BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER SYSTEMS
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

EVOLUTIONARY ANALOG AMPLIFIER OPTIMISATION

EVOLUČNÍ OPTIMALIZACE ANALOGOVÝCH ZESILOVAČŮ

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR **MAREK BIELIK**
AUTOR PRÁCE

SUPERVISOR **Ing. MICHAL BIDLO, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2017

Abstract

The standard way to design an amplifier is to calculate the values of its components by mathematical equations. The goal of this thesis is to use artificial intelligence and make the computer determine the values without the equations.

Abstrakt

Do tohoto odstavce bude zapsán výtah (abstrakt) práce v českém (slovenském) jazyce.

Keywords

artificial intelligence, evolutionary computation, evolution strategy, SPICE, ngSPICE, analogue amplifier,

Klíčová slova

Sem budou zapsána jednotlivá klíčová slova v českém (slovenském) jazyce, oddělená čárkami.

Reference

BIELIK, Marek. *Evolutionary Analog Amplifier Optimisation.* Brno, 2017. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Bidlo Michal.

Evolutionary Analog Amplifier Optimisation

Declaration

Hereby I declare that this bachelor's thesis was prepared as an original author's work under the supervision of Ing. Michal Bidlo, Ph.D. All the relevant information sources, which were used during preparation of this thesis, are properly cited and included in the list of references.

.....
Marek Bielik
March 26, 2017

Acknowledgements

Thank you all, my family, supervisor and Georgina.

Contents

Chapter 1

Preface

The standard way to design an amplifier is to calculate the values of its components by mathematical equations. The goal of this thesis is to use artificial intelligence and make the computer determine the values without the equations.

Chapter 2

Evolutionary algorithms

> *'Owing to this struggle for life, variations, however slight and from whatever cause proceeding, if they be in any degree profitable to the individuals of a species, in their infinitely complex relations to other organic beings and to their physical conditions of life, will tend to the preservation of such individuals, and will generally be inherited by the offspring. The offspring, also, will thus have a better chance of surviving, for, of the many individuals of any species which are periodically born, but a small number can survive. I have called this principle, by which each slight variation, if useful, is preserved, by the term Natural Selection'.* [1, p. 115]

In biological evolution, species are selected depending on their relative success in surviving and reproducing in the environment and mutations play a crucial role in this process as they are the key to the adaptation of the species. This concept has been used as a metaphor in computer science and it inspired the formation of a whole new area of artificial intelligence called Evolutionary Computation. This area mainly deals with an *optimization problem* which can be expressed in the following way: We have a function $f : A \to \mathbb{R}$ from a set A to the real numbers and we are looking for an element x_0 in A such that $f(x_0) \leq f(x)$ or $f(x_0) \geq f(x)$ for all x in A (we are looking for either the global minimum or maximum of the function f). The domain A of f is called the *search space* and the elements of A are called *candidate solutions*. The function f can be called variously (an *objective function*, a *loss function*), in this text we will call the function f a *fitness function*. A candidate solution that gives us the global maximum or minimum of the fitness function is called an *optimal solution*. The search space A is usually in the form of $\mathbb{R}^n$ n-dimensional Euclidean space and the function f is also n-dimensional.
A simple approach to find the optimal solution would be to scour the whole search space and try every candidate solution. This approach will always give us the optimal solution, but the search space can be so vast, that we are not able to try every candidate solution in reasonable time.
A more sophisticated approach would be to use the technique presented in algorithm 1. The *population* $P \subseteq A$ would be a subset of the search space A and every *individual* of the population would represent one element of A. The individuals can be also called *chromosomes* in certain types of evolutionary algorithms. Every loop in the algorithm represents one *generation* of individuals. This approach helps us to avoid those elements in the search space which wouldn't provide any good solution but the finding of the optimal solution is not guaranteed.

From the biological point of view, this algorithm can be seen as the driving force behind evolution and from the mathematical point of view it can be seen as a stochastic, derivative-free numerical method for finding global extrema of functions that are too hard or impossible to find analytically. Biology uses this approach to create well adapted beings and humans can use this method to find optimal solutions to problems that are difficult to solve analytically.

Algorithm 1 Evolutionary algorithm

1: Create a *population* P of randomly selected candidate solutions;
2: **while** terminating condition **do**
3: *(Selection)* Select the appropriate individuals (parents) for breeding from P;
4: *(Mutation)* Generate new individuals (offspring) from the parents;
5: *(Reproduction)* Replace some or all individuals in P with the new individuals;
6: **end while**

2.1 Evolution Strategies

Evolution strategies (ES) is a family of stochastic optimization algorithms which belongs to the general class of evolutionary algorithms. It was created by Rechenberg and Schwefel in the early 60s and attracted a lot of attention due to its strong capability to solve real-valued engineering problems. ES typically uses a real-valued representation of chromosomes and it relies primarily on selection and mutation to drive the evolutionary process. ES also often implements *self-adaptaion* of the parameters used for mutating the parents in the population during the evolution, which means that these parameters coevolve with the individuals. This feature is natural in evolution strategies and other evolutionary algorithms have adopted it as well over the last years.
Algorithm 2 is a basic non-self-adaptive form of ES. Chromosomes in this algorithm are in the form of vectors of real numbers $x = (x_1, ..., x_n) \in \mathbb{R}^n$. Every vector represents one solution of the search space in our problem domain and the goal of the algorithm is to find such vector that produces global extremum of our fitness function.
The terminating condition of the algorithm can have various forms which can be combined together:

- Wait until the algorithm finds a chromosome with a fitness function value that meets our requirements.
- Limit the total number of generations.
- Track the best chromosome in every generation and stop the evolution if the fitness doesn't change anymore.

The last two approaches are suitable when the algorithm gets stuck in a local optimum and doesn't leave it after a significant amount of time.
This form of the algorithm is also implemented in this thesis with various extensions in order to produce better results. These extensions are described and discussed in the rest of this chapter.

Algorithm 2 Non-self-adaptive Evolution Strategies $(\mu + \lambda)$

1: Randomly create an initial population $\{x^1, ..., x^\mu\}$ of parent vectors, where each x^i is of the form $x^i = (x_1^i, ..., x_n^i), i = 1, ..., \mu$;
2: Evaluate the fitness function of each chromosome;
3: **while** terminating condition **do**
4: **repeat**
5: Randomly select a parent from $\{x^i : i = 1, ...\mu\}$;
6: Create a child vector by applying a mutation operator to the parent;
7: **until** λ children are generated
8: Rank the $\mu + \lambda$ chromosomes (children and parents) from the best to worst;
9: Select the best μ of these chromosomes to continue into the next generation;
10: **end while**

2.2 Selection

There are two main parameters in evolutionary strategies that are used to describe the type of the selection process:

1. **Parameter $\boldsymbol{\mu}$** specifies the number of chromosomes (parents) that are selected for reproduction
2. **Parameter $\boldsymbol{\lambda}$** specifies the number of children that are produced in every generation.

There are also two types of selection:

1. **$(\boldsymbol{\mu} + \boldsymbol{\lambda})$-ES** selection scheme
2. **$(\boldsymbol{\mu}, \boldsymbol{\lambda})$-ES** selection scheme

In the first scheme, denoted by $(\mu+\lambda)$-ES, both parents from the previous generation and newly produced children are mixed together and compete for survival in every generation where only the best μ chromosomes (parents for the next generation) are selected and get the chance to be reproduced. This scheme implements so called *elitizm*, because if there is a chromosome with a very good fitness function, it can survive for many generations and it can be replaced only if a better individual occurs. This approach has a higher natural tendency to get stuck in a local optima than the second scheme.
In the second type of selection, denoted (μ, λ)-ES, the parents are selected only from the set of children λ, so every chromosome lives only in one generation. Even a chromosome with a very good fitness function is discarted and replaced by a child with potencially worse fitness, so the convergence is not as strict as in the previous scheme, but it is easier to move away from a local optimum for this method.
Both schemes are implemented in this thesis with various values of μ and λ parameters ($\mu \ll \lambda$). Schemes such as $(1 + 1)$-ES and $(1, \lambda)$-ES are also possible ($(1 + 1)$-ES was mainly studied when evolution strategies was invented), but they show a slower convergency properties.

2.3 Mutation

Chromosomes in evolution strategies are represented as vectors of real values, formula 2.1.

$$x = (x_1, ..., x_n) \tag{2.1}$$

These values correspond to the solution variables. Mutations can be performed in various ways, the simplest form is shown in formula 2.3 where m is a real number drawn from a normal (Gaussian) distribution with the mean in 0 and the stadard deviation σ is chosen by the user. The parameter σ represents the mutation step size and its value is crucial since it determines the effect of the mutation operator. The value should be kept relatively small to the problem domain in order not to perform large mutations too often. The parent chromosome is denoted by $x(t)$ and $x(t+1)$ denotes a child.

$$m = N(0, \sigma) \tag{2.2}$$

$$x(t+1) = x(t) + m \tag{2.3}$$

This approach ignores two important facts:

1. When the search is close to the global optimum, it is appropriate to perform only subtle changes to the chromosomes so that the the algorithm will not leave the good region.

2. Every dimension in the solution space can require different scaling, so that one dimension can require large mutation steps while another only small ones.

Both these facts became crucial to the ability of the algorithm to find an optimal solution during the implementation, so they are discussed and examined in the following sections.

2.3.1 Mutation with One Step Size

In this version of mutation, we change the the vector x, formula 2.4.

$$x = ((x_1, ..., x_n), \sigma) \tag{2.4}$$

$(x_1, ..., x_n)$ is the original vector and σ is a real-valued strategy parameter which controls the mutation process for every chromosome separately. The mutation process is described in formulas 2.6 and 2.7. The constant τ is set by the user and it is inversely proportional to the square root of the problem size. It represents the learning rate of the algorithm. The σ parameter is then mutated by multiplying with a variable with log-normal distribution, which ensures that σ is always positive. The reasons for this form of mutation of σ by a log-normal distribution are stated in [**?**]. The newly mutated σ is then used to mutate the solution values for the child where $i = 1, ..., n$ are the n elements making up one chromosome. It is important to keep the right order of the mutation - to mutate the value of σ first and then mutate the vector's elements itself.

$$\tau \propto \frac{1}{\sqrt{n}} \tag{2.5}$$

$$\sigma(t+1) = \sigma(t) * e^{\tau * N(0,1)} \tag{2.6}$$

$$x_i(t+1) = x_i(t) + \sigma(t+1) * N_i(0,1), \quad i = 1, ..., n \tag{2.7}$$

This approach allows the mutation step size to coevolve with the chromosomes, since the fitness function indirectly evaluates the suitability of the mutation. It is important to vary the step size during the evolution run as we want the algorithm to prefer longer steps when it is far away from the optima and small steps when it is close to the optima.

2.3.2 Mutation with n-Step Size

The fitness function surface can have different inclination in every dimension, so while one dimension requires large mutation steps another one might require only subtle ones. We can solve this problem by adding a strategy parameter σ to every element of the chromosome's solution vector, formula 2.8.

$$x = ((x_1, ..., x_n), \sigma_1, ..., \sigma_n) \tag{2.8}$$

The mutation process is then described in formulas 2.10 and 2.11.

$$\tau' \propto \frac{1}{\sqrt{2\sqrt{n}}} \tag{2.9}$$

$$\sigma_i(t+1) = \sigma_i(t) * e^{\tau' * N(0,1)' + \tau * N_i(0,1)} \tag{2.10}$$

$$x_i(t+1) = x_i(t) + \sigma_i(t+1) * N(0,1)_i, \quad i = 1, ..., n \tag{2.11}$$

Equation 2.6 differs from 2.10. A simple modification of 2.6 which we could do is shown in formula 2.12. The reason why we add one more normally distributed variable in the actual algorithm (equation 2.10) is because we want keep the overall change of the mutability but we also want a finer granularity on the coordinate level.

$$\sigma_i(t+1) = \sigma_i(t) * e^{\tau * N_i(0,1)} \tag{2.12}$$

Chapter 3

Implementation

3.1 Fitness function

A fitness function provides us means by which we can determine how good the evolved solution is, so that we can distinguish the more appropriate candidates from the less appropriate ones. It is important to mention that the quality of this function is essential for every evolutionary algorithm, because the ability of the algorithm to find the most suitable solution is highly dependent on the ability of the fitness function to distinguish between the individuals. The value of the function is a positive real number in our case and the exact value is not very important, because the algorithm uses this value only for comparing the individuals between each other and we consider those individuals with the lower value as the better ones. There are three fitness functions implemented in this thesis.

3.1.1 Best match with the reference solution

This fitness function was used as the first solution in order to find out if the evolution has the ability to find the desired solution. It is based on the method of least squares which is used in regression analysis. In terms of fitness function, every individual is represented by a vector

Chapter 4

Conclusion

Bibliography

[1] Knuth, D. E.: *The TEXbook*. Addison-Wesley Publishing Company. 1996. ISBN 0-201-13447-0.

Appendices

Appendix A

Jak pracovat s touto šablonou

V této kapitole je uveden popis jednotlivých částí šablony, po kterém následuje stručný návod, jak s touto šablonou pracovat.

Jedná se o přechodnou verzi šablony. Nová verze bude zveřejněna do konce roku 2016 a bude navíc obsahovat nové pokyny ke správnému využití šablony, závazné pokyny k vypracování bakalářských a diplomových prací (rekapitulace pokynů, které jsou dostupné na webu) a nezávazná doporučení od vybraných vedoucích. Jediné soubory, které se v nové verzi změní, budou projekt-01-kapitoly-chapters.tex a projekt-30-prilohy-appendices.tex, jejichž obsah každý student vymaže a nahradí vlastním. Šablonu lze tedy bez problémů využít i v současné verzi.

Popis částí šablony

Po rozbalení šablony naleznete následující soubory a adresáře:

bib-styles Styly literatury (viz níže).

obrazky-figures Adresář pro Vaše obrázky. Nyní obsahuje placeholder.pdf (tzv. TODO obrázek, který lze použít jako pomůcku při tvorbě technické zprávy), který se s prací neodevzdává. Název adresáře je vhodné zkrátit, aby byl jen ve zvoleném jazyce.

template-fig Obrázky šablony (znak VUT).

fitthesis.cls Šablona (definice vzhledu).

Makefile Makefile pro překlad, počítání normostran, sbalení apod. (viz níže).

projekt-01-kapitoly-chapters.tex Soubor pro Váš text (obsah nahraďte).

projekt-20-literatura-bibliography.bib Seznam literatury (viz níže).

projekt-30-prilohy-appendices.tex Soubor pro přílohy (obsah nahraďte).

projekt.tex Hlavní soubor práce – definice formálních částí.

Výchozí styl literatury (czechiso) je od Ing. Martínka, přičemž anglická verze (englishiso) je jeho překladem s drobnými modifikacemi. Oproti normě jsou v něm určité odlišnosti, ale na FIT je dlouhodobě akceptován. Alternativně můžete využít styl od

Ing. Radima Loskota nebo od Ing. Radka Pyšného[1]. Alternativní styly obsahují určitá vylepšení, ale zatím nebyly řádně otestovány větším množstvím uživatelů. Lze je považovat za beta verze pro zájemce, kteří svoji práci chtějí mít dokonalou do detailů a neváhají si nastudovat detaily správného formátování citací, aby si mohli ověřit, že je vysázený výsledek v pořádku.

Makefile kromě překladu do PDF nabízí i další funkce:

- přejmenování souborů (viz níže),
- počítání normostran,
- spuštění vlny pro doplnění nezlomitelných mezer,
- sbalení výsledku pro odeslání vedoucímu ke kontrole (zkontrolujte, zda sbalí všechny Vámi přidané soubory, a případně doplňte).

Nezapomeňte, že vlna neřeší všechny nezlomitelné mezery. Vždy je třeba manuální kontrola, zda na konci řádku nezůstalo něco nevhodného – viz Internetová jazyková příručka[2].

Pozor na číslování stránek! Pokud má obsah 2 strany a na 2. jsou jen "Přílohy" a "Seznam příloh" (ale žádná příloha tam není), z nějakého důvodu se posune číslování stránek o 1 (obsah "nesedí"). Stejný efekt má, když je na 2. či 3. stránce obsahu jen "Literatura" a je možné, že tohoto problému lze dosáhnout i jinak. Řešení je několik (od úpravy obsahu, přes nastavení počítadla až po sofistikovanější metody). **Před odevzdáním proto vždy překontrolujte číslování stran!**

Doporučený postup práce se šablonou

1. **Zkontrolujte, zda máte aktuální verzi šablony.** Máte-li šablonu z předchozího roku, na stránkách fakulty již může být novější verze šablony s aktualizovanými informacemi, opravenými chybami apod.

2. **Zvolte si jazyk**, ve kterém budete psát svoji technickou zprávu (česky, slovensky nebo anglicky) a svoji volbu konzultujte s vedoucím práce (nebyla-li dohodnuta předem). Pokud Vámi zvoleným jazykem technické zprávy není čeština, nastavte příslušný parametr šablony v souboru projekt.tex (např.: `documentclass[english]{fitthesis}` a přeložte prohlášení a poděkování do angličtiny či slovenštiny.

3. **Přejmenujte soubory.** Po rozbalení je v šabloně soubor projekt.tex. Pokud jej přeložíte, vznikne PDF s technickou zprávou pojmenované projekt.pdf. Když vedoucímu více studentů pošle projekt.pdf ke kontrole, musí je pracně přejmenovávat. Proto je vždy vhodné tento soubor přejmenovat tak, aby obsahoval Váš login a (případně zkrácené) téma práce. Vyhněte se však použití mezer, diakritiky a speciálních znaků. Vhodný název tedy může být např.: "xlogin00-Cisteni-a-extrakce-textu.tex". K přejmenování můžete využít i přiložený Makefile:

   ```
   make rename NAME=xlogin00-Cisteni-a-extrakce-textu
   ```

[1] BP Ing. Radka Pyšného http://www.fit.vutbr.cz/study/DP/BP.php?id=7848
[2] Internetová jazyková příručka http://prirucka.ujc.cas.cz/?id=880

4. Vyplňte požadované položky v souboru, který byl původně pojmenován projekt.tex, tedy typ, rok (odevzdání), název práce, svoje jméno, ústav (dle zadání), tituly a jméno vedoucího, abstrakt, klíčová slova a další formální náležitosti.

5. Nahraďte obsah souborů s kapitolami práce, literaturou a přílohami obsahem svojí technické zprávy. Jednotlivé přílohy či kapitoly práce může být výhodné uložit do samostatných souborů – rozhodnete-li se pro toto řešení, je doporučeno zachovat konvenci pro názvy souborů, přičemž za číslem bude následovat název kapitoly.

6. Nepotřebujete-li přílohy, zakomentujte příslušnou část v projekt.tex a příslušný soubor vyprázdněte či smažte. Nesnažte se prosím vymyslet nějakou neúčelnou přílohu jen proto, aby daný soubor bylo čím naplnit. Vhodnou přílohou může být obsah přiloženého paměťového média.

7. Nascanované zadání uložte do souboru zadani.pdf a povolte jeho vložení do práce parametrem šablony v projekt.tex (`documentclass[zadani]{fitthesis}`).

8. Nechcete-li odkazy tisknout barevně (tedy červený obsah – bez konzultace s vedoucím nedoporučuji), budete pro tisk vytvářet druhé PDF s tím, že nastavíte parametr šablony pro tisk: (`documentclass[zadani,print]{fitthesis}`). Barevné logo se nesmí tisknout černobíle!

9. Vzor desek, do kterých bude práce vyvázána, si vygenerujte v informačním systému fakulty u zadání. Pro disertační práci lze zapnout parametrem v šabloně (více naleznete v souboru fitthesis.cls).

10. Nezapomeňte, že zdrojové soubory i (obě verze) PDF musíte odevzdat na CD či jiném médiu přiloženém k technické zprávě.

Pokyny pro oboustranný tisk

- Zapíná se parametrem šablony: `\documentclass[twoside]{fitthesis}`
- Po vytištění oboustranného listu zkontrolujte, zda je při prosvícení sazební obrazec na obou stranách na stejné pozici. Méně kvalitní tiskárny s duplexní jednotkou mají často posun o 1–3 mm. Toto může být u některých tiskáren řešitelné tak, že vytisknete nejprve liché stránky, pak je dáte do stejného zásobníku a vytisknete sudé.
- Za titulním listem, obsahem, literaturou, úvodním listem příloh, seznamem příloh a případnými dalšími seznamy je třeba nechat volnou stránku, aby následující část začínala na liché stránce (\cleardoublepage).
- Konečný výsledek je nutné pečlivě překontrolovat.

Užitečné nástroje

Následující seznam není výčtem všech využitelných nástrojů. Máte-li vyzkoušený osvědčený nástroj, neváhejte jej využít. Pokud však nevíte, který nástroj si zvolit, můžete zvážit některý z následujících:

MikTeX LaTeX pro Windows – distribuce s jednoduchou instalací a vynikající automatizací stahování balíčků.

TeXstudio Přenositelné opensource GUI pro LaTeX. Ctrl+klik umožňuje přepínat mezi zdrojovým textem a PDF. Má integrovanou kontrolu pravopisu, zvýraznění syntaxe apod. Pro jeho využití je nejprve potřeba nainstalovat MikTeX.

JabRef Pěkný a jednoduchý program v Javě pro správu souborů s bibliografií (literaturou). Není potřeba se nic učit – poskytuje jednoduché okno a formulář pro editaci položek.

InkScape Přenositelný opensource editor vektorové grafiky (SVG i PDF). Vynikající nástroj pro tvorbu obrázků do odborného textu. Jeho ovládnutí je obtížnější, ale výsledky stojí za to.

GIT Vynikající pro týmovou spolupráci na projektech, ale může výrazně pomoci i jednomu autorovi. Umožňuje jednoduché verzování, zálohování a přenášení mezi více počítači.

Overleaf Online nástroj pro LaTeX. Přímo zobrazuje náhled a umožňuje jednoduchou spolupráci (vedoucí může průběžně sledovat psaní práce), vyhledávání ve zdrojovém textu kliknutím do PDF, kontrolu pravopisu apod. Zdarma jej však lze využít pouze s určitými omezeními (někomu stačí na disertaci, jiný na ně může narazit i při psaní bakalářské práce) a pro dlouhé texty je pomalejší.

Užitečné balíčky pro LaTeX

Studenti při sazbě textu často řeší stejné problémy. Některé z nich lze vyřešit následujícími balíčky pro LaTeX:

- `amsmath` – rozšířené možnosti sazby rovnic,
- `float, afterpage, placeins` – úprava umístění obrázků,
- `fancyvrb, alltt` – úpravy vlastností prostředí Verbatim,
- `makecell` – rozšíření možností tabulek,
- `pdflscape, rotating` – natočení stránky o 90 stupňů (pro obrázek či tabulku),
- `hyphenat` – úpravy dělení slov,
- `picture, epic, eepic` – přímé kreslení obrázků.

Některé balíčky jsou využity přímo v šabloně (v dolní části souboru fitthesis.cls). Nahlédnutí do jejich dokumentace může být rovněž užitečné.

Sloupec tabulky zarovnaný vlevo s pevnou šířkou je v šabloně definovaný “L” (používá se jako “p”).