FH Vorarlberg
University of Applied Sciences

Title

Subtitle

Master Thesis
Submitted in Fulfillment of the Degree

Master of Science in Engineering, MSc

Vorarlberg University of Applied Sciences
Master's in Mechatronics

Support Vorarlberg University of Applied Sciences

Dr.-Ing. Finck Steffen

Handed in by

Schwartze Nicolai, BSc.

matrikelnummer

Dornbirn, 05.04.2020

Kurzreferat

Deutscher TITEL

Ein Kurzreferat macht die Relevanz der Arbeit sowie die innovativen Gedankengänge ersichtlich. Alleiniges Ziel ist es, in jeweils einem Absatz einen gerafften Überblick der Arbeit zu geben, so dass die Nutzer/innen entscheiden können, ob die vorliegende Arbeit für das eigene Forschungsvorhaben relevant ist oder nicht. Dementsprechend müssen darin die zentralen Abschnitte in neutraler, nicht wertender Perspektive beschrieben werden, vergleichbar einem Text über den Text von einem imaginierten Dritten.

Das Abstract muss für sich alleine verständlich sein. Es sollte zudem die zentralen Schlagwörter, die das Thema der Arbeit treffend umreißen, enthalten, um eine Indexierung in einer bibliographischen Referenzdatei zu erleichtern. Der Umfang von 1200 Anschlägen (d.h. Zeichen mit Leerzeichen; ca. 20 Zeilen) sollte nicht überschritten werden.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Abstract

English TITLE

The abstract must be relevant to the topic and show the innovative ideas of the work. It should summarize your research in order to inform the reader, and give him/her the choice of whether it should/not be selected for his/her work. It should be a factual and impartial presentation of the main ideas.

The abstract should be only one paragraph long. It must provide the required information so that it is not necessary to read the other parts or even the complete work. It must also include key words to outline the subject for indexing and for bibliographic reference.

The length should not be more than 1200 characters (i.e. symbols and spaces; ca. 20 lines).

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Contents

List of Tables

List of Algorithms

1 Introduction

Differential equations are a very powerful mathematical tool to describe the universe. From fluid dynamic and heat flow in mechanical engineering or electrodynamics and circuit-design in electrical engineering to the fluctuation of stock market prices and even the movement of astronomical objects - most dynamic processes can be formulated by them. But only a tiny fraction of them are actually analytically solvable. The rise of the computer paved the way for approximating the solution of a differential equation numerically. There exists and abundance of different solvers ranging from simple single step solvers like the Forward-Euler (FE) method over more complex multistep solvers like the Adams–Bashforth (AB) method to whole Finite Element Method (FEM) solver packages. Most of these solvers are prone for one kind differential equation, leveraging some special properties of the problem to be most efficient in time and error. But there are very few generalised methods that can solve many different types of equations. This lack of a universal solver is the main motivation of the present master thesis.

There is already a small, yet steadily growing research community interested in tying together the concepts of computational intelligence and numerical solver for differential equation. In chapter 2.2 a brief overview of related work done within the last 20 years is given. The main idea of all listed papers is to reformulate the original problem into an optimisation problem, which in turn is then solved using an evolutionary optimisation algorithm. The main focus in this thesis lies on finding approximating solutions for Partial Differential Equation (PDE)s. The results are then compared to the analytical solution as well as a state of the art FEM solver.

This master thesis tries to answer the following questions: Is it possible to improve the results obtained in other research papers by using a modern variant of the Differential Evolution (DE) optimisation algorithm? How well does this method stack up against classical FEM package for solving PDEs considering time and memory usage as well as numerical error? Is there a meaningful way to reduce the time consumption by making use of a parallel computation architecture?

2 State of the Art

This chapter provides an overview of the current state of the art in solving PDE. Included are the widely used FEM as well as heuristic optimisation methods. Further, an introduction to the DE framework is given, which provides the basis for the algorithm described in this thesis.

2.1 Finite Element Method

Currently the finite element method is the go-to approach to solve partial differential equations. The domain Ω on which the PDE is posed, is discretised into multiple smaller elements - as the name suggests. Thus, FEM counts to the category of meshed methods. The underlying solution function $u(\mathbf{x})$ to the PDE is then approximated by so called "basis-functions" $\Phi(\mathbf{x})$ limited to these test functions. This thesis uses the open-source Netgen/NGSolve FEM package (Schöberl, Lackner, and Hochsteger 2020).

The general steps taken to solve a PDE with an FEM solver are:

- Strong Form
  This is the standard formulation of the PDE with a Dirichlet boundary condition:

$$\begin{aligned} u, f, g : \Omega &\to \mathbf{R} \\ \mathbf{L}u(\mathbf{x}) &= f(\mathbf{x}) \text{ on } \Omega \\ u(\mathbf{x})|_{\partial\Omega} &= g(\mathbf{x}) \end{aligned} \tag{2.1}$$

- Weak Form
  This is equivalent to the strong form but written in an integral formulation.

$$\begin{aligned} \int_\Omega -(\nabla^T A \nabla) u(\mathbf{x}) v(\mathbf{x}) dV - \int_\Omega b^T \nabla u(\mathbf{x}) v(\mathbf{x}) dV \\ + \int_\Omega c u(\mathbf{x}) v(\mathbf{x}) dV = \int_\Omega f(\mathbf{x}) v(\mathbf{x}) dV \end{aligned} \tag{2.2}$$

  In this equation, the A, b and c correspond to the constant factors of the derivatives in strong form.

- Discretisation of Ω
  Create a mesh of finite elements that span the whole domain. Usually these are triangles.
- Basis functions
  Choose a basis function $\Phi(\mathbf{x})$ that can be used to approximate the solution $u(\mathbf{x}) \approx u_h(\mathbf{x}) = \sum_{i=1}^{N} u_i \Phi(\mathbf{x})$. A common choice are linear basis functions.
- Solution
  Solve the resulting linear system of equations to determine the factors u_i. In the simple case where only second order derivatives are used in the strong from, the resulting linear system looks like this:

$$\begin{aligned} \sum_{j=1}^{N} v_j \sum_{i=1}^{N} u_i a(\Phi_i, \Phi_j) &= \sum_{j=1}^{N} v_j L(\Phi_j) \\ \underbrace{\sum_{i=1}^{N} u_i a(\Phi_i, \Phi_j)}_{\mathbf{Au}} &= \underbrace{L(\Phi_j)}_{\mathbf{b}} \end{aligned} \tag{2.3}$$

Modern solvers include more complex and advanced techniques to further improve the solution error and the computation time. Some of the most important concepts that are also available in NGSolve are listed here.

- Static Condensation:
  Depending on the number of discrete elements, the $\mathbf{A}$ matrix can be very big. Inverting big matrices is very time consuming. Condensation reduces this dimensionality by exploiting the structure of $\mathbf{A}$.
- Adaptive Mesh Refinement:
  The accuracy of a FEM-approximated solution mainly depends on the granularity of the mesh. Finer meshes produce more accurate solutions, but the computation time takes longer. This trade-off can be overcome by a self-adaptive mesh. NGSolve implements that in an adaptive loop that executes:
  - Solve PDE (with coarse mesh)
  - Estimate Error (for every element)
  - Mark Elements (that have the greatest error)
  - Refine Elements (that were previously marked)
  - ↻ until degrees of freedom exceed a specified N
- Preconditioner:
  Instead of solving the $\mathbf{A}^{-1}$ exactly, this can also be approximated by a matrix that is similar to $\mathbf{A}^{-1}$. The actual invers can be iteratively approximated.

NGSolve implements multiple different preconditioners and it even allows to create your own method.

2.2 Computational Intelligence Methods

The research community interested in computational intelligence solvers for differential equations has been steadily growing over the past 20 years. This chapter summarises the most important works done in the general field of development and application of such statistical numerical solvers. The following table 2.1 gives a brief overview of these papers and sorts them by relevance.

In general, all of these papers from the table use the weighted residual method (WRM), or some variant of that concept, to transform their differential equation into an optimisation problem. This serves as the fitness function and is necessary to evaluate a possible candidate solution and perform the evolutionary selection. The WRM is described in chapter 3.1 with more detail.

In their paper Chaquet and Carmona 2019 describe an algorithm that approximates a solution with a linear combination of gaussian radial basis function (RBF) as kernels. The objective function can be seen in equation 2.4.

$$F(\hat{u}(\mathbf{x})) = \frac{\sum_{i=1}^{n_C} \xi(\mathbf{x}_i)||\mathbf{L}\mathbf{u}(\mathbf{x}_i) - \mathbf{f}(\mathbf{x}_i)||^2 + \phi \sum_{j=1}^{n_B} ||\mathbf{B}\mathbf{u}(\mathbf{x}_j) - \mathbf{g}(\mathbf{x}_j)||^2}{m(n_C + n_B)} \tag{2.4}$$

The limit of the sum n_C denote the number of inner collocation points within the boundary Ω, whereas n_B is the number of discrete points on the boundary $\partial\Omega$. The first term represents the differential equation itself where $\mathbf{L}$ is a differential operator and $\mathbf{f}(\mathbf{x})$ is the inhomogeneous part. Similar, the second term stands for the boundary condition, where $\mathbf{B}$ is the differential operator and $\mathbf{g}(\mathbf{x})$ is the value on the boundary. The multipliers ξ and ϕ are weighting factors for either the inner or the boundary term. The whole term is normalised with the number of collocation points. The parameter of the kernels are determined via a Covariance Matrix Adaption Evolution Strategy (CMA-ES). To further improve the solution, the evolutionary algorithm is coupled with a Downhill-Simplex (DS) method to carry out the local search. The authors can show empirically that the local search significantly improves the performance by testing the algorithm on a set of 32 differential equations.

Babaei 2013 takes a different approach. They approximate a solution using a partial Fourier series. The main advantage of this candidate representation is, that it is backed up with a solid foundation of convergence theory. The optimal parameters for the candidates are found using a Particle Swarm Optimisation (PSO) algorithm. The fitness function consists of two parts, one for the inner area (equation 2.5) and one for the boundary (equation 2.6). These are added together resulting in

$F(\hat{u}(\mathbf{x})) = WRF + PFV.$

$$WRF = \int_{\Omega} |\mathbf{W}(\mathbf{x})||\mathbf{R}(\mathbf{x})|dx \tag{2.5}$$

This is exactly the formulation of the WRM, where $\mathbf{R}$ is the residual and $\mathbf{W}$ is an arbitrary weighting function. Instead of using a sum of collocation points, the integral is evaluated using a numerical integration scheme.

$$PFV = WRF \cdot \sum_{m=1}^{n_B} K_m g_m \tag{2.6}$$

In the penalty function from equation 2.6 the g_m stands for the boundary condition and K_m are penalty multipliers. The concept of this penalty function originates from Rajeev S. and Krishnamoorthy C. S. 1992.

Sobester, Nair, and Keane 2008 tried a radical different approach to incorporate the boundary condition into the solution. They found, that using Genetic Programming (GP) for the inner domain is only effective if the algorithm does not have to consider the boundary. They split the solution into two parts where $u(\mathbf{x})_{GP}$ represents the solution for the inner domain and $u(\mathbf{x})_{RBF}$ ensures the boundary condition, as seen in equation 2.7.

$$\hat{u}(\mathbf{x}) = u(\mathbf{x})_{GP} + u(\mathbf{x})_{RBF} \tag{2.7}$$

After the GP step produced a trail solution according to the objective function 2.8, a linear combination of radial basis functions $u(\mathbf{x})_{RBF} = \sum_{i=1}^{n_b} \alpha_i \Phi(||\mathbf{x} - \mathbf{x}_{i+n_d}||)$ is determined, that ensures the boundray condition at all $\mathbf{x}_{i+n_d}$ points on $\partial\Omega$. Finding the parameters α_i can be formulated as a least squares problem.

$$F(\hat{u}(\mathbf{x})) = \sum_{i=1}^{n_C} ||\mathbf{L}\mathbf{u}(\mathbf{x}_i) - \mathbf{f}(\mathbf{x}_i)||^2 \tag{2.8}$$

In Chaquet and Carmona 2012 they use a simple self-adaptive Evolution Strategy (ES) to evolve the coefficients of a partial Fourier series. The fitness function from equation 2.4 is reused. To reduce the search dimension (represented by the number of harmonics), they developed a scheme that only optimises one harmonic at a time and freezes the other coefficients. This scheme is based on the often observed principle that lower frequencies are more important in reconstructing a signal than higher ones. Albeit this concept might not be valid for all possible functions, it worked on all differential equations of their testbed.

Sadollah et al. 2017 compares three different optimisation algorithms to approximate differential equations: PSO, Harmony Search (HS) and Water Cycle Algorithm (WCA). The objective function for all algorithms is the same. They use the formulation in equation 2.5, where the weighting function is the same as the residual $|\mathbf{W}(\mathbf{x})| = |\mathbf{R}(\mathbf{x})| \implies WRF = \int_{\Omega} |\mathbf{R}(\mathbf{x})|^2 dx$. The integral is again approximated using a

numerical integration scheme. They find that the PSO is slightly better at producing low error solutions, however the WCA is better at satisfying the boundary condition.

Fateh et al. 2019 use a simple variant of DE where the candidates are extendet to matrices. They take discrete function value points within the domain to appximate a solution of elliptic partial differential equations. This is a radical brute force approach that results in a massive search space dimension. Yet the main advantage is, that the solution is not limited to a decomposition of kernel functions and thus even non-smooth functions can be approximated. Since this approach does not produce an analytical solution, the boundary condition can be easily incorporated into the fitness function by simply taking the sum of squared residuals at every grid point, as seen in equation 2.9.

$$F(\hat{u}(\mathbf{x})) = \sqrt{\sum_{i=0}^{n} R(\mathbf{x})_i^2} \tag{2.9}$$

Howard, Brezulianu, and Kolibal 2011 uses a GP scheme to find the solution to a specific set of simplified convection-diffusion equations. They also represent a candidate as function value points over the domain. The function between these points is interpolated. The fitness function is similar to equation 2.8 with the exception that the n_C points are not predetermined. These points are sampled randomly in the domain, thus allowing the algorithm to approximate the solution aside from the base points.

Panagant and Bureerat 2014 uses polynomials as a candidate representation. They did not specify the order or the type of the polynomial. Five different simple versions of DE were tested. Further, they introduce a so called DE-New that increases the population size after every generation. Their proposition is that greater population sizes are better at finding good solutions.

Tsoulos and Lagaris 2006 uses a Grammatical Evolution (GE) to find solutions to various differential equations. In contrary to GP, GE uses vectors instead of trees to represent the candidate string. The solution is evaluated as an analytical string, constructed of the functions *sin*, *cos*, *exp* and *log*, as well as all digits and all four basic arithmetic operations. The solution is measured by the same idea as displayed in equation 2.8. The algorithm was tested on multiple problems of Ordinary Differential Equation (ODE), system of ODEs and PDE. Only the results for ODEs were promising.

Mastorakis 2006 couples a Genetic Algorithm (GA) with a DS method for the local solution refinement. The candidates are represented as polynomials of the order 5 where the coefficients are optimised. The boundary condition is directly incorporated into the candidate, thus simplifying the objective function to equation 2.8. The focus here lays on unstable ODEs, that can not be solved with finite difference methods.

Kirstukas, Bryden, and Ashlock 2005 proposes a three-step procedure. The first step is time consuming and employs genetic programming techniques to find basis functions

that span the solution space. The second step is faster and uses a Gram–Schmidt algorithm to compute the basis function multipliers to develop a complete solution for a given set of boundary conditions. Using linear solver methods, a set of coefficients is found that produces a single function that both satisfies the differential equation and the boundary or initial conditions at all collocation points.

Howard and Roberts 2001 is one of the first advances in this field. They approximate a subset of the convection-diffusion equations with GP. Their main idea is to use a polynomial of variable length as the candidate representation. They use the same fitness function as already described in equation 2.7.

2.3 Differential Evolution

The differential evolution optimisation framework was first introduced in Storn and Price 1997. Due to its simple and flexible structure, it quickly became one of the most successful evolutionary algorithm. Over the years, several adaptions to the original framework have been proposed and some of them currently count to the best performing algorithms, as the 100-Digit Challenge at the GECCO 2019 (Suganthan 2020) shows.

The main DE framework consists of three necessary steps, that continuously update a population of possible solutions. The population can be interpreted as a matrix, where each vector is a possible candidate for the optimum of the fitness function $f : \mathbf{R}^n \to \mathbf{R}$. These steps are performed in a loop until some termination condition is reached. Each of these steps are controlled by a user-defined parameter:

- mutation:
  mutation strength parameter F;
  uses the information from within the population to create a trail vector v_i;
  this is done by scaling the difference between random vectors x_r in the population - hence the name *differential* evolution; a simple */rand/1* mutation operator can be seen in equation 2.10;

$$v_i = x_i + F(x_{r1} - x_{r2}) \tag{2.10}$$

- crossover:
  crossover probability parameter CR;
  randomly mix the information between the trail vector v_i and a random candidate from the population x_i to a new trail vector u_i; the binomial crossover from equation 2.11 randomly takes elements from both vectors, where K is a random index to ensure that at least one element from the trail vector v_i is

taken;

$$u_{ij} = \begin{cases} v_{ij}, & \text{if } j = K \vee rand[0,1] \leq CR \\ x_{ij}, & \text{otherwise} \end{cases} \tag{2.11}$$

- selection:
  population size N;
  replace the old candidate x_i if the trail candidate u_i is better (according to the fitness function);

In modern DE variants, these parameters are self-adapted during the evolutionary process. This means that the algorithms can balance out between exploration of the search-space and exploitation of promising locations.

A prominent example of a modern DE with self-adaption is JADE, which was developed by Zhang and Sanderson 2009. The adaption is performed by taking successful F and CR parameter of the last generation into account. If a certain setting is successful in generating better candidates, newly selected F and CR tend towards that setting. The pseudocode is represented in the appendix A.1.

This idea was later refined by Tanabe and A. Fukunaga 2013. They propose a similar self-adaptive scheme but extend the "memory" for good F and CR parameters over multiple generations. This idea should improve the robustness as compared to JADE. The pseudocode in appendix A.2 shows the outline of this so called SHADE algorithm.

The latest iteration of SHADE is called L-SHADE (Tanabe and A. S. Fukunaga 2014), which further improves the performance by including a deterministic adaptive concept for the population size. At first, L-SHADE starts with a big population size, and reduces the number of individuals in a linear fashion by deleting bad candidates. This has the effect of reducing the number of unnecessary function evaluation. The code is displayed in the appendix A.3.

Paper	Algorithm	Coding	Problems
Chaquet and Carmona 2019	CMA-ES (global); DS (local)	linear combination of Guassian kernals	testbench of ODEs system of ODEs and PDEs
Babaei 2013	PSO	partial sum of Fourier series	integro-differential equation systme of linear ODEs Brachistochrone nonlinear Bernoulli
Sobester, Nair, and Keane 2008	GP and RBF-NN	algebraic term for inner; RBF for boundary	Elliptic PDEs
Chaquet and Carmona 2012	self-adaptive ES	partial sum of Fourier series	testbench of ODEs system of ODEs and PDEs
Sadollah et al. 2017	PSO HS WCA	partial sum of Fourier series	singular BVP
Fateh et al. 2019	DE	function value gird	Elliptic PDEs
Howard, Brezulianu, and Kolibal 2011	GP	function value grid	convection–diffusion equation at different Peclet numbers
Panagant and Bureerat 2014	DE	polynomial of unspecified order	set of 6 different PDEs
Tsoulos and Lagaris 2006	GE	algebraic term	set of ODEs system of ODEs and PDEs
Mastorakis 2006	GA (global); DS (local)	5th order polynomial	unstable ODEs
Kirstukas, Bryden, and Ashlock 2005	GP	algebraic expression	heating of thin rod heating by current
Howard and Roberts 2001	GP	polynomial of arbitrary lengt	one-dimensional steady-state model of convection-diffusion equation

Table 2.1: Literature research on the general topic of stochastic slovers and their application

3 Problem Definition

3.1 Optimisation Problem

introduce notation: $u(\mathbf{x})$, $\hat{u}(\mathbf{x})$, $F(\hat{u}(\mathbf{x}))$

There is no easy way to approach this optimisation problem. It can not be transformed into a least squares problem, also calculating the gradient is not an option, thus justifying the usage of a heuristic algorithm.

3.2 Fitness Function

The Fitness function is formulated as ...

3.3 Candidate Representation

4 Algorithm

5 Experiment Results

5.1 Experiment 1

5.1.1 Hypotheses

5.1.2 Experiment Design

5.1.3 Result

5.1.4 Interpretation

6 Limitations

6.1 Poisson 2D on Unit Square

Sensitive towards the boundary condition:
Chaquet PDE8 as described in dissertation vs. same PDE but with 0 on boundary

7 Conclusion

8 Summary and Further Work

Advantage: Can be applied to nearly any PDE without restrictions to a specific problem set Problematic: Might be applied to a problem although, there is an algorithm that is faster and has better convergence

Acronyms

AB Adams–Bashforth. 1

CMA-ES Covariance Matrix Adaption Evolution Strategy. 4, 8

DE Differential Evolution. 1, 2, 5–8
DS Downhill-Simplex. 4, 6, 8

ES Evolution Strategy. 5, 8

FE Forward-Euler. 1
FEM Finite Element Method. 1, 2

GA Genetic Algorithm. 6, 8
GE Grammatical Evolution. 5, 8
GP Genetic Programming. 4–6, 8

HS Harmony Search. 5, 8

ODE Ordinary Differential Equation. 6

PDE Partial Differential Equation. 1, 2, 6
PSO Particle Swarm Optimisation. 4, 5, 8

RBF radial basis function. 3

WCA Water Cycle Algorithm. 5, 8
WRM weighted residual method. 3, 4

Bibliography

Babaei, M. (July 1, 2013): "A general approach to approximate solutions of nonlinear differential equations using particle swarm optimization". In: *Applied Soft Computing* 13.7, pp. 3354–3365. ISSN: 1568-4946. DOI: 10.1016/j.asoc.2013.02.005. URL: http://www.sciencedirect.com/science/article/pii/S1568494613000598 (visited on 02/27/2020) (cit. on pp. 4, 9).

Chaquet, Jose M. and Carmona, Enrique J. (Sept. 1, 2012): "Solving differential equations with Fourier series and Evolution Strategies". In: *Applied Soft Computing* 12.9, pp. 3051–3062. ISSN: 1568-4946. DOI: 10.1016/j.asoc.2012.05.014. URL: http://www.sciencedirect.com/science/article/pii/S1568494612002505 (visited on 03/02/2020) (cit. on pp. 5, 9).

Chaquet, Jose M. and Carmona, Enrique J. (Mar. 1, 2019): "Using Covariance Matrix Adaptation Evolution Strategies for solving different types of differential equations". In: *Soft Computing* 23.5, pp. 1643–1666. ISSN: 1433-7479. DOI: 10.1007/s00500-017-2888-9. URL: https://doi.org/10.1007/s00500-017-2888-9 (visited on 03/02/2020) (cit. on pp. 4, 9).

Fateh, Muhammad Faisal et al. (Oct. 1, 2019): "Differential evolution based computation intelligence solver for elliptic partial differential equations". In: *Frontiers of Information Technology & Electronic Engineering* 20.10, pp. 1445–1456. ISSN: 2095-9230. DOI: 10.1631/FITEE.1900221. URL: https://doi.org/10.1631/FITEE.1900221 (visited on 02/11/2020) (cit. on pp. 6, 9).

Howard, Daniel; Brezulianu, Adrian, and Kolibal, Joseph (Jan. 1, 2011): "Genetic programming of the stochastic interpolation framework: convection–diffusion equation". In: *Soft Computing* 15.1, pp. 71–78. ISSN: 1433-7479. DOI: 10.1007/s00500-009-0520-3. URL: https://doi.org/10.1007/s00500-009-0520-3 (visited on 03/14/2020) (cit. on pp. 6, 9).

Howard, Daniel and Roberts, Simon C. (July 7, 2001): "Genetic Programming solution of the convection-diffusion equation". In: *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*. GECCO'01. San Francisco, California: Morgan Kaufmann Publishers Inc., pp. 34–41. ISBN: 978-1-55860-774-3. (Visited on 02/27/2020) (cit. on pp. 7, 9).

Kirstukas, Steven J.; Bryden, Kenneth M., and Ashlock, Daniel A. (June 1, 2005): "A hybrid genetic programming approach for the analytical solution of differential equations". In: *International Journal of General Systems* 34.3. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/03081070500065676, pp. 279–299. ISSN: 0308-1079. DOI: 10.1080/03081070500065676. URL: https://doi.org/10.1080/03081070500065676 (visited on 03/02/2020) (cit. on pp. 6, 9).

Mastorakis, Nikos E (Aug. 21, 2006): "Unstable Ordinary Differential Equations: Solution via Genetic Algorithms and the method of Nelder-Mead". In: WSEAS Int. Conf. on Systems Theory & Scientific Computation. Elounda, Greece, p. 7. URL: https://www.researchgate.net/profile/Nikos_Mastorakis2/publication/261859052_Unstable_ordinary_differential_equations_Solution_via_genetic_algorithms_and_the_method_of_Nelder-Mead/links/573b254e08ae9f741b2d7853.pdf (visited on 02/19/2020) (cit. on pp. 6, 9).

Panagant, Natee and Bureerat, Sujin (2014): "Solving Partial Differential Equations Using a New Differential Evolution Algorithm". In: *Mathematical Problems in Engineering* 2014. ISSN: 1024-123X Library Catalog: www.hindawi.com Pages: e747490 Publisher: Hindawi Volume: 2014. DOI: https://doi.org/10.1155/2014/747490. URL: https://www.hindawi.com/journals/mpe/2014/747490/ (visited on 02/27/2020) (cit. on pp. 6, 9).

Rajeev S. and Krishnamoorthy C. S. (May 1, 1992): "Discrete Optimization of Structures Using Genetic Algorithms". In: *Journal of Structural Engineering* 118.5. Publisher: American Society of Civil Engineers, pp. 1233–1250. DOI: 10.1061/(ASCE)0733-9445(1992)118:5(1233). URL: https://ascelibrary.org/doi/abs/10.1061/(ASCE)0733-9445(1992)118:5(1233) (visited on 03/23/2020) (cit. on p. 5).

Sadollah, Ali et al. (July 4, 2017): "Metaheuristic optimisation methods for approximate solving of singular boundary value problems". In: *Journal of Experimental & Theoretical Artificial Intelligence* 29.4. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/0952813X.2016.1259271, pp. 823–842. ISSN: 0952-813X. DOI: 10.1080/0952813X.2016.1259271. URL: https://doi.org/10.1080/0952813X.2016.1259271 (visited on 03/02/2020) (cit. on pp. 5, 9).

Schöberl, Joachim; Lackner, Christopher, and Hochsteger, Matthias (Apr. 2, 2020): *NGSolve/ngsolve*. original-date: 2017-07-18T08:47:19Z. URL: https://github.com/NGSolve/ngsolve (visited on 04/02/2020) (cit. on p. 2).

Sobester, AndrÁs; Nair, Prasanth B., and Keane, Andy J. (Aug. 2008): "Genetic Programming Approaches for Solving Elliptic Partial Differential Equations". In: *IEEE Transactions on Evolutionary Computation* 12.4. Conference Name: IEEE Transactions on Evolutionary Computation, pp. 469–478. ISSN: 1941-0026. DOI: 10.1109/TEVC.2007.908467 (cit. on pp. 5, 9).

Storn, Rainer and Price, Kenneth (Dec. 1, 1997): "Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces". In: *Journal of Global Optimization* 11.4, pp. 341–359. ISSN: 1573-2916. DOI: 10.1023/A:1008202821328. URL: https://doi.org/10.1023/A:1008202821328 (visited on 04/05/2019) (cit. on p. 7).

Suganthan, Ponnuthurai Nagaratnam (Jan. 22, 2020): *P-N-Suganthan/CEC2019*. original-date: 2019-07-01T11:46:13Z. URL: https://github.com/P-N-Suganthan/CEC2019 (visited on 03/20/2020) (cit. on p. 7).

Tanabe, Ryoji and Fukunaga, Alex (June 2013): "Success-history based parameter adaptation for Differential Evolution". In: *2013 IEEE Congress on Evolutionary Computation*. 2013 IEEE Congress on Evolutionary Computation. ISSN: 1941-0026, pp. 71–78. DOI: 10.1109/CEC.2013.6557555 (cit. on p. 8).

Tanabe, Ryoji and Fukunaga, Alex S. (July 2014): "Improving the search performance of SHADE using linear population size reduction". In: *2014 IEEE Congress on Evolutionary Computation (CEC)*. 2014 IEEE Congress on Evolutionary Computation (CEC). ISSN: 1941-0026, pp. 1658–1665. DOI: 10.1109/CEC.2014.6900380 (cit. on p. 8).

Tsoulos, I. G. and Lagaris, I. E. (Mar. 1, 2006): "Solving differential equations with genetic programming". In: *Genetic Programming and Evolvable Machines* 7.1, pp. 33–54. ISSN: 1573-7632. DOI: 10.1007/s10710-006-7009-y. URL: https://doi.org/10.1007/s10710-006-7009-y (visited on 02/15/2020) (cit. on pp. 6, 9).

Zhang, Jingqiao and Sanderson, Arthur C. (Oct. 2009): "JADE: Adaptive Differential Evolution With Optional External Archive". In: *IEEE Transactions on Evolutionary Computation* 13.5. Conference Name: IEEE Transactions on Evolutionary Computation, pp. 945–958. ISSN: 1941-0026. DOI: 10.1109/TEVC.2009.2014613 (cit. on p. 8).

Appendices

A Differential Evolution Pseudocodes

A.1 JADE Pseudocode

Algorithm A.1: JADE Pseudocode

```
1  Function JADE(x_{g=0}, p, c, function, minError, maxGen):
2      fValue_{g=0} ← function(x_{g=0})
3      μ_CR ← 0.5
4      μ_F ← 0.5
5      A ← ∅
6      for g = 1 to G do
7          S_F ← ∅
8          S_CR ← ∅
9          for i = 1 to NP do
10             F_i ← randc_i(μ_F, 0.1)
11             v_i ← mutationCurrentToPBest1(x_{i,g}, A, fValue_g, F_i, p)
12             CR_i ← randn_i(μ_CR, 0.1)
13             u_i ← crossoverBIN(x_{i,g}, v_i, CR_i)
14             if function(x_{i,g}) ≥ function(u_{i,g}) then
15                 x_{i,g+1} ← x_{i,g}
16             end
17             else
18                 x_{i,g+1} ← u_{i,g}
19                 fValue_{i,g+1} ← function(u_{i,g})
20                 x_{i,g} → A
21                 CR_i → S_CR
22                 F_i → S_F
23             end
24         end
25         // resize A to size of x_g
26         if |A| > NP then
27             A ← A \ A_{rand_i}
28         end
29         μ_CR ← (1 − c) · μ_CR + c · arithmeticMean(S_CR)
30         μ_F ← (1 − c) · μ_F + c · lehmerMean(S_F)
31     end
```

A.2 SHADE Pseudocode

Algorithm A.2: SHADE Pseudocode

Function SHADE($\mathbf{x}_{G=0}$, p, H, $function$, $minError$, $maxGen$):
  $M_{CR} \leftarrow 0.5$; $M_F \leftarrow 0.5$; $A \leftarrow \emptyset$; $G \leftarrow 0$; $k \leftarrow 1$
  $fValue_{G=0} \leftarrow function(\mathbf{x}_{G=0})$
  while *termination condition not met* **do**
    $S_{CR} \leftarrow \emptyset$; $S_F \leftarrow \emptyset$
    for $i = 1$ *to* N **do**
      $r_i \leftarrow rand_{int}(1, H)$
      $CR_{i,G} \leftarrow randn_i(M_{CR,r_i}, 0.1)$; $F_{i,G} \leftarrow randc_i(M_{F,r_i}, 0.1)$
      $v_i \leftarrow mutationCurrentToPBest1(\mathbf{x}_{i,G}, A, fValue_G, F_i, p)$
      $u_i \leftarrow crossoverBIN(pop, v_i, CR)$
    end
    for $i = 1$ *to* N **do**
      if $function(u_{i,G}) \leq function(x_{i,G})$ **then**
        $x_{i,G+1} \leftarrow u_{i,G}$; $fValue_{i,G+1} \leftarrow function(\mathbf{u}_{i,G})$
      end
      else
        $x_{i,G+1} \leftarrow x_{i,G}$
      end
      if $function(u_{i,G}) < function(x_{i,G})$ **then**
        $x_{i,G} \rightarrow A$; $CR_{i,G} \rightarrow S_{CR}$; $F_{i,G} \rightarrow S_F$
      end
    end
    if $|A| > N$ **then**
      $A \leftarrow A \setminus A_{rand_i}$
    end
    if $S_{CR} \neq \emptyset \wedge S_F \neq \emptyset$ **then**
      $$M_{CR,k,G+1} = \begin{cases} arithmeticMean(S_{CR}) & \text{if } S_{CR} \neq \emptyset \\ M_{CR,k,G} & otherwise \end{cases}$$
      $$M_{F,k,G+1} = \begin{cases} lehmerMean(S_F) & \text{if } S_F \neq \emptyset \\ M_{F,k,G} & otherwise \end{cases}$$
      $k \leftarrow k + 1$
      if $k > H$ **then**
        $k \leftarrow 1$
      end
    end
    $G \leftarrow G + 1$
  end

A.3 L-SHADE Pseudocode

Algorithm A.3: L-SHADE Pseudocode

Function LSHADE($\mathbf{x}_{G=0}$, p, H, $function$, $minError$, $maxGen$):
  $M_{CR} \leftarrow 0.5$; $M_F \leftarrow 0.5$; $A \leftarrow \emptyset$; $G \leftarrow 0$; $k \leftarrow 1$
  $fValue_{G=0} \leftarrow function(\mathbf{x}_{G=0})$; $NG_{init} \leftarrow size(\mathbf{x}_{G=0})$; $NG_{min} = \lceil 1/p \rceil$
  while *termination condition not met* **do**
    $S_{CR} \leftarrow \emptyset$; $S_F \leftarrow \emptyset$
    for $i = 1$ *to* N **do**
      $r_i \leftarrow rand_{int}(1, H)$
      $CR_{i,G} \leftarrow randn_i(M_{CR,r_i}, 0.1)$; $F_{i,G} \leftarrow randc_i(M_{F,r_i}, 0.1)$
      $v_i \leftarrow mutationCurrentToPBest1(\mathbf{x}_{i,G}, A, fValue_G, F_i, p)$
      $u_i \leftarrow crossoverBIN(pop, v_i, CR)$
    end
    for $i = 1$ *to* N **do**
      if $function(u_{i,G}) \leq function(x_{i,G})$ **then**
        $x_{i,G+1} \leftarrow u_{i,G}$; $fValue_{i,G+1} \leftarrow function(\mathbf{u}_{i,G})$
      end
      else
        $x_{i,G+1} \leftarrow x_{i,G}$
      end
      if $function(u_{i,G}) < function(x_{i,G})$ **then**
        $x_{i,G} \rightarrow A$; $CR_{i,G} \rightarrow S_{CR}$; $F_{i,G} \rightarrow S_F$
      end
    end
    if $|A| > N$ **then**
      $A \leftarrow A \setminus A_{rand_i}$
    end
    if $S_{CR} \neq \emptyset \wedge S_F \neq \emptyset$ **then**
      $$M_{CR,k,G+1} = \begin{cases} arithmeticMean(S_{CR}) & \text{if } S_{CR} \neq \emptyset \\ M_{CR,k,G} & otherwise \end{cases}$$
      $$M_{F,k,G+1} = \begin{cases} lehmerMean(S_F) & \text{if } S_F \neq \emptyset \\ M_{F,k,G} & otherwise \end{cases}$$
      $k \leftarrow k + 1$
      if $k > H$ **then**
        $k \leftarrow 1$
      end
    end
    $\mathbf{x}_{G+1} \leftarrow popSizeRed(\mathbf{x}_{G+1}, fValue, G, maxGen, NG_{init}, NGmin)$
    $G \leftarrow G + 1$
  end

Statement of Affirmation

I hereby declare that this thesis was in all parts exclusively prepared on my own, without using other resources than those stated. The thoughts taken directly or indirectly from external sources are properly marked as such. This thesis or parts of it were not previously submitted to another academic institution and have also not yet been published.

Dornbirn, 05.04.2020 Schwartze Nicolai, BSc.