České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra řídicí techniky

BAKALÁŘSKÁ PRÁCE

Rychlé numerické metody pro prediktivní řízení

Praha, 2011 **Pavel Otta**

Prohlášení

Prohlašuji, že jsem svou bakalářskou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze dne _____ _____

podpis

Poděkování

Na těchto řádcích chci poděkovat svému vedoucímu bakalářské práce Ing. Ondřeji Šantinovi za pozitivně motivující vedení, vstřícné rady a věcné připomínky. Děkuji také kolegům Michalu Stachovi a Martinu Frodlovi za vytvoření tvůrčí atmosféry na pracovišti.

Zvláštní poděkování patří přátelům a rodině za neutuchající podporu a dvěma nejmenovaným dívkám za nezměrnou inspiraci.

Abstrakt

Prediktivní řízení (MPC) je moderní metoda řízení průsmyslových procesů. Přirozenou výhodou této metody, vyplývající z jejího principu, je možnost brát v úvahu fyzické omezení řízené soustavy a pro definované kritérium nalézt optimální akční zásah. Jádrem regulace je tedy optimalizační proces, při němž je nejčastěji řešen kvadratický program (při zvoleném kvadratickém kritériu optimality, které je možné interpretovat jako energetické kritérium). MPC vede na optimalizační úlohu s omezením, pro které neexistuje analytické řešení jako pro některé optimalizace bez omezení. Pro řízení systémů s krátkou periodou vzorkování je třeba řešit optimalizaci co nejrychleji, a to vhodnými algoritmy. Nicméně pro jednoduchá omezení, např. tvaru box, existují efektivní algoritmy s jejichž pomocí lze regulovat i rychle vzorkované systémy. V rámci této práce byl regulátor s takovým algoritmem implementován.

Abstract

Predictive control (MPC) is modern method for controlling industrial processes. Natural advantage of this method according to its principle is possibility of considering limits of controlled system and then for defined criteria optimal control action can be found. Thus, main element of control is the optimalization process, where mostly quadratic program is being solved (when qaudratic criteria of optimality which may be stated as a energetic criteria is selected). MPC leads to optimalization problem with constraints, where no analytic result exists as for some optimalizations without constraints. Short sampling interval systems control has to be solved as fast as possible by appropriate algorithms. However, for simple constraints f.e. the box shape, effective algorithms with ones aid of controlling even fast sampled systems is possible exist. In terms of this thesis new controller with this algorithm was implemented.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Stavové prediktivní řízení (Model Predictive Control (MPC)) je moderní princip řízení, stále více užívaný v průmyslu. Jeho hlavní myšlenkou je řešení optimalizační úlohy po každém vzorku. Přitom vychází z predikcí, počítaných ze známého diskrétního lineárního modelu reálné soustavy. V každém časovém okamžiku se tedy minimalizuje dané kritérium optimality (nejčastěji kvadratické) přes horizont predikce, v případě blokování vstupů přes horizont korekce. Pro zavedení zpětné vazby se užívá principu zvaného *klouzavý horizont*.

MPC vede na multi-parametrický kvadratický (v případě kvadratického kritéria) program (mp-QP), který je možné řešit *offline*, a jehož parametry jsou stavové vektory [1]. Potom regulátor *online* pouze vyhledá v tabulce a přiřadí každému stavovému vektoru odpovídající akční zásah. Jedná se o velmi efektivní přistup, ale jak se ukazuje, jen pro systémy s málo stavy a krátkým horizontem korekce, jinak je příliš paměťově náročný [2].

Druhou možností jsou *online* solvery, které vypočítávají odpovídající akční zasahy v každém kroce. Pro rychlé solvery aplikované na velké a relativně rychlé systémy se užívají v zasadě dvě rozdílné numerické metody a jejich modifikace. Metoda aktivních množin (Active Set Method (ASM)), jež má levné iterace ($\mathcal{O}(n^2)$), avšak k nalezení optima je jich třeba mnoho, naproti metodě vnitřního bodu (Interior-Point Method (IP)) s komplexitou $\mathcal{O}(n^3)$ a relativně málo iteracemi [3].

Motivací pro tuto práci jsou rychle vzorkované systémy, k nimž je zapotřebí rychlý QP solverů. Po přihlédnutí k výše zmíněným nedostatkům explicitního řešení jsme se rozhodli vytvořit *online* MPC regulátor s implementovaným algoritmem gradientní projekce (GP), vycházející z ASM.

Struktura tohoto dokumentu je tato: po úvodu následuje druhá kapitola o prediktivním řízení a jeho provázaností s numerickými metodami. Celá třetí kapitola je věnována

numerickým metodám a čtvrtá pak implementaci algoritmu jedné z metod, postupu implementace a zhodnocení hotového algoritmu. Na závěr pohovoříme o dosaženém výsledku a přínosu této práce.

1.1 Značení

V této práci je užito následujícího značení:

$\boldsymbol{A}$	matice
$\boldsymbol{A} \succ 0$	pozitivně definitní matice
$\boldsymbol{A} \succeq 0$	pozitivně semidefinitní matice
$\boldsymbol{x}_k$	vektor
$\boldsymbol{x}_{k,N}$	posloupnost vektorů $[\boldsymbol{x}_k^T, \boldsymbol{x}_{k+1}^T, ..., \boldsymbol{x}_{k+N}^T]^T$
$\boldsymbol{x}_{k,N}^*$	optimální posloupnost vektorů
$\boldsymbol{I}_n$	jednotková matice řádu n
$\mathbb{N}, \mathbb{R}$	množina přirozených, reálných čísel
$\mathcal{A}$	množina
α	konstanta

Kapitola 2

Prediktivní řízení

V teorii řízení se obvykle setkáme se dvěma typy úloh. Je to tzv. problém regulátoru a problém sledování. V moderní teorii řízení jsou oba tyto problémy řešeny jako problémy optimalizační. Veškeré požadavky na řízení jsou zahrnuty v kriteriu kvality řízení, které optimalizujeme (minimalizujeme), a je zároveň prostředkem k dosažení našich cílů (sledování, stabilizace, atd.). Navíc, nejenže řídíme, ale v jistém ohledu řídíme nejlépe [4].

Ačkoli forem prediktivního řízení je mnoho, stalo se synonymem pro MPC (stavové řízení), dosahujícího skvělých výsledků. Prediktivní řízení, jak již z názvu vyplývá, predikuje chování (stavy a výstupy) systému na základě jeho modelu na horizontu predikce. Na základě této predikce je optimalizováno kritérium kvality řízení. Výsledkem je optimální akční zásah, vedoucí k požadovanému chování systému podél horizontu predikce. Obecně tedy regulační systém v každém časovém okamžiku k vykoná:

1. Predikce chování systému

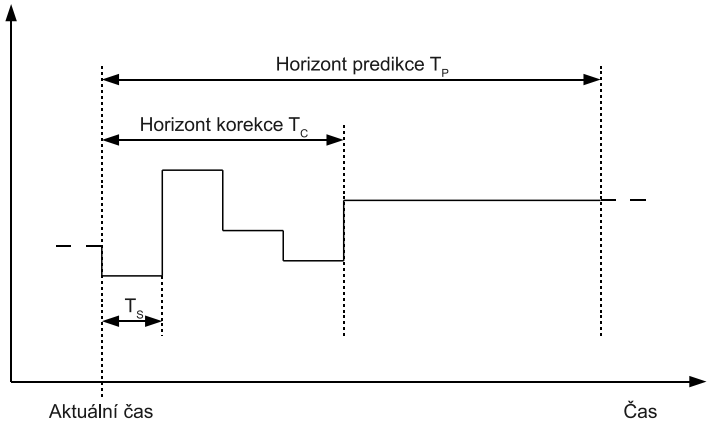
2. Výpočet optimálního akčního zásahu na základě predikce

Hlavními výhodami MPC je především schopnost brát v úvahu omezení reálného systému. Dále pak dovoluje systematický a jednoduchý návrh řízení i pro soustavy s mnoha vstupy a mnoha výstupy (MIMO), kde je řízení klasickými metodami příliš obtížné [5].

Tato práce se zabývá prediktivním řízením s kvadratickým kritériem optimality s omezením typu „box"(limity na akční zásah), aplikovaným na lineárně časově invariantní (LTI) diskrétní model systému.

2.1 Predikce chování systému

V MPC se nejprve provede predikce chování systému na horizontu predikce T_P, následně řešení optimalizační úlohy na horizontu korekce T_C. Poté se aplikuje nalezené řešení. Pojmy jsou blíže specifikovány na obr. 2.1 a T_S je perioda vzorkování.



Obrázek 2.1: Horizont predikce a korekce

Obrázek 2.1 vystihuje případ, kdy dochází k blokování vstupů ($T_C < T_P$). Hlavním ukazatelem složitosti příslušné optimalizační úlohy jsou stupně volnosti. Stupně volnosti (složitost úlohy) redukujeme fixováním (blokováním) vstupů (nebo změny vstupů) za horizontem korekce ($T_C < t < T_P$), tedy řízení probíhá na horizontu korekce a po zbylý čas zůstává konstantní na hodnotě posledního akčního zásahu [6].

Níže uvedený postup predikce lze provádět pouze nad lineárními modely. Díky tomu je predikce takto jednoduchá. Jak lze predikovat na nelineárních modelech je popsáno mimo jiné i v [7].

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k \\ \boldsymbol{y}_k &= \boldsymbol{C}\boldsymbol{x}_k + \boldsymbol{D}\boldsymbol{u}_k, \end{aligned} \tag{2.1}$$

kde $\boldsymbol{u}_k \in \mathbb{R}^m$ je vektor řídicího vstupu, $\boldsymbol{x}_k \in \mathbb{R}^n$ je stavový vektor, $\boldsymbol{y}_k \in \mathbb{R}^p$ je vektor výstupu a kde $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, $\boldsymbol{B} \in \mathbb{R}^{n\times m}$, $\boldsymbol{C} \in \mathbb{R}^{p\times n}$, $\boldsymbol{D} \in \mathbb{R}^{p\times m}$ jsou matice popisující chování systému.

Výjdeme-li z modelu 2.1, analogicky můžeme psát pro sled stavů v časových okamžicích

$k+1, k+2, \ldots$:

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k \\ \boldsymbol{x}_{k+2} &= \boldsymbol{A}\boldsymbol{x}_{k+1} + \boldsymbol{B}\boldsymbol{u}_{k+1}. \\ &\vdots \end{aligned} \tag{2.2}$$

Pro sled výstupních vektorů v časových okamžicích $k+1, k+2, \ldots$:

$$\begin{aligned} \boldsymbol{y}_{k+1} &= \boldsymbol{C}\boldsymbol{x}_{k+1} + \boldsymbol{D}\boldsymbol{u}_{k+1} \\ \boldsymbol{y}_{k+2} &= \boldsymbol{C}\boldsymbol{x}_{k+2} + \boldsymbol{D}\boldsymbol{u}_{k+2}. \\ &\vdots \end{aligned} \tag{2.3}$$

Rekurzivním dosazováním vyjádříme v předchozích rovnicích $\boldsymbol{x}_k$.

$$\underbrace{\begin{bmatrix} \boldsymbol{x}_{k+1} \\ \boldsymbol{x}_{k+2} \\ \vdots \\ \boldsymbol{x}_{k+T_P} \end{bmatrix}}_{\boldsymbol{x}_{k+1,T_P}} = \underbrace{\begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{A}^2 \\ \vdots \\ \boldsymbol{A}^{T_P-1} \end{bmatrix}}_{\boldsymbol{V}_x} \boldsymbol{x}_k + \underbrace{\begin{bmatrix} \boldsymbol{B} & \boldsymbol{0} & & \\ \boldsymbol{A}\boldsymbol{B} & \boldsymbol{B} & & \\ \vdots & & \ddots & \\ \boldsymbol{A}^{T_P-1}\boldsymbol{B} & & & \boldsymbol{B} \end{bmatrix}}_{\boldsymbol{S}_x} \underbrace{\begin{bmatrix} \boldsymbol{u}_k \\ \boldsymbol{u}_{k+1} \\ \vdots \\ \boldsymbol{u}_{k+T_P-1} \end{bmatrix}}_{\boldsymbol{u}_{k,T_P-1}} \tag{2.4}$$

$$\underbrace{\begin{bmatrix} \boldsymbol{y}_{k+1} \\ \boldsymbol{y}_{k+2} \\ \vdots \\ \boldsymbol{y}_{k+T_P} \end{bmatrix}}_{\boldsymbol{y}_{k+1,T_P}} = \underbrace{\begin{bmatrix} \boldsymbol{C}\boldsymbol{A} \\ \boldsymbol{C}\boldsymbol{A}^2 \\ \vdots \\ \boldsymbol{C}\boldsymbol{A}^{T_P} \end{bmatrix}}_{\boldsymbol{V}_y} \boldsymbol{x}_k + \underbrace{\begin{bmatrix} \boldsymbol{C}\boldsymbol{B} & \boldsymbol{D} & & \\ \boldsymbol{C}\boldsymbol{A}\boldsymbol{B} & \boldsymbol{C}\boldsymbol{B} & \ddots & \\ \vdots & & \ddots & \boldsymbol{D} \\ \boldsymbol{C}\boldsymbol{A}^{T_P-1}\boldsymbol{B} & & & \boldsymbol{C}\boldsymbol{B} \end{bmatrix}}_{\boldsymbol{S}_y} \boldsymbol{u}_{k,T_P-1}, \tag{2.5}$$

Potom soustavě rovnic 2.2 odpovídá 2.4, sledu 2.3 maticová rovnice 2.5, kde $T_P \in \mathbb{N}$ je konečný horizont predikce tj. časový okamžik, po který bude prováděn výpočet predikce. Zkráceně lze předchozí rovnice napsat v kompaktnějším tvaru:

$$\begin{aligned} \boldsymbol{x}_{k+1,T_P} &= \boldsymbol{V}_x\boldsymbol{x}_k + \boldsymbol{S}_x\boldsymbol{u}_{k,T_P-1} \\ \boldsymbol{y}_{k+1,T_P} &= \boldsymbol{V}_y\boldsymbol{x}_k + \boldsymbol{S}_y\boldsymbol{u}_{k,T_P-1}. \end{aligned} \tag{2.6}$$

Blokování zavedeme do predikce pozměněním předešlých rovnic:

$$\begin{aligned} \boldsymbol{x}_{k+1,T_P} &= \boldsymbol{V}_x\boldsymbol{x}_k + \acute{\boldsymbol{S}}_x\boldsymbol{u}_{k,T_C-1} \\ \boldsymbol{y}_{k+1,T_P} &= \boldsymbol{V}_y\boldsymbol{x}_k + \acute{\boldsymbol{S}}_y\boldsymbol{u}_{k,T_C-1}, \end{aligned} \tag{2.7}$$

kde $\acute{\boldsymbol{S}}_* = \boldsymbol{S}_* \boldsymbol{M}_b$, $\boldsymbol{M}_b$ je matice jednotkových bloků, která má pro systémy s jedním vstupem a jedním výstupem (SISO) rozměr $T_P \times T_C$ a pro MIMO $m \cdot T_P \times m \cdot T_C$. Pro příklad blokující matice pro SISO soustavu a $T_P = 4$, $T_C = 2$ má tvar:

$$\boldsymbol{M}_b = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix}^T . \tag{2.8}$$

Predikce přímo vychází z modelu systému. Je jasné, že přesnost takového odhadu bude závislá na přesnosti modelu. Čím přesnější model máme k dispozici, tím přesnější bude predikce.

2.2 Základní algoritmus prediktivního řízení

MPC využívá explicitního stavového popisu pro určení predikce na horizontu T_P. V každém časovém okamžiku řeší omezenou optimalizační úlohu, kterou nalezne řídicí zásahy na horizontu T_C. Přitom se užívají postupy neodlučitelně spjaté s MPC, jako blokování vstupů (nejen vstupů) pro zjednodušení optimalizačního problému (snížení výpočetní náročnosti) nebo klouzavý horizont, přizpůsobující algoritmus pro řešení reálných úkolů.

2.2.0.1 Klouzavý horizont

Výsledkem optimalizační úlohy MPC je řešení na konečném horizontu, tedy řízení v otevřené smyčce. Takové řešení by bylo vhodné pouze v ideálním případě, kdy bychom měli k dispozici uplně přesný model systému, na systém by nepůsobila žádná vnější porucha a měření by nebylo zatíženo šumem, což ale není reálné. Proto se užívá klouzavého horizontu (RHC - Reciding Horizont Control), který do řízení zavádí zpětnou vazbu [8]. Jedná se o iterační metodu, která v každém kroku k (časový okamžik):

1. Naměří stavy systému.
2. Vypočte vektor optimálních (dle určitého kriteria) akčních zásahů $\boldsymbol{u}^*_{k,T_P-1}$ na časovém horizontu $k+1 \leq t \leq k+T_P$ vyřešením optimalizační úlohy.
3. Aplikuje řízení $\boldsymbol{u}^*_k = [\boldsymbol{I}, \boldsymbol{0}, ..., \boldsymbol{0}]\, \boldsymbol{u}^*_{k,T_P-1}$, tedy pouze první prvek řídicí sekvence.

Tímto postupem jsme získali akční zásah pro okamžik k. Pro následující okamžik $k+1$ se provede celý algoritmus znovu, s tím rozdílem, že za k dosadíme $k+1$, atd..

2.2.1 MPC bez omezení

MPC bez omezení je nejjednoduší, avšak spíše akademický případ řízení, neboť omezení je to, co se od MPC nejvíce očekává a prakticky nejvíce využívá. Díky absenci omezení je možné řízení řešit jako neomezený optimalizační problém. Ten jsme schopni řešit poměrně efektivně, o čemž je zmínka v následující kapitole.

2.2.1.1 Problém regulátoru

Pokud jsou dynamické vlastnosti systému nevyhovující, je cílem řízení vytvořit takovou řídicí strukturu, aby vyhovující byly (problém regulátoru). Může se jednat například o nestabilní systém, ke kterému navrhneme regulátor systém stabilizující.

Pro tento problém můžeme zformulovat ztrátovou funkci, která v plném znění má tvar:

$$\begin{aligned} J &= \frac{1}{2}\left(\sum_{j=k+1}^{k+T_P} \boldsymbol{x}_j^T \boldsymbol{Q} \boldsymbol{x}_j + \sum_{j=k}^{k+T_C-1} \boldsymbol{u}_j^T \boldsymbol{R} \boldsymbol{u}_j\right) = \qquad (2.9a) \\ &= \left(\boldsymbol{V}_x \boldsymbol{x}_k + \acute{\boldsymbol{S}}_x \boldsymbol{u}_{k,T_C-1}\right)^T \acute{\boldsymbol{Q}} \left(\boldsymbol{V}_x \boldsymbol{x}_k + \acute{\boldsymbol{S}}_x \boldsymbol{u}_{k,T_C-1}\right) + \boldsymbol{u}_{k,T_C-1}^T \acute{\boldsymbol{R}} \boldsymbol{u}_{k,T_C-1} = \\ &= \boldsymbol{u}_{k,T_C-1}^T \left(\acute{\boldsymbol{S}}_x^T \acute{\boldsymbol{Q}} \acute{\boldsymbol{S}}_x + \acute{\boldsymbol{R}}\right) \boldsymbol{u}_{k,T_C-1} + 2\mathbf{x}_k^T \boldsymbol{V}_x^T \acute{\boldsymbol{Q}} \acute{\boldsymbol{S}}_x \boldsymbol{u}_{k,T_C-1} + \boldsymbol{x}_k^T \boldsymbol{V}_x^T \acute{\boldsymbol{Q}} \boldsymbol{V}_x \boldsymbol{x}_k, \end{aligned}$$

za předpokladu:

$$\boldsymbol{Q} = \boldsymbol{Q}^T \succeq 0, \ \boldsymbol{R} = \boldsymbol{R}^T \succ 0, \qquad (2.9b)$$

přičemž $\boldsymbol{Q} \in \mathbb{R}^{p \cdot T_P \times p \cdot T_P}$ a $\boldsymbol{R} \in \mathbb{R}^{m \cdot T_C \times m \cdot T_C}$. Optimální akční zásah nalezneme minimalizací kritéria, které je ekvivalentní s předchozím, ačkoli zanedbáme konstantní člen:

$$\begin{aligned} \boldsymbol{u}_{k,T_C-1}^* &= \min_{\boldsymbol{u}_{k,T_C-1}} J(\boldsymbol{u}_{k,T_C-1} \mid \mathbf{x}_k) = \qquad (2.10) \\ &= \min_{\boldsymbol{u}_{k,T_C-1}} \frac{1}{2} \boldsymbol{u}_{k,T_C-1}^T \underbrace{\left(\acute{\boldsymbol{S}}_x^T \acute{\boldsymbol{Q}} \acute{\boldsymbol{S}}_x + \acute{\boldsymbol{R}}\right)}_{\boldsymbol{H}} \boldsymbol{u}_{k,T_C-1} + \boldsymbol{x}_k^T \underbrace{\boldsymbol{V}_x^T \acute{\boldsymbol{Q}} \acute{\boldsymbol{S}}_x}_{\boldsymbol{F}} \boldsymbol{u}_{k,T_C-1} = \\ &= \min_{\boldsymbol{u}_{k,T_C-1}} \frac{1}{2} \boldsymbol{u}_{k,T_C-1}^T \boldsymbol{H} \boldsymbol{u}_{k,T_C-1} + \boldsymbol{x}_k^T \boldsymbol{F} \boldsymbol{u}_{k,T_C-1}, \end{aligned}$$

V 2.10 jsou $\boldsymbol{H} \succ 0$ a $\boldsymbol{F}$ matice definující problém a $\boldsymbol{x}_k$ je známý vektor stavů pro daný krok. Pro řešení MPC je ještě výhodné definovat vektor $\boldsymbol{f}^T = \boldsymbol{x}_k^T \boldsymbol{F}$, jak uvidíme dále.

2.2.2 MPC s omezením

Největší výhodou MPC je bezpochyby možnost zahrnout do regulace omezení reálného systému (stavy, vstupy, atd.). V této práci se budeme zabývat pouze omezením rozsahů

řídicích signálů. Pro tento případ obecně platí:

$$\boldsymbol{A}\boldsymbol{u}_k \leq \mathbf{b}, \tag{2.11}$$

je podmínka omezení tvaru mnohostěnu, ze kterého vhodným zvolením matice A a vektoru b, dostaneme omezení tvaru „box"následujícím způsobem:

$$\overbrace{\begin{bmatrix} \boldsymbol{I} \\ -\boldsymbol{I} \end{bmatrix}}^{\boldsymbol{A}} \boldsymbol{u}_k \leq \overbrace{\begin{bmatrix} \boldsymbol{UB} \\ -\boldsymbol{LB} \end{bmatrix}}^{\boldsymbol{b}}, \tag{2.12}$$

kde $\boldsymbol{LB}$ (lower bound) je vektor dolních omezení a $\boldsymbol{UB}$ (upper bound) je vektor horních omezení. Předchozí soustavu rovnic přepíšeme do jediné podmínky:

$$\boldsymbol{LB} \leq \boldsymbol{u}_k \leq \boldsymbol{UB}, \tag{2.13}$$

což je podmínka box omezení.

Obecně lze dělit omezení na dva základní typy:

- Pevná omezení (Hard constraints) - Vyplývá z omezení systému a porušení tohoto omezení je nemyslitelné. Např. není možné požadovat, aby byl ventil více než úplně otevřený, nebo méně než úplně zavřený.

- Měkká omezení (Soft constraints) - Omezení může být porušeno. Příkladem může být vytápění haly, kde je energeticky výhodnější regulovat v rozsahu teplot (například 18-22°C). V případě překročení meze je řízení penalizováno. Takové řízení bývá označováno jako řízení v pásmu (Range Control), tj. případ, kdy je měkké omezení na horní a dolní mez [9].

2.2.2.1 Problém sledování s omezením

Další často řešenou úlohou řízení je problém sledování (tracking problem). Od řešení úlohy tohoto typu nejenže požadujeme změnu vlastností systému na nám vyhovující, jako v předešlé sekci, ale ještě navíc se snažíme, co nejpřesněji sledovat referenci. Pro tyto účely se rozšiřuje stavový popis o referenci:

$$\underbrace{\begin{bmatrix} \boldsymbol{x}_{k+1} \\ \boldsymbol{x}_{ref_{k+1}} \end{bmatrix}}_{\tilde{\boldsymbol{x}}_{k+1}} = \underbrace{\begin{bmatrix} \boldsymbol{A} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{I} \end{bmatrix}}_{\tilde{\boldsymbol{A}}} \underbrace{\begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{x}_{ref_k} \end{bmatrix}}_{\tilde{\boldsymbol{x}}_k} + \underbrace{\begin{bmatrix} \boldsymbol{B} \\ \mathbf{0} \end{bmatrix}}_{\tilde{\boldsymbol{B}}} \boldsymbol{u}_k \tag{2.14}$$

$$\boldsymbol{y}_k = \underbrace{\begin{bmatrix} \boldsymbol{C} & \boldsymbol{0} \end{bmatrix}}_{\tilde{\boldsymbol{C}}} \tilde{\boldsymbol{x}}_k + \boldsymbol{D}\boldsymbol{u}_k, \tag{2.15}$$

Z definice regulační odchylky $\boldsymbol{e} = \boldsymbol{y} - \boldsymbol{r}$ vyplývá:

$$\boldsymbol{e}_k = \underbrace{\begin{bmatrix} \boldsymbol{C} & -\boldsymbol{I} \end{bmatrix}}_{\tilde{\boldsymbol{C}}_e} \tilde{\boldsymbol{x}}_k + \boldsymbol{D}\boldsymbol{u}_k, \tag{2.16}$$

Pro účely sledování je nezbytné vyjádřit predikční vektor regulační odchylky, který bude podobný predikčnímu vektoru výstupu:

$$\underbrace{\begin{bmatrix} \boldsymbol{e}_{k+1} \\ \boldsymbol{e}_{k+2} \\ \vdots \\ \boldsymbol{e}_{k+T_P} \end{bmatrix}}_{\boldsymbol{e}_{k+1,T_P}} = \underbrace{\begin{bmatrix} \tilde{\boldsymbol{C}}_e\tilde{\boldsymbol{A}} \\ \tilde{\boldsymbol{C}}_e\tilde{\boldsymbol{A}}^2 \\ \vdots \\ \tilde{\boldsymbol{C}}_e\tilde{\boldsymbol{A}}^{T_P} \end{bmatrix}}_{\boldsymbol{V}_e} \tilde{\boldsymbol{x}_k} + \underbrace{\begin{bmatrix} \tilde{\boldsymbol{C}}_e\tilde{\boldsymbol{B}} & \tilde{\boldsymbol{D}} & & \\ \tilde{\boldsymbol{C}}_e\tilde{\boldsymbol{A}}\tilde{\boldsymbol{B}} & \tilde{\boldsymbol{C}}_e\tilde{\boldsymbol{B}} & \ddots & \\ \vdots & & \ddots & \tilde{\boldsymbol{D}} \\ \tilde{\boldsymbol{C}}_e\tilde{\boldsymbol{A}}^{T_P-1}\tilde{\boldsymbol{B}} & & & \tilde{\boldsymbol{C}}_e\tilde{\boldsymbol{B}} \end{bmatrix}}_{\boldsymbol{S}_e} \boldsymbol{u}_{k,T_P-1}, \tag{2.17}$$

Zkráceně píšeme:

$$\boldsymbol{e}_{k+1,T_P} = \boldsymbol{V}_e\tilde{\boldsymbol{x}}_k + \acute{\boldsymbol{S}}_e\boldsymbol{u}_{k,T_C-1}, \tag{2.18}$$

kde $\acute{\boldsymbol{S}}_e = \boldsymbol{S}_e\boldsymbol{M}_b$. Pozměníme ztrátové funkci 2.9a tak, že nahradíme vektory výstupů vektory regulačních odchylek:

$$\begin{aligned} J &= \frac{1}{2}\left(\sum_{j=k+1}^{k+T_P} \boldsymbol{e}_j^T\boldsymbol{Q}\boldsymbol{e}_j + \sum_{j=k}^{k+T_C-1} \boldsymbol{u}_j^T\boldsymbol{R}\boldsymbol{u}_j\right) = \ldots \\ &= \boldsymbol{u}_{k,T_C-1}^T\left(\acute{\boldsymbol{S}}_e^T\acute{\boldsymbol{Q}}\acute{\boldsymbol{S}}_e + \acute{\boldsymbol{R}}\right)\boldsymbol{u}_{k,T_C-1} + 2\tilde{\boldsymbol{x}}_k^T\boldsymbol{V}_e^T\acute{\boldsymbol{Q}}\acute{\boldsymbol{S}}_e\boldsymbol{u}_{k,T_C-1} + \tilde{\boldsymbol{x}}_k^T\boldsymbol{V}_e^T\acute{\boldsymbol{Q}}\boldsymbol{V}_e\tilde{\boldsymbol{x}}_k, \end{aligned} \tag{2.19}$$

Po úpravě předcházející ztrátové funkce (přenásobení konstantou a odstraněním na vstupu nezávislém členu) dostáváme ekvivalentní optimalizační úlohu. Navíc ještě přidejme omezení na řídicí veličinu. Pozměněná optimalizační úloha bude mít tvar:

$$\begin{aligned} \boldsymbol{u}_{k,T_C-1}^* &= \min_{\boldsymbol{u}_{k,T_C-1}} J(\boldsymbol{u}_{k,T_C-1} \mid \tilde{\boldsymbol{x}}_k) = \qquad (2.20a) \\ &= \min_{\boldsymbol{u}_{k,T_C-1}} \frac{1}{2}\boldsymbol{u}_{k,T_C-1}^T \underbrace{\left(\acute{\boldsymbol{S}}_e^T\acute{\boldsymbol{Q}}\acute{\boldsymbol{S}}_e + \acute{\boldsymbol{R}}\right)}_{\acute{\boldsymbol{H}}}\boldsymbol{u}_{k,T_C-1} + \underbrace{\tilde{\boldsymbol{x}}_k^T\boldsymbol{V}_e^T\acute{\boldsymbol{Q}}\acute{\boldsymbol{S}}_e}_{\tilde{\boldsymbol{f}}^T}\boldsymbol{u}_{k,T_C-1} = \\ &= \min_{\boldsymbol{u}_{k,T_C-1}} \frac{1}{2}\boldsymbol{u}_{k,T_C-1}^T\tilde{\boldsymbol{H}}\boldsymbol{u}_{k,T_C-1} + \tilde{\boldsymbol{f}}^T\boldsymbol{u}_{k,T_C-1}, \end{aligned}$$

za podmínky:

$$\boldsymbol{L}\boldsymbol{B} \;\leq\; \boldsymbol{u}_k \leq \mathbf{UB},\; k = n, ..., n + T_C - 1. \tag{2.20b}$$

$\boldsymbol{u}_{k,T_C-1}^*$ je nalezené optimální (dle kriteria optimality J) řízení na horizontu $\boldsymbol{T}_C$.

2.2.2.2 Vážení změny řízení

Očekáváme-li od MPC regulátoru ,,integrační" charakter, můžeme 2.20a přeformulovat na tzv. Δ-u formulaci:

$$\min_{\boldsymbol{u}_k} \frac{1}{2} \left(\sum_{j=k+1}^{k+T_P} \boldsymbol{e}_j^T \boldsymbol{Q} \boldsymbol{e}_j + \sum_{j=k}^{k+T_C-1} \Delta \boldsymbol{u}_j^T \boldsymbol{R} \Delta \boldsymbol{u}_j \right),$$

při zachování stávajících podmínek. Důležité je, že optimalizace probíhá podle $\boldsymbol{u}_{k,T_C-1}$ a nikoli podle $\Delta \boldsymbol{u}_{k,T_C-1}^T$, a proto není nutné transformovat omezující podmínky. Analogicky k předešlému můžeme i v tomto případě přepsat úlohu do předpisu tzv. kvadratického programu za podmínky 2.20b:

$$\min_{\boldsymbol{u}_{k,T_C-1}} \frac{1}{2} \boldsymbol{u}_{k,T_C-1}^T \hat{\boldsymbol{H}} \boldsymbol{u}_{k,T_C-1} + \boldsymbol{q}^T \hat{\boldsymbol{F}}^T \boldsymbol{u}_{k,T_C-1},$$

kde za předpokladu problému sledování je vektor $\boldsymbol{q}$ složený ze stavů, referencí a posledních předešlých akčních zásahů. Podrobněji je toto téma rozebráno v [3].

Všechna dříve uvedená kvadratická kritéria optimality vedou na parametrické kvadratické programování, kde parametrem je stav systému $\boldsymbol{x}$, a které je i s postupem řešení popsané v následující kapitole.

Kapitola 3

Numerické metody pro kvadratické programování

Prediktivní regulátor s kvadratickým kritériem optimality vede na kvadratické programování (QP). Odteď tedy uvažujme vždy, že optimalizační problém, který řešíme, je QP. Složitější optimalizačních úlohy obvykle nemají analytické řesení a jedinou možností, jak je řešit, jsou numerické metody. Pro neomezené kvadratické programy vychází nejlépe, co se do počtu iterací týče, Newtonova metoda. Pro omezené QP jsou to především modifikované metody aktivních množin a metoda vnitřních bodů, kterými je možné efektivně hledat optima daného problému [10]. Naproti tomu závěr kapitoly je věnovaný explicitnímu řešení QP.

Optimalizačním problémem budeme rozumět hledání minima. Hledání maxima je obdobné, protože $\min f(\boldsymbol{x}) = -\max(-f(\boldsymbol{x}))$.

3.1 Kvadratické programování

Pojmem kvadratické programování se označují optimalizační úlohy s kvadratickým kritériem optimality. Tento problém bývá obecně vyjadřován úlohou:

$$\min_{\boldsymbol{z}} \frac{1}{2}\boldsymbol{z}^T\boldsymbol{H}\boldsymbol{z} \quad + \quad \boldsymbol{f}^T\boldsymbol{z} + \boldsymbol{c}, \tag{3.1a}$$

za předpokladu:

$$\boldsymbol{A}\boldsymbol{z} \quad \leq \quad \boldsymbol{b}, \tag{3.1b}$$

kde $\boldsymbol{H}$ je symetrická čtvercová ($\boldsymbol{H} = \boldsymbol{H}^T \in \mathbb{R}^{n\times n}$) pozitivně semi-definitní ($\boldsymbol{H} \succeq 0$) matice, $\boldsymbol{f} \in \mathbb{R}^n$, $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, $\boldsymbol{b} \in \mathbb{R}^m$ jsou známá data. Konstantní vektor $\boldsymbol{c}$ může být zanedbán, neboť na výsledek optimalizace nemá vliv. Výše uvedené vztahy tedy vyjadřují minimalizaci kvadratické funkce přes polytop.

V případě, že $\boldsymbol{H}$ je pozitivně definitní, jako v našem případě [11], kvadratický program má pouze jeden globální extrém, protože $\boldsymbol{H}$, matice druhých derivací, je Hessova matice.

3.2 Optimalizační problémy bez omezení

Mějme kvadratický problém

$$\min_{\boldsymbol{z}} \frac{1}{2}\boldsymbol{z}^T\boldsymbol{H}\boldsymbol{z} + \boldsymbol{f}^T\boldsymbol{z},\ \boldsymbol{z} \in \mathbb{R}^n, \tag{3.2}$$

na který budeme následující algoritmy aplikovat. Zde uvedené numerické metody bez omezení jsou iterativní, využívaje znalosti gradientu (směr největšího růstu) funkce, nebo dokonce druhých derivací (Newtonova metoda). Lze je tedy aplikovat pouze na problémy nejméně jednou, v případě Newtonovy metody, dvakrát derivovatelné, což QP splňuje. Pro konvexní funkce, při zvolení postačujících parametrů, vždy konvergují do globálního extrému. Tyto metody jsou základem pro metody s omezením, jenž použijeme později v regulátoru, jako základ optimalizačního algorimtu MPC.

3.2.1 Podmínka prvního řádu

Představíme-li si eliptický paraboloid, který representuje funkci kvadratického programu, jehož Hessian je pozitivně definitní, pak je zřejmé, že tato funkce bude mít pouze jeden globální extrém, pro který musí platit:

$$\nabla f(\boldsymbol{x}^*) = \boldsymbol{0}. \tag{3.3}$$

Této podmínce se také říká podmínka optimality prvního řádu [11].

3.2.2 Metoda nejstrmějšího sestupu

Gradientní algoritmy v každé iteraci k vypočítávají derivaci (gradient) funkce $\boldsymbol{g}_k = \nabla f(\boldsymbol{x}_k)$. V případě kvadratického problému (3.2) je gradient $\boldsymbol{g}_k = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{f}$. Pro klasické gradientní metody je typické, že platí $f(\boldsymbol{x}_{k+1}) < f(\boldsymbol{x}_k), \forall k \in \mathbb{N}$, to znamená, že

v každé iteraci klesá hodnota minimalizované funkce. Principem gradientní metody je posun z minulé do následující pozice ve směru záporně vzatého gradientu:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \boldsymbol{g}_k, \tag{3.4}$$

kde $\alpha_k > 0$ je velikost kroku. Iterace provádíme, dokud není splněna podmínka 3.3. Algoritmus nazýváme metodou nejstrmějšího sestupu, pokud parametr α, určující délku posunu, je v každé iteraci optimální ve smyslu, že dosáhne minima ve směru záporně vzatého gradientu. Optimální α v kroce k dostaneme jako

$$\alpha_k = \frac{\boldsymbol{g}_k^T \boldsymbol{g}_k}{\boldsymbol{g}_k^T \boldsymbol{H} \boldsymbol{g}_k}. \tag{3.5}$$

Metoda nejstrmějšího sestupu má tedy algoritmus:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \frac{\boldsymbol{g}_k^T \boldsymbol{g}_k}{\boldsymbol{g}_k^T \boldsymbol{H} \boldsymbol{g}_k} \boldsymbol{g}_k, \tag{3.6}$$

Pro velmi ,,zploštělé"problémy (hodnocené pomocí čísla podmíněnosti) a nevhodně zvolenou počáteční pozici $\boldsymbol{x}_0$, bude algoritmus tzv. zig-zagovat (viz. obr. 3.1). Vylepšenou gradientní metodou, odstraňující tento neduh, je metoda sdružených gradientů.

3.2.3 Metoda sdružených gradientů

Lepších výsledků dosahuje upravená gradientní metoda, totiž metoda sdružených gradientů. Pro QP metoda konverguje nejvýše v n iteracích, kde n je dimenze problému, avšak ukazuje se, že pro určitá rozložení vlastních čísel Hessovy matice algoritmus konverguje mnohem rychleji [12]. Principiální rozdíl je, že namísto pohybu ve směru záporně vzatého gradientu se algoritmus přesouvá konjungovaným směrem, který se z gradientu počítá [13]. Analogicky ke klasické gradientní metodě píšeme:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \boldsymbol{s}_k, \tag{3.7}$$

kde $\alpha_k > 0$ je optimální takové, že algoritmus se přesune v daném směru do minima:

$$\alpha_k = \frac{\boldsymbol{s}_k^T \boldsymbol{g}_k}{\boldsymbol{s}_k^T \boldsymbol{H} \boldsymbol{s}_k}. \tag{3.8}$$

Konjugovaný (sdružený) směr odvodíme z gradientu jako:

$$\boldsymbol{s}_{k+1} = \boldsymbol{g}_{k+1} - \beta_k \boldsymbol{s}_k, \tag{3.9}$$

kde $\beta_k > 0$ je optimální ve smyslu $\beta_k = arg\min_\beta(\boldsymbol{x}_{k+1} - \alpha_{k+1}(\boldsymbol{g}_{k+1} - \beta_k \boldsymbol{s}_k))$ odtud plyne:

$$\beta_k = \frac{\boldsymbol{s}_k^T \boldsymbol{H} \boldsymbol{g}_{k+1}}{\boldsymbol{s}_k^T \boldsymbol{H} \boldsymbol{s}_k}\,. \tag{3.10}$$

Chceme-li se vyhnout počtu s Hessovou maticí, je nasnadě použít například Fletcher-Reevesův algoritmus [14], podle kterého je

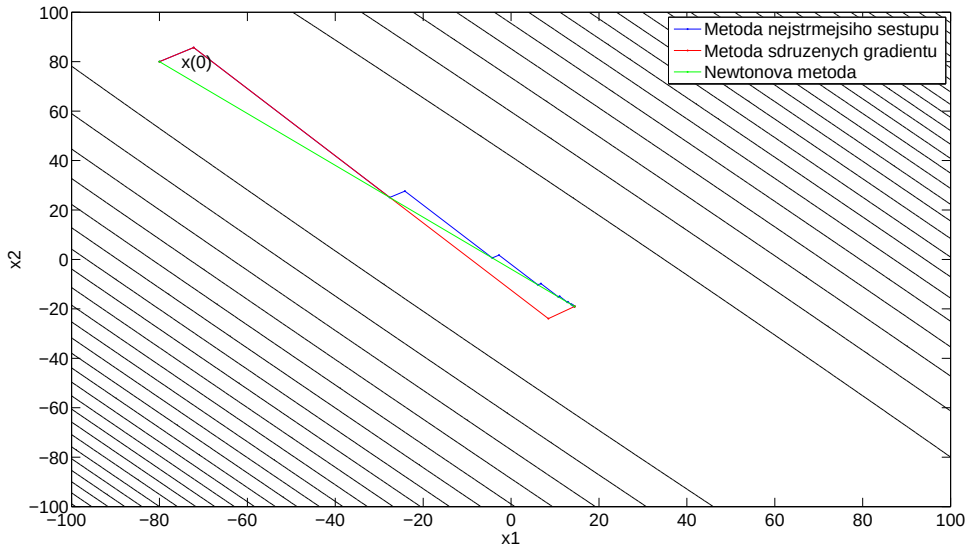
$$\beta_k = \frac{\boldsymbol{g}_{k+1}^T \boldsymbol{g}_{k+1}}{\boldsymbol{g}_k^T \boldsymbol{g}_k}\,, \tag{3.11}$$

3.2.4 Newtonova metoda

Pro kvadratickou funkci Newtonův algoritmus dosáhne minima již v první iteraci, a to po přesunu o Newtonův krok $\boldsymbol{p}_k = -\boldsymbol{H}^{-1}\boldsymbol{g}_k$. V obecnějším případě nekvadratické funkce se Hessova matice nahrazuje maticí druhých derivací [13]. Obecně má Newtonova iterace tvar:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - (\nabla^2 f(\boldsymbol{x}))^{-1}\boldsymbol{g}_k \tag{3.12}$$

Algoritmus se dostane do minima v extrémě nízkém počtu iterací, pro QP bez omezení v první iteraci, avšak výpočet Newtonova kroku, resp. nalezení matice $\boldsymbol{H}^{-1}$ je výpočetně náročný. Lze toho dosáhnout například Choleského faktorizací, pak pro pozitivně definitní $\boldsymbol{H}$ je složitost této metody je $\mathcal{O}(n^3)$.



Obrázek 3.1: Porovnání průběhů iterací metod na neomezeném problému

3.2.5 Porovnání metod

Metoda nejstrmějšího sestupu se ze všech uvedených algoritmů jeví, co do počtu iterací nejhůře (obr. 3.1). Lépe je na tom metoda sdružených gradientů, která odstranila problém zvaný *zig-zaging*. Dalo by se říci, že sdružené gradienty jsou přechodem od nejstrmějšího sestupu, využívajícího informace o gradientu, k Newtonově metodě, pracující se všemi druhými parciálními derivacemi. Sama počítá se sdruženými gradienty, odvozenými např. ze dvou předešlých gradientů (metoda Fletchera-Reevese (3.11)).

3.3 Optimalizační problémy s omezeními

Nyní můžeme uvažovat kvadratický program včetně jeho omezení (3.1). Uvědomme si, že hledané minimum nemusí být shodné s minimem funkce bez omezení.

Nejčastěji používanou metodou na problémy s obecným omezením je metoda aktivních množin. Pro případy jednodušších omezení jsou výhodnější metody projekce gradientu. Obě tyto metody jsou popsány v této sekci. Dále nás budou více zajímat problémy jednoduše omezené, proto předpokládejme omezení typu box a tedy zjednodušenou omezující podmínku 2.13.

Dalšími, jako jsou například metoda přípustných směrů, metoda vnitřního bodu, metoda barierových funkcí, nebo metoda pokutových funkcí, se zabývat nebudeme. Pozn. pro úplnost: ke všem těmto metodám existuje mnoho modifikací, vynikajících v určitých případech, a jejich přehled je možné nalézt např. v [16].

3.3.1 KKT podmínky

Pro omezené optimalizační problémy nemusí být hledané minimum extrémem funkce, a proto musíme do algoritmů přidávat podmínky, které dokáží posoudit, zda-li bod ležící na omezení je minimum na množině dané omezeními. Jednou z možností je vyjít z metody Lagrangeových multiplikátorů. Definujme Lagrangeovu funkci pro QP problém 3.1 jako:

$$L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \frac{1}{2}\,\boldsymbol{x}^T \boldsymbol{H}\boldsymbol{x} + \boldsymbol{f}^T\boldsymbol{x} + \boldsymbol{\lambda}(-\boldsymbol{x} + \boldsymbol{LB}) + \boldsymbol{\mu}(\boldsymbol{x} - \boldsymbol{UB}), \tag{3.13}$$

kde $\boldsymbol{\lambda}$ je multiplikátor pro dolní omezení a $\boldsymbol{\mu}$ pro horní. Z podmínky nulovosti první derivace Lagrangianu 3.13 kvadratického programu dostaneme **Karush-Kuhn-Tuckerovy**

podmínky [11] ve tvaru:

$$\nabla L(x) = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{f}^T - \boldsymbol{\lambda} + \boldsymbol{\mu} = \mathbf{0}, \tag{3.14a}$$

$$\boldsymbol{LB} \leq \boldsymbol{x} \leq \boldsymbol{UB}, \tag{3.14b}$$

$$\boldsymbol{\lambda}^T(-\boldsymbol{x} + \boldsymbol{LB}) = \mathbf{0}, \tag{3.14c}$$

$$\boldsymbol{\mu}^T(\boldsymbol{x} - \boldsymbol{UB}) = \mathbf{0}, \tag{3.14d}$$

$$\boldsymbol{\lambda}, \boldsymbol{\mu} \geq \mathbf{0}. \tag{3.14e}$$

Vztahy 3.14c a 3.14d jsou podmínky komplementarity, které nám říkají, že v případě bodu ležícím na omezení je Lagrangeův multiplikátor tohoto omezení nenulový a naopak. Pro box omezení 3.14b navíc platí pro bod na omezení, že pokud je Lagrangeův multiplikátor odpovídající dolnímu, nebo horního omezení nenulový, tak pro komplementární (horní, nebo dolní) omezení nulový. Z toho plyne, že pro body ležící na omezení bude mít rovnice 3.14a pouze jeden neznámý multiplikátor (druhý bude nulový).

3.3.2 Metoda aktivních množin

Algoritmus využívá Newtonovy iterace. Pokud je hledaný bod uvnitř omezení, řešení bude stejné jako u neomezeného problému, v opačném případě:
Řešíme-li úlohu $\min\{f(\boldsymbol{x}) : lb_i \leq x_i \leq ub_i,\ i = 1, ..., n\}$, kde lb_i a ub_i jsou komponenty $\boldsymbol{LB}$ a $\boldsymbol{LB}$ a kde x_i (popř. x_i^*) je příslušná komponenta $\boldsymbol{x}$ (popř. $\boldsymbol{x}^*$). Pak pro optimální bod $\boldsymbol{x}^*$ platí: $b_i = x_i^*,\ i \in \mathcal{A}$, kde b_i je komponenta omezení a $\mathcal{A}$ je konečná množina aktivních omezení. Ostatní omezení nejsou podstatná, čimž se problém zjednodušší na $\min\{f(\boldsymbol{x}) : b_i = x_i,\ i \in \mathcal{A}\}$. Množinu aktivních omezení $\mathcal{A}$ ale neznáme, a tak si volíme pracovní množinu $\mathcal{W}$, od níž se k $\mathcal{A}$ dopracujeme v průběhu algoritmu. Počáteční zvolené pozici algoritmu odpovídá $b_i = x_i,\ i \in \mathcal{W}$ a $b_i \neq x_i,\ i \notin \mathcal{W}$ [13]. Nejprve se spočítá a aplikuje Newtonův krok. Následně jsou dvě možnosti:

- Narazí-li algoritmus na další omezení, které není aktivní, přidá jej do pracovní množiny.
- Nenarazí-li, otestuje aktuální pozici na podmínky 3.14. Splňuje-li je, pozice je hledaný bod, nesplňuje-li, pak nalezne a vyřadí z pracovní množiny takové omezení, které brání nalezení lepšiho řešení.

Algoritmus je poměrně jednoduchý a efektivní a užívá se také proto, že je vhodný i pro obecná omezení. Jeho výpočetní náročnost je $\mathcal{O}(n^2)$. Důležité mezení tohoto algoritmu

je možnost přidat, či odebrat pouze jedno omezení do množiny aktivních omezení v jedné iteraci, a to pro rozsáhle problémy s velkým počtem aktivních omezení v optimu znamená velké množství iterací se složitostí $\mathcal{O}(n^2)$. QP pocházející z MPC má obvykle velké množství aktivních omezení v optimu [17].

Mnohem efektivnější je pro jednoduchá omezení projekce gradientu, vycházející z myšlenky aktivních množin.

3.3.3 Projekce gradientu

Postup je podobný jako u neomezeného problému. V prvé řadě algoritmus nalezne $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \boldsymbol{g}_k$ (3.4), kde $\alpha_k > 0$ můžeme stejně jako v neomezené variaci problému volit v zásadě trojím způsobem: optimální (3.5), konstantní, anebo tak, že $\|\alpha_k \boldsymbol{g}_k\| = konst.$. Pro obecně zvolený konstantní krok α_k není konvergence zaručena. Nejjednoduší volbou se ukazuje být $\alpha_k = \frac{1}{L}$, pro které konvergence dokázána je. L je Lipschitzova konstanta, která je rovna největšímu vlastnímu číslu Hessianu [18].

Za další je třeba zaručit přípustnost určené pozice $\boldsymbol{x}_{k+1}$ a to její projekcí, zobrazením $\mathbb{R}^n \to \mathcal{X}$, kde $\mathcal{X} \subseteq \mathbb{R}^n$ je neprázdná konvexní množina přípustných bodů.

Pro obecné omezení je tato projekce řešením optimalizační úlohy

$$\mathcal{P}(\boldsymbol{y}) = \arg\min_{\boldsymbol{x}} \|\boldsymbol{x} - \boldsymbol{y}\|, \boldsymbol{x} \in \mathcal{X}. \tag{3.15}$$

Gradientní algoritmy mají levné iterace a proto i u tohoto algoritmu je celková složitost pouze $\mathcal{O}(n^2)$, protože se počítá pouze gradient. Projekce gradientu svojí myšlenkou vychází z metody aktivních množin. Oproti ní má však výhodu, že v každé iteraci může být přidáno nebo odebráno libovolné množství omezení do (z) množiny aktivních omezení. Pro tuto metodu lze vypočítat horní odhad iterace, do které algoritmus nalezne ϵ-optimální řešení [18].

Na druhou stranu, vedle výše uvedených výhod algoritmus konverguje pouze lineárně, a tak v poměrně velkém počtu iterací. Řešení 3.15 je relativně náročné, a proto pokud je to možné, snažíme se použít omezení, vyžadující méně výpočetně komplikované postupy. Takovýmito omezeními jsou například omezení tvaru simplexu, omezení v kvadrantu nebo box omezení [10].

3.3.3.1 Projekce gradientu pro box omezení

Metoda projekce gradientu pro box omezení se liší od obecné, popsané v předešlé sekci, v kroku projekce. U box omezení 2.13 lze v každé dimenzi problému určit interval, na

kterém je nalezená pozice přípustná. Hledáme tedy funkci, která nám v každém rozměru naprojektuje potencionálně nepřípustnou souřadnici, tak aby platilo: $\{lb \leq x \leq ub, \forall x \in \mathcal{X}\}$, kde lb a ub jsou prvky vektorů $\boldsymbol{LB}$ a $\boldsymbol{UB}$ odpovídající dimenze. Touto funkcí je v nejjednodušším případě funkce median:

$$\mathcal{P}(\boldsymbol{y}) = \text{med}(\boldsymbol{LB}, \boldsymbol{y}, \boldsymbol{UB}), \tag{3.16}$$

myšleno, že *median* je aplikován na každou komponentu zvlášť a projekce $\mathcal{P}(\boldsymbol{y})$ je poskládána pro každý rozměr mediánem „vybrané" přípustné pozice.

3.4 Explicitní řešení

Řešení určené pro parametrické QP, tj.

$$\min_{\boldsymbol{z}} \frac{1}{2} \boldsymbol{z}^T \boldsymbol{H} \boldsymbol{z} + \boldsymbol{q}^T \boldsymbol{F} \boldsymbol{z}, \ \boldsymbol{z} \in \mathbb{R}^n, \tag{3.17}$$

kde $\boldsymbol{q}$ je parametr obsahující, v řízení na referenci, stavy a reference. Ne vždy je výhodné provádět optimalizaci *online*. Ukazuje se, že optimální akční zásah je po částech affiní funkce parametru a je možné tuto funkci vypočítat *offline*. Při řízení se jen nachází výsledky s odpovídajícími akčními zásahy. Jak se ukazuje, samotné hledání je samo o sobě časově náročné a to srovnatelně s rychlými metodami implicitního řešení [10]. Navíc s rostoucí velikostí problému rostou paměťové nároky exponenciálně s počtem omezení [19].

V této práci se zabýváme řízením, kde optimalizace probíhá v každém kroku, se vzorkovací periodou T_S, řízením, využívající tedy *online* řešení. V další kapitole je uveden jeden z možných použitelných algoritmů.

Kapitola 4

Implementace MPC s algoritmem rychlé projekce gradientu

Cílem této práce bylo vytvořit MPC regulátor pro soustavu s krátkou periodou vzorkování. Při požadavku na krátké vzorkovací periody v řádech *ms* je zapotřebí rychlého a efektivního algoritmu. Dostupná literatura [18] ukazuje, že vhodným kandidátem je algoritmus projekce gradientu. Výkon výsledného regulátoru můžeme navýšit zvolením adekvátního programovacího jazyka, rychlých matematických knihoven a optimalizací kódu. Právě vytvoření regulátoru je předmětem této kapitoly.

4.1 Rychlá gradientní metoda

V roce 1983 byl Yuriem Nesterovem publikován článek [20], ve kterém představil modifikovanou gradientní metodu, dnes známou jako optimální nebo rychlá gradientní metoda. Tato sekce úzce sleduje práci [18], kde je rychlá gradientní metoda pro omezený konvexní minimalizační problém přehledně popsána.

Opět mějme QP s box omezením tentokrát ve znění:

$$\min_{\boldsymbol{u}} \frac{1}{2}\boldsymbol{u}^T\boldsymbol{H}\boldsymbol{u} \quad + \quad \boldsymbol{f}^T\boldsymbol{u},\ \boldsymbol{u} \in \mathcal{B}, \tag{4.1}$$

kde $\mathcal{B}$ je množina box omezených přípustných bodů, tedy platí, že $\boldsymbol{LB} \leq \boldsymbol{u} \leq \boldsymbol{UB}$, $\forall \boldsymbol{u} \in \mathcal{B}$. Lipschitzova konstanta L a parametr konvexity μ se vypočte z Hessianu výše uvedeného problému jako $L = \lambda_{\max}(\boldsymbol{H})$ a $\mu = \lambda_{\min}(\boldsymbol{H})$. Postup řešení rychlou gradientní metodou je potom dle 4.1.

Algoritmus 4.1: Rychlá gradientní metoda pro řešení box omezeného konvexního minimalizačního problému [18]

Předpoklad: $\boldsymbol{y}_0 = \boldsymbol{u}_0 \in \mathcal{B}, 0 < \sqrt{\frac{\mu}{L}} \leq \alpha_0 < 1$;

for $i = 1...\mathrm{m}$ **do**

1 $\quad \boldsymbol{u}_{f_i} = \boldsymbol{y}_i - \frac{1}{L}\boldsymbol{g}_k(\boldsymbol{y}_i)$;

2 $\quad \boldsymbol{u}_{i+1} = \mathrm{median}(\boldsymbol{LB}, \boldsymbol{u}_{f_i}, \boldsymbol{UB})$; $\backslash\backslash \boldsymbol{u}_{i+1} \in \mathcal{B}$

$\quad \backslash\backslash \alpha_{i+1} \in (0,1)$ vypočteme z $\alpha_{i+1}^2 = (1-\alpha_i)\alpha_i^2 + \mu\alpha_{i+1}/L$, v implementaci užito:

3 $\quad \alpha_{i+1} = (-\alpha_i^2 L + \mu + \sqrt{4\alpha_i^2 L^2 + (\alpha_i^2 L - \mu)^2})/2L$;

4 $\quad \beta_i = \alpha_i(1-\alpha_i)/(\alpha_i^2 - \alpha_{i+1})$;

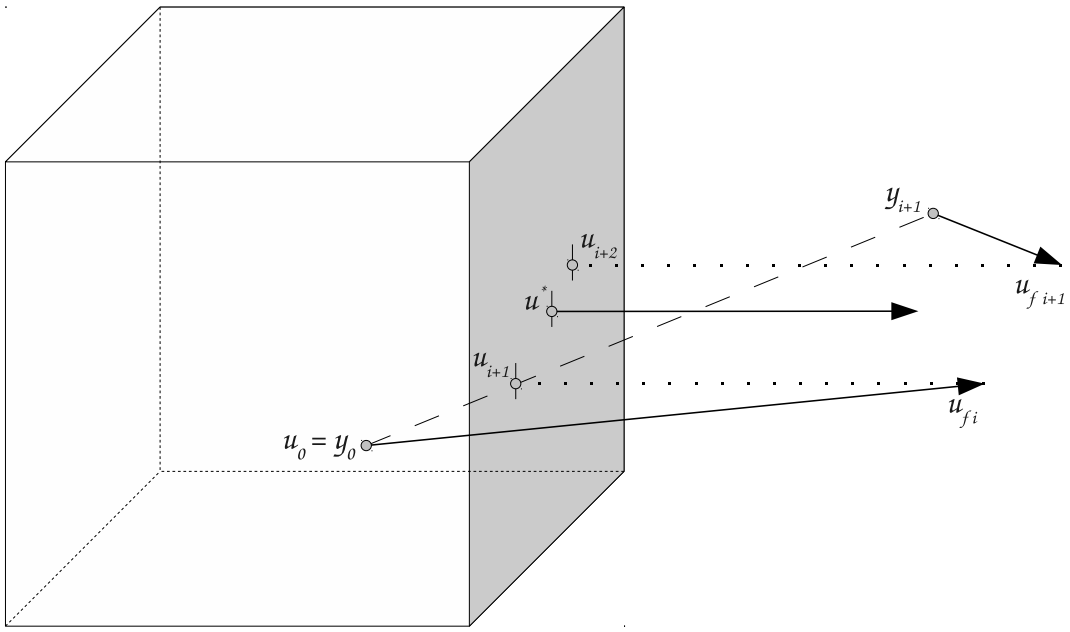
5 $\quad \boldsymbol{y}_{i+1} = \boldsymbol{u}_{i+1} + \beta_i(\boldsymbol{u}_{i+1} - \boldsymbol{u}_i)$;

Na obrázku 4.1 je naznačen postup algoritmu 4.1 v prvních dvou iteracích. V prvním kroce metody se pro gradientní metody běžným postupem vypočítá nová pozice v optimalizačním prostoru. Tato pozice může být nepřípustná, a proto se v následujícím kroce (**2**) provede projekce funkcí *median*, aplikovanou po jednotlivých komponentách vektoru. Takto jsme získali potenciální optimum, v případě, že bude algoritmus ukončen. Pokud ne, vypočítá se koeficient α a z něho následně β. Konstanta β se na konci postupu (v kroce **5**) bude podílet jako váha na výpočtu výchozí pozice pro následující iteraci. Poslední krok je totiž váhový součet dvou předešlých výsledků.

Výpočet činitele β je závislý pouze na α_{i+1} a α_i, v limitním případě na volbě $\alpha_0 \in <\sqrt{\frac{\mu}{L}}, 1)$, dále na minimalizačním problému, respektive na vlastních číslech Hessovi matice. Nic tedy nebrání tomu, aby výpočet posloupnosti $\{\beta_i\}_0^m$ (řádky **3** a **4** algoritmu 4.1) probíhal *offline*, tedy v případě napevno daného počtu iterací.

Algoritmus probíhá v cyklu a je ukončen při dosažení optima. Optima je dosaženo, jsou-li splněny KKT podmínky. Avšak vzhledem k tomu, že v implementaci je kontrola KKT podmínek relativně výpočetně náročná, pro *realtime* aplikace lze použít právě pevného počtu iterací, např. předem spočítaného podle [18].

Podstatným rysem této metody, narozdíl od tradiční gradientní metody, je absence podmínky $f(\boldsymbol{u}_{k+1}) < f(\boldsymbol{u}_k), \forall k \in \mathbb{N}$, a tedy není nezbytně nutné, aby se funkční hodnota minimalizované funkce v každé iteraci zmenšovala. Namísto toho jsou zde podmínky na posloupnosti, podmiňující konvergenci algoritmu. Dvojici posloupností $\{\phi_i(\boldsymbol{u})\}_0^\infty$ a $\{\lambda_i\}_0^\infty$, pro které platí: $\lambda_i \to 0$ a $\phi_i(\boldsymbol{u}) \geq (1-\lambda_i)f(\boldsymbol{u}) + \lambda_i\phi_0(\boldsymbol{u}), \forall i \geq 0, \forall \boldsymbol{u} \in \mathcal{B}$ nazýváme tzv. odhadované posloupnosti.



Obrázek 4.1: Ilustrace rychlé gradientní metody na 3D box (krychle) omezeném optimalizačním problému; šipka značí směr největšího poklesu funkční hodnoty, tečkovaná čára projekci na omezení, čárkovaná čára novou neprojektovanou pozici

Připomeňme, že velikost matice problému $\boldsymbol{H}$ má pro MPC rozměr $m \cdot T_C \times m \cdot T_C$, kde T_C může být velké, a tedy explicitní řešení není možné použít z důvodu neúměrných paměťových nároků.

Dalším ukazatelem složitosti optimalizační úlohy je číslo podmíněnosti

$$\kappa = \frac{L}{\mu}\,. \tag{4.2}$$

Číslo podmíněnosti lze snížit vhodnou transformací proměnných $\boldsymbol{V} = \boldsymbol{P}^{-1}\boldsymbol{U}$, kde $\boldsymbol{P} \in \mathbb{R}^{m \cdot T_C \times m \cdot T_C}$ je diagonální positivně definitní transformační matice. Hessova matice v nových proměnných $\boldsymbol{V}$ bude $\boldsymbol{H}_P = \boldsymbol{P}^T \boldsymbol{H} \boldsymbol{P}$. Lze nalézt optimalní matici $\boldsymbol{P}^*$ vyřešením optimalizace $\min_{\boldsymbol{P}} \frac{\lambda_{max}(\boldsymbol{H}_P)}{\lambda_{min}(\boldsymbol{H}_P)}$ [21].

4.2 Užité explicitní knihovny

Jedním z trendů na poli moderního řízení je řízení velmi rychlých systémů. Navrhujeme-li MPC regulátor pro takové soustavy, je třeba, aby výpočet optimalizace probíhal nejkratší možnou dobu. Protože algebraické výpočty zabírají nejvíce času regulátoru, je nasnadě ušetřit čas právě zde. Již dlouhou dobu existují programátorské knihovny i pro složitější matematické algebraické operace speciálně optimalizované na výkon. Pro implementace optimalizačních algoritmů, jichž se v moderních regulačních soustavách užívá, je vhodné sáhnout po nižší programovacích jazycích (např. C), které jsou méně náročné na režii než programovací jazyky vyšší (např. Java). Při tvorbě MPC byla snaha naplno využít potenciálu knihovny BLAS, a proto je popsána v následující podsekci.

4.2.1 BLAS

Poprvé byl Basic Linear Algebra Subprograms popsán jako balík 38 nízkoúrovňových podprogramů, napsaných v jazyce Fortran, v roce 1979 v [22]. BLAS je standardem pro programové knihovny s na výkon optimalizovanými funkcemi pro práci s vektory a maticemi. Převážná většina vlastních implementací je napsána v jazyce C, nebo Fortran. Konkrétní implementace jsou například ATLAS, IntelMKL, GotoBLAS, NetlibBLAS. Každá implementace BLASu má tři třídy rutin, rozdělené podle náročnosti na FLOPsy (operace s plovoucí desetinou čárkou).

- Level 1 BLAS - Obsahuje vektor-vektor operace (např. kopírování, součet, skalární součin, násobení vektorů konstantou) realizující zobrazení $\boldsymbol{y} \leftarrow \alpha\mathbf{x}+\boldsymbol{y}$. Komplexita funkcí této třídy je $\mathcal{O}(n)$.
- Level 2 BLAS - Obsahuje matice-vektor operace (např. součiny matice-vektor optimalizované pro různé typy matic (trojúhelníkové, symetrické, atd.)) realizující zobrazení $\boldsymbol{y} \leftarrow \alpha\mathbf{A}\boldsymbol{x} + \beta\mathbf{y}$. Komplexita funkcí této třídy je $\mathcal{O}(n^2)$.
- Level 3 BLAS - Obsahuje operacemi mezi maticemi (např. násobení hermitovských, nebo symetrických matic) realizující zobrazaní $\boldsymbol{C} \leftarrow \alpha\mathbf{A}\boldsymbol{B}+\beta\mathbf{C}$. Komplexita funkcí této třídy je $\mathcal{O}(n^3)$.

Všechny úrovně dohramady obsahují všechny možné základní algebraické operace, jejichž správnou posloupností lze realizovat fundamentální funkce lineární algebry. BLASovské knihovny slouží jako základ pro větší softwarové balíky jako je LAPACK [23], implementující komplexnější funkce [24].

4.3 Popis implementace

Implementace, kompilace i simulace byly provedeny pod operačním systémem *MS Windows 7*. Jako vývojové prostředí bylo použito MS Visual C++ a prostředí pro běh simulací Matlab/Simulink. Velkou výhodou volby těchto dvou nástrojů je možnost spárování pro debugování, tj. poté je možné debugovat přímo za běhu simulace.

Vybraný algoritmus (rychlá gradientní metoda) je implementován v podobě S-funkce v prostředí Simulink, jejíž výkonné jádro je napsáno v jazyce C, jako tzv. MEX-soubor. Aby blok S-funkce pracoval správně, je třeba, aby tento MEX-soubor před kompilací obsahoval určité metody a byl zkompilován do dynamické knihovny s příponou ,,*.mexw32*". Jednu z jednodušších struktur souboru, které byla použita jako výchozí, je možné si prohlédnout a popř. editovat v Matlabu příkazem ,,*edit([matlabroot,'/simulink/src/sfuntmpl_basic.c'])*".

4.3.1 Nastavení pracovních nástrojů

Před kompilací musíme v Matlabu nastavit vhodný kompilátor příkazem ,,*mex -setup*". Aby nenastaly komplikace při pozdějším debugování je dobré zvolit právě kompilátor MS Visual C++. Kompilujeme v Matlabu řádkou ,,*mex jmeno_souboru.c knihovna další_knihovna...*" (pro kompilaci s debug značkami přidáme přepínač *-g*). V našem případě se nejčastěji kompilovalo příkazem ,,*mex -g mpc.c libmwblas.lib*" (libmwblas.lib je BLAS knihovna implicitně obsažena v Matlabu).

4.4 Řízený systém

Řízenou soustavou je MIMO systém helikoptéry (Obr. 4.2) od firmy Humusoft, s.r.o., umístěný v laboratoři K26 (více na [25]) na Karlově náměstí DCE, FEL, ČVUT. Model je k podložce připevněná kostra, se dvěma stupni volnosti (elevace a azimut), osazená dvěma stejnosměrnými motorky (hlavní a vedlejší rotor helikoptéry) a IRC senzory pro měření úhlů (elevační a azimutální). Pro více technických informací shlédněte [26]. Soustava má tedy následující vstupy a výstupy:

- U_M vstupní napětí do hlavního motoru
- U_T vstupní napětí do směrového motoru

- ψ elevační úhel
- ϕ azimutální úhel

V Matlabu/Simulinku je možné do rotorů pouštět signály v rozmezí $< -1, 1 >$, a tedy jimi točit na obě strany. Tato sekce čerpá z práce [15], která se problematikou identifikace i řízení systému vrtulníku zabývá detailněji.

Obrázek 4.2: Laboratorní model CE 150 [26]

4.4.1 Stavový model

Stavové matice identifikované v [15] reálné soustavy helikoptéry jsou:

$$\boldsymbol{A} = \begin{bmatrix} -4,0381 & 0 & 0 & 0 & 0 & 0 \\ 0 & -2,8106 & 0 & 0 & 0 & 0 \\ 0,0077 & -0,0070 & -0,4583 & 0 & 0 & 0 \\ 0,0233 & -0,0008 & 0 & -1,1935 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \quad \boldsymbol{B} = \begin{bmatrix} 612 & 0 \\ 0 & -1087 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

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

Pro náš regulátor potřebujeme diskrétní stavový popis systému helikoptéry, aby bylo možné počítat predikce (viz. 2.1), tedy popis ve tvaru:

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{A}_D \boldsymbol{x}_k + \boldsymbol{B}_D \boldsymbol{u}_k \\ \boldsymbol{y}_k &= \boldsymbol{C}_D \boldsymbol{x}_k + \boldsymbol{D}_D \boldsymbol{u}_k. \end{aligned} \tag{4.3}$$

Matice odpovídající diskrétnímu popisu dostaneme diskretizací s periodou $T_S = 0.01s$ podle:

$$\begin{aligned} \boldsymbol{A}_D &= e^{\boldsymbol{A}T_S} & \boldsymbol{B}_D &= e^{\boldsymbol{A}T_S} \int_0^{T_S} e^{\boldsymbol{A}\tau}\,\mathrm{d}\tau. \\ \boldsymbol{C}_D &= \boldsymbol{C} & \boldsymbol{D}_D &= \boldsymbol{D}. \end{aligned}$$

Diskrétní popis, který použijeme pro náš MPC regulátor při simulacích v další sekci pak bude:

$$\boldsymbol{A}_D = \begin{bmatrix} 0,9604 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0,9723 & 0 & 0 & 0 & 0 \\ 0,0001 & -0,0010 & 0,9954 & 0 & 0 & 0 \\ 0,0002 & 0 & 0 & 0,9881 & 0 & 0 \\ 0 & 0 & 0 & 0,0099 & 1 & 0 \\ 0 & 0 & 0,0100 & 0 & 0 & 1 \end{bmatrix} \quad \boldsymbol{B}_D = \begin{bmatrix} 5,9938 & 0 \\ 0 & -10,7200 \\ 0 & 0,0004 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

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

Odpovídající stavové vektory ke stavovému popisu jsou:

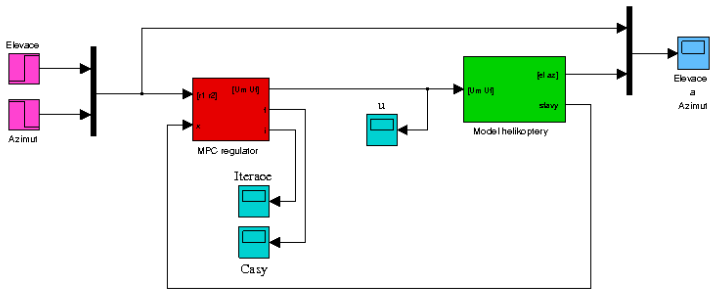
$$\boldsymbol{u} = \begin{bmatrix} U_M & U_T \end{bmatrix}^T$$

$$\boldsymbol{x} = \begin{bmatrix} \omega_M & \omega_T & \omega_V & \omega_H & \psi & \phi \end{bmatrix}^T$$

$$\boldsymbol{y} = \begin{bmatrix} \psi & \phi \end{bmatrix}^T$$

4.5 Simulace s implementovaným regulátorem

Všechny výsledky simulací uvedené v této podkapitole byly získány na Simulinkovém modelu (obr. 4.3), tzn. na modelu helikoptéry. Váhové matice optimalizačního kritéria byly nakonec, po mnoha simulacích, zvoleny $\boldsymbol{Q} = 50\boldsymbol{I}$ a $\boldsymbol{R} = \boldsymbol{I}$. Vzhledem k omezení laboratorního modelu stanoveného výrobcem je omezení regulace nastaveno $\boldsymbol{LB} = [-1, -1]$ a $\boldsymbol{UB} = [1, 1]$.



Obrázek 4.3: Simulinkové simulační schéma s MPC regulátorem

Vliv volby délky horizontu predikce je ilustrován na obr. 4.4. Všimněme si, že pro kratší horizont systém více překmitává, pro ještě kratší by se regulovaná soustava stala nestabilní. Naopak pro nejdelší simulovaný horizont je řízení příliš ,,opatrné", a tedy pomalejší než pro námi vybraný referenční horizont predikce $T_P = 200$. Větší horizont znamená také větší číslo podmíněnosti řešeného QP. Optimální je volit horizont predikce o málo delší než je doba náběhu přechodové charakteristiky systému.

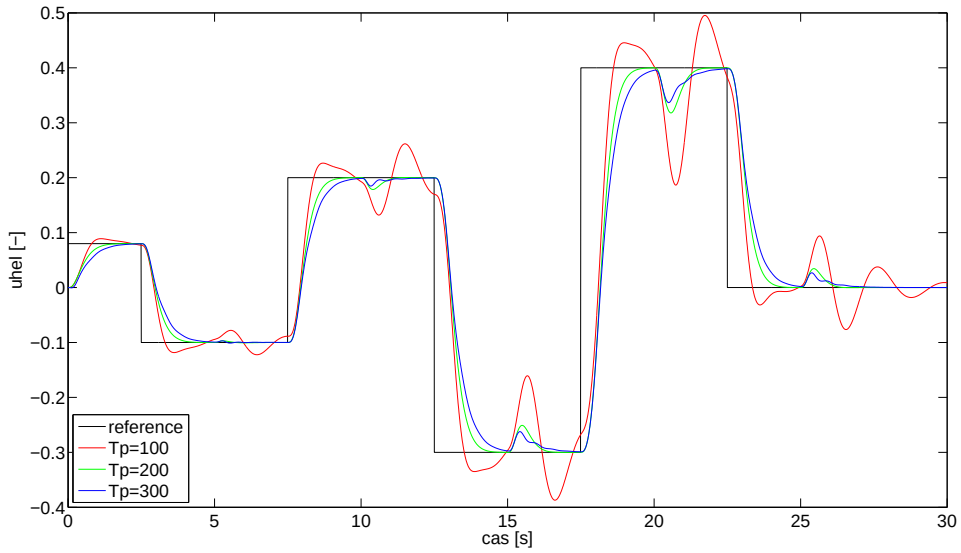
Jinak se na optimalizaci a následném řízení projeví volba horizontu korekce. Platí, že velikost QP je přímo úměrná počtu vstupů řízené soustavy a délce horizontu korekce. Z předešlého tvrzení vyplývá rostoucí složitost řešení QP s rostoucím horizontem korekce při zachování počtu vstupů. Hlavně ale s horizontem roste kvalita řízení, máme-li dostatečný horizont predikce a přesný matematický popis. Horizont korekce se proto volí hlavně v závislosti na dostupném výpočetním výkonu, co nejdelší. Nicméně, jak si můžeme všimnout na obr. 4.5, u vyšších horizontů není zlepšení příliš markantní.

Na obrázcích 4.6a a 4.6b je ukázáno řízení pro parametry regulátoru, námi zvolenými jako vyhovující ($T_C = 3$ a $T_P = 200$), při kterých jsme dosahli dobrých výsledků řízení.

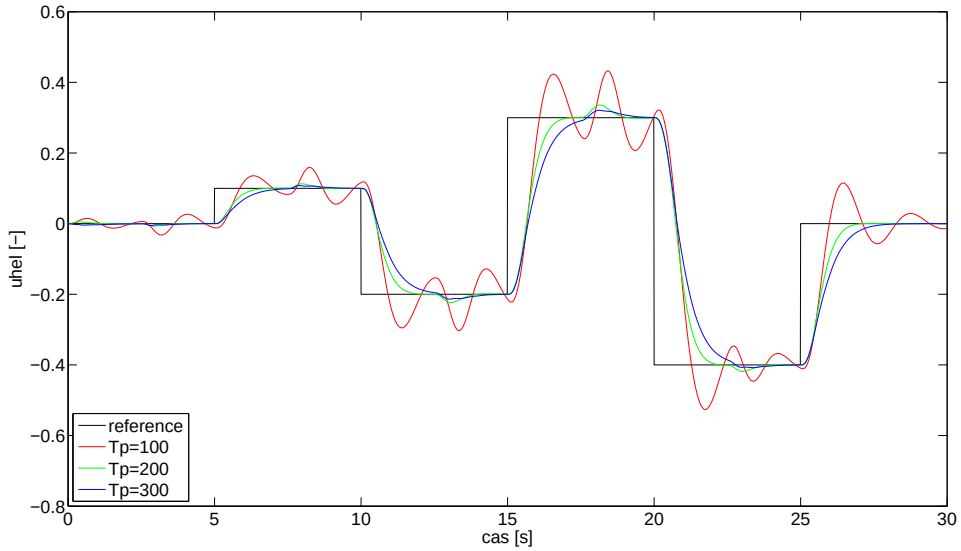
Pro krátké vzorkovací periody je výhodnější volit pevný počet iterací optimalizačního procesu. Jak totiž ukazuje obr. 4.7(c), kontrola na KKT podmínky je schopna ušetřít procesorový čas, ovšem sama kontrola zabírá nezanedbatelnou dobu výpočtů. Stejný počet iterací s a bez kontroly KKT podmínek se v našem případě liší přibližně v 200 sekundách na 1000 iterací ve prospěch pevných iterací (obr. 4.7).

Rychlost optimalizace pro různý počet iterací implementovaného solveru v kontrastu se solverem *quadprog* dostupným v Matlabu můžete vidět na obr. 4.8.

Simulace byly prováděny na počítači s CPU Core2Duo P8400 taktovaném na 2,26GHz. K měření času (např. obr. 4.7) byl využito funkce Windows *QueryPerformanceCounter*.

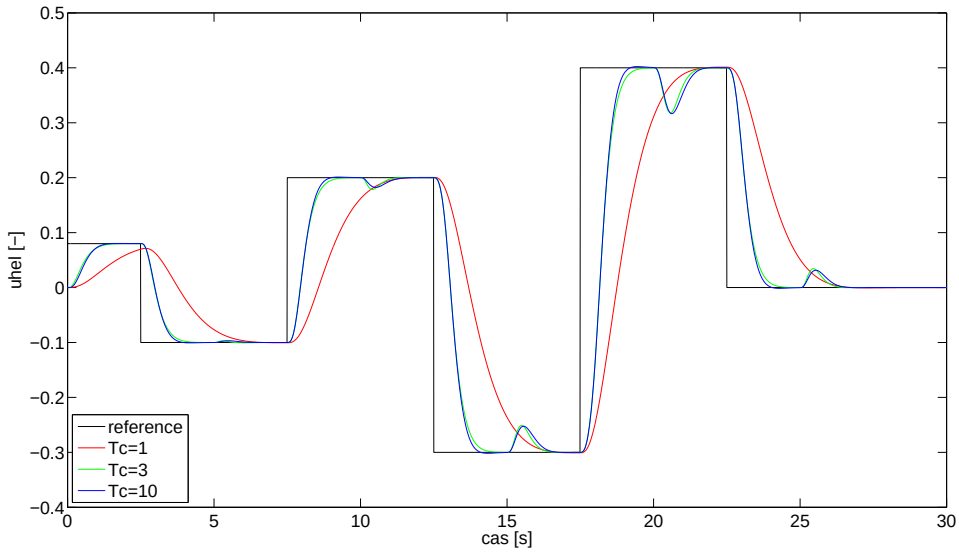


(a) Výstup systému - řízení v elevaci

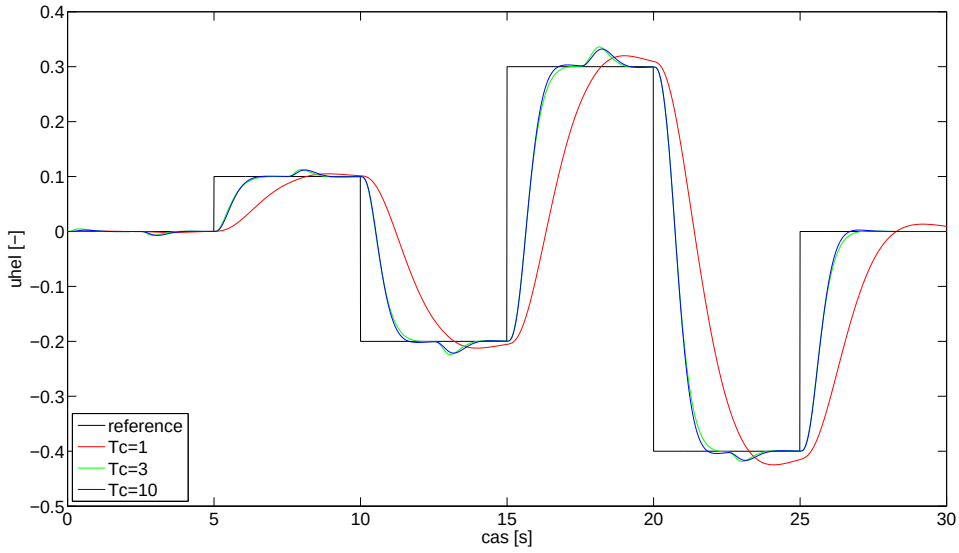


(b) Výstup systému - řízení v azimutu

Obrázek 4.4: Srovnání regulace s horizonty predikce různé délky a konstantní horizont korekce ($T_C = 3$) pro 1000 iterací



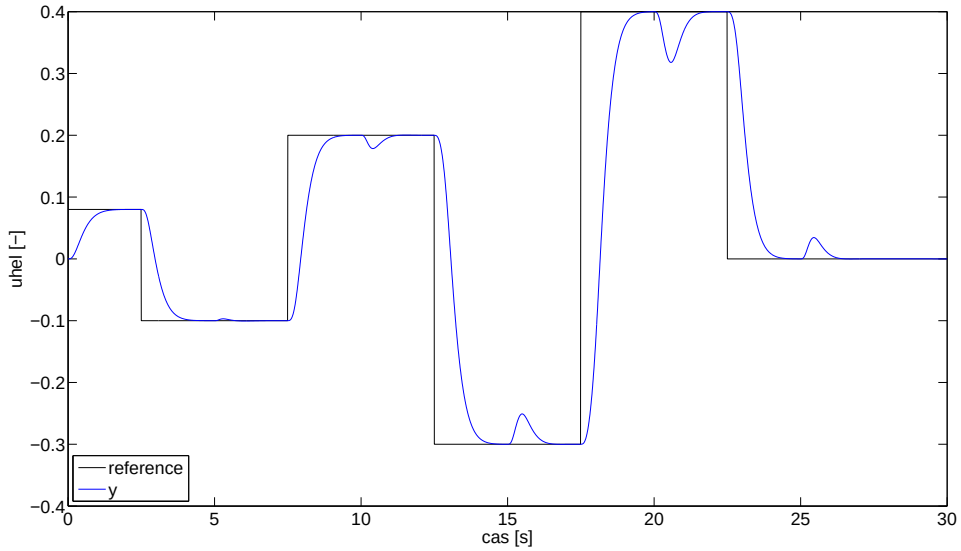
(a) Výstup systému - řízení v elevaci



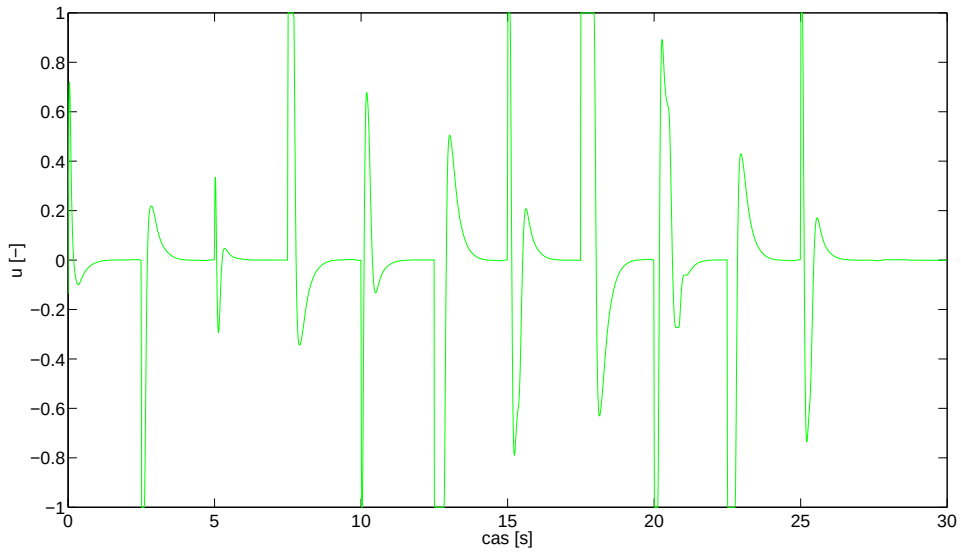
(b) Výstup systému - řízení v azimutu

Obrázek 4.5: Srovnání regulace s horizonty korekce různé délky a konstantní horizont predikce ($T_P = 200$) pro 1000 iterací

[H]

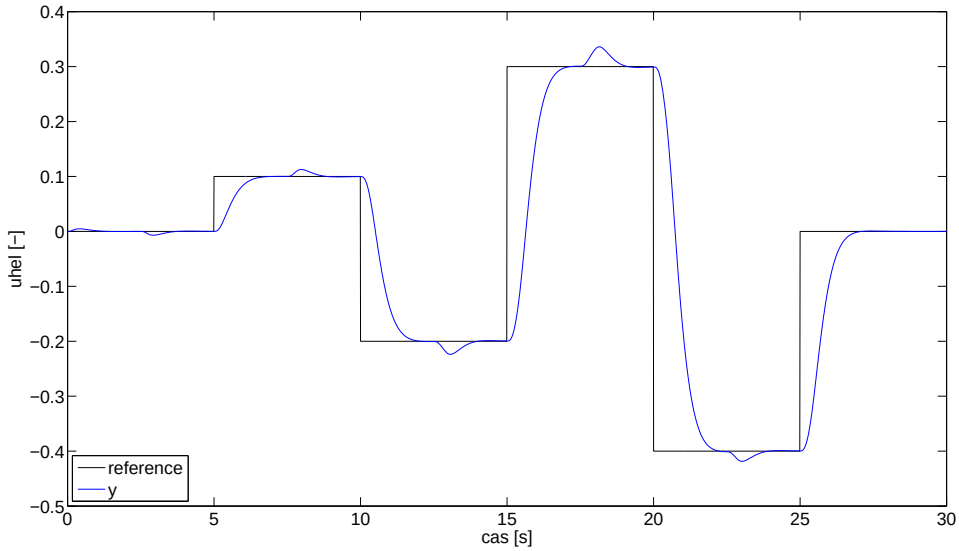


(a) Výstup systému

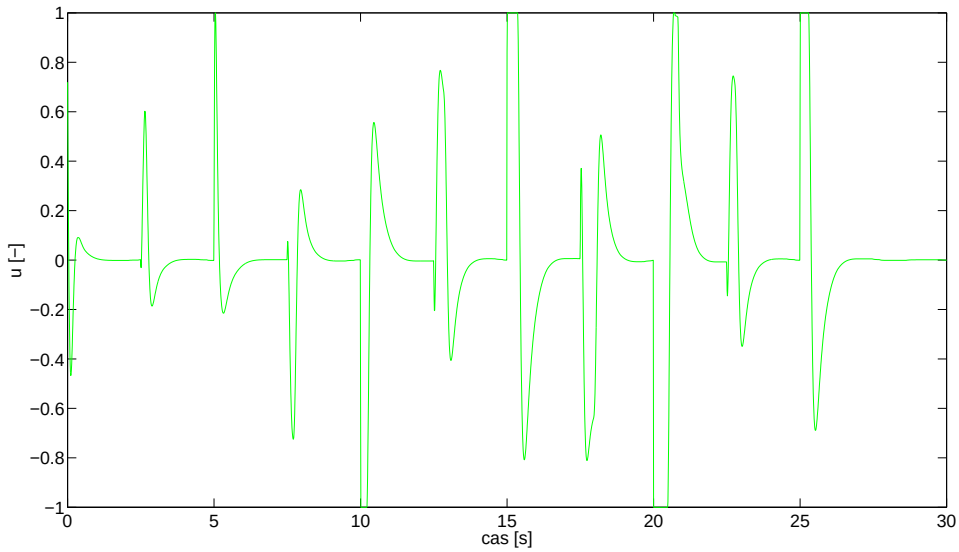


(b) Akční zásah

Obrázek 4.6a: Chování systému v elevaci pro regulaci s $T_C = 3$ a $T_P = 200$ (1000 iterací)

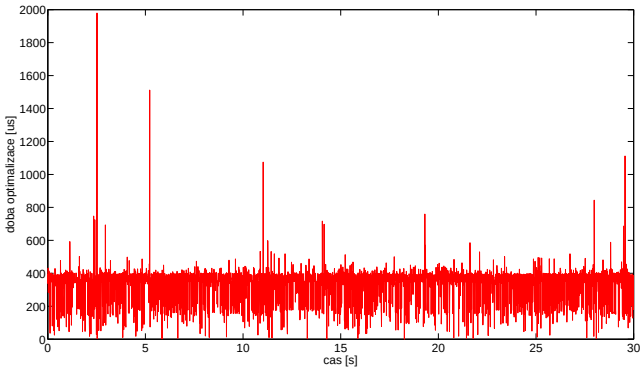


(a) Výstup systému

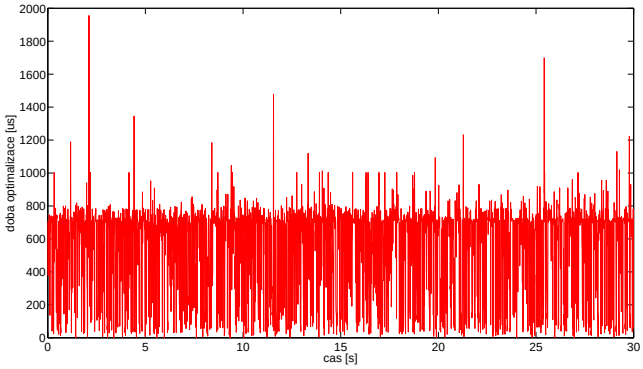


(b) Akční zásah

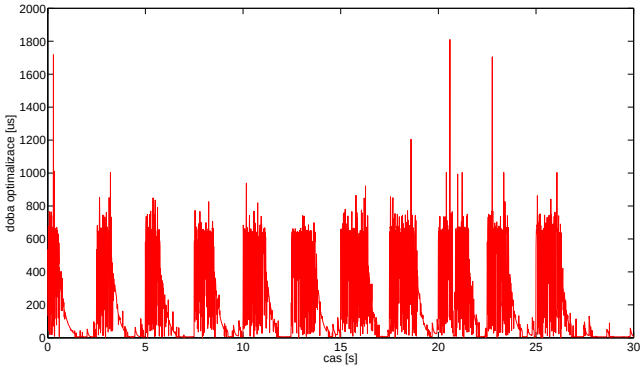
Obrázek 4.6b: Chování systému v azimutu pro regulaci s $T_C = 3$ a $T_P = 200$ (1000 iterací)



(a) Regulace s $T_C = 3$ a pevný počet iterací (1000)

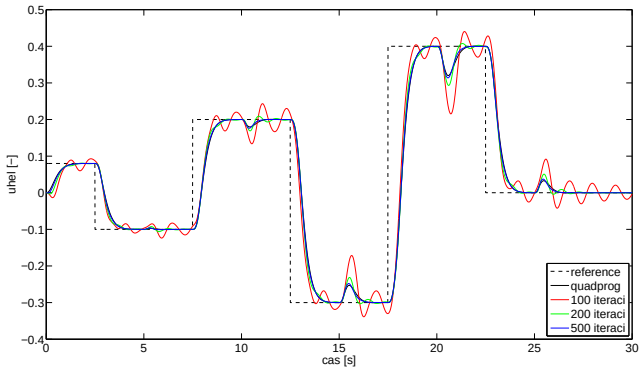


(b) Regulace s $T_C = 10$ a pevný počet iterací (1000)

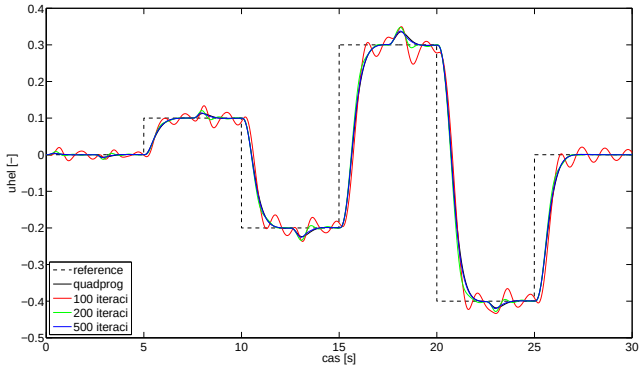


(c) Regulace s $T_C = 3$ a kontrola na KKT podmínky (maximálně 1000 iterací)

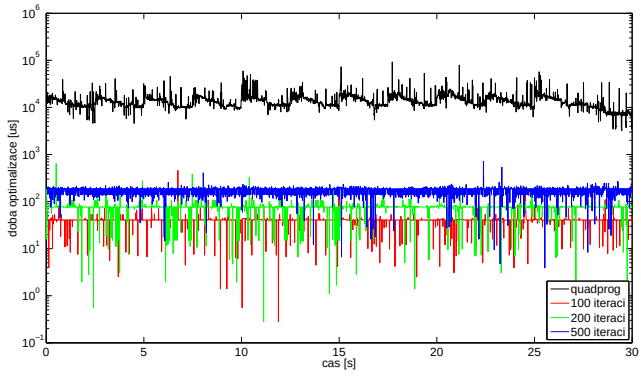
Obrázek 4.7: Srovnání časů trvání optimalizace pro regulaci s $T_P = 200$



(a) Výstup systému - řízení v elevaci



(b) Výstup systému - řízení v azimutu



(c) Časy trvání optimalizace

Obrázek 4.8: Srovnání řízení s optimálním řešením a pro různý počet iterací

Kapitola 5

Závěr

Princip činosti MPC (výpočet predikčních matic, optimalizace kvadratického kritéria) byl včetně jednoduchých odvození popsán v druhé kapitole 2. Dále (kap. 3) jsme se zabývali rozborem řešení optimalizace QP, tj. nejpoužívanějšími numerickými metodami, jimiž lze optimalizace, a to hlavně s omezeními, řešit. Cílem práce bylo implementovat, odsimulovat a odměřit MPC regulátor založený na algoritmu, vhodném i pro rychle vzorkované systémy (kap. 4).

Bohužel nebylo možné odměřit implementovaný regulátor na reálném modelu helikoptéry H1, pro kterou byl navrhován, protože získaný matematický popis (z [15]) již neodpovídal skutečnosti. Cílem práce nebylo reidentifikovat laboratorní model (a nebylo to možné ani z časových důvodů), a tak vlastnosti regulátoru jsou popsány pouze skrze simulace.

Dle získaných dat prezentovaných v sekci 4.5, dosahuje vytvořený MPC regulátor, co do kvality a rychlosti výpočtu akčního zásahu, solidních výsledků. Ukázalo se, že regulátor je schopny kvalitně řídit náš systém (model helikoptéry) při vzorkování s periodou až přibližně $400\mu s$. Regulátor je s minimálními úpravami schopný regulovat různé MIMO soustavy s jednoduchými omezeními a s relativně krátkymi vzorkovacími periodami. Pamatujme však, že je nezbytně nutné mít k dispozici přesný stavový popis.

Literatura

[1] Tondel P., Johansen T.A., Bemporad A.: *An Algorithm for Multi-Parametric Quadratic Programming and Explicit MPC Solutions*
Decision and Control, 2001. Proceedings of the 40th IEEE Conference on

[2] Wang, Y.; Boyd, S.: *Fast Model Predictive Control Using Online Optimization*
The 17-th IFAC World Congress, Jul 2008

[3] Šantin, O., Havlena, V.: *Combined Partial Conjugate Gradient and Gradient Projection solver for MPC*
In: 2011 IEEE Multi-Conference on Systems and Control, 2011 (přijato)

[4] Havlena V., Štecha J.: *Moderní teorie řízení*
Czech Republic, Czech Technical University in Prague, Faculty of Electrical Engineering, 1999

[5] Šantin, O.: *Influence of Model Uncertainty on Constraints Handling in Predictive Control - Diploma Thesis [en]*
Czech Republic, Czech Technical University in Prague, Faculty of Electrical Engineering, 2009

[6] Cagienard, R., Grieder, P., Kerrigan, E. and Morari, M.: *Move Blocking Strategies in Receding Horizon Control*
Decision and Control, 2004. CDC. 43rd IEEE Conference on, Volume: 2

[7] Lee Y.I., Kouvaritakis B., Cannon M.: *Constrained Receding Horizon Predictive Control for Nonlinear Systems*
Proceedings of the 38rd IEEE Conference on Decision and Control, 1999

[8] Rossiter, J.A.: *Model-Based Predictive Control: A Practical Approach*
USA, Florida: CRC Press LCC, 2003

[9] Pekař, J.: *Robust Model Predictive Control - Dissertation*
Czech Republic, Czech Technical University in Prague, Faculty of Electrical Engineering, 2005

[10] Šantin, O., Havlena, V.: *Combined Gradient and Newton Projection Quadratic Programming Solver for MPC*
18th IFAC World Congress in Milano, 2011 (přijato)

[11] Boyd S., Vandenberghe L.: *Model-Based Predictive Control: A Practical Approach*
United Kingdom, Cambridge: Cambridge University Press, 2004

[12] Nocedal J., Wright S.J.: *Numerical Optimization*, 2nd ed.
USA, New York: Springer, 2006.

[13] Štecha J.: *Optimílní rozhodování a řízení*
Czech Republic, Czech Technical University in Prague, Faculty of Electrical Engineering, 1999

[14] Fletcher R., Reeves C.M.: *Function minimization by conjugate gradients*
Computer Journal, 7 (1964)

[15] Hoč, M.: *Helicopter in virtual space - Diploma Thesis [en]*
Czech Republic, Czech Technical University in Prague, Faculty of Electrical Engineering, 2008

[16] Gould N.I.M., Toint P.L.; *A Quadratic Programming Bibliography*
Numerical Analysis Group Internal Report 2000-1, Rutherford Appleton Laboratory, Chilton, Oxfordshire, UK

[17] Bertsekas, D.P.: *On the Goldstein - Levitin - Polyak Gradient ProjectionMethod*
Automatica 21 (1976) 2

[18] Richter, S.; Jones, C.N.; Morari M.: *Real-Time MPC with Input Constraints Using the Fast Gradient Method*
Switzerland, Dept. of Electr. Eng., Swiss Fed. Inst. of Technol. Zurich, 2010

[19] Bemporad, A.; Morari, M.; Dua V.; Pistikopoulos, E. N.: *The explicit linear quadratic regulator for constrained systems*
Automatica 38 (2002) 3-20

[20] Nesterov, YU.E.: *A method of solving a convex programming problem with convergence rate* $\mathcal{O}(1/k^2)$
Switzerland, Dept. of Electr. Eng., Swiss Fed. Inst. of Technol. Zurich, 2010

[21] Boyd S.; Ghanoui L.E.; Feron E.; Balakrishnan V.: *Linear Matrix Inequalities in System & Control Theory*
Society for Industrial and Applied Mathematic, July 1994

[22] Lawson C.L., Hanson R.J., Kincaid D.R., Krogh F.T. *Basic Linear Algebra Subprograms for Fortran Usage*
USA, New York: Springer, 2006.

[23] LAPACK [online]. Poslední aktualizace 2011-04-19 [cit. 2011-05-05].
Netlib.org. Dostupné z WWW: http://www.netlib.org/lapack.

[24] BLAS [online]. Poslední aktualizace 2005-07-25 [cit. 2011-05-05].
Netlib.org. Dostupné z WWW: http://www.netlib.org/blas.

[25] Laboratoř teorie automatického řízení K26 [online]. Poslední aktualizace neznáma [cit. 2011-05-22].
Support.dce.felk.cvut.cz. Dostupné z WWW: http://support.dce.felk.cvut.cz/lab26.

[26] CE 150 Helicopter Model - Educational Models [online]. Poslední aktualizace neznáma [cit. 2011-05-22].
Humusoft.cz. Dostupné z WWW: http://www.humusoft.cz/produkty/models/ce150/.

Příloha A

Obsah CD

K této práci je přiloženo CD s adresáři:

- Dokumenty - Obsahuje zadání, text i prohlášení o samostatné tvorbě bakalářské práce.
- Latex - Obsahuje zdrojový kód textu BP napsaného v jazyce LaTeX.
- Matlab+Simulink - Obsahuje skripty pro Matlab, modely pro Simulink a MEX-soubor (MPC.c - implementaci MPC).