ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ

FAKULTA ELEKTROTECHNICKÁ

KATEDRA ŘÍDICÍ TECHNIKY

# DIPLOMOVÁ PRÁCE

## Adaptivní řízení elektrohydraulických servomechanismů

Vedoucí práce: Ing. Daniel Pachner

leden 2004

Jaroslav Neuhauser

## Čestné prohlášení

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

Nemám závažný důvod proti užití tohoto školního díla ve smyslu § 60 zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne ..... .....

podpis

# Abstrakt

Tato práce přibližuje problematiku adaptivního řízení elektrohydraulických servomechanismů. V krátkosti představíme vlastnosti a použití elektrohydraulických zatěžovacích strojů a také i důvody, proč se v průmyslu v takové míře používají. V teoretické části se zaměříme na problematiku rekurzivní identifikace parametrů systému, sledování časově proměnných parametrů systému. Představíme algoritmus adaptivního regulátoru.

Cílem této práce bylo ověřit zlepšení řízení těchto strojů použitím adaptivního přístupu oproti PID regulaci. Dosavadní PID regulátory často nevyhovují, protože dynamika systému se mění s testováním různých vzorků. V rámci této práce jsme prakticky ověřili vlastnosti použitého adaptivního algoritmu řízení na univerzálním elektrohydraulickém zatěžovacím stroji. Ukážeme typické problémy při praktickém použití adaptivního přístupu.

# Abstract

Electrohydraulic servomechanisms adaptive control will be proposed in this paper. Electrohydraulic testing machines will be briefly introduced and their industrial applications will be discussed. We will present recursive system parameters identification, time-variant parameters tracking and adaptive control algorithm in this paper.

The goal of this work was to verify testing machines control improvement if the adaptive control theory would be used instead of PID controllers. Today's PID controllers are unsufficient, because system dynamics is varying with testing specimens changes. Within this work we have verified features of adaptive control algorithm applied to general purpose electrohydraulic testing machine. Known problems with practical use of adaptive approach will be also discussed.

## Poděkování

Na tomto místě bych chtěl velmi poděkovat vedoucímu práce Ing. Danielu Pachnerovi za pomoc a vedení mé diplomové práce. Děkuji také Ing. Janu Formánkovi, ostatním zaměstnancům společnosti Inova Praha s. r. o. za podporu při tvorbě této diplomové práce a také Ing. H. Lauschmannovi, CSc. za zpřístupnění laboratoře dynamického namáhání. V neposlední řadě bych rád poděkoval své rodině, přítelkyni Míle a dalším, kteří mě po celou dobu studia podporovali.

# Obsah

# Kapitola 1

# Úvod

Tato práce se věnuje adaptivní regulaci elektrohydraulických servomechanismů, jak již napovídá její název. Použití elektrohydraulických servomechanismů je velmi široké. Jednou z velmi zajímavých oblastí, kde je lze potkat, je obor zatěžovacích strojů. Jsou to tedy stroje, které umožňují testovat mechanické vlastnosti různých komponent. Tento obor je na rozmezí poměrně velkého počtu různých oblastí technického vědění. Za všechny jmenujme např. hydraulické systémy, strojírenské obory (mechanika, pružnost, pevnost), silnoproudá elektrotechnika, elektronika, výpočetní technika a v neposlední řadě také měřící a regulační technika.

V průmyslové praxi se stává, že zkušení technici „nevěří“ novým řídicím algoritmům, natož adaptivním. Tento jev nastává v těch situacích, kdy daný algoritmus řízení je poměrně složitý a není zcela průhledný. V případech, kdy se dokonce i prokáže zlepšení regulačního pochodu použitím nových metod řízení oproti klasickým metodám (např. PID regulace), je z velké části vybráno řešení klasickými metodami kvůli své jednoduchosti, průhlednosti, obrovským zkušenostem obsluhy apod. Nové metody řízení se zatím v praxi používají pouze pro případy, kdy již klasické metody řízení selhávají – a to je škoda. Tato práce by měla z části objasnit některé aspekty řízení procesů pomocí nových metod řízení a hlavně prokázat, že nové metody řízení jsou i použitelné v každodenní praxi.

V rámci této práce byl navržen algoritmus adaptivního řízení, který byl také prakticky nasazen při řízení elektrohydraulických zatěžovacích strojů. Při návrhu adaptivního řízení těchto strojů jsme nekladli důraz na výrazné zlepšení regulačního pochodu, ale spíše na jednoduchost použití a univerzalitu nasazení tohoto typu řízení. V nedávné minulosti totiž bylo nutné, aby u některých typů strojů před skoro každou zkouškou obsluha nastavila

konstanty PID regulátoru. To již je v dnešní době značně nevhodné řešení. Navíc obsluha měla jen velmi omezené znalosti teorie řízení. PID regulátor byl v mnoha případěch nastaven „od oka“.

V kapitole 2 ve stručnosti nahlédneme do oboru věnujícímu se zatěžovacím (testovacím, zkušebním) strojům. Nepůjde ani tak o řešení problémů z této oblasti, ale spíše o přehled a popis zatěžovacích strojů a komponent, které obsahují. V následující kapitole 3 ukážeme nejdůležitější pojmy a myšlenky bayesovkého odhadování a adaptivního řízení. Především se zaměříme na identifikaci modelu systému - jako na nejdůležitější aspekt adaptivního řízení. Použitý algoritmus řízení bude představen v kapitole 4. V kapitole 5 se zmíníme o praktických poznatcích, které jsme získali při experimentech na univerzálním zkušebním stroji umístěném v laboratoři dynamické pevnosti na katedře materiálů, FJFI, ČVUT v Praze.

# Kapitola 2

# Zatěžovací stroje

Každá větší strojírenská společnost dnes využívá pro testování svých prototypů i vzorků, vybraných ze sériové výroby, testovací (zkušební, zatěžovací) stroje. Používají se jak v odděleních vývoje, tak i v odděleních kontroly kvality.

Je-li zkušební stroj dobře navržen, může se při jeho správném použití dosáhnout podstatného snížení finanční, ale i časové náročnosti na vývoj dané komponenty. Z tohoto důvodu je použití těchto strojů v dnešní době v podstatě nezbytné, protože vedení společností stále zvyšují tlak na vývojová oddělení, aby urychlili uvedení nových produktů na trh. Většina výrobců se také snaží produkovat výrobky, které vyhovují českým i mezinárodním normám (ČSN, EN, ISO, DIN, a další). Jednou z cest, jak poměrně rychle, bezpečně a s nízkými náklady zajistit otestování výrobku, je právě použití zkušebních strojů.
Myšlenka konstruktérů zkušebních strojů je v podstatě velmi jednoduchá:

> *Vytvořit stroj, pomocí kterého budeme schopni co nejvěrohodněji vytvořit (simulovat) takové podmínky, za kterých by daný vzorek pracoval při reálném zatížení a podmínkách.*

Tyto stroje se samozřejmě také poměrně často používají na vysokých školách (především na katedrách materiálového inženýrství), výzkumných ústavech aj. Zde jsou v největší míře zastoupeny univerzální zkušební stroje (viz obr. 2.1).

V průmyslové praxi se lze setkat především s testovacími stroji vyrobenými přesně na míru resp. přesně dle požadavků zákazníka. To tedy znamená, že lze na tomto stroji testovat pouze vzorky podobných vlastností a rozměrů. Výjimkou také nejsou univerzální testovací stroje, používané v odděleních vývoje.


Obrázek 2.1: Univerzální testovací stroj, Inova FU 160

Mezi hlavní průmyslová odvětví, kde se tyto stroje hojně používají, patří:

- *automobilový průmysl* – zkoušení vlastností karosérií automobilů, testování tlumičů, silenbloků, popř. i celých náprav,
- *letecký průmysl* – např. zjišťování vlivu vibrací, pevnosti a pružnosti křídel za letu,
- *mikroelektronika* – testování odolnosti elektronických součástek proti vibracím a dalším mechanickým vlivům,
- *stavební inženýrství* – testování nových stavebních materiálů, odolnost konstrukcí např. vůči zemětřesení (seismické zkoušky),
- *materiálové inženýrství* – zkoušení mechanických vlastností nových materiálů.

Tento výčet určitě není úplný. Jistě by bylo možné nalézt i další obory, kde se tyto stroje používají. Z hlediska výkonového členu popř. velikosti jmenovité síly lze zatěžovací stroje

rozdělit na:

– **Elektromechanické stroje.** Tyto stroje se používají pro nižší jmenovité síly (5 kN až 250 kN). Mezi jejich výhody patří snadná instalace, nízká pořizovací cena, menší hlučnost aj.

– **Elektrohydraulické stroje.** Pro vyšší jmenovité síly je výhodnější použít hydraulické stroje. Využívá se zde hydraulický princip především z důvodu velkého výkonového zesílení. Toto řešení znamená větší nároky na prostor, instalaci, servis a v neposlední řadě i finance.

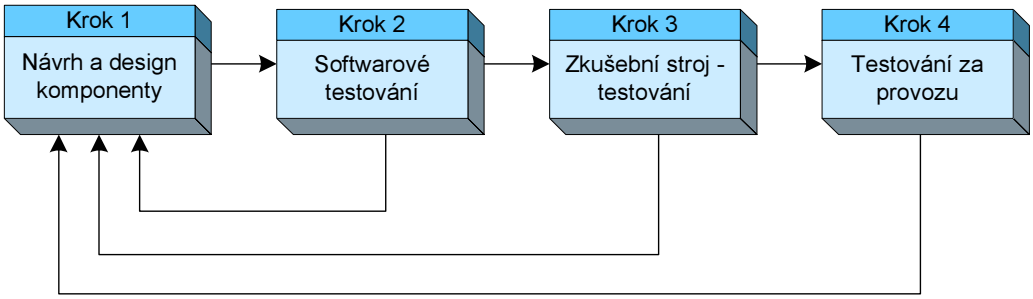
V dalším textu se budeme věnovat pouze elektrohydraulickým (EH) strojům.

## 2.1 Proč a co vlastně testovat?

V dnešní době zažíváme obrovský rozmach výpočetní techniky. Mnoho světových softwarových společností vyvíjí nepřeberné množství programů pro podporu inženýrských výpočtů, vývoje a výroby. Za všechny jmenujme společnosti The MathWorks Inc. (MatLab, MatLab Simulink), Autodesk Inc. (rodina produktů AutoCAD), Cadence Design Systems Inc. (OrCAD) a mnoho dalších. Tato vývojová prostředí nabízejí obrovské množství funkcí i pro simulování navržených komponent. Na první pohled se tedy zdá, že bychom zkušební stroje již nemuseli používat a využívat simulačních možností softwarových vývojových nástrojů. Podíváme-li se na tento problém obecněji, zjistíme, že softwarové testování je pouze druhý krok z mnoha na cestě k výrobě a používání vyvíjené součástky (komponenty), prvním krokem byl samozřejmě návrh (design) komponenty. Na obr. 2.2 je velmi zjednodušené schéma vývoje nějaké jednoduché komponenty.

Přestože se softwarová vývojová prostředí stále zdokonalují, ukazuje se, že krok 3 na obr. 2.2 nelze opomenout. V praxi poměrně často nastává situace, kdy je vyvíjená komponenta natolik složitá, že ji nelze dostatečně komplexně odsimulovat pomocí softwarových nástrojů. Jednoduše proto, že to neumí používaný software nebo proto, že ani neexistuje způsob, jak danou věc vypočítat nebo je to přinejmenším velmi náročné. Nyní nastal okamžik, kdy je použití zkušebních strojů v podstatě nezbytné.

Zkušební stroje je výhodné využívat pro únavové zkoušky a provozní namáhání. V prvním případě jde o zjišťování charakteristických vlastností komponenty až do doby jejího




Obrázek 2.2: Zjednodušené schéma výroby typické komponenty

zničení resp. do doby, kdy již její charakteristiky nevyhovují normám nebo požadavkům konstruktérů. Jedná se tedy o zjišťování parametrů komponenty jako funkcí času při běžné nebo extrémní zátěži – např. se zjišťuje počet pracovních cyklů než se součástka znehodnotí.

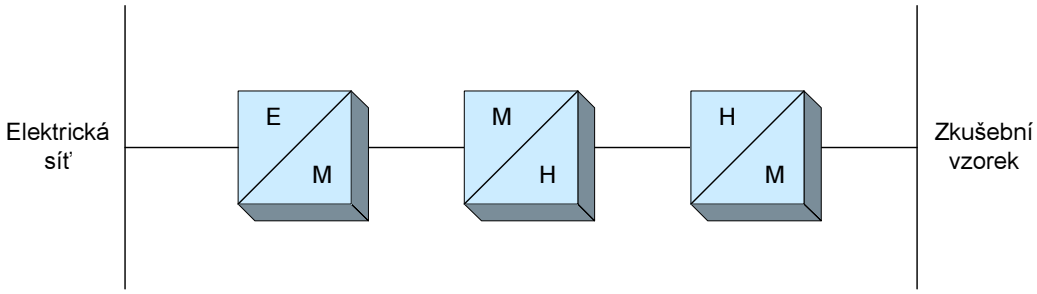
V případě provozního namáhání se simuluje za normálních provozních podmínek. Po konci testu se porovnají vlastnosti součástky před zahájením testu a po jeho ukončení. Tohoto typu testu lze využít například v řízení kvality a jakosti výroby.

## 2.2 Elektrohydraulické stroje

Dynamické zkušební stroje jsou nejčastěji založeny na elektrohydraulickém principu. Pokud se podíváme na elektrohydraulické (EH) stroje z pohledu přenosu energie, lze říci, že se jedná o seskupení hydraulických prvků, které jsou schopny přeměnit mechanickou energii přiváděnou do hydrogenerátoru na tlakovou energii kapaliny, přivést ji na jiné místo a zde ji opět přeměnit na mechanickou práci [6].

Na obr. 2.3 je znázorněno zjednodušené energetické schéma zkušebního EH stroje. Je zde vidět řada převodníků: E/M – elektromechanický (elektromotor), M/H – mechanickohydraulický (čerpadlo), H/M – hydraulickomechanický (servoválec). Vlivem účinností jednotlivých převodníků dochází k energetickým ztrátám. Tato nevýhoda je kompenzována známými výhodami hydraulických mechanismů (např. velmi velké výkonové zesílení). popis a vlastnosti všech těchto převodníků uvedeme dále v kapitolách 2.2.1 - 2.2.6.

Akčním členem každého EH stroje je přímočarý nebo torzní hydraulický motor (hydromotor). Hydromotor by měl vykazovat minimální pasivní odpory kvůli požadované přesnost sledování zatěžovacích průběhů. Tomuto požadavku vyhovují hydromotory s hyd-




Obrázek 2.3: Zjednodušené energetické schéma hydraulického stroje

rostatickým uložením. Vícestupňové servoventily umožňují přesné řízení pohybu pístnice při velkých přesně řízených průtocích v širokém propustném frekvenčním pásmu (až do 400 Hz). Těmto poměrům jsou přizpůsobeny veškeré hydraulické připojovací bloky a multiventily [10].

Při simulacích provozního namáhání se nepříznivě projevují vlastní frekvence mechanických a hydraulických komponent celého zatěžovacího řetězce, které leží v aktuálním zatěžovacím frekvenčním rozsahu. Proto je nutná vysoká tuhost celého stroje. Spoje jejich součástí jsou např. obvykle předepnuty silou větší než je zatěžovací síla, aby k tomuto jevu nedocházelo [10].

Blokové schéma struktury jednoduchého testovacího stroje je na obrázku 2.4. Dále podrobněji popíšeme hlavní části elektrohydraulického zkušebního stroje.


Obrázek 2.4: Zjednodušené schéma hydraulického stroje

### 2.2.1 Hydraulický agregát

Hydraulický agregát je, zjednodušeně řečeno, spojení elektromotoru s hydraulickým čerpadlem (hydrogenerátor). Jeho součástí je také frekvenční měnič pro dobrý rozběh elektromotoru a řídicí elektronika, která hlídá např. teplotu, množství a tlak oleje. Při zapnutí

agregátu se otevírá hydraulický by-pass, tím se ulehčí rozběh elektromotoru. V podstatě je možno říci, že jde o rozběh naprázdno. Hydraulické čerpadlo je sice připojeno, ale tlak oleje je v této fázi zanedbatelný. Po rozběhu se by-pass přeruší a začíná spojitě narůstat tlak v přívodním vedení mezi agregátem a napáječem. Další prvky, které každý hydraulický agregát obsahuje je prachotěsná olejová nádrž, čistič oleje, a hydraulický akumulátor.

Tento agregát musí být schopen vytvořit pracovní tlak 25 – 28 MPa. Jmenovitý průtok bývá v rozmezí 7 – 240 l/min. Obr. 2.5 ukazuje hydraulický agregát s jmenovitým průtokem 165 l/min.

Obrázek 2.5: Hydraulický agregát Inova HU 165

Pro případ poruchy je olejová nádrž hydraulického agregátu vybavena hladinovým spínačem. V případě rychlého poklesu hladiny (např. prasknutí tlakové hadice) se celý stroj bezpečně odstaví.

Při stlačování jakékoliv látky dochází k nárůstu kinetické energie a tudíž i ke zvyšování její teploty. Kdyby teplota látky, v našem případě hydraulického oleje, překročila stanovenou mez, mohlo by dojít k poškození některé části (čerpadla, hydraulické cesty, ...). Proto mezi další ochranné prvky patří i termostat, používaný pro detekci nedostatečného chlazení oleje. Pro chlazení oleje se využívá vodní oběhové čerpadlo s vodním chladičem.

### 2.2.2 Hydraulické cesty

Vedení mezi zdrojem tlaku a napáječem slouží k vedení hlavní části proudu, zpětného proudu a zpětného proudu z řídicího obvodu servoventilu a hydraulických prvků napáječe [7]. Pro výrobu se používají čisté ocelové trubky nebo pryžové tlakové hadice – v případech, kdy dochází k vzájemnému pohybu prvků. Délka vedení V1 (obr. 2.4) je dána vzdáleností mezi zdrojem tlaku a napáječem, může dosahovat i několika desítek metrů. Do tohoto vedení (viz obr. 2.4) se zařazují podle potřeby bezpečnostní prvky (jednosměrné a uzavírací ventily), čističe, případně další prvky. Vedení V2 mezi napáječem na servoválcem slouží k vedení hlavního proudu, řídicího a zpětného proudu obvodu servoventilu, zpětného proudu svodového a pomocného proudu [7]. Pro maximální využití dynamických vlastností celé hydraulické soustavy je nutné dodržet co nejkratší délku vedení V2.

### 2.2.3 Napáječ

Napáječ je hydraulické zařízení, zajišťující maximální využití dynamických vlastností hydraulické soustavy a plnění kombinovaných funkcí. Podle konstrukčního uspořádání plní napáječ v soustavě tyto funkce:

- Snížení tlakových pulzací v hlavním a zpětném vedení. Vyrovnávání tlaku v hlavním vedení způsobené nerovnoměrným odběrem hydraulického proudu. Čištění oleje v hlavním vedení od nečistot, které se mohou dostat do kapaliny potrubí nebo z hadic mezi zdrojem tlaku a napáječem.
- Snížení kolísání tlaku ve vedení řídicího obvodu servoventilu při změnách tlaku v hlavním vedení. Čištění oleje v řídicím odvodu servoventilu.
- Hrazení průtoku v hlavním vedení a snížení tlaku oleje na vstupu servoventilu, propojením hlavního a zpětného vedení při havarijním stavu.
- Snížení tlaku redukčním ventilem v hlavním vedení před servoventilem při plném tlaku v hlavním vedení před napáječem.
- Možnost připojení pomocných hydraulických obvodů bloků systému na hlavní vedení.

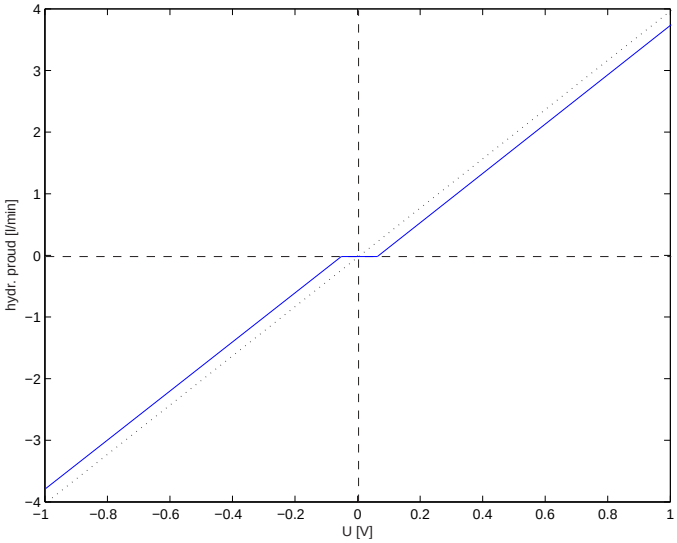
### 2.2.4 Hydraulický válec

Hydraulický válec je v podstatě jiný název pro přímý hydraulický motor. Dnes se většinou používají hydraulické válce s hydrostatickým uložením. Válec s tímto uložením nemá skoro žádné tření. Jedná se o poměrně drahé součásti a jsou velmi náročné na výrobu.

### 2.2.5 Hydraulický servoventil

Tato součást je z pohledu řízení velmi důležitá. Servoventily se skládají z elektromechanického převodníku, převádějící elektrický signál na výchylku kotvy a 2-stupňového hydraulického zesilovače, jehož první stupeň tvoří převodník klapka-tryska, druhý stupeň je tvořen 4-cestným hydraulickým ventilem. Vztah hydraulického výstupu servoventilu k vstupnímu elektrickému signálu zajišťuje hydromechanická zpětná vazba.

Na tyto servoventily jsou kladeny velké nároky na dynamické vlastnosti (rychlost celého hydraulického stroje z velké části závisí právě na použitém servoventilu). Pro lepší řiditelnost se servoventily navrhují s poměrně lineární převodní charakteristikou (na obr. 2.6), nelinearita bývá často způsobena suchým třením (necitlivost kolem nuly). Tuto nelinearitu lze potlačit předbuzením servoventilu na vysoké frekvenci s nulovou střední hodnotou. Tato frekvence by měla být větší, než je propustné pásmo celého stroje, aby neovlivňovala probíhající zkoušky.




Obrázek 2.6: Typická statická převodní charakteristika servoventilu

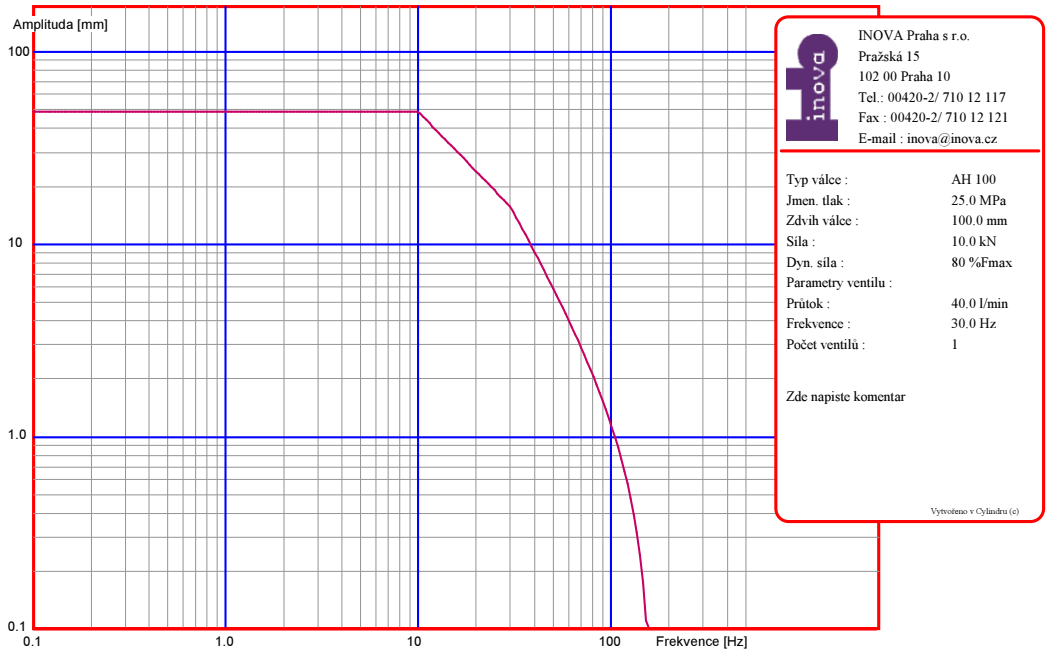
Charakteristické vlastnosti servoventilu jsou jmenovitý průtok (od 10 l/min do 250 l/min), statická převodní charakteristika el. napětí → hydraulický proud a frekvenční charakteristika.

### 2.2.6 Hydraulický servoválec

V podstatě jde o spojení hydraulického válce se servoventilem. Válec se nepohybuje v okamžiku, když se rovná hydraulický proud tekoucí pod píst proudu tekoucímu nad píst. Tento proud je ovládán servoventilem. Na obr. 2.7 je vyobrazen servoválec Inova AH 630 -250 s hydraulickými akumulátory, jež jsou jeho součástí. Použití těchto prvků je nutné pro zabezpečení dostatečného množství oleje v případech, kdy je servoválec značně dynamicky namáhán. Další úlohou, kterou akumulátory plní, je potlačení tlakových špiček v celém hydraulickém obvodu.

Obrázek 2.7: Hydraulický servoválec Inova AH 630 – 250

Na obr. 2.8 je ukázána amplitudová frekvenční charakteristika typického elektrohydraulického servoválce. Z uvedeného obrázku je patrné, že pro plný zdvih válce je propustné pásmo do přibližně 10 Hz.




Obrázek 2.8: Frekvenční charakteristika hydraulického servoválce

# Kapitola 3

# Teorie adaptivního řízení

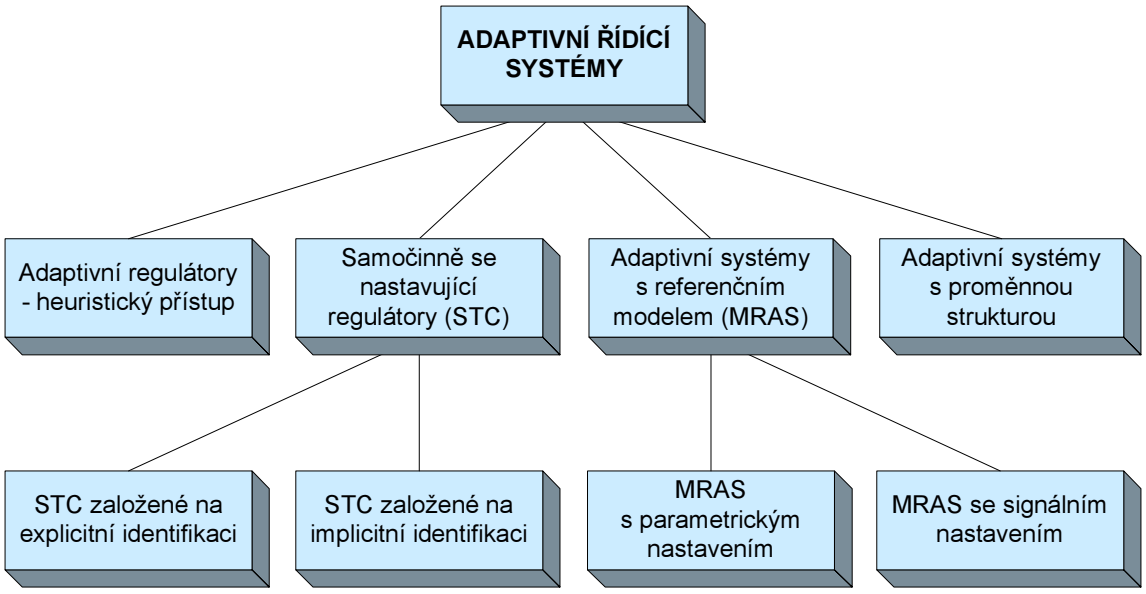
Převážná většina procesů, se kterými se lze setkat v průmyslové praxi, má stochastický charakter. Klasické regulátory s pevně nastavenými parametry pro řízení takových procesů často nevyhovují, neboť při změnách parametrů procesu je řízení neoptimální a dochází ke ztrátám materiálu, energie, snižování životnosti zařízení atd. Změna parametrů procesu je způsobena změnami v provozních režimech, změnami vlastností surovin, paliva, stárnutím zařízení apod., se kterými se pevně nastavené regulátory nemohou vyrovnat.

Jednou z možností zvýšení kvality řízení takových procesů je použití adaptivních řídicích systémů, jejichž nasazování umožnil vývoj moderních číslicových automatizačních prostředků založených na mikroprocesorové technice. Předpokladem je však i vývoj a zdokonalování adaptivních řídicích algoritmů, poznání jejich možností, předností a omezení.

Rozdíl mezi klasickým zpětnovazebním regulátorem a regulátorem adaptivním spočívá v tom, že klasický regulátor využívá principu zpětné vazby k tomu, aby kompenzoval neznámé poruchy a stavy v procesu. Zpětná vazba je pevně nastavená a zesiluje či jinak upravuje regulační odchylku $e = w - y$, kde $w$ je žádaná hodnota regulované veličiny $y$. Tím je tedy určena hodnota vstupního signálu neboli akční veličiny $u$ do soustavy. V každé situaci je způsob zpracování regulační odchylky stejný. Podstatou adaptivního systému je, že se mění způsob zpracování regulační odchylky, tj. adaptuje řídicí zákon na neznámé podmínky a rozšiřuje oblast praktických případů, ve kterých lze dosáhnout kvalitnější regulace. Adaptivitu lze chápat jako zpětnou vazbu vyšší úrovně, která mění parametry regulátoru podle kvality regulačního pochodu.

## 3.1 Klasifikace adaptivních řídicích systémů

Adaptivní systémy založené na heuristickém přístupu, samočinně se nastavující regulátory (STC - Self Tunning Controller) a adaptivní systémy s referenčním modelem (MRAS - Model Reference Adaptive Systems) tvoří v současné době tři základní přístupy k problému adaptivního řízení. Na obr. 3.1 je ukázáno jedno z možných rozdělení adaptivních řídicích systémů [4].




Obrázek 3.1: Klasifikace adaptivních řídicích systémů

### Adaptivní regulátory založené na heuristickém přístupu

U metod využívajících tohoto přístupu se zajišťuje adaptivita přímo vyhodnocováním průběhu regulované veličiny (případně její odchylky) nebo vybraného kritéria kvality regulačního pochodu. Často se v těchto případech využívá algoritmus číslicového PID regulátoru a jako kritérium se obvykle volí míra kmitavosti regulované veličiny nebo její odchylky. Tyto metody nevyžadují identifikaci regulované soustavy. V některých případech není třeba ani sledovat poruchové veličiny, či zavádět zvláštní zkušební signály.

### Adaptivní systémy s referenčním modelem

Významnou vlastností tohoto adaptivního systému je jeho duální charakter, takže je možno jej použít jak pro řízení, tak i pro identifikaci parametrů modelu procesu nebo estimaci stavu systému. Popisu adaptivních systémů s referenčním modelem je v literatuře věnována značná pozornost, přesto je však poměrně málo zveřejněných praktických aplikací. Zdá se, že je to dáno tím, že zpočátku se vyvíjely adaptivní systémy s referenčním modelem pouze pro úzkou třídu systémů (návrhy autopilotů a servomechanismů). Určitým omezením těchto systémů je skutečnost, že jsou vhodné pouze pro deterministické řízení [4].

### Samočinně se nastavující regulátory

Ve dvou výše uvedených přístupech nebyly pro návrh adaptivního regulátoru potřeba detailní znalosti dynamického chování regulované soustavy. Další přístup k adaptivnímu řízení je založen na průběžném odhadování vlastností soustavy a poruch, postupném upřesňování a tím i sledování možných změn. Na základě dosažené znalosti lze vhodnými metodami navrhnout optimální regulátor. Takový regulátor, založený na identifikaci neznámého procesu s následnou syntézou řízení (adaptivní řízení s průběžnou identifikací), je v literatuře označován jako samočinně se nastavující regulátor (STC - Self Tunning Controller).

Prakticky nejlépe využitelných výsledků bylo dosaženo především pro řízení jednorozměrových systémů, pro které byla navržena celá řada numericky stabilních a různě složitých algoritmů. Rozšíření pro řízení vícerozměrových systému nečiní v mnoha případech zásadnější problémy. Tyto algoritmy je pak možno implementovat na řídicí počítače, které jsou vybaveny vhodnou jednotkou pro styk s technologickým prostředím.

Použitím adaptivního řízení s průběžnou identifikací pak podle povahy řízeného procesu lze očekávat tyto vlastnosti:

- Automatické seřízení číslicového regulátoru.
- Zlepšení regulace při přítomnosti nestacionárních poruch.
- Zachycení změn parametrů řízené soustavy, které mohou být způsobeny různými technologickými příčinami, např. provozem zařízení v různých provozních režimech.
- Následné zlepšení regulačních pochodů daného procesu vhodnou změnou parametrů číslicového regulátoru.

Je zřejmé, že v dosažení výše uvedených cílů hraje důležitou roli kromě vlastní optimální strategie řízení rovněž poznání dynamických a statických vlastností daného procesu – jeho identifikace. Z teorie odhadu parametrů je však známo, že odhady parametrů jsou vždy zatíženy jistou neurčitostí (chybou). Tato neurčitost závisí nejen na počtu identifikačních kroků (tj. na počtu vzorkovaných dat) a na volbě struktury matematického modelu řízeného procesu, ale i na průběhu akčních veličin, periodě vzorkování a volbě filtrů akční a regulované veličiny. To znamená, že každá provedená změna akční veličiny, kromě požadovaného řídicího účinku, vybuzuje rovněž řízenou soustavu, a tím vytváří podmínky pro její identifikaci. Jinak řečeno, chceme-li, aby byl řízený systém správně identifikován, je nutné klást na průběh akční veličiny určité podmínky.

Cílem celého snažení je tedy najít takový regulátor, který bude věrně sledovat žádanou hodnotu, potlačovat poruchy působící na tento systém a zároveň nejlépe vybuzovat řízený systém kvůli jeho bezpečné identifikaci. Takové řízení se nazývá duální. Bohužel, optimální duální řízení je pro svoji značnou výpočetní složitost natolik náročné, že pro naprostou většinu reálných případů je nepoužitelné. Přestože bylo duálnímu optimálnímu řízení teoreticky věnováno mimořádné úsilí, nepodařilo se jej ani pomocí různých zjednodušujících aproximací dovést do takového stavu, aby jej bylo možno prakticky použít.

Proto bylo nutné řešit danou úlohu zjednodušeně na základě experimentálních zkušeností a intuice. Toto řešení se nazývá vnucená separace identifikace a řízení (Certainty Equivalence). Princip tohoto zjednodušení spočívá v následujícím postupu [4]:

1. Vektor parametrů $\Theta$ modelu řízeného procesu se pro daný krok řízení považuje za známý, a to roven jeho bodovému odhadu, který je v daném okamžiku k dispozici, tj. $\Theta = \hat{\Theta}(k-1)$.

2. Za tohoto předpokladu se navrhne strategie řízení pro zvolené kritérium kvality řízení a vypočítá se právě potřebný akční zásah $u(k)$.

3. Po získání nového vzorku regulované veličiny $y(k)$ (resp. externí měřené poruchy $v(k)$) a známého akčního zásahu $u(k)$ se provede další identifikační krok pomocí rekurzivního identifikačního algoritmu. To znamená, že nová informace o procesu, kterou nese trojice dat $\{u(k), y(k), v(k)\}$, se použije k aktualizaci odhadu $\hat{\Theta}(k-1)$ a celý postup se opakuje pro nový odhad $\hat{\Theta}(k)$.

Experimentální zkušenosti ukázaly, že převážná část praktických úloh adaptivního řízení s průběžnou identifikací vyhovuje danému zjednodušenému přístupu. Z výše uvedeného postupu vyplývá i vnitřní algoritmická struktura samočinně se nastavujícího regulátoru, schématicky uvedená na obr. 3.2.

Obrázek 3.2: Vnitřní algoritmická struktura samočinně se nastavujícího regulátoru

Vnucenou separací identifikace a řízení se rozpadá vnitřní struktura regulátoru na identifikační a řídicí část, které jsou spojeny pouze přenosem bodových odhadů parametrů $\hat{\Theta}(k)$. Z uvedené struktury je patrné, že základní podmínkou pro dobrou činnost regulátoru je rychle konvergující a spolehlivá identifikace.

## 3.2 Identifikace procesu

V adaptivním řízení je úloha identifikace přinejmenším stejně důležitá jako role syntézy regulátoru. Identifikace pro adaptivní řízení má ovšem své specifické vlastnosti a požadavky, které vedou k tomu, že se v převážné míře odhadují parametry regresního (ARX, ARMAX, aj.) modelu při použití metody nejmenších čtverců. Zabýváme-li se identifikací určitého systému, měli bychom postupovat podle následujícího schématu [4]:

1. Příprava identifikačního experimentu. Vybíráme nejvhodnější vstupní (budící) signál jako kompromis mezi teoreticky optimálním vybuzením a tím, co lze aplikovat z hlediska technologie. Průběh identifikačního experimentu lze sledovat, lze jej přerušit a upravit vstupní signál.

2. Data naměřená při experimentu lze uchovat a následně zpracovávat různými metodami a různými modely, filtrovat apod.

3. Získané parametry modelu lze verifikovat na jiných vzorcích dat.

4. Identifikační experiment lze opakovat, eventuálně již s využitím znalosti získaných předchozími experimenty.

5. Lze testovat či verifikovat podmínky nestrannosti odhadů.

Naproti tomu, při identifikaci pro adaptivní řízení musíme vycházet z následujících podmínek:

1. Data (vstupy) jsou generovány zpětnovazebním regulátorem.

2. Cílem regulátoru je kompenzovat poruchy, stabilizovat proces. To jsou okolnosti, které zhoršují možnosti identifikace parametrů nedostatečným vybuzováním systému.

3. Identifikační proces u adaptivního řízení trvá velmi dlouho (nekonečně dlouho). Proto lze jen stěží předpokládat neměnnost odhadovaných parametrů. Jsou nezbytné metody odhadování časově proměnných parametrů.

4. Identifikace musí být funkční za různých pracovních podmínek systému (v období relativního stacionárního stavu, při poruchách či přechodech mezi různými stavy).

5. Strukturu identifikovaného modelu (řád) obvykle nelze v průběhu identifikace měnit.

6. Identifikační algoritmus musí být numericky stabilní a dostatečně rychlý.

### 3.2.1 Algoritmy identifikace

Z hlediska identifikace parametrů regresního modelu není třeba rozlišovat parametry $a_i$ nebo $b_i$, ale je možné pracovat s vektorem neznámých parametrů $\Theta$ a vektorem dat – regresorem – $\phi$. Pro účely řízení samočinně se nastavujících regulátorů nás zajímají pouze ty způsoby experimentální identifikace, které můžeme realizovat v reálném čase. Pro odhadování parametrů v reálném čase jsou nejvhodnější průběžné (rekurzivní) algoritmy, kde odhady v kroku $k$ se získají tak, že novými daty aktualizujeme staré odhady $\Theta(k-1)$ v čase $k-1$. Nejznámější jsou následující rekurzivní algoritmy:

- pro odhady parametrů modelu ARX:
  1. Rekurzivní metoda nejmenších čtverců.
  2. Rekurzivní metoda instrumentální proměnné.
  3. Metoda stochastické aproximace.
- pro odhady parametrů modelu ARMAX:
  1. Rekurzivní metoda rozšířených nejmenších čtverců.
  2. Rekurzivní metoda maximální věrohodnosti.

Pro odhadování parametrů modelu ARX se v největší míře používá metoda nejmenších čtverců, popsaná např. Peterkou [8].

### 3.2.2 ARX a ARMAX model

Podívejme se nyní podrobněji na vlastnosti ARX a ARMAX modelu, srovnáme výhody a nevýhody každého z nich.

### ARX model

ARX (AutoRegressive model with eXternal input) model je popsán diferenční rovnicí ve tvaru:

$$y(k)+a_1y(k-1)+\ldots+a_{n_a}y(t-n_a)=b_0u(t)+b_1u(t-1)+\ldots+b_{n_b}u(t-n_b)+e(t). \quad (3.1)$$

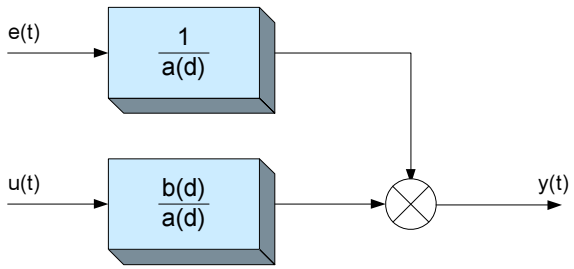
Vyjádříme-li rovnici (3.1) ve tvaru polynomu, dostaneme:

$$a(d)y(t)=b(d)u(t)+e(t), \quad (3.2)$$

kde $a(d)=1+a_1(d)+\ldots+a_{n_a}d^{n_a}$ a $b(d)=b_0+b_1(d)+\ldots+b_{n_b}d^{n_b}$. Rovnici (3.2) lze také upravit do tvaru:

$$y(t)=\frac{b(d)}{a(d)}u(t)+\frac{1}{a(d)}e(t), \quad (3.3)$$

který nám umožní popsat další vlastnosti. Z předchozí rovnice (3.3) je vidět největší omezení tohoto modelu. Toto omezení spočívá v tom, že zvolíme-li jmenovatel deterministické části, bude tím jednoznačně určen filtr šumu. Na obr. 3.3 je blokové vyjádření rovnice (3.3). V případě, kdy přenosy $e(t) \rightarrow y(t)$ a $u(t) \rightarrow y(t)$ mají podobné frekvenční vlastnosti, bude




Obrázek 3.3: ARX model

se ARX model snažit odhadnout deterministickou část. Problém nastává v okamžiku, kdy dynamika šumu je řádově rychlejší než deterministická, pak ARX model odhaduje více přenos šumu, což ale není cílem identifikace.

Predikovaná střední hodnota výstupu $\hat{y}(t|t-1)$ je lineární funkcí měřených dat, proto lze k odhadu parametrů použít lineární regresi. Největší výhodou ARX modelu je jeho jednoduchost. V některých reálných aplikací postačuje, proto je i velmi používaný.

**ARMAX model**

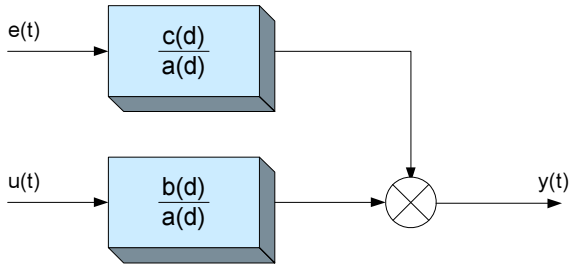
ARMAX (AutoRegressive Moving Average model with eXternal input) model umožňuje nezávislé modelování stochastické složky. Polynomiální popis ARMAX modelu:

$$a(d)y(t) = b(d)u(t) + c(d)e(t), \tag{3.4}$$

kde $c(d) = 1 + c_1(d) + \ldots + c_{n_c}d^{n_c}$. Rovnici (3.4) lze vyjádřit ve tvaru:

$$y(t) = \frac{b(d)}{a(d)}u(t) + \frac{c(d)}{a(d)}e(t). \tag{3.5}$$

Na obr. 3.4 je vyobrazení ARMAX modelu.




Obrázek 3.4: ARMAX model

ARMAX model umožňuje nezávisle volit vlastnosti deterministické části i stochastické složky. Predikovaná střední hodnota výstupu $\hat{y}(t|t-1)$ není lineární funkcí měřitelných dat a není tedy možné rekurzivně odhadovat parametry ARMAX modelu použitím bayesovského přístupu. Pro známé hodnoty parametrů $c_i$ je možné přesně odhadovat parametry $a_i$ a $b_i$.

Tento model se většinou používá v případech, kdy nevyhovuje ARX model z hlediska možnosti nezávislého modelování přenosu $e(t) \rightarrow y(t)$ a $u(t) \rightarrow y(t)$. Pro výpočet predikované střední hodnoty výstupu lze použít pseudolineární regrese. Jedná se pouze o přibližnou metodu, ale vzhledem k její výpočetní jednoduchosti v případě rekurzivního výpočtu je často používaná.

### 3.2.3 Jednorázový odhad parametrů metodou nejmenších čtverců

Metoda nejmenších čtverců patří mezi metody regresní analýzy, které jsou vhodné pro vyšetřování statických i dynamických vztahů mezi veličinami měřenými na vyšetřovaném

objektu. Uvažujme jednorozměrový stochastický proces popsaný ARX modelem ve vektorové formě

$$y(k) = \Theta^T(k)\phi(k) + e(k), \tag{3.6}$$

kde $\Theta^T(k) = [a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_n]$ a
$\phi(k) = [-y(k-1), -y(k-2), \ldots, -y(k-n), u(k-1), u(k-2), \ldots, u(k-n)]$.

Generování výstupní veličiny $y(k)$ potom můžeme v jednotlivých časových okamžicích vyjádřit maticovou rovnicí

$$\mathbf{y} = \mathbf{Z\Theta} + \mathbf{e}, \tag{3.7}$$

kde matice $\mathbf{Z}$ o rozměru $(N-n; 2n)$ a vektory $y$, $e$ o rozměru (N-n) mají tvar:

$$\mathbf{y}^T = \begin{bmatrix} y(n+1) & y(n+2) & \ldots & y(N) \end{bmatrix} \tag{3.8}$$

$$\mathbf{e}^T = \begin{bmatrix} e(n+1) & e(n+2) & \ldots & e(N) \end{bmatrix} \tag{3.9}$$

$$\mathbf{Z} = \begin{bmatrix} -y(n) & \ldots & -y(1) & u(n-1) & \ldots & u(1) \\ -y(n+1) & \ldots & -y(2) & u(n) & \ldots & u(2) \\ \vdots & & & & & \vdots \\ -y(N-1) & \ldots & -y(N-n) & u(N-2) & \ldots & u(N-n) \end{bmatrix}, \tag{3.10}$$

N je počet souborů naměřených vstupních a výstupních dat. Z rovnice (3.7) vyjádříme chybu

$$\mathbf{e} = \mathbf{y} - \mathbf{Z\Theta} \tag{3.11}$$

a zavedeme kritérium

$$J = \mathbf{e}^T\mathbf{e} = (\mathbf{y} - \mathbf{Z\Theta})^T(\mathbf{y} - \mathbf{Z\Theta}), \tag{3.12}$$

jehož minimum získáme, když derivaci (3.12) podle vektoru parametrů $\Theta$ položíme rovnu 0, tj.

$$\frac{\partial J}{\partial \Theta} \mid_{\Theta=\hat{\Theta}} = 0. \tag{3.13}$$

Řešením rovnice (3.13) získáme základní maticový tvar pro odhad parametrů modelu ARX metodou nejmenších čtverců ve tvaru

$$\hat{\mathbf{\Theta}} = (\mathbf{Z}^T\mathbf{Z})^{-1}\mathbf{Z}^T\mathbf{y}. \tag{3.14}$$

Vztah (3.14) slouží pro jednorázový výpočet odhadů parametrů modelu procesu použitím $N$ souborů naměřených dat. Výpočetně je tato metoda poměrně náročná z hlediska velikosti paměti počítače, kde je třeba uchovávat všechny naměřené údaje.

### 3.2.4 Průběžná identifikace metodou nejmenších čtverců

K výpočtu odhadů parametrů modelu procesu pro samočinně se nastavující regulátor není možné použít vztah (3.14), ale pouze jeho rekurzivní verzi, vhodnou pro identifikaci v reálném čase. Při této modifikaci se používají nově naměřené hodnoty pouze pro opravu (korekci, update) původních odhadů, čímž klesá výpočetní složitost identifikačních algoritmů, a tedy i náročnost na použitou výpočetní techniku. Rekurzivní algoritmy umožňují sledovat změny vlastností (parametrů) procesu v reálném čase, a proto jsou základem samočinně se nastavujících regulátorů.

### 3.2.5 Sledování časově proměnných parametrů - zapomínání

Máme-li identifikovat systém s časově proměnnými parametry, bylo by dobré, umět potlačit relativně „staré informace“ o řízeném systému. Tato stará změřená data již nemusí vyjadřovat přesnou informaci o stavu a vlastnostech systému v aktuálním čase. Již se mohly změnit parametry systému, změnil se pracovní bod systému apod.

Technika, jak se vyrovnat s tímto jevem, se nazývá zapomínání. Dá se říci, že je to technika, která umožní sledovat časově proměnné parametry identifikovaného systému tím, že zneváží stará data, která již nenesou zcela správnou informaci o aktuálním stavu systému. Jinak řečeno, ze souboru naměřených dat se zapomínání snaží zvětšit váhu novějším datům a zmenšit váhu starším.

Pokud chceme identifikovat parametry časově proměnného systému, musíme náš rekurzivní algoritmus identifikace rozšířit o tzv. časový krok. Časový krok je v tomto případě jiný termín pro zapomínání. V každém kroku algoritmu se tedy provádí datový (filtrační) a časový krok. V datovém kroku algoritmu se podle změřených dat zaktualizují odhady parametrů. Kovarianční matice se jakoby ztenčí ve směru příchozích dat, tím se sníží rozptyl parametrů, příslušející danému směru. V následujícím časovém kroku se podle typu zapomínání „nafoukne“ resp. zvětší kovarianční matice, např. pro exponenciální zapomínání se jednoduše celá kovarianční matice vynásobí koeficientem $\varphi^{-1}$, kde $\varphi \in (0; 1\rangle$. Koeficient zapomínání $\varphi$ se v nějaké formě vyskytuje v některých typech zapomínání.

### Lineární zapomínání

Prvním typem zapomínání je lineární zapomínání. Zachycuje časové změny (drift) parametrů explicitním modelem:

$$\theta(t+1) = \theta(t) + \nu(t), \qquad \nu(t) \sim \mathcal{N}\Big(0, \sigma_e^2 \mathbf{V}(t)\Big), \tag{3.15}$$

$$\mathbf{P}(t+1|t) = \mathbf{P}(t|t) + \mathbf{V}(t), \tag{3.16}$$

kde $\mathbf{V}(t)$ obsahuje apriorní informaci o možných směrech změn parametrů. V podstatě se jedná o modelování náhodnou procházkou. Lineární zapomínání neřeší problém sledování časově proměnného rozptylu šumu.

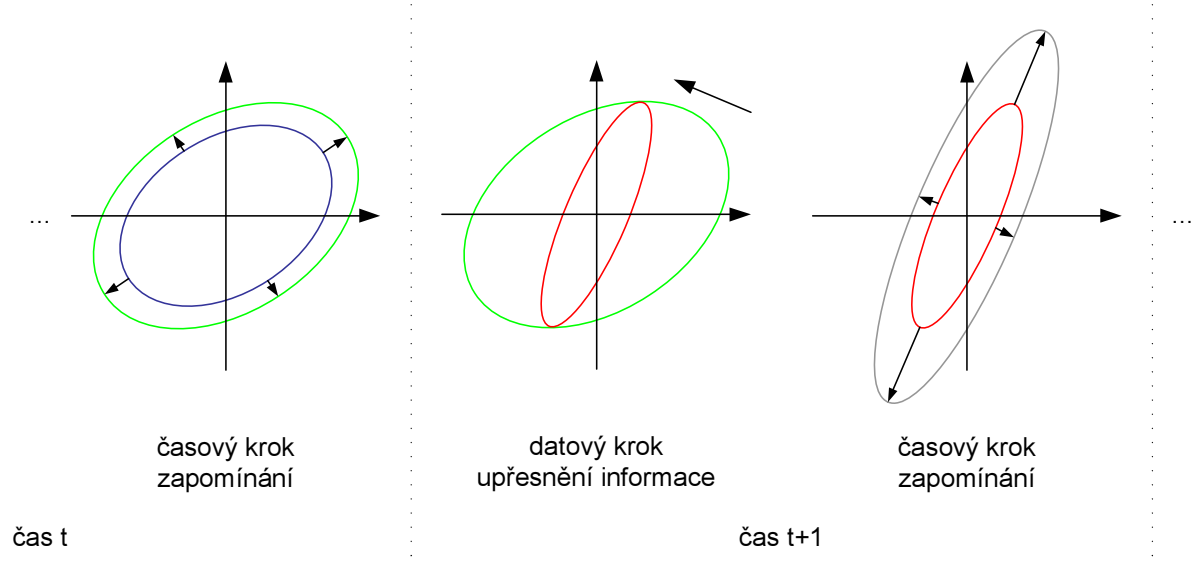
### Exponenciální zapomínání

Exponenciální zapomínání zvyšuje neurčitost parametrů přímo:

$$\hat{\theta}(t+1|t) = \hat{\theta}(t|t), \tag{3.17}$$

$$\mathbf{P}(t+1|t) = \frac{1}{\varphi}\,\mathbf{P}(t|t), \tag{3.18}$$

kde $\varphi$ je faktor zapomínání. Na obr. 3.5 je znázorněn časový vývoj kovarianční matice parametrů. V datovém kroku se upřesní odhadované parametry. V následujícím časovém kroku




Obrázek 3.5: Tvarování kovarianční matice při exp. zapomínání

se tyto upřesněné parametry uměle zneváží (zvýší se neurčitost) tím, že se kovarianční matice vynásobí koeficientem $\varphi^{-1}$. Na první pohled to vypadá tak, že se nic zásadního pro sledování časově proměnných parametrů nestalo. Opak je pravdou. Tím, že uměle zvětšíme kovarianční matici (nejistotu) parametrů, může se rychleji posunout střední hodnota parametrů při změně vlastností systému.

Nyní jsme přiblížili techniku exponenciálního zapomínání, kde hraje velkou roli koeficient zapomínání $\varphi$. Jaký je vztah mezi koeficientem zapomínání a počtem vzorků, které ještě mají vliv na výpočet? Pro jeho určení nám může pomoci následující vztah 3.19:

$$t_{ef} = \frac{1}{1-\varphi}, \tag{3.19}$$

kde $t_{ef}$ je efektivní počet vzorků použitých pro výpočet. V tabulce 3.1 jsou vypočtené koeficienty zapomínání pro některé vybrané počty efektivních vzorků. Volba koeficientu zapomínání je ovlivněna tím, jak rychle se mění parametry řízeného systému. Jako rozumné se jeví zvolit $\varphi$ z intervalu $\langle 0.95\,;1\rangle$.

| $\varphi$ | 1 | 0.999 | 0.99 | 0.95 | 0.9 |
|---|---|---|---|---|---|
| $t_{ef}$ | $\infty$ | 1000 | 100 | 20 | 10 |

Tabulka 3.1: Počet efektivních vzorků pro různé koef. zapomínání

Při použití tohoto typu zapomínání může dojít k tzv. ztrátě informace – Covariance Wind-up. Tento jev je způsoben tím, že měřená data se stávají lineárně závislými, vybuzují systém v jednom směru, resp. ve velmi podobných směrech. To nastává především při identifikaci v uzavřené smyčce (viz. kapitola 3.2.6). Dochází k tomu, že se upřesňuje hodnota pouze některých parametrů a naopak ztrácíme informaci o střední hodnotě ostatních parametrů.

Vlastní čísla kovarianční matice nám dávají jistou informaci o tvaru kovarianční matice. Jsou-li absolutní hodnoty všech vlastních čísel řádově stejně velké je vše v pořádku. Začne-li se výrazně zvětšovat absolutní hodnota některého z vlastních čísel, je to známka „zvětšování“ kovarianční matice v tomto směru. O střední hodnotě parametrů, odpovídajících tomuto směru, v tomto okamžiku nelze tvrdit skoro nic.

V kapitole 3.2.6, věnující se identifikaci v uzavřené smyčce, jsou uvedeny některé možnosti, jak se vyrovnat s jevem Covariance Wind-up. Jedním z možných řešení, které je také v tomto výčtu, může být i směrové omezení zapomínání.

#### Směrově omezené zapomínání

Směrově omezené zapomínání [9] je v podstatě založeno na zvyšování neurčitosti pouze ve směru, ve kterém byla data ve filtračním kroku modifikována - tj. ve směru regresoru. Tím dosáhneme toho, že nebude docházet k jevu Covariance Wind-up, protože ve směrech, kde není v datovém kroku získána nová informace, se kovarianční matice nezvětší.

#### Regularizované zapomínání

Dalším typem zapomínání je regularizované zapomínání. Při tomto typu dochází k vytváření jakési směsi informací z referenčního modelu a informací získaných z naměřených dat. Informace získaná z naměřených dat je tedy z části nahrazována informací z „bezpečného“ modelu. Tím se lze bránit tomu, aby se identifikovaný model příliš vzdálil od modelu apriorního. Tento druh zapomínání je dobré použít tam, kde není systém dostatečně vybuzován a popř. také tam, kde máme k dispozici poměrně přesný model řízeného systému a nepožadujeme velkou míru adaptivity řízení.

Další možností, jak rozšířit regularizované zapomínání, je regularizovat pouze zapomínání rozptylu odhadovaných parametrů, ale dovolit větší možnost aktualizace střední hodnoty parametrů např. použitím směrově omezeného zapomínání. Výsledkem pak bude pružnější sledování střední hodnoty parametrů, ale omezení nechtěného růstu neurčitosti (rozptylu) parametrů.

### 3.2.6 Identifikace v uzavřené smyčce

Při identifikaci v uzavřené smyčce dochází k problémům s identifikací systému. V části 3.1 o samočinně se nastavujících regulátorech jsme se zmínili o druhé roli adaptivního regulátoru, vedle co nejpřesnějšího sledování reference ještě musí dostatečně vybudit systém – pro jeho dobrou identifikaci. Právě toto je důvod, proč je identifikace v uzavřené smyčce obtížná.

Představme si případ, kdy řídíme systém na který působí velmi malé vnější poruchy a referenční veličina bude konstantní funkce. Po uzavření zpětné vazby a odeznění přechodového děje dojde k tomu, že systém přestane být dostatečně vybuzen. Změna akční veličiny se bude blížit k nule. V tomto případě již měřená data neobsahují dostatek informace pro dobrou identifikaci systému – stávají se lineárně závislá.

V případě jednorázové identifikace tento stav nevadí. Provede se další experiment s lepším průběhem referenční veličiny pro identifikaci. Co ale udělat v případě rekurzivní identifikace? Aby k tomuto stavu při identifikaci nedocházelo, musíme pozměnit identifikační algoritmus. Možností řešení je několik [3]:

- použití exponenciálního zapomínání s proměnným faktorem zapomínání;
  - stopa kovarianční matice udržována konstantní,
  - omezeno největší vlastní číslo matice $\mathbf{P}(t+1,t)$,
- udržování konstantního informačního obsahu kovarianční matice;
  - časově proměnný koeficient exponenciálního zapomínání,
  - změna parametrů → vzroste chyba predikce → sníží se koeficient zapomínání,
- omezení lineárního/exponenciálního zapomínání;
  - v predikčním kroku algoritmu zapomínat pouze informaci ve směru, ve kterém byla ve filtračním kroku daty modifikována.

## 3.3 Syntéza zákona řízení

V okamžiku, kdy máme spolehlivě odhadnuté parametry modelu systému, můžeme přistoupit k syntéze zákona řízení resp. k návrhu posloupnosti akčních veličin $u$. Tento návrh lze provést mnoha způsoby. Můžeme použít např. teorii PID regulace, lineárně kvadratické (LQ) přístupy nebo i metody prediktivního řízení navrhující optimální vstupní posloupnost řízení. Tomuto poslednímu přístupu je v dnešní době věnována značná pozornost i v průmyslu, jelikož je zde možné do regulátoru velmi jednoduše zahrnout nejrůznější omezení.

V dalším textu se zaměříme na oblast lineárně kvadratického (LQ) řízení, tedy při syntéze řízení budeme předpokládat lineární systém a kvadratické kritérium optimalizace.

### 3.3.1 Kritérium optimalizace

Typickými příklady optimalizačních kritérií jsou například [2]:

$$J_1 = \sum_{t=0}^{N-1} 1 \,, \quad \mathbf{x}(N) = \mathbf{x}_f, \tag{3.20}$$

jehož optimalizací dosáhneme přechodu z libovolného počátečního stavu $\mathbf{x}_0$ do žádaného koncového stavu $\mathbf{x}_f$ v minimálním čase (počtu kroků), kritérium

$$J_2 = \sum_{t=0}^{N-1} |\mathbf{u}(t)| \ , \quad \mathbf{x}(N) = \mathbf{x}_f, \tag{3.21}$$

jehož optimalizací dosáhneme přechodu z libovolného počátečního stavu $\mathbf{x}_0$ do žádaného koncového stavu $\mathbf{x}_f$ s minimální „spotřebou paliva“, která je úměrná amplitudě řídicího vstupu, nebo kritérium

$$J_3 = \sum_{t=0}^{N-1} \mathbf{u}^T(t)\mathbf{R}\mathbf{u}(t) \ , \quad \mathbf{x}(N) = \mathbf{x}_f, \tag{3.22}$$

jehož optimalizací dosáhneme přechodu z libovolného počátečního stavu $\mathbf{x}_0$ do žádaného koncového stavu $\mathbf{x}_f$ s minimální vynaloženou energií, která je úměrná kvadratické funkci amplitudy řídicího vstupu.

Podobně, chceme-li, aby výstup řízeného systému sledoval danou referenční trajektorii $r(t)$ a dosáhnout přitom kompromisu mezi vynaloženou energií a kvalitou sledování referenční hodnoty, je vhodné použít kvadratické kritérium ve tvaru

$$\begin{aligned} J &= \frac{1}{2}\mathbf{x}^T(N)\mathbf{Q}(N)\mathbf{x}(N) + \\ &+ \frac{1}{2}\sum_{t=0}^{N-1}\Big\{(\mathbf{y}(t)-\mathbf{r}(t))^T\mathbf{Q}_y(t)(\mathbf{y}(t)-\mathbf{r}(t)) + \mathbf{u}^T(t)\mathbf{R}(t)\mathbf{u}(t)\Big\}, \end{aligned} \tag{3.23}$$

kde posloupnost matic $\mathbf{R}$(t) váží vynaloženou energii řízení a posloupnost matic $\mathbf{Q}_y(t)$ váží odchylky výstupu od referenční hodnoty.

V teorii řízení je velmi často používané kvadratické kritérium optimalizace. Vede k tomu řada důvodů [2]:

- úlohy kvadratické optimalizace jsou poměrně snadno řešitelné,
- řadu optimalizačních kritérií lze v okolí jejich minima aproximovat kvadratickou funkcí,
- pro linearizovanou soustavu v okolí daného pracovního bodu je optimální regulátor rovněž lineární a lze ho realizovat stavovou zpětnou vazbou,
- pro nekonečný horizont optimalizace lze najít časově invariantní regulátor, který stabilizuje (za známých podmínek) řízenou soustavu,
- tento časově invariantní regulátor má příznivé vlastnosti z hlediska robustnosti (zvláště ve spojitém případě).

### 3.3.2 Princip dynamického programování

Dynamické programování je velmi účinný nástroj k numerickému řešení problémů optimalizace. Používá se při řešení nejrůznějších problémů od problémů optimalizace k problémům umělé inteligence.

Základ této metody tvoří princip optimality, vyjadřující princip postupné analýzy problému. Princip optimality i celá metoda dynamického programování jsou spojené s pracemi amerického matematika R. Bellmana. Vedle principu optimality je v metodě dynamického programování velmi důležitá myšlenka vnoření konkrétního optimalizačního problému do třídy analogických problémů. Tento princip se nazývá princip invariantního vnoření.

**Princip optimality** tvrdí, že optimální posloupnost rozhodování v mnohastupňovém rozhodovacím procesu má tu vlastnost, že ať jsou jakékoliv vnitřní stavy procesu a předchozí rozhodování, zbylá rozhodování musí tvořit optimální posloupnost vycházející ze stavu, který je výsledkem předchozích rozhodování. [5]

Jinými slovy, v každém rozhodovacím stupni musíme volit optimální rozhodnutí, přičemž vycházíme ze stavu, ve kterém se právě nacházíme. Stav, ve kterém právě jsme, je výsledkem předchozích rozhodování a tento stav již nemůžeme ovlivnit, ale naše další rozhodování musí být optimální.

**Princip invariantního vnoření** znamená to, že náš optimalizační problém vnoříme do celé třídy analogických problémů. Tuto celou třídu problémů vyřešíme a tím také jaksi mimoděk vyřešíme náš jediný problém [5].

Označme optimální hodnotu ztrátové funkce pro stav $\mathbf{x}(t)$ v čase $t$ jako

$$V^*(\mathbf{x}(t), t) = \min_{u(.)} V(\mathbf{x}(t), \mathbf{u}_t^{N-1}, t) \ , \tag{3.24}$$

pak princip optimality lze pro kritérium

$$J = \frac{1}{2}\mathbf{x}^T(N)\mathbf{Q}(N)\mathbf{x}(N) + \frac{1}{2}\sum_{t=0}^{N-1}\left\{\mathbf{x}^T(t)\mathbf{Q}(t)\mathbf{x}(t) + \mathbf{u}^T(t)\mathbf{R}(t)\mathbf{u}(t)\right\} \tag{3.25}$$

formálně zapsat rovnicí (3.26) [2]

$$V^*(\mathbf{x}(t), t) = \min_{u(t)}\Big\{\frac{1}{2}\begin{bmatrix}\mathbf{x}^T(t) & \mathbf{u}^T(t)\end{bmatrix}\begin{bmatrix}\mathbf{Q}(t) & \mathbf{S}(t)\\ \mathbf{S}^T(t) & \mathbf{R}(t)\end{bmatrix}\begin{bmatrix}\mathbf{x}(t)\\ \mathbf{u}(t)\end{bmatrix} + V^*(\mathbf{x}(t+1), t+1)\Big\}, \tag{3.26}$$

kde $\mathbf{S}(t)$ váží součiny stavu a vstupu a v čase $t = N$ platí koncová podmínka

$$V^*(\mathbf{x}(N), N) = \frac{1}{2}\mathbf{x}^T(N)\mathbf{Q}(N)\mathbf{x}(N). \tag{3.27}$$

Rovnice (3.26) popisuje jeden krok dynamického programování. Optimální řízení $\mathbf{u}^*(t)$ lze získat v každém kroku minimalizací kvadratické funkce (3.26). Tuto minimalizaci kvadratické formy lze provést např. doplněním na úplný čtverec. Optimální hodnota ztrátové funkce je kvadratickou funkcí stavu

$$V^*(\mathbf{x}(t), t) = \frac{1}{2}\mathbf{x}^T(t)\mathbf{P}(N)\mathbf{x}(t), \tag{3.28}$$

kde matice $\mathbf{P}(t)$ je určena vztahem

$$\begin{aligned} \mathbf{P}(t) &= \mathbf{A}^T(t)\mathbf{P}(t+1)\mathbf{A} + \mathbf{Q}(t) - \\ &- (\mathbf{S}(t) + \mathbf{A}^T(t)\mathbf{P}(t+1)\mathbf{B}(t))(\mathbf{R}(t) + \mathbf{B}^T(t)\mathbf{P}(t)\mathbf{B}(t))^{-1} \\ &\quad (\mathbf{S}^T(t) + \mathbf{B}^T(t)\mathbf{P}(t+1)\mathbf{A}(t)). \end{aligned} \tag{3.29}$$

Tím jsme dospěli k maticové diferenční rovnici, která se často nazývá Riccatiho diferenční rovnice. Pak lineární časově proměnný systém má optimální řízení minimalizující kritérium (3.25) tvar časově proměnné stavové zpětné vazby

$$\mathbf{u}^*(t) = -\mathbf{K}(t)\mathbf{x}(t), \tag{3.30}$$

kde optimální (Kalmanovo) zesílení

$$\mathbf{K}(t) = \left[\mathbf{R}(t) + \mathbf{B}^T(t)\mathbf{P}(t+1)\mathbf{B}(t)\right]^{-1}\mathbf{B}^T(t)\mathbf{P}(t+1)\mathbf{A}(t) \tag{3.31}$$

určíme na základě matice $\mathbf{P}(t)$, které získáme řešením Riccatiho rovnice (3.29) s koncovou podmínkou $\mathbf{P}(N) = \mathbf{Q}(N)$. Tím jsme získali stavovou realizaci LQ regulátoru.

Položme si otázku, zda je možné časově proměnný kvadraticky optimální regulátor nahradit regulátorem suboptimálním, ale časově invariantním. Pozorováním průběhu řešení Riccatiho rovnice zjistíme, že v mnoha případech se řešení $\mathbf{P}(t)$ pro rostoucí $N - t$ blíží jisté ustálené hodnotě $\mathbf{P}$. Tomuto ustálenému řešení Riccatiho rovnice samozřejmě odpovídá ustálená hodnota Kalmanova zesílení $\mathbf{K}$ a také matice uzavřené regulační smyčky je konstantní [2].

# Kapitola 4

# Algoritmus řízení

Při experimentech jsme pro řízení elektrohydraulického zatěžovacího stroje použili dva druhy řídicího algoritmu. Prvním z nich byl adaptivní regulátor s výpočtem Riccatiho rovnice rozloženým v čase – IST (Iterations Spread in Time). Druhým z nich byla jakási kombinace metod IST a klouzavého horizontu. Tento algoritmus se také nazývá anticipativní regulátor. Pokud zvolíme horizont anticipace nulový, přejde anticipativní regulátor v první typ adaptivního regulátoru (IST). Podívejme se na tyto dva přístupy trochu podrobněji [4].

## 4.1 Iterace rozložené v čase

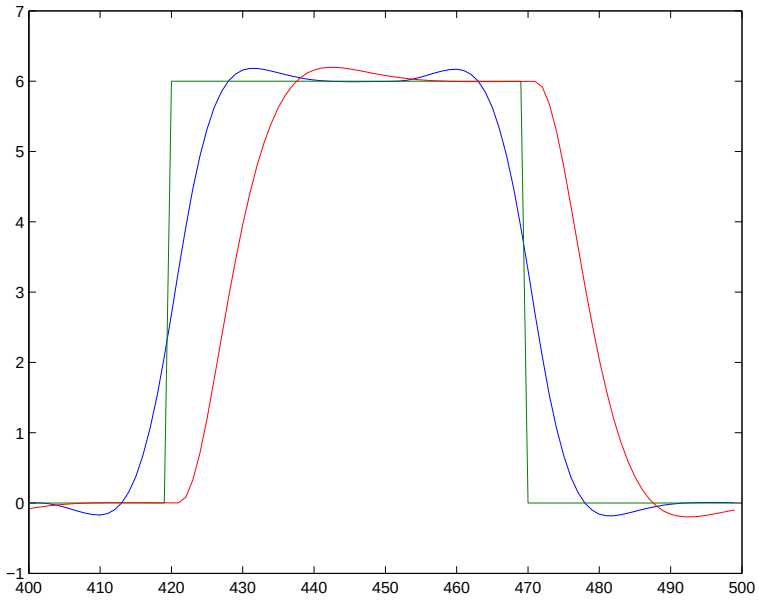
Pokud chceme dosáhnout řízení odpovídající nekonečnému horizontu a přitom spočítat výpočet on-line, je třeba iterace Riccatiho rovnice rozložit v čase (Iterations Spread in Time). To znamená, že v každé periodě řízení provedeme jen pevný počet iterací, a to z předchozího dosaženého stavu, nikoli z počáteční podmínky. Tím dosáhneme toho, že každá perioda řízení zvětšuje horizont kritéria o zvolený počet. Po určité době dosáhneme toho, že se řídicí zákon přiblíží ustálenému řešení. To ovšem za předpokladu, že jsme ve všech periodách použili pro výpočet stejné parametry modelu. V opačném případě se dopustíme systematické chyby, která je ovšem únosná pro pomalu a spojitě se měnící parametry modelu. Zkušenost ukazuje, že se tato strategie, nazývaná iterace rozložené v čase (IST), dobře osvědčuje i v případech, kdy se využije jen jedné iterace v periodě řízení, což reprezentuje nejkratší možnou dobu výpočtu.

## 4.2 Metoda klouzavého horizontu

Metoda klouzavého horizontu minimalizuje kritérium řízení s konečným horizontem. Délka tohoto horizontu musí být taková, aby se odpovídající iterace Riccatiho rovnice uskutečnila ve vymezeném čase. Tento postup lze doplnit testem konvergence zákona řízení. V případě, že se již málo mění lze výpočet přerušit před provedením všech iterací. Pokud zvolíme jen velmi málo iterací (krátký horizont optimalizace), bude počáteční podmínka řešení Riccatiho rovnice hrát významnou roli jak pro stabilitu, tak i pro kvalitu řízení. Je tedy třeba, ji volit co nejblíže konečnému řešení.

## 4.3 Anticipativní regulátor

Pro zlepšení schopnosti regulátoru sledovat žádanou hodnou lze použít přístup zvaný anticipace. Zjednodušeně řečeno, jde o to, že regulátoru dáme informaci o tom, jak bude vypadat žádaná hodnota (reference) na několik kroků dopředu. Přibude nám tedy další parametr regulátoru, který pojmenujeme horizont anticipace. Jedním z omezení pro použití tohoto přístupu je znalost průběhu žádané hodnoty na intervalu $\langle t, t + \tau \rangle$, kde $t$ je aktuální čas a $\tau$ je délka horizontu optimalizace. V podstatě se dá říci, že jde o jakousi




Obrázek 4.1: Vliv předvídání reference při jinak stejném nastavení.

kombinaci výpočtu optimálního řízení na klouzavém horizontu a iteracemi rozloženými v čase. Dalo by se říci, že si tento přístup vybírá od každé metody to lepší.

Jako zatím velmi těžko překonatelné omezení anticipativního regulátoru se ukázala jeho výpočetní náročnost. Dimenze matic potřebných pro výpočet se zvyšuje lineárně s horizontem anticipace, tj. s dopředu známou délkou posloupnosti žádané hodnoty. Zkušenosti ukázaly, že je dobré volit horizont anticipace cca 5 – 20 kroků. Na obr. 4.1 je ukázán rozdíl mezi anticipativním (modře) a klasickým regulátorem (červeně), který nepočítá s dopředu známou žádanou hodnotou. Z obrázku by se na první pohled mohlo zdát, že jde pouze o posunutí (zpoždění) žádané hodnoty vůči skutečné. Ve skutečnosti je to tak, že regulátor na daném horizontu optimalizace spočítá optimální řízení. Proto se začíná měnit skutečná hodnota již před skokem žádané hodnoty.

## 4.4 Navržený algoritmus

Kompletní popis (literární program) navrženého algoritmu je v příloze A - Popis algoritmu řízení. I přesto zde uvedeme krátký popis jeho vlastností a principů, které využívá.

### 4.4.1 Identifikační část

Algoritmus je navržen jako mnoharozměrový – pro řízení MIMO systémů. Je zde tedy implementována rekurzivní identifikace mnoharozměrového lineárního ARX modelu metodou nejmenších čtverců ve tvaru:

$$\sum_{n=0}^{\text{order}} \left\{ \sum_{k=1}^{\text{ny}} a_{kn}^{j} y_k(t-n) + \sum_{h=1}^{\text{nu}} b_{hn}^{j} u_h(t-n) \right\} + d^j + e^j(t), \tag{4.1}$$

kde *order* je řád odhadovaného modelu, *ny* je počet výstupů modelu a *nu* počet vstupů modelu. Pro sledování časově proměnných parametrů systému se využívají různé typy zapomínání:

- exponenciální,
- směrově omezené,
- regularizované,
- modifikované regularizované zapomínání - pouze pro rozptyl.

Více informací o jednotlivých typech zapomínání je uvedeno v části 3.2.5.

### 4.4.2 Syntéza řízení

Regulátor používá aditivní kvadratické kritérium řízení. Aditivita a kauzalita systému jsou dvě podmínky, které umožňují minimalizovat toto kritérium odzadu rekurzivně na horizontu $H$. Minimalizované kritérium řízení uvádí následující rovnice (4.2):

$$J(H) = \sum_{k=t}^{t+H} \left\{ ||W_y(y(k) - r(k))||^2 + ||W_u u(k)||^2 + ||W_{\Delta u}(u(k) - u(k-1))||^2 \right\}, \qquad (4.2)$$

kde $W_y$ je matice vah penalizující regulační odchylku, $W_u$ penalizuje energii řízení a $W_{\Delta u}$ energii diference řízení. Kritérium řízení je minimalizované metodou dynamického programování.

Kvůli své rychlosti a také kvůli poměrně vysoké numerické stabilitě jsou všechny výpočty, jak dynamického programování, tak identifikace, prováděny v podobě faktorizovaných matic. Zavedení faktorizovaných tvarů umožňuje velmi elegantní implementaci navrženého algoritmu.

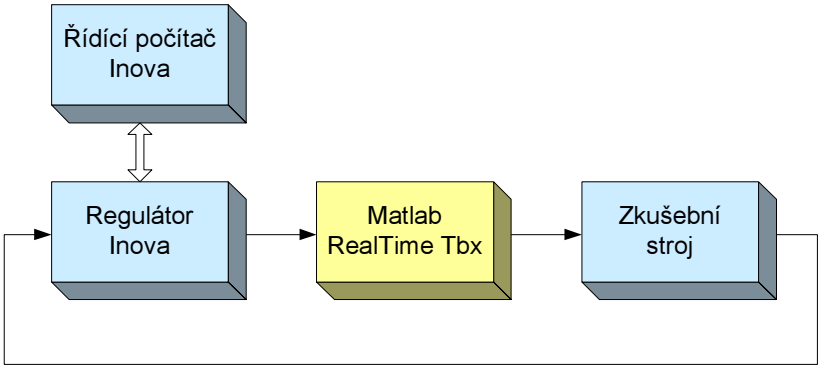
# Kapitola 5

# Řízení elektrohydraulického zkušebního stroje

Od počátku bylo cílem celé této práce ověření teoretických poznatků při řízení reálného stroje. Tomuto cíli jsme podřídili všechny další potřebné práce.

Díky společnosti Inova Praha s. r. o. a Ing. H. Lauschmannovi nám bylo umožněno experimentovat na univerzálním zkušebním stroji umístěném na katedře materiálů v budově Fakulty jaderné a fyzikálně inženýrské Českého vysokého učení technického v Praze.

## 5.1 Realizace experimentů




Obrázek 5.1: Logické schéma experimentu

Zpočátku jsme zvažovali, jak by bylo nejjednodušší řídit zkušební stroj tak, aby nebylo nutné výrazně zasahovat do současné technologie. Přiklonili jsme se k použití dalšího PC s analogovou vstupně-výstupní kartou. Takto připravený počítač jsme začlenili do zpětné vazby, jak je ukázáno na obr. 5.1. V normálním provozu jsou tedy zapojeny pouze modré bloky. Popišme nyní jednotlivé bloky, které jsou zakresleny na obr. 5.1:

- *Řídicí počítač Inova* – na tomto počítači je v normálním provozu spuštěn ovládací program – LabExpert. Umožňuje styk obsluhy se zkušebním strojem. Pomocí tohoto programu lze konfigurovat daný stroj a definovat průběhy zkoušek. Slouží také ke zpracování a zobrazení naměřených dat, definování průběhů žádané hodnoty apod.

- *Regulátor Inova* – toto zařízení realizuje úlohy, které jsou časově kritické, jako je např. přesné vzorkování, měření fyz. veličin, filtrace naměřených signálů, PID regulace, hlídání bezpečnostních mezí síly a polohy a mnoho dalších. Tento blok je v jistém smyslu podřízený řídicímu počítači. V režimu manipulace a před zahájením zkoušky přijímá povely z řídicího počítače. Při spuštění zkoušky přebírá řízení celého stroje a mimo jiné posílá změřená data do řídicího počítače.

- *Matlab, RealTime Tbx.* – v normálním provozu tento blok chybí. Zde byl spuštěn MatLab s RealTime Toolboxem, pro jednoduché začlenění do stávajícího systému. Algoritmus, který se zde prováděl, pouze generoval průběh referenční veličiny, měřil skutečné hodnoty a z nich počítal potřebné akční zásahy. Poslední úloha tohoto bloku byla archivace naměřených dat.

- *Zkušební stroj* – vlastní zkušební stroj.

Po celou dobu našich experimentů byl na řídicím počítači spuštěn pouze jednoduchý program (Watch) pro nastavení regulátoru Inova (zapínání oleje, otevírání zpětné vazby a především hlídání bezpečnostních mezí). Regulátor Inova byl nastaven tak, aby byl vyřazen vnitřní PID regulátor (P=1, I=0, D=0, otevřená smyčka). Zpětná vazba se musela otevřít v okamžiku, kdy se spustil náš regulační algoritmus na vyhrazeném počítači. Od tohoto okamžiku regulátor Inova nijak nezasahoval do řízení.

Obrázek 5.2: Řídicí systém - řídicí počítač Inova (vlevo), regulátor Inova (vpravo)

## 5.2 Použité technické prostředky

Pro řízení stroje jsme použili obyčejné PC (Intel Pentium 4 2.4 GHz, RAM 256 MB) se vstupně-výstupní kartou AD 512. Tato karta obsahuje 12ti bitové A/D a D/A převodníky s minimální vzorkovací periodou 1 ms. Použité softwarové nástroje: MS Windows 2000 Professional, MatLab 6.5, MatLab Simulink a RealTime Toolbox 3.11.

Volba softwarového nástroje MatLab se při experimentech ukázala jako velmi výhodná, především prostředí MatLab Simulink. Během velmi krátké doby jsme byli schopni provést mnoho experimentů a následně pak i velmi lehce naměřená data analyzovat.

Řídicí algoritmus byl napsán jako S-funkce pro MatLab Simulink. Vzhledem k vyšší výpočetní složitosti použitého řídicího algoritmu nastaly problémy s rychlostí výpočtu. Použitý hardware a software umožňoval v nejlepším případě (nejjednodušší varianta algoritmu, malý řád odhadovaného systému apod.) minimální vzorkovací periodu 2 ms.

Na obr. 5.3 je ukázáno jedno ze simulačních schémat v prostředí MatLab Simulink. Následuje popis použitých hlavních bloků RealTime Toolboxu, které jsou obsaženy v simulačním schématu:

- `Adaptive_S` – řídicí algoritmus byl napsán jako S-funkce. Toto je grafické rozhraní pro začlenění v simulaci.

- `RT In` – 12bitový A/D převodník. Vstupní napětí 10 V. Blok zprostředkovává převod analogové veličiny na hodnotu přímo použitelnou v Simulinku.

- `RT Out` – 12bitový D/A převodník. Výstupní napětí 10 V. Blok zprostředkovává převod z hodnoty používané v Simulinku na analogovou veličinu.

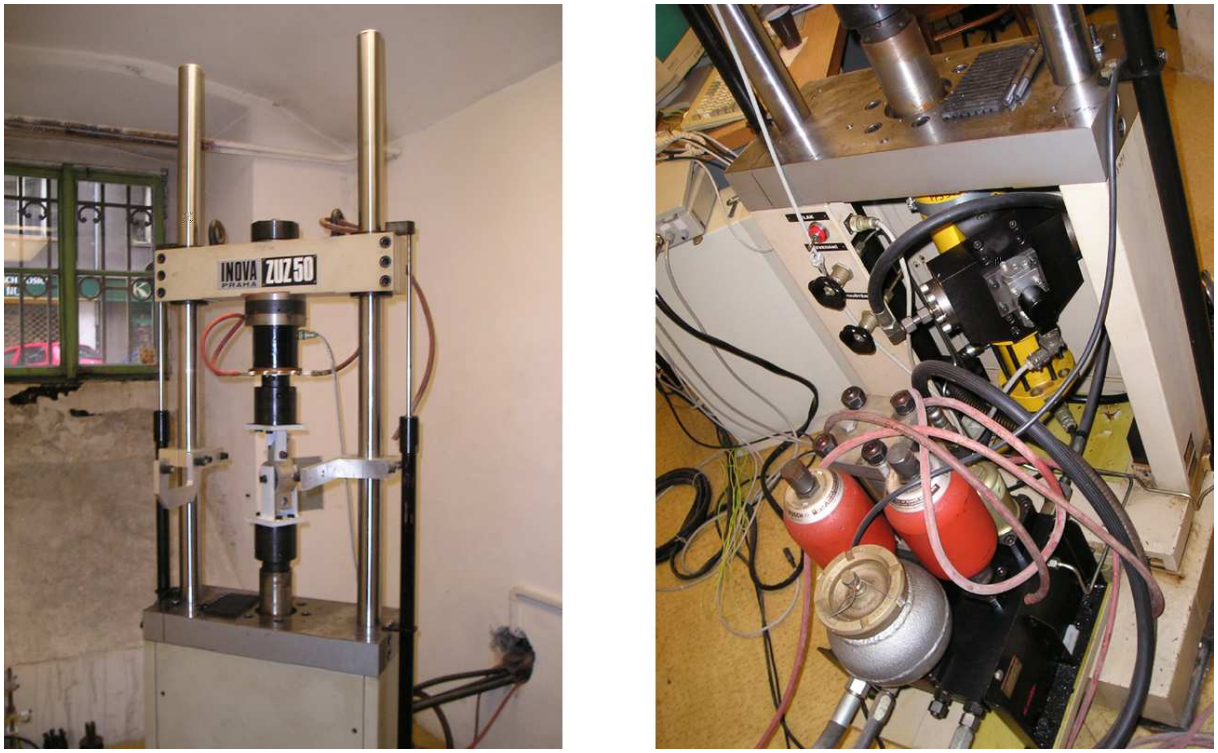
- `Performace Monitor`

## 5.3 Univerzální zkušební stroj ZUZ 50

Univerzální stroj ZUZ 50 (na obr. 5.4) nepatří mezi nejnovější stroje, ale pro naše účely byl plně postačující. Jak již bylo řečeno, technické prostředky a implementace algoritmu nám neumožňovaly zvolit periodu vzorkování 1 ms, jak je u strojů této třídy obvyklé. Technické parametry tohoto stroje jsou v tabulce 5.1. Stroj prošel v posledních letech modernizací, byl vyměněn hydraulický válec, servoventil a řídicí systém. I po modernizaci však patří tento stroj k těm pomalejším, proto jsme mohli vzorkovat také pomaleji (až 6 ms) bez zásadních problémů.

| | | |
|---|---|---|
| Napájecí tlak | 25 – 28 | MPa |
| Jmenovitá síla | 10 | kN |
| Jmenovitý zdvih válce | 100 | mm |

Tabulka 5.1: Technické parametry řízeného stroje



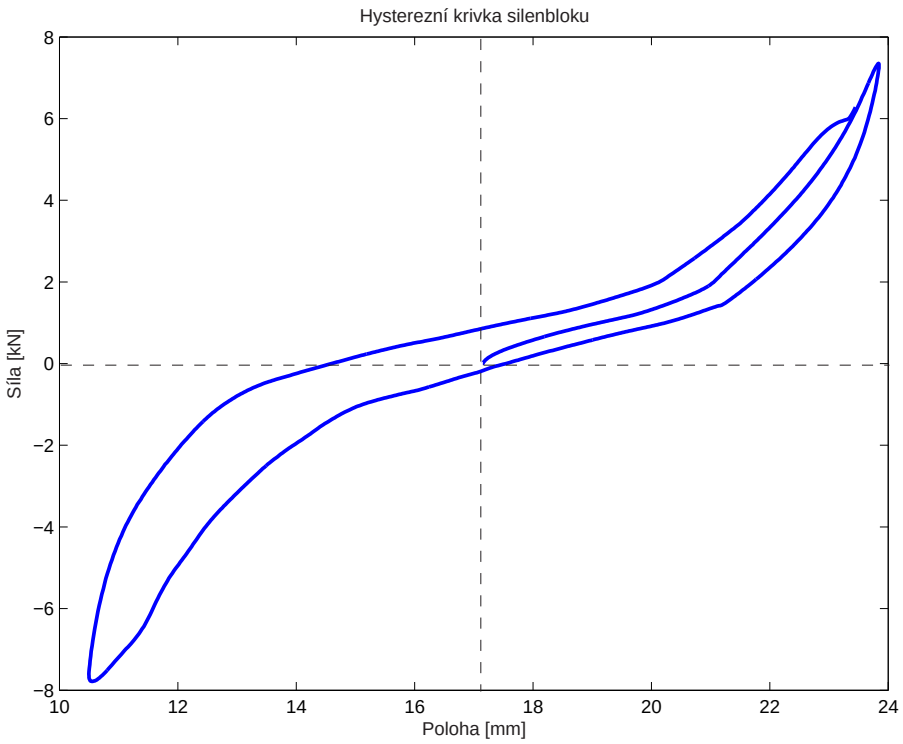

Obrázek 5.4: Univerzální zkušební stroj ZUZ 50 s upnutým vzorkem

Na pístnici zkušebního stroje byl upnut gumový vzorek - silenblok použitý v zadní nápravě vozu Volkswagen Passat (viz. obr. 5.5). Upnutí vzorku k pístnici a rámu stroje bylo značně improvizované. Během testování se stávalo, že se uvolňoval šroub umístěný ve středu silenbloku. Tím se stal systém nejen nelineární ale i časově proměnný. Čas od času bylo nutné tento šroub dotáhnout, opět došlo ke změně vlastností systému. Tím jsme samozřejmě, sice nechtěně, ale velmi lehce otestovali adaptivitu navrženého algoritmu, jak bude ukázáno dále.

Obrázek 5.5: Upnutý vzorek - silenblok

Na obr. 5.6 je ukázána hysterezní křivka stroje s upnutým vzorkem. Podle informací a zkušeností techniků společnosti Inova Praha s.r.o. jsou zatěžovací stroje bez vzorku navrhovány jako poměrně lineární a bez hystereze. Dá se tedy říci, že tato hysterezní křivka je dána především vlastnostmi upnutého vzorku.

Počátek měření hysterezní křivky na obr. 5.6 začíná při nulové síle v poloze $17,15\ mm$. Z obrázku je patrné, že vzorek je tužší na svých krajích. Lépe řečeno, že tuhost vzorku




Obrázek 5.6: Hysterezní křivka silenbloku

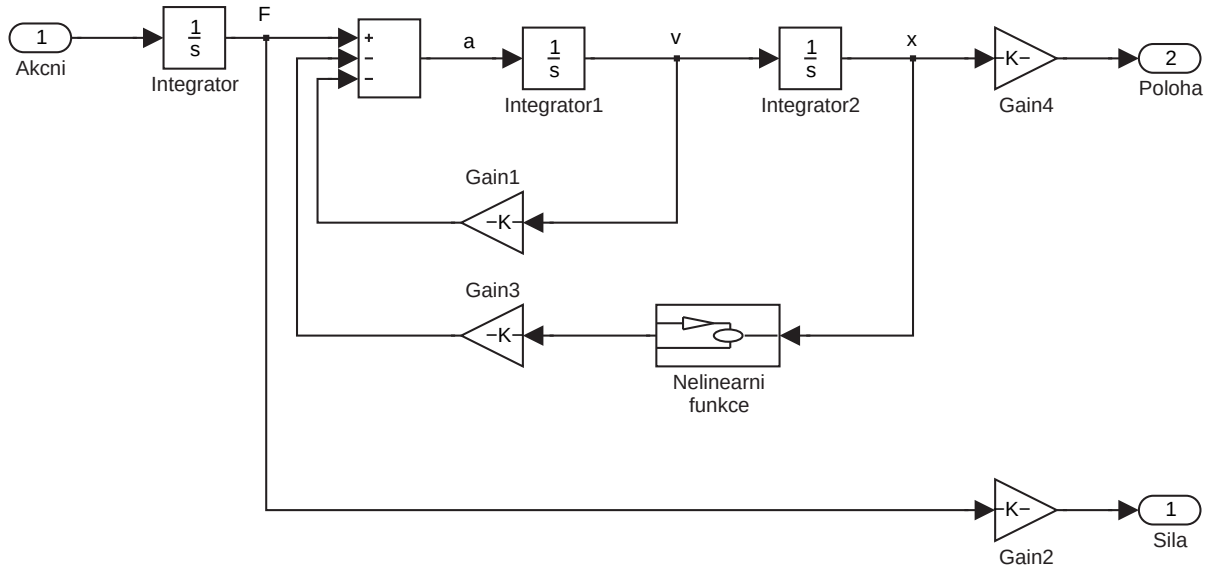
je větší při větších výchylkách pístnice. Je to způsobeno vlastnostmi použitého materiálu a konstrukcí vzorku. Potvrzují to i empirické zkušenosti. Pokusíme-li se stlačovat např. obyčejnou gumu, velmi rychle zjistíme, že potřebujeme stále větší sílu pro vytvoření stejné deformace při nepředepnuté a poté při předepnuté gumě.

## 5.4 Modely systému

V předchozím odstavci je na obr. 5.6 ukázána statická hysterezní křivka daného vzorku resp. celého zatěžovacího stroje s upnutým vzorkem. Z tohoto obrázku vyplývá, že řízený systém má nelineární chování. Pokoušeli jsme se tedy navrhnout model systému také jako nelineární.

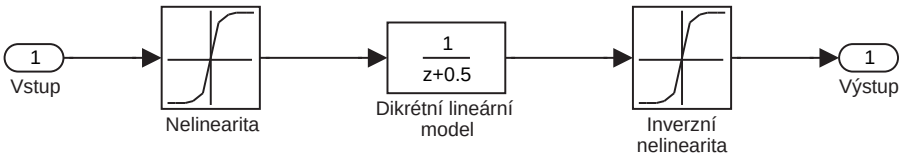
Pokud chceme modelovat dominující dynamické vlastnosti, pak pro většinu jednodušších technických systémů postačuje model třetího až čtvrtého řádu. My jsme zvolili model 2. řádu. Hydraulický servoválec se chová jako integrátor. Proto jsme také k modelu tento

integrátor přidali. Strukturu modelu jsme tedy zvolili jako model 3. řádu s astatismem 1. řádu.




Obrázek 5.7: Model systému

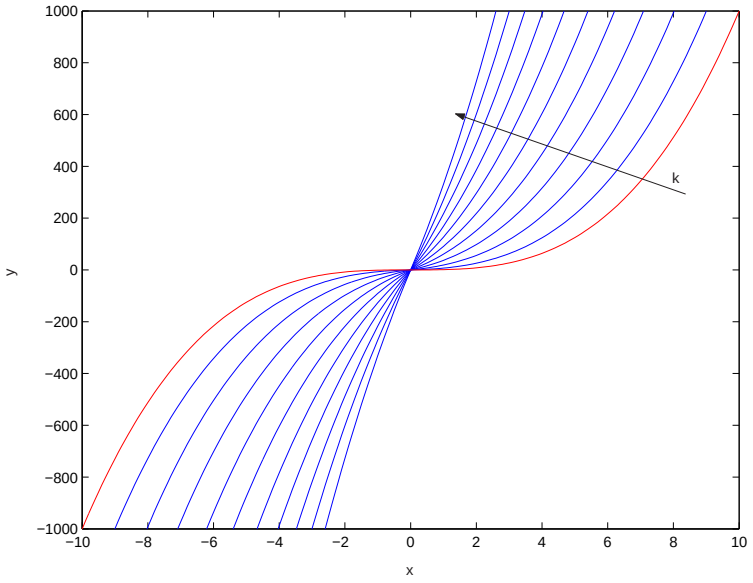
Jednou z možných cest, jak popsat chování některých nelineárních systémů, je použití Wiener-Hammersteinových modelů. Wiener-Hammersteinovy (W-H) modely jsou určeny pro systémy, které mají nelinearity na vstupu nebo na výstupu. Lze tedy říci, že W-H modely vykompenzují nelinearitu na vstupu, odhadnou se parametry řízeného systému jako lineární a následně transformují výstup přes inverzní nelinearitu. Nelinearity se v této verzi neodhadují adaptivně, ale pouze jednorázově před spuštěním řízení. Toto řešení postačuje pro většinu typů nelinearit. Blokově je tato myšlenka zobrazena na obr. 5.8.




Obrázek 5.8: Wiener-Hammesteinův model

Parametry nelineární funkce na obr. 5.6 byly odhadovány ze změřené hysterezní křivky. Změřenou hysterezní křivku jsme nejprve aproximovali po částech lineární funkcí se dvěma body zlomu. Tato volba měla své nevýhody, např. nespojitost derivací v bodech zlomu.

Abychom se tomuto jevu vyhnuli, byla pro modelování vstupních a výstupních nelinearit použita třída polynomiálních funkcí. Polynomiální funkce (jako je např. $x^3$, $x^5$) mají ale také jednu obrovskou nevýhodu, a to takovou, že mají v počátku nulovou derivaci. Jejich inverze tedy mají v počátku derivaci rostoucí nade všechny meze. Proto jsme použili upravenou třídu polynomiálních funkcí s vynechaným intervalem definované délky kolem počátku. Parametr $k$ určuje šířku vynechaného intervalu. Funkce, popsané výše, jsou schématicky ukázány na obr. 5.9.




Obrázek 5.9: Třída funkcí pro modelování nelinearit W-H modelů

Pro řešení jsme vybrali adaptivní algoritmus řízení s lineárním ARX modelem a následnou syntézou lineárně-kvadratického (LQ) regulátoru. Použitý adaptivní algoritmus řízení byl tedy navržen jako lineární, ale je schopen řídit i časově proměnné systémy a také systémy nelineární, kde se pracovní bod systému mění tak, aby se stačily zaktualizovat odhady parametrů modelu. Lineární model lze v tomto případě použít, jelikož se nelineární vlastnosti systému z pohledu regulátoru jevily jako nízkofrekvenční poruchy. Tyto poruchy je regulátor schopen kompenzovat, kvůli svému integračnímu charakteru. S tímto algoritmem jsme provedli poměrně velké množství experimentů, které se pokusíme zhodnotit v následující části. V obecném případě lze před a za tento lineární algoritmus zařadit jednorázově odhadnuté vstupní a výstupní nelinearity, a tak vytvořit výše popisovaný W-H model.

## 5.5 Experimenty

Amplitudy všech naměřených průběhů odpovídají procentům rozsahu dané fyzikální veličiny, nikoli absolutní hodnotě měřené fyzikální veličiny. V tabulce 5.2 jsou uvedeny rozsahy měřených fyzikálních veličin, které se měřením zobrazují do intervalu $\langle -1;\ 1 \rangle$. Akční veličina, tedy výstup regulátoru, byla omezena na stejný interval. Toto zjednodušení není na úkor obecnosti, ale musíme mít na paměti, jaké amplitudy fyzikálních veličin se za změřenými průběhy skrývají. Orientace směru působení síly je podle konvencí v oboru zatěžovacích strojů brána jako kladná, v případě zatěžování v tlaku a záporná v tahu.

| | min | max |
|---|---|---|
| $F[kN]$ | -10 | 10 |
| $x[mm]$ | -50 | 50 |

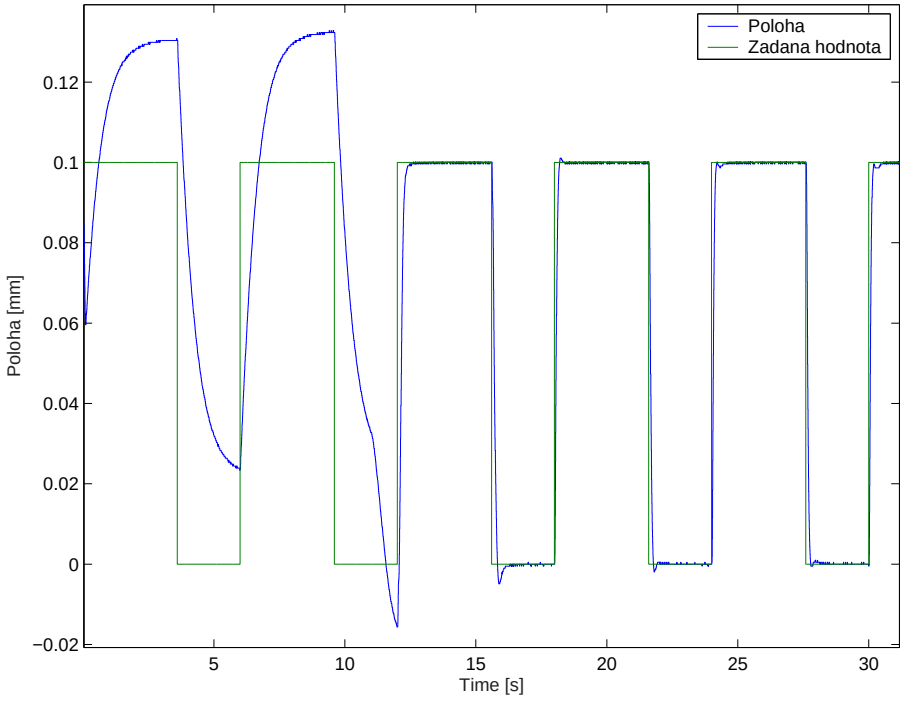
Tabulka 5.2: Rozsahy měřených fyzikálních veličin

### 5.5.1 Inicializace algoritmu

Pro praktické použití adaptivního algoritmu řízení je velmi důležitá otázka bezproblémové inicializace, což v první verzi navrženého algoritmu chybělo. Zpočátku tedy docházelo k velikému kmitání výstupu a někdy až k nestabilitě řízeného systému (nerespektování nelinearit v návrhu řídicího algoritmu), jelikož ještě nestačily zkonvergovat odhady parametrů vytvářeného modelu ke skutečným hodnotám. V této fázi se v podstatě řídilo „naslepo“. Po poměrně krátké době se kmity začaly utlumovat – parametry modelu začaly velice rychle konvergovat se správným hodnotám. Od tohoto okamžiku se dalo říci, že řízení systému probíhalo poměrně stabilně a docházelo jen k malým změnám parametrů. Délka časového intervalu, kdy probíhala inicializační adaptace, velmi závisela na řádu odhadovaného ARX modelu. Tato první verze algoritmu nebyla prakticky použitelná, protože málokterý stroj snese takové zacházení.

Pro další praktické experimenty se tedy musel najít způsob, jak bezpečně inicializovat řídicí algoritmus. Jednou z možností, kterou jsme nakonec použili, byla inicializace fiktivními daty resp. vytvoření referenčního modelu. Před spuštěním algoritmu se zpracovala data, která obsahovala, v obecném případě, hodnoty všech akčních veličin a všech výstupů systému. Mohli jsme použít buď data přímo naměřená na reálném systému, nebo vygenerovat tato data pomocí velmi hrubého modelu.

Pro první experiment jsme použili data, která byla vygenerována modelem 2. řádu s astatismem 1. řádu, kde byly nadhodnoceny časové konstanty a také zesílení systému. Tím, že jsme uvažovali velmi pomalý systém s velkým zesílením, jsme dali regulátoru příliš pesimistickou informaci o řízeném systému. Regulátor tedy zpočátku reagoval „opatrněji“ (s menším zesílením i pomaleji) a s přibývajícími informacemi, získanými přímo z naměřených dat on-line, se model v regulátoru stále více přibližoval fyzikální realizaci. Začal se tedy vzdalovat od referenčního (inicializačního) modelu. Tento případ ukazuje obr. 5.10. Zde je vidět, jak regulátor opatrně zasahoval na počátku experimentu. Za přibližně 11 sekund od začátku řízení je vidět výraznou změnu v dynamických vlastnostech uzavřené smyčky. V tomto čase dochází k upřesnění parametrů modelu a začíná se řídit více podle on-line získaných informací na úkor informací z referenčního modelu.



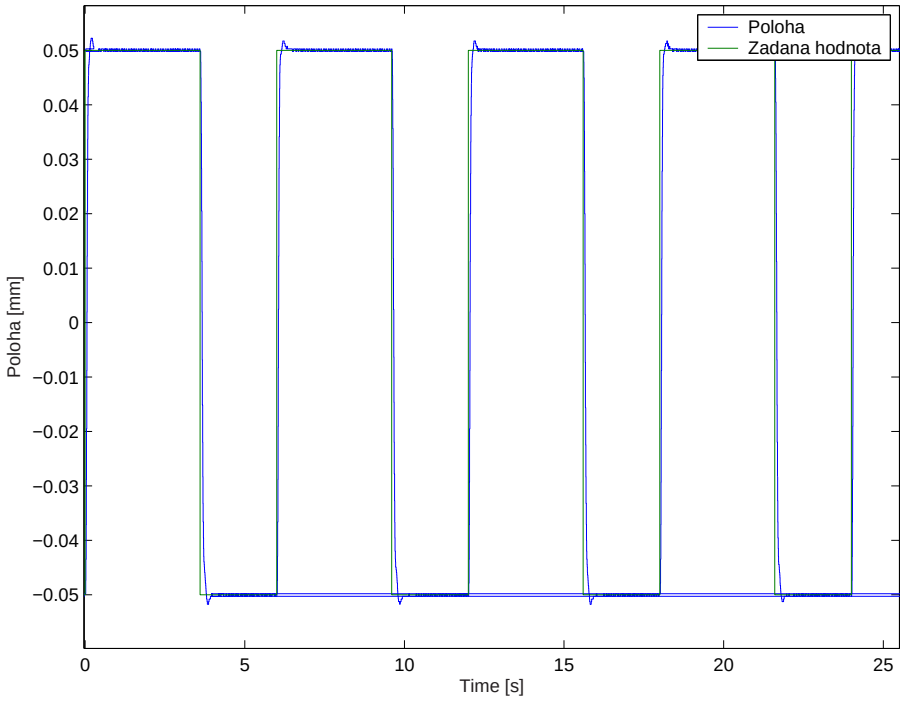

Obrázek 5.10: Inicializace algoritmu uměle vygenerovanými daty

Při použití regularizovaného zapomínání byl aktuální model systému stále ovlivněn referenčním modelem (viz. část 3.2.5). Regulační pochod byl tedy jistým kompromisem mezi řízením podle referenčního modelu a modelu z reálných dat. To bylo znát i na dynamických vlastnostech řízeného systému – pomalejší odezvy na žádanou hodnotu aj. Zlepšení dyna-

mických vlastností uzavřené smyčky je možné použitím směrově omezeného zapomínání, ovšem za cenu ztráty vazby k referenčnímu (bezpečnému) modelu. Chceme-li tuto vazbu zachovat, je výhodné použít jiný typ regularizovaného zapomínání – regularizovat pouze rozptyl odhadu parametrů.

Při dalších experimentech se ukázalo jako výhodné použít pro inicializaci algoritmu, resp. tvorbu referenčního modelu, vhodné úseky dat z minulých experimentů, samozřejmě až po dostatečně dlouhé době po adaptaci. V tomto případě lze získat velmi věrný odhad parametrů model hned od počátku. Zmizí veškeré parazitní překmity po spuštění algoritmu, jak je ukázáno na obr. 5.11.

Velmi zajímavé bylo zjištětní doby, za jak dlouho po startu řízení převáží informace získaná z on-line měření na úkor informace z inicializačních dat. Tato doba byla velmi ovlivněna řádem odhadovaného modelu systému. Pro čtvrtý řád byla tato doba rovna přibližně 11 s. V tomto čase, pak velmi rychle (ale spojitě) převážila informace získaná přímo z měření.

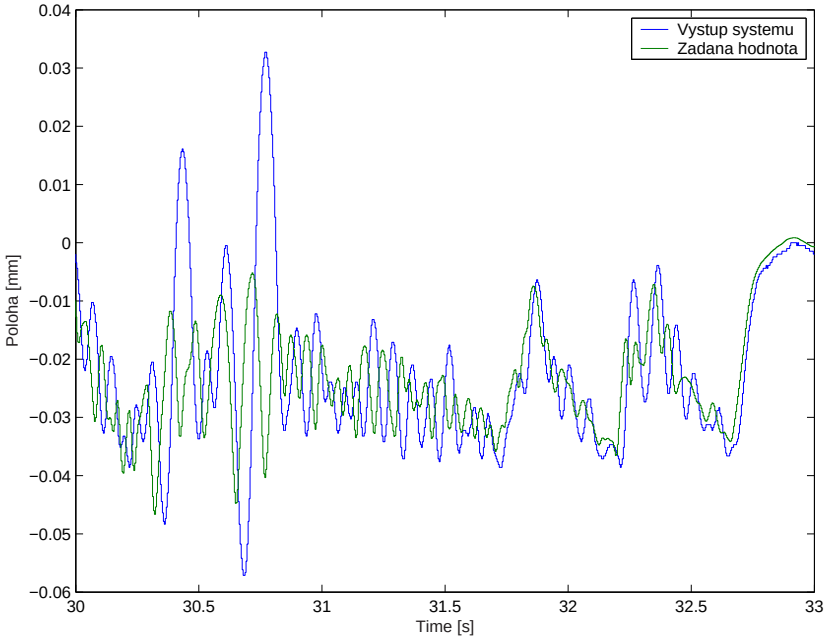



Obrázek 5.11: Inicializace algoritmu vybranými úseky změřených dat

Poté, co jsme vyřešili inicializaci adaptivního algoritmu řízení, podívejme se na některé další vlastnosti navrženého algoritmu.

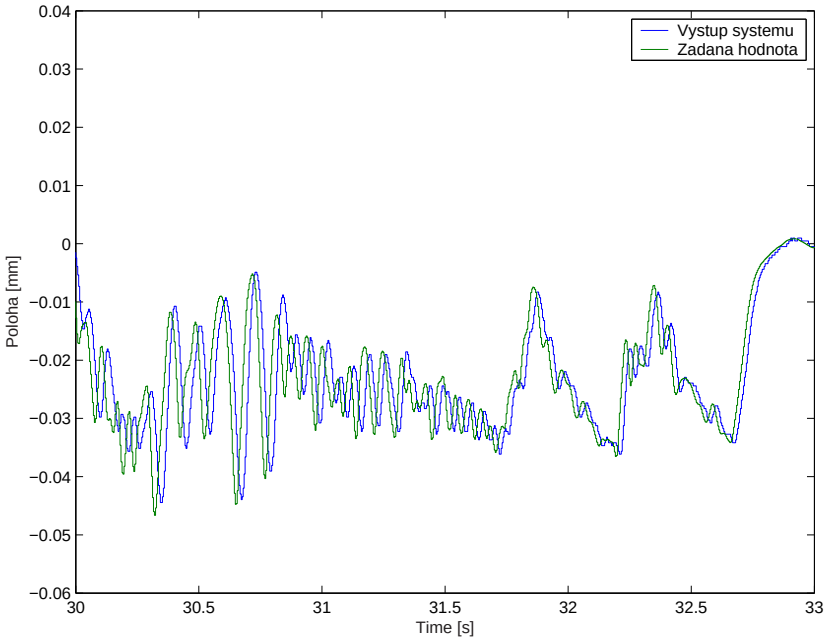
### 5.5.2 Porovnání s PID regulací

Pro další experiment jsme zvolili typický průběh žádané hodnoty při zkouškách provozního namáhání. Jedná se v podstatě o frekvenčně omezený šum.




Obrázek 5.12: PID regulátor

Porovnejme uvedené obrázky 5.12 a 5.13. Použitím PI regulátoru je kvalita sledování žádané hodnoty poměrně špatná. PI regulátor nebyl nastaven optimálně, ale v podstatě jen „od oka“, přesně tak, jak je to nyní běžné. Hlavně nám šlo o srovnání stávajících možností a námi navrhovaného přístupu. Na obr. 5.12 lze najít úseky, kde výstup systému nesleduje referenci ani fázově, ale ani amplitudově. Narozdíl od adaptivního algoritmu řízení na obr. 5.13. Amplitudové poměry jsou zde zachovány. Skutečná hodnota je pouze zpožděná o pevný počet kroků. Fázové zpoždění průběhu skutečné hodnoty (výstupu systému) za průběhem žádané hodnoty je pravděpodobně způsobeno fyzikálními vlastnostmi stroje. Tento fázový posun není z praktického hlediska nijak důležitý, pokud je řízený systém jednoosý (s jedním hydraulickým servoválcem). V případě, že bychom stáli před úlohou řídit víceosý stroj, stačilo by zaručit stejné zpoždění pro všechny válce. Uvážíme-li, že pravděpodobně všechny válce v tomto stroji budou mít velmi podobné fyzikální vlastnosti, není tato podmínka tolik omezující.



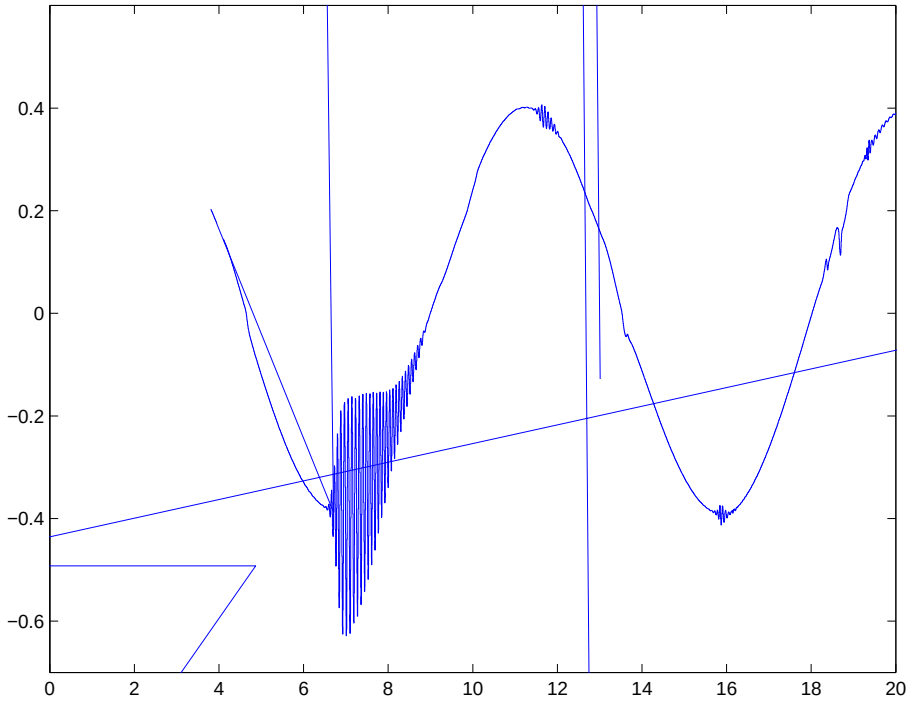

Obrázek 5.13: Adaptivní regulátor

Další možností, jak lze toto fázové zpoždění zmenšit resp. zvětšit robustnost jeho změny, je použití anticipativního regulátoru. Zvolíme-li dostatečně dlouhý horizont, kde známe průběh žádané hodnoty, mělo by být fázové zpoždění konstantní. Regulátor v tomto případě začne zasahovat dříve, než se změní žádaná hodnota. Nejde ale jen o posunutí (zpoždění) žádané hodnoty, nýbrž o výpočet optimálního řízení na daném horizontu.

Porovnejme nyní možnosti řízení nelineárních systémů použitím lineárního PID regulátoru a námi navrženého algoritmu řízení. Lineární PID regulátor, který se v dnešní době běžně používá, v podstatě není schopen dobře řídit tyto nelineární systémy. Použitím adaptivního algoritmu řízení lze dobře řídit nelineární systémy, kde se nelinearita projevuje pomalu. Regulátor ji pak považuje za nízkofrekvenční poruchu, kterou umí dobře kompenzovat. V době, kdy jsme měli k dispozici zatěžovací stroj, bohužel ještě nebylo zcela dokončeno odhadování vstupních a výstupních nelinearit Wienner-Hammersteinových modelů. Dá se předpokládat, že by se kvalita řízení jejich použitím zvýšila.

### 5.5.3 Problémové situace při řízení systému

Při nastavení vah optimalizačního kritéria příliš agresivně (velká váha na sledování výstupu, malé váhy na energii řízení a energii diference řízení) může nastat situace, kdy regulátor vlivem nelinearit ztratí dočasně stabilitu. Tento jev nemůže nastat v případě řízení zcela lineárního systému. Z obr. 5.14 je patrná krátkodobá ztráta stability řízeného systému. Abychom se tohoto jevu vyvarovali, je dobré změnit váhové matice tak, abychom snížili přesnost sledování reference. Tím by mělo dojít ke zvýšení stability uzavřené smyčky.




Obrázek 5.14: Lokální ztráta stability řízeného systému

# Kapitola 6

# Závěr

V předchozím textu jsme představili obor zatěžovacích strojů, strojů, které jsou používány v průmyslu pro nejrůznější testování mechanických vlastností celých výrobků nebo jejich komponent. Podrobněji jsme se věnovali popisu zatěžovacích strojů založených na elektrohydraulickém principu. Stručně jsme popsali jejich jednotlivé komponenty a stručně jsme vysvětlili funkce dané části v rámci celého stroje.

V teoretické části tohoto dokumentu jsme se pokusili nastínit vybrané problémy, které se mohou vyskytnout při použití adaptivního přístupu. Tato část nebyla psána jako vyčerpávající výklad teorií adaptivního řízení, ale spíš měla za úkol nastínit různé alternativy, jak tyto problémy řešit s odkazy na další zdroje, kde je podáno komplexní vysvětlení.

Navrhli jsme adaptivní algoritmus řízení, který jsme implementovali ve formě S-funkce pro Matlab Simulink. Algoritmus řízení je navržen jako vícerozměrový, tedy pro řízení MIMO systémů. Algoritmus patří do skupiny nepřímých samonastavujících se regulátorů (STC - Self Tunning Controller), je založený na odhadování parametrů ARX modelu a syntéze řídicího zákona s kvadratickým kritériem minimalizovaném metodou dynamického programování.

Tento algoritmus řízení jsme použili při experimentech na univerzálním jednoosém zatěžovacím stroji s upnutým nelineárním vzorkem. Během celkem dvou dnů jsme provedli poměrně velké množství experimentů, na kterých jsme prakticky ověřili možnosti a limity použití adaptivní regulace na této třídě strojů. Technické prostředky resp. efektivita implementace algoritmu nám umožnily volit vzorkovací periodu větší než 2 ms, oproti obvyklé 1 ms. Tento aspekt neměl velký vliv, jelikož hydraulický servoválec použitý v řízeném stroji byl relativně pomalý, ale pro rychlejší stroje by větší vzorkovací perioda měla mnohem horší

důsledky. Vzorkovací periodu 2 ms jsme mohli použít pouze pro algoritmus používající IST bez anticipace. V případě použití vícerozměrového řízení s anticipací velmi rychle roste dimenze matic použitých pro výpočet. Nejdelší vzorkovací perioda, kterou jsme použili, byla 6 ms, a to pro anticipativní regulátor s horizontem optimalizace 20 kroků a sedmým řádem odhadovaného modelu systému.

Motivací celé této práce bylo především zjednodušení nastavování regulátorů těchto strojů. Místo nastavování konstant $P$, $I$ a $D$, které se vzájemně ovlivňují a málokdo z obsluhy jim správně rozumí, jsme se snažili vytvořit takový algoritmus, aby se v ideálním případě nemusel nijak nastavovat. I když se u navrženého algoritmu musí nastavit váhové matice, dá se říci, že je to pro obsluhu jednodušší práce, než nastavovat přímo konstanty PID regulátoru. Koeficienty, které figurují ve váhových maticích mají poměrně názorné vysvětlení a také jednoduchá pravidla, jak je volit.

Jedním ze zajímavých a pro praktické použití i velmi důležitých problémů je otázka správné a bezproblémové inicializace algoritmu řízení. Pro řešení jsme vybrali inicializaci pomocí fiktivních dat. Jak již bylo uvedeno, jedná se o zahrnutí apriorní informace o řízeném systému do kovarianční matice odhadů parametrů. Tyto inicializační data mohou být dvou druhů: vybraná část přímo změřená na řízeném stroji (např. přechodová charakteristika) nebo data vygenerovaná simulací velmi jednoduchého systému, který vyjadřuje nejzákladnější vlastnosti pro bezproblémový start algoritmu. Druhá alternativa se ukázala zatím jako výhodnější.

V případě, že není žádoucí adaptivní chování řídicího algoritmu, lze tento algoritmus použít jako pevně nastavený LQ regulátor s předcházejícím jednorázovým automatickým nastavením.

Zatím poslední verze navrženého algoritmu je určena pouze pro experimenty. K tomu, aby jej bylo možné použít i v průmyslové praxi bylo by nutné tento algoritmus zrychlit tak, aby spolehlivě fungoval i s periodou 1 ms. To předpokládá jeho implementaci v jazyku nižší úrovně, např. v C/C++.

Závěrem lze říci, že obor zkušebních (zatěžovacích) strojů je v dnešní době velmi dynamicky se rozvíjející obor, který spojuje velké množství oblastí technického vědění. V tomto oboru se často stává, že se zatěžovací stroje konstruují na míru podle požadavků zákazníků, a proto je zde velký prostor pro vývoj a aplikaci nových, netypických a v jistém smyslu i originálních řešení.

# Literatura


[1] Štecha, J. – Havlena V. *Teorie dynamických systémů.* Praha: Vydavatelství ČVUT, 2000.

[2] Havlena, V. – Štecha J. *Moderní teorie řízení.* Praha: Vydavatelství ČVUT, 2000.

[3] Havlena, V. *Odhadování a filtrace (doplňkové skriptum).* Praha: Vydavatelství ČVUT, 2002.

[4] Bobál, V., et al. *Praktické aspekty samočinně se nastavujících regulátorů Algoritmy a implementace.* Brno: Vydavatelství VUT, 1999.

[5] Štecha, J. *Optimální rozhodování a řízení.* Praha: Vydavatelství ČVUT, 2000.

[6] *Kurs obsluhy a údržby elektrohydraulických zařízení INOVA.* Plzeň: Inova, 1975.

[7] *Den nové techniky – nový elektrohydraulický systém.* Praha: Inova, 1979.

[8] Peterka, V. *Číslicové řízení procesů s náhodnými poruchami a neurčitými charakteristikami.* Doktorská disertační práce. Praha: ÚTIA ČSAV, 1975.

[9] Kulhavý, R. Restricted Exponential Forgetting in Real Time Identification. *Automatica.* 1987, vol. 23, p. 589 – 600.

[10] Krátký, M. Zkušební stroje INOVA zlepšují ve světě pověst českých výrobků. *Technický týdeník.* 2002, vol. 51-52, p. 6.

[11] Peterka, V. Control of Uncertain Processes: Applied Theory and Algorithms. *Kybernetika.* 1986, vol. 22.

# Seznam obrázků

# Seznam tabulek

# Příloha A

# Popis algoritmu řízení

# Adaptivní regulátor

self tuned controller

# S-funkce

## Obsah

# 1 Úvod

Tento literární program implementuje algoritmus adaptivního regulátoru pro Matlab-Simulink. Jedná se o prototyp algoritmu, který je určen pouze pro experimenty. Efektivnější implementace v C++ bude odvozena z tohoto prototypu.
Kód regulátoru zapsaný vektorovým zápisem Matlabu je velmi krátký a stručný.
Adaptivní regulátor je takzvaný *nepřímý samonastavující se regulátor* založený na lineárním modelu ARX a kvadratickém kritériu řízení minimalizovaném metodou dynamického programování. Regulátor, tedy i model, je mnoharozměrový.
Podstatou regulátoru je souběžná rekurzivní minimalizace dvou úloh nejmenších čtverců. Jedna úloha se vztahuje k odhadu parametrů lineárního modelu a druhá úloha se vztahuje k výpočtu regulačního zásahu pro daný model.
Regulátor řeší úlohu sledování. Jeho vstupem je ny výstupů soustavy a ny referencí. Pokud by se stalo, že některý výstup $y_j$ je měřen, ale nemá být regulován, potom je správné formálně nastavit jakoukoliv hodnotu reference $r_j$ pro tento výstup, a do kritéria zahrnout regulační odchylku $j$ s váhou nula.
Minimalizovaná kvadratická forma je popsána jako kvadratická norma nějaké lineární funkce argumentu. Obecně jakákoliv pozitivně semidefinitní forma může být takto zapsána. Takže kvadratická forma $q(x)$ se zapíše:

$$q(x) = ||Qx + f||^2 + g \tag{1}$$

Místo obvyklejší formy $x'Ax + b'x + c$. Norma $||v||^2$ vetoru $v$ je jiným zápisem $v'v$ nebo

$$\sum_i v_i^2.$$

Odsud název metoda nejmenších čtverců. Metodou nejmenších čtverců i odhadujeme parametry modelu, i podle tohoto kritéria řídíme.
Zde matice $Q$ a vektor $f$ jsou parametry lineární funkce. Jelikož otáčení nebo reflexe vektoru nemění jeho délku, je možné vektor $Qx + f$ libovolně otáčet nebo zrcadlit, aniž by se $q(x)$ měnila. Maticově to lze zapsat jako přechod od $||Qx + f||^2$ k $||T(Qx + f)||^2$, kde $T$ je libovolná ortonormální matice. Takže do minimalizační úlohy (1) lze bez obav vložit jakoukoliv matici $T$, která nemění délku. Takové matice jsou právě jen ortonormální matice a geometricky představují rotace a reflexe.
Ještě jednodušší maticový zápis než (1) dostaneme, když spojíme vektor $x$ s jedničkou do jednoho vektoru.

$$q(x) = \left|\left| \begin{pmatrix} Q & f \\ o' & \sqrt{g} \end{pmatrix} \begin{pmatrix} x \\ 1 \end{pmatrix} \right|\right|^2 = ||Qx + f||^2 + g \tag{2}$$

Minimalizace horní trojúhelníkové formy (1) vůči $x$ lze provést rozkladem na úplný čtverec:

$$x^* = Q^{-1} f \tag{3}$$

Forma se nejprve ztrojúhelníkuje vhodnou maticí (transformací) $T$ a potom se minimalizuje triviálně doplněním na čtverec. Optimální hodnota kritéria je přímo číslo $g$. V následujícím zahrneme jedničku do vektoru $(x \quad 1)$ a $f, g$ do $Q$, takže budeme mít jednoduchou funkci

$$\left|\left| Q \begin{pmatrix} x \\ 1 \end{pmatrix} \right|\right|^2 .$$

Pokud takto změníme značení, potom matice, která se invertuje v (3) není celá matice $Q$, ale její levý horní čtvercová submatice rozměru dim$x \times$ dim$x$.

# 2 Rekurzivní minimalizace rotacemi a reflexemi

Jak identifikační tak i regulační část regulátoru minimalizuje součet kvadratických funkcí. To lze zapsat jako součet kvadratických norem

$$\sum_k \left|\left| Q_k \begin{pmatrix} x \\ 1 \end{pmatrix} \right|\right|^2$$

Počet sčítanců roste v čase. Všimneme si ale nejprve třeba tří norem:

$$\left|\left| Q_3^1 \begin{pmatrix} x \\ 1 \end{pmatrix} \right|\right|^2 = \left|\left| \begin{pmatrix} Q_1 \\ Q_2 \\ Q_3 \end{pmatrix} \begin{pmatrix} x \\ 1 \end{pmatrix} \right|\right| \tag{4}$$

Minimalizaci těchto tří norem můžeme provést zatrojúhelníkováním matice $Q_3^1$, a nebo zatrojúhelníkováním nejprve $Q_2^1$, čímž vznikne $Q$, a potom $\begin{pmatrix} Q \\ Q_3 \end{pmatrix}$. Výsledek bude numericky stejný, ale efektivita a paměťové nároky lepší ve druhém případě. Navíc jednotlivé složky kritéria se objevují v čase s novými daty, takže je nutné započítávat je postupně a mít k dispozici poslední aktuální hodnotu minima.

Regulační část regulátoru pracuje na tomto stejném principu jako estimační. Jedna minimalizuje součet čtverců chyb predikce, nebo spíše záporný logaritmus podmíněné hustoty pravděpodobnosti parametrů, a druhá minimalizuje kritérium řízení, které je také kvadratické. Liší se jen v tom, že jedna přibírá nové členy $Q_k$ dopředu a druhá dozadu ve směru času.

Matice $Q_k$ nemusí být čtvercová. Protože potřebujeme pro minimalizaci doplněním na úplný čtverec, aby blok matice $Q$, který se má invertovat, byl regulární a čtvercová, je tento algoritmus definován až tehdy, až má matice $Q$ dosáhne plné sloupcové hodnosti. Jednodušší je zvolit pro matici $Q$ vhodnou počáteční podmínku tak, aby hned tato počáteční podmínka měla nezávislé sloupky. Potom dalším přidáváním řádků se tato nezávislost sloupků již neztratí a inverze bloku $Q$ je zaručeně definována. Protože ale vliv počáteční podmínky časem mizí, je ale potřeba ještě mít na paměti další opatření (viz. regularizace, matice $R$).

Počáteční podmínka pro algoritmus odhadování souvisí s apriorní hustotou pravděpodobnosti parametrů.

# 3 Rekurzivní identifikace ARX modelu

Tento regulátor je navržen tak, že model je ve tvaru:

$$\sum_{n=0}^{\mathrm{order}} \left\{ \sum_{k=1}^{\mathrm{ny}} a_{kn}^{j} y_k(t-n) + \sum_{h=1}^{\mathrm{nu}} b_{hn}^{j} u_h(t-n) \right\} + d^j + e^j(t) \tag{5}$$

Model říká, že nějaká lineární funkce konečného počtu posledně naměřených dat ve střední hodnotě nulová, což je vlastnost lineárního stochastického systému. Povšimněte si, že v modelu je i absolutní člen $d^j$. Pokud nechceme tento člen uvažovat, změníme hodnotu proměnné `rejectd.` Někdy je známo, že absolutní člen je nulový.
Spolu s koeficienty $a$ a $b$ je to další parametr modelu. K modelu musíme připojit několik vysvětlujících poznámek:

1. Index $j$ označuje rovnici. Obecně může být model s ny výstupy popsán nejvýše ny lin. nezávislými rovnicemi, jinak by byl sporný. Nejčastěji v modelu pro řízení je rovnic právě ny. Pro účely validace dat stačí rovnic méně.

2. V každé rovnici se vyskytuje chyba rovnice $e^j(t)$. Tento člen vyjadřuje skutečnost, že model se se skutečným systémem shoduje pouze v průměru. U tohoto členu předpokládáme statistické vlastnosti: nulová střední hodnota, neznámý rozptyl (výkon), nezávislost $e^j(t)$ a $e^i(s)$ jeli alespoň jedna z následujících podmínek splněna: $t \neq s$ nebo $i \neq j$. Zde je nutno zdůraznit, že první podmínka je podmínka bělosti a druhá podmínka nezávislosti mezi rovnicemi. Obě podmínky musíme vysvětlit dále. Ještě předpokládáme normalitu rozdělení pravděpodobnosti $e(t)$.

3. Z důvodu jednoduchosti a přehlednosti jsou všechny veličiny $y$ a $u$ uvažovány ve stejně hluboké historii délky order.

Model napíšeme úsporněji maticově, třeba pro $\mathrm{ny} = 2$ a $\mathrm{nu} = 2$:

$$e(t) = \Theta \begin{pmatrix} y_1(t) & y_2(t) & u_1(t) & u_2(t) & \ldots & u_2(t-\mathrm{order}) & 1 \end{pmatrix} \tag{6}$$

Geometricky to znamená, že vektor dat, zvaný regressor $\phi(t)$, je v průměru kolmý na řádky nějaké matice $\Theta$ nebo `Theta`, to jest matice modelu. Skalární součin je potom $e(t)$. Matice $\Theta$ v sobě obsahuje všechny parametry.
Matice $\Theta$ se blíží předpisu, jak předpovídat budoucí výstupy pomocí vstupů a několika minulých hodnot vstupů a výstupů, vlastně stavu systému. Pokud bychom chtěli předpovídat, museli bychom soustavu lineárních rovnic (6) vyřešit vzhledem k vektoru neznámých $y_1(t)$ a $y_2(t)$ a psát $t+1$ namísto $t$. Jedním z důležitých vlastností našeho regulátoru je to, že nepotřebuje mít rovnice explicitně vyřešeny vzhledem k výstupu. Následkem toho se náš regulátor nemusí obávat situace, že model je náhodou takový, že nelze jednoznačně predikovat budoucí hodnoty výstupů. Jak uvidíme dále, regulátor může provádět své vnitřní výpočty třeba i s nulovou maticí $\Theta$, která

neobsahuje žádnou informaci. Jeho řídicí zásahy jsou pro takovou matici nulové, ale podstatné je, že po nějaké době, pokud je vše v pořádku, tato situace vymizí a řízení začne opět fungovat. Explicitní prediktor by bylo možno také popsat maticí $\Theta$ ovšem jejích prvních ny sloupců by byla jednotková matice.
Toto souvisí i s vzájemnou nezávislostí rovnic. Chyba predikce, tedy chyba modelu explicitně vyřešeného vzhledem k $y(t)$, není nikdy nezávislá. Uvažme třeba příklad teploty a tlaku páry v kotli: za okamžik $T_s$ můžeme předpovědět hodnotu teploty $T$ a tlak $p$. Pokud ale dojde ke zvýšenému ochlazování během $T_s$, potom lze očekávat $T-\delta T$ a $p-\delta p$, a lze předpovědět, jaké kombinace obou chyb jsou pravděpodobné. Proto by bylo nutno kovarianční matici prediktoru $\mathcal{E}\{e(t)e(t)'\}$ uvažovat jako obecnou pozitivně semidefinitní matici a odhadovat ji, pokud nechceme predikovat nepřesně. Protože ale implicitní model obsahuje navíc právě tolik parametrů, kolik obsahuje kovarianční matice, můžeme v implicitním modelu uvažovat nezávislost rovnic, což není totéž, jako nezávislost chyb predikce. Dokonce bychom mohli uvažovat jednotkovou matici $\mathcal{E}\{e(t)e(t)'\}$. Matice $\Theta$ už potom obsahuje nutné stupně volnosti modelu a nelze ji dále omezovat. V tomto programu počítáme s diagonální maticí var$\{e\}$ (ny parametrů navíc) a matice $\Theta$ pak obsahuje ny jedniček.
Jak tedy odhadujeme matici $\Theta$ : Definujeme následující úlohu nejmenších čtverců:

$$q(x)=\left|\left|\begin{pmatrix} y_1(0) & y_2(0) & \dots & u(0-\text{order}) & 1 \\ y_1(1) & y_2(1) & \dots & u(1-\text{order}) & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ y_1(t) & y_2(t) & \dots & u(t-\text{order}) & 1 \end{pmatrix}\begin{pmatrix} x_1 \\ 1 \\ x_2 \end{pmatrix}\right|\right|^2=\left|\left|Q\begin{pmatrix} x \\ 1 \end{pmatrix}\right|\right|^2 \tag{7}$$

Minimalizací $q(x)$ vůči $x$ dostaneme jeden řádek matice $\Theta$. Další řádek dostaneme stejně jako první když přidáme podmínku, že musí být na prvním nezávislý. S maticí $Q$ zacházíme jak bylo uvedeno na začátku, to jest rotujeme vektor $Q(x;1)$ maticí rotace a reflexe $T$ tak, aby byla matice $TQ$ byla stále horní trojúhelníková. Matici $TQ$ získáme v Matlabu snadno QR dekompozicí. QR dekompozice matice $Q$ je její vyjádření jako součinu ortonormální matice $T$ a horní trojúhelníkové matice $R$. Matice $R$ pak vlastně nahrazuje $Q$.

$$Q=TR \qquad R=T'Q \tag{8}$$

Poznamenejme, že transponovaná matice $T'$ je také ortonormální a je inverzní k $T$. Minimalizace $||Q(x;1)||$ vede na $x=0$, tedy lineárně závislou rovnici. Připočteme tedy omezení $\theta_{jj}=1$, to jest jeden koeficient zvolíme jedna. Potom minimalizaci provedeme, jak bylo naznačeno v (3).
Matici $\Theta$ odhaduje následující úsek programu: **estimate** $\Theta$. Postupuje tak, že permutuje sloupky matice kvadratické formy v cyklu $Q$ tak, aby to $x_i$, které se má rovnat 1, bylo na konci. Potom lze totiž jednoduše minimalizovat doplněním na úplný čtverec. Pokud je splněna podmínka ftype = 'regul2', potom se neminimalizuje kvadratická forma $||Q(x;1)||$, ale součet

$$||Q(x;1)||+\kappa^2||x-\theta'_j||, \tag{9}$$

čímž lze omezit rychlost změn odhadu parametrů, což je dosti užitečné. Číslo $\kappa$ zkusíme zvětšit, pokud chceme změny odhadu zatlumit. Nový řádek matice theta je penalizován, pokud se příliš odlišuje od minulého odhadu.

⟨estimate $\Theta$ 6⟩ ≡

```
⟨forgetting 7⟩
[q, Q] = qr([Q;regressor], 0);
count = sqrt(lambda)*count + 1;
for k = 1 : ny
   [q, F] = qr(Q( : , [1 : k-1, k+1 : end, k]), 0);
   if strcmp(ftype, 'regul2')
      y = [Theta(k, [1 : k-1, k+1 : end]), 0];
      G = eye(N)*1e-2;
      G( : , end) = G*y';
      [q, F] = qr([F;G], 0);
   end
   x = inv(F(1 : end-1, 1 : end-1))*F(1 : end-1, end);
   Theta(k, :) = [x(1 : k-1)', -1, x(k : end)'];
   Sigma(k) = sqrt((F(end, end)^2+1e-5)/count);
end
*
```

Macro referenced in scraps 14a, 19.

# 4 Zapomínání

Kvůli schopnosti algoritmu sledovat časově proměnné parametry matice modelu je potřeba odstraňovat starou informaci, což se nejsnáze provede exponeniciálním zmenšováním starých čtverců, takže stará data mají exponenciálně se zmenšující vliv na odhad parametrů. Úsek programu **forgetting** implementuje exponenciální zapomínání, ale i mnohem užitečnější regularizované zapomínání, kdy je stará informace nahrazována jinou regularizující informací. Ta je zadána jako matice $R$ a její význam je úplně týž jako matice $Q$. Kritérium nejmenších čtverců je potom:

$$q(x) = \left|\left| \begin{pmatrix} \lambda^{t/2} & (y_1(0) & y_2(0) & \dots & u(0-\text{order}) & 1) \\ \lambda^{(t-1)/2} & (y_1(1) & y_2(1) & \dots & u(1-\text{order}) & 1) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ & (y_1(t) & y_2(t) & \dots & u(t-\text{order}) & 1) \end{pmatrix} \begin{pmatrix} x_1 \\ 1 \\ x_2 \end{pmatrix} \right|\right| + ||Rx||, \quad \lambda \in (0,1\rangle \tag{10}$$

⟨forgetting 7⟩ ≡

```
if strcmp(ftype, 'dircon')
   Nul = null(regressor);
   [q, Q] = qr([Q*sqrt(lambda); sqrt(1-lambda)*Q*Nul*Nul'], 0);
elseif strcmp(ftype, 'regul2')
   Q = Q*sqrt(lambda);
elseif strcmp(ftype, 'exp')
   Q = Q*sqrt(lambda);
elseif strcmp(ftype, 'regul')
   [q, Q] = qr([Q*sqrt(lambda);sqrt(1-lambda)*R], 0);
elseif strcmp(ftype, 'none')
else
   disp('unknown forgetting type, known types are dircon, exp, none')
end
*
```

Macro referenced in scrap 6.

Zapomínání je v našem programu transformace matice $Q$. Transformovanou matici (po zapomínání) označíme $Q'$. Například pro exponenciální zapomínání $Q' = \sqrt{\lambda}Q$.
Dalším užitečným způsobem zapomínání je směrově omezené zapomínání. Jde o to, že měřená data neakumulují informaci v prostoru parametrů $\theta_{i,j}$ rovnoměrně, ale nahodile podle způsobu vybuzení řízeného systému. Geometricky se to projevuje tak, že konfidenční elipsoid v prostoru parametrů se v některých směrech velmi protahuje vlivem zapomínání rychlostí $\lambda^{-t/2}$. Jde o to, že v těch směrech, ve kterých mají data informační hodnotu a jsou schopny konfidenční elipsoid zmenšovat, je možno zapomínat. Naopak ve směrech, ve kterých data konfidenční elipsoid nezmenšují, je lepší ponechat aspoň starou informaci. Uspořádání výpočtu je v úseku **forgetting** na dvou řádcích. Nejprve je celá matice $Q$ vynásobena $\sqrt{\lambda}$, jako při neselektivním exponenciálním zapomínání. Potom je ale informace ve směrech kolmých k současnému regressoru zpět přičtena informace násobená $\sqrt{1-\lambda}$. Ve výsledku je ve směrech kolmých k regressoru je pro $N = \mathrm{Ker}\phi$ je $NN'x_1 = x_1$, a $NN'x_2 = 0$ ve směru regressoru. Ve směrech kolmých se nezapomíná:

$$\begin{aligned} ||Q'x_1||^2 &= ||\sqrt{\lambda}Qx_1||^2 + ||\sqrt{1-\lambda}Qx_1||^2 \\ &= ||Qx_1||^2 \end{aligned}$$

A ve směru regressoru se zapomíná exponenciálně s koeficientem zapomínání $\lambda$ :

$$||Q'x_2||^2 = ||\sqrt{\lambda}Qx_2||^2$$

# 5 Odhad Σ

Vektor `Sigma` je odmocninou diagonály odhadu kovarianční matice $\mathcal{E}\{e^j(t)e^i(t)\}$. Rozptyl $e^j(t)$ odhadneme jako podíl minimální kvadratické normy signálu $\{e^j(0),\ldots,e^j(t)\}$ a času $t$. Čas $t$ se v algoritmu udržuje v proměnné `count`. Takže (stříška značí odhad):

$$\hat{\sigma}_e^2 = \sum_{k=0}^{t} \frac{e^2(k)}{t} = \frac{g}{t} \tag{11}$$

Tento odhad systematicky podceňuje, protože normu $e_t^0$ odhadujeme minimální hodnotou, ale toto podcenění má zmenšující se vliv s rostoucím $t$. Číslo $g$ v rovnici (11) má stejný význam jako v (1). Při exponenciálním zapomínání s koeficientem $\lambda$ nelze $\sigma^2$ odhadovat jako $g/t$, protože $g$ není $\sum e^2(t)$, ale je:

$$g = \sum_{k=0}^{t} \lambda^{t-k} e^2(t) \tag{12}$$

A z toho potom plyne pro $\hat{\sigma}^2$

$$\hat{\sigma}_e^2 = \frac{g}{\sum_{k=0}^{t} \lambda^{t-k}} = g/\texttt{count} \tag{13}$$

Takže `count` je čas, který roste zpočátku lineárně, ale potom se jeho růst zpomaluje a ustálí se na hodnotě, která souvisí s délkou datového okna. Odhad parametrů je založen hlavně na datech z datového okna. Mimo datové okno jsou již data "zapomenuta". Tím se také vysvětluje proměnná `count,` která se neshoduje s proměnnou $t$, což je skutečný čas v sekundách. Pokud je zapomínání s koeficientem $\lambda$, je `count` rovno

$$\texttt{count} = \frac{1-\lambda^{t/T_s}}{1-\lambda}.$$

# 6 Dynamické programování

Regulátor používá aditivní kvadratické kritérium řízení. Aditivita kritéria a kauzalita systému jsou dvě podmínky, které umožňují minimalizovat toto kritérium odzadu rekurzivně na horizontu $H$. Spočívá to v tom, že kritérium $J$ je suma následujícího typu:

$$J(H) = \sum_{k=t}^{t+H} q_k(y(k),u(k),r(k)) \tag{14}$$

Minimalizace probíhá za omezení typu rovnost která je dána vztahem mezi vstupy a výstupy systému. Tuto sumu rozdělíme na dvě:

$$J(H) = \sum_{k=t}^{t+H-1} q_k(y(k),u(k),r(k)) + q_H(y(t+H),u(t+H),r(t+H)) \tag{15}$$

Pokud minimalizujeme poslední sčítanec vůči $u(t+H)$ a dosadíme optimální hodnotu $u^*(t+H)$ do kritéria, potom se nám podaří snížit počet proměnných o dim$u$ + dim$y$. Je to proto, že $u(t+H)$ bylo dosazeno a $y(t+H)$ vypadne vlivem omezení, pokud je omezení jednoznačné. Pokud to rekurzivně aplikujeme, potom nakonec dostaneme v posledním kroku řízení v čase $t$, to je teď. Toto řízení regulátor aplikuje na vstup systému.
Výhoda je ta, že namísto minimalizace obrovského množství sčítanců současně pracujeme vždy jen s jedním relativně jednoduchým členem $q_k$. To je kvadratická forma typu

$$q_k = ||W_y(y(k) - r(k))||^2 + ||W_u u(k)||^2 + ||W_{\Delta u}(u(k) - u(k-1))||^2 \tag{16}$$

My tento člen zapíšeme jako normu $||V\phi(t)||^2$, kde $\phi(t)$ je regressor $(y_1(t), \ldots, 1)$. Matice $V$ je statická proměnná, která se inicializuje na podkladě matic vah $W$. Matice $W$ zadáváme jako diagonální. Motivace pro toto je pouze ta, že s diagonálními maticemi ve většině případů vystačíme, a zadávat celé matice je zdlouhavé.
Jeden člen penalizuje regulační odchylku, druhý energii řízení a třetí energii diference řízení. Tři matice $W$ jsou váhové matice, kterými lze regulátor ladit. Jedno číslo v těchto matic9ch je přebytečné, protože záleží jen na poměru vah, na tvaru kritéria. Samotné přenásobení všech vah kladným číslem nezmění nic.
Kauzalita systému je zde potřebná, protože na časovém úseku $\langle t, t+H\rangle$ ovlivňuje řízení $u(t+H)$ právě nejvýše jen poslední člen a ostatní $q$ nejsou kvůli kauzalitě funkcí $u(t+H)$. Při minimalizaci posledního sčítance v $J$ nedostaneme $u^*(t+H)$ jako číselný vektor, ale jako funkci několika minulých hodnot $u$ a $y$. Tuto funkci dosadíme do $J(H)$ a získáme tím $J(H-1)$, až nakonec $J(0)$.
Následující úsek kódu **optimize** aktualizuje kvadratickou $||P(x;1)||^2$, kterou si zhruba můžeme představit jako právě poslední $q_k$ v rekurzivní minimalizaci. Není to tak úplně přesně, protože na rozdíl od $q$ je zde ještě více zpožděných $y$ a $u$ jako argumenty, aby bylo možno zahrnout i omezení. Kvůli omezením nepracujeme jen s posledním $q$, ale se složitější funkcí. Tato funkce je shodou okolností funkcí regressoru v daném časovém okamžiku, protože lineární omezení dané modelem je právě funkcí regressoru. Je to funkce: $||V\phi(k)||^2$.
V části **optimize** se děje toto:

1. Přičítají se k $||P(x;1)||^2$ další členy

   $$||W_y(y(t) - r(t))||^2 + ||W_u u(t)||^2 + ||W_{\Delta u}(u(t) - u(t-1))||^2.$$

   Tyto členy jsou uspořádány v matici $V$.

2. Potom se přičítá člen $10^7||\Theta\phi(t)||^2$. Tím se omezení typu rovnost převádí na kvadratickou funkci, která nabývá rychle velkých hodnot tam, kde omezení není splněno. To je lepší a jednodušší, než opravdu zahrnovat omezení typu rovnost, což by vyžadovalo explicitně vyjádřit $y(t)$. $10^7$ zde vystupuje jako $\infty$ a pokud by problém byl tak škálován, že by se vyskytovaly váhy srovnatelné, potom by věc již nefungovala a regulátor by přestal řídit.

3. Potom se najde $u^*(t) = K\phi(t)$ minimalizací doplněním na úplný čtverec. Přesněji není $u^*(t)$ lineární funkcí celého regressoru, ale pouze podvektoru regressoru počínaje $y_1(t-1)$. Současné hodnoty $y(t)$ (a pochopitelně $u(t)$) nejsou pro výpočet $u(t)$ potřeba. Je to proto, že systém může sám o sobě mít přímou vazbu (a model ji má) a potom výpočet $u(t)$ z $y(t)$ nedává smysl, protože $y(t)$ už reaguje na $u(t)$. Regulátor tak má právě jednu periodu vzorkování na výpočet zásahu do řízení.

4. Potom se forma $||P(x;1)||^2$ změní tak, aby odpovídala dalšímu sčítanci, který se bude minimalizovat příště. Zbaví se proměnných $y(t)$ a $u(t)$. Za $u(t)$ i $y(t)$ se dosadí minimalizující hodnoty. Naopak se přidají proměnné $y(t-\mathrm{order}-1)$ a $u(t-\mathrm{order}-1)$. Takže následující člen již je funkcí $\phi(t-1)$.

⟨optimize 10⟩ ≡

```
[q, P] = qr([P; V;
          [Theta, zeros(ny, ny)]*1e7], 0);
K = -inv(P([1 : nu]+ny, [1 : nu]+ny))*P([1 : nu]+ny, ny+nu+1 : end);
P = P(ny+nu+1 : end, ny+nu+1 : end);
P = insertm(P, zeros(ny+nu, ny+nu), N-ny-nu-1);
u = K*[regressor(1 : end-nu-ny-1)';1; ref];
*
```

Macro referenced in scrap 14a.

# 7 Horizont optimalizace

Ve výkladu dynamického programování bylo vysvětleno, že lze tímto algoritmem snadno minimalizovat kvadratické kritérium řízení na horizontu $H$. Je známo, že pokud je horizont příliš krátký, potom může být tento zákon řízení nestabilní. Pro delší horizont je ale potřeba mnoho výpočtů. To lze ale obejít tak, že si poslední člen minimalizace uschováme a použijeme ho v další vzorkovací periodě jako výchozí člen. Tím se dostaneme na neustále rostoucí horizont optimalizace. V první vzorkovací periodě je horizont optimalizace 1, potom 2, pak 3, atd. Stačí v každé vzorkovací periodě provést jen jednu minimalizaci kvadratické formy v prostoru regressoru. Je sice pravda, že lineární omezení započítávané do kritéria se ve vzdálenějších (dříve minimalizovaných) sčítancích nerovná aktuálnímu odhadu parametrů, ale pokud se parametry mění jen pomalu, ve vztahu k časovým konstantám systému, potom to příliš nevadí. Tento algoritmus reaguje na skokovou změnu parametru hůře, než například regulátor na klouzavém horizontu, ale zase je rychlejší co se doby výpočtu týče.

# 8 Algoritmus jako S-funkce

Pro pokusy je výhodné mít algoritmus ve formě bloku pro Simulink. Vezměme šablonu S-funkce a doplňme ji o algoritmus. Parametry `t`, `x`, `u`, a `flag` jsou povinné. Povinný parametr `u` (vstup bloku) jsme oproti šabloně přejmenovali na `inp`, protože $u$ znamená vstup do řízeného systému $u(t)$, což je právě výstup regulátoru. Nebylo vhodné pojmenovat tento parametr ani jako $y$, protože `inp` obsahuje i referenci. Vstupem regulátoru je vektor $[y(t); r(t)]$.
Další parametry jsou ladící a konfigurační a jsou už speciální pro tuto S-funkci:

| | |
|---|---|
| `ny` | počet výstupů řízeného systému |
| `nu` | počet vstupů řízeného systému |
| `order` | řád modelu (pro jednoduchost stejný pro všechny polynomy, viz. Závěr) |
| `Wy` | diagonála váhy regulační odchylky, vektor nezáporných čísel |
| `Wu` | diagonála váhy vstupu, vektor nezáporných čísel |
| `Wd` | diagonála váhy čas. změny vstupu, vektor nezáporných čísel |
| `lambda` | faktor zapomínání |
| `ftype` | řetězec, viz. **forgetting** |
| `Data` | regularizující informace |
| `umax` | saturace vstupu soustavy $\lvert u(t)\rvert \leq u_{\max}$ |
| `Ts` | perioda vzorkování |

"adaptive_S.m" 11 ≡

```
function [sys, x0, str, ts] = adaptive_S(t, x, inp, flag, ...
   ny, nu, order, Wy, Wu, Wd, lambda, ftype, Data, umax, Ts)
⟨define static vars 12a⟩
if  abs(flag) == 2
    ⟨next regulator state 14a⟩
elseif flag == 3
    ⟨regulator output 14b⟩
elseif flag == 4
   sys = t + 1;
elseif flag == 0
    ⟨initialize the regulator 13b⟩
else
   sys = [];
end
*
```

Pravá S-funkce nemůže obsahovat statické proměnné, protože potom nelze současně spouštět dva regulátory implementované stejnou S-funkcí. Pro pokusy připustíme statické proměnné, ale nesmíme zapomenout, že může být spuštěn vždy jen jeden adaptivní regulátor, a nebo dva, ale spuštěné z S-funkce uložené pod jiným názvem. Statické proměnné obcházejí simulink, který

právě předpokládá, že stav bloku je uložen v jeho stavu $x(t)$. Stavem regulátoru jsou: regressor, dvě matice $Q$, $P$ a count. Ostatní statické proměnné jsou parametry, které sám regulátor nemění: $R, V$ ale musí si je pamatovat, a nebo si je naopak nepotřebuje pamatovat, ale jsou statické, aby je nebylo nutné vždy znovu alokovat: $\Theta$ a $\Sigma$. `N` je jen pomocný integer, délka regressoru.

⟨define static vars 12a⟩ ≡

```
persistent N;
persistent V;
persistent regressor;
persistent P;
persistent Theta;
persistent Sigma;
persistent count;
persistent Q;
persistent R;
*
```

Macro referenced in scraps 11, 18b.

Během inicializace si jednak nainicializujeme vlastní statické proměnné, a potom řekneme simulinku nějaké informace o naší S-funkci. Tyto informace jsou uloženy ve struktuře sizes.

⟨initialize static vars I 12b⟩ ≡

```
N = (order+1)*(nu+ny)+1;
regressor = [zeros(1, N-1), 1];
Q = eye(N, N)*1e-2;
Theta = zeros(ny, N);
Sigma = zeros(ny, 1);
count = 1;
*
```

Macro referenced in scraps 13b, 18a.

Další blok inicializace označíme jako II, protože se bude odlišovat v dalším regulátoru, kdežto blok proměnných I bude společný.

⟨initialize static vars II 13a⟩ ≡

```
V = zeros(ny+2*nu, N+ny);
V(ny, [1 : ny]) = diag(Wy);
V(ny, N+[1 : ny]) = -diag(Wy);
V(ny+[1 : nu], ny+[1 : nu]) = diag(Wu);
V(ny+nu+[1 : nu], ny+[1 : nu]) = diag(Wd);
V(ny+nu+[1 : nu], ny+nu+ny+[1 : nu]) = -diag(Wd);
Psize = (ny+nu)*(order+1)+ny+1;
P = zeros(Psize, Psize);
```

∗

Macro referenced in scrap 13b.

⟨initialize the regulator 13b⟩ ≡

⟨initialize static vars I 12b⟩
⟨initialize static vars II 13a⟩
⟨process the initial data 15⟩
⟨define sizes 13c⟩
∗

Macro referenced in scrap 11.

⟨define sizes 13c⟩ ≡

```
sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 1;
sizes.NumOutputs     = nu;
sizes.NumInputs      = ny*2;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0 = 0;
ts = [Ts 0];
```

∗

Macro referenced in scraps 13b, 18a.

Vlastní algoritmus probíhá v úseku kódu, který má spočítat nový stav regulátoru. Zde se rozbalí vektor `inp` na výstup a referenci. Potom se aktualizuje regressor. Nejprve se do něj nahraje $y(t)$, přičemž $u(t)$ už v něm je od minulého volání regulátoru. Potom se aktualizuje odhad parametrů

modelu a následně se provede jeden krok dynamického programování. Toto dynamické programování poskytne optimální $u(t+1)$. Tato hodnota se uloží do regressoru, kde bude připraveno na další vzorkovací periodu. Zároveň se aplikuje na vstup řízené soustavy.
Pokud je známo, že na vstupu soustavy je omezení, potom do regressoru v případě, že $u(t+1)$ tuto amplitudu přesahuje, uložíme omezenou hodnotu. Regulátor sám sice s omezením nepočítá, ale alespoň se dozví, že jeho výsledek někdo změnil. Lze snadno zjistit, že ukládání neoříznutých $u(t+1)$ má velmi špatný vliv na regulaci. To se ale projeví jen pokud se omezení opravdu uplatňuje.
Všimneme si také, že do času `order` nelze odhadovat parametry, protože ještě nejsou k dispozici data $u(t-k)$ a $y(t-k)$ do regressoru. To sice neznamená, že do času `order` by opravdu nešlo odhad parametrů aktualizovat, nějaká informace k dispozici je už dříve, ale nebylo by to jednoduché.

⟨next regulator state 14a⟩ ≡

```
y = inp(1 : ny);
ref = inp(ny + [1 : ny]);
regressor([1 : ny]) = y';
if t > order
    ⟨estimate Θ 6⟩
end
⟨optimize 10⟩
regressor = [zeros(1, nu+ny), regressor(1, 1 : end-1-nu-ny), 1];
u = max([min([u';ones(1, nu)*umax]);-ones(1, nu)*umax]);
regressor(ny+[1 : nu]) = u';
sys = 0;
*
```

Macro referenced in scrap 11.

Část kódu, která má poskytnout výstup regulátoru, vrací hodnotu *u* předtím spočítanou.

⟨regulator output 14b⟩ ≡

```
sys = regressor(ny + [1 : nu])';
*
```

Macro referenced in scraps 11, 18b.

Součástí inicializace je i zpracování regularizačních dat. Tato data si můžeme představit jako data, která jsou k dispozici už v čase $t < \mathrm{order}$. Navíc tato data se nezapomínají, ale stále se udržují v informaci pro odhad parametrů. Jde o jakousi pojistku proti různým problémům, které při odhadování parametrů v uzavřené smyčce vznikají. Zároveň se tím regulátor dá donutit k

rozumnému chování hned v prvních okamžicích, pokud je regularizační informace dostatečně přesná.
Matice `Data` obsahují po řádcích vektory

$$\begin{pmatrix} y_1(t) & \dots & y_{\mathrm{ny}}(t) & u_1(t) & \dots & u_{\mathrm{nu}}(t) \end{pmatrix}$$

Matice `Data` se převede na kvadratickou formu $||Rx||^2$ definovanou ’381n074287(c)-0.154.607 0 Td [(:)-111.7

"datamatrix.m" 16a ≡

```
plant2 = zpk([], [0 -.1], 0.1);
plant = zpk([], [0], 1);
h = step(plant, [0 : 1 : 100]);
Data = [h, ones(size(h))];
[PID, K, FFPID] = data2pid(c2d(plant, 1), [], 0, 1, 1, .9);
[PI, K, FFPI] = data2pid(c2d(plant, 1), [], 0, 1, [], .9);
*
```

# 9 Anticipativní regulátor

Nevýhodou regulátoru **adaptive_S** je to, že považuje během horizontu optimalizace referenci $r(t)$ za konstantu *r*. To vyhovuje pro příklady, kdy se reference mění pomalu nebo zřídkakdy. Pro referenci neustále se měnící po dopředu známé trajektorii je lepší tuto trajektorii reference uvažovat. Předpokládejme, že existuje informace o referenci `horiz` kroků dopředu. Potom přepíšeme dynamické programování tak, že regulátor vidí na `horiz` kroků dopředu referenci a za tímto horizontem vidí referenci již konstantní $r(t+\mathrm{horiz})$. Nevýhodou je, že minimalizace provedená na úseku $[t,t+\mathrm{horiz}]$ je nepoužitelná pro další výpočty a v další vzorkovací periodě se startuje z členu $t+\mathrm{horiz}$.
Regulátor nepracuje na horizontech 1, 2, 3 atd, ale 1+horiz, 2+horiz atd. Vlastní optimalizace se tedy skládá z horiz+1 kroků dynamického programování, přičemž první mezivýsledek *P* si ukládáme. Ten potom překopírujeme do $P_2$ a provedeme vlastní optimalizaci na intervalu $\langle t,t+\mathrm{horiz}\rangle$.

⟨optimize (anticipative) 16b⟩ ≡

```
k = 0;
⟨optimize horiz + 1 17a⟩
P2 = P;
⟨optimize 1 : horiz 17b⟩
*
```

Macro referenced in scrap 19.

⟨optimize horiz + 1 17a⟩ ≡

```
V([1 : ny], N+ny*k+[1 : ny]) = -diag(Wy);
[q, P] = qr([P; V;
          [Theta, zeros(ny, ny*(horiz+1))]*1e7], 0);
V([1 : ny], N+ny*k+[1 : ny]) = zeros(ny, ny);
P = P(ny+nu+1 : end, ny+nu+1 : end);
P = insertm(P, zeros(ny+nu, ny+nu), N-ny-nu-1);
*
```

Macro referenced in scrap 16b.

⟨optimize 1 : horiz 17b⟩ ≡

```
for k = [1 : horiz]
   V([1 : ny], N+ny*k+[1 : ny]) = -diag(Wy);
   [q, P2] = qr([P2; V;
             [Theta, zeros(ny, ny*(horiz+1))]*1e7], 0);
   V([1 : ny], N+ny*k+[1 : ny]) = zeros(ny, ny);
   if k == horiz
      K = -inv(P2([1 : nu]+ny, [1 : nu]+ny))*P2([1 : nu]+ny, ny+nu+1 : end);
      u = K*[regressor(1:end-nu-ny-1)';1; ref; href'];
      break;
   end
   P2 = P2(ny+nu+1 : end, ny+nu+1 : end);
   P2 = insertm(P2, zeros(ny+nu, ny+nu), N-ny-nu-1);
end
*
```

Macro referenced in scrap 16b.

Inicializace regulátoru proběhne stejně jako inicializace neanticipativního regulátoru pouze s tím rozdílem, že matice $V$ a $P$ jsou jiných rozměrů a že existuje další proměnná `href` – historie reference, kde je uložena budoucnost reference `horiz` kroků dopředu. V čase $t$ je na vstupu regulátoru reference $r(t + \mathrm{horiz})$.

⟨initialize the regulator (anticipative) 18a⟩ ≡

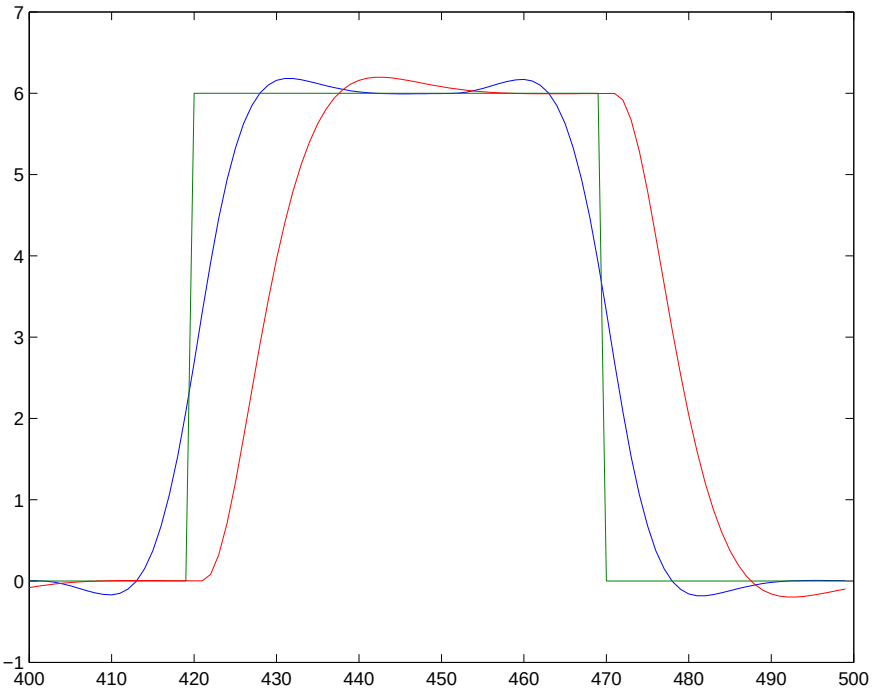
```
href = zeros(1, ny*horiz);
⟨initialize static vars I 12b⟩
V = zeros(ny+2*nu, N+ny*(horiz+1));
V([1 : ny], [1 : ny]) = diag(Wy);
V(ny+[1 : nu], ny+[1 : nu]) = diag(Wd);
V(ny+[1 : nu], ny+nu+ny+[1 : nu]) = -diag(Wd);
V(ny+nu+[1 : nu],ny+[1:nu]) = diag(Wu);
Psize = (ny+nu)*(order+1)+ny*(horiz+1)+1;
P = zeros(Psize,Psize);
⟨process the initial data 15⟩
⟨define sizes 13c⟩
*
```

Macro referenced in scrap 18b.

Vlastní S-funkce anticipativního regulátoru je téměř shodná s neanticipativním regulátorem.

"predadaptive_S.m" 18b ≡

```
function [sys,x0,str,ts] = predadaptive_S(t,x,inp,flag,...
   ny,nu,order,Wy,Wu,Wd,lambda,ftype,Data,umax,horiz,Ts)
⟨define static vars 12a⟩
persistent href;
if  abs(flag) == 2
    ⟨next regulator state (anticipative) 19⟩
elseif flag == 3
    ⟨regulator output 14b⟩
elseif flag == 4
   sys = t + 1;
elseif flag == 0
    ⟨initialize the regulator (anticipative) 18a⟩
else
   sys=[];
end
*
```




Obrázek 1: Vliv předvídání reference při jinak stejném nastavení.

⟨next regulator state (anticipative) 19⟩ ≡

```
y = inp(1 : ny);
ref = inp(ny + [1 : ny]);
regressor([1 : ny]) = y';
if t > order
    ⟨estimate Θ 6⟩
end
⟨optimize (anticipative) 16b⟩
href = [ref', href(1:end-ny)];
regressor = [zeros(1, nu+ny), regressor(1, 1 : end-1-nu-ny), 1];
u = max([min([u';ones(1, nu)*umax]);-ones(1, nu)*umax]);
regressor(ny+[1 : nu]) = u';
sys = 0;
```

∗

Macro referenced in scrap 18b.

Na obrázku (1) je typický průběh sledování reference pro anticipativní (`horiz` = 20) a neanticipativní regulátor (`horiz` = 0).

# 10 Pomocné funkce

Pomocná funkce **insertm** vkládá blok do matice na její diagonálu za *n* diagonální prvek.

"insertm.m" 20 ≡

```
function y = insertm(a,b,n)

a1 = a(1:n,1:n);        a2 = a(1:n,n+1:end);
a3 = a(n+1:end,1:n);    a4 = a(n+1:end,n+1:end);

z1 = zeros(n,size(b,2));
z2 = zeros(size(b,1),n);
z3 = zeros(size(b,1),size(a,2)-n);
z4 = zeros(size(a,1)-n,size(b,2));

y = [
    a1, z1, a2
    z2, b,  z3
    a3, z4, a4];
*
```

# 11 Závěr

Algoritmus je poměrně kompaktním zápisem samonastavujícího se regulátoru. V současné době máme tento algoritmus doplněn o model s neznámými (ale odhadovanými) monotonními nelinearitami na vstupu a výstupu, což je tzv. Wiener Hammersteinův model. Dále máme tento algoritmus doplněn o automatický Bayesovský odhad řádu modelu. Plánujeme rozšíření o model ARMAX a další vlastnosti, jako podpora robustnosti pomocí frekvenčně vážených nejmenších čtverců, kde amplitudová charakteristika filtru se automaticky ladí tak, aby min. singulární číslo zpětné diference bylo maximalizováno. To má pozitivní vliv na robustní stabilitu.

# 12 Index

Indexováno číslem stránky a na stránce potom od shora *a*, *b* atd.

## 12.1 Soubory

## 12.2 Makra

⟨define sizes 13c⟩ Referenced in scraps 13b, 18a.
⟨define static vars 12a⟩ Referenced in scraps 11, 18b.
⟨estimate Θ 6⟩ Referenced in scraps 14a, 19.
⟨forgetting 7⟩ Referenced in scrap 6.
⟨initialize static vars II 13a⟩ Referenced in scrap 13b.
⟨initialize static vars I 12b⟩ Referenced in scraps 13b, 18a.
⟨initialize the regulator (anticipative) 18a⟩ Referenced in scrap 18b.
⟨initialize the regulator 13b⟩ Referenced in scrap 11.
⟨next regulator state (anticipative) 19⟩ Referenced in scrap 18b.
⟨next regulator state 14a⟩ Referenced in scrap 11.
⟨optimize (anticipative) 16b⟩ Referenced in scrap 19.
⟨optimize 1 : horiz 17b⟩ Referenced in scrap 16b.
⟨optimize horiz + 1 17a⟩ Referenced in scrap 16b.
⟨optimize 10⟩ Referenced in scrap 14a.
⟨process the initial data 15⟩ Referenced in scraps 13b, 18a.
⟨regulator output 14b⟩ Referenced in scraps 11, 18b.

## 12.3 Proměnné

`count`: 6, 12a, 12b, 15.
`Data`: 11, 15, 16a, 18b.
`ftype`: 6, 7, 11, 18b.
`horiz`: 16b, 17ab, 18a, 18b.
`href`: 17b, 18a, 18b, 19.
`K`: 10, 16a, 17b.
`lambda`: 6, 7, 11, 18b.
`N`: 6, 10, 12a, 12b, 13a, 15, 17ab, 18a.
`nu`: 10, 11, 12b, 13ac, 14ab, 15, 17ab, 18ab, 19.
`ny`: 6, 10, 11, 12b, 13ac, 14ab, 15, 17ab, 18ab, 19.
`order`: 11, 12b, 13a, 14a, 15, 18ab, 19.
`P`: 10, 12a, 13a, 16b, 17a, 18a.
`P2`: 16b, 17b.
`Q`: 6, 7, 12a, 12b, 15.
`R`: 7, 12a, 15.
`regressor`: 6, 7, 10, 12a, 12b, 14ab, 15, 17b, 19.
`rejectd`: 15.
`Sigma`: 6, 12a, 12b, 15.
`sizes`: 13b, 13c, 18a.
`Theta`: 6, 10, 12a, 12b, 14a, 15, 17ab, 19.
`Ts`: 11, 13c, 18b.
`u`: 10, 14a, 15, 17b, 19.
`umax`: 11, 14a, 18b, 19.

V: 10, 12a, 13a, 17ab, 18a.
Wd: 11, 13a, 18ab.
Wu: 11, 13a, 18ab.
Wy: 11, 13a, 17ab, 18ab.