

Submitted for publication: August 12, 2002.

Running title:

PINTÉR

GLOBAL AND CONVEX OPTIMIZATION IN MODELING ENVIRONMENTS

Global and Convex Optimization in Modeling Environments: Compiler-Based, Excel, and *Mathematica* Implementations

JÁNOS D. PINTÉR

Pintér Consulting Services, Inc. & Dalhousie University

129 Glenforest Drive, Halifax, NS, Canada B3M 1J2

jdpinter@hfx.eastlink.ca jdpinter@is.dal.ca

<http://is.dal.ca/~jdpinter>

Abstract. We present a review of several software products that serve to analyze and solve highly nonlinear—specifically including global—optimization problems across different hardware and software platforms. The implementations discussed are LGO, as a stand-alone, but compiler-dependent modeling and solver environment; its Excel platform implementation; and *MathOptimizer*, a native solver package for *Mathematica* users. The discussion advocates the use of hybrid solution strategies, in the context of practical global optimization. Several non-trivial illustrative applications, taken from the literature or received from users of our software products, are also reviewed.

Keywords: nonlinear (convex and global) optimization; LGO model development and solver system; LGO solver engine for Excel; *MathOptimizer* for *Mathematica*; illustrative applications and case studies.

AMS Subject Classification: 65K30, 90C05, 90C31.

1. Introduction

Nonlinear models—and modeling paradigms even beyond a straightforward, nonlinear function-based description—are of relevance in many areas of the sciences, engineering, and economics. Bracken and McCormick (1968), Rich (1973), Eigen and Winkler (1975), Mandelbrot (1983), Murray (1983), Casti (1990), Hansen and Jørgensen, eds. (1991), Schroeder (1991), Stewart (1995), Gershenfeld (1999), Edgar, Himmelblau and Lasdon (2001), Jacob (2001), and Wolfram (2002), as well as many other authors,

present far-reaching discussions and an extensive repertoire of examples to illustrate this point.

Decision-making based on a nonlinear system description frequently leads to complex models that (may or provably) have multiple—local and global—optima. The objective of global optimization (GO) is to find the 'absolutely best' solution of nonlinear optimization (NLO) models, under such circumstances. Many of the most important GO model types and promising solution approaches are discussed in the *Handbook of Global Optimization* volumes, edited by Horst and Pardalos (1995), and by Pardalos and Romeijn (2002). Several dozens of other textbooks and quite a few informative web sites are also devoted to this emerging subject.

Throughout this paper, we shall consider a fairly general GO model form defined by the following ingredients:

- x decision vector, an element of the real Euclidean n -space \mathbf{R}^n ;
- $f(x)$ continuous objective function, $f: \mathbf{R}^n \rightarrow \mathbf{R}^1$;
- D non-empty set of admissible decisions, a proper subset of \mathbf{R}^n ;
 D is defined by
- l, u explicit, finite bounds of x (an embedding 'box' in \mathbf{R}^n);
- $g(x)$ m -vector of continuous constraint functions, $g: \mathbf{R}^n \rightarrow \mathbf{R}^m$.

Applying the notation given above, the continuous global optimization (CGO) model is stated as

- (1) $\min f(x)$ s.t. x belongs to
- (2) $D = \{x: l \leq x \leq u, g(x) \leq 0\}$.

In (2), all vector inequalities are meant component-wise (l, u , are n -vectors and the zero denotes an m -vector). The set of additional constraints could be empty, thereby leading to—often much simpler, although still potentially multiextremal—box-constrained models.

Observe that the above stated 'minimal' assumptions guarantee (by the classical theorem of Weierstrass) that the optimal solution set X^* in the CGO model is non-empty. For reasons of numerical tractability, the following additional requirements are often postulated:

- D is a full-dimensional subset ('body') in \mathbf{R}^n ;
- the set of globally optimal solutions to (1)-(2) is at most countable;
- f and g (the latter component-wise) are Lipschitz-continuous functions on $[l, u]$.

Note that, for instance, models defined by continuously differentiable (C_1) functions f and g certainly belong to the CGO or even to the Lipschitz class; however, even this 'minimal' smooth structure is not essential (e.g., 'saw-tooth' like functions are also Lipschitzian). This comment shows the very general class of models covered by (1)-(2).

As a consequence of this generality, this model class includes also extremely difficult instances. To perceive this difficulty, one can think of 'finding the lowest valley across a range of islands', based on an intelligent (adaptive, automatic), but otherwise completely 'blind' sampling procedure. For illustration, a merely one-dimensional, box-constrained model instance is shown in Figure 1. This model is a frequently used classical GO test problem, originally due to Shubert: it is defined as

$$\min_{x \in [-10, 10]} \sum_{k=1, \dots, 5} k \sin(k + (k+1)x)$$

Figure 1.
One-dimensional, box-constrained CGO model.

Model complexity may increase dramatically, even in very low dimensional cases. For example, both the amplitude and the frequency of the trigonometric components in the model of Figure 1 could be increased arbitrarily, leading to more difficult problems.

Furthermore, increasing dimensionality *per se* can lead to a tremendous—theoretically exponential—increase of model complexity (e.g., in terms of the number of local/global solutions, for a given type of multiextremal models). To illustrate this point, see the—still merely two-dimensional, box-constrained, yet visibly challenging—objective function below. This model is based on Problem 4 in a set of numerical challenges by Trefethen (2002), and it is stated as

$$\begin{aligned} \min_{x, y \in [-3, 3]} & (x^2 + y^2)/4 + \exp(\sin(50x)) - \sin(10(x+y)) + \sin(60\exp(y)) + \sin(70\sin(x)) + \sin(\sin(80y)) \\ \text{s.t.} & -3 \leq x \leq 3, -3 \leq y \leq 3. \end{aligned}$$

Figure 2.
Two-dimensional, box-constrained CGO model. (Source: Trefethen, 2002.)

Needless to say, not all CGO models are as difficult as indicated by Figure 2. However, it is obvious that proper GO strategies have to offer far more, than the concepts and tools of local optimization. All traditional numerical methods search only for local optima, typically assuming the availability of a 'sufficiently good' initial solution vector. Especially Figure 2 suggests that this may not always be a realistic assumption...

Models with less 'dramatic' difficulty, but in (perhaps much) higher dimensions also indicate the need for global search. For instance, in advanced engineering design, models with hundreds (or even hundreds of thousands) of variables and constraints are of practical relevance. Therefore entirely different, global scope modeling paradigms and strategies are needed to solve GO problems. In similar cases to those illustrated by Figures 1 and 2, even an approximate, but genuine global search strategy may—and

typically will—give better results than the most sophisticated local search approach ‘started from the wrong valley’...

2. A Solver Suite Approach to Practical Global Optimization

The general development philosophy followed by the software products discussed in this work is the prudent and seamless combination of rigorous global and efficient local search strategies.

Theoretically, the existence of valid overestimates of the actual (smallest possible) Lipschitz-constants, for f and for each component of g in the model (1)-(2), is sufficient to guarantee the global convergence of suitably defined adaptive partition algorithms. That is, applying a proper branch-and-bound search strategy—that is partially based on the Lipschitz information referred to above—the generated sequence of sample points will converge (exactly) to the points of the global solution set X^* of the model instance considered. For further details related to the theoretical background, including also a discussion of implementation aspects, consult Pintér (1996) and references therein.

In numerical practice, deterministically guaranteed global convergence means that after a finite number of search steps—i.e., sample points and corresponding function evaluations—one has an incumbent solution (with a corresponding upper bound of the typically unknown optimum value), as well as a verified lower bound estimate. Furthermore, the ‘gap’ between these estimates converges to zero, as the number of sample points tends to infinity. For instance, interval arithmetic based approaches follow this avenue: consult, e.g., Ratschek and Rokne (1995) or Kearfott (1996); Corliss and Kearfott (1999) review a number of successful applications.

The essential difficulty of applying such rigorous approaches to ‘all’ GO models is that their computational demand typically grows at a fast (exponential) pace with the size of the models considered. For this reason, in a practical GO context, other search strategies also need to be considered.

Properly constructed stochastic search algorithms also possess strong theoretical global convergence properties (with probability 1): consult, for instance, the review of Boender and Romeijn (1995). For a very simple example that illustrates this point, one can think of a pure random search mechanism applied to solve the CGO model: this will eventually converge, if the ‘basin of attraction’ of the (say, unique) global optimizer x^* has a positive volume. (The not necessarily trivial issues of effectively implementing such a stochastic sampling strategy and using the information derived are not discussed here.)

In addition, stochastic sampling methods can also be directly combined with search steps of other—notably, efficient local—search strategies, and their global convergence will be still maintained. The theoretical background of such ‘hybrid’ algorithms is discussed by Pintér (1996).

The underlying general convergence theory of such combined methods allows for a broad range of implementations. In particular, a hybrid optimization program system supports the flexible usage of a selection of component solvers: one can execute a fully automatic global or local search based optimization run, can combine solvers, as well can design various interactive runs.

Obviously, there remains a significant theoretical issue regarding the (typically unforeseeable best) 'switching point' from strategy to strategy: this is however, unavoidable, when choosing between mathematical rigor and numerical efficiency. For example, in the stochastic search framework outlined above, it would suffice to find just one sample point in the 'region of attraction' of the (unique) global solution, and then that solution estimate can be rapidly 'polished' by a suitably robust and efficient local solver. Of course, the region of attraction of the optimal solution is rarely known, and one needs to rely on computationally expensive estimates of the model structure (again, the reader is referred, e.g., to the review of Boender and Romeijn, 1995).

In practice, there are several good reasons to apply such heuristic global-to-local search 'switching points' with a reasonable chance of success, namely:

- even multiextremal models are often (much) easier than reflected by the pictures displayed above;
- we often have a priori knowledge regarding good quality solutions, based on practical, model-specific knowledge (for example, one can think of solving systems of equations: here a global solution that 'nearly' satisfies the system can be deemed as a sufficiently good point from which local search can be started);
- one may always place more or less emphasis on rigorous search vs. efficiency, by selecting the appropriate solver and allocating search effort (time, function evaluations) to it;
- practical circumstance and limitations may dictate the use of additional numerical stopping and switching rules that can be built into the software: this may be the case when model evaluations are time-consuming, and therefore an upper bound on the total number of function evaluations has to be placed.

Based on the philosophy outlined, and further dictated by practical user/client demands, we have been developing several software implementations that are based on suitable global and local solver combinations. Three such software products will be discussed below; further related work is in progress.

3. Software Implementations

3.1. Modeling Systems and User Demands

In recent years, there has been a rapidly growing interest towards modeling languages and environments, built-in or optionally linked solvers, and complete decision support systems. Consult, for example, the topical *Annals of Operations Research* volumes edited

by Maros and Mitra (1995), Maros, Mitra, and Sciomachen (1997), Vladimirov, Maros, and Mitra (2000), Coullard, Fourer, and Owen (2001). Additional useful information can be found, for example, at the web sites maintained by Fourer (2002), Mittelmann and Spellucci (2002), and Neumaier (2002).

Examples of widely used systems that are focused on optimization related modeling include AMPL (Fourer, Gay, and Kernighan, 1993), GAMS (Brooke, Kendrick, and Meeraus, 1988), the Excel PSP Solver Engines (Frontline Systems, 2001), the LINDO Solver Suite (LINDO Systems, 1996), and MPL (Maximal Software, 2002). There exists also a fairly large collection of more specific, often (Basic, C, Fortran, Pascal, etc.) compiler platform dependent systems: consult e.g. the web sites mentioned. At the other end of the spectrum, there is also significant development related to fully integrated scientific and technical computing, visualization, and documentation systems such as MAPLE (Waterloo Maple, 2002), *Mathematica* (Wolfram, 1999), and MATLAB (The MathWorks, 2001). These more comprehensive systems also incorporate more or less optimization-related functionality, have a significant user basis, and are supplemented by specific application products. (Please note that the literature references cited above may not reflect the current status of the modeling systems listed: for the latest information, contact the developers.)

The modeling environments encompassed by the present discussion are aimed at meeting the demands of a broad range of clients. These include educational users (instructors and students); research scientists, engineers, economists, and consultants (possibly, but not necessarily equipped with an in-depth optimization related background); optimization experts, application developers, and other ‘power users’. The user categories listed are not necessarily disjoint, of course (one could be a top expert and software developer in some research area, with a more modest optimization expertise). The pros and cons of the individual software products—in terms of ease of model prototyping and detailed code development/maintenance, overall level of integration, immediate availability of auxiliary tools, program execution speed, quality of related documentation and support—make such systems more or less attractive for certain user groups.

It is also worth mentioning that—especially in the advanced nonlinear modeling and optimization context—it may be a salient idea to tackle challenging problems by making use of several modeling systems and solver tools, if available. In general, GO/NLO model formulations are far less easy to ‘standardize’ than (continuous) linear models: one typically needs an explicit, specific formula to describe a particular constraint or the objective(s). Such formulae are relatively straightforward to transfer from one modeling system into another: some of the systems listed above even have such built-in capabilities, and their standard function syntax is typically quite similar (whether it is $x**2$ or x^2 , $\sin(x)$ or $\text{Sin}[x]$, $\text{bernoulli}(n,x)$ or $\text{BernoulliB}[n,x]$, and so on).

In Sections 3.2 to 3.4 we shall summarize the principal features of three different nonlinear optimization software implementations that have been developed with several—arguably quite diverse—user groups in mind. One of these is a stand-alone, but compiler-platform based modeling environment: the core of this software version can be

made available also as a solver option in other modeling systems. The other implementations are directly connected to two widely used modeling platforms, Microsoft Excel and *Mathematica*.

Note that these software products are professionally developed and supported, and that they are commercially available. For this reason (and also in line with the objectives of this paper) some of the algorithmic technical details are only briefly mentioned. Additional information is available upon request; please consult also the publicly available references, including the software documentation and several web sites. The web site <http://is.dal.ca/~jdpinter> provides summary information regarding the software versions discussed here.

3.2. LGO Solver System

The Lipschitz Global Optimizer (LGO) software has been developed and used for more than a decade (as of 2002). Detailed technical descriptions and user guidance appeared elsewhere: consult, for instance, Pintér (1996, 1997, 2000a, 2001a), and the software review by Benson and Sun (2000).

In accordance with the approach advocated in Section 2, LGO is based on a combined implementation of a suite of global and local scope nonlinear solvers. These can be invoked both automatically and in fully interactive mode. Currently the stand-alone LGO implementation includes the following solver options:

- branch-and-bound based global search, enhanced with a stochastic sampling procedure that supports the usage of statistical bound estimation methods;
- adaptive global random search, enhanced with a statistical bound estimation technique (as above);
- exact penalty (aggregated merit) function based local search;
- local search, based on sequential model linearization;
- local search, based on a reduced gradient approach.

The core LGO system can be used as a solver option linked to modeling environments. Furthermore, it can be equipped—as a readily available implementation option—with a user-friendly MS Windows interface. This version is called LGO Integrated Development Environment (IDE). Figure 3 shows the LGO IDE front page, with its main menu.

Figure 3.
LGO Integrated Development Environment.

The menu options support model formulation, solution and result analysis. Optimization models are formulated, making use of given template files. Following compilation, an application specific executable file is generated. This program can be run—possibly repeatedly—making use of an input (text) file that sets several key solver parameters

(such as the maximal number of function evaluations, or required solution accuracy measures), in addition to variable bounds and a user-supplied nominal solution vector. (Most of these options will be discussed and shown in Figure 4, Section 3.3.)

The results are automatically written to a summary and a detailed report file. The LGO IDE also has a built-in model visualization capability, to assist the modeling and result verification procedure. Namely, objective and constraint function projections can be generated in interactively selected subspaces. Concise help files provide summary technical information and an outline of the application development procedure.

As noted previously, this LGO implementation is compiler-based: user models can be connected to LGO using one of several widely used programming languages. Currently these include the following:

- all professional Fortran 77/90/95 and several C software platforms can be used to develop and to run 'silent' LGO implementations (without the Windows interface); a frequently used development environment is Compaq/Digital Visual Fortran that also supports C/Fortran mixed language application development projects;
- connectivity to the LGO IDE is specifically supported (by Lahey Computer Systems, the developers of LF77/90/95) with respect to the following development platforms: Borland C/C++, Borland Delphi, Lahey Fortran 77/90/95, Microsoft Visual Basic, and Microsoft Visual C/C++ (for details, consult Lahey Computer Systems, 1999).

Other customized versions can also be made available upon request, especially since the vendors of development environments often expand the list of compatible platforms.

This LGO software implementation (in both versions) fully supports communication with sophisticated user models, including even entirely closed or confidential ('black box') systems. External system calls to other applications—that are available on the client's computer—are also supported. Note furthermore that these LGO implementations are particularly advantageous in application areas, where program execution (solution) speed is a major concern: in the GO/NLO context, many projects fall into this category. The added features of the LGO IDE can also greatly assist in educational and research projects.

The LGO software is accompanied by a (nearly 60-page) User Guide. In addition to installation and technical notes, this document provides a brief introduction to GO; describes LGO and its solvers; discusses the model development procedure, including modeling and solution tips; and reviews a list of applications. The appendices provide examples of the user (main, model, and input parameter) files, as well as of the resulting output files; connectivity issues and workstation implementations are also discussed.

LGO has been applied by researchers and practitioners (so far, in over a dozen countries) to solve complex models in up to 1000 variables and 1000 constraints, both in educational/research and business contexts. Articles and numerical examples, specifically related to this LGO implementation, are available from the author upon request.

3.3. LGO Solver Engine for Excel

The LGO Global Solver Engine for Microsoft Excel has been developed in close cooperation with Frontline Systems, Inc., the developers of the Excel Solver product family. For details on the Excel Solver and the currently available advanced engine options, consult the (nearly 80-page) User Guide published—in cooperation with the other engine developers—by Frontline Systems (2001), and visit www.solver.com. This web site contains additional useful information, including for instance, tutorial material, modeling tips, various spreadsheet examples, and suggested literature. The User Guide provides a brief introduction to the solver engines, covers the installation steps, discusses the diagnosis of solver results, solver options and reports; it also contains a section on solver VBA functions. Note that solver engine specific information can also be invoked through Excel's on-line help system.

In the Excel-specific implementation, LGO is a field-installable Solver Engine that seamlessly connects to the Premium Solver Platform: the latter is fully compatible with the standard Excel Solver, but it has enhanced algorithmic capabilities and features.

LGO for Excel is designed for finding the global (or local, if required) solution of problems formulated with arbitrary continuous functions, currently with up to 1,000 variables and 1,000 constraints. It also provides basic support for handling integer variables: this feature has been implemented—as a generic option for all advanced solver engines—by Frontline Systems.

The LGO solver options available are essentially based on the stand-alone 'silent' version of the software, with a few simplifications. The Excel-specific LGO implementation also has numerous added features: these serve to seamlessly connect to and operate on Excel model formulations, in a standardized fashion.

Similarly to the LGO IDE interface, the LGO Solver Engine Options dialog (shown on Figure 4) lets the user to control solver choices and several heuristic approaches (switches) that will often greatly speed up the process of finding a good quality solution.

Figure 4.
LGO for Excel: solver options and parameters dialog.

As shown in this figure, in addition to the standard Solver stopping criteria (defined by time limit, iterations limit, numerical precision and numerical convergence parameters), the following LGO-specific options are available:

- theoretical global convergence related stopping criterion parameter, for the branch-and-bound search option;
- global search result acceptability threshold value (to activate switch to local search);

- the maximal number of global search phase iterations (before automatically switching to local search);
- the maximal number of subsequent global search phase iterations without improvement (again, before switching to local search);
- the radio-button selector list of LGO usage options: global branch-and-bound, global adaptive random search, and local search from the given nominal solution.

Note that these LGO options are also available—through the input parameter file or via dialogs—in the implementations discussed in Section 3.2.

The Excel/LGO implementation has been developed mainly with business users in mind, for whom Excel is the primary tool of choice; of course, it can also be well applied in corresponding educational and research settings. Spreadsheet examples, specifically related to the LGO Global Solver Engine, are available from the author upon request.

3.4. *MathOptimizer for Mathematica Users*

Mathematica offers a sophisticated, fully integrated environment for scientific education and research, modeling, algorithm prototyping, and detailed (symbolic as well as numerical) computing. *Inter alia*, it supports several—functional, rule-based and more conventional (procedural)—programming styles, enables advanced visualization, and produces publication-quality documentation. For further information, please consult the authoritative reference Wolfram (1999), and visit www.wolfram.com: this site provides detailed information regarding also the impressive range of products and services related to *Mathematica*.

MathOptimizer is a native *Mathematica* product that—similarly to LGO—enables the global and local solution of the general class of optimization problems encompassed by the CGO model (1)-(2).

MathOptimizer currently consists of two core solver packages and a solver integrator package. One of the solvers serves for (in practice, typically approximate) global optimization on a given interval range, in the possible presence of added constraints. This package is based on an efficient adaptive stochastic search method, combined with a statistical bounding procedure. (The latter provides an estimate of the global optimum value, based on the global sampling results.) The second solver package is based on a constrained nonlinear (convex) programming approach: this solver can be used for 'precise' local optimization, based on a given initial solution. The solver integrator supports the individual or combined use of these two solver packages, but the solvers can also be used separately. Further solver modules are under development, and will be made available soon.

The currently over 70-page *MathOptimizer* User Guide (Pintér, 2002a) is 'live' *Mathematica* (notebook) document that can be directly invoked through the on-line help system. The User Guide includes installation and technical notes, and provides concise mathematical background information and modeling tips. In addition to basic usage

description, it also discusses a number of simple and more challenging test problems, and a few application examples in detail. For illustration, Figure 5 displays a randomly generated test model discussed in the *MathOptimizer* User Guide.

Figure 5.

A randomly generated test problem solved by *MathOptimizer*.
(The surface plot and contour plot of the objective function are shown.)

MathOptimizer can be installed and used on all hardware platforms that are suitable to run *Mathematica* (3.0 or later version) itself. It is available for all Microsoft Windows (95 and later) operating systems, as well as for all other platforms that have corresponding current *Mathematica* implementations: several prominent examples are Linux, Macintosh, Mac OSX, HP, SGI, Solaris, Sun platforms and their operating systems. (Actual software installation and tests with identical file systems were successfully completed on over half a dozen different hardware platforms and operating systems, at the time of this writing.)

Mathematica has a significant, typically very ‘computer-literate’ user basis worldwide. *MathOptimizer* has been developed mainly with these (business, research and educational) users in mind. *MathOptimizer* application examples are available from the author upon request.

4. Illustrative Applications

In this section, we shall provide a very brief description of several recent computational studies and applications in which the software implementations discussed above were used. Far more extensive lists of GO/NLO applications have been reviewed elsewhere: consult, for instance, the works cited in Section 1, or Pintér (1996, 1999, 2001a, and 2002a, b, c), with numerous further references and pointers.

Due to the extent and depth of some of the illustrative studies, we have to omit most technical details, providing appropriate references instead. Only the global optimization context is highlighted here, with a brief—often merely qualitative—summary of the results that will appear (or appeared) elsewhere.

4.1. Electrical Circuit Design

This model has been extensively studied in the global optimization literature, as a well-known computational challenge: see, e.g., Ratschek and Rokne (1993), with detailed historical notes and further references.

In this study, a bipolar transistor is modeled by an electrical circuit. Mathematically, this model leads to the following square system of nonlinear equations

$$a_k(x)=0 \quad k=1,\dots,4; \quad b_k(x)=0 \quad k=1,\dots,4; \quad c(x)=0;$$

the variable x belongs to \mathbf{R}^9 . The individual equations are defined as follows:

$$a_k(x)=(1-x_1x_2) x_3 \{ \exp[x_5 (g_{1k} - g_{3k} x_7 - g_{5k} x_8)] - 1 \} - g_{5k} + g_{4k} x_2 \quad k=1,\dots,4;$$

$$b_k(x)=(1-x_1x_2) x_4 \{ \exp[x_6 (g_{1k} - g_{2k} - g_{3k} x_7 + g_{4k} x_9)] - 1 \} - g_{5k} x_1 + g_{4k} \quad k=1,\dots,4;$$

$$c(x)=x_1x_3-x_2x_4.$$

The numerical constants $g_{ik} \quad i=1,\dots,5, \quad k=1,\dots,4$ are listed in the paper of Ratschek and Rokne (1993), and will not be repeated here. (Note that, in order to make the model functions more readable, several constants are simply aggregated in the above formulae, when compared to that paper.)

To solve the circuit design model, Ratschek and Rokne applied a combination of interval arithmetic, subdivision and branch-and-bound strategies. They concluded that the solution was extremely costly (billions of model function evaluations were used), in order to arrive at a *guaranteed* interval (i.e., embedding box) estimate that is component-wise within at least 10^{-4} precision of the postulated *approximate* solution:

$$x^*=(0.9, 0.45, 1.0, 2.0, 8.0, 8.0, 5.0, 1.0, 2.0).$$

Obviously, by taking, e.g., the Euclidean norm of the overall error in the model equations, finding a solution(s) can be perceived as a global optimization problem. Following the related literature, we shall assume that $0 \leq x_i \leq 10$ (component-wise, in \mathbf{R}^9).

This model has been set up as a test for the Excel LGO solver engine. Figure 6 shows the solution found by LGO in less than 35,000 search steps. (The solution takes about 30 seconds on a personal computer equipped with an Intel Pentium 4 class 1.6 GHz processor.)

Figure 6.

Electrical circuit design model: Excel model form and solution by LGO engine.

As one can see, the approximate solution found by LGO agrees to significant precision with the solution postulated (the detailed Excel answer report shows agreement with x^* to at least 10^{-6} precision, for each coordinate).

It is emphasized that the LGO implementation has built-in heuristic features, and hence it may occasionally miss the optimum in typical usage. However, it is seen that—at least in this particular case—a well-known GO challenge is solved by LGO to remarkable precision. If one thinks of the usual precision needs of practical modeling, and the unavoidable modeling and numerical errors, then the result obtained seems to be perhaps even more satisfactory.

4.2. Globally Optimized Point Configurations

The ‘well-balanced’ distribution of points over a given surface (e.g. of a sphere) or within a volume (such as an n -dimensional box) is of significant interest in various scientific and engineering studies. Examples of applications are related to experimental design, numerical approximation and integration, computational complexity studies, packing and covering problems, potential energy models in physics and chemistry, molecular structure modeling, and viral morphology. We only mention here the extensive literature that has been devoted to these subjects; the references cited below provide further pointers to that literature.

The quality of point configurations is expressed by a context-specific objective function that, most typically, possesses a large number of local optima. Therefore the application of a global scope search strategy—quite possibly used in conjunction with problem-specific knowledge—is essential.

For illustration, let us consider the following problem, as one of the point arrangement model variants:

$$\max_{\{x_i\}} \{ \min_{1 \leq i < j \leq k} \|x_i - x_j\| \} \quad x_i \in [x_l, x_u] \subset \mathbf{R}^d \quad i=1, \dots, k$$

That is, given the d -dimensional interval $[x_l, x_u]$, find a k -tuple of points $\{x_i\}$ in this box such that the minimum of the pair-wise distances between all points becomes maximal. (Note that—following an elementary scaling procedure and a simple transformation—this model is closely related to and generalizes the problem of uniform circle packings in the unit square.)

To illustrate the complex multiextremality issue referred to above, Figure 7 displays a projection of the max-min objective function: in the specific example shown, $d=2$, $k=13$.

Figure 7.

Max-min model, for 13 points in 2 dimensions: a projection of the objective function.

This model—as well as several other model variants—has been recently solved for illustrative point configurations, by using LGO. Most of these configurations consist of tens of points, hence representing relatively small, yet non-trivial instances. The number of decision variables is typically a (exactly or nearly) linear function of the number of points, since these variables define the point positions. The best results known (for postulated or provably optimal point configurations) have been reproduced to a close approximation in all cases studied: for modeling and numerical details, consult Pintér (2000b, 2001a,b). Stortelder, de Swart, and Pintér (2001) discuss one of these models (the elliptic Fekete problem) solving problems up to 150 points. The LGO software—in ‘silent’ and IDE-enabled implementations—has been applied in all these computational

studies. (Note that Stortelder and Swart used an entirely different solution approach: please consult our joint paper for details.)

Concluding this example, it is emphasized that in the numerical examples reviewed no structural insight was used, i.e., only a ‘blind’ global optimization approach has been applied. In more thorough and exhaustive studies, incorporation of model-specific knowledge—tighter variable bounds, perhaps postulated good configurations—could (and certainly should) be used.

4.3. *Laser Equipment Design*

The laser is a device that produces a beam of coherent light, by a process known as stimulated emission. (As probably most readers will recall, the word *laser* is an acronym for the phrase ‘Light Amplification by Stimulated Emission of Radiation’.) Stimulated emission was proposed by Albert Einstein in 1916. It took another four decades to build the first lasers as a scientific research tool, but soon they found numerous significant applications.

Figure 8.
Index-coupled, distributed feedback laser.
(Source: Isenor, 2001.)

Figure 8 displays a schematic picture of a distributed feedback laser that has been recently studied by Isenor (2001), with an overview of and references to the state-of-art therein. The minimization of the so-called *field flatness function* (related to emission quality) leads to a CGO model that has only a few optimization parameters, but features a ‘black box’ objective and a similarly complicated constraint, in addition to finite variable bounds. The ‘black box’ model functions are numerically evaluated, by a system of coded (Fortran 77) routines: these are then coupled to and jointly compiled with the LGO optimizer object file. For all engineering and numerical details, we refer to Isenor (2001); the key optimization issues and the solution approach are summarized in Isenor, Pintér, and Cada (2002).

The LGO software—in its IDE implementation—has been applied to solve this problem in a detailed numerical study. The (discretization-dependent) number of variables in this study was kept below ten, to avoid even longer runtimes that were in the range of tens of minutes to a few hours. The computational study has resulted in a significant laser performance improvement: namely, over 90% improvement of the field flatness function (when compared to known nominal settings) has been attained, using the global optimization approach in combination with the detailed laser model.

4.4. *Transducer Design*

The electric circuit shown below (see Figure 9) simulates a piezoelectric flex-tensional sonar projector. This circuit model was developed by Moffett et al. (1992), to describe the so-called barrel stave projector. This is an equivalent circuit model: that is, electric circuit components simulate the mass, compliance and damping of mechanical components by corresponding inductances, capacitances, and resistances.

Figure 9.

Electric circuit model, developed to simulate a piezoelectric flex-tensional sonar projector. (Source: Moffett et al., 1992.)

The conversion from mechanical to electrical degrees of freedom is handled by a transformer: a corresponding parameter represents its transformer turns ratio. Two other transformers (modeled by respective turns ratios) represent the flex-tensional mechanical transformers.

The optimization problem consists of finding a set of design parameters that gives an overall ‘nice’ broad efficiency versus frequency. Efficiency is measured as the ratio of radiated acoustic power, delivered to the radiation resistance R_r , to input electric power provided by the voltage source V (see Figure 9). The ideal efficiency curve is given by a Hamming window function, a distribution commonly seen in signal processing applications. For further technical details, outside of the scope of this paper, consult Moffett et al. (1992).

For the purposes of this illustrative numerical example, some of the design parameters have been fixed, but 7 parameters have been left to optimize. The design parameters have all been normalized so that the frequency range of interest is assumed to be between 0 and 1 Hz. The parameters belong to given finite ranges; there are no other (equality or inequality) constraints.

The objective function in this design problem is described by a rather complicated numerical model. The corresponding *Mathematica* code was developed by C. J. Purcell, and it is included in the *MathOptimizer* User Guide. The model was subsequently solved using *MathOptimizer*, and the results were deemed (by Dr. Purcell and his research team) of superior quality, when compared to results found earlier by using the built-in local optimizer functionality of *Mathematica*.

4.5. Radiotherapy Planning

Development of new equipment for radiation therapy provides exciting opportunities for designing (patient-specific) targeted dose delivery plans. A group of researchers at the University of Kuopio (Finland) has been involved in studies related to static and dynamic delivery techniques based on the operations of the so-called multileaf collimator (MLC) equipment. Profiting also from related research and literature, they have been developing sophisticated models to optimize treatment schedules. Specifically—in the multiple static

dose delivery context—they developed a mathematical model that enables the calculation of the radiation intensity distribution as a function of the MLC leaf-head positions, in a sequence of positional settings. This leads to an optimization model in which the (globally) best, technologically feasible delivery is sought. The quality of the delivery plan is defined by several closely related factors. The primary objective is the irradiation of the target volume at a prescribed level; at the same time, the dangerous irradiation of close-by organs at risk, and of the surrounding body tissue should be kept as little as possible.

The resulting model versions are typically large-scale, complicated constrained global optimization problems, with hundreds of variables and constraints (or even more, if the modeling procedure is refined.) Corresponding runtimes on current top-of-the-line PCs can be several hours, or even more. This fact again supports the use of heuristic shortcuts and flexible operational modes as implemented in LGO. (This research group has been using the LGO IDE implementation.)

The detailed technical description is again far beyond the scope of this paper: consult, for instance, Tervo, Kolmonen, Lyyra-Laitinen, Pintér, and Lahtinen (2002), and references therein. To provide at least a pictorial representation of the type of results derived, see Figure 10.

Figure 10.

Globally optimized radiation effect in a numerical example.
(Source: J. Tervo's research group, University of Kuopio.)

This picture is based on a numerical experiment that solves a two-dimensional, simplified test problem. The modeled body section is represented by the hexagonal area shown; the near centrally positioned, larger polygon symbolizes the radiation target; a nearby organ at risk is represented by the smaller polygon. The scale on the right shows the normalized radiation level. As this illustrative result indicates, a fairly precise delivery plan has been realized that provides a well-targeted overall dose, while basically avoiding harmful radiation side effects. The research team continues working on this issue, with the objective of making our experimental results part of a realistic treatment plan.

4.6. The Hundred-Dollar, Hundred-Digit Challenge Problems: Problem 4

As for a final, more light-hearted illustrative example, let us revisit Problem 4 posted by Trefethen (2002); please recall Figure 2. This problem has been set up as a test model in the *MathOptimizer* User Guide, and it has been solved numerically (in February 2002) in about 0.15 seconds, on the personal computer mentioned earlier. The solution found equals (to 10-digit accuracy, as was requested by Trefethen, for the challenges posted)

$$x^* \sim -3.306868647.$$

Based on the statistical estimation method built into *MathOptimizer* (using a 10^6 -size random sample of points taken from the box $[-3,3]^2$) it has been conjectured in the User Guide that the quality of this solution is about 0.999338, assuming that the quality of the true solution is 1.

To my slight(?) surprise, I found out in August 2002 that the true (unique) optimum value to 10-digit accuracy is indeed -3.306868647, as cited at the Wolfram Research web site http://mathworld.wolfram.com/news/2002-05-25_challenge/. The WR team used an interval arithmetic based method to rigorously verify this solution.

Needless to say, it would be very ‘optimistic’ to claim that the software implementations discussed in this work are always this successful... Global optimization is and will remain a field of extreme numerical difficulty, not only when considering ‘all possible’ problems, but also in attempts to handle practical models in an acceptable timeframe. However, we think that the applications highlighted show the merits of a carefully balanced, global/local solver suite based GO approach.

5. Conclusions

In this paper, a review of several nonlinear (convex and global) optimization software products is presented. The software implementations discussed are LGO (compiler platform specific versions), LGO for Excel, and *MathOptimizer*. It is our objective to add functionality to the existing products, and to develop further implementations, in order to meet the needs of a broad range of users.

The discussion advocates a practically motivated approach that combines rigorous global optimization strategies with efficient local search methodology, in seamlessly operated, flexible solver suites. The illustrative list of non-trivial application examples and the good quality numerical results (all obtained in a realistic timeframe) show the practical merits of this approach.

We are interested to learn suggestions regarding future development directions. GO/NLO test problems and challenges are also welcome.

Acknowledgements

Several application examples reviewed in this paper are based on actual user models and test problems. For valuable discussions, for communicating these problems, and for their work on the corresponding models and application code, I wish to thank the following colleagues:

- Walter Stortelder and Jacques de Swart, at the time of our cooperation working at the National Research Institute for Mathematics and Computer Science (CWI), Amsterdam (elliptic Fekete problem);

- Glenn Isenor, NRC Canada, Halifax, NS (laser design problem);
- Chris Purcell, DRDC-Atlantic, Dartmouth, NS (transducer design problem);
- Jouko Tervo and his research group at the University of Kuopio, Finland (radiotherapy planning problem).

The stimulating discussions with Dick den Hertog and Jack Kleijnen (University of Tilburg, Netherlands) regarding experimental design issues (max-min and other formulations) are also acknowledged.

The development of the core LGO ('silent') version and of the LGO IDE version was made easier by quality software and technical support provided by Lahey Computer Systems, Inc.

The Excel solver engine implementation of LGO was completed in cooperation with Frontline Systems, Inc. I wish to thank Dan Fylstra, Tom Aird, and Edwin Straver for providing valuable advice, software components, and technical documentation to support this development work.

The development of *MathOptimizer* greatly profited from expert advice, books, software, and a visiting scholarship received from Wolfram Research, Inc. Although I never worked together formally with anyone from WR on this development project, I owe to my colleagues there at least a collective acknowledgement. Chris Purcell provided numerous insightful comments and tips during the development, testing, and documentation of *MathOptimizer*, in the framework of DRDC Contract W7707-01-0746. Thanks are due also to Anton Rowe for advice and useful pointers.

The research and development work reviewed in this paper have been partially supported by the National Research Council of Canada (NRC IRAP Project 362093), the Hungarian Scientific Research Fund (OTKA Grant T 034350), and the Dutch Technology Foundation (STW Grant CWI55.3638).

References

Benson, H.P., and Sun, E. (2000) LGO—Versatile Tool for Global Optimization. *OR/MS Today* 27 (5), 52-55.

Boender, C.G.E. and Romeijn, H.E. (1995) Stochastic Methods. In: Horst and Pardalos, eds. (1995), pp. 829-869.

Bracken, J. and McCormick, G.P. (1968) *Selected Applications of Nonlinear Programming*. Wiley, New York.

Brooke, A., Kendrick, D. and Meeraus, A. (1988) *GAMS: A User's Guide*. The Scientific Press, Redwood City, CA. (Revised versions are available from the GAMS Corporation.) See also <http://www.gams.com>.

Casti, J.L. (1990) *Searching for Certainty*. Morrow & Co., New York.

Corliss, G.F. and Kearfott, R.B. (1999) *Rigorous Global Search: Industrial Applications*. In: Csendes, T., ed. *Developments in Reliable Computing*, pp. 1-16. Kluwer Academic Publishers, Dordrecht.

Coullard, C., Fourer, R., and Owen, J. H., eds. (2001) *Annals of Operations Research Volume 104: Special Issue on Modeling Languages and Systems*. Kluwer Academic Publishers, Dordrecht.

Edgar, T.F., Himmelblau, D.M., and Lasdon, L.S. (2001) *Optimization of Chemical Processes*. (Second Edn.) McGraw-Hill, New York.

Eigen, M. and Winkler, R. (1975) *Das Spiel*. Piper & Co., München.

Fourer, R. (2001) *Nonlinear Programming Frequently Asked Questions*. Optimization Technology Center of Northwestern University and Argonne National Laboratory, <http://www-unix.mcs.anl.gov/otc/Guide/faq/nonlinear-programming-faq.html>.

Fourer, R., Gay, D.M., and Kernighan, B.W. (1993) *AMPL—A Modeling Language for Mathematical Programming*. The Scientific Press, Redwood City, CA. (Reprinted by Boyd and Fraser, Danvers, MA, 1996.) See also <http://www.ampl.com>.

Frontline Systems (2001) *Premium Solver Platform — Solver Engines. User Guide*. Frontline Systems, Inc. Incline Village, NV. See also <http://www.solver.com>. Regarding the LGO solver, see <http://www.solver.com/xlslgoeng.htm>; a summary is provided also at <http://is.dal.ca/~jdpinter>.

Gershenfeld, N. (1999) *The Nature of Mathematical Modeling*. Cambridge University Press, Cambridge.

Hansen, P.E. and Jørgensen, S.E., eds. (1991) *Introduction to Environmental Management*. Elsevier, Amsterdam.

Horst, R. and Pardalos, P.M., eds. (1995) *Handbook of Global Optimization. Volume 1*. Kluwer Academic Publishers, Dordrecht.

Isenor, G. (2001) *A Novel Approach to the Reduction of a Distributed Feedback Laser's Intensity Profile Non-Uniformity Using Global Optimization*. Ph.D. Thesis, Faculty of Engineering, Dalhousie University, Halifax, NS.

Isenor, G., Pintér, J.D., and Cada, M. (2002) Globally Optimized Steady State Characteristics of a Distributed Feedback Laser. *Optimization and Engineering* (to appear).

Jacob, C. (2001) *Illustrating Evolutionary Computation with Mathematica*. Morgan Kaufmann Publishers, San Francisco.

Kearfott, R.B. (1996) *Rigorous Global Search: Continuous Problems*. Kluwer Academic Publishers, Dordrecht.

Lahey Computer Systems (1999) *Fortran 90 User's Guide*. Lahey Computer Systems, Inc., Incline Village, NV. <http://www.lahey.com>.

LINDO Systems (1996) *Solver Suite*. LINDO Systems, Inc., Chicago, IL. See also <http://www.lindo.com>.

Mandelbrot, B.B. (1983) *The Fractal Geometry of Nature*. Freeman & Co., New York.

Maros, I., and Mitra, G., eds. (1995) *Annals of Operations Research Volume 58: Applied Mathematical Programming and Modeling II (APMOD 93)*. J.C. Baltzer AG, Science Publishers, Basel.

Maros, I., Mitra, G., and Sciomachen, A., eds. (1997) *Annals of Operations Research Volume 81: Applied Mathematical Programming and Modeling III (APMOD 95)*. J.C. Baltzer AG, Science Publishers, Basel.

Mittelman, H.D. and Spellucci, P. (2002) *Decision Tree for Optimization Software*. <http://plato.la.asu.edu/guide.html>.

Maximal Software (2002) *MPL Modeling System*. Maximal Software, Inc. Arlington, VA. See also <http://www.maximal-usa.com>.

Moffett, M.B., Lindberg, J.F., McLaughlin, E.A., and Powers, J.M. (1992) An Equivalent Circuit Model for Barrel-Stave Flextensional Transducers. In: McCollum, M.D., Hamonic, B.F. and Wilson, O.B., eds. *Proc. Transducers for Sonics and Ultrasonics*, pp. 170-180 Technomic, Lancaster, PA.

Murray, J.D. (1983) *Mathematical Biology*. Springer-Verlag, Berlin.

Neumaier, A. (2002) *Global Optimization*. <http://www.mat.univie.ac.at/~neum/glopt.html>.

Pardalos, P.M. and Romeijn, H.E., eds. (2002) *Handbook of Global Optimization. Volume 2*. Kluwer Academic Publishers, Dordrecht.

Pintér, J.D. (1996) *Global Optimization in Action*. Kluwer Academic Publishers, Dordrecht.

Pintér, J.D. (1997) LGO — A Program System for Continuous and Lipschitz Optimization. In: Bomze, I.M., Csendes, T., Horst, R. and Pardalos, P.M., eds. *Developments in Global Optimization*, pp. 183-197. Kluwer Academic Publishers, Dordrecht.

Pintér, J.D. (1999) Continuous Global Optimization: An Introduction to Models, Solution Approaches, Tests and Applications. *Interactive Transactions in Operations Research and Management Science* 2, No. 2. <http://catt.bus.okstate.edu/itorms/pinter/>.

Pintér, J.D. (2000a) *LGO — A Model Development System for Continuous Global Optimization. User's Guide*. Pintér Consulting Services, Inc., Halifax, NS. For a summary, see <http://is.dal.ca/~jdpinter>.

Pintér, J.D. (2000b) Extremal Energy Models and Global Optimization. In: Laguna, M. and González-Velarde, J-L., eds. *Computing Tools for Modeling, Optimization and Simulation*, pp. 145-160. Kluwer Academic Publishers, Dordrecht.

Pintér, J.D. (2001a) *Computational Global Optimization in Nonlinear Systems*. Lionheart Publishing Inc., Atlanta, GA.

Pintér, J.D. (2001b) Globally Optimized Spherical Point Arrangements: Model Variants and Illustrative Results. *Annals of Operations Research* 104, 213-230.

Pintér, J.D. (2002a) *MathOptimizer - An Advanced Modeling and Optimization System for Mathematica Users. User Guide*. Pintér Consulting Services, Inc., Halifax, NS.

For a summary, see <http://www.wolfram.com/products/applications/mathoptimizer/>, and <http://is.dal.ca/~jdpinter>.

Pintér, J.D. (2002b) Global Optimization: Software, Test Problems, and Applications. In: Pardalos and Romeijn, eds. (2002), pp. 515-569.

Pintér, J.D., ed. (2002c) *Global Optimization—Selected Case Studies*. (To appear.) Kluwer Academic Publishers, Dordrecht.

Ratschek, H., and Rokne, J. (1993) Experiments Using Interval Analysis for Solving a Circuit Design Problem. *Journal of Global Optimization* 3, 501-518.

Ratschek, H. and Rokne, J. (1995) Interval Methods. In: Horst and Pardalos, eds. (1995), pp. 751-828.

Rich, L.G. (1973) *Environmental Systems Engineering*. McGraw-Hill, Tokyo.

Schroeder, M. (1991) *Fractals, Chaos, Power Laws*. Freeman & Co., New York.

Stewart, I. (1995) *Nature's Numbers*. Basic Books / Harper and Collins, New York.

Stortelder, W.J.H., de Swart, J.J.B., and Pintér, J.D. (2001) Finding Elliptic Fekete Point Sets: Two Numerical Solution Approaches. *Journal of Computational and Applied Mathematics* 130, 205-216.

Tervo, J., Kolmonen, P., Lyyra-Laitinen, T., Pintér, J.D., and Lahtinen, T. (2002) An Optimization-Based Approach to the Multiple Static Delivery Technique in Radiation Therapy. *Annals of Operations Research* (to appear).

Trefethen, L.N. (2002) The Hundred-Dollar, Hundred-Digit Challenge Problems. *SIAM News*, January-February issue, page 3.

The MathWorks (2000) *Using MATLAB*. The MathWorks, Inc., Natick, MA. See also <http://www.mathworks.com>.

Vladimirou, H., Maros, I., and Mitra, G., eds. (2000) *Annals of Operations Research Volume 99: Applied Mathematical Programming and Modeling IV (APMOD 98)*. J.C. Baltzer AG, Science Publishers, Basel, Switzerland.

Waterloo Maple (2002) *MAPLE User Guide Set*. Waterloo Maple, Inc., Waterloo, ON. See also <http://www.maplesoft.com>.

Wolfram, S. (1999) *The Mathematica Book*. (Fourth Edn.) Wolfram Media, Champaign, IL, and Cambridge University Press, Cambridge. See also <http://www.wolfram.com>.

Wolfram, S. (2002) *A New Kind of Science*. Wolfram Media, Champaign, IL, and Cambridge University Press, Cambridge.

Figure 1.
One-dimensional, box-constrained CGO model.

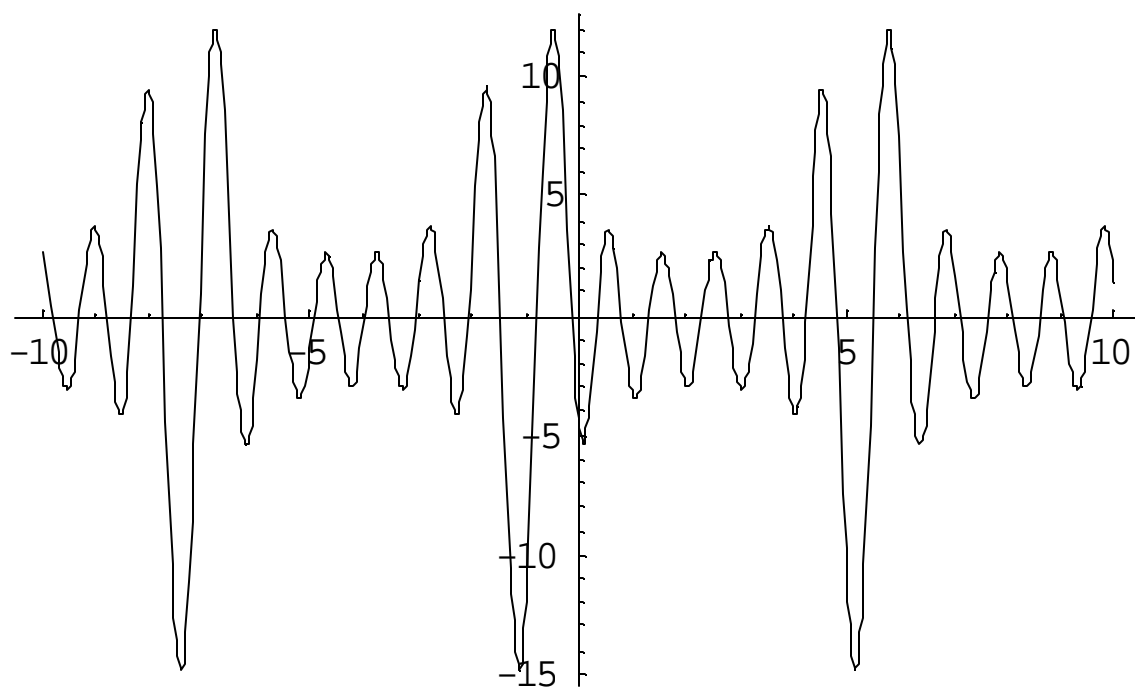


Figure 2.

A difficult two-dimensional, box-constrained CGO model. (Source: Trefethen, 2002.)

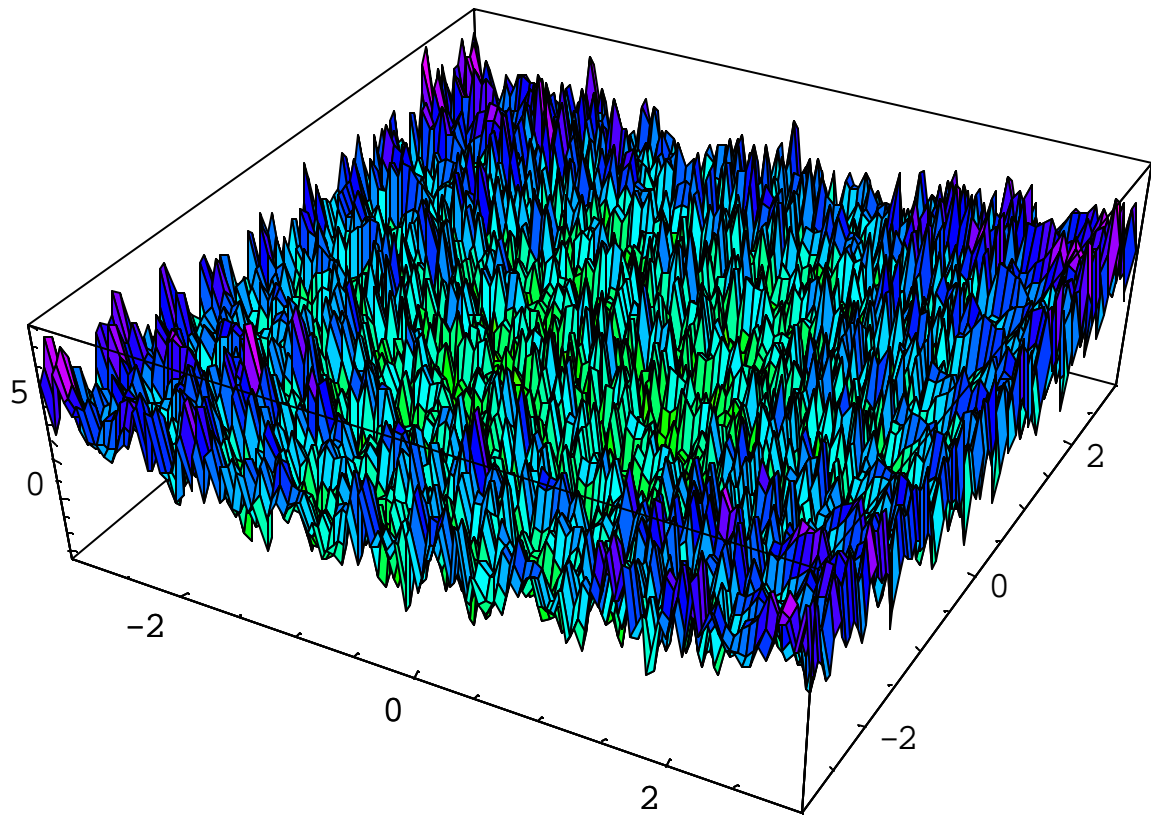


Figure 3.
LGO Integrated Development Environment.



Figure 4.
LGO for Excel: solver options and parameters dialog.

LGO Global Solver Options [?] [X]

Max Time:	<input type="text" value="1000"/>	seconds	<input type="button" value="OK"/>
Iterations:	<input type="text" value="10000"/>		<input type="button" value="Cancel"/>
Precision:	<input type="text" value="0.000001"/>		<input type="button" value="Integer Options..."/>
Convergence:	<input type="text" value="0.000001"/>		<input type="button" value="Load Model..."/>
Global Convergence:	<input type="text" value="0.00001"/>		<input type="button" value="Save Model..."/>
Global Phase Cutoff:	<input type="text" value="-1E+30"/>		<input type="button" value="Help"/>
Global Phase Iterations:	<input type="text" value="2000"/>		
Global Phase Iterations w/o Improvement:	<input type="text" value="2000"/>		
Local Phase Cutoff:	<input type="text" value="-1E+30"/>		

☐ Show Iteration Results
☐ Use Automatic Scaling
☐ Assume Non-Negative
☐ Bypass Solver Reports

LGO Search Options
☐ Global Branch & Bound
☒ Global Adaptive Random Search
☐ Local Search from Nominal Solution

Figure 5.

A randomly generated test problem solved by *MathOptimizer*.
(The surface plot and contour plot of the objective function are shown.)

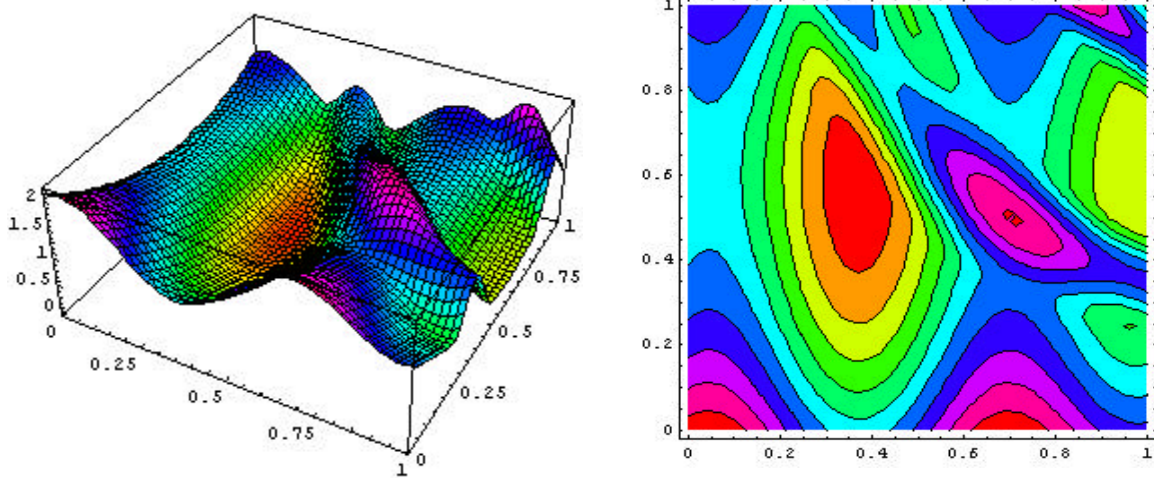


Figure 6.

Electrical circuit design model: Excel model form and solution by LGO engine.

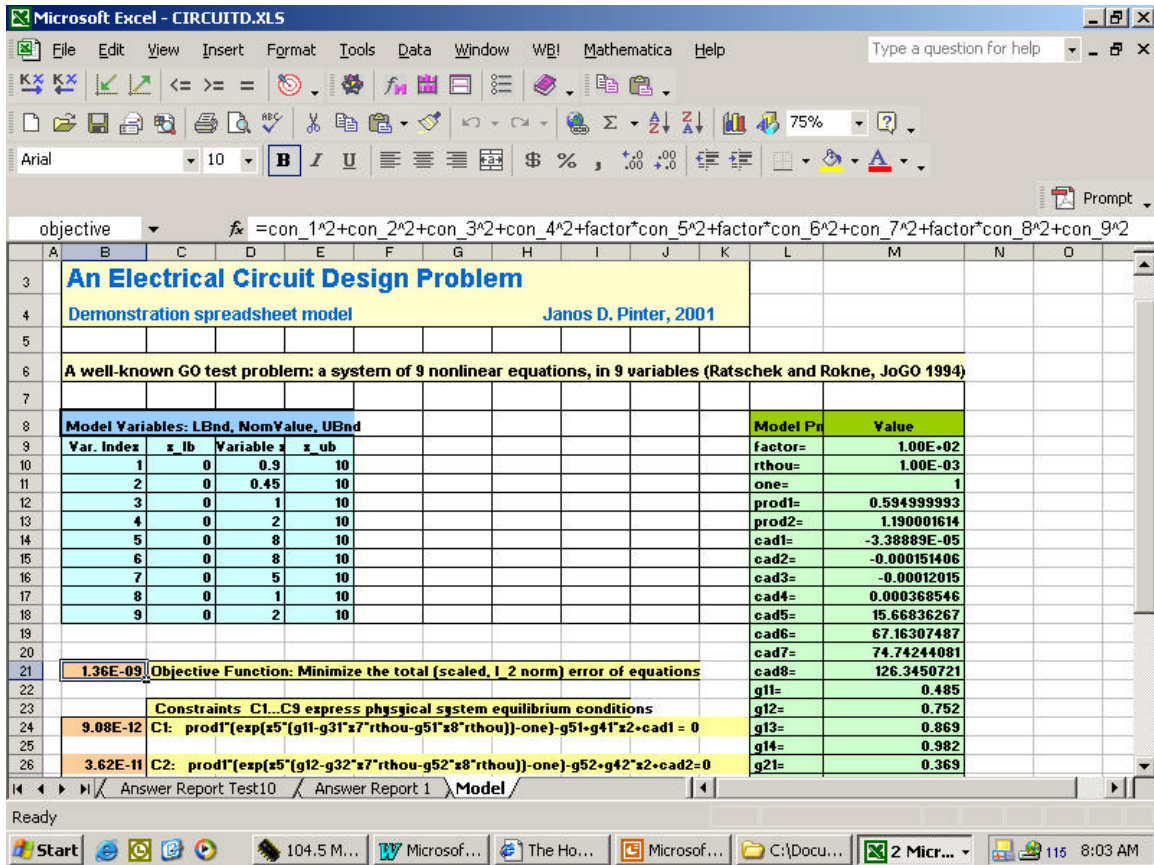


Figure 7.

Max-min model, for 13 points in 2 dimensions: a projection of the objective function.

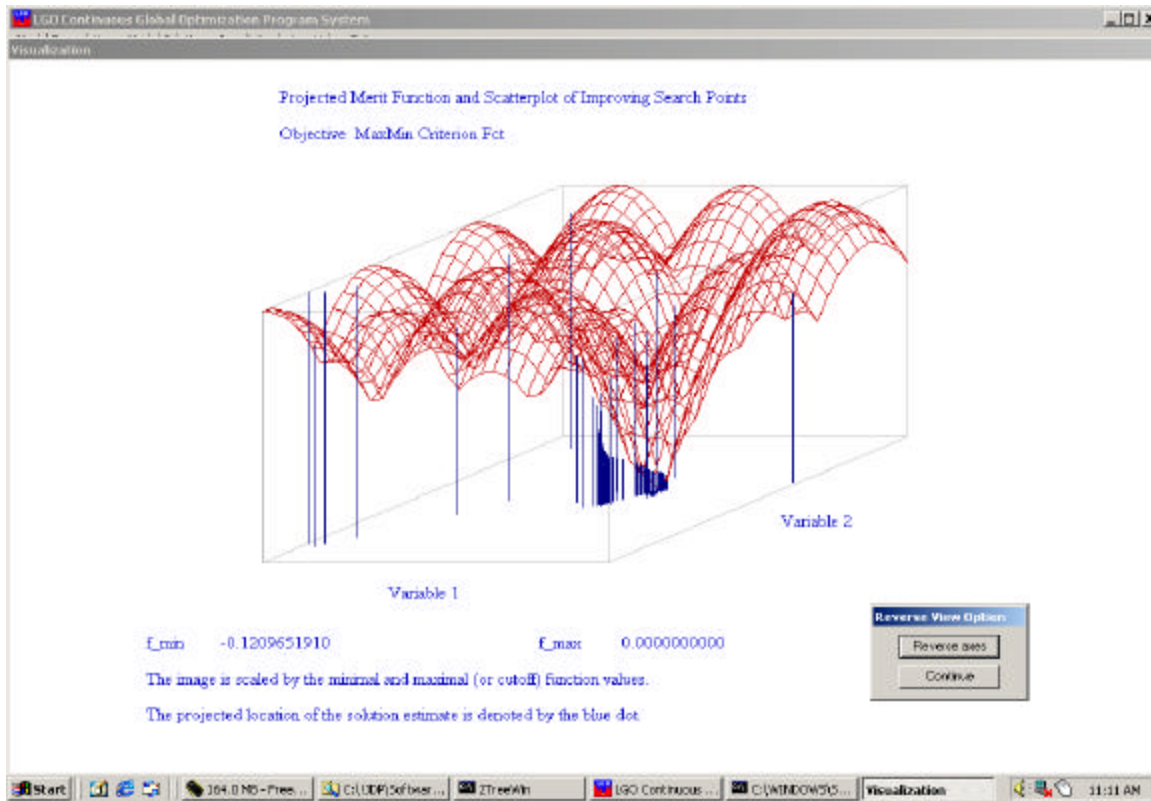
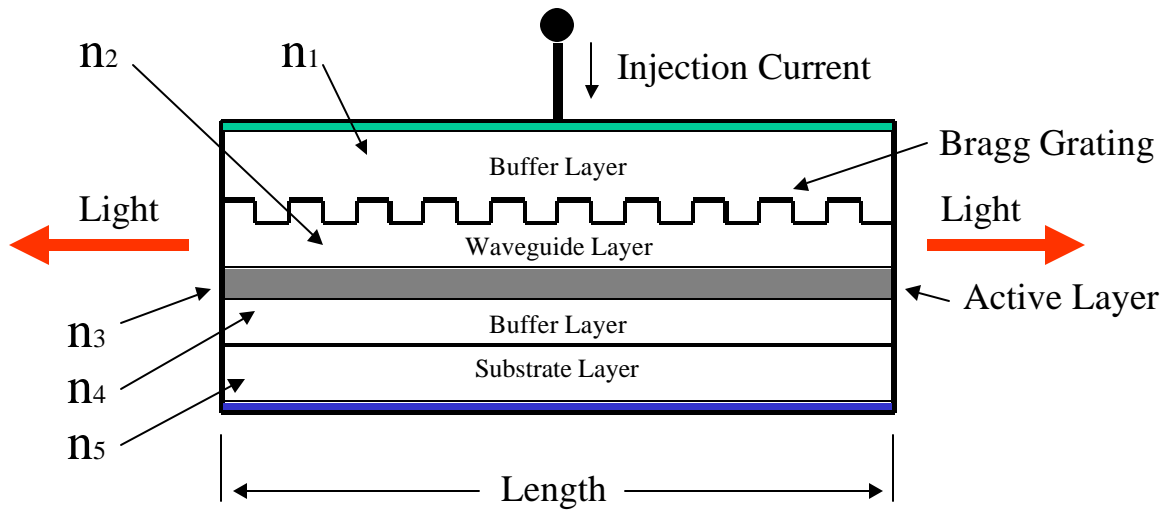


Figure 8.
Index-coupled, distributed feedback laser. (Source: Isenor, 2001.)



$$n_{1,2,3,4,5} = \text{Index of Refraction} \quad n_5 \leq n_1 < n_4 \leq n_2 < n_3$$

Figure 9.

Electric circuit model, developed to simulate a piezoelectric flex-tensional sonar projector. (Source: Moffett et al., 1992.)

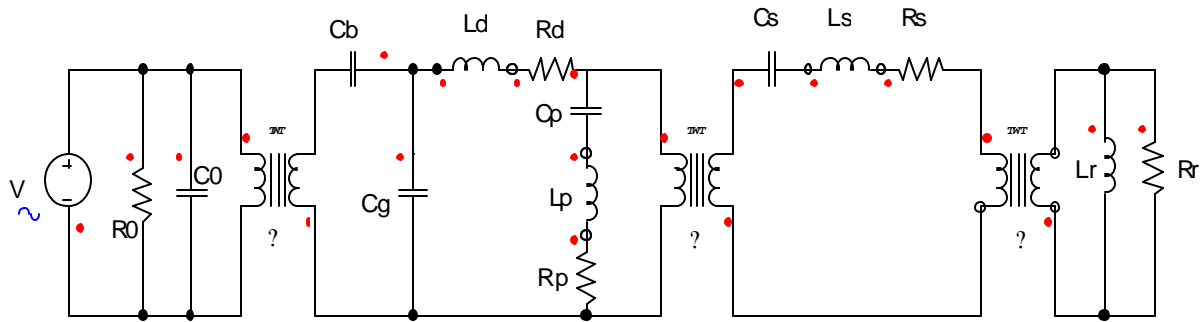


Figure 10.

Globally optimized radiation effect in numerical example.
(Source: J. Tervo's research group, University of Kuopio.)

