INTERVAL ANALYSIS: INTERMEDIATE TERMS

Introduction.

In global optimization algorithms, the computer must repeatedly evaluate an objective function, as well as, possibly, inequality and equality constraints. Such functions are given as algebraic expressions or as subroutines or sections of computer code. When such computer code is executed, operations are applied to the *independent variables*, producing *intermediate terms*. These intermediate terms are, in turn, combined to produce other intermediate terms, or, eventually, the objective function value. For example, consider the problem

$$\boxed{\begin{array}{lcl} \text{minimize} \quad \phi(x) & = & x_1^2 - x_1^2 x_2^2 + x_2^2 \\ \text{over the box} \quad \boldsymbol{x} & = & ([-1,1],[-1,1])^T. \end{array}} \tag{1}$$

To evaluate ϕ, the computer may start with the independent variable values $v_1 = x_1$ and $v_2 = x_2$ internally produce quantities v_3, v_4, v_5, and v_6, to finally produce the dependent variable value $\phi(x) = v_7$. The following table indicates how this may be done.

$$\begin{array}{|llcl|} & v_1 & = & x_1, \\ & v_2 & = & x_2, \\ \hline \text{(i)} & v_3 & = & v_1^2, \\ \text{(ii)} & v_4 & = & v_2^2, \\ \text{(iii)} & v_5 & = & v_4 v_3, \\ \text{(iv)} & v_6 & = & v_3 - v_5, \\ \hline \text{(v)} & v_7 & = & v_6 + v_4. \end{array} \tag{2}$$

A list such as in (2) may be represented as a table of addresses of variables and operations. For example, if the operation $x_p \leftarrow x_q^2$ corresponds to operation code 5, $x_p \leftarrow x_q x_r$ corresponds to operation code 4, $x_p \leftarrow x_q + x_r$ corresponds to operation code 20, and $x_p \leftarrow x_q - x_r$ corresponds to operation code 21, then the set of relations (2) is represented by the table

OP	p	q	r
5	3	1	-
5	4	2	-
4	5	4	3
21	6	3	5
20	7	6	4

(3)

Such a sequence of operations is called a *code list*, but is sometimes called other things, such as a *tape*. Assuming the axioms of real arithmetic hold for evaluation, code lists for a given algebraic expression or portion of a computer program are not unique.

The concept of a code list is familiar to computer science students who have worked with compilers, since a compiler produces such lists while translating algebraic expressions into machine language. However, code lists and access to the intermediate expressions are of particular importance in interval global optimization, for the following reasons.

- Code lists provide a convenient internal representation for the objective and constraints, to be used for *automatic differentiation*, for both point and interval evaluation of objectives, gradients, and *Hessian matrices*.
- The values of the intermediate quantities can be used within the optimization algorithm in processes that reduce the size of the search region.
- Symbolic manipulation can reduce the overestimation, or *interval dependency* that would otherwise occur with interval evaluations.

Details are given below.

In Automatic Differentiation.

A code list can be used either as a pattern to specify the computations in the *forward mode* of automatic differentiation or as a symbolic representation of the system of equations to be solved

independent variables
intermediate terms
code list
automatic differentiation
Hessian matrices
interval dependency
forward mode→ automatic differentiation: forward mode

in the *backward mode*. See [5] for an in-depth look at the forward mode of automatic differentiation, and see [3] for somewhat more recent research on the subject. See [4, pp. 37–39] for some examples and additional references. Also see the articles on automatic differentiation in this encyclopedia.

In Constraint Satisfaction Techniques. Since each intermediate variable in the code list is connected to one or two others via an elementary, invertible operation, narrow bounds on one such intermediate variable can be used to obtain narrow bounds on others. For example, suppose that the code list (2) has been symbolically differentiated, to get the code list (5). Then, if the sub-box $\boldsymbol{x} = ([0.5, 1], [-1, -0.5])^T$ is to be considered for possible inclusion of optima, the derivative code list (5) can be evaluated by *forward substitution* to obtain the interval set of intermediate values (6). Furthermore, since (1) is an unconstrained problem, an optimum must occur where $\partial\phi/\partial x_1 = 0$ and $\partial\phi/\partial x_2 = 0$. In particular, any global optimizer x^* must have

$$v_{10}(x^*) = 0. \tag{4}$$

Using (4) in line (viii) of the derivative code list (5),

$$v_9 = v_8 - v_{10},$$

whence

$$\begin{array}{rcl} \tilde{\boldsymbol{v}}_9 & \leftarrow & [1,2] - 0 \\ \boldsymbol{v}_9 & \leftarrow & \tilde{\boldsymbol{v}}_9 \cap \boldsymbol{v}_9 = [1,2]. \end{array}$$

Now, using (vii) of (5),

$$\begin{array}{rcl} \tilde{\boldsymbol{v}}_4 & \leftarrow & \dfrac{\boldsymbol{v}_9}{\boldsymbol{v}_8} = \dfrac{[1,2]}{[1,2]} = [0.5, 2], \\ \boldsymbol{v}_4 & \leftarrow & \tilde{\boldsymbol{v}}_4 \cap \boldsymbol{v}_4 = [0.5, 1]. \end{array}$$

Now using (ii) of (5) gives

$$\begin{array}{rcl} \tilde{\boldsymbol{v}}_2 & \leftarrow & \sqrt{\boldsymbol{v}_4} \cup -\sqrt{\boldsymbol{v}_4} \\ & \subseteq & [0.70, 1] \cup [-1, -0.70], \\ \boldsymbol{v}_2 & \leftarrow & \tilde{\boldsymbol{v}}_2 \cap \boldsymbol{v}_2 = [-1, -0.70]. \end{array}$$

The last computation represents a narrowing of the range of one of the independent variables.

$$\begin{array}{|lrcl|} \hline & v_1 & = & x_1, \\ & v_2 & = & x_2, \\ \hline \text{(i)} & v_3 & = & v_1^2, \\ \text{(ii)} & v_4 & = & v_2^2, \\ \text{(iii)} & v_5 & = & v_4 v_3, \\ \text{(iv)} & v_6 & = & v_3 - v_5, \\ \text{(v)} & v_7 & = & v_6 + v_4, \\ \text{(vi)} & v_8 & = & 2v_1, \\ \text{(vii)} & v_9 & = & v_8 v_4, \\ \text{(viii)} & v_{10} & = & v_8 - v_9, \\ \text{(ix)} & v_{11} & = & 2v_2, \\ \text{(x)} & v_{12} & = & v_3 v_{11}, \\ \text{(xi)} & v_{13} & = & -v_{12}, \\ \text{(xii)} & v_{14} & = & v_{13} + v_{11}, \\ \hline & \phi & = & v_7, \\ \hline & \frac{\partial\phi}{\partial x_1} & = & v_{10}, \\ & \frac{\partial\phi}{\partial x_2} & = & v_{14}. \\ \hline \end{array} \tag{5}$$

$$\begin{array}{|rcc|} \hline \boldsymbol{v}_1 & = & [.5, 1], \\ \boldsymbol{v}_2 & = & [-1, -.5], \\ \hline \boldsymbol{v}_3 & = & [.25, 1], \\ \boldsymbol{v}_4 & = & [.25, 1], \\ \boldsymbol{v}_5 & = & [.0625, 1], \\ \boldsymbol{v}_6 & = & [-.75, .9375], \\ \boldsymbol{v}_7 & = & [-.5, 1.9375], \\ \boldsymbol{v}_8 & = & [1, 2], \\ \boldsymbol{v}_9 & = & [.25, 2], \\ \boldsymbol{v}_{10} & = & [-1, 1.75], \\ \boldsymbol{v}_{11} & = & [-2, -1], \\ \boldsymbol{v}_{12} & = & [-2, -.25], \\ \boldsymbol{v}_{13} & = & [.25, 2], \\ \boldsymbol{v}_{14} & = & [-1.75, 1], \\ \hline \phi & \in & [-.5, 1.9375], \\ \frac{\partial\phi}{\partial x_1} & \in & [-1, 1.75], \\ \frac{\partial\phi}{\partial x_2} & \in & [-1.75, 1]. \\ \hline \end{array} \tag{6}$$

(A similar computation could also have been carried out to obtain narrower bounds on v_1.)

If in addition an upper bound $\overline{\phi} = 0$ for the global optimum of ϕ is known, then

$$v_7 \in [-\infty, 0] \cap [-0.5, 1.9375] = [-0.5, 0].$$

backward mode$\longrightarrow$ *automatic differentiation:backward mode*
forward substitution

This can now be used in (v) of (5), along with new intermediate variable bounds, wherever possible, to obtain

$$\begin{aligned}\tilde{\boldsymbol{v}}_4 &\leftarrow \boldsymbol{v}_7 - \boldsymbol{v}_6\\ &= [-0.5, 0] - [-0.75, 0.9375]\\ &= [-1.4375, 0.75],\\ \boldsymbol{v}_4 &\leftarrow \boldsymbol{v}_4 \cap \tilde{\boldsymbol{v}}_4 = [0.5, 0.75].\end{aligned}$$

Now using (vii) of (5),

$$\begin{aligned}\tilde{\boldsymbol{v}}_9 &\leftarrow [1,2][0.5,0.75] = [0.5, 1.5],\\ \boldsymbol{v}_9 &\leftarrow \tilde{\boldsymbol{v}}_9 \cap \boldsymbol{v}_9 = [1, 1.5],\end{aligned}$$

then using (viii) and $v_{10} = 0$ gives $\boldsymbol{v}_8 = [1, 1.5]$. Finally, using (vi) of (5) gives

$$\boldsymbol{v}_1 \leftarrow [0.5, 0.75] \cap [0.5, 1] = [0.5, 0.75]. \qquad (7)$$

Now, evaluating ϕ in (1) (or redoing the forward substitution represented in (6)) at $(\boldsymbol{x}_1, \boldsymbol{x}_2) = ([0.5, 0.75], [-1, -0.70])$ gives

$$\begin{aligned}\phi &\in [.5,.75]^2 - [.5,.75]^2[-1,-.7]^2\\ &\qquad\qquad + [-1,-.7]^2\\ &= [.25,.5625] - [.25,.5625][.49,1] + [.49,1]\\ &= [.25,.5625] + [-.5625,-.1225] + [.49,1]\\ &= [.1775, 1.44],\end{aligned}$$

contradicting the known upper bound $\overline{\phi} = 0$. This proves that there can be no global optimizer of (1) within $([0.5,1],[-1,-0.5])^T$. (Note that, in fact, there are no global optimizers in $([-1,1],[-1,1])^T$ if the problem is considered to be unconstrained.)

The above procedure is easily automated, as is done in, say, GlobSol [2, 4], UniCalc [1], or other interval constraint propagation software.

This example illustrates a more general technique, associated with *constraint propagation* and *logic programming*. See [6] for an introduction to this view of the subject, and see [7] for alternate techniques of interval constraint satisfaction.

In Symbolic Preprocessing.

To understand how symbolic analysis based on the code list may help, consider the following example.

$$\boxed{\begin{array}{l}\text{Find all solutions to } f(x) = 0,\\ f = (f_1, f_2)^T, \text{ within the box}\\ \boldsymbol{x} = ([-2,0],[-1,1])^T, \text{ where}\\ f_1(x_1,x_2) = x_1^3 + x_1^2 x_2 + x_2^2 + 1\\ f_2(x_1,x_2) = x_1^3 - 3x_1^2 x_2 + x_2^2 + 1.\end{array}} \qquad (8)$$

A possible code list is

	v_1	$=$	x_1,	
	v_2	$=$	x_2,	
(i)	v_3	$=$	v_1^2,	
(ii)	v_4	$=$	v_2^2,	
(iii)	v_5	$=$	$v_3 v_2$,	(9)
(iv)	v_6	$=$	v_1^3,	
(v)	v_7	$=$	$v_6 + v_4$,	
(vi)	v_8	$=$	$v_7 + 1$,	
(vii)	$v_8 + v_5$	$=$	0,	
(viii)	$v_8 - 3v_5$	$=$	0.	

There is much interval dependency in this system, both in the individual equations (since each variable occurs in various terms), and between the equations (since the equations share common terms). However, examination of the code list (9) reveals that a change of variables can make the system more amenable to interval computation. Seeing that (vii) and (viii) are linear in v_5 and $v_8 = v_4 + v_6 + 1$, define

$$\begin{aligned}y_1 &= v_5 = x_1^2 x_2,\\ y_2 &= v_4 + v_6 = x_1^3 + x_2^2.\end{aligned} \qquad (10)$$

Then the system becomes

$$\begin{aligned}y_2 + y_1 + 1 &= 0\\ y_2 - 3y_1 + 1 &= 0.\end{aligned} \qquad (11)$$

Thus, the linear system (11) may be solved easily for y_1 and y_2. The interval bounds may then be plugged into (10) to obtain x_1 and x_2. There is no overestimation in any of the expressions for function components or partial derivatives in either (11) or (10).

Additional research should reveal how to automate this change of variables process.

constraint propagation
logic programming

References

[1] Babichev, A. B., Kadyrova, O. B., Kashevarova, T. P., Leshchenko, A. S., and Semenov, A. L.: 'UniCalc, a Novel Approach to Solving Systems of Algebraic Equations', *Interval Computations* **1993**, no. 2 (1993), 29–47.

[2] Corliss, G. F., and Kearfott, R. B.: 'Rigorous Global Search: Industrial Applications', in T. Csendes (ed.): *(Special issues of the journal "Reliable Computing")*, Kluwer, 1998.

[3] Griewank, A., and Corliss, G. F. (eds.): *Automatic Differentiation of Algorithms: Theory, Implementation, and Application* (Philadelphia, 1991), SIAM.

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

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

[6] Van Hentenryck, P.: *Constraint Satisfaction in Logic Programming*, MIT Press, Cambridge, MA, 1989.

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

R. Baker Kearfott
Department of Mathematics
University of Southwestern Louisiana
U.S.L. Box 4-1010, Lafayette, LA 70504-1010 USA
E-mail address: rbk@usl.edu

AMS 1991 Subject Classification: 65G10, 65H20.
Key words and phrases: expression parsing, constraint satisfaction techniques, interval dependencies, verification, interval computations, global optimization.