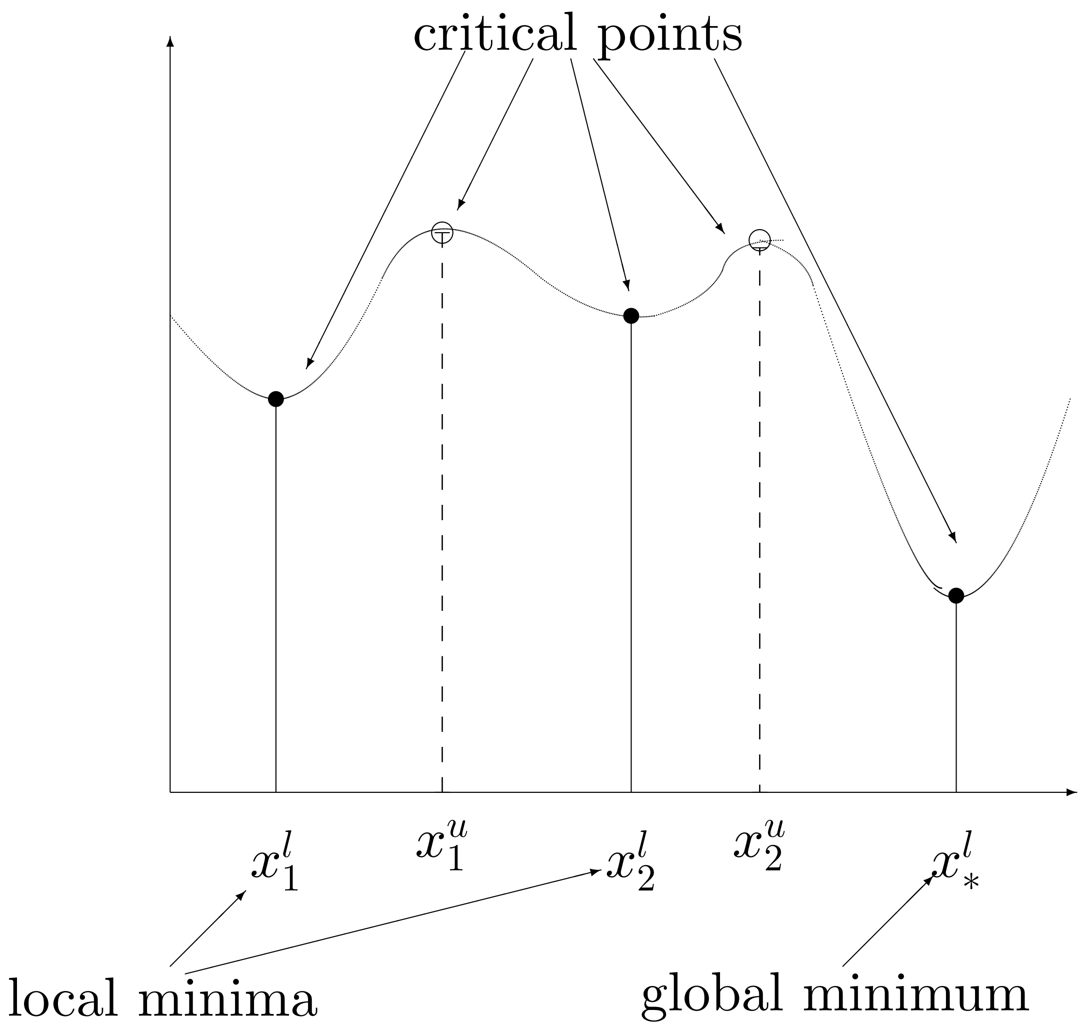
The GlobSol Software: An Overview and Examples

by R. Baker Kearfott
Department of Mathematics, University of Southwestern Louisiana
U.S.L. Box 4-1010, Lafayette, LA 70504-1010
U.S.A.
rbk@usl.edu

This talk will

- Briefly review underlying principles in global optimization.
- Present a short history of the GlobSol package.
- Present unique features of GlobSol.
- Present examples of GlobSol's use.

Local Versus Global Optimization



Local Optimization Versus Global Optimization

Local Optimization

- The model is steepest descent with univariate line searches (for monotone decrease of the objective function). (Start a ball on a hill and let it roll to the bottom of the nearest valley.)
- Algorithm developers speak of "globalization," but mean only the design of algorithm variants that increase the domain of convergence. (See J. E. Dennis and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Least Squares*, Prentice–Hall, 1983.)
- Algorithms contain many heuristics, and do not always work. However, many useful implementations exist.

Local Optimization Versus Global Optimization

Global Optimization

- is a much harder problem. Progress has accelerated with increases in computing power.
- Early milestones are L. C. W. Dixon and G. P. Szego, *Towards Global Optimization* (North–Holland, 1975), and *Towards Global Optimization 2* (North–Holland, 1977).
- Two types of algorithms: stochastic and deterministic.
- Deterministic algorithms can be either rigorous or heuristic.

Global Optimization

Deterministic Optimization

- involves some kind of systematic global search over the domain.
- The various algorithms rely on estimates of the range of the objective function over subdomains.
- Some algorithms (due to Mladineo, Schubert, Wood, etc.) rely on Lipschitz constants to obtain estimates of ranges.
- Bounds on ranges or approximate bounds on ranges are also obtained with outwardly rounded interval arithmetic or non-rigorous interval arithmetic, respectively.

Deterministic Global Optimization

Interval Methods

- Evaluation of a an objective function $\phi(X)$ at an interval vector $\mathbf{X}$ gives bounds on the actual range of ϕ over $\mathbf{X}$.
  - If directed rounding is used, the bounds rigorously contain the mathematical range.
  - The bounds, in general, are overestimates.
- If the lower bound of $\phi(\mathbf{X})$ is greater than a previously computed objective value $\phi(X)$, then $\mathbf{X}$ can be discarded.
- Interval Newton Methods, combined with directed rounding, can *prove* existence and uniqueness of critical points, as well as reduce the size of regions $\mathbf{X}$.

On the State of the Art

- Minimizing a function over a compact set in $\mathbb{R}^n$ is an NP-complete problem.
- Thus, barring monumental discoveries, any *general* algorithm will fail for some high-dimensional problems.
- There are many practical problems that can be solved in low-dimensional spaces.
- Some low-dimensional problems are difficult.
- Advances in computer speed and algorithm construction have allowed many more practical problems to be solved, including high-dimensional ones.

Interval Methods

Advantages

easier to use: Obtaining bounds with interval methods involves programming the objective function, while using Lipschitz constant-based methods may require extensive preliminary analysis.

more efficient: Despite interval overestimation of ranges, the overestimation is often less than with a fixed Lipschitz constant. *(But keep in mind the success of hybrid deterministic / stochastic algorithms.)*

more capable: With directed roundings, interval methods cannot lie. Also, interval Newton iteration results in quadratic convergence effects.

On Constraints

- Constrained problems are more difficult, since an objective function value $\phi(X)$ does not represent an upper bound on the minimum unless X is feasible.
- The Fritz–John system (Lagrange multipliers) may be used in the interval Newton methods, but other techniques must also be incorporated for practical algorithms.
- Constraints may be handled heuristically (by solving a perturbed problem) or rigorously.
- Alternate techniques are available for handling bound constraints.

GlobSol

What is GlobSol?

- A Fortran 90 package
  - well-tested.
  - self-contained.
- Solves constrained and unconstrained global optimization problems
- Separate program solves square algebraic systems of equations.
- Utility programs for interval and point evaluation, etc.
- Subroutine / module libraries for interval arithmetic, automatic differentiation, etc.

GlobSol

Special Features

- Objective function and constraints are input simply as Fortran 90 programs.
- Can use constraint propagation (substitution/iteration) on the intermediate quantities in objective function evaluation.
- Can use an overestimation-reducing "peeling" process for bound-constraints.
- Uses an effective point method to find approximate feasible points.
- Has a special augmented system mode for least squares problems.

GlobSol

Special Features, continued

- Uses epsilon-inflation and set-complementation, with carefully controlled tolerances,
  - to avoid singularity problems.
  - to facilitate verification.

GlobSol Features

(continued)

- Has extensive error-checking (user input, internal errors, etc.)
- Has on-line web page documentation.
- The algorithm is configurable.
- Has various levels of printing, for various algorithm aspects.
- Source code and libraries for components are available.
  - Automatic differentiation access.
  - Interval arithmetic access.
  - User-modifiable, with adequate study.
- Gives performance statistics, both in report form and for input to spreadsheets.

GlobSol History

- Developed as a SunSoft cooperative research and development effort during 1997–1998.
- Evolved from the `INTOPT_90` research code used by Kearfott and his students during the early 1990's.
- Uses experience from *ACM Transactions on Mathematical Software* Algorithm 681, "`INTBIS`, A Portable Interval Newton / Bisection Package (FORTRAN77)".
- Is *much* improved over `INTOPT_90`.
  - Better user interface
  - Easier installation
  - Better constraint handling
  - *Many* bug fixes

GlobSol History

(continued)

- Has elements from
  - *ACM Transactions on Mathematical Software* Algorithm 737, "A Portable FORTRAN 77 Interval Standard Function Library" and
  - *ACM Transactions on Mathematical Software* Algorithm 763, "`INTERVAL_ARITHMETIC`: A Fortran 90 Module for an Interval Data Type".
- Uses Kearfott's research dating from the early 1980's and interval research dating from the 1960's.

Use of GlobSol

An Example

The following Fortran 90 program defines the objective function

$$\text{minimize} \quad \phi(X) = -2 * x_1^2 - x_2^2$$

subject to constraints

$$\begin{aligned} x_1^2 + x_2^2 - 1 &\le 0 \\ x_1^2 - x_2 &\le 0 \\ x_1^2 - x_2^2 &= 0 \end{aligned}$$

Use of GlobSol

An Example, continued

```
PROGRAM SIMPLE_MIXED_CONSTRAINTS
  USE CODELIST_CREATION
      PARAMETER (NN=2)
      PARAMETER (NSLACK=0)
      TYPE(CDLVAR), DIMENSION(NN+NSLACK):: X
      TYPE(CDLLHS), DIMENSION(1):: PHI
      TYPE(CDLINEQ), DIMENSION(2):: G
      TYPE(CDLEQ), DIMENSION(1) :: C

      OUTPUT_FILE_NAME='MIXED.CDL'
      CALL INITIALIZE_CODELIST(X)

  PHI(1) = -2*X(1)**2 - X(2)**2
  G(1) = X(1)**2 + X(2)**2 - 1
  G(2) = X(1)**2 - X(2)
  C(1) = X(1)**2 - X(2)**2

      CALL FINISH_CODELIST
END PROGRAM SIMPLE_MIXED_CONSTRAINTS
```

GlobSol Example

(continued)

1. Running the above program produces an internal representation, or code list.
2. The optimization code interprets the code list at run time to produce floating point and interval evaluations of the objective function, gradient, and Hessian matrix.
3. A separate data file defines the initial search box, the bound constraints, and the initial guess, if any.
4. Separate data files supply algorithm options, such as which interval Newton method to use and how to precondition the linear systems.

GlobSol Example

The Data File

```
1D-5        ! General domain tolerance
 0   1      ! Bounds on the first variable
 0   1      ! Bounds on the second variable
F F         ! X(1) has no bound constraints
F F         ! X(2) has no bound constraints
```

Subsequent optional lines can give an initial guess point.

GlobSol Example

Output File – first part

```
Output from FIND_GLOBAL_MIN on  06/28/1998  at  16:28:09.
Version for the system is: June 15, 1998

Codelist file name is: MIXEDG.CDL
Box data file name is: MIXED.DT1

Initial box:
[    0.0000E+00,   0.1000E+01 ] [    0.0000E+00,   0.1000E+01 ]

BOUND_CONSTRAINT:
  F F   F F

---------------------------------------
CONFIGURATION VALUES:

EPS_DOMAIN:   0.1000E-04   MAXITR:   60000
DO_INTERVAL_NEWTON: T  QUADRATIC: T  FULL_SPACE: F
VERY_GOOD_INITIAL_GUESS:F
USE_SUBSIT:T
OUTPUT UNIT:7 PRINT_LENGTH:1
Default point optimizer was used.
```

`INTOPT_90` Example

Output File – abridged second part

```
 THERE WERE NO BOXES IN COMPLETED_LIST.

 LIST OF BOXES CONTAINING VERIFIED FEASIBLE POINTS:

 Box no.:1
 Box coordinates:
 [    0.7071E+00,   0.7071E+00 ] [    0.7071E+00,   0.7071E+00 ]
PHI:
 [   -0.1500E+01,  -0.1500E+01 ]
 Level: 3
 Box contains the following approximate root:
    0.7071E+00   0.7071E+00
 OBJECTIVE ENCLOSURE AT APPROXIMATE ROOT:
 [   -0.1500E+01,  -0.1500E+01 ]
 Unknown = T   Contains_root =T
 U0:
 [    0.3852E+00,   0.3852E+00 ]
 U:
 [    0.5777E+00,   0.5777E+00 ] [    0.0000E+00,   0.1000E+01 ]
 V:
 [    0.1926E+00,   0.1926E+00 ]
 INEQ_CERT_FEASIBLE:
  F T
 NIN_POSS_BINDING:1
```

GlobSol Example

Output File – abridged third part

```
ALGORITHM COMPLETED WITH LESS THAN THE MAXIMUM NUMBER,
      60000  OF BOXES.
Number of bisections: 3
No. dense interval residual evaluations -- gradient code list: 83
Number of orig. system inverse midpoint preconditioner rows: 3
Number of orig. system C-LP preconditioner rows: 109
Number of Gauss--Seidel steps on the dense system: 112
Number of gradient evaluations from a gradient code list: 13
Total number of dense slope matrix evaluations: 55
Number of times the interval Newton method made a coordinate
      interval smaller: 60
Number of times a box was rejected because the constraints were
      not satisfied: 1
Total time spent doing linear algebra (preconditioners
      and solution processes): 0.15018546581268311
Number of times the approximate solver was called: 2
Number Fritz-John matrix evaluations: 19
Number times SUBSIT decreased one or more
      coordinate widths: 1
Number times a box was rejected due infeasible
      inequality constraints: 2
BEST_ESTIMATE:   -0.1500E+01
Total number of boxes processed in loop: 7
Overall CPU time:    0.2277E+00
CPU time in PEEL_BOUNDARY:    0.1365E-03
CPU time in REDUCED_INTERVAL_NEWTON:    0.1458E+00
```

Future GlobSol Development

- Java / World Wide Web window user interface.
- Compiled, rather than interpreted, function and derivative evaluation.
- Improved capabilities of least squares mode.
- Improved efficiency when inequality constraints are present.
- Capabilities for embedded use within larger systems.
- Better parallel implementation, along with metering schemes, etc.
- Stronger verification algorithms.
- Better arithmetic, better printing of intervals.

GlobSol References

- For the source, installation instructions, user guide, etc.:

  `http://www.mscs.mu.edu/~globsol/`

- *Rigorous Global Search: Continuous Problems*, R. B. Kearfott, Kluwer Academic Publishers, 1996. Contains
  - An introduction to interval methods
  - An introduction to global search algorithms
  - Some specifics for INTOPT_90.
- For these transparencies:
  - `http://interval.usl.edu/preprints/SIAM_GS.ps` (Postscript)
  - `http://interval.usl.edu/preprints/SIAM_GS.dvi` (TEX DVI)