UNIVERSITY OF CALABRIA

Department of Computer Engineering, Modelling, Electronics and Systems Science

Ph.D. Thesis in

Information and Communication Technologies

Global Optimization, Ordinary Differential Equations and Infinity Computing

Marat S. Mukhametzhanov

Scientific Advisor
Prof. Yaroslav D. Sergeyev

Coordinator
Prof. Felice Crupi

Academic Year 2017–2018

In memory of my brother

Contents

Introduction

The main research topic studied in this work is global optimization—a field studying theory, methods, and implementation of models and strategies for solving multiextremal optimization problems. The rapidly growing interest to this area is explained by both the raising number of applied decision-making problems, that are described by multiextremal objective functions, and the significant recent development of advanced computer facilities.

Global optimization problems arise frequently in many real-life applications [88, 152, 157, 192, 215, 229]: in engineering, statistics, decision making, optimal control, machine learning, etc. A general global optimization problem requires to find a point x^* and the corresponding value $f(x^*)$ being the global (i.e., the deepest) minimum of a function $f(x)$ over an $N-$dimensional domain D, where $f(x)$ can be non-differentiable, multiextremal, hard to evaluate even in one point, and given as a "black box". Therefore, traditional local optimization methods [141, 145] cannot be used in this situation.

One of the important applied fields of efficient global optimization methods is the investigation of control systems under uncertain values of their parameters, in order to afford the desired safe functioning of a controllable object. For example, many important problems of robust control can be reduced to the problem of establishing the positiveness of multiextremal functions. This problem can be successfully solved with the availability of global optimization methods: it is sufficient to establish that the global minimum of a function describing the system is positive.

It can be noted in this context that not only multidimensional global optimization problems but also univariate problems of this kind arise frequently in different real-life applications, for instance, in engineering (see, e. g., [82, 93, 105, 113, 230]) and statistics (see, e. g., [26, 68, 72]). In particular, in structured low rank approximation (see, e. g., [74]), it can be necessary to solve perturbed problems as well as the original ones. The Lipschitz constant for the objective function in that case can be very large and the problem becomes very difficult to solve. Electrical engineering applications (see, e. g., [28, 38, 113, 190, 191, 183, 215]) can also require Lipschitz

global optimization, for example, for solving the minimal root problem (see, e. g., [114, 193, 189]). This kind of problems very often can be met in the multidimensional case, as well (see, e. g., [13, 120, 149, 151, 207]).

In the global optimization literature, there exist several ways to consider various global optimizations strategies (see, e. g., [144, 152, 154]). Such considerations are usually either 'problem-oriented' or 'methodology-oriented'. The problem-oriented point of view takes into account the problem information which can be used by a method during the search for the global solution. For example, in continuous global optimization, derivative-free or derivative-based methods can be considered depending on whether the objective function and constraints are differentiable and the derivatives can or cannot be computed or estimated.

The methodology-oriented point of view is more suitable for black-box problems and mainly based on the methodology applied for solving these problems. For example, global optimization algorithms can be divided into deterministic and stochastic. Assuming exact computations and arbitrarily long run time, deterministic methods ensure that after a finite time an approximation of a global minimizer will be found (within prescribed tolerances). Stochastic methods only offer a probabilistic guarantee of locating the global solution: their convergence theory usually states that the global minimum will be identified in an infinite time with probability one.

Among the vast group of deterministic algorithms for solving black-box global optimization problems the so-called direct (or derivative-free) search methods should be mentioned (see, e.g., [30, 33, 122, 158]). They are frequently used in engineering design (as, e.g., the DIRECT method, the response surface, or surrogate model methods, pattern search methods, etc.; see [54] for details). Black-box global optimization techniques based on an adaptive sampling and partition of the domain D are also widely used in practice (see, e.g., [89, 151, 152, 189, 216]).

Adaptive stochastic search strategies are mainly based on random sampling in the feasible set. Such techniques as adaptive random search, simulated annealing, evolution and genetic algorithms, tabu search, etc., can be cited here (see [54, 134, 147, 157, 224] for details). Stochastic approaches can often deal with the described black-box problems in a simpler manner than the deterministic algorithms (being also suitable for the problems where the evaluations of the functions are corrupted by noise). However, there can be difficulties with some of these methods, as well (e.g., in studying convergence properties of metaheuristics). Several restarts can also be involved, requiring more expensive functions evaluations. Moreover, solutions found by many stochastic algorithms (especially, by popular heuristic nature-inspired methods like evolutionary algorithms, simulated annealing, etc.; see, e.g.,

[97, 98, 147, 161, 224]) can be only local solutions to the problems, far from the global ones. This can preclude such methods from their usage in practice, when an accurate estimate of the global solution is required.

Obviously, the problem of a comparison of existing numerical algorithms for solving global optimization problems arises. The traditional way to do this is to use a collection of test functions and to show that on this collection a new method is better in some sense than its competitors. Then, a trade-off between the number of test functions, reliability of the comparison, and visibility of results arises. Clearly, a small number of test functions does not lead to a reliable comparison and a huge number of functions produces huge tables with a lot of data that sometimes are difficult for a fast visualization and an immediate comprehension.

Another difficulty in a convincing demonstration consists in the existence of methods having a completely different structure. A typical example is the principle trouble arising when one needs to test a deterministic method A with a stochastic algorithm B. The method A applied to a certain set of functions returns always the same results while the method B should be run several times and the results of these runs are always different. Consequently the method A is compared with some average characteristics of the method B.

In the literature, there exist some approaches for a graphical comparison of methods, as for example, operational characteristics (proposed in 1978 in [78], see also [214, 215]), subsequently generalized as performance profiles (see, e. g., [46]) and re-considered later as data profiles (see, e. g., [141]). Although they are very similar, performance profiles are mainly based on the relative behavior of the considered solvers on a chosen test set, while operational characteristics (and data profiles, which are quite close to operational characteristics) are more suitable for analyzing performance of a black-box optimization solver (or solvers) with respect to expensive function evaluations budget, independently of the behavior of the other involved methods on the same benchmark set.

All these techniques are, however, not always suitable for the comparison of methods of a different nature (for example, metaheuristics having a stochastic nature and deterministic Lipschitz methods), although an attempt of the usage of operational characteristics to study the behavior of a method with different parameters' values has been made in [214].

Today, a rapidly growing interest to modern supercomputers leads to the necessity of development of new algorithms and methods for working with novel supercomputing technologies (e.g., [169] for Infinity Computing, [19] for Quantum Computing, [2] for Biocomputing, etc.). In this work, the Infinity Computing, a novel methodology allowing one to work numerically with infinite and infinitesimal numbers, is studied.

This numeral system proposed in [172, 176, 180] is based on an infinite unit of measure expressed by the numeral ① called *grossone* and introduced as the number of elements of the set $\mathbb{N}$ of natural numbers (a clear difference with non-standard analysis can be seen immediately since non-standard infinite numbers are not connected to concrete infinite sets and have a purely symbolic character). Other symbols dealing with infinities and infinitesimals (∞, Cantor's ω, $\aleph_0, \aleph_1, ...$, etc.) are not used together with ①. Similarly, when the positional numeral system and the numeral 0 expressing zero had been introduced, symbols V, X, and other symbols from the Roman numeral system had not been involved.

In order to see the place of the new approach in the historical panorama of ideas dealing with infinite and infinitesimal, see [100, 125, 126, 128, 138, 174, 175, 185]. In particular, connections of the new approach with bijections are studied in [128] and metamathematical investigations on the theory and its non-contradictory can be found in [126]. The new methodology has been successfully used in several fields. We can mention numerical differentiation and optimization (see [39, 177, 233]), models for percolation and biological processes (see [91, 92, 179, 219]), hyperbolic geometry (see [129, 130]), fractals (see [91, 92, 171, 173, 179]), infinite series (see [96, 174, 178, 228]), lexicographic ordering, and Turing machines (see [175, 185, 186]), cellular automata (see [34, 35, 36]), etc.

It is well-known that in ill-conditioned systems, numerical methods can lead to incorrect results. However, it has been shown in [180], that in some cases ill-conditioning can be avoided using ① and the well-known Gauss method can be used without pivoting to solve the systems of linear equations. So, it can be very advantageous to use Infinity Computing to handle with ill-conditioning in optimization, as well. In this work, the advantages of applying the Infinity Computing are studied with respect to the traditional methodologies in order to handle with ill-conditioning occurred in optimization.

It should be noticed that the advantages of the Infinity Computing are not limited to working with ill-conditioning only. It has been shown in [181], that the numerical derivatives of a black-box function $y(x)$ can be calculated exactly using ①. Moreover, it has been also shown that the derivatives can be calculated exactly even if the function $y(x)$ is not given explicitly, but it is a solution to some ordinary differential equation. So, in this work, the advantages of the Infinity Computing are studied in the field of ordinary differential equations, as well.

The main aims of this research can be formulated as follows.

– Development of new powerful acceleration techniques in the framework of

univariate Lipschitz global optimization and new algorithms based on them.

– Theoretical and experimental study of the proposed algorithms.

– Development of new efficient methodologies allowing one to compare graphically global optimization algorithms of a different nature.

– Massive experimental comparison of several widely-used nature-inspired metaheuristic algorithms with several deterministic approaches using the proposed comparison techniques.

– Development of a new generator of multidimensional test problems with non-linear constraints, based on the GKLS generator of box-constraints test problems, allowing one to test different constrained global optimization algorithms.

– Application of the Infinity Computing in order to handle with ill-conditioning occurred in optimization. In particular, two different applications are considered: univariate Lipschitz global optimization problems and multidimensional convex non-smooth optimization problems.

– Development of new explicit and implicit methods for solving ordinary differential equations on the Infinity Computer and a theoretical study of their convergence properties.

Scientific novelty and practical importance of the present research consists of the following:

– Several new ideas that can be used to speed up the search in the framework of univariate Lipschitz global optimization algorithms are introduced. Proposed local tuning and local improvement techniques can lead to significant acceleration of the search and enjoy the following advantages:

  – the accelerated global optimization methods automatically realize a local behavior in the promising subregions without the necessity to stop the global optimization procedure;
  – all the evaluations of the objective function executed during the local phases are used also in the course of the global ones.

  It should be emphasized that proposed global optimization methods have a similar structure and a smart mixture of new and traditional computational steps leads to 22 different global optimization algorithms. All of them are studied and compared on several sets of

tests. Performed numerical experiments confirm the advantages of the proposed techniques.

– Two practical engineering problems are studied: finding the minimal root of a non-linear equation problem from electrical engineering and a sum of damped sinusoids from noisy data fitting. Numerical experiments on the presented classes of engineering problems confirmed the advantages of the proposed techniques, as well.

– Two efficient methodologies allowing one to compare global optimization algorithms of different nature, called "Operational zones" and "Aggregated operational zones", are proposed in this work. A massive experimental study of several widely-used nature-inspired metaheuristic and deterministic global optimization algorithms is performed on more than 1000 test problems with more than 1 000 000 runs. It is shown that this new graphical methodology for comparing global optimization methods of a different nature is quite representative. Almost all qualitative characteristics that can be studied from numerical tables can be also observed from operational zones. Moreover, the best, the worst, and average performances of stochastic methods can be easily found, as well.

– Collections of test problems are used usually in the framework of continuous constrained global optimization (see, e.g., [53]) due to absence of test classes and generators for such a type of problems. This work introduces a new generator of test problems with non-linear constraints, known global minimizers, and parameterizable difficulty, where both the objective function and constraints are continuously differentiable.

– Application of the Infinity Computing in order to handle of ill-conditioning in optimization shows promising results. In particular, we show in this work that several ill-conditioned problems in the traditional computational framework become well-conditioned if the Infinity Computing is applied. Presented techniques can be used in different fields, where ill-conditioning appears.

– Finally, several explicit numerical methods for solving ordinary differential equations are proposed. Theoretical convergence properties of the proposed methods are studied. It is shown that the methods of higher order can be used with the calculation of the derivatives exactly using the Infinity Computer. Experimental results show the competitive ability of the proposed methods with respect to the well-known Runge-Kutta and Taylor methods.

Obtained scientific results have been presented on 7 international conferences. Moreover, 8 papers have been published in the international journals and 1 paper has been submitted, 1 contribution to the book and 9 papers in proceedings of the international conferences have been also published.

This work consists of the introduction, 4 chapters, conclusion, references and 2 appendices.

The first Chapter is dedicated to univariate global optimization problems. Geometric and information frameworks for constructing global optimization algorithms are considered and several new ideas to speed up the search are proposed. A general scheme of univariate Lipschitz global optimization methods is presented. Convergence properties of the proposed algorithms are studied. Numerical experiments over several sets of test problems from the literature including two classes of practical engineering problems show the advantages of the proposed techniques.

The second Chapter is dedicated to a numerical comparison of global optimization algorithms of different nature. First, box-constraints problems are considered. A traditional comparative analysis using test benchmarks is studied and new methodologies for the comparison are proposed for two classes of algorithms: deterministic and nature-inspired metaheuristic methods. A new generator of test problems with non-linear constraints called "Emmental-type GKLS-based generator of test problems" is proposed.

The third Chapter is related to handling with the ill-conditioning in optimization using numerical infinities and infinitesimals. First, the Infinity Computing methodology is introduced very briefly. Then, univariate Lipschitz global optimization problems are considered in geometric and information frameworks for constructing global optimization algorithms. Finally, a multidimensional variable metric method is considered in the Infinity Computing framework, as well. Experimental results on the software simulator of the Infinity Computer confirm theoretical analysis.

The fourth Chapter is dedicated to numerical solution of ordinary differential equations on the Infinity Computer. The Infinity Computer studied in the third Chapter is applied to numerical methods for solving initial value problems. Several explicit methods are introduced. Properties of the proposed methods are studied theoretically. Finally, it is shown that an experimental study substantiates the obtained theoretical results.

Finally, some conclusion remarks are provided. Classes of test problems used during the work are described in Appendix.

Chapter 1

Univariate Lipschitz Global Optimization

In global optimization, it is necessary to find the global minimum f^* and the respective minimizer x^* of the objective function $f(x)$ over a set D, i. e.,

$$f^* = f(x^*) = \min f(x), \quad x \in D \subset \mathbb{R}^N, \tag{1.1}$$

where D is a bounded set (often, an N-dimensional hyperinterval is considered). In this Chapter, black-box Lipschitz global optimization problems are considered in their univariate statement, i.e., $N = 1$ in (1.1). Problems of this kind attract a great attention of the global optimization community. This happens because, first, there exists a huge number of real-life applications where it is necessary to solve univariate global optimization problems (see, e. g., [25, 26, 28, 37, 68, 72, 74, 83, 109, 155, 170, 193, 203, 166, 187, 189, 230, 234]). This kind of problems is often encountered in scientific and engineering applications (see, e. g., [82, 93, 105, 114, 113, 123, 152, 193, 166, 189, 214, 215]), and, in particular, in electrical engineering optimization problems (see, e. g., [37, 38, 183, 189, 215]). On the other hand, it is important to study one-dimensional methods because they can be successfully generalized in several ways. For instance, they can be extended to the multi-dimensional case by numerous schemes (see, for example, one-point based, diagonal, simplicial, space-filling curves, and other popular approaches in [32, 54, 88, 120, 149, 150, 151, 152, 168, 189, 207, 213, 215]). Another possible generalization consists of developing methods for solving problems where the first derivative of the objective function satisfies also the Lipschitz condition with an unknown constant (see, e. g., [69, 111, 119, 190, 191, 167, 189, 215]).

In the seventies of the XXth century two algorithms for solving the above mentioned problems have been proposed in [155, 213]. The first method was introduced by Piyavskij (see also [210]) by using geometric ideas (based on

the Lipschitz condition) and an a priori given overestimate of the Lipschitz constant for the objective function. The method [155] constructs a piecewise linear auxiliary function, being a minorant for the objective function, that is adaptively improved during the search. The latter algorithm [212, 213] was introduced by Strongin who developed a statistical model that allowed him to calculate probabilities of locating global minimizers within each of the subintervals of the search interval taken into consideration. Moreover, this model provided a dynamically computed estimate of the Lipschitz constant during the process of optimization. Both the methods became sources of multiple generalizations and improvements (see, e. g., [49, 149, 151, 152, 189, 207, 215, 232]) giving rise to classes of *geometric* and *information* global optimization methods.

Very often in global optimization (see, e. g., [54, 88, 152, 189, 215, 229]) local techniques are used to accelerate the global search and frequently global and local searches are realized by different methods having completely alien structures. Such a combination introduces at least two inconveniences. First, evaluations of the objective function (called hereinafter *trials*) executed by a local search procedure are not used usually in the subsequent phases of the global search or, at least, results of only some of these trials (for instance, the current best found value) are used and the other ones are not taken into consideration. Second, there arises the necessity to introduce both a rule that stops the global phase and starts the local one and a rule that stops the local phase and decides whether it is necessary to re-start the global search. Clearly, a premature stop of a global phase of the search can lead to the loss of the global solution while a late stop of the global phase can slow down the search.

In this work, both frameworks, geometric and information, are taken into consideration and a number of derivative-free techniques that were proposed to accelerate the global search are studied and compared.

1.1 Acceleration techniques in Lipschitz global optimization

Many of the algorithms of both deterministic and stochastic types have a similar structure. Hence, a number of general frameworks for describing computational schemes of global optimization methods and providing their convergence conditions in a unified manner have been proposed. The "Divide-the-Best" approach DBA (see [168, 189, 194, 202]), which generalizes both the schemes of adaptive partition [152] and characteristic [80, 189, 215] algo-

rithms, can be successfully used for describing and studying numerical global optimization methods.

In the DBA scheme, given a set of the method parameters, an adaptive partition of the admissible region D from (1.1) into subsets D_i^k is considered at each iteration k. The 'merit' (called characteristic) R_i of each subset for performing a subsequent, more detailed, investigation is estimated on the basis of the obtained information about the objective function. The best (in some predefined sense) characteristic obtained over some subregion D_t^k corresponds to a higher possibility to find the global minimizer within D_t^k. Subregion D_t^k is, therefore, subdivided at the next iteration of the algorithm, thus, improving the current approximation of the solution to problem (1.1).

Efficient deterministic global optimization methods belonging to the DBA scheme (as surveyed, e. g., in [112]) can be developed in the framework of Lipschitz global optimization (LGO) working with the Lipschitz objective functions $f(x)$ in (1.1), i. e., with the functions $f(x)$ satisfying the following condition:

$$|f(x') - f(x'')| \leq L\|x' - x''\|, \quad x', x'' \in D, \tag{1.2}$$

where $0 < L < \infty$ is the Lipschitz constant (usually, unknown for black-box functions).

Condition (1.2) is realistic for many practical black-box problems and allows the solvers to obtain accurate global optimum estimates after performing a limited number of trials.

1.1.1 Local Tuning and Local Improvement techniques

The considered univariate global optimization problem can be formulated as follows:

$$f^* := f(x^*) = \min f(x), \ \ x \in [a, b], \tag{1.3}$$

where the function $f(x)$ satisfies the Lipschitz condition (1.2) over the interval $[a, b]$ with the Lipschitz constant $L, 0 < L < \infty$. It is supposed that the objective function $f(x)$ can be multiextremal, non-differentiable; black-box; with an unknown Lipschitz constant L; and evaluation of $f(x)$ even at one point is a time-consuming operation.

As mentioned above, the geometric and information frameworks are taken into consideration in this work. The original geometric and information methods, apart the origins of their models, have the following important difference. Piyavskij's method requires for its correct work an overestimate of the value L that usually is hard to get in practice. In contrast, the information method of Strongin adaptively estimates L during the search. As it was shown in [165, 166] for both the methods, these two strategies for obtaining

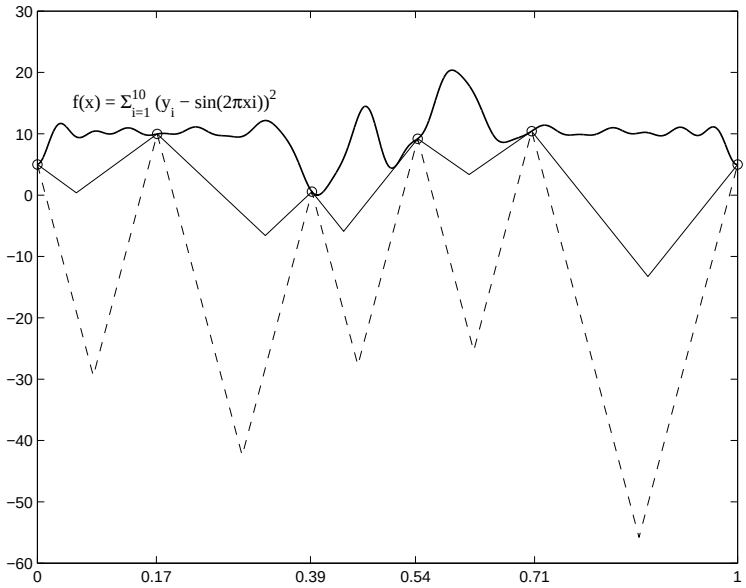


Figure 1.1: An auxiliary function (solid thin line) and a minorant function (dashed line) for a Lipschitz function $f(x)$ over $[a, b]$, constructed by using estimates of local Lipschitz constants and by using the global Lipschitz constant, respectively (trial values are circled).

the Lipschitz information can be substituted by the so-called "local tuning approach". In fact, the original methods of Piyavskij and Strongin use estimates of the global constant L during their work (the term "global" means that the same estimate is used over the whole interval $[a, b]$). However, the global estimate can provide a poor information about the behavior of the objective function $f(x)$ over every small subinterval $[x_{i-1}, x_i] \subset [a, b]$. In fact, when the local Lipschitz constant related to the interval $[x_{i-1}, x_i]$ is significantly smaller than the global constant L, then the methods using only this global constant or its estimate can work slowly over such an interval (see, e. g., [166, 189, 215]).

In Fig. 1.1, an example of the auxiliary function for a Lipschitz function $f(x)$ over $[a, b]$ constructed by using estimations of local Lipschitz constants over subintervals of $[a, b]$ is shown by a solid thin line; a minorant function for $f(x)$ over $[a, b]$ constructed by using an overestimate of the global Lipschitz constant is represented by a dashed line. Note that the former piecewise function estimates the behavior of $f(x)$ over $[a, b]$ more accurately than the latter one, especially over subintervals where the corresponding local Lipschitz constants are smaller than the global one.

The *local tuning technique* proposed in [165, 166] adaptively estimates

local Lipschitz constants at different subintervals of the search region during the course of the optimization process. Estimates l_i of local Lipschitz constants L_i are computed for each interval $[x_{i-1}, x_i]$, $i = 2, ..., k$, as follows:

$$l_i = r \cdot \max\{\lambda_i, \gamma_i, \xi\}, \tag{1.4}$$

where

$$\lambda_i = \max\{H_{i-1}, H_i, H_{i+1}\}, \quad i = 2, ..., k, \tag{1.5}$$

$$H_i = \frac{|z_i - z_{i-1}|}{x_i - x_{i-1}}, \ i = 2, ..., k, \tag{1.6}$$

$$H^k = \max\{H_i : i = 2, ..., k\}. \tag{1.7}$$

Here, $z_i = f(x_i), i = 1, ..., k$, i. e., values of the objective function calculated at the previous iterations at the trial points $x_i, i = 1, ..., k$, (when $i = 2$ and $i = k$ only H_2, H_3, and H_{k-1}, H_k, should be considered, respectively). The value γ_i is calculated as follows:

$$\gamma_i = H^k \frac{(x_i - x_{i-1})}{X^{max}}, \tag{1.8}$$

with H^k from (1.7) and

$$X^{max} = \max\{x_i - x_{i-1} : \ i = 2, ..., k\}. \tag{1.9}$$

Let us give an explanation of these formulae. The parameter $\xi > 0$ from (1.4) is a small number that is required for a correct work of the local tuning at initial steps of optimization, where it can happen that $\max\{\lambda_i, \gamma_i\} = 0$; $r > 1$ is the reliability parameter. The two components, λ_i and γ_i, are the main players in (1.4). They take into account, respectively, the local and the global information obtained during the previous iterations. When the interval $[x_{i-1}, x_i]$ is large, the local information represented by λ_i can be not reliable and the global part γ_i has a decisive influence on l_i thanks to (1.4) and (1.8). In this case $\gamma_i \to H^k$, namely, it tends to the estimate of the global Lipschitz constant L. In contrast, when $[x_{i-1}, x_i]$ is small, then the local information becomes relevant, the estimate γ_i is small for small intervals (see (1.8)), and the local component λ_i assumes the key role. Thus, the local tuning technique automatically balances the global and the local information available at the current iteration. It has been proved for a number of global optimization algorithms that the usage of the local tuning can accelerate the search significantly (see [110, 170, 193, 203, 166, 167, 189, 215]). This local tuning strategy will be called *"Maximum" Local Tuning* hereinafter.

Recently, a new local tuning strategy called hereinafter *"Additive" Local Tuning* has been proposed in [66, 67, 214] for certain information algorithms. It proposes to use the following additive convolution instead of (1.4):

$$l_i = r \cdot \max\{\frac{1}{2}(\lambda_i + \gamma_i), \xi\}, \tag{1.10}$$

where r, ξ, λ_i, and γ_i have the same meaning as in (1.4). The first numerical examples executed in [66, 214] have shown a very promising performance of the "Additive" Local Tuning. These results induced us to execute in the present research a broad experimental testing and a theoretical analysis of the "Additive" Local Tuning. In particular, geometric methods using this technique are proposed here (remind that the authors of [66, 214] have introduced it in the framework of information methods only). During our study some features suggesting a careful usage of this technique have been discovered, especially, in cases where it is applied to geometric global optimization methods.

In order to start our analysis of the "Additive" Local Tuning, let us remind (see, e. g., [152, 155, 189, 207, 213, 215]) that in both the geometric and the information univariate algorithms, an interval $[x_{t-1}, x_t]$ is chosen in a certain way at the $(k+1)$-th iteration of the optimization process and a new trial point x^{k+1}, where the $(k+1)$-th evaluation of $f(x)$ is executed, is computed as follows:

$$x^{k+1} = \frac{x_t + x_{t-1}}{2} - \frac{z_t - z_{t-1}}{2l_t}. \tag{1.11}$$

For a correct work of this kind of algorithms it is necessary that x^{k+1} is such that $x^{k+1} \in (x_{t-1}, x_t)$. It is easy to see that the necessary condition for this inclusion is $l_t > H_t$, where H_t is calculated following (1.6). Notice that l_t is obtained by using (1.10), where the sum of two addends plays the leading role. Since the estimate γ_i is calculated as shown in (1.8), it can be very small for small intervals, creating so the possibility of occurrence of the situation $l_t \leq H_t$ and, consequently, $x^{k+1} \notin (x_{t-1}, x_t)$. Obviously, by increasing the value of the parameter r this situation can be easily avoided and the method should be re-started. In fact, in information algorithms where $r \geq 2$ is usually used this risk is less pronounced while in geometric methods where $r > 1$ is applied it becomes more probable. On the other hand, it is well known in Lipschitz global optimization (see, e. g., [152, 189, 207, 215]) that increasing the parameter r can slow down the search. In order to understand better the functioning of the "Additive" Local Tuning, it is broadly tested together with other competitors.

The analysis provided above shows that the usage of the "Additive" Local Tuning can become tricky in some cases. In order to avoid the necessity to check the satisfaction of the condition $x^{k+1} \in (x_{t-1}, x_t)$ at each iteration, we propose a new strategy called the *"Maximum-Additive" Local Tuning* where, on the one hand, this condition is satisfied automatically and, on the other hand, advantages of both the local tuning techniques described above are incorporated in the unique strategy. This local tuning strategy calculates the estimate l_i of the local Lipschitz constants as follows:

$$l_i = r \cdot \max\{H_i, \frac{1}{2}(\lambda_i + \gamma_i), \xi\}, \tag{1.12}$$

where r, ξ, H_i, λ_i, and γ_i have the usual meaning. It can be seen from (1.12) that this strategy both maintains the additive character of the convolution and satisfies condition $l_i > H_i$. The latter condition provides that in case the interval $[x_{i-1}, x_i]$ is chosen for subdivision (i. e., $t := i$ is assigned), the new trial point x^{k+1} will belong to (x_{t-1}, x_t). Notice that in (1.12) the equal usage of the local and global estimate is applied. Obviously, a more general scheme similar to (1.10) and (1.12) can be used where $\frac{1}{2}$ is substituted by different weights for the estimates λ_i and γ_i, for example, as follows:

$$l_i = r \cdot max\{H_i, \frac{\lambda_i}{r} + \frac{r-1}{r}\gamma_i, \xi\}$$

(r, H_i, λ_i, γ_i, and ξ are as in (1.12)).

Let us now present another acceleration idea. It consists of the following observation related to global optimization problems with a fixed budget of possible evaluations of the objective function $f(x)$, i. e., when only, for instance, 100 or 1 000 000 evaluations of $f(x)$ are allowed. In these problems, it is necessary to obtain the best possible value of $f(x)$ as soon as possible. Suppose that f_k^* is the best value (the record value) obtained after k iterations. If a new value $f(x^{k+1}) < f_k^*$ has been obtained, then it can make sense to try to improve this value locally, instead of continuing the usual global search phase. As was already mentioned, traditional methods stop the global procedure and start a local descent: trials executed during this local phase are not then used by the global search since the local method has usually a completely different nature.

Here, we propose two *local improvement techniques*, the *"optimistic"* and the *"pessimistic"* one, that perform the local improvement within the global optimization scheme. The optimistic method alternates the local steps with the global ones and, if during the local descent a new promising local minimizer is not found, the global method stops when a local stopping rule is satisfied. The pessimistic strategy does the same until the satisfaction of

the required accuracy on the local phase and then switches to the global phase where the trials performed during the local phase are also taken into consideration.

All the methods described in this Chapter have a similar structure and belong to the class of "Divide the Best" global optimization algorithms introduced in [168] (see also [189]; for methods using the "Additive" Local Tuning this holds if the parameter r is such that $l_{i(k)} > rH_{i(k)}$ for all i and k). The algorithms differ in the following:

– methods are either geometric or information;

– methods differ in the way the Lipschitz information is used: an a priori estimate, a global estimate, and a local tuning;

– in cases where a local tuning is applied methods use 3 different strategies: Maximum, Additive, and Maximum-Additive;

– in cases where a local improvement is applied methods use either the optimistic or the pessimistic strategy.

Let us describe the General Scheme (GS) of the methods used in this work. A concrete algorithm will be obtained by specifying one of the possible implementations of Steps 2–4 in this (GS).

Step 0. *Initialization.* Execute first two trials at the points a and b, i. e., $x^1 := a$, $z^1 := f(a)$ and $x^2 := b$, $z^2 := f(b)$. Set the iteration counter $k := 2$.

Let $flag$ be the local improvement switch to alternate global search and local improvement procedures; set its initial value $flag := 0$. Let $i_{\min}$ be an index (being constantly updated during the search) of the current record point, i. e., $z_{i_{\min}} = f(x_{i_{\min}}) \leq f(x_i)$, $i = 1, \ldots, k$ (if the current minimal value is attained at several trial points, then the smallest index is accepted as $i_{\min}$).

Suppose that $k \geq 2$ iterations of the algorithm have already been executed. The iteration $k + 1$ consists of the following steps.

Step 1. *Reordering.* Reorder the points $x^1, \ldots, x^k$ (and the corresponding function values $z^1, \ldots, z^k$) of previous trials by subscripts so that

$$a = x_1 < \ldots < x_k = b, \quad z_i = f(x_i),\ i = 1, \ldots, k.$$

Step 2. *Estimates of the Lipschitz constant.* Calculate the current estimates l_i of the Lipschitz constant for each subinterval $[x_{i-1}, x_i]$, $i = 2, \ldots, k$, in one of the following ways.

Step 2.1. *A priori given estimate.* Take an a priori given estimate $\hat{L}$ of the Lipschitz constant for the whole interval $[a, b]$, i. e., set $l_i := \hat{L}$.

Step 2.2. *Global estimate.* Set $l_i := r \cdot \max\{H^k, \xi\}$, where r and ξ are two parameters with $r > 1$ and ξ sufficiently small, H^k is from (1.7).

Step 2.3. *"Maximum" Local Tuning.* Set l_i following (1.4).

Step 2.4. *"Additive" Local Tuning.* Set l_i following (1.10).

Step 2.5. *"Maximum-Additive" Local Tuning.* Set l_i following (1.12).

Step 3. *Calculation of characteristics.* Compute for each subinterval $[x_{i-1}, x_i]$, $i = 2, \ldots, k$, its characteristic R_i by using one of the following rules.

Step 3.1. *Geometric methods.*

$$R_i = \frac{z_i + z_{i-1}}{2} - l_i \frac{x_i - x_{i-1}}{2}.$$

Step 3.2. *Information methods.*

$$R_i = 2(z_i + z_{i-1}) - l_i(x_i - x_{i-1}) - \frac{(z_i - z_{i-1})^2}{l_i(x_i - x_{i-1})}.$$

Step 4. *Subinterval selection.* Determine subinterval $[x_{t-1}, x_t]$, $t = t(k)$, for performing the next trial by using one of the following rules.

Step 4.1. *Global phase.* Select the subinterval $[x_{t-1}, x_t]$ corresponding to the minimal characteristic, i. e., such that $t = \arg\min_{i=2,\ldots,k} R_i$.

Steps 4.2–4.3. *Local improvement.*

if $flag = 1$ **then** *(perform local improvement)*
  if $z^k = z_{i_{\min}}$, **then** $t = \arg\min\{R_i : i \in \{i_{\min} + 1, i_{\min}\}\}$;
  else alternate the choice of subinterval between $[x_{i_{\min}}, x_{i_{\min}+1}]$ and $[x_{i_{\min}-1}, x_{i_{\min}}]$ starting from the right subinterval $[x_{i_{\min}}, x_{i_{\min}+1}]$.
  end if
else $t = \arg\min_{i=2,\ldots,k} R_i$ *(do not perform local improvement at the current iteration).*
end if

The subsequent part of this Step differs for two local improvement techniques.

Step 4.2. *Pessimistic local improvement.*

if $flag = 1$ and

$$x_t - x_{t-1} \leq \delta, \tag{1.13}$$

where $\delta > 0$ is the local search accuracy,
then $t = \arg\min_{i=2,\ldots,k} R_i$ *(local improvement is not performed since the local search accuracy has been achieved).*

end if

Set $flag := NOT(flag)$ *(switch the local/global flag).*

Step 4.3. *Optimistic local improvement.*

Set $flag := NOT(flag)$ *(switch the local/global flag: the accuracy of local search is not separately checked in this strategy).*

Step 5. *Global stopping criterion.* **If**

$$x_t - x_{t-1} \leq \varepsilon, \tag{1.14}$$

where $\varepsilon > 0$ is a given accuracy of the global search, **then Stop** and take as an estimate of the global minimum f^* the value $f_k^* = \min_{i=1,\ldots,k}\{z_i\}$ obtained at a point $x_k^* = \arg\min_{i=1,\ldots,k}\{z_i\}$.

Otherwise, go to Step 6.

Step 6. *New trial.* Execute the next trial at the point x^{k+1} from (1.11): $z^{k+1} := f(x^{k+1})$. Increase the iteration counter $k := k + 1$, and go to Step 1.

All the Lipschitz global optimization methods considered in the Chapter are summarized in Table 1.1 from which concrete implementations of Steps 2–4 in the GS can be individuated. As shown experimentally in Subsection 1.1.2, the methods using an a priori given estimate of the Lipschitz constant or its global estimate lose, as a rule, in comparison with methods using local tuning techniques, in terms of the trials performed to approximate the global solutions to problems. Therefore, local improvement accelerations (Steps 4.2–4.3 of the GS) were implemented for methods using local tuning strategies only. In what follows, the methods from Table 1.1 are furthermore specified (for the methods known in the literature the respective references are provided).

Method	Step2					Step3		Step4		
	2.1	2.2	2.3	2.4	2.5	3.1	3.2	4.1	4.2	4.3
Geom-AL	+					+		+		
Geom-GL		+				+		+		
Geom-LTM			+			+		+		
Geom-LTA				+		+		+		
Geom-LTMA					+	+		+		
Geom-LTIMP			+			+			+	
Geom-LTIAP				+		+			+	
Geom-LTIMAP					+	+			+	
Geom-LTIMO			+			+				+
Geom-LTIAO				+		+				+
Geom-LTIMAO					+	+				+
Inf-AL	+						+	+		
Inf-GL		+					+	+		
Inf-LTM			+				+	+		
Inf-LTA				+			+	+		
Inf-LTMA					+		+	+		
Inf-LTIMP			+				+		+	
Inf-LTIAP				+			+		+	
Inf-LTIMAP					+		+		+	
Inf-LTIMO			+				+			+
Inf-LTIAO				+			+			+
Inf-LTIMAO					+		+			+

Table 1.1: Description of the considered methods, the signs "+" show a combination of implementations of Steps 2–4 in the GS for each method.

1. **Geom-AL**: Piyavskij's method with the a priori given Lipschitz constant (see [155, 210] and [189] for generalizations and discussions): GS with Step 2.1, Step 3.1 and Step 4.1.

2. **Geom-GL**: Geometric method with the global estimate of the Lipschitz constant (see [189]): GS with Step 2.2, Step 3.1 and Step 4.1.

3. **Geom-LTM**: Geometric method with the "Maximum" Local Tuning (see [166, 189, 215]): GS with Step 2.3, Step 3.1 and Step 4.1.

4. **Geom-LTA**: Geometric method with the "Additive" Local Tuning: GS with Step 2.4, Step 3.1 and Step 4.1.

5. **Geom-LTMA**: Geometric method with the "Maximum-Additive" Local Tuning: GS with Step 2.5, Step 3.1 and Step 4.1.

6. **Geom-LTIMP**: Geometric method with the "Maximum" Local Tuning and the pessimistic strategy of the local improvement (see [119, 189]): GS with Step 2.3, Step 3.1 and Step 4.2.

7. **Geom-LTIAP**: Geometric method with the "Additive" Local Tuning and the pessimistic strategy of the local improvement: GS with Step 2.4, Step 3.1 and Step 4.2.

8. **Geom-LTIMAP**: Geometric method with the "Maximum-Additive" Local Tuning and the pessimistic strategy of the local improvement: GS with Step 2.5, Step 3.1 and Step 4.2.

9. **Geom-LTIMO**: Geometric method with the "Maximum" Local Tuning and the optimistic strategy of the local improvement: GS with Step 2.3, Step 3.1 and Step 4.3.

10. **Geom-LTIAO**: Geometric method with the "Additive" Local Tuning and the optimistic strategy of the local improvement: GS with Step 2.4, Step 3.1 and Step 4.3.

11. **Geom-LTIMAO**: Geometric method with the "Maximum-Additive" Local Tuning and the optimistic strategy of the local improvement: GS with Step 2.5, Step 3.1 and Step 4.3.

12. **Inf-AL**: Information method with the a priori given Lipschitz constant (see [189]): GS with Step 2.1, Step 3.2 and Step 4.1.

13. **Inf-GL**: Strongin's information-statistical method with the global estimate of the Lipschitz constant (see [212, 213, 215]): GS with Step 2.2, Step 3.2 and Step 4.1.

14. **Inf-LTM**: Information method with the "Maximum" Local Tuning (see [165, 207, 215]): GS with Step 2.3, Step 3.2 and Step 4.1.

15. **Inf-LTA**: Information method with the "Additive" Local Tuning (see [66, 214]): GS with Step 2.4, Step 3.2 and Step 4.1.

16. **Inf-LTMA**: Information method with the "Maximum-Additive" Local Tuning: GS with Step 2.5, Step 3.2 and Step 4.1.

17. **Inf-LTIMP**: Information method with the "Maximum" Local Tuning and the pessimistic strategy of the local improvement [118, 207]: GS with Step 2.3, Step 3.2 and Step 4.2.

18. **Inf-LTIAP**: Information method with the "Additive" Local Tuning and the pessimistic strategy of the local improvement: GS with Step 2.4, Step 3.2 and Step 4.2.

19. **Inf-LTIMAP**: Information method with the "Maximum-Additive" Local Tuning and the pessimistic strategy of the local improvement: GS with Step 2.5, Step 3.2 and Step 4.2.

20. **Inf-LTIMO**: Information method with the "Maximum" Local Tuning and the optimistic strategy of the local improvement: GS with Step 2.3, Step 3.2 and Step 4.3.

21. **Inf-LTIAO**: Information method with the "Additive" Local Tuning and the optimistic strategy of the local improvement: GS with Step 2.4, Step 3.2 and Step 4.3.

22. **Inf-LTIMAO**: Information method with the "Maximum-Additive" Local Tuning and the optimistic strategy of the local improvement: GS with Step 2.5, Step 3.2 and Step 4.3.

1.1.2 Convergence study and experimental analysis

Let us spend a few words regarding convergence of the methods belonging to the GS. To do this we study an infinite trial sequence $\{x^k\}$ generated by an algorithm belonging to the general scheme GS for solving the problem (1.3), (1.2) with $\delta = 0$ from (1.13) and $\varepsilon = 0$ from (1.14).

Theorem 1.1. *Assume that the objective function $f(x)$ satisfies the Lipschitz condition* (1.2) *with a finite constant $L > 0$, and let x' be any limit point of $\{x^k\}$ generated by an algorithm belonging to the GS that does not use the "Additive" Local Tuning and works with one of the estimates* (1.4), (1.7), (1.12). *Then the following assertions hold:*

1. *if $x' \in (a, b)$ then convergence to x' is bilateral, i.e., there exist two infinite subsequences of $\{x^k\}$ converging to x' one from the left, the other from the right;*

2. *$f(x^k) \geq f(x')$, for all trial points x^k, $k \geq 1$;*

3. *if there exists another limit point $x'' \neq x'$, then $f(x'') = f(x')$;*

4. *if the function $f(x)$ has a finite number of local minima in $[a, b]$, then the point x' is locally optimal;*

5. (Sufficient conditions for convergence to a global minimizer). *Let x^* be a global minimizer of $f(x)$. If there exists an iteration number k^* such that for all $k > k^*$ the inequality*

$$l_{j(k)} > L_{j(k)} \tag{1.15}$$

holds, where $L_{j(k)}$ is the Lipschitz constant for the interval $[x_{j(k)-1}, x_{j(k)}]$ containing x^*, and $l_{j(k)}$ is its estimate. Then the set of limit points of the sequence $\{x^k\}$ coincides with the set of global minimizers of the function $f(x)$.

Proof. Since all the methods mentioned in the Theorem belong to the "Divide the Best" class of global optimization algorithms introduced in [168], the proofs of assertions 1–5 can be easily obtained as particular cases of the respective proofs in [168, 189]. □

Corollary 1.1. *Assertions* 1–5 *hold for methods belonging to the GS and using the "Additive" Local Tuning if the condition $l_{i(k)} > rH_{i(k)}$ is fulfilled for all i and k.*

Proof. Fulfillment of the condition $l_{i(k)} > rH_{i(k)}$ ensures that: (i) each new trial point x^{k+1} belongs to the interval (x_{t-1}, x_t) chosen for partitioning; (ii) the distances $x^{k+1} - x_{t-1}$ and $x_t - x^{k+1}$ are finite. The fulfillment of these two conditions implies that the methods belong to the class of "Divide the Best" global optimization algorithms and, therefore, proofs of assertions 1–5 can be easily obtained as particular cases of the respective proofs in [168, 189]. □

Notice that in practice, since both ε and δ assume finite positive values, methods using the optimistic local improvement can miss the global optimum and stop in the δ-neighborhood of a local minimizer (see Step 4 of the GS).

The next Theorem ensures existence of the values of the reliability parameter r satisfying condition (1.15), providing so that all global minimizers of $f(x)$ will be determined by the proposed methods that do not use the a priori known Lipschitz constant.

Theorem 1.2. *For any function $f(x)$ satisfying the Lipschitz condition* (1.2) *with $L < \infty$ and for methods belonging to the GS and using one of the estimates* (1.4), (1.7), (1.10), (1.12) *there exists a value r^* such that for all $r > r^*$ condition* (1.15) *holds.*

Proof. It follows from, (1.4), (1.7), (1.10), (1.12), and the finiteness of $\xi > 0$ that approximations of the Lipschitz constant l_i in the methods belonging to the GS are always positive. Since L in (1.2) is finite and any positive value

of the parameter r can be chosen in (1.4), (1.7), (1.10), (1.12), it follows that there exists an r^* such that condition (1.15) will be satisfied for all global minimizers for $r > r^*$. □

Experimental study of the described methods is presented below. Six series of numerical experiments were executed on the following three sets of test functions to compare 22 global optimization methods described previously:

1. the widely used set of 20 test functions from [83] reported in Appendix A;

2. 100 randomly generated Pintér's functions from [153];

3. 100 Shekel type test functions from [215].

Geometric and information methods with and without the local improvement techniques (optimistic and pessimistic) were tested in these experimental series. In the first two series of experiments, the accuracy of the global search was chosen as $\varepsilon = 10^{-5}(b-a)$, where $[a,b]$ is the search interval. The accuracy of the local search was set as $\delta = \varepsilon$ in the algorithms with the local improvement. Results of numerical experiments are reported in Tables 1.2–1.15 where the number of function trials executed until the satisfaction of the stopping rule is presented for each considered method (the best results for the methods within the same class are shown in bold).

The first series of numerical experiments was carried out with geometric and information algorithms without the local improvement on 20 test functions given in Appendix A. Parameters of the geometric methods Geom-AL, Geom-GL, Geom-LTM, Geom-LTA, and Geom-LTMA were chosen as follows. For the method Geom-AL, the estimates of the Lipschitz constants were computed as the maximum between the values calculated as relative differences on 10^{-7}-grid and the values given in [83]. For the methods Geom-GL, Geom-LTM, and Geom-LTMA, the reliability parameter $r = 1.1$ was used as recommended in [189]. The technical parameter $\xi = 10^{-8}$ was used for all the methods with the local tuning (Geom-LTM, Geom-LTA, and Geom-LTMA). For the method Geom-LTA, the parameter r was increased with the step equal to 0.1 starting from $r = 1.1$ until all 20 test problems were solved (i. e., for all the problems the algorithm stopped in the ε-neighborhood of a global minimizer: $|x^k - x^*| \leq \varepsilon$, where x^k is the point generated at Step 6 of GS and x^* is the global minimizer (known a priori for all test problems)). This situation happened for $r = 1.8$: the corresponding results are shown in the column Geom-LTA of Table 1.2.

As can be seen from Table 1.2, the performance of the method Geom-LTMA was better with respect to the other geometric algorithms tested. The experiments also showed that the additive convolution (Geom-LTA) did

#	Geom-AL	Geom-GL	Geom-LTM	Geom-LTA	Geom-LTMA
1	595	446	50	44	**35**
2	457	373	49	52	**39**
3	577	522	176	202	**84**
4	1177	1235	57	73	**47**
5	383	444	57	65	**43**
6	301	299	70	73	**50**
7	575	402	53	51	**41**
8	485	481	164	183	**82**
9	469	358	55	57	**41**
10	571	481	55	58	**42**
11	1099	1192	100	109	**78**
12	993	1029	93	96	**68**
13	2833	2174	93	88	**68**
14	379	303	56	60	**39**
15	2513	1651	89	118	**72**
16	2855	2442	102	120	**83**
17	2109	1437	125	171	**122**
18	849	749	55	58	**41**
19	499	377	49	47	**39**
20	1017	166	53	58	**40**
Avg	1036.80	828.05	80.05	89.15	**57.70**

Table 1.2: Number of trials performed by the considered geometric methods without the local improvement on 20 tests from Appendix A.

not guarantee the proximity of the found solution to the global minimum with the common value $r = 1.1$. With an increased value of the reliability parameter r, the average number of trials performed by this method on 20 tests was also slightly worse than that of the method with the maximum convolution (Geom-LTM) but better than the averages of the methods using global estimates of the Lipschitz constants (Geom-AL and Geom-GL).

Results of numerical experiments with information methods without the local improvement techniques (methods Inf-AL, Inf-GL, Inf-LTM, Inf-LTA, and Inf-LTMA) on the same 20 tests from Appendix A are shown in Table 1.3. Parameters of the information methods were chosen as follows. The estimates of the Lipschitz constants for the method Inf-AL were the same as for the method Geom-AL. The reliability parameter $r = 2$ was used in the methods Inf-GL, Inf-LTM, and Inf-LTMA, as recommended in [189, 213, 215]. For all the information methods with the local tuning techniques (Inf-LTM, Inf-LTA, and Inf-LTMA), the value $\xi = 10^{-8}$ was used. For the method Inf-LTA,

#	Inf-AL	Inf-GL	Inf-LTM	Inf-LTA	Inf-LTMA
1	422	501	46	35	**32**
2	323	373	47	38	**36**
3	390	504	173	72	**56**
4	833	1076	51	56	**47**
5	269	334	59	47	**37**
6	208	239	65	46	**45**
7	403	318	49	38	**37**
8	157	477	163	113	**63**
9	329	339	54	48	**42**
10	406	435	51	42	**38**
11	773	1153	95	78	**75**
12	706	918	88	71	**64**
13	2012	1351	54	54	**51**
14	264	349	55	44	**38**
15	1778	1893	81	82	**71**
16	2023	1592	71	67	**64**
17	1489	1484	128	121	**105**
18	601	684	52	**43**	**43**
19	352	336	44	34	**33**
20	681	171	55	**39**	**39**
Avg	720.95	726.35	74.05	58.40	**50.80**

Table 1.3: Number of trials performed by the considered information methods without the local improvement on 20 tests from Appendix A.

the parameter r was increased (starting from $r = 2$) up to the value $r = 2.3$ when all 20 test problems were solved.

As can be seen from Table 1.3, the performance of the method Inf-LTMA was better (as also verified for its geometric counterpart) with respect to the other information algorithms tested. The experiments also showed that the average number of trials performed by the Inf-LTA method with $r = 2.3$ on 20 tests was better than that of the method with the maximum convolution (Inf-LTM).

The second series of experiments (see Table 1.4) was executed on the class of 100 Pintér's test functions from [153] with all geometric and information algorithms without the local improvement (i. e., all the methods used in the first series of experiments). Each Pintér's test function is defined over the interval $[a, b] = [-5, 5]$ as follows:

Method	Average	StDev	Method	Average	StDev
Geom-AL	1080.24	91.17	Inf-AL	750.03	66.23
Geom-GL	502.17	148.25	Inf-GL	423.19	109.26
Geom-LTM	58.96	9.92	Inf-LTM	52.13	5.61
Geom-LTA	70.48	17.15	Inf-LTA	**36.47**	6.58
Geom-LTMA	**42.34**	6.63	Inf-LTMA	38.10	5.96

Table 1.4: Results of numerical experiments with the considered geometric and information methods without the local improvement on 100 Pintér's test functions from [153].

$$f_n(x) = 0.025(x - x_n^*)^2 + \sin^2[(x - x_n^*) + (x - x_n^*)^2] + \sin^2(x - x_n^*), \quad 1 \le n \le 100, \tag{1.16}$$

where the global minimizer x_n^*, $1 \le n \le 100$, is chosen randomly and differently for all functions from the search interval by means of the random number generator used in the GKLS-generator of multidimensional test functions (see [65] for details). The GKLS-generator can be free-downloaded, so the source code is available for repeating all the experiments with random functions. Parameters of the methods Geom-AL, Geom-GL, Geom-LTM, Geom-LTMA, and Inf-AL, Inf-GL, Inf-LTM and Inf-LTMA were the same as in the first experimental series ($r = 1.1$ for all the geometric methods and $r = 2$ for the information methods). The reliability parameter for the method Geom-LTA was increased from $r = 1.1$ to $r = 1.8$ (when all 100 problems were solved). All the information methods were able to solve all 100 test problems with $r = 2$ (see Table 1.4). The average performance of the Geom-LTMA and the Inf-LTA methods was the best among the other considered geometric and information algorithms, respectively.

In the following several series of experiments, the local improvement techniques were compared on the same sets of test functions. In the third series (see Table 1.5), six methods (geometric and information) with the optimistic local improvement (methods Geom-LTIMO, Geom-LTIAO, Geom-LTIMAO and Inf-LTIMO, Inf-LTIAO and Inf-LTIMAO) were compared on the class of 20 test functions from [83] (see Appendix A). The reliability parameter $r = 1.1$ was used for the methods Geom-LTIMO and Geom-LTIMAO and $r = 2$ was used for the method Inf-LTIMO. For the method Geom-LTIAO r was increased to 1.6 and for the methods Inf-LTIMAO and Inf-LTIAO r was increased to 2.3. As can be seen from Table 1.5, the best average result among all the algorithms was shown by the method Geom-LTIMAO (while the Inf-LTIMAO was the best in average among the considered information methods).

#	Geom LTIMO	Geom LTIAO	Geom LTIMAO	Inf LTIMO	Inf LTIAO	Inf LTIMAO
1	45	41	**35**	47	**35**	37
2	47	49	**35**	45	**37**	41
3	49	45	**39**	55	**45**	51
4	47	53	**43**	**49**	53	53
5	55	49	**47**	51	**47**	**47**
6	51	49	**45**	47	**43**	47
7	45	45	**39**	49	**37**	39
8	37	41	**35**	**41**	45	47
9	49	51	**41**	51	51	**40**
10	47	49	**41**	51	**43**	**43**
11	49	53	**45**	**55**	59	**55**
12	43	53	**35**	53	67	**45**
13	**51**	53	57	**41**	51	55
14	45	45	**43**	49	**43**	45
15	**45**	57	47	**45**	55	53
16	**49**	55	53	**47**	49	53
17	93	**53**	95	59	55	**53**
18	45	47	**37**	49	**41**	44
19	45	43	**35**	46	**33**	35
20	43	45	**37**	49	**35**	39
Avg	49.00	48.80	**44.20**	48.95	46.20	**46.10**

Table 1.5: Number of trials performed by the considered geometric and information methods with the *optimistic* local improvement on 20 tests from Appendix A.

#	Geom LTIMP	Geom LTIAP	Geom LTIMAP	Inf LTIMP	Inf LTIAP	Inf LTIMAP
1	49	46	**36**	47	38	**35**
2	49	50	**38**	47	37	**35**
3	165	212	**111**	177	**56**	57
4	56	73	**47**	51	56	**46**
5	63	66	**48**	57	47	**38**
6	70	71	**51**	64	46	**45**
7	54	53	**41**	51	39	**38**
8	157	182	**81**	163	116	**99**
9	53	57	**43**	52	52	**43**
10	56	59	**42**	52	43	**39**
11	100	114	**77**	95	78	**72**
12	93	97	**69**	87	73	**64**
13	97	86	**68**	55	52	**50**
14	58	197	**43**	60	46	**42**
15	79	120	**76**	79	82	**70**
16	97	115	**81**	71	66	**60**
17	140	189	**139**	127	129	**100**
18	55	60	**42**	51	**42**	**42**
19	52	50	**36**	46	33	**32**
20	54	56	**40**	51	**37**	40
Avg	79.85	97.65	**60.45**	74.15	58.40	**52.35**

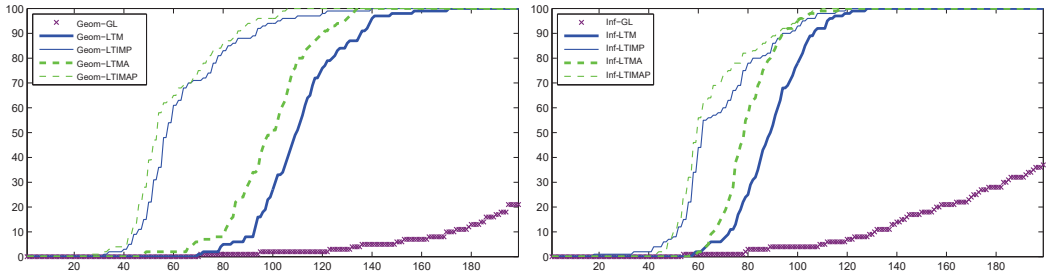
Table 1.6: Number of trials performed by the considered geometric and information methods with the *pessimistic* local improvement on 20 tests from Appendix A.

Optimistic strategy				Pessimistic strategy			
Method	r	Average	StDev	Method	r	Average	StDev
Geom-LTIMO	1.3	49.52	4.28	Geom-LTIMP	1.1	66.44	21.63
Geom-LTIAO	1.9	48.32	5.02	Geom-LTIAP	1.8	93.92	197.61
Geom-LTIMAO	1.4	**45.76**	5.83	Geom-LTIMAP	1.1	**48.24**	14.12
Inf-LTIMO	2.0	48.31	4.29	Inf-LTIMP	2.0	53.06	7.54
Inf-LTIAO	2.1	**36.90**	5.91	Inf-LTIAP	2.0	**37.21**	7.25
Inf-LTIMAO	2.0	38.24	6.36	Inf-LTIMAP	2.0	39.06	6.84

Table 1.7: Results of numerical experiments with the considered geometric and information methods with the local improvement techniques on 100 Pintér's test functions from [153].

In the fourth series of experiments, six methods (geometric and information) using the pessimistic local improvement were compared on the same 20 test functions. The obtained results are presented in Table 1.6. The usual values $r = 1.1$ and $r = 2$ were used for the geometric (Geom-LTIMP and Geom-LTIMAP) and the information (Inf-LTIMP and Inf-LTIMAP) methods, respectively. The values of the reliability parameter ensuring the solution to all the test problems in the case of methods Geom-LTIAP and Inf-LTIAP were set as $r = 1.8$ and $r = 2.3$, respectively. As can be seen from Table 1.6, the "Maximum" and the "Maximum-Additive" local tuning techniques were more stable and generally allowed us to find the global solution for all test problems without increasing r. Moreover, the methods Geom-LTIMAP and Inf-LTIMAP showed the best performance with respect to the other techniques in the same geometric and information classes, respectively.

In the fifth series of experiments, the local improvement techniques were compared on the class of 100 Pintér's functions. The obtained results are presented in Table 1.7. The values of the reliability parameter r for all the methods were increased, starting from $r = 1.1$ for the geometric methods and $r = 2$ for the information methods, until all 100 problems from the class were solved. It can be seen from Table 1.7, that the best average number of trials for both the optimistic and pessimistic strategies was almost the same (36.90 and 37.21 in the case of information methods and 45.76 and 48.24 in the case of geometric methods, for the optimistic and for the pessimistic strategies, respectively). However, the pessimistic strategy seemed to be more stable since its reliability parameter (needed to solve all the problems) generally remained smaller than that of the optimistic strategy. In average, the Geom-LTMA and the Inf-LTA methods was the best among the other considered geometric and information algorithms, respectively.



(a) (b)

Figure 1.2: Operational characteristics for the geometric (a) and information (b) Lipschitz global optimization algorithms over 100 Shekel-type test functions

Finally, in the sixth series of experiments, the following 10 methods have been compared on the class of 100 Shekel-type test functions: Geom-GL, Geom-LTM, Geom-LTIMP, Geom-LTMA, Geom-LTIMAP, Inf-GL, Inf-LTM, Inf-LTIMP, Inf-LTMA, Inf-LTIMAP. In this series of experiments, control parameters for each algorithm have been set as follows. First, for each method the technical parameter ξ was set to 10^{-8}. Second, for each method the initial value $r = 2$ was used over all the classes of test functions and it was increased until all test problems were solved. Each test problem was considered to be solved if an algorithm has generated a point x^k after k trials such that:

$$|x^k - x^*| \leq \epsilon, \tag{1.17}$$

where x^* is the global minimizer for the problem (1.3),(1.2), and ϵ is the given accuracy (that was set 10^{-6} in our experiments). So, for the algorithms Geom-GL, Geom-LTM, and Geom-LTIMP r was set to 2, for Geom-LTMA – 2.5, for Geom-LTIMAP – 2.3, for Inf-GL – 2.8, for Inf-LTM – 3, for Inf-LTIM – 3.7, for Inf-LTMA – 4, and for Inf-LTIMAP – 4.2. However, it should be noticed that many test problems of the class can be solved also with smaller values of the parameter r for each algorithm. The use of a common value of the parameters for the whole class of test functions can increase the number of trials.

One of the most efficient methods for a numerical comparison on the class of test problems in terms of costly function evaluations uses operational characteristics proposed by Grishagin in [78]. The operational characteristic of a method on a class of test problems is a non-decreasing function that indicates the number of problems solved by this method after each trial (see the next Section for details). It is convenient to represent the operational characteristics of a method in a graph (see Fig. 1.2.a and Fig. 1.2.b). Among

different methods the better method is that with the highest operational characteristic.

As we can see from Fig. 1.2, all the methods can be divided into three groups with respect to their performance: methods with the global estimate of the Lipschitz constant (methods Geom-GL and Inf-GL) with the lower operational characteristics, local tuning with local improvement (methods Geom-LTMP, Geom-LTIMAP, Inf-LTMP, and Inf-LTIMAP) with the highest operational characteristics, and local tuning without local improvement (methods Geom-LTM, Geom-LTMA, Inf-LTM, and Inf-LTMA), that have the medium operational characteristics. Since the methods Geom-GL and Inf-GL use the global Lipschitz constants, they need more trials to achieve the desired accuracy as the Lipschitz constant can be large in few subintervals and small in other ones. That is the advantage of the local tuning techniques that use the local Lipschitz constants for each subinterval and avoid this problem. We can see also that the use of local improvement techniques improves the results also and a combination of the local tuning with the local improvement techniques can accelerate the search significantly.

Figure 1.2 demonstrates also that the methods with the "maximum-additive" local tuning show better performance with respect to the other techniques: among all geometric (information) algorithms over the presented class of test problems the best one was Geom-LTIMAP (Inf-LTIMAP) and among all geometric (information) algorithms without local tuning the best one was Geom-LTMA (Inf-LTMA).

1.2 Solving practical engineering problems

As mentioned above, univariate Lipschitz global optimization are very important from a practical engineering point of view. In this work, two practical engineering problems are considered. The first problem is the sinusoidal parameter estimation problem that is considered to fit a sum of damped sinusoids to a series of noisy observations. It can be formulated as a nonlinear least-squares global optimization problem (see, e.g., [72, 73, 195]). A one-parametric case study is examined to determine an unknown frequency of a signal. The second problem is finding the minimal root of the non-linear equation. In this problem, a device whose behavior depends on some characteristic function $\phi(x)$ is considered. The device works correctly only if the value of $f(x)$ is greater than zero. It is necessary to find the first point x for which the value of $\phi(x)$ is equal to zero or to demonstrate that $\phi(x)$ is positive over the whole search interval. Obviously, the value of $\phi(x)$ at the initial point $x_0 = 0$ should be positive. This problem was reformulated as

the global optimization problem in [38, 188].

1.2.1 Applications in noisy data fitting and electrical engineering

A general nonlinear regression model can be often considered in the form of fitting a sum of damped sinusoids to a series of observations corrupted by noise (see, e. g., [20, 27, 63, 73, 121]). The sinusoidal functions are frequently used in many real-life applications such as signal processing (see, e. g., [31, 48, 51, 71, 76, 131, 156]). Parameters $\mathbf{x}$ of these functions (consisting of amplitudes, frequencies, and phases) can be estimated by solving the following minimization problem:

$$f(\mathbf{x}^*) = f^* = \min_{\mathbf{x}\in D} f(\mathbf{x}), \quad f(\mathbf{x}) = \sum_{i=1}^{T} (y_{t_i} - \phi(\mathbf{x}, t_i))^2, \quad \mathbf{x} \in D \subset \mathbb{R}^n, \quad (1.18)$$

where

$$\phi(\mathbf{x}, t) = \sum_{l=1}^{s} a_l e^{d_l t} \sin(2\pi\omega_l t + \theta_l), \quad t = t_1, \ldots, t_T, \quad (1.19)$$

and $s \geq 1$ is a fixed integer, $\mathbf{x} = (\mathbf{a}, \mathbf{d}, \omega, \theta)$ with $\mathbf{a} = (a_1, \ldots, a_s)$, $\mathbf{d} = (d_1, \ldots, d_s)$, $\omega = (\omega_1, \ldots, \omega_s)$, and $\theta = (\theta_1, \ldots, \theta_s)$.

It is supposed that real-valued observations y_{t_i} are affected by noise:

$$y_{t_i} = \phi(\bar{\mathbf{x}}, t_i) + \xi_{t_i}, \quad i = 1, \ldots, T, \quad (1.20)$$

where $\bar{\mathbf{x}}$ is the true vector of parameters (it coincides with the estimator $\mathbf{x}^*$ from (1.18) in the case of noise-free observations) and ξ_{t_i}, $i = 1, \ldots, T$, are independently and identically distributed random variables with zero mean and a given variance σ^2.

As a case study to illustrate the performance of various techniques for solving the global optimization problem (1.18)–(1.20), let us consider the sine function with unknown frequency only in (1.19) (i. e., $s = 1$ and $a_1 = 1$, $d_1 = 0$, $\theta_1 = 0$ in (1.19)) over uniformly sampled observations y_i, $i = 1, \ldots, T$ in (1.20). In this case, the vector of parameters $\mathbf{x}$ consists of only one component $x := \omega = \omega_1$. In spite of its apparent simplicity, such a problem is however representative from the practical point of view (see, e. g., [20, 63, 156]) and useful to obtain conclusions on the problem behavior that can be then generalized to a general multiparametric model (1.18)–(1.20).

Problem (1.18)–(1.20) is thus reduced to the following one-dimensional global minimization problem:

$$f(x^*) = f^* = \min_{x\in[a,b]} f(x), \quad f(x) = \sum_{i=1}^{T}(y_i - \sin(2\pi x\, i))^2, \quad x \in [a,b] \subset \mathbb{R}, \tag{1.21}$$

where for any fixed number of observations T the function $f(x)$ is Lipschitzian with the Lipschitz constant L, $0 < L < \infty$, and continuously differentiable (see [72, 75]) with the Lipschitz constant K, $0 < K < \infty$, for its Lipschitzian first derivative $f'(x)$ over the search interval $[a, b]$ (taken here as $[0, 1]$ due to the periodicity of the sine function in (1.21)).

By changing the number of observations T in (1.20), the objective functions (1.21) of different shapes can be obtained, with the number of local minima and the average function value proportional to T. In our case study, the values $T = 10$, 50, and 100 were considered. The noise-free observations y_i, $i = 1, \ldots, T$, were obtained numerically to keep the estimator x^* in (1.21) equal to the true parameter $\bar{x}$ with $f(x^*) = 0$ (in our experiments, two values of $\bar{x}$ were used: $\bar{x} = 0.4$ and $\bar{x} = 0.7$). The noisy observations were then produced by adding random variables ξ_i, $i = 1, \ldots, T$, taken from normal distribution ($\mathbf{N}(0, \sigma^2)$) with $\sigma^2 = 9$ (see, e. g., [73, 74] for other noise parameters), to the corresponding noise-free values. Consequently, the global minimizer x^* in (1.21) was shifted from the true parameter value $\bar{x}$ (see the second column in Table 1.2.1) and the (non-normalized) objective function (1.21) became more erratic (with an unknown positive minimum value f^*). The search for the global minimizer x^* from (1.21), rather than for the true parameter value $\bar{x}$, was performed in this case too, in order to estimate the behavior of the numerical methods from the global optimization viewpoint.

Another important problem is finding the minimal root of the non-linear equation. It is a common problem in electrical engineering and electronic measurements. The behavior of a device depends on a characteristic $f(x)$, $x \in [a, b]$, where the function $f(x)$ can be, e. g, a multiextremal electrical signal measured by a computer aided system (as a black box) over a time interval $[a, b]$ (see the function graph in in Fig. 1.3). The device work is correct if $f(x) > 0$. It is required to examine the device performance over the time interval $[a, b]$ either by finding the point ζ such that

$$f(\zeta) = 0, \quad f(x) > 0, \quad x \in [a, \zeta), \ \zeta \in (a, b],$$

or by demonstrating the positiveness of $f(x)$ over the whole interval $[a, b]$ (in this case the device works correctly for the whole time period, but the information about the (positive) global minimum of $f(x)$ is useful to measure the device reliability).

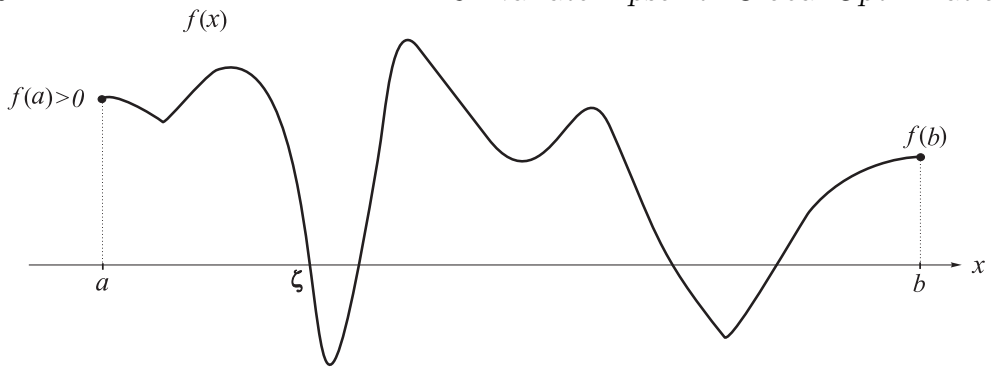


Figure 1.3: The problem of finding the minimal root of equation $f(x) = 0$ with multiextremal left part arising in applied engineering problems

Function (T, σ^2)	x^*	Lipschitz constant L	Lipschitz constant K
$(10, 0)$	$\bar{x} = 0.7$	354.1	30 567.2
$(50, 0)$	$\bar{x} = 0.7$	7 216.4	3 390 330.5
$(100, 0)$	$\bar{x} = 0.7$	28 126.7	26 717 323.0
$(10, 9)$	0.6126936	978.2	47 952.8
$(50, 9)$	0.7650428	19 486.4	6 602 640.8
$(100, 9)$	0.3055982	56 272.4	26 724 566.7

Table 1.8: The Lipschitz constants L and K for the objective functions $f(x)$ from (1.21) and their derivatives $f'(x)$, respectively, considered in our case study with different numbers of observations T, noise terms from normal distribution $\mathbf{N}(0, \sigma^2)$, and the true frequency value $\bar{x} = 0.7$.

This problem is equivalent to the problem of finding the minimal root (the first root from the left) of the equation $f(x) = 0$, $x \in [a, b]$, in the presence of certain initial conditions. Since the function $f(x)$ is multiextremal (see Fig. 1.3) the problem is quite difficult because many roots can exist in $[a, b]$ and, therefore, classical root finding techniques (often using local search ideas) can be inappropriate. The problem reformulation in terms of global optimization (see, e. g., [38, 183, 215]) allows one to apply global optimization techniques for finding its solution.

1.2.2 Experimental study

In the first series of the experiments, six particular functions (1.21) were considered in this study: three of them were based on noise-free terms in (1.20) and the other three—on the corresponding noisy observations. In what follows, these functions are labelled as pairs (T, σ^2), with $T = 10, 50$, and 100

and $\sigma^2 = 9$. The values (determined over 10^{-7} grid) of the Lipschitz constants L and K for $f(x)$ and $f'(x)$, respectively, are reported in Table 1.2.1. It can be seen from this Table, how the complexity of the functions increases both with the increasing of T and with adding the noise. Moreover, high values of the Lipschitz constants (especially, for K) can be observed from Table 1.2.1: they make the usage of Lipschitz global optimization methods challenging to solve the stated problem either in its one-parametric variant (1.21) or in its general form (1.18)–(1.20).

The complexity of the considered instances of problem (1.18)–(1.20) from the global optimization point of view can be also seen from the graphs of the objective functions $f(x)$ from Table 1.2.1 and their first derivatives reported in Figures 1.4 and 1.5 for the cases of noise-free ($\sigma^2 = 0$) and noisy ($\sigma^2 = 9$) observations, respectively. As one can note, the objective functions are highly multiextremal and irregular, with the global minimizers having narrow attraction regions, although well separated with respect to the global minimum values for higher numbers of observations T. Hence, already for our case study which is a relatively simple case with respect to the general problem (1.18)–(1.20), a particular attention should be paid to the choice of numerical methods able to tackle efficiently the stated global optimization problem.

The following Lipschitz global optimization methods belonging to the general scheme described above are used in this part of the study: Geom-AL, Geom-GL, Inf-GL and Geom-LTM. Moreover, the following two algorithms with smooth auxiliary functions are also studied:

Smooth-AK: Geometric method constructing **Smooth** auxiliary functions based on the usage of the first derivative information $f'(x)$ and **A** priori estimate of the Lipschitz constant $\boldsymbol{K}$ for $f'(x)$ (see, e. g., [69, 167, 189, 215]);

Smooth-LTM: Geometric derivative-based method constructing **Smooth** auxiliary functions and using the **L**ocal **T**uning technique with the **M**aximum convolution product of local and global estimates of the Lipschitz constant for $f'(x)$ (see, e. g., [165, 166, 189, 207]);

These two methods use the smooth characteristics R_i from [189]. The "Maximum" local tuning for the first derivative has been used as follows: set the estimate $K_j := r \cdot \max\{\Lambda_j, \Gamma_j, \eta\}$ of the Lipschitz constant K for the first derivative $f'(x)$, where

$$\Lambda_j = \max\{G_{\tilde{j}} : 2 \le \tilde{j} \le k, j-1 \le \tilde{j} \le j+1\},$$

$$\Gamma_j = \max\{G_j : j = 2, \ldots, k\} \cdot (x_j - x_{j-1})/X^{\max},$$

$$G_j = \frac{|2(z_{j-1} - z_j) + (\mathrm{d}z_i + \mathrm{d}z_{i-1})(x_j - x_{j-1})| + \delta_j}{(x_j - x_{j-1})^2},$$

with

$$\delta_j = \{[2(z_{j-1} - z_j) + (\mathrm{d}z_j + \mathrm{d}z_{j-1})(x_j - x_{j-1})]^2 + \\ + (\mathrm{d}z_j - \mathrm{d}z_{j-1})^2(x_j - x_{j-1})^2\}^{1/2}$$

and $\mathrm{d}z_i = f'(x_i)$, $X^{\max} = \max\{x_j - x_{j-1} : \ j = 2, \ldots, k\}$, and r and η have the same sense as in the general scheme presented above.

In order to estimate the behavior of the Lipschitz global optimization methods, some widely used metaheuristic algorithms were taken for the comparison, namely: **D**ifferential **E**volution (DE), **P**article **S**warm **O**ptimization (PSO), **A**rtificial **B**ee **C**olony (ABC), and **F**ire**F**ly (FF) algorithms (see the next Chapter for their detailed description).

Parameters of these methods were chosen as follows (see, e. g., [107, 106]; all the parameters were thoroughly studied to respect the multiextremal character of the objective functions, as suggested in the literature). For the DE algorithm, the differential weight F was set equal to 0.7; the crossover rate CR was set equal to 0.5; and the mutation strategy DE/rand/1/exp was used as one of the most powerful strategies. For the PSO algorithm, the static inertia weight ω was set equal to 0.6; the cognitive ϕ_l and social ϕ_g parameters were set both equal to 2.0; and the velocity $v_{\max}$ was set equal to 15% of the search interval. For the ABC method, the number of the employed bees was set equal to the number of the onlooker bees and was equal to half population; the number of scout bees was set equal to 1; and the *limit* parameter was set equal to the number of food sources (candidate solutions) as half population. Finally, for the FF algorithm, the randomization parameter α was set equal to $0.005(b-a)$; the absorbtion coefficient γ was set equal to $0.01/\sqrt{b-a}$; and the attractiveness parameter β_0 was set equal to 1.

Func	Geom-AL	Geom-GL	Inf-GL	Geom-LTM	Smooth-AK	Smooth-LTM
$(10, 0)$	109	83	72	45	26	21
$(50, 0)$	149	130	116	80	128	65
$(100, 0)$	273	173	185	150	250	128
$(10, 9)$	113	87	111	50	23	17
$(50, 9)$	199	191	188	145	116	84
$(100, 9)$	329	315	323	290	149	138
Avg	195.3	163.2	165.8	126.7	115.3	77.5

Table 1.9: Number of trials for the considered Lipschitz methods stopped by their internal stopping criterion (1.14) with $\varepsilon = 10^{-4}$

Func	Geom-AL	Geom-GL	Inf-GL	Geom-LTM	Smooth-AK	Smooth-LTM
$(10, 0)$	677	535	668	66	32	23
$(50, 0)$	433	395	353	101	135	67
$(100, 0)$	452	352	385	174	256	129
$(10, 9)$	845	763	730	73	28	21
$(50, 9)$	513	501	424	165	120	85
$(100, 9)$	545	511	516	314	153	141
Avg	577.5	509.5	512.7	148.8	120.7	77.7

Table 1.10: Number of trials for the considered Lipschitz methods stopped by their internal stopping criterion (1.14) with $\varepsilon = 10^{-6}$

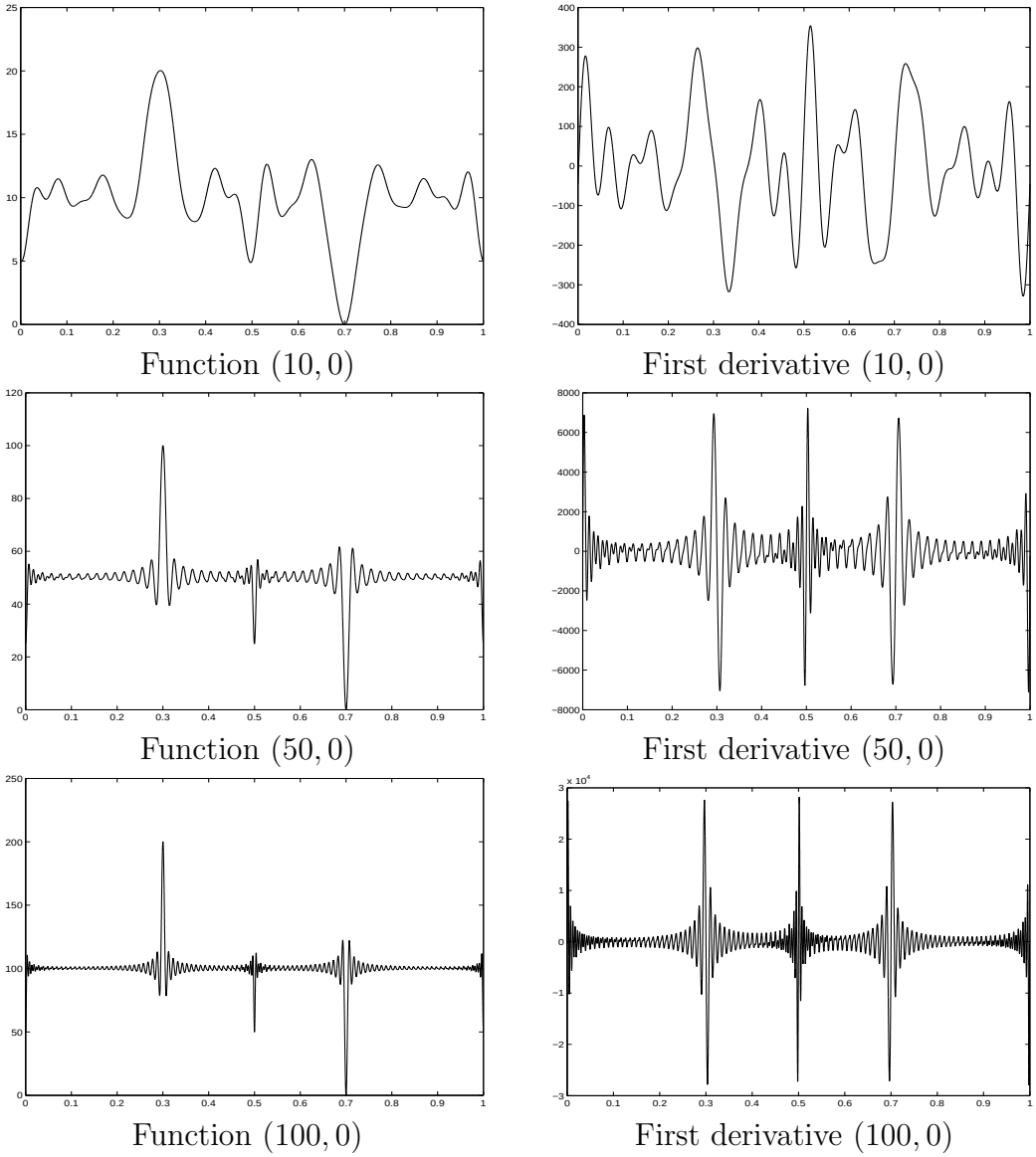
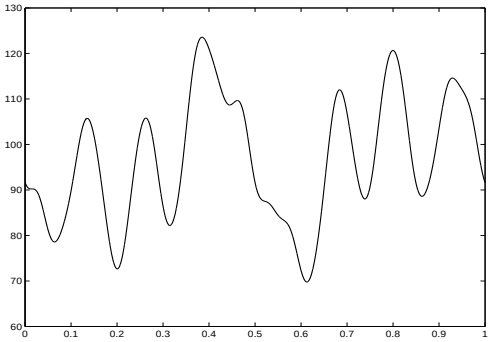
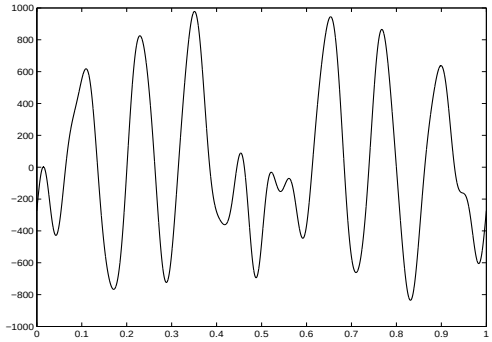


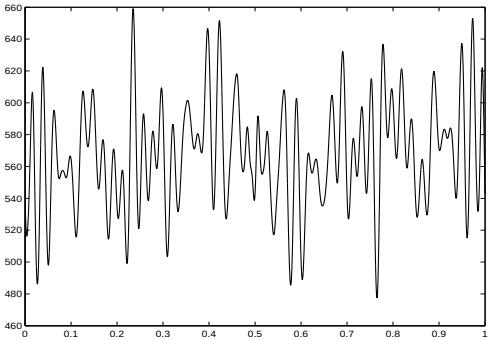
Figure 1.4: Graphs of the objective functions (left column) and their first derivatives (right column) for the considered noise-free problems from Table 1.2.1



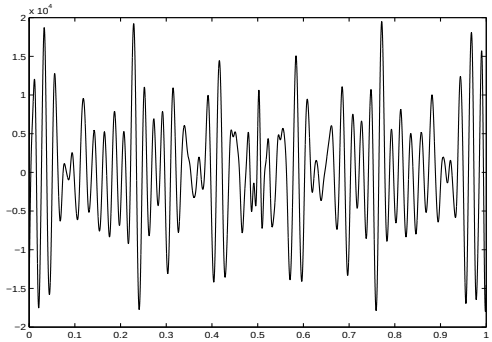
Function $(10, 9)$



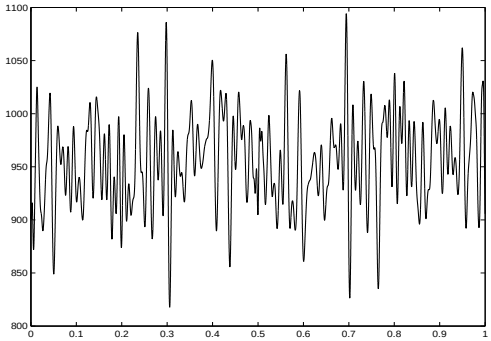
First derivative $(10, 9)$



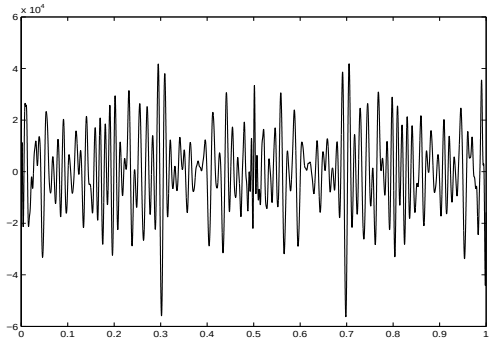
Function $(50, 9)$



First derivative $(50, 9)$



Function $(100, 9)$



First derivative $(100, 9)$

Figure 1.5: Graphs of the objective functions (left column) and their first derivatives (right column) for the considered noisy problems from Table 1.2.1

Func	Geom-AL	Geom-GL	Inf-GL	Geom-LTM	Smooth-AK	Smooth-LTM	DE		PSO		ABC		FF	
							Avg	Fails	Avg	Fails	Avg	Fails	Avg	Fails
$(10, 0)$	55	27	35	34	22	16	254.1	0%	187.0	0%	548.3	0%	188.2	0%
$(50, 0)$	130	109	87	44	71	63	472.0	0%	433.4	0%	1493.2	0%	277.2	0%
$(100, 0)$	130	134	136	135	220	100	635.7	0%	669.6	7%	1456.4	1%	410.3	0%
$(10, 9)$	60	58	54	40	19	15	258.4	4%	232.1	0%	338.8	0%	183.0	7%
$(50, 9)$	153	156	159	129	112	73	658.0	20%	541.5	52%	2668.7	8%	853.7	0%
$(100, 9)$	251	240	195	262	146	130	769.2	16%	737.1	37%	2722.2	1%	1133.7	0%
Avg	129.8	120.7	111.0	107.3	98.3	66.2	507.9	6.7%	466.8	16.0%	1537.9	1.7%	507.7	1.2%

Table 1.11: Number of trials for the considered Lipschitz and metaheuristic methods stopped by the first-successful-point stopping criteria with $\varepsilon = 10^{-4}$

Func	Geom-AL	Geom-GL	Inf-GL	Geom-LTM	Smooth-AK	Smooth-LTM	DE		PSO		ABC		FF	
							Avg	Fails	Avg	Fails	Avg	Fails	Avg	Fails
$(10, 0)$	145	199	131	59	29	20	384.9	0%	479.9	0%	3620.6	20%	4088.6	23%
$(50, 0)$	277	228	151	97	132	65	615.7	0%	607.1	0%	5850.5	84%	4045.1	20%
$(100, 0)$	342	162	221	169	253	127	798.2	0%	717.3	8%	5035.1	90%	4239.3	25%
$(10, 9)$	386	281	85	60	25	19	399.4	4%	528.8	0%	1878.7	0%	3701.1	27%
$(50, 9)$	293	221	202	157	118	83	859.1	20%	860.6	52%	4330.7	97%	3866.7	63%
$(100, 9)$	386	298	406	309	150	138	1011.4	16%	1053.4	38%	4956.0	89%	5538.3	74%
Avg	304.8	231.5	199.3	141.8	117.8	75.3	678.1	6.7%	707.9	16.3%	4278.6	63.3%	4246.5	38.7%

Table 1.12: Number of trials for the considered Lipschitz and metaheuristic methods stopped by the first-successful-point stopping criteria with $\varepsilon = 10^{-6}$

The reliability parameter r of the considered Lipschitz global optimization methods (except the methods with a priori given Lipschitz constants) was set close to 1 (namely, 1.1) for the geometric methods Geom-GL, Geom-LTM, Smooth-AK, and Smooth-LTM, and to 2.0 for the Inf-GL method, as recommended by the convergence study of these algorithms.

For all metaheuristic algorithms the population size was set equal to 10, the number of runs was set equal to 100 (i. e., each metaheuristic algorithm was launched 100 times for a given problem) and for all the methods the maximal number of trials was set equal to 10 000. Since the metaheuristic methods do not have any internal stopping criterion, each their run has been arrested when a trial point in an ε-neighborhood of the global minimizer x^* (known for the considered benchmark instances) has been generated, i.e., when the algorithm has generated a point x^k after k trials such that

$$|x^k - x^*| \leq \varepsilon.$$

In what follows, such a stopping criterion will be termed as the first-successful-point stopping criterion.

The results of the experiments are presented in Tables 1.9–1.12. Particularly, the numbers of trials generated by the considered Lipschitz global optimization methods when using their internal stopping criterion (aiming at the global optimality certification) with different accuracy coefficients ε are given in Tables 1.9–1.10. It can be seen from Tables 1.9–1.10 that the usage of the local information (as in the methods Geom-LTM, Smooth-AK, and Smooth-LTM) allowed us to solve the problems by executing less trials with respect to the methods that used in their work only global information (as the Geom-AL, Geom-GL, and Inf-GL methods). Moreover, the local tuning methods were less sensitive (in terms of the performed trials) to increasing the accuracy (compare Tables 1.9 and 1.10). In this context, it would be interesting to notice that the functions based on $T = 50$ and $T = 100$ observations were solved (in the case of a higher precision $\varepsilon = 10^{-6}$; see Table 1.10) faster by the three methods Geom-AL, Geom-GL, and Inf-GL than the functions with the smaller observations number $T = 10$. This happened because the functions $(50, \sigma^2)$ and $(100, \sigma^2)$ had the global minimum values much smaller than the mean function values (proportional to T) and, thus, the functions $(10, \sigma^2)$ were more difficult for the methods using only global information during the search. It should be also noted that due to extremely high values of the Lipschitz constants K for $f'(x)$ in our case study (see Table 1.2.1), the first derivative information gave less advantage with respect to the situations when more regular objective functions are minimized.

Results of the numerical comparison of all the methods when using the first-successful-point criterion with different accuracy coefficients ε are shown

in Tables 1.11–1.12, where all the average values for the metaheuristic algorithms were calculated without taking into consideration the failed runs. Even in this condition advantageous for the metaheuristics, their performance was worse than that of the Lipschitz-based methods in our case study.

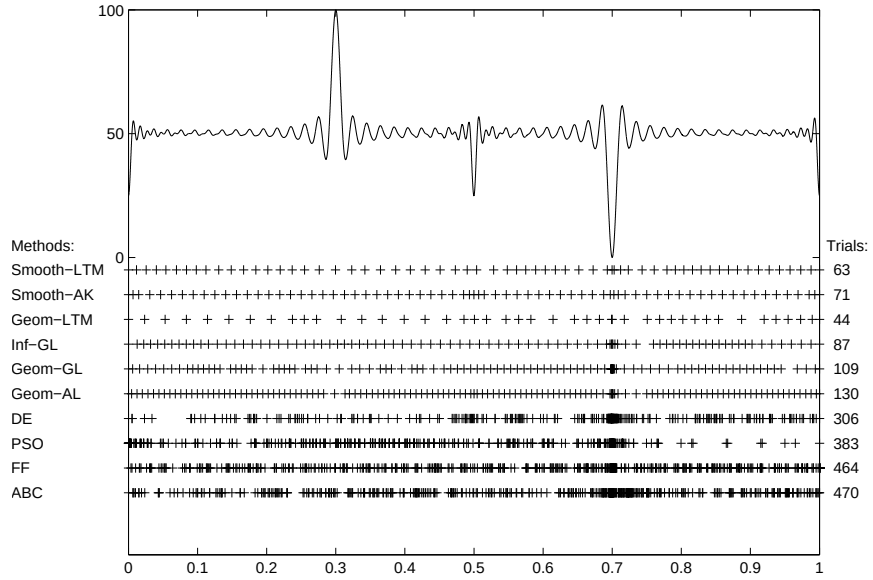


Figure 1.6: Distribution of trial points generated by the tested methods when minimizing the function $(50, 0)$ with the accuracy $\varepsilon = 10^{-4}$ in the first-successful-point stopping criterion

The distribution of trial points generated by all the methods when minimizing the functions $(50, 0)$ ($T = 50$ noise-free observations) and $(50, 9)$ ($T = 50$ noisy observations) with the first-successful-point stopping criterion and two different accuracy coefficients $\varepsilon = 10^{-4}$ and $\varepsilon = 10^{-6}$ is illustrated in Figures 1.6–1.7 (for $\varepsilon = 10^{-4}$) and Figures 1.8–1.9 (for $\varepsilon = 10^{-6}$), respectively.

The other series of the experiments is dedicated to both the sinusoidal parameter estimation problem and finding the minimal root of the equation. Four test problems are studied in this part (see Table 1.13): first two problems dedicated to finding the minimal root are from [189] and the other two functions are from (1.21).

First, all the methods without the local improvement used in the previous subsection (Geom-AL, Geom-GL, Geom-LTM, Geom-LTA, Geom-LTMA and Inf-AL, Inf-GL, Inf-LTM, Inf-LTA, Inf-LTMA) were tested and all the parameters for these methods were the same as above, except the reliability parameters of the methods Geom-LTA and Inf-LTA. Particularly, the app-

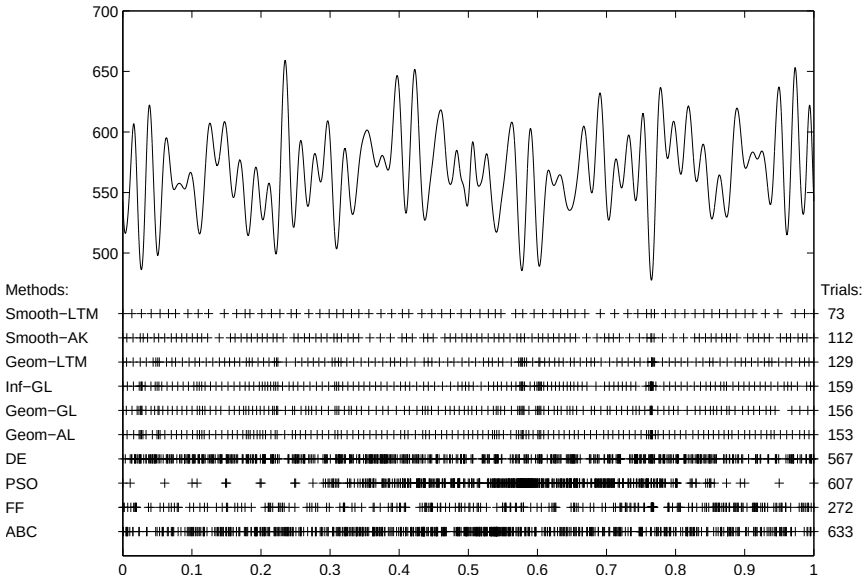


Figure 1.7: Distribution of trial points generated by the tested methods when minimizing the function $(50, 9)$ with the accuracy $\varepsilon = 10^{-4}$ in the first-successful-point stopping criterion

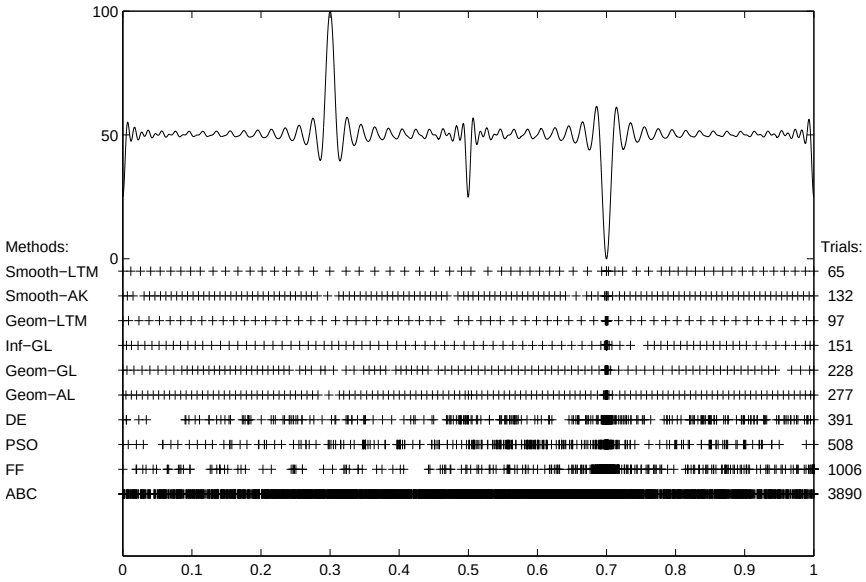


Figure 1.8: Distribution of trial points generated by the tested methods when minimizing the function $(T = 50, \sigma^2 = 0)$ with the accuracy $\varepsilon = 10^{-6}$

lied problem 4 was not solved by the Geom-LTA method with $r = 1.1$. With the increased value $r = 1.8$, the obtained results (reported in Table 1.14)

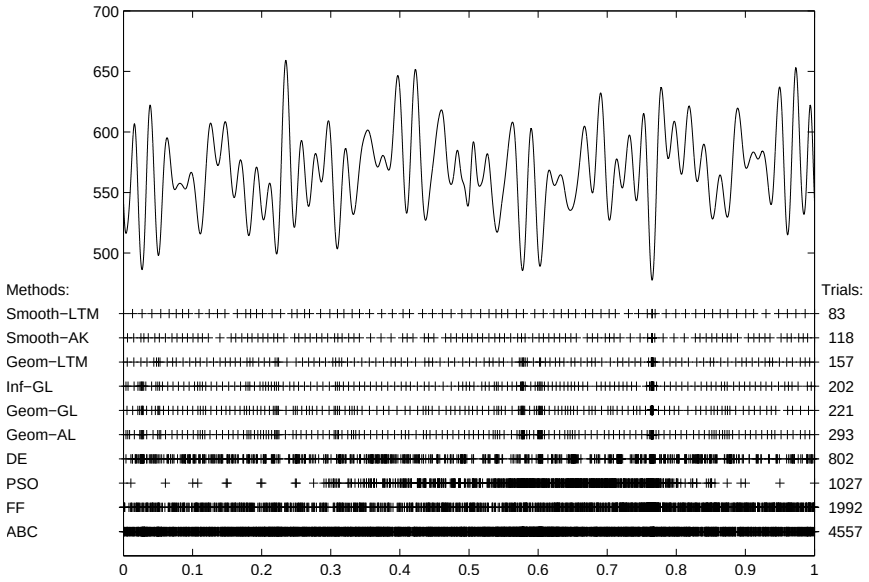


Figure 1.9: Distribution of trial points generated by the tested methods when minimizing the function ($T = 50, \sigma^2 = 9$) with the accuracy $\varepsilon = 10^{-6}$

#	Function $f(x)$	$[a, b]$	L	Global Minimizers	Ref.
1	$\lvert\sin^3(x) + \cos^3(x)\rvert + 0.1$	$[0, 2\pi]$	2.2	$x_1^* = 2.356$, $x_2^* = 5.498$	[189]
2	$x\lvert\sin(x)\rvert + 6$	$[-10, 10]$	9.7	$x^* = -7.979$	[189]
3	$\sum_{i=1}^{10}(y_i^{(1)} - \sin(2\pi x i))^2$	$[0, 1]$	432.0	$x^* = 0.4$	[74]
4	$\sum_{i=1}^{100}(y_i^{(2)} - \sin(2\pi x i))^2$	$[0, 1]$	28690.8	$x^* = 0.4$	[74]

Table 1.13: Four one-dimensional test functions taken from practical applications.

of this geometric method were worse than the results of the other geometric methods with the local tuning (Geom-LTM and Geom-LTMA). The method Inf-LTA solved all the four applied problems also with a higher value $r = 2.3$ and was outrun by the Inf-LTMA method (the latter one produced the best average result among all the competitors; see Table 1.14).

Second, the local improvement techniques presented above were compared on these four applied test problems. The results are presented in Table 1.15, where in the third column the values of the reliability parameter used to solve all the problems are also indicated. As well as in the previous subsection, the pessimistic local improvement strategy seemed to be more stable in the case of this test set, since the optimistic strategy required a significant increase

Method	Test problem				Average
	1	2	3	4	
Geom-AL	37	395	261	332	256.25
Geom-GL	39	388	216	307	237.50
Geom-LTM	37	54	59	232	95.50
Geom-LTA	74	58	68	204	101.00
Geom-LTMA	**33**	**39**	**48**	**137**	**64.25**
Inf-AL	**12**	278	180	187	164.25
Inf-GL	35	333	215	229	203.00
Inf-LTM	25	53	56	212	86.50
Inf-LTA	19	**35**	**40**	165	64.75
Inf-LTMA	24	**35**	**40**	**122**	**55.25**

Table 1.14: Number of trials performed by the considered geometric and information methods without the local improvement on four test functions from Table 1.13.

	Method	r	Test problem				Average
			1	2	3	4	
Optimistic LI	GeomLTIMO	6.5	59	55	63	79	64.00
	Geom-LTIAO	1.8	55	**49**	**49**	**41**	**48.50**
	GeomLTIMAO	6.9	63	59	71	75	67.00
	Inf-LTIMO	6.5	49	55	67	77	62.00
	Inf-LTIAO	9.4	**47**	55	71	71	61.00
	Inf-LTIMAO	8.0	55	55	71	73	63.50
Pessimistic LI	Geom-LTIMP	1.1	39	64	71	228	100.50
	Geom-LTIAP	1.8	243	102	63	1254	415.50
	Geom-LTIMAP	1.1	31	46	51	**106**	**58.50**
	Inf-LTIMP	2.0	25	52	58	185	80.00
	Inf-LTIAP	2.3	**18**	36	49	174	69.25
	Inf-LTIMAP	2.0	24	**35**	**43**	134	59.00

Table 1.15: Number of trials performed by the considered geometric and information methods with the local improvement techniques on four test functions from Table 1.13.

of the parameter r to determine global minimizers of these applied problems (although the best average value obtained by the optimistic Geom-LTIAO method was smaller that that of the best pessimistic Geom-LTIMAP method, see the last column in Table 1.15).

Chapter 2

A Systematic Comparison of Global Optimization Algorithms of Different Nature

Among existing derivative-free global optimization methods two classes of algorithms can be marked out: stochastic metaheuristic algorithms (see, e.g., [41, 62, 87, 97, 102, 157, 225, 222]) and deterministic mathematical programming methods (see, e.g., [59, 87, 88, 94, 152, 188, 192, 215], etc.) The former, due to their simplicity and attractive nature-inspired interpretations (genetic algorithms [41, 87, 222], particle swarm optimization [62], firefly algorithms [225, 222], etc.), are used by a broad community of engineers and practitioners to solve real-life problems whereas the latter are actively studied in academia due to their interesting theoretical properties including a guaranteed convergence. Historically, these two communities are almost completely disjoint: they have different journals, different conferences, and different test functions. Due to the hardness of global optimization problems and different nature of methods from these two groups, the problem of their comparison is very difficult and methods are collated on some dozens of test functions (see, e.g., [45, 55, 62, 88, 94, 108, 152]) giving so a very poor information and non reliable results. In order to bridge the gap between the two communities new efficient visual techniques for a systematic comparison of global optimization algorithms having different nature is proposed.

The first Section of this Chapter is dedicated to traditional ways of the comparison: using a number of test functions and showing that a new method is better in some sense than its competitors. In this methodology, a trade-off between the number of test functions, reliability of the comparison, and visibility of results is searched. The second Section of this Chapter is dedicated to graphical methods of the comparison of algorithms. Instead of traditional

comparisons executed just on several dozens of tests in this Section more than 800 000 runs on 900 randomly generated test problems have been performed for a systematic comparison of the methods. One known and two novel methodologies for comparing global optimization algorithms are applied here: Operational Characteristics from [78] for comparing deterministic algorithms and new Operational Zones and Aggregated Operational Zones generalizing ideas of operational characteristics to collate multidimensional stochastic algorithms.

Finally, in the third Section of this Chapter, a new generator of multidimensional test problems for methods solving constrained Lipschitz global optimization problems is proposed. New classes of GKLS-based multidimensional test problems with non-differentiable, continuously differentiable and twice continuously differentiable multiextremal objective functions and nonlinear constraints are described. In these constrained problems, the global minimizer does not coincide with the global minimizer of the respective unconstrained test problem, and is always located on the boundaries of the admissible region. Four types of constraints are introduced. The possibility to choose the difficulty of the admissible region is also available.

2.1 Numerical Comparison of Nature-Inspired Metaheuristics Using Benchmarks

The following univariate Lipschitz global optimization problem is considered in this Section:

$$f^* := f(x^*) = \min f(x), \quad x \in [a, b] \subset \mathbb{R}, \tag{2.1}$$

where the function $f(x)$ satisfies the Lipschitz condition (1.2) over the interval $[a, b]$ with the Lipschitz constant $L, 0 < L < \infty$. It is supposed that the objective function $f(x)$ can be multiextremal, non-differentiable; black-box; with an unknown Lipschitz constant L; and evaluation of $f(x)$ even at one point is a time-consuming operation.

In this Section, several widely-used nature-inspired metaheuristic algorithms are presented and studied in their univariate implementations. Moreover, the first steps in systematic comparison of methods of different nature are performed using traditional ways: using tables with different numerical characteristics (average, best and worst number of trials, percentage of fails, standard deviations, etc.). Different constraint handling strategies to launch nature-inspired algorithms are studied. Moreover, multiple runs with different randomly chosen initial points are performed on each test problem.

2.1.1 Description of algorithms

In this work, the widely used nature-inspired metaheuristic algorithms such as Genetic algorithm (GA), Differential Evolution algorithm (DE), Particle Swarm Optimization (PSO), Artificial Bee Colony algorithm (ABC), Firefly algorithm (FA) and Simulated Annealing (SA) are considered and compared with several deterministic Lipschitz global optimization algorithms.

Metaheuristic methods form an important part of the state-of-the-art global optimization algorithms. These algorithms are often nature-inspired with multiple interacting agents (see, e. g., [54, 224]). A significant subset of metaheuristcs consists of the so-called swarm intelligence algorithms, developed by mimicking the behavioral characteristics of biological agents such as fish, birds, bees, and so on. For example, particle swarm optimization is based on the swarming intelligence of birds and fish (see, e. g., [102]), the firefly algorithm reflects the flashing pattern of tropical fireflies (see, e. g., [223, 224]). A great number of metaheuristic algorithms such as differential evolution, particle swarm optimization, artificial bee colony, and firefly algorithms have appeared and shown their potential in solving important engineering decision-making problems (see, e. g., the references given in [54, 147, 224]). These methods are briefly described in this Section and their parameters are specified as those often recommended in the literature to solve black-box global optimization problems. Since these algorithms were originally designed for the multidimensional case, they have been adopted to the case of univariate problems, as explained below.

GA. *Genetic* algorithm proposed in [87]. It is one of the basic population based evolutionary algorithms that simulates the evolution on a genotype level. This method uses in its work three main operators: crossover, mutation, and selection. We used the real-coded genetic algorithm with the simulated binary crossover(see, e. g., [40, 41]). The realization of the algorithm was taken from `http://www.egr.msu.edu/~kdeb/codes/rga/rga.tar`.

DE. *Differential Evolution* algorithm in its standard form from [157] (as implemented in `http://www1.icsi.berkeley.edu/~storn/DenewC.zip`). As a population-based algorithm, it solves problem (2.1) by maintaining a population of candidate solutions and creating new candidate solutions by combining existing ones with the usage of the crossover, mutation and selection operators, and then keeping the candidate solution with the currently best score (or fitness).

PSO. *Particle Swarm Optimization* algorithm in its classical version, e. g., from [102]. It solves problem (2.1) starting from a population (swarm)

of candidate solutions (particles) and moving these particles in the search domain according to the particle's position and velocity. At each iteration, a particle of the swarm updates its position by following the best local particle's position and the best solution of the whole swarm, thus guiding the swarm toward the best solutions. The algorithms has been implemented in C++.

ABC. *Artificial Bee Colony* algorithm as described, e. g., in [97, 98], and implemented in `http://sci2s.ugr.es/eamhco/#Software`. ABC simulates the intelligent foraging behavior of a honey-bee swarm. It provides a population-based search procedure in which candidates (foods positions) are modified by the artificial bees with time. The bee's aim is to discover the places of food sources with higher nectar amounts (related to the objective function). In an ABC system, some of the artificial bees (employed and onlooker bees) choose food sources depending on the experience of themselves and their nest mates and adjust their positions. Other bees (scouts) fly and choose the food sources randomly without using experience. Therefore, during the work, the ABC system combines the local search (carried out by employed and onlooker bees) with the global one (managed by onlookers and scouts), thus attempting to balance exploration and exploitation processes.

FA. *Firefly* algorithm as described in [224, 225] (see also [52] for implementation `http://github.com/firefly-cpp/Firefly-algorithm--FFA-`). FF belongs to the swarm intelligence algorithms and is inspired by the flashing behavior of fireflies. Each firefly (candidate solution) flashes its lights with some brightness (associated with the objective function). This light attracts other fireflies within its neighborhood. This attractiveness depends on the (Euclidean) distance r between the two fireflies and is determined by $\beta(r) = \beta_0 e^{-\gamma r^2}$, where β_0 is the attractiveness at $r = 0$. Hence, the search domain is explored by moving the fireflies towards more attractive neighbors (with some randomized moves allowed), thus improving the current best solution to problem (2.1).

SA. *Simulated Annealing* algorithm as described, e.g., in [1, 124]. It is a variance of widely used Monte-Carlo Methods. It is based on the imitation of the physical process of the crystallization of substances. Algorithm generates a sequence of points $x_0, x_1, ..., x_k$ starting from an initial random point x_0. To select the next trial point x_{k+1} it is necessary to realize a uniform roulette and to make a choice depending the result from this roulette, like standard Monte-Carlo methods. At each iteration i the acceptance of a new point is a probabilistic procedure

Accepted(x). It is based on Metropolis Monte Carlo method proposed in [134] and was extended to continuous global optimization problems. The algorithm has been implemented in C++.

2.1.2 Results of the comparison

In this subsection, traditional ways to compare the algorithms are presented. In the first series of experiments, 5 nature-inspired algorithms DE, PSO, ABC, FA and SA are compared with 5 Lipschitz global optimization algorithms Geom-AL, Geom-GL, Inf-GL, Geom-LTM and Geom-LTIMP, with with the reliability parameter $r = 2.0$ for Inf-GL, $r = 1.2$ for Geom-GL, and $r = 1.1$ with $\xi = 10^{-6}$ for Geom-LTM and Geom-LTIMP, over the set of 24 test problems. The first 20 test problems are from Appendix A and the last 4 test problems are from the Table 1.13.

The population number in the considered one-dimensional case was set equal to 10 for all the population-based methods (i. e., DE, PSO, ABC, and FA algorithms), as recommended, e. g., in [157]. Since they require a random initialization of the initial population, $m = 100$ different runs (with different randomly chosen initial populations) were executed by a method for minimizing a particular test function.

A percentage of failed runs (indicating in some sense the method's reliability) was calculated for all the nature-inspired methods over m independent minimizations of each test function. Thus, for each test problem, both the average number Avg of the performed trials t_j, $1 \leq j \leq m_s$, over m_s successful runs of a method ($1 < m_s \leq m$) and the standard deviation

$$\mathrm{StDev} = \sqrt{\frac{1}{m_s - 1} \sum_{j=1}^{m_s} (t_j - \mathrm{Avg})^2}$$

were registered over the total 100 runs of the method, as often done for metaheuristic algorithms. Obviously, by taking into consideration all m runs (and not only successful m_s ones), the average and standard deviation would increase (sometimes, significantly) for a concrete test problem.

All the metaheuristic methods described above were analyzed and adapted to the one-dimensional case as follows. First, it was observed that a strategy of bound handling is very important for the performance of methods. Different strategies were analyzed and the one with the best performance was chosen. So, for each metaheuristic algorithm, first two trials were executed at the boundaries a and b from (2.1). Then, the bound handling strategy, which is known as "Reflect-Z" (see, e. g., [84]), was used to keep trials within

the boundaries. Then, in order to keep the random mutations for each solutions, small perturbations (depending on the method) to each new solution were produced. Other characteristics were taken as was recommended by the authors of the methods.

In particular, the bound handling strategy "Reflect-Z" is determined as follows. Let x^k be the next trial point, generated by an algorithm. If x^k is out of the interval $[a, b]$ from (2.1), then it is reflected from the respective boundaries, till the point belongs to the search interval, i. e.,

while $x^k \notin [a, b]$

$$\textit{if } x^k < a \textit{ then } x^k = 2 \cdot a - x^k,$$
$$\textit{else if } x^k > b \textit{ then } x^k = 2 \cdot b - x^k.$$

end while

Parameters of the nature-inspired algorithms have been set as follows. The values $F = 0.7$ and $CR = 0.5$ of the control parameters of DE have been used in the experiments, as recommended, e. g., in [157]. Cognitive φ_l and social φ_g parameters of PSO have been set to 2.0, while the inertia weight ω and the maximal velocity v_{max} have been set to 0.6 and 15% of the search space, respectively. The parameter *limit* of ABC have been set to 5 as in previous experiments. The parameters of FA have been set as follows: attractiveness parameter β_0 was set equal to 1.0, the absorbtion coefficient γ was set equal to $0.01/\sqrt{l}$, and the randomization parameter α was set equal to $0.005l$, as recommended, e. g., in [225], where l is the average scaling factor of problem (2.1) ($l = \sum_{i=1}^{N}(b_i - a_i)/N$ for the hyperinterval $D = [a, b]$ in (2.1), with $N = 1$ in our case). Finally, the parameters of SA have been set as follows. The decreasing static cooling schedule T_k, $k \geq 1$, was realized by following [1] as $T_k := cT_{k-1}$, where c is the cooling factor (usually chosen between 0.8 and 0.99). The initial temperature T_0 was set equal to 10 and c was set equal to 0.8 in order to obtain a faster cooling. The parameter σ, as considered in [124], was set equal to 10% of the respective search interval for each component (one in our case) of the candidate solution, in order to avoid a chaotic behavior and to generate solutions more closely to the previous one. Moreover, T_{min} from the re-annealing procedure (see, e. g., [124]) was set equal to 0.001 (to avoid too many re-annealing runs) and the number of evaluations on each temperature T_k was set equal to 100.

For all the one-dimensional tests, a global minimizer $x^* \in D = [a, b]$ from (2.1) was considered to be found (and, therefore, the problem (2.1) solved) when the tested algorithm generated a trial point x' in an ε-neighborhood of

x^*, i. e.,

$$|x' - x^*| \leq \varepsilon(b - a). \tag{2.2}$$

An algorithm stopped either when the maximal number of trials equal to 10 000 was reached, or when condition (2.2) was satisfied. Such a type of stopping criterion is acceptable only when the global minimizer x^* is known, i. e., for the case of test functions. When a real black-box objective function is minimized and global minimization algorithms have an internal stopping criterion (this is the case of the considered DBA-LGO methods and is not the case of the aforementioned nature-inspired algorithms), they execute a number of iterations after a 'good' estimate of $f(x)$ has been obtained in order to demonstrate a 'goodness' of the found solution (see, e. g., the related discussion in [188]). The maximal number of trials was used in the averages calculations when condition (2.2) was not satisfied for some tests in the case of the nature-inspired methods.

In view of the high computational complexity of each trial of the objective function in real-world black-box optimization problems, the methods were compared in terms of the numbers of evaluations of $f(x)$ required to solve the test problems. The maximum number of trials was set equal to 10 000. Two values $\varepsilon = 10^{-4}$ and $\varepsilon = 10^{-5}$ were used in (2.2).

In Tables 2.1 and 2.2, results of numerical experiments on the 24 test functions for the metaheuristics with stopping criterion (2.2) are reported, where the obtained numbers of trials are given for the accuracy ε equal to 10^{-4} and 10^{-5}, respectively. To estimate these numbers, the following worst-case results produced by a pure random search method on this test set can be taken into consideration: Avg = 3051.5, StDev = 2434.5, the average number of fails equal to 10.7% for $\varepsilon = 10^{-4}$; and Avg = 6058.3, StDev = 2178.3, the average number of fails equal to 76.1% for $\varepsilon = 10^{-5}$.

It can be observed from Tables 2.1 and 2.2 that when the accuracy was increased, the numbers of trials performed to stop the methods (together with the averages and standard deviations), as well as the numbers of fails, were also increased. The methods DE and PSO were less sensitive to the accuracy increment with respect to the other metaheuristics. On a higher accuracy, their performance (expressed by the numbers of trials and the numbers of successful runs) on the considered tests was significantly better than that of the ABC, FA, and SA methods. Moreover, while the ABC method's behavior was generally acceptable on the used test problems, a poor performance of the FA and SA algorithms can be noticed from Tables 2.1 and 2.2, especially on a higher accuracy (see Table 2.2). Particularly, the SA method (in its standard sequential form as indicated in the previous subsection) performed too many trials when solving all the tests for both the accuracies, with high values of the

standard deviation and many unsuccessful runs, thus resulting not suitable to tackle multiextremal one-dimensional functions. To better capture the multimodality of the objective functions, the usage of some population-based versions of the SA algorithm can be suggested.

Results of numerical experiments on the same 24 test functions for the DBA-LGO methods with the heuristic stopping criterion (2.2) are reported in Table 2.3, where the trial numbers are given for both the accuracies $\varepsilon = 10^{-4}$ and $\varepsilon = 10^{-5}$. As expected, the numbers of trials performed to stop the methods were increased when the accuracy coefficients were reduced. This increment, however, was less pronounced for the Geom-LTM and Geom-LTIMP algorithms using the local tuning approach since they balance the local and global information about $f(x)$ in a smart way during their work. These two algorithms were better, in average, among the considered DBA-LGO methods on the given test set. Moreover, the DBA-LGO methods behaved usually much better (except for some singular cases regarding the methods Geom-AL and Geom-GL with global estimates of the Lipschitz constants) than the considered nature-inspired algorithms not only when they stopped due to criterion (2.2), as reported in Table 2.3 (that can be used only for benchmark tests with known solutions), but when their internal stopping criteria was satisfied too, i.e., when the length of the interval for the next subdivision is smaller, than ϵ (see the first Chapter for details):

$$x_t - x_{t-1} \leq \epsilon, \tag{2.3}$$

where $[x_t, x_{t-1}]$ is the interval for the next subdivision.

This fact is extremely important for solving practical black-box problems when some guarantee of the solution is required. Therefore, the considered DBA-LGO methods can be suggested as good candidates for solving practical multiextremal optimization problems.

#	DE			PSO			ABC			FA			SA		
	Avg	StDev	Fails	Avg	StDev	Fails	Avg	StDev	Fails	Avg	StDev	Fails	Avg	StDev	Fails
1	**145.5**	42.8	0%	178.2	71.1	0%	201.1	118.4	0%	645.8	635.5	0%	1116.9	1214.0	0%
2	**177.8**	47.9	0%	200.5	65.4	0%	182.1	113.7	0%	412.6	318.2	0%	2105.1	1966.2	0%
3	362.3	151.0	0%	**233.6**	109.6	0%	663.4	691.5	0%	1650.8	1689.1	1%	1287.3	1258.0	0%
4	158.3	41.2	0%	194.4	70.3	0%	**145.1**	82.0	0%	237.8	161.3	0%	2395.7	1960.2	0%
5	216.7	54.4	2%	288.7	227.2	6%	452.0	370.5	0%	**183.3**	95.3	7%	2006.4	1181.5	1%
6	**174.4**	39.1	0%	202.5	70.3	0%	198.1	142.8	0%	1807.3	1572.0	7%	2089.0	1408.3	0%
7	**183.8**	52.8	0%	194.8	67.5	0%	228.1	146.2	0%	386.9	320.7	2%	1855.9	1395.8	0%
8	340.1	136.7	0%	**229.1**	71.5	0%	521.9	465.3	0%	1464.2	1310.3	1%	1303.1	1180.8	0%
9	**182.3**	39.5	0%	213.6	83.5	0%	474.7	336.7	0%	912.5	807.3	0%	2265.2	1862.5	2%
10	**155.9**	43.2	0%	187.4	68.4	0%	310.0	250.6	0%	649.9	601.2	0%	1405.4	1108.5	0%
11	256.7	91.8	0%	221.2	91.4	0%	**196.1**	143.8	0%	591.9	559.1	0%	1686.0	1353.2	0%
12	261.1	95.6	0%	**195.1**	72.0	0%	201.6	153.7	0%	488.5	439.6	0%	1587.9	1404.1	0%
13	163.8	36.8	0%	189.1	76.0	0%	**162.9**	95.1	0%	210.1	152.0	0%	2655.6	1764.9	3%
14	178.5	39.8	0%	**154.9**	62.0	4%	442.7	363.0	0%	318.5	215.2	0%	2524.8	1160.8	2%
15	**160.7**	41.3	0%	187.9	68.5	0%	172.2	112.6	0%	716.9	610.9	0%	1997.8	1566.4	1%
16	**160.5**	42.5	0%	188.6	65.3	0%	163.8	86.9	0%	502.0	422.3	0%	1619.1	1339.9	0%
17	253.3	93.4	0%	**200.5**	72.8	0%	328.2	268.4	0%	697.0	624.3	0%	1206.5	1261.7	0%
18	163.5	48.1	0%	197.7	70.2	0%	**162.3**	98.9	0%	457.8	400.3	0%	2023.5	1483.0	0%
19	**145.7**	46.3	0%	187.3	65.4	0%	173.8	120.2	0%	504.2	500.0	0%	1876.2	1402.7	2%
20	**156.6**	50.1	0%	192.2	67.6	0%	162.9	98.3	0%	1430.9	1339.6	1%	2731.0	2020.2	3%
21	255.1	79.9	0%	**218.5**	89.0	0%	237.8	181.4	0%	469.7	403.7	0%	1686.9	1515.7	0%
22	**161.9**	43.1	0%	187.4	68.9	0%	410.8	413.7	0%	939.5	947.1	0%	1487.4	1106.9	0%
23	244.5	73.2	0%	215.6	83.0	0%	379.7	306.5	0%	**160.3**	94.5	0%	1640.2	1327.4	0%
24	688.1	304.7	1%	**404.4**	647.5	0%	1456.6	1302.7	0%	457.2	266.5	0%	1400.7	1230.3	0%
Avg	222.8	72.3	0.1%	**211.0**	104.4	0.4%	334.5	269.3	0.0%	679.0	603.6	0.8%	1831.4	1436.4	0.6%

Table 2.1: Numbers of trials for metaheuristics on 24 univariate problems, accuracy $\varepsilon = 10^{-4}$ in (2.2)

#	DE			PSO			ABC			FA			SA		
	Avg	StDev	Fails	Avg	StDev	Fails	Avg	StDev	Fails	Avg	StDev	Fails	Avg	StDev	Fails
1	**210.3**	47.0	0%	323.8	91.3	0%	402.5	300.0	0%	3817.0	2903.0	13%	4390.5	2828.2	45%
2	**242.9**	48.7	0%	329.2	92.7	0%	382.7	279.7	0%	2891.8	2287.4	4%	4809.0	2739.1	51%
3	541.5	162.0	0%	**374.6**	126.8	0%	2566.1	2407.3	1%	4640.0	2817.9	51%	4090.5	2972.4	55%
4	**223.4**	44.7	0%	335.4	92.4	0%	288.0	210.5	0%	1245.5	1091.9	0%	4570.0	2553.4	59%
5	**283.8**	61.2	2%	424.9	235.4	6%	1197.1	981.9	0%	900.3	767.5	7%	4587.1	2337.1	61%
6	**234.7**	50.6	0%	333.8	96.0	0%	474.0	396.0	0%	4403.1	2658.5	58%	4266.3	2452.3	52%
7	**253.5**	51.3	0%	333.4	90.5	0%	540.9	488.4	0%	2985.0	2448.2	4%	4385.1	2384.1	55%
8	496.3	172.0	0%	**372.4**	121.1	0%	2098.4	1956.8	2%	4750.8	2958.4	52%	4029.5	3311.0	46%
9	**245.6**	40.3	0%	349.0	104.9	0%	1103.4	997.6	0%	4041.6	3009.4	35%	4559.2	2702.3	54%
10	**225.6**	46.5	0%	337.4	98.9	0%	706.8	611.2	0%	4229.1	3019.4	29%	4471.0	2680.6	44%
11	380.4	118.8	0%	**357.6**	100.3	0%	446.3	352.4	0%	3158.0	2183.2	15%	5485.0	2992.9	40%
12	395.2	120.2	0%	**336.3**	100.9	0%	527.7	471.0	0%	3735.5	2762.7	14%	3946.1	2806.3	48%
13	**219.2**	48.4	0%	333.5	108.7	0%	310.6	216.2	0%	692.6	463.7	0%	4275.1	2553.5	65%
14	**240.2**	46.3	0%	300.6	84.3	4%	1404.8	1425.7	0%	2283.9	1867.9	2%	4338.9	2071.3	66%
15	**220.7**	43.2	0%	327.2	104.1	0%	401.6	370.0	0%	3954.9	2619.6	19%	4645.4	2774.8	56%
16	**229.6**	46.6	0%	322.6	80.2	0%	334.2	220.4	0%	2602.1	2092.5	11%	4450.6	2764.8	48%
17	378.4	116.4	0%	**327.6**	97.5	0%	1024.1	935.3	0%	4258.6	2843.5	12%	3806.7	2822.2	36%
18	**227.2**	50.4	0%	326.4	86.1	0%	317.1	255.3	0%	3126.5	2410.2	6%	4482.6	2162.6	51%
19	**218.4**	42.6	0%	325.5	82.8	0%	413.8	362.6	0%	2937.7	2211.6	9%	4743.1	2920.3	45%
20	**231.2**	47.7	0%	322.7	89.3	0%	342.7	249.7	0%	4595.7	3057.0	55%	4556.1	1981.2	63%
21	393.5	129.4	0%	**336.0**	116.0	0%	664.8	576.4	0%	3651.8	2743.4	12%	4697.8	2857.7	46%
22	**233.0**	37.8	0%	314.4	82.4	0%	1160.5	1057.1	0%	4265.2	3073.2	39%	4081.5	2577.0	52%
23	**310.9**	72.5	0%	327.7	122.2	0%	991.8	857.4	0%	814.6	625.7	0%	4564.0	2977.8	50%
24	813.4	253.8	1%	**476.0**	690.3	0%	3881.1	2454.2	40%	1255.5	900.1	0%	5181.9	2812.7	39%
Avg	**310.4**	79.1	0.1%	343.7	129.0	0.4%	915.9	768.0	1.8%	3134.9	2242.3	18.6%	4475.5	2668.2	51.1%

Table 2.2: Numbers of trials for metaheuristics on 24 univariate problems, accuracy $\varepsilon = 10^{-5}$ in (2.2)

#	accuracy $\varepsilon = 10^{-4}$ in (2.2)					accuracy $\varepsilon = 10^{-5}$ in (2.2)				
	Geom-AL	Geom-GL	Inf-GL	Geom-LTM	Geom-LTIMP	Geom-AL	Geom-GL	Inf-GL	Geom-LTM	Geom-LTIMP
1	**26**	38	41	30	28	151	201	175	**40**	**40**
2	65	40	38	32	**26**	230	134	93	46	**38**
3	88	132	121	123	**32**	97	246	178	130	**44**
4	208	58	133	26	**22**	414	334	266	47	**42**
5	51	**32**	35	43	36	51	140	129	51	**48**
6	66	67	**20**	25	36	130	102	101	56	**44**
7	**13**	75	33	31	24	155	75	163	44	**24**
8	99	109	89	98	**20**	99	138	89	112	**32**
9	50	51	49	**29**	30	169	188	90	48	**44**
10	40	70	59	36	**26**	40	264	110	52	**38**
11	130	63	79	62	**24**	388	253	205	68	**44**
12	131	85	96	45	**26**	507	96	96	75	**38**
13	230	172	78	55	**28**	994	731	669	70	**46**
14	65	57	46	33	**28**	109	77	100	41	**36**
15	143	356	55	36	**32**	143	356	952	63	**42**
16	167	278	193	68	**32**	706	278	193	88	**44**
17	200	117	**35**	77	76	390	581	**35**	109	84
18	38	36	121	**27**	28	304	151	121	**27**	36
19	48	51	31	**21**	24	48	205	117	**37**	38
20	317	36	43	38	**12**	424	77	70	45	**36**
21	26	30	26	22	**18**	29	35	29	31	**21**
22	83	50	52	36	**30**	83	117	52	43	**42**
23	45	51	40	**30**	32	106	83	85	50	**44**
24	132	166	167	131	**64**	132	166	182	131	**64**
Avg	102.5	92.5	70.0	48.1	**30.6**	245.8	209.5	179.2	62.7	**42.0**

Table 2.3: Numbers of trials for DBA-LGO methods on 24 univariate problems, accuracies $\varepsilon = 10^{-4}$ and $\varepsilon = 10^{-5}$ in (2.2)

#	accuracy $\epsilon = 10^{-4}$ in (2.3)					accuracy $\epsilon = 10^{-5}$ in (2.3)				
	Geom-AL	Geom-GL	Inf-GL	Geom-LTM	Geom-LTIMP	Geom-AL	Geom-GL	Inf-GL	Geom-LTM	Geom-LTIMP
1	149	138	127	**37**	**37**	595	468	501	50	**49**
2	155	134	135	**36**	38	457	513	373	**49**	**49**
3	196	228	224	145	**132**	577	562	504	176	**165**
4	413	334	379	45	**43**	1177	972	1076	57	**56**
5	151	140	126	**46**	53	383	360	334	**57**	63
6	129	102	101	**58**	59	301	324	239	**70**	**70**
7	153	138	115	41	**40**	575	412	318	**53**	54
8	187	196	188	**126**	**126**	485	510	477	164	**157**
9	119	132	125	**44**	45	469	368	339	55	**53**
10	203	194	157	**43**	45	571	522	435	**55**	58
11	367	364	405	**74**	77	1099	1156	1153	**100**	**100**
12	327	344	271	**71**	72	993	954	918	**93**	**93**
13	993	516	472	**73**	75	2833	1654	1351	**93**	97
14	145	136	108	**43**	47	379	358	349	**56**	58
15	629	714	471	**62**	65	2513	2010	1893	89	**79**
16	997	570	557	**79**	**79**	2855	2304	1592	102	**97**
17	549	506	470	**100**	115	2109	1630	1484	**125**	140
18	303	268	243	44	**43**	849	845	684	**55**	**55**
19	125	143	117	39	**37**	499	403	336	**49**	52
20	493	76	70	**43**	**43**	1017	200	171	**53**	54
21	34	38	**31**	33	35	37	44	**35**	37	39
22	151	116	124	**42**	51	395	322	333	**54**	64
23	105	82	64	**49**	57	261	202	215	**59**	71
24	271	271	**182**	219	215	332	313	229	232	**228**
Avg	306.0	245.0	219.3	**66.3**	67.9	906.7	725.3	639.1	**82.6**	83.4

Table 2.4: Numbers of trials on 24 problems from Appendix A for DBA-LGO methods stopped by their internal criteria with the accuracy coefficients $\epsilon = 10^{-4}$ and $\epsilon = 10^{-5}$ in (2.3)

In the second series of the experiments, the same methods with the same parameters have been compared over the set of 100 randomly generated Pintér test functions. Similar conclusions with regard to the tested methods can be done also on this class of test problems. Results of numerical experiments with all the methods on this multiextremal and noisy class are summarized in Table 2.5 (all the numbers were averaged over 100 functions). As estimates, the pure random search results can be considered: Avg = 3360.0, StDev = 304.5, the average number of fails equal to 13.42% for $\varepsilon = 10^{-4}$ in (2.2); and Avg = 6199.7, StDev = 2432.5, the average number of fails equal to 80.37% for $\varepsilon = 10^{-5}$ in (2.2). The neat advantage of the DBA-LGO methods (except the rather theoretical Geom-AL algorithm) with respect to the examined nature-inspired algorithms (among which the DE algorithm was again the best one) can be observed from Table 2.5. The local tuning methods Geom-LTM and Geom-LTIMP behaved particularly well on these noisy-type functions, due to the methods computational schemes tuning on the functions shape.

Algorithm	$\varepsilon = 10^{-4}$ in (2.2)		$\epsilon = 10^{-4}$ in (2.3)		$\varepsilon = 10^{-5}$ in (2.2)		$\epsilon = 10^{-5}$ in (2.3)	
	Avg	StDev	Avg	StDev	Avg	StDev	Avg	StDev
Geom-AL	154.7	48.8	401.6	57.3	419.3	136.4	1080.2	91.2
Geom-GL	68.8	28.5	166.0	44.7	179.3	76.4	489.8	127.6
Inf-GL	54.0	26.8	144.7	39.1	149.0	66.4	423.2	109.3
Geom-LTM	37.7	11.2	**47.6**	**9.8**	48.3	11.0	**59.0**	**9.9**
Geom-LTIMP	**29.6**	**6.8**	54.5	19.2	**39.9**	**6.7**	66.5	21.7
DE	**184.6** (99.99%)	**15.6**	—		**251.4** (99.99%)	**15.1**	—	
PSO	190.8 (99.75%)	26.7	—		318.4 (99.75%)	35.6	—	
ABC	299.9 (100%)	112.3	—		728.0 (99.96%)	358.2	—	
FA	676.6 (100%)	116.8	—		3732.4 (80.05%)	435.7	—	
SA	2249.7 (99.48%)	230.8	—		4787.6 (41.41%)	354.2	—	

Table 2.5: Average numbers of trials and standard deviations over 100 functions (1.16) for the considered DBA-LGO and nature-inspired methods with different stopping criteria (percentage of successful runs are indicated for metaheuristics)

A distribution of trial points generated by all the methods when minimizing function 63 from the class (1.16) with the accuracies $\varepsilon = 10^{-4}$ and $\varepsilon = 10^{-5}$ in (2.2) is illustrated in Fig. 2.1 and Fig. 2.2, respectively (particular runs of the methods DE, PSO, ABC, FA, and SA have been chosen to obtain results close to the averages in Table 2.5). It can be seen from these Figures, how the density of trial points was increased around the global minimizer x^* and was higher for a higher precision (see Fig. 2.2). The methods ABC, FA,

and SA performed many trials around local minimizers of $f(x)$, thus manifesting an excessive locally oriented behavior on such multiextremal functions. Trial points generated by DE and PSO were distributed in a more smart manner with respect to the other metaheuristics and even better than the distribution obtained by the Geom-AL algorithm. It is also evident from Figures 2.1–2.2 that once the tested methods determined a neighborhood of the global minimizer, an additional number of trials (which was higher for a higher accuracy as in Fig. 2.2) was still required to stop each method, as it often happens in global optimization. This density, however, was not excessive for the Geom-LT and Geom-LTI methods, on both the accuracies. In fact, since these methods usually balance better the local and global information during the search for the global minimizer x^*, they captured a close neighborhood of x^* much faster than the other considered methods.

In the third series of experiments, the same nature-inspired algorithms have been compared on ten constrained non-differentiable functions from [50] with the following methods:

Pen-AL. The Geom-AL method combined with the penalty approach used to reduce the constrained problem to a box-constrained one (see [50, 193] for the choice of the penalty coefficients).

Index-AL. The index branch-and-bound algorithm from [184] using the index scheme (see, e. g., [193, 215]; in this scheme a constrained problem is reduced to a discontinuous box-constrained one without introducing additional parameters or variables) in combination with the branch-and-bound approach and the known Lipschitz constants.

Index-LT. The geometric method from [193] based on the index approach and using "Maximum" local tuning.

In this class of benchmarks, problems 1–3 have one constraint, problems 4–7 have two constraints, and problems 8–10 have three constraints (see `http:\\wwwinfo.dimes.unical.it\~yaro\constraints.html` for a detailed description of this constrained test set, where all the problems are illustrated and their Lipschitz constants and global solutions are reported). In these experiments, all nature-inspired algorithms were used in combination with the penalty approach (the same penalty coefficients from [50] as for the Pen-AL were taken to reduce the constrained problems to the box-constrained ones).

Results of numerical experiments with the considered methods on the constrained problems are reported in Table 2.6. Here, Lipschitz methods were stopped by their internal stopping criterion (2.3), while the metaheuristics were stopped by condition (2.2). Criterion (2.3) is aimed at proving the

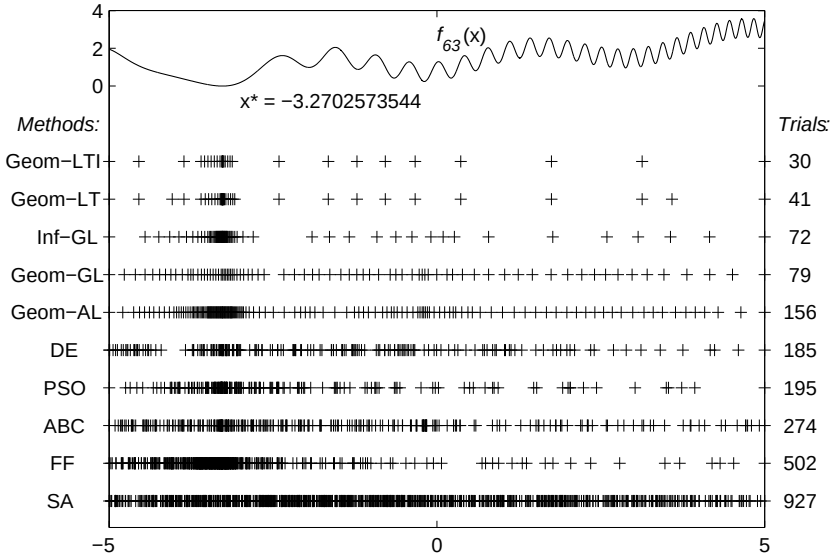


Figure 2.1: Graph of the function number 63 from (1.16) and the trial points (+) generated by the tested methods while minimizing this function with the accuracy $\varepsilon = 10^{-4}$ in (2.2)

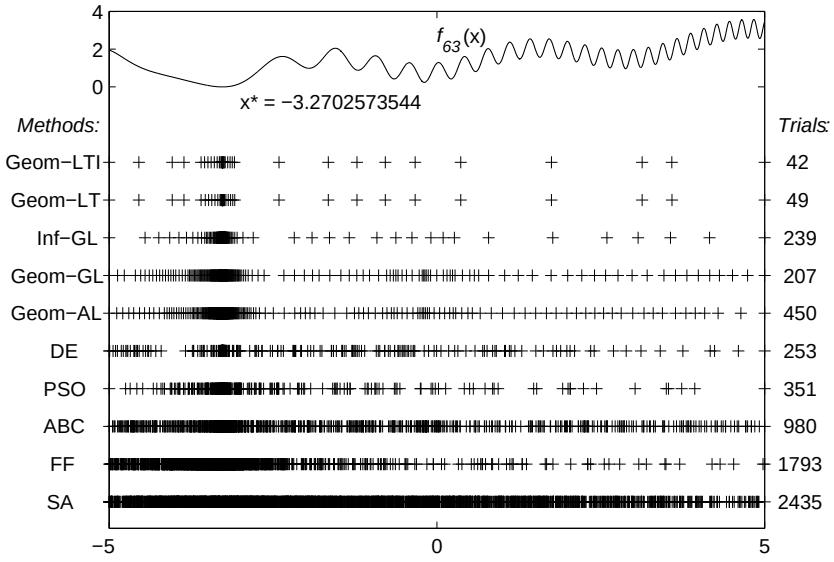


Figure 2.2: Graph of the function number 63 from (1.16) and the trial points (+) generated by the tested methods while minimizing this function with the accuracy $\varepsilon = 10^{-5}$ in (2.2)

global optimality of the found solution and, therefore, is computationally more expensive than criterion (2.2). However, also when stopped by this more challenging criterion, Lipschitz methods based on the index approach (Index-AL and Index-LT) performed very well with respect to the metaheuristics also on this constrained test set. Again, the DE algorithm manifested a better performance over the tested metaheuristics.

In order to illustrate graphically the methods performance on the constrained test set, a distribution of trial points generated by all the methods when minimizing function 9 from the constrained benchmark set [50] with the accuracies 10^{-4} and 10^{-5} is illustrated in Fig. 2.3 and Fig. 2.4, respectively (these accuracies are referred to condition (2.3) for the DBA-LGO methods Pen-AL, Index-AL, and Index-LT and to condition (2.2) for the considered metaheuristics, for which particular runs have been chosen to obtain results close to the averages in Table 2.6). This test problem has three constraints and is defined by the following formulae:

$$\begin{aligned} \min_{x\in[0,4]} f(x) &= 3 - 2\exp\left(-\tfrac{1}{2}\left(\tfrac{22}{5} - x\right)\right)\left|\sin\left(\pi\left(\tfrac{22}{5} - x\right)\right)\right| \\ \text{subject to} & \\ g_1(x) &= \tfrac{4}{5} - \left(\left|\sin\left(\tfrac{24}{5} - x\right)\right| + \tfrac{6}{25} - \tfrac{x}{20}\right) \le 0, \\ g_2(x) &= \begin{cases} 6\left(x - \tfrac{1}{2}\right)^2 - \tfrac{1}{2}, & x \le \tfrac{1}{2} \\ \tfrac{1}{4}\left(x - \tfrac{5}{2}\right), & x > \tfrac{1}{2} \end{cases} \le 0, \\ g_3(x) &= 3\left(\exp\left(-\left|\sin\left(\tfrac{5}{2}\sin\left(\tfrac{11}{5}x\right)\right)\right|\right) + \tfrac{1}{100}x^2 - \tfrac{1}{2}\right) \le 0, \end{aligned} \tag{2.4}$$

leading to disjoint feasible region $\Omega = \{x \in [0,4] : g_i(x) \le 0, i = 1,2,3\}$ shown by three continuous horizontal bold lines in Figures 2.3–2.4. The global minimizer of the objective function $f(x)$ is located at the point $x^* = 0.95024$. The line of symbols '+' drawn under the graphs of the function $f(x)$ (solid line) and the constraints $g_i(x), i = 1,2,3$ (dashed lines) in these Figures shows points at which trials have been executed by the corresponding methods.

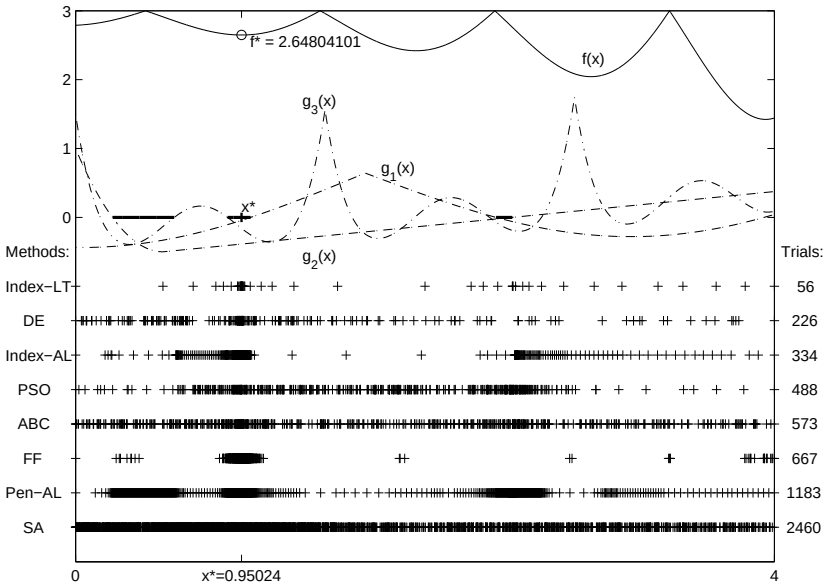


Figure 2.3: Graphs of the objective function (solid line) and three constraints (dashed lines) from (2.4) and the trial points (+) generated by the tested methods with accuracy 10^{-4}

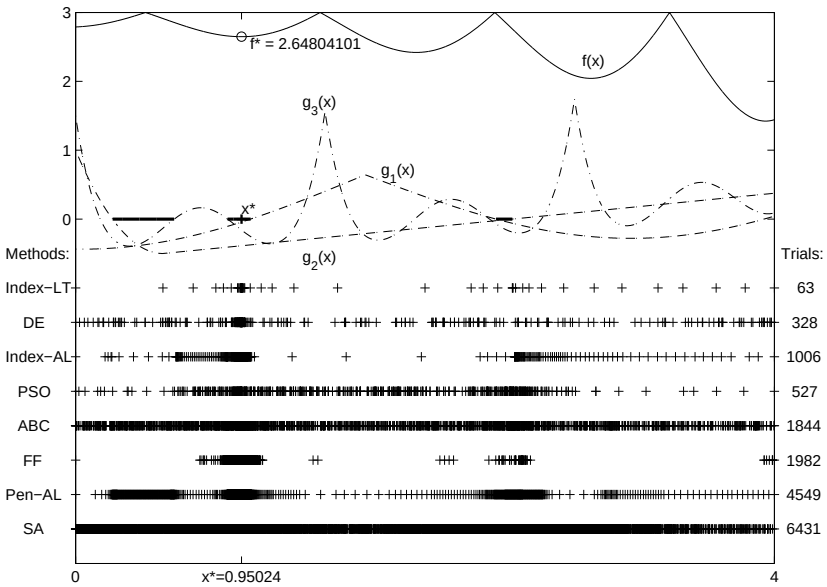


Figure 2.4: Graphs of the objective function (solid line) and three constraints (dashed lines) from (2.4) and the trial points (+) generated by the tested methods with accuracy 10^{-5}

accuracy $\varepsilon/\epsilon = 10^{-4}$												
#	Pen-AL	Index-AL	Index-LT	DE		PSO		ABC		FA		SA	
				Avg	Fails	Avg	Fails	Avg	Fails	Avg	Fails	Avg	Fails
1	247	51	**44**	206.7	0%	207.5	0%	440.3	0%	647.2	0%	2057.9	1%
2	241	34	**34**	246.2	7%	195.2	0%	1277.6	0%	1501.2	1%	3707.2	18%
3	917	190	**21**	279.2	3%	204.3	4%	2707.3	2%	1354.4	0%	2554.7	10%
4	273	235	**93**	216	0%	210.8	0%	982.4	0%	401.3	0%	2241.4	4%
5	671	283	**132**	339.7	41%	342.9	24%	1635.8	0%	2375.5	3%	2065.9	0%
6	909	629	**74**	231.4	6%	90.4	0%	709.7	0%	419.6	9%	3377.2	19%
7	199	120	**100**	354.1	31%	189.5	61%	2691.5	8%	2513.5	3%	2960.7	18%
8	365	64	**59**	408.6	16%	228.5	3%	2301.8	0%	2570.1	2%	2697.4	11%
9	1183	334	**56**	240.6	3%	370.9	1%	665.1	0%	414.2	0%	3068	5%
10	135	75	**42**	201.2	0%	192.7	0%	466.8	0%	481.9	0%	1863.1	0%
Avg	514	201.5	**65.5**	272.4	10.7%	223.3	9.3%	1387.8	1%	1267.9	1.8%	2659.3	8.6%
accuracy $\varepsilon/\epsilon = 10^{-5}$													
1	419	57	**46**	270.5	0%	355.8	0%	982.1	0%	3495.2	20%	4656.5	49%
2	313	42	**38**	331.4	7%	376.8	0%	3508.1	17%	4235.9	46%	5106.2	75%
3	2127	196	**24**	350.9	3%	353.9	4%	5044.4	53%	4303.9	49%	5454.7	59%
4	861	411	**104**	279.2	0%	347.4	0%	3128.1	3%	2244	1%	4732.5	47%
5	1097	315	**137**	440.4	41%	499.3	24%	3453.8	27%	4963.4	65%	4586.7	63%
6	6367	2719	**96**	289.2	6%	146.5	0%	1802.7	0%	2883.3	13%	4588.7	74%
7	221	127	**110**	457.4	31%	296.2	61%	5057.9	61%	4403.8	37%	3650.7	75%
8	415	69	**62**	483.2	16%	378.8	3%	4238.1	29%	4264.7	60%	4465.1	72%
9	4549	1006	**63**	309.3	3%	531.2	1%	1881.6	1%	2594.4	2%	4579.1	64%
10	169	77	**47**	265.4	0%	326.5	0%	967	0%	2914.5	8%	4535	55%
Avg	1653.8	501.9	**72.7**	347.7	10.7%	361.2	9.3%	3006.4	19.1%	3630.3	30.1%	4635.5	63.3%

Table 2.6: Numbers of trials for 10 constrained problems from [50] with accuracy coefficients $\varepsilon/\epsilon = 10^{-4}$ and 10^{-5} in the internal stopping criterion (2.3) for the DBA-LGO methods and in the heuristic criterion (2.2) for the metaheuristic algorithms

The advantages of the LGO algorithms Index-AL and Index-LT constructed in the framework of the index scheme with respect to the penalty method PEN-AL can be seen from these Figures (confirming the results from Table 2.6). The local tuning method Index-LT demonstrated a significant improvement (more pronounced when the search accuracy increases) in terms of the executed trials with respect to the Index-AL algorithm with a priori given Lipschitz constants. As observed, e. g., in [193], this improvement becomes higher when the difference between estimates of the local Lipschitz constants (for the objective function or for the constraints) becomes greater. This fact is particularly evident if the global minimizer is located inside a feasible region with a small value of the local Lipschitz constant with respect to the global one, as for the considered problem 9 (similar conclusions can be done for problem 6, too; see Table 2.6).

2.2 A systematic comparison using classes of randomly generated test problems

As it has been emphasized previously, the concept of operational characteristics for comparing deterministic algorithms was proposed by Grishagin in [78]. The operational characteristic of a method on a class of test problems is a non-decreasing function that indicates the number of problems solved by this method after each function evaluation within a prescribed trials budget.

It can be asserted that the operational characteristics approach is a very representative technique for comparing deterministic algorithms. However, it has not been used so far for a comparison of stochastic methods. In this Section, it is shown how this approach can be adapted for the study of stochastic methods. In particular, a new comparison methodology developed in this work is applied to the metaheuristic algorithms that have multiple launches with different randomly chosen initial populations (see [196] for details).

2.2.1 Operational zones for comparing metaheuristic and deterministic univariate algorithms

Operational characteristics approach can be illustrated very easily using a graph. For example, the operational characteristics of the methods Geom-AL and Geom-GL on the class of 100 Pintér's test functions from [153] are presented in Fig. 2.5.

It can be seen that after 100 trials the method Geom-AL has solved 17

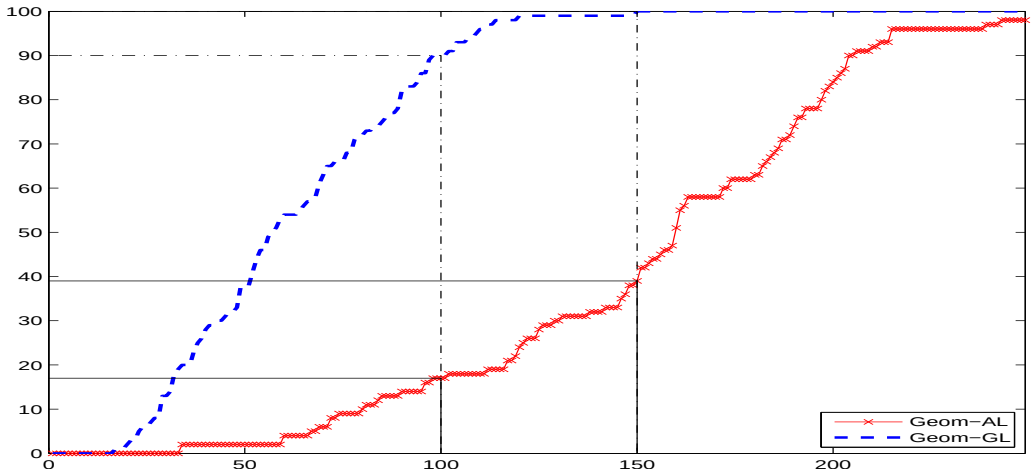


Figure 2.5: Operational characteristics for the methods Geom-AL and Geom-GL

test problems and the method Geom-GL has solved 90 test problems. Then, it can be also seen from Fig. 2.5 that the method Geom-AL has executed 150 trials to solve 39 test problems whereas the method Geom-GL has solved 39 test problems in about 50 trials. To solve all 100 test problems, the method Geom-GL has executed only 150 trials whereas the method Geom-AL has solved only 39 test problems after the computational budget of 150 trials.

This approach has been generalized for the stochastic metaheuristic algorithms in [198] (see also [201] and [196]) as follows. Each stochastic algorithm is launched several times. First, each launch (among the total number of launches set equal to 100 in our experiments) of a metaheuristic method is considered as a particular method. Then, for each of 100 launches of the algorithm the respective 100 operational characteristics are constructed: see Fig. 2.6.a where 100 operational characteristics for the method DE are shown. Finally, the upper and lower bounds of all operational characteristics are constructed as it is illustrated in Fig. 2.6.b.

It should be noticed that these two characteristics (upper and lower bounds) are the best and the worst cases, respectively, for the tested method and they usually are not realized in practice since in general case they can consist of parts of several operational characteristics for all runs.

Theese bounds determine a zone where all constructed operational characteristics are presented and the average operational characteristic for all runs (like in Figure 2.6.c) can be computed. Let us denominate this zone the *operational zone*. It should be noticed that for all considered metaheuristics the operational zones constructed by using 30 and 100 launches differ

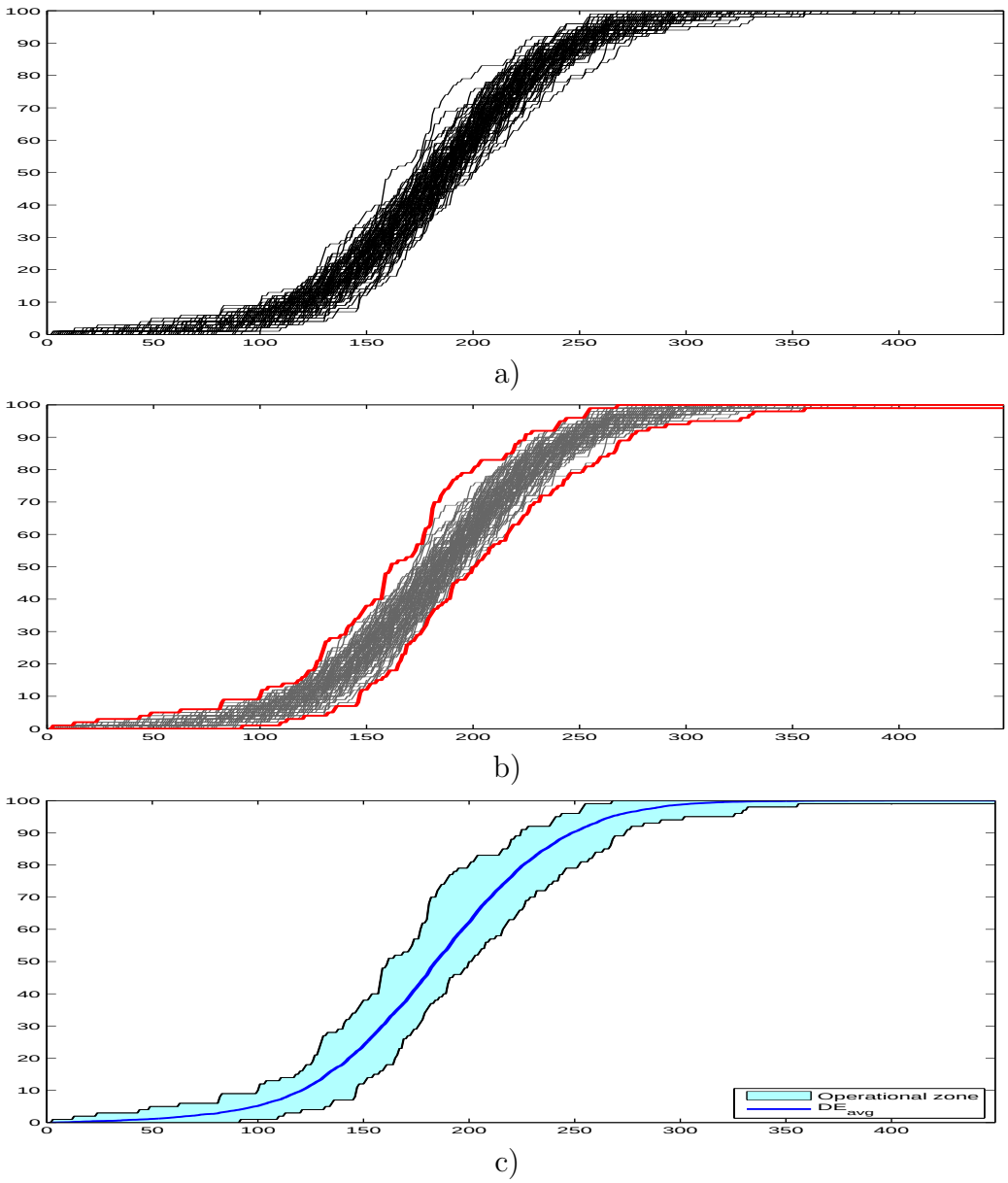


Figure 2.6: (a) 100 Operational characteristics for the DE method; (b) their lower and upper bounds shown in bold; and (c) operational zone with the average operational characteristic

insignificantly.

We can, thus, compare operational zones of stochastic algorithms with operational characteristics of deterministic methods. In this Section, 5 metaheuristic algorithms described above (GA, DE, PSO, ABC and FA) are

compared with several univariate and multidimensional Lipschitz global optimization algorithms on the randomly generated classes of test functions.

In the first series of experiments, numerical experiments on the class of 100 univariate randomized Pintér's test functions (1.16) from [153] were carried out. The nature-inspired algorithms have been adapted to the univariate case as previously using the box-constraints handling strategy "Reflect-Z". In this part, the algorithms GA, DE, PSO, ABC and FA are compared with three univariate Lipschitz global optimization methods described in the first Section: Geom-AL, Geom-GL and Geom-LTIMAP.

For each test function the problem (2.1) was considered to be solved if an algorithm generated a trial point x^k in an ε-neighborhood of the global minimizer x^*, i. e.,

$$|x^k - x^*| \leq \varepsilon(b-a). \tag{2.5}$$

The values $\varepsilon = 10^{-4}$ and $\varepsilon = 10^{-6}$ were used in our experiments. The operational characteristic of method m on the class of Pintér's test problems is defined as the function

$$\phi_m(k) = \phi_m(k-1) + p_m^k, \;\; 1 \leq k \leq k_{\max}, \quad \phi_m(0) = 0,$$

where p_m^k is the number of test problems solved by the method m at k-th trial and $k_{\max}$ is the maximum number of trials (this computational budget was set equal to 10 000 trials in our experiments).

For the tested geometric methods the following values of control parameters were used. For the method Geom-AL that uses an a priori given Lipschitz constants the maximum absolute values of relative differences over the 10^{-7}-grid were taken to obtain an estimate of the Lipschitz constant. The method Geom-GL uses in its work the reliability control parameter r that was set equal to 1.1 in our experiments. The method Geom-LTIMAP uses the reliability parameter r, the technical parameter ξ and the local improvement accuracy δ, that were set equal to 1.1, 10^{-8}, and $\delta = \varepsilon(b-a)$, respectively, where ε is from (2.5).

Parameters of the metaheuristic algorithms were set as follows. Following recommendations from [41] the crossover and mutation probabilities in GA were set equal to 0.95 and 0.25, respectively, in our experiments. The parameter η of "Simulated Binary Crossover" (SBX) was set equal to 4 by following recommendations from [40]. In DE, the exponential crossover as one with a better performance and the mutation strategy "DE/rand/2" from [157] were used. The control parameters of DE were set as follows: the differential weight F from the mutation unit was set equal to 0.7 and the crossover rate (CR) was set equal to 0.5 by following the recommendations from [157]. The cognitive ϕ_l and social ϕ_g parameters of PSO were set equal to 2.0, while

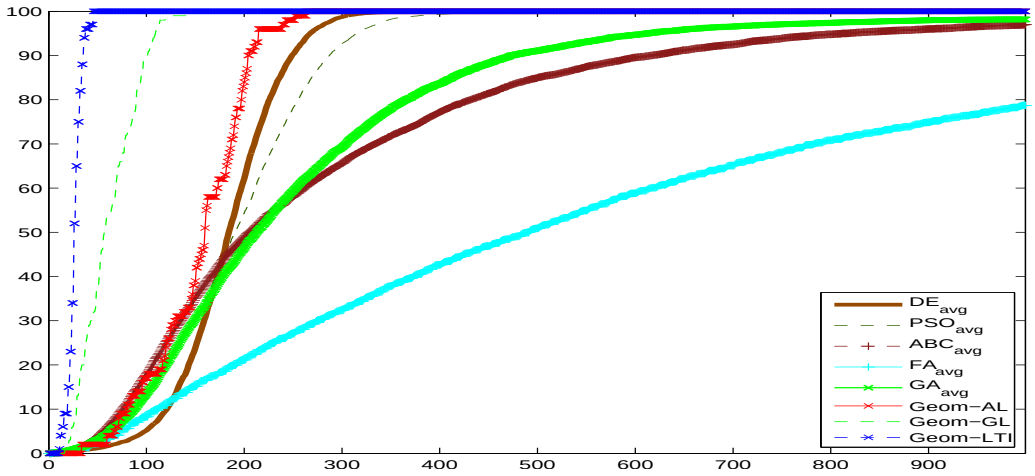
the inertia weight and maximum velocity value were set to 0.6 and 15% of the search domain, respectively. The parameter *limit* of ABC was set to 5 as previously. The attractiveness parameter β_0, the absorbtion coefficient γ and the randomization parameter α of FA were set equal to 1.0, $0.01/\sqrt{l}$ and $0.005l$, respectively, where l is the average scaling factor of the problem (i.e., $l = b - a$ in our one-dimensional case).

First, the operational characteristics of deterministic methods (Geom-AL, Geom-GL and Geom-LTIMAP) were compared with the average operational characteristics of metaheuristic methods. The results are presented in Fig. 2.7 with the accuracies $\varepsilon = 10^{-4}$ and $\varepsilon = 10^{-6}$ from (2.5).

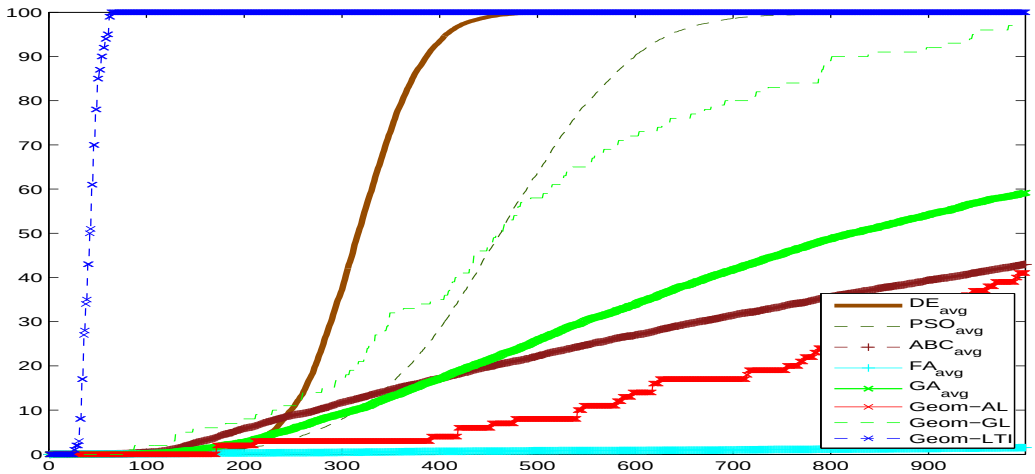
It can be seen from Figures 2.7.a and 2.7.b that the method with local tuning and local improvement (Geom-LTIMAP) demonstrates a better performance with respect to the methods with global estimates of the Lipschitz constant (Geom-AL and Geom-GL). Moreover, the global estimate of the Lipschitz constant (Geom-GL) gives better results with respect to the usage of an a priori given Lipschitz constant (Geom-AL). It can be also seen from Fig. 2.7 that the advantage of the method Geom-LTIMAP becomes more pronounced when the search accuracy increases.

It can be also seen from Fig. 2.7 that, with respect to metaheuristics, the best average performance on the class of Pintér's test functions has been shown by the DE method: it has solved all 100 problems faster in average than the other tested metaheuristic methods. The worst performance on this test class was shown by the FA method that has executed a huge number of trials to solve all test problems.

Then, the operational characteristics of deterministic methods were compared with the operational zones of metaheuristic methods. It can be seen from Figures 2.8, 2.9, and 2.10 that with the accuracy ($\varepsilon = 10^{-4}$) all Lipschitz methods have solved all 100 problems faster than metaheuristic methods even with respect to the upper bound of the metaheuristics operational zones, i. e., with respect to the best unrealizable case. By increasing the accuracy the situation was changed. Using a higher accuracy ($\varepsilon = 10^{-6}$), the best three metaheuristic algorithms (GA, DE and PSO) have solved all test problems much faster in average than the geometric methods with global estimates of the Lipschitz constant (Geom-AL and Geom-GL), being however inferior with respect to the local tuning method Geom-LTIMAP. This happens due to the fact that the Lipschitz constant of the objective function in a small neighborhood of the global minimizer is very small since Pintér's functions are differentiable. So, the methods with global estimates of the Lipschitz constants (Geom-AL and Geom-GL) use too overestimated values of the Lipschitz constant at their final phase, thus slowing down the search. In contrast, the algorithm Geom-LTIMAP has mechanisms of an adaptation



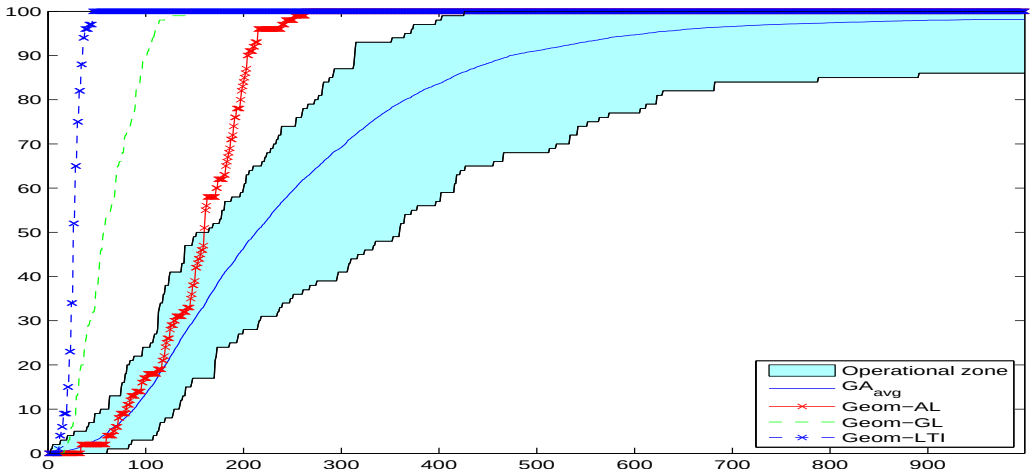
a)



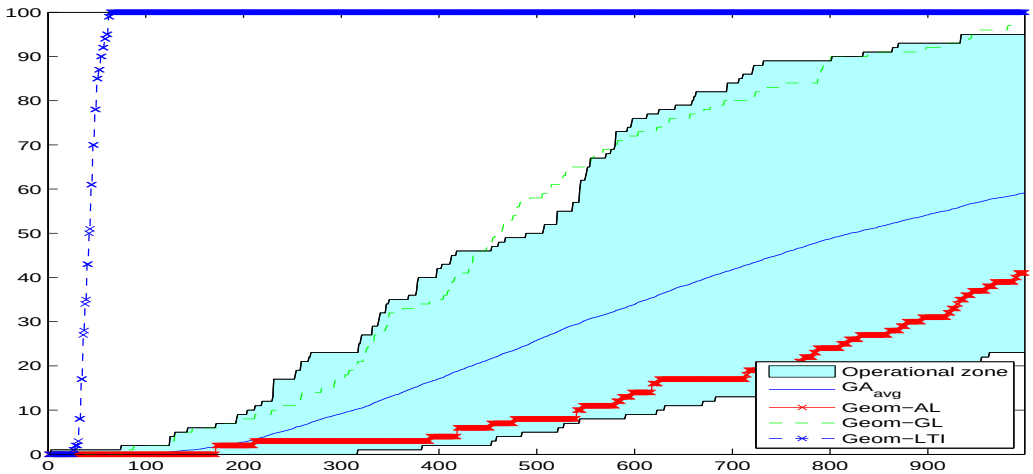
b)

Figure 2.7: Average operational characteristics for metaheuristic methods and operational characteristics for Lipschitz methods with $\varepsilon = 10^{-4}$ (a) and $\varepsilon = 10^{-6}$ (b) from (2.5)

to a local information received during the search and, as a result, it has the possibility to accelerate the final searching phase. Here, one of the advantages of the local tuning techniques can be seen: the method Geom-LTIMAP uses local estimates of the local Lipschitz constants avoiding so a significant increase of the number of trials inherent in the Geom-AL and Geom-GL algorithms at their final search stage. Mechanisms of a local adaptation in the metaheuristic methods also allow them to improve the performance without a significant increase of the costly trials. However, it should be noticed that



a)



b)

Figure 2.8: Operational zones for the GA method and operational characteristics for the geometric methods with $\varepsilon = 10^{-4}$ (a) and $\varepsilon = 10^{-6}$ (b) from (2.5)

in some runs the metaheuristic methods did not find the global solutions getting stuck in the local solutions. Although the number of trials performed by some metaheuristic algorithms can be small in average (as it happens, for example, for the DE method), the number of failed runs puts obstacles in a "good" behavior of the operational zones for such methods.

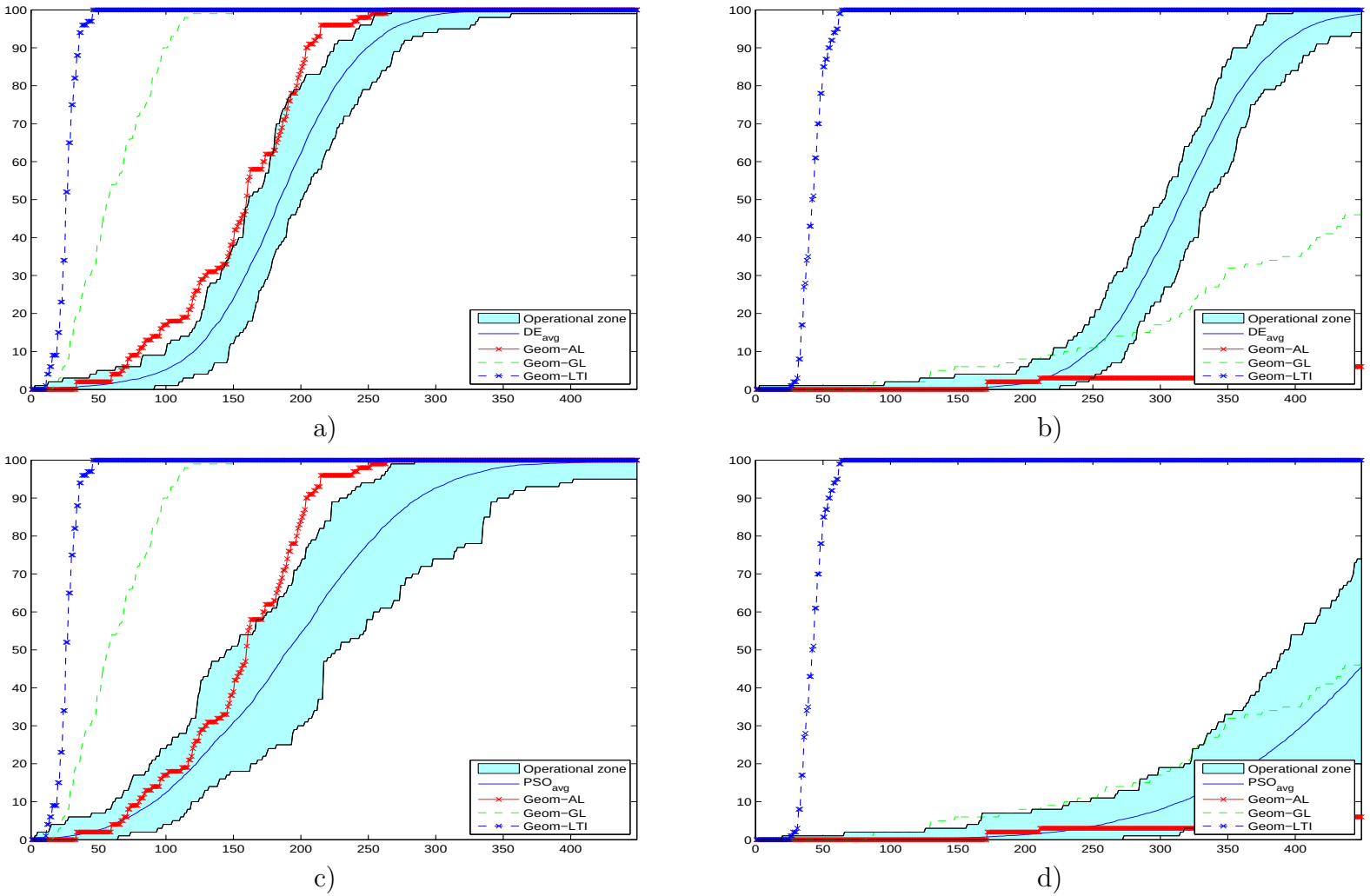


Figure 2.9: Operational zones for the DE and PSO methods and operational characteristics for the geometric methods with $\varepsilon = 10^{-4}$ ((a) for DE and (c) for PSO) and $\varepsilon = 10^{-6}$ ((b) for DE and (d) for PSO) from (2.5)

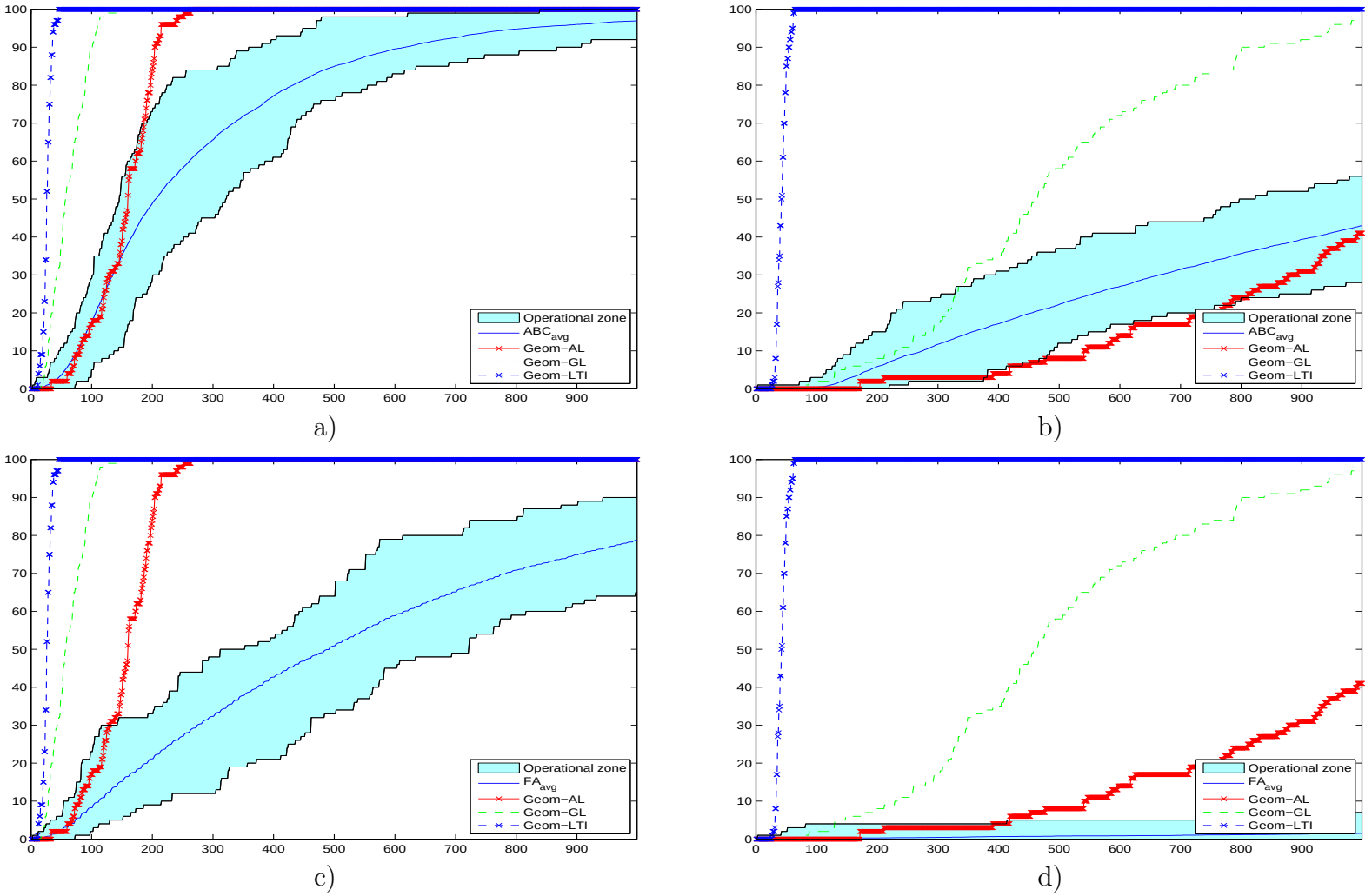


Figure 2.10: Operational zones for the ABC and FA methods and operational characteristics for the geometric methods with $\varepsilon = 10^{-4}$ ((a) for ABC and (c) for FA) and $\varepsilon = 10^{-6}$ ((b) for ABC and (d) for FA) from (2.5)

2.2.2 Techniques for comparing multidimensional stochastic and deterministic methods

In another series of experiments, multidimensional randomly generated test problems are used for a systematic comparison of the methods. In order to make this comparison more reliable, parameters of all tested algorithms were fixed following recommendations of their authors and then were used in all the experiments as follows.

The parameters of GA have been set as follows: the crossover probability was set to 0.95, mutation probabilities for binary and real coded variables were set to 0.2 and 0.25, respectively, in our experiments. $\lceil D/2 \rceil$ binary-coded variables and $\lfloor D/2 \rfloor$ real-coded variables were used. The chromosome length for binary-coded variables was set to 16. Finally, the parameter η of Simulated Binary Crossover (SBX) was set to 4.

The first control parameter F of DE was set to 0.7, while the second one CR was set to 0.5 in the experiments. The strategy "DE/Rand/2/exp" was used as one of the most powerful mutation strategies.

Cognitive ϕ_l and social ϕ_g parameters of PSO were set to their default value 2.0, while the inertia weight was set to 0.6 as for difficult and multimodal functions. The maximal velocity value was set equal to 15% of the longest side of the search domain.

The parameter *limit* of ABC was set to $FoodNumber \cdot N$, where the $FoodNumber$ is the number of food sources, i.e., the number of employed bees and N is the dimension of the problem.

Finally, the attractiveness parameter β_0, the absorbtion coefficient γ and the randomization parameter α of FA were set to 1, $1/\sqrt{l}$ and 0.2, respectively, where l is the average scaling factor of the problem (i.e., $l = \sum_{i=1}^{N}(b_i - a_i)/N$ for the search hyperinterval $\{x = (x_1, ..., x_N) | x_i \in [a_i, b_i]\}$.

In the traditional *local optimization* (see, e.g., [145]), where strong assumptions on the structure of the objective function (such as convexity, continuity, differentiability, etc.) are made, the dimensionality of the solved problem is often a measure of the goodness of optimization algorithms. In contrast, as was proved in [211], if the only information about the objective function $f(x)$ from the global optimization problem (2.1), (1.2) is that it belongs to the class of Lipschitz functions and the Lipschitz constant L is unknown, there does not exist any deterministic or stochastic algorithm that, after a finite number of function evaluations, is able to provide an accurate ε-estimate of the global minimum f^*. That is why in this case instead of the theoretical statement (P1) *Construct an algorithm able to stop in a given time and to provide an ε-approximation of the global minimum f^** the more practical statement (P2) *Construct an algorithm able to stop after a fixed*

number M of evaluations of $f(x)$ and to return the lowest found value of $f(x)$ is used. Under the latter statement, not the dimension of the problem (that is important in local optimization) but the number of allowed function evaluations (often called *budget*) becomes critical. In other words, when one has the possibility to evaluate $f(x)$ M times in the global optimization problem of the dimension 5, 10 or 100, then the quality of the found solution after M evaluations is crucial and not the dimensionality of $f(x)$. This happens because it is not possible to adequately explore the multi-dimensional search region D at this limited budget of expensive evaluations of $f(x)$. For instance, if $D \subset \mathbb{R}^{20}$ is a hypercube, then it has 2^{20} vertices. This means that one million of trials is not sufficient not only to explore well the whole region D but even to evaluate $f(x)$ at all vertices of D. Thus, the statement (P2) makes sense both because in practice the budget is always limited and because the problem under consideration is NP-hard.

As a result, the goal of global optimization methods is often to obtain a better estimate of f^* and x^* given a fixed limited budget of evaluations of $f(x)$. In fact, in global optimization the words "A method has solved a global optimization problem" very often do not mean that the global solution f^* has been found. They mean just that the found solution was better than solutions found by other competitors (this is especially true for highly dimensional global optimization problems where the global solutions are unknown). That is why the possibility to compare the found solutions with the known global optimum offered by the generator of classes of test functions (see, e.g., [65] for details) is very precious. It allows us not only to see that a solution A found by one method is better than a solution B found by another method, but to check whether these solutions are in a prefixed ε-neighborhood of the global optimum, i.e., to consider (P1) instead of (P2).

Two methodologies for comparing global optimization algorithms are applied here: Operational Characteristics for comparing deterministic algorithms and Operational Zones described above. Another methodology called "Aggregated Operational Zones" generalizing ideas of operational characteristics and operational zones to collate multidimensional stochastic algorithms (see [201] for details) is also studied in the next subsection.

First of all, to construct classes of test functions, the popular GKLS generator (see [65]) of multidimensional, multiextremal test functions was used (see also Section 2.3 for details). This generator allows one to generate randomly classes of 100 test problems each having the same dimension, number of local minima, and difficulty. The property making this generator especially attractive consists of the fact that for each function a complete information of coordinates and values of all local minima (including the global one) is provided. Here, 8 different classes from [188] were used. For each dimen-

sion N from $N = 2$ to $N = 5$ two classes (a simple and a hard one) have been generated.

In these experiments, the nature-inspired algorithms given above are compared with three multidimensional Lipschitz global optimization methods: DIRECT method from [94], its locally-biased version DIRECT-L from [59], and the algorithm from [188] based on adaptive diagonal curves (called ADC hereinafter). The results of these three methods have been taken from [189].

Fig. 2.11.a shows operational characteristics for methods DIRECT, DIRECT-L, and ADC. Higher is a characteristic of a method with respect to characteristics of its competitors better is the behavior of this method. Operational characteristics allow us also to see the best performers in dependence on the available budget of evaluations of $f(x)$. For instance, it can be seen from Fig. 2.11.a that if the search budget is less than 14 000 possible trials than DIRECT method shows the best performance whereas for a budget superior to 14 000 the best method is ADC.

As it has been mentioned previously, to build a zone, a tested stochastic method should be launched K times (in our experiments each metaheuristic was launched $K = 100$ times for each of 100 test problems from each of 8 classes) with different randomly chosen populations and a maximum number of trials N_{max} (in our experiments, $N_{max} = 10^6$). Then, each run of a tested metaheuristic was considered as a particular method and its operational characteristic was constructed. The totality of all 100 operational characteristics form the respective operational zone (see Fig. 2.11.b for an operational zone obtained by FA). Then, the upper and the lower boundaries of the zone (shown in Fig. 2.11.b as dark blue curves) can be outlined (notice that they can contain parts of several characteristics) representing the best and the worst performances of the tested method, respectively. The graph for the average performance within the zone can be also depicted (see Fig. 2.12.b where the average performance of FA is shown as a continuous black line inside the yellow operational zone).

Fig. 2.12 shows results on the 5-dimensional simple and hard classes for metaheuristics FA, GA, and ABC. Figs. 2.12.a and b compare, respectively, performance of the three deterministic methods and FA on the simple (with $N_{max} = 2 \cdot 10^4$) and the hard (with $N_{max} = 10^5$) classes. The joint representation of operational zones together with characteristics offers a lot of visual information. It can be seen, for example, in Fig. 2.12.a that operational characteristics of DIRECT and ADC are higher than the upper boundary of the zone of FA and, therefore, on this class deterministic methods have a better performance. Fig. 2.12.b shows that the lower boundary of the FA zone is higher than characteristics of DIRECT and DIRECT-L and, therefore, FA outperforms these competitors. If the budget is less than 30 000 trials (see

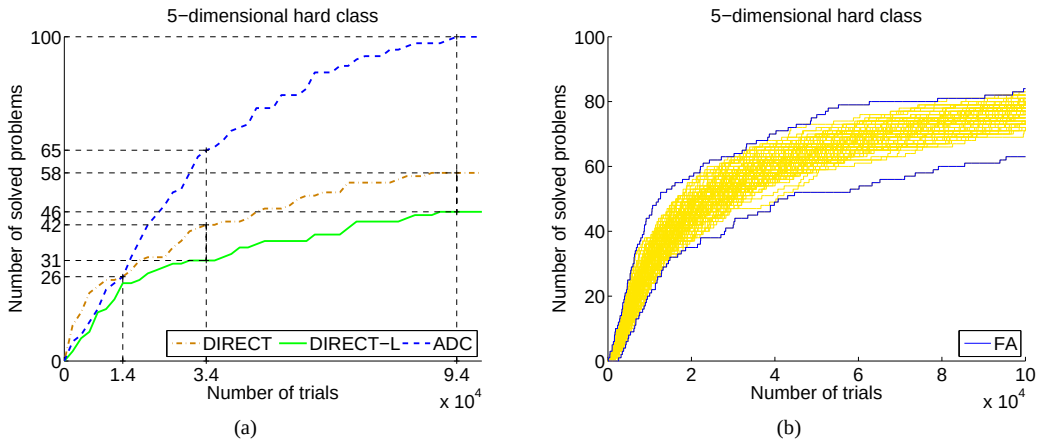


Figure 2.11: Construction of operational characteristics for deterministic methods and of the operational zone for metaheuristic Firefly Algorithm (FA) built on the hard 5-dimensional class of 100 GKLS test functions. (a) Operational characteristics for methods DIRECT, DIRECT-L, and ADC. After executing 34,000 trials DIRECT-L has solved 31 problem, DIRECT 42 problems, and ADC 65 problems; after performing 94,000 trials DIRECT-L has solved 46 problems, DIRECT 58 problems, and ADC all 100 problems. (b) The operational zone built using 10,000 runs performed by FA (100 runs for each of the 100 problems). The upper and the lower boundaries of the zone are shown as dark blue curves.

Fig. 2.12.b) than in average FA is better than ADC, as well. If the budget is higher than 40 000 trials than ADC behaves better since its characteristic is higher than the upper boundary of this FA zone. Notice also that Fig. 2.12.b shows that after 10^5 trials only the method ADC was able to solve all 100 test problems of the class. For the same two test classes, Fig. 2.12 presents operational zones for metaheuristics GA and ABC and for the three deterministic methods.

In order to see the advantages of the proposed methodologies for comparing methods, Table 2.7 constructed in a traditional way is shown. Due to the huge amount of data, only average results can be considered and included in Table 2.7. Notice that for deterministic methods and metaheuristics, due to the stochastic nature of the latter ones, different averages should be used: for metaheuristics the results on 10 000 runs for each class are used, whereas for the deterministic algorithms results on 100 runs (one run for each of 100 functions). This creates difficulties in comparing (see, e.g., [18]). For instance, which method is better on the 5-dimensional simple class: DIRECT or FA? On the one hand, DIRECT did not solve only one problem in

100 runs, demonstrating so success rate of 99%, whereas FA did not solve 16 problems in 10 000 runs, demonstrating, i.e., 99.84% of success. On the other hand, the average number of trials for DIRECT was only 16 057.5, while for FA 47 203.1 trials required in average. Moreover, Table 2.7 cannot give results for 50% or 75% of solved test problems, that can be also important. To see the detailed results, larger tables with hundreds of rows and columns should be used, complicating so the visual analysis of the results.

In contrast, operational zones very well present visually performance of tested methods giving the entire panorama of their behavior for different budgets. For instance, it can be seen from Fig. 2.12 that metaheuristics perform very well on small budgets showing better results w.r.t. deterministic algorithms whereas the best algorithm for the higher budget on the used test classes is the algorithm ADC since it was able to solve all 100 test problems faster than other methods on both the classes. Even though this result can be also obtained from Table 2.7, the operational zones allow us to observe the performance of methods at all the stages of the search for each class. The average, the best, and the worst cases for each metaheuristic can be easily obtained from the graphs for any chosen number of trials. Moreover, the number of trials required to solve 50% (or 75%, 90%, etc.) of problems can be easily obtained and performance of methods is visualized clearly.

Let us see now another way for a statistical comparison of the two groups of algorithms using the same data. Let X_A^C be a random variable describing the consumed percentage of the computational budget N_{max} performed by an algorithm A for solving a problem from the test class C. Let us consider the sample x_A^C of 100 realizations of X_A^C for $A \in \{\text{ADC, DIRECT, DIRECT-L}\}$ and $100 \times 100 = 10000$ realizations of X_A^C for $A \in \{\text{FA, GA, ABC, PSO, DE}\}$, i.e., if, for instance, the algorithm ADC solved the 2-dimensional hard test problem number 5 after 574 trials, then $x_{ADC}^{2-hard} = \frac{574}{10^6} \times 100\% = 0.0574\%$. Then, after the construction of the cumulative distribution functions $F_{X_A^C}(x)$, one can obtain the sampled distribution quantiles of X_A^C. For instance, in Tables 2.8–2.9, the sampled 25%, 50%, 75%, and 90% quantiles are presented for simple and hard classes, respectively. These results can be interpreted as follows. The 90%-quantile for the FA on the 5-dimensional simple class is 14.11%, whereas the same quantile for the ADC is 1.02%. This means that with the probability 90% FA will consume no more than 14.11% of the computational budget (i.e., no more than 141 100 trials), while ADC will consume no more than 1.02% of the computational budget (i.e., no more than 10 200 trials) to solve successfully the test problem. As it can be seen from Table 2.8, GA for the same test class and the same confidence level will consume 100% of the computational budget. This means that it cannot

be claimed with the probability 90% that GA will solve the problem in the selected computational budget. However, it should be noted that with the probability 75% GA will resolve the test problem of the same class with no more than 9.24% of the computational budget (i.e., with no more than 92 400 trials).

On can see that the presented quantiles correspond to the results presented in Figures 2.11–2.19. In particular, the results presented in Tables 2.8–2.9 correspond to the average operational zones for each metaheuristic algorithm presented in Figures 2.12–2.19.

2.2.3 Aggregated operational zones and restarts of metaheuristics

One can see also that in many runs metaheuristics got trapped into local minima and were not able to exit from their attraction regions producing so operational zones with long horizontal parts (see, e.g., Fig. 2.12.d where metaheuristic GA works significantly better than DIRECT and DIRECT-L if the budget is less than 40 000 trials and then almost does not improve the number of solved problems remaining, however, always better than the two deterministic methods). This means that increasing the number of trials does not improve results in this case and it is necessary to restart metaheuristics. *Aggregated operational zones* proposed here show what happens in this case. They are constructed as follows.

First, an algorithm is launched K times ($K = 100$ was used again here in order to have the same computational resources available for constructing operational zones) with an allowed number of trials $n_{max} < N_{max}$ (in our experiments $n_{max} = 50000$, $N_{max} = 10^6$ for each metaheuristic). Then, for non-solved problems the algorithm is launched again with the same number n_{max} of allowed trials. Thus, if the algorithm did not solve a problem p in the first $n, 1 \leq n < T, T = \lfloor N_{max}/n_{max} \rfloor$, runs but has solved it in the $(n+1)$-th run in $t, 1 \leq t \leq n_{max}$, trials then the number of trials to solve the problem p is equal to $n * n_{max} + t$. Otherwise, if the algorithm did not solve the problem p in T runs, then the number of executed trials for the problem p is set equal to the maximal allowed number N_{max} (in order to remind that more than N_{max} trials are required to solve this problem the mark "$> 10^6$" is used in Table 2.7). In this way, T runs are executed to complete the aggregated characteristic. Finally, $k = K/T$ aggregated operational characteristics are used to build the aggregated operational zone in the same way as operational characteristics are used to construct an operational zone. The lower and upper boundaries are defined analogously.

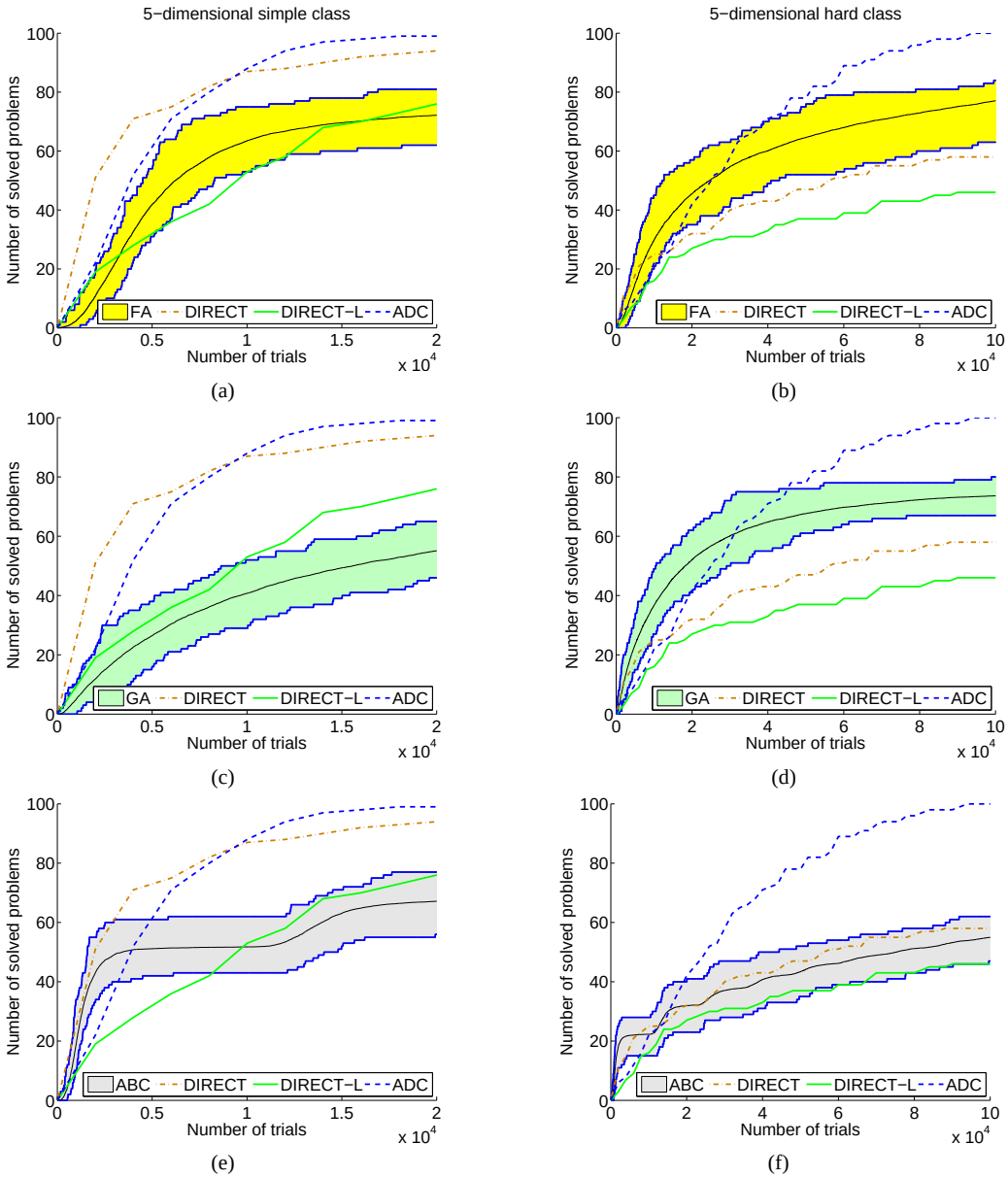


Figure 2.12: Operational characteristics built on two 5-dimensional classes of 100 GKLS test functions for deterministic methods DIRECT, DIRECT-L, and ADC and operational zones for stochastic metaheuristics Firefly Algorithm (FA), Genetic Algorithm (GA) and Artificial Bee Colony (ABC). (a) Operational characteristics for deterministic methods and the operational zone for FA for the simple 5-dimensional class. (b) The same as (a) for the hard class. (c) Operational characteristics for deterministic methods and the operational zone for GA for the simple 5-dimensional class. (d) The same as (c) for the hard class. (e) Operational characteristics for deterministic methods and the operational zone for ABC for the simple 5-dimensional class. (f) The same as (e) for the hard class.

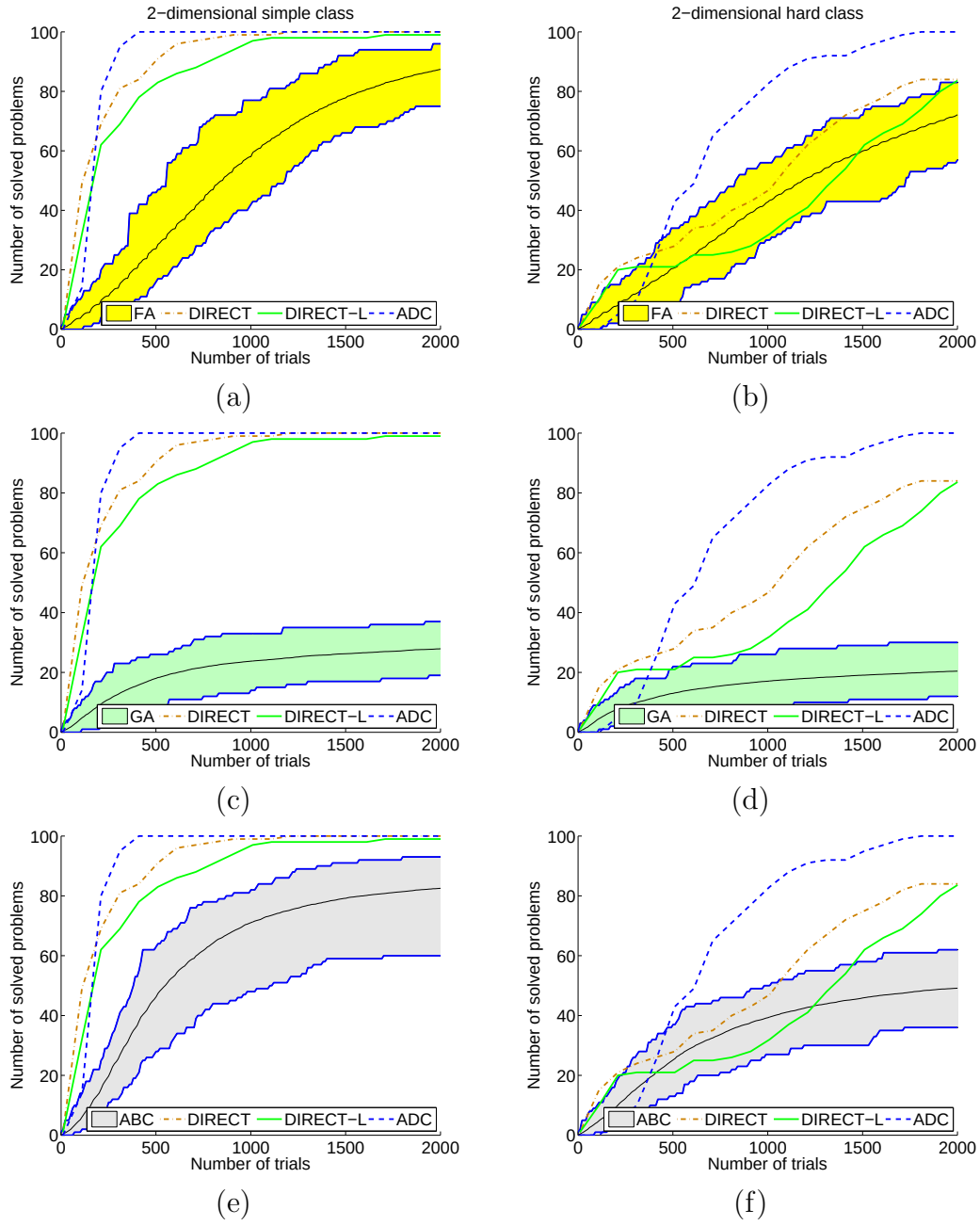


Figure 2.13: Operational characteristics and operational zones for the 2-dimensional GKLS classes for metaheuristics FA, GA, and ABC. (a) Operational characteristics for deterministic methods and operational zones for FA are presented for the simple class. (b) The same as (a) for the hard class. (c) Operational characteristics for deterministic methods and operational zones for GA are presented for the simple class. (d) The same as (c) for the hard class. (e) Operational characteristics for deterministic methods and operational zones for ABC are presented for the simple class. (f) The same as (e) for the hard class.

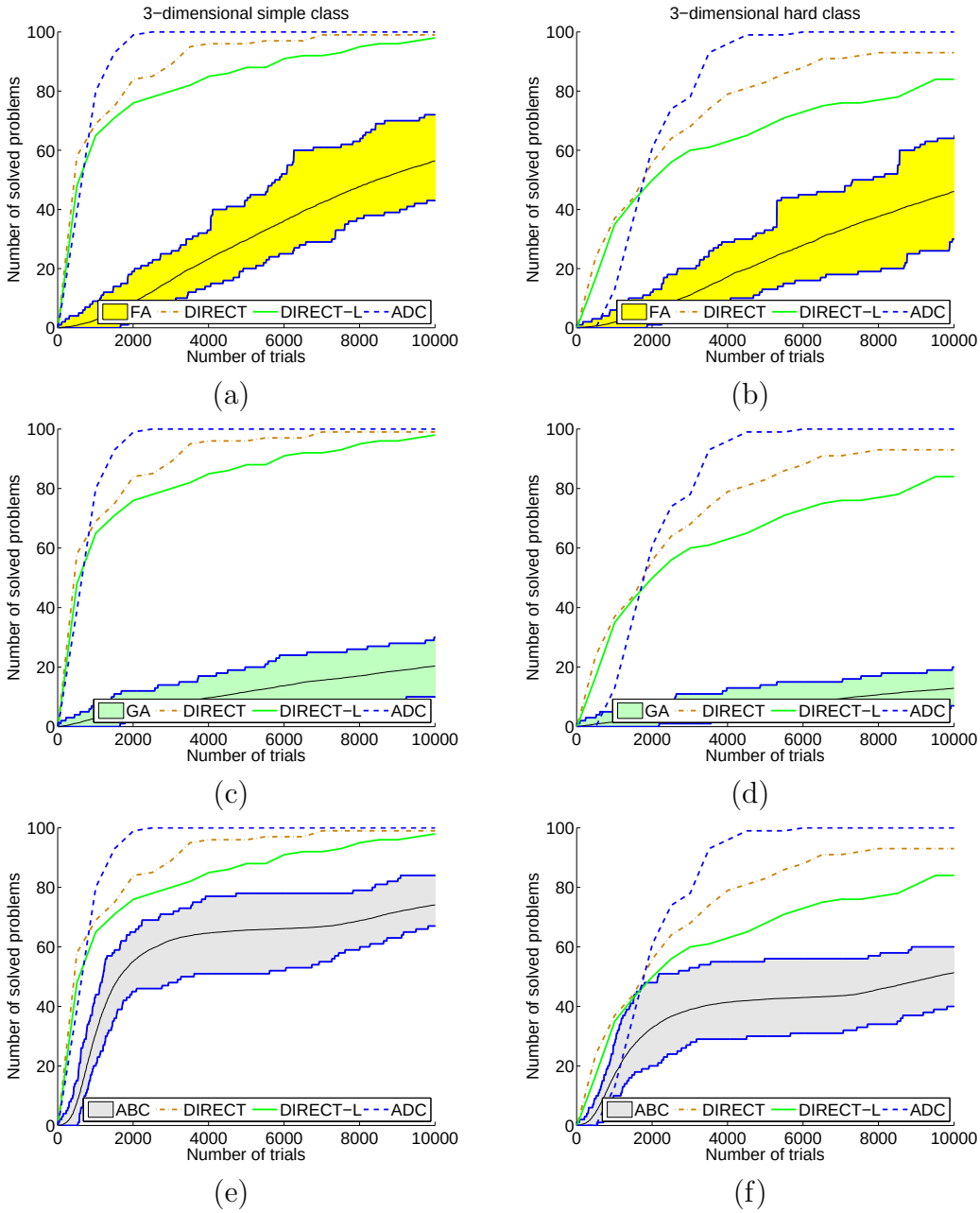


Figure 2.14: Operational characteristics and operational zones for the 3-dimensional GKLS classes for metaheuristics FA, GA, and ABC. (a) Operational characteristics for deterministic methods and operational zones for FA are presented for the simple class. (b) The same as (a) for the hard class. (c) Operational characteristics for deterministic methods and operational zones for GA are presented for the simple class. (d) The same as (c) for the hard class. (e) Operational characteristics for deterministic methods and operational zones for ABC are presented for the simple class. (f) The same as (e) for the hard class.

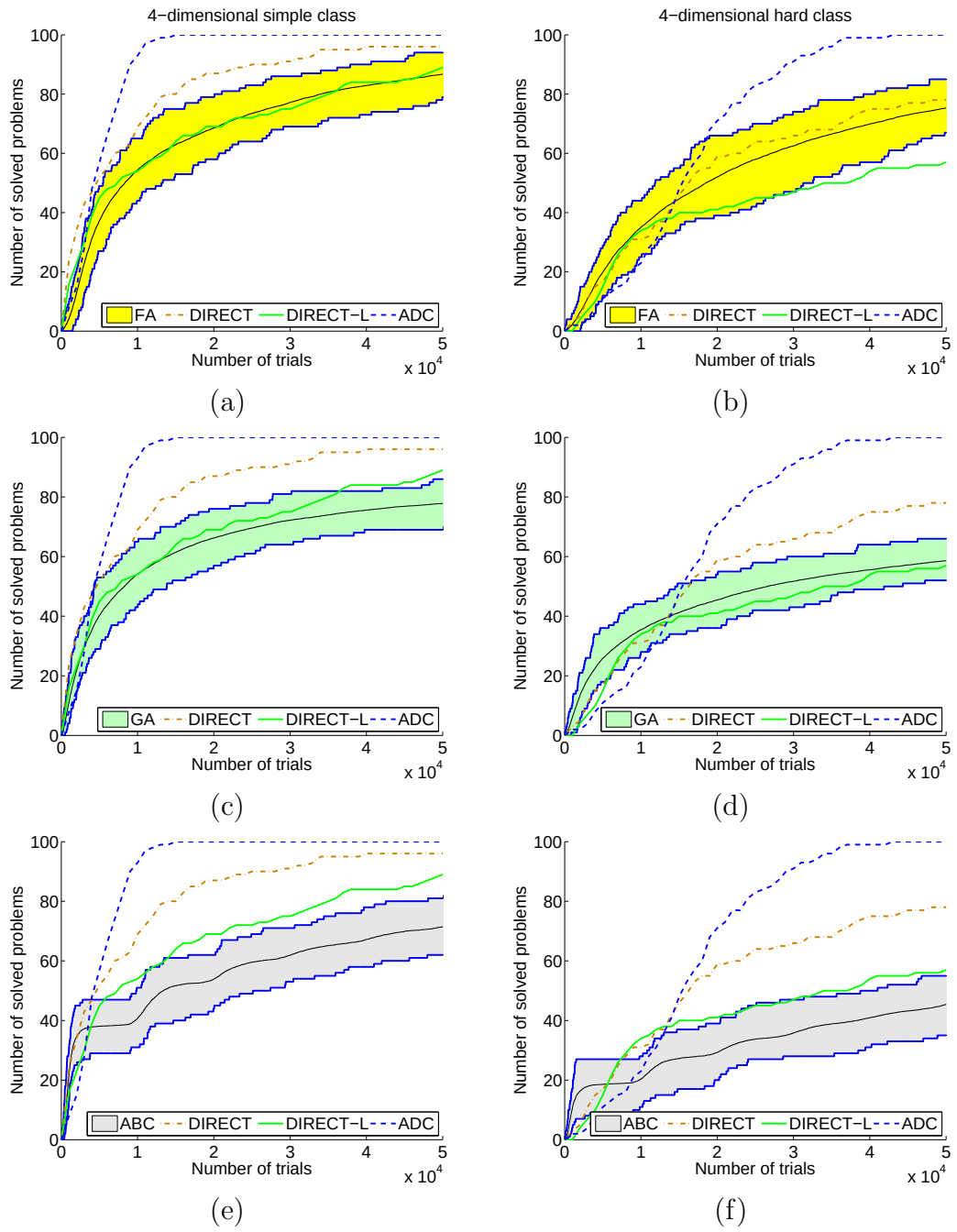


Figure 2.15: Operational characteristics and operational zones for the 4-dimensional GKLS classes for metaheuristics FA, GA, and ABC. (a) Operational characteristics for deterministic methods and operational zones for FA are presented for the simple class. (b) The same as (a) for the hard class. (c) Operational characteristics for deterministic methods and operational zones for GA are presented for the simple class. (d) The same as (c) for the hard class. (e) Operational characteristics for deterministic methods and operational zones for ABC are presented for the simple class. (f) The same as (e) for the hard class.

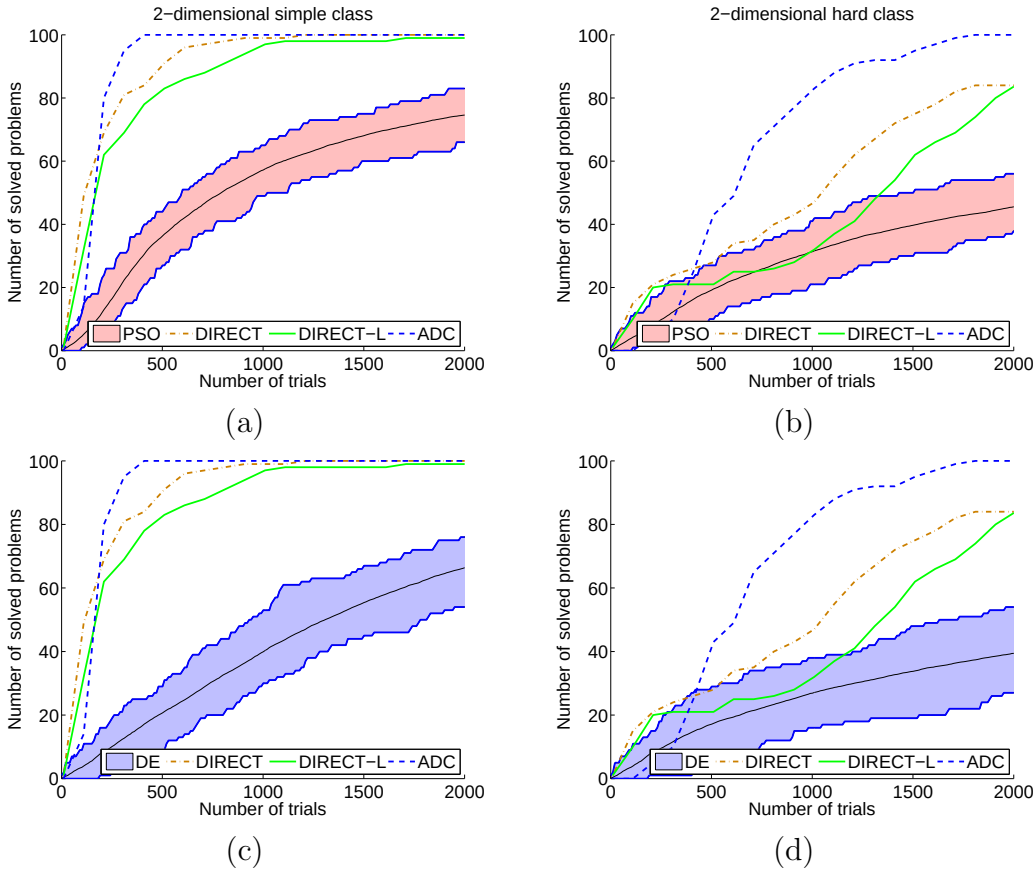


Figure 2.16: Operational characteristics and operational zones for the 2-dimensional GKLS classes for metaheuristics PSO and DE. (a) Operational characteristics for deterministic methods and operational zones for PSO are presented for the simple class. (b) The same as (a) for the hard class. (c) Operational characteristics for deterministic methods and operational zones for DE are presented for the simple class. (d) The same as (c) for the hard class.

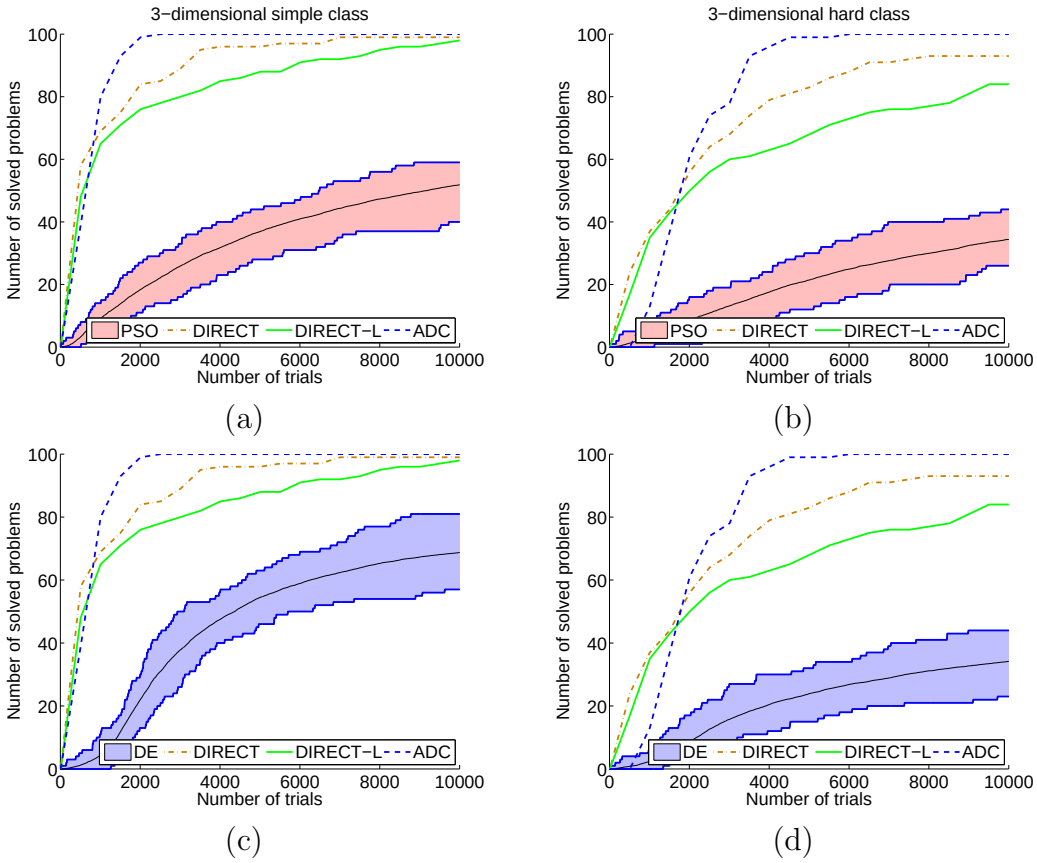


Figure 2.17: Operational characteristics and operational zones for the 3-dimensional GKLS classes for metaheuristics PSO and DE. (a) Operational characteristics for deterministic methods and operational zones for PSO are presented for the simple class. (b) The same as (a) for the hard class. (c) Operational characteristics for deterministic methods and operational zones for DE are presented for the simple class. (d) The same as (c) for the hard class.

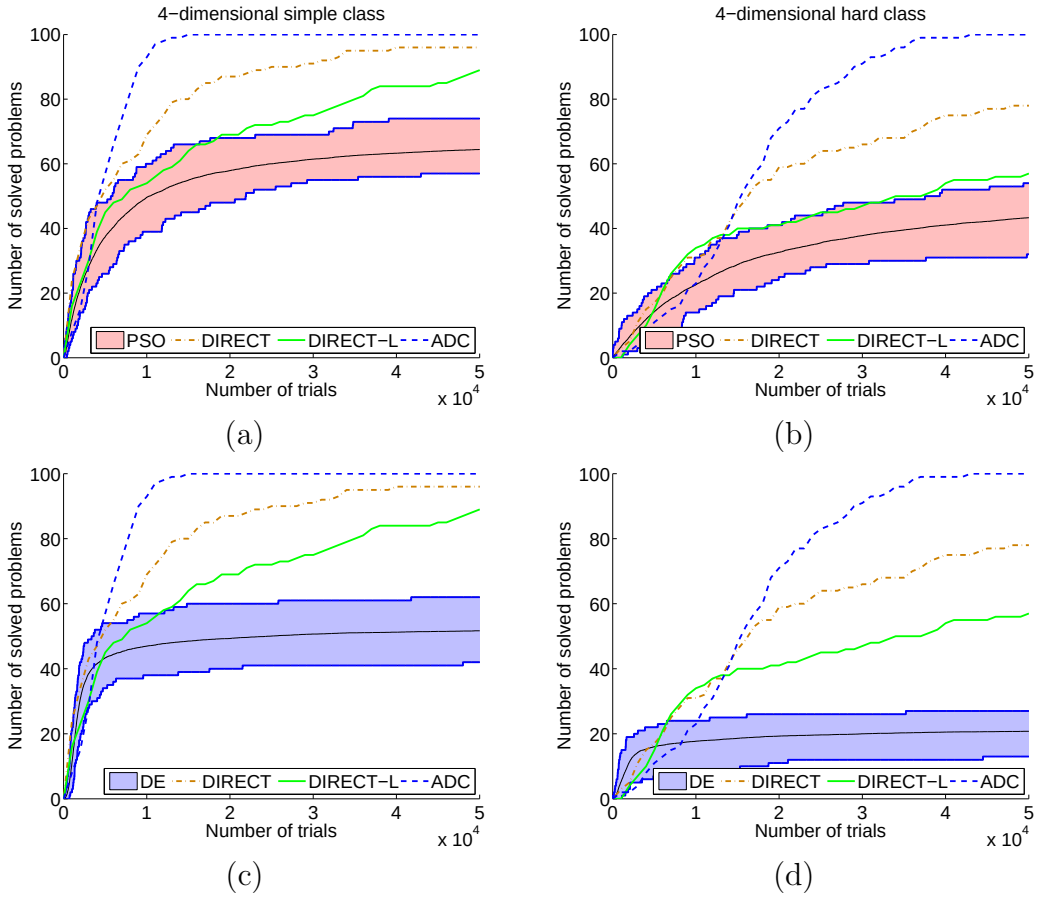


Figure 2.18: Operational characteristics and operational zones for the 4-dimensional GKLS classes for metaheuristics PSO and DE. (a) Operational characteristics for deterministic methods and operational zones for PSO are presented for the simple class. (b) The same as (a) for the hard class. (c) Operational characteristics for deterministic methods and operational zones for DE are presented for the simple class. (d) The same as (c) for the hard class.

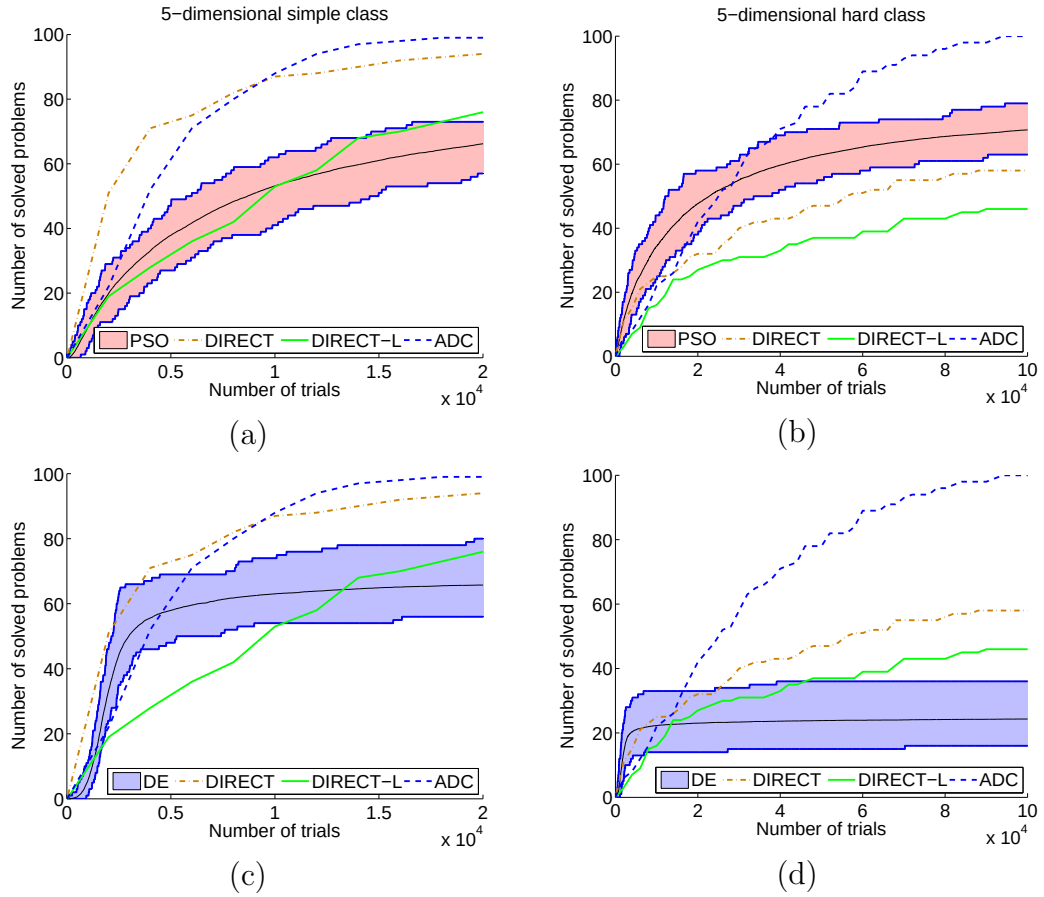


Figure 2.19: Operational characteristics and operational zones for the 5-dimensional GKLS classes for metaheuristics PSO and DE. (a) Operational characteristics for deterministic methods and operational zones for PSO are presented for the simple class. (b) The same as (a) for the hard class. (c) Operational characteristics for deterministic methods and operational zones for DE are presented for the simple class. (d) The same as (c) for the hard class.

N	Class	**Metaheuristic algorithms** (10000 runs for each algorithm and class)					**Deterministic algorithms** (100 runs for each algorithm and class)		
		Differential Evolution	Particle Swarm Optimization	Genetic Algorithm	Artificial Bee Colony	Firefly Algorithm	DIRECT	DIRECT-L	Diagonal Algorithm
2	simple	>52910.38(511)	>110102.74(1046)	>327452.5(2735)	2120.8	1190.3	198.9	292.8	176.3
2	hard	>357467.49(3556)	>247232.35(2282)	>370907.3(3149)	10366.2	>4299.6(3)	1063.8	1267.1	675.7
3	simple	>165125.02(1515)	>170320.10(1489)	>242231.7(1599)	10245.0	15269.2	1117.7	1785.7	735.8
3	hard	>476251.20(4603)	>285499.04(2501)	>412037.8(2874)	26254.2	>21986.3(1)	>42322.7(4)	4858.9	2006.8
4	simple	>462401.52(4546)	>303436.36(2785)	>150597.5(1290)	>57669.5(19)	23166.7	>47282.9(4)	18983.6	5014.1
4	hard	>773481.03(7676)	>456996.08(4157)	>247860.8(1900)	>150706.5(255)	40380.7	>95708.3(7)	68754.0	16473.0
5	simple	>294839.01(2815)	>181805.17(1561)	>237392.9(2208)	>38068.0(14)	>47203.1(16)	>16057.5(1)	16758.4	5129.9
5	hard	>751930.00(7473)	>250462.63(2109)	>249965.6(2311)	>230192.9(879)	>79555.2(38)	>217215.6(16)	>269064.4(4)	30471.8

†The record ">m(i)" means that the algorithm did not solve a global optimization problem i times in 100 runs $\times$100 problems (i.e., in 10000 runs for metaheuristics and in 100 runs for deterministic algorithms). In this case, the maximal number of trials set to 10^6 was used to calculate the average number of trials m.

Table 2.7: Results of the experiments. For each test class the average number of trials required to solve all 100 problems is presented for each deterministic algorithm. For each metaheuristic method, the average number of trials required to solve each problem on 100 runs has been calculated, and the average of these 100 values is presented.†

N	Q	**Metaheuristic algorithms** (10,000 runs for each algorithm and class)					**Deterministic algorithms** (100 runs for each algorithm and class)		
		Differential Evolution	Particle Swarm Optimization	Genetic Algorithm	Artificial Bee Colony	Firefly Algorithm	DIRECT	DIRECT-L	ADC
2	25	0.07%	0.04%	0.13%	0.03%	0.05%	0.01%	0.01%	0.01%
	50	0.14%	0.08%	3.28%	0.05%	0.08%	0.01%	0.02%	0.02%
	75	0.26%	0.21%	99.53%	0.12%	0.14%	0.02%	0.04%	0.02%
	90	0.60%	95.74%	100.00%	0.69%	0.22%	0.04%	0.07%	0.02%
3	25	0.22%	0.29%	1.35%	0.09%	0.43%	0.02%	0.03%	0.05%
	50	0.44%	0.92%	4.83%	0.16%	0.84%	0.04%	0.06%	0.06%
	75	1.99%	3.60%	26.64%	1.06%	1.66%	0.15%	0.19%	0.10%
	90	35.70%	97.61%	99.83%	2.94%	2.91%	0.30%	0.57%	0.12%
4	25	0.17%	0.25%	0.24%	0.12%	0.32%	0.13%	0.23%	0.28%
	50	2.49%	1.04%	0.83%	1.34%	0.83%	0.50%	0.73%	0.41%
	75	89.48%	97.25%	3.78%	6.03%	2.69%	1.12%	3.07%	0.73%
	90	100.00%	100.00%	96.73%	15.89%	6.18%	2.21%	5.57%	0.89%
5	25	0.17%	0.27%	0.46%	0.12%	0.33%	0.06%	0.32%	0.21%
	50	0.29%	0.87%	1.56%	0.34%	0.62%	0.16%	0.93%	0.39%
	75	97.05%	4.04%	9.24%	3.73%	2.86%	0.64%	1.90%	0.64%
	90	100.00%	99.78%	100.00%	10.89%	14.11%	1.54%	3.54%	1.02%

Table 2.8: Results of the experiments. For each algorithm, quantiles Q_{25}, Q_{50}, Q_{75}, and Q_{90} for the number of trials for simple test classes are presented.

N	Q	**Metaheuristic algorithms** (10,000 runs for each algorithm and class)					**Deterministic algorithms** (100 runs for each algorithm and class)		
		Differential Evolution	Particle Swarm Optimization	Genetic Algorithm	Artificial Bee Colony	Firefly Algorithm	DIRECT	DIRECT-L	ADC
2	25	0.09%	0.08%	0.43%	0.05%	0.07%	0.03%	0.06%	0.04%
	50	0.40%	0.26%	6.12%	0.23%	0.12%	0.11%	0.13%	0.06%
	75	2.96%	25.03%	99.05%	1.32%	0.22%	0.14%	0.18%	0.09%
	90	100.00%	100.00%	100.00%	2.87%	0.47%	0.22%	0.22%	0.12%
3	25	0.54%	0.61%	2.80%	0.13%	0.54%	0.04%	0.08%	0.12%
	50	14.88%	2.28%	18.64%	0.95%	1.11%	0.17%	0.20%	0.17%
	75	32.84%	99.91%	99.63%	3.20%	2.14%	0.37%	0.62%	0.27%
	90	100.00%	100.00%	100.00%	7.20%	4.02%	0.64%	1.25%	0.35%
4	25	93.83%	1.18%	0.48%	1.23%	0.63%	0.69%	0.67%	1.04%
	50	93.83%	12.42%	2.66%	6.16%	1.85%	1.61%	3.32%	1.51%
	75	100.00%	99.10%	26.78%	18.51%	4.93%	4.31%	11.31%	2.16%
	90	100.00%	100.00%	99.57%	41.96%	10.52%	13.40%	18.89%	2.90%
5	25	45.86%	0.59%	0.55%	1.27%	0.82%	0.89%	1.90%	1.31%
	50	95.77%	2.28%	1.82%	7.51%	2.46%	5.51%	18.29%	2.46%
	75	100.00%	21.34%	13.81%	30.74%	8.97%	15.09%	47.71%	4.44%
	90	100.00%	100.00%	100.00%	88.36%	21.57%	19.01%	69.96%	6.43%

Table 2.9: Results of the experiments. For each algorithm, quantiles Q_{25}, Q_{50}, Q_{75}, and Q_{90} for the number of trials for hard test classes are presented.

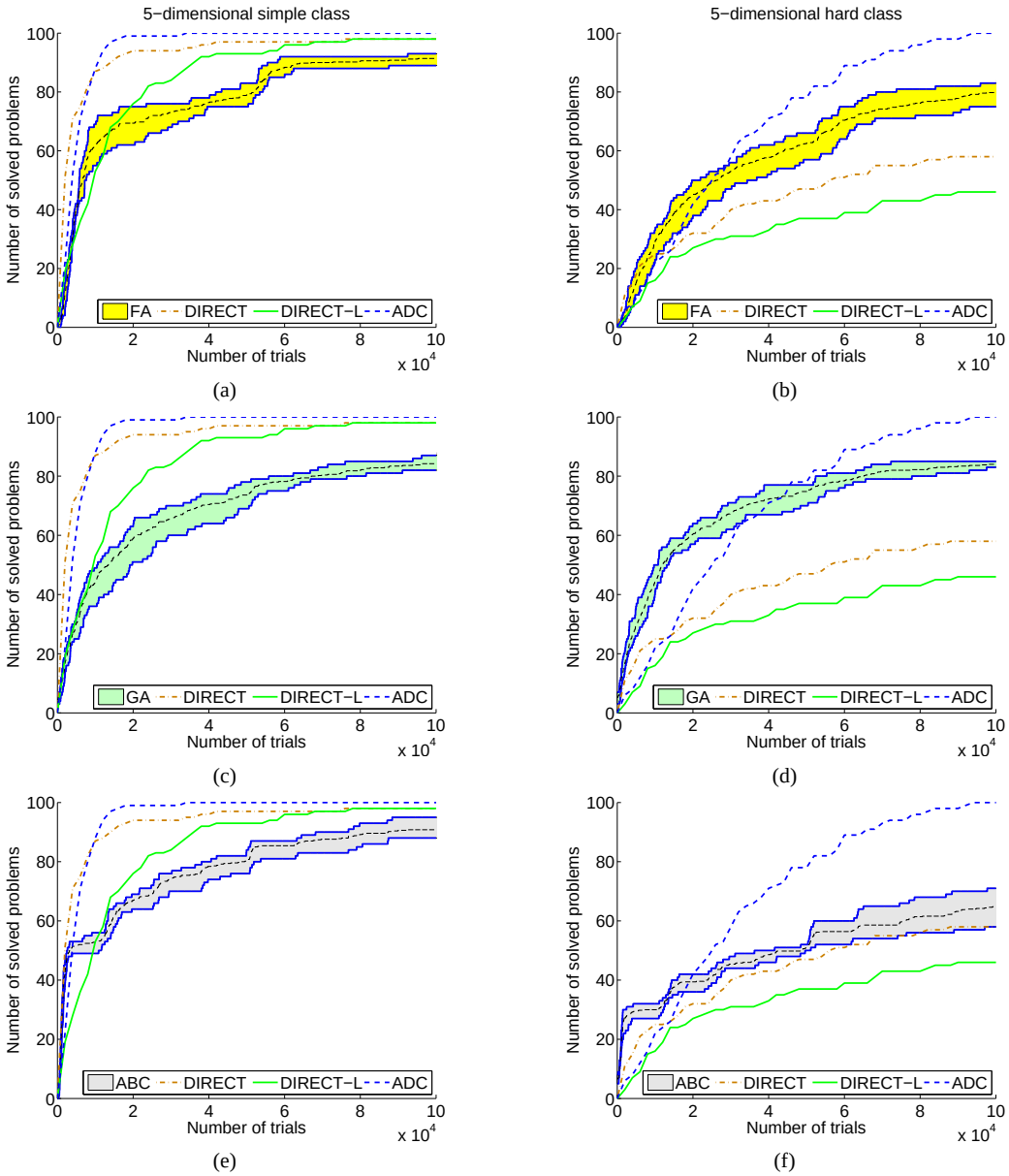


Figure 2.20: Aggregated operational zones for stochastic metaheuristics Firefly Algorithm (FA), Genetic Algorithm (GA), and Artificial Bee Colony (ABC) and operational characteristics for deterministic methods DIRECT, DIRECT-L, and ADC built on two 5-dimensional classes of 100 GKLS test functions. (a) Operational characteristics for deterministic methods and the aggregated operational zone for FA for the simple 5-dimensional class. (b) The same as (a) for the hard class. (c) Operational characteristics for deterministic methods and the aggregated operational zone for GA for the simple 5-dimensional class. (d) The same as (c) for the hard class. (e) Operational characteristics for deterministic methods and the aggregated operational zone for ABC for the simple 5-dimensional class. (f) The same as (e) for the hard class.

Fig. 2.20 shows results of experiments for the three deterministic methods and metaheuristics FA, GA, and ABC. It should be stressed that the aggregated operational zones allow one to emphasize better the potential of nature-inspired metaheuristics. In fact, the advantage of the aggregated zones with respect to the operational zones can be illustrated, for example, by situations shown in Fig. 2.12.f and Fig. 2.20.f. It can be seen from Fig. 2.12.f that operational characteristics of deterministic methods DIRECT and DIRECT-L are located inside the zone of the metaheuristic ABC and, therefore, it is not possible to determine which of the three methods behaves better. In contrast, the aggregated zone of ABC is higher than the characteristics of both deterministic methods, i.e., it can be concluded that ABC outperforms them.

In conclusion, the proposed operational zones and aggregated operational zones allow one to compare effectively deterministic and stochastic global optimization algorithms having different nature and give a handy visual representation of this comparison for different computational budgets. Nature-inspired metaheuristics and deterministic Lipschitz algorithms have been compared on 800 of tests giving so a new understanding for both classes of methods and opening a dialog between the two communities. It can be seen that both classes of algorithms are competitive and surpass one another in dependence on the available budget of function evaluations.

2.3 Emmental-type GKLS-based generator of test classes for global optimization with nonlinear constraints

Continuous constrained global optimization problems are considered here. A problem of this kind can be formally stated as follows (see, e. g., references in [54, 192, 207, 215]):

$$f^* = f(x^*) = \min_{x \in D} f(x), \quad D \subset \mathbb{R}^N, \tag{2.6}$$

where D is a bounded N-dimensional region defined as

$$D = \{x \in \Omega : \, g_j(x) \leq 0, \, 1 \leq j \leq p\}, \tag{2.7}$$

$$\Omega = [a, b] = \{x \in \mathbb{R}^N : a_i \leq x_i \leq b_i, \, 1 \leq i \leq N\}, \quad a, b \in \mathbb{R}^N. \tag{2.8}$$

The objective function $f(x)$ from (2.6) and constraints $g_j(x)$, $1 \leq j \leq p$, from (2.7) are continuous, nonlinear, and can be multiextremal.

A new generator of classes of multidimensional test problems for benchmarking continuous constrained global optimization methods is described in this Section. It is based on the GKLS-generator as given in Algorithm 829 [65] and extends the previous generation procedure from the box-constrained case to the case of nonlinear constraints (see [197] and [199] for details). The user has the possibility to fix the difficulty of tests in an intuitive way by choosing several types of constraints. A detailed information (including the global solution) for each of 100 generated problems in each class is provided to the user.

2.3.1 Box-constrained GKLS generator of test problems

There exists a huge number of methods for solving problem (2.6)–(2.8) (see, e. g., references in [54, 207, 215]). As a consequence, the necessity of their fair numerical comparison on a well-structured benchmarking set arises, especially when methods having a completely different nature are to be compared (see, e. g., [107] for the related discussion on a numerical comparison of metaheuristic and deterministic global optimization algorithms). In global optimization tests taken from real-life applications, the lack of such information as the number of local minimizers, their locations and attraction regions, and local and global minimum values creates additional difficulties in verifying validity of the algorithms (see benchmark classes and related discussions on testing global optimization algorithms, e. g., in [18, 55, 140, 158, 160, 162, 188, 215, 229]).

One of the important aspects to be considered in this context is an intuitive visualization of numerical results. For example, in [198] (see also Section 2.2), new tools called *operational zones* and *aggregated operational zones* for comparing global optimization algorithms of a different nature (e. g., stochastic and deterministic methods) by means of a particular graphical representation of the obtained numerical results have been proposed. To use this tool, classes of test problems are required. On the one hand, there exist many generators of test problems for continuous box-constrained local and global optimization (see, e. g., [60] for the landscape generators; [146] for the large-scale nonconvex quadratic problems generator; [17], [21], and [55] for some interesting test classes; and [4], [78], [153], and [162] for other generators of global optimization test problems). On the other hand, in the framework of a general continuous *constrained* global optimization only collections of benchmark test problems are usually used (see, e. g., [53, 148, 193]) due to the absence of test classes and generators for this type of problems.

The research meets this lack and introduces a new generator of classes of 100 random global optimization test problems with similar characteristics,

nonlinear constraints, known minimizers, and a parameterizable difficulty. In order to obtain a powerful instrument for testing constrained global optimization methods, the following requirements have been particularly taken into consideration:

1. The global solution of the constrained and box-constrained problems should be known but differ from each other.

2. The global solution of the constrained problem should be placed on the boundary of the feasible region, since in practice this is a frequently encountered situation.

3. The difficulty of the admissible region should be controllable in several ways.

4. The gradient value of the objective function at the global minimizer of the constrained problem should be different from zero[1].

5. The number of active constraints at the global solution to the constrained problem should be controllable[2].

In order to guarantee the properties given above, the *Emmental-type generator of classes of constrained test problems* is built by extending the GKLS-generator (see [65]) of classes of test problems with non-differentiable, differentiable, and twice continuously differentiable objective functions for box-constrained global optimization. Since 2003, when the GKLS-generator has been proposed, it has gain a high popularity within the global optimization community for testing numerical methods (in fact, it is now used in more than 40 countries of the world) due to its simplicity and flexibility. Therefore, its extension to the case of constrained global optimization problems seems to be an opportune and useful contribution in such an important research area as benchmarking numerical global optimization software.

Constrained Emmental-type tests are built using one of the following three classes produced by the GKLS-generator [65]: non-differentiable (ND-type), differentiable (D-type), and twice continuously differentiable (D2-type). We recall that box-constrained test problems in the GKLS classes are generated by defining a convex quadratic function distorted by polynomials of orders 2, 3, or 5 for ND, D, or D2 classes, respectively, in order to

[1]Suggested by Julius Žilinskas.
[2]Suggested by Renato De Leone.

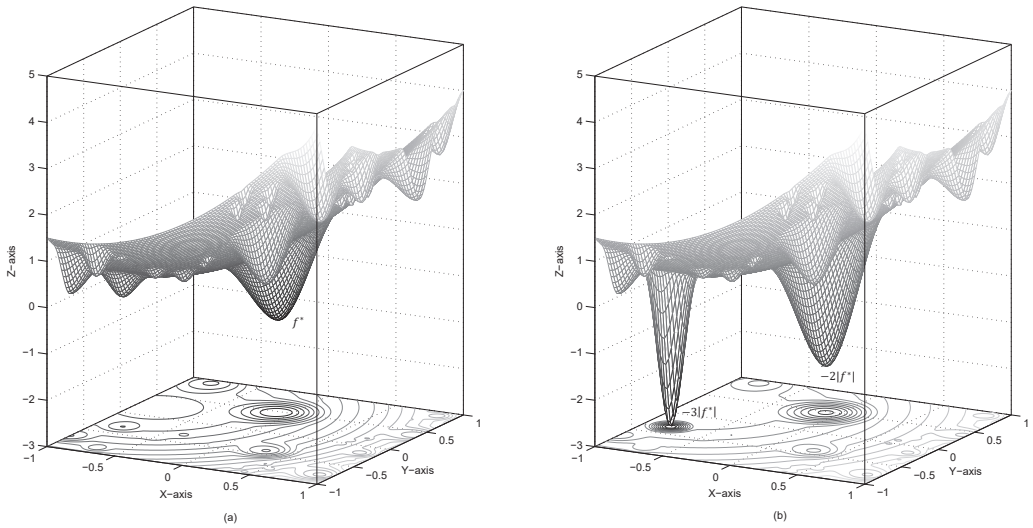


Figure 2.21: Examples of a box-constrained D-type GKLS test problem (a) and the corresponding box-constrained Emmental-type test problem (b) over $\Omega = [-1, 1]^2$: the functions do not coincide even in the absence of nonlinear constraints.

introduce local minima. In particular, the objective function of a GKLS class is constructed by modifying a paraboloid Z

$$Z: \;\; z(x) = ||x - T||^2 + t, \quad x \in \Omega, \tag{2.9}$$

with the minimum t at a point $T \in int(\Omega)$, where $int(\Omega)$ denotes the interior of Ω from (2.8) and $||\cdot||$ denotes the Euclidean norm, in such a way that the resulting box-constrained problem has m, $m \geq 2$, local minimizers: the point T from (2.9) and points $M_i \in int(\Omega)$, $M_i \neq T$, $M_i \neq M_j$, $2 \leq i, j \leq m$, $i \neq j$. The paraboloid Z from (2.9) is modified by the polynomials within balls $S_i \subset \Omega$ (not necessarily entirely contained in Ω) around each point M_i, $2 \leq i \leq m$ (with $M_1 := T$ being the vertex of the paraboloid and M_2 being the global minimizer of the box-constrained test problem), where

$$S_i = \{x \in \mathbb{R}^N : ||x - M_i|| \leq \rho_i, \;\; \rho_i > 0\}. \tag{2.10}$$

Each GKLS test class includes 100 functions and is defined by five parameters to be chosen by the user (see [65] for details):

(1) the problem dimension $N \geq 2$;

(2) the number of local minimizers $m \geq 2$ (including the paraboloid vertex $M_1 = T$ from (2.9));

(3) the global minimum value $f^* < t$ ($t = 0$ by default in (2.9));

(4) the radius ρ^* of the attraction region of the global minimizer M_2;

(5) the distance r^* from the global minimizer M_2 to the paraboloid vertex M_1.

Many other necessary parameters are set randomly by the GKLS-generator for each test function of the selected class. It is important to notice that the generator produces the same test classes for a fixed set of the user-defined parameters thus allowing the repeatability of numerical experiments.

As an illustration, Fig. 2.21.a shows an example of a GKLS test function defined over the box $\Omega = [-1, 1]^2$. This function has number 3 in the D-type test class with the following parameters: (1) $N = 2$; (2) $m = 30$; (3) $f^* = -1$; (4) $\rho^* = \frac{2}{5}$; and (5) $r^* = \frac{4}{5}$. The randomly generated global minimizer of this function is $M_2 \simeq (-0.086, 0.478)$ and the paraboloid vertex is $M_1 \simeq (-0.842, 0.216)$.

Let us show now, how the global minimum points and values of the box-constrained GKLS problems are modified in order to take into account the first of the requirements for constrained tests mentioned previously. By doing this, the *box-constrained* Emmental-type test problem corresponding to a box-constrained GKLS test problem but different from the latter one will be obtained. Then, nonlinear constraints will be introduced in the resulting box-constrained Emmental-type test problems to construct the desired *constrained* Emmental-type tests. The global minimum value of the constrained Emmental-type test problem remains equal to that of the corresponding box-constrained GKLS problem (as one of the user-defined parameters) but its location is different and is shifted to the boundary of the admissible region D from (2.7) defined by a set of active nonlinear constraints.

2.3.2 Generator with parameterizable difficulty and known global solution

In order to ensure that the global solution to the newly constructed box-constrained and constrained Emmental-type problems and the original box-constrained GKLS problems differ from each other, the following modification to the GKLS tests is performed (see [199] for details). For a particular box-constrained GKLS test problem with $m \geq 3$ (cf. [65] where $m \geq 2$ is allowed) local minimizers, the local (not global M_2) minimizer $M_{i_{\min}}$ having

the attraction region closest to the paraboloid vertex $M_1 = T$ from (2.9) is determined by index $i_{\min}$ as follows

$$i_{\min} = \arg \min_{3 \le i \le m} \{||M_i - M_1|| - \rho_i\}, \tag{2.11}$$

where ρ_i is the radius of the ball S_i from (2.10) (index i starts from 3 in (2.11), since $i = 1$ and $i = 2$ are reserved for the paraboloid vertex and the global minimizer, respectively). Polynomial $P_{i_{\min}}$ for the ND-type GKLS function (polynomials $C_{i_{\min}}$ or $Q_{i_{\min}}$ for D-type or D2-type GKLS functions, respectively), defined in [65] by formula (15) (formulae (8) or (13)) is then modified as follows. Its smallest value over $S_{i_{\min}}$ is set equal to $-3 \cdot |f^*|$, where f^* is the user-defined global minimum value of the considered box-constrained GKLS problem (recall that f^* should be smaller than the function value t at the paraboloid vertex from (2.9); t is set equal to 0 by default in the GKLS-generator). After this modification, the global solution to the box-constrained problem moves to the point $M_{i_{\min}}$ (with $i_{\min}$ from (2.11)) with a new global minimum value equal to $-3 \cdot |f^*|$. In this way, the global minimum point and value of the box-constrained GKLS and Emmental-type problems are different (see Fig. 2.21.a and Fig. 2.21.b). The updated information about the global solution of the box-constrained Emmental-type problem under consideration is provided to the user.

In order to have a global minimizer x^* of the constrained Emmental-type problem different from the corresponding box-constrained GKLS and Emmental-type problems and with a non-zero gradient value (the fourth requirement), x^* is placed between the points M_1 (the paraboloid vertex form (2.9)) and M_2 (the global minimizer of the box-constrained GKLS problem) with an unknown distance d^* from the point M_2, $d^* = ||x^* - M_2||$,

$$x^* = \frac{d^*}{||M_1 - M_2||} M_1 + (1 - \frac{d^*}{||M_1 - M_2||}) M_2. \tag{2.12}$$

To ensure that the global minimum value of the constrained Emmental-type problem (2.6) at the point x^* is equal to f^* (i. e., equal to the user-defined global minimum value of the box-constrained GKLS test problem), the distorting polynomial $P_2(x)$, $C_2(x)$, or $Q_2(x)$ of order 2, 3, or 5 in the box-constrained Emmental-type problem of ND, D, or D2 types, respectively (see formulae (15), (8), or (13) in [65]), is redefined over the attraction region S_2 of the point M_2 by setting its smallest value in S_2 equal to $-2 \cdot |f^*|$. Taking into consideration that the points M_1, x^*, and M_2 are aligned along the ray from M_1 and denoting $d = ||x - M_2||$ (where d expresses the distance from M_2 to a point $x \in S_2$ at the one-dimensional segment $[M_1, M_2]$), this

polynomial can be written along the one-dimensional segment $[M_1, M_2] \subset S_2$ in terms of the argument d as

$$P(d) = \begin{cases} P_2(d), \text{ for ND-type classes,} \\ C_2(d), \text{ for D-type classes,} \\ Q_2(d), \text{ for D2-type classes,} \end{cases} \tag{2.13}$$

where the polynomials $P_2(d)$, $C_2(d)$, and $Q_2(d)$ are defined as follows

$$\begin{aligned} P_2(d) &= (1 - \tfrac{2||M_1-M_2||}{\rho^*} + \tfrac{1}{\rho^{*2}}A_2)d^2 - 2\cdot|f^*|, \\ C_2(d) &= (\tfrac{2||M_1-M_2||}{\rho^{*2}} - \tfrac{2}{\rho^{*3}}A_2)d^3 + (1 - \tfrac{4||M_1-M_2||}{\rho^*} + \tfrac{3}{\rho^{*2}}A_2)d^2 - 2\cdot|f^*|, \\ Q_2(d) &= [-\tfrac{6||M_1-M_2||}{\rho^{*4}} + \tfrac{6}{\rho^{*5}}A_2 + \tfrac{1}{\rho^{*3}}(1-\tfrac{\delta}{2})]d^5 + \\ &\quad +[\tfrac{16||M_1-M_2||}{\rho^{*3}} - \tfrac{15}{\rho^{*4}}A_2 - \tfrac{3}{\rho^{*2}}(1-\tfrac{\delta}{2})]d^4 + \\ &\quad +[-\tfrac{12||M_1-M_2||}{\rho^{*2}} + \tfrac{10}{\rho^{*3}}A_2 + \tfrac{3}{\rho^*}(1-\tfrac{\delta}{2})]d^3 + \tfrac{1}{2}\delta d^2 - 2\cdot|f^*|, \end{aligned} \tag{2.14}$$

with ρ^* being the radius of the attraction region S_2 of M_2 (one of the five user-defined parameters in the GKLS-generator), $A_2 = ||M_1 - M_2||^2 + t + 2\cdot|f^*|$ (t is from (2.14)) and δ being a properly chosen random number (see [65] for details).

The distance d^* in (2.12) can be, therefore, found as the solution to the nonlinear equation

$$E_d(d) := P(d) - f^* = 0, \tag{2.15}$$

where $P(d)$ is from (2.13)–(2.14). It can be seen from (2.15) that $E_d(0) = -2\cdot|f^*| - f^* < 0$. On the other hand, $E_d(\rho^*) = P(\rho^*) - f^* > 0$ due to the fact that $P(\rho^*)$ is the polynomial (2.13) value at the border of S_2 (and so it is equal to a value of the paraboloid Z from (2.9)) and, by the generator settings, $f^* < t = \min\limits_{x\in\Omega} z(x)$, where $z(x)$ is from (2.9). All the values $P(d)$ with $d \in [0, \rho^*]$ correspond to the functions $P_2(d)$, $C_2(d)$, or $Q_2(d)$ from (2.14) and, as a consequence, to the polynomials $P_2(x)$, $C_2(x)$, or $Q_2(x)$ with $x \in S_2$. Since the point M_2 is the unique local minimizer of these polynomials over the ball S_2, they are monotonous in the direction $(M_1 - M_2)$. Thus, $P(d)$ is also monotonous along the one-dimensional segment $[0, \rho^*]$, with its values of opposite signs on the endpoints 0 and ρ^*. It follows that the solution to (2.15) exists, is unique and the location of the global minimizer x^* is uniquely defined in (2.12). This location can be determined either analytically for the functions of ND-type, or numerically for the functions of D- or D2-type by using some simple numerical methods (e. g., bisection method) for finding solution to (2.15) over a one-dimensional segment. Once defined, the global

minimizer x^* of the constrained Emmental-type test problem is provided to the user.

Four types of constraints are proposed for the Emmental-type constrained test problems. Satisfaction of each constraint $g_j(x)$, $1 \leq j \leq p$, from (2.7) gives the exterior of a ball with a center G_j and a radius r_j, i. e.,

$$g_j(x) = r_j - ||x - G_j|| \leq 0, \quad 1 \leq j \leq p. \tag{2.16}$$

Constraints of the first type are related to the local minimizers of the corresponding box-constrained Emmental-type test problem; the number p_1 of these constraints is such that $3 \leq p_1 \leq m$. Constraints of the second type are connected with the vertices of Ω from (2.8); the number p_2 of these constraints is such that $0 \leq p_2 \leq 2^N$, where N is the problem dimension from (2.8). Constraints of the third type are constructed by definition (with some safeguards) of a number of random balls in Ω and their number $p_3 \geq 0$. Finally, p_4 constraints ($p_4 \geq 0$) of the fourth type ensure the required number $N_{active} \geq 1$ of active constraints, $p_4 = N_{active} - 1$. They are generated by defining several random balls of the same radius in such a way that they cross at the global minimizer x^* from (2.12). The total number of nonlinear constraints is equal to $p = p_1 + p_2 + p_3 + p_4$, $p \geq 3$. Fig. 2.22 illustrates constraints of different types for the differentiable objective function from Fig. 2.21.b.

Let us describe more in detail how nonlinear constraints are generated in a constrained Emmental-type test problem, starting from p_1 constraints of the first type. A ball with its center at the point $G_1 = M_1$ being the paraboloid vertex from (2.9) is considered as constraint $g_1(x)$ in (2.16). Radius r_1 of this ball (see Fig. 2.22.a) is taken as the distance between M_1 and the attraction region of the global minimizer $M_{i_{\min}}$ ($i_{\min}$ is from (2.11)) of the corresponding box-constrained Emmental-type test problem, ensuring meanwhile that the balls S_1 and S_2 do not overlap, i. e.,

$$r_1 = \min\{||M_1 - M_{i_{\min}}|| - \rho_{i_{\min}}, ||M_2 - M_1|| - \rho_2\}. \tag{2.17}$$

In order to satisfy the first requirement for constrained tests described previously, the ball $S_{i_{\min}}$ ($i_{\min}$ is from (2.11)) is taken as the following constraint of the first type (see Fig. 2.22.a),

$$g_2(x) = \rho_{i_{\min}} - ||x - M_{i_{\min}}||.$$

To place the global minimizer of the constrained test problem on the boundary of the feasible region D from (2.7), the following constraint $g_3(x)$ in (2.16) is built by using a ball with its center at the point

$$G_3 = -\frac{\rho_2 - d^*}{||M_1 - M_2||} M_1 + (1 + \frac{\rho_2 - d^*}{||M_1 - M_2||}) M_2, \tag{2.18}$$

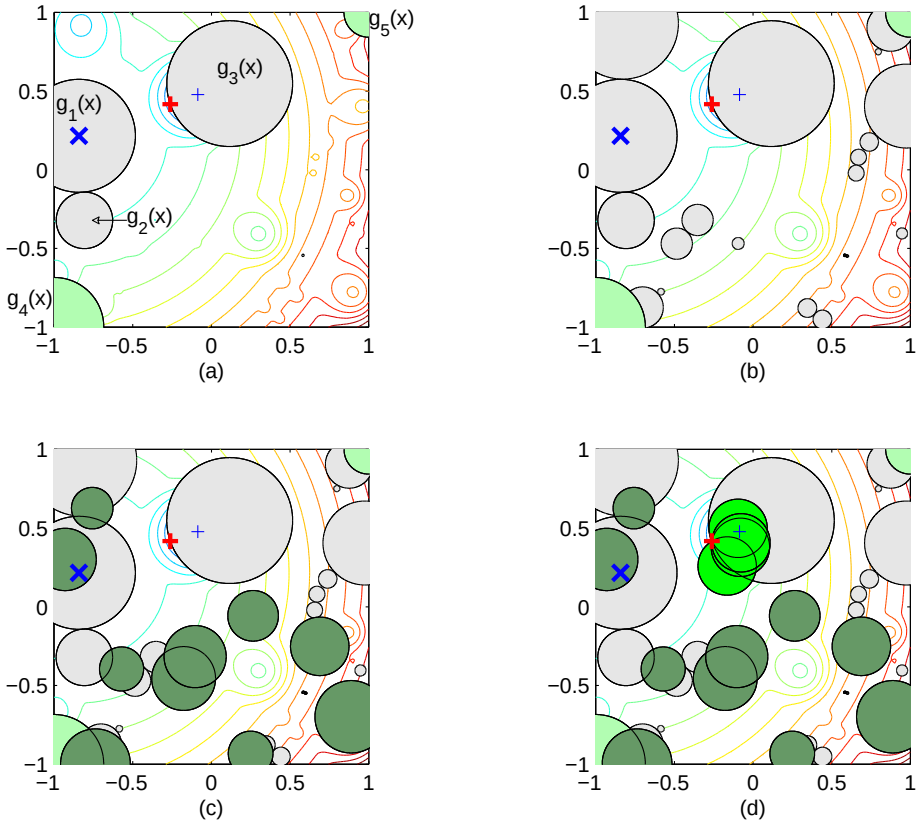


Figure 2.22: The feasible regions and the level curves of the constrained Emmental-type problem from Fig. 2.21.b with the following numbers of constraints (a) $p_1 = 3$, $p_2 = 2$, $p_3 = 0$, $N_{active} = 1$; (b) $p_1 = 20$, $p_2 = 2$, $p_3 = 0$, $N_{active} = 1$; (c) $p_1 = 20$, $p_2 = 2$, $p_3 = 10$, $N_{active} = 1$; (d) $p_1 = 20$, $p_2 = 2$, $p_3 = 10$, $N_{active} = 5$. The admissible region in (d) contains 3 disjoint subregions. p_1 constraints of the first type are in light grey; p_2 constraints of the second type are in light green; p_3 constraints of the third type are in dark green; and $p_4 = N_{active} - 1$ constraints of the fourth type are in green. The vertex of the paraboloid is indicated by $\times$, the global minimizer of the constrained problem is indicated by red bold $+$, and the global minimizer of the corresponding box-constrained D-type GKLS problem is indicated by blue thin $+$.

where d^* is from (2.12) (as found by (2.15)), and radius $r_3 = \rho_2$ (see the constraint $g_3(x)$ in Fig. 2.22.a). This choice of the radius value r_3 ensures that the ball (G_3, r_3) contains in its interior all possible points x such that $f(x) = f^*$ and that only the global minimizer x^* is placed on its border.

Moreover, as it follows from subsection 2.2, the function gradient value at x^* from (2.12) is different from zero (in Fig. 2.21.b, the gradient $\nabla f(x^*) \simeq (-7.843, -2.716)$).

Finally, the other possible constraints $g_j(x)$, $4 \le j \le p_1$, of the first type are constructed by eliminating from Ω in (2.8) some balls $S_{k(j)}$, $3 \le k(j) \le m$, $k(j) \ne i_{\min}$ ($i_{\min}$ is from (2.11)) being the attraction regions of the corresponding local minimizers in Ω (i. e., $r_j = \rho_{k(j)}$ and $G_j = M_{k(j)}$, $4 \le j \le p_1$, in (2.16)), see light grey balls in Fig. 2.22.b–d.

It should be noticed that, by construction of the GKLS tests (see [65]), the balls corresponding to the constraints of the first type do not overlap.

In order to make the admissible region more challenging, $p_2 \ne 0$ constraints of the second and $p_3 \ne 0$ constraints of the third types can be constructed as follows. For generating constraints of the second type some vertices

$$c^q = (c_1^q, \ldots, c_N^q), \quad c_i^q \in \{a_i, b_i\}, \quad 1 \le q \le p_2,$$

of the search hyperinterval Ω from (2.8) are randomly taken. For each selected vertex c^q, the nearest local or global minimizer $M_{j(q)}$ is found even if the ball $S_{j(q)}$ has been already used for constructing a constraint of the first type. So, the $(p_1 + q)$-th constraint in (2.16) is built as a ball with center $G_{p_1+q} = c^q$ and radius

$$r_{p_1+q} = ||c^q - M_{j(q)}||, \quad 1 \le q \le p_2,$$

if the global minimizer x^* from (2.12) does not belong to this ball, or with radius

$$r_{p_1+q} = \frac{1}{2}||c^q - x^*||, \quad 1 \le q \le p_2, \tag{2.19}$$

otherwise. Constraints of the second type are shown by light green in Fig. 2.22.b–d.

It should be noticed that constraints of the second type can intersect constraints of the first type (including the constraint $g_3(x)$ related to the global solution of the constrained test problem), but (due to the choice of the radii r_{p_1+q} by (2.19)) they do not cover the neighborhood of the global minimizer x^* of the constrained Emmental-type test problem.

Randomly generated p_3 constraints of the third type are built in such a way that they do not intersect the attraction region of the global solution M_2 to the original box-constrained GKLS problem. Consequently, they do not cover the global minimizer x^* of the corresponding constrained Emmental-type problem and x^* is also not isolated due to this fact. In particular, random points $G_{p_1+p_2+j}$, $1 \le j \le p_3$, are taken as centers of the balls (2.12) in Ω from (2.8). Radii $r_{p_1+p_2+j}$ are also chosen randomly within interval

$[\rho_{\text{avg}}, 2\rho_{\text{avg}}]$ (where ρ_{avg} is the average between ρ_i, $1 \le i \le m$, from (2.10)), ensuring that if a ball with a center in $G_{p_1+p_2+j}$ and a radius $r_{p_1+p_2+j}$ intersects with the ball S_2, it is replaced by other random ball to avoid this intersection.

Let us finally consider the last, fourth, type of constraints. Up to now, only constraint $g_3(x)$ is active at the global solution x^* from (2.12). In order to allow the user to choose the number $N_{active} \ge 1$ of active constraints at the global minimizer x^*, $p_4 = N_{active} - 1$ different random points $G_{p_1+p_2+p_3+j}$, $1 \le j \le p_4$, are taken on the sphere with the center x^* and the radius d^* from (2.12) (determined by (2.15)), such that the angle α_j between the vectors $G_{p_1+p_2+p_3+j} - x^*$ and $M_2 - x^*$ is strictly smaller than $\frac{\pi}{2}$. The balls with these centers $G_{p_1+p_2+p_3+j}$, $1 \le j \le p_4$, and the radius d^* from (2.12) are taken, thereby, as p_4 constraints of the fourth type in (2.16) and, as a consequence, the number N_{active} of active constraints at x^* is controllable by the user.

Let us now demonstrate a theoretical result, important from the viewpoint of practical applicability of the constrained Emmental-type problems in benchmarking global optimization software. We need the following definition first:

Definition 2.1. *A global minimizer x^* of a constrained test problem is called* accessible *if it is not isolated and its neighborhood has always a positive volume.*

Theorem 2.1. *The global minimizer x^* of a constrained Emmental-type test problem is known and always accessible.*

Proof. By construction, the global minimizer x^* of the constrained problem is known, see (2.12)–(2.15). Moreover, due to the same (2.12)–(2.15), it is an interior point of the attraction region S_2 of the point M_2 (Fig. 2.22.a illustrates this fact). It remains to prove now that x^* from (2.12) is not isolated and its neighborhood has a positive volume.

Let us study the constraints of the constrained Emmental-type test problems. Constraint $g_1(x)$ of the first type can touch (for some random combinations in (2.17)) the ball S_2 only at one point at the boundary of S_2 but, due to (2.12)–(2.15), $g_1(x)$ does not intercept the neighborhood of the point x^* inside of S_2. Constraint $g_2(x)$ of the same first type does not intersect the ball S_2, by construction of the GKLS tests (see [65]). Constraint $g_3(x)$ of the first type touches the point x^*, but there is always some subregion of S_2 which is not covered by $g_3(x)$: this subregion is placed on the opposite side with respect to the point G_3 from (2.18) and, as a consequence, to the

point M_2. Finally, the other constraints of the first type do not intersect S_2, by construction of the GKLS tests.

Constraints of the second type leave always some subregion of a positive volume around x^* even if they intersect S_2 (see (2.19)). Constraints of the third type do not intersect the ball S_2, by their construction (as described just before this Theorem). All p_4 constraints of the fourth type cross at the point x^*. Therefore, let us study this type of constraints better.

Recall that, by construction, points $G_{p_1+p_2+p_3+j}$, $1 \le j \le p_4 = N_{active} - 1$, are placed in such a way that the angle α_j between the vectors $G_{p_1+p_2+p_3+j} - x^*$ and $M_2 - x^*$ is strictly smaller than $\frac{\pi}{2}$. This means that even if there would be two symmetric constraints g_{j_1} and g_{j_2}, $1 \le j_1, j_2 \le p_4$, $j_1 \ne j_2$, with respect to the point x^*, i. e., two constraints with the corresponding angles $\alpha_{j_1} = -\alpha_{j_2} = \frac{\pi}{2}$, the balls in (2.16) with centers G_{j_1}, G_{j_2} and radii r_{j_1}, r_{j_2} would be tangential balls with respect to each other. Thus, even in this (worst) case there would be always a subregion of S_2 with a positive volume in the direction from M_2 to the paraboloid vertex M_1. Hence, although only one constraint of the first type $g_3(x)$ and the constraints of the fourth type cross at the point x^*, they, however, leave a subregion of S_2 with a positive volume that is not covered by them in the direction from M_2 to M_1. So, the global solution x^* is always accessible from this subregion, that is, x^* is not isolated and belongs to a subregion of a positive volume.

□

Since in practice the objective function can be undefined in the unfeasible regions, the control parameter $defined\_out$ can be used in order to keep track of this fact. If $defined\_out = 1$ then the objective function $f(x)$ can be calculated at each point x belonging to Ω from (2.8). Otherwise, $f(x)$ can be calculated only in the feasible points from (2.7). In this case, the value of the function $f(x)$ in the unfeasible points can be set, e. g., equal to ϵ^{-1}, where ϵ is the machine precision.

The proposed Emmental-type GKLS-based generator of constrained test problems generate ND, D, and D2 classes of 100 test functions each. The global minimizer $M_{i_{\min}}$ ($i_{\min}$ is from (2.11)) of a box-constrained Emmental-type test problem of the class is placed in a random point with a random radius of its attraction region. The global solution x^* from (2.12) to the respective constrained Emmental-type test problem is built in such a way that it is unique and located in the neighborhood of the global minimizer M_2 of the original box-constrained GKLS test problem. Due to this fact, the control parameters of the original GKLS-generator of test problems can be used in order to simplify or to complicate the test functions: the problem dimension; the number of local minima (i. e., the upper bound for the number p_1 of constraints of the first type); the value f^* of the global minimum

(as a consequence, the value of the global minimum of the box-constrained Emmental-type test problems fixed to $-3\cdot|f^*|$ in order to be smaller than the global minimum value of the original box-constrained GKLS test problem); the radius of the attraction region of the nearest local minimizer to the solution of the constrained Emmental-type test problem (and, as a consequence, the radius of constraint $g_3(x)$ of the first type); and the distance between the nearest from the paraboloid vertex local minimizer and the solution to the constrained test problem. The other necessary parameters regarding each objective function $f(x)$ from a particular test class are chosen randomly and checked automatically as in the GKLS-generator from [65].

In addition to the five parameters of the GKLS-generator of test problems, the Emmental-type GKLS-based generator has the following control parameters to be fixed by the user:

(1) the number of constraints of the first type, p_1, $3 \leq p_1 \leq m$, where $m \geq 3$ is the number of local minima (default value is $p_1 = 3$);

(2) the number of constraints of the second type, p_2, $0 \leq p_2 \leq 2^N$, where N is the problem dimension from (2.8) (default value is $p_2 = 0$);

(3) the number of constraints of the third type, p_3, $0 \leq p_3$ (default value is $p_3 = 0$);

(4) the number N_{active} of active constraints at the global solution x^*, $N_{active} \geq 1$ and $N_{active} = p_4 + 1$, where $p_4 \geq 0$ is the number of constraints of the fourth type (default value is $N_{active} = 1$, i. e., $p_4 = 0$);

(5) the parameter $defined\_out$, $defined\_out \in \{0, 1\}$ (default value is 1).

Once all the user-defined parameters are defined, the generator produces a class of 100 constrained test problems (of ND-, D-, or D2-type) and provides the user with a comprehensive information about each test function, including the location of all minimizers of the box-constrained problem, their attraction regions and function values, as well as the global minimizer and the global minimum value of the constrained and box-constrained Emmental-type problem.

To conclude, the presented here Emmental-type GKLS-based generator of non-differentiable, differentiable, and twice continuously differentiable classes, each containing 100 random constrained test problems with similar characteristics, has the following properties:

- For each particular user-defined Emmental-type test class, the global minimizers of both a box-constrained and the corresponding nonlinearly constrained Emmental-type problem $f(x)$ are known and they do

not coincide. In particular, the unique global minimizer x^* of the constrained Emmental-type problem is placed on the boundary of the admissible region and is always accessible.

- The difficulty of the constrained problem can be changed in several ways. First, the control parameters of the original box-constrained GKLS-generator can be used in order to simplify or to complicate the objective function $f(x)$ (see, e. g., [188]). Then, different numbers of constraints of the first, second, and third type can also be used in order to simplify or to complicate the admissible region. In particular, the simplest admissible region has $p_1 = 3$, $p_2 = 0$, and $p_3 = 0$ constraints (see light grey balls in Fig. 2.22.a and the corresponding objective function surface in Fig. 2.21.b) and the hardest one has $p_1 = m$, $p_2 = 2^N$ and $p_3 >> 0$ constraints. Moreover, the admissible region can be simply connected, biconnected or multiconnected.

- The gradient $\nabla f(x^*)$ of the function $f(x)$ at the global minimizer x^* is not zero.

- The number of active constraints at the global solution x^* can be changed by the user.

- Finally, the simplicity of the proposed generator allows the user to introduce his/her own constraints of a different type.

Chapter 3

Handling of Ill-Conditioning in Optimization via Infinity Computing

It is well-known that in ill-conditioned systems, numerical methods can lead to incorrect results. Moreover, in practice it is not always possible to work simultaneously with large and small numbers due to overflows and underflows present if traditional computers and numeral systems are used. However, this is not the case when the Infinity Computing framework is applied. In this Chapter, the Infinity Computing is used in order to handle of several instances of ill-conditioning in optimization.

This Chapter is organized as follows. In the first Section, the Infinity Computing paradigm is described briefly. It is shown that working with infinite and infinitesimal numbers simultaneously does not lead to ill-conditioning if the Infinity Computing is applied.

The second Section is dedicated to handling of ill-conditioning in univariate Lipschitz global optimization. It is shown that in cases, when it is required to solve the scaled problem with small or large scaling constants, ill-conditioning can be provoked by scaling. It is shown also that this situation can be avoided using numerical infinities and infinitesimals.

Finally, the third Section is dedicated to handling of ill-conditioning in multidimensional optimization. The variable-metric Diagonal Bundle method with limited memory (D-Bundle method) is studied. It is shown that the matrix defining the metric can be ill-conditioned due to discontinuities in the derivatives. A new approach to approximate this matrix using the Infinity Computing is proposed. It is shown that the obtained matrix is always well-conditioned.

3.1 Infinity Computing methodology

In our everyday activities with finite numbers the *same* finite numerals[1] are used for *different* purposes (e.g., the same numeral 9 can be used to express the number of elements of a set, to indicate the position of an element in a sequence, and to execute practical computations). In contrast, when we face the necessity to work with infinities or infinitesimals, the situation changes drastically. In fact, in this case *different* numerals are used to work with infinities and infinitesimals in *different* situations. To illustrate this fact it is sufficient to mention that we use the symbol ∞ in standard analysis, ω for working with ordinals, $\aleph_0, \aleph_1, ...$ for dealing with cardinal numbers..

Many theories dealing with infinite and infinitesimal quantities have a symbolic (not numerical) character. For instance, many implementations of non-standard analysis (see [159]) are symbolic, since they have no numeral systems to express their numbers by a finite number of symbols (the finiteness of the number of symbols is necessary for organizing numerical computations). Namely, if we consider a finite n, then it can be taken $n = 72$, or $n = 30$ or any other numeral used to express finite quantities and consisting of a finite number of symbols. In contrast, if we consider a non-standard infinite m then it is not clear which numerals can be used to assign a concrete value to m. Analogously, in non-standard analysis, if we consider an infinitesimal h then it is not clear which numerals consisting of a finite number of symbols can be used to assign a concrete value to h and to write $h = ...$ In fact, very often in non-standard analysis texts, a *generic* infinitesimal h is used and it is considered as a symbol, i.e., only symbolic computations can be done with it. Approaches of this kind leave unclear such issues, e.g., whether the infinite $1/h$ is integer or not or whether $1/h$ is the number of elements of an infinite set.

In order to allow one to execute *numerical* computations with different infinities and infinitesimals and to use the same numerals in all the situations (as it happens with numerals expressing finite quantities), a new computational methodology and the respective numeral system have been developed in [169, 172, 176]. This numeral system avoids indeterminate forms and situations similar to $\infty + 1 = \infty$ and $\infty - 1 = \infty$ providing results ensuring that if a is a numeral written in this numeral system then for any a (i.e., a can be finite, infinite, or infinitesimal) it follows $a + 1 > a$ and $a - 1 < a$.

[1]There exists an important distinction between *numbers* and *numerals*. A *numeral* is a symbol (or a group of symbols) that represents a *number*. A *number* is a concept that a *numeral* expresses. The same number can be represented by different numerals. For example, the symbols '9', 'nine', 'IIIIIIIII', and'IX', are different numerals, but they all represent the same number

The numeral system is based on a new infinite unit of measure expressed by the numeral ① called *grossone* that is introduced as the number of elements of the set of natural numbers. At the same time, with the introduction of ① in the mathematical language all other symbols (like ∞, Cantor's ω, $\aleph_0, \aleph_1, ...$, etc.) traditionally used to deal with infinities and infinitesimals are excluded from the language because ① and other numbers constructed with its help not only can be used instead of all of them but can be used with a higher accuracy. Analogously, when zero and the positional numeral system had been introduced in Europe, Roman numerals I, V, X, etc. had not been involved and new symbols 0, 1, 2, etc. have been used to express numbers. The new element – zero expressed by the numeral 0 – had been introduced by describing its properties in the form of axioms. Analogously, ① is introduced by describing its properties postulated by the Infinite Unit Axiom added to axioms for real numbers (see [172, 176] for a detailed discussion).

Let us see now how one can write down different numerals expressing different infinities and infinitesimals and to execute computations with all of them. Instead of the usual symbol ∞ different infinite and/or infinitesimal numerals can be used thanks to ①. Indeterminate forms are not present and, for example, the following relations hold for infinite numbers ①, $①^{2.7}$ and $①^{-1}$, $①^{-2.7}$ (that are infinitesimals), as for any other (finite, infinite, or infinitesimal) number expressible in the new numeral system

$$0 \cdot ① = ① \cdot 0 = 0, \quad ① - ① = 0, \quad \frac{①}{①} = 1, \quad ①^0 = 1, \quad 1^① = 1, \quad 0^① = 0, \tag{3.1}$$

$$0 \cdot ①^{-1} = ①^{-1} \cdot 0 = 0, \quad ①^{-1} > 0, \quad ①^{-2.7} > 0, \quad ①^{-1} - ①^{-1} = 0,$$

$$\frac{①^{-1}}{①^{-1}} = 1, \quad (①^{-1})^0 = 1, \quad ① \cdot ①^{-1} = 1, \quad ① \cdot ①^{-2} = ①^{-1},$$

$$\frac{①^{-2.7}}{①^{-2.7}} = 1, \quad \frac{①^{2.7}}{①} = ①^{1.7}, \quad \frac{①^{-1}}{①^{-2}} = ①, \quad ①^{2.7} \cdot ①^{-2.7} = 1.$$

The introduction of the numeral ① allows us to represent infinite and infinitesimal numbers in a unique framework and to work with all of them numerically on the Infinity Computer (see the patent [164]). For this purpose a numeral system similar to traditional positional numeral systems was introduced in [169, 172]. To construct a number C in the numeral positional system with base ①, we subdivide C into groups corresponding to powers of ①:

$$C = c_{p_m}①^{p_m} + \ldots + c_{p_1}①^{p_1} + c_{p_0}①^{p_0} + c_{p_{-1}}①^{p_{-1}} + \ldots + c_{p_{-k}}①^{p_{-k}}. \tag{3.2}$$

Then, the record

$$C = c_{p_m}①^{p_m} \ldots c_{p_1}①^{p_1} c_{p_0}①^{p_0} c_{p_{-1}}①^{p_{-1}} \ldots c_{p_{-k}}①^{p_{-k}} \tag{3.3}$$

represents the number C, where all numerals $c_i \neq 0$, they belong to a traditional numeral system and are called *grossdigits*. They express finite positive or negative numbers and show how many corresponding units $①^{p_i}$ should be added or subtracted in order to form the number C. Note that in order to have a possibility to store C in the computer memory, values k and m should be finite.

Numbers p_i in (3.3) are sorted in the decreasing order with $p_0 = 0$

$$p_m > p_{m-1} > \ldots > p_1 > p_0 > p_{-1} > \ldots p_{-(k-1)} > p_{-k}.$$

They are called *grosspowers* and they themselves can be written in the form (3.3). In the record (3.3), we write $①^{p_i}$ explicitly because in the new numeral positional system the number i in general is not equal to the grosspower p_i. This gives the possibility to write down numerals without indicating grossdigits equal to zero.

The term having $p_0 = 0$ represents the finite part of C since $c_0①^0 = c_0$. Terms having finite positive grosspowers represent the simplest infinite parts of C. Analogously, terms having negative finite grosspowers represent the simplest infinitesimal parts of C. For instance, the number $①^{-1} = \frac{1}{①}$ mentioned above is infinitesimal. Note that all infinitesimals are not equal to zero. In particular, $\frac{1}{①} > 0$ since it is a result of division of two positive numbers.

A number represented by a numeral in the form (3.3) is called *purely finite* if it has neither infinite nor infinitesimal parts. For instance, 14 is purely finite and $14 + 5.3①^{-1.5}$ is not. All grossdigits c_i are supposed to be purely finite. Purely finite numbers are used on traditional computers and for obvious reasons have a special importance for applications. All of the numbers introduced above can be grosspowers, as well, giving thus a possibility to have various combinations of quantities and to construct terms having a more complex structure. In this research, hereinafter all *purely finite* numbers will be called as *finite* just for simplicity.

It should be stressed that the Infinity Computing approach allows us a full numerical treatment of both infinite and infinitesimal numbers whereas the non-standard analysis (see [159]) has a symbolic character and, therefore, allows symbolic computations only (see a detailed discussion on this topic in [182]).

It can be seen from (3.1) that the infinities and infinitesimals of different order can be used simultaneously in this computational system without undetermined forms such as $\infty - \infty$ or $\frac{\infty}{\infty}$: $① - ① = 0$, $① - ①^2 = -1①^2 1①$, $\frac{①}{①^{-1}} = ①^2$, etc. Moreover, it can be easily seen that the ill-conditioned operations in traditional framework can be well-conditioned in the Infinity

Computing framework. For instance, the results of the following operations can be representable in the Infinity Computer: $①^{-1}+①+①^2 = 1①^2 1①^1 1①^{-1}$, $2①^2 - 1.5①^{-3} + 6①^5 = 6①^5 2①^2 - 1.5①^{-3}$. It should be noted that the same operations with very small and large numbers are ill-conditioned in the traditional floating-point arithmetic: for instance, on the traditional computers, the result of $10^{20} + 10^{50}$ will be 10^{50} with the error 10^{20}. In this Chapter, several instances of ill-conditioning in optimization are studied in the framework of the Infinity Computing.

It should be noted also that handling of ill-conditioning is not the unique advantage of the Infinity Computing framework. It is only a particular property that can be successfully used. Another useful property is the possibility to calculate exact derivatives of functions even if they are given as a black box. It can be very useful for the numerical methods, where the derivatives are required. For instance, many numerical methods for solving ordinary differential equations require the computation of the derivative of the unknown function $y(t)$ at some specific points. In particular, this is the case with methods based on Taylor expansion. The following theorem from [177] shows an important result allowing one to compute the derivatives exactly.

Theorem 3.1. *Suppose that: (i) for a function $f(x)$ calculated by a procedure implemented at the Infinity Computer there exists an unknown Taylor expansion in a finite neighborhood $\delta(y)$ of a finite point y; (ii) $f(x)$, $f'(x)$, $f''(x), ..., f^{(k)}(x)$ assume finite values or are equal to zero for $x \in \delta(y)$; (iii) $f(x)$ has been evaluated at a point $y + ①^{-1} \in \delta(y)$. Then the Infinity Computer returns the results of this evaluation in the positional numeral system with the infinite radix ① in the following form*

$$f(y + ①^{-1}) = c_0 ①^0 c_{-1} ①^{-1} c_{-2} ①^{-2} ... c_{-(k-1)} ①^{-(k-1)} c_{-k} ①^{-k}, \tag{3.4}$$

where

$$f(y) = c_0, \;\; f'(y) = c_{-1}, \;\; f''(y) = 2! \cdot c_{-2}, ..., f^{(k)}(y) = k! \cdot c_{-k}. \tag{3.5}$$

Proof. The proof can be found in [177]. □

Let us show how the usage of infinitesimals on the Infinity Computer allows one to calculate exact derivatives of $y(t)$ numerically according to the theorem given above. First, suppose that $y(t)$ is given as a "black box" but it can be evaluated at any point t. Traditionally, in order to approximate the first derivative of y at some point x, the finite integration step h and the finite differences of different order can be used: for instance, the forward differences of the first order:

$$y'(x) \approx \frac{y(x+h) - y(x)}{h}, \tag{3.6}$$

the backward differences of the first order:

$$y'(x) \approx \frac{y(x) - y(x-h)}{h}, \tag{3.7}$$

or the central differences of the second order:

$$y'(x) \approx \frac{y(x+h) - y(x-h)}{2h}. \tag{3.8}$$

It is well-known that the above mentioned finite differences have the differentiation error of order $O(h)$ (forward and backward differences) and $O(h^2)$ (central differences), respectively. However, due to the numerical cancelation errors in the floating-point arithmetic, it is not always possible to obtain smaller errors using smaller values of h.

Example 3.1. *Let us consider the procedure realizing the following function:* $y(t) = \frac{t+1}{t-1}$. *Suppose that we want to compute the first derivative of the function* $y(t)$ *at the point* $x = 3$. *In Fig. 3.1 the relative errors* $\frac{|y'(x)-F_h(x)|}{1+|y'(x)|}$ *in the logarithmic form are presented for different values of* h *for three above mentioned formulae, where* $F_h(x)$ *is the approximation of the first derivative* $y'(x)$ *using (3.6), (3.7) or (3.8), respectively. It can be seen from Fig.3.1, that there is no such the value* h *with which the error will be smaller than* 10^{-12}, *for the second-order difference, and* 10^{-9} *for the first-order differences. Moreover, it can be also seen that starting from the value* $h \approx 8 \cdot 10^{-6}$, $h \approx 3 \cdot 10^{-8}$ *and* $h \approx 1.5 \cdot 10^{-8}$ *for the central, forward and backward differences, respectively, the error significantly increases.*

At the same time, the Infinity Computer performs the following operations evaluating the function $y(t)$ *at the point* $x + ①^{-1}$ *using infinitesimal stepsize* h*:*

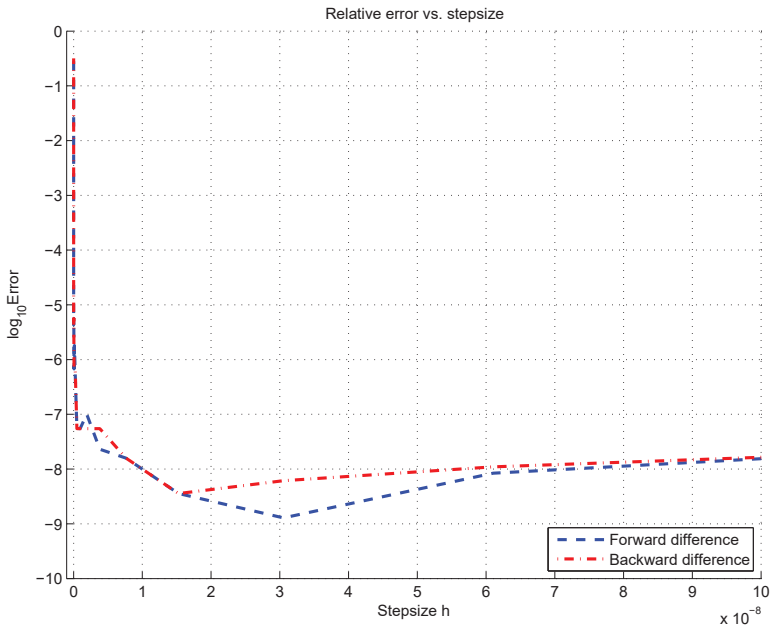
$$y(3+①^{-1}) = \frac{3①^0 + ①^{-1} + 1①^0}{3①^0 + ①^{-1} - 1①^0} = 2①^0 - 0.5①^{-1} + 0.25①^{-2} - 0.125①^{-3} + ..., \tag{3.9}$$

from where, it can be easily obtained that $y(3) = 2$, $y'(3) = -0.5 \cdot 1! = -0.5$, $y''(3) = 0.25 \cdot 2! = 0.5$, $y'''(3) = -0.125 \cdot 3! = -0.75$, *being the exact values of* $y(3)$, $y'(3)$, $y''(3)$, *and* $y'''(3)$.

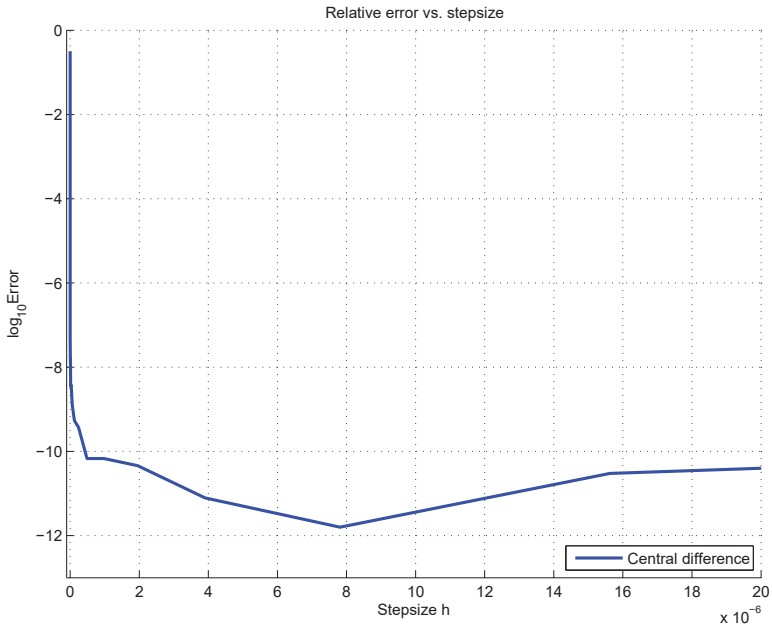
Let us consider now the case, when the function $y(t)$ is not given explicitly. Suppose that we have the following ordinary differential equation:

$$\begin{cases} y' = f(t, y(t)), \quad t \in [t_0, T], \\ y(t_0) = y_0, \end{cases} \tag{3.10}$$

where $f : [t_0, T] \times \mathbb{R}^n \to \mathbb{R}^n$ is assumed sufficiently differentiable.



a) First order differences



b) Second order difference

Figure 3.1: Relative errors vs. stepsize for (a) the forward and backward differences; (b) the central difference

Let us denote by $y_i^{(j)}$ an estimate of the j-th derivative of the solution $y(x)$ at the point x_i and suppose that $f(x, y)$ assumes purely finite values at

purely finite x and y. It has been shown in [181] that in order to calculate the j-th derivative at the point x_i j infinitesimals steps from the point x_i using the Euler formula with $h = ①^{-1}$ should be executed as follows

$$y_{i1} = y_i + ①^{-1} f(x_i, y_i), \qquad y_{i2} = y_{i1} + ①^{-1} f(x_i + ①^{-1}, y_{i1}), \ \ldots$$

$$y_{ik} = y_{ik-1} + ①^{-1} f(x_i + (k-1)①^{-1}, y_{ik-1}).$$

Then, since approximations of the derivatives can be obtained by the forward differences $\Delta_h^j,\ 1 \le j \le k$, with $h = ①^{-1}$ as follows

$$\Delta^k_{①^{-1}} = \sum_{j=0}^{k} (-1)^j C_j^k y_{x_i + (k-j)①^{-1}}, \tag{3.11}$$

we obtain

$$y^{(k)}(x_i) = \frac{\Delta^k_{①^{-1}}}{①^{-k}} + O(①^{-1}). \tag{3.12}$$

Since the error of the approximation is $O(①^{-1})$, the finite part of the value $\frac{\Delta^k_{①^{-1}}}{①^{-k}}$ gives us the *exact*[2] derivative $y^{(k)}(x_i)$.

On the other hand, finite differences may be employed directly on the value of f as follows:

$$y^{(k)}(t_i) = f^{(k-1)}(y_i) = \frac{\Delta^{k-1}_{①^{-1}}}{①^{-k-1}} + O(①^{-1}) \tag{3.13}$$

where now we could use forward differences:

$$\Delta^k_{①^{-1}} = \sum_{j=0}^{k-1} (-1)^j \binom{k-1}{j} f(y_{i,k-j-1}) \quad \text{and } f(y_{i,0}) = f(y_i). \tag{3.14}$$

or in a similar way, we could use central differences. However, since the differentiation error is already infinitesimal using the simplest forward differences of the first order, there is no necessity to use the formulae of the higher order, e.g., central differences.

Example 3.2. *In order to see an example of calculating the derivatives using the presented techniques, let us find $y''(0)$ for the problem*

$$y'(t) = \frac{y - 2ty^2}{1+t},\ y(0) = y_0 = 0.4. \tag{3.15}$$

[2]the word "exact" means that calculations are performed with the machine precision.

One can find that the exact $y''(0) = -0.32$ *since the exact solution of (3.15) is* $y(x) = \frac{1+x}{2.5+x^2}$. *In order to find* $y''(0)$ *we start by calculating* y_1, y_2 *and* $\frac{\triangle^2_{①^{-1}}}{①^{-2}}$:

$$y_1 = 0.4 + ①^{-1} f(0, 0.4) = 0.4 + 0.4①^{-1},$$

$$y_2 = y_1 + ①^{-1} f(①^{-1}, y_1) = 0.4 + 0.8①^{-1} - 0.32①^{-2} - 0.32①^{-3}.$$

$$\frac{\triangle^2_{①^{-1}}}{①^{-2}} = \frac{y_0 - 2y_1 + y_2}{①^{-2}} = \frac{-0.32①^{-2} - 0.32①^{-3}}{①^{-2}} = -0.32 - 0.32①^{-1} = y''(0) + O(①^{-1}),$$

and we have $y''(0) = -0.32$.

At the same time, using the second approach, we can obtain:

$$y_1 = 0.4 + ①^{-1} f(0, 0.4) = 0.4 + 0.4①^{-1},$$

$$\frac{\triangle^1_{①^{-1}}}{①^{-1}} = \frac{f(t_1, y_1) - f(t_0, y_0)}{①^{-1}} =$$

$$= \frac{f(0.4 + 0.4①^{-1}) - 0.4}{①^{-1}} = \frac{0.4 - 0.32①^{-1} + O(①^{-2}) - 0.4}{①^{-1}} =$$

$$= \frac{-0.32①^{-1} + O(①^{-2})}{①^{-1}} = -0.32 + O(①^{-1}) = y''(0) + O(①^{-1}),$$

and we have $y''(0) = -0.32$, *as well. It should be noted, that the second approach requires less computations than the first one, giving the same result.*

3.2 Strong homogeneity of a class of global optimization algorithms working with infinite and infinitesimal scales

As it has been mentioned in the first Chapter, in many applied problems it is required to find the global minimum of multiextremal non-differentiable functions. Due to the presence of multiple local minima and non-differentiability of the objective function, classical local optimization techniques cannot be used for solving these problems and global optimization methods should be developed (see, e.g., [54, 94, 151, 153, 192, 213, 215, 229, 233, 235]). One of the desirable properties of global optimization methods (see [47, 213, 233]) is their *strong homogeneity* meaning that a method produces the same sequences of trial points (i.e., points where the objective function $f(x)$ is evaluated)

independently of both shifting $f(x)$ vertically and its multiplication by a scaling constant. In other words, it can be useful to optimize a scaled function

$$g(x) = g(x; \alpha, \beta) = \alpha f(x) + \beta, \qquad \alpha > 0, \tag{3.16}$$

instead of the original objective function $f(x)$. The concept of strong homogeneity has been introduced in [233] where it has been shown that both the P-algorithm (see [231]) and the one-step Bayesian algorithm (see [137]) are strongly homogeneous. The case $\alpha = 1, \beta \neq 0$ was considered in [47, 213] where a number of methods enjoying this property and called *homogeneous* were studied. It should be mentioned that there exist global optimization methods that are homogeneous or strongly homogeneous and algorithms (see, for instance, the DIRECT algorithm from [94] and a huge number of its modifications) that do not possess this property. These methods belong to the class of "Divide-the-best" algorithms introduced in [168]. Two kinds of algorithms are taken into consideration in this work: geometric and information ones (see [192, 207, 213, 215]).

In this research, it will be shown that several fast univariate methods using local tuning techniques to accelerate the search through a smart balancing of the global and local information collected during the search (see recent surveys in [192, 205]) enjoy the property of the strong homogeneity (see [200] for details). In particular, it will be proved that this property is valid for the considered methods not only for finite values of the constants α and β but for infinite and infinitesimal ones, as well. To prove this result, a new class of global optimization problems with the objective function having infinite or infinitesimal Lipschitz constants is introduced. Numerical computations with functions that can assume infinite and infinitesimal values are executed using the Infinity Computing paradigm allowing one to work *numerically* with a variety of infinities and infinitesimals on a patented in Europe and USA new supercomputer called the Infinity Computer (see, e.g., surveys [172, 182]).

3.2.1 A class of global optimization problems with infinite and infinitesimal Lipschitz constants

The importance to have the possibility to work with infinite and infinitesimal scaling/shifting constants α and β has an additional value due to the following fact. It can happen that even if a method possesses the strong homogeneity property theoretically and the original objective function $f(x)$ is well-conditioned, numerically very small and/or large finite constants α and β can lead to the ill-conditioning of the global optimization problem

involving $g(x)$ due to overflow and underflow taking place when $g(x)$ is constructed from $f(x)$. Thus, global minimizers can change their locations and the values of global minima can change, as well. As a result, applying methods possessing the strong homogeneity property to solve these problems will lead to finding the changed values of minima related to $g(x)$ and not the desired global solution of the original function $f(x)$ we are interested in. In this work, it is shown that numerical infinities and infinitesimals and the Infinity Computing framework can help in this situation.

Let us consider the following univariate global optimization problem where it is required to find the global minimum f^* and global minimizers x^* such that

$$f^* = f(x^*) = \min\ f(x), \quad x \in D = [a, b] \subset \mathbb{R}. \tag{3.17}$$

It is supposed that the objective function $f(x)$ can be multiextremal and non-differentiable. Moreover, the objective function $f(x)$ is supposed to be Lipschitz continuous over the interval D, i.e., $f(x)$ satisfies the following condition

$$|f(x_1) - f(x_2)| \leq L|x_1 - x_2|, \quad x_1, x_2 \in D, \tag{3.18}$$

where L is the Lipschitz constant, $0 < L < \infty$.

A vast literature is dedicated to the problem (3.17), (3.18) and algorithms for its solving (see, e.g., [54, 67, 79, 107, 151, 192, 198, 207, 215, 229]). In particular, in practice it can be useful to optimize a scaled function $g(x)$ from (3.16) instead of the original objective function $f(x)$ (see, e.g., [47, 213, 233]). For this kind of problems, the concept of strong homogeneity for global optimization algorithms has been introduced in [233]: An algorithm is called *strongly homogeneous* if it generates the same sequences of trials (evaluations of the objective function) during optimizing the original objective function $f(x)$ and the scaled function $g(x)$ from (3.16), where $\alpha > 0$ and β are constants (notice that homogeneous methods corresponding to the case $\alpha = 1, \beta \neq 0$ have been considered originally in [47, 213]). Unfortunately, in practice it is not always possible to obtain correct values of $g(x)$ for huge and small values of $\alpha > 0$ and β due to overflows and underflows present if traditional computers and numeral systems are used for evaluation of $g(x)$ even if the original function $f(x)$ is well-conditioned.

As an illustration, let us consider the following test problem from [83] shown in Fig. 3.2.a:

$$f_3(x) = \sum_{k=1}^{5} -k \cdot sin[(k+1)x + k], \quad x \in D = [-10, 10]. \tag{3.19}$$

The function $f_3(x)$ has been chosen from the set of 20 test functions described in [83] because it has the highest number of local minima among these

functions and the following three global minimizers

$$x_1^* = -0.491, \quad x_2^* = -6.775, \quad x_3^* = 5.792 \tag{3.20}$$

corresponding to the global minimum

$$f^* = f(x_1^*) = f(x_2^*) = f(x_3^*) = -12.0312. \tag{3.21}$$

Let us take $\alpha = 10^{-17}$ and $\beta = 1$ obtaining so the following function

$$g_3(x) = 10^{-17} f_3(x) + 1. \tag{3.22}$$

It can be seen from Fig. 3.2.a and Fig. 3.2.b that $f_3(x)$ and $g_3(x)$ are completely different. If we wish to reestablish $f_3(x)$ from $g_3(x)$, i.e., to compute the inverted scaled function $\widehat{f}_3(x) = 10^{17}(g_3(x)-1)$, then it will not coincide with $f_3(x)$. Fig. 3.2.c shows $\widehat{f}_3(x)$ constructed from $g_3(x)$ using MATLAB® and the piece-wise linear approximations with the integration step $h = 0.0001$.

Thus, this scaling leads to an ill-conditioning. Due to underflows taking place in commonly used numeral systems (in this case, the type *double* in MATLAB®), the function $g_3(x)$ degenerates over many intervals in constant functions and many local minimizers disappear (see Fig. 3.2.b). In the same time, due to overflows, several local minimizers become global minimizers of the scaled function $g_3(x)$. In particular, using the following two commands in MATLAB®

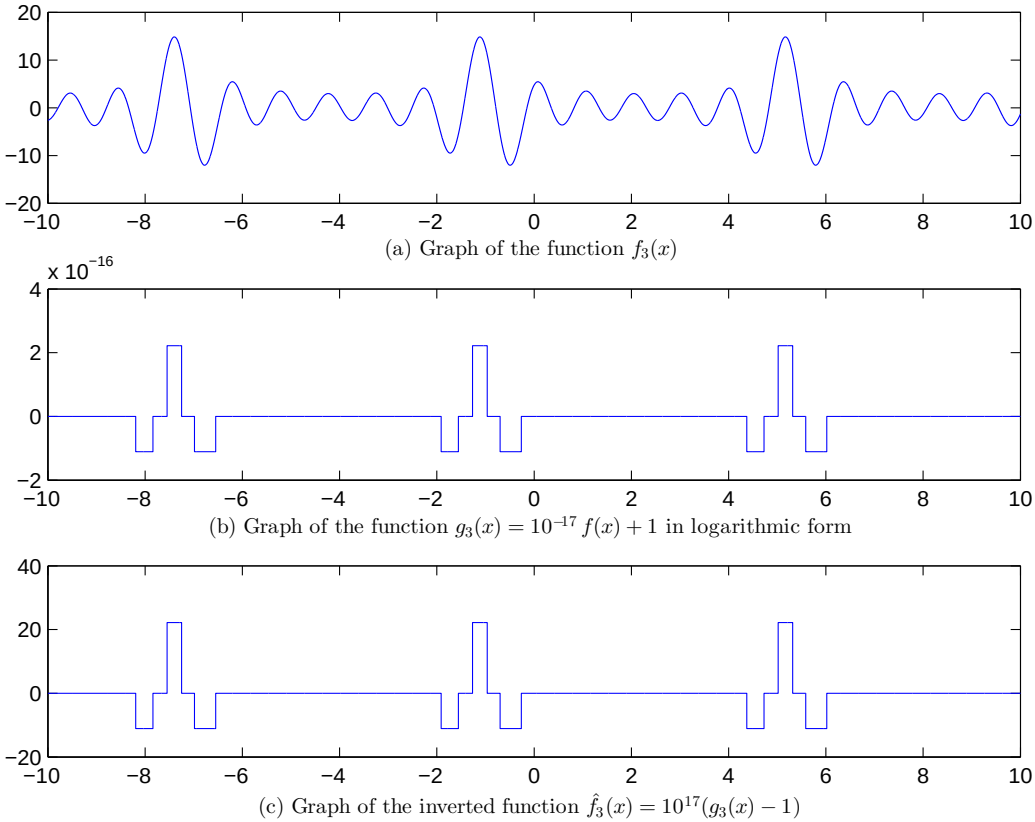
$$[gmin,\ imin] = min(y), \qquad xmin = x(imin)$$

we can calculate an approximation of the global minimum for $g_3(x)$. Using the array y containing the values of $g_3(x)$ calculated with the stepsize $h = 0.0001$, i.e.,

$$y_i = 10^{-17} f_3(x_i) + 1, \qquad x_i = -10 + h \cdot (i-1),\ i \geq 1,$$

we get $(xmin, gmin) = (-8.194, 1.0)$ being an approximation of the global minimum $(x^*, g_3(x^*))$ of $g_3(x)$ that does not coincide with the global minima (3.20), (3.21) of the original function $f_3(x)$. Thus, due to underflows and overflows, the "wrong" global minimum of the scaled function $g_3(x)$ has been found. Analogously, due to the same reasons, the inverted function $\widehat{f}_3(x) = 10^{17}(g_3(x) - 1)$ has also different global minima with respect to the original function $f_3(x)$ (see Fig. 3.2.c). Clearly, a similar situation can be observed if larger values of α and β are used (for instance, $\alpha = 10^{17}$ and $\beta = 10^{35}$).

This example shows that in case of very huge or very small finite values of constants α and β, even if it has been proved theoretically that a method



(a) Graph of the function $f_3(x)$

(b) Graph of the function $g_3(x) = 10^{-17} f(x) + 1$ in logarithmic form

(c) Graph of the inverted function $\hat{f}_3(x) = 10^{17}(g_3(x) - 1)$

Figure 3.2: Graphs of (a) the test function (3.19), (b) the scaled function $g_3(x)$ from (3.22) in the logarithmic form, (c) the inverted scaled function $\widehat{f}_3(x) = 10^{17}(g_3(x) - 1)$. It can be seen that the form of the functions $g_3(x)$ and $\widehat{f}_3(x)$ are qualitatively different with respect to the original function $f_3(x)$ due to overflows and underflows.

is strongly homogeneous, it does not make sense to talk about this property since it is not possible to construct correctly the corresponding scaled functions on the traditional computers.

The introduction of the Infinity Computer paradigm allows us to consider univariate global optimization problems with the objective function $g(x)$ from (3.16) that can assume not only finite values, but also infinite and infinitesimal ones. It is supposed that the original function $f(x)$ can assume finite values only and it satisfies condition (3.18) with a finite constant L. However, since in (3.16) the scaling/shifting parameters α and β can be not only finite, but also infinite and infinitesimal and, therefore, to work with $g(x)$, the Infinity Computing framework is required. Thus, the following

optimization problem is introduced

$$\min\ g(x) = \min\ (\alpha f(x) + \beta), \quad x \in D = [a, b] \subset \mathbb{R}, \alpha > 0, \tag{3.23}$$

where the function $f(x)$ can be multiextremal, non-differentiable, and Lipschitz continuous with a finite value of the Lipschitz constant L from (3.18). In their turn, the values α and β can be finite, infinite, and infinitesimal numbers representable in the numeral system (3.3).

The finiteness of the original Lipschitz constant L from (3.18) is the essence of the Lipschitz condition allowing people to construct optimization methods for traditional computers. The scaled objective function $g(x)$ can assume not only finite, but also infinite and infinitesimal values and, therefore, in these cases it is not Lipschitzian in the traditional sense. However, the Infinity Computer paradigm extends the space of functions that can be treated theoretically and numerically to functions assuming infinite and infinitesimal values. This fact allows us to extend the concept of Lipschitz functions to the cases where the Lipschitz constant can assume infinite/infinitesimal values.

Let us indicate hereinafter by "$\widehat{\ }$" all the values related to the function $g(x)$ from (3.23) and without "$\widehat{\ }$" the values related to the function $f(x)$. The following lemma shows a simple but important property of the Lipschitz constant for the objective function $g(x)$.

Lemma 3.1. *The Lipschitz constant $\widehat{L}$ of the function $g(x) = \alpha f(x) + \beta$, where $f(x)$ assumes only finite values and has the finite Lipschitz constant L over the interval $[a, b]$ and $\alpha,\ \alpha > 0$, and β can be finite, infinite, and infinitesimal, is equal to αL.*

Proof. The following relation can be obtained from the definition of $g(x)$ and the fact that $\alpha > 0$

$$|g(x_1) - g(x_2)| = \alpha |f(x_1) - f(x_2)|, \quad x_1,\ x_2 \in [a, b].$$

Since L is the Lipschitz constant for $f(x)$, then

$$\alpha |f(x_1) - f(x_2)| \leq \alpha L |x_1 - x_2| = \widehat{L} |x_1 - x_2|, \quad x_1,\ x_2 \in [a, b],$$

and this inequality proves the lemma. □

Thus, the new Lipschitz condition for the function $g(x)$ from (3.16) can be written as

$$|g(x_1) - g(x_2)| \leq \alpha L |x_1 - x_2| = \widehat{L} |x_1 - x_2|, \quad x_1, x_2 \in D, \tag{3.24}$$

where the constant L from (3.18) is finite and the quantities α and $\widehat{L}$ can assume infinite and infinitesimal values.

Notice that in the introduced class of functions infinities and infinitesimals are expressed in numerals (3.3), and Lemma 1 describes the first property of this class. Notice also that symbol ∞ representing a generic infinity cannot be used together with numerals (3.3) allowing us to distinguish a variety of infinite (and infinitesimal) numbers. Analogously, Roman numerals (I, II, III, V, X, etc.) that do not allow to express zero and negative numbers are not used in the positional numeral systems where new symbols (0, 1, 2, 3, 5, etc.) are used to express numbers.

Some geometric and information global optimization methods (see [153, 155, 192, 205, 207, 213, 215]) used for solving the traditional Lipschitz global optimization problem (3.17) are adopted hereinafter for solving the problem (3.23). A general scheme describing these methods is presented in the next subsection.

3.2.2 Univariate Lipschitz global optimization algorithms and strong homogeneity

Methods studied in this research have a similar structure and belong to the class of "Divide-the-best" global optimization algorithms introduced in [168]. They can have the following differences in their computational schemes distinguishing one algorithm from another:

(i) Methods are either Geometric or Information (see [192, 213, 215] for detailed descriptions of these classes of methods);

(ii) Methods can use different approaches for estimating the Lipschitz constant: an a priori estimate, a global adaptive estimate, and two local tuning techniques: Maximum Local Tuning (MLT) and Maximum-Additive Local Tuning (MALT) (see [192, 205, 215] for detailed descriptions of these approaches).

The first difference, (i), consists of the choice of characteristics R_i for the subintervals $[x_{i-1}, x_i]$, $2 \leq i \leq k$, where the points x_i, $1 \leq i \leq k$, are called *trial points* and are points where the objective function $g(x)$ has been evaluated during previous iterations:

$$R_i = \begin{cases} \frac{z_i+z_{i-1}}{2} - l_i\frac{x_i-x_{i-1}}{2}, & \text{for geometric methods,} \\ 2(z_i+z_{i-1}) - l_i(x_i-x_{i-1}) - \frac{(z_i-z_{i-1})^2}{l_i(x_i-x_{i-1})}, & \text{for information methods,} \end{cases} \tag{3.25}$$

where $z_i = g(x_i)$ and l_i is an estimate of the Lipschitz constant for the subinterval $[x_{i-1}, x_i],\ 2 \le i \le k$.

The second distinction, (ii), is related to four different strategies used to estimate the Lipschitz constant L. The first one consists of applying an a priori given estimate $\overline{L} > L$. The second way is to use an adaptive global estimate of the Lipschitz constant L during the search (the word *global* means that the same estimate is used for the whole region D). The global adaptive estimate $\overline{L}_k$ can be calculated as follows

$$\overline{L}_k = \begin{cases} r \cdot H^k, & \text{if } H^k > 0, \\ 1, & \text{otherwise,} \end{cases} \tag{3.26}$$

where $r > 0$ is a reliability parameter and

$$H^k = \max\{H_i : 2 \le i \le k\}, \tag{3.27}$$

$$H_i = \frac{|z_i - z_{i-1}|}{x_i - x_{i-1}},\ 2 \le i \le k. \tag{3.28}$$

Finally, the Maximum (MLT) and Maximum-Additive (MALT) local tuning techniques consist of estimating local Lipschitz constants l_i for each subinterval $[x_{i-1}, x_i], 2 \le i \le k$, as follows

$$l_i^{MLT} = \begin{cases} r \cdot \max\{\lambda_i, \gamma_i\}, & \text{if } H^k > 0, \\ 1, & \text{otherwise,} \end{cases} \tag{3.29}$$

$$l_i^{MALT} = \begin{cases} r \cdot \max\{H_i, \frac{\lambda_i + \gamma_i}{2}\}, & \text{if } H^k > 0, \\ 1, & \text{otherwise,} \end{cases} \tag{3.30}$$

where H_i is from (3.28), and λ_i and γ_i are calculated as follows

$$\lambda_i = \max\{H_{i-1}, H_i, H_{i+1}\},\ \ 2 \le i \le k, \tag{3.31}$$

$$\gamma_i = H^k \frac{(x_i - x_{i-1})}{X^{max}}, \tag{3.32}$$

with H^k from (3.27) and

$$X^{max} = \max\{x_i - x_{i-1} :\ 2 \le i \le k\}. \tag{3.33}$$

When $i = 2$ and $i = k$ only H_2, H_3, and H_{k-1}, H_k, should be considered, respectively, in (3.31).

After these preliminary descriptions we are ready to describe the General Scheme 2 (GS-2) of algorithms studied in this section.

Step 0. *Initialization.* Execute first two trials at the points a and b, i. e., $x^1 := a$, $z^1 := g(a)$ and $x^2 := b$, $z^2 := g(b)$. Set the iteration counter $k := 2$. Suppose that $k \geq 2$ iterations of the algorithm have already been executed. The iteration $k+1$ consists of the following steps.

Step 1. *Reordering.* Reorder the points $x^1, \ldots, x^k$ (and the corresponding function values $z^1, \ldots, z^k$) of previous trials by subscripts so that

$$a = x_1 < \ldots < x_k = b, \quad z_i = g(x_i),\ 1 \leq i \leq k.$$

Step 2. *Estimates of the Lipschitz constant.* Calculate the current estimates l_i of the Lipschitz constant for each subinterval $[x_{i-1}, x_i]$, $2 \leq i \leq k$, in one of the following ways.

Step 2.1. *A priori given estimate.* Take an a priori given estimate $\overline{L}$ of the Lipschitz constant for the whole interval $[a, b]$, i. e., set $l_i := \overline{L}$.

Step 2.2. *Global estimate.* Set $l_i := \overline{L}_k$, where $\overline{L}_k$ is from (3.26).

Step 2.3. *"Maximum" local tuning.* Set $l_i := l_i^{MLT}$, where l_i^{MLT} is from (3.29).

Step 2.4. *"Maximum-Additive" local tuning.* Set $l_i := l_i^{MALT}$, where l_i^{MALT} is from (3.30).

Step 3. *Calculation of characteristics.* Compute for each subinterval $[x_{i-1}, x_i]$, $2 \leq i \leq k$, its characteristic R_i by using one of the following rules.

Step 3.1. *Geometric methods.*

$$R_i = \frac{z_i + z_{i-1}}{2} - l_i \frac{x_i - x_{i-1}}{2}. \tag{3.34}$$

Step 3.2. *Information methods.*

$$R_i = 2(z_i + z_{i-1}) - l_i(x_i - x_{i-1}) - \frac{(z_i - z_{i-1})^2}{l_i(x_i - x_{i-1})}. \tag{3.35}$$

Step 4. *Interval selection.* Determine an interval $[x_{t-1}, x_t]$, $t = t(k)$, for performing the next trial as follows

$$t = \min \arg \min_{2 \leq i \leq k} R_i. \tag{3.36}$$

Step 5. *Stopping rule.* **If**

$$x_t - x_{t-1} \leq \varepsilon, \tag{3.37}$$

where $\varepsilon > 0$ is a given accuracy of the global search, **then Stop** and take as an estimate of the global minimum g^* the value $g_k^* = \min_{1 \leq i \leq k}\{z_i\}$ obtained at a point $x_k^* = \arg\min_{1 \leq i \leq k}\{z_i\}$.

Otherwise, go to Step 6.

Step 6. *New trial.* Execute the next trial $z^{k+1} := g(x^{k+1})$ at the point

$$x^{k+1} = \frac{x_t + x_{t-1}}{2} - \frac{z_t - z_{t-1}}{2l_t}. \tag{3.38}$$

Increase the iteration counter $k := k + 1$, and go to Step 1.

Let us study now the strong homogeneity of algorithms described above. This study is executed simultaneously in the traditional and in the Infinity Computing frameworks. In fact, so far, whether these methods were strongly homogeneous or not was an open problem even for finite constants α and β. Here, it is shown that methods belonging to GS enjoy the strong homogeneity property for finite, infinite, and infinitesimal scaling and shifting constants. Recall that all the values related to the function $g(x)$ are indicated by "$\widehat{\ }$" and the values related to the function $f(x)$ are written without "$\widehat{\ }$".

The following lemma shows how the adaptive estimates of the Lipschitz constant $\widehat{\overline{L}}_k$, $\widehat{l}_i^{MLT}$, and $\widehat{l}_i^{MALT}$ that can assume finite, infinite, and infinitesimal values are related to the respective original estimates $\overline{L}_k$, l_i^{MLT}, and l_i^{MALT} that can be finite only.

Lemma 3.2. *Let us consider the function $g(x) = \alpha f(x) + \beta$, where $f(x)$ assumes only finite values and has a finite Lipschitz constant L over the interval $[a, b]$ and α, $\alpha > 0$, and β can be finite, infinite and infinitesimal numbers. Then, the adaptive estimates $\widehat{\overline{L}}_k$, $\widehat{l}_i^{MLT}$ and $\widehat{l}_i^{MALT}$ from (3.26), (3.29) and (3.30) are equal to $\alpha\overline{L}_k$, αl_i^{MLT} and αl_i^{MALT}, respectively, if $H^k > 0$, and to 1, otherwise.*

Proof. It follows from (3.28) that

$$\widehat{H}_i = \frac{|\widehat{z}_i - \widehat{z}_{i-1}|}{x_i - x_{i-1}} = \frac{\alpha|z_i - z_{i-1}|}{x_i - x_{i-1}} = \alpha H_i. \tag{3.39}$$

If $H^k \neq 0$, then $H^k = \max\limits_{2 \leq i \leq k} \frac{|z_i - z_{i-1}|}{x_i - x_{i-1}}$ and $H^k \geq H_i$, $2 \leq i \leq k$. Thus, using (3.39) we obtain $\alpha H^k \geq \alpha H_i = \widehat{H}_i$, and, therefore, $\widehat{H}^k = \alpha H^k$ and

from (3.26) it follows $\widehat{\overline{L}}_k = \alpha \overline{L}_k$. On the other hand, if $H^k = 0$, then both estimates for the functions $g(x)$ and $f(x)$ are equal to 1 (see (3.26)).

The same reasoning can be used to show the respective results for the local tuning techniques MLT and MALT (see (3.29) and (3.30))

$$\widehat{\lambda}_i = \max\{\widehat{H}_{i-1}, \widehat{H}_i, \widehat{H}_{i+1}\} = \alpha \max\{H_{i-1}, H_i, H_{i+1}\},$$

$$\widehat{\gamma}_i = \widehat{H}^k \frac{x_i - x_{i-1}}{X^{max}} = \alpha H^k \frac{x_i - x_{i-1}}{X^{max}} = \alpha \gamma_i,$$

$$\widehat{l}_i^{MLT} = \begin{cases} r \cdot \max\{\widehat{\lambda}_i, \widehat{\gamma}_i\}, & \text{if } \widehat{H}^k > 0, \\ 1, & \text{otherwise.} \end{cases}$$

$$\widehat{l}_i^{MALT} = \begin{cases} r \cdot \max\{\widehat{H}_i, \frac{\widehat{\lambda}_i + \widehat{\gamma}_i}{2}\}, & \text{if } \widehat{H}^k > 0, \\ 1, & \text{otherwise.} \end{cases}$$

Therefore, we can conclude that

$$\widehat{l}_i^{\{MLT,MALT\}} = \begin{cases} \alpha l_i^{\{MLT,MALT\}}, & \text{if } H^k > 0, \\ 1, & \text{otherwise.} \end{cases}$$ □

Lemma 3.3. *Suppose that characteristics $\widehat{R}_i$, $2 \le i \le k$, for the scaled objective function $g(x)$ are equal to an affine transformation of the characteristics R_i calculated for the original objective function $f(x)$*

$$\widehat{R}_i = \widehat{\alpha}_k R_i + \widehat{\beta}_k, \quad 2 \le i \le k, \tag{3.40}$$

where scales $\widehat{\alpha}_k$, $\widehat{\alpha}_k > 0$, and $\widehat{\beta}_k$ can be finite, infinite, or infinitesimal and possibly different for different iterations k. Then, the same interval $[x_{t-1}, x_t]$, $t = t(k)$, from (3.36) is selected at each iteration for the next subdivision during optimizing $f(x)$ and $g(x)$, i.e., it follows $\widehat{t(k)} = t(k)$.

Proof. Since due to (3.36) $t = \arg\min_{2 \le i \le k} R_i$, then $R_t \le R_i$ and

$$\widehat{\alpha}_k R_t + \widehat{\beta}_k \le \widehat{\alpha}_k R_i + \widehat{\beta}_k, \quad 2 \le i \le k.$$

That, due to (3.40), can be re-written as

$$\widehat{R}_t = \min_{2 \le i \le k} \widehat{R}_i = \widehat{\alpha}_k R_t + \widehat{\beta}_k.$$

Notice that if there are several values j such that $R_j = R_t$, then (see (3.36)) we have $t < j, j \ne t$, i.e., even in this situation it follows $\widehat{t(k)} = t(k)$. This observation concludes the proof. □

The following Theorem shows that methods belonging to the GS-2 enjoy the strong homogeneity property.

Theorem 3.2. *Algorithms belonging to the GS-2 and applied for solving the problem (3.23) are strongly homogeneous for finite, infinite, and infinitesimal scales $\alpha > 0$ and β.*

Proof. An algorithm will generate the same sequences of trials optimizing two functions $f(x)$ and $g(x)$ if the following conditions hold:

(i) The same interval $[x_{t-1}, x_t]$, $t = t(k)$, from (3.36) is selected at each iteration for the next subdivision during optimizing functions $f(x)$ and $g(x)$, i.e., it follows $\widehat{t(k)} = t(k)$.

(ii) The next trial at the selected interval $[x_{t-1}, x_t]$ is performed at the same point during optimizing functions $f(x)$ and $g(x)$, i.e., in (3.38) it follows $\widehat{x}^{k+1} = x^{k+1}$.

In order to prove assertions (i) and (ii), let us consider computational steps of the GS-2. For both functions, $f(x)$ and $g(x)$, Steps 0 and 1 of the GS-2 work with the same interval $[a, b]$, do not depend on the objective function, and, as a result, do not influence (i) and (ii). Step 2 is a preparative one, it is responsible for estimating the Lipschitz constants for all the intervals $[x_{i-1}, x_i]$, $2 \le i \le k$ and was studied in Lemmas 1–2 above. Step 3 calculates characteristics of the intervals and, therefore, is directly related to the assertion (i). In order to prove it, we consider computations of characteristics $\widehat{R}_i$ for all possible cases of calculating estimates l_i during Step 2 and show that there always possible to indicate constants $\widehat{\alpha}_k$ and $\widehat{\beta}_k$ from Lemma 3.

Lemmas 1 and 2 show that for the a priori given finite Lipschitz constant L for the function $f(x)$ (see Step 2.1) it follows $\widehat{L} = \alpha L$. For the adaptive estimates of the Lipschitz constants for intervals $[x_{i-1}, x_i]$, $2 \le i \le k$, (see (3.26), (3.29), (3.30) and Steps 2.2 – 2.4 of the GS-2) we have $\widehat{l}_i = \alpha l_i$, if $H^k > 0$, and $\widehat{l}_i = l_i = 1$, otherwise (remind that the latter corresponds to the situation $z_i = z_1, 1 \le i \le k$). Since Step 3 includes substeps defining information and geometric methods, then the following four combinations of methods with Lipschitz constant estimates computed at one of the substeps of Step 2 can take place:

(a) The value $\widehat{l}_i = \alpha l_i$ and the geometric method is used. From (3.34) we obtain

$$\widehat{R}_i = \frac{\widehat{z}_{i-1} + \widehat{z}_i}{2} - \widehat{l}_i \frac{x_i - x_{i-1}}{2} = \alpha\left(\frac{z_{i-1} + z_i}{2} - l_i \frac{x_i - x_{i-1}}{2}\right) + \beta = \alpha R_i + \beta.$$

Thus, in this case we have $\widehat{\alpha}_k = \alpha$ and $\widehat{\beta}_k = \beta$.

(b) The value $\widehat{l}_i = \alpha l_i$ and the information method is used. From (3.35) we get

$$\widehat{R}_i = 2(\widehat{z}_i + \widehat{z}_{i-1}) - \widehat{l}_i(x_i - x_{i-1}) - \frac{(\widehat{z}_i - \widehat{z}_{i-1})^2}{\widehat{l}_i(x_i - x_{i-1})} =$$

$$2\alpha(z_i + z_{i-1}) + 4\beta - \alpha l_i(x_i - x_{i-1}) - \frac{\alpha^2(z_i - z_{i-1})^2}{\alpha l_i(x_i - x_{i-1})} = \alpha R_i + 4\beta.$$

Therefore, in this case it follows $\widehat{\alpha}_k = \alpha$ and $\widehat{\beta}_k = 4\beta$.

(c) The value $\widehat{l}_i = l_i = 1$ and the geometric method is considered. Since in this case $z_i = z_1, 1 \le i \le k$, then for the geometric method (see (3.34)) we have

$$\widehat{R}_i = \frac{\widehat{z}_{i-1} + \widehat{z}_i}{2} - \widehat{l}_i\frac{x_i - x_{i-1}}{2} = \widehat{z}_1 - \frac{x_i - x_{i-1}}{2} =$$

$$\alpha z_1 + \beta - \frac{x_i - x_{i-1}}{2} = R_i + \alpha z_1 - z_1 + \beta.$$

Thus, in this case we have $\widehat{\alpha}_k = 1$ and $\widehat{\beta}_k = z_1(\alpha - 1) + \beta$.

(d) The value $\widehat{l}_i = l_i = 1$ and the information method is used. Then, the characteristics (see (3.35)) are calculated as follows

$$\widehat{R}_i = 2(\widehat{z}_i + \widehat{z}_{i-1}) - \widehat{l}_i(x_i - x_{i-1}) - \frac{(\widehat{z}_i - \widehat{z}_{i-1})^2}{\widehat{l}_i(x_i - x_{i-1})} =$$

$$4\widehat{z}_1 - (x_i - x_{i-1}) = 4\alpha z_1 + 4\beta - (x_i - x_{i-1}) = R_i + 4\alpha z_1 - 4z_1 + 4\beta.$$

Therefore, in this case it follows $\widehat{\alpha}_k = 1$ and $\widehat{\beta}_k = 4(z_1(\alpha - 1) + \beta)$.

Let us show now that assertion (ii) also holds. Since for both the geometric and the information approaches the the same formula (3.38) for computing x^{k+1} is used, we should consider only two cases related to the estimates of the Lipschitz constant:

(a) If $\widehat{l}_t = \alpha l_t$, then it follows

$$\widehat{x}^{k+1} = \frac{x_t + x_{t-1}}{2} - \frac{\widehat{z}_t - \widehat{z}_{t-1}}{2\widehat{l}_t} = \frac{x_t + x_{t-1}}{2} - \frac{\alpha(z_t - z_{t-1})}{2\alpha l_t} = x^{k+1}.$$

(b) If $\widehat{l}_t = l_t = 1$, then $z_i = z_1,\ 1 \le i \le k$, and we have

$$\widehat{x}^{k+1} = \frac{x_t + x_{t-1}}{2} - \frac{\widehat{z}_t - \widehat{z}_{t-1}}{2\widehat{l}_t} = \frac{x_t + x_{t-1}}{2} = x^{k+1}.$$

This result concludes the proof. □

In order to illustrate the behavior of methods belonging to the GS-2 in the Infinity Computer framework, the following three algorithms being examples of concrete implementations of the GS-2 have been tested:

- **Geom-AL:** Geometric method with an a priori given overestimate of the Lipschitz constant. It is constructed by using Steps 2.1 and 3.1 in the GS-2.
- **Inf-GL:** Information method with the global estimate of the Lipschitz constant. It is formed by using Steps 2.2 and 3.2 in the GS-2.
- **Geom-LTM:** Geometric method with the "Maximum" local tuning. It is built by applying Steps 2.3 and 3.1 in the GS-2.

The algorithm Geom-AL has one parameter – an a priori given overestimate of the Lipschitz constant. In algorithms Geom-LTM and Inf-GL, the Lipschitz constant is estimated during the search and the reliability parameter r is used. In this work, the values of the Lipschitz constant of the functions $f(x)$ for the algorithm Geom-AL have been taken from [107] (and multiplied by α for the function $g(x)$). The values of the parameter r for the algorithms Geom-LTM and Inf-GL have been set to 1.1 and 1.5, respectively. The value $\varepsilon = 10^{-4}(b-a)$ has been used in the stopping criterion (3.37).

Recall that huge or very small scaling/shifting constants can provoke the ill-conditioning of the scaled function $g(x)$ in the traditional computational framework. In the Infinity Computing framework, the positional numeral system (3.3) allows us to avoid ill-conditioning and to work safely with infinite and infinitesimal scaling/shifting constants if the respective grossdigits and grosspowers are not too large or too small. In order to illustrate this fact the following two pairs of the values α and β have been used in our experiments: $(\alpha_1, \beta_1) = (①^{-1}, ①)$ and $(\alpha_2, \beta_2) = (①, ①^2)$. The corresponding grossdigits and grosspowers involved in their representation are, respectively: 1 and -1 for α_1; 1 and 1 for β_1; 1 and 1 for α_2; and 1 and 2 for β_2. It can be seen that all of these constants are numbers that do not provoke instability in numerical operations. Hereinafter scaled functions constructed using constants (α_1, β_1) are indicated as $g(x)$ and functions using (α_2, β_2) are designated as $h(x)$.

The algorithms Geom-AL, Inf-GL, and Geom-LTM have been tested on 20 global optimization problems from [83, 107] (see Appendix A) and on the respective scaled functions $g(x)$ and $h(x)$ constructed from them. It has been obtained that on all 20 test problems with infinite and infinitesimal constants (α_1, β_1) and (α_2, β_2) the results on the original functions $f(x)$ from [83, 107] and on scaled functions $g(x)$ and $h(x)$ coincide. To illustrate this fact, let

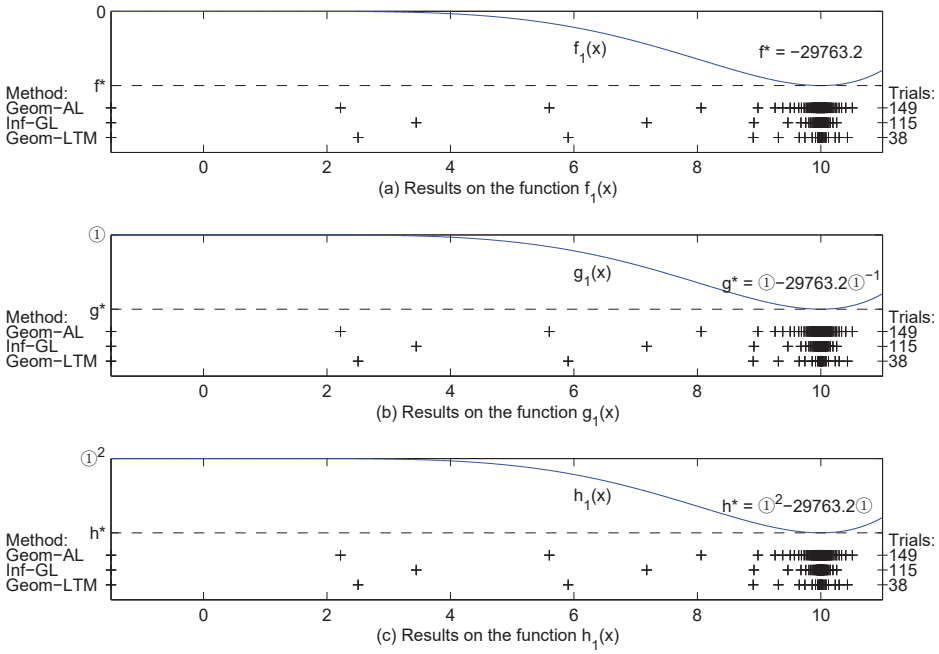


Figure 3.3: Results for (a) the original test function $f_1(x)$ from [83, 107], (b) the scaled test function $g_1(x) = ①^{-1}f_1(x) + ①$, (c) the scaled test function $h_1(x) = ①f_1(x) + ①^2$. Trials are indicated by the signs "+" under the graphs of the functions and the number of trials for each method is indicated on the right. The results coincide for each method on all three test functions.

us consider the first three problems from the set of 20 tests (see Fig. 3.3.a, Fig. 3.4.a, and Fig. 3.5.a). They are defined as follows

$$f_1(x) = \frac{1}{6}x^6 - \frac{52}{25}x^5 + \frac{39}{80}x^4 + \frac{71}{10}x^3 - \frac{79}{20}x^2 - x + \frac{1}{10},$$

$$f_2(x) = sin(x) + sin\frac{10x}{3},$$

$$f_3(x) = \sum_{k=1}^{5} -k \cdot sin[(k+1)x + k].$$

In Fig. 3.3.b, Fig. 3.4.b, and Fig. 3.5.b, the results for the scaled functions

$$g_i(x) = ①^{-1}f_i(x) + ①, \quad i = 1, 2, 3,$$

are presented and in Fig. 3.3.c, Fig. 3.4.c, and Fig. 3.5.c, the results for the scaled functions

$$h_i(x) = ①f_i(x) + ①^2, \quad i = 1, 2, 3,$$

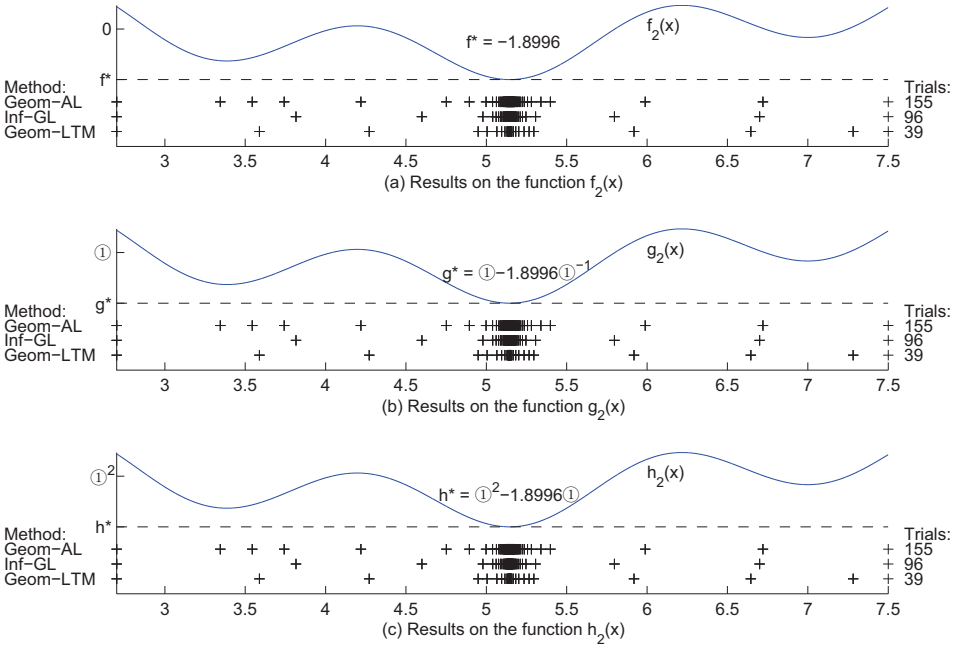


Figure 3.4: Results for (a) the original test function $f_2(x)$ from [83, 107], (b) the scaled test function $g_2(x) = ①^{-1}f_2(x) + ①$, (c) the scaled test function $h_2(x) = ①f_2(x) + ①^2$. Trials are indicated by the signs "+" under the graphs of the functions and the number of trials for each method is indicated on the right. The results coincide for each method on all three test functions.

are shown. It can be seen that the results coincide for all three methods on all three test functions $f_i(x), g_i(x)$, and $h_i(x), i = 1, 2, 3$. Analogous results hold for the remaining test problems from [83, 107].

In particular, it can be seen from these experiments that even if the scaling constants α and β have a different order (e.g., when α is infinitesimal and β is infinite) the scaled problems continue to be well-conditioned (cf. discussion on ill-conditioning in the traditional framework with finite scaling/shifting constants, see Fig. 3.2). This fact suggests that even if finite constants of significantly different orders are required, ① can also be used to avoid the ill-conditioning by substituting very small constants by $①^{-1}$ and very huge constants by ①. In this case, if, for instance, α is too small (as, e.g., in (3.22), $\alpha = 10^{-17}$) and β is too large (as, e.g., in (3.22), $\beta = 1 \gg 10^{-17}$), the values $\alpha_1 = ①^{-1}$ and $\beta_1 = ①$ can be used in computations instead of $\alpha = 10^{-17}$ and $\beta = 1$ avoiding so underflows and overflows. After the conclusion of the optimization process, the global minimum of the original function f^* can be

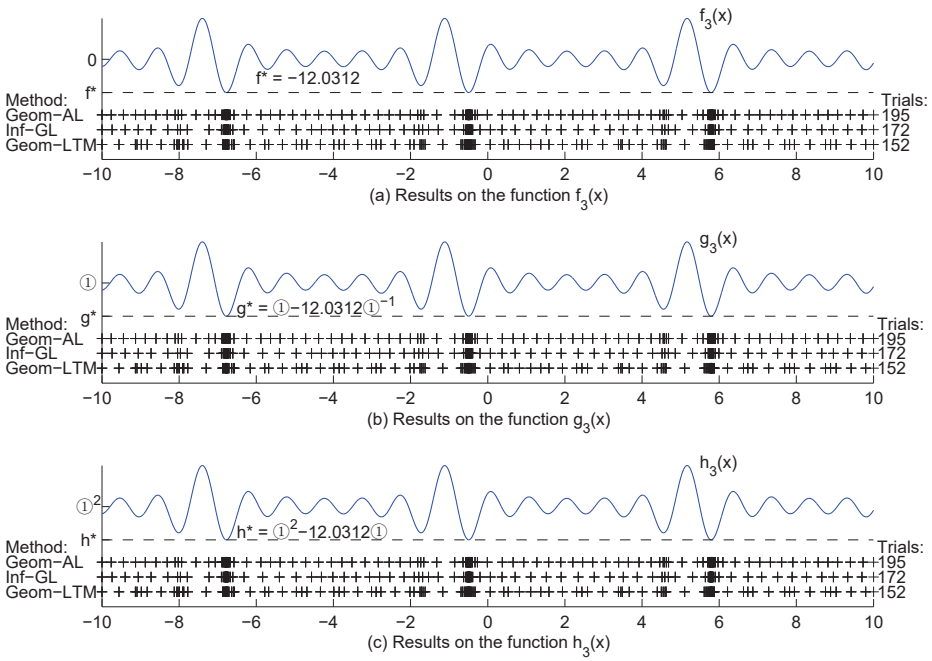


Figure 3.5: Results for (a) the original test function $f_3(x)$ from [83, 107], (b) the scaled test function $g_3(x) = ①^{-1} f_3(x) + ①$, (c) the scaled test function $h_3(x) = ① f_3(x) + ①^2$. The results coincide for each method on all three test functions. The number of trials for each method is indicated on the right.

easily extracted from the solution $g^* = \alpha_1 f^* + \beta_1 = ①^{-1} f^* + ①$ of the scaled problem using $①^{-1}$ and ① and the original finite constants α and β can be used to get the required value $g^* = \alpha f^* + \beta$ (in our case, $g^* = 10^{-17} f^* + 1$).

3.3 Numerical infinitesimals in convex non-smooth optimization

The objective of this Section is to evaluate the impact of the infinity computing paradigm on practical solution of multidimensional optimization problems. In particular, nonsmooth unconstrained optimization problems, where the objective function is assumed to be convex and not necessarily differentiable, are considered. For such family of problems, the occurrence of discontinuities in the derivatives may result in failures of the algorithms suited for smooth problems (see [64] for details).

A family of nonsmooth optimization methods based on a variable metric

approach are considered, and the infinity computing techniques are used for numerically dealing with some quantities which can assume values arbitrarily small or large, as a consequence of nonsmoothness. In particular, the case, treated in the literature, where the metric is defined via a diagonal matrix with positive entries, is considered.

The computational results of the implementation on a set of benchmark test-problems from scientific literature is provided. Obtained results show that the infinity computing can be successfully used in order to handle with ill-conditioning in this case, as well.

3.3.1 Variable metric method based on the limited-memory bundle approach

Nonsmooth (or nondifferentiable) optimization is about finding a local minimum of a real-valued function of several variables, that is solving the following unconstrained optimization problem

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}), \tag{3.41}$$

where $f : \mathbb{R}^n \mapsto \mathbb{R}$ is a function not necessarily differentiable.

Although nonsmoothness generally occurs at a zero-measure set of the function domain, convergence to a minimum is not ensured in case an algorithm designed for treating smooth problems is applied to nonsmooth ones.

For the above reasons, intensive research activities have been developed in last decades, and several proposals for dealing with nonsmoothness are offered in the literature. As for historical contributions, the books [43, 103, 209] and the seminal papers [29, 101] are mentioned here.

In this work, the unconstrained minimization of a convex function under no differentiability hypothesis is focused. In this area two research mainstreams are active: subgradient type methods (see the classic version in [209] and, among the others, the more recent variants in [11, 56, 142]) and bundle type methods. The latter stems from the seminal paper [116], and benefits from both the cutting plane model [29, 101] and the conjugate subgradient approach [221]. The term *bundle* is referred to a certain amount of information (values and subgradients of the objective functions at a set of points in its domain) which is accumulated and possibly updated as the algorithm proceeds. Bundle methods [86] require, at each iteration, the solution of a linear (or, more often, quadratic) program aimed at finding a tentative displacement from the current approximation of the minimizer. Due to nonsmoothness, even in case a line search is adopted, a sufficient decrease in the objective

function is not guaranteed and, consequently, this family of methods accommodates for the so called *null step*, a situation where no progress toward the minimum is achieved and, instead, some additional information about the local behavior of the objective function is accumulated into the bundle.

Literature on bundle methods is very rich (see, e.g., the contributions given in [42, 57, 58, 104, 127]). Some authors have designed methods that retain many features of the classic bundle approach, while trying to simplify the displacement finding phase, thus avoiding solution of a too burdensome subproblem at each iteration, see [81, 99]. In addition, such methods utilize some ideas coming from the vast literature of the variable metric approach to smooth optimization, as they embed variants of standard Quasi–Newton updating formulae for approximating the Hessian matrix (see [44] for a classic survey on Quasi–Newton methods and [12], with the references therein, for the applications of the variable metric approach to nonsmooth optimization).

A Quasi–Newton method for minimizing a differentiable function $f : \mathbb{R}^n \mapsto \mathbb{R}$ is an iterative method where, at any point $\mathbf{x}^k \in \mathbb{R}^n$, a matrix B^k is generated as an approximation of the Hessian matrix such that it satisfies the equation

$$B^k \mathbf{s}^k = \mathbf{u}^k, \tag{3.42}$$

where

$$\mathbf{s}^k \triangleq \mathbf{x}^k - \mathbf{x}^{k-1} \tag{3.43}$$

and

$$\mathbf{u}^k \triangleq \nabla f(\mathbf{x}^k) - \nabla f(\mathbf{x}^{k-1}). \tag{3.44}$$

Sometimes equation (3.42) is replaced by

$$H^k \mathbf{u}^k = \mathbf{s}^k, \tag{3.45}$$

where H^k is the inverse of B^k.

Methods extending the Quasi–Newton idea to the convex nondifferentiable case still require at each iteration satisfaction of equation (3.42), the only difference being in the definition of vector $\mathbf{u}^k$. In fact, denoting by $\partial f(\mathbf{x})$ the subdifferential and by $\mathbf{g} \in \partial f(\mathbf{x})$ any subgradient of f at $\mathbf{x}$, see [86], $\mathbf{u}^k$ can be restated as

$$\mathbf{u}^k \triangleq \mathbf{g}_k - \mathbf{g}_{k-1}, \tag{3.46}$$

with $\mathbf{g}_k$ and $\mathbf{g}_{k-1}$ being subgradients of f at the points $\mathbf{x}^k$ and $\mathbf{x}^{k-1}$, respectively.

However, in the case of nondifferentiable functions, the search for a matrix B^k satisfying (3.42) is a problem inherently ill-conditioned, due to possible

discontinuities in the first order derivatives, which in turn may result in "large" $\mathbf{u}^k$ corresponding to "small" $\mathbf{s}^k$.

This observation is the main motivation of this research, as we propose to tackle equation (3.42) by applying the Infinity Computing approach to handle infinite and infinitesimal numbers, and based on the *grossone* concept, a numeral resuming the properties of the entire set of natural number.

Let us remark that the objective of the research in this Section is not to introduce yet another method for solving convex nondifferentiable minimization problems, but to experiment how ideas coming from the grossone-based arithmetic can be cast into a traditional numerical optimization framework.

In order to evaluate the role and the impact of the infinity computing paradigm on the effectiveness of variable-metric methods applied to the nonsmooth problem (3.41), the Diagonal Bundle method (`D-Bundle`) introduced in [99] is studied. `D-Bundle` is based on the limited-memory bundle approach [81], where ideas coming from the variable-metric bundle approach [117, 127] are combined with the extension to nonsmooth problems of the limited-memory approach introduced in [24]. The main feature of the method is the use of the diagonal update formula of the variable-metric matrix introduced in [85]. Let us remark that `D-Bundle` is suited for dealing with nonconvexity as well as with nonsmoothness. A thorough description of the method can be found in [99] along with its convergence properties and several numerical results. Let us restrict our attention to the convex nonsmooth case, where most of the literature on nonsmooth optimization has concentrated in last decades. This allows us to adopt a simplified structure of `D-Bundle`, thus highlighting the specific impact of the infinity computing paradigm. The relevant details of our `Grossone-D-Bundle` algorithm are presented in the following.

Assume that at each point $\mathbf{x} \in \mathbb{R}^n$ it is possible to calculate $f(\mathbf{x})$ and a subgradient $\mathbf{g} \in \partial f(\mathbf{x})$. Letting $\mathbf{x}^k$ be the estimate of the minimum at iteration k, the search direction $\mathbf{d}^k$ adopted to locate the next iterate is defined as

$$\mathbf{d}^k = -H^k \boldsymbol{\xi}_a^k, \tag{3.47}$$

where $\boldsymbol{\xi}_a^k$ is the current aggregate subgradient (see the details below), and H^k is the inverse of B^k, the positive definite variable-metric $n \times n$ matrix, resembling the classic approximation of the Hessian matrix adopted in quasi-Newton methods for smooth optimization. Let us remark that in calculating $\mathbf{d}^k$, unlike standard bundle methods, no solution of any optimization subproblem is required.

Once the search direction $\mathbf{d}^k$ is available, a line search along $\mathbf{d}^k$ is per-

formed, which can return two possible outcomes: *serious* step, whenever a sufficient decrease with respect to the desirable one is achieved, and *null* step otherwise. In the former case, a new approximation $\mathbf{x}^{k+1}$ of a minimizer is obtained, while in the latter an auxiliary point $\mathbf{y}^{k+1}$ is gathered with an associate subgradient, to enrich the information about the local behavior of the function. In fact, the definition of the aggregate subgradient in formula (3.47) is different in the two cases. If a serious step has been performed (i.e., the new iterate $\mathbf{x}^{k+1}$ has been located) the aggregate subgradient $\boldsymbol{\xi}_a^{k+1}$ is any subgradient of f at $\mathbf{x}^{k+1}$. On the other hand, in the null-step case, no move from the current estimate of the minimizer is made (that is $\mathbf{x}^{k+1} = \mathbf{x}^k$), hence the aggregate subgradient $\boldsymbol{\xi}_a^{k+1}$ is obtained as a convex combination of the three vectors $\mathbf{g}^k \in \partial f(\mathbf{x}^k)$, $\mathbf{g}^{k+1} \in \partial f(\mathbf{y}^{k+1})$, and $\boldsymbol{\xi}_a^k$, with multipliers λ_1^*, λ_2^*, and λ_3^*, respectively, which minimize the following function

$$\phi(\lambda_1, \lambda_2, \lambda_3) \triangleq \frac{1}{2}\left(\lambda_1 \mathbf{g}^k + \lambda_2 \mathbf{g}^{k+1} + \lambda_3 \boldsymbol{\xi}_a^k\right)^\top H^k \left(\lambda_1 \mathbf{g}^k + \lambda_2 \mathbf{g}^{k+1} + \lambda_3 \boldsymbol{\xi}_a^k\right) +$$

$$+\lambda_2 \alpha^{k+1} + \lambda_3 \alpha_a^k, \tag{3.48}$$

where α^{k+1} is the standard (nonnegative) linearization error

$$\alpha^{k+1} \triangleq f(\mathbf{x}^k) - f(\mathbf{y}^{k+1}) - (\mathbf{g}^{k+1})^\top (\mathbf{x}^k - \mathbf{y}^{k+1}),$$

and α_a^k is the aggregated linearization error. The aggregated linearization error is updated by the following recursive equality

$$\alpha_a^{k+1} = \lambda_2^* \alpha^{k+1} + \lambda_3^* \alpha_a^k,$$

and it is initialized to zero every time a serious step takes place. We remark that minimization of function (3.48) can be easily obtained, see [127, Section 4].

3.3.2 Handling of ill-conditioning using infinitesimal thresholds and Grossone-D-Bundle method

Let us focus now on the updating technique of matrix B^k. As in [99] matrix B^k must be kept diagonal and positive definite. Let us first introduce a simplified version of the procedure indicated in [99]. In particular, at iteration k one can only store the information about the previous iterate, and calculate two correction vectors $\mathbf{s}^k$ and $\mathbf{u}^k$, according to the definitions (3.43) and (3.46), respectively. Hence, following [99], matrix B^k is obtained by solving

the following optimization problem

$$\begin{aligned} \min \quad & \|B\mathbf{s}^k - \mathbf{u}^k\| && (3.49)\\ \text{s.t.} \quad & B_{ii} \geq \epsilon, \qquad \forall i \in \{1, \ldots, n\}, && (3.50)\\ & B_{ij} = 0, \quad \forall i \neq j \in \{1, \ldots, n\}, && (3.51) \end{aligned}$$

for some $\epsilon > 0$, whose optimal solution can be expressed as

$$B_{ii}^k = \max\left(\epsilon, \frac{\mathbf{u}_i^k}{\mathbf{s}_i^k}\right), \;\; i = 1, \ldots, n. \tag{3.52}$$

It should be noted that the nonsmoothness of f is likely to cause ill-conditioning of problem (3.49)-(3.51), as the elements of matrix B^k may result arbitrarily large and, consequently, those of H^k arbitrarily small, since

$$H_{ii}^k = (B_{ii}^k)^{-1}, \;\; i = 1, \ldots, n. \tag{3.53}$$

This is where the infinity computing paradigm comes into play. In order to control ill-conditioning we replace first the correction vectors $\mathbf{s}^k$ and $\mathbf{u}^k$ with vectors $\boldsymbol{\delta}^k$ and $\boldsymbol{\gamma}^k$, respectively, whose components are defined as follows

$$\boldsymbol{\delta}_i^k = \begin{cases} \mathbf{s}_i^k, & \text{if } |\mathbf{s}_i^k| > \epsilon, \\ ①^{-1}, & \text{otherwise}, \end{cases} \tag{3.54}$$

and

$$\boldsymbol{\gamma}_i^k = \begin{cases} \mathbf{u}_i^k, & \text{if } |\mathbf{u}_i^k| > \epsilon, \\ ①^{-1}, & \text{otherwise}. \end{cases} \tag{3.55}$$

Then, the ratio $\dfrac{\boldsymbol{\gamma}_i^k}{\boldsymbol{\delta}_i^k}$ can be corrected by introducing the following

$$\mathbf{b}_i^k = \begin{cases} ①^{-1}, & \text{if } 0 < \dfrac{\boldsymbol{\gamma}_i^k}{\boldsymbol{\delta}_i^k} \leq \epsilon \\ \dfrac{\boldsymbol{\gamma}_i^k}{\boldsymbol{\delta}_i^k}, & \text{otherwise}. \end{cases} \tag{3.56}$$

Finally, by analogy with the solution of problem (3.49)-(3.51), the elements of the diagonal matrix B^k can be set according to the rule

$$B_{ii}^k = \max\left(①^{-1}, \mathbf{b}_i^k\right), \;\; i = 1, \ldots, n. \tag{3.57}$$

Note that matrix B^k may contain infinite and infinitesimal numbers. However, since calculation of $\mathbf{d}^k$ according to formula (3.47) has to take place in

$$B_\epsilon = \begin{pmatrix} 1.0 \times 10^{-3} & 0 & 0 \\ 0 & 2.0 \times 10^{7} & 0 \\ 0 & 0 & 1.0 \times 10^{-1} \end{pmatrix} \qquad B_{①} = \begin{pmatrix} 1.0 & 0 & 0 \\ 0 & 2.0 \times 10^{1}① & 0 \\ 0 & 0 & 1.0 \end{pmatrix}$$

$$H_\epsilon = \begin{pmatrix} 1.0 \times 10^{3} & 0 & 0 \\ 0 & 5.0 \times 10^{-8} & 0 \\ 0 & 0 & 1.0 \times 10^{1} \end{pmatrix} \qquad H_{①} = \begin{pmatrix} 1.0 & 0 & 0 \\ 0 & 5.0 \times 10^{-2} & 0 \\ 0 & 0 & 1.0 \end{pmatrix}$$

Figure 3.6: Variable-metric matrices with $\epsilon = 10^{-3}$

a classical arithmetic framework, we need to get rid of the dependence on ① and $①^{-1}$ in the definition of matrix H^k. Therefore the following scheme is chosen which, as far as finite numbers are involved, reflects the standard inverse calculation scheme:

$$H^k_{ii} = \begin{cases} (B^k_{ii})^{-1} & \text{if } B^k_{ii} \text{ neither depends on ① nor on } ①^{-1}, \\ (B^k_{ii})^{-1} \cdot ① & \text{if } B^k_{ii} \text{ is of the type } \alpha①,\ \alpha \in \mathbb{R} \\ (B^k_{ii})^{-1} \cdot ①^{-1} & \text{if } B^k_{ii} \text{ is of the type } \alpha①^{-1},\ \alpha \in \mathbb{R} \end{cases} \tag{3.58}$$

A better understanding of the consequences of (3.58) can be gained by examining the following example.

Example 3.3. *Let the vectors*

$$\mathbf{s}^\top = \begin{pmatrix} 1.0 \times 10^{-4} & 1.0 \times 10^{-6} & 1.0 \times 10^{-4} \end{pmatrix}$$

and

$$\mathbf{u}^\top = \begin{pmatrix} -1.0 \times 10^{-4} & 2.0 \times 10^{1} & 1.0 \times 10^{-5} \end{pmatrix}$$

be given, where we have dropped the superscript k for notational simplicity. For each value of $\epsilon \in \{10^{-3}, 10^{-5}, 10^{-8}\}$*, the matrices* B_ϵ *and* H_ϵ *can be calculated according to (3.52) and (3.53), and the matrices* $B_{①}$ *and* $H_{①}$ *– according to (3.57) and (3.58), and the results are reported in Figures 3.6, 3.7, and 3.8, respectively. The effect of adopting the rules (3.54)–(3.58) can be seen by comparing* $H_{①}$ *against* H_ϵ*, since* $H_{①}$ *"stabilizes" as we take smaller and smaller values of* ϵ*. In fact, one can easily observe that taking smaller values than* 10^{-8} *for* ϵ*, then the ill-conditioning of* B_ϵ *gets worse and worse, while* $H_{①}$ *remains unchanged, see Figure 3.8.*

Let us introduce now the formal statement of the `Grossone-D-Bundle` method, reported in Algorithm 1. The following parameters need to be set: the sufficient decrease parameter $m \in (0, 1)$, the matrix updating threshold $\epsilon > 0$, the stepsize reduction parameter $\sigma \in (0, 1)$, the stopping parameter

$$B_\epsilon = \begin{pmatrix} 1.0 \times 10^{-5} & 0 & 0 \\ 0 & 2.0 \times 10^{7} & 0 \\ 0 & 0 & 1.0 \times 10^{-1} \end{pmatrix} \quad B_{①} = \begin{pmatrix} ①^{-1} & 0 & 0 \\ 0 & 2.0 \times 10^{1}① & 0 \\ 0 & 0 & ①^{-1} \end{pmatrix}$$

$$H_\epsilon = \begin{pmatrix} 1.0 \times 10^{5} & 0 & 0 \\ 0 & 5.0 \times 10^{-8} & 0 \\ 0 & 0 & 1.0 \times 10^{1} \end{pmatrix} \quad H_{①} = \begin{pmatrix} 1.0 & 0 & 0 \\ 0 & 5.0 \times 10^{-2} & 0 \\ 0 & 0 & 1.0 \end{pmatrix}$$

Figure 3.7: Variable-metric matrices with $\epsilon = 10^{-5}$

$$B_\epsilon = \begin{pmatrix} 1.0 \times 10^{-8} & 0 & 0 \\ 0 & 2.0 \times 10^{7} & 0 \\ 0 & 0 & 1.0 \times 10^{-1} \end{pmatrix} \quad B_{①} = \begin{pmatrix} ①^{-1} & 0 & 0 \\ 0 & 2.0 \times 10^{7} & 0 \\ 0 & 0 & 1.0 \times 10^{-1} \end{pmatrix}$$

$$H_\epsilon = \begin{pmatrix} 1.0 \times 10^{8} & 0 & 0 \\ 0 & 5.0 \times 10^{-8} & 0 \\ 0 & 0 & 1.0 \times 10^{1} \end{pmatrix} \quad H_{①} = \begin{pmatrix} 1.0 & 0 & 0 \\ 0 & 5.0 \times 10^{-8} & 0 \\ 0 & 0 & 1.0 \times 10^{1} \end{pmatrix}$$

Figure 3.8: Variable-metric matrices with $\epsilon = 10^{-8}$

$\eta > 0$, and the null step parameter $\theta > 0$. Few comments on the algorithm mechanism are in order. At Step 4, a search direction $\mathbf{d}^k$ at the current point $\mathbf{x}^k$ is selected according to (3.47). Next, at Step 5, the desirable reduction w^k of the objective function is calculated, and the algorithm stops at Step 6 if w^k is very close to zero. Otherwise, a line-search procedure along $\mathbf{d}^k$ is started at Step 9, whose termination depends on fulfillment of the sufficient decrease condition at Step 10. In such a case, a serious step takes place along with the *grossone*-based update of matrix H^k. If no sufficient decrease is attained at Step 10, then the line-search is iterated at Step 21, unless the step size has become very small, in that case a null step occurs, which amounts to updating the aggregate gradient $\boldsymbol{\xi}_a^k$ and the linearization error α_a^k, but not the matrix H^k. For further details, especially regarding the definition of the desirable descent w^k, and the calculation of $\boldsymbol{\xi}_a^k$ and α_a^k in the null-step case, we refer the reader to [99].

Convergence properties of the proposed algorithm are consequence of those described in [99], where it is required that no update of matrix H_k takes place in case of null step (this is what actually occurs in our implementation too), and that matrix H_k stays bounded. In our case, boundedness can be easily checked by enumerating all possible cases in the construction of matrix H_k, taking into account, in particular, the definition (3.56).

Numerical experiments have been executed on three different classes of large-scale test problems taken from [12] and reported in Table 3.1.

Algorithm 1 `Grossone-D-Bundle`

Input: a starting point $\mathbf{x}^0 \in \mathbb{R}^n$, parameters $\theta > 0$, $\eta > 0$, $\epsilon > 0$, $m \in (0,1)$, $\sigma \in (0,1)$

Output: an approximate local minimizer $\mathbf{x}^* \in \mathbb{R}^n$

1: Calculate $\mathbf{g}^0 \in \partial f(\mathbf{x}^0)$, and set $\boldsymbol{\xi}_a^0 = \mathbf{g}^0$ ▷ Initialization
2: Set $H^0 = I$ (i.e., the $n \times n$ identity matrix) ▷ |
3: Set $\alpha^0 = \alpha_a^0 = 0$, and $k = 0$ ▷ |
4: Set $\mathbf{d}^k = -H^k \boldsymbol{\xi}_a^k$ ▷ Find a search direction at $\mathbf{x}^k$
5: Set $w^k = (\boldsymbol{\xi}_a^k)^\top \mathbf{d}^k - 2\alpha_a^k$ ▷ Set the desirable reduction
6: **if** $w^k \geq -\eta$ **then** ▷ Stopping test
7: set $\mathbf{x}^* = \mathbf{x}^k$ and **exit** ▷ Return $\mathbf{x}^*$ as an approximate minimizer
8: **end if**
9: Set $t_1 = 1$ and $s = 1$ ▷ Start the line-search
10: **if** $f(\mathbf{x}^k + t_s \mathbf{d}^k) \leq f(\mathbf{x}^k) + m t_s w^k$ **then** ▷ Descent test
11: Set $\mathbf{x}^{k+1} = \mathbf{x}^k + t_s \mathbf{d}^k$ ▷ Make a serious step
12: Calculate $\mathbf{g}^{k+1} \in \partial f(\mathbf{x}^{k+1})$ ▷ |
13: Set $\boldsymbol{\xi}_a^{k+1} = \mathbf{g}^{k+1}$ ▷ |
14: Calculate H^{k+1} according to (3.58) ▷ Grossone matrix update
15: Set $k = k + 1$ and **go to 4**
16: **end if**
17: **if** $t_s \leq \theta$ **then** ▷ Closeness test
18: Calculate $\mathbf{g}_+ \in \partial f(\mathbf{x}^k + t_s \mathbf{d}^k)$ and $\boldsymbol{\xi}_a \in \mathrm{conv}\{\mathbf{g}^k, \boldsymbol{\xi}_a^k, \mathbf{g}_+\}$ ▷ Make a null step
19: Set $\alpha_+ = f(\mathbf{x}^k) - f(\mathbf{x}^k + t_s \mathbf{d}^k) + t_s g_+^\top \mathbf{d}^k$ ▷ |
20: Calculate $\alpha_a \in \mathrm{conv}\{0, \alpha_a^k, \alpha_+\}$ ▷ |
21: Update $\boldsymbol{\xi}_a^k = \boldsymbol{\xi}_a$, $\alpha_a^k = \alpha_a$, and **go to 4** ▷ |
22: **else**
23: Set $t_{s+1} = \sigma t_s$, $s = s + 1$ and **go to 10** ▷ Iterate the line-search
24: **end if**

We have adopted different values of the space dimension n, running our experiments for $n = 50,\ 100,\ 200$. The algorithm has been coded in C++, compiled with Microsoft Visual Studio 2010 Professional on a HP-15-ba090ur machine, under the MS Windows 10 operating system. We report the results obtained by stopping the algorithm after a given number N_f of function evaluations, where $N_f \in \{50, 100, 200, 300, 400, 500\}$. In particular, the objective function value f^*, the relative error $e_r = \frac{|f^* - f(x^*)|}{1+|f(x^*)|}$, the number of serious steps N_S, and the number of H^k updates that involved the use of grossone $N_{①}$ are reported. The algorithm for several values of ϵ is tested and, for simplicity of presentation, the results obtained adopting the two extreme values $\epsilon = 10^{-2}$, see Tables 3.2 to 3.4, and $\epsilon = 10^{-10}$, see Tables 3.5 to 3.7, are only reported. The remaining parameters of the algorithms have been set as follows: $\sigma = 0.7$, $m = 0.1$, $\eta = 10^{-10}$ and $\theta = 10^{-4}$.

The results demonstrate, as could be expected that large values of ϵ, the threshold for switching to the use of grossone, imply an increased number of grossone-based steps, with corresponding lack of accuracy. In fact, it can be seen that the ratio of grossone-based steps over the total number of serious steps is smaller as ϵ decreases. Summing up, the use of grossone may allow reasonable treatment of ill-conditioning provided that the threshold ϵ is sufficiently small.

The proposed `Grossone-D-Bundle` approach has been also numerically compared against its standard counterpart, namely, Algorithm 1 where at Step 14 formula (3.53) for calculating H_k replaces formula (3.58).

The comparison is made for different values of ϵ in (3.52) and by stopping the algorithm after a given number N_f of function evaluations.

In Table 3.8, the results obtained by focusing on problems with size $n = 100$, and setting $\epsilon \in \{10^{-2}, 10^{-5}, 10^{-10}\}$ and $N_f \in \{50, 100, 200\}$ are reported. In particular, the relative errors $e_r^{\text{alg}} = \frac{|f^* - f(x^*)|}{1+|f(x^*)|}$, where alg $= D$ and alg $= G$ refer, respectively, to $B^k = B_{①}$ and $B^k = B_\epsilon$, are presented. It is worth noting that the results provided by the `Grossone-D-Bundle` approach are most of the times better than the standard approach, this outcome being more evident for smaller values of ϵ.

Chained LQ

$$f(\mathbf{x}) = \sum_{i=1}^{n-1} \max\left\{-x_i - x_{i+1}, -x_i - x_{i+1} + (x_i^2 + x_{i+1}^2 - 1)\right\}$$

$\mathbf{x}_i^0 = -0.5$, for all $i = 1, \ldots, n$

$\mathbf{x}_i^* = 1/\sqrt{2}$, for all $i = 1, \ldots, n$

$f(\mathbf{x}^*) = -(n-1)\sqrt{2}$

Chained CB3 I

$$f(\mathbf{x}) = \sum_{i=1}^{n-1} \max\left\{x_i^4 + x_{i+1}^2, (2 - x_i)^2 + (2 - x_{i+1})^2, 2e^{-x_i + x_{i+1}}\right\}$$

$\mathbf{x}_i^0 = 2$, for all $i = 1, \ldots, n$

$\mathbf{x}_i^* = 1$, for all $i = 1, \ldots, n$

$f(\mathbf{x}^*) = 2(n-1)$

Chained CB3 II

$$f(\mathbf{x}) = \max\left\{\sum_{i=1}^{n-1}(x_i^4 + x_{i+1}^2), \sum_{i=1}^{n-1}\left((2 - x_i)^2 + (2 - x_{i+1})^2\right), \sum_{i=1}^{n-1}(2e^{-x_i + x_{i+1}})\right\}$$

$\mathbf{x}_i^0 = 2$, for all $i = 1, \ldots, n$

$\mathbf{x}_i^* = 1$, for all $i = 1, \ldots, n$

$f(\mathbf{x}^*) = 2(n-1)$

Table 3.1: Test problems ($\mathbf{x}^0$ is the starting point, $\mathbf{x}^*$ is the minimizer)

	$n = 50$				$n = 100$				$n = 200$			
N_f	f^*	e_r	N_S	$N_{①}$	f^*	e_r	N_S	$N_{①}$	f^*	e_r	N_S	$N_{①}$
50	-66.7329331	3.65E-02	8	5	-135.7207701	3.04E-02	8	6	-276.9103190	1.60E-02	8	6
100	-67.1750430	3.02E-02	11	8	-135.9990819	2.84E-02	10	8	-277.4285342	1.42E-02	10	8
200	-68.1049532	1.69E-02	16	13	-136.1352791	2.75E-02	12	10	-277.4421523	1.41E-02	12	10
300	-68.4659193	1.18E-02	21	18	-136.8200990	2.26E-02	15	13	-277.5469159	1.37E-02	14	12
400	-68.4721827	1.17E-02	22	19	-138.7923614	8.62E-03	20	18	-277.5469159	1.37E-02	14	12
500	-68.4809312	1.16E-02	24	21	-139.3581970	4.60E-03	24	22	-277.5469159	1.37E-02	14	12

Table 3.2: Results on the Chained LQ test problem with dimensions $n = 50, 100, 200$ and $\epsilon = 10^{-2}$.

	$n = 50$				$n = 100$				$n = 200$			
N_f	f^*	e_r	N_S	$N_{①}$	f^*	e_r	N_S	$N_{①}$	f^*	e_r	N_S	$N_{①}$
50	142.0278332	4.45E-01	7	5	276.0654964	3.92E-01	8	5	513.8191333	2.90E-01	7	5
100	118.9363346	2.11E-01	10	8	223.3313657	1.27E-01	11	8	411.7056803	3.44E-02	11	9
200	113.6568233	1.58E-01	16	14	218.4138518	1.03E-01	16	13	403.6445105	1.41E-02	16	14
300	104.2485194	6.31E-02	20	18	218.0286772	1.01E-01	20	17	400.5369972	6.36E-03	20	18
400	102.7413588	4.79E-02	25	23	200.6511044	1.33E-02	27	24	398.4287379	1.07E-03	24	22
500	99.2882901	1.30E-02	30	28	199.0866922	5.46E-03	31	28	398.1830522	4.59E-04	27	25

Table 3.3: Results on the Chained CB3 I test problem with dimensions $n = 50, 100, 200$ and $\epsilon = 10^{-2}$.

	$n = 50$				$n = 100$				$n = 200$			
N_f	f^*	e_r	N_S	$N_{①}$	f^*	e_r	N_S	$N_{①}$	f^*	e_r	N_S	$N_{①}$
50	125.5297664	2.78E-01	7	4	227.4325378	1.48E-01	7	4	458.0950749	1.51E-01	8	5
100	109.5757036	1.17E-01	10	7	203.0976658	2.56E-02	11	8	428.6046364	7.67E-02	10	7
200	105.7547185	7.83E-02	13	10	202.4957777	2.26E-02	12	9	424.6396931	6.68E-02	12	9
300	101.5635593	3.60E-02	17	14	202.4957777	2.26E-02	12	9	424.6396931	6.68E-02	12	9
400	100.9229376	2.95E-02	22	19	202.4957777	2.26E-02	12	9	423.8254683	6.47E-02	14	11
500	100.6930428	2.72E-02	24	21	202.4957777	2.26E-02	12	9	409.6297910	2.91E-02	19	16

Table 3.4: Results on the Chained CB3 II test problem with dimensions $n = 50, 100, 200$ and $\epsilon = 10^{-2}$.

	$n = 50$				$n = 100$				$n = 200$			
N_f	f^*	e_r	N_S	$N_{①}$	f^*	e_r	N_S	$N_{①}$	f^*	e_r	N_S	$N_{①}$
50	-69.0981172	2.82E-03	10	4	-139.4816288	3.73E-03	9	4	-280.4361446	3.51E-03	9	4
100	-69.1800716	1.66E-03	13	7	-139.7674121	1.70E-03	12	6	-280.8968967	1.88E-03	11	5
200	-69.1801058	1.66E-03	14	8	-139.7711371	1.67E-03	13	7	-280.8968967	1.88E-03	11	5
300	-69.1801058	1.66E-03	14	8	-139.7711371	1.67E-03	13	7	-280.8968967	1.88E-03	11	5
400	-69.1801058	1.66E-03	14	8	-139.7711371	1.67E-03	13	7	-280.8968967	1.88E-03	11	5
500	-69.1801058	1.66E-03	14	8	-139.7711371	1.67E-03	13	7	-280.8968967	1.88E-03	11	5

Table 3.5: Results on the Chained LQ test problem with dimensions $n = 50, 100, 200$ and $\epsilon = 10^{-10}$.

	$n = 50$				$n = 100$				$n = 200$			
N_f	f^*	e_r	N_S	$N_{①}$	f^*	e_r	N_S	$N_{①}$	f^*	e_r	N_S	$N_{①}$
50	100.6694475	2.70E-02	8	1	199.9535378	9.82E-03	9	3	400.6092141	6.54E-03	8	1
100	98.2901649	2.93E-03	13	3	199.9446641	9.77E-03	9	3	398.8440820	2.12E-03	10	3
200	98.2261786	2.28E-03	17	4	199.2083426	6.07E-03	14	8	398.4688649	1.18E-03	13	5
300	98.2022949	2.04E-03	18	5	198.5907641	2.97E-03	19	11	398.4288154	1.07E-03	17	8
400	98.2022949	2.04E-03	18	5	198.5907641	2.97E-03	19	11	398.2626085	6.58E-04	20	11
500	98.2022949	2.04E-03	18	5	198.5907641	2.97E-03	19	11	398.2282233	5.72E-04	23	14

Table 3.6: Results on the Chained CB3 I test problem with dimensions $n = 50, 100, 200$ and $\epsilon = 10^{-10}$.

	$n = 50$				$n = 100$				$n = 200$			
N_f	f^*	e_r	N_S	$N_{①}$	f^*	e_r	N_S	$N_{①}$	f^*	e_r	N_S	$N_{①}$
50	111.4655678	1.36E-01	9	5	210.9095945	6.49E-02	9	5	440.0572743	1.05E-01	9	5
100	106.0858985	8.17E-02	11	7	201.9139633	1.97E-02	14	10	404.7495782	1.69E-02	12	8
200	106.0858985	8.17E-02	11	7	201.3641400	1.69E-02	18	14	404.7495782	1.69E-02	12	8
300	106.0858985	8.17E-02	11	7	201.3641400	1.69E-02	18	14	404.7495782	1.69E-02	12	8
400	106.0858985	8.17E-02	11	7	201.3641400	1.69E-02	18	14	404.7495782	1.69E-02	12	8
500	106.0858985	8.17E-02	11	7	201.3641400	1.69E-02	18	14	404.7495782	1.69E-02	12	8

Table 3.7: Results on the Chained CB3 II test problem with dimensions $n = 50, 100, 200$ and $\epsilon = 10^{-10}$.

	N_f	$\epsilon = 10^{-2}$		$\epsilon = 10^{-5}$		$\epsilon = 10^{-10}$	
		e_r^G	e_r^D	e_r^G	e_r^D	e_r^G	e_r^D
Chained LQ	50	3.04E-02	4.75E-01	7.64E-03	7.54E-01	3.73E-03	7.54E-01
	100	2.84E-02	2.13E-01	7.64E-03	7.49E-01	1.70E-03	7.54E-01
	200	2.75E-02	5.73E-02	7.64E-03	6.88E-01	1.67E-03	7.54E-01
Chained CB3-I	50	3.92E-01	7.43E-02	7.86E-03	1.05E-02	9.82E-03	1.05E-02
	100	1.27E-01	7.43E-02	5.76E-03	1.05E-02	9.77E-03	1.05E-02
	200	1.03E-01	1.36E-02	5.51E-04	1.05E-02	6.07E-03	1.05E-02
Chained CB3-II	50	1.48E-01	1.16E+00	6.47E-02	3.31E+00	6.49E-02	3.31E+00
	100	2.56E-02	5.47E-01	5.97E-02	1.42E+00	1.97E-02	2.98E+00
	200	2.26E-02	3.49E-01	5.97E-02	2.30E-01	1.69E-02	2.98E+00

Table 3.8: Comparisons on Chained LQ, Chained CB3 I, and Chained CB3 II, with size $n = 100$.

Chapter 4

Infinity Computing and Ordinary Differential Equations

New algorithms for the numerical solution to Ordinary Differential Equations (ODEs) with initial conditions are proposed in this Chapter. The algorithms are designed for work on the Infinity Computer. Due to this fact, the Infinity Computer allows one to calculate the exact derivatives of functions using infinitesimal values of the stepsize. As a consequence, the new methods described here are able to work with the exact values of the derivatives, instead of their approximations.

First, a well-known drawback of algorithms based on Taylor series formulae is considered. Several variants of a one-step multi-point explicit method closely related to the classical Taylor formula are considered. Theoretical convergence properties of the proposed methods are studied. Experimental comparison on a set of benchmark test problems from a literature with the well-known Runge-Kutta and Taylor methods is provided.

Then, implicit Obrechkoff methods (see for example [217, 218] and the references therein) are studied, as well. New Taylor-Obrechkoff methods are proposed. To get numerical evidence of the proposed algorithms, a few test problems are solved by means of the new methods and the obtained results are presented.

4.1 Generalized Taylor-based methods in standard and infinity floating-point arithmetic

The Taylor series method is one of the earliest algorithms to approximate the solution of initial value problems

$$\begin{cases} y' = f(t, y), & t \in [t_0, T], \\ y(t_0) = y_0, \end{cases} \tag{4.1}$$

where $f : [t_0, T] \times \mathbb{R}^n \to \mathbb{R}^n$ is assumed sufficiently differentiable.

Newton and Euler describe this approach in their seminal works of the 18th century. Since then, many authors mention Taylor series methods and some codes have been developed for both ODEs and Differential Algebraic Equations (DAEs): within the recent literature, we mention [14, 15, 16, 95, 220, 226].

A well-known drawback of the algorithms based on Taylor series formulae is that the explicit calculation of higher order derivatives formally is an over-elaborate task, especially when the dimension n of the system is not small. To avoid the analytical computation of the successive partial derivatives involved in the truncated Taylor expansion of f, a numerical differentiation approach has been often considered (see, for example, [135, 136]). A further interest in Taylor series methods stemmed from considering automatic rather than numerical differentiation, which makes use of specific tools based on the involved elementary functions (see [163]) and allows for a speed up of the overall computation.

Two instances for which the use of Taylor series methods has proved to be a powerful tool are reported:

- The analysis of the stability properties of equilibria and periodic orbits of dynamical systems often requires an accurate integration of the variational equations. Within this context, high-order Taylor series methods have been successfully exploited to correctly reproduce the highly oscillatory behavior of their solutions, avoiding extremely small stepsizes during the integration procedure. In most cases the variational equations are slight modifications of the original ones, so that it is possible to formulate the integration algorithm for both systems with little added effort.

- In some physical problems [5, 6] it is important to approximate the solution with a very high precision, as in the determination of normal

forms of differential systems, initial conditions for periodic problems, numerical detection of periodic orbits, computation of physical constants, etc. The Taylor method, just by increasing the degree of the formulae, permits high-precision integration, provided a multi-precision library is also used.

A recent alternative to numeric and automatic differentiation is based on the calculation of higher derivatives by using the Infinity Computer for performing numerical computations with infinite and infinitesimal quantities. The possibility to work with numerical infinitesimals allows one both to calculate the exact values of the derivatives numerically without finding the respective derivatives analytically and to work with infinitesimal stepsizes (see Section 3.1 for details). The first attempts to use the Infinity Computer in this direction have been done in [177, 204, 181].

The main idea of the usage of numerical infinitesimals and a core method for solving the IVP (4.1) on the Infinity Computer has been proposed in [181].

Since we are able to compute values of k derivatives of the function $y(x)$ at a generic point x_0 (see Section 3.1), we can estimate $y(x)$ in a neighborhood of the point x_0 by its Taylor expansion

$$y(x) \approx \hat{y}(x) = y_0 + \sum_{i=1}^{k} \frac{y^{(i)}(x_0)}{i!}(x - x_0)^i. \tag{4.2}$$

In cases where the radius of convergence of the Taylor expansion $\hat{y}(x)$ from (4.2) covers the whole interval $[a, b]$ of our interest, then there is no necessity to execute several steps with a finite value of h. In fact, thanks to the exact derivatives calculated numerically on the Infinity Computer (see Section 3.1) the function $y(x)$ can be approximated in the neighborhood of the initial point x_0 by its Taylor expansion $\hat{y}(x)$ with an order k depending on the desired accuracy and then $\hat{y}(x)$ can be evaluated at any point $x \in [a, b]$. This method is called *Taylor for the Infinity Computer (TIC)* hereinafter. This method does not assume the execution of several iterations with the step h.

Table 4.1 presents results of numerical experiments executed on a class of 12 test functions taken from the literature and described in the Appendix B. The method TIC is compared over the interval $[0, 0.2]$ with the Runge-Kutta method of the fourth order (RK4) with the integration step $h = 0.04$, i.e., to obtain an approximation at the point $x = 0.2$ the method RK4 executes 5 steps and 20 evaluations of the function $f(x, y)$ from (4.1). After the results for RK4 had been obtained, the TIC method was applied to each of 12

	y_RK4	ε_RK4	y_TIC	ε_TIC	N_TIC
1	0.837462	-8.62538e-009	0.837462	-5.91687e-009	6
2	1.242806	8.11157e-009	1.242806	4.19151e-009	6
3	1.221403	4.12685e-009	1.221403	2.13248e-009	6
4	1.221403	3.89834e-008	1.221403	2.13248e-009	6
5	1.491817	1.27726e-007	1.491817	1.13693e-008	7
6	0.135416	-5.96529e-004	0.135379	-3.24420e-004	10
7	36.154673	-8.16405e-005	36.149608	5.84540e-005	9
8	35.968459	-8.18293e-005	35.963409	5.85817e-005	9
9	1.239230	-5.78803e-009	1.239230	-4.08211e-009	10
10	0.781397	-1.76949e-009	0.781397	7.94128e-011	7
11	1.153846	8.98577e-009	1.153846	4.09600e-009	11
12	0.472441	2.95775e-010	0.472441	-1.60782e-010	10

Table 4.1: Results of a comparison of the Taylor for the Infinity Computer method with the Runge-Kutta method of the fourth order that executes 20 evaluations of $f(x, y)$ to reach the accuracy ε_RK4

problems. The method TIC stopped when the accuracy ε_TIC at the point $x = 0.2$ was better than the accuracy ε_RK4 of the method RK4. The last column, N_TIC, in Table 4.1 presents the number of evaluations of $f(x, y)$ executed by the TIC to reach the accuracy ε_TIC. In other words, it shows the number of infinitesimal steps executed by the TIC that is equal to the number of exact derivatives calculated by this method. The respective solutions y_RK4 and y_TIC are also shown in the table. For the considered problem the TIC method executes fewer evaluations of $f(x, y)$, in comparison with the Runge-Kutta method. The Taylor method with automatic differentiation, provides, if applicable, the same solution given by TIC, we observe, however that, if the evaluation of $f(x, y)$ involves k elementary functions (*; =; ln; exp; sin; cos; ...) then the computational complexity of the evaluation of the first $n - 1$ derivatives is $kn^2 + O(n)$. For the TIC the computational cost is exactly n, considering that the arithmetic operation are executed on the Infinity Computer Arithmetic.

Suppose now that the interval $[a, b]$ of our interest is wider than the radius of convergence of the Taylor expansion, or that the rate of convergence is so slow as to make the method unsuitable for the problem at hand. Then finite values of the integration step h should be used together with infinitesimal ones. So, we consider a mesh of $n + 1$ points x_i where

$$x_0 = a,\ x_{i+1} = x_i + h,\ 0 \leq i \leq n - 1,\ x_n = b,$$

where h is a finite integration step.

	y_RK4	ε_RK4	y_1.0	ε_1.0	N_1.0
1	0.735759	-2.20568e-008	0.735759	-1.51306e-008	30
2	3.436564	3.26429e-008	3.436564	1.68677e-008	30
3	2.718282	2.06343e-008	2.718282	1.06624e-008	30
4	2.718281	3.02546e-007	2.718282	1.65499e-008	30
5	7.388579	6.38533e-007	7.388584	5.66017e-008	35
6	0.000046	-2.98620e-003	0.000045	-1.62315e-003	50
7	20.026862	-1.22480e-006	20.026819	8.76400e-007	45
8	18.474354	-1.34674e-006	18.474311	9.47222e-007	45
9	2.732051	-7.46806e-009	2.732051	-8.00658e-010	50
10	-0.301169	6.85909e-008	-0.301169	-3.02846e-010	35
11	1.000000	3.82195e-008	1.000000	1.37934e-009	55
12	0.571429	7.69103e-009	0.571429	-2.01651e-011	50

Table 4.2: Results of a comparison of the Method 1.0 with the Runge-Kutta method of the fourth order that executes 100 evaluations of $f(x, y)$ to reach the accuracy ε_RK4

for $i = 0; i < n; i = i + 1$
$\quad y(x, x_i) = y_i + \sum_{j=1}^{k} \frac{y_i^{(j)}}{j!}(x - x_i)^j,$
$\quad y_{i+1} = y(x_{i+1}, x_i)$
endfor

Figure 4.1: Method 1.0 from [181]

Let us consider the Method 1.0 from [181] being our core algorithm that is used hereinafter for further developments. At each iteration it calculates k derivatives (in [181] the particular case $k = 2$ has been considered) at a point x_i using infinitesimal steps and then executes a finite step of the length $h = (b - a)/n$ to the point x_{i+1}. So, at each iteration it applies the TIC method. The Method 1.0 uses the initial values $x_0 = a$, $y_0 = y(x_0)$, from (4.1) and its detailed description is presented in Figure 4.1.

Table 4.2 presents results for the Method 1.0 extending numerical experiments described in Table 4.1 from the interval $[0, 0.2]$ to the interval $[0, 1]$. Thus, the Method 1.0 executed five finite steps during which the method TIC was applied five times at the points $x_i = a + ih$, where $a = 0,\ h = 0.2$. The Method 1.0 is compared with the method RK4 with the same integration step as before ($h = 0.04$). At each interval $[x_i, x_{i+1}]$ the method TIC executed N_TIC evaluations of the function $f(x, y)$, where N_TIC was taken from sixth column of the Table 4.1 for each test function. It can be seen from Table 4.2 that the accuracy of the Method 1.0 is better than the accuracy of the Runge-Kutta method of the fourth order and the Method 1.0 executes

	Method 1.0		Method RK2		Method 1.2	
#	y_n	ε_n	y_n	ε_n	y_n	ε_n
1	0.74148	-7.77538e-003	0.74148	-7.77538e-003	0.73262	4.26152e-003
2	3.40542	9.06351e-003	3.40542	9.06351e-003	3.42709	2.75755e-003
3	2.70271	5.72923e-003	2.70271	5.72923e-003	2.71354	1.74310e-003
4	2.69451	8.74561e-003	2.65824	2.20893e-002	2.70459	5.03795e-003
5	7.10043	3.89998e-002	7.10041	3.90024e-002	7.24952	1.88217e-002
6	1.00000	-2.20255e+004	1.00000	-2.20255e+004	-2.33333	5.13961e+004
7	31.63147	-5.79454e-001	31.63147	-5.79454e-001	-55.88025	3.79027e+000
8	30.04452	-6.26285e-001	30.05380	-6.26787e-001	-57.20706	4.09657e+000
9	2.74018	-2.97481e-003	2.73309	-3.80229e-004	2.72931	1.00341e-003
10	-0.30737	-2.05896e-002	-0.29889	7.56958e-003	-0.29849	8.89270e-003
11	0.99078	9.21515e-003	0.99824	1.76122e-003	1.00396	-3.96140e-003
12	0.57150	-1.33171e-004	0.57099	7.59569e-004	0.57166	-4.02684e-004

Table 4.3: For all the methods taken into consideration resulting values y_n at the point $x = 1$ and the respective relative error $\varepsilon_n = \frac{y(1)-y_n}{y(1)}$ are reported, where $y(1)$ is the exact solution

fewer evaluations of $f(x, y)$ in comparison with the RK4 method.

Let us describe now a new algorithm called *Method 1.2* hereinafter. It is a generalization of the Method 1.1 from [181]. The main idea is to use derivatives calculated at the point x_{i+1} in order to return to the point x_i and to construct at this point a correction leading to a new approximation y^c_{i+1} that is better than the original value y_{i+1} provided by the Method 1.0.

The Method 1.2 works as follows. First, initial values are chosen in the same way as in the Method 1.0 and values $y_i,\ i = 1, ..., n$, and functions $y(x, x_i)$ are calculated as in the Method 1.0. Then for each $i = 1, ..., n-1$, the backward functions $\overline{y}_i(x)$ using forward differences from (3.12) are computed as follows

$$\overline{y}_i(x) = y(x_i, x_{i-1}) + \sum_{j=1}^{k} \frac{y^{(j)}(x_i, x_i)}{j!}(x - x_i)^j. \tag{4.3}$$

Note that for $i = n$ the backward differences and the points $x_n - ①^{-1}, x_n - 2①^{-1}, ..., x_n - k①^{-1}$ should be used (see Corollary 1 in [181]) to calculate the derivatives $y^{(j)}(x_n, x_n)$. After that, the function $r_i(x)$ is defined as follows:

$$\begin{aligned} r_i(x) &= y_{i-1} + [p_0 y_{i-1} - (1 - p_0)\overline{y}_i(x_{i-1})] + \\ &+ \textstyle\sum_{j=1}^{k} \frac{1}{j!}[p_j y^{(j)}(x_{i-1}, x_{i-1}) + (1 - p_j)\overline{y}_i^{(j)}(x_{i-1})](x - x_{i-1})^j, \end{aligned} \tag{4.4}$$

where the weights $p_j \in [0, 1],\ j = 0, ..., k$, are parameters of the Method 1.2. So, the global correction c_i can be obtained following the rule

$$c_i = c(x_i) = c(x_{i-1}) + r_i(x_i) - y(x_i, x_{i-1}), \quad i = 1, ..., n, \tag{4.5}$$

with $c_0 = 0$. As a result, the desired corrected value y_i^c can be computed

$$y_n^c = y(x_n, x_{n-1}) + c(x_n). \tag{4.6}$$

The choice of the parameters of the Method 1.2 can be done using different criteria. For instance, the simplest choice for $k = 2$, $p_0 = p_1 = p_2 = 0.5$ gives us the Method 1.1 from [181]. If we apply the method to the standard test equation $y' = \lambda y$ we could choose the parameters by imposing that the method should become equivalent to the Taylor series method of the highest possible order. For example, for $k = 2$, it is shown in the next Section that the choice of parameters $p_0 = 0, p_1 = 5/6, p_2 = 0.5$ makes the method equivalent to the Taylor series method with $k = 4$. This means that for the test equations the order increases from two to four. Results of numerical experiments executed with this choice of the parameters, compared with the method 1.0 used with $k = 2$ and with the Runge-Kutta method of second order on the same class of test functions from the Appendix B are shown in Table 4.3.

As can be seen from Table 4.3, the introduced correction has allowed us to improve the results on some problems with respect to the Method 1.0. Again, as it was with the Method 1.0, among the Runge-Kutta methods a natural competitor for the Method 1.2 with $k = 2$ is the Runge-Kutta method of the second order since both methods execute $f(x, y)$ two times at each iteration. Then, the behavior of the Method 1.2 is comparable with that of the Runge-Kutta method of the second order on the considered test problems.

Another possible way to approximate the solution to the problem (4.1) is introduced in the Method 1.3 described below. The main two differences between the Method 1.2 and the Method 1.3 are the following:

i.) The Method 1.2 executes n iterations of the Method 1.0 to calculate the approximated values of y_i, $i = 1, ..., n$, and then these values are corrected by the backward function $\overline{y}_i(x)$ from (4.3) and the mixed function $r_i(x)$ from (4.4). The Method 1.3 at each subinterval $[x_{i-1}, x_i]$ executes the evaluation of the approximated value y_i and then immediately evaluates the backward function $\widetilde{y}_i(x)$ and the mixed function $\widetilde{r}_i(x)$ before moving to the next interval $[x_i, x_{i+1}]$.

ii.) The Method 1.2 in the formula (4.3) of $\overline{y}_i(x)$ uses for the forward function $y(x, x_i)$ at each point $(x_i, y(x_i, x_{i-1}))$ old values of derivatives calculated at points (x_i, y_i) before the correction. However, the values y_i and $y(x_i, x_{i-1})$ can be different and, as a consequence, the respective derivatives can be also different. Thus, the Method 1.3 after the correction and before moving to the next interval $[x_i, x_{i+1}]$ calculates the exact derivatives at each point $(x_i, y(x_i, x_{i-1}))$.

for $i = 1; i \leq n; i = i + 1$

$y(x, x_{i-1}) = y_{i-1} + \sum_{j=1}^{k} \frac{y_{i-1}^{(j)}}{j!}(x - x_{i-1})^j,$

$\widetilde{y}_i(x) = y(x_i, x_{i-1}) + \sum_{j=1}^{k} \frac{\tilde{y}_i^{(j)}}{j!}(x - x_i)^j,$

$\widetilde{r}_i(x) = y_{i-1} + [p_0 y_{i-1} - (1 - p_0)\widetilde{y}_i(x_{i-1})] +$

$+ \sum_{j=1}^{k} \frac{1}{j!}[p_j y^{(j)}(x_{i-1}, x_{i-1}) + (1 - p_j)\widetilde{y}_i^{(j)}(x_{i-1})](x - x_{i-1})^j,$

$y_i = \widetilde{r}_i(x_i)$

endfor

Figure 4.2: Method 1.3

	Method 1.3		Method RK2		Method RK3		Method RK4	
#	y_n	ε_n	y_n	ε_n	y_n	ε_n	y_n	ε_n
1	0.73577	-1.57578e-005	0.74148	-7.77538e-003	0.73547	3.91315e-004	0.73577	-1.57578e-005
2	3.43650	1.78619e-005	3.40542	9.06351e-003	3.43502	4.49549e-004	3.43650	1.78619e-005
3	2.71825	1.12909e-005	2.70271	5.72923e-003	2.71751	2.84169e-004	2.71825	1.12909e-005
4	2.71718	4.03706e-004	2.65824	2.20893e-002	2.71351	1.75595e-003	2.71787	1.52387e-004
5	7.38632	3.06560e-004	7.10041	3.90024e-002	7.35996	3.87457e-003	7.38632	3.06113e-004
6	0.00412	-8.96439e+001	1.00000	-2.20255e+004	-0.00412	9.16439e+001	0.00412	-8.96439e+001
7	20.11564	-4.43440e-003	31.63147	-5.79454e-001	20.00000	1.34005e-003	20.11564	-4.43440e-003
8	18.56287	-4.79261e-003	30.05380	-6.26787e-001	18.44666	1.49775e-003	18.56337	-4.81998e-003
9	2.73185	7.36503e-005	2.73309	-3.80229e-004	2.73178	9.74546e-005	2.73207	-5.35061e-006
10	-0.30091	8.73137e-004	-0.29889	7.56958e-003	-0.30105	3.94699e-004	-0.30116	4.13631e-005
11	1.00100	-1.00013e-003	0.99824	1.76122e-003	1.00093	-9.33299e-004	0.99997	3.18508e-005
12	0.57176	-5.73749e-004	0.57099	7.59569e-004	0.57164	-3.64397e-004	0.57143	5.73049e-006

Table 4.4: For all the methods taken into consideration resulting values y_n at the point $x = 1$ and the respective relative error $\varepsilon_n = \frac{y(1)-y_n}{y(1)}$ are reported, where $y(1)$ is the exact solution. The Method 1.3 uses the following parameters: $p_0 = 0$, $p_1 = 5/6, p_2 = 0.5$

Let us denote now as $y_{i-1}^{(j)}$ the approximation of the j-th derivative using the rule (3.12) calculated at the point (x_{i-1}, y_{i-1}) and as $\tilde{y}_i^{(j)}$ the approximation of the j-th derivative again using the rule (3.12) but at the point $(x_i, y(x_i, x_{i-1}))$. Notice that for $i = 1, ..., n-1$, the forward differences are used and the backward ones (see Corollary 1 from [181]) are applied for $i = n$. Taking into consideration that the initial values are the same as in the previous cases and the values $p_j, j = 0, ..., k$, are parameters of the Method 1.3 having the same meaning as in the Method 1.2, the Method 1.3 is described in Figure 4.2.

Results of numerical experiments for the Method 1.3 with the value $k = 2$ and the same parameters used in the Method 1.2 are given in Table 4.4 for the same test problems. As can be seen from this table, the attained accuracy of the Method 1.3 is better with respect to the Runge-Kutta method of second order for all test problems. This is due to the choice of the parameters, that

	Method 1.4		Method RK2		Method RK3		Method RK4	
#	y_n	ε_n	y_n	ε_n	y_n	ε_n	y_n	ε_n
1	0.73495	1.09797e-003	0.74148	-7.77538e-003	0.73547	3.91315e-004	0.73577	-1.57578e-005
2	3.43265	1.13895e-003	3.40542	9.06351e-003	3.43502	4.49549e-004	3.43650	1.78619e-005
3	2.71632	7.19955e-004	2.70271	5.72923e-003	2.71751	2.84169e-004	2.71825	1.12909e-005
4	2.71142	2.52314e-003	2.65824	2.20893e-002	2.71351	1.75595e-003	2.71787	1.52387e-004
5	7.32003	9.27820e-003	7.10041	3.90024e-002	7.35996	3.87457e-003	7.38632	3.06113e-004
6	0.03704	-8.14795e+002	1.00000	-2.20255e+004	-0.00412	9.16439e+001	0.00412	-8.96439e+001
7	23.46140	-1.71498e-001	31.63147	-5.79454e-001	20.00000	1.34005e-003	20.11564	-4.43440e-003
8	21.89863	-1.85355e-001	30.05380	-6.26787e-001	18.44666	1.49775e-003	18.56337	-4.81998e-003
9	2.73104	3.68986e-004	2.73309	-3.80229e-004	2.73178	9.74546e-005	2.73207	-5.35061e-006
10	-0.30030	2.87314e-003	-0.29889	7.56958e-003	-0.30105	3.94699e-004	-0.30116	4.13631e-005
11	1.00311	-3.10616e-003	0.99824	1.76122e-003	1.00093	-9.33299e-004	0.99997	3.18508e-005
12	0.57188	-7.83660e-004	0.57099	7.59569e-004	0.57164	-3.64397e-004	0.57143	5.73049e-006

Table 4.5: For all the methods taken into consideration resulting values y_n at the point $x = 1$ and the respective relative error $\varepsilon_n = \frac{y(1)-y_n}{y(1)}$ are reported, where $y(1)$ is the exact solution. The Method 1.4 uses the following parameters: $p_0 = 0$, $p_1 = 5/6, p_2 = 0.5$

	Method 1.0		Method 1.2		Method 1.3		Method 1.4	
#	y_n	ε_n	y_n	ε_n	y_n	ε_n	y_n	ε_n
1	0.74148	-7.77538e-003	0.73262	4.26152e-003	0.73577	-1.57578e-005	0.73495	1.09797e-003
2	3.40542	9.06351e-003	3.42709	2.75755e-003	3.43650	1.78619e-005	3.43265	1.13895e-003
3	2.70271	5.72923e-003	2.71354	1.74310e-003	2.71825	1.12909e-005	2.71632	7.19955e-004
4	2.69451	8.74561e-003	2.70459	5.03795e-003	2.71718	4.03706e-004	2.71142	2.52314e-003
5	7.10043	3.89998e-002	7.24952	1.88217e-002	7.38632	3.06560e-004	7.32003	9.27820e-003
6	1.00000	-2.20255e+004	-2.33333	5.13961e+004	0.00412	-8.96439e+001	0.03704	-8.14795e+002
7	31.63147	-5.79454e-001	-55.88025	3.79027e+000	20.11564	-4.43440e-003	23.46140	-1.71498e-001
8	30.04452	-6.26285e-001	-57.20706	4.09657e+000	18.56287	-4.79261e-003	21.89863	-1.85355e-001
9	2.74018	-2.97481e-003	2.72931	1.00341e-003	2.73185	7.36503e-005	2.73104	3.68986e-004
10	-0.30737	-2.05896e-002	-0.29849	8.89270e-003	-0.30091	8.73137e-004	-0.30030	2.87314e-003
11	0.99078	9.21515e-003	1.00396	-3.96140e-003	1.00100	-1.00013e-003	1.00311	-3.10616e-003
12	0.57150	-1.33171e-004	0.57166	-4.02684e-004	0.57176	-5.73749e-004	0.57188	-7.83660e-004

Table 4.6: Comparison of the Methods 1.0–1.4

make the methods of order at least three. A formal discussion related to the convergence properties of this method is presented in [206], where it has been proved that the order of convergence of this method is 3. The behavior of the Method 1.3 is comparable with respect to the Runge-Kutta method of third order on all the test problems. Finally, we observe that similar results are obtained with respect to the Runge-Kutta method of fourth order for linear problems. It should be mentioned that the obtained improvement has its price. In fact, the Method 1.3 executes $2kn$ evaluations of the function $f(x, y)$ from (4.1) whereas the Method 1.2 performs just $kn + k$ evaluations of $f(x, y)$.

The main idea of this method is to avoid calculation of the derivatives at the close points (x_i, y_i) and $(x_i, y(x_i, x_{i-1}))$. Since these points are close, the difference between y_i and $y(x_i, x_{i-1})$ can be relatively small. Thus, instead of

for $i = 1; i \le n; i = i + 1$
$\widehat{y}_i(x) = y(x_i, x_{i-1}) + \sum_{j=1}^{k} \frac{\widehat{y}_i^{(j)}}{j!}(x - x_i)^j,$
$r_i(x) = y_{i-1} + [p_0 y_{i-1} - (1 - p_0)\widehat{y}_i(x_{i-1})] +$
$+ \sum_{j=1}^{k} \frac{1}{j!}[p_j y^{(j)}(x_{i-1}, x_{i-1}) + (1 - p_j)\widehat{y}_i^{(j)}(x_{i-1})](x - x_{i-1})^j,$
$y_i = r_i(x_i)$
$y(x, x_i) = y_i + \sum_{j=1}^{k} \frac{\widehat{y}_i^{(j)}}{j!}x^j,$
endfor

Figure 4.3: Method 1.4

recalculating derivatives at the points (x_i, y_i) as it is done in the Method 1.3, the values of the derivatives calculated at the points $(x_i, y(x_i, x_{i-1}))$ can be used also at the points (x_i, y_i). This is the main difference between the Methods 1.3 and 1.4.

Let us again denote the approximation of the j-th derivative using infinitesimals starting from the point (x_i, y_i) as $y_i^{(j)}$ and the approximation of the j-th derivative using infinitesimals but starting from the point $(x_i, y(x_i, x_{i-1}))$ as $\widehat{y}_i^{(j)}$ for $i = 1, ..., n-1$, (for $i = n$ the backward approximation, see Corollary 1 from [181], is used). The initial values are the same as above and $p_j, j = 0, ..., k$ are parameters of the method. Then the Method 1.4 is described in Figure 4.3.

The results of the experiments on the same class of test functions are given in Table 4.5. For the Method 1.4 the value $k = 2$ and the same optimal parameters used for the Method 1.3.

As can be seen from Table 4.5, the attained accuracy of the Method 1.4 is better than the accuracy of the Runge-Kutta method of the second order for many test problems. The Method 1.4 executes less evaluations of the function $f(x, y)$ than the Method 1.3. Namely, it works doing the same number of evaluations of $f(x, y)$ as the Method 1.2, i.e., $kn + k$. The accuracy of the Method 1.4 is worse with respect to the Runge-Kutta methods of higher orders in all test problems. The reason is that the Method 1.4 does not use the exact derivatives at points (x_i, y_i) as the Method 1.3 does and the difference between the derivatives at the points (x_i, y_i) and $(x_i, y(x_i, x_{i-1}))$ causes errors. However, we can observe that the Method 1.4 executes the number of evaluations of the function $f(x, y)$ that is similar to the Runge-Kutta method of the second order and at the same time the accuracy of the Method 1.4 is better with respect to RK2.

Previously, we have already seen the nice performance of the Method 1.0 with large values of k. Let us now analyze this method with k up to

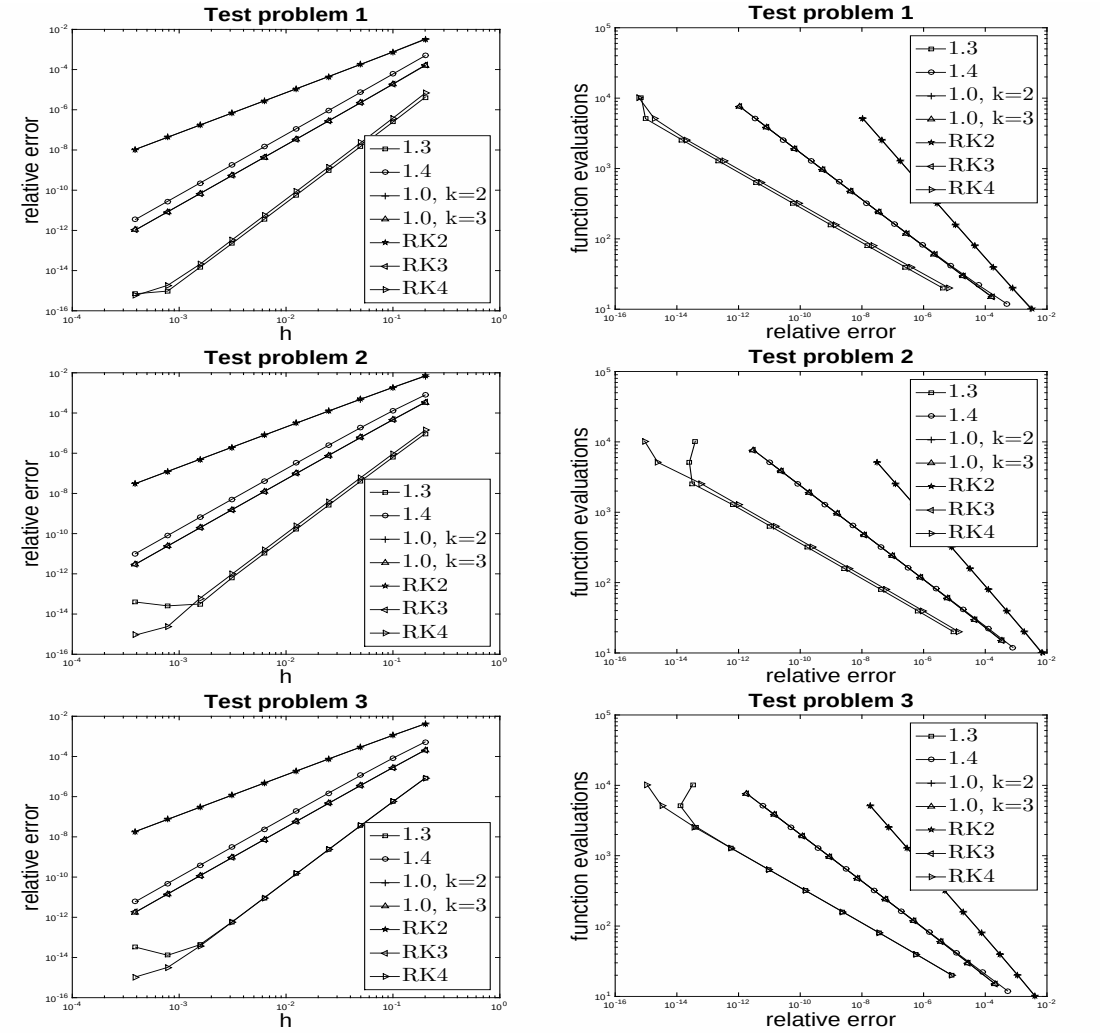


Figure 4.4: Relative error versus stepsize and function evaluation versus relative error for problems 1,2 and 3.

three in order to compare its behavior with the Methods 1.3 and 1.4 and the Runge-Kutta Methods of order 2 and 3.

Figures 4.4–4.7 show the behavior of the error for all the twelve considered test problems, changing the stepsize and the computational cost using the number of function evaluations (observe that the latter are performed in the Infinity Computer Arithmetic). From the pictures it could be seen that the behavior of the Method 1.3 is similar to the RK4 for linear problems, while for nonlinear ones, the order 3 of the method is experienced and the behavior is very close to the one of the Method 1.4 and of the Method 1.0 with $k = 3$. The Method 1.4 has order 3 for all the test problems and

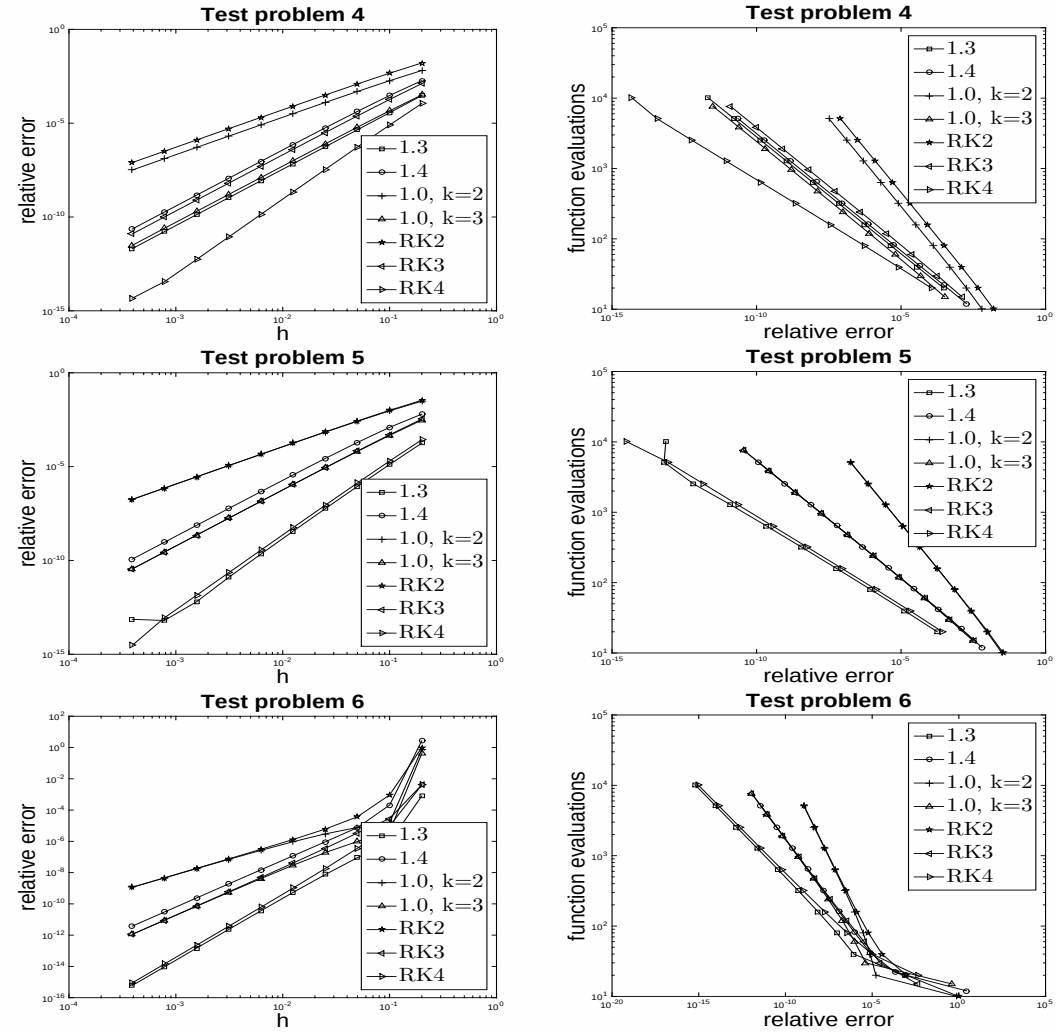


Figure 4.5: Relative error versus stepsize and function evaluation versus relative error for problems 4,5 and 6.

for the nonlinear tests requires a smaller computational effort to reach the same precision of the Method 1.3. The main potentiality of the numerical schemes using derivatives is that the use of the Infinity Computer allows one to compute the derivatives without error by a linear combinations of the computed infinitesimal values. This means that computing the exact derivatives does not give any computational problem to the method. We are aware that the Infinity Computer Arithmetic requires a computational effort that is higher than the one required by standard one, but for a computer based on this arithmetic all the complexity effort is hidden to the user, who needs only to use, in the arithmetic operations, the new numeral ①.

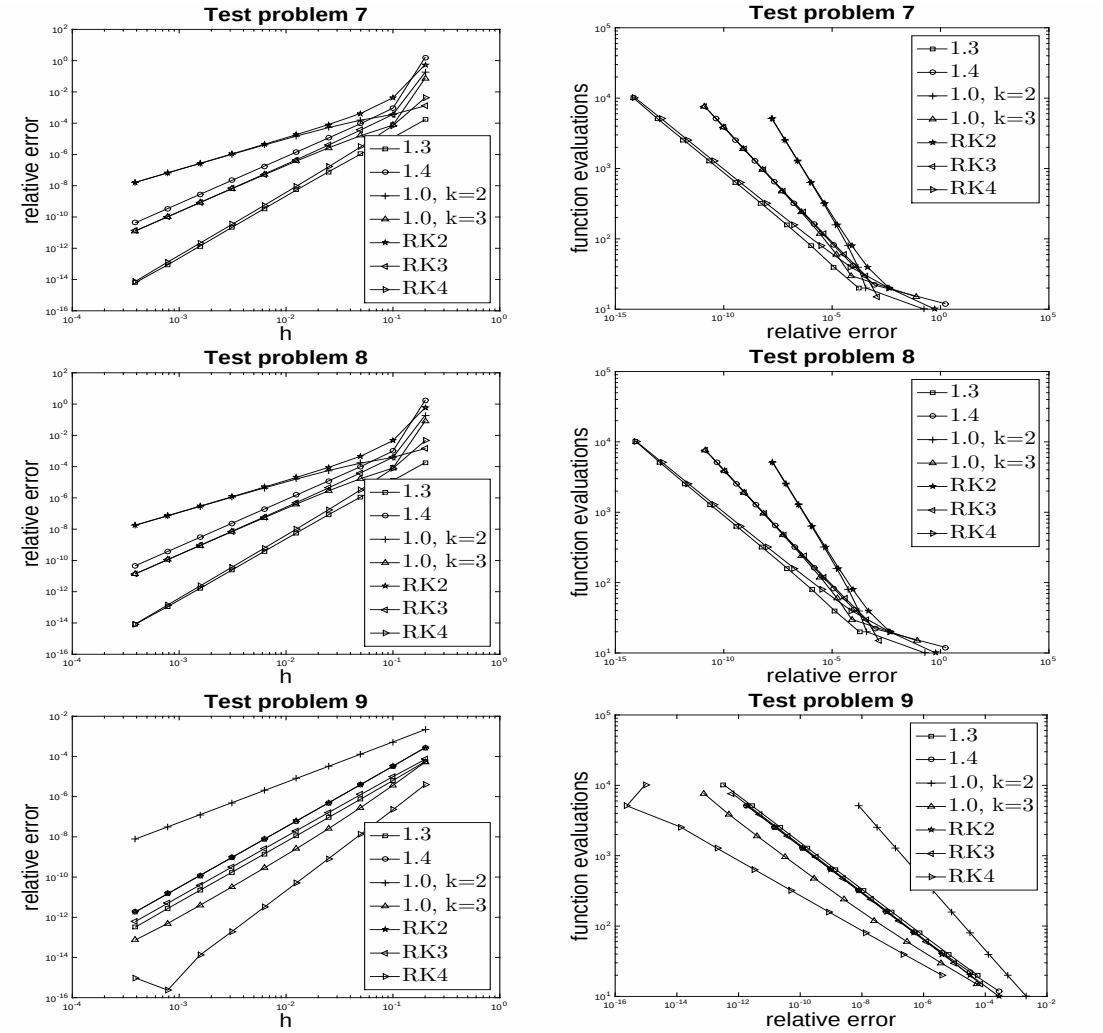


Figure 4.6: Relative error versus stepsize and function evaluation versus relative error for problems 7,8 and 9.

4.2 Convergence and stability analysis of the proposed one-step multi-point methods

In this subsection, theoretical convergence properties of two algorithms considered above are studied. The first method is the Method 1.4 being a one-step multi-point method of the form $(w_1, y_1) = \Phi_h(w_0, y_0)$ closely related to the classical Taylor formula of order three. Here h stands for the integration stepsize and w_1 and y_1 are approximations to $y(t_1)$, with $t_1 = t_0 + h$. More specifically, the extra-point w_1 is to be meant as a preliminary low order approximation to $y(t_1)$ which is then exploited to derive the more accurate

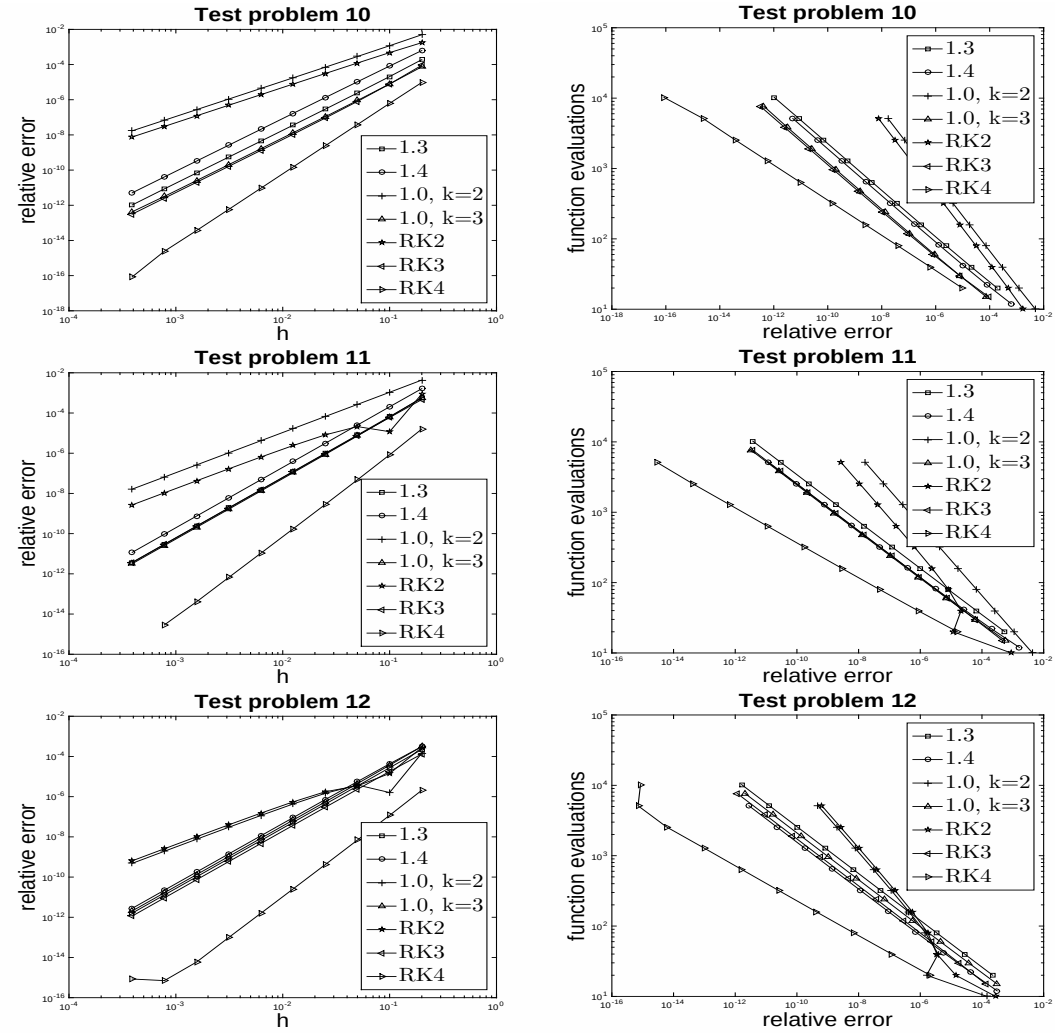


Figure 4.7: Relative error versus stepsize and function evaluation versus relative error for problems 10,11,12.

approximation y_1.

First, let us re-define the algorithms starting from the Method 1.3, which is an *auxiliary* method, denoted by $\hat{y}_1 = \widehat{\Phi}_h(y_0)$, which will prove very helpful to properly define method Φ_h as well as to study its convergence and stability properties. For sake of simplicity, but without loss of generality, in the sequel we assume that problem (4.1) is scalar and autonomous. However, results pertaining to the non-autonomous and vector cases are provided, as well. Notice that we avoid the computation of high-order mixed derivatives on the Infinity Computer, which will be the object of a future research. For later use, we list the shape of the first four derivatives of the solution $y(t)$ of (4.1)

evaluated at t_0, in terms of the function f and its derivatives:

$$\begin{aligned} y'(t_0) &= f(y_0), \\ y''(t_0) &= f'(y_0)f(y_0), \\ y'''(t_0) &= f''(y_0)(f(y_0))^2 + (f'(y_0))^2 f(y_0), \\ y^{(iv)}(t_0) &= f'''(y_0)(f(y_0))^3 + 4f''(y_0)f'(y_0)(f(y_0))^2 + (f'(y_0))^3 f(y_0). \end{aligned} \tag{4.7}$$

On the Infinity Computer, the derivatives appearing in (4.7) are evaluated with the aid of formulae (3.11)-(3.12). In standard arithmetic they are provided analytically, even though we also illustrate the effects of approximating them by suitable divided differences.

The following result regards the computational cost to compute the Taylor coefficients.

Theorem 4.1 ([14, 139])**.** *If the evaluation of $f(y)$ involves r elementary functions, the computational complexity of the evaluation of $f(y)$, $f'(y)$, $\dots f^{(s-1)}(y)$ is*

$$C = rs^2 + O(s). \tag{4.8}$$

The efficiency of Taylor methods as compared with classical Runge–Kutta methods has been discussed in [14]. For a numerical approach to the computation of the coefficients in Taylor expansions see [136].

Let us consider first the standard Taylor formula of order two to obtain an initial guess, say v_1, of $y(t_1)$ (see (4.7)):

$$v_1 = y_0 + hy'(t_0) + \frac{h^2}{2}y''(t_0) \equiv y_0 + hf(y_0) + \frac{h^2}{2}f'(y_0)f(y_0). \tag{4.9}$$

Let us use again the same formula to obtain an approximation to the solution of (4.1), denoted by $p_2(t)$, in a neighborhood of t_1:

$$p_2(t) = v_1 + f(v_1)(t - t_1) + \frac{1}{2}f'(v_1)f(v_1)(t - t_1)^2. \tag{4.10}$$

Rather than advancing the solution in time, we exploit the information brought by $p_2(t)$ to improve the accuracy of the numerical solution at time t_1. To this end, we first recast $p_2(t)$ as a polynomial expanded around t_0

$$\begin{aligned} p_2(t) &= v_1 + f(v_1)(t - t_0 + t_0 - t_1) + \frac{1}{2}f'(v_1)f(v_1)(t - t_0 + t_0 - t_1)^2 \\ &= v_1 - hf(v_1) + \frac{h^2}{2}f'(v_1)f(v_1) + (1 - hf'(v_1))\, f(v_1)(t - t_0) \\ &\quad + \frac{1}{2}f'(v_1)f(v_1)(t - t_0)^2, \end{aligned} \tag{4.11}$$

and then we blend the coefficients of the polynomial $p_2(t)$ with the corresponding ones in the classical Taylor formula to form a new second degree polynomial $q_2(t)$, namely

$$\begin{aligned} q_2(t) &= \alpha_0 y_0 + (1-\alpha_0)\big(v_1 - hf(v_1) + \frac{h^2}{2} f'(v_1)f(v_1)\big) \\ &\quad + \big(\alpha_1 f(y_0) + (1-\alpha_1)\,(1-hf'(v_1))\,f(v_1)\big)(t-t_0) \\ &\quad + \frac{1}{2}\big(\alpha_2 f'(y_0)f(y_0) + (1-\alpha_2)f'(v_1)f(v_1)\big)(t-t_0)^2, \end{aligned} \tag{4.12}$$

which will be used to advance the solution, by setting $\hat{y}_1 = \widehat{\Phi}_h(y_0) = q_2(t_1)$. The parameters α_i, $i = 0, 1, 2$, will be selected in order to improve the convergence and stability properties of the standard second order Taylor formula. The use of convex combinations in (4.12) comes from imposing the consistency conditions up to order two. In other words, order two is achieved independently of the choice of the parameters α_i. Furthermore, without loss of generality, we assume $\alpha_0 = 1$ since it can be shown that the value of α_0 does not alter the shape of the resulting method. The following scheme then summarizes the implementation details of the method to construct the numerical approximation $\hat{y}_k \simeq y(t_k)$, with $t_k = t_0 + kh$.

$$\begin{aligned} &\hat{y}_0 = y_0 \\ &h = (T - t_0)/n \\ &\text{for} \quad k = 1, \ldots, n \\ &\qquad v_k = \hat{y}_{k-1} + hf(\hat{y}_{k-1}) + \frac{h^2}{2} f'(\hat{y}_{k-1})f(\hat{y}_{k-1}) \\ &\qquad \hat{y}_k = \hat{y}_{k-1} + h\big(\alpha_1 f(\hat{y}_{k-1}) + (1-\alpha_1)\,(1-hf'(v_k))\,f(v_k)\big) \\ &\qquad\qquad + \frac{h^2}{2}\big(\alpha_2 f'(\hat{y}_{k-1})f(\hat{y}_{k-1}) + (1-\alpha_2)f'(v_k)f(v_k)\big) \\ &\text{end} \end{aligned} \tag{4.13}$$

To reduce the computational effort per step associated with the implementation of the method defined by $\widehat{\Phi}_h$, we consider a variant consisting in approximating the values of $f(\hat{y}_{k-1})$ and $f'(\hat{y}_{k-1})$ appearing in algorithm (4.13) by means of suitable known quantities available for free. More precisely, we assume that the very first step is performed by method $\widehat{\Phi}_h$, and we set $w_1 = v_1$ and $y_1 = \hat{y}_1 = \widehat{\Phi}_h(y_0)$.

At the second step, we avoid the evaluations of $f(y_1)$ and $f'(y_1)$, as required by (4.9), and instead approximate them by $f(w_1)$ and $f'(w_1)$ respectively, which we inherit from the previous step. More in general, to compute the subsequent approximations $y_k = \Phi_h(y_{k-1})$, $k = 2, 3, \ldots$, we replace the

quantities $f(y_{k-1})$ and $f'(y_{k-1})$, required by algorithm (4.13), by $f(w_{k-1})$ and $f'(w_{k-1})$ respectively. The implementation details of the method defined by Φ_h are summarized below.

$$
\begin{aligned}
&h = (T - t_0)/n \\
&w_1 = y_0 + hf(y_0) + \frac{h^2}{2} f'(y_0) f(y_0) \\
&y_1 = y_0 + h\big(\alpha_1 f(y_0) + (1-\alpha_1)\,(1 - hf'(w_1))\, f(w_1)\big) \\
&\qquad + \frac{h^2}{2}\big(\alpha_2 f'(y_0) f(y_0) + (1-\alpha_2) f'(w_1) f(w_1)\big) \\
&\text{for} \quad k = 2, \ldots, n \\
&\qquad w_k = y_{k-1} + hf(w_{k-1}) + \frac{h^2}{2} f'(w_{k-1}) f(w_{k-1}) \\
&\qquad y_k = y_{k-1} + h\big(\alpha_1 f(w_{k-1}) + (1-\alpha_1)\,(1 - hf'(w_k))\, f(w_k)\big) \\
&\qquad\qquad + \frac{h^2}{2}\big(\alpha_2 f'(w_{k-1}) f(w_{k-1}) + (1-\alpha_2) f'(w_k) f(w_k)\big) \\
&\text{end}
\end{aligned}
\tag{4.14}
$$

Remark 4.1. The shape of Algorithms (4.13) and (4.14) does not change for vector-valued functions. In such a case, $f'(y)$ denotes the Jacobian matrix of $f(y)$. We also recall that a non-autonomous system $y' = f(t, y)$, with $y \in \mathbb{R}^n$, may be always recast as an autonomous system of the form $z' = F(z)$, with $z \in \mathbb{R}^{n+1}$, by setting:

$$
z = \begin{pmatrix} y \\ t \end{pmatrix} \qquad \text{and} \quad F(z) = \begin{pmatrix} f(y) \\ 1 \end{pmatrix}.
$$

Modulo this transformation, the two algorithms above may also handle the non-autonomous case.

The analysis of the integrator described in algorithm (4.14) will be carried out by interpreting Φ_h as a perturbation of $\widehat{\Phi}_h$. For this reason, we begin with stating some preliminary results pertaining to this latter formula.

Theorem 4.2. *If the coefficients α_1 and α_2 satisfy*

$$
\alpha_1 - \alpha_2 = \frac{1}{3} \tag{4.15}
$$

the method $\widehat{\Phi}_h$ has order $p = 3$. In addition, if the coefficients α_1 and α_2 are selected as

$$
\alpha_1 = \frac{5}{6}, \quad \alpha_2 = \frac{1}{2}, \tag{4.16}
$$

(thus also satisfying (4.15)*),* $\widehat{\Phi}_h$ *becomes the standard fourth-order Taylor formula when applied to the linear problem* $y' = \lambda y$*,* $\lambda \in \mathbb{C}$*, where* $\mathbb{C}$ *denotes the set of complex numbers.*

Proof. The local truncation error associated with the method $\widehat{\Phi}_h$ at the first step is

$$\tau(h) = y(t_0 + h) - \widehat{\Phi}_h(y_0) = \sum_{k\geq 0} \frac{y^{(k)}(t_0)}{k!} h^k - \widehat{\Phi}_h(y_0), \tag{4.17}$$

assuming f analytical. Thus, to estimate $\tau(h)$ we need to expand $\widehat{\Phi}_h(y_0)$ in powers of h. With reference to (4.13) with $k = 1$, we consider the expansions

$$\begin{aligned} f(v_1) &= f\big(y_0 + hf(y_0) + \tfrac{h^2}{2} f'(y_0)f(y_0)\big) \\ &= f(y_0) + f'(y_0)\big(hf(y_0) + \tfrac{h^2}{2} f'(y_0)f(y_0)\big) \\ &\quad + \tfrac{f''(y_0)}{2}\big(hf(y_0) + \tfrac{h^2}{2} f'(y_0)f(y_0)\big)^2 + O(h^3) \\ &= f(y_0) + hf(y_0)f'(y_0) \\ &\quad + \tfrac{h^2}{2}\big[f(y_0)(f'(y_0))^2 + (f(y_0))^2 f''(y_0)\big] + O(h^3), \end{aligned}$$

$$\begin{aligned} f'(v_1) &= f'\big(y_0 + hf(y_0) + \tfrac{h^2}{2} f'(y_0)f(y_0)\big) \\ &= f'(y_0) + hf(y_0)f''(y_0) \\ &\quad + \tfrac{h^2}{2}\big[f(y_0)f'(y_0)f''(y_0) + (f(y_0))^2 f'''(y_0)\big] + O(h^3). \end{aligned}$$

Plugging them into the equation in (4.13) defining $\hat{y}_1$ yields

$$\begin{aligned} \widehat{\Phi}_h(y_0) &= \hat{y}_1 = y_0 + f(y_0)h + \frac{1}{2} f(y_0)f'(y_0)h^2 \\ &\quad + \frac{\alpha_1 - \alpha_2}{2} f(y_0)\big((f'(y_0))^2 + f(y_0)f''(y_0)\big)h^3 + O(h^4). \end{aligned} \tag{4.18}$$

Inserting (4.18) into (4.17) and taking into account relations (4.7) we deduce that the method defined by $\widehat{\Phi}_h$ has order three if condition (4.15) is fulfilled.[1]

In the specific case where the problem is linear, namely $f(y) = \lambda y$ and hence $f^{(k)}(y) = \lambda^k y$, a direct computation based upon the previous argument shows that

$$\tau(h) = \big(\frac{1}{3!} - \frac{\alpha_1}{2} + \frac{\alpha_2}{2}\big)(h\lambda)^3 y_0 + \big(\frac{1}{4!} + \frac{1}{4} - \frac{\alpha_1}{2} + \frac{\alpha_2}{4}\big)(h\lambda)^4 y_0 + \sum_{k\geq 5} \frac{y^{(k)}(t_0)}{k!} h^k,$$

[1] One may check that it is not possible to achieve order four under the assumption that the coefficients α_i are independent of the problem at hand.

and order four is achieved by imposing

$$\begin{cases} \dfrac{\alpha_1}{2} - \dfrac{\alpha_2}{2} &= \dfrac{1}{3!}, \\ \dfrac{\alpha_1}{2} - \dfrac{\alpha_2}{4} &= \dfrac{1}{4!} + \dfrac{1}{4}, \end{cases}$$

which has solution (4.16). □

More in general, method $\widehat{\Phi}_h$ has order four when applied to autonomous linear problems $y' = Ay + b$. Consequently, an increase of order is also experienced numerically for nonlinear problems when the dynamics takes place in a neighborhood of an equilibrium point where the Lyapunov first approximation theorem holds true.

The linear stability analysis amounts to study the (global) asymptotic behavior of the sequence $\hat{y}_n = \widehat{\Phi}_h(\hat{y}_{n-1})$ when the method is applied to the well-known linear test equation $y' = \lambda y$, with $\lambda \in \mathbb{C}$. In such an event, as has been shown in Theorem 4.2, the method $\widehat{\Phi}_h$ is equivalent to the fourth order Taylor formula and, as a direct consequence, we can state the following result.

Corollary 4.1. *Method $\widehat{\Phi}_h$ shares the same linear stability properties of the fourth-order Taylor method.*

More specifically, setting $q = h\lambda$, we have $\hat{y}_n = \widehat{R}(q)^n y_0$, where

$$\widehat{R}(q) = \sum_{k=0}^{4} \frac{q^k}{k!}$$

is the stability function. We recall that the region of absolute stability of a generic method providing a sequence y_n when applied to the linear test equation, is defined as

$$\mathcal{D} = \{q \in \mathbb{C} : \, y_n \to 0, \text{ as } n \to \infty\}$$

In our case, we see that $q \in \mathcal{D} \Leftrightarrow |\widehat{R}(q)| < 1$.

Let us move now to the study of the method corresponding to the map Φ_h. In particular, let us take advantage of the results previously obtained for the method defined by $\widehat{\Phi}_h$, by regarding Φ_h as a perturbation of $\widehat{\Phi}_h$.

Lemma 4.1. *Under the assumption* (4.15)*, the sequences* $(v_k, \hat{y}_k)$ *and* (w_k, y_k) *defined in algoritms* (4.13) *and* (4.14) *respectively, are related as*

$$w_k = v_k + O(h^4), \qquad y_k = \hat{y}_k + O(h^4), \quad \textit{with } k = 0, 1, \ldots, N, \tag{4.19}$$

where N is a positive constant integer, independent of h.

Proof. Let us use an induction argument on the index k. For $k = 1$, (4.19) is obviously true since, by definition, (4.13) and (4.14) provide the same approximations ($w_1 = v_1$ and $y_1 = \hat{y}_1$). Assume that property (4.19) holds true for $k-1$. From the proof of Theorem 4.2 we deduce that

$$\begin{aligned}\hat{y}_k &= \hat{y}_{k-1} + hf(\hat{y}_{k-1}) + \frac{h^2}{2} f(\hat{y}_{k-1}) f'(\hat{y}_{k-1}) \\ &\quad + \frac{h^3}{3!}\big(f(\hat{y}_{k-1})(f'(\hat{y}_{k-1}))^2 + (f(\hat{y}_{k-1}))^2 f''(\hat{y}_{k-1})\big) + O(h^4),\end{aligned}$$

thus, comparing with the definition of v_k in (4.13), we conclude that

$$\hat{y}_k - v_k = O(h^3), \qquad \text{for any} \quad k = 0, 1, \ldots. \tag{4.20}$$

Exploiting the induction hypothesis and (4.20), we finally get

$$\begin{aligned}w_k &= y_{k-1} + f(w_{k-1})h + f(w_{k-1}) f'(w_{k-1}) \frac{h^2}{2} \\ &= \hat{y}_{k-1} + O(h^4) + f(v_{k-1} + O(h^4))h \\ &\quad + f(v_{k-1} + O(h^4)) f'(v_{k-1} + O(h^4)) \frac{h^2}{2} \\ &= \hat{y}_{k-1} + f(v_{k-1})h + f(v_{k-1}) f'(v_{k-1}) \frac{h^2}{2} + O(h^4) \\ &= v_k + O(h^4),\end{aligned}$$

and analogously

$$\begin{aligned}y_k &= y_{k-1} + \big(\alpha_1 f(w_{k-1}) + (1-\alpha_1)\,(1 - hf'(w_k))\, f(w_k))\big)h \\ &\quad + \big(\alpha_2 f'(w_{k-1}) f(w_{k-1}) + (1-\alpha_2) f'(w_k) f(w_k)\big) \frac{h^2}{2} \\ &= \hat{y}_{k-1} + O(h^4) + \big[\alpha_1 f(v_{k-1} + O(h^4)) \\ &\qquad + (1-\alpha_1)\,(1 - hf'(v_k + O(h^4)))\, f(v_k + O(h^4)))\big]h \\ &\quad + \big[\alpha_2 f'(v_{k-1} + O(h^4)) f(v_{k-1} + O(h^4)) \\ &\qquad + (1-\alpha_2) f'(v_k + O(h^4)) f(v_k + O(h^4))\big] \frac{h^2}{2} \\ &= \hat{y}_{k-1} + \big(\alpha_1 f(v_{k-1}) + (1-\alpha_1)\,(1 - hf'(v_k))\, f(v_k))\big)h \\ &\quad + \big(\alpha_2 f'(v_{k-1}) f(v_{k-1}) + (1-\alpha_2) f'(v_k) f(v_k)\big) \frac{h^2}{2} + O(h^4) \\ &= \hat{y}_{k-1} + \big(\alpha_1 f(\hat{y}_{k-1} + O(h^3)) + (1-\alpha_1)\,(1 - hf'(v_k))\, f(v_k))\big)h \\ &\quad + \big[\alpha_2 f'(\hat{y}_{k-1} + O(h^3)) f(\hat{y}_{k-1} + O(h^3)) \\ &\qquad + (1-\alpha_2) f'(v_k) f(v_k)\big] \frac{h^2}{2} + O(h^4) \\ &= \hat{y}_{k-1} + \big(\alpha_1 f(\hat{y}_{k-1}) + (1-\alpha_1)\,(1 - hf'(v_k))\, f(v_k))\big)h \\ &\quad + \big(\alpha_2 f'(\hat{y}_{k-1}) f(\hat{y}_{k-1}) + (1-\alpha_2) f'(v_k) f(v_k)\big) \frac{h^2}{2} + O(h^4) \\ &= \hat{y}_k + O(h^4).\end{aligned}$$

This completes the proof. □

The following result then comes from standard computation, by letting now the integer N in (4.19) to increase as h decreases, under the constraint that $t = t_0 + Nh$ is a fixed time in the integration interval $[t_0, t_f]$.

Theorem 4.3. *The method defined by the map Φ_h and described in algorithm* (4.14) *has order of convergence $p = 3$, that is*

$$|y_N - y(t)| = O(h^3), \qquad \textit{with} \quad h = \frac{t - t_0}{N}. \tag{4.21}$$

Let us observe that a result analogous to (4.21) applies to the error $|y_N - \hat{y}_N|$ and consequently, unlike $\widehat{\Phi}_h$, the method Φ_h does not increase its order when applied to linear problems.

Concerning the linear stability analysis, the application of the method defined by Φ_h to the test equation $y' = \lambda y$ yields

$$\begin{aligned} w_k &= y_{k-1} + qw_{k-1} + \frac{1}{2}q^2 w_{k-1} \\ y_k &= y_{k-1} + \alpha_1 q w_{k-1} + (1-\alpha_1)q(1-q)w_k + \frac{1}{2}\alpha_2 q^2 w_{k-1} + \frac{1}{2}(1-\alpha_2)q^2 w_k, \end{aligned}$$

or, in matrix form,

$$\begin{pmatrix} 1 & 0 \\ (1-\alpha_1)q(1-q) - \frac{1}{2}(1-\alpha_2)q^2 & 1 \end{pmatrix} \begin{pmatrix} w_k \\ y_k \end{pmatrix} = \begin{pmatrix} q + \frac{1}{2}q^2 & 1 \\ \alpha_1 q + \frac{1}{2}\alpha_2 q^2 & 1 \end{pmatrix} \begin{pmatrix} w_{k-1} \\ y_{k-1} \end{pmatrix}.$$

Inverting the matrix at the left-hand side, we arrive at

$$z_k = R(q) z_{k-1},$$

with $z_k = (w_k, y_k)^\top$, and the real matrix $R(q) = \big(r_{ij}(q)\big)$ defined as

$$\begin{aligned} r_{11}(q) &= \frac{q^2}{2} + q, \\ r_{12}(q) &= 1, \\ r_{21}(q) &= (\frac{\alpha_1}{2} + \frac{\alpha_2}{4} - \frac{3}{4})q^4 + (\frac{\alpha_1}{2} + \frac{\alpha_2}{2} - 1)q^3 + (\frac{\alpha_2}{2} - \alpha_1 + 1)q^2 + \alpha_1 q, \\ r_{22}(q) &= (\alpha_1 + \frac{\alpha_2}{2} - \frac{3}{2})q^2 + (1-\alpha_1)q + 1. \end{aligned} \tag{4.22}$$

Denoting by $\lambda_1(q)$ and $\lambda_2(q)$ the two eigenvalues of $R(q)$, it turns out that the absolute stability region of the method Φ_h is given by

$$\mathcal{D} \;=\; \left\{ q \in \mathbb{C} : \max_{i=1,2} |\lambda_i(q)| < 1 \right\}.$$

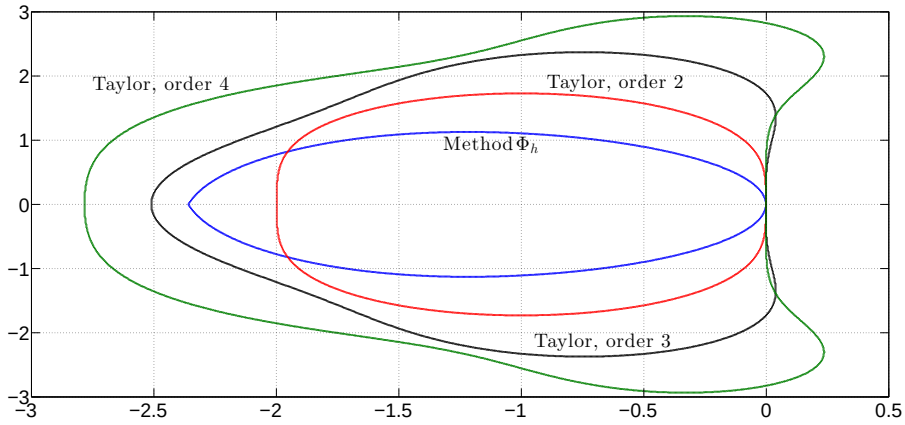


Figure 4.8: Absolute stability region of method Φ_h compared with those corresponding to the Taylor methods of order 2, 3, and 4.

Figure 4.8 displays the absolute stability regions related to methods $\widehat{\Phi}_h$ (i.e., for linear problems, the fourth order Taylor method), and Φ_h for the choice of parameters as in (4.16). The Taylor formula of order two has also been considered for comparison purposes.

To get numerical evidence of the theoretical results presented above, we solve a few test problems by means of the methods defined at (4.13) and (4.14) and compare their performance with Taylor methods of order up to four. Such comparisons are carried out by evaluating the error in the numerical approximations at the end of the time integration interval t_f. As a reference solution against which to measure the obtained accuracy, we use the theoretical solution of the problem when available, or a very accurate numerical solution obtained in MATLAB® with the aid of a solver in the ODE suite. For a m-dimensional problem, we use the following mixed-type error

$$E = \max_{1 \le i \le m} \frac{|y^{(i)}(t_f) - y_N^{(i)}|}{1 + |y^{(i)}(t_f)|}, \tag{4.23}$$

where $y^{(i)}(t_f)$ denotes the ith component of the reference solution evaluated at time t_f and $y_N^{(i)}$ is the corresponding numerical approximation ($t_f = t_0 + Nh$).

Both the Infinity Computer and analytic differentiation lead to an accurate approximation of the derivatives, thus yielding equivalent results. Consequently, we do not provide comparisons between these two different implementation procedures.

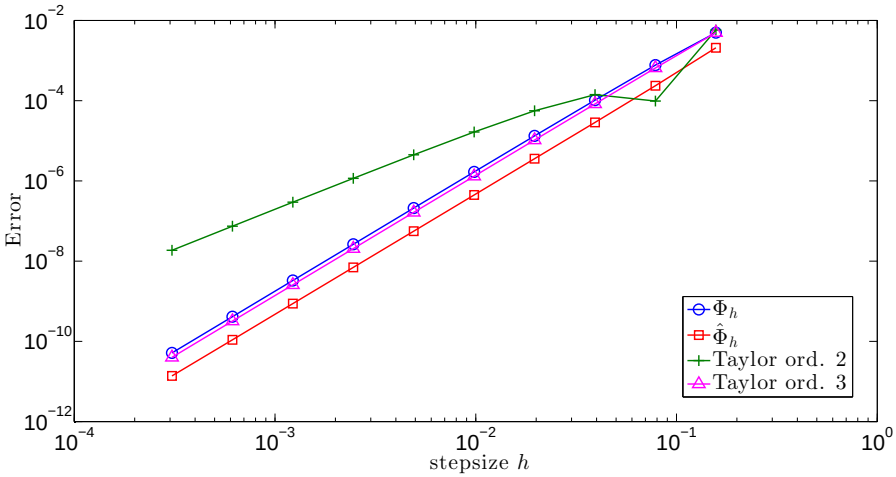


Figure 4.9: Problem 1. Errors versus stepsize.

Problem 1. Consider the scalar (non-autonomous) initial value problem

$$\begin{cases} y' = \dfrac{\cos(\pi t)}{1+y}, \quad t \in [0, \pi], \\ y(0) = 0, \end{cases} \tag{4.24}$$

admitting solution

$$y(t) = \sqrt{\frac{2}{\pi}\sin(\pi t) + 1} - 1.$$

Let us solve problem (4.24) for decreasing values of the stepsize h

$$h_n = \frac{h_0}{2^n}, \quad \text{with} \quad h_0 = \frac{\pi}{20}, \quad \text{and} \quad n = 0, 1, 2, \ldots, 9,$$

and compare the numerical approximations at the end of the time interval with the exact one, according to formula (4.23). Figure 4.9 summarizes the obtained results. As is expected, the errors produced by the new methods Φ_h and $\widehat{\Phi}_h$ decay with order three with respect to the stepsize. Though avoiding the computation of $y''(t)$, the performance of both methods is analogous to the third-order Taylor formula. It is worth noting that the implementation of method Φ_h requires precisely the same computational cost as the second-order Taylor formula, but achieving much better results (see Figure 4.9).

As was emphasized previously, if the two methods are implemented on the Infinity Computer, the user may avoid to provide the analytic expression

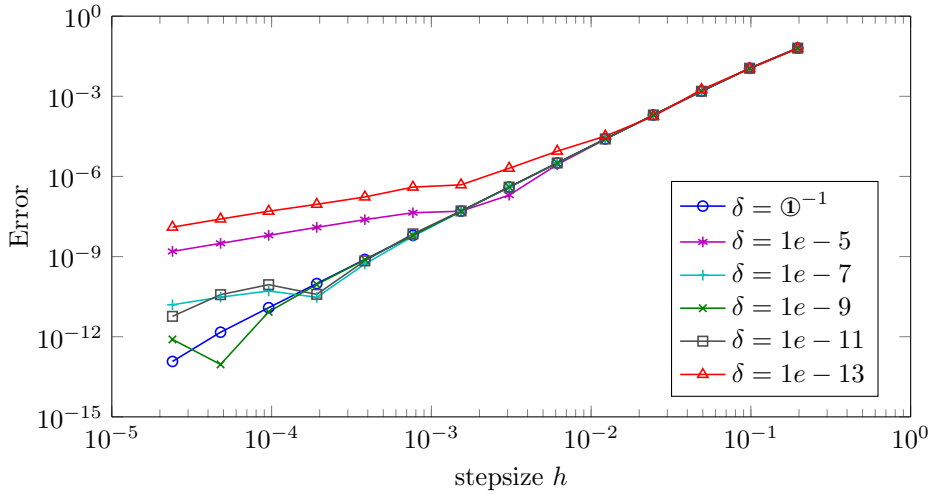


Figure 4.10: Problem 1. Errors generated by method Φ_h for different choices of the parameter δ to approximate the required derivatives of f, according to formula (4.25).

of the derivative $f'(y)$, which is instead obtained with the aid of the first order difference formula

$$f'(y) \approx \frac{f(y+\delta) - f(y)}{\delta}, \tag{4.25}$$

by setting $\delta = ①^{-1}$. Of course, in case of vector or non-autonomous systems, a formula equivalent to (4.25) is used to evaluate the first partial derivatives of the Jacobian matrix (see Remark 4.1). Let us recall that this choice yields the exact value of $f'(y)$ up to machine precision. Therefore, as a further experiment, it makes sense to see how a numerical approximation of the derivatives corresponding to decreasing values of the parameter δ, now taken as a positive real number in the standard arithmetic, may affect the overall behavior of the integrator, as compared to the choice $\delta = ①^{-1}$, also considering that for all these choices, including the latter one, the computational effort remains unchanged.

In Figure 4.10, the performance of method Φ_h, for $\delta = 10^{-5}$, 10^{-7}, 10^{-9}, 10^{-11}, 10^{-13}, and $\delta = ①^{-1}$, is compared. Inside the range of the prescribed stepsizes, one can see that choosing a value of δ not sufficiently small results in an eventual loss of convergence rate. As an example, for $\delta = 10^{-5}$ this happens when h becomes as small as 10^{-3} (solid line with asterisks in the picture). This simply means that, for $h < 10^{-3}$, the derivatives are not accurately computed and a smaller value of δ is then required to recover the

order three convergence rate. Improvements are in fact obtained by reducing the value of δ to 10^{-7} and 10^{-9}.

Contrary to what is expected, a further reduction of the discretization step δ in (4.25) results in a loss of efficiency: for $\delta = 10^{-13}$ the performance of the method is poorer than for the largest value considered. The reason is that approximating the derivative by means of formulae such as (4.25) may lead to an ill-conditioned problem for small values of the parameter δ due to cancelation issues related to the difference at numerator. Table 4.7 reports the errors in the approximation of $y'' = f'(y)f(y)$ in the last computed point, obtained by replacing $f'(w_n)$ with the corresponding first order difference formula (see (4.7) and (4.14)). A loss of significant digits is experienced starting from $\delta = 10^{-9}$, independently of the stepsize h used. This unpleasant outcome is an effect of the use of finite arithmetic, and is responsible for the eventual order reduction phenomenon inferred from Figure 4.10 and discussed above.

This is not the case when the same computation is carried out on the Infinity Computer. In fact, while the computer representation of $x \pm \delta$, with x a floating point number and $\delta \in \mathbb{R}$, produces an error, the quantities $x \pm ①$ are precisely represented and thus do not give rise to any digit cancelation phenomenon. For completeness, we also report, in Figure 4.11, the result obtained by approximating the first derivative by means of the second order difference formula

$$f'(y) \approx \frac{f(y+\delta) - f(y-\delta)}{2\delta}, \tag{4.26}$$

which is more frequently used by codes when a numerical evaluation of the Jacobian matrix $f'(y)$ is required.[2] We can see that an analogous reduction of order takes place in this case, as well. This means that the loss of accuracy cannot be prevented by improving the accuracy of the discretization formula, but is an unavoidable outcome of the standard floating-point arithmetic.

Problem 2. In Theorem 4.2, the special feature of method $\widehat{\Phi}_h$ to become a fourth order formula, actually the fourth order Taylor formula, when applied to a linear problem, is shown. One may argue that the performance of the method may benefit from this property even when applied to nonlinear systems whose dynamics takes place in a neighborhood of an equilibrium point. To illustrate this aspect, we consider the dynamics of a pendulum under influence of gravity. Using Lagrangian mechanics, its motion can be

[2]For $\delta = ①^{-1}$ we continue to use formula (4.25), which is the one implemented on the Infinity Computer.

h \ δ	10^{-5}	10^{-7}	10^{-9}	10^{-11}	10^{-13}
$\pi/2^4$	2.36e-05	2.37e-07	2.61e-08	1.00e-05	1.56e-03
$\pi/2^5$	3.86e-05	3.87e-07	1.92e-08	2.57e-06	1.57e-03
$\pi/2^6$	5.39e-05	5.35e-07	1.19e-07	3.87e-05	2.33e-03
$\pi/2^7$	6.77e-05	6.80e-07	3.68e-07	3.64e-05	1.80e-03
$\pi/2^8$	5.99e-05	6.01e-07	4.17e-07	1.86e-05	2.96e-03
$\pi/2^9$	5.65e-05	5.58e-07	3.35e-07	4.28e-05	7.18e-03
$\pi/2^{10}$	5.49e-05	5.43e-07	4.55e-07	4.44e-05	5.76e-03
$\pi/2^{11}$	5.41e-05	5.35e-07	2.93e-07	3.59e-05	1.66e-04
$\pi/2^{12}$	5.37e-05	5.38e-07	4.63e-07	2.05e-05	4.60e-04
$\pi/2^{13}$	5.35e-05	5.37e-07	5.23e-07	3.45e-05	2.46e-03
$\pi/2^{14}$	5.35e-05	5.28e-07	2.14e-07	4.62e-05	6.34e-03
$\pi/2^{15}$	5.34e-05	5.30e-07	1.50e-07	5.17e-05	7.17e-03
$\pi/2^{16}$	5.34e-05	5.28e-07	3.69e-07	5.51e-05	5.84e-03
$\pi/2^{17}$	5.34e-05	5.27e-07	3.22e-07	3.87e-05	2.70e-03

Table 4.7: Errors in the approximation of y'' in the last computed point, generated by replacing f' with the corresponding first order difference formula (4.25).

described by the dimensionless nonlinear equation

$$\begin{cases} y_1' = y_2, \\ y_2' = -\sin y_1, \end{cases} \tag{4.27}$$

where y_1 is the angle that the pendulum forms with its stable rest position, and y_2 is the angular velocity. In case of small amplitude oscillations around the equilibrium point $(y_1, y_2) = (0, 0)$, the problem may be well described by its linearized version, namely the so called *harmonic oscillator* $y_1'' + y_1 = 0$, obtained through the approximation $\sin y_1 \approx y_1$. We compare the behavior of methods $\widehat{\Phi}_h$ and Φ_h with Taylor methods of order 2, 3 and 4 for the following decreasing values of stepsize:

$$h_n = \frac{h_0}{2^n}, \quad \text{with} \quad h_0 = \frac{\pi}{5}, \quad \text{and} \quad n = 0, 1, 2, \ldots, 7,$$

We use $[t_0, t_f] = [0, 2\pi]$ as integration interval and the two initial conditions $y_0 = (0.5, 0)^\top$ and $y_0 = (1, 0)^\top$ to simulate the system under the circumstance that the nonlinear part of (4.27) may be neglected or not.

Figure 4.12 summarizes the results. The picture at the top shows the error (4.23) versus the stepsize, when the pendulum undertakes mild oscillations so

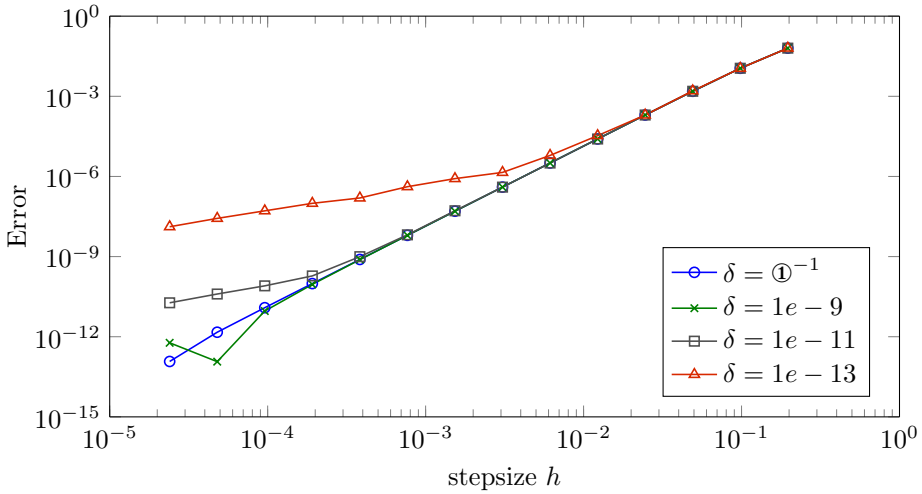


Figure 4.11: Problem 1. Errors generated by method Φ_h for different choices of the parameter δ to approximate the required derivatives of f, according to formula (4.26).

that its dynamic is essentially equivalent to that of a harmonic oscillator. We see that, in accord with Theorem 4.2, the behavior of method $\widehat{\Phi}_h$ is precisely the same as that of the fourth-order Taylor method, while Φ_h displays order three and yields errors similar to the corresponding Taylor formula. In the bottom picture in Figure 4.12 we show the results obtained after doubling the amplitude of oscillations of the pendulum. We see that, in this case, the nonlinear part of the vector field is no longer negligible even though the benefits of the order four do not completely evaporate. In fact, for large stepsizes, the behavior of method $\widehat{\Phi}_h$ and the Taylor method of order four is again quite similar and both methods produce comparable errors. As the stepsize decreases, the accuracy curve associated with method Φ_h departs from the order four slope and eventually reveals an error reduction rate typical of a third-order method.

4.3 Taylor-Obrechkoff method of order 4 using exact derivatives

The Obrechkoff methods for the solution of (4.1) are linear multistep methods of the following form ($y_{n+j}^{(i)}$ denotes the approximation to the i-th derivative

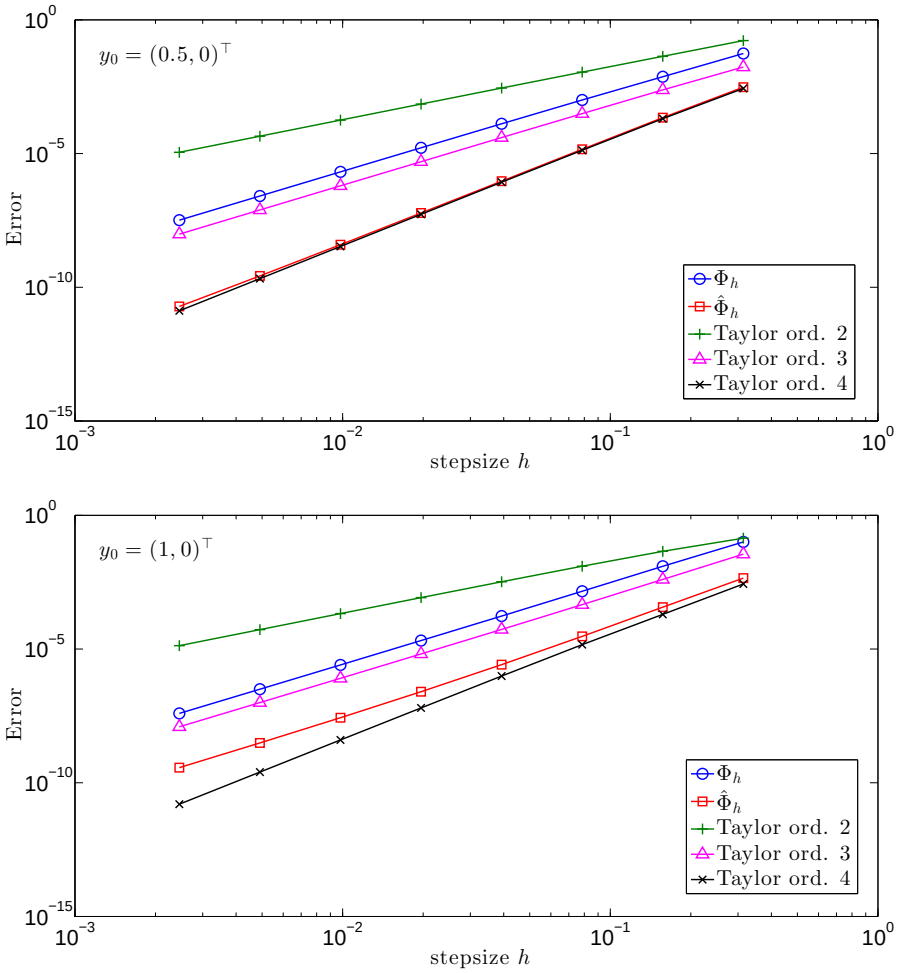


Figure 4.12: Problem 2. Errors versus stepsize.

of the solution at t_{n+j}):

$$\sum_{j=0}^{k} \alpha_j y_{n+j} = \sum_{i=1}^{l} h^i \sum_{j=0}^{k} \beta_{ij} y_{n+j}^{(i)}.$$

In literature these methods and some variants are mainly used for the numerical solution of second order problems (see for example [132, 217, 218] and the references therein).

One issue with these implicit methods is that high order derivatives are usually approximated by finite differences that involve terms in y and y'. The behavior of the methods strongly depends on these approximations [227]. This drawback is overcome on the Infinity Computer since, in order to calculate the k-th derivative at the point t_i, k infinitesimals steps from the point t_i using the Euler formula with $h = ①^{-1}$ could be executed as follows (see Section 3.1 for details)

$$y_{i,1} = y_i + ①^{-1} f(y_i), \quad y_{i,2} = y_{i,1} + ①^{-1} f(y_{i,1}), \ \ldots$$

$$y_{i,k} = y_{i,k-1} + ①^{-1} f(y_{i,k-1}).$$

Then, approximations of the derivatives can be obtained by the forward differences Δ_h^k, with $h = ①^{-1}$ as follows

$$y^{(k)}(t_i) = \frac{\Delta^k_{①^{-1}}}{①^{-k}} + O(①^{-1}) \quad \text{where} \quad \Delta^k_{①^{-1}} = \sum_{j=0}^{k} (-1)^j \binom{k}{j} y_{i,k-j} \quad \text{and } y_{i,0} = y_i.$$

Since the error of the approximation is $O(①^{-1})$, the finite part of the value $\Delta^k_{①^{-1}}/①^{-k}$ gives the *exact* derivative $y^{(k)}(t_i)$.

Obrechkoff methods of high order have stability problems when applied to the numerical solution of first order ordinary differential equations. These problems have been overcame in literature in many way. The use of these methods as boundary value methods [70], or the introduction of some variants considering future steps points [208] are mentioned here. It is also interesting to see the connection of these methods with the so called superimplicit schemes [143], that can be considered as a class of boundary value methods (BVMs) as defined in their original formulation [8, 9, 22, 90, 133]. Here, the following Obrechkoff method of order 4 is studied:

$$y_k = y_{k-1} + \frac{h}{2}\big(f(y_{k-1}) + f(y_k)\big) - \frac{h^2}{12}\big(f'(y_k)f(y_k) - f'(y_{k-1})f(y_{k-1})\big). \quad (4.28)$$

This is an implicit A-stable order 4 method [115]. Simple generalizations of this method, in a predictor corrector form, called Taylor-Obrechkoff methods

hereinafter, are presented. The resulting schemes are similar to the one presented in [7, 206]. Numerical tests show the effectiveness of these variants.

The new method, following the results presented in [7, 206], uses the Taylor method of order 2 as predictor to obtain, using formula (4.28) as a corrector, a second derivative explicit method. The formula is used in a PEC (one prediction step, one evaluation and one corrector step) way which means that the last evaluation is avoided. The resulting scheme is the following:

$$\begin{aligned} w_k &= y_{k-1} + hf(w_{k-1}) + \frac{h^2}{2} f'(w_{k-1}) f(w_{k-1}) \\ y_k &= y_{k-1} + \frac{h}{2}\big(f(w_{k-1}) + f(w_k)\big) - \frac{h^2}{12}\big(f'(w_k)f(w_k) - f'(w_{k-1})f(w_{k-1})\big) \end{aligned} \tag{4.29}$$

Let us call this method TEO. Since the Taylor scheme is only order two, the use of only one corrector step, reduces the order of the resulting scheme [115]. To obtain an order 4 method, more corrector steps are required, and we use them in the form $PE_2(E_1C)^3$, where now E_2 means evaluation of the second derivative and E_1 evaluation of the first derivative. We call this method $TE_2(E_1O)^3$:

$$\begin{aligned} w_k &= y_{k-1} + hf(v'_{k-1}) + \frac{h^2}{2} f'(w_{k-1}) f(w_{k-1}) \\ v_k &= y_{k-1} + \frac{h}{2}\big(f(v'_{k-1}) + f(w_k)\big) - \frac{h^2}{12}\big(f'(w_k)f(w_k) - f'(w_{k-1})f(w_{k-1})\big) \\ v'_k &= y_{k-1} + \frac{h}{2}\big(f(v'_{k-1}) + f(v_k)\big) - \frac{h^2}{12}\big(f'(w_k)f(w_k) - f'(w_{k-1})f(w_{k-1})\big) \\ y_k &= y_{k-1} + \frac{h}{2}\big(f(v'_{k-1}) + f(v'_k)\big) - \frac{h^2}{12}\big(f'(w_k)f(w_k) - f'(w_{k-1})f(w_{k-1})\big) \end{aligned} \tag{4.30}$$

It can be proved that this scheme is now of order 4. Both the methods have a finite region of absolute stability that is plotted in figure 4.13 and compared with the method called Φ_h in [7].

In order to study the proposed methods experimentally, we consider two numerical tests substantiating the convergence rates of the proposed algorithms. The first one, taken from [23] is

$$y' = \frac{y - 2ty^2}{1+t}, \quad 0 \le t \le t_f = 1, \qquad y(0) = 1, \quad y(t) = \frac{1+t}{1+t^2}. \tag{4.31}$$

The second test is the pendulum problem as solved in [7]

$$\begin{cases} y'_1 = y_2, \\ y'_2 = -\sin y_1, \qquad 0 \le t \le t_f = 2\pi \qquad y(0) = (0.5, 1)^T. \end{cases} \tag{4.32}$$

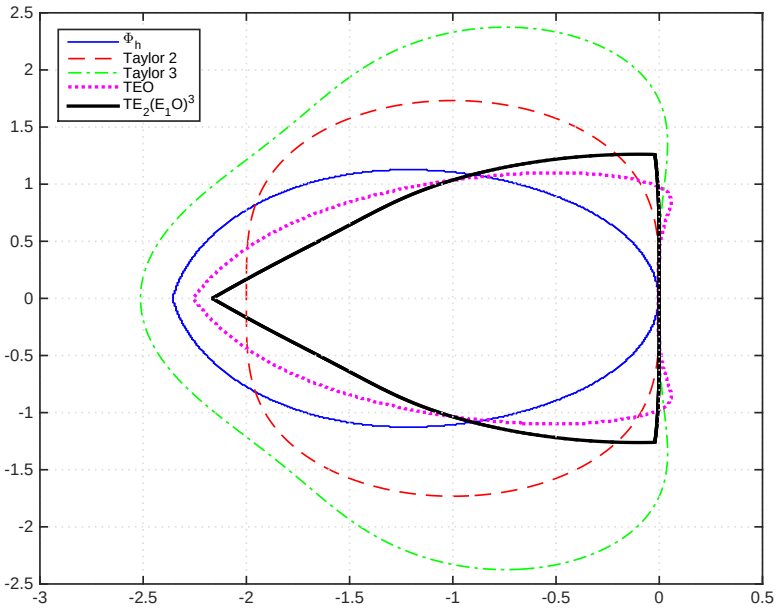


Figure 4.13: Absolute stability region of methods TEO and $TE_2(E_1O)^3$ compared with those corresponding to the Taylor methods of order 2, 3 and method Φ_h defined in [7].

For both the problems the behavior of the proposed Taylor-Obrechkoff methods is compared with the methods $\widehat{\Phi}_h$ and Φ_h from the previous sections and with Taylor methods of order 2, 3 and classical explicit Runge-Kutta methods of order 3 and 4 for the following decreasing values of stepsize: $h_n = h_0/2^n$, with $h_0 = (t_f - t_0)/5$, and $n = 0, 1, 2, \ldots, 9$. As is expected, the errors produced by the new methods TEO and $TE_2(E_1O)^3$ decay with order three and four with respect to the stepsize. It is worth noting that the implementation of method TEO requires precisely the same computational cost as the second-order Taylor formula, the methods $TE_2(E_1O)^3$, requires more evaluations only of the first derivative, with respect to the second order formula, but both achieve much better results (see Figure 4.14).

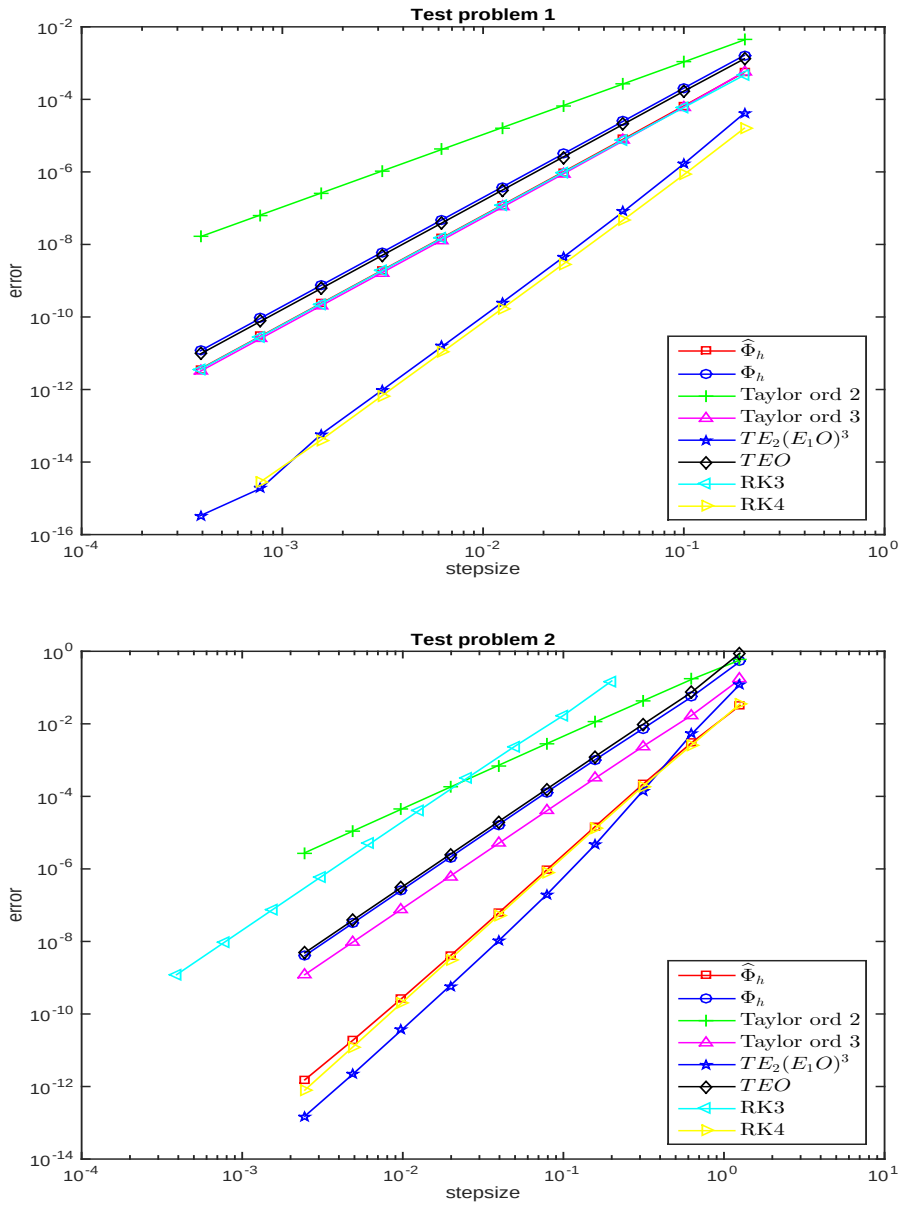


Figure 4.14: Test problem 1 (at the top) and test problem 2 (at the bottom), errors versus stepsize.

Conclusion

The main obtained results of this thesis consist of the following:

– New acceleration techniques to speed up the global search have been introduced in univariate derivative-free global optimization. They can be used in geometric and information frameworks for construction univariate Lipschitz global optimization algorithms. All of the considered methods automatically switch from the global search to the local one and back avoiding so the necessity to stop the global phase manually. An original mixture of new and traditional computational steps has allowed to construct 22 different global optimization algorithms having, however, a similar structure. In particular, 9 instances of this mixture can lead to known global optimization methods and the remaining 13 methods described in this work are new. All of them have been studied theoretically and numerically compared on more than 100 theoretical and applied benchmark tests. It has been shown that the introduced acceleration techniques allow the global optimization methods to significantly speed up the search with respect to some known algorithms.

– Two practical engineering problems have been studied: finding the minimal root of a non-linear equation problem from electrical engineering and a sum of damped sinusoids from statistics. Numerical experiments on the presented classes of engineering problems confirmed the advantages of the proposed techniques, as well.

– Two new graphical techniques (called "operational zones" and "aggregated operational zones") for a systematic comparison of deterministic and stochastic global optimization algorithms have been introduced and analyzed in this work. It has been shown that these new graphical methodologies for comparing global optimization methods of a different nature are quite representative. Almost all qualitative characteristics that can be represented by numerical tables can be also observed from these graphs. Moreover, the best, the worst, and average performances of stochastic methods can be easily found, as well. A number of

popular metaheuristics having a stochastic nature have been compared in this research with deterministic Lipschitz methods by using proposed methodologies. Massive experimental study of global optimization algorithms of different nature have been performed in more than 1000 test problems with more than 1 000 000 runs.

– A new generator of test problems with non-linear constraints based on the GKLS-generator for testing algorithms of constrained global optimization has been introduced. It can generate classes of test problems with non-linear constraints, known global minimizers, and parameterizable difficulty, where the objective function can be non-differentiable, continuously differentiable and twice continuously differentiable, while the constraints are continuously differentiable.

– The Infinity Computing framework has been applied to handling of ill-conditioning in univariate and multidimensional optimization. It has been shown that several ill-conditioned problems in the traditional computational framework become well-conditioned if the Infinity Computing is applied. Presented techniques can be used in different fields, where ill-conditioning appears.

– A new class of problems with the objective function having infinite or infinitesimal Lipschitz constants has been introduced. The strong homogeneity of several univariate algorithms for Lipschitz global optimization problems has been studied in the framework of the Infinity Computing paradigm. Finally, it has been shown that in certain cases the usage of numerical infinities and infinitesimals can avoid ill-conditioning produced by scaling.

– The impact of the Infinity Computing paradigm on practical solution of nonsmooth unconstrained optimization problems, where the objective function is assumed to be convex and not necessarily differentiable, has been studied. The main attention has been focused on a family of nonsmooth optimization methods based on a variable metric approach, and the infinity computing techniques have been used for numerically dealing with some quantities which can assume values arbitrarily small or large, as a consequence of nonsmoothness.

– Finally, several explicit numerical methods for solving ordinary differential equations have been proposed. Theoretical convergence properties of the proposed methods have been studied. It has been shown that the methods of higher order can be used with the calculation of the derivatives exactly using the Infinity Computer.

Based on the obtained results, 8 papers have been published in the international journals and 1 paper has been submitted, 1 contribution to the book and 9 proceedings of the international conferences have been also published:

1. Ya.D. Sergeyev, M.S. Mukhametzhanov, D.E. Kvasov and D. Lera, "Derivative-free local tuning and local improvement techniques embedded in the univariate global optimization", *Journal of Optimization Theory and Applications* **171** (1), pp. 186 - 208, 2016.

2. Ya.D. Sergeyev, D.E. Kvasov, M.S. Mukhametzhanov, "Operational zones for comparing metaheuristic and deterministic one-dimensional global optimization algorithms", *Mathematics and Computers in Simulation* **141**, pp. 96 – 109, 2017.

3. Ya.D. Sergeyev, M.S. Mukhametzhanov, F. Mazzia, F. Iavernaro and P. Amodio, "Numerical Methods for Solving Initial Value Problems on the Infinity Computer", *International Journal of Unconventional Computing* **12** (1), pp. 3-23, 2016.

4. P. Amodio, F. Iavernaro, F. Mazzia, M.S. Mukhametzhanov, Ya.D. Sergeyev, "A generalized Taylor method of order three for the solution of initial value problems in standard and infinity floating-point arithmetic", *Mathematics and Computers in Simulation* **141**, pp. 24 – 39, 2017.

5. M. Gaudioso, G. Giallombardo, M.S. Mukhametzhanov, "Numerical infinitesimals in a variable metric method for convex nonsmooth optimization", *Applied Mathematics and Computation*, **318**, pp.312–320, 2018.

6. D.E. Kvasov, M.S. Mukhametzhanov, "Metaheuristic vs. Deterministic global optimization algorithms: The univariate case", *Applied Mathematics and Computation*, **318**, pp. 245–259, 2018.

7. Ya. D. Sergeyev, D.E. Kvasov, M.S. Mukhametzhanov, "On strong homogeneity of a class of global optimization algorithms working with infinite and infinitesimal scales", *Communications in Nonlinear Science and Numerical Simulation*, **59**, pp. 319–330, 2018.

8. Ya. D. Sergeyev, D.E. Kvasov, M.S. Mukhametzhanov, "On the efficiency of nature-inspired metaheuristics in expensive global optimization with limited budget", *Scientific Reports*, **8**, n. 453, 2018, doi: 10.1038/s41598-017-18940-4.

9. Ya. D. Sergeyev, D.E. Kvasov, M.S. Mukhametzhanov, "Remark on Algorithm 829: Classes of multiextremal test problems with nonlinear constraints and known global solutions", 2018, submitted.

10. Ya.D. Sergeyev, D.E. Kvasov, M.S. Mukhametzhanov. "On the least-squares fitting of data by sinusoids". In: "Advances in Stochastic and Deterministic Global Optimization" (ed. by P.M. Pardalos, A. Zhigljavsky, and J. Zilinskas), series "Springer Optimization and Its Applications", vol. 107, chapter 11. Springer, Switzerland, 2016.

11. Ya.D. Sergeyev, D.E. Kvasov, M.S. Mukhametzhanov, "Emmental-type GKLS-based multiextremal smooth test problems with non-linear constraints", in: Learning and Intelligent Optimization Conference (LION 2017), Lecture Notes in Computer Science, vol. 10556, Springer (2017), pp. 383–388.

12. Ya.D. Sergeyev, D.E. Kvasov, M.S. Mukhametzhanov, A. De Franco, "Acceleration Techniques in the Univariate Lipschitz Global Optimization", in Numerical Computations: Theory and Algorithms: NUMTA 2016: the 2nd international conference and summer school, edited by Ya.D. Sergeyev, D.E. Kvasov, F. Dell'Accio and M.S. Mukhametzhanov (AIP Conference Proceedings, 2016) vol. 1776, 090051.

13. F. Mazzia, Ya.D. Sergeyev, F. Iavernaro, P. Amodio, M.S. Mukhametzhanov, "Numerical Methods for solving ODEs on the Infinity Computer", in Numerical Computations: Theory and Algorithms: NUMTA 2016: the 2nd international conference and summer school, edited by Ya.D. Sergeyev, D.E. Kvasov, F. Dell'Accio and M.S. Mukhametzhanov (AIP Conference Proceedings, 2016) vol. 1776, 090033.

14. Ya.D. Sergeyev, D.E. Kvasov, M.S. Mukhametzhanov, "Operational Zones for Global Optimization Algorithms", in Proceedings of the XIII Global Optimization Workshop GOW'16, 4-8 September 2016, University of Minho, Braga, Portugal, pp. 85-88, ISBN 978-989-20-6764-3

15. Ya.D. Sergeyev, D.E. Kvasov, M.S. Mukhametzhanov, "Comments upon the usage of derivatives in Lipschitz global optimization", in ICNAAM 2015: 13th International Conference of Numerical Analysis and Applied Mathematics, vol. 1738, edited by T. Simos (AIP Conference Proceedings, 2016), pp. 400004-1 – 4.

16. D.E. Kvasov, M.S. Mukhametzhanov, "One-dimensional global search: Nature-inspired vs. Lipschitz methods", in ICNAAM 2015: 13th International Conference of Numerical Analysis and Applied Mathematics,

vol. 1738, edited by T. Simos (AIP Conference Proceedings, 2016), pp. 400012-1 – 4.

17. D.E. Kvasov, M.S. Mukhametzhanov, Ya.D. Sergeyev, "A Numerical Comparison of Some Deterministic and Nature-Inspired Algorithms for Black-Box Global Optimization", in B.H.V. Topping, P. Ivanyi (Editors), "Proceedings of the Twelfth International Conference on Computational Structures Technology", Civil-Comp Press, Stirlingshire, UK, Paper 169, 2014. Doi:10.4203/ccp.106.169

18. Ya.D. Sergeyev, M.S. Mukhametzhanov, D.E. Kvasov, "Examples of solving ODEs given as a Black-Box on the Infinity Computer", in Proceedings of the International Conference "New Trends in Numerical Analysis 2015 (NETNA 2015)", Falerna (CZ), Italy, 18-21 June 2015, pp.90-91.

19. D.E. Kvasov, M.S. Mukhametzhanov, Ya.D. Sergeyev, "Numerical methods for solving black-box global optimization problems", in Proceedings of the International Conference "New Trends in Numerical Analysis 2015 (NETNA 2015)", Falerna (CZ), Italy, 18-21 June 2015, pp.73-74.

The Emmental-type generator of test problems from Section 2.3 has been implemented in C. All the algorithms proposed in this work have been implemented in $C++$. In particular, univariate geometric and information algorithms have been implemented in a unique framework according to the General Scheme (see Section 1.1). Finally, all graphical visualizations have been realized using MATLAB®.

Acknowledgements

First of all, I would like to thank my scientific advisor Prof. Yaroslav Sergeyev for his guidance, support and patience during my Ph.D. study. Words cannot express how grateful I am to him. It was a honor to be his Student.

I would like to thank also Prof. Dmitri Kvasov for his contribution and priceless help during my study. This research became significantly better due to his clear thinking and useful comments on my Ph.D. thesis.

My special gratitude to Prof. Vladimir Grishagin, Prof. Renato De Leone, Prof. Anatoly Zhigljavsky, Prof. Antanas Žilinskas, Prof. Julius Žilinskas for useful criticism and comments. Besides the above mentioned, I sincerely thank all my co-authors for their contribution to this research: Prof. Daniela Lera, Prof. Francesca Mazzia, Prof. Felice Iavernaro, Prof. Pierluigi Amodio, Prof. Manlio Gaudioso, Prof. Giovanni Giallombardo, and Dr. Angela De Franco.

My appreciation also to Prof. Sergio Greco, being the Director of the Department of Computer Engineering, Modelling, Electronics and Systems Science at the University of Calabria, and to Prof. Felice Crupi, being the Coordinator of the Ph.D. program, for the possibility to work at one of the best departments in Italy.

Last but not the least, my cordial thanks to my parents for supporting me during my Ph.D. Study. I am eternally grateful for everything I have learnt from you.

Bibliography

[1] E. Aarts, J. Korst, and W. Michiels. Simulated annealing. In E. K. Burke and G. Kendall, editors, *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, pages 265–285. Springer, 2014.

[2] A. Adamatzky. Unconventional computing. *International Journal of General Systems*, 43(7):671 – 672, 2014.

[3] R. A. Adams. *Single variable calculus*. Pearson Education Canada, Ontario, 5 edition, 2003.

[4] B. Addis and M. Locatelli. A new class of test functions for global optimization. *Journal of Global Optimization*, 38(3):479–501, 2007.

[5] P. Amodio, L. Brugnano, and F. Iavernaro. Energy-conserving methods for hamiltonian boundary value problems and applications in astrodynamics. *Advances in Computational Mathematics*, 41:881–905, 2015.

[6] P. Amodio, C. J. Budd, O. Koch, G. Settanni, and E. Weinmüller. Asymptotical computations for a model of flow in saturated porous media. *Applied Mathematics and Computation*, 237:155–167, 2014.

[7] P. Amodio, F. Iavernaro, F. Mazzia, M. S. Mukhametzhanov, and Ya. D. Sergeyev. A generalized Taylor method of order three for the solution of initial value problems in standard and infinity floating-point arithmetic. *Mathematics and Computers in Simulation*, 141:24–39, 2016.

[8] P. Amodio and F. Mazzia. Boundary value methods for the solution of differential-algebraic equations. *Numerische Mathematik*, 66(1):411–421, 1994.

[9] P. Amodio and F. Mazzia. Boundary value methods based on adams-type methods. *Applied Numerical Mathematics*, 18(1-3):23–35, 1995.

[10] P. Amodio and D. Trigiante. *Elementi di calcolo numerico (in Italian)*. Pitagora Editrice Bologna, 126, 1993.

[11] A. M. Bagirov, B. Karasozen, and M. Sezer. Discrete gradient method: Derivative-free method for nonsmooth optimization. *Journal of Optimization Theory and Applications*, 137:317–334, 2008.

[12] A. M. Bagirov, N. Karmitsa, and M. M. Mäkelä. *Introduction to Nonsmooth Optimization: Theory, Practice and Software*. Springer, 2014.

[13] K. Barkalov, V. Gergel, and I. Lebedev. Solving global optimization problems on GPU cluster. In T. E. Simos, editor, *AIP Conference Proceedings*, volume 1738 (400006), 2016.

[14] R. Barrio. Performance of the Taylor series method for ODEs/DAEs. *Applied Mathematics and Computation*, 163(2):525–545, 2005.

[15] R. Barrio, F. Blesa, and M. Lara. VSVO formulation of the Taylor method for the numerical solution of ODEs. *Computers & Mathematics with Applications*, 50:93–111, 2005.

[16] R. Barrio, M. Rodríguez, A. Abad, and F. Blesa. Breaking the limits: The Taylor series method. *Applied Mathematics and Computation*, 217:7940–7954, 2011.

[17] J. E. Beasley. Obtaining test problems via Internet. *Journal of Global Optimization*, 8(4):429–433, 1996.

[18] V. Beiranvand, W. Hare, and Y. Lucet. Best practices for comparing optimization algorithms. *Optimization and Engineering*, 18(4):815–848, 2017.

[19] P. Benioff. The computer as a physical system: A microscopic quantum mechanical hamiltonian model of computers as represented by turing machines. *Journal of Statistical Physics*, 22(5):563 – 591, 1980.

[20] P. Bloomfield. *Fourier Analysis of Time Series: An Introduction*. John Wiley & Sons, New York, 2000.

[21] I. N. Bocharov and A. A. Feldbaum. Automatic optimizator for search of least of several minimums (global optimizator). *Automation and Remote Control*, 23(3):289–301, 1962.

[22] L. Brugnano and D. Trigiante. *Solving Differential Problems by Multistep Initial and Boundary Value Methods*. Gordon & Breach,Amsterdam, 1998.

[23] J. C. Butcher. *Numerical methods for ordinary differential equations*. John Wiley & Sons, Chichester, 2 edition, 2003.

[24] R. H. Byrd, J. Nocedal, and R. B. Schnabel. Representations of quasi-Newton matrices and their use in limited memory methods. *Mathematical Programming*, 63:129–156, 1994.

[25] J. M. Calvin, Y. Chen, and A. Žilinskas. An adaptive univariate global optimization algorithm and its convergence rate for twice continuously differentiable functions. *Journal of Optimization Theory and Applications*, 155(2):628–636, 2012.

[26] J. M. Calvin and A. Žilinskas. One-dimensional global optimization for observations with noise. *Computers & Mathematics with Applications*, 50(1-2):157–169, 2005.

[27] D. L. Carnì and G. Fedele. Multi-sine fitting algorithm enhancement for sinusoidal signal characterization. *Computer Standards & Interfaces*, 34(6):535–540, 2012.

[28] L. G. Casado, I. García, and Ya. D. Sergeyev. Interval algorithms for finding the minimal root in a set of multiextremal non-differentiable one-dimensional functions. *SIAM Journal of Scientific Computing*, 24(2):359–376, 2002.

[29] E. W. Cheney and A. A. Goldstein. Newton's method for convex programming and Tchebycheff approximation. *Numerische Mathematik*, 1:253–268, 1959.

[30] A. R. Conn, K. Scheinberg, and L. N. Vicente. *Introduction to Derivative-Free Optimization*. SIAM, Philadelphia, USA, 2009.

[31] S. Costanzo. Synthesis of multi-step coplanar waveguide-to-microstrip transition. *Progress in Electromagnetics Research*, 113:111–126, 2011.

[32] T. Csendes, editor. *Developments in Reliable Computing*. Kluwer Academic Publishers, Dordrecht, 1999.

[33] A. L. Custódio, J. F. Aguilar Madeira, A. I. F. Vaz, and L. N. Vicente. Direct multisearch for multiobjective optimization. *SIAM Journal on Optimization*, 21(3):1109–1140, 2011.

[34] L. D'Alotto. Cellular automata using infinite computations. *Applied Mathematics and Computation*, 218(16):8077–8082, 2012.

[35] L. D'Alotto. A classification of two-dimensional cellular automata using infinite computations. *Indian Journal of Mathematics*, 55:143–158, 2013.

[36] L. D'Alotto. A classification of one-dimensional cellular automata using infinite computations. *Applied Mathematics and Computation*, 255:15–24, 2015.

[37] P. Daponte, D. Grimaldi, A. Molinaro, and Y. D. Sergeyev. An algorithm for finding the zero-crossing of time signals with Lipschitzean derivatives. *Measurement*, 16(1):37–49, 1995.

[38] P. Daponte, D. Grimaldi, A. Molinaro, and Ya. D. Sergeyev. Fast detection of the first zero-crossing in a measurement signal set. *Measurement*, 19(1):29–39, 1996.

[39] S. De Cosmis and R. De Leone. The use of grossone in mathematical programming and operations research. *Applied Mathematics and Computation*, 218(16):8029–8038, 2012.

[40] K. Deb and R. B. Agrawal. Simulated binary crossover for continuous search space. *Complex Systems*, 9(2):115–148, 1995.

[41] K. Deb and A. Kumar. Real-coded genetic algorithms with simulated binary crossover: Studies on multimodal and multiobjective problems. *Complex Systems*, 9:431–454, 1995.

[42] A. V. Demyanov, A. Fuduli, and G. Miglionico. A bundle modification strategy for convex minimization. *European Journal of Operational Research*, 180:38–47, 2007.

[43] V. F. Demyanov and V. N. Malozemov. *Introduction to Minimax*. Wiley, 1974.

[44] J. E. Dennis and J. J. Moré. Quasi-newton methods, motivation and theory. *SIAM Review*, 19:46 – 89, 1977.

[45] J. G. Digalakis and K. G. Margaritis. On benchmarking functions for genetic algorithms. *International Journal of Computer Mathematics*, 77(4):481–506, 2001.

[46] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91:201–213, 2002.

[47] S. M. Elsakov and V. I. Shiryaev. Homogeneous algorithms for multiextremal optimization. *Computational Mathematics and Mathematical Physics*, 50:1642–1654, 2010.

[48] J. B. Elsner and A. A. Tsonis. *Singular Spectrum Analysis: A New Tool in Time Series Analysis*. Springer, New York, 1996.

[49] Yu. G. Evtushenko. *Numerical Optimization Techniques*. Translations Series in Mathematics and Engineering. Springer–Verlag, Berlin, 1985.

[50] D. Famularo, Ya. D. Sergeyev, and P. Pugliese. Test problems for Lipschitz univariate global optimization with multiextremal constraints. In G. Dzemyda, V. Šaltenis, and A. Žilinskas, editors, *Stochastic and Global Optimization*, pages 93–109. Kluwer Academic Publishers, 2002.

[51] G. Fedele and A. Ferrise. A frequency-locked-loop filter for biased multi-sinusoidal estimation. *IEEE Transactions on Signal Processing*, 62(5):1125–1134, 2014.

[52] I. Fister, I. Fister Jr., X.-S. Yang, and J. Brest. A comprehensive review of firefly algorithms. *Swarm and Evolutionary Computation*, 13:34–46, 2013.

[53] C. A. Floudas and P. M. Pardalos. *A Collection of Test Problems for Constrained Global Optimization Algorithms*. Springer-Verlag, Berlin Heidelberg, 1990.

[54] C. A. Floudas and P. M. Pardalos, editors. *Encyclopedia of Optimization (6 Volumes)*. Springer, 2nd edition, 2009.

[55] C. A. Floudas et al. *Handbook of Test Problems in Local and Global Optimization*. Kluwer Academic Publishers, Dordrecht, 1999.

[56] A. Frangioni, B. Gendron, and E. Gorgone. On the computational efficiency of subgradient methods: a case study in combinatorial optimization. Technical Report 2015-41, CIRRELT, Montreal, Canada, 2015.

[57] A. Fuduli and M. Gaudioso. Tuning strategy for the proximity parameter in convex minimization. *Journal of Optimization Theory and Applications*, 130(1):95–112, 2006.

[58] A. Fuduli, M. Gaudioso, G. Giallombardo, and G. Miglionico. A partially inexact bundle method for convex semi-infinite minmax problems. *Communications in Nonlinear Science and Numerical Simulation*, 21:172–180, 2014.

[59] J. M. Gablonsky and C. T. Kelley. A locally-biased form of the DIRECT algorithm. *Journal of Global Optimization*, 21:27–37, 2001.

[60] M. Gallagher and B. Yuan. A general-purpose tunable landscape generator. *IEEE Transactions on Evolutionary Computation*, 10(5):590–603, 2006.

[61] W. Gander, M. J. Gander, and F. Kwok. *Scientific Computing. An introduction using Maple and Matlab*. Springer, 900, 2014.

[62] Yang Gao, Wenbo Du, and Gang Yan. Selectively-informed particle swarm optimization. *Scientific Reports*, 5:9295, 2015.

[63] H. Garnier and L. Wang, editors. *Identification of Continuous-Time Models from Sampled Data*. Springer-Verlag, London, 2008.

[64] M. Gaudioso, G. Giallombardo, and M. S. Mukhametzhanov. Numerical infinitesimals in a variable metric method for convex nonsmooth optimization. *Applied Mathematics and Computation*, 318:312–320, 2018.

[65] M. Gaviano, D. E. Kvasov, D. Lera, and Ya. D. Sergeyev. Algorithm 829: Software for generation of classes of test functions with known local and global minima for global optimization. *ACM Transactions on Mathematical Software*, 29(4):469–480, 2003.

[66] A. V. Gergel, V. A. Grishagin, and R. G. Strongin. Development of the parallel adaptive multistep reduction method. *Vestnik of Lobachevsky State University of Nizhni Novgorod*, 6(1):216–222, 2013. (In Russian).

[67] V. Gergel, V. A. Grishagin, and A. Gergel. Adaptive nested optimization scheme for multidimensional global search. *Journal of Global Optimization*, 66:35–51, 2016.

[68] V. P. Gergel. A global search algorithm using derivatives. In Yu. I. Neymark, editor, *Systems Dynamics and Optimization*, pages 161–178. N. Novgorod University Press, 1992. (In Russian).

[69] V. P. Gergel and Ya. D. Sergeyev. Sequential and parallel algorithms for global minimizing functions with Lipschitzian derivatives. *Computers & Mathematics with Applications*, 37(4–5):163–179, 1999.

[70] P. Ghelardoni and P. Marzulli. Stability of some boundary value methods for IVPs. *Applied Numerical Mathematics*, 18(1-3):141–153, 1995.

[71] J. W. Gillard. Cadzow's basic algorithm, alternating projections and singular spectrum analysis. *Statistics and its Interface*, 3(3):335–343, 2010.

[72] J. W. Gillard and D. E. Kvasov. Lipschitz optimization methods for fitting a sum of damped sinusoids to a series of observations. *Statistics and its Interface*, 10:59–70, 2017.

[73] J. W. Gillard and A. A. Zhigljavsky. Analysis of structured low rank approximation as an optimisation problem. *Informatica*, 22(4):489–505, 2011.

[74] J. W. Gillard and A. A. Zhigljavsky. Optimization challenges in the structured low rank approximation problem. *Journal of Global Optimization*, 57(3):733–751, 2013.

[75] J. W. Gillard and A. A. Zhigljavsky. Stochastic algorithms for solving structured low-rank matrix approximation problems. *Communications in Nonlinear Science and Numerical Simulation*, 21(1-3):70–88, 2015.

[76] N. Golyandina, V. Nekrutkin, and A. A. Zhigljavsky. *Analysis of Time Series Structure: SSA and Related Techniques*. Chapman & Hall/CRC, Boca Raton, 2001.

[77] D. F. Griffiths and D. J. Higham. *Numerical methods for Ordinary differential equations. Initial Value Problems*. Springer Undergraduate Mathematics Series, 271, 2010.

[78] V. A. Grishagin. Operating characteristics of some global search algorithms. In *Problems of Stochastic Search*, volume 7, pages 198–206. Zinatne, Riga, 1978. In Russian.

[79] V. A. Grishagin, R. A. Israfilov, and Ya. D. Sergeyev. Convergence conditions and numerical comparison of global optimization methods based on dimensionality reduction schemes. *Applied Mathematics and Computation*, 318:270–280, 2018.

[80] V. A. Grishagin, Ya. D. Sergeyev, and R. G. Strongin. Parallel characteristic algorithms for solving problems of global optimization. *Journal of Global Optimization*, 10(2):185–206, 1997.

[81] N. Haarala, K. Miettinen, and M. M. Mäkelä. Globally convergent limited memory bundle method for large-scale nonsmooth optimization. *Mathematical Programming*, 109(1):181–205, 2007.

[82] K. Hamacher. On stochastic global optimization of one-dimensional functions. *Physica A: Statistical Mechanics and its Applications*, 354:547–557, 2005.

[83] P. Hansen and B. Jaumard. Lipschitz optimization. In R. Horst and P. M. Pardalos, editors, *Handbook of Global Optimization*, volume 1, pages 407–493. Kluwer Academic Publishers, Dordrecht, 1995.

[84] S. Helwig, J. Branke, and S. Mostaghim. Analysis of bound handling techniques in particle swarm optimization. Technical Report 3010, Institute AIFB and Karlsruhe Institute of Technology, 2010.

[85] J. Herskovits and E. Goulart. Sparse quasi-Newton matrices for large scale nonlinear optimization. In *Proceedings of the 6th World Congress on Structural and Multidisciplinary Optimization*, 2005.

[86] J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms I–II*. Springer-Verlag, Berlin, 1993.

[87] J. H. Holland. *Adaptation in Natural and Artificial Systems: an Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. University of Michigan Press, 1975.

[88] R. Horst and P. M. Pardalos, editors. *Handbook of Global Optimization*, volume 1. Kluwer Academic Publishers, Dordrecht, 1995.

[89] R. Horst and H. Tuy. *Global Optimization – Deterministic Approaches*. Springer–Verlag, Berlin, 1996.

[90] F. Iavernaro and F. Mazzia. Convergence and stability of multistep methods solving nonlinear initial value problems. *SIAM Journal on Scientific Computing*, 18(1):270–285, 1997.

[91] D. I. Iudin, Ya. D. Sergeyev, and M. Hayakawa. Interpretation of percolation in terms of infinity computations. *Applied Mathematics and Computation*, 218(16):8099–8111, 2012.

[92] D. I. Iudin, Ya. D. Sergeyev, and M. Hayakawa. Infinity computations in cellular automaton forest-fire model. *Communications in Nonlinear Science and Numerical Simulation*, 20(3):861–870, 2015.

[93] D. E. Johnson. *Introduction to Filter Theory*. Prentice Hall Inc., New Jersey, 1976.

[94] D. R. Jones, C. D. Perttunen, and B. E. Stuckman. Lipschitzian optimization without the Lipschitz constant. *Journal of Optimization Theory and Applications*, 79:157–181, 1993.

[95] Á. Jorba and M. Zou. A software package for the numerical integration of ODEs by means of high-order Taylor methods. *Experimental Mathematics*, 14(1):99–117, 2005.

[96] V. Kanovei and V. Lyubetsky. Grossone approach to Hutton and Euler transforms. *Applied Mathematics and Computation*, 255:36–43, 2015.

[97] D. Karaboga and B. Akay. A comparative study of Artificial Bee Colony algorithm. *Applied Mathematics and Computation*, 214:108–132, 2009.

[98] D. Karaboga and B. Basturk. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *Journal of Global Optimization*, 39(3):459–471, 2007.

[99] N. Karmitsa. Diagonal bundle method for nonsmooth sparse optimization. *Journal of Optimization Theory and Applications*, 166(3):889–905, 2015.

[100] L. H. Kauffman. Infinite computations and the generic finite. *Applied Mathematics and Computation*, 255:25–35, 2015.

[101] J. E. Kelley. The cutting-plane method for solving convex programs. *Journal of SIAM*, 8(4):703–712, 1960.

[102] J. Kennedy, R. C. Eberhart, and Y. Shi. *Swarm Intelligence*. The Morgan Kaufmann Series in Evolutionary Computation. Morgan Kaufmann, San Francisco, USA, 2001.

[103] K. C. Kiwiel. *Methods of descent for nondifferentiable optimization*, volume 1133 of *Lecture notes in mathematics*. Springer-Verlag, 1985.

[104] K. C. Kiwiel. Proximity control in bundle methods for convex nondifferentiable minimization. *Mathematical Programming*, 46:105–122, 1990.

[105] D. E. Kvasov, D. Menniti, A. Pinnarelli, Ya. D. Sergeyev, and N. Sorrentino. Tuning fuzzy power-system stabilizers in multi-machine systems by global optimization algorithms based on efficient domain partitions. *Electric Power Systems Research*, 78(7):1217–1229, 2008.

[106] D. E. Kvasov and M. S. Mukhametzhanov. One-dimensional global search: Nature-inspired vs. Lipschitz methods. In T. E. Simos, editor, *AIP Conference Proceedings*, volume 1738 (400012). 2016.

[107] D. E. Kvasov and M. S. Mukhametzhanov. Metaheuristic vs. deterministic global optimization algorithms: The univariate case. *Applied Mathematics and Computation*, 318:245–259, 2018.

[108] D. E. Kvasov, M. S. Mukhametzhanov, and Ya. D. Sergeyev. A numerical comparison of some deterministic and nature-inspired algorithms for black-box global optimization. In B. H. V. Topping and P. Iványi, editors, *Proceedings of the Twelfth International Conference on Computational Structures Technology*, page 169, Stirlingshire, United Kingdom, 2014. Civil-Comp Press. doi: 10.4203/ccp.106.169.

[109] D. E. Kvasov, M. S. Mukhametzhanov, and Ya. D. Sergeyev. Numerical methods for solving black-box global optimization problems. In *Proceedings of the International Conference "New Trends in Numerical Analysis 2015 (NETNA 2015)", Falerna (CZ), Italy, 18-21 June 2015*, pages 73–74, 2015.

[110] D. E. Kvasov, C. Pizzuti, and Ya. D. Sergeyev. Local tuning and partition strategies for diagonal GO methods. *Numerische Mathematik*, 94(1):93–106, 2003.

[111] D. E. Kvasov and Ya. D. Sergeyev. A univariate global search working with a set of Lipschitz constants for the first derivative. *Optimization Letters*, 3(2):303–318, 2009.

[112] D. E. Kvasov and Ya. D. Sergeyev. Deterministic gobal optimization methods for solving engineering problems. In B. H. V. Topping, editor, *Proceedings of the Eleventh International Conference on Computational Structures Technology*, page 62, Stirlingshire, United Kingdom, 2012. Civil-Comp Press. doi: 10.4203/ccp.99.62.

[113] D. E. Kvasov and Ya. D. Sergeyev. Lipschitz global optimization methods in control problems. *Automation and Remote Control*, 74(9):1435–1448, 2013.

[114] D. E. Kvasov and Ya. D. Sergeyev. Deterministic approaches for solving practical black-box global optimization problems. *Advances in Engineering Software*, 80:58–66, 2015.

[115] J. D. Lambert. *Numerical Methods for Ordinary Differential Systems: The Initial Value Problem*. John Wiley & Sons, Chichester, 1991.

[116] C. Lemaréchal. An algorithm for minimizing convex functions. In J. L. Rosenfeld, editor, *Proceedings IFIP '74 Congress 17*, pages 552–556. North-Holland, Amsterdam, 1974.

[117] C. Lemaréchal and C. Sagastizabal. Variable metric bundle methods: From conceptual to implementable forms. *Mathematical Programming*, 76:393–410, 1997.

[118] D. Lera and Ya. D. Sergeyev. An information global minimization algorithm using the local improvement technique. *Journal of Global Optimization*, 48(1):99–112, 2010.

[119] D. Lera and Ya. D. Sergeyev. Acceleration of univariate global optimization algorithms working with Lipschitz functions and Lipschitz first derivatives. *SIAM Journal on Optimization*, 23(1):508–529, 2013.

[120] D. Lera and Ya. D. Sergeyev. Deterministic global optimization using space-filling curves and multiple estimates of Lipschitz and Hölder constants. *Communications in Nonlinear Science and Numerical Simulation*, 23(1-3):328–342, 2015.

[121] Y. Li, K. J. R. Liu, and J. Razavilar. A parameter estimation scheme for damped sinusoidal signals based on low-rank hankel approximation. *IEEE Transactions on Signal Processing*, 45(2):481–486, 1997.

[122] G. Liuzzi, S. Lucidi, and V. Piccialli. A partition-based global optimization algorithm. *Journal of Global Optimization*, 48(1):113–128, 2010.

[123] G. Liuzzi, S. Lucidi, V. Piccialli, and A. Sotgiu. A magnetic resonance device designed via global optimization techniques. *Mathematical Programming*, 101(2):339–364, 2004.

[124] M. Locatelli. Simulated annealing algorithms for continuous global optimization. In P. M. Pardalos and H. E. Romeijn, editors, *Handbook of Global Optimization*, volume 2, pages 179–229. Kluwer, Dordrecht, 2002.

[125] G. Lolli. Infinitesimals and infinites in the history of mathematics: A brief survey. *Applied Mathematics and Computation*, 218(16):7979–7988, 2012.

[126] G. Lolli. Metamathematical investigations on the theory of grossone. *Applied Mathematics and Computation*, 255:3–14, 2015.

[127] L. Lukšan and J. Vlček. Globally convergent variable metric method for convex nonsmooth unconstrained minimization. *Journal of Optimization Theory and Applications*, 102:593–613, 1999.

[128] M. Margenstern. Using grossone to count the number of elements of infinite sets and the connection with bijections. *p-Adic Numbers, Ultrametric Analysis and Applications*, 3(3):196–204, 2011.

[129] M. Margenstern. An application of grossone to the study of a family of tilings of the hyperbolic plane. *Applied Mathematics and Computation*, 218(16):8005–8018, 2012.

[130] M. Margenstern. Fibonacci words, hyperbolic tilings and grossone. *Communications in Nonlinear Science and Numerical Simulation*, 21(1–3):3–11, 2015.

[131] I. Markovsky. Bibliography on total least squares and related methods. *Statistics and its Interface*, 3(3):329–334, 2010.

[132] F. Mazzia, Ya. D. Sergeyev, F. Iavernaro, P. Amodio, and M. S. Mukhametzhanov. Numerical methods for solving ODEs on the Infinity Computer. In *AIP Conference Proceedings*, volume 1776, page 090033. New York, 2016.

[133] F. Mazzia, A. Sestini, and D. Trigiante. B-spline linear multistep methods and their continuous extensions. *SIAM Journal on Numerical Analysis*, 44(5):1954–1973, 2006.

[134] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

[135] E. Miletics and G. Molnárka. Taylor series method with numerical derivatives for initial value problems. *Journal of Computational Methods in Sciences and Engineering*, 4(1–2):105–114, 2004.

[136] E. Miletics and G. Molnárka. Implicit extension of Taylor series method with numerical derivatives for initial value problems. *Computers & Mathematics with Applications*, 50(7):1167–1177, 2005.

[137] J. Mockus. *Bayesian Approach to Global Optimization*. Kluwer Academic Publishers, Dodrecht, 1988.

[138] F. Montagna, G. Simi, and A. Sorbi. Taking the Pirahã seriously. *Communications in Nonlinear Science and Numerical Simulation*, 21(1–3):52–69, 2015.

[139] R. A. Moore. *Interval Analysis*. Prentice-Hall, Englewood Cliffs, NJ, 1966.

[140] J. J. Moré, B. S. Garbow, and K. E. Hillstrom. Testing unconstrained optimization software. *ACM Transactions on Mathematical Software*, 7(1):17–41, 1981.

[141] J. J. Moré and S. Wild. Benchmarking derivative free optimization algorithms. *SIAM Journal of Optimization*, 20:172–191, 2009.

[142] Y. Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103:127–152, 2005.

[143] B. Neta and T. Fukushima. Obrechkoff versus super-implicit methods for the solution of first- and second-order initial value problems. *Computers and Mathematics with Applications*, 45(1-3):383–390, 2003.

[144] A. Neumaier. Complete search in continuous global optimization and constraint satisfaction. In A. Iserles, editor, *Acta Numerica 2004*, volume 13, pages 271–369. Cambridge University Press, UK, 2004.

[145] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization, 2nd ed.* Springer, New York, 2006.

[146] P. M. Pardalos. Generation of large-scale quadratic programs for use as global optimization test problems. *ACM Transactions on Mathematical Software*, 13(2):133–137, 1987.

[147] P. M. Pardalos and H. E. Romeijn, editors. *Handbook of Global Optimization*, volume 2. Kluwer Academic Publishers, Dordrecht, 2002.

[148] P. M. Pardalos and J. B. Rosen. *Constrained Global Optimization: Algorithms and Applications*, volume 268 of *Springer Lecture Notes In Computer Science*. Springer–Verlag, New York, 1987.

[149] R. Paulavičius, Ya. D. Sergeyev, D. E. Kvasov, and J. Žilinskas. Globally-biased DISIMPL algorithm for expensive global optimization. *Journal of Global Optimization*, 59(2-3):545–567, 2014.

[150] R. Paulavičius, J. Žilinskas, and A. Grothey. Investigation of selection strategies in branch and bound algorithm with simplicial partitions and combination of Lipschitz bounds. *Optimization Letters*, 4(2):173–183, 2010.

[151] R. Paulavičius and J. Žilinskas. *Simplicial Global Optimization*. SpringerBriefs in Optimization. Springer, New York, 2014.

[152] J. D. Pintér. *Global Optimization in Action (Continuous and Lipschitz Optimization: Algorithms, Implementations and Applications)*. Kluwer Academic Publishers, Dordrecht, 1996.

[153] J. D. Pintér. Global optimization: software, test problems, and applications. In P. M. Pardalos and H. E. Romeijn, editors, *Handbook of Global Optimization*, volume 2, pages 515–569. Kluwer Academic Publishers, Dordrecht, 2002.

[154] J. D. Pintér, editor. *Global Optimization: Scientific and Engineering Case Studies*. Springer–Verlag, Berlin, 2006.

[155] S. A. Piyavskij. An algorithm for finding the absolute extremum of a function. *USSR Computational Mathematics and Mathematical Physics*, 12(4):57–67, 1972. in Russian: Zh. Vychisl. Mat. Mat. Fiz., 12(4) (1972), pp. 888–896.

[156] D. S. G. Pollock. *A Handbook of Time Series Analysis, Signal Processing, and Dynamics*. Academic Press Inc., London, 1999.

[157] K. Price, R. M. Storn, and J. A. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization*. Natural Computing Series. Springer, New York, 2005.

[158] L. M. Rios and N. V. Sahinidis. Derivative-free optimization: A review of algorithms and comparison of software implementations. *Journal of Global Optimization*, 56:1247–1293, 2013.

[159] A. Robinson. *Non-standard Analysis*. Princeton Univ. Press, Princeton, 1996.

[160] K. Schittkowski. *More Test Examples for Nonlinear Programming Codes*, volume 282 of *Lecture Notes in Economics and Mathematical Systems*. Springer–Verlag, Berlin, 1987.

[161] J. J. Schneider and S. Kirkpatrick. *Stochastic Optimization*. Springer, Berlin, 2006.

[162] F. Schoen. A wide class of test functions for global optimization. *Journal of Global Optimization*, 3(2):133–137, 1993.

[163] D. Estévez Schwarz and R. Lamour. Projector based integration of DAEs with the Taylor series method using automatic differentiation. *Journal of Computational and Applied Mathematics*, 262:62–72, 2014.

[164] Ya. D. Sergeyev. *Computer system for storing infinite, infinitesimal, and finite quantities and executing arithmetical operations with them*. USA patent 7,860,914 (2010), EU patent 1728149 (2009), RF patent 2395111 (2010).

[165] Ya. D. Sergeyev. An information global optimization algorithm with local tuning. *SIAM Journal on Optimization*, 5(4):858–870, 1995.

[166] Ya. D. Sergeyev. A one-dimensional deterministic global minimization algorithm. *Computational Mathematics and Mathematical Physics*, 35(5):705–717, 1995.

[167] Ya. D. Sergeyev. Global one-dimensional optimization using smooth auxiliary functions. *Mathematical Programming*, 81(1):127–146, 1998.

[168] Ya. D. Sergeyev. On convergence of "Divide the Best" global optimization algorithms. *Optimization*, 44(3):303–325, 1998.

[169] Ya. D. Sergeyev. *Arithmetic of Infinity*. Edizioni Orizzonti Meridionali, CS, 2003, 2nd ed. 2013.

[170] Ya. D. Sergeyev. Univariate global optimization with multiextremal non-differentiable constraints without penalty functions. *Computational Optimization and Applications*, 34(2):229–248, 2006.

[171] Ya. D. Sergeyev. Blinking fractals and their quantitative analysis using infinite and infinitesimal numbers. *Chaos, Solitons & Fractals*, 33(1):50–75, 2007.

[172] Ya. D. Sergeyev. A new applied approach for executing computations with infinite and infinitesimal quantities. *Informatica*, 19(4):567–596, 2008.

[173] Ya. D. Sergeyev. Evaluating the exact infinitesimal values of area of Sierpinski's carpet and volume of Menger's sponge. *Chaos, Solitons & Fractals*, 42(5):3042–3046, 2009.

[174] Ya. D. Sergeyev. Numerical point of view on Calculus for functions assuming finite, infinite, and infinitesimal values over finite, infinite, and infinitesimal domains. *Nonlinear Analysis Series A: Theory, Methods & Applications*, 71(12):e1688–e1707, 2009.

[175] Ya. D. Sergeyev. Counting systems and the First Hilbert problem. *Nonlinear Analysis Series A: Theory, Methods & Applications*, 72(3-4):1701–1708, 2010.

[176] Ya. D. Sergeyev. Lagrange Lecture: Methodology of numerical computations with infinities and infinitesimals. *Rendiconti del Seminario Matematico dell'Università e del Politecnico di Torino*, 68(2):95–113, 2010.

[177] Ya. D. Sergeyev. Higher order numerical differentiation on the Infinity Computer. *Optimization Letters*, 5(4):575–585, 2011.

[178] Ya. D. Sergeyev. On accuracy of mathematical languages used to deal with the Riemann zeta function and the Dirichlet eta function. *p-Adic Numbers, Ultrametric Analysis and Applications*, 3(2):129–148, 2011.

[179] Ya. D. Sergeyev. Using blinking fractals for mathematical modelling of processes of growth in biological systems. *Informatica*, 22(4):559–576, 2011.

[180] Ya. D. Sergeyev. Numerical computations with infinite and infinitesimal numbers: Theory and applications. In Sorokin A. and Pardalos P.M., editors, *Dynamics of Information Systems: Algorithmic Approaches*, pages 1–66. Springer, New York, 2013.

[181] Ya. D. Sergeyev. Solving ordinary differential equations by working with infinitesimals numerically on the infinity computer. *Applied Mathematics and Computation*, 219(22):10668–10681, 2013.

[182] Ya. D. Sergeyev. Numerical infinities and infinitesimals: Methodology, applications, and repercussions on two Hilbert problems. *EMS Surveys in Mathematical Sciences*, 4:219–320, 2017.

[183] Ya. D. Sergeyev, P. Daponte, D. Grimaldi, and A. Molinaro. Two methods for solving optimization problems arising in electronic measurements and electrical engineering. *SIAM Journal on Optimization*, 10(1):1–21, 1999.

[184] Ya. D. Sergeyev, D. Famularo, and P. Pugliese. Index branch-and-bound algorithm for Lipschitz univariate global optimization with multiextremal constraints. *Journal of Global Optimization*, 21(3):317–341, 2001.

[185] Ya. D. Sergeyev and A. Garro. Observability of Turing machines: A refinement of the theory of computation. *Informatica*, 21(3):425–454, 2010.

[186] Ya. D. Sergeyev and A. Garro. Single-tape and multi-tape Turing machines through the lens of the Grossone methodology. *Journal of Supercomputing*, 65(2):645–663, 2013.

[187] Ya. D. Sergeyev and V. A. Grishagin. A parallel method for finding the global minimum of univariate functions. *Journal of Optimization Theory and Applications*, 80(3):513–536, 1994.

[188] Ya. D. Sergeyev and D. E. Kvasov. Global search based on efficient diagonal partitions and a set of Lipschitz constants. *SIAM Journal on Optimization*, 16(3):910–937, 2006.

[189] Ya. D. Sergeyev and D. E. Kvasov. *Diagonal Global Optimization Methods*. FizMatLit, Moscow, 2008. in Russian.

[190] Ya. D. Sergeyev and D. E. Kvasov. A deterministic global optimization using smooth diagonal auxiliary functions. *Communications in Nonlinear Science and Numerical Simulation*, 21:99–111, 2015.

[191] Ya. D. Sergeyev and D. E. Kvasov. On deterministic diagonal methods for solving global optimization problems with Lipschitz gradients. In A. Migdalas and A. Karakitsiou, editors, *Optimization, Control, and Applications in the Information Age*, volume 130 of *Springer Proceedings in Matematics and Statistics*, pages 319–337. Springer, Switzerland, 2015.

[192] Ya. D. Sergeyev and D. E. Kvasov. *Deterministic Global Optimization: An Introduction to the Diagonal Approach*. SpringerBriefs in Optimization. Springer, New York, 2017.

[193] Ya. D. Sergeyev, D. E. Kvasov, and F. M. H. Khalaf. A one-dimensional local tuning algorithm for solving GO problems with partially defined constraints. *Optimization Letters*, 1(1):85–99, 2007.

[194] Ya. D. Sergeyev, D. E. Kvasov, and M. S. Mukhametzhanov. Comments upon the usage of derivatives in Lipschitz global optimization. In T. E. Simos, editor, *AIP Conference Proceedings*, volume 1738 (400004), 2016.

[195] Ya. D. Sergeyev, D. E. Kvasov, and M. S. Mukhametzhanov. On the least-squares fitting of data by sinusoids. In P. M. Pardalos, A. Zhigljavsky, and J. Žilinskas, editors, *Advances in Stochastic and Deterministic Global Optimization*, pages 209–226. Springer, 2016.

[196] Ya. D. Sergeyev, D. E. Kvasov, and M. S. Mukhametzhanov. Operational zones for global optimization algorithms. In *Proceedings of the XIII Global Optimization Workshop GOW'16, 4-8 September 2016, University of Minho, Braga, Portugal*, pages 85–88, 2016. ISBN 978-989-20-6764-3.

[197] Ya. D. Sergeyev, D. E. Kvasov, and M. S. Mukhametzhanov. Emmental-type GKLS-based multiextremal smooth test problems with non-linear constraints. In *Learning and Intelligent Optimization Conference (LION 2017), Lecture Notes in Computer Science*, volume 10556, pages 383–388. Springer, 2017.

[198] Ya. D. Sergeyev, D. E. Kvasov, and M. S. Mukhametzhanov. Operational zones for comparing metaheuristic and deterministic one-dimensional global optimization algorithms. *Mathematics and Computers in Simulation*, 141:96 – 109, 2017.

[199] Ya. D. Sergeyev, D. E. Kvasov, and M. S. Mukhametzhanov. Classes of multiextremal test problems with nonlinear constraints and known global solutions. 2018. submitted.

[200] Ya. D. Sergeyev, D. E. Kvasov, and M. S. Mukhametzhanov. On strong homogeneity of a class of global optimization algorithms working with infinite and infinitesimal scales. *Communications in Nonlinear Science and Numerical Simulation*, 59:319–330, 2018.

[201] Ya. D. Sergeyev, D. E. Kvasov, and M. S. Mukhametzhanov. On the efficiency of nature-inspired metaheuristics in expensive global optimization with limited budget. *Scientific Reports*, 8:453, 2018.

[202] Ya. D. Sergeyev, D. E. Kvasov, M. S. Mukhametzhanov, and A. De Franco. Acceleration techniques in the univariate Lipschitz global optimization. In Ya. D. Sergeyev, D. E. Kvasov, F. Dell'Accio, and M. S. Mukhametzhanov, editors, *AIP Conference Proceedings*, volume 1776 (090051), 2016.

[203] Ya. D. Sergeyev and D. L. Markin. An algorithm for solving global optimization problems with nonlinear constraints. *Journal of Global Optimization*, 7(4):407–419, 1995.

[204] Ya. D. Sergeyev, M. S. Mukhametzhanov, and D. E. Kvasov. Examples of solving ODEs given as a Black-Box on the Infinity Computer. In *Proceedings of the International Conference "New Trends in Numerical Analysis 2015 (NETNA 2015)", Falerna (CZ), Italy, 18-21 June 2015*, pages 90–91, 2015.

[205] Ya. D. Sergeyev, M. S. Mukhametzhanov, D. E. Kvasov, and D. Lera. Derivative-free local tuning and local improvement techniques embedded in the univariate global optimization. *Journal of Optimization Theory and Applications*, 171(1):186–208, 2016.

[206] Ya. D. Sergeyev, M. S. Mukhametzhanov, F. Mazzia, F. Iavernaro, and P. Amodio. Numerical methods for solving initial value problems on the Infinity Computer. *International Journal of Unconventional Computing*, 12(1):3–23, 2016.

[207] Ya. D. Sergeyev, R. G. Strongin, and D. Lera. *Introduction to Global Optimization Exploiting Space-Filling Curves*. SpringerBriefs in Optimization. Springer, New York, 2013.

[208] A. Shokri and A. A. Shokri. The new class of implicit l-stable hybrid obrechkoff method for the numerical solution of first order initial value problems. *Computer Physics Communications*, 184(3):529–531, 2013.

[209] N. Z. Shor. *Minimization methods for nondifferentiable functions*. Springer-Verlag, Berlin, 1985.

[210] B. O. Shubert. A sequential method seeking the global maximum of a function. *SIAM Journal on Numerical Analysis*, 9(3):379–388, 1972.

[211] C. P. Stephens and W. Baritompa. Global optimization requires global information. *Journal of Optimization Theory and Applications*, 96(3):575–588, 1998.

[212] R. G. Strongin. On the convergence of an algorithm for finding a global extremum. *Engineering Cybernetics*, 11:549–555, 1973.

[213] R. G. Strongin. *Numerical Methods in Multiextremal Problems (Information-Statistical Algorithms)*. Nauka, Moscow, 1978. in Russian.

[214] R. G. Strongin, V. P. Gergel, V. A. Grishagin, and K. A. Barkalov. *Parallel Computing for Global Optimization Problems*. Moscow University Press, Moscow, 2013. (In Russian).

[215] R. G. Strongin and Ya. D. Sergeyev. *Global Optimization with Non-Convex Constraints: Sequential and Parallel Algorithms*. Kluwer Academic Publishers, Dordrecht, 2000. 3rd ed. by Springer, 2014.

[216] M. Tawarmalani and N. V. Sahinidis. *Convexification and Global Optimization in Continuous and Mixed–Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications*. Kluwer Academic Publishers, Dordrecht, 2002.

[217] M. Van Daele and G. Vanden Berghe. P-stable obrechkoff methods of arbitrary order for second-order differential equations. *Numerical Algorithms*, 44(2):115–131, 2007.

[218] G. Vanden Berghe and M. Van Daele. Exponentially-fitted obrechkoff methods for second-order differential equations. *Applied Numerical Mathematics*, 59(3–4):815 – 829, 2009.

[219] M. C. Vita, S. De Bartolo, C. Fallico, and M. Veltri. Usage of infinitesimals in the Menger's Sponge model of porosity. *Applied Mathematics and Computation*, 218(16):8187–8196, 2012.

[220] V. Šátek, J. Kunovský, and A. Szöllös. Explicit and implicit Taylor series solutions of stiff systems. In F. Breitenecker and I. Troch, editors, *MathMod Vienna 2012 - 7th Vienna Conference on Mathematical Modelling*, 2012.

[221] P. Wolfe. A method of conjugate subgradients for minimizing nondifferentiable functions. In M.L. Balinski and P. Wolfe, editors, *Nondifferentiable optimization*, volume 3 of *Mathematical Programming Study*, pages 145–173. North-Holland, Amsterdam, 1975.

[222] X.-S. Yang. *Nature-inspired metaheuristic algorithms*. Luniver Press, Frome, 2008.

[223] X.-S. Yang. Firefly algorithms for multimodal optimization. In *Lecture Notes in Computer Science*, volume 5792, pages 169–178. Springer-Verlag Berlin, 2009.

[224] X.-S. Yang. *Engineering Optimization: An Introduction with Metaheuristic Applications*. Wiley, USA, 2010.

[225] X.-S. Yang and X. He. Firefly algorithm: Recent advances and applications. *International Journal of Swarm Intelligence*, 1(1):36–50, 2013.

[226] S. Yalçinbaş and M. Sezer. A Taylor collocation method for the approximate solution of general linear Fredholm–Volterra integro-difference equations with mixed argument. *Applied Mathematics and Computation*, 175:675–690, 2006.

[227] D. Zhao, Z. Wang, and Y. Dai. Importance of the first-order derivative formula in the obrechkoff method. *Computer Physics Communications*, 167(2):65–75, 2005.

[228] A. A. Zhigljavsky. Computing sums of conditionally convergent and divergent series using the concept of grossone. *Applied Mathematics and Computation*, 218(16):8064–8076, 2012.

[229] A. A. Zhigljavsky and A. Žilinskas. *Stochastic Global Optimization*. Springer, New York, 2008.

[230] A. Žilinskas. Optimization of one-dimensional multimodal functions: Algorithm AS 133. *Applied Statistics*, 23:367–375, 1978.

[231] A. Žilinskas. Axiomatic characterization of a global optimization algorithm and investigation of its search strategies. *Operations Research Letters*, 4:35–39, 1985.

[232] A. Žilinskas. On similarities between two models of global optimization: Statistical models and radial basis functions. *Journal of Global Optimization*, 48(1):173–182, 2010.

[233] A. Žilinskas. On strong homogeneity of two global optimization algorithms based on statistical models of multimodal objective functions. *Applied Mathematics and Computation*, 218:8131–8136, 2012.

[234] A. Žilinskas. A one-step worst-case optimal algorithm for bi-objective univariate optimization. *Optimization Letters*, 8(7):1945–1960, 2014.

[235] A. Žilinskas and J. Žilinskas. A hybrid global optimization algorithm for non-linear least squares regression. *Journal of Global Optimization*, 56(2):265–277, 2013.

Appendix

Appendix A: 20 univariate Lipschitz global optimization test problems used in numerical experiments.

#	Function $f(x)$	$[a,b]$	L	Global Minimizers
1	$\frac{1}{6}x^6 - \frac{52}{25}x^5 + \frac{39}{80}x^4 + \frac{71}{10}x^3 - \frac{79}{20}x^2 - x + \frac{1}{10}$	$[-1.5, 11]$	13870	$x^* = 10$
2	$\sin x + \sin\frac{10x}{3}$	$[2.7, 7.5]$	4.3	$x^* = 5.146$
3	$-\sum_{k=1}^{5} k\sin[(k+1)x + k]$	$[-10, 10]$	68.5	$x_1^* = -0.491$, $x_2^* = -6.775$, $x_3^* = 5.792$
4	$-(16x^2 - 24x + 5)\, e^{-x}$	$[1.9, 3.9]$	3.0	$x^* = 2.868$
5	$(3x - 1.4)\, \sin 18x$	$[0, 1.2]$	36.0	$x^* = 0.966$
6	$-(x + \sin x)\, e^{-x^2}$	$[-10, 10]$	2.5	$x^* = 0.680$
7	$\sin x + \sin\frac{10x}{3} + \ln x - 0.84x + 3$	$[2.7, 7.5]$	6.0	$x^* = 5.200$
8	$-\sum_{k=1}^{5} k\cos[(k+1)x + k]$	$[-10, 10]$	69.5	$x_1^* = -0.800$, $x_2^* = -7.084$, $x_3^* = 5.483$
9	$\sin x + \sin\frac{2x}{3}$	$[3.1, 20.4]$	1.7	$x^* = 17.040$
10	$-x\sin x$	$[0, 10]$	11.0	$x^* = 7.979$
11	$2\cos x + \cos 2x$	$[-1.57, 6.28]$	3.6	$x_1^* = 4.189$, $x_2^* = 2.094$
12	$\sin^3 x + \cos^3 x$	$[0, 6.28]$	2.2	$x_1^* = 4.712$, $x_2^* = 3.142$
13	$-x^{2/3} + (x^2 - 1)^{1/3}$	$[0.001, 0.99]$	8.5	$x^* = 0.707$
14	$-e^{-x}\sin 2\pi x$	$[0, 4]$	6.5	$x^* = 0.225$
15	$\frac{x^2 - 5x + 6}{x^2 + 1}$	$[-5, 5]$	6.5	$x^* = 2.414$
16	$2(x-3)^2 + e^{0.5x^2}$	$[-3, 3]$	294.1	$x^* = 1.591$
17	$x^6 - 15x^4 + 27x^2 + 250$	$[-4, 4]$	2520.0	$x_1^* = -3$, $x_2^* = 3$
18	$\begin{cases} (x-2)^2, & x \le 3 \\ 2\ln(x-2) + 1, & x > 3 \end{cases}$	$[0, 6]$	4.0	$x^* = 2$
19	$-x + \sin 3x - 1$	$[0, 6.5]$	4.1	$x^* = 5.873$
20	$(\sin x - x)\, e^{-x^2}$	$[-10, 10]$	1.3	$x^* = 1.195$

Appendix B: 12 initial value test problems used in numerical experiments.

#	ODE	y_0	y_n	solution	source
1	$y' = x - y$	1	0.735759	$y(x) = x - 1 + 2e^{-x}$	[3]
2	$y' = x + y$	1	3.436564	$y(x) = 2e^{x} - x - 1$	[3]
3	$y' = y$	1	2.718282	$y(x) = e^{x}$	[23]
4	$y' = 2y - e^{x}$	1	2.718282	$y(x) = e^{x}$	[23]
5	$y' = 2y(1 - 0.00001y)$	1	7.388584	$y(x) = 10^5 e^{2x}/(10^5 + e^{2x} - 1)$	[10]
6	$y' = -10y$	1	0.000045	$y(x) = e^{-10x}$	[61]
7	$y' = -8(y - 20)$	100	20.026837	$y(x) = 80e^{-8x} + 20$	[77]
8	$y' = -8(y - 15e^{-x/8} - 5)$	100	18.474329	$y(x) = \frac{1675}{21}e^{-8x} + \frac{320}{21}e^{-x/8} + 5$	[77]
9	$y' = \frac{y+x}{y-x}$	1	2.732051	$y(x) = x + \sqrt{1 + 2x^2}$	[23]
10	$y' = -y \cdot tan(x) - \frac{1}{cos(x)}$	1	-0.301169	$y(x) = \cos(x) - \sin(x)$	[23]
11	$y' = \frac{y-2xy^2}{1+x}$	1	1	$y(x) = \frac{1+x}{1+x^2}$	[23]
12	$y' = \frac{y-2xy^2}{1+x}$	0.4	0.571429	$y(x) = \frac{1+x}{2.5+x^2}$	[23]