GlobSol: History, Composition, and Advice on Use

R. Baker Kearfott

University of Louisiana at Lafayette
Department of Mathematics, Box 4-1010
Lafayette, LA 70504-1010, USA
Telephone: (337) 482-5270
`rbk@louisiana.edu`

Abstract. The GlobSol software package combines various ideas from interval analysis, automatic differentiation, and constraint propagation to provide verified solutions to unconstrained and constrained global optimization problems. After briefly reviewing some of these techniques and GlobSol's development history, we provide the first overall description of GlobSol's algorithm. Giving advice on use, we point out strengths and weaknesses in GlobSol's approaches. Through examples, we show how to configure and use GlobSol.

Keywords: verified global optimization, interval analysis, GlobSol, constraint propagation, automatic differentiation

1 Introduction

Specific forms of the following related problems occur throughout scientific computing:

$$\boxed{\begin{array}{l}\text{Given a system of equations } F(x)=0\text{, find a point}\\ \check{x} \text{ such that } F(\check{x})=0,\end{array}} \tag{1}$$

and

$$\boxed{\begin{array}{l}\text{minimize } \varphi(x)\\ \text{subject to } c_i(x)=0,\ i=1,\dots,m_1,\\ \qquad\qquad g_i(x)\le 0,\ i=1,\dots,m_2,\\ \text{where } \varphi:\mathbb{R}^n\to\mathbb{R} \text{ and } c_i, g_i:\mathbb{R}^n\to\mathbb{R}.\end{array}} \tag{2}$$

Numerous algorithms, such as those in [4, 6], use heuristics to find points $\check{x}$ to approximately solve (1) or (2). Although successful and in wide use for a number of practical problems, such algorithms come with no guarantees or error bounds on the resulting approximate solutions $\check{x}$. Indeed, there are instances where approximate solutions $\check{x}$ have been published as true, but have been far from true solutions, and where subsequent rigorous investigation has revealed true solutions to the model. Furthermore, these true solutions happen to be more meaningful physically; see [5].

Thus, although more difficult with regard to both computational complexity and implementation, it is sometimes useful to validate approximate solutions $\check{x}$. For problem(2) (and with related algorithms for problem (1)), this is done with *deterministic global optimization*. In the context of deterministic global optimization, problem (2) becomes[1]

$$\boxed{\begin{array}{l} \text{Given a box } \boldsymbol{x} = ([\underline{x}_1, \overline{x}_1], \dots [\underline{x}_n, \overline{x}_n]), \text{ find narrow boxes} \\ \boldsymbol{x}^* = ([\underline{x}_1^*, \overline{x}_1^*], \dots [\underline{x}_n^*, \overline{x}_n^*]) \text{ such that any solutions of} \\ \qquad \text{minimize } \varphi(x) \\ \qquad \text{subject to } c_i(x) = 0,\ i = 1, \dots, m_1, \\ \qquad\qquad\qquad g_i(x) \le 0,\ i = 1, \dots, m_2, \\ \text{where } \varphi : \mathbb{R}^n \to \mathbb{R} \text{ and } c_i, g_i : \mathbb{R}^n \to \mathbb{R} \\ \text{are guaranteed to be within one of the } \boldsymbol{x}^* \text{ that has been found.} \end{array}} \tag{3}$$

Deterministic global optimization algorithms, instances of *branch and bound* procedures, contain a method of bounding the ranges of φ, c, and g over $\boldsymbol{x}$ and over sub-boxes of $\boldsymbol{x}$, combined with some method of subdividing a box $\boldsymbol{x}$ into smaller sub-boxes. In interval-based algorithms, interval arithmetic is used to bound the ranges. Since, with directed rounding, interval arithmetic is rigorous, interval-based branch and bound algorithms give bounds on the solution with mathematical certainty.

2 Elements of GlobSol

Specific interval branch and bound algorithms contain additional techniques to more efficiently eliminate subregions of the original region $\boldsymbol{x}$ that do not contain solutions, and to provide time-saving user interfaces. Such techniques include

1. automatic differentiation,
2. constraint propagation,
3. interval Newton methods,
4. additional, specialized techniques.

The GlobSol software is a Fortran 90-based package for global optimization that uses an interval branch and bound procedure with these additional techniques. Before describe GlobSol's overall algorithm, we provide pointers to the literature for these underlying techniques, along with synopses of how the techniques are used in GlobSol.

2.1 Automatic Differentiation

GlobSol uses automatic differentiation so the user need only input the objective φ and constraints $\boldsymbol{c}_i$ and $\boldsymbol{g}_i$, without need to worry about programming gradients and Hessian matrices or about programming separate floating point and

[1] Throughout, as in formula (3), we denote interval vectors with boldface.

interval versions, etc. Although various methods are possible, GlobSol uses *operator overloading* in a preprocessing step to first compute an internal, symbolic representation of the objective and gradients, which we call a "code list." The actual GlobSol branch and bound procedure then interprets this code list. The user inputs the objective and constraints by programming it in Fortran 90; see §5 below.

GlobSol's automatic differentiation was designed for small problems whose code lists are not difficult to evaluate. Although some removal of redundant operations is done, both the structure of GlobSol's code list and interpretive nature of its evaluation lead to less efficiency than possible by hand-coding functions and derivatives. For example, at present, the code list for the function and gradients is evaluated in its entirety, even if only a specific gradient component is required.

The operator overloading technique in GlobSol is similar to the "tape" concept in the ADOLC automatic differentiation system [8]. We give simple examples of our code list creation scheme in [16, §1.4 and §2.2.2 to §2.2.4].

An early reference on automatic differentiation is [28], while well-edited conference proceedings include [9, 1, 2]. A recent book detailing sophisticated efficiency measures is [7].

2.2 Constraint Propagation

GlobSol uses constraint propagation as an efficiency-gaining device in its overall branch and bound algorithm. GlobSol applies constraint propagation at the level of individual operations in the code list. For example, such an operation may be

$$x_p = x_q + x_r,$$

where x_p, x_q, and x_r are intermediate variables in the process of evaluating and objective, constraint, gradient, or Hessian matrix. If, say, better bounds on x_p are obtained, then a constraint propagation step would be to form $\boldsymbol{x}_q = \boldsymbol{x}_p - \boldsymbol{x}_r$ and $\boldsymbol{x}_r = \boldsymbol{x}_p - \boldsymbol{x}_q$ to try to obtain better bounds on x_q and x_r. This is in contrast to some systems, where constraint propagation is applied across larger expressions, or where the system itself is written in a language specifically designed for constraint propagation. We give examples of our use of constraint propagation in [16, Chapter 7], while our original study of the technique appears in [13].

A system more heavily emphasizing constraint propagation than GlobSol is Numerica [31]. However, the designers of Numerica have also recognized that constraint propagation alone could not provide all the needed efficiency in the branch and bound algorithm, when there is significant coupling between the equations of the gradient; Newton-type iterations are used then.

2.3 Interval Newton Methods

Interval Newton methods are perhaps the most important acceleration technique for larger problems. Abstractly, an interval Newton method is of the form

$$\tilde{\boldsymbol{x}} = \boldsymbol{N}(F; \boldsymbol{x}, \check{x}), \tag{4}$$

where F represents a system of nonlinear equations as in (1), where $\boldsymbol{x}$ is a domain box, where $\check{x}$, generally taken so $\check{x} \in \boldsymbol{x}$, is an initial guess point, and where $\tilde{\boldsymbol{x}}$ is the image of the interval Newton method.

The primary advantages of an interval Newton method are the following.

1. Under conditions that usually hold for small enough box widths, interval Newton methods reduce the diameter (i.e. reduce the maximum coordinate width) of a box $\boldsymbol{x}$ at a rate such that the diameter $\mathrm{d}(\tilde{\boldsymbol{x}})$ of the image $\tilde{\boldsymbol{x}}$ obeys $\mathrm{d}(\tilde{\boldsymbol{x}}) = \mathcal{O}\left(\mathrm{d}(\boldsymbol{x})^2\right)$. This is particularly true when F represents an approximately linear or quadratic system, and interval dependencies have been eliminated symbolically Also, obtaining $\tilde{\boldsymbol{x}}$ can be done in $\mathcal{O}\left(n^3\right)$ operations, when F represents a dense system of n equations in n unknowns.
2. Any solutions x^*, $F(x^*) = 0$ with $x^* \in \boldsymbol{x}$ must have $x^* \in \tilde{\boldsymbol{x}}$.
3. Under weak additional conditions, if $\tilde{\boldsymbol{x}} \subset \boldsymbol{x}$, then this proves that there is a unique $x^* \in \boldsymbol{x}$ such that $F(x^*) = 0$.

Numerous authors have discussed interval Newton methods. We introduce them (with more pointers to the literature) in [16, §1.5 and §6.2], and in [19]. See [27] for a careful theoretical treatment.

Within GlobSol, our interval Newton method is used with the Fritz–John (i.e. generalized Lagrange multiplier) system to reduce the sizes of boxes $\boldsymbol{x}$ produced during the subdivision process. However, GlobSol does not presently use the Fritz–John system within the context of verifying that given boxes contain unique solutions; instead, GlobSol works with the system of equality constraints $c(x) = 0$ and active inequality constraints $g(x)$ to prove existence of a feasible point. This is because, even though some coordinates can be reduced when the Fritz–John system is used with the interval Gauss–Seidel method over large boxes, we have found practical difficulties getting interval Newton methods applied to the Fritz–John system to reduce each coordinate. (In some cases, the interval Newton method may only be effective over very small boxes, or good Lagrange multiplier estimates cannot be produced.)

GlobSol combines the interval Gauss–Seidel method with special preconditioners. The interval Gauss–Seidel method generally produces narrower image boxes $\tilde{\boldsymbol{x}}$ than, say, the Krawczyk method. (See [27, p. 138].) Furthermore, our Linear programming preconditioners for the interval Gauss–Seidel method, described in [16, Chapter 3], are more effective for wide boxes $\boldsymbol{x}$; also, our timings within GlobSol have shown that obtaining these preconditioners is not overly costly within the overall branch and bound algorithm.

2.4 Additional Techniques

Use of Already-Found Optima One additional fairly common technique is use of approximate optima $\check{x}$ to eliminate portions of the original box $\boldsymbol{x}$ that do not contain optima other than those already found. We explain this in [17]. GlobSol contains a special variant of the technique that avoids excessive work when the Jacobi matrix of F is ill-conditioned or singular near solutions; see [16, step 4(c), p. 148]. This is closely related to the procedure in [12, step 4, Algorithm 2.6].

Van Iwaarden refers to such use of approximate optima as “back-boxing” [32].

Bound Constraints GlobSol has a special pre-processing step, called “peeling,” used to handle bound constraints, when one or more of the lower coordinate bounds $\underline{x}_i$ or upper coordinate bounds $\overline{x}_i$ of the original box $\boldsymbol{x}$ represents not only a limit of the search region, but also a bound constraint. See [16, §5.2.3], and see §5 below for advice on use of this technique.

Choice of Coordinate to Bisect GlobSol uses a scaled version of “maximum smear” advocated by Ratz [29]. (See also [16, (5.1), p. 175].) For constrained problems, the technique is applied to the sum of the objective function and active constraints.

3 History of GlobSol

Our original validated nonlinear systems solver, developed jointly with Manuel Novoa, was INTBIS, now an *ACM Transactions on Mathematical Software* algorithm [22]. In this small FORTRAN 77 package, polynomial systems can be solved with validation, merely by inputting the coefficients, in the format described in [26]. INTBIS follows closely [12, Algorithm 2.6], where the “root inclusion test” $\mathcal{T}_F$ of step 3 of that algorithm is represented by an interval Newton method. The interval Newton method in INTBIS differs from that of GlobSol mainly in the use of preconditioners: Jacobi matrices are preconditioned only by the inverse of the midpoint matrix, and not by the special linear programming preconditioners of [16, Ch. 3].

Subsequently, we provided a Fortran 77 interval transcendental function library INTLIB [20], with which users of INTBIS could write their own function and Jacobi matrix routines, for more or less arbitrary functions. Stadtherr et al have specially modified INTBIS to obtain impressive success in solving practical problems in chemical engineering, including an early success with larger sparse problems [30], parameter-fitting problems where previous non-verified solutions were erroneous [5], etc. Stadtherr continues to use these specially developed codes derived from INTBIS, as the overhead associated with using interpretive code lists, etc. in GlobSol appears to add substantially to execution time (perhaps a factor of 10 in some cases). The main disadvantage, besides certain outdated algorithms in the distribution version of INTBIS, is that objective and residuals must be programmed with subroutine calls, in an assembly-language-like way.

Prior to development of the automatic differentiation foundation of GlobSol, we experimented in FORTRAN 77 with our version of constraint propagation ("substitution-iteration", [13]) and with bound-constrained optimization [14]; in [14], we introduced both our aforementioned "peeling" idea and experimented with our special preconditioners as well.

During our experiments, we found ourselves struggling to hand-program various examples. This led us to initially develop the Fortran 90 interface to INTLIB (subsequently polished, with John Reid's help, and published as module `INTERVAL_ARITHMETIC` [15]). At that time, with aid from a National Science Foundation grant, we also developed the basic automatic differentiation interface and interpreters, represented in GlobSol by module `CODELIST_CREATION` in `overload/overload.f90` and various interpreters in GlobSol's `function` subdirectory, such as `FORWARD_SUBSTITUTION_POINT` in the file
`function/intermediate_variable_evaluators/forwsubpt.f90`.
This early version of our package, which we called "INTOPT90", contained a separate nonlinear systems solver and optimizer. It also contained various alternate specific subalgorithms (such as different preconditioner computations, interval Gauss–Seidel versus interval Gaussian elimination, and different normalizations for the Fritz–John equations), as well as a facility for performance statistics, including timing information. While much of the specific information about our package in [16] remains valid for the present version of GlobSol, some of that information applies only to this early version.

In 1998–1999, collaborating with George Corliss and others under G. William Walster in a Sun Microsystems Cooperative Research and Development project, we extensively polished the user interface to GlobSol, including providing scripts to unify the code list creation, code list optimization, and search phases, cleaning the configuration file, etc. At this time, we also enhanced the underlying package, providing inequality constraints, providing additional functions recognizable by the code list creator and interpreter, making certain changes to increase efficiency, etc. We also reorganized the directory structure in which GlobSol is shipped to make it clearer, and we reorganized the top-level subroutine calling structure to make it easier to call GlobSol and utilize its results in other packages. As part of this Sun Microsystems project, we did extensive formal testing, leading to the discovery and correction of significant numbers of bugs. The Sun Microsystems project also prominently included utilization of GlobSol for various practical problems [3].

Unfortunately, due to constraints on our time, not all of the possible experimental paths (e.g. not all possible formulations of the Fritz–John equations) have been uniformly tested; only those previously proven to be generally better were tested. Users of GlobSol should not find problems with default configurations; see §5 below.

We added additional capabilities to the distribution version of GlobSol subsequent to completion of the Sun Microsystems project. For example, we provided a facility for dealing with non-zero-width parameters, considered as "constants," but alterable after the code list is created; see [24] for a study of such parameters.

The distribution version of GlobSol is presently available at

`http://interval.louisiana.edu/GlobSol/download_GlobSol.html`

This includes source code and installation scripts, free of charge.

In addition to the distribution version of GlobSol, we maintain an experimental version, with capabilities that, due to either licensing restrictions or lack of thorough testing, are not presently included in the distribution version. See §6 below.

Any history of interval-based global optimization should mention Eldon Hansen, whose pioneering techniques are summarized in [10].

4 GlobSol's Overall Algorithm

Verified solution of a global optimization problem proceeds most conveniently with the "`globsol`" script, which does the following:

1. Compile and run the user-provided Fortran 90 program defining the objective and constraints, to produce the code list.
2. Optimize the code list (removing redundant operations).
3. Run the global search algorithm.
4. Clean the directory of temporary files.

The actual global search algorithm consists of

1. Initial I/O (in routine `INITIALIZE_FIND_GLOBAL_MIN` in file `f90intbi/initialize_find_global_min.f90`)
2. The "peeling" process of [16, §5.2.3] (in routine `PROCESS_INITIAL_BOX` in file `f90intbi/process_initial_box.f90`.
3. The actual global search routine (in routine `RIGOROUS_GLOBAL_SEARCH` in file `f90intbi/rigorous_global_search.f90`)
4. Final I/O (in routine `FINISH_FIND_GLOBAL_MIN` in file `f90intbi/finish_find_global_min.f90`)

These four routines are called from the driver routine `FIND_GLOBAL_MIN` in file `f90intbi/find_global_min.f90`.

The actual search routine `RIGOROUS_GLOBAL_SEARCH` contains the following algorithm.

Algorithm 1 *(GlobSol's global search algorithm)*
INPUT: A list $\mathcal{L}$ of boxes $\boldsymbol{x}$ to be searched.
OUTPUT: A list $\mathcal{U}$ of small boxes and a list $\mathcal{C}$ of boxes verified to contain feasible points, such that any global mimimizer must lie in a box in $\mathcal{U}$ or $\mathcal{C}$.

```
DO WHILE ($\mathcal{L}$ is non-empty)
  1. Remove a box $\boldsymbol{x}$ from the list $\mathcal{L}$.
  2. IF $\boldsymbol{x}$ is sufficiently small THEN
```

```
    (a) Place $\boldsymbol{x}$ on either $\mathcal{U}$ or $\mathcal{C}$.
    (b) CYCLE
    END IF
 3. (Constraint Propagation)
    (a) Use constraint propagation to possibly narrow the coordinate widths of
        the box $\boldsymbol{x}$.
    (b) IF constraint propagation has shown that $\boldsymbol{x}$ cannot contain solutions
        THEN CYCLE
 4. (Interval Newton)
    (a) Perform an interval Newton method to possibly narrow the coordinate
        widths of the box $\boldsymbol{x}$.
    (b) IF the interval Newton method has shown that $\boldsymbol{x}$ cannot contain solutions
        THEN CYCLE
 5. IF the coordinate widths of $\boldsymbol{x}$ are now sufficiently narrow THEN
    (a) Place $\boldsymbol{x}$ on either $\mathcal{U}$ or $\mathcal{C}$.
    (b) CYCLE
 6. (Subdivide)
    (a) Choose a coordinate index $k$ to bisect (i.e. to replace $[\underline{x}_k, \overline{x}_k]$ by $[\underline{x}_k, (\underline{x}_k +
        \overline{x}_k)/2]$ and $[(\underline{x}_k + \overline{x}_k)/2, \overline{x}_k]$).
    (b) Bisect $\boldsymbol{x}$ along its $k$-th coordinate, forming two new boxes; place these
        boxes on the list $\mathcal{L}$.
    (c) CYCLE

END DO
```

End Algorithm 1
Notes:

1. Traditional techniques, such as the "midpoint test" and "gradient test" for rejecting boxes with high values of the global optimum or boxes that cannot contain critical points, are included in steps 3 and 4, but are irrelevant here.
2. Determining when a box $\boldsymbol{x}$ is "small enough" is more sophisticated than simply checking the widths of the coordinates; see [24] for details.
3. Steps 2 and 5 of Algorithm 1 are more involved in GlobSol than simply placing the result box on the list. In particular, an attempt is made to find approximate optima and do ϵ-inflation (as in [16, §4.2]), to avoid excessive subdivisions around solutions. The process includes the box complementation scheme of [16, §4.3.1]. Embodied in internal subroutine `HANDLE_LEAF` in file `f90intbi/rigorous_global_search.f90`. This process is subject to change in the future, with an eye towards simplification.

GlobSol has numerous sub-algorithms. For example, when there are equality constraints, the "midpoint test" for determining a rigorous upper bound on global minima necessarily is more complicated than evaluating the objective function at a point. In particular, the point of evaluation must be known to be feasible. We explain the process in GlobSol for verifying feasibility in [18] and

[16, §5.2.4]. This process is presently a weakness that prevents some equality-constrained problems from being handled as efficiently as unconstrained or certain inequality-constrained problems.

GlobSol also attempts to find approximate feasible points, for use in the midpoint test. A generalized Newton method is used for this purpose. See the report "An Iterative Method for Finding Approximate Feasible Points" at

```
    http://interval.louisiana.edu/GlobSol/
                                Dian-approximate-optimizer.pdf
```

5 Use of GlobSol: Examples and Advice

Once installed, GlobSol requires the following files to run.

`GlobSol.CFG`: The GlobSol configuration file, this comes with default settings that the user need not initially change; however, see below.

`*.DT?`: The box data file, this user-supplied file defines the limits of the search box $\boldsymbol{x}$ and specifies which of the coordinate bounds are to be considered bound constraints for the purposes of "peeling".

`*.f90`: The user supplies a Fortran 90 program to define the objective and constraints.

An example is the simple illustration of mixed constraints, found in the `integration_test_data` subdirectory of GlobSol. The supplied box data file, named "`mixed.DT1`", is:

```
 1D-5
 0   1
 0   1
F F
F F
```

The first line signifies a tolerance of 10^{-5}; actual answer boxes can be expected to have relative widths up to the square root of this tolerance. The second and third lines specify bounds $\boldsymbol{x}_1 = [0, 1]$ and $\boldsymbol{x}_2 = [0, 1]$ on the initial search region $\boldsymbol{x}$. The last two lines specify that none of the search region bounds are to be considered as bound constraints for the purposes of the peeling process.

The supplied Fortran 90 source file, named "`mixed.f90`", is:

```
 PROGRAM SIMPLE_MIXED_CONSTRAINTS
  USE CODELIST_CREATION

      PARAMETER (NN=2)
      TYPE(CDLVAR), DIMENSION(NN):: X
      TYPE(CDLLHS), DIMENSION(1):: PHI
      TYPE(CDLINEQ), DIMENSION(2):: G
      TYPE(CDLEQ), DIMENSION(1) :: C

      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
```

Here, the objective is defined with the special variable type CDLLHS ("code list left hand side"), the inequality constraints with CDLINEQ, and the equality constraints with CDLEQ. (Observe the slight difference with the explanation in [16].) Each of these variables should be considered "write-once;" that is, they should appear only once and in left-hand-sides of assignment statements.

Issuing the command "globsol mixed 1" invokes the GlobSol script to produce the output file mixed.OT1. Here is an abridgement of this file:

```
 Output from FIND_GLOBAL_MIN on  07/28/2002  at  14:53:47.
 Version for the system is: October 10, 2000

 Codelist file name is: mixedG.CDL
 Box data file name is: mixed.DT1

 Initial box:
 [    0.0000D+00,   0.1000D+01 ] [    0.0000D+00,   0.1000D+01 ]
 BOUND_CONSTRAINT:
   F F   F F
 ---------------------------------------
 CONFIGURATION VALUES:
 EPS_DOMAIN:   0.1000D-04   MAXITR:   20000
 MAX_CPU_SECONDS:    0.3600D+04
 DO_INTERVAL_NEWTON: T  QUADRATIC: T  FULL_SPACE: F
 ...
 (additional configuration variables are printed here.)
 ...
 Default point optimizer was used.

 THERE WERE NO BOXES IN THE LIST OF SMALL BOXES.
 LIST OF BOXES CONTAINING VERIFIED FEASIBLE POINTS:
 Box no.:           1
 Box coordinates:
 [    0.7071D+00,   0.7071D+00 ] [    0.7071D+00,   0.7071D+00 ]
PHI:
 [   -0.1500D+01,  -0.1500D+01 ]
 B%LIUI(1,*):
 F F
 B%LIUI(2,*):
 F F
 B%SIDE(*):
 F F
 B%PEEL(*):
 T T
 Level:            1
 Box contains the following approximate root:
    0.7071D+00   0.7071D+00
 OBJECTIVE ENCLOSURE AT APPROXIMATE ROOT:
 [   -0.1500D+01,  -0.1500D+01 ]
 Unknown =  T    Contains_root = T
 Changed coordinates:
 T F
 U0:
 [    0.3852D+00,   0.3852D+00 ]
 U:
 [    0.5777D+00,   0.5777D+00 ] [    0.0000D+00,   0.1000D+01 ]
 V:
 [    0.1926D+00,   0.1926D+00 ]
 INEQ_CERT_FEASIBLE:
  F T
 NIN_POSS_BINDING:           1
 -------------------------------------------------
 ALGORITHM COMPLETED WITH LESS THAN THE MAXIMUM NUMBER,
       20000  OF BOXES.
 Number of bisections:            1
 No. dense interval residual evaluations -- gradient code list:           53
 ...
```

```
(Additional performance information is printed here.)
...
Total number of boxes processed in loop:           4
BEST_ESTIMATE:   -0.1500D+01
Overall CPU time:    0.1001D-01
CPU time in PEEL_BOUNDARY:    0.0000D+00
CPU time in REDUCED_INTERVAL_NEWTON:    0.0000D+00
```

Here, the "default point optimizer" is the generalized Newton method of [21]. The components `LIUI`, `SIDE`, and `PEEL` of the box data type `B` need not concern the user. Generally, the "approximate root" is the midpoint of the box, while `U0`, `U`, and `V` represent bounds on the generalized Lagrange multipliers for the objective, inequality constraints, and equality constraints. Bounds of $[0, 1]$ often correspond to inequality constraints that are not active; this is the case for this problem, since, from the line below the Lagrange multiplier bounds, we see that the second inequality constraint is "certainly feasible," meaning an interval evaluation gave a non-positive result[2] . The `BEST_ESTIMATE` is the best estimate for the global minimum, obtained through point evaluations or, when equality constraints are present, through evaluation of the objective over a small box within which a feasible point has been proven to exist. (This value is called $\overline{f}$ in some of the literature.)

5.1 Advice on Interpretation of Results

The only guarantees with GlobSol's algorithm are that any possible global minimizers must be contained in the union of the two lists of output boxes. Occasionally, the problem will not have global minimizers, but one or both lists are non-empty[3] . If a reasonable `BEST_ESTIMATE` has been found, then the boxes in the output lists should have relatively low objective values, even if they don't correspond to minimizers.

Conversely, GlobSol may complete with both the list of small boxes and list of boxes containing verified feasible points empty. For example, we expect both lists to be empty for a linear objective function without constraints. Adding constraints (or setting a sufficient number of flags in the box data file to indicate bound constraints) will cause GlobSol to output non-empty lists. Conceptually (although not from an implementation point of view), one can think of box limits not corresponding to bound constraints as topologically "open," and box limits corresponding to bound constraints as "closed"; extrema are guaranteed to exist only over closed, bounded regions.

5.2 Advice on the Configuration File

With numerous in-line comments, GlobSol's configuration file is to a large extent self-documenting. The file is organized into sections, consisting of:

[2] If a constraint is certainly feasible over a box, the constraint is dropped from the computation, but the corresponding Lagrange multiplier is actually zero. A future version of GlobSol will print "0" for such Lagrange multipliers.

[3] This can happen, for example, for a monotonic function without constraints, where interval overestimation prevents monotonicity from being detected.

1. limits associated with code list creation,
2. switches to control printing,
3. stopping tolerances,
4. limits on the algorithm, such as CPU time,
5. miscellaneous algorithm controls.

The limits associated with code list creation are necessary since not all memory could be dynamically allocated during this process. GlobSol should give an appropriate error message indicating which of these limits has been exceeded, if it is necessary to change these.

Levels of printing are independently available for the overall algorithm and various sub-algorithms. Most of these levels are for debugging, and will result in excessive output on practical problems. However, some may reveal useful information to aid in solving problems efficiently. Users may wish, for example, to set `PRINTING_IN_OVERALL_ALGORITHM` to 2 (from its default of 0).

Resource Limits and Tolerances We recommend setting `MAX_CPU_SECONDS` as desired, and setting `MAXITR`, representing the maximum number of boxes to be processed, higher if the default is exceeded.

Stopping tolerances may possibly be changed from defaults to enable GlobSol to complete for certain problems. Users should read [24] to understand the meaning of these tolerances.

Except as noted below, we do not envision it as usually appropriate for the user to change other switches that control the algorithms.

Turning On and Off Constraint Propagation Constraint propagation as explained in §2.2 above is "on" by default; experimentation has shown that, within GlobSol's overall algorithm, this constraint propagation usually saves significant amounts of computational effort. However, constraint propagation is not useful for a few problems. Furthermore, since GlobSol does not yet have full implementations for all inverses of all functions supported by the code list, constraint propagation is not possible for certain functions. GlobSol's constraint propagation at the code list level can be turned off by setting the configuration variable `USE_SUBSIT` to "`F`".

We at one time contemplated having the user define relations and inverses at a higher level than the elemental operations of the code list, for use in constraint propagation. Although this may be useful, it is not fully implemented in the present version of GlobSol.

5.3 Advice on Using the "Peeling" Process

The peeling process is an effective way of handling bound constraints when the number of variables n is small or when the number of actual bound constraints is small, especially when the number of additional equality constraints is small. This is because, during the process, we evaluate the objective on lower-dimensional sub-boxes representing the bound constraints and intersections of bound constraints, and can thus obtain better upper bounds (i.e. better `BEST_ESTIMATE` or $\overline{f}$) with minimal overestimation due to interval dependency.

However, although only subfaces with sufficiently low `BEST_ESTIMATE` are processed, peeling generates up to 2^m sub-boxes, where m is the number of bound constraints that have been set. (The number is 3^m if both lower and upper bounds correspond to bound constraints.) Therefore, for large numbers of variables, it is not advisable to set large numbers of bound constraints for peeling. One alternative is to selectively set a few bound constraints, experimenting until GlobSol gives non-empty answer lists. Another alternative is to make the search box slightly larger, and define the bound constraints as equality constraints. However, see below.

5.4 Advice on Equality Constraints

A present weakness in GlobSol is its treatment of equality constraints. In particular, because GlobSol verifies feasibility by forming a system from the equality and active inequality constraints, the number of equality constraints cannot exceed the number of unknowns. We are presently working on alternate paradigms to circumvent this problem. (See below.)

5.5 Advice on Data Fitting

GlobSol presently does not seem to handle data fitting problems (least squares, minimax, or least absolute value) well when there are large numbers of data points. (However, see [5].) We are presently working on an alternate paradigm; see §6.1 below.

6 The Experimental Version and GlobSol's Future

Nonlinear Systems We have quit maintaining the separate nonlinear system solver that was in early versions of the GlobSol package. Instead, in our experimental version of GlobSol, we are including a "nonlinear system" algorithm path. This allows us to maintain a single overall algorithm, as well as to take advantage of GlobSol's inequality constraint handling capabilities.

Taylor Models Our experimental version of GlobSol includes an ability to switch on and off Taylor arithmetic in evaluation of the objective and constraints, as well as to "symbolically" precondition interval Jacobi matrices with Taylor arithmetic, for interval Gauss–Seidel iteration;. See [2, 23] for results of using such Taylor models within our experimental version of GlobSol. For the Taylor arithmetic, we have interfaced GlobSol with the COSY-Infinity package of Berz et al, using Jens Hoefkens' Fortran 90 module [11].

6.1 Nonlinear Data Fitting

Although least squares, minimax, and ℓ_1 data fitting problems can be formulated within the present distribution GlobSol environment using Lemaréchal's conditions [25], directly as sums of squares, or with use of the augmented system (using the `LEAST_SQUARES` configuration variable), these approaches lead to excessive computation times for many problems. We speculate that this is because, with many data points, the resulting constraints or equations are approximately linearly dependent.

In [33], we proposed a new paradigm for linear least squares. This paradigm can be adapted to the nonlinear case. We have an algorithm path in the experimental version of GlobSol within which we are presently experimenting with this possibility.

References

[1] M. Berz, C. Bischof, G. Corliss, and A. Griewank, editors. *Computational Differentiation: Techniques, Applications, and Tools*, Philadelphia, 1996. SIAM.

[2] G. Corliss, Ch. Faure, A. Griewank, L. Hascoët, and U. Naumann, editors. *Automatic Differentiation of Algorithms: From Simulation to Optimization*, New York, 2002. Springer-Verlag.

[3] G. F. Corliss and R. B. Kearfott. Rigorous global search: Industrial applications. In *Developments in Reliable Computing*, pages 1–16, Dordrecht, Netherlands, 2000. Kluwer.

[4] J. E. Dennis and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Least Squares.* Prentice-Hall, Englewood Cliffs, NJ, 1983.

[5] C.-Y. Gau and M. A. Stadtherr. Nonlinear parameter estimation using interval analysis. *AIChE Symp. Ser*, 94(304):445–450, 1999.

[6] P. E. Gill, W. Murray, and M. Wright. *Practical Optimization.* Academic Press, New York, 1981.

[7] A. Griewank. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation.* Frontiers in Applied Mathematics. SIAM, Philadelphia, 2000.

[8] A. Griewank. ADOL-C, a package for automatic differentiation of algorithms written in C/C++, 2002. `http://www.math.tu-dresden.de/wir/project/adolc/`.

[9] A. Griewank and ed. Corliss, G. F., editors. *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, Philadelphia, 1991. SIAM.

[10] E. R. Hansen. *Global Optimization Using Interval Analysis.* Marcel Dekker, Inc., New York, 1992.

[11] J. Hoefkens. *Rigorous Numerical Analysis with High-Order Taylor Models.* PhD thesis, Department of Mathematics, Michigan State University, 2001.

[12] R. B. Kearfott. Abstract generalized bisection and a cost bound. *Math. Comp.*, 49(179):187–202, July 1987.

[13] R. B. Kearfott. Decomposition of arithmetic expressions to improve the behavior of interval iteration for nonlinear systems. *Computing*, 47(2):169–191, 1991.

[14] R. B. Kearfott. An interval branch and bound algorithm for bound constrained optimization problems. *Journal of Global Optimization*, 2:259–280, 1992.

[15] R. B. Kearfott. Algorithm 763: `INTERVAL_ARITHMETIC`: A Fortran 90 module for an interval data type. *ACM Trans. Math. Software*, 22(4):385–392, December 1996.

[16] R. B. Kearfott. *Rigorous Global Search: Continuous Problems.* Kluwer, Dordrecht, Netherlands, 1996.

[17] R. B. Kearfott. Empirical evaluation of innovations in interval branch and bound algorithms for nonlinear algebraic systems. *SIAM J. Sci. Comput.*, 18(2):574–594, March 1997.

[18] R. B. Kearfott. On proving existence of feasible points in equality constrained optimization problems. *Math. Prog.*, 83(1):89–100, September 1998.

[19] R. B. Kearfott. Interval analysis: Interval Newton methods. In *Encyclopedia of Optimization*, volume 3, pages 76–78. Kluwer, 2001.

[20] R. B. Kearfott, M. Dawande, K.-S. Du, and C.-Y. Hu. Algorithm 737: INTLIB, a portable FORTRAN 77 interval standard function library. *ACM Trans. Math. Software*, 20(4):447–459, December 1994.

[21] R. B. Kearfott and J. Dian. An iterative method for finding approximate feasible points, 1998. preprint, `http://interval.louisiana.edu/GlobSol/Dian-approximate-optimizer.pdf`.

[22] R. B. Kearfott and M. Novoa. Algorithm 681: INTBIS, a portable interval Newton/bisection package. *ACM Trans. Math. Software*, 16(2):152–157, June 1990.

[23] R. B. Kearfott and Walster G. W. Symbolic preconditioning with Taylor models: Some examples, 2001. accepted for publication in *Reliable Computing.*

[24] R. B. Kearfott and G. W. Walster. On stopping criteria in verified nonlinear systems or optimization algorithms. *ACM Trans. Math. Software*, 26(3):373–389, September 2000.

[25] C. Lemaréchal. Nondifferentiable optimization. In M. J. D. Powell, editor, *Nonlinear Optimization 1981*, pages 85–89, New York, 1982. Academic Press.

[26] A. P. Morgan. *Solving Polynomial Systems Using Continuation for Engineering and Scie ntific Problems.* Prentice-Hall, Englewood Cliffs, NJ, 1987.

[27] A. Neumaier. *Interval Methods for Systems of Equations.* Cambridge University Press, Cambridge, England, 1990.

[28] L. B. Rall. *Automatic Differentiation: Techniques and Applications.* Lecture Notes in Computer Science no. 120. Springer, Berlin, New York, etc., 1981.

[29] D. Ratz and T. Csendes. On the selection of subdivision directions in interval branch-and-bound methods for global optimization. *J. Global Optim.*, 7:183–207, 1995.

[30] C. A. Schnepper. *Large Grained Parallelism in Equation-Based Flowsheeting Using Interval Newton / Generalized Bisection Techniques.* PhD thesis, University of Illinois, Urbana, 1992.

[31] P. Van Hentenryck, L. Michel, and Y. Deville. *Numerica: A Modeling Language for Global Optimization.* MIT Press, Cambridge, MA, 1997.

[32] R. J. Van Iwaarden. *An Improved Unconstrained Global Optimization Algorithm.* PhD thesis, University of Colorado at Denver, 1996.

[33] J. Yang and R. B. Kearfott. Interval linear and nonlinear regression: New paradigms, implementations, and experiments, or new ways of thinking about data fitting, 2002, available at `http://interval.louisiana.edu/preprints/2002_SIAM_minisymposium.pdf`