

AIMMS/LGO Solver Engine

A Brief Introduction and User's Guide

János D. Pintér

Pintér Consulting Services, Inc., Halifax, NS, Canada, B3M 1J2
jdpinter@hfx.eastlink.ca <http://www.pinterconsulting.com>

Summary

AIMMS – abbreviating Advanced Integrated Multidimensional Modeling Software – is a state-of-art environment for developing, analyzing, solving, and deploying optimization models. The Lipschitz(-continuous) Global Optimizer (LGO) solver engine serves to handle nonlinear optimization models under very general conditions. LGO seamlessly integrates a suite of robust and efficient global and local solver strategies. This document presents a concise discussion of the AIMMS/LGO solver engine implementation. Following an introduction to the global optimization modeling framework and to the core LGO solver suite, AIMMS specific implementation details and technical notes are provided with examples and references.

1 Global Optimization

Nonlinearity plays a significant role in the development of both natural and man-made objects, formations, processes, and systems. Consequently, nonlinear descriptive models are relevant in many areas of the sciences, engineering, and economics.

Quantitative decision-making – more specifically, optimization – studies based on a nonlinear system description frequently lead to complex models that may possess multiple – (local and global) – optima. The objective of global optimization (GO) is to find the “absolutely best” solution of nonlinear optimization models under such circumstances.

The following concise exposition partially draws on Pintér (1996, 2002, 2005a, b, c): we refer to these works for detailed discussions.

To illustrate the relevance of GO by a mathematical example, a merely one-dimensional, box-constrained model is shown in Figure 1 (see next page). This figure displays a frequently cited nonlinear optimization test problem, due to Shubert, that is defined as

$$\min_{x \in [0, 10]} \sum_{k=1, \dots, 5} k \sin(k + (k+1)x)$$

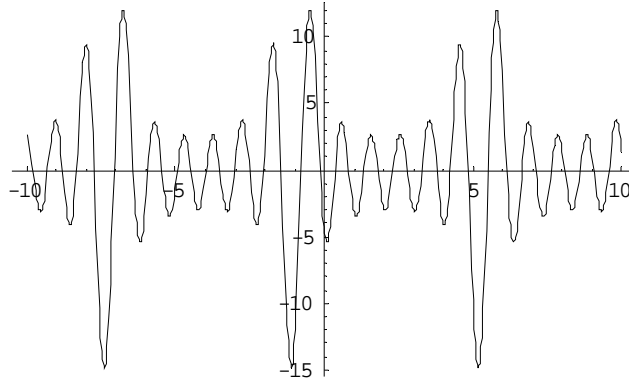


Figure 1

Shubert's one-dimensional, box-constrained GO test model.

This figure clearly illustrates the point that traditional local optimization methods will not work, unless started from a sufficiently close neighbourhood of the global solution. In various practical contexts, this means that one may easily miss the best possible solution in problems that have multiple (global or local) optima – unless making use of a genuine global search methodology.

Model complexity may be dramatic, already in low dimensions. For example, both the amplitude and frequency of the trigonometric components in Shubert's model could be increased arbitrarily, leading to more and more difficult numerical GO problems. Furthermore, increasing dimensionality in itself can lead to a rapid (theoretically exponential) increase of model complexity, in terms of the number of local/global solutions, for a given type of multi-extremal models. To illustrate this point, consider the two-dimensional, box-constrained objective function shown in Figure 2. This model is Problem 4 of the numerical challenges proposed by Trefethen (2002): it is stated as

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

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

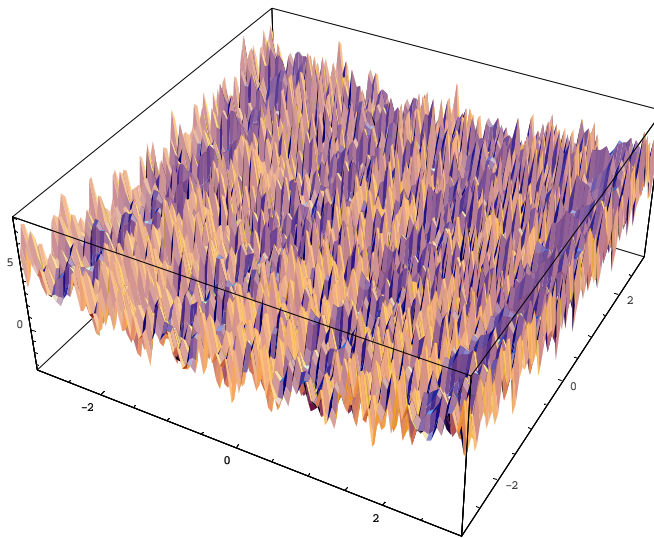


Figure 2

Trefethen's 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 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 file, dynamic link library (dll), or even as an executable program. In such and similar 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. A thorough discussion of global optimization is beyond the scope of the present discussion. Here we refer only to the volumes edited by Horst and Pardalos (1995), Pardalos and Romeijn (2002) that – in contributed chapters by leading experts – discuss the foundations of global optimization theory, with specific detailed expositions devoted to the most prominent GO model-types and solution approaches.

In this document, we shall consider the general global optimization model defined by the following components:

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

The feasible set D is defined by

- l, u explicit, finite n -vector bounds of x ;
- $g(x)$ m -vector of continuous constraint functions, $g: \mathbf{R}^n \rightarrow \mathbf{R}^m$.

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

$$(1) \quad \min f(x)$$

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

In (2) all vector inequalities are interpreted component-wise: l, x, u , are n -dimensional vectors; g is an m -dimensional vector function and the corresponding zero denotes an m -dimensional vector. (Note that the set of the additional constraints g could be empty.) Note also that formally more general optimization models – that include = and \neq constraint relations and/or explicit lower and upper bounds on the constraint function values – can be simply reduced to the model form (1)-(2).

The CGO model itself is very general: in fact, it trivially includes linear programming and convex nonlinear programming models (under corresponding additional structural specifications). Furthermore, it also includes the entire class of pure integer (combinatorial) and mixed integer programming problems. To see this, observe that all bounded integer variables can be represented by a corresponding set of binary variables; and that every binary variable $y \in \{0,1\}$ can be equivalently represented e.g. by its continuous extension $y \in [0,1]$ and the non-convex constraint $y(1-y) \leq 0$. Of course, we do not claim that the above approach is best – or even suitable – for “all” combinatorial or mixed integer optimization models. However, it certainly shows the generality of the CGO model framework (and the approach outlined even works in some cases, at least for models with a small number of binary variables).

2 LGO Solver Suite

The Lipschitz(-continuous) Global Optimizer software system integrates a suite of solver methods, to handle nonlinear programming models stated in the general form (1)-(2). In contrast to several widely used nonlinear local optimization solver engines (such as CONOPT or MINOS), its main scope of application is global optimization. However, LGO also has its own built-in local optimization capability.

An important point to emphasize is that model sparsity or other special model structure and features are not assumed, expected, or exploited by LGO. Hence, nonlinear models handled by LGO can be made up by arbitrary computable functions that – for reasons of theoretical validation of the numerical results obtained – should be (at least) continuous or Lipschitz-continuous over the given finite range $[l,u]$ of the (continuous) decision variables. These mild analytical requirements are met by many practical models. Without going into details, note that, for instance, all models defined by continuously differentiable functions have a suitable Lipschitz structure (and, of course, are also continuous).

The LGO solver system has been developed and gradually extended for more than a decade. The present discussion is primarily related to the AIMMS/LGO implementation. A more detailed documentation of the stand-alone LGO solver system is provided by (Pintér, 2005a, c).

LGO currently integrates the following global and local search algorithms:

- adaptive partition and search (branch-and-bound) based global search (this solver mode is abbreviated below as BB)
- adaptive global random search (single-start) (GARS)
- adaptive global random search (multi-start) (MS)
- constrained local search (LS), by the generalized reduced gradient method.

The theoretical foundations of the global search methods implemented in LGO are discussed in detail by (Pintér, 1996), with extensive further references. Here we provide only a concise summary of the key theoretical results that serve as the foundation of the global scope search components in LGO.

The search option BB (numerically) exploits the Lipschitz information referred to above: BB generates a sequence of sample points that theoretically converges to the (unique) global solution $x^* \in X^*$ of the model instance considered. If the model has a finite or countable number of global solutions X^* , then – again, theoretically, and under very general conditions – subsequences of points are generated that respectively converge to the points of X^* . Here we assume that a valid estimate of the Lipschitz model characteristics is used in the algorithmic search. In practice, such information often needs to be estimated based on the sampling procedure implemented. (This is the approach followed in the LGO BB implementation.)

Regarding the two other global search options, it is known that properly constructed stochastic search algorithms possess global convergence properties, with probability 1. To assure such convergence, the continuity of the model structure is essentially sufficient. Both the GARS and MS search methods are theoretically convergent: the sequence of improving global solution estimates converges to a point of X^* , with probability 1. (More precisely, each convergent subsequence of the point-sequence mentioned has this property.)

Evidently, any software implementation of theoretically convergent methods will always only approximate the theory. (Even local solvers would need, in general, an infinite number of itera-

tions.) For reasons of numerical efficiency, each of the global search options in LGO (that can be selected by the LGO user) is automatically followed by a local search strategy. Note that the switching point from global to local search can be triggered by several numerical options that are controlled by the user. This approach supports an – optionally more or less thorough – global search phase before starting local search from the best point (following BB or GARS), or from a selection of candidate points (following MS).

In all LGO global search modes the model functions are aggregated by an exact penalty (merit) function defined by

$$(3) \quad \min f(x) + \sum_{j \in E} |g_j(x)| + \sum_{j \in I} \max[g_j(x), 0].$$

Here the sets E and I denote the respective subsets of equality and inequality relations; $E \cup I = \{1, \dots, m\}$. A penalty multiplier parameter (selected by the LGO user) can be used to place more or less emphasis on the constraints. Its default value 1 corresponds to the formula (3). Note in passing that the formulation (3) tacitly assumes that the model functions are well-scaled in relation to each other. This aspect is important, and users should attempt to keep all model functions in similar ranges: this will help to improve numerical performance (program execution speed) and the results obtained.

The local search method implemented in LGO is based on the generalized reduced gradient (GRG) method. GRG is discussed in numerous textbooks, consult e.g. Edgar, Himmelblau, and Lasdon (2001). In the local search phase all model functions are considered and treated individually (as opposed to the global solvers). The LS solver option can be used also as a stand-alone option: in such applications LGO will rely on a given initial point supplied by the user, or (if absent) automatically generated by LGO.

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 goal to handle also models with merely computable, continuous functions, including “black box” systems. This may specifically include the optimization of models in which certain functions are evaluated by complex numerical procedures, or even by external software linked to the AIMMS model that is solved by LGO.

The overall solution approach followed by LGO supports the flexible usage of the component solvers. As a result, one can reasonably expect that LGO will return a close numerical estimate of the global solution – or at least a good quality, feasible local solution (if such solution, in fact, exists). It is worth emphasizing that nonlinear models may be very difficult numerically. In such cases it makes sense to try all three global solvers as well as the local solver mode, perhaps also changing the default parameter settings, in order to verify or to improve the numerical solution in challenging problems.

3 AIMMS: Key Features

The Advanced Integrated Multidimensional Modeling Software (AIMMS, by Paragon Decision Technology, 2005) is a state-of-art modeling environment for developing, analyzing, solving, and deploying optimization models. The website www.aimms.com provides a wealth of information related to AIMMS, including a free downloadable trial version of the software as well as detailed documentation, case studies and reviews. We highlight three of the downloadable documents, and a topical book chapter below.

The *AIMMS User's Guide* (Bisschop and Roelofs, 2005a) provides a general overview of how to use the AIMMS system itself. It is aimed at application builders, and explores AIMMS's capabilities to help creating a model-based application that is easy to maintain. The guide also describes the various graphical tools that the AIMMS system offers for this task.

The *AIMMS Language Reference* (Bisschop and Roelofs, 2005b) provides a complete description of the AIMMS modeling language, its underlying data structures and advanced language constructs. It is aimed at model builders, and provides the ultimate reference to the model constructs that can be used to get the most out of model formulations.

The document titled *AIMMS Optimization Modeling* (Bisschop, 2005) provides an introduction to general optimization modeling, and it also offers a suite of detailed examples. This work is aimed at users who are new to modeling as well as at those who have limited modeling experience. The book discusses both basic concepts and more advanced modeling techniques.

A concise summary of AIMMS features is described by (Bisschop and Roelofs, 2004).

In the context of the present discussion, below we will focus on the specifics of linking LGO to AIMMS. AIMMS offers an Open Solver Interface (OSI) that defines an Application Programming Interface (API) between AIMMS and its solvers. The OSI describes a complete set of functions to handle the most general communication scenarios between the AIMMS modeling system and the solvers.

AIMMS currently can handle the following model types: linear programming (LP), mixed integer linear programming (MIP), quadratic and quadratic constraint programming (QP and QCP), nonlinear programming (NLP, including both global and local optimization), mixed integer nonlinear programming (MINLP), and mixed complementarity problems (MCP). Although LGO in principle covers all of these models, it makes sense to use it primarily as a global and/or local NLP solver.

Similarly to other solvers implemented as an optional engine to use with AIMMS, LGO has user-modifiable parameters that can be used to change its operations. AIMMS is equipped with corresponding options that serve to set these LGO parameters, using the options dialog box.

4 AIMMS/LGO Solver: Input and Output Information

Input information

Model descriptors

The model characteristics listed below are directly defined through the AIMMS model.

- Model name
- Number of model variables
- Number of model constraints
- Variable names
- Objective function name
- Constraint function names
- Constraint types (equality or inequality)

- Variable lower bounds
- Variable nominal settings
- Variable upper bounds

Solver options

The solver parameters listed below are defined through the corresponding AIMMS LGO solver dialog tree.

- LGO operational mode
- Maximal number of function evaluations in LGO global search phase
- Maximal number of function evaluations in LGO global search phase, without improvement
- Penalty multiplier of model constraints in the merit (aggregated exact penalty) function
- Acceptability threshold value in global search phase (triggers switch to local search phase)
- Target merit function value in global search phase (switch to local search phase)
- Merit function value improvement tolerance (local search phase stopping criterion)
- Constraint satisfaction tolerance (local search phase stopping criterion)
- Kuhn-Tucker condition tolerance (local search phase stopping criterion)
- Built-in LGO random number generator seed value
- Report level indicator

Output information

The information components listed below are documented through the AIMMS progress report and log file, and hence are directly available to the user from AIMMS. In addition, optional summary and more detailed LGO output files can be generated. (These are simple text files readable by any text editor.)

Optimal solution summary

- Numerical solution vector (estimate) returned by LGO solver
- Constraint values at numerical solution
- Objective function value at numerical solution
- Number of model function evaluations
- Maximal constraint violation at numerical solution
- Solution time

Model status

- 1 Globally optimal solution is found (this message indicates that the obtained solution has been generated by using global and local search)
- 2 Locally optimal solution is found (the obtained solution has been generated by using only local search)
- 3 Unbounded solution (in presence of given finite bounds, this can happen only due to model errors)
- 4 Infeasible solution (the solution found does not meet the constraint satisfaction tolerance set by the user)
- 7 Intermediate (in general, possibly non-optimal) solution returned

Solver status

- 1 Normal completion
- 2 Iteration interrupt (currently not used)
- 3 Time limit (set by user) exceeded
- 4 Local solver procedure stopped or stalled
- 7 Licensed model size exceeded (model too large)

Note that the last solver status occurs only in size-limited versions; such limits will not be present in standard AIMMS/LGO shipments.

5 A Model Example Solved by AIMMS/LGO

In the following example, we use a small, but truly difficult model that we provided for the AIMMS nonlinear optimization model library. This model (denoted here by m53) is based on Trefethen's Problem 4, recall Figure 2. Let us remark that a collection of similar small size models (with more or less dramatic difficulty) is presented by Pintér, Bagirov, and Zhang (2003). The AIMMS model formulation is shown below.

MAIN MODEL Main_globopt_m53

Comment : "This is model m53 from the technical report

An Illustrated Collection of Global Optimization Test Problems

Authors: János D. Pintér † Adil Bagirov ‡ Jiapu Zhang ‡

† Pintér Consulting Services, Inc., Halifax, Nova Scotia, Canada

jdpinter@hfx.eastlink.ca www.pinterconsulting.com

‡ Centre for Informatics and Applied Optimization

School of Information Technology & Mathematical Sciences

University of Ballarat, Ballarat, Victoria, Australia

a.bagirov@ballarat.edu.au j.zhang@students.ballarat.edu.au

<http://www.ballarat.edu.au/ard/itms/CIAO/ciao.shtml>

December 2003.

Approximate numerical solution $x^* = (-0.0244030796, 0.2106124272)$

Approximate optimum value $f^* = -3.306868647$."

DECLARATION SECTION

VARIABLE:

identifier : x1

range : [-3, 3];

VARIABLE:

identifier : x2

range : [-3, 3];

VARIABLE:

identifier : f

definition : $(1/4)*(x1^2 + x2^2) + \exp(\sin(50*x1)) - \sin(10*(x1 + x2)) + \sin(60*\exp(x2)) + \sin(70*\sin(x1)) + \sin(\sin(80*x2))$;

```

MATHEMATICAL PROGRAM:
  identifier : m53
  objective  : f
  direction  : minimize
  type       : nlp ;

ENDSECTION ;

PROCEDURE
  identifier : MainInitialization

ENDPROCEDURE ;

PROCEDURE
  identifier : MainExecution
  body      :

  solve m53;

ENDPROCEDURE ;

ENDMODEL Main_globopt_m53 ;

```

The standardized model formulation syntax shown above is essentially self-explanatory. The itemized declaration of model components is followed by the initialization and solver execution steps.

The corresponding result – retrieved from the AIMMS log file, with a slightly edited formatting and notation – is displayed below.

Name	Lower Bound	Optimized Value	Upper Bound
x1	-3.0000000000	-0.024403079	3.0000000000
x2	-3.0000000000	0.210612427	3.0000000000
f	-inf	-3.306868647	inf

This result coincides with the proven global optimum of this model, to (at least) 10-digit precision; consult e.g. the corresponding detailed discussion in Bornemann, Laurie, Wagon, and Waldvogel (2004).

Figure 3 below displays the AIMMS model environment: one can see that LGO needed nearly 100,000 function evaluations to find the solution, but the overall runtime is less than 0.6 seconds (on a Pentium 4 1.6 GHz processor based personal computer that runs under Windows XP Professional). Further detailed numerical examples and test results will appear elsewhere.

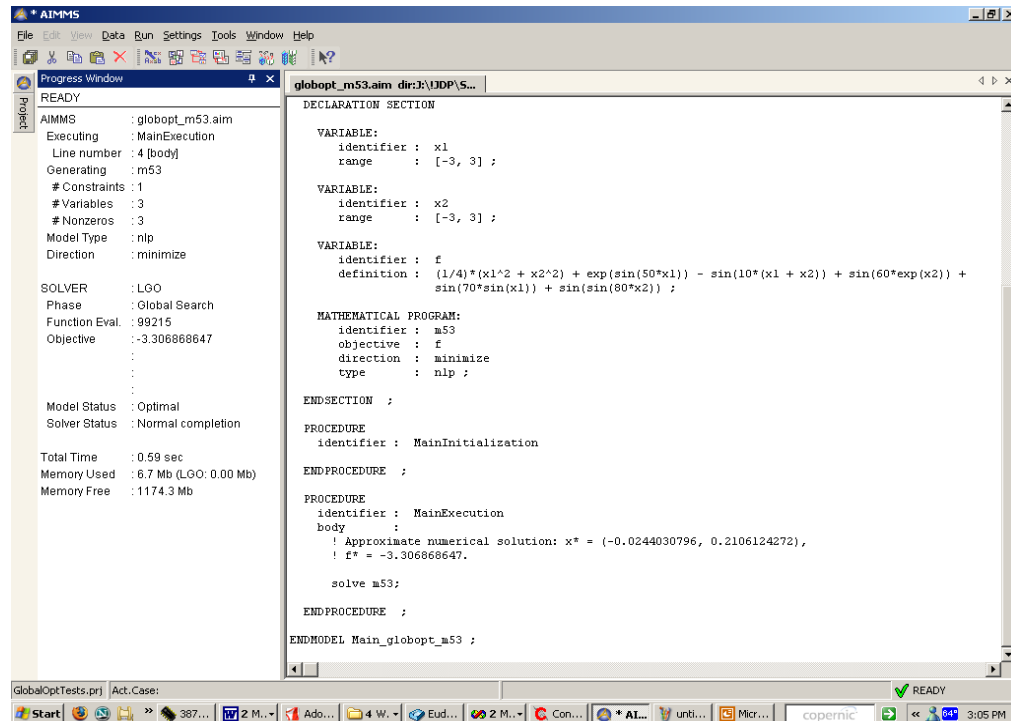


Figure 3
AIMMS Model and Progress Window (Trefethen's Problem 4).

Acknowledgements

The author expresses his thanks to Johannes Bisschop, Gertjan de Lange, and Marcel Hunting for useful discussions and contributions to the AIMMS/LGO solver link development project.

References

- Bisschop, J. (2005) *AIMMS – Optimization Modeling*. Paragon Decision Technology BV, Haarlem, The Netherlands.
- Bisschop, J. and Roelofs, M. (2004) The Modeling Language AIMMS. Chapter 6 in: Kallrath, J., Ed. *Modeling Languages in Mathematical Optimization*. Kluwer Academic Publishers, Dordrecht.
- Bisschop, J. and Roelofs, M. (2005a) *AIMMS – The User's Guide*. Paragon Decision Technology BV, Haarlem, The Netherlands.
- Bisschop, J. and Roelofs, M. (2005b) *AIMMS – Language Reference*. Paragon Decision Technology BV, Haarlem, The Netherlands.
- 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.
- Edgar, T.F., Himmelblau, D.M., and Lasdon, L.S. (2001) *Optimization of Chemical Processes*. (2nd Edn.) McGraw-Hill, New York.
- Horst, R. and Pardalos, P.M., Eds. (1995) *Handbook of Global Optimization, Volume 1*. Kluwer Academic Publishers, Dordrecht.
- Paragon Decision Technology (2005) *AIMMS* (Current version 3.6). Paragon Decision Technology B.V., Haarlem, The Netherlands. www.aimms.com.

- Pardalos, P.M. and Romeijn, H.E., Eds. (2002) *Handbook of Global Optimization, Volume 2*. Kluwer Academic Publishers, Dordrecht.
- Pintér, J.D. (1996) *Global Optimization in Action*. Kluwer Academic Publishers, Dordrecht.
- Pintér, J.D. (2002) Global Optimization: Software, Tests and Applications. In: Pardalos and Romeijn, Eds. *Handbook of Global Optimization, Volume 2*, pp. 515-569. Kluwer Academic Publishers, Dordrecht.
- Pintér, J.D., Bagirov, A., and Zhang, J. (2003) *An Illustrated Collection of Global Optimization Test Problems*. Technical Project Report. Centre for Informatics and Applied Optimization, School of Information Technology and Mathematical Sciences, University of Ballarat, Victoria, Australia.
- Pintér, J.D. (2005a) *LGO – A Model Development System for Continuous Global Optimization. User's Guide*. (Current revision.) Pintér Consulting Services, Inc., Halifax, NS. For a summary and related links, see <http://www.pinterconsulting.com>.
- Pintér, J.D. (2005b) Nonlinear Optimization in Modeling Environments: Software Implementations for Compilers, Spreadsheets, Modeling Languages, and Integrated Computing Systems. In: Jeyakumar, V. and Rubinov, A.M., Eds., *Continuous Optimization: Current Trends and Applications*; pp. 147-174. Springer Science + Business Media, New York, 2005.
- Pintér, J.D. (2005c) *Applied Nonlinear Optimization in Modeling Environments*. Chapman and Hall / CRC Press, Boca Raton, FL. (To appear.)
- Trefethen, N.L. (2002) The Hundred-Dollar, Hundred-Digit Challenge Problems. *SIAM News*, Issue 1, p. 3.