An implementation of a Lipschitzian global optimization procedure

J.D. Pint'er

Department of Numerical Mathematics

NM-R9522 1995

Report NM-R9522
ISSN 0169-0388

CWI
P.O. Box 94079
1090 GB Amsterdam
The Netherlands

LGO: An Implementation of a Lipschitzian Global Optimization Procedure

User's Guide

János D. Pintér

Pintér Consulting Services
and
Dalhousie University

129 Glenforest Drive
Halifax, N.S., Canada B3M 1J2
e-mail: pinter@tuns.ca

Abstract

Decision problems are frequently modelled by optimizing the value of a primary objective function under stated feasibility constraints. Specifically, we shall consider here the following global optimization problem:

$$\min f(x) \quad \text{subject to} \quad x \in D \subset \mathbb{R}^n \ . \tag{GOP}$$

We shall assume that in (GOP) $f : D \to \mathbb{R}$ is a continuous function, and D is a bounded, robust subset ('body') in the Euclidean n-space. In addition, the Lipschitz-continuity of f on D will also be postulated, when necessary.
The above assumptions define a fairly general class of optimization problems, and typically reflect a paradigm in which a rather vaguely defined, 'large' search region is given on which a (potentially) multiextremal function f is minimized. It will also be assumed that the set of global solutions $X^* \subset D$ is, at most, countable.
To solve (GOP), a general family of adaptive partition strategies can be introduced: consult Pintér (1992a, 1995) and references therein. Necessary and sufficient convergence conditions can be established: these lead to a unified view of numerous GO algorithms, permitting their straightforward generalization and various extensions to handle specific cases of (GOP).
The present report discusses a Lipschitzian global optimization program system, for use in the workstation environment at CWI. Implementation aspects are detailed, numerical experience, existing and prospective applications are also highlighted. Application areas include, e.g., the following (Pintér, 1992b, 1995): general (Lipschitzian) nonlinear approximation, systems of nonlinear equations and inequalities, calibration (parameterization) of descriptive system models, data classification, general configuration design, aggregation of negotiated expert opinions, product/mixture design, 'black box' design and operation of engineering/environmental systems.

AMS Subject Classification (1991): 65K05, 90C30, 90C31.

Keywords & Phrases: Global optimization, Lipschitz-continuity, adaptive partition strategies, multiextremal optimization, implementation aspects.

Note: During the summer of 1995 the author was a visitor of the NW2-group at CWI. This research has been supported by the Dutch Technology Foundation (STW) under grant CWI55.3638.

1 Introduction

In the sciences, engineering and economics, decision problems are frequently modelled as optimizing the value of a (primary) objective (criterion, performance, loss etc.) function, under feasibility constraints to be met by all 'acceptable' decisions. To reflect this paradigm, the classical theory of constrained optimization, including also the body of mathematical programming, investigates problems of the generic form

$$\min f(x) \quad \text{subject to} \quad x \in D \subset I\!R^n \; . \tag{1.1}$$

The function f symbolizes the objective in the decision problem, and D denotes the set of all a priori admissible n-vectors x.

Note that, most typically, it is assumed that (1.1) has a unique optimal solution, corresponding to the unambiguously 'best' decision. Uniextremality is frequently postulated, or is implied by the mathematical problem structure (for example, by the strict convexity of the objective function f, and the convexity of the feasible set D). This generic optimization model form corresponds to the situation in which one, supposedly, has a sufficiently close initial 'guess' of the feasible (sub)region where the unknown optimal solution x^* is located; then x^* can be subsequently found by applying a suitable local search technique. Hence, the global optimality of the solution directly follows, having found the single local —and thus also global—optimum of the criterion function f on D.

Although many practically important optimization models naturally belong to this category, there is also a broad class of problems in which the property of uniextremality cannot be simply postulated or verified. Consider, for example, a system of nonlinear equations in which only rather vague ranges can be given with respect to the location of the solution(s). Then it may well happen that the system does not have a solution in the region specified, or it has a number of suboptimal 'solutions' (with corresponding non-zero residuals), besides one or several true solutions. In other cases—for example, in studying complex engineering or environmental systems—the system performance (objective) function f and/or some constraints defining the feasible region D are determined by running some computationally intensive numerical model or embedded algorithm, or by performing a set of experiments, Monte Carlo simulation cycles, etc. This way, the performance of each particular decision alternative needs to be evaluated by activating a large number of 'oracle' or 'black box' operations. One may refer, for example, to the calibration of descriptive physical, chemical, biological, ecological, etc. system models, the optimized 'tuning' of industrial or laboratory instruments and processes, and so on.

Frequently, in these or similar circumstances the underlying optimization problem does not possess the (verified) uniextremality property. In addition, the number of 'candidate' local optima is typically also unknown, and some of the locally optimal solutions may be very far—in terms of the criterion function f—from the best possible decision. Therefore, in such cases, it is quite natural to search for the global solution.

The emerging field of global optimization deals with the theory, implementation and application of models and strategies to solve multiextremal decision problems. Consult, for instance, Dixon and Szegö, (1975, 1978), Strongin (1978), Wilde (1978), Fedorov (1985), Pardalos and Rosen (1987), Van Laarhoven and Aarts (1987), Forgó (1988), Ratschek and Rokne (1988), Mockus (1989), Törn and Žilinskas (1989), Horst and Tuy (1990, 1993), Zhigljavsky (1991), Floudas and Pardalos (1992), Hansen (1992), Horst and Pardalos (1995), with extensive lists of additional references.

The variety of uniextremal optimization strategies (for example, to solve problems in linear programming, general convex minimization under linear constraints, nonlinear convex programming) is essentially based on assumptions concerning the underlying problem structure. Similarly, there exists a broad range of global optimization problem-types such as, for instance, concave minimization under linear or general convex constraints, (indefinite) quadratic programming, reverse convex programming, differential convex programming, fractional programming, (general) geometric programming, Lipschitz-continuous, and continuous global optimization. The underlying specific theory and the solution approaches to such diverse global optimization problems—as can be expected—also vary to a considerable extent.

Pintér (1995) provides a comprehensive discussion of a broad class of deterministic, direct (gradient-free) global optimization algorithms. These methods are based on adaptive, sequential partition and search in the feasible set D. The complete, 'exhaustive' search feature of such procedures guarantees their theoretical convergence to the global solution (set). In addition, a variety of fairly efficient numerical implementations can be realized on personal computers, workstations or modern mainframes, including parallel computers. The methodology discussed here can primarily be recommended to solve Lipschitzian and continuous global optimization problems, in which the lack of further analytical information realistically excludes the possibility of applying more specifically 'tailored' approaches (in a stand alone mode).

2 Problem Statement

We shall consider the following general global optimization problem (GOP) statement: given a bounded, robust set $D \subset \mathbb{R}^n$, and a continuous function $f : D \to \mathbb{R}$, solve

$$\min f(x) \quad \text{subject to} \quad x \in D \; . \tag{2.1}$$

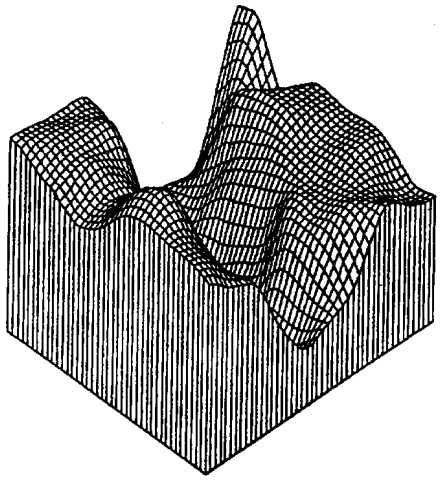

Figure 2.1: A multiextremal function in two variables

For illustration, see Figure 2.1, which displays a problem-instance of (2.1); in this example, D is simply a two-dimensional interval ('box'). In the figure one can observe several local minima of the objective function, and can immediately visualize the potential difficulty of the general problem (2.1).

Denote by X^* the set of solutions to the GOP (2.1); frequently, X^* consists only of a single point x^*, but this is not necessarily the case in general. In fact, it is easy to show instances of (2.1) in which X^* is finite, countable, non-countable, or even is a subset of D which has a positive volume. For the sake of algorithmic tractability, we shall assume that X^* is, at most, countable (enumerable).

Introducing the notation $f^* = f(x^*)$, the more precise verbal meaning of the problem statement (2.1) is the following:

$$\text{' find all elements of } X^* \text{ and the function value } f^* \text{ '}. \tag{2.2}$$

In numerous important cases, it is difficult—if not impossible—to solve the GOP (2.1) exactly: hence, the requirement (2.2) may be too 'ambitious'. Consequently, one has to accept diverse numerical approximations, for example, in the following sense:

find an $x \in D$ which satisfies the relation

$$\rho(x, X^*) := \min_{x^* \in X^*} \rho(x, x^*) \leq \delta \ , \tag{2.3}$$

in which ρ is a given distance function in $\mathbb{R}^n$ (typically defined by the norm introduced), and $\delta > 0$ is a fixed tolerance parameter; or

$$\begin{aligned} &\text{find an } x \in D \text{ which satisfies the relation} \\ &f(x) \leq f^* + \varepsilon \ , \end{aligned} \tag{2.4}$$

in which $\varepsilon > 0$ is another fixed tolerance parameter.

To ensure solvability in the sense of (2.4)—on the basis of a finite sample point sequence from D—we shall assume Lipschitz-continuity of f: in other words, the inequality

$$|f(x_1) - f(x_2)| \leq L\|x_1 - x_2\| \quad x_1, x_2 \in D \tag{2.5}$$

is postulated. In (2.5), $L = L(D, f)$ is the—typically unknown—Lipschitz-constant of f on D.

The problem (2.1), with the additional specification (2.5), is still very general and, in fact, it subsumes most (continuous) GOPs of practical relevance. As a consequence of this generality, it is also a very difficult numerical problem: it is easy to find GOP instances which pose a challenge in any computational environment available today, and tomorrow.

At the same time, the general framework and underlying convergence theory of adaptive partition algorithms in global optimization leave ample room for constructing robust and effective, computationally viable algorithms. The program system described below has been implemented (in gradually developed versions), extensively tested, and applied by the author in the past decade. This, of course, does not imply any notion of 'perfection': in fact, the program modules have often been (and surely will be) modified and improved, not infrequently in connection with new implementations and applications.

3 The LGO Program System

3.1 Scope of Application

The LGO program system serves for finding—that is, numerically approximating—the best solution or, theoretically, all solutions of the (possibly) multiextremal optimization problem in its standardized, box-constrained form

$$\begin{aligned} &\min f(x) \ , \\ &a \leq x \leq b \ . \end{aligned} \tag{3.1}$$

In (3.1), $a,\ x,\ b$ are finite real n-vectors, and f is Lipschitz-continuous on $[a, b]$. Note that a far more general class of Lipschitz-continuous optimization problems can be approximated in the form (3.1), following a penalty transformation based incorporation of a finite number of additional Lipschitzian constraints into the objective function. The penalty multipliers associated with these constraints can be adaptively changed, if necessary, to enforce (approximate) feasibility and optimality of the solution found. This and other solution strategies are also discussed by Pintér (1995), and references therein.

3.2 Problem Scaling

LGO performs self-scaling with respect to the search interval $[a, b]$ which is automatically transformed into $[0, 1]^n$. This transformation helps a more 'uniform', scale-free screening of the feasible set. At the same time—depending on the original problem structure—it often may lead to very 'steep' objective functions on $[0, 1]^n$, and hence, to quite huge Lipschitz-constants. (Note that, of course, the initial problem may already display such characteristics.) To avoid related numerical problems during the solution procedure, it is advisable to scale also the objective function. This

can be easily done, applying a suitable multiplier which transforms the function values, say, into the range of $[-100, 100]$. Some prior experimentation (argument and function value sampling) may be advisable in connection with complicated problems, since appropriate scaling may indeed lead to improved numerical performance: LGO supports such experiments, when applied in an interactive run mode.

3.3 Solution Methodology

The current program system includes, in an integrated fashion, the following optimization options:

- Lipschitzian global optimization (by adaptive partition and search) on $[a, b]$
- global optimization by pure random search on $[a, b]$
- local optimization launched from the current estimate of the global optimum (Fletcher-Reeves-Polak-Ribière method, FRPR)
- local optimization launched from the current estimate of the global optimum (Powell-Brent method, PB).

A few comments related to these solvers are in order: for technical details, consult Pintér (1995), and the appropriate references therein. As known, pure random search is a very simple, 'folklore' approach to global optimization, which has been used extensively. In the present context, it can be applied, for example, by novice users who want to 'explore' the feasible set, or as a preliminary search phase for possibly reducing the initially chosen search region. The Lipschitzian solver is far more sophisticated, and—enabled with proper parameterization—it should serve to generate a reasonable approximation of the global optimizer point(s), before the LGO system 'switches' to local search. Among the two local methods included, FRPR relies on gradient information and, for sufficiently 'smooth' problems, it often assures a rapid termination of the search. PB is gradient-free, and is somewhat more 'cautious', but—according to computational experience—is also more robust, than FRPR.

Note that both of the local search algorithms FRPR and PB are discussed, for example, by Press, Teukolsky, Vetterling and Flannery (1992), with further references therein. For the present implementation, FRPR and PB were both modified and adapted, to assure their proper integration with the global optimization search phase.

The listed solver options can be activated either in an interactive fashion (user control mode), or can be launched by the LGO system (automatic mode). In the first case, the user can select a solver, and then specify the computational effort to be spent by the chosen option; following a sequence of optional solver sessions, the run can be terminated. In automatically controlled runs, a Lipschitzian global optimization phase is followed by a PB local refinement. Additional options to solve (3.1) are under development.

3.4 Structure of LGO Application Programs

The following abbreviations will be used (they reflect a generic FORTRAN development environment on workstations):

NAME	- name of the application, given by user
NAME-DR.F	- source code of the main driver, written/adapted by user
NAME-OF.F	- objective function file, written/adapted by user
NAME.IN	- input parameter file, written/adapted by user
USERSRC.F	- additional source code provided by user
LGOSBRS.O	- object file of LGO program system routines
NAME.OUT	- detailed output file generated by LGO
NAME.SUM	- summary output file generated by LGO.

Commented templates of the files NAME-DR.F, NAME-OF.F, and NAME.IN are provided, to assist users (please see Appendices I and II). This way, the setup of the application-dependent files is quite straightforward, and does not require special skills or intimate knowledge, related to the LGO system. Of course, the parameterization of LGO in NAME.IN profits from an understanding of the basic principles of the optimizer, but sample parameter values and commentary on parameter choices are also provided in the template input file.

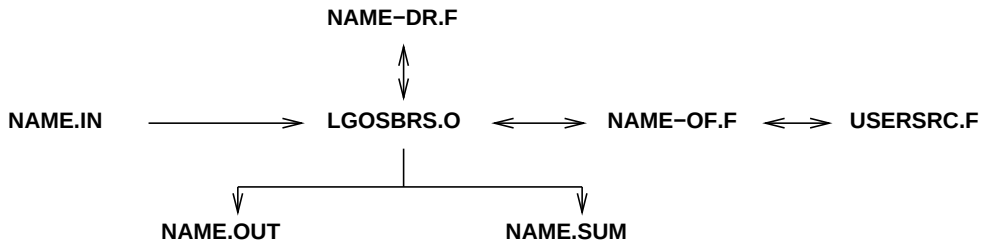

Figure 3.1: Basic structure of LGO application program systems

Figure 3.1 shows the principal structure of LGO application programs. A brief description of the program structure, outlined by Figure 3.1, follows. The main driver file NAME-DR.F serves for the (optional) printout of application information; it opens the (application-specific) input parameter file NAME.IN and output files NAME.OUT and NAME.SUM, and then calls the LGO system. On return, the results stored in NAME.OUT and NAME.SUM can be additionally reported in a form specified by the user, and other tasks—analysis of the results, repeated program runs, etc.—can also be initiated.

LGOSBRS.O is the core of the optimizer: it integrates all subroutines and functions which are necessary to operate the solver variants. Its operations are governed by the input parameter file NAME.IN. In this file the lower and upper bounds (vectors a and b) of the optimization problem (3.1) are defined, followed by the parameterization of the LGO program system. This structure facilitates rerunning the same executable program under different feasible range and/or LGO system parameter specifications.

During the optimization run, LGOSBRS.O iteratively calls the (compiled) objective function segment NAME-OF.F which—for each parameter vector x chosen by LGO—returns the corresponding function value. Frequently—for example, in solving nonlinear systems of equations by global optimization—NAME-OF.F is simply a programmed 'translation' of an explicitly given numerical problem. In other cases—notably, in 'black box' system optimization—the objective function value is calculated via running an attached program system, denoted here by USERSRC.F. For instance, USERSRC.F may be the descriptive model of a complex industrial, engineering, ecological, etc. system which is to be parameterized (calibrated) applying global optimization. Note that—for all typical software development purposes—NAME-DR.F and NAME-OF.F can be given in the same (ASCII text) file: this applies also to USERSRC.F when present. Observe also that the USERSRC program (system) can be also attached, as an object file ('black box'), if proper communication with NAME-OF.F is ensured.

Two output files are generated by LGO, in a complete optimization run. NAME.OUT is a more detailed result file: the level of detail is controlled by a parameter choice in the input file NAME.IN. Typically, NAME.OUT can be most useful during the development of an application, in which the problem statement is verified and the optimization process is traced. Another application can be related to very difficult or poorly defined problems, in which it is informative to inspect a variety of feasible solutions generated by LGO.

The file NAME.SUM contains the summary results of the optimization run. It is useful in 'routine' LGO runs, since it provides concise information, without the need of browsing a possibly quite large amount of additional details in NAME.OUT.

4 Current System Requirements and Problem Size Limitations

4.1 Hardware and Software

In Pintér (1995), details of implementing LGO in a PC environment have been discussed. The present implementation is targeted for the workstations available at CWI: therefore a detailed discussion of the hardware requirements is not necessary. For similar reasons, it is only mentioned here that, in general, a professional FORTRAN development environment is a prerequisite. The FORTRAN 77 system, available at CWI, provides an adequate work environment.

4.2 Problem Size Limitations

For reasons of keeping the program storage requirements (and runtimes) within reasonable bounds, the following limitations were introduced:

- The present (explicitly declared) array structure in LGO supports the formulation and solution of problems up to 50 variables.
- Maximum 10,000 (currently active) partition subintervals can be simultaneously present in the Lipschitz global optimization mode.

According to our experience, these bounds are not too restrictive with respect to many applications. Note also that the above limitations can be easily relaxed, but memory requirements and computational times may (will) correspondingly increase.

5 Using LGO in an Interactive Environment

The current PC version of the program system LGO can be activated and used in an interactive fashion, using a menu-based DOS interface. (A Windows version is currently under development, and will also be available soon.) The following information refers to the DOS version, with a view towards a similar development on the workstation platform.

Typing 'LGO %1' (%1 stands for the application filename given by the user) at, say, the C:\LGO prompt, a menu shell is activated, which shows the following screen:

LIPSCHITZIAN GLOBAL OPTIMIZATION
PROGRAM SYSTEM
Version 3.1

©Janos D. Pinter, 1986...1995

Application Menu (Options)	**Option Symbols**
Introduction (General Information)	**I**
Demonstration Program Information	**D**
Edit Optimization Input File(s)	**E %1**
Compile Optimization Program Files	**C %1**
Run Optimization Program System	**R %1**
View Optimization Output File(s)	**V %1**
Graphical Analysis of Results	**G %1**
Quit (Return to MS-DOS)	**Q**

Please select an option, and press ENTER.
(If you are not familiar with LGO, choose first I, and then D.)

For instance, typing 'LGO TEST2' when invoking this menu initiates an interactive session, for running a variety of two-dimensional test problems (on PC, given in the file TEST2.FOR) which can he modified and adapted by the user.

Note that it would be quite straightforward to develop a similar interface also for the workstation environment, but this was not done (due to lack of time). It is obvious, however, that embedding LGO into the FORTRAN 77 environment, and using an appropriate ASCII editor (such as, e.g., EMACS or vi), the development of a simple menu interface should be a straightforward task. Of course, the underlying implementation and activation of the options does depend on that environment. For instance, graphics options can be programmed in FORTRAN, or some other suitable environment with graphics capabilities (Maple, Mathematica, Matlab, etc.) can be invoked.

To assist a possible menu-driven implementation, a brief summary of the PC menu options follows.

Introduction (General Information)

The selection of 'I' in the main menu launches a brief information session. This provides a concise description related to multivariate global optimization of continuous and Lipschitz functions on interval regions (for novice users). The CGO/LGO problem class is defined; the graph of a univariate multiextremal function f is shown, indicating local and global optima. This is followed by a summary of the solution approach applied in LGO, and a list of several existing and prospective applications. Finally, basic information related to running the LGO program system is provided. With respect to program names, the following conventions are applied (recall Section 3.4 and observe some slight modifications):

filename	-	name of the application given by the user;
filename.F	-	source code of the main driver and objective function file, prepared by user;
filename.IN	-	input parameter file (definition of bounds $[a, b]$, and parameterization of program system), this file is also prepared by the user;
filename.OUT	-	detailed output file generated by the LGO program system;
filename.SUM	-	summary output file generated by LGO.

At CWI, templates and sample programs are provided for the .F and .IN files, under the names LGOCWI.F and LGOCWI.IN (please see Appendices I and II).

Upon completing the information session, the program returns to the main (applications) menu.

Demonstration Program Information

Selecting 'D' in the main menu, information related to running TEST2 is provided. In any single run, the TEST2 program can be executed, to minimize one of a given set of two-variate multiextremal test functions. The form of these functions can be modified by the user in the file TEST2.F; the parameterization of the optimizer system can also be modified, in the file TEST2.IN. Both TEST2.F and TEST2.IN are provided; they can be edited, the program can be compiled and run, and the output can be analysed, as indicated by the main menu. For instance, typing 'V TEST2'—after running the TEST2 program—enables the browsing and analysis of the results.

Note that user applications can also be written and then run, making use of the templates provided by the files TEST2.F and TEST2.IN.

At the end of the demonstration program information session, the main menu is displayed again.

Edit Optimization Input Files

Selecting 'E filename' in the main menu, the optimization input files (filename.F and filename.IN) specified by the user are invoked into a text editor. (In the LGO system on the PC, the IBM editor PE is called, but other simple ASCII editors could also be used.) At this point, the user can check and modify the optimization input files: as mentioned above, the sample files TEST2.F and TEST2.IN can also serve as a basis for preparing new files. After the editing session, the program returns to the main menu.

Compile Optimization Program Files

Selecting 'C filename' in the main menu, compilation of the user-dependent source code (filename.F file) follows. The actual realization of this step obviously depends on the programming environment available to the user (that is, it needs to be executed, depending on the available FORTRAN compiler). In case of compilation and/or linking errors, corresponding error messages appear, and the menu program prompts the user to return to the editing option. If the program compilation was successful, then—communicating this message to the user—control is returned to the main menu level.

Run Optimization Program System

Having a successfully compiled and linked code, one can select 'R filename' in the main menu. This invokes the runtime environment, and the program will be executed (again, according to the FORTRAN system available to the user). After the optimization run is terminated (succesfully, or with runtime error messages), the system returns again to the main menu: repeated editing and/or analysis of results may then follow.

View Optimization Output Files

Following a (formally) successful optimization run, one can invoke the result analysis (browsing) mode, typing 'V filename'. The optimization output (filename.SUM and filename.OUT) files are called into the built-in ASCII editor, and they can be viewed or possibly edited (for purposes of subsequent use and/or reporting). Following the analysis of output files, control is returned to the main menu level.

Graphical Analysis of Results

Following an optimization run—again depending on the available environment—a graphical summary of the results can be activated by typing 'G filename'. This step typically needs the (automatic) loading of the output file(s) filename.OUT and filename.SUM into the corresponding graphical environment. For instance, the sample points generated by the optimizer (their two-dimensional projection into a specified subspace), or a graph of the best function value found versus the actual number of steps, etc., may follow. After inspecting these graphs, control is returned again to the main menu.

Quit

Typing 'Q' at the main menu, the LGO program system application is terminated, and the program returns to the operating system (DOS) from which LGO was called.

6 Illustrative Examples

To illustrate the use of the LGO program system, the solution of a 50-variable test problem of the form (6.1) is discussed below. For additional test examples—and details on a number of

applications—consult Pintér (1995), and references therein.

Example 1

In Example 1, the objective function f is defined as follows.

$$\begin{aligned} f(x) &= f_1(x) + f_2(x) + f_3(x)\ , \quad \text{in which} \\ f_1(x) &= c\sum_{i=1}^{n} i x_i^2 \qquad (c > 0) \\ f_2(x) &= \sum_{i=1}^{n} \left(x_{i-1} + 5\sin x_i + x_{i+1}^2\right)^2 \\ f_3(x) &= \sum_{i=1}^{n} \ln^2\left(1 + \mid i\sin^2 x_{i-1} + 2x_i + 3x_{i+1} \mid\right)\ . \end{aligned} \tag{6.1}$$

By definition, in (6.1) $x_0 = x_n$ and $x_{n+1} = x_1$; $\ln(\cdot)$ denotes the natural logarithm function. Moreover, $c > 0$ is a given parameter.

Let us make a few comments related to the function (6.1). The first sum, f_1, consists of the simplest possible quadratic terms. Nevertheless—due to the multipliers i of the individual terms, in case of large n—the resulting sum will become 'badly scaled': specifically, the rate of the largest and smallest axes in the ellipsoid level sets of the function f_1 equals $\sqrt{n}$. Note also that component f_1 serves to model the situation in which the 'far side' of the 'vaguely defined' search region $[a, b]$ contains only inferior solutions: this is typical in many well-posed global optimization problems of the form (3.1). Observe also the 'orientation' effect provided by f_1 towards the unique global solution $x^* = 0$, which is scaled by the parameter $c > 0$: the larger c is, the 'easier' the GOP (6.1) is likely to become.

The two other component functions of f (the sums f_2 and f_3) model a noise structure superimposed on f_1, leading to a multiextremal function f in n variables. The function f_2 is a combination of embedded linear, trigonometric and quadratic terms, in simple quadratic function addenda. The oscillation amplitude of the trigonometric term is magnified by its multiplier factor. Finally, the function f_3 is made up by logarithmic terms, with embedded trigonometric and linear arguments. All terms in f_2 and f_3 depend on three subsequent components of the vector x. One can also observe that the noise terms become relatively more significant in a closer neighbourhood of the optimal solution, especially when c is small. (In the numerical example we chose c equal to 1.) In summary, although function f defined by (6.1) is not very complicated, it surely has somewhat 'tricky' nonlinearities and multiextremal structure, and in case of a large initial region of uncertainty (given by $[a, b]$) most local scope methods would probably have difficulties in finding the global solution—the zero vector.

Appendix I shows the corresponding main program and objective function file. In Appendix II the input parameter file is given. In the test example, we chose $n = 50$; the components of the upper and lower bound vectors given there show that the search region is relatively large. Appendix III provides a summary of the results of an LGO run (displaying the LGOCWI.SUM output file).

Example 2

As a second example we included the following problem, which has many local maxima and saddle points. In Example 2 we are interested in the global maximum of the objective function f, which is defined as follows.

$$\begin{aligned} f(x) &= \prod_{1 \le i < j \le n} \| x_i - x_j \|\ , \\ \text{s.t.} \| x_i \| &= 1, \quad i \in \{1, \ldots, n\}\ , \end{aligned} \tag{6.2}$$

with $x_i \in I\!R^3$, for all $i \in \{1, \ldots, n\}$, $\| \ . \ \|$ indicates the Euclidian distance. The points $x_1, \ldots, x_n$ which give the global maximum of (6.2) are called the elliptic Fekete points (of order n). See Stortelder and Pintér (1995) for a more extensive description of this problem.

In Appendix IV, illustrative results for $n = 13$ points are shown. The points on the unit sphere, x_i, are transformed to spherical coordinates: $x_i = (\cos\alpha_i \sin\beta_i, \sin\alpha_i \sin\beta_i, \cos\beta_i)$, with $0 \leq \alpha_i < 2\pi$ and $0 \leq \beta_i < \pi, \ i \in \{1, \ldots, 13\}$. Because of rotation symmetries three angles can be chosen a priori; we chose $\alpha_1 = \beta_1 = \alpha_2 = 0$, the decision variables consist of the remaining 23 angles.

7 Concluding Remarks

Global optimization is a challenging field in mathematical programming and its applications. The present report has provided a very brief discussion of the GOP, with an emphasis on (general) continuous and Lipschitzian GOPs. An implementation of LGO, a procedure to solve such problems, is described. Illustrative test results are also included, together with other implementation specific information.

Numerous existing and prospective applications are discussed and highlighted by Pintér (1995), where an extensive list of additional references can also be found.

Acknowledgements

I wish to express my thanks to the Dutch Technology Foundation (STW) and CWI, especially to Walter Stortelder and Pieter Hemker, for the kind invitation and hospitality during my stay in Amsterdam. I also thank both of them for helpful comments during the preparation of this report, and for assistance in the implementation work.

References

Dixon, L.C.W., and Szegö, G.P., eds. (1975, 1978). *Towards Global Optimisation.* Vols. 1-2. North-Holland, Amsterdam.

Fedorov, V.V., ed. (1985). *Problems of Cybernetics: Models and Methods in Global Optimization.* USSR Academy of Sciences, Moscow. (In Russian.)

Floudas, C.A., and Pardalos, P.M., eds. (1992). *Recent Advances in Global Optimization.* Princeton University Press.

Forgó, F. (1988). *Nonconvex Programming.* Akadémiai Kiadó, Budapest.

Hansen, E.R. (1992). *Global Optimization Using Interval Analysis.* Marcel Dekker, New York.

Horst, R., and Pardalos, P.M., eds. (1995). *Handbook of Global Optimization.* Kluwer, Dordrecht.

Horst, R., and Tuy, H. (1990). *Global Optimization: Deterministic Approaches.* Springer, Berlin. (2nd edn., 1993.)

Mockus, J. (1989). *Bayesian Approach to Global Optimization.* Kluwer, Dordrecht.

Pardalos, P.M., and Rosen, J.B. (1987). *Constrained Global Optimization: Algorithms and Applications. Lecture Notes in Computer Science* 268. Springer, Berlin.

Pintér, J. (1992a). Convergence qualification of partition algorithms in global optimization. *Mathematical Programming 56*, pp. 343-360.

Pintér, J. (1992b). Lipschitzian global optimization: Some prospective applications. In: *Recent Advances in Global Optimization*, (eds. C.A. Floudas and P.M. Pardalos), pp. 399-432. Princeton University Press.

Pintér, J. (1995). *Global Optimization in Action.* Kluwer, Dordrecht. (In press.)

Press, W.H., Teukolsky, S.A., Vetterling, W.T., and Flannery, B. (1992). *Numerical Recipes in FORTRAN: The Art of Scientific Computing,* Cambridge University Press.

Ratschek, H., and Rokne, J.G. (1988). *New Computer Methods for Global Optimization.* Ellis Horwood, Chichester.

Stortelder, W., and Pintér, J. (1995). Numerical Approximation of Elliptic Fekete Point Sets: A Global Optimization Approach, In: *Proceedings of the Third Workshop on Global Optimization*, Szeged, Hungary, December, 1995 (to be published).

Strongin, R.G. (1978). *Numerical Methods for Multiextremal Problems.* Nauka, Moscow. (In Russian.)

Wilde, D.J. (1978). *Globally Optimal Design.* Wiley Interscience, New York.

Törn, A.A., and Žilinskas, A. (1989). *Global Optimization. Lecture Notes in Computer Science 350.* Springer, Berlin.

Van Laarhoven, P.J.M., and Aarts, E.H.L. (1987). *Simulated Annealing: Theory and Applications.* Kluwer, Dordrecht.

Zhigljavsky, A.A. (1991). *Theory of Global Random Search*, (ed. J. Pintér). Kluwer, Dordrecht.

Appendix I

Source Code of User Program Segments of Example 1

Note: This file—called lgocwi.f, and located in the directory `/ufs/walterst/local/progf/LGO`—can serve as a simple frame (template) to write new applications.

```
c     LGOCWI.F FILE

c     J. Pinter
c     July 25, 1995

c     Main program and objective function shells
c     provided for users of global optimization program system

c     Workstation version
c     built-in timer
       real dtime, tarray(2),ttotal
       external dtime


c     SOLUTION OF ILLUSTRATIVE TEST PROBLEMS (see CWI report for details)
c     The example test function is defined in the segment FCT below

c     OPEN INPUT AND OUTPUT FILES
c     Note:
c     Usage of logical unit numbers 8, 9 and 19 is mandatory;
c     they can be changed only by modifying the optimization
c     program system (provided as an object file)
c     The names themselves can be changed

      OPEN(8,FILE='lgocwi.in',STATUS='OLD')
c     lgocwi.in serves for parameterization of the global optimization
c     driver (i.e., this main) program

      OPEN(9,FILE='lgocwi.out',STATUS='UNKNOWN')
c     lgocwi.out serves for (optional) post-run analysis: it contains the
c     principal information describing the problem, and the main findings
c     obtained during the optimization procedure

      OPEN(19,FILE='lgocwi.sum',STATUS='UNKNOWN')
c     lgocwi.sum contains a brief summary of the run results

c     CALL TIMER
c     at beginning of program execution, timer initialized
       ttotal=dtime(tarray)

c     CALL OPTIMIZATION PROGRAM SYSTEM MAIN DRIVER ROUTINE
       call OPTDRIVER

c     CALL TIMER
c     at end of program execution, timer checked again
       ttotal=dtime(tarray)
       write(9,99) tarray(1), tarray(2), ttotal
       write(19,99) tarray(1), tarray(2), ttotal
99    format (///,5x,'user time=',e12.4
     +                /,5x,'system time=',e12.4
     +               /,5x,'total time=',e12.4)

      stop
      end


      FUNCTION FCT(X)
```

```
c     Problem-specific objective function - defined by user - follows here

c     BASIC DECLARATIONS FOR (TEST) PROBLEMS
c     Note: These all need to be included, as they are


c     Maximal number of decision variables
      parameter(maxdim=50)

c     Range and shift of decision variables (in original problem)
      common/scale/ range(maxdim),aorg(maxdim)

c     The vector xtrf corresponds to the decision (optimization) variable x,
c     expressed in the original scale of [a,b]
      dimension x(maxdim),xtrf(maxdim)

c     Actual number of decision variables (can be defined externally)
      common/decvars/nact

c     Actual number of variables is explicitly defined below (for CWI test runs)
      nact=50

c     TRANSFORMATION OF OPTIMIZED DECISION VARIABLES
C     [0,1]^n -> [a,b] ("back to original scale of optimization problem")
      do 1 i=1,nact
1     xtrf(i)=aorg(i)+x(i)*range(i)


c     DEFINITION OF TEST FUNCTION (for CWI User Guide)

C     Main (orientation) term: very simple, but badly scaled quadratic function
      fct1=0.
      do 2 i=1,nact
 2    fct1=fct1+float(i)*xtrf(i)**2

c     Basic function value fct1 (of 'orientation' term) can be scaled.
c     Consequently, the added 'noise terms' are more/less dominant, and
c     make it more/less difficult to solve the test problem
c     Example 1: c1=1. (corresponds to original scale)
      c1=1.
      fct1=c1*fct1

c     Noise terms follow, implying multiextremality of test function
      fct2=0.
      fct3=0.
      do 11 i=1,nact
       i1=i-1
       if(i1-1)111,112,112
 111    i1=nact
 112   i2=i+1
       if(i2-nact)114,114,113
 113    i2=1
 114   fct2=fct2+(xtrf(i1)+5.*sin(xtrf(i))+xtrf(i2)**2)**2
       aux=1.+abs(float(i)*(sin(xtrf(i1)))**2+2.*xtrf(i)+3.*xtrf(i2))
       fct3=fct3+(alog(aux))**2
 11    continue
      fct=fct1+fct2+fct3

c     SCALING
c     can be included, to avoid very 'steep' changes in objfct value
      scal=0.01
      fct=fct*scal

      return
      end
```

Appendix II

Input Parameter File of Example 1

Note: This file—located in the directory `/ufs/walterst/local/progf/LGO`, and called lgocwi.in—can serve as a simple frame (template) to write new applications.

```
c     LGOCWI.IN FILE    for 50-variable test problem
c     INPUT DATA TO LIPSCHITZIAN GLOBAL OPTIMIZATION PROGRAM SYSTEM
c     NOTE: THE STRUCTURE OF THIS FILE IS TO BE FOLLOWED EXACTLY;
c     PLEASE LEAVE ALL COMMENT LINES AND KEEP THE INPUT DATA FORMATS!

c     DIMENSIONALITY AND INTERVAL FEASIBLE REGION IN OPTIMIZATION PROCEDURE
         n=     50                        *  (12x,i15)
       a(1)=    -8.8                      * 50-variable test problem;
       b(1)=     1.4                      * data begin: ranges of
       a(2)=    -6.2                      * decision variables (12x,f15.6)
       b(2)=     0.9                      *
       a(3)=    -8.7                      *
       b(3)=     1.7                      *
       a(4)=    -7.7                      *
       b(4)=     0.8                      *
       a(5)=    -3.2                      *
       b(5)=     5.3                      *
       a(6)=    -3.5                      *
       b(6)=     7.9                      *
       a(7)=    -5.1                      *
       b(7)=     8.7                      *
       a(8)=    -2.2                      *
       b(8)=     4.7                      *
       a(9)=    -9.1                      *
       b(9)=     3.8                      *
       a(10)=   -6.3                      *
       b(10)=    1.7                      *
       a(11)=   -7.8                      *
       b(11=     3.2                      *
       a(12)=   -5.2                      *
       b(12)=    3.9                      *
       a(13)=   -6.1                      *
       b(13)=    1.8                      *
       a(14)=   -2.7                      *
       b(14)=    4.2                      *
       a(15)=   -5.6                      *
       b(15)=    3.3                      *
       a(16)=   -7.1                      *
       b(16)=    2.9                      *
       a(17)=   -2.1                      *
       b(17)=    6.7                      *
       a(18)=   -5.2                      *
       b(18)=    3.7                      *
       a(19)=   -4.1                      *
       b(19)=    2.8                      *
       a(20)=   -7.3                      *
       b(20)=    4.7                      *
       a(21)=   -8.5                      *
       b(21)=    7.2                      *
       a(22)=   -1.2                      *
       b(22)=    4.9                      *
       a(23)=   -5.7                      *
       b(23)=    3.5                      *
       a(24)=   -7.7                      *
       b(24)=    1.5                      *
       a(25)=   -8.6                      *
       b(25)=    5.3                      *
```

```
      a(26)=   -9.5                     *
      b(26)=    6.8                     *
      a(27)=   -5.1                     *
      b(27)=    3.7                     *
      a(28)=   -6.7                     *
      b(28)=    1.7                     *
      a(29)=   -4.1                     *
      b(29)=    1.8                     *
      a(30)=   -4.3                     *
      b(30)=    6.7                     *
      a(31=    -3.5                     *
      b(31)=    1.9                     *
      a(32)=   -6.2                     *
      b(32)=    3.9                     *
      a(33)=   -7.1                     *
      b(33)=    3.5                     *
      a(34)=   -7.7                     *
      b(34)=    4.8                     *
      a(35)=   -5.6                     *
      b(35)=    2.3                     *
      a(36)=   -6.5                     *
      b(36)=    2.8                     *
      a(37)=   -5.1                     *
      b(37)=    8.7                     *
      a(38)=   -3.2                     *
      b(38)=    1.7                     *
      a(39)=   -5.1                     *
      b(39)=    1.8                     *
      a(40)=   -3.3                     *
      b(40)=    7.7                     *
      a(41)=   -5.5                     *
      b(41)=    2.2                     *
      a(42)=   -3.2                     *
      b(42)=    4.9                     *
      a(43)=   -4.3                     *
      b(43)=    7.8                     *
      a(44)=   -4.7                     *
      b(44)=    2.5                     *
      a(45)=   -3.6                     *
      b(45)=    8.3                     *
      a(46)=   -4.5                     *
      b(46)=    1.9                     *
      a(47)=   -4.1                     *
      b(47)=    1.7                     *
      a(48)=   -7.2                     *
      b(48)=    3.2                     *
      a(49)=   -4.1                     *
      b(49)=    1.8                     *
      a(50)=   -5.3                     * 50-variable test problem data:
      b(50)=    1.3                     *    end of range definitions

c     SPECIFICATION OF OPTIMIZATION PROGRAM PARAMETERS

c     GLOBAL SEARCH PHASE PARAMETERS
c     global search termination criterion parameter acctr:
c     solution found in global search phase is better than acctr
       acctr= 1.                  ******** change, if necessary! *********
c     global search termination criterion parameter :
c     best approximate solution - Lipschitz bound estimate < eps
       eps=   10.                 ******** change, if necessary! *********
c     global search termination criterion parameter :
c     L1-norm (sum of edges) of selected partition subinterval < delt
       delt=  1.e-3               ******** change, if necessary! *********
c     global search termination criterion parameter :
c     maximal no. of major global iterations (subinterval partition cycles)
       maxit= 50000               ******** change, if necessary! *********
c     global search termination criterion parameter :
```

```
c    maximal no. of objfct evaluations in global search phase
      maxfct= 20000               ******** change, if necessary! *********
c    no. of alternating global/local search cycles
c    in the present realization, ngphas=1 is the recommended option
      ngphas= 1                   ******* leave, as it is! ***************
c    global search termination criterion parameter :
c    if in actual global search phase optimum estimate is improved, then
c    after kloc steps local search phase follows
      kloc= 10000000              ******** change, if necessary! *********
c    global search termination criterion parameter :
c    if in actual global search phase no improvement is attained in nosuc
c    consecutive steps,then the optimization procedure is terminated
c    (after local search phase)
      nosuc= 10000000             ******** change, if necessary! *********
c    global search termination criterion parameter :
c    if the no. of (currently stored) subintervals attains limsto, then
c    the optimization procedure is terminated (after local search phase)
      limsto= 10000               ******* leave, as it is! ***************
c    multiplier in EOS based local Lipschitz-constant estimates (saf>1)
c    N.B. larger saf increases safety, but decreases search efficiency
      saf=  2.0                   ******** increase, if necessary! *********
c    restart option: range contraction parameter crate;  0 < crate <= 1
c    small/large crate -> fast/thorough search; crate=1 -> 'exact' LGO search
      crate= 1.                   ******** change, if necessary! *********
c    THESE GLOBAL SEARCH PARAMETERS ARE PROBLEM-DEPENDENT
c    (CHECK RUNS AND MODIFY THEM IF NECESSARY!)

c    LOCAL SEARCH PARAMETERS (FLETCHER-REEVES-POLAK-RIBIERE ALGORITHM)
c    improved/modified conjugate directions method (JP)
c    prior lower estimate ('guess'!) of optimum value
      est=  0.                    ******** change, if necessary! ***********
c    initialized value of linear search accuracy
      hmin= 1.e-4                 ******** increase, if necessary! *********
c    stopping criterion: threshold value of objfct improvement
      epf=  1.e-6                 ******** increase, if necessary! *********
c    stopping criterion: threshold value of gradient norm square
      epgg= 1.e-8                 ******** increase, if necessary! *********
c    stopping criterion: maximal number of major conjugate directions
c    iteration cycles
      icmx= 10000                 ******* leave, as it is! *****************
c    stopping criterion: maximal number of fct+grad evaluations
      ifmx= 20000                 ******* increase, if necessary ***********
c    note: expected maximal no. of function evaluations in local search
c    phase equals ifmx*(2*n+1)
c    THESE LOCAL SEARCH PARAMETERS ARE PROBLEM-DEPENDENT
c    (CHECK RUNS AND MODIFY THEM IF NECESSARY!)

c    PRINTOUT OPTIONS IN OPTIMIZATION PROCEDURE
c    select printout option indicators ipr,ipopt
c    ipr=1 ipopt=0: detailed runtime printout (for test runs)
c    ipr=0 ipopt=1: runtime printout of improving solutions (suggested option)
c    ipr=0 ipopt=0: only basic and final printouts (for standardized runs)
      ipr=0 ipopt=1               ******** change, if necessary! ***********

c    FINITE DIFFERENCE APPPROXIMATION OF GRADIENTS: relative componentwise
c    step-size stp (in FUNCT), following self-scaling (in FCT);
c    1.e-10 < stp < 1.e-2 is recommended
      stp= 1.e-6                  ******** change, if necessary! ***********
c    biggr: safety factor to avoid possible overflow
c    suggested value: float(n)*10**2
      biggr= 50000.               ******** decrease, if necessary! *********

c    PARAMETERS OF RANDOMIZED SAMPLING
c    number of sample points per subinterval (nr>=2; e.g. nr=2*n recommended)
      nr=500                      ******** change, if necessary! ***********
c    initialization of U[0,1] random number generator ** leave, as it is! **
      idum=-1
```

```
c    LOCAL SEARCH PARAMETERS (POWELL-BRENT ALGORITHM)
c    search precision parameter
      t0= 1.0d-6                  ******** change, if necessary! ***********
c    machine accuracy (valid for Intel 386 DX PS/2 computers)
      machep= 0.542101086243d-19  ******* leave, as it is! *****************
c    maximum initial step size (estimated initial distance from optimum)
      h0= 1.d0                    ******* change, if necessary! ************
c    maximal number of objfct evaluations (recommended: O(n^2)  steps)
      maxfctbr= 20000             ******* change, if necessary *************
c    print control parameter: amount of output (can be 1,2,3,4; 1:minimal)
      prin= 1                     ******* change, if necessary! ************
c    heuristic parameter settings follow: they also may be changed!
c    set scbd=10., if the axes are badly scaled;  otherwise set scbd=1.
      scbd=10.d0                  ******* change, if necessary! ************
c    if the problem is ill-conditioned, set illc=1; otherwise illc=0
      illc= 1                     ******* leave, as it is! *****************
c    ktm is the number of main iterations without improvement before the algo-
c    rithm terminates; ktm=4 is cautious; even ktm=1 could be satisfactory
      ktm=4                       ******* change, if necessary! ************
```

Appendix III

Output File of Example 1

Note: This file—called lgocwi.sum, and located in the directory `/ufs/walterst/local/progf/LGO`—shows the output after a run of the optimization program LGO.

```
                LIPSCHITZIAN GLOBAL OPTIMIZATION PROGRAM SYSTEM

                              VERSION 3.1

                       (C)   J. PINTER   1986,...,1995


TOTAL NUMBER OF FUNCTION EVALUATIONS       38775

ESTIMATED OPTIMUM VALUE            0.000000

ESTIMATED LIPSCHITZ LOWER BOUND OF OPTIMUM VALUE        -1981.800903

ESTIMATED OPTIMAL SOLUTION VECTOR (EXPRESSED IN ORIGINAL SCALE)

  0.000002      -0.000002      -0.000003       0.000003       0.000000
  0.000000       0.000002       0.000000      -0.000001      -0.000004
  0.000006      -0.000009       0.000003      -0.000004       0.000003
 -0.000006       0.000012      -0.000005      -0.000011      -0.000001
  0.000001       0.000003      -0.000004       0.000003      -0.000001
  0.000002       0.000002       0.000003       0.000000      -0.000002
  0.000000       0.000000       0.000002       0.000002      -0.000004
 -0.000002      -0.000004       0.000000      -0.000004       0.000006
 -0.000005       0.000003       0.000002      -0.000004       0.000003
 -0.000003       0.000000      -0.000003       0.000003      -0.000002

*** LGO *** OPTIMIZATION RUN TERMINATED


user time=  0.1185E+02
system time=  0.9000E-01
total time=  0.1194E+02
```

Appendix IV

Output File of Example 2

Note: This file gives an impression of the output after a run of the optimization program for the compution of the Fekete points (6.2).

```
                NUMERICAL APPROXIMATION OF THE ELLIPTIC FEKETE POINTS

                             NUMBER OF POINTS: 13


                   LIPSCHITZIAN GLOBAL OPTIMIZATION PROGRAM SYSTEM

                                    VERSION 3.1

                           (C)   J. PINTER   1986,...,1995


 TOTAL NUMBER OF FUNCTION EVALUATIONS       29236

 ESTIMATED OPTIMUM VALUE          -35.873280

 ESTIMATED LIPSCHITZ LOWER BOUND OF OPTIMUM VALUE         -140.219498

 ESTIMATED OPTIMAL SOLUTION VECTOR (EXPRESSED IN ORIGINAL SCALE)

   1.988685       3.751593       2.065001       1.186166       1.984741
   2.378756       2.367821       3.450802       1.031831       1.680420
   1.046585       5.640462       1.108059       5.017105       2.029361
   0.519966       1.090310       4.519078       1.166383       5.683055
   2.966060       2.582749       1.463237


 *** LGO *** OPTIMIZATION RUN TERMINATED

 THE CORRESPONDING SOLUTION OF THE FEKETE PROBLEM

 ESTIMATED OPTIMUM VALUE (EXPRESSED IN LOG2 SCALE)           35.873280

 ESTIMATED OPTIMAL SOLUTION VECTOR (EXPRESSED IN ORIGINAL SCALE)

   0.000000       0.000000       0.000000       1.988685       3.751593
   2.065001       1.186166       1.984741       2.378756       2.367821
   3.450802       1.031831       1.680420       1.046585       5.640462
   1.108059       5.017105       2.029361       0.519966       1.090310
   4.519078       1.166383       5.683055       2.966060       2.582749
   1.463237

 user time=  0.4037E+02
 system time=  0.2200E+00
 total time=  0.4059E+02
```