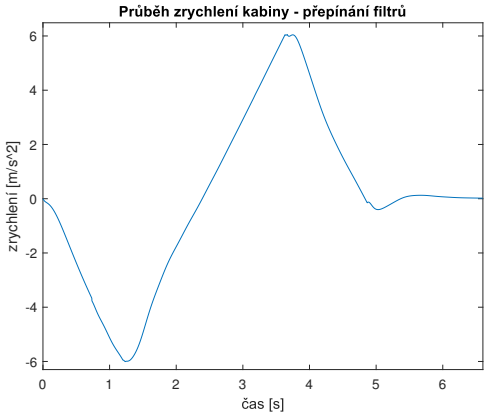
není zde velký problém. Je třeba ale zmínit, že i při těchto nulových odchylkách neuvažuji jistou konstantní odchylku od koncové polohy, ale čas, kdy se přestala odchylka jako taková měnit. Ta celková odchylka vznikla vlivem filtrů použitých v přímé vazbě a bylo by třeba využít ještě polohovou smyčku, nebo pomalý dojezd na koncový spínač, což je řešení v této práci již vícekrát zmiňované. Zároveň někde je také brána hranice, kdy odchylka již spadá do stanovené meze, ale ještě se dále zmenšuje k nule.

Tabulka 2: Zpoždění dojezdu kabiny při různých přístupech

	čas dojezdu podle generátoru	**skutečný čas dojezdu**	**rozdíl**
Robustní regulátor	24.248 s	26.596	2.348 s
Přepínání filtrů s vysledováním	24.248 s	24.248 s	0 s
Linearizace podél trajektorie při vhodné hmotnosti	24.248 s	24.248 s	0 s
Linearizace podél trajektorie při nevhodné hmotnosti	24.248 s	24.427 s	0.179 s
FIR filtr	24.248 s	25.040 s	0.792 s

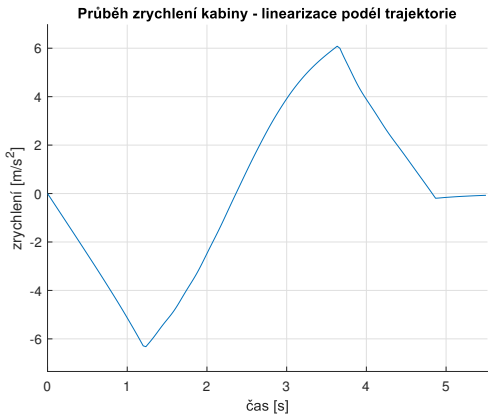


(a) jeden robustní regulátor

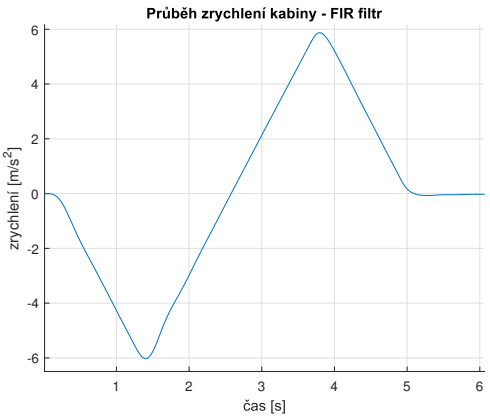


(b) Přepínání přímovazebních filtrů

Obrázek 88: Porovnání průběhů zrychlení kabiny při zvýšení rychlosti pro různé přístupy



(a) linearizace podél trajektorie



(b) FIR filtr

Obrázek 89: Porovnání průběhů zrychlení kabiny při zvýšení rychlosti pro různé přístupy 2

Co je tedy skutečným závěrem a který z přístupů by byl nejvhodnější? Nelze to říci tak snadno, závisí na prioritách dané aplikace. Obecně je však nejuniverzálnějším řešením FIR filtr, který poskytuje přiměřené výsledky ve všech hodnotících kategoriích (včetně snadné praktické implementace a výpočetní náročnosti). Linearizace podél trajektorie zajišťuje nejlepší sledování reference, ale kromě výpočetní náročnosti má i spoustu dalších nevýhod jako je například velká citlivost na neurčitosti a obecně nepřesnosti v modelu. Přijatelnější variantou je tedy v tomto směru gain-scheduling přímovazebních filtrů. Ani obyčejný robustní PI regulátor se nedá zavrhnout, protože funguje poměrně dobře a je dosti robustní v různých kritériích, ale pokud klademe vyšší a konkrétnější požadavky ohledně rychlosti dojezdu, sledování reference atd. je potřeba použít diskutovaná přímovazební rozšíření.

6.1 Poznámky k softwareovému řešení

Všechny výše zmíněné přístupy je možné si vyzkoušet a odsimulovat. Nachází se společně ve složce `primovazebni_final`, protože využívají množství stejných funkcí (umístěných ve složce `utils`) a i modely jsou velmi podobné jen s rozdílem implementace přímovazebního filtru.

Prvním případem je samotný robustní regulátor, který je nasimulován v souboru `robustni_regulator.slx`. Pro tuto simulaci potřebujeme mít načtené jen základní parametry systému, tudíž pro její zkompilování postačí spustit libovolný z ostatních skriptů (doporučuji `primovazebni_reg.m`). Tento soubor slouží rovněž k odsimulování gain-schedulingu přímovazebních filtrů, tyto složitější funkce jsem objasnil v příslušné kapitole v části věnující se softwareovému řešení. Po spuštění souboru `FIR.m` lze odsimulovat `primovazebni_FIR.slx`, což je simulace s navrženým FIR filtrem. Po spuštění skriptu dojde k odsimulování automaticky a jsou vykresleny odchylky od polohy a rychlosti kabiny a také průběh zrychlení kabiny. Pokud však chcete vidět více, je nutné jít do příslušného `.slx` souboru a tam jsou připravené *scopy* pro všechny zajímavé signály a je možné zde porovnávat i kritéria hladkosti zmiňovaná v tomto shrnutí.

Nakonec je zde soubor `linearizace_podel_trajekotrie.m`. Zde probíhá poměrně náročná operace vytvoření polynomů na základě spočtené trajektorie a následné dosazení pro daný čas s daným vzorkováním. Standardně se tedy pohybujeme kolem 25 000 matic, ze kterých se pak vytváří *state-spacy* a z těch přenosy pro filtry. Tato operace je zdlouhavá a proto jsem uložil již napočtené matice A pro příklad přejezdu mezi 3. a 8. patrem. Pokud by si někdo přál vyzkoušet simulaci v jiných patrech, je třeba ve složce `utils` smazat stávající soubor A_p a libovolně přenastavit parametry. Pak proběhne linearizace a vytvoří se nová struktura A_p, kam se uloží průběh matice A v čase. Stejným způsobem jsou uložené i sady přenosů `P_inv_kabina.mat` a `P_motor.mat` – ty by bylo také potřeba smazat. Důvod je zřejmý z následujícího kódu:

```
try
    load("P_inv_kabina.mat");
    load("P_motor.mat");
catch
    for i=1:length(t_)
        sysik = tf(LKT(:, :, i));
        P_motor(i) = minreal(sysik(6,1));
        P_inv_kabina(i) = minreal(inv(sysik(5,1)));
    end
end
```

Výpočet však trvá skutečně dlouho a je potřeba počítat i s dlouhým časem simulace. Pokud však chcete zkoušet například reakci na neurčitost ve hmotnosti, tak nechcete předělávat tyto matice, které jsou nějak navrženy za neznalosti přesné hmotnosti. Toto nastavení fyzikálních parametrů pro simualci je až dole ke konci kódu.

Rovněž je na ukázce kódu hezky vidět, že se vytvoří mnoho (řádově pravděpodobně několik tisíc, záleží na periodě vzorkování a délce času simulace) stavových modelů a z nich filtrů, které jsou následně použity k řízení a v každém časovém okamžiku jsou změněny – kdybychom šli až do spojitého času, bylo by jich nekonečně mnoho a získali bychom tedy obyčenjou po částech spojitou funkci, o které jsme v teoretické části mluvili.

7 Reálný model

Kvůli náročnějšímu teoretickému základu a problémům, které bylo třeba v průběhu práce vyřešit a které nebyly očekávány, bohužel nezbyl dostatek času na nasazení řešení na reálný model. V následující části práce se zabývám právě přípravou všeho potřebného pro reálné použití – jak návrhem řídícího systému v REXYGENu, tak vývojem nadřazené aplikace, která dokáže jednoduchým způsobem ovládat řízení v REXYGENu. Tuto aplikaci si můžeme představit i jako jakýsi ovládací panel pro cestující, protože je zde možnost jen zadat požadované patro a výtah jede – dojde ke všem zmiňovaným krokům jako je například výpočet trajektorie atd. Kromě toho poskytuje aplikace vizualizaci jedoucího výtahu a zobrazuje důležité grafy (polohu, průběh zrychlení kabiny, odchylku polohy kabiny od požadované hodnoty a možnost zachytit kritický moment v průběhu zrychlení). Poslední vrstvou je pak navržený algoritmus a provedená simulace logiky přejezdů kabiny mezi patr podle volání – tedy simulace front v jednotlivých patrech a co nejefektivnější rozvržení jízdy. Tím je vlastně výtah jako takový kompletně navržen od základu, kterým je regulace a odstranění kmitů, až po nezbytné součásti pro praktické použití, čímž je právě ovládání a zmíněná logika.

Proč tedy nebylo dost času na reálné testování, když je vše připravené? Je to tím, že stand má nakonec jinou strukturu, dynamiku. Není to tříhmotový výtah na laně, jaký jsme celou dobou uvažovali, ale kabina je umístěna na pásu, který vede kolem dokola a hnaný je motorem v horní části, protizávaží zde vůbec není. Prvním problémem by tedy bylo vytvoření nového matematického modelu a zjištění, zda dostatečně odpovídá realitě. Sem by spadala experimentální identifikace, kdy by bylo třeba naměřit přechodové děje a podle znalosti matematického modelu estimovat data pomocí přenosu v určitém tvaru. Na základě těchto přenosů by se pak naladil nový robustní regulátor, k čemuž bychom opět potřebovali nalézt první rezonanční frekvenci, kterou bychom chtěli potlačit pomocí notch filtru. Zbytek práce by již provedly připravené skripty, které jsou automatické, a tak není problém na základě přenosů navrhnout přímovazební filtry, linearizaci podél trajektorie atd. Poměrně dlouho by však trvalo vytvořit vhodný matematický model, naměřit data a na základě nich postavit postavit model systému, který bychom mohli použít k návrhu řízení. Z tohoto důvodu jsem již nestihl zařadit tuto část do své práce a pokračovat v ní budu až později.

8 Simulace a vizualizace

Úvod

V této kapitole se budu zabývat dvěma úlohami, které jsou od základního tématu této práce trochu stranou, ale i přesto si myslím, že k dané tématice patří a jsou vhodným zpestřením. Jedná se o dvě komponenty – tou první je simulace řídicího algoritmu, druhou vizualizace a interaktivní aplikace pro zobrazování a ovládání dat v REXYGENu.

Simulace je napsaná v jazyce Java, konkrétně za použití knihovny *JavaSimulation* [12]. Zde se nejedná již o řízení ve smyslu regulace a odstraňování kmitů, ale spíše o algoritmické řešení toho, jak bude výtah vybírat, kam pojede, jak, kde zastaví, kolik lidí smí nastoupit a tak dále. Tato simulace představuje řešení s vyobrazeným časem, ze kterého lze vyčíst kvalitu systému, dobu, jakou musí lidé na výtah čekat a tak dále. Zkrátka je to důležitá součást reálného nasazení a myslím si, že k takovéto práci rovněž patří.

Vizualizace je oproti tomu aplikací, která komunikuje s programem REXYGEN pomocí REST API. Umožňuje mi tak ovládat jen uživatelsky důležité parametry v programu REXYGEN (takže v samotném řešení nemůže nikdo nic rozbít) a zároveň vizualizovat názorně výsledná data, kterými jsou informace o poloze a zrychlení kabiny, zátěže či o odchylkách od požadované hodnoty. Takovýmto způsobem je tedy možné se připojit jak k řídícímu systému reálného zařízení a skrz aplikaci je ovládat (to mi v podstatě může simulovat skutečně knoflíky, kterými reálný uživatel komunikuje s reálným výtahem), ale rovněž můžeme jen příjemně ovládat a hezky zobrazovat simulaci v REXYGENu vytvořenou.

8.1 Simulace řídícího algoritmu výtahu

8.1.1 Algoritmus

Zvolil jsem poměrně jednoduchý ale účinný algoritmus pro řízení výtahu někdy také nazývaný SCAN algoritmus. Jeho podrobnější postup je popsaný v následujících bodech. Počítá s tím, že existují dva knoflíky na přivolání výtahu, kterým osoba dává najevo, kterým chce jet směrem. Samozřejmě algoritmus trochu rozhází, když někdo ohlásí požadovaný směr jízdy nahoru a nakonec chce jet dolů – především dojde ke zdržení. Já ve svém řešení však uvažuji disciplinovaného uživatele a neošetřuji takovéto případy.

- Na počátku všeho je kabina dole v prvním patře
- Jakmile je vytvořen požadavek, jede jej splnit
- Během toho už mohou být vytvářeny nové požadavky, které registruje a třídí do front podle toho, kterým chtějí jet lidé směrem
- Jakmile kabina míjí patro, kde je registrovaný člověk, který chce jet stejným směrem, jakým se pohybuje kabina, zastaví
- Jakmile dojdou požadavky na daný směr jízdy, proběhne pokus o zjištění, jestli někde dál tímto směrem není někdo, kdo by chtěl jet opačným směrem
- Pokud ano, dojede se pro něj a směr jízdy se změní
- Pokud ne, směr jízdy se mění okamžitě
- Pokud nejsou již žádní lidé, kteří by někam jeli, výtah zůstane, kde je, a tato pozice je novou výchozí polohou.

8.1.2 Popis programu

Program je napsán v jazyce Java za použití knihovny *JavaSimulation*, která přebírá množství rysů z dřívější *Simuly*. Jednou z nejdůležitějších vlastností je právě oddědění třídy *Process*, která obsahuje metodu *actions()*. Tato metoda může být volána a tím je zaktivován proces, která daná třída má vykonávat. Většinou se pak používá nekonečná smyčka, jelikož je umožněno tuto akci rovněž přerušit / uspat pomocí příkazu *passivate()*. Proces tak může být neustále volán i zastavován.

Celkově je kód rozdělen do více tříd kvůli přehlednosti – jsou zde jednotlivá *podlaží* – *Floor*, do kterých se generují *lidé* – *Person* a ještě se třídí do front podle toho, jakým chtějí jet směrem *queueUP, queueDOWN*. Toto již velmi zjednodušuje orientaci při vykonávání algoritmu ve třídě *Cabin*, i když je pak kód jako takový možná o něco delší, než by být musel. Zdálo se mi to však jako přiměřená cena za pochopitelnost programu. Následuje popis jednotlivých tříd, kde je lépe rozebráno, jak vše funguje a jak spolu jednotlivé části komunikují.

8.1.3 class Elevator

Tato část kódu je spouštěcím a vytvářecím základem. Obsahuje metodu *main*, která je v jazyku Java nutná pro spuštění. Zde je asi nejdůležitější zmínit, že právě v metodě *main* je možné při vytváření objektu zvolit počet pater a omezení na kapacitu kabiny.

Následně je aktivován generátor, který zajistí již připravení celého modelu, který tato třída reprezentuje.

8.1.4 class Generator

Třída vytvářející prakticky model budovy. Nejprve vytvoří zadaný počet pater s odvoláním na třídu *Floor*, která je popsána níže. Následně vytvoří instanci kabiny. Pak už je jeho práce jen neustále generovat náhodně do různých pater lidi s různými požadavky s náhodným časovým intervalem. Přicházet mohou i skupinky lidí, což zařizuje parametr *countOfPeople*.

8.1.5 class Floor

Třída *Floor* obsahuje dvě fronty – *queueUP, queueDOWN*. V každém podlaží jsou tedy aktuálně čekající lidé roztříděni na ty, kteří chtějí jet nahoru, a ty, kteří chtějí dolů.

Kromě toho je zde metoda *person()*, která do daného patra vygeneruje nově příchozího s náhodným požadavkem. Tato metoda je právě cyklicky volána v *generátoru*.

8.1.6 class Person

Tato třída generuje do zadaného patra člověka. Kromě toho, že mu udá nějaký požadavek, také jej rovnou zařadí do fronty podle toho, jestli chce jet nahoru anebo dolů. Je zde také ošetřeno, aby někdo nechtěl jet do toho samého patra, ve kterém se nachází, nebo že když je nahoře, výš už jet nemůže. To je ale, řekl bych, poměrně dobře zdokumentované přímo v kódu.

8.1.7 class Cabin

Nakonec je zde zřejmě nejdůležitější třída, která se stará o logiku výtahu jako takového. Zde je implementované rozhodnutí o tom, kam kabina pojede, zda a kde zastaví, jak se zachová. Algoritmus byl již popsaný na začátku, tento kód je tedy spíše o implementačních detailech. Opět je zdokumentovaný, ale to nejdůležitější ještě zdůrazním:

Kabina se rozjede daným směrem. To znamená, že prohledá stavový prostor směrem nahoru / dolů. Pokud je tam někdo, kdo chce jet stejným směrem jako jede kabina, tak tam skutečně jede (rozumějme simulace se pozastaví na čas potřebný pro přejetí x pater, pak se otevírají dveře, nastupuje se a vystupuje se, zavírají se dveře, tiskne se výpis). Takhle to jde dál, neustále se ukládá, kde chtějí lidé vystupovat a podle toho se staví. Pokud už nikdo, kdo chce jet daným směrem, není, zkusí se, jestli tam není někdo, kdo by chtěl jet opačným směrem (aby tam nezůstal navždy). Pokud ano, zajede se pro něj a pak se změní směr, pokud ne, změní se směr rovnou.

Toto ukážu spíše na příkladu:

Kabina jede nahoru, vysadila posledního člověka jedoucího směrem nahoru například v 6. patře. Teď by měla změnit směr, ale ještě se musí přesvědčit, jestli výše není někdo, kdo by chtěl jet dolů, protože pak by ještě směr neměnila. To je proto, aby někdo nahoře nezůstal zapomenutý.

Tento proces se točí v nekonečné smyčce a je uspáván, pokud nemá práci, a naopak buzen, pokud se dostaví lidé.

Ještě je možná dobré upozornit na maličkost, která zde hodně znamená z hlediska simulace jako takové – všechny čekací časy jsou zde roztříděné podle toho, jakou mají funkci. Je tu čas na otevírání a zavírání dveří, v generátoru (viz výše) náhodný čas, se kterým se generují lidé, a ještě čas na vystupování a nastupování, ve kterém je také zohledněna určitá část náhody.

8.1.8 Výsledky

Výsledkem je program umožňující simulaci přepravy osob ve výškové budově s libovolným počtem pater pomocí jednoho výtahu. Výpis je prováděn do konzole a vypadá například takovýmto způsobem:

```
Cabin is in floor 1 with 0 people at time 0.0 s
New person: 8 ->   up ->  9
Cabin arrived from floor 8 with 1 people at time 76 s
New person: 6 ->  down -> 4
New person: 1 ->   up ->  4
New person: 6 ->   up ->  9
Cabin arrived from floor 9 with 0 people at time 93 s
Cabin change the direction
Cabin arrived from floor 6 with 1 people at time 130 s
Cabin arrived from floor 4 with 0 people at time 157 s
Cabin change the direction
Cabin arrived from floor 1 with 1 people at time 194 s
New person: 6 ->   up ->  9
New person: 4 ->  down -> 3
New person: 6 ->   up ->  7
Cabin arrived from floor 4 with 0 people at time 231 s
New person: 5 ->   up ->  9
New person: 5 ->  down -> 2
New person: 6 ->  down -> 2
New person: 3 ->  down -> 2
Cabin arrived from floor 5 with 1 people at time 248 s
Cabin arrived from floor 6 with 4 people at time 265 s
New person: 8 ->   up ->  10
New person: 10 ->  down -> 3
New person: 2 ->   up ->  6
New person: 9 ->   up ->  10
New person: 3 ->   up ->  6
New person: 6 ->  down -> 1
Cabin arrived from floor 7 with 3 people at time 282 s
Cabin arrived from floor 8 with 4 people at time 299 s
Cabin arrived from floor 9 with 2 people at time 316 s
Cabin arrived from floor 10 with 0 people at time 332 s
Cabin change the direction
Cabin arrived from floor 10 with 1 people at time 339 s
Cabin arrived from floor 6 with 3 people at time 385 s
Cabin arrived from floor 5 with 4 people at time 402 s
Cabin arrived from floor 4 with 5 people at time 419 s
Cabin arrived from floor 3 with 4 people at time 435 s
Cabin arrived from floor 2 with 1 people at time 453 s
Cabin arrived from floor 1 with 0 people at time 470 s
Cabin change the direction
Cabin arrived from floor 2 with 1 people at time 488 s
Cabin arrived from floor 3 with 2 people at time 505 s
Cabin arrived from floor 6 with 0 people at time 542 s
```

Obrázek 90: Výpis v konzoli + barevné vysvětlení

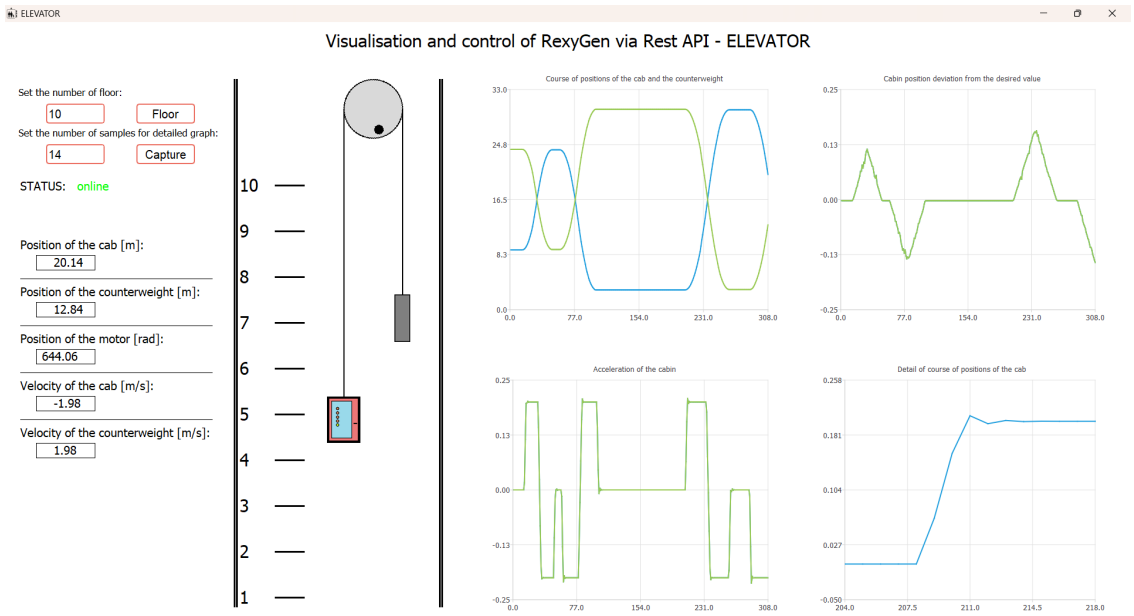
Na samém začátku je výtah v 1. patře. Vždy se vypisuje, kde právě stojí, tyto výpisy jsou přerušené jen přicházejícími lidmi (výpis kde jsou, jakým směrem a do jakého patra chtějí jet), které postupně obsluhuje pokud možno v rozumném pořadí, ale zejména tak, jak mu to vychází cestou dolu nebo nahoru. Proto se občas stane, že i když někdo výtah zavolá dřív, počká si o trochu déle než ten, kdo jej zavolal později. Z celkové efektivity přepravy je to takto ale lepší. Princip fungování znázorněný výpisem jsem se pokusil barevně vyznačit (viz obrázek 90)). Zde je zvolená kapacita schválně 6 lidí. Schválně nevypisuji vše (jako otevírání dveří, zavírání, vystupování a nastupování lidí) kvůli přehlednosti výpisu, ale všechny tyto aspekty jsou v kódu zohledněny a můžeme je vidět v podstatě schované ve výpisech časů, kdy výtah je v daných patrech.

8.2 Vizualizace modelu v REXYGENu

8.2.1 Prostředí aplikace

Na obrázku 91 lze vidět prostředí vyvinuté aplikace. Jsou zde tři hlavní části:

- Ovládací panel – zde se nachází komponenty nazvané SpinBox, do kterých je možné zadat libovolné číslo a jsou omezené příslušnou hranicí (například pro podlaží 1 až 10). Kromě podlaží je zde ještě tlačítko pro zachycení posledních x vzorků polohy kabiny (pokud chceme vidět lépe vibrace kabiny). Tento panel se v budoucnosti pravděpodobně ještě rozroste podle toho, co vše budu potřebovat nastavovat (očekávám, že by to nemusela být jen požadovaná hodnota, ale například bych mohl měnit parametry regulátoru a zjišťovat tak rozdíly mezi nimi, přidávat či ubírat přímovazební kompenzace, uměle působit poruchami pro zjištění schopnosti regulátoru je odregulovat atd.). Stejně tak se zde nachází *status*, který říká, zda je aplikace připojená k REXYGENu či nikoliv. Následuje ještě výpis aktuálních hodnot.
- Kompletní vizualizace modelu – lze vidět otáčející se špulku motoru a aktuální polohu kabiny a protizátěže, sledovat, jak jezdí atd. Tato část je zejména přehledová, jsou zde naznačená patra (po třech metrech), model je však příliš malý na to, abychom dobře viděli vibrace.
- Grafy – první graf, pokud je status *online*, zobrazuje průběh poloh kabiny i protizátěže. Druhý graf ukazuje odchylku polohy kabiny od požadované hodnoty. Třetí zobrazuje průběh zrychlení kabiny a čtvrtý se váže k tlačítku *Capture* a zobrazí zadaný počet vzorků, tedy námi zvolený detail právě z grafu zrychlení – je tedy možné zachytit kritické části průběhu.



Obrázek 91: Vzhled aplikace

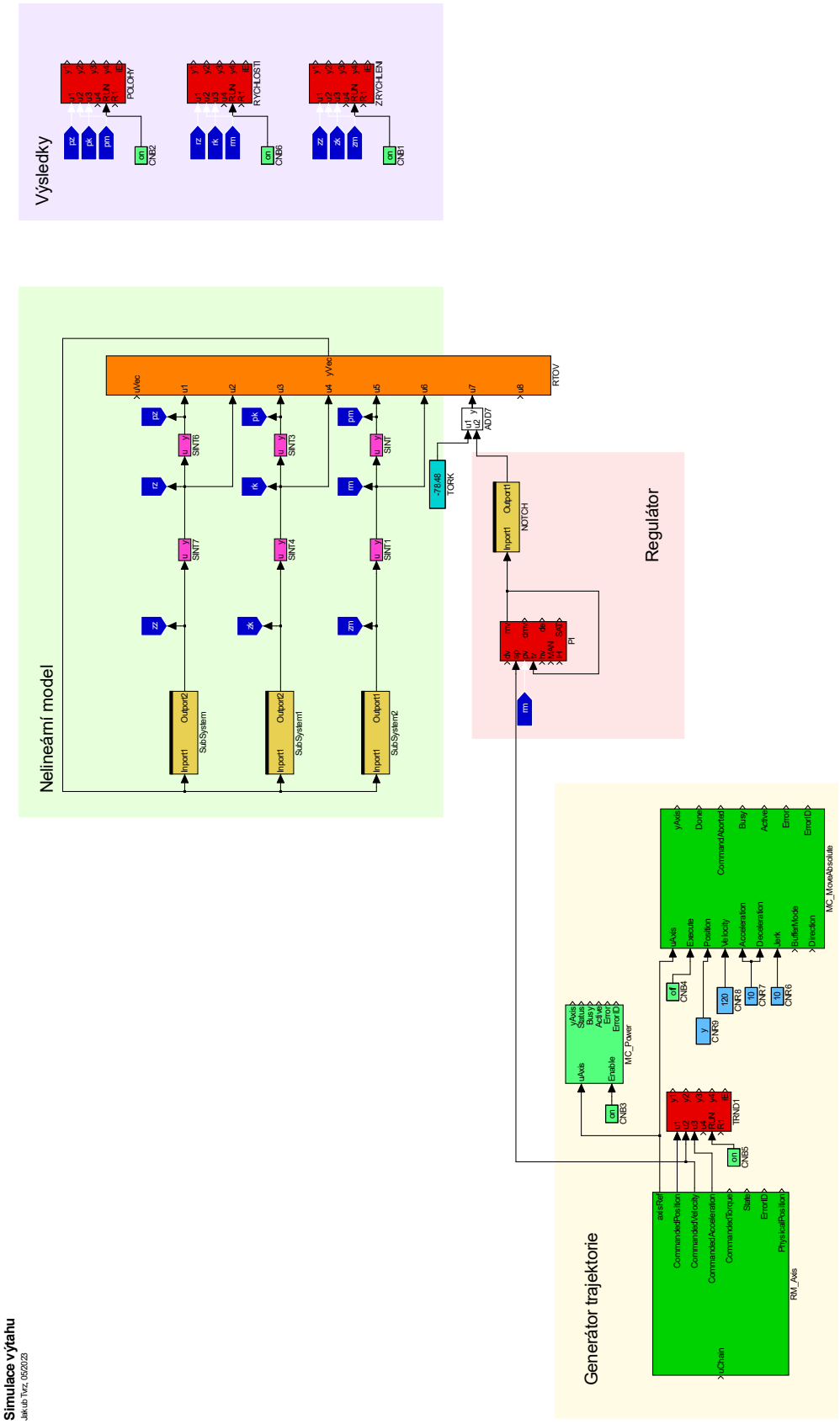
8.2.2 Model v REXYGENu

Vzhledem k náročnosti a průtahům teoretické práce nakonec nedošlo k testování na reálném standu. Avšak i tak jsem řídicí systém v REXYGENu připravil (viz obrázek 92) a bylo by snadné jej ke skutečnému zařízení připojit. Právě ten reálný systém zde reprezentuje *nelineární model* vytvořený stejným způsobem, jako bylo na samém začátku ukázáno v simulinku. Jedná se o namodelování diferenciálních rovnic popisujících dynamiku výtahu pomocí integrátorů. V jednotlivých subsystémech je (o trochu složitěji) ručně vymodelováno právě působení jednotlivých stavů na zrychlení dané hmoty tříhmotového systému, přičemž uvažujeme i tření v motoru, ve výtahové šachtě apod.

Další částí je *generátor trajektorie*, který bylo prozměnu možné navrhnout podstatně jednodušeji. V REXYGENu jsou již připravené bloky, které k tomuto účelu slouží – je třeba jen zadefinovat parametry virtuální osy, která simuluje tu skutečnou (motor) a následně připojit bloček, ve kterém probíhají výpočty, které byly zmíněny v kapitole o generátoru trajektorie, který jsem programoval ručně. Rovněž se zde zadávají maximální omezení na jerk, zrychlení a rychlost.

Poslední částí řízení je zřejmě ta nejdůležitější – samotný regulátor. Na obrázku je vidět implementace robustního PI regulátoru s notch filtrem, jehož zásah je přičítán k momentu, který je potřeba pro vytvoření ustáleného stavu, ve kterém je možné provést linearizaci a pomocí ní navrhnout regulátor.

Ještě je zde sekce *výsledky*, která je pro aplikaci velmi důležitá – ze zobrazovacích bločků se totiž berou hodnoty pro zobrazení a vizualizaci. Aplikace sama o sobě zapisuje pak jen do požadavků generátoru trajektorie – zadává požadovanou polohu (přepočtenou z pater na polohu motoru v radiánech kvůli řízené) a přivádí náběžnou hranu potřebnou pro spuštění generátoru trajektorie a tedy postupné změny požadované hodnoty. Při nasazení na reálný systém by se tedy jen nahradil *nelineární systém* příslušnými "senzory"– zkrátka výstupy, které je možné měřit. Dostávali bychom tak především rychlost motoru (případně polohu), která by se přímo vedla do regulátoru. Pak je zde logicky jeden vstup, kterým je moment motoru. Ten by se jen převáděl na příslušné napětí.



Obrázek 92: Základní schéma v prostředí REXYGENu

Program sám o sobě se skládá z několika částí, které spolu logicky komunikují. Jejich funkčnost se pokusím popsat v rámci jednotlivých tříd.

8.2.3 MyWidget

Obsahuje protected metodu `void paintEvent(QPaintEvent* event)`, která je minimálně z hlediska uživatele tím nejdůležitějším – vykresluje všechny animace na obrazovku. Pomocí instance `QPainter painter` je možné vykreslovat různé obrazce a tvary, nastavovat barvy atd., ale také je možné vykreslovat takzvané pixelové mapy – tedy obrázky. Toho jsem nakonec využil, protože mi to přišlo snazší, a hlavně přehlednější v kódu, než to postupně celé vykreslovat jen pomocí čar a obdélníků. Zároveň si myslím, že i programu tím nějakou tu práci ušetřím. Také je možné použít propracovanější obrázky a vizualizaci provést například i pomocí obrázků vymodelovaný v nějakém CAD programu. Já jsem ale zůstal u svého původního jednoduchého návrhu složeného ze základních geometrických tvarů, protože mi přišel poměrně čistý a jednoduchý.

Pomocí této metody tedy vykresluji kabinu, protizávaží, motor, měřítko a zkrátka vše potřebné k animaci a vizualizaci jako takové. Také jsem sem zakomponoval aktualizování grafů (přidávání hodnot).

Naopak pro funkčnost je velmi důležitý konstruktor:
`MyWidget::MyWidget():QWidget()`, ve kterém se nastaví a spustí `timer`, který určuje, s jakou periodou se budou provádět určité úkony. Pro nás to znamená, jak často budu načítat data z REXYGENu a vykreslovat je. Já mám zvolenou periodu 10ms, která mi přijde v tuto chvíli dostačující. Konkrétně jsou zde nastaveny cookies všem pManagerům a jejich propojení s metodami, které se mají volat při vyčtení jsonu. To, že je to provedené právě zde v konstruktoru, je poměrně důležité z hlediska paměti. Vzhledem k tomu, že vyčítám data stokrát za sekundu, se paměť rychle zahltí, pokud vytvářím neustále nová spojení. Následuje ukázka provedení:

```
1     cJar->insertCookie(QNetworkCookie("token", "YWRtaW46"));
2     pManagerPZ.setCookieJar(cJar);
3     connect(&pManagerPZ, &QNetworkAccessManager::finished,
    this,
4       &MyWidget::onReplyPolohaZateze);
```

`Timer` je propojen s metodou `void MyWidget::onTimeout()`,která je volaná v podstatě jako „při přerušení“. Zde se provede vyčtení hodnot přes REST API a pak se zavolá `update()`, který překreslí obrazovku. Vyčtení hodnot probíhá tímto způsobem:

```
1     requestPolohaZateze("http://127.0.0.1:8008/api/tasks/
    myproject_task/POLOHY:
2     u1?data&mime=application/json&token=YWRtaW46");
```

kde v odkazu je uvedeno, k jakému bločku z REXYGENu se snažím dostat.

Následuje tedy provedení metody `requestPolohaZateze(QString title)`, která se pokusí připojit k danému odkazu (musí například právě obsahovat cookies, která v sobě uchovává informaci o přihlášení) a uložit data. Jestliže je tato akce skončená, volá se metoda `onReplyPolohaZateze(QNetworkReply* reply)`, jak bylo již zmíněno v konstruktoru. Ta dostane odpověď a zpracuje ji – rozebere json a vezme si hodnotu, kterou potřebuje znát. Tuto hodnotu uloží do globální proměnné, pomocí které se vykresluje objekt na danou pozici.

8.2.4 Vypis

Tato třída je opět typu *QWidget*. Stará se o výpis aktuálních hodnot. Kromě toho kontroluje připojení k REXYGENu a vypisuje status pomocí této metody: `widget->connection == true`.

8.2.5 Graf

Opět třída typu *QWidget*. Stará se o vykreslování nebo spíše zanášení hodnot do grafů. Obsahuje tři důležité metody:

`void pridejHodnoty(double hodnota1, double hodnota2);` Tato metoda zadá hodnotu polohy kabiny a záteže do dvou `QLineSeries`. Rovněž je přiřadí do `QChart` a vytvoří aktualizované a zobrazitelné `QChartView`.

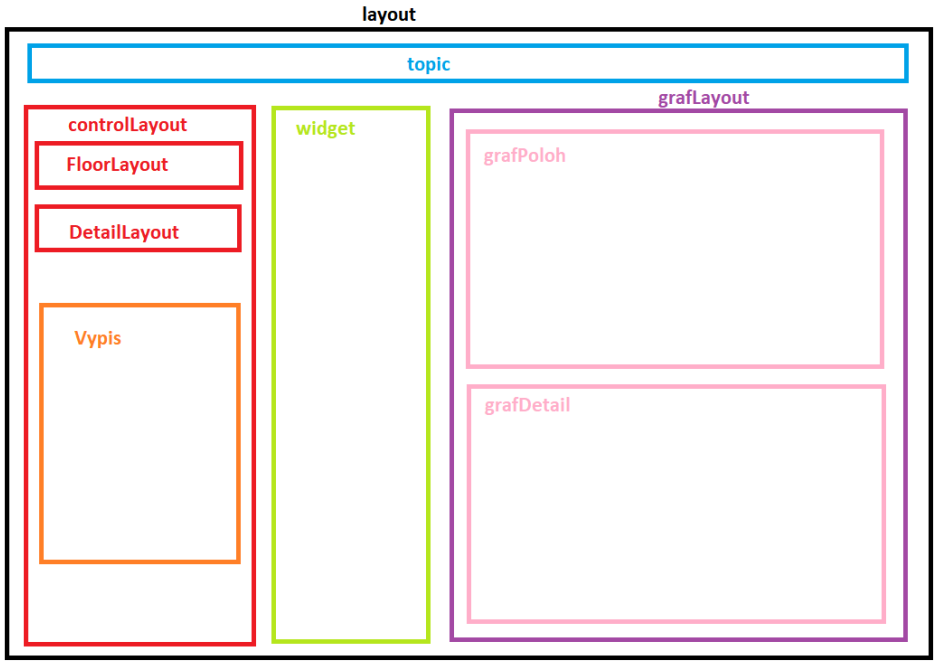
`void jenPridej(double hodnota);` Tato metoda skutečně jen přidá hodnotu polohy kabiny do `QLineSeries`. Neaktualizuje grafy ani nic podobného, jen sbírá data.

`void vzorky(int pocet);` Tato metoda vezme vzorky polohy kabiny a zobrazí posledních x zadaných v parametru *pocet*. Zde také dojde k aktualizování grafu.

Grafy vygenerované touto třídou mají nastavenou nějakou minimální velikost, ale jinak jsou responzivní. Aplikace se automaticky zobrazí v nějaké základní minimální velikosti, ale je možné ji libovolně roztahovat a zvětšovat, grafy se tomu přizpůsobují svou velikostí, ostatní komponenty zůstávají stejné.

8.2.6 App

Třída App obsahuje mimo jiné instanci výše rozebrané třídy MyWidget: `widget = new MyWidget();`. Vše je uspořádané pomocí layoutů, jak je znázorněno na následujícím schématu (viz obrázek 93):



Obrázek 93: Rozložení aplikace

V konstruktoru jsou opět vytvořená spojení pro akční prvky – spinBoxy a tlačítka. U spinBoxů se neděje nic jiného než že se hodnota, kterou do nich zadáme, zapisuje opět

do nějaké globální proměnné. Při stisknutí tlačítka je to zajímavější – zde chceme již onu hodnotu ze spinBoxu skutečně zapsat do REXYGENu. To probíhá pomocí metody post. Zde je však ještě potřeba nastavit hlavičky a opět cookies. Tvar, v jakém je možné jsonu předat hodnotu, jsem vyčetl přímo na stránce s api a pomáhal jsem si programem Fidler, který umožňuje sledovat komunikaci (něco jako WireShark, ale jednodušší), abych viděl, co se doopravdy odeslalo apod. Konkrétní provedení je vidět pak v přiloženém kódu, zde je ukázka vystavení požadavku:

```
1     QNetworkRequest req2(QString("http://127.0.0.1:8008/api/
    tasks/myproject_
2     task/CNR:ycn"));
3         req2.setHeader(QNetworkRequest::KnownHeaders::
    ContentTypeHeader,
4         "application/x-www-form-urlencoded");
5         req2.setRawHeader("Cookie", "token=YWRtaW46");
6         pManager.post(req2, payload);
```

Také je ještě v konstruktoru nastaveno rozložení, ve kterém jsou již schované všechny ostatní `layouty setLayout(layout);`. Tato třída se tedy stará již o sestavení celkového okna aplikace a jen do sebe přibírá výše zmíněný widgety, které se starají o samotnou animaci a časově proměnná data. A také slouží ke komunikaci uživatele s REXYGENem.

8.2.7 main

Zde je již jen vytvořená instance třídy App a vytvořen spustitelný exec soubor, který rozjede okno s celou aplikací. Dále jsem zde nastavoval detaily zobrazení jako například, aby se aplikace zobrazovala automaticky v maximální velikosti, nadefinoval jsem tu styly tlačítek a spinboxů, layoutů, pozadí, přidal ikonu atd.

8.2.8 Instrukce k použití

Nejprve je samozřejmě potřeba spustit `q make` nad soubory ve složce aplikace. Tím se vytvoří již spustitelné soubory. Dále musíme mít spuštěný (stažený a zkompilovaný) model v REXYGENU – ten je ve složce model. Poté je možné spustit aplikaci, která již dokáže s REXYGENem navázat spojení (lze zkontrolovat, jestli je STATUS online).

V tuto chvíli je možné používat tlačítka a spinBoxy. Jak je popsáno výše i u tlačítka, první s nápisem *Floor* slouží k zadání patra, do kterého chceme, aby výtah dojel. Další tlačítko zachytí posledních pár zadaných vzorků grafu do druhého grafu. Tím tedy můžeme zobrazit detail průběhu polohy kabiny, kde mohou jít vidět případné vibrace apod.

Ukázka chování aplikace a jejího použití včetně spuštění a zkompilování modelu v REXYGENu je vidět v přiloženém videu `ukazka_aplikace.wmv`.

Shrnutí

Tyto dvě komponenty byly zpracovány navrch a stranou od zadání bakalářské práce základně v rámci jiných předmětů, ale nadále jsem je rozšiřoval, aby co nejlépe vyhovovaly cílenému využití. Celkově však rozšiřují určitým způsobem možnosti a aplikovatelnost této práce.

Cílem práce na vizualizaci bylo připravit si na pohled hezkou a uživatelsky přívětivou aplikaci pro budoucí použití. Ještě počítám s tím, že ji budu pro další potřeby dodělávat a rozšiřovat (například by se dalo přidat přepínání mezi jednotlivými regulátory, změna parametrů motoru pro generátor trajektorie atd., na druhou stranu by byl ztracen půvab jednoduchosti a přímosti aplikace). Každopádně je vyřešený hlavní problém – komunikace s REXYGENem pomocí REST API, a to jak vyčítání, tak zapisování dat. Také je připraveno základní rozložení, do kterého se dají ještě přidávat tlačítka a různé ovládací prvky, animaci považuji za kompletní.

Závěr

Shrnutí a porovnání jednotlivých výsledků byla provedena vždy v příslušných kapitolách či po úsecích práce, kdy se to zdálo vhodné. Na úplný závěr by tak bylo dobré pouze říct, že jsem úspěšně navrhl kompletní systém výtahu od nejnižších vrstev jako je regulace a přímovazební filtry na řízení či jako tvarování reference, až po ty nejnadřazenější, jako je generátor trajektorie a nad ním ještě ovládací algoritmus a logika pohybu kabiny výtahu.

Byl úspěšně navržen a otestován robustní PI regulátor s notch filtrem, k němuž jsem vyvýjel přímovazební řízení. Přímovazební filtry kvůli komplikované dynamice tříhmotového systému získaly poněkud složitější strukturu, ale díky generátoru trajektorie bylo možné použít derivace k *předpovědi* a namodelovat tak i nekauzální přenosy nutné k ideálnímu vysledování. Při aplikaci těchto filtrů na nelineární model se ukázala potřeba jejich přepínání, a to bylo řešeno více přístupy, které byly vzájemně porovnány. Jedním z nich byl gain-scheduling – neboli přepínání filtrů podle dané veličiny. Tento přístup byl plně zautomatizován (stačí zadat počáteční a koncové patro, pak proběhne výpočet trajektorie, rozhodnutí o počtu přepínaných filtrů podle průběhu trajektorie atd.) a doplněn o způsob vysledování výstupů přepýnaných filtrů kvůli bezrázovému přepnutí. Další vyzkoušenou metodou bylo využití linearizace podél trajekotrie. Ukázalo se, že ač tato metoda dává výborné výsledky, je poměrně dost citlivá na neurčitosti v matematickém modelu a není příliš robustní, nehledě na její výpočetní náročnost (na druhou stranu, implementačně je již mnohem snazší než přepínání filtrů s vysledováním a také umožňuje odstranit notch filtr a zvýšit tak schopnost odregulování poruch). Posledním přímovazebním přístupem byla speciální metoda pro návrh FIR filtru, který se nakonec ukázal jako velmi výhodný díky odstranění notch filtru (zvětšení šířky pásma regulátoru) a přitom vytvoření hladkého přechodového děje zrychlení kabiny výtahu. Zároveň zanáší do systému dopravní zpoždění, které brání okamžitému vysledování reference.

Dále byl navržen řídicí systém v prostředí REXYGEN, který je možné snadno propojit s reálným hardwarem. Nad ním byla vyvinuta aplikace v C++ za pomoci Qt, která umožňuje snadné ovládání tohoto řídicího systému a poskytuje jednoduchou vizualizaci chování i náhled důležitých průběhů v grafech. Pro testování aplikace i řídicího systému jsem vymodeloval nelineární systém, na kterém je možné provádět experimenty.

Poslední částí byla nejvyšší vrstva, a to logika přejezdů výtahu mezi jednotlivými party tak, aby bylo obsloužení front v jednotlivých patrech co nejefektivnější. To bylo testováno pomocí navržené simulace s možným scénářem za pomoci knihovny JavaSimulation.

Reference

[1] Břetislav Kubeš and Jakub Tvrz. KKY/PAŘR5: Robustní řízení nelineárního modelu výtahu ve vícero rovnovážných bodech, KKY FAV ZČU, 2023.

[2] Michal Špirk. H_∞ pro elektromechanické soustavy: Uživatelská příručka, KKY FAV ZČU, 2022.

[3] Miloš Schlegel and Michal Brabec. Pidlab — the first advanced tool for analysis and hinf design of pid ... *Design of PID Controllers: H infinity Region Approach*.

[4] Marian Blejan and Robert Blejan. Mathematics for real-time s-curve profile generator. "HIDRAULICA" (No. 4/2020) Magazine of Hydraulics, Pneumatics, Tribology, Ecology, Sensorics, Mechatronics - ISSN 1453 – 7303:7–25, 12 2020.

[5] Kupčeková. *Výtah*. Wikimedia Foundation.

[6] Martin Goubej. Fundamental performance limitations in pid controlled elastic two-mass systems. In *2016 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 828–833, 07 2016.

[7] Milos Schlegel, Martin Goubej, and Jana Königsmarková. Active vibration control of two-mass flexible system using parametric jordan form assignment. *IFAC Proceedings Volumes*, 45:477–482, 12 2012.

[8] Michal Špirk. Automatické ladění regulátorů pro elektrické servopohony. Bachelor's thesis, KKY FAV ZČU, 2021.

[9] Benoit Clement and Gilles Duc. An interpolation method for gain-scheduling. volume 2, pages 1310 – 1315 vol.2, 02 2001.

[10] Kubeš Břetislav. Zpětnovazební řízení výtahů pro kompenzaci nežádoucích vibrací. pages 45–54, KKY FAV ZČU, 05 2023.

[11] Martin Goubej. Design report on intelligent motion control algorithms. pages 35–56, KKY FAV ZČU in the project iMOCO4.E (Intelligent Motion Control under Industry 4.E), 05 2022.

[12] Keld Helsgaun. Discrete event simulation in java. 06 2000.