Global Optimization Toolbox for Maple: An Introduction with Illustrative Applications

János D. Pintér
PCS Inc., 129 Glenforest Drive, Halifax, NS, Canada B3M 1J2
jdpinter@hfx.eastlink.ca *www.pinterconsulting.com*

David Linder
Maplesoft, 615 Kumpf Drive, Waterloo, ON, Canada N2V 1K8
dlinder@maplesoft.com

Paulina Chin
Maplesoft, 615 Kumpf Drive, Waterloo, ON, Canada N2V 1K8
pchin@maplesoft.com

Abstract

This article presents a concise review of the scientific–technical computing system Maple[1] and its application potentials in Operations Research, systems modeling and optimization. The primary emphasis is placed on nonlinear optimization models that may involve complicated functions, and/or may have multiple – global and local – optima. We introduce the Global Optimization Toolbox to solve such models, and illustrate its use by numerical examples.

Keywords: advanced systems modeling and optimization; Maple; Global Optimization Toolbox; application examples.

1 Model Development and Optimization: An O.R. Framework

In today's competitive global economy, government organizations and private businesses all aim for resource-efficient operations that deliver high quality products and services. This demands effective and timely decisions in an increasingly complex and dynamic environment. Operations Research (O.R.) provides a scientifically established methodology to assist analysts and decision-makers in finding "good" (feasible) or "best" (optimal) solutions. O.R. has undergone remarkable progress since its beginnings: for example, the 50th Anniversary Issue of the journal *Operations Research* (2002) reviews an impressive range of real-world applications.

To motivate the present discussion, we shall review the key steps of the O.R. modeling framework. A formal procedure aimed at making optimized decisions consists of the following main steps.

1. Conceptual description of the decision problem at a suitable level of abstraction, retaining only the essential attributes, while omitting secondary details and circumstances.
2. Development of a quantitative model (or system of models) that captures the key elements of the decision problem, in terms of decision variables and the relevant functional relationships among these, expressed by constraints and objectives.
3. Development and/or adaptation of an algorithmic solution procedure, in order to explore the set of feasible solutions and to select the best decision (or a list of good alternative decisions).

[1] Maple is a trademark of Maplesoft, a division of Waterloo Maple Inc.

4. Numerical solution of the model and its verification; interpretation and summary of results.
5. Posterior analysis and implementation of the decision(s) selected.

Steps 1 and 2 arguably require an element of modeling art, in addition to technical knowledge. Step 3 requires also a certain "taste" to select a suitable, computationally tractable approach to solving the problem. Steps 1, 4 and 5 will greatly benefit from interaction with the actual decision-makers or users of the decision support system developed. The importance of these aspects, in order to obtain meaningful solutions to real-life decision problems – and thereby also to increase the visibility that the field of O.R. rightly deserves – cannot be over-emphasized.

The problems tackled by O.R. and related disciplines are often so complex that the correct model and solution procedure may not be clear at the beginning. Furthermore, drawbacks to a modeling or solution approach that looked promising in Steps 2 and 3 may only become apparent at Step 4, either because the solution failed to capture the key attributes of the decision problem identified in Step 1, or because the solution approach was not computationally tractable. Therefore, modelers and decision makers often must carry out the steps listed above in an iterative fashion. That is, they repeatedly modify and refine the model formulation and solution procedure until the model captures the essence of the problem, it is computationally tractable, and its solution is deployable in a real setting.

The "standard" steps of model development and optimization (reviewed above) put into perspective the usage of high-level, integrated software tools that can effectively assist in performing the related tasks.

2 Maple in Systems Modeling and Optimization

The integrated computing system Maple (Maple, 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 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). Note finally that Maple is portable across all major hardware platforms and operating systems (Windows, Macintosh, Linux, and Unix versions). All these capabilities accelerate and expand the scope of model development and solution.

To emphasize the key features pertaining to advanced systems modeling and optimization, a concise listing of these 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" (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 graphical user interfaces (by using "Maplet" applications)
- supports advanced technical documentation, desktop publishing, and presentation
- provides links to external software products.

Further details on Maple features relevant to the current discussion will be presented in Sections 5, 6 and 7. For additional information on topics that are outside of the scope of the present discussion, we refer to Maplesoft's Web site (www.maplesoft.com) which provides a wealth of information. See also the Maple Application Center (www.mapleapps.com) which offers over 2,500 freely downloadable resources including ready-made materials from simple classroom demos to curricula for entire courses, as well as a range of tutorials and examples. As of today (2005), nearly two hundred books and thousands of articles have been written on Maple and its applications. Maplesoft's Web site also includes links to software reviews written about Maple. The current product version is Maple 9.5 released in April 2004; the Global Optimization Toolbox (Maple, 2004b) was released in June 2004.

3 Global Optimization

Within the broad context of systems modeling and optimization, we see a particularly strong case for using an integrated computing system – such as Maple – to analyze nonlinear systems that are often defined by individually formulated functional relations. Some of the model functions may require embedded computational procedures defined by special functions, integrals, systems of differential equations, deterministic or stochastic simulation, and so on. Examples of nonlinear models – possibly defined by "expensive" functions – are abundant in various scientific and engineering applications: consult e.g. Aris (1999), Beltrami (1993), Edgar, Himmelblau and Lasdon (2001), Gershenfeld (1999), Grossmann (1996), Mandelbrot (1983), Murray (1983), Papalambros and Wilde (2000), Parlar (2000), Pardalos and Resende (2002), Pearson (1986), Pintér (1996, 2001, 2005), Press, Teukolsky, Vetterling, and Flannery (1992), Schroeder (1991), Schittkowski (2002), Tawarmalani and Sahinidis (2002), Zabinsky (2003), and Zwillinger (1989) for related discussions.

To illustrate this point with a relatively simple (but not quite trivial) numerical example, consider the following optimization problem:

(1) *minimize* $f(x) := 0.1(x-3)^2 + \sin^2(x^2 + 10x - \Gamma(x))$

subject to $1 \le x \le 5.$

In (1) $\Gamma(x)$ denotes the gamma function $\Gamma(x) = \int_0^\infty \exp(-t)t^{x-1}dt$.

Figure 1 shows the picture of the objective function $f(x)$. (All numerical examples presented in this paper have been formulated, solved and visualized using the features of Maple and the Global Optimization Toolbox.)

This example illustrates two types of difficulty. First of all, there is an embedded integral in the optimization model that has to be evaluated for algorithmically selected arguments *x*. Second, the model may have a multi-extremal structure, and this fact makes necessary the application of a global scope optimization strategy. Note that – unlike in this relatively simple one-dimensional example –

one often can not directly explore the model functions, thereby verifying model convexity or non-convexity. With the exception of (important, but limited in scope) special cases, it is not easy to verify convex structure in complex nonlinear models. Hence, if we are interested in the best possible solution, then it is prudent to approach such models as global optimization (GO) problems.

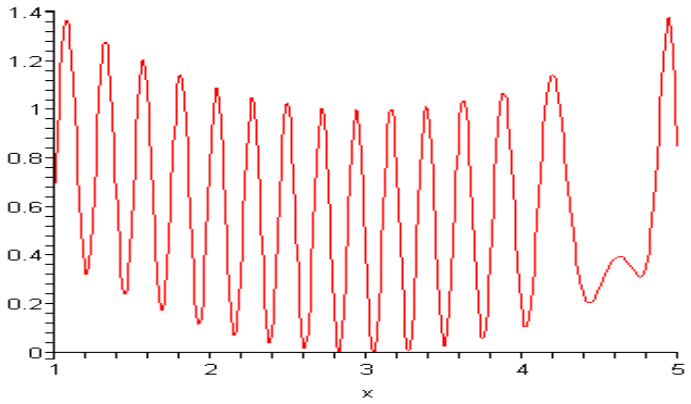

Figure 1 The objective function $f(x)$ in model (1).

In order to formalize the general GO paradigm discussed here, we shall use the following notation:

- x decision vector, an element of the real Euclidean n-space $\boldsymbol{R}^n$
- $f(x)$ objective function, $f: \boldsymbol{R}^n \rightarrow \boldsymbol{R}$
- D non-empty set of admissible decisions; D is defined by
- x_l, x_u explicit, finite bounds of x (an embedding "box" in $\boldsymbol{R}^n$)
- $g(x)$ m-vector of constraint functions, $g: \boldsymbol{R}^n \rightarrow \boldsymbol{R}^m$.

Applying the notation given above, the GO model can be concisely stated as

$$(2) \qquad \min f(x) \quad x \in D := \{x: \; x_l \le x \le x_u \;\; g(x) \le 0\}.$$

Note that all inequalities in (2) are interpreted component-wise. Observe that the model statement (2) covers a broad class of formally more general models. For example, instead of ≤ relations, arbitrary combinations of inequality and equality relations could be used in the functions g; one could state explicit bounds on the constraint function values; and one could even use a combination of continuous and discrete (finitely bounded) decision variables.

Under fairly general analytical conditions, model (2) has a global solution(s): for example, if D is non-empty, and functions f, g (component-wise) are continuous, then the model has a global solution (set). However, if we use "traditional" local scope search methods, then – depending on the starting point of the search – we will find only locally optimal solutions of varying quality, recall Figure 1. In order to find (i.e., to properly approximate) the "true" solution, a genuine global scope search effort is needed.

In higher dimensions, complexity could increase rapidly as shown by the next example. Assume that we wish to find a numerical solution to the following system of equations in a given variable range:

(3) $$e^{x-y} + \sin(2x) - \cos(y) = 0$$
$$4x - e^{-y} + 5\sin(6x - y) + 3\cos(3xy) = 0$$
$$-2 \le x \le 2, \quad -1 \le y \le 3.$$

Figure 2 shows the least squares (l_2-norm based) error function induced by the problem statement (3). (Note that – as discussed e.g. in Pintér (1996) – the general system of equations $g(x)=0$, $x \in \boldsymbol{R}^n$, $g: \boldsymbol{R}^n \to \boldsymbol{R}^m$ can be reformulated as a global optimization model with the objective *min* $\|g\|$; here $\|.\|$ can be chosen as an arbitrary norm function in $\boldsymbol{R}^m$.)

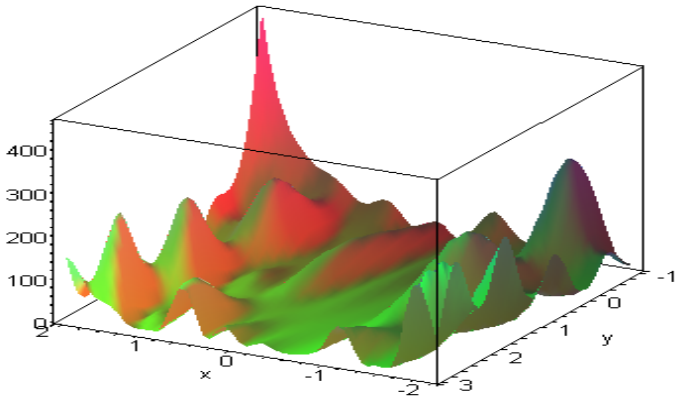

Figure 2 Least squares error function associated with problem (3).

Nonlinear models are literally ubiquitous in advanced engineering design, biotechnology, computational physics and chemistry, data analysis, econometrics, environmental management, financial modeling, medical and pharmaceutical research, process control, risk management, and other areas. Their solution often requires a global scope search approach such as the software implementation discussed in this paper.

4 LGO Solver Engine for Nonlinear Optimization

The core of the Global Optimization Toolbox is the embedded implementation of the LGO solver suite. (LGO abbreviates Lipschitz Global Optimizer, originally named after one of its key solver components.) The theoretical foundations of LGO, some key implementation details, test examples, and detailed case studies are presented in Pintér (1996). The proprietary LGO software has been discussed in subsequent works, and it has been used in commercial applications, in research, and in educational environments for more than a decade.

LGO offers a fully integrated implementation of global and local scope nonlinear solvers. The current solver suite includes the following components that can be optionally selected by the user:

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

It is emphasized here that LGO's global search methods do not require derivative or higher order information; only the local search phase uses gradient approximations based on central differences. We plan on extending the features of this phase by adding an option for automatic differentiation: however, this may not always be appropriate (namely, in cases when analytical derivatives in fact could not be produced).

A few words regarding the theoretical convergence properties of the listed algorithm components are in order. First of all, the global search methods in LGO are indeed globally convergent – i.e., they are directly aimed at finding the global solution (set) – either deterministically (BB), or with probability 1 (GARS and MS). To guarantee theoretical global convergence, BB requires Lipschitz-continuous model structure; the stochastic search methods require only continuous model functions. Note also that – again, theoretically – the local search method GRG requires continuously differentiable model structure.

The actual code implementation is based on a numerical approximation of the underlying theory (as it is always the case). In principle, global search should be used *ad infinitum*, to guarantee exhaustive search and global convergence. Instead of this, LGO applies a limited global search effort that works with an aggregated merit (exact penalty) function. This approach assumes at least "reasonable" model scaling: that is, all model function values are supposed to vary in similar ranges (e.g. between -1000 and 1000). Furthermore, the "switch point" from global to local search and a number of other parameter settings represent heuristic elements in LGO (again, similarly to all other numerical optimization methods). To mention another practically relevant point, the "best possible" Lipschitz-information – for each model function – is typically unknown, and poor estimates may lead to inefficiency, or to missing the global optimum.

In spite of the caveats mentioned above, the LGO solver suite typically will provide good quality, globally optimized solutions when applying a reasonable computational effort (that is controllable by the user). This is true, even if the model functions are merely computable, assuming at least "overall" continuity, with possible exceptions. In solving smooth and "non-pathological" models, the LGO solver run will produce at least a high-quality feasible local solution when completed, assuming that the model has such solution(s) indeed. More details regarding numerical performance can be found in Pintér (1996, 2001, 2002, 2003). For example, the computational study (Pintér, 2003) – that has been conducted with assistance from the GAMS Corporation, using third-party GAMS models and a fully automated evaluation methodology – shows the relative advantages of LGO over state-of-art local solvers when solving GO models. The models in that study are defined by up to a few hundred variables and constraints.

The largest models solved by various LGO implementations so far had a few thousand variables and (general) constraints, in addition to variable bounds. The corresponding runtimes, on today's personal computers, are (or would be) in the range of minutes or hours. Of course, this should be interpreted only as a qualitative statement: for example, the model function evaluations themselves may require significant computational effort. The general theoretical complexity of GO is non-polynomial, and some GO model instances can be (numerically) far more or less difficult than others. Acceptable precision and computational effort, as well as solver mode and option settings also play a role in performance evaluation: consult the discussion in Khompatraporn, Pintér and Zabinsky (2003), with further topical references therein.

The optional choice of global search methods often helps in solving difficult models, since BB, GARS, and MS apply different algorithmic strategies. The parameterization of these solvers (e.g., intensified global search) can also help, although the internally set default search effort typically leads to a close numerical approximation of the global solution. The latter statement has been verified by

solving (also) some difficult GO challenges, in which the known or putative globally optimal solution is publicly available, and it has been reproduced by LGO: see e.g. Pintér (2001, 2002, 2003, 2005).

5 Global Optimization Toolbox for Maple

Maple's mathematics engine consists of two main components. The kernel, which is written in the C language, is responsible for all basic "low-level" operations. The library, which contains most of the mathematical algorithms in Maple, is written in the Maple language itself. External libraries and applications – often implemented in the C language – can be linked into Maple through its external calling mechanism. On top of the mathematics engine is the user interface, including a sophisticated GUI that offers interactive visualization functionality, context-sensitive menus, and the Maplet system which allows customization of the Maple environment.

The Maple (Global Optimization Toolbox) implementation of LGO offers a powerful combination of Maple's sophisticated user interface, numerical and visualization capabilities with the robustness and efficiency of the LGO solver suite. The Toolbox is seamlessly integrated with Maple 9.5 and – after installation – it appears within Maple as the GlobalOptimization package. A package is generally a collection of related routines and data implemented using a Maple module. Maple provides packages for various mathematical topics, such as e.g. linear algebra, geometry, and group theory. Maple 9.5 also includes the Optimization package: the latter is based on the NAG Optimization Library, and it serves for local optimization of continuously differentiable functions.

The following command allows the commands in the GlobalOptimization package to be invoked directly:

```
> with(GlobalOptimization);
```

The Toolbox automatically sets default parameter values for its operations, partly based on the model to solve: these are suitable in most cases, but the user can assign (i.e., override) them. Namely, one can select the following options and parameter values:

- minimization or maximization model
- search method (BB+LS, GARS+LS, MS+LS, or stand-alone LS)
- initial solution vector setting (used by the stand-alone LS operational mode, if available)
- constraint penalty multiplier: this is used by BB, GARS and MS, in an aggregated merit function; the LS search phase handles all model functions individually
- maximal number of merit function evaluations in the selected global search mode
- maximal number of merit function evaluations in the global search mode, without merit function value improvement
- acceptable target value for the merit function, to trigger a "switch over" to LS
- feasibility tolerance used in LS mode
- Karush-Kuhn-Tucker local optimality tolerance in LS mode
- Solution (computation) time limit.

Further information regarding the Toolbox can be found on the product web page (Maple, 2004b), and in the related Maple help system entries.

The GlobalSolve command is used below to solve two illustrative global optimization problems. The function calls will be immediately followed by the corresponding results.

In the first of these examples we wish to find an optimized argument value of the *BesselJ* function. *BesselJ*(*v*,*x*) is a function that satisfies Bessel's differential equation

(4) $x^2 y'' + x y' + (x^2 - v^2) y = 0.$

Here x is the function argument, and v is the order or index parameter of the function. The evaluation of *BesselJ* requires the solution (function) of the differential equation (4), for the given value of v, and then the calculation of the corresponding function value for argument x: for example, *BesselJ*(0,2)~0.2238907791. Consequently, the illustrative optimization model shown below is aimed at finding the minimal value of the "black box" function *BesselJ* in the interval $x \in [0,100]$, when $v=0$.

```
> GlobalSolve(BesselJ(0, x), x=0..100);
[-.402759395702549983, [x = 3.83170597154937420]].
```

The next figure graphically verifies the solution found above:

```
> plot(BesselJ(0, x), x=0..100);
```

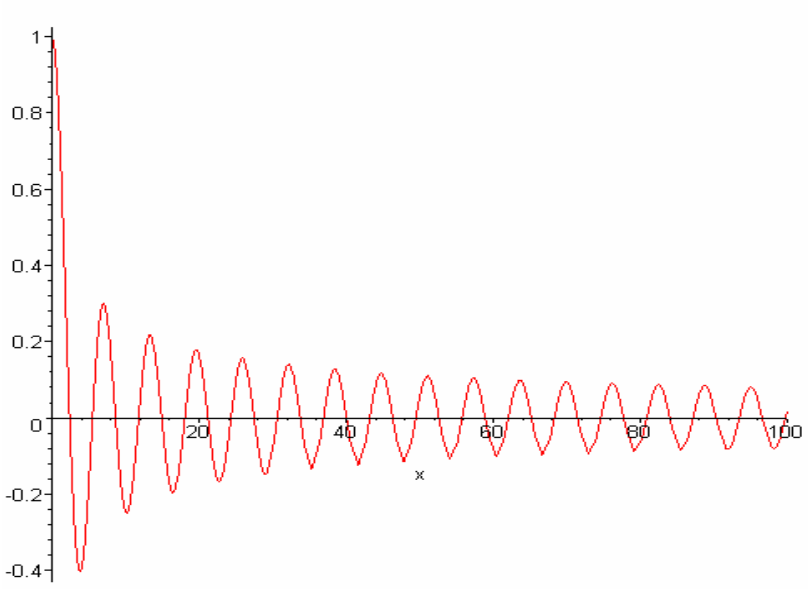

Figure 3 Function *BesselJ*(0,*x*) in the interval $x \in [0,100]$.

In the second illustrative example, we are given a set of 10 points: we first fit a spline interpolating function to these points, and then directly maximize the resulting function:

```
> points := [[0,0],[1,2],[2,-3],[3,0],[4,2],[5,-1],[6,0],[7,5],
[8,-4],[9,10],[10,5]];

> GlobalSolve(CurveFitting[Spline](points, v), v=0..10, maximize);
[11.0556303479370399, [v = 9.24463829668650839]]
```

Figure 4 shows the spline curve constructed (and it can serve to verify the solution).

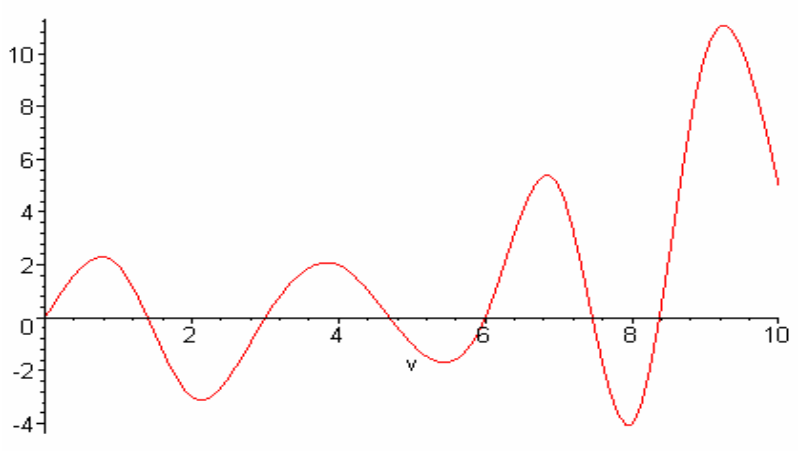

Figure 4 Spline interpolation curve, based on a set of given points.

Both of the examples presented above are relatively simple, since they are only box-constrained and involve a single optimization variable. (Examples of multivariate constrained problems will be shown later.) However, the examples highlight some key advantages of Maple and the Global Optimization Toolbox. Namely, it is easy to use the full power of the Maple library, including special function evaluations (in the first example) and specialized functionality (computation of a spline interpolating function, given a list of points, in the second case). The model visualization capabilities are also of obvious use in the present context.

The second example also illustrates one of the options available to control the behaviour of the Toolbox: the option 'maximize' indicates that maximization, rather than (the default) minimization, should be attempted.

An overall advantage of the Maple platform is that the user can provide optimization model functions and constraints as high-level algebraic or functional expressions, thus keeping the focus on the mathematical modeling aspects, rather than on "lower level" programming details. Note in this context that the GlobalSolve command allows the model input to be provided as Maple procedures, in those cases where the objective or constraint functions cannot be easily written as a simple algebraic expression. A more complex form of input, called "Matrix Form", which uses procedures involving vector and matrix parameters, is available for those users whose primary goal is efficiency rather than ease-of-use. An interactive Maplet application, which requires no knowledge of the command-line syntax, is also available and will be described later on.

Though the input may be provided in symbolic form, the underlying computation is numeric. Maple provides both hardware floating-point computation and software (arbitrary precision) floating-point computation. Generally, all computations in the Global Optimization Toolbox are performed with hardware floats whenever possible, and the solver engine itself uses hardware floats exclusively. However, it is possible to specify an arbitrary number of digits to be used for computation outside the solver engine: this includes also the evaluation of the model functions, thereby leading to more accurate results.

By default, only the final objective value (estimated optimum) and the corresponding (estimated global) solution vector is returned. However, it is possible to get output that provides more information about default settings and computed results, by setting an information level option.

6 Model Formulation and Function Evaluation Options

The LGO solver engine used by Maple Global Optimization Toolbox exists as a dynamic library of compiled routines rather than as a Maple language implementation. The solver engine is accessed by the Toolbox through Maple's external-calling mechanism. This communication is not directly visible to the Maple user, who interacts with the Global Optimization Toolbox either through the GlobalSolve command (as illustrated above) or through the interactive Maplet graphical interface (this option will be discussed later on).

The information passed to the solver engine consists of the values for LGO's optional settings along with handles to Maple procedures for the optimization model functions. During each evaluation of the objective and constraint functions the solver engine will make a callback to Maple. These callback actions consist of passing the current evaluation point's numerical coordinates to Maple through the appropriate handle, and receiving a floating-point result in return. The Toolbox's user-level routine GlobalSolve automatically provides the solver engine with Maple procedures to be used as callback handles, whether the user provides the model objective and constraint functions as expressions, procedures, or matrices.

There are two principal callback types that an external function can make to Maple, in order to obtain the necessary floating-point evaluations. One type of callback performs evaluation using Maple's double-precision evalhf function, while the other makes use of Maple's arbitrary precision evalf function. In evalhf mode evaluation is performed at double-precision using the machine's available hardware floating-point. In contrast, evaluation done in evalf mode consists of entirely software floating-point computation, which may be done at arbitrary precision. The primary benefit of hardware floating-point evaluation to the Global Optimization Toolbox is speed of execution. The benefits of arbitrary precision evaluation include avoidance of round-off error, and complete access to all of Maple's numeric and symbolic functionality. Note however, that evalf computations are slower than working in evalhf mode.

The Global Optimization Toolbox will automatically manage the provision of callback modes on the user's behalf. The behaviour is modified according to the value of the environment flag Digits, which is used in Maple to control the precision of its radix-10 floating-point computations. The default behaviour within the Global Optimization Toolbox is to attempt hardware floating-point evaluation first and to fall back to software floating-point computation, if the objective or constraints are found to not evaluate in Maple's evalhf mode. If the value of Digits is more than the cut-off value for evalhf mode calculation (14 or 15 on most systems) then only software floating-point computation is attempted.

The simple basic restrictions on the Maple procedures passed to the external solver engine – namely, that they accept numeric data and return a floating-point scalar value – allow for a wide variety of "black-box" type problems to be solved, including problems which perform intermediate symbolic analysis. For example, one can also use Maple's evalM environment to make function calls to the MATLAB software product – if present – to evaluate arbitrary expressions or functions defined in that product's language or source file format. Such calls can also be directly included in Maple procedures that are passed to GlobalSolve, as components of an optimization model formulation.

Maple's external-calling mechanism can evaluate precompiled external functions (e.g. those written in C), and these can also be used in functions passed to the Global Optimization Toolbox. For example, suppose that we compiled the C source of function fun1 shown below and linked this into a dynamic link (Windows) or shared (Unix, Linux, Macintosh) library, using a suitable C compiler and linker. In the example below, we will use the name libfun1.so for this dynamic shared library object. Then a Maple procedure can be defined which calls the external function, and this procedure can be used e.g. as the objective function passed to the Toolbox's user-level entry-point. The next example illustrates this approach. (Maple comment lines start with #.)

```
> # Print the simple C source file that defines the function fun1.
> # This function will be precompiled, to generate libfun1.so.
> system("cat fun1.c"):

#include <math.h>
double fun1( double x )
{
  return x * sin(x);
}

> # Link the corresponding external function into Maple.
> func1 := define_external( fun1, a::float[8],
>             RETURN::float[8], LIB="libfun1.so" ):

> # func1 may now serve as a model component function.
> # First here is a simple evaluation test:
> func1(-3);
0.423360024179601612

> # Solve an optimization problem defined by using func1 as its
> # objective function:
> GlobalSolve(func1, -20..20);
[-17.3076086078585014, [17.3363779125964896]]
```

In this way, the LGO solver engine can be passed handles to compiled versions of both objective and constraint functions. Note that this approach requires the most code manipulation (and external compilation), but it leads to the fastest code when compared to both evalhf and evalf. Therefore this approach is most appropriate for problems whose objective or constraints are CPU-intensive to evaluate numerically, assuming that they can be expressed e.g. in the C language. Combined with the ability of Maple's CodeGeneration package to automatically translate Maple procedures and expressions to C source code (as well as to several other languages), the Global Optimization Toolbox becomes part of a powerful hybrid symbolic-numeric system for problem prototyping, analysis, and rapid numerical solution.

It is also possible to handle expressions of some procedures which may not always evaluate to a numeric value in Maple. There are several situations in which this can occur. The request to Maple for evaluation may return a complex numeric value for some points in the domain, or may return a non-numeric value. In some situations an unevaluated Maple procedure call is returned.

In the following example we will use Maple’s capability to return a floating-point infinite value, whenever the underlying expression has not evaluated to a numeric result. The Maple symbol Float(infinity) will be used to indicate a floating-point value that is too large to be otherwise represented. (Note that: it does not represent the mathematical concept of infinity; rather, it is used as an “emergency exit” in the example.)

Assume that we wish to minimize a given parameterized function (defined by the procedure prob that depends on the parameter a) that includes the evaluation of an integral. The procedure output value – for a given argument a – is stored in the local variable sol, including the floating-point infinite return option for unevaluated cases.

```
> prob := proc(a)
  local sol;
  sol := -(2 + sin(10.0*a))*evalf(Int(x^a*sin(a/(2 - x)),
     x=0..2, epsilon=1.0e-4));
  if not type(sol,numeric) then
     return Float(infinity);
  else
     return sol;
  end if;
  end proc:
```

Now, the procedure defined above can be directly incorporated in an optimization model (e.g. as the objective function), and then solved by the Global Optimization Toolbox:

```
> GlobalSolve(prob, 0..5, evaluationlimit=50, timelimit=2000);
[-3.03370503706232997, [0.786652356047620738]]
```

Notice the time limit set in seconds: this gives an indication of the embedded numerically intensive evaluation procedure.

7 Further Examples

In the examples below, we shall illustrate some of the Toolbox usage options; full details are provided in the on-line Maple help system.

First, we recall and solve the model stated in (1) using the Global Optimization Toolbox:

```
> GlobalSolve(0.1*(x-3)^2+sin(x^2+10*x-GAMMA(x))^2, x=1..5,
method=singlestart);
[0.244301980521535647e-3, [x = 3.04941453301099052]]
```

Notice that, by contrast, the local solver (the Minimize command in the Optimization package that comes with Maple 9.5) typically returns one of the sub-optimal solutions displayed in Figure 1. Interestingly, in this example this is true, even if the "hand-picked" initial point happens to be very close to the global solution:

```
> with(Optimization):
> Minimize(0.1*(x-3)^2+sin(x^2+10*x-GAMMA(x))^2, x=1..5,
initialpoint={x=3});
[0.155114582272816368e-1, [x = 2.60625271450358564]]
```

Of course, this is not a criticism of the quality of the local solver *per se*. However, the example shows the inherent limitations of local scope search, unless one knows a priori and with sufficient precision where to initiate the search. Clearly, this would not always be a simple task: recall also the other examples presented.

Note passing by that the local solver option of the Global Optimization Toolbox actually finds the global solution from the same initial point that was supplied to the function Minimize:

```
> GlobalSolve(0.1*(x-3)^2+sin(x^2+10*x-GAMMA(x))^2, x=1..5,
initialpoint={x=3}, method=reducedgradient);
[0.244301980521534346e-3, [x = 3.04941453303258170]]
```

In fact, the built-in local search method of the Toolbox is of competitive quality for handling dense nonlinear (local) optimization models, without a postulated special structure: this point is also demonstrated by detailed results in the computational study (Pintér, 2003).

Consider next the system of equations stated in (3). The corresponding Maple code and the result are shown below:

```
> eq1:= exp(x-y)+sin(2*x)-cos(y):
eq2:= 4*x-exp(-y)+5*sin(6*x-y)+3*cos(3*x*y):
GlobalSolve(0,{eq1=0, eq2=0}, x=-2..2, y=-1..3);
[0., [x = -.436305827080927910, y = 2.35254045775939958]]
```

Notice in this model formulation the absence of the objective function, substituted by the constant function $f(x)=0$. Multiple solutions could be found sequentially in a straightforward fashion (assuming their existence), e.g., by adding linear constraints to exclude a small neighborhood of the solution(s) found. This approach could be easily automated within a Maple procedure that incorporates the GlobalSolve function.

For illustration, the next statement shows another solution found by adding an option in GlobalSolve. Note that the added option is different from the default setting, since otherwise the solver package would exactly reproduce the previous result.

```
> GlobalSolve(0,{eq1=0, eq2=0}, x=-2..2, y=-1..3, evaluationlimit=10000);
[0., [x = 0.0688826582476782524, y = 1.01941327313764352]]
```

The next statement will drive GlobalSolve to search for the minimum l_2-norm solution of the given system of equations:

```
> GlobalSolve(x^2+y^2,{eq1=0, eq2=0}, x=-2..2, y=-1..3);
[0.0548956677682248818,
[x = -0.0842289884206296191, y = -0.218634730264572392]]
```

Consider now another optimization model that has a multimodal objective and two non-convex constraints as shown below:

$$
\begin{array}{lll}
(5) & \textit{minimize} & (\sin(2x^2 + xy^2))^2 + (\sin(4y + x^2 - 12xy))^2 \\
 & \textit{subject to} & \log(1 + x^4) + 8\sin(x^2 - y) \le 0.01 \\
 & & x - x^2 + \sin(x) + y^2 - 5y \le -1.2 \\
 & & -2 \le x \le 3 \\
 & & -4 \le y \le 2.
\end{array}
$$

The next two Maple statements and the corresponding answers show again the possible difference between local and global search results. Notice the constraint definitions that, for added emphasis, we placed on the second line; the variable bounds are on the third line of the function calls:

```
> Minimize(sin(2*x^2+x*y^2)^2+sin(4*y+x^2-12*x*y)^2,
{log(1+x^4)+8*sin(x^2-y)<=0.01, x-x^2+sin(x)+y^2-5*y<=-1.2},
x=-2..3, y=-4..2);
[.496830263822040430, [x = .479207342500193644, y = 2.]]

> GlobalSolve(sin(2*x^2+x*y^2)^2+sin(4*y+x^2-12*x*y)^2,
{log(1+x^4)+8*sin(x^2-y)<=0.01, x-x^2+sin(x)+y^2-5*y<=-1.2},
x=-2..3, y=-4..2);
[0.149088414199065262e-18,
[x = 0.101572134982289432e-9, y = 1.57079632734697694]]
```

As one can see, the global solver has found a (numerical) global solution, while the local search approach has led to a suboptimal one.

Finally, we shall also demonstrate the interactive solver mode using this example model:

```
> Interactive(sin(2*x^2+x*y^2)^2+sin(4*y+x^2-12*x*y)^2,
{log(1+x^4)+8*sin(x^2-y)<=0.01, x-x^2+sin(x)+y^2-5*y<=-1.2},
x=-2..3, y=-4..2);
[1.490884141990652621 0e-19,
[x = 1.015721349822894321 0e-10, y = 1.57079632734697694]]
```

The command shown above (that now uses the Interactive function instead of GlobalSolve) opens the Optimization Assistant dialog (Figure 5). This Assistant is part of the Global Optimization Toolbox graphical user interface. The modeler can now directly formulate an optimization model through the dialog: this feature supports fast model prototyping and subsequent model analysis options.

After solving the model, a model visualization option can also be activated. The Optimization Plotter dialog (Figure 6) supports visualization in model variable subspaces, defined by variable pairs (for $n \geq 2$, in the one-dimensional case a standard function plot is displayed). The variable subspaces can be interactively selected by the modeler, while keeping all other variables at the computed optimum. The plots themselves can also be flexibly adjusted: in addition to the objective function, constraints can be shown as planes orthogonal to or as lines on the surfaces of the (projected) objective function; these figures can also be rotated, and so on. The graphics features outlined can effectively help users within the context of the model development and verification process.

Further examples, including other tests and illustrative applications with a practical flavor, are described in the Toolbox documentation at the product page, and also in a forthcoming book (Pintér, 2005).

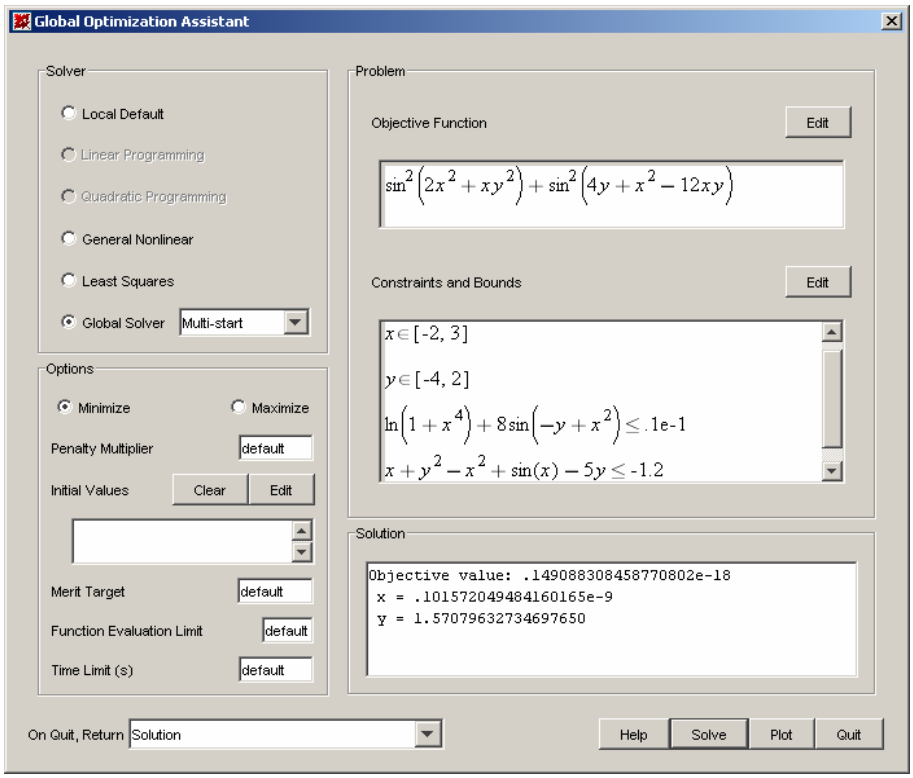

Figure 5 Optimization Assistant dialog.

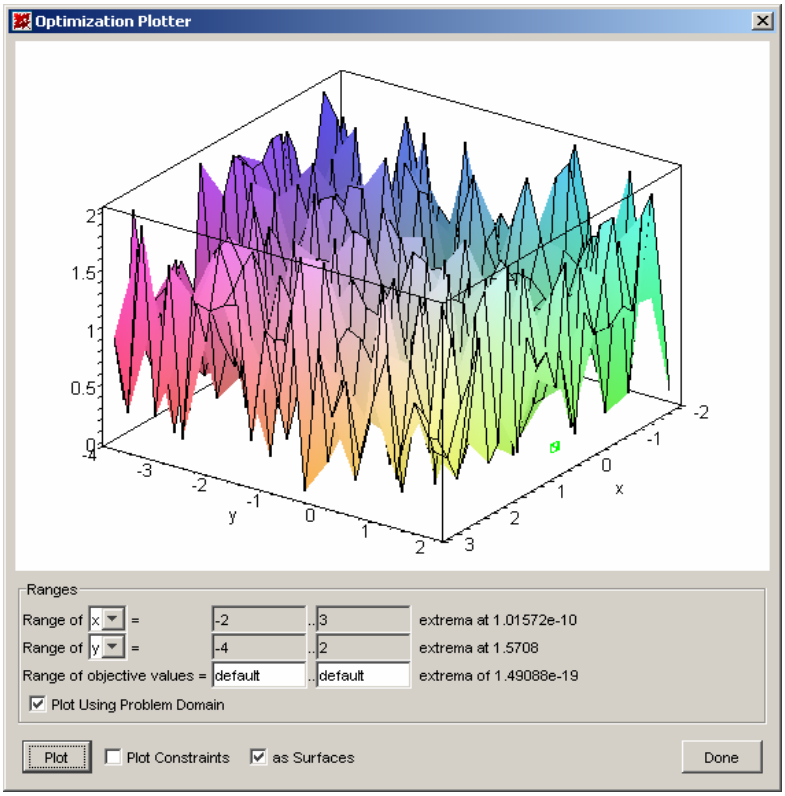

Figure 6 Optimization Plotter dialog.

8 Application Perspectives

As mentioned earlier, the LGO solver system has been used for over a decade in a significant variety of commercial, research and educational applications. Nonlinear/global optimization models with up to a few thousand variables and constraints have been solved (so far). In addition to the case studies available from the Toolbox product page, we mention here only two other recent studies. Isenor, Pintér, and Cada (2003) describe a laser design application; Tervo, Kolmonen, Lyyra-Laitinen, Pintér, and Lahtinen (2003) present a detailed model and its solution in relation to cancer therapy planning. Consult also the review (Pintér, 2002) and the numerical study (Pintér, 2003): in the latter article, over one hundred practically motivated models are solved, in a reproducible manner, from the publicly available GAMS Model Library.

We see strong application potentials for the Global Optimization Toolbox in the advanced analysis of nonlinear systems. Several broad (and obviously overlapping) classes of application areas, with a potential need for global optimization, are:

- models with a provably non-convex structure
- optimization of complex "black box" systems
- optimal control of dynamic systems
- decision-making under uncertainty.

Models belonging to these broad categories are ubiquitous in research and commercial applications of mathematics, sciences, engineering, econometrics and finance. Indeed, the rapidly growing user base of the Toolbox includes researchers from world-class engineering, biotechnology, healthcare, and financial organizations, as well as from top universities and research centers.

In addition to the ones already illustrated or mentioned, a few recent application examples are listed below:

- acoustics equipment (loudspeaker, transducer) design
- model development and optimization in chemical and process engineering
- data analysis, classification, and visualization
- economic and financial forecasting model development
- environmental risk assessment and management
- financial optimization
- industrial (product) design
- model fitting to data (calibration)
- optimization in numerical mathematics (parametric integrals and differential equations)
- optimal operation of completely "closed" (including confidential) engineering systems
- potential energy models in computational physics and chemistry
- packing and other object arrangement design problems
- robot design and manipulations.

In conclusion, Maple's integrated environment and the solver capabilities of the Global Optimization Toolbox can be put to good use in a rapidly growing range of professional applications, as well as in research and education.

Acknowledgements

We wish to express our thanks to the management and development team of Maplesoft – especially to Laurent Bernardin, Jim Cooper, Tom Lee, Paul Mansfield, Darren McIntyre, Eithne Murray, Raqeeb Rasheed, Jason Schattman, and Allan Wittkopf – for their support, cooperation and contributions to the Global Optimization Toolbox project.

References

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

Beltrami, E. (1993) *Mathematical Models in the Social and Biological Sciences*. Jones and Bartlett, Boston.

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

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

Grossmann, I., ed. (1996) *Global Optimization in Engineering Design.* Kluwer, Dordrecht, 1996.

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

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

Khompatraporn, C., Pintér, J.D. and Zabinsky, Z.B. (2003) Comparative assessment of algorithms and software for global optimization. *Journal of Global Optimization.* (To appear.)

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

Maple (2004a) *Maple 9.5.* Maplesoft, a division of Waterloo Maple Inc., Waterloo, ON.
See http://www.maplesoft.com.

Maple (2004b) *Global Optimization Toolbox.* Maplesoft, a division of Waterloo Maple Inc. Waterloo, ON. See http://www.maplesoft.com/products/toolboxes/globaloptimization/index.aspx.

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

Operations Research: 50th Anniversary Issue (2002) INFORMS, Linthicum, MD.

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

Pardalos, P.M. and Resende, M.G.C., Eds. (2002) *Handbook of Applied Optimization*. Oxford University Press, Oxford.

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

Parlar, M. (2000) *Interactive Operations Research with Maple: Models and Methods*. Birkhäuser, Boston.

Pearson, C.E. (1986) *Numerical Methods in Engineering and Science.* Van Nostrand Reinhold, New York.

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

Pintér, J.D. (2001) *Computational Global Optimization in Nonlinear Systems: An Interactive Tutorial*. Lionheart Publishing, Atlanta, GA.

Pintér, J.D. (2002) Global optimization: software, test problems, and applications. Chapter 15 (pp. 515-569) in: *Handbook of Global Optimization, Volume 2* (Pardalos, P. M. and Romeijn, H. E., Eds.) Kluwer Academic Publishers, Dordrecht.

Pintér, J.D. (2003) GAMS/LGO nonlinear solver suite: key features, usage, and numerical performance. (Submitted for publication.) *GAMS Development Corporation – Solver Documentation Pages.* See http://www.gams.com/solvers/solvers.htm#LGO.

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

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (1992) *Numerical Recipes in Fortran.* (2nd Edition.) Cambridge University Press, Cambridge.

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

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

Tawarmalani, M. and Sahinidis, N.V. (2002) Convexifixation and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming. Kluwer Academic Publishers, Dordrecht.

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.

The MathWorks (2003) MATLAB (Release 13). The MathWorks, Inc., Natick, MA.

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

Zwillinger, D. (1989) *Handbook of Differential Equations*. (3rd Edition.) Academic Press, New York.