A review of the Global Optimization Toolbox for Maple

Didier Henrion

April 11, 2006

Global optimization is aimed at finding the best solution of a constrained nonlinear optimization problem by performing a complete search over the set of feasible solutions. In contrast with local optimization, a complete search exhaustively checks the entire feasible region. For a comprehensive up-to-date archive of online information on global optimization, see [6].

As surveyed in [7], there are numerous mathematical and engineering problem classes for which a complete search is required. An example is the 300 year old Kepler problem of finding the densest packing of equal spheres in 3-dimensional Euclidean space, for which a computer-assisted proof is proposed by T. C. Hales [4]. The proof consists in reducing the problem to several thousands of linear programs and using interval calculations to ensure rigorous handling of rounding errors when establishing correctness of inequalities. Many other famous difficult optimization problems, such as the traveling salesman problem and the protein folding problem, are global optimization problems.

The Global Optimization Toolbox (GOT for short), first released in June 2004 with Maple 9.5, is part of the Maple Professional Toolbox series of add-on products that must be purchased separately. The GOT is jointly developed by Maplesoft and Pintér Consulting Services (PCS). PCS is led by János D. Pintér, who has been developing optimization software for over 15 years, in collaboration with various academic and industrial partners. The core of the GOT is the LGO (Lipschitz Global Optimizer), an integrated suite of global and local solvers based on the research summarized in the monograph [9] which was awarded the 2000 INFORMS Computing Society Prize for Research Excellence. The LGO is at the core of many other commercial global optimization packages. Solver engines using the LGO are available for C and Fortran compiler platforms, Excel, AIMMS, GAMS, Matlab (via Tomlab), MPL, and Mathematica.

With the GOT, the LGO first performs a global search using either (1) a branch and probability bound algorithm, or (2) an adaptive stochastic search, or (3) a multi-start stochastic search (default). Constraints are incorporated into a penalty term. After the global search, the solver attempts to refine the solution with a local search based on a

generalized reduced-gradient algorithm (a standard technique of local nonlinear optimization) using the solution from the global search as its initial point. If required, the user can disable the global search phase so that only local search is performed. Interestingly, recent versions of Maple also include an alternative internal optimization package based on the NAG optimization library, which serves exclusively for local optimization.

Global convergence of the LGO is guaranteed theoretically under the assumption that the objective function and constraints are continuous (required for the stochastic methods) or Lipschitz continuous (required for the branch and bound method) [9]. Optimization problems not satisfying these conditions can still be handled, but the quality of the results obtained may vary. For example, the solver can handle nondifferentiable functions because it estimates derivatives (Lipschitz constants) numerically using function values only, that is with a genuine black-box knowledge of the functions to optimize or evaluate. The local phase uses gradient approximations, but not the penalty function. Derivatives or higher order information cannot be provided by the user.

Besides the well-written on-line help available under Maple, the main features of the GOT are nicely described in [12] thanks to several simple but non-trivial numerical examples. It is immediately apparent that the core LGO solver is perfectly integrated within the Maple environment. In particular, the user can provide optimization model functions and constraints as high-level Maple algebraic, functional, or matrix expressions. See [12, Section 5] for two nice illustrative examples of this interaction (minimization of a black-box Bessel function, and spline curve fitting). Many other interesting examples are described in the colorful and didactical on-line example Maple worksheet "Applications of the Global Optimization Toolbox."

Even though the input to the GOT can be symbolic, the underlying computation is numeric. Maple provides both hardware floating-point arithmetic and software floating-point arithmetic at arbitrary precision, but internally the GOT uses only hardware arithmetic, which is generally much more efficient. However, the user can still exploit Maple's capability of evaluating the model functions using higher precision arithmetic.

The LGO solver was recently benchmarked by Arnold Neumaier's team within the COCONUT project funded by the European Union [8]. The recent survey [7] reports a dozen of high-quality solvers for constrained global optimization, including the LGO in its GAMS implementation. With the methodology and criteria chosen in [8], the LGO is not among the best codes in terms of performance or reliability, but it must be recalled that it is one of the few codes running using function values only. J. Pintér informed us that a much better performance could have been achieved if the ratio between the global and local search efforts were selected according to the particular problem class. This automatic parameter tuning is effective in both GAMS and GOT implementations, but not for the benchmarks reported in [8]. See also [11] for a performance evaluation of LGO on a test model set available in GAMS.

It is claimed in [12] that in its current implementations, the LGO can handle problems with up to a few thousand variables and constraints. Corresponding running times on a

standard desk computer are in the range of minutes or hours. Recall that the theoretical computational complexity of global optimization algorithms is non-polynomial. Moreover, at fixed dimension, some problem instances can be much more numerically difficult than others. It must also be emphasized that the LGO finds (when possible) only one global optimizer. In the presence of multiple global optimizers, it does not enumerate all of them automatically.

Bound constraints on optimization variables must be explicitly specified when using the GOT. As recalled in [8], some other global optimization packages do not require explicit bounds, but they use bound estimates. So this may be viewed as a limitation of the GOT. On the other hand, citing [7, Section 23]: "unbounded variables are perhaps the dominant reason for failure of current complete global optimization codes on problems with few variables." One can then argue whether it is reasonable or not to assume that a designer has an a priori good knowledge of bounds on all the decision variables in a given optimization problem.

Readers of this Magazine may be interested in learning that control engineering is not among the many existing and potential application areas of the LGO solver listed on the PCS website. Generally speaking, one may wonder why global optimization software is not that popular in the control community. A few attempts have been reported and published in the control or global optimization literature, but no genuine computer-aided control system design tools have ever emerged. This is in sharp contrast with numerical linear algebra, which is at the core of many control packages (e.g. Lyapunov or Riccati equation solvers for Matlab or Scilab), or, to a lesser extent, convex semidefinite programming over linear matrix inequalities.

In summary, the Global Optimization Toolbox for Maple provides a widely applicable, fully integrated development environment to support (also) control engineering applications.

Numerical examples

Let us now illustrate the main features of the GOT on standard examples of nonconvex optimization problems arising in control. With the help of these basic examples, we illustrate some typical difficulties that can be faced by a control engineer attempting to solve simple control problems using the GOT, or any other global optimization software.

Robust stability margin - First example

Our first example is a classical robust stability margin computation problem for a linear system affected by interval uncertainty. The example was first described in [3]. It is labeled Example 7.3.1 in the optimization benchmark database compiled in the book [2]

and available for download at `titan.princeton.edu/TestProblems`. The optimization problem reads

$$\begin{array}{ll} \min & k \\ \text{s.t.} & 10q_2^2q_3^3 + 10q_2^3q_3^2 + 200q_2^2q_3^2 + 100q_2^3q_3 + 100q_2q_3^3 + q_1q_2q_3^2 + q_1q_2^2q_3 + 1000q_2q_3^2 \\ & +8q_1q_3^2 + 1000q_2^2q_3 + 8q_1q_2^2 + 6q_1q_2q_3 - q_1^2 + 60q_1q_3 + 60q_1q_2 - 200q_1 \leq 0 \\ & 800 - 800k \leq q_1 \leq 800 + 800k \\ & 4 - 2k \leq q_2 \leq 4 + 2k \\ & 6 - 3k \leq q_3 \leq 6 + 3k \end{array}$$

and the decision variables are the four scalar parameters k, q_1, q_2, q_3. This problem can be solved with the GOT as follows:

```
>with(GlobalOptimization):
>GlobalSolve(k, {10*q2^2*q3^3 + 10*q2^3*q3^2 + 200*q2^2*q3^2
  + 100*q2^3*q3 + 100*q2*q3^3 + q1*q2*q3^2 + q1*q2^2*q3
  + 1000*q2*q3^2 + 8*q1*q3^2 + 1000*q2^2*q3 + 8*q1*q2^2
  + 6*q1*q2*q3 - q1^2 + 60*q1*q3 + 60*q1*q2 - 200*q1 <= 0,
  -800*k - q1 <= -800, -800*k + q1 <= 800,
  -2*k - q2 <= -4, -2*k + q2 <= 4,
  -3*k - q3 <= -6, -3*k + q3 <= 6},
  k = -10000..10000, q1 = -10000..10000, q2 = -10000..10000,
  q3 = -10000..10000);
```

The first input argument is the objective function to be minimized, the second input argument gathers the problem constraints, and the remaining input arguments are explicit bounds on the decision variables. Note that it is not possible to omit these bounds. Here we just chose sufficiently large intervals.

The GOT immediatly returns the solution $k = 0.34174$, $q_1 = 1073.4$, $q_2 = 3.3165$, $q_3 = 4.9748$ (rounded here to five significant digits) which is the global optimum reported in [2]. By invoking the GOT with the above commands, we use the default multi-start algorithm.

Note that by using the `Minimize` command of the Optimization Package using the same input arguments as above, Maple returns $k = 1.0075$, $q_1 = 384.82$, $q_2 = 1.9845$, $q_3 = 2.9815$. In this case, it is a sequential quadratic programming algorithm which is used, and only a suboptimal solution is returned. This example illustrates the advantage of using the GOT over a local optimization algorithm.

Robust stability margin - Second example

Example 7.3.3 in [2], proposed first in [1], involves five parameters:

$$\begin{array}{ll} \min & k \\ \text{s.t.} & q_3 + 9.625q_1w + 16q_2w + 16w^2 + 12 - 4q_1 - q_2 - 78w = 0 \\ & 16q_1w + 44 - 19q_1 - 8q_2 - q_3 - 24w = 0 \\ & 2.25 - 0.25k \leq q_1 \leq 2.25 + 0.25k \\ & 1.5 - 0.50k \leq q_2 \leq 1.5 + 0.50k \\ & 1.5 - 1.50k \leq q_3 \leq 1.5 + 1.50k. \end{array}$$

The GOT, with bounds `-10000..10000` on all decision variables and its default multi-start algorithm, quickly returns the point $k = 0.87648$, $q_1 = 2.0309$, $q_2 = 1.9382$, $q_3 = 2.8147$, $w = 1.5196$. We can check that the point satisfies the constraints, but it is not possible to know whether it is globally optimal. Note that this is a problematic feature which is not specific to the GOT: a very few global optimization codes provide numerical certificates of global optimality.

With an additional argument `method=branchandbound`, the branch-and-bound algorithm of the GOT yields the point $k = 1.1448$, $q_1 = 1.9638$, $q_2 = 0.92761$, $q_3 = -0.21717$, $w = 0.069534$. This latter point satisfies the constraints, but it yields a larger objective function than the point returned by the multi-start algorithm, hence it is certainly not globally optimal. We may conjecture it is locally optimal, but the GOT does not provide such an information.

If we refine the search bounds to `-100..100` for all five variables, the multi-start algorithm of the GOT returns $k = 0.81753$, $q_1 = 2.4544$, $q_2 = 1.9088$, $q_3 = 2.7263$, $w = 1.3510$, which yields a smaller objective function than the point obtained previously with larger search bounds. It turns out that this point is the global optimum reported in [2]. Note how problematic is here the dependence of the solver performance on the tightness of the bounds on decision variables.

Fixed order controller design

We focus on the basic issue of finding a fixed-order linear controller maximizing the closed-loop asymptotic decay rate of a linear system. Mathematically speaking, this problem is called spectral abscissa minimization. Given matrices $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{p\times n}$, the problem consists of finding matrix $K \in \mathbb{R}^{m\times p}$ minimizing the spectral abscissa $\alpha(A+BKC)$ defined as the largest real part of the eigenvalues of matrix $A+BKC$. For further reference, let

$$\inf_K \quad \alpha(A + BKC) \tag{1}$$

denote the spectral abscissa minimization problem. Note that the constraint $\alpha(A + BKC) < 0$ is equivalent to closed-loop asymptotic stabilization.

Optimization problem (1) is difficult for at least two reasons. First, $\alpha(A+BKC)$ is a nonconvex function of K. Typically, it has many local minimizers and stationary points. Second, $\alpha(A+BKC)$ is continuous, but not necessarily differentiable in K. In particular, it is typically nondifferentiable, or nonsmooth, at local minimizers.

Note also that problem (1) is unconstrained, that is, we have no a priori knowledge on the relative magnitudes of entries of a globally optimal matrix K^*. Moreover we do not know if problem (1) is bounded below, that is whether the global minimum $\alpha(A+BK^*C)$ is finite and attained. In the sequel, we will see what are the practical implications of these observations when using the GOT to solve problem (1) numerically.

Let us consider a simple but non-trivial occurrence of problem (1), namely asymptotic decay rate minimization for the standard two-mass-spring benchmark setup [13] controlled with a second-order linear controller. Normalized problem data are as follows:

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

In order to use the GOT to solve problem (1), we first have to build up a Maple procedure which computes spectral abscissa $\alpha(A+BKC)$ given matrix K. Let us call this procedure `specabs`. It invokes the `Eigenvalues` function of the `LinearAlgebra` package, extracts the real parts, and sorts them out.

From the GOT, the LGO solver is called with the main command `GlobalSolve`. In its simplest form for unconstrained optimization, the command accepts two input arguments: an objective function to be minimized and a list of lower and upper bounds for all variables. Our decision variable K is unconstrained, so we have to choose arbitrary, but sufficiently large bounds for a start. For example, we select `-10...10` as the search interval for all variables. Note that this choice may have a significant impact on the performance of the LGO solver.

First we run the built-in local search, using `method=reducedgradient` as a third input argument to the command `GlobalSolve`. After 66 function evaluations, we obtain a spectral abscissa approximately equal to zero. As far as the underlying control problem is concerned, this solution is of a little interest since the resulting controller gain K is not stabilizing. Surprisingly, we obtain the same result using the branch-and-bound global search, using `method=branchandbound` and 2136 function evaluations.

With `method=singlestart`, the adaptive random search with a single starting point,

the GOT is able to stabilize the plant but convergence is so slow that a `'timelimit'=60` option has to be specified to stop the solver after 60 seconds of computation. The best achieved spectral abscissa is -0.026300. This can be improved with `method=multistart`, the adaptive random search with multiple starting points. After 60 seconds, the spectral abscissa is pushed further to -0.42190 for

$$K = \begin{bmatrix} -1.7538 & 0.89258 & -1.4675 \\ -1.5828 & -0.78232 & -1.4468 \\ 2.1337 & 2.2768 & 1.9587 \end{bmatrix}.$$

Relaxing the maximum computation time to 5 minutes does not significantly improve the results on the same search region. Optimizing over tighter intervals around the above estimate of K allows to reduce the spectral abscissa to -0.46656 with

$$K = \begin{bmatrix} -1.7180 & 0.95091 & -1.7737 \\ -1.3804 & -1.0959 & -1.3488 \\ 2.4611 & 2.7804 & 2.2544 \end{bmatrix}.$$

The corresponding closed-loop input step response is satisfactory, with a slight overshoot of 5%, a rise time[1] of 3.4s and a settling time[2] of 9.4s.

Note however that we are still quite far from the best spectral abscissa known for this example, which is $\alpha^* = -\frac{\sqrt{15}}{5} \approx -0.77460$, achieved with, for example,

$$K^* = \begin{bmatrix} 0 & 4 & 0 \\ -\frac{7}{4} & -\frac{6\sqrt{15}}{5} & 8 \\ -\frac{236}{125} & -\frac{168\sqrt{15}}{125} & \frac{43}{5} \end{bmatrix} \approx \begin{bmatrix} 0 & 4 & 0 \\ -1.7500 & -4.6476 & 8 \\ -1.8880 & -5.5053 & 8.6000 \end{bmatrix}.$$

M. L. Overton pointed out that recent results from nonsmooth variational analysis can be used to prove local optimality of this point, which was obtained analytically by pole placement, see [5] for details.

We believe that the GOT is not able to minimize the spectral abscissa from -0.46656 down to -0.77460 precisely because it is a nonsmooth function at local optimizers. Around a local optimizer, first-order derivatives of the spectral abscissa are typically very large. Practically speaking it means that a tiny perturbation around the locally optimal matrix K^* given above will result in a significant change on the value of $\alpha(A + BKC)$. In other words, the performance achieved by controller K^* is highly non-robust. This is typical of an ill-posed optimization problem in control. Indeed, in a realistic control engineering problem, a decay rate constraint would be only one of many closed-loop performance requirements in a typical multiobjective optimization setting, together with e.g. time-domain, H_2, or H_∞ constraints.

In summary, on the one hand, the GOT is not able to find a locally optimal controller because the spectral abscissa function is too sensitive around local optima. But on the other hand, such a locally optimal controller is highly non-robust, and hardly useful in practice.

[1]The rise time is the time required for the step response to rise from 10% to 90% of its final value.

[2]The settling time is the time required for the step response to decline and stay at 5% of its final value.

Acknowledgments

This review benefited from many constructive comments by Dennis Bernstein, Vikram Kapila, Marcel Mongeau, Arnold Neumaier, and János Pintér. Thanks to David Linder of Maplesoft for the core code implementation of function `specabs`. GOT calculations for the fixed order controller design problem were carried out by János Pintér.

References

[1] J. Ackermann, D. Kaesbauer, R. Muench. Robust gamma-stability analysis in a plant parameter space. Automatica, Vol. 27, No. 1, pp. 75–85, 1991.

[2] C. A. Floudas, P.M. Pardalos et al. Handbook of Test Problems for Local and Global Optimization. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999.

[3] R. R. E. de Gaston, M. G. Safonov. Exact calculation of the multiloop stability margin IEEE Transactions on Automatic Control, Vol. 33, No. 2. pp. 156–171, 1988.

[4] T. C. Hales. A proof of the Kepler conjecture. Annals of Mathematics, Vol. 162, pp. 1065–1185, 2005.

[5] D. Henrion, M. L. Overton. Maximizing the closed loop asymptotic decay rate for the two-mass-spring control problem. `arxiv.org:math.OC/0603681`, submitted for publication, March 2006.

[6] A. Neumaier. Global Optimization. `www.mat.univie.ac.at/~neum/glopt.html`

[7] A. Neumaier. Complete search in continuous global optimization and constraint satisfaction. pp. 271-369 in: Acta Numerica 2004 (A. Iserles, Ed.), Cambridge University Press, UK, 2004.

[8] A. Neumaier, O. Shcherbina, W. Huyer, T. Vinkó. A comparison of complete global optimization solvers. Mathematical Programming B, Vol. 103, pp. 335–356, 2005.

[9] J. D. Pintér. Global Optimization in Action. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1996.

[10] J. D. Pintér. LGO. A model development and solver system for continuous global optimization. User guide. Pintér Consulting Services, Halifax, Canada, 1996-2005.

[11] J. D. Pintér. GAMS/LGO nonlinear solver suite: key features, usage and numerical performance. GAMS Development Corporation, Washington, USA, November 2003. Submitted for publication.

[12] J. D. Pintér, D. Linder, P. Chin. Global Optimization Toolbox for Maple. An introduction with illustrative applications. Optimization Online, February 2005. To appear in Optimization Methods and Software.

[13] B. Wie, D. S. Bernstein. Benchmark problems for robust control design. AIAA Journal of Guidance, Control and Dynamics, Vol. 15, No. 5, pp. 1057-1059, 1992.

Didier Henrion (henrion@laas.fr) is with LAAS-CNRS, Toulouse, France and the Faculty of Electrical Engineering, Czech Technical University in Prague, Czech Republic. His research interests include numerical algorithms for polynomial matrices, convex optimization over linear matrix inequalities (LMI), robust multivariable control and computer-aided control system design (CACSD).