VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ

ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY

FACULTY OF MECHANICAL ENGINEERING

INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

EVOLUČNÍ OPTIMALIZACE ŘÍDICÍCH ALGORITMŮ

EVOLUTIONARY OPTIMIZATION OF CONTROL ALGORITHMS

DIZERTAČNÍ PRÁCE

DOCTORAL THESIS

AUTOR PRÁCE Ing. ROMAN WEISSER

AUTHOR

VEDOUCÍ PRÁCE prof. Ing. PAVEL OŠMERA, CSc.

SUPERVISOR

BRNO 2010

Bibliografická citace:

WEISSER, R. *Evoluční optimalizace řídicích algoritmů.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2010. 157 s. Vedoucí dizertační práce prof. Ing. Pavel Ošmera, CSc.

Poděkování:

Děkuji prof. Ing. Pavlu Ošmerovi, CSc. za odborné vedení, poskytnutí studijních materiálů a konzultací v průběhu realizace této disertační práce; své rodině za všestrannou pomoc a podporu nejen v době mého studia, ale v průběhu celého mého života; přítelkyni za trpělivost a podporu při psaní této práce; Ondrovi za odborné rady týkající se sázení textu v typografickém systému LaTeX a v neposlední řadě Tomovi za praktické rady z oblasti C#.NET.

Roman.weisser@gmail.com

Prohlašuji, že jsem tuto disertační práci vypracoval samostatně a v seznamu literatury uvádím veškeré informační zdroje, které jsem použil.

V Brně, dne 18. června 2010 .....

Abstrakt

Dizertační práce se zaměřuje na evoluční optimalizaci řídicích algoritmů. V první části práce jsou popsány evoluční optimalizační metody, jejich principy a dílčí algoritmy, zejména pak ty, které se používají ve dvoufázové transplantační evoluci. Dále práce popisuje gramatickou evoluci, jejíž modifikovaný algoritmus se stal podnětem pro vytvoření transplantační evoluce. Transplantační evoluce a její dvoufázová modifikace jsou nové evoluční algoritmy navržené během řešení této práce, jenž byly použity pro optimalizaci struktury a parametrů řídicích algoritmů obecných regulátorů. Algoritmus transplantační evoluce a jeho rozšířená dvoufázová modifikace jsou podrobně popsány v následujících kapitolách.

Pro zajištění minimalizace času optimalizace a nalezení řešení blížícímu se globálnímu optimu je důležité správné nastavení evolučních algoritmů. Vhodnému nastavení parametrů diferenciální evoluce se stručně věnuje kapitola meta-evoluce.

V další části jsou popsány základní pojmy z oblasti regulace, vybrané metody identifikace soustav a nastavení regulátorů. Tato část také popisuje algoritmy číslicových regulátorů a některé metody používané během jejich realizace.

V experimentální části jsou ukázky optimalizace řídicích algoritmů různých typů regulátorů. Optimalizované řídicí algoritmy obecných regulátorů jsou porovnány na různých soustavách s různě nastavenými PSD regulátory, jejichž nastavení bylo realizováno vybranými algebraickými metodami, nebo diferenciální evolucí.

V závěru práce je uvedeno doporučení pro další vývoj v oblasti evoluční optimalizace regulátorů se zaměřením na oblast paralelních a distribuovaných výpočtů.

Klíčová slova

umělá inteligence, genetické algoritmy, gramatika, gramatická evoluce, diferenciální evoluce, transplantační evoluce, dvoufázová transplantační evoluce, optimalizace, regulátory

Abstract

The dissertation thesis deals with Evolution optimization of control algorithms. The first part of the thesis describes the principles and partial methods of evolution optimization methods especially those used in two-level transplant evolution method. Later the grammatical evolution method is described, which modified algorithm became impulse for creation of transplant evolution method. The transplant evolution method and its two-level modification are new evolutionary algorithms proposed in this work, which were used for optimization of structure and parameters of general controllers control algorithms. The transplant evolution algorithm and its extended two-level modification are described in detail in next chapters.

The proper settings of evolutionary algorithms are important for minimization the time of optimization and for finds results approaching the global optimum. For proper setting the parameters of differential evolution was created meta-evolution algorithm that is described in chapter named meta-evolution.

The basic concepts of control, chosen methods of system identification and controller parameters settings are described in next part. This part describes algorithms of digital controllers and some specific methods uses in digital control.

The demonstrations of control algorithm optimizations of various types of controllers are showed in experimental part. The optimized algorithms of general controllers are compared with various types of PSD controllers which were set by various algebraic methods or differential evolution for various models of systems.

In the conclusion of this work is stated a recommendation for further development of evolutionary optimization of controllers are focusing on parallel and distributed computing.

Keywords

artificial intelligence, genetic algorithms, grammatical evolution, differential evolution, transplant evolution, two–level transplant evolution, optimization, controllers

Obsah

1 Úvod **7**
1.1 Cíle práce . . . 8
1.2 Organizace práce . . . 9

2 Evoluční algoritmy **11**
2.1 Evoluční strategie . . . 11
2.2 Stochastické metody . . . 12

3 Genetické algoritmy **13**
3.1 Biologické pozadí . . . 13
3.2 Princip GA . . . 15
3.3 Teorie schémat . . . 16
3.4 Prohledávaný prostor . . . 17
3.5 Inicializace . . . 18
3.6 Kódování chromozómů . . . 19
3.6.1 Binární kódování . . . 19
3.6.2 Celočíselné kódování . . . 20
3.6.3 Kódování pomocí reálných čísel . . . 20
3.6.4 Kódování pomocí stromu . . . 21
3.7 Reprodukce . . . 21
3.8 Křížení . . . 22
3.8.1 Uniformní křížení . . . 22
3.8.2 Bodové křížení . . . 22
3.8.3 Aritmetické křížení . . . 23
3.9 Mutace . . . 23
3.10 Selekce . . . 24
3.10.1 Roulette-wheel selection . . . 24
3.10.2 Linear ranking . . . 25
3.10.3 Exponential ranking . . . 26
3.10.4 Tournament selection . . . 27
3.10.5 Sigma scaling . . . 27
3.10.6 Boltzmann selection . . . 28
3.10.7 Zkrácený výběr (Truncation selection) . . . 30
3.10.8 Disruptive selection . . . 30
3.10.9 Elitismus . . . 30
3.11 Zabránění incestu (Incest prevention) . . . 31
3.12 Selekční tlak . . . 31

4 Diferenciální evoluce **33**
4.1 Řídící parametry . . . 33
4.2 Inicializace populace . . . 34
4.3 Reprodukční cyklus . . . 34
4.3.1 Selekční schémata . . . 34
4.4 Auto–adaptivní diferenciální evoluce - SaDE . . . 36

5 Jazyky a gramatika **39**
5.1 Chomského hierarchie gramatik a jazyků . . . 40
5.2 BNF gramatiky . . . 41

6 Gramatická evoluce **43**
6.1 Překlad genotypu na fenotyp . . . 44
6.2 Operátory . . . 46
6.2.1 Křížení . . . 46
6.2.2 Mutace . . . 47
6.2.3 Operátor wrap . . . 48
6.3 Zpětné zpracování pravidel a značkování kodonů . . . 48
6.4 Modifikované křížení . . . 51
6.5 Modifikovaná mutace . . . 52
6.5.1 Strukturální mutace . . . 52
6.5.2 Nestrukturální mutace . . . 53
6.5.3 Chybně provedená mutace . . . 53
6.6 Párování značek . . . 54
6.7 Generování čísel pomocí gramatiky . . . 55

7 Transplantační evoluce **57**
7.1 Inicializace jedince . . . 57
7.2 Křížení - Transplantace . . . 60
7.2.1 Křížení částí stromů . . . 60
7.2.2 Spojování stromů . . . 62
7.3 Mutace . . . 62
7.3.1 Nestrukturální mutace . . . 63
7.3.2 Strukturální mutace . . . 64
7.3.3 Pravděpodobnost výběru typu mutace . . . 67
7.4 Pravděpodnost výběru pravidla . . . 67

8 Dvoufázová transplantační evoluce **69**
8.1 Architektura . . . 71
8.2 Modul transplantační evoluce . . . 72
8.3 Modul diferenciální evoluce . . . 72
8.4 Modul genetického algoritmu . . . 72
8.5 Modul výpočtu fitness . . . 72
8.6 Mezivrstva . . . 73
8.7 Výstupní modul . . . 73
8.8 Aritmetická redukce stromu . . . 74

9 Meta–evoluce **77**

10 Regulační obvody **79**
10.1 Historie průmyslové regulace . . . 79
10.2 Základní zapojení regulačního obvodu . . . 79
10.3 Rozvětvené jednorozměrové regulační obvody . . . 80
10.4 Identifikace soustav . . . 81

10.4.1 Nekmitavé proporcionální regulované soustavy 81
10.4.2 Kmitavé proporcionální regulované soustavy 83
10.4.3 Integrační regulované soustavy 84
10.5 Regulátory 85
10.5.1 Diskrétní regulátory 86
10.6 Řídicí algoritmy regulátorů 87
10.6.1 Regulátory s pevně danou strukturou 87
10.6.2 Modifikace řídicího algoritmu 90
10.6.3 Obecné regulátory 91
10.7 Význam jednotlivých složek regulátorů 91
10.8 Antiwind-up 91
10.9 Seřízení regulátorů 92
10.9.1 Seřízení regulátorů Ziegler–Nicholsovou metodou 92
10.9.2 Seřízení regulátorů podle přechodové charakteristiky regulované soustavy 94
10.9.3 Seřízení regulátorů metodou inverze dynamiky 95
10.9.4 Seřízení regulátoru podle funkcí standardního tvaru 96
10.9.5 Metoda pokusu a omylu 96
10.10 Kvalita regulace 97
10.10.1 Relativní překmit 97
10.10.2 Doba regulace 97
10.10.3 Velikost regulační plochy 98

11 Praktická část 99
11.1 Úplná gramatika a slovník ART 99
11.2 Kriteriální funkce 99
11.2.1 Hodnocení simulace v případě chyby 102
11.3 Modifikované schéma DE 104
11.4 Podmínky zastavení 104

12 Experimentální část 107
12.1 Porovnání regulačních pochodů PSD regulátorů pro různé metody nastavení parametrů 107
12.2 Porovnání regulačních pochodů Takahashiho regulátorů pro různé metody nastavení parametrů 110
12.3 Porovnání PSD regulátorů a obecného regulátoru (diferenciální rovnice stejného řádu) – proporcionální soustava 112
12.4 Porovnání PSD regulátoru a obecného regulátoru – kmitavá proporcionální soustava 113
12.5 Porovnání regulátorů – integrační soustava 116
12.6 Optimalizace obecných regulátorů 120
12.6.1 Ukázky chybně nastavené optimalizace 121
12.6.2 Ukázky chyb v průběhu simulace 124
12.6.3 Porovnání kriteriálních funkcí 125
12.6.4 Porovnání řídicích rovnic regulátorů vytvořených různými sadami pravidel a různými kriteriálními funkcemi 127

12.7 Porovnání obecných regulátorů se setrvačností prvního řádu a PSD regulátorů – kmitavá proporcionální soustava bez dopravního zpoždění 129

13 Paralelní a distribuované výpočty **133**

14 Závěr **135**

Přílohy **139**
14.1 Reprezentace čísla dle normy IEEE 754 139
14.2 Kódování *real* to *dec* to *bin* to *Gray code* 140

Literatura **141**

Seznam zkratek **146**

Seznam symbolů **147**

Typy použitých notací **148**

Seznam obrázků **149**

Seznam tabulek **151**

Seznam algoritmů **152**

1 Úvod

V současnosti se teorie řízení dynamických systémů opírá o dobře propracovaný matematický aparát, který efektivně využívá. Při návrhu řídících struktur se nestudují přímo dynamické systémy, které jsou předmětem řízení, ale jejich vhodně navrhnuté modely. Tvorba modelů vychází ze dvou základních principů. První princip se opírá o sledování chování vstupních a výstupních veličin, bez nutnosti detailně poznat řízený systém, tzv. *(black box)* přístup. Na základě tohoto chování jsme schopni sestavit exaktní model, který vystihuje chování skutečného systému. Druhou možností je proniknout do podstaty systému a na základě znalostí jeho vnitřních zákonitostí vytvořit matematický model. Na popis modelu je možno použít soustavy lineárních, nelineárních diferenciálních, případně diferenčních rovnic, nebo přenosu.

Nejběžnějším nástrojem používaným v aplikacích průmyslového řízení jsou PID regulátory, které jsou používány k regulaci rychlosti, teploty, tlaku a mnoha dalších procesních proměnných. V současnosti se PID regulace využívá ve více než 95 % případů procesního řízení. V posledních letech se používají především mikroprocesorem řízené číslicové regulátory.

Za počátek éry PID regulátorů lze považovat období 1915–1940, kdy vznikaly proslulé regulační firmy Bristol, Fisher, Foxboro, Honeywell, Leeds & Nortrup a Taylor Instrument. Proporcionálně integrační regulátory však byly v průmyslu užívány mnohem dříve. Proporcionální zpětná vazba tvoří základ dobře známého odstředivého regulátoru vynalezeného kolem roku 1750, který byl využit pro řízení otáček větrného mlýnu. Podobný regulátor řídil také otáčky Wattova parního stroje[1]. Na odstředivý regulátor se tehdy pohlíželo jako na jediné zařízení bez rozlišení čidla regulované veličiny, ústředního a akčního členu regulátoru. Následné porozumění významu jednotlivých částí tohoto zařízení bylo klíčovým bodem pro jeho další postupné vylepšování. Konkrétně oddělení čidla od akčního členu umožnilo vývoj hydraulického mechanismu realizujícího proporcionálně–integrační regulátor. První regulátor s derivační složkou byl sestrojen v roce 1935[2].

Prudký rozvoj mikroelektroniky v nedávné době umožnil přechod na číslicové regulátory. To způsobilo další posun ve vývoji regulační techniky a základní řídicí algoritmus průmyslového regulátoru byl doplněn dalšími expertními funkcemi jako je self–tuning, adaptace parametrů regulátoru na změnu okolního prostředí, beznárazové přepínání režimů, apod. V poslední době se začínají uplatňovat také fuzzy regulátory a neuronové sítě.

Při vývoji regulátorů je snaha o zabezpečení vysoké robustnosti a zajištění stability systému, současně s požadavkem na vysokou kvalitu řízení. V současné etapě vývoje neexistuje žádný řídicí algoritmus, který by dokázal splnit všechny formulované požadavky.

Je ovšem třeba konstatovat, že exaktní porovnání i u stejného typu řídicího algoritmu je problematické. Někdy je požadován co nejrychlejší přechodový děj a překmit regulované veličiny nevadí, jindy musí být přechodový děj s minimálním překmitem či zcela bez překmitu. To samozřejmě omezuje použití integrálních kritérií pro vyhodnocení kvality regulačního pochodu. Zejména v reálném provozu, při působení obecných náhodných poruch, může být vyhodnocení nejlepšího nastavení regulátoru velmi obtížné, protože tyto poruchy nejsou zpravidla měřitelné. Navíc reálný systém je velmi často silně nelineární

[1] Wattův parní stroj: 1788

[2] Taylor Instrument: první pneumatický regulátor s derivační složkou (1935)

a parametry systému se značně mění s časem nebo např. podle okamžitého výkonu, na kterém systém právě pracuje. Optimální nastavení parametrů řídicího algoritmu je pak zpravidla kompromisem mezi více možnostmi nastavení. Tuto optimalizaci zatím v naprosté většině případů provádí specialista, který má pro tuto specializaci předpoklady. Kvalitní seřízení regulačního obvodu je velmi náročná činnost, kterou je nutné pravidelně opakovat, protože se stárnutím zařízení dochází ke změnám v dynamice procesu.

I když stovky výzkumných ústavů a univerzit na celém světě pracují na vývoji nových adaptivních a inteligentních regulátorů, které by měly pomoci realizovat automatické nastavení parametrů řídicích algoritmů, bylo v průmyslové oblasti dosaženo zatím jen dílčích úspěchů.

Dobře naprogramovaný a nastavený PID regulátor stále zůstává v praxi daleko nejčastěji používaným regulátorem při relativně jednoduchém nastavování. Každá význačnější regulační firma má vypracovány své vlastní varianty řídicích algoritmů a metodiku jejich nastavování. Tyto postupy se vzájemně liší a někdy bývá obtížné najít společnou podstatu, protože algoritmus PID regulátoru je možné realizovat mnoha způsoby. Nejen v české, ale i v cizojazyčné literatuře bývá spojitý PID regulátor často odbyt „knižní“ verzí PID regulátoru a z diskrétních tvarů algoritmů bývá uvedena přírůstková verze PSD regulátoru nebo její ekvivalent v Z–transformaci. Spojitý PID regulátor a jeho diskrétní tvar ovšem mají řadu dalších variant, které mají mnohem vhodnější vlastnosti pro implementaci v průmyslových procesech, než jaké nabízí základní definice PID regulátoru.

1.1 Cíle práce

Cílem práce je zaměřit se na oblast evolučních metod návrhu řídicích algoritmů.

1. Vybrat a popsat vhodné nástroje pro optimalizaci řídicích algoritmů
2. Optimalizovat řídicí algoritmy obecných regulátorů
3. Vybrat regulovanou soustavu vhodnou pro testování navržených algoritmů
4. Simulačně porovnat vlastnosti evolučně navržených řídicích algoritmů s řídicími algoritmy PSD regulátorů
5. Navrhnout kriteriální funkci evolučního algoritmu, která dle zadaných kritérií umožní dosáhnout vysokou kvalitu regulačního pochodu

Klíčovou částí v oblasti optimalizace řídicích rovnic regulátorů pomocí evolučních algoritmů je zejména vytvoření evolučního algoritmu, jenž bude schopen generovat a optimalizovat komplexní struktury řídicích rovnic regulátorů a současně i její parametry.

1.2 Organizace práce

V kapitole *Genetické algoritmy* jsou popsány základní principy a pojmy nutné pro popis genetických algoritmů, diferenciální, gramatické a transplantační evoluce.

V kapitole 4 je popsána metoda diferenciální evoluce, která je součástí dvoufázové transplantační evoluce. V této části jsou popsány základní principy a dílčí algoritmy, které jsou použité v dalších kapitolách. O diferenciální evoluci ještě jednou pojednává kapitola praktická část, kde je představeno nové selekční schéma diferenciální evoluce.

Kapitola 5 ve stručnosti hovoří o *jazycích a gramatikách*, jejichž principy využívá gramatická a transplantační evoluce.

Kapitola *Gramatická evoluce* popisuje základní a modifikovaný algoritmus gramatické evoluce, který se stal podnětem pro vytvoření transplantační evoluce.

Kapitola 7 podrobně pojednává o nově vytvořeném evolučním algoritmu *Transplantační evoluce*, jejích výhodách a nevýhodách a navazuje na principy gramatické evoluce jenž jsou popsány v předchozí kapitole.

Kapitola nazvaná *Dvoufázová transplantační evoluce* popisuje další zcela nový evoluční algoritmus, který pracuje na principu dvoufázové optimalizace založené na kombinaci transplantační evoluce a diferenciální evoluce, genetického algoritmu, nebo jakéhokoliv jiného modulu pro optimalizaci parametrů z oboru reálných čísel. V této kapitole je také poměrně podrobně popsána architektura navrženého řešení.

Kapitola 9 pojednává o *Meta–evoluci*, která byla použita pro optimalizaci nastavení diferenciální evoluce, jenž byla využívána pro optimalizaci abstraktních numerických parametrů struktury řídicích rovnic regulátorů.

V kapitole *Regulační obvody* je stručný teoretický úvod do regulačních obvodů, jenž pojednává základních principech a pojmech z oblasti regulace.

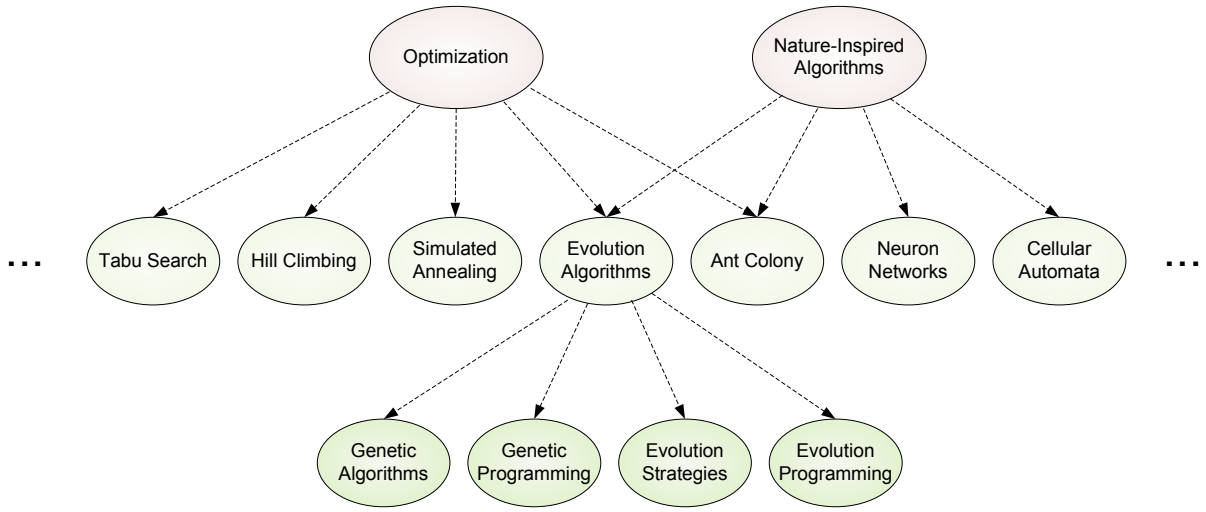
V kapitole *Praktická část* je uvedena kompletní gramatika a slovník pro algebraickou redukci struktury řešení, dále jsou popsána kritéria hodnocení kvality regulace použitá pro optimalizaci řídicích rovnic regulátorů. V této kapitole je také představeno modifikované schéma diferenciální evoluce a také podmínky zastavení, které urychlují dobu optimalizace.

V kapitole nazvané *Experimentální část* jsou popsány některé testovací optimalizační úlohy na kterých byly testovány nové evoluční algoritmy. Výsledky optimalizací jsou porovnávány na různých typech regulovaných soustav současně s různými typy PSD regulátorů, nastavenými klasickými výpočetními, nebo evolučními metodami. Nejzajímavější z této části je podkapitola pojednávající o optimalizaci obecných regulátorů

Kapitola 13 popisuje *paralelní a distribuované výpočty* a obsahuje doporučení pro další vývoj v oblasti optimalizace řídicích rovnic obecných regulátorů.

2 Evoluční algoritmy

Evoluční algoritmy (EA) jsou zastřešujícím termínem pro různé přístupy využívající modely evolučních procesů pro účely nemající téměř nic společného s biologií. Pro účely optimalizace se snaží využívat představy o hnacích silách evoluce živé hmoty. Evoluční algoritmy se především používají pro řešení velkých komplexních optimalizačních problémů s mnoha lokálními extrémy, protože je zde menší pravděpodobnost, že uváznou v lokálním extrému než je tomu u tradičních gradientních metod. Důležitým rysem evolučních algoritmů je jejich generační prohledávací strategie. EA se řadí mezi optimalizační algoritmy inspirované přírodou, viz. obrázek 1.



Obrázek 1: Rozdělení a příslušnost evolučních algoritmů

Z matematického hlediska se evoluční algoritmy řadí mezi stochastické metody prohledávání stavového prostoru a dají se rozdělit na čtyři základní směry:

- Evoluční strategie (viz. kapitola 2.1)
- Evoluční programování - technika, při níž se aplikují náhodné mutace na konečné automaty reprezentující řešení problému. Pochází z roku 1966, kdy ji ve svém díle *Artificial Intelligence through Simulated Evolution* prezentovali autoři L.J. Fogel, A. J.Owens a M. J.Walsch.
- Genetické algoritmy (viz. kapitola 3)
- Genetické programování (viz. kapitola 3)

2.1 Evoluční strategie

Evoluční strategie (ES) patří mezi první úspěšné stochastické algoritmy. Myšlenka evolučních strategií byla poprvé publikována v roce 1973, I. Rechenbergem v práci *Evolutionsstrategie: Optimisierung technischer Systeme nach Prinzipien der biologischen Evolution*. ES jsou primárně určeny pro parametrickou optimalizaci a dosahují velmi dobrých výsledků. Parametrická optimalizace je optimalizace vektoru n nezávislých proměnných

(parametrů) $x \in \mathbb{R}^n$, které ovlivňují chování nějakého systému. Pomocí evoluční strategie se snažíme nalézt co nejvýhodnější kombinaci hodnot jednotlivých parametrů vektoru x_{opt} tak, aby účelová funkce (fitness funkce) $f(x)$ dávala nejvyšší, resp. nejnižší hodnotu. Formálně:

$$x_{opt} = arg\, min_{x \in <a,b>^n} f(x) \tag{2.1}$$

Kde hodnoty jednotlivých parametrů náleží intervalu $x_i \in < a, b >^n \in \mathbb{R}^n$.

ES pracují s vektorem reálných hodnot, což má své opodstatnění v případě generování potomka z rodiče $x \rightarrow x'$, které se provádí přičtením náhodně vygenerované hodnoty s Gausovou distribucí.

2.2 Stochastické metody

Stochastické algoritmy jsou iterativní výpočetní metody, které v průběhu činnosti používají náhodné operace. V průběhu hledání řešení připouštějí dočasné zhoršení řešení za předpokladu budoucího zlepšení. Každý krok stochastického algoritmu je zatížen určitou neurčitostí a může vést jak ke zlepšení, tak také ke zhoršení řešení. Tato vlastnost umožňuje stochastickým metodám překonání lokálního extrému. Do stochastických prohledávacích metod se řadí například:

- Metoda simplexů *(Nelder–Mead)*
- Monte Carlo *(Brute–Force Search)*
- Horolezecký algoritmus *(Hill climbing)*
- Simulované žíhání *(Simulated Annealing)*
- Zakázané prohledávání *(Tabu search)*
- Náhodné prohledávání *(Random Walk)*

3 Genetické algoritmy

Genetické algoritmy (GA)[27] jsou poměrně mladou disciplínou inspirovanou Darwinovou teorii evoluce a v současnosti jsou předmětem intenzivního studia. Poprvé byly představeny v publikaci *Adaptation in Natural and Artificial Systems* [18], kterou v roce 1975 napsal John H. Holland, jenž se v té době věnoval studiu buněčných automatů na Michiganské Univerzitě. Holland se kromě popisu algoritmu věnoval také teoretickému zdůvodnění jeho funkčnosti (viz. kapitola 3.3).

Výhodou GA je, že dobře fungují při řešení problémů, kde téměř všechny ostatní algoritmy selhávají, např. pro NP-úplné problémy, tj. kde výpočetní čas je exponenciálně nebo faktoriálně závislý na počtu proměnných. Pro jejich časovou náročnost je však nemá smysl používat u relativně jednoduchých optimalizovaných funkcí nebo u funkcí, pro které existují specializované algoritmy.

V roce 1992 představil John Koza[3] [21] originální modifikaci genetického algoritmu, kterou nazval **genetické programování (GP)**. Při tomto přístupu je původní reprezentace chromozómů ve formě znakových řetězců nahrazena složitějšími funkcemi ve formě stromových struktur, což upřednostňuje jazyky, které tyto struktury přirozeně zahrnují[4]. V nejjednodušší verzi GP se funkce rovnají výrazům obsahujícím proměnné, konstanty, základní aritmetické operace a elementární funkce, případně to mohou být složitější stavební bloky, jako jsou tranzistory, zesilovače, prvky kybernetického systému, apod.

Své uplatnění mají GA v oblastech plánování výroby, logistice, ekonomických optimalizačních problémech, konstrukčních úloh ve strojírenství, řízení a robotice. Dále se uplatňují v návrhu elektronických obvodů, automatickém generování programů, apod.

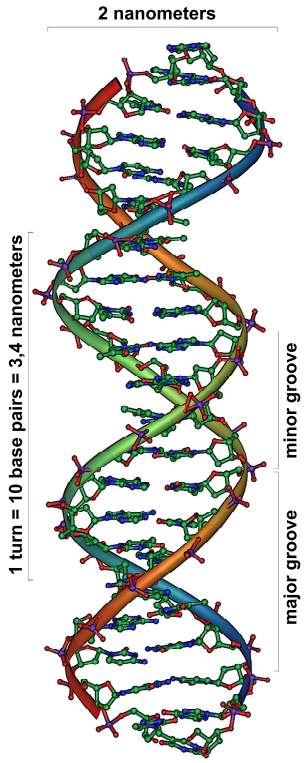
3.1 Biologické pozadí

GA jsou založeny na biologickém pozadí a snaží se napodobit děje v živé přírodě [19, 52, 53, 54, 55, 56, 57, 58]. Všechny živé organismy se skládají z buněk. Každá buňka obsahuje DNA. **DNA**, viz. obrázek 2a, je komplexní polymer tvaru dvoušroubovice, který vytváří dlouhé řetězce známé jako **chromozómy**, viz. obrázek 2b.

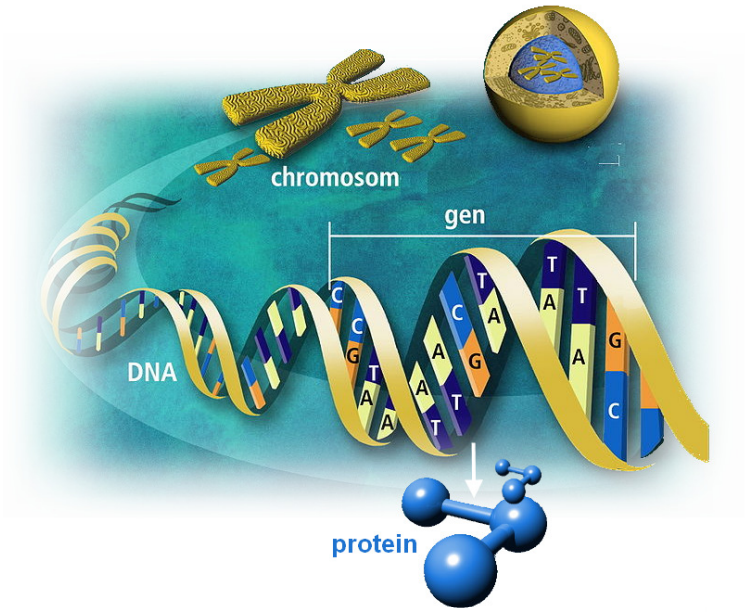
Lidé mají celkem 46 chromozómů, které jsou uspořádány ve 23.párech v každém buněčném jádře. Tedy s dvěma výjimkami – lidské červené krvinky nemají jádro vůbec (až na některé případy z Blízkého východu) a na **gamety** (pohlavní buňky) – jak spermie tak vajíčko jsou **haploidní** (mají jen polovinu chromozómů). Jednotlivé chromozómy se skládají z **genů**, jejichž pozice v chromozómu je určena **lokusem**. Jednotlivé geny jsou složeny z řetězců **nukleových kyselin**. Nukleová kyselina je biochemická makromolekulární látka tvořená polynukleotidovým řetězcem, který ve své struktuře uchovává genetickou informaci. DNA je kombinováno ze čtyř druhů nukleových bází zvaných **cytosin (C)**, **guanin (G)**, **adenin (A)** a **thymin (T)**. Pořadí nukleových bází určuje variantu genu. Variantě DNA pro daný lokus se říká **alela**, která v podstatě vyjadřuje nastavení určitého rysu (například barva očí – modrá, hnědá, apod.). Kompletní sadě genetického materiálu se říká **genom**. Konkrétní sada genů v genomu se nazývá **genotyp**. Fyzikální a mentální projev genotypu se nazývá **fenotyp**. Lidské DNA obsahuje zhruba 3 biliony párů bází

[3]kniha: *Genetic Programming: On the Programming of Computers by Means of Natural Selection*

[4]Lisp - funkcionální programovací jazyk, specifikován v roce 1958. Po Fortranu se jedná o nejstarší vysokoúrovňový jazyk.



(a) DNA



(b) gen, DNA, chromozóm, protein, buňka

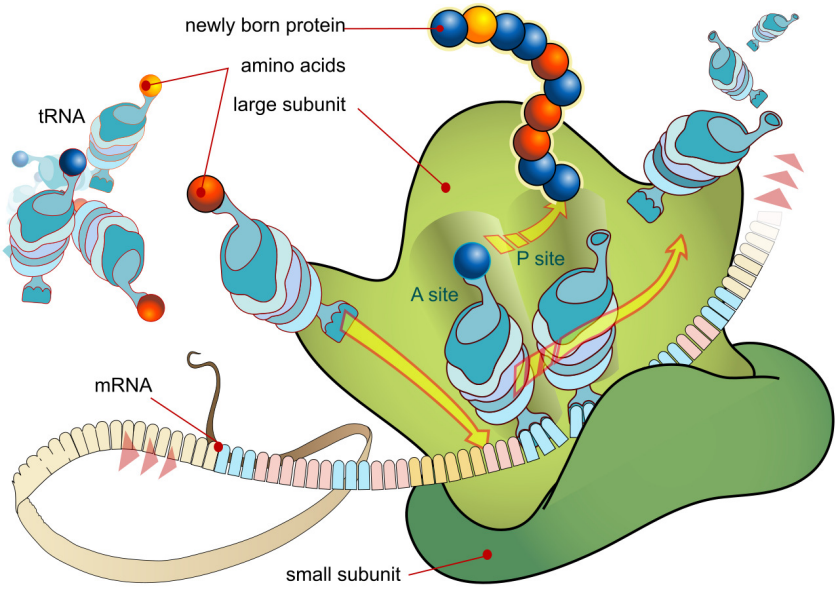
Obrázek 2: Uložení genetické informace

(bp) uspořádaných v lineární sekvenci na chromozómech. Každý chromozóm (s výjimkou pohlavních) je přítomen v páru (**diploidní**). Jeden z chromozómů pochází od matky a druhý od otce. Počet možných sekvenčních kombinací je astronomický.

Transkripcí DNA vzniká mRNA. **mRNA** (mediátorová RNA) je **jednovláknová nukleová kyselina (RNA)**, která slouží k vytvoření **bílkoviny** na základě genetické informace přepsané podle genetického kódu. RNA je tvořena kombinacemi čtyř bází nukleových kyselin, kterými jsou **cytosin (C)**, **guanin (G)**, **adenin (A)** a **uracil (U)**. Kromě informace o stavbě proteinů mají ribonukleové kyseliny (RNA) v buňkách mnoho jiných úloh. **tRNA** plní funkce transportu aminokyselin do ribozomu, **rRNA** se podílí na jeho stavbě, **miRNA** se podílí na regulaci genové exprese a další ....

Ribozom, viz. obrázek 3 je organela nacházející se ve vysokých počtech v **cytoplazmě** všech známých buněk. Jejich hlavní funkcí je tvorba proteinů – bílkovin. Ribozomy jsou poměrně velké komplexní struktury složeny zejména z rRNA a proteinů. K ribozomu se napojuje mediátorová RNA (mRNA), která obsahuje přepis genetické informace. Podle pořadí trojic bází v této genetické informaci přichází k ribozomu na základě genetického kódu jednotlivé **aminokyseliny** napojené na tRNA. Tyto aminokyseliny jsou následně díky katalytickým vlastnostem ribozomu spojeny v protein.

Schopnost RNA nést genetickou informaci a zároveň působit jako katalyzátor biologických reakcí je základem jedné z **teorií vzniku života**, tzv. **teorie RNA světa**.



http://commons.wikimedia.org/wiki/File:Ribosome_mRNA_translation_en.svg

Obrázek 3: Ribozom

3.2 Princip GA

Algoritmus je inicializován sadou řešení reprezentovanou chromozómy. Tato sada řešení se nazývá **populace**. Principem práce genetického algoritmu je postupné zkvalitňování populace tvorbou nových generací jedinců. Tito jedinci jsou do nových populací vytvářeni **reprodukcí**, obvykle dvou rodičů (chromozómů). Každý jedinec populace představuje jedno řešení daného problému. Tradičně je řešení reprezentováno binárními čísly, řetězci nul a jedniček, nicméně používají se i jiné reprezentace (strom, pole, matice, apod.). Typicky je na začátku simulace (v první generaci) populace složena z naprosto náhodných členů. V přechodu do nové generace je každý jedinec ohodnocen tzv. **kriteriální funkcí**, která určí jeho kvalitu mezi ostatními jedinci v populaci. Podle této kvality jsou stochasticky vybráni jedinci k další reprodukci. Reprodukcí se rozumí podrobení rodičů operátorům křížení a mutace, z nichž vzejdou obvykle dva noví jedinci do další generace. Tento postup se iterativně opakuje, čímž se kvalita řešení v populaci postupně vylepšuje. Algoritmus se obvykle zastaví při dosažení postačující kvality řešení, případně po předem dané době. Jednou z variant GA je tzv. **generační model** (*generational GA*), viz. algoritmus 1. U tohoto modelu je vždy celá generace jedinců nahrazena novou generací. Další z variant je **ustálený model** (*steady-state GA*), viz. algoritmus 2, který v každém kroku nahrazuje nejhoršího jedince v populaci jedincem nově vytvořeným.

Výhodou GA je, že nevyžadují speciální znalosti o cílovém řešení, mají vysokou odolnost vůči lokálním extrémům a vykazují dobré výsledky v případě řešení optimalizačních problémů s rozsáhlými množinami přípustných řešení. Jejich nevýhodou je problém s nalezením přesného optima a velké množství vyhodnocování dílčích řešení, což prodlužuje dobu k nalezení postačujícího řešení.

Algoritmus 1: Generační GA (generational GA)

```
t = 0;
InitPopulation P(t);
Evaluate P(t);
while stop condition not satisfied do
    t = t + 1;
    while new population not created do
        P' = SelectParents P(t);
        O = Recombine P'(t);
        Mutate O(t);
        Evaluate O(t);
        O' = AddOffspring O(t);
    P(t + 1) = O';
```

Algoritmus 2: Ustálený GA (steady-state GA)

```
t = 0;
InitPopulation P(t);
Evaluate P(t);
while stop condition not satisfied do
    t = t + 1;
    P' = SelectParents P(t);
    O = Recombine P'(t);
    Mutate O(t);
    Evaluate O(t);
    P_worst(t) = O(t);
```

3.3 Teorie schémat

Teorie schémat *(Schema Theorem)* [18, 14, 33, 35, 17], kterou představil Holland je matematickým popisem, který se snaží objasnit důvod fungování GA. Zakládá se na poměrně jednoduché myšlence sledování genotypu řešení a jeho vlivu na kvalitu jedince.

Pro zjednodušení budeme předpokládat binární chromozómy s konstantní délkou L. Každý gen takového chromozómu může nabývat hodnoty z abecedy:

$$a \in A; \quad A = \{0, 1\} \tag{3.1}$$

Abecedu A rozšíříme o zástupný symbol $*$, který lze použít místo libovolného jiného symbolu z abecedy A:

$$A^+ = \{0, 1, *\}; \quad * \in \{0, 1\} \tag{3.2}$$

Schéma H je řetězcem z abecedy A^+ a představuje šablonu podobnosti jednotlivých chromozómů (řetězců) v populaci z původní abecedy A. Příkladem buď {1**1} jenž představuje vzor, kterému odpovídají řetězce {1001}, {1011}, {1101}, {1111}. U schématu definujeme dva základní charakterizující pojmy. $o(H)$ popisuje **řád schématu** (*schema order*), který je roven počtu definovaných znaků schématu z původní abecedy A. **Definiční délka schématu** (*defining length*) $\delta(H)$ je rovna maximální vzdálenosti definovaných znaků. U předešlého schématu je $o(H) = 2$ a $\delta(H) = 3$. Pro chromozóm délky L a abecedou o kardinalitě k existuje $(k+1)^L$ schémat. V populaci o velikosti n je 2^L až $n \cdot 2^L$

schémat v závislosti na diverzitě populace. V případě, že označíme počet znaků $*$ ve schématu jako r, je počet řetězců, které dané schéma reprezentuje k^r. Do nových generací se šíří schémata prostřednictvím operátoru selekce. Rychlost šíření je přitom závislá na velikosti selekčního tlaku, čili na použitém operátoru selekce. Operátorem křížení se do nových generací šíří především schémata s krátkou délkou, jelikož u delších schémat se zvyšuje pravděpodobnost, že budou křížením rozdělena. V případě použití binárního kódování, proporcionální selekce, jednobodového křížení a náhodné jednobodové mutace můžeme rychlost šíření krátkých schémat s nejlepším fitness definovat jako:

$$E[m(H,t+1)] \geq m(H,t) \cdot \frac{f(H,t)}{\bar{f}(t)} \cdot \left\{ 1 - p_c \frac{\delta(H)}{1-L} \cdot p_{diff}(H,t) - o(H) \cdot p_m \right\} \tag{3.3}$$

Kde: E[m(H,t + 1)] představuje očekávaný počet schématu H v generaci $t+1$,
$m(H,t)$ je počet instancí schématu H v generaci t,
$f(H,t)$ je průměrné fitness chromozómů reprezentující schéma H,
$\bar{f}(t)$ je průměrné fitness populace v čase t,
$p_{diff}(H,t)$ je pravděpodobnost, že druhý rodič neobsahuje schéma H,
p_c je pravděpodobnost křížení,
p_m je pravděpodobnost mutace.

3.4 Prohledávaný prostor

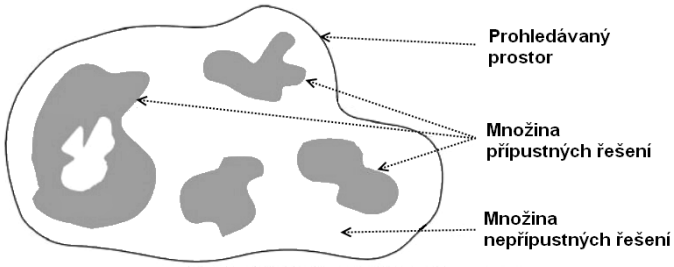
Množina všech potenciálních řešení se nazývá prohledávaný prostor řešení, viz. obrázek 4. Každý bod v tomto prostoru reprezentuje jedno možné řešení, které je označeno hodnotou fitness, představující kvalitu jedince v prohledávaném prostoru. Vzhledem k velikosti stavového prostoru může být jeho systematické prohledávání velmi neefektivní, jelikož se zbytečně prohledává značná část stavového prostoru, která nevede k cíli. Velikost prohledávaného prostoru lze omezit vložením omezujících podmínek vyplývajících ze znalostí řešeného problému. Znalosti mají někdy empirický charakter a mohou to být neexaktní poznatky, které jsou často užitečné při řešení, ale nezaručují, že povedou k řešení. Pomocí GA hledáme v prostoru potenciálních řešení globální extrém (minimum nebo maximum), který představuje nejlepší řešení daného problému z množiny všech potencionálních řešení. Základní problém spočívá v tom, že prohledávání může být velmi komplikované. Často je těžké vůbec řešení najít nebo rozhodnout, kde s prohledáváním začít. Existuje řada optimalizačních metod, které se snaží najít globálně optimální řešení, která jsou ovšem často pouze suboptimální, jelikož ve většině řešených případů nemáme představu o tom, jak by vlastně mělo optimální řešení vypadat.

Evoluční proces prohledávání prostoru potenciálních řešení vyžaduje rovnováhu dvou protichůdných cílů:

- co nejrychleji najít lokální extrém v malém okolí výchozího bodu
- co nejlépe prohledat prostor všech možných řešení.

Jednotlivé optimalizační metody se liší svým zaměřením k těmto dvěma cílům a je možno je orientačně seřadit podle schopnosti najít lokální optimum až k metodám prohledávajícím velký prostor řešení takto:

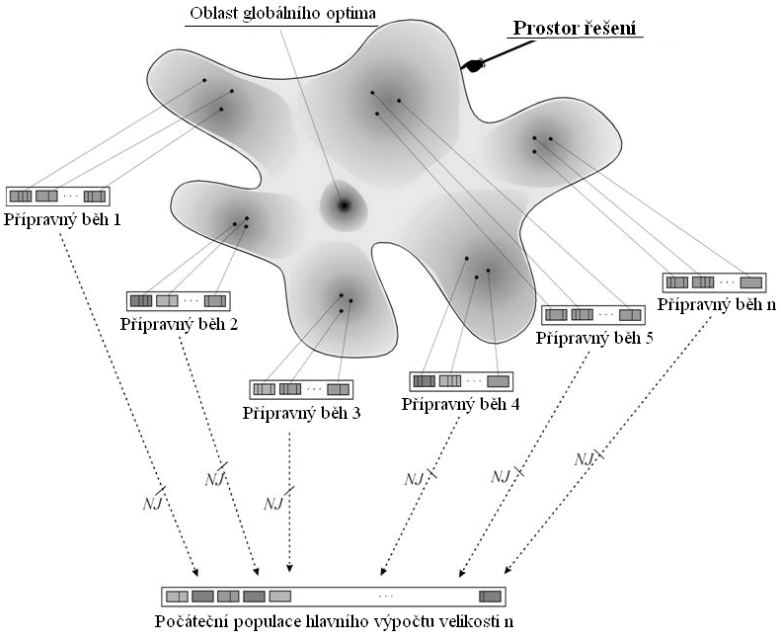
1. stochastický horolezecký algoritmus *(Stochastic Hill Climbing)*
2. zakázané prohledávání *(Tabu Search)*
3. simulované žíhání *(Simulated Annealing)*
4. evoluční strategie *(Evolution Strategies)*
5. genetické algoritmy *(Genetic Algorithms)*



Obrázek 4: Prohledávaný prostor

3.5 Inicializace

Po spuštění GA je nutno inicializovat první populaci jedinců. Od této populace se bude odvíjet další vývoj optimalizace. Často se používá náhodná inicializace, která však není vhodná např. při práci s omezeními, kdy je vhodné vytvořit ji z jedinců obsahujících přípustné řešení. V určitých případech je proto nutné vytvořit vlastní způsob inicializace populace pro konkrétní řešený problém. Počáteční populace totiž musí obsahovat dostatek



Obrázek 5: Předzpracování počáteční populace jedinců GA

informací, ze kterých by GA mohl vyvinout kvalitní řešení, takže by měla rovnoměrně pokrývat celý stavový prostor, viz. obrázek 5.

3.6 Kódování chromozómů

Kódování chromozómů je jedním z úkolů při návrhu GA. Kódování je závislé na typu optimalizačního problému. Chromozóm může být tvořen například jediným celočíselným genem, celočíselným vektorem, případně vektorem složeným z reálných čísel, apod. [51].

3.6.1 Binární kódování

Binární kódování, viz. tabulka 1, poskytuje mnoho variant chromozómů s nízkým počtem alel. Prohledávaný prostor Σ je určen jako $\Sigma = \{0, 1\}^L$, kde L je délka chromozómu c. V případě použití tohoto kódování pro uložení celých čísel, je nutné zvolit vhodně kódování tak, aby nedocházelo ke zkreslenému zobrazení. To se projevuje zejména při křížení a mutaci, kdy malá změna genotypu způsobí rozsáhlou změnu fenotypu a naopak. Tento efekt se nazývá **Hammingova bariéra** *(Hamming Cliff)*, viz. tabulka 1, a může snižovat výkonnost algoritmu.

jedinec	bin	dec
chromosome A	01111111	127
chromosome B	10000000	128

bin: dvojková soustava, **dec:** desítková soustava

Tabulka 1: Binární kódování, Hammingova bariéra

V důsledku tohoto je výhodné použít **Grayův kód**, ve kterém se sousední hodnoty liší vždy v jednom bitu. Převod z Grayova kódu do binárního a naopak lze provést např. [2]:

```
int BinaryToGray(int num)
{
    return (num >> 1) ^ num;
}

int GrayToBinary(int num)
{
    int temp = num ^ (num >> 8);
    temp ^= (temp >> 4);
    temp ^= (temp >> 2);
    temp ^= (temp >> 1);
    return temp;
}
```

Binární kódování lze použít také pro uložení reálných čísel. Ty sice jsou v počítači uložena binárně, ale jejich uložení se řídí normou **IEEE 754** [48], (viz. kapitola 14.1), tabulka 27. Díky této specifikaci se pro GA stává nepoužitelným a je vhodné zvolit jinou metodu

kódování. K transformaci reálných čísel do binárního tvaru a naopak lze použit vztahy uvedené níže:

$$x_{real} \rightarrow x_{10} = \left[\frac{x_{real} - a}{b - a}(2^k - 1)\right] \tag{3.4}$$

$$x_{10} \rightarrow x_{real} = \left[\frac{x_{10}(b - a)}{2^{k-1}} + a\right] \tag{3.5}$$

$$k = \underbrace{\left\lceil \frac{\ln[(b - a)]10^q}{\ln 2} \right\rceil}_{\beta} \tag{3.6}$$

Kde: x_{real} reálné číslo,
x_{10} dekadický tvar reálného čísla po transformaci,
$a, b \in \mathbb{R}$, je minimální, resp. maximální hodnota kódovaného intervalu,
$x_{real} \in \langle a, b \rangle$,
k je délka binárního řetězce po transformaci,
q je přesnost kódovaného čísla a udává počet platných desetinných míst,
$\lceil \beta \rceil$ je nejbližší **větší** celočíselná hodnota reálného čísla β.

Podle těchto vztahů je možno transformovat reálné číslo s požadovanou přesností na dekadický tvar a v této podobně jej převést do binárního tvaru a naopak. Ukázka transformace $\mathbb{R} \rightarrow \mathbb{N} \rightarrow$ Gray Code $\rightarrow$ *bin*, (viz. kapitola 14.2), tabulka 28.

3.6.2 Celočíselné kódování

Je používáno pro úlohy, které jsou přímo definované na množině $\mathbb{N}$, jelikož řešení není nutno nijak převádět. Prohledávaný prostor Σ je určen jako $\Sigma = \varkappa^k$, kde $\varkappa$ je délka celočíselného řetězce, k je kardinalita (mohutnost) množiny. Jedním z typů celočíselného kódování je *pořadové kódování*, viz. tabulka 2, které může být použito pro úlohy hledání nejkratší cesty v grafu. Touto úlohou může být například *problém obchodního cestujícího.*

chromosome A	1 5 3 2 6 4 7 9 8
chromosome B	8 5 6 7 2 3 1 4 9

Tabulka 2: Celočíselné pořadové kódování

3.6.3 Kódování pomocí reálných čísel

Je používáno pro optimalizační úlohy ve kterých se optimalizují proměnné z $\mathbb{R}$, zejména proto, že nedochází ke ztrátám přesnosti při kódování a dekódování čísla. Velikost prohledávaného prostoru je:

$$\Sigma = \frac{(a_{max} - a_{min})^L}{s} \tag{3.7}$$

Kde: a_{max} a a_{min} je horní, resp. dolní mez prohledávaného intervalu,
s je nejmenší krok, se kterým je možné se v intervalu pohybovat,
L je délka chromozómu.

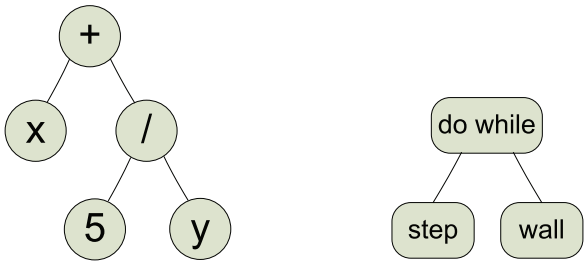
Tento typ kódování je používán zejména na hranici metod genetických algoritmů a evolučních strategií a je využíván např. V diferenciální evoluci, (viz. kapitola 4).

chromosome A	1.2324 5.3243 0.4556 2.3293 2.4545
chromosome B	9.0004 3.1253 7.3214 0.0003 2.5575

Tabulka 3: Kódování pomocí reálných čísel

3.6.4 Kódování pomocí stromu

Je používáno hlavně ve spojení s GP pro generování programů, nebo výrazů. V tomto kódování je každý chromozóm reprezentován stromem objektů. Objektem může být např. funkce, příkaz jazyka, apod. viz. obrázek 6.



Obrázek 6: Kódování pomocí stromu

Způsob kódování je silně implementačně závislý na řešené úloze a měl by se volit s důrazem na minimální zkreslení. Pokud není zachována blízkost fenotypů, dochází v interním prohledávaném prostoru algoritmu k vytváření lokálních minim, která v prohledávaném prostoru nejsou. V důsledku tohoto se zbytečně zvyšuje složitost a časová náročnost úlohy.

Kromě výše zmíněných druhů kódování může být použito i kódování hodnotou *(value encoding)*, tedy ukládat do chromozómu přímo hodnoty fenotypu bez kódování. Výhoda tohoto způsobu je v tom, že interní prohledávaný prostor je totožný s prostorem řešení a nedochází tedy k vytváření více lokálních extrémů, než je v původní úloze. V závislosti na řešeném problému je možné za určitých předpokladů do tohoto typu kódování zařadit i výše zmíněné způsoby kódování. V závislosti na použitém druhu kódování je nutno navrhnout specifické druhy operátorů křížení a mutace.

3.7 Reprodukce

Klasické GA využívají pro vytváření nových generací jedinců operátory selekce (viz. kapitola 3.10), křížení (viz. kapitola 3.8), mutace (viz. kapitola 3.9) a reprodukce. Operátor reprodukce kopíruje chromozómy do nové populace, přičemž jsou zvýhodňováni kvalitnější jedinci, jenž jsou vybíráni podle hodnoty fitness. Zvolená metoda výběru jedinců do

další generace ovlivňuje velikost selekčního tlaku a velikost prohledávaného prostoru (viz. kapitola 3.12).

Genetické operátory se aplikují na rodičovské chromozómy, či jejich dvojice, ze kterých se vytvářejí potomci, kteří se poté pomocí operátoru reprodukce začlení do nové populace. Při použití běžných selektivních metod se může stát, že k reprodukci nebude vybrán nejlepší jedinec, nebo několik nejlepších jedinců z populace. Tomuto se dá zabránit začleněním elitářství (elitismu), (viz. kapitola 3.10.9).

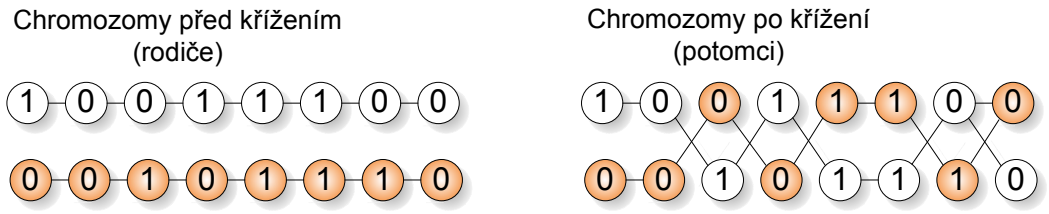
3.8 Křížení

Operace křížení *(crossover)* je považována za jeden ze základních pilířů GA zaručujících jejich efektivnost a je nástrojem pro vytváření nových řešení do dalších populací. Křížení je metoda, která slouží k zajištění konvergence a je závislá na použité metodě kódování, protože pracuje na úrovni chromozómů a genů. Křížení je hlavní způsob výměny informací mezi jedinci. Z populace jsou vybráni nejčastěji dva jedinci (rodiče), (viz. kapitola 3.10) a kombinací jejich genetických informací vznikne jeden, nebo několik potomků (nejčastěji dva). V ideálním případě je potomek *(offspring)* kvalitnější, než rodiče a GA postupuje ve směru gradientu kriteriální funkce. S určitou pravděpodobností je však možný i stav, kdy potomek je méně kvalitní než jeho rodiče, GA postupuje proti směru gradientu kriteriální funkce, což mu umožňuje překonat lokální extrém účelové funkce. Křížení je obvykle aplikováno jen s určitou (poměrně vysokou) pravděpodobností p_c a v ostatních případech se potomci vytvoří jako identické kopie svých rodičů.

Operátory křížení se dají realizovat mnoha způsoby, ale dají se obecně rozdělit do tří základních skupin:

3.8.1 Uniformní křížení

Zaměňuje s určitou pravděpodobností jednotlivé geny rodičovských chromozómů, viz. obrázek 7. Obecně tato metoda způsobuje velkou změnu chromozómu a může podobně jako mutace narušovat konvergenci.

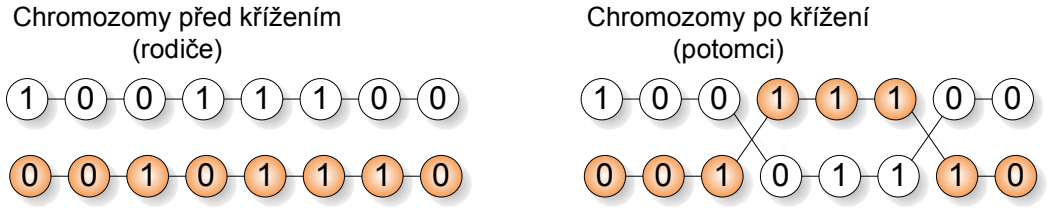


Obrázek 7: Uniformní křížení

3.8.2 Bodové křížení

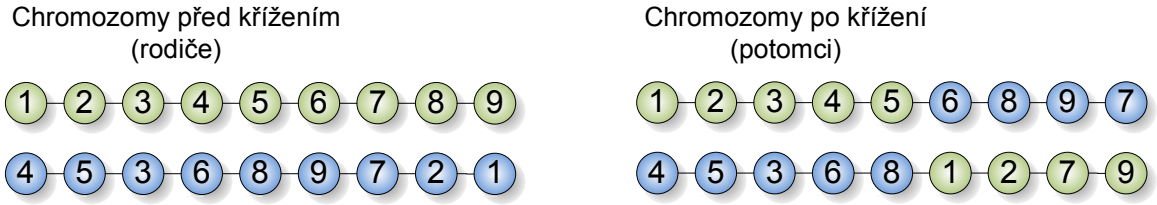
Tento způsob křížení zaměňuje jednotlivé úseky rodičovských chromozómů. Obecně N bodové křížení zvolí N různých pozic v rodičovských chromozómech a za každou z těchto pozic vzájemně zaměňuje jejich části, viz. obrázek 8.

V případě pořadového kódování chromozómů je možno realizovat jednobodové křížení následujícím způsobem: Předpokládejme dva chromozómy délky L, náhodně je vybrán



Obrázek 8: Dvoubodové křížení

bod křížení k pro který platí $\{k \in \mathbb{Z}; 0 \leq k \leq (L-2)\}$. Potomek se vytvoří kombinací dvou chromozómů tak, že obsah jednoho rodičovského chromozómu je zkopírován od počátku až do zvoleného bodu křížení do potomka. Zbytek se doplní z druhého rodiče, který se postupně prochází od počátku do konce. Prvek z druhého rodiče je přidán do potomka v případě, že ještě není v potomkovi obsažen, viz. obrázek 9.



Obrázek 9: Jednobodové křížení (pořadové kódování)

3.8.3 Aritmetické křížení

Může být v binární reprezentaci chromozómu realizováno např. pomocí funkce AND. Tímto druhem křížení vznikne z vybraných dvou rodičů jeden potomek. V chromozómu kódovaném reálnými čísly může být křížení realizováno např. jako v případě DE, (viz. kapitola 4.3.1).

3.9 Mutace

Mutace je unární operátor, který se aplikuje s pravděpodobností p_m na vybrané jedince v populaci. Účel operátoru mutace je vložit do populace novou genetickou informaci, kterou tato populace neobsahuje. Pokud budeme předpokládat např. chromozóm s binárním kódováním, jednobodové křížení a populaci $P(t)$ v čase t, v níž ani jeden chromozóm $c_i^j(t)$ neobsahuje na pozici i symbol 1, nemá operátor křížení prostředky k tomu, jak jedince se symbolem 1 na pozici i získat. Přitom právě tyto geny mohou být pro nalezení dostatečně kvalitního řešení klíčové. Na druhou stranu však mutace významně narušuje genetickou informaci, tudíž je účelné ji aplikovat s nízkou pravděpodobností. V případě binárního kódování chromozómů se realizuje jako bitová inverze, viz. obrázek 10.

V případě pořadového kódování se v chromozómu vzájemně vymění vybrané dva geny, viz. obrázek 11.

V chromozómech kódovaných pomocí reálných čísel je možno mutaci realizovat přičtením, resp. odečtením malé hodnoty od mutovaného genu, případně tak jako je tomu v DE, (viz. kapitola 4.3.1).

Obrázek 10: Inverzní mutace jednoho genu

Obrázek 11: Jednobodová mutace (pořadové kódování)

3.10 Selekce

Selekce se snaží napodobit Darwinovskou[5] teorii přirozeného výběru a přežití nejlepších. Jejím úkolem je vybrat z populace jedince, kteří se budou podílet na vytvoření nových jedinců. Pro zajištění konvergence jsou kvalitnější jedinci vybíráni s větší pravděpodobností s předpokladem, že budou plodit kvalitnější potomky. Operátor selekce patří mezi nejdůležitější fáze návrhu GA a rozhoduje o jeho schopnosti nalézt globální optimum, nebo se mu maximálně přiblížit. Rychlost konvergence je dána selekčním tlakem, (viz. kapitola 3.12) a závisí na zvolené metodě použité k výběru jedinců. Selekční metody je možno realizovat jako jednokriteriální, nebo vícekriteriální. **Jednokriteriální selekční metody** vybírají jedince pouze podle hodnoty fitness, **vícekriteriální selekční metody** zohledňují také další parametry, jako např. stáří jedince, složitost, apod. Metody selekce se dají realizovat mnoha způsoby z nichž základní jsou uvedeny níže. Podrobné srovnání metod je uvedeno v [6, 34]:

3.10.1 Roulette-wheel selection

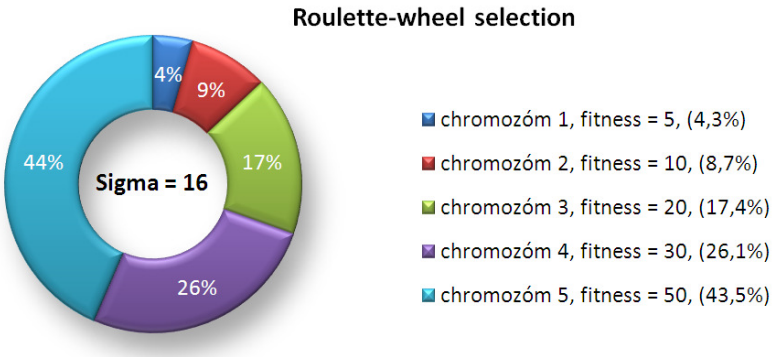
Selekce ruletovým kolem, také známá jako **proporcionální selekce** *(proportionate selection)* je jednou z nejstarších selekčních metod. Pravděpodobnost výběru jedince je dána vztahem:

$$p_i = \frac{F_i}{\sum_{j=1}^{n} F_j}; \quad i \in (1, \ldots, n) \tag{3.8}$$

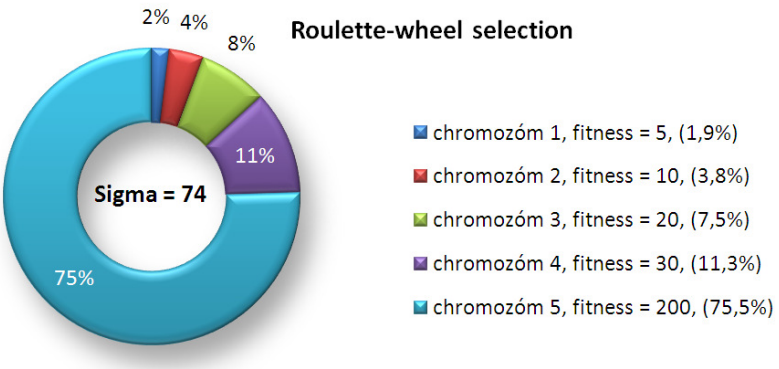
Kde: n je velikost populace,
F_i je fitness i-tého jedince v populaci.

Hlavní nevýhodou této selekční metody je její přímá závislost na hodnotě fitness, která se může za určitých okolností projevit velkým selekčním tlakem a předčasnou konvergencí. Tento jev může nastat, když jeden jedinec má výrazně lepší fitness, než ostatní, viz. obrázek 12b.

[5] Charles Robert Darwin (1809 – 1882)



(a) Směrodaná odchylka $\sigma = 16$



(b) Směrodaná odchylka $\sigma = 74$

Obrázek 12: Roulette-wheel selection

3.10.2 Linear ranking

Lineární klasifikace patří mezi metody založené na pořadí *(rank-based methods)*. Pro výběr jedinců se nevyužívá přímo hodnota fitness, ale pozice jedince v populaci, která je vzestupně seřazená podle fitness. To má například za následek různou pravděpodobnost výběru jedinců se stejnou hodnotou fitness, jelikož pravděpodobnost výběru závisí na indexu jedince v populaci. Pravděpodobnost výběru k-tého jedince z populace je možno vyjádřit jako [10]:

$$p(x_k^{(t)}) = \frac{1}{n}(\mu^- + \frac{(\mu^+ - \mu^-)(k-1)}{n-1}); \quad k \in (1, \ldots, n) \tag{3.9}$$

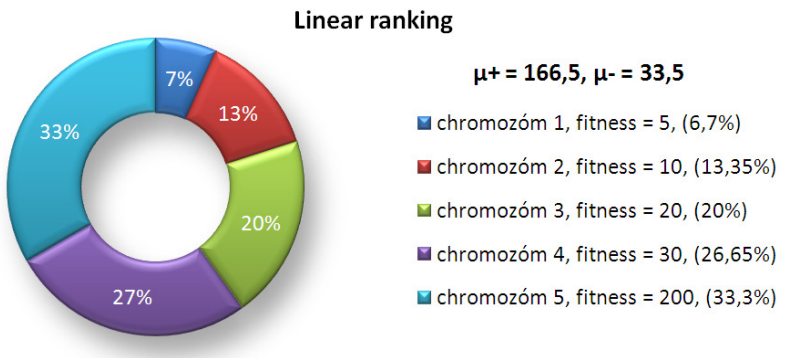
Kde: n je velikost populace,

F_i je fitness i-tého jedince v populaci,

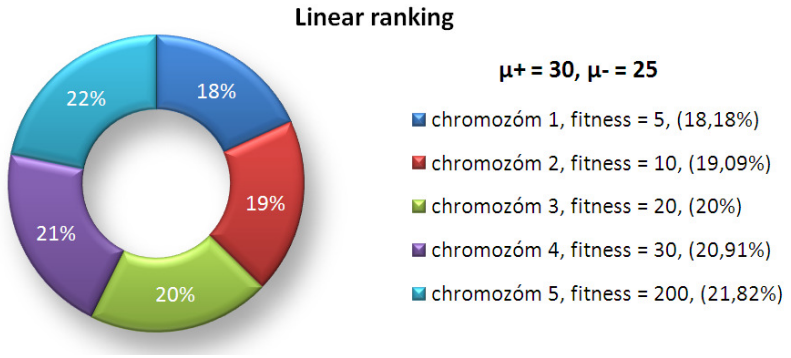
μ^- a μ^+ jsou volitelné parametry, které ovlivňují pravděpodobnost výběru nejhoršího a nejlepšího jedince

Pravděpodobnost výběru nejlepšího, (resp. nejhoršího) jedince je vyjádřena jako:

$$p_{x_{best}^{(t)}} = \frac{\mu^+}{n}, \qquad p_{x_{worst}^{(t)}} = \frac{\mu^-}{n} \tag{3.10}$$



(a) Linear ranking s trojúhelníkovou distribucí pravděpodobnosti ($\mu^- = 33{,}5, \mu^+ = 166{,}5$)



(b) Linear ranking ($\mu^- = 25, \mu^+ = 30$)

Obrázek 13: Linear ranking

Vliv parametrů na rozložení pravděpodobnosti výběru i-tého jedince je ukázán na obrázku, viz. obrázek 13.

3.10.3 Exponential ranking

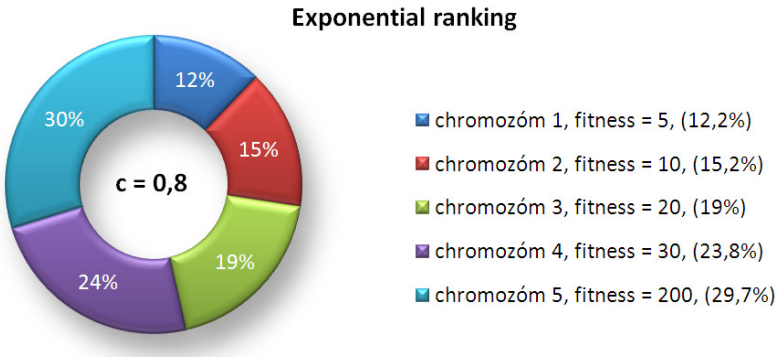
Exponenciální klasifikace je stejně jako lineární klasifikace metodou, která vybírá jedince z populace seřazené vzestupně podle fitness. Od lineární klasifikace se liší tím, že jedinci mají exponenciální rozložení pravděpodobnosti výběru, které je možno vyjádřit vztahem:

$$p(x_k^{(t)}) = \frac{c^{k-1}}{\Sigma_{j=1}^{n} c^{j-1}}; \quad k \in (1, \ldots, n) \tag{3.11}$$

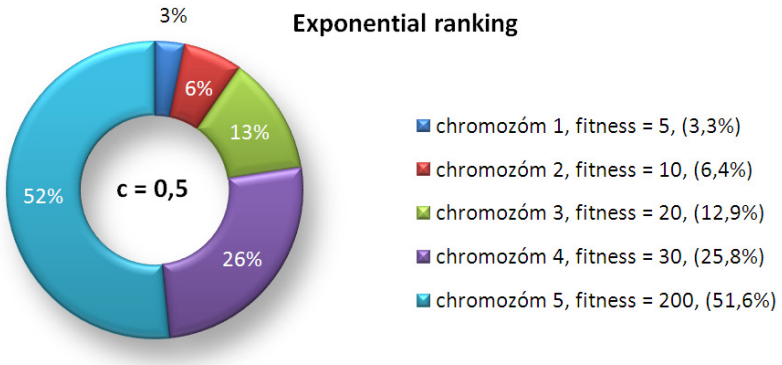
Kde: n je velikost populace,

c je volitelný parametr, který ovlivňuje rozložení pravděpodobnosti výběru jedinců, pro který platí $0 < c < 1$.

Ukázka rozložení pravděpodobnosti a vliv parametru c je ukázán na obrázku 14. Exponenciální klasifikace umožňuje poměrně kvalitní nastavení poměru selekčního tlaku a diverzity populace. Nevýhodou této metody je vyšší časová složitost $O(n \ln(n))$.



(a) $c = 0,8$



(b) $c = 0,5$

Obrázek 14: Exponential ranking

3.10.4 Tournament selection

Metoda turnajové selekce náhodně vybere z populace velikosti n skupinu t jedinců, pro které platí: $t \in \mathbb{Z}; 2 \leq t \leq n$. Mezi vybranými jedinci uspořádá turnaj a nejlepší jedinec z turnaje je vybrán k reprodukci.

Existuje i další varianta turnajového výběru [33], která obsahuje volitelný parametr $r > 0{,}5$, jenž ovlivňuje výběr jedince v turnaji. Zároveň se skupinou t jedinců je náhodně vygenerováno náhodně číslo $k \in \langle 0, 1 \rangle$. V případě, že $k < r$, je vybrán jedinec s vyšším fitness, v opačném případě jedinec s nižším fitness.

3.10.5 Sigma scaling

Metoda se snaží udržet selekční tlak po dobu evoluce relativně konstantní. K určení pravděpodobnosti výběru jedince se využívá transformovaná hodnota fitness, která je funkcí fitness, průměrného fitness v populaci a výběrové směrodatné odchylky a je možno ji vyjádřit ve tvaru:

$$f_{T_k}^{(t)} = \begin{cases} 1 + \frac{f_i - \bar{f}}{2\sigma} & \text{pro } \sigma \neq 0 \\ 1 & \text{pro } \sigma = 0 \end{cases} \tag{3.12}$$

Kde: $f_{T_k}^{(t)}$ je transformovaná hodnota fitness,
$\bar{f}$ je průměrné fitness v populaci, (Eq. 3.13),

σ je výběrová směrodatná odchylka, (Eq. 3.14).

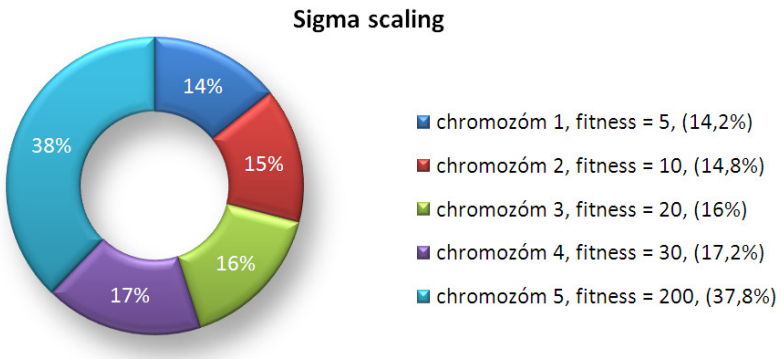
$$\bar{f} = \frac{1}{n}\sum_{j=1}^{n} f_j \tag{3.13}$$

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{j=1}^{n}(f_j - \bar{f})^2} \tag{3.14}$$

Pravděpodobnost výběru k-tého jedince je tedy možno vyjádřit ve tvaru:

$$p(f_{T_k}^{(t)}) = \frac{f_{T_k}^{(t)}}{\sum_{i=1}^{n} f_{T_i}^{(t)}} \tag{3.15}$$

Rozložení pravděpodobnosti výběru jedinců je ukázáno na obrázku 15.

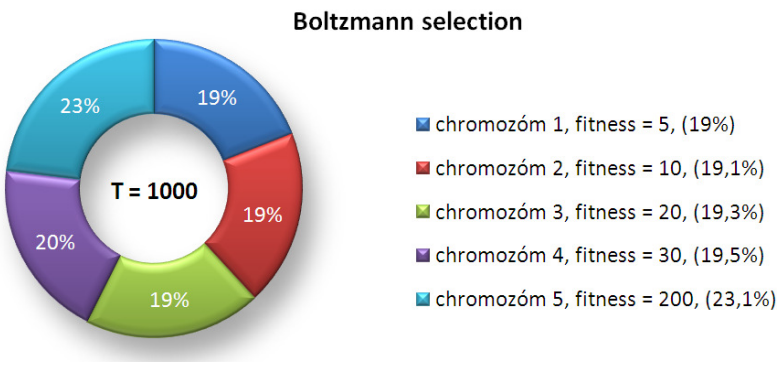


Obrázek 15: Sigma scalling

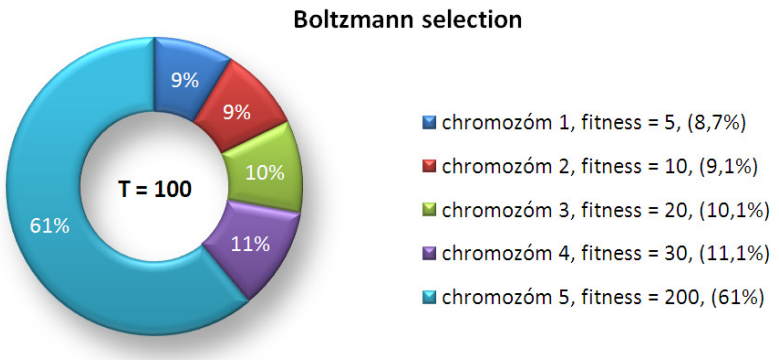
3.10.6 Boltzmann selection

Boltzmanův výběr je metoda inspirovaná metodou simulovaného žíhání. K prohledávání prostoru používá řízený selekční tlak, který je řízen teplotou. Na počátku evoluce je teplota vysoká, která způsobí, že všichni jedinci získají přibližně stejnou pravděpodobnost výběru. Díky tomu bude na populaci působit nižší selekční tlak a GA prohledá větší prostor řešení. Teplota se v čase postupně snižuje, což způsobí zesílení selekčního tlaku a urychlení konvergence GA. Vliv teploty na rozložení pravděpodobnosti výběru je ukázán na obrázku 16. Srovnání vlivu teploty ve dvou stejných populacích je na obrázku 16a a 16b. Na obrázku 16c je ukázáno rozdělení pravděpodobnosti výběru jedinců ke konci evoluce, kdy se všichni jedinci shlukují kolem extrému funkce. Pravděpodobnost výběru jedince je dána jeho transformací fitness a je možné ji vyjádřit ve tvaru:

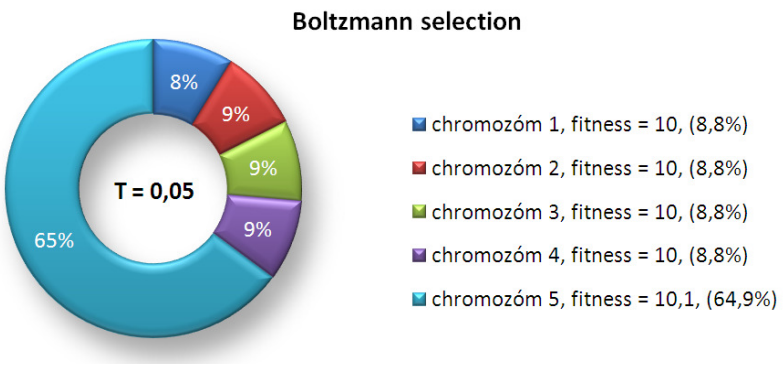
$$f_{T_k}^{(t)} = \frac{n \cdot e^{\frac{f_k}{\tau}}}{\sum_{j=1}^{n} f_j} \tag{3.16}$$



(a) $T = 1000$



(b) $T = 100$



(c) $T = 0,05$

Obrázek 16: Boltzmann selection

Kde: $f_{T_k}^{(t)}$ je hodnota transformovaného fitness k–tého jedince v populaci,
n je velikost populace,
τ je teplota,
f_k, f_j jsou hodnoty fitness před transformací.

Pravděpodobnost výběru k-tého jedince je tedy možno vyjádřit ve tvaru:

$$p(f_{T_k}^{(t)}) = \frac{f_{T_k}^{(t)}}{\sum_{i=1}^{n} f_{T_i}^{(t)}} \tag{3.17}$$

3.10.7 Zkrácený výběr (Truncation selection)

Tato metoda vybírá z populace N nejlepších jedinců seřazených podle fitness. Volitelným parametrem této metody je, tzv. práh zkrácení *(trunc)*, který udává procentuální podíl vybraných jedinců. Zřejmou nevýhodou této metody je ztráta diverzity populace a předčasná konvergence. Vzhledem k nutnosti seřazení populace podle fitness je časová složitost metody $O(N \log N)$.

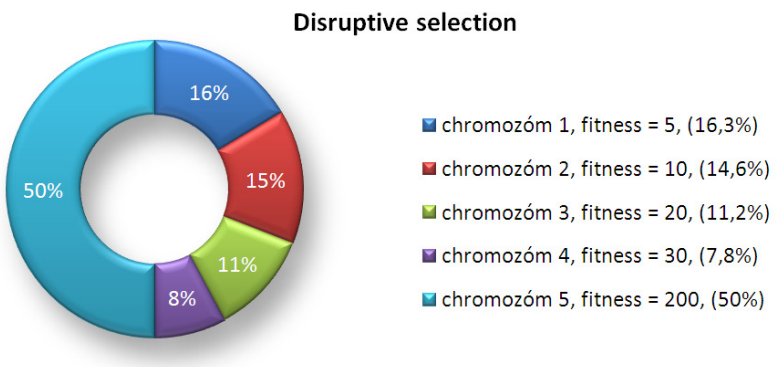
3.10.8 Disruptive selection

Tento typ selekce potlačuje průměrné jedince. Jedincům se přiřadí transformovaná hodnota fitness, která je vyjádřena jako absolutní hodnota rozdílu fitness jedince a průměru populace. To má za následek větší pravděpodobnost výběru jedinců s větší absolutní hodnotou rozdílu fitness od průměrného fitness v populaci. Pravděpodobnost výběru k-tého jedince je vyjádřena jako:

$$p(f_{T_k}^{(t)}) = \frac{|f_k - \bar{f}|}{\sum_{i=1}^{n} |f_i - \bar{f}|} \tag{3.18}$$

Kde: f_k je fitness k-tého jedince v populaci před transformací,
$\bar{f}$ je průměrná hodnota fitness v populaci před transformací.

Rozložení pravděpodobnosti výběru jedinců v případě použití této selekční metody je ukázáno na obrázku 17.



Obrázek 17: Disruptive selection

3.10.9 Elitismus

Elitismus *(elitism)* má dva významy:

- První pohled na elitismus je takový, kdy účelem je přímé přenesení jednoho, nebo několika nejlepších jedinců do další generace. Své opodstatnění má v případě použití selekčních metod u kterých se může stát, že nejlepší jedinec nebude přenesen do další generace. Takovými metodami jsou například ruletové kolo, tournament, apod. V takovém případě může použití elitismu výrazně zvýšit výkon GA, jelikož nedochází ke ztrátám nejlepších řešení z populace.

- Druhý pohled upřednostňuje lepší řešení před ostatními. Např. V případě turnajové selekce, (viz. kapitola 3.10.4), je turnaj $t = 3$ více elitistický, než turnaj $t = 2$.

3.11 Zabránění incestu (Incest prevention)

Je jedním z mechanismů, který se může použít na zabránění předčasné konvergence GA a pracuje na principu porovnávání podobnosti jedinců.

- Pro porovnání podobnosti jedinců, jejichž chromozóm je kódovaný ve formě binárních čísel je možno použít Hammingovy metriky. **Hammingova vzdálenost** je rovna počtu lišících se bodů dvou chromozómů stejné délky.
- Pro výpočet vzdálenosti dvou jedinců kódovaných ve formě reálných čísel je možno použít **Euklidovské metriky**, (Eq. 3.19):

$$m_e(\vec{a}, \vec{b}) = \sqrt{\sum_{i=1}^{d} (a_i - b_i)^2} \tag{3.19}$$

Kde: d značí počet dimenzí vektoru reálných čísel.

V případě, že dva vybraní jedinci mají hodnotu Euklidovské (resp. Hammingovy) vzdálenosti menší, než je zadaný práh, tak se vzájemně nekříží. Hodnota prahu se může během evolučního vývoje snižovat společně se zvyšujícím se uspořádáním populace [15].

3.12 Selekční tlak

Velikost selekčního tlaku je možno vyjádřit vztahem:

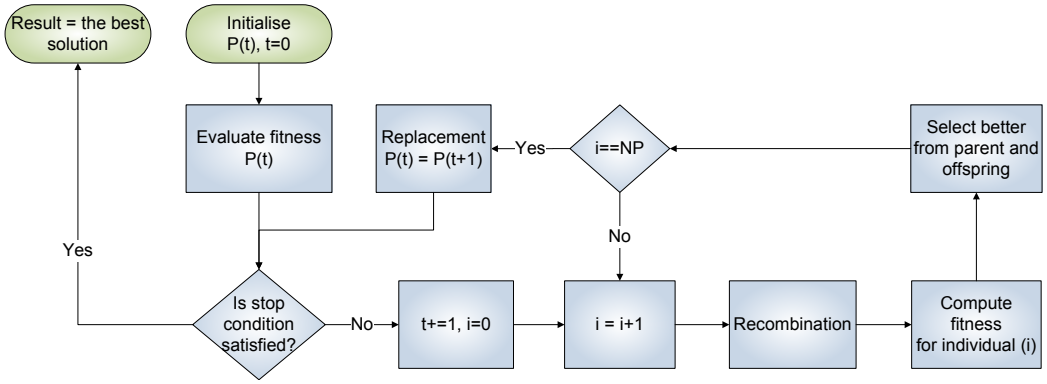
$$I = \frac{\bar{F}^* - \bar{F}}{\bar{\sigma}} \tag{3.20}$$

Kde: $\bar{F}$ je průměrná hodnota fitness před selekcí,
$\bar{F}^*$ je průměrná hodnota fitness po selekci,
$\bar{\sigma}$ je rozptyl fitness po selekci.

Velikost selekčního tlaku má přímý vliv na rychlost konvergence a velikost prohledávaného prostoru. Rychlost konvergence *(exploitation)* a velikost prohledávaného prostoru *(exploration)* jsou dva protichůdné požadavky. V případě, že je selekční tlak příliš velký, dojde v populaci ke ztrátě diverzity a hrozí riziko uvíznutí v lokálním extrému. Prohledávaný prostor řešení je tím menší, čím větší je selekční tlak a tím pádem i rychlost konvergence. Naopak se snižujícím se selekčním tlakem se zvyšuje velikost prohledávaného prostoru a klesá rychlost konvergence. Cílem je nastavit velikost selekčního tlaku tak, aby GA našel co nejrychleji optimální řešení zadaného problému a zároveň prohledal co největší část prostoru řešení.

4 Diferenciální evoluce

Diferenciální evoluce (DE) je variantou genetického algoritmu, která je navržena pro numerickou optimalizaci parametrů v oboru reálných čísel, ale je možno ji využít i k řešení diskrétních úloh [24]. Výhoda DE spočívá v jednoduchosti implementace, rychlosti a efektivitě [36]. Zjednodušený vývojový diagram DE je ukázán na obrázku 18.



P(**t**): populace v čase t, **NP**: velikost populace

Obrázek 18: Vývojový diagram diferenciální evoluce

4.1 Řídící parametry

DE pracuje s množinou NP, D-dimenzionálních parametrických vektorů $x_i^D(t)$, kde $i = 1, 2, \ldots, NP$, které tvoří populaci každé generace G v čase t. Pro optimální funkci diferenciální evoluce je důležité správné nastavení velikosti populace NP, faktoru zesílení β a pravděpodobnosti křížení CR. Velikost dimenze D vektoru je dána počtem optimalizovaných parametrů řešeného problému.

- **Velikost populace:** V rovnici (4.3) je ukázáno, že velikost populace NP má přímý vliv na velikost prohledávaného prostoru. Se zvětšující se velikosti populace, roste velikost prohledávaného prostoru, ale také časová složitost algoritmu. Empiricky bylo zjištěno, že velikost populace NP $\approx$ 10D. Z důvodu zkrácení výpočetního času se ve skutečnosti používá menší počet jedinců za cenu snížení prohledávaného prostoru NP $> 2n_v + 1$, kde n_v je počet použitých rozdílových vektorů, viz. (4.3.1)

- **Faktor zesílení:** Určuje velikost mutace, neboli zesílení diferenčního vektoru, viz. (4.3.1). Platí, že $\beta \in (0, \infty)$, přičemž malá hodnota faktoru zesílení má za následek delší dobu optimalizace a možnost uváznutí v lokálním minimu, a naopak velká hodnota β vede k riziku přeskočení globálního optima. Faktor zesílení by měl být snížen s rostoucí velikostí populace. Empirické studie dokazují, že velká hodnota NP a β často způsobí předčasnou konvergenci. Doporučená hodnota $\beta = 0{,}5$.

- **Pravděpodobnost křížení:** Má přímý vliv na diverzitu populace diferenciální evoluce. Velikost CR určuje počet zaměněných parametrů rodičovského vektoru $x_i^k(t)$, viz. (4.3). Velká hodnota parametru CR vede k zvýšení diverzity populace a k snížení prohledávání blízkého okolí jednotlivých jedinců v populaci. Zvýšení hodnoty

CR často vede k rychlejší konvergenci, na druhou stranu snížení pravděpodobnosti křížení zvyšuje robustnost *DE*.

4.2 Inicializace populace

Inicializace populace se provádí náhodným generováním *NP* vektorů o velikosti k, která je dána počtem optimalizovaných parametrů:

$$\vec{x}_i^{D_k}(0) = \vec{a}^{\min_k} + r \cdot (\vec{a}^{\max_k} - \vec{a}^{\min_k}), \tag{4.1}$$

kde r je náhodně generované reálné číslo z intervalu $< 0, 1 >$,
$\vec{a}^{\max_k}, \vec{a}^{\min_k}$ jsou konstantní vektory určující hranice jednotlivých částí každého vektoru,
$i = 1, \ldots, NP$ je index vektoru v populaci,
$k = 1, \ldots, D$ je parametr vektoru.

4.3 Reprodukční cyklus

Pro každého jedince v populaci je vygenerován zkušební vektor zk, který je vytvořen kombinací rodičovského vektoru a šumového vektoru ν. Poté je pro každou dimenzi generováno náhodné číslo r z intervalu $< 0, 1 >$. Je-li toto číslo menší než práh křížení *CR*, je zkušebnímu jedinci pro aktuální rozměr přiřazen parametr ze šumového vektoru, v opačném případě parametr rodiče, tj.:

$$zk_i^k(t+1) = \begin{cases} \nu_i^k(t+1) & \text{if } (r < CR) \\ P_i^k(t) & \text{else.} \end{cases} \tag{4.2}$$

Kde: $P_i^k(t)$ je rodičovský vektor aktuální generace v čase t,
$i = 1, \ldots, NP$,
$k = 1, \ldots, D$.

Po vytvoření zkušebního vektoru zk a jeho ohodnocení se tento vektor porovnává z rodičovským vektorem. Do nové generace, přechází kvalitnější z obou vektorů.

K výpočtu šumového vektoru je možno použít více, než jednoho diferenčního vektoru, který určuje velikost mutace. Jestliže n_v je počet použitých diferenčních vektorů a *NP* je velikost populace, potom celkový počet diferenčních odchylek je dán:

$$\binom{NP}{2n_v} 2n_v! \approx O(NP^{2n_v}) \tag{4.3}$$

4.3.1 Selekční schémata

Šumový vektor ν se vytváří podle některého ze selekčních schémat. Tyto schémata se označují jako *DE/x/y/z* a představují různé druhy výběru vektorů, počet použitých diferenčních vektorů a způsob křížení. v případě použití tohoto zápisu, x určuje způsob výběru cílového vektoru, y udává počet diferenčních vektorů a z určuje způsob křížení.

DE/rand/1 je základním selekčním schématem DE. Šumový vektor je počítán ze tří náhodně vybraných jedinců z aktuální populace v čase (t) dle vztahu:

$$\vec{\nu}_i(t+1) = \vec{x}_{r_1^i}(t) + \beta \cdot (\vec{x}_{r_2^i}(t) - \vec{x}_{r_3^i}(t)) \tag{4.4}$$

kde r_1^i, r_2^i, $r_3^i \in$ [0, NP-1], pro které platí $r_1^i \neq r_2^i \neq r_3^i$.

DE/rand/2 je modifikací základního schématu *DE/rand/1*, ovšem k výpočtu diferenčního vektoru se používá 4. náhodně vybraných jedinců. Šumový vektor $\vec{\nu}$ je počítán dle vztahu:

$$\vec{\nu}_i(t+1) = \vec{x}_{r_1^i}(t) + \beta \cdot (\vec{x}_{r_2^i}(t) + \vec{x}_{r_3^i}(t) - \vec{x}_{r_4^i}(t) - \vec{x}_{r_5^i}(t)) \tag{4.5}$$

kde $r_1^i, \ldots, r_5^i \in$ [0, NP-1], pro které platí $r_1^i \neq \ldots \neq r_5^i$,

DE/best/1 je podobné schématu *DE/rand/1*. Rozdílný je typ základního vektoru. Základním vektorem je v tomto případě $\vec{x}_{r_{best}}(t)$ namísto $\vec{x}_{r_1^i}(t)$, ke kterému bude přičten zesílený (mutovaný) rozdílový vektor. Šumový vektor $\bar{\nu}$ je generován podle vztahu:

$$\vec{\nu}_i(t) = \vec{x}_{r_{best}}(t) + \beta \cdot (\vec{x}_{r_1^i}(t) - \vec{x}_{r_2^i}(t)) \tag{4.6}$$

kde $\vec{x}_{r_{best}}(t)$ je nejlepší vektor v populaci (t),
r_1^i, $r_2^i \in [0,\ NP-1]$, pro které platí $r_1^i \neq r_2^i \neq r_{best}$.

DE/best/2 je modifikací schématu *DE/rand/2*. Základní vektor $\vec{x}_{r_1^i}(t)$ je opět nahrazen nejlepším vektorem v populaci $\vec{x}_{r_{best}}(t)$. Šumový vektor $\bar{\nu}$ je generován podle vztahu:

$$\vec{\nu}_i(t+1) = \vec{x}_{r_{best}}(t) + \beta \cdot (\vec{x}_{r_1^i}(t) + \vec{x}_{r_2^i}(t) - \vec{x}_{r_3^i}(t) - \vec{x}_{r_4^i}(t)) \tag{4.7}$$

kde $r_1^i, \ldots, r_4^i \in$ [0, NP-1], pro které platí $r_1^i \neq \ldots \neq r_4^i$.

Ve skutečnosti může být pro mutaci použito více lineárních kombinací vektorů. Obecný zápis je možno vyjádřit ve tvaru [9]:

$$\vec{\nu}_i(t+1) = \vec{y}_i(t) + \beta \cdot \frac{1}{N} \sum_{n=0}^{N-1} (\vec{x}_{r_{2n+1}}(t) - \vec{x}_{r_{2n+2}}(t)) \tag{4.8}$$

kde $\vec{y}_i(t)$ je základní vektor a měl by být jiný od ostatních vektorů v rovnici (4.8).

DE/x/$\mathbf{n_v}$ je schéma, ve kterém je užito více, než jednoho diferenčního vektoru. Počet diferenčních vektorů je určen n_v. s rostoucí hodnotou n_v se zvětšuje velikost prohledávaného prostoru v jedné generaci (4.3), viz. také [13]. Šumový vektor $\bar{\nu}$ je počítán podle vztahu:

$$\vec{\nu} = \vec{x}_{r_1^i}(t) + \beta \cdot \sum_{i=1}^{n_v} (\vec{x}_{r_2^i}(t) - \vec{x}_{r_3^i}(t)) \tag{4.9}$$

kde r_1, r_2, $r_3 \in [0,\ NP-1]$,
$\vec{x}_{r_1^i}(t)$ může být cíleně vybrán vhodnou selekční metodou, $r_2^i \neq r_3^i$.

DE/current-to-best/1 schéma [9] vytváří pro každého jedince v populaci jeden šumový vektor $\bar{\nu}$ dle vztahu:

$$\vec{\nu}_i(t+1) = \vec{x}_i(t) + \beta \cdot (\vec{x}_{r_{best}}(t) - \vec{x}_i(t)) + \beta \cdot (\vec{x}_{r_1^i}(t) - \vec{x}_{r_2^i}(t)) \tag{4.10}$$

kde $\vec{x}_{r_{best},G}$ je nejlepší jedinec v populaci, $\vec{x}_i(t)$ je aktuální jedinec populace, r_1, r_2, $i \in [0,\ NP-1]$ a platí $r_1 \neq r_2$.

DE/rand-to-best/n$_\mathbf{v}$ schéma kombinuje k výpočtu šumového vektoru $\vec{\nu}$ strategie *rand* a *best* [9]:

$$\vec{\nu}_i(t+1) = \gamma \cdot \vec{x}_{r_{best}}(t) + (1-\gamma) \cdot \vec{x}_{r_1^i}(t) + \beta \sum_{i=1}^{n_v} (\vec{x}_{r_2^i}(t) - \vec{x}_{r_3^i}(t)) \tag{4.11}$$

kde $\vec{x}_{r_{best}}(t)$ je nejlepší jedinec v populaci,
$\vec{x}_{r_1^i}(t)$ je náhodně vybraný jedinec z aktuální populace,
parametr $\gamma \in < 0, 1 >$ určuje rychlost konvergence v poměru k prohledávanému prostoru.

V případě, že se hodnota γ blíží k 1, rychlost konvergence se zvyšuje a naopak velikost prohledávaného prostoru klesá. Vzhledem k této vlastnosti je dobrou strategií adaptivní změna hodnoty γ, pro kterou platí, že $\gamma(0) = 0$ a s každou další vytvořenou generací se blíží k 1. Jestliže $\gamma = 0$, pak schéma odpovídá strategii *DE/rand/y/z*. Naopak, jestliže $\gamma = 1$, pak *DE/best/y/z*.

DE/current-to-best/1 + n$_\mathbf{v}$ vytváří pro každého jedince v populaci jeden šumový vektor $\bar{\nu}$. K výpočtu $\vec{\nu}$ se používá minimálně dvou diferenčních vektorů. Jeden diferenční vektor je počítán z nejlepšího a aktuálního vektoru a zbylé diferenční vektory jsou počítány z náhodně zvolených jedinců z aktuální populace:

$$\vec{\nu}_i(t+1) = \vec{x}_i(t) + \beta(\vec{x}_{r_{best}}(t) - \vec{x}_i(t)) + \beta \sum_{i=1}^{n_v} (\vec{x}_{r_1^i}(t) - \vec{x}_{r_2^i}(t)) \tag{4.12}$$

Empirické studie ukazují, že strategie *DE/rand/1* dosahuje velké rozmanitosti populace. Na druhou stranu *DE/current-to-best/2* vykazuje rychlejší konvergenci. v důsledku těchto studií vznikl algoritmus, který umožňuje dynamické přepínání mezi těmito strategiemi, tzv. auto-adaptivní diferenciální evoluce – *self-adaptive DE - (SaDE)*

4.4 Auto–adaptivní diferenciální evoluce - SaDE

Auto-adaptivní DE *(Self–adaptive Differential Evolution)* (SaDE) [46, 37] kombinuje 2 odlišné způsoby mutace v jedné variantě diferenciální evoluce. Tyto mutace jsou voleny v závislosti na parametru p, který značí pravděpodobnost vybrání dané metody mutace a je postupně adaptován v závislosti na vývoji evoluce.

K výpočtu šumového vektoru se využívají schémata 4.4 a 4.10. Výpočet $\vec{\nu}$ se provádí podle:

$$\vec{\nu} = \begin{cases} Eq.(4.4), & \text{if } (U_i(0,1) < p) \\ Eq.(4.10), & else. \end{cases} \tag{4.13}$$

Řídící parametr výběru schématu p je na počátku nastaven na hodnotu 0.5 a v průběhu evoluce se adaptuje v cyklech o délce adaptační periody, která je v případě *SaDE* 50 cyklů.:

$$p = \frac{ns_1 \cdot (ns_2 + nf_2)}{ns_2 \cdot (ns_1 + nf_1) + ns_1 \cdot (ns_2 + nf_2)} \tag{4.14}$$

kde ns_1, ns_2 je počet jedinců vygenerovaných rovnicí 4.4, resp. 4.10, kteří prošli do další generace; nf_1, nf_2 je počet zahozených jedinců, během generování rovnicí 4.4, resp. 4.10. Proměnné ns_1, ns_2, nf_1, nf_2 jsou resetovány vždy po nastavení p.

Faktor zesílení β je v *SaDE* nastaven podle:

$$\beta_i = N_i(0.5, 0.3) \tag{4.15}$$

kde $N_i(0.5, 0.3)$ je náhodně generované číslo s Gaussovým rozložením se střední hodnotou 0.5 a standardní odchylkou 0.3.

Auto-adaptace *CR* (SaCR): *SaDE* alokuje proměnnou CR_i pro každého jedince v generaci podle:

$$CR_i = N_i(CRm, 0.1) \tag{4.16}$$

Na počátku evoluce je *CRm* nastaven na hodnotu 0.5. Tyto hodnoty se v průběhu několika prvních generací (5 generací v SaDE) nemění. Poté je nová hodnota generována použitím stejné rovnice 4.16. Během každé generace, jsou hodnoty CR_i, jenž jsou asociované s potomky, kteří úspěšně vstoupili do nové generace uloženy do pole CR_{rec}. Po specifickém počtu generací (25 generací v SaDE), je *CRm* updatováno podle:

$$CRm = \frac{1}{|CR_{rec}|} \sum_{k=1}^{|CR_{rec}|} CR_{rec}(k) \tag{4.17}$$

Proměnná CR_{rec} je resetována pokaždé, když je updatována proměnná *CRm*.

Další z auto–adaptivních modelů DE je popsán v [46] - *SaNSDE - Self-Adaptive Differential Evolution with Neighborhood Search.* Tento model dále vylepšuje možnosti *SaDE.*

5 Jazyky a gramatika

Gramatika G je soustava pravidel, podle kterých se vytvářejí všechny řetězce **jazyka** L definovaného gramatikou. Jedním ze základních pojmů pro vymezení jazyka jsou pojmy *abeceda* a *řetězec* [7]. **Abecedou** Σ rozumíme neprázdnou množinu prvků, které nazýváme **symboly abecedy**. **Řetězcem** nad danou abecedou rozumíme každou konečnou posloupnost symbolů abecedy. Prázdnou posloupnost symbolů, tj. posloupnost, která neobsahuje žádný symbol, nazýváme *prázdný řetězec* ϵ. Formálně lze definovat řetězec nad abecedou Σ takto:

- prázdný řetězec ϵ je řetězec nad abecedou Σ,
- je-li x řetězec nad Σ a $a \in \Sigma$, pak xa je řetězec nad Σ
- y je řetězec nad Σ, když a jen když lze y získat aplikací pravidel 1 a 2.

Symbol Σ^* označuje všechny řetězce nad abecedou Σ včetně prázdného řetězce ϵ, symbol Σ^+ značí množinu všech řetězců nad Σ bez prázdného symbolu ϵ, tj. $\Sigma^* = \Sigma^+ \cup \{\epsilon\}$. Množina L, pro níž platí $L \subseteq \Sigma^*$ se nazývá *jazyk* L nad abecedou Σ. **Jazykem** tedy může být libovolná podmnožina řetězců nad danou abecedou.

Gramatiky byly původně navrženy pro popis přirozených jazyků, ale běžně se používají pro popis formálních jazyků. Podle způsobu použití ji pak můžeme rozdělit na *generativní gramatiku* a *analytickou gramatiku.*

Generativní gramatika: představuje generativní systém, ve kterém lze aplikací přepisovacích pravidel generovat řetězce tvořené pouze terminálními symboly, které reprezentují věty jazyka definovaného touto gramatikou.

Je popsána uspořádnou čtveřicí symbolů $G = (N, \Sigma, P, S)$, kde N je množina neterminálních symbolů, Σ je množina terminálních symbolů, $S \in N$ je počáteční symbol gramatiky. Jádrem gramatiky je konečná množina P *přepisovacích (produkčních) pravidel*, kde každé přepisovací pravidlo má tvar uspořádané dvojice (α, β) řetězců a stanovuje možnou substituci řetězce β namísto řetězce α, který se vyskytuje jako podřetězec generovaného řetězce. Řetězec α obsahuje alespoň jeden neterminální symbol, řetězec β je prvek množiny $(N \cup \Sigma)^*$. Přepisovací pravidlo se značí $\alpha \rightarrow \beta$. Množinu přepisovacích pravidel P je možno vyjádřit jako:

$$P \subseteq (N \cup \Sigma)^* N (N \cup \Sigma)^* \times (N \cup \Sigma)^* \tag{5.1}$$

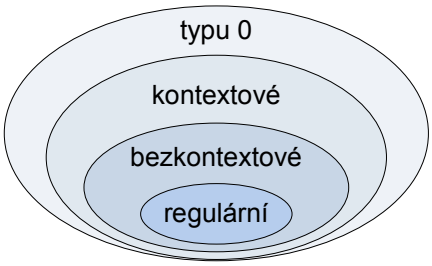
Analytická gramatika: je gramatika, kterou je možno použít k rozpoznání syntaktické správnosti programu. Je popsána čtveřicí $G^{-1} = (N, \Sigma, P^{-1}, S)$, kde N je množina neterminálních symbolů, Σ je množina terminálních symbolů, $S \in N$ je počáteční symbol gramatiky. Jádrem gramatiky je konečná množina P^{-1} pravidel, která je inverzní množinou k P.

Terminály jsou postupně nahrazovány řetězci terminálních a neterminálních symbolů, s cílem získat startovní symbol S gramatiky G. Formálně se zapisuje ve tvaru $\beta \rightarrow \alpha$, kde řetězec α obsahuje alespoň jeden neterminální symbol, řetězec β je prvek množiny $(N \cup \Sigma)^*$.

O tom zda daný řetězec patří, resp. nepatří do jazyka popsaného danou gramatikou rozhoduje, zdali se podaří získat startovní symbol, či nikoli.

5.1 Chomského hierarchie gramatik a jazyků

Lingvista Noam Chomsky rozdělil jazyky do čtyř skupin (typů), viz. obrázek 19, na základě různých omezení tvaru pravidel na [7, 8]:



Obrázek 19: Hierarchie jazyků

Gramatika typu 0: Je obecná, rekurzivně vyčíslitelná gramatika. Tato gramatika může být rozpoznaná Touringovým strojem. Tvar pravidel nemá žádné zvláštní omezení:

$$\alpha \rightarrow \beta,\ \alpha \in (N \cup \Sigma)^* N (N \cup \Sigma)^*,\ \beta \in (N \cup \Sigma)^*$$

Gramatika typu 1: Kontextová gramatika (*Context-Sensitive – CSG*) je gramatika rozpoznatelná lineárně ohraničeným Turingovým strojem. Tato gramatika obsahuje pravidla ve tvaru:

$$\alpha A \beta \rightarrow \alpha \gamma \beta,\ A \in N,\ \alpha,\ \beta \in (N \cup \Sigma)^*,\ \gamma \in (N \cup \Sigma)^+ \mid S \rightarrow \epsilon$$

Gramatiky typu 1 se nazývají kontextové, jelikož neterminál A může být nahrazen řetězcem γ pouze tehdy, je-li jeho pravým kontextem řetězec β a levým kontextem řetězec α.

Kontextové gramatiky neobsahují pravidla tvaru $\alpha A \beta \rightarrow \alpha \beta$, tj. nepřipouštějí, aby byl neterminál nahrazen prázdným řetězcem. Jedinou přípustnou výjimkou je pravidlo $S \rightarrow \epsilon$. V důsledku tohoto nemůže dojít ke zkrácení generovaných řetězců, tj. platí-li $\lambda \Rightarrow \mu$, pak $|\lambda| \leq |\mu|$ (s výjimkou $S \Rightarrow \epsilon$)

Gramatika typu 2: Bezkontextová gramatika (*Context-Free – CFG*) je rozpoznatelná zásobníkovým automatem. Obsahuje pravidla ve tvaru:

$$A \rightarrow \gamma,\ A \in N,\ \alpha \in (N \cup \Sigma)^*$$

Gramatiky typu 2 se nazývají kontextové, jelikož neterminál A může být nahrazen řetězcem γ bez ohledu na kontext, ve kterém je uložen neterminál A. Na rozdíl od kontextových gramatik, smí bezkontextové gramatiky obsahovat i pravidla tvaru $A \rightarrow \epsilon$.

Gramatika typu 3: Pravá lineární gramatika je rozpoznatelná konečným automatem a obsahuje pravidla ve tvaru:

$$A \rightarrow xB \mid A \rightarrow x;\ A, B \in N,\ x \in \Sigma^*$$

Gramatiky typu 3 se nazývají pravé lineární, jelikož jediný možný neterminál pravé strany pravidla stojí úplně napravo. K této gramatice lze sestrojit ekvivalentní speciální pravou lineární gramatiku s pravidly ve tvaru:

$$A \rightarrow aB \mid A \rightarrow a;\ A, B \in N,\ x \in \Sigma \mid S \rightarrow \epsilon$$

Tato gramatika se nazývá pravá regulární gramatika. Gramatiky typu 3 se proto také nazývají *regulárními gramatikami.*

5.2 BNF gramatiky

Backusova–Naurova forma (BNF) *(Backus–Naur form)* je metasyntaxe používaná k vyjádření bezkontextové gramatiky, která se používá pro popis formálních jazyků. Vytvořil ji původně John Warner Backus a dále ji zjednodušil Peter Naur pro popis programovacího jazyka *Algol 60.*
Produkční pravidla v BNF jsou vyjádřeny ve tvaru:

$$<\text{symbol}>::= \text{expression}$$

kde $<\text{symbol}>$ je neterminál, tj $<\text{symbol}>\in N$ a je vždy uzavřen v závorkách $< \ldots >$, *expression* sestává z jedné, nebo více sekvencí terminálních a neterminálních symbolů oddělených |, které jsou možnou substitucí neterminálu na levé straně pravidla.

Příklad část definice pravidel pro zpracování čísel v Algol 60 [41]:

```
<number>            ::= <unsigned number> | + <unsigned number > |
                        - <unsigned number >
<unsigned number>   ::= <decimal number> | <exponent part> |
                        <decimal number><exponent part>
<decimal number>    ::= <unsigned integer> | <decimal fraction> |
                        <unsigned integer><decimal fraction>
<unsigned integer>  ::= <digit> | <unsigned integer><digit>
<digit>             ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<decimal fraction>  ::= .<unsigned integer>
<exponent part>     ::= #<integer>
<integer>           ::= <unsigned integer> | + <unsigned integer> |
                        - <unsigned integer>
```

BNF gramatiky se dočkaly mnoha rozšíření, které vznikly z důvodu dosažení větší jednoduchosti nebo stručnosti, popřípadě za účelem adaptace BNF pro specifické účely. Jedním společným znakem řady variant BNF je použití opakovacích operátorů z regulárních výrazů, např. $*$ a $+$. Jedním z rozšíření je tzv. *Rozvinutá Backusova-Naurova forma (Extended Backus–Naur form, EBNF)*, původně navržená Niklausem E. Wirthem k vyjádření bezkontextové gramatiky. Dnes je většina proměnných EBNF standardizována [47] a definována normami, zejm. ISO-14977 pod kódovým označením ISO/IEC 14977:1996(E).

V tabulce 4 je přehled některých používaných meta-symbolů v BNF a EBNF.

symbol	význam	BNF	EBNF
::=	definice	•	
=	definice		•
|	nebo	•	•
< ... >	neterminál	•	
" ... "	víceznakový terminál		•
' ... '	jednoznakový terminál		•
,	zřetězení		•
(...)	skupina		•
[...]	označuje skupinu, která může být obsažena nanejvýš jednou		•
{...}	označuje skupinu, která se může opakovat (0, ... n-krát)		•
*	stejný význam jako {...}		•
+	označuje, že se něco může opakovat jednou, nebo víckrát		•
?	stejný význam jako [...]		•
(* ... *)	komentář		•
@	prázdný řetězec		•

Tabulka 4: Symboly používané k vyjádření BNF, EBNF

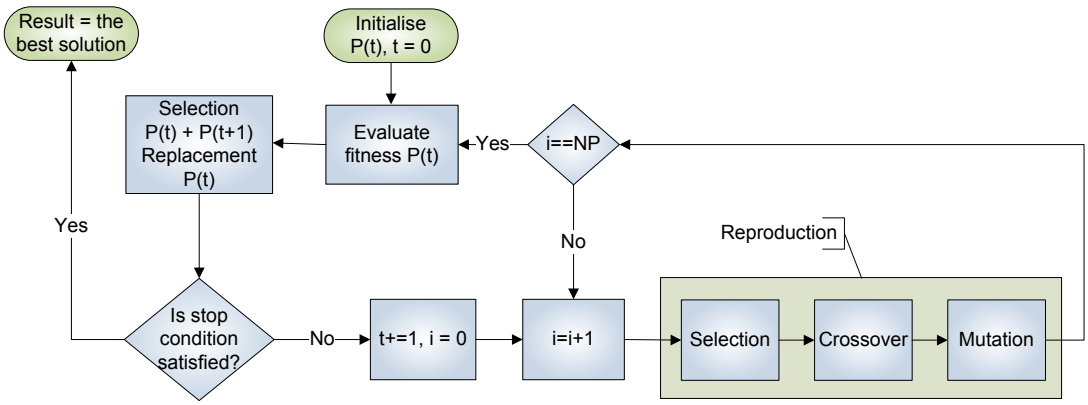
6 Gramatická evoluce

Gramatická evoluce (GE) je populačně založená evoluční metoda, která vznikla kombinací genetického algoritmu a gramatiky [39, 31, 30, 28, 35, 44]. Gramatika slouží k popisu jazyka, jenž může být popsán právě touto gramatikou, (viz. kapitola 5). Za představitele GE jsou považování M. O'Neill, J. J. Collins a C. Ryan, kteří představili ji v roce 1998, [38].

Mezi unikátní vlastnosti gramatické evoluce patří především její schopnost vytvářet symbolický model řešené úlohy a následně jej vyjádřit v jazyce daném zvolenou gramatikou. Gramatická evoluce tedy kromě řešení poskytuje i vlastní postup, jak řešení vytvořit. Nevýhodou GE je její vysoká výpočetní náročnost.

Genetický algoritmus se v gramatické evoluci nepoužívá přímo pro hledání konkrétních řešení, ale k hledání vhodného odvození řešení pomocí regulární, nebo bezkontextové gramatiky. Obecnost metody zvyšuje nezávislost GA na použité gramatice, kterou je možno nahradit za jinou, aniž by bylo nutno programovat GA.

Populace ve standardní gramatické evoluci je složena z jedinců z nichž každý představuje jedno řešení. V průběhu evoluce je populace postupně zkvalitňována nahrazováním nekvalitních jedinců kvalitnějšími. Noví jedinci se vytvářejí použitím reprodukčních operátorů křížení a mutace, (viz. kapitola 6.2). Evoluce probíhá tak dlouho, dokud není splněna podmínka zastavení. Evoluce je obvykle zastavena po určitém počtu generací, nebo při dosažení určité hodnoty fitness. Základní vývojový diagram GE je na obrázku 20.



Obrázek 20: Vývojový diagram gramatické evoluce

Každý jedinec GE obsahuje chromozóm, který je reprezentován celočíselným, nebo binárním řetězcem. V případě, že je použito binární kódování jsou jednotlivé bity uloženy v genech na bitové hladině. Jednotlivé **geny** jsou sloučeny do skupin pevné délky. Skupina genů je nazývána **kodon** a představuje celočíselnou hodnotu, která je použita v překladu genotypu na fenotyp, (viz. kapitola 6.1). Chromozóm gramatické evoluce je ukázán na obrázku 21.

codons-level	10				14				5				10				3				1				11				12				8				12				13				9			
bits-level	1	0	1	0	1	1	1	0	0	1	0	1	1	0	1	0	0	0	1	1	0	0	0	1	1	0	1	1	1	1	0	0	1	0	0	0	1	1	0	0	1	1	0	1	1	0	0	1

Obrázek 21: Chromozóm v gramatické evoluci

6.1 Překlad genotypu na fenotyp

Překlad genotypu na fenotyp *(genotype to phenotype mapping)* je klíčovou částí gramatické evoluce, která ji odlišuje od genetických algoritmů. Překlad spočívá v postupném výběru neterminálů z těla jedince a jejich přepisování na jiné neterminální a terminální symboly pomocí pravidel dané gramatiky, viz. algoritmus 3.

Algoritmus 3: Překlad genotypu na fenotyp

```
   Input: G – context-free grammar; N – nonterminal; C – codons
   Result: Phenotype
 1 P ← array of rules for the nonterminal N in G indexed from zero;
 2 n ← length[P];
 3 if n = 1 then
 4 |   r ← 0;
 5 else if length(C) > 0 then
 6 |   r ← Pop(C) mod n ;                                  // celočíselné dělení
 7 else
 8 |   if if wrap allowed then
 9 |   |   r ← Wrap ;                        // Nastaví ukazatel na počátek chromozómu
10 |   else
11 |   |   throw error "Out of codons";
12 |   end
13 end
14 s ← string of symbols at the right-hand side of the rule P[r] ;     // Pravá strana pravidla
15 t ← λ;
16 foreach A ← s do
17 |   if A is terminal then
18 |   |   t ← t join A;
19 |   else
20 |   |   tt ← GenotypeToPhenotypeMapping(G, A, C) ;                        // rekurze
21 |   |   t ← t join tt;
22 |   end
23 end
24 return t;
```

Vstupem v tomto algoritmu je gramatika G v Backus-Naurově-Formě (BNF), (viz. kapitola 5.2), neterminální symbol N (na počátku nastavený na startovní symbol S gramatiky G) a chromozóm jedince C. V prvním kroku je přečten neterminální symbol N a do proměnné P jsou vloženy pravidla možných přepisů tohoto symbolu podle gramamtiky G. Těchto pravidel je obvykle více a je třeba rozhodnout, které pravidlo bude použito. Výběr pravidla, které bude použito k přepisu neterminálního symbolu se provádí pomocí aritmetické operace celočíselného dělení **(modulo)**, která transformuje větší rozsah hodnoty kodonu na menší rozsah pravidel. Vybrané pravidlo je použito k přepisu neterminálu. Tento přepis je proveden v cyklu, který prochází jednotlivé podřetězce tohoto pravidla a v případě, že narazí na další neterminální symbol, je rekurzivně zavolána metoda pro překlad, do které je jako parametr vložen tento neterminální symbol. Cyklus se opakuje do doby než je jedinec složen ze samých neterminálních symbolů.

Nevýhodou tohoto přístupu je neexistence zpětné vazby fenotypu na genotyp, což se negativně projeví v případě křížení a mutace. Další nevýhodou je možné nevyužití všech kodonů v chromozómu jedince, nebo naopak jejich nedostatek při transformaci genotypu

na fenotyp. Problém nedostatku kodonů v chromozómu jedince řeší operátor wrap, (viz. kapitola 6.2.3).

Příklad: Princip překladu genotypu na fenotyp je demonstrován na úloze vygenerování rekurentní rovnice regulátoru ve tvaru $U_{k-1} + 2 \cdot E_k + 5 \cdot E_{k-1}$. K překladu budou použity pravidla bezkontextové gramatiky v Backus-Naurově formě, (viz. kapitola 5.2), definované v tabulce 5. Postup překladu je znázorněn na obrázku 22.

$\langle S \rangle$	$::=$	$\langle exp \rangle$
$\langle exp \rangle$	$::=$	$\langle fnc1 \rangle \langle exp \rangle \mid \langle fnc2 \rangle \langle exp \rangle \langle exp \rangle \mid \langle var \rangle \mid \langle const \rangle$
$\langle fnc1 \rangle$	$::=$	$U+ \mid U-$
$\langle fnc2 \rangle$	$::=$	$+ \mid - \mid * \mid /$
$\langle var \rangle$	$::=$	$U_{k-1} \mid U_{k-2} \mid U_{k-3} \mid U_{k-4} \mid U_{k-5} \mid$ $E_k \mid E_{k-1} \mid E_{k-2} \mid E_{k-3} \mid E_{k-4} \mid E_{k-5} \mid$ $dE_k \mid dE_{k-1} \mid dE_{k-2} \mid dE_{k-3} \mid dE_{k-4}$
$\langle const \rangle$	$::=$	$1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \mid 0$

Tabulka 5: Pravidla gramatiky v prefixovém BNF tvaru

A	B	C
10	mod 1 = 0	**<exp>**
33	mod 4 = 1	**<fnc2> (<exp>, <exp>)**
8	mod 4 = 0	+ (<exp>, <exp>)
18	mod 4 = 2	+ (**<var>**, <exp>)
64	mod 16 = 0	+ (**U_{k-1}**, <exp>)
5	mod 4 = 1	+ (U_{k-1}, **<fnc2>(<exp>, <exp>)**)
4	mod 4 = 0	+ (U_{k-1}, **+**(<exp>, <exp>))
17	mod 4 = 1	+ (U_{k-1}, +(**<fnc2>(<exp>, <exp>)**, <exp>))
34	mod 4 = 2	+ (U_{k-1}, +(*****(<exp>, <exp>), <exp>))
35	mod 4 = 3	+ (U_{k-1}, +(*****(**<const>**, <exp>), <exp>))
51	mod 10 = 1	+ (U_{k-1}, +(*****(**2**, <exp>), <exp>))
14	mod 4 = 2	+ (U_{k-1}, +(*****(2, **<var>**), <exp>))
37	mod 16 = 5	+ (U_{k-1}, +(*****(2, **E_k**), <exp>))
9	mod 4 = 1	+ (U_{k-1}, +(*****(2, E_k), **<fnc2>(<exp>, <exp>)**))
10	mod 4 = 2	+ (U_{k-1}, +(*****(2, E_k), *****(<exp>, <exp>)))
15	mod 4 = 3	+ (U_{k-1}, +(*****(2, E_k), *****(**<const>**, <exp>)))
4	mod 10 = 4	+ (U_{k-1}, +(*****(2, E_k), *****(**5**, <exp>)))
6	mod 4 = 2	+ (U_{k-1}, +(*****(2, E_k), *****(5, **<var>**)))
38	mod 16 = 6	+ (U_{k-1}, +(*****(2, E_k), *****(5, **E_{k-1}**)))

sloupec A: chromozóm, **sloupec B**: Výběr pravidla, **sloupec C**: Stav překladu. Červené symboly značí neterminály, které budou v dalším kroku nahrazeny, tučně jsou právě nahrazené

Obrázek 22: Překlad genotypu na fenotyp

Ve sloupci A je řetězec chromozómu, sloupec B značí výběr pravidla a sloupec C stav překladu. Jak je vidět, tak k vytvoření rovnice je potřeba 19-ti kodonů. Kodony se po-

užívají pro výběr pravidla z množiny pravidel, kterými je možno přepsat neterminální symbol. Tento výběr se provádí pomocí operace modulo. Důsledkem použití operace modulo je možnost změnit hodnotu genu, (resp. kodonu), bez ovlivnění fenotypu (10 mod 3 $\Leftrightarrow$ 16 mod 3). Tzn. že v případě mutace, popř. křížení nemusí dojít ke změně fenotypu i když se změní genotyp jedince. Tuto vlastnost je možno částečně eliminovat zvýšením pravděpodobnosti mutace s předpokladem, že určité procento mutovaných jedinců se nezmění. Změnu genu, která nezpůsobí změnu fenotypu je vyjádřena v rovnici (6.1).

$$a_i = a_i \pm c \cdot Q \tag{6.1}$$

Kde: a_i je hodnota i-tého kodonu jedince,
$c \in \mathbb{N}$,
Q je počet pravidel, kterými je možno přepsat daný neterminál,
platí pro $a_i \pm c \cdot Q \geq 0$.

Gramatická evoluce může také sloužit ke generování hodnot z oboru reálných čísel. To je možno realizovat pomocí gramatiky definované v kapitole 5.2, případně 6.7, nebo gramatikou definovanou v tabulce 6 [12].

⟨*expressions*⟩	::=	⟨*expressions*⟩⟨*op*⟩⟨*numExp*⟩ | ⟨*exp*⟩
⟨*exp*⟩	::=	MA(⟨*numExp*⟩) | ⟨*numExp*⟩
⟨*numExp*⟩	::=	⟨*int*⟩⟨*op*⟩⟨*int*⟩ | ⟨*int*⟩
⟨*op*⟩	::=	+ | - | * | /
⟨*digit*⟩	::=	1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0

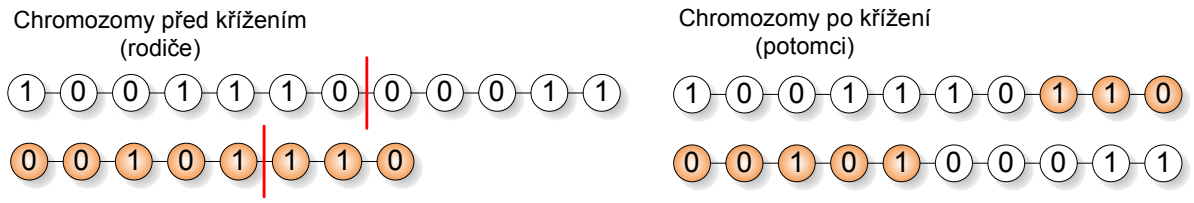
Tabulka 6: Pravidla použitá pro zpracování reálných čísel [12]

6.2 Operátory

Ve standardním algoritmu gramatické evoluce jsou definovány operátory křížení, mutace, wrap a také operátor zdvojování.

6.2.1 Křížení

Křížení *(crossover - sexual reproduction)* v GE je realizováno nad genotypem jedince na úrovní genů, nebo kodonů. V případě, že daný typ GE nepoužívá fixní délku chromozómů jedinců, je v obou jedincích náhodně vybrán bod křížení a jednotlivé segmenty se vzájemně vymění. Tento postup je znázorněn na obrázku 23. Křížicí body obou jedinců



Obrázek 23: Křížení jedinců s variabilní délkou chromozómu

jsou vyznačeny červenou čárou v levé části obrázku. Jak je vidět, tak tento druh křížení způsobuje změnu délky chromozómu.

Jestliže typ GE používá fixní délku chromozómu, je vybrán bod křížení a části jedinců se vzájemně vymění ve stejných bodech. Tento postup je ekvivalentní postupu popsaném v 3.8.2. Při použití tohoto typu křížení se délka chromozómů nemění.

Oba zmíněné typy křížení mají poměrně destruktivní vliv na strukturu jedince. Z toho důvodu je vhodné použít zpětné zpracování pravidel se značkováním podskupin, které umožní křížit chromozómy bez destrukce struktury jedince, (viz. kapitola 6.3).

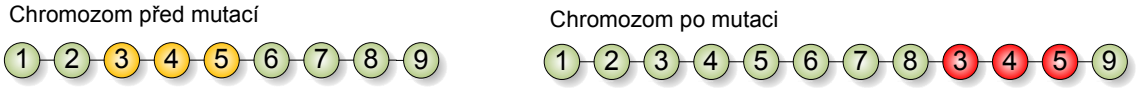
6.2.2 Mutace

Operátor mutace je stejně jako křížení realizováno nad genotypem jedince. Mutaci je možno realizovat na úrovni genů, nebo kodonů.

Mutace na úrovni genů *(bits-level mutation)* je bitová mutace a může být provedena sekvenčním procházením chromozómu jedince a invertováním hodnoty bitů se specifikovanou pravděpodobností. Efektivní rychlost mutace je ovlivněna degenerací populace. Rychlost degenerace populace je ovlivněná počtem bitů v kodonu, délkou chromozómu, strukturou použitých pravidel gramatiky a použitím *operátoru wrap*. Nemá smysl srovnávat hodnoty pravděpodobnosti mutace mezi GE a GP, nebo mezi různými nastaveními GE. Tento typ mutace je v evolučních algoritmech jedním z nejčastějších [14, 28]. Princip inverzní mutace ukázán je na obrázku 10.

Mutace na úrovni kodonů *(codons-level mutation)* je realizována sekvenčním procházením jednotlivých kodonů, které s určitou pravděpodobností nastavuje na náhodně vygenerovanou hodnotu. Výhoda oproti mutaci na úrovni genů spočívá v menší časové složitosti výpočtu. Metoda vyžaduje pro každý kodon nanejvýš dvě volání generátoru náhodných čísel (jedno volání rozhodne o provedení mutace a druhé volání vygeneruje novou hodnotu kodonu). Tento typ mutace byl aplikován v GEVA[6].

Zdvojování *(duplication)* je zvláštním případem mutace, která není příliš používána [30, 28]. Z chromozómu je náhodně zvolen počet kodonů ke zdvojení a pozice prvního kodonu. Tato množina je zkopírována před poslední kodon v chromozómu jedince. Důvod začlenění na předposlední pozici jedince je v usnadnění začlenění do fenotypu. Jestliže by duplikovaná množina kodonů byla přidána za poslední kodon chromozómu, nemuselo by dojít k jejich začlenění do fenotypu. Přínos tohoto operátoru je sporný. Operace zdvojování je ukázána na obrázku 24. Pro zdvojení byly vybrány 3 kodony jejichž startovní pozice začíná kodonem 3.

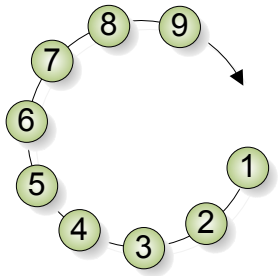


Obrázek 24: Zdvojování

[6]GEVA (Grammatical Evolution in Java) je open source implementace gramatické evoluce, která je vyvíjena na univerzitě v Dublinu [31]

6.2.3 Operátor wrap

Operátor wrap se uplatní v případě GE, která má definovanou omezenou délku chromozómů. Tento operátor má za cíl kompenzovat nedostatečnou délku chromozómu v případech transformace genotypu na fenotyp (viz. kapitola 6.1), kdy je dosažen konec chromozómu, ale fenotyp dosud nebyl plně vygenerován. V případech, kdy je délka chromozómu omezena, může k tomuto jevu docházet docela běžně. Když v průběhu překladu dojde k tomu, že je přečtena hodnota posledního kodonu, operátor wrap nastaví ukazatel opět na počátek chromozómu [30, 35]. To způsobí vícenásobné čtení genů, viz. obrázek 25. Význam tohoto operátoru je sporný a pro použití v upravené metodě překladu genotypu zpětným zpracováním pravidel je nepoužitelný, (viz. kapitola 6.3).



Obrázek 25: Operátor wrap

6.3 Zpětné zpracování pravidel a značkování kodonů

Vzhledem k dříve zmíněným nevýhodným vlastnostem originálně navržených operátorů křížení a mutace byla vyvinuta metoda zpětného zpracování pravidel gramatiky [35, 44]. Cílem této modifikace je zdokonalení křížení a mutace, které nepoškozuje strukturu jedince. Mezi nejdůležitější vlastnosti této metody patří:

- úplné využití chromozómu jedince
- první generování chromozómu během překladu genotypu na fenotyp
- zpětné zpracování pravidel gramatiky
- značkování kodonů během procesu překladu z genotypu na fenotyp
- použití upravených metod křížení a mutace

Překlad genotypu na fenotyp je ukázán v algoritmu 4. Tento algoritmus zpracovává pravidla gramatiky zprava, díky čemuž dojde k jistému seskupení genů, které vymezuje určitou část vygenerované struktury. Každou část této struktury je možno označit a toto značení využít v případě křížení a mutace. Aby mohl být splněn tento požadavek, je nutné dodržet jistý tvar pravidel gramatiky. Pravidla musí být v prefixové notaci a první symbol pravé strany pravidla musí být neterminál pro jehož překlad může být použito pouze pravidel, ze kterých je možno generovat pouze terminály (T–pravidla)(Eq. 6.2). To přináší další požadavek na pravidla, která jsou rozdělena na **T–pravidla** a **N–pravidla**. T–pravidla mohou na pravé straně obsahovat pouze neterminální symboly a N–pravidla jsou pravidla

Algoritmus 4: Překlad genotypu na fenotyp - backward processing

Input: G – context-free grammar; N – nonterminal
Result: Individual (phenotype and marked genotype)

1 $P \leftarrow$ array of rules for the nonterminal N in G indexed from zero;
2 $n \leftarrow$ length$[P]$;
3 $codon \leftarrow$ random $\in (0, \text{MaxInt})$; `// náhodné vygenerování kodonu`
4 **if** $n = 1$ **then**
5 | $r \leftarrow 0$;
6 **else**
7 | $r \leftarrow codon$ mod n ; `// celočíselné dělení`
8 **end**
9 $s \leftarrow$ string of symbols at the right-hand side of the rule $P[r]$;
10 $t \leftarrow \lambda$;
11 **foreach** $A \leftarrow$ reverse(s) **do**
12 | **if** A *is terminal* **then**
13 | | $t \leftarrow t$ join A;
14 | | $codon.mark \leftarrow E$; `// označení kodonu - koncový`
15 | | $C \leftarrow C$ join $codon$; `// připojení kodonu k chromozómu`
16 | **else**
17 | | $codon.mark \leftarrow B$; `// označení kodonu - počáteční`
18 | | $C \leftarrow C$ join $codon$; `// připojení kodonu k chromozómu`
19 | | $tt \leftarrow$ GenotypeToPhenotypeMapping(G, A, C) ; `// rekurze`
20 | | $t \leftarrow t$ join tt;
21 | **end**
22 | **if** $C.length > maxLength$ **then**
23 | | Do some action ; `// vyvolat nějakou akci, když délka chromozómu je větší, než maximální přednastavená`
24 | **end**
25 **end**
26 **return** t, C;

na jejichž pravé straně se nachází alespoň jeden neterminální symbol.

$$\langle exp \rangle ::= \langle fnc \rangle \, \langle exp \rangle \, \langle exp \rangle \tag{6.2}$$

Původní gramatika navržena v infixové notaci [30], (Eq. 6.3) je pro algoritmus zpětného zpracování pravidel a označování kodonů nepoužitelná.

$$\langle exp \rangle ::= \langle exp \rangle \, \langle op \rangle \, \langle exp \rangle \tag{6.3}$$

Postup procesu zpětného zpracování pravidel a označování genů je ukázán v algoritmu 4. Vstupem tohoto algoritmu je BNF gramatika v žádaném tvaru (prefixová notace, N–pravidla, T–pravidla) a neterminální symbol, (resp. symboly). Vzhledem k tomu, že chromozóm jedince není zatím vytvořen, je nutné jen generovat během překladu. Generování kodonu je provedeno na *řádku 3* v uvedeném algoritmu a sice jako vygenerování náhodné hodnoty z intervalu $\langle 0, MaxInt \rangle$. Poté se za pomoci operace modulo vybere pravidlo z množiny pravidel P. Pravá strana tohoto pravidla se sekvenčně prochází v obráceném pořadí *(řádek 12)* a každý symbol je zkontrolován zdali patří do množiny terminálů, nebo neterminálů. Jestliže kontrolovaný symbol je terminálem, označí se kodon značkou E, v opačném případě značkou B.

Od původního algoritmu, viz. algoritmus 3, se liší zejména generováním chromozómu současně s překladem těla jedince a značkováním kodonů. Překlad genotypu na fenotyp se ukončí ve chvíli, kdy tělo jedince bude složeno ze samých terminálních symbolů. Uvedený algoritmus neobsahuje ošetření generování nekonečně dlouhého jedince, které je možno řešit různými způsoby (například nové generování jedince). Výhodnější chování v tomto směru poskytuje algoritmus transplantační evoluce, diskutovaný v kapitole 7, jelikož umožnuje dynamickou změnu pravidel během evolučního procesu a tím i ovlivnění výsledného tvaru jedince, (viz. kapitola 7.4).

Na obrázku 26 je ukázán postup překladu pravidel a způsob značkování genů. Tento příklad předpokládá použití pravidel definovaných v tabulce 5. Ve *sloupci A* jsou náhodně,

A	B			C	D	E
33	mod 4	=	1	**<fnc2> (<exp>, <exp>)**	B	
9	mod 4	=	1	<fnc2> (<exp>, **<fnc2> (<exp>, <exp>)**)	B	
17	mod 4	=	1	<fnc2> (<exp>, <fnc2> (<exp>, **<fnc2> (<exp>, <exp>)**))	B	
62	mod 4	=	2	<fnc2> (<exp>, <fnc2> (<exp>, <fnc2> (<exp>, <var>)))	B	
70	mod 16	=	6	<fnc2> (<exp>, <fnc2> (<exp>, <fnc2> (<exp>, $\mathbf{E_{k-1}}$)))	E	
3	mod 4	=	3	<fnc2> (<exp>, <fnc2> (<exp>, <fnc2> (<const>, E_{k-1})))	B	
34	mod 10	=	4	<fnc2> (<exp>, <fnc2> (<exp>, <fnc2> (**5**, E_{k-1})))	E	
10	mod 4	=	2	<fnc2> (<exp>, <fnc2> (<exp>, *(5, E_{k-1})))	E	
25	mod 4	=	1	<fnc2> (<exp>, <fnc2> (**<fnc2> (<exp>, <exp>)**, *(5, E_{k-1})))	B	
34	mod 4	=	2	<fnc2> (<exp>, <fnc2> (<fnc2> (<exp>, <var>), *(5, E_{k-1})))	B	
37	mod 16	=	5	<fnc2> (<exp>, <fnc2> (<fnc2> (<exp>, $\mathbf{E_k}$), *(5, E_{k-1})))	E	
15	mod 4	=	3	<fnc2> (<exp>, <fnc2> (<fnc2> (<const>, E_k), *(5, E_{k-1})))	B	
11	mod 10	=	1	<fnc2> (<exp>, <fnc2> (<fnc2> (**2**, E_k), *(5, E_{k-1})))	E	
18	mod 4	=	2	<fnc2> (<exp>, <fnc2> (* (2, E_k), *(5, E_{k-1})))	E	
16	mod 4	=	0	<fnc2> (<exp>, + (* (2, E_k), *(5, E_{k-1})))	E	
42	mod 4	=	2	<fnc2> (<var>, + (* (2, E_k), *(5, E_{k-1})))	B	
16	mod 16	=	0	<fnc2> ($\mathbf{U_{k-1}}$, + (* (2, E_k), *(5, E_{k-1})))	E	
8	mod 4	=	0	+ (U_{k-1}, + (* (2, E_k), *(5, E_{k-1})))	E	

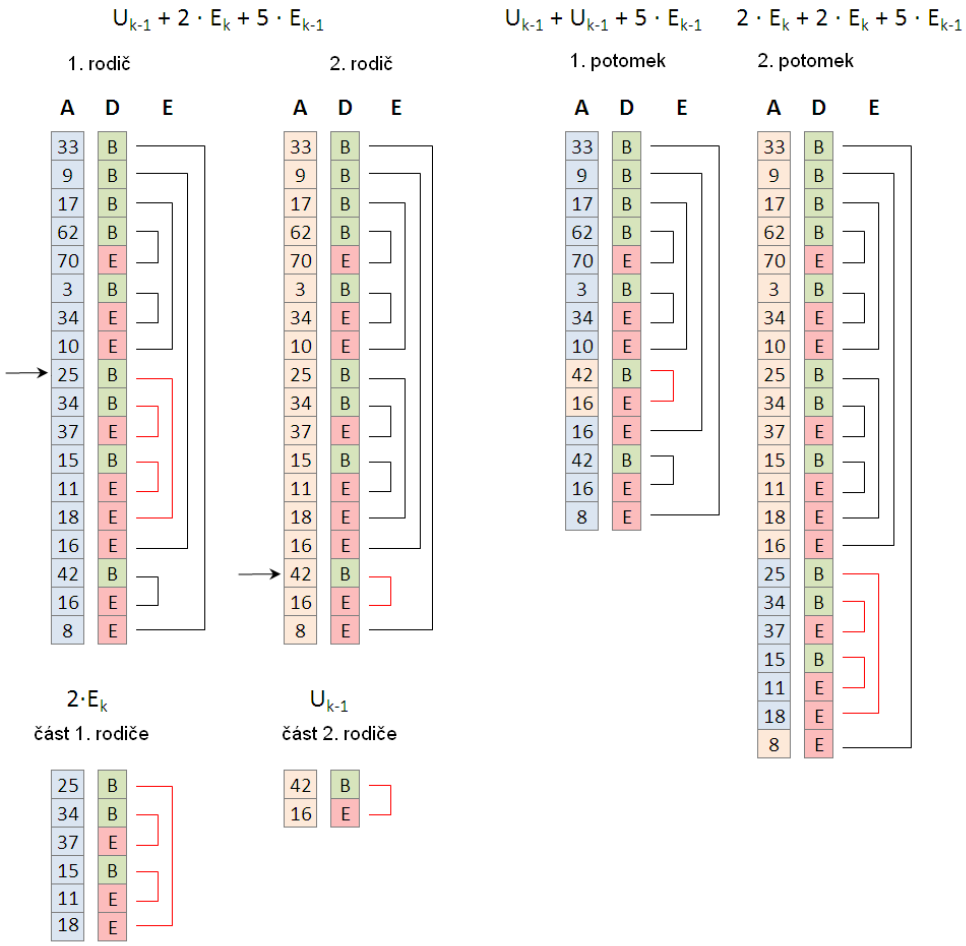
sloupec A: chromozóm, **sloupec B**: Výběr pravidla, **sloupec C**: Stav překladu. Červené symboly značí neterminály, které budou v dalším kroku nahrazeny, tučně jsou právě nahrazené, **sloupec D**: značky kodonů, **sloupec E**: označení párů kodonů

Obrázek 26: Překlad genotypu na fenotyp - backward processing

během překladu vygenerované hodnoty kodonů, *sloupec B* označuje aritmetickou operaci výběru pravidla, ve *sloupci C* je postup překladu těla jedince. V tomto sloupci tučné symboly označují právě vygenerovanou část těla jedince a červeně jsou označeny symboly, které budou nahrazeny. *Sloupec D* zobrazuje **vektor značek**, který je složen ze symbolů *B*, *E*. Symbol *B* označuje kodon, který je použit k výběru prvního pravidla určité části těla jedince, symbol *E* označuje kodon, kterým se generuje poslední terminální symbol daného bloku. Značky *B* a *E* jsou párové. Jednotlivé páry jsou vyznačeny ve *sloupci E*. Tyto páry značí část těla jedince, kterou lze zaměnit za jinou, bez ovlivnění zbývající části těla jedince. Tato vlastnost je dále vysvětlena v kapitolách 6.4 a 6.5.

6.4 Modifikované křížení

Tento druh křížení se vztahuje ke kapitole 6.3, která popisuje zpětné zpracování pravidel gramatiky a značkování kodonů. Díky tomuto značkování bylo možno aplikovat modifikovanou metodu křížení, která k zjištění vhodných křížících bodů používá právě tento vektor značek, viz. obrázek 27, *sloupec D*. Vektor značek $\vec{m} = (m_1, m_2, \ldots, m_n); \quad m \in \{B, E\}$ nese informaci o fenotypu jedince a slouží pro nalezení vhodných křížících bodů a jejich počtu.



sloupec A: chromozóm, **sloupec D**: značky kodonů, **sloupec E**: označení párů kodonů

Obrázek 27: Modifikované křížení

Příklad tohoto typu křížení je ukázán na obrázku 27, v jehož levé horní části jsou uvedeny rodičovské chromozómy, které byly vybrány ke křížení. Za zmínku stojí, že oba chromozómy mají stejný genotyp. Hodnoty kodonů jsou ve *sloupci A* a vektor značek je u každého rodiče ve *sloupci D*. Fenotyp obou rodičů je ve tvaru $U_{k-1} + 2 \cdot E_k + 5 \cdot E_{k-1}$. V obou rodičích se náhodně zvolí počáteční bod křížení. Tímto bodem může být libovolný bod z vektoru značek, označený B. K tomuto bodu je nutno nalézt jeho párový bod a celý blok od *značky B* po párovou značku E se vymění s vybraným blokem druhého rodiče. Počáteční křížící body jsou v tomto obrázku vyznačeny šipkou. Jednotlivé části jedinců jsou ve *sloupci E* vyznačeny spojnicemi. Blok, který je určen ke křížení je ve *sloupci E*

zvýrazněn červeně. Části prvního i druhého rodiče, které byly vybrány k záměně, jsou v levé části obrázku zobrazeny pod rodičovskými chromozómy. V pravé části obrázku jsou potomci, kteří vznikli vzájemnou výměnou bloků obou rodičů. V obou potomcích jsou jednotlivé části chromozómů vybarveny různou barvou podle genotypové příslušnosti k rodiči.

Jak je vidět, tento druh křížení nemá destruktivní vliv na strukturu již vygenerovaného jedince. Také bylo ukázáno, že ze dvou stejných rodičů je možno vytvořit dva rozdílné potomky i bez použití mutace. Tohoto není možné v případě klasického křížení dosáhnout, (viz. kapitola 3.8.2).

6.5 Modifikovaná mutace

Tento druh mutace se stejně jako modifikované křížení vztahuje ke kapitole 6.3 a shodně s metodou modifikovaného křížení používá vektor značek $\vec{m}$. Podle tohoto vektoru je možno určit části jedince k mutaci a rozhodnout zdali se bude mutovat struktura jedince, nebo část která neovlivní strukturu. Typ mutace tedy může být **strukturální** a **nestrukturální**. O typu mutace rozhoduje místo ve kterém dojde ke změně hodnoty kodonu (resp. kodonů).

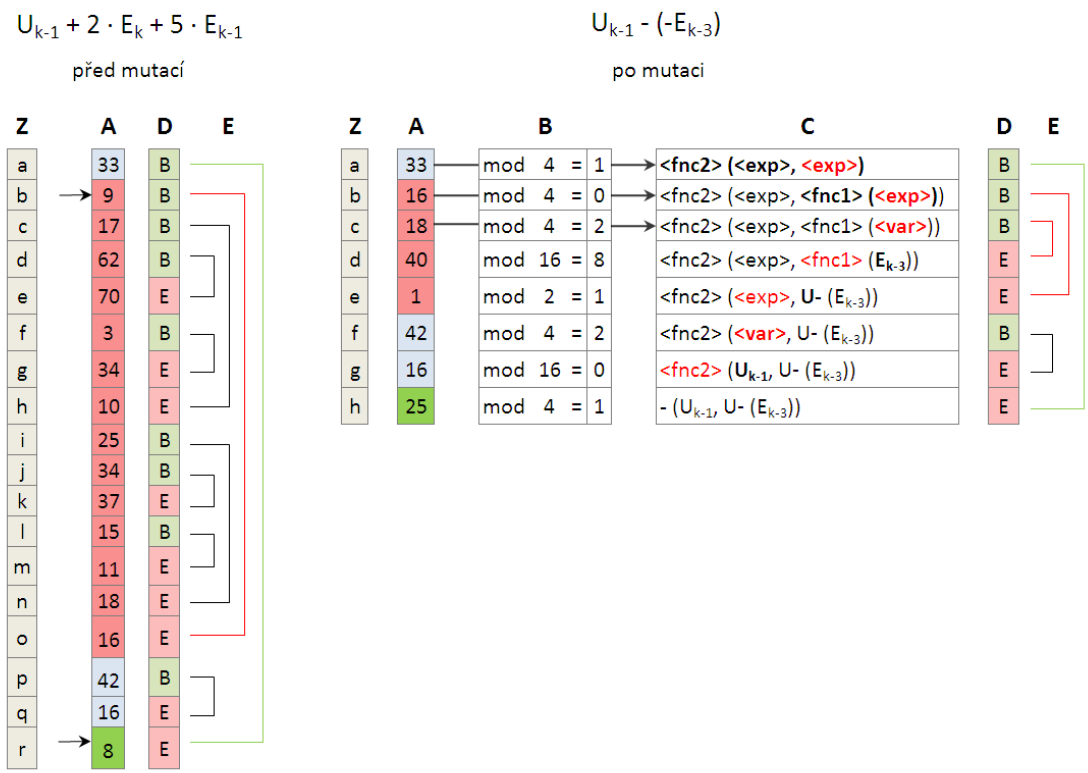
Příklad mutace je uveden na obrázku 28. Předpokladem tohoto příkladu je použití prefixového tvaru BNF gramatiky, jenž byla definována v tabulce 5.

6.5.1 Strukturální mutace

K tomuto typu mutace dojde vždy, když dojde ke změně hodnoty kodonu v místě označeném značkou B, pro kterou je splněna podmínka rovnice 6.1. V případě, že dojde ke změně hodnoty strukturálního kodonu, je potřeba zmutovat celou část jedince, která začíná v bodě B a končí v párovém bodě E. V případě, že by došlo k mutaci jen jednoho kodonu, který je označen B a zbytek bloku zůstal nezměněn, došlo by k tomu, že bude ovlivněna i jiná část těla jedince. To způsobí buď nevyužití všech kodonů v následujícím k překladu genotypu na fenotyp, nebo naopak jejich nedostatek.

Mutaci celé části bloku je možno realizovat novým generováním části bloku, podobně jak je tomu v případě inicializace jedince. Nově vygenerovaná část vymění část jedince, která je určena k mutaci, viz. obrázek 28.

V levé části obrázku je zobrazen jedinec před použitím operátoru mutace a v pravé části je tentýž jedinec po provedení mutace. Ve sloupci A je chromozóm jedince a sloupec D obsahuje vektor značek $\vec{m}$. Jednotlivé části těla jedince a párové kodony jsou vyznačeny ve sloupci E. V těle jedince byly určeny k provedení operátoru mutace dvě části chromozómu, které jsou označeny šipkami. První šipka u kodonu s pořadím b ve sloupci Z ukazuje na kodon, který je označen značkou B. To znamená, že tento kodon vybírá z **N–pravidla** a ovlivňuje strukturu jedince. S hodnotou tohoto kodonu je nutné změnit hodnoty kodonů celého bloku b až o (sloupec Z). Ve sloupci A a E jsou hodnoty kodonů, které podléhají strukturální mutaci vyznačeny červeně. V pravé části obrázku jsou červenou barvou vyznačeny nově vygenerované kodony, které nahradily původní blok vybraný k mutaci. Tento nově vygenerovaný blok je v pravé části obrázku ve sloupci Z označen pořadím b až e.



sloupec A: chromozóm, **sloupec B**: Výběr pravidla, **sloupec C**: Stav překladu. Červené symboly značí neterminály, které budou v dalším kroku nahrazeny, tučně jsou právě nahrazené, **sloupec D**: značky kodonů, **sloupec E**: označení párů kodonů, **sloupec Z**: označení pořadí kodonů

Obrázek 28: Modifikovaná mutace

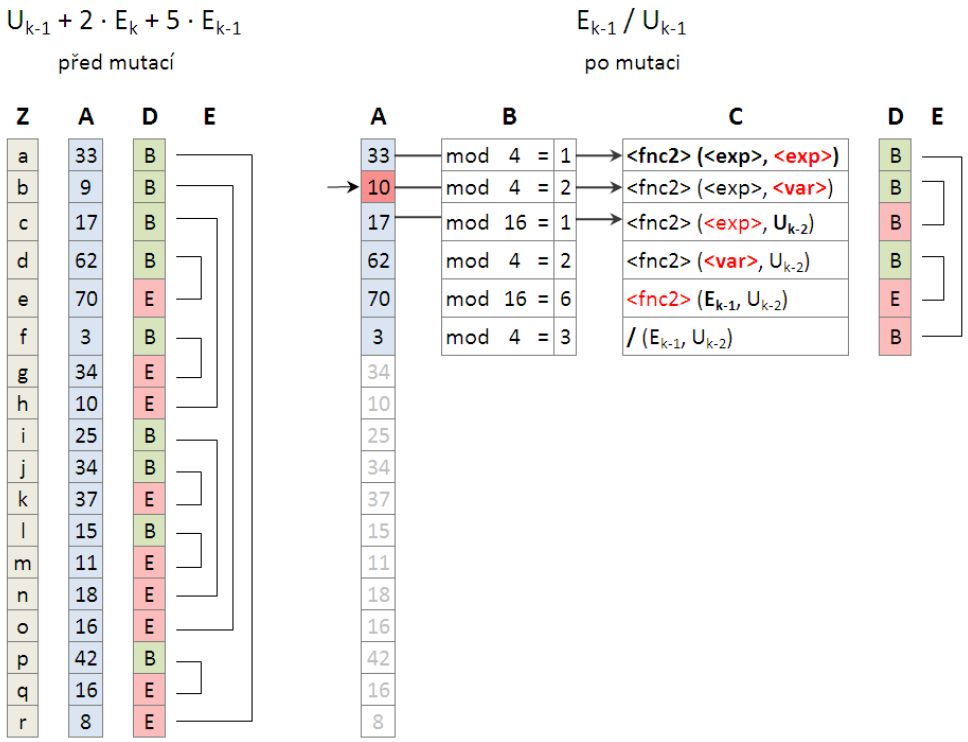
6.5.2 Nestrukturální mutace

V případě požadavku na nestrukturální mutaci, tedy takovou, která nezpůsobí změnu struktury fenotypu, je nutno provést změnu hodnoty kodonu, který rozhoduje o výběru z **T–pravidla**. Tyto kodony jsou ve sloupci D označeny značkou E. Úspěšnost mutace závisí na splnění podmínky rovnice 6.1. Tento druh mutace je ukázán v obrázku 28, ve kterém je označen druhou šipkou u posledního kodonu, který má ve sloupci Z pořadí r. Hodnota kodonu před změnou a po změně je v obou sloupcích A vyznačena zeleně a stejně tak odpovídající blok ve sloupci E. Nová hodnota kodonu je v pravé části obrázku ve sloupci Z v pořadí h. Je vidět, že změna hodnoty tohoto kodonu nezpůsobila změnu struktury fenotypu.

6.5.3 Chybně provedená mutace

Na obrázku 29 je ukázka důsledku chybně provedené změny hodnoty kodonu, který ovlivňuje výběr **N–pravidla**, tedy pravidla, které ovlivňuje výslednou strukturu fenotypu jedince. Tento kodon je vyznačen šipkou a zvýrazněn červeně. V levé části obrázku je původní chromozóm před aplikováním operátoru mutace a v pravé části obrázku je jedinec po mutaci. Ve sloupci A jsou šedou barvou vybarveny kodony, které zůstaly v chromozómu jedince a nepodílejí se na vytvoření těla jedince. Je vidět, že i malá změna hodnoty jedi-

ného strukturálního kodonu způsobí rozsáhlou změnu struktury fenotypu jedince. Opačným příkladem chybně provedené mutace je mutace, která vznikne v důsledku nedostatku kodonů k překladu.



sloupec A: chromozóm, **sloupec B**: Výběr pravidla, **sloupec C**: Stav překladu. Červené symboly značí neterminály, které budou v dalším kroku nahrazeny, tučně jsou právě nahrazené, **sloupec D**: značky kodonů, **sloupec E**: označení párů kodonů

Obrázek 29: Ukázka chybně provedené mutace strukturálního kodonu

Z hlediska mutace je problematickou vlastností různá váha kodonů v chromozómu. Tato váha, resp. strukturální významnost je nejvyšší na počátku chromozómu a postupně se snižuje. To znamená, že změna hodnoty kodonu, který je blíže k počátku chromozómu má na výsledný fenotyp větší destrukční účinek, než změna hodnoty posledního kodonu. V případě, že dojde ke změně hodnoty prvního kodonu, bude jedinec celý změněn.

6.6 Párování značek

Aby bylo možno určit bloky chromozómu vhodné ke křížení a mutaci je nutné zjistit jednotlivé páry kodonů. Každá část chromozómu začíná libovolnou značkou *B* a končí párovým kodonem, označeným *E*. Každý blok chromozómu je nezávislou, úplně definovanou částí fenotypu jedince. To je znázorněno například na obrázku 28. V tomto obrázku jsou jednotlivé bloky chromozómu označeny ve sloupci *E*.

Způsob, kterým je možno realizovat párování značek, je ukázán níže, viz. algoritmus 5. Vstupem tohoto algoritmu je vektor značek. Výstupem je počáteční a koncový index kodonu, který vymezuje náhodně zvolený blok.

Algoritmus 5: Párování značek

```
Input: M – vector of marks
Result: block of chromosome (start, end index)
startIdx ← GetRandomBIdx(M) ;                    // náhodné vybrání B kodonu
endIdx ← startIdx;
pairNum ← 1;
while pairNum > 0 do
    Increase(endIdx) ;                           // zvýšení o jedničku
    if M[endIdx] = 'B' then
        Increase(pairNum) ;   // zvýšení o jedničku, jestliže značka kodonu je B
    else
        Decrease(pairNum) ;                      // snížení o jedničku
    end
end
return startIdx, endIdx;
```

6.7 Generování čísel pomocí gramatiky

GE je vhodná pro vytváření komplexních struktur, kterými mohou být například programy, funkce, logický obvod, apod. V optimalizačních úlohách je poměrně často vyžadována také optimalizace parametrů. Příkladem může být návrh regulátoru, kde je nutno nejprve nalézt strukturu regulátoru a poté nastavit optimální parametry. Jiným příkladem může být například řešení symbolické regrese, apod.

Příklady v předchozích kapitolách byly pro jednoduchost představeny s upravenými pravidly gramatiky, které umožňovaly generování pouze celých čísel z intervalu $\langle 0, 9\rangle$. Tento rozsah parametrů je pro praktické použití nedostatečný a je třeba zvolit jiný způsob generování parametrů, nejlépe z oboru reálných čísel $\mathbb{R}$.

V kapitole 6.1, viz. tabulka 6 byly ukázány pravidla, které je možno použít ke generování parametrů z oboru *reálných čísel*. Další možností je použít pravidla uvedená v tabulce 7. Těmito pravidly je možno generovat pomocí aritmetických operací všechny **iracionální čísla**.

$\langle exp\rangle$	::=	$\langle fnc\rangle\langle exp\rangle\langle exp\rangle \mid \langle const\rangle$
$\langle const\rangle$	::=	1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
$\langle fnc\rangle$	::=	/ | +

Tabulka 7: Generování reálných čísel

Další možností je použít gramatiku, která obsahuje pravidla pro přímé generování čísel. Tato gramatika zavádí funkci **desetinná tečka** a má tři argumenty, z nichž první argument určuje pozici desetinné tečky a zbylé dva argumenty určují jednotlivé cifry čísla [35].

Generování čísel pomocí gramatiky má poměrně velkou složitost a pro optimalizaci parametrů regulátoru je prakticky nepoužitelné. Optimalizace parametrů regulátoru by pomocí gramatiky byla náročnější, než samotné hledání jeho struktury. V případě použití pravidel z tabulky 7 je k vygenerování čísla $0,5$ nutné použít 6 kodonů. Komplexita generování konstanty roste se vzdáleností od základního rozsahu přímo generovaných čísel, které jsou možným přepisem neterminálu $\langle const\rangle$.

Vzhledem k výše zmíněným skutečnostem je vhodné využit pro generování konstant jinou, sofistikovanější metodu, sloužící pro optimalizaci numerických parametrů. Takovou metodou může být například diferenciální evoluce, (viz. kapitola 4), genetický algoritmus, (viz. kapitola 3), apod.

Požadavek na současnou optimalizaci struktury i parametrů jedince vedl k vytvoření dvoufázového evolučního algoritmu, který v první fázi optimalizuje strukturu řešení a ve druhé fázi optimalizuje neznámé parametry této struktury. V této práci byla použita **dvoufázová transplantační evoluce**, (viz. kapitola 8), která byla pro tuto práci navržena a testována ve složení:

- transplantační evoluce + diferenciální evoluce
- transplantační evoluce + genetický algoritmus

Principy **transplantační evoluce** jsou vysvětleny v kapitole 7.

7 Transplantační evoluce

Transplantační evoluce (TE) *(Transplant Evolution)* je nový typ evolučního algoritmu, který vznikl spojením nejlepších vlastností *genetického programování* [21] a *gramatické evoluce*, (viz. kapitola 6) [43, 45, 42].

TE je populačně založený evoluční algoritmus, který používá k popisu řešení pravidla gramatiky. Gramatikou se řídí inicializace jedinců, operátory křížení i mutace. Tento algoritmus přebírá všechny výhody GE se zpětným zpracováním pravidel gramatiky a přidává navíc další výhodné vlastnosti, kterými jsou:

- efektivní změna pravidel gramatiky v průběhu evoluce, (viz. kapitola 7.1)
- nastavení různé pravděpodobnosti výběru pro každé pravidlo gramatiky, (viz. kapitola 7.4)
- použití metody přímého zkracování stromu pomocí algebraických operací (viz. kapitola 8.8)
- operátory křížení a mutace jsou řízeny pravidly gramatiky, (viz. kapitoly 7.2, 7.3)
- nezávislost na směru zpracování gramatiky, (viz. kapitola 7.1)

Vzhledem k tomu, že jedinci v TE jsou reprezentováni pouze **fenotypem** ve formě stromu, není důležitý postup jeho vytvoření. **Genotyp** jedince je vytvářen pouze v průběhu inicializace stromu a po jeho vytvoření je zahozen. Křížení a mutace pracuje přímo s fenotypem, za použití pravidel gramatiky. Proces křížení a mutace je tedy ekvivalentní transplantaci orgánů v biologii, při které se rovněž nepracuje na úrovni **DNA**. Díky této ekvivalenci získala metoda jméno **transplantační evoluce**.

7.1 Inicializace jedince

Postup zpracování gramatiky probíhá podobně jako v případě GE se zpětným zpracováním pravidel, (viz. kapitola 6.3) s rozdílem, že výběr pravidel probíhá náhodným výběrem, dle vztahu 7.1:

$$random(0, maxInt) \ \% \ rules_count \tag{7.1}$$

Kde: $random$ je generátor náhodných čísel,
$maxInt$ je nějaké vysoké číslo,
% značí aritmetickou operaci celočíselného dělení
$rules_count$ značí počet pravidel k výběru.

Uvedené ukázky inicializace jedince, viz. algoritmus 6 a hlavně obrázek 30, předpokládají gramatiku definovanou v tabulce 5 (strana 45).

Z algoritmu je možno vyčíst, že zpracování pravidel je prováděno odzadu, ale malou změnou algoritmu lze bez následků aplikovat i dopředné zpracování gramatiky, které by ve finále způsobilo pouze jiné pořadí inicializace stromu, viz. obrázek 31. Kromě již zmíněné *prefixové BNF gramatiky*, jsou vstupními parametry tohoto algoritmu *pole pravidel*, na počátku nastavené na startovní symbol gramatiky $\langle S \rangle$ a *maximální počet uzlů*. Výstupem algoritmu je řešení, reprezentované ve stromové struktuře. Výběr pravidla, které přepíše neterminální symbol, se provádí na řádku 9, voláním funkce *Get_random_rule*. Přidání větve k aktuálně procházenému uzlu se provádí na řádku 12 a nastavení terminálního

Algoritmus 6: Inicializace jedince

Input: G – context-free grammar; P – array of rules; L_max – max nodes count
Result: individual

1 **if** length*[this]* $\geq$ *L_max* **then**
2 | Disable_rules('function') ; `// zakázání pravidel, které prodlužují strom jedince`
3 **end**
4 **for** $i \leftarrow$ length*[P]* **to** $i \geq 1$ **do**
5 | **if** *P[i] is terminal symbol* **then**
6 | | $rule \leftarrow P[i]$
7 | **else**
8 | | $P_temp \leftarrow$ array of rules for the nonterminal $P[i]$ in G indexed from zero;
9 | | $rule \leftarrow$ Get_random_rule(P_temp) ; `// náhodný výběr pravidla z P_temp`
10 | **end**
11 | $sub_tree \leftarrow$ CreateSubTree($G; rule; L_max$) ; `// rekurze`
12 | $this \leftarrow$ add child sub_tree ; `// přidá do stromu další větev k aktuálnímu uzlu`
13 **end**
14 **if** *P[0] is terminal* **then**
15 | $this.terminal = P[0]$; `// přiřazení terminálního symbolu aktuálnímu uzlu`
16 **else**
17 | $sub_tree \leftarrow$ CreateSubTree($G; P[0]; L_max$) ; `// rekurze`
18 | $this.terminal \leftarrow sub_tree$;
19 **end**
20 **return** *this*;

symbolu se děje až po předchozí inicializaci celého podstromu na řádku 18. Poměrně důležitý je řádek 2, kterým je možno **dynamicky měnit pravidla**, když počet uzlů stromu přesáhne určitou délku, definovanou v proměnné *maximální počet uzlů*. Zakázáním pravidel, které slouží k větvení stromu, dojde k jeho uzavření. Pravidla, které prodlužují délku stromu jsou například $\langle fnc1 \rangle$, nebo $\langle fnc2 \rangle$. Tímto se TE liší od klasické GE, jelikož GE neumožňuje dynamickou změnu pravidel během evoluce. Změněná gramatika v důsledku přesáhnutí délky stromu je ukázána v tabulce 8. V této tabulce je možné si všimnout,

$\langle S \rangle$	::=	$\langle exp \rangle$
$\langle exp \rangle$	::=	$\langle var \rangle \mid \langle const \rangle$
$\langle fnc1 \rangle$	::=	$U+ \mid U-$
$\langle fnc2 \rangle$	::=	$+ \mid - \mid * \mid /$
$\langle var \rangle$	::=	$U_{k-1} \mid U_{k-2} \mid U_{k-3} \mid U_{k-4} \mid U_{k-5} \mid$ $E_k \mid E_{k-1} \mid E_{k-2} \mid E_{k-3} \mid E_{k-4} \mid E_{k-5} \mid$ $dE_k \mid dE_{k-1} \mid dE_{k-2} \mid dE_{k-3} \mid dE_{k-4}$
$\langle const \rangle$	::=	$1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \mid 0$

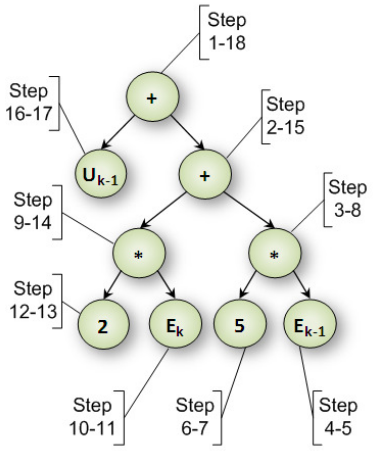
Tabulka 8: Změněná gramatika v důsledku překročení délky stromu

že **N–pravidlo** $\langle exp \rangle$ již není možné nahradit původně definovanými čtyřmi přepisy, ale pouze dvěma, jelikož pravidla $\langle fnc1 \rangle \langle exp \rangle$ a $\langle fnc2 \rangle \langle exp \rangle \langle exp \rangle$ byly odstraněny. Odstranění pravidel je možno realizovat nastavením nulové pravděpodobnosti výběru.

Proces vytváření stromu je ukázán na obrázku 30. *Sloupec A* obsahuje náhodně generované hodnoty během překladu, *sloupec B* ukazuje způsob výběru z množiny gramatických pravidel, *sloupec C* ukazuje stav překladu, ale je třeba si uvědomit, že tento překlad se děje na úrovni stromu. Pro přehlednost je formulován tak, jak byl prezentován v předcho-

A	B			C	F
33	mod 4 =	1	→	**<fnc2> (<exp>, <exp>)**	1
9	mod 4 =	1	→	<fnc2> (<exp>, **<fnc2> (<exp>, <exp>)**)	2
17	mod 4 =	1	→	<fnc2> (<exp>, <fnc2> (<exp>, **<fnc2> (<exp>, <exp>)**))	3
62	mod 4 =	2		<fnc2> (<exp>, <fnc2> (<exp>, <fnc2> (<exp>, **<var>**)))	4
70	mod 16 =	6		<fnc2> (<exp>, <fnc2> (<exp>, <fnc2> (<exp>, **E_{k-1}**)))	5
3	mod 4 =	3		<fnc2> (<exp>, <fnc2> (<exp>, <fnc2> (**<const>**, E_{k-1})))	6
34	mod 10 =	4		<fnc2> (<exp>, <fnc2> (<exp>, <fnc2> (**5**, E_{k-1})))	7
10	mod 4 =	2		<fnc2> (<exp>, <fnc2> (<exp>, *****(5, E_{k-1})))	8
25	mod 4 =	1		<fnc2> (<exp>, <fnc2> (**<fnc2> (<exp>, <exp>)**, *(5, E_{k-1})))	9
34	mod 4 =	2		<fnc2> (<exp>, <fnc2> (<fnc2> (<exp>, **<var>**), *(5, E_{k-1})))	10
37	mod 16 =	5		<fnc2> (<exp>, <fnc2> (<fnc2> (<exp>, **E_k**), *(5, E_{k-1})))	11
15	mod 4 =	3		<fnc2> (<exp>, <fnc2> (<fnc2> (**<const>**, E_k), *(5, E_{k-1})))	12
11	mod 10 =	1		<fnc2> (<exp>, <fnc2> (<fnc2> (**2**, E_k), *(5, E_{k-1})))	13
18	mod 4 =	2		<fnc2> (<exp>, <fnc2> (***** (2, E_k), *(5, E_{k-1})))	14
16	mod 4 =	0		<fnc2> (<exp>, **+** (* (2, E_k), *(5, E_{k-1})))	15
42	mod 4 =	2		<fnc2> (**<var>**, + (* (2, E_k), *(5, E_{k-1})))	16
16	mod 16 =	0		<fnc2> (**U_{k-1}**, + (* (2, E_k), *(5, E_{k-1})))	17
8	mod 4 =	0		**+** (U_{k-1}, + (* (2, E_k), *(5, E_{k-1})))	18

G



sloupec A: náhodně generované hodnoty během inicializace stromu, **sloupec B**: výběr pravidla, **sloupec C**: stav překladu. Červené symboly značí neterminály, které budou v dalším kroku nahrazeny, tučně jsou právě nahrazené, **sloupec F**: pořadí operace, **sloupec G**: strom funkce s vyznačeným postupem v jednotlivých krocích (první číslice značí krok ve kterém byl uzel vytvořen, druhé číslo značí krok, kdy byl plně inicializován)

Obrázek 30: Inicializace jedince - zpětné zpracování pravidel

zích kapitolách. Jednotlivé kroky řešení jsou očíslovány ve *sloupci F*, který je využit pro popis postupu vytváření stromu, který je znázorněn ve *sloupci G*. Každý uzel stromu ve *sloupci G* je popsán číslem ve formátu *Step X – Y*, které znamená, že uzel byl vytvořen v kroku *X* a byl plně inicializován, po vytvoření jeho podřízených větví, v kroku *Y*.

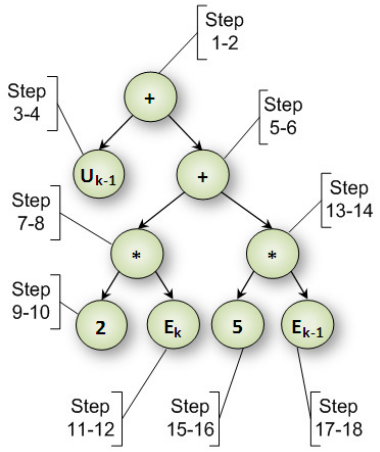
Z algoritmu 6 a obrázku 30 je možno vyčíst, že narozdíl od GE popsané v kapitole 6.3 se náhodně vygenerované hodnoty (kodony), které rozhodují o výběru pravidla, neukládají do chromozómu. Také je možné si všimnout, že není aplikováno značkování jednotlivých částí jedince, jelikož v případě TE není potřeba. Ve *sloupci G* je vidět, že jednotlivé části jedince, tak jak byly značkovány v GE, jsou ve stromu podřízeny svému rodičovskému uzlu.

Na obrázku 31 je ukázán postup vytváření stromu v případě použití **dopředného překladu** pravidel gramatiky. Tento obrázek, shodně s obrázkem 30, předpokládá gramatiku definovanou v tabulce 5 (strana 45). Pro srovnání je výsledný strom fenotypově stejný se stromem definovaným na obrázku 30, kde je ukázáno vytvoření stromu zpětným překladem pravidel gramatiky.

Vzhledem k tomu, že každá větev stromu představuje úplně vyčíslitelnou část, tedy takovou, která může stát i samostatně, lze aplikovat křížení, (viz. kapitola 7.2) a mutaci, (viz. kapitola 7.3) přímo na tento vygenerovaný strom. Díky tomu, že TE nepoužívá genotyp pro reprezentaci těla jedince, je možno použít metodu přímého zkracování stromu,

A	B	C	F
33	mod 4 = 1	**<fnc2> (<exp>, <exp>)**	1
8	mod 4 = 0	+ (<exp>, <exp>)	2
18	mod 4 = 2	+ (**<var>**, <exp>)	3
64	mod 16 = 0	+ (**U_{k-1}**, <exp>)	4
5	mod 4 = 1	+ (U_{k-1}, **<fnc2>(<exp>, <exp>)**)	5
4	mod 4 = 0	+ (U_{k-1}, **+**(<exp>, <exp>))	6
17	mod 4 = 1	+ (U_{k-1}, +(**<fnc2>(<exp>, <exp>)**, <exp>))	7
34	mod 4 = 2	+ (U_{k-1}, +(*****(<exp>, <exp>), <exp>))	8
35	mod 4 = 3	+ (U_{k-1}, +(*****(**<const>**, <exp>), <exp>))	9
51	mod 10 = 1	+ (U_{k-1}, +(*****(**2**, <exp>), <exp>))	10
14	mod 4 = 2	+ (U_{k-1}, +(*****(2, **<var>**), <exp>))	11
37	mod 16 = 5	+ (U_{k-1}, +(*****(2, **E_k**), <exp>))	12
9	mod 4 = 1	+ (U_{k-1}, +(*****(2, E_k), **<fnc2>(<exp>, <exp>)**))	13
10	mod 4 = 2	+ (U_{k-1}, +(*****(2, E_k), *****(<exp>, <exp>)))	14
15	mod 4 = 3	+ (U_{k-1}, +(*****(2, E_k), *****(**<const>**, <exp>)))	15
4	mod 10 = 4	+ (U_{k-1}, +(*****(2, E_k), *****(**5**, <exp>)))	16
6	mod 4 = 2	+ (U_{k-1}, +(*****(2, E_k), *****(5, **<var>**)))	17
38	mod 16 = 6	+ (U_{k-1}, +(*****(2, E_k), *****(5, **E_{k-1}**)))	18

G



sloupec A: náhodně generované hodnoty během inicializace stromu, **sloupec B**: výběr pravidla, **sloupec C**: stav překladu. Červené symboly značí neterminály, které budou v dalším kroku nahrazeny, tučně jsou právě nahrazené, **sloupec F**: pořadí operace, **sloupec G**: strom funkce s vyznačeným postupem v jednotlivých krocích (první číslice značí krok ve kterém byl uzel vytvořen, druhé číslo značí krok, kdy byl plně inicializován)

Obrázek 31: Inicializace jedince - dopředné zpracování pravidel

která zajistí minimalizaci jeho délky, (viz. kapitola 8.8).

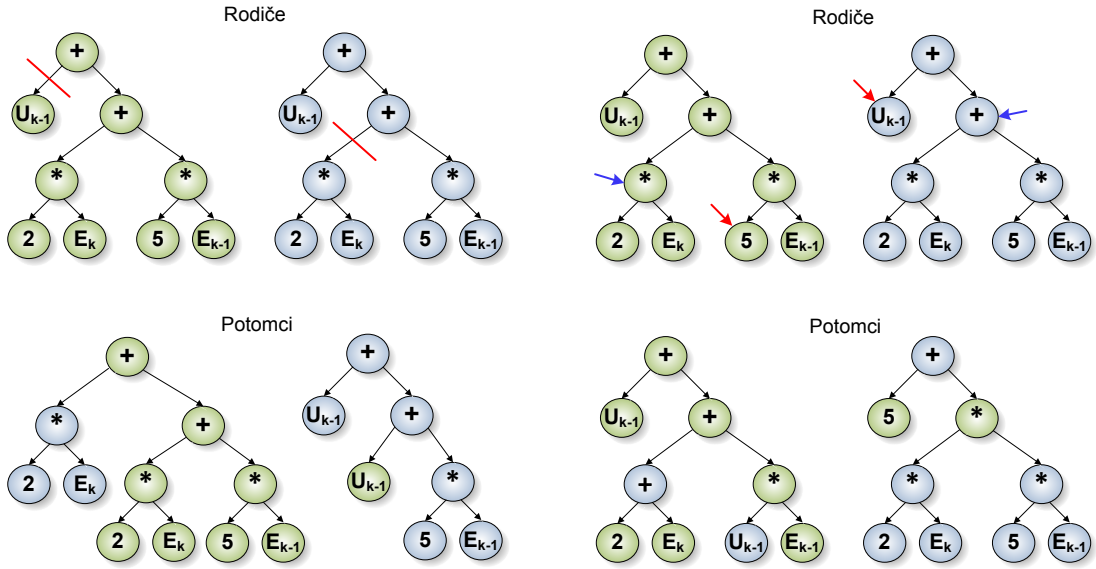
7.2 Křížení - Transplantace

Díky tomu, že TE používá genotyp jen pro inicializaci jedince a dále je řešení reprezentováno pouze fenotypem ve formě stromu, je metoda křížení navržena pro operace se stromovou strukturou, podobně jako v případě GP [21]. Křížení v TE tedy pracuje na vyšší úrovni než v případě jiných evolučních algoritmů, jejichž operace a zásah do fenotypu se děje na úrovni *DNA* a z něj je poté překladem vytvořen fenotyp. Proces křížení v TE se tedy dá přirovnat k *transplantační chirurgii*, kde dochází k výměně již hotových orgánů. Z toho důvodu bylo křížení v TE nazváno **transplantace** *(transplantation)*.

V této práci byly navrženy dva typy křížení. V prvním z nich vznikají *dva potomci* křížením vybraných částí stromů dvou rodičů, (viz. kapitola 7.2.1) a ve druhé metodě vzniká *jeden potomek* spojením stromů, dvou náhodně vybraných rodičů, (viz. kapitola 7.2.2).

7.2.1 Křížení částí stromů

Během tohoto typu křížení vznikají *dva potomci* výměnou částí stromů *dvou rodičů*. Tento způsob křížení je ukázán na obrázku 32 a zahrnuje křížení celých podstromů a křížení uzlů.



(a) Křížení podstromů (b) Křížení uzlů

Obrázek 32: Křížení částí stromů

Křížení podstromů: je metodou, která kombinuje celé části rodičovských podstromů a ekvivalentně se dá nazvat **úplnou transplantací**. Tento typ křížení je ukázán na obrázku 32a, v jehož horní části jsou dva rodiče a v dolní části jejich potomci, kteří vznikli rekombinací částí stromů obou rodičů. Stojí za připomínku, že oba rodiče mají stejný fenotyp $U_{k-1} + 2 \cdot E_k + 5 \cdot E_{k-1}$. U obou rodičů je červenou čárou naznačen řez, ve kterém bude provedeno křížení. Z tohoto obrázku je patrné, že i kombinací dvou naprosto identických rodičů mohou vzniknout různí potomci.

Křížení uzlů: je metodou, kerá kombinuje zvolené uzly rodičovských stromů. Tento typ křížení je ekvivalentní **částečné transplantaci**. Metoda je ukázána na obrázku 32b, v jehož horní části jsou dva rodiče, kteří mají pro větší názornost stejný fenotyp jako v případě *křížení podstromů*. U každého rodiče jsou ke křížení vybrány dva uzly, které jsou označeny červenou a modrou šipkou. Barva šipky určuje typ uzlu, který může být zaměněn pouze uzlem stejného typu. Záměrně byly v obou rodičích zvoleny typově různé uzly, z nichž ten, který je označen *modrou šipkou* je **strukturální uzel**, který má dva následníky a *červenou šipkou* je označen **nestrukturální uzel**, který je koncovým uzlem, čili bez následníků. **Strukturální uzel** je uzel, který se podílí na vytvoření struktury stromu jedince. V tomto konkrétním případě se jedná o uzly, jenž byly vytvořeny překladem **neterminálního symbolu** $\langle fnc1 \rangle$, nebo $\langle fnc2 \rangle$. **Nestrukturální uzel** je takový uzel, který se nepodílí na vytvoření struktury stromu jedince a v tomto konkrétním případě se jedná o uzly, jenž vznikly překladem **neterminálního symbolu** $\langle const \rangle$, nebo $\langle var \rangle$.

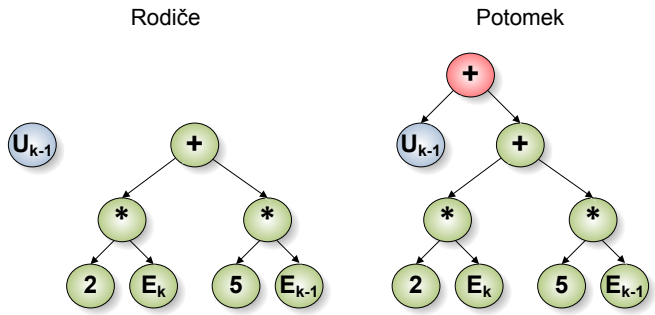
Jak již bylo uvedeno, tento typ křížení se řídí pravidlem záměny typově stejných uzlů, čili uzlů, které mají stejný počet následníků. Pro gramatiku definovanou v tabulce 5 (strana 45) tedy platí vztah 7.2, který vyjadřuje, že lze vzájemně kombinovat uzly, jejichž

terminální symbol vznikl překladem typově stejného **neterminálního symbolu**.

$$\langle fnc1\rangle \nLeftrightarrow \langle fnc2\rangle \nLeftrightarrow \{\langle var\rangle \Leftrightarrow \langle const\rangle\} \tag{7.2}$$

7.2.2 Spojování stromů

Tento typ křížení, i když má obecnou použitelnost i mimo tento případ, vznikl s cílem zabránění vytvoření identických kopií rodičů v případech, kdy jsou pro křížení zvoleny jejich kořenové uzly. Při tomto druhu křížení je ze *dvou rodičů* vytvořen *jeden potomek*. Tímto se metoda podstatně liší od metody *křížení částí stromů*, popsané v kapitole 7.2.1. Princip tohoto druhu křížení je ukázán na obrázku 33. V levé části obrázku jsou zobra-



Obrázek 33: Spojování stromů

zeny dva rodičovské stromy, z nichž každý je vybarven jinou barvou a v pravé části je potomek, který vznikl *spojením* stromů obou rodičů. Toto spojení vzniklo pomocí nově, náhodně vygenerovaného uzlu, jenž se stal novým kořenem stromu. V případě gramatiky definované v tabulce 5 (strana 45), se kořenovým uzlem mohou stát všechny uzly, které vzniknou přepisem pravidla, v jehož *kontextu* se nacházejí dva neterminální symboly $\langle exp\rangle$. Takovým pravidlem může být například $\langle fnc2\rangle$, jelikož právě tento neterminální symbol je v pravidlech vyjádřen v kontextu dvou neterminálních symbolů $\langle exp\rangle$, které vyjadřují, že uzlu, který je vygenerován přepisem neterminálu $\langle fnc2\rangle$, jsou podřízeny dva uzly, definované přepisem dvou neterminálů $\langle exp\rangle$, viz. algoritmus 6. Vzhledem k tomu, že přepisem neterminálního symbolu $\langle exp\rangle$ může dojít k odvození jakéhokoliv terminálního symbolu definovaného gramatikou popsanou v tabulce 5, mohou být neterminály $\langle exp\rangle$ nahrazeny stromy obou rodičů. Na obrázku 33 je nově vygenerovaný kořenový uzel potomka vybarven červeně.

7.3 Mutace

Mutace je jedním z operátorů rekombinace a zajišťuje diverzitu populace vkládáním nových znaků do struktury jedince. Mutace je v TE navržena pro operace se stromovou strukturou, tedy s fenotypem jedince, stejně tak jako křížení, (viz. kapitola 7.2). Tímto se opět liší od tradiční GE a GA obecně, jelikož u těchto evolučních algoritmů dochází k mutaci částí numericky vyjádřených chromozómů, ze kterých se následně procesem translace genotypu získá fenotyp, viz. kapitola 3.9, 6.2.2.

Proces mutace se v TE provádí použitím pravidel **analytické** G^{-1} a **generativní** G gramatiky a lze jej vyjádřit ve tvaru $\beta_1 \rightarrow \alpha \rightarrow \beta_2$, kde $(\beta_1, \beta_2) \in \Sigma;\ \beta_1 \neq \beta_2;\ \alpha \in N$,

(viz. kapitola 5). To mj. znamená, že terminální symboly před použitím operátoru mutace a po ní jsou vzájemně různé. Jednotlivá pravidla analytické gramatiky mohou mít různou **pravděpodobnost výběru**, stejně tak, jako je tomu v případě generativní gramatiky.

K množině přepisovacích pravidel, je vytvořena **inverzní množina pravidel** P^{-1}, která řídí možný přepis terminálního symbolu na neterminální. V průběhu procesu mutace je náhodně vybrán terminální symbol, poté je ze seznamu inverzních pravidel P^{-1} analytické gramatiky G^{-1} vybrán jeho možný přepis na neterminální symbol a následně je tento symbol náhodně přepsán podle pravidel P, generativní gramatiky G.

V této práci byly navrženy dva typy mutace, které se dělí podle rozsahu změn které na fenotypu jedince (stromu) způsobí:

- nestrukturální mutace, (viz. kapitola 7.3.1)
- strukturální mutace (prodlužující, zkracující), (viz. kapitola 7.3.2)

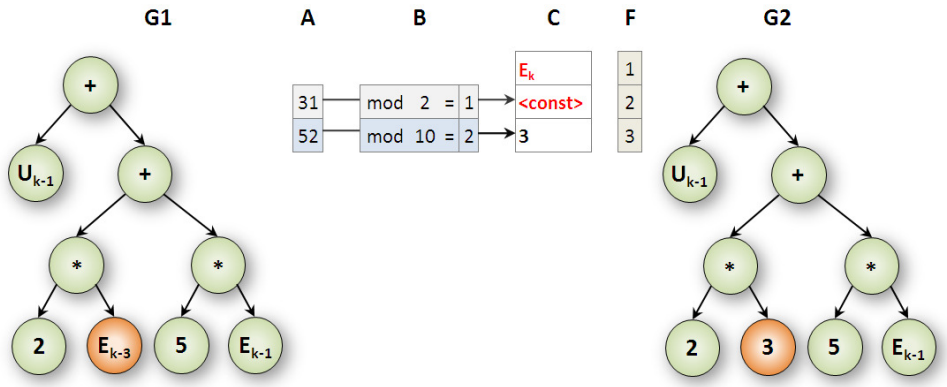
7.3.1 Nestrukturální mutace

Nestrukturálním mutace *(non-structural mutation)* nemění strukturu již vytvořených řešení (stromů). Aby bylo zajištěno, že se nezmění struktura stromu, mohou se mutovat jen typově stejné uzly. To znamená, že počet přímých následníků uzlu před mutací i po mutaci musí být stejný. Tento typ mutace probíhá ve dvou krocích. V prvním kroku

$U_{k-1} \mid U_{k-2} \mid U_{k-3} \mid U_{k-4} \mid U_{k-5}$	$::=$	$\langle var \rangle \mid \langle const \rangle$
$E_k \mid E_{k-1} \mid E_{k-2} \mid E_{k-3} \mid E_{k-4} \mid E_{k-5}$	$::=$	$\langle var \rangle \mid \langle const \rangle$
$dE_k \mid dE_{k-1} \mid dE_{k-2} \mid dE_{k-3} \mid dE_{k-4}$	$::=$	$\langle var \rangle \mid \langle const \rangle$
$decimal$	$::=$	$\langle var \rangle \mid \langle const \rangle$
$U+ \mid U-$	$::=$	$\langle fnc1 \rangle$
$+ \mid - \mid * \mid /$	$::=$	$\langle fnc2 \rangle$

Tabulka 9: Pravidla analytické gramatiky pro nestrukturální mutaci

je terminální symbol zvoleného uzlu přepsán na neterminál podle pravidel definovaných v tabulce 9 a ve druhém kroku, je tento neterminální symbol přepsán zpět na terminální, podle pravidel definovaných v tabulce 5 (strana 45). V případě, že terminální symbol zvoleného uzlu není v tabulce analytické gramatiky uveden, není jeho mutace dovolena. Z uvedené tabulky 9 je možno vyčíst, že obsahuje na levé straně *terminální symboly* a na pravé straně jejich možné přepisy na *neterminální symboly.* Dále se na straně terminálních symbolů vyskytuje terminál *decimal*, který je zástupným symbolem pro terminály, reprezentované numerickými hodnotami. Uvedený druh mutace je ukázán na obrázku 34. V tomto obrázku *sloupec A* obsahuje náhodně generované hodnoty, podle kterých se řídí překlad terminálních, resp. neterminálních symbolů, ve *sloupci B* je ukázána operace pro výběr pravidla z množiny pravidel generativní, nebo analytické gramatiky. Ve *sloupcích A, B* je stejnou barvou vyznačeno pro jakou gramatiku operace probíhá. *Šedou* barvou je označen výběr z množiny pravidel analytické gramatiky a *modrou* barvou je označen výběr z množiny pravidel generativní gramatiky. Ve *sloupci F* jsou vyznačeny pořadí operací. *Sloupec G1* obsahuje strom před použitím operátoru mutace a ve *sloupci G2* je tentýž strom po mutaci.



sloupec A: náhodně generované hodnoty během mutace, **sloupec B**: výběr pravidla z analytické, nebo generativní gramatiky, **sloupec C**: stav mutace uzlu. Červené symboly značí neterminály, nebo terminály, které budou v dalším kroku nahrazeny, tučně jsou právě nahrazené, **sloupec F**: pořadí operace, **sloupec G1**: strom funkce před použitím operátoru mutace. Uzel k mutaci je vyznačen oranžovou barvou, **sloupec G2**: strom funkce po použití operátoru mutace. Zmutovaný uzel je vyznačen oranžově.

Obrázek 34: Nestrukturální mutace

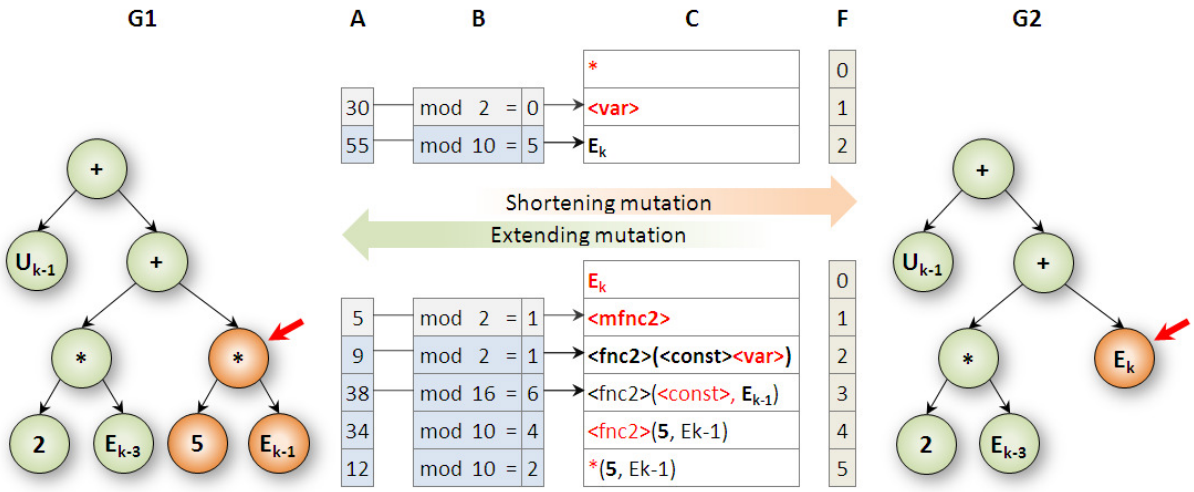
Z obrázku 34 je vidět, že při použití nestrukturální mutace nedojde za žádných okolností ke změně délky stromu, narozdíl od mutace strukturální, která mění délku stromu vždy, (viz. kapitola 7.3.2).

7.3.2 Strukturální mutace

Strukturální mutace narozdíl od nestrukturální mutace, (viz. kapitola 7.3.1) ovlivňuje délku stromu. Ke strukturální mutaci dojde tehdy, změní-li se počet přímých následníků mutovaného uzlu. Podle typu změny délky stromu, resp. počtu jeho uzlů se strukturální mutace dělí na **prodlužující** a **zkracující**.

Oba typy strukturální mutace jsou ukázány na obrázku 35. V obrázku jsou uvedeny dva stromy funkce a dvě tabulky postupu překladu s ukázkou výběru typu pravidla a způsobu jakým bylo toto pravidlo vybráno. Uvedené příklady předpokládají gramatiku definovanou v tabulkách 5, 10, 11, 12.

Rozsah a typ strukturální mutace závisí na tvaru pravidel analytické gramatiky. Čím více k základnímu symbolu se dostáváme použitím přepisovacího pravidla G^{-1} gramatiky, jenž přepíše terminální symbol, tím více nepředvídatelný bude výsledek použití operátoru mutace. V případě, že máme definovanou analytickou gramatiku v níž je pravidlo, které dovoluje přepsat terminální symbol na symbol $\langle exp \rangle$ a generativní gramatiku, definovanou v tabulce 5, dostaneme naprosto nepředvídatelný výsledek mutace, ať už ji aplikujeme na jakýkoliv uzel, viz. kapitoly 6.1, 6.3, 7.1. Výsledek použití tohoto typu pravidla je možno označit za **obecnou mutaci**. V případě, že chceme kontrolovat typ mutace, je nutné definovat gramatiku tak, aby byl překlad $\beta_1 \rightarrow \alpha \rightarrow \beta_2$ do jisté míry *předvídatelný*, (viz. kapitola 7.3.1).



sloupec A: náhodně generované hodnoty během mutace, **sloupec B**: výběr pravidla z analytické, nebo generativní gramatiky, **sloupec C**: stav mutace uzlu. Červené symboly značí neterminály, nebo terminály, které budou v dalším kroku nahrazeny, tučně jsou právě nahrazené. Nad šipkama je zobrazen výběr pravidel zkracující mutace, pod šipkama je zobrazen výběr pravidel prodlužující mutace, **sloupec F**: pořadí operace, **sloupec G1**: strom funkce před použitím operátoru mutace. Uzel k mutaci je vyznačen oranžovou barvou, **sloupec G2**: strom funkce po použití operátoru mutace. Zmutovaný uzel je vyznačen oranžově.

Obrázek 35: Strukturální mutace

Zkracující mutace: *(shortening mutation)*, viz. obrázek 35, použitím tohoto operátoru vznikne ze stromu definovaného ve *sloupci G1* použitím pravidel analytické a generativní gramatiky mutovaný strom, zobrazený ve *sloupci G2*. Pravidla použitá k překladu jsou zobrazeny nad šipkama, v horní části *sloupce C*. Šedý řádek ve *sloupcích A, B* značí použití pravidla analytické gramatiky pro přepsání terminálního symbolu, jenž je označen ve *sloupci G1* červenou šipkou. Je vidět, že terminální symbol strukturálního uzlu $*$ byl použitím pravidla analytické gramatiky přepsán na neterminální symbol $\langle var \rangle$ a z něj se poté použitím pravidla generativní gramatiky vytvoří terminální symbol E_k. Výsledný uzel je ve *sloupci G2* označen červenou šipkou. Je možné si všimnout, že došlo ke zkrácení stromu a zahození původních následníků uzlu $*$.

Aby bylo možno provést zkracující mutaci, je nutno definovat pravidla gramatiky, které zajistí zkrácení stromu. Pravidla P^{-1} analytické gramatiky G^{-1} lze definovat ve tvaru uvedeném v tabulce 10. Z uvedené tabulky je možno vyčíst, že na levé straně pře-

$U+ \mid U-$	$::=$	$\langle var \rangle \mid \langle const \rangle$
$+ \mid - \mid * \mid /$	$::=$	$\langle var \rangle \mid \langle const \rangle$

Tabulka 10: Pravidla analytické gramatiky pro zkracující strukturální mutaci

pisovacího pravidla stojí terminální symboly *jednoargumentových* a *dvouargumentových funkcí*. To jsou takové symboly, jejichž uzel má minimálně jednoho následníka a podílejí se tedy výhradně na větvení stromu. Na pravé straně těchto pravidel jsou neterminální symboly $\langle var \rangle$ a $\langle const \rangle$, které lze pomocí generativní gramatiky uvedené v tabulce 5

(strana 45) přepsat na terminální symboly, jejichž počet následníků je nulový. Nulový počet argumentů způsobí zahození celého podstromu mutovaného uzlu. Tento přepis tím pádem způsobí snížení počtu uzlů stromu. Rozsah změn struktury stromu závisí na uzlu který je mutován. Čím blíže kořenu stromu je mutace provedena, tím větší bude výsledná změna struktury jedince. Mutace kořene stromu způsobí úplnou změnu fenotypu. Tímto jsou si GE se zpětným zpracováním pravidel a TE vzájemně podobné, (viz. kapitola 6.5).

Prodlužující mutace: *(extending mutation)*, viz. obrázek 35, použitím tohoto operátoru vznikne z původního stromu definovaného ve *sloupci G2*, použitím pravidel definovaných ve spodní části *sloupce C*, strom jedince definovaný ve *sloupci G1*. Uzel, který byl vybrán k mutaci, je ve *sloupci G2* označen červenou šipkou. Strom (podstrom), který vznikl postupným přepisem uzlu E_k, aplikováním pravidel z G^{-1} a G je ve *sloupci G1* zvýrazněn oranžovou barvou. Tento nově vygenerovaný podstrom je napojen na strukturální uzel jeho předchůdce $+$. Strom je vygenerován postupem uvedeným v kapitole 7.1. Použitá pravidla k překladu jsou ve *sloupcích A, B* odlišena šedou a modrou barvou. Šedá barva označuje pravidlo, které bylo vybráno z množiny pravidel gramatiky G^{-1} a modrou barvou jsou označena pravidla, která byla vybrána z gramatiky G. Ve *sloupci F* jsou operace označeny čísly. V prvním kroku je tedy terminální symbol, přepsán na neterminální, použitím pravidla P^{-1}, dále je tento neterminál přepisován pravidly gramatiky G na terminální symboly, až vznikne výsledný podstrom vyjadřující funkci $5 * E_{k-1}$. Je vidět, že původně jeden uzel byl nahrazen stromem, který má uzly celkem tři. To znamená, že došlo k rozšíření stromu a tedy k prodlužující strukturální mutaci.

Aby bylo možné provést prodlužující mutaci, je nutno definovat pravidla gramatiky, které zajistí prodloužení stromu. Pravidla P^{-1} analytické gramatiky G^{-1} je možné definovat ve tvaru uvedeném v tabulce 11. Z uvedené tabulky je možno vyčíst, že na levé straně

$U_{k-1} \mid U_{k-2} \mid U_{k-3} \mid U_{k-4} \mid U_{k-5}$	$::=$	$\langle mfnc1 \rangle \mid \langle mfnc2 \rangle$
$E_k \mid E_{k-1} \mid E_{k-2} \mid E_{k-3} \mid E_{k-4} \mid E_{k-5}$	$::=$	$\langle mfnc1 \rangle \mid \langle mfnc2 \rangle$
$dE_k \mid dE_{k-1} \mid dE_{k-2} \mid dE_{k-3} \mid dE_{k-4}$	$::=$	$\langle mfnc1 \rangle \mid \langle mfnc2 \rangle$
$U+ \mid U-$	$::=$	$\langle mfnc2 \rangle$

Tabulka 11: Pravidla analytické gramatiky pro prodlužující strukturální mutaci

přepisovacího pravidla stojí terminální symboly proměnných. To jsou takové symboly, které nemají žádného následníka a tvoří tedy výhradně listy stromu. Kromě těchto terminálních symbolů proměnných jsou dále na levé straně uvedeny *unární funkce* $+$ a $-$, které mají dle pravidel z tabulky 5 (strana 45) jednoho následníka. Na pravé straně těchto uvedených pravidel leží nově definované neterminální symboly $\langle mfnc1 \rangle$ a $\langle mfnc2 \rangle$, sloužící pouze pro účely prodlužující strukturální mutace. Pro tyto neterminály je nutné přidat do gramatiky G pravidla, které je mohou přepsat zpět na terminální symbol, resp. symboly. Rozšiřující přepisovací pravidla gramatiky G lze zapsat ve tvaru uvedeném v tabulce 12.

$\langle mfnc1 \rangle$	$::=$	$\langle fnc1 \rangle \langle var \rangle \mid \langle fnc1 \rangle \langle const \rangle$
$\langle mfnc2 \rangle$	$::=$	$\langle fnc2 \rangle \ \langle var \rangle \langle var \rangle \mid \langle fnc2 \rangle \langle const \rangle \langle var \rangle$

Tabulka 12: Rozšiřující pravidla generativní gramatiky

7.3.3 Pravděpodobnost výběru typu mutace

Pravděpodobnost strukturální a nestrukturální mutace se může v průběhu evoluce měnit. Změna pravděpodobnosti může být zapotřebí v případě, kdy použitím metody *zkracující strukturální mutace*, je možno docílit snížení počtu uzlů ve stromu jedince, (viz. kapitola 7.3.2). Pravděpodobnost použití zkracující strukturální mutace je možno ovlivnit použitím vztahu pro přepočet pravděpodobnosti mutace, (Eq. 7.3):

$$Pc(zm) = \begin{cases} P(zm) & \text{pro } N_c \leq N_m \\ P(zm) \cdot \left(1 + \frac{N_c - M_c}{M_c}\right) & \text{pro } N_c > N_m \end{cases} \tag{7.3}$$

Kde: $Pc(zm)$ je přepočítaná pravděpodobnost použití zkracující strukturální mutace,
$P(zm)$ je pravděpodobnost použití zkracující strukturální mutace,
N_c je počet uzlů obsažených ve stromu,
N_m je hranice omezení počtu uzlů

7.4 Pravděpodnost výběru pravidla

Pro jednoduchost se až dosud ve všech uváděných příkladech předpokládalo, že pravděpodobnost výběru pravidla z množiny pravidel je pro všechny pravidla stejná. Ve skutečnosti tomu tak být nemusí a pravděpodobnost výběru může být pro každé pravidlo jiná a může se dokonce během překladu libovolně měnit. Tímto se TE výrazně liší od všech známých forem evolučních algoritmů založených na gramatice, jelikož změna pravděpodobnosti výběru pravidla by v GE způsobila, že ze stejného genotypu by se vždy vygeneroval jiný fenotyp, což by v nejhorších případech vedlo až k havárii programu[7], viz. např. [30, 29, 31, 32, 35] a další ....

Prakticky lze výběr pravidla zkonstruovat např. pomocí ruletového kola, (viz. kapitola 3.10.1), kdy po přiřazení různé pravděpodobnosti každému z pravidel, se jednotlivým pravidlům přiřadí různý úsek kruhové výseče podle jeho pravděpodobnosti a náhodným generováním čísla $r \in \langle 0, 1 \rangle$ se provede výběr pravidla, když platí vztah (7.4) [16]:

$$\sum_{j=1}^{i-1} p_j < r \leq \sum_{j=1}^{i} p_j, \quad i \in \{1, \ldots, N\} \tag{7.4}$$

Kde: N je počet pravidel,
p_j je pravděpodobnost výběru pravidla.

Nastavení pravděpodobnosti výběru pravidla může být užitečné v případech, kdy je zapotřebí omezit délku stromu a zajistit tak sníženou pravděpodobnost výběru pravidel, které zajišťují větvení stromu. Tato pravděpodobnost se může postupně snižovat s rostoucím počtem uzlů ve stromu. Přepočet pravděpodobnosti výběru pravidel lze realizovat použitím vztahů (7.5), (7.6):

$$Pc(s)_j = \begin{cases} P(s)_j & \text{pro } N_c \leq N_m \\ P(s)_j \left(1 - \frac{N_c - M_c}{M_c}\right) & \text{pro } N_c > N_m \end{cases} \tag{7.5}$$

[7] Murphyho zákon: Může-li se něco pokazit, pokazí se to

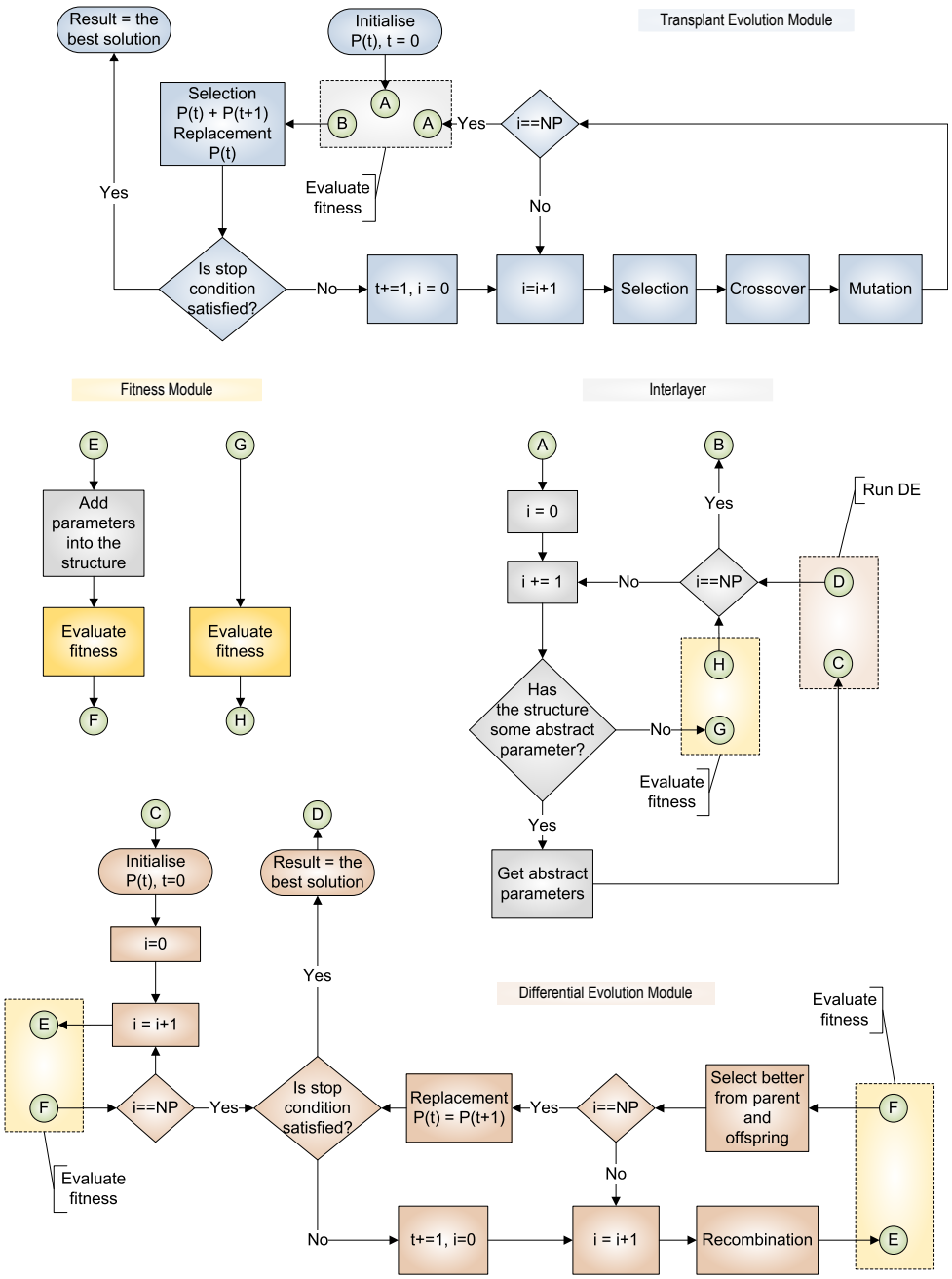
$$Pc(n)_i = \begin{cases} P(n)_i & \text{pro } N_c \leq N_m \\ P(n)_i + \dfrac{1}{N_j}\left(1 - \sum_{j=1}^{N_j} Pc(s)_j + \sum_{i=1}^{N_i} P(n)_i\right) & \text{pro } N_c > N_m \end{cases} \tag{7.6}$$

Kde: $\mathbf{Pc(s)_j}$ vyjadřuje novou hodnotu pravděpodobnosti výběru strukturálního pravidla s indexem j, $\mathbf{P(s)_j}$ vyjadřuje přednastavenou hodnotu pravděpodobnosti výběru strukturálního pravidla s indexem j, $\mathbf{N_c}$ je počet uzlů obsažených ve stromu, $\mathbf{N_m}$ je hranice omezení počtu uzlů, $\mathbf{Pc(n)_i}$ vyjadřuje novou hodnotu pravděpodobnosti výběru i-tého nestrukturálního pravidla, $\mathbf{P(n)_i}$ označuje přednastavenou hodnotu pravděpodobnosti výběru i-tého nestrukturálního pravidla, $\mathbf{N_j}$ je počet strukturálních pravidel, $\mathbf{N_i}$ je počet nestrukturálních pravidel.

8 Dvoufázová transplantační evoluce

Dvoufázová transplantační evoluce (TLTE), *(Two Level Transplant Evolution)* je novým typem evolučního algoritmu, který vznikl kombinací algoritmu *transplantační evoluce*, (viz. kapitola 7) a *diferenciální evoluce* (viz. kapitola 4), případně jakéhokoliv jiného heuristického algoritmu pro optimalizaci numerických parametrů [43, 45, 42]. Spojení dvou evolučních algoritmů má za cíl odstranit obtížné generování parametrů z oboru reálných čísel $\mathbb{R}$, které již bylo diskutováno v kapitole 6.7.

Původně vzniklo spojení dvou evolučních algoritmů v kombinaci gramatická evoluce



Obrázek 36: Diagram dvoufázové transplantační evoluce

+ diferenciální evoluce, které bylo diskutováno v publikacích [29, 35, 32]. Důvodem ke vzniku *dvoufázové gramaticko–diferenciální evoluce* bylo velké množství použitých genů v chromozómu jedince GE pro vygenerování čísla a z toho vyplývající nestabilita operátorů křížení a mutace. Tyto problémy sice *transplantační evoluce* částečně odstraňuje z důvodu reprezentace jedince pouze fenotypem, (viz. kapitola 7), nicméně neřeší tento problém úplně.

Principem činnosti TLTE je optimalizace řešení ve dvou fázích. V **první fázi** probíhá optimalizace struktury řešení pomocí TE, poté je tato struktura předána do **druhé fáze**, ve které jsou řešeny její neznámé parametry pomocí DE, GA, ... *Diferenciální evoluce* byla vybrána z důvodu rychlosti a schopnosti nalézt řešení, jelikož optimalizace neznámých parametrů rovnice regulátoru je velice náročný a zdlouhavý proces, během kterého dochází k velkému počtu pokusů nalézt optimální parametry, pro které mnohdy ani nevíme v jakém intervalu se tyto parametry nacházejí, viz. kapitoly 11.2.1, 12. Výhodou DE je, že nepoužívá žádné kódování chromozómů a nedochází tedy ke zkreslování prohledávaného prostoru. Náročnost přechodu mezi jednotlivými reálnými čísly je úměrná vzdálenosti těchto dvou čísel. Kromě DE byl v této práci testován také GA, (viz. kapitola 8.4), ale z důvodu větší časové náročnosti mu nebyla věnována taková pozornost, jako právě DE.

Pro zajištění **generování abstraktních numerických parametrů**, je zapotřebí upravit dosud definovanou gramatiku G, viz. tabulky 5 (s.45), 9 (s.63), 10 (s.65), 11 (s.66), 12 (s.66). Do této gramatiky je potřeba vložit nové pravidlo, které může být ve tvaru definovaném v tabulce 13. Terminální symbol pravé strany tohoto pravidla je abstraktním numerickým parametrem.

$\langle var \rangle$	::=	num

Tabulka 13: Rozšiřující pravidlo generativní gramatiky

Za účelem mutace abstraktních numerických parametrů je potřeba do analytické gramatiky G^{-1} vložit pravidlo, které zajistí přepis terminálního symbolu *num*. Takové pravidlo je uvedeno v tabulce 14.

num	::=	$\langle var \rangle$

Tabulka 14: Rozšiřující pravidlo analytické gramatiky

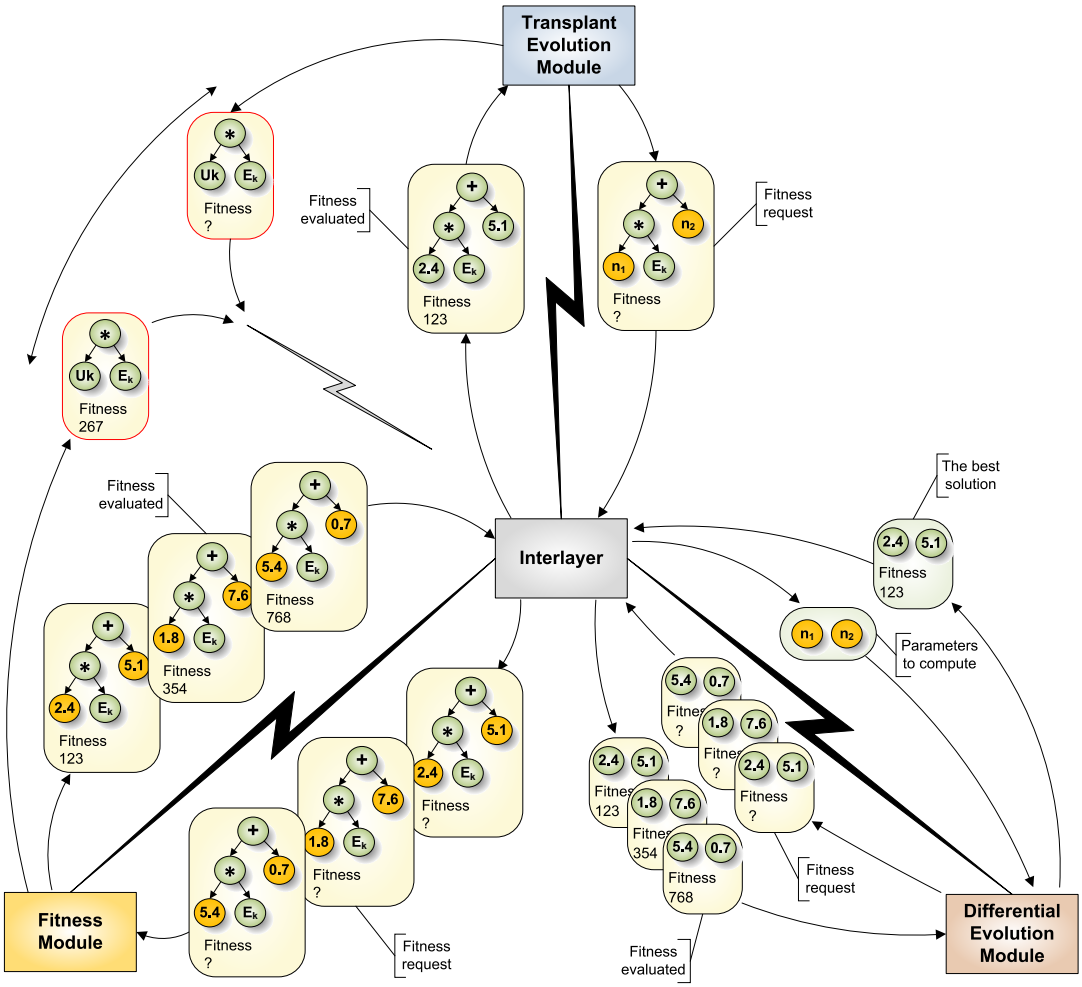
Z tabulky 13 je možné vyčíst, že do gramatiky bylo vloženo pravidlo, které dokáže přepsat neterminál $\langle var \rangle$ na terminální symbol *num*. Těchto *stejných* neterminálních symbolů, ovšem může být ve stromu obsaženo i více a každý označuje jinou proměnnou. Z toho důvodu byla implementována metoda, která vykonává **značkování abstraktních proměnných** po inicializaci stromu na počátku evoluce, křížení, nebo mutaci. Značkování probíhá tak, že se projde celý strom a každé abstraktní proměnné se přiřadí jedinečné číslo, které potom slouží k identifikaci konkrétní proměnné v případě čtení těchto parametrů, nebo nastavení konkrétních hodnot.

Nevýhodou spojení dvou evolučních algoritmů je poměrně náročná implementace. Další nevýhodou je velká **časová složitost** pro nalezení optimálního řešení. Tato časová složitost je způsobena tím, že pro každého jedince TE, jenž obsahuje alespoň jeden abstraktní

parametr, je spuštěna druhá fáze evoluce, která má za úkol tyto parametry optimalizovat. V případě optimalizace struktury a parametrů regulátoru se jedná o desítky hodin až dnů. Časová složitost dvoufázového algoritmu je v nejlepším případě (když jedinec neobsahuje žádné abstraktní parametry) $O(N)$, v nejhorším případě $O(N^2)$. Vzhledem k tomu, že dvoufázová transplantační evoluce je navržena pro optimalizaci struktury i parametrů, jedinec obsahuje abstraktní parametry téměř vždy. Z toho vyplývá časová složitost $O(N^2)$. V případě použití TLTE pro optimalizaci regulátoru závisí celkový čas řešení na počtu jedinců v první a druhé vrstvě a na délce regulačního intervalu.

8.1 Architektura

Z pohledu implementace je nejnáročnějším procesem propojení dvou evolučních algoritmů. Z důvodu flexibility bylo zvoleno propojení tří samostatných výpočetních modulů, které jsou pro okolí černé skříňky *(black boxes)* a komunikují s okolím přes *komunikační interface*. Díky tomuto návrhu lze moduly použít i samostatně pro libovolný jiný optimalizační problém, případně daný modul jednoduše zaměnit za jiný (DE – GA, ...).



Obrázek 37: Datový tok ve dvoufázové transplantační evoluci s oddělenými moduly

Architektura celého řešení je ukázána na obrázcích 36 a 37. Pro snadnější orientaci jsou

si obrázky barevně ekvivalentní. **Modul transplantační evoluce**, (viz. kapitola 8.2), je zvýrazněn *modře*, **modul diferenciální evoluce**, (viz. kapitola 8.3), je zvýrazněn *hnědou*, **modul výpočtu fitness**, (viz. kapitola 8.5), je zvýrazněn *žlutě* a **mezivrstva**, (viz. kapitola 8.6) je zvýrazněna *šedou*.

Datový tok a vazby mezi jednotlivými moduly jsou ukázány na obrázku 37.

8.2 Modul transplantační evoluce

Modul transplantační evoluce (MTE) *(Transplant Evolution Module)*, viz. obrázek 36, 37, zajišťuje optimalizaci struktury řešení. Princip činnosti byl popsán v kapitole (viz. kapitola 7). Jeho náplní je vytvářet populace jedinců, použitím gramatiky G, G^{-1}. Tento modul, i když je toho schopen, neprovádí optimalizaci numerických parametrů ve struktuře řešení. Optimalizace parametrů je zajištěna moduly MDE, (MGA), viz. kapitoly 8.3, 8.4. Jako prostředník pro komunikaci mezi těmito vrstvami slouží *mezivrstva*, (viz. kapitola 8.6), která zajišťuje přípravu dat v požadovaném tvaru pro jednotlivé vrstvy. **Vstupním parametrem** tohoto modulu je gramatika G a G^{-1} podle níž se tvoří populace jedinců. **Výstupním parametrem** této vrstvy je kompletně optimalizované řešení, které zahrnuje jak vhodnou strukturu, tak i její parametry.

8.3 Modul diferenciální evoluce

Modul diferenciální evoluce (MDE) *(Differential Evolution Module)*, viz. obrázek 36, 37, zajišťuje optimalizaci abstraktních parametrů ve struktuře jedince. Princip funkce byl popsán v kapitole 4. Tento modul je volán *mezivrstvou*, (viz. kapitola 8.6), která mu jako jeden ze **vstupních parametrů** předá vektor proměnných $\vec{num} = (num_1, \ldots, num_n)$. Počet parametrů tohoto vektoru je roven počtu proměnných, které obsahovala struktura řešení, jenž přišla z MTE. **Výstupní parametr** této vrstvy je optimalizovaný vektor parametrů, který je předán do *mezivrstvy*.

8.4 Modul genetického algoritmu

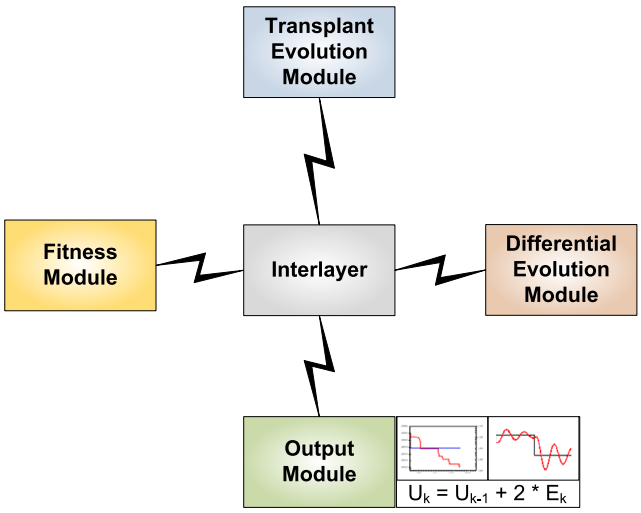
Modul genetického algoritmu (MGA) *(Genetic Algorithm Module)* vykonává stejnou funkci jako MDE. Tyto moduly jsou vzájemně zaměnitelné. Vzhledem k tomu, že GA používá binární kódování, bylo potřeba vyřešit kódování reálných čísel do binární podoby. Nativní uložení reálných čísel v PC se řídí normou IEEE 754 a není z toho důvodu pro GA použitelné, (viz. kapitola 14.1), tabulka 27. Kódování reálných čísel do binární podoby bylo tedy zajištěno přepočtem reálných čisel s požadovanou přesností do celočíselného tvaru a poté kódováno do binárního tvaru dle vztahů uvedených v kapitole 3.6.1.

8.5 Modul výpočtu fitness

Modul modul výpočtu fitness (FM) *(Fitness Module)*, viz. obrázek 36, 37, zajišťuje hodnocení modelů. Modelem je struktura řešení, včetně parametrů. Tento model, společně s regulovanou soustavou je **vstupním** parametrem simulace. Pro zadaný model regulátoru a soustavu, spustí simulaci regulace. Průběh regulačního pochodu poté ohodnotí dle zadaných parametrů a ohodnocený model poté vrátí do *mezivrstvy*.

8.6 Mezivrstva

Mezivrstva *(Interlayer)*, viz. obrázek 36, 37, je sice zmíněná jako poslední, nicméně z celé architektury je nejdůležitější. Mezivrstva je hlavním modulem, který drží instance všech výpočetních modulů, mezi kterými zajišťuje komunikaci přes komunikační *interface*. Tento modul má jako jediný představu o tom co se má optimalizovat. Řídí vstupní parametry všech modulů (MTE, MDE, MGA, FM), kterými jsou např. velikosti populací jedinců TE, DE, GA, počet generací evoluce, volba selekčních metod, nastavení pravděpodobností křížení a mutací, atd. Kromě jiného navíc zprostředkovává komunikaci s **výstupním modulem**, (viz. kapitola 8.7), zajišťujícím zobrazování výstupů evolucí, simulací regulace, apod.



Obrázek 38: Datový tok zobrazení výstupů

Jejím nejdůležitějším úkolem je příprava dat v žádaném tvaru během komunikace mezi moduly, např. při požadavku na výpočet fitness, viz. obrázek 37. Z tohoto pohledu se jeho úkoly dají rozdělit tímto způsobem:

1. vytvořit vektor symbolických proměnných pro MDE (MGA) během komunikace s MTE.
2. přiřadit abstraktním proměnným ve stromu jedince konkrétní proměnné, během komunikace MDE (MGA) a FM.
3. přiřadit abstraktním proměnným ve stromu jedince konkrétní proměnné, po dokončení optimalizace parametrů, během komunikace MDE (MGA) a MTE.
4. Předat úplný model řešení (struktura + parametry) FM, během požadavku MTE na výpočet kvality jedince, který neobsahuje žádný abstraktní parametr.

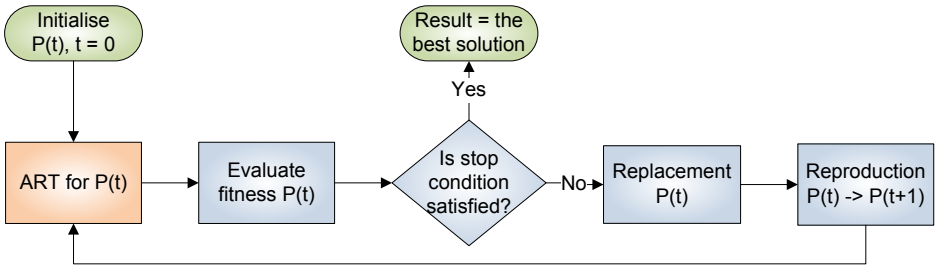
8.7 Výstupní modul

Výstupní modul (OM) *(Output Module)*, viz. obrázek 38, slouží pro zobrazení výstupů průběhů evoluce, simulace regulace, zobrazování výsledků, apod. Kromě jiného umí tento modul částečně ovládat chování evoluce, simulace, apod. tím, že dokáže své instanci nařídit

pozastavení / pokračování / zastavení běhu, nařídí *načtení nového nastavení*, kterým je např. změna parametrů regulace, změna parametrů evoluce, apod.

8.8 Aritmetická redukce stromu

Aritmetická redukce stromu (ART), *(Arithmetic Tree Reducing)* je metodou, která vyhledává ve stromu jedince takové části, které lze bezztrátově aritmeticky redukovat. V případě, že nalezne takovou část stromu, kterou má obsaženou ve *slovníku*, provede její redukci. Metoda vznikla právě díky TE, jelikož se nemusí provádět obtížná transformace numericky vyjádřeného chromozómu, jako v případě GE se zpětným zpracováním pravidel a značkováním kodonů, (viz. kapitola 6.3), kde by tato metoda šla využít jen s velkými obtížemi a v případě klasické GE by dokonce nešla použít vůbec, (viz. kapitola 6).



Obrázek 39: Začlenění ART do TE (TLTE)

Metoda je sice použitelná v kombinaci s metodou TE, (viz. kapitola 7), ale její síla se projeví právě ve spojení s TLTE, kde je možno využít symbolických konstant v podobě terminálu num.

P1	U+(e)	::=	e
	U-(U-(e))	::=	e
	U-(U+(e)) | U+(U-(e))	::=	U-(e)
P2	$f_{2m}(num, num)$ | $f_{2s}(num, num)$	::=	num
P3	$+(a_1, a_2)$	::=	$*(num, a)$
	$-(a_1, a_2)$	::=	num
	$/(a_1, a_2)$	::=	num

první sloupec: (P1, P2, P3) slouží pro označení sad pravidel a vyjadřují náročnost dané operace (P1 – nejjednodušší, P5 – nejnáročnější)

Tabulka 15: Pravidla pro ART (základní operace)

Slovník definuje pravidla pro redukční operace se stromem, jenž jsou definována ve tvaru uvedeném v tabulkách 15, 16. Pravidla v uvedených tabulkách jsou definována prefixovým zápisem terminálních symbolů, které představují určitý tvar stromu. Význam jednotlivých symbolů je následující:
num je terminální symbol reprezentující symbolickou konstantu, která bude optimalizována druhou fází evoluce (MDE, MGA), viz. kapitoly 8.3, 8.4,
f_{2s}, f_{2m}, a, a_1, a_2, $pair(a)$ jsou zástupné symboly terminálních symbolů,

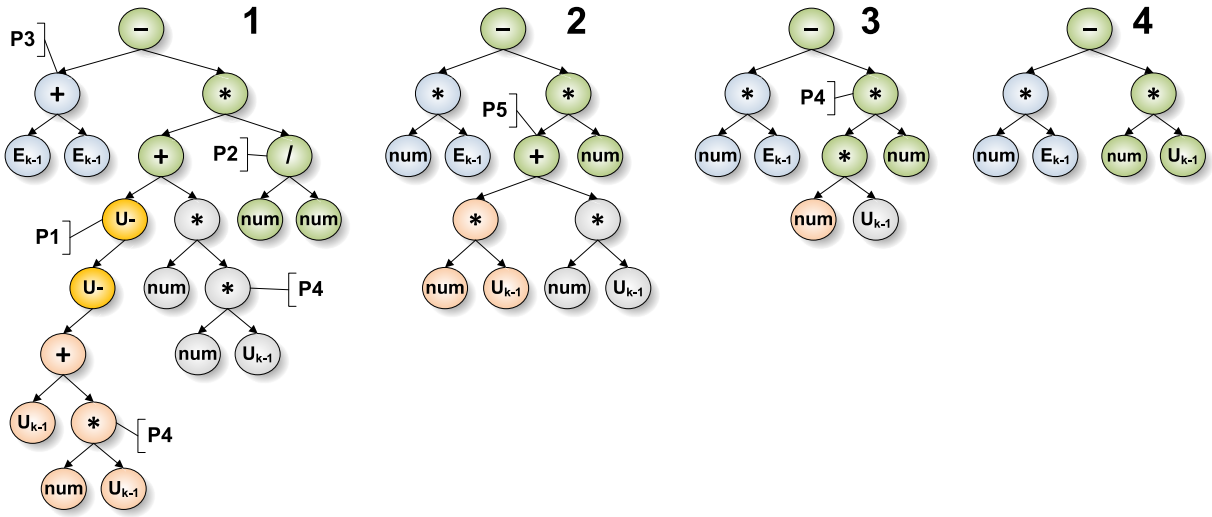
P4	$f_{2m}(num,\, pair(a)) \mid f_{2m}(pair(a),\, num)$	$::=$	$*(num,\, a)$
	$f_{2s}(num,\, pair(a)) \mid f_{2s}(pair(a),\, num)$	$::=$	$*(num,\, a)$
P5	$f_{2s}(pair(a),\, pair(a))$	$::=$	$*(num,\, a)$

první sloupec: (P4, P5) slouží pro označení sad pravidel a vyjadřují náročnost dané operace (P1 – nejjednodušší, P5 – nejnáročnější)

Tabulka 16: Pravidla pro ART (pokročilé operace)

$f_{2s} \in \Sigma^{2s}$ a $f_{2m} \in \Sigma^{2m}$ definují množiny terminálních symbolů, reprezentované dvouargumentovými funkcemi,
$\Sigma^{2s} \in \{+, -\}$,
$\Sigma^{2m} \in \{*, /\}$,
$(a, a_1, a_2) \in \Sigma^{rv}$, pro které platí $a = a_1 = a_2$, $e \in \Sigma^{re}$. Obecně platí vztah $\Sigma^{rv} \subset \Sigma^{re} \subset \Sigma$.
$\Sigma^{rv} = \{E_k, \ldots, E_{k-5}, U_k, \ldots, U_{k-5}, dE_k, \ldots, dE_{k-4},\}$ je množina terminálních symbolů *proměnných*, které jsou určeny k redukci.
$\Sigma^{re} = \{+, -, *, /\} \cup \Sigma^{rv}$ je množina terminálních symbolů, které jsou určeny k redukci.
$pair(a)$ je definován jako $f_{2m}(num, a) \mid f_{2m}(a, num)$.

Pravidla v uvedených tabulkách dovolují přepsat sekvenci terminálních symbolů na levé straně pravidla, jinou sekvencí terminálních symbolů, uvedených na pravé straně pravidla. Symboly uvedené v tabulkách, jsou významově totožné s gramatikou vyjádřenou v *BNF* tvaru, (viz. kapitola 5.2).



P1 – P5: určují sadu pravidel podle které bude provedena redukce, (tabulky 15, 16).

Obrázek 40: ART – vybrané typy možných redukcí

V tabulce 15 jsou definovány pravidla pro základní (jednoduché) operace se stromem. Tyto pravidla rozšiřuje tabulka 16, ve které jsou definovány pravidla pro pokročilejší operace se strukturou stromu. Tabulky byly rozděleny podle hloubky stromu se kterou pracují. V první tabulce, viz. tabulka 15, jsou definovány pravidla, jenž pracují se dvěmi úrovněmi a v tabulce 16 jsou uvedeny pravidla, které pracují se třemi úrovněmi stromu. V první jmenované tabulce jsou sice zařazeny i pravidla, které pracují ve třech úrovních,

ale to se týká pravidel, které nevětví strom a tím pádem jsou zařazeny do jednoduchých operací. Jedná se o operace s unárními funkcemi U-, U+. V první sloupci v obou tabulkách je označení skupiny pravidel. Pravidla jsou v tabulkách řazeny od nejjednodušších, po nejnáročnější. Některé uvedené pravidla se mohou zdát nesmyslná, ale je třeba si uvědomit, že symbolická proměnná *num* je optimalizována druhou fází evoluce.

Princip metody spočívá v několikanásobném průchodu stromem a jeho postupné redukci, dle zadaného slovníku *(pravidel)*. Některé z možných typů redukcí jsou ukázány na obrázku 40.

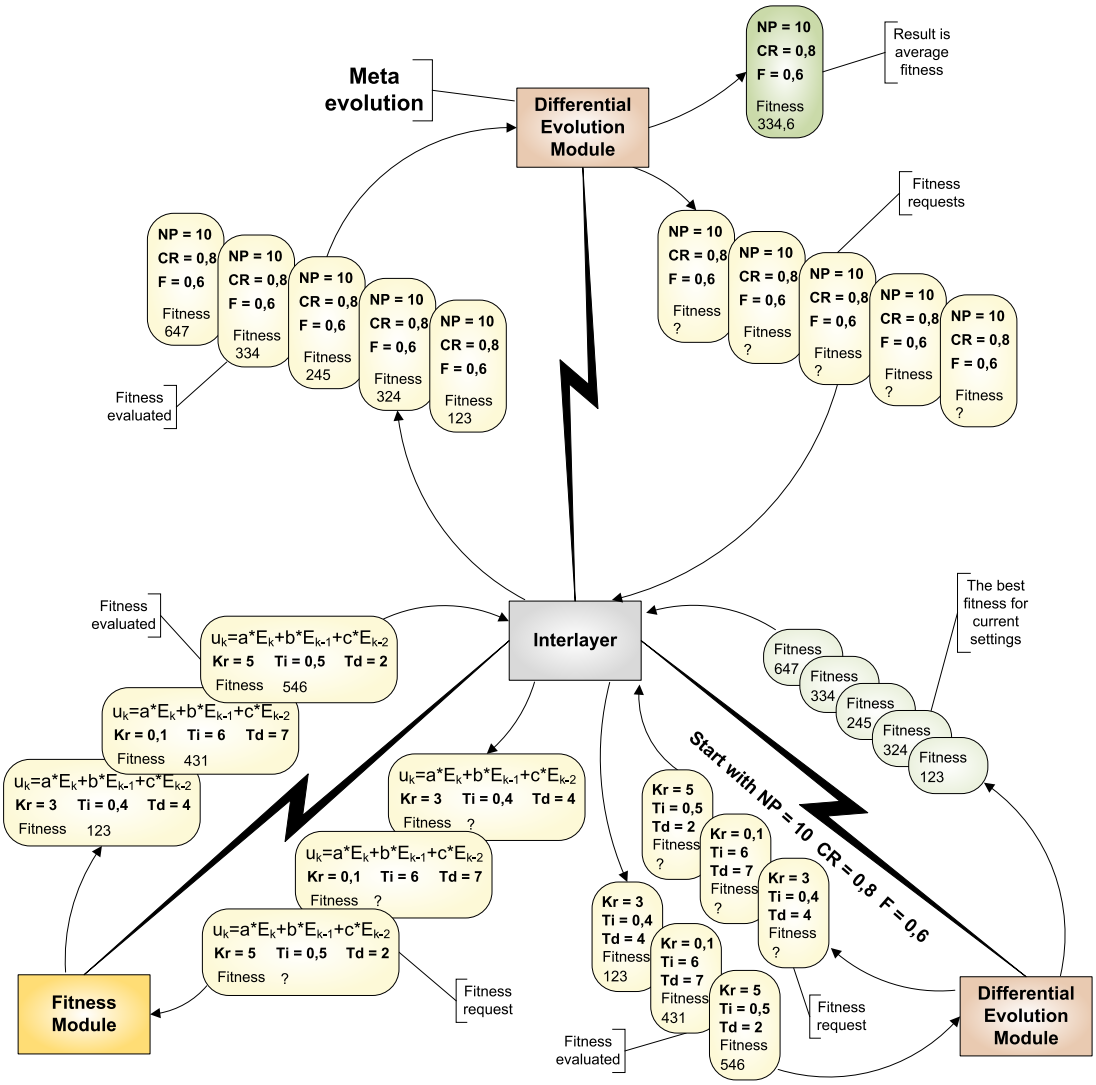
Na tomto obrázku jsou čtyři řešení ve formě stromu. Každý strom je očíslovaný. Výchozí stav stromu je na obrázku pod číslem 1, cílový stav je strom číslo 4. Redukce každé části jsou barevně odlišeny a kořenový uzel redukce je označen odkazem s číslem (P1 – P5). Tyto odkazy označují sadu pravidel, která byla použita k redukci, viz. tabulky 15 a 16.

Je vidět, že výchozí strom je složen z 21 uzlů a cílový ze 7 uzlů. Prakticky to neznamená jen kratší výsledné řešení, ale také přibližně trojnásobné *zrychlení výpočtu* během procházení stromu.

Metoda ART je volána po každém vytvoření jedince, během inicializace i během reprodukce, po volání metod křížení a mutace, viz. obrázek 39.

9 Meta–evoluce

V kapitole 11.2.1 je ukázáno, že optimalizace symbolických konstant rovnice obecného regulátoru je velice zdlouhavý proces. Ve snaze o zefektivnění celého evolučního procesu byla vytvořena **meta–evoluce**, která měla za cíl optimalizovat nastavení parametrů DE z hlediska rychlosti a kvality nalezeného řešení. Cílem bylo nalézt optimální nastavení parametrů *NP, CR, F*, (viz. kapitola 4).



Obrázek 41: Datový tok meta–evoluce

Vzhledem k velké časové náročnosti byla tato optimalizace použita pouze dvakrát. V prvním případě se jednalo o optimalizaci parametrů DE (*NP, CR, F*), která optimalizovala parametry *PSD regulátoru* (K_R, T_I, T_D). Výsledkem tohoto evolučního procesu je optimální nastavení $NP = 9$, $CR = 0{,}58$, $F = 1{,}07$, pro zvolené schéma *DE/Rand/1*. Jako optimalizační nástroj byla opět zvolena diferenciální evoluce s nastavením $NP = 10$, $CR = 0{,}5$, $F = 0{,}5$ a schématem *DE/Rand/1*.

Podruhé byla meta–evoluce použita pro optimalizaci parametrů DE, která optimalizovala symbolické proměnné v rovnici obecného regulátoru, jenž byla zadána ve tvaru $u_k =$

$u_{k-1}+num\cdot E_k+num\cdot E_{k-1}+num\cdot E_{k-2}$. Výsledkem tohoto optimalizačního procesu bylo nastavení řídících proměnných DE, $NP = 13, CR = 0{,}74, F = 0{,}72$. Jako optimalizační nástroj byla v tomto případě použita DE s nastavením $NP = 9,\ CR = 0{,}58,\ F = 1{,}07$.

Každý jedinec meta–evoluce obsahuje řešení „optimálního“ nastavení podřízené vrstvy DE. Aby bylo možné jednotlivé řešení meta–evoluce hodnotit, je nutné provést optimalizační proces s tímto nastavením. Optimalizační proces pomocí evolučního algoritmu je heuristický proces a dopadne vždy s jiným výsledkem. Pro zvýšení objektivity ohodnocení řešení, byla optimalizace s jedním nastavením spuštěna pětkrát a výsledné fitness jedince bylo počítáno jako průměrná hodnota z těchto optimalizačních procesů.

Schéma datového toku meta–evoluce je ukázáno na obrázku 41, kde je možno si všimnout dvou MDE. Ve skutečnosti se jedná o dvě různé instance jednoho modulu MDE, z nichž každá dělá něco jiného. Jedna instance MDE, označená jako *meta–evoluce*, optimalizuje hodnoty parametrů druhé instance MDE, která optimalizuje parametry PSD regulátoru. Popisy jednotlivých modulů jsou uvedeny v kapitole 8.1.

10 Regulační obvody

Regulační obvod je systém, který je tvořen regulovanou soustavou a regulátorem [5, 40]. Regulátor na základě rozdílu mezi žádanou a regulovanou veličinou nastaví akční veličinu tak, aby byl splněn cíl regulace, tzn. $y \to w$ resp. $e \to 0$.

10.1 Historie průmyslové regulace

První známý regulační mechanismus se datuje do roku 270 př.n.l., kdy Řek Ktésibios vynalezl plovákový regulátor vodních hodin. Funkcí tohoto mechanického regulátoru bylo udržet vodní hladinu v rezervoáru na konstantní výšce. Tato konstantní výška hladiny udržovala odtok u dna 1. nádrže, který naplňoval 2. rezervoár konstantní rychlostí. Výška hladiny ve 2. rezervoáru tedy byla lineárně závislá na čase.

Za počátek průmyslového využití regulace je považován počátek průmyslové revoluce (konec 18. - počátek 19. století). V této době vynalezl James Watt (1736 – 1819) odstředivý regulátor otáček pro parní stroj a v roce 1783 prodal 1. regulátor pro průmyslové využití.

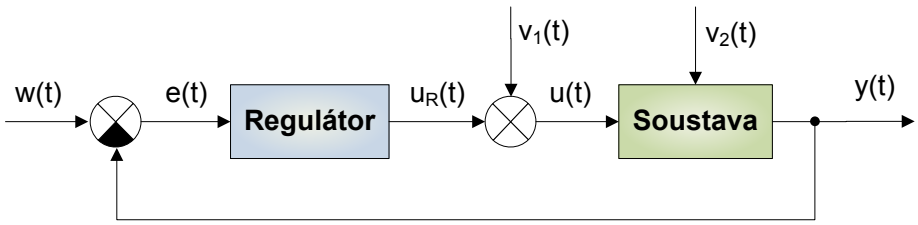
Parní stroje s Wattovým odstředivým regulátorem se však dostávaly do nestabilních stavů. James Clerk Maxwell (1831 - 1879) jako první linearizoval pohybové diferenciální rovnice tohoto systému a analyzoval jeho stabilitu z charakteristické rovnice. Studoval vliv parametrů systému na jeho stabilitu a dokázal, že jsou-li parametry systému voleny tak, aby kořeny charakteristické rovnice ležely v levé polorovině komplexní roviny, je systém stabilní. Položil tak základy teorie řídicích systémů.

V první polovině 20. století došlo k zásadnímu zvratu v teorii řízení. Do popředí se dostala transformace z časové do frekvenční oblasti, kterou o 150 let dříve zavedli Pierre Simon de Laplace (1749 – 1827) a Jean Baptiste Joseph Fourier (1768 – 1830). Frekvenční charakteristiku využili Harry Nyquist (1889 - 1976) a Hendrik Wade Bode (1905 – 1982) při zavedení záporné zpětné vazby k potlačení šumů zesilovačů ve sdělovacích sítích. Došlo k prudkému vývoji frekvenčních metod syntézy regulačních obvodů, které umožnily řešit otázky stability regulačních obvodů, přesnost a kvalitu regulace. K dalšímu prudkému rozvoji řídící techniky došlo během druhé světové války, ke kterému přispěli Nathaniel B. Nichols (1914 - 1997), Hubert Maxwell James (1908 - 1986), Ralph S. Phillips (1913 - 1998), Antonín Svoboda (1907 – 1980). Principy zpětnovazebního řízení se začaly uplatňovat u autopilotů letadel, řízení radarových antén, zaměřování zbraní, apod. V 50. letech začala být pozornost věnována nelineárním systémům.

V roce 1960 publikoval Rudolf Emil Kalman (1930) rovnice pro návrh LQ *(Linear-quadratic)* regulátoru. S rozvojem výpočetní techniky se v teorii řízení začala uplatňovat metoda stavového prostoru, která mimo jiné umožnila řešení mnohorozměrových systémů včetně problematiky diskrétního řízení. V 90. letech se pozornost přesunula zejména k optimalizačním úlohám a robustnosti systémů. Typickým nástrojem pro návrh řídicích systémů se staly softwarové balíky Program CC, MATRIXx, MATLAB, Easy5, aj.

10.2 Základní zapojení regulačního obvodu

Možné zapojení regulátoru do regulačního obvodu je znázorněno na obrázku 42. $w(t)$ je žádaná hodnota regulované veličiny, $e(t)$ odchylka od požadované hodnoty, $u_r(t)$ akční



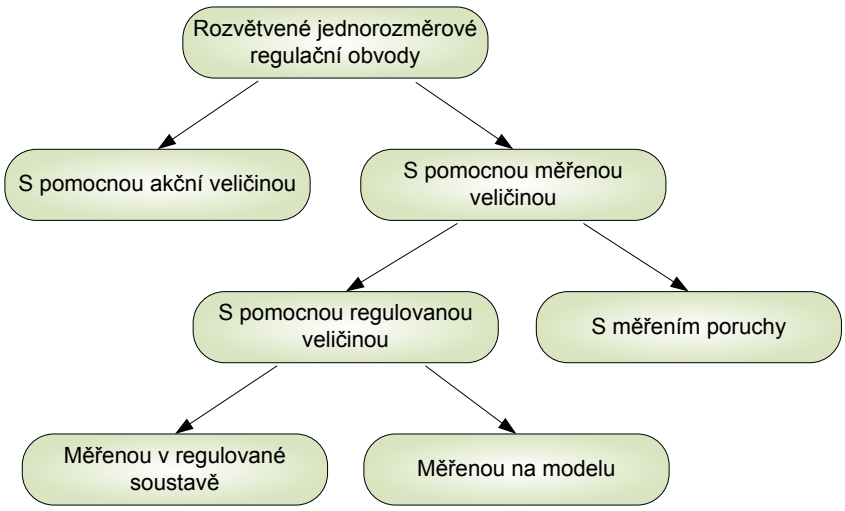
Obrázek 42: Základní zapojení jednoduchého regulačního obvodu

zásah regulátoru, $u(t)$ akční zásah ovlivněný chybou $v_1(t)$, $y(t)$ regulovaná veličina, $v_2(t)$ chyba vstupující do soustavy.

Regulační obvody se dají rozdělit na spojité a diskrétní. Diskrétní regulační obvody jsou takové, v nichž alespoň jeden člen pracuje diskrétně, tj. informaci přijímá, nebo vydává v diskrétních časových okamžicích.

10.3 Rozvětvené jednorozměrové regulační obvody

Jednoduché jednorozměrové regulační obvody splňují většinu běžných regulačních úkolů [5]. Při vyšších požadavcích na přesnost a dynamiku regulace, hlavně u složitějších regulovaných soustav, jsou však jejich možnosti značně omezené.



Obrázek 43: Rozdělení jednorozměrových regulačních obvodů

S vyšším řádem setrvačnosti regulované soustavy se při rostoucím zesílení a při zkracování integrační časové konstanty regulátoru zvyšuje sklon k nestabilitě regulačního obvodu a zhoršuje se kvalita regulačního pochodu. Zavedením dalších proměnných do regulačního obvodu a tedy uspořádáním dalších smyček regulačního obvodu, se získají výhodnější dynamické i statické vlastnosti celého systému.

Způsoby rozvětvení jednorozměrných regulačních obvodů jsou schématicky vyjádřeny na obrázku 43. Blíže je o jednotlivých implementacích diskutováno v [5].

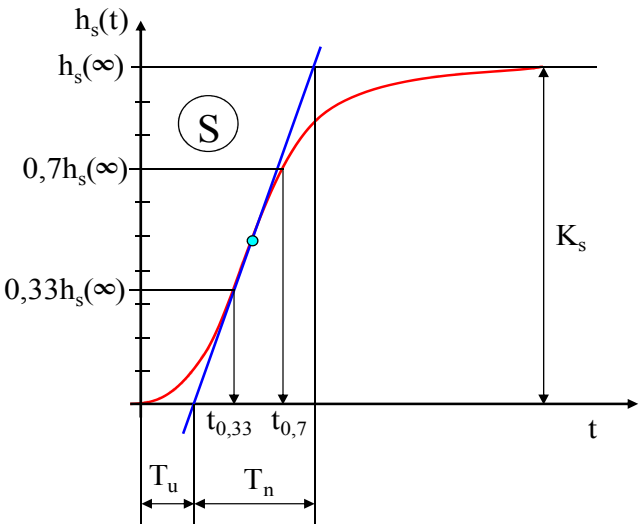
10.4 Identifikace soustav

Cílem identifikace je zhotovení matematického modelu na základě informací získaných měřením. K tomuto účelu je možno využít mnoha metod, na jejichž výběru závisí přesnost náhradního modelu. Základními metodami jsou:

1. fyzikálně–matematická analýza na základě konstrukčních a fyzikálních dat soustavy,
2. vyhodnocování přechodových charakteristik,
3. vyhodnocování signálů obecných tvarů,
4. vyhodnocování frekvenčních charakteristik v komplexní rovině,
5. vyhodnocování frekvenčních charakteristik v logaritmických souřadnicích,
6. vyhodnocování náhodných signálů,
7. identifikace s adaptivním modelem.

10.4.1 Nekmitavé proporcionální regulované soustavy

[20] Nejjednodušší identifikační metodou je určení doby průtahu T_u a doby náběhu T_n, viz. obrázek 44. Tyto doby jsou určeny tečnou vedenou inflexním bodem na přechodové



Obrázek 44: Nekmitavá proporcionální regulovaná soustava

charakteristice $h_s(t)$. Součet obou dob dává dobu přechodu:

$$T_p = T_u + T_n \tag{10.1}$$

Náhradní přenos regulované soustavy 1.řádu s dopravním zpožděním lze zapsat ve tvaru:

$$G_s(s) = \frac{k_s}{T_n s + 1} e^{-T_u s} \tag{10.2}$$

Koeficient přenosu k_s je dán ustáleným stavem na přechodové charakteristice:

$$k_s = h_s(\infty), \text{ případně } k_s = \frac{\Delta y}{\Delta u} \tag{10.3}$$

Přesnější aproximaci regulované soustavy lze docílit odečtením hodnot z přechodové charakteristiky v čase $t_{0,33}$ a $t_{0,7}$, viz. obrázek 44 a určit náhradní přenos ve tvaru:

$$G_s(s) = \frac{k_s}{T_1 s + 1} e^{-T_{d1} s} \tag{10.4}$$

Časovou konstantu T_1 a dopravní zpoždění T_{d1} lze určit na základě vztahů:

$$\begin{aligned} T_1 &= 1{,}245(t_{0,7} - t_{0,33}) \\ T_{d1} &= 1{,}498(t_{0,33} - 0{,}498 t_{0,7}) \end{aligned} \tag{10.5}$$

Ze vztahů 10.2 a 10.5 je patrné, že obě metody vedou na stejný přenos, ale s rozdílnými časovými konstantami a dopravním zpožděním.

V případě, že je na přechodové charakteristice výrazně vystupující inflexní bod, lze výrazně lepší aproximaci docílit použitím náhradního přenosu ve tvaru:

$$G_s(s) = \frac{k_s}{(T_2 s + 1)^2} e^{-T_{d2} s} \tag{10.6}$$

Časové konstanty T_2 a T_{d2} je možno získat na základě vztahů:

$$\begin{aligned} T_2 &= 0{,}794(t_{0,7} - t_{0,33}) \\ T_{d2} &= 1{,}937(t_{0,33} - 0{,}937 t_{0,7}) \end{aligned} \tag{10.7}$$

Ještě přesnější aproximaci přechodové charakteristiky regulované soustavy s výrazným inflexním bodem lze získat použitím náhradního přenosu s rozdílnými časovými konstantami T_1 a T_2:

$$G_s(s) = \frac{k_s}{(T_1 s + 1)(T_2 s + 1)} e^{-T_{d2} s} \tag{10.8}$$

Časové konstanty lze získat ze vztahů:

$$\begin{aligned} T_{1,2} &= \frac{1}{2}\left(D_2 \pm \sqrt{D_2^2 - 4D_1^2}\right), \\ T_{d2} &= 1{,}937(t_{0,33} - 0{,}937 t_{0,7}), \\ D_1 = 0{,}794(t_{0,7} - T_{0,33}), \quad D_2 &= \frac{S}{h_s(\infty)} - T_{d2} \end{aligned} \tag{10.9}$$

Z výše uvedených rovnic vyplývá, že musí platit $D_2 \geq 2D_1$. V opačném případě je nutno použít jiný druh náhrady přenosu.

Při experimentální identifikaci proporcionálních nekmitavých (aperiodických) regulovaných soustav vyššího řádu se často používá **Strejcova metoda**. Předpokladem jsou reálné a záporné kořeny charakteristické rovnice. Metoda využívá k aproximaci přenosu informaci o dobách průtahu T_u a náběhu T_n. Aproximační přenos vyšetřované soustavy je ve tvaru:

$$G_s(s) = \frac{k_s}{(T_i s + 1)^i} e^{-T_{di} s} \tag{10.10}$$

kde T_i je časová konstanta proporcionální regulované soustavy se setrvačností i-tého řádu, T_{di} je dopravní zpoždění.

Rekurentní rovnice nekmitavé proporcionální soustavy, která byla v experimentální části použita pro simulaci regulace je dána ve tvaru:

$$y(kT) = A_1 y[(k-1)T] + A_2 y[(k-2)T] + b_0 u(kT) \tag{10.11}$$

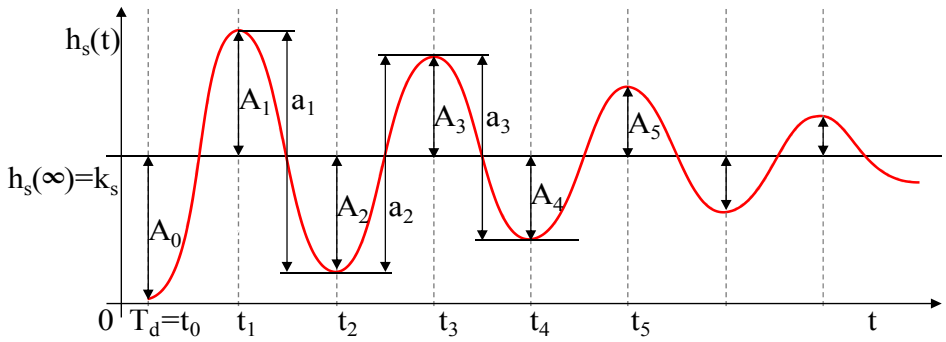
kde A_1, A_2, A_3 jsou konstanty:

$$A_1 = \frac{2T_1T_2 + T(T_1+T_2)}{T_1T_2 + T(T_1+T_2) + T^2}, \quad A_2 = -\frac{T_1T_2}{T_1T_2 + T(T_1+T_2) + T^2},$$
$$b_0 = \frac{k_S T^2}{T_1T_2 + T(T_1+T_2) + T^2} \tag{10.12}$$

Rekurentní rovnice vznikla transformováním přenosu nekmitavé proporcionální soustavy (Eq. 10.8). První a druhá derivace byla nahrazena první, resp. druhou diferencí.

10.4.2 Kmitavé proporcionální regulované soustavy

Vyhodnocování přechodových charakteristik kmitavých soustav, viz. obrázek 45 je velmi náročné.



Obrázek 45: Kmitavá proporcionální regulovaná soustava

V případě, že je možno použít náhradní přenos ve tvaru:

$$G_s(s) = \frac{k_s}{T_0^2 s^2 + 2\xi_0 T_0 s + 1} e^{-T_d s} \tag{10.13}$$

platí vztahy:

$$\frac{A_{i+1}}{A_i} = \frac{a_{i+1}}{a_i}, \quad \text{resp.} \quad \frac{A_{i+2}}{A_i} = \frac{a_{i+2}}{a_i}, \quad \text{pro} \quad i = 1, 2, \ldots, N+1 \tag{10.14}$$

kde N je počet odečtených hodnot z přechodové charakteristiky regulované soustavy dle obrázku 45, kdy přechodová funkce $h_s(t)$ má svá lokální maxima a minima. Koeficient poměrného tlumení lze určit dle:

$$\xi_0 = -\frac{\ln \frac{a_{i+1}}{a_i}}{\sqrt{\pi^2 + \ln^2 \frac{a_{i+1}}{a_i}}} \quad resp. \quad \xi_0 = -\frac{\ln \frac{a_{i+2}}{a_i}}{\sqrt{4\pi^2 + \ln^2 \frac{a_{i+2}}{a_i}}} \tag{10.15}$$

Časovou konstantu:

$$T_0 = \frac{\sqrt{1-\xi_0^2}}{\pi N}(t_{N+1} - t_1) \tag{10.16}$$

a dopravní zpoždění:

$$T_d = \frac{1}{N}\left[\sum_{i=1}^{N} t_i - \frac{N+1}{2}(t_{N+1} - t_1)\right] \tag{10.17}$$

kde N je počet uvažovaných úseků. Výsledná hodnota poměrného tlumení se určí jako střední hodnota vztahů 10.15.

Uvedený postup dává dobré výsledky v případě, kdy časová konstanta T_0 je podstatně větší, než další setrvačné časové konstanty regulované soustavy.

Rekurentní rovnice kmitavé proporcionální soustavy, která byla v experimentální části použita pro simulaci regulace je dána ve tvaru:

$$y(kT) = A_1 y[(k-1)T] + A_2 y[(k-2)T] + b_0 u(kT) \tag{10.18}$$

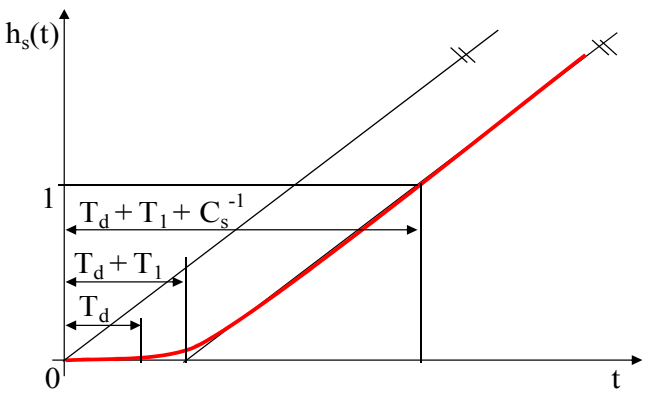
kde A_1, A_2, A_3 jsou konstanty:

$$A_1 = \frac{2 + 2\xi T_0 T}{T_0^2 + 2\xi T_0 T + T^2}, \quad A_2 = \frac{1}{T_0^2 + 2\xi T_0 T + T^2}, \quad b_0 = \frac{k_S T^2}{T_0^2 + 2\xi T_0 T + T^2} \tag{10.19}$$

Rekurentní rovnice vznikla transformováním z přenosu kmitavé proporcionální soustavy (Eq. 10.13). První a druhá derivace byla nahrazena první, resp. druhou diferencí.

10.4.3 Integrační regulované soustavy

Integrační soustavy nemají, na rozdíl od soustav proporcionálních, samoregulační schopnost. Po vyvedení soustavy z rovnovážného stavu a odeznění přechodového děje se výstupní signál mění konstantní rychlostí, viz. obrázek 46. Vyhodnocování přechodových



Obrázek 46: Integrační regulovaná soustava

charakteristik integračních regulovaných soustav může být prováděno v souladu s uvedeným obrázkem za předpokladu, že náhradní přenos má tvar:

$$G_s(s) = \frac{c_s}{s(T_1 s + 1)} e^{-T_d s} \tag{10.20}$$

kde c_s je statický součinitel rychlosti c_s $[s^{-1}]$ je vyjádřen ve tvaru:

$$c_s = \frac{1}{a_1} \tag{10.21}$$

Rekurentní rovnici integrační soustavy, která byla v experimentální části použita pro simulaci regulace je dána ve tvaru:

$$y(kT) = A_1 y[(k-1)T] + A_2 y[(k-2)T] + b_0 u(kT) \tag{10.22}$$

kde A_1, A_2, A_3 jsou konstanty:

$$A_1 = \frac{2T_1 + T}{T_1 + T}, \quad A_2 = -\frac{T_1}{1+T}, \quad b_0 = \frac{k_S T^2}{T_1 + T} \tag{10.23}$$

Rekurentní rovnice vznikla transformováním přenosu integrační soustavy (Eq. 10.20). První a druhá derivace byla nahrazena první, resp. druhou diferencí.

10.5 Regulátory

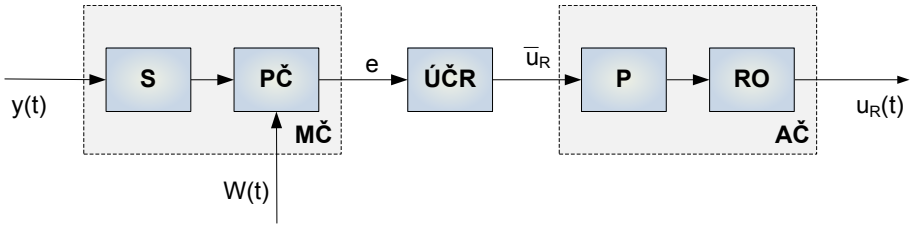
Regulátorem je nazýváno zařízení v regulačním obvodu, kterým se uskutečňuje proces automatické regulace. Podstata činnosti regulátoru spočívá ve vyhodnocení a zpracování regulační odchylky (10.24) podle zákona řízení a ve vytvoření výstupního signálu (akční veličiny) $u_r(t)$ s cílem eliminovat odchylku $e(t)$.

$$e(t) = w(t) - y(t) \tag{10.24}$$

Typy regulátorů můžeme rozdělit:

- podle přívodu energie: přímé / nepřímé
- podle nositele signálu: elektrické / hydraulické / pneumatické (často se využívá kombinovaných regulačních systémů)
- podle průběhu vstupní veličiny: spojité / nespojité

Blokové Schéma nepřímého regulátoru je ukázáno na obrázku 47. **Měřící člen** slouží k



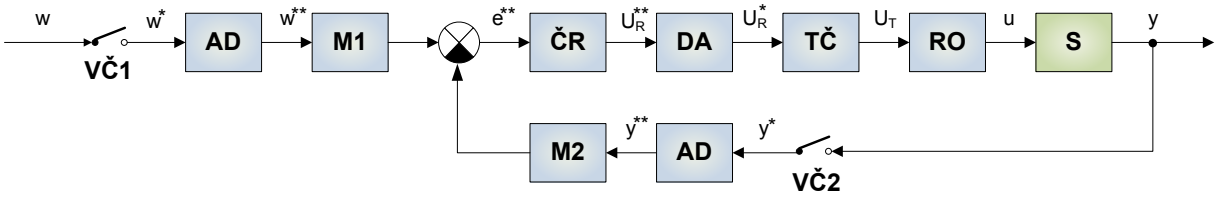
MČ: měřící člen **S**: snímač **PČ**: porovnávací člen **ÚČR**: ústřední člen regulátoru **AČ**: akční člen **P**: pohon **RO**: regulační orgán

Obrázek 47: Blokové schéma spojitého regulátoru

určení regulační odchylky. Skládá se ze **snímače**, jenž měří skutečnou hodnotu regulované veličiny a z **porovnávacího členu**, který porovnává tuto naměřenou hodnotu s požadovanou hodnotou a počítá její rozdíl, (Eq. 10.24). **Ústřední člen regulátoru** zpracovává regulační odchylku (např. podle regulačního zákona P, PI, PD, PID, apod.) a vrátí velikost akčního zásahu, kterou předá akčnímu členu. **Akční člen** je složen z **pohonu** a **regulačního orgánu**, který určuje velikost akčního zásahu $u_r(t)$.

10.5.1 Diskrétní regulátory

Diskrétní (číslicové) regulátory [5] se liší od spojitých regulátorů tím, že pracují s posloupností diskrétních hodnot. Pro převedení spojitých signálů do diskrétních a naopak je potřeba do regulačního obvodu zařadit *AD/DA (AČ/ČA) převodník, vzorkovač, tvarovač a paměťový člen* tak, jak je znázorněno na obrázku 48. Spojité signály y, w se vzor-



VČ1, VČ2: vzorkovací člen, **AD, DA**: převodníky, **M1, M2**: paměti, **ČR**: číslicový regulátor, **TČ**: tvarovací člen, **RO**: regulační orgán, **S**: soustava

Obrázek 48: Blokové schéma diskrétního regulátoru

kují pomocí vzorkovače na posloupnost vzorků y^* a w^*, které se pomocí *AD* převodníku převádějí na posloupnost číslicových hodnot y^{**} a w^{**}. Jednotlivé hodnoty y^{**} a w^{**} se zapisují do *paměti*, odkud se postupně počítají regulační odchylky e^{**}. Z těchto regulačních odchylek se poté v regulačním členu počítají akční zásahy u_R^{**}. V *DA* převodníku se vytvoří posloupnost impulzů u_R^*, která se v *tvarovacím členu* tvaruje na akční veličinu u_T vstupující do *regulačního orgánu*, jehož výstupem je akční veličina u.

Vzorkování: Vzorkovací člen mění spojitý signál $f(t)$ na diskrétní $f^*(t)$. Výstupním signálem je posloupnost impulzů zanedbatelně malé šířky $\gamma \to 0$, jejichž výška je rovna hodnotám vstupního signálu v okamžicích T, představujících vzorkovací periodu. Vzorkovací kmitočet je vyjádřen vztahem:

$$\omega_v = \frac{2\pi}{T} \tag{10.25}$$

Vzorkovací perioda: Velikost vzorkovací periody má podstatný vliv na stabilitu diskrétního regulačního obvodu a další vlastnosti. Vzorkovací perioda T, resp. vzorkovací kmitočet ω_v závisí na maximálním kmitočtu spojitého signálu $e(t)$. Důležitá je relace mezi *vzorkovacím kmitočtem* ω_v a *maximálním kmitočtem spojitého signálu* $e(t)$ (ω_m).

Aby nedošlo ke zkreslení *(aliasing)* kmitočtového spektra měřeného signálu $e(t)$ v rozsahu jeho kmitočtového pásma $\langle -\omega_m, +\omega_m \rangle$, musí být kmitočet vzorkování ω_v minimálně dvakrát větší, než nejvyšší kmitočet ω_m kmitočtového pásma měřeného signálu $e(t)$:

$$\omega_v \geq 2\omega_m, \quad T \leq \frac{\pi}{\omega_m} \tag{10.26}$$

Podmínka 10.26 se nazývá **Shannon–Kotělníkovův teorém**.

Závislost vzorkovací periody na kmitočtovém pásmu signálu je ukázána na obrázcích 49(a–f). Obrázky jsou seřazeny od nejvyšší vzorkovací frekvence po nejnižší. V uvedených obrázcích je modrou křivkou označen původní sinusový signál s frekvencí 10Hz a

vzorkovaný signál je označen červeně. Obrázek 49c a 49d zobrazuje hraniční vzorkovací frekvenci, jejíž další snížení by vedlo k aliasingu. Aliasing je ukázán na obrázcích 49e–49h. Pro vzorkovací frekvenci shodnou s frekvencí přenášeného harmonického signálu a jejich násobků $f_v = k \cdot f_s$ může nastat situace, kdy je původní signál vzorkován vždy v okamžiku průchodu původního signálu nulou. V tomto krajním případě je výsledkem úplná ztráta signálu. Tento jev je ukázán na obrázku 49h. f_v je vzorkovací frekvence, f_s frekvence signálu.

Pro přibližné určení vzorkovací periody[5] lze použít některého z níže uvedených vztahů:

$$T \approx T_1/10 \tag{10.27}$$

$$T \approx (1/6 \text{ až } 1/15)T_{95} \tag{10.28}$$

$$T \approx (1/4 \text{ až } 1/8)T_d \tag{10.29}$$

$$T \approx (1/2 \text{ až } 1/4)\sum_i \tau_i \tag{10.30}$$

kde T_1 je největší časová konstanta soustavy, T_{95} je čas, kdy přechodová charakteristika regulované soustavy dosáhne 95% své ustálené hodnoty, $\sum \tau_i$ je součet časových konstant regulované soustavy, T_d je dopravní zpoždění (vztah se používá u soustav s výrazným dopravním zpožděním)

Tvarování: Při využívání impulzů z číslicového regulátoru, viz. obrázek 48 pro spojitou soustavu, je třeba tyto impulzy tvarovat. Impulzy mají zanedbatelnou délku $\gamma \to 0$ a jejich amplituda je nositelem informace. Tvarování diskrétního signálu je v podstatě jeho přeměna na po částech spojitý signál, který je schopen předávat následujícímu členu jak informaci, tak potřebnou energii. Tvarovací člen tvaruje vstupní impulz šířky $\gamma \Leftrightarrow \delta$ na signál trvající jednu periodu vzorkování. δ značí Diracův impulz.

10.6 Řídicí algoritmy regulátorů

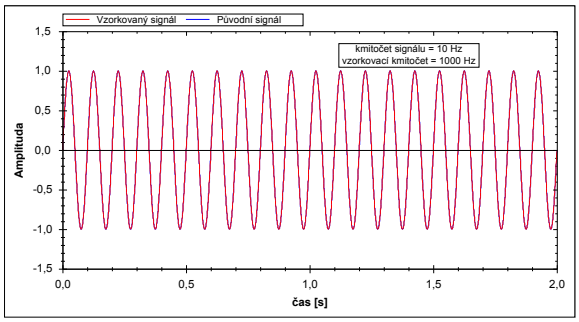
Řídící algoritmus je součástí *ústředního členu regulátoru*, viz. obrázek 47. Tento algoritmus zpracovává regulační odchylku a vrátí velikost akčního zásahu. Správný návrh tohoto algoritmu, případně jeho parametrů je klíčovou záležitostí syntézy regulačního obvodu.

10.6.1 Regulátory s pevně danou strukturou

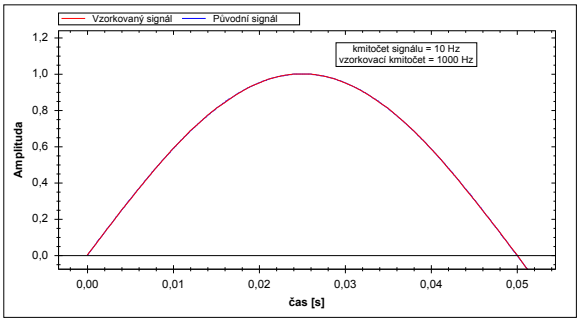
Tyto regulátory mají předem známý tvar popsaný přenosem, nebo rovnicí. Typickými představiteli takových regulátorů jsou číslicové analogie spojitých PID regulátorů. V číslicových verzích PID regulátorů nebývají zahrnuty filtrační a omezující vlastnosti, které se vyskytují v PID regulátorech a během regulace se pozitivně projevují. Proto se standardní verze číslicových regulátorů obvykle doplňují algoritmy zlepšující jejich funkci.

PID regulátory: Proporcionálně–integračně–derivační (PID) jsou spojité regulátory. Činnost ideálního PID regulátoru lze popsat rovnicí (Eq. 10.31), případně L–přenosem (přenos v Laplaceově transformaci) (Eq. 10.32):

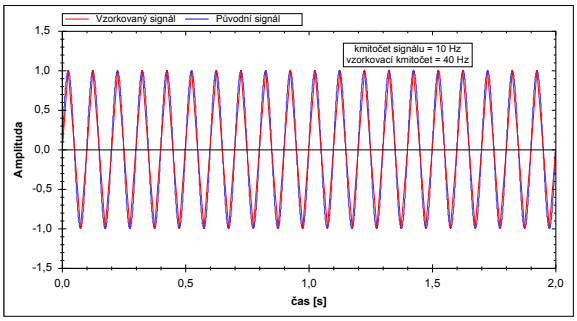
$$u(t) = r_0 e(t) + r_{-1}\int_0^t e(\tau)\mathrm{d}\tau + r_1\frac{\mathrm{d}e(t)}{\mathrm{d}t} \tag{10.31}$$



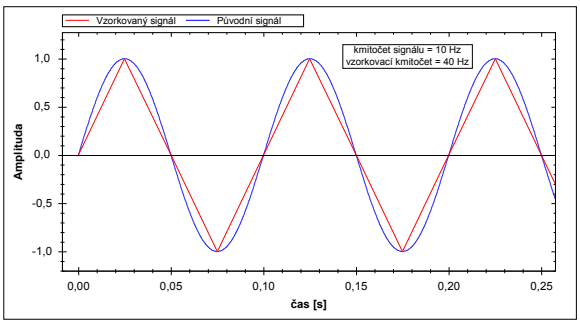
(a) $f_s = 10Hz,\ f_v = 1000Hz$



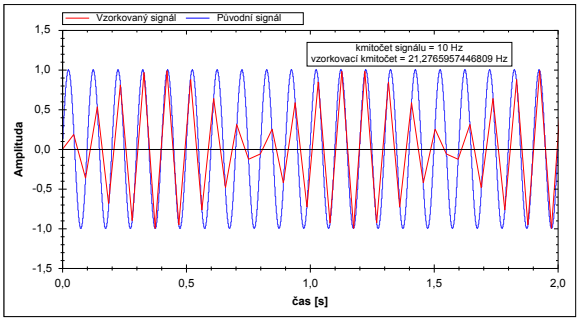
(b) $f_s = 10Hz,\ f_v = 1000Hz$ (detail)



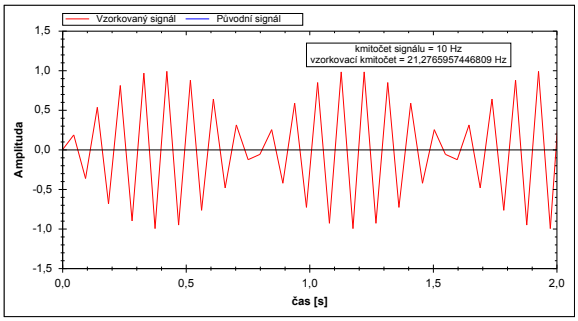
(c) $f_s = 10Hz,\ f_v = 40Hz$



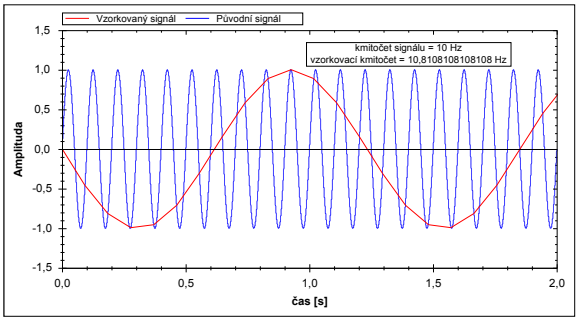
(d) $f_s = 10Hz,\ f_v = 40Hz$ (detail)



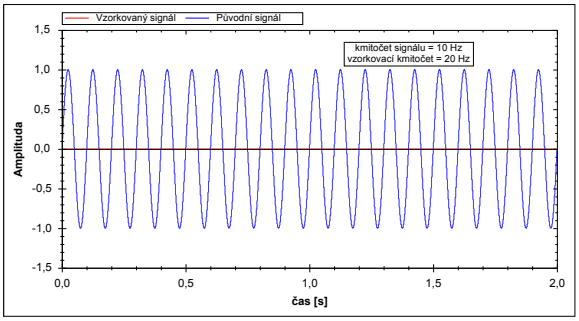
(e) $f_s = 10Hz,\ f_v = 21,28Hz$



(f) $f_s = 10Hz,\ f_v = 21,28Hz$ (zvýrazněno)



(g) $f_s = 10Hz,\ f_v = 10,81Hz$



(h) $f_s = 10Hz,\ f_v = 20Hz \Leftrightarrow 10Hz \Leftrightarrow j\omega$

Obrázek 49: Vzorkovací perioda a aliasing

$$G_R(s) = \frac{U(s)}{E(s)} = r_0 \left(1 + \frac{1}{T_I s} + T_D s\right) \tag{10.32}$$

Kde: r_{-1} je integrační konstanta regulátoru,
r_1 je derivační konstanta regulátoru,
$r_0 = k_R$ je zesílení regulátoru

$T_I = r_0/r_{-1}$ je integrační konstanta regulátoru
$T_D = r_1/r_0$ je derivační konstanta regulátoru

PSD regulátory: Proporcionálně–sumačně–diferenční (PSD) regulátor je číslicová verze PID regulátoru. PID regulátor definuje hodnotu akční veličiny $u(t)$ v libovolném časovém okamžiku t na základě znalosti průběhu regulační odchylky $e(t)$. V době kdy tento okamžik splyne s k-tým vzorkovacím okamžikem diskrétního regulátoru, tj. $t = kT$ platí následující rovnice, vycházející z (Eq. 10.31):

$$u(kT) = k_R e(kT) + \frac{k_R}{T_I} I(kT) + k_R T_D D(kT) \tag{10.33}$$

kde $I(kT)$, $D(kT)$ je integrál a derivace v časovém okamžiku kT. Pro výpočet těchto hodnot se využívají jednoduché způsoby přibližných náhrad spojitých integrací a derivací. Pro **náhradu integrace** se používá:

1. Zpětná obdelníková náhrada (ZOBD)

$$I(kT) = \int_0^t e(\tau)\mathrm{d}\tau \approx T \sum_{i=1}^{k} e(iT) \tag{10.34}$$

2. Dopředná obdelníková náhrada (DOBD)

$$I(kT) = \int_0^t e(\tau)\mathrm{d}\tau \approx T \sum_{i=0}^{k-1} e(iT) \tag{10.35}$$

3. Lichoběžníková náhrada (LICHO)

$$I(kT) = \int_0^t e(\tau)\mathrm{d}\tau \approx T \sum_{i=1}^{k} \frac{e(iT) - e[(i-1)T]}{2} \tag{10.36}$$

Náhrada derivace se provádí zpětnou diferencí:

$$D(kT) = \frac{\mathrm{d}e(t)}{\mathrm{d}t} \approx \frac{e(kT) - e[(k-1)T]}{T} \tag{10.37}$$

Polohový algoritmus PSD regulátoru: Pro malou vzorkovací periodu T lze integraci a derivaci v rovnici PID nahradit sumací a zpětnou diferencí 1.řádu. Pro nejběžněji používanou metodu ZOBD je možno vyjádřit algoritmus PSD regulátoru diferenční rovnicí:

$$u(kT) = k_R \left\{ e(kT) + \frac{T}{T_I} \sum_{i=1}^{k} e(iT) + \frac{T_D}{T} \Big\{ e(kT) - e[(k-1)T] \Big\} \right\} + u(0) \tag{10.38}$$

Tato rovnice PSD regulátoru je *nerekurentní*, tj. pro výpočet akčního zásahu je potřeba uchovat v paměti všechny regulační ochylky $e(iT)$.

Přírustkový algoritmus PSD regulátoru: tzv. *rekurentní tvar* rovnice PSD regulátoru je vyjádřen ve tvaru:

$$\nabla u(kT) = u(kT) - u[(k-1)T] = k_R\left\{\nabla e(kT) + \frac{T}{T_I}e(kT) + \frac{T_D}{T}\nabla^2 e(kT)\right\} \tag{10.39}$$

$\nabla e(kT)$ je zpětná diference prvního řádu, vyjádřená:

$$\nabla e(kT) = e(kT) - e[(k-1)T] \tag{10.40}$$

$\nabla^2 e(kT)$ je zpětná diference druhého řádu, vyjádřená:

$$\nabla^2 e(kT) = e(kT) - 2e[(k-1)T] + e[(k-2)T] \tag{10.41}$$

V rovnici (Eq. 10.39), je integrál nahrazen sumací pomocí zpětné obdelníkové metody (ZOBD) a derivace je nahrazena zpětnou diferencí. Tento vztah lze dále zobecnit na tvar uvedený v (Eq. 10.42).

$$\nabla u(kT) = q_0 e(kT) + q_1 e[(k-1)T] + q_2 e[(k-2)T] \tag{10.42}$$

kde: $\nabla u(kT) = u(kT) - u[(k-1)T]$ je přírustek oproti hodnotě v předchozím kroku. Hodnoty parametrů q_0, q_1, q_2 jsou závislé na typu náhrady integrace. Vztahy pro výpočet těchto parametrů jsou uvedeny v tabulce 17. Odvození algoritmu, včetně stavitelných

metoda \ parametr	q_0	q_1	q_2
ZOBD	$k_R\left(1+\frac{T_D}{T}+\frac{T}{T_I}\right)$	$-k_R\left(1+2\frac{T_D}{T}\right)$	$k_R\frac{T_D}{T}$
DOBD	$k_R\left(1+\frac{T_D}{T}\right)$	$-k_R\left(1+2\frac{T_D}{T}-\frac{T}{T_I}\right)$	$k_R\frac{T_D}{T}$
LICHO	$k_R\left(1+\frac{T_D}{T}+\frac{T}{2T_I}\right)$	$-k_R\left(1+2\frac{T_D}{T}-\frac{T}{2T_I}\right)$	$k_R\frac{T_D}{T}$

Tabulka 17: Varianty přepočtu parametrů PID na PSD

parametrů je uvedeno v [5].

10.6.2 Modifikace řídicího algoritmu

Takahashiho algoritmus: Nevýhodou přírustkového algoritmu, (Eq. 10.42) je poměrně velká hodnota parametrů q_0, q_1, q_2 při krátké periodě vzorkování, což se nepříznivě projevuje na průběhu regulačního pochodu, zejména při náhlých změnách žádané hodnoty regulované veličiny w. Navíc na počátku regulačního pochodu generuje řídicí algoritmus příliš velké akční zásahy $\nabla u(kT)$. Z těchto důvodů vznikly modifikace tohoto algoritmu, které umožňují větší tlumení akční veličiny u. Jedna z modifikací se nazývá *Takahashiho algoritmus*, který je vyjádřen ve tvaru:

$$\begin{aligned}\nabla u(kT) = k_R\Bigg\{&\Big\{-y(kT) + y[(k-1)T]\Big\} + \frac{T}{T_I}e(kT) + \\ &+\frac{T_D}{T}\Big\{-y(kT) + 2y[(k-1)T] - y[(k-2)T]\Big\}\Bigg\}\end{aligned} \tag{10.43}$$

kde $\nabla u(kT) = u(kT) - u[(k-1)T]$ je přírustek akční veličiny oproti hodnotě v předchozím kroku.

10.6.3 Obecné regulátory

Z výše uvedených algoritmů PSD regulátorů, (viz. kapitola 10.6.1), jsme vázáni pevně danou strukturou regulátoru, tedy složkami P, S, D. V tomto případě je úkolem syntézy najít vhodné parametry právě těchto složek. Výhodnější je použít *obecný číslicový regulátor* ve tvaru:

$$G_R(z) = \frac{Q(z)}{P(z)} = \frac{q_0 + q_1 z^{-1} + \cdots + q_q z^{-q}}{1 + p_1 z^{-1} + \cdots + p_p z^{-p}} \tag{10.44}$$

V takovém případě je úkolem syntézy určit strukturu regulátoru, tj. stupně polynomů $Q(z)$ a $P(z)$ a jeho parametry, tj. koeficienty zmíněných polynomů. Tento přístup umožňuje do procesu syntézy zapracovat základní požadavky na chování uzavřeného regulačního obvodu (např. fyzikální realizovatelnost, stabilitu), ale i některé další již ne zcela běžné požadavky, kterými můžeme přechodový děj podle zvoleného kritéria optimalizovat (Dead–beat regulace, řízení s omezením akční veličiny, minimalizace kvadratické regulační plochy, minimalizace překmitu regulované veličiny, omezení kmitání akčního zásahu, apod.)

10.7 Význam jednotlivých složek regulátorů

Pásmo proporcionality (pp) [%]: určuje o jakou procentuálně vyjádřenou hodnotu se musí změnit vstupní signál regulátoru, aby se akční člen přenastavil z jedné krajní polohy do druhé.

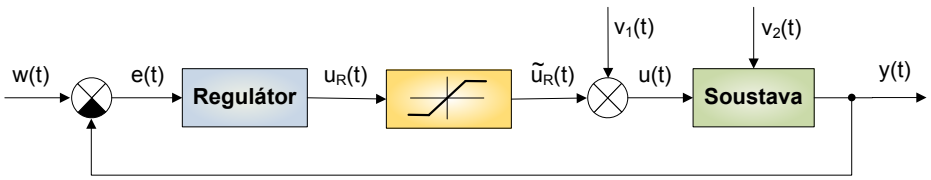
$$pp = \frac{1}{r_0} \cdot 100 \quad [\%] \tag{10.45}$$

Integrační časová konstanta (T_I) [s]: je čas, který by potřeboval čistě integrační regulátor, aby přenastavil akční člen do polohy, které dosáhne PI regulátor v čase $t = 0$ vlivem své proporcionální složky.

Derivační časová konstanta (T_D) [s]: je čas, který by potřeboval čistě proporcionální regulátor, aby přenastavil akční člen do polohy, které dosáhne PD regulátor v čase t = 0 vlivem své derivační složky.

10.8 Antiwind-up

Jde o opatření, které má zabránit pokračující integraci (načítání) při generování hodnoty signálu akční veličiny poté, co akční veličina dosáhla saturace. Na obrázku 50 je ukázána regulační smyčka s respektováním saturace akční veličiny.



Obrázek 50: Regulační smyčka s respektováním saturace akční veličiny

Jestliže není aplikována metoda, která zabrání unášení integrační složky diskrétního regulátoru, dojde v případě *saturace* akčního zásahu k nežádoucímu načítání integrační složky, který se zastaví až změnou znaménka regulační odchylky. To ovšem způsobí určité zpoždění regulace do doby, než dojde ke zpětnému odintegrování a než se regulační orgán odpoutá od dorazu. Tento jev se také nazývá *wind-up zpoždění* a je příčinou zhoršení kvality regulace.

Algoritmické zastavení integrace umožňuje wind-up zpoždění eliminovat způsoby uvedenými níže:

- u **polohových algoritmů** se integrace (sumace) zastaví zavedením nulové regulační odchylky na vstup integrátoru (sumátoru) při splnění podmínky (Eq. 10.46) za předpokladu, že kladná regulační odchylka působí ve smyslu přírustku akční veličiny a současně způsobuje pokles hodnoty regulované veličiny.

$$e(t) = 0 \Leftarrow \begin{cases} u(t) > u_{max} \wedge e(t) > 0 \\ u(t) < u_{min} \wedge e(t) < 0 \end{cases} \tag{10.46}$$

- u **přírustkových algoritmů** lze provést omezení vypočtené hodnoty akční veličiny na interval $\langle u_{min}, u_{max} \rangle$ a v časech $u[(k-1)T]$ uchovávat hraniční hodnoty z tohoto intervalu.

10.9 Seřízení regulátorů

Pro správnou funkci regulátoru je nutné provést nastavení jeho parametrů podle typu regulované soustavy. Následující kapitoly jsou stručným přehledem vybraných metod používaných pro nastavení parametrů spojitých a diskrétních regulátorů.

10.9.1 Seřízení regulátorů Ziegler–Nicholsovou metodou

Je experimentálním postupem nastavení *P, PI, PD, PID*, regulátorů včetně jejich diskrétních verzí. Tento přístup je také nazýván *metoda kritického zesílení regulátoru.* Metoda nevyžaduje žádné znalosti o regulované soustavě. Základní myšlenkou metody je přivést obvod na hranici stability a tím získat tzv. *kritické zesílení* r_{ok} a *kritickou periodu kmitů* (T_k). Postup seřizování metodou *Ziegler-Nicholse* je následující:

1. regulátoru se vyřadí integrační a derivační složka nastavením $T_I = \infty$ a $T_D = 0$,
2. pomalu se zvyšuje zesílení r_0 až do doby, kdy nastanou netlumené kmity o konstantní amplitudě a konstantní periodě. Tím se získá *kritické zesílení* r_{0k} a *kritická doba kmitu* T_k.
3. skutečné hodnoty regulátoru se určí z tabulky 18 podle r_{0k} a T_k.

Seřízení regulátorů metodou Ziegler–Nicholse je velmi jednoduché a zaručuje dobrý průběh regulačního pochodu, o kterém ovšem nelze tvrdit, že je optimálním nastavením. Ukázka kritického zesílení je na obrázku 51.

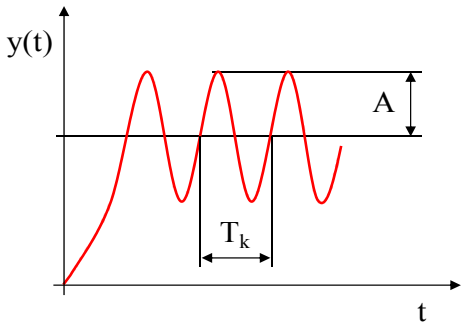
Metoda není použitelná pro strukturálně stabilní a strukturálně nestabilní obvody. Strukturálně stabilní obvody nelze přivést s použitím P regulátoru do kritického stavu a pro strukturálně nestabilní obvody je seřízení pomocí P regulátoru nesmyslné.

	analogové			číslicové		
	r_0	T_I	T_D	r_0	$r_0\frac{T}{T_I}$	$r_0\frac{T_D}{T}$
P	$0{,}5r_{0k}$	–	–	$0{,}5r_{0k}$	–	–
PI (PS)	$0{,}45r_{0k}$	$0{,}83T_k$	–	$0{,}45r_{0k} - 0{,}27r_{0k}\frac{T}{T_k}$	$0{,}54r_{0k}\frac{T}{T_k}$	–
PD	$0{,}4r_{0k}$	–	$0{,}05T_k$	–	–	–
PID (PSD)	$0{,}6r_{0k}$	$0{,}5T_k$	$0{,}12T_k$	$0{,}6r_{0k}\left(1-\frac{T}{T_k}\right)$	$1{,}2r_{0k}\frac{T}{T_k}$	$\frac{3}{40}r_{0k}\frac{T}{T_k}$

Tabulka 18: Parametry regulátorů podle Ziegler–Nicholse

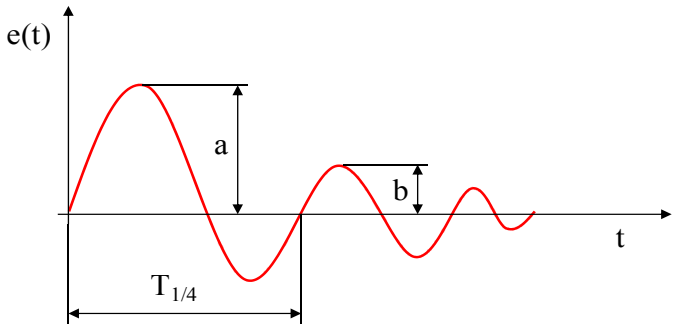
	k_R^*	T_I^*	T_D^*
Takahashi	$0{,}6k_{Rk}\left(1-\frac{T}{T_k}\right)$	$\frac{k_R^* T_k}{1{,}2k_{Rk}}$	$\frac{3}{40}\frac{k_{Rk}T_k}{k_R^*}$

Tabulka 19: Parametry Takahashiho regulátoru



Obrázek 51: Kritické zesílení

U mnohých regulovaných soustav je nepřípustné, aby nastalo rozkmitání obvodu. V tomto případě se používá modifikace Ziegler–Nicholsovy metody, nazývané **metoda čtvrtinového tlumení**, která spočívá v uvedení obvodu do takového stavu, aby podíl amplitud dvou po sobě jdoucích kmitů byl 0,25, viz. obrázek 52 ($\frac{b}{a} \approx \frac{1}{4}$). Poté je postup stejný jako u klasické metody a optimální nastavení se určí dle tabulky 20, ve které je místo r_{0k} zesílení při útlumu 4 (amplituda každého dalšího kmitu je čtyřikrát menší, než předcházejícího), tedy $r_{0-1/4}$ a $T_{k-1/4}$ místo T_k.



Obrázek 52: Ziegler–Nichols (metoda čtvrtinového tlumení)

	r_0	T_I	T_D
P	$r_{0-1/4}$	–	–
PI	$0{,}9r_{0-1/4}$	$T_{1/4}$	–
PID	$1{,}2r_{0-1/4}$	$0{,}6T_{1/4}$	$0{,}15T_{1/4}$

Tabulka 20: Seřízení spojitých regulátorů podle Ziegler–Nicholse (čtvrtinové tlumení)

Výhoda tohoto postupu spočívá v tom, že nevyžaduje mít znalosti o regulované soustavě. Nevýhodou této metody je, že potřebuje mít zapojený regulační obvod.

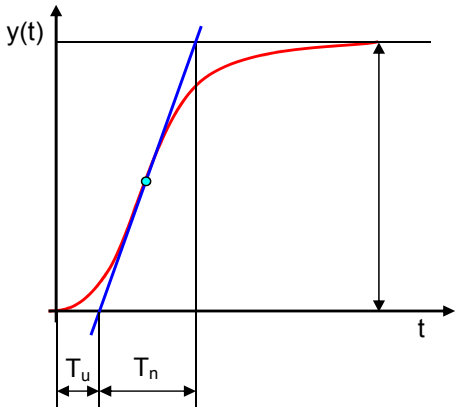
Metodu lze realizovat i početně, ale v tomto případě již potřebujeme mít znalosti o soustavě. Z přenosové rovnice soustavy vyjádříme přenos rozpojeného obvodu $G_0(S)$. Z tohoto dostaneme charakteristickou rovnici, ze které poté určíme kritické zesílení dosazením do Hurwitzova determinantu. *Hurwitzův determinant* je možno určit z *Hurwitzovy matice* H (10.47), která je stejného řádu jako charakteristický polynom.

$$H = \begin{bmatrix} a_{n-1} & a_{n-3} & a_{n-5} & \dots & 0 \\ a_n & a_{n-2} & a_{n-4} & \dots & 0 \\ 0 & a_{n-1} & a_{n-3} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & \dots & a_0 \end{bmatrix} \tag{10.47}$$

Z kritického zesílení již můžeme určit skutečné zesílení, integrační i derivační konstantu podle tabulky 18.

10.9.2 Seřízení regulátorů podle přechodové charakteristiky regulované soustavy

Metoda je určena pro proporcionální regulované soustavy s nekmitavým průběhem přechodové charakteristiky, u nichž lze stanovit dobu průtahu T_u, dobu náběhu T_n a zesílení soustavy K_s. Princip metody je ukázán na obrázku 53. Velikost T_u a T_n určuje tečna sestrojená v inflexním bodě.



Obrázek 53: Určení doby průtahu T_u a doby náběhu T_n

Po určení T_u, T_n lze nastavení regulátoru provést dle tabulky 21.

	$K_r^* \equiv r_0^*$	T_I^*	T_D^*
P	$\frac{T_n}{T_u}\frac{1}{K_s}$	–	–
PI	$0{,}9\frac{T_n}{T_u}\frac{1}{K_s}$	$3{,}5T_u$	–
PD	$1{,}2\frac{T_n}{T_u}\frac{1}{K_s}$	–	$0{,}25T_u$
PID	$1{,}25\frac{T_n}{T_u}\frac{1}{K_s}$	$2T_u$	$0{,}5T_u$

Tabulka 21: Nastavení spojitých regulátoru podle T_u, T_n

10.9.3 Seřízení regulátorů metodou inverze dynamiky

Metoda je také nazývaná *metoda požadovaného modelu* a umožňuje seřízení analogových i číslicových regulátorů [5, 40]. Má-li regulovaná soustava dopravní zpoždění, pak daná metoda umožňuje seřídit regulační obvod na předem zvolený relativní překmit přechodové charakteristiky v mezích $0 - 50\%$. U soustav bez dopravního zpoždění dává nekmitavé průběhy přechodové charakteristiky.

κ	0	0,05	0,10	0,15	0,20	0,25	0,30	0,35	0,40	0,45	0,50
α	1,282	0,984	0,884	0,832	0,763	0,697	0,669	0,64	0,618	0,599	0,577
β	2,718	1,944	1,720	1,561	1,437	1,337	1,248	1,172	1,104	1,045	0,992

Tabulka 22: Závislost koeficientů α, β na relativním překmitu κ

regulovaná soustava	PD, PID (PSD) regulátor			
	r_0^*		T_I^*	T_D^*
	$T_d = 0$	$T_d > 0$		
$\frac{c_s}{s(T_1 s+1)}e^{-T_d s}$	$\frac{2}{c_s(2T_W+T)}$	$\frac{a}{c_s}$	–	$T_1 - \frac{T}{2}$
$\frac{k_1}{(T_1 s+1)(T_2 s+1)}e^{-T_d s}$	$\frac{2T_I^*}{k_1(2T_W+T)}$	$\frac{aT_I^*}{k_1}$	T_1+T_2-T	$\frac{T_1T_2}{T_1+T_2} - \frac{T}{4}$
$\frac{k_S}{T_0^2 s^2 + 2\xi_0 T_0 s\ +1}e^{-T_d s}$	$\frac{2T_I^*}{k_1(2T_W+T)}$	$\frac{aT_I^*}{k_1}$	$2\xi_0 T_0 + -T$	$\frac{T_0}{2\xi_0} - \frac{T}{4}$

regulátor: **analogový:** $T = 0$, **číslicový:** $T > 0$

Tabulka 23: Nastavení PID (PSD) regulátoru metodou inverze dynamiky

V případě, že regulovaná soustava neobsahuje dopravní zpoždění ($T_d = 0$), je předpokládán přenos řízení ve tvaru:

$$G_W(s) = \frac{1}{T_W s + 1}, \quad G_W(z) = \frac{1 - e^{-\frac{T}{T_W}}}{z - e^{-\frac{T}{T_W}}} \tag{10.48}$$

kde T_W je časová konstanta uzavřeného regulačního obvodu. Pro určení vzorkovací periody je doporučen vztah:

$$T < 0{,}3T_W \tag{10.49}$$

V případě, že regulovaná soustava obsahuje dopravní zpoždění ($T_D > 0$) je předpokládán přenos řízení ve tvaru:

$$G_W(s) = \frac{a}{s + ae^{-T_d s}} e^{-T_d s}, \quad G_W(z) = \frac{aT}{z - 1 + aTz^{-\frac{T_d}{T}}} z^{-\frac{T_d}{T}} \tag{10.50}$$

kde a je zesílení otevřeného regulačního obvodu, T_d je dopravní zpoždění, T je vzorkovací perioda pro jejíž určení je doporučen vztah:

$$T < 0{,}32 T_d \tag{10.51}$$

Zesílení otevřeného regulačního obvodu a je závislé na zvoleném relativním překmitu κ přechodové charakteristiky uzavřeného regulačního obvodu, které je dáno vztahem:

$$a = \frac{1}{\alpha T + \beta T_d} \tag{10.52}$$

Hodnoty koeficientů α, β jsou uvedeny v tabulce 22.

V tabulce 23 je uvedeno nastavení PID (PSD) regulátoru pro proporcionální soustavu s dvěmi časovými konstantami a dopravním zpožděním, pro kterou platí $T_1 \geq T_2$. Možnosti nastavení jiných regulátorů pro jiné soustavy jsou uvedeny například v publikacích [5, 40].

10.9.4 Seřízení regulátoru podle funkcí standardního tvaru

Metoda vychází z tzv. **Whiteleových standardních tvarů** přenosové funkce uzavřeného regulačního obvodu, které dávají vyhovující regulační pochod. Podle těchto funkcí se upravují parametry regulátorů tak, aby se koeficienty a tvar přenosové funkce rovnaly vybrané vzorové funkci. Standardní tvary přenosových funkcí zaručují, že při dodržení předepsaných koeficientů charakteristické rovnice pro daný typ přenosu regulačního obvodu se nepřekročí daná hodnota maximálního přeregulování $\Delta y_{\max}$, případně doba regulace T_R, nebo jiné charakteristiky regulačního pochodu.

Blíže je o této metodě diskutováno například v [5].

10.9.5 Metoda pokusu a omylu

Metoda pokusu a omylu je v praxi nejčastěji používaným postupem spočívajícím v přímém experimentování s uzavřenou smyčkou. Metodou pokus–omyl jsou voleny hodnoty parametrů regulátoru podle tvaru odezvy na skokovou změnu žádané hodnoty, nebo uměle vyvolanou skokovou změnu poruchové veličiny. Existuje cela řada pravidel, které se snaží tento proces zefektivnit. Nejznámějším postupem je následující:

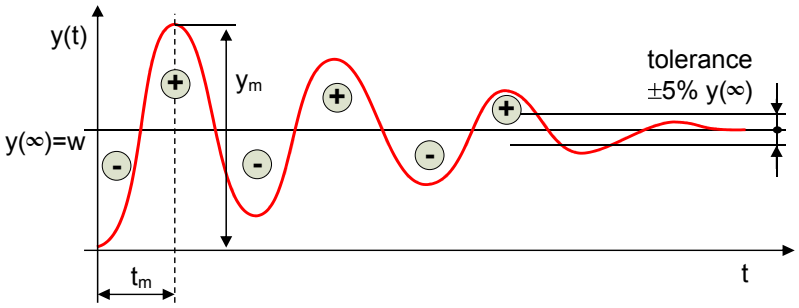
1. Nastav $T_I = \infty$, $T_D = 0$
2. Zvětšuj k_R až nastanou trvalé kmity o stejné amplitudě. Poté nastav $k_R = \frac{1}{2} k_{Rk}$
3. Snižuj T_I až nastanou trvalé kmity o stejné amplitudě (T_{Ik}). Poté nastav $T_I = 3T_{Ik}$
4. Zvyšuj T_D až nastanou trvalé kmity o stejné amplitudě (T_{Dk}). Poté nastav $T_D = \frac{1}{3} T_D$

10.10 Kvalita regulace

Kvalita seřízení regulátorů může být hodnocena v několika oblastech:

- časová oblast
- kmitočtová oblast
- komplexní rovina (rozložení kořenů)
- stavový prostor (stavová rovina)

V této práci je pozornost věnována hodnocení kvality regulace v **časové oblasti**. V tomto případě lze kvalitu regulace hodnotit podle průběhu regulované veličiny $y(t)$. Běžně se kvalita regulace v časové oblasti hodnotí podle odezvy na jednotkový skok žádané veličiny $w(t)$, nebo poruchové veličiny $v(t)$.



Obrázek 54: Hodnocení kvality regulace

Kvalitu regulace lze hodnotit ze statického, nebo dynamického hlediska. Do statického hlediska spadá například hodnocení **trvalé regulační odchylky**. Dynamické hledisko zohledňuje průběh regulačního pochodu. V případě dynamického hodnocení průběhu regulace se hodnotí například *relativní překmit*, *doba regulace*, *perioda kmitů*, *velikost regulační plochy*.

10.10.1 Relativní překmit

Relativní překmit (přeregulování) regulované veličiny κ je vyjádřen vztahem:

$$\kappa = \frac{y_m - y(\infty)}{y(\infty)} \tag{10.53}$$

kde y_m je maximální hodnota regulované veličiny. Jestliže $\kappa = 0$, jedná se o nekmitavý (aperiodický) průběh regulačního pochodu, který je požadován u tepelných, chemických procesů, případně v řízení robotů. U nekmitavých regulačních pochodů je často požadována minimální doba regulace t_R.

10.10.2 Doba regulace

Doba regulace t_R je doba, za kterou regulační odchylka y trvale klesne pod danou toleranci odchylky v ustálené hodnotě $y(\infty)$.

10.10.3 Velikost regulační plochy

Ideálním případem regulace by bylo, kdyby se změna regulované veličiny y projevila okamžitě dle požadavku změny žádané hodnoty w s chybou $e = w - y = 0$. Vzhledem k setrvačným hmotám, apod. samozřejmě tohoto stavu nelze docílit, ale je snahou aby regulační plocha byla minimální. Požadavek na dosažení minimální regulační plochy znamená zvolit kompromis mezi maximálním překmitem y_m a minimální dobou regulace t_r, jelikož minimální překmit a regulované veličiny a minimální doba regulace jsou dva protichůdné požadavky. S rostoucím zesílením regulátoru $r_0(k_r)$ klesá doba regulace, ale naopak roste maximální překmit. Pro výpočet regulační plochy je možno použít některé z následujících kritérií [5, 26].

Lineární regulační plocha *(Integral of Error)* (IE) je vhodná pro hodnocení aperiodického průběhu regulované veličiny. Je dána vztahem:

$$I = \int_0^\infty [y(t) - y(\infty)]\mathrm{d}t \tag{10.54}$$

Absolutní regulační plocha *(Integral of Absolute Error)* (IAE) odsraňuje nedostatek IE. Kritérium je vhodná pro hodnocení periodického průběhu regulované veličiny. Je dána vztahem:

$$I = \int_0^\infty |y(t) - y(\infty)|\mathrm{d}t \tag{10.55}$$

Kvadratická regulační plocha *(Integral of Squared Error)* (ISE) je vhodná pro hodnocení jakýchkoliv průběhů regulované veličiny. Kvadratická regulační plocha je popsána:

$$I = \int_0^\infty [y(t) - y(\infty)]^2\mathrm{d}t \tag{10.56}$$

ITAE *(Integral of Time multiplied by Absolute Error)* se používá k současné minimalizaci absolutní regulační plochy i doby regulace t_r. Kritérium je vyjádřeno vztahem:

$$I_{\mathrm{ITAE}} = \int_0^\infty t|y(t) - y(\infty)|\mathrm{d}t \tag{10.57}$$

ITSE *(Integral of Time-weighted Squared Error)* je vyjádřeno vztahem:

$$I_{\mathrm{ITSE}} = \int_0^\infty t[y(t) - y(\infty)]^2\mathrm{d}t \tag{10.58}$$

11 Praktická část

V této části jsou popsány konkrétní metody, které byly použity pro optimalizaci řídicích rovnic regulátorů. Je zde uveden také kompletní výpis pravidel gramatiky, které byly s různými variacemi použity pro generování struktury řídicích rovnic obecných regulátorů a také kompletní slovník sloužící pro snížení počtu redundantních částí ve struktuře řídicí rovnice obecného regulátoru.

11.1 Úplná gramatika a slovník ART

V předcházejících kapitolách byl poměrně podrobně vysvětlen princip TLTE, nicméně uváděná gramatika byla v určitých ohledech pro jednoduchost ilustrativní. Finální a úplná G a G^{-1} gramatika použitá pro optimalizaci řídicích rovnic obecných regulátorů je uvedena v tabulkách 24, 25. Pro úplnost je uveden také slovník ART viz. tabulka 26, ale je shodný se slovníkem definovaným dříve.

$\langle S \rangle$	$::=$	$\langle exp \rangle$
$\langle exp \rangle$	$::=$	$\langle fnc1 \rangle \langle exp \rangle \mid \langle fnc2 \rangle \langle exp \rangle \langle exp \rangle \mid \langle fnc2 \rangle \langle exp \rangle \langle pair \rangle \mid \langle pair \rangle \mid \langle var \rangle$
$\langle pair \rangle$	$::=$	$* \; num \; \langle var \rangle$
$\langle fnc1 \rangle$	$::=$	$U-$
$\langle mfnc1 \rangle$	$::=$	$U+$
$\langle fnc2 \rangle$	$::=$	$+ \mid - \mid * \mid /$
$\langle var \rangle$	$::=$	$U_{k-1} \mid U_{k-2} \mid U_{k-3} \mid U_{k-4} \mid U_{k-5} \mid$ $Y_k \mid Y_{k-1} \mid Y_{k-2} \mid Y_{k-3} \mid Y_{k-4} \mid Y_{k-5} \mid$ $E_k \mid E_{k-1} \mid E_{k-2} \mid E_{k-3} \mid E_{k-4} \mid E_{k-5} \mid$ $dE_k \mid dE_{k-1} \mid dE_{k-2} \mid dE_{k-3} \mid dE_{k-4}$

Tabulka 24: Pravidla generativní gramatiky pro optimalizaci regulátorů

Z uvedených pravidel P, P^{-1} je patrné, že narozdíl od předchozích uvedených pravidel lze pomocí gramatiky přímo generovat pouze symbolické konstanty num, které jsou dále optimalizovány v MDE, MGA, apod. Dále je vidět, že *terminální symbol* reprezentující *unární plus* nelze ze startovního symbolu vygenerovat. Terminál $U+$ lze vygenerovat pouze použitím operátoru mutace za pomocí neterminálu $\langle mfnc1 \rangle$. Z důvodu urychlení optimalizace přibyl také redundantní neterminál $\langle pair \rangle$. Také je možné si všimnout, že přibyly terminální symboly $Y_k, \ldots, Y_{k-5}$, reprezentující hodnotu regulované veličiny v čase kT podle *Takahashiho tvaru řídicí rovnice diskrétního regulátoru*, (viz. kapitola 10.6).

Je vhodné si připomenout, že pravidla pro ART se sice řídí pravidly gramatiky, ale v tabulce 26 nejsou pravidla v pravém slova smyslu. Symboly $pair, f_{2m}, f_{2s}$ jsou pouze zástupnými symboly *terminálních symbolů*, (viz. kapitola 8.8). Pozor na záměnu $pair$ s neterminálem definovaným v tabulce gramatiky G (viz. tabulka 24).

11.2 Kriteriální funkce

Vzhledem k optimalizaci pomocí TLTE bylo nutné navrhnout kriteriální (ohodnocovací) funkci. Tento návrh nebyl vůbec snadnou záležitostí, jelikož optimalizace struktury regulátoru není tak přímočará jako optimalizace konstant PID (PSD) regulátorů.

NSM	$U_{k-1} \mid U_{k-2} \mid U_{k-3} \mid U_{k-4} \mid U_{k-5}$	$::=$	$\langle var \rangle \mid num$
	$Y_k \mid Y_{k-1} \mid Y_{k-2} \mid Y_{k-3} \mid Y_{k-4} \mid Y_{k-5}$	$::=$	$\langle var \rangle \mid num$
	$E_k \mid E_{k-1} \mid E_{k-2} \mid E_{k-3} \mid E_{k-4} \mid E_{k-5}$	$::=$	$\langle var \rangle \mid num$
	$dE_k \mid dE_{k-1} \mid dE_{k-2} \mid dE_{k-3} \mid dE_{k-4}$	$::=$	$\langle var \rangle \mid num$
	num	$::=$	$\langle var \rangle$
	$U-$	$::=$	$\langle mfnc1 \rangle$
	$U+$	$::=$	$\langle fnc1 \rangle$
	$+ \mid - \mid * \mid /$	$::=$	$\langle fnc2 \rangle$
SSM	$U+ \mid U-$	$::=$	$\langle var \rangle \mid \langle num \rangle$
	$+ \mid - \mid * \mid /$	$::=$	$\langle var \rangle \mid \langle num \rangle \mid \langle fnc1 \rangle$
GM	$U_{k-1} \mid U_{k-2} \mid U_{k-3} \mid U_{k-4} \mid U_{k-5}$	$::=$	$\langle exp \rangle$
	$E_k \mid E_{k-1} \mid E_{k-2} \mid E_{k-3} \mid E_{k-4} \mid E_{k-5}$	$::=$	$\langle exp \rangle$
	$dE_k \mid dE_{k-1} \mid dE_{k-2} \mid dE_{k-3} \mid dE_{k-4}$	$::=$	$\langle exp \rangle$
	$U+ \mid U-$	$::=$	$\langle exp \rangle$

NSM: nestrukturální mutace *(Non-Structural Mutation)*, **SSM**: Zkracující strukturální mutace *(Shortening Structural Mutation)*, **GM**: obecná mutace *(General Mutation)*

Tabulka 25: Pravidla analytické gramatiky pro optimalizaci regulátorů

P1	U+(e)	$::=$	e
	U-(U-(e))	$::=$	e
	U-(U+(e)) | U+(U-(e))	$::=$	U-(e)
P2	$f_{2m}(num, num) \mid f_{2s}(num, num)$	$::=$	num
P3	$+(a_1, a_2)$	$::=$	$*(num, a)$
	$-(a_1, a_2)$	$::=$	num
	$/(a_1, a_2)$	$::=$	num
P4	$f_{2m}(num,\ pair(a)) \mid f_{2m}(pair(a),\ num)$	$::=$	$*(num,\ a)$
	$f_{2s}(num,\ pair(a)) \mid f_{2s}(pair(a),\ num)$	$::=$	$*(num,\ a)$
P5	$f_{2s}(pair(a),\ pair(a))$	$::=$	$*(num,\ a)$

první sloupec: (P1, P2, P3, P4, P5) slouží pro označení sad pravidel a vyjadřují náročnost dané operace (P1 – nejjednodušší, P5 – nejnáročnější)

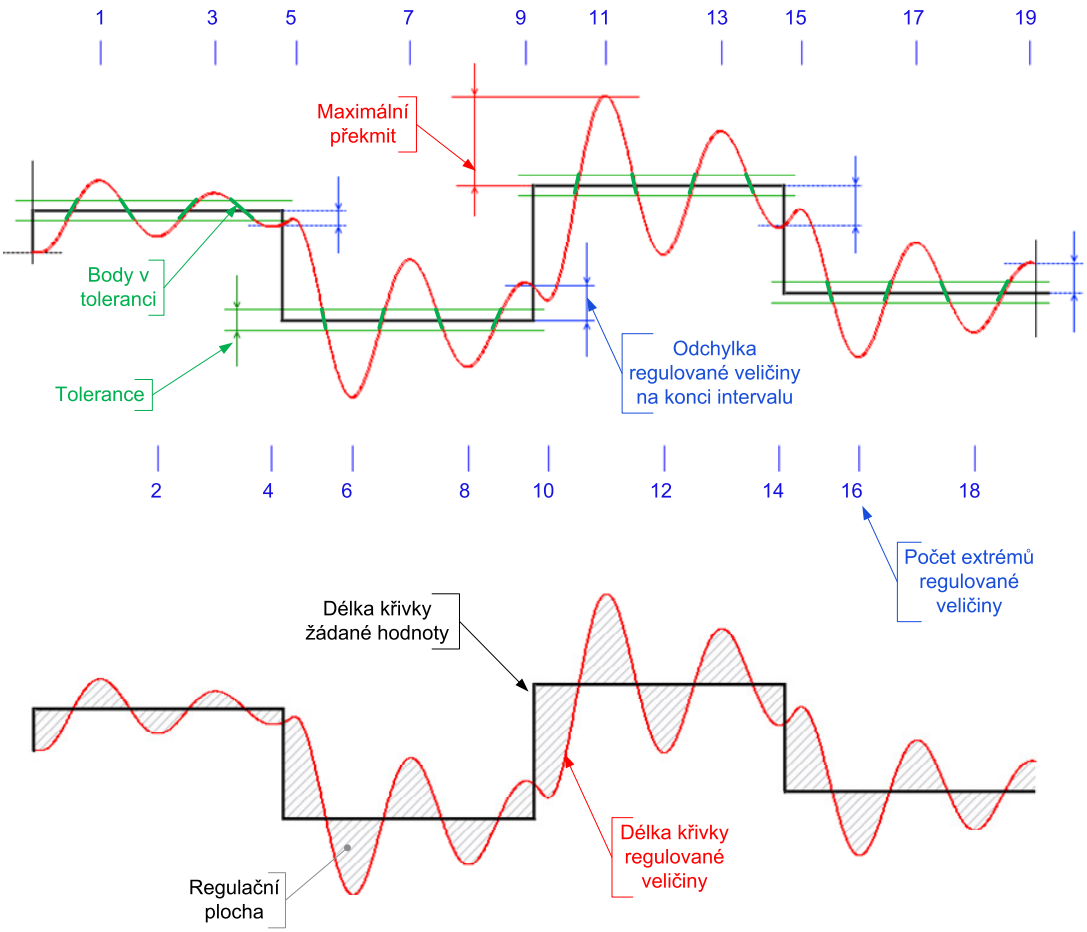
Tabulka 26: Pravidla pro ART

Problémem návrhu struktury řídicí rovnice regulátoru je nemožnost použít hodnocení kvality regulace v jednom intervalu, pro jednu hodnotu, ..., jelikož v tomto případě by došlo k nalezení takového řešení, které sice dokáže nastavit řídící veličinu na požadovanou velikost, ale to ještě neznamená, že daná struktura je regulátorem, resp. kvalitním regulátorem. Velice jednoduchým případem může být například rovnice regulátoru ve tvaru $u_k = 5$ ($u_k = num$; $num = 5$). V případě proporcionální regulované soustavy, kdy $k_s = 1$ a žádané hodnoty $w = 5$, sice časem dojde k nastavení regulované veličiny na hodnotu 5, ale tato struktura regulátoru je naprosto nepoužitelná. Tento příklad byl velice jednoduchou ukázkou toho, proč optimalizace obecných regulátorů není vůbec snadnou záležitostí. Pro hodnocení kvality regulace také nelze použít *pouze* integrálního kritéria, (viz. kapitola 10.10.3), jelikož malá velikost regulační plochy ještě neznamená, že daná

struktura regulátoru je kvalitní. Příklady, jenž demonstrují problémy návrhu jsou ukázány dále, (viz. kapitola 12). Kvůli výše uvedenému je vhodné provádět simulaci regulace na relativně dlouhém intervalu, díky čemuž se prodlužuje doba optimalizace.

Pro ohodnocení průběhu regulace bylo použito multikriteriální funkce, která vyhodnocuje simulaci regulace z hlediska:

- velikost kvadratické regulační plochy: $I = \int_0^{t_s} e(t)\mathrm{d}t$, kde $e(t)$ je regulační odchylka v čase t, t_s je délka simulace
- počet extrémů regulované veličiny
- průměrná chyba na konci jednotlivých intervalů
- počet bodů na křivce regulované veličiny v tolerančním pásmu
- poměr délek křivky regulované veličiny a řídící veličiny (l_y/l_w)
- absolutní hodnota maximálního relativního překmitu regulované veličiny
- absolutní hodnota maximálního relativního překmitu akční veličiny vzhledem k povolenému intervalu akčního zásahu $\{u_{min}, u_{max}\}$



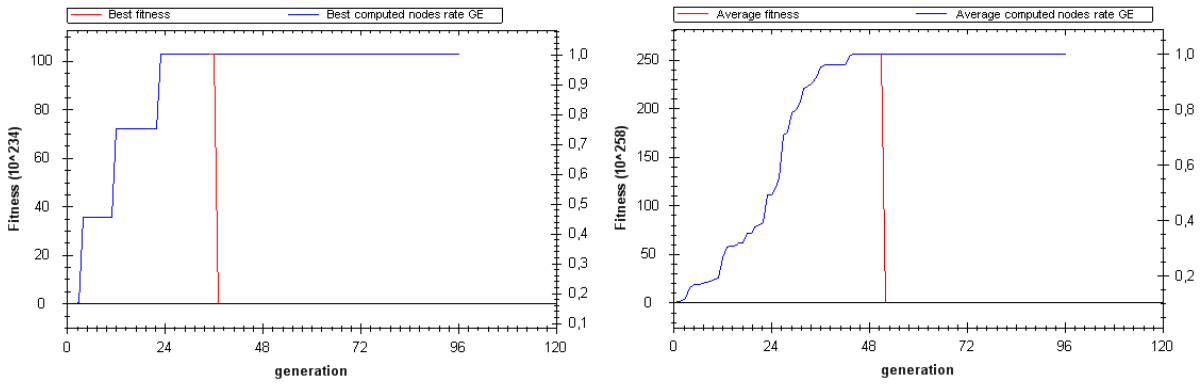
Obrázek 55: Kritéria hodnocení

Uvedená kritéria jsou ukázána na obrázku 55. Snahou optimalizace je minimalizace regulační plochy, počtu extrémů regulované veličiny, průměrné relativní regulační odchylky na konci každého intervalu, relativního překmitu regulované, nebo akční veličiny. Dále je cílem optimalizování poměru délek křivek regulované a řídící veličiny, tak aby $l_y/l_w = 1$ a maximalizace počtu bodů na křivce regulované veličiny, které jsou v povoleném tolerančním pásmu.

11.2.1 Hodnocení simulace v případě chyby

Aby bylo možno ohodnotit simulaci podle kritérií uvedených v kapitole 11.2, je nutné úspěšně dokončit celý proces simulace. V případě, že proces simulace nebude z jakéhokoliv důvodu dokončen (dělení nulou, $\pm\infty$, apod.), musí být algoritmus schopen hodnotit simulaci podle jiných kritérií. Tyto kritéria se liší v závislosti na optimalizovaném problému.

V případě, že je spuštěna **optimalizace PSD regulátoru**, je fitness hodnoceno podle počtu vypočítaných cyklů rekurentní rovnice PSD regulátoru. Přesněji podle poměru (c_c/c_{cp}), kde c_c je počet skutečně vypočítaných cyklů a c_{cp} je předpokládaný počet cyklů. **Příklad:** doba simulace je $10s$, vzorkování $0{,}1s$. Z toho vyplývá, že celkový předpokládaný počet výpočtů rekurentní rovnice PSD regulátoru je 100. V případě, že simulace bude z jakéhokoliv důvodu přerušena během výpočtu rekurentní rovnice v čase 8 sekund, bude poměr $c_c/c_p = 79/100 = 0{,}79$ (79 úspěšně dokončených výpočtů).



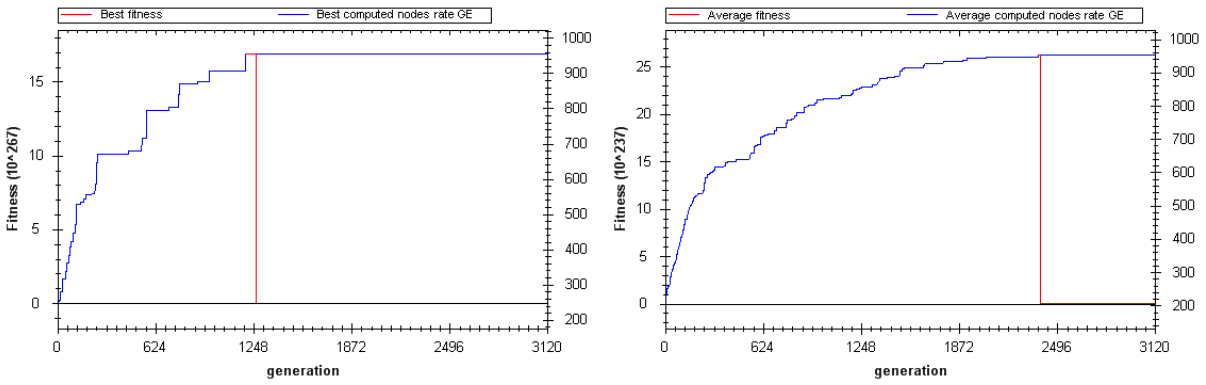
Obrázek 56: Průběh optimalizace PSD regulátoru pomocí DE

Optimalizace PSD regulátoru pomocí DE je ukázána na obrázku 56. Na levé straně je graf vývoje nejlepšího řešení a na pravé straně je zobrazen vývoj průměrného fitness v populaci. Na obou grafech je možno vidět dvě křivky. *Modrá* křivka zobrazuje poměr c_c/c_{cp}. *Červená* křivka zobrazuje vypočítané fitness v případě, že se podařilo úspěšně dokončit celou simulaci regulace. Je vidět, že fitness bylo možné vyčíslit až v případě, kdy $c_c/c_{cp} = 1$.

Jestliže je spuštěna **optimalizace obecného regulátoru**, je fitness počítáno podle počtu propočítaných uzlů ve stromu jedince za dobu simulace regulačního pochodu. Přesněji podle poměru (c_n/c_{nc}), kde c_n je skutečně propočítaný počet uzlů a c_{nc} je počet uzlů ve stromu jedince.
Příklad: Doba simulace je $10s$, vzorkování $0{,}1s$, rovnice regulátoru je ve tvaru $u_k = 3 \cdot E_k$. Z uvedené rovnice vyplývá, že strom funkce má 3 uzly ($c_{nc} = 3$), (viz. kapitola 7). Při

každém úspěšném výpočtu rekurentní rovnice tedy dojde k průchodu a propočtu 3 uzlů. V případě, že simulace bude z jakéhokoliv důvodu přerušena během výpočtu rekurentní rovnice v čase 8 sekund, při výpočtu součinu uzlů 3 a E_k bude výsledný počet propočítaných uzlů $79 \cdot 3 + 1 \cdot 2$ (79 krát úspěšně dokončen výpočet 3 uzlů a v posledním výpočtu byly vyčísleny dva uzly a u výpočtu jejich součinu došlo k chybě). Z toho vyplývá, že poměr $c_n/c_{nc} = 239/3 = 79,6\bar{6}$. V případě, že tento poměr vydělíme délkou simulace dělenou vzorkovací periodou, dostaneme poměr stejný jako v případě optimalizace PSD regulátoru (79,66/100 = 0,7966). Počítání uzlů namísto počtu úspěšně vypočítaných cyklů rekurentní rovnice je přesnější. Toho si lze všimnout i porovnáním s předchozím příkladem.

Optimalizace obecného regulátoru pomocí TLTE je ukázána na obrázku 57. Na levé straně je graf vývoje nejlepšího řešení a na pravé straně je zobrazen vývoj průměrného fitness v populaci. Na obou grafech je možno vidět dvě křivky. *Modrá* křivka zobrazuje poměr c_n/c_{nc}. *Červená* křivka zobrazuje vypočítané fitness v případě, že se podařilo úspěšně dokončit celou simulaci regulace. Z obrázku je mimo jiné vidět, že fitness bylo možné vyčíslit až v případě, kdy $c_n/c_{nc} = l_s \cdot c_{nc}$, kde l_s je délka simulace.



Obrázek 57: Průběh optimalizace parametrů obecného regulátoru pomocí DE

To, že je **simulace úspěšně dokončena**, ještě neznamená, že se podaří vypočítat fitness. Vzhledem k tomu, že fitness funkce provádí multikriteriální hodnocení simulace, může se stát, že v nějaké části výpočtu dojde k dosažení $\pm\infty$. V tom případě, je fitness počítáno podle *počtu úspěšně vypočítaných kritérií.*

Z výše uvedeného vyplývá, že ohodnocení simulace regulace může probíhat třemi způsoby, které jsou uvedeny níže v pořadí od nejhoršího případu, k nejlepšímu:

1. simulaci se nepodařilo z nějakého důvodu dokončit a fitness je počítáno z poměru c_c/c_{cp}, nebo c_n/c_{nc}
2. simulace byla úspěšně dokončena, ale nepodařilo se úspěšně vyčíslit všechny kritéria fitness funkce. Fitness je počítáno podle počtu úspěšně vyhodnocených kritérií.
3. simulace i výpočet fitness se podařilo úspěšně dokončit

Z uvedeného výčtu vyplývá, že *jedinec 3* má vždy lepší fitness, než *jedinec 2* a ten má lepší fitness, než *jedinec 1*, ($f_{i3} < f_{i2} < f_{i1}$).

Hodnocení podle počtu úspěšně vypočítaných kritérií fitness funkce není v grafech uvedeno.

Z obrázků 56 a 57 je vidět, že tento typ hodnocení je velice důležitý a bez něj by se s vysokou pravděpodobností vůbec nepodařilo strukturu obecného regulátoru, případně parametry PSD regulátoru optimalizovat, jelikož by jedince nebylo podle čeho hodnotit. Zvláště důležitý je tento typ hodnocení během optimalizace obecného regulátoru, viz. obrázek 57. Z tohoto obrázku je vidět, že celý průběh simulace pro jednoho jedince TLTE se podařilo dokončit až po cca 1200 generacích DE a průměrné fitness v populaci se podařilo dle kriteriální funkce hodnocící průběh simulace vyčíslit dokonce až po 2400 generacích.

11.3 Modifikované schéma DE

V kapitole 8.3 bylo řečeno, že k optimalizaci symbolických konstant ve struktuře jedince TLTE byla použita DE, (viz. kapitola 4). Tento typ evolučního algoritmu byl na počátku vývoje testován se schématy *DE/Rand/1* a *DE/Best/1*, (viz. kapitola 4.3.1). V průběhu testování nastal případ, kdy tyto schémata nepodávaly očekávané výsledky, hlavně v případech optimalizace parametrů z intervalu $\mathbb{R}$. Ve snaze vyřešit tento problém vznikla modifikace těchto schémat. Nové schémata byly pojmenovány **DE/Rand/1_mod** a **DE/Best/1_mod**, případně **DE/Best/1plus**, **DE/Rand/1plus** [45].

Rozdíl oproti schématům *DE/Rand/1* a *DE/Best/1* je ve způsobu výpočtu **diferenčního vektoru**. *Šumový vektor* je definován ve tvaru:

$$\vec{\nu}_i(t+1) = \vec{x}_{r_1^i}(t) + \beta \cdot f(\vec{x}_{r_2^i}(t), \vec{x}_{r_3^i}(t)) \tag{11.1}$$

Diferenční vektor je v tomto případě funkcí:

$$f(\vec{x}_{r_2^i}(t), \vec{x}_{r_3^i}(t)) \tag{11.2}$$

Výpočet tohoto vektoru závisí na hodnotách jednotlivých dimenzí vektorů $\vec{x}_{r_2^i}(t)$ a $\vec{x}_{r_3^i}(t)$:

$$f(\vec{x}_{r_2^i}, \vec{x}_{r_3^i}) = \begin{cases} \vec{x}_{r_2^i} - \vec{x}_{r_3^i} & \text{pro } [(\vec{x}_{r_2^i} \geq 0) \wedge (\vec{x}_{r_3^i} \geq 0)] \vee [(\vec{x}_{r_2^i} \leq 0) \wedge (\vec{x}_{r_3^i} \leq 0)] \\ \vec{x}_{r_2^i} + \vec{x}_{r_3^i} & \text{jinak} \end{cases} \tag{11.3}$$

Zápis byl pro přehlednost zjednodušen a platí $f(\vec{x}_{r_2^i}, \vec{x}_{r_3^i}) \Leftrightarrow f(\vec{x}_{r_2^i}(t), \vec{x}_{r_3^i}(t))$. Z rovnice 11.3 je patrné, že výpočet rozdílového vektoru závisí znaménku jednotlivých dimenzí rozdílových vektorů.

S odstupem času, navíc po optimalizaci parametrů DE, je přínos těchto modifikací diskutabilní, jelikož po vytvoření *meta–evoluce*, která slouží pro optimalizaci parametrů DE, dávají klasická schémata podobné výsledky, jako jejich modifikace. Více o optimalizaci parametrů DE diskutuje kapitola 9.

11.4 Podmínky zastavení

Optimalizace parametrů rovnice regulátoru vyžaduje velké množství ohodnocení kriteriální funkce. Počet generací DE byl experimentálně nastaven na 1500. Počet jedinců DE byl nastavován v intervalu 9 – 15. To znamená, že pro jednoho jedince TLTE se provádí až 22,5 tisíce simulací regulačního pochodu a ohodnocení kriteriální funkce! Tak vysoký počet ohodnocení sebou přináší velkou časovou náročnost výpočtu.

Pro urychlení výpočtu a nejen kvůli tomu, byla vytvořena metoda ART, která tím, že redukuje strom, přináší časovou úsporu během simulace. Více o metodě ART diskutuje kapitola 8.8.

Aby se zbytečně neprodlužovala doba optimalizace neznámých numerických parametrů, byly experimentálně vytvořeny zastavovací podmínky, které ukončí výpočet dříve, než po předem nastaveném počtu generací. Algoritmus zastavovacích podmínek je uveden níže, viz. algoritmus 7. Proměnná *OldBestValue* je hodnota fitness z předešlé generace

Algoritmus 7: Podmínky zastavení

```
Input: OldBestValue; BestValue; AverageValue
Result: EndAllowed: true or false
1  if (BestValue.Fitness is NaN) and (OldBestValue.Fitness is NaN) then
2      increasedOBV ⇐ percentage increase of OldBestValue.NodesComputedCountRate;
3      if BestValue.NodesComputedCountRate ≤ increasedOBV then
4          return true;
5      else
6          return false;
7      end
8  end
9  decreasedOldBestFitness ⇐ percentage decrease of OldBestValue.Fitness;
10 if BestValue.Fitness > decreasedOldBestFitness then
11     return true;
12 end
13 if (BestValue.Fitness is not NaN) and (AvgValue.Fitness is not NaN) and (BestValue.Fitness =
   AvgValue.Fitness) then
14     return true;
15 end
16 return false;
```

vytažená z bufferu (FIFO) starých nejlepších hodnot fitness v jednotlivých generacích. Tento buffer je typicky nastaven na 100 generací a na počátku inicializován na první nejlepší hodnotu fitness.

12 Experimentální část

Táto část disertační práce prezentuje výsledky experimentů. V následujících kapitolách budou prezentovány ukázky regulačních pochodů, včetně jejich vzájemného porovnání. V úvodu každé podkapitoly je vždy uvedena soustava pro kterou proběhla simulace a je popsáno co na této soustavě bylo pozorováno. Porovnávány jsou různé možnosti seřízení regulátorů, z nichž je větší váha kladena na evoluční metody. Nejdůležitější z této části jsou kapitoly pojednávající o optimalizaci struktury, včetně parametrů obecných regulátorů, (viz. kapitola 12.6).

12.1 Porovnání regulačních pochodů PSD regulátorů pro různé metody nastavení parametrů

Pro proporcionální soustavu danou přenosem (Eq. 12.1)

$$G_S(S) = \frac{2}{(5s+1)(2s+1)} e^{-12s} \tag{12.1}$$

bylo provedeno porovnání regulačních pochodů *neupravených* PSD regulátorů, jejichž parametry byly nastaveny metodou:

1. Inverze dynamiky (metoda požadovaného modelu) (viz. kapitola 10.9.3)
2. Ziegler-Nichols (metoda kritického stavu) (viz. kapitola 10.9.1)
3. Diferenciální evoluce (viz. kapitola 4)

Inverze dynamiky: Pro výpočet parametrů byl povolen relativní překmit $\kappa = 0{,}05$ (5%) Velikost vzorkovací periody byla zvolena $T = 2s \Leftarrow T < 0{,}32T_d = 0{,}32 \cdot 12 = 3{,}84$
Dle tabulky 23 je zesílení soustavy:

$$a = \frac{1}{\alpha T + \beta T_d} = \frac{1}{0{,}984 \cdot 2 + 1{,}944 \cdot 12} \doteq 0{,}04$$

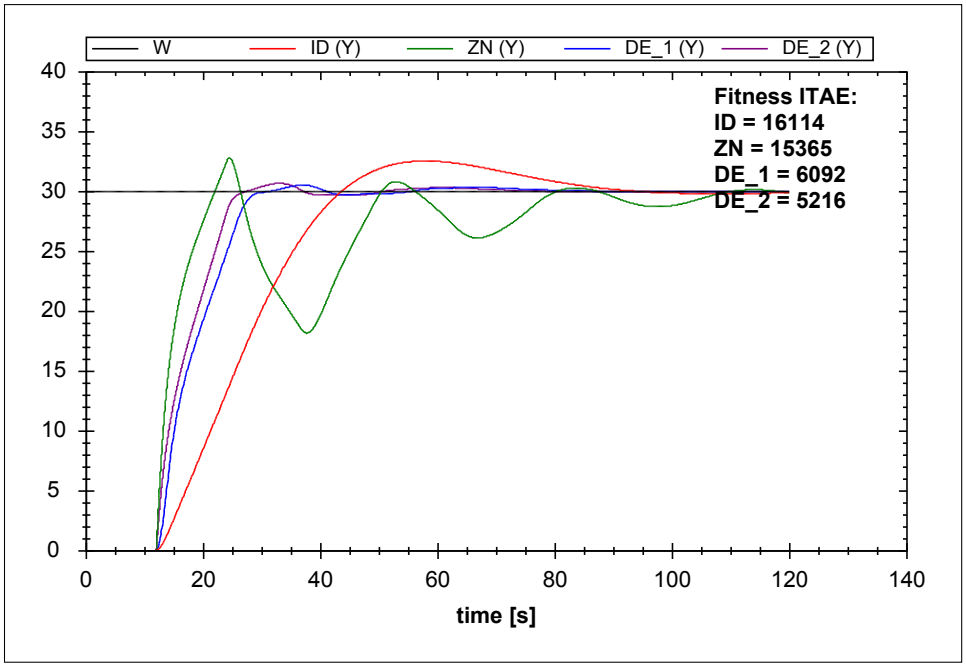
Z tabulky 23 se získá zesílení regulátoru k_R^*, integrační konstanta T_I^*, derivační konstanta T_D^*:

$$T_I^* = T_1 + T_2 - T = 5 + 2 - 2 = 5$$

$$T_D^* = \frac{T_1 T_2}{T_1 + T_2} - \frac{T}{4} = \frac{5 \cdot 2}{5+2} - \frac{2}{4} \doteq 0{,}93$$

$$k_R^* = \frac{a T_I^*}{k_S} = \frac{0{,}04 \cdot 5}{2} \doteq 0{,}1$$

Ziegler–Nichols: Soustava byla pomocí diskrétního P regulátoru s nízkou vzorkovací periodou přivedena na hranici stability. Z přechodové charakterisktiky soustavy byla odečetena kritická perioda kmitů $T_k = 36{,}48s$, kritické zesílení regulátoru bylo $k_{Rk} = 0{,}702$. Z tabulky 18 bylo určeno nastavení spojitého PID regulátoru:



ID: parametry PSD nastaveny metodou *inverze dynamiky*, **ZN:** parametry PSD nastaveny metodou *Ziegler-Nicholse*, **DE_1:** parametry PSD nastaveny pomocí *diferenciální evoluce* pro $T = 2s$, **DE_2:** parametry PSD nastaveny pomocí *diferenciální evoluce*, včetně T

Obrázek 58: Porovnání regulačních pochodů PSD regulátorů

$$r_0^* = 0{,}6k_{Rk} = 0{,}6 \cdot 0{,}702 = 0{,}4212$$

$$r_{-1}^* = 0{,}5T_k = 0{,}5 \cdot 36{,}48 = 18{,}24$$

$$r_1^* = 0{,}12T_k = 0{,}12 \cdot 36{,}48 = 4{,}3776$$

Z těchto hodnot byl podle tabulky 17 proveden převod metodou *ZOBD* na parametry přírustkového regulátoru zadaného ve tvaru (Eq. 10.42). Vzorkovací perioda byla zvolena $T = 2s$.

$$q_0 = k_R\left(1 + \frac{T_D}{T} + \frac{T}{T_I}\right) = 0{,}4212\left(1 + \frac{4{,}3776}{2} + \frac{2}{18{,}24}\right) = 1{,}3893$$

$$q_1 = -0{,}4212\left(1 + 2\frac{4{,}3776}{2}\right) = -2{,}2650$$

$$q_2 = k_R\frac{T_D}{T} = 0{,}4212 \cdot \frac{4{,}3776}{2} = 0{,}9219$$

Regulační obvod s regulátorem, jehož parametry byly vypočítané metodou *Ziegler–Nicholse* pro vzorkovací periodu $T = 2s$ je nestabilní. Proto byla experimentálně zvolena perioda $T = 0{,}1$ a parametry regulátoru byly přepočítány:

$$q_0 = k_R\left(1 + \frac{T_D}{T} + \frac{T}{T_I}\right) = 0{,}4212\left(1 + \frac{4{,}3776}{0{,}1} + \frac{0{,}1}{18{,}24}\right) = 18{,}8619$$

$$q_1 = -0{,}4212\left(1 + 2\frac{4{,}3776}{0{,}1}\right) = -37{,}2981$$

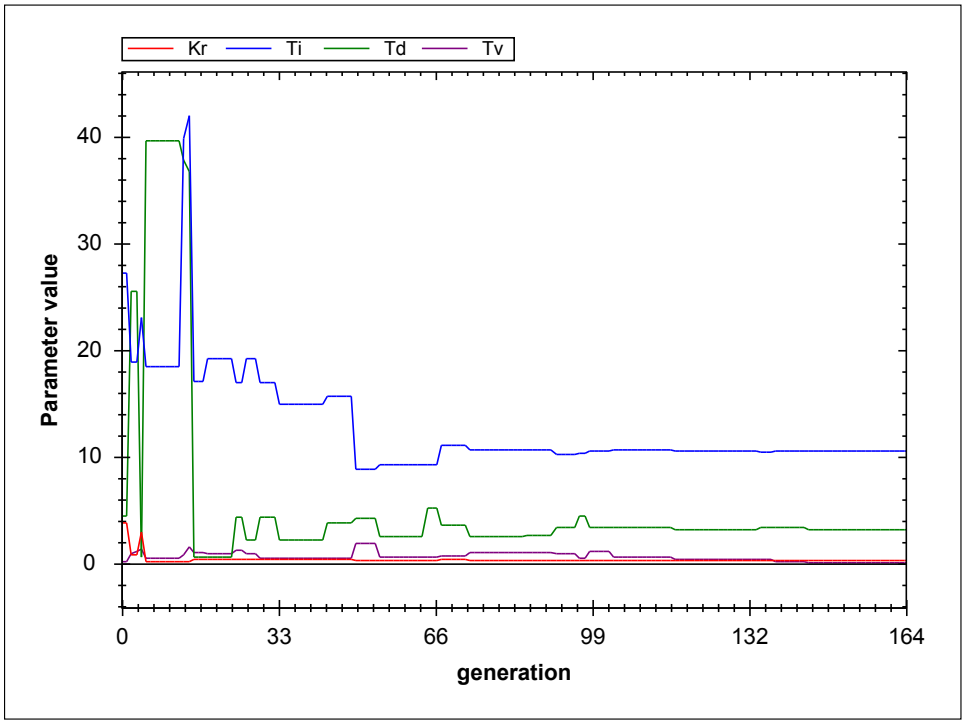
$$q_2 = k_R\frac{T_D}{T} = 0{,}4212 \cdot \frac{4{,}3776}{0{,}1} = 18{,}4384$$

Regulační pochod s tímto nastavením je ukázán na obrázku 58.

Diferenciální evoluce: Parametry diferenciální evoluce byly nastaveny na: G = 1500, NP = 9, F = 1,07, CR = 0,58, schéma: DE/Rand/1_mod, zastavovací podmínky podle kapitoly 11.4.

Výsledek optimalizace pro vzorkovací periodu $T = 2s$ je: $k_R = 0{,}247455$, $T_I = 9{,}560313$, $T_D = 3{,}773534$

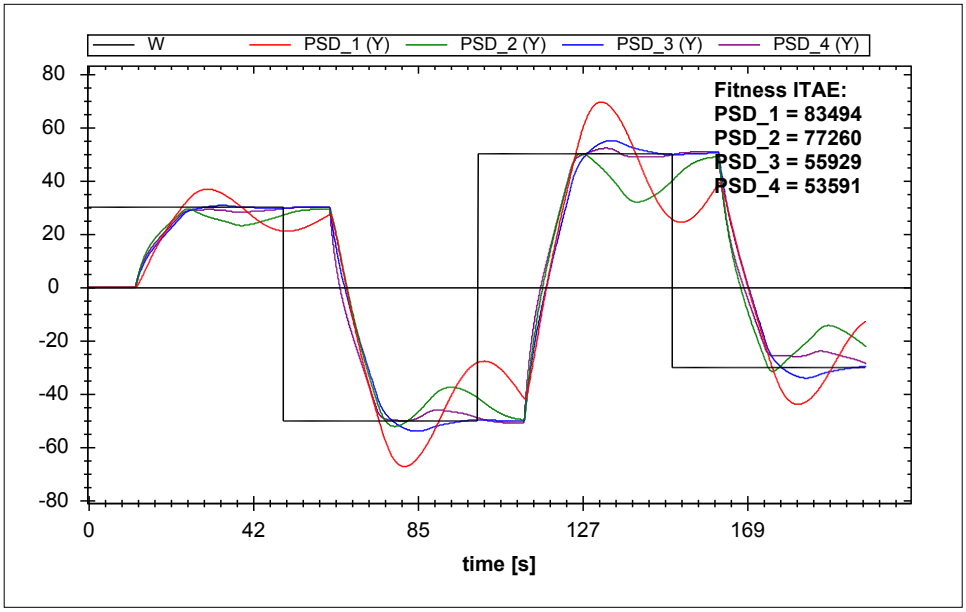
Výsledek optimalizace včetně vzorkovací periody je: $k_R = 0{,}313681$, $T_I = 10{,}661475$, $T_D = 3{,}630165$, $T = 0{,}021196$



Kr: zesílení PSD regulátoru **Ti:** integrační konstanta PSD regulátoru **Td:** derivační konstanta PSD regulátoru **Tv:** vzorkovací perioda

Obrázek 59: Průběh evoluce parametrů PSD regulátoru v MDE

Jednotlivé regulační pochody jsou ukázány na obrázku 58. Křivka označená *ID* označuje PSD regulátor, jehož parametry byly nastaveny pomocí *metody inverze dynamiky* pro $T = 2s$. Křivka *ZN* označuje regulátor jehož parametry byly nastaveny metodou *Zigler–Nicholse* pro $T = 0{,}1s$. Křivka *DE_1* označuje regulátor jehož parametry (k_R, T_I, T_D) byly optimalizovány pomocí *diferenciální evoluce* pro $T = 2s$. Pro srovnání s předchozím nastavením bylo ponecháno vzorkování na $T = 2s$. Křivka *DE_2* označuje regulátor,



PSD_1: nejstarší generace (≈ 50.generace), ... **PSD_4:** nejmladší generace (≈ 600.generace)

Obrázek 60: Průběhy regulací v různých generacích evoluce

který byl rovněž nastaven pomocí DE. V tomto případě byly optimalizovány parametry k_R, T_I, T_D, T.

V obou případech nastavení parametrů pomocí DE je vidět, že výsledný regulátor má lepší ohodnocení, než regulátor nastaven metodou *inverze dynamiky*, případně metodou *Ziegler-Nicholse*. V případě PSD regulátoru označeného *ID* je doba regulace $T_R = 72{,}7s$, doba regulace regulátoru nastaveného metodou Ziegler-Nicholse je $T_R = 75{,}6s$. Regulátor se vzorkováním $T = 2s$, který byl nastaven pomocí DE má dobu regulace $T_R = 26{,}7$. Regulátor, jemuž byla optimalizována i perioda vzorkování má dobu regulace dokonce $T_R = 24{,}6$.

V pravém horním rohu tohoto obrázku jsou vidět výsledky hodnocení jednotlivých regulačních pochodů. Hodnocení je pro jednoduchost ukázáno metodou *ITAE*, i když kriteriální funkce DE hodnotila regulační pochod podle jiných kritérií. Z uvedeného obrázku je mj. patrné, že kritérium *ITAE* k hodnocení kvality regulace nestačí. Viz. regulátor seřízen metodou *inverze dynamiky* vs. *Ziegler-Nichols*.

Ukázka části průběhu evoluce parametrů je zobrazena na obrázku 59 a ukázka regulačních pochodů z vybraných fází evolučního procesu pro PSD regulator, jenž byl v předchozím textu označen *DE_2* je ukázána na obrázku 60.

12.2 Porovnání regulačních pochodů Takahashiho regulátorů pro různé metody nastavení parametrů

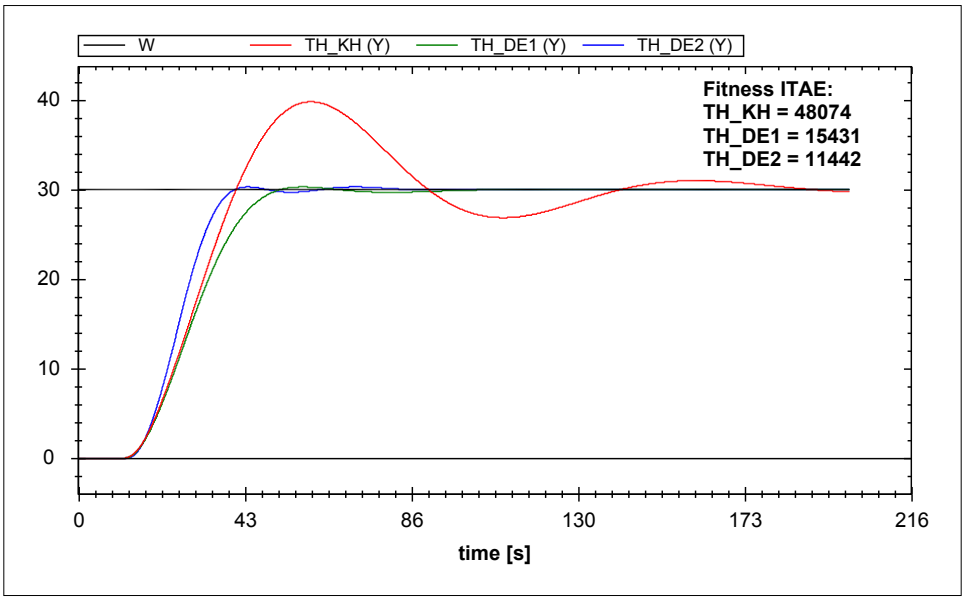
Pro proporcionální soustavu z kapitoly 12.1 danou přenosem (Eq. 12.1)

$$G_S(S) = \frac{2}{(5s+1)(2s+1)} e^{-12s} \tag{12.2}$$

bylo provedeno porovnání regulačních pochodů *Takahashiho regulátorů*, jejichž parametry byly nastaveny:

1. z kritických hodnot přechodové charakteristiky (viz. kapitola 10.9.1)
2. diferenciální evolucí (viz. kapitola 4)

Nastavení z kritických hodnot: Kritické hodnoty již byly určeny v kapitole 12.1. Z těchto hodnot je možné přepočítat parametry pro Takahashiho regulátor dle tabulky 19. Vzorkovací perioda $T = 2s$ byla zvolena shodně s předchozím příkladem, jehož výsledky byly prezentovány na obrázku 58.



TH_KH: parametry Takahashiho regulátoru nastaveny z kritických hodnot pro $T = 2s$, **TH_DE1:** parametry Takahashiho regulátoru nastaveny z kritických hodnot pro $T = 2s$, **TH_DE2:** parametry Takahashiho regulátoru nastaveny z kritických hodnot pro $T = 0{,}1s$

Obrázek 61: Porovnání regulačních pochodů Takahashiho regulátoru

Výpočet parametrů z kritických hodnot pro Takahashiho regulátor je uveden níže:

$$k_R^* = 0{,}6k_{Rk}\left(1 - \frac{T}{T_k}\right) = 0{,}6 \cdot 0{,}702 \cdot \left(1 - \frac{2}{36{,}48}\right) = 0{,}3981$$

$$T_I^* = \frac{k_R^* T_k}{1{,}2k_{Rk}} = \frac{0{,}3981 \cdot 36{,}48}{1{,}2 \cdot 0{,}702} = 17{,}2397$$

$$T_D^* = \frac{3}{40} \cdot \frac{k_{Rk}T_k}{k_R^*} = \frac{3}{40} \cdot \frac{0{,}702 \cdot 36{,}48}{0{,}3981} = 4{,}8246$$

Diferenciální evoluce: Pomocí DE byly parametry Takahashiho regulátoru optimalizovány pro periodu vzorkování $T = 2s$ a $T = 0{,}1s$. Parametry diferenciální evoluce byly

nastaveny na: G = 1500, NP = 9, F = 1,07, CR = 0,58, schéma: DE/Rand/1_mod, (viz. kapitola 11.3), zastavovací podmínky podle kapitoly 11.4.

Výsledek optimalizace pro vzorkovací periodu $T = 2s$ je: $k_R = 39{,}074646$, $T_I = 1779{,}599509$, $T_D = 94{,}704670$

Výsledek optimalizace pro vzorkovací periodu $T = 0{,}1s$ je: $k_R = 0{,}745325$, $T_I = 22{,}388003$, $T_D = 7{,}552242$

Srovnání obou regulačních pochodů je ukázáno na obrázku 61. Křivka označená *TH_KH* je Takahashiho regulátor, jehož parametry byly vypočítány podle kritických hodnot. Výpočet je proveden výše. Křivka označená *TH_DE1* a *TH_DE2* je výsledek regulačního procesu Takahashiho regulátoru, jenž byl optimalizován pomocí DE. *TH_DE1* je výsledek optimalizace pro $T = 2s$, *TH_DE2* je výsledek optimalizace pro $T = 0{,}1s$.

V obou případech nastavení parametrů pomocí DE je vidět, že výsledný regulátor má lepší ohodnocení, než regulátor nastaven algebraickou metodou z *kritických hodnot* přechodové charakteristiky. V případě Takahashiho regulátoru označeného *TH_HK* je doba regulace $T_R = 128{,}6s$, doba regulací regulátorů *TH_DE1* a *TH_DE2* je $T_R = 45{,}9s$, resp. $T_R = 36{,}8s$.

V pravém horním rohu tohoto obrázku jsou zobrazeny výsledky hodnocení jednotlivých regulačních pochodů. Pro jednoduchost a názornost je kritérium hodnocení regulačních pochodů zobrazeno pro metodu *ITAE*.

12.3 Porovnání PSD regulátorů a obecného regulátoru (diferenciální rovnice stejného řádu) – proporcionální soustava

Pro proporcionální soustavu z kapitoly 12.1 a 12.2 danou přenosem:

$$G_S(S) = \frac{2}{(5s+1)(2s+1)} e^{-12s} \tag{12.3}$$

bylo provedeno porovnání regulačních pochodů *PSD regulátoru* a *obecného regulátoru*, jejichž parametry byly nastaveny pomocí DE. V obou případech byly optimalizovány parametry rovnice (Eq. 12.4), (viz. kapitola 10.6):

$$u(kT) = u[(k-1)T] + q_0 e(kT) + q_1 e[(k-1)T] + q_2 e[(k-2)T] \tag{12.4}$$

V prvním případě byla provedena optimalizace parametrů PSD regulátoru k_R, T_I, T_D, T, viz. tabulka 17, ve druhém případě byla provedena přímá optimalizace parametrů q_0, q_1, q_2 pro vzorkovací periodu shodnou s periodou nalezenou pro PSD regulátor.

Výsledek optimalizace parametrů PSD regulátoru pro tuto soustavu byl prvně diskutován v kapitole 12.1 (na obrázku 58 je pojmenována *DE_2*, na obrázku 62 pojmenována *PSD_1*). Tyto parametry jsou: $k_R = 0{,}313681$, $T_I = 10{,}661475$, $T_D = 3{,}630165$, $T = 0{,}021196$

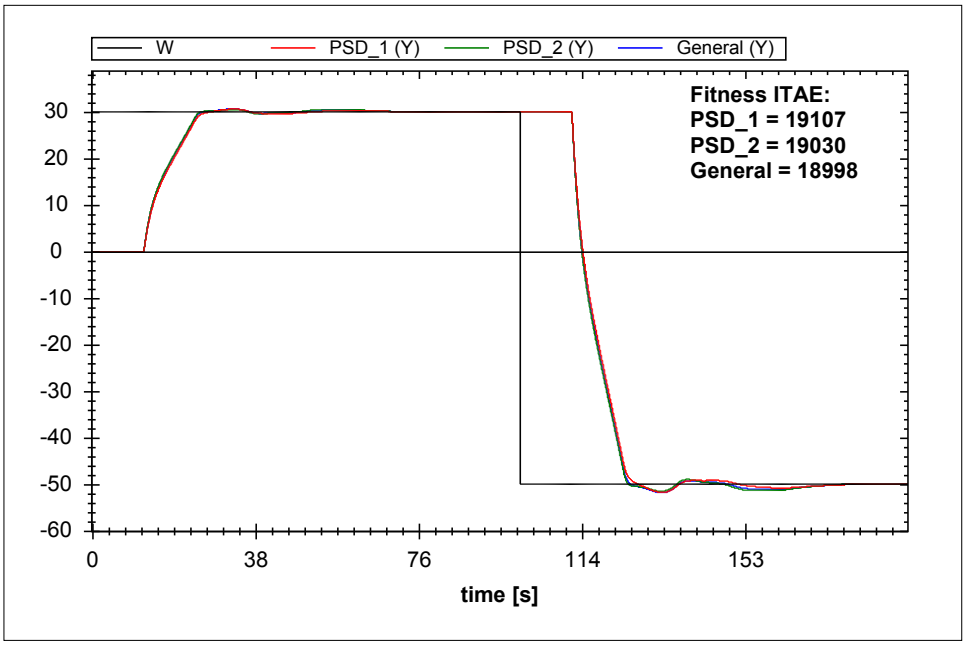
Nyní byly parametry optimalizovány za pomocí kriteriální funkce, která hodnotila kvalitu regulace podle *ITAE* (na obrázku 62 pojmenována *PSD_2*). Výsledkem optimalizace je nastavení regulátoru: $k_R = 0{,}317715$, $T_I = 10{,}492045$, $T_D = 3{,}932147$.

Výsledek přímé optimalizace parametrů q_0, q_1, q_2 je: $q_0 = 57{,}307437$, $q_1 = -114{,}297034$, $q_2 = 56{,}990233$. Tvar řídicí rovnice tedy je: $u(kT) = u[(k-1)T] + 57{,}307437e(kT) - 114{,}297034e[(k-1)T] + 56,990233e[(k-2)T]$

Regulační pochod s těmito parametry je na obrázku 62 označen *General*. I v tomto případě byla použita kriteriální funkce hodnotící podle *ITAE*.

Ve všech třech případech byly parametry optimalizovány pro vzorkovací periodu $T = 0{,}021196$.

Parametry diferenciální evoluce byly v případě optimalizace parametrů *PSD regulátoru* nastaveny na: G = 1500, NP = 9, F = 1,07, CR = 0,58, schéma: DE/Rand/1_mod, zastavovací podmínky podle kapitoly 11.4. Pro přímou optimalizaci parametrů q_0, q_1, q_2 byl zvýšen počet jedinců DE na 20, což sice způsobilo prodloužení optimalizace, ale vedlo k prohledání většího prostoru řešení a nalezení kvalitnějšího nastavení.



PSD_1: PSD regulátor (parametry k_R, T_I, T_D, T optimalizovány DE, kritéria hodnocení: (viz. kapitola 11.2)), **PSD_2:** PSD regulátor (parametry k_R, T_I, T_D optimalizovány DE pro $T = 0{,}021196$, kritérium hodnocení: *ITAE*), **General:** Obecný regulátor (parametry q_0, q_1, q_2 optimalizovány DE pro $T = 0{,}021196$, kritérium hodnocení: *ITAE*)

Obrázek 62: Porovnání PSD regulátoru a obecného regulátoru stejného řádu

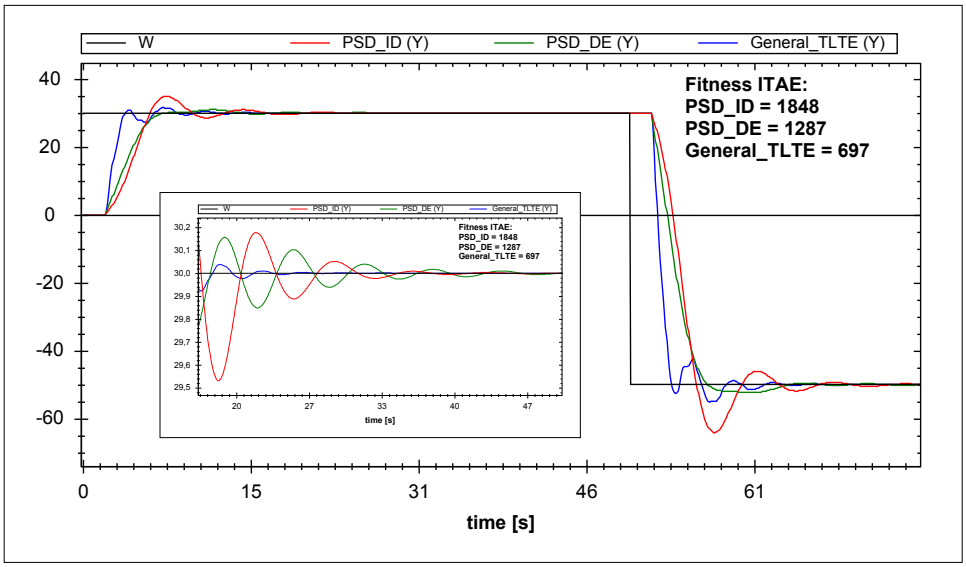
V pravém horním rohu obrázku je umístěno porovnání kvality regulace podle kritéria *ITAE*. Z obrázku je vidět, že všechny zobrazené křivky jsou si svým průběhem podobné. Z toho mimo jiné vyplývá, že přepočet parametrů PSD regulátoru dle tabulky 17 je optimální a že pro tuto soustavu není možno získat kvalitnější regulační pochod v případě použití diferenciální rovnice ve tvaru 10.42 (případně 12.4).

12.4 Porovnání PSD regulátoru a obecného regulátoru – kmitavá proporcionální soustava

Pro proporcionální soustavu s kmitavým průběhem danou přenosem

$$G_S(S) = \frac{k_S}{T_0^2 s^2 + 2\xi T_0 s + 1} e^{-T_d s} \tag{12.5}$$

s parametry $k_S = 1$, $T_0 = 1$, $\xi = 0{,}1$, $T_d = 2s$ bylo provedeno porovnání regulačních pochodů *PSD regulátoru* a *obecného regulátoru*. Parametry *PSD regulátoru* byly v prvním případě nastaveny metodou *inverze dynamiky* (viz. kapitola 10.9.3), optimalizovány pomocí DE (viz. kapitola 4), struktura a parametry *obecného regulátoru* byly optimalizovány TLTE (viz. kapitola 8).



PSD_ID: PSD regulátor (parametry k_R, T_I, T_D, T nastaveny metodu *inverze dynamiky*), **PSD_DE:** PSD regulátor (parametry k_R, T_I, T_D optimalizovány DE pro $T = 0{,}1s$, kritérium hodnocení: *ITAE*), **General_TLTE:** Obecný regulátor (struktura a parametry optimalizovány TLTE pro $T = 0{,}1s$, kritérium hodnocení: *ITAE*)

Obrázek 63: Porovnání PSD regulátorů a obecného regulátoru s obecnou řídicí rovnicí

Inverze dynamiky: Pro výpočet parametrů byl povolen relativní překmit $\kappa = 0{,}05$ (5%). Velikost vzorkovací periody byla zvolena $T = 0{,}1s \Leftarrow T < 0{,}32T_d = 0{,}32 \cdot 2 = 0{,}64$. Dle tabulky 23 je zesílení soustavy:

$$a = \frac{1}{\alpha T + \beta T_d} = \frac{1}{0{,}984 \cdot 0{,}1 + 1{,}944 \cdot 2} \doteq 0{,}2508$$

Z tabulky 23 se získá zesílení regulátoru k_R^*, integrační konstanta T_I^*, derivační konstanta T_D^*:

$$T_I^* = 2\xi_0 T_0 - T = 2 \cdot 0{,}1 \cdot 1 - 0{,}1 = 0{,}1$$

$$T_D^* = \frac{T_0}{2\xi_0} - \frac{T}{4} = \frac{1}{2 \cdot 0{,}1} - \frac{0{,}1}{4} \doteq 4{,}975$$

$$k_R^* = \frac{aT_I^*}{k_S} = \frac{0{,}2508 \cdot 0{,}1}{1} \doteq 0{,}02508$$

Diferenciální evoluce: Parametry diferenciální evoluce byly nastaveny na: G = 1500, NP = 9, F = 1,07, CR = 0,58, schéma: DE/Rand/1_mod, zastavovací podmínky podle kapitoly 11.4.

Vzorkovací perioda byla v tomto případě zvolena stejná jako v případě nastavení metodou *inverze dynamiky*, tj. $T = 0{,}1s$. Výsledek optimalizace je: $k_R = 0{,}068693$, $T_I = 0{,}270703$, $T_D = 3{,}613283$.

Dvoufázová transplantační evoluce: Parametry *TE* byly nastaveny: Počet jedinců = 50, počet nově vytvořených jedinců v každé generaci = 25, maximální počet uzlů jedince = 30, pravděpodobnost křížení = 90%, pravděpodobnost mutace rodičů = 25%, pravděpodobnost mutace potomků = 25%, metoda ART, (viz. kapitola 8.8), byla aktivní, parametry *MDE* byly nastaveny: G = 1500, NP = 9, F = 1,07, CR = 0,58, schéma: DE/Rand/1_mod, zastavovací podmínky podle kapitoly 11.4. Optimalizace byla z důvodu časové náročnosti přerušena po 72 generacích TLTE (25,5 hodin CPU času, 3GHz CPU).

Vzorkovací perioda byla nastavena stejná, jako v obou předchozích případech, tj. $T = 0{,}1s$. Výsledná optimalizovaná řídicí rovnice obecného regulátoru je:

```
Uk=(((Yk_3*15,3645727059621)+(((Ek*3,80273057221998)-
   ((Ek-0,000331083646885983)+((Uk_5*0,571665466205751)+
   (Yk_4*13,7928925600503))))-(Ek_3*0,958174106578814)))-Ek_4)
```

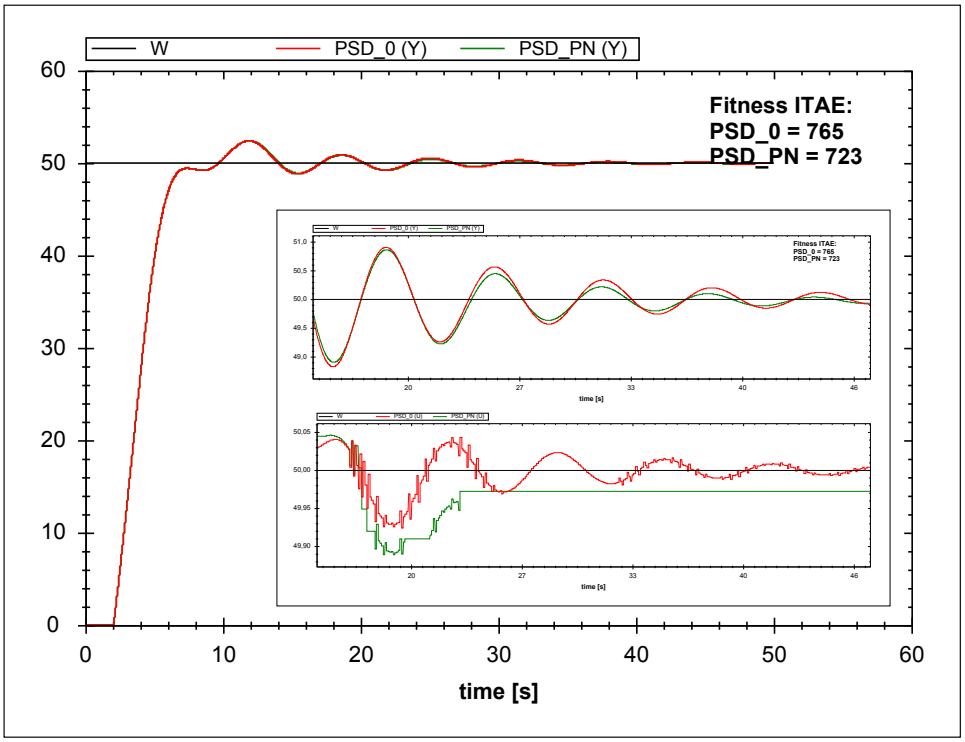
Rovnice byla ponechána v původním tvaru v jakém jej vrátila TLTE. Význam jednotlivých symbolů.

Kriteriální funkce byla nastavena pro hodnocení kvality regulace metodou *ITAE*, což se projevilo hlavně v případě optimalizace *obecného regulátoru* metodou TLTE, který vykazuje větší známky kmitání. V případě zapojení hodnocení jenž penalizuje překmit, apod. byl výsledek optimalizace rovnice *obecného regulátoru* ještě kvalitnější. Nicméně i tak je z obrázku 63 patrné, že dle kritéria *ITAE* má nejlepší ohodnocení. Jiná kritéria hodnocení kvality regulace byly popsány v kapitole 11.2.

Na obrázku 63 jsou zobrazeny 3 regulační pochody. Regulační pochod označený *PSD_ID* je výsledkem nastavení *PSD regulátoru* dle metody *inverze dynamiky*. *PSD_DE* je výsledek nastavení parametrů K_R, T_I, T_D *PSD regulátoru* pomocí *DE*. *General_TLTE* je výsledek optimalizace struktury i parametrů *obecného regulátoru* pomocí metody TLTE.

V pravém horním rohu jsou zobrazeny výsledky ohodnocení regulačního pochodu metodou *ITAE*. V levém dolním rohu je detail regulačního pochodu v časovém intervalu od ≈ 15 do ≈ 50 vteřin. Velký obrázek i jeho detailní zobrazení barevně korespondují.

Na obrázku 64 jsou ukázány dva regulační pochody pro jeden regulátor se stejnými parametry. *PSD_0* je označen regulátor jenž měl za úkol docílit nulové regulační odchylky, *PSD_PN* je regulátor, který měl nastaveno pásmo necitlivosti na $\pm 5\%$ žádané hodnoty při zpětné diferenci regulované veličiny $\nabla y(kT) = 0{,}1$. V pravé dolní části obrázku jsou zobrazeny detaily regulačních pochodů, včetně akčních zásahů regulátorů v intervalu od ≈ 13 do ≈ 50 sekund. Průběh regulovaných veličin je v horní části detailního obrázku, průběh akčních veličin je v dolní části detailního obrázku. Z dolní části je patrné, že regulátor *PSD_PN* drží konstantní velikost akčního zásahu od doby, kdy regulovaná veličina dosáhla nastaveného pásma necitlivosti. Stojí za povšimnutí, že regulátor s nastaveným pásmem necitlivosti (*PSD_PN*) má paradoxně lepší ohodnocení *ITAE*, než druhý regulátor



PSD_0: PSD regulátor (regulace na nulovou regulační odchylku), **PSD_PN:** PSD regulátor (regulace s pásmem necitlivosti)

Obrázek 64: Proporcionální kmitavá soustava (pásmo necitlivosti)

(*PSD_0*), jehož úkolem bylo dosažení nulové regulační odchylky (kriteriální funkce penalizuje regulační odchylku i v pásmu necitlivosti). Průběhy regulovaných veličin a akčních zásahů v detailním zobrazení vzájemně korespondují s hlavním obrázkem.

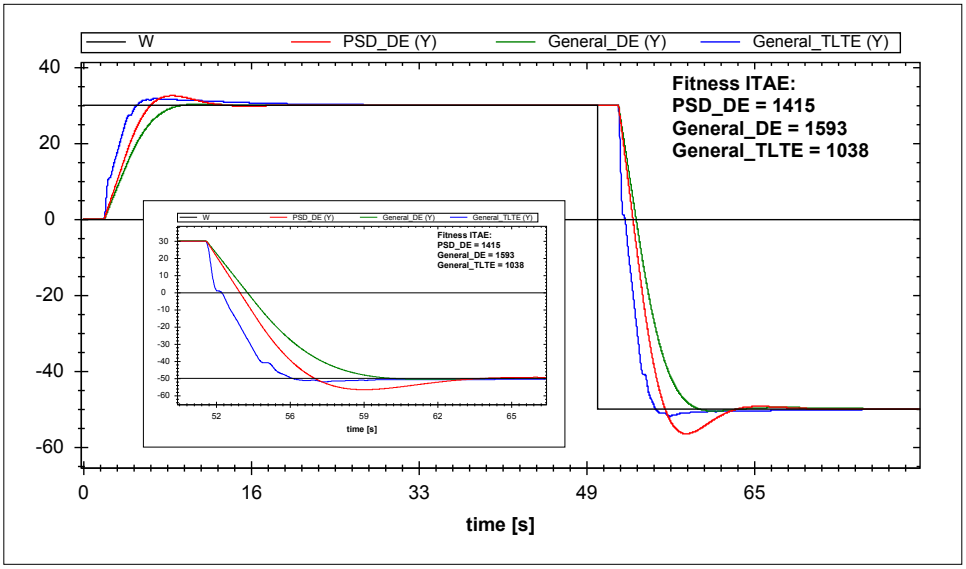
12.5 Porovnání regulátorů – integrační soustava

Pro integrační soustavu zadanou přenosem:

$$G_S(S) = \frac{5}{s(9s+1)} e^{-2s} \tag{12.6}$$

bylo provedeno porovnání regulačních pochodů *PSD regulátoru*, *obecného regulátoru* s diferenciální rovnicí druhého řádu, ve tvaru (Eq. 10.42) a *obecného regulátoru* s obecnou řídicí rovnicí. Vzorkovací frekvence byla ve všech případech optimalizace zvolena $T = 0{,}1s$.

Diferenciální evoluce: Pro optimalizaci parametrů *PSD regulátoru* (k_R, T_I, T_D) a pro přímou optimalizaci parametrů q_0, q_1, q_2 byla použita DE s nastavením G = 1500, NP = 15, F = 1,07, CR = 0,58, schéma: DE/Rand/1_mod, zastavovací podmínky podle kapitoly 11.4. Výsledkem optimalizace je nastavení parametrů *PSD regulátoru* $k_R = 5{,}299811 \cdot 10^{-2}$, $T_I = 3813{,}553351$, $T_D = 9{,}082405$. Výsledek optimalizace parametrů řídicí rovnice (Eq. 10.42) *obecného regulátoru* je $q_0 = 3{,}997177$, $q_1 = -7{,}952002$, $q_2 = 3{,}954828$.



PSD_DE: PSD regulátor (parametry k_R, T_I, T_D optimalizovány DE pro $T = 0{,}1s$, kritéria hodnocení: (viz. kapitola 11.2)), **General_DE:** obecný regulátor, rovnice (Eq. 10.42) (parametry q_0, q_1, q_2 optimalizovány DE pro $T = 0{,}1s$, kritéria hodnocení: (viz. kapitola 11.2)), **General_TLTE:** Obecný regulátor (struktura a parametry optimalizovány TLTE pro $T = 0{,}1s$, kritéria hodnocení: (viz. kapitola 11.2))

Obrázek 65: Porovnání PSD regulátoru, obecného regulátoru s řídící rovnicí druhého řádu a obecného regulátoru s obecnou řídicí rovnicí

Dvoufázová transplantační evoluce: Pro optimalizaci struktury a parametrů řídicí rovnice *obecného regulátoru* byla použita metoda TLTE. Parametry *TE* byly nastaveny: Počet jedinců = 50, počet nově vytvořených jedinců v každé generaci = 25, maximální počet uzlů jedince = 30, pravděpodobnost křížení = 90%, pravděpodobnost mutace rodičů = 25%, pravděpodobnost mutace potomků = 25%, metoda ART, (viz. kapitola 8.8), byla aktivní. Pro optimalizaci struktury byla využita kompletní sada pravidel definovaná v kapitole 11.1, kromě terminálního symbolu „/“, reprezentující funkci dvou proměnných a dále terminálních symbolů $dE_k, \ldots, dE_{k-4}$, reprezentujících zpětné diference regulačních odchylek a pravidel s tím souvisejících. Parametry *MDE* byly nastaveny: G = 1500, NP = 9, F = 1,07, CR = 0,58, schéma: DE/Rand/1_mod, zastavovací podmínky podle kapitoly 11.4. Výsledkem optimalizace je řídicí rovnice *obecného regulátoru* ve tvaru:

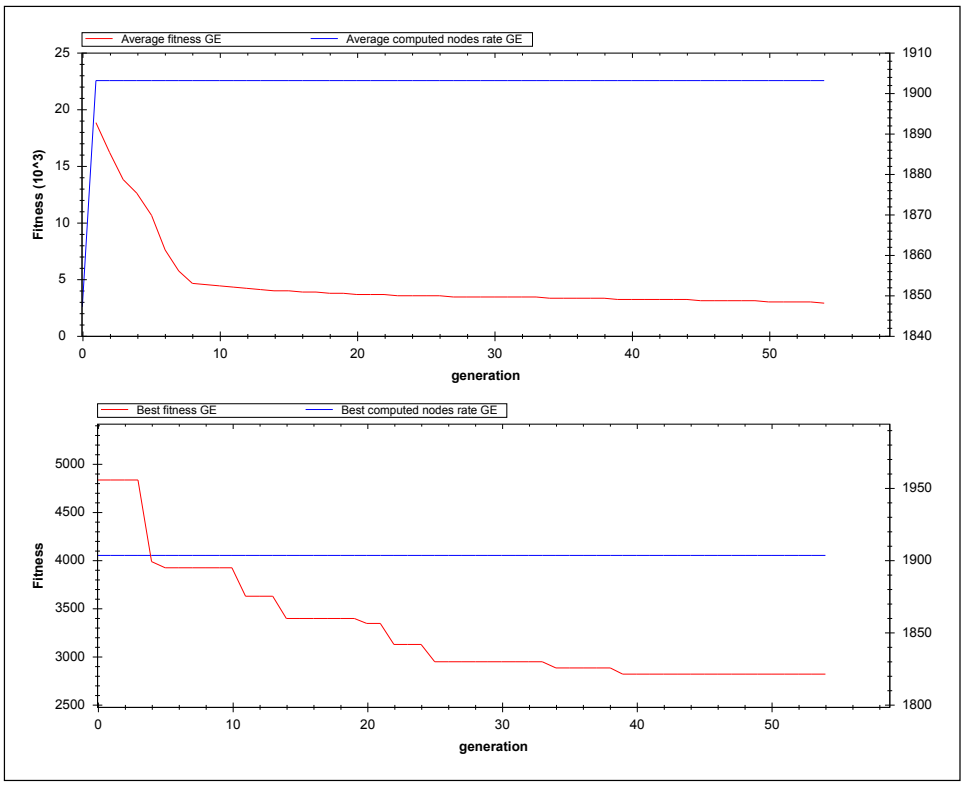
```
Uk = (Ek_5+(((-((Ek_5*(-4,69210604912152))-(-((((Yk*5,14847786853854)*
     Yk_3*1,59643919322167))+((Ek_5+22,2867094434306)*Ek_2))*
     (Uk_2*(-3,80995100289523E-06))))))+((11,9436857350138*Ek)+
     (((-0,322492294234783)*Uk_3)+((-18,5463669625012)*Ek_2))))+
     (Ek_1+(-0,00399257533723234))))
```

Rovnice byla ponechána v původním tvaru v jakém jej vrátila TLTE. Význam jednotlivých symbolů je uveden v dodatcích.

Kriteriální funkce použitá pro ohodnocení kvality jedinců v DE, TLTE hodnotila podle kritérií uvedených v kapitole 11.2. Z těchto kritérií byla vyloučena penalizace kmitání akčního zásahu a penalizace překmitu regulované veličiny.

Počet generací DE byl v případě optimalizace parametrů *PSD regulátoru* (k_R, T_I, T_D) 232. V případě optimalizace parametrů *obecného regulátoru* q_0, q_1, q_2 byl počet generací 1442. V průběhu vývoje vyšlo najevo, že optimalizace parametrů *obecného regulátoru* je ve všech případech časově náročnější, než optimalizace parametrů *PSD regulátoru*. V tomto případě byla optimalizace *obecného regulátoru* přibližně 6,5 krát náročnější na počet generací, než optimalizace parametrů *PSD regulátoru*.

Optimalizace struktury a parametrů *obecného regulátoru* metodou TLTE byla z důvodu časové náročnosti přerušena po 54 generacích TLTE (30,7 hodin CPU času, 3GHz CPU). Graf průběhu optimalizace je ukázán na obrázku 66. V horní části obrázku je ukázáno průměrné fitness v populaci, v dolní části je nejlepší fitness v populaci. Význam jednotlivých křivek je popsán v kapitole 11.2.1.

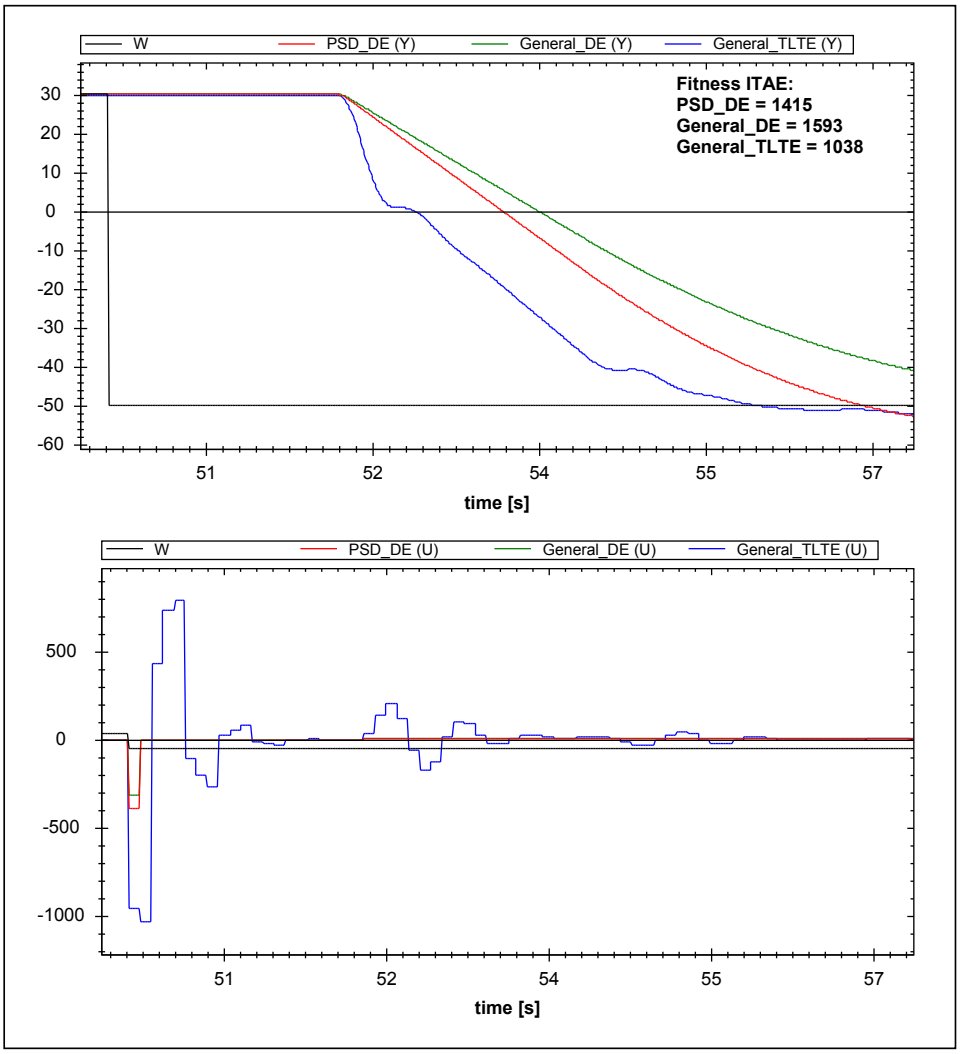


Horní graf: průměrné fitness v populaci, **Dolní graf:** nejlepší fitness v populaci, **modrá + červená křivka:** (viz. kapitola 11.2.1)

Obrázek 66: Průběh optimalizace struktury a parametrů obecného regulátoru metodou TLTE

Výsledné regulační pochody je možno porovnat na obrázku 65 a 67. Na těchto obrázcích jsou ukázány tři regulační pochody. *PSD_DE* je *PSD regulátor* s parametry (k_R, T_I, T_D), *General_DE* je *obecný regulátor* s parametry (q_0, q_1, q_2), *General_TLTE* je *obecný regulátor*, jehož struktura a parametry byly optimalizovány TLTE. V levé dolní části obrázku 65 je zobrazen detail regulačních pochodů od ≈ 51 do ≈ 66 sekund.

Na obrázku 67 je zobrazen detail akčních zásahů jednotlivých typů regulátorů. V horní části obrázku jsou zobrazeny průběhy regulovaných veličin, v dolní části jsou akční zásahy

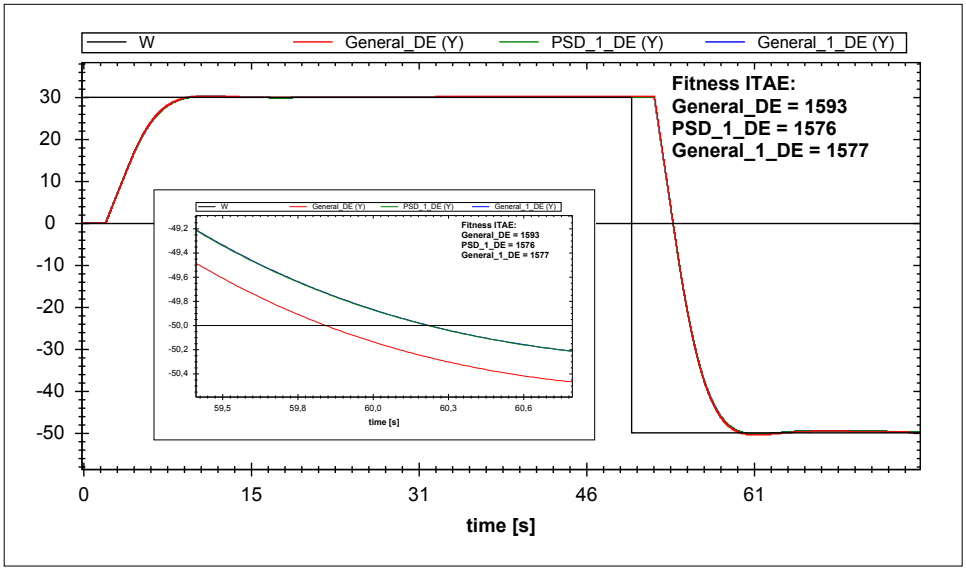


PSD_DE: PSD regulátor (parametry k_R, T_I, T_D optimalizovány DE pro $T = 0{,}1s$, kritéria hodnocení: (viz. kapitola 11.2)), **General_DE:** obecný regulátor, rovnice (Eq. 10.42) (parametry q_0, q_1, q_2 optimalizovány DE pro $T = 0{,}1s$, kritéria hodnocení: (viz. kapitola 11.2)), **General_TLTE:** Obecný regulátor (struktura a parametry optimalizovány TLTE pro $T = 0{,}1s$, kritéria hodnocení: (viz. kapitola 11.2))

Obrázek 67: Porovnání akčních zásahů PSD regulátoru, obecného regulátoru s řídící rovnicí druhého řádu a obecného regulátoru s obecnou řídicí rovnicí

regulátorů. Jednotlivé průběhy v horním a dolním obrázku barevně korespondují. Tyto barvy jsou shodné také s obrázkem 65. Z tohoto obrázku je vidět, že akční zásah obecného regulátoru s obecnou strukturou řídicí rovnice *General_TLTE* vykazuje známky kmitání. To je způsobeno tím, že z kriteriální funkce byla vyloučena penalizace kmitání akčního zásahu.

Na obrázku 68 je ukázáno porovnání regulačních pochodů jednoho *PSD regulátoru* označeného *PSD_1_DE* a dvou obecných regulátorů s diferenciální rovnicí druhého řádu (Eq. 10.42), označených *General_DE* a *General_1_DE*. Regulátor *General_DE* je totožný s



General_DE: obecný regulátor, rovnice (Eq. 10.42) (parametry q_0, q_1, q_2 optimalizovány DE pro $T = 0{,}1s$, kritéria hodnocení: (viz. kapitola 11.2) bez penalizace překmitu), **General_1_DE:** obecný regulátor, rovnice (Eq. 10.42) (parametry q_0, q_1, q_2 optimalizovány DE pro $T = 0{,}1s$, kritéria hodnocení: (viz. kapitola 11.2) s penalizací překmitu), **PSD_1_DE:** PSD regulátor (parametry k_R, T_I, T_D optimalizovány DE pro $T = 0{,}1s$, kritéria hodnocení: (viz. kapitola 11.2) s penalizací překmitu)

Obrázek 68: Porovnání PSD regulátoru, obecného regulátoru s řídící rovnicí druhého řádu, kriteriální funkce s penalizací překmitu regulované veličiny

regulátorem z obrázků 65, 67. Zbylé dvě nastavení regulátorů (*PSD_1_DE*, *General_1_DE*) byly optimalizovány za pomocí DE, jejíž kriteriální funkce penalizovala překmit regulované veličiny, (viz. kapitola 11.2). Cílem bylo ověřit, jestli je *PSD regulátor* schopen nastavením svých parametrů k_R, T_I, T_D zajistit stejně kvalitní regulační pochod, jako regulátor jehož konstanty q_0, q_1, q_2, (Eq. 10.42) byly nastaveny přímo. Na obrázku 68 je v pravém horním rohu zobrazeno hodnocení regulačních pochodů kritériem *ITAE*. V levém dolním rohu je zobrazen detail regulačního pochodu v intervalu od $\approx 59{,}4$ do $\approx 60{,}7$ sekund. Jak je vidět, tak ohodnocení *ITAE* obou regulátorů *PSD_1_DE*, *General_1_DE* je totožné a regulační pochody obou regulátorů se překrývají. Z toho vyplývá, že *PSD regulátor* je schopen zajistit nastavením svých parametrů stejně kvalitní regulační pochod jako obecný regulátor s řídící rovnicí (Eq. 10.42), jehož parametry q_0, q_1, q_2 byly optimalizovány přímo.

Optimalizace parametrů *PSD_1_DE* a *General_1_DE* byla provedena DE s nastavením stejným jako v předchozích případech. Výsledné nastavení obou regálátorů je pro *PSD_1_DE*: $k_R = 4{,}186242 \cdot 10^{-2}$, $T_I = 6901{,}866396$, $T_D = 9{,}418879$. Výsledek optimalizace parametrů řídicí rovnice (Eq. 10.42) *obecného regulátoru* je $q_0 = 3{,}983159$, $q_1 = -7{,}924456$, $q_2 = 3{,}941298$.

12.6 Optimalizace obecných regulátorů

V této kapitole budou ukázány příklady optimalizace obecných regulátorů metodou TLTE s různými typy pravidel gramatiky a kriteriálními funkcemi. Regulační pochody budou

demonstrovány na proporcionální soustavě se setrvačností druhého řádu, jejíž přenos je:

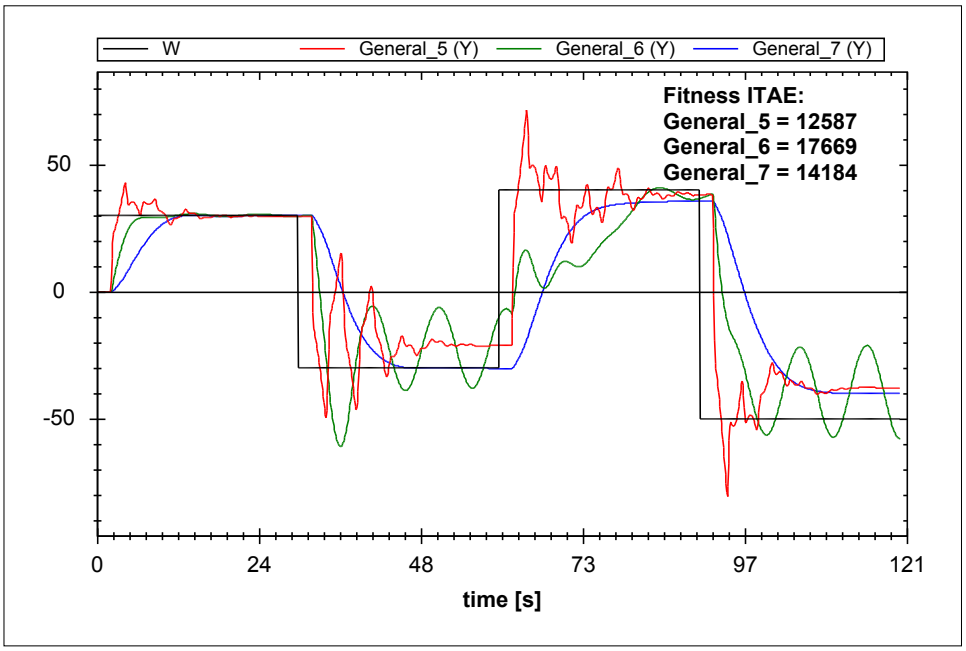
$$G_S(S) = \frac{4}{(3s+1)(5s+1)} e^{-2s} \tag{12.7}$$

Dvoufázová transplantační evoluce: Pro optimalizaci struktury a parametrů řídicí rovnice *obecných regulátorů* byla použita metoda TLTE. Parametry *TE* byly nastaveny: Počet jedinců = 50, počet nově vytvořených jedinců v každé generaci = 25, maximální počet uzlů jedince = 30, pravděpodobnost křížení = 90%, pravděpodobnost mutace rodičů = 25%, pravděpodobnost mutace potomků = 25%.

Pro optimalizaci parametrů byl použit *MDE* s parametry G = 1500, NP = 9, F = 1,07, CR = 0,58, schéma: DE/Rand/1_mod, zastavovací podmínky podle kapitoly 11.4.

12.6.1 Ukázky chybně nastavené optimalizace

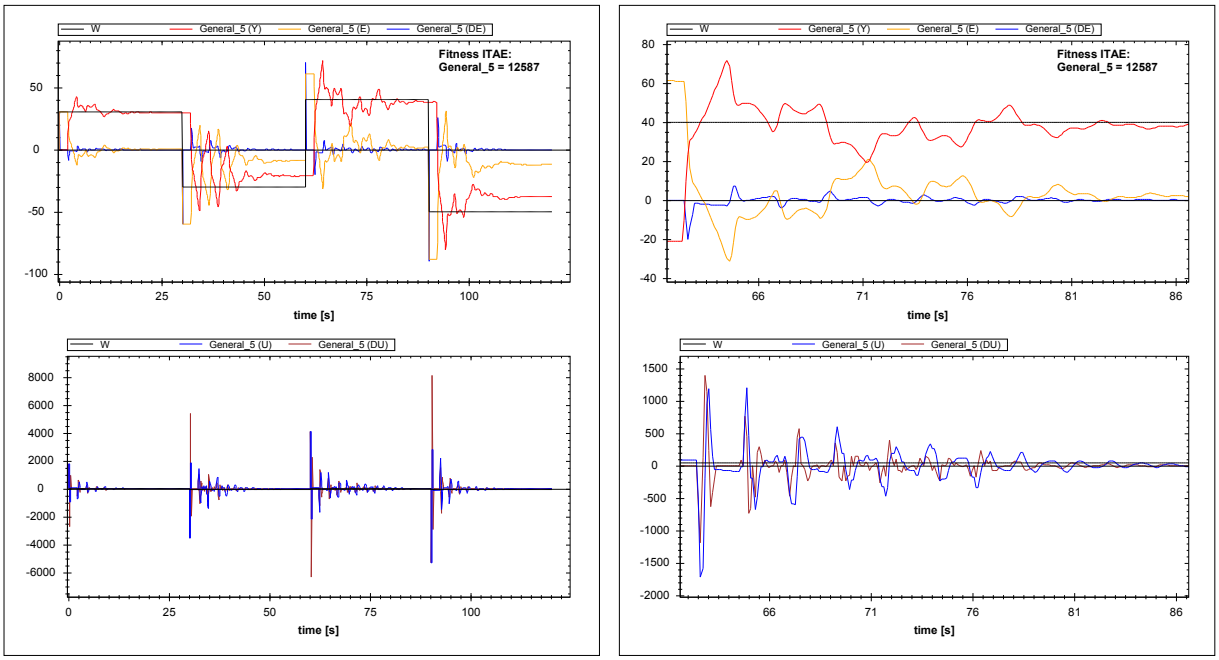
Na obrázku 69 jsou ukázky regulačních pochodů obecných regulátorů, jenž jsou výsledkem chybně nastavené optimalizace obecných regulátorů. V průběhu mnoha experimentů vyšlo najevo pár podstatných faktů o nastavení optimalizačního procesu. Nejdůležitějšími postřehy jsou:



Obrázek 69: Regulační pochody chybně optimalizovaných obecných regulátorů

1. V případě použití kritéria *IAE*, (viz. kapitola 10.10.3) je důležité současně hodnotit a penalizovat i kmitání akčního zásahu, případně regulované veličiny. Důkazem je regulační pochod regulátoru označeného *General_5*, viz. obrázek 70, který byl během optimalizace hodnocen pouze kritériem *IAE* a optimalizován pouze pro jeden interval s žádanou hodnotou $w = 30$.

2. Předcházející podmínka je podmínkou nutnou, nikoliv postačující. Tímto důkazem je regulační pochod regulátoru *General_6*. Tento regulátor byl během optimalizace hodnocen kritériem *IAE* a bylo penalizováno kmitání akčního zásahu. Optimalizace proběhla na jednom intervalu pro jednu žádanou hodnotu $w = 30$. Z obrázku 69, 71 je vidět, že nastavení regulované veličiny na žádanou hodnotu $w = 30$ proběhlo bez problémů, ale pro nastavení jiné žádané hodnoty je regulátor naprosto nevyhovující. Z tohoto vyplývá, že nestačí provést optimalizaci pro jednu hodnotu žádané veličiny.
3. Křivka *General_7*, na obrázku 69, 72 ukazuje regulační pochod regulátoru který byl během optimalizace hodnocen kritériem *IAE*, bylo penalizováno kmitání akčního zásahu, průměrná regulační odchylka na konci každého intervalu, počet bodů v toleranci a délka křivky, (viz. kapitola 11.2). Regulátor byl optimalizován pro dva intervaly, z nichž jeden byl $w = 30$ a druhý byl $w = -30$. Z obrázku je patrné, že regulátor dokáže bez problémů nastavit regulovanou veličinu na hodnotu $y = \pm 30$, ale pro jakoukoliv jinou žádanou hodnotu w je nepoužitelný. Z toho vyplývá, že nestačí optimalizovat regulátor na dvou intervalech, ve kterých je žádaná hodnota regulované veličiny $w = \pm a$, kde $w_1 = a$ je žádaná hodnota regulované veličiny v prvním intervalu a $w_2 = -a$ je žádaná hodnota regulované veličiny ve druhém intervalu.

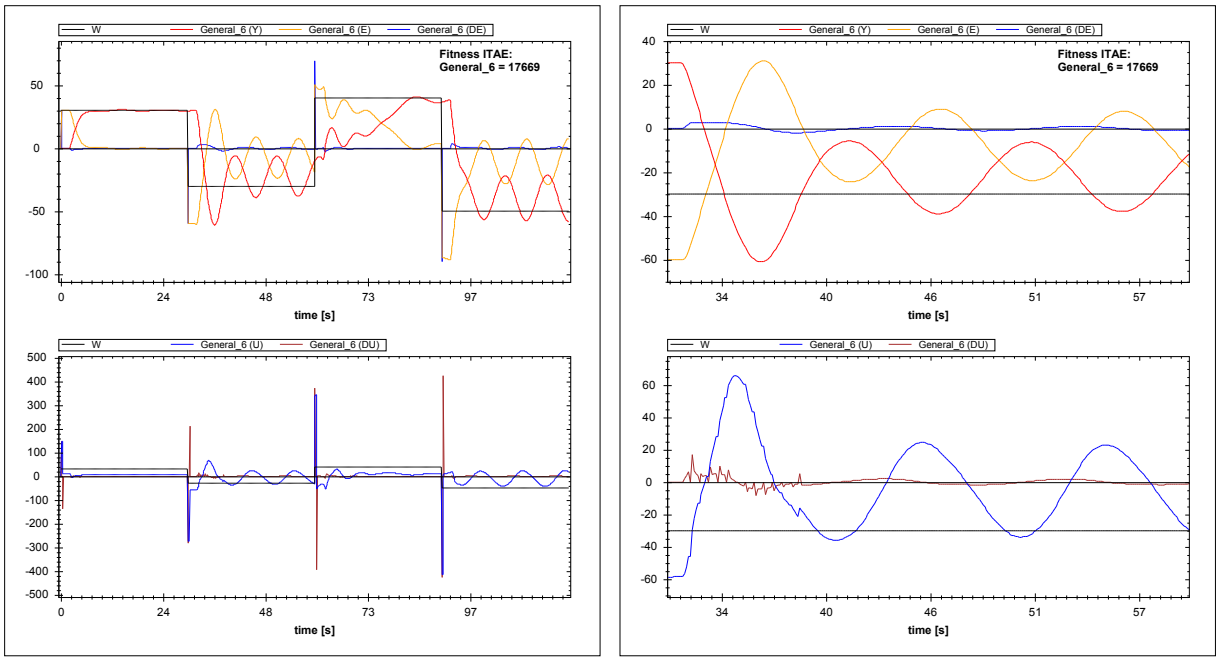


(a) Celý regulační pochod

(b) Detail regulačního pochodu

Horní graf: průběh regulačního pochodu, **Dolní graf:** průběh akčních zásahů, **červená (Y):** regulovaná veličina, **oranžová (E):** regulační odchylka, **modrá (DE):** derivace regulační odchylky, **modrá (U):** akční zásah regulátoru, **hnědá (DU):** derivace akčního zásahu, **černá (W):** žádaná hodnota regulované veličiny

Obrázek 70: Regulační pochod obecného regulátoru *General_5*



(a) Celý regulační pochod

(b) Detail regulačního pochodu

Horní graf: průběh regulačního pochodu, **Dolní graf:** průběh akčních zásahů, **červená (Y):** regulovaná veličina, **oranžová (E):** regulační odchylka, **modrá (DE):** derivace regulační odchylky, **modrá (U):** akční zásah regulátoru, **hnědá (DU):** derivace akčního zásahu, **černá (W):** žádaná hodnota regulované veličiny

Obrázek 71: Regulační pochod obecného regulátoru *General_6*

General_5: Regulační pochod *General_5* přísluší regulátoru s řídicí rovnicí. Tato rovnice je výsledkem optimalizace po 75.generaci:

```
Uk = (((32,5582238063037*Ek_5)+(58,7985461639848*Ek))+
     (7,08170165073634+(-89,9739865620223*Ek_2)))
```

Optimalizace struktury proběhla s pravidly definovanými v kapitole (viz. kapitola 11.1), ze kterých byly vyloučeny terminální symboly Yk, . . . , Yk_1 a „/“.

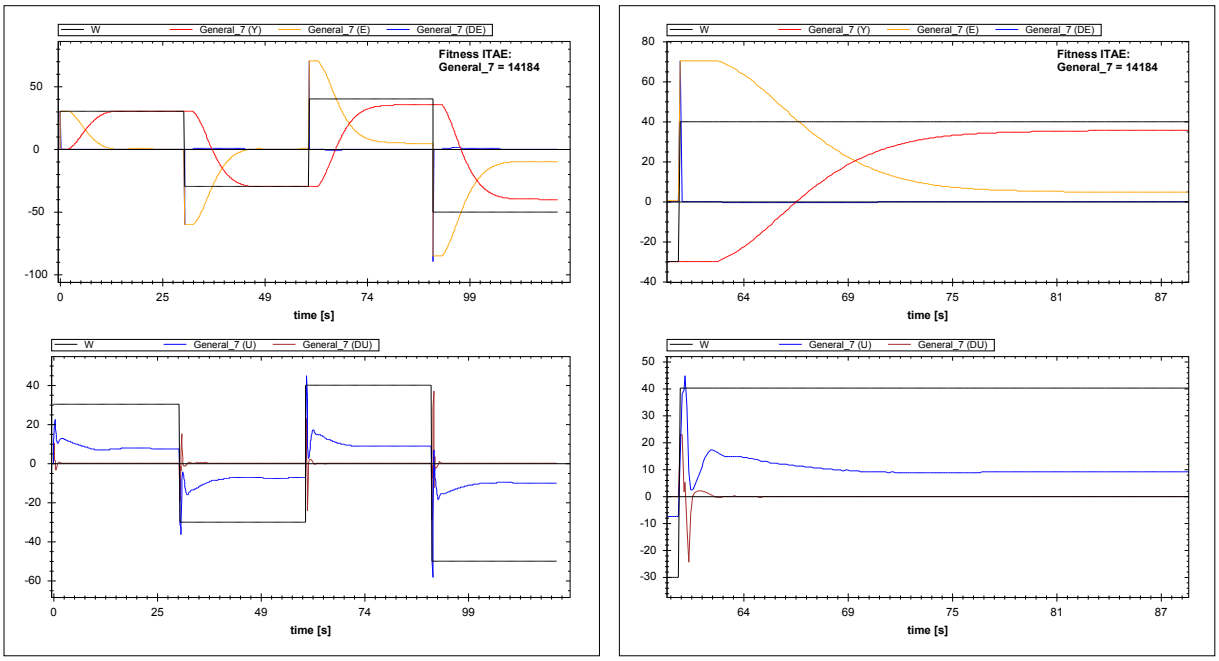
Regulační pochod, včetně zobrazeného detailu v intervalu od $\approx 62s$ do $\approx 68s$ je na obrázku 70a, resp. 70b.

General_6: Regulační pochod *General_6* přísluší regulátoru s řídicí rovnicí. Tato rovnice je výsledkem optimalizace po 121.generaci:

```
Uk = (7,62193225390625+((Ek*4,66547982026174)+(Ek_4*(-3,41207562721883+
     ((-(((45,7536105413915+Ek)+8,54175596253857))*
     (Ek*0,0305737222361965))*0,0148294215892667)))))
```

Optimalizace struktury proběhla s pravidly definovanými v kapitole 11.1 ze kterých byly vyloučeny terminální symboly Yk, . . . , Yk_1 a „/“.

Regulační pochod, včetně zobrazeného detailu v intervalu od $\approx 32s$ do $\approx 59s$ je na obrázku 71a, resp. 71b.



(a) Celý regulační pochod (b) Detail regulačního pochodu

Horní graf: průběh regulačního pochodu, **Dolní graf:** průběh akčních zásahů, **červená (Y):** regulovaná veličina, **oranžová (E):** regulační odchylka, **modrá (DE):** derivace regulační odchylky, **modrá (U):** akční zásah regulátoru, **hnědá (DU):** derivace akčního zásahu, **černá (W):** žádaná hodnota regulované veličiny

Obrázek 72: Regulační pochod obecného regulátoru *General_7*

General_7: Regulační pochod *General_7* přísluší regulátoru s řídicí rovnicí:

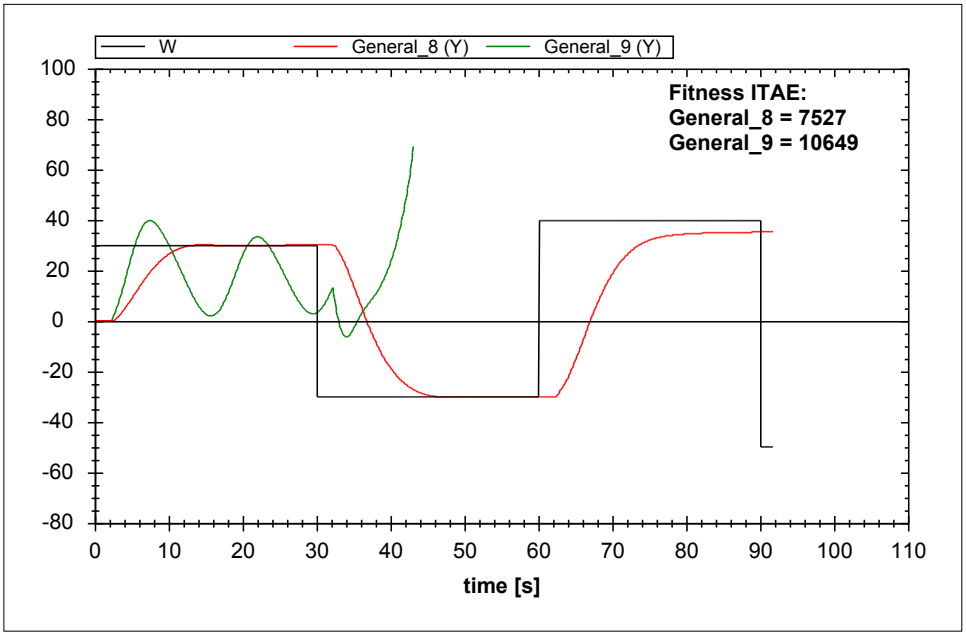
```
Uk = (((-((-(Ek_2*(-0,301663064996273)))-(Uk_1*1,03071349951939)))+
      ((Uk_1*(-7,23369179910225E-05))*((-(Ek*
      ((-((-(DEk_3/(-37,8122868930685)))-Uk_1))/(-31,2077417822251))))-
      (Uk_3*(Uk_4*(-7,4022042108816))))))-(Ek*(-0,325440780327943)))
```

Optimalizace struktury proběhla s pravidly definovanými v kapitole 11.1, ze kterých byly vyloučeny terminální symboly Yk, . . . , Yk_1.

Regulační pochod, včetně zobrazeného detailu v intervalu od $\approx 61s$ do $\approx 88s$ je na obrázku 72a, resp. 72b.

12.6.2 Ukázky chyb v průběhu simulace

Na obrázku 73 jsou ukázány dva příklady chyb v průběhu simulace regulace. V obou zobrazených případech pokračuje regulovaná, příp. akční veličina do nekonečna. V případě *General_8* dosáhne $-\infty$ v čase 57,5s poté co se regulátor snaží vyrovnat prudce narůstající chybu regulované veličiny. Regulovaná veličina regulátoru *General_9* dosáhne hodnoty ∞ v 90,7s díky náhlému zákmitu akční veličiny. Z tohoto obrázku mj. vyplývá důležitost použití několikafázového ohodnocení regulačního pochodu, které se uplatňuje právě v případě takovýchto chyb, (viz. kapitola 11.2.1).



červená: General_8, **zelená:** General_9

Obrázek 73: Ukázky chyb ($\pm\infty$) v průběhu simulace regulace

General_8: Tento regulační pochod je výsledkem řídicí rovnice regulátoru:

```
Uk = ((((-(DEk_3*(-0,554605924578443)))-(Uk_1*(-1,03526653482844)))+
     ((Uk_1*(Uk_1*0,000491656195958113))*((-(Uk_4))-
     (Uk_3*0,25300123214001))))-(Ek*(-0,0236843739347194)))
```

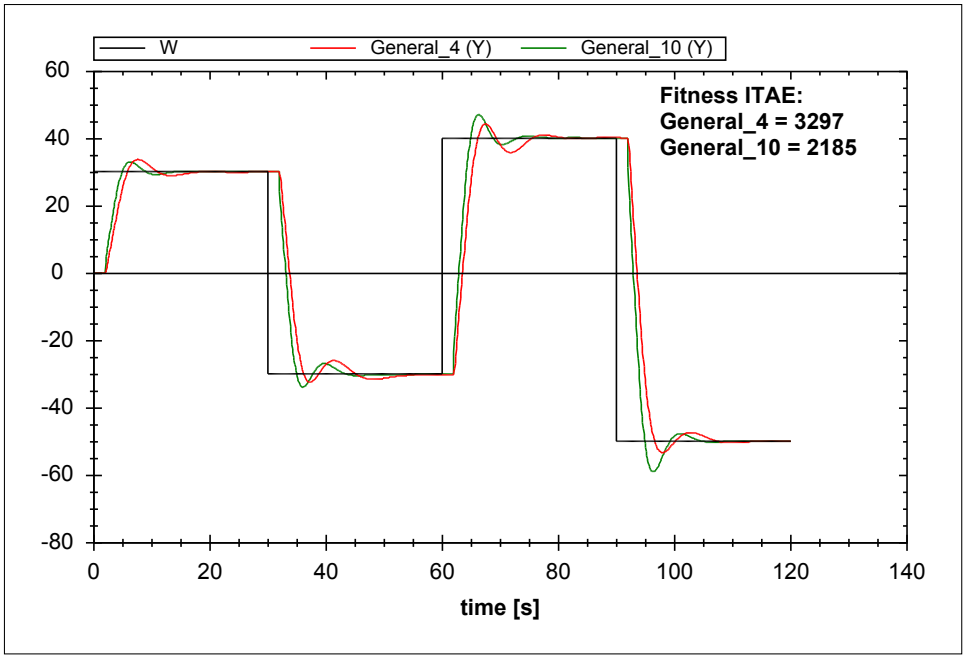
General_9: Regulační pochod je výsledkem řídicí rovnice regulátoru:

```
Uk = (-((((((-47,0642949272105)+(DEk_1*DEk_4))*(-3,50176842352982))+
     ((-(Ek_2*(Ek/1,41946999419199)))-(Ek_2*6,04369731616662)))*
     0,0455554709203227)+((Ek_2*(((Uk_1+(DEk_2*(-76,3179821444081)))-
     (Ek+79,2532577811589))*(-0,000180769843021228)))-((Ek_2*(DEk_2*
     (-0,00188164990393764)))*(DEk_2*(-4,62774508630033))))))
```

Tento regulátor je opět jedním z příkladů, který dokazuje, že nestačí optimalizovat regulátor na dvou intervalech, ve kterých je žádaná hodnota regulované veličiny $w = \pm a$, kde $w_1 = a$ je žádaná hodnota regulované veličiny v prvním intervalu a $w_2 = -a$ je žádaná hodnota regulované veličiny ve druhém intervalu. Z obrázku 73 je patrné, že pro první dva intervaly, ve kterých byla žádaná hodnota $w = 30$, resp. $w = -30$ proběhla regulace bez jakýchkoliv problémů.

12.6.3 Porovnání kriteriálních funkcí

V této kapitole jsou srovnány různé možnosti hodnocení regulačních pochodů. V případě, že se použijí kritéria hodnocení dle kapitoly 11.1, je možno dosáhnout větší flexibility nastavení regulátoru. Na druhou stranu se jedná o mnohdy náročnou implementaci mnoha



červená (General_4): Kritéria hodnocení 11.2, **zelená (General_10):** Kritérium *ITAE*

Obrázek 74: Porovnání kriteriálních funkcí

kritérií a neméně důležité sestavení samotné kriteriální funkce. Z tohoto důvodu je výhodnější použít kritérium *ITAE*, příp. *ITSE*, (viz. kapitola 10.10.3) a k němu v případě potřeby přidat jiné doplňující kritéria.

Na obrázku 74 jsou vidět regulační pochody regulátoru *General_4* a *General_10*.

General_4: je regulační pochod obecného regulátoru s řídíci rovnicí:

```
Uk = (Uk_1-((2,66414343914243*Ek_4)+(((DEk*(-3,28788061116549))+
     (((((DEk*76,1825131087609)+1,26443835876666)-
     ((-1,27123447444844)*Ek_1))*((9,32622370528897E-07)*Ek_5))*Ek_1))-
     (2,6711609052335*Ek_5))))
```

Tato rovnice je výsledkem optimalizačního procesu s kriteriální funkcí hodnotící podle (viz. kapitola 11.2).

General_10: je regulační pochod obecného regulátoru s řídíci rovnicí:

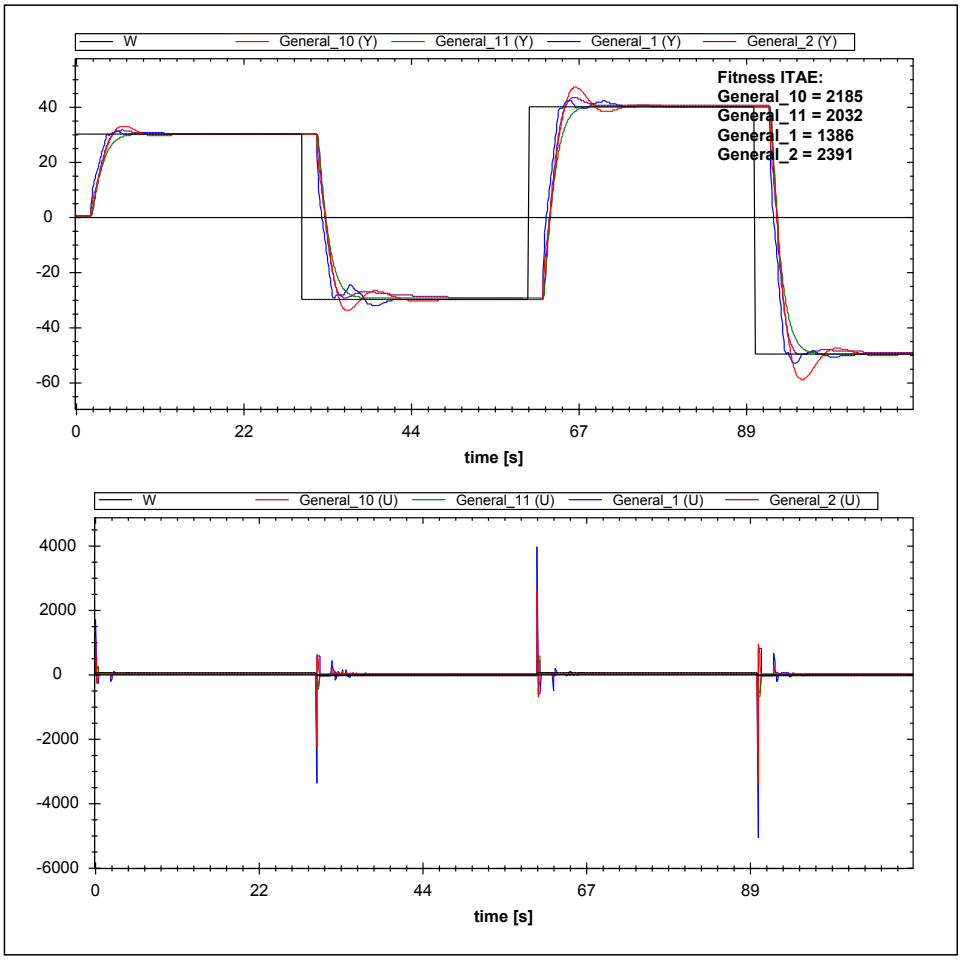
```
Uk = (Uk_1+((37,7949275202727*Ek)+((((-85,7168395987696)*Ek_1)+
     (51,5425586964905*Ek_2))+(Ek_5*(-3,61245065902024)))))
```

Tato rovnice je výsledkem optimalizačního procesu s kriteriální funkcí hodnotící podle *ITAE*.

Z obrázku je vidět, že pomocí kritérií z kapitoly 11.2 lze dosáhnout obdobného výsledku jako v případě použití kritéria *ITAE*.

12.6.4 Porovnání řídicích rovnic regulátorů vytvořených různými sadami pravidel a různými kriteriálními funkcemi

V této kapitole budou srovnány obecné regulátory, které byly optimalizovány se dvěma různými sadami pravidel, definovanými v kapitole 11.1. V prvním případě to bude regulátor *General_10*, který byl již představen v kapitole 12.6.3. Řídicí rovnice tohoto regulátoru byla optimalizována bez použití terminálních symbolů Yk, . . . , Yk_5 a „/“. Další regulátory *General_1*, *General_2*, *General_11* byly optimalizovány s použitím terminálních symbolů Yk, . . . , Yk_5. Každý z těchto regulátorů byl optimalizován s použitím jiné kriteriální funkce, zaměřené na různé optimalizační cíle.

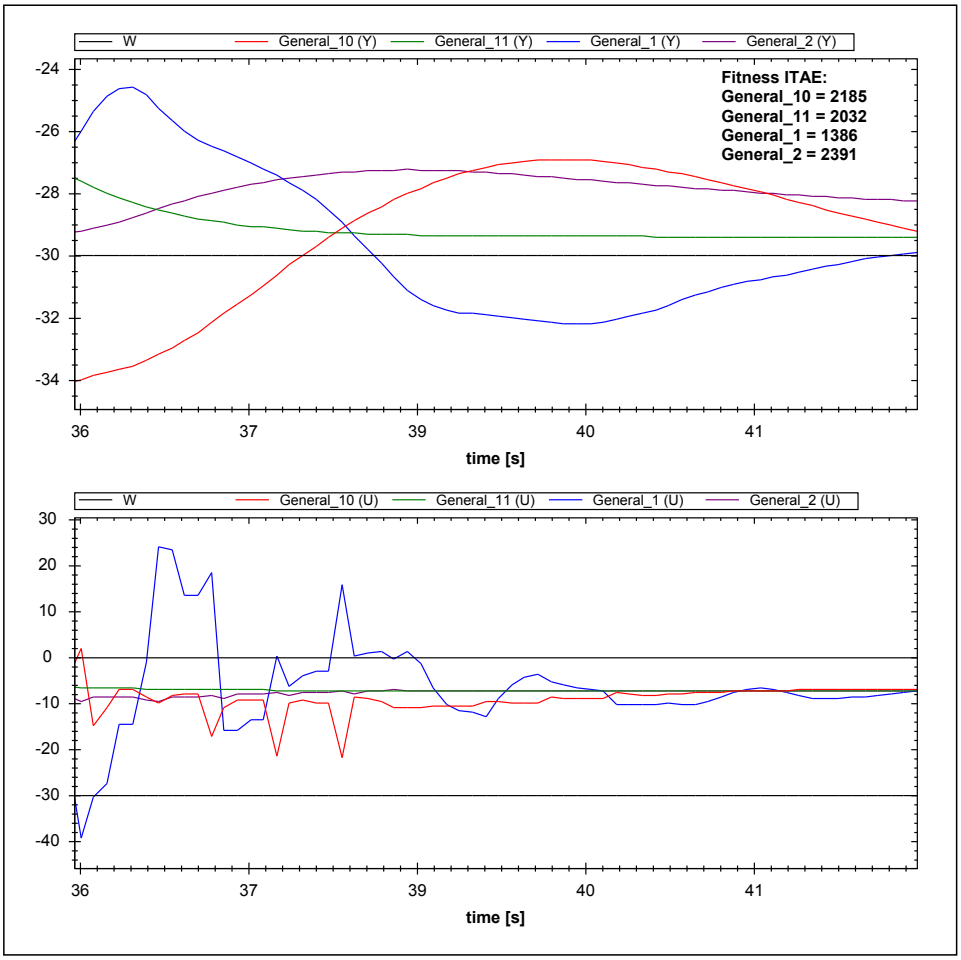


Horní graf: průběh regulačního pochodu, **Dolní graf:** průběh akčních zásahů

Obrázek 75: Porovnání řídicích rovnic regulátorů (různé sady pravidel, různé kriteriální funkce)

General_1: Řídící rovnice tohoto regulátoru byla optimalizována za použití kriteriální funkce, jenž regulační pochod hodnotila dle kritérií uvedených v kapitole 11.2. Cílem bylo minimalizovat regulační plochu. Výsledná řídící rovnice regulátoru je:

```
Uk = ((((-0,749999348556586)*Yk_1)-((Ek_5*(-9,75561281861947))-
     ((Ek_1*(-65,6717833232466))+Yk_1)))+
     (Ek*((Ek_1*(-3,47554843341052E-05))+56,8098513777129)))
```



Horní graf: průběh regulačního pochodu, **Dolní graf:** průběh akčních zásahů

Obrázek 76: Detailní porovnání řídicích rovnic regulátorů (různé sady pravidel, různé kriteriální funkce)

General_2: Řídící rovnice tohoto regulátoru byla optimalizována za použití kriteriální funkce, jenž regulační pochod hodnotila také dle kritérií uvedených v kapitole 11.2 s rozdílem, že byla snaha o snížení počtu zákmitů akční veličiny se zachováním malé regulační plochy. Výsledná řídící rovnice regulátoru je:

```
Uk = (((-((Ek_4*6,10907856808768)+(((Ek*48,1979429365881)+
     (Yk_5*6,11933271280889))-(-((Ek*(-20,0222697934244))+
     (((-(Ek_4*17,1675893977531))+(Yk_5*(Ek_4*(-0,706908323728534))))*
     (-(Yk*(3,58131795119276E-05))))))))-(Ek*(-34,5355973303833)))-
     (Yk_2*(-6,3696989730917)))
```

General_11: Řídící rovnice tohoto regulátoru byla optimalizována za použití kriteriální funkce, která hodnotila regulační pochod dle kritéria *ITAE* a navíc byla přidána penalizace kmitání akčního zásahu. To znamená, že cíl byl obdobný jako v případě *General_2*. Rovnice ve 13.generaci TLTE měla tvar:

```
Uk = (((Yk_1*8,23510211558202)-(Ek_4*8,20735775293061))-
     (((-8,25063155276869)*Ek_1)-(Yk_4*(-7,98497699739693))))
```

Výsledné regulační pochody jsou ukázány na obrázcích 75, 76. Na obrázku 76 je ukázán detail regulačního pochodu v intervalu od $\approx 36s$ do $\approx 42s$. Z detailního zobrazení je možné vyčíst, že regulátor *General_1* má sice nejmenší regulační plochu, ale vykazuje velké známky kmitání akčního zásahu. Na druhou stranu regulátor *General_11* má naprosto klidný průběh, bez známek kmitání.

12.7 Porovnání obecných regulátorů se setrvačností prvního řádu a PSD regulátorů – kmitavá proporcionální soustava bez dopravního zpoždění

Pro proporcionální soustavu s kmitavým průběhem, bez dopravního zpoždění, danou přenosem:

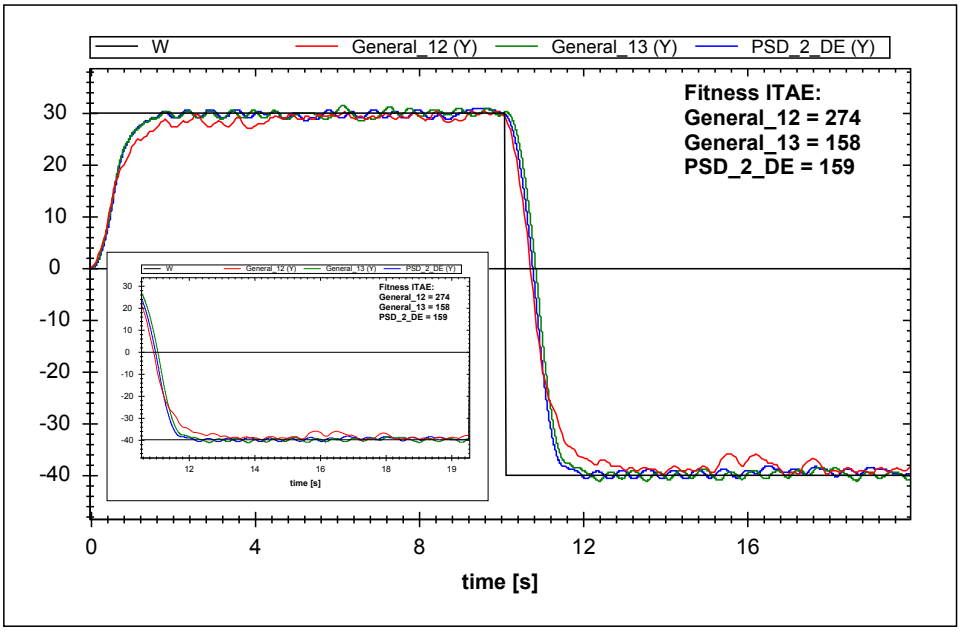
$$G_S(S) = \frac{k_S}{T_0^2 s^2 + 2\xi T_0 s + 1} \tag{12.8}$$

s parametry $k_S = 0.5$, $T_0 = 1$, $\xi = 0{,}1$ bylo provedeno porovnání regulačních pochodů *PSD regulátoru* a *obecného regulátoru*. Parametry *PSD regulátoru* byly optimalizovány DE (viz. kapitola 4), struktura a parametry obecných regulátorů byly optimalizovány TLTE (viz. kapitola 8). Vzorkovací frekvence byla v případě regulátoru *General_12* zvolena $T = 0{,}1s$. Regulátory *General_13* a *PSD_2_DE* měly vzorkovací periodu nastavenou na $T = 0{,}01s$. Limit akčního zásahu byl u všech regulátorů nastaven v rozsahu ± 200. Pásmo necitlivosti regulátoru nastaveno na 1% žádané hodnoty, při zpětné diferenci regulované veličiny $\nabla y(kT) = 0{,}1$. Během regulace byl také použit filtr druhého řádu.

Pro optimalizaci struktury a parametrů obecných regulátorů byla využita sada pravidel definovaná v kapitole 11.1. Z této sady byly v případě optimalizace regulátoru *General_12* odebrány pravidla obsahující terminální symboly „/“, $Y_{k-3}, \ldots, Y_{k-5}$, $E_{k-3}, \ldots, E_{k-5}$, $dE_k, \ldots, dE_{k-4}$ a pravidla obsahující $U_{k-1}, \ldots, U_{k-5}$. V případě optimalizace regulátoru *General_13* byly odebrány pravidla „/“, $dE_k, \ldots, dE_{k-4}$ a pravidla obsahující $U_{k-1}, \ldots, U_{k-5}$.

Diferenciální evoluce: Pro optimalizaci parametrů *PSD regulátoru* (k_R, T_I, T_D) byla použita DE s nastavením G = 1500, NP = 20, F = 1,07, CR = 0,58, schéma: DE/Rand/1, zastavovací podmínky podle kapitoly 11.4. Výsledkem optimalizace je nastavení parametrů *PSD regulátoru* $k_R = 3858{,}862378$, $T_I = 0{,}510784$, $T_D = 0{,}122012 \cdot 10^{-2}$.

Dvoufázová transplantační evoluce: Pro optimalizaci struktury a parametrů řídicí rovnice *obecného regulátoru* byla použita metoda TLTE. Parametry *TE* byly nastaveny: Počet jedinců = 30, počet nově vytvořených jedinců v každé generaci = 15, maximální počet uzlů jedince = 30, pravděpodobnost křížení = 90%, pravděpodobnost mutace rodičů



General_12: Obecný regulátor (struktura a parametry optimalizovány TLTE pro $T = 0{,}1s$, kritéria hodnocení: ITAE + maximální překmit „y“, 7.generace TLTE, setrvačnost 1.řádu, pásmo necitlivosti 1% ($\nabla y(kT) = 0{,}1$), filtr 2.řádu, omezená pravidla: Yk,..., Yk_2; Ek,..., Ek_2), **General_13:** Obecný regulátor (struktura a parametry optimalizovány TLTE pro $T = 0{,}01s$, kritéria hodnocení: ITAE + maximální překmit „y“, 22.generace TLTE, setrvačnost 1.řádu, pásmo necitlivosti 1% ($\nabla y(kT) = 0{,}1$), filtr 2.řádu), **PSD_2_DE:** PSD regulátor (parametry optimalizovány DE pro $T = 0{,}01s$, kritéria hodnocení: ITAE + maximální překmit „y“, 1500.generací DE, pásmo necitlivosti 1% ($\nabla y(kT) = 0{,}1$), filtr 2.řádu)

Obrázek 77: Porovnání obecných regulátorů se setrvačností prvního řádu a PSD regulátorů, s pásmem necitlivosti – kmitavá proporcionální soustava bez dopravního zpoždění

= 25%, pravděpodobnost mutace potomků = 25%, metoda ART, (viz. kapitola 8.8), byla aktivní. Parametry *MDE* byly nastaveny: G = 1500, NP = 20, F = 1,07, CR = 0,58, schéma: DE/Rand/1 (pro General_12), DE/Rand/1_mod (pro General_13), zastavovací podmínky podle kapitoly 11.4. Výsledkem optimalizace v 7.generaci TLTE je řídicí rovnice *obecného regulátoru General_12* ve tvaru:

```
Uk = Uk_1+((-(Ek+(Yk_1-14,4044473368695)))+(-((((Ek_2*
     (-1,69526341729436))-(Ek*41,3497823420514))-(Ek_1*
     (-38,0841704174065)))*(Ek-(-71,1561559627034)))))
```

Rovnice byla ponechána v původním tvaru v jakém jej vrátila TLTE. V průběhu optimalizace již byla do rovnice uměle přidána setrvačnost prvního řádu ve formě členu Uk_1.

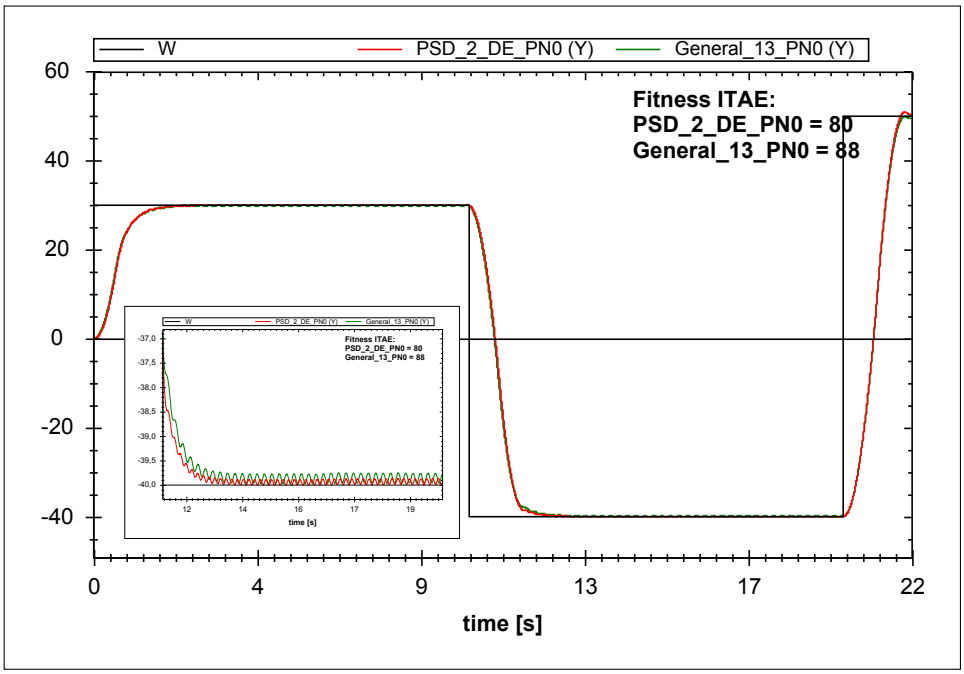
Výsledkem optimalizace ve 22.generaci TLTE je řídicí rovnice *obecného regulátoru General_13* ve tvaru:

```
Uk = Uk_1+(-(((-((Ek_1*46,1238341489317)-((Ek*(-6,86712128949357))*
     (-6,84720209198847))))*(-50,3217781662577))-0,723919624277033))
```

Rovnice byla ponechána v původním tvaru v jakém jej vrátila TLTE. V průběhu optimalizace již byla do rovnice uměle přidána setrvačnost prvního řádu ve formě členu Uk_1, stejně jako v případě optimalizace *General_12*.

Kriteriální funkce ve všech případech hodnotila dle kritéria *ITAE*, ke kterému byla přidána penalizace překmitu regulované veličiny. Do řídicích rovnic obecných regulátorů byl uměle přidán setrvačný člen 1.řádu, pásmo necitlivosti bylo ve všech případech nastaveno na 1% při zpětné diferenci $\nabla y(kT) = 0{,}1$, všechny regulátory používaly filtr regulační odchylky 2.řádu.

Výsledné regulační pochody je možno porovnat na obrázku 77. Na tomto obrázku jsou ukázány tři regulační pochody. V pravém horním rohu obrázku je zobrazeno hodnocení ITAE, v levém dolním rohu je ukázán detail regulačních pochodů v intervalu od $\approx 11s$ do $\approx 19s$.

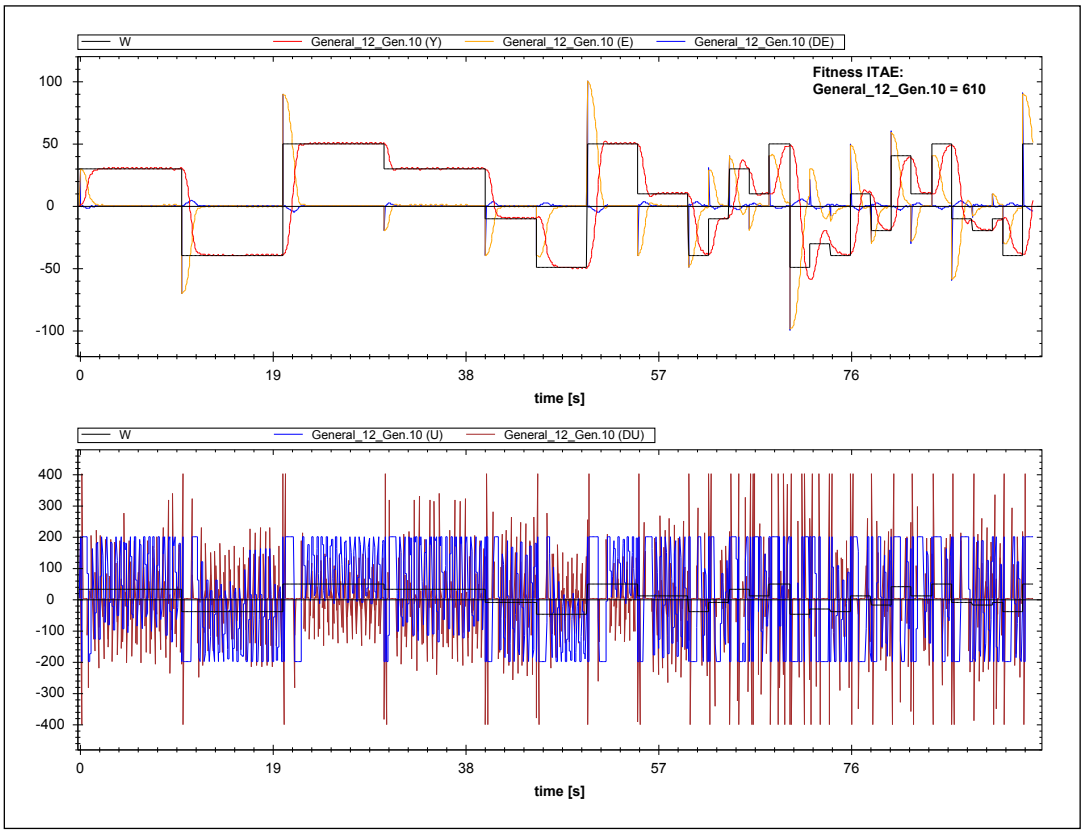


General_13_PN0: Obecný regulátor (struktura a parametry optimalizovány TLTE pro $T = 0{,}01s$, kritéria hodnocení: ITAE + maximální překmit „y“, 22.generace TLTE, setrvačnost 1.řádu, filtr 2.řádu), **PSD_2_DE_PN0:** PSD regulátor (parametry optimalizovány DE pro $T = 0{,}01s$, kritéria hodnocení: ITAE + maximální překmit „y“, 1500.generací DE, filtr 2.řádu)

Obrázek 78: Porovnání obecných regulátorů se setrvačností prvního řádu a PSD regulátorů, bez pásma necitlivosti – kmitavá proporcionální soustava bez dopravního zpoždění

Na obrázku 78 jsou ukázány regulační pochody regulátorů *General_13* a *PSD_2_DE* s vypnutými pásmy necitlivosti. Tyto regulátory jsou označeny *General_13_PN0* a *PSD_2_DE_PN0*. V pravém horním rohu obrázku je zobrazeno hodnocení ITAE, v levém dolním rohu je ukázán detail regulačních pochodů v intervalu od $\approx 11s$ do $\approx 19s$.

Na obrázku 79 je výsledek optimalizace řídicí rovnice obecného regulátoru *General_12* v 10.generaci TLTE. Průběh regulačního pochodu je ukázán v intervalu $\langle 0, 90\rangle$ sekund, přičemž interval změny žádané hodnoty se na počátku mění periodicky po $5s$ a postupně



Horní graf: průběh regulačního pochodu, **Dolní graf:** průběh akčních zásahů, **červená (Y):** regulovaná veličina, **oranžová (E):** regulační odchylka, **modrá (DE):** derivace regulační odchylky, **modrá (U):** akční zásah regulátoru, **hnědá (DU):** derivace akčního zásahu, **černá (W):** žádaná hodnota regulované veličiny

Obrázek 79: Průběh regulačního pochodu regulátoru *General_12* v 10.generaci TLTE

se snižuje až na 1s. Změny hodnoty žádané veličiny jsou náhodné.

Na tomto obrázku je vidět kmitání akčního zásahu, jenž je shora a zdola omezeno daným rozsahem regulátoru, který je nastaven v intervalu ±200. Toto kmitání je způsobeno tím, že pro hodnocení kvality regulace během optimalizace nebylo použito kritéria, které penalizuje kmitání akčního zásahu regulátoru.

Rovnice tohoto regulátoru má tvar:

```
Uk = Uk_1 + (((((Ek_2*(-0,663520535763126))-((((Ek_2*(-19,4200644798541))-
     (Ek*62,8320479267121))-(Ek_1*(-74,5999125652272)))*
     9,69180579919801))-(Ek_1*25,2196416841911))+
     (Ek*7,51412533075408))-(-6,26645661250988))
```

Zobrazená rovnice byla ponechána v původním tvaru v jakém jej vrátila TLTE po 10.generaci.

Z časového hlediska byla optimalizace přerušena po 11.generacích TLTE (10,5 hodin CPU času 3,2GHz CPU). Za tuto dobu bylo vyčísleno 351,7 tis. generací DE s průměrných počtem 1056,3 generací DE na jednoho jedince TE.

13 Paralelní a distribuované výpočty

Během experimentů bylo zjištěno, že nejslabším článkem optimalizace je diferenciální evoluce, která slouží k optimalizaci abstraktních parametrů řídicí rovnice obecných regulátorů. Optimalizace parametrů vyžaduje velké množství ohodnocení cílové funkce a z toho vyplývající požadavek na velký počet generací diferenciální evoluce. Běžně byl maximální počet generací diferenciální evoluce v průběhu optimalizace z časového hlediska omezen na 1500 generací s ukončovacími podmínkami, jenž byly definovány v kapitole 11.4 s cílem zkrátit dobu optimalizace. Statisticky bylo zjištěno, že průměrný počet potřebných generací DE pro optimalizaci parametrů je na počátku TLTE nižší a postupně roste se zvyšujícím se počtem generací TLTE a s rostoucím počtem abstraktních numerických parametrů ve struktuře rovnice obecného regulátoru. Průběh evoluce časově náročné optimalizace parametrů byl ukázán na obrázku 57, který byl diskutován v kapitole 11.2.1.

V případech většího počtu parametrů občas dochází k uvíznutí v lokálním extrému, což je částečně způsobeno nízkým počtem jedinců DE, který byl během inicializace TLTE z důvodu minimalizace času optimalizace nastavován v intervalu 9 až 15 jedinců DE.

Doba optimalizace také závisí na délce intervalu simulace regulačního pochodu, jehož délku je potřeba zvolit v závislosti na rychlosti přechodu soustavy z jednoho stavu do druhého. Délka simulačního intervalu se zvyšuje úměrně s součtem časových konstant soustavy.

V neposlední řadě prodlužuje dobu optimalizace zvyšující se počet prvků řídicí rovnice regulátoru. Z tohoto důvodu je velice přínosnou metoda ART, která redukuje redundantní části a tím zkracuje délku rovnice.

Orientačně: 100 generací TLTE s 50.jedinci, aktivní metodou ART, maximálním počtem generací DE omezeným na 1500, s ukončujícími podmínkami z kapitoly 11.4 a počtem 9 jedinců DE, vyžaduje přibližně 15 – 50 hodin procesorového času 3,2GHz CPU (aplikace v C#.NET).

Časovou náročnost optimalizace je možno zkrátit snížením počtu jedinců TLTE, ale za cenu snížení variability populace.

Vzhledem k výše zmíněné náročnosti této optimalizace by bylo přínosem paralelizování výpočtu na dvou úrovních [25]:

1. *globální paralelní model*, jenž by rozdělil proces optimalizace na více samostatných výpočetních jednotek s cílem snížit výpočetní čas.
2. *paralelizace na úrovni algoritmu*, tzv. *grained EA*, který by prohledal větší prostor řešení s cílem najít globální optimum.

O efektivitě globální paralelizace pojednává tzv. **Amdahlův zákon** *(Amdahl's law)*, jenž definuje vztahy pro zrychlení paralelního procesu, účinnost paralelizace, atd. [23].

Pro vyjádření těchto vztahů je nutno zavést následující značení: T_c je celkový čas potřebný pro zpracování úlohy na jednom procesoru,
T_p čas potřebný pro zpracování části, kterou lze rozdělit na nezávislé podúlohy,
T_s čas potřebný pro zpracování části, kterou je nutné zpracovat sekvenčně.

Z definice plyne, že $T_c = T_p + T_s$. V případě, že paralelní část lze rozdělit na k nezávislých a stejně časově náročných úloh a je k dispozici alespoň k procesorů, mohou být

všechny podúlohy zpracovány současně a celý proces optimalizace bude dokončen v čase T(k), který lze vyjádřit vztahem:

$$T(k) = T_s + \frac{T_p}{k} \tag{13.1}$$

Zrychlení se dá poté vyjádřit jako:

$$S(k) = \frac{T_c}{T(k)} \tag{13.2}$$

Účinnost paralelizace lze vyjádřit vztahem:

$$\epsilon(k) = \frac{S(k)}{k} \tag{13.3}$$

V případě že by pro paralelní část bylo použito $k = NP_{nTLTE}$ procesorů, kde NP_{nTLTE} je počet jedinců vytvářených v každé generaci TLTE, byl by čas výpočtu jedné generace TLTE roven času potřebnému pro optimalizaci parametrů nejpomalejšího výpočtu DE. Tedy v případě, že se v každé generaci TLTE vytváří 25 jedinců a neparalelní proces trval 35 hodin a pokud zanedbáme případ, kdy do nové generace vstupuje kopie rodiče (závisí na pravděpodobnosti křížení – již se neprovádí optimalizace parametrů), trval by paralelní výpočet $\approx$ 1,5 hod.

V případě, že by se paralelizovala i druhá vrstva, čili vrstva DE a bylo by k dispozici nejméně $NP_{nTLTE} \cdot NP_{DE}$ procesorů, byl by výpočet dokončen za $\approx$ 10 min (za předpokladu $NP_{nTLTE} = 25$, $NP_{DE} = 9$), kde NP_{DE} je počet jedinců DE.

14 Závěr

Mým záměrem při tvorbě této práce bylo proniknout do podstaty řídicích algoritmů regulátorů a pokusit se optimalizovat řídicí algoritmy obecných regulátorů pro danou soustavu a žádaný průběh regulačního pochodu. Dílčím úkolem bylo vytvoření části aplikace, jenž bude pro zadanou soustavu a řídicí rovnici regulátoru simulovat regulační pochod, který poté dle zadaných kritérií ohodnotí. Dalším úkolem bylo vytvoření evolučního algoritmu, který je schopen optimalizovat parametry PSD regulátoru. Pro splnění tohoto úkolu byla implementována dobře známá metoda *diferenciální evoluce*, jejímž popisem se zabývají kapitoly 4 a 11.3.

Hlavní náplní této práce a současně jedním z největších přínosů je optimalizace struktur řídicích rovnic obecných regulátorů pro danou soustavu a požadavky na průběh regulačního pochodu. Pro splnění tohoto cíle byl vytvořen originální evoluční algoritmus nazvaný *transplantační evoluce* a její rozšíření nazvané *dvoufázová transplantační evoluce*. Tyto typy evolučních algoritmů jsou popsány v kapitolách 7 a 8.

Klíčovou vlastností a zároveň největší výhodou transplantační evoluce je schopnost vytvořit model řešené úlohy, který je srozumitelný uživateli a je použitelný i mimo evoluční metodu. V tomto ohledu tedy svými schopnostmi transplantační evoluce předčí neuronové sítě, protože model vytvořený neuronovou sítí není použitelný mimo ni. Další výhodou tentokráte v porovnání s gramatickou evolucí je možnost vložit řešení do transplantační evoluce a toto řešení dále optimalizovat bez nutnosti složitě vytvářet genotyp jedince. Vzhledem k tomu, že jedinec tranpslantační evoluce je reprezentován jen fenotypem, bylo možno vytvořit pokročilé metody křížení a mutace, které jsou plně řízeny pravidly gramatiky, narozdíl od gramatické evoluce, v níž křížení a mutace probíhá na úrovní genotypu a gramatika se používá až v další fázi, během translace genotypu na fenotyp. Další výhodou transplantační evoluce v porovnání s gramatickou evolucí je možnost změny pravidel v průběhu evoluce, bez totální destrukce již vygenerovaných řešení. Vzhledem k tomu, že požadavkem optimalizace na výsledný model je získání strukturálně minimálního (optimálního) řešení, je jedním z největších přínosů transplantační evoluce možnost použití *přímé aritmetické redukce modelu*, která zajistí redukci jeho redundantních částí, což má mj. přímý vliv na zkrácení doby optimalizace. Přínosem dvoufázové transplantační evoluce, která byla inspirována nedávno vytvořeným algoritmem dvoufázové gramatické evoluce je přímá a efektivní optimalizace abstraktních numerických parametrů obsažených ve struktuře řídicí rovnice regulátoru, jejíž správné nastavení je rozhodující pro zajištění kvalitního průběhu regulačního pochodu a pro nalezení strukturálně optimálního řešení. Vzhledem k citlivosti řídicí rovnice regulátoru na nastavení parametrů, je tato činnost v gramatické, nebo transplantační evoluci naprosto neřešitelná.

Pro zajištění optimálního chodu druhé fáze optimalizace byl vytvořen meta–evoluční algoritmus, jehož cílem byla optimalizace parametrů diferenciální evoluce, která je použita pro optimalizaci parametrů řídicích rovnic regulátorů jako součást dvoufázové transplantační evoluce, nebo samostatně pro optimalizaci parametrů PSD regulátoru. Popisu *meta–evoluce* se věnuje kapitola 9.

V průběhu experimentů bylo zjištěno, že standardní schéma diferenciální evoluce DE/Rand/1 (DE/Best/1) v určitých situacích nekonverguje k očekávaným hodnotám, proto bylo vytvořeno opět původní schéma, jenž bylo nazváno *DE/Rand/1mod*, případně *DE/Rand/1plus* (*DE/Best/1mod*, *DE/Best/1plus*). Přínos tohoto schématu je sice dis-

kutabilní, jelikož po vytvoření meta–evolučního algoritmu a optimalizování parametrů diferenciální evoluce vykazuje podobné výsledky jako původní schéma, nicméně i přesto bylo toto schéma v průběhu optimalizací používáno. Uvedené schéma je popsáno v kapitole 11.3.

Pro ohodnocení kvality regulačních pochodů byly vytvořeny opět originální metody, které byly kombinovány s běžnými integrálními kritérii velikosti regulační plochy. Tyto kritéria jsou schopny plně nahradit kritérium *ITAE*, případně jsou použitelné jako doplňková kritéria ohodnocení kvality regulačního pochodu ke kritériím *ITAE*, apod. pro zajištění žádaného průběhu regulované veličiny.

Výsledky optimalizací byly simulačně porovnány s *PSD regulátory*, nastavenými pomocí klasických výpočetních i evolučních metod. Ve všech případech optimalizace se podařilo vygenerovat kvalitnější řídicí rovnici, než rovnice *PSD regulátorů*, případně její známá modifikace v *Takahashiho* tvaru. Nově vygenerované řídicí algoritmy ve všech případech vykazovaly kvalitnější průběh regulačního pochodu současně se sníženou dobou regulace.

Výsledkem experimentů je jednoznačný závěr, že optimalizací řídicích rovnic obecných regulátorů je možno získat kvalitnější průběh regulačního pochodu, než jaký nabízejí klasické *PSD regulátory* a jejich známé modifikace. Časová náročnost pro nalezení rovnice *obecného regulátoru* je ovšem mnohem větší, než nalezení parametrů *PSD regulátoru*.

V závěru práce je uvedeno doporučení pro další vývoj v tomto oboru, jenž se týká paralelních a distribuovaných výpočtů a týká se hlavně zkrácení doby optimalizace.

Hlavní přínos práce je shrnut do následujících bodů:

- Byly vytvořeny původní metody *transplantační evoluce* a její modifikace *dvoufázová transplantační evoluce*, které jsou obecně využitelné pro různé optimalizační problémy, (viz. kapitoly 7 a 8),
- Byla vytvořena původní metoda přímé aritmetické redukce modelu řešení (ART), (viz. kapitola 8.8),
- Byla vytvořena nová metoda křížení spojováním částí jedince, (viz. kapitola 7.2.2),
- Byly vytvořeny gramatikou řízené metody křížení a mutace, (viz. kapitoly 7.2, 7.3),
- Byly vytvořeny původní kritéria, hodnotící kvalitu regulace, (viz. kapitola 11.2),
- Byly vytvořeny unikátní kritéria hodnocení kvality regulačního pochodu v případě chyby v průběhu regulace, (viz. kapitola 11.2.1),
- Bylo vytvořeno nové schéma diferenciální evoluce, (viz. kapitola 11.3),
- Byly porovnány regulační pochody PSD regulátorů, Takahashiho regulátorů a obecných regulátorů pro různé možnosti nastavení, různé regulované soustavy a omezující podmínky. Tyto experimenty jsou uvedeny v kapitole 12.
- Pro všechny vybrané soustavy se podařilo nalézt takovou řídící rovnici obecného regulátoru, která svou strukturou a parametry dokáže zajistit kvalitnější průběh regulačního pochodu, než rovnice PSD regulátoru, případně její modifikace v Takahashiho tvaru.

Pro zajištění vysoké typografické kvality byla práce vysázena v programu LaTeX a všechny obrázky byly vlastnoručně nakresleny ve vysokém rozlišení. Všechny experimenty, optimalizační úlohy a výstupy simulací byly prováděny v programu, který jsem vytvořil pro tyto účely v jazyku C#.NET

Přílohy

14.1 Reprezentace čísla dle normy IEEE 754

double value	binary representation
-20	1100000000110100000000000000000000000000000000000000000000000000
-19	1100000000110011000000000000000000000000000000000000000000000000
-18	1100000000110010000000000000000000000000000000000000000000000000
-17	1100000000110001000000000000000000000000000000000000000000000000
-16	1100000000110000000000000000000000000000000000000000000000000000
-15	1100000000101110000000000000000000000000000000000000000000000000
-14	1100000000101100000000000000000000000000000000000000000000000000
-13	1100000000101010000000000000000000000000000000000000000000000000
-12	1100000000101000000000000000000000000000000000000000000000000000
-11	1100000000100110000000000000000000000000000000000000000000000000
-10	1100000000100100000000000000000000000000000000000000000000000000
-9	1100000000100010000000000000000000000000000000000000000000000000
-8	1100000000100000000000000000000000000000000000000000000000000000
-7	1100000000011100000000000000000000000000000000000000000000000000
-6	1100000000011000000000000000000000000000000000000000000000000000
-5	1100000000010100000000000000000000000000000000000000000000000000
-4	1100000000010000000000000000000000000000000000000000000000000000
-3	1100000000001000000000000000000000000000000000000000000000000000
-2	1100000000000000000000000000000000000000000000000000000000000000
-1	1011111111110000000000000000000000000000000000000000000000000000
0	0000000000000000000000000000000000000000000000000000000000000000
0,1	0011111110111001100110011001100110011001100110011001100110011010
0,2	0011111111001001100110011001100110011001100110011001100110011010
0,3	0011111111010011001100110011001100110011001100110011001100110100
0,4	0011111111011001100110011001100110011001100110011001100110011010
0,5	0011111111100000000000000000000000000000000000000000000000000000
0,6	0011111111100011001100110011001100110011001100110011001100110011
0,7	0011111111100110011001100110011001100110011001100110011001100110
0,8	0011111111101001100110011001100110011001100110011001100110011001
0,9	0011111111101100110011001100110011001100110011001100110011001100
1	0011111111110000000000000000000000000000000000000000000000000000
2	0100000000000000000000000000000000000000000000000000000000000000
3	0100000000001000000000000000000000000000000000000000000000000000
4	0100000000010000000000000000000000000000000000000000000000000000
5	0100000000010100000000000000000000000000000000000000000000000000
6	0100000000011000000000000000000000000000000000000000000000000000
7	0100000000011100000000000000000000000000000000000000000000000000
8	0100000000100000000000000000000000000000000000000000000000000000
9	0100000000100010000000000000000000000000000000000000000000000000
10	0100000000100100000000000000000000000000000000000000000000000000
11	0100000000100110000000000000000000000000000000000000000000000000
12	0100000000101000000000000000000000000000000000000000000000000000
13	0100000000101010000000000000000000000000000000000000000000000000
14	0100000000101100000000000000000000000000000000000000000000000000
15	0100000000101110000000000000000000000000000000000000000000000000
16	0100000000110000000000000000000000000000000000000000000000000000
17	0100000000110001000000000000000000000000000000000000000000000000
18	0100000000110010000000000000000000000000000000000000000000000000
19	0100000000110011000000000000000000000000000000000000000000000000
20	0100000000110100000000000000000000000000000000000000000000000000
1,79769313486232E+308	0111111111101111111111111111111111111111111111111111111111111111
-1,79769313486232E+308	1111111111101111111111111111111111111111111111111111111111111111
Není číslo	1111111111111000000000000000000000000000000000000000000000000000
+nekonečno	0111111111110000000000000000000000000000000000000000000000000000
-nekonečno	1111111111110000000000000000000000000000000000000000000000000000
4,94065645841247E-324	0000000000000000000000000000000000000000000000000000000000000001

Tabulka 27: Reprezentace čísla dle normy IEEE 754 [48]

14.2 Kódování *real* to *dec* to *bin* to *Gray code*

Význam jednotlivých řádků: **double** (reálné číslo), **to dec** (reálné číslo zakódované do dekadického tvaru), **dec to bin** (binární podoba dekadického čísla), **Gray dec** (dekadické číslo transformované do Grayova kódu v dekadické formě), **Gray bin** (dekadické číslo transformované do Grayova kódu v binární podobě, (viz. kapitola 3.6.1).

double	to dec	dec to bin	Gray dec	Gray bin	double	to dec	dec to bin	Gray dec	Gray bin
-5	0	0000000	0	0000000	0	64	1000000	96	1100000
-4,9	2	0000010	3	0000011	0,1	65	1000001	97	1100001
-4,8	3	0000011	2	0000010	0,2	67	1000011	98	1100010
-4,7	4	0000100	6	0000110	0,3	68	1000100	102	1100110
-4,6	6	0000110	5	0000101	0,4	69	1000101	103	1100111
-4,5	7	0000111	4	0000100	0,5	70	1000110	101	1100101
-4,4	8	0001000	12	0001100	0,6	72	1001000	108	1101100
-4,3	9	0001001	13	0001101	0,7	73	1001001	109	1101101
-4,2	11	0001011	14	0001110	0,8	74	1001010	111	1101111
-4,1	12	0001100	10	0001010	0,9	75	1001011	110	1101110
-4	13	0001101	11	0001011	1	77	1001101	107	1101011
-3,9	14	0001110	9	0001001	1,1	78	1001110	105	1101001
-3,8	16	0010000	24	0011000	1,2	79	1001111	104	1101000
-3,7	17	0010001	25	0011001	1,3	81	1010001	121	1111001
-3,6	18	0010010	27	0011011	1,4	82	1010010	123	1111011
-3,5	20	0010100	30	0011110	1,5	83	1010011	122	1111010
-3,4	21	0010101	31	0011111	1,6	84	1010100	126	1111110
-3,3	22	0010110	29	0011101	1,7	86	1010110	125	1111101
-3,2	23	0010111	28	0011100	1,8	87	1010111	124	1111100
-3,1	25	0011001	21	0010101	1,9	88	1011000	116	1110100
-3	26	0011010	23	0010111	2	89	1011001	117	1110101
-2,9	27	0011011	22	0010110	2,1	91	1011011	118	1110110
-2,8	28	0011100	18	0010010	2,2	92	1011100	114	1110010
-2,7	30	0011110	17	0010001	2,3	93	1011101	115	1110011
-2,6	31	0011111	16	0010000	2,4	94	1011110	113	1110001
-2,5	32	0100000	48	0110000	2,5	96	1100000	80	1010000
-2,4	34	0100010	51	0110011	2,6	97	1100001	81	1010001
-2,3	35	0100011	50	0110010	2,7	98	1100010	83	1010011
-2,2	36	0100100	54	0110110	2,8	100	1100100	86	1010110
-2,1	37	0100101	55	0110111	2,9	101	1100101	87	1010111
-2	39	0100111	52	0110100	3	102	1100110	85	1010101
-1,9	40	0101000	60	0111100	3,1	103	1100111	84	1010100
-1,8	41	0101001	61	0111101	3,2	105	1101001	93	1011101
-1,7	42	0101010	63	0111111	3,3	106	1101010	95	1011111
-1,6	44	0101100	58	0111010	3,4	107	1101011	94	1011110
-1,5	45	0101101	59	0111011	3,5	108	1101100	90	1011010
-1,4	46	0101110	57	0111001	3,6	110	1101110	89	1011001
-1,3	47	0101111	56	0111000	3,7	111	1101111	88	1011000
-1,2	49	0110001	41	0101001	3,8	112	1110000	72	1001000
-1,1	50	0110010	43	0101011	3,9	114	1110010	75	1001011
-1	51	0110011	42	0101010	4	115	1110011	74	1001010
-0,9	53	0110101	47	0101111	4,1	116	1110100	78	1001110
-0,8	54	0110110	45	0101101	4,2	117	1110101	79	1001111
-0,7	55	0110111	44	0101100	4,3	119	1110111	76	1001100
-0,6	56	0111000	36	0100100	4,4	120	1111000	68	1000100
-0,5	58	0111010	39	0100111	4,5	121	1111001	69	1000101
-0,4	59	0111011	38	0100110	4,6	122	1111010	71	1000111
-0,3	60	0111100	34	0100010	4,7	124	1111100	66	1000010
-0,2	61	0111101	35	0100011	4,8	125	1111101	67	1000011
-0,1	63	0111111	32	0100000	4,9	126	1111110	65	1000001
0	64	1000000	96	1100000	5	127	1111111	64	1000000

Tabulka 28: Real to Dec to Bin to GrayCode

Literatura

[1] Al-Somani, T. F.; Quershi, K. A New Parallel Genetic Algorithm Model. In *proceedings of 10th Annual IEEE Technical Exchange Meeting*, KFUPM, Dhahran, Saudi Arabia, March 23-24, 2003. [online]. Dostupné z https://eprints.kfupm.edu.sa/1484/1/s10p1.PDF

[2] AVINS, J. *Digital Signal Processing Central* [online]. 1999-2010, 28th. Sep. 2000 [cit. 2010-04-25]. Gray Code Conversion. Dostupné z WWW: <`http://www.dspguru.com/dsp/tricks/gray-code-conversion`>.

[3] BÄCK, T.; HAMMEL, U.; SCHWEFEL, H. P. Evolutionary computation : comments on the history and current state. In . *IEEE Transactions on Evolutionary Computation.* 1. [s.l.] : [s.n.], 1997. s. 3–17. ISSN 1089–778X.

[4] BACKUS, J. W., et al. *ALGOL–like Languages.* Volume 1. Cambridge, MA, USA : Birkhauser Boston Inc., 1997. Revised report on the algorithmic language ALGOL 60, s. 19–49. ISBN 0–8176–3880–6.

[5] BALÁTĚ, J. *Automatické Řízení.* 2. přepracované vydání. Praha : BEN, 2004. 664 s. ISBN 80-7300-148-9.

[6] BLICKLE, T.; THIELE, L. *A comparison of selection schemes used in evolutionary algorithms. Evolutionary Computation.* 1996, Volume 4, Issue 4, s. 361–394. ISSN 1063–6560.

[7] ČEŠKA, M.; RÁBOVÁ, Z. *Gramatiky a jazyky.* Brno : VUT Brno, 1992. 137 s. ISBN 80–214–0449–3.

[8] ČERNÁ, I.; KŘETÍNSKÝ, M.; KUČERA, A. *Automaty a formální jazyky I.* Brno : Učební text FI MU, 1999. 154 s.

[9] CHAKRABORTY, Uday K. *Advances in Differential Evolution.* Berlin: Springer, 2008. 319 s. ISBN 978–3–540–68827–3.

[10] CHAKRABORTY, U. K.; DEB, K.; CHAKRABORTY, M. Analysis of Selection Algorithms : A Markov Chain Approach. *Evolutionary Computation.* 1996, Volume 4, Issue 2, s. 133-167. ISSN 1063–6560.

[11] DAS, Swagatam; ABRAHAM, Ajith; KONAR, Amit. *Metaheuristic Clustering.* Berlin Heidelberg : Springer-Verlag, 2009. 254 s. ISBN 978–3–540–92172–1.

[12] DEMPSEY, I.; O'NEILL, M.; BRABAZON, A. Meta-grammar constant creation with grammatical evolution by grammatical evolution. In . *Genetic and Evolutionary Computation Conference.* Washington DC, USA : GECCO, 2005. s. 1665–1671. ISBN 1-59593-010-8.

[13] ENGELBRECHT, Andries P. *Computational Intelligence: An Introduction.* 2nd ed. England: Willey, 2007. 581 s. ISBN 978–0–470–03561–0.

[14] GOLDBERG, D. E. *Genetic Algorithms in Search, Optimization and Machine Learning.* 1. [s.l.] : Addison–Wesley Professional, 1989. 432 s. ISBN 0201157675, ISBN-13: 978–0201157673.

[15] HAVLÍN, V. *Preferenční křížení v evolučních algoritmech.* Praha, 2009. 70 s. Bakalářská práce. České vysoké učení technické v Praze, Fakulta elektrotechnická.

[16] HYNEK, J. *Genetické algoritmy a genetické programování.* [ČR] : Grada Publishing a.s., 2008. 182 s. ISBN 8024726955

[17] HEYWOOD, M. I. *Holland's GA Schema Theorem* [online]. 2010 [cit. 2010–04–11]. Genetic Algorithm and Programming. Dostupné z WWW: <`http://users.cs.dal.ca/~mheywood/CSCI6506/HandOuts/N02-GASchema.pdf`>

[18] HOLLAND, J. H. *Adaptation in Natural and Artificial Systems : An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence.* 2nd ed. [s.l.] : The MIT Press, 1992. 228 s. ISBN 0–262–58111–6, ISBN–13: 978–0–262–58111–0.

[19] HRSTKA, M. *Obecná biologie.* 1.vydání, Brno : Vysoké učení technické v Brně, fakulta chemická, 2005. 112 s. ISBN 80–214–3057–5.

[20] JAKOUBEK, P. Identifikace parametrů stabilních nekmitavých soustav. *Automatizace.* 2009, 52, 2, s. 84–89. Dostupný také z WWW: <`http://www.automatizace.cz/download/article/2444_a0209_84.pdf`>.

[21] KOZA, J.R. *Genetic Programming : On the Programming of Computers by Means of Natural Selection.* USA : The MIT Press, 1992. 835 s. ISBN 0–262–11170–5, ISBN–13: 978–0–262–11170–6.

[22] KRASNOGOR, N., et al. *Studies in Computational Inteligence : Nature Inspired Cooperative Strategies for Optimisation.* Berlin Heidelberg : Springer–Verlag, 2009. Genetic Algorithm and Advanced Tournament Selection Concept, s. 187–196. ISBN 978–3–642–03210–3.

[23] KŘENA, B. *Metody analýzy objektově orientovaných Petriho sítí.* Brno : Vysoké učení technické v Brně, 2004. 128 s. Dizertační práce. Fakulta informačních technologií, Vysoké učení technické v Brně.

[24] LAMPINEN, J.; ZELINKA, I. Mixed integer-discrete-continuous optimization by differential evolution: Part 1: the optimization method. In . *5th International Mendel Conference on Soft Computing.* Brno: Mendel, 1999. s. 71–76. ISBN 80–214–11.

[25] LEPS, M. *Moderní metody optimalizace* [online]. 25. Listopadu 2009 [cit. 2010-06-10]. Dostupné z WWW: <`http://klobouk.fsv.cvut.cz/~leps/teaching/mmo/`>.

[26] LIPTÁK, Béla G. *Process Control and Optimization : Instrument Engineers' Handbook.* 4th edition. Boca Raton : CRC Press, 2006. 2386 s. ISBN 0–8493–1081–4.

[27] MITCHELL, M. *An Introduction to Genetic Algorithms.* Cambridge, Massachusetts : The MIT Press, 1998. 221 s. ISBN 0–262–63185–7, ISBN–13: 978–0–262–63185–3.

[28] NOHEJL, A. *Grammatical Evolution.* [s.l.], 2009. 127 s. Bakalářská práce. Charles University in Prague, Faculty of Mathematics and Physics.

[29] O'NEILL, M; BRABAZON, A. Grammatical Differential Evolution. In . *International Conference on Artificial Intelligence, ICAI.* Volume 1. Las Vegas, Nevada, USA : CSREA Press, 2006. s. 231–236. ISBN 1-932415-96-3.

[30] O'NEILL, M.; RYAN, C. *Grammatical Evolution : Evolutionary Automatic Programming in an Arbitrary Language.* 1st edition. [s.l.] : Springer, 2003. 160 s. ISBN 1-4020-7444-1, ISBN-13: 978-1402074448.

[31] O'NEILL, M. *Grammatical-evolution.org* [online]. Last updated 01/04/2010 [cit. 2010-05-02]. Dostupné z WWW: <`http://www.grammatical-evolution.org`>.

[32] OŠMERA, P. *Advances in Computational Algorithms and Data Analysis : Lecture Notes in Electrical Engineering.* Netherlands : Springer, 2009. Two Level Parallel Grammatical Evolution, s. 509–525. ISBN 978-1-4020-8918-3.

[33] PAVLOVIČ, J. *Multikriteriální hybridní evoluční algoritmy pro výběr a optimalizaci dekontaminačních technologií* [online]. Brno : Masarykova univerzita, 2006. 30 s. Teze disertační práce. Masarykova univerzita. Dostupné z WWW: <`http://is.muni.cz/th/4035/fi_r/teze_extended.pdf`>

[34] POHLHEIM, H. *GEATbx : The Genetic and Evolutionary Algorithm Toolbox for Matlab* [online]. Berlin, Germany : 1994, 2007 [cit. 2010–04–15]. Dostupné z WWW: <`http://www.geatbx.com`>.

[35] POPELKA, O. *Použití evolučních a genetických algoritmů v ekonomických aplikacích.* Brno, 2009. 133 s. Disertační práce. Mendelova zemědělská a lesnická univerzita v Brně, Provozně ekonomická fakulta.

[36] PRICE, Kenneth V.; STORN, Rainer M.; LAMPINEN, Jouni A. *Differential Evolution: A Practical Approach to Global Optimization.* Berlín: Springer–Verlag, 2005. 535 s. ISBN 3–540–20950–6.

[37] QIN, A. K.; HUANG, V. L.; SUGANTHAN, P. N. Differential Evolution Algorithm With Strategy Adaptation for Global Numerical Optimization. In *IEEE: transactions on evolutionary computation.* vol. 13. New York: Institute of Electrical and Electronics Engineers, 2009. s. 398–417. ISSN 1089–778X.

[38] RYAN, C.; COLLINS, J. J.; O, M. Grammatical Evolution : Evolving Programs for an Arbitrary Language. In . *EuroGP : Proceedings of the First European Workshop on Genetic Programming.* London, UK : Springer-Verlag, 1998. s. 83–96. ISBN 3–540–64360–5.

[39] Ryan C., O'Neill M., Collins J.J. *Grammatical Evolution: Solving Trigonometric Identities.* In proceedings of Mendel 1998: 4th International Mendel Conference on Genetic Algorithms, Optimisation Problems, Fuzzy Logic, Neural Networks, Rough Sets. June 24–26 1998, Brno, Czech Republic, pages 111–119. ISBN 80–214–1199–6

[40] ŠVARC, I.; ŠEDA, M.; VÍTEČKOVÁ, M. *Automatické řízení.* první vydání. Brno : Akademické nakladatelství CERM, s.r.o., 2007. 324 s. ISBN 978–80–214–3491–2.

[41] TREUTWEIN, B. *CDC Algol–60 : Reference Manual* [online]. Version 5.Control Data Corporation , 1979 [cit. 2010–04–08]. Dostupné z WWW: <http://www.lrz-muenchen.de/~bernhard/Algol-BNF.html>

[42] WEISSER, R., *Optimization of Structure and Parameters of Controller.* In International Conference on Soft Computing Applied in Computer and Economic Environments ICSC. 8th. Hodonín : Evropský polytechnický institut, s.r.o., 2010. s. 159–166. ISBN 978-80-7314-201-8.

[43] WEISSER, R., et al, *Transplant Evolution for Optimization of General Controllers.* In European Conference on Modelling and Simulation. 24th. Kuala Lumpur (Malaysia) : ECMS 2010. s. 250 – 260. ISBN 978–0–9564944–0–5.

[44] WEISSER, R. Parallel Grammatical Evolution : Solving Trigonometric Identities with Backward Processing. In *International Confeence on Soft Computing MENDEL.* Brno : MENDEL, 2005. CD version. ISBN 80–214–2961–5.

[45] WEISSER, R., OŠMERA, P., MATOUŠEK, R., Transplant Evolution with Modified Schema of Differential Evolution : Optimization Structure of Controllers. In *International Confeence on Soft Computing MENDEL.* Brno : MENDEL, 2010.

[46] YANG, Z.; TANG, K.; YAO, X. Self-adaptive Differential Evolution with Neighborhood Search. In . *IEEE Congress on Evolutionary Computation.* Hong Kong, China: IEEE, 2008. s. 1110–1116. ISBN 978–1–4244–1822–0.

[47] ISO/IEC 14977:1996(E). *Information technology : Syntactic metalanguage–Extended BNF.* [s.l.] : ISO, 1996. 24 s. Dostupné z WWW: <http://standards.iso.org/ittf/licence.html>

[48] *IEEE Std 754-2008. IEEE Standard for Floating-Point Arithmetic.* [s.l.] : [s.n.], 2008. 70 s. ISBN: 978-0-7381-5753-5.

[49] *Rex Controls* [online]. 2000 [cit. 2010–05–18]. Dostupné z WWW: <http://www.rexcontrols.cz>.

[50] *TEX Users Group* [online]. 1980, 28.3.2010 [cit. 2010-04-25]. Dostupné z WWW: <http://www.tug.org/>.

[51] ZELINKA, I., et al. *Evoluční výpočetní techniky : principy a aplikace.* 1. vyd. Praha : BEN, 2009. 536 s. ISBN 978–80–7300–218–3.

[52] DNA *In Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 8 May 2010 [cit. 2010-05-10]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Dna>

[53] Messenger RNA *In Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 9 May 2010 [cit. 2010–05–10]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Messenger_RNA>

[54] Nucleic acid *In Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 9 May 2010 [cit. 2010–05–10]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Nucleic_acid>

[55] Ribosomal RNA *In Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 10 May 2010 [cit. 2010–05–10]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Ribosomal_RNA>

[56] Ribosome In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 4 May 2010 [cit. 2010–05–10]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Ribosome>.

[57] RNA *In Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 4 May 2010 [cit. 2010–05–10]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Rna>

[58] Transfer RNA *In Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 5 May 2010 [cit. 2010–05–10]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Transfer_RNA>

Seznam zkratek

AD Analogově digitální převodník (Analog-to-Digital converter)

ART Aritmetická redukce stromu (Arithmetic Tree Reducing)

EA Evoluční algoritmy (Evolution Algorithms)

BNF Backusova–Naurova forma (Backus–Naur form)

DA Digitálně analogový převodník (Digital-to-Analog converter)

DE Diferenciální evoluce (Differential Evolution)

DOBD Dopředná obdelníková náhrada

ES Evoluční strategie (Evolution Strategies)

FM Modul modul výpočtu fitness (Fitness Module)

GA Genetické algoritmy (Genetic Algorithms)

GE Gramatická evoluce (Grammatical Evolution)

GP Genetické programování (Genetic Programming)

IE Lineární regulační plocha (Integral of Error)

ISE Kvadratická regulační plocha (Integral of Squared Error)

ITAE ITAE (Integral of Time multiplied by Absolute Error)

LICHO Lichoběžníková náhrada

MDE Modul diferenciální evoluce (Differential Evolution Module)

MGA Modul genetického algoritmu (Genetic Algorithm Module)

MTE Modul transplantační evoluce (Transplant Evolution Module)

OM Výstupní modul (Output Module)

PD Proporcionálně–derivační

PI Proporcionálně–integrační

PID Proporcionálně–integračně–derivační

PSD Proporcionálně–sumačně–diferenční

TE Transplantační evoluce (Transplant Evolution)

TLTE Dvoufázová transplantační evoluce (Two Level Transplant Evolution)

ZOBD Zpětná obdelníková náhrada

Seznam symbolů

a_{max}	Horní mez prohledávaného intervalu
a_{min}	Dolní mez prohledávaného intervalu
CR	Pravděpodobnost křížení v diferenciální evoluci
c_s	Statický součinitel rychlosti $[s^{-1}]$
δ	Diracův impuls
$\Delta y_{\max}$	Maximální překmit (přeregulování)
$D(kT)$	Derivace v časovém okamžiku kT
$e(kT)$	Regulační odchylka v časovém okamžiku kT
f_s	Frekvence signálu
f_v	Vzorkovací frekvence
G	Generativní gramatika
G^{-1}	Analytická gramatika
$G(t)$	Generací evoluce v čase t
$G_R(S)$	Přenos regulátoru
$G_S(S)$	Přenos regulované soustavy
$G_W(S)$	Přenos řízení
γ	Šířka impulzu
$I(kT)$	Integrál v časovém okamžiku kT
k	Kardinalita množiny
κ	Relativní překmit
$\varkappa$	Délka celočíselného řetězce
k_R	Zesílení regulátoru
k_{Rk}	Kritické zesílení regulátoru
k_S	Statické zesílení regulované soustavy
L	Jazyk definovany gramatikou G
N	Množina neterminálních symbolů
NP	Velikost populace
ω_f	Vzorkovací kmitočet
ω_m	Maximální kmitočet spojitého signálu
ω_v	Vzorkovací kmitočet
P	Množina přepisovacích pravidel
P^{-1}	Inverzní množina pravidel k množině P
pp	Pásmo proporcionality
$P(T)$	Populace v čase t
$\nabla u(kT)$	Přirustek akční veličiny v časovém okamžiku kT
q_0, q_1, q_2	Parametry diskrétního regulátoru
r_0	Zesílení regulátoru
r_{0k}	Kritické zesílení regulátoru
r_{-1}	Integrační konstanta regulátoru
r_1	Derivační konstanta regulátoru
S	Startovní symbol gramatiky
Σ	Prostor řešení

T	Perioda vzorkování
T_0	Délka kmitu
T_d	Dopravní zpoždění
T_D	Derivační časová konstanta
T_{Dk}	Kritická hodnota derivační časové konstanty
T_I	Integrační časová konstanta
T_{Ik}	Kritická hodnota integrační časové konstanty
T_k	Kritická perioda kmitu
T_n	Doba náběhu
T_R	Doba regulace
T_u	Doba průtahu
$u(kT)$	Akční veličina regulátoru v časovém okamžiku kT
$u(t)$	Akční veličina regulátoru v čase t
$w(t)$	Žádaná hodnota v čase t
ξ_0	Koeficient poměrného tlumení
$y(kT)$	Regulovaná veličina v časovém okamžiku kT
$y(t)$	Regulovaná veličina v čase t

Typy použitých notací

Yk_n	$\Leftrightarrow$	Y_{k-n}	$\Leftrightarrow$	$y(k-n)T$	Regulovaná veličina
Ek_n	$\Leftrightarrow$	E_{k-n}	$\Leftrightarrow$	$e(k-n)T$	Regulační odchylka
Uk_n	$\Leftrightarrow$	U_{k-n}	$\Leftrightarrow$	$u(k-n)T$	Akční veličina regulátoru
DEk_n	$\Leftrightarrow$	dE_{k-n}	$\Leftrightarrow$	$\nabla e(k-n)T$	Regulační odchylka

Seznam obrázků

1 Rozdělení a příslušnost evolučních algoritmů . . . 11
2 Uložení genetické informace . . . 14
3 Ribozom . . . 15
4 Prohledávaný prostor . . . 18
5 Předzpracování počáteční populace jedinců GA . . . 18
6 Kódování pomocí stromu . . . 21
7 Uniformní křížení . . . 22
8 Dvoubodové křížení . . . 23
9 Jednobodové křížení (pořadové kódování) . . . 23
10 Inverzní mutace jednoho genu . . . 24
11 Jednobodová mutace (pořadové kódování) . . . 24
12 Roulette-wheel selection . . . 25
13 Linear ranking . . . 26
14 Exponential ranking . . . 27
15 Sigma scalling . . . 28
16 Boltzmann selection . . . 29
17 Disruptive selection . . . 30
18 Vývojový diagram diferenciální evoluce . . . 33
19 Hierarchie jazyků . . . 40
20 Vývojový diagram gramatické evoluce . . . 43
21 Chromozóm v gramatické evoluci . . . 43
22 Překlad genotypu na fenotyp . . . 45
23 Křížení jedinců s variabilní délkou chromozómu . . . 46
24 Zdvojování . . . 47
25 Operátor wrap . . . 48
26 Překlad genotypu na fenotyp - backward processing . . . 50
27 Modifikované křížení . . . 51
28 Modifikovaná mutace . . . 53
29 Ukázka chybně provedené mutace strukturálního kodonu . . . 54
30 Inicializace jedince - zpětné zpracování pravidel . . . 59
31 Inicializace jedince - dopředné zpracování pravidel . . . 60
32 Křížení částí stromů . . . 61
33 Spojování stromů . . . 62
34 Nestrukturální mutace . . . 64
35 Strukturální mutace . . . 65
36 Diagram dvoufázové transplantační evoluce . . . 69
37 Datový tok ve dvoufázové transplantační evoluci s oddělenými moduly . . . 71
38 Datový tok zobrazení výstupů . . . 73
39 Začlenění ART do TE (TLTE) . . . 74
40 ART – vybrané typy možných redukcí . . . 75
41 Datový tok meta–evoluce . . . 77
42 Základní zapojení jednoduchého regulačního obvodu . . . 80
43 Rozdělení jednorozměrových regulačních obvodů . . . 80
44 Nekmitavá proporcionální regulovaná soustava . . . 81

45 Kmitavá proporcionální regulovaná soustava 83
46 Integrační regulovaná soustava 84
47 Blokové schéma spojitého regulátoru 85
48 Blokové schéma diskrétního regulátoru 86
49 Vzorkovací perioda a aliasing 88
50 Regulační smyčka s respektováním saturace akční veličiny 91
51 Kritické zesílení 93
52 Ziegler–Nichols (metoda čtvrtinového tlumení) 93
53 Určení doby průtahu T_u a doby náběhu T_n 94
54 Hodnocení kvality regulace 97
55 Kritéria hodnocení 101
56 Průběh optimalizace PSD regulátoru pomocí DE 102
57 Průběh optimalizace parametrů obecného regulátoru pomocí DE 103
58 Porovnání regulačních pochodů PSD regulátorů 108
59 Průběh evoluce parametrů PSD regulátoru v MDE 109
60 Průběhy regulací v různých generacích evoluce 110
61 Porovnání regulačních pochodů Takahashiho regulátoru 111
62 Porovnání PSD regulátoru a obecného regulátoru stejného řádu 113
63 Porovnání PSD regulátorů a obecného regulátoru s obecnou řídicí rovnicí . 114
64 Proporcionální kmitavá soustava (pásmo necitlivosti) 116
65 Porovnání PSD regulátoru, obecného regulátoru s řídící rovnicí druhého řádu a obecného regulátoru s obecnou řídicí rovnicí 117
66 Průběh optimalizace struktury a parametrů obecného regulátoru metodou TLTE 118
67 Porovnání akčních zásahů PSD regulátoru, obecného regulátoru s řídící rovnicí druhého řádu a obecného regulátoru s obecnou řídicí rovnicí 119
68 Porovnání PSD regulátoru, obecného regulátoru s řídící rovnicí druhého řádu, kriteriální funkce s penalizací překmitu regulované veličiny 120
69 Regulační pochody chybně optimalizovaných obecných regulátorů 121
70 Regulační pochod obecného regulátoru *General_5* 122
71 Regulační pochod obecného regulátoru *General_6* 123
72 Regulační pochod obecného regulátoru *General_7* 124
73 Ukázky chyb ($\pm\infty$) v průběhu simulace regulace 125
74 Porovnání kriteriálních funkcí 126
75 Porovnání řídicích rovnic regulátorů (různé sady pravidel, různé kriteriální funkce) 127
76 Detailní porovnání řídicích rovnic regulátorů (různé sady pravidel, různé kriteriální funkce) 128
77 Porovnání obecných regulátorů se setrvačností prvního řádu a PSD regulátorů, s pásmem necitlivosti – kmitavá proporcionální soustava bez dopravního zpoždění 130
78 Porovnání obecných regulátorů se setrvačností prvního řádu a PSD regulátorů, bez pásma necitlivosti – kmitavá proporcionální soustava bez dopravního zpoždění 131
79 Průběh regulačního pochodu regulátoru *General_12* v 10.generaci TLTE . 132

Seznam tabulek

1 Binární kódování, Hammingova bariéra 19
2 Celočíselné pořadové kódování 20
3 Kódování pomocí reálných čísel 21
4 Symboly používané k vyjádření BNF, EBNF 42
5 Pravidla gramatiky v prefixovém BNF tvaru 45
6 Pravidla použitá pro zpracování reálných čísel [12] 46
7 Generování reálných čísel 55
8 Změněná gramatika v důsledku překročení délky stromu 58
9 Pravidla analytické gramatiky pro nestrukturální mutaci 63
10 Pravidla analytické gramatiky pro zkracující strukturální mutaci 65
11 Pravidla analytické gramatiky pro prodlužující strukturální mutaci 66
12 Rozšiřující pravidla generativní gramatiky 66
13 Rozšiřující pravidlo generativní gramatiky 70
14 Rozšiřující pravidlo analytické gramatiky 70
15 Pravidla pro ART (základní operace) 74
16 Pravidla pro ART (pokročilé operace) 75
17 Varianty přepočtu parametrů PID na PSD 90
18 Parametry regulátorů podle Ziegler–Nicholse 93
19 Parametry Takahashiho regulátoru 93
20 Seřízení spojitých regulátorů podle Ziegler–Nicholse (čtvrtinové tlumení) . 94
21 Nastavení spojitých regulátoru podle T_u, T_n 95
22 Závislost koeficientů α, β na relativním překmitu κ 95
23 Nastavení PID (PSD) regulátoru metodou inverze dynamiky 95
24 Pravidla generativní gramatiky pro optimalizaci regulátorů 99
25 Pravidla analytické gramatiky pro optimalizaci regulátorů 100
26 Pravidla pro ART 100
27 Reprezentace čísla dle normy IEEE 754 [48] 139
28 Real to Dec to Bin to GrayCode 140

Seznam algoritmů

1 Generační GA (generational GA) 16
2 Ustálený GA (steady-state GA) 16
3 Překlad genotypu na fenotyp 44
4 Překlad genotypu na fenotyp - backward processing 49
5 Párování značek 55
6 Inicializace jedince 58
7 Podmínky zastavení 105

Rejstřík

Backus-Naur form, *viz* BNF gramatiky
BNF gramatiky, 41

diferenciální evoluce, 33
  auto-adaptivní diferenciální evoluce, 36
    SaCR, 37
  DE/Best/1, 35
  DE/Best/1_mod, 104
  DE/Best/1plus, *viz* DE/Best/1_mod
  DE/Best/2, 35
  DE/current-to-best/$1 + n_v$, 36
  DE/current-to-best/1, 35
  DE/rand-to-best/n_v, 36
  DE/Rand/1, 34
  DE/Rand/1_mod, 104
  DE/Rand/1plus, *viz* DE/Rand/1_mod
  DE/Rand/2, 35
  DE/x/n_v, 35
  diferenční vektor, 34, 104
  faktor zesílení, 33
  inicializace populace, 34
  parametrický vektor, 33
  pravděpodobnost křížení, 33
  práh křížení, 34
  reprodukční cyklus, 34
  SaDE, *viz* auto-adaptivní diferenciální evoluce
  SaNSDE, 37
  selekční schémata, 34
  velikost populace, 33
  vývojový diagram, 33
  zesílení diferenčního vektoru, 33
  zkušební vektor, 34
  šumový vektor, 34, 104
  řídící parametry, 33
dvoufázová transplantační evoluce, 56, 69
  architektura, 71
  diferenciální evoluce, 71
  druhá fáze, 70
    diferenciální evoluce, 70
    genetický algoritmus, 70
  generování abstraktních numerických parametrů, 70
  mezivrstva, 71, 73
    interface, 73
  model řešení, 72
  modul DE, 71, 72
    vstupní parametry, 72
    výstupní parametry, 72
  modul GA, 72
  modul TE, 71, 72
    vstupní parametry, 72
    výstupní parametry, 72
  modul výpočtu fitness, 71, 72
    multikriteriální hodnocení, 72
    vstupní parametry, 72
    výstupní parametry, 72
  nevýhody, 70
  první fáze, 70
  transplantační evoluce, 71
  výstupní modul, 73
  značkování abstraktních proměnných, 70
  časová složitost, 70

Euklidovská metrika, 31
evoluční algoritmy, 11
  evoluční strategie, 11
  rozdělení evolučních algoritmů, 11

genetické algoritmy, 13
  adenin, 13, 14
  alela, 13
  aminokyseliny, 14
  binární kódování, 19
  biologické pozadí, 13
  bílkovina, 14
  chromozóm, 13
  cytoplazma, 14
  cytosin, 13, 14
  diploidní genom, 14
  DNA, 13, 57
  fenotyp, 13, 57
  gameta, 13

gen, 13
generační model, 15
genetické programování, 13
genom, 13
genotyp, 13, 57
Grayův kód, 19
guanin, 13, 14
Hammingova bariéra, 19
haploidní genom, 13
inicializace, 18
jednovláknová nukleová kyselina (RNA), 14
kriteriální funkce, 15
křížení, 22
  aritmetické křížení, 23
  bodové křížení, 22
  uniformní křížení, 22
lokus, 13
miRNA, 14
mRNA, 14
mutace, 23
nukleové kyseliny, 13
populace, 15
prohledávaný prostor, 17
reprodukce, 15, 21
ribozom, 14
rRNA, 14
selekce, 24
  Boltzmann selection, 28
  disruptive selection, 30
  elitismus, 30
  exponential ranking, 26
  linear ranking, 25
  proporcionální selekce, *viz* roulette-wheel selection
  roulette-wheel selection, 24
  sigma scaling, 27
  tournament selection, 27
  zkrácený výběr, 30
selekční tlak, 31
stavový prostor, *viz* prohledávaný prostor
teorie RNA světa, *viz* teorie vzniku života
teorie schémat, 16
  definiční délka, 16
  schéma, 16
  řád schématu, 16
teorie vzniku života, 14
thymin, 13
transkripce DNA, 14
tRNA, 14
uracil, 14
ustálený model, 15
zabránění incestu, 31
geny, 43
gramatická evoluce, 43
  chybně provedená mutace, 53
  generování čísel pomocí gramatiky, 55
    desetinná tečka, 55
    iracionální čísla, 55
  genotype to phenotype mapping, *viz* překlad genotypu na fenotyp
  modifikovaná mutace, 52
  modifikované křížení, 51
  nestrukturální mutace, 52, 53
  operátory, 46
    bits-level mutation, 47
    codons-level mutation, 47
    duplication, 47
    křížení, 46
    mutace, 47
    wrap, 48
  párování značek, 54
  předklad genotypu na fenotyp, 44, 45
  předklad genotypu na fenotyp – backward processing, *viz* zpětné zpracování pravidel
  strukturální mutace, 52
  vektor značek, 50
  značkování kodonů, 48
  zpětné zpracování pravidel, 48–50

Hammingova vzdálenost, 31

IEEE 754, 19, 139

jazyky a gramatika, 39
  abeceda, 39
  analytická gramatika, 39, 62, 64, 66
  generativní gramatika, 39, 62, 64, 66
  gramatika, 39
  inverzní množina pravidel, 63

jazyk, 39
N–pravidla, 48, 52, 53, 58
neterminální symbol, 61–64, 66
pravidla analytické gramatiky, 62, 63
prázdný řetězec, 39
symboly abecedy, 39
T–pravidla, 48, 53
terminální symbol, 62–64, 66
řetězec, 39
jednokriteriální selekční metody, 24

kodon, 43

meta–evoluce, 77
diferenciální evoluce, 77
MDE, 78
optimalizace parametrů DE, 77
modulo, 44

paralelní a distribuované výpočty, 133
Amdahl's law, *viz* Amdahlův zákon
Amdahlův zákon, 133
praktická část, 99
kriteriální funkce, 99
hodnocení simulace v případě chyby, 102
podmínky zastavení, 104, 105
úplná gramatika, 99

regulační obvody, 79
akční veličina, 79
akční zásah regulátoru, 79
aperiodické soustavy, 82
chyba, 79
diskrétní, 80
doba náběhu, 81
doba průtahu, 81
doba přechodu, 81
dopravní zpoždění, 81
identifikace soustav, 81
inflexní bod, 81, 82
Integrační regulované soustavy, 84
kmitavé proporcionální regulované soustavy, 83
koeficient poměrného tlumení, 83
koeficient přenosu soustavy, *viz* zesílení soustavy
nekmitavé proporcionální soustavy, 81
nekmitavé soustavy, *viz* aperiodické soustavy
přechodová charakteristika, 81
regulační odchylka, 79
regulovaná soustava, 79
regulovaná veličina, 79
regulátor, 79
rozvětvené jednorozměrové regulační obvody, 80
spojité, 80
Strejcova metoda, 82
základní zapojení, 79
žádaná hodnota regulované veličiny, 79
regulátory, 85
absolutní regulační plocha, 98
akční veličina, 85
automatická regulace, 85
blokové schéma, 85
derivační časová konstanta, 91
diskrétní regulátory, 86
AD/DA převodník, 86
aliasing, 86
amplituda, 87
antiwind-up, 91
AČ/ČA převodník, *viz* AD/DA
Diracův impulz, 87
impulz, 86
paměťový člen, 86
přibližné určení vzorkovací periody, 87
regulační orgán, 86
Shannon–Kotělníkovův teorém, 86
tvarovač, 86
tvarování, 87
vzorkovací kmitočet, 86
vzorkovací perioda, 86
vzorkovač, 86
vzorkování, 86
wind-up zpoždění, 92
dělení regulátorů, 85
historie, 79
hranice stability, 92
Hurwitzova matice, 94
Hurwitzův determinant, 94
Integral of Absolute Error, *viz* abso-

lutní regulační plocha
Integral of Error, *viz* lineární regulační plocha
Integral of Squared Error, *viz* kvadratická regulační plocha
Integral of Time multiplied by Absolute Error, *viz* ITAE
Integral of Time-weighted Squared Error, *viz* ITSE
integrační časová konstanta, 91
ITAE, 98
ITSE, 98
kritická perioda kmitů, 92
kritické zesílení, 92
kritické zesílení regulátoru, 94
kvalita regulace, 97
doba regulace, 97
dynamické hledisko, 97
maximální překmit, 98
minimální doba regulace, 98
perioda kmitů, 97
přeregulování, *viz* relativní překmit
relativní překmit, 97
statické hledisko, 97
trvalá regulační odchylka, 97
velikost regulační plochy, 97, 98
časová oblast, 97
lineární regulační plocha, 98
metoda požadovaného modelu, *viz* Seřízení regulátorů metodou inverze dynamiky
metoda čtvrtinového tlumení, 92
modifikace algoritmů diskrétních regulátorů, 90
měřící člen, 85
odchylka, 85
PID regulátor, 87
přenos v Laplaceově transformaci, 87
rovnice ideálního regulátoru, 87
porovnávací člen, 85
PSD regulátor, 89
náhrada derivace zpětnou diferencí, 89
náhrada integrace DOBD, 89
náhrada integrace LICHO, 89
náhrada integrace ZOBD, 89
polohový algoritmus, 89
přírustkový algoritmus, 89
vzorkovací perioda, 89
pásmo proporcionality, 91
regulační orgán, 85
regulátory s pevně danou strukturou, 87
saturace, 92
seřízení regulátoru metodou pokusu a omylu, 96
seřízení regulátoru podle funkcí standardního tvaru, 96
Seřízení regulátorů metodou inverze dynamiky, 95
seřízení regulátorů metodou kritického zesílení
see(seřízení regulátorů metodou Ziegler–Nichols), 92
seřízení regulátorů metodou Ziegler–Nichols, 92
seřízení regulátorů podle přechodové charakteristiky, 94
snímače, 85
Takahashiho algoritmus, 90, 99
zesílení regulátoru, 98
číslicové regulátory, *viz* diskrétní regulátory
řídicí algoritmy regulátorů, 87
ústřední člen, 85

stochastické metody, 12
Brute–Force Search, 12
Hill climbing, 12
Nelder–Mead, 12
Random Walk, 12
Simulated Annealing, 12
Tabu search, 12

transplant differential evolution, *viz* dvoufázová transplantační evoluce
transplant evolution, *viz* transplantační evoluce
architecture, *viz* architektura
Arithmetic Tree Reducing, *viz* aritmetická redukce stromu
crossover, *viz* křížení

DE module, *viz* modul DE
extending structural mutation, *viz* prodlužující strukturální mutace, *viz* prodlužující strukturální mutace
Fitness Module, *viz* modul výpočtu fitness
GA module, *viz* modul GA
Interlayer, *viz* mezivrstva
mutation, *viz* mutace
non-structural mutation, *viz* nestrukturální mutace
structural mutation, *viz* strukturální mutace
Output Module, *viz* výstupní modul
shortening mutation, *viz* zkracující strukturální mutace
shortening structural mutation, *viz* zkracující strukturální mutace
TE module, *viz* modul TE
transplantation, *viz* crossover
transplantační evoluce, 56, 57
aritmetická redukce stromu, 74
bezztrátová redukce, 74
pravidla, 74
slovník, 74
DNA, 60
dopředné zpracování pravidel, 57
dopředný překlad, 59
dynamická změna pravidel, 58
gramatika, 57
inicializace jedince, 57, 58
křížení, 60
křížení podstromů, 60
křížení uzlů, 61
křížení částí stromů, 60
spojování stromů, 62
listy stromu, 66
mutace, 62
cílená prodlužující mutace, *viz* prodlužující strukturální mutace
cílená zkracující mutace, *viz* zkracující strukturální mutace
diverzita populace, 62
fenotyp, 62
nestrukturální mutace, 63
obecná mutace, 64
pravděpodobnost výběru typu mutace, 67
prodlužující strukturální mutace, 64, 66
rekombinace, 62
stromová struktura, 62
strukturální mutace, 64
translace genotypu, 62
zkracující strukturální mutace, 64, 67
nestrukturální uzel, 61
pravděpodobnost výběru pravidla, 63, 67
strukturální uzel, 61
transplantace, 60
větvení stromu, 64
zpětné zpracování pravidel, 57
částečná transplantace, *viz* křížení uzlů
úplná transplantace, *viz* křížení podstromů
two level transplant evolution, *viz* dvoufázová transplantační evoluce

vícekriteriální selekční metody, 24