NONLINEAR OPTIMIZATION IN MODELING ENVIRONMENTS
Software Implementations for Compilers, Spreadsheets, Modeling Languages, and Integrated Computing Systems

JÁNOS D. PINTÉR
Pintér Consulting Services, Inc. *129 Glenforest Drive, Halifax, NS, Canada B3M 1J2*
jdpinter@hfx.eastlink.ca *http://www.pinterconsulting.com*

To appear in:

Jeyakumar and Rubinov, Eds., *Continuous Optimization: Current Trends and Applications.*

Springer Science + Business Media, New York, 2005.

Running title (header):
JÁNOS D. PINTÉR
NONLINEAR OPTIMIZATION IN MODELING ENVIRONMENTS

Chapter

NONLINEAR OPTIMIZATION IN MODELING ENVIRONMENTS

Software Implementations for Compilers, Spreadsheets, Modeling Languages, and Integrated Computing Systems

JÁNOS D. PINTÉR
Pintér Consulting Services, Inc. *129 Glenforest Drive, Halifax, NS, Canada B3M 1J2*
jdpinter@hfx.eastlink.ca *http://www.pinterconsulting.com*

Abstract: We present a review of several professional software products that serve to analyze and solve nonlinear (global and local) optimization problems across a variety of hardware and software environments. The product versions discussed have been implemented for compiler platforms, spreadsheets, algebraic (optimization) modeling languages, and for integrated scientific-technical computing systems. The discussion highlights some of the key advantages of these implementations. Test examples, well-known numerical challenges and client applications illustrate the usage of the current software implementations.

Key words: nonlinear (convex and global) optimization; LGO solver suite and its implementations; compiler platforms, spreadsheets, optimization modeling languages, scientific-technical computing systems; illustrative applications and case studies.

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

1. INTRODUCTION

Nonlinearity is literally ubiquitous in the development of natural objects, formations and processes, including also living organisms of all scales. Consequently, nonlinear descriptive models − and modeling paradigms even beyond a straightforward (analytical) function-based description − are of relevance in many areas of the sciences, engineering, and economics. For example, Bracken and McCormick (1968), Rich (1973), Eigen and Winkler (1975), Mandelbrot (1983), Murray (1983), Casti (1990), Hansen and Jørgensen (1991), Schroeder (1991), Bazaraa, Sherali, and Shetty (1993), Stewart (1995), Grossmann (1996), Pardalos, Shalloway, and Xue (1996), Pintér (1996a), Aris (1999), Bertsekas (1999), Gershenfeld (1999), Lafe (2000), Papalambros and Wilde (2000), Chong and Zak (2001), Edgar, Himmelblau and Lasdon (2001), Jacob (2001), Schittkowski (2002), Tawarmalani and Sahinidis (2002), Wolfram (2002), Diwekar (2003), Zabinsky (2003), Neumaier (2004b), Hillier and Lieberman (2005), Kampas and Pintér (2005), Pintér (2005) − as well as many other authors − present discussions and an extensive repertoire of examples to illustrate this point.

Decision-making (optimization) models that incorporate such a nonlinear system description frequently lead to complex models that (may or provably do) 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.

The most important (currently available) GO model types and solution approaches are discussed in the *Handbook of Global Optimization* volumes, edited by Horst and Pardalos (1995), and by Pardalos and Romeijn (2002). As of 2004, over a hundred textbooks and a growing number of informative web sites are devoted to this emerging subject.

We shall consider a general GO model form defined by the following ingredients:

- x decision vector, an element of the real Euclidean n-space $\boldsymbol{R}^n$;
- $f(x)$ continuous objective function, f: $\boldsymbol{R}^n$ ° $\boldsymbol{R}^1$;
- D non-empty set of admissible decisions, a proper subset of $\boldsymbol{R}^n$.

The feasible set D is defined by

- l, u explicit, finite vector bounds of x (a "box" in $\boldsymbol{R}^n$) ;
- $g(x)$$m$-vector of continuous constraint functions, g: $\boldsymbol{R}^n$ ° $\boldsymbol{R}^m$.

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

(1) min $f(x)$ s.t. x belongs to

(2) $D=\{x: \;\; l \le x \le u \;\;\; g(x) \le 0\}$.

Note that in (2) all vector inequalities are meant component-wise ($l, u,$ are n-vectors and the zero denotes an m-vector). Let us also remark that the set of the additional constraints g could be empty, thereby leading to – often much simpler, although still potentially multi-extremal – box-constrained models. Finally, note that formally more general optimization models (that include also = and ¥ constraint relations and/or explicit lower bounds on the constraint function values) can be simply reduced to the canonical model form (1)-(2). The canonical model itself is already very general: in fact, it trivially includes linear programming and convex nonlinear programming models (under corresponding additional specifications). Furthermore, it also includes the entire class of pure and mixed integer programming problems, since all (bounded) integer variables can be represented by a corresponding set of binary variables; and every binary variable y e {0,1} can be equivalently represented by its continuous extension y e [0,1] and the non-convex constraint $y(1\text{-}y) \le 0$. Of course, we do not claim that the above approach is best – or even suitable – for "all" optimization models: however, it certainly shows the generality of the CGO modeling framework.

Let us observe next that the above stated "minimal" analytical assumptions already guarantee that the optimal solution set X^* in the CGO model is non-empty. This key existence result directly follows by the classical theorem of Weierstrass (that states the existence of the minimizer point (set) of a continuous function over a non-empty compact set). For reasons of numerical tractability, the following additional requirements are also often postulated:

- D is a full-dimensional subset ("body") in $\boldsymbol{R}^n$;
- 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 the first two of these requirements support the development and (easier) implementation of globally convergent algorithmic search procedures. Specifically, the first assumption − i.e., the fact that D is the closure of its non-empty interior − makes algorithmic search possible within the set D. This requirement also implies that e.g., nonlinear equality constraints need to be directly incorporated into the objective function as discussed in Pintér (1996a), Chapter 4.1.

With respect to the second assumption, let us note that in most well-posed practical problems the set of global optimizers consists only of a single point, or at most of several points. However, in full generality, GO models may have even manifold solution sets: in such cases, software implementations will typically find a single solution, or several of them. (There are theoretically straightforward iterative ways to provide a sequence of global solutions.)

The third assumption is a sufficient condition for estimating f^* on the basis of a finite set of feasible search points. (Recall that the real-valued function h is Lipschitz-continuous on its domain of definition D**Õ**$\boldsymbol{R}^n$, if $h(x_1)$-$h(x_2)$ **§** $L||x_1$-$x_2||$ holds for all pairs x_1**e**D, x_2**e**D; here L=$L(D,h)$ is a suitable Lipschitz-constant of h on the set D: the inequality above directly supports lower bound estimates on sets of finite size.) We emphasize that the factual knowledge of the smallest suitable Lipschitz-constant − for each model function − is not required, and in practice such information is typically unavailable indeed.

Let us remark here that e.g., models defined by continuously differentiable functions f and g certainly belong to the CGO or even to the Lipschitz model class. In fact, even such "minimal" smooth structure is not essential: since e.g., "saw-tooth" like functions are also Lipschitz-continuous. This comment also implies that CGO indeed covers a very general class of optimization models. As a consequence of this generality, the CGO model class includes also many extremely difficult instances. To perceive this difficulty, one can think of model-instances that would require "the finding of the lowest valley across a range of islands" (since the feasible set may well be disconnected), based on an intelligent (adaptive, automatic), but otherwise completely "blind" sampling procedure…

For illustration, a merely one-dimensional, box-constrained model is shown in Figure 1. This is a frequently used classical GO test problem, due to Shubert: it is defined as

$$min \ ' \ _{k=1,\dots,5} \ k \ sin(k+(k+1)x) \qquad 10 \le x \le 10.$$

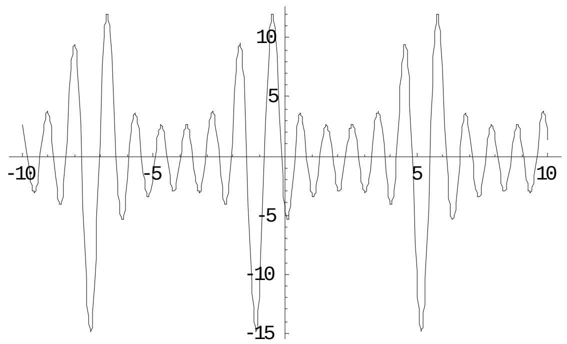


Figure 1
One-dimensional, box-constrained CGO model

Model complexity may – and frequently will – increase dramatically, already in (very) low dimensions. 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 and 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 multi-extremal models). To illustrate this point, consider the – merely two-dimensional, box-constrained, yet visibly challenging – objective function shown in Figure 2 below. The model is based on Problem 4 of the *Hundred-Dollar, Hundred-Digit Challenge Problems* by Trefethen (2002), and it is stated as

$$min\ (x^2+y^2)/4+ exp(sin(50x)) -sin(10(x+y))+sin(60exp(y))+sin(70sin(x))+sin(sin(80y))$$

$$-3\leq x\leq 3 \quad -3\leq y\leq 3.$$

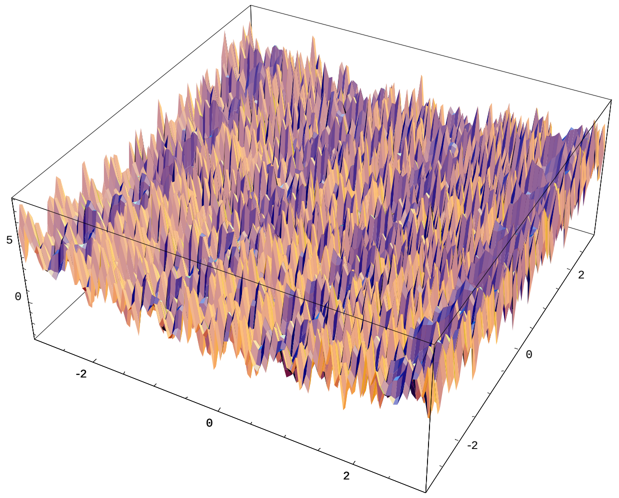


Figure 2
Two-dimensional, box-constrained CGO model

Needless to say, not all – and especially not all *practically motivated* – CGO models are as difficult as indicated by Figures 1 and 2. At the same time, we do not always have the possibility to directly inspect and estimate the difficulty of an optimization model, and perhaps unexpected complexity can be met under such circumstances. An important case in point is when the software user (client) has a confidential or otherwise visibly complex model that needs to be analyzed and solved. The model itself can be presented to the solver engine as an object code, dynamic link library (dll), or even as an executable program: in such situations, direct model inspection is simply not an option. In many other cases, the evaluation of the optimization model functions may require the numerical solution of a system of differential equations, the evaluation of special functions or integrals, the execution of a complex system of program code, stochastic simulation, even some physical experiments, and so on.

Traditional numerical optimization methods – discussed in most topical textbooks such as e.g. Bazaraa, Sherali, and Shetty (1993), Bertsekas (1999), Chong and Zak (2001) – search only for local optima. This approach is based on the tacit assumption that a "sufficiently good" initial solution (that is located in the region of attraction of the "true" solution) is immediately available. Both Figures 1 and 2 suggests that this may not always be a realistic assumption… Models with less "dramatic" difficulty, but in (perhaps much) higher dimensions also imply the need for global optimization. For instance, in advanced engineering design, models with hundreds or thousands of variables and constraints are analyzed. In similar cases to those mentioned above, even an approximately completed, but genuine global (exhaustive) search strategy may – and typically will – yield better results than the most sophisticated local search approach when "started from the wrong valley"…

2. A SOLVER SUITE APPROACH TO PRACTICAL GLOBAL OPTIMIZATION

The general development philosophy followed by the software implementations discussed here is based on the seamless combination of rigorous (i.e., theoretically convergent) global and efficient local search strategies.

As it is well-known (Horst and Tuy, 1996; Pintér, 1996a), 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. In other words, the application of a proper branch-and-bound search strategy (that exploits the Lipschitz information referred to above) generates a sequence of sample points that converges exactly to the (unique) global solution $x^*=\{X^*\}$ of the model instance considered. If the model has a finite or countable number of global solutions, then − theoretically, and under very general conditions − sub-sequences of search points are generated that respectively converge to the points of X^*. For further details related to the theoretical background, including also a detailed discussion of algorithm implementation aspects, consult Pintér (1996a) 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 generated search points tends to infinity. For instance, interval arithmetic based approaches follow this avenue: consult, e.g., Ratschek and Rokne (1995), Kearfott (1996), and Neumaier (2004b); Corliss and Kearfott (1999) review a number of successful applications of rigorous search methods.

The essential difficulty of applying such rigorous approaches to "all" GO models is that their computational demand typically grows at an exponential pace with the size of the models considered. For example, the Lipschitz information referred to above is often not precise enough: "carefree" overestimates of the best possible (smallest) Lipschitz-constant lead to a search procedure that will, in effect, be close in efficiency to a passive uniform grid search. For this reason, in a practical GO context, other search strategies also need to be considered.

It is also well-known that properly constructed stochastic search algorithms also possess general theoretical global convergence properties (with probability 1): consult, for instance, the review of Boender and Romeijn (1995), or Pintér (1996a). For a very simple example that illustrates this point, one can think of a pure random search mechanism applied in the interval $l \leq x \leq u$ 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. In addition, stochastic sampling methods can also be directly combined with search steps of other − various global and efficient local − search strategies, and the overall global convergence of such strategies will be still maintained. The theoretical background of stochastic "hybrid" algorithms is discussed by Pintér (1996a). 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, and also can design various interactive runs.

Obviously, there remains a significant issue regarding the (typically unforeseeable best) "switching point" from strategy to strategy: this is however, unavoidable, when choosing between theoretical rigor and numerical efficiency. (Even local nonlinear solvers would need, in theory, an infinite iterative procedure to converge, except in idealized special cases…) 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 x^*, and then that solution estimate could be refined by a suitably robust and efficient local solver. Of course, the region of attraction of x^* (e.g., its shape and relative size) 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). Another important numerical aspect is that one loses the deterministic (lower) bound guarantees when applying a stochastic search procedure: instead, suitable statistical estimation methods can be applied, consult Pintér (1996a) and topical references therein. Again, the implementation of such methodology is far from trivial.

To summarize the discussion, there are good reasons to apply various search methods and heuristic global-to-local search "switching points" with a reasonable expectation of numerical success. Namely,

- one needs to apply proper global search methods to generate an initial good "coverage" of the search space
- it is also advantageous to apply quality local search that enables the fast improvement of solution estimates generated by a preceding global search phase
- using several – global or local – search methods based on different theoretical strategies, one has a better chance to find quality solutions in difficult models (or ideally, confirm the solution by comparing the results of several solver runs)
- one can always place more or less emphasis on rigorous search vs. efficiency, by selecting the appropriate solver combination, and by allocating search effort (time, function evaluations);
- 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 directly started);
- practical circumstance and resource limitations may (will) dictate the use of additional numerical stopping and switching rules that can be flexibly built into the software implementation.

Based on the design philosophy outlined – that has been further confirmed and dictated by practical user demands – we have been developing for over a decade nonlinear optimization software implementations that are based on global and local solver combinations. The currently available software products will be briefly discussed below with illustrative examples; further related work is in progress.

3. MODELING SYSTEMS AND USER DEMANDS

Due to advances in modeling, optimization methods and computer technology, there has been a rapidly growing interest towards modeling languages and environments. Consult, for example, the topical *Annals of Operations Research* volumes edited by Maros and Mitra (1995), Maros, Mitra, and Sciomachen (1997), Vladimirou, Maros, and Mitra (2000), Coullard, Fourer, and Owen (2001), and the volume edited by Kallrath (2004). Additional useful information can be found, for example, at the web sites maintained by Fourer (2004), Mittelmann and Spellucci (2004), and Neumaier (2004a).

Prominent examples of widely used modeling systems that are focused on optimization include AIMMS (Paragon Decision Technology, 2004), AMPL (Fourer, Gay, and Kernighan, 1993), GAMS (Brooke, Kendrick, and Meeraus, 1988), the Excel Premium Solver Platform (Frontline Systems, 2001), ILOG (2004), the LINDO Solver Suite (LINDO Systems, 1996), MPL (Maximal Software, 2002), TOMLAB (2004). (Please note that the literature references cited may not always reflect the current status of the modeling systems listed: for the latest information, contact the developers and/or visit their website.)

In addition, there exists also a large variety of core compiler platform-based solver systems with some built-in model development functionality: in principle, these all can be linked to the modeling languages listed above. At the other end of the spectrum, there is also significant development related to fully integrated scientific and technical computing (ISTC) systems such as Maple (Maplesoft, 2004), *Mathematica* (Wolfram Research, 2004), and MATLAB (The MathWorks, 2004). The ISTCs also incorporate a growing range of optimization-related functionality, supplemented by application products.

The modeling environments listed above are aimed at meeting the needs and demands of a broad range of clients. Major client groups 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, vertical application developers, and other "power users". Obviously, the user categories listed above are not necessarily disjoint: e.g., someone can be an expert researcher and software developer in a certain professional area, with a perhaps more modest optimization expertise. The pros and cons of the individual software products – in terms of ease of model prototyping, detailed code development and maintenance, optimization model processing tools, availability of solvers and other auxiliary tools, program execution speed, overall level of system integration, quality of related documentation and support – make such systems more or less attractive for these user groups.

It is also worth mentioning at this point that – especially in the context of nonlinear modeling and optimization – it can be a salient idea to tackle challenging problems by making use of several modeling systems and solver tools, if available. In general, dense NLO model formulations are far less easy to "standardize" than linear or even mixed integer linear models, since one typically needs an explicit, specific formula to describe a particular model function. Such formulae are relatively straightforward to transfer from one modeling system into another: some of the systems listed above even have such built-in converter capabilities, and their syntaxes are typically quite similar (whether it is x**2 or x^2, sin(x) or Sin[x], bernoulli(n,x) or BernoulliB[n,x], and so on).

In subsequent sections we shall summarize the principal features of several current nonlinear optimization software implementations that have been developed with quite diverse user groups in mind. The range of products reviewed in this paper includes the following:

- LGO Solver System with a Text I/O Interface
- LGO Integrated Development Environment
- LGO Solver Engine for Excel
- MathOptimizer Professional (LGO Solver Engine for *Mathematica*)
- Maple Global Optimization Toolbox (LGO Solver Engine for Maple)

We will also present relatively small, but non-trivial test problems to illustrate some key functionality of these implementations.

Note that all software products discussed 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 technical information is available upon request; please consult also the publicly available references, including the software documentation and topical web sites.

In order to keep the length of this article within reasonable bounds, further product implementations not discussed here are

- LGO Solver Engine for GAMS

- LGO Solver Engine for MPL
- TOMLAB/LGO for MATLAB
- MathOptimizer for *Mathematica*

With respect to these products, consult e.g. the references Pintér (2002a, 2003), Kampas and Pintér (2004b, 2005), Pintér, Holmström, Göran and Edvall (2004), Pintér and Kampas (2005).

4. SOFTWARE IMPLEMENTATION EXAMPLES

4.1. LGO Solver System with a Text I/O Interface

The Lipschitz Global Optimizer (LGO) software has been developed and used for more than a decade (as of 2004). Detailed technical descriptions and user documentation have appeared elsewhere: consult, for instance, Pintér (1996a, 1997, 2001a, 2004), and the software review by Benson and Sun (2000). Let us also remark here that LGO was chosen to illustrate global optimization software (in connection with a demo version of the MPL modeling language) in the well-received textbook by Hillier and Lieberman (2005).

Since LGO serves as the core of most current implementations (with the exception of one product), we will provide its somewhat more detailed description, followed by concise summaries of the other platform-specific implementations.

In accordance with the approach advocated in Section 2, LGO is based on a seamless combination of a suite of global and local scope nonlinear solvers. Currently, LGO includes the following solver options:

- adaptive partition and search (branch-and-bound) based global search (BB)
- adaptive global random search (single-start) (GARS)
- adaptive global random search (multi-start) (MS)
- constrained local search (generalized reduced gradient method) (LS).

The global search methodology was discussed briefly in Section 2; the well-known GRG method is discussed in numerous textbooks, consult e.g. Edgar, Himmelblau, and Lasdon (2001). Note that in all three global search modes the model functions are aggregated by an exact penalty (aggregated merit) function. By contrast, in the local search phase all model functions are considered and treated individually. Note also that the global search phases are equipped with stochastic sampling procedures that support the usage of statistical bound estimation methods.

All LGO search algorithms are derivative-free: specifically, in the local search phase central differences are used to approximate gradients. This choice reflects again our objective to handle (also) models with merely computable, continuous functions, including ‘black box’ systems.

The compiler-based LGO solver suite is used as an option linked to various modeling environments. In its core text I/O based version, the application-specific LGO executable program (that includes a driver file and the model function file) reads an input text file that contains all remaining application information (model name, variable and constraint names, variable bounds and nominal values, and constraint types), as well as a few key solver options (global solver type, precision settings, resource and time limits). Upon completing the LGO run, a summary and a detailed report file are available. As can be expected, this LGO version has the lowest demands for hardware, it also runs fastest, and it can be directly embedded into vertical and proprietary user applications.

4.2. LGO Integrated Development Environment

LGO can be also equipped − as a readily available implementation option − with a simple, but functional and user-friendly MS Windows interface. This enhanced version is referred to as the LGO Integrated Development Environment (IDE). The LGO IDE provides a menu environment that supports model development, compilation, linking, execution, and the inspection of results. To this end, a text editor is used that can be chosen optionally such as e.g. the freely downloadable ConTEXT and PFE editors, or others. Note here that even the simple Notebook Windows accessory − or the more sophisticated and still free Metapad text editor − would do. The IDE also includes external program call options and two concise help files: the latter discuss global optimization basics and the main application development steps when using LGO.

As already noted, this LGO implementation is compiler-based: user models can be connected to LGO using one of several programming languages on personal computers and workstations. Currently supported platforms include essentially all professional Fortran 77/90/95 and C compilers and some others: prominent examples are Borland C/C++ and Delphi, Compaq/Digital Visual Fortran; Lahey Fortran 77/90/95; Microsoft Visual Basic and C/C++; and Salford Fortran 77/95. 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 discussed above) fully supports communication with sophisticated user models, including entirely closed or confidential “black box” systems. These LGO versions are particularly advantageous in application areas, where program execution (solution) speed is a major concern: in the GO context, many projects fall into this category. The added features of the LGO IDE can also greatly assist in educational and research (prototyping) projects.

LGO deliveries are accompanied by an approximately 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.

For a simple illustration, we display below the LGO model function file (in C format), and the input parameter file that correspond to a small, but not quite trivial GO model (this is a constrained extension of Shubert’s model discussed earlier):

$$\min \;{}' \;{}_{k=1,\ldots,5}\, k\, \sin(k+(k+1)x) \qquad 10\le x\le 10.$$
$$x^2+3x+5\sin(x)\le 6$$

Both files are slightly edited for the present purposes. Note also that in the simplest usage mode, the driver file contains only a single statement that calls LGO: therefore we skip the display of that file. (Additional pre- and post-solver manipulations can also be inserted in the driver file: this can be useful in various customized applications.)

Model function file

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>

_stdcall USER_FCT( double x[], double fox[1], double gox[])
{
```

```
fox[0] = sin(1. + 2.*x[0]) + 2.*sin(2. + 3.*x[0]) + 3.*sin(3. + 4.*x[0]) +
4.*sin(4. + 5.*x[0]) + 5.*sin(5. + 6.*x[0]);
gox[0] = -6. + pow(x[0],2.) + sin(x[0]) + 3.*x[0];
return 0;
}
```

Input parameter file

```
---Model Descriptors---
LGO Model           ! ModelName
1                   ! Number of Variables
1                   ! Number of Constraints
Variable names      ! Lower Bounds        Nomimal Values      Upper Bounds
x                   -10.                0.                  10.
ObjFct              ! Objective Function Name
! Constraint Names and Constraint Types (0 for ==, -1 for <=)
Constraint1           -1

!--- SOLVER OPTIONS AND PARAMETERS ---
1                   ! Operational modes 0: LS; 1: BB+LS; 2: GARS+LS; 3: MS+LS
2000                ! Maximal no. of fct evals in global search phase
400                 ! Maximal no. of fct evals in global search w/o improvement
1.                  ! Constraint penalty multiplier
-1000000.           ! Target objective function value in global search phase
-1000000.           ! Target objective function value in local search phase
0.000001            ! Merit function precision improvement threshold in local
                    ! search phase
0.000001            ! Constraint violation tolerance in local search phase
0.000001            ! Kuhn-Tucker local optimality conditions tolerance in
                    ! local search phase
0                   ! Built-in random number generator seed value
300                 ! Program execution time limit (seconds)
```

Summary result file

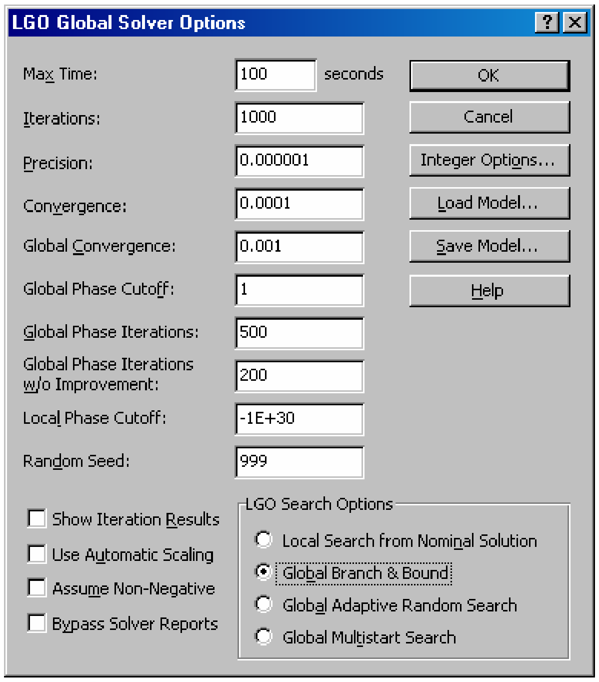
```
--- LGO Solver Results Summary ---

Model name: LGO Model

Total number of function evaluations           997

Objective function: ObjFct                    -14.8379500257

Solution vector components
1                                 x            -1.1140996879

Constraint function values at optimum estimate
1   Constraint1                               -8.9985950759

Solver status indicator value     4   TERMINATED BY SOLVER

Model status indicator value      1   GLOBALLY OPTIMAL SOLUTION FOUND

LGO solver system execution time (seconds)     0.01

For additional runtime information, please consult the LGO.OUT file.

--- LGO application run completed. ---
```

4.3. LGO Solver Engine for Excel Users

The LGO global solver engine for Microsoft Excel has been developed in cooperation with Frontline Systems (2001). For details on the Excel Solver and the currently available advanced engine options visit Frontline's web site (www.solver.com). The site contains useful information, including for instance, tutorial material, modeling tips, and various spreadsheet examples. The User Guide provides a brief introduction to all current solver engines; discusses the diagnosis of solver results, solver options and reports; and it also contains a section on Solver VBA functions. Note that this information can also be invoked through Excel's on-line help system. In this 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, in addition to continuous global and local capabilities, 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 some modifications and added features. The LGO Solver Options dialog shown on Figure 3 allows the user to control solver choices and several other settings.



Max Time:	100 seconds
Iterations:	1000
Precision:	0.000001
Convergence:	0.0001
Global Convergence:	0.001
Global Phase Cutoff:	1
Global Phase Iterations:	500
Global Phase Iterations w/o Improvement:	200
Local Phase Cutoff:	-1E+30
Random Seed:	999

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

LGO Search Options
○ Local Search from Nominal Solution
● Global Branch & Bound
○ Global Adaptive Random Search
○ Global Multistart Search

Figure 3
Excel/ LGO solver engine: solver options and parameters dialog

To illustrate the usage of the Excel/LGO implementation, we shall present and solve the *Electrical Circuit Design* (ECD) test problem. The ECD 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 the ECD problem, a bipolar transistor is modeled by an electrical circuit: 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 individual equations are defined as follows:

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

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

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

By assumption, the vector variable x belongs to the box region $[0,10]^{\mathbf{9}}$. The numerical values of the constants g_{ik} i=1,…,5, k=1,…,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 ECD model rigorously, Ratschek and Rokne applied a combination of interval arithmetic, subdivision and branch-and-bound strategies. They concluded that the rigorous solution was extremely costly (billions of model function evaluations were needed), 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, the problem of finding the solution can be formulated as a global optimization problem. This model has been set up in a demo spreadsheet, and then solved by the Excel LGO solver engine. The numerical solution found by LGO – directly imported from the answer report – is shown below:

```
Microsoft Excel 10.0 Answer Report
Worksheet: [CircuitDesign_9_9.XLS]Model
Report Created: 12/16/2004 12:39:29 AM
Result: Solver found a solution.  All constraints and optimality conditions are satisfied.

Engine: LGO Global Solver

Target Cell (Min)
        Cell    Name            Original Value  Final Value
        $B$21   objective       767671534.2     9.02001E-11

Adjustable Cells
        Cell    Name            Original Value  Final Value
        $D$10   x_1             1               0.900000409
        $D$11   x_2             2               0.450000021
        $D$12   x_3             3               1.000000331
        $D$13   x_4             4               2.000001476
        $D$14   x_5             5               7.999999956
        $D$15   x_6             6               7.999998226
        $D$16   x_7             7               4.999999941
        $D$17   x_8             8               1.000000001
        $D$18   x_9             9               1.999999812
```

The error of the solution found is within 10^{-6} to the verified solution, for each component. The numerical solution of the ECD model in Excel takes less than 5 seconds on a personal computer (Intel Pentium 4, 2.4 GHz processor, 512 Mb RAM). Let us note that we have solved this model also using core LGO implementations with various C and Fortran compilers, with essentially identical success. Although this finding should not lead to overly optimistic claims, it certainly shows the robustness and efficiency of LGO on this particular (non-trivial) example.

4.4. MathOptimizer Professional

Mathematica is an integrated environment for scientific and technical computing. This ISTC system supports functional, rule-based and procedural programming styles. *Mathematica* also offers advanced multimedia (graphics, image processing, animation, sound generation) tools, and it can be used to produce publication-quality documentation. For further information, consult the key reference Wolfram (2003); the website www.wolfram.com provides detailed information regarding also the range of other products and services related to *Mathematica*.

MathOptimizer Professional (Pintér and Kampas, 2003) combines the model development power of *Mathematica* with the robust performance and efficiency of the LGO solver suite. To this end, *MathLink* a general-purpose interface is used that supports communication between *Mathematica* and external programs. The functionality of *MathOptimizer Professional* is summarized by the following stages (note that all steps are fully automatic, except – obviously – the first one):

- model formulation in *Mathematica*
- translation of the *Mathematica* optimization model into C or Fortran code, to generate the LGO model function file
- generation of LGO input parameter file
- compilation of the C or Fortran model code into object code or dynamic link library (dll): this step makes use of a corresponding compiler
- call to the LGO solver engine: the latter is typically provided as object code or an executable program that is now linked together with the model function object or dll file
- numerical solution and report generation by LGO
- report of LGO results back to the calling *Mathematica* notebook.

A "side-benefit" of using *MathOptimizer Professional* is that the *Mathematica* models formulated are automatically translated into C or Fortran format: this feature can be put to good use in a variety of contexts. (For example, the LGO model function and input parameter file examples shown in Section 4.2 were generated automatically.)

Let us also remark that the approach outlined supports "only" the solution of models defined in *Mathematica* that can be directly converted into C or Fortran program code. Of course, this model category still allows the handling of a broad range of optimization problems. The approximately 150-page *MathOptimizer Professional* manual is a "live" document (notebook) that can be directly invoked through *Mathematica*'s on-line help system. In addition to basic usage description, the User Guide also discusses a large number of simple and more challenging test problems, and several realistic application examples in detail.

As an illustrative example, we will present the solution of a new – and rather difficult – object packing model: we wish to find (numerically) the "best' non-overlapping arrangement of a set of non-uniform size circles in an embedding circle. Notice that this is not a standard model type (unlike uniform circle packings that have been studied for decades, yet still only special cases are solved to guaranteed optimality). Our approach can be directly generalized to find essentially arbitrary object arrangements.

The best packing is defined here by a combination of two criteria: the radius of the circumscribed circle, and the average pair-wise distance between the centers of the embedded circles. The relative weight of the two objective function components can be selected as a model-instance parameter.

Detailed numerical results are reported by Kampas and Pintér, (2004a) for circles defined by the sequence of radii $r_i=i^{-0.5}$, $i=1,\dots,N$, up to $N=40$-circle configurations. Observe that the required (pair-wise) non-overlapping arrangement leads to $N(N-1)/2$ non-convex constraints, in addition to

2N+1 bound constraints on the circle center and circumscribed radius decision variables. Hence, in the 40-circle example, LGO solves this model with nearly 780 non-convex constraints: the corresponding runtime was about 3.5 hours on a P4 1.6 GHz personal computer.

As an illustration, the configuration found for the case of N=20 circles is displayed below. In this example, equal consideration (weight) is given to minimizing the radius of the circumscribed circle and the average distance between the circle centers. As the picture shows, the circumscribed radius is about 2.2: in fact, the numerical value found is ~2.1874712123. Detailed results appeared and will appear in Kampas and Pintér (2004a, 2005).

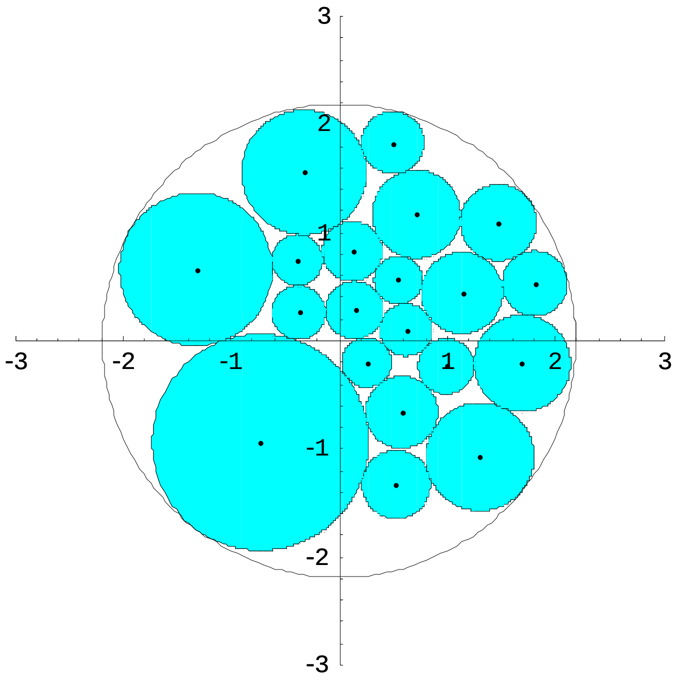


Figure 4
An illustrative non-uniform circle packing result for N=20 circles with radii r_i=$i^{-0.5}$, i=1,…,N

Let us also remark that we have attempted to solve instances of the same circle packing problem applying the built-in *Mathematica* function NMinimize for nonlinear (global) optimization, but – using it in all of its default solver modes – it could not find a solution of acceptable quality already for the case N=5. Again, this is just a numerical observation, as opposed to an "all-purpose" conclusion, to show the merits of the LGO solver suite. We have also conducted detailed numerical studies that provide a more systematic comparison of global solvers available for use with *Mathematica*: these results will appear in Kampas and Pintér (2005).

Finally, let us mention that *MathOptimizer Professional* is included in a recent peer review of optimization capabilities using *Mathematica* (Cogan, 2003).

4.5. Maple Global Optimization Toolbox

The integrated computing environment Maple (Maplesoft, 2004a) enables the development of sophisticated interactive documents that seamlessly combine technical description, calculations, simple and advanced computing, and visualization. Maple includes an extensive mathematical library: its more than 3,500 built-in functions cover virtually all research areas in the scientific and technical disciplines. Maple also incorporates numerous supporting features and enhancements such as e.g. detailed on-line documentation, a built-in mathematical dictionary with definitions for more than 5000 mathematical terms, debugging tools, automated (ANSI C, Fortran 77, Java, Visual Basic and MATLAB) code generation, and document production (including HTML, MathML, TeX, and RTF converters). All these capabilities accelerate and expand the scope of optimization model development and solution.

To emphasize the key features pertaining to advanced systems modeling and optimization, a concise listing of these capabilities is provided below. Maple

- supports rapid prototyping and model development
- performance scales well to modeling large, complex problems
- offers context-specific “point and click” (essentially syntax-free) operations, including various “Assistants” (these are windows and dialogs that help to execute various tasks)
- has an extensive set of built-in mathematical and computational functions
- has comprehensive symbolic calculation capabilities
- supports advanced computations with arbitrary numeric precision
- is fully programmable, thus extendable by adding new functionality
- has sophisticated visualization and animation tools
- supports the development of GUIs (by using “Maplets”)
- supports advanced technical documentation, desktop publishing, and presentation
- provides links to external software products

Maple is portable across all major hardware platforms and operating systems (Windows, Macintosh, Linux, and Unix versions). Without going into further details that are outside of the scope of the present discussion, we refer to the web site www.maplesoft.com that provides a wealth of further topical information and product demos.

The core of the recently released Maple Global Optimization Toolbox (GOT) is a customized implementation of the LGO solver suite (Maplesoft, 2004b). To this end, LGO was auto-translated into C code, and then fully integrated with Maple. The advantage of this approach is that, in principle, the GOT can handle all (thousands) of functions that are defined in Maple, including their further extensions.

As an illustrative example, let us revisit Problem 4 posted by Trefethen (2002); recall Figure 2 from Section 1. We can easily set up this model in Maple:

```
> f := exp(sin(50*x1))+sin(60*exp(x2))+sin(70*sin(x1))+sin(sin(80*x2))-
sin(10*(x1+x2))+(x1^2+x2^2)/4;
```

$$f := \mathbf{e}^{\sin(50\,x1)} + \sin(60\,\mathbf{e}^{x2}) + \sin(70\sin(x1)) + \sin(\sin(80\,x2)) - \sin(10\,x1 + 10\,x2) + \frac{1}{4}\,x1^2 + \frac{1}{4}\,x2^2$$

Now using the bounds [-3,3] for both variables, and applying the Global Optimization Toolbox we receive the numerical solution:

```
> GlobalSolve(f, x1=-3..3, x2=-3..3, evaluationlimit=100000,
noimprovementlimit=100000);
```

$$[-3.30686864747523535, [x1 = -0.0244030794174338178, x2 = 0.210612427162285371]]$$

We can compare the optimum estimate found to the corresponding 40-digit precision value as stated at the website http://web.comlab.ox.ac.uk/oucl/work/nick.trefethen/hundred.html (of Trefethen). The website provides the 40-digit numerical optimum value -3.306868647 4752372800 7611377089 8515657166... Hence, the solution found by the Maple GOT (using default precision settings) is accurate to 15 digits.

It is probably just as noteworthy that one can find a reasonably good solution even in a much larger variable range, with the same solution effort:

```
> GlobalSolve(f, x1=-100..100, x2=-100..100, evaluationlimit=100000,
noimprovementlimit=100000);
```

```
[-3.06433688275856530, [x1 = -0.233457978266705634e-1, x2 = .774154819772443825]]
```

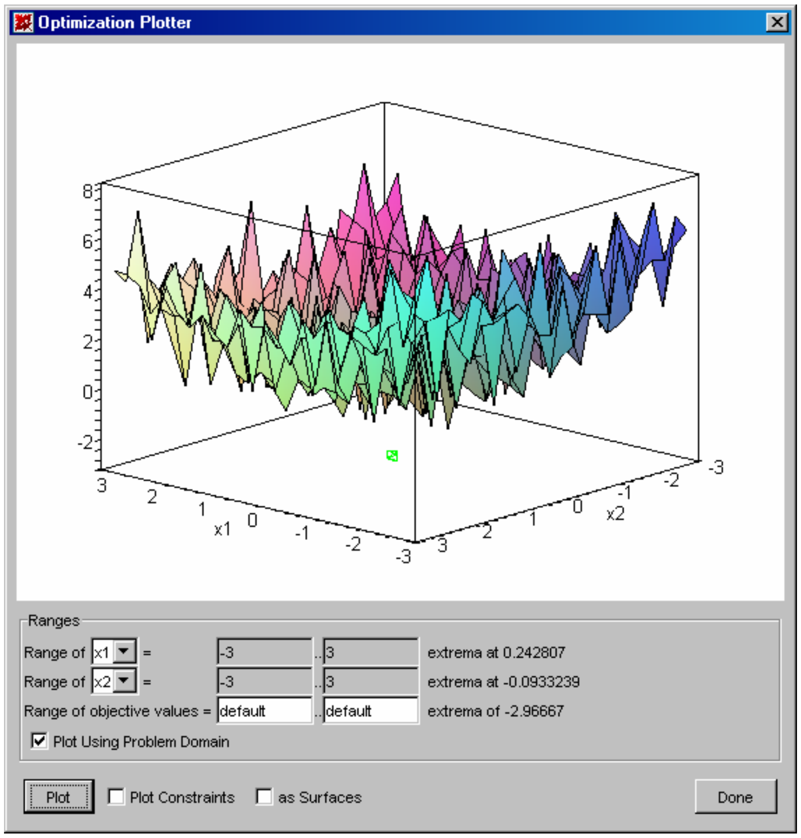
A partial explanation is that the shape of the objective function *f* is close to quadratic, at least "from a distance". Note at the same time that the local solver produces much inferior results on the larger region (and it also misses the global optimum when using the variable bounds [-3,3], as can be expected):

```
> Minimize(f, x1=-100..100, x2=-100..100);
```

```
[-.713074709310511201, [x1 = -0.223022309405313465e-1, x2 = -0.472762143202519123e-2]]
```

The corresponding GOT runtimes are a little more than one second in both cases. (Note that all such runtimes are approximate, and may vary a bit even between consecutive test runs, depending on the machine's actual runtime environment).

One of the advantages of using ISTCs that one can visualize models and verify their perceived difficulty. Figure 5 is based on using the Maple Optimization Plotter dialog, a feature that can be used in conjunction with the GOT: it shows the box-constrained Trefethen model in the range $[-3,3]^2$; observe also the location of the optimal solution (green dot).



Range of x1 = -3 .. 3 extrema at 0.242807
Range of x2 = -3 .. 3 extrema at -0.0933239
Range of objective values = default .. default extrema of -2.96667

Figure 5
Trefethen's Problem 4 solved and visualized using the Maple GOT

5. FURTHER APPLICATIONS

For over a decade, LGO has been applied in a variety of professional, as well as academic research and educational contexts (in some 20 countries, as of 2004). In recent years, LGO has been used to solve models in up to a few thousand variables and constraints. The software seems to be particularly well-suited to analyze and solve complex, sophisticated applications in advanced engineering, biotechnology, econometrics, financial modeling, process industries, medical studies, and in various other areas of scientific modeling.

Without aiming at completeness, let us refer to some recent (published) applications and case studies that are related to the following areas:

- model calibration (Pintér, 2003a)
- potential energy models in computational chemistry (Pintér, 2000, 2001b), (Stortelder, de Swart, and Pintér, 2001)
- laser design (Isenor, Pintér, and Cada, 2003)
- cancer therapy planning (Tervo, Kolmonen, Lyyra-Laitinen, Pintér, and Lahtinen, 2003)
- combined finite element modeling and optimization in sonar equipment design (Pintér and Purcell, 2003)
- Configuration analysis and design (Kampas and Pintér, 2004b).

Note additionally that some of the LGO software users develop other advanced (but confidential) applications. Articles and numerical examples, specifically related to various LGO implementations are available from the author upon request. The forthcoming volumes (Kampas and Pintér, 2005; Pintér, 2005a, b) also discuss a large variety of GO applications, with extensive further references.

6. CONCLUSIONS

In this paper, a review of several nonlinear optimization software products has been presented. We introduce the LGO solver suite, and then briefly review several of its currently available implementations for use with compiler platforms, spreadsheets, optimization modeling languages, and ISTCs (such as *Mathematica* and Maple). It is our objective to add customized functionality to the existing products, and to develop further implementations, in order to meet the needs of a broad range of users.

Global optimization is and will remain a field of extreme numerical difficulty, not only when considering “all possible” GO models, but also in practical attempts to handle complex, sizeable problems in an acceptable timeframe. Therefore the discussion advocates a practically motivated approach that combines rigorous global optimization strategies with efficient local search methodology, in integrated, flexible solver suites. The illustrative − yet non-trivial − application examples and the numerical results show the practical merits of such an approach.

We are interested to learn suggestions regarding future development directions. Test problems and challenges – as well as prospective application areas – are welcome.

ACKNOWLEDGEMENTS

First of all, I wish to thank my developer partners and colleagues for their cooperation and many useful discussions, quality software, documentation, and technical support: these partners include AMPL LLC, Frontline Systems, the GAMS Development Corporation, Lahey Computer Systems, LINDO Systems, Maplesoft, Maximal Software, Paragon Decision Technology, TOMLAB AB, and Wolfram Research.

Several application examples reviewed or cited in this paper are based on cooperation with (a large number of) colleagues: all such cooperation is gratefully acknowledged and is duly reflected by the references cited.

In addition to professional contributions and in-kind support offered by developer partners, the work summarized and reviewed in this paper has received financial support from the following organizations: DRDC Atlantic Region, Canada (Contract W7707-01-0746), the Dutch Technology Foundation (STW Grant CWI55.3638), the Hungarian Scientific Research Fund (OTKA Grant T 034350), Maplesoft, the National Research Council of Canada (NRC IRAP Project 362093), the University of Ballarat, Australia; the University of Kuopio, Finland; and the University of Tilburg, Netherlands.

REFERENCES

Aris, R. (1999) *Mathematical Modeling: A Chemical Engineer's Perspective*. Academic Press, San Diego, CA..

Bazaraa, M.S., Sherali, H.D. and Shetty, C.M. (1993) *Nonlinear Programming: Theory and Algorithms.* Wiley, New York.

Benson, H.P., and Sun, E. (2000) LGO – Versatile tool for global optimization. *OR/MS Today 27 (5)*, 52-55.

Bertsekas, D.P. (1999) *Nonlinear Programming.* (2nd Edition.) Athena Scientific, Cambridge, MA.

Boender, C.G.E. and Romeijn, H.E. (1995) Stochastic methods. In: Horst and Pardalos, Eds. *Handbook of Global Optimization. Volume 1*, pp. 829-869.

Bornemann, F., Laurie, D., Wagon, S., and Waldvogel, J. (2004) *The SIAM 100-Digit Challenge. A Study in High-Accuracy Numerical Computing*. SIAM, Philadelphia, PA.

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.

Cogan, B. (2003) How to get the best out of optimization software. *Scientific Computing World* 71 (2003) 67-68. See http://www.scientific-computing.com/scwjulaug03review_ optimisation.html.

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, Boston / Dordrecht /London.

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, Boston / Dordrecht /London.

Chong, E.K.P. and Zak, S.H. (2001) *An Introduction to Optimization*. (2nd Edition.) Wiley, New York.

Diwekar, U. (2003) *Introduction to Applied Optimization*. Kluwer Academic Publishers, Boston / Dordrecht /London.

Edgar, T.F., Himmelblau, D.M., and Lasdon, L.S. (2001) *Optimization of Chemical Processes.* (2nd Edn.) McGraw-Hill, New York.

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

Fourer, R. (2004) *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 http://www.solver.com, and http://www.solver.com/xlslgoeng.htm.

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

Grossmann, I.E., Ed. (1996) *Global Optimization in Engineering Design.* Kluwer Academic Publishers, Boston / Dordrecht / London.

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

Hillier, F.J. and Lieberman, G.J. (2005) *Introduction to Operations Research*. (8th Edition.) McGraw-Hill, New York.

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

Horst, R. and Tuy, H. (1996) *Global Optimization – Determinsitic Approaches*. (3rd Edn.) Springer-Verlag, Berlin / Heidelberg / New York.

ILOG (2004) ILOG OPL Studio and Solver Suite. http://www.ilog.com.

Isenor, G., Pintér, J.D., and Cada, M (2003) A global optimization approach to laser design. *Optimization and Engineering* 4, 177-196.

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

Kallrath, J., Ed. (2004) *Modeling Languages in Mathematical Optimization.* Kluwer Academic Publishers, Boston / Dordrecht / London.

Kampas, F.J. and Pintér, J.D. (2004a) Generalized circle packings: model formulations and numerical results. *Proceedings of the International Mathematica Symposium* (Banff, AB, Canada, August 2004).

Kampas, F.J. and Pintér, J.D. (2004b) Configuration analysis and design by using optimization tools in *Mathematica*. *The Mathematica Journal* (to appear).

Kampas, F.J. and Pintér, J.D. (2005) *Advanced Optimization: Scientific, Engineering, and Economic Applications with Mathematica Examples*. Elsevier, Amsterdam (to appear).

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

Lafe, O. (2000) *Cellular Automata Transforms*. Kluwer Academic Publishers, Boston / Dordrecht / London.

Lahey Computer Systems (2002) *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. http://www.lindo.com.

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

Maplesoft (2004a) *Maple*. (Current version: 9.5.) Maplesoft, Inc., Waterloo, ON. http://www.maplesoft.com.

Maplesoft (2004b) *Global Optimization Toolbox.* Maplesoft, Inc. Waterloo, ON. http://www.maplesoft.com.

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.

Mittelmann, H.D. and Spellucci, P. (2004) *Decision Tree for Optimization Software.* http://plato.la.asu.edu/guide.html.

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

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

Neumaier, A. (2004a) *Global Optimization.* http://www.mat.univie.ac.at/~neum/glopt.html.

Neumaier, A. (2004b) Complete search in continuous global optimization and constraint satisfaction. In: Iserles, A. (Ed.) *Acta Numerica 2004.* Cambridge University Press, Cambridge.

Papalambros, P.Y. and Wilde, D.J. (2000) *Principles of Optimal Design*. Cambridge University Press, Cambridge.

Paragon Decision Technology (2004) AIMMS (Current version 3.5). Paragon Decision Technology BV, Haarlem, The Netherlands. See http://www.aimms.com.

Pardalos, P.M., Shalloway, D. and Xue, G. (1996) *Global Minimization of Nonconvex Energy Functions: Molecular Conformation and Protein Folding*. DIMACS Series, Vol. 23, American Mathematical Society, Providence, RI.

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

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

Pintér, J.D. (1996b) Continuous global optimization software: A brief review. *Optima* 52, 1-8. (Web version is available at: http://plato.la.asu.edu/gom.html.)

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, Boston / Dordrecht / London.

Pintér, J.D. (2000) 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, Boston / Dordrecht / London.

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 also http://www.wolfram.com/products/applications/mathoptimizer/.

Pintér, J.D. (2002b) Global optimization: software, test problems, and applications. In: Pardalos and Romeijn, Eds. *Handbook of Global Optimization. Volume 2*, pp. 515-569.

Pintér, J.D. (2003a) Globally optimized calibration of nonlinear models: techniques, software, and applications. *Optimization Methods and Software* 18 (2003) (3) 335-355.

Pintér, J.D. (2003b) GAMS /LGO nonlinear solver suite: key features, usage, and numerical performance. Submitted for publication. Downloadable at http://www.gams.com/solvers/lgo.

Pintér, J.D. (2004) *LGO – A Model Development System for Continuous Global Optimization. User's Guide*. (Current revision.) Pintér Consulting Services, Inc., Halifax, NS. For a summary, see http://www.pinterconsulting.com.

Pintér, J.D. (2005a) *Applied Nonlinear Optimization in Modeling Environments.* CRC Press, Baton Rouge, FL. (To appear.)

Pintér, J.D., Ed. (2005b) *Global Optimization – Selected Case Studies.* (To appear.) Kluwer Academic Publishers, Boston / Dordrecht / London.

Pintér, J.D., Holmström, K., Göran, A.O., and Edvall, M.M. (2004) User's Guide for TOMLAB /LGO. TOMLAB Optimization AB, Västerås, Sweden. http://www.tomlab.biz.

Pintér, J.D. and Kampas, F.J. (2003) *MathOptimizer Professional – An Advanced Modeling and Optimization System for Mathematica Users with an External Solver Link. User Guide.* Pintér Consulting Services, Inc., Halifax, NS, Canada. For a summary, see also http://www.wolfram.com/products/ applications/mathoptpro/.

Pintér, J.D. and Kampas, F.J. (2005) Model development and optimization with *Mathematica.* In: Golden, B., Raghavan, S. and Wasil, E., Eds. *Proceedings of the 2005 INFORMS Computing Society Conference* (Annapolis, MD, January 2005), pp. 285-302. Kluwer Academic Publishers, Boston / Dordrecht / London.

Pintér, J.D. and Purcell, C.J. (2003) Optimization of finite element models with *MathOptimizer* and *ModelMaker*. Lecture presented at the *2003 Mathematica Developer Conference*, Champaign, IL. (Extended abstract is available upon request, and also from http://www.library.com.)

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., *Handbook of Global Optimization. Volume 1*, pp. 751-828.

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

Schittkowski, K. (2002) *Numerical Data Fitting in Dynamical Systems*. Kluwer Academic Publishers, Boston / Dordrecht / London.

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.

Tawarmalani, M. and Sahinidis, N.V. (2002) *Convexification and Global Optimization in Continuous and Mixed-integer Nonlinear Programming.* Kluwer Academic Publishers, Boston / Dordrecht / London.

Tervo, J., Kolmonen, P., Lyyra-Laitinen, T., Pintér, J.D., and Lahtinen, T. (2003) An optimization-based approach to the multiple static delivery technique in radiation therapy. *Annals of Operations Research* 119, 205-227.

TOMLAB Optimization (2004) TOMLAB. TOMLAB Optimization AB, Västerås, Sweden. http://www.tomlab.biz.

Trefethen, L.N. (2002) The Hundred-Dollar, Hundred-Digit Challenge Problems. *SIAM News*, Issue 1, p. 3.

The MathWorks (2004) *MATLAB.* (Current version: 6.5) The MathWorks, Inc., Natick, MA. 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.

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

Wolfram, S. (2003) *The Mathematica Book*. (Fourth Edn.) Wolfram Media, Champaign, IL, and Cambridge University Press, Cambridge.

Wolfram Research (2004) *Mathematica* (Current version: 5.1). Wolfram Research, Inc., Champaign, IL. http://www.wolfram.com.

Zabinsky, Z.B. (2003) *Stochastic Adaptive Search for Global Optimization.* Kluwer Academic Publishers, Boston / Dordrecht / London.