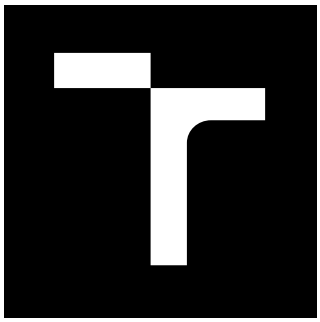



# VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

## FAKULTA STROJNÍHO INŽENÝRSTVÍ

FACULTY OF MECHANICAL ENGINEERING

## ÚSTAV AUTOMATIZACE A INFORMATIKY

INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE


# NELINEÁRNÍ ŘÍZENÍ KOMPLEXNÍCH SOUSTAV S VYUŽITÍM EVOLUČNÍCH PŘÍSTUPŮ

NONLINEAR CONTROL OF COMPLEX SYSTEMS BY UTILIZATION OF EVOLUTIONARY APPROACHES

**DIZERTAČNÍ PRÁCE**
DOCTORAL THESIS


**AUTOR PRÁCE** — **Ing. Petr Minář**
AUTHOR

**ŠKOLITEL** — **doc. Ing. Radomil Matoušek, Ph.D.**
SUPERVISOR



**BRNO 2017**

## ABSTRAKT

Problematika optimalizace složitých soustav za použití algoritmů umělé inteligence, je relativně nový vědní obor a má mnohé způsoby využití v technické praxi. Vhodné algoritmy na řešení podobných úloh jsou třeba genetický algoritmus, diferenciální evoluce, algoritmus HC12, metoda nelder-mead, fuzzy logika a gramatická evoluce. Kompletní řešení je prezentováno na vybraných příkladech od matematických soustav nelineárních systémů, až po praktické úlohy spolu s návrhem antén a stabilizace deterministického chaosu.

Práce si klade za cíl navržení jednotlivých postupů využití algoritmů umělé inteligence při vícekriteriální optimalizaci. K dosažení optimálních výsledků slouží navržené softwarové řešení na základě multi-platformové aplikace v rámci Matlab a Java rozhraní. Softwarové řešení spojuje všechny algoritmy do ucelené aplikace a dále rozšiřuje možnosti uplatnění výsledků na reálných soustavách a v technické praxi.

## KLÍČOVÁ SLOVA

soft computing, umělá inteligence, optimalizace, automatizace, regulace, nelineární systémy, fuzzy logika, chaosové systémy, logistická mapa, duffingova rovnice, uda-yagi, java, matlab, simulink, paralelní výpočty, PID, HC12, DE, NM, GE, GA, TDAS, ETDAS, ITAE

## ABSTRACT

Control theory of complex systems by utilization of artificial intelligent algorithms is relatively new science field and it can be used in many areas of technical practise. Best known algorithms to solved similar tasks are genetic algorithm, differential evolution, HC12 Nelder-Mead method, fuzzy logic and grammatical evolution. Complex solution is presented at selected examples from mathematical nonlinear systems to examples of anthems design and stabilization of deterministic chaos.

The goal of this thesis is present examples of implementation and utilization of artificial algorithms by multi-objective optimization. To achieve optimal results is used designed software solution by multi-platform application, which used Matlab and Java interfaces. The software solution integrate every algorithms of this thesis to complex solution and it extends possible application of those approaches to real systems and practical world.

## KEYWORDS

soft computing, artificial intelligence, optimization, automation, control theory, nonlinear systems, fuzzy logic, chaos systems, logistic map, duffing equation, uda-yagi, java, matlab, simulink, parallel computing, PID, HC12, DE, NM, GE, GA, TDAS, ETDAS, ITAE

MINÁŘ, Petr *Nelineární řízení komplexních soustav s využitím evolučních přístupů*: dizertační práce. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav procesního a ekologického inženýrství, 2016. 177 s. Vedoucí práce byl doc. Ing. Radomil Matoušek, Ph.D.

# PROHLÁŠENÍ

Prohlašuji, že svou doktorskou práci na téma „Nelineární řízení komplexních soustav s využitím evolučních přístupů“ jsem vypracoval samostatně pod vedením vedoucího doktorské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené doktorské práce dále prohlašuji, že v souvislosti s vytvořením této doktorské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

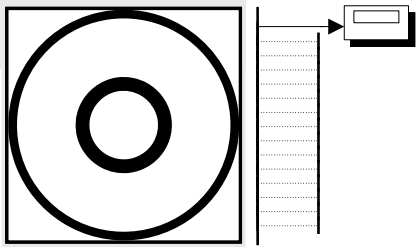
Brno 2017/04/05

(podpis autora)

# PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu doktorské práce panu doc. Ing. Radomilovi Matouškovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

# OBSAH



# ÚVOD

V oblasti technické praxe existuje mnoho složitých optimalizačních problémů, které se dají jen velmi složitě řešit standardními numerickými metodami a někdy se vyskytují i takové příklady, kdy optimální řešení numerickými metodami je jen velice těžko dohledatelné. V této práci jsou proto využívány a prezentovány výsledky pro algoritmy umělé inteligence a heuristických metod k vyhledání optimálních výsledků vybraných složitých regulačních úloh a specifických příkladů, jako jsou soustavy chaosových systémů a návrh antén. Celkově tyto příklady představují stručný souhrn možných řešení v dané problematice a představují i aplikační výzkum nových způsobů pří řešení podobných problémů v praxi.

Algoritmy založené na umělé inteligenci (Artificital Intelegence), popřípadě počítačové inteligenci (Computer intelligence) a soft computing metody, jsou relativně nový vědní obor, který se dynamicky rozvíjí do všech směrů technické praxe. Jejich hlavní a nesporná výhoda je univerzálnost aplikace na většinu optimalizačních problémů, dle nastavení hodnotící funkce. V této době je využití těchto algoritmů ještě umocněné zvyšující se výkonností počítačových stanic, kdy je možno využít algoritmy umělé inteligence i na velice složité problémy, které dosahují takového stupně složitosti, při kterých už nejde uspokojivě využít matematické metody řešení a dokáží najít optimální řešení problémů v relativně uspokojivém čase. Tento fakt umocňuje i nástup tzv. paralelních výpočtů, kdy jeden složitý problém je možno počítat více entitami, jak na jedné stanici, tak v rámci cloudového řešení s více stanicemi v jednom clusteru. Algoritmy umělé inteligence jsou velice jednoduše aplikovatelné na paralelní strukturu výpočtu a obecně jejich univerzálnost a jednoduchost při aplikovaní v různých programovacích jazycích.

Do oblasti umělé inteligence a soft computing metod patří i algoritmy, které jsou v práci dále rozebírány a použity k řešení optimalizačních problémů. Mezi metody patří například genetické algoritmy (Genetic algoritm), diferenciální evoluce (Differential evolution) a hlavní algoritmus této teze HC12, který je porovnáván v průběhu celé práce s velkou množinou vybraných algoritmů na různých typech problémů a různých typech nastavení hodnotící funkce. Dále je v průběhu práce využívána metoda Nelder-Mead jako zástupce standardního způsobu řešení optimalizačních pro-

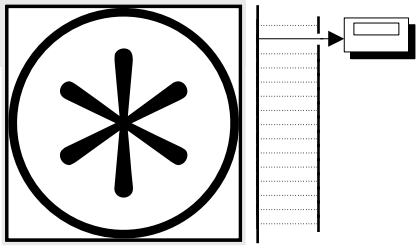
blémů v rámci počítačové inteligence. Poslední používaný způsob optimalizace je generování vlastních regulačních pravidel dle metody gramatické evoluce (Grammatical evolution) a fuzzy logiky optimalizované pomocí soft computing metod.

Hlavní cíl optimalizace v technické praxi v rámci regulace systémů je regulace na požadovanou hodnotu. V této práci jsou k demonstraci využívány nelineární soustavy, jak čistě matematického modelu, tak praktických úloh regulace sypkých hmot a regulace polohy tělesa v magnetickém poli. Další modely v práci se zabývají hojně rozvíjenými metodami stabilizace deterministického chaosu, a to konkrétně logistické mapy a duffingovy rovnice a porovnáním ke standardním metodám řešení. Poslední ukázka využití umělé inteligence slouží k navrhování neméně atraktivního modelu z hlediska technické praxe, a to navrhovaní struktury Yagi-Uda antény. Všechny tyto problémy mají společnou netriviálnost řešení a jsou velice složité na dosažení optimálního výsledku, který neobsahuje jen regulaci na jednu žádanou hodnotu, ale může obsahovat více druhů kritérií z hlediska hodnocení výsledné hodnoty. Proto musíme zavést vlastní hodnotící funkce na optimalizaci vícekriteriálních hodnot.

V prezentovaném řešení je důležité i ukázat nejenom obecné způsoby aplikovaní algoritmů na jednotlivé příklady, ale i prezentovat možné řešení celkového vypočtu v praxi. V dnešní době existuje mnoho programovacích jazyků a mnoho výpočetních / simulačních aplikací, které mají specifické vlastnosti, s kterými je důležité se seznámit a aplikovat. Proto je velice důležité i softwarová stránka práce a jakým způsobem je možné jednotlivé postupy aplikovat v rámci počítačové vědy a využít co největší potenciál jednotlivých řešení, například distribuovaných výpočtů. Existuje mnoho přístupů k tomuto návrhu, kde neexistuje jen jedno globálně používané řešení, každé řešení je specifické na daný typ zadaní a problému. Proto řešení prezentované dále v této práci je zvoleno s ohledem na co největší univerzálnost a co největší možnost uplatnění dále v praxi.

Celá práce je rozdělena na teoretickou část, kde je velice stručně vysvětlena problematika regulace nelineárních systémů a obecný pohled na optimalizaci, spolu s popisem jednotlivých metod optimalizace. V praktické části jsou prezentovány jednotlivé příklady a stručný popis jejich modelů a navržené optimalizace dle způsobu využití algoritmů umělé inteligence. V neposlední řadě také návrh samotné hodnotící funkce pro multikriteriální optimalizaci. V této sekci je i stručné zhodnocení a statistika jednotlivých metod ke standardnímu postupu regulace a stabilizace. Každý jednotlivý příklad a navrhovaný postup řešení je popsaný v samostatné kapitole. Aplikační řešení je prezentováno v rámci samostatné aplikace, jejího podrobného popisu a přikladu využití tohoto řešení na sérii testovacích příkladů.

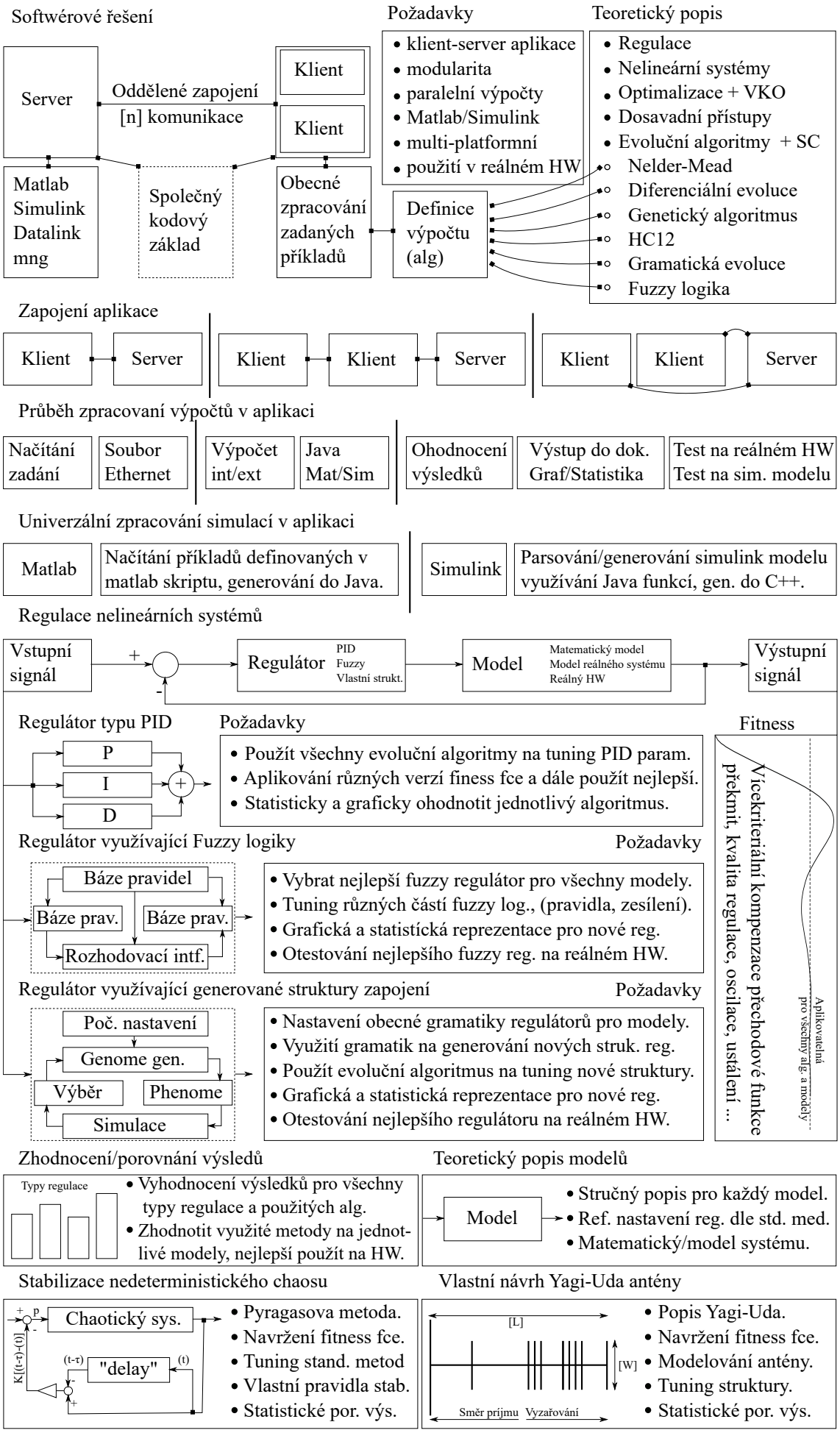
V dizertační práci jsou předložené mnohé výsledky, které autor nabyl v průběhu studia evolučních technik, umělé inteligence a regulace systémů. Představují souhrn aplikačních i teoretických výsledků autorových prací a prezentují hlavní proud zájmu autora v oblasti umělé inteligence a aplikování soft computingových metod. Aplikační část práce představuje návaznost na autorovu diplomovou práci na Ústavu automatizace a informatiky v oblasti distribuovaných výpočtů.



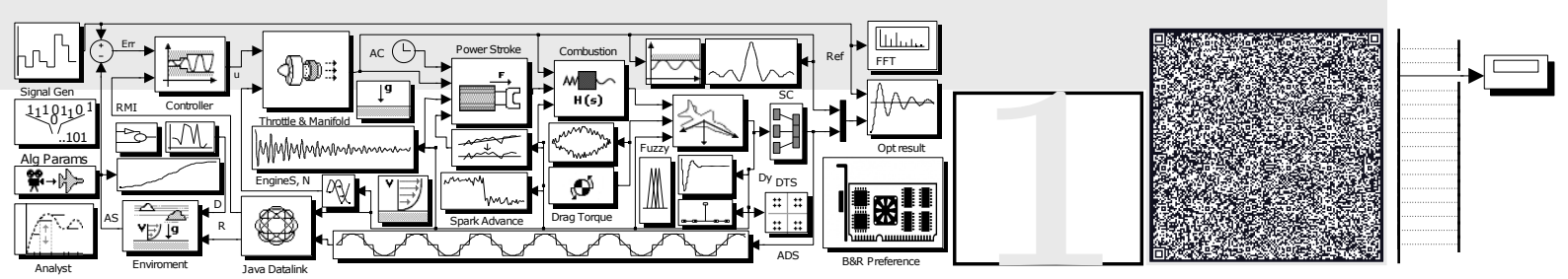
# CÍLE PRÁCE

Cílem této disertační práce je efektivní implementace pokročilých optimalizačních metaheuristik, návrh optimalizačních postupu pro vybrané inženýrské úlohy, návrh algoritmu řízení pro uvažované komplexní soustavy a analýza výsledků a zhodnocení řešení. Dále jsou cíle aplikované ve vytvoření univerzální multiplatformní aplikace typu server-klient pro otestovaní, vypočet a vyhodnocení výsledků z oblasti evolučních algoritmů s využitím nejnovějších metod soft computing na teorii řízení a stabilizace chaosu. Práce si klade za cíl prezentovat jednotlivé postupy na jednotlivých typových příkladech. Využití těchto metod je v praxi hojně využíváno a v této práci i odkazováno na různé způsoby řešení, každý z příkladů znázorňuje nové přístupy k této problematice a snaží se ukázat vhodný přístup řešení daného problému. Všechny způsoby jsou prezentovány na základě simulací a testovány na reálné soustavě. Konečné výsledky jsou pak zhodnoceny na základě statistických metod. K dosáhnutí tohoto cíle je zapotřebí napsaní vlastní optimalizační aplikace, která bude aplikovat všechny druhy algoritmů na všech typových příkladech. Aplikace má prezentovat způsob řešení této problematiky v praxi, kde neexistuje žádné univerzální řešení. Dílčí cíle této práce by se daly formulovat takto (obr: 1):

- Přiblížit obecně problematiku regulace nelineárních systémů.
- Obecný popis využívaných evolučních algoritmů a provést rešerši dosavadních postupů.
- Vytvořit a otestovat programové nástroje schopné použít nastíněné postupy řešení, v programovém jazyce Java schopné zpracovat Matlab a Simulink simulace.
- Otestovat a rozšířit postupy při regulaci komplexních nelineárních soustav na předem daných příkladech.
- Otestovat a rozšířit postupy při stabilizaci logistické mapy a duffyngovy rovnice, obecně nedeterministického chaosu.
- Aplikovat poznatky evolučních algoritmů na návrh Yagi-Uda antény.
- Zpracovaní naměřených (simulovaných) hodnot.
- Porovnaní dosazených výsledků proti standardně používaným metodám.




**Obrázek 1:** Ideové schéma a formulace řešení problémů prezentovaných v disertační práci.




# KAPITOLA 1

# POZNÁMKY K TEORII ŘÍZENÍ



Kapitola poznámky k teorii řízení slouží jako velmi stručný úvod do problematiky řízení systémů s nelineárním prvkem, a tedy i popisu obecných prvků, které se používají při řízení těchto systémů, a je převzata z univerzitních textů a učebnic k teorii řízení a regulace nelineárních systémů. Smysl kapitoly je popsat systémy, na kterých bude dále prováděna následná optimalizace a velmi stručně představit standardní metody návrhu regulace, a tedy i slouží jako úvod do problematiky této práce.

Kapitola je členěna na několik základních částí, první je obecný popis systémů a představení nejvíce používaných nelineárních prvků. Celkový popis je založený na práci *Švarc Ivan*, *Matoušek Radomil*, *Šeda Miloš*, *Vítečková Miluše*: *Automatické řízení*[38], *Modrlák Osvald*: *Teorie automatického řízení 2: Studijní materiály*[37] a *Nise S. Norman*: *Control Systems Engineering*[39]. Systém s nelineárním prvkem je považován jako nelineární systém sám o sobě, a proto jsou zde i znázorněny metody linearizace nelineárních prvků, dle práce *Dorf C. Richard*, *Bishop H. Robert*: *Modern control systems*[40] a konkrétně Padého metoda z práce *Bandyopadhyay B.*, *Rao A.*, *Singh H.*: *On pade approximation for multivariable systems*[59]. Tato metoda je dále v této práci uplatňována na linearizaci dopravního zpoždění u jednoho testovacího systému.



Poslední část se zabývá samotnou regulací lineárního, popřípadě linearizovaného systému s využitím nejvíce používaného regulátoru typu PID. Popis soustavy s regulátorem a způsob jeho nastavení je převzatý z práce *Ogata Katsuhiko*: *Modern Control Engineering*[56], *Nise S. Norman*: *Control Systems Engineering*[39] a vlastní design je odvozeny od *Yum Li*, *Kiam Heong Ang*, *Chong G.C.Y.*: *PID control system analysis and design*[57].

## 1.1 Nelineární systémy

Lineární systémy jsou nejčastěji popsány lineární diferenciální rovnicí, příklad diferenciálního popisu sytému (vz: 1.1) popřípadě pomocí přenosové funkce (vz: 1.2). Tyto systémy jsou velice lehko řešitelné standardními metodami, ale ve skutečnosti se jedná jen o aproximaci systému a v reálném světě jsou všechny systémy nelineární. Nelineární systém je soubor prvků, kdy aspoň jeden je nelineární. Nelineární systémy se vyznačují těmito vlastnostmi podle *Švarc Ivan*, *Matoušek Radomil*, *Šeda Miloš*, *Vítečková Miluše*: *Automatické řízení*[38]:

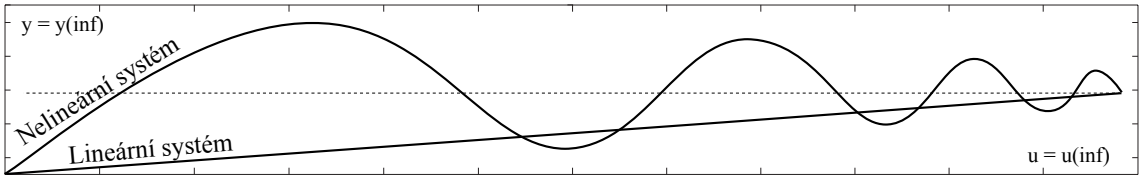


- statická charakteristika není lineární - viz obrázek (obr: 1.1),
- neplatí princip superpozice (výstup systému se lišší pro měnící se amplitudu vstupního signálu),
- rovnovážný stav mimo počátek souřadnic,
- počáteční podmínky mají vliv na stabilitu systému,
- možnost vzniků auto oscilací (samobuzené kmity) pro jiné frekvence než při budícím signálu,
- při změně budícího signálu dochází ke skokovým změnám amplitudy výstupu,
- nejednoznačná závislost výstupní veličiny na vstupní *Modrlák Osvald*: *Teorie automatického řízení 2: Studijní materiály*[37].

Nelineární charakteristika nelze matematicky popsat v celém rozsahu vstupní veličiny jedinou rovnicí přímky *Švarc Ivan*, *Matoušek Radomil*, *Šeda Miloš*, *Vítečková Miluše*: *Automatické řízení*[38].

$$a_n y^{(n)} + a_{n-1} y^{(n-1)} + \cdots + a_1 y' + a_0 y = b_m u^{(m)} + \cdots + b_1 u' + b_0 u \ \ (m \leq n) \quad (1.1)$$

$$G(s) = \frac{b_m s^m + b_{m-1} s^{m-1} + \ldots b_1 s + b_0}{a_n s^n + a_{n-1} s^{n-1} \ldots a_1 s + a_0} \ \ (m \leq n) \quad (1.2)$$




**Obrázek 1.1:** Příklad statické charakteristiky.

Nelineární systémy s jedním výstupem a jedním vstupem jsou popsány dvěma způsoby. První je diferenciální rovnicí s nelineární částí a řádu $n$ příklad $(A)$, nebo $n$ nelineárních rovnic prvního řádu příklad $(B)$. Druhým způsobem je kombinace lineárního členu s nelineárním, který je popsaný statickou charakteristikou *Modrlák Osvald*: *Teorie automatického řízení 2: Studijní materiály*[37].

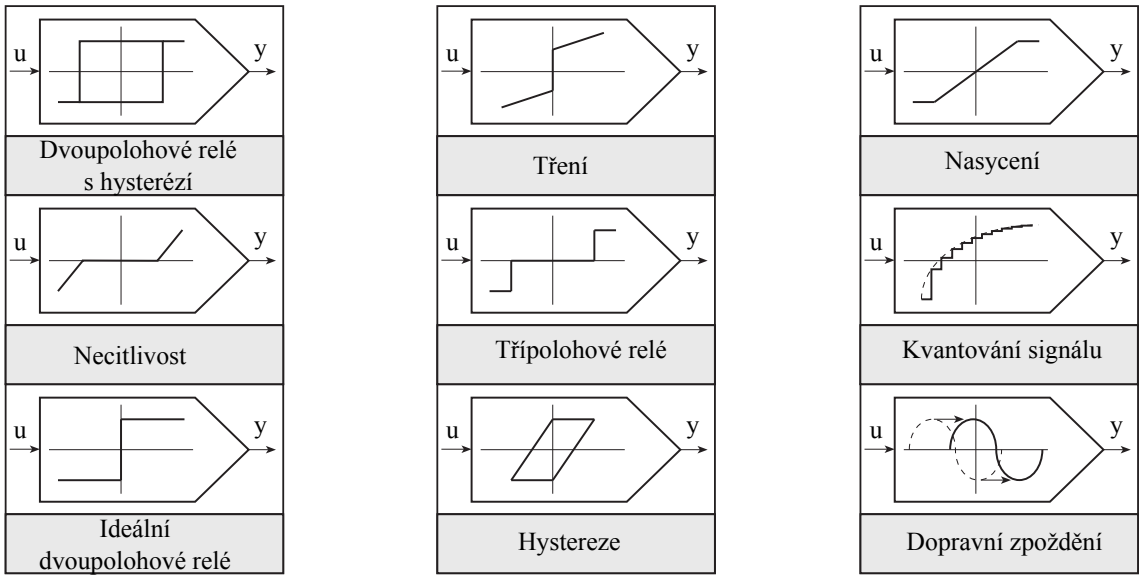
(A) $$y''' + 3(y')^2 y'' + y'\sqrt{y''} + 2y = u$$

(B)

| | | |
|---|---|---|
| $x_1 = y$ | $\dot{x_1} = x_2$ | $f_1(x_1, x_2, x_3, u)$ |
| $x_2 = y'$ | $\dot{x_2} = x_3$ | $f_2(x_1, x_2, x_3, u)$ |
| $x_3 = y''$ | $\dot{x_3} = -3(x_2^2)x_3 - x_2\sqrt{x_3} - 2x_1 + u$ | $f_3(x_1, x_2, x_3, u)$ |

### 1.1.1 Nelinearní členy

Obecně se nelinearity dělí na přirozené tzv. parazitní, které nejsou žádoucí ve výsledném systému, a nelinearity úmyslně vytvářené, jež tyto členy se využívají pro jednoduché regulátory a někdy ke zlepšení stability. Následující schémata obrázek (obr: 1.2) ukazují nejběžnější nelineární členy *Modrlák Osvald*: *Teorie automatického řízení 2: Studijní materiály*[37].




**Obrázek 1.2:** Běžné nelineární členy.

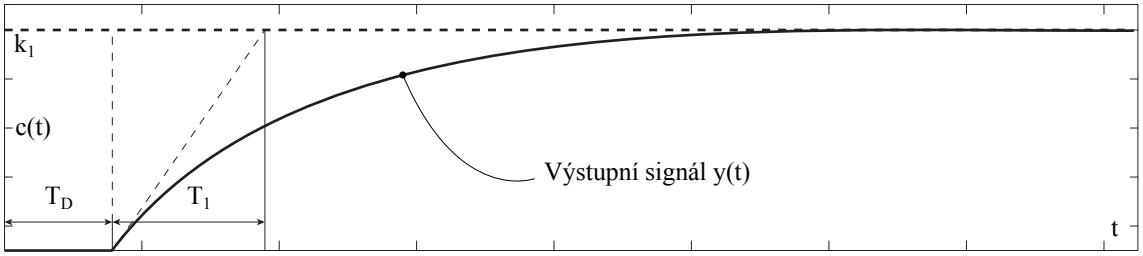


#### Dopravní zpoždění

Dopravní zpoždění je zpožděná reakce výstupní veličiny $y$ na změnu vstupní veličiny $u$, obrázek (obr: 1.3). Zpoždění se udává v čase $T_D$. Matematický zápis pomocí diferenciální rovnice a přenosu (vz: 1.3) a (vz: 1.4) na proporcionální systém se setrvačností 1. řádu. Nyquistova (Amplitudo-Fázová kmitočtová) charakteristika dopravního zpoždění se projevuje specificky kroužením okolo počátku a v kmitočtové oblasti fáze klesá k nekonečnu, proto se členy s dopravním zpožděním nazývají jako členy s minimální fází. V řízených systémech se často projevují vlastnosti dopravního zpoždění. Tento člen je obsažen v regulované soustavě nebo přímo v regulátoru (například číslicový regulátor). Pro využití syntézy regulačních obvodů pro lineární soustavy je nutno odstranit exponenciální funkci v přenosu (vz: 1.4), k odstranění se využívá aproximace například Taylorův polynom (vz: 1.7) nebo Padého rozvoj (vz: 1.8) *Dorf C. Richard*, *Bishop H. Robert*: *Modern control systems*[40]. Dopravní zpoždění je i jeden z příkladů systémů s nelineárním prvkem v této práci a jsou na něm prezentovány různé způsoby regulace takových systémů. Podrobný popis systému je obsažen v sekci (sekce: 4.2.2 - viz str. 49).

$$a_n y^{(n)}(t)+\cdots+a_1 y'(t)+a_0 y(t) = b_m u^{(m)}(t-T_D)+\cdots+b_1 u'(t-T_D)+b_0 u(t-T_D) \quad (1.3)$$

$$G(s) = \frac{b_m s^m \dots b_1 s + b_0}{a_n s^n + \dots a_1 s + a_0} e^{-T_D s} \quad (1.4)$$




**Obrázek 1.3:** Přechodová charakteristika proporcionálního systému se setrvačností 1. řádu s dopravním zpožděním.




**Obrázek 1.4:** Nyquistova charakteristika pro proporcionální systém se setrvačností 1. řádu s dopravním zpožděním (vlevo) a samotné dopravní zpoždění (vpravo).

Nasycení je limitní hodnota výstupu, které je možno dosáhnout $y_1$ (vz: 1.5). Tento typ nelinearity je důsledek technických nebo fyzikálních jevů, kdy akční zásah nemůže nabývat nekonečných hodnot.

$$y_1 = \begin{cases} e_a & \text{pro } u \geq a \\ (e_a/a) \times u & \text{pro } b < u < a \\ e_b & \text{pro } u \leq b \end{cases}, y_2 = \begin{cases} (u-a) \times \tan\alpha & \text{pro } u \geq a \\ 0 & \text{pro } b < u < a \\ (u+b) \times \tan\beta & \text{pro } u \leq b \end{cases} \tag{1.5}$$

Časová oblast, v které výstupní hodnota nereaguje na vstup se nazývá necitlivost $y_2$ (vz: 1.5). Vyskytuje se hlavně v mechanických převodech. Časový bod, v kterém je skoková hodnota výstupní veličiny, je nelinearita typu tření $y_1$ (vz: 1.6). Simuluje se tímto účinek suchého i viskózního tření. Stav výstupu závisí na předešlém stavu a historii vstupů $y_2$ (vz: 1.6). Při hysterezi je výstup v cyklickém opakování a příkladem výskytu je závislost magnetické indukce na intenzitě magnetického pole.

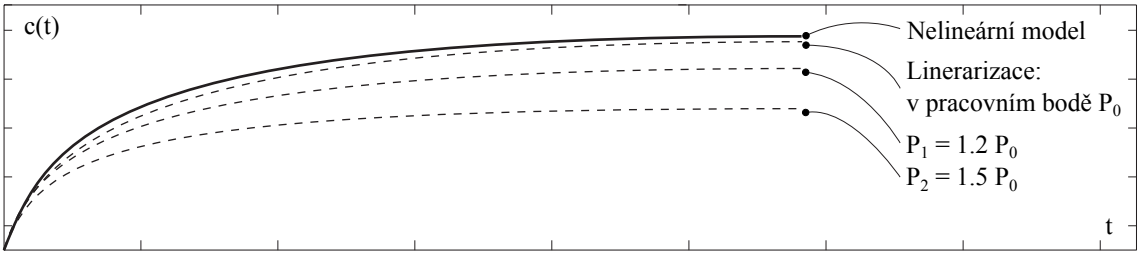
$$y_1 = \begin{cases} e_a + u \ \times \tan\alpha & \text{pro } u \geq a \\ e_b - u \ \times \tan\beta & \text{pro } u < b \end{cases}, y_2 = \begin{cases} e_a & \text{pro } u > a \\ e_b & \text{pro } u < b \\ \leftarrow e_a \rightarrow e_b & \text{pro } b \leq u \geq a \end{cases} \tag{1.6}$$

### 1.1.2 Linearizace

Základním principem převodu nelineárního systému na lineární je využití metod linearizace. Často se využívá metoda rozkladu nelineární funkce v Talylorovu nekonečnou řadu (vz: 1.7) pro pracovní bod $P$ a následné omezení na první dva členy rozvoje. Tento postup je tzv. aproximace Taylorovým polynomem prvního stupně. Pro co největší přesnost metody se musí vhodně zvolit pracovní bod, ke kterému se vypočítá aproximace. Pracovní bod se volí nejčastěji v oblasti očekávaného chodu systému. V pracovním bodě je aproximace k nelineární funkci nejpřesnější, čím dále jsme od tohoto bodu, tím narůstá nepřesnost. Další možností v linearizaci je využití

metody Padého rozvoje (vz: 1.8), která je nejlepší aproximační metodou pomocí racionální funkce daného stupně. Je často úspěšná i v případech, kdy Taylorův rozvoj nekonverguje. Při této aproximaci je interpolující funkce $R_{m,n}$ definována jako podíl polynomů *Dorf C. Richard*, *Bishop H. Robert*: *Modern control systems*[40].

$$f(x) = \underbrace{f(p) + \frac{f'(p)}{1!}(x-p)}_{linearizace} + \frac{f''(p)}{2!}(x-p)^2 \cdots = \sum_{k=0}^{\infty} \frac{f^{(k)}(p)}{k!}(x-p)^k \tag{1.7}$$



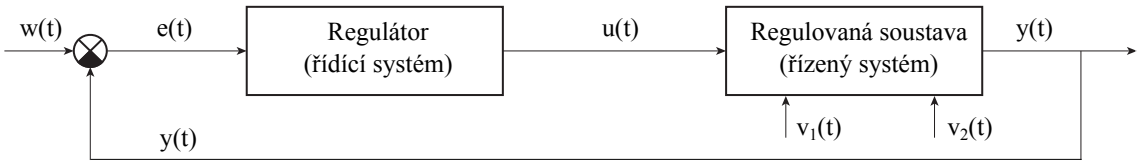

**Obrázek 1.5:** Vliv posunutí pracovního bodu na linearizaci.

$$R_{m,n}(X) = \frac{\sum_{j=0}^{m} a_j x^j}{1 + \sum_{k=0}^{n} b_k x^k}, \text{ pro } m \geq 0 \text{ a } n \geq 0 \tag{1.8}$$

## 1.2 Regulační systém

Regulační systém se skládá z regulované soustavy, tedy procesu, který chceme regulovat, a systému, pomocí kterého budeme provádět regulaci tzv. řídicí systém. Celý proces regulace spočívá v ustálení hodnoty výstupu $y$ regulované soustavy na požadované hodnotě $w$. Obrázek (obr: 1.6) znázorňuje jednoduché schéma řízení se zpětnou vazbou. Zpětná vazba sleduje aktuální stav systému a porovnává ji s požadovanou hodnotou. Výsledkem je chyba regulace $e$ (vz: 1.9), tato hodnota je vstupem do regulátoru. Regulátor zpracovává chybu a výstupem je akční zásah $u$ do regulované soustavy, který může být ovlivňován i nežádoucími poruchovými veličinami $v$. Ke správnému chodu regulace je zásadní zvolit vhodný typ regulátoru a nastavení jeho hodnot *Nise S. Norman*: *Control Systems Engineering*[39].

$$e(t) = w(t) - y(t) \tag{1.9}$$




**Obrázek 1.6:** Schéma regulačního systému s příkladem působení poruchové veličiny.

### 1.2.1 Regulátor typu PID

Nejznámějším a nejvíce využívaným regulátorem je typ (PID), tedy regulátor s proporcionální, integrační a derivační složkou. Tato práce se zabývá optimalizací těchto

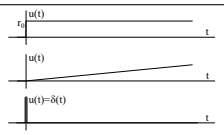
parametrů k dosažení co nejlepšího výsledku regulace. Matematický popis regulátoru je dle tabulky (tab: 1.1) a celkový popis regulátoru se může zapsat dle časových konstant (vz: 1.11), nebo dle konstant zesílení (vz: 1.10). Tato práce používá první verzi zápisu (PID) a tedy optimalizaci v časové oblasti regulátoru, viz dále v implementační části práce.

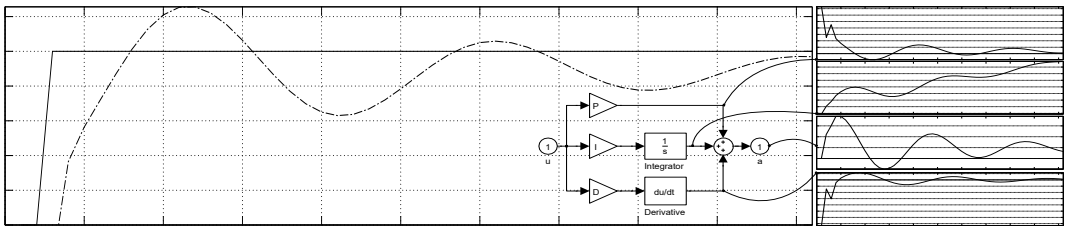
$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{d}{dt} e(t) \tag{1.10}$$

$$u(t) = r_0 \left( e(t) + \frac{1}{T_i} \int_0^t e(\tau)d\tau + T_d \frac{d}{dt} e(t) \right) \tag{1.11}$$



Přechodové charakteristiky, pro jednotlivé parametry regulátoru a dynamické vlastnosti těchto parametrů na typovém přikladu. Parametr $P$ prezentuje šířku pásma proporcionality a je vyjádřená v měřených jednotkách, $I$ je Integrační parametr a eliminuje ztráty regulované soustavy neboli trvalou regulační odchylku. Derivační hodnota se uplatňuje při rychlých změnách žádané hodnoty.

| ⊞ | Rovnice | Přenos $G_r(s)$ | Přechodová charakteristika |
|---|---|---|---|
| P | $u = r_0 e$ | $r_0$ |  |
| I | $u = r_{-1} \int e dt$ | $\frac{r_{-1}}{s}$ | |
| D | $u = r_1 e \frac{d}{dt}$ | $r_1 s$ | |



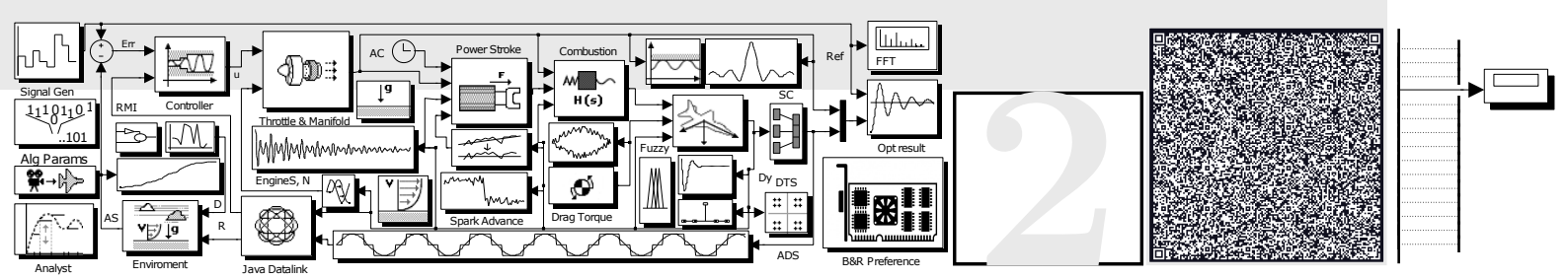

**Tabulka 1.1:** Matematický popis PID a dynamické vlastnosti spojitých regulátorů.

### 1.2.2 Ziegler-Nichols

Ziegler-Nichols (ZN) je velmi rozšířená empirická metoda nastavení regulátoru typu (PID). Principem metody je přivést regulovaný systém s regulátorem na hranici stability, to znamená výstupní veličina periodicky osciluje okolo žádané hodnoty s konstantní amplitudou. Tento stav lze nalézt i numericky tj. nalézt polohu kořenu na hranici stability a odečíst velikost zesílení prvku $P$ ($r_{0k}$), periodu kmitání $T_k$ a potom dle tabulky (tab: 1.2) vypočítat optimální nastavení regulátoru. Na hranici stability se dostaneme metodou odstranění integračního a derivačního členu, tedy $T_i \to \infty$, $T_d \to 0$ respektive $r_1 \to 0$, $r_1 \to 0$ a zjištěním kritického zesílení $r_o k$.

| ⊞ | $r_0$ | $T_i$ | $T_d$ |
|---|---|---|---|
| P | $0.5r_{0k}$ | – | |
| PI | $0.45r_{0k}$ | $0.83T_k$ | – |
| PD | $0.4r_{0k}$ | – | $0.05T_k$ |
| PID | $0.6r_{0k}$ | $0.5T_k$ | $0.12T_k$ |

**Tabulka 1.2:** Seřízení PID podle Ziegler-Nicholse.




# KAPITOLA 2

# OPTIMALIZACE A APLIKACE



Kapitola optimalizace a aplikace má ve své podstatě DOUFAM stejný význam jako předchozí kapitola zabývající se teorií řízení a popisuje jednotlivé metody optimalizace používaných v této práci a jejich stručnou charakteristiku. Kapitola se zaměřuje na hlavní část práce, a to na využití optimalizačních metod pro řízení regulovaného systému, zde aplikované na řízení nelineárních systémů. Obecný popis optimalizace vychází z prací *Hamming Richard*: *Numerical methods for scientists and engineers*[1], *Fletcher R.*: *Practical methods of optimization*[62] a jen stručně rozděluje nejčastěji používané optimalizační techniky.

Celkový popis v kapitole je rozdělený na stručný úvod do problematiky optimalizace a popisu optimalizace vícekriteriálních systémů. V této práci jsou z hlediska vícekriteriálních vlastností chápány rozličné parametry výstupního signálu z řízeného systému, například čas regulace, velikost překmitu, atd. Zde obsažený stručný popis je derivací prací *Marler R. T.*, *Arora J. S.*: *Survey of multi-objective optimization methods for engineering*[4] a *Kalyanmoy Deb*: *Multi-objective optimization using evolutionary algorithms*[60].



Hlavní část kapitoly je věnována popisu a rozdělení evolučních algoritmů, jakožto obecného optimalizačního algoritmu práce. Popis zahrnuje klady i zápory zvolené metody a důvody proč jsou evoluční algoritmy vhodné k aplikaci na teorii řízení a jejich přínos v této oblasti do budoucna. Evoluční algoritmy a soft computing metody jsou popsány na základě prací *Zelinka Ivan*, *Oplatková Zuzana*, *Šeda Miloš*, *Ošmera*

*Pavel*, *Včelař František*: *Evoluční výpočetní techniky: Principy a aplikace*[3], *Jin Y.*: *A Definition of Soft Computing - adapted from L.A. Zadeh*[11], *Fakhri Karray*, *De Silva W. Clarence*: *Soft Computing and intelligent systems design: theory, tools and applications: theory and applications*[63] a hlavní výhoda těchto algoritmů spočívá ve velmi spolehlivém hledání globálního extrému cílové funkce $f$ (na rozdíl od geometrických optimalizačních metod).

Po úvodu do problematiky evolučních algoritmů a soft computing metod je dále kapitola věnována rozsáhlejšímu popisu jednotlivých genetických algoritmů, které se využívají v této práci. V první části sekce je popis základního horolezeckého algoritmu, jako podklad pro další popis následujících algoritmů (konkrétně algoritmu HC12) dle práce *Russell Stuart*, *Norvig Peter*: *Artificial intelligence: A modern approach*[64]. V rámci obecné sekce je obsažen i popis simplexové metody Nelder-Mead dle práce *Nelder John*, *Mead Roger*: *A simplex method for function minimization*[9], která je dále používána jako referenční algoritmus k porovnání dosažených výsledků z důvodu popularity algoritmu a jeho hojného používání v praxi.



Dále jsou v kapitole popsány hlavní evoluční algoritmy používané při samotných výpočtech. Prvním z nich je genetický algoritmus dle práce *Haupt L. Randy*, *Haupt Sue Ellen*: *Practical Genetic Algorithms*[2], *Mitchell*, *Melanie*: *An introduction to genetic algorithms*[6] a prezentace principu algoritmu na stručném popisu. Genetický algoritmus byl jeden z prvních evolučních algoritmů *Holland*, *H. John*: *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*[7] a výsledky z tohoto algoritmu jsou porovnávány z důvodu jeho velké popularity ve výpočetních prostředcích. Druhým z evolučních algoritmů je diferenciální evoluce dle práce *Storn Rainer*, *Price Kenneth*: *Differential Evolution – a Simple and Efficient Heuristic for Global Optimization*[16] s popisem algoritmu a nejčastějším nastavením parametrů. Diferenciální evoluce je k porovnání výsledků s genetickým algoritmem zahrnuta, protože diferenciální evoluce na rozdíl od genetického algoritmu dokáže daný problém vyřešit rychleji, přesněji a dokáže pracovat s libovolným typem čísla (integer, float, binary) nebo jejich kombinacemi, což u genetických algoritmů činí potíže, jinak je však jejich struktura velice podobná. Předešlé články na toto téma dokládají schopnosti diferenciální evoluce *Tvrdík Josef*: *Evoluční Algoritmy*[12], *Lampinen Jouni*: *Multi-Constrained Nonlinear Optimization by the Differential Evolution Algorithm*[13], *Feoktistov Vitaliy*: *Differential Evolution: In Search of Solutions*[15] najít optimální výsledek s velkou přesností hledaných parametrů. Posledním algoritmem k hledání optimálních parametrů v této práci je algoritmus HC12 z práce *Matoušek Radomil*: *HC12: Highly Scalable Optimization Algorithm*[19], který je hlavní optimalizační algoritmus využívaný k řešení optimalizačních problémů a jeho výsledky jsou prezentovány pro každý zvolený příklad dále v sekci (aplikace výsledků na testovacích soustavách). Algoritmus HC12 je zvolený jako hlavní metoda z důvodu jeho jednoduché škálovatelnosti, jednoduché implementaci a rychlé schopnosti dosáhnout optimálního výsledku, jak dokazují předešlé práce na toto téma *Matoušek Radomil*, *Žampachová Eva*: *Promising GAHC and HC12 algorithms in global optimization tasks*[18], *Matoušek Radomil*: *Pokročilé*

*metody počítačové intligence*[61].

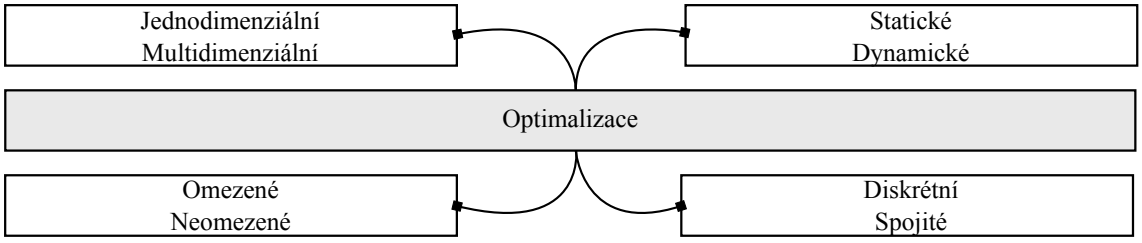
Dva poslední algoritmy jsou obsaženy z důvodu širší optimalizace výsledných regulátorů z hlediska zaměření na úpravu samotné struktury regulátoru (funkce) oproti vyhledávání nejvhodnějších dílčích parametrů v pevné struktuře. Hlavním zástupcem v problematice je gramatická evoluce dle práce *Ryan Conor*, *Collins J.J.*, *O'Neill Michael*: *Grammatical Evolution: Evolving Programs for an Arbitrary Language*[22] a na základě *Langdon B. William*, *Poli Riccardo*: *Foundations of Genetic Programming*[5]. Gramatická evoluce je velice mocný nástroj na řešení optimalizačních problémů a je velice vhodná i kvůli její jednoduché aplikovatelnosti na různé druhy problémů od hledání struktury funkce až k aplikaci v různých odvětvích vědy. Předchozí práce *O'Neill Michael*, *Rayn Conor*: *Grammatical evolution : Evolution Automatic Programming in an Arbitrary Language*[20], *Dempsey Ian*, *O'Neill Michael*, *Brabazon Anthony*: *Foundations in Grammatical Evolution for Dynamic Environments*[21] prezentují její potenciál na toto téma a její vhodnost k řešení problémů v teorii řízení. Tato práce využívá gramatické evoluce k dalšímu porovnání výsledků a rozšíření nastavení regulátoru o optimalizaci přes jiné druhy algoritmů (konkrétně HC12). Poslední část popisovaných algoritmů je věnována stručnému popisu fuzzy logiky dle práce *Zadeh A. Lotfi*: *Fuzzy sets*[27]. Fuzzy logika je vhodný nástroj k regulaci a schopnosti vytvářet velkou variabilitu regulovaných prvků, jak prezentovaly práce *Sugeno Michio*: *An introductory survey of fuzzy control*[58], *Modrlák Osvald*: *Fuzzy řízení a regulace: Studujní materiály*[24], *Passino M. Kevin*: *Fuzzy Control*[25], *Jantzen Jan*: *Foundations of Fuzzy Control*[26]. V této práci je rozvíjena možnost zapojení fuzzy logiky při využitím poznatků při regulaci klasických regulátorů pomocí předešlých metod, konkrétně se zde využívá aplikaci fuzzy logiky na strukturu PID regulátoru.



Poslední část kapitoly stručně shrnuje dosavadní vývoj v oblasti požívání soft computingových metod v oblasti teorii řízení a prezentuje další možnosti kam se ubírají jiné práce v oboru.

Optimalizace je proces, při kterém hledáme takové nastavení vstupních hodnot parametrů, pro které účelová funkce nabývá optimálních hodnot (minimum / maximum), přičemž může jít o exaktní matematický výpočet, numerický postup nebo metaheuristiku. Optimalizace je vhodná pro řešení mnoha úloh, například kombinatorická optimalizace, problém obchodního cestujícího, nebo pro řešení fyzikálních jevů, kde se řeší velikost energie systému při rovnovážném stavu. Optimalizační problémy je možné například rozdělit do následujících čtyř kategorií dle obrázku (obr: 2.1) *Haupt L. Randy*, *Haupt Sue Ellen*: *Practical Genetic Algorithms*[2], který představuje jen velice hrubé rozdělení optimalizačních problémů. Další rozdělení můžeme najít v různých pracích, ale pro účely této kapitoly disertační práce, která je jen velice stručný souhrn informací o optimalizačních postupů, je dostatečné.

$$x_{opt} = arg \underset{x \epsilon D}{\text{opt}} f(x) \tag{2.1}$$




**Obrázek 2.1:** Příklad druhů optimalizačních problémů.

### Vícekriteriální optimalizace

Má-li na výsledek optimalizačního procesu působit současně více hodnotících - účelových funkcí, tak se jedná o vícekriteriální optimalizaci (VKO), nebo také vícekriteriální programování, matematický zápis (VKO) (vz: 2.2).



$$\mathbf{x}_{opt} = arg \underset{\mathbf{x}\epsilon D}{\text{opt}} f(\mathbf{x}) \tag{2.2}$$

$$g_j(x) \leq 0, j = 1, 2 \ldots, m h_l(x_e) = 0, l = 1, 2, \ldots, e \tag{2.3}$$

Existuje mnoho problémů pro využití (VKO), například finance, technický design nebo všude tam, kde se bere v úvahu více než jedno kritérium optimalizace. Pro netriviální vícekriteriální problém nejde identifikovat jediné řešení, které obsahuje nejlepší hodnotu pro všechny kritéria. Cíl (VKO) je najít takové řešení, které nejde nahradit lepším řešením bez toho, aniž by nedošlo ke zhoršení jednoho z kritérii. Takové řešení se nazývá Parteto eficientní nebo také nedominantní (kompromisní) řešení, proto výsledek (VKO) problému je až nekonečno Pareto bodů *Marler R. T.*, *Arora J. S.*: *Survey of multi-objective optimization methods for engineering*[4].

**Definice** Pareto eficientní: Bod $x^* \in X$ je pareto eficientní jestliže neexistuje další bod $x \in X$, pro který platí $F(x) \leq F(x^*)$ a $F_i(x) \in F_i(x^*)$ pro aspoň jednu hodnotící funkci.

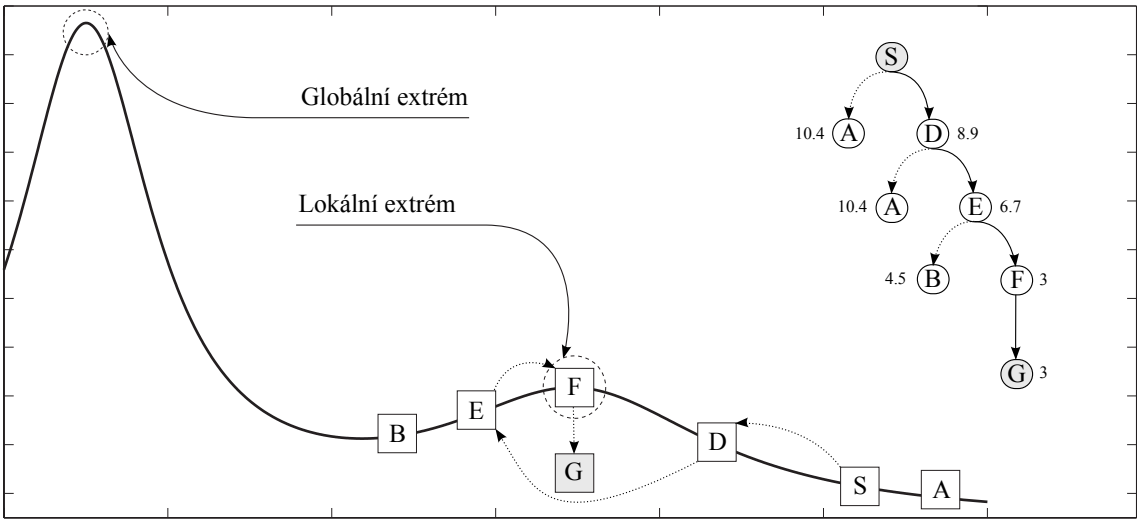
K řešení VKO problému existuje mnoho metod, které se dají rozdělit do těchto kategorií:

- S apriorními informacemi: je skupina metod, které mají informace před vlastním výpočtem kompromisního řešení. Tato skupina převádí VKO na úlohu jedné nebo několika jedno kriteriálních úloh. Příkladem metod může být Lexigrafická metoda, váhové metody nebo cílové programování.
- Průběžnou informací: je využití interakce mezi rozhodovatelem a analytikem pro převod lokální informace z dosaženého průběžného výsledku. Na základě dodatečných řešení se hledá nové průběžné řešení, dokud rozhodovatel nerozhodne o kompromisním řešení. Příkladem je Nashova metoda.
- S aposteriorní informací: metody vycházejí z množiny už hotových kompromisních řešení a dle nich a dodatečných informací se nacházejí další řešení. Příkladem je NBI metoda.
- Evoluční algoritmy: pomocí vhodného sestavení fitness funkce, která obsahuje všechna kritéria problému.

## 2.1 Princip horolezeckého algoritmu

Horolezecký princip (HC) patří do skupiny informovaných metod prohledávání a pod oblast gradientních algoritmů (gradientní algoritmus vždy expanduje ten bod, který byl doposud vyhodnocen pomocí hodnotící funkce jako nejlepší, a vyhodnocuje jeho následovníky). Horolezecký princip spočívá v systematickém prohledávání stavového prostoru, prostoru možných, respektive přípustných řešení. První krok je volba inicializačního bodu (většinou volen náhodně), ke kterému se vygeneruje oblast nových bodů, tato část je jádro každého algoritmu založeného na tomto principu. V těchto nových bodech se zkoumá, jestli došlo ke zlepšení problému. Když je zvolen nový výchozí bod k expanzi, tak se celý proces opakuje. Horolezecký princip začne na ne-zcela optimálním řešením a pomocí malých změn se dostává k lepším výsledkům. Ukončovací podmínky mohou být například omezení počtu iterací nebo nenalezení lepšího výsledku ve stávající iteraci. Jednoduchý postup principu je znázorněn na obrázku (obr: 2.2), *Minář Petr*: *Distribuované výpočty s využitím technologie ActionScript*[10].






**Obrázek 2.2:** Příklad průběhu horolezeckého algoritmu.

## 2.2 Nelder-Mead simplexová metoda

Nelder-Mead (NM) je optimalizační metoda často označovaná jako nelineární simplexová metoda, která je dobře využitelná při optimalizaci vícedimenzionálních nelineárních úloh. Metoda nepočítá gradient funkce, ale svůj postup odvozuje z funkčních hodnot, tedy se její využití neomezuje pouze na diferencovatelné problémy. Problém se dvěma parametry je simplex ve tvaru trojúhelníku. Metoda je založena na vytváření a porovnávání nových vrcholů simplexu, kdy je vždy nahrazován nejhorší bod simplexu nově vygenerovaným. Metoda byla poprvé představena v publikaci *Nelder John*, *Mead Roger*: *A simplex method for function minimization*[9]. Existuje mnoho variant (NM) odvozených z typu problému *Mathews H. John*, *Kurtis K. Fink*: *Numerical Methods Using Matlab*[8].

Obecný princip metody je velice jednoduchý, na začátku se vygeneruje počáteční

simplex s vrcholy a seřadí se od nejlepší po nejhorší dle hodnotící funkce $B$ [Best], $G$ [Good], $W$ [Worst]. Pomocí logiky znázorněné na algoritmu (alg: 1) se postupně vypočítají vrcholy $M$ [Midpoint], $R$ [Reflection], $E$ [Expansion], $C_n$ [Contraction], $S$ [Shrink] a jestli je nějaký vrchol lepší než nejhorší $W$ tak ho nahradí ve výchozím trojúhelníku. Celý proces se opakuje, dokud není splněna ukončovací podmínka.

$$B = (x_1, y_1), G = (x_2, y_2) \text{ a } W = (x_3, y_3) \text{ (pro dva parametry)} \tag{2.4}$$

$$M = \frac{B+G}{2} = \left(\frac{x_1+x_2}{2}, \frac{y_1+y_2}{2}\right) \tag{2.5}$$

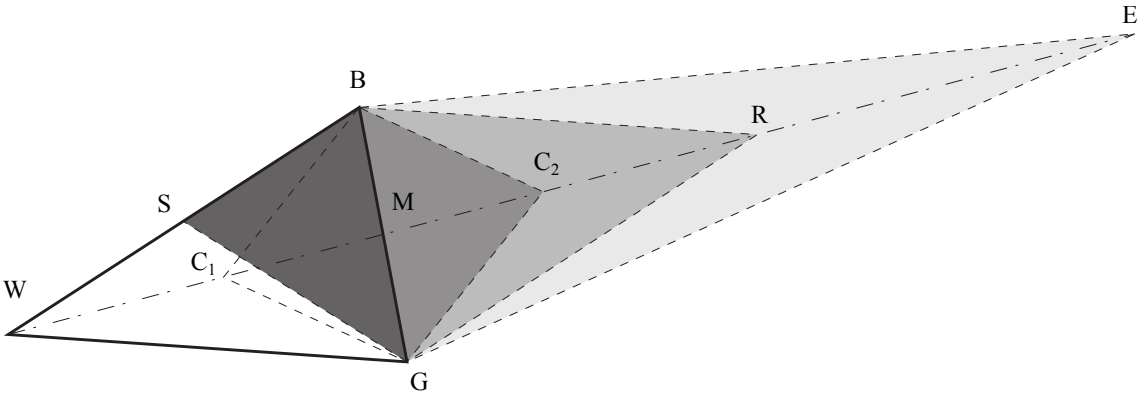
$$R = M + (M - W) = 2M - W \tag{2.6}$$

$$E = R + (R - M) = 2R - M \tag{2.7}$$

$$C_1 = \frac{M+R}{2} \text{ a } C_2 = \frac{W+M}{2} \tag{2.8}$$

$$S = \frac{B+W}{2} \tag{2.9}$$






**Obrázek 2.3:** Nelder-Mead vytváření nových vrcholů.

## 2.3 Soft computing metody

Obsah pojmů soft computing (SC), počítačová inteligence (Computational Intelligence, CI) a umělá inteligence (Artificial Intelligence, AI) není zcela jednotně chápán. Rovněž popis heuristických metod, či dále optimalizačních metaheuristik ve vztahu k SC, CI, a AI nabízí více pohledů *Matoušek Radomil*: *Pokročilé metody počítačové intligence*[61]. Soft computing je odvětví počítačových věd, které vzniklo začátkem devadesátých let a dnes je velmi populární k řešení složitých soustav. Přístup k (SC) je odvozeno od lidského myšlení a přírodních procesů. Základní idea pro vznik (SC) je odvozena z mnoha zdrojů, jedním z hlavních je *Zadeh A. Lotfi*: *Fuzzy sets*[27], *Holland, H. John*: *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*[7] a mnoha dalších. Velkou výhodou (SC) metod je jejich jednoduchost a možnost aplikace pro paralelní výpočty. Existuje mnoho metod (SC), ale ne každá se hodí

pro všechny druhy problémů, proto je vhodné vzájemně metody kombinovat pro dosažení nejlepšího výsledku. Nejpoužívanější metody (SC) jsou Fuzzy logika (FL), Neuronové sítě (NT), Strojové učení (SU), Evoluční algoritmy (EA) a další *Jin Y.*: *A Definition of Soft Computing - adapted from L.A. Zadeh*[11].

Skupina evolučních algoritmů (EA) jsou stochastické vyhledávací metody využívané k optimalizaci, které se v jádře pokoušejí napodobit proces přírodní evoluce a jsou velice jednoduché k softwarové implementaci. Na počátku (EA) nemá žádné informace o prohledávaném prostoru a závisí zcela na informacích od operátorů (EA), jako je třeba (reprodukce, křížení, mutace). Pomocí těchto a dalších procesů se snaží EA vylepšovat, popřípadě reprodukovat populaci potomků a ve finále najít nejlepší (optimální) výsledek dle zadané hodnotící funkce. (EA) jsou schopné velmi dobře najít optimální výsledky $i$ v $n$ dimensionálním prostoru bez obecných potíží, jako mají třeba gradientní algoritmy (zaseknutí v lokálním minimu). Některé výhody (EA) zahrnují *Haupt L. Randy*, *Haupt Sue Ellen*: *Practical Genetic Algorithms*[2]:



- všechny EA jsou globální optimalizační metody,
- optimalizace pro spojité i diskrétní parametry,
- nepotřebují výpočet derivací,
- pracují dobře s multidimenzionálním prohledáváním,
- jsou vhodné k paralelním výpočtům,
- nemají tendenci zaseknutí v lokálním minimu i ve velmi komplexním prohledávacím prostoru,
- jsou vhodné pro velkou škálu problémů, od analytických funkcí, po experimentální data.

### 2.3.1 Obecná aplikace

Aplikace (EA) v teorii řízení je velice výhodná hlavně v oblasti nelineárních systémů. V analytickém řešení těchto systémů se často musí systém linearizovat, což znamená zavedení určité nepřesnosti do výsledného systému k nalezení parametrů regulátoru. Tento způsob u (EA) odpadá, protože při optimalizaci parametrů můžeme pracovat přímo se skutečným matematickým modelem soustavy a odkazovat se jen na výslednou charakteristiku ve formě fitness funkce. Další výhoda (EA) je možnost uplatnění vícekriteriální optimalizace, kde se může sledovat například výsledný přechodový děj a akční zásah od regulátoru. Experimentální testy a příklady z praxe dokazují, že použití stochastických vyhledávacích metod k nalezení optimálního regulátoru přináší povětšinou lepší výsledky oproti využití analytických metod řešení *Matoušek Radomil*, *Minář Petr*: *Optimalizace parametrů PSD regulátoru pomocí algoritmu HC12*[47], *Matoušek Radomil*, *Minář Petr*, *Lang Stanislav*, *Pivoňka Petr*: *HC12: Efficient Method in Optimal PID Tuning*[48].

### 2.3.2 Genetický algoritmus

První z rodiny (EA) je genetický algoritmus (GA), který se snaží chovat jako reálná genetická evoluce, tedy je inspirovaný Darwinovu teorií evoluce - přežití nejsilnějšího

a Mendelovou teorií selekce. V přírodě jedinci mezi sebou bojují o přežití, a to stejné funguje i v (GA), který je odvozen od přírodní selekce, mutace a křížení. Metoda (GA) je adaptivní stochastický algoritmus a patří mezi jedny z nejpopulárnější EA. K dnešní době existuje mnoho mutací (GA), které dramaticky vylepšují původní návrh. Algoritmus pracuje s populací jedinců (Mating pool), na které se provádí všechny operace GA. Každý jedinec se nazývá chromozom, parametry v chromozomu jsou tady nazvány jako geny a mají binární hodnoty *Zelinka Ivan*, *Oplatková Zuzana*, *Šeda Miloš*, *Ošmera Pavel*, *Včelař František*: *Evoluční výpočetní techniky: Principy a aplikace*[3]. Algoritmus byl poprvé představený v publikaci *Holland*, *H. John*: *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*[7].



$$chromozom = \underbrace{101100}_{gen_1}\underbrace{111000}_{gen_2}\ldots\underbrace{001001}_{gen_{N_{par}}} \tag{2.10}$$

$$gen = [bit_1, bit_2, bit_3 \ldots bit_{N_{bit}}]\ bit_i \in \{0,1\}^n_{bit} \tag{2.11}$$

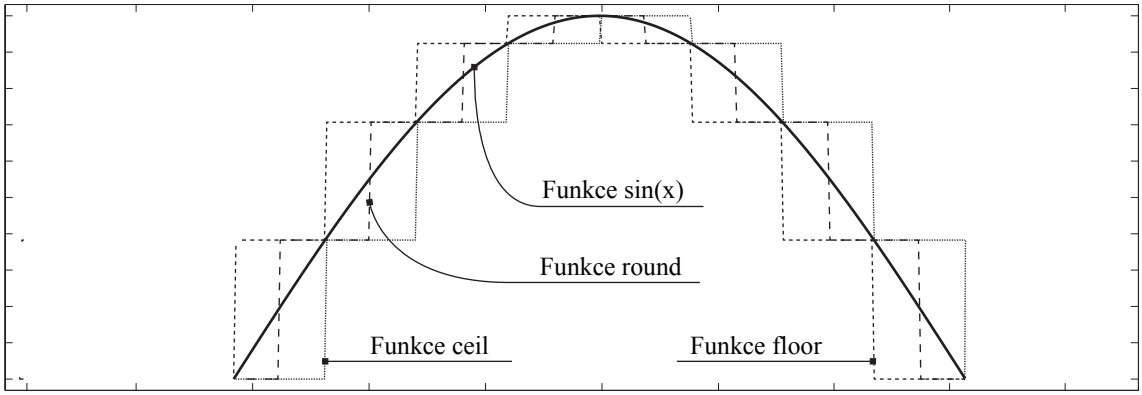
Převedení spojitého parametru na binární číslo, se musí dekódovat v přesnosti genu. Přesnost genu se udává podle jeho bitové délky. Příklad gen má $Nbit = 3$ a spojitý parametr nabývá hodnoty $< 0, 1 >$, tedy parametr může nabývat 8 hodnot v rozsahu $< 0, 0.125 >$. Pro lepší představu je znázorněno dekódování v tabulce (tab: 2.1) pro tři druhy zaokrouhlení (floor, ceil a střední hodnota). V první části tabulky jsou všechny hodnoty parametru a v druhé jsou dekódovány dle určité hodnoty spojité veličiny. Obrázek (obr: 2.4) znázorňuje vliv binárního dekódování na funkci $sin(x)$. Vzorec (vz: 2.12) respektive (vz: 2.13) je výpočet skutečné hodnoty genu z binárního kódování.

$$p_{norm} = \sum_{m=1}^{N_{bit}} gen[m]^{-m} + 2^{-(M+1)} \tag{2.12}$$

$$p_{gen} = p_{norm}(p_{hi} - p_{lo}) + p_{lo} \tag{2.13}$$

| Gen | Dec | Floor | Ceil | Střed | Param | Gen | Floor | Ceil | Střed |
|---|---|---|---|---|---|---|---|---|---|
| 000 | 0 | | 0.125 | 0.063 | 0.510 | 100 | 0.500 | 0.625 | 0.563 |
| 001 | 1 | 0.125 | 0.250 | 0.188 | 0.760 | 110 | 0.750 | 0.875 | 0.813 |
| 010 | 2 | 0.250 | 0.375 | 0.313 | 0.120 | 000 | 0 | 0.125 | 0.063 |
| 011 | 3 | 0.375 | 0.500 | 0.438 | 0.330 | 010 | 0.250 | 0.375 | 0.313 |
| 100 | 4 | 0.500 | 0.625 | 0.563 | 0.930 | 111 | 0.875 | 1 | 0.938 |
| 101 | 5 | 0.625 | 0.750 | 0.688 | 0.430 | 011 | 0.375 | 0.500 | 0.438 |
| 110 | 6 | 0.750 | 0.875 | 0.813 | 0.010 | 000 | 0 | 0.125 | 0.063 |
| 111 | 7 | 0.875 | 1 | 0.938 | 0.620 | 101 | 0.625 | 0.750 | 0.688 |

**Tabulka 2.1:** Příklad kódování do binárního čísla pro tři bity.




**Obrázek 2.4:** Binární verze sin(x) dle použité funkce zokrouhlení.

### Selekce



Selekce je první část (GA), při která se vybírají jedinci pro reprodukci potomků. Existuje mnoho variant selekce většinou založených na kvalitách jedince dle fitness funkce, zde jsou vyjmenované čtyři základní typy.

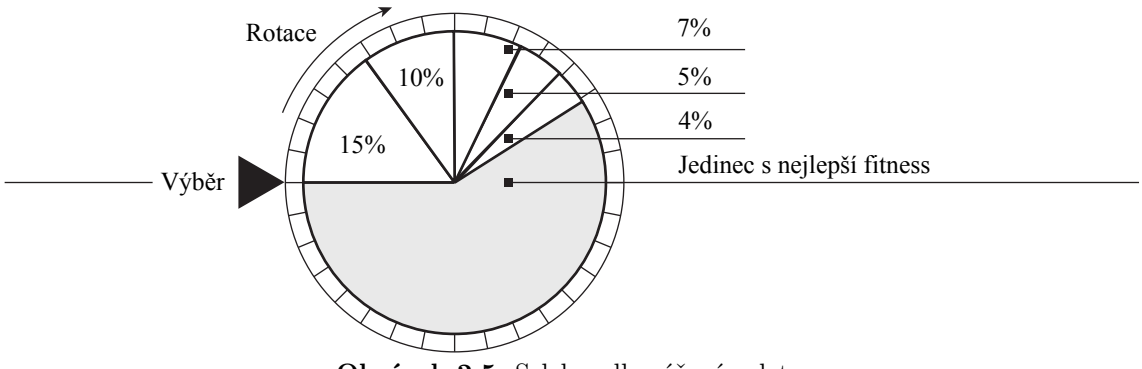
**Náhodný výběr** je nejjednodušší variantou výběru. Selekce je čistě náhodná bez žádného rozhodování o kvalitách jedinců. Nejedná se o selekční mechanismus v kontextu přirozeného výběru.

**Ořezání** seřadí populaci jedinců od nejlepšího k nejhoršímu a dle koeficientu $0 \leq SR \leq 1$ se vyřadí nejhorší jedinci. Ze zbývajících řešení se pak dle náhodně nebo dle deterministické metody vyberou jedinci ke křížení.

**Turnajová metoda** napodobuje přírodní mechanizmy, kdy se vybere náhodně z populace skupina jedinců (dva až tři) a jedinci s nejlepší fitness se stávají rodiči. Metoda se opakuje do té doby, dokud není zaplněna celá populace.

**Vážená ruleta** je asi nejpoužívanější metodou selekce. Princip je založen na ruletě, kdy každý jedinec populace dostane procentuální podíl rulety, dle své fitness funkce, podíl se spočítá pomocí vzorce (vz: 2.14). S vytvořenou ruletou se pak losují rodiče pro křížení. Princip je naznačen na obrázku (obr: 2.5), *Langdon B. William*, *Poli Riccardo*: *Foundations of Genetic Programming*[5].

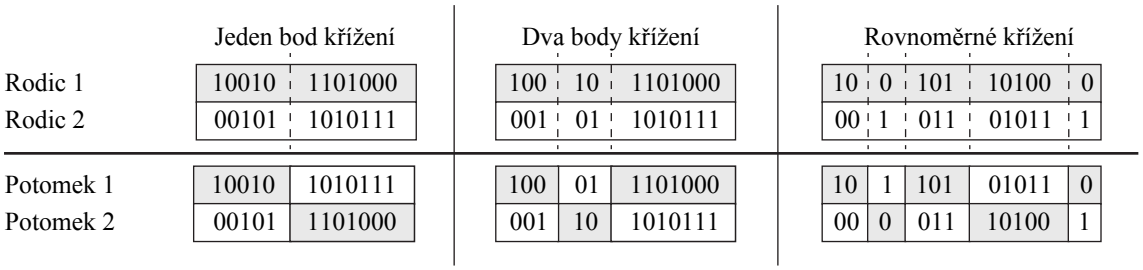
$$p_i = \frac{f_i}{\sum_{i=1}^{NP} f_i} \tag{2.14}$$




**Obrázek 2.5:** Selekce dle vážené rulety.

### Křížení

Křížení následuje hned po selekci rodičů k reprodukci. Jako v přírodě i v (EA) selekce je podstata vyměnění informace od dvou rodičů a vytvoření potomka. Nejjednodušší metody selekce jsou jedno nebo dvou bodové křížení, kdy jsou náhodně vybrány body a podle těchto bodů jsou vyměněny informace. Další možností je rovnoměrné křížení, toto křížení pracuje pomocí koeficientu reprodukce $0 \leq CR \leq 1$. Koeficient je procentuální hodnota kolik informací z jednotlivých rodičů je přeneseno na výsledné potomky. Princip vybraných křížení je na obrázku (obr: 2.6).




**Obrázek 2.6:** Vybrané křížení chromozomů.

### Mutace

Mutace je poslední část (EA) při kterém se pozmění informace v nové populaci. Při mutaci se projde celý chromozom a s určitou pravděpodobností $F$ se změní bity genu z $0 \leftarrow 1$ a opačně. Mutace je další možnost (EA) při zkoumání prostoru, i tam kde prozatím rodiče nebyli, tedy nemohli předat informace. Počet mutujících bitů v populaci se spočítá pomocí funkce (vz: 2.15).

$$N_b = F(NP - 1)N_{bit} \tag{2.15}$$

## 2.3.3 Diferenciální evoluce

Diferenciální evoluce (DE) je stochastická metoda hledání minima v multidimenzionálním prostoru. Algoritmus byl poprvé představený v publikaci *Storn Rainer*, *Price Kenneth*: *Differential Evolution – a Simple and Efficient Heuristic for Global Optimization*[16]. Samotný algoritmus do dnešních dnů velice zpopulárněl a existují mnoho variací. Výhoda (DE) je v jednoduché implementaci a výpočetně nenáročnému generování nových populací. Další velká výhoda algoritmu je schopnost pracovat se spojitými parametry bez nutnosti generování chromozomů. Nevýhoda algoritmu je považována velká závislost na vstupním nastavení metody $F$ [koeficient mutace], $CR$ [koeficient křížení] a $NP$ [velikost populace]. R. STORN a K. PRICE navrhli defaultní nastavení jako *Storn Rainer*, *Price Kenneth*: *Differential Evolution – a Simple and Efficient Heuristic for Global Optimization*[16]:

$$F = 0.8,\ CR = 0.5 \text{ a } NP = 10 \cdot N_{par}$$

Při experimentálních testech i na příkladech z praxe se ukazuje, že (DE) konverguje rychleji než jiné stochastické metody. Obecný algoritmus (DE) generuje novou

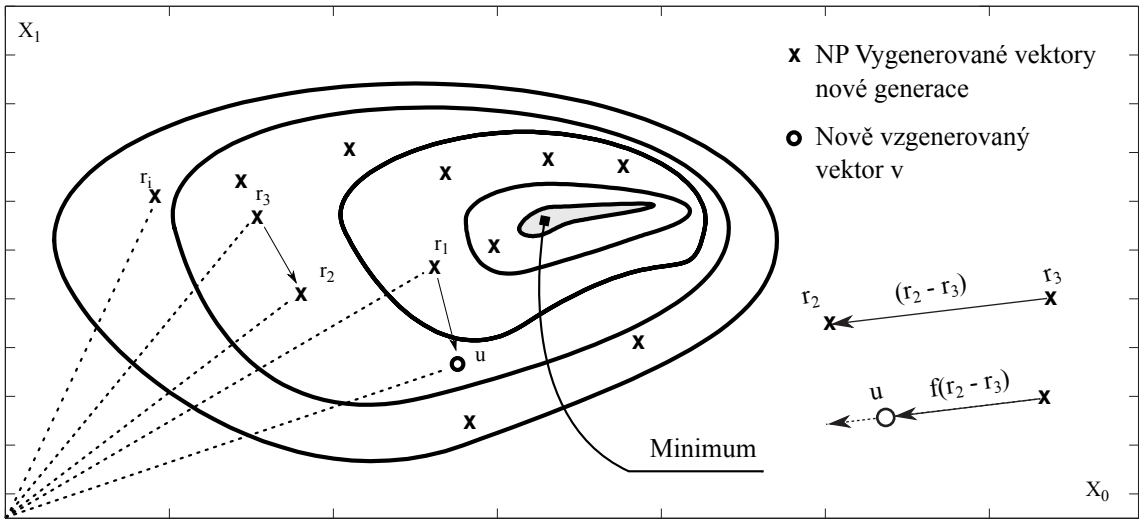
populaci $Q$ postupem vytvoření pro každý bod $x_i$ staré populace konkurenta a porovnává jejich fitness. Jestli je fitness nového bodu lepší, než u stávající populace, tak je nahrazen tímto bodem. Podrobný popis (DE) je na diagramu (obr: 2.8) a algoritmu (alg: 7), *Tvrdík Josef*: *Evoluční Algoritmy*[12].

### Vyhledávací strategie

Algoritmus generuje nove body populace zpracováváním informace ze vzdálenosti a směru vygenerovaných vektorů, tímto adaptivním postupem dokáže (DE) prezentovat excelentní schopnost konvergovat ke globálnímu minimu. Při vývoji (DE) algoritmu bylo testováno mnoho variant, speciálně při generování nových vektorů, tato sekce se zabývá nejužívanější variantou dle obrázku (obr: 2.7). Další varianty příkladů mohou být podle vzorce následující (vz: 2.16). Vyhledávací strategie a generování nového bodu $u$, jsou nejdůležitější části (DE). Existuje mnoho druhů strategii, zde jsou uvedeny čtyři základní a nejvíce využívané typy *Feoktistov Vitaliy*: *Differential Evolution: In Search of Solutions*[15]:



$$\begin{aligned} u &= r_1 + F(r_2 - r_3) \\ u &= r_5 + F(r_1 + r_2 - r_3 - r_4) \\ u &= x_{best} + F(r_1 - r_2) \\ u &= x_{best} + F(r_1 + r_2 - r_3 - r_4) \end{aligned} \tag{2.16}$$



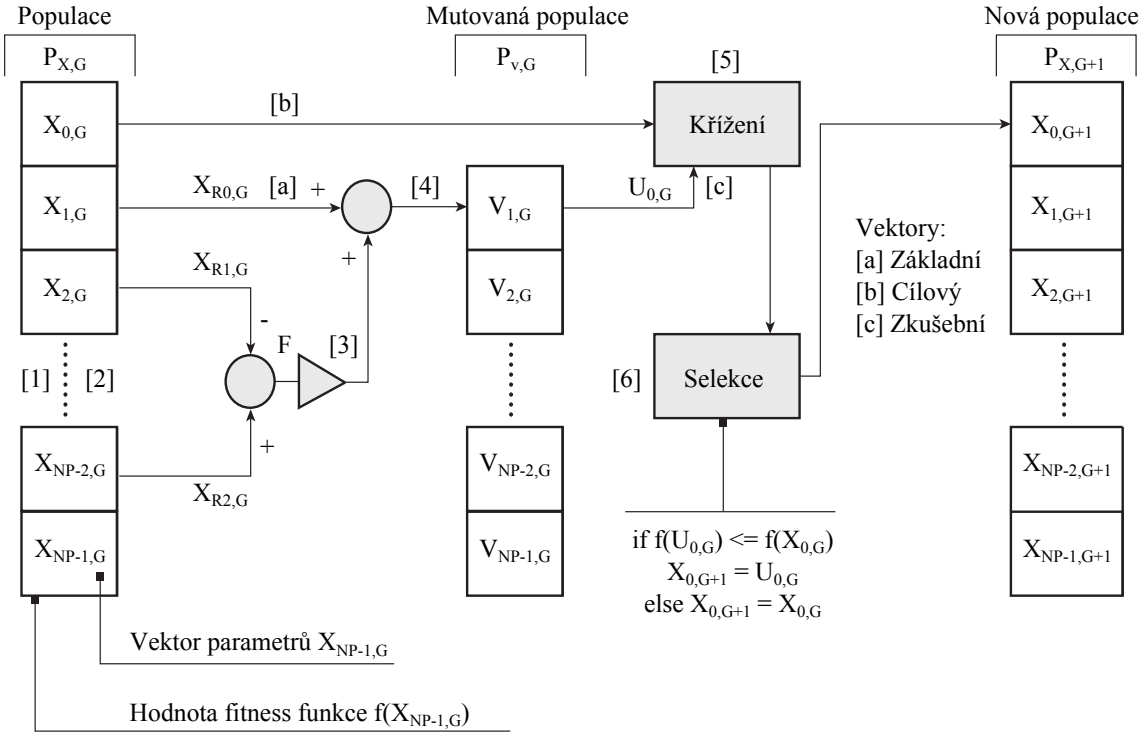

**Obrázek 2.7:** Vyhledávací strategie dle první funkce (vz: 2.16).

$r_1, r_2 \ldots r_n$ jsou různé body náhodně zvolené z populace $P$ a různé od aktuálního bodu $x_1$. $F$ je koeficient mutace $0 \leq F \leq 2$, která zajišťuje prohledávání i neznámých oblastí. Generování bodu $u$ (zkušebního vektoru) je znázorněno graficky pro příklad prohledávání na obrázku (obr: 2.7), dle první vyhledávací strategie. Tento proces je znám také jako mutace *Tvrdík Josef*: *Evoluční Algoritmy*[12].

$$y_j = \begin{cases} u_j & \text{když } R_j \leq CR \quad \text{nebo} \quad j = I \\ x_{ij} & \text{když } R_j > CR \quad \text{a} \qquad j \neq I \end{cases} \tag{2.17}$$

Nový vektor $y_j$ vznikne po křížení základního vektoru $x_i$, tak že jakýkoliv prvek

$x_ij$ je nahrazen hodnotou $u_j$ s pravděpodobností $CR$. Přiřazení do vektoru $y$ je za pomocí funkce (vz: 2.17). Tato operace se nazývá selekce *Tvrdík Josef*: *Evoluční Algoritmy*[12].




**Obrázek 2.8:** Princip differencialni evoluce.

### 2.3.4 HC12

Algoritmus (HC12) je globální stochastická metoda využívaná k optimalizaci, poprvé publikována v práci *Matoušek Radomil*: *HC12: Highly Scalable Optimization Algorithm*[19]. Princip metody je založen na horolezeckém principu prohledávání, definice horolezeckého principu je v sekci (sekce: 2.1 - viz str. 15). (HC12) využívá ke generování nových bodů v každé iteraci binární masky (vz: 2.25) a (vz: 2.26), kde princip definice problému a jeho dekódováni z binárního řetězce je podobné jak u (GA) (vz: 2.10) a (vz: 2.11) a využívá grayův kód k reprezentaci hodnot parametrů. Popis principu (HC12) je v algoritmu (alg: 4), *Matoušek Radomil*, *Žampachová Eva*: *Promising GAHC and HC12 algorithms in global optimization tasks*[18].

Jedinečnost algoritmu HC12 spočívá v kombinaci specifického kódování problému a způsobu generování populace řešení, tedy tzv. okolí. Na těchto principech byl prakticky zrealizován perspektivní optimalizační algoritmus, který přes svoji jednoduchost prokazuje až překvapivě dobré vlastnosti. Algoritmus HC12 umí efektivně prohledávat prostor a nalézat řešení spojitých i kombinatorických problémů. Navíc v dnešní době velmi užitečné a je velmi dobře škálovatelný *Matoušek Radomil*: *Pokročilé metody počítačové intligence*[61].

Implementace HC12 jsou založeny na restartech, což významně posiluje schopnost algoritmu nalézat globální řešení. Pro aplikaci HC12 algoritmů musí být expertně zvážen typ úlohy a požadovaná přesnost výpočtu, dále výkon výpočetní platformy

a typ implementace. Implementované varianty HC12 disponují následujícími možnostmi:

- Volby účelové funkce s případnou parametrizací (například čas) a definici počtu optimalizovaných proměnných omezenou pouze dostupnými zdroji, viz dále *Matoušek Radomil*: *Pokročilé metody počítačové intligence*[61].
- Volby rozsahu binární reprezentace pro každou reálnou proměnnou, což u spojitých úloh determinuje přesnost výpočtu. Vzhledem k nejefektivnější implementaci je tento parametr určen maximálním rozsahem datového typu *uint32*, tedy délkou 32 bit/proměnnou *Matoušek Radomil*: *Pokročilé metody počítačové intligence*[61].
- Volby definičních intervalů $[x_{min,i}, x_{max,i}]$ všech optimalizovaných proměnných $x_i$ úlohy, tedy v podstatě volby omezujících podmínek typu $x_{min,i} \leq x_i \leq x_{max,i}$. V případě zadání pouze jednoho intervalu je tento interval automaticky použit na všechny optimalizované proměnné *Matoušek Radomil*: *Pokročilé metody počítačové intligence*[61].

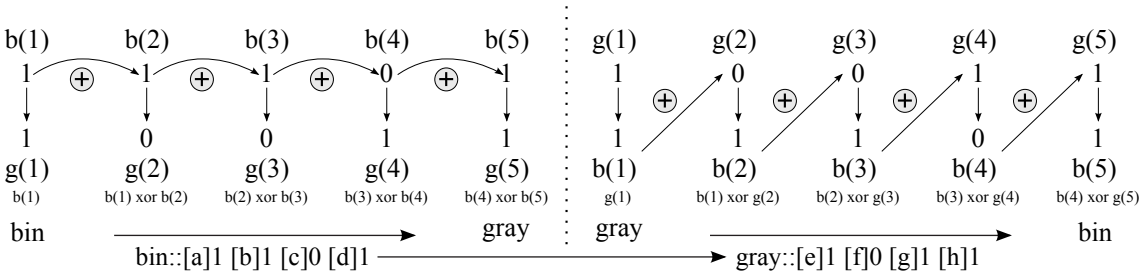


### Grayův kód

Algoritmus (HC12) využívá grayovo kódování k reprezentaci hodnot parametrů. Vlastnost grayova kódování spočívá vtom, že každé po sobě jdoucí binární číslo se liší jen o jeden bit. Hammingova vzdálenost (vz: 2.22) hodnot grayového kódování lépe odpovídá rozdílnosti reálných čísel při přepočtu na reálné hodnoty. Pro přepočet binárního kódování se používá metoda na základě jednotlivých bitů. Princip převodu je následující:

- Dvojková číslice přímého binárního kódu s nejvyšší vahou se ponechá beze změny.
- Každá následující dvojková číslice přímého binárního kódu se invertuje, když ji v přímém kódu předchází na vyšší váze jednička.

Příklad převodu je na obrázku (obr: 2.9). Hodnoty $a$ (bit s nejvyšší váhou), $b$ (invertující znak 1), $c$ (neinvertující znak 0), $d$ (poslední znak - neinvertující), $e$ (bit s nejvyšší váhou se nemění), $f$ (invertovaný znak - předchází 1), $g$ (invertovaný znak - předchází 1) a $h$ (neinvertovaný znak - předchází 0).




**Obrázek 2.9:** Převod mezi bin a gray kódováním.

### Populace

Populace v algoritmu je stejně jak (GA) reprezentována chromozomy, zde reprezentována binárními vektory v grayově kódování (vz: 2.18). Populace je pak zapsána

v matici jedinců (vz: 2.19), kdy každý řádek této matice reprezentuje právě jednoho jedince.

$$a = (a_1, a_2, a_3 \dots a_n) \in \{0, 1\}^n \tag{2.18}$$

$$A_p = \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & \dots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{NP,1} & a_{NP,2} & \dots & a_{NP,n} \end{pmatrix} \tag{2.19}$$

K převodu chromozomu na číselnou hodnotu se nejdříve musí provést převod z Grayova kódu na binární a pak provést lineární transformaci dle funkce (vz: 2.20) respektive (vz: 2.21).



$$y = kp_n + p_{lo} \tag{2.20}$$

$$x = \frac{|p_{hi} - p_{lo}|}{N_{bit}^2} \tag{2.21}$$

### Hammingova vzdálenost

Hammingova vzdálenost je počet bitů o kolik se liší dva binární vektory stejné velikosti. Mějme funkci $\rho$ pro výpočet vzdálenosti v prostoru, tak funkce $\rho_h$ dle hamingových pravidel vypadá pro dva vektory $a$, $b$ následovně:

$$\rho_h(a, b) = \sum_{i=1}^{N_{bit}} |a_i - b_i| . \tag{2.22}$$

### Generující maska

K výpočtu nových potomků v každé iteraci je použita generující maska, metoda (HC12) využívá binární masky v hammingově vzdálenosti 1 a 2 - odtud jméno HC12. Jednotlivé matice masek jsou pojmenovány $M_0$, $M_1$ a $M_2$. Počet řádků $M_1$ je dán vztahem $\binom{n}{1}$ a pro $M_2$ je $\binom{n}{2}$. Celkový počet řádků matice je znázorněn vztahem (vz: 2.23).

$$m = 1 + \underbrace{\binom{n}{1}}_{c_1} + \underbrace{\binom{n}{2}}_{c_2} \tag{2.23}$$

Maska $M_0$, vektor (vz: 2.24) nemění žádné informace v chromozomu (ponechává původní řešení).

$$M_0 = (0_{1,1}, 0_{1,2}, 0_{1,3} \dots 0_{1,n}) \tag{2.24}$$

Maska $M_1$, matice (vz: 2.25) mění vždy jeden bit v chromozomu a má počet řádů rovno počtu $c_1$.

$$M_1 = \begin{pmatrix} 1_{1,1} & 0_{1,2} & 0_{1,3} & \dots & 0_{1,n} \\ 0_{2,1} & 1_{2,2} & 0_{2,3} & \dots & 0_{2,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0_{c_1,1} & 0_{c_1,2} & 0_{c_1,3} & \dots & 1_{c_1,n} \end{pmatrix} \tag{2.25}$$

Maska $M_2$, matice (vz: 2.26) mění vždy dva bity v chromozomu a má počet řádků $c_2$.

$$M_2 = \begin{pmatrix} 1_{1,1} & 1_{1,2} & 0_{1,3} & \dots & 0_{1,n} \\ 1_{2,1} & 0_{2,2} & 1_{2,3} & \dots & 0_{2,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0_{c_2,1} & 0_{c_2,2} & \dots & 1_{c_2,n-1} & 1_{c_2,n} \end{pmatrix} \tag{2.26}$$



Příklad použití generující masky na řešení ve vektoru **00000** je v tabulce (tab: 2.2). Tabulka ukazuje i převod mezi Grayovým kódováním a binárním.

| Mask | ID | Gray | Bin | Mask | ID | Gray | Bin | ID | Gray | Bin |
|---|---|---|---|---|---|---|---|---|---|---|
| M1 | 1 | 10000 | 11111 | M2 | 1 | 11000 | 10000 | 6 | 01010 | 01100 |
| | 2 | 01000 | 01111 | | 2 | 10100 | 11000 | 7 | 01001 | 01110 |
| | 3 | 00100 | 00111 | | 3 | 10010 | 11100 | 8 | 00110 | 00101 |
| | 4 | 00010 | 00011 | | 4 | 10001 | 11110 | 9 | 00101 | 00110 |
| | 5 | 00001 | | | 5 | 01100 | 01000 | 10 | 00011 | 00010 |

**Tabulka 2.2:** Příklad použití generující masky na řešení ve vektoru 00000.

## 2.4 Gramatická evoluce

Gramatická evoluce (GE) je jeden z nejnovějších přístupů v evolučních algoritmech a je založena na automatickém generování počítačových programů v jazycích popsaných Backus-Naurovou formou (BNF). Algoritmus byl poprvé představený v publikaci *Ryan Conor*, *Collins J.J.*, *O'Neill Michael*: *Grammatical Evolution: Evolving Programs for an Arbitrary Language*[22]. Gramatická evoluce je v podstatě princip kódování/dekódování (GA) pomocí (BNF), tj. genotyp je interpretován stejně jako v (GA) a fenotyp - dekódování proběhne podle pravidel (GE). Při (GE) se hlavně využívá jednobodové křížení a bodová mutace *O'Neill Michael*, *Rayn Conor*: *Grammatical evolution : Evolution Automatic Programming in an Arbitrary Language*[20].

### Backus-Naurova forma

Backus-Naurova forma (BNF) je gramatika popisující jazyk pomocí produkčních pravidel. Produkční pravidla mohou být terminály, tedy konečné prvky jazyka, které se už dál nerozšiřují např. (sin, +, -). Neterminály se pak rozšiřují na další neterminály nebo terminály. Gramatiku budeme uvažovat zápis (vz: 2.27), *Dempsey Ian*,

```
expression = cos((x+x+(x-x)))*(sin(x)+tan((1+x)))
```

**GE výsledek 2.1:** Vypočtený výraz pomocí GE gramatiky [Typ (gramatika: 2.1)] pro testovací funkci [(tan(x+1)+sin(x))*cos(2*x)].

*O'Neill Michael*, *Brabazon Anthony*: *Foundations in Grammatical Evolution for Dynamic Environments*[21].

$$G = \{N, T, P, S\} \tag{2.27}$$

Příkladem jednoduché gramatiky BNF je hledání aritmetických výrazů a je zobrazeno na gramatice (gramatika: 2.1) a GE výsledek (expr: 2.1) je jeden z možných výsledků takovéto gramatiky, při hledaní tvaru jednoduché funkce Příklad rozdělený do jednotlivých kroků je v příloze (příloha: X.18).

Takto vytvořená gramatika se pak kóduje za pomocí chromozomů (GA), je ve formátu binárního řetězce. Chromozomy (GE) nenesou oproti (GA) hodnotu aktuálních parametrů, ale geny představují posloupnost produkčních pravidel. Dekódovaný chromozom, tedy představuje výraz předepsané gramatiky.



### Dekódování chromozomu

Chromozomy v (GE) mají proměnou délku a obsahují tzv. kodony, tedy části bitového kódu o velikosti 8bit (0-255). Tyto kodony rozhodují, v jakém pořadí se bude využívat produkční pravidla. Ve většině případů nemáme 255 pravidel, proto se využívá přepočet (vz: 2.28). Příklad dekódování kodonu 11000110 je hodnota 198 kdy použít pravidlo $\langle var \rangle$. Výsledný index je 2 tedy použijeme terminál $''/''$.

$$p_{id} = MOD\,(k_v, p_{NP}) \tag{2.28}$$

Dekódování chromozomu končí nastane-li jedna ze tří podmínek ukončení.

- Všechny kodony byly převedeny na produkční pravidla a výsledný výraz neobsahuje žádné neterminály.
- Výraz neobsahuje žádné neterminály, ale nebyly převedeny všechny kodony, tyto kodony se budou ignorovat.
- Výsledný výraz obsahuje neterminály a všechny kodony byly převedeny na produkční pravidla. V tomto případě se pokračuje dekódování chromozomu od začátku, ale hrozí nebezpečí zacyklení, kdy nebudeme moc najít konečné řešení bez neterminálu, proto se definuje maximální počet opakování chromozomu.

```
        N = { expr, op, pre_op, var} | T = { sin, cos, tan, +, -, *, /, X, 1 }
        S = { expr }

        Produkční pravidla jsou zapsaná v následujícím pořadí.

<expr>  ::<expr> <op> <expr> | (<expr> <op> <expr>) | <pre_op> (<expr>) | <var>
<op>    ::+ | - | / | *
<pre_op>::sin | cos | tan
<var>   ::X | 1
```

**Gramatika 2.1:** Příklad Backus–Naurovy gramatiky k hledání aritmetických výrazů.

### 2.4.1 Aplikace

Gramatické evoluce se může využít v teorii řízení na návrh vlastních struktur lineárních i nelineárních regulátorů. Struktůra regulátoru udává zvolená gramatika, viz příklad gramatiky (vz: 2.27). Jedna z možností návrhu regulátoru je nastavení pólů a nul v komplexní rovině (pole placement). Tento typ regulátoru vykazuje kvalitnější výsledky než standartní typy, jak dokazují experimentální testy *Matoušek Radomil*, *Minář Petr*, *Lang Stanislav*, *Pivoňka Petr*: *Evolutionary Design of Polynomial Controller*[46]. Výsledná struktura regulátoru může být dále optimalizována ve svých parametrech jinou optimalizační metodou, například (DE) nebo (HC12). Tento typ optimalizace můžeme označit jako dvouúrovňový návrh.

## 2.5 Fuzzy logika

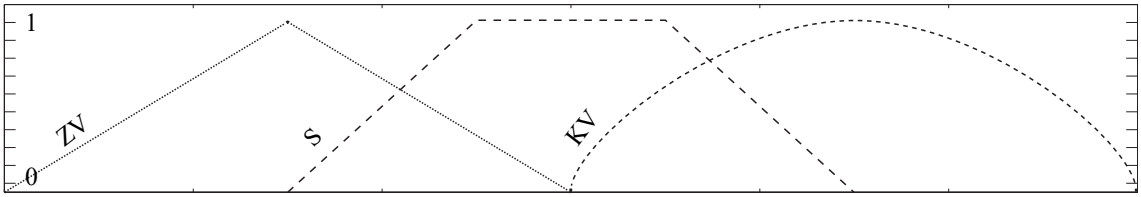


Fuzzy logika (FL) rozšiřuje klasické dvoupolohové pojetí příslušnosti prvku v množině na vícehodnotové, které mnohem lépe odpovídá realitě. Princip fuzzy množin byl založen na základě publikace *Zadeh A. Lotfi*: *Fuzzy sets*[27] a spočívá ve funkcích příslušnosti. Funkce příslušnosti určuje stupeň příslušnosti prvků v množině a nabývá hodnot v rozsahu $< 0, 1 >$. Klasická teorie množin na rozdíl od fuzzy nemají funkci příslušnosti a buď prvek paří do množiny nebo ne, tedy nabývá hodnot 0 nebo 1. FL obsahuje syntaxe a sémantiky k formulování vlastních zkušeností a daném problém k nalezení řešení *Modrlák Osvald*: *Fuzzy řízení a regulace: Studijní materiály*[24].

#### Fuzzy množiny

Fuzzy množiny neboli neostré množiny jsou založeny na funkcích příslušnosti, které určují, jak moc daná proměnná patří do konkrétní množiny. Na obrázku (obr: 2.10) je znázorněný typový příklad fuzzy množiny s třemi různými funkcemi příslušnosti. Matematický zápis fuzzy množiny v univerzu $U$ (vz: 2.29) *Passino M. Kevin*: *Fuzzy Control*[25].

$$A = (U, \mu_A),\ \mu_A \leftarrow \langle 0, 1 \rangle \tag{2.29}$$




**Obrázek 2.10:** Příklad fuzzy množiny s třemi funkcemi příslušnosti.

#### Lingvistické proměnné

Lingvistické proměnné se používají k využití a přenesení empirických zkušeností člověka do teorie fuzzy množin. Ligistická proměnná je taková proměnná jejíž hodnoty jsou výrazy nějakého jazyka. Tato hodnota se dá interpretovat jako fuzzy množina.

Příkladem takovýchto proměnných je obrázek (obr: 2.10), kde funkce příslušnosti jsou nazvané $ZV$ (záporná velká) $S$ (střední) $KV$ (kladná velká).

### Operátory fuzzy logiky

Ve (FL) můžeme interpretovat tři základní booleovské operace s množinami například: (AND, OR, NOT).

Fuzzy komplement (doplněk množiny A) C = NOT A.

$$\mu_C(x) = \mu_{A\neg}(x) = 1 - \mu_A(x) \tag{2.30}$$

Fuzzy průnik množin (logický součin) C = A AND B.

$$\mu_C(x) = \mu_{A\wedge B}(x) = min\left(\mu_A(x), \mu_B(x)\right) \tag{2.31}$$

Fuzzy sjednocení množin (logický součet) C = A OR B .

$$\mu_C(x) = \mu_{A\vee B}(x) = max\left(\mu_A(x), \mu_B(x)\right) \tag{2.32}$$

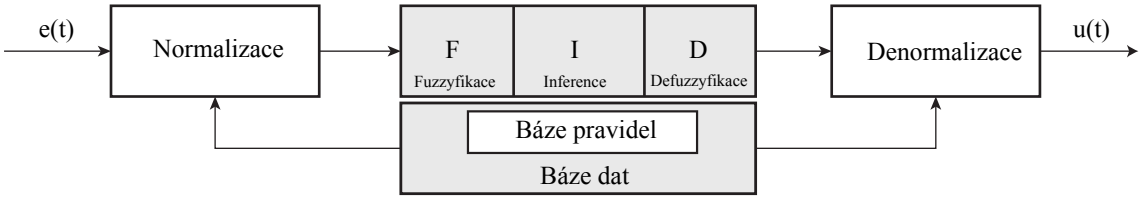


## 2.5.1 Aplikace

Využití fuzzy logiky při návrhu řízení systémů je možno navrhnout fuzzy regulátor několika způsoby. První možností návrhu klasický fuzzy regulátor znázorněný na obrázku (obr: 2.11), kdy se nastavují jen funkce příslušnosti při vhodném zvolení fuzzyfikace a defuzzyfikace vstupní proměnné, neboli implementací fuzzy (PID) regulátoru - obrázek (obr: 2.12). Při tomto typu regulátoru je velice vhodná kombinace stochastických optimistických algoritmů a fuzzy logiky. Při návrhu regulátoru se optimalizační algoritmus využívá pro tento typ parametrů:

- návrh vstupních a výstupních zesilovačů při lineární rozložení fuzzy funkcí příslušnosti,
- návrh rozloženi fuzzy funkcí příslušnosti při výpočtu vstupních a výstupních zesilovačů, dle optimálního (PID) regulátoru (vz: 2.33),
- návrh tvarů fuzzy funkcí příslušnosti,
- návrh pravdivostní báze,
- kombinace předešlých přístupů.

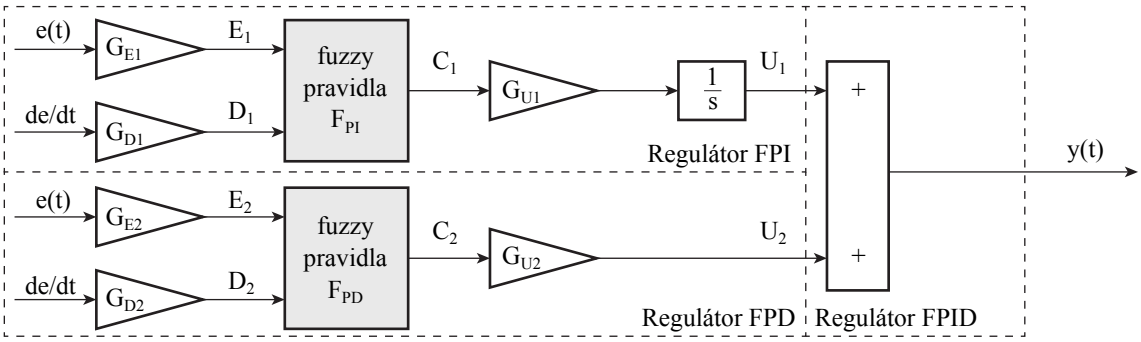
Při nastavení funkcí příslušnosti jinak, než lineárním rozložením vytváří nelineární typ regulátoru. Mnohdy tento typ je lepší a rychlejší pro navrhovaný vstupní signál než klasické lineární regulátory jako například (PID) *Jantzen Jan*: *Foundations of Fuzzy Control*[26].




**Obrázek 2.11:** Schéma klasického fuzzy regulátoru.

$$G_{E1,2} = 100(|max(u(t))|) \tag{2.33}$$

$$G_{D1} = T_i(G_{E1,2}) \text{ a } G_{D2} = T_d(G_{E1,2}), G_{U1,2} = \frac{r_0}{G_{E1,2}} \tag{2.34}$$




**Obrázek 2.12:** Schéma FPID (FPD + FPI) modelu.

## 2.6 Možné přístupy k evolučnímu návrhu regulátorů

2

Řízení soustav s využitím metod umělé inteligence je dnes v oblasti návrhu regulátoru velmi aktuální a je mnoha autory intenzivně studováno. Problematika optimálního návrhu PID je dnes dobře vyřešena klasickými postupy teorie řízení, přičemž je rovněž řešena pomoci soft computingových metod. Návrh optimálních parametrů PID byl řešen například *Fabijanski P.*, *Lagoda R.*: *On-line PID controller tuning using genetic algorithm and DSP PC board*[42], *Gao F.*, *Tong H. Q.*: *Differential Evolution: An Efficient Method in Optimal PID Tuning and on-line Tuning*[43], optimalizace hejnem částic *Muangsong*, *R.*, *Koolpiruck*, *D.*, *Khantachawana*, *A.*, *Niranatlumpong P.*: *A particle swarm optimization approach for optimal design of PID controller for position control using Shape Memory Alloys*[44] nebo mravenčí algoritmy *Nagaraj B.*, *Vijayakumar P.*: *A comparative study of PID controller tuning using GA, EP, PSO and ACO*[41]. Další možností využití evolučních algoritmů představuje řešení *Kukal Jaromír*, *Jakeš Bohumil*, *Bártová Darina*: *Využití diferenciální evoluce k robustní identifikaci parametrů systémů*[14].

Společným jmenovatelem uvedených řešení je majoritně následující nedostatek: řešení zohledňují jen jedno, v reálném provozu povětšinou nedostatečné kritérium (například integrační kritéria). Přitom je zřejmé, že požadavky na průběh regulačního procesu jsou z podstaty multikriteriální. Zohledňujeme například překmit, dobu regulace, oscilace atd. Dalším nedostatkem uvedených metod je povětšinou uvazovaný jen lineárně definovaný systém.

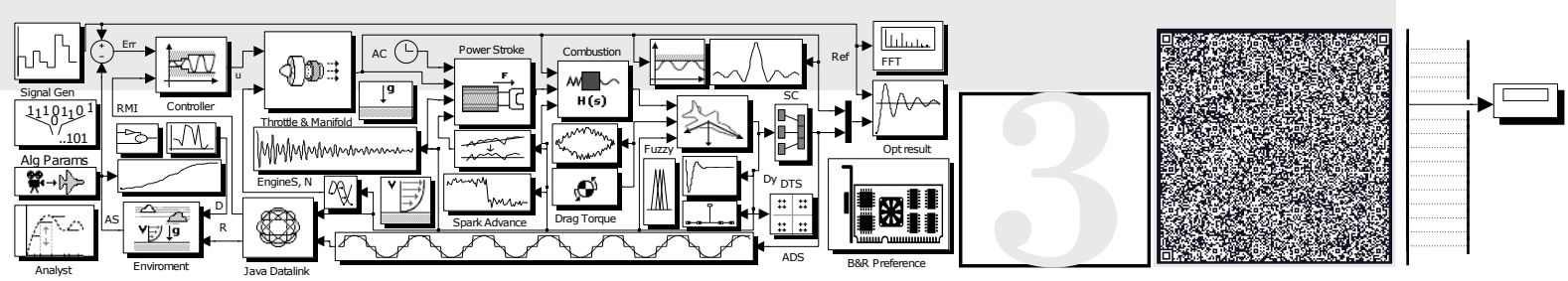
Problematika fuzzy regulace představuje v dnešní době rovněž solidně propracovaný aparát ke stabilizaci regulačního procesu, dle prací *Chuen Chien Lee*: *Fuzzy logic in control systems: fuzzy logic controller-parts 1 and 2*[34] a *Chuen Chien Lee*: *Fuzzy logic in control systems: fuzzy logic controller-parts 1 and 2*[28]. Práce *Mamdani H. Ebrahim*: *Application of fuzzy algorithms for the control of a simple dynamic plant*[30] i *Sugeno Michio*: *Theory of fuzzy integrals and its applications*[31] prakticky ukázaly dobrou aplikovatelnost této třídy regulátorů na reálné problémy,

například *García-Pérez Lía*, *Cañas M. José*, *García-Alegre C. María*, *Yáñez Pablo*, *Domingo Guinea*: *Fuzzy control of an electropneumatic actuator*[35], *King E. Robert*, *Stathaki A.*: *Fuzzy Gain Scheduling Control of Nonlinear Processes*[36]. Tyto typy regulátorů mají obecně nevýhodu ve složitosti nastavení, ale výhodu v robustnosti i dobré použitelnosti pro eliminaci některých nelinearit. Z podstaty se jedná o nelineární regulátory. S těmito regulátory již byly realizovány experimenty, které dokládají jejich použitelnost a vhodnost dalšího výzkumu, například práce *Ošmera Pavel*: *Evolvable Fuzzy Controllers using Parallel Evolutionary Algorithms*[32], *Matoušek Radomil*, *Ošmera Pavel*: *Automatic Optimal Design of Fuzzy Controllers*[33] nebo *Lacevic Bakir*, *Velagic Jasmin*: *Evolutionary Design of Fuzzy Logic*[29].

Výše uvedené principy v podstatě realizovaly regulátory s pevnou strukturou. V práci budou rozvíjeny metody gramatické evoluce, které umožňují současný návrh, jak struktury, tak parametrů obecného regulátoru. Podobné přístupy jsou velmi inovativní a již byly studovány a na Ústavu informatiky a automatizace rovněž rozvíjeny. Příkladem je *Zelinka Ivan*: *Analytic programming by means of new evolutionary algorithms*[52], *Burbidge Robert*, *Wilson S. Myra*: *Grammatical evolution of a robot controller*[23] nebo využití principu transplantační evoluce *Ošmera Pavel*, *Roupec Jan*, *Matoušek Radomil*, *Weisser R.*: *Two-Level Transplant Evolution for Optimization of General Controllers*[53].



Z hlediska aplikování evolučních algoritmů na problém řízení deterministického chaosu jsou nastíněné různé možností postupů a hlavní jsou z prací profesora zelinky a jeho týmu a to konkrétně *Šenkeřík Roman*, *Zelinka Ivan*, *Davendra Donald*, *Oplatková Zuzana*: *Utilization of SOMA and differential evolution for robust stabilization of chaotic Logistic equation*[79], *Zelinka Ivan*: *Investigation on Realtime Deterministic Chaos Control by Means of Evolutionary Algorithms*[78] a *Šenkeřík Roman*: *Optimal Control of Deterministic Chaos*[51]. Problematika řízení chaosu je perspektivní odvětví využití evolučních přístupů k řešení problému.




# KAPITOLA 3

# SOFTWAROVÉ ŘEŠENÍ



Z důvodu ověření navrhovaných algoritmů implementovaných majoritně v teorii řízení bylo potřeba vytvořit vlastní optimalizační aplikaci, která obsahuje všechny předešlé algoritmy a je schopná spouštět matematické modely v různých formátech zadání. Zadání modelů je ve formě testovacích příkladů *May M. Robert*: *Simple mathematical models with very complicated dynamics*[55] nebo modely k teorii řízení dle syntaxe Matlab skript a Simulink model design a vycházejí z práce *Fedorov V. Valerii*, *Leonov L. Sergei*: *Optimal Design for Nonlinear Response Models*[68].

Využití matlab prostředí k řešení optimalizačních úloh je jednoduché a efektivní, implementace vychází z textu *Mathews H. John*, *Kurtis K. Fink*: *Numerical Methods Using Matlab*[8]. Celkový koncept aplikace vychází z autorových předešlých prací *Minář Petr*: *Interaktivní webové aplikace vytvořené pomocí technologie Adobe Flash - Teorie Kódování*[71] a *Minář Petr*: *Distribuované výpočty s využitím technologie ActionScript*[10] ze kterých byly odvozené způsoby komunikace mezi výslednými aplikacemi a implementování evolučních algoritmů a jejich škálování v multi-procesorovém prostředí. Dále se z těchto prací používá návrh rozdělení aplikací na *Server* a *Klient* část z důvodu co možná největší univerzálnosti výsledného řešení. Samotná aplikace je napsaná programovacím jazykem Java *Sierra Kathy*, *Bates Bert*: *Head first Java*[49], kvůli její multi-platformosti a celkové robustnosti zvoleného jazyka. Využité technologie v rámci komunikace jsou TCP/IP *Calvert L. Kenneth*, *Donahoo J. Michael*: *TCP/IP Sockets in Java: Practical Guide for Programmers*[65] a vzdálené volaní funkcí ser-

ver aplikace *Grosso William*: *Java RMI*[66]. Pro dosažení co nejlepšího výpočetního času na počítačích s více jádrovými procesory se využívá spouštění modelů v rámci technologie Matlabpool *Kepner Jeremy*: *Parallel MATLAB for Multicore and Multinode Computers*[67] v aplikaci Matlab / Simulink a samotné škálování výpočtů v Java aplikaci dle *Pacheco Peter*: *An Introduction to Parallel Programming*[72] a *Tomek Eduard*: *Škálování implementace optimalizačního algoritmu na mnohajádrových strojích*[17]. Komunikační propojení serverové aplikace a matlab prostředí je provedeno dle technologie Java Matlab Interface *Altman M. Yair*: *Undocumented Secrets of MATLAB-Java Programming*[50], který implementuje socket komunikaci mezi hostitelskou Java aplikací a Matlab programem. Implementace evolučních algoritmů a zapojení matlab prostředí k externímu výpočtu problému přebírá některé programové techniky z již existujících prací a to konkrétně *Linder Marek*, *Pernecký Daniel*, *Sekaj Ivan*: *Grid Computing in Matlab for Solving Evolutionary*[73] a *Sekaj Ivan*, *Linder Marek*: *Evolutionary Algorithm Based Power Plant Working Point Optimisation Using PLC and Simulink Model*[74]. Výsledky vycházející z běhu optimalizace této aplikace a byly navrženy s co možná největší možností dalšího uplatnění na reálných soustavách. Jejich podoba podporuje implementaci do programovatelných automatů od firmy B&R Automation, pomocí technologie Target for Simulink *B&R : Automation Studio Target for Simulink*[69] a implantování do technologie dSpace *Ghaffari Azad*: *dSPACE and Real-Time Interface in Simulink*[70] v rámci Simulink aplikace.



Kapitola softwarové řešení je rozdělena na obecný popis aplikace, kde je představený samotný návrh řešení s popisem jednotlivých částí klient a server aplikace. Další část je věnována podrobnému popisu aplikace a designu řešení v rámci komunikace, aplikačního zapojení a zadávání testovacího prostředí. Tato část je prezentována i podrobnými příklady a UML popisem v příloze této práce. Sekce aplikace výsledů v reálných soustavách stručně představuje možnosti použití hotových řešení v rámci (PLC) a dSpace technologie. Poslední část je experimentální výpočet na sérii testovacích příkladů, který prezentuje výslednou aplikaci, její schopnost nalézt řešení s různými algoritmy a různými způsoby zadání optimalizovaného problému.

## 3.1 Obecný popis aplikace

Softwarová implementace je navržená s ohledem na maximální obecnost možné definice optimalizačního problému a volnost definice optimalizačních metaheuristik. Výsledná aplikace je z tohoto důvodu rozdělena na dvě části. Jedna část reprezentuje optimalizační problém a druhá část implementuje optimalizační algoritmy. Schématické rozdělení je na obrázku (obr: 3.1). Optimalizační část (též nazvaná klientská aplikace) zajišťuje jen chod optimalizačního algoritmu a komunikačního protokolu s výpočtovou částí (server aplikace). Toto část vůbec neobsahuje výpočetní prostředky pro řešení daného problému, jen se odkazuje na server a posílá data o optimalizovaných parametrech a server zpátky vrací vyřešený problém, například ve formě fitness funkce. Serverová aplikace je v jádru jen datalinkový program, který

zajišťuje komunikaci jak od klientů, tak komunikaci s výpočetní částí (model optimalizačního problému), v našem případě aplikací Matlab / Simulink. Podoba vytvořených aplikací jsou prezentovány v příloze (příloha: X.8) pro klientskou aplikaci a v příloze (příloha: X.7) pro serverovou aplikaci.

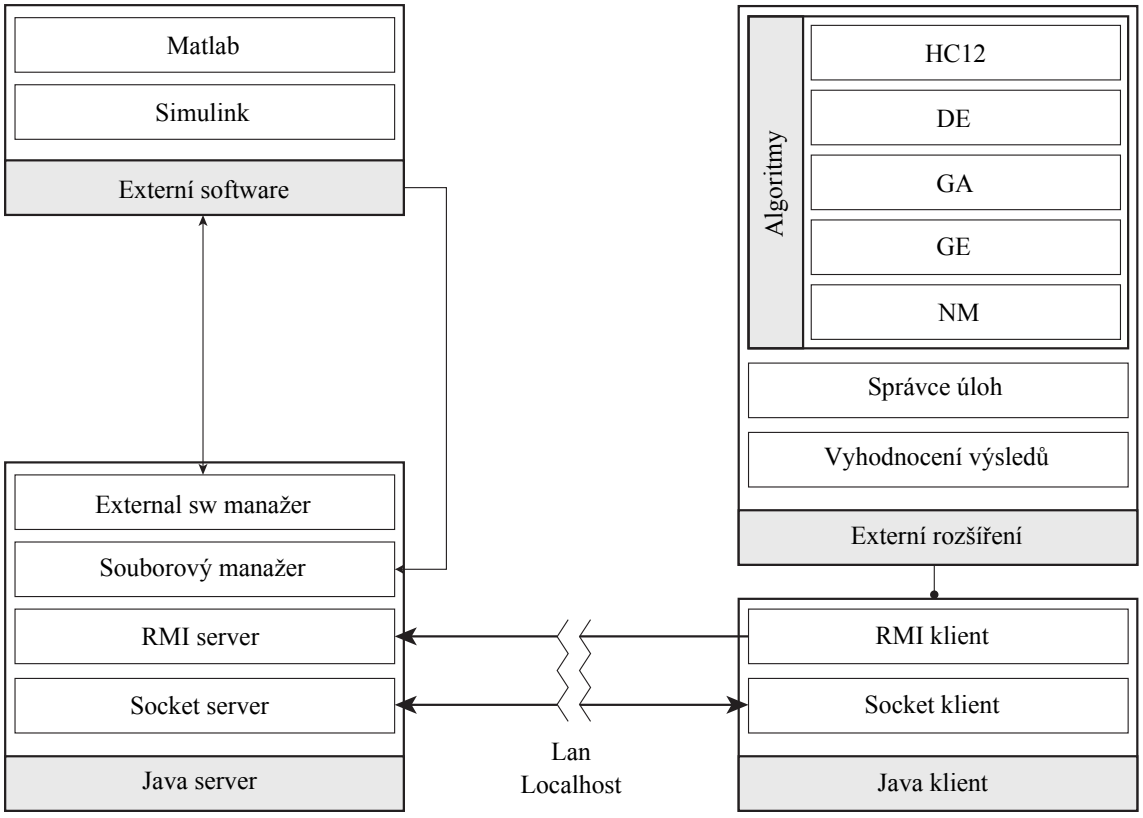
V rámci vývoje výpočetní aplikace se vytvořili i další podpůrné prostředky na analýzu výsledků a podporu řešených problémů. Aplikace pro analýzu problému je část nazvaná jako "Simulation Info generator", její účel je zpracování výpočetních dat a prezentace výsledků na požadovaných hodnotách a výstupech (jak grafická prezentace, tak statistické rozdělení výsledků). V této aplikaci lze i dodatečně analyzovat problém i po ukončeném výpočtu na základě změny parametrů optimalizace a fitness funkce. Další podpůrná aplikace je tzv. "Simulink MDL file parser", jehož účel je zpracování dat z *MDL* souboru, používaném v aplikaci Simulink. Hlavní předností aplikace je využití v rámci algoritmu (GE) a v rámci grafické analýzy jednotlivých Simulink systémů. Aplikace jsou graficky prezentovány na příloze (příloha: X.14). Z logiky aplikací je jejich umístění následující, klientská aplikace vlastní Simulation Info generator a MDL file parser je v serverové části aplikace.



Výhoda uvedeného řešení, kdy server vlastní interní / externí výpočetní část, je schopnost serveru zcela ovládat Matlab a tak využít všech jeho předností a může se zcela soustředit jen na síťovou komunikaci mezi klienty. Další nespornou výhodou je možnost využití distribuovaných výpočtů v režimu $x$ klientů 1 server.

Příklad komunikace, kdy klient chce řešit optimalizaci regulátoru na zadaných parametrech pomocí daného algoritmu. Při inicializaci úlohy klient pošle serveru jen data, co chce řešit a na jakých parametrech. Server si přečte poslaná data a předá je výpočtové části (Matlab nebo Simulink), ta si vytvoří strukturu problému a definuje si návratovou fitness funkci. V průběhu chodu algoritmu klient jen posílá data parametrů serveru, ten je přeposílá výpočtové části a ta vrací hodnotu fitness, která se opět přes server vrací ke klientu. Jeden z možných příkladů komunikace mezi serverem a klientem a grafická prezentace rozdělení úlohy na výpočetní pod-úkoly a jejich logika při vytváření jsou prezentovány v příloze (příloha: X.13). Rozdělení výpočetního problému na pod-úkoly přináší zlepšení efektivního využití distribuovaných výpočtů a využití multi-thread procesorů, tedy se jedná o distribuovaný model vypočtu se vsemi přednostmi.

- Maximální obecné řešení oddělené od optimalizovaného problému, klient vůbec neřeší, jak se daný problém vypočítá.
- Velice lehká implementace nových algoritmů (vytvoření nového modulu v klient aplikaci).
- Možnost spuštění klientské aplikace na pomalém (obyčejném) PC a připojit se k serveru pomocí lokální sítě nebo internetu.
- Jednoduchá implementace paralelních výpočtů, kdy k jednomu serveru se mohou připojit vice klientů.
- Server i klient aplikace jsou napsány v jazyce java, tedy jsou plně mutiplatformní.
- Výpočetní část zahrnuje více vláknový výpočet dle funkce *matlabpool*.




**Obrázek 3.1:** Blokové schéma softwerového řešení (klient/server aplikace).

## 3.2 Design dokument

Hlavní myšlenka aplikace je rozdělení softwaru na klienta a server část, a z tohoto rozdělení vyplívají dva hlavní druhy mezi-softwarové komunikace ve formě socket a (RMI). Z hlediska design dokumentu je logika popsaná dle UML reprezentace pro RMI handler (příloha: X.2), (příloha: X.4) a socket handler (příloha: X.3). Z hlediska spouštění optimalizačních algoritmů z klientské aplikace je logika popsána dle (příloha: X.5). Hlavní výpočetní nástroj serverové aplikace je Matlab / Simulink, kdy obecný Matlab handler je popsaný dle (příloha: X.1) spolu se sekvenci logiky funkcí v samotné Matlab aplikaci (příloha: X.6). V aplikaci se využívají i externí knihovny ke zpracování výpočetních dat a různých typů komunikace, přehledová tabulka všech knihoven je následující:

| Název | Verze | Popis |
|---|---|---|
| matlabcontrol | 3.1.0 | Komunikace a logika ovládání Matlab aplikace. |
| jmi | 7.11 | Knihovna interní komunikace mezi Matlab s externí aplikací. |
| sigar | 2.6 | Systémové informace o použitém SW a HW. |
| jcommon | 1.0.16 | Obecná knihovna serializace a zpracování dat. |
| itext | 2.1.5 | Generování PDF dokumentů. |
| jchart | 1.0.13 | Zpracování grafových dat. |
| jdiagram | 1.4.10 | Zpracování diagramových dat. |
| speedJG | 1.0 | Zpracování XML souborů. |

**Tabulka 3.1:** Seznam použitých externích java knihoven.

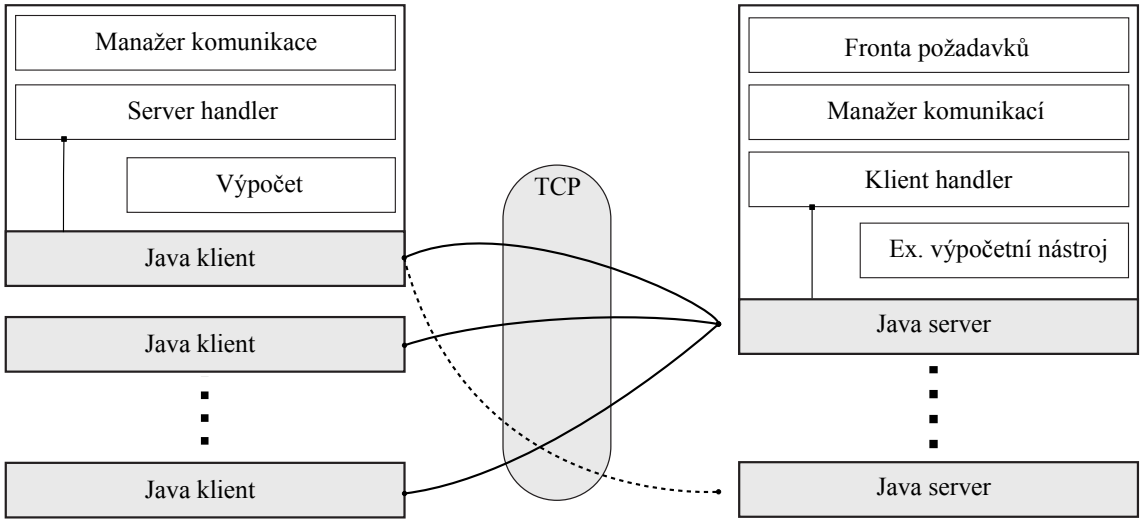
### 3.2.1 Mezi softwarová komunikace

Způsob komunikace mezi aplikacemi je rozdělen na dva druhy. První je místní komunikace, kdy server zajišťuje spojení mezi výpočtovou částí prostřednictvím protokolu Java Matlab Interface (JMI) a vzdálenou, tedy komunikace mezi serverem a klientem. Tato komunikace je dvojího druhu, první pomocí Remote Method Invocation (RMI), kdy klient volá přímo metody obsažené v serveru. (RMI) komunikace je jednostranná ve směru *klient* → *server*. Poslední využívaný protokol je Socket Interface, pro předávání zpráv mezi klientem a severem v textové podobě, tento typ komunikace je obousměrná *klient* ↔ *server*. Vzdálená komunikace lze použít i ve formě localhostu, tedy server i klient jsou spuštěny na jednom PC.

### 3.2.2 Možnosti aplikačního spojení



Mezi klientem a serverem existuje více možností propojení. První základní verze je spojení, ala localhost na jednom PC, kde je jeden klient připojený na jeden server. Tento způsob je vhodný kvůli oddělení výpočetní části s algoritmy od prosté simulace systémů. Další mozností je alokovaní samostatného výkonného PC jen pro server s výpočetní části a na tento server připojený více samostatných klientů. Výhoda tohoto zapojení je, možnost výpočtu v jednom okamžiku několika optimalizačních úloh, které úkolují jen výpočet modelů na externím server PC. Z hlediska času zabere většinu výpočet simulace modelu než samostatný chod algoritmu. Proto je možné spustit klienta třeba na pomalém PC (například notebook) a úkolovat sever, který je nainstalovány na velmi silném PC (workstation). Poslední možností je instalovaní více serverů na více PC a jeden klient připojit na těchto více serverů a při každé nové iteraci rozdělit populaci mezi tyto server počítače. Důvod je urychlení optimalizační úlohy, jak již bylo napsáno, většinu času zabere simulace a výpočet fitness na modelu než samotný chod algoritmu, proto tento způsob poskytuje nejlepší možnost úspory času. Princip jednotlivých zapojení je zobrazen na následujícím obrázku (obr: 3.2).



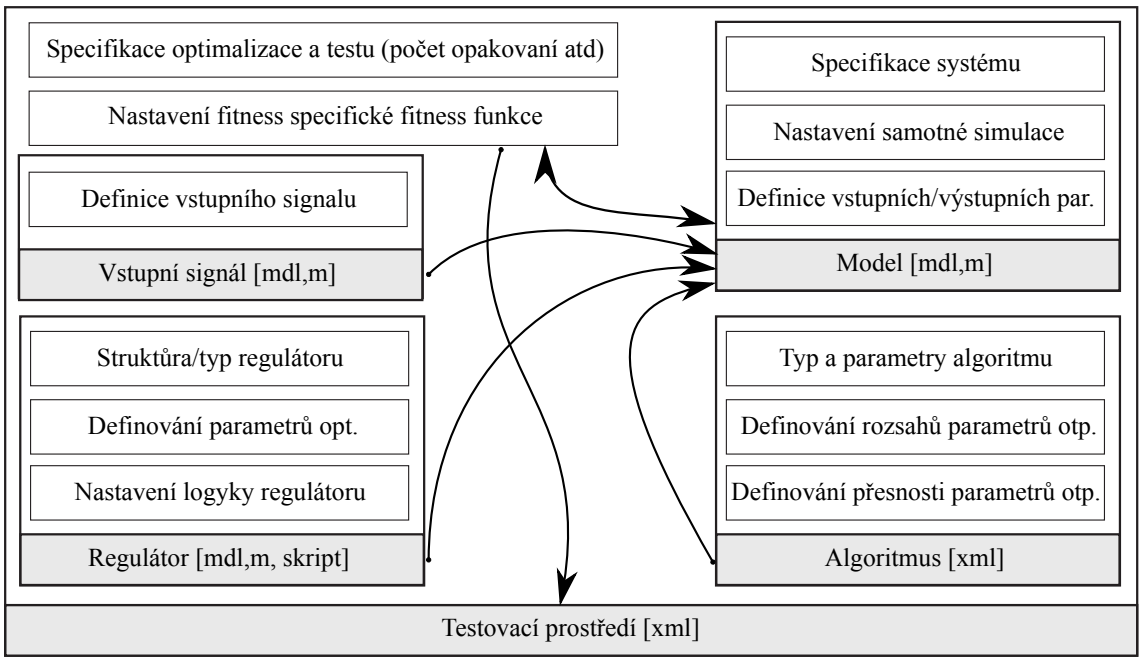

**Obrázek 3.2:** Možnosti aplikačního spojení mezi klientem a serverem.

### 3.2.3 Nastavení testovacího prostředí

Samotné nastavení optimalizace je uloženo v souboru typu XML. Tento definiční soubor obsahuje přesné specifikace optimalizace a nastavení simulace modelů. Celkové nastavení je rozděleno do pěti oddělených částí dle obrázku (obr: 3.3) . První obsahuje definici optimalizace, konkrétně počet opakování a definici fitness funkce. Druhá část je specifikace modelu s definicí vstupních a výstupních parametrů, definice simulace pro Matlab nebo Simulink a model systému. Třetí část je definice zadaného referenčního signálu pro simulaci. Čtvrtá část je definice modelu regulátoru spolu s parametry optimalizace a logiky regulace. Poslední definice je nastavení parametrů optimalizačního algoritmu, typu algoritmu a nastavení rozsahů parametrů pro optimalizaci s přesností parametrů.

Tento způsob nastavení testovacího prostředí je zvolen kvůli možnosti modulárního spouštění příkladů s rozdílnými definicemi a rozdílnými typy regulátorů. Další výhoda je možnost definování systému, regulátoru a referenčního signálu v různých typech výpočetní definice ($mdl$ nebo $m$) a různě je mezi sebou kombinovat.






**Obrázek 3.3:** Model nastavení XML specifikace výpočtu.

## 3.3 Možná aplikace v reálných soustavách

K řízení reálných soustav se využívají embedded systémy s PC, mikrokontroléry nebo programovatelné automaty (PLC). V případě této práce se využívá řízení pomocí (PLC) nebo přímo z PC. Pro řízení dle (PLC) je zvolen automat $X20CP1483$ od společnosti B&R. Tento automat je schopen provádět řídící smyčku rychlostí až 2 milisekundy. Technologie BR je zvolena také z důvodu lehké aplikovatelnosti výsledku optimalizace na vybrané (PLC) a to generováním kódu (C, C++) přímo z prostředí Simulink a k tomuto účelu se využívá toolbox B&R Automation Studio Target. V této práci je majoritně využíván jako výpočetní nástroj software Simulink, v kterém je i uložen výsledek optimalizace, proto výsledek je lehce aplikovatelný na

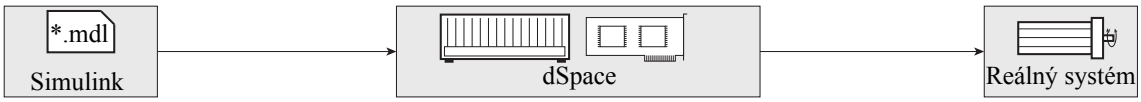
vybraný automat. Obrázek (obr: 3.4) je schéma využití simulace na reálné soustavě. Tento postup má několik předností.

- Možnost využití i velice složitých struktur výsledného regulátoru,
- jednoduchá aplikovatelnost simulačního modelu na reálnou soustavu,
- ušetření času při vývoji kódu pro programovatelné automaty,
- jeden kód pro celou řadu typů regulátorů.


**Obrázek 3.4:** Posloupnost kroků ke generování PLC kódu z prostředí Simulink.

K real-time řízení systémů lze využít technologii dSpace, a to jak klasické PC s přídavným HW, tak specializovaná real-time HW řešení. Tato technologie se používá k řízení a simulace v reálném čase, kombinuje se výkonný hardware a automatickým generováním kódu ze Simulink modelu. dSpace je velmi dobrý nástroj pro rychlý vývoj (rapid-prototyping) řídících systémů a simulací hardware-in-the-loop (HIL). Ovládací program je generován z prostředí Simulink dle knihovny realtime toolbox. Blokové schéma (obr: 3.5) představuje posloupnost kroků k řízení reálného systému dle technologie dSpace z prostředí Simulink. Hardware dSpace disponuje značným rozsahem výkonů od jednodeskového řešení až po multiprocesorové systémy. Pomocí této technologie je možno dosáhnout rychlosti řízení až v desítkách mikrosekund.






**Obrázek 3.5:** Posloupnost kroků k řízení reálného systému dle technologie dSpace.

## 3.4 Testování funkcionality aplikace

K ověření funkcionality aplikace bylo použito série testů, které dokládají schopnost aplikace najít výsledek a schopnost aplikace pracovat v různém prostředí s použitím různých výpočetních nástrojů bez efektu na konečný výsledek. Testy se prováděly na samostatné klientské aplikaci / nebo více aplikací spolu se zapojením serverové aplikace. Samotné nastavení aplikace je provedeno dle sekce (sekce: 3.2.3 - viz str. 36) a příklady nastavení jsou znázorněny v přílohách pro lokální nastavení (příloha: X.9), Matlab (příloha: X.10), Simulink (příloha: X.11) a dodatek k nastavení Simulink modelů (příloha: X.12).

### 3.4.1 Schopnost nalezení výsledku

První série testů je zaměřena na schopnost nalezení optimálního výsledku, při použití různých algoritmů vyhledávání s měřením výsledného času výpočtu a počtu výsledných iterací algoritmu. Použité algoritmy v testu jsou (HC12), (DE), (GA) a (NM). Testovací funkce jsou zvoleny dvě z balíčku standardních funkcí k ověřování funkčnosti algoritmů a jsou to Rastriginova funkce (sekce: 3.4.1 - viz str. 38) a Schwefelova funkce (sekce: 3.4.1 - viz str. 38). Celkový test je prováděn na čtyřech

případových studií dle obrázku (obr: 3.6). První verze je výpočet algoritmu v rámci lokálního prostředí klientské aplikace (Java), s vyloučením externího výpočetního nástroje, tabulky výsledků jsou (tab: 3.2) a (tab: 3.6). Druhá verze je podobná s první verzí, jen s rozdílem zapojení externího výpočetního nástroje (Matlab) na serverové straně, tabulky výsledků jsou (tab: 3.3) a (tab: 3.7). Třetí verze je test funkcionality serverové aplikace zpracovávat požadavky od více klientů v jednom čase, test obsahuje dvě klientské aplikace počítající stejný problém a propojené na stejný server, tabulky výsledků jsou (tab: 3.4) a (tab: 3.8). Poslední verze testu je podobný s předešlou verzí s rozdílem, že klientské aplikace jsou spuštěné na různých PC, tabulky výsledků jsou (tab: 3.5) a (tab: 3.9). Celkové výsledky jsou znázorněny v grafu (graf: 3.3) pro Rastriginovu funkci a v grafu (graf: 3.4) pro Schwefelovu funkci.

### Rastriginova funkce



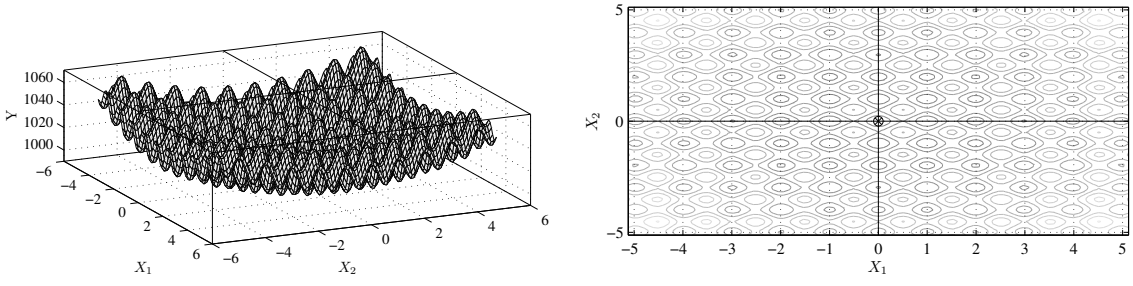
Rastriginova funkce je založena na testovací funkci dle De Jonga s rozšířením o cosinovou modulací s požadavkem o navýšení počtu lokálních minim. Ve výsledku je Rastriginova funkce vysoce multimodální, ale lokace lokálních minim je deterministicky distribuovaná. Rastriginova funkce má následující definici (vz: 3.1) a její průběh je v grafech (graf: 3.1a) a (graf: 3.1b) pro $d = 2$.

Testovací area je omezena na oblast $-5.12 \leq x_i \leq 5.12, i = 1, \ldots, d$ s globálním minimem v hodnotě $f(x^*) = 0$ pro hodnoty parametru $x^* = (0, \ldots, 0)$.

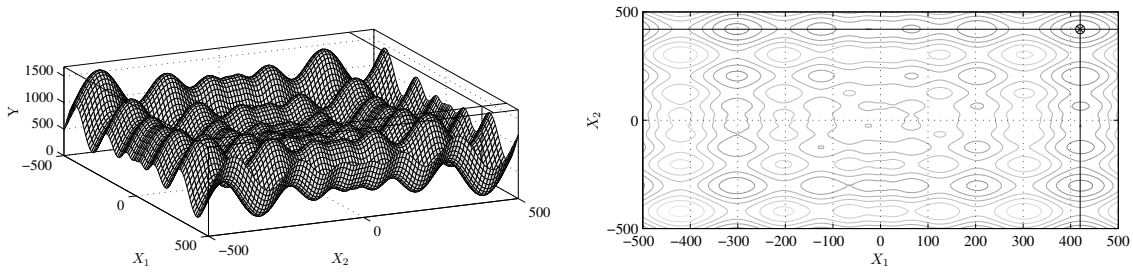
### Schwefelova funkce

Schwefelova funkce je vysoce multimodální a komplexní funkce na testovaní optimalizačních algoritmů. Speciální vlastností této funkce je poloha globálního minima a je geometricky vzdálená od následné polohy nejlepšího z lokálních minim, pro tuto vlastnost je Schwefelova funkce z pohledu optimalizačních algoritmů velice ošidná z důvodu možnosti konvergence ve špatném směru vyhledávaní. Definice Schwefelovy funkce je znázorněna na definici (vz: 3.2) a grafech (graf: 3.2a) a (graf: 3.2b) pro $d = 2$.

Testovací oblast je omezena na $-500 \leq x_i \leq 500, i = 1, \ldots, d$ s globálním minimem v hodnotě $f(x^*) = 0$ a pro hodnoty parametru $x^* = (420.9687, \ldots, 420.9687)$.




**Graf 3.1:** Testovací funkce 1. Pořadí grafů: (a: Rastriginova funkce.) a (b: Konturový graf.).




**Graf 3.2:** Testovací funkce 2. Pořadí grafů: (a: Schwefelova funkce.) a (b: Konturový graf.).

| ⊞ | HC12 | | DE | | GA | | NM | |
|---|---|---|---|---|---|---|---|---|
| | Čas | Iterace | Čas | Iterace | Čas | Iterace | Čas | Iterace |
| 2 | 15.741 | 21 | 24.244 | 36 | 24.165 | 30 | 23.208 | 24 |
| 5 | 45.339 | 27 | 49.173 | 53 | 41.078 | 44 | 55.275 | 41 |
| 10 | 120.049 | 76 | 89.021 | 93 | 78.042 | 88 | 201.428 | 88 |

**Tabulka 3.2:** Rastriginova funkce:Test 1.

| ⊞ | HC12 | | DE | | GA | | NM | |
|---|---|---|---|---|---|---|---|---|
| | Čas | Iterace | Čas | Iterace | Čas | Iterace | Čas | Iterace |
| 2 | 42.180 | 18 | 68.112 | 24 | 59.664 | 24 | 46.092 | 23 |
| 5 | 109.980 | 38 | 100.548 | 48 | 122.877 | 35 | 130.130 | 43 |
| 10 | 318.240 | 71 | 245.818 | 81 | 229.554 | 90 | 412.854 | 81 |

**Tabulka 3.3:** Rastriginova funkce:Test 2.

| ⊞ | HC12 | | DE | | GA | | NM | |
|---|---|---|---|---|---|---|---|---|
| | Čas | Iterace | Čas | Iterace | Čas | Iterace | Čas | Iterace |
| 2 | 98.955 | 24 | 144.240 | 26 | 127.344 | 20 | 142.876 | 24 |
| 5 | 204.570 | 41 | 273.714 | 53 | 208.116 | 56 | 276.925 | 56 |
| 10 | 735.360 | 72 | 371.219 | 79 | 402.870 | 86 | 1104.696 | 74 |

**Tabulka 3.4:** Rastriginova funkce:Test 3.

| ⊞ | HC12 | | DE | | GA | | NM | |
|---|---|---|---|---|---|---|---|---|
| | Čas | Iterace | Čas | Iterace | Čas | Iterace | Čas | Iterace |
| 2 | 103.365 | 14 | 187.056 | 32 | 144.528 | 25 | 132.664 | 26 |
| 5 | 299.925 | 31 | 256.270 | 52 | 319.349 | 51 | 386.705 | 45 |
| 10 | 691.320 | 87 | 696.959 | 90 | 581.412 | 71 | 946.107 | 61 |

**Tabulka 3.5:** Rastriginova funkce:Test 4.

| ⊞ | HC12 | | DE | | GA | | NM | |
|---|---|---|---|---|---|---|---|---|
| | Čas | Iterace | Čas | Iterace | Čas | Iterace | Čas | Iterace |
| 2 | 19.034 | 25 | 31.537 | 35 | 29.980 | 29 | 25.683 | 33 |
| 5 | 54.842 | 31 | 55.483 | 46 | 42.514 | 38 | 60.465 | 59 |
| 10 | 141.015 | 86 | 92.466 | 88 | 81.498 | 89 | 213.864 | 76 |

**Tabulka 3.6:** Schwefelova funkce:Test 1.

| ⊞ | HC12 | | DE | | GA | | NM | |
|---|---|---|---|---|---|---|---|---|
| | Čas | Iterace | Čas | Iterace | Čas | Iterace | Čas | Iterace |
| 2 | 50.122 | 16 | 63.302 | 35 | 80.910 | 31 | 64.525 | 28 |
| 5 | 125.334 | 42 | 124.575 | 48 | 96.390 | 43 | 175.620 | 53 |
| 10 | 339.951 | 88 | 239.752 | 89 | 226.152 | 89 | 629.628 | 73 |

**Tabulka 3.7:** Schwefelova funkce:Test 2.

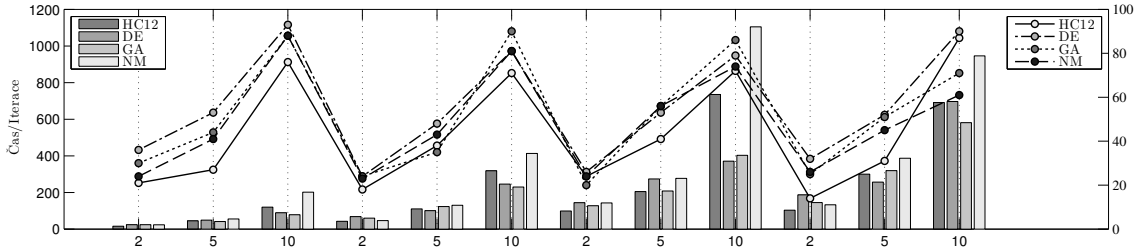
| ⊞ | HC12 | | DE | | GA | | NM | |
|---|---|---|---|---|---|---|---|---|
| | Čas | Iterace | Čas | Iterace | Čas | Iterace | Čas | Iterace |
| 2 | 91.048 | 18 | 153.419 | 33 | 146.189 | 23 | 151.675 | 26 |
| 5 | 256.878 | 55 | 254.705 | 47 | 240.492 | 54 | 411.600 | 55 |
| 10 | 684.273 | 85 | 400.016 | 81 | 472.149 | 101 | 867.762 | 71 |

**Tabulka 3.8:** Schwefelova funkce:Test 3.

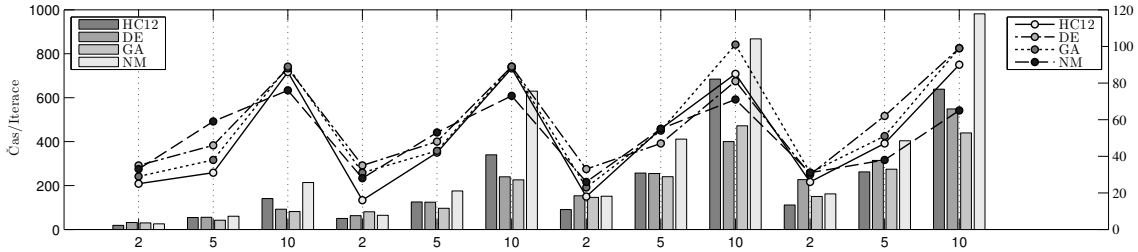
| ⊞ | HC12 | | DE | | GA | | NM | |
|---|---|---|---|---|---|---|---|---|
| | Čas | Iterace | Čas | Iterace | Čas | Iterace | Čas | Iterace |
| 2 | 111.758 | 26 | 227.261 | 30 | 150.626 | 31 | 162.400 | 31 |
| 5 | 262.062 | 47 | 314.380 | 62 | 274.302 | 51 | 403.680 | 38 |
| 10 | 638.871 | 90 | 548.596 | 99 | 439.506 | 99 | 981.717 | 65 |

**Tabulka 3.9:** Schwefelova funkce:Test 4.






**Graf 3.3:** Graf závislosti času a iterací pro test s Rastriginovou funkcí.




**Graf 3.4:** Graf závislosti času a iterací pro test se Schwefelovou funkcí.

| ⊞ | Nastavení |
|---|---|
| HC12 | 10bit na parametr |
| DE | populace 200, 500 generací |
| GA | populace 200, 10bit na parametr, křížení 2, 500 generací |
| NM | 500 generací |
| GE | 20 kodonů, 8bit na kodon, křížení 2, 500 generací |

**Tabulka 3.10:** Nastavení algoritmů.

Z výsledků je patrné, že nejrychlejší způsob výpočtu je lokální výpočet bez použití externího nástroje, který zatěžuje výsledný čas komunikací se serverovou stranou, tak samotná rychlost výpočtu (Matlab/Simulink). Lokální výpočet je vhodný pro jednoduché problémy, které se dají jednoduše napsat na klientské straně (např. hledání minima funkce), na vše ostatní je vhodné použití externího nástroje, který přináší větší flexibilitu i za cenu zvětšení výsledného času výpočtu. Poslední testy dokládají, že není velký rozdíl kde jsou klientské aplikace spuštěné, protože hlavní část času zabírá výpočet na serverové straně, tedy čas pro dva klienty na stejném PC a dva klienty na různých PC jsou podobné. Výsledný čas a iterace jsou zprůměrované ze 100 opakování algoritmu na dané konfiguraci. Tabulka nastavení algoritmů (tab: 3.10).

$$fce1(x) = 10d + \sum_{i=1}^{d} \left[ x_1^2 - 10cos(2\pi x_i) \right] \tag{3.1}$$

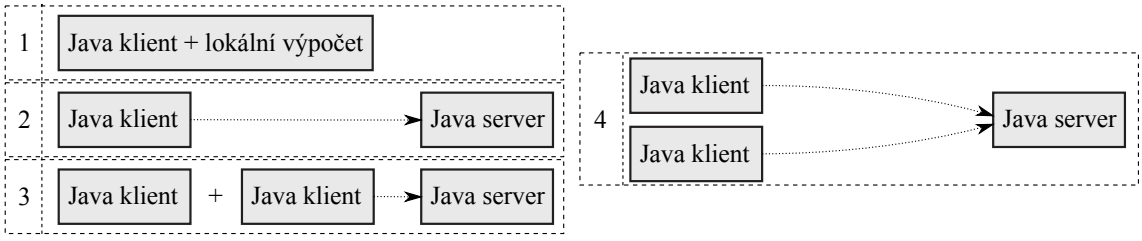
$$fce2(x) = 418.9829d - \sum_{i=i}^{d} x_i sin\left(\sqrt{|x_i|}\right) \tag{3.2}$$



### 3.4.2 Testování platformové jednoty

Druha série testů aplikačního řešení je zaměřena na otestování platformové jednoty, tedy ověření, že výpočetní nástroj nemá vliv na kvalitu výsledku, při jednotném nastavení optimalizace. Test je prováděn na třech různých výpočetních prostředí, první je lokální výpočet v rámci klientské aplikace (Java), druhé je externí nástroj (Matlab) a poslední prostředí je externí nástroj (Simulink) v rámci server aplikace. Testovaná úloha je testovací příklad z gramatické evoluce, při hledání matematického zápisu neznámé funkce. Hledaná funkce je ve tvaru (vz: 3.3), definice gramatické evoluce je dle gramatiky (gramatika: 3.1) a nastavení samotného algoritmu je v tabulce (tab: 3.10).

$$fge(x) = (cos(x+1) + sin(x)) * cos(2 * x) \tag{3.3}$$




**Obrázek 3.6:** Grafické znázornění případových studií.

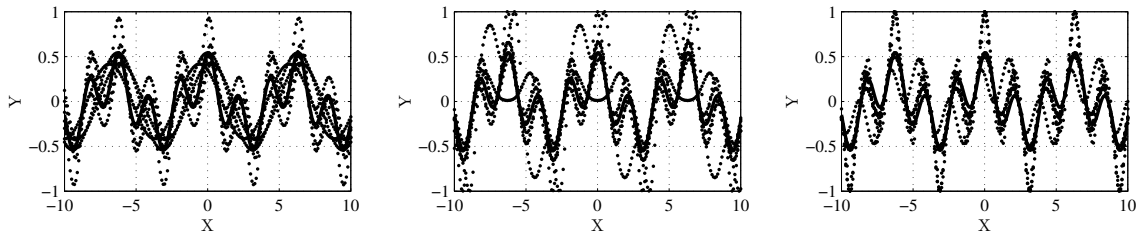
| ⊞ | Čas PC1 | Iterace PC1 | Čas PC2 | Iterace PC2 | Výsledek |
|---|---|---|---|---|---|
| Lokal | 548 | 12 | 784 | 15 | (graf: 3.5a) |
| Matlab | 981 | 18 | 1243 | 21 | (graf: 3.5b) |
| Simulink | 1267 | 15 | 1695 | 14 | (graf: 3.5c) |

**Tabulka 3.11:** Výsledky gramatické evoluce pro hledání obecného zápisu funkce.

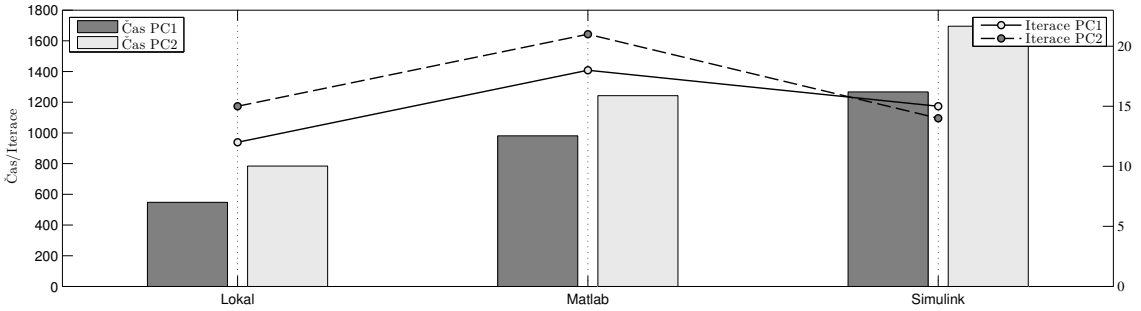
```
        N = { expr, op, pre_op, var} | T = {sin, cos, +, -, *, /, X, 1, 2 }
        S = { expr }
<expr>  ::<expr> <op> <expr> | (<expr> <op> <expr>) | <pre_op> (<expr>) | <var>
<op>    ::+ | - | / | *
<pre_op>::sin | cos
<var>   ::X | 1 | 2

genetype    phenotype                                                       system
sin /cos    Math.sin() / Math.cos()                                         Local
sin /cos    sin() / cos()                                                   Matlab
sin /cos    Block.Trigonometry.Operator=$sin$ / Block.Trigonometry.Operator=$cos$   Simulink
```

**Gramatika 3.1:** Obecná gramatika k nalezení testovací funkce.



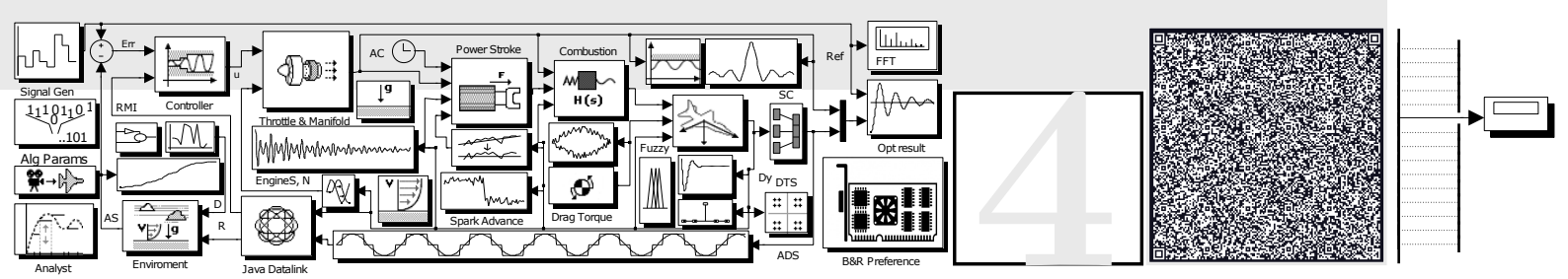

**Graf 3.5:** Výsledky gramatické evoluce na hledaní zápisu matematické funkce. Pořadí grafů: (a: Lokal), (b: Matlab) a (c: Simulink).




**Graf 3.6:** Graf závislosti času a iterací pro test s GE.

Konečné výsledky jsou prezentovány v tabulce (tab: 3.11) spolu s odkazy na vybrané výsledky v podobě grafů finálních funkcí. Celý výpočet, pro všechny prostředí, byl spuštěn na dvou různých PC sestavách k porovnání výsledků časové náročnosti aplikace a porovnání konečných výsledků optimalizace. Počítačové konfigurace pro $PC1$ je Intel Core i5-750 CPU (8M Cache, 2.66 GHz), 4GB RAM a pro $PC2$ je Intel Core i3-370M CPU (3M cache, 2.40 GHz), 2GB RAM. Hodnoty časové náročnosti a finálního řešení jsou znázorněny na grafu (graf: 3.6). Z finálních výsledků je prezentováno oddělení algoritmu od výpočetní části, kdy výpočet běží na lokálním prostředí, Matlab nebo Simulink nastavení a dosahuje velmi podobných výsledků. Výsledný čas a iterace jsou zprůměrované ze 100 opakování algoritmu na dané konfiguraci.

Výsledná aplikace lze tedy použít pro následující příklady a modely prezentované dále v této práci, s tím že kvalita optimalizace není ovlivněna na základě použitého výpočetního nástroje. Aplikace je napsaná v modulárním konvenci a lze jednoduše rozšířit na nové typy problémů a výpočetní nástroje pomocí konfiguračních souborů, prezentovaných v přílohách.




# KAPITOLA 4

# APLIKACE VÝSLEDKŮ NA TESTOVACÍCH SOUSTAVÁCH



Kapitola aplikace výsledků na testovacích soustavách je zhodnocení výsledků použitých algoritmů v rámci teorii řízení nelineárního řízení komplexních systému a aplikování evolučních algoritmů na různé typy modelů a prezentuje výsledek této práce na dané téma. Celá kapitola je založena na autorových předešlých pracích (Seznam vlastních publikací - viz str. 137) a rozšiřuje způsob řešení o nové přístupy. Využívaní evolučních algoritmů v teorii řízení a dalších oborech technické praxe je stále relativně nový vědní obor, který poskytuje mnohé možnosti, jak zlepšit stávající přístupy k řešení optimalizačních úloh. Hlavní cil této sekce je vybrat takové modely a takové příklady na kterých bude jednoznačně demonstrováno jejich přednosti a jejich možnosti, jak při využívaní ve standartních přístupech, tak při aplikování nových přístupů řešení.

Nové přístupy v teorii řízení v této kapitole jsou zaměřeny k využití vícekriteriální optimalizace, kdy standartní postupy jsou zaměřeny většinou na jednu dominantní vlastnost a podle ní se provádí samotná optimalizace. U vícekriteriálního postupu se nezaměřuje jen na jednu vlastnost, ale snaží se příklad řešit ve více rovinách ohodnocení. Tento přístup přináší lepší výsledky a robustnější nastavení výsledných regulátorů. S využitím znalostí z řízení systémů je vícekriteriální optimalizace dále úspěšně aplikována i na řešení stability a navrhovaní struktury. Některé další postupy v této problematice jsou prezentovány v autorových

předešlých pracích. Zhodnocení všech výsledků je na konci každé kapitoly a celkové porovnaní na přič všemi regulátory je v kapitole (sekce: 8 - viz str. 107) a představuje souhrnný výsledek této práce. Všechny modely a analýza jsou provedeny v aplikaci Matlab nebo Simulink, jak bylo prezentováno v předešlé praktické sekci.

Praktická sekce této práce je zaměřena na testování algoritmů (SC) na testovacích modelech. Modely zahrnují referenční příklady k testování účelnosti hodnoticí funkce až po modely reálných soustav, které si kladou za cíl ověření výsledku optimalizace na modelu i pro skutečnou soustavu. Využité algoritmy k optimalizaci jsou (HC12) jako hlavní algoritmus, spolu s porovnáním na (DE) jako jeden z nejvíce rozšířeným (SC) algoritmem a (NM) jako představitelem základního typu algoritmu. Dále je v práci je využita (GE) k nastavení vlastní struktury regulátoru a porovnaní této metody s klasickým přístupem nastavení regulátoru.

| ⊞ | Syntax | X | Y | Z |
|---|---|---|---|---|
| Verze 1 | A:<X,Y> | Začátek intervalu | Konec intervalu | - |
| Verze 2 | B:<X,Y,Z> | | | Počet bitů |
| Verze 3 | C:<X,Y> | Počet kodonů | Počet bitů | - |

**Tabulka 4.1:** Ukázka zápisu nastavení parametrů optimalizace.



Typy příkladů optimalizací jsou nastavení parametrů (PID) regulátoru dle předem daných okrajových podmínek - klasický přístup a slouží k porovnání vhodnosti algoritmu a vhodnosti nastavení fitness funkce a dále je tento příklad nastavení regulátoru využit jako referenční bod pro pokročilejší nastavení dle dalších možností. Další příklady jsou fuzzy (PID) regulátor z možností nastavení jen čistého (PID) s lineárním nastavením funkcí příslušností spolu s nastavením funkcí příslušností a kombinace těchto možností. Pokročilým příkladem k nastavení regulátoru je využití (GE) k nastavení vlastní struktury a s možností zpětné optimalizace jednotlivých parametrů. Celkově tyto příklady představují přehledné možnosti využití (SC) algoritmů k optimalizaci nastavení regulátorů jak jednoduchých, tak složitých soustav v teorii řízení.

Extra kapitoly v této práci pojednávají o speciálním způsobu řízení složitých soustav, a to konkrétně ukázka stabilizace chaotického systému zvaného jako logistická mapa. Na tomto příkladu je ukázáná možnost použiti genetických algoritmů, jak na optimalizaci stávajícího způsobu stabilizace, tak generovaní vlastních pravidel a navržený způsob doladění generovaných pravidel dle specifického algoritmu. Stabilizaci chaotického systému se ještě věnuje příklad stabilizace duffinovy mapy, která rozšiřuje postupy při řešení stabilizace podobných systémů. Hlavní zaměření této kapitoly je na různé využití upravené Pyraglasovy metody. Poslední z předaných kapitol řešených příkladů je návrh modelu specifických antén, v tomto případě typu Yagi-Uda antény. Kapitola je zaměřena na hledání tvaru antény při zohlednění více hledisek a aplikaci nové fitness funkce.

Každá z kapitol práce obsahuje zpětné zhodnocení výsledků jak regulace, stabilizace a návrhnu modelu na různé typy algoritmů k referenčnímu nastavení popsané v této kapitole.

Kvůli zjednodušení a zachování jmenné konvence pro všechny části vypočtu se

jednotlivé příklady a sekce této kapitoly nazývají svými kódovými názvy a celkový popis všech použitých systémů je obsažen v sekci (sekce: 4.2 - viz str. 47).

- Popis testovacích modelů jako **Popis-systemu**.
- Nastavení PID regulátoru jako **PID**.
- Nastavení FUZZY PID regulátoru jako **FUZZY**.
- Generovaní struktury regulátoru jako **GE**.
- Příklad stabilizace logistické mapy jako **Chaos-LM**.
- Příklad stabilizace duffingovy rovnice jako **Chaos-DF**.
- Příklad návrhu antény jako **Yagi**.

Nastavení jednotlivých parametrů optimalizace je popsáno pro každý typ modelu a algoritmu pomocí třech druhů zápisu. Popis syntaxe nastavení je v tabulce (tab: 4.1). Popis nastavení optimalizačních algoritmů se skládá z klíčových proměnných a jejich kompletní seznam je uveden v tabulce (tab: 4.2).

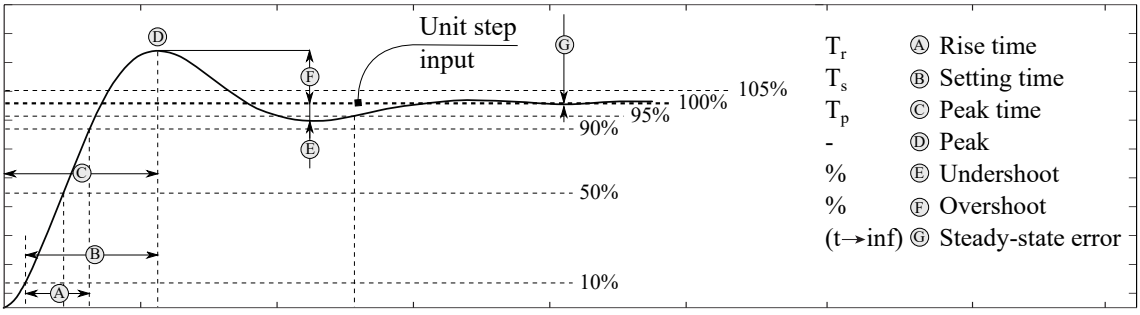
| Zkratka | Význam |
|---|---|
| RUNS | Počet opakování algoritmu |
| NP | Velikost populace (řešení v jedné iteraci) algoritmu |
| F | Koeficient mutace, použití dle konkrétního algoritmu |
| CR | Koeficient křížení |
| GEN | Počet iterací v jednom chodu algoritmu |
| R | Koeficient přeživších jedinců v populaci |
| PC | Body křížení v operaci křížení |
| W | Počet opakování binárního řetězce v gramatické evoluci |

**Tabulka 4.2:** Popis parametrů nastavení algoritmu optimalizace.



## 4.1 Ohodnocení výsledků

K hodnocení kvality regulace je využita standardní metoda, a to metoda skokové odezvy uzavřeného regulačního obvodu. Tato metoda je zaměřena na zohlednění sedmi základních charakteristik, které jsou popsány na následujícím obrázku (obr: 4.1). K výpočtu hodnot jednotlivých charakteristik je použita funkce *stepinfo* zahrnuta v programu Matlab. Funkce je definována se základním nastavením pro všechny typy příkladů i pro všechny typy modelů kvůli jednoznačnosti porovnaní výsledků.




**Obrázek 4.1:** Ohodnocení přechodové charakteristiky.

$$fitness = A \times log_{10}(f_{ITAE}) + B \times log_{10}(f_{Overshot} + 1) + C \times log_{10}(f_{Wave} + 1) \quad (4.1)$$

### 4.1.1 Fitness funkce

Pro účely optimalizace byla vytvořena speciální fitness funkce s možností kompenzace různých obecných faktorů, které se obecně vyskytují při hodnocení přechodové funkce. Cela hodnotící funkce je zaměřena na přechodovou charakteristiku dle zadaného vstupu a skládá se primárně ze tří částí, dle obrázku (obr: 4.2).

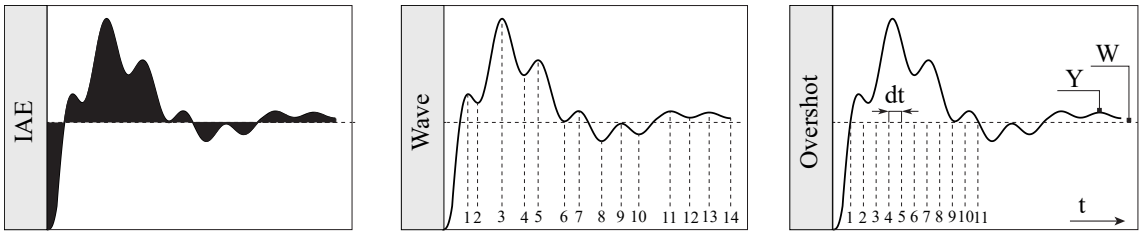
- **ITAE** - funkce itae je klasická hodnoticí funkce založena na funkci (IAE) v časové oblasti viz definice dle obrázku (obr: 4.2) sekce $IAE$ a definice $f_{ITAE}$ (vz: 4.2). Funkce představuje součet integrační plochy výsledné přechodové funkce oproti zadanému vstupnímu signálu a vynásobená časem. Další možností jsou využití klasické (IAE), nebo (ISE) s větším důrazem na kvadratickou plochu, případě v časové oblasti (ISTAE). Z testování je zvolena (ITAE) kvůli nejlepším výsledkům při vyhodnocení kvality regulace.
- **Wave** - kompenzační část hodnotící funkce k eliminaci kmitání přechodové funkce. Funkce je založena na sumarizaci kmitání a princip je znázorněný na obrázku (obr: 4.2) sekce $Wave$ spolu s definicí $f_{Wave}$ (vz: 4.2).
- **Overshot** - funkce overshot je kompenzátor přechodové funkce na překmit a je založena na časové sumarizaci hodnoty ve stavu kdy přechodová funkce je v překmitu. Princip je znázorněný na obrázku (obr: 4.2) sekce $Overshot$ a definice $f_{Overshot}$ (vz: 4.2).



$$
\begin{aligned}
f_{ITAE} &= \int_0^t t|e(t)|dt \\
f_{Wave} &= n \text{ kde } \frac{d}{dt}e(\{t_1 \dots t_n\}) = 0 \\
f_{Overshot} &= \sum_{i=1}^{n-1} \frac{(t_{i+1} - t_i)}{dt}, i+1 \text{ kde } e(\{t_1 \dots t_n\}) = W
\end{aligned}
\tag{4.2}
$$

Kompletní fitness funkce je zapsaná ve tvaru (vz: 4.1) a jednotlivé charakteristické prvky jsou hodnotami logaritmické funkce při základu deset, kvůli eliminaci nežádoucích hodnot. Jednotlivé prvky fitness funkce se dají ovlivňovat pomoci váhových proměnných ($A$, $B$, $C$), které určí váhu určité oblasti ve vícekriteriální oblasti optimalizace, rozmezí váhových proměnných je mezi $< 0, 1 >$.

Do výsledné fitness funkce se dají zanést i další podpůrné parametry specifické pro daný typ optimalizace a modelu jako je například filtrovaní přechodové funkce na extrémní hodnoty, nebo zadaný časový údaj náběhu, ale kvůli obecnosti a použitelnosti a porovnatelnosti pro všechny typy příkladů a modelů je využita verze fitness funkce dle (vz: 4.1).




**Obrázek 4.2:** Prvky fitness funkce.

## 4.2 Popis testovacích modelů

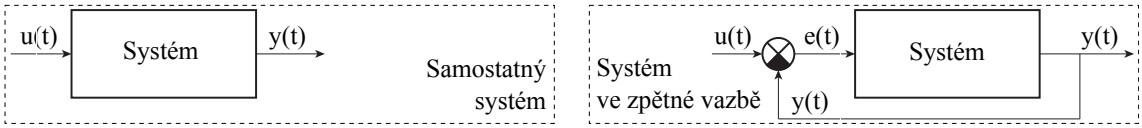
Tato sekce je zaměřena na popis jednotlivých modelů v rámci tohoto testování. Popis je míněný jako stručný úvod k jednotlivým modelům a představení jednotlivých vlastností a typu využití modelů. U každého popisu, jestli je to možné, je uvedeno i referenční nastavení regulátoru typu (PID). Následující modely byly vybrány, aby co nejvíce obsahovaly největší množství vlastností při regulaci systémů, lineární systémy, nelineární modely reálných systémů až po diskrétní řízení. Pro jednoduchost jsou modely označovány jejich kódovými názvy a to:

- Model čtvrtého řádu jako **system4**.
- Model čtvrtého řádu s nelineárním prvkem jako **system4delay**.
- Model kuličky v obruči jako **ballandloop**.
- Model magnetické levitace jako **maglev**.
- Logistická mapa jako **logisticmap**.
- Duffingova rovnice jako **duffing**.
- Yagi-Uda anténa jako **yagi**.

Pro modelování systému je využit software Matlab a pro určité typy optimalizace je využit jeho nástavba Simulink, dle nastavení popsaných u jednotlivých modelů, kvůli jednoznačnosti simulace.



Z důvodu určení stability systému je model testován na skokový signál pomocí obrázku (obr: 4.3), v samostatném obvodu nebo v obvodu ve zpětné vazbě.




**Obrázek 4.3:** Schéma zapojení systému k testovacím účelům na vstupní signál.

### 4.2.1 Model čtvrtého řádu

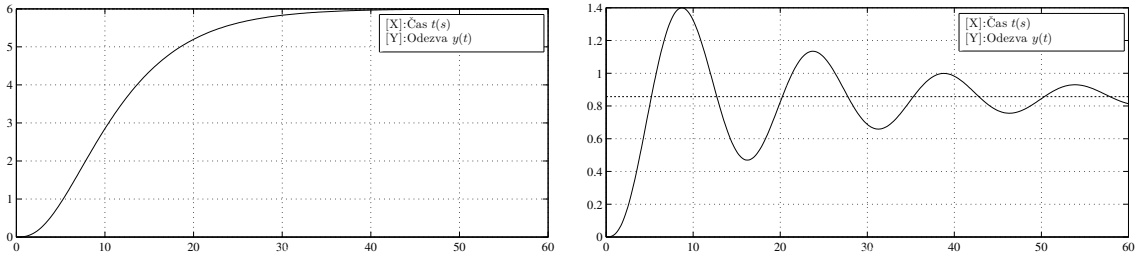
První systém určený pro testování optimalizačních metod a kvalitu fitness funkce je zvolen jednoduchý systém čtvrtého řádu. Systém je sám osobě stabilní a má sloužit primárně jako referenční bod pro nastavení základních parametrů optimalizace. Systém je popsán diferenciální rovnicí (vz: 4.3) anebo popřípadě přenosem (vz: 4.4). Systém má ještě jednu velkou výhodu, lze na něm jednoduše demonstrovat porovnání se základními druhy nastavení regulátorů. Systém je uvažován v jeho základní podobě tedy bez dopravního zpoždění $T_D = 0$ sek.

$$48y'''(t) + 44y''(t) + 12y'(t) + y(t) = 6u(t - T_D) \tag{4.3}$$

$$G(s) = \frac{6}{48s^3 + 44s^2 + 12s + 1} e^{-T_D s} \tag{4.4}$$

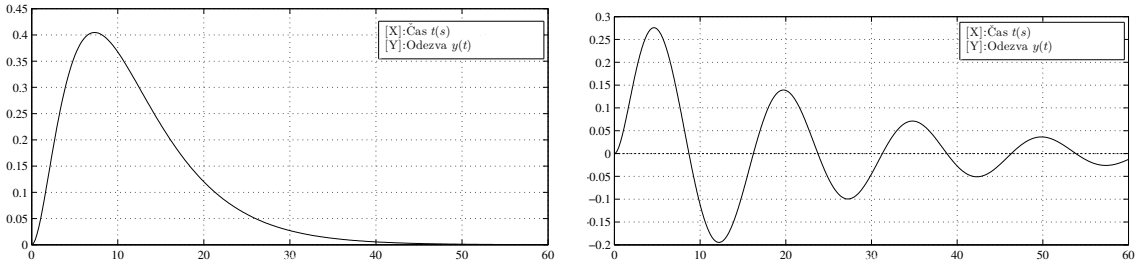
Charakteristiky obecné přechodové funkce čtvrtého řádu jsou zobrazeny na následujících grafech, konkrétně odezva systému na skokový signál (obr: 4.3) (graf: 4.1a), odezva systému na skokový signál ve zpětné vazbě (obr: 4.3) (graf: 4.1b), odezva

systému na impulzní signál (obr: 4.3) (graf: 4.2a), odezva systému na impulzní signál ve zpětné vazbě (obr: 4.3) (graf: 4.2b), Root-Locus charakteristika (graf: 4.3a), Nyquistova frekvenční charakteristika (graf: 4.3b), které znázorňují chovaní matematického modelu spolu se základním nastavením (PID) regulátoru (sekce: 1.2.1 - viz str. 9) dle Zeigler-Nichols (sekce: 1.2.2 - viz str. 10) a metody optimálního modulu (graf: 4.4a), (tab: 4.4) a ohodnocení skokové odezvy (obr: 4.1) na systém s regulátorem spolu s ohodnocením fitness funkce pomocí nastavení (sekce: 4.1.1 - viz str. 46) $A = 1$, $B = 1$, $C = 1$ (tab: 4.3).

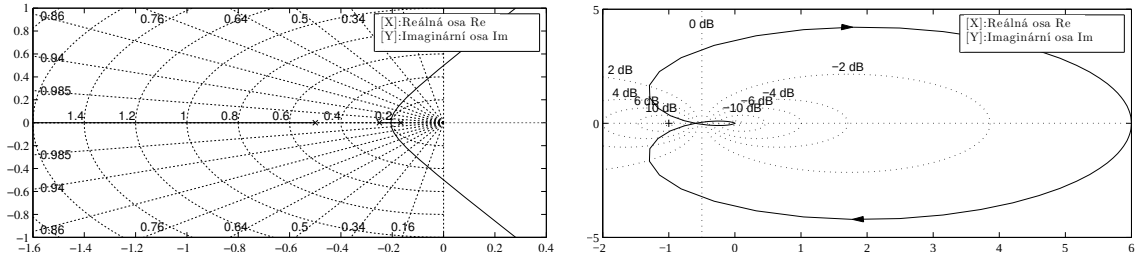



**Graf 4.1:** Pořadí grafů: (a: Odezva systému na jednotkový skokový signál pro System4.) a (b: Odezva systému na jednotkový skokový signál pro System4 ve zpětné vazbě.).






**Graf 4.2:** Pořadí grafů: (a: Odezva systému na jednotkový impulzní signál pro System4.) a (b: Odezva systému na jednotkový impulzní signál pro System4 ve zpětné vazbě.).



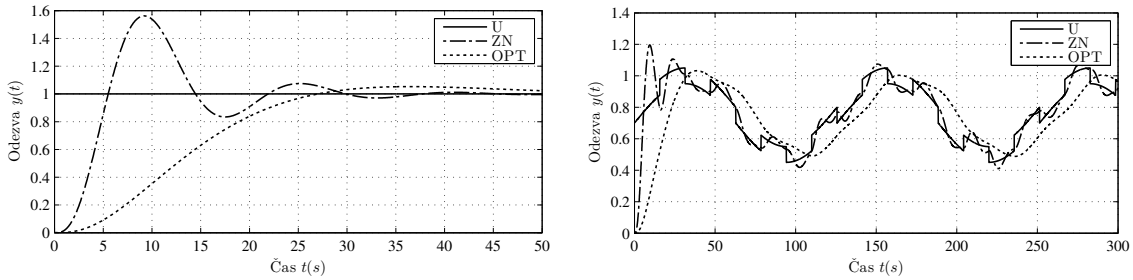

**Graf 4.3:** Pořadí grafů: (a: Root-Locus charakteristika pro System4.) a (b: Nyquistova frekvenční charakteristika pro System4.).

| ⊞ | $T_r$ | $T_s$ | $Min_s$ | $Max_s$ | Over | Under | Peak | $T_p$ |
|---|---|---|---|---|---|---|---|---|
| Výs. [ZN] | 3.234 | 35.203 | 0.834 | 1.562 | 56.200 | 0 | 1.562 | 9.200 |
| Výs. [OPT] | 16.818 | NaN | 0.901 | 1.053 | 5.268 | | 1.053 | 36.500 |

**Tabulka 4.3:** Tabulka ohodnocení odezvy na skokový signál dle nejlepších výsledků [Regulátor Popis-systemu Alg GENERAL] pro System4.

| ⊞ | P | I | D | Fit. | ITAE sin/pwm |
|---|---|---|---|---|---|
| Výs. [ZN] | 0.990 | 0.159 | 1.575 | 4.825 | 1363.587 |
| Výs. [OPT] | 0.083 | 0.166 | 1.333 | 4.047 | 2785.528 |

**Tabulka 4.4:** Tabulka nastavení regulátoru [Regulátor Popis-systemu Alg GENERAL] pro System4.




**Graf 4.4:** Pořadí grafů: (a: Průběh přechodové charakteristiky dle obecného nastavení pro System4.) a (b: Průběh přechodové charakteristiky na SIN/PWM signálu obecného nastavení pro System4.).

### 4.2.2 Model čtvrtého řádu s nelineárním prvkem

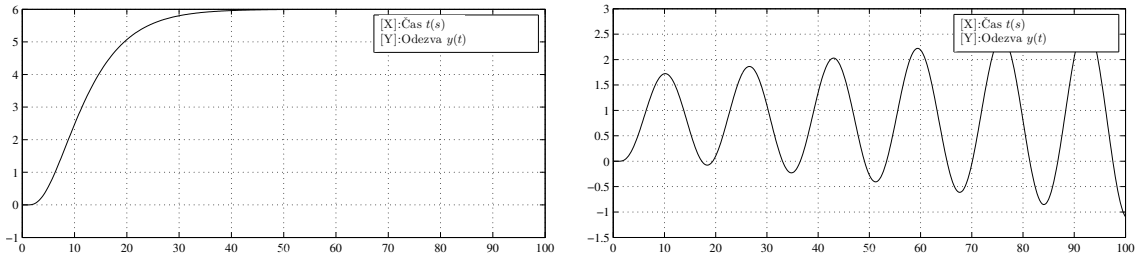
Systém s dopravním zpožděním spadá do kategorie čistě matematických modelů a slouží jako předešlý model k referenčnímu nastavení optimalizace. Samotný systém je shodný se systémem čtvrtého řádu (vz: 4.3) respektive (vz: 4.4) a doplněným o nelineární prvek dopravního zpoždění v hodnotě $T_D = 1$ (sek). Spolu s tímto prvkem se celá soustava stává nelineární a proto je vhodná k testovacím účelům navržené fitness funkce (sekce: 4.1.1 - viz str. 46). Kvůli analytickému řešení regulátoru je potřeba celý systém linearizovat pomocí Padého rozvoje (vz: 1.8) druhého řádu na dopravní zpoždění. Přenos linearizovaného dopravního zpoždění (vz: 4.5) a graf porovnání linearizace (graf: 4.8b). Přenos celkového linearizovaného systému (vz: 4.6). Systém je sám osobě nestabilní ve zpětné vazbě, a proto nelze použít základní nastavení pomocí (ZN). Kvůli tomuto omezení se standartní regulátor stanovil dle metody (odečtu hodnot z grafu) Root-Locus pro nastavení vhodného zesílení regulátoru, a tedy nastavení regulátoru typu P. Přechodové charakteristiky (graf: 4.9a), (tab: 4.6) a ohodnocení skokové odezvy (obr: 4.1) na systém s regulátorem spolu s ohodnocením fitness funkce dle (sekce: 4.1.1 - viz str. 46) $A, B, C = 1$ (tab: 4.5).

$$G(s) = \frac{s^2 - 6s + 12}{s^2 - 6s + 12} \tag{4.5}$$

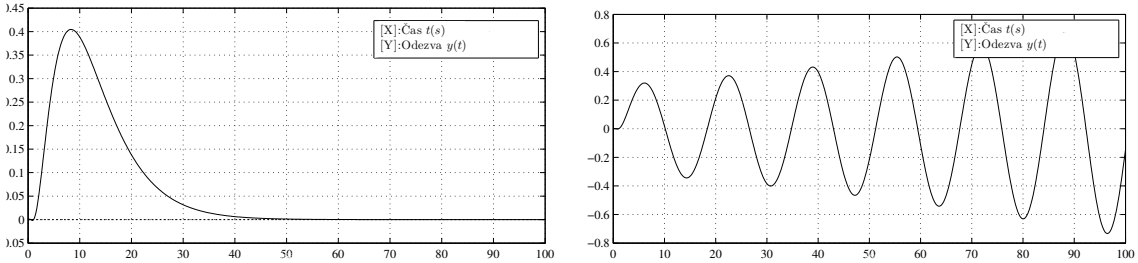
$$G(s) = \frac{6s^2 - 36s + 72}{48s^5 + 332s^4 + 852s^3 + 601s^2 + 150s + 12} \tag{4.6}$$

Charakteristiky obecné přechodové funkce čtvrtého řádu s dopravním zpožděním $T_D = 1$ (sek) jsou zobrazeny na následujících grafech, konkrétně odezva systému na skokový signál (obr: 4.3) (graf: 4.5a), odezva systému na skokový signál ve zpětné vazbě (obr: 4.3) (graf: 4.5b), odezva systému na impulzní signál (obr: 4.3) (graf: 4.6a), odezva systému na impulzní signál ve zpětné vazbě (obr: 4.3)

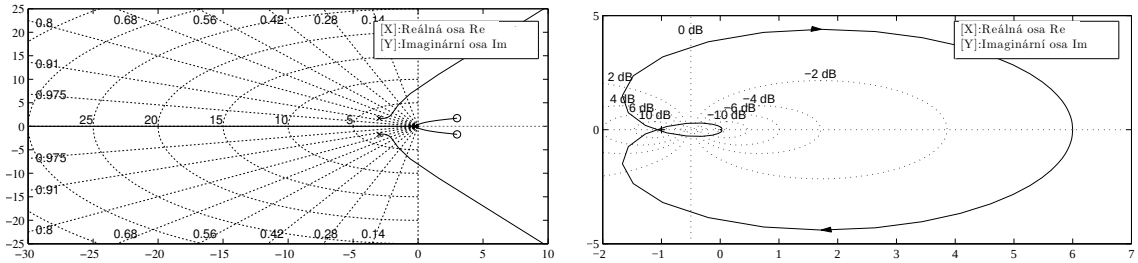
(graf: 4.6b), Root-Locus charakteristika (graf: 4.7a) a Nyquistova frekvenční charakteristika (graf: 4.7b), které znázorňují chovaní linearizovaného matematického modelu spolu s porovnáním skutečného dopravního zpoždění s linearizací pomocí Pádeho linearizace druhého řádu (graf: 4.8b).



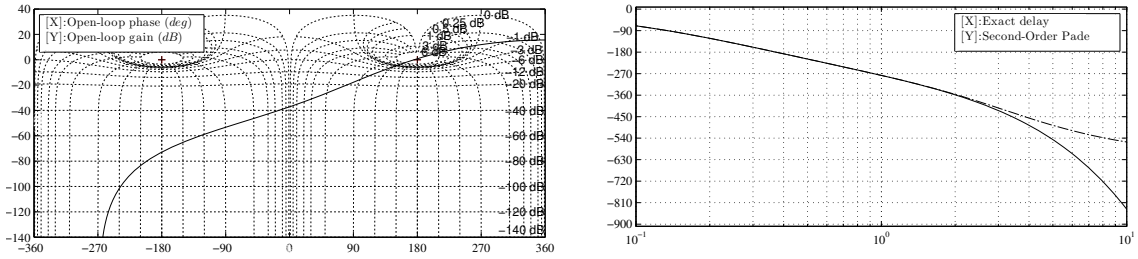

**Graf 4.5:** Pořadí grafů: (a: Odezva systému na jednotkový skokový signál pro System4Delay.) a (b: Odezva systému na jednotkový skokový signál pro System4Delay ve zpětné vazbě.).




**Graf 4.6:** Pořadí grafů: (a: Odezva systému na jednotkový impulzní signál pro System4Delay.) a (b: Odezva systému na jednotkový impulzní signál pro System4Delay ve zpětné vazbě.).




**Graf 4.7:** Pořadí grafů: (a: Root-Locus charakteristika pro System4Delay.) a (b: Nyquistova frekvenční charakteristika pro System4Delay.).



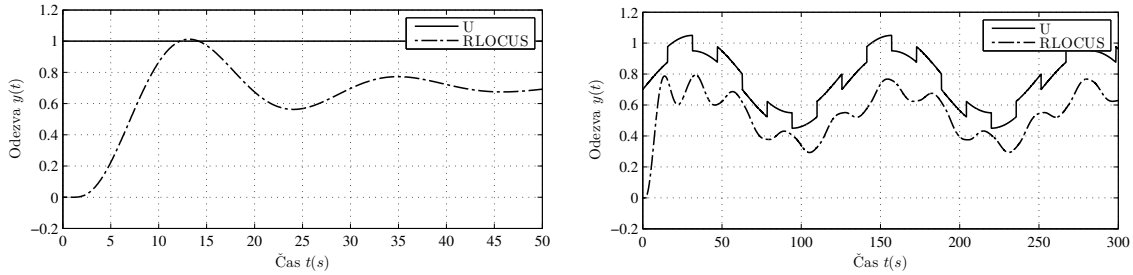

**Graf 4.8:** Pořadí grafů: (a: Nicholsova frekvenční charakteristika pro System4Delay.) a (b: Fázová charakteristika lin. dopravního zpoždění dle (2. řádu padého lin.) pro System4Delay.).

| ⊞ | $T_r$ | $T_s$ | $Min_s$ | $Max_s$ | Over | Under | Peak | $T_p$ |
|---|---|---|---|---|---|---|---|---|
| Výs. [RLOCUS] | 6.578 | NaN | 0.563 | 1.013 | 1.269 | 0.013 | 1.013 | 13.178 |

**Tabulka 4.5:** Tabulka ohodnocení odezvy na skokový signál dle nejlepších výsledků [Regulátor Popis-systemu Alg GENERAL] pro System4Delay.

| ⊞ | P | Fit. | ITAE sin/pwm |
|---|---|---|---|
| Výs. [RLOCUS] | 0.400 | 5.270 | 8870.571 |

**Tabulka 4.6:** Tabulka nastavení regulátoru [Regulátor Popis-systemu Alg GENERAL] pro System4Delay.

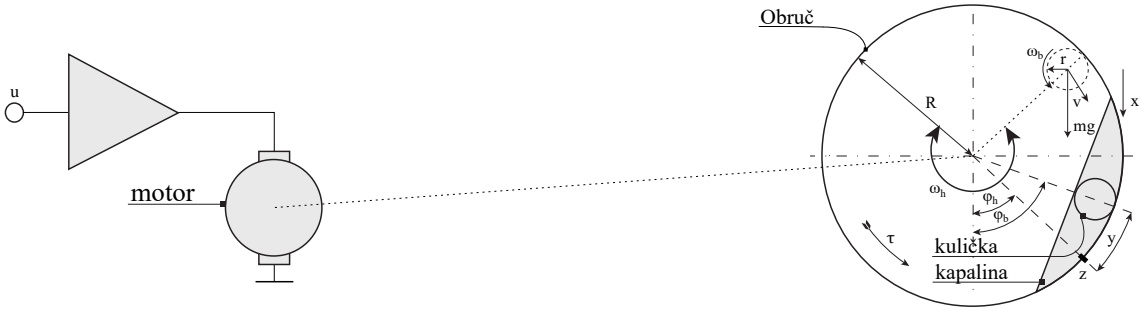



**Graf 4.9:** Pořadí grafů: (a: Průběh přechodové charakteristiky dle obecného nastavení pro System4Delay.) a (b: Průběh přechodové charakteristiky na SIN/PWM signálu obecného nastavení pro System4Delay.).

### 4.2.3 Model kuličky v obruči

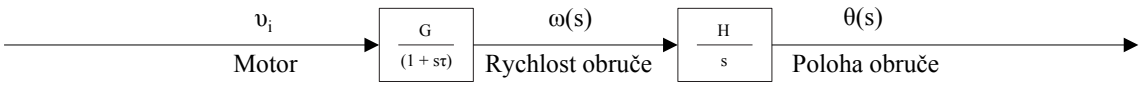


Systém kuličky v obruči slouží k demonstraci chovaní sypkých, popřípadě kapalných látek s volným povrchem v kontejneru a při působení vnější síly. Princip pohybu kuličky v obruči je znázorněný na obrázku (obr: 4.4). Řídicím členem systému je stejnosměrný motor, při vstupním signálu $u$ a výstupem ze systému je aktuální poloha referenční značky $z$ na obruči $\varphi_h$ vůči vertikální ose, signál snímače uhlové rychlosti otáčení obruče $\omega_h$, poloha kuličky $\varphi_b$ k vzhledem k vertikální ose a uhlová rychlost kuličky $\omega_b$. Matematicky popis soustavy je přechodový systém (vz: 4.12) čtvrtého řádu, matematicky odvozený v práci *Griffin Ian*: *On-line PID Controller Tuning using Genetic Algorithms*[54] a je založený na reálné soustavě při použití stejnosměrného motoru.




**Obrázek 4.4:** Principiální schéma pro BallandLoop.

Identifikace modelu je založena na hlavním akčním členu soustavy, a to stejnosměrném motoru. Výsledná identifikovaná soustava tedy má shodné vlastnosti s použitým motorem, a to konkrétně časové konstanty a zesílení. Obecný systém pro kuličku v obruči je znázorněný na obrázku (obr: 4.5), *Griffin Ian*: *On-line PID Controller Tuning using Genetic Algorithms*[54].




**Obrázek 4.5:** Základní schéma přechodové funkce pro BallandLoop.

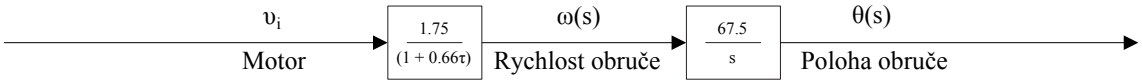
Konstanta zesílení v ustáleném stavu je vypočtena z grafu změny rychlosti otáčení obruče a výsledné zesílení je tedy odvozeno ze změny rychlosti (vz: 4.7) a po měření je vypočtena finální hodnota (vz: 4.8). Časová konstanta systému je naměřena na motoru při $3000RPM$ otáčkách a při napětí $4V$. Časová konstanta je rovna $0.63\Delta\omega$ a je rovna rychlostní změně otáček (vz: 4.9). Zesílení pohybu obruče je vypočteno z grafu ustáleného stavu k otáčkám obruče a výsledné zesílení je rovno rovnici (vz: 4.10). Při dosazeni všech konstant a manuálních úpravách dostaneme výslednou přechodovou funkci ve tvaru (obr: 4.6), *Griffin Ian*: *On-line PID Controller Tuning using Genetic Algorithms*[54].

$$\Delta\omega = g \cdot \Delta v_i \tag{4.7}$$

$$\Delta v_i = 2V => G = \frac{2.21 - naměřeno}{2} = 1.105 \tag{4.8}$$

$$\tau = 0.66(sek) - naměřeno \tag{4.9}$$

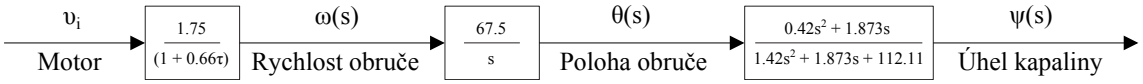
$$H = \frac{135}{2} = 67.5 - odečteno\ z\ grafu \tag{4.10}$$




**Obrázek 4.6:** Obecná přechodová funkce pro BallandLoop.

Pro výslednou rovnici systému je třeba započítat pohyb kuličky v obruči, a to při započítaní tření a tlumeného kmitaní po úpravách dostaneme výslednou přechodovou funkci ve tvaru (vz: 4.11) a výsledný sytém je znázorněný na obrázku (obr: 4.7). Po konečných úpravách je tedy systém kuličky v obruči popsán přechodovou funkcí čtvrtého řádu (vz: 4.12).

$$G(s) = \frac{0.42s^2 + 1.873s}{1.42s^2 + 1.873s + 112.11} \tag{4.11}$$

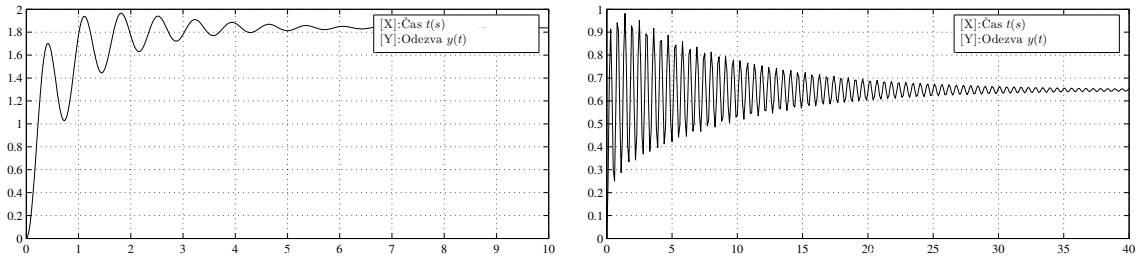



**Obrázek 4.7:** Finální přechodová funkce pro BallandLoop.

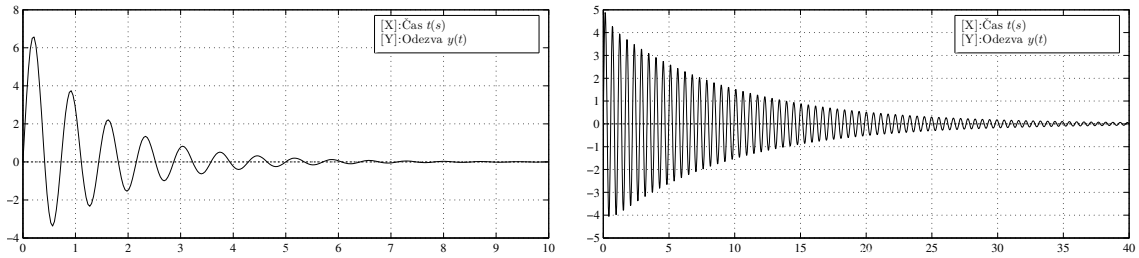
$$G(s) = \frac{46.21s^2 + 206.1s}{0.9372s^4 + 2.656s^3 + 75.87s^2 + 112.1s} \tag{4.12}$$

Charakteristiky přechodové funkce čtvrtého řádu pro systém kuličky v obruči jsou zobrazeny na následujících grafech, konkrétně odezva systému na skokový signál (obr: 4.3) (graf: 4.10a), odezva systému na skokový signál ve zpětné vazbě (obr: 4.3) (graf: 4.10b), odezva systému na impulzní signál (obr: 4.3) (graf: 4.11a), odezva systému na impulzní signál ve zpětné vazbě (obr: 4.3) (graf: 4.11b), Root-Locus

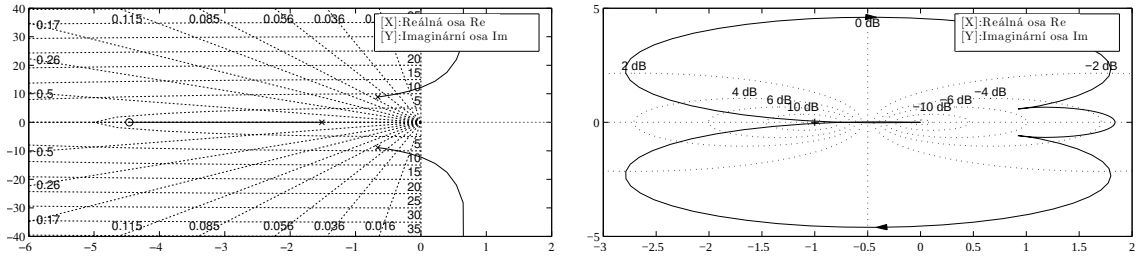
charakteristika (graf: 4.12a) a Nyquistova frekvenční charakteristika (graf: 4.12b), které znázorňují chovaní matematického modelu spolu se základním nastavením PID regulátoru (sekce: 1.2.1 - viz str. 9) dle Zeigler-Nichols (sekce: 1.2.2 - viz str. 10) (graf: 4.13a), (tab: 4.8). Ohodnocení skokové odezvy (obr: 4.1) na systém s regulátorem spolu s ohodnocením fitness funkce pomocí (sekce: 4.1.1 - viz str. 46) $A = 1$, $B = 1$, $C = 1$ (tab: 4.7).




**Graf 4.10:** Pořadí grafů: (a: Odezva systému na jednotkový skokový signál pro BallAndLoop.) a (b: Odezva systému na jednotkový skokový signál pro BallAndLoop ve zpětné vazbě.).




**Graf 4.11:** Pořadí grafů: (a: Odezva systému na jednotkový impulzní signál pro BallAndLoop.) a (b: Odezva systému na jednotkový impulzní signál pro BallAndLoop ve zpětné vazbě.).



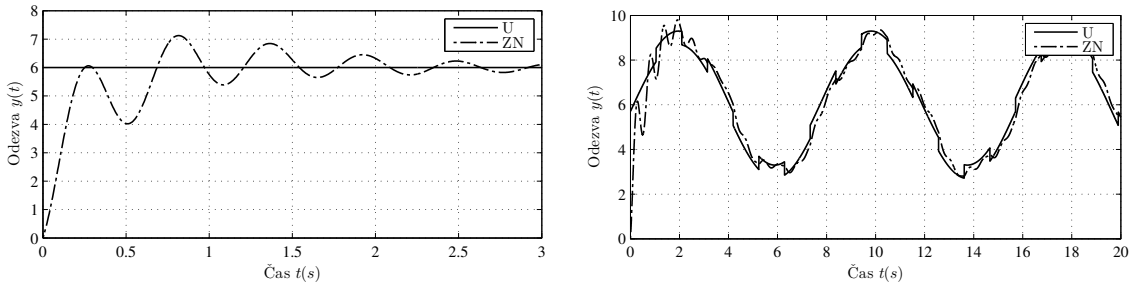

**Graf 4.12:** Pořadí grafů: (a: Root-Locus charakteristika pro BallAndLoop.) a (b: Nyquistova frekvenční charakteristika pro BallAndLoop.).

| ⊞ | $T_r$ | $T_s$ | $Min_s$ | $Max_s$ | Over | Under | Peak | $T_p$ |
|---|---|---|---|---|---|---|---|---|
| Výs. [ZN] | 0.172 | 2.842 | 4.019 | 7.125 | 18.742 | 0 | 7.125 | 0.815 |

**Tabulka 4.7:** Tabulka ohodnocení odezvy na skokový signál dle nejlepších výsledků [Regulátor Popis-systemu Alg GENERAL] pro BallAndLoop.

| ⊞ | P | I | D | Fit. | ITAE sin/pwm |
|---|---|---|---|---|---|
| Výs. [ZN] | 0.948 | 0.250 | 0.063 | 4.851 | 537.800 |

**Tabulka 4.8:** Tabulka nastavení regulátoru [Regulátor Popis-systemu Alg GENERAL] pro BallAndLoop.



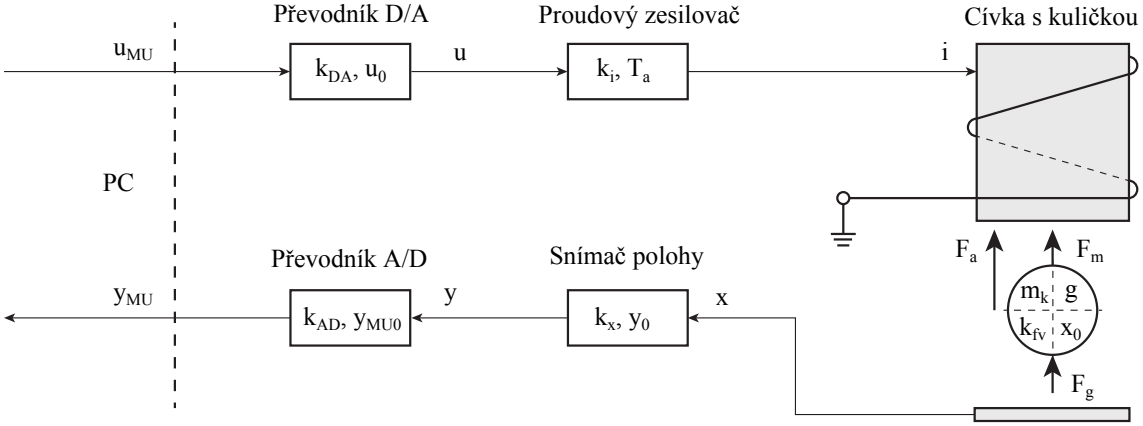

**Graf 4.13:** Pořadí grafů: (a: Průběh přechodové charakteristiky dle obecného nastavení pro BallAndLoop.) a (b: Průběh přechodové charakteristiky na SIN/PWM signálu obecného nastavení pro BallAndLoop.).

### 4.2.4 Model magnetické levitace

Model magnetické levitace CE 152 od firmy Humusoft je sériově vyráběný reálný model. Jedná se o jednorozměrný nestabilní nelineární systém, na kterém je možno snadno demonstrovat návrh regulátorů a řízení v reálných soustavách. Model se skládá ze solenoidní cívky s jádrem, ocelové kuličky a indukčním snímačem polohy. Při průchodu proudu cívkou vzniká magnetické pole, které působí na kuličku. Kulička vytváří magnetické pole opačné orientace, a tedy je přitahována k jádru cívky.

4

Princip levitace je rovnováha sil působící na kuličku. Řízení modelu je přes stolní PC a pomocí programu Matlab / Simulink s toolboxem Real Time Toolbox. Model je připojen k PC pomocí A/D a D/A převodníků. Schématické zapojení je na obrázku (obr: 4.8) a vytvořený model k výpočtu v programu Simulink je dle schématu (model: 4.1). Reálný model soustavy magnetické levitace obsahuje 5 částí které jsou D/A převodník, proudový zesilovač, cívka s kuličkou, snímač polohy a A/D převodník.




**Obrázek 4.8:** Schématické znázornění modelu magnetické levitace CE 152.

#### D/A A/D převodník

D/A převodní zajišťuje komunikaci s modelem a PC, kde převádí digitální signál na analogový. Dynamika převodníku je oproti celkovému systému zanedbatelná, proto jeho charakteristiku považujeme za lineární. A/D převodník převádí spojitý signál na digitální s charakteristikou (vz: 4.14).

$$u = k_{DA}u_{MU} + u_0 \tag{4.13}$$

$$y_{MU} = k_{AD}y + y_{MU0} \tag{4.14}$$

### Cívka s kuličkou

Cívka vytváří magnetické pole působící na kuličku. Princip magnetické levitace na rovnováze sil působící na objekt. Jedná se o gravitační sílu $F_g$ a magnetickou sílu $F_m$. Magnetická síla je funkcí dvou proměnných, polohy kuličky a proudu. Matematická formulace pohybové rovnice:

$$F_a = F_m - F_g \tag{4.15}$$

$$F_m = \frac{i^2 k_c}{(x - x_0)^2} \tag{4.16}$$

$$F_g = m_k g \tag{4.17}$$

$$F_a = m_k \ddot{x} - k_{fv}\dot{x} \tag{4.18}$$

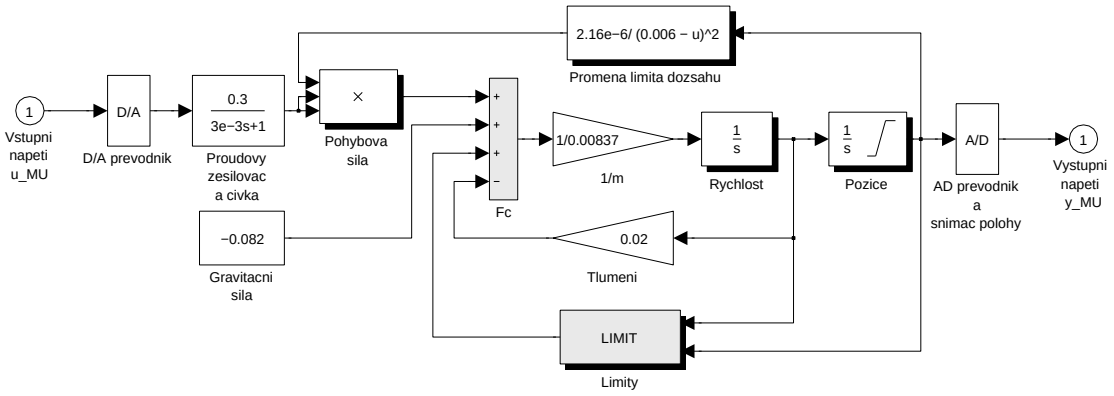
Chování kuličky v magnetickém poli popisuje diferenciální rovnice (vz: 4.19) druhého řádu, kde proud $i$ je vstupní parametr a poloha kuličky $x$ je výstupní veličina.

$$m_k \ddot{x} - k_{fv}\dot{x} = \frac{i^2 k_c}{(x - x_0)^2} - m_k g \tag{4.19}$$

### Snímač polohy

K měření polohy se využívá indukční snímač s lineární charakteristikou (vz: 4.20).

$$y = k_x x + y_0 \tag{4.20}$$




**Model 4.1:** Model magnetické levitace pro Simulink.

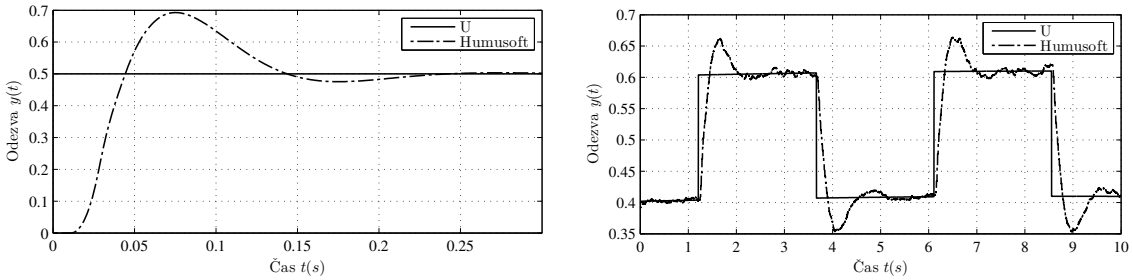
Charakteristiky modelu magnetické levitace jsou zobrazeny na následujících grafech se základním nastavením PID regulátoru (sekce: 1.2.1 - viz str. 9) (graf: 4.14a), (tab: 4.10) a ohodnocení skokové odezvy (obr: 4.1) na systém s regulátorem spolu s ohodnocením fitness funkce dle (sekce: 4.1.1 - viz str. 46) $A = 1$, $B = 1$, $C = 1$ (tab: 4.9). Základní nastavení regulátoru je otestované na reálné soustavě pro PWM vstupní signál (graf: 4.14b).

| ⊞ | $T_r$ | $T_s$ | $Min_s$ | $Max_s$ | Over | Under | Peak | $T_p$ |
|---|---|---|---|---|---|---|---|---|
| Výs. [Default] | 0.021 | 0.215 | 0.451 | 0.693 | 38.549 | 0 | 0.693 | 0.075 |

**Tabulka 4.9:** Tabulka ohodnocení odezvy na skokový signál dle nejlepších výsledků [Regulátor Popis-systemu Alg GENERAL] pro Maglev.

| ⊞ | P | I | D | Fit. | ITAE sin/pwm |
|---|---|---|---|---|---|
| Výs. [Default] | 4.431 | 0.028 | 0.027 | 2.770 | 1.467 |

**Tabulka 4.10:** Tabulka nastavení regulátoru [Regulátor Popis-systemu Alg GENERAL] pro Maglev.




**Graf 4.14:** Pořadí grafů: (a: Průběh přechodové charakteristiky dle obecného nastavení pro Maglev.) a (b: Průběh přechodové charakteristiky na reálné soustavě pro Maglev.).

4

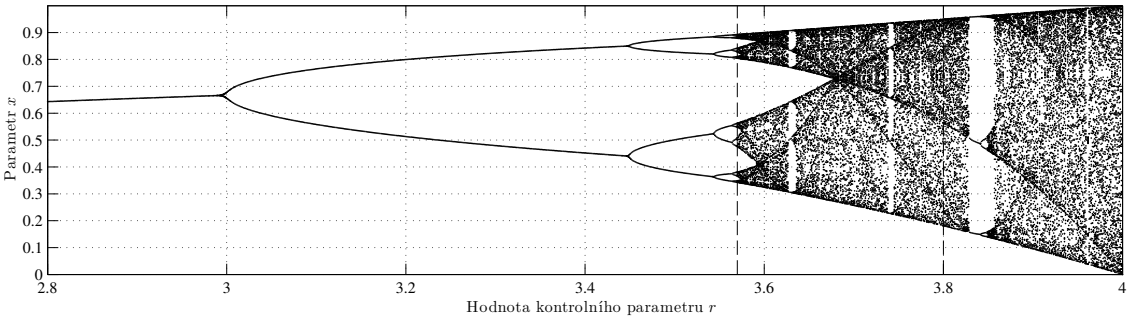
### 4.2.5 Logistická mapa

Logistická mapa je nejznámější a nejvíce využívaný příklad z problematiky stabilizace deterministického chaosu a příkladů použití chaosových systémů. Existuje mnoho druhů a postupů při stabilizaci systémů vykazující chaotické chování, jeden z příkladů je i prezentovaný v práci *Šenkeřík Roman*: *Optimal Control of Deterministic Chaos*[51] a je i základem pro obecný popis logistické mapy v této kapitole a dále i při příkladu stabilizace. Logistická mapa je názorný příklad, jak jednorozměrná diskrétní rovnice může přejít v určitých částech na chaotický systém. Tato rovnice byla představena a zpopularizována Robertem Mayem *May M. Robert*: *Simple mathematical models with very complicated dynamics*[55] v roce 1976 a je hojně využívaná v biologii jako příklad simulace vztahů modelové živočišné populace, kde $x_n$ je v rozsahu $< 0, 1 >$ a parametr reprodukce $r$ v rozsahu $< 0, 4 >$. Matematický popis rovnice je (vz: 4.21).

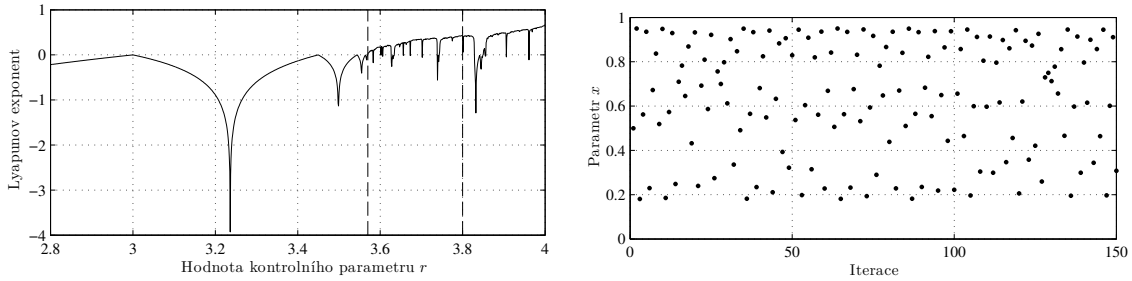
$$x_{n+1} = rx_n(1 - x_n) \tag{4.21}$$

Jednotlivé prvky rovnice mají z biologického hlediska vlastní význam, $x_n$ je velikost populace v roce $n$ a $r$ je kladné číslo udávající koeficient reprodukce a hladomoru v populaci. Velikost chaotického chování ovlivňuje hodnota koeficientu $r$, kde $r$ v hodnotě 3.57 je počátek chaotického chování a konec periodické oscilaci stavů, $r$ větší než 3.57 se systém dostává do pravého chaotického stavu. Průběh systému dle parametru $r$ je znázorněný na bifurkační diagram logistické mapy (graf: 4.15). Kvalitu chaosového systému také znázorňuje Lyapunov exponent (graf: 4.16a), který udává při kladné hodnotě chaotické chování logistické mapy. Pro regulaci chaosu je

zvolený výchozí koeficient $r$ v hodnotě 3.8, kde systém už je v pokročilé chaotické fázi. Neregulovaný systém ve zvolené hodnotě je znázorněný na grafu (graf: 4.16b).

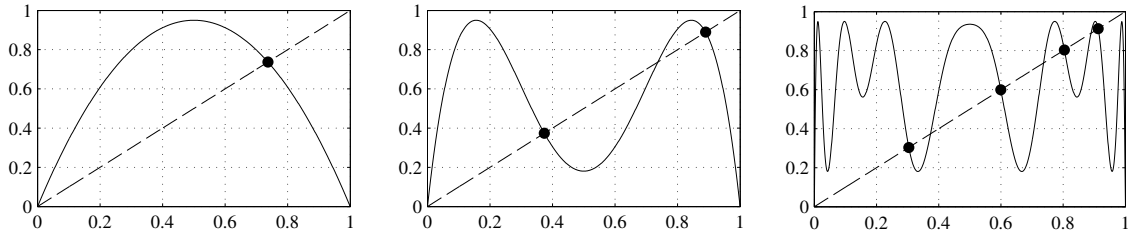



**Graf 4.15:** Bifurkační diagram logistické mapy.




**Graf 4.16:** Pořadí grafů: (a: Lyapunov exponent pro logistickou mapu.) a (b: Neřízená logistická mapa v parametru r (3.8).).





**Graf 4.17:** Grafický výpočet fixních bodů pro logistickou mapu v řádu 1, 2 a 4. Pořadí grafů: (a: LM 1. řádu v parametru r (3.8), $\lambda$ 45.), (b: LM 3. řádu v parametru r (3.8), $\lambda$ 45.) a (c: LM 4. řádu v parametru r (3.8), $\lambda$ 45.).

Stabilizace logistické mapy, se provádí na fixní body neboli hodnoty (UPO) a jsou to hodnoty při kterých logistická mapa dokáže nabývat stabilních hodnot nebo periodicky kmitat mezi těmito body. Výpočet těchto bodů se provádí buď analyticky nebo graficky, jak je prezentováno na grafech pro různé hodnoty fixních bodů, pro jeden fixní bod v hodnotě $r$ (graf: 4.17a), pro dva fixní body (graf: 4.17b) a pro čtyři fixní body (graf: 4.17c). Výsledné fixní hodnoty jsou pak v tabulce (tab: 4.11) a jsou dále použity při příkladu stabilizace logistické mapy.

| ⊞ | Fixní bod 1. | Fixní bod 2. | Fixní bod 3. | Fixní bod 4. |
|---|---|---|---|---|
| XF 1. (graf: 4.17a) | 0.737 | - | | |
| XF 2. (graf: 4.17b) | 0.374 | 0.889 | - | |
| XF 4. (graf: 4.17c) | 0.304 | 0.600 | 0.804 | 0.912 |

**Tabulka 4.11:** Hodnoty fixních bodů pro logistickou mapu v řádu 1, 2 a 4.

### 4.2.6 Duffingova rovnice

Duffingova rovnice (nebo také Duffingův oscilátor) byla představena v roce 1918 Georgem Duffingem a je to nelineární diferenciální rovnice druhého řádu, která se využívá k modelování určitých typů buzených oscilátorů. Jeden z nejznámějších příkladů je oscilátor typu UEDA, který se skládá z buzení pomocí elektromagnetického pole tvořeného dvěma elektromagnety a tento typ oscilátoru je velice vhodný na prezentaci příkladů chaotických systémů. Duffingův oscilátor je definovan rovnicí (vz: 4.22). Tento typ oscilátoru představuje dynamický systém, který může být dále využíván na modelování fyzických vlastností biologických systémů *Chapman Stephen J.*: *MATLAB Programming for Engineers*[83].

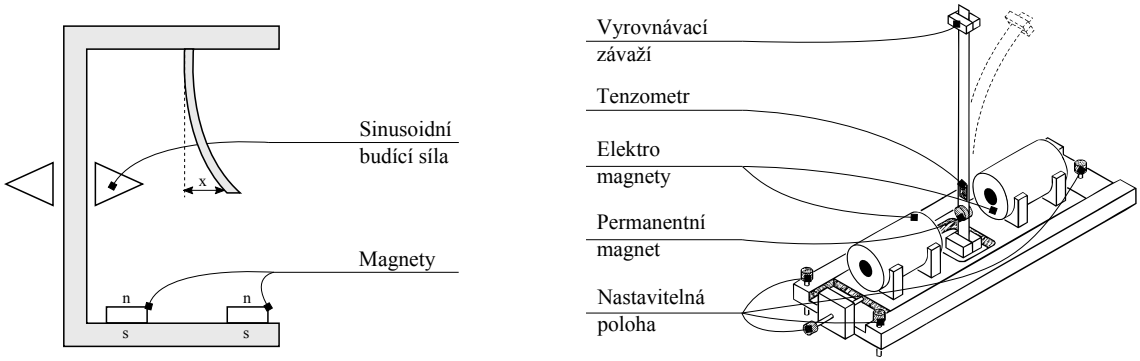
$$\ddot{x} = f\left(x, \dot{x}, t\right) \tag{4.22}$$

Funkce $f$ závisí na hodnotě času, a neobsahuje podmínku o jednoznačnosti řešení, kdy je možnost protnutí trajektorií ve fázové rovině. Jako příklad tohoto chování je duffingova rovnice (vz: 4.23). Rovnice popisuje fyzický model, kde $k$ je koeficient tlumení, $\Gamma$ je budící amplituda a $\omega$ je frekvence budící síly.



$$\dot{x} = y, \dot{y} = x - ky - x^3 + \Gamma cos\left(\omega t\right) \tag{4.23}$$

Duffingova rovnice lze použít na vyjádření konstantně buzeného kyvadla, kde $x(t)$ reprezentuje rozdíl polohy kyvadla, $\dot{x}$ je rychlost tělesa a $\Gamma cos\left(\omega t\right)$ je sinusoidní budící sila. Obrázek (obr: 4.9) zobrazuje přibližné schéma jednoduchého kyvadla a jeho možný reálný model pomocí budících elektromagnetů *Berger J E.*, *Nunes G.*: *A mechanical Duffing oscillator for the undergraduate laboratory*[84].




**Obrázek 4.9:** Schéma Duffingova oscilátoru a příklad realizace oscilátoru v praxi.

Duffingova rovnice je hojně studovaný problém, jak v rovině stability, harmonického řešení, chaosu atd. Jakýmakoliv neautonomní buzená diferenciální rovnice muže být reprezentována jako autonomní, při zavedení třetí proměnné $\theta = \omega t$. Při aplikaci na původní rovnici (vz: 4.23) a periodou $\frac{2\pi}{\omega}$ dostáváme rovnici dynamického systému (vz: 4.24).

$$\dot{x} = y, \dot{y} = x - ky - x^3 \tag{4.24}$$

Duffingova mapa (vz: 4.25) je diskrétní verze dynamického systému (vz: 4.24) a používá se jako příklad soustavy projevující chaotické chování. Konstanty $a$ a $b$ jsou ve většině případů nastaveny pro hodnoty $a = 2.75$ a $b = 0.2$. Grafické znázornění

duffingovy mapy je v grafu (graf: 4.19b), průběh hodnoty $x$ v závislosti na hodnotě iterace (graf: 4.20b) a její rozložení na ose $y$ pro 600 iterací (graf: 4.20a).

$$x_{n+1} = y_n, y_{n+1} = -bx_n + ay_n - y^3 \tag{4.25}$$

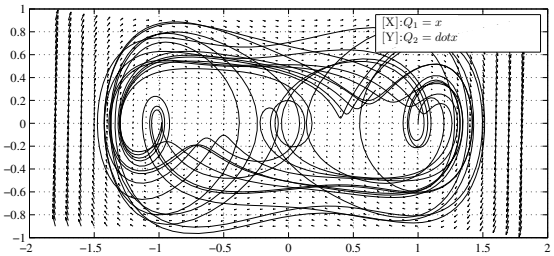
Při výpočtu kritických hodnot systému, kdy $\dot{x} = \dot{y} = 0$, dostáváme $M = (-1, 0)$, $N = (1, 0)$ a $O = (0, 0)$, které určují stabilitu v kritických bodech pomocí Jakubiánovy rovnice (vz: 4.26).

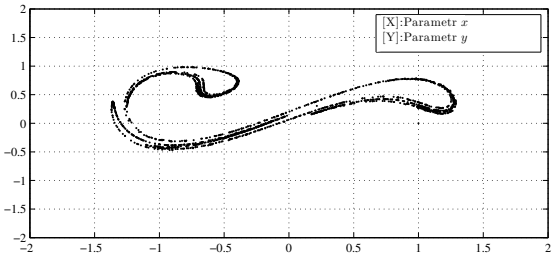
$$J = \begin{pmatrix} \frac{\partial P}{\partial x} & \frac{\partial P}{\partial y} \\ \frac{\partial Q}{\partial x} & \frac{\partial Q}{\partial y} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 - x^2 & -k \end{pmatrix} \tag{4.26}$$

K vyjádření chaotického chování duffingovy rovnice můžeme použít fázových grafů a Poincardova grafu při hodnotě T = 0.5 (graf: 4.18a) a (graf: 4.18b). Je třeba poznamenat, že rozdílné chaotické chování a chaotické atraktory lze vyjádřit na základě různých hodnot duffingovy rovnice.

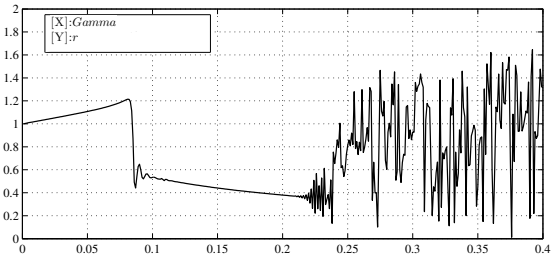
Bifurkační diagram pro horní náběh amplitudy (graf: 4.19a) pro funkci (vz: 4.23) v rozmezí hodnot $0 < \Gamma < 0.4$ a blízko kritického bodu $N$ znázorňuje hodnoty chaotického chování systému na základě hodnoty $\Gamma$. Hodnota $0 \leq \Gamma < 0.25$ udává poměrné stabilní systém, hodnota vyšší než 0.25 posune systém do chaotické roviny.

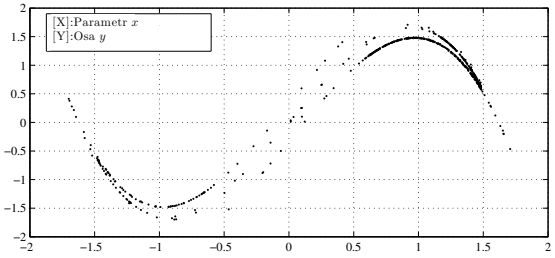




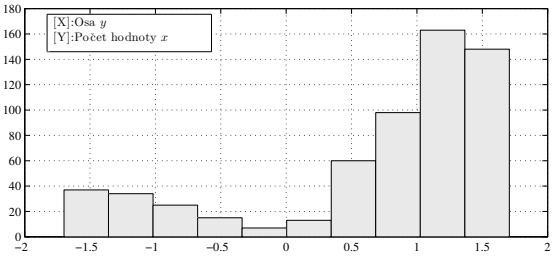



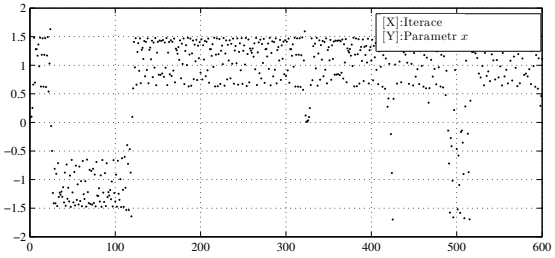

**Graf 4.18:** Pořadí grafů: (a: Směrové pole a fázová rovina duffingovy rovnice.) a (b: Poincare průběh duffingovy rovnice.).




**Graf 4.19:** Pořadí grafů: (a: Bifurkační diagram duffingovy rovnice.) a (b: Průběh duffingovy mapy.).




**Graf 4.20:** Pořadí grafů: (a: Rozložení hodnoty $x$ dle osy $y$ v duffingově mapě.) a (b: Duffingova mapa v bodech $x$.).

### 4.2.7 Yagi-Uda anténa

V dnešní době bezdrátových technologií je stále více kladen důraz na kvalitní návrh antén. Dobrý návrh antén však vyžaduje znalosti, inteligenci a zkušenosti. Tyto požadavky by mohly plně zastoupit automatické konstrukční systémy, které však chybí. Jako vhodný nástroj pro návrh a optimalizaci antén se nabízejí evoluční algoritmy, které jsou schopny efektivně prohledat neznámé konstrukční prostory a poměrně zefektivnit návrh antén. Jako referenční typ antény byla vybrána anténa typu Yagi-Uda, jenž představuje dostatečně složitou konstrukci s vhodnými optimalizačními problémy.

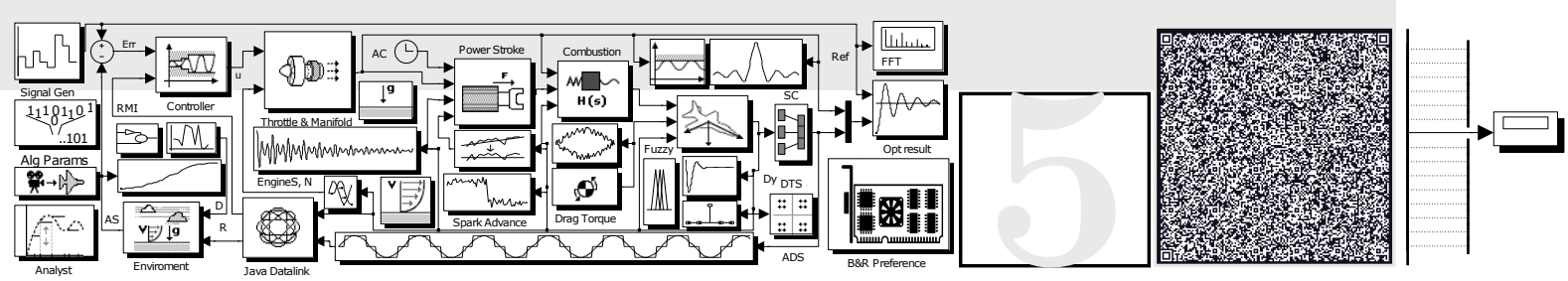

**Obrázek 4.10:** Schéma Yagi-Uda antény.



Yagi-Uda je populární směrová lineárně polarizační anténa s průměrnou ziskovostí sestávající z jednoho napájeného prvku (obvykle tvořeného složeným půl vlnným dipólem) a několika parazitních prvků (obvykle jeden reflektor a dva až několik direktorů). Praktické využiti antény je HF (High Frequency), VHF (Very High Frequency) a UHF (Ultra High Frequency) pásmu. Uda anténa byla vyvinuta v Japonsku v roce 1926 profesorem Shintaro Uda v Tohoku Imperial University a dále rozvíjena profesorem Hidetsugu Yagi, od té doby se tento typ antény jmenuje Yagi - Uda anténa.

Typická konstrukce Yagi-Uda antény se skládá z napájeného elementu (driven element), který velikostně odpovídá pod hranicí $\lambda/2$, typicky 0.45-0.49$\lambda$. Parazitující elementy ve směru vyzařování (directors) jsou konstruovány s menší velikostí než napájecí element, přibližně 0.4 - 0.45$\lambda$. Poslední typ elementu antény je nazván (reflector) a jeho velikost přesahuje napájený element. Velikost mezery se volí ne menší než 0.3$\lambda$. Konstrukci antény zobrazuje obrázek (obr: 4.10). Velikost elementů musí být kompenzována pomocí velikosti ramene a pomocí průměru jednotlivých prvků antény. Tento typ antény se může optimalizovat na několik typů požadavků, a to zisk, impedance a šířka pásma. Z hlediska výsledného návrhu antény je vždy kompromis mezi těmito požadavky, a proto je tento typ antény velice vhodný na vícekriteriální optimalizaci a vhodné sestavení hodnotící funkce.

K dosažení optimálního výsledku konstrukce antény používáme různé velikosti individuálních elementů. V minulosti byl návrh antén (velikosti elementu a mezery mezi jednotlivými prvky) závislí na expertním systému navrhování *T.Joines Eric*, *T.Joines William*: *Design of Yagi-Uda Antenna Using Genetic Algorithms*[80], v dnešní době je základní model antény navržený v simulačním programu a posléze optimalizován na základě numerických metod. Samotný výpočet vhodné konstrukce antény je komplexní problém založený na požadavcích antény a expertize řešitele.




# KAPITOLA 5

# NASTAVENÍ PID REGULÁTORU

Prvním příkladem nastavení regulátoru je samostatné nastavení hodnot (PID). Tento typ regulátoru je zvolen jako nejpoužívanější typ s možností odkázání na referenční nastavení, dle předchozí kapitoly. Tato kapitola je také míněna jako referenční bod k nastavení fitness funkce a jejích váhových hodnot (sekce: 4.1.1 - viz str. 46). Pro jednoduchost se berou čtyři základní nastavení, kdy váhové funkce zapínají jednotlivé prvky fitness funkce, tedy hodnota $A$, $B$, $C$ je buď 0 nebo 1 ($true/false$) a poslední možností, kdy jsou aktivovány všechny kombinace fitness funkce. Tedy kombinace nastavení funkce jsou tato, dle kódového označení F100 až F111.



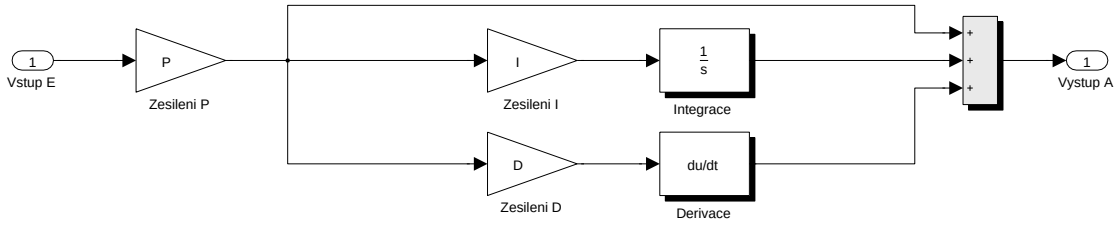
V rámci testování jsou využité všechny kombinace jen pro první model, tedy model $system4$, k porovnání vlivu jednotlivých váhových proměnných. Další modely už využívají jen nejvíce komplexní nastavení fitness funkce $F111$ k porovnaní s referenčním nastavením.

V poslední řadě tato sekce slouží jako porovnaní vhodnosti jednotlivých optimalizačních algoritmů pomocí statistiky na 100 opakovaní pro každý chod optimalizace. U každého algoritmu se porovnává schopnost najít výsledek, v jakém rozptylu a v potřebném počtu iterací viz grafy *Průběh fitness hodnoty pro všechny opakovaní s histogramy iterací a hodnoty fitness hodnoty* a tabulka statistiky *Tabulka statistiky všech opa-*

*kovaní dle iterací a fitness hodnoty.*

Všechny modely budou řízeny pomocí regulátoru (PID) v časové rovině pomocí vzorce (vz: 1.11), kromě prvního modelu *system4*, který využívá mírně upravenou strukturu (vz: 1.11) v integračním zesilovači viz model (model: 5.1). Tabulka *nastavení regulátoru* ukazuje finální nastavení regulátoru pro konkrétní algoritmus a tabulka *ohodnocení odezvy na skokový signál dle nejlepších výsledků* obsahuje hodnoty ohodnocení skokového signálu pro výsledné přechodové funkce dle sekce (sekce: 4.1 - viz str. 45). Výsledné přechodové funkce jsou zobrazeny na grafu *Průběh přechodové charakteristiky* pro každý konkrétní algoritmus.

- Prostá ITAE funkce kde $A = 1$, $B = 0$, $C = 0$ jako **F100**.
- ITAE funkce spolu s funkcí Overshot $A = 1$, $B = 1$, $C = 0$ jako **F110**.
- ITAE funkce spolu s funkcí Wave $A = 1$, $B = 0$, $C = 1$ jako **F101**.
- ITAE funkce spolu s funkcí Overshot a funkcí Wave $A = 1$, $B = 1$, $C = 1$ jako **F111**.




**Model 5.1:** Struktura upraveného PID regulátoru.

5

## 5.1 Model čtvrtého řádu

Nastavení regulátoru (PID) pro model *system4* pomocí algoritmu (HC12), (DE) a (NM). Následující sekce prezentuje samotné nastavení algoritmu, hodnotící funkce a konečné výsledky s vybraným statistickým porovnáním. Z prezentovaných hodnot a grafů v této sekci je zřejmé, že model čtvrtého řádu, který je podrobně popsaný i s referenčním nastavením regulátoru v sekci (sekce: 4.2.1 - viz str. 47), je velice dobře regulovatelný už s pomocí (PID) regulátoru. Co se týká jednotlivých algoritmů je dobře vidět jak (HC12) i (DE) dokáží poměrně spolehlivě pro každé jednotlivé spuštění najít optimální výsledek. Na rozdíl od (NM), který ve většině případů také najde optimální výsledek, ale nemá takovou procentuální úspěšnost jako předešlé dva algoritmy.

### 5.1.1 Nastavení dle algoritmu HC12

Sekce nastavení (PID) regulátoru pro model *system4* dle algoritmu (HC12). Regulátor je nastaven dle fitness funkce *F100*, *F110*, *F101* a *F111*. Kvalita algoritmu je ukázána na grafech *Průběh fitness hodnoty* pro *F100* (graf: 5.2a), *F110* (graf: 5.3a), *F101* (graf: 5.4a) a *F111* (graf: 5.5a), *Box graf hodnoty fitness* pro *F100* (graf: 5.2b), *F110* (graf: 5.3b), *F101* (graf: 5.4b) a *F111* (graf: 5.5b) a *Histogram hodnoty iterací* pro *F100* (graf: 5.2c), *F110* (graf: 5.3c), *F101* (graf: 5.4c) a *F111* (graf: 5.5c).

Celková statistika algoritmu je v tabulce (tab: 5.1). Finální nastavení regulátoru je v tabulce (tab: 5.3) a ohodnocení přechodové funkce je v tabulce (tab: 5.2). Hodnoty přechodových funkcí jsou zobrazeny na skokové funkci (graf: 5.1a) a grafu SIN/PWM (graf: 5.1b) k zobrazení dynamických vlastností regulátoru.

| ⊞ | max | min | mean | median | std | runs |
|---|---|---|---|---|---|---|
| Výsledky [F100] | 5.498 | 2.101 | 2.307 | 2.199 | 0.499 | 100 |
| Iterace [F100] | 29 | 13 | 21.440 | 22 | 5.078 | |
| Výsledky [F110] | 6.640 | 2.231 | 2.605 | 2.357 | 0.845 | |
| Iterace [F110] | 30 | 13 | 22.390 | 23 | 5.029 | |
| Výsledky [F101] | 7.565 | 2.400 | 2.673 | 2.539 | 0.667 | |
| Iterace [F101] | 30 | 13 | 21.150 | 21 | 4.891 | |
| Výsledky [F111] | 7.205 | 2.698 | 2.895 | 2.834 | 0.458 | |
| Iterace [F111] | 29 | 15 | 21.830 | 22 | 4.599 | |

**Tabulka 5.1:** Tabulka statistiky všech opakovaní dle iterací a fitness hodnoty [Regulátor PID Alg HC12] pro System4.

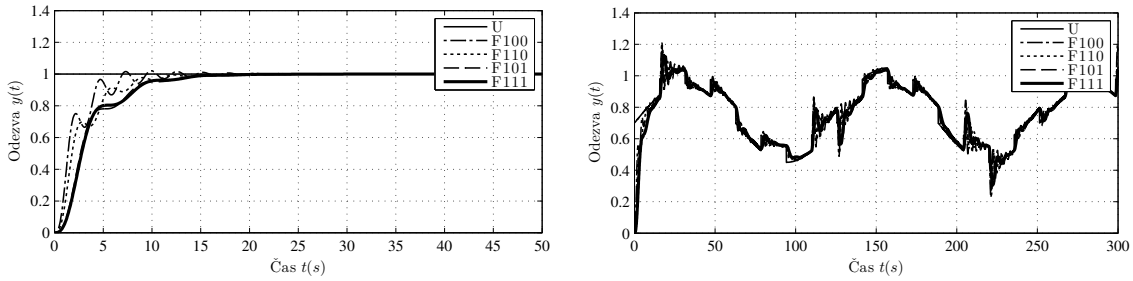
| ⊞ | $T_r$ | $T_s$ | $Min_s$ | $Max_s$ | Over | Under | Peak | $T_p$ |
|---|---|---|---|---|---|---|---|---|
| Výs. [F100] | 3.515 | 11.612 | 0.869 | 1.021 | 2.074 | | 1.021 | 9.950 |
| Výs. [F110] | 4.657 | 11.281 | 0.889 | | | | | 18.600 |
| Výs. [F101] | 6.758 | 13.340 | 0.900 | 1 | 0 | | 1 | 50 |
| Výs. [F111] | 6.988 | 13.796 | 0.901 | | | | | |

**Tabulka 5.2:** Tabulka ohodnocení odezvy na skokový signál dle nejlepších výsledků [Regulátor PID Alg HC12] pro System4.

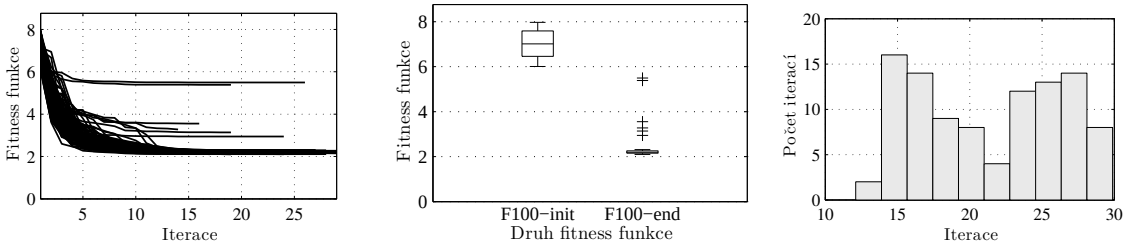


| ⊞ | P | I | D | Fit. | ITAE sin/pwm |
|---|---|---|---|---|---|
| Výs. [F100] | 20.808 | 0.004 | 2.141 | 2.101 | 126.042 |
| Výs. [F110] | 10.959 | 0.005 | 2.762 | 2.231 | 170.354 |
| Výs. [F101] | 3.668 | 0.012 | 3.261 | 2.400 | 250.924 |
| Výs. [F111] | 3.225 | 0.013 | 3.241 | 2.698 | 250.194 |
| P:(tab: 4.1) | B:<0.0,5.0,10> | B:<0.0,0.5,10> | | ⊡ | |
| AI:(tab: 4.2) | M12 [Default], RUNS [100] | | | | |

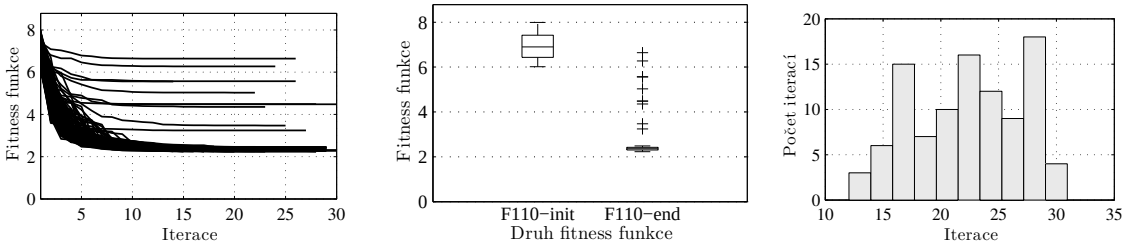
**Tabulka 5.3:** Tabulka nastavení regulátoru [Regulátor PID Alg HC12] pro System4.



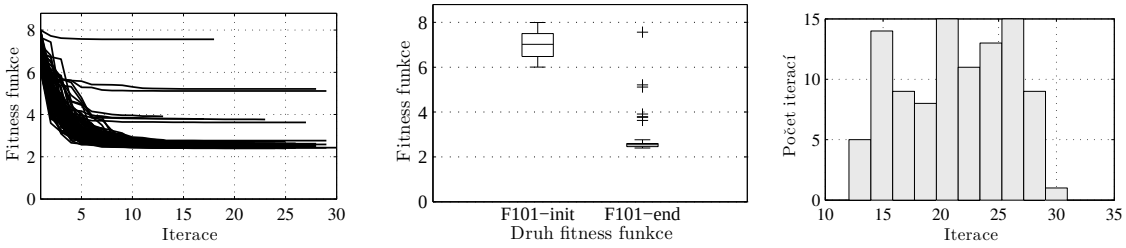

**Graf 5.1:** Pořadí grafů: (a: Průběh přechodové charakteristiky na skokovém signálu dle [Regulátor PID Alg HC12] pro System4.) a (b: Průběh přechodové charakteristiky na SIN/PWM signálu dle [Regulátor PID Alg HC12] pro System4.).



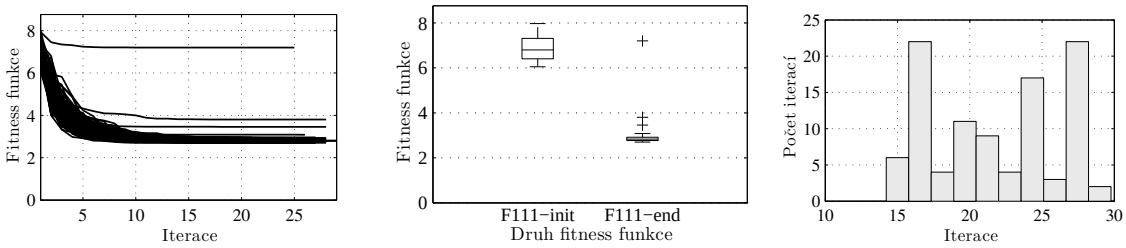

**Graf 5.2:** Průběh fitness hodnoty pro všechny opakovaní s histogramy iterací a ohodnocení fitness funkce [Regulátor PID Alg HC12 Typ F100] pro System4. Pořadí grafů: (a: Průběh fitness hodnoty.), (b: Box graf hodnoty fit. hod.) a (c: Histogram hodnoty iterací.).




**Graf 5.3:** Průběh fitness hodnoty pro všechny opakovaní s histogramy iterací a ohodnocení fitness funkce [Regulátor PID Alg HC12 Typ F110] pro System4. Pořadí grafů: (a: Průběh fitness hodnoty.), (b: Box graf hodnoty fit. hod.) a (c: Histogram hodnoty iterací.).




**Graf 5.4:** Průběh fitness hodnoty pro všechny opakovaní s histogramy iterací a ohodnocení fitness funkce [Regulátor PID Alg HC12 Typ F101] pro System4. Pořadí grafů: (a: Průběh fitness hodnoty.), (b: Box graf hodnoty fit. hod.) a (c: Histogram hodnoty iterací.).




**Graf 5.5:** Průběh fitness hodnoty pro všechny opakovaní s histogramy iterací a ohodnocení fitness funkce [Regulátor PID Alg HC12 Typ F111] pro System4. Pořadí grafů: (a: Průběh fitness hodnoty.), (b: Box graf hodnoty fit. hod.) a (c: Histogram hodnoty iterací.).

### 5.1.2 Nastavení dle algoritmu DE

Sekce nastavení (PID) regulátoru pro model *system4* dle algoritmu (DE). Regulátor je nastaven dle fitness funkce *F100*, *F110*, *F101* a *F111*. Kvalita algoritmu je ukázána na grafech *Průběh fitness hodnoty* pro *F100* (graf: 5.7a), *F110* (graf: 5.8a), *F101* (graf: 5.9a) a *F111* (graf: 5.10a), *Box graf hodnoty fitness* pro *F100* (graf: 5.7b), *F110* (graf: 5.8b), *F101* (graf: 5.9b) a *F111* (graf: 5.10b) a *Histogram hodnoty iterací* pro *F100* (graf: 5.7c), *F110* (graf: 5.8c), *F101* (graf: 5.9c) a *F111* (graf: 5.10c). Celková statistika algoritmu je v tabulce (tab: 5.4). Finální nastavení regulátoru je v tabulce (tab: 5.6) a ohodnocení přechodové funkce je v tabulce (tab: 5.5). Hodnoty

přechodových funkcí jsou zobrazeny na skokové funkci (graf: 5.6a) a grafu SIN/PWM (graf: 5.6b) k zobrazení dynamických vlastností regulátoru. Pomocí nastavení fitness funkce ve formě kompletní penalizace jak ITAE, Wave, a Overshot hodnoty neboli *F111* dostáváme nejlepší hodnotu ze 100 spuštění algoritmu 2.703, která je oproti referenčnímu nastavení dle algoritmu (HC12) přibližně o $< 1\%$ větší.

| ⊞ | max | min | mean | median | std | runs |
|---|---|---|---|---|---|---|
| Výsledky [F100] | 6.739 | 2.123 | 2.358 | 2.253 | 0.628 | 100 |
| Iterace [F100] | 34 | 18 | 25.980 | 26 | 4.981 | |
| Výsledky [F110] | 5.616 | 2.230 | 2.507 | 2.386 | 0.603 | |
| Iterace [F110] | 33 | 16 | 24.270 | 24 | 4.954 | |
| Výsledky [F101] | 2.697 | 2.409 | 2.537 | 2.534 | 0.077 | |
| Iterace [F101] | 32 | 16 | 24.470 | 24.500 | 5.042 | |
| Výsledky [F111] | 6.776 | 2.703 | 3.053 | 2.880 | 0.781 | |
| Iterace [F111] | 34 | 16 | 24.630 | 24.500 | 5.374 | |

**Tabulka 5.4:** Tabulka statistiky všech opakovaní dle iterací a fitness hodnoty [Regulátor PID Alg DE] pro System4.

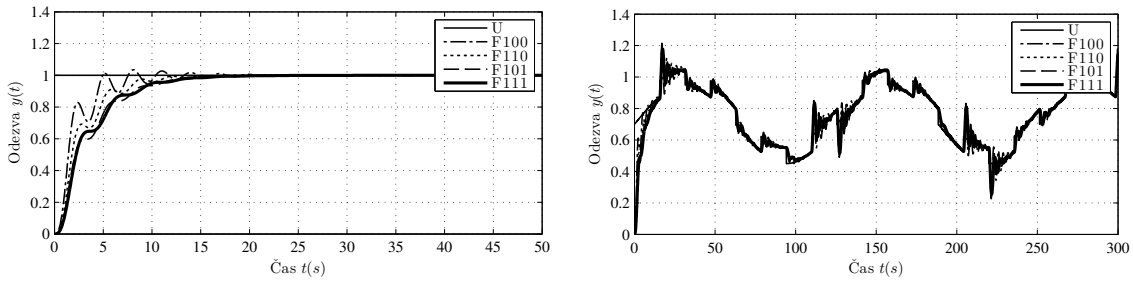
| ⊞ | $T_r$ | $T_s$ | $Min_s$ | $Max_s$ | Over | Under | Peak | $T_p$ |
|---|---|---|---|---|---|---|---|---|
| Výs. [F100] | 3.750 | 12.836 | 0.889 | 1.035 | 3.492 | | 1.035 | 8.100 |
| Výs. [F110] | 4.655 | 11.185 | 0.886 | | | | | 18.400 |
| Výs. [F101] | 7.242 | 14.192 | 0.902 | 1 | 0 | | 1 | 50 |
| Výs. [F111] | 7.604 | 13.344 | 0.900 | | | | | |

**Tabulka 5.5:** Tabulka ohodnocení odezvy na skokový signál dle nejlepších výsledků [Regulátor PID Alg DE] pro System4.

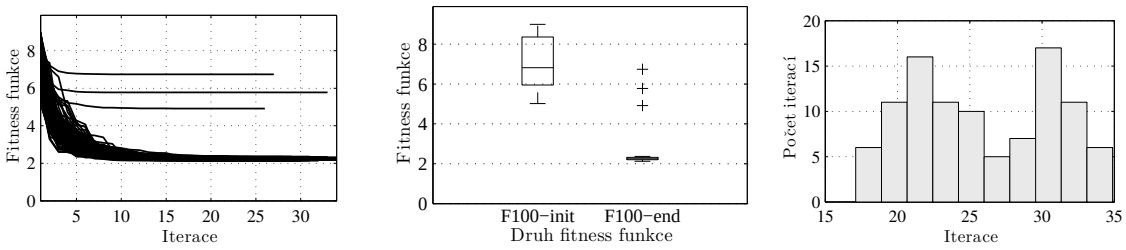


| ⊞ | P | I | D | Fit. | ITAE sin/pwm |
|---|---|---|---|---|---|
| Výs. [F100] | 17.842 | 0.004 | 2.014 | 2.123 | 132.619 |
| Výs. [F110] | 11.186 | 0.005 | 2.769 | 2.230 | 169.905 |
| Výs. [F101] | 8.856 | | 3.458 | 2.409 | 256.520 |
| Výs. [F111] | 6.999 | 0.006 | 3.405 | 2.703 | 252.904 |
| P:(tab: 4.1) | A:<0.0,5.0> | A:<0.0,0.5> | | ⊡ | |
| AI:(tab: 4.2) | NP [200], F [0.9], CR [0.5], GEN [200], RUNS [100] | | | | |

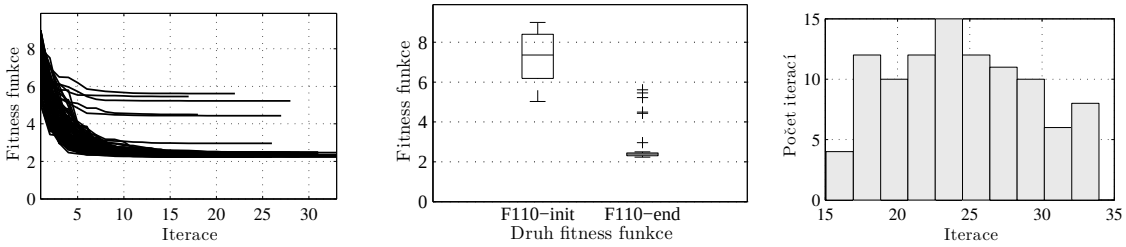
**Tabulka 5.6:** Tabulka nastavení regulátoru [Regulátor PID Alg DE] pro System4.



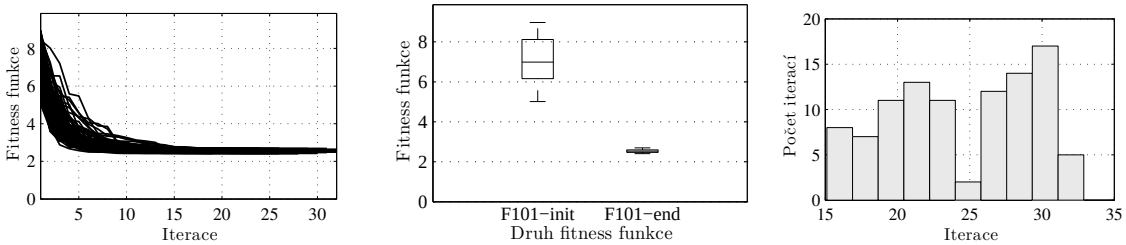

**Graf 5.6:** Pořadí grafů: (a: Průběh přechodové charakteristiky na skokovém signálu dle [Regulátor PID Alg DE] pro System4.) a (b: Průběh přechodové charakteristiky na SIN/PWM signálu dle [Regulátor PID Alg DE] pro System4.).



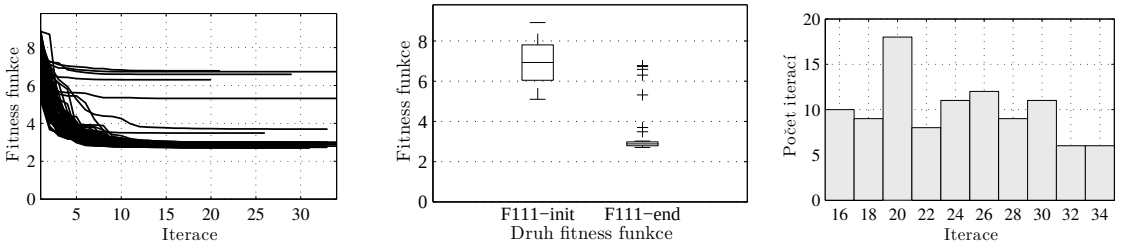

**Graf 5.7:** Průběh fitness hodnoty pro všechny opakovaní s histogramy iterací a ohodnocení fitness funkce [Regulátor PID Alg DE Typ F100] pro System4. Pořadí grafů: (a: Průběh fitness hodnoty.), (b: Box graf hodnoty fit. hod.) a (c: Histogram hodnoty iterací.).




**Graf 5.8:** Průběh fitness hodnoty pro všechny opakovaní s histogramy iterací a ohodnocení fitness funkce [Regulátor PID Alg DE Typ F110] pro System4. Pořadí grafů: (a: Průběh fitness hodnoty.), (b: Box graf hodnoty fit. hod.) a (c: Histogram hodnoty iterací.).




**Graf 5.9:** Průběh fitness hodnoty pro všechny opakovaní s histogramy iterací a ohodnocení fitness funkce [Regulátor PID Alg DE Typ F101] pro System4. Pořadí grafů: (a: Průběh fitness hodnoty.), (b: Box graf hodnoty fit. hod.) a (c: Histogram hodnoty iterací.).




**Graf 5.10:** Průběh fitness hodnoty pro všechny opakovaní s histogramy iterací a ohodnocení fitness funkce [Regulátor PID Alg DE Typ F111] pro System4. Pořadí grafů: (a: Průběh fitness hodnoty.), (b: Box graf hodnoty fit. hod.) a (c: Histogram hodnoty iterací.).

### 5.1.3 Nastavení dle algoritmu NM

Sekce nastavení (PID) regulátoru pro model *system4* dle algoritmu (NM). Regulátor je nastaven dle fitness funkce *F100*, *F110*, *F101* a *F111*. Kvalita algoritmu je ukázána na grafech *Průběh fitness hodnoty* pro *F100* (graf: 5.12a), *F110* (graf: 5.13a), *F101* (graf: 5.14a) a *F111* (graf: 5.15a), *Box graf hodnoty fitness* pro *F100* (graf: 5.12b), *F110* (graf: 5.13b), *F101* (graf: 5.14b) a *F111* (graf: 5.15b) a *Histogram hodnoty iterací* pro *F100* (graf: 5.12c), *F110* (graf: 5.13c), *F101* (graf: 5.14c) a *F111* (graf: 5.15c). Celková statistika algoritmu je v tabulce (tab: 5.7). Finální nastavení regulátoru je v tabulce (tab: 5.9) a ohodnocení přechodové funkce je v ta-

bulce (tab: 5.8). Hodnoty přechodových funkcí jsou zobrazeny na skokové funkci (graf: 5.11a) a grafu SIN/PWM (graf: 5.11b) k zobrazení dynamických vlastností regulátoru. Pomocí nastavení fitness funkce ve formě kompletní penalizace hodnoty neboli *F111* dostáváme nejlepší hodnotu ze 100 spuštění algoritmu 2.824, která je oproti referenčnímu nastavení dle algoritmu (HC12) přibližně o $< 5\%$ větší.

| ⊞ | max | min | mean | median | std | runs |
|---|---|---|---|---|---|---|
| Výsledky [F100] | 7.663 | 2.423 | 3.112 | 2.782 | 1.044 | 100 |
| Iterace [F100] | 24 | 9 | 15.900 | 16 | 4.076 | |
| Výsledky [F110] | 7.663 | 2.831 | 3.577 | 3.328 | 0.949 | |
| Iterace [F110] | 21 | 8 | 14.860 | 15 | 3.890 | |
| Výsledky [F101] | 7.903 | 2.992 | 3.702 | 3.476 | 0.859 | |
| Iterace [F101] | 21 | 8 | 14.860 | 15 | 3.838 | |
| Výsledky [F111] | 6.769 | 2.824 | 3.361 | 3.141 | 0.763 | |
| Iterace [F111] | 23 | 10 | 16.990 | 17 | 3.794 | |

**Tabulka 5.7:** Tabulka statistiky všech opakovaní dle iterací a fitness hodnoty [Regulátor PID Alg NM] pro System4.

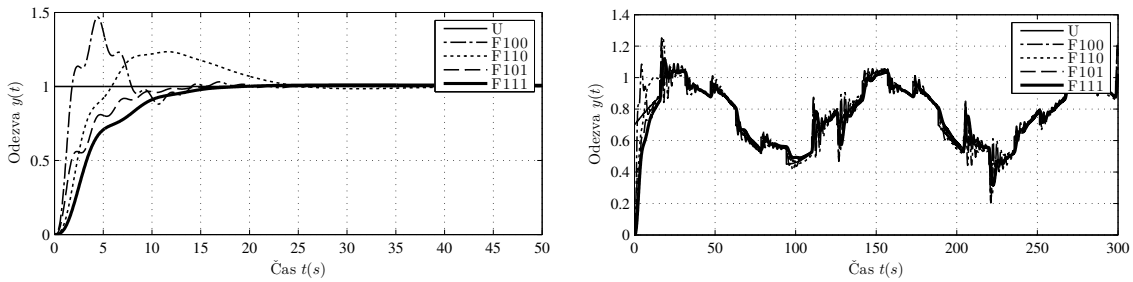
| ⊞ | $T_r$ | $T_s$ | $Min_s$ | $Max_s$ | Over | Under | Peak | $T_p$ |
|---|---|---|---|---|---|---|---|---|
| Výs. [F100] | 1.033 | 17.508 | 0.881 | 1.469 | 46.898 | 0 | 1.469 | 4.400 |
| Výs. [F110] | 3.569 | 23.250 | 0.901 | 1.235 | 23.475 | | 1.235 | 11.600 |
| Výs. [F101] | 5.910 | 11.530 | 0.903 | 1.002 | 0.224 | | 1.002 | 29.650 |
| Výs. [F111] | 8.186 | 15.096 | 0.902 | 1.010 | 1.016 | | 1.010 | 31.350 |

**Tabulka 5.8:** Tabulka ohodnocení odezvy na skokový signál dle nejlepších výsledků [Regulátor PID Alg NM] pro System4.

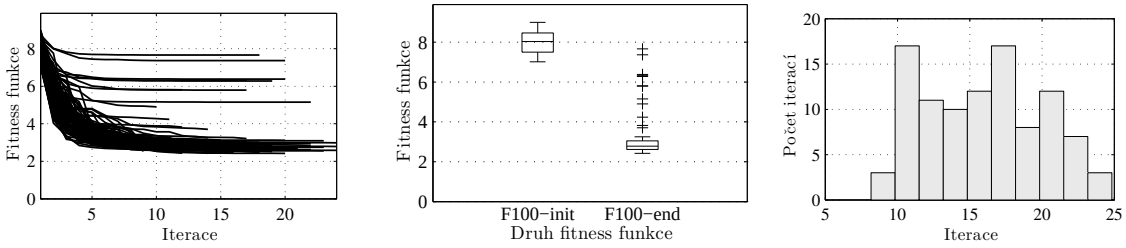


| ⊞ | P | I | D | Fit. | ITAE sin/pwm |
|---|---|---|---|---|---|
| Výs. [F100] | 24.800 | 0.646 | 2.126 | 2.423 | 264.866 |
| Výs. [F110] | 4.724 | 0.157 | 3.543 | 2.831 | 848.489 |
| Výs. [F101] | 16.140 | 0.004 | 3.071 | 2.992 | 229.170 |
| Výs. [F111] | 2.362 | 0.018 | 4.016 | 2.824 | 527.308 |
| P:(tab: 4.1) | A:<0.0,5.0> | A:<0.0,0.5> | | ⊡ | |
| AI:(tab: 4.2) | GEN [200], RUNS [100] | | | | |

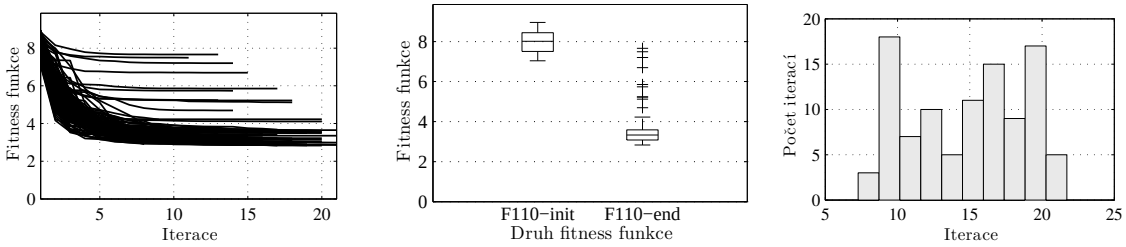
**Tabulka 5.9:** Tabulka nastavení regulátoru [Regulátor PID Alg NM] pro System4.



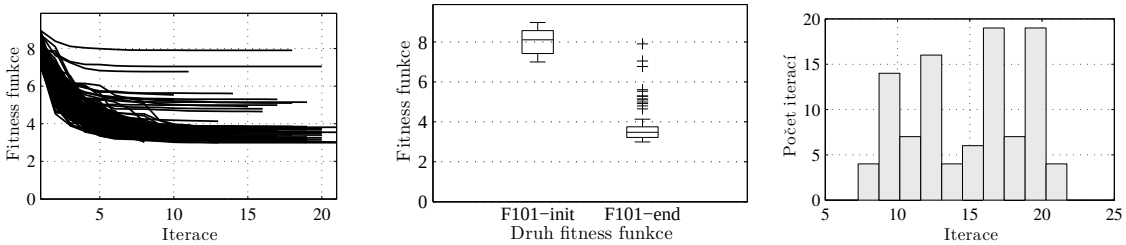

**Graf 5.11:** Pořadí grafů: (a: Průběh přechodové charakteristiky na skokovém signálu dle [Regulátor PID Alg NM] pro System4.) a (b: Průběh přechodové charakteristiky na SIN/PWM signálu dle [Regulátor PID Alg NM] pro System4.).



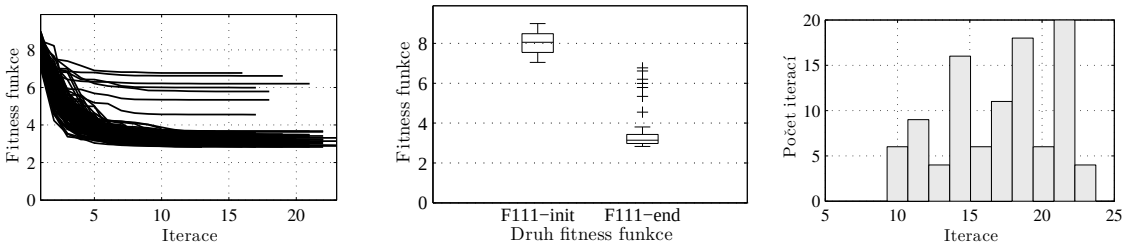

**Graf 5.12:** Průběh fitness hodnoty pro všechny opakovaní s histogramy iterací a ohodnocení fitness funkce [Regulátor PID Alg NM Typ F100] pro System4. Pořadí grafů: (a: Průběh fitness hodnoty.), (b: Box graf hodnoty fit. hod.) a (c: Histogram hodnoty iterací.).




**Graf 5.13:** Průběh fitness hodnoty pro všechny opakovaní s histogramy iterací a ohodnocení fitness funkce [Regulátor PID Alg NM Typ F110] pro System4. Pořadí grafů: (a: Průběh fitness hodnoty.), (b: Box graf hodnoty fit. hod.) a (c: Histogram hodnoty iterací.).




**Graf 5.14:** Průběh fitness hodnoty pro všechny opakovaní s histogramy iterací a ohodnocení fitness funkce [Regulátor PID Alg NM Typ F101] pro System4. Pořadí grafů: (a: Průběh fitness hodnoty.), (b: Box graf hodnoty fit. hod.) a (c: Histogram hodnoty iterací.).




**Graf 5.15:** Průběh fitness hodnoty pro všechny opakovaní s histogramy iterací a ohodnocení fitness funkce [Regulátor PID Alg NM Typ F111] pro System4. Pořadí grafů: (a: Průběh fitness hodnoty.), (b: Box graf hodnoty fit. hod.) a (c: Histogram hodnoty iterací.).

## 5.2 Model čtvrtého řádu s nelineárním prvkem

Nastavení regulátoru (PID) pro model *system4delay* pomocí algoritmu (HC12), (DE) a (NM). Následující sekce prezentuje samotné nastavení algoritmu, hodnotící funkce a konečné výsledky s vybraným statistickým porovnáním. Druhý systém, na kterém je prezentováno nastavení PID regulátoru je model čtvrtého řádu s nelineárním prvkem a je to i první model, který obsahuje nelineární prvek v podobě dopravního zpoždění, popis (sekce: 4.2.2 - viz str. 49). Model vychází z předešlého modelu, a proto je možno porovnat jednotlivé výsledky. Co se týče nastavení (PID) regulá-

toru na jednotkový skok jsou jednotlivé výsledky podobné, ale markantnější rozdíl nastává u vyhodnocení na dynamický impuls v podobně SIN/PWM signálu.

### 5.2.1 Nastavení dle algoritmu HC12

Sekce nastavení (PID) regulátoru pro model *system4delay* dle algoritmu (HC12). Regulátor je nastaven dle fitness funkce *F111*. Kvalita algoritmu je ukázána na grafech *Průběh fitness hodnoty* pro *F111* (graf: 5.17a), *Box graf hodnoty fitness* pro *F111* (graf: 5.17b) a *Histogram hodnoty iterací* pro *F111* (graf: 5.17c). Celková statistika algoritmu je v tabulce (tab: 5.10). Finální nastavení regulátoru je v tabulce (tab: 5.12) a ohodnocení přechodové funkce je v tabulce (tab: 5.11). Hodnoty přechodových funkcí jsou zobrazeny na skokové funkci (graf: 5.16a) a grafu SIN/PWM (graf: 5.16b) k zobrazení dynamických vlastností regulátoru.

| ⊞ | max | min | mean | median | std | runs |
|---|---|---|---|---|---|---|
| Výsledky [F111-A] | 6.172 | 3.654 | 3.958 | 3.821 | 0.446 | 100 |
| Iterace [F111-A] | 31 | 13 | 22.330 | 23 | 5.375 | |

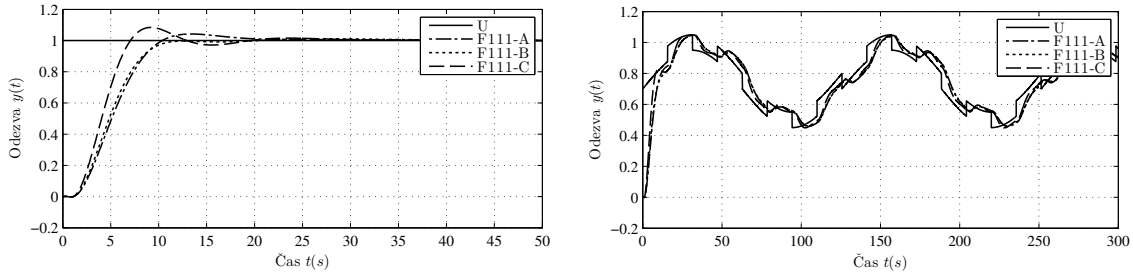
**Tabulka 5.10:** Tabulka statistiky všech opakovaní dle iterací a fitness hodnoty [Regulátor PID Alg HC12] pro System4Delay.

| ⊞ | $T_r$ | $T_s$ | $Min_s$ | $Max_s$ | Over | Under | Peak | $T_p$ |
|---|---|---|---|---|---|---|---|---|
| Výs. [F111-A] | 6.027 | 17.804 | 0.901 | 1.042 | 4.219 | 0.182 | 1.042 | 13.386 |
| Výs. [F111-B] | 5.977 | 10.403 | 0.900 | 1.012 | 1.200 | 0.211 | 1.012 | 27.023 |
| Výs. [F111-C] | 4.036 | 17.336 | 0.901 | 1.084 | 8.418 | 0.293 | 1.084 | 9.174 |

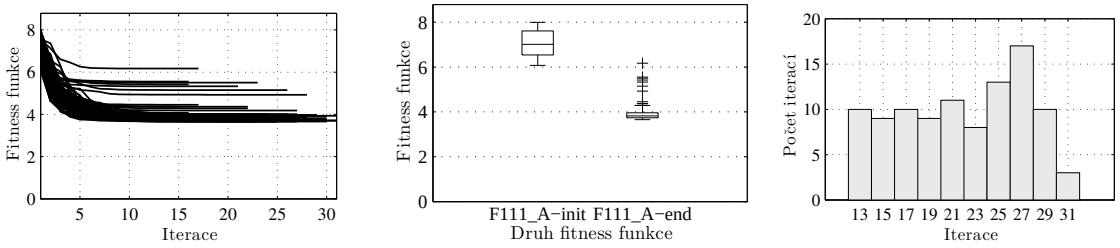
**Tabulka 5.11:** Tabulka ohodnocení odezvy na skokový signál dle nejlepších výsledků [Regulátor PID Alg HC12] pro System4Delay.

| ⊞ | P | I | D | Fit. | ITAE sin/pwm |
|---|---|---|---|---|---|
| Výs. [F111-A] | 0.339 | 29.499 | 0.974 | 3.654 | 564.028 |
| Výs. [F111-B] | 0.340 | 29.990 | 1.131 | 3.801 | 527.034 |
| Výs. [F111-C] | 0.458 | 22.458 | 1.572 | 4.037 | 453.504 |
| P:(tab: 4.1) | B:<0.0,50.0,15> | | | ⊡ | |
| AI:(tab: 4.2) | M12 [Default], RUNS [100] | | | | |

**Tabulka 5.12:** Tabulka nastavení regulátoru [Regulátor PID Alg HC12] pro System4Delay.




**Graf 5.16:** Pořadí grafů: (a: Průběh přechodové charakteristiky na skokovém signálu dle [Regulátor PID Alg HC12] pro System4Delay.) a (b: Průběh přechodové charakteristiky na SIN/PWM signálu dle [Regulátor PID Alg HC12] pro System4Delay.).




**Graf 5.17:** Průběh fitness hodnoty pro všechny opakovaní s histogramy iterací a ohodnocení fitness funkce [Regulátor PID Alg HC12 Typ F111-A] pro System4Delay. Pořadí grafů: (a: Průběh fitness hodnoty.), (b: Box graf hodnoty fit. hod.) a (c: Histogram hodnoty iterací.).

### 5.2.2 Nastavení dle algoritmu DE

Sekce nastavení (PID) regulátoru pro model *system4delay* dle algoritmu (DE). Regulátor je nastaven dle fitness funkce *F111*. Kvalita algoritmu je ukázána na grafech *Průběh fitness hodnoty* pro *F111* (graf: 5.19a), *Box graf hodnoty fitness* pro *F111* (graf: 5.19b) a *Histogram hodnoty iterací* pro *F111* (graf: 5.19c). Celková statistika algoritmu je v tabulce (tab: 5.13). Finální nastavení regulátoru je v tabulce (tab: 5.15) a ohodnocení přechodové funkce je v tabulce (tab: 5.14). Hodnoty přechodových funkcí jsou zobrazeny na skokové funkci (graf: 5.18a) a grafu SIN/PWM (graf: 5.18b) k zobrazení dynamických vlastností regulovaného systému. Pomocí nastavení fitness funkce ve formě kompletní penalizace jak ITAE, Wave a Overshot hodnoty neboli *F111* dostáváme nejlepší hodnotu ze 100 spuštění algoritmu 3.656, která je oproti referenčnímu nastavení dle algoritmu (HC12) přibližně o $< 1\%$ větší.



| ⊞ | max | min | mean | median | std | runs |
|---|---|---|---|---|---|---|
| Výsledky [F111-A] | 5.607 | 3.656 | 3.934 | 3.849 | 0.330 | 100 |
| Iterace [F111-A] | 35 | 16 | 25.500 | 25 | 6.141 | |

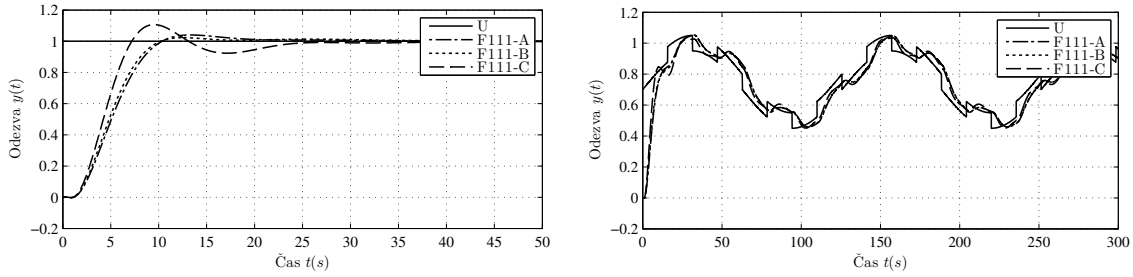
**Tabulka 5.13:** Tabulka statistiky všech opakovaní dle iterací a fitness hodnoty [Regulátor PID Alg DE] pro System4Delay.

| ⊞ | $T_r$ | $T_s$ | $Min_s$ | $Max_s$ | Over | Under | Peak | $T_p$ |
|---|---|---|---|---|---|---|---|---|
| Výs. [F111-A] | 6.004 | 17.610 | 0.901 | 1.040 | 4.021 | 0.185 | 1.040 | 13.344 |
| Výs. [F111-B] | 5.731 | 13.794 | 0.903 | 1.023 | 2.317 | 0.208 | 1.023 | 12.510 |
| Výs. [F111-C] | 4.159 | 23.206 | 0.902 | 1.107 | 10.679 | 0.254 | 1.107 | 9.508 |

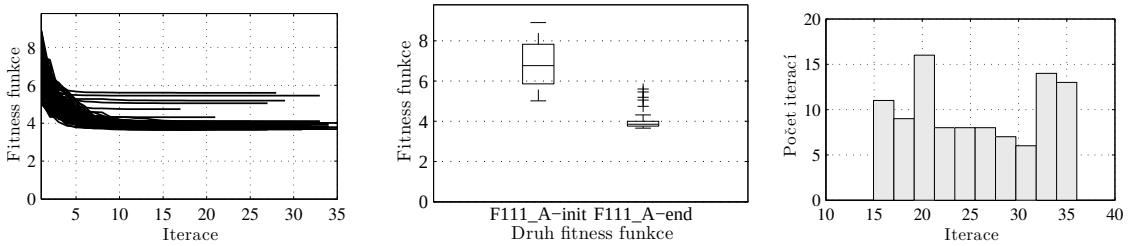
**Tabulka 5.14:** Tabulka ohodnocení odezvy na skokový signál dle nejlepších výsledků [Regulátor PID Alg DE] pro System4Delay.

| ⊞ | P | I | D | Fit. | ITAE sin/pwm |
|---|---|---|---|---|---|
| Výs. [F111-A] | 0.340 | 29.347 | 0.988 | 3.656 | 565.847 |
| Výs. [F111-B] | 0.350 | 28.374 | 1.113 | 3.807 | 534.557 |
| Výs. [F111-C] | 0.490 | 27.473 | 1.357 | 4.173 | 710.035 |
| P:(tab: 4.1) | A:<0.0,50.0> | | | ⊡ | |
| AI:(tab: 4.2) | NP [200], F [0.9], CR [0.5], GEN [200], RUNS [100] | | | | |

**Tabulka 5.15:** Tabulka nastavení regulátoru [Regulátor PID Alg DE] pro System4Delay.




**Graf 5.18:** Pořadí grafů: (a: Průběh přechodové charakteristiky na skokovém signálu dle [Regulátor PID Alg DE] pro System4Delay.) a (b: Průběh přechodové charakteristiky na SIN/PWM signálu dle [Regulátor PID Alg DE] pro System4Delay.).




**Graf 5.19:** Průběh fitness hodnoty pro všechny opakovaní s histogramy iterací a ohodnocení fitness funkce [Regulátor PID Alg DE Typ F111-A] pro System4Delay. Pořadí grafů: (a: Průběh fitness hodnoty.), (b: Box graf hodnoty fit. hod.) a (c: Histogram hodnoty iterací.).

### 5.2.3 Nastavení dle algoritmu NM



Sekce nastavení (PID) regulátoru pro model *system4delay* dle algoritmu (NM). Regulátor je nastaven dle fitness funkce *F111*. Kvalita algoritmu je ukázána na grafech *Průběh fitness hodnoty* pro *F111* (graf: 5.21a), *Box graf hodnoty fitness* pro *F111* (graf: 5.21b) a *Histogram hodnoty iterací* pro *F111* (graf: 5.21c). Celková statistika algoritmu je v tabulce (tab: 5.16). Finální nastavení regulátoru je v tabulce (tab: 5.18) a ohodnocení přechodové funkce je v tabulce (tab: 5.17). Hodnoty přechodových funkcí jsou zobrazeny na skokové funkci (graf: 5.20a) a grafu SIN/PWM (graf: 5.20b) k zobrazení dynamických vlastností regulátoru. Pomocí nastavení fitness funkce dostáváme nejlepší hodnotu ze 100 spuštění algoritmu 3.661, která je oproti referenčnímu nastavení dle algoritmu (HC12) přibližně o $< 1\%$ větší.

| ⊞ | max | min | mean | median | std | runs |
|---|---|---|---|---|---|---|
| Výsledky [F111-A] | 6.615 | 3.661 | 4.424 | 4.301 | 0.616 | 100 |
| Iterace [F111-A] | 24 | 7 | 16.070 | 16 | 4.717 | |

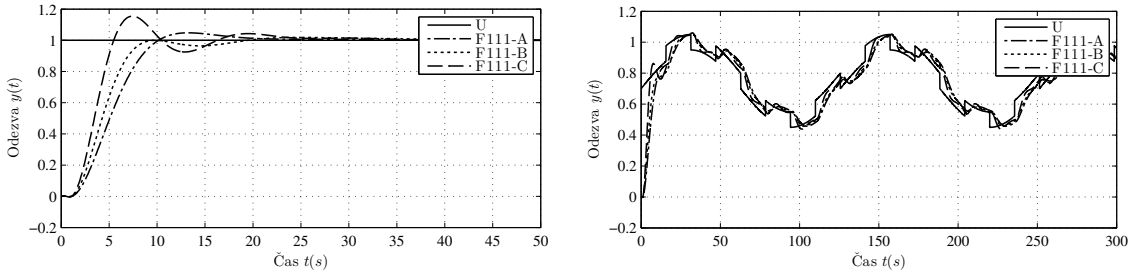
**Tabulka 5.16:** Tabulka statistiky všech opakovaní dle iterací a fitness hodnoty [Regulátor PID Alg NM] pro System4Delay.

| ⊞ | $T_r$ | $T_s$ | $Min_s$ | $Max_s$ | Over | Under | Peak | $T_p$ |
|---|---|---|---|---|---|---|---|---|
| Výs. [F111-A] | 5.934 | 18.235 | 0.903 | 1.048 | 4.784 | 0.183 | 1.048 | 13.261 |
| Výs. [F111-B] | 4.813 | 17.251 | 0.901 | 1.019 | 1.859 | 0.275 | 1.019 | 24.687 |
| Výs. [F111-C] | 3.030 | 23.236 | 0.900 | 1.155 | 15.490 | 0.418 | 1.155 | 7.423 |

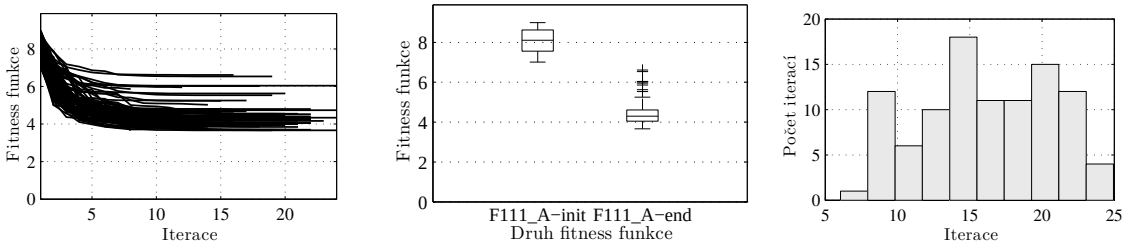
**Tabulka 5.17:** Tabulka ohodnocení odezvy na skokový signál dle nejlepších výsledků [Regulátor PID Alg NM] pro System4Delay.

| ⊞ | P | I | D | Fit. | ITAE sin/pwm |
|---|---|---|---|---|---|
| Výs. [F111-A] | 0.343 | 28.967 | 0.981 | 3.661 | 572.356 |
| Výs. [F111-B] | 0.384 | 26.184 | 1.474 | 3.820 | 550.481 |
| Výs. [F111-C] | 0.544 | 17.191 | 2.245 | 4.352 | 561.865 |
| P:(tab: 4.1) | A:<0.0,50.0> | | | ⊡ | |
| AI:(tab: 4.2) | GEN [200], RUNS [100] | | | | |

**Tabulka 5.18:** Tabulka nastavení regulátoru [Regulátor PID Alg NM] pro System4Delay.




**Graf 5.20:** Pořadí grafů: (a: Průběh přechodové charakteristiky na skokovém signálu dle [Regulátor PID Alg NM] pro System4Delay.) a (b: Průběh přechodové charakteristiky na SIN/PWM signálu dle [Regulátor PID Alg NM] pro System4Delay.).




**Graf 5.21:** Průběh fitness hodnoty pro všechny opakovaní s histogramy iterací a ohodnocení fitness funkce [Regulátor PID Alg NM Typ F111-A] pro System4Delay. Pořadí grafů: (a: Průběh fitness hodnoty.), (b: Box graf hodnoty fit. hod.) a (c: Histogram hodnoty iterací.).

## 5.3 Model kuličky v obruči

Nastavení regulátoru (PID) pro model *ballandloop* pomocí algoritmu (HC12), (DE) a (NM). Následující sekce prezentuje samotné nastavení algoritmu, hodnotící funkce a konečné výsledky s vybraným statistickým porovnáním. Popis systému je v sekci (sekce: 4.2.3 - viz str. 51) a oproti referenčnímu nastavení ve formě (ZN) vykazuje řešení lepší výsledky jak skokového impulzu, tak i dynamického PWM/SIN signálu.

### 5.3.1 Nastavení dle algoritmu HC12

Sekce nastavení (PID) regulátoru pro model *ballandloop* dle algoritmu (HC12). Regulátor je nastaven dle fitness funkce *F111*. Kvalita algoritmu je ukázána na grafech *Průběh fitness hodnoty* pro *F111* (graf: 5.23a), *Box graf hodnoty fitness* pro *F111* (graf: 5.23b) a *Histogram hodnoty iterací* pro *F111* (graf: 5.23c). Celková statistika algoritmu je v tabulce (tab: 5.19). Finální nastavení regulátoru je v tabulce

(tab: 5.21) a ohodnocení přechodové funkci je v tabulce (tab: 5.20). Hodnoty přechodových funkcí jsou zobrazeny na skokové funkci (graf: 5.22a) a grafu SIN/PWM (graf: 5.22b) k zobrazení dynamických vlastností regulátoru.

| ⊞ | max | min | mean | median | std | runs |
|---|---|---|---|---|---|---|
| Výsledky [F111-A] | 5.462 | 3.685 | 3.936 | 3.900 | 0.286 | 100 |
| Iterace [F111-A] | 37 | 16 | 27.420 | 28 | 6.193 | |

**Tabulka 5.19:** Tabulka statistiky všech opakovaní dle iterací a fitness hodnoty [Regulátor PID Alg HC12] pro BallAndLoop.

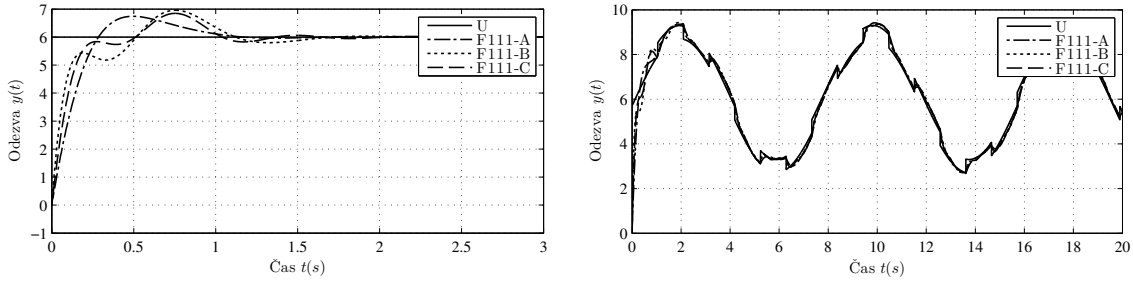
| ⊞ | $T_r$ | $T_s$ | $Min_s$ | $Max_s$ | Over | Under | Peak | $T_p$ |
|---|---|---|---|---|---|---|---|---|
| Výs. [F111-A] | 0.211 | 0.994 | 5.420 | 6.742 | 12.365 | 0 | 6.742 | 0.495 |
| Výs. [F111-B] | 0.159 | 1.504 | 5.179 | 6.959 | 15.978 | | 6.959 | 0.755 |
| Výs. [F111-C] | 0.165 | 1.258 | 5.410 | 6.846 | 14.104 | | 6.846 | 0.753 |

**Tabulka 5.20:** Tabulka ohodnocení odezvy na skokový signál dle nejlepších výsledků [Regulátor PID Alg HC12] pro BallAndLoop.

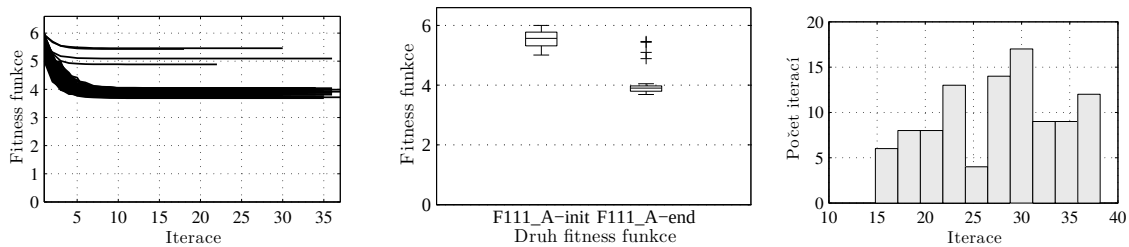
| ⊞ | P | I | D | Fit. | ITAE sin/pwm |
|---|---|---|---|---|---|
| Výs. [F111-A] | 0.165 | 0.105 | 0.120 | 3.685 | 115.405 |
| Výs. [F111-B] | 0.503 | 0.092 | 0.265 | 3.906 | 191.874 |
| Výs. [F111-C] | 0.505 | 0.097 | 0.174 | 4.054 | 141.556 |
| P:(tab: 4.1) | B:<0.0,10.0,10> | | | ⊡ | |
| AI:(tab: 4.2) | M12 [Default], RUNS [100] | | | | |

**Tabulka 5.21:** Tabulka nastavení regulátoru [Regulátor PID Alg HC12] pro BallAndLoop.






**Graf 5.22:** Pořadí grafů: (a: Průběh přechodové charakteristiky na skokovém signálu dle [Regulátor PID Alg HC12] pro BallAndLoop.) a (b: Průběh přechodové charakteristiky na SIN/PWM signálu dle [Regulátor PID Alg HC12] pro BallAndLoop.).




**Graf 5.23:** Průběh fitness hodnoty pro všechny opakovaní s histogramy iterací a ohodnocení fitness funkce [Regulátor PID Alg HC12 Typ F111-A] pro BallAndLoop. Pořadí grafů: (a: Průběh fitness hodnoty.), (b: Box graf hodnoty fit. hod.) a (c: Histogram hodnoty iterací.).

### 5.3.2 Nastavení dle algoritmu DE

Sekce nastavení (PID) regulátoru pro model *ballandloop* dle algoritmu (DE). Regulátor je nastaven dle fitness funkce *F111*. Kvalita algoritmu je ukázána na grafech *Průběh fitness hodnoty* pro *F111* (graf: 5.25a), *Box graf hodnoty fitness* pro *F111* (graf: 5.25b) a *Histogram hodnoty iterací* pro *F111* (graf: 5.25c). Celková statistika algoritmu je v tabulce (tab: 5.22). Finální nastavení regulátoru je v tabulce (tab: 5.24) a ohodnocení přechodové funkce je v tabulce (tab: 5.23). Hodnoty přechodových funkcí jsou zobrazeny na skokové funkci (graf: 5.24a) a grafu SIN/PWM (graf: 5.24b) k zobrazení dynamických vlastností regulátoru. Pomocí nastavení fitness funkce ve formě kompletní penalizace jak ITAE, Wave a Overshot hodnoty neboli *F111* dostáváme nejlepší hodnotu ze 100 spuštění algoritmu 3.569, která je oproti referenčnímu nastavení dle algoritmu (HC12) přibližně o $< 5\%$ menší.

| ⊞ | max | min | mean | median | std | runs |
|---|---|---|---|---|---|---|
| Výsledky [F111-A] | 4.897 | 3.569 | 3.801 | 3.773 | 0.182 | 100 |
| Iterace [F111-A] | 30 | 14 | 21.990 | 22.500 | 4.364 | |

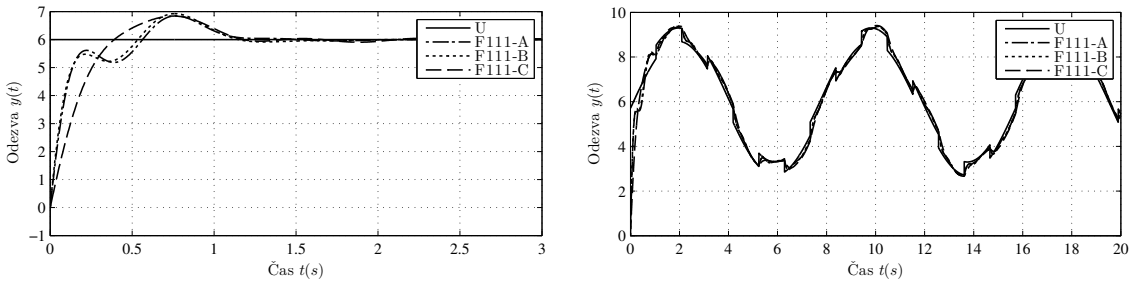
**Tabulka 5.22:** Tabulka statistiky všech opakovaní dle iterací a fitness hodnoty [Regulátor PID Alg DE] pro BallAndLoop.

| ⊞ | $T_r$ | $T_s$ | $Min_s$ | $Max_s$ | Over | Under | Peak | $T_p$ |
|---|---|---|---|---|---|---|---|---|
| Výs. [F111-A] | 0.157 | 1.098 | 5.183 | 6.850 | 14.160 | 0 | 6.850 | 0.768 |
| Výs. [F111-B] | 0.168 | 1.080 | 5.221 | 6.926 | 15.425 | | 6.926 | 0.760 |
| Výs. [F111-C] | 0.275 | 1.106 | 5.421 | 6.840 | 13.999 | | 6.840 | 0.753 |

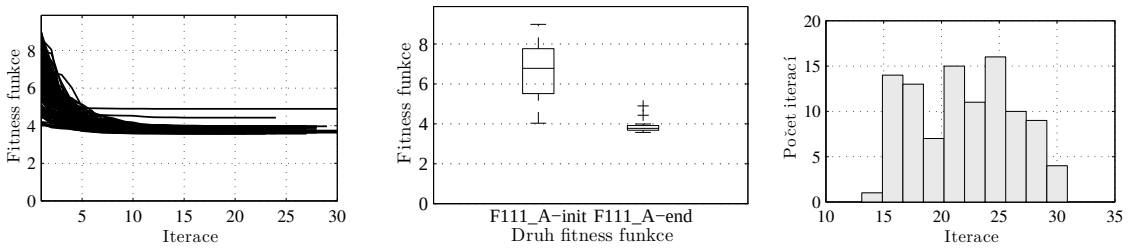
**Tabulka 5.23:** Tabulka ohodnocení odezvy na skokový signál dle nejlepších výsledků [Regulátor PID Alg DE] pro BallAndLoop.

| ⊞ | P | I | D | Fit. | ITAE sin/pwm |
|---|---|---|---|---|---|
| Výs. [F111-A] | 0.719 | 0.116 | 0.188 | 3.569 | 154.345 |
| Výs. [F111-B] | 0.569 | 0.104 | 0.217 | 3.691 | 163.787 |
| Výs. [F111-C] | 0.151 | 0.148 | 0.093 | 4.117 | 204.358 |
| P:(tab: 4.1) | A:<0.0,10.0> | | | ⊡ | |
| AI:(tab: 4.2) | NP [200], F [0.9], CR [0.5], GEN [200], RUNS [100] | | | | |

**Tabulka 5.24:** Tabulka nastavení regulátoru [Regulátor PID Alg DE] pro BallAndLoop.




**Graf 5.24:** Pořadí grafů: (a: Průběh přechodové charakteristiky na skokovém signálu dle [Regulátor PID Alg DE] pro BallAndLoop.) a (b: Průběh přechodové charakteristiky na SIN/PWM signálu dle [Regulátor PID Alg DE] pro BallAndLoop.).




**Graf 5.25:** Průběh fitness hodnoty pro všechny opakovaní s histogramy iterací a ohodnocení fitness funkce [Regulátor PID Alg DE Typ F111-A] pro BallAndLoop. Pořadí grafů: (a: Průběh fitness hodnoty.), (b: Box graf hodnoty fit. hod.) a (c: Histogram hodnoty iterací.).

### 5.3.3 Nastavení dle algoritmu NM

Sekce nastavení (PID) regulátoru pro model *ballandloop* dle algoritmu (NM). Regulátor je nastaven dle fitness funkce *F111*. Kvalita algoritmu je ukázána na grafech *Průběh fitness hodnoty* pro *F111* (graf: 5.27a), *Box graf hodnoty fitness* pro *F111* (graf: 5.27b) a *Histogram hodnoty iterací* pro *F111* (graf: 5.27c). Celková statistika algoritmu je v tabulce (tab: 5.25). Finální nastavení regulátoru je v tabulce (tab: 5.27) a ohodnocení přechodové funkce je v tabulce (tab: 5.26). Hodnoty přechodových funkcí jsou zobrazeny na skokové funkci (graf: 5.26a) a grafu SIN/PWM (graf: 5.26b) k zobrazení dynamických vlastností regulátoru. Pomocí nastavení fitness funkce ve formě kompletní penalizace jak ITAE, Wave a Overshot hodnoty neboli *F111* dostáváme nejlepší hodnotu ze 100 spuštění algoritmu 3.746, která je oproti referenčnímu nastavení dle algoritmu (HC12) přibližně o $< 2\%$ větší.



| ⊞ | max | min | mean | median | std | runs |
|---|---|---|---|---|---|---|
| Výsledky [F111-A] | 7.569 | 3.746 | 4.499 | 4.410 | 0.640 | 100 |
| Iterace [F111-A] | 26 | 11 | 18.440 | 18.500 | 4.641 | |

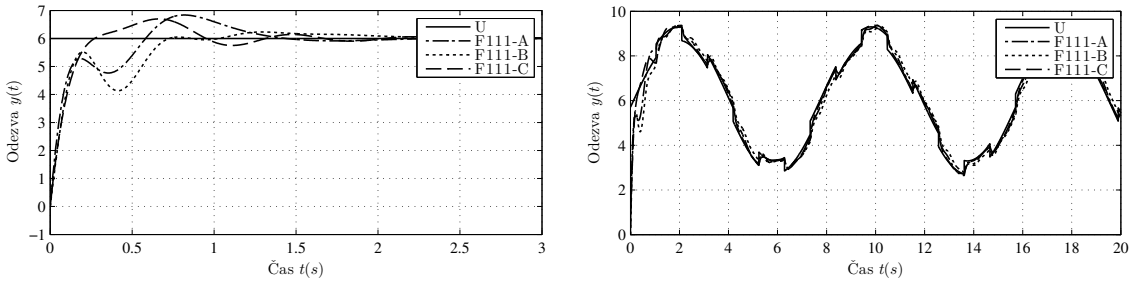
**Tabulka 5.25:** Tabulka statistiky všech opakovaní dle iterací a fitness hodnoty [Regulátor PID Alg NM] pro BallAndLoop.

| ⊞ | $T_r$ | $T_s$ | $Min_s$ | $Max_s$ | Over | Under | Peak | $T_p$ |
|---|---|---|---|---|---|---|---|---|
| Výs. [F111-A] | 0.492 | 1.298 | 5.410 | 6.838 | 13.962 | 0 | 6.838 | 0.818 |
| Výs. [F111-B] | 0.156 | 1.873 | 4.134 | 6.232 | 3.862 | | 6.232 | 1.295 |
| Výs. [F111-C] | 0.170 | 1.542 | 5.406 | 6.698 | 11.632 | | 6.698 | 0.665 |

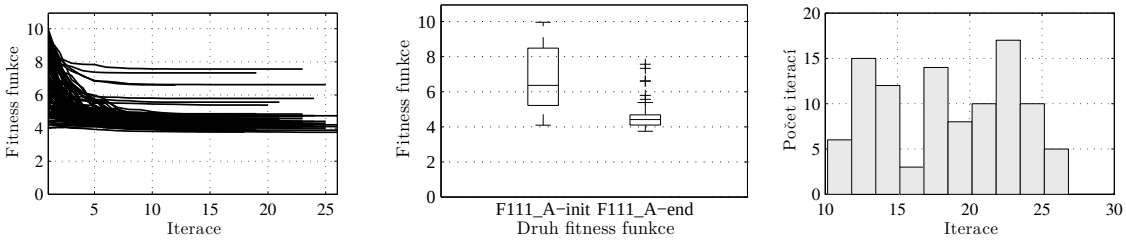
**Tabulka 5.26:** Tabulka ohodnocení odezvy na skokový signál dle nejlepších výsledků [Regulátor PID Alg NM] pro BallAndLoop.

| ⊞ | P | I | D | Fit. | ITAE sin/pwm |
|---|---|---|---|---|---|
| Výs. [F111-A] | 0.626 | 0.128 | 0.238 | 3.746 | 232.092 |
| Výs. [F111-B] | 1.164 | 0.261 | 0.158 | 3.826 | 223.068 |
| Výs. [F111-C] | 0.257 | 0.081 | 0.183 | 4.042 | 172.024 |
| P:(tab: 4.1) | A:<0.0,10.0> | | | ⊡ | |
| AI:(tab: 4.2) | GEN [200], RUNS [100] | | | | |

**Tabulka 5.27:** Tabulka nastavení regulátoru [Regulátor PID Alg NM] pro BallAndLoop.




**Graf 5.26:** Pořadí grafů: (a: Průběh přechodové charakteristiky na skokovém signálu dle [Regulátor PID Alg NM] pro BallAndLoop.) a (b: Průběh přechodové charakteristiky na SIN/PWM signálu dle [Regulátor PID Alg NM] pro BallAndLoop.).




**Graf 5.27:** Průběh fitness hodnoty pro všechny opakovaní s histogramy iterací a ohodnocení fitness funkce [Regulátor PID Alg NM Typ F111-A] pro BallAndLoop. Pořadí grafů: (a: Průběh fitness hodnoty.), (b: Box graf hodnoty fit. hod.) a (c: Histogram hodnoty iterací.).

## 5.4 Model magnetické levitace

5

Nastavení regulátoru (PID) pro model *maglev* pomocí algoritmu (HC12), (DE) a (NM). Následující sekce prezentuje samotné nastavení algoritmu, hodnotící funkce a konečné výsledky s vybraným statistickým porovnáním. Posledním modelem, na kterém je prezentováno nastavení obecného regulátoru typu (PID) je model magnetické levitace. Model je popsán v sekci (sekce: 4.2.4 - viz str. 54) a je porovnáván s referenčním nastavením regulátoru dle firmy Humusoft. Tento systém je specifický tím, že je vysoce nelineární systém, který je obtížně regulovatelný. Jedná se také o reálný model, a proto jsou jednotlivé výsledky porovnávány jak na simulačním modelu, tak na skutečném modelu s reálným měřením odchylky levitující kuličky (graf: 5.34). Přechodová charakteristika PWM ukazuje limity regulátoru a možnosti zlepšení pro další typy nastavení.

### 5.4.1 Nastavení dle algoritmu HC12

Sekce nastavení (PID) regulátoru pro model *maglev* dle algoritmu (HC12). Regulátor je nastaven dle fitness funkce *F111*. Kvalita algoritmu je ukázána na grafech *Průběh fitness hodnoty* pro *F111* (graf: 5.29a), *Box graf hodnoty fitness* pro *F111* (graf: 5.29b) a *Histogram hodnoty iterací* pro *F111* (graf: 5.29c). Celková statistika algoritmu je v tabulce (tab: 5.28). Finální nastavení regulátoru je v tabulce (tab: 5.30) a ohodnocení přechodové funkce je v tabulce (tab: 5.29). Hodnoty přechodových funkcí jsou zobrazeny na skokové funkci (graf: 5.28a) a grafu PWM reálné soustavy (graf: 5.28b) k zobrazení dynamických vlastností regulátoru.

| ⊞ | max | min | mean | median | std | runs |
|---|---|---|---|---|---|---|
| Výsledky [F111-A] | 4.121 | 2.058 | 2.233 | 2.177 | 0.295 | 100 |
| Iterace [F111-A] | 39 | 11 | 26.530 | 26 | 8.347 | |

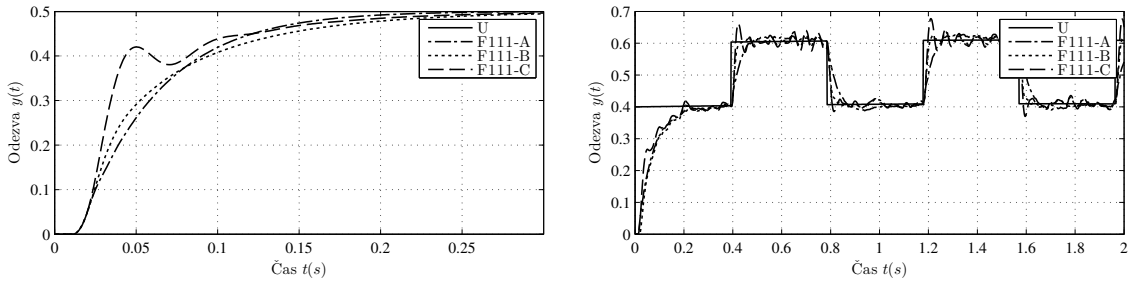
**Tabulka 5.28:** Tabulka statistiky všech opakovaní dle iterací a fitness hodnoty [Regulátor PID Alg HC12] pro Maglev.

| ⊞ | $T_r$ | $T_s$ | $Min_s$ | $Max_s$ | Over | Under | Peak | $T_p$ |
|---|---|---|---|---|---|---|---|---|
| Výs. [F111-A] | 0.101 | 0.191 | 0.451 | 0.500 | | | 0.500 | |
| Výs. [F111-B] | 0.121 | 0.255 | 0.450 | 0.495 | 0 | | 0.495 | 0.300 |
| Výs. [F111-C] | 0.101 | 0.230 | | 0.496 | | | 0.496 | |

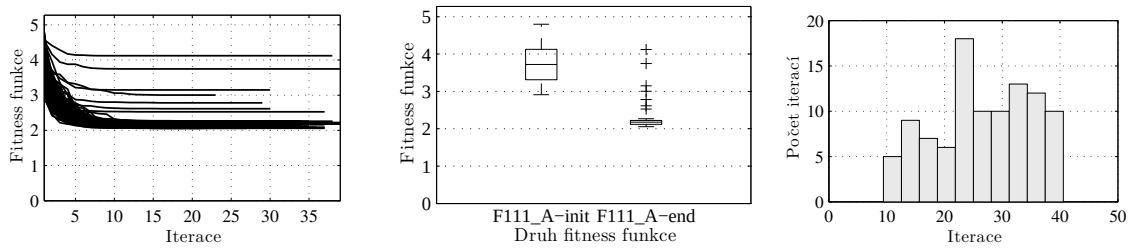
**Tabulka 5.29:** Tabulka ohodnocení odezvy na skokový signál dle nejlepších výsledků [Regulátor PID Alg HC12] pro Maglev.

| ⊞ | P | I | D | Fit. | ITAE sin/pwm |
|---|---|---|---|---|---|
| Výs. [F111-A] | 3.953 | 0.141 | 0.036 | 2.058 | 1.438 |
| Výs. [F111-B] | 2.059 | 0.078 | 0.018 | 2.267 | 1.851 |
| Výs. [F111-C] | 2.330 | 0.070 | 0.012 | 2.300 | 1.326 |
| P:(tab: 4.1) | B:<0.0,5.0,12> | B:<0.0,0.2,10> | | ⊡ | |
| AI:(tab: 4.2) | M12 [Default], RUNS [100] | | | | |

**Tabulka 5.30:** Tabulka nastavení regulátoru [Regulátor PID Alg HC12] pro Maglev.




**Graf 5.28:** Pořadí grafů: (a: Průběh přechodové charakteristiky na skokovém signálu dle [Regulátor PID Alg HC12] pro Maglev.) a (b: Průběh přechodové charakteristiky na SIN/PWM signálu dle [Regulátor PID Alg HC12] pro Maglev.).




**Graf 5.29:** Průběh fitness hodnoty pro všechny opakovaní s histogramy iterací a ohodnocení fitness funkce [Regulátor PID Alg HC12 Typ F111-A] pro Maglev. Pořadí grafů: (a: Průběh fitness hodnoty.), (b: Box graf hodnoty fit. hod.) a (c: Histogram hodnoty iterací.).

## 5.4.2 Nastavení dle algoritmu DE

Sekce nastavení (PID) regulátoru pro model *maglev* dle algoritmu (DE). Regulátor je nastaven dle fitness funkce *F111*. Kvalita algoritmu je ukázána na grafech

*Průběh fitness hodnoty* pro *F111* (graf: 5.31a), *Box graf hodnoty fitness* pro *F111* (graf: 5.31b) a *Histogram hodnoty iterací* pro *F111* (graf: 5.31c). Celková statistika algoritmu je v tabulce (tab: 5.31). Finální nastavení regulátoru je v tabulce (tab: 5.33) a ohodnocení přechodové funkce je v tabulce (tab: 5.32). Hodnoty přechodových funkcí jsou zobrazeny na skokové funkci (graf: 5.30a) a grafu SIN/PWM reálné soustavy (graf: 5.30b) k zobrazení dynamických vlastností regulátoru. Pomocí nastavení fitness funkce ve formě kompletní penalizace jak ITAE, Wave a Overshot hodnoty neboli *F111* dostáváme nejlepší hodnotu ze 100 spuštění algoritmu 2.141, která je oproti referenčnímu nastavení dle algoritmu (HC12) přibližně o $< 5\%$ větší. Při testech na reálném modelu se jednotlivé nejlepší nastavené regulátory pomocí různých algoritmů chovají velice podobně, a proto lze říci, že každý algoritmus našel optimální řešení regulátoru. Celkové porovnání úspěšnosti algoritmů v této sekci je shrnuto v kapitole porovnání výsledků pro předchozí modely(sekce: 8 - viz str. 107).

| ⊞ | max | min | mean | median | std | runs |
|---|---|---|---|---|---|---|
| Výsledky [F111-A] | 4.144 | 2.141 | 2.307 | 2.269 | 0.250 | 100 |
| Iterace [F111-A] | 28 | 20 | 23.650 | 23 | 2.418 | |

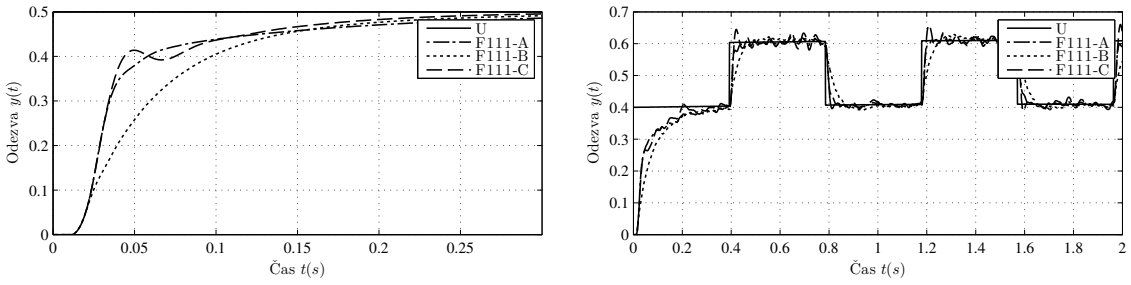
**Tabulka 5.31:** Tabulka statistiky všech opakovaní dle iterací a fitness hodnoty [Regulátor PID Alg DE] pro Maglev.

| ⊞ | $T_r$ | $T_s$ | $Min_s$ | $Max_s$ | Over | Under | Peak | $T_p$ |
|---|---|---|---|---|---|---|---|---|
| Výs. [F111-A] | 0.107 | NaN | 0.450 | 0.486 | 0 | | 0.486 | 0.300 |
| Výs. [F111-B] | 0.120 | 0.285 | | 0.491 | | | 0.491 | |
| Výs. [F111-C] | 0.100 | 0.241 | | 0.495 | | | 0.495 | |

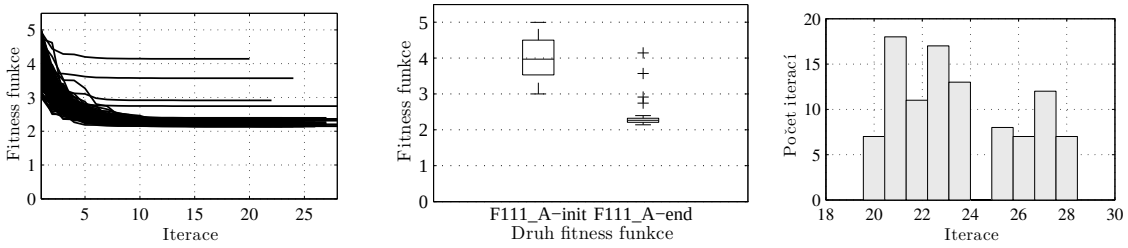
**Tabulka 5.32:** Tabulka ohodnocení odezvy na skokový signál dle nejlepších výsledků [Regulátor PID Alg DE] pro Maglev.

| ⊞ | P | I | D | Fit. | ITAE sin/pwm |
|---|---|---|---|---|---|
| Výs. [F111-A] | 4.161 | 0.142 | 0.015 | 2.141 | 1.740 |
| Výs. [F111-B] | 3.638 | 0.148 | 0.034 | 2.303 | 2.011 |
| Výs. [F111-C] | 2.601 | 0.078 | 0.013 | 2.309 | 1.354 |
| P:(tab: 4.1) | A:<0.0,5.0> | A:<0.0,0.2> | | ⊡ | |
| AI:(tab: 4.2) | NP [200], F [0.9], CR [0.5], GEN [200], RUNS [100] | | | | |

**Tabulka 5.33:** Tabulka nastavení regulátoru [Regulátor PID Alg DE] pro Maglev.




**Graf 5.30:** Pořadí grafů: (a: Průběh přechodové charakteristiky na skokovém signálu dle [Regulátor PID Alg DE] pro Maglev.) a (b: Průběh přechodové charakteristiky na SIN/PWM signálu dle [Regulátor PID Alg DE] pro Maglev.).




**Graf 5.31:** Průběh fitness hodnoty pro všechny opakovaní s histogramy iterací a ohodnocení fitness funkce [Regulátor PID Alg DE Typ F111-A] pro Maglev. Pořadí grafů: (a: Průběh fitness hodnoty.), (b: Box graf hodnoty fit. hod.) a (c: Histogram hodnoty iterací.).

### 5.4.3 Nastavení dle algoritmu NM

Sekce nastavení (PID) regulátoru pro model *maglev* dle algoritmu (NM). Regulátor je nastaven dle fitness funkce *F111*. Kvalita algoritmu je ukázána na grafech *Průběh fitness hodnoty* pro *F111* (graf: 5.33a), *Box graf hodnoty fitness* pro *F111* (graf: 5.33b) a *Histogram hodnoty iterací* pro *F111* (graf: 5.33c). Celková statistika algoritmu je v tabulce (tab: 5.34). Finální nastavení regulátoru je v tabulce (tab: 5.36) a ohodnocení přechodové funkce je v tabulce (tab: 5.35). Hodnoty přechodových funkcí jsou zobrazeny na skokové funkci (graf: 5.32a) a grafu SIN/PWM reálné soustavy (graf: 5.32b) k zobrazení dynamických vlastností regulátoru. Pomocí nastavení fitness funkce ve formě kompletní penalizace jak ITAE, Wave a Overshot hodnoty neboli *F111* dostáváme nejlepší hodnotu ze 100 spuštění algoritmu 2.113, která je oproti referenčnímu nastavení dle algoritmu (HC12) přibližně o $< 3\%$ větší.



| ⊞ | max | min | mean | median | std | runs |
|---|---|---|---|---|---|---|
| Výsledky [F111-A] | 4.513 | 2.113 | 2.660 | 2.525 | 0.476 | 100 |
| Iterace [F111-A] | 18 | 12 | 15.090 | 15 | 1.832 | |

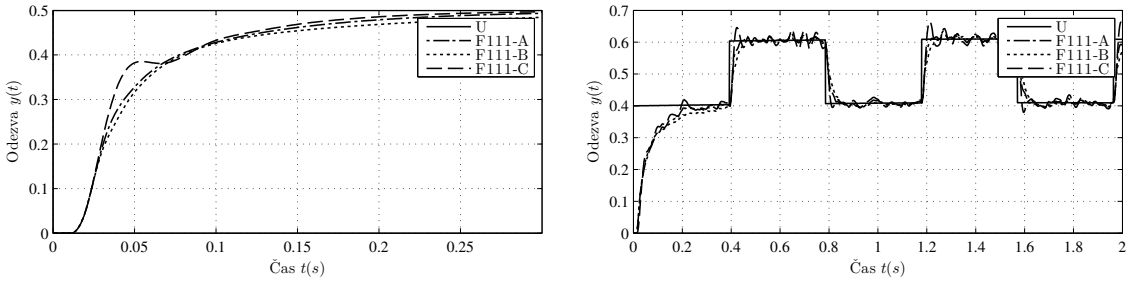
**Tabulka 5.34:** Tabulka statistiky všech opakovaní dle iterací a fitness hodnoty [Regulátor PID Alg NM] pro Maglev.

| ⊞ | $T_r$ | $T_s$ | $Min_s$ | $Max_s$ | Over | Under | Peak | $T_p$ |
|---|---|---|---|---|---|---|---|---|
| Výs. [F111-A] | 0.108 | 0.264 | 0.450 | 0.493 | 0 | | 0.493 | 0.300 |
| Výs. [F111-B] | 0.119 | NaN | | 0.484 | | | 0.484 | |
| Výs. [F111-C] | 0.100 | 0.224 | 0.451 | 0.497 | | | 0.497 | |

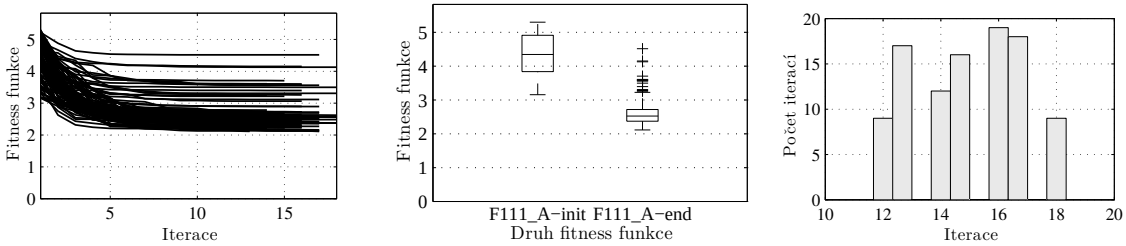
**Tabulka 5.35:** Tabulka ohodnocení odezvy na skokový signál dle nejlepších výsledků [Regulátor PID Alg NM] pro Maglev.

| ⊞ | P | I | D | Fit. | ITAE sin/pwm |
|---|---|---|---|---|---|
| Výs. [F111-A] | 2.752 | 0.095 | 0.019 | 2.113 | 1.634 |
| Výs. [F111-B] | 4.039 | 0.158 | 0.024 | 2.308 | 2.032 |
| Výs. [F111-C] | 2.149 | 0.067 | 0.013 | 2.417 | 1.353 |
| P:(tab: 4.1) | A:<0.0,5.0> | A:<0.0,0.2> | | ⊡ | |
| AI:(tab: 4.2) | GEN [200], RUNS [100] | | | | |

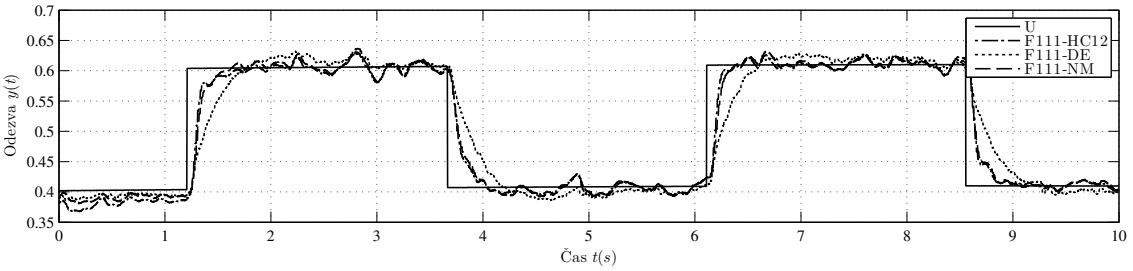
**Tabulka 5.36:** Tabulka nastavení regulátoru [Regulátor PID Alg NM] pro Maglev.




**Graf 5.32:** Pořadí grafů: (a: Průběh přechodové charakteristiky na skokovém signálu dle [Regulátor PID Alg NM] pro Maglev.) a (b: Průběh přechodové charakteristiky na SIN/PWM signálu dle [Regulátor PID Alg NM] pro Maglev.).

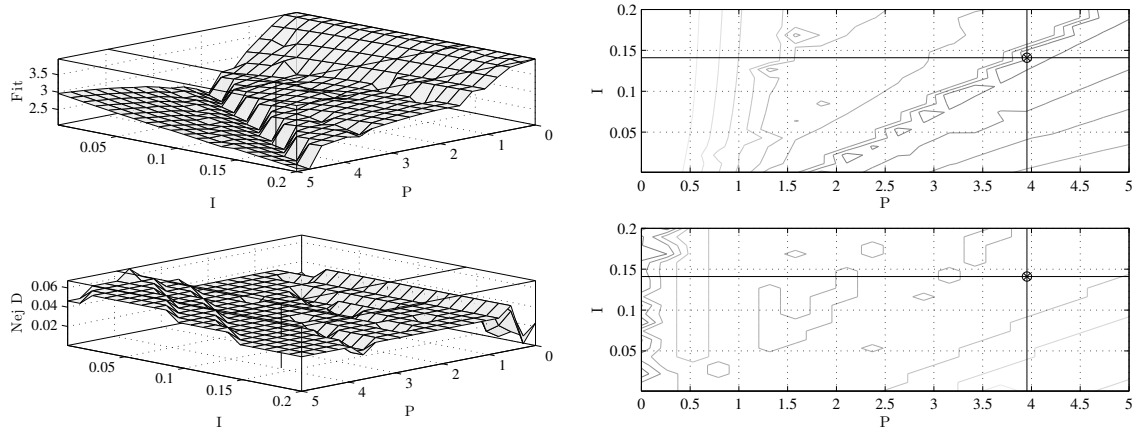



**Graf 5.33:** Průběh fitness hodnoty pro všechny opakovaní s histogramy iterací a ohodnocení fitness funkce [Regulátor PID Alg NM Typ F111-A] pro Maglev. Pořadí grafů: (a: Průběh fitness hodnoty.), (b: Box graf hodnoty fit. hod.) a (c: Histogram hodnoty iterací.).

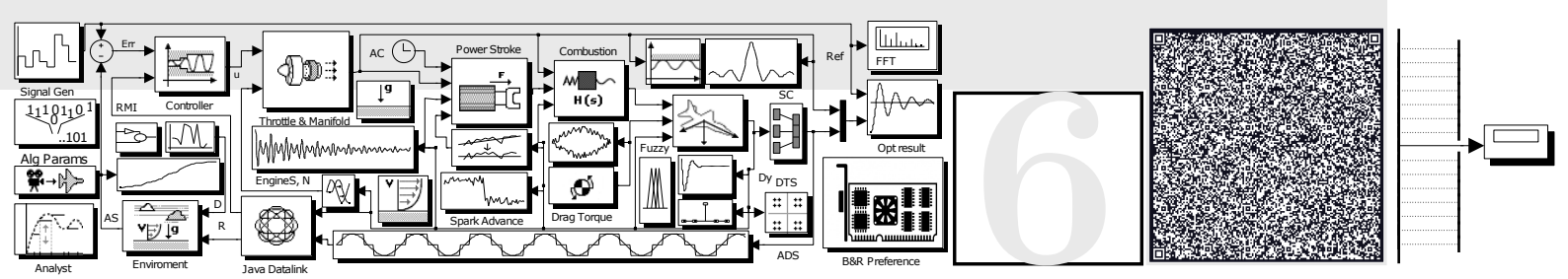



**Graf 5.34:** Průběh přechodové charakteristiky na reálné soustavě pro Maglev.

Pro model magnetické levitace můžeme prezentovat výsledek a kompletní rozložení fitness hodnoty, a to konkrétně pro všechny hodnoty parametrů $P$, $I$ a výběrem nejlepší hodnoty $D$ v grafech (graf: 5.35a) a (graf: 5.35b). Samotné nejlepší hodnoty parametru $D$ jsou znázorněny v grafech (graf: 5.35c) a (graf: 5.35d).




**Graf 5.35:** Grafická podoba fitness pro [Maglev]. Pořadí grafů: (a: Fitness hodnota F111 v závislosti na PI hodnotách.), (b: Konturový graf.), (c: Nejlepší hodnota D dle fitness hodnoty.) a (d: Konturový graf.).




# KAPITOLA 6

# NASTAVENÍ FUZZY PID REGULÁTORU

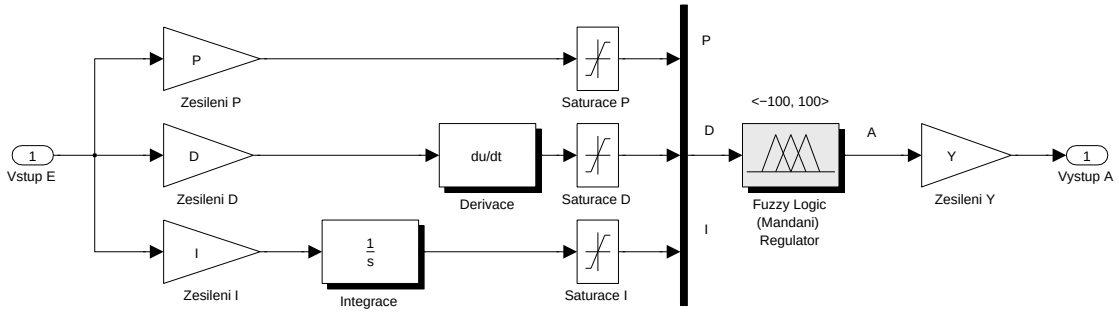
Druhý příklad nastavení regulátoru je nastavení fuzzy logického kontroléru s využitím znalostí z (PID) regulátoru. Fuzzy kontrolér spadá do oblasti pokročilého nastavení a oproti samostatnému (PID) má schopnost být velice robustním regulátorem. Tato kapitola kombinuje znalosti z předchozí kapitoly, kdy struktura optimalizovaného regulátoru je kombinací PID struktury (kde se využívají jak derivační a integrační člen) a fuzzy logiky. Tento přístup dává vetší prostor nastavení regulátoru oproti klasickému PID, kdy můžeme vhodně kombinovat funkce příslušnosti, z hlediska jejich tvarů a pozice a vstupních/výstupních zesilovačů k fuzzy kontroléru. Celková struktura optimalizovaného regulátoru je zobrazena na modelu (model: 6.1) a je využívána pro všechny typy modelů v této kapitole.





Struktura samotného fuzzy kontroléru obsahuje tři vstupy, každý s třemi funkcemi příslušnosti, popsanými Gaussovou rovnicí přímky, pozicí a rozptylem. Jeden výstup s jedenácti funkcemi příslušnosti, tak abychom popsali všechny možnosti vstupů a výstupů. Všechny funkce příslušnosti mají rozsah od (-100 do 100). Celková struktura fuzzy regulátoru je tedy popsaná dvaceti-sedmi bázovými pravidly dle tabulky (tab: 6.1). Celková komplexnost regulátoru tedy dává možnost nastavení až čtyřiceti parametrů (dva pro každou funkci příslušnosti). K vypočítání hodnoty výstupu z báze pravidel se využívá metody *centroid*. Centriod je funkce defuzzifikace, kde se výstup ztotožňuje s těžištěm celkové plochy všech pravidel (vz: 6.1) a (obr: 6.1). Funkční pravidla jsou popsána logickou operací *AND*, tedy minimální

funkcí z výsledku funkčního pravidla, operátor $AND$ je popsán ve vzorci (vz: 2.31). Fuzzy regulátor je typu Mandani.



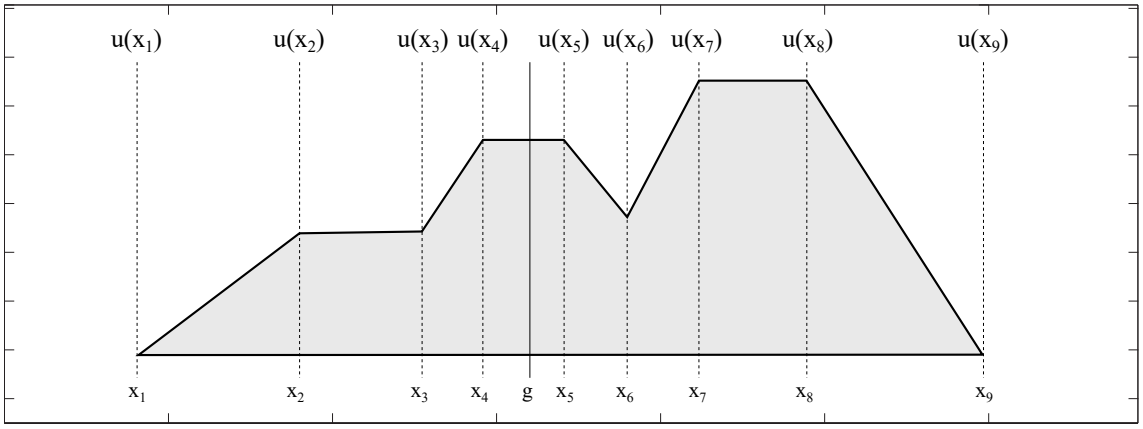

**Model 6.1:** Struktura FUZZY PID regulátoru.

Z hlediska optimalizace je celý proces nastavení rozdělen do tří způsobů výsledných postupů, na kterých je znázorněn vliv jednotlivých nastavení. První způsob nastavení je čistý fuzzy PID, kdy se nastavují jen vstupní a výstupní zesílení, při lineárním rozložení funkcí příslušnosti. První způsob by měl ukázat kvalitativně podobné výsledky jak předchozí čisté nastavení (PID) regulátoru. Druhý způsob je nastavení funkcí příslušnosti s konstantním vstupním a výstupním zesílením na hodnotu 1, Nastavení probíhá na Gaussových funkcích, kdy se nastavuje jejich rozptyl a jejich pozice v jednotlivých vstupech nebo výstupu. Poslední možností nastavení fuzzy regulátoru je kompletní nastavení, jak vstupních zesilovačů, tak funkcí příslušnosti v jedné optimalizaci. Tento způsob je nejvíce komplexní z hlediska nastavení regulátoru a má mnohem vetší stavový prostor řešení než předcházející PID regulátor. Kódové označení jednotlivých způsobu nastavení jsou tyto:

6

- Prosté nastavení fuzzy PID při lineárním rozložení funkcí příslušnosti jako **FPID**.
- Nastavení funkcí příslušnosti pro konstantní vstupní a výstupní nastavení jako **FP**.
- Komplexní nastavení fuzzy PID jak pro zesílení vstupů a výstupu, spolu s tvarem a pozicí funkcí příslušnosti jako **FPID-FP**.

$$g = \frac{\sum_{i=1}^{n} x_i u(x_i)}{\sum_{i=1}^{n} u(x_i)} \tag{6.1}$$




**Obrázek 6.1:** Výpočet defuzzyfikace pomoci funkce centroid.

| ⊞ | Bázové pravidlo | Váha |
|---|---|---|
| 1 | **IF** (P = P1) **AND** (I = I1) **AND** (D = D1) **THEN** (A = A1) | |
| 2 | **IF** (P = P1) **AND** (I = I1) **AND** (D = D2) **THEN** (A = A2) | |
| 3 | **IF** (P = P1) **AND** (I = I1) **AND** (D = D3) **THEN** (A = A3) | |
| 4 | **IF** (P = P1) **AND** (I = I2) **AND** (D = D1) **THEN** (A = A4) | |
| 5 | **IF** (P = P1) **AND** (I = I2) **AND** (D = D2) **THEN** (A = A5) | |
| 6 | **IF** (P = P1) **AND** (I = I2) **AND** (D = D3) **THEN** (A = A6) | |
| 7 | **IF** (P = P1) **AND** (I = I3) **AND** (D = D1) **THEN** (A = A7) | |
| 8 | **IF** (P = P1) **AND** (I = I3) **AND** (D = D2) **THEN** (A = A8) | |
| 9 | **IF** (P = P1) **AND** (I = I3) **AND** (D = D3) **THEN** (A = A9) | |
| 10 | **IF** (P = P2) **AND** (I = I1) **AND** (D = D1) **THEN** (A = A2) | |
| 11 | **IF** (P = P2) **AND** (I = I1) **AND** (D = D2) **THEN** (A = A3) | |
| 12 | **IF** (P = P2) **AND** (I = I1) **AND** (D = D3) **THEN** (A = A4) | |
| 13 | **IF** (P = P2) **AND** (I = I2) **AND** (D = D1) **THEN** (A = A5) | |
| 14 | **IF** (P = P2) **AND** (I = I2) **AND** (D = D2) **THEN** (A = A6) | 1 |
| 15 | **IF** (P = P2) **AND** (I = I2) **AND** (D = D3) **THEN** (A = A7) | |
| 16 | **IF** (P = P2) **AND** (I = I3) **AND** (D = D1) **THEN** (A = A8) | |
| 17 | **IF** (P = P2) **AND** (I = I3) **AND** (D = D2) **THEN** (A = A9) | |
| 18 | **IF** (P = P2) **AND** (I = I3) **AND** (D = D3) **THEN** (A = A10) | |
| 19 | **IF** (P = P3) **AND** (I = I1) **AND** (D = D1) **THEN** (A = A3) | |
| 20 | **IF** (P = P3) **AND** (I = I1) **AND** (D = D2) **THEN** (A = A4) | |
| 21 | **IF** (P = P3) **AND** (I = I1) **AND** (D = D3) **THEN** (A = A5) | |
| 22 | **IF** (P = P3) **AND** (I = I2) **AND** (D = D1) **THEN** (A = A6) | |
| 23 | **IF** (P = P3) **AND** (I = I2) **AND** (D = D2) **THEN** (A = A7) | |
| 24 | **IF** (P = P3) **AND** (I = I2) **AND** (D = D3) **THEN** (A = A8) | |
| 25 | **IF** (P = P3) **AND** (I = I3) **AND** (D = D1) **THEN** (A = A9) | |
| 26 | **IF** (P = P3) **AND** (I = I3) **AND** (D = D2) **THEN** (A = A10) | |
| 27 | **IF** (P = P3) **AND** (I = I3) **AND** (D = D3) **THEN** (A = A11) | |

**Tabulka 6.1:** Nastavení bázových pravidel.



Stejně jak přechozí část i zde je využitý kompletní způsob nastavení jen u prvního modelu (Model čtvrtého řádu), který představuje referenční bod k ostatním nastavením, kdy se používají dále jen komplexní FPID-FPnastavení fuzzy PID.

Fuzzy regulátor je velice komplexní regulátor, který nabízí množství druhů nastavení oproti klasickému (PID). Další mozností je nastavení počtu funkcí příslušnosti s kombinací s vice druhů funkcí nebo generovaní báze pravidel dle genetických algoritmů spolu s optimalizací váhových proměnných pro jednotlivá pravidla.

Obecně fuzzy nastavení je zejména vhodné pro hledání robustního kontroléru, v této oblasti fuzzy regulace vykazuje velice dobré výsledky.

## 6.1 Model čtvrtého řádu

Nastavení regulátoru fuzzy logiky pro model *system4* pomocí algoritmu (HC12). V této sekci je prezentováno nastavení fuzzy logiky pro optimalizaci v rámci nastavení funkcí příslušnosti. Konečné výsledky jsou demonstrovány na přechodových

funkcí a v grafech funkcí příslušnosti jak pro jednotlivé vstupy, tak v prostorové rovině. Při nastavení regulátoru v této kapitole se využívá struktury (PID) kontroléru, doplněné o fuzzy komponenty. Nejlepší hodnoty na tomto nastavení pro 100 spuštění algoritmu jsou 2.355, co představuje podobné ohodnocení jako (PID).

### 6.1.1 Nastavení dle algoritmu HC12

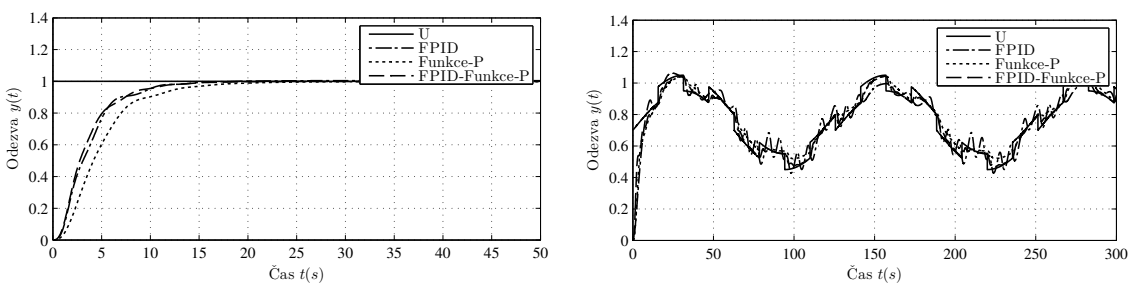
Sekce nastavení fuzzy regulátoru pro model *system4* dle algoritmu (HC12). Regulátor je nastaven dle fitness funkce *F111*. Fuzzy regulátor je optimalizovaný pro verze *FPID* s grafem funkcí příslušnosti (graf: 6.5), *FP* s grafem funkcí příslušnosti (graf: 6.6) a *FPID-FP* s grafem funkcí příslušnosti (graf: 6.7). Finální nastavení regulátoru je v tabulce (tab: 6.3) a ohodnocení přechodové funkce je v tabulce (tab: 6.2). Hodnoty přechodových funkcí jsou zobrazeny na skokové funkci (graf: 6.1a) a grafu SIN/PWM (graf: 6.1b) k zobrazení dynamických vlastností regulátoru.

| ⊞ | $T_r$ | $T_s$ | $Min_s$ | $Max_s$ | Over | Under | Peak | $T_p$ |
|---|---|---|---|---|---|---|---|---|
| Výs. [FPID] | 5.723 | 11.850 | 0.901 | 1.005 | 0.454 | | 1.005 | |
| Výs. [FP] | 8.118 | 16.917 | 0.900 | 0.998 | 0 | | 0.998 | 50 |
| Výs. [FPID-FP] | 6.291 | 12.218 | 0.902 | 1 | | | 1 | |

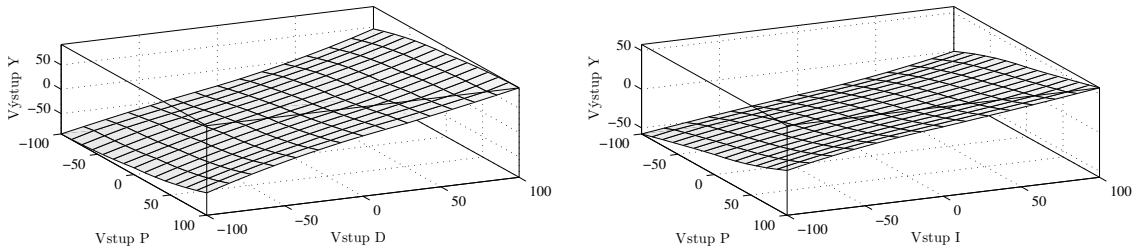
**Tabulka 6.2:** Tabulka ohodnocení odezvy na skokový signál dle nejlepších výsledků [Regulátor FUZZY Alg HC12] pro System4.

| ⊞ | P | I | D | Y |
|---|---|---|---|---|
| | MF | | Fit. | ITAE sin/pwm |
| Výs. [FPID] | 183.752 | 0.326 | 216.655 | 0.291 |
| | (graf: 6.2a), (graf: 6.2b) a (graf: 6.5) | | 2.796 | 312.367 |
| Výs. [FP] | 1 | | | |
| | (graf: 6.3a), (graf: 6.3b) a (graf: 6.6) | | 2.654 | 451.084 |
| Výs. [FPID-FP] | 216.766 | 1.159 | 235.008 | 0.159 |
| | (graf: 6.4a), (graf: 6.4b) a (graf: 6.7) | | 2.355 | 226.442 |
| P:(tab: 4.1) | B:<0.0,1000.0,20> | | | B:<0.0,1.0,10> |
| AI:(tab: 4.2) | M12 [Default], RUNS [100] | | | |

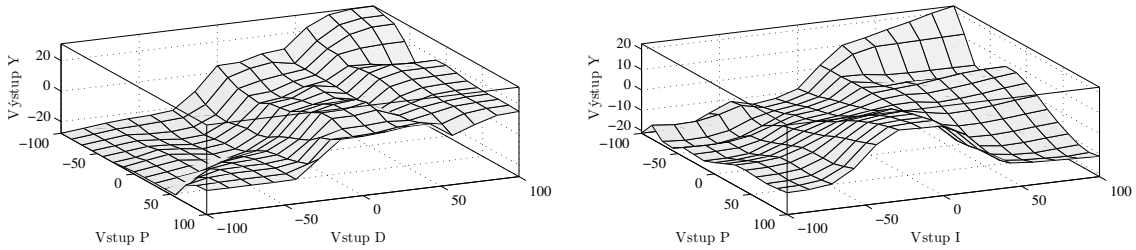
**Tabulka 6.3:** Tabulka nastavení regulátoru [Regulátor FUZZY Alg HC12] pro System4.



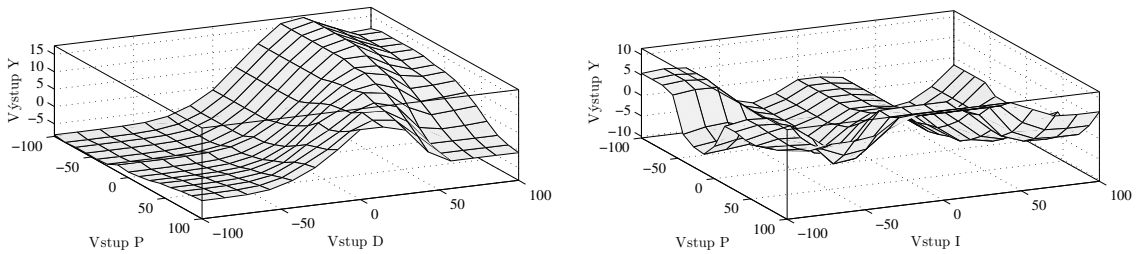

**Graf 6.1:** Pořadí grafů: (a: Průběh přechodové charakteristiky na skokovém signálu dle [Regulátor FUZZY Alg HC12] pro System4.) a (b: Průběh přechodové charakteristiky na SIN/PWM signálu dle [Regulátor FUZZY Alg HC12] pro System4.).




**Graf 6.2:** Plocha funkcí příslušnosti pro všechny vstupy [Typ FPID Alg HC12] pro System4 Pořadí grafů: (a: Plocha vstupů P, D ku Y.) a (b: Plocha vstupů P, I ku Y.).

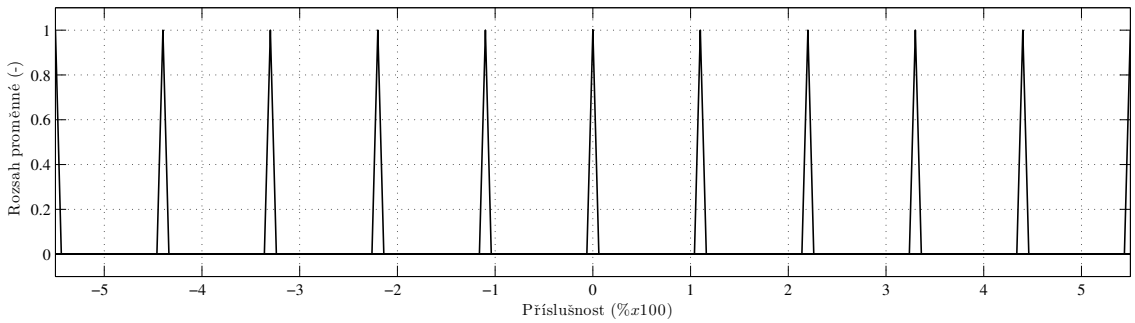



**Graf 6.3:** Plocha funkcí příslušnosti pro všechny vstupy [Typ FP Alg HC12] pro System4 Pořadí grafů: (a: Plocha vstupů P, D ku Y.) a (b: Plocha vstupů P, I ku Y.).

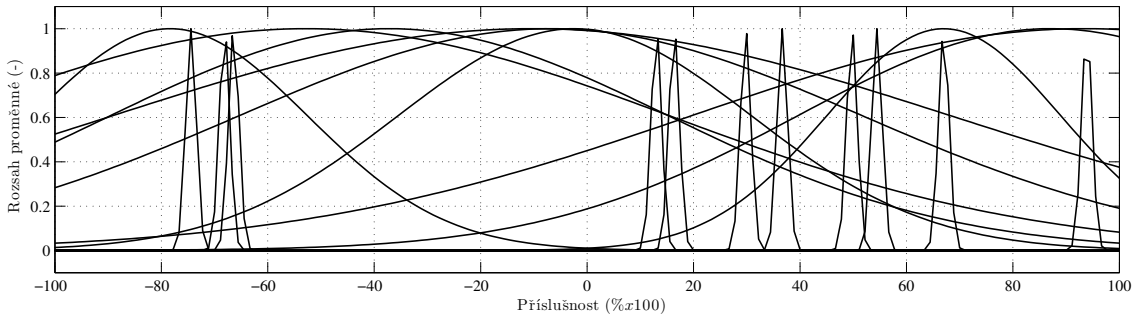



**Graf 6.4:** Plocha funkcí příslušnosti pro všechny vstupy [Typ FPID-FP Alg HC12] pro System4 Pořadí grafů: (a: Plocha vstupů P, D ku Y.) a (b: Plocha vstupů P, I ku Y.).

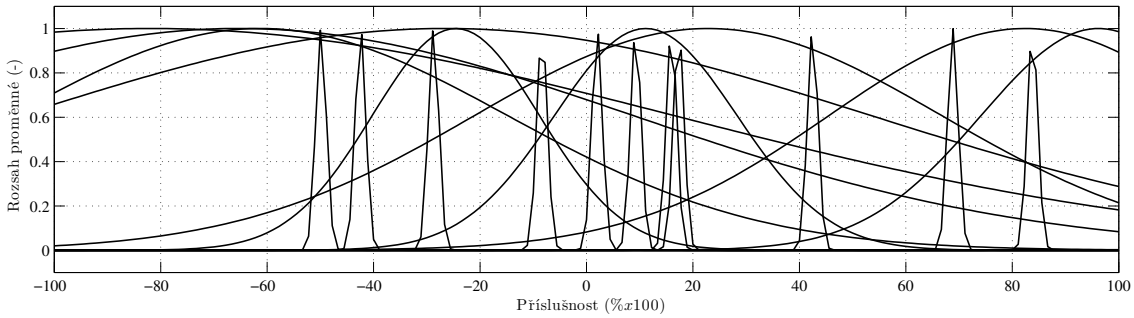





**Graf 6.5:** Funkce příslušnosti všech vstupů a výstupu [Typ FPID Alg HC12] pro System4.




**Graf 6.6:** Funkce příslušnosti všech vstupů a výstupu [Typ FP Alg HC12] pro System4.




**Graf 6.7:** Funkce příslušnosti všech vstupů a výstupu [Typ FPID-FP Alg HC12] pro System4.

## 6.2 Model čtvrtého řádu s nelineárním prvkem

Nastavení regulátoru fuzzy logiky pro model *system4delay* pomocí algoritmu (HC12). V této sekci je prezentováno nastavení fuzzy logiky pro optimalizaci v rámci nastavení funkcí příslušnosti. Konečné výsledky jsou demonstrovány na přechodových funkcí a v grafech funkcí příslušnosti, jak pro jednotlivé vstupy, tak v prostorové rovině. Na rozdíl od předešlé sekce se kapitola už soustřeďuje jen na nastavení regulátoru pomocí algoritmu (HC12). Při nastavení regulátoru v této kapitole se využívá struktury (PID) kontroléru, doplněné o fuzzy komponenty, proto mužem předešlé výsledky porovnat s nejlepším nastavením regulátoru typu *FPID-FP*. Nejlepší hodnoty na tomto nastavení pro 100 spuštění algoritmu jsou 3.986, co představuje přibližně podobné ohodnocení s porovnáním ke klasickému nastavení dle (PID). Celkový popis modelu je v sekci (sekce: 4.2.2 - viz str. 49). Podrobné srovnání výsledků pro jednotlivé regulátory a pro model čtvrtého řádu s nelineárním prvkemje prezentováno v sekci (sekce: 8.2 - viz str. 110).



### 6.2.1 Nastavení dle algoritmu HC12

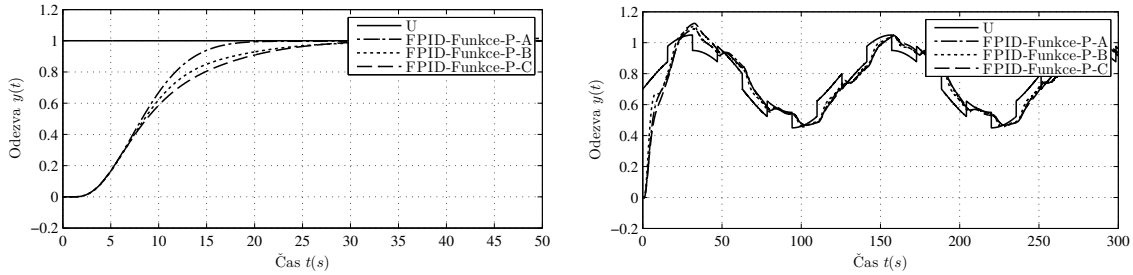
Sekce nastavení fuzzy regulátoru pro model *system4delay* dle algoritmu (HC12). Regulátor je nastaven dle fitness funkce *F111*. Fuzzy regulátor je optimalizovaný pro verze *FPID-FP A* s grafem funkcí příslušnosti (graf: 6.10), *FPID-FP B* s grafem funkcí příslušnosti (graf: 6.11) a *FPID-FP C* s grafem funkcí příslušnosti (graf: 6.12). Finální nastavení regulátoru je v tabulce (tab: 6.5) a ohodnocení přechodové funkce je v tabulce (tab: 6.4). Hodnoty přechodových funkcí jsou zobrazeny na skokové funkci (graf: 6.8a) a grafu SIN/PWM (graf: 6.8b) k zobrazení dynamických vlastností regulátoru.

| ⊞ | $T_r$ | $T_s$ | $Min_s$ | $Max_s$ | Over | Under | Peak | $T_p$ |
|---|---|---|---|---|---|---|---|---|
| Výs. [FPID-FP-A] | 9.551 | 17.493 | 0.902 | 0.999 | 0 | 0.010 | 0.999 | 50 |
| Výs. [FPID-FP-B] | 13.463 | 27.921 | 0.900 | | | 0.009 | | |
| Výs. [FPID-FP-C] | 15.252 | 27.713 | | 1.015 | 1.525 | 0.010 | 1.015 | |

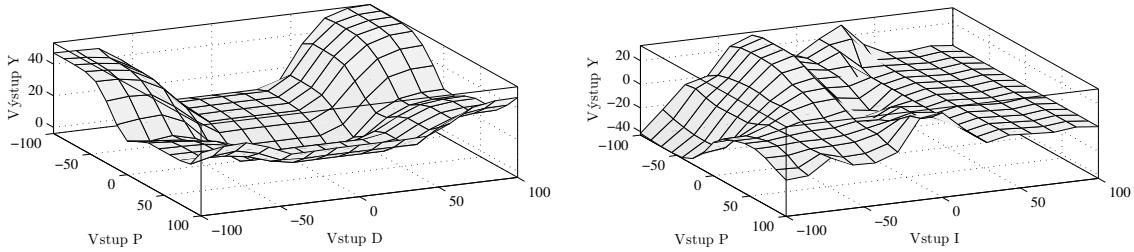
**Tabulka 6.4:** Tabulka ohodnocení odezvy na skokový signál dle nejlepších výsledků [Regulátor FUZZY Alg HC12] pro System4Delay.

| ⊞ | P | I | D | Y |
|---|---|---|---|---|
| | MF | | Fit. | ITAE sin/pwm |
| Výs. [FPID-FP-A] | 128.953 | 998.604 | 192.499 | 0.005 |
| | (graf: 6.9a), (graf: 6.9b) a (graf: 6.10) | | 3.442 | 922.070 |
| Výs. [FPID-FP-B] | 131.294 | 859.674 | 196.225 | 0.005 |
| | (graf: 6.11) | | 3.617 | 1380.050 |
| Výs. [FPID-FP-C] | 135.823 | 941.523 | 204.140 | 0.005 |
| | (graf: 6.12) | | 3.986 | 1612.325 |
| P:(tab: 4.1) | B:<0.0,1000.0,20> | | | B:<0.0,1.0,10> |
| AI:(tab: 4.2) | M12 [Default], RUNS [100] | | | |

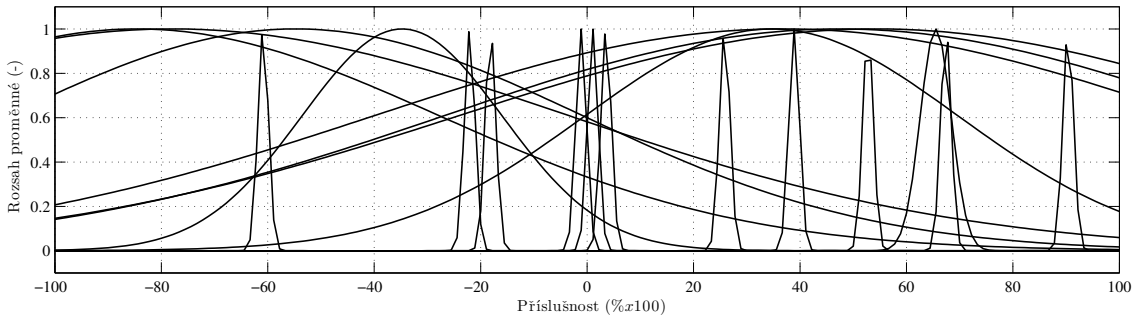
**Tabulka 6.5:** Tabulka nastavení regulátoru [Regulátor FUZZY Alg HC12] pro System4Delay.



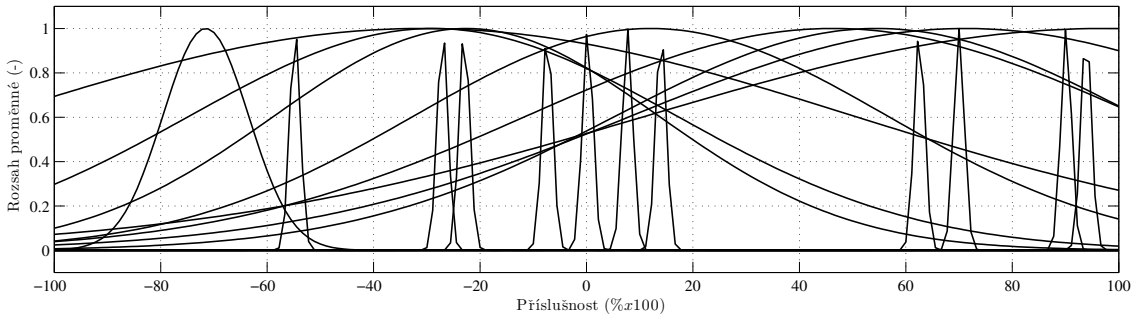

**Graf 6.8:** Pořadí grafů: (a: Průběh přechodové charakteristiky na skokovém signálu dle [Regulátor FUZZY Alg HC12] pro System4Delay.) a (b: Průběh přechodové charakteristiky na SIN/PWM signálu dle [Regulátor FUZZY Alg HC12] pro System4Delay.).



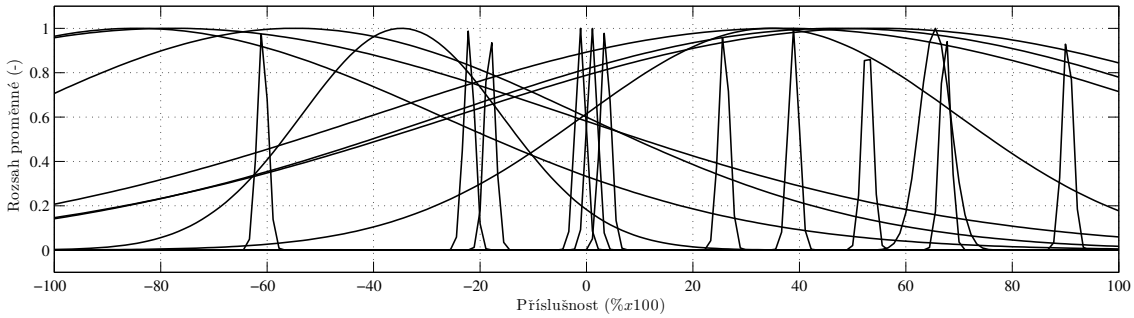

**Graf 6.9:** Plocha funkcí příslušnosti pro všechny vstupy [Typ FPID-FP-A Alg HC12] pro System4Delay Pořadí grafů: (a: Plocha vstupů P, D ku Y.) a (b: Plocha vstupů P, I ku Y.).




**Graf 6.10:** Funkce příslušnosti všech vstupů a výstupu [Typ FPID-FP-A Alg HC12] pro System4Delay.




**Graf 6.11:** Funkce příslušnosti všech vstupů a výstupu [Typ FPID-FP-B Alg HC12] pro System4Delay.




**Graf 6.12:** Funkce příslušnosti všech vstupů a výstupu [Typ FPID-FP-C Alg HC12] pro System4Delay.

## 6.3 Model kuličky v obruči

6

Nastavení regulátoru fuzzy logiky pro model *ballandloop* pomocí algoritmu (HC12). V této sekci je prezentováno nastavení fuzzy logiky pro optimalizaci v rámci nastavení funkcí příslušnosti. Konečné výsledky jsou demonstrovány na přechodových funkcí a v grafech funkci příslušnosti, jak pro jednotlivé vstupy, tak v prostorové rovině. Na rozdíl od předešlé sekce se kapitola už soustřeďuje jen na nastavení regulátoru pomocí algoritmu (HC12). Při nastavení regulátoru v této kapitole se využívá struktury (PID) kontroléru, doplněné o fuzzy komponenty, proto mužem předešlé výsledky porovnat s nejlepším nastavením regulátoru typu *FPID-FP*. Nejlepší hodnoty na tomto nastavení pro 100 spuštění algoritmu jsou 2.224, co představuje přibližně podobné ohodnocení s porovnáním ke klasickému nastavení dle (PID) zlepšené o přibližně o 40 %, co vypovídá o lepších dynamických vlastnostech fuzzy regulátoru. Celkový popis modelu je v sekci (sekce: 4.2.3 - viz str. 51). Podrobné srovnání výsledků pro jednotlivé regulátory a pro (model kuličky v obruči) je prezentováno v sekci (sekce: 8.3 - viz str. 111).

### 6.3.1 Nastavení dle algoritmu HC12

Sekce nastavení fuzzy regulátoru pro model *system4delay* dle algoritmu (HC12). Regulátor je nastaven dle fitness funkce *F111*. Fuzzy regulátor je optimalizovaný pro verze *FPID-FP A* s grafem funkcí příslušnosti (graf: 6.15), *FPID-FP B* s grafem funkcí příslušnosti (graf: 6.16) a *FPID-FP C* s grafem funkcí příslušnosti (graf: 6.17).

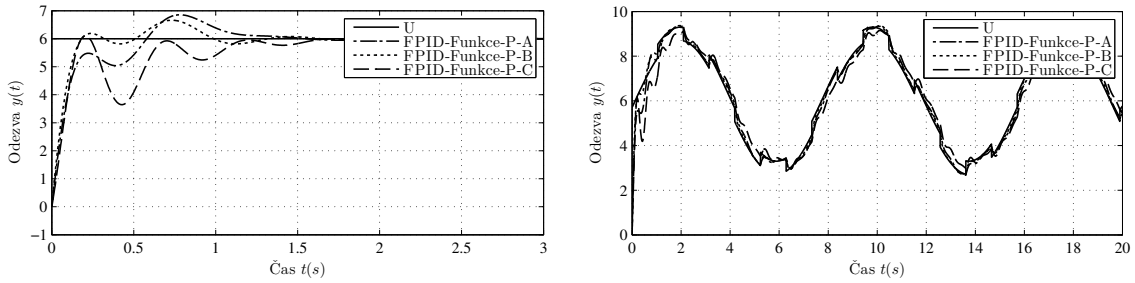
Finální nastavení regulátoru je v tabulce (tab: 6.7) a ohodnocení přechodové funkce je v tabulce (tab: 6.6). Hodnoty přechodových funkcí jsou zobrazeny na skokové funkci (graf: 6.13a) a grafu SIN/PWM (graf: 6.13b) k zobrazení dynamických vlastností regulátoru.

| ⊞ | $T_r$ | $T_s$ | $Min_s$ | $Max_s$ | Over | Under | Peak | $T_p$ |
|---|---|---|---|---|---|---|---|---|
| Výs. [FPID-FP-A] | 0.177 | 1.205 | 5.039 | 6.855 | 14.254 | 0 | 6.855 | 0.783 |
| Výs. [FPID-FP-B] | 0.126 | 1.219 | 5.437 | 6.661 | 11.024 | | 6.661 | 0.733 |
| Výs. [FPID-FP-C] | 0.127 | 1.543 | 3.646 | 6.109 | 1.809 | | 6.109 | 0.205 |

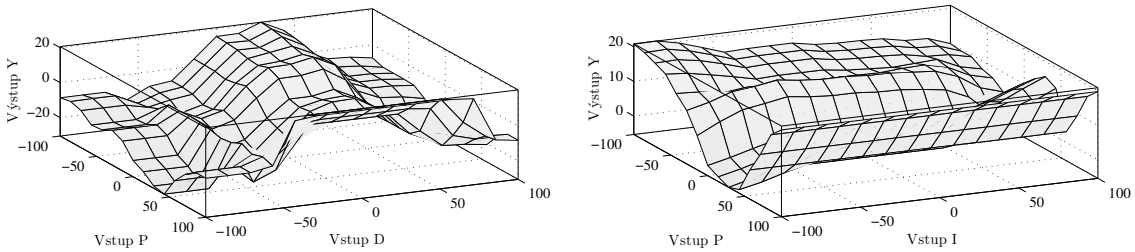
**Tabulka 6.6:** Tabulka ohodnocení odezvy na skokový signál dle nejlepších výsledků [Regulátor FUZZY Alg HC12] pro BallAndLoop.

| ⊞ | P | I | D | Y |
|---|---|---|---|---|
| | MF | | Fit. | ITAE sin/pwm |
| Výs. [FPID-FP-A] | 165.688 | 3.602 | 2.168 | 0.986 |
| | (graf: 6.14a), (graf: 6.14b) a (graf: 6.15) | | 3.644 | 183.737 |
| Výs. [FPID-FP-B] | 186.272 | 17.497 | 2.269 | 0.980 |
| | (graf: 6.16) | | 4.038 | 109.225 |
| Výs. [FPID-FP-C] | 169.961 | 51.283 | 2.224 | 0.993 |
| | (graf: 6.17) | | 4.396 | 239.551 |
| P:(tab: 4.1) | B:<0.0,1000.0,20> | | | B:<0.0,1.0,10> |
| AI:(tab: 4.2) | M12 [Default], RUNS [100] | | | |

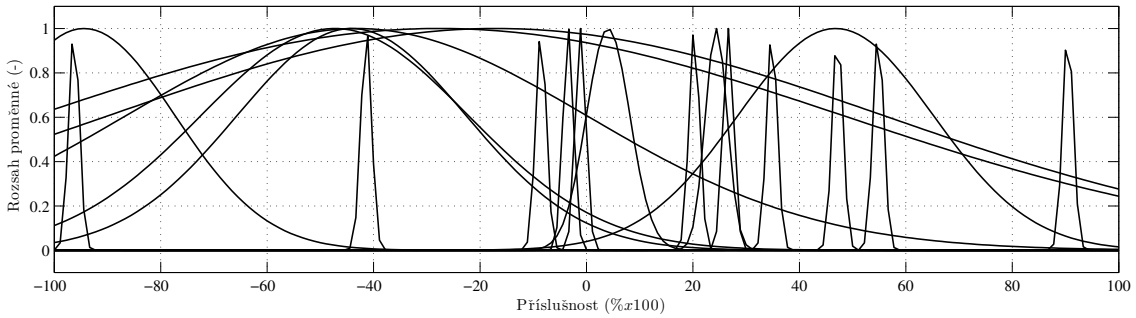
**Tabulka 6.7:** Tabulka nastavení regulátoru [Regulátor FUZZY Alg HC12] pro BallAndLoop.



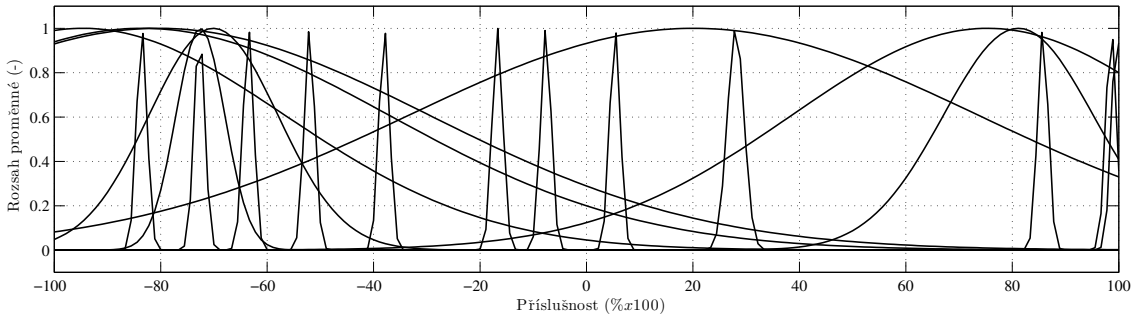

**Graf 6.13:** Pořadí grafů: (a: Průběh přechodové charakteristiky na skokovém signálu dle [Regulátor FUZZY Alg HC12] pro BallAndLoop.) a (b: Průběh přechodové charakteristiky na SIN/PWM signálu dle [Regulátor FUZZY Alg HC12] pro BallAndLoop.).



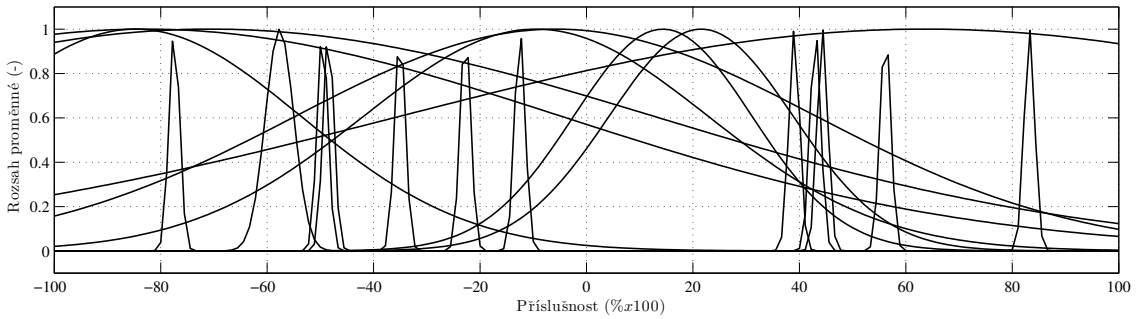

**Graf 6.14:** Plocha funkcí příslušnosti pro všechny vstupy [Typ FPID-FP-A Alg HC12] pro BallAndLoop Pořadí grafů: (a: Plocha vstupů P, D ku Y.) a (b: Plocha vstupů P, I ku Y.).




**Graf 6.15:** Funkce příslušnosti všech vstupů a výstupu [Typ FPID-FP-A Alg HC12] pro BallAndLoop.




**Graf 6.16:** Funkce příslušnosti všech vstupů a výstupu [Typ FPID-FP-B Alg HC12] pro BallAndLoop.




**Graf 6.17:** Funkce příslušnosti všech vstupů a výstupu [Typ FPID-FP-C Alg HC12] pro BallAndLoop.



## 6.4 Model magnetické levitace

Nastavení regulátoru fuzzy logiky pro model *maglev* pomocí algoritmu (HC12). V této sekci je prezentováno nastavení fuzzy logiky pro optimalizaci v rámci nastavení funkcí příslušnosti. Konečné výsledky jsou demonstrovány na přechodových funkcí a v grafech funkci příslušnosti jak pro jednotlivé vstupy, tak v prostorové rovině. Na rozdíl od předešlé sekce se kapitola už soustřeďuje jen na nastavení regulátoru pomocí algoritmu (HC12). Při nastavení regulátoru v této kapitole se využívá struktury (PID) kontroléru, doplněné o fuzzy komponenty, proto mužem předešlé výsledky porovnat s nejlepším nastavením regulátoru typu *FPID-FP*. Nejlepší hodnoty na tomto nastavení pro 100 spuštění algoritmu jsou 2.225, co představuje přibližně podobné ohodnocení s porovnáním ke klasickému nastavení dle (PID). Celkový popis modelu je v sekci (sekce: 4.2.4 - viz str. 54).

### 6.4.1 Nastavení dle algoritmu HC12

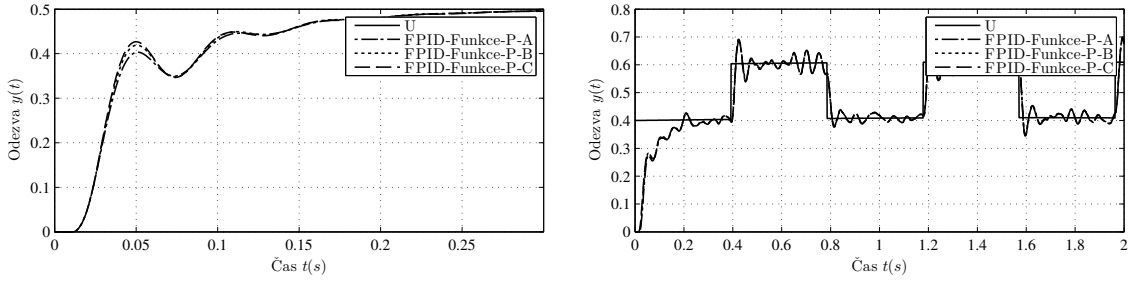
Sekce nastavení fuzzy regulátoru pro model *system4delay* dle algoritmu (HC12). Regulátor je nastaven dle fitness funkce *F111*. Fuzzy regulátor je optimalizovaný pro verze *FPID-FP A* s grafem funkcí příslušnosti (graf: 6.20), *FPID-FP B*, (graf: 6.21) a *FPID-FP C* a (graf: 6.22). Finální nastavení regulátoru je v tabulce (tab: 6.9) a ohodnocení přechodové funkce je v tabulce (tab: 6.8). Hodnoty přechodových funkcí jsou zobrazeny na skokové funkci (graf: 6.18a) a grafu SIN/PWM (graf: 6.18b) k zobrazení dynamických vlastností regulátoru. Jako u předešlé kapitoly jsou výsledné nastavení regulátoru porovnány na reálné soustavě, konkrétně pro nejlepší nastaveni regulátoru *FPID-FP* B. Měření na skutečné soustavě prezentuje použitelnost regulátoru FPID a jeho možnosti při regulaci vysoce nelineárních systémů. Hodnoty měření prezentuje graf (graf: 6.23). Výhodou tohoto typu regulátoru oproti předešlému nastavení, je lepší adaptivnost na změnu požadované polohy a variabilita při nastavování hodnot regulátoru.

| ⊞ | $T_r$ | $T_s$ | $Min_s$ | $Max_s$ | Over | Under | Peak | $T_p$ |
|---|---|---|---|---|---|---|---|---|
| Výs. [FPID-FP-A] | 0.121 | 0.239 | 0.450 | 0.496 | 0 | | 0.496 | 0.300 |
| Výs. [FPID-FP-B] | | 0.240 | 0.451 | | | | | |
| Výs. [FPID-FP-C] | | 0.247 | | 0.495 | | | 0.495 | |

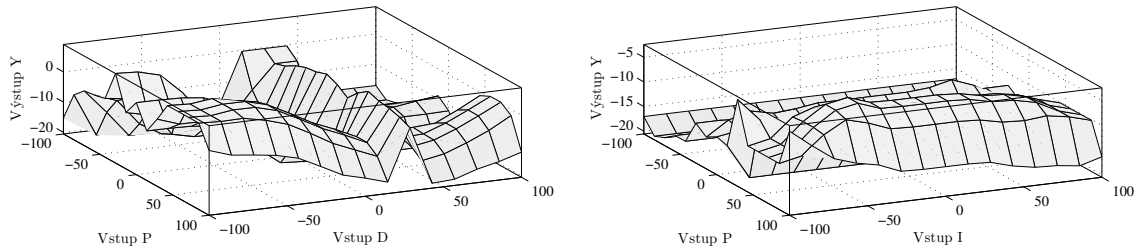
**Tabulka 6.8:** Tabulka ohodnocení odezvy na skokový signál dle nejlepších výsledků [Regulátor FUZZY Alg HC12] pro Maglev.

| ⊞ | P | I | D | Y |
|---|---|---|---|---|
| | MF | | Fit. | ITAE sin/pwm |
| Výs. [FPID-FP-A] | 571.723 | 831.067 | 0.469 | 0.724 |
| | (graf: 6.19a), (graf: 6.19b) a (graf: 6.20) | | 2.087 | 1.540 |
| Výs. [FPID-FP-B] | 571.723 | 831.067 | 0.469 | 0.724 |
| | (graf: 6.21) | | 2.222 | 1.502 |
| Výs. [FPID-FP-C] | 571.723 | 831.067 | 0.469 | 0.724 |
| | (graf: 6.22) | | 2.225 | 1.513 |
| P:(tab: 4.1) | B:<0.0,1000.0,20> | | | B:<0.0,1.0,10> |
| AI:(tab: 4.2) | M12 [Default], RUNS [100] | | | |

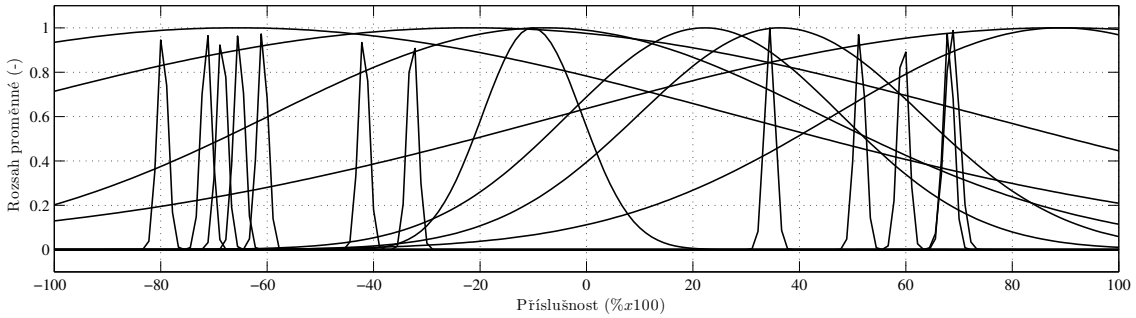
**Tabulka 6.9:** Tabulka nastavení regulátoru [Regulátor FUZZY Alg HC12] pro Maglev.



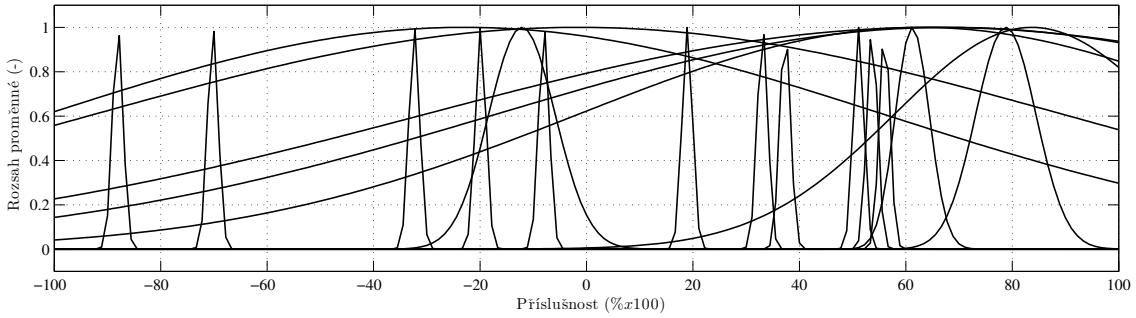

**Graf 6.18:** Pořadí grafů: (a: Průběh přechodové charakteristiky na skokovém signálu dle [Regulátor FUZZY Alg HC12] pro Maglev.) a (b: Průběh přechodové charakteristiky na SIN/PWM signálu dle [Regulátor FUZZY Alg HC12] pro Maglev.).



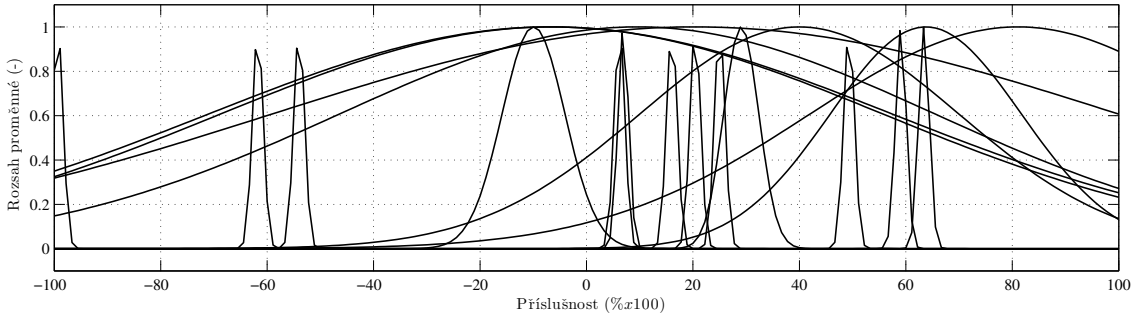

**Graf 6.19:** Plocha funkcí příslušnosti pro všechny vstupy [Typ FPID-FP-A Alg HC12] pro Maglev Pořadí grafů: (a: Plocha vstupů P, D ku Y.) a (b: Plocha vstupů P, I ku Y.).



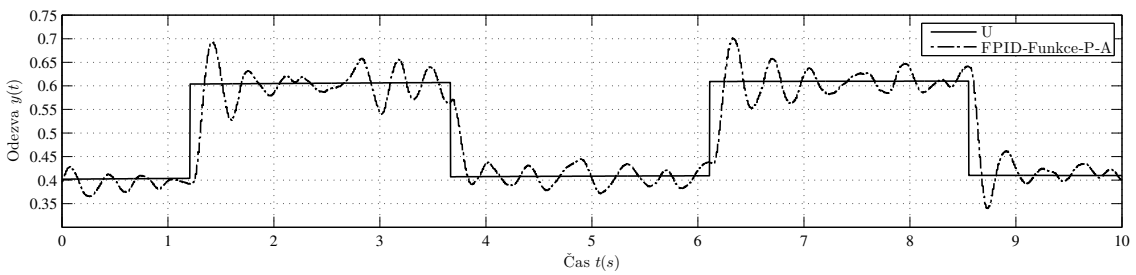

**Graf 6.20:** Funkce příslušnosti všech vstupů a výstupu [Typ FPID-FP-A Alg HC12] pro Maglev.



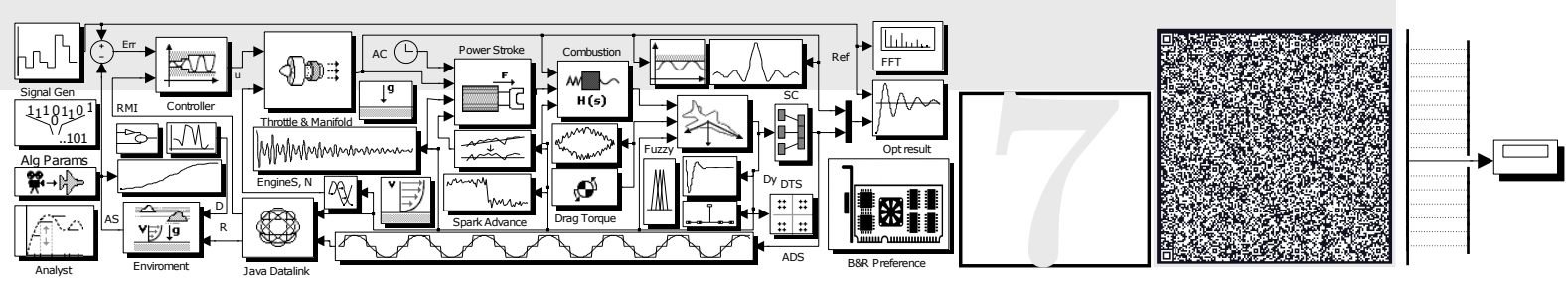

**Graf 6.21:** Funkce příslušnosti všech vstupů a výstupu [Typ FPID-FP-B Alg HC12] pro Maglev.




**Graf 6.22:** Funkce příslušnosti všech vstupů a výstupu [Typ FPID-FP-C Alg HC12] pro Maglev.




**Graf 6.23:** Průběh přechodové charakteristiky na reálné soustavě pro Maglev.




# KAPITOLA 7

# GENEROVANÍ STRUKTURY REGULÁTORU

Poslední příklad v regulaci systémů je regulace dle vytvořených regulačních zákonů pomocí gramatické evoluce s postupným doladěním v (HC12) algoritmu. Generovaní regulačních zákonů dává největší svobodu při navrhovaní výsledného regulátoru a závisí na kvalitě používané gramatiky. Z hlediska tohoto přikladu se používá Backus-Naurova forma gramatiky (sekce: 2.4 - viz str. 25) ve dvou základních verzích, první verze je pro systémy (model čtvrtého řádu, model čtvrtého řádu s nelineárním prvkem, model kuličky v obruči) a je založena na generovaní pravidel v Matlab syntaxi, druhá verze je pro (model magnetické levitace) a je založena na generovaní simulinkovských schémat.

Dvě verze gramatik se pozívají z důsledku prezentace, kdy oba dva druhy je možné úspěšně použít k řešení problému a z důsledku nutnosti v modelu *maglev* použít regulátor v simulinkovské syntaxi. Kvalitativně oba dva způsoby jsou na tom podobně, kdy oba způsoby dokáži najit optimalizovaný regulátor, jen zápis v Matlab syntaxi je rychlejší na vypočet regulace, a proto je vhodnější pro běžné použití.



Matlab verze gramatického zápisu je znázorněna na rovnici (gramatika: 7.2) a hlavním prvkem gramatiky je tvorba obecných přechodových funkcí pomocí zápisu rovnice (vz: 1.2). Kvůli možnosti generovaní až dokonalých regulátorů je nutno zavést omezující podmínky pro

```
        N = { init, node, block_input_1, block_input_2, inputs, consts }
        T = { +, -, *, /, input_1, integrator, derivator, 1 - 9 } | S = { init }
<init>           ::(<node><block_input_1>) | (<node><node><block_input_2>)
<node>           ::(<init>) | (<inputs>) | (<consts>)
<block_input_1>  ::integrator | derivator
<block_input_2>  ::+ | - | / | *
<inputs>         ::input_1
<consts>         ::1 | ...| 9
```

**Gramatika 7.1:** Obecná gramatika k regulaci systémů v prostředí Simulink.

výsledný regulátor. Hlavní podmínkou je vygenerovaní fyzikálně realizovatelného systému, tedy výsledný systém bude obsahovat více pólů než nul ($m \leq n$) v rovnici přenosu. Další podmínkou je vygenerovaní reálného regulátoru z omezení maximálního výstupních i vstupních koeficientů diferenciální rovnice na hodnotu 100.

Gramatika popsaná bloky v Simulink syntaxi je založená na generovaní klasického regulátoru s integrační a derivační složkou spolu s obecnými matematickými operacemi. Omezení pro tento typ regulace je stejný jak pro verzi v Matlab zápisu. Celková gramatika je zobrazena pomocí rovnic (gramatika: 7.1).

Poslední části této kapitoly je ladění výsledných regulačních zákonů dle algoritmu (HC12). Ladění je prováděno v důsledku, kdy gramatická evoluce dokáže najit použitelný regulační zákon, ale jemné doladění všech koeficientů je vhodné využít odlišný algoritmus k numerické optimalizaci. Pro Matlab verzi gramatiky se dolaďují výsledné póly a nuly přechodové rovnice dle zadaného rozsahu a pro Simulink verzi je ladění prováděno na vytvořených zesilovačích vstupního signálu. Každý model této kapitoly má zobrazený, jak výraz po gramatické evoluci bez uprav, tak výraz, který je použít v algoritmu (HC12) na finální optimalizaci.

Oproti předešlým kapitolám se v této kapitole používají všechny způsoby regulace pro všechny modely, je to hlavně kvůli rozmanitosti generovaných struktur a výsledné dolaďovací optimalizaci. Kvůli zprůhlednění výsledků a zachovaní jmenné konvenci se pozívají kódové označeni jednotlivých způsobů a jsou to tyto:



- Generovaní regulačních zákonů pomocí gramatické evoluce jako **Version**.
- Doladění finálních koeficientů vygenerovaného regulačního zákona jako **Tuning**.

Generovaní vlastních regulačních pravidel obecně dokáže najít velmi kvalitní systém k regulaci, který v mnohém předčí standardní regulátory jako je třeba (PID). Tento systém má, ale nevýhodu vysoké výpočetní náročnosti a je tedy vhodný jen v podmínkách kdy chceme najit specifický regulátor s přesně definovanými podmínkami. Takové podmínky můžou být, jako v této kapitole, fyzikální realizovatelnost, maximální koeficienty rovnice anebo třeba jen databanka možných součástek, které mužem použít při realizovaní regulátoru. Tento typ nastavení tedy dává velice velké moznosti při nastavení výsledné regulace. Hlavně ve spojitosti s doladěním výsledné struktury dle optimalizačního algoritmu (HC12), s tímto finálním procesem se zlepši kvalita regulátoru na úroveň velmi robustního řízení.

```
        N = { expr, expr2, op, tr_func, vector, var }
        T = { +, -, *, /, X, -1, 0, 1 - 9 } | S = { expr }
<expr>   ::(<tr_func><op><tr_func>) | (<var><op><tr_func>)
<expr2>  ::(<expr2> <op> <expr2>) | <var> | (<var> <op> <var>) | <vector>
<op>     ::+ | - | / | *
<tr_func>::tf(<vector>, <vector>) | tf(<vector>, <vector>)<op><expr>
<vector> ::[<expr2>, <expr2>]
<var>    ::-1 | 0 | 1 | ...| 9
```

**Gramatika 7.2:** Obecná gramatika k regulaci systémů v prostředí Matlab.

# 7.1 Model čtvrtého řádu

Nastavení struktury regulátoru dle (GE) s gramatikou (gramatika: 7.2) pro model *system4* a doladění výsledné struktury pomocí (HC12). Výsledky jsou vyhodnoceny na výsledné přechodové funkci při jednotkovém vstupu a konečné výsledky jsou prezentovány, jak v neupravené formě po dokončení vypočtu gramatické evoluce, tak v upravené formě přichystané na doladění dle algoritmu (HC12) spolu s nastavením pro jednotlivé algoritmy. Výsledné nejlepší regulátory jsou porovnány na zaklade Root-Locus charakteristiky (graf: 7.2a) a (graf: 7.2b).

## 7.1.1 Nastavení dle algoritmu GE

Generovaní regulačních zákonů v prostředí Matlab pro model *system4* dle algoritmu (GE). Regulátor je nastaven dle fitness funkce *F111*. Konečné výsledky jsou v tabulce (tab: 7.2) spolu s nastavením optimalizačního algoritmu. Výsledek gramatické evoluce je prezentován vždy ve dvou verzích, první je neupravený výsledek dle aktivní gramatiky a druha je verze po úpravě k dalšímu doladění vybraných parametrů $param_x$. Ohodnocení přechodové funkce je v tabulce (tab: 7.1) a výsledné hodnoty přechodových funkcí jsou zobrazeny na skokovém grafu (graf: 7.1a) a grafu SIN/PWM (graf: 7.1b) k zobrazení dynamických vlastností regulátoru.



| ⊞ | $T_r$ | $T_s$ | $Min_s$ | $Max_s$ | Over | Under | Peak | $T_p$ |
|---|---|---|---|---|---|---|---|---|
| Výs. [Version-A] | 4.865 | 11.625 | 0.901 | 1.027 | 2.740 | 0 | 1.027 | 9.950 |
| Výs. [Version-B] | 5.345 | 14.120 | 0.900 | 1.042 | 4.162 | | 1.042 | 10.950 |
| Výs. [Version-C] | 4.465 | 26.024 | 0.901 | 1.125 | 12.502 | | 1.125 | 11.150 |

**Tabulka 7.1:** Tabulka ohodnocení odezvy na skokový signál dle nejlepších výsledků [Regulátor Vlastni-struktura Alg GE] pro System4.

| ⊞ | Výsledek | Fit. | ITAE sin/pwm |
|---|---|---|---|
| Výs. [Version-A] | GE výsledek (expr: 7.1) a (vz: 7.1) | 2.577 | 236.611 |
| Výs. [Version-B] | GE výsledek (expr: 7.2) a (vz: 7.2) | 3.108 | 257.197 |
| Výs. [Version-C] | GE výsledek (expr: 7.3) a (vz: 7.3) | 3.440 | 692.258 |
| P:(tab: 4.1) | C:<40,8> | ⊡ | |
| AI:(tab: 4.2) | NP [500], F [0.2], R [0.5], PC [2], W [10], GEN [200], RUNS [100] | | |

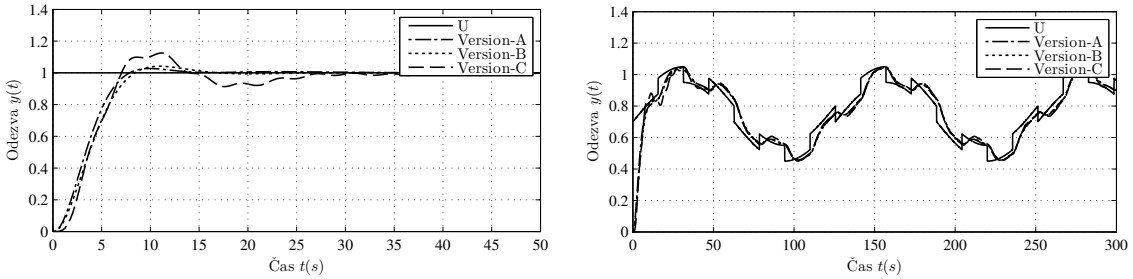
**Tabulka 7.2:** Tabulka nastavení regulátoru [Regulátor Vlastni-struktura Alg GE] pro System4.

```
expression = (tf([(8-2),[0,((1/5)-((8/2)/(7-6)))]],[[((2/8)*((7+4)-9)),4],0])+
tf([(5-3),((8/2)/(7-6))],[[((2/8)*((7+4)-9)),4],0]));
```

**GE výsledek 7.1:** Vypočtený výraz dle GE gramatiky [Typ Version-A] pro System4.

```
expression = (tf([(1+5),(-1/-1)],[5,(0*5)])/tf([(2/2),9],[9,2]));
```

**GE výsledek 7.2:** Vypočtený výraz dle GE gramatiky [Typ Version-B] pro System4.



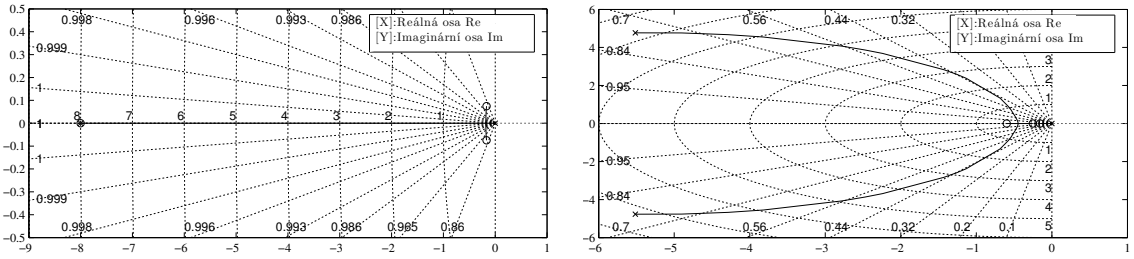

**Graf 7.1:** Pořadí grafů: (a: Průběh přechodové charakteristiky na skokovém signálu dle [Regulátor Vlastni-struktura Alg GE] pro System4.) a (b: Průběh přechodové charakteristiky na SIN/PWM signálu dle [Regulátor Vlastni-struktura Alg GE] pro System4.).

$$\begin{aligned} Gr_{GE}(s) &= \frac{3s^4 + 25s^3 + 8.1s^2 + 0.8s^1}{0.25s^4 + 4s^3 + 16s^2} \\ Gr_{HC12}(s) &= \frac{Param1s^4 + Param2s^3 + Param3s^2 + Param4s^1}{Param5s^4 + Param6s^3 + Param7s^2} \end{aligned} \tag{7.1}$$

$$\begin{aligned} Gr_{GE}(s) &= \frac{54s^2 + 21s^1 + 2}{5s^2 + 45s^1} \\ Gr_{HC12}(s) &= \frac{Param1s^2 + Param2s^1 + Param3}{Param4s^2 + Param5s^1} \end{aligned} \tag{7.2}$$

$$\begin{aligned} Gr_{GE}(s) &= \frac{9s^2 + 6s^1 + 0.42857}{4s^3 + 0.5s^2 + 11s^1} \\ Gr_{HC12}(s) &= \frac{Param1s^2 + Param2s^1 + Param3}{Param4s^3 + Param5s^2 + Param6s^1} \end{aligned} \tag{7.3}$$

7




**Graf 7.2:** Pořadí grafů: (a: Root-Locus charakteristika pro regulátor Version-A System4.) a (b: Root-Locus charakteristika pro regulátor Tuning-A System4.).

Root-Locus charakteristika znázorňuje pozici nul a pólů v jednotlivých systémech. Pro každý systém v této kapitole je prezentovaná Root-Locus analýza pro nejlepší regulátor z prvotního generovaní struktury a nejlepší regulátor z doladění parametrů.

```
expression = (3/tf([[[4,(2/4)],(2+9)],(0*9)],[3,[2,(1/7)]]));
```

**GE výsledek 7.3:** Vypočtený výraz dle GE gramatiky [Typ Version-C] pro System4.

### 7.1.2 Nastavení dle algoritmu HC12

Ladění konečných výsledků gramatické evoluce pro model *system4* dle algoritmu (HC12). Regulátor je nastaven dle fitness funkce *F111*. Konečné výsledky jsou v tabulce (tab: 7.4) spolu s nastavením optimalizačního algoritmu. Optimalizované struktury regulátoru jsou výsledky gramatické evoluce z předešlé části a používá se upravená forma generovaného výsledku, která obsahuje optimalizované parametry $param_x$. Ohodnocení přechodové funkce je v tabulce (tab: 7.3) a výsledné hodnoty přechodových funkcí jsou zobrazeny na skokové funkci (graf: 7.3a) a grafu SIN/PWM (graf: 7.3b) k zobrazení dynamických vlastností regulátoru.

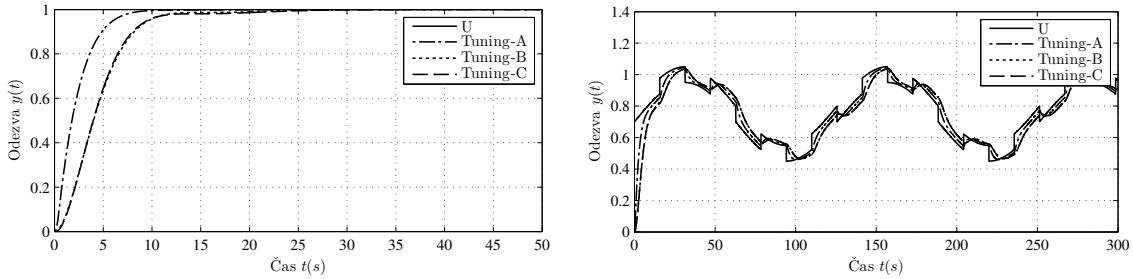
| ⊞ | $T_r$ | $T_s$ | $Min_s$ | $Max_s$ | Over | Under | Peak | $T_p$ |
|---|---|---|---|---|---|---|---|---|
| Výs. [Tuning-A] | 4.385 | 7.481 | 0.902 | 1 | 0 | | 1 | 50 |
| Výs. [Tuning-B] | 6.799 | 11.823 | 0.901 | | | | | |
| Výs. [Tuning-C] | 6.603 | 13.336 | | | | | | |

**Tabulka 7.3:** Tabulka ohodnocení odezvy na skokový signál dle nejlepších výsledků [Regulátor Vlastni-struktura Alg HC12] pro System4.

| ⊞ | Param1 | Param2 | Param3 | Param4 | Param5 |
|---|---|---|---|---|---|
| | Fit. | | | ITAE sin/pwm | |
| Výs. [Tuning-A] | 69.461 | 69.918 | 19.713 | 1.639 | 0.433 |
| | 2.005 | | | 101.213 | |
| Výs. [Tuning-B] | 75.729 | 27.190 | 2.474 | 0.334 | 68.659 |
| | 2.324 | | | 334.383 | |
| Výs. [Tuning-C] | 81.908 | 28.515 | 2.549 | 0.985 | 5.904 |
| | 2.343 | | | 349.044 | |
| P:(tab: 4.1) | B:<0.0,100.0,15> | | | | |
| AI:(tab: 4.2) | M12 [Default], RUNS [100] | | | | |

**Tabulka 7.4:** Tabulka nastavení regulátoru [Regulátor Vlastni-struktura Alg HC12] pro System4.






**Graf 7.3:** Pořadí grafů: (a: Průběh přechodové charakteristiky na skokovém signálu dle [Regulátor Vlastni-struktura Alg HC12] pro System4.) a (b: Průběh přechodové charakteristiky na SIN/PWM signálu dle [Regulátor Vlastni-struktura Alg HC12] pro System4.).

## 7.2 Model čtvrtého řádu s nelineárním prvkem

Nastavení struktury regulátoru dle (GE) s gramatikou (gramatika: 7.2) pro model *system4delay* a doladění výsledné struktury pomocí (HC12). Výsledky jsou vyhod-

noceny na výsledné přechodové funkci při jednotkovém vstupu a konečné výsledky jsou prezentovány, jak v neupravené formě po dokončení výpočtu gramatické evoluce, tak v upravené formě přichystané na doladění dle algoritmu (HC12) spolu s nastavením pro jednotlivé algoritmy. Výsledné nejlepší regulátory jsou porovnány na zaklade Root-Locus charakteristiky (graf: 7.5a) a (graf: 7.5b).

### 7.2.1 Nastavení dle algoritmu GE

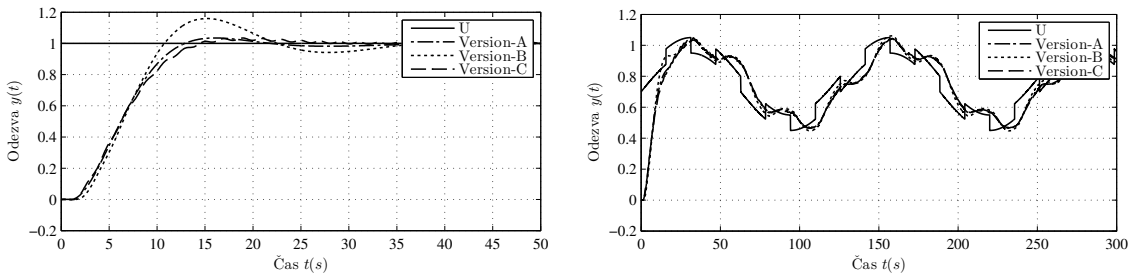
Generovaní regulačních zákonů v prostředí Matlab pro model *system4* dle algoritmu (GE). Regulátor je nastaven dle fitness funkce *F111*. Konečné výsledky jsou v tabulce (tab: 7.6) spolu s nastavením optimalizačního algoritmu. Výsledek gramatické evoluce je prezentován vždy ve dvou verzích, první je neupravený výsledek dle aktivní gramatiky a druha je verze po úpravě k dalšímu doladění vybraných parametrů $param_x$. Ohodnocení přechodové funkce je v tabulce (tab: 7.5) a výsledné hodnoty přechodových funkcí jsou zobrazeny na skokovém grafu (graf: 7.4a) a grafu SIN/PWM (graf: 7.4b) k zobrazení dynamických vlastností regulátoru.

| ⊞ | $T_r$ | $T_s$ | $Min_s$ | $Max_s$ | Over | Under | Peak | $T_p$ |
|---|---|---|---|---|---|---|---|---|
| Výs. [Version-A] | 7.602 | 19.141 | 0.901 | 1.035 | 3.495 | 0.111 | 1.035 | 16.083 |
| Výs. [Version-B] | 6.378 | 34.340 | 0.902 | 1.159 | 15.940 | 0.012 | 1.159 | 15.208 |
| Výs. [Version-C] | 8.559 | 23.367 | 0.900 | 1.035 | 3.478 | 0.129 | 1.035 | 18.917 |

**Tabulka 7.5:** Tabulka ohodnocení odezvy na skokový signál dle nejlepších výsledků [Regulátor Vlastni-struktura Alg GE] pro System4Delay.

| ⊞ | Výsledek | Fit. | ITAE sin/pwm |
|---|---|---|---|
| Výs. [Version-A] | GE výsledek (expr: 7.4) a (vz: 7.4) | 3.998 | 834.864 |
| Výs. [Version-B] | GE výsledek (expr: 7.5) a (vz: 7.5) | 4.130 | 1422.749 |
| Výs. [Version-C] | GE výsledek (expr: 7.6) a (vz: 7.6) | 4.307 | 850.121 |
| P:(tab: 4.1) | C:<40,8> | ⊡ | |
| AI:(tab: 4.2) | NP [500], F [0.2], R [0.5], PC [2], W [10], GEN [200], RUNS [100] | | |

**Tabulka 7.6:** Tabulka nastavení regulátoru [Regulátor Vlastni-struktura Alg GE] pro System4Delay.




**Graf 7.4:** Pořadí grafů: (a: Průběh přechodové charakteristiky na skokovém signálu dle [Regulátor Vlastni-struktura Alg GE] pro System4Delay.) a (b: Průběh přechodové charakteristiky na SIN/PWM signálu dle [Regulátor Vlastni-struktura Alg GE] pro System4Delay.).

```
expression = (tf([9,(6-3)],[(9-7),(((7*5)+7)-(9/(4+-1)))])*tf([8,1],[3,0]));
```

**GE výsledek 7.4:** Vypočtený výraz dle GE gramatiky [Typ Version-A] pro System4Delay.

```
expression = tf([5,(6/2)],[(0+3),[(9/7),(9+4)]])*tf([9,(3-2)],[(6+2),(7*0)]);
```

**GE výsledek 7.5:** Vypočtený výraz dle GE gramatiky [Typ Version-B] pro System4Delay.

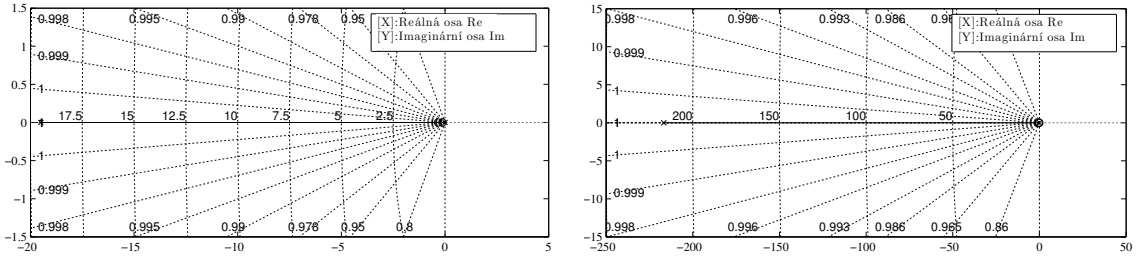
```
expression = (tf([5,(5/4)],[1,(1+-1)])/tf([((3+2)*(1/1)),(6*[0,(6+2)])],[1,[(2
*3),1]]));
```

**GE výsledek 7.6:** Vypočtený výraz dle GE gramatiky [Typ Version-C] pro System4Delay.

$$\begin{aligned} Gr_{GE}(s) &= \frac{72s^2 + 33s^1 + 3}{6s^2 + 117s^1} \\ Gr_{HC12}(s) &= \frac{Param1s^2 + Param2s^1 + Param3}{Param4s^2 + Param5s^1} \end{aligned} \tag{7.4}$$

$$\begin{aligned} Gr_{GE}(s) &= \frac{45s^2 + 32s^1 + 3}{24s^3 + 10.2857s^2 + 104s^1} \\ Gr_{HC12}(s) &= \frac{Param1s^2 + Param2s^1 + Param3}{Param4s^3 + Param5s^2 + Param6s^1} \end{aligned} \tag{7.5}$$

$$\begin{aligned} Gr_{GE}(s) &= \frac{5s^3 + 31.25s^2 + 12.5s^1 + 1.25}{5s^3 + 48s^1} \\ Gr_{HC12}(s) &= \frac{Param1s^3 + Param2s^2 + Param3s^1 + Param4}{Param5s^3 + Param6s^1} \end{aligned} \tag{7.6}$$




**Graf 7.5:** Pořadí grafů: (a: Root-Locus charakteristika pro regulátor Version-A System4Delay.) a (b: Root-Locus charakteristika pro regulátor Tuning-A System4Delay.).



### 7.2.2 Nastavení dle algoritmu HC12

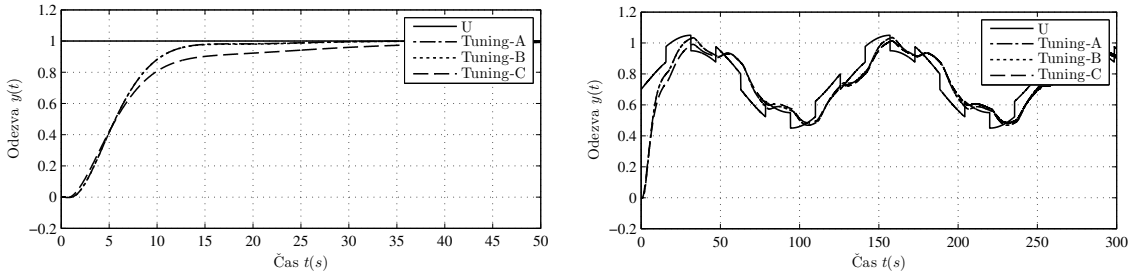
Ladění konečných výsledků gramatické evoluce pro model *system4* dle algoritmu (HC12). Regulátor je nastaven dle fitness funkce *F111*. Konečné výsledky jsou v tabulce (tab: 7.8) spolu s nastavením optimalizačního algoritmu. Optimalizované struktury regulátoru jsou výsledky gramatické evoluce z předešlé části a používá se upravená forma generovaného výsledku, která obsahuje optimalizované parametry $param_x$. Ohodnocení přechodové funkce je v tabulce (tab: 7.7) a výsledné hodnoty přechodových funkcí jsou zobrazeny na skokové funkci (graf: 7.6a) a grafu SIN/PWM (graf: 7.6b) k zobrazení dynamických vlastností regulátoru.

| ⊞ | $T_r$ | $T_s$ | $Min_s$ | $Max_s$ | Over | Under | Peak | $T_p$ |
|---|---|---|---|---|---|---|---|---|
| Výs. [Tuning-A] | 7.812 | 15.390 | 0.902 | 1 | 0 | 0.154 | 1 | 50 |
| Výs. [Tuning-B] | 7.804 | 18.950 | | | | 0.177 | | |
| Výs. [Tuning-C] | 12.440 | 39.088 | 0.900 | 0.991 | | 0.312 | 0.991 | |

**Tabulka 7.7:** Tabulka ohodnocení odezvy na skokový signál dle nejlepších výsledků [Regulátor Vlastni-struktura Alg HC12] pro System4Delay.

| ⊞ | Param1 | Param2 | Param3 | Param4 | Param5 | Param6 |
|---|---|---|---|---|---|---|
| | Fit. | | | ITAE sin/pwm | | |
| Výs. [Tuning-A] | 64.681 | 22.607 | 2.038 | 0.360 | 78.043 | - |
| | 3.213 | | | 685.233 | | |
| Výs. [Tuning-B] | 81.734 | 28.565 | 2.554 | 0.404 | 1.174 | 98.412 |
| | 3.227 | | | 707.894 | | |
| Výs. [Tuning-C] | 24.002 | 64.221 | 21.298 | 1.534 | 0 | 75.737 |
| | 3.587 | | | 1620.673 | | |
| P:(tab: 4.1) | B:<0.0,100.0,15> | | | | | |
| AI:(tab: 4.2) | M12 [Default], RUNS [100] | | | | | |

**Tabulka 7.8:** Tabulka nastavení regulátoru [Regulátor Vlastni-struktura Alg HC12] pro System4Delay.




**Graf 7.6:** Pořadí grafů: (a: Průběh přechodové charakteristiky na skokovém signálu dle [Regulátor Vlastni-struktura Alg HC12] pro System4Delay.) a (b: Průběh přechodové charakteristiky na SIN/PWM signálu dle [Regulátor Vlastni-struktura Alg HC12] pro System4Delay.).

## 7.3 Model kuličky v obruči

7

Nastavení struktury regulátoru dle (GE) s gramatikou (gramatika: 7.2) pro model *ballandloop* a doladění výsledné struktury pomocí (HC12). Výsledky jsou vyhodnoceny na výsledné přechodové funkci při jednotkovém vstupu a konečné výsledky jsou prezentovány, jak v neupravené formě po dokončení výpočtu gramatické evoluce, tak v upravené formě přichystané na doladění dle algoritmu (HC12) spolu s nastavením pro jednotlivé algoritmy. Výsledné nejlepší regulátory jsou porovnány na zaklade Root-Locus charakteristiky (graf: 7.8a) a (graf: 7.8b).

### 7.3.1 Nastavení dle algoritmu GE

Generovaní regulačních zákonů v prostředí Matlab pro model *system4* dle algoritmu (GE). Regulátor je nastaven dle fitness funkce *F111*. Konečné výsledky jsou v tabulce (tab: 7.10) spolu s nastavením optimalizačního algoritmu. Výsledek gramatické evoluce je prezentován vždy ve dvou verzích, první je neupravený výsledek dle aktivní gramatiky a druha je verze po úpravě k dalšímu doladění vybraných parametrů $param_x$. Ohodnocení přechodové funkce je v tabulce (tab: 7.9) a výsledné hodnoty přechodových funkcí jsou zobrazeny na skokovém grafu (graf: 7.7a) a grafu SIN/PWM (graf: 7.7b) k zobrazení dynamických vlastností regulátoru.

```
expression = (3-tf([(7-1),(0-9)],[(2/((4+8)*(7/3))),([3,(((3-1)-(3-9))/(7-1))]
-1)]));
```

**GE výsledek 7.7:** Vypočtený výraz dle GE gramatiky [Typ Version-A] pro BallAndLoop.

```
expression = (1*tf([[1,1],(9*8)],[[(3/9),8],1]));
```

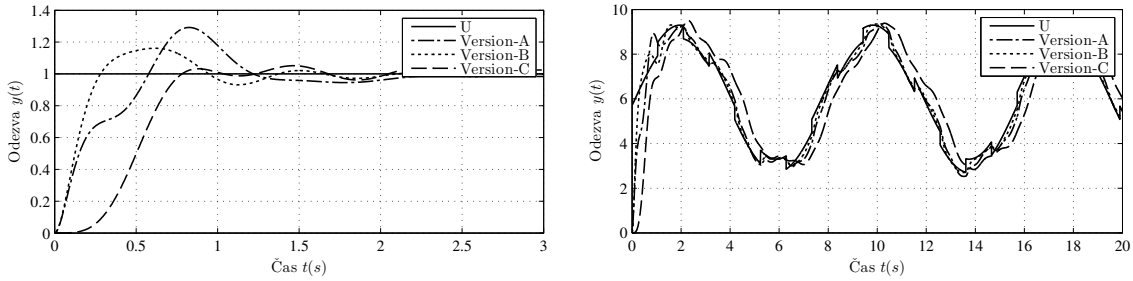
**GE výsledek 7.8:** Vypočtený výraz dle GE gramatiky [Typ Version-B] pro BallAndLoop.

| ⊞ | $T_r$ | $T_s$ | $Min_s$ | $Max_s$ | Over | Under | Peak | $T_p$ |
|---|---|---|---|---|---|---|---|---|
| Výs. [Version-A] | 0.478 | 2.166 | 0.901 | 1.292 | 29.188 | | 1.292 | 0.825 |
| Výs. [Version-B] | 0.190 | 1.958 | 0.905 | 1.161 | 16.117 | 0 | 1.161 | 0.618 |
| Výs. [Version-C] | 0.398 | NaN | 0.902 | 1.052 | 5.157 | | 1.052 | 1.468 |

**Tabulka 7.9:** Tabulka ohodnocení odezvy na skokový signál dle nejlepších výsledků [Regulátor Vlastni-struktura Alg GE] pro BallAndLoop.

| ⊞ | Výsledek | Fit. | ITAE sin/pwm |
|---|---|---|---|
| Výs. [Version-A] | GE výsledek (expr: 7.7) a (vz: 7.7) | 3.036 | 90.536 |
| Výs. [Version-B] | GE výsledek (expr: 7.8) a (vz: 7.8) | 3.404 | 45.255 |
| Výs. [Version-C] | GE výsledek (expr: 7.9) a (vz: 7.9) | 3.753 | 88.491 |
| P:(tab: 4.1) | C:<40,8> | ⊡ | |
| AI:(tab: 4.2) | NP [500], F [0.2], R [0.5], PC [2], W [10], GEN [200], RUNS [100] | | |

**Tabulka 7.10:** Tabulka nastavení regulátoru [Regulátor Vlastni-struktura Alg GE] pro BallAndLoop.




**Graf 7.7:** Pořadí grafů: (a: Průběh přechodové charakteristiky na skokovém signálu dle [Regulátor Vlastni-struktura Alg GE] pro BallAndLoop.) a (b: Průběh přechodové charakteristiky na SIN/PWM signálu dle [Regulátor Vlastni-struktura Alg GE] pro BallAndLoop.).

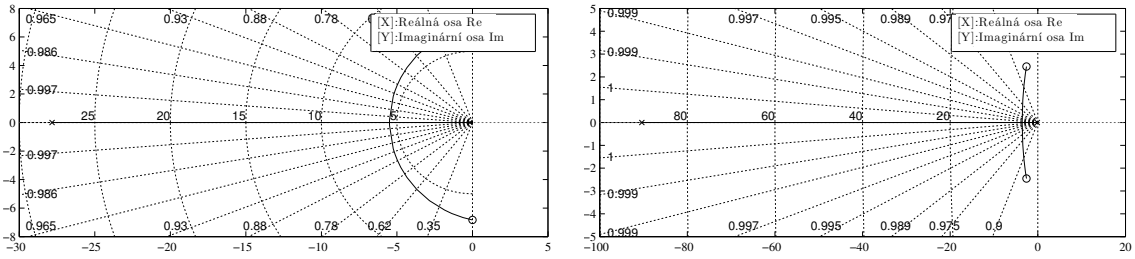
$$\begin{aligned} Gr_{GE}(s) &= \frac{0.21429s^2 + 10}{0.071429s^2 + 2s^1 + 0.33333} \\ Gr_{HC12}(s) &= \frac{Param1s^2 + Param2}{Param3s^2 + Param4s^1 + Param5} \end{aligned} \tag{7.7}$$

$$\begin{aligned} Gr_{GE}(s) &= \frac{1s^2 + 1s^1 + 72}{0.33333s^2 + 8s^1 + 1} \\ Gr_{HC12}(s) &= \frac{Param1s^2 + Param2s^1 + Param3}{Param4s^2 + Param5s^1 + Param6} \end{aligned} \tag{7.8}$$

```
expression = (7*tf([4,9],[3,[(2*8),[(9*6),0]]]));
```

**GE výsledek 7.9:** Vypočtený výraz dle GE gramatiky [Typ Version-C] pro BallAndLoop.

$$Gr_{GE}(s) = \frac{28s^1 + 63}{3s^3 + 16s^2 + 54s^1}$$
$$Gr_{HC12}(s) = \frac{Param1s^1 + Param2}{Param3s^3 + Param4s^2 + Param5s^1} \quad (7.9)$$




**Graf 7.8:** Pořadí grafů: (a: Root-Locus charakteristika pro regulátor Version-A BallAndLoop.) a (b: Root-Locus charakteristika pro regulátor Tuning-A BallAndLoop.).

### 7.3.2 Nastavení dle algoritmu HC12

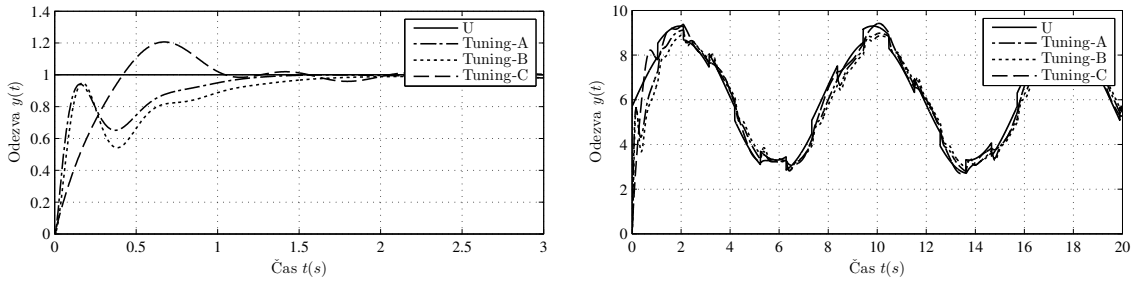
Ladění konečných výsledků gramatické evoluce pro model *system4* dle algoritmu (HC12). Regulátor je nastaven dle fitness funkce *F111*. Konečné výsledky jsou v tabulce (tab: 7.12) spolu s nastavením optimalizačního algoritmu. Optimalizované struktury regulátoru jsou výsledky gramatické evoluce z předešlé části a používá se upravená forma generovaného výsledku, která obsahuje optimalizované parametry $param_x$. Ohodnocení přechodové funkce je v tabulce (tab: 7.11) a výsledné hodnoty přechodových funkcí jsou zobrazeny na skokové funkci (graf: 7.9a) a grafu SIN/PWM (graf: 7.9b) k zobrazení dynamických vlastností regulátoru.

| ⊞ | $T_r$ | $T_s$ | $Min_s$ | $Max_s$ | Over | Under | Peak | $T_p$ |
|---|---|---|---|---|---|---|---|---|
| Výs. [Tuning-A] | 0.105 | 1.200 | 0.651 | 0.999 | 0 | | 0.999 | 1.698 |
| Výs. [Tuning-B] | 0.112 | 1.699 | 0.542 | 0.990 | | | 0.990 | 2.193 |
| Výs. [Tuning-C] | 0.329 | 2.591 | 0.901 | 1.207 | 20.670 | 0 | 1.207 | 0.673 |

**Tabulka 7.11:** Tabulka ohodnocení odezvy na skokový signál dle nejlepších výsledků [Regulátor Vlastni-struktura Alg HC12] pro BallAndLoop.

| ⊞ | Param1 | Param2 | Param3 | Param4 | Param5 | Param6 |
|---|---|---|---|---|---|---|
| | Fit. | | | ITAE sin/pwm | | |
| Výs. [Tuning-A] | 7.665 | 86.360 | 9.131 | 74.108 | 9.202 | - |
| | 2.280 | | | 47.656 | | |
| Výs. [Tuning-B] | 8.088 | 36.362 | 74.528 | 1 | 36.272 | 3.612 |
| | 2.499 | | | 78.934 | | |
| Výs. [Tuning-C] | 1.167 | 86.948 | 0 | 14.307 | 1.036 | - |
| | 2.717 | | | 51.428 | | |
| P:(tab: 4.1) | B:<0.0,100.0,15> | | | | | |
| AI:(tab: 4.2) | M12 [Default], RUNS [100] | | | | | |

**Tabulka 7.12:** Tabulka nastavení regulátoru [Regulátor Vlastni-struktura Alg HC12] pro BallAndLoop.




**Graf 7.9:** Pořadí grafů: (a: Průběh přechodové charakteristiky na skokovém signálu dle [Regulátor Vlastni-struktura Alg HC12] pro BallAndLoop.) a (b: Průběh přechodové charakteristiky na SIN/PWM signálu dle [Regulátor Vlastni-struktura Alg HC12] pro BallAndLoop.).

## 7.4 Model magnetické levitace

Nastavení vlastni struktury regulátoru gramatickou evolucí s aktivní gramatikou (gramatika: 7.1) pro model *maglev* a doladění výsledné struktury pomocí algoritmu (HC12). Výsledky jsou vyhodnoceny na výsledné přechodové funkci pomocí statistiky při jednotkovém vstupu a konečné výsledky jsou prezentovány, jak v neupravené formě.

### 7.4.1 Nastavení dle algoritmu GE

Generovaní regulačních zákonů v prostředí Matlab pro model *system4* dle algoritmu (GE). Regulátor je nastaven dle fitness funkce *F111*. Konečné výsledky jsou v tabulce (tab: 7.14) spolu s nastavením optimalizačního algoritmu. Výsledek gramatické evoluce je prezentován vždy ve dvou verzích, první je neupravený výsledek dle aktivní gramatiky a druha je verze po úpravě k dalšímu doladění vybraných parametrů $param_x$. Ohodnocení přechodové funkce je v tabulce (tab: 7.13) a výsledné hodnoty přechodových funkcí jsou zobrazeny na skokovém grafu (graf: 7.10a) a grafu PWM (graf: 7.10b) k zobrazení dynamických vlastností regulátoru.

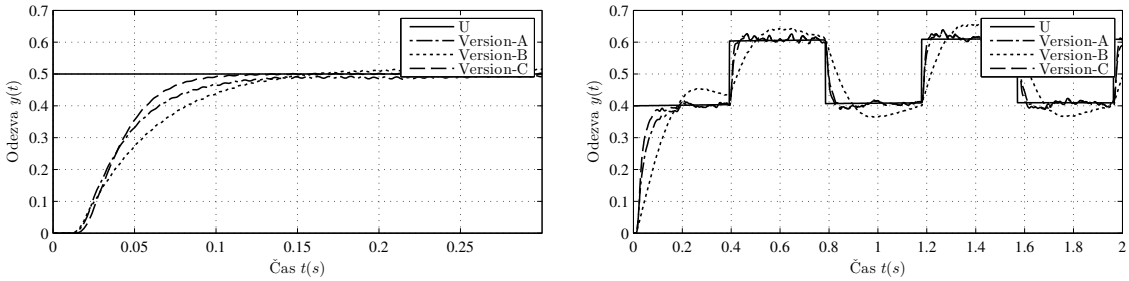


| ⊞ | $T_r$ | $T_s$ | $Min_s$ | $Max_s$ | Over | Under | Peak | $T_p$ |
|---|---|---|---|---|---|---|---|---|
| Výs. [Version-A] | 0.065 | 0.293 | 0.450 | 0.496 | 0 | | 0.496 | 0.289 |
| Výs. [Version-B] | 0.087 | NaN | 0.451 | 0.519 | 3.700 | 0 | 0.519 | 0.272 |
| Výs. [Version-C] | 0.047 | 0.099 | | 0.502 | 0.481 | | 0.502 | 0.286 |

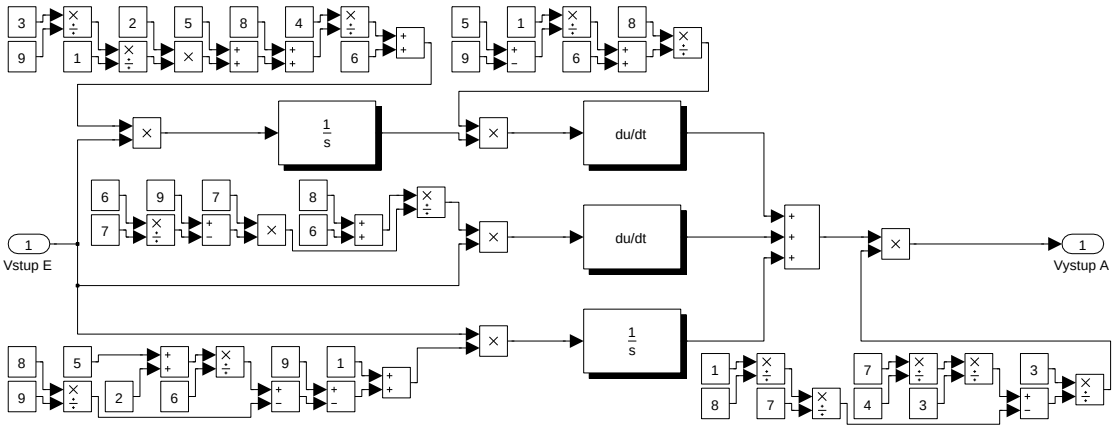
**Tabulka 7.13:** Tabulka ohodnocení odezvy na skokový signál dle nejlepších výsledků [Regulátor Vlastni-struktura Alg GE] pro Maglev.

| ⊞ | Výsledek | Fit. | ITAE sin/pwm |
|---|---|---|---|
| Výs. [Version-A] | (model: 7.1) a (model: 7.2) | 1.667 | 1.098 |
| Výs. [Version-B] | (model: 7.3) a (model: 7.4) | 1.745 | 1.461 |
| Výs. [Version-C] | (model: 7.5) a (model: 7.6) | 1.816 | 0.603 |
| P:(tab: 4.1) | C:<40,8> | ⊡ | |
| AI:(tab: 4.2) | NP [500], F [0.2], R [0.5], PC [2], W [10], GEN [200], RUNS [100] | | |

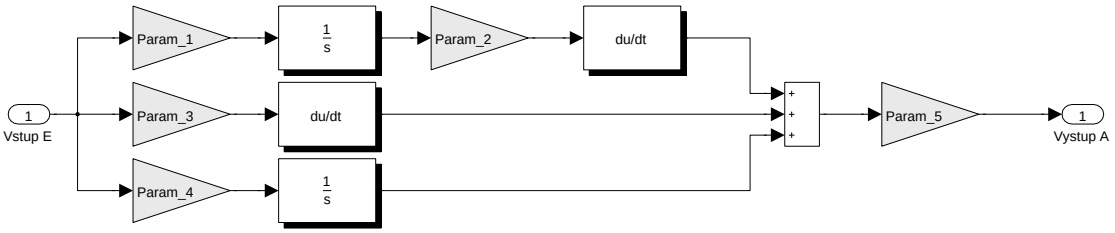
**Tabulka 7.14:** Tabulka nastavení regulátoru [Regulátor Vlastni-struktura Alg GE] pro Maglev.



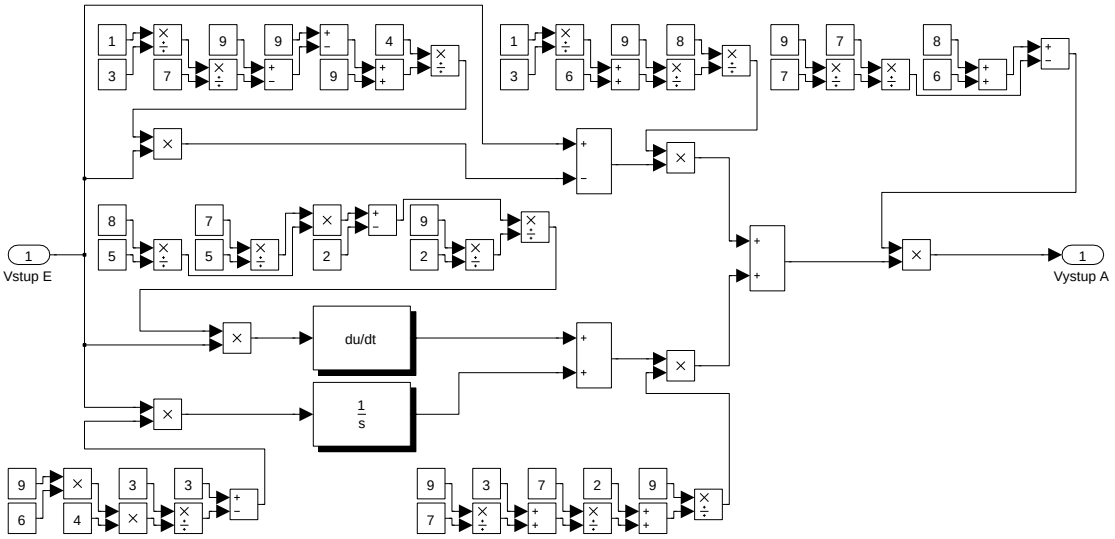

**Graf 7.10:** Pořadí grafů: (a: Průběh přechodové charakteristiky na skokovém signálu dle [Regulátor Vlastni-struktura Alg GE] pro Maglev.) a (b: Průběh přechodové charakteristiky na SIN/PWM signálu dle [Regulátor Vlastni-struktura Alg GE] pro Maglev.).



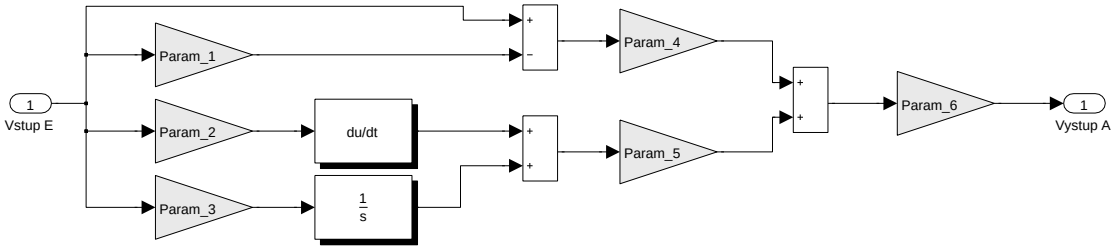

**Model 7.1:** Vygenerovaný model dle GE gramatiky [Typ Version-A] pro Maglev.



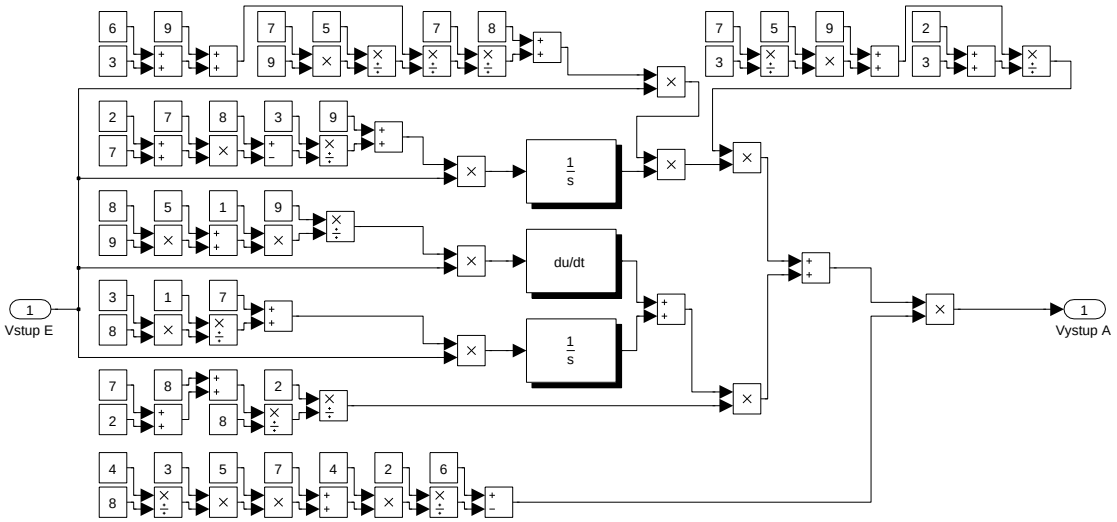

**Model 7.2:** Upravený GE model k optimalizaci dle HC12 [Typ Version-A] pro Maglev.



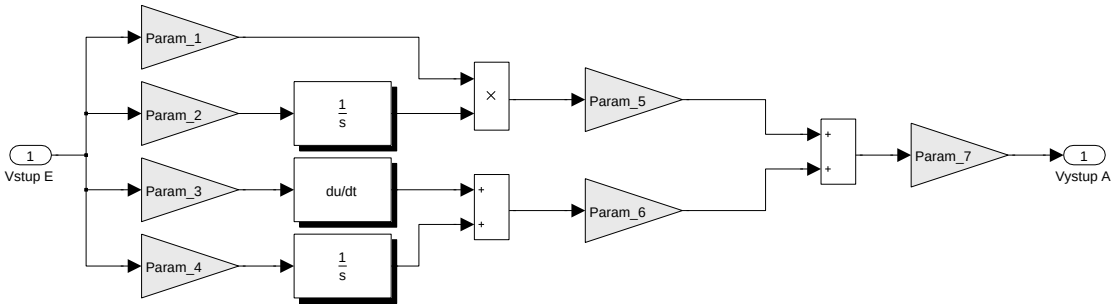

**Model 7.3:** Vygenerovaný model dle GE gramatiky [Typ Version-B] pro Maglev.




**Model 7.4:** Upravený GE model k optimalizaci dle HC12 [Typ Version-B] pro Maglev.




**Model 7.5:** Vygenerovaný model dle GE gramatiky [Typ Version-C] pro Maglev.




**Model 7.6:** Upravený GE model k optimalizaci dle HC12 [Typ Version-C] pro Maglev.



## 7.4.2 Nastavení dle algoritmu HC12

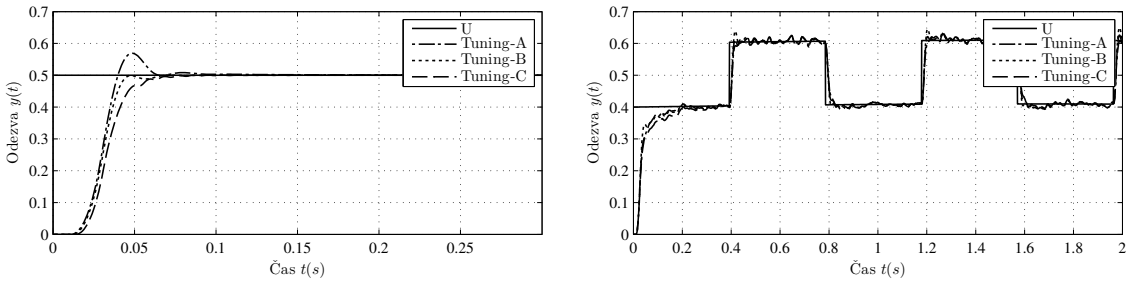
Ladění konečných výsledků gramatické evoluce pro model *system4* dle algoritmu (HC12). Regulátor je nastaven dle fitness funkce *F111*. Konečné výsledky jsou v tabulce (tab: 7.16) spolu s nastavením optimalizačního algoritmu. Optimalizované struktury regulátoru jsou výsledky gramatické evoluce z předešlé části a používá se upravená forma generovaného výsledku, která obsahuje optimalizované parametry $param_x$. Ohodnocení přechodové funkce je v tabulce (tab: 7.15) a výsledné hodnoty přechodových funkcí jsou zobrazeny na skokové funkci (graf: 7.11a) a grafu PWM (graf: 7.11b) k zobrazení dynamických vlastností regulátoru.

| ⊞ | $T_r$ | $T_s$ | $Min_s$ | $Max_s$ | Over | Under | Peak | $T_p$ |
|---|---|---|---|---|---|---|---|---|
| Výs. [Tuning-A] | 0.017 | 0.061 | 0.467 | 0.569 | 13.807 | 0 | 0.569 | 0.049 |
| Výs. [Tuning-B] | 0.019 | 0.060 | 0.453 | 0.500 | 0.092 | | 0.500 | 0.071 |
| Výs. [Tuning-C] | 0.023 | 0.065 | 0.456 | | 0.079 | | | 0.095 |

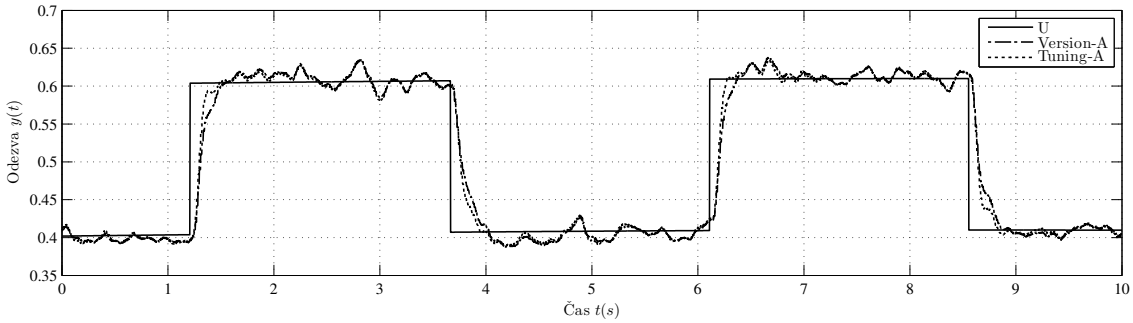
**Tabulka 7.15:** Tabulka ohodnocení odezvy na skokový signál dle nejlepších výsledků [Regulátor Vlastni-struktura Alg HC12] pro Maglev.

| ⊞ | Param1 | Param2 | Param3 | Param4 | Param5 | Param6 |
|---|---|---|---|---|---|---|
| | Param7 | | Fit. | | ITAE sin/pwm | |
| Výs. [Tuning-A] | 0.810 | 0.006 | 7.462 | 8.624 | 3.092 | 2.810 |
| | - | | 1.080 | | 0.316 | |
| Výs. [Tuning-B] | 0.427 | 2.114 | 0.011 | 8.906 | 6.275 | - |
| | - | | 1.385 | | 0.253 | |
| Výs. [Tuning-C] | 3.725 | 8.600 | 0.061 | 8.791 | 7.130 | 0.703 |
| | 8.067 | | 1.430 | | 0.326 | |
| P:(tab: 4.1) | B:<0.0,10.0,15> | | | | | |
| AI:(tab: 4.2) | M12 [Default], RUNS [100] | | | | | |

**Tabulka 7.16:** Tabulka nastavení regulátoru [Regulátor Vlastni-struktura Alg HC12] pro Maglev.

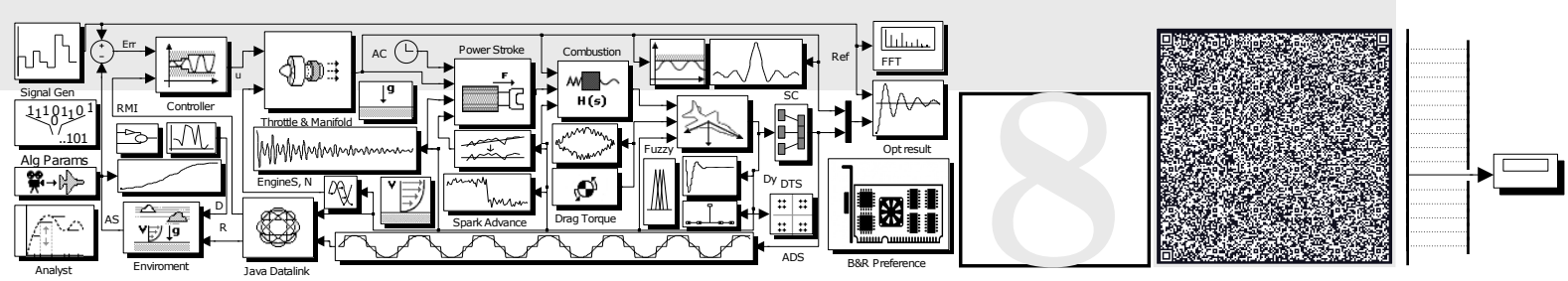



**Graf 7.11:** Pořadí grafů: (a: Průběh přechodové charakteristiky na skokovém signálu dle [Regulátor Vlastni-struktura Alg HC12] pro Maglev.) a (b: Průběh přechodové charakteristiky na SIN/PWM signálu dle [Regulátor Vlastni-struktura Alg HC12] pro Maglev.).




**Graf 7.12:** Průběh přechodové charakteristiky na reálné soustavě pro Maglev.

Po vytvoření struktury regulátoru a po aplikaci výsledného doladění pomocí (HC12) byly regulátory vyzkoušené na reálné soustavě (graf: 7.12). Graf prezentuje stabilitu polohy na změnu žádané hodnoty dle PWM. Oproti předešlým typům regulátorům výsledek *Tuning-A* vykazuje větší stupeň stability na regulovaném systému.




# KAPITOLA 8

# POROVNÁNÍ VÝSLEDKŮ PRO PŘEDCHOZÍ MODELY

Hodnocení konečných výsledků jsou porovnány na základě fitness funkce ze sekce (sekce: 4.1.1 - viz str. 46) v podobě $A = 1$, $B = 1$ a $C = 1$ v označení *F111* ke standartnímu nastavení pro jednotlivé systémy, které je prezentované v popisu jednotlivých systémů ze sekce (sekce: 4.2 - viz str. 47).

Dále je porovnáván dynamické chování systému na základě (ITAE) funkce (vz: 4.2) k základnímu nastavení SIN/PWM přechodové funkci. Fitness *F111* je prezentované graficky pro všechny opakování algoritmu v daném typu regulace (100 opakování) jako hlavní kritérium hodnocení v této práci. Výsledky jsou rozdělené do třech základních částí, první představuje regulaci dle (PID) regulátoru pro algoritmy HC12, diferenciální evoluci a Nelder-Mead metodu ze sekce (sekce: 5 - viz str. 61). Druhá část výsledů je fuzzy regulace pro typy *FPID*, *FP* a *FPID-FP*, pro první systém (model čtvrtého řádu), ostatní systémy se prezentují ve formě *FPID-FP*, pro algoritmus (HC12), výsledky ze sekce (sekce: 6 - viz str. 81). Poslední typ výsledů je generování struktury samotného regulátoru, sekce (sekce: 7 - viz str. 93), a představují výsledky z gramatické evoluce a následující doladění v podobě algoritmu HC12 na vybraných parametrech struktury regulátoru z (GE) výpočtu. Finální hodnoty výsledů jsou dále zobrazeny v tabulce pro nejlepší hodnocení, pro medián a pro standardní odchylku rozptylu ze všech opakování algoritmu, jak pro *F111* tak pro dynamické chování SIN/PWM funkce.





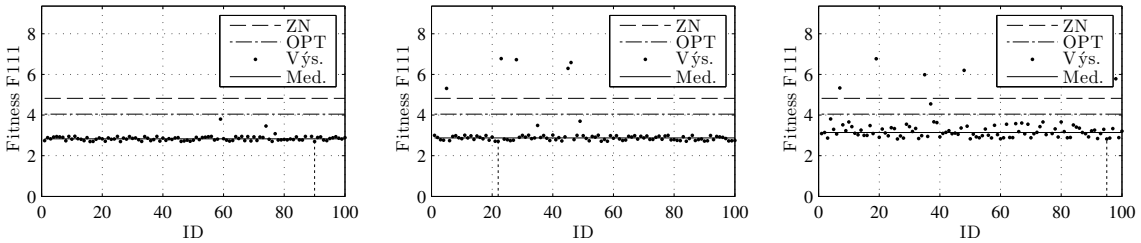
Poslední graf pro jednotlivé porovnání představuje průběh nejlepší fitness hodnoty *F111* pro jednotlivé typy regulace a zobrazuje procentuální zlepšení těchto typů regulátorů / optimalizace oproti základnímu nastavení. Procentuální hodnoty jsou zobrazeny jak pro *F111* fitness funkci tak pro ohodnocení SIN/PWM přecho-

dové funkce ve formě základní ITAE funkce. Následující výsledky ukazují vhodnost použití evolučních algoritmů na hledání optimálních regulátorů pro různé typy systémů. První část prezentuje optimalizaci (PID), kdy při použití algoritmů (HC12), (DE) a (NM) dokáží nalézt vhodnější řešení k dané fitness funkci oproti standartním metodám nastavení. Další výhodou je vhodnost použití tohoto postupu při řešení nelineárních systémů, kdy není možno použít standartní metody typu (ZN) a další. Při výpočtu fuzzy regulace jsou prezentované obdobné výsledky jak pro (PID) s lepší dynamickou odezvou při SIN/PWM přechodové funkci. Posední část je vytváření vlastních struktur regulátoru a prezentuje vhodnost metody v případě použití atypického regulátoru, který nemá pevně danou strukturu nastavení. Při správném navolení definice gramatiky generování struktury dosahují konečné výsledky obdobné výsledky jak nejlepší nastavení (PID) regulátoru. Poslední možnost je následné doladění výsledné struktury (GE) regulace pomocí (HC12) algoritmu na vybraných parametrech. Kdy tato metoda představuje nejlepší výsledky jak pro fitness *F111* tak z hlediska dynamické přechodové funkce.

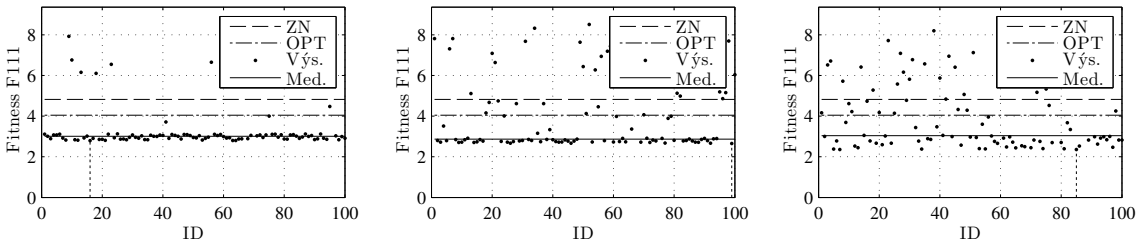
## 8.1 Model čtvrtého řádu

Výsledné hodnoty pro model (model čtvrtého řádu) a jejich grafické znázornění. Graf fitness hodnoty *F111* pro nastavení PID regulátoru ke standartnímu nastavení, algoritmus HC12 (graf: 8.1a), DE (graf: 8.1b) a NM (graf: 8.1c), nastavení fuzzy regulátoru pro typ regulace, *FPID* (graf: 8.2a), *FP* (graf: 8.2b) a *FPID-FP* (graf: 8.2c), výsledky ohodnocení generování vlastní struktury pro GE (graf: 8.3a) a následné doladění dle algoritmu HC12 (graf: 8.3b). Grafická reprezentace zlepšení použitých algoritmů oproti základnímu nastavení spolu s nejlepšími hodnotami *F111* fitness funkce (graf: 8.4). Konečné nejlepší výsledky pro jednotlivé typy regulace (tab: 8.1), medián (tab: 8.2) a standartní odchylka (tab: 8.3) pro všechny opakování algoritmu. Box statistika fitness všech typů regulátorů je v grafu (graf: 8.5).

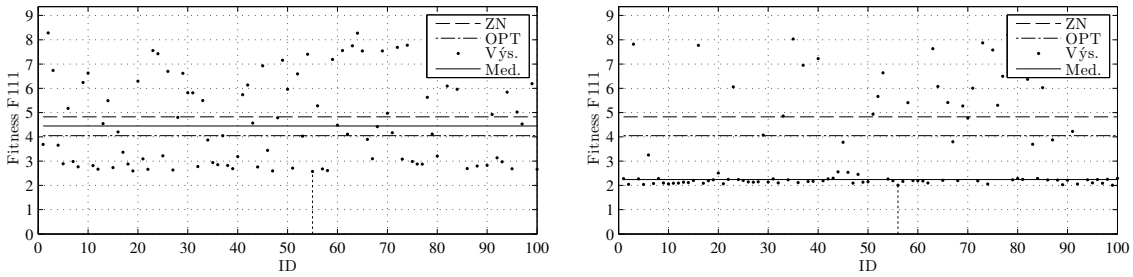




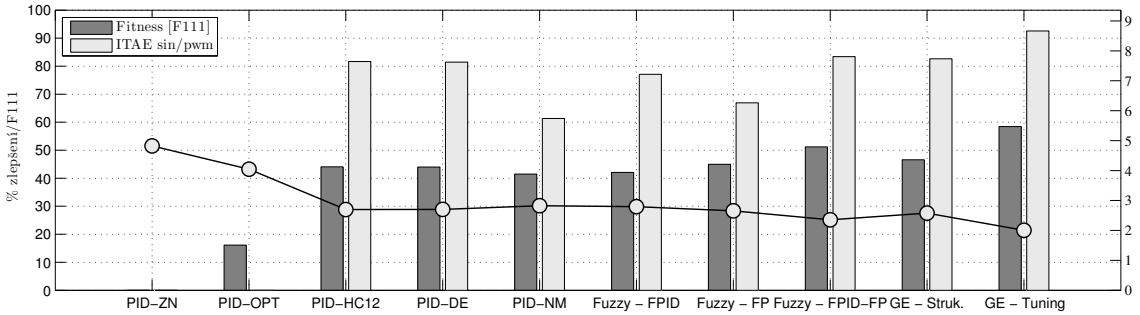

**Graf 8.1:** Hodnoty všech fitness hodnocení [F111] k základnímu nastavení pro výpočet [PID] regulátoru. Pořadí grafů: (a: PID-HC12), (b: PID-DE) a (c: PID-NM).




**Graf 8.2:** Hodnoty všech fitness hodnocení [F111] k základnímu nastavení pro výpočet [FUZZY] regulátoru. Pořadí grafů: (a: Fuzzy - FPID), (b: Fuzzy - FP) a (c: Fuzzy - FPID-FP).




**Graf 8.3:** Hodnoty všech fitness hodnocení [F111] k základnímu nastavení pro výpočet [GE-Tuning] regulátoru. Pořadí grafů: (a: GE - Struk.) a (b: GE - Tuning).




**Graf 8.4:** Hodnoty fitness funkce pro všechny typy regulace a procentuální zlepšení oproti základnímu nastavení regulátoru [model čtvrtého řádu].

| PID-ZN | | PID-OPT | | PID-HC12 | | PID-DE | | PID-NM | |
|---|---|---|---|---|---|---|---|---|---|
| 4.82 | 1363.59 | 4.05 | 2785.53 | 2.70 | 250.19 | 2.70 | 252.90 | 2.82 | 527.31 |
| Fuzzy - FPID | | Fuzzy - FP | | Fuzzy - FPID-FP | | GE - Struk. | | GE - Tuning | |
| 2.80 | 312.37 | 2.65 | 451.08 | 2.35 | 226.44 | 2.58 | 236.61 | 2.01 | 101.21 |
| Nejlepší hodnoty fitness funkce ve formátu: Fitness [F111] \| ITAE SIN/PWM | | | | | | | | | |

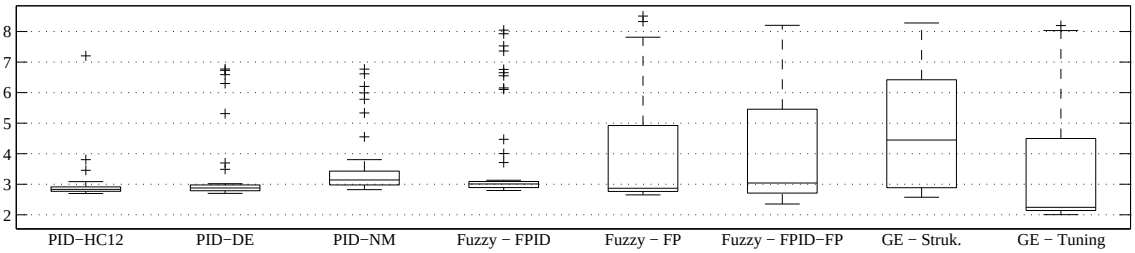
**Tabulka 8.1:** Hodnoty výsledných fitness hodnot ze všech typů regulace [model čtvrtého řádu].

| PID-ZN | | PID-OPT | | PID-HC12 | | PID-DE | | PID-NM | |
|---|---|---|---|---|---|---|---|---|---|
| 4.82 | 1363.59 | 4.05 | 2785.53 | 2.83 | 287.97 | 2.88 | 299.42 | 3.14 | 614.88 |
| Fuzzy - FPID | | Fuzzy - FP | | Fuzzy - FPID-FP | | GE - Struk. | | GE - Tuning | |
| 3.01 | 372.25 | 2.87 | 581.71 | 3.04 | 727.53 | 4.45 | 1360.62 | 2.24 | 121.19 |
| Medián hodnot fitness funkce ve formátu: Fitness [F111] \| ITAE SIN/PWM | | | | | | | | | |

**Tabulka 8.2:** Hodnoty mediánů fitness hodnot ze všech typů regulace [model čtvrtého řádu].

| PID-ZN | | PID-OPT | | PID-HC12 | | PID-DE | | PID-NM | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | | | | 0.46 | 372.48 | 0.78 | 436.98 | 0.76 | 432.20 |
| Fuzzy - FPID | | Fuzzy - FP | | Fuzzy - FPID-FP | | GE - Struk. | | GE - Tuning | |
| 1.19 | 427.81 | 1.73 | 662.90 | 1.75 | 780.72 | 1.85 | 759.02 | 1.93 | 626.86 |
| STD hodnoty fitness funkce ve formátu: Fitness [F111] \| ITAE SIN/PWM | | | | | | | | | |

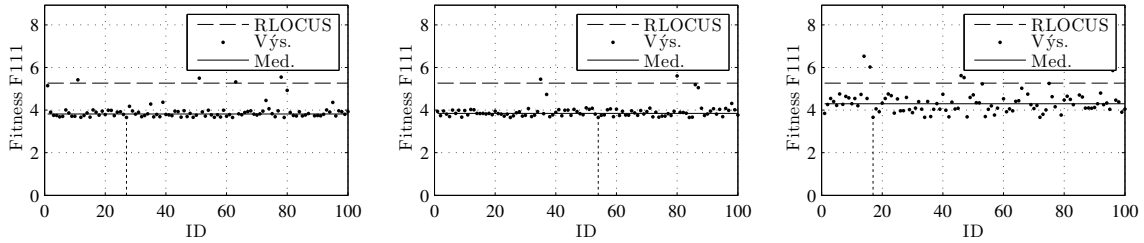
**Tabulka 8.3:** Hodnoty výsledných fitness STD hodnot ze všech typů regulace [model čtvrtého řádu].




**Graf 8.5:** Box graf hodnot fitness funkce pro všechny typy regulace [model čtvrtého řádu].

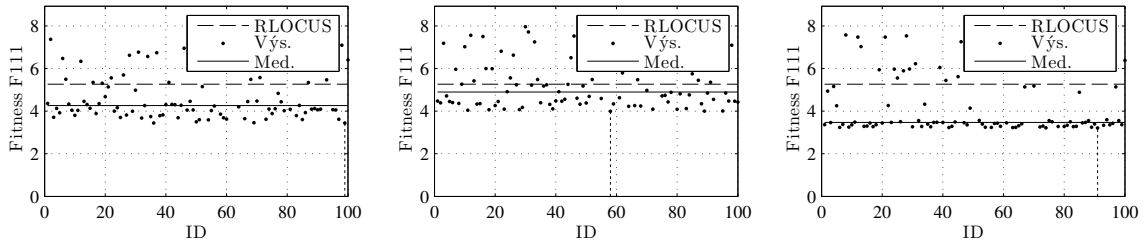
## 8.2 Model čtvrtého řádu s nelineárním prvkem

Výsledné hodnoty a jejich grafické znázornění. Graf fitness hodnoty *F111* pro nastavení PID regulátoru dle algoritmu HC12 (graf: 8.6a), DE (graf: 8.6b) a NM (graf: 8.6c), nastavení fuzzy regulátoru pro typ regulace, *FPID-FP* (graf: 8.7a), výsledky ohodnocení generování vlastní struktury pro GE (graf: 8.7b) a následné doladění dle algoritmu HC12 (graf: 8.7c). Grafická reprezentace zlepšení použitých algoritmů oproti základnímu nastavení spolu s nejlepšími hodnotami *F111* fitness funkce (graf: 8.8). Konečné nejlepší výsledky pro jednotlivé typy regulace (tab: 8.4), medián (tab: 8.5) a standartní odchylka (tab: 8.6) pro všechny opakování algoritmu. Box statistika fitness všech typů regulátorů je v grafu (graf: 8.9).

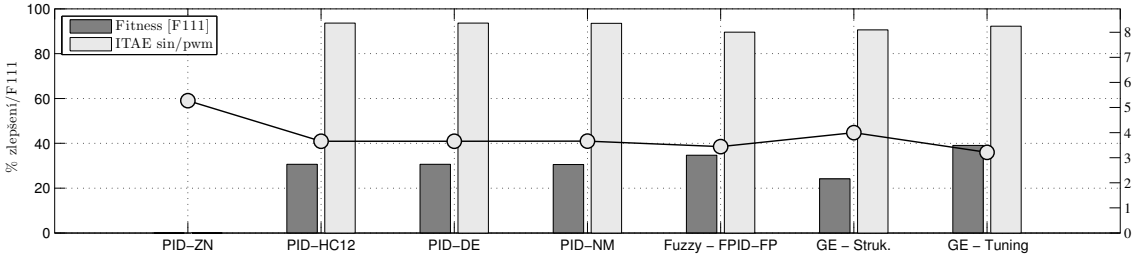



**Graf 8.6:** Hodnoty všech fitness hodnocení [F111] k základnímu nastavení pro výpočet [PID] regulátoru. Pořadí grafů: (a: PID-HC12), (b: PID-DE) a (c: PID-NM).






**Graf 8.7:** Hodnoty všech fitness hodnocení [F111] k základnímu nastavení pro výpočet [FUZZY-GE-Tuning] regulátoru. Pořadí grafů: (a: Fuzzy - FPID-FP), (b: GE - Struk.) a (c: GE - Tuning).




**Graf 8.8:** Hodnoty fitness funkce pro všechny typy regulace a procentuální zlepšení oproti základnímu nastavení regulátoru [model čtvrtého řádu s nelineárním prvkem].

| PID-ZN | | PID-HC12 | | PID-DE | | PID-NM | |
|---|---|---|---|---|---|---|---|
| 5.270 | 8870.571 | 3.654 | 564.028 | 3.656 | 565.847 | 3.661 | 572.356 |
| Fuzzy - FPID-FP | | GE - Struk. | | GE - Tuning | | ⊡ | |
| 3.442 | 922.070 | 3.998 | 834.864 | 3.213 | 685.233 | ⊡ | |
| Nejlepší hodnoty fitness funkce ve formátu: Fitness [F111] \| ITAE SIN/PWM | | | | | | | |

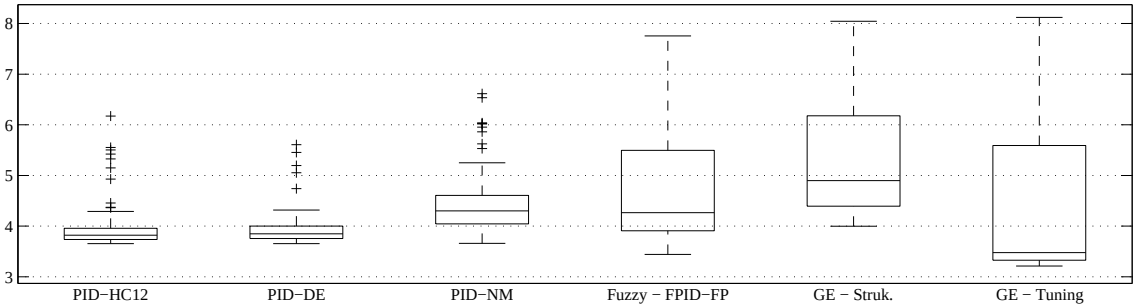
**Tabulka 8.4:** Hodnoty výsledných fitness hodnot ze všech typů regulace [model čtvrtého řádu s nelineárním prvkem].

| PID-ZN | | PID-HC12 | | PID-DE | | PID-NM | |
|---|---|---|---|---|---|---|---|
| 5.270 | 8870.571 | 3.821 | 652.929 | 3.849 | 662.450 | 4.301 | 681.910 |
| Fuzzy - FPID-FP | | GE - Struk. | | GE - Tuning | | ⊡ | |
| 4.267 | 6102.508 | 4.900 | 7590.906 | 3.476 | 797.747 | ⊡ | |
| Medián hodnot fitness funkce ve formátu: Fitness [F111] \| ITAE SIN/PWM | | | | | | | |

**Tabulka 8.5:** Hodnoty mediánů fitness hodnot ze všech typů regulace [model čtvrtého řádu s nelineárním prvkem].

| PID-ZN | | PID-HC12 | | PID-DE | | PID-NM | |
|---|---|---|---|---|---|---|---|
| 0 | | 0.446 | 3290.366 | 0.330 | 3570.528 | 0.616 | 4361.508 |
| Fuzzy - FPID-FP | | GE - Struk. | | GE - Tuning | | ⊡ | |
| 1.183 | 5864.189 | 1.165 | 5533.259 | 1.528 | 3303.873 | ⊡ | |
| STD hodnoty fitness funkce ve formátu: Fitness [F111] \| ITAE SIN/PWM | | | | | | | |

**Tabulka 8.6:** Hodnoty výsledných fitness STD hodnot ze všech typů regulace [model čtvrtého řádu s nelineárním prvkem].

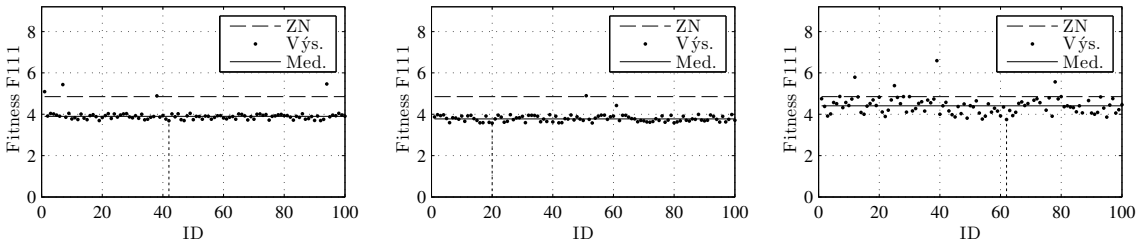



**Graf 8.9:** Box graf hodnot fitness funkce pro všechny typy regulace [model čtvrtého řádu s nelineárním prvkem].

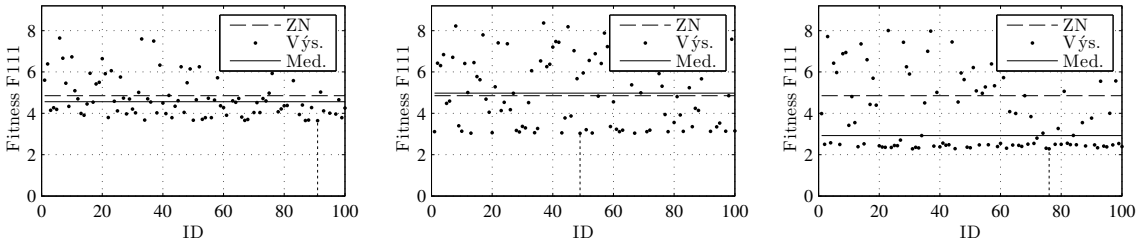
## 8.3 Model kuličky v obruči

Výsledné hodnoty pro model (model čtvrtého řádu s nelineárním prvkem) a jejich grafické znázornění. Graf fitness hodnoty *F111* pro nastavení PID regulátoru ke standartnímu nastavení, algoritmus HC12 (graf: 8.10a), DE (graf: 8.10b) a NM (graf: 8.10c), nastavení fuzzy regulátoru pro typ regulace, *FPID-FP* (graf: 8.11a), výsledky ohodnocení generování vlastní struktury pro GE (graf: 8.11b) následné doladění dle algoritmu HC12 (graf: 8.11c). Grafická reprezentace zlepšení použitých algoritmů oproti základnímu nastavení spolu s nejlepšími hodnotami *F111* fitness

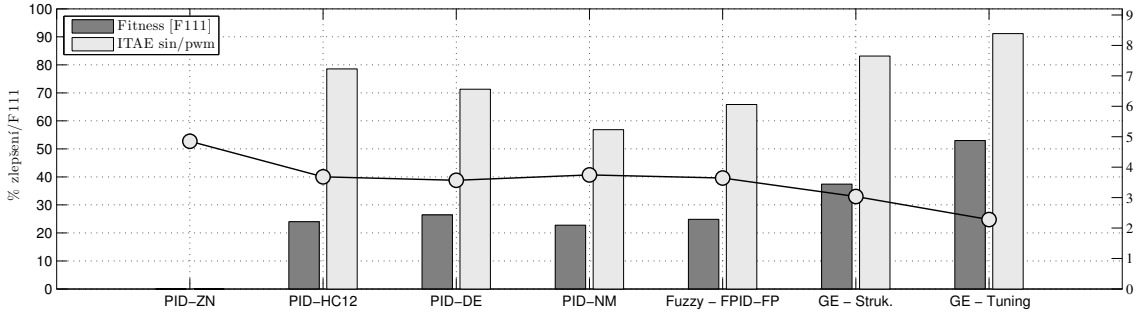
funkce (graf: 8.12). Konečné nejlepší výsledky pro jednotlivé typy regulace (tab: 8.7), medián (tab: 8.8) a standartní odchylka (tab: 8.9) pro všechny opakování algoritmu. Box statistika fitness všech typů regulátorů je v grafu (graf: 8.13).




**Graf 8.10:** Hodnoty všech fitness hodnocení [F111] k základnímu nastavení pro výpočet [PID] regulátoru. Pořadí grafů: (a: PID-HC12), (b: PID-DE) a (c: PID-NM).




**Graf 8.11:** Hodnoty všech fitness hodnocení [F111] k základnímu nastavení pro výpočet [FUZZY-GE-Tuning] regulátoru. Pořadí grafů: (a: Fuzzy - FPID-FP), (b: GE - Struk.) a (c: GE - Tuning).




**Graf 8.12:** Hodnoty fitness funkce pro všechny typy regulace a procentuální zlepšení oproti základnímu nastavení regulátoru [model kuličky v obruči].



| PID-ZN | | PID-HC12 | | PID-DE | | PID-NM | |
|---|---|---|---|---|---|---|---|
| 4.851 | 537.800 | 3.685 | 115.405 | 3.569 | 154.345 | 3.746 | 232.092 |
| Fuzzy - FPID-FP | | GE - Struk. | | GE - Tuning | | ⊡ | |
| 3.644 | 183.737 | 3.036 | 90.536 | 2.280 | 47.656 | ⊡ | |
| Nejlepší hodnoty fitness funkce ve formátu: Fitness [F111] \| ITAE SIN/PWM | | | | | | | |

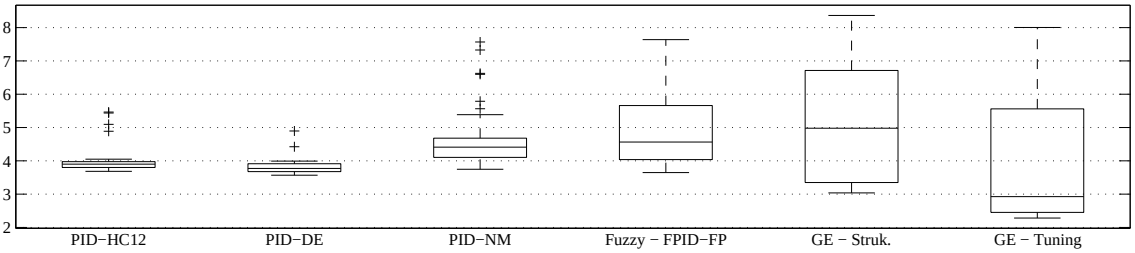
**Tabulka 8.7:** Hodnoty výsledných fitness hodnot ze všech typů regulace [model kuličky v obruči].

| PID-ZN | | PID-HC12 | | PID-DE | | PID-NM | |
|---|---|---|---|---|---|---|---|
| 4.851 | 537.800 | 3.900 | 135.755 | 3.773 | 175.669 | 4.410 | 285.358 |
| Fuzzy - FPID-FP | | GE - Struk. | | GE - Tuning | | ⊡ | |
| 4.562 | 327.765 | 4.978 | 552.588 | 2.925 | 53.804 | ⊡ | |
| Medián hodnot fitness funkce ve formátu: Fitness [F111] \| ITAE SIN/PWM | | | | | | | |

**Tabulka 8.8:** Hodnoty mediánů fitness hodnot ze všech typů regulace [model kuličky v obruči].

| PID-ZN | | PID-HC12 | | PID-DE | | PID-NM | |
|---|---|---|---|---|---|---|---|
| 0 | | 0.286 | 159.112 | 0.182 | 135.335 | 0.640 | 189.633 |
| Fuzzy - FPID-FP | | GE - Struk. | | GE - Tuning | | ⊡ | |
| 1.078 | 274.149 | 1.705 | 325.631 | 1.822 | 124.496 | ⊡ | |
| STD hodnoty fitness funkce ve formátu: Fitness [F111] \| ITAE SIN/PWM | | | | | | | |

**Tabulka 8.9:** Hodnoty výsledných fitness STD hodnot ze všech typů regulace [model kuličky v obruči].

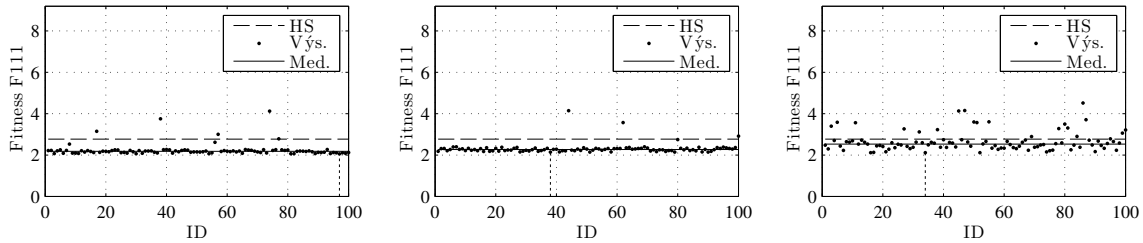



**Graf 8.13:** Box graf hodnot fitness funkce pro všechny typy regulace [model kuličky v obruči].

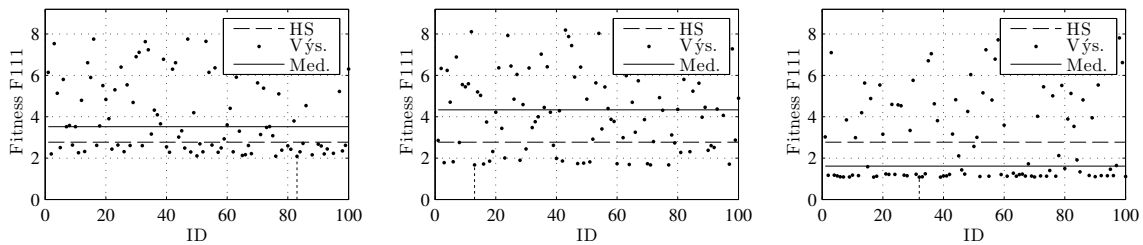
## 8.4 Model magnetické levitace

Výsledné hodnoty a jejich grafické znázornění. Graf fitness hodnoty *F111* pro nastavení PID regulátoru ke standartnímu nastavení, algoritmus HC12 (graf: 8.14a), DE (graf: 8.14b) a NM (graf: 8.14c), nastavení fuzzy regulátoru pro typ regulace, *FPID-FP* (graf: 8.15a), výsledky ohodnocení generování vlastní struktury pro GE (graf: 8.15b) a následné doladění dle algoritmu HC12 (graf: 8.15c). Grafická reprezentace zlepšení použitých algoritmů oproti základnímu nastavení spolu s nejlepšími hodnotami *F111* fitness funkce (graf: 8.16). Konečné nejlepší výsledky pro jednotlivé typy regulace (tab: 8.10), medián (tab: 8.11) a standartní odchylka (tab: 8.12) pro všechny opakování algoritmu. Box statistika fitness všech typů regulátorů (graf: 8.17).

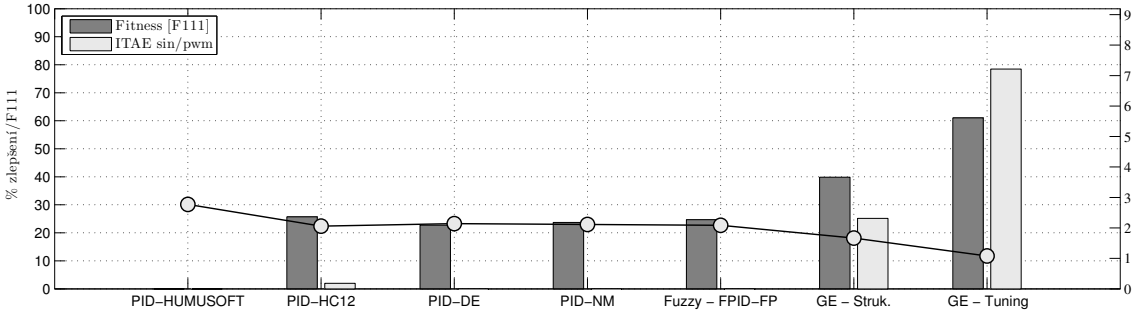





**Graf 8.14:** Hodnoty všech fitness hodnocení [F111] k základnímu nastavení pro výpočet [PID] regulátoru. Pořadí grafů: (a: PID-HC12), (b: PID-DE) a (c: PID-NM).




**Graf 8.15:** Hodnoty všech fitness hodnocení [F111] k základnímu nastavení pro výpočet [FUZZY-GE-Tuning] regulátoru. Pořadí grafů: (a: Fuzzy - FPID-FP), (b: GE - Struk.) a (c: GE - Tuning).




**Graf 8.16:** Hodnoty fitness funkce pro všechny typy regulace a procentuální zlepšení oproti základnímu nastavení regulátoru [model magnetické levitace].

| PID-HUMUSOFT | | PID-HC12 | | PID-DE | | PID-NM | |
|---|---|---|---|---|---|---|---|
| 2.770 | 1.467 | 2.058 | 1.438 | 2.141 | 1.740 | 2.113 | 1.634 |
| Fuzzy - FPID-FP | | GE - Struk. | | GE - Tuning | | ⊡ | |
| 2.087 | 1.540 | 1.667 | 1.098 | 1.080 | 0.316 | ⊡ | |
| Nejlepší hodnoty fitness funkce ve formátu: Fitness [F111] \| ITAE SIN/PWM | | | | | | | |

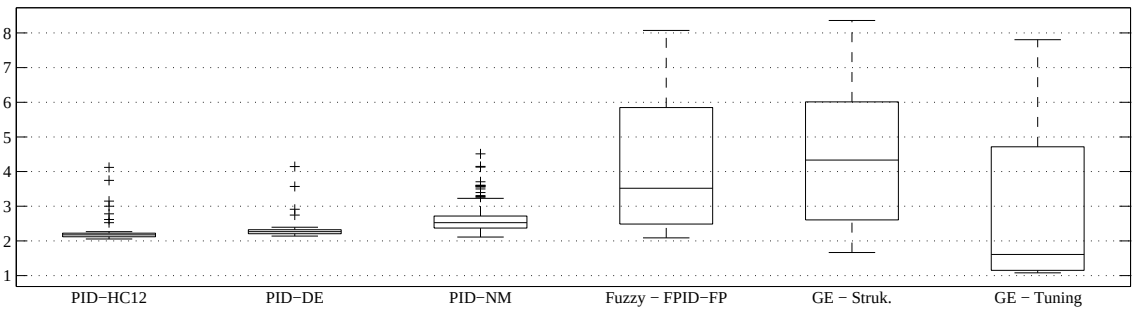
**Tabulka 8.10:** Hodnoty výsledných fitness hodnot ze všech typů regulace [model magnetické levitace].

| PID-HUMUSOFT | | PID-HC12 | | PID-DE | | PID-NM | |
|---|---|---|---|---|---|---|---|
| 2.770 | 1.467 | 2.177 | 1.620 | 2.269 | 1.949 | 2.525 | 1.813 |
| Fuzzy - FPID-FP | | GE - Struk. | | GE - Tuning | | ⊡ | |
| 3.520 | 1.723 | 4.331 | 1.461 | 1.611 | 0.374 | ⊡ | |
| Medián hodnot fitness funkce ve formátu: Fitness [F111] \| ITAE SIN/PWM | | | | | | | |

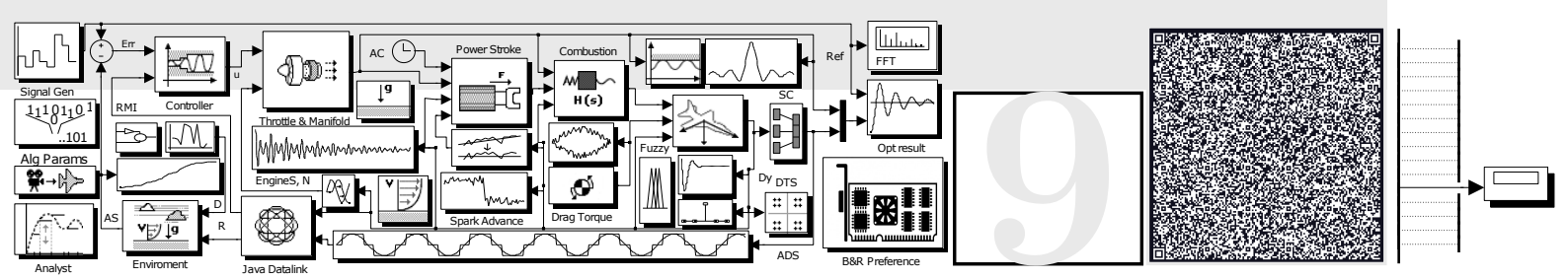
**Tabulka 8.11:** Hodnoty mediánů fitness hodnot ze všech typů regulace [model magnetické levitace].

| PID-HUMUSOFT | | PID-HC12 | | PID-DE | | PID-NM | |
|---|---|---|---|---|---|---|---|
| 0 | | 0.295 | 0.137 | 0.250 | 0.159 | 0.476 | 0.153 |
| Fuzzy - FPID-FP | | GE - Struk. | | GE - Tuning | | ⊡ | |
| 1.900 | 0.203 | 1.998 | 0.364 | 2.123 | 0.392 | ⊡ | |
| STD hodnoty fitness funkce ve formátu: Fitness [F111] \| ITAE SIN/PWM | | | | | | | |

**Tabulka 8.12:** Hodnoty výsledných fitness STD hodnot ze všech typů regulace [model magnetické levitace].




**Graf 8.17:** Box graf hodnot fitness funkce pro všechny typy regulace [model magnetické levitace].




# KAPITOLA 9

# PŘÍKLAD STABILIZACE LOGISTICKÉ MAPY

Sekce stabilizace chaosu se zabývá vytvořením stabilizačních pravidel pro chaotický systém, tedy na logistickou mapu v kontrolním bode $r = 3.8$, popis logistické mapy je v sekci (sekce: 4.2.5 - viz str. 56). Systémy v chaotickém stavu jsou stabilizovány pomoci malých akčních zásahů na aktuálním stavu sytému v iteraci k dosažní požadované hodnoty. Požadovaná hodna je určena dle fixních bodů (UPO), výpočet těchto bodů je znázorněn v grafech (graf: 4.17a) - (graf: 4.17c).



Pro tento příklad regulace jsou (UPO) vypočítány pro logistickou mapu a přesné hodnoty jsou v tabulce (tab: 4.11) pro jeden, dva a čtyři stabilizující orbity. Obecný využívaný způsob stabilizace logistické mapy je metoda zpožděné zpětné vazby (Pyragasova metoda). Pro názorný příklad stabilizace je tedy nastavený akční zásah pomocí Pyragasova metody jako základní regulátor k porovnání výsledků s rovnicemi vygenerovanými dle gramatické evoluce (sekce: 2.4 - viz str. 25). Experiment by měl dokázat využití obecných pravidel dle (GE) pro nastavení základního stabilizačního pravidla jako vhodná alternativa k Pyragasově metodě i s pozdějším doladěním dle metod soft computing, zde metoda (HC12). Základní popis metody je ve vzorci (vz: 9.1), předpokládaný systém $P$ obsahuje proměnou hodnotu $x$ a hodnotu $F$, která značí externí řízený parametr (stabilizátor). Systém bez stabilizace obsahuje hodnotu $F$ rovna 0, tedy bez akčního zásahu.

$$\frac{dx}{dt} = P(x) + F(t) \tag{9.1}$$

Fitness hodnota k optimalizaci stabilizace je počítána pomocí základní funkce

```
        N = { start, weight, expr, expr2, pre_op, op, op2, var, var2}
        T = { +, -, *, /, 1 - 9} | S = { start }

<start> ::(<weight>*<expr><op><expr>)
<weight>::(<weight><op><weight>) | (<var><op><weight>) | (<weight><op><var>)|
          (<var><op><var>)
<expr>  ::<expr> | (<expr><op><expr>) | (<var><op><expr>) | <pre_op><op><expr>|
          <pre_op><op><pre_op>
<pre_op>::x( index_check( n - <expr2> ) )
<expr2> ::(<var2> <op2> <var2>) | <var2>
<op>    ::+ | - | / | *
<op2>   ::+
<var>   ::1 | ...| 9
<var2>  ::1 | <var2>
```

**Gramatika 9.1:** Nastavení gramatiky k regulaci chaosu pro nízký počet bodů (UPO).

```
        N = { start, weight, expr, expr2, pre_op, op, op2, var, var2}
        T = { +, -, *, /, 1 - 9} | S = { start }

<start> ::<weight>*(<weight>*<expr><op><weight>*<expr>)
<weight>::(<weight><op><weight>) | (<var><op><weight>) | (<weight><op><var>)|
          (<var><op><var>)
<expr>  ::<expr> | (<expr><op><expr>) | (<var><op><expr>) | <pre_op><op><expr>|
          <pre_op><op><pre_op>
<pre_op>::x( index_check( n - <expr2> ) )
<expr2> ::(<var2> <op2> <var2>) | <var2>
<op>    ::+ | - | / | *
<op2>   ::+
<var>   ::1 | ...| 9
<var2>  ::1 | <var2>
```

**Gramatika 9.2:** Nastavení gramatiky k regulaci chaosu pro 4 body (UPO).

(IAE) v časové (iterační) oblasti (vz: 4.2) pro jeden stabilizující orbit. K vice orbitů je využita obdobně (ITAE) metoda, ale s úpravou pro vice požadovaných hodnot s atraktorem hodnoty, viz princip dle algoritmu (alg: 6).



Stabilizace chaosu v tomto případě je rozdělena na tři části. První část je stabilizace dle Pyragasové metody s využitím optimalizační metody (HC12), druhá část je vytvoření vlastních pravidel pomoci gramatické evoluce s využitím základních hodnot regulátoru (TDAS)/(ETDAS) a poslední část je doladění předchozí části dle algoritmu (HC12). Pro vytváření vlastních stabilizačních pravidel jsou navrženy dvě základní verze gramatik. První verze je znázorněna na popisu (gramatika: 9.1) a hodí se k regulovaní jen nízkého počtu stabilizujících (UPO), zde je používána na jeden a dva fixní body, gramatika generuje pravidla, která pro další optimalizaci zahrnují celkem tři základní body a to $F_{max}$, $X_1$ a vytvořený $param_1$. Druha verze gramatiky je opravená první verze s přidáním vice optimalizujících parametrů na $param_1$, $param_2$ a $param_3$ a je popsaná dle (gramatika: 9.2). Sekce generovaní pravidel obsahuje jak formu po vygenerovaní pravidla, tak formu po úpravě k další optimalizaci dle (HC12) algoritmu.

## 9.1 Pyragasova metoda

Metoda zpožděné zpětné vazby aplikuje drobné akční zásahy na řízenou hodnotu k dosažení hledaného výsledku. Tato metoda je taky známa pod názvem Time Delayed Auto Synchronization (TDAS), metoda je velice jednoduchá na implementaci a její využití je velmi vhodné pro stabilizaci nízkého počtu stabilizujících orbitů. V této práci je metoda (TDAS) využívána na stabilizaci jednoho orbitu. Stabilizační zákon pro (UPO) s periodou $\tau$ s logickou podmínkou (vz: 9.2) $F$ je zobrazen na vzorci (vz: 9.3). Hodnota $K$ určuje velikost akčního zásahu. Ke stabilizaci chaosu je zapotřebí jen drobných akčních zásahů, a proto je zavedena do logické struktury regulátoru i podmínka na maximální zásah stabilizační metody $F_{max}$. Poslední hodnotu optimalizace je počáteční hodnota parametru $x$, která může nabývat hodnot v definiční oblasti chaotického chování, v našem případě logistické mapy je v rozmezí 0 až 1. Tedy optimalizované parametry metody (TDAS) jsou $K$, $F_{max}$ a $X_1$ a k samotné optimalizaci je využita metoda (HC12).

$$x(t+\tau) = x(t) \tag{9.2}$$

$$F(t) = K[x(t-\tau) - x(t)] \tag{9.3}$$

Ke stabilizaci více orbitů je nutno rozšířit původní metodu TDAS na metodu Extended Time Delayed Auto Synchronization (ETDAS). Metoda zavadí nový parametr $S$ dle rovnice (vz: 9.4), který pracuje s předchozími stavy systému k dosažení kvalitnější regulace. Upravený výsledný regulátor $F$ je znázorněný v rovnici (vz: 9.5). Metoda (ETDAS) přidává nový nastavitelný parametr $R$ k optimalizaci.

$$S(t) = x(t) + RS(t-\tau)] \tag{9.4}$$

$$F(t) = K[(1-R)S(t-\tau) - x(t)] \tag{9.5}$$

Předchozí rovnice k metodám (TDAS) a (ETDAS) byly zapsány pro spojité systémy a pro potřeby využití těchto metod v diskrétních systémech je nutné upravit jejich zápis do diskrétní oblasti. Regulovaný systém je diskrétně zapsán (vz: 9.6), pro metodu (TDAS) je diskrétní verze zapsaná dle vzorce (vz: 9.7) a pro verzi (ETDAS) je dle vzorce (vz: 9.8) a (vz: 9.9).

$$x_{n+1} = P(x_n) + Fn \tag{9.6}$$

$$F_n = K[x_{n-m} - x_n] \tag{9.7}$$

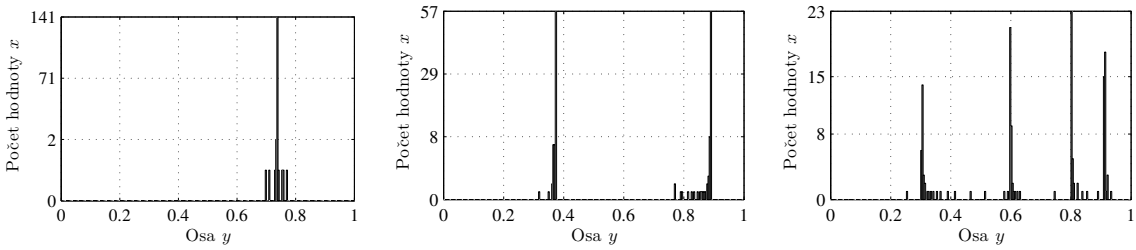
$$F_n = K[(1-R)S_{n-m} - x_n] \tag{9.8}$$

$$S_n = x_n + RS_{n-m} \tag{9.9}$$

Kde parametr $n$ je aktuální iterace systému a parametr $m$ je požadovaný řád stabilizace (UPO).

## 9.2 Optimalizace Pyragasovy metody dle HC12

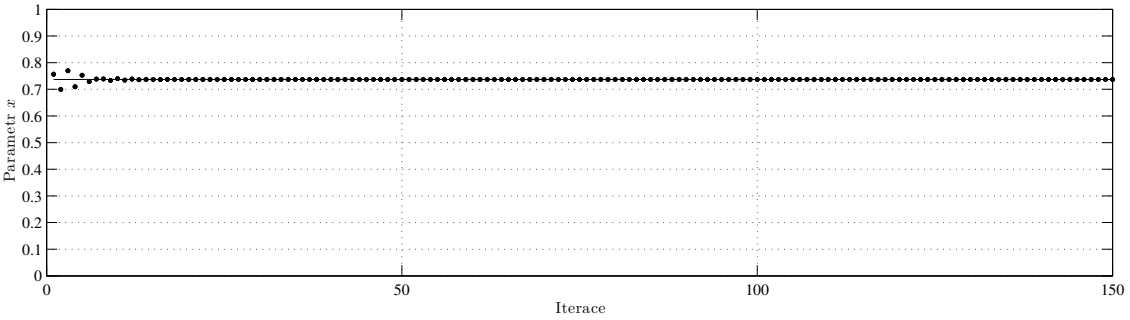
Natavení pravidel ke stabilizaci chaosu dle Pyragasovy metody, konkrétně diskrétní verze metody (TDAS) (vz: 9.7) a metody (ETDAS) (vz: 9.8). Optimalizační metoda je algoritmus (HC12). Výsledky optimalizace parametrů jsou zobrazeny na následujících tabulkách spolu s nastavením optimalizačního algoritmu a ohodnocením výsledku dle funkce (ITAE). Tabulka pro 1UPO (tab: 9.1), pro 2UPO (tab: 9.2) a 4UPO (tab: 9.3). Výsledky konečného nastavení systému se stabilizační metodou jsou zobrazeny pro 1UPO (graf: 9.2), 2UPO (graf: 9.3) a 4UPO (graf: 9.4).




**Graf 9.1:** Rozložení polohy regulované hodnoty $x$ pro nastavení [ETDAS] v logistické mapě. Pořadí grafů: (a: UPO1), (b: UPO2) a (c: UPO4).

| ⊞ | $X_1$ | K | $F_{max}$ | R | Fit. |
|---|---|---|---|---|---|
| Výs. [TDAS-A] | 0.757 | -0.508 | 0.189 | - | 16.989 |
| Výs. [TDAS-B] | | | | | 18.009 |
| Výs. [TDAS-C] | | | | | 23.318 |
| P:(tab: 4.1) | B:<0.0,1.0,10> | B:<-2.0,2.0,10> | B:<0.0,1.0,10> | | ⊡ |
| AI:(tab: 4.2) | M12 [Default], RUNS [100] | | | | |

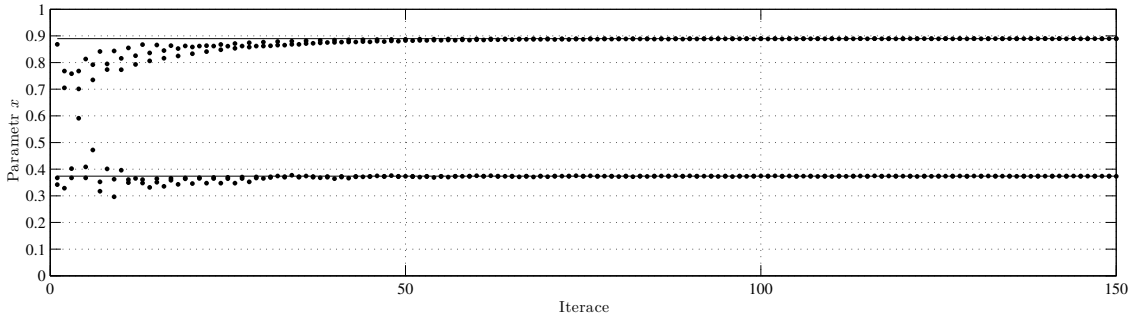
**Tabulka 9.1:** Tabulka nastavení regulátoru [Regulátor Stabilizace-chaosu Alg HC12-TDAS1] pro Logisticka-mapa.




**Graf 9.2:** Řízená logistická mapa v parametru r (3.8) pro 1UPO Alg HC12-TDAS1.

| ⊞ | $X_1$ | K | $F_{max}$ | R | Fit. |
|---|---|---|---|---|---|
| Výs. [ETDAS-A] | 0.367 | 0.893 | 0.619 | 0.795 | 30.378 |
| Výs. [ETDAS-B] | 0.342 | 1.041 | 0.775 | 0.831 | 33.443 |
| Výs. [ETDAS-C] | 0.869 | 1.051 | 0.211 | 0.853 | 34.775 |
| P:(tab: 4.1) | B:<0.0,1.0,10> | B:<-2.0,2.0,10> | B:<0.0,1.0,10> | | ⊡ |
| AI:(tab: 4.2) | M12 [Default], RUNS [100] | | | | |

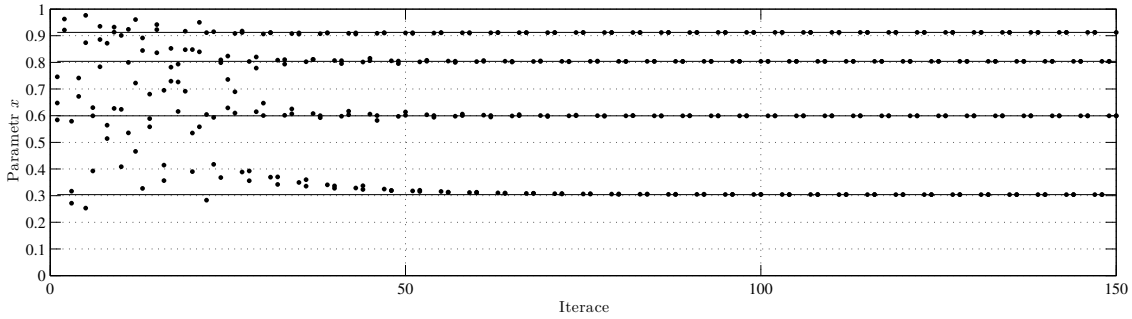
**Tabulka 9.2:** Tabulka nastavení regulátoru [Regulátor Stabilizace-chaosu Alg HC12-ETDAS2] pro Logisticka-mapa.




**Graf 9.3:** Řízená logistická mapa v parametru r (3.8) pro 2UPO Alg HC12-ETDAS2.

| ⊞ | $X_1$ | K | $F_{max}$ | R | Fit. |
|---|---|---|---|---|---|
| Výs. [ETDAS-A] | 0.746 | -0.586 | 0.962 | 0.725 | 37.955 |
| Výs. [ETDAS-B] | 0.584 | -0.548 | 0.878 | 0.666 | 38.281 |
| Výs. [ETDAS-C] | 0.647 | -0.533 | 0.360 | 0.634 | 39.682 |
| P:(tab: 4.1) | B:<0.0,1.0,10> | B:<-2.0,2.0,10> | B:<0.0,1.0,10> | | ⊡ |
| AI:(tab: 4.2) | M12 [Default], RUNS [100] | | | | |

**Tabulka 9.3:** Tabulka nastavení regulátoru [Regulátor Stabilizace-chaosu Alg HC12-ETDAS4] pro Logisticka-mapa.




**Graf 9.4:** Řízená logistická mapa v parametru r (3.8) pro 4UPO Alg HC12-ETDAS4.

Grafické znázornění rozložení výsledné regulované veličiny $x$ prezentuje graf pro jeden fixní bod (graf: 9.1a), dva fixní body (graf: 9.1b) a čtyři fixní body (graf: 9.1c). Výsledky ukazují vhodnost využití Pyragosovy vety ke stabilizaci logistické mapy.



## 9.3 Hledání vlastních pravidel dle GE

Generování vlastních pravidel ke stabilizaci chaotického systému. Optimalizační metoda je algoritmus (GE) s aktivní gramatikou (gramatika: 9.1) a (gramatika: 9.2). Výsledky optimalizace parametrů jsou zobrazeny na následujících tabulkách spolu s nastavením optimalizačního algoritmu a ohodnocením výsledku dle funkce (ITAE). Tabulka pro 1UPO (tab: 9.4), pro 2UPO (tab: 9.5) a 4UPO (tab: 9.6). Výsledky konečného nastavení systému stabilizační metody jsou zobrazeny pro 1UPO (graf: 9.5), 2UPO (graf: 9.6) a 4UPO (graf: 9.8). Grafické znázornění rozložení výsledné veličiny $x$ prezentuje graf pro jeden fixní bod (graf: 9.7a), dva fixní body (graf: 9.7b) a čtyři fixní body (graf: 9.7c). Generované struktury stabilizující funkce k logistické mapě můžou nabývat mnoha podob, závislé na druhu použité gramatické definice,

```
expression = ((1/((4*((((9-8)+3)-4)+(8*(5+9))))+8))*((5/(8*x(index-check(n-(((
1+(1+(1+1)))+1)+1)))-x(index-check(n-(1+1)))+x(index-check(n-1))))*x(index-che
ck(n-((1+(1+(1+1)))+1)))*x(index-check(n-1)))-x(index-check(n-(1+1)))+x(index-
check(n-1)));
```

**GE výsledek 9.1:** Vypočtený výraz dle GE gramatiky [Typ Version-A] pro Logisticka-mapa.

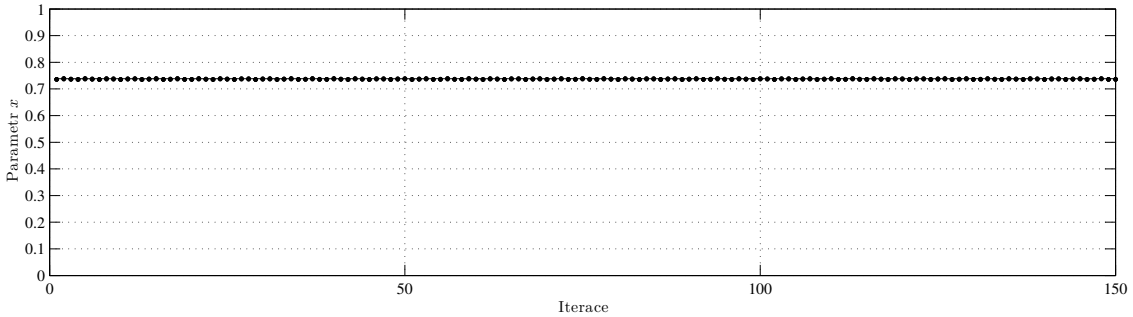
```
expression = (param-1*((5/(8*x(n-5)-x(n-2)+x(n-1)))*x(n-5)*x(n-1))-x(n-2)+x(n-
1));
```

**GE výsledek 9.2:** Upravený GE výraz pro následnou optimalizaci [Typ Version-A] pro Logisticka-mapa.

zde prezentované výsledky jsou jen jedna z možností využiti (GE) na stabilizaci chaotického systému. Funkce *index − check* v nalezeném řešení je funkce ke kontrole rozsahu použitých indexů a jejich kladných hodnot. Rozsah generovaného řešení udává nastavení gramatické evoluce.

| ⊞ | Výsledek | Fit. |
|---|---|---|
| Výs. [Version-A] | GE výsledek (expr: 9.1) a GE výsledek (expr: 9.2) | 6.064 |
| Výs. [Version-B] | GE výsledek (expr: 9.3) a GE výsledek (expr: 9.4) | 6.869 |
| Výs. [Version-C] | GE výsledek (expr: 9.5) a GE výsledek (expr: 9.6) | 8.078 |
| P:(tab: 4.1) | C:<40,8> | ⊡ |
| AI:(tab: 4.2) | NP [500], F [0.2], R [0.5], PC [2], W [10], GEN [200], RUNS [100] | |

**Tabulka 9.4:** Tabulka nastavení regulátoru [Regulátor Stabilizace-chaosu Alg GE-UPO1] pro Logisticka-mapa.




**Graf 9.5:** Řízená logistická mapa v parametru r (3.8) pro 1UPO Alg GE-UPO1.

| ⊞ | Výsledek | Fit. |
|---|---|---|
| Výs. [Version-A] | GE výsledek (expr: 9.7) a GE výsledek (expr: 9.8) | 26.192 |
| Výs. [Version-B] | GE výsledek (expr: 9.9) a GE výsledek (expr: 9.10) | 29.167 |
| Výs. [Version-C] | GE výsledek (expr: 9.11) a GE výsledek (expr: 9.12) | 30.043 |
| P:(tab: 4.1) | C:<40,8> | ⊡ |
| AI:(tab: 4.2) | NP [500], F [0.2], R [0.5], PC [2], W [10], GEN [200], RUNS [100] | |

**Tabulka 9.5:** Tabulka nastavení regulátoru [Regulátor Stabilizace-chaosu Alg GE-UPO2] pro Logisticka-mapa.

```
expression = (((((7-(7/7))/(7/(((2/3)-((9+(3+1))+(4/1)))-(1+2))))*1)/((7-(7/7)
)/(7/(((2/3)-((9+(3+1))+(4/1)))-(1+2)))))*x(index-check(n-(1+1)))*x(index-chec
k(n-1))/x(index-check(n-(1+1)))-x(index-check(n-(1+1))));
```

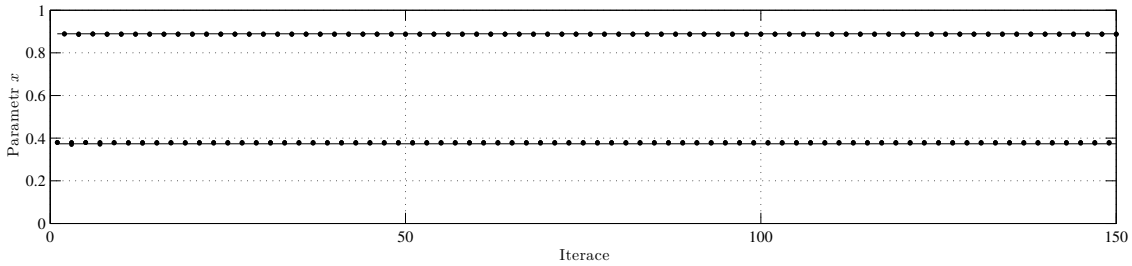
**GE výsledek 9.3:** Vypočtený výraz dle GE gramatiky [Typ Version-B] pro Logisticka-mapa.

```
expression = (param-1*x(n-2)*x(n-1)/x(n-2)-x(n-2));
```

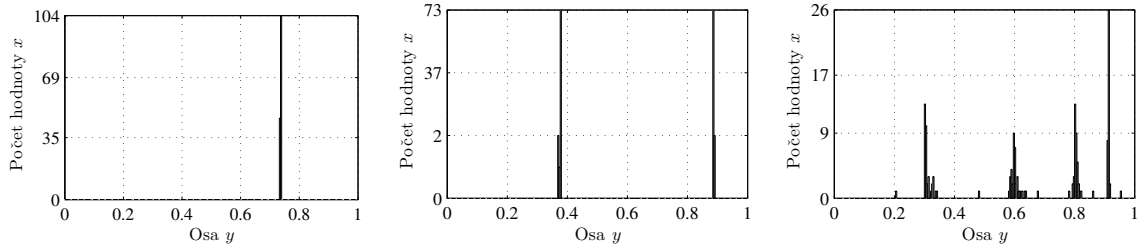
**GE výsledek 9.4:** Upravený GE výraz pro následnou optimalizaci [Typ Version-B] pro Logisticka-mapa.

```
expression = (((((2-(6/(9-8)))+7)/(((3-5)-(((3+3)+(9/1))-9))*9))/(8+9))*x(inde
x-check(n-1))*x(index-check(n-(1+1)))+x(index-check(n-1))-x(index-check(n-(1+1
))));
```

**GE výsledek 9.5:** Vypočtený výraz dle GE gramatiky [Typ Version-C] pro Logisticka-mapa.



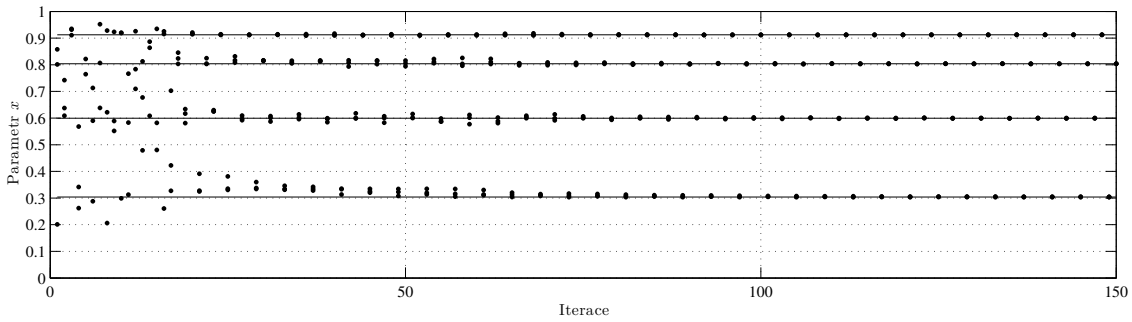

**Graf 9.6:** Řízená logistická mapa v parametru r (3.8) pro 2UPO Alg GE-UPO2.




**Graf 9.7:** Rozložení polohy regulované hodnoty $x$ pro nastavení [GE-Struk] v logistické mapě. Pořadí grafů: (a: UPO1), (b: UPO2) a (c: UPO4).

| ⊞ | Výsledek | Fit. |
|---|---|---|
| Výs. [Version-A] | GE výsledek (expr: 9.13) a GE výsledek (expr: 9.14) | 31.364 |
| Výs. [Version-B] | GE výsledek (expr: 9.15) a GE výsledek (expr: 9.16) | 33.214 |
| Výs. [Version-C] | GE výsledek (expr: 9.17) a GE výsledek (expr: 9.18) | 33.884 |
| P:(tab: 4.1) | C:<40,8> | ⊡ |
| AI:(tab: 4.2) | NP [500], F [0.2], R [0.5], PC [2], W [10], GEN [200], RUNS [100] | |

**Tabulka 9.6:** Tabulka nastavení regulátoru [Regulátor Stabilizace-chaosu Alg GE-UPO4] pro Logisticka-mapa.




**Graf 9.8:** Řízená logistická mapa v parametru r (3.8) pro 4UPO Alg GE-UPO4.

```
expression = (param-1*x(n-1)*x(n-2)+x(n-1)-x(n-2));
```

**GE výsledek 9.6:** Upravený GE výraz pro následnou optimalizaci [Typ Version-C] pro Logisticka-mapa.

```
expression = ((3-(((4-(6+(8-9)))*(4/3))+7))*(x(index-check(n-(1+(1+1))))*x(ind
ex-check(n-1))+x(index-check(n-1))+x(index-check(n-1))*((x(index-check(n-1))/x
(index-check(n-(1+1)))-x(index-check(n-(1+(1+1))))-(2-(x(index-check(n-1))*x(i
ndex-check(n-1))*(4/x(index-check(n-(1+1)))-x(index-check(n-(1+(1+1)))))))-(2
-(x(index-check(n-1))*x(index-check(n-1))*(4/x(index-check(n-(1+1)))-x(index-c
heck(n-(1+(1+1)))))))-(2-(x(index-check(n-1))*x(index-check(n-1))*(4/x(index
-check(n-(1+1)))-x(index-check(n-(1+(1+1)))))));
```

**GE výsledek 9.7:** Vypočtený výraz dle GE gramatiky [Typ Version-A] pro Logisticka-mapa.

```
expression = (param-1*(x(n-3)*3*x(n-1)*((x(n-1)/x(n-2)-x(n-3)-(2-(x(n-1)*x(n-1
)*(4/x(n-2)-x(n-3)))))-(2-(x(n-1)*x(n-1)*(4/x(n-2)-x(n-3))))))-(2-(x(n-1)*x(n-
1)*(4/x(n-2)-x(n-3)))));
```

**GE výsledek 9.8:** Upravený GE výraz pro následnou optimalizaci [Typ Version-A] pro Logisticka-mapa.

```
expression = ((7/4)*x(index-check(n-1))*x(index-check(n-1))-x(index-check(n-(1
+1)))-x(index-check(n-(1+(1+1))))/x(index-check(n-(1+(1+1)))));
```

**GE výsledek 9.9:** Vypočtený výraz dle GE gramatiky [Typ Version-B] pro Logisticka-mapa.

```
expression = (param-1*x(n-1)*x(n-1)-x(n-2)-x(n-3)/x(n-3));
```

**GE výsledek 9.10:** Upravený GE výraz pro následnou optimalizaci [Typ Version-B] pro Logisticka-mapa.

```
expression = ((3/4)*x(index-check(n-1))-x(index-check(n-(1+1)))/x(index-check(
n-(1+1)))+x(index-check(n-1))*x(index-check(n-(1+1))));
```

**GE výsledek 9.11:** Vypočtený výraz dle GE gramatiky [Typ Version-C] pro Logisticka-mapa.

```
expression = (param-1*x(n-1)-x((n-2))/x(n-2)+x(n-1)*x(n-2));
```

**GE výsledek 9.12:** Upravený GE výraz pro následnou optimalizaci [Typ Version-C] pro Logisticka-mapa.

```
expression = (((1+(8-2))+1)/4)*(((1/7)/(4+(((((((2*2)*7)+1)/9)*2)/2)-(((8-2)+(
4+7))*(((3-4)+2)/3)))))*(1*x(index-check(n-1))-x(index-check(n-2)))*(2/2)*x(in
dex-check(n-6))/((1/x(index-check(n-1))/(3-x(index-check(n-5))/(1*x(index-chec
k(n-1))-x(index-check(n-2))))*x(index-check(n-8))+x(index-check(n-8))));
```

**GE výsledek 9.13:** Vypočtený výraz dle GE gramatiky [Typ Version-A] pro Logisticka-mapa.

```
expression = param-1*(param-2*(1*x(n-1)-x(n-2))*param-3*x(n-6)/((1/x(n-1)/(3-x
(n-5)/(1*x(n-1)-x(n-2))))*x(n-8)+x(n-8)));
```

**GE výsledek 9.14:** Upravený GE výraz pro následnou optimalizaci [Typ Version-A] pro Logisticka-mapa.

```
expression = (((4-(1*(1/(9*8))))+3)/((4+6)+8))*((((5*9)/(1-(8+8)))+(((4-(1*(1/
(9*8))))+3)/((4+6)+8)))*(x(index-check(n-9))/x(index-check(n-2))*x(index-check
(n-4))-x(index-check(n-1))*x(index-check(n-1))+x(index-check(n-4)))/(((4+6)+8)
+3)*x(index-check(n-9))/x(index-check(n-2)));
```

**GE výsledek 9.15:** Vypočtený výraz dle GE gramatiky [Typ Version-B] pro Logisticka-mapa.

```
expression = param-1*(param-2*(x(n-9)/x(n-2)*x(n-4)-x(n-1)*x(n-1)+x(n-4))/para
m-3*x(n-9)/x(n-2));
```

**GE výsledek 9.16:** Upravený GE výraz pro následnou optimalizaci [Typ Version-B] pro Logisticka-mapa.

```
expression = ((1+1)/4)*(((1-((((9/((((6-(3+5))/4)+4)-(4+8)))-(3*1))*1)+1))/4)*
x(index-check(n-9))/x(index-check(n-9))/((((6-(3+5))/4)+4)-(4+8))*x(index-chec
k(n-4))/x(index-check(n-2)));
```

**GE výsledek 9.17:** Vypočtený výraz dle GE gramatiky [Typ Version-C] pro Logisticka-mapa.

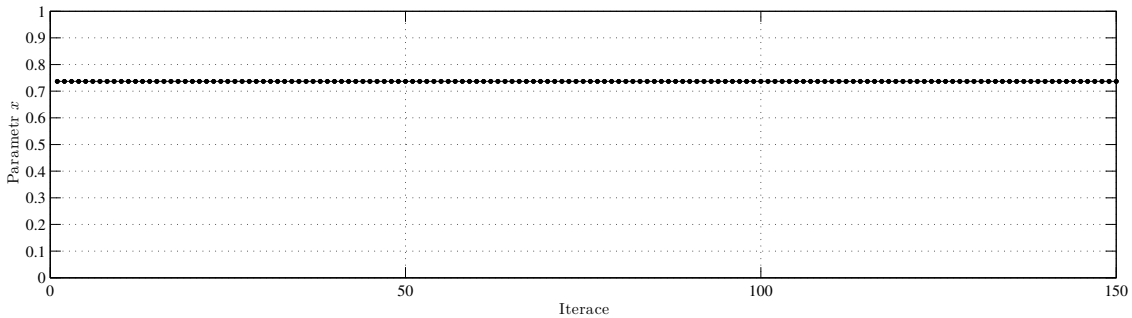
## 9.4 Doladění vlastních pravidel dle HC12

Poslední zde prezentovaná metoda na stabilizaci chaotického systému, vycházející ze znalosti aplikovaných v kapitole generovaní struktury regulátoru(sekce: 7 - viz str. 93) a používá stejný postup při nalezení optimálního řešení vygenerovaných rovnic pomocí (GE) algoritmu. Tato sekce se tedy zabývá ladění výsledné struktury po optimalizaci gramatickou evolucí z předešlé sekce. Optimalizační metoda je algoritmus (HC12). Výsledky optimalizace parametrů jsou zobrazeny na následujících tabulkách spolu s nastavením optimalizačního algoritmu a ohodnocením výsledku dle funkce (ITAE). Tabulka pro 1UPO (tab: 9.7), pro 2UPO (tab: 9.8) a 4UPO (tab: 9.9). Výsledky konečného nastavení systému se stabilizací jsou zobrazeny pro 1UPO (graf: 9.9), 2UPO (graf: 9.11) a 4UPO (graf: 9.12). Grafické znázornění rozložení výsledné regulované veličiny $x$ prezentuje graf pro jeden fixní bod (graf: 9.10a), dva fixní body (graf: 9.10b) a čtyři fixní body (graf: 9.10c). Výsledky prezentují optimální nastavení algoritmu.

| ⊞ | $X_1$ | $F_{max}$ | $Param_1$ | Fit. |
|---|---|---|---|---|
| Výs. [Tuning-A] | 0.737 | 0.290 | -2.442E-4 | 4.487 |
| Výs. [Tuning-B] | | 0.275 | 1 | 5.512 |
| Výs. [Tuning-C] | | 0.018 | -6.190E-4 | 5.935 |
| P:(tab: 4.1) | B:<0.0,1.0,10> | B:<-2.0,2.0,10> | B:<-5.0,-5.0,12> | ⊡ |
| AI:(tab: 4.2) | M12 [Default], RUNS [100] | | | |

**Tabulka 9.7:** Tabulka nastavení regulátoru [Regulátor Stabilizace-chaosu Alg HC12-TUNING-UPO1] pro Logisticka-mapa.

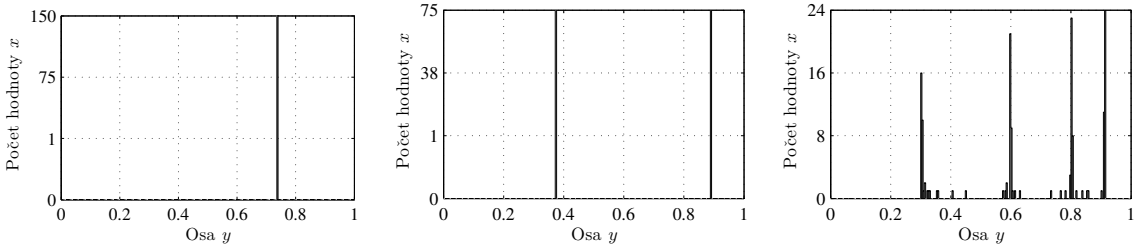





**Graf 9.9:** Řízená logistická mapa v parametru r (3.8) pro 1UPO Alg HC12-TUNING-UPO1.

```
expression = param-1*(param-2*x(n-9)/x(n-9)/param-3*x(n-4)/x(n-2));
```

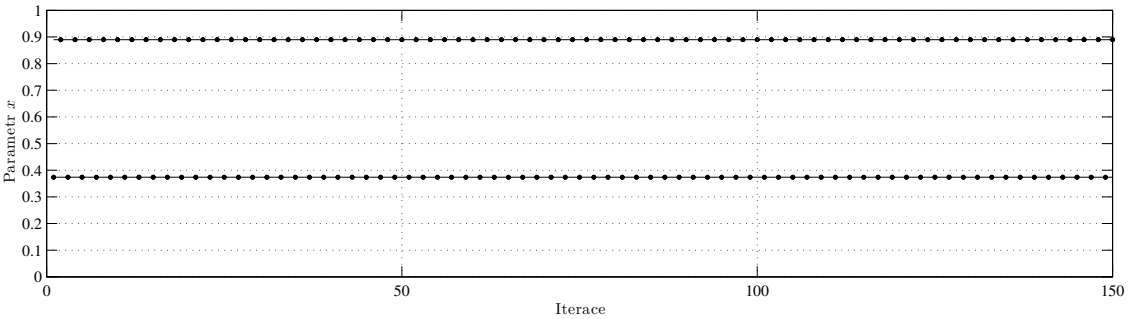
**GE výsledek 9.18:** Upravený GE výraz pro následnou optimalizaci [Typ Version-C] pro Logisticka-mapa.




**Graf 9.10:** Rozložení polohy regulované hodnoty $x$ pro nastavení [GE-Tuning] v logistické mapě. Pořadí grafů: (a: UPO1), (b: UPO2) a (c: UPO4).

| ⊞ | $X_1$ | $F_{max}$ | $Param_1$ | Fit. |
|---|---|---|---|---|
| Výs. [Tuning-A] | 0.374 | 1.899E-4 | -2.669 | 12.043 |
| Výs. [Tuning-B] | | 2.271E-4 | 1.737 | 14.428 |
| Výs. [Tuning-C] | | 1.519E-4 | 0.751 | 15.941 |
| P:(tab: 4.1) | B:<0.0,1.0,10> | B:<-2.0,2.0,10> | B:<-5.0,5.0,12> | ⊡ |
| AI:(tab: 4.2) | M12 [Default], RUNS [100] | | | |

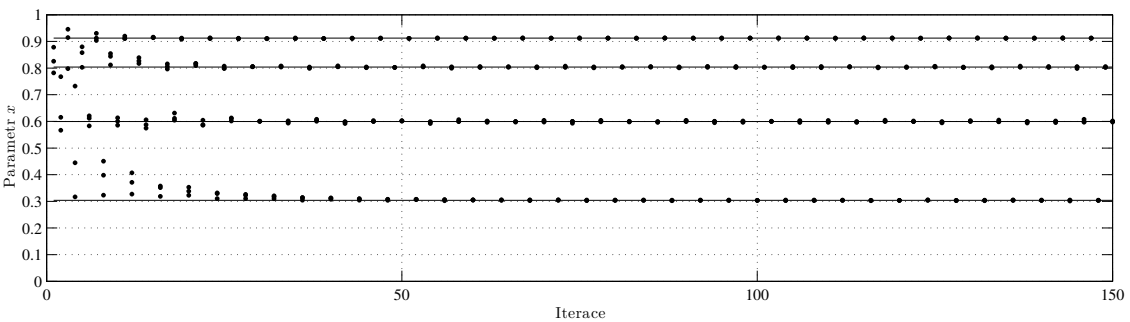
**Tabulka 9.8:** Tabulka nastavení regulátoru [Regulátor Stabilizace-chaosu Alg HC12-TUNING-UPO2] pro Logisticka-mapa.




**Graf 9.11:** Řízená logistická mapa v parametru r (3.8) pro 2UPO Alg HC12-TUNING-UPO2.

| ⊞ | $X_1$ | $F_{max}$ | $Param_1$ | $Param_2$ | $Param_3$ | Fit. |
|---|---|---|---|---|---|---|
| Výs. [Tuning-A] | 0.229 | 0.077 | -0.231 | 0.022 | -32.324 | 21.663 |
| Výs. [Tuning-B] | 0.404 | 0.463 | 0.030 | 26.434 | -16.077 | 22.273 |
| Výs. [Tuning-C] | 0.969 | 0.112 | 35.654 | 43.886 | -17.082 | 24.926 |
| P:(tab: 4.1) | B:<0.0,1.0,10> | | B:<-50.0,50.0,15> | | | ⊡ |
| AI:(tab: 4.2) | M12 [Default], RUNS [100] | | | | | |

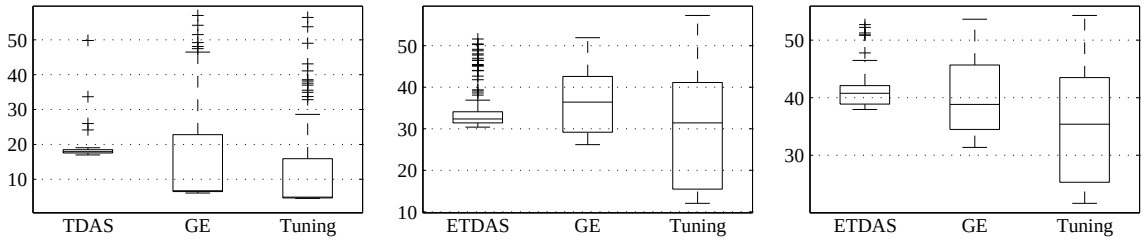
**Tabulka 9.9:** Tabulka nastavení regulátoru [Regulátor Stabilizace-chaosu Alg HC12-TUNING-UPO4] pro Logisticka-mapa.




**Graf 9.12:** Řízená logistická mapa v parametru r (3.8) pro 4UPO Alg HC12-TUNING-UPO4.

## 9.5 Porovnání jednotlivých výsledků

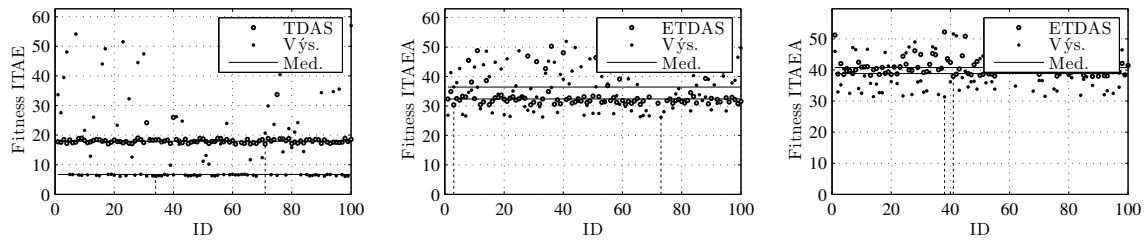
V této sekci porovnání výsledků je shrnuto využití jednotlivých algoritmů použitých v této práci na stabilizaci deterministického chaosu a strukturou vyhází z kapitoly (sekce: 8 - viz str. 107). Rozdělení jednotlivých výsledků a jejich grafická analýza pomocí box grafů je prezentována pro výsledky fitness funkce (graf: 9.13a), (graf: 9.13b) a (graf: 9.13c). Optimalizace stability deterministického chaosu je složitá procedura a závisí na samotné chaosové funkci a funkci použité k výsledné stabilizaci. Standartní požívaná funkce ke stabilizaci chaosu je považována Pyragasova metoda (sekce: 9.1 - viz str. 117) ve formátu (TDAS) pro jeden (UPO) fixní bod nebo (ETDAS) pro více (UPO) fixních bodů, proto je tato metoda požitá jako referenční nastavení dle (HC12) algoritmu ze sekce (sekce: 9.2 - viz str. 118). V této práci je hlavně rozvíjená možnost stabilizace chaosu dle gramatické evoluce se speciálně nastavenou gramatikou stabilizace (gramatika: 9.1) a (gramatika: 9.2), samotný výpočet a výsledky je dle (sekce: 9.3 - viz str. 119). Grafické zobrazení výsledků pro všechny typy (UPO) stabilizace je dle grafů (graf: 9.14a), (graf: 9.14b) a (graf: 9.14c). Následující doladění dle metody (HC12) v (sekce: 9.4 - viz str. 123) na vygenerovaných funkčních pravidel z předešlé části je reprezentované grafech (graf: 9.15a), (graf: 9.15b) a (graf: 9.15c).



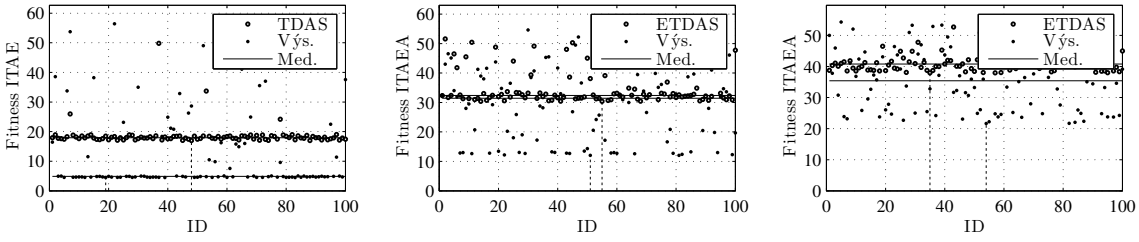

**Graf 9.13:** Box grafy pro hodnoty fitness. Pořadí grafů: (a: UPO1), (b: UPO2) a (c: UPO4).

Tento postup využití gramatické evoluce a následné požití dolaďovacích postupů dle (HC12) se velmi dobře uplatnil v předešlé části k regulaci nelineárních systémů a pro stabilizaci deterministického chaosu také vykazuje velice dobré výsledky, jak prezentuje graf (graf: 9.16), který zobrazuje jednotlivé procentuální zlepšení pro všechny (UPO) fixní body k základnímu nastavení (Pyragasova metoda dle algoritmu HC12) pro gramatickou evoluci a následné doladění dle (HC12) algoritmu. Graf také zobrazuje jednotlivé nejlepší ohodnocení pro fitness (vz: 4.2) nebo speciální ITAE funkce (alg: 6) pro více fixních bodů.

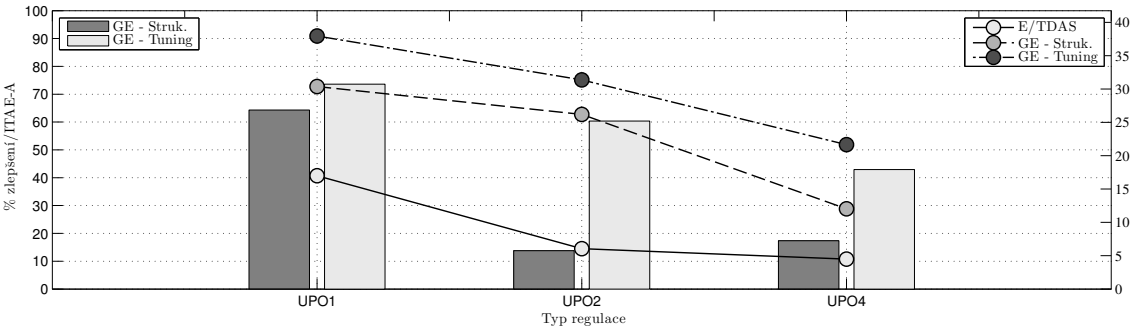
Výsledné hodnoty jsou v tabulce (tab: 9.10) pro nejlepší hodnoty fitness, medián a standartní odchylku rozptylu pro všechna opakování výpočtů.




**Graf 9.14:** Hodnoty všech fitness hodnocení [ITAE/ITAEA] k základnímu nastavení pro výpočet [GE] regulátoru. Pořadí grafů: (a: UPO1 - TDAS - GE), (b: UPO2 - ETDAS - GE) a (c: UPO4 - ETDAS - GE).




**Graf 9.15:** Hodnoty všech fitness hodnocení [ITAE/ITAEA] k základnímu nastavení pro výpočet [Tuning] regulátoru. Pořadí grafů: (a: UPO1 - TDAS - Tuning), (b: UPO2 - ETDAS - Tuning) a (c: UPO4 - ETDAS - Tuning).



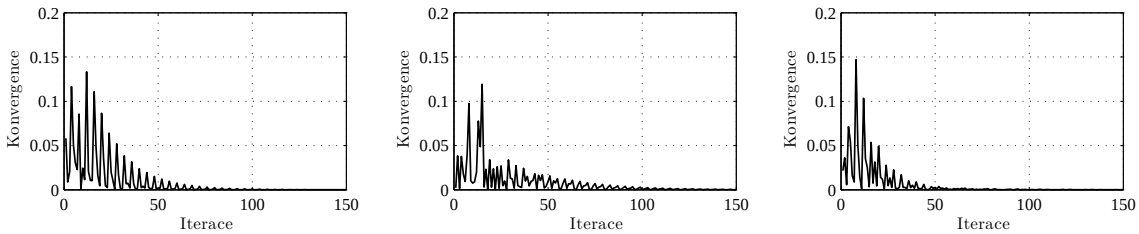

**Graf 9.16:** Hodnoty fitness funkce pro všechny typy regulace a procentuální zlepšení oproti základnímu nastavení regulátoru [logistická mapa].

| ⊞ | HC12 E\|TDAS | | | GE - Struk. | | | HC12 Tuning | | |
|---|---|---|---|---|---|---|---|---|---|
| | Nej. | MED. | STD | Nej. | MED. | STD | Nej. | MED. | STD |
| UPO1 | 16.989 | 17.940 | 3.702 | 6.064 | 6.716 | 13.909 | 4.487 | 4.866 | 12.896 |
| UPO2 | 30.378 | 32.371 | 5.776 | 26.192 | 36.423 | 7.204 | 12.043 | 31.421 | 13.461 |
| UPO4 | 37.955 | 40.766 | 3.185 | 31.364 | 38.834 | 6.139 | 21.663 | 35.398 | 9.720 |

**Tabulka 9.10:** Hodnoty výsledných fitness hodnot ze všech typů regulace [logistická mapa].

Kvalita stabilizace chaosového systému lze posoudit i schopností výsledného řešení konvergovat k žádané hodnotě. Grafy (graf: 9.17a) ETDAS, (graf: 9.17b) GE-Struk a (graf: 9.17c) GE-Tuning zobrazují průběh ustálení pro nejlepší nastavení metody pro 4 fixní body ze všech sekcí tohoto příkladu a v tabulce (tab: 9.11) jsou jednotlivé hodnoty konvergence. Hodnota konvergence je procentuální hodnota, kdy funkce konverguje v určitém požadovaném rozsahu.

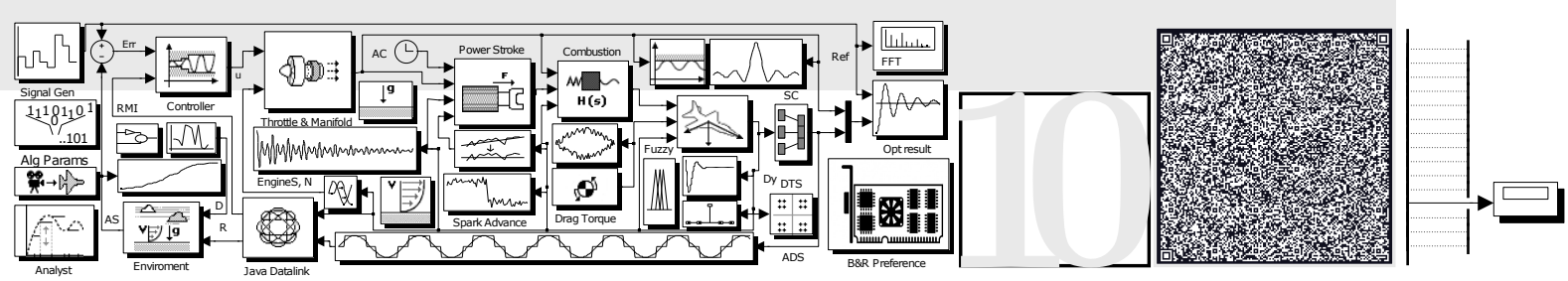





**Graf 9.17:** Grafy konvergence k žádané hodnotě pro UPO4 stabilizaci chaosu [logistická mapa]. Pořadí grafů: (a: ETDAS), (b: GE-Struk) a (c: GE-Tuning).

| ETDAS kongt | GE-Struk kongt | GE-Tuning kongt |
|---|---|---|
| 65.847 | 68.458 | 73.514 |

**Tabulka 9.11:** Hodnota konvergence k žádané hodnotě pro UPO4 stabilizaci chaosu [logistická mapa].




# KAPITOLA 10

# PŘÍKLAD STABILIZACE DUFFINGOVY ROVNICE

Stabilizace duffingovy rovnice je jako další příklad stabilizace tzv. deterministického chaosu, kdy systém je upravená duffingova mapa (vz: 4.25) a celkový popis systému je v sekci (sekce: 4.2.6 - viz str. 58). Systém je stabilizovaný na dva fixní body (UPO) v hodnotách 0.85725557 a 1.40285623, které byly dosaženy na základe analytického výpočtu rovnice ze stabilizační sekvence (vz: 10.1), při počátečních podmínkách $x_0 = 0.1$ a $y_0 = 0.1$. Celková stabilizace systému je provedena na rozsah 600 iterací, kdy graf nestabilizované funkce je (graf: 4.19b), průběh hodnoty $x$ (graf: 4.20b) a histogram konečné hodnoty $x$ (graf: 4.20a).



K nalezení optimálního stabilizátoru v tomto případě je rozděleno na dvě části, první část používá Pyragasovu metodu (sekce: 9.1 - viz str. 117) pro více fixních bodů (ETDAS) a druhá část je zaměřena na doladění hodnoty výsledku pomocí druhého chodu metody na základě speciálně upravených pravidel výběru kontrolované veličiny. Kdy doladění původní metody může přinést zkvalitnění výsledku už optimální stabilizační metody.



Fitness funkce, k ohodnocení parametru $x$ na základe iterace, je použitá stejná metoda jako v případě stabilizace logistické mapy, teda metoda (ITAE) s atraktorem hodnoty (alg: 6).

Optimalizace parametrů obou metod hledání stabilizační metody je pomocí algoritmu (HC12), jako u předešlého přikladu, který představil vhodnost použití algoritmu na hledaní optimálního nastavení metody u deterministického chaosu.

$$\begin{aligned} F_1 &= 0.718039 - 0.451373x_{n-1} \\ F_2 &= 0.376078 - 0.329025x_{n-1} \end{aligned} \tag{10.1}$$

## 10.1 Optimalizace Pyragasovy metody dle HC12

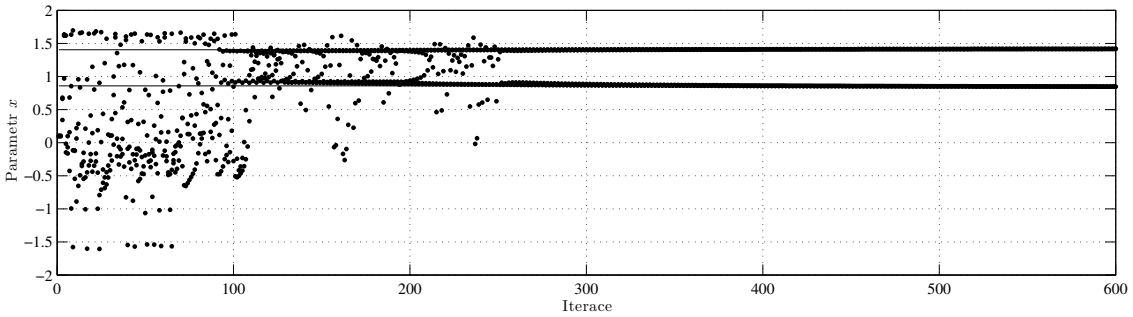
První sekce přikladu stabilizace je na základě Pyragasovy metody, použité u logistické mapy. Metoda je tomto případě rozšířena o atraktor řízené veličiny (alg: 2), kdy výsledný algoritmu je zapsaný v následujícím algoritmu (alg: 8) a dále se funkce v příkladu nazývá (ETDASC). Rozšířená metoda (ETDASC) je schopna lépe zpracovat přechody mezi jednotlivými fixními body a lepe stabilizovat výsledný systém, použití metody na duffingovu mapu je znázorněno dle rovnice (vz: 10.2). Parametry optimalizace jsou v tomto případě podobné jako v předchozím přikladu, tedy $K$, $F_{max}$, $R$ a rozšířeny o parametr $F_{tuning}$, který udává hodnotu zpracování $F_{max}$.

$$x_{n+1} = y_n, y_{n+1} = -bx_n + ay_n - y^3 - ETDASC \tag{10.2}$$

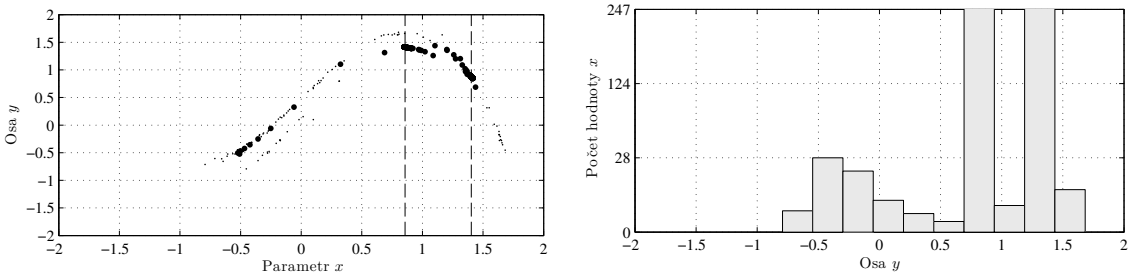
Výsledné parametry stabilizace jsou v tabulce (tab: 10.1) spolu s fitness hodnotou výsledku a nastavení algoritmu (HC12). Grafická prezentace výsledků jsou v grafech hodnoty parametru $x$ na iteraci (graf: 10.1), histogram rozložení hodnoty $x$ v grafu (graf: 10.2b) a výsledná duffingova mapa při použití metody (graf: 10.2a).

| ⊞ | K | $F_{max}$ | R | $F_{tuning}$ | Fit. |
|---|---|---|---|---|---|
| Výs. [Version-A] | -0.538 | 0.447 | 0.997 | 0.292 | 6.064 |
| Výs. [Version-B] | -0.614 | 0.386 | | 4.381 | 6.424 |
| Výs. [Version-C] | -0.563 | 0.879 | 0.996 | 0.592 | 6.742 |
| P:(tab: 4.1) | B:<-2.0,2.0,10> | | | B:<0.0,10.0,10> | ⊡ |
| AI:(tab: 4.2) | M12 [Default], RUNS [100] | | | | |

**Tabulka 10.1:** Tabulka nastavení regulátoru [Regulátor Stabilizace-chaosu Alg HC12-ETDAS] pro Duffingova-rovnice.




**Graf 10.1:** Řízená Duffingova mapa pro 2UPO dle metody ETDASC.




**Graf 10.2:** Pořadí grafů: (a: Průběh duffingovy mapy.) a (b: Rozložení hodnoty $x$ dle osy $y$.).

## 10.2 Doladění Pyragasovy metody dle HC12

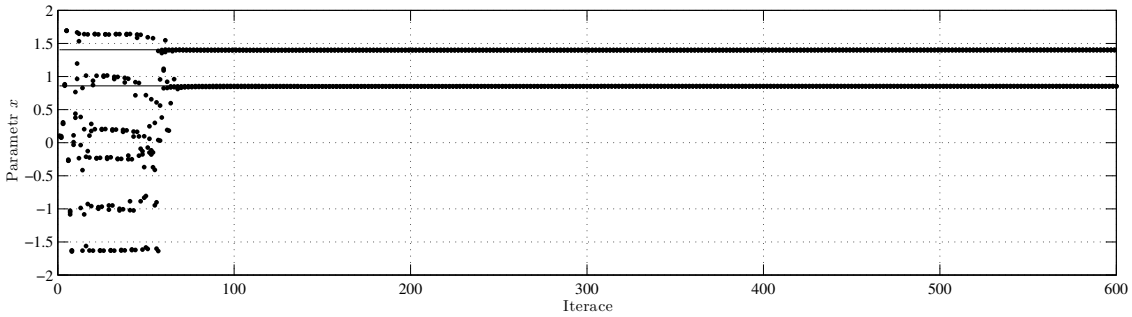
Druhá metoda v rámci stabilizace duffingové metody je sekundární průchod metody ETDASC. Metoda spočívá použitím už jednou nastavené stabilizace, v našem případe verze $A$ z předešlého příkladu a aplikování metody ETDASC na duffingovu mapu pomocí vzorce (vz: 10.3), kde $ETDASC_{prev}$ je už nastavená stabilizace typu $A$.

$$x_{n+1} = y_n - ETDASC, y_{n+1} = -bx_n + ay_n - y^3 - ETDASCprev \tag{10.3}$$

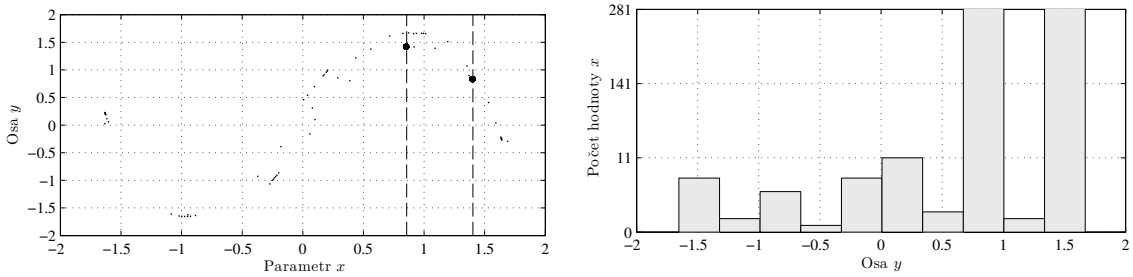
Použitím sekundárního průchodu metody ETDASC a aplikováním stabilizace na základě vzorce (vz: 10.3) můžeme dosáhnout optimálních výsledků, jak dokládají grafy samotného výsledku na duffingově mapě (graf: 10.4a), hodnoty $x$ na iteraci (graf: 10.3) a histogram rozložení hodnoty $x$ (graf: 10.4b). Předně (graf: 10.3) duffingovy mapy znázorňuje daleko přesnější a rychlejší nalezení výsledku než nastavení v předešlé sekci a histogram hodnoty $x$ zase poměrně více bodů $x$ v požadovaném rozsahu fixních bodů (UPO). Výsledné parametry optimalizace, které jsou shodné s předešlou sekcí příkladu, a parametry algoritmu jsou v tabulce (tab: 10.2).

| ⊞ | K | $F_{max}$ | R | $F_{tuning}$ | Fit. |
|---|---|---|---|---|---|
| Výs. [Version-A] | 0.544 | 0.208 | 1.660 | 9.982 | 4.937 |
| Výs. [Version-B] | 0.035 | 0.064 | 1.717 | 2.942 | 5.151 |
| Výs. [Version-C] | 1.778 | 0.165 | 1.745 | 7.088 | 5.189 |
| P:(tab: 4.1) | B:<-2.0,2.0,10> | | | B:<0.0,10.0,10> | ⊡ |
| AI:(tab: 4.2) | M12 [Default], RUNS [100] | | | | |

**Tabulka 10.2:** Tabulka nastavení regulátoru [Regulátor Stabilizace-chaosu Alg HC12-TUNING] pro Duffingova-rovnice.



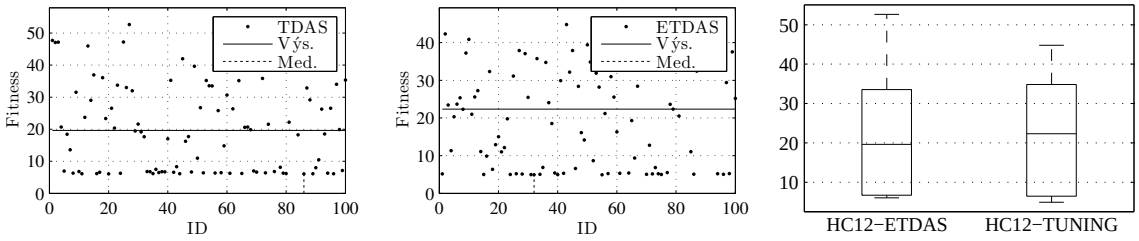

**Graf 10.3:** Řízená Duffingova mapa pro 2UPO dle sekundárním chodu metody ETDASC




**Graf 10.4:** Pořadí grafů: (a: Průběh duffingovy mapy.) a (b: Rozložení hodnoty $x$ dle osy $y$.).

## 10.3 Zhodnocení výsledků

Zhodnocení výsledků pro předchozí dva příklady stabilizace duffingovy mapy. Statistika se provedla na algoritmus (HC12) při 100 opakování a pro každý příklad stabilizace. Jednotlivé výsledky jsou uvedeny v tabulce (tab: 10.3) s grafickým znázorněním nejlepší fitness hodnoty pro stabilizaci pomocí Pyragasovy metody (graf: 10.5a) a pro doladění metody pomocí sekundárního průběhu algoritmu (graf: 10.5b). Dle obou prezentovaných příkladů stabilizace duffingovy mapy lze najít optimální řešení problému a celková úspěšnost algoritmu je v box grafu (graf: 10.5c). Ke stabilizaci chaotického systému lze použít i (GE) algoritmus, kterému se podrobněji věnuje kapitola příklad stabilizace logistické mapy, (sekce: 9.3 - viz str. 119).

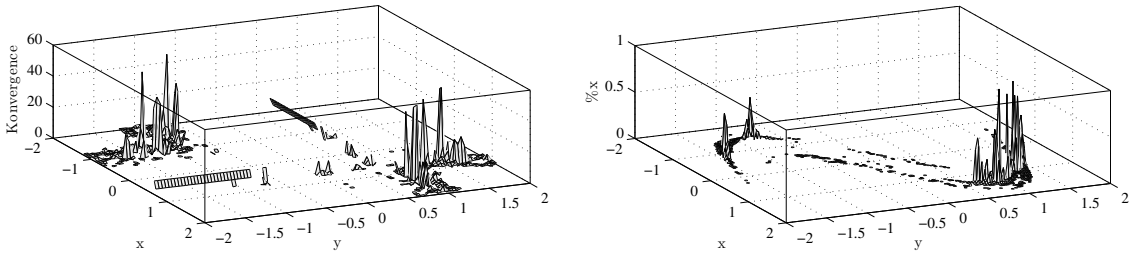



**Graf 10.5:** Hodnoty všech fitness hodnocení. Pořadí grafů: (a: HC12-ETDAS), (b: HC12-TUNING) a (c: Box graf hodnot fitness funkce pro všechny typy stabilizace [duffingova rovnice].).

| ⊞ | HC12-ETDAS | | | HC12-TUNING | | |
|---|---|---|---|---|---|---|
| | Nej. | MED. | STD | Nej. | MED. | STD |
| FIT | 6.064 | 19.648 | 14.119 | 4.937 | 22.333 | 13.325 |

**Tabulka 10.3:** Hodnoty výsledných fitness hodnot ze všech typů stabilizace [duffingova rovnice].

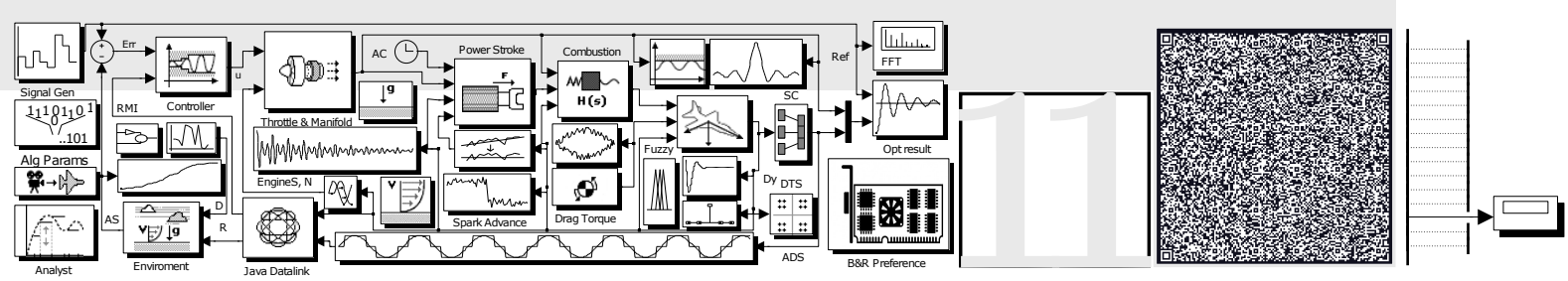
Kvalita hodnoty konvergence pro různé pozice fixních bodů se provedla na nejlepší výsledek stabilizace ze (sekce: 10.2 - viz str. 129), (graf: 10.6a) prezentuje konvergence na celém rozsahu mapy $x, y < -2, 2 >$, rozdělené dle $Nx, Ny = 300$. Jednotlivé hodnoty jsou v tabulce (tab: 10.4), pro hodnoty původních hodnot (UPO), nejlepší nalezené hodnoty (UPO) a konvergenci celé mapy. Všechny hodnoty $x$ ze všech konvergujících možností nastavení (UPO) jsou v grafu (graf: 10.6b).




**Graf 10.6:** Grafy rozložení konvergence [duffingova rovnice]. Pořadí grafů: (a: Hodnota konvergence v závislosti na změně polohy UPO.) a (b: Hodnoty $x$ pro všechny konvergující polohy UPO.).

| Kongt | Alt. UPO1 | Alt. UPO2 | Alt kongt | Mapa kongt |
|---|---|---|---|---|
| 50.167 | -1.434 | -0.869 | 54.500 | 24.570 |

**Tabulka 10.4:** Hodnota rozložení konvergence při změně polohy UPO [duffingova rovnice].




# KAPITOLA 11

# PŘÍKLAD NÁVRHU ANTÉNY

Příklad návrhu Yagi-Uda antény pro co nejlepší parametry ve stanoveném frekvenčním pásmu 290 až 310 MHz. Popis Yagi-Uda antény je v sekci (sekce: 4.2.7 - viz str. 60). Toto pásmo bylo vybráno z toho důvodu, jelikož veškeré geometrické délky prvků antény jsou vztahovány k délce vlny $\lambda$. V případě frekvenčního pásma okolo 300 MHz je délka vlny zhruba rovna 1.

To nám dává lepší představu o celkových rozměrech navržené antény. Kvalita konstrukce je vyjádřena matematicky objektivní fitness funkcí. Tato funkce byla navržena experimentálně a zahrnuje tři vybrané parametry antény, které definují její celkovou kvalitu. Jelikož mezi jednotlivými parametry antény existuje jistá souvislost, takže výsledná anténa je vždy jakýmsi kompromisem mezi těmito parametry, kterými jsou zisk antény $G$, vstupní impedance antény vyjádřená pomocí poměru stojatých vln VSWR (zkratka vychází z anglického slovního spojení voltage standing wave ratio) a předozadní poměr $F/B$. Výsledná fitness funkce $F$ má tvar.



$$F = -G_{min} + (a * vswr_{max}) + \left(b * \frac{F}{B_{min}}\right) \tag{11.1}$$

Pří návrhu antén je zvykem navrhovat anténu pouze pro střed uvažovaného pásma, což je v našem případě 300 MHz. Jelikož využíváme od samého počátku automatizovaného návrhu pomocí evolučních algoritmů, tak si můžeme dovolit strategii, kdy uvažujeme parametry v celé šířce pásma a z tohoto pásma jsme vybrali nejhorší variantu parametrů. Tento přístup omezil počet vyhovujících řešení, ale na druhou stranu se vyhneme situacím, kdy by výsledná anténa měla výborné výsledky pro střed pásma, ale pro okolní frekvence budou tyto parametry nevyhovující. V případě zisku a předozadního poměru uvažujeme jejich nejnižší hodnotu $G\ L$ a $F/B$ min. U poměru stojatých vln uvažujeme jeho nejvyšší hodnotu $vswr$ max. Pomocí konstant $a$ a $b$ můžeme následně upravit váhu, jakou budou mít parametry $vswr$ a $F/B$ vůči zisku $G\ L$. Konstanta $a$ určuje váhu poměru stojatých vln a nabývá těchto hodnot. V případě konstanty $b$ odpovídající váze parametru předozadního poměru

je hodnota následující.

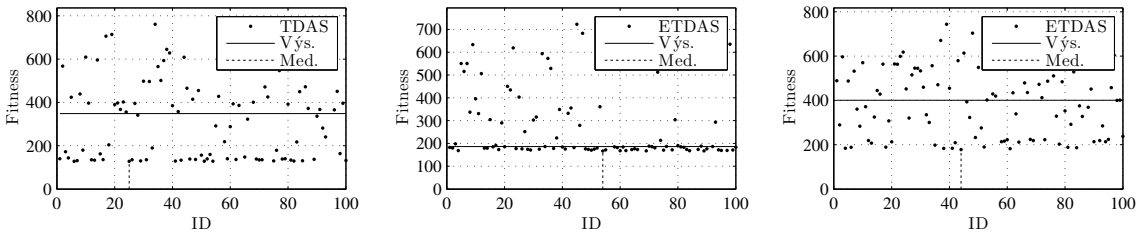
$$a = \begin{cases} 0.1 & if\ vswr \leq 1.8 \\ 30 & if\ vswr > 1.8 \end{cases}, b = 30 \tag{11.2}$$

## 11.1 Zhodnocení výsledků

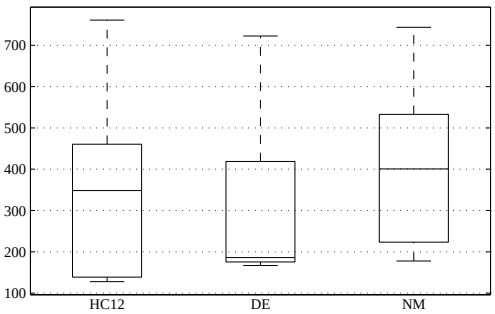
Na základě stanovené fitness funkce (vz: 11.1) byla provedena série simulací pro každý vybraný algoritmus. Nejlepší dosažené výsledky pro jednotlivé algoritmy jsou uvedeny v následující tabulce (tab: 11.2). Tabulka obsahuje i rozměry jednotlivých elementů antény a rozteče mezi jednotlivými elementy. Na konci tabulky jsou uvedeny dosažené hodnoty tří vybraných parametrů, které byly zahrnuty do fitness funkce. Průměrné hodnoty společně s extrémy od zisku po délku antény jsou uvedeny v tabulce (tab: 11.3) a grafech (graf: 11.3a), (graf: 11.3b) a (graf: 11.3c). Vyzařovací diagramy pro nejlepší nastavení antén dle jednotlivých algoritmů znázorňují grafy (graf: 11.2a), (graf: 11.2b) a (graf: 11.2c). Grafické znázornění jednotlivých antén je v grafech (graf: 11.4a), (graf: 11.4b) a (graf: 11.4c). Poslední porovnaní udává kvalitu jednotlivých algoritmů pro 100 opakovaní na jednom zadání dle grafu (graf: 11.1b), (graf: 11.1a) a (graf: 11.1c) společně s tabulkou statistik fitness hodnot (tab: 11.1).

| ⊞ | HC12 | | | DE | | | NM | | |
|---|---|---|---|---|---|---|---|---|---|
| | Nej. | MED. | STD | Nej. | MED. | STD | Nej. | MED. | STD |
| FIT | 128 | 348.319 | 191.589 | 167 | 186.262 | 175.026 | 178 | 400.629 | 157.192 |

**Tabulka 11.1:** Hodnoty výsledných fitness hodnot ze všech typů regulace [Yagi-Uda anténa].

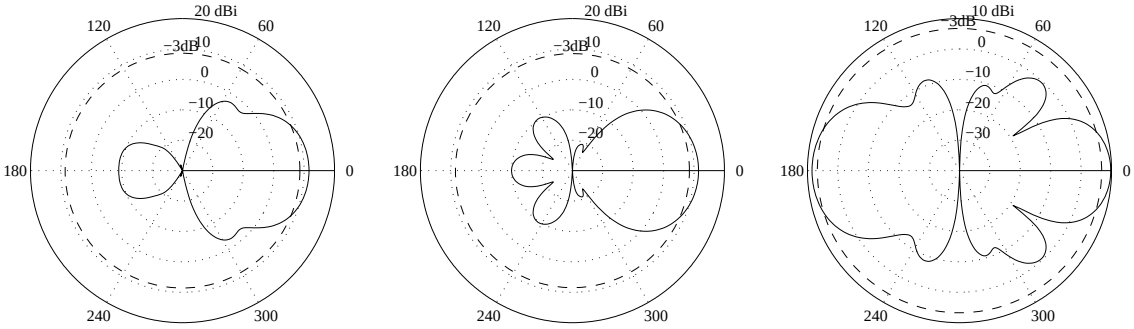



**Graf 11.1:** Hodnoty všech fitness hodnocení. Pořadí grafů: (a: HC12), (b: DE) a (c: NM).

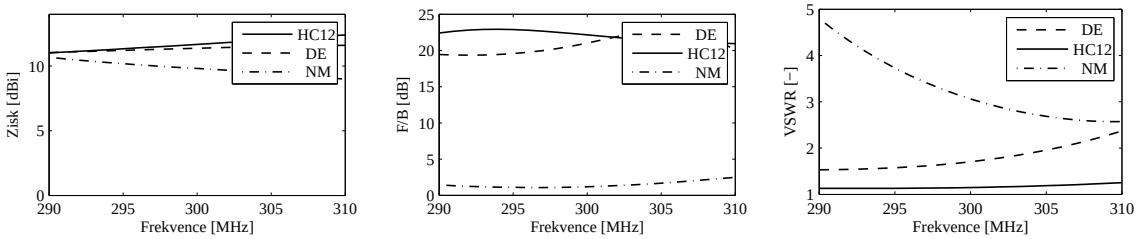



Návrh parametrů antény je komplexní problém skládající se z mnoha faktorů, které více či méně ovlivňují výsledný návrh antény. Mezi tyto faktory patří zejména směrová charakteristika antény, předozadní poměr, impedance a hlavně zisk. Všechny tyto funkce jsou vysoce nelineární. V tomto příkladu byl představen návrh Yagi-Uda antény. Na tomto konkrétním typu antény bylo demonstrováno použití algoritmů umělé inteligence. Byly využity algoritmy (NM), (DE) a (HC12). Fitness funkce (vz: 11.1) zahrnuje parametry zisku, předozadního poměru a poměru stojatých vln *vswr*. Veškeré simulace

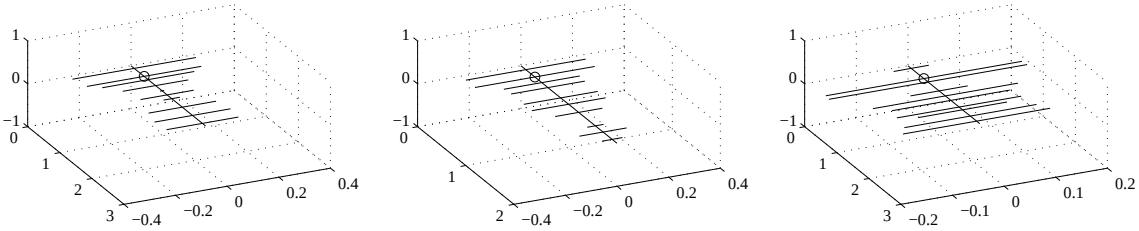
probíhaly v programu superNEC. Výsledky těchto simulací jsou uvedeny v tabulce (tab: 11.2). Použití genetických algoritmů pro návrh parametrů antén je vhodné zejména tam, kde je potřeba návrh časově urychlit.




**Graf 11.2:** Vyzařovací diagramy. Pořadí grafů: (a: Návrh dle DE.), (b: Návrh dle HC12.) a (c: Návrh dle NM.).




**Graf 11.3:** Grafy průběhů. Pořadí grafů: (a: Průběh zisku.), (b: Průběh předozadního poměru.) a (c: Průběh poměru stojatých vln.).



**Graf 11.4:** Návrh konečných antén. Pořadí grafů: (a: Návrh dle DE.), (b: Návrh dle HC12.) a (c: Návrh dle NM.).

| Algoritmus | NM | | DE | | HC12 | |
|---|---|---|---|---|---|---|
| Element | Délka [m] $\lambda$ | Mezera [m]$\lambda$ | Délka | Mezera | Délka | Mezera |
| 1 | 0.068 | 0.395 | 0.479 | 0.310 | 0.464 | 0.190 |
| 2 | 0.380 | 0.130 | 0.443 | 0.126 | | 0.145 |
| 3 | 0.384 | 0.344 | 0.357 | 0.230 | 0.294 | 0.127 |
| 4 | 0.111 | 0.196 | 0.254 | 0.202 | 0.286 | 0.266 |
| 5 | 0.281 | 0.171 | 0.016 | 0.158 | | 0.164 |
| 6 | 0.228 | 0.258 | 0.206 | 0.145 | | 0.209 |
| 7 | 0.241 | 0.108 | 0.063 | 0.329 | 0.183 | 0.347 |
| 8 | 0.172 | 0.243 | 0.262 | 0.317 | 0.063 | 0.158 |
| 9 | 0.256 | 0.194 | 0.294 | 0.310 | 0.183 | 0.190 |
| 10 | 0.285 | - | 0.278 | - | 0.079 | - |

| | | | |
|---|---|---|---|
| Zisk [dBi] | 9.0 - 10.5 | 11.0 - 12.5 | 11.0 - 11.6 |
| VSWR [-] | 2.55 - 4.95 | 1.55 - 2.25 | 1.13 - 1.25 |
| F/B [dB] | 1.1 - 2.5 | 19 - 23 | 21 - 23 |

**Tabulka 11.2:** Souhrn parametrů navržených antén.

| ⊞ | NM | | | DE | | | HC12 | | |
|---|---|---|---|---|---|---|---|---|---|
| | Min | Prů | Max | Min | Prů | Max | Min | Prů | Max |
| Zisk [dBi] | -1.26 | 2.69 | 6.31 | 7.24 | 8.93 | 13.2 | 10.1 | 10.3 | 12.7 |
| PSV [-] | 1.6 | 2.13 | 3.45 | 1.22 | 1.70 | 3.77 | 1.20 | 1.74 | 3.30 |
| F/B [dB] | -3.18 | 1.60 | 7.57 | 4.88 | 20.6 | 30.9 | 11.5 | 19.6 | 35.1 |
| Delka [m] | 1.57 | 2.08 | 2.38 | 1.69 | 1.92 | 2.50 | 1.40 | 1.97 | 2.70 |

**Tabulka 11.3:** Porovnání statistik a extrémů vlastností antén.

SuperNEC je výpočetní program vyvinutý za účelem simulačního výpočtu antén a rozšiřuje dřívější verzi (NEC2). Program je nástavba simulačního prostředí Matlab. Program NEC2 využívá k simulaci momentovou metodu (MOM), tato metoda vyžaduje, aby problém byl diskreditovaný ve formě tzv. "wire segments". Každý segment a mřížka diskreditovaného problému je ve velikosti přibližně 1/10 požadované vlnové délky a využívá (UTD) metodu. SuperNEC program používá této metody pro výpočty, které jsou náročné na základě vlnové délky, a pro definici problému využívá rovinných mřížek a cylindrů. Metoda (UTD) doplňuje metodu (MOM), pro výpočetně náročné příklady.

Program implementuje i paralelní výpočty na základě (PVM) nástrojů a umožňuje tak spuštění programu na více počítačích (operačních systémů), které jsou propojeny na základě standardního síťového protokolu TCP/IP.

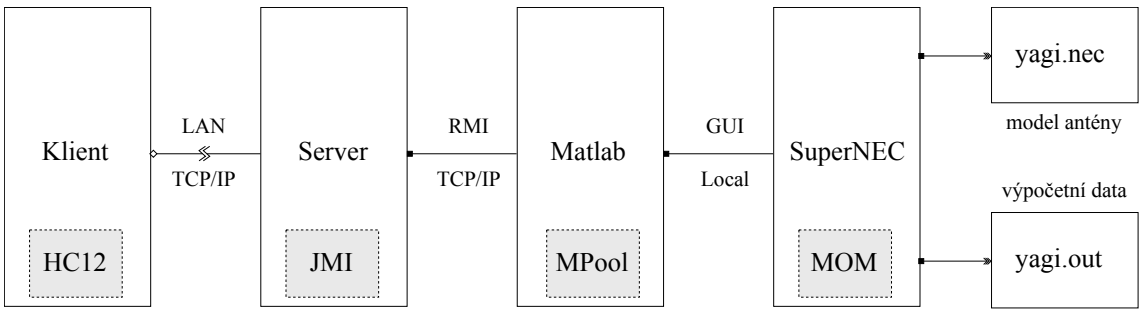
$$L(J) = E \tag{11.3}$$

Technika výpočtu zahrnuta v programu definuje hybridní metodu (MOM) a její velice základní popis je znázorněn na příkladu použití momentové metody (vz: 11.3), kde $L$ znamená lineární operátor.

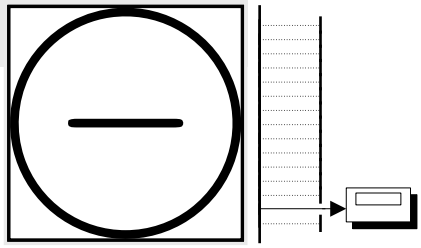
$$J = \sum_{n=1}^{N} \alpha_n J_n \tag{11.4}$$

Lineární operátor v programu SuperNEC je Pocklingtonova rovnice, která operuje na základě elektrického pole a proudů. V momentové metodě je proud $J$ rozšířen na sérii jednoduchých funkcí $J_1$, $J_2$, $J_3$ ... (vz: 11.4), kde $N$ je počet jednoduchých funkcí pro popis proudu a $\alpha_n$ je neznámý komplexní koeficient.

Schéma zapojení simulačního programu na výpočet Yagi-Uda antén je znázorněný na obrázku (obr: 11.1) a zahrnuje jednu z možností komunikace se SuperNEC, Matlab - server, klient.




**Obrázek 11.1:** Schéma simulačního programu na výpočet Yagi-Uda antén.



# ZÁVĚR

Problematika nelineárního řízení komplexních soustav a analýza výsledků je dosti složité téma, protože zanedbaní některých částí muže výrazně ovlivnit samotný výsledek. Proto je nutné ke každému problém přistupovat individuálně a pečlivě připravit a nastavit kvalifikační požadavky na požadovaný výsledek. V rámci této disertační práce byly prezentovány v problematice řízení komplexních soustav variance druhů regulace systémů od klasického matematického modelu až po reálný vysoce nelineární systém a reálný hardware. Dále byla práce rozšířena o nové postupy při stabilizaci nedeterministického chaosu a přikladu návrhu struktury antén.

V oboru regulace systémů je nutno postupovat dle platných předpisů právě díky závažným důsledkům, které by mohly pramenit ze špatné interpretace výsledku měření či dokonce nevhodně zvoleném postupu regulace.

Všechny zde uvedené příklady jsou porovnány se standartním způsobem řešení používaným v praxi a doplněny o nové poznatky s využitím evolučních algoritmů. Hodnocení výsledků je prezentováno na statistických grafech a ukazuje mnohé výhody využití tohoto postupu pro daný typ přikladu. Každá z kapitol v práci představuje jeden způsob řešení problému a představuje dále podpis daného postupu.

K dosažení všech výsledků bylo nutno vytvořit aplikační prostředky na kterých byly demonstrovány různé způsoby přístupů k dané problematice. Vytvořená aplikace splňuje mnohé výrazné vlastnosti a moderní přístupy k vývoji aplikací. Hlavní předností aplikace jsou způsoby zapojení dle server / klient konfigurace, multi plaformový design, využívání paralelních výpočtů a mnohé další vlastnosti prezentované ve vlastní kapitole.

Rozdělení práce je na teoretickou část, která se zabývá popisem problému v oblasti řízení systémů a optimalizace spolu s evolučními algoritmy, v kapitole 1 až 2, a praktickou část. Praktická část zahrnuje popis samostatné aplikace, kapitola 3, stručný popis všech řešených systémů spolu s návrhem hodnoticí funkce, kapitola 4. Kapitola 5 až 7 prezentuje postupy při regulaci komplexních soustav a kapitola 8 obsahuje zhodnoceni výsledků napříč všemi řešeními. Ke stabilizaci deterministického chaosu je věnována kapitola 9 až 10. Poslední kapitola představuje řešení návrhu obecných antén.

Práce nelineární řízení komplexních soustav s využitím evolučních algoritmů prezentuje mnohé nové přístupy a poznatky, které se dají dále rozvíjet a využít na reálných problémech v technické praxi. Celkové výsledky této práce se dají rozdělit na tyto dílčí výsledky:

- Obecně popsaná problematika regulace teorie řízení a optimalizace s popisem všech využívaných evolučních algoritmů v této práci.
- Systematický popis možného aplikačního přístupu.
- Aplikovaní problematiky do vlastního softwarového řešení.
- Využití znalostí multikriteriální optimalizace k návrhu vlastní hodnoticí funkce, zohledňující variabilitu vlastností při regulaci systémů.
- Provedení série testů na nové způsoby regulace systémů, a to konkrétně v rámci návrhu PID regulátoru.
- Využití Fuzzy logiky pro lepší návrh komplexního regulátoru.
- Postup vlastního návrhu struktury regulátoru a komplexní doladění s využitím evolučních algoritmů.
- Nové přístupy pro stabilizaci deterministického chaosu.
- Nové postupy a fitness funkce při navrhovaní struktury Yagi-Uda antény.
- Zpracovaní a analýza výsledků v grafech a statistických metodách.
- Porovnaní výsledků pro různé přístupy k řešení daného problému.
- Vytvoření samostatné aplikace umožňující další vývoj a použití v technické praxi.

# SEZNAM VLASTNÍCH PUBLIKACÍ

2017

*Zuth Daniel*, *Minář Petr*: *Aplikace fuzzy množin pro určení technického stavu stroje.* Diago 2017. Technická diagnostika, ročník. 26. z1 s. 87-92. ISSN: 1210- 311X.

2015

*Matyáš Pavel*, *Minář Petr*, *Ošmera Pavel*: *Design of Yagi- Uda antenna using HC12 algorithm.* In Mendel 2015. Mendel Journal series. 21th International Conference on Soft Computing. Mendel Journal series. 2. 2015. s. 143-149. ISBN: 978-3-319-19823-1. ISSN: 1803- 3814.

2014

*Matoušek Radomil*, *Dobrovský Ladislav*, *Minář Petr*, *Mouralova Kateřina*: *A Note about Robust Stabilization of Chaotic Hénon System Using Grammatical Evolution.* A Note about Robust Stabilization of Chaotic Hénon System Using Grammatical Evolution. Advances in Intelligent Systems and Computing, 2014, roč. 2014, č. 289, s. 219-228. ISSN: 2194- 5357.

*Matyáš Pavel*, *Minář Petr*: *Návrh Yagi- Uda antény pomocí evolučních algoritmů.* In Technical Computing Bratislava 2014. 2014. Slovakia, Bratislava: HUMUSOFT, 2014. s. 1-4. ISBN: 978-80-7080-898- 6.

2013

*Matoušek Radomil*, *Minář Petr*: *Stabilization of Chaotic Logistic Equation Using HC12 and Grammatical Evolution.* Advances in Intelligent Systems and Computing, 2013, roč. 2013, č. 210, s. 137-146. ISSN: 2194- 5357.

2012

*Matoušek Radomil*, *Minář Petr*, *Lang Stanislav*, *Pivoňka Petr*: *HC12: Efficient Method in Optimal PID Tuning.* Engineering Letters, 2012, roč. 20, č. 1, s. 42-48. ISSN: 1816- 093X.

2011

*Matoušek Radomil*, *Minář Petr*, *Lang Stanislav*, *Šeda Miloš*: *HC12: Efficient PID Controller Design*. Computational Engineering in Systems, IAASAT, 2011. s. 194-199. ISBN: 978-1-61804-026- 8.

*Matoušek Radomil*, *Minář Petr*, *Lang Stanislav*, *Pivoňka Petr*: *HC12: Efficient Method in Optimal PID Tuning*. Lecture Notes in Engineering and Computer Science, 2011, roč. 2011, č. 1, s. 463-468. ISSN: 2078- 0958.

*Matoušek Radomil*, *Minář Petr*, *Lang Stanislav*, *Pivoňka Petr*: *Evolutionary Design of Polynomial Controller*. An international Journal of Science, Engineering and Technology, World Academy of Science Engineering and Technology, 2011, roč. 2011, č. 59, s. 639-644. ISSN: 2010- 376X.

*Minář Petr, Matoušek Radomil CDF v1; Control Design Framework (PID, Fuzzy)*. VUT Brno. URL: http://www.uai.fme.vutbr. cz. (software).

2010

*Matoušek Radomil*, *Minář Petr*: *Optimalizace parametrů PSD regulátoru pomocí algoritmu HC12*. In Technical Computing Bratislava: RT Systems, 2010. s. 68-70. ISBN: 978-80-970519-0- 7.

2009

*Minář Petr*: *Distribuované výpočty s využitím technologie ActionScript*. Brno, 2009. Diplomová práce. Vyské učení technické v Brně.

*Minář Petr*: *Interaktivní webové aplikace vytvořené pomocí technologie Adobe Flash - Teorie Kódování*. Brno, 2009. VUT Brno. URL: http://www.uai.fme.vutbr. cz. (software).

# LITERATURA


[1] *Hamming Richard*: *Numerical methods for scientists and engineers*. New York: Dover, 1986. ISBN 0486652416.

[2] *Haupt L. Randy*, *Haupt Sue Ellen*: *Practical Genetic Algorithms*. Wiley-Interscience, 2004. ISBN 0471455652.

[3] *Zelinka Ivan*, *Oplatková Zuzana*, *Šeda Miloš*, *Ošmera Pavel*, *Včelař František*: *Evoluční výpočetní techniky: Principy a aplikace*. Brno: Ben, 2008. ISBN 9788073002183.

[4] *Marler R. T.*, *Arora J. S.*: *Survey of multi-objective optimization methods for engineering*. Structural and Multidisciplinary Optimization, Vol. 26, No. 6. (1Duben 2004), pp. 369-395, doi:10.1007/s00158-003-0368-6.

[5] *Langdon B. William*, *Poli Riccardo*: *Foundations of Genetic Programming*. London: Springer, 2010. ISBN 3642076327.

[6] *Mitchell*, *Melanie*: *An introduction to genetic algorithms*. Cambridge: MIT Press, 1996. ISBN 0262631857.

[7] *Holland*, *H. John*: *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. (1975) University of Michigan Press, Ann Arbor

[8] *Mathews H. John*, *Kurtis K. Fink*: *Numerical Methods Using Matlab*. Upper Saddle River, New Jersey, USA: Prentice-Hall Inc., 2004, s. 8. ISBN 0130652482.

[9] *Nelder John*, *Mead Roger*: *A simplex method for function minimization*. (1965) Computer Journal 7: 308–313.

[10] *Minář Petr*: *Distribuované výpočty s využitím technologie ActionScript*. Brno, 2009. Diplomová práce. Vyské učení technické v Brně.

[11] *Jin Y.*: *A Definition of Soft Computing - adapted from L.A. Zadeh* [online]. Computational intelligence, natural computing, and organic computing [online]. 2000 [cit. 2012-07-13] Dostupné z URL: <`www.soft-computing.de/def.html`>.

[12] *Tvrdík Josef*: *Evoluční Algoritmy*. Ostrava, 2004. Učební texty. Ostravská univerzita.

[13] *Lampinen Jouni*: *Multi-Constrained Nonlinear Optimization by the Differential Evolution Algorithm* [online]. 2000 [cit. 2012-07-14]. Dostupné z URL: <`www2.it.lut.fi/kurssit/05-06/Ti5216300/DECONSTR.PDF`>.

[14] *Kukal Jaromír*, *Jakeš Bohumil*, *Bártová Darina*: *Využití diferenciální evoluce k robustní identifikaci parametrů systémů*. Automatizace. 2009, roč. 52, č. 3.

[15] *Feoktistov Vitaliy*: *Differential Evolution: In Search of Solutions*. USA: Springer, 2006. ISBN 0387368957.

[16] *Storn Rainer*, *Price Kenneth*: *Differential Evolution – a Simple and Efficient Heuristic for Global Optimization*. (1997) J. Global Optimization, 11, 341–359.

[17] *Tomek Eduard*: *Škálování implementace optimalizačního algoritmu na mnohajádrových strojích*. Brno, 2012. Bakalářská práce. Masarykova univerzita.

[18] *Matoušek Radomil*, *Žampachová Eva*: *Promising GAHC and HC12 algorithms in global optimization tasks*. OPTIMIZATION METHODS & SOFTWARE, 2011, roč. 26, č. 3, s. 405-419. ISSN: 1055- 6788.

[19] *Matoušek Radomil*: *HC12: Highly Scalable Optimization Algorithm*. Studies in Computational Intelligence, Springer Berlin / Heidelberg, 2010, roč. 2010, č. 284, s. 177-183. ISSN: 1860- 949x.

[20] *O'Neill Michael*, *Rayn Conor*: *Grammatical evolution : Evolution Automatic Programming in an Arbitrary Language*. Kluwer Academic Publishers 2003. 145 p. ISBN: 1-4020-7444-1.

[21] *Dempsey Ian*, *O'Neill Michael*, *Brabazon Anthony*: *Foundations in Grammatical Evolution for Dynamic Environments*. USA: Springer, 2010. ISBN 3642101402.

[22] *Ryan Conor*, *Collins J.J.*, *O'Neill Michael*: *Grammatical Evolution: Evolving Programs for an Arbitrary Language*. EuroGP 1998, pages 83–96.

[23] *Burbidge Robert*, *Wilson S. Myra*: *Grammatical evolution of a robot controller*. Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on , vol., no., pp.357-362, 10-15 Oct. 2009 doi: 10.1109/IROS.2009.5354411.

[24] *Modrlák Osvald*: *Fuzzy řízení a regulace: Studujní materiály*. Liberec: Technická univerzita v Liberci, 2002.

[25] *Passino M. Kevin*: *Fuzzy Control*. USA: Addison Wesley, 1997. ISBN 020118074X.

[26] *Jantzen Jan*: *Foundations of Fuzzy Control*. Chichester, England: Wiley, 2007. ISBN 0470029633.

[27] *Zadeh A. Lotfi*: *Fuzzy sets*. (1965) Information and Control, 8 (3), pp. 338-353.

[28] *Cordón Oscar*: *Genetic Fuzzy Systems: Evolutionary Tuning And Learning Of Fuzzy Knowledge Bases*. USA: WSPC, 2002. ISBN 9810240171.

[29] *Lacevic Bakir*, *Velagic Jasmin*: *Evolutionary Design of Fuzzy Logic*. Based Position Controller for Mobile Robot. Journal of Intelligent and Robotic Systems. 2011, c. 63, s. 595-614.

[30] *Mamdani H. Ebrahim*: *Application of fuzzy algorithms for the control of a simple dynamic plant*. In Proc IEEE (1974), 121-159.

[31] *Sugeno Michio*: *Theory of fuzzy integrals and its applications*. Tokyo, Japan, 1974. Ph.D. thesis. okyo Institute of Technology.

[32] *Ošmera Pavel*: *Evolvable Fuzzy Controllers using Parallel Evolutionary Algorithms*. In EUSFLAT 2003. Zittau: University of Applied Sciences, 2003. s. 349 (s.)ISBN: 3-9808089-4- 7.

[33] *Matoušek Radomil*, *Ošmera Pavel*: *Automatic Optimal Design of Fuzzy Controllers*. GALESIA. 1997.

[34] *Chuen Chien Lee*: *Fuzzy logic in control systems: fuzzy logic controller-parts 1 and 2*. IEEE Transactions on Systems, Man, and Cybernetics, Vol. 20, No. 2, pp 404-435, 1990.

[35] *García-Pérez Lía*, *Cańas M. José*, *García-Alegre C. María*, *Yáñez Pablo*, *Domingo Guinea*: *Fuzzy control of an electropneumatic actuator*. IEEE Trans. System, Man and Cybernetics, Jan.1996.

[36] *King E. Robert*, *Stathaki A.*: *Fuzzy Gain Scheduling Control of Nonlinear Processes*. Department of Electrical and Computer Engineering, University of Patras, Greece.

[37] *Modrlák Osvald*: *Teorie automatického řízení 2: Studijní materiály*. Liberec: Technická univerzita v Liberci, 2004.

[38] *Švarc Ivan*, *Matoušek Radomil*, *Šeda Miloš*, *Vítečková Miluše*: *Automatické řízení*. Brno: Cerm, 2012. ISBN 9788021443983.

[39] *Nise S. Norman*: *Control Systems Engineering*. USA: Wiley, 2010. ISBN 0470547561.

[40] *Dorf C. Richard*, *Bishop H. Robert*: *Modern control systems*. London: Upper Saddle River, N.J, 2010. ISBN 0136024580.

[41] *Nagaraj B.*, *Vijayakumar P.*: *A comparative study of PID controller tuning using GA, EP, PSO and ACO*. Communication Control and Computing Technologies (ICCCCT). 2010, c. 1, 305 - 313.

[42] *Fabijanski P.*, *Lagoda R.*: *On-line PID controller tuning using genetic algorithm and DSP PC board*. Power Electronics and Motion Control Conference, 2008. EPE-PEMC 2008. 13th , vol., no., pp.2087-2090, 1-3 Sept. 2008.

[43] *Gao F.*, *Tong H. Q.*: *Differential Evolution: An Efficient Method in Optimal PID Tuning and on-line Tuning*. Proceedings of the First International Conference on Complex Systems and Applications, 2006.6, Wuxi, China (SCI).

[44] *Muangsong, R., Koolpiruck, D., Khantachawana, A., Niranatlumpong P.: A particle swarm optimization approach for optimal design of PID controller for position control using Shape Memory Alloys*. Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, 2008. ECTI-CON 2008. 5th International Conference on , vol.2, no., pp.677-680, 14-17 May 2008.

[45] *Matoušek Radomil, Minář Petr, Lang Stanislav, Šeda Miloš: HC12: Efficient PID Controller Design*. Engineering Letters, 2012, roč. 20, č. 1, s. 42-48. ISSN: 1816093X.

[46] *Matoušek Radomil, Minář Petr, Lang Stanislav, Pivoňka Petr: Evolutionary Design of Polynomial Controller*. An international Journal of Science, Engineering and Technology, World Academy of Science Engineering and Technology, 2011, roč. 2011, č. 59, s. 639-644. ISSN: 2010- 376X.

[47] *Matoušek Radomil, Minář Petr: Optimalizace parametrů PSD regulátoru pomocí algoritmu HC12*. In Bratislava: RT Systems, 2010. s. 68-70. ISBN: 978-80-970519-0- 7.

[48] *Matoušek Radomil, Minář Petr, Lang Stanislav, Pivoňka Petr: HC12: Efficient Method in Optimal PID Tuning*. Lecture Notes in Engineering and Computer Science, 2011, roc. 2011, c. 1, s. 463-468. ISSN: 2078-0958.

[49] *Sierra Kathy, Bates Bert: Head first Java*. Sebastopol, Calif.: O'Reilly, 2005. ISBN 0596009208.

[50] *Altman M. Yair: Undocumented Secrets of MATLAB-Java Programming*. London: Chapman and Hall/CRC, 2011. ISBN 1439869030.

[51] *Šenkeřík Roman: Optimal Control of Deterministic Chaos*. Zlín, 2008. Disertacní práce. Univerzita Tomáše Bati, Zlín. Vedoucí práce Zelinka, Ivan.

[52] *Zelinka Ivan: Analytic programming by means of new evolutionary algorithms*. (2001) Proceedings of 1st International Conference on New Trends in Physics'01, Brno, Czech Republic, pp. 210–214.

[53] *Ošmera Pavel, Roupec Jan, Matoušek Radomil, Weisser R.: Two-Level Transplant Evolution for Optimization of General Controllers*. New Trends in Technologies: Control, Management, Computational Intelligence and Network Systems, pp.50-68, ISBN 978-953-307-213-5, (2010), Sciyo.

[54] *Griffin Ian: On-line PID Controller Tuning using Genetic Algorithms*. Dublin City University, 2003.

[55] *May M. Robert: Simple mathematical models with very complicated dynamics*. Nature 261(5560):459-467. (1976)

[56] *Ogata Katsuhiko: Modern Control Engineering*. USA: Prentice Hall, 2009. ISBN 978-0136156734.

[57] *Yum Li, Kiam Heong Ang, Chong G.C.Y.: PID control system analysis and design*. Control Systems, IEEE, On page(s): 32 - 41 Volume: 26, Issue: 1, Feb. 2006.

[58] *Sugeno Michio: An introductory survey of fuzzy control*. Information Sciences, Volume 36, Issues 1–2, July–August 1985, Pages 59–83.

[59] *Bandyopadhyay B., Rao A., Singh H.: On pade approximation for multivariable systems*. Circuits and Systems, IEEE Transactions on , vol.36, no.4, pp.638,639, Apr 1989 doi: 10.1109/31.92898.

[60] *Kalyanmoy Deb: Multi-objective optimization using evolutionary algorithms*. Chichester, UK: Wiley; 2001. ISBN 978-0-471-87339-6.

[61] *Matoušek Radomil: Pokročilé metody počítačové intligence*. Brno, 2014. Habilitační práce. Vyské učení technické v Brně. ISBN 978-80.-214-4341-0

[62] *Fletcher R.: Practical methods of optimization*. Wiley-Interscience, 2000. ISBN 978-0471494638.

[63] *Fakhri Karray, De Silva W. Clarence: Soft Computing and intelligent systems design: theory, tools and applications: theory and applications*. Addison Wesley, 2004. ISBN 978-0321116178.

[64] *Russell Stuart, Norvig Peter: Artificial intelligence: A modern approach*. Prentice Hall, 2004. ISBN 978-0136042594.

[65] *Calvert L. Kenneth, Donahoo J. Michael: TCP/IP Sockets in Java: Practical Guide for Programmers*. Morgan Kaufmann, 2008. ISBN 978-0123742551.

[66] *Grosso William: Java RMI*. O'Reilly Media, 2001. ISBN 978-1565924529.

[67] *Kepner Jeremy*: *Parallel MATLAB for Multicore and Multinode Computers*. SIAM-Society for Industrial and Applied Mathematics, 2009. ISBN 978-0898716733.

[68] *Fedorov V. Valerii*, *Leonov L. Sergei*: *Optimal Design for Nonlinear Response Models*. CRC Press, 2013. ISBN 978-1439821510.

[69] *B&R* : *Automation Studio Target for Simulink* [online]. Automation Studio tutorials [online]. Dostupné z URL: <`http://kyb.fei.tuke.sk/laboratoria/prezentacie/br%20automation%20studio%20target%20for%20simulink.pdf`>.

[70] *Ghaffari Azad*: *dSPACE and Real-Time Interface in Simulink* [online]. San Diego State University study materials [online]. Dostupné z URL: <`http://flyingv.ucsd.edu/azad/dSPACE_tutorial.pdf`>.

[71] *Minář Petr*: *Interaktivní webové aplikace vytvořené pomocí technologie Adobe Flash - Teorie Kódování*. Brno, 2007. Bakalářská práce. Vyské učení technické v Brně.

[72] *Pacheco Peter*: *An Introduction to Parallel Programming*. Morgan Kaufmann, 2013. ISBN 978-0123742605.

[73] *Linder Marek*, *Pernecký Daniel*, *Sekaj Ivan*: *Grid Computing in Matlab for Solving Evolutionary*. Technical Computing Bratislava 2012 [elektronický zdroj] : 20th Annual Conference Proceedings.

[74] *Sekaj Ivan*, *Linder Marek*: *Evolutionary Algorithm Based Power Plant Working Point Optimisation Using PLC and Simulink Model*. Technical Computing Bratislava 2010 : 18th Annual Conference Proceedings. Bratislava, Slovak Republic, 20.10.2010. - Bratislava : RT Systems, 2010. - ISBN 978-80-970519-0-7.

[75] *Matoušek Radomil*, *Minář Petr*: *Stabilization of Chaotic Logistic Equation Using HC12 and Grammatical Evolution*. Advances in Intelligent Systems and Computing, 2013, roč. 2013, č. 210, s. 137-146. ISSN: 2194-5357.

[76] *Abbadi Ahmad*, *Matoušek Radomil*, *Minář Petr*, *Šoustek Petr*: *RRTs Review and Options. In Computational Engineering in Systems Applications*. Romania: IAASAT Press, 2011. s. 194-199. ISBN: 978-1-61804-026- 8.

[77] *Matoušek Radomil*, *Dobrovský Ladislav*, *Minář Petr*, *Mouralova Kateřina*: *A Note about Robust Stabilization of Chaotic Hénon System Using Grammatical Evolution*. Advances in Intelligent Systems and Computing, 2014.

[78] *Zelinka Ivan*: *Investigation on Realtime Deterministic Chaos Control by Means of Evolutionary Algorithms*. 1st IFAC Conference on Analysis and Control of Chaotic Systems, Reims, France, June 28-30, 2006.

[79] *Šenkeřík Roman*, *Zelinka Ivan*, *Davendra Donald*, *Oplatková Zuzana*: *Utilization of SOMA and differential evolution for robust stabilization of chaotic Logistic equation*. Journal Computers & Mathematics with Applications Volume 60 Issue 4, August, 2010 Pages 1026-1037.

[80] *T.Joines Eric*, *T.Joines William*: *Design of Yagi-Uda Antenna Using Genetic Algorithms*. IEEEAntennas Propag., Vol.45, NO.9, 1386- 1392, September.1997.

[81] *Matyáš Pavel*, *Minář Petr*: *Návrh Yagi- Uda antény pomocí evolučních algoritmů*. In Technical Computing Bratislava 2014. 2014. Slovakia, Bratislava: HUMUSOFT, 2014. s. 1-4. ISBN: 978-80-7080-898- 6.

[82] *Matyáš Pavel*, *Minář Petr*, *Ošmera Pavel*: *Design of Yagi- Uda antenna using HC12 algorithm*. In Mendel 2015. Mendel Journal series. 21th International Conference on Soft Computing. Mendel Journal series. 2. 2015. s. 143-149. ISBN: 978-3-319-19823- 1. ISSN: 1803- 3814.

[83] *Chapman Stephen J.*: *MATLAB Programming for Engineers*. CL Engineering; 4 edition, 2007. ISBN 049524449X.

[84] *Berger J E.*, *Nunes G.*: *A mechanical Duffing oscillator for the undergraduate laboratory*. American journal of physics / v.65 no.9. 1997, pp.841-.

[85] *Zuth Daniel*, *Minář Petr*: *Aplikace fuzzy množin pro určení technického stavu stroje*. Diago 2017. Technická diagnostika, ročník. 26. z1 s. 87-92. ISSN: 1210- 311X.

# SEZNAM ZKRATEK A SYMBOLŮ

| | |
|---|---|
| $CR$ | Parametr křížení $CR \in [0, 1]$ |
| $F$ | Koeficient mutace |
| $F_a$ | Akcelerační síla $[N]$ |
| $F_g$ | gravitační síla $[N]$ |
| $F_m$ | magnetická síla $[N]$ |
| $G$ | Gramatika jazyka |
| $I_{rand}$ | Náhodné číslo z $1, 2, \ldots N_{par}$ |
| $N$ | Konečná množina neterminálních symbolů |
| $NP$ | Velikost výpočetní populace |
| $N_{fb}$ | Počet mutujících bitů |
| $P$ | Množina přepisovacích pravidel |
| $R_j$ | Náhodně číslo $R_j \in (0, 1)$ |
| $S$ | Počáteční symbol, kdy $S \in N$ |
| $T$ | Konečná množina terminálních symbolů, kdy $N \cap T = 0$ |
| $\mu_A$ | Funkce příslušnosti k fuzzy množině A |
| $f_i$ | Hodnota fitness funkce i-tého jedince |
| $g_j$ | Omezující podmínky nerovnosti |
| $h_l$ | Omezující podmínky rovnosti |
| $i_{mag}$ | proud $[A]$ |
| $k_c$ | konstanta cívky $[Nm^2A^{-2}]$ |
| $k_v$ | Hodnota kodonu |
| $m_k$ | hmotnost kuličky $[kg]$ |
| $n_{bit}$ | Počet bitů v parametru |
| $p_{NP}$ | Počet produkčních pravidel |
| $p_{gen}$ | Hodnota parametru v genu |
| $p_{hi}$ | Maximalni hodnata parametru |
| $p_{id}$ | Index pravidla |
| $p_i$ | Pravděpodobnost výběru i-tého jedince |
| $p_{lo}$ | Minimalni hodnota parametru |
| $p_{norm}$ | Normalizovaný parametr, $0 \leq p_{norm} \leq 1$ |
| $p_n$ | Aktuální hodnota parametru |
| $x_e$ | Ektor parametrů optimalizace |
| $x_0$ | offset cívky $[m]$ |
| $x_{mag}$ | poloha kuličky $[m]$ |
| BNF | Backus-Naurova forma zápisu gramatiky - definice (sekce: 2.4 - viz str. 25) |
| DE | Diferenciální evoluce - definice (sekce: 2.3.3 - viz str. 20) |
| EA | Evoluční algoritmy |
| ETDAS | [ENG] Extended time delayed auto synchronization - definice (sekce: 9.1 - viz str. 117) |
| ETDASC | [ENG] Extended time delayed auto synchronization controlled - definice (sekce: 9.1 - viz str. 117) |
| FL | Fuzzy logika - definice (sekce: 2.5 - viz str. 27) |
| GA | Genetický algoritmus - definice (sekce: 2.3.2 - viz str. 17) |

GE . . . . . Gramatická evoluce - definice (sekce: 2.4 - viz str. 25)

HC . . . . . Horolezecký algoritmus - definice (sekce: 2.1 - viz str. 15)
HC12 . . . . . Algoritmus HC12 - definice (sekce: 2.3.4 - viz str. 22)
HIL . . . . . [ENG] Hardware in the loop

IAE . . . . . [ENG] Integral of the absolute magnitue of error - definice (sekce: 4.1.1 - viz str. 46)
ISE . . . . . [ENG] Integral of the square of the error - definice (sekce: 4.1.1 - viz str. 46)
ISTAE . . . . . [ENG] Integral of squared time-multiplied absolute value of error - definice (sekce: 4.1.1 - viz str. 46)
ITAE . . . . . [ENG] Integral of time-multiplied absolute value of error - definice (sekce: 4.1.1 - viz str. 46)

JMI . . . . . [ENG] Java to matlab interface

MOM . . . . . Momentová metoda

NEC2 . . . . . [ENG] Numerical Electromagnetic Code, Version 2
NM . . . . . Nelder-Mead simplexová metoda - definice (sekce: 2.2 - viz str. 15)
NT . . . . . Neuronové sítě

PID . . . . . Proporcionální, integrační a derivační regulátor
PLC . . . . . [ENG] Programmable logic controller
PVM . . . . . [ENG] Parallel Virtual Machine

RMI . . . . . [ENG] Java remote method

SC . . . . . Soft computing - definice (sekce: 2.3 - viz str. 16)
SU . . . . . Strojové učení

TDAS . . . . . [ENG] Time delayed auto synchronization - definice (sekce: 9.1 - viz str. 117)

UPO . . . . . [ENG] Unstable periodic orbit
UTD . . . . . [ENG] Uniform Geometric Theory of Diffraction

VKO . . . . . Vícekriteriální optimalizace - definice (sekce: 2 - viz str. 14)

ZN . . . . . Ziegler-Nichols metoda nastavení regulátoru

# SEZNAM OBRÁZKŮ

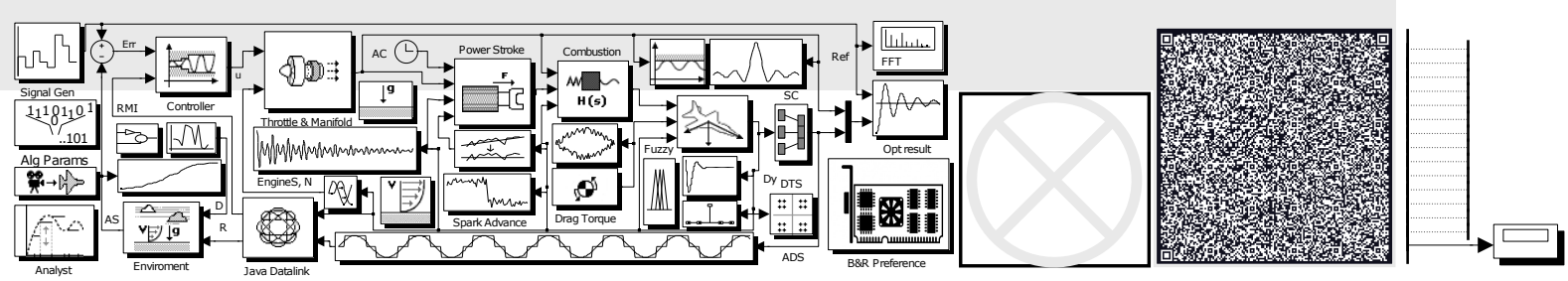
# SEZNAM TABULEK

# SEZNAM GRAFŮ

# SEZNAM MODELŮ

# SEZNAM GRAMATIK

# SEZNAM ALGORITMŮ

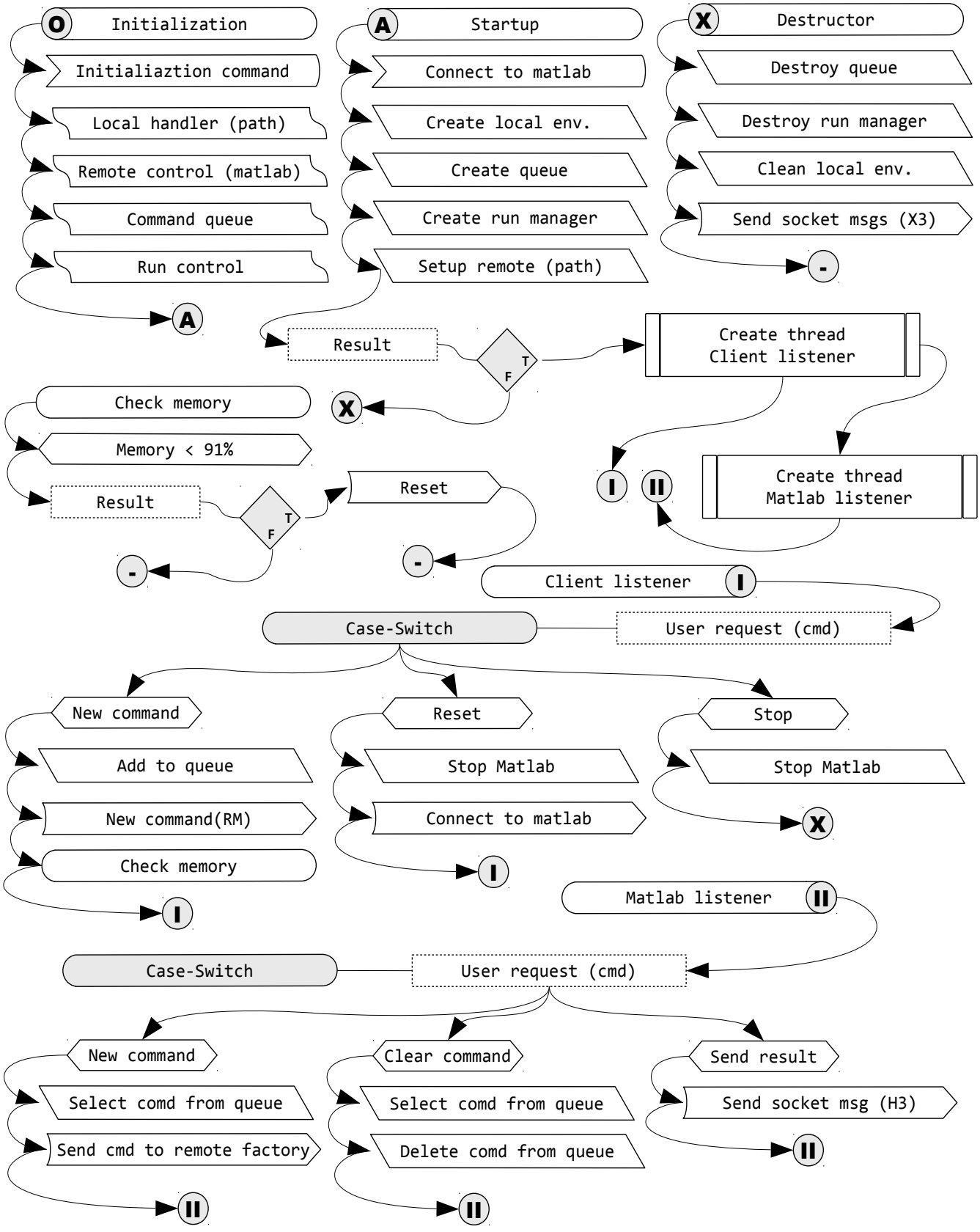



# KAPITOLA X

# PŘÍLOHY

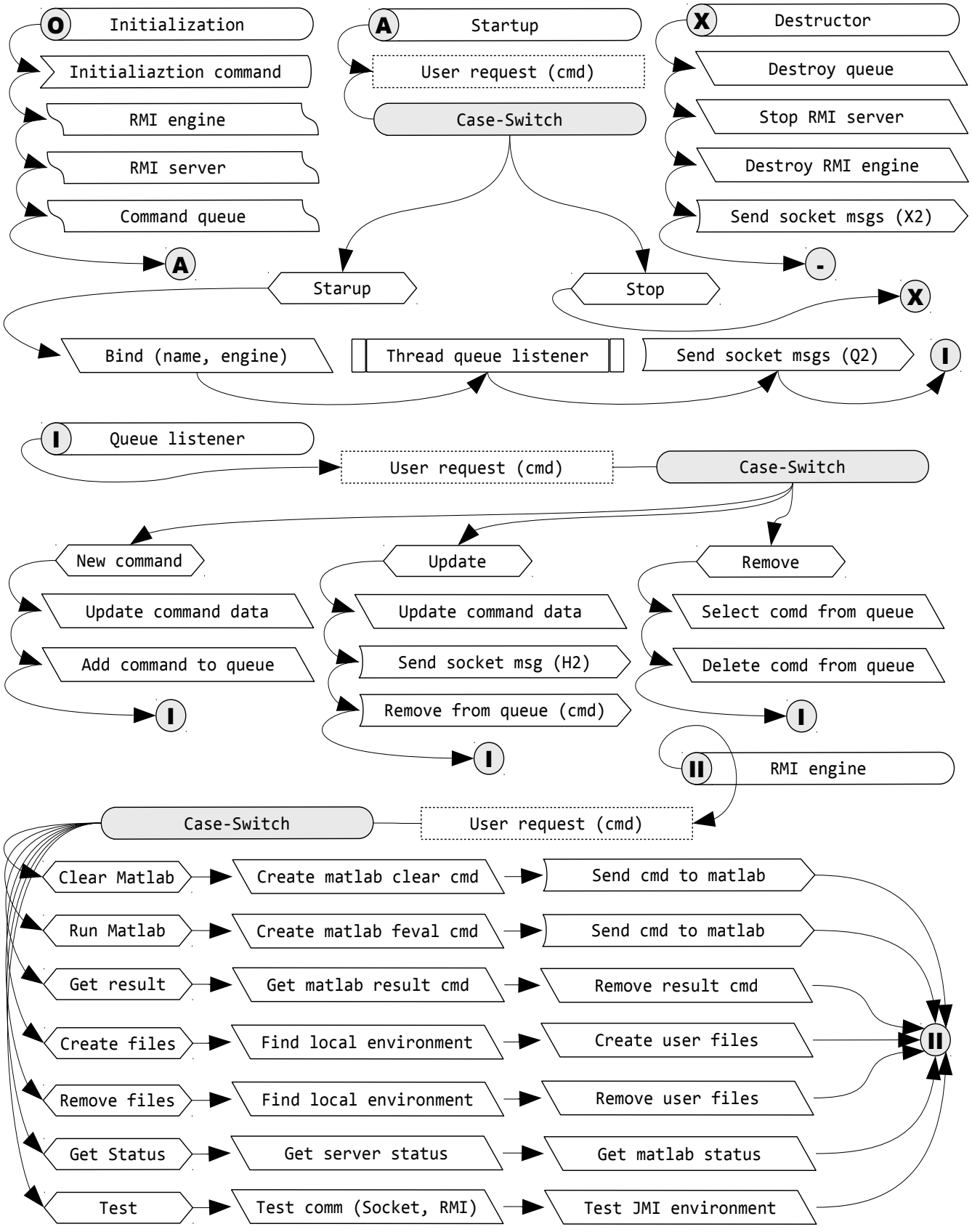
# X.1 Příloha A

**Příloha X.1:** [Server] Matlab handler.



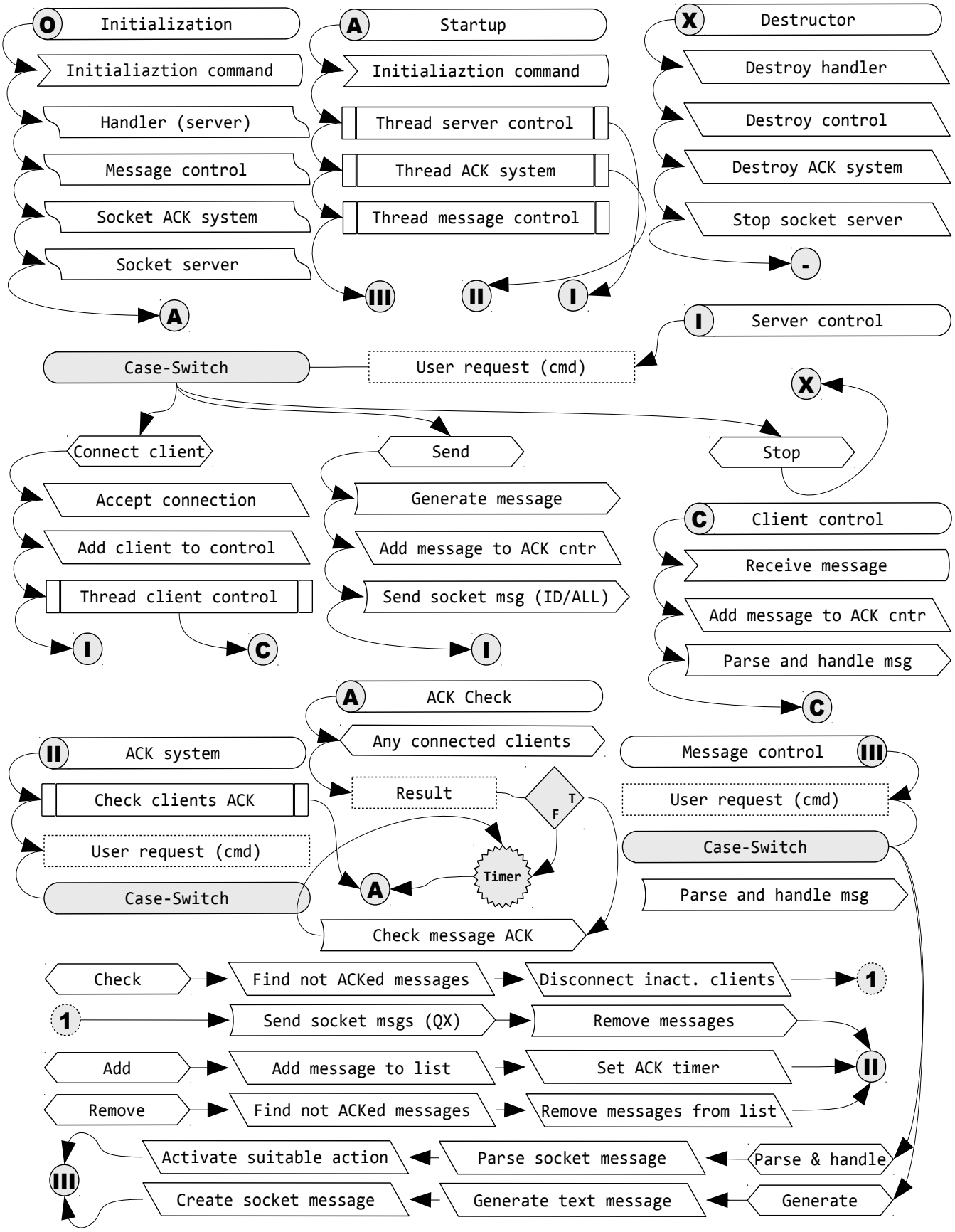
# X.2 Příloha B

**Příloha X.2:** [Server] RMI handler.



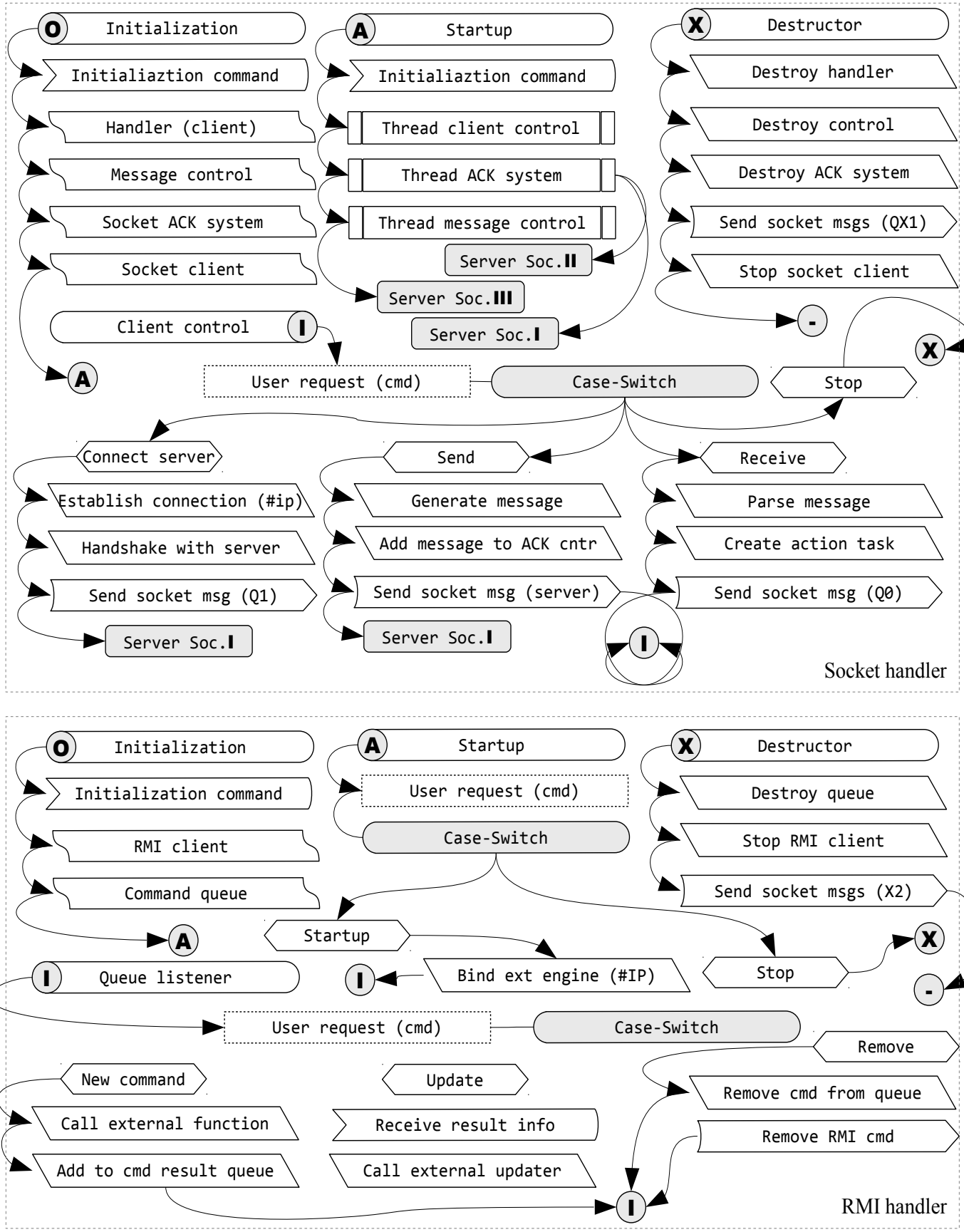
## X.3 Příloha C

**Příloha X.3:** [Server] Socket handler.



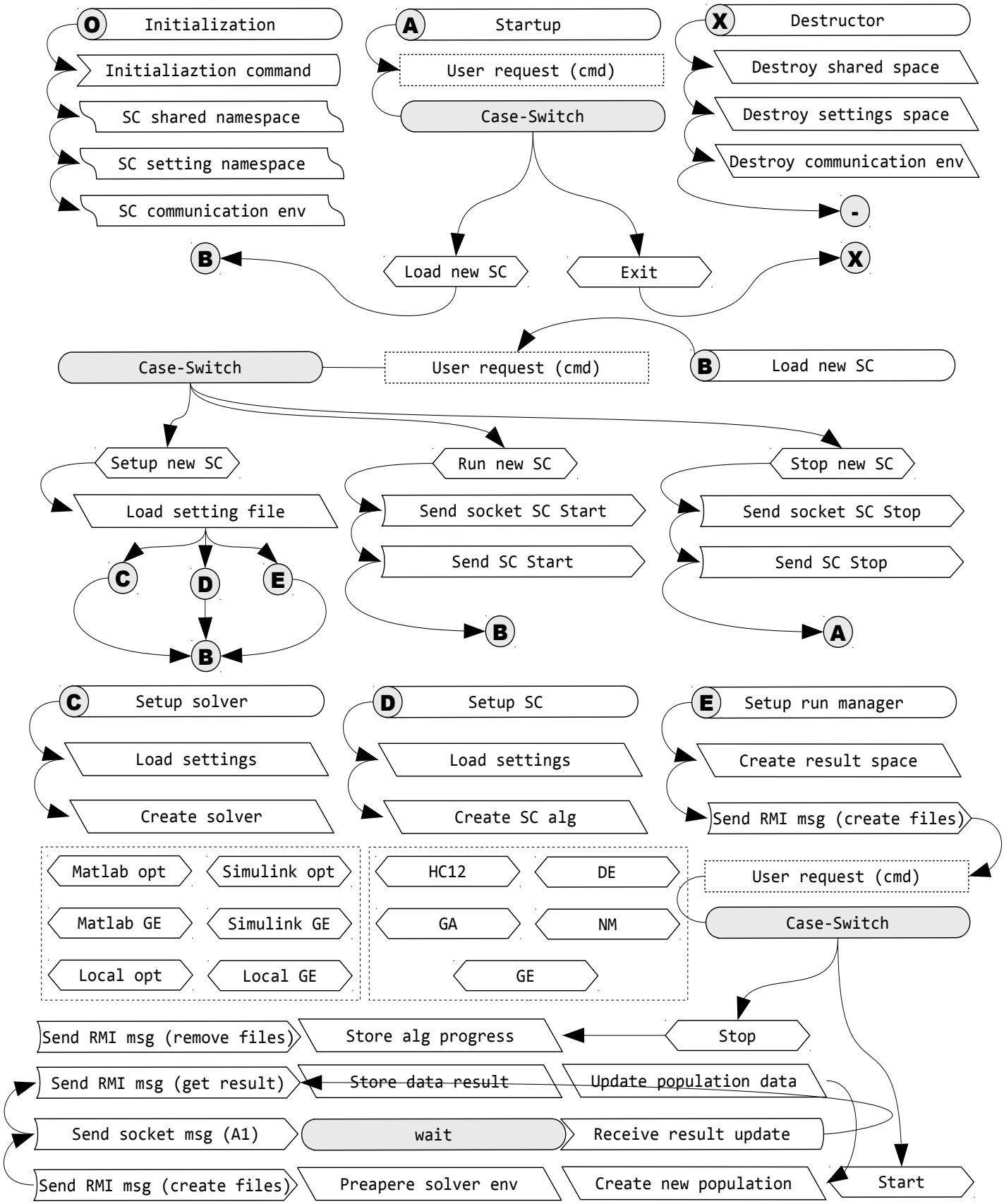
# X.4 Příloha D

**Příloha X.4:** [Client] Socket handler / RMI handler.



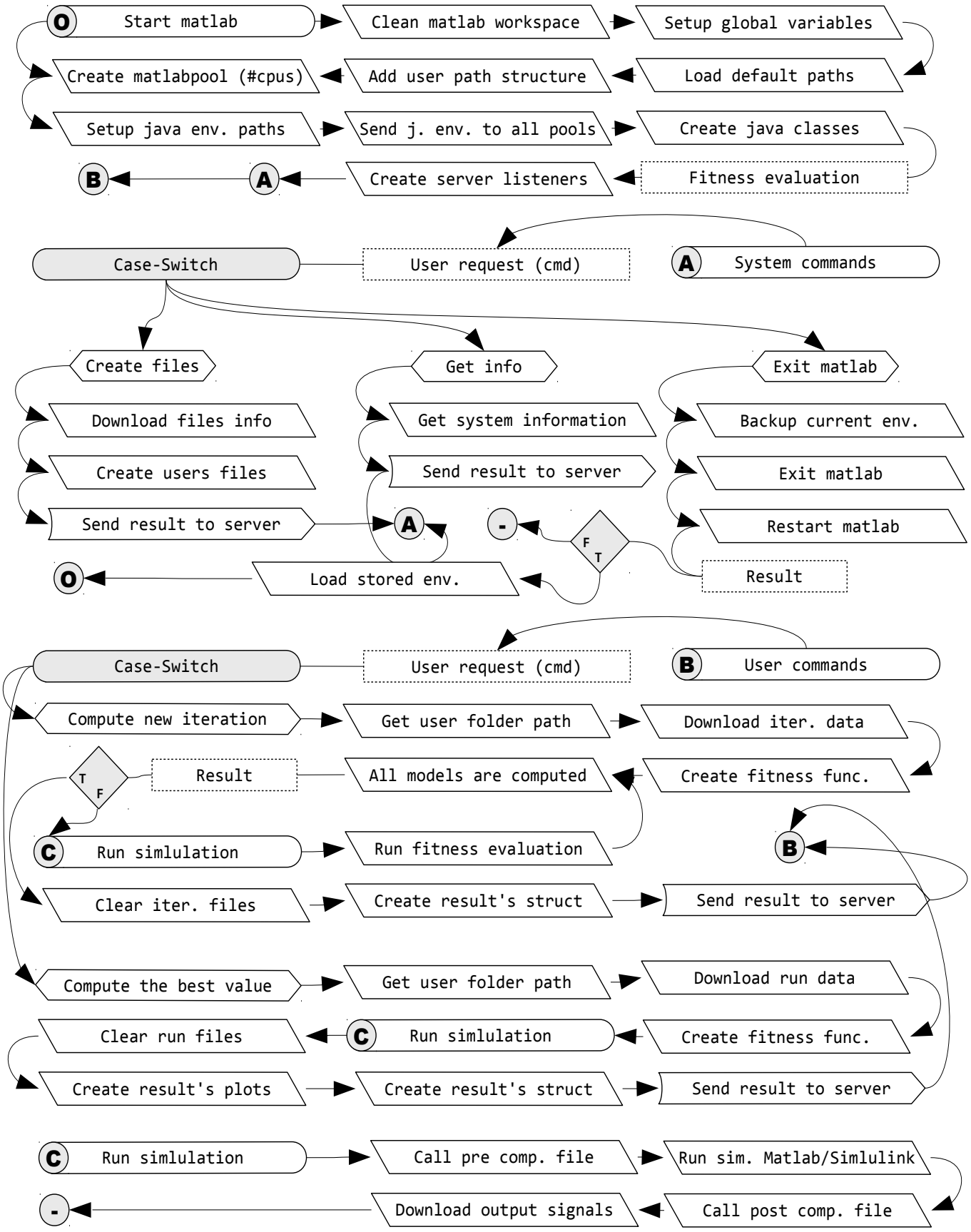
# X.5 Příloha E

**Příloha X.5:** [Client] General soft-computing algorithm execution.



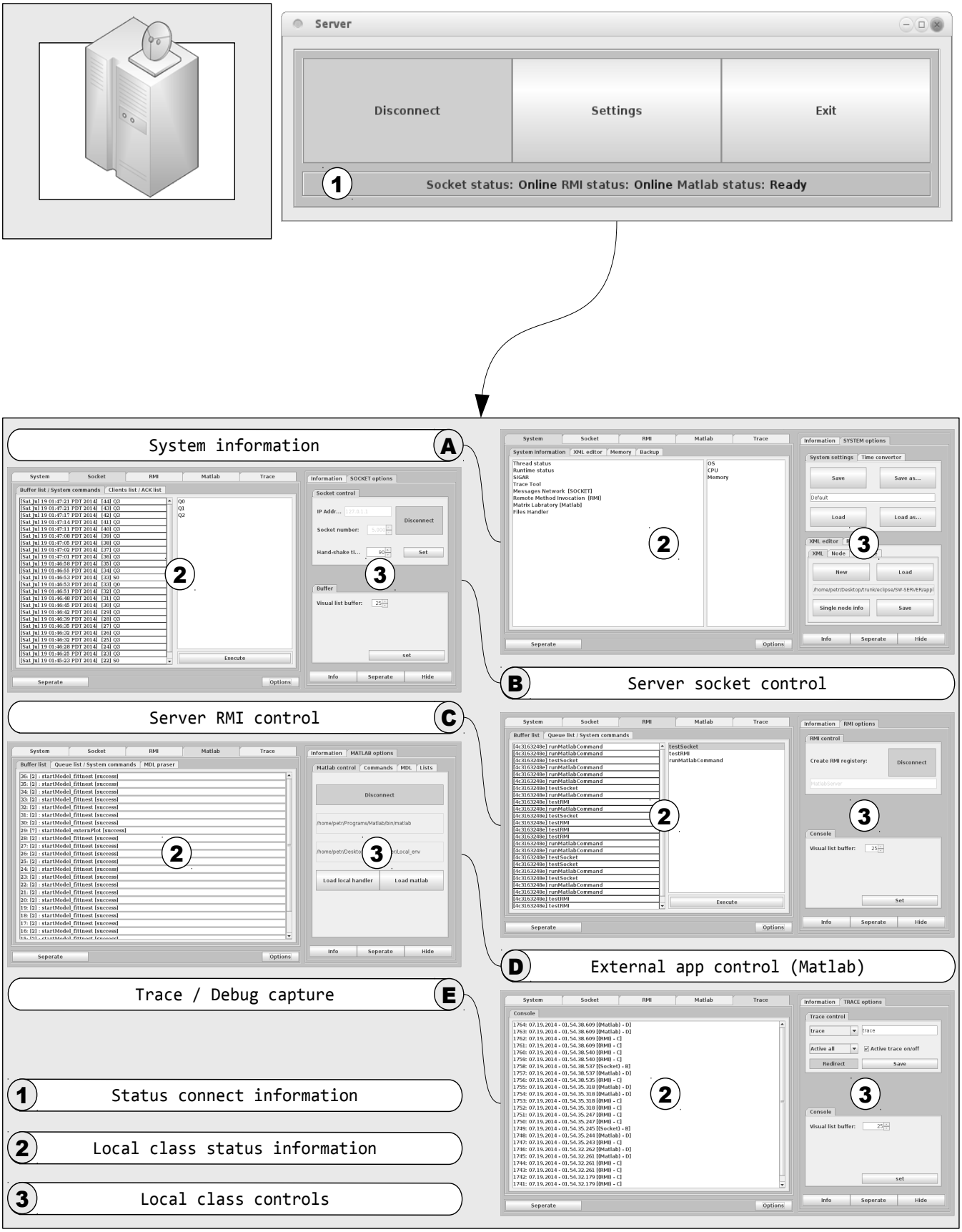
# X.6 Příloha F

**Příloha X.6:** [Server] Matlab system functions.



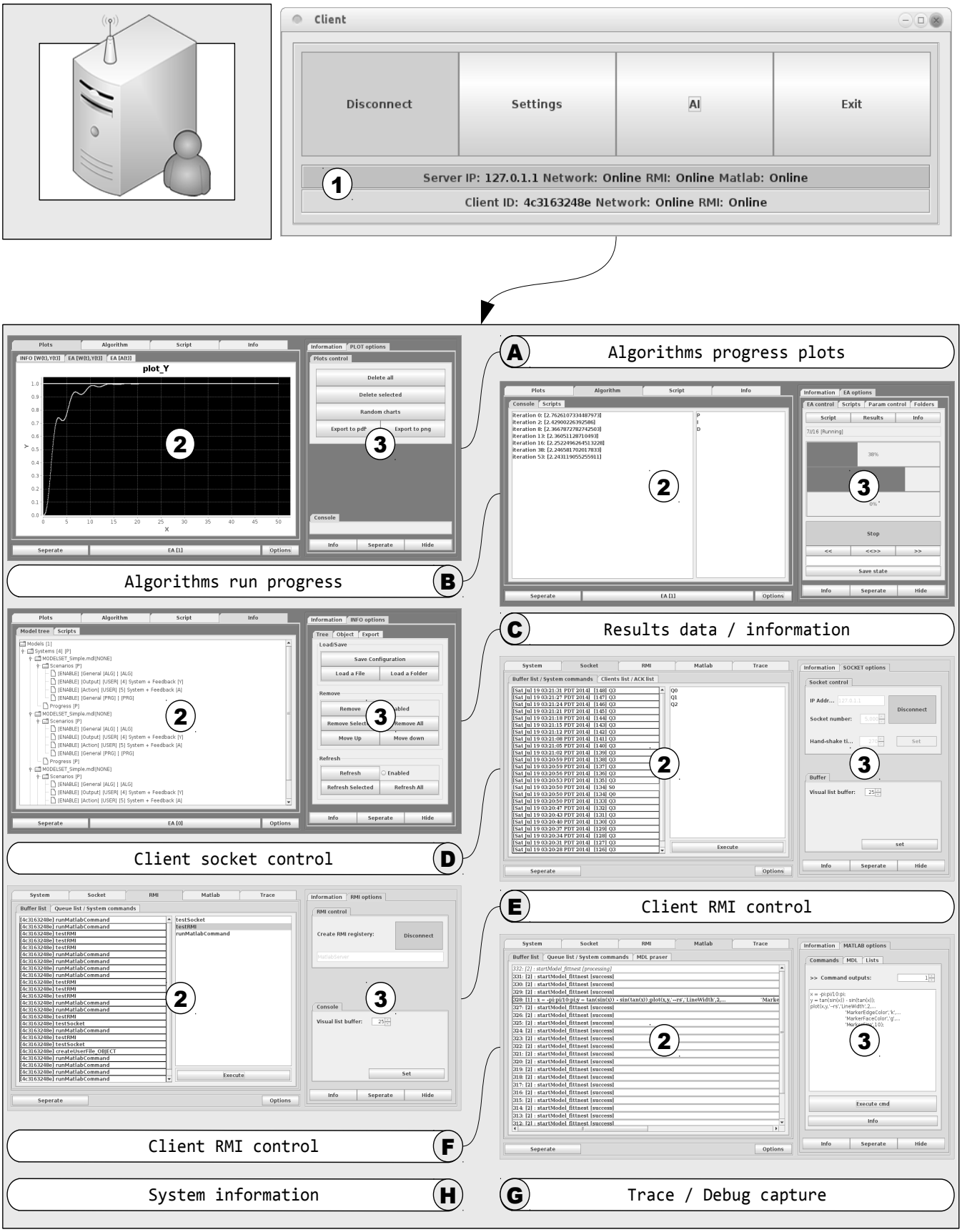
# X.7 Příloha G

**Příloha X.7:** [Server] Application screenshots.



# X.8 Příloha H

**Příloha X.8:** [Client] Application screenshots.



# X.9 Příloha I

**Příloha X.9:** Local environment setting example.

```
<?xml version="1.0" encoding="UTF-8"?>
<test name="test_env">
  <system>
    <repeats count="32" />
    <models name1="LOCALE_FCE_doc4_8.xml" name2="LOCALE_FCE_doc4_5.xml"
      name3="LOCALE_FCE_doc5_8.xml" name4="LOCALE_FCE_doc5_5.xml"
      name5="LOCALE_FCE_test1_GE.xml" name6="LOCALE_FCE_test2_GE.xml"
      name7="LOCALE_LETTER_test1_GE.xml" />
    <controllers />
    <subs name1="sub_LOCALE_FCE_doc4A5_8_HC12.xml"
      name2="sub_LOCALE_FCE_doc4A5_8_DE.xml"
      name3="sub_LOCALE_FCE_doc4A5_8_GA.xml"
      name4="sub_LOCALE_FCE_doc4A5_8_NM.xml"
      name5="sub_LOCALE_FCE_doc4A5_8_RAND.xml" name6="HC12_5" name7="DE_5"
      name8="GA_5" name9="NM_5" name10="RAND_5"
      name11="sub_LOCALE_FCE_test1_GE.xml"
      name12="sub_LOCALE_LETTER_test1_GE.xml" />
  </system>
  <runs>
    <!-- opt fce 4 all alg file sub -->
    <run ID="1-2" modelID="1" subID="1" />
    <run ID="2-4" modelID="1" subID="2" />
    <run ID="3-5" modelID="1" subID="3" />
    <run ID="6-7" modelID="1" subID="4" />
    <run ID="8-9" modelID="1" subID="5" />
    <!-- opt fce 4 all alg local sub -->
    <run ID="10-11" modelID="2" subID="6" />
    <run ID="12-13" modelID="2" subID="7" />
    <run ID="14-15" modelID="2" subID="8" />
    <run ID="16-17" modelID="2" subID="9" />
    <run ID="18-19" modelID="2" subID="10" />
    <!-- opt fce 5 all alg file sub -->
    <run ID="20" modelID="3" subID="1" />
    <run ID="21" modelID="3" subID="2" />
    <run ID="22" modelID="3" subID="3" />
    <run ID="23" modelID="3" subID="4" />
    <run ID="24" modelID="3" subID="5" />
    <!-- opt fce 5 all alg local sub -->
    <run ID="25" modelID="4" subID="6" />
    <run ID="26" modelID="4" subID="7" />
    <run ID="27" modelID="4" subID="8" />
    <run ID="28" modelID="4" subID="9" />
    <run ID="29" modelID="4" subID="10" />
    <!-- ge fce 1 -->
    <run ID="30" modelID="5" subID="11" />
    <!-- ge fce 2 -->
    <run ID="31" modelID="6" subID="11" />
    <!-- ge letters -->
    <run ID="32" modelID="7" subID="12" />
  </runs>
  <subs>
    <sub name="HC12_5" type="255" ai="0">
      <params>
        <param name="X1" min="-5" max="5" accurency="12" />
        <param name="X2" min="-5" max="5" accurency="12" />
        <param name="X3" min="-5" max="5" accurency="12" />
        <param name="X4" min="-5" max="5" accurency="12" />
        <param name="X5" min="-5" max="5" accurency="12" />
      </params>
    </sub>
    <sub name="DE_5" type="255" ai="1" ai_in1="40" ai_in2="0.9"
      ai_in3="0.5" ai_in4="2000">
      <!-- type="subtype" ai="AItype" in1="NP" in2="F" in3="CR" in4="GEN" -->
      <params>
        <param name="X1" min="-5" max="5" />
        <param name="X2" min="-5" max="5" />
        <param name="X3" min="-5" max="5" />
        <param name="X4" min="-5" max="5" />
        <param name="X5" min="-5" max="5" />
      </params>
    </sub>
    <sub name="GA_5" type="255" ai="2" ai_in1="40" ai_in2="0.2" ai_in3="0.5"
      ai_in4="-2" ai_in5="2000">
      <!-- type="subtype" ai="AItype" in1="NP" in2="F" in3="RATE" in4="CROSS"
        (< 0 == parameter acc constraint) in5="GEN" -->
      <params>
        <param name="X1" min="-5" max="5" accurency="10" />
        <param name="X2" min="-5" max="5" accurency="10" />
        <param name="X3" min="-5" max="5" accurency="10" />
        <param name="X4" min="-5" max="5" accurency="10" />
        <param name="X5" min="-5" max="5" accurency="10" />
      </params>
    </sub>
    <sub name="NM_5" type="255" ai="4" ai_in1="2000">
      <!-- type="subtype" ai="AItype" in1="GEN" -->
      <params>
        <param name="X1" min="-5" max="5" />
        <param name="X2" min="-5" max="5" />
        <param name="X3" min="-5" max="5" />
        <param name="X4" min="-5" max="5" />
        <param name="X5" min="-5" max="5" />
      </params>
    </sub>
    <sub name="RAND_5" type="255" ai="5" ai_in1="40" ai_in2="2000">
      <!-- type="subtype" ai="AItype" in1="NP" in2="GEN" -->
      <params>
        <param name="X1" min="-5" max="5" />
        <param name="X2" min="-5" max="5" />
        <param name="X3" min="-5" max="5" />
        <param name="X4" min="-5" max="5" />
        <param name="X5" min="-5" max="5" />
      </params>
    </sub>
  </subs>
</test>
```

test_local_environment.xml

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<sub name="sub_LOCALE_FCE_doc4A5_8_HC12" type="255" ai="0">
  <params>
    <param name="X1" min="-5" max="5" accurency="12"/>
    <param name="X2" min="-5" max="5" accurency="12"/>
    <param name="X3" min="-5" max="5" accurency="12"/>
    <param name="X4" min="-5" max="5" accurency="12"/>
    <param name="X5" min="-5" max="5" accurency="12"/>
    <param name="X6" min="-5" max="5" accurency="12"/>
    <param name="X7" min="-5" max="5" accurency="12"/>
    <param name="X8" min="-5" max="5" accurency="12"/>
  </params>
</sub>
```

sub_LOCALE_FCE_doc4A5_8_HC12.xml

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<sub name="sub_LOCALE_FCE_test2_GE" type="256" evalID="log10(iae(U,Y)+1)+log10(W)"
  ai="3" ai_in1="200" ai_in2="0.1" ai_in3="0.5" ai_in4="-2" ai_in5="200"
  ai_in6="4" ai_in7="grammatic_GE_fce.xml">
<!-- type="subtype" ai="AItype" in1="NP" in2="F" in3="RATE"
  in4="CROSS"(< 0 == parameter acc constraint) in5="GEN" in6="Wrapping"
  in7="Grammatic" -->
  <params>
    <param name="codom_1" accurency="8"/>
    <param name="codom_2" accurency="8"/>
    <param name="codom_3" accurency="8"/>
    <param name="codom_4" accurency="8"/>
    <param name="codom_5" accurency="8"/>
    <param name="codom_6" accurency="8"/>
    <param name="codom_7" accurency="8"/>
    <param name="codom_8" accurency="8"/>
    <param name="codom_9" accurency="8"/>
    <param name="codom_10" accurency="8"/>
    <param name="codom_11" accurency="8"/>
    <param name="codom_12" accurency="8"/>
    <param name="codom_13" accurency="8"/>
    <param name="codom_14" accurency="8"/>
    <param name="codom_15" accurency="8"/>
    <param name="codom_16" accurency="8"/>
    <param name="codom_17" accurency="8"/>
    <param name="codom_18" accurency="8"/>
    <param name="codom_19" accurency="8"/>
  </params>
</sub>
```

sub_LOCALE_FCE_test2_GE.xml

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<grammatic name="grammatic_GE_fce">
  <terminal>
    <T genetype="+" phenotype="+"/>
    <T genetype="-"/>
  </terminal>
  <nonterminal start="expr">
    <N name="expr">
      <P rule="[expr][op][expr]"/>
      <P rule="([expr][op][expr])"/>
      <P rule="[var][op][var]"/>
      <P rule="([var][op][var])"/>
      <P rule="[pre-op]([expr])"/>
      <P rule="[var]"/>
    </N>
    <N name="pre-op">
      <P rule="sin"/>
      <P rule="cos"/>
      <P rule="tan"/>
    </N>
    <N name="op">
      <P rule="+"/>
      <P rule="-"/>
      <P rule="*"/>
      <P rule="/"/>
    </N>
    <N name="var">
      <P rule="x"/>
      <P rule="1"/>
    </N>
  </nonterminal>
</grammatic>
```

grammatic_GE_fce.xml

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<test name="LOCALE_FCE_doc4_5">
  <fce>
    sum(100*(X[i]*X[i]-X[i+1])*(X[i]*X[i]-X[i+1]) + (1-X[i])*(1-X[i])){1:4,i}
  </fce>
</test>
```

LOCALE_FCE_doc4_5.xml

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<test name="LOCALE_FCE_doc5_5">
  <fce>
    10*5 + sum((X[i]*X[i])-(10*cos(2*PI*X[i]))) {1:5,i}
  </fce>
</test>
```

LOCALE_FCE_doc5_5.xml

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<test name="LOCALE_FCE_test1_GE">
  <fce start="-10" stop="10" step="0.1">
    (cos(x+1)+sin(x))*cos(2*x)
  </fce>
</test>
```

LOCALE_FCE_test1_GE.xml

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<test name="LOCALE_FCE_test2_GE">
  <fce start="-10" stop="10" step="0.1">
    cos(x)+sin(x)
  </fce>
</test>
```

LOCALE_FCE_test2_GE.xml

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<test name="LOCALE_LETTER_test1_GE">
  <string>
    hellohello
  </string>
</test>
```

LOCALE_LETTER_test1_GE.xml

# X.10 Příloha J

**Příloha X.10:** Matlab environment setting example.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes" ?>
<test name="test_matlab">
  <system>
    <repeats count="4"/>
    <models
      name1="MATLAB_Simple_TPID_step1.m"
      name2="MATLAB_DC_P4_LM1.m"
    />
    <controllers/>
    <sources/>
    <subs
      name1="sub_MATLAB_COMMON_TPID_DE_Simple0.xml"
      name2="sub_MODELSET_SIMPLE1_matlab0_GE.xml"
      name3="sub_MODELSET_Chaos_matlab0_GE.xml"
      name4="sub_MATLAB_COMMON_DE_TDAS_LM0.xml"
    />
  </system>
  <runs>
    <!-- test PID settings 1 -->
    <run ID="1" subID="1" modelID="1"
      evalID="1*log10(itae(T,U,Y))+1*log10(over(U,Y)+1)+1*log10(wave(Y)+1)"/>
    <!-- test GE control settings 1 -->
    <run ID="2" subID="2" modelID="1"
      evalID="1*log10(itae(T,U,Y))+1*log10(over(U,Y)+1)+1*log10(wave(Y)+1)"/>
    <!-- test GE chaos settings 1 -->
    <run ID="3" subID="3" modelID="2"
      evalID="itae(T,U,Y)+filter_line(0,Y)"/>
    <!-- test chaos ETDAS settings 1 -->
    <run ID="4" subID="4" modelID="2"
      evalID="itae2(T,U,Y)+filter_line(0,Y)"/>
  </runs>
</test>
```

test_matlab_environment.xml

```
%% [GLOBAL] %%
  time_start = 0; time_end = 50; time_sample = 1001;
  array_T_submodel=linspace(time_start,time_end,time_sample);
%% /[GLOBAL] %%
%% [SOURCE] %%
  array_U_submodel=ones(1, length(array_T_submodel)) + 0.0;
%% /[SOURCE] %%
%% [SYSTEM] %%
  sys = tf([6],[48 44 12 1]);
%% /[SYSTEM] %%
%% [CONTROL] %%
  CONTROL_result = tf(gainP, 1) + tf(1, [gainI, 0]) + tf([gainD, 0], 1);
%% /[CONTROL] %%

reg = CONTROL_result;

sys_final = feedback(sys*reg,1);
array_Y_submodel = lsim(sys_final,array_U_submodel,array_T_submodel);
array_Y_submodel = array_Y_submodel';
array_A_submodel = zeros(1, length(array_Y_submodel));
```

MATLAB_Simple_TPID_step1.m

```
%% [GLOBAL] %%
r = 3.8;
M = 150; % number of iterations of logistics equation
array_T_submodel=1:M;
%% /[GLOBAL] %%

%% [SOURCE] %%
Xf(1) = 0.3038; Xf(2) = 0.8037; Xf(3) = 0.5995; Xf(4) = 0.9124;
array_U_submodel=zeros(length(Xf), length(array_T_submodel));
for i=1:length(Xf)
    array_U_submodel(i,:) = zeros(1, length(array_T_submodel)) + Xf(i);
end
%% /[SOURCE] %%

x=zeros(M,1); % allocate memory
s=zeros(M,1); % allocate memory

x(1) = param_X1; % initial condition (can be anything from 0 to 1)
s(1) = 1;

UPO = length(Xf);

for n = 2:M, % iterate
    %% [CONTROL] %%
    s(n-1) = x(n-1) + param_R*s( index_check( n-1-UPO ) );

    Fn = param_K*((1-param_R)*s( index_check(n-1-UPO) ) - x( index_check(n-1)) );
    CONTROL_result = Fn;
    %% /[CONTROL] %%

    if(CONTROL_result > (param_Fmax/2)), CONTROL_result = (param_Fmax/2); end
    if(CONTROL_result < -(param_Fmax/2)), CONTROL_result = -(param_Fmax/2); end

    x(n) = r*x(n-1)*(1-x(n-1))+CONTROL_result;
end

array_Y_submodel = x';
array_A_submodel = zeros(1, length(array_Y_submodel));
```

MATLAB_DC_P4_LM1.m

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<sub name="sub_MATLAB_COMMON_TPID_DE_Simple0" type="-100" ai="1" ai_in1="15"
  ai_in2="0.9" ai_in3="0.5" ai_in4="5" evalID="itae(T,U,Y)">
<!-- ai="AItype" in1="NP" in2="F" in3="CR" in4="GEN" -->
  <params>
    <param name="gainP" min="0" max="30"/>
    <param name="gainI" min="0" max="30"/>
    <param name="gainD" min="0" max="30"/>
  </params>
</sub>
```

sub_MATLAB_COMMON_TPID_DE_Simple0.xml

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<sub name="sub_MODELSET_SIMPLE1_matlab0_GE" type="-103" ai="3" ai_in1="15" ai_in2="0.2"
  ai_in3="0.5" ai_in4="-1" ai_in5="5" ai_in6="30" ai_in7="grammatic_GE_simple1_matlab2.xml"
  evalID="iae(U,Y)+filter_line(Y)">
<!-- type="subtype" ai="AItype" in1="NP" in2="F" in3="RATE" in4="CROSS"
  (< 0 == parameter acc constraint) in5="GEN" in6="Wrapping" in7="Grammatic" -->
  <params>
    <!-- codoms -->
  </params>
</sub>
```

sub_MODELSET_SIMPLE1_matlab0_GE.xml

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<grammatic name="grammatic_GE_simple1_matlab2">
  <terminal>
    <!-- genetype=Gramatic expresion; phenotype=Matlab expresion;
    if missing phenotype phenotype=genetype -->
  </terminal>
  <nonterminal start="expr">
    <N name="expr">
      <P rule="({tr-func}{op}{tr-func})"/> <P rule="({var}{op}{tr-func})"/>
    </N>
    <N name="expr2">
      <P rule="({expr2}{op}{expr2})"/> <P rule="({var}{op}{var})"/> <P rule="{var}"/>
      <P rule="{vector}"/>
    </N>
    <N name="vector">
      <P rule="[{expr2}, {expr2}]"/>
    </N>
    <N name="tr-func">
      <P rule="tf({vector}, {vector})"/>    <!-- TRF -->
      <P rule="tf({vector}, {vector}){op}{expr}"/>  <!-- TRF -->
    </N>
    <N name="op">
      <P rule="+"/> <P rule="-"/> <P rule="*"/> <P rule="/"/>
    </N>
    <N name="var">
      <P rule="-1"/> <P rule="0"/> <P rule="1"/> <P rule="2"/> <P rule="3"/> <P rule="4"/>
      <P rule="5"/> <P rule="6"/> <P rule="7"/> <P rule="8"/> <P rule="9"/>
    </N>
  </nonterminal>
</grammatic>
```

grammatic_GE_simple1_matlab2.xml

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<sub name="sub_MODELSET_Chaos_matlab0_GE" type="-103" ai="3" ai_in1="15" ai_in2="0.2"
  ai_in3="0.5" ai_in4="-2" ai_in5="5" ai_in6="60" ai_in7="grammatic_GE_chaos1_matlab.xml"
  evalID="iae(U,Y)+filter_line(Y)">
<!-- type="subtype" ai="AItype" in1="NP" in2="F" in3="RATE" in4="CROSS"
  (< 0 == parameter acc constraint) in5="GEN" in6="Wrapping" in7="Grammatic" -->
  <params>
    <!-- codoms -->
  </params>
</sub>
```

sub_MODELSET_Chaos_matlab0_GE.xml

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<grammatic name="grammatic_GE_chaos1_matlab">
  <terminal>
    <!-- genetype=Gramatic expresion; phenotype=Matlab expresion;
      if missing phenotype phenotype=genetype -->
  </terminal>
  <nonterminal start="start">
    <N name="start">
      <P rule="({weight}*{expr}{op2}{expr})"/>
    </N>
    <N name="weight">
      <P rule="({weight}{op}{weight})"/> <P rule="({var}{op}{weight})"/>
      <P rule="({weight}{op}{var})"/> <P rule="({var}{op}{var})"/>
    </N>
    <N name="expr">
      <P rule="({expr}{op2}{expr})"/> <P rule="({var}{op2}{expr})"/>
      <P rule="{pre-op}{op2}{expr}"/> <P rule="{pre-op}{op2}{pre-op}"/> <P rule="{expr}"/>
    </N>
    <N name="expr2">
      <P rule="({var2}{op3}{var2})"/>
      <P rule="{var2}"/>
    </N>
    <N name="pre-op">
      <P rule="x( index_check( n - {expr2} ) )"/>  <!-- INT -->
    </N>
    <N name="op">
      <P rule="+"/> <P rule="-"/> <P rule="/"/> <P rule="*"/>
    </N>
    <N name="op2">
      <P rule="+"/> <P rule="-"/> <P rule="/"/> <P rule="*"/>
    </N>
        <N name="op3">
          <P rule="+"/>
        </N>
        <N name="var">
    <P rule="1"/> <P rule="2"/> <P rule="3"/> <P rule="4"/> <P rule="5"/> <P rule="6"/>
    <P rule="7"/> <P rule="8"/> <P rule="9"/>
    </N>
    <N name="var2">
      <P rule="1"/> <P rule="{expr2}"/>
    </N>
  </nonterminal>
</grammatic>
```

grammatic_GE_chaos1_matlab.xml

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<sub name="sub_MATLAB_COMMON_DE_TDAS_LM0" type="-100" ai="1" ai_in1="15" ai_in2="0.9"
  ai_in3="0.5" ai_in4="5" evalID="itae2(T,U,Y)">
<!-- ai="AItype" in1="NP" in2="F" in3="CR" in4="GEN" -->
  <params>
    <param name="param_K" min="-2" max="2"/>
    <param name="param_Fmax" min="0" max="1"/>
    <param name="param_R" min="0" max="1"/>
    <param name="param_X1" min="0" max="1"/>
  </params>
</sub>
```

sub_MATLAB_COMMON_DE_TDAS_LM0.xml

# X.11 Příloha K

**Příloha X.11:** Simulink environment setting example.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes" ?>
<test name="test_simulink">
 <system>
  <repeats count="15"/>
  <models
    name1="MODELSET_Simple.mdl"
    name2="MODELSET_FCE_test1_GE.mdl"
    name3="MODELSET_DC_LM1.mdl"
  />
  <controllers
    name1="script_Common_TPID.m"
    name2="script_Fuzzy_PID_3MF_FPID_Simple1.m"
    name3="controller_Common_ZPC.mdl"
  />
  <sources
    name1="SOURCE_step.mdl"
    name2="SOURCE_step_gauss.mdl"
  />
  <subs
    name1="sub_COMMON_TPID_HC12_Simple1.xml"
    name2="HC12_PID"
    name3="DE_PID"
    name4="sub_PARAMSET_fPID3_DE_Peak_Var_Output3.xml"
    name5="sub_COMMON_ZP_DE_test.xml"
    name6="sub_MODELSET_FCE_test2_GE.xml"
    name7="sub_MODELSET_Control_test2_GE.xml"
  />
 </system>
 <runs>
  <!-- test ZP settings 1 -->
  <run ID="1" subID="5" controllerID="3" modelID="1"/>
  <!-- test local settings -->
  <run ID="1" sourceID="1" controllerID="1" subID="2">
    <param name="D" value="6" accurency="6" enabled="0"/>
    </run>
  <!-- test HC12 file /  + - value -->
  <run ID="2" sourceID="1" controllerID="1" subID="2" flSub="1"/>
  <run ID="3" modelID="1" controllerID="1" sourceID="1" subID="1"/>
  <!-- test local settings -->
  <run ID="4" subID="-1"/>
  <run ID="5-6">
    <param name="I" value="0" enabled="0"/>
    <param name="D" bin="01"/>
  </run>
  <run ID="7" sourceID="1" subID="-1">
    <param name="P" inherited="1"/>
    <param name="I" enabled="1"/>
  </run>
  <run ID="8" sourceID="2"  controllerID="1" subID="2"
    evalID="1*log10(itae(T,U,Y))+1*log10(over(U,Y)+1)+1*log10(wave(Y)+1)">
       <param name="D" value="1" enabled="0"/>
    </run>
  <!-- test DE local /  + - value -->
  <run ID="9" modelID="1" controllerID="1" sourceID="2" subID="3"/>
  <run ID="10" subID="-3"/>
  <!-- fuzzy test 1 -->
  <run ID="11" subID="-4" controllerID="2" modelID="1"/>
  <!-- test ZP settings 1 -->
  <run ID="12" subID="5" controllerID="3" modelID="1"/>
  <!-- test GE fce -->
  <run ID="13" modelID="2" subID="6"/>
  <!-- test GE control -->
  <run ID="14" modelID="1" subID="7"/>
 </runs>
<subs>
  <sub name="HC12_PID" type="0" ai="0" evalID="itae(T,U,Y)">
    <params>
      <param name="P" min="5" max="50" accurency="2" enabled="1"/>
      <param name="I" min="0" max="256" accurency="3" enabled="0"/>
      <param name="D" min="0" max="8" accurency="4"/>
    </params>
  </sub>
  <sub name="DE_PID" type="-1" ai="1" ai_in1="40" ai_in2="0.9" ai_in3="0.5"
    ai_in4="15" evalID="itae(T,U,Y)">
  <!-- type="subtype" ai="AItype" in1="NP" in2="F" in3="CR" in4="GEN" -->
    <params>
      <param name="P" min="5" max="50" enabled="1"/>
      <param name="I" min="0" max="256" enabled="1"/>
      <param name="D" min="0" max="8" enabled="1"/>
    </params>
  </sub>
</subs>
</test>
```

test_simulink_environment.xml

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<sub name="sub_PARAMSET_fPID3_DE_Peak_Var_Output3" type="0" ai="1" ai_in1="100"
  ai_in2="0.9" ai_in3="0.5" ai_in4="2" evalID="itae(T,U,Y)">
<!-- ai="AItype" in1="NP" in2="F" in3="CR" in4="GEN" -->
  <params>
    <param name="gainP" min="0" max="1000"/>
    <param name="gainI" min="0" max="1000"/>
    <param name="gainD" min="0" max="1000"/>
    <param name="gainY" min="0" max="1"/>
    <param name="Peak_input_1_1" min="-100" max="100"/>
    <!-- copy: Peak_input_1_2 Peak_input_1_3 Peak_input_2_1 Peak_input_2_2 Peak_input_2_3
    Peak_input_3_1 Peak_input_3_2 Peak_input_3_3 -->

    <param name="Variance_input_1_1" min="1" max="100"/>
    <!-- copy: Variance_input_1_2 Variance_input_1_3 Variance_input_2_1
    Variance_input_2_2 Variance_input_2_3 Variance_input_3_1
    Variance_input_3_2 Variance_input_3_3 -->

    <param name="Peak_output_1_1" min="-100" max="100"/>
    ***
    <param name="Peak_output_1_11" min="-100" max="100"/>
  </params>
</sub>
```

sub_PARAMSET_fPID3_DE_Peak_Var_Output3.xml

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<grammatic name="grammatic_GE_fce_simulink">
  <terminal>
    <T genetype="1" phenotype="Block.Constant.Value=$1$"/>
    <T genetype="input_1" phenotype="Object.input1"/>
    <T genetype="t" phenotype="Block.Clock"/>
    <T genetype="+" phenotype="Block.Sum.Inputs=$++$"/>
    <T genetype="-" phenotype="Block.Sum.Inputs=$+-$"/>
    <T genetype="*" phenotype="Block.Product.Inputs=$**$"/>
    <T genetype="/" phenotype="Block.Product.Inputs=$*/$"/>
    <T genetype="sin" phenotype="Block.Trigonometry.Operator=$sin$"/>
    <T genetype="cos" phenotype="Block.Trigonometry.Operator=$cos$"/>
  </terminal>
  <nonterminal start="node">
    <N name="node">
      <P rule="{[node][block_input_1]}"/> <P rule="{[node][node][block_input_2]}"/>
      <P rule="{[source]}"/>
    </N>
    <N name="block_input_1">
      <P rule="sin"/> <P rule="cos"/>
    </N>
    <N name="block_input_2">
      <P rule="+"/> <P rule="-"/> <P rule="*"/> <P rule="/"/>
    </N>
    <N name="source">
      <P rule="t"/> <P rule="input_1"/> <P rule="1"/>
    </N>
  </nonterminal>
</grammatic>
```

grammatic_GE_fce_simulink.xml

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<grammatic name="grammatic_GE_control1_simulink">
  <terminal>
    <T genetype="input_1" phenotype="Object.input1"/>
    <T genetype="1" phenotype="Block.Constant.Value=$1$"/>
    ***
    <T genetype="9" phenotype="Block.Constant.Value=$9$"/>
    <T genetype="int" phenotype="Block.Integrator"/>
    <T genetype="der" phenotype="Block.Derivative"/>
    <T genetype="+" phenotype="Block.Sum.Inputs=$++$"/>
    <T genetype="-" phenotype="Block.Sum.Inputs=$+-$"/>
    <T genetype="*" phenotype="Block.Product.Inputs=$**$"/>
    <T genetype="/" phenotype="Block.Product.Inputs=$*/$"/>
  </terminal>
  <nonterminal start="init">
    <N name="init">
      <P rule="{[node][block_input_1]}"/> <P rule="{[node][node][block_input_2]}"/>

    </N>
    <N name="node">
      <P rule="{[init]}"/> <P rule="{[inputs]}"/> <P rule="{[consts]}"/>
    </N>
    <N name="block_input_1">
      <P rule="int"/> <P rule="der"/>
    </N>
    <N name="block_input_2">
      <P rule="+"/> <P rule="-"/> <P rule="*"/> <P rule="/"/>
    </N>
    <N name="inputs">
      <P rule="input_1"/>
    </N>
    <N name="consts">
      <P rule="1"/> <P rule="2"/> <P rule="3"/> <P rule="4"/> <P rule="5"/> <P rule="6"/>
      <P rule="7"/> <P rule="8"/> <P rule="9"/>
    </N>
  </nonterminal>
</grammatic>
```

grammatic_GE_control1_simulink.xml

```
<sub name="sub_MODELSET_FCE_test2_GE" type="3" ai="3" ai_in1="10" ai_in2="0.2" ai_in3="0.5"
  ai_in4="-2" ai_in5="15" ai_in6="3" ai_in7="grammatic_GE_fce_simulink.xml"
  evalID="iae(U,Y)+filter_line(Y)">
```

sub_MODELSET_FCE_test2_GE.xml

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<sub name="sub_MODELSET_Control_test2_GE" type="3" ai="3" ai_in1="30" ai_in2="0.2" ai_in3="0.5"
  ai_in4="-2" ai_in5="5" ai_in6="10" ai_in7="grammatic_GE_control1_simulink.xml"
  evalID="iae(U,Y)+filter_line(Y)">
```

sub_MODELSET_Control_test2_GE.xml

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<sub name="sub_COMMON_ZP_DE_test" type="2" sub_in1="-50" sub_in2="-600" sub_in3="60"
  ai="1" ai_in1="50" ai_in2="0.9" ai_in3="0.5" ai_in4="15" evalID="itae(T,U,Y)">
<!-- ai="AItype" in1="NP" in2="F" in3="CR" in4="GEN" -->
<!-- type="subtype" in1="real_shadowZone" in2="real_deadZone"
  in3="img_shadowZoneAngle" [-1 = dont use img_shadowZoneAngle] -->
  <params>
    <param name="Gain" min="0" max="10"/>
    <param name="Zero1" min="-60" max="0"/> <param name="Zero2" min="0" max="30" bound="Zero1"/>
    <param name="Zero3" min="-60" max="0"/> <param name="Zero4" min="0" max="30" bound="Zero3"/>
    <param name="Zero5" min="-60" max="0"/> <param name="Zero6" min="0" max="30" bound="Zero5"/>
    <param name="Pole1" min="-60" max="0"/> <param name="Pole2" min="0" max="30" bound="Pole1"/>
    <param name="Pole3" min="-60" max="0"/> <param name="Pole4" min="0" max="30" bound="Pole3"/>
    <param name="Pole5" min="-60" max="0"/> <param name="Pole6" min="0" max="30" bound="Pole5"/>
  </params>
</sub>
```

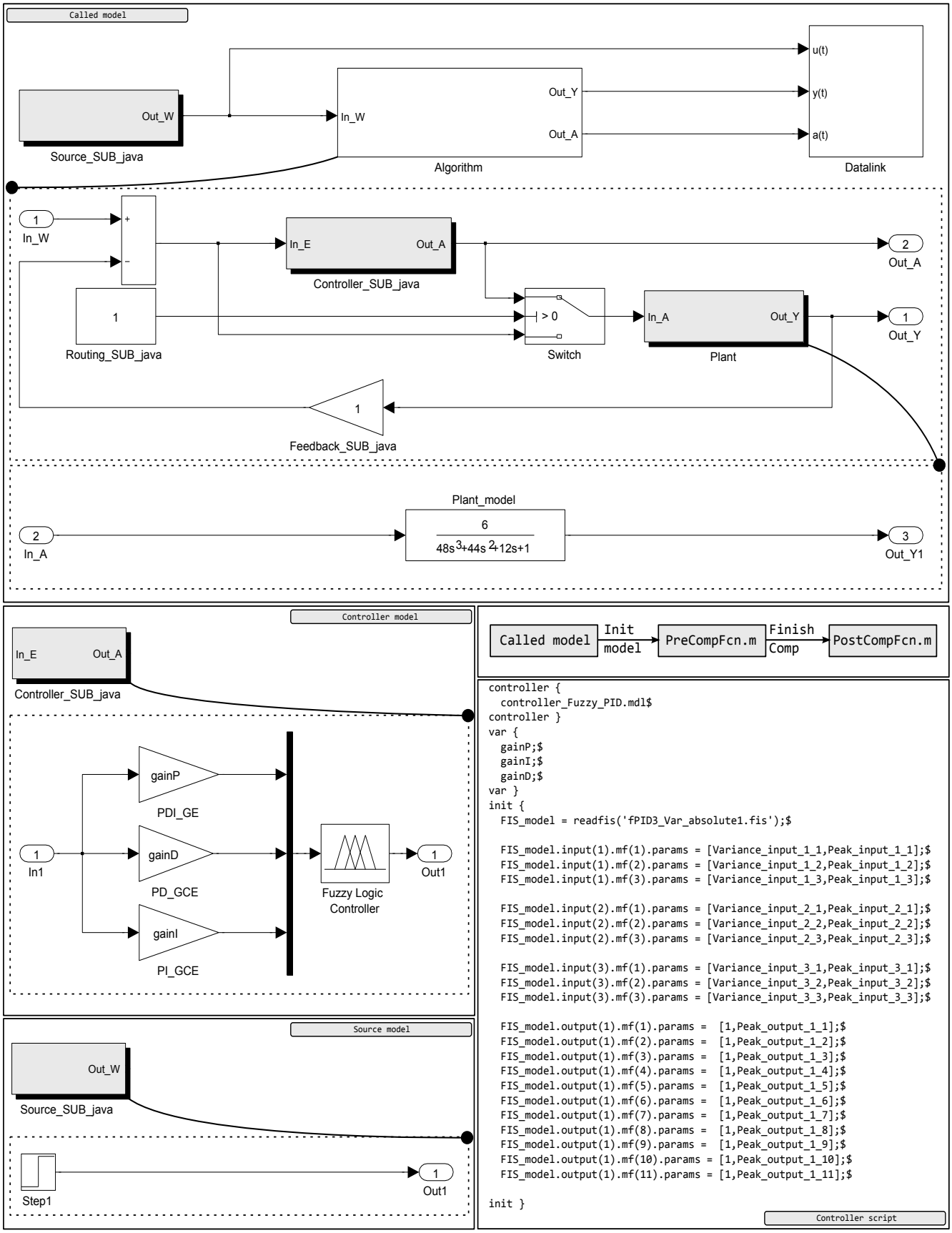
sub_COMMON_ZP_DE_test.xml

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<sub name="sub_COMMON_TPID_HC12_Simple1" type="1" ai="0" evalID="itae(T,U,Y)">
  <params>
    <param name="P" min="0" max="50" accurency="6"/>
    <param name="I" min="1" max="256" accurency="6"/>
    <param name="D" min="0" max="8" accurency="6"/>
  </params>
</sub>
```

sub_COMMON_TPID_HC12_Simple1.xml

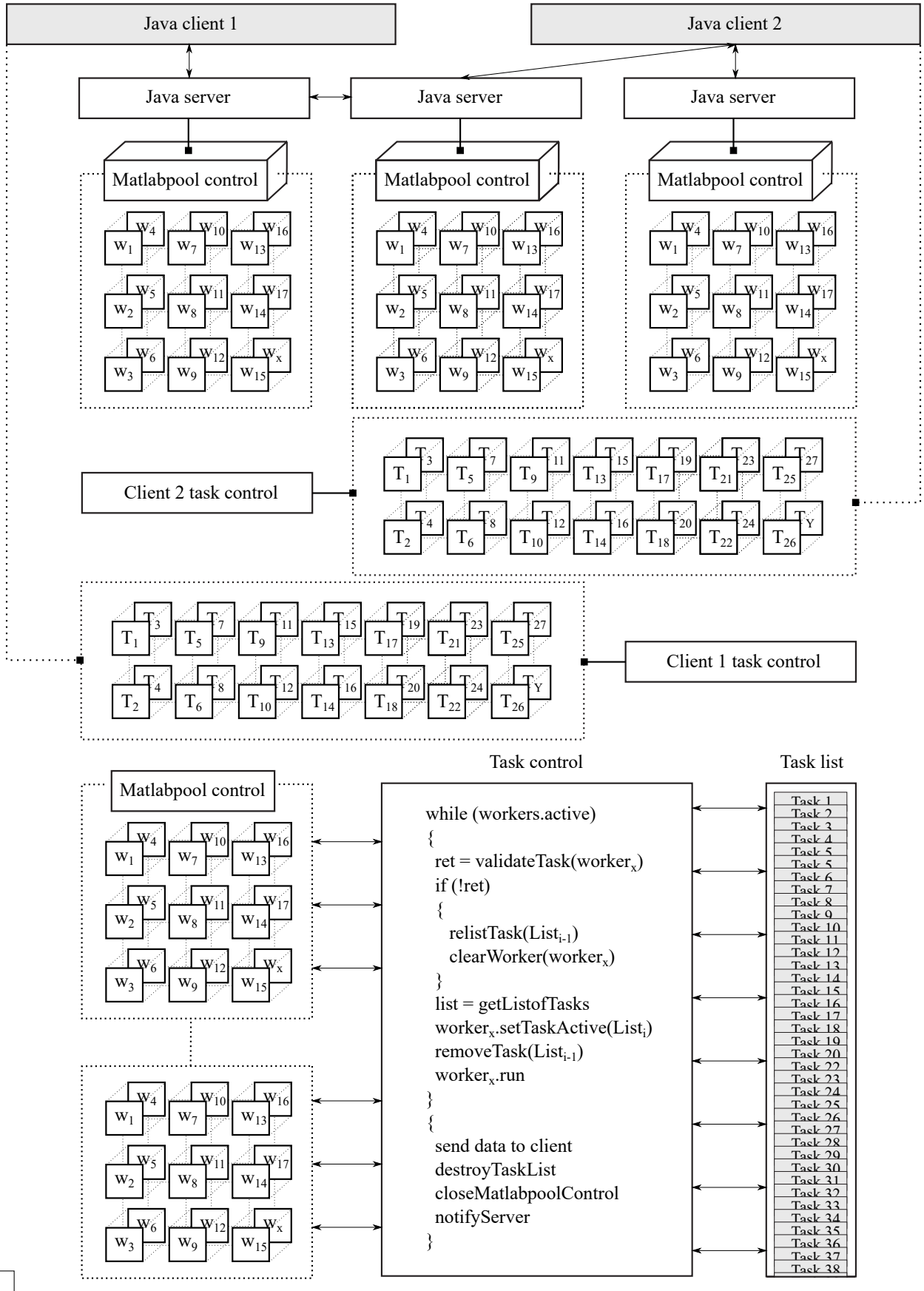
# X.12 Příloha L

**Příloha X.12:** Simulink model setting example.



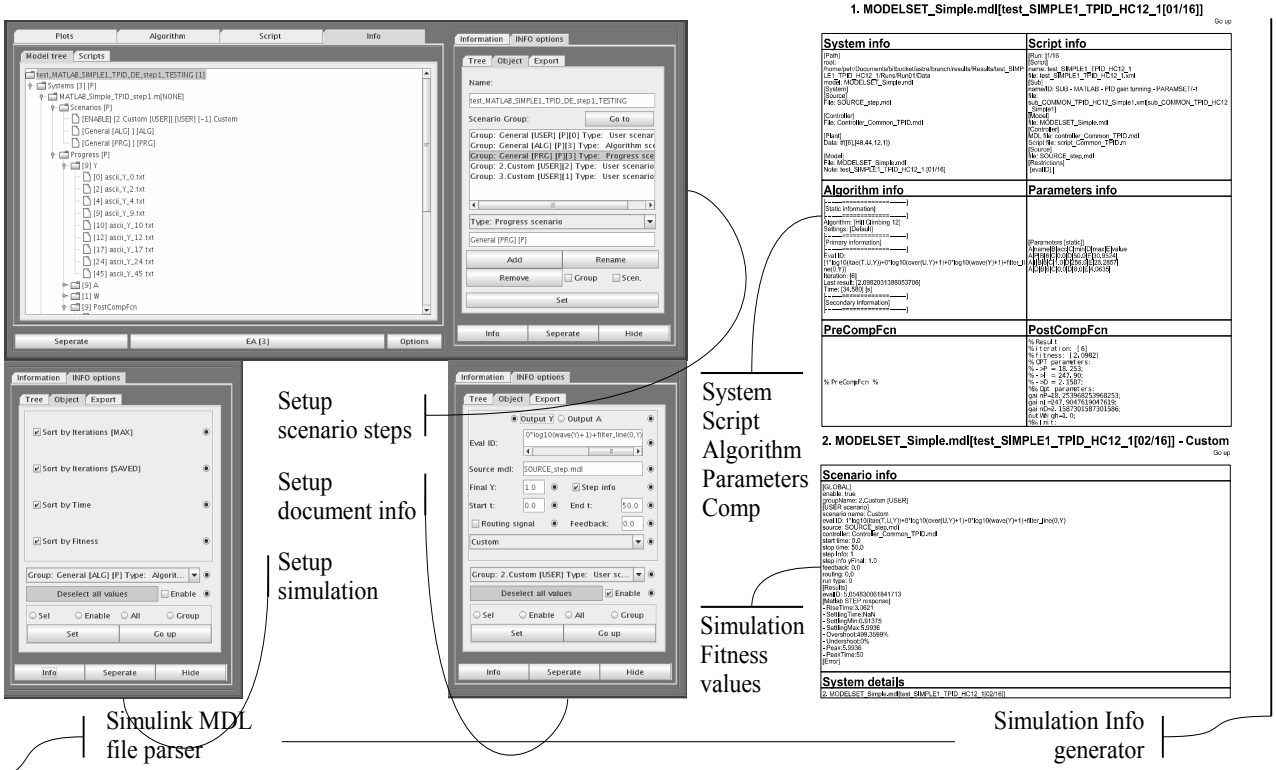
# X.13 Příloha M

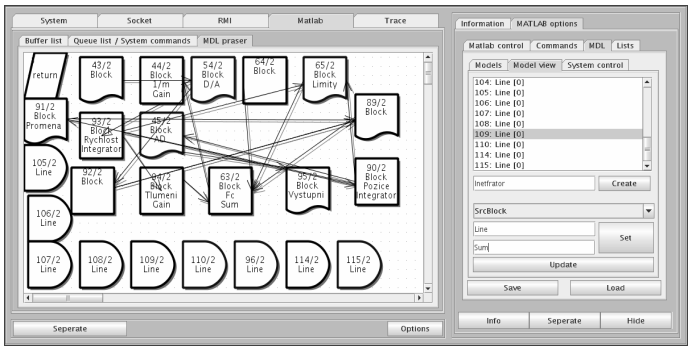
**Příloha X.13:** Client server worker control.



# X.14 Příloha N

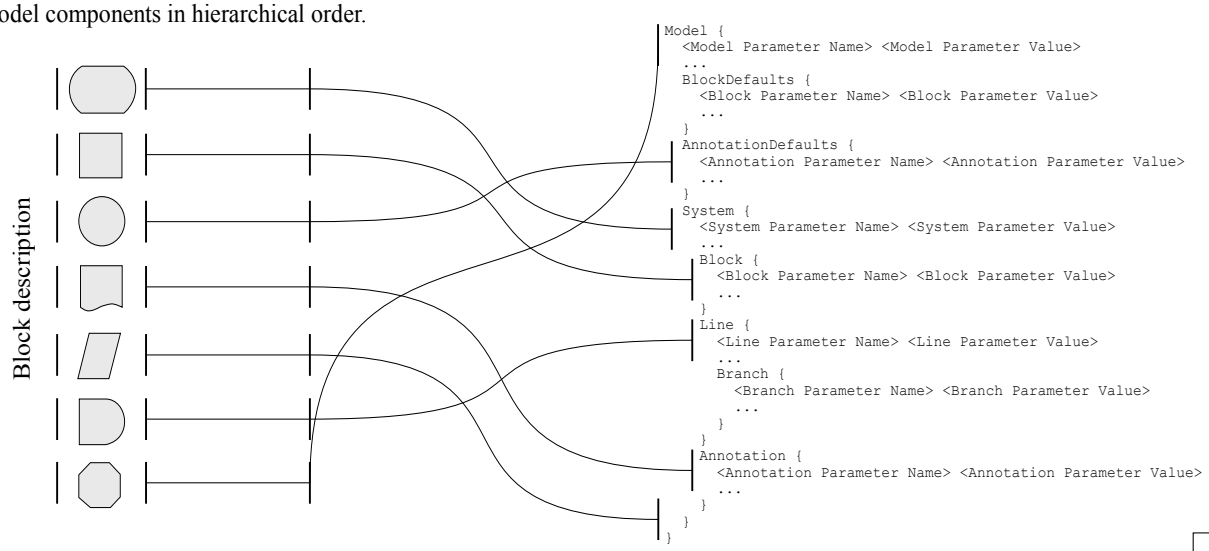
**Příloha X.14:** Simulation Info generator and Simulink MDL file parser.



The model file consists of sections that describe different model components:
# The Model section defines model parameters.
# The BlockDefaults section contains default settings for blocks in the model.
# The AnnotationDefaults section contains default settings for annotations in the model.
# The System section contains parameters that describe each system (including the top-level system and each subsystem) in the model. Each System section contains block, line, and annotation descriptions.

A model file is a structured ASCII file that contains keywords and parameter-value pairs that describe the model. The file describes model components in hierarchical order.



# X.15 Příloha O

**Příloha X.15:** Nelder-Mead metoda - standardní verze a Upravená metoda výběru akcni veličiny v závislosti na UPO.

**Algoritmus 1:** Nelder-Mead metoda - standardní verze.

```
1  begin
2      define Fitness funkce a parametrů;
3      do Vygenerování počátečních vrcholů B [Best], G [Good], W [Worst];
4      do Vyhodnocení fitness funkce dle aktuálních vrcholů;
5      sort Seřazení vrcholů vzestupně dle fitness B < G < W;
6      repeat
7          do Vypočet vrcholu M [Midpoint];
8          do Vypočet vrcholu R [Reflect];
9          do Výpočet fitness pro R;
10         if f(R) < f(W) then
11             do Vypočet vrcholu E [Expansion];
12             if f(E) < f(R) then
13                 do W ⟸ E;
14             else
15                 do W ⟸ R;
16             end
17         else if f(R) == f(W) then
18             do Vypočet vrcholů C1 a C2 [Contraction];
19             if f(C1) < f(W) || f(C2) < f(W) then
20                 do W ⟸ [C1 < C2 || C2 < C1];
21             else
22                 go Zmenšení;
23             end
24         else
25             label Zmenšení;
26             do Vypočet vrcholu S [Shrink];
27             do G ⟸ M;
28             do W ⟸ S;
29         end
30     until Dosažení ukončovací podmínky;
31     Finální řešení [B];
32 end
```

**Algoritmus 2:** Upravená metoda výběru akcni veličiny v závislosti na UPO.

```
1  begin
2      do Control_value = inf;
3      repeat
4          if Control sequence = y then
5              do temp = (y_{n-1} - UPO(i));
6          else
7              do temp = (x_{n-1} - UPO(i));
8          end
9          if abs(Control_value) > abs(temp) then
10             do Control_value = temp;
11         end
12     until i == počet UPO hodnot;
13 end
```

# X.16 Příloha P

**Příloha X.16:** Genetický, HC12 a Evolučních algoritm.

**Algoritmus 3:** Obecný princip genetického algoritmu.

```
1  begin
2      define GEN [maximální počet generací], F [koeficient mutace];
3      define NP [velikost výpočetní populace], CP [počet bodů křížení];
4      define SR [poměr selekce], Fitness funkce a chromozomy (parametry);
5      do Vygenerování náhodné počáteční populace;
6      repeat
7          do Dekódování chromozomů populace;
8          do Vyhodnocení fitness funkce dle aktuálních genů;
9          do Selekce zbylých jedinců pro křížení;
10         do Křížení (vytvoření nových potomků k doplnění populace);
11         do Mutace;
12     until Dosažení ukončovací podmínky;
13     Finální řešení;
14 end
```

**Algoritmus 4:** Obecný princip HC12 algoritmu.

```
1  begin
2      comment Soustava M matic pro realizaci algoritmu HC12.;
3      do M ← (M_0, M_1, M_2)^T;
4      comment Náhodně vygenerovaný či na základě jiné heuristiky zvolený binární vektor délky n, tzv.
       kandidát řešení.;
5      do a_opt;
6      repeat
7          comment Zde se uplatňuje Grayův kód. Nejdříve dochází k převodu binárního vektoru z GC na
           BC (zvlášť pro každou proměnnou x_i ), poté je BC převeden na celé číslo uint, případně na typ
           real x_i .;
8          do a_ker ← a_opt & A ← a_ker ⊕ M;
9          do a_opt ← arg minf (Γ (a)) (∀a ∈ A);
10     until a_opt = a_ker;
11 end
```

**Algoritmus 5:** Obecný princip evolučních algoritmů.

```
1  begin
2      define Parametry algoritmu a ukončovacích podmínek;
3      define Fitness funkce a parametrů;
4      do Vygenerování počáteční populace;
5      repeat
6          do Dekódování parametrů populace;
7          do Vyhodnocení fitness funkce dle aktuálních parametrů;
8          do Vygenerování nové populace (Selekce, Křížení, Mutace);
9      until Dosažení ukončovací podmínky;
10     Finální řešení;
11 end
```

# X.17 Příloha Q

**Příloha X.17:** Metoda ITAE s atraktorem více hodnot, Diferenciální evoluce a ETDASC metoda.

**Algoritmus 6:** Upravená metoda ITAE s atraktorem více hodnot.

```
1  begin
       input     : Požadovaný signál U
       input     : Kontrolovaný signál Y
2      define U levels [počet a hodnoty požadovaného signálu];
3      define Fitness in iteration(U levels)(2) [hodnaty fitness v iteraci];
4      define Index [nejlepší index hodnoty fitness v iteraci];
5      define Fitness [výsledna hodnota fitness];
6      init Fitness in iteration(1 ... U levels)(2) = 0;
7      init Fitness = 0;
8      init I = 1;
9      repeat
10         init J = 1;
11         repeat
12             do Fitness in iteration(J)(1) = ...
13             ... Fitness in iteration(J)(2) + ITAE(I,Y,U levels(J));
14         until J == počet levelů signálu U;
15         do Index = index(min(Fitness in iteration(:)(1)));
16         do Fitness in iteration(Index)(2) = inf;
17         if Fitness in iteration(:)(2) == inf then
18             init Fitness in iteration(1 ... U levels)(2) = 0;
19         end
20         do Fitness += Fitness in iteration(Index)(1);
21     until I == počet iterací;
22     result Fitness;
23 end
```

**Algoritmus 7:** Obecný princip diferenciální evoluce.

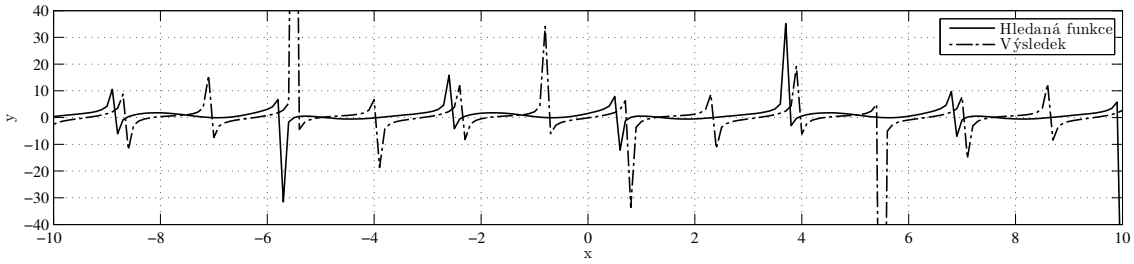
```
1  begin
2      define GEN [maximální počet generací], F [koeficient mutace];
3      define NP [velikost výpočetní populace], CP [počet bodů křížení];
4      define CR [koeficient selekce], Fitness funkce a chromozomy (parametry);
5      define L,H [Omezení parametrů];
6      do Vygenerování náhodné počáteční populace Pop_ij ← rand_ij[L, H];
7      for g = 1 to GEN do
8          for j = 1 to NP do
9              do Vyhodnocení fitness funkce dle aktuálních genů Fit_j ← f(Pop_j);
10             do Výběr náhodných jedinců r_1,2,3 ∈ [1, ..., NP], r_1 ≠ r_2 ≠ r_3 ≠ j;
11             do Vygenerování zkušebního vektoru X ← S(r, F, CR, Pop);
12             do Kontrola omezení parametrů if (x_i ∉ [L, H])x_i ← rand[L, H];
13             do Výběr lepšího řešení (X||Pop_j) a je překonáno → iBest;
14         end
15     end
16     Finální řešení [iBest];
17 end
```

**Algoritmus 8:** ETDASC metoda.

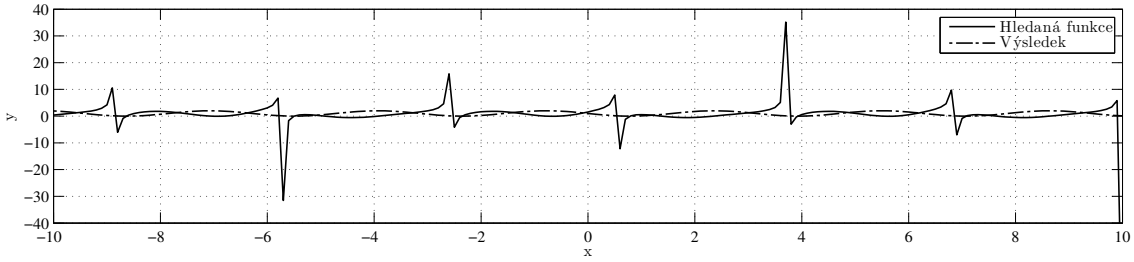
```
1  begin
2      do s_n−1 = Control_value + param_R * s_n−1−UPO;
3      do Fn = param_K * ((1 − param_R) * s_n−1−UPO − Control_value);
4      if Fn > (param_Fmax/F_tuning) then
5          do Fn = (param_Fmax/F_tuning);
6      else
7          do Fn = −(param_Fmax/F_tuning);
8      end
9  end
```

# X.18 Příloha R

**Příloha X.18:** Výsledek hledáné jednoduché funkce pomocí Gramatické Evoluce.



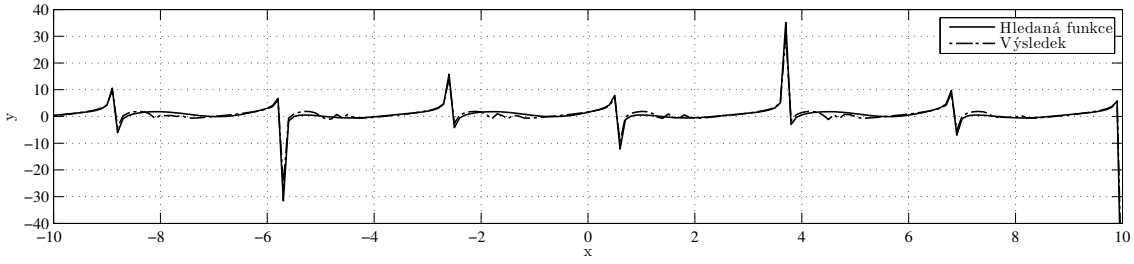

**Graf X.1:** Výsledek testování Gramatické Evoluce krok [1].



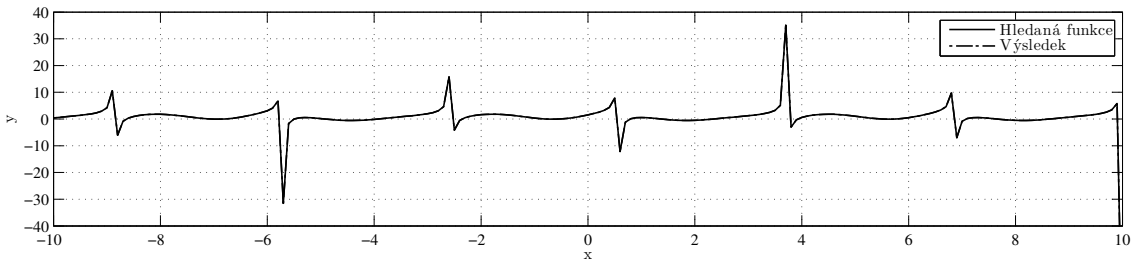

**Graf X.2:** Výsledek testování Gramatické Evoluce krok [2].

| Expression | Difference | Nastavení | Krok |
|---|---|---|---|
| ::(tan(x+x)-sin(x+1)*sin(1+1+(x+1))) | 5.847 | ::Gramatická evoluce NP: 1000, F: 0.1, R: 0.5, | [1] |
| ::(1+1-((1/1)-(1-1)+sin(x+x)))+(1-1) | 4.534 | ::W: 4, Fitness: log10(iae(U,Y)+1)+log10(W), | [2] |
| ::(cos((tan((x*1))+(1*x)))/tan(cos((1*1)+x/1))) | 1.794 | ::CR: 2, C: 20, AC: 8bit, G: (gramatika: 2.1), | [3] |
| ::cos((x+x+(x-x)))*(sin(x)+tan((1+x))) | 0 | ::GEN: 1000, F: (tan(x+1)+sin(x))*cos(2*x); | [4] |

**Tabulka X.1:** Hodnoty příkladu hledání jednoduché funkce pomocí Gramatické Evoluce.




**Graf X.3:** Výsledek testování Gramatické Evoluce krok [3].

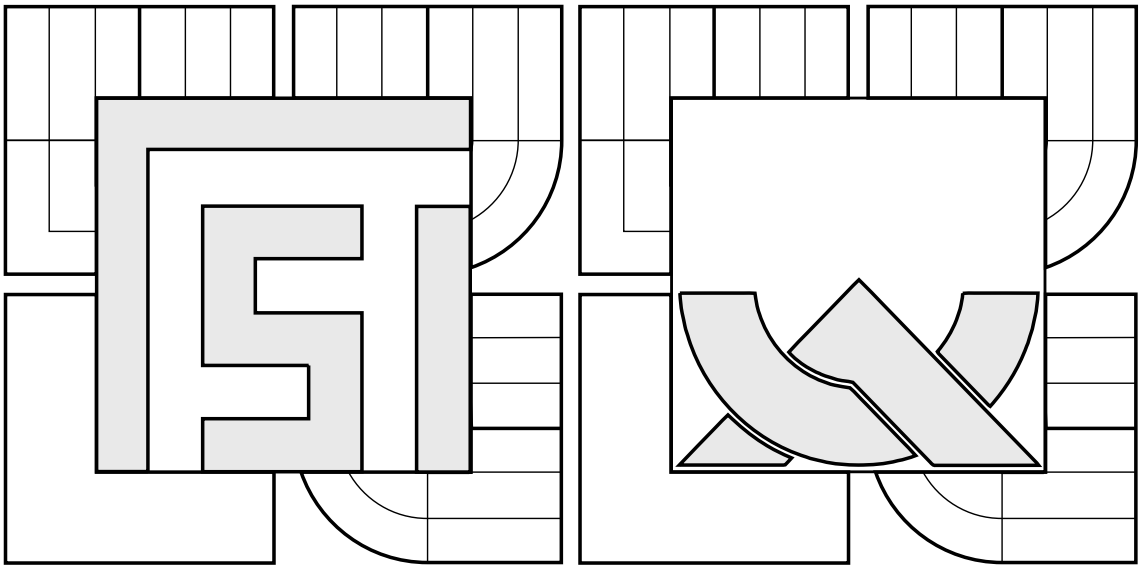



**Graf X.4:** Výsledek testování Gramatické Evoluce krok [4].

| ⊞ | Document Metadata |
|---|---|
| Author | Petr Minář |
| Title | Nelineární řízení komplexních soustav s využitím evolučních přístupů |
| Subject | Dizertační práce |
| Creator | TeXnicCenter 2.02 Stable (64 bit) |
| Producer | Vysoké učení technické Brno / Fakulta strojního inženýrství |
| Keywords | soft computing, umělá inteligence, optimalizace, automatizace, regulace, nelineární systémy, fuzzy logika, chaosové systémy, logistická mapa, duffingova rovnice, uda-yagi, java, matlab, simulink, paralelní výpočty, PID, HC12, DE, NM, GE, GA, TDAS, ETDAS, ITAE |
| Language | Czech |
| PDF Version | 1.6 |
| Page Format | A4, Text height: 674pt, Text width: 426pt |
| Page Count | 187 |
| Statistics | cite-author: 12, acronym: 65, section: 120, fig: 37, math: 100, tab: 118, struct: 28, grammatic: 6, plot: 290, group: 101, model: 9, cite: 85, appendix: 18, chapter: 11 |
| MD5 | FC0C4A1D93DC0AFE0E1A71ABAE160117 |
| File Size | 13305834[B] |
| File Name | VUT-MinarPetr-dezertacniPrace.pdf |
| Created at | 2017/04/05 at 12:42:56 |

QR code metadata: (a)Document detailed information (b)PDF detailed information

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ



NELINEÁRNÍ ŘÍZENÍ KOMPLEXNÍCH SOUSTAV S VYUŽITÍM EVOLUČNÍCH PŘÍSTUPŮ