VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV AUTOMATIZACE A INFORMATIKY
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

OPTIMALIZACE PID REGULÁTORU POMOCÍ EVOLUČNÍCH VÝPOČETNÍCH TECHNIK
OPTIMIZATION OF PID CONTROLLER USING EVOLUTIONARY COMPUTING TECHNIQUES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Jakub Kočí**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. et Ing. Stanislav Lang**
SUPERVISOR

BRNO 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání bakalářské práce

Ústav:	Ústav automatizace a informatiky
Student:	**Jakub Kočí**
Studijní program:	Strojírenství
Studijní obor:	Základy strojního inženýrství
Vedoucí práce:	**Ing. et Ing. Stanislav Lang, Ph.D.**
Akademický rok:	2017/18

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Optimalizace PID regulátoru pomocí evolučních výpočetních technik

Stručná charakteristika problematiky úkolu:

Bakalářské práce bude věnována problematice optimalizace parametrů PID regulátoru pomocí evolučních výpočetních technik. Kvalita regulace bude hodnocena na základě integrálního kritéria (např. ITAE), případně i dalších vhodně zvolených požadavků na regulační děj.

Cíle bakalářské práce:

Popište vybrané evoluční výpočetní techniky.
Stručně shrňte problematiku regulace (věnujte pozornost PID regulátoru).
Realizujte simulační model v prostředí Matlab & Simulink.
Optimalizujte parametry PID regulátoru s využitím evolučních výpočetních technik.
Srovnejte a zhodnoťte dosažené výsledky.

Seznam doporučené literatury:

ŠVARC, I., ŠEDA, M. a VÍTEČKOVÁ, M. (2007): Automatické řízení. CERM, Brno.

BLAHA, P. a VAVŘÍN, P. (2005): Řízení a regulace I. Základy regulace lineárních systémů - spojité a diskrétní. VUT FEKT, Brno.

Abstrakt

Tato bakalářská práce se zabývá použitím evolučních výpočetních technik při nastavování PID regulátoru. V rešeršní části shrnuje základní informace o problematice regulace a další potřebné podkladové informace o kvalitě regulace a použitém kritériu ITAE. V praktické části jsou autorem implementovány tři evoluční výpočetní techniky - diferenciální evoluce, evoluční strategie a genetický algoritmus. Společně s genetickým algoritmem z MATLABu jsou tyto metody srovnány na dvou soustavách vzájemně i oproti prvotnímu nastavení regulátoru metodou Ziegler-Nichols. Součástí srovnání je i statistické vyhodnocení metod na jedné ze soustav.

Summary

This bachelor thesis deals with using evolutionary computation for tuning up PID controller. In research part there are summarised information about regulation and another background information about quality of regulation and ITAE criterion. Practical part consist of implementing three evolutionary computation algorithms - differential evolution, evolution strategy and genetic algorithm. These and MATLAB's function ga() are compared on two systems mutually and to Ziegler-Nichols rule. Basic comparsion is followed by statistical evaluation on second system.

Klíčová slova

Evoluční výpočetní techniky, genetické algoritmy, evoluční strategie, diferenciální evoluce, PID regulátor, optimalizace nastavení regulátoru, Matlab, Simulink

Keywords

Evolutionary computation, genetic algorithm, evolutionary strategy, differential evolution, PID controller, controller settings optimalization, Matlab, Simulink

KOČÍ, J. *Optimalizace PID regulátoru pomocí evolučních výpočetních technik*. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2018. 63 s. Vedoucí bakalářské práce Ing. et Ing. Stanislav Lang.

Prohlašuji, že tato práce je mým původním dílem, zpracoval jsem ji samostatně pod vedením Ing. et Ing. Stanislava Langa a s použitím literatury uvedené v seznamu literatury.

V Brně dne 20. května 2018 Jakub Kočí

Rád bych poděkoval svému vedoucímu za všestrannou pomoc a cenné připomínky při psaní této práce.

Jakub Kočí

Obsah

1 Úvod

Pro správnou funkci regulačního obvodu je důležité správně vybrat a nastavit regulátor odpovídající soustavě. Tato práce se bude zabývat pouze nastavením PID regulátoru jako průmyslově nejpoužívanějšího regulátoru. Metody nastavení PID regulátoru jsou vesměs tyto: pokus-omyl, heuristické metody (např. nastavení dle Zieglera a Nicholse), analytické a simulační. Metoda pokusu a omylu má zjevné problémy - vyžaduje zkušené pracovníky, není deterministická a nelze snadno určit optimální nastavení, nepotřebuje ale žádné výpočetní prostředky a lze jí nastavit regulátor na místě. Heuristické metody nejsou matematicky dokázané, vyžadují přípravu a manuální zásahy ještě před výpočty (které jsou ale velmi jednoduché), jejich výsledky jsou ale velmi hrubé (metoda Ziegler-Nichols produkuje nastavení se třemi až čtyřmi viditelnými kmity a překmit okolo 25%, což může vést k opotřebení akčních členů). Analytické metody jsou náročné na výpočet a vyžadují co nejpřesnější model soustavy. Tyto nedostatky spolu s dostupnější a výkonnější výpočetní technikou vedou k vzestupu simulačních metod.

Evoluční výpočetní prostředky řeší problém hledání optima kriteriální funkce za pomocí většího množství řešení, které podle pravidel dané metody podstupují změny, které převážně vedou k lepším hodnotám kriteriální funkce. Nepožadují prakticky žádné vlastnosti kriteriální funkce - tato musí pouze vystihovat, které řešení je přípustné a číselně ohodnotit jeho kvalitu. Kriteriální funkce nemusí být ani spojitá, což je vlastnost, kterou požadují všechny konvenční (např. gradientní) metody. Navíc jsou velice odolné proti prohlášení lokálního extrému za globální optimum.

Výhody použití evolučních výpočetních technik pro nastavení parametrů PID regulátoru (za použití vhodného kritéria kvality regulace jako kriteriální funkci) jsou zřejmé. Nevýhodou použití evolučních výpočetních technik je jejich výpočetní náročnost (je potřeba velkého množství vyhodnocení kriteriální funkce) a jejich nedeterminičnost. Je možné, že evoluční postupy nedojdou ke globálnímu optimu, ale v praxi se ukazuje, že při dobrém nastavení evolučního algoritmu (velikost populace a podmínka ukončení) dojde algoritmus k řešení blízkému skutečnému optimu. Tento výsledek lze použít jako vstup pro konvenční (např. gradientní) optimalizační metody.

2 Regulace

[1, 2, 3]

Cílem regulace je působit na řízenou soustavu tak, aby bylo dosaženo požadovaného chování. V případě regulace jde o řízení obvodu se zpětnou vazbou (na rozdíl od ovládání, kde se zpětná vazba nevyskytuje).

2.1 Základní pojmy

Rozlišujeme regulované obvody lineární a nelineární. Lineární systémy lze popsat lineárními diferenciálními rovnicemi, platí pro ně následující pravidla:

- Princip superpozice. Odezva systému na několik vstupů působících zároveň je součtem odezev vyvolaných jednotlivými vstupy.
- Násobení konstantou. Na k-násobný vstup odpoví soustava k-násobným výstupem.

V praxi se vyskytuje jen velmi málo lineárních obvodů, ale některé systémy lze (zejm. v okolí pracovního bodu) linearizovat se zanedbatelnou chybou. Nelineární rovnice lze poté v blízkém okolí pracovního bodu nahradit lineárními. Dobře lze linearizovat obvody s tzv. parazitními nelinearitami, tj. nelinearitami, které vznikají z konstrukčních důvodů. Špatně nebo vůbec se nedají linearizovat soustavy s podstatnými nelinearitami, které jsou zamýšlenou součástí obvodu. Nejčastější podstatné nelinearity jsou prvky, které dávají jen několikahodnotový výstup - např. relé.

Dále lze rozlišovat obvody podle toho, jestli regulační zásah probíhá spojitě (spojité řízení), nebo jen v určitých časových okamžicích (diskrétní řízení).

V regulovaných obvodech rozlišujeme následující veličiny:

Regulovaná veličina (y) je veličina na výstupu soustavy.

Řídicí veličina (w) též nazývaná žádaná hodnota. Cílem regulace je přiblížit průběh regulované veličiny průběhu řídicí veličiny.

Regulační odchylka (e) je rozdíl mezi žádanou hodnotou a regulovanou veličinou

Akční veličina (x, popř. u) je vstupní veličina soustavy a výstupní veličina regulátoru. Změnou akční veličiny se působí na řídicí veličinu.

Porucha (v) je veličina působící na obecném místě soustavy, obecně nemá známý průběh. Poruchy dělíme na signálové (působí na některou veličinu soustavy) a parametrické (dochází ke změnám parametrů soustavy)

Podle průběhu řídicí veličiny lze dále rozlišovat tři případy

- Řídicí veličina je po celou dobu regulace konstantní (po částech konstantní). Úlohou regulátoru je kompenzace poruch. Vyskytuje se u řízení základních fyzikálních veličin - teplota, tlak, délka (např. výška hladiny), otáčky...
- Řídicí veličina se mění s předem známým průběhem. Úlohou regulátoru je jak kompenzace poruch, tak působení na soustavu, aby regulovaná veličina co nejpřesněji sledovala průběh žádané hodnoty. Takové řízení nazýváme programovým.

- Řídicí veličina se mění s předem neznámým průběhem. Hlavní úlohou regulátoru je působit akční veličinou tak, aby regulovaná veličina co nejpřesněji sledovala průběh řídicí veličiny. Kompenzace poruch je obvykle druhořadá. Typickým příkladem je servomechanismus - sledování polohy/natočení.

2.2 Stabilita obvodu

[2, 3]

Pro zajištění funkcí regulace je nezbytné, aby regulační obvod (regulátor a řízená soustava) byl stabilní. Za stabilní prohlásíme ten obvod, který je schopen se po vychýlení z rovnovážného stavu a odstranění vzruchu, který vychýlení způsobil, znovu ustálit v rovnovážném stavu. Pokud obvod začne kmitat s konstantní amplitudou, hovoříme o obvodu na hranici stability. Pokud se obvod rozkmitává nade všechny meze, hovoříme o obvodu nestabilním. Nutnou a postačující podmínkou stability je tzv. obecná podmínka stability.

2.2.1 Obecná podmínka stability

Obvod je stabilní, pokud jsou všechny kořeny charakteristické rovnice 2.1 se zápornou reálnou částí (G_o je přenos otevřeného obvodu). Pokud některý z kořenů charakteristické rovnice leží na imaginární ose, je obvod na hranici stability. Dále lze odvodit, že aby byl obvod stabilní, všechny koeficienty charakteristické rovnice musí být kladné. Tato podmínka je nutná, nikoliv však postačující[1].

$$G_o(s) + 1 = a_n s^n + a_{n-1} s^{n-1} + ... + a_1 s + a_0 = 0 \tag{2.1}$$

Protože řešení rovnice n-tého stupně je náročné, byly odvozeny vhodnější metody zjišťování stability obvodu - kritéria stability. Dělíme je na algebraická (Hurwitzovo a Routh-Schurovo) a frekvenční (Michajlov-Leonhardovo a Nyquistovo). Zde uvedený výčet kritérií není úplný, tato kritéria jsou ale nejpoužívanější.

2.2.2 Hurwitzovo kritérium stability

Z koeficientů charakteristické rovnice sestrojíme matici H_n (2.2) a matice H_{n-1} až H_1tak, že z předchozí matice vynecháme poslední řádek a poslední sloupec. Obvod je stabilní, pokud jsou všechny determinanty matic H (nazývané Hurwitzovy determinanty) kladné. Pokud je některý z determinantů nulový, je obvod na hranici stability. Hurwitzovo kritériem je mimo jiné vhodné i pro zjišťování intervalů parametrů, kdy je obvod stabilní.

[1] V případě, že charakteristická rovnice je kvadratická, je tato podmínka nutná a postačující.

$$H_n = \begin{pmatrix} a_{n-1} & a_{n-3} & a_{n-5} & \dots & 0 & 0 & 0 & 0 \\ a_n & a_{n-2} & a_{n-4} & \dots & 0 & 0 & 0 & 0 \\ 0 & a_{n-1} & a_{n-3} & \dots & 0 & 0 & 0 & 0 \\ 0 & a_n & a_{n-2} & \dots & 0 & 0 & 0 & 0 \\ 0 & 0 & a_{n-1} & \dots & 0 & 0 & 0 & 0 \\ 0 & 0 & a_n & \dots & 0 & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & a_3 & a_1 & 0 & 0 \\ 0 & 0 & 0 & \dots & a_4 & a_2 & a_0 & 0 \\ 0 & 0 & 0 & \dots & a_5 & a_3 & a_1 & 0 \\ 0 & 0 & 0 & \dots & a_6 & a_4 & a_2 & a_0 \end{pmatrix} \tag{2.2}$$

2.2.3 Routh-Schurovo kritérium stability

Routh-Schurovo kritérium vychází též z charakteristické rovnice obvodu. V každém kroku se rovnice zredukuje o jeden řád.

Algoritmus redukce

1. Sepíšeme sestupně všechny koeficienty.
2. Sudé koeficienty sepíšeme do druhého řádku o jedno místo posunuté vlevo a vynásobíme podílem dvou nejvyšších koeficientů $\frac{a_n}{a_{n-1}}$ (pokud je místo prázdné, uvažujeme nulu).
3. Druhý řádek odečteme od prvního, tím dostáváme třetí řádek. Ve třetím řádku jsou koeficienty rovnice po redukci.

Redukci provádíme opakovaně, než dojdeme k rovnici druhého stupně. Platí, že pokud jsou všechny koeficienty a_n všech rovnic (charakteristické rovnice obvodu i rovnic vzniklých redukcí) kladné, je obvod stabilní.

2.2.4 Michajlov-Leonhardovo kritérium stability

Toto frekvenční kritérium vychází z charakteristické rovnice obvodu. Z levé strany utvořme funkci $H(s) = a_n s^n + a_{n-1} s^{n-1} + ... + a_1 s + a_0$. Kritérium říká, že obvod je stabilní, pokud koncový bod vektoru $H(j\omega)$ pro $\omega \in \langle 0; \infty)$projde popořadě proti směru hodinových ručiček tolika kvadranty, jako je stupeň charakteristické rovnice. Pokud křivka prochází bodem $[0; 0]$, je obvod na hranici stability.

2.2.5 Nyquistovo kritérium stability

Nyquistovo kritérium vychází z frekvenční charakteristiky rozpojeného obvodu. Platí, že pokud bod $[-1; 0]$ leží vlevo od frekvenční charakteristiky $G_0(j\omega)$ pro $\omega \in \langle 0; \infty)$, je obvod stabilní. Pokud křivka prochází kritickým bodem $[-1; 0]$, je obvod na hranici stability.

Výhodou Nyquistova kritéria je, že dokáže obvod hodnotit také kvalitativně (málo stabilní/hodně stabilní), a není ani potřeba znát analytický tvar frekvenční charakteristiky, stačí experimentálně získaná křivka. Lze použít také na soustavy s dopravním zpožděním, kdy algebraická kritéria selhávají.

2.3 Kvalita regulace

[4, 5]

Aby se dalo mluvit o optimálním nastavení regulátoru, je potřeba nejprve definovat, jak se nastavení regulátoru bude hodnotit. Zde uvedená kritéria jsou kritéria integrální, kvalitu určují podle odezvy systému v časové oblasti. V systému se vyvolá skoková (a jednotková) změna žádané veličiny a zjišťuje se odezva systému - např. simulací. Kritéria jsou vesměs ve tvaru

$$J = \int_0^\infty t^n \left| e\left(t\right) - e\left(\infty\right)\right|^m \mathrm{d}t \tag{2.3}$$

2.3.1 Kritérium lineární plochy

Též známé jako lineární integrální kritérium, je hodnota obsahu plochy vymezené regulační odchylkou a ustálenou regulační odchylkou. Základní verze kritéria (2.4) je vhodná pro nekmitavé (aperiodické) děje. Pokud by došlo ke kmitavému ději, je nutné použít místo rozdílu absolutní hodnotu rozdílu. Jako u všech integrálních kritérií platí, že má-li obvod astatismus nejméně prvního stupně, ustálená regulační odchylka $e\left(\infty\right) = 0$.

$$J_l = \int_0^\infty \left[e\left(t\right) - e\left(\infty\right)\right] \mathrm{d}t \tag{2.4}$$

2.3.2 Kritérium kvadratické plochy

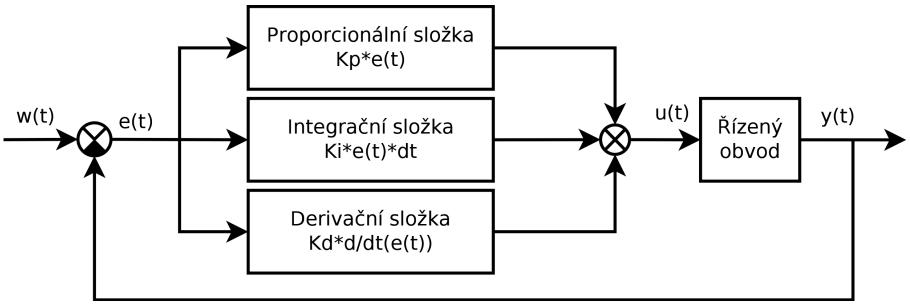
Kvadratické integrální kritérium je hodnota kvadrátu regulační plochy. Na rozdíl od lineárního kritéria zde již není třeba rozlišovat mezi kladnou a zápornou plochou, neboť její čtverec bude jistě kladný. Kritérium přikládá větší váhu větší odchylce, což při minimalizaci vede k řešením, která jsou rychle kmitající a s větším relativním překmitem. Dá se ale snadno vypočítat i bez simulace, pomocí aplikace Nekolného doplňku k Routh-Schurovu kritériu z přenosu odchylky obvodu.

$$J_s = \int_0^\infty \left[e\left(t\right) - e\left(\infty\right)\right]^2 \mathrm{d}t \tag{2.5}$$

2.3.3 Kritérium ITAE

Kritérium ITAE (z anglického Integral of Time multiplied by Absolute value of Error) odstraňuje nevýhody kvadratického kritéria (velkou kmitavost a překmit přechodového děje). Patří mezi váhová kritéria, odchylka je vážená časem, v němž se vyskytuje.

$$J_I = \int_0^\infty t \cdot \left|e\left(t\right) - e\left(\infty\right)\right| \mathrm{d}t \tag{2.6}$$



Obrázek 1: Schéma regulačního obvodu s PID regulátorem

2.3.4 Další nároky na průběh přechodového děje

V praxi se můžou vyskytovat další nároky na průběh přechodového děje, typicky požadavek na rychlejší reakci soustavy (což ovšem vede ke kmitavému ději s vyšším překmitem a delší dobou ustálení), nebo požadavek na maximální dovolený překmit či nekmitavý děj (který má výrazně pomalejší náběh a delší dobu ustálení).

2.4 PID regulátor

PID regulátor je v současné době nejpoužívanější regulátor v průmyslu. Nespornou výhodou je spolehlivost, která plyne z konstrukční jednoduchosti. Díky ní lze s regulátorem i velmi jednoduše pracovat. Skládá se ze tří složek - integrační, derivační a proporcionální (jak je znázorněno na obrázku 1).

PID regulátor má obecně přenos

$$G_{PID} = K_p + K_i \cdot \frac{1}{s} + K_d \cdot s = K_p \left(1 + \frac{1}{T_i s} + T_d s\right) \tag{2.7}$$

kde s je Laplaceova proměnná.

2.4.1 Algoritmus funkce PID regulátoru

Dnes vyráběné regulátory jsou postaveny na mikroprocesorech, což samozřejmě znamená, že ze své podstaty nemohou být nikdy integrační (spojité), a tudíž by se mělo mluvit o PSD regulátorech (proporcionálně-sumačně-diferenčních). Pokud je ale vzorkovací perioda vůči času přechodového děje dostatečně (alespoň 1000×) kratší, můžeme nespojitost zanedbat. Zde uvedený algoritmus (napsán podle [6]) je psán v pseudokódu.

Program 1: Algoritmus funkce PID/PSD regulátoru

```
predchozi_chyba = 0
integral = 0
while sleep(dt):
        chyba    = zadana_hodnota - regulovana_vel
        P_slozka = Kp*chyba
        integral = integral + chyba*dt
        I_slozka = Ki*integral
        derivace = (chyba - predchozi_chyba)/dt
        D_slozka = Kd*derivace
        vystup   = P_slozka + I_slozka + D_slozka
        predchozi_chyba = chyba
end while
```

2.4.2 Popis složek PID regulátoru

P složka

Reakce proporcionální složky je přímo úměrná regulační odchylce. Sama o sobě nedosáhne nulové ustálené regulační odchylky. Laděním P složky lze dosáhnout jen snížení ustálené regulační odchylkky, ovšem za cenu vzrůstu maximálního překmitu.

I složka

Použití integrační složky regulátoru má za následek dosažení nulové regulační odchylky v ustáleném stavu, ale její zařazení zvyšuje kmitavost děje, zejména překmit.

D složka

Derivační složka regulátoru kompenzuje růst kmitavosti děje integrační složky. Reaguje na rychlost změny chyby. Ideální D složka (ovšem z fyzikální podstaty nerealizovatelná) by předvídala budoucí vývoj soustavy.

2.4.3 Konvenční metody nastavení - metoda Ziegler-Nichols

[9, 10]

Metoda Ziegler-Nichols je jednoduchá metoda nastavení parametrů PID regulátoru. Spočívá v přivedení obvodu na hranu stability zvyšováním proporcionální složky regulátoru (I a D složky jsou vyřazeny). Zjistí se hodnoty K_C kritického zesílení a T_C kritické periody kmitů. Poté se použijí empirické vztahy pro nastavení P, I a D složky regulátoru. Vztahů pro výpočet parametrů metodou Ziegler-Nichols existuje samozřejmě větší množství, podle toho, která charakteristika přechodového děje je pro nás v dané chvíli nejdůležitější. Parametry uvedené v tabulce 2 jsou tzv. „klasické“.

Problémem metody Ziegler-Nichols (a podobných) jsou soustavy principiálně stabilní (stabilní pro všechny hodnoty zesílení K). Takové soustavy nelze z principu přivést na hranici stability a výpočet metodou Ziegler-Nichols pro ně zcela selhává. Pro takové soustavy (nebo na doladění parametrů podle jiné metody, pokud je potřeba nějakou charakteristiku upřednostnit) lze použít nastavení regulátoru empirické, podle tabulky 1.

Tabulka 1: Vliv změny parametrů na přechodový děj PID regulátoru

[7, 8]

Složka	Změna	Čas náběhu	Překmit	Čas ustálení	Ustálená odchylka	Stabilita
K_p	zvýšení	klesá	roste	malý vliv	klesá	zhoršuje se
	snížení	roste	klesá		roste	zlepšuje se
K_i	zvýšení	klesá	roste	roste	nulová	zhoršuje se
	snížení	roste	klesá	klesá		zlepšuje se
K_d	zvýšení	malý vliv	klesá	klesá	bez vlivu	zlepšení při malých K_d
	snížení		roste	roste		zhoršuje se

Tabulka 2: Nastavení parametů metodou Ziegler-Nichols

Regulátor	K_p	T_i	T_d
P	$0.50 \cdot K_c$	-	-
PI	$0.45 \cdot K_c$	$\frac{T_c}{1.2}$	-
PD	$0.80 \cdot K_c$	-	$\frac{T_c}{8}$
PID	$0.60 \cdot K_c$	$\frac{T_c}{2}$	$\frac{T_c}{8}$

3 Evoluční výpočetní techniky

Mezi evoluční výpočetní techniky patří genetické algoritmy, kterým bude věnována hlavní část této kapitoly, dále genetické programování, evoluční strategie, evoluční programování a další. Z evolučních metod se některé hodí na problém hledání extrému obecné funkce, jiné řeší NP-těžké/NP-úplné problémy (například problém obchodního cestujícího - optimalizace mravenčí kolonií[2]), případě evolucí vytvářejí celou strukturu programu (genetické programování), na rozdíl od metod genetického algoritmu, které hledají číselné hodnoty proměnných v jediném nastaveném programu. Mimo genetických algoritmů zde budou popsány ještě evoluční strategie a metoda diferenciální evoluce.

3.1 Popis obecného genetického algoritmu

[11, 12]

První publikace o genetických algoritmech pochází z roku 1975 (Adaptation in Natural and Artificial Systems, autor John Holland). Jedná se o přístup s inspirací v přírodě - Darwinova evoluce řeší problém za pomoci mnoha jedinců, kteří podléhají optimalizaci a tlaku prostředí.

Algoritmus (vývojový diagram na obr. 2) začíná vytvořením prvotní generace. Poté jedince ohodnotí (kriteriální funkcí, 3.1.6) a vybere jedince pro křížení (podrobněji v 3.1.4). Nové jedince (potomky) vzniklé křížením případně podrobí mutaci. Následně vzniká nová generace z potomků a případně i rodičů. Tento cyklus se opakuje, dokud není splněna podmínka ukončení - může jí být například počet generací, změna nejlepšího fitness, dosažení požadované kvality řešení nebo jiná podmínka. Nejlepší jedinec je algoritmem považován za řešení problému.

3.1.1 Genom

Genom je souborem vlastností jedince, reprezentací vlastností řešení pro počítačový algoritmus. Genomy se kódují zpravidla jako řetězce - binární, číselné, nebo textové. Podle možností jazyka lze ale genomy ukládat i jinak - například jako prvky pole.

3.1.2 Populace

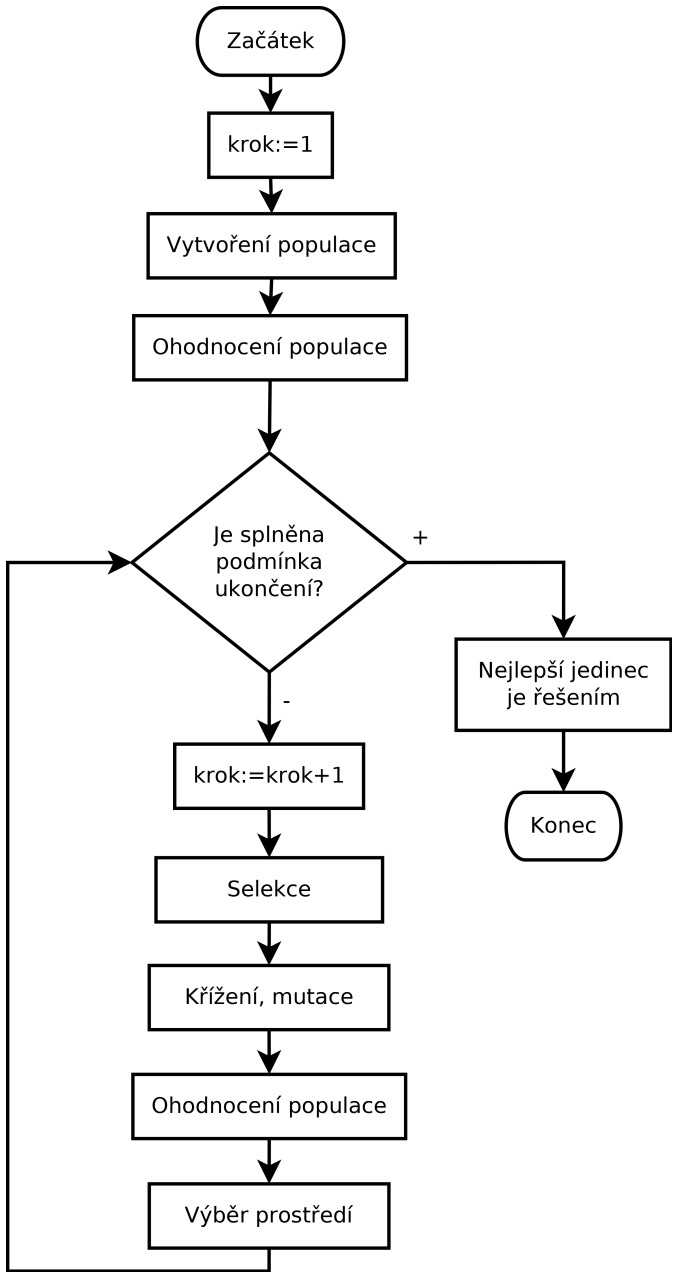
Populací rozumíme soubor všech jedinců v generaci. Výsledek genetického algoritmu může ovlivnit volba generace nula. Nevhodné nastavení (př. na obr. 3b) generace nula a zároveň použití nevhodně nastaveného evolučního algoritmu může vést k neoptimálním výsledkům (typicky uvíznutí v lokálním extrému funkce). Toho se lze vyvarovat rozšířením generace nula na širší interval (př. na obr. 3a) nebo zvýšením pravděpodobnosti mutace.

Samostatnou kapitolou je výběr v populaci pro křížení (selekce). Existuje mnoho přístupů - zde uvedený výčet metod selekce není úplný.

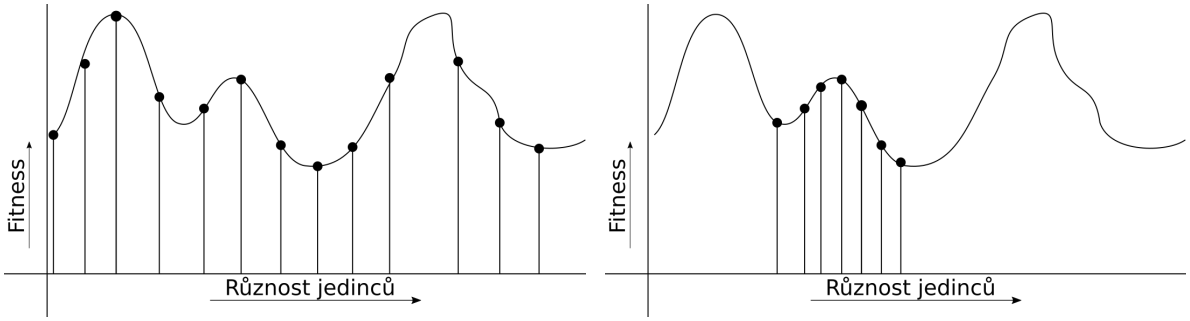
3.1.3 Metody výběru

Zvolení vhodné metody výběru může výrazně snížit nároky na samotný výpočet - sníží se počet generací i velikost populace, nutné k dosažení dobrého výsledku.

[2] https://cs.wikipedia.org/wiki/Optimalizace_mraven%C4%8D%C3%AD_koloni%C3%AD



Obrázek 2: Obecné schéma genetického algoritmu



(a) Vhodné (b) Nevhodné

Obrázek 3: Zvolení generace nula

Uniformní metoda

Všichni jedinci z dané generace mají stejnou šanci na výběr ke křížení. Tato metoda výběru je nejjednodušší, ale nejméně odolná proti náhodným jevům. Bez dalších úprav může snadno dojít ke ztrátě lepších jedinců a zhoršení kvality celé populace. Opětovné navrácení na původní kvalitu může trvat i mnoho generací - při tom může dojít k zastavení procesu evoluce podmínkou ukončení (například maximálním přípustným počtem generací nebo strojového času).

Ruletová metoda

Každý jedinec má šanci na výběr ke křížení úměrnou vlastnímu fitness (obr. 4a). Silnější jedinci jsou vybíráni častěji, ale výběr snadno umožňuje i postup slabších jedinců. Tento způsob omezuje (ačkoliv zcela neodstraňuje) možnost ztráty nejlepších řešení, metoda jen spoléhá na to, že ztráta velkého množství dobrých jedinců je nepravděpodobná.

Stochastická univerzální metoda

V případě velkého množství jedinců naráží ruletová metoda výběru na potřebu generování velkého množství náhodných čísel. I když problém nedostatečného množství entropie není kritický (není potřeba mít kryptograficky bezpečná náhodná čísla), výrazná pravidelnost je evolučnímu mechanismu na závadu, nehledě na to, že generovat velké množství náhodných čísel může být časově náročné. Stochastická univerzální metoda tento nedostatek odstraňuje tak, že místo generování náhodného čísla pro výběr každého jedince zvlášť se vygeneruje jen jedno (pro výběr prvního jedince) a ostatní jedinci jsou vybíráni v konstantních krocích (obr. 4b). Stejně jako ruletová metoda zcela neodstraňuje možnost ztráty nejlepších jedinců vlivem náhody.

Turnajový výběr

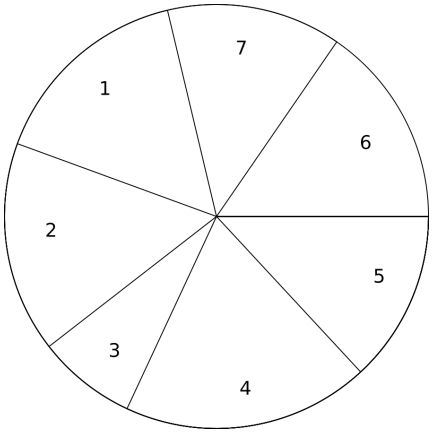
Turnajový výběr kombinuje výhody uniformního výběru (diverzita populace) a výběru seříznutím (selekční tlak, zachovávání lepší části populace). V prvním kroku výběru jsou náhodně vybíráni a seskupováni jedinci z původní generace. Tito jedinci jsou potom seřazeni podle své fitness a nejlepší jedinec („vítěz turnaje“) postupuje ke křížení. Sestavuje se mnoho skupin a díky tomu má každý jedinec větší šanci mít ve své skupině horší jedince, které dokáže porazit. Tato metoda je v současnosti nejpoužívanější.

Výběr seříznutím

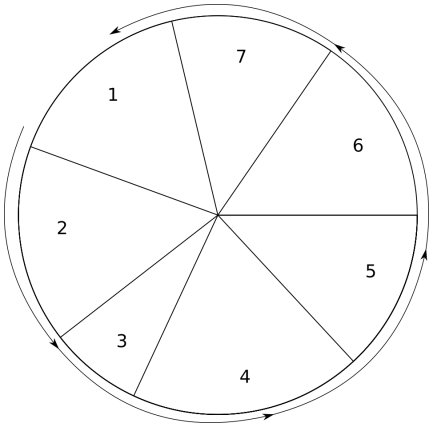
Výběr seříznutím se provádí seřazením všech jedinců podle jejich fitness a odříznutím méně výkonné části generace. Ze zbylých jedinců se dalšími metodami vytvářejí potomci. Tato metoda je náchylná na uvíznutí v lokálním extrému - může dojít k tomu, že globální extrém kriteriální funkce se nachází na malé ploše vstupních parametrů, zatímco některý z lokálních extrémů (s nižší hodnotou fitness) pokrývá daleko větší plochu. Odříznutím získáme většinu jedinců v lokálním extrému, kteří během několika generací můžou v generaci zcela zavládnout (jak je ukázáno na obr. 4c).

Elitismus

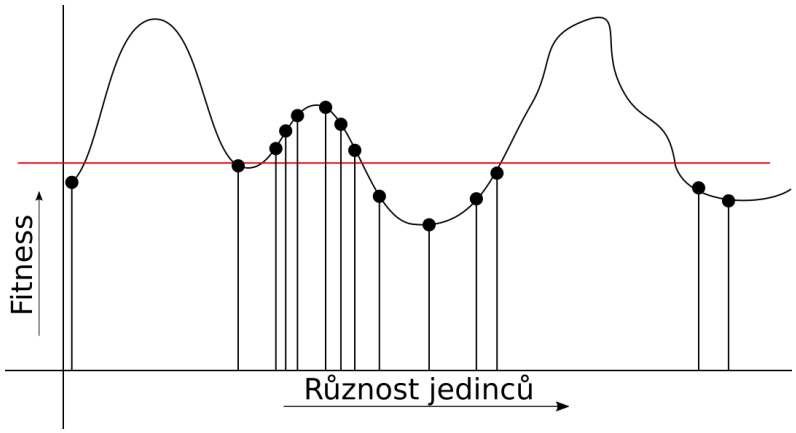
Elitismus je rozšířením obecné výběrové metody. Zajišťuje postup určitého počtu nejlepších jedinců do nové generace také přímo, bez nutnosti podléhat další výběrové metodě.



(a) Ruletová metoda



(b) Stochastická univerzální metoda



(c) Výběr seříznutím - uváznutí v lokálním extrému

Obrázek 4: Selekce

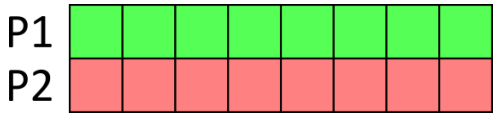
Díky danému přenosu nejlepšího jedince do nové generace nemůže dojít k zhoršování výsledku.

3.1.4 Křížení

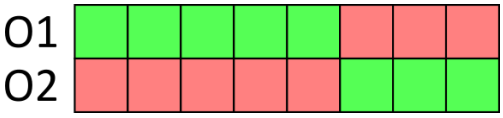
Jedinci vybraní selekcí (rodiče) křížením vyprodukují potomky. Genomy rodičů se v určitém místě (místech) přeruší a vymění. Podle počtu přerušení genomů mluvíme o n-bodovém (jednobodovém - obrázek 5b, dvoubodovém - obrázek 5c atp.) křížení. Zkřížením genomů vznikají potomci. Nejobecnější metodou křížení je metoda uniformního křížení (obrázek 5d). Pro křížení se náhodně získá binární číslo s tolika bity, jako je znaků v genomu. Pokud je na místě daného genomu v masce jednička, potomek získává vlastnost od prvního rodiče, pokud nula, od druhého. S určitou malou pravděpodobností ke křížení nemusí dojít - rodič a potomek jsou pak totožní, a říkáme, že došlo k přenosu (migraci) rodiče do nové generace.

3.1.5 Mutace

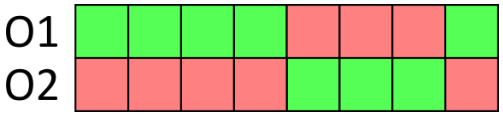
Mutace je metoda, díky které je genetický algoritmus odolný vůči uvíznutí v lokálním extrému funkce. S určitou (malou) pravděpodobností se genom jedince pozmění. Používané metody mutace jsou zejména náhodné (hodnota vlastnosti je nahrazena náhodnou hodno-



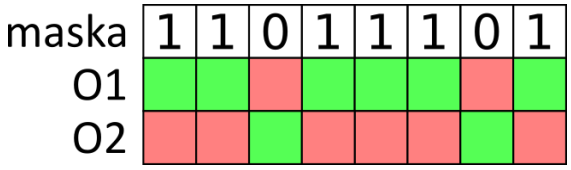
(a) Rodiče



(b) Jednobodově zkřížení potomci



(c) Dvoubodově zkřížení potomci



(d) Uniformně zkřížení potomci

Obrázek 5: Křížení

Obrázek 6: Mutace

tou), aditivní (k hodnotě vlastnosti je přičteno/odečteno náhodné číslo) a multiplikativní (hodnota vlastnosti se vynásobí/vydělí náhodným číslem).

Pravděpodobnost mutace může být jak stálá (typicky okolo 2-5%), tak proměnná, kdy algoritmus sám hodnotí, jestli jsou jedinci v dané generaci příliš blízko sebe, a případně hodnotu zvýší. Díky vyšší hodnotě mutace získáváme odlišnější jedince, populace může snáze vyváznout z lokálního extrému. Po rozvinutí jedinců po větší ploše kriteriální funkce se pravděpodobnost mutace opět sníží.

3.1.6 Kriteriální (fitness) funkce

Volba kriteriální funkce je pro úspěch genetického algoritmu kritická. Špatně zvolená fitness funkce způsobí totální selhání. Fitness funkce přiřazuje každému jedinci číselné ohodnocení kvality jeho řešení. Na kriteriální funkci nejsou kladeny větší nároky než je existence (a nezápornost) hodnoty pro validní vstupy. Kriteriální funkce nemusí být dokonce ani spojitá, což je oproti jiným optimalizačním metodám velká výhoda.

V případě optimalizace PID regulátoru je to hodnota některého z kritérií kvality regulace (2.3). Pokud se vyskytují zvláštní požadavky na průběh přechodového děje, je nutné je v kriteriální funkci správně implementovat a nevyhovujícím jedincům přiřadit odpovídající hodnotu fitness (řádově vyšší než dosahují vyhovující jedinci).

3.2 Evoluční strategie

[13, 14]

Evoluční strategie je z implementačního hlediska velice podobná genetickým algoritmům, ale z biologického hlediska nacházíme podstatný rozdíl: genetické algoritmy berou všechny jedince jako stejný, jediný druh, a tedy umožňují řešení křížit, zatímco evoluční strategie pohlíží na jedince jako na jediné představitele daného druhu (každý jedinec je druh sám pro sebe) a proto operátor křížení nepoužívá. Náhodnou mutací vybraného rodiče vzniká potomek, který se svým rodičem soutěží o postup do další generace - typicky postupuje rodič, pokud je lepší než potomek.

3.3 Diferenciální evoluce

[13, 15]

Diferenciální evoluce pracuje pouze s operátorem křížení. Kromě pravděpodobnosti křížení CR zavádí také novou veličinu - váhu rozdílu F. Potomek P vzniká tak, že k jeho vzniku jsou zapotřebí čtyři různí jedinci - označme je A, B, C, D. Nejprve se vypočte náhodný jedinec $N_i = B_i + F\,(C_i - D_i)$, který je následně křížen s rodičem A. Poté se potomek testuje vůči rodiči A - pokud je horší než rodič, zůstává v generaci rodič, pokud ne, zaujme potomek P místo rodiče A. Jak je patrné z předpisu pro výpočet hodnoty genu potomka, diferenciální evoluce dostala své jméno z diference hodnot genu, nikoliv podle diferenciálního počtu.

3.4 Vhodnost použití evolučních algoritmů

Evoluční výpočetní techniky jsou vhodné zejména pro optimalizační úlohy, rozhodování a klasifikace. Jejich nespornou výhodou je fakt, že není zapotřebí znát celý matematický popis systému, dostačující je informace o podobě výsledku (jaké řešení je přípustné) a způsobu, jak výsledky hodnotit (kriteriální funkce). Jsou odolné proti nedokonalostem „klasického“ řešení - zejména uváznutí v lokálním extrému.

Problémy genetických algoritmů jsou zejména tyto:

- nedeterminičnost (několik provedení algoritmu může vyprodukovat různé výsledky)
- neschopnost obecně nalézt optimální řešení (dostaneme pouze řešení blízké optimálnímu)
- časová náročnost výpočtu (v závislosti na složitosti kriteriální funkce)

Dále jsou velice citlivé na nastavení počáteční populace, míry mutace, rozdílné výsledky budeme samozřejmě dostávat při použití různých selekčních, křížících a mutačních metod.

4 Optimalizace PID regulátoru pomocí evolučních metod

Všechny metody počítají s přenosem regulátoru ve tvaru

$$G_{PID} = P + I \cdot \frac{1}{s} + D \cdot \frac{N}{1 + \frac{N}{s}} \tag{4.1}$$

s parametrem filtrace N=100.

4.1 Kriteriální funkce

Jako základní kriteriální funkci bylo zvoleno kritérium ITAE (2.3.3), případnou volbu jiného kritéria je možné provést v nastavení při spuštění metody. Je možné volit mezi základními třemi kritérii uvedenými v kapitole 2.3. Dále je uvedeno čtvrté kritérium - modifikace ITAE, která za nevyhovující označí jedince s určeným překmitem.

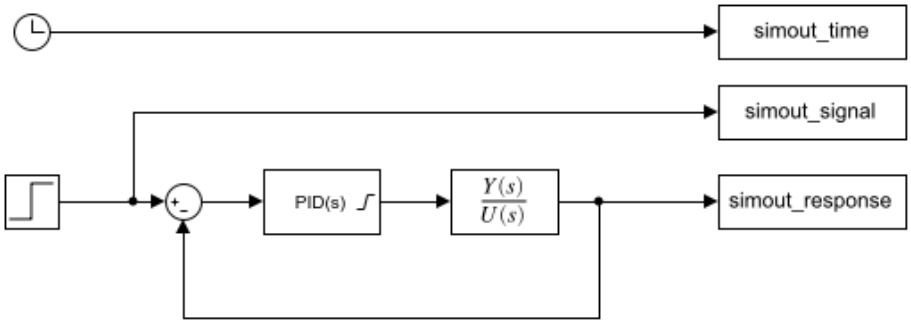
4.2 Popis algoritmů implementovaných v MATLABu

Pro vlastní implementaci v MATLABu byla zvolena jednodušší varianta uvedených metod. Datová struktura populace je velice primitivní - jedná se o pole $4 \times$POPULATION_SIZE, v prvním sloupci je uložena hodnota P složky regulátoru, ve druhém I složky, ve třetím D složky. Čtvrtý sloupec je připraven pro zápis hodnot fitness.

Nejprve je vytvořena výchozí populace. Nastaví se výchozí hodnoty pro cyklus a cyklus se spustí. Provede se ohodnocení populace dle vybraného kritéria, nejlepší jedinec se uloží do pole nejlepších řešení jednotlivých generací (datová struktura je totožná). Provede se úprava populace dle dané metody. Poté se vypíše nejlepší řešení z generace a dojde ke zkontrolování podmínky ukončení (zde pro jednoduchost jen počet generací).

Po poslední generaci algoritmus znovu vyhodnotí populaci a uloží nejlepšího jedince do vyhrazeného pole. Za řešení se považuje nejepší jedinec z pole nejlepších jedinců. Jsou vykresleny grafy průběhu nejlepšího fitness a odezva obvodu na jednotkový skok řídicí veličiny.

Odezva obvodu se zjišťuje pomocí modelu v SIMULINKu (obrázek 7). Celý model je plně parametrický, přebírá svoje nastavení (maximální čas simulace, parametry regulátoru i regulované soustavy) z proměnných v MATLABu.



Obrázek 7: Zjištění odezvy systému - model SIMULINKu

4.2.1 Genetický algoritmus

Pro výpočet metodou genetického algoritmu je použito uniformního výběru s elitismem. Mutace je provedena aditivní, její pravděpodobnost i mutační skok mají proměnou hodnotu, která se zvyšuje, pokud určenou dobu nedošlo ke zlepšení nejlepšího jedince (tím se pomáhá k případnému vymanění se z lokálního extrému funkce).

4.2.2 Evoluční strategie

Hlavní smyčka je podobná jako v případě genetického algoritmu. Míra mutace zde zůstává konstantní (výchozí nastavení je na 98%), mění se jen mutační skok. V každé iteraci algoritmu se populace podrobí mutaci a výběr probíhá nad oběma populacemi - do další generace postupuje ten z dvojice rodič-jemu odpovídající potomek, který má lepší hodnotu fitness.

4.2.3 Diferenciální evoluce

Pro výběr je použito metody uniformní, tedy že každý jedinec má stejnou šanci být vybrán. V metodě diferenciální evoluce bylo zapotřebí dořešit ještě jeden implementační detail, jak se bude zacházet s jedincem, který některým znakem překročí nastavené hranice. Tyto hranice lze považovat jen za úvodní nastavení, a dál počítat bez omezování vstupních hodnot, to by ale vedlo k rychlému „rozprchnutí“ jedinců na všechny strany a algoritmus by nedovedl dobře konvergovat. Proto přicházejí v úvahu dvě metody - buď jedince „zastavit“ na hranici parametrů, nebo nahrazení příslušného znaku jinou hodnotou. Protože umístění jedince na hranici je náchylnější na uvíznutí v lokálním extrému, byla zvolena možnost nahrazení jinou, v tomto případě náhodnou hodnotou.

4.2.4 Použití funkce ga() v MATLABu

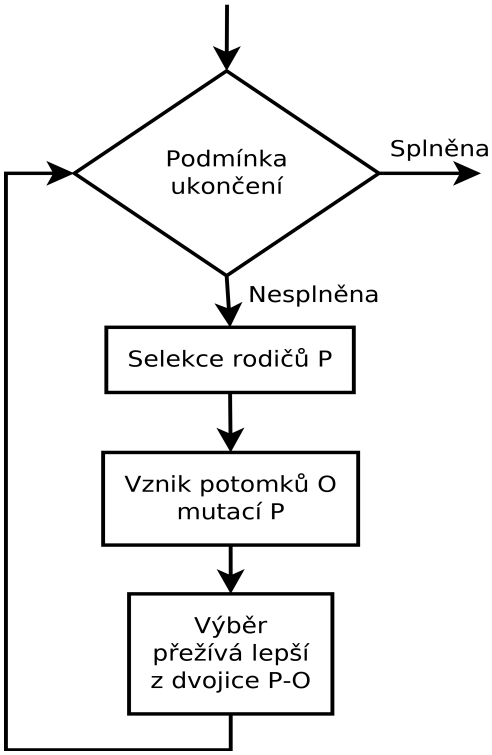
[16]

MATLABovská funkce `ga()` přijímá argumenty v několika formách, pro snadnější čitelnost byla zvolena forma jediného argumentu `problem` typu `struct`. V problému se nastaví zejména kriteriální funkce a počet rozměrů problému. Ostatní nastavení je nepovinné.

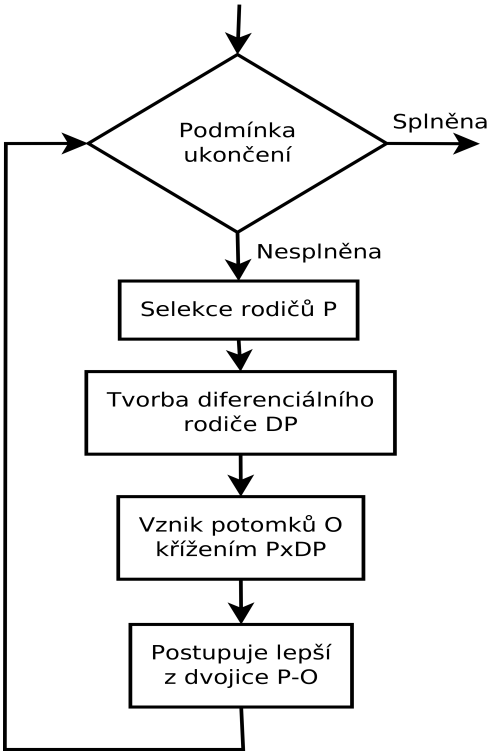
Algoritmus může svojí činnost ukončit z několika důvodů. Těmi jsou mimo jiné vyčerpání maximálního počtu generací, počet generací, v nichž se řešení nelepší, nebo pokud je změna průměrného fitness pod určitou tolerancí. Funkci `ga()` volá obslužná funkce, která má za úkol jednak zajistit potřebné nastavení, jednak poskytnout navenek stejné rozhraní jako ostatní implementované algoritmy.

4.3 Další optimalizace výsledků

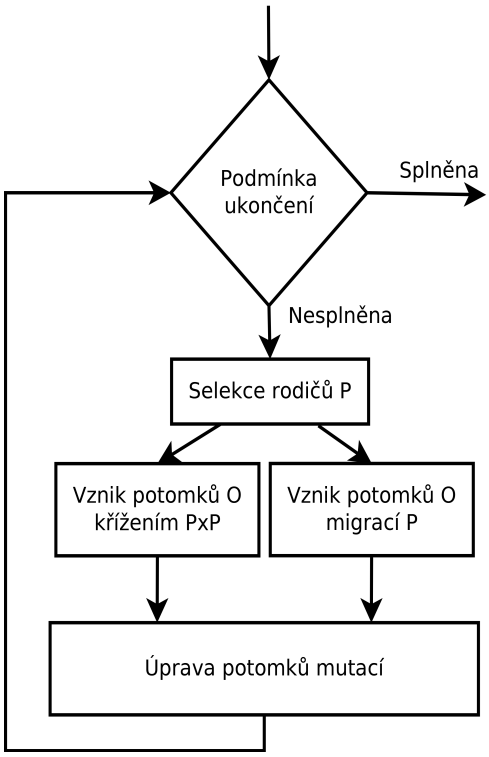
Evoluční metody díky svému náhodnostnímu charakteru obecně nenalézají přesné místo extrému funkce, jen bod v jeho okolí. Proto se může dosáhnout dalšího zlepšení, pokud tento nalezený bod použijeme jako výchozí pro optimalizaci gradientní metodou.



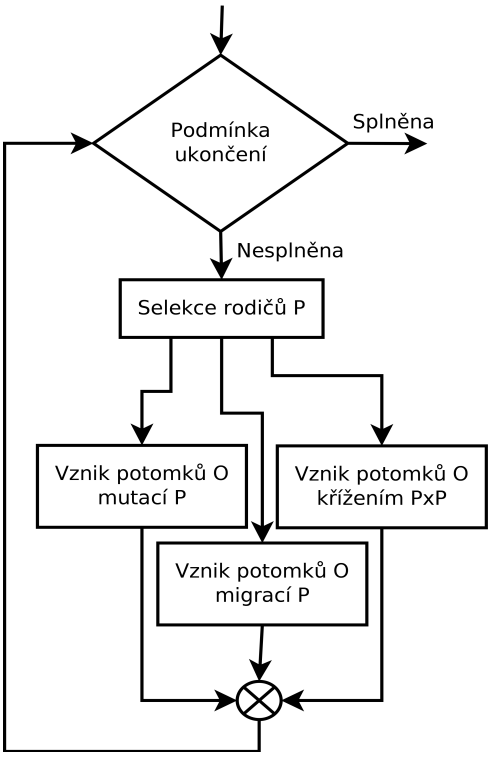
(a) Evoluční strategie



(b) Diferenciální evoluce



(c) Genetický algoritmus



(d) `ga()`

Obrázek 8: Vývojové diagramy použitých metod

5 Srovnání evolučních a konvenčních metod

5.1 První soustava

Mějme soustavu s přenosem

$$G_S = \frac{s+2}{s^3+4s^2+4s+8} \tag{5.1}$$

Jak je patrno z výsledku volání `get_ZN_params()` s touto soustavou (alg. 2), kritické zesílení je nekonečně velké, soustava je tedy principiálně stabilní. Ziegler-Nicholsova metoda zcela selhává.

Přechodové děje s regulátory získanými pomocí evolučních výpočetních technik jsou vyobrazeny na grafech 9a, jednotlivé parametry přechodové charakteristiky v tab. 3, graficky na obrázku 9b. Jednotlivé přechodové děje a další obrázky jsou dostupné v přílohách (příloha A).

Další optimalizací (např. metodou `fminsearch` v MATLABu) lze získat další významné zlepšení fitness vypočtených regulátorů[3], jak lze vidět při srovnání tab. 3 a 4. V blízkém okolí bodu vypočteného evolučními algoritmy bude kriteriální funkce se značnou pravděpodobností splňovat všechny podmínky pro funkci konvenčních algoritmů pro hledání extrému funkce. Samotná optimalizace výsledků genetických algoritmů pak znamená řádově menší počet vyhodnocení kriteriální funkce než celý průběh genetického algoritmu. Srovnání přechodových charakteristik optimalizovaných regulátorů je na obr. 10a. Nejlepšího výsledku v tomto případě dosáhla metoda evoluční strategie, ale v závislosti na provedení regulačního obvodu by mohlo být účelné použít výsledek genetického algoritmu (fitness metody GA je sice oproti ES o polovinu vyšší, získáme ale charakteristiku bez výrazného překmitu).

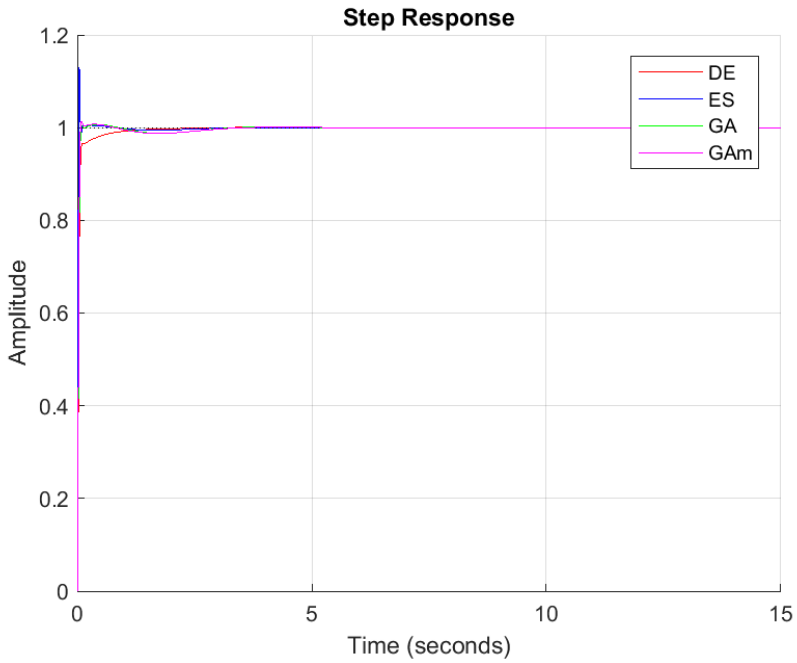
Tabulka 3: Řešení první soustavy - neoptimalizované výsledky

Method	P	I	D	Fitness	Overshoot [%]	Rise Time [s]	Settling Time [s]
DE	16.6779	74.2603	32.906	0.055041	0.073072	0.052597	0.44072
ES	173.1016	230.2709	81.53	0.020064	12.9727	0.020278	0.062677
GA	60.2682	96.0295	35.1673	0.065501	0.65547	0.044042	0.067898
GAm	71.7684	96.9664	37.3868	0.063932	1.3428	0.040459	0.060688
ZN	Inf	Inf	NaN	Inf	NaN	NaN	NaN

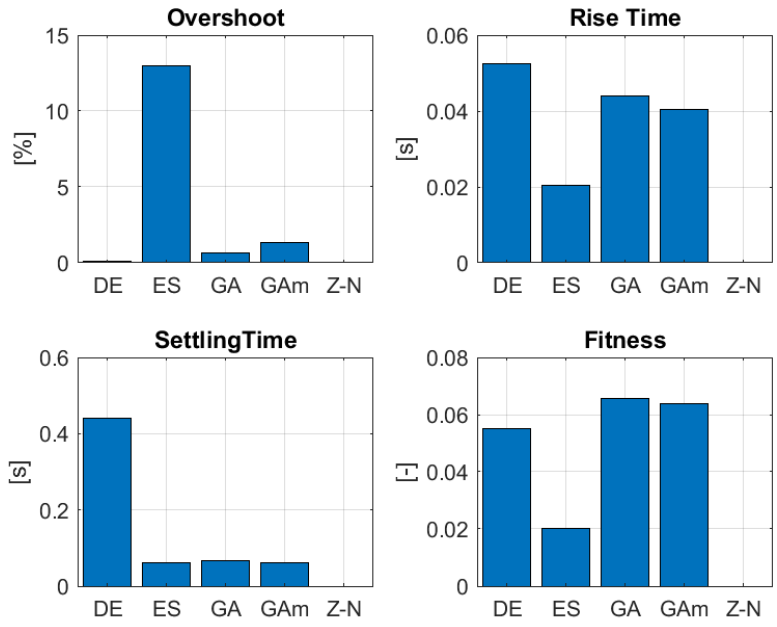
Tabulka 4: Řešení první soustavy - optimalizované výsledky

Method	P	I	D	Fitness	Overshoot [%]	Rise Time [s]	Settling Time [s]
DE	15.2466	74.4768	32.5406	0.033721	0.0011162	0.052578	0.46413
ES	173.1016	241.7844	81.53	0.017727	12.9781	0.020278	0.062677
GA	60.3615	100.8392	35.172	0.065501	0.76056	0.044042	0.067898
GAm	67.3589	107.7001	37.4067	0.059738	1.1011	0.040407	0.062446

[3]Typicky asi 25%, zlepšení se pohybovalo od cca 10% do cca 50%

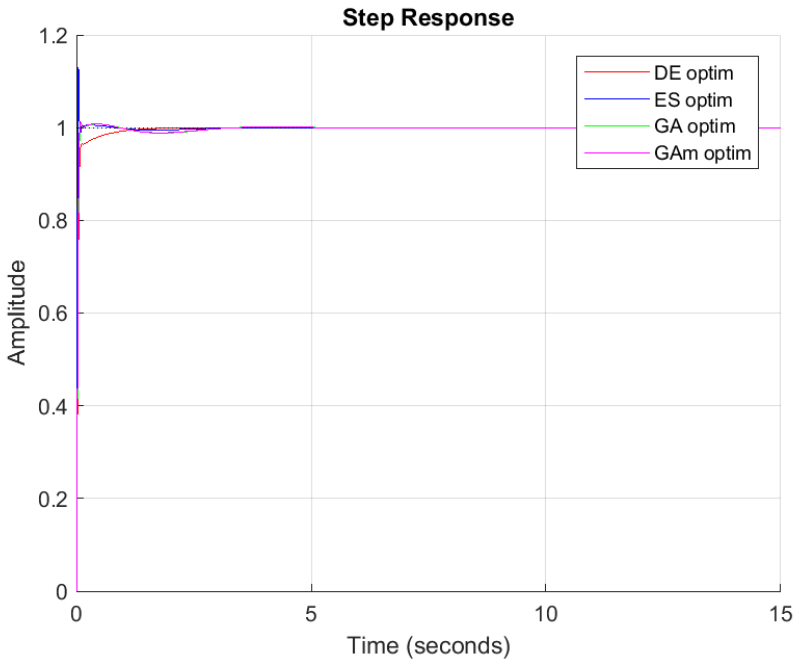


(a) Přechodové děje

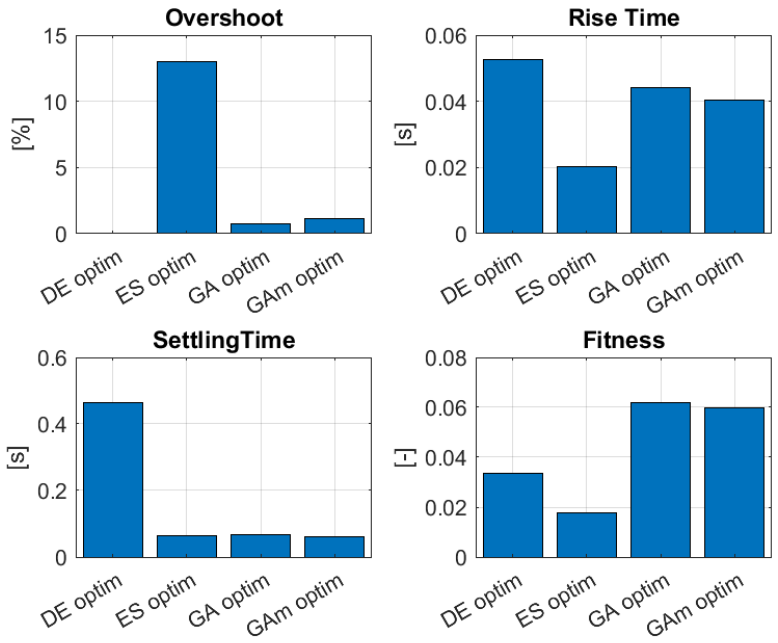


(b) Srovnání charakteristik

Obrázek 9: Řešení první soustavy - výsledky evolučních algoritmů



(a) Přechodové děje



(b) Srovnání charakteristik

Obrázek 10: Řešení první soustavy - výsledky po optimalizaci

5.2 Druhá soustava

Mějme soustavu s přenosem

$$G_S = \frac{6}{48s^3 + 44s^2 + 12s + 1} \tag{5.2}$$

Na tuto soustavu již Ziegler-Nicholsovu metodu aplikovat lze a podle předpisu tab. 2 získáváme regulátor s parametry P=0.9833, I=0.2653, D=2.5133. Soustava s tímto regulátorem má čas ustálení asi 30 sekund a překmit 40%. Oproti tomu EVT dávají řešení s polovičním až třetinovým časem ustálení a překmitem do 80%, jak je vidět v tab.5. Jak je vidět na obr. 11a, EVT řešení mají 3-5 viditelných kmitů, což by se i pro mechanické akční členy mohlo považovat za přijatelné. Jak je vidět ze srovnání tab. 5 a 6, jediné viditelné zlepšení proběhlo u řešení funkce `ga()`, což lze ale snadno vysvětlit brzkým splněním podmínky ukončení. Můžeme proto usuzovat, že příslušná kriteriální funkce je složitá a s velkým množstvím extrémů.

V tomto případě dosahuje nejlepších výsledků metoda diferenciální evoluce, nízká hodnota fitness, tři viditelné kmity a nejnižší překmit ze všech EVT metod znamenají, že tento regulátor je zcela přijatelný i pro soustavy s mechanickým akčním členem (kde by v případě více kmitavého řešení došlo k jeho dřívějšímu opotřebení).

Tabulka 5: Řešení druhé soustavy - neoptimalizované výsledky

Method	P	I	D	Fitness	Overshoot [%]	Rise Time [s]	Settling Time [s]
DE	5.6123	0.70026	23.5619	6.3233	51.1205	0.84677	11.2903
ES	15.1799	2.1729	97.0001	6.7417	73.3505	0.35944	11.8614
GA	4.4706	0.41189	19.9419	6.3152	46.556	0.54902	10.4314
GAm	13.3725	30.0698	65.0785	15.8616	79.0387	0.33816	16.0064
ZN	0.98333	0.26526	2.5133	46.5403	41.7527	2.4138	29.7701

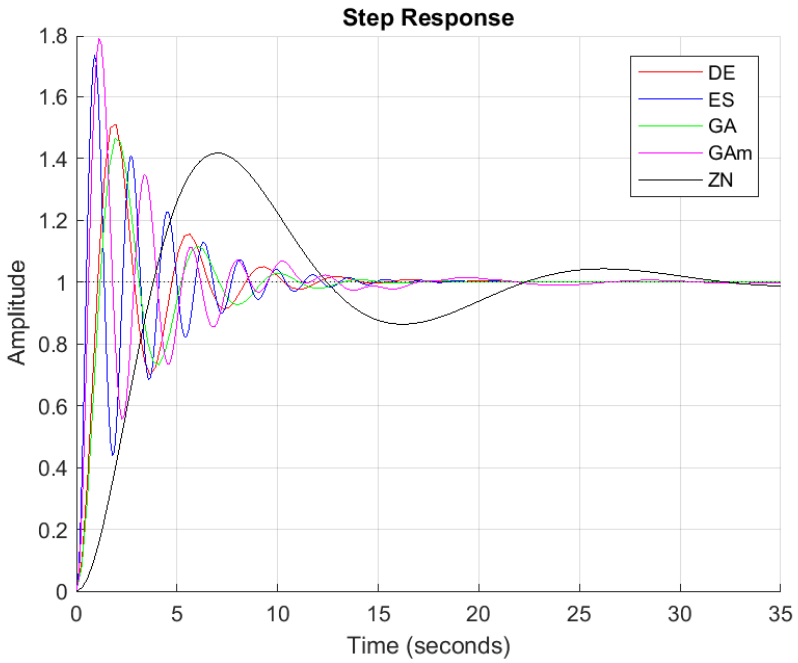
Tabulka 6: Řešení druhé soustavy - optimalizované výsledky

Method	P	I	D	Fitness	Overshoot [%]	Rise Time [s]	Settling Time [s]
DE	5.6114	0.69806	23.636	6.2823	51.2043	0.84337	11.245
ES	15.1799	2.2816	97.0001	6.7393	73.3671	0.35944	11.8614
GA	4.5708	0.4063	20.4787	6.3152	47.2992	0.54902	12.0784
GAm	14.4868	28.0651	61.8246	9.4542	79.8107	0.34711	14.4628

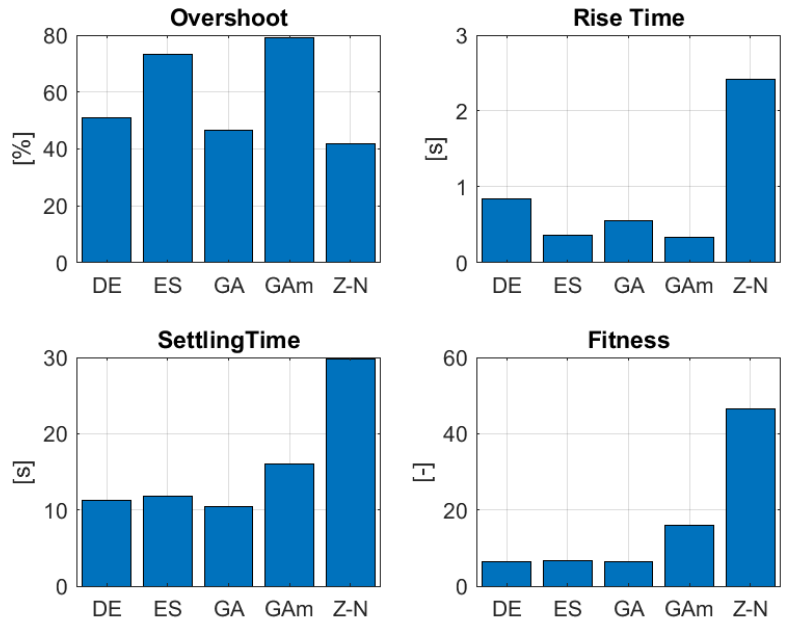
5.2.1 Statistické vyhodnocení

Proveďme na této soustavě statistické vyhodnocení čtyř použitých metod. Každý algoritmus byl na uvedené soustavě spuštěn $40\times$[4]. Jak je vidět z uvedených histogramů (obrázek 13), je pravděpodobnější, že EVT algoritmus najde řešení spíš blíže reálnému extrému.

[4]Pro vypovídající závěry by se slušelo získat větší statistický soubor. Velikost souboru byla nicméně limitována výpočetní a časovou náročností.

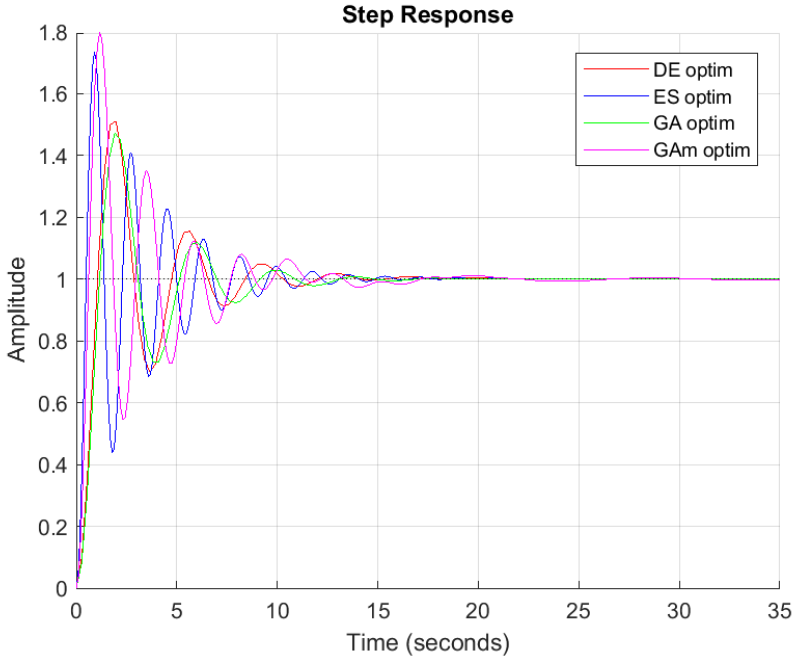


(a) Přechodové děje

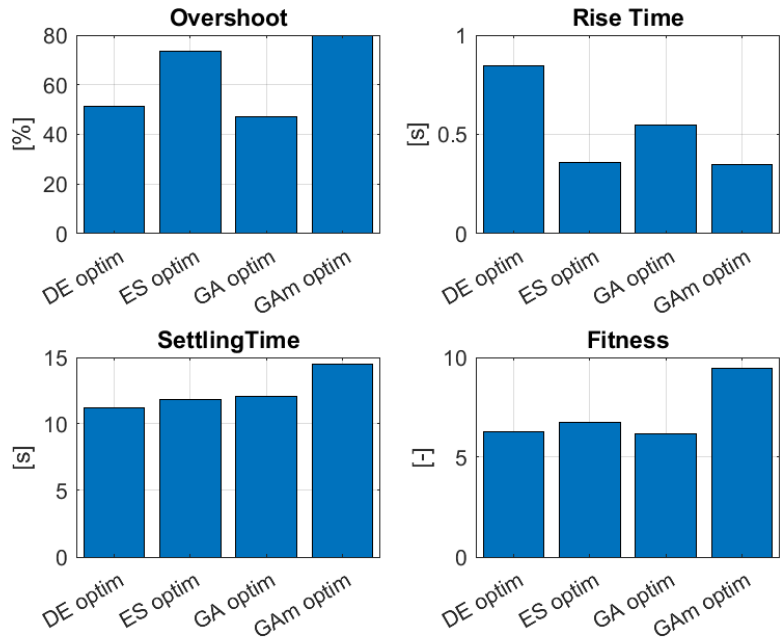


(b) Srovnání charakteristik

Obrázek 11: Řešení druhé soustavy - výsledky evolučních algoritmů

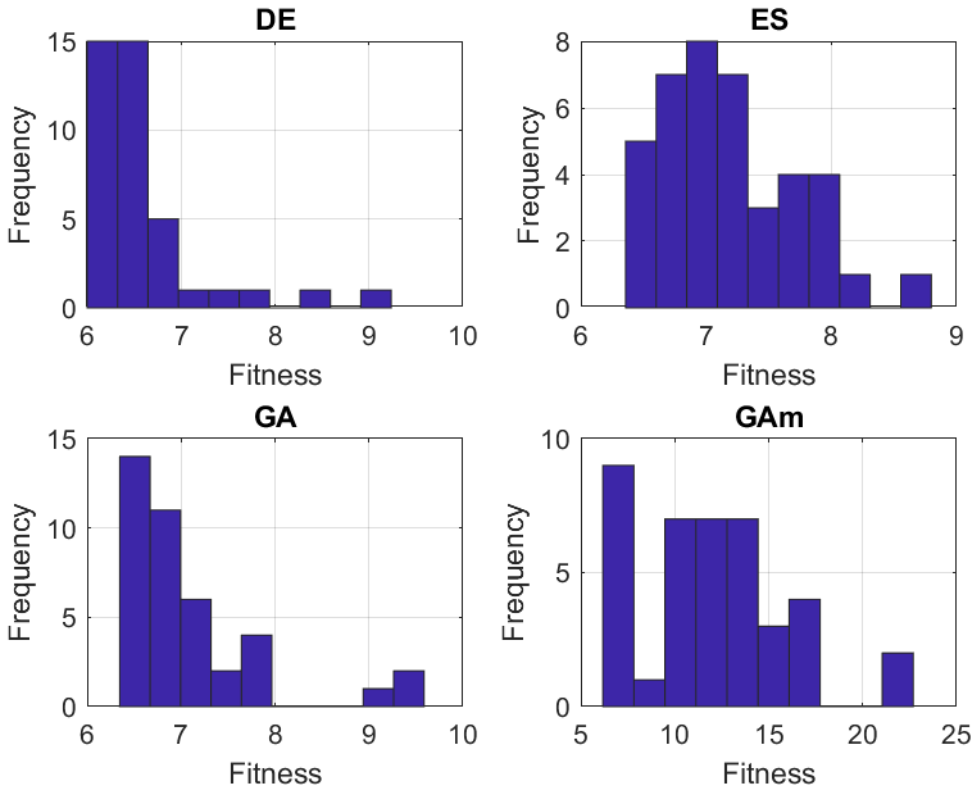


(a) Přechodové děje



(b) Srovnání charakteristik

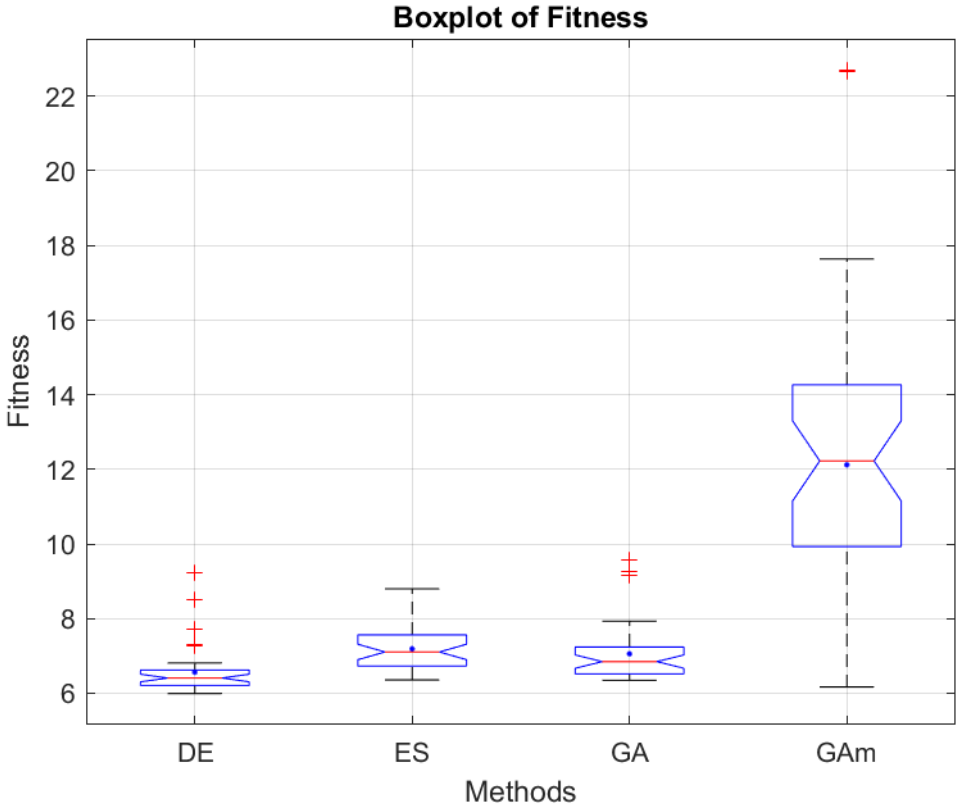
Obrázek 12: Řešení druhé soustavy - výsledky po optimalizaci



Obrázek 13: Histogramy fitness

Řešení, která jsou od extrému vzdálená (mají větší fitness) se objevují s menší pravděpodobností. Pro vzájemné srovnání jednotlivých metod nám poslouží boxplot na obrázku 14. Krabice vyznačuje 25. a 75. percentil hodnot, linie uprostřed značí medián. Aritmetický průměr je pro srovnání vyznačen bodem. Vousy jsou dlouhé nejvýše 1,5násobek rozdílu 25. a 75. percentilu. Případné odlehlé hodnoty jsou vyznačeny křížkem.

Vidíme, že s největším rozptýlením hodnot skončila funkce `ga()`, což se ale snadno dá vysvětlit tím, že funkce nemusí vyčerpat maximální počet generací a v závislosti na vývoji nejlepšího řešení může skončit i po několika desítkách generací. Použitím lepšího nastavení místo výchozího by se pravděpodobně dalo dosáhnout lepších výsledků. Ostatní metody mají medián přibližně stejný, ačkoliv diferenciální evoluce má na hladině spolehlivosti 95% nižší medián než zbylé dvě (zářezy na krabicích se nepřekrývají). Statisticky významná je zřejmě spojitost mezi mediány evoluční strategie a genetického algoritmu.



Obrázek 14: Boxplot hodnot fitness

6 Závěr

Evoluční výpočetní techniky jsou efektivní způsob, jak nastavit PID regulátor. Výpočet pomocí čtyř zmíněných evolučních algoritmů (diferenciální evoluce, evoluční strategie a dvou implementací genetických algoritmů) trvá i přesto, že výpočet probíhá v MATLABu a na osobním počítači, cca 60-80 minut, a to včetně následné optimalizace. Pokud by se celý simulační model přepsal do kompilovaného jazyka, došlo by k dalšímu zrychlení výpočtů (ovšem za cenu práce spotřebované na přepis programu). Při použití výkonnějšího počítače by bylo možné získat nastavení regulátoru podstatně dřív, při použití jediného algoritmu pravděpodobně v řádu jednotek minut, t.j. takříkajíc „na počkání“.

I přes použití neoptimálních postupů při výpočtu (jednoduché metody výběru, poměrně malá velikost generace), dosahuje se rozumně dobrých nastavení PID regulátoru jak co do minimální hodnoty použitého kritéria, tak do přijatelného tvaru přechodové charakteristiky už základní verzí kriteriální funkce, bez dalších požadavků na tvar přechodové charakteristiky.

Seznam použité literatury

[1] Švarc, I.; Matoušek, R.; Šeda, M. & Vítečková, M. Automatizace-Automatické řízení, skriptum VUT v Brně, FSI, CERM Brno 2002, ISBN 80-214-2087-1

[2] Ing. Petr Bláha, P. & Prof. Ing. Petr Vavřín, D. Řízení a regulace I. Fakulta elektrotechniky a komunikačních technologií VUT v Brně, Ústav automatizace a měřící techniky, 2004.

[3] Doc. Ing. Švarc, I, Teorie automatického řízení. Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2003.

[4] Ingimundarson, A.; Hägglund, T. & Aström, K. J. Criteria of deisgn of PID controllers. IFAC Proceedings Volumes. 2003/18, s. 23-28.

[5] Cheng, S.-L. & Hwang, C. Designing PID controllers with a minimum IAE criterion by a differential evolution algorithm. Chemical Engineering Communications. Taylor & Francis. 1998/170, s. 83-115.

[6] Schaenzle, J. Introduction to the PID Control Algorithm [online]. 28.6.2016, [cit. 28.4.2018]. Dostupné z: https://spin.atomicobject.com/2016/06/28/intro-pid-control/

[7] Zhong, J. PID Controller Tuning: A Short Tutorial [online]. 2006, [cit. 28.4.2018]. Dostupné z: http://saba.kntu.ac.ir/eecd/pcl/download/PIDtutorial.pdf

[8] Ang, K. H.; Chong, G. & Li, Y. PID control system analysis, design, and technology. IEEE Transactions on Control Systems Technology. 2005/4, s. 559-576. ISSN 1063-6536

[9] prof. Farid A. Tolbah & El-Bardy, I. W. A. PID Tuning using Zieglers Nicholas [online]. 8.12.2014, [cit. 28.4.2018]. Dostupné z: https://www.slideshare.net/wbadry/pid-tuning-42460795

[10] Trammell, L. Ziegler-Nichols Tuning Rules for PID [online]. 2018 [cit. 28.4.2018]. Dostupné z: http://www.mstarlabs.com/control/znrule.html

[11] Studnička, V. Genetické algoritmy - multi-core CPU implementace. Brno, 2010. Diplomová práce. Vysoké učení technické v Brně, Ústav automatizace a informatiky. Vedoucí práce Ing. Radomil Matoušek, PhD.

[12] Chalupník, V. Seriál Biologicky inspirované algoritmy [online]. 2012 [cit. 28.4.2018]. Dostupné z: https://www.root.cz/serialy/biologicky-inspirovane-algoritmy/

[13] Matěj Lepš. Evoluční strategie [online]. 16.11.2007 [cit. 28.4.2018]. Dostupné z: https://mech.fsv.cvut.cz/~leps/teaching/mmo/prednasky/prednaska07_ES_DE.pdf

[14] Rechenberg, I. (1973). Evolution strategy: Optimization of technical systems by means of biological evolution. Fromman-Holzboog, Stuttgart.

[15] Price, K. & Storn, R. Differential Evolution for Continuous Function Optimization [online]. [cit. 28.4.2018]. Dostupné z: http://www1.icsi.berkeley.edu/~storn/code.html

[16] MathWorks, Inc. How the Genetic Algorithm Works - Matlab & Simulink [online]. 2017, [cit. 28.4.2018]. Dostupné z: https://www.mathworks.com/help/gads/how-the-genetic-algorithm-works.html

Seznam obrázků

Seznam tabulek

Seznam programů

Seznam použitých zkratek

EVT evoluční výpočetní techniky

DE diferenciální evoluce

ES evoluční strategie

GA genetický algoritmus

GAm genetický algoritmus - matlabovská funkce `ga()`

[**metoda**] **optim** optimalizované nastavení nalezené danou metodou

Seznam příloh k bakalářské práci

- CD - kontrolní součet sha256 souboru checksums.txt je

  `c45d2beb32fc8d1aa99e77614bdd3d50128169be3476dbb7b0fcc60c0be310c3`

Přílohy

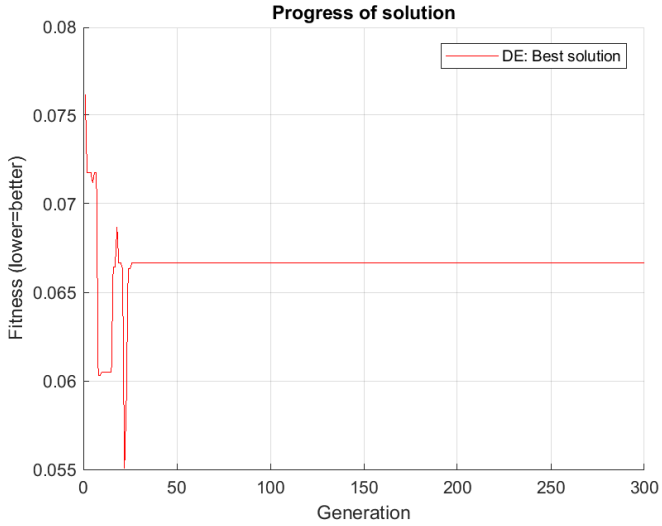
A Řešení první soustavy

Jak je vidět na obrázku 15a, diferenciální evoluce nalézá svoje řešení velice rychle - před 50. generací - ale jak je z grafu vidět, řešení bylo nalezeno náhodně a je ihned ztraceno a do konce běhu algoritmu už není znovu nalezeno. Původní přechodová charakteristika je téměř bez překmitu obr. 15b), optimalizovaná je již zcela aperiodická (obr. 15c). Řešení má velice rychlý čas ustálení.

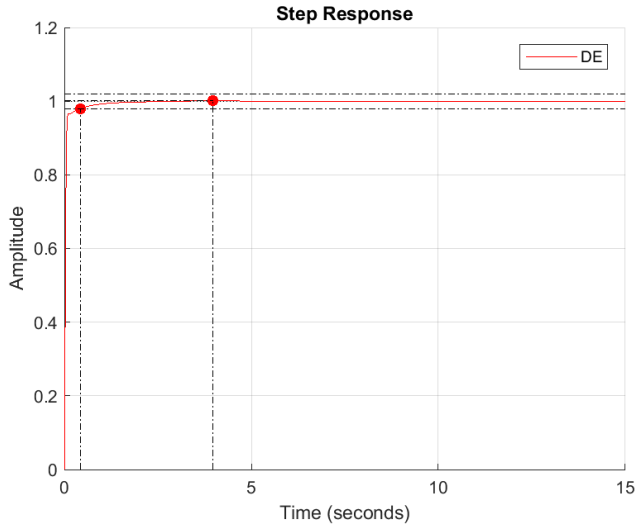
Řešení evoluční strategií nalézá své řešení pomaleji (obr. 16a), ale na rozdíl od diferenciální evoluce nedochází ke zhoršování výsledku, což je dáno podstatou metody (nejedná se o náhodný jev). První řešení algoritmu (obr. 16b) je s nevysokým překmitem a extrémně nízkou dobou ustálení. Optimalizace nevede k velké změně (obr. 16c), původní nalezené řešení bylo velmi blízko skutečnému extrému.

Genetický algoritmus vyprodukoval podobné řešení jako diferenciální evoluce, a ačkoliv je konečná hodnota fitness vyšší, řešení nebylo náhodně ztraceno a mohlo dojít k jeho vylepšení už genetickým algoritmem (obr. 17a). Přechodové charakteristiky (původní na obr. 17b, optimalizovaná na obr. 17c) jsou téměř aperiodické, s extrémně nízkým časem ustálení a maximálním překmitem stále v 2% pásmu ustálení.

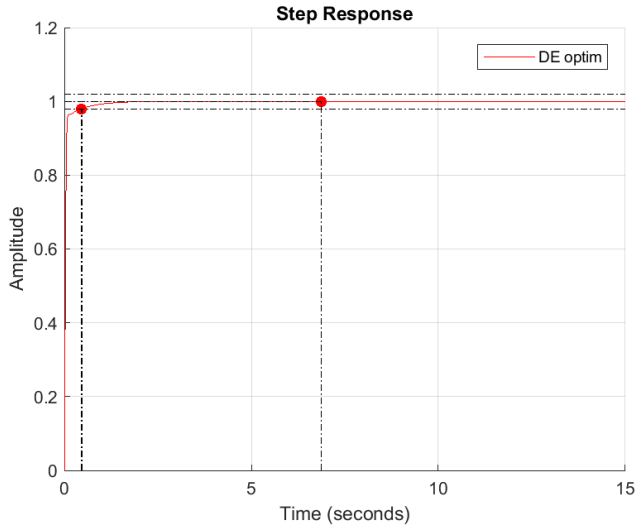
Funkce `ga()` snižovala hodnotu fitness ze všech metod nejrychleji (během deseti generací se dosáhlo konečného výsledku). Došlo zde k uplatnění ukončovací podmínky změny fitness, algoritmus se zastavil po necelých 60 generacích (obr. 18a). Přechodová charakteristika (původní na obr. 18b, optimalizovaná na obr. 18c) má extrémně nízký čas ustálení a je opět téměř aperiodická, nejvyšší překmit se nachází v 2% pásmu ustálení.



(a) Postup řešení

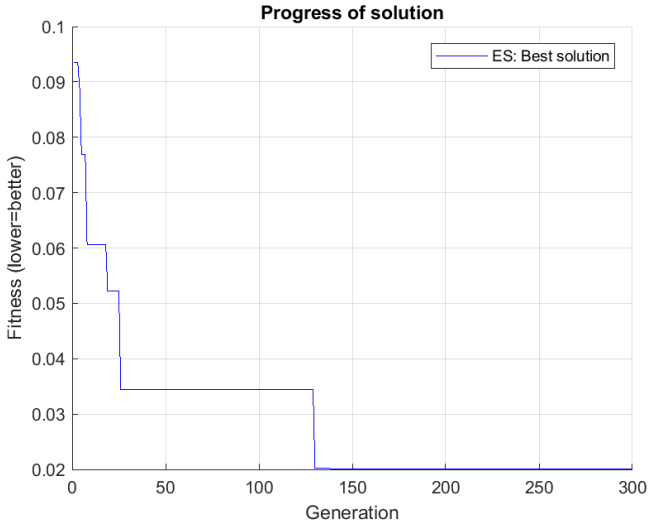


(b) Přechodový děj

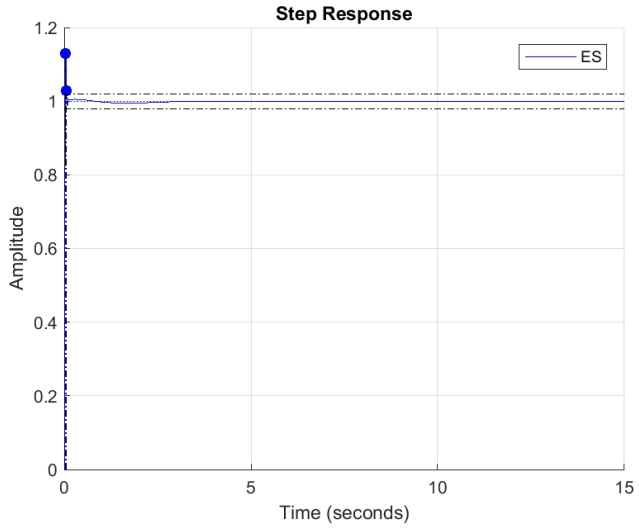


(c) Přechodový děj (po optimalizaci)

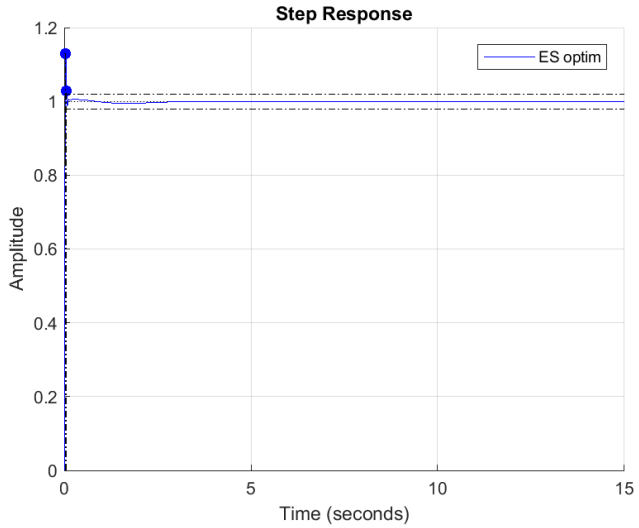
Obrázek 15: První soustava - řešení diferenciální evolucí



(a) Postup řešení

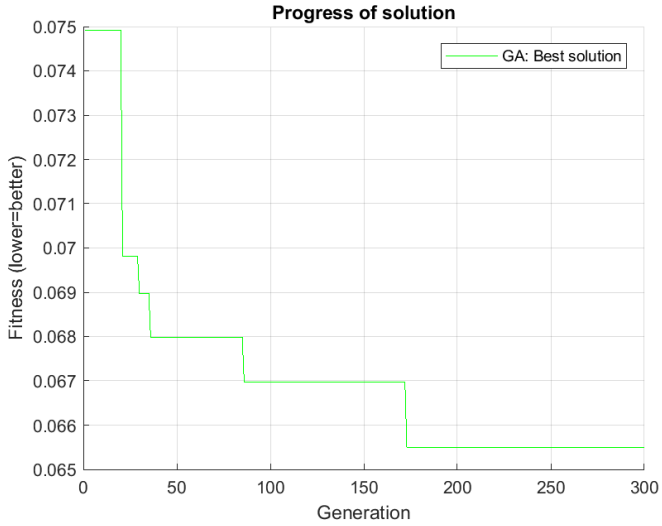


(b) Přechodový děj

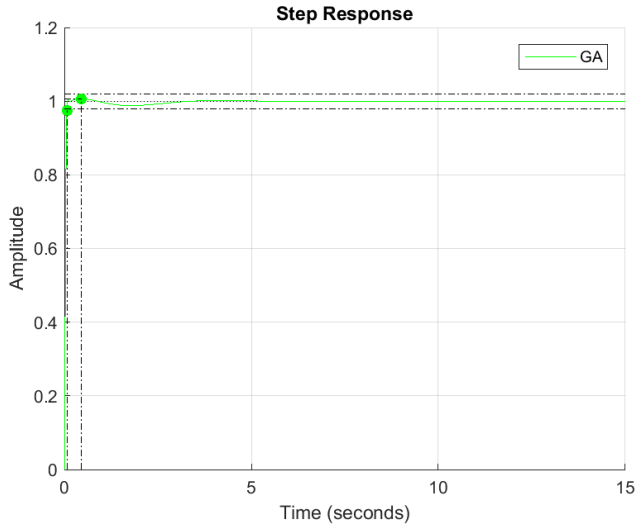


(c) Přechodový děj (po optimalizaci)

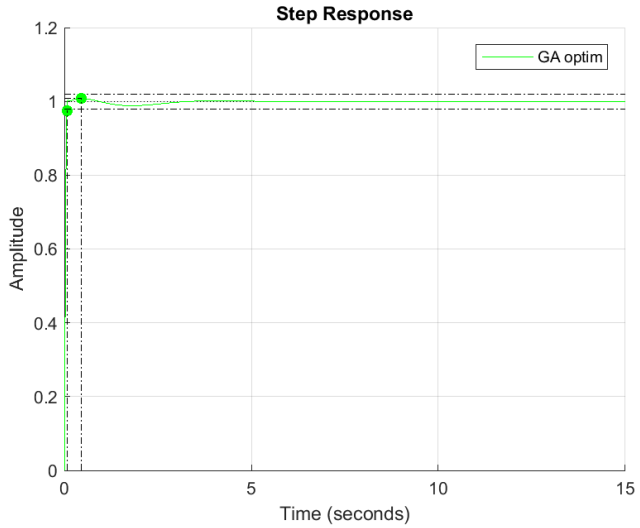
Obrázek 16: První soustava - řešení evoluční strategií



(a) Postup řešení

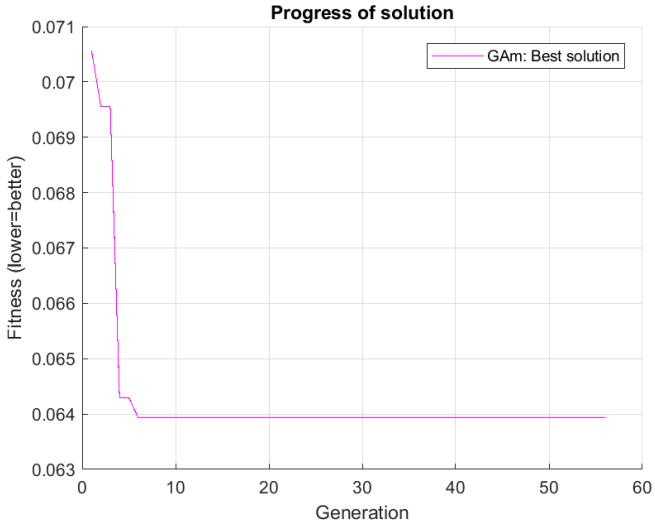


(b) Přechodový děj

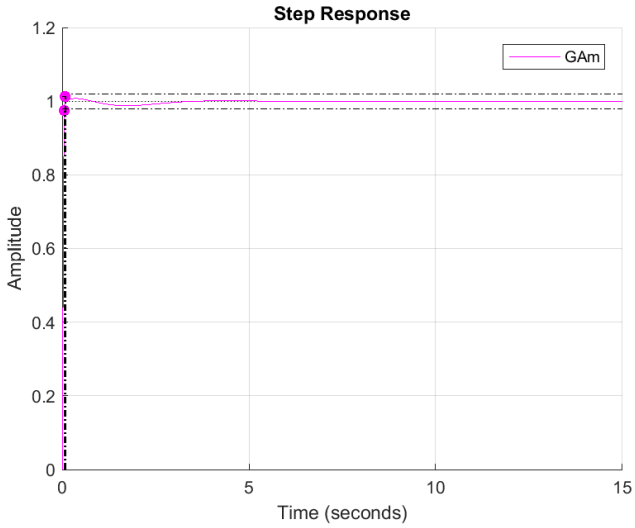


(c) Přechodový děj (po optimalizaci)

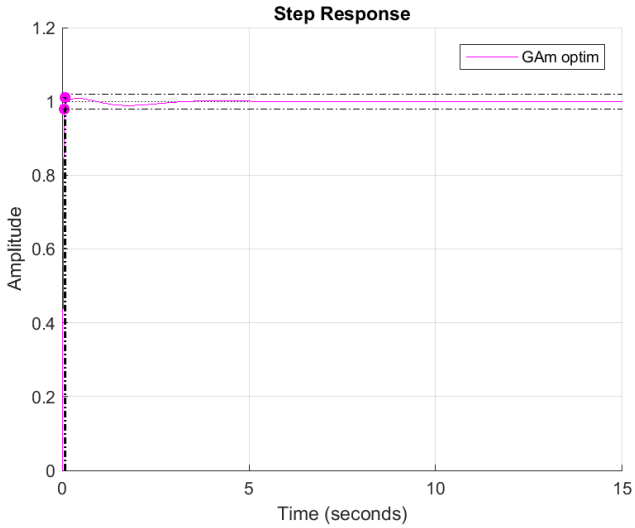
Obrázek 17: První soustava - řešení genetickým algoritmem



(a) Postup řešení

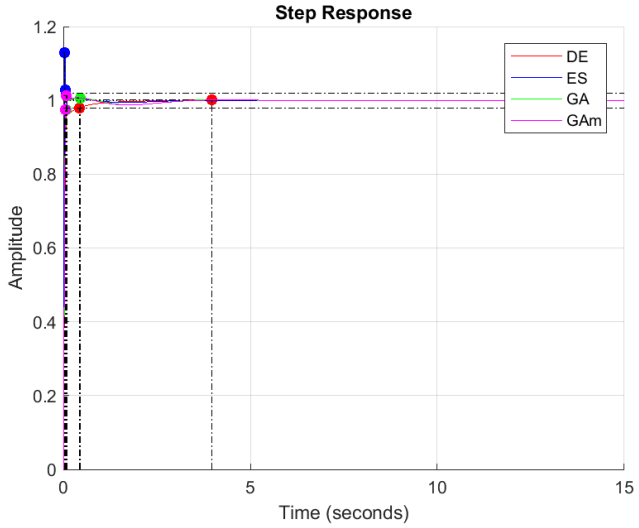


(b) Přechodový děj

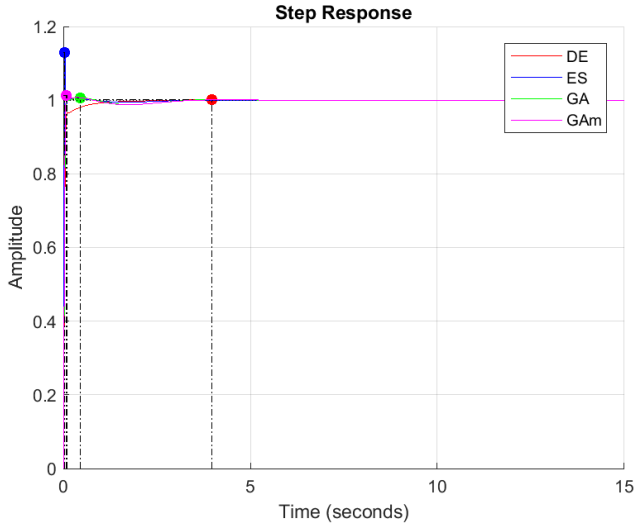


(c) Přechodový děj (po optimalizaci)

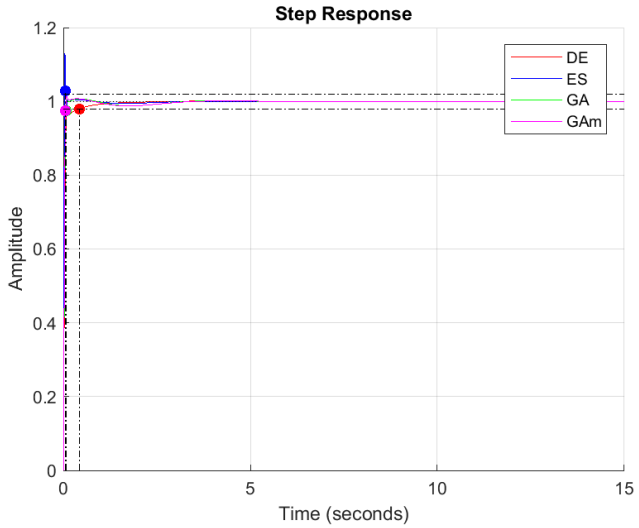
Obrázek 18: První soustava - řešení funkcí `ga()`



(a) Řešení soustavy se zvýrazněním překmitu a času ustálení

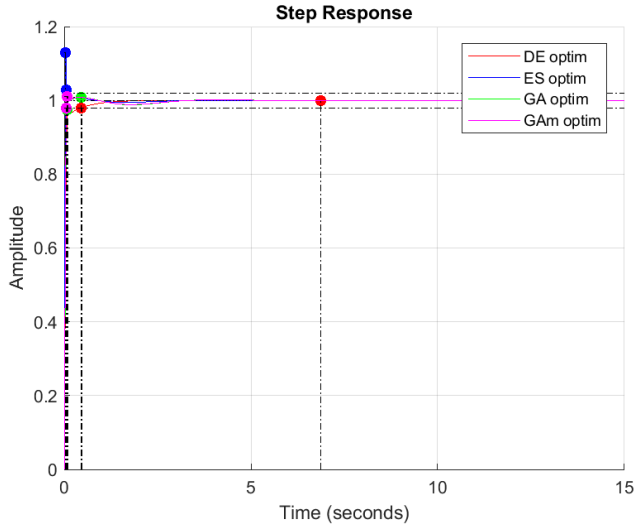


(b) Řešení soustavy se zvýrazněním překmitu

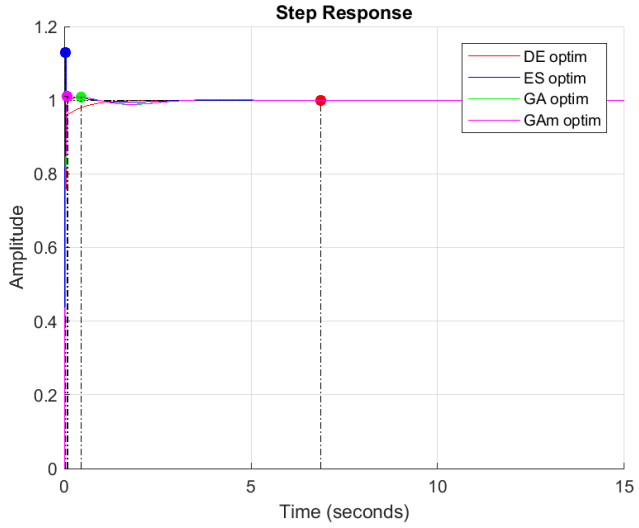


(c) Řešení soustavy se zvýrazněnímbčasu ustálení

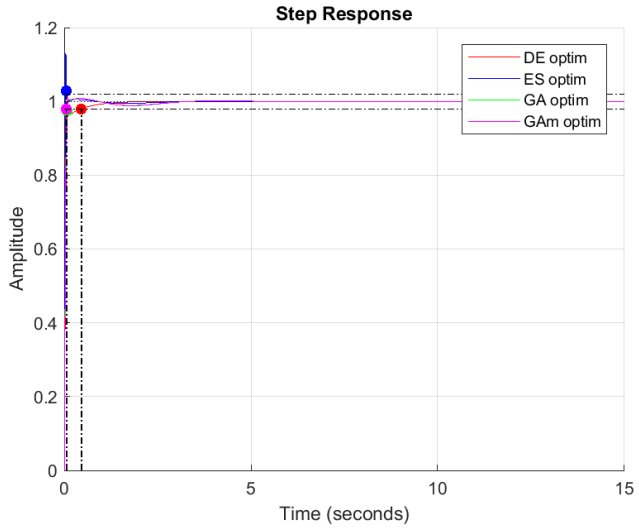
Obrázek 19: První soustava - srovnání přechodových charakteristik (před optimalizací)



(a) Řešení soustavy se zvýrazněním překmitu a času ustálení



(b) Řešení soustavy se zvýrazněním překmitu



(c) Řešení soustavy se zvýrazněnímbčasu ustálení

Obrázek 20: První soustava - srovnání přechodových charakteristik (po optimalizaci)

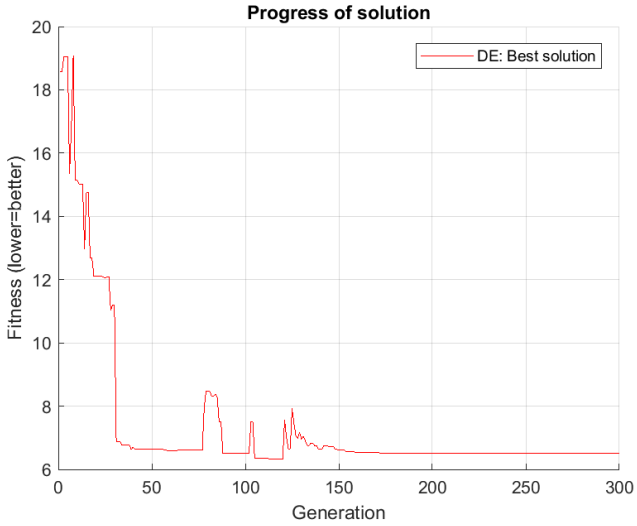
B Řešení druhé soustavy

Diferenciální evoluce v tomto případě již nenalezne řešení „náhodně“, ale zamýšlenou funkcí algoritmu. Řešení se udrželo po několik generací (jak můžeme vidět na obr. 21a), než bylo náhodnostním procesem ztraceno. Přechodové charakteristiky jsou si velice podobné (obr. 21b a 21c), algoritmus našel řešení velice blízko skutečnému extrému. Následná optimalizace nemá viditelné výsledky.

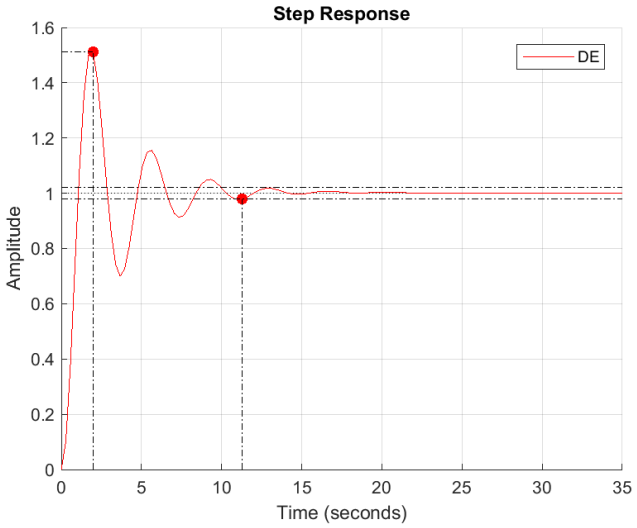
Nalezení konečného řešení metodou evoluční strategie proběhlo velice brzy (cca u 115. generace, jak lze vidět na obr. 22a). Řešení je bohužel s pěti až šesti viditelnými kmity a vysokým překmitem (obr. 22b), i když čas ustálení je velice podobný řešení diferenciální evolucí. Nalezené řešení je taktéž velice blízko skutečnému extrému a optimalizace konvenčními metodami nemá viditelný vliv.

Řešení genetickým algoritmem je se třemi viditelnými kmity (obr. 23b) a překmitem asi 50% (což ale není v porovnání s ostatními metodami vysoká hodnota). Následnou optimalizací zde dojde k určitému zlepšení fitness, což je ale vykoupeno nárůstem času ustálení (obr. 23c).

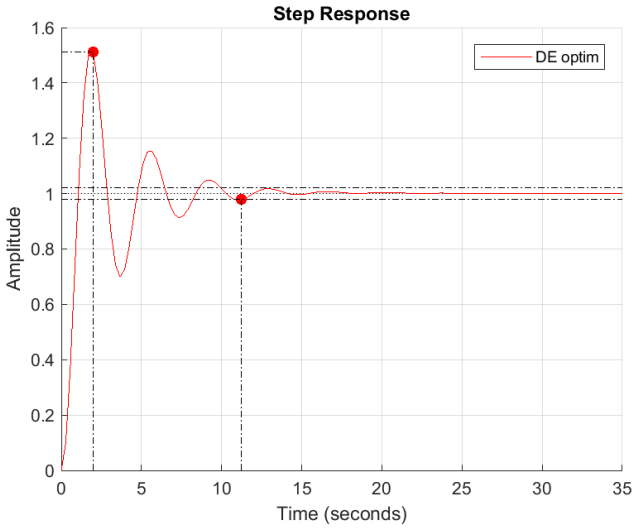
Funkce `ga()` se i u této soustavy zastaví před vyčerpání maximálního počtu generací (obr. 24a) s dvojnásobným fitenss oproti ostatním metodám. To je zde vykoupeno následnou optimalizací, při které se řešení přiblíží hodnotám ostatních EVT, i když jich zcela nedosáhne (fitness je stále zřetelně vyšší).



(a) Postup řešení

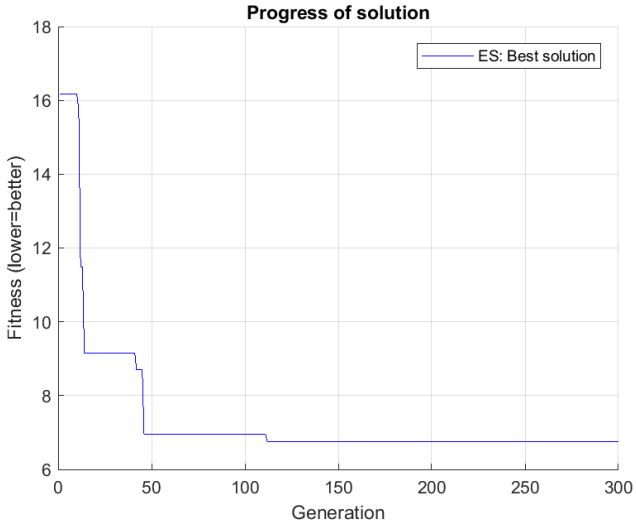


(b) Přechodový děj

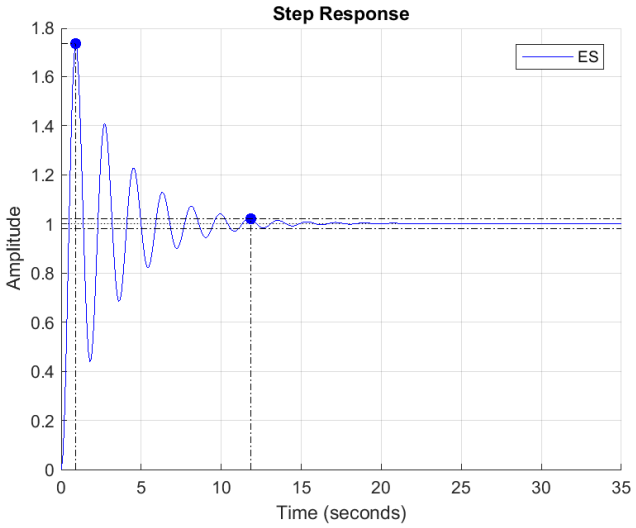


(c) Přechodový děj (po optimalizaci)

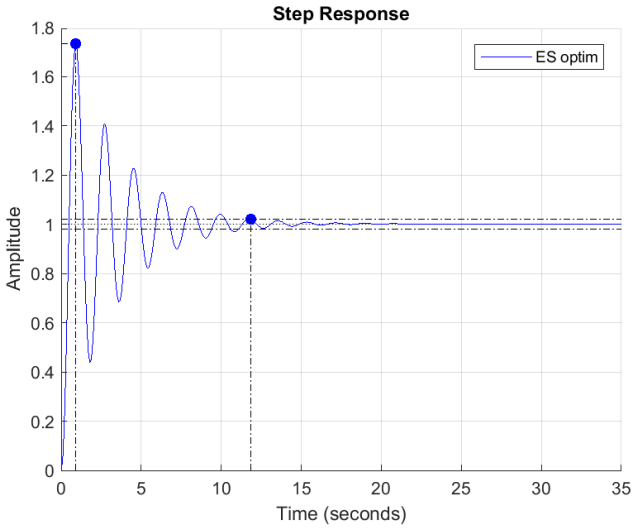
Obrázek 21: Druhá soustava - řešení diferenciální evolucí



(a) Postup řešení

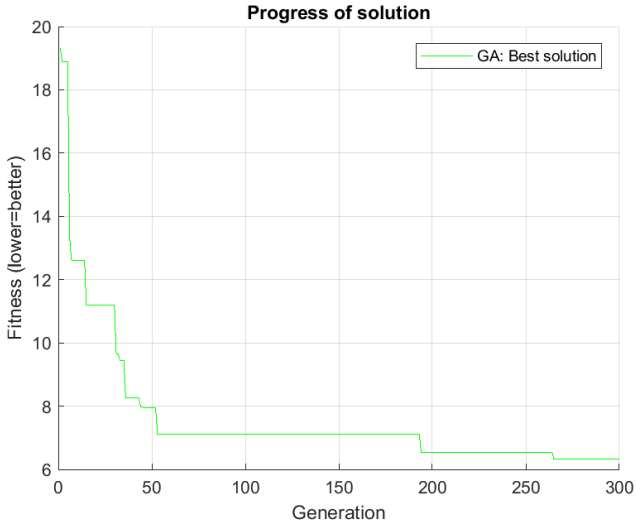


(b) Přechodový děj

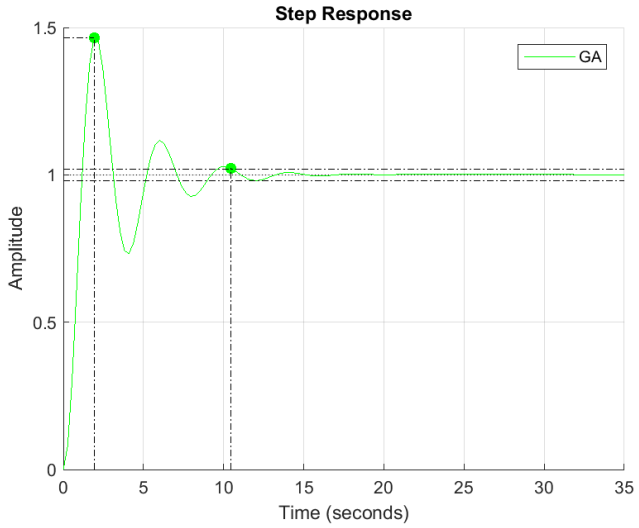


(c) Přechodový děj (po optimalizaci)

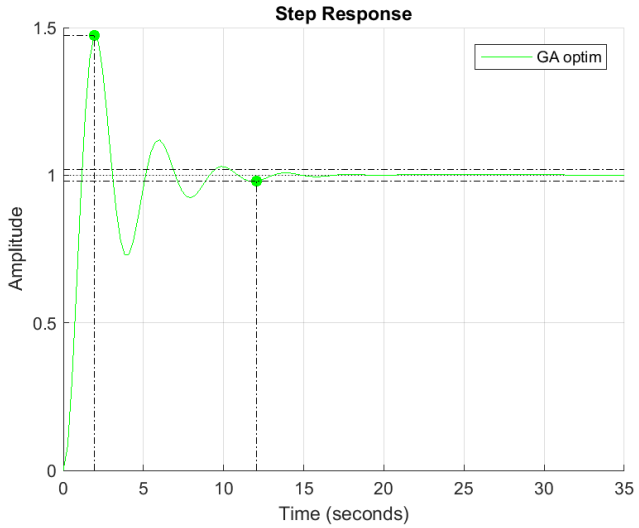
Obrázek 22: Druhá soustava - řešení evoluční strategií



(a) Postup řešení

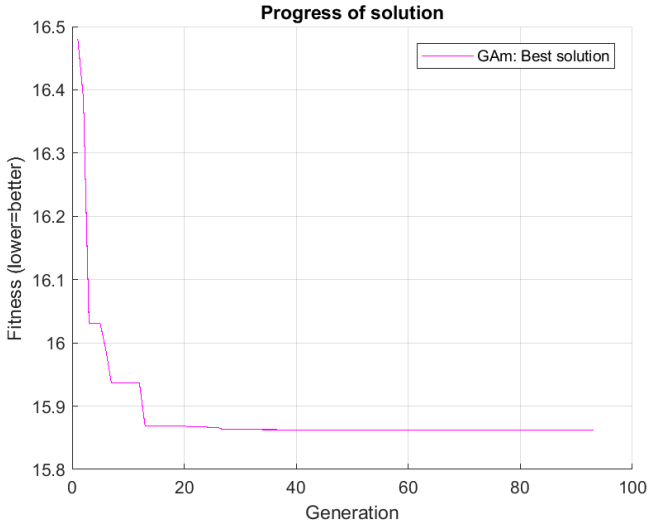


(b) Přechodový děj

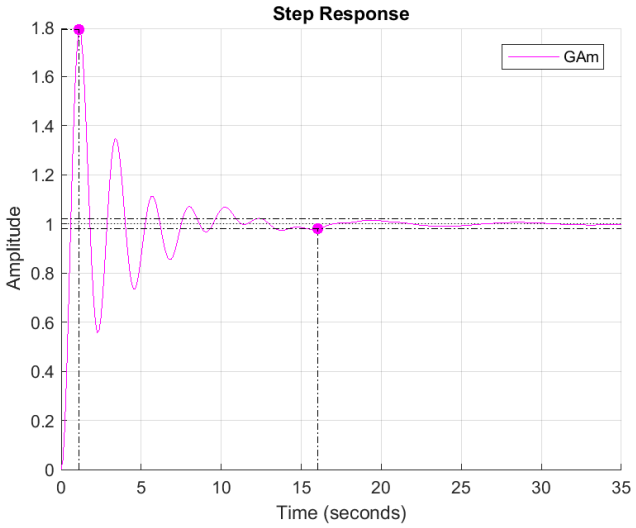


(c) Přechodový děj (po optimalizaci)

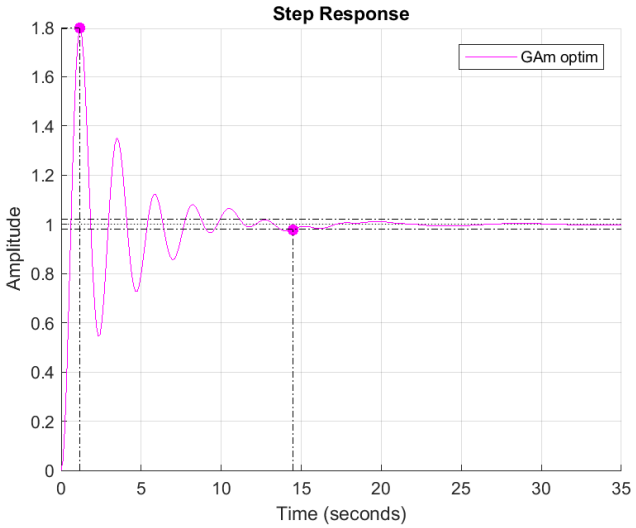
Obrázek 23: Druhá soustava - řešení genetickým algoritmem



(a) Postup řešení

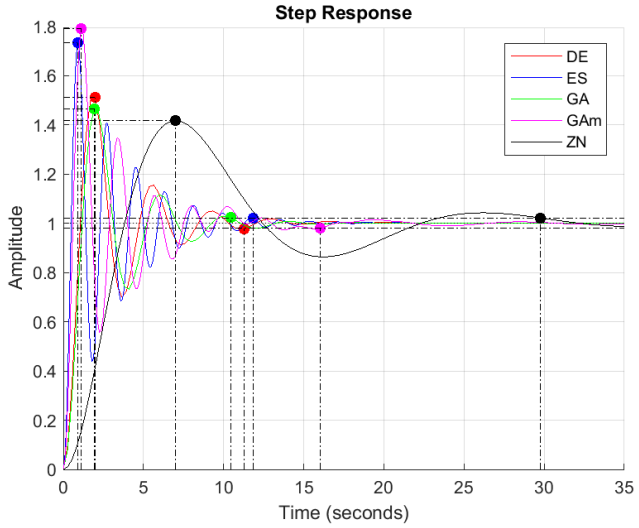


(b) Přechodový děj

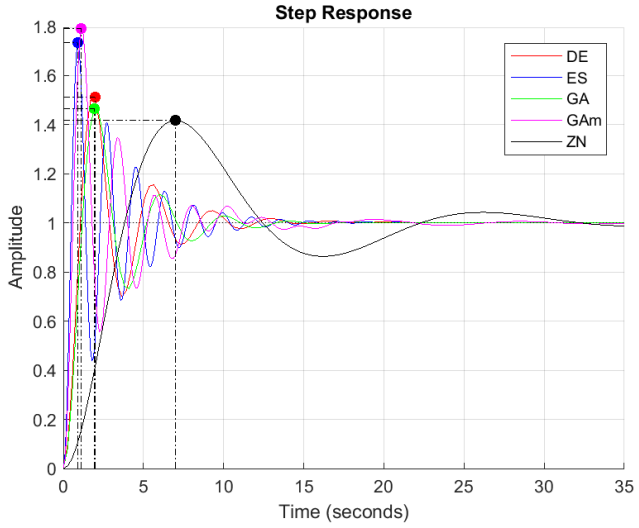


(c) Přechodový děj (po optimalizaci)

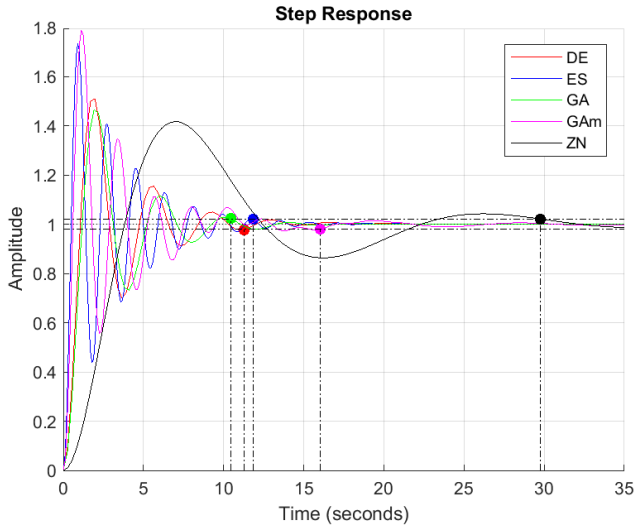
Obrázek 24: Druhá soustava - řešení funkcí `ga()`



(a) Řešení soustavy se zvýrazněním překmitu a času ustálení

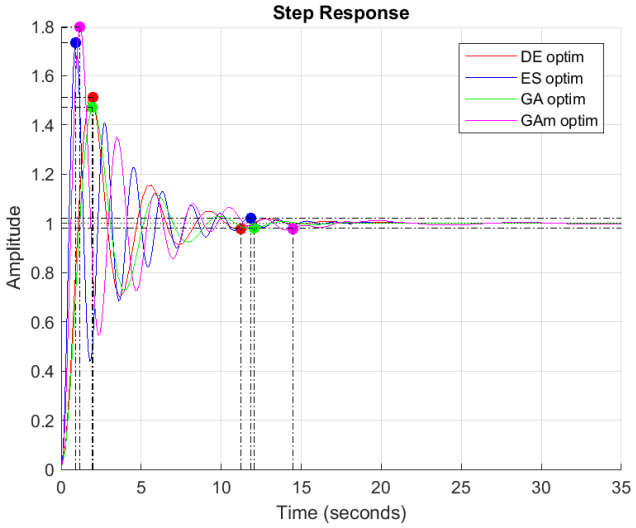


(b) Řešení soustavy se zvýrazněním překmitu

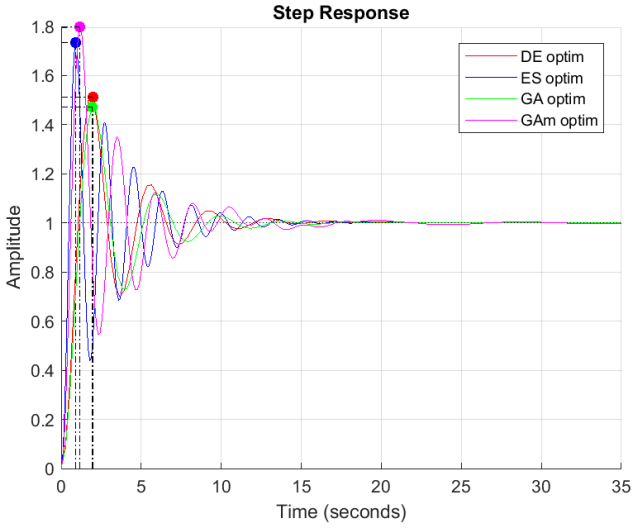


(c) Řešení soustavy se zvýrazněnímbčasu ustálení

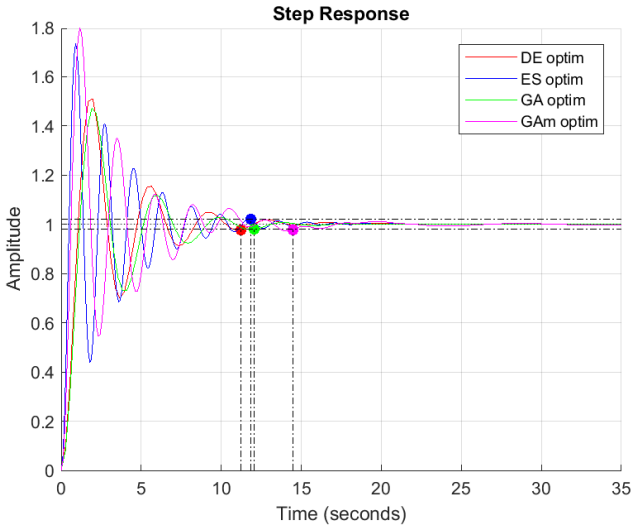
Obrázek 25: Druhá soustava - srovnání přechodových charakteristik (před optimalizací)



(a) Řešení soustavy se zvýrazněním překmitu a času ustálení



(b) Řešení soustavy se zvýrazněním překmitu



(c) Řešení soustavy se zvýrazněnímbčasu ustálení

Obrázek 26: Druhá soustava - srovnání přechodových charakteristik (po optimalizaci)

C Některé z použitých funkcí

C.1 Get_ZN_params

Program 2: Nastavení regulátoru metodou Ziegler-Nichols

```
function params = get_ZN_params(system)
        % calculates parameters of PID controller by
        % Ziegler-Nichols method
        % @param system continuous plant model, as in tf()
        % @returns params vector of parameters in [P I D]
           form
        % to be used with get_controller function
        m=allmargin(system);
        if(m.Stable==0)
                throw(MException('E:NO_STABLE','Closed loop
                   is not stable'));
        end

        [Kc, ~, fr, ~]=margin(system);
        Pu=2*pi/fr;
        % recalculate params to use in Kp+Ki/s+Kd*s form
        Kp=Kc*0.59;
        Ki=Kc/(Pu*0.5);
        Kd=Kc*(Pu*0.12);

        params=[Kp Ki Kd];
end
```