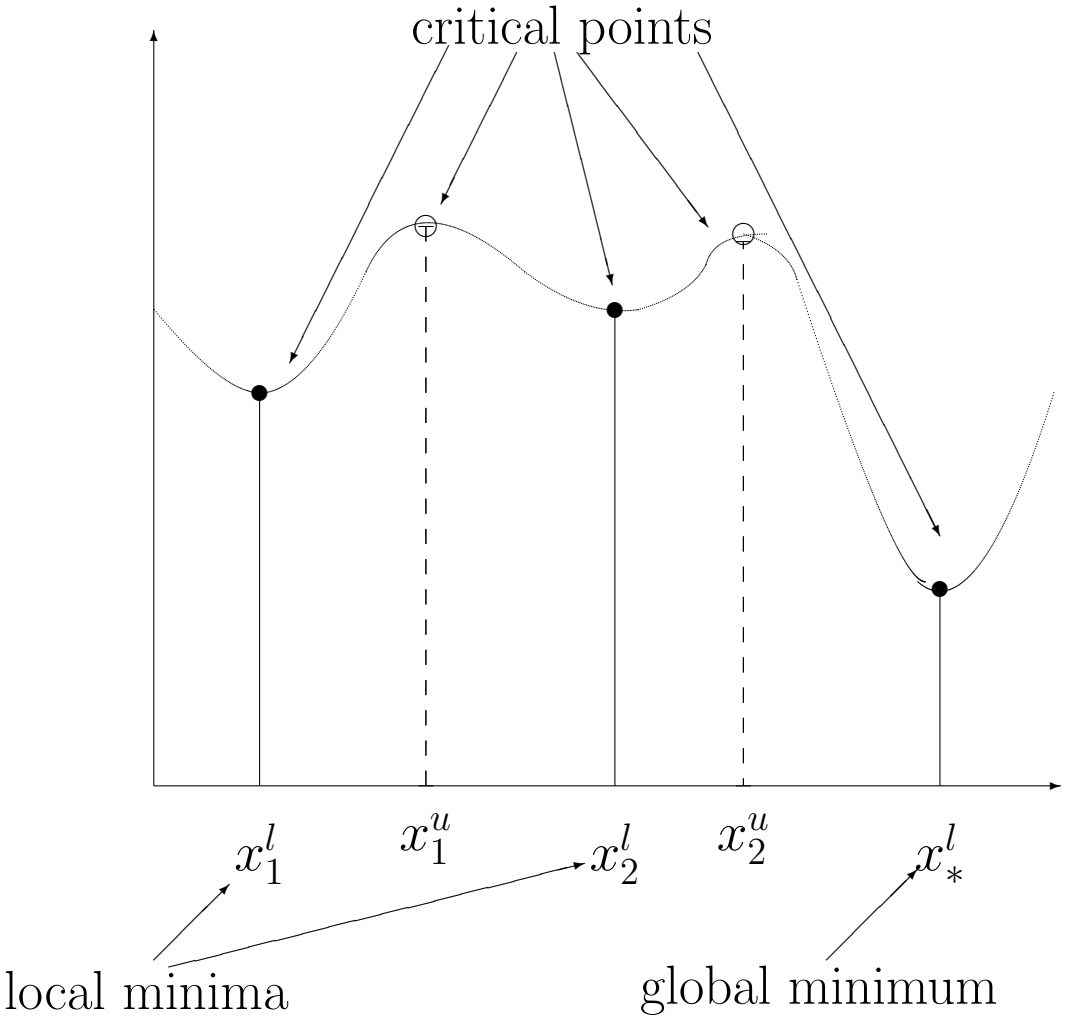
Rigorous Global Optimization and the GlobSol Package

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:

- introduce global versus local optimization,
- highlight the nature of deterministic global optimization,
- discuss capabilities of the GlobSol software package, and
- give an example of how to use GlobSol.

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

Stochastic Algorithms

Monte-Carlo search: Random points are generated in the search space. The point with lowest objective value is taken to be the global optimum.

Simulated annealing: is similar to a local optimization method, although larger objective values are accepted with a probability that decreases as the algorithm progresses.

Genetic algorithms: Attributes, such as values of a particular coordinate, correspond to particular "genes." "Chromosomes" of these genes are recombined randomly, and only the best results are kept. Random "mutations" are introduced.

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.

Global Optimization

Hybrid Stochastic / Deterministic Algorithms

Recently, several people have combined statistical methods with deterministic methods.

- Janos Pinter uses a statistical model to estimate local approximations to Lipschitz constants for a global search.
- Donald Jones constructs a cumulative statistical model of of the objective, and uses deterministic global optimization with a simpler objective to determine optimal placement of the next sample point.
- Kaj Madsen uses multiple starts of a local optimizer to simulate a rigorous global search with interval methods.

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.

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 $\mathbf{X}$.

Computational Existence / Uniqueness

A Simple Example

If $n = 1$ and

$$f(x) = x^2 - 4,$$

then the linear interval system becomes

$$2\mathbf{X}_i(\tilde{\mathbf{x}}_i - x_i) = -f(x_i).$$

The interval Newton iteration equation becomes

$$\tilde{\mathbf{x}}_i = \tilde{\mathbf{x}} = x_i - \frac{f(x_i)}{2\mathbf{X}_i} = x - \frac{f(x)}{2\mathbf{x}}.$$

If we choose initial interval and point

$$\mathbf{X}_k = \mathbf{x}_i = \mathbf{x} = [1, 2.5], \qquad x_i = x = 1.75,$$

then

$$\begin{aligned} \tilde{\mathbf{x}} = 1.75 - \frac{-.9375}{[2,5]} &= 1.75 - [.1875, .46875] \quad (1) \\ &= [1.9375, 2.21875] \subset \mathbf{x}, \quad (2) \end{aligned}$$

so we may conclude that there is a unique root of f in $\mathbf{x}$.

Preconditioned Interval Gauss–Seidel

An Example

$$\begin{aligned} f_1(x_1, x_2) &= x_1^2 - 4x_2, \\ f_2(x_1, x_2) &= x_2^2 - 2x_1 + 4x_2, \end{aligned}$$

and

$$\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2)^T = ([-0.1, 0.1], [-0.1, 0.3])^T.$$

$X^* = (0, 0)^T$, of $F(X^*) = 0$ is unique in $\mathbf{X}$.
Take $\check{X} = (0, .1)^T$, so $F(\check{X}) = (-.4, .41)^T$,
and

$$\begin{aligned} \mathbf{F}'(\mathbf{X}) &= \begin{pmatrix} 2\mathbf{x}_1 & -4 \\ -2 & 2\mathbf{x}_2 + 4 \end{pmatrix} \\ &= \begin{pmatrix} [-.2, .2] & -4 \\ -2 & [3.8, 4.6] \end{pmatrix}. \end{aligned}$$

An Example

(Continued)

Take Y to be the inverse of the midpoint matrix of $\mathbf{F}'(\mathbf{X})$:

$$Y = \{\mathrm{m}(\mathbf{F}'(\mathbf{X}))\}^{-1} = \begin{pmatrix} -.525 & -.5 \\ -.25 & 0 \end{pmatrix}.$$

Then $Y\mathbf{F}'(\mathbf{X}) = \begin{pmatrix} [0.895, 1.105] & [-.2, .2] \\ [-.05, .05] & 1 \end{pmatrix}$, $YF(\check{X}) = (.005, .1)^T$, and

$$\begin{aligned} \tilde{\mathbf{x}}_1 &= 0 - \frac{-.005 + [-.2, .2][-.2, .2]}{[.895, 1.105]} \\ &\subseteq \frac{[-.035, .045]}{[.895, 1.105]} \subseteq [-.0392, .0503] \\ &\subset \mathbf{x}_1 = [-.1, .1]. \end{aligned}$$

Similarly, $\tilde{\mathbf{x}}_2 \subset \mathbf{x}_2$, so there is a unique root of F in $\mathbf{X}$.

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.

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.
- Is publicly available free of charge.

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, equality, and inequality constraint 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.

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

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

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.
```

GlobSol 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 ]
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

Number of bisections: 3
BEST_ESTIMATE:   -0.1500E+01
Total number of boxes processed in loop: 7
Overall CPU time:    0.2277E+00
```

Recent Improvements to GlobSol

- The installation process has been greatly simplified.
- Makefiles are now available for a number of compiler / operating system combinations.
- A number of troublesome bugs have been eliminated.
- Numerical techniques for handling interval constants in the objective and constraint definitions have been incorporated.
- Constraint propagation for equality and inequality constraints has been enabled.

Simple Example of "Thick" Constants

$$\text{minimize } (x_1 - [1,2])^2 + (x_2 - [3,4])^2$$

```
PROGRAM THICK_PARAMETER_EXAMPLE
 USE CODELIST_CREATION
 IMPLICIT NONE

 INTEGER, PARAMETER:: NN=2
 TYPE(CDLVAR), DIMENSION(NN):: X
 TYPE(CDLLHS) :: PHI

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

 PHI = (X(1) - INTERVAL(1,2))**2 &
       + (X(2)-INTERVAL(3,4))**2

 CALL FINISH_CODELIST
END PROGRAM THICK_PARAMETER_EXAMPLE
```

Example with Thick Constants

(continued)

The "solution" is the set of all possible minima with constants in $[1, 2]$ and $[3, 4]$. Thus, a minimum minimum and maximum minimum are obtained. The solution is

$$x_1 \in [1, 2], \quad x_2 \in [3, 4], \quad \text{and } \phi = 0.$$

With initial box ([-10,10], [-10,10]), GlobSol gives

```
 Box no.:           1
 Box coordinates:
 [ 0.1000D+01, 0.1500D+01 ] [ 0.3000D+01, 0.4000D+01 ]
PHI:
 [ 0.0000D+00, 0.2000D+01 ]

 Box no.:           2
 Box coordinates:
 [ 0.1500D+01, 0.2000D+01 ] [ 0.3000D+01, 0.4000D+01 ]
PHI:
 [ 0.0000D+00, 0.2000D+01 ]

 BEST_ESTIMATE:    0.5000D+00
 Total number of boxes processed in loop:            4
 Overall CPU time:    0.0000D+00
```

Thick Constants

The basic idea

- The algorithm is made practical by stopping bisection when

  $$\text{w(value at center)} > \beta \text{w(interval image)}$$

  for adjustable parameter $\beta \in [0, 1]$.
- Smaller β leads to less work, but may lead to additional overestimation.
- Details are outside the scope of this survey.

A Practical Application

- This is being applied to a non-trivial maintenance interval optimization problem.
- This is ongoing work with Claudio Rocco.

GlobSol Installation and Use

Installation and use has been greatly simplified. Installation is basically as follows.

1. Complete instructions are available from the Download Globsol page at
   `http://interval.usl.edu/GlobSol/download_GlobSol.html`
2. Obtain the GlobSol zip file from this URL
3. Run an "unpack" script, obtainable from the above URL.
4. Change an installation directory in the makefile and in a file "fmake".
5. Run "make".

GlobSol is available for various machine and compiler combinations, and can be easily modified for others.

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
  - Most of the basic ideas underlying GlobSol.
  - Structure of the research code that eventually became GlobSo.
- For these transparencies:
  `http://interval.usl.edu/preprints/1999_U_of_H.ps` (Postscript)
  `http://interval.usl.edu/preprints/1999_U_of_H.dvi` (TEX DVI)