

Nelineární Prediktivní Řízení

Bc. Martin Novák

Diplomová práce
2014-02-10



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně

Fakulta aplikované informatiky

akademický rok: 2013/2014

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Martin Novák**
Osobní číslo: **A12408**
Studijní program: **N3902 Inženýrská informatika**
Studijní obor: **Automatické řízení a informatika**
Forma studia: **prezenční**

Téma práce: **Nelineární prediktivní řízení**

Zásady pro vypracování:

1. Vypracujte literární rešerši zabývající se metodami nelineárního prediktivního řízení a srovnajte lineární a nelineární prediktivní řízení.
2. Popište časovou náročnost algoritmů nelineárního prediktivního řízení a posuďte jejich vhodnost pro řízení v reálném čase.
3. Realizujte vybrané algoritmy nelineárního prediktivního řízení v prostředí MATLAB/Simulink a proveďte jejich simulační ověření na silně nelineárním simulinkovém modelu soustavy tří nádrží, případně na jiných vhodných simulačních modelech.
4. Modifikujte použité algoritmy tak, aby mohly být použity pro řízení laboratorní soustavy v reálném čase, případně navrhnete vlastní algoritmus.
5. Ověřte nelineární prediktivní řízení na vhodné reálné laboratorní soustavě.

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: tištěná/elektronická

Seznam odborné literatury:

1. ALLGÖWER, Frank a Alex ZHENG. Nonlinear model predictive control. Basel: Birkhäuser Verlag, 2000, vii, 472 s. ISBN 3764362979.
2. BOBÁL, Vladimír. Adaptivní a prediktivní řízení. 1. vyd. Zlín: Univerzita Tomáše Bati ve Zlíně, 2008, 134 s. ISBN 978-80-7318-662-3.
3. CAMACHO, E a Carlos BORDONS. Model predictive control. 2nd ed. London: Springer, c2007, xxii, 405 s. ISBN 1-85233-694-3.
4. MAGNI, L. a R. SCATTOLINI. Annual Reviews in Control. 2004, vol. 28, issue 1, s. 1-11. DOI: 10.1016/j.arcontrol.2004.01.001. Dostupné z: <http://linkinghub.elsevier.com/retrieve/pii/S1367578804000021>
5. HUANG, Sunan, Tong Heng LEE a Kok Kiong TAN. Applied predictive control. London: Springer, 2002, xvii, 264 s. ISBN 1852333383.
6. ROSSITER, J. Model-based predictive control: a practical approach. Boca Raton: CRC Press, c2003, 318 s. ISBN 0849312914.

Vedoucí diplomové práce:

Ing. Petr Chalupa, Ph.D.

Ústav řízení procesů

Datum zadání diplomové práce:

7. března 2014

Termín odevzdání diplomové práce:

11. června 2014

Ve Zlíně dne 7. března 2014

prof. Ing. Vladimír Vašek, CSc.
děkan



prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval.
V případě publikace výsledků budu uveden jako spoluautor.
- ~~že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.~~

Ve Zlíně

.....
podpis diplomanta

ABSTRAKT

Prediktivní řízení založené na modelu řízeného systému (MPC - Model Predictive Control) zaznamenává v posledních letech výrazný rozvoj. MPC je většinou založeno na diskretních modelech. Model řízeného procesu se využívá k predikci jeho budoucích výstupů na základě minulých vstupů, výstupů nebo stavů a navržené sekvence budoucích akčních zásahů. Výpočetní aspekty metod predikce signálů mohou být limitující pro jejich praktické nasazení.

Diplomová práce je zaměřena na metody nelineárního prediktivního řízení a srovnává lineární a nelineární prediktivní řízení. Dále popisuje časovou náročnost algoritmů nelineárního prediktivního řízení a popisuje vhodnost v reálném čase. V této práci jsou realizovány vybrané algoritmy nelineárního prediktivního řízení v prostředí MATLAB/Simulink a je provedena jejich simulační ověření na silně nelineárním simulinkovém modelu tří nádrží. V teoretické části jsou studovány a popsány metody Linear Quadratic, Linear Quadratic Gaussian, Generalized Predictive Control a Receding Horizon Control. Praktická část se zabývá implementací optimalizačního algoritmu k reálnému modelu. V prostředí Matlab/Simulink byly vytvořeny simulační programy, které umožňují ověřit správnost simulací a výpočetní složitost.

Klíčová slova: Prediktivní řízení, simulace, výpočetní složitost

ABSTRACT

Predictive control based on the model of the controlled system (MPC - Model Predictive Control) has recorded a significant development in the last few years. MPC is mostly based on discrete models. The process model is used to predict the future outcomes on the basis of past inputs, outputs and states and designed sequence of future action control efforts. Computational aspects of methods for predicting signal may be limiting for their practical use.

The thesis is focused on methods of non-linear predictive control and compares linear and nonlinear predictive control. It also describes the time-consuming nonlinear predictive control algorithms and describes the suitability in real time. In this work are executed selected nonlinear predictive control algorithms in MATLAB / Simulink and has been performed to verify the simulation of strongly nonlinear Simulink model of three tanks. In the theoretical part are studied and the methods of Linear Quadratic, Linear Quadratic Gausi-

an, and Generalized Predictive Control Receding Horizon Control. The practical part deals with the implementation of the optimization algorithm to a real model. In a Matlab/Simulink simulation programs have been developed that allow simulations to verify the accuracy and computational complexity.

Keywords: Predictive control, simulation, computational complexity

Poděkování:

Děkuji vedoucímu své diplomové práce Ing. Petrovi Chalupovi za odborné vedení mé diplomové práce, podmětné připomínky a rady udílené při vypracování práce. Také děkuji za konzultace při vytváření diplomové práce.

Motto:

Sláb jenom ten, kdo v sebe ztratil víru, a malý ten, kdo zná jen malý cíl.

Svatopluk Čech

Poděkování, motto a čestné prohlášení, že odevzdaná verze diplomové práce a verze elektronická, nahraná do IS/STAG jsou totožné ve znění:

Prohlašuji, že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD.....	10
I TEORETICKÁ ČÁST.....	11
2 ZÁKLADNÍ POJMY	12
2.2 LINEÁRNÍ SYSTÉM.....	12
2.3 NELINEÁRNÍ SYSTÉM.....	12
2.4 PREDIKTIVNÍ ŘÍZENÍ	12
3 REŠERŠE.....	13
3 ÚVOD DO PREDIKTIVNÍHO ŘÍZENÍ.....	18
3.1 KVADRATICKÉ KRITÉRIUM	19
4 VÝPOČETNÍ SLOŽITOST ALGORITMU.....	21
5 OPTIMALIZAČNÍ ALGORITMY.....	22
5.2 ALGORITMUS LEVENBERG-MARQUARDT	22
5.2.1 Numerická implementace.....	22
5.3 ALGORITMUS TRUST-REGION-REFLECTIVE (DŮVĚRA-REGIONU).....	23
5.4 ALGORITMUS NELDER-MEAD SIMPLEX DIRECT SEARCH	24
5.5 ALGORITMUS SQP	26
5.5.1 Základní informace o algoritmu.....	27
5.6 ALGORITMUS ACTIVE SET	27
5.6.1 Příklad použití	28
5.7 ALGORITMUS LINE SEARCH	28
5.7.1 Příklad použití	29
5.7.2 Algoritmy	29
5.7.2.1 Direct search methods.....	29
5.7.3 Příklad použití	30
5.8 GENETIC ALGORITHM	30
5.9 PATTERN SEARCH ALGORITMUS.....	30
5.9.1 Konvergence	31
5.9.2 Příklad použití	31
II PRAKTICKÁ ČÁST	32
6 VÝPOČETNÍ ČAS ALGORITMŮ	33
6.1 ALGORITMUS LQ	34
6.2 ALGORITMUS GPC	35
38	
7 NELINEÁRNÍ ŘÍZENÍ POMOCÍ LINEARIZACE.....	40
7.2 ŘÍZENÍ NELINEÁRNÍHO SYSTÉMU	40
7.2.1 Řízení nelineárního systému při horizontech $NU = 10$ $N2 = 40$	42
7.2.2 Řízení nelineárního systému při horizontech $NU = 10$ $N2 = 10$	43
7.2.3 Řízení nelineárního systému při horizontech $NU = 30$ $N2 = 40$	44
8 NELINEÁRNÍ ŘÍZENÍ POMOCÍ OPTIMALIZACE.....	46
8.2 NELINEÁRNÍ ČTVERCE POMOCÍ PŘÍKAZU LSQNONLIN.....	46
8.2.1 Simulace levenberg-marquardt při horizontech $NU = 10$ $N2 = 40$	46

8.2.2	Simulace levenberg-marquardt při horizontech $NU = 30$ $N2 = 40$	47
8.2.3	Simulace levenberg-marquardt při horizontech $NU = 10$ $N2 = 10$	47
8.2.4	Simulace trust-region – reflective při horizontech $NU = 30$ $N2 = 40$	48
8.2.5	Simulace trust-region – reflective při horizontech $NU = 10$ $N2 = 40$	49
8.2.6	Simulace trust-region – reflective při horizontech $NU = 10$ $N2 = 10$	49
8.3	POMOCÍ PŘÍKAZU FMINSEARCH	51
8.3.1	Simulace nelder-mead simplex při horizontech $NU = 10$ $N2 = 10$	51
8.4	POMOCÍ PŘÍKAZU FMINUNC	52
8.4.1	Simulace In-line – search při horizontech $NU = 10$ $N2 = 10$	52
8.4.2	Simulace active - set při horizontech $NU = 10$ $N2 = 10$	53
8.4.3	Simulace SQP při horizontech $NU = 10$ $N2 = 10$	53
8.4.4	Simulace genetic algorithm při horizontech $NU = 10$ $N2 = 10$	54
8.4.5	Patternsearch algorithm při horizontech $NU = 10$ $N2 = 10$	55
8.5	POROVNÁNÍ JEDNOTLIVÝCH ALGORITMŮ	56
9	REÁLNÉ MĚŘENÍ	58
9.2	REGULACE SIMULINGOVÉHO MODELU	59
9.2.1	Regulace pomocí nastavení regulátoru PID	60
9.2.2	Regulace pomocí optimalizace	61
9.2.3	Porovnání regulací pomocí kritérií	62
9.3	REÁLNÉ MĚŘENÍ	63
9.4	POROVNÁNÍ MODELU A REÁLU	63
9.4.1	Regulace regulátoru PID	64
9.4.2	Regulace pomocí optimalizace	65
9.4.3	Porovnání reálné regulace pomocí kritérií	66
ZÁVĚR		67
SEZNAM POUŽITÉ LITERATURY		68
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK		74
SEZNAM OBRÁZKŮ		75
SEZNAM TABULEK		77
SEZNAM PŘÍLOH		78

ÚVOD

V současné době je výpočetní technika důležitým aspektem pro úspěch člověka v téměř všech odvětvích lidského snažení, obzvláště důležitou roli hraje v komplexním vývoji informačních technologií. Používá se převážně v oblastech, kde je důležité zpracovávání dat, které by bylo klasickými metodami časově a technicky náročné. Díky použití výpočetní techniky v těchto oblastech se tedy dosahuje snížení celkových nákladů. Mezi tyto oblasti patří zejména grafické simulace průběhu procesu na základě vložených informací, dále pak složité výpočetní operace, rozsáhlé databáze dat a jiné operace, které by bez pomoci programů navržených pro tyto účely byly dnes již snad neřešitelné. V dnešní době nelze již pokládat softwarové vybavení výpočetní techniky jen za pomocné nástroje při zpracovávání dat, ale jejich úroveň je již tak velká, že jsou schopny samostatného řešení zadaných úloh, jejich kontroly a návrhu, provozu, řízení a ovládání [1], [2].

Jedním z oborů, kde výpočetní technika hraje obzvláště důležitou roli, je automatizace a řízení. V některých fázích dokonce zcela zastupuje lidský faktor. Jejím použitím dochází k usnadnění návrhu regulátorů a jejich parametrů, mezi které patří volba typu regulátoru a jeho struktury, atd. Výpočetní technika má své nezastupitelné místo i při průběžné analýze, simulaci, monitorování atd. [1], [2]. V oblasti automatizace a řízení se využívá velké množství programů sloužící např. k regulaci měřené soustavy, simulaci regulačního pochodu, atd. Jedním z nejpoužívanějších programů, které tyto operace provádí, je bezesporu MATLAB. Tento program obsahuje velké množství nástrojů, pomocí kterých je schopen na základě vložených dat provádět numerické či grafické řešení zadaných úloh, identifikaci systému, jeho řízení, optimalizaci a další [1], [2].

Mezi nejrozšířenější regulátory v oblasti automatizace a řízení patří bezesporu PID regulátory a to zejména díky jejich jednoduchosti. V praxi se objevují jak v analogové tak i v číslicové formě. Existuje u nich spousta nedostatků, obzvláště u systému s dopravním zpožděním, při kompenzaci poruch, apod. Rozvoj číslicové techniky umožnil návrh nových struktur regulátorů, což vedlo k zlepšení této situace [3].

Stávající algoritmy byly však stále navrhovány a odvozovány pro spojité systémy, ale postupem času se díky výzkumu objevily i algoritmy odvozené pro diskrétní systémy.

A právě modelování procesů v diskrétní formě umožnilo předpovídat budoucí hodnoty výstupu procesu. Tato strategie je považována za předchůdce současného prediktivního řízení, které umožňuje plné využití predikce výstupu systému pro řízení v reálném čase.

I. TEORETICKÁ ČÁST

2 ZÁKLADNÍ POJMY

2.2 Lineární Systém

Lineární systém je takový systém, v němž lze uplatnit princip superpozice. To lze ilustrovat na následujícím příkladu: jestliže $f(x) = 0$ a současně $f(y) = 0$, potom také $f(x + y) = 0$.

Pro funkci f lze potom využít principu superpozice. Superpozice využíváme při řešení velkého množství problémů, například při řešení průtoku elektrického proudu v elektronických obvodech nebo ve fyzice při skládání působení sil na hmotný bod. Obecně platí, že je-li systém lineární a lze využít superpozice, je řešení takového systému často velmi jednoduché a jednoznačné. Chování takových systémů lze předpovědět i do budoucnosti.[53]

2.3 Nelineární systém

Nelineární systém je takový systém, kde neplatí princip superpozice. To znamená, že jestliže $f(x) = 0$ a současně $f(y) = 0$, není zaručeno, že také $f(x + y) = 0$. V nelineárním systému platí princip superpozice pouze pro malou množinu izolovaných bodů, kterým říkáme *fixní body*. Je-li systém nelineární a nelze tedy využít principu superpozice, je nutné pro výpočet změny stavu systému řešit diferenciální rovnice, což je mnohdy velmi složité. Také není zaručeno, že se nám podaří předpovědět stav systému i do budoucnosti. V případě elektrických obvodů je nelineárním prvkem například dioda. Pro diodu neplatí klasický Ohmův zákon a je-li dioda ve složitějším elektrickém obvodu, nelze tento obvod řešit běžnými metodami (superpozice, smyčkové proudy). Někdy se takový systém pro snazší výpočty linearizuje, tj. nelineární závislost se nahradí závislostí lineární.[54]

2.4 Prediktivní řízení

Prediktivní řízení je skupina řídicích technik, které používají model řízeného systému k predikci budoucích hodnot systému tak, aby počítaly sekvence budoucích řídicích akcí optimalizujících kritérium. Není tedy jediným přesně vymezeným přístupem řízení, ale rozsáhlou skupinou metod řízení založených na několika společných myšlenkách, jejichž propojení vede k algoritmům s navzájem poměrně blízkou strukturou. [55]

3 REŠERŠE

Pro provoz průmyslových zařízení v optimálním provozu za jakýchkoliv podmínek je nevyhnutelné, aby vypočtené a generované akční zásahy byly optimální vzhledem na dané kritéria, jako např. minimalizace spotřeby vstupních surovin, případné maximalizace výsledných získaných výrobků. Tento způsob řízení se běžně označuje jako optimální řízení. Jedním ze způsobů integrace optimálního řízení v průmyslové praxi je tzv. prediktivní řízení s modelem (Model Predictive Control, MPC), který je v současné době jedním z nejvíce rozvíjejících se přístupů v automatickém řízení a jako jeden z mála teoretických přístupů je i masivně nasazovaný v praxi tam, kde není zajištěna dostačující kvalita řízení při použití klasických regulátorů. Je to dáno skutečností, že první návrhy prediktivního řízení, tzv. DMC (Dynamix Matrix Control) byly navrženy pro účely praxe [4].

V tomto směru se hledají akční zásahy, které optimalizují zvolenou účelovou funkci vzhledem na procesní omezení. Důležitým faktorem v MPC je princip predikce neboli předpovídání budoucího chování systému na základě znalosti vhodného matematického popisu. Predikce umožňuje při tomto přístupu řízení „vidět“ do krátké budoucnosti a optimalizovat akční zásahy potřebné na dosažení daného cíle s minimálním vynaložením energie [5].

První algoritmy prediktivního řízení se už před více než dvaceti pěti lety a možná i dřív začaly využívat v průmyslu jako účinný způsob řízení mnoha rozměrových průmyslových procesů s omezeními. Z důvodů relativně velkých výpočetních nároků bylo použití prediktivního řízení omezeno hlavně na řízení pomalých procesů. V posledních letech byl však zaznamenán i významný pokrok v oblasti zdokonalování a vývoje výpočetních prostředků, v čteně nových výkonných numerických metod, a z tohoto hlediska i toto omezení ztrácí význam. Za jednu z předností prediktivního řízení je schopnost včlenit do algoritmu prediktivního regulátoru omezení vstupních, výstupních i stavových veličin. Téměř všechny průmyslové procesy taková omezení vykazují. Strategie prediktivního řízení z tohoto důvodu opravuje chyby existujících optimálních metod, jakými jsou např. Linear Quadratic (LQ) a Linear Quadratic Gaussian (LQG) metody, které pracují na konečném horizontu bez schopnosti řešení omezení. Metody prediktivního řízení se vyvíjely nezávisle ve třech různých směrech. První směr prediktivního řízení MPC byl publikován v několika příspěvcích už koncem sedmdesátých let minulého století. Druhý směr prediktivního řízení nazvaný GPC (Generalized Predictive Control) byl vyvíjen v oblasti adaptivního řízení [6].

Třetí směr prediktivního řízení označený jako RHC (Receding Horizon Control) byl vyvinut v rámci akademického výzkumu jako alternativa LQ nebo LQG řízení. Stabilizující metoda RHC, minimalizující kvadratické kritérium při koncovém stabilizujícím omezení ve tvaru podmínek rovnosti, byla navržena v příspěvku [7]. Problematika RHC je souhrnně uvedena v monografii [8].

Všechny výše zmíněné směry prediktivního řízení byly v jejich začátcích vyvíjeny nezávisle. Začátkem devadesátých let minulého století bylo publikováno několik pokusů o vyjasnění souvislostí

mezi jednotlivými trendy a vytvoření určitých způsobů sjednocení těchto směrů [9, 10,11]. Přehledně o současných metodách prediktivního řízení pojednávají monografie [12, 13, 14, 15, 16].

První fází řešení problémů optimalizace a optimálního řízení je sestavení vhodného matematického modelu, který musí splňovat dva základní požadavky. Jednak musí tento model s dostatečnou přesností popisovat chování uvažovaného procesu ve všech provozních stavech, aby se dal efektivně přetransformovat na matematickou formulaci problému optimálního řízení. Vlastní proces získání optimálního řešení může být realizován buď on-line (opakovaným řešením optimalizačního problému v každé periodě vzorkování pro jednu konkrétní hodnotu počátečních podmínek (viz např. [4]) anebo off-line (získáním optimálního řízení pro všechny možné počáteční podmínky splňující omezení). Off-line řešení [17, 18], rovněž označované jako explicitní prediktivní řízení, je atraktivní především pro řízení procesů s rychlou dynamikou. V tomto přístupu výsledné řešení nabývá podobu vyhledávací tabulky. Proces získání optimálního akčního zásahu při tomto přístupu se potom redukuje na sekvenci jednoduchých vyhledávacích operací, které se dají realizovat v reálném čase v řádu mili- a mikro-sekund. Pro návrh prediktivních regulátorů je potřeba mít k dispozici matematický model řízeného systému. Při standardně používaných lineárních a nelineárních modelech se stále častěji používají tzv. hybridní modely, což jsou modely kombinující elementy spojité dynamiky a diskrétní logiky. Jako příklad může sloužit chemický reaktor, jehož dynamické chování je popsáno diferenciálními rovnicemi, ale některé akční veličiny mohou nabývat pouze diskrétní hodnoty, jako např. zapnuté a vypnuté chlazení, případně míchání reakční směsi způsobem, kdy jsou otáčky míchadla regulovatelné pouze v nespojitých krocích (např. tři úrovně otáček). Hybridní matematické modely rovněž umožňují popsat elementy rozhodování, jako jsou např. sekvence operací popsaných pomocí IF-THEN-ELSE podmínek. Toto dovoluje velmi efektivně popsat chování nelineárních systémů pomocí metody vícenásobné linearizace. Je třeba připomenout, že i když ve všeobecnosti hybridní modely náleží do třídy nelineárních systémů, návrh optimálního řízení je nepoměrně jednodušší, když je možné optimalizační problém formulovat jako problém minimalizace účelové funkce za přítomnosti celočíselných proměnných. Pro tuto třídu proměnných existují ve světě programové prostředky (balíky), které takové optimalizační úlohy řeší velmi efektivně [19, 20].

V současné době se používají ve spojitosti s prediktivním řízením rovněž nástroje umělé inteligence - neuronové sítě [21, 22], genetické algoritmy [23,24] a další. Slibnou oblastí pro budoucí výzkum je prediktivní řízení nelineárních systémů pomocí evolučních technik [25], kde jsou využívány techniky genetického programování, analytického programování či gramatické evoluce. Metody prediktivního řízení rovněž využívají přístupy založené na více modelech (MM – Multiple Model), kdy pracovní oblast systému je rozdělena na několik pracovních režimů, ve kterých je systém reprezentován lineárním lokálním modelem [26] umožňují snadnější interpretaci lokálních modelů a také snadné využití expertních znalostí pro tvor-

bu takového modelu. Kombinací těchto lokálních modelů je vytvořen globální dynamický model systému [27].

Teorii a aplikacím prediktivního řízení je věnována celosvětově stále velká pozornost. Skutečnost, že problematika prediktivního řízení není jen předmětem výzkumu na akademických pracovištích, ale teoretické výsledky jsou úspěšně přenášeny do průmyslové praxe, je dokladována v poměrně rozsáhlém přehledu průmyslových aplikací v příspěvcích [28-31]. Na pracovišti Ústavu řízení procesu FAI UTB ve Zlíně byla metoda prediktivního řízení aplikována pro řízení několika laboratorních modelů v reálném čase. V příspěvku [32] je uvedeno prediktivní řízení inverzního kyvadla, náplní příspěvků [33-36] je adaptivní řízení nelineárního servomotoru a obsahem příspěvku [37] je aplikace samočinně se nastavujícího regulátoru pro řízení průtokového tepelného výměníku. Do návrhu prediktivního řízení je možné zahrnout měřenou poruchovou veličinu, kterou regulátor kompenzuje, než by se mohla projevit na výstupu regulované soustavy. Tímto se zabývají příspěvky [38-40]. Prediktivním řízením s měřenou poruchovou veličinou u procesů s dopravním zpožděním se zabývá příspěvek [41] a u MIMO procesů příspěvek [42].

2.1. Porovnání jednotlivých směrů

LQ

Zde je úloha řešena numericky jako úloha kvadratického programování (QP) s omezujícími podmínkami ve tvaru lineárních nerovností. Nepočítá s budoucími kroky simulace.

GPC

Základní myšlenka spočívá v diskrétní oblasti ve výpočtu budoucího akčního zásahu minimalizací kritéria sumy kvadrátů rozdílů mezi predikovanými výstupy soustavy a budoucími žádanými hodnotami a sumy kvadrátů přírůstků akčních zásahů na konečném horizontu. V případě výpočtu akčních zásahů bez zahrnutí omezení, je možné provést výpočet analyticky, v případě výpočtu s omezením, je nezbytné postupovat numericky (úloha kvadratického programování).[56]

RHTC (Receding Horizont Tracking Control)

lišící se v použitém modelu soustavy (stavový model) a v tom, že tato metoda je rekurzivní.[57]

2.1.1. Nelineární soustavy

Nelineární soustavou se obecně myslí taková soustava, tedy má nelineární statickou charakteristiku a pro sestavení jejího matematického modelu je nutná alespoň jedna nelineární rovnice. Nelineární soustavy jsou rovněž v praxi velmi časté. Nicméně jejich regulace je mnohonásobně složitější a náročnější jednoho či vícerozměrných lineárních soustav. [56]

2.1.2. Řešení

Pro regulaci nelineárních soustav existují v principu dvě možnosti:

a) **On-line dohledávání parametrů regulátoru na základě aktuálního stavu soustavy**

Jedná se o analogii regulace lineárních soustav, přičemž regulátor musí velmi často měnit svoje nastavení a parametry.

Příkladem jsou principy adaptivního řízení. Predikční model ve tvaru diferenční rovnice mění na základě on-line identifikace metodou nejmenších čtverců svoje parametry, a nahrazuje tak nelineární chování soustavy lineárním (v daném okamžiku nejlepším) [56]

b) **Nelineární nastavení regulátoru**

Regulátor je zcela odlišný od své lineární verze. Jeho nastavení je od počátku nelineární. Příkladem je sestavení nelineárního matematického modelu pro predikci výstup soustavy nebo aproximaci nelineárního chování soustavy umělou neuronovou sítí. Úskalím tohoto přístupu je stabilita regulátoru, neboť nelineární rovnice se obvykle řeší numerickým postupem, který může za určitých podmínek velice rychle kolabovat. Pro takový regulátor je tedy nutné přidat celou řadu dalších podmínek pro případy nestability. [56]

2.1.3. Linearizace modelu

Linearizace matematického modelu je nezbytná, protože matematický model bude použit pro návrh regulace. Návrh složitějších regulátorů z matematického modelu vychází. Standardní návrh regulátoru předpokládá popis chování řízeného systému lineárním modelem obvykle ve tvaru diferenciální rovnice nebo stavového modelu. Nelineární model je možné využít tak, že bude linearizován ve vybraném bodě statické charakteristiky a pro návrh regulátoru se použije linearizovaný model v tomto bodě. [56]

Linearizovaný matematický model vychází z nelineárního modelu. Nejprve je nutné vypočítat ustálený stav, ve kterém bude provedena linearizace (jedná se o nelineární rovnice, výpočet je možný obvykle pouze numericky). Následně je možné pro ustálený stav vypočítat parametry linearizovaného modelu.[56]

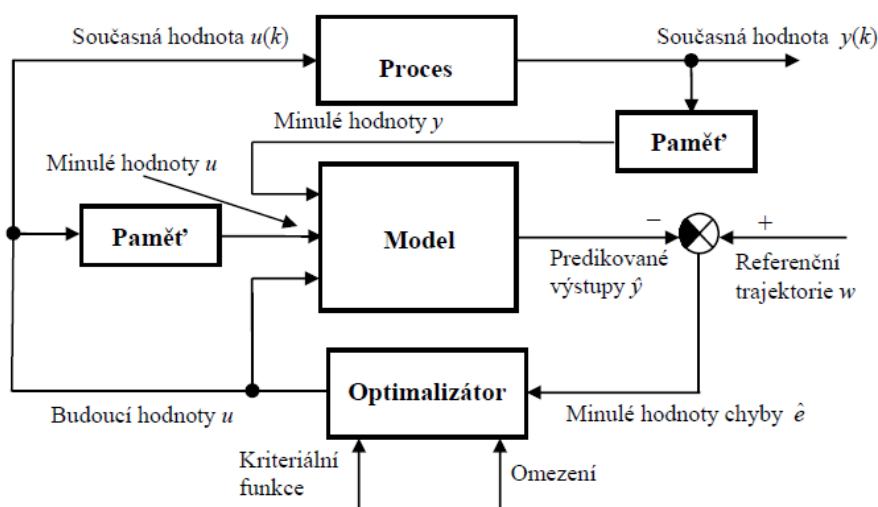
3 ÚVOD DO PREDIKTIVNÍHO ŘÍZENÍ

Prediktivní řízení je jednou z metod návrhů řízení, která si získala v posledních letech velkou popularitu. Prediktivní řízení je ve své podstatě založeno na použití diskretních respektive vzorkovaných modelech procesů, a proto odvození příslušných řídicích algoritmů je realizováno hlavně v diskretní oblasti.

Pod pojmem prediktivní řízení chápeme třídu metod řízení, které spojují určité společné charakteristiky:

- Matematický model řízení systému je použitý na predikci budoucího řízení výstupu systému.
- Je znám průběh trajektorie žádané hodnoty regulované veličiny v budoucím čase.
- Výpočet posloupnosti budoucích řídicích zásahů zahrnuje minimalizaci vhodné účelové funkce (obvykle kvadratické) s budoucími trajektoriemi přírůstků řízení a regulační odchylky.
- Jen první akční zásah je skutečně realizovaný a celý postup minimalizace funkcionálu se opakuje v další periodě vzorkování.

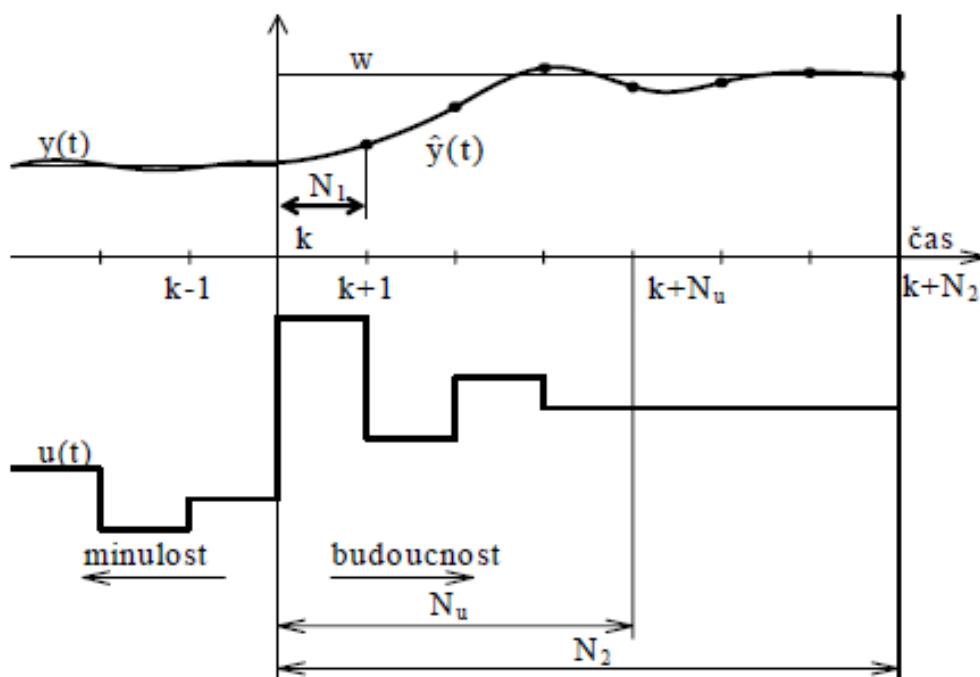
Jednou z předností prediktivního řízení je možnost uvažovat omezení vstupních a výstupních (případně stavových) veličin přímo při návrhu regulátoru. Z tohoto důvodu je jeho použití vhodné v průmyslových aplikacích. Algoritmy prediktivního řízení jsou při řízení procesů mnohostranně použitelné a robustní. Kvalita řízení je obvykle vyšší ve srovnání s PID regulátory. Jsou aplikovatelné na neminimálně fázové, nestabilní a mnoha rozměrové procesy, a rovněž na procesy s dopravním zpožděním. [41]



Obrázek 1 – Základní struktura prediktivního řízení[41]

Základní struktura systému prediktivního řízení je uvedena na obr. 1, jeho princip, uvedený na obr. 2, je následující:

1. Model řízeného procesu je explicitní součástí regulátoru a je používán na predikci N budoucích výstupů procesu \hat{y} . Predikce jsou vypočítané vzhledem k informacím dostupným do času k a vzhledem k neznámé trajektorii akčních zásahů, které je třeba určit.
2. Trajektorie budoucích akčních zásahů je určena z řešení optimalizačního problému, obsahujícího vhodnou účelovou funkci a omezení. Účelová funkce zahrnuje budoucí predikce výstupu, budoucí trajektorii žádané veličiny a budoucích akčních zásahů.
3. I když v předešlém kroku byla počítána celá trajektorie akčních zásahů, jen první člen $u(k)$ je použit pro řízení procesu. V další periodě vzorkování se celý postup opakuje. Tento princip je známý jako strategie pohyblivého horizontu. [41]



Obrázek 2 - Princip prediktivního řízení (N_1 , N_2 , N_u – minimální, maximální, řídicí horizont) [41]

3.1 Kvadratické kritérium

Metody minimalizují kvadratické kritérium s penalizací akční veličiny

$$J = \sum_{k=0}^{\infty} \{ [w(k) - y(k)]^2 + q_u [u(k)]^2 \} \quad (2.1)$$

kde q_u je tzv. penalizační konstanta, která udává podíl akční veličiny na hodnotě kritéria (konstantu u prvního členu kritéria uvažujeme rovnu jedné). Standardní postup minimali-

zace kritéria (1) vychází ze stavového popisu soustavy a vede na řešení Riccatiho rovnice. Budeme-li posloupnosti hodnot regulační odchylky $e(k)$ a vstupního signálu $u(k)$ považovat za polynomy, lze kritérium (2.1) zapsat jako

$$J = \langle E(z)E(z^{-1}) + q_u U(z)U(z^{-1}) \rangle \quad (2.2)$$

Kde $\{x(z) = x(0)\}$, tj. v celém rozsahu je za proměnnou z resp. z^{-1} dosazena nula. $E(z)$ a $U(z)$ jsou sdružené polynomy k polynomům $E(z^{-1})$ a $U(z^{-1})$, tj. mají záporné mocniny z nahrazené kladnými. Např. sdružený polynom k polynomu

$$E(z^{-1}) = 1 + e_1 z^{-1} + e_2 z^{-2} \text{ je } E(z) = 1 + e_1 z + e_2 z^2$$

Roznásobíme-li polynomy v kritériu (2.2) a vynulujeme členy se z resp. z^{-1} , obdržíme sumy kvadrátů ve shodě se zápisem kritéria v rovnici (2.1). Do kritéria (2.2) dosadíme polynom regulační odchylky.[41]

4 VÝPOČETNÍ SLOŽITOST ALGORITMU

V praxi je třeba zajistit, aby algoritmus skončil „v rozumném“ čase. Za rozumný čas lze v praxi považovat takový čas, který nám umožní smysluplně využít výsledek. Např. existuje jednoduchý algoritmus, který dokáže určit, zda si v dané šachové pozici může hráč na tahu vynutit vítězství a zároveň dokáže určit nejlepší možný tah. Tento algoritmus se však nedá použít, protože by na svou činnost potřeboval ohromné množství času, jakkoli je toto množství konečné. Mimoto by takový algoritmus spotřeboval ohromné množství paměti, což je další praktický zřetel, který se uplatňuje při volbě algoritmu. I když průměrná počítačová paměť stále narůstá, pro některé algoritmy jí nebude nikdy dost. Pro vyčíslení výpočetní složitosti algoritmů v závislosti na velikosti vstupních dat se používá asymptotický zápis závislosti výpočetního času na rozsahu úlohy (typicky na počtu vstupních údajů). Například $O(\log N)$ znamená, že počet kroků algoritmu závisí logaritmicky na velikosti vstupních dat. Pokud u takového algoritmu zdvojnásobíme rozsah vstupních údajů, doba výpočtu se zvýší o jednu jednotku času, pokud bude vstupních dat čtyřikrát více, doba výpočtu se prodlouží o dvě jednotky času, a tak dále. To je třeba případ nalezení jednoho prvku o určité hodnotě v seznamu prvků seřazeném podle hodnoty (např. nalezení jména v telefonním seznamu).[42]

Z čehož můžeme vyčíst, že pro lineární systémy bude výpočetní čas menší než pro nelineární systémy.

5 OPTIMALIZAČNÍ ALGORITMY

Optimalizace je v informatice takový proces nalezení minima kriteriální funkce, která vede k jeho vyšší efektivitě nebo ke snížení nároků celého řídicího systému. Výpočetním systémem může být počítačový program, počítač, celá počítačová síť, komplexní řešení určitého problému a podobně.

Například program může být optimalizován tak, aby pracoval rychleji, potřeboval pro provedení výpočtu méně operační paměti, méně systémových prostředků, případně se rychleji spustil a podobně.

U optimalizace záleží na rychlosti provádění kódu, na velikosti výsledného kódu a na paměťové náročnosti.[51]

5.2 Algoritmus levenberg-marquardt

Levenberg - Marquardt algoritmus je adaptivně se měnící aktualizací parametrů mezi aktualizacemi klesání gradientu a aktualizací Gauss-Newton,

$$[J^T W J + \lambda I] h_{lm} - J^T W (y - \hat{y}) \quad (4.1)$$

kde pro malé hodnoty parametru λ je algoritmický výsledek v aktualizaci Gauss-Newton a velké hodnoty λ výsledku v přechodu aktualizace sestupu. Parametr λ je inicializován, pokud je velký. Pokud se v iteraci stane, že je výsledek horší aproximace, λ se zvýší. Když se řešení se blíží minimu, λ se sníží, metoda Levenberg-Marquardt s přístupem metody Gauss-Newton, výsledek obvykle rychle konverguje k lokálnímu minimu.

5.2.1 Numerická implementace

V iteraci i , krok h se vyhodnocuje porovnáním $x^2(p)$ do $x^2(p + h)$ (4.2). Krok je akceptován, jestliže metrické ρ_i je větší než uživatelem zadaná hodnota ϵ_4 . Pokud je v iteračním $\rho_i(h) > \epsilon_4$ pak $p + h$ je dostatečně lepší než p , p se nahrazuje $p + h$, a λ je snížena faktorem. Jinak λ se zvýší o faktor, a algoritmus výnosů na další iteraci. [43]

Další informace jsou zahrnuty v článku. [44]

storu. Dvourozměrný podprostor S se určuje za pomoci stabilizované konjugované s přechodem procesu popsaného níže. Řešitel definuje S , jako lineární prostor od $S1$ a $S2$, kde $S1$ je ve směru gradientu g , a $S2$ je směr Newton, tj. řešení

$$H^*s_2 = -g \quad (4.6)$$

nebo směr negativního zakřivení,

$$s_2^T * H * s_2 < 0 \quad (4.7)$$

Filozofie této volbě S je přinutit globální konvergenci (přes nejstrmější směr sestupu nebo záporném směru zakřivení) a dosáhnout rychlé lokální konvergence (pomocí kroku Newton, když to existuje). Skica neomezenou minimalizací pomocí myšlenky (důvěry regionu) je nyní snadné, aby se postupovalo podle těchto kroků:

1. Formulovat dvourozměrný trust-region podproblém.
2. Řešení rovnice (4.4) určit zkušební krok S .
3. Je-li $f(x + y) < f(x)$, pak $x = x + y$.
4. Nastavit Δ .

Tyto čtyři kroky se opakují, dokud je konvergence. Důvěry region rozměr Δ je upravený podle standardních pravidel. Zejména se snižuje, je-li zkušební krok nepřijat, tj. $f(x+y) \geq f(x)$. Viz [46] a [49] pro diskuzi o tomto aspektu.

5.4 Algoritmus Nelder-Mead simplex direct search

Nelder-Mead simplex algoritmus, tento algoritmus používá simplex z $n + 1$ bodů pro n -rozměrné vektory x . Algoritmus nejprve vytvoří simplex kolem počátečního odhadu $x0$ přidáním 5% pro každou složku $X0(i)$ k $X0$, a použití těchto n vektorů jako prvky simplexu kromě $x0$. (Používá 0,00025 jako složku i v případě, $x0(i) = 0$). Poté algoritmus opakovaně mění simplex podle následujícího postupu.

1. Nechť $x(i)$ označuje seznam bodů v aktuálním simplex, $i = 1, \dots, n + 1$.
2. Objednejte body simplexu od nejnižší hodnoty funkce $f(x(1))$ na nejvyšší $f(x(n + 1))$. Na každém kroku iterace algoritmu zahodí aktuální nejhorší bod $x(n + 1)$, a přijme další bod do simplexu. [Nebo, v případě, že krok 7 je níže, změní všech n hodnot s hodnotami nad $f(x(1))$].

3. Generuje odrazný bod

$$r = 2m - x(n+1), (4.8)$$

kde

$$m = \sum x(i)/n, i = 1...n, (4.9)$$

a výpočet $f(r)$.

4. Pokud $f(x(1)) \leq f(r) < f(x(n))$, přijmout r a ukončit tuto iteraci.

5. If $f(r) < f(x(1))$, výpočet rozšířeného bodu S

$$s = m + 2(m - x(n+1)), (4.10)$$

výpočet $f(s)$.

- a. Je-li $f(y) < f(r)$, přijme s a ukončení iterace.
 - b. V opačném případě, přijímat r a ukončí iteraci.
6. Je-li $f(r) \geq f(x(n))$, provést kontrakci mezi m a lepší $x(n+1)$ a r :
- a. Je-li $f(r) < f(x(n+1))$ (tj., r je lepší než $x(n+1)$), výpočet

$$c = m + (r - m)/2 (4.11)$$

a výpočet $f(c)$. Je-li $f(c) < f(r)$, přijímat c a ukončení iterace. V opačném případě pokračujte krokem 7.

- b. If $f(r) \geq f(x(n+1))$, výpočítá

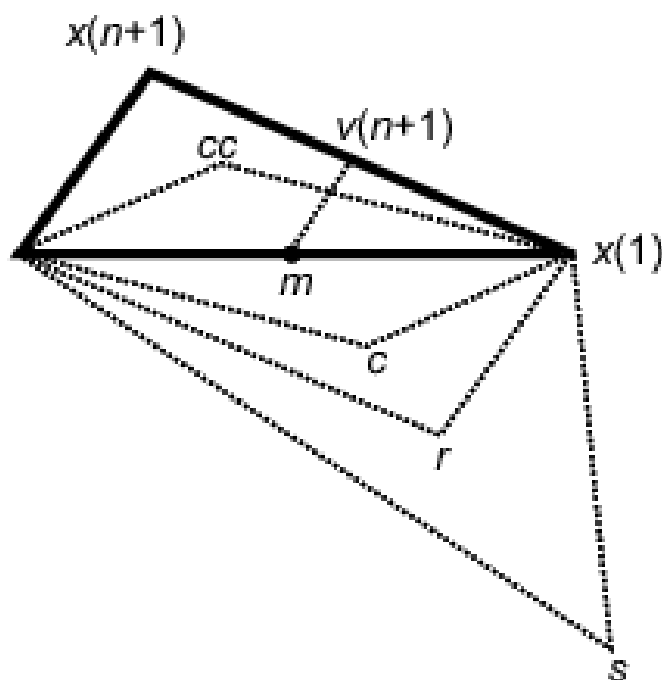
$$cc = m + (x(n+1) - m)/2 (4.12)$$

a vypočítat $f(cc)$. Je-li $f(cc) < f(x(n+1))$, přijmout CC a ukončení iterace. V opačném případě pokračujte krokem 7.

7. Výpočet n bodů

$$v(i) = x(1) + (x(i) - x(1))/2 (4.13)$$

a vypočítat $f(v(i))$, $i = 2, \dots, n+1$. Simplex na další iteraci je $x(1), v(2), \dots, v(n+1)$. Následující obrázek ukazuje body, které příkaz *fminsearch* může převést v řízení, spolu s každým možným novým simplexem. Původní simplex má smělé obrysy. V iteraci aplikace pokračuje, dokud splňují kritérium pro zastavení. [58]



Obrázek 3 – Naznačení Nelder-Mead algoritmu

5.5 Algoritmus SQP

Sekvenční kvadratické programování (SQP) je iterační metoda pro nelineární optimalizaci. SQP metody jsou používány na problémy, pro které účelová funkce a omezení jsou dvakrát spojitě diferencovatelná.

SQP metody řeší posloupnost optimalizace podproblému, z nichž každý optimalizuje kvadratický model objektivního předmětu k linearizaci omezení. Pokud je problém neomezený, pak metoda redukuje na Newtonovu metodu pro nalezení místa, kde gradient cíle mizí. Pokud problém má pouze omezení rovnosti, pak je metoda ekvivalentní použití Newtonovu metodu na podmínky optimality prvního řádu, nebo Karushovy-Kuhn-Tucker podmínky, problému. SQP metody byly implementovány v mnoha obalech, včetně NPSOL, SNOPT, NLPQL, OPSYC, OPTIMA, MATLAB a SQP.

5.5.1 Základní informace o algoritmu

Uvažujme nelineární programování problému formuláře:

$$\min_x f(x) \quad (4.14)$$

$$s. t. \quad b(x) \geq 0 \quad (4.15)$$

$$c(x) = 0 \quad (4.16)$$

Lagrange pro tento problém je;

$$(x, \lambda, \sigma) = \mathcal{L} f(x) - \lambda^T b(x) - \sigma^T c(x) \quad (4.17)$$

Zejména v kvantové teorii pole se používá hustota lagrangiánu, vyjadřující jeho prostorové rozložení. Vzájemná souvislost je dána vztahem, kde λ a σ jsou Lagrangeovy multiplikátory. Při iteraci x_k základní sekvenční kvadratické programovací algoritmus definuje příslušný směr hledání jako řešení kvadratické programování podproblému.

$$\min_d f(x_k) + \nabla f(x_k)^T d + \frac{1}{2} d^T \nabla^2 \mathcal{L}(x_k, \lambda_k, \sigma_k) d \quad (4.18)$$

$$s. t. \quad b(x_k) + \nabla b(x_k)^T d \geq 0 \quad (4.19)$$

$$c(x_k) + \nabla c(x_k)^T d = 0 \quad (4.20)[59]$$

5.6 Algoritmus Active set

V matematické optimalizaci, problém je definován za použití objektivní funkce, aby se minimalizovalo nebo maximalizovalo, a sadu omezení.

$$g_1(x) \geq 0, \dots, g_k(x) \geq 0 \quad (4.21)$$

při definování proveditelné oblasti, to znamená, že množina všech x hledá optimální řešení.

Vzhledem k tomu, bod x v proveditelné oblasti, a omezení

$$g_i(x) \geq 0 \quad (4.22)$$

se nazývá aktivní x jestliže $g_i(x) = 0$ a neaktivní když x jestli $g_i(x) > 0$ Omezení jsou vždy aktivní. Aktivní sada na se skládá z těchto omezení $g_i(x)$, které jsou aktivní v aktuálním místě.

Aktivní sada je zvláště důležitá v teorii optimalizace, které omezení bude mít vliv na konečný výsledek optimalizace. Například, při řešení problému lineárního programování, aktivní sestava dává nad rovinami, které se protínají v bodě řešení. V kvadratickém programování, protože řešení není nutně na jednom z okrajů ohraničovacího mnohoúhelníku, odhad aktivní sady nám dává podmnožinu nerovností sledování při hledání řešení, které snižuje složitost vyhledávání.

5.6.1 Příklad použití

Obecně aktivní sadu algoritmus má následující strukturu:

1. Najít proveditelný výchozí bod.
2. Opakovat, dokud "dost optimální".
3. Vyřešit problém rovnosti definované aktivní sady (přibližně).
4. Vypočítat Lagrangeovy multiplikátory aktivního souboru.
5. Odebrat jen některé omezení s negativními Lagrangeových multiplikátorů.
6. Hledat nemožné omezení.
7. Konec opakování.

Metody, které mohou být popsány, jako účinné metody uvedené jsou:

- Postupné lineární programování (SLP)
- Sekvenční kvadratické programování (SQP)
- Sekvenční lineární kvadratické programování (SLQP)
- Snížená metoda gradient (RG)
- Zobecněná metoda snižující gradient (DRG)

[60]

5.7 Algoritmus Line Search

V optimalizaci, linka hledání strategie je jedním ze dvou základních iteračních postupů, jak najít lokální minimum X^* objektivní funkce $f: \mathbb{R}^n \rightarrow \mathbb{R}$. Druhý přístup je důvěra region. Hledání dle přístupu nejprve zjistí, směr klesání, podél níže objektivní funkce se sníží a poté vypočítá velikost kroku, který určuje, jak daleko se X musí pohybovat podél tohoto směru. Směr klesání lze vypočítat různými způsoby, jako je například přechodem původu, Newtonovy metody a metody Quasi-Newtona. Velikost kroku může být určena buď přesně, nebo nepřesně.

5.7.1 Příklad použití

Zde je příklad metoda přechod, který používá hledání řádku v kroku 4.

1. Nastavte iterace $k = 0$ pult, a provést počáteční odhad, X_0 za minimální
2. Opakovat:
3. Vypočítat směr sestupu P_k
4. Vybrat si "volně" minimalizovat $h(\alpha) = f(x_k) + \alpha P_k$ (4.23) na $\alpha \in \mathbb{R}_+$
5. Aktualizace $X_{K+1} = X_k + \alpha_k P_k$ a $k = k+1$ (4.24)
6. Do $\|\nabla f(X_k)\| < \text{tolerance}$ (4.25)

Na řádku vyhledávacího kroku (4) algoritmus může buď přesně minimalizovat h , při řešení $h'(\alpha_k) = 0$, nebo volně, tím, že žádá o dostatečné snížení hod. Jeden příklad za směr je konjugát metoda gradientu. Ta se nazývá vyhledávání nepřesné vedení, a mohou být provedeny mnoha způsoby, jako je například vyhledávání zpětné hledání vedení nebo pomocí Wolfe podmínek. Stejně jako ostatní metody optimalizace, vedení hledání lze kombinovat simulované žíhání, aby mohla skákat přes nějaké místní minima.

5.7.2 Algoritmy

5.7.2.1 Direct search methods

V této metodě, minimum, musí nejprve být v závorkách, takže algoritmus musí identifikovat body x_1 a x_2 tak, aby hledal minimum ležící mezi nimi. Interval je pak rozdělen na počítání ve dvou vnitřních bodech, X_3 a X_4 , a odmítnutí podle toho, která z těchto dvou vnějších bodů nesousedí, že X_3 a X_4 , který má nejnižší hodnotu funkce. Následné kroky, pouze jeden navíc vnitřní bod, musí být vypočteny. Z různých způsobů dělení intervalu, hledání zlatého řezu je obzvláště jednoduché a efektivní, protože interval proporce jsou zachovány bez ohledu na to, jak bude hledání probíhat:

$$\frac{1}{\phi}(x_2 - x_1) = x_4 - x_1 = x_2 - x_3 = \phi(x_2 - x_4) = \phi(x_3 - x_1) = \phi^2(x_4 - x_3) \quad (4.26)$$

Kde:

$$\phi = \frac{1}{2}(1 + \sqrt{5}) \approx 1,618 \quad (4.27)$$

5.7.3 Příklad použití

- hledání zpětné sledování linky
- Metoda Secant
- metoda Newton-Raphson
- hledání Pattern (optimalizace)
- Metoda Nelder-Mead
- Hledání zlatého řezu [61]

5.8 Genetic Algorithm

Genetický algoritmus (GA) je metoda pro řešení jak omezených a neomezených problémů s optimalizací na základě přirozeného procesu výběru, který napodobuje biologickou evoluci. Algoritmus opakovaně upravuje populaci jednotlivých řešení. Na každém kroku, genetický algoritmus náhodně vybere jednotlivce ze současné populace a používá je, jako rodiče vyrábějící děti pro příští generace. V průběhu po sobě jdoucích generací, se populace "vyvíjí" k optimálnímu řešení. Můžete použít genetický algoritmus pro řešení problémů, které nejsou vhodné pro standardní optimalizační algoritmy, včetně problémů, které účelová funkce je discontinu-organizační jednotky, nediferenciované, stochastické, nebo vysoce nelineární. [62]

5.9 Pattern search Algoritmus

(PS) je rodina numerických optimalizačních metod, které nevyžadují sklon problému, který musí být optimalizován. Z tohoto důvodu může být použit PS na funkcích, které nejsou kontinuální nebo diferencovatelná. Tyto metody optimalizace jsou také známé jako metody přímého vyhledávání, derivátových, nebo černé-krabice. Jméno, vzor hledání, byl vytvořen Hooke a Jeeves. Brzy a jednoduchý PS na usměvavý je přičítán Fermi a Metropolis, když pracoval v Los Alamos National Laboratory, jak je popsáno Davidon , který shrnuje algoritmus následujícím způsobem: Lišily je jeden teoretický parametr najednou kroky ve stejném rozsahu. Pokud žádné takové zvýšení nebo snížení v průběhu jednoho parametru dále zlepšuje kondici, aby údaje o experimentálních, že poloviční velikost kroku opakoval postup, dokud kroky byly považovány za dostatečně malé.

5.9.1 Konvergence

Konvergentní metoda vzor-hledání byla navržena Yu, který ukázal, že se sblížil s použitím teorie pozitivních základů. Později, Torczon, Lagarias, a spoluautoři, používá kladný základ techniky dokázat, konvergence jiným způsobem vzor vyhledávání na určitou třídu funkcí. Mimo těchto tříd, vzor vyhledávání je heuristický, který může poskytnout užitečné přibližné řešení nějakého problému, ale může selhat na ostatní. Mimo těchto tříd, vzor hledání není iterační metoda, která konverguje k řešení; opravdu, mohou metody vzor vyhledávání konvergovat k non-stacionární body na některých poměrně krotké problémy.

5.9.2 Příklad použití

- hledání Zlatý řez koncepčně podobá PS ve svém zúžení vyhledávání-řady, pouze pro jednodimenzionální hledání prostorů.
- Metoda Nelder-Mead. Simplexová metoda koncepčně podobá PS ve svém zúžení vyhledávání-rozsah pro multi-dimenzionální hledací prostor, ale činí tak tím, že udržuje $n + 1$ body pro n -rozměrný hledací prostor, zatímco PS metody $2n + 1$ bodů (centrální bod a dva body v každém rozměru).
- Luus-Jaakola vzorky z rovnoměrných rozložení kolem aktuální polohy používá jednoduchý vzorec pro exponenciálně snižující rozsah odběru vzorků.
 - Náhodné vyhledávání související rodina optimalizačních metodách, které vzorek z hyper koule v okolí aktuální pozice.
- Náhodná optimalizace je příbuzný rodiny optimalizačních metod, které vzorek z normálního rozdělení okolí aktuální pozice. [63]

II. PRAKTICKÁ ČÁST

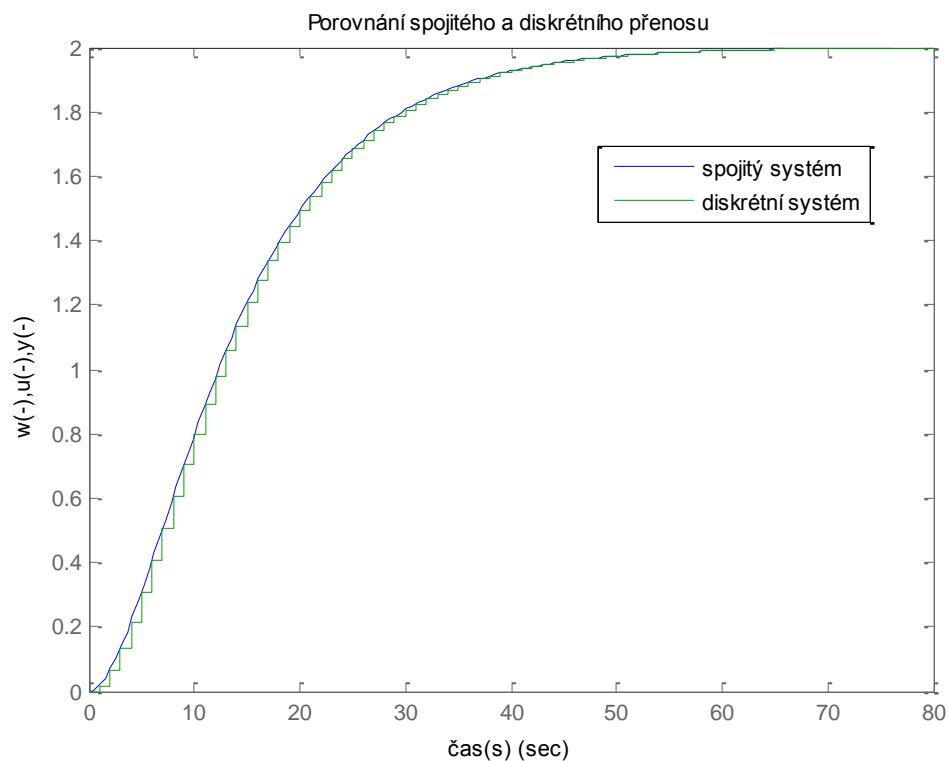
6 VÝPOČETNÍ ČAS ALGORITMŮ

Pro posouzení výpočetního času jsem si zvolil přenos 2. řádu jako referenční. Pro tento přenos se dále budou porovnávat jednotlivé algoritmy:

- Algoritmus LQ
- Algoritmus GPC

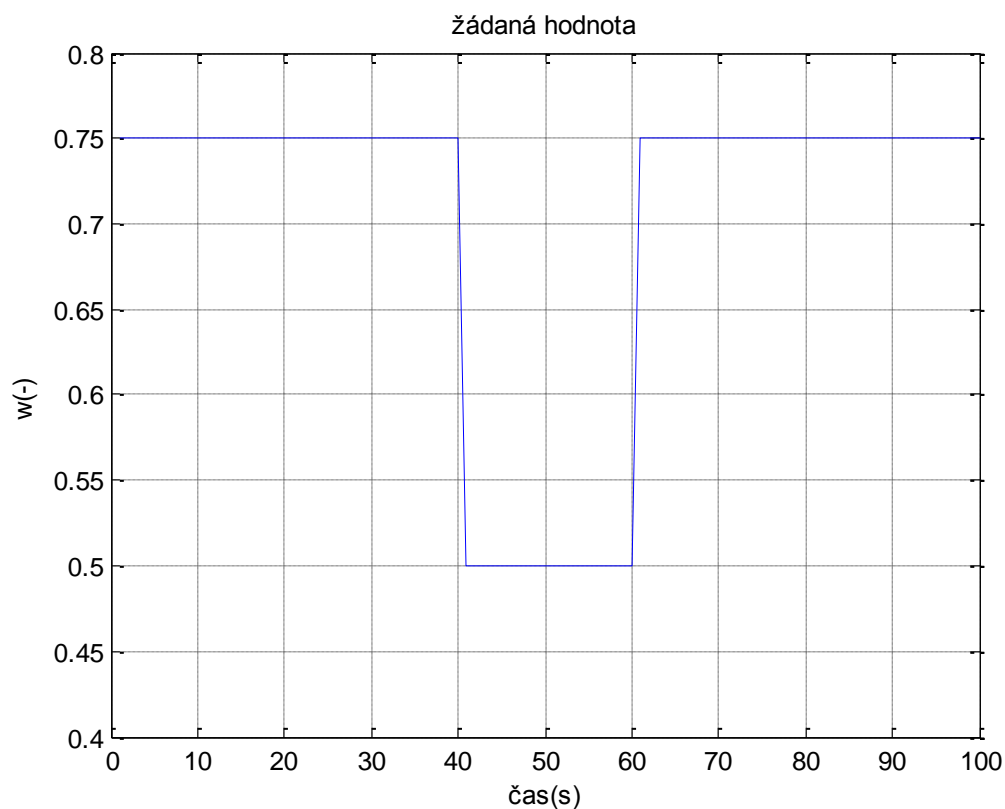
Přenos: $G = \frac{2}{50 s^2 + 15 s + 1}$. Pro periodu vzorkování $T = 1s$ je pak přenos:

$$Gd = \frac{0,01811 z + 0,01639}{z^2 - 1,724 z + 0,7408}$$



Obrázek 4 – Porovnání diskretizace systému se spojitým systémem

Zde na tomto grafu vidíme, že perioda vzorkování je zvolena dobře. Periody 1s dostatečně vystihuje dynamiku systému. Lambda byla zvolena 2 pro všechny simulace.



Obrázek 5 – žádaná hodnota, která je pro všechny algoritmy stejná

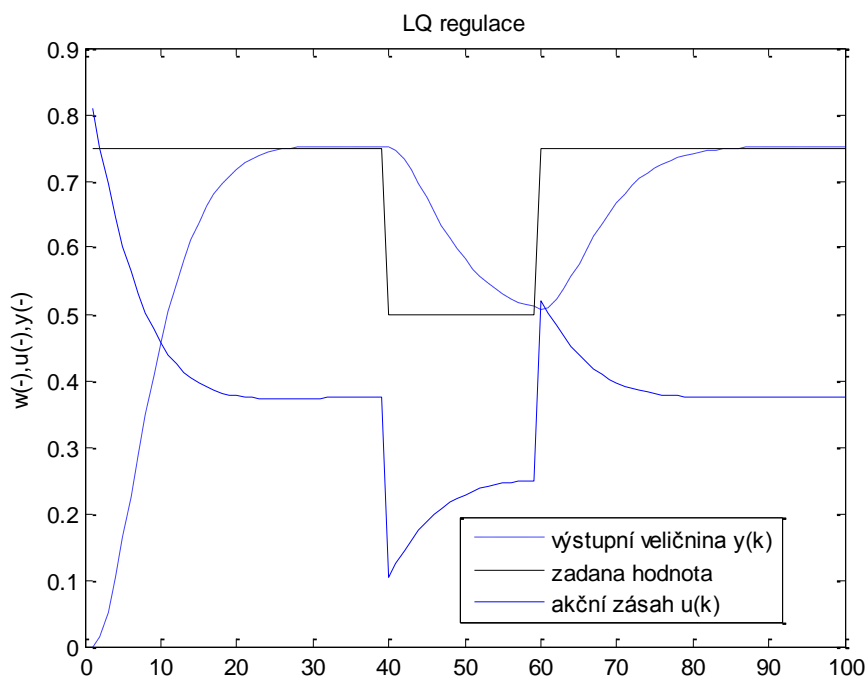
Jednotlivé výpočty budou provedeny 1000 krát po sobě, aby se předešlo chybám, typu počítač instaloval aktualizace. Výpočetní čas je změřen pomocí příkazu v matlabu tic a toc.

6.1 Algoritmus LQ

Tabulka 1 – Výpočetní čas

spuštění	výpočetní čas (s)
1	0,935887
2	0,771494
3	0,741644
4	0,781330
průměr	0,80758875

Zde jak vidíme tak první spuštění trvalo nejdéle a to nejspíše z důvodu prvního načtení do paměti počítače. Další měření mělo jen minimální odchylky. Kód matlabu bude jednak na příloze PI a také na přiloženém CD



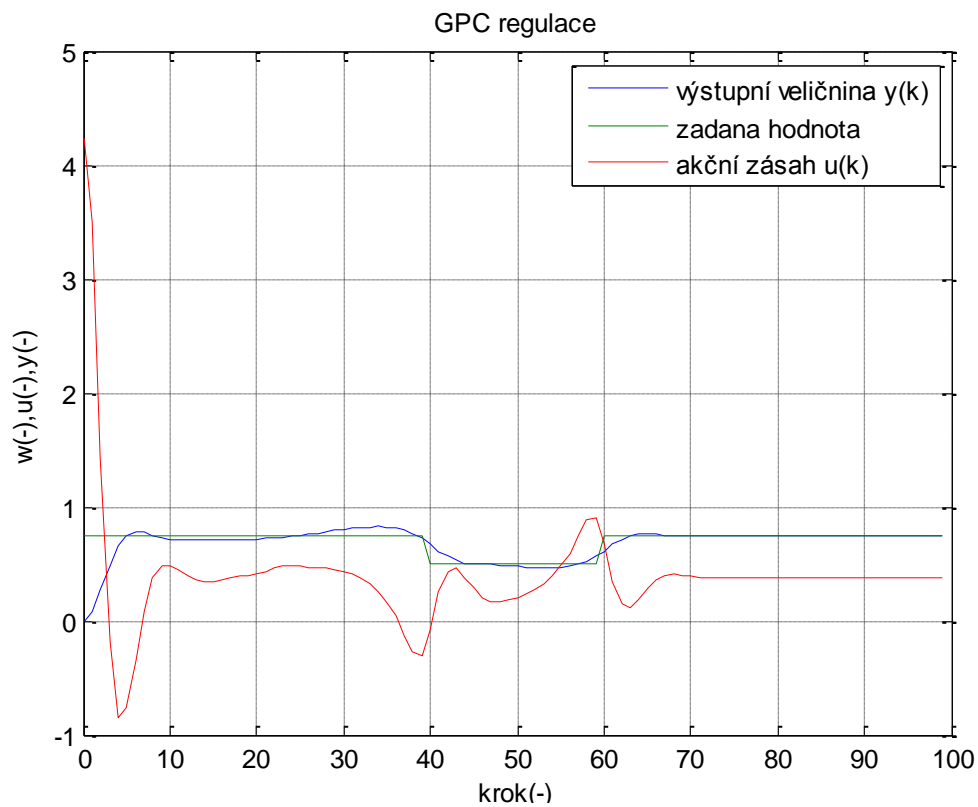
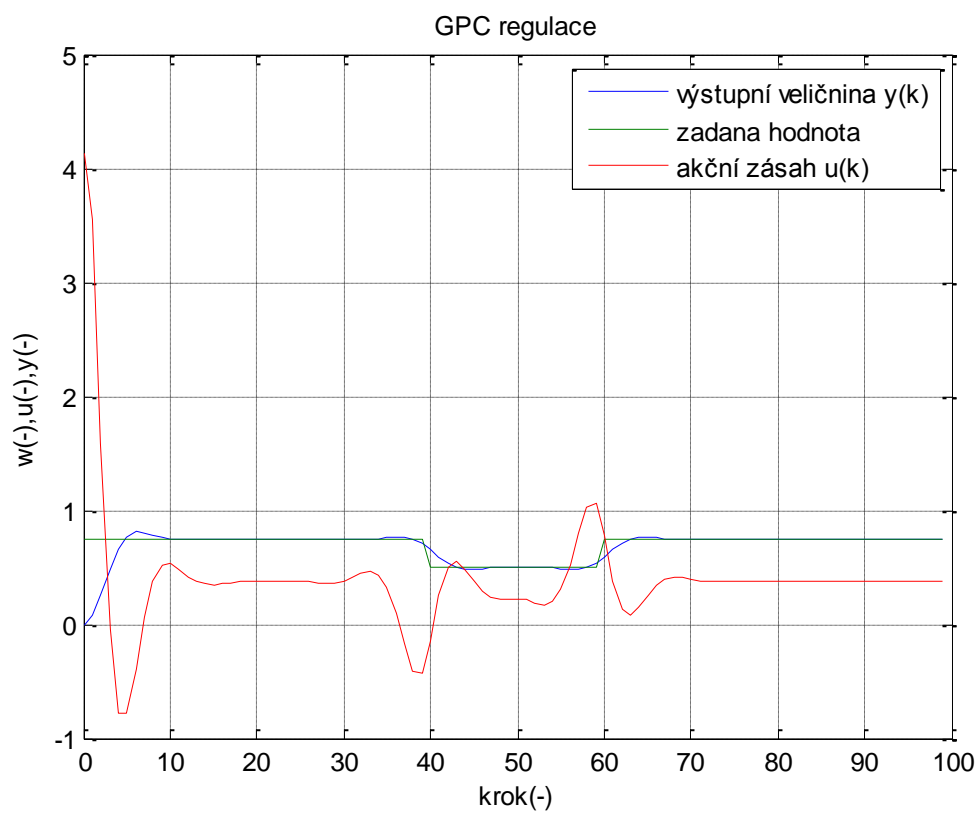
Obrázek 6 – Výsledek simulace algoritmu LQ pro měření výpočetního času

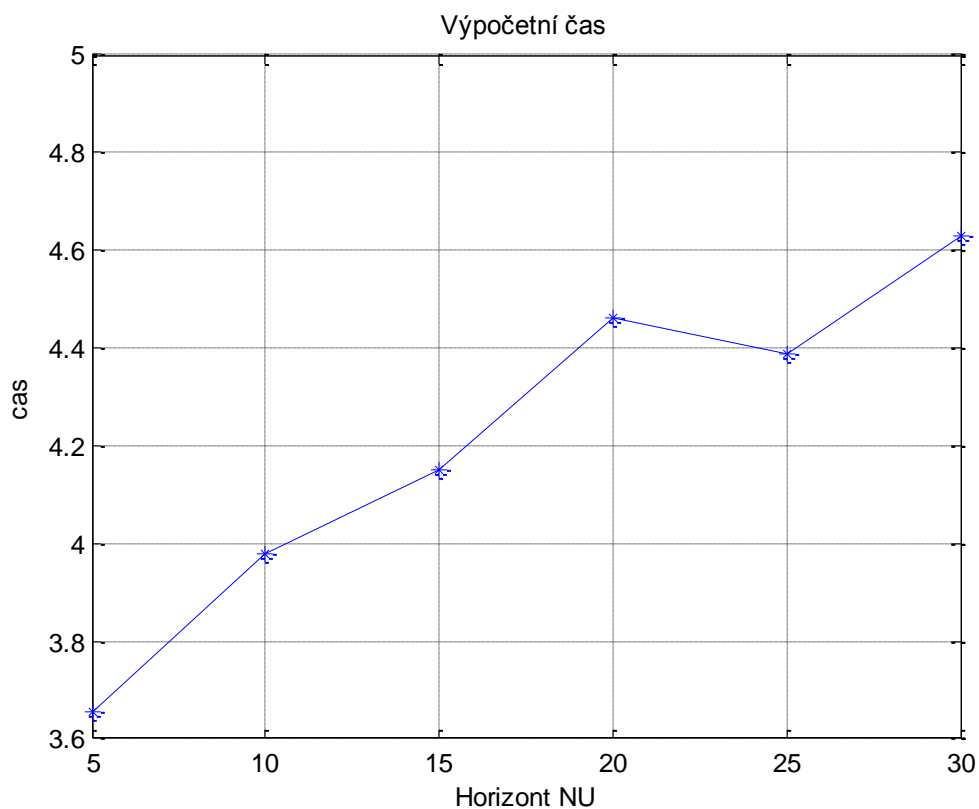
6.2 Algoritmus GPC

Výpočetní čas algoritmu jsem vyhodnocoval pomocí MPC toolbox Malabu, který mohl mít na mém PC pomalejší čas běhu. Windows je multitaskingový systém tudíž není zajištěno, že ve stejný čas běžel pouze jeden proces. Kód Malabu bude v příloze P II a na CD. Výsledky jednotlivých změn N_2 a N_u budou v tabulkách 2 a 3.

Tabulka 2 – Výpočetní čas při konstantním $N_2 = 40$ a změně horizontu N_u

spuštění	$N_u=5$	$N_u=10$	$N_u=15$	$N_u=20$	$N_u=25$	$N_u=30$
1	3,767021	3,596388	4,282756	4,191375	4,210350	4,476788
2	3,631038	3,753513	4,463274	4,470636	4,221219	4,670767
3	3,656156	4,426695	3,916184	4,855042	4,641520	4,656146
4	3,566616	4,138824	3,943313	4,324610	4,485646	4,720716
průměr	3,655208	3,978855	4,151382	4,460416	4,389684	4,631104

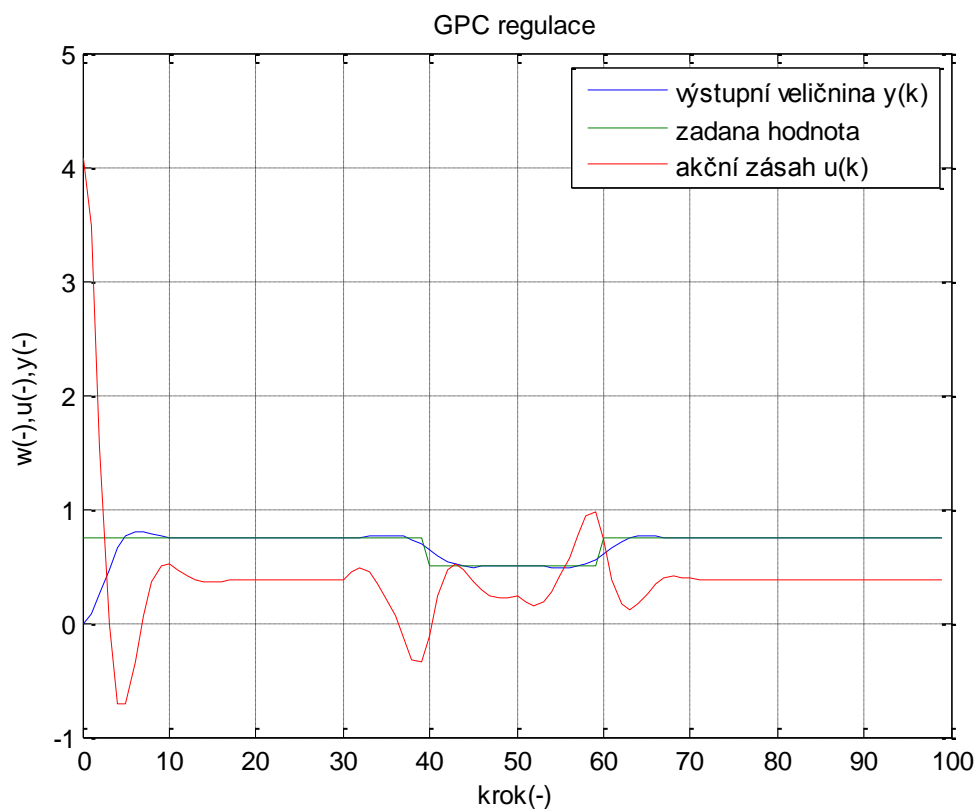
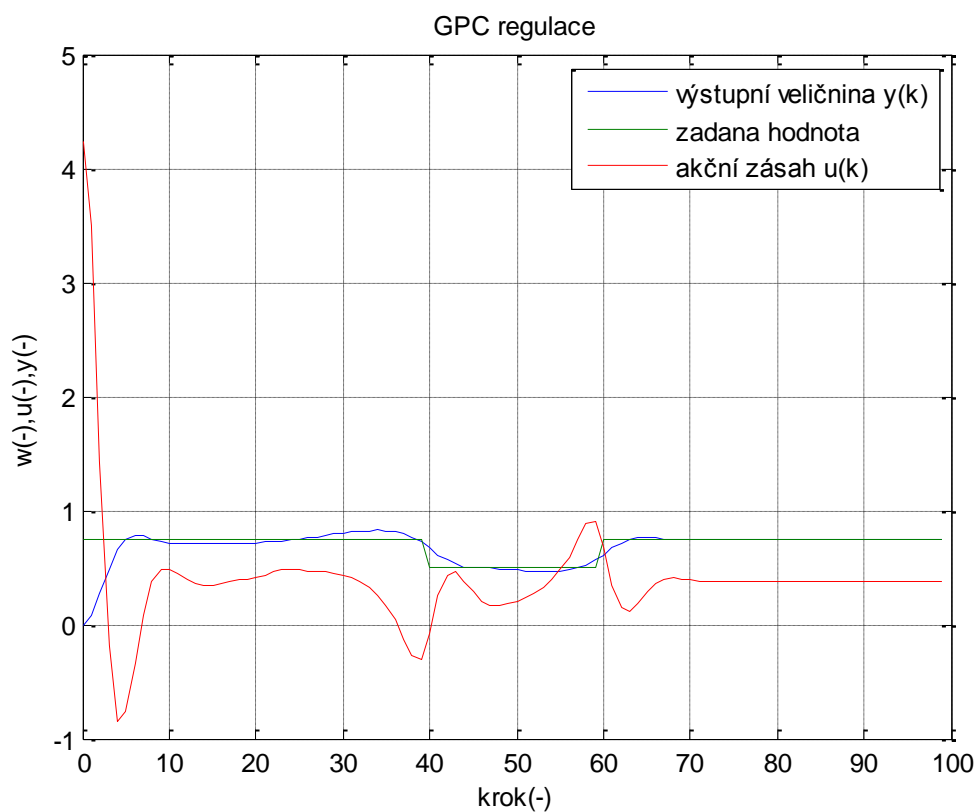
Obrázek 7 – $N_2=40$ $NU=5$ Obrázek 8 – $N_2=40$ $NU=30$

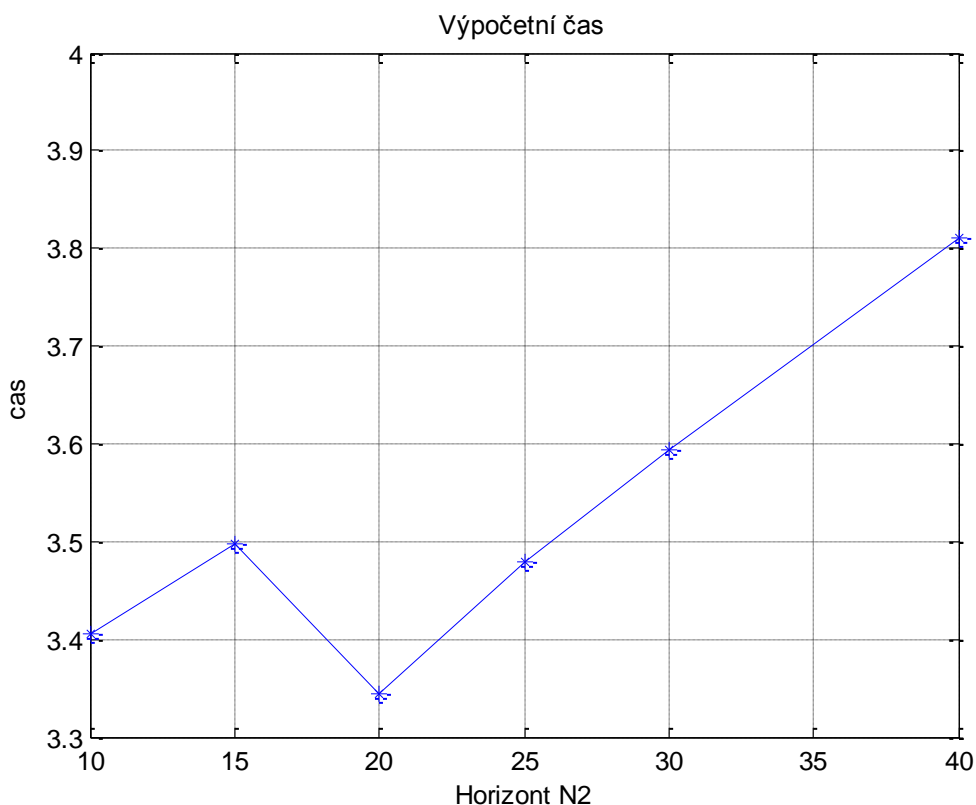


Obrázek 9 – Graf závislosti jak se mění výpočetní čas při konstantním horizontu N2 a změnách horizontu N2

Tabulka 3- Výpočetní čas při konstantním NU = 5 a změně horizontu N2

spuštění	N2=10	N2=15	N2=20	N2=25	N2=30	N2=40
1	3,390333	3,367217	3,335532	3,430973	3,900747	4,730843
2	3,313561	3,611167	3,361326	3,545133	3,438655	3,600313
3	3,369048	3,573174	3,290294	3,516417	3,516959	3,419224
4	3,552617	3,436879	3,389047	3,420623	3,520768	3,494056
průměr	3,40639	3,497109	3,34405	3,478287	3,594282	3,811109

Obrázek 10 – GPC při horizontech $NU = 5$ $N2=10$ Obrázek 11 - GPC při horizontech $NU = 5$ $N2=10$ $NU = 5$ $N2=40$



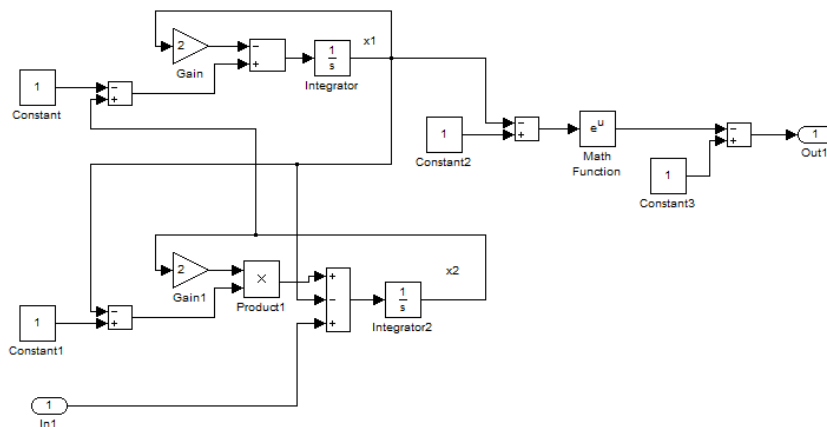
Obrázek 12 – Graf závislosti jak se mění výpočetní čas při konstantním horizontu NU a změnách horizontu N2

Zhodnocení:

Když se podíváme na tabulky 1, 2 a 3, tak vidíme, že algoritmus LQ je na výpočet rychlejší než GPC, protože nepoužívá při výpočtu znalost budoucích hodnot žádané hodnoty. U algoritmu GPC zase můžeme měnit horizonty a tím i ovlivnit průběh regulované veličiny tak, že začne reagovat před skokem žádané veličiny. Tabulky 2 a 3 znázorňují, jak se mění výpočetní čas, když měníme jeden horizont a druhý necháme stejný. Grafické znázornění je na obrázcích 8 a 11. Jak je zřejmé, výpočetní čas v závislosti na změně horizontu není lineární. Obrázky 6 a 7 ukazují, jak se chová systém, když jeden horizont necháme stejný a druhý měníme - horizont NU. Obrázky 10 a 9 zase znázorňují, jak se chová systém při změně horizontu N2 při konstantním horizontu NU.

7 NELINEÁRNÍ ŘÍZENÍ POMOCÍ LINEARIZACE

Jako vhodný nelineární systém pro zkoušení prediktivního řízení jsem si zvolil model, který matlab nabízí. Jeho simulinkové schéma je na obrázku 13. Vybral jsem ho z důvodu, že je dohledatelný a není extra složitý. Tento model systému byl uložen jako testovací v matlabu, což je výhodné pro testování více lidmi.

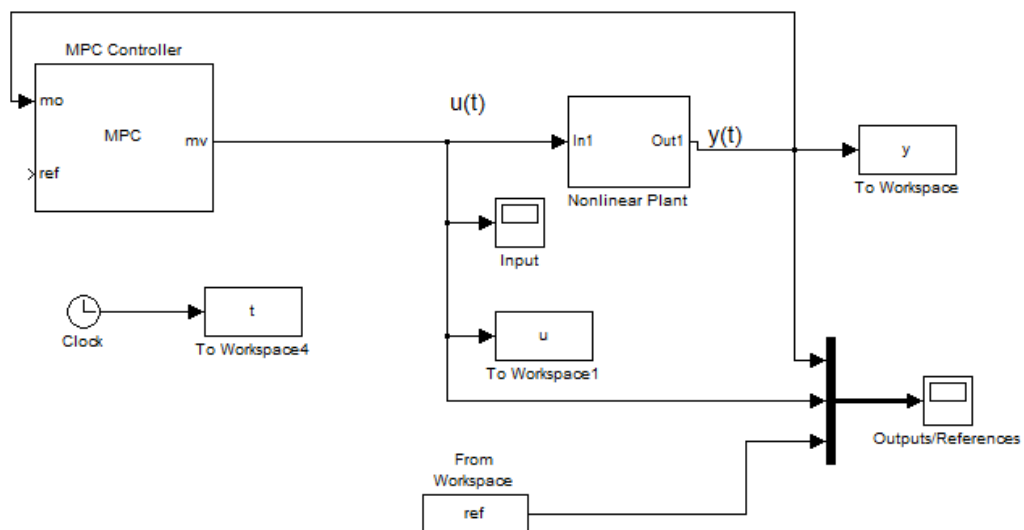


Obrázek 13 – nelineární systém

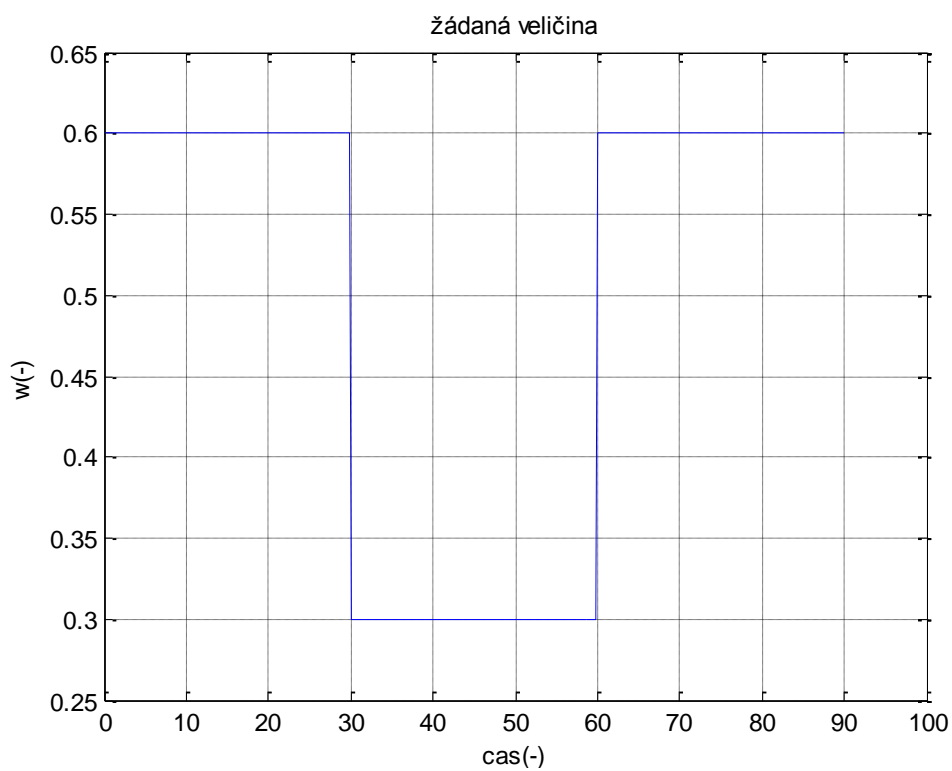
Jakmile jsem měl schéma, bylo potřeba zjistit jakou má statickou charakteristiku a limity. Horní limit pro tento systém je 1.

7.2 Řízení nelineárního systému

U tohoto schématu je výhodné to, že mohu zaměnit jakýkoliv systém. Tudiž můžeme použít pro řízení více systémů pouze výměnou systému. Toto schéma je zobrazeno na obrázku 14.

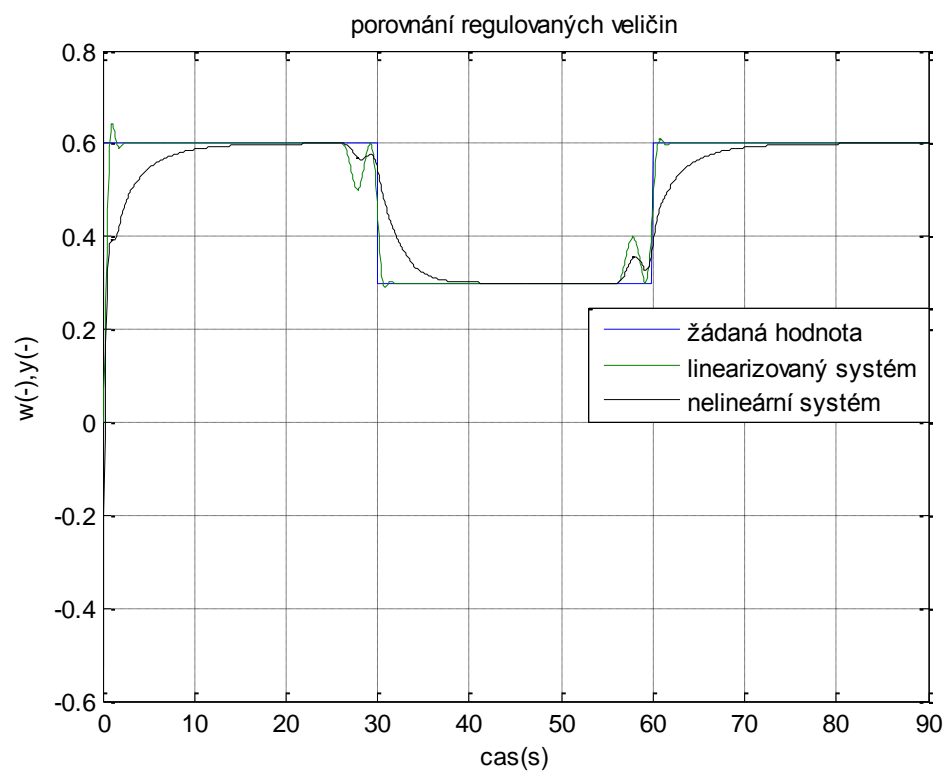


Obrázek 14 - schéma řízení nelineárního řízení

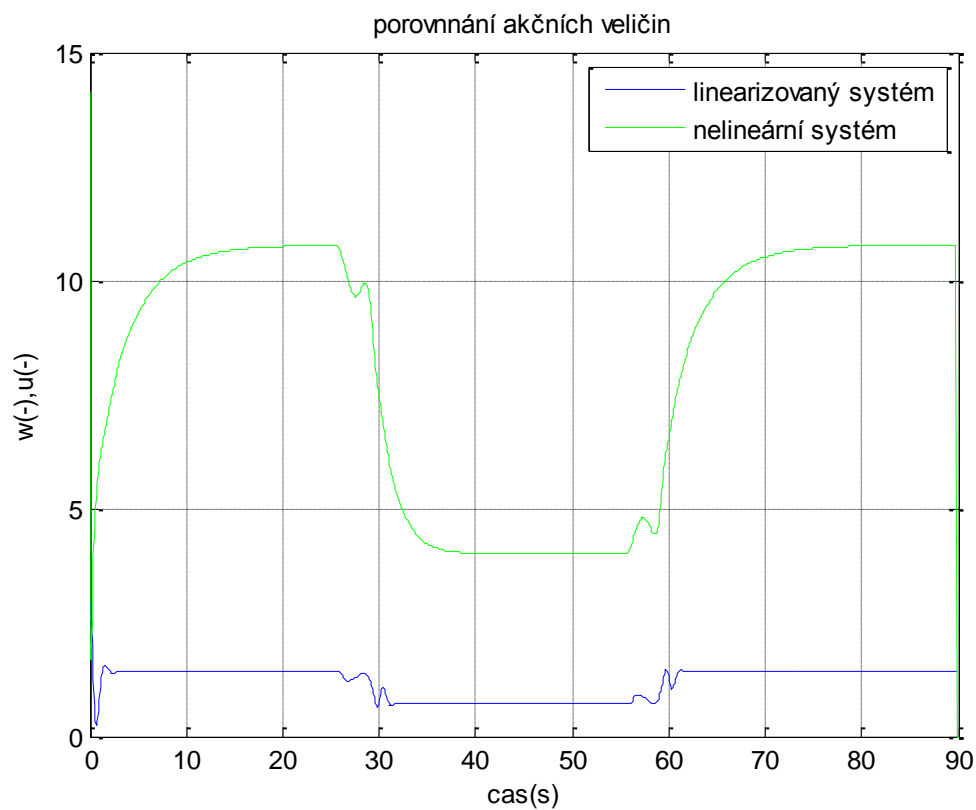


Obrázek 15 – žádaná hodnota

7.2.1 Řízení nelineárního systému při horizontech $N_U = 10$ $N_2 = 40$



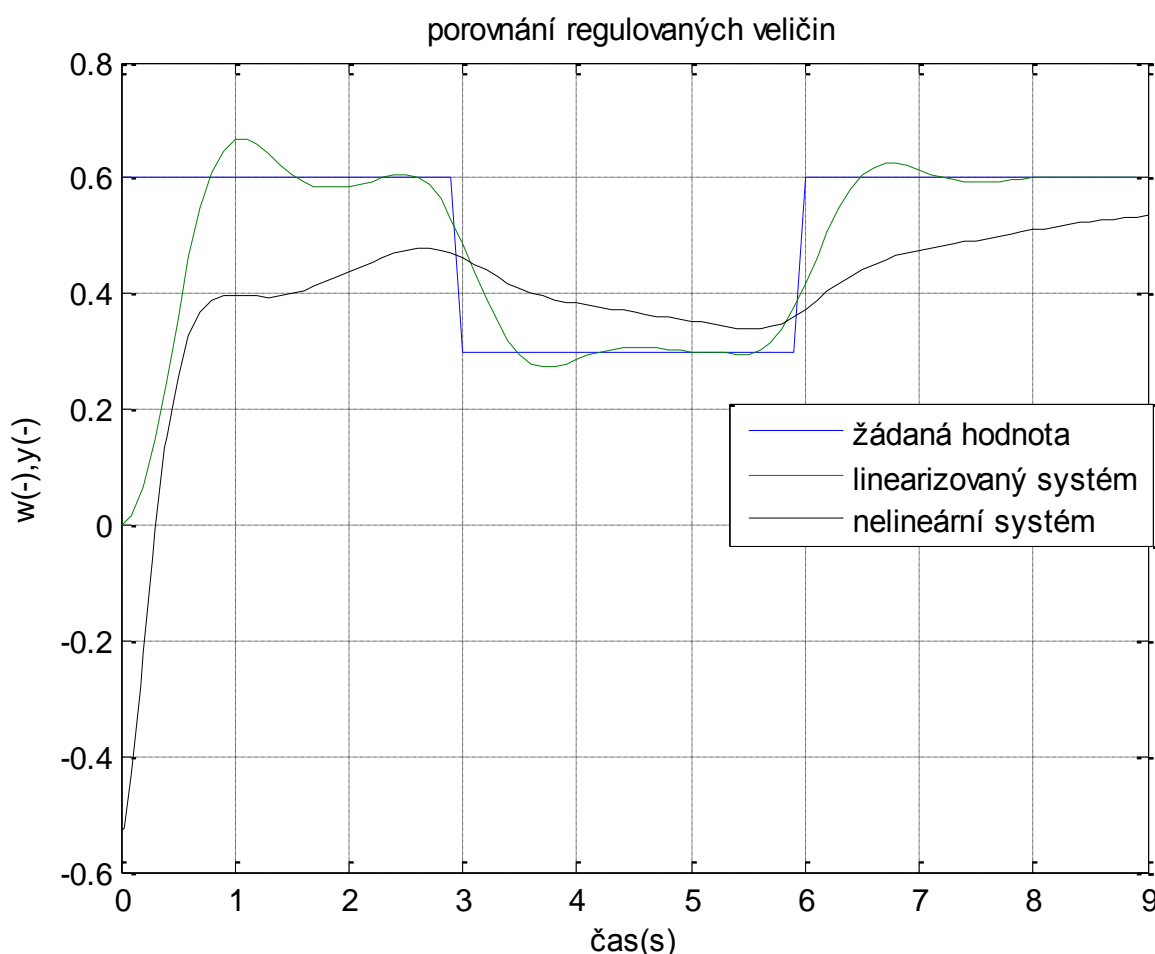
Obrázek 16 – graf porovnání regulovaných veličin



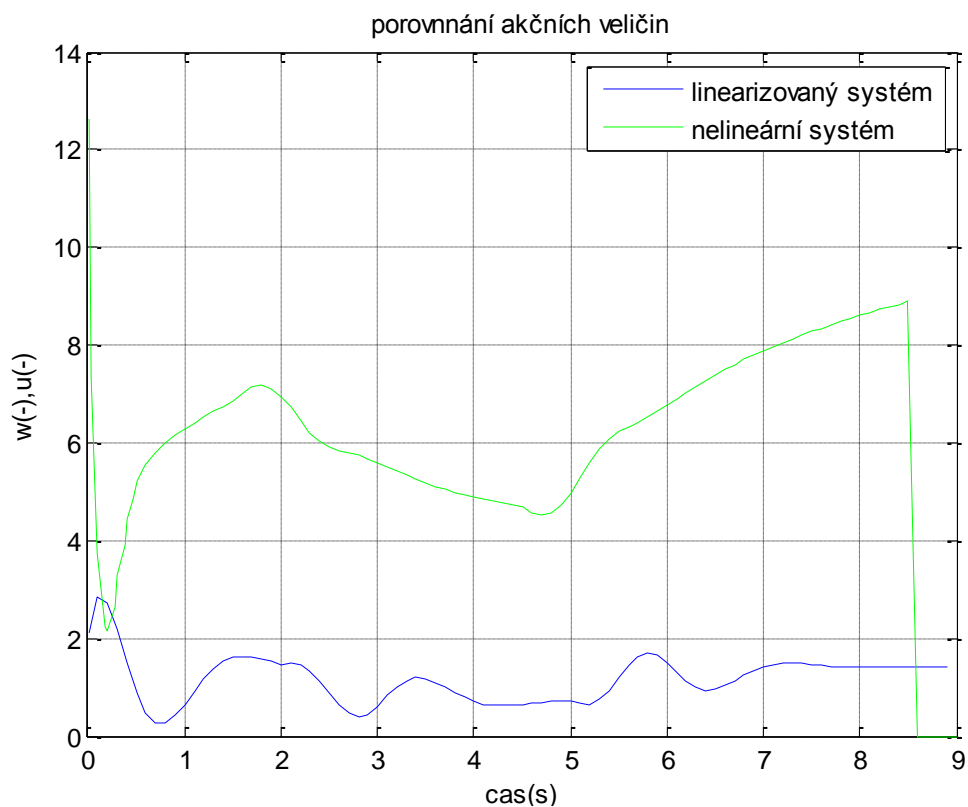
Obrázek 17– graf porovnání akčních veličin

Zhodnocení:

Zde máme 2 grafy, kde porovnávám řízení linearizovaného systému a nelineárního systému. Pro přehlednost jsem veličiny zobrazil ve 2 grafech. První graf porovnává regulované veličiny. Na obrázku 16 vidíme, že linearizovaný systém reaguje rychleji s menším překmitem i akční veličina má pozvolný průběh viz obrázek 17. To bude z důvodu, že regulace je navržena na tento systém. V případě, že navrhne regulaci na linearizovaný systém a připojíme nelineární, tak se bude chovat akční veličina, jak je ukázáno na obrázku 17. Přechod má aperiodický charakter a pomalejší reakce. Na žádanou hodnotu se dostane, ale pomaleji. Akční veličina má také průběh jak aperiodický systém. Jeho hodnoty zase dosahují větších hodnot.

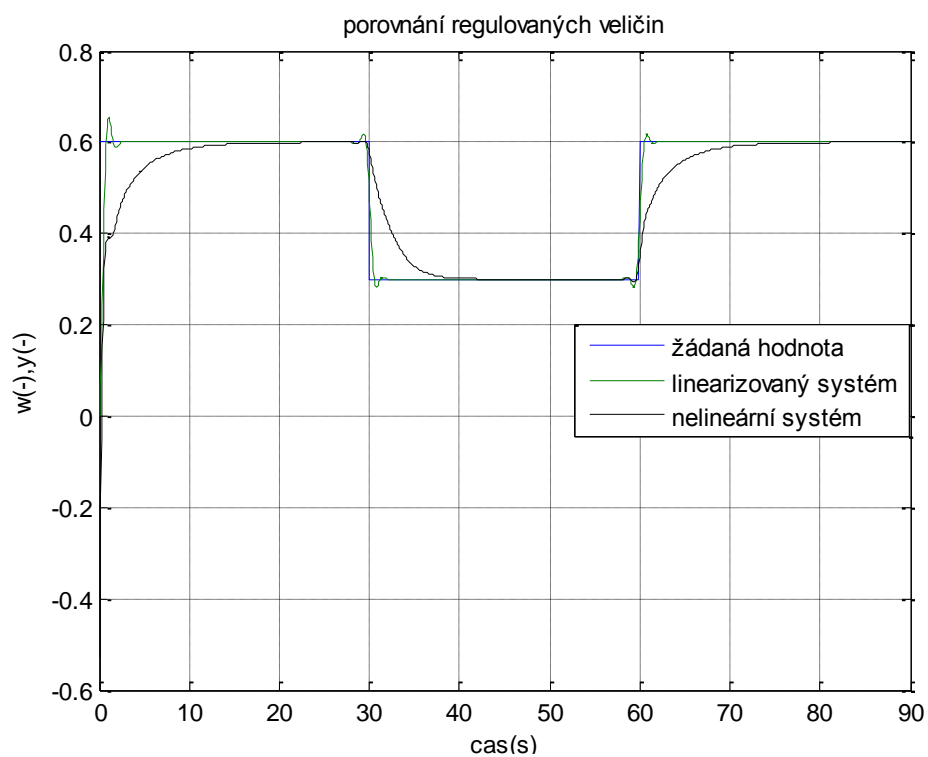
7.2.2 Řízení nelineárního systému při horizontech $NU = 10$ $N2 = 10$ 

Obrázek 18-porovnání regulovaných veličin při stejných horizontech

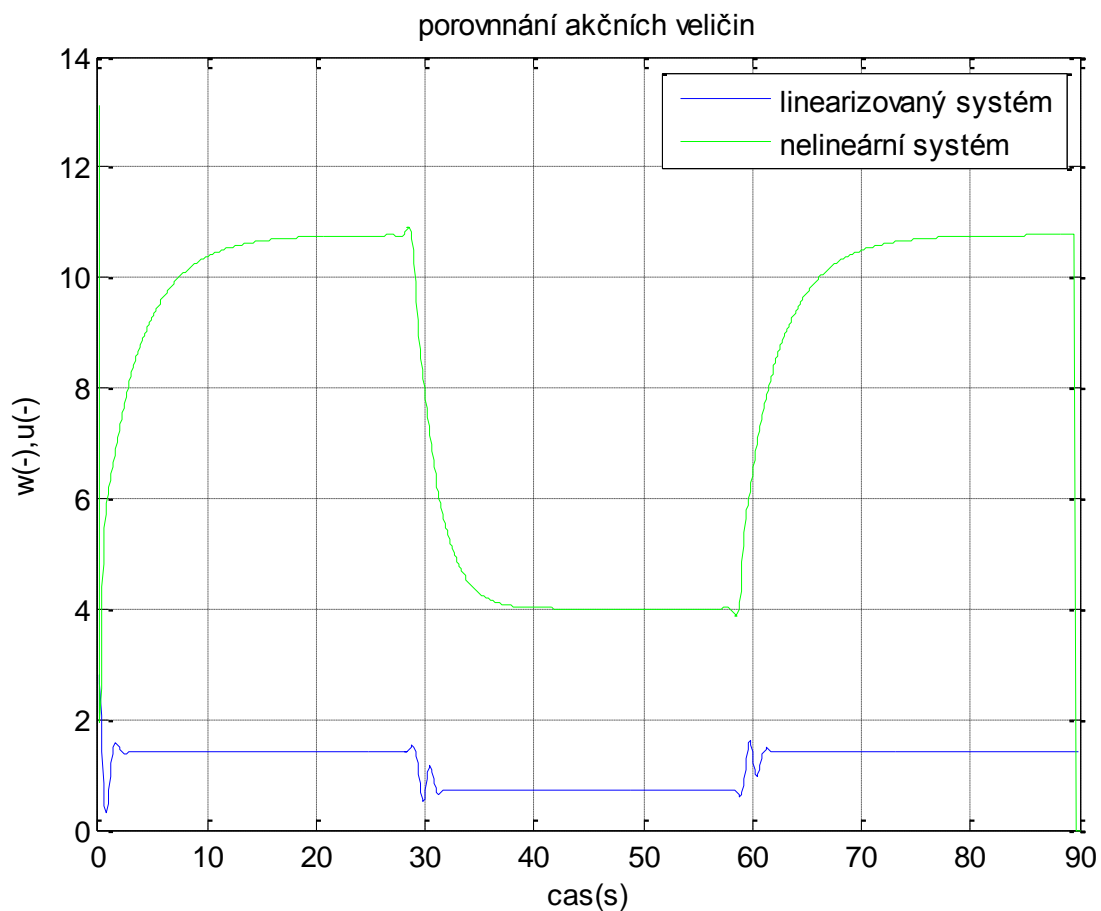


Obrázek 19 – Porovnání akčních veličin při stejných horizontech

7.2.3 Řízení nelineárního systému při horizontech $N_U = 30$ $N_2 = 40$



Obrázek 20 – graf porovnání regulovaných veličin



Obrázek 21 – graf porovnání akčních veličin

Zhodnocení:

Podobně je to jako u předchozích horizontů jen s tím rozdílem, že před změnou žádané hodnoty se systém nechová tak divoce jako v předchozím případě. Což je jistě způsobeno změnou horizontů a regulátor v tomto případě nemá takové problémy v optimalizaci.

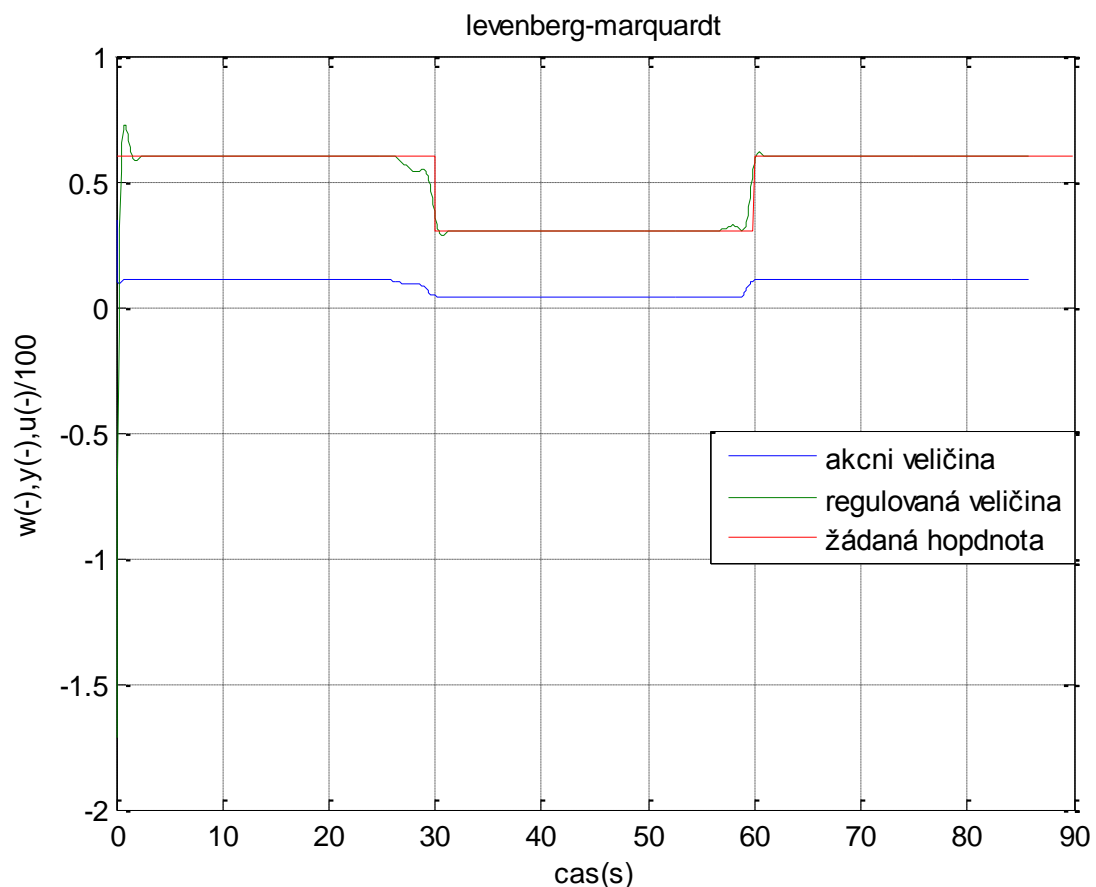
8 NELINEÁRNÍ ŘÍZENÍ POMOCÍ OPTIMALIZACE

V této kapitole je porovnání simulací a časů, kde pro jednotlivé metody je několik algoritmů. Nastavení jednotlivých simulací byla stejná stejně jako průběh žádané hodnoty. Pro všechny simulace byla použita perioda vzorkování 0,1s. Lambda byla nastavena na 1. V této kapitole se neprovádí linearizace modelu, ale používá se přímo návrh řízení založený na nelineárním modelu.

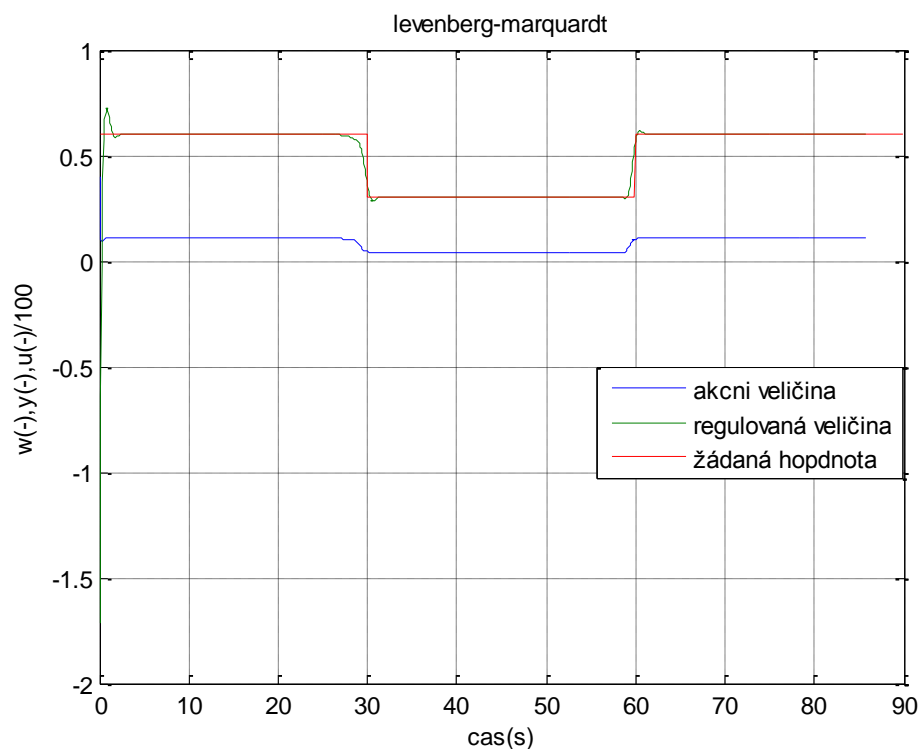
8.2 Nelineární čtverce pomocí příkazu `lsqnonlin`

Zde pro simulaci (výpočet) je možno volit z 2 algoritmů. Těmito algoritmy jsou levenberg-marquardt a trust-region – reflective. V matlabu do nastavení options v parametru Algorithm stačí napsat název algoritmu. Pro změření výpočetního času jsem použil příkazy `tic` a `toc`. Těmito příkazy jsem ohraničil výpočetní část algoritmu a tím jsem po skončení získal čas.

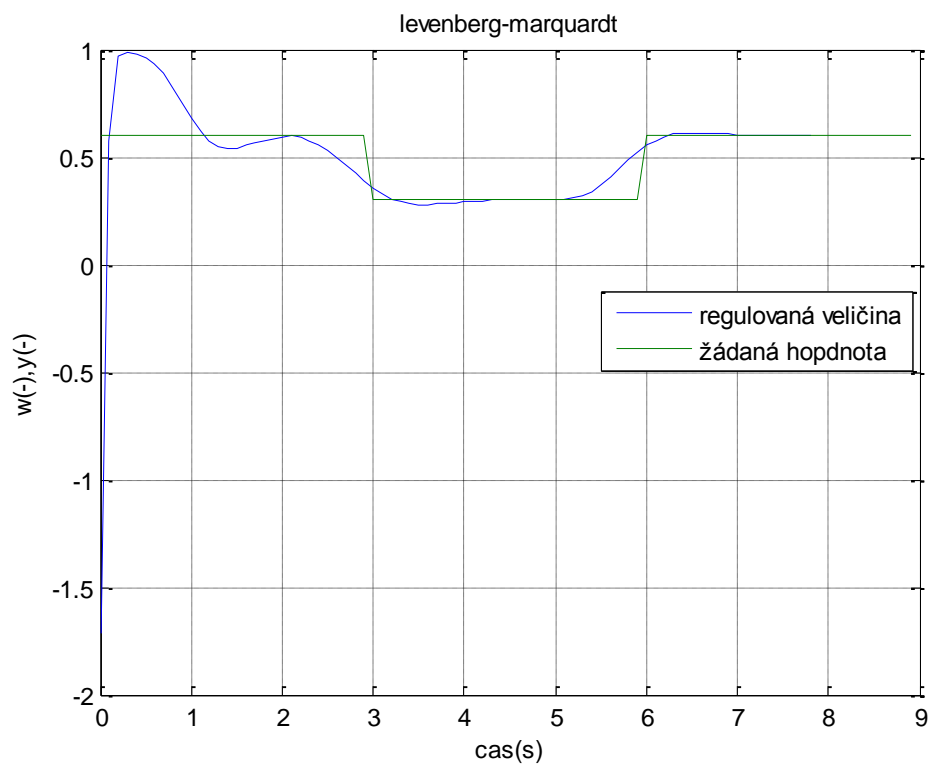
8.2.1 Simulace levenberg-marquardt při horizontech $N_U = 10$ $N_2 = 40$



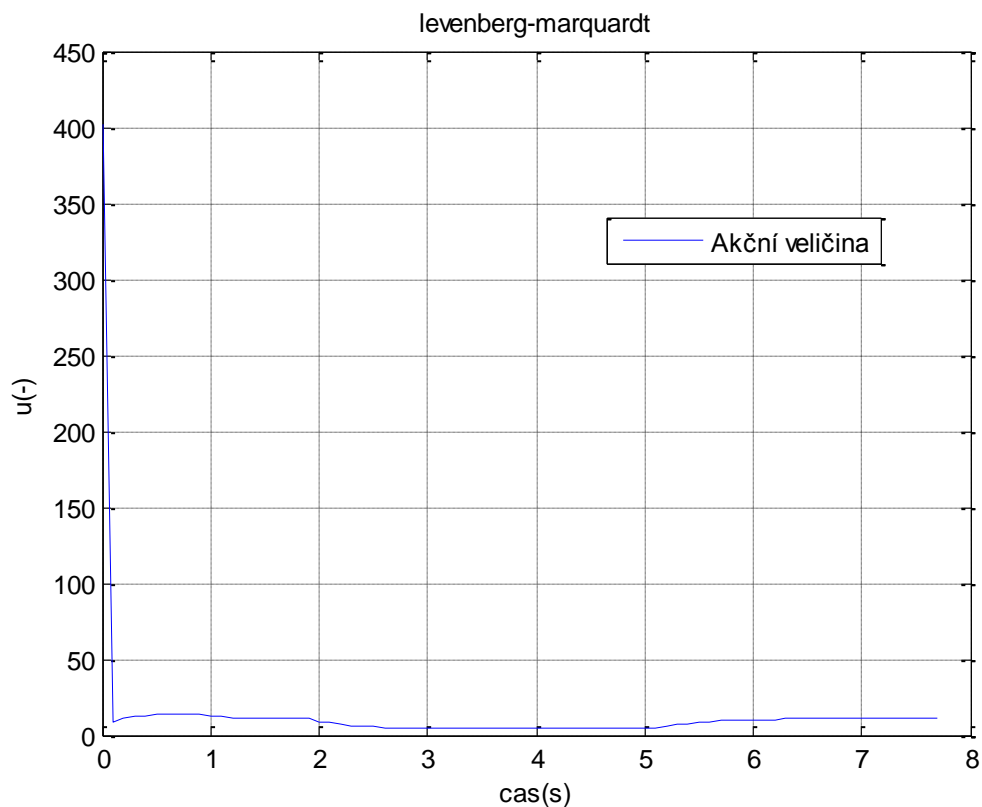
Obrázek 22 - regulace pomocí algoritmu levenberg-marquardt

8.2.2 Simulace levenberg-marquardt při horizontech $NU = 30$ $N2 = 40$ 

Obrázek 23 - regulace pomocí algoritmu levenberg-marquardt

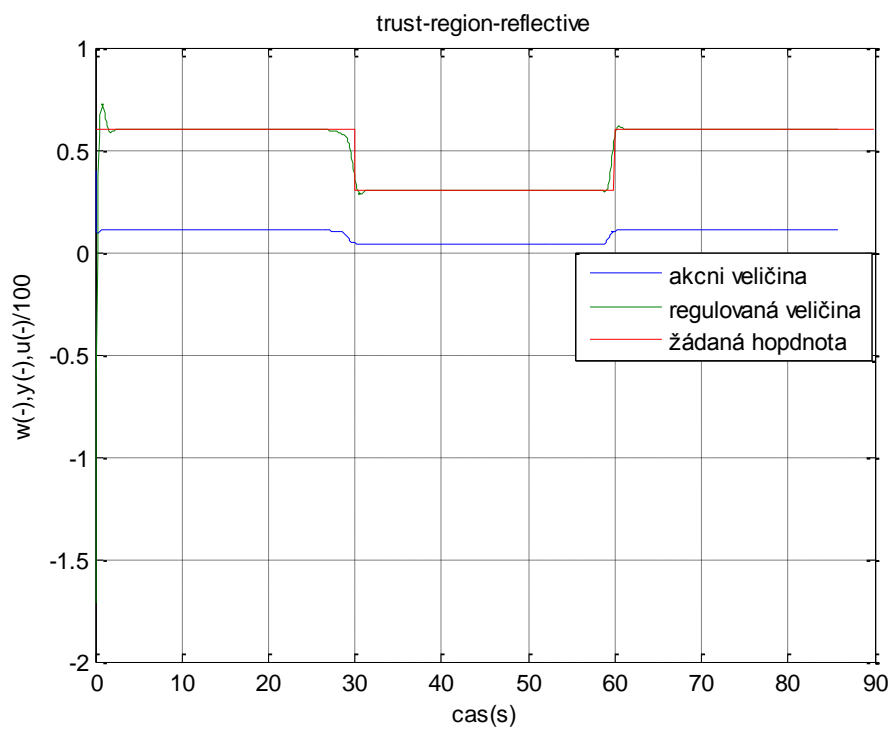
8.2.3 Simulace levenberg-marquardt při horizontech $NU = 10$ $N2 = 10$ 

Obrázek 24 - regulace pomocí algoritmu levenberg-marquardt

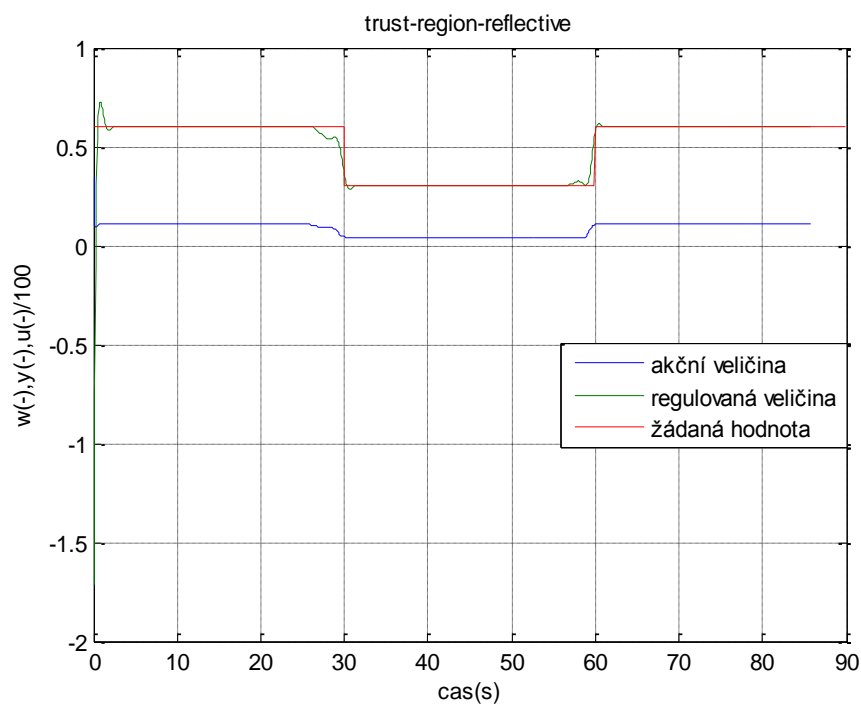


Obrázek 25 – Akční zásah při levenberg – marquardt algoritmu

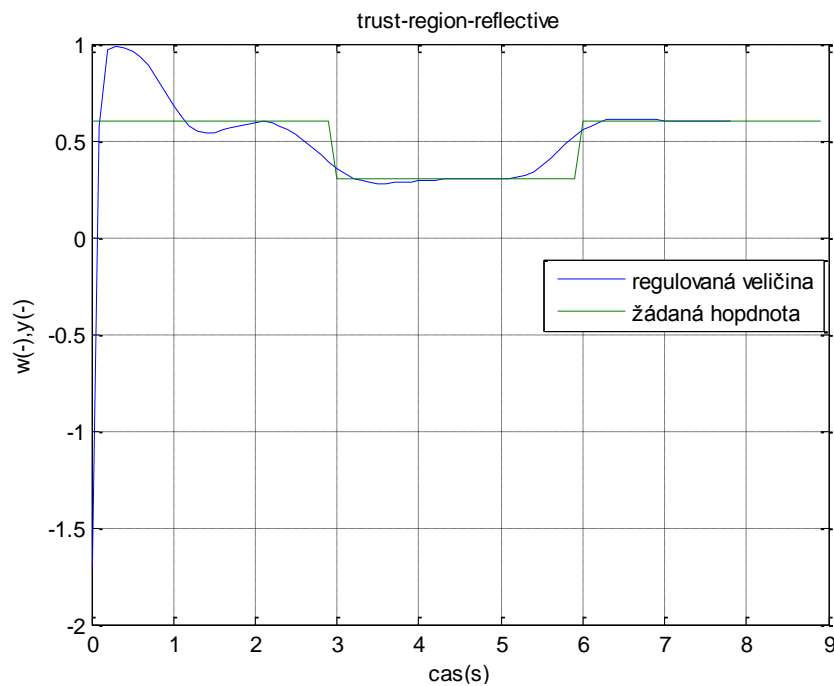
8.2.4 Simulace trust-region – reflective při horizontech $NU = 30$ $N2 = 40$



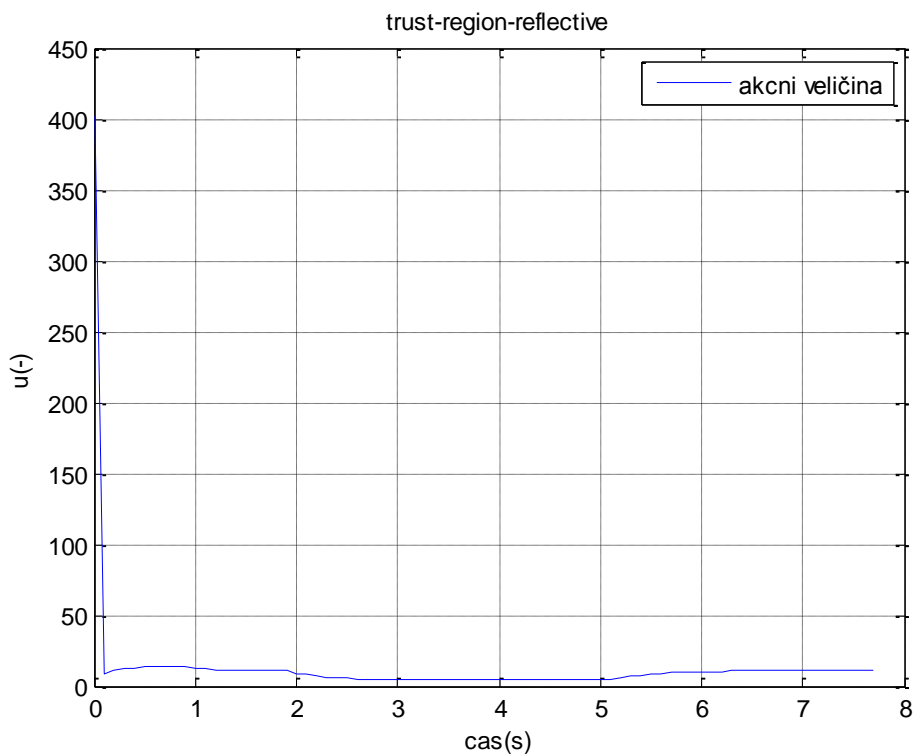
Obrázek 26 – regulace pomocí algoritmu trust-region-reflective

8.2.5 Simulace trust-region – reflective při horizontech $NU = 10$ $N2 = 40$ 

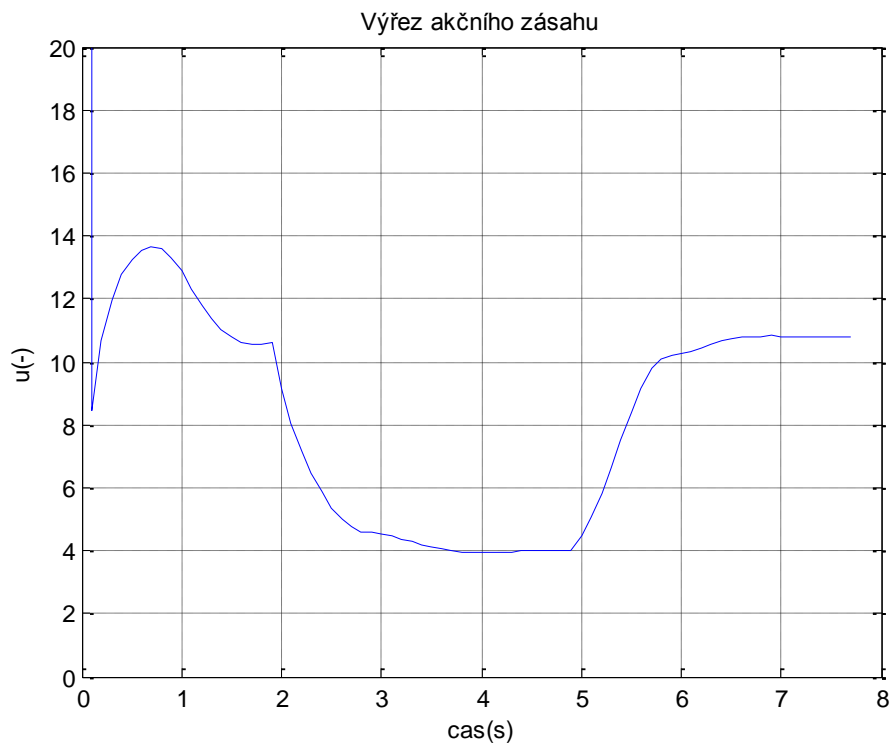
Obrázek 27 - regulace pomocí algoritmu trust-region-reflective při horizontech
 $NU=10$ $N2 = 40$

8.2.6 Simulace trust-region – reflective při horizontech $NU = 10$ $N2 = 10$ 

Obrázek 28 - regulace pomocí algoritmu trust-region-reflective při horizontech
 $NU=10$ $N2 = 10$



Obrázek 29 – průběh akčního zásahu pomocí algoritmu trust-region-reflective při horizontech $NU=10$ $N2 = 10$



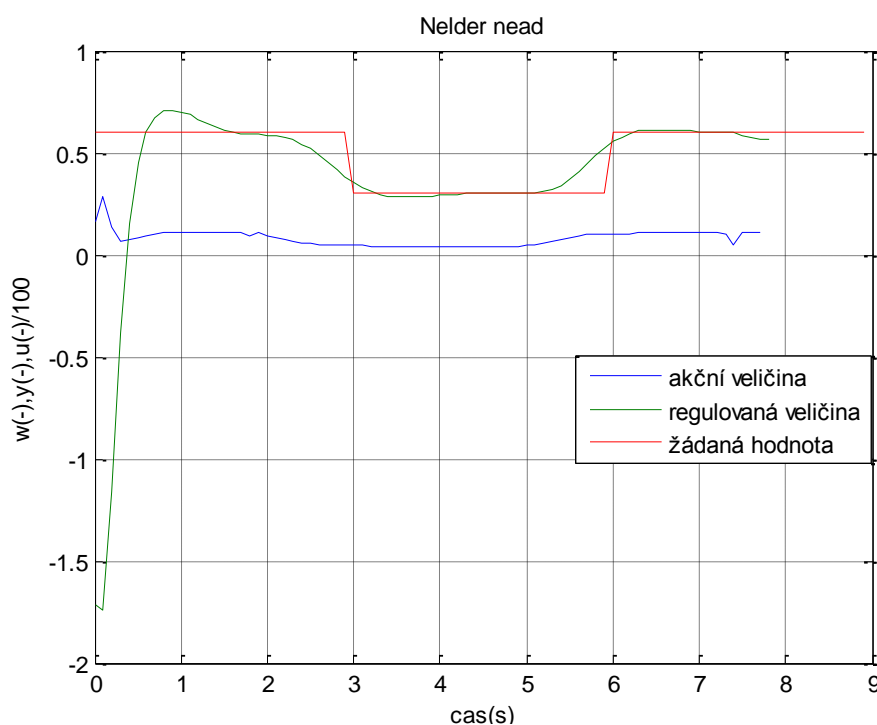
Obrázek 30 – Výřez akčního zásahu

Zhodnocení:

Jak můžeme vidět, tak oba algoritmy dávají stejný výsledek. Průběh regulované veličiny je s malým překmitem u obou simulací při delších horizontech. Stejně jako akční veličina nemá nikde kromě začátku divoké průběhy. Při stejných horizontech a zkrácení regulačního času je vidět, že došlo ke zhoršení regulace. Na začátku je velký regulační překmit a regulovaná veličina se pro první skok nestačila ustálit. Pro další skoky už překmit není tak velký a stačila se ustálit. Řešení pro ustálení u prvního skoku je prodloužit čas. Takže jako hlavní rozdíl mezi těmito algoritmy shledávám, že trust- region- reflective je rychlejší. Pro ověření uvedu oba kódy s simulingovým modelem do příloh jak na CD, tak i na konec této práce. Dále na konci této kapitoly je tabulka, která porovná časy výpočtu optimalizací pro různé algoritmy.

8.3 Pomocí příkazu Fminsearch

Tento příkaz nemá na výběr tolik algoritmů jako v předchozím případě lsqnonlin. Zde je možný pouze jeden a to Nelder-nead simplex. Stejně jako vždy na konec uvedu do příloh kod matlabu, aby každý kdo má v matlabu optimization toolbox mohl změřit na svém počítači výpočetní čas.

8.3.1 Simulace nelder-nead simplex při horizontech $N_U = 10$ $N_2 = 10$ 

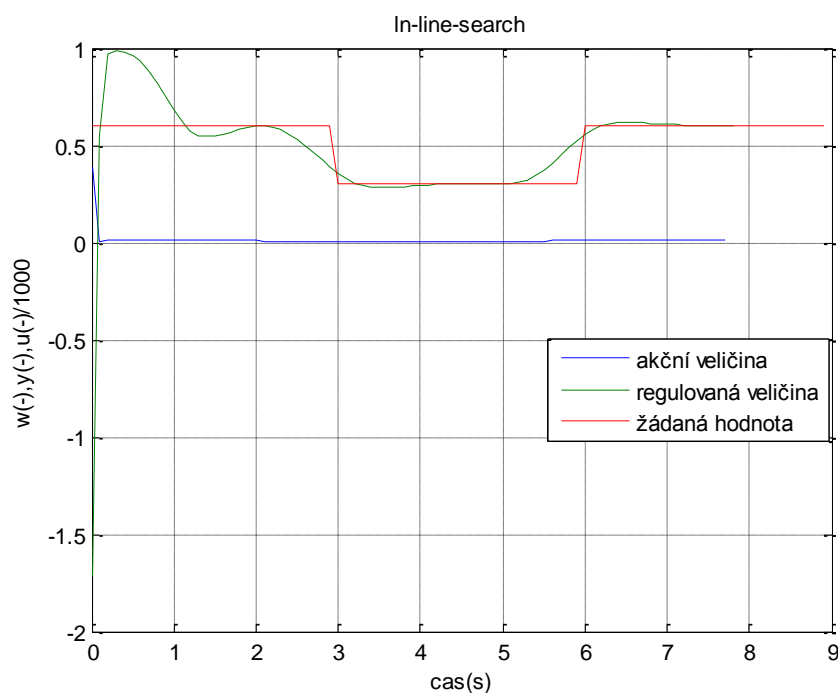
Obrázek 31 – simulace pomocí fminsearch

Zhodnocení:

Tento algoritmus na rozdíl od předešlých na začátku nemá takový regulační překmit. Pro další regulaci vidíme, že se stačila ustálit. U dalších skoků jsou překmity nepatrné. Nevýhodou je, že tomuto algoritmu trvá dlouho, než optimalizuje akční zásahy viz. tabulka porovnání. Z důvodu tento algoritmus nemá v sobě zabudované kvadrát regulační odchylky.

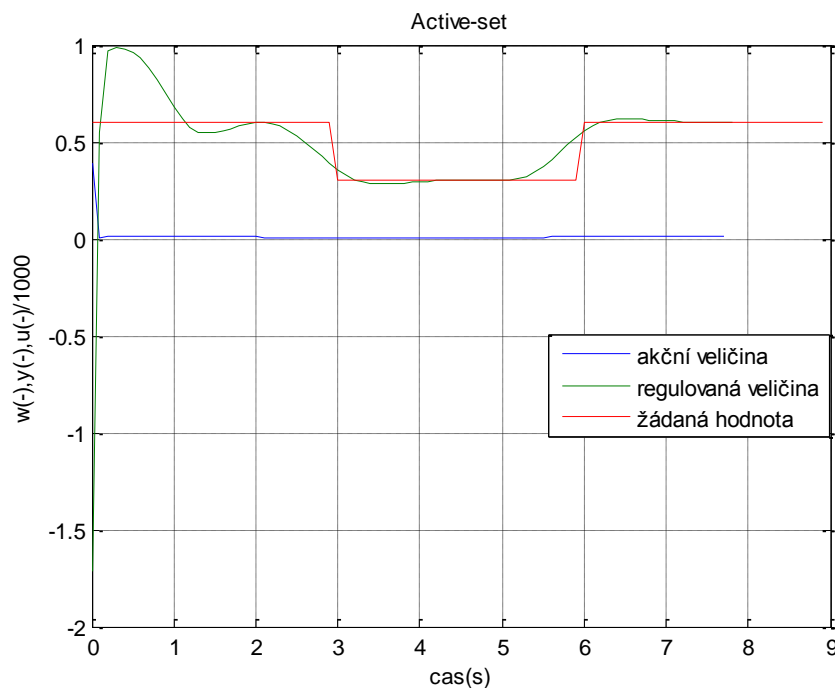
8.4 Pomocí příkazu Fminunc

Zde je možno volit mezi In- line, active-set a sqp algoritmy. Všechny algoritmy budou dávat naprosto stejné výsledky, ale každému bude výpočet trvat pokaždé jiný čas.

8.4.1 Simulace In-line – search při horizontech $NU = 10$ $N2 = 10$ 

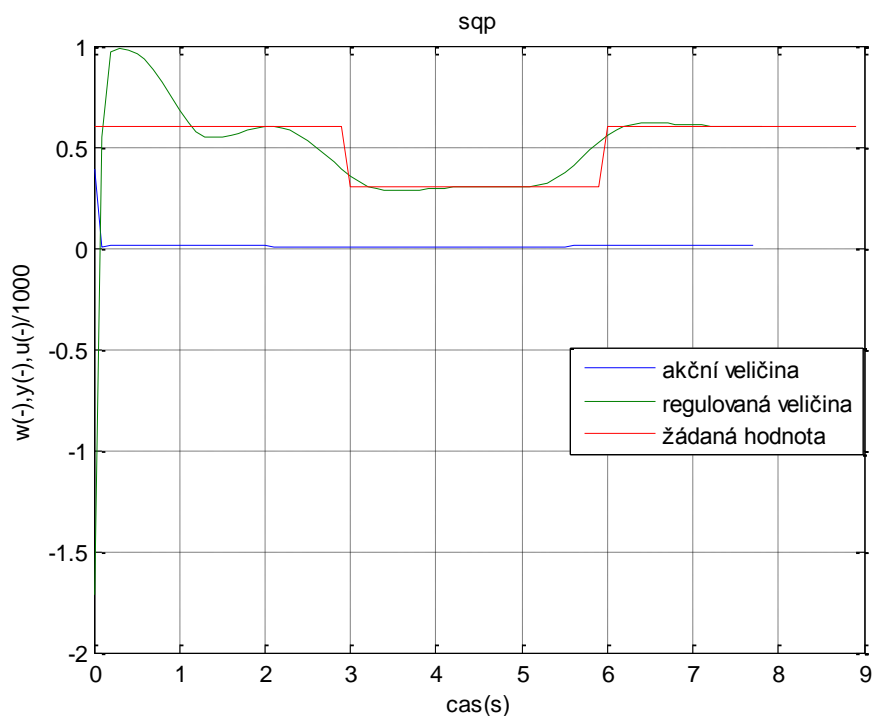
Obrázek 32 - regulace pomocí algoritmu In – line -search při horizontech $NU=10$
 $N2 = 10$

8.4.2 Simulace active - set při horizontech $NU = 10$ $N2 = 10$



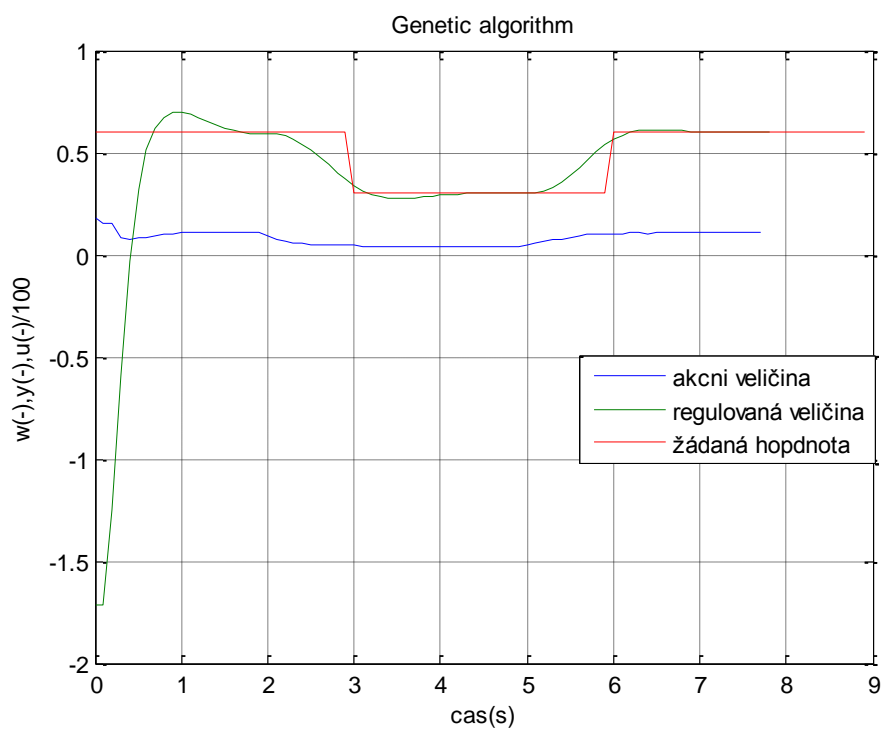
Obrázek 33 - regulace pomocí algoritmu Active - set při horizontech $NU=10$ $N2 = 10$

8.4.3 Simulace SQP při horizontech $NU = 10$ $N2 = 10$

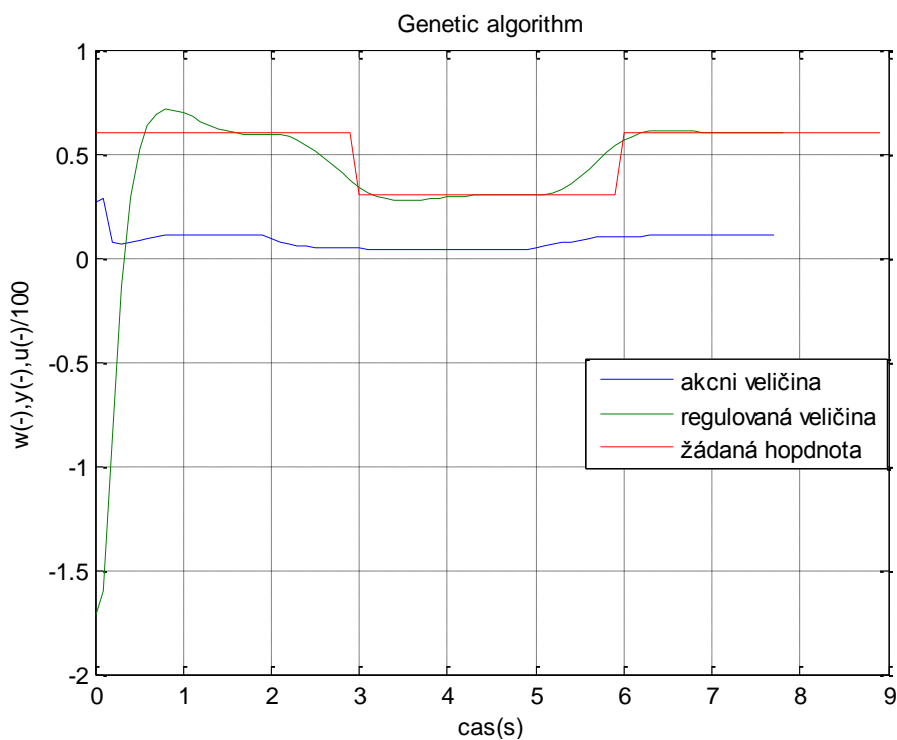


Obrázek 34 - regulace pomocí algoritmu SQP při horizontech $NU=10$ $N2 = 10$

8.4.4 Simulace genetic algorithm při horizontech $NU = 10$ $N2 = 10$



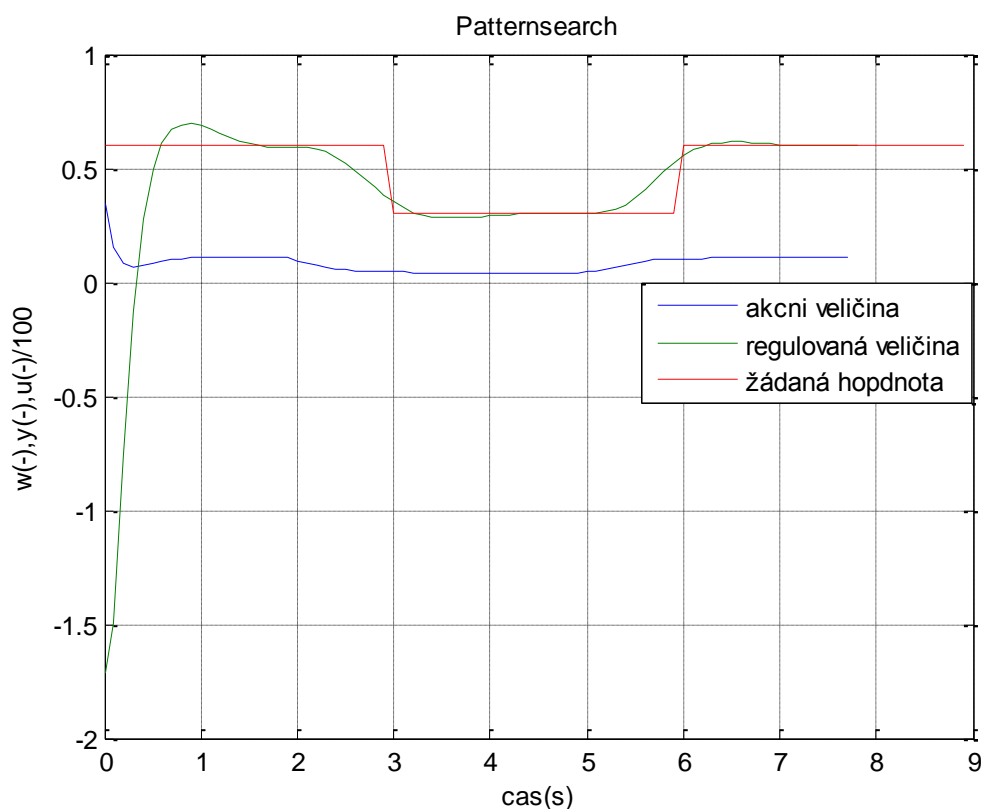
Obrázek 35 - regulace pomocí genetic algoritmu při horizontech $NU=10$ $N2 = 10$
a populaci 10



Obrázek 36 - regulace pomocí genetic algoritmu při horizontech $NU=10$ $N2 = 10$
a populaci 40

Zhodnocení:

Při použití genetického algoritmu je nutné si předem zvážit velikost populace. Jak lze vidět na obrázcích 36 a 37, tak na průběhu regulované veličiny velikost populace má nepatrný vliv. Na výpočetní čas už je vliv markantní vliv tabulka níže. Při zkoumání akčního zásahu je jediný větší rozdíl při různých volbách populace počáteční akční zásah. Kde při populaci 40 je zde pozvolnější průběh než při populaci 10.

8.4.5 Patternsearch algorithm při horizontech $NU = 10$ $N2 = 10$ 

Obrázek 37 – Simulace průběhu pattern search

Zhodnocení:

Jak zde vidíme tak tento způsob optimalizace dává dobrý regulační pochod. Na začátku je sice větší akční zásah, ale další průběh akční veličiny je pozvolný. Regulovaná veličina na začátku má malý překmit od žádané hodnoty. Při dalších skocích žádané veličiny je překmit už velice nepatrný. Tento algoritmus dává nejlepší výsledky za cenu nejdelšího času výpočtu optimalizovaných akčních zásahů. Viz porovnání tabulka 4 níže.

8.5 Porovnání jednotlivých algoritmů

Tabulka 4 – Tabulka porovnání jednotlivých optimalizačních algoritmů

Použitý algoritmus	čas simulace při NU = 10 N2 = 10	Výpočetní čas na 1 krok při NU = 10 N2 = 10
levenberg-marquardt	380,14 s	4,87 s
trust-region – reflective	308,94 s	3,96 s
Nelder-Nead simplex	9 201,33 s	117,96 s
In-line – search	1126,49 s	14,44 s
Active – set	1372,78 s	17,59 s
sqp	1217,18 s	15,6 s
Genetický algoritmus Populace (40)	7133,47 s	91,45 s
Genetický algoritmus Populace (10)	2144,69 s	27,49 s
Pattern search	43241,42 s	554,38 s

Zhodnocení:

V této tabulce jsem zadal do porovnání jednotlivé výpočetní časy algoritmů. Jeden sloupec je čas celé simulace. Druhý udává, jak dlouho mu trvalo vypočítat optimum akčního zásahu pro jeden krok. V této tabulce je zřejmé, že nejrychlejší je algoritmus trust-region – reflective a nejpomalejší je metoda Pattern search. Dále by bylo vhodné podle průměrných časů výpočtu na jeden krok volit vhodné soustavy s vhodnou periodou vzorkování. Periodu vzorkování bych volil nejméně 2x větší než je průměr. Z důvodu dostatečného času pro výpočet optima. Jednotlivé výpočty by se daly optimalizovat např. zvolení méně náročného kritéria. Výpočet by se dal urychlit použitím výkonnějšího počítače, který podporuje optimalizační výpočty. Při kratších horizontech a časech se algoritmy trust- region a le-

venberg se pro první skok regulovaná veličina nestačila ustálit. Řešením je minimálně prodloužit čas pro první skok. Dále se naskytuje řešení prodloužit horizonty. Jako vhodné algoritmy pro reálné měření bych volil levenberg-marquardt nebo trust-region – reflective z důvodu rychlosti výpočtu. Nedostatky typu velký překmit nebo neustálení pro první skok se dá jednoduše odstranit vhodnými horizonty nebo prodloužení času. Ostatní algoritmy by byli vhodné pro velice pomalé soustavy, jak nám napovídají průměrné časy výpočtu pro jeden krok. Kde jsem vzal celkový čas simulace a podělil počtem kroků. Tento typ výpočtu není přesný, ale pro orientační porovnání to stačí. Podle těchto časů se dá odhadnout, pro které soustavy se hodí. Tyto časy by se daly zmenšit, jak jsem popsal výše.

9 REÁLNÉ MĚŘENÍ

Podle výpočetních časů a grafů, které byly změřeny jsem zvolil jako nejrychlejší algoritmus trust region nebo levenberg-marquard. Tyto algoritmy založené na nelineárních čtvercích dávají dobré výsledky a dostatečně rychle vypočítají regulační krok. Jako model, na kterém ověřím tento způsob regulace, jsem vybral model 3. nádrží. Hlavně díky tomu, že je dostatečně pomalá. U tohoto modelu nevádí perioda vzorkování kolem 60s. Model obsahuje 2 čerpadla, 6 ventilů a již zmíněné 3 nádrže. Ventily propojují nádrže a umožňují vypouštění kapaliny z jednotlivé nádrže do zásobníku, který je pod modelem. Ten je umístěn na fakultě aplikované informatiky ve Zlíně v laboratoři řízení reálných procesů.

Tento model obsahuje tři nádrže, 6 ventilů a dvě čerpadla. Jedno čerpadlo napouští první nádrž a druhé poslední nádrž. Dva ventily propojují jednotlivé nádrže mezi sebou a zbytek umožňuje vypouštění jednotlivých nádrží. Po dobu měření byly otevřeny na plno ventily, které propojují jednotlivé nádrže. Pootevřený byl ventil pro vypouštění z poslední nádrže. Ostatní byly zavřeny. Regulátor byl realizován v programovém prostředí MATLAB/Simulink. U všech regulací je akčním zásahem řídicí napětí čerpadla, pro první nádrž. Výstupem je hladina ve 3. nádrži. Maximální hladiny v nádržích je 600mm. Po překročení této hodnoty se čerpadla automaticky vypnou. Akční zásah se pohybuje v matlabovských jednotkách od -1 až po 1. To znamená -1 čerpadlo vypnuto a 1 znamená čerpadlo jede na plno.



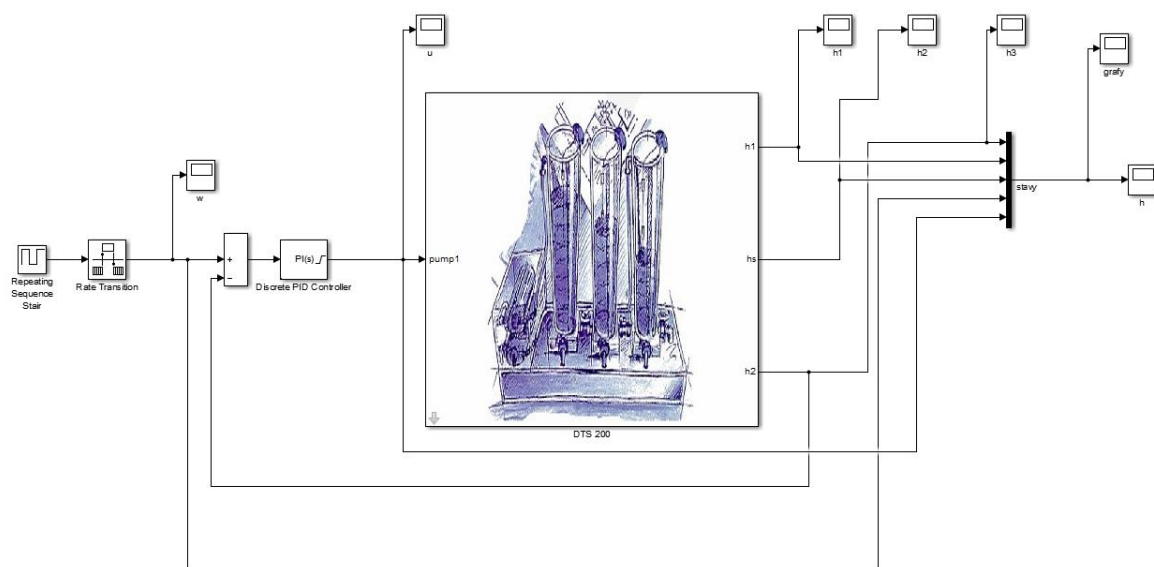
Obrázek 38 – pohled na model 3 nádrží

9.2 Regulace simulingového modelu

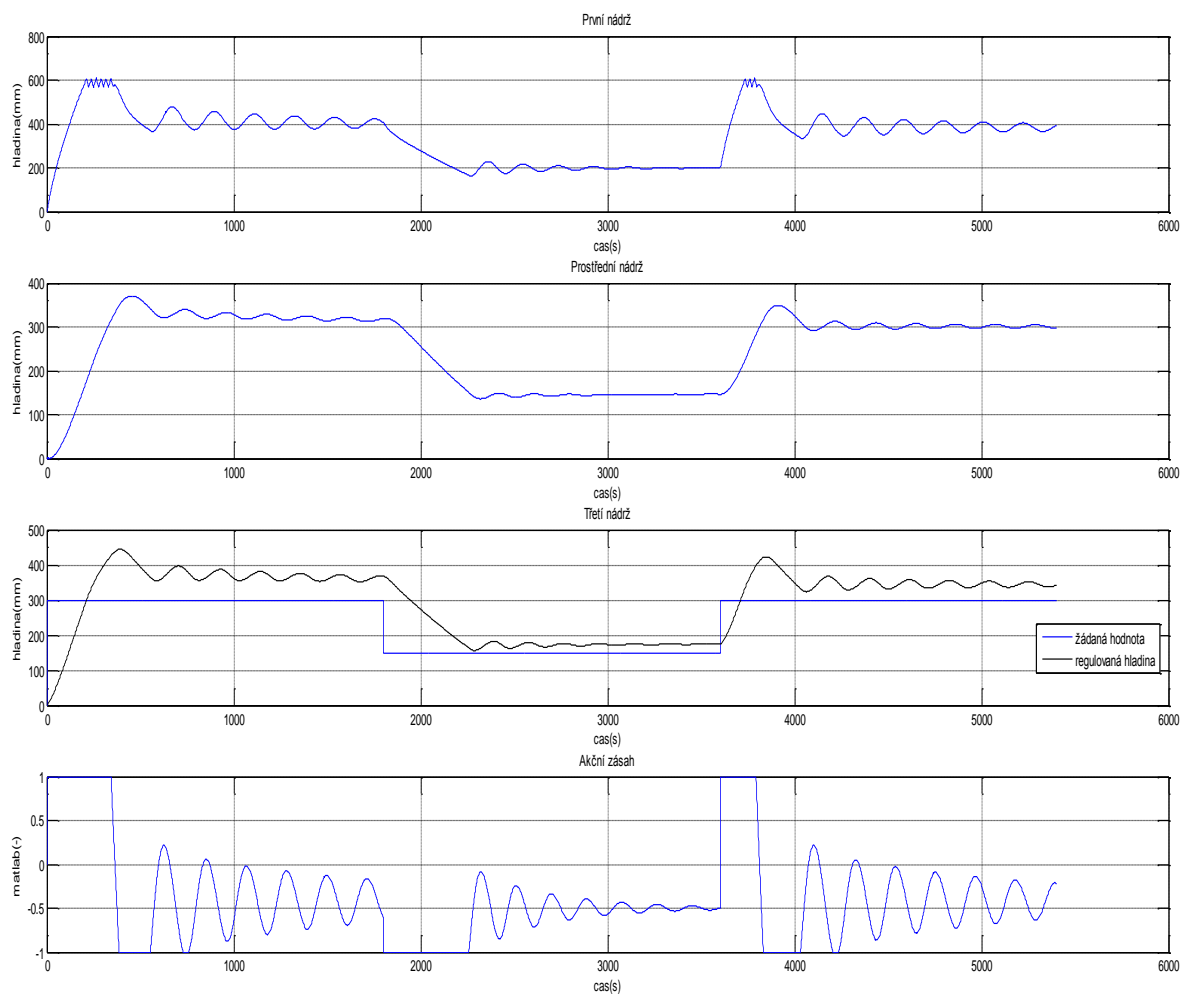
Pro provedení simulačního ověření prediktivního řízení modelu reálné soustavy DTS200 je nejprve nutné spustit m-file *Nadrze_simulace.m*, jehož kód je zobrazen v příloze P III. Tento skript si zavolá m-file *nadrze_optimizace_simulace.m*, jehož kód je zobrazen v příloze P IV. V tomto kódu je vlastní optimalizační algoritmus pomocí nelineárních čtverců. Zde si nastavíme parametry optimalizačního algoritmu a predikční horizont. Jako poslední je zavolán skript *crit_nadrze_simulace.m*, který obsahuje vlastní kvadratické kritérium. V kódu skriptu si volíme nastavení parametru λ . Programy simulují chování modelu soustavy. Simulinkové schéma simulace je zobrazeno na Obrázku 41.

9.2.1 Regulace pomocí nastavení regulátoru PID

Parametry regulátoru jsem nejprve nastavil podle *ziegler nichols* metody. Poté jsem provedl menší změny parametrů, které jsem od simuloval. Výsledné parametry jsou: $P = 0,06$ a $I = 0,000035$.

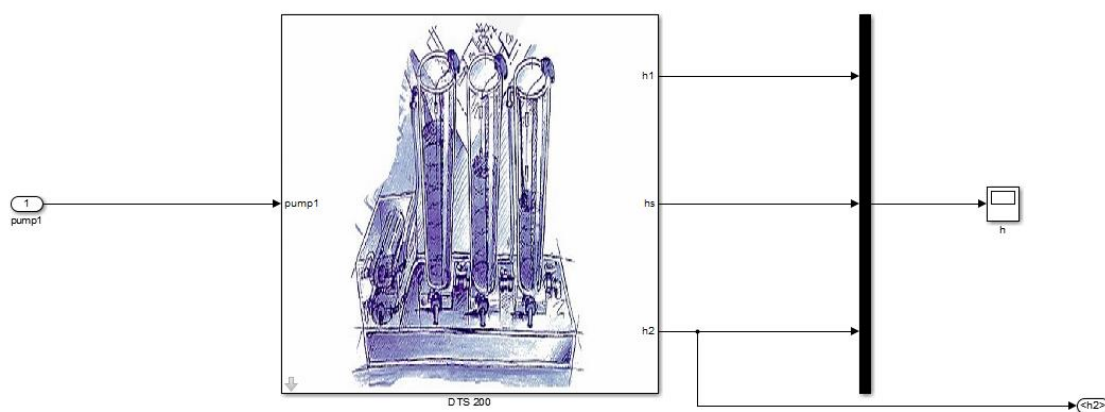


Obrázek 39 – Schéma zapojení pro regulaci pomocí PID regulátoru

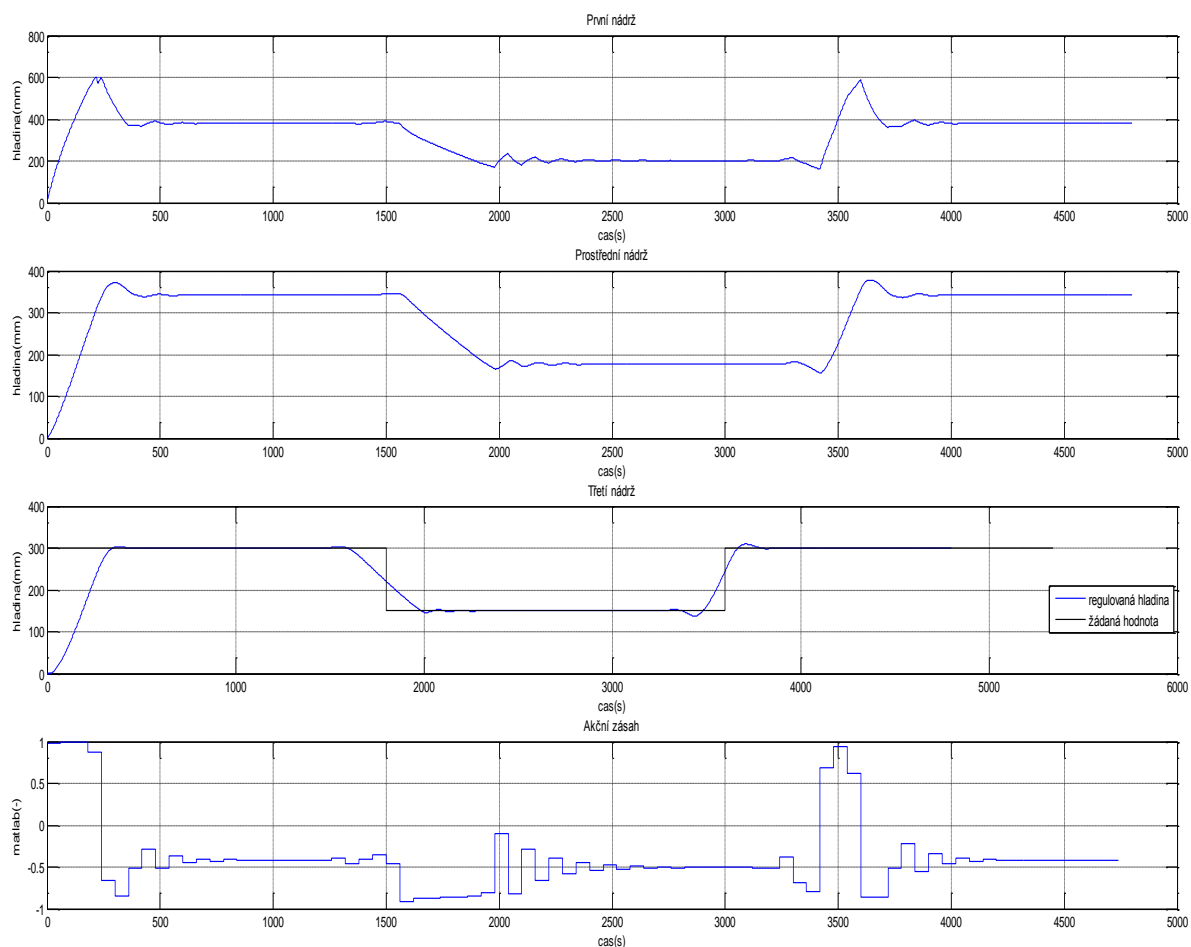


Obrázek 40 – simulace soustavy pomocí PID regulátoru

9.2.2 Regulace pomocí optimalizace



Obrázek 41 – schéma pro simulaci



Obrázek 42 – simulace soustavy tří nádrží pomocí optimalizace

V těchto grafech je zobrazeno průběh hladin v jednotlivých nádržích poslední graf ukazuje průběh akčního zásahu.

9.2.3 Porovnání regulací pomocí kritérií

Jednotlivé průběhy byly dále srovnány pomocí kritéria kvadrátu odchylky výstupní veličiny y od žádané veličiny w pomocí vztahu $Sy = \sum_{i=1}^N (w(i) - y(i))^2$, kde N je počet naměřených dat, a pomocí kritéria sumy přírůstků akční veličiny u podle vztahu $Su = \sum_{i=1}^N \Delta u^2(i) = \sum_{i=1}^N (u(i) - u(i-1))^2$, kde N je počet naměřených dat. V následující tabulce je porovnání mezi regulací s PID regulátorem a pomocí optimalizace.

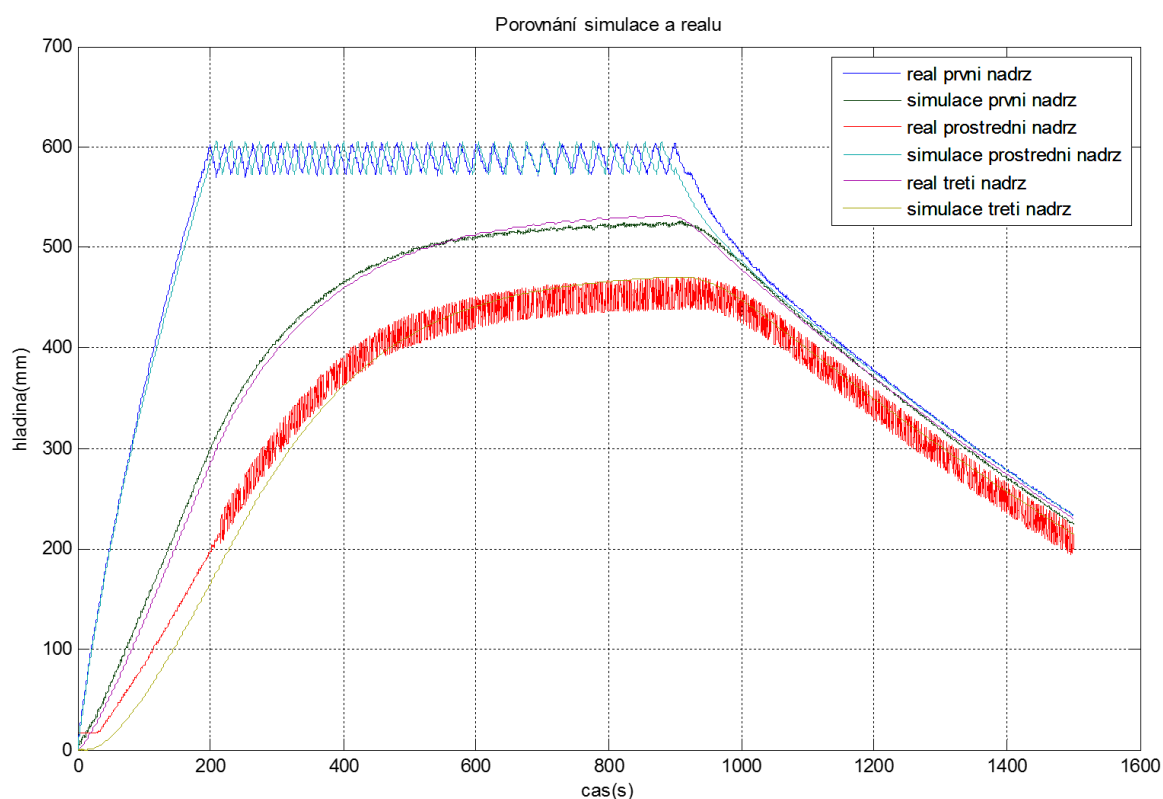
Tabulka 5 – Porovnání simulací pomocí kritéria

	Su	Sy
Optimalizace	9,5137	$5,3845 \cdot 10^3$
PID regulátor	3,8360	$5,7317 \cdot 10^3$

9.3 Reálné měření

Na reálném modelu jsem nejprve naměřil přechodovou charakteristiku. Poté se snažil co nejlépe přiblížit simulaci. Nastavení bylo: mezi nádržemi ventily plně otevřeny, ventil pro vypouštění z nádrže otevřen do $\frac{1}{4}$. Pro toto otevření bylo nutné najít konstantu ventilu. Jak je vidět na porovnání obrázek 43 shoda není dokonalá. Pode grafu ale můžeme vidět, že je dostatečná.

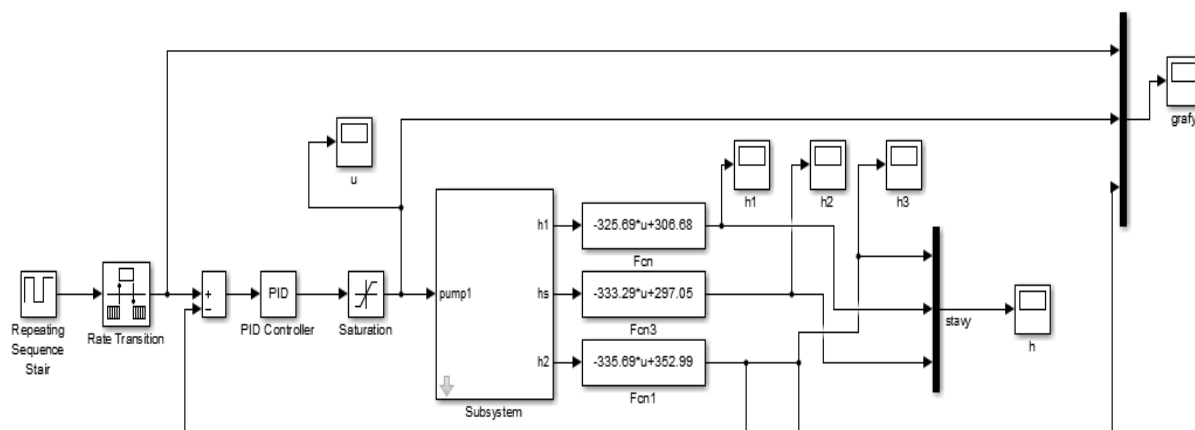
9.4 Porovnání modelu a reálu



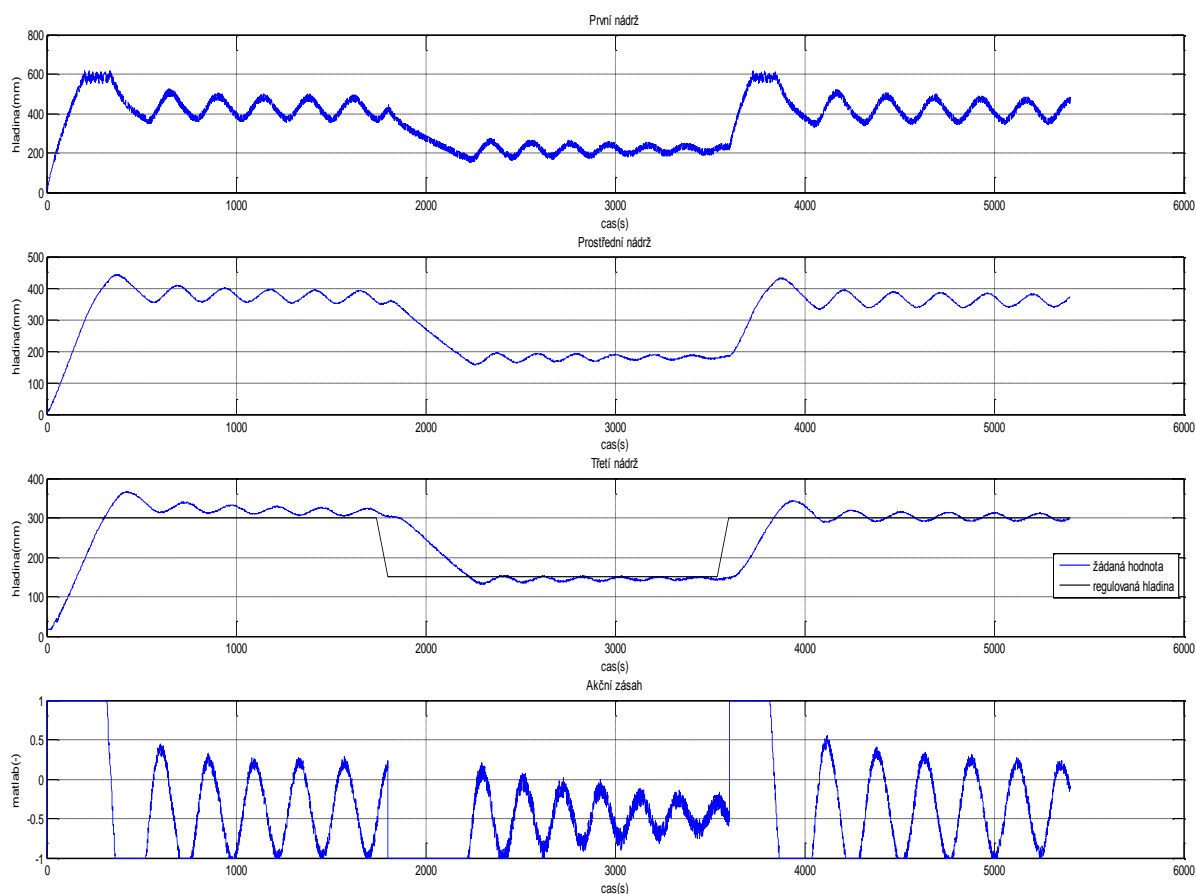
Obrázek 43 – Porovnání modelu a reálu

V tomto grafu vidíme srovnání přechodové charakteristiky reálného modelu a simulace. Reálná data ze senzoru měření hladiny ze začátku byly nezašuměna po uplynutí 200 s se šum zvětšil. To mohlo být způsobeno nejpravděpodobněji zahřátím elektronické součástky ve vyhodnocovací jednotce. Podle porovnání první a druhé nádrže, které se dobře shodují, mohou tvrdit, že model dobře souhlasí s reálným měření.

9.4.1 Regulace regulátoru PID



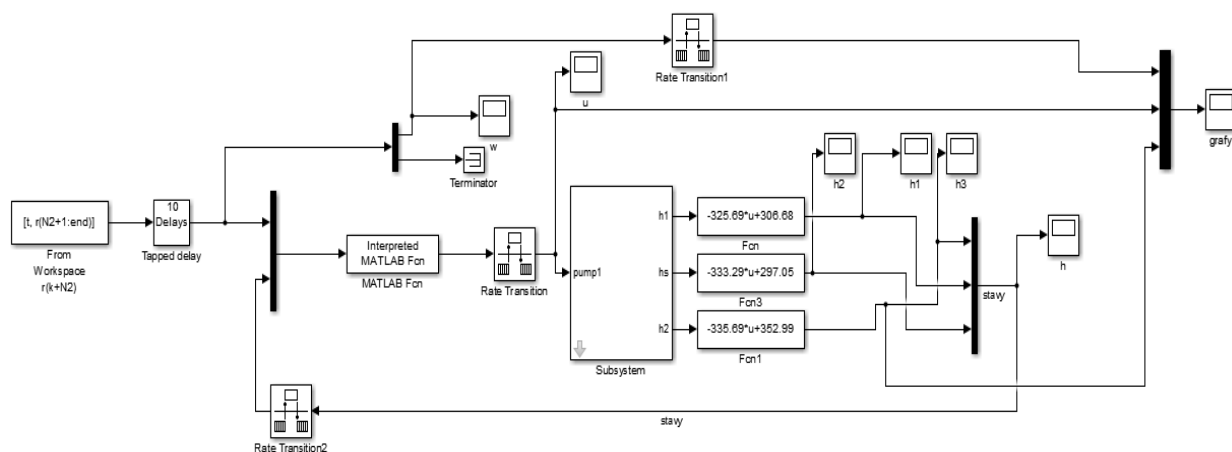
Obrázek 44 – schéma pro reálnou regulaci pomocí PID



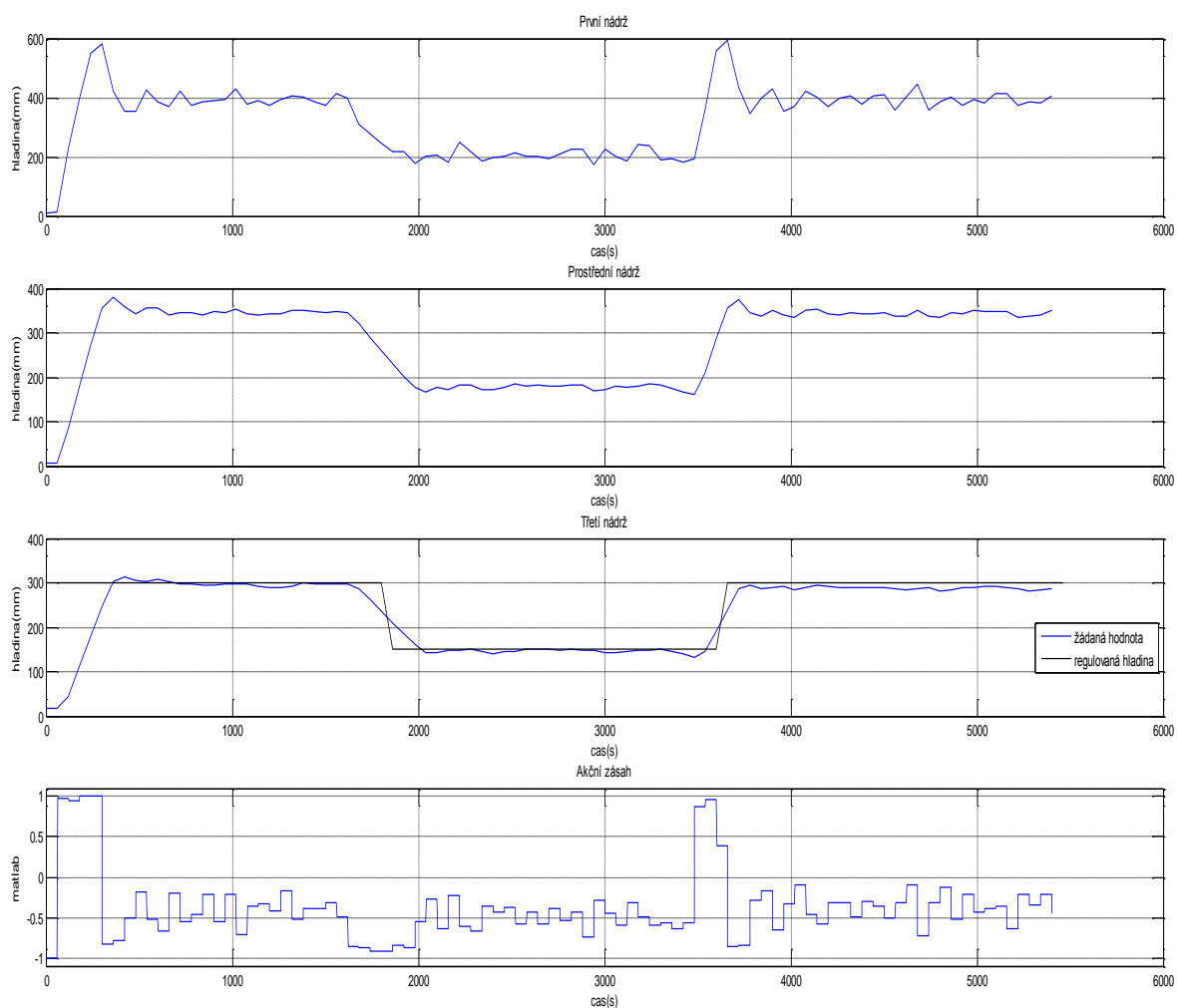
Obrázek 45 – grafy regulace na reálném modelu pro

Zde na tomto grafu je vidět regulace reálného modelu pomocí PID regulátoru. Data ze senzoru snímání hladiny první nádrže jsou trochu zašuměna. Toto je nejspíše způsobeno buď tím, že se do této nádrže napouštěla tekutina. U této regulace byla nastavena perioda vzorkování 1s.

9.4.2 Regulace pomocí optimalizace



Obrázek 46 – schéma pro reálné měření



Obrázek 47 – Graf regulace na reálném modelu

Natavení u této regulace byla perioda vzorkování 60s a lambda 1.

9.4.3 Porovnání reálné regulace pomocí kritérií

Stejně jako u simulací byly provedeno porovnání podle stejných kritérií, které jsem pro porovnání použil u simulací.

Tabulka 6 – porovnání pomocí kritérií reálné měření

	Su	Sy
Optimalizace	16,6858	$3,4708 \cdot 10^3$
PID	34,5947	$8,2204 \cdot 10^6$

ZÁVĚR

Náplní diplomové práce je porovnání optimalizačních algoritmů a následné měření pomocí vhodného algoritmu na reálné soustavě DTS200.

Teoretická část obsahuje teoretický základ a princip jednotlivých optimalizačních algoritmů. Zde je popsáno, jak jednotlivé algoritmy hledají minimum kritériální funkce.

V praktická části jsem se nejprve věnoval výpočetnímu času jednotlivých algoritmů. Jako nejrychlejší mi vyšly nelineární čtverce, kde bylo možné zvolit trust-region – reflective nebo levenberg-marquardt. Nejrychlejší vyšel s časem simulace 308,94s a průměrným časem na jeden krok 3,96s. Jako nejpomalejší vyšel Pattern search s časem simulace 43241,42 s a průměrný čas výpočtu pro jeden krok s časem 554,38s. Model, který byl použit pro toto měření je nl-offsets. Ten se nachází v matlabu a je ho možné použít jako pokusný pro více simulací. Výpočetní časy byly měřeny na Lenovo sl500 Thinkpad.

Pro reálné měření jsem vybral nejrychlejší algoritmus. U reálné měření byl potřeba ověřit čas regulace a vhodně zvolit periodu vzorkování. Z důvodu jiného a složitějšího modelu. V reálném měření na soustavě DTS200 byla ověřena schopnost regulátoru uregulovat tuto soustavu, přičemž bylo provedeno srovnání se spojitým PID regulátorem. Srovnání jednotlivých regulací je uvedeno v tabulkách 5 a 6.

Výhoda nelineárního regulátoru spočívá v tom, že už on sám počítá s nelinearitami soustavy. Těmito nelinearitami mohou být mezení soustavy, kdy se zapíná a vypíná čerpadlo. Nevýhodou je, že výpočet akčního zásahu trvá delší čas. Výpočetní čas je možné zkrátit díky dobrému nastavení algoritmu nebo použití rychlejšího počítače. Nelineární prediktivní regulace má smysl, pokud se použije na pomalé soustavy.

Podle kritérií vyšla simulace pomocí optimalizace v kritérií $S_u = 9,5137$ a $S_y = 5,3845 \cdot 10^3$. Porovnání simulace PID podle kritérií vyšla: $S_u = 3,8360$ a $S_y = 5,7317 \cdot 10^3$. Reálné měření vyšlo: $S_u = 16,6858$ a $S_y = 3,4708 \cdot 10^3$. Reálné měření pomocí PID vyšlo: $S_u = 34,5947$ a $S_y = 8,2204 \cdot 10^6$. Rozdíl mezi simulací a reálném měření je způsoben rozdílem mezi modelem a reálnou soustavou. V této diplomové práci není vytvořen žádný kompenzátor, který by kompenzoval rozdíl mezi modelem a reálnou soustavou. Toto se taky dá počítat jako další důvod proč je rozdíl mezi simulací a reálnou soustavou. Celkově vyšla regulace pomocí optimalizace jako lepší než pomocí PID.

SEZNAM POUŽITÉ LITERATURY

- [1] Navrátil, P. *Informační systém CAAC – Počítačová podpora automatického řízení*, Disertační práce, UTB – FT ve Zlíně, 2004.
- [2] Nguyen, T.C.P. *Počítačová podpora automatického řízení CAAC*, Disertační práce, VUT – FSI v Brně, 2001.
- [3] Giesl, P.: Prediktivní řízení laboratorních modelů. Diplomová práce, UTB ve Zlíně, FAI, 2006.
- [4] Cutler, C. R. and B. L. Ramaker, Dynamic Matrix Control. In: *Proc. Joint Automatic Control Conference*, volume 1, San Francisco, CA. Paper No. WP5-B, 1980.
- [5] Mikleš, J. and M. Fikar, *Process Modelling, Optimisation and Control*. Springer-Verlag, Berlin, 2008.
- [6] D. W. Clarke, D. W., C. Mohtadi, and P. S. Tuffs, Generalized predictive control. Part I. The basic algorithm, Generalized predictive control. Part II. Extensions and interpretations. *Automatica*, 23, 137-160, 1987.
- [7] Fikar, M. and S. Engell, Receding horizon predictive control based upon Youla-Kucera parametrisation. *European Journal of Control*, 3, 304–316, 1997.
- [8] Kwon, W. H. and S. Han, *Receding Horizon Control*. Springer-Verlag, London, 2005.
- [9] Bitmead, R. R., M. Gevers and V. Wertz, *Adaptive Optimal Control: The Thinking Man's GPC*. Prentice Hall, 1990.
- [10] Soeterboek, R. *Predictive Control-a Unified Approach*. Prentice-Hall, Englewood Cliffs, New York, 1992.
- [11] Kwon, W. H., H. H. Choi, D. G. Byun and S. B. Noh, Recursive solution of generalized predictive control and its equivalence to receding horizon tracking control. *Automatica*, 28, 1235-1238, 1992.
- [12] Camacho, E. F. and C. Bordons, *Model Predictive Control*. Springer-Verlag, London, 2004.
- [13] P.P. Kanjilal, *Adaptive Prediction and Predictive Control*. Peter Peregrinus, U. K., 1995.
- [14] Sunan, H., T. K. Kiong and L. T. Heng, *Applied Predictive Control*. Springer-Verlag, London, 2002.
- [12] Maciejowski, J. M. *Predictive Control with Constraints*. Prentice Hall, Harlow,

2002.

- [15] Rossiter, J. A., *Model Based Predictive Control: A Practical Approach*. CRC Press, Boca Raton, Florida, 2003.
- [16] Bemporad, A., M. Morari, V. Dua, V. and E. N. Pistikopoulos, The explicit linear quadratic regulator for constrained systems, *Automatica*, 38, 2002, 3-20.
- [17] Grieder, P., M. Kvasnica, M. Baotić and M. Morari, Stabilizing low complexity feedback control of constrained piecewise affine systems. *Automatica*, 41, 1683–1694, 2005.
- [18] Kvasnica, M.: Efficient Software Tools for Control and Analysis of Hybrid Systems. ETH Zurich, Physikstrasse 3, 8092 Zurich, Switzerland, 2008.
- [19] Kvasnica, M., P. Grieder and M. Baotić, Multi-Parametric Toolbox (MPT), <http://control.ee.ethz.ch/~mpt/>, 2004.
- [20] Henson, M. A., Nonlinear model predictive control: current status and future directions. *Computers and Chemical Engineering*, 23, 1998, 187–202.
- [21] Aggelogiannaki E. and H. Sarimveis, Nonlinear model predictive control for distributed parameter systems using data driven artificial neural network models. *Computers & Chemical Engineering*, 32, 2008, 1225-1237.
- [22] Martínez, M., J. S. Senent and X. Blasco, Generalized predictive control using genetic algorithms (GAGPC). *Engineering Applications of Artificial Intelligence*,
- [23] Onnen C., R. Babuška, U. Kaymak, J. M. Sousa, H. B. Verbruggen and R., Genetic algorithms for optimization in predictive control. *Control Engineering Practice*, 5, 1997, 1363-1372.
- [24] Zelinka, I., Z. Oplatková, M. Šeda, P. Ošmera and F. V_ela_, *Evolution of the Computer Technique – Principle and Application* (in Czech), BEN, Prague, 2008.
- [25] Murray-Smith, R. and T. A. Johansen, *Multiple Model Approaches to Modelling and Control*. Taylor and Francis, London, 1997.
- [26] Z.K. Xue, Z. K. and S.Y. Li, Multi-model modelling and predictive control based on local model networks. *Control and Intelligent Systems archive*, 34 , 2006, 105 - 112.
- [27] M. Morari, M. and J. H. Lee, Model predictive control: past, present and future. *Computers and Chemical Engineering*, 23, 1999, 667-682.
- [28] Quin, S. J and T. A. Bandgwell, An overview of industrial model predictive control technology. In: *Proceedings of the Chemical Process Control – V.*, 93,

AIChE Symposium Series. CACHE and AIChE. Tahoe City, CA, USA, 1996, 232-256.

[29] Quin, S. J. and T. A. Bandgwell, An overview of nonlinear model predictive control applications. *Nonlinear Model Predictive Control* (F. Allgöwer & A. Zheng, Ed.), Birkhäuser Verlag, Basel – Boston – Berlin, 2000, 369-392.

[30] S. J. Quin, S. J. and T. A. Bandgwell, A survey of industrial model predictive control technology. *Control Engineering Practice*, 11, 2003, 733-764.

[31] Chalupa, P. and V. Bobál, Modelling and predictive control of inverted pendulum. In: *Proc. of the 22th European Conference on Modelling and Simulation*, Nicosia, Cyprus, 2008, 531-537.

[32] Bobál, V., M. Kubalčík, P. Chalupa and P. Dostál, Adaptive predictive control of nonlinear system with constraint of manipulated variable. In: *Proc. of the 28th IASTED International Conference Modelling Identification and Control (MIC 2009)*, Innsbruck, Austria, 2009, 349-354.

[33] Bobál, V., M. Kubalčík, P. Chalupa and P. Dostál, Self-tuning control of nonlinear servo system: comparison of LQ and predictive approach. In: *Proc. of the 17th Mediterranean Conference on Control and Automation MED'09*, Thessaloniki, Greece, 2009, 240-245.

[34] Bobál, V., Chalupa, P., Kubalčík, M. and P. Dostál, Self-tuning Predictive Control of Nonlinear Servo-motor. *Journal of Electrical Engineering*, 61, 2010, 365- 372.

[35] Kubalčík, M. and V. Bobál, Adaptive predictive control applied to coupled drives process. In: *Proc. of the 28th IASTED International Conference Modelling Identification and Control (MIC 2009)*, Innsbruck, Austria, 2009, 331-336.

[36] Bobál, V., M. Kubalčík, P. Chalupa and P. Dostál, Self-tuning predictive control of through-flow heat exchanger. In: *Proc. of the 36th International Conference of Slovak Society of Chemical Engineering*, Tatranské Matliare, Slovakia, 2009, 089-1 - 089-10.

[37] Zhaoa, F., Y. P. Gupta, A simplified predictive control algorithm for disturbance rejection. In: *The Instrumentation, Systems, and Automation Society*, Volume 44, Issue 2, April 2005, P. 187-198

[38] Jiguang, Z., Shi Ren, Linan, M., Mengxiao. W, Disturbance Rejection Performance of Generalized Predictive Control. In: *Intelligent Control and Automation*, 2002, Proceedings of the 4th World Congress on, P. 295 – 29

- [38] Rossiter, J.A. Chisci, L. Disturbance rejection in constrained predictive control. In: *Control '98. UKACC International Conference on* (Conf. Publ. No. 455), Sep 1998
- [39] Jun Yang, Shihua Li, Xisong Chen and Qi Li, Disturbance rejection of dead-time processes using disturbance observer and model predictive control. In: *Chemical Engineering Research and Design*, Volume 89, Issue 2, February 2011, P. 125-135
- [40] Manzie, C., H. C. Watson, A novel approach to disturbance rejection in idle speed control towards reduced idle fuel consumption. In: *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, 2003
- [41] Bobál, V. Adaptivní a prediktivní řízení. Zlín: Univerzita Tomáše Bati ve Zlíně, Akademické centrum 2009, ISBN 978-80-7318-662-3
- [42] Algoritmus. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2014-04-18]. Dostupné z: <http://cs.wikipedia.org/wiki/Algoritmus>
- [43] The Levenberg-Marquardt method for nonlinear least squares curve-fitting problems. In: *People.duke.edu* [online]. Duke University, October 9, 2013 [cit. 2014-04-18]. Dostupné z: <http://people.duke.edu/~hpgavin/ce281/lm.pdf>
- [44] An algorithm for Least- Squares Estimation Nonlinear Parametrs. *Journals, primary sources, and now BOOKS* [online]. 2004 [cit. 2014-05-03]. Dostupné z: http://www.dista.unibo.it/~bittelli/materiale_lettura_fisica_terreno/marquardt_63.pdf
- [45] Branch, M.A., T.F. Coleman, and Y. Li, "A Subspace, Interior, and Conjugate Gradient Method for Large-Scale Bound-Constrained Minimization Problems," *SIAM Journal on Scientific Computing*, Vol. 21, Number 1, pp 1-23, 1999.
- [46] Byrd, R.H., R.B. Schnabel, and G.A. Shultz, "Approximate Solution of the Trust Region Problem by Minimization over Two-Dimensional Subspaces," *Mathematical Programming*, Vol. 40, pp 247-263, 1988.
- [47] Coleman, T.F. and A. Verma, "A Preconditioned Conjugate Gradient Approach to Linear Equality Constrained Minimization," submitted to *Computational Optimization and Applications*.
- [48] Moré, J.J. and D.C. Sorensen, "Computing a Trust Region Step," *SIAM Journal on Scientific and Statistical Computing*, Vol. 3, pp 553-572, 1983

- [49] Sorensen, D.C., "Minimization of a Large Scale Quadratic Function Subject to an Ellipsoidal Constraint," Department of Computational and Applied Mathematics, Rice University, Technical Report TR94-27, 1994.
- [50] Steihaug, T., "The Conjugate Gradient Method and Trust Regions in Large Scale Optimization," *SIAM Journal on Numerical Analysis*, Vol. 20, pp 626-637, 1983.
- [51] Optimalizace (informatika). In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2014-05-05]. Dostupné z: http://cs.wikipedia.org/wiki/Optimalizace_%28informatika%29
- [52] Lagarias, J. C., J. A. Reeds, M. H. Wright, and P. E. Wright. "Convergence Properties of the Nelder-Mead Simplex Method in Low Dimensions." *SIAM Journal of Optimization*, Vol. 9, Number 1, 1998, pp. 112–147.
- [53] TISNOVSKÝ, Pavel. Linerní systém. *Vysoké Učení Technické v Brně* [online]. 1999 [cit.2014-05-05].Dostupné z: <http://www.fit.vutbr.cz/~tisnovpa/publikace/diplomka/doc/node30.html>
- [54] TISNOVSKÝ, Pavel. Nelineární systém. *Vysoké Učení Technické v Brně* [online]. 1999 [cit. 2014-05-05]. Dostupné z: <http://www.fit.vutbr.cz/~tisnovpa/publikace/diplomka/doc/node31.html>
- [55] JANEČEK, Michal. *Prediktivní řízení systémů*. Zlín, 2006. Dostupné z: https://dspace.k.utb.cz/bitstream/handle/10563/2098/jane%20ek_2006_dp.pdf?sequence=1. Diplomová práce. UTB. Vedoucí práce Ing. Hana Vyoralová.
- [56] MAREŠ, Jan a Pavel HRNČIŘÍK. Vydavatelství. *Vysoká škola chemicko technologická* [online]. 2012 [cit. 2014-05-05]. Dostupné z: http://vydavatelstvi.vscht.cz/katalog/uid_isbn-978-80-7080-823-8/anotace/
- [57] MATYS, Libor. *Prediktivní regulátory s principy umělé inteligence v prostředí MATLAB* – B&R. Dostupné z: https://dspace.vutbr.cz/bitstream/handle/11012/13039/xmatys05_dp.pdf?sequence=1. Diplomová práce. Vysoké učení technické v Brně Fakulta elektrotechniky a komunikačních technologií Ústav automatizace a měřicí techniky. Vedoucí práce prof. Ing. Petr Pivoňka,
- [58] Lagarias, J. C., J. A. Reeds, M. H. Wright, and P. E. Wright. "Convergence Properties of the Nelder-Mead Simplex Method in Low Dimensions." *SIAM Journal of Optimization*, Vol. 9, Number 1, 1998, pp. 112–147.

[59] Sequential quadratic programming. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2014-05-10]. Dostupné z: http://en.wikipedia.org/wiki/Sequential_quadratic_programming

[60] Active set method. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2014-05-10]. Dostupné z: http://en.wikipedia.org/wiki/Active_set_method

[61] Line search. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2014-05-10]. Dostupné z: http://en.wikipedia.org/wiki/Line_search

[62] Genetic Algorithm. *MathWorks* [online]. 1994-2014 [cit. 2014-05-10]. Dostupné z: <http://www.mathworks.com/discovery/genetic-algorithm.html>

[63] Pattern search (optimization). In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2014-05-10]. Dostupné z: http://en.wikipedia.org/wiki/Pattern_search_%28optimization%29

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

$F(x)$	Funkční hodnota v x-ové souřadnici.
$F(y)$	Funkční hodnota v y-ové souřadnici.
LQ	Linear quadratic
LQG	Linear quadratic gaussian
MPC	Model predictive control
DMC	Dynamic Matrix Control
GPC	Generalized Predictive Control
RHC	Receding Horizon control
MM	Multiple Model
RHTC	Receding Horizon tracking control
$W(k)$	Žádaná hodnota v kroku k .
$Y(k)$	Regulovaná veličina v kroku k .
$U(k)$	Akční zásah v kroku k .
N	Počet dat
N1	Minimální predikční horizont.
N2	Maximální predikční horizont.
Nu	Řídící horizont

SEZNAM OBRÁZKŮ

Obrázek 1 – Základní struktura prediktivního řízení[41]	18
Obrázek 2 - Princip prediktivního řízení (N_1 , N_2 , N_u – minimální, maximální, řídicí horizont) [41].....	19
Obrázek 3 – Naznačení Nelder-Mead algoritmu	26
Obrázek 4 – Porovnání diskretizace systému se spojitým systémem	33
Obrázek 5 – žádaná hodnota, která je pro všechny algoritmy stejná.....	34
Obrázek 6 – Výsledek simulace algoritmu LQ pro měření výpočetního času	35
Obrázek 7 – $N_2=40$ $N_u=5$	36
Obrázek 8 – $N_2=40$ $N_u=30$	36
Obrázek 9 – Graf závislosti jak se mění výpočetní čas při konstantním horizontu N_2 a změnách horizontu N_2	37
Obrázek 10 – GPC při horizontech $N_u = 5$ $N_2=10$	38
Obrázek 11 - GPC při horizontech $N_u = 5$ $N_2=10$ $N_u = 5$ $N_2=40$	38
Obrázek 12 – Graf závislosti jak se mění výpočetní čas při konstantním horizontu N_u a změnách horizontu N_2	39
Obrázek 13 – nelineární systém.....	40
Obrázek 14 - schéma řízení nelineárního řízení	41
Obrázek 15 – žádaná hodnota	41
Obrázek 16 – graf porovnání regulovaných veličin.....	42
Obrázek 17– graf porovnání akčních veličin	42
Obrázek 18-porovnání regulovaných veličin při stejných horizontech	43
Obrázek 19 – Porovnání akčních veličin při stejných horizontech	44
Obrázek 20 – graf porovnání regulovaných veličin.....	44
Obrázek 21 – graf porovnání akčních veličin	45
Obrázek 22 - regulace pomocí algoritmu levenberg-marquardt	46
Obrázek 23 - regulace pomocí algoritmu levenberg-marquardt	47
Obrázek 24 - regulace pomocí algoritmu levenberg-marquardt	47
Obrázek 25 – Akční zásah při levenberg – marquardt algoritmu	48
Obrázek 26 – regulace pomocí algoritmu trust-region-reflective.....	48
Obrázek 27 - regulace pomocí algoritmu trust-region-reflective při horizontech $N_u=10$ $N_2 = 40$	49

Obrázek 28 - regulace pomocí algoritmu trust-region-reflective při horizontech NU=10 N2 = 10	49
Obrázek 29 – průběh akčního zásahu pomocí algoritmu trust-region-reflective při horizontech NU=10 N2 = 10	50
Obrázek 30 – Výřez akčního zásahu.....	50
Obrázek 31 – simulace pomocí fminsearch	51
Obrázek 32 - regulace pomocí algoritmu In – line -search při horizontech NU=10 N2 = 10.....	52
Obrázek 33 - regulace pomocí algoritmu Active - set při horizontech NU=10 N2 = 10	53
Obrázek 34 - regulace pomocí algoritmu SQP při horizontech NU=10 N2 = 10.....	53
Obrázek 35 - regulace pomocí genetic algoritmu při horizontech NU=10 N2 = 10 a populaci 10	54
Obrázek 36 - regulace pomocí genetic algoritmu při horizontech NU=10 N2 = 10 a populaci 40	54
Obrázek 37 – Simulace průběhu pattern search.....	55
Obrázek 38 – pohled na model 3 nádrží	59
Obrázek 39 – Schéma zapojení pro regulaci pomocí PID regulátoru.....	60
Obrázek 40 – simulace soustavy pomocí PID regulátoru	61
Obrázek 41 – schéma pro simulaci	61
Obrázek 42 – simulace soustavy tří nádrží pomocí optimalizace.....	62
Obrázek 43 – Porovnání modelu a reálu	63
Obrázek 44 – schéma pro reálnou regulaci pomocí PID	64
Obrázek 45 – grafy regulace na reálném modelu pro	64
Obrázek 46 – schéma pro reálné měření.....	65
Obrázek 48 – Graf regulace na reálném modelu	65

SEZNAM TABULEK

Tabulka 1 – Výpočetní čas.....	34
Tabulka 2 – Výpočetní čas při konstantním $N_2 = 40$ a změně horizontu N_u	35
Tabulka 3- Výpočetní čas při konstantním $N_u = 5$ a změně horizontu N_2	37
Tabulka 4 – Tabulka porovnání jednotlivých optimalizačních algoritmů.....	56
Tabulka 5 – Porovnání simulací pomocí kritéria.....	62
Tabulka 6 – porovnání pomocí kritérií reálné měření	66

SEZNAM PŘÍLOH

P1 – LQ algoritmus

P2 – GPC algoritmus

P3 - Script nadrze_simulace

P4 - Script nadrze_optimalizace_simluace

P5 - Script crit_nadrze_simulace

CD – obsahuje veškeré naměřené a simulovaná data s vytvořenými skripty a simulingo-
výmy modely. Dále obsahuje text diplomové práce.

PŘÍLOHA P I: ALGORITMUS LQ PRO MĚŘENÍ VÝPOČETNÍHO ČASU

```
clear all;
clc;
T0=1;
w=0.75;%Žádaná hodnota
[cit,jm]=c2dm([2],[50 15 1],T0);
b1=cit(2);
%načtení hodnot z přenosu
b2=cit(3);
a1=jm(2);
a2=jm(3);
qu=1;
%výpočet parametrů polynomů M
mr0=qu*(1+a1^2+a2^2)+b1^2+b2^2;
mr1=qu*(a1+a1*a2)+b1*b2;
mr2=qu*a2;
lamda=(mr0/2)-mr2+sqrt(((mr0/2)+mr2)^2-mr1^2);
%výpočet parametrů polynomů D
delta=((lamda+sqrt(lamda^2-4*(mr2^2)))/2);
d1=(mr1)/(delta+mr2);
d2=(mr2)/(delta);
r0=(1+d1+d2)/(b1+b2);
D=zeros(10,1);
%výpočet parametrů regulátoru
x1=d1-a1+1 ;
x2=d2+a1-a2 ;
x3=a2 ;
x4=0;
matsyst=[b1 0 0 1; b2 b1 0 a1-1; 0 b2 b1 a2-a1; 0 0 b2 -a2];
pravstr=[x1; x2; x3; x4];
parametry=inv(matsyst)*pravstr;
q0=parametry(1);
q1=parametry(2);
q2=parametry(3);
p1=parametry(4);
tic % začátek měření času
for i=1:1000
    w=0.75;%Žádaná hodnota
    D=zeros(10,1);
```

```

for k=1:100
    if k==40
        w=0.5;
    end
    if k==60
        w=0.75;
    end
    krok(k)=k;
    % výpočet všech výstupních veličin a akčních zásahů
    y1(k,i)=-a1*D(1)-a2*D(2)+b1*D(4)+b2*D(5);
    u1(k,i)=r0*w-q0*y1(k,i)-q1*D(1)-q2*D(2)+(1-p1)*D(4)+p1*D(5);
    % Cyklicka zamenave vektoru dat
    D(3)=D(2);
    D(2)=D(1);
    D(1)=y1(k);
    D(5)=D(4);
    D(4)=u1(k);
    ww(k)=w;
end
end;
toc% konec měření času
y=y1(:,1);
u=u1(:,1);
figure;
plot(krok,y,'--b',krok,ww,'k',krok,u,'b')
legend('výstupní veličnina y(k)', 'zadana hodnota', 'akční zásah u(k)')
xlabel('krok(-)')
ylabel('w(-),u(-),y(-)')
title('LQ regulace')

```


PŘÍLOHA P II: ALGORITMUS GPC PRO MĚŘENÍ VÝPOČETNÍHO ČASU

```
clc;
clear all;
% spojitý přenos
G = tf([2],[50 15 1]);
% diskrétní přenos
Ts = 1;
Gd = c2d(G, Ts);
% vytvoření gpc
GPCoptions = gpc2mpc;
% Hu
GPCoptions.NU = 5; %controlovaný horizont
% Hp
GPCoptions.N2 = 40; %konec predikce horizontu
% GPCoptions.N1 = 1; %začátek predikce horizontu
% R
GPCoptions.Lam = 0.1;
% GPCoptions.T = [3]; %čitatel rušení
% převod GPC na mpc regulátor
mpc = gpc2mpc(Gd, GPCoptions);
Tf=100; % počet simulačních kroků
% nastavení simulace
params=mpcsimopt(mpc);
params.MDLookAhead='on';
params.RefLookAhead='on';
w=[0.75*ones(40,1); 0.5*ones(20,1);0.75*ones(40,1)];
tic;
%simulace
for i=1:1000 % 1000 krát výpočet
[y1(:,i),t1(:,i),u1(:,i)]=sim(mpc,Tf,w,params);
end
toc;
% graf
y=y1(:,1);
u=u1(:,1);
t=t1(:,1);
figure()
plot(t,[y,w,u])
legend('výstupní veličnina y(k)', 'zadana hodnota', 'akční zásah u(k)')
```

```
xlabel('krok(-)')  
ylabel('w(-),u(-),y(-)')  
title('GPC regulace')  
grid on
```

PŘÍLOHA P III: SCRIPT NADRZE_SIMULACE

```
clc
% clear all
global Upred
Upred = [];

T0=60;      %perioda vzorkování regulátoru
N2=10;      %predikční horizont
r=[300*ones(30,1);150*ones(30,1);300*ones(30,1)]; %zadana
hodnota
t=((0:length(r)-N2-1)*T0)'; % o N2 vzorku kratši než r,
prvních N2 vrorku r se použije
                        % jako počáteční stav v bloku Tapped
Delay

X0=[0 0 0]';
Y=[];
X=[];
U=[];
tic
for k=1:length(r)-N2
%     figure
    input = [r(k+1:k+N2); X0];
    Uk=nadrze_optimilizace_simulace(T0, N2, input)
    [tout,xout,yout]=sim('nadrze_model_2_simulace',[0
T0],simset('InitialState',X0),[[0;T0] [Uk; Uk]]);
    X=[X;xout(1:end-1,:)];
    Y=[Y;yout(1:end-1,:)];
    U=[U;Uk];
    X0=xout(end,:)';
end
toc
```

PŘÍLOHA P IV: SCRIPT NADRZE_OPTIMALIZACE_SIMLUACE

```
function U=nadrze_optimilizace_simulace(T0, N2, input)
% parametry:
% T0 - perioda vzorkování regulátoru
% N2 - horizont (v průběhu simulace konstantní)
% input - vektor složený z N2 vzorků w, následují stavy re-
gulovaného systému:
%         input(1:N2) - žádané hodnoty v průběhu horizontu
%         input(N2+1:end) - stavy regulovaného systému

NU =10;          % mohlo by byt také jako parametr
U0 = zeros(NU, 1); % počáteční podmínky 50%
r = input(1:N2);
X0 = input(N2+1:end)';

global Upred
if length(Upred)>0,
    U0=[Upred(2:end,end);Upred(end,end)];
end

% r=[0.6*ones(30,1);0.3*ones(30,1);0.6*ones(30,1)];

% options = op-
timset('Largescale','off','TolX',0.001,'TolFun',0.001,'Algori-
thm','trust-region-reflective',...
% 'DiffMaxChange',1,'Display','final');
options = op-
timset('Largescale','off','TolX',0.001,'TolFun',0.001,...
'MaxFunEvals',35, 'DiffMaxChange',Inf,'Display','iter');
t=((0:N2)*T0)';

% [Uopt,resnorm,residual,exitflag] = lsqnon-
lin(@(U)crit_nadrze(U,X0,r,t), U0, -1*ones(size(U0)),
1*ones(size(U0)), options);
[Uopt,resnorm,residual,exitflag] = lsqnon-
lin(@(U)crit_nadrze_simulace(U,X0,r,t), U0, -
1*ones(size(U0)), 1*ones(size(U0)), options);
% Uopt = lsqnonlin(@(U)crit_nadrze(U,X0,r,t), U0,[],[], opti-
ons);
U=Uopt(1);

Upred = [Upred, Uopt];

[tout,xout,yout]=sim('nadrze_model_2_simulace',t,simset('Init-
ialState',X0),[t,[Uopt; Uopt(end)]]);
plot(tout, xout,t(1:end-1), Uopt*100, t(2:end), r);
grid
```

PŘÍLOHA P IV: SCRIPT CRIT_NADRZE_SIMULACE

```
function F = crit_nadrze_simulace(U,X0,r,t)
% řešitel pro model

lambda = 1;
%konstantní pozice ventilů
% pozice_ventilu = [-0.3 -0.3 0.1 0.4 0.4 0.4];
T0_model=0.01; %perioda vzorkovani simulinkoveho mo-
delu (Fixed step)
T0_reg = t(2)-t(1); %perioda vzorkovani regulátoru:
T0_reg = m * T_model (kde m je prirodzene cislo)
m = T0_reg / T0_model;
% opt = simset('solver','ode45','SrcWorkspace','Current');
uin = [U; U(end)*ones(length(t)-1-length(U),1)];
%
[tout,xout,yout]=sim('nadrze_model',t,simset('InitialState',[
X0, pozice_ventilu]),[t,uin]);
[tout,xout,yout]=sim('nadrze_model_2_simulace',t,simset('Init
ialState',X0),[t,[uin; uin(end)]]);
% F = yout-1; % výpočet chyby
yout = yout(m+1:m:end,1); %prevzorkovat
e = r-yout;
% F = sum(e.^2) + lambda*sum(diff(U).^2);
F = [e; lambda*diff(U)];
% F = yout-r; % výpočet chyby

plot(tout, xout,t(1:end-1),uin*100, t(2:end), r);
grid on;
pause(0.1);
```