Numerica: a Modeling Language for Global Optimization

Pascal Van Hentenryck
Brown University
Box 1910
Providence, RI 02912, USA
pvh@cs.brown.edu

1 Introduction

Many science and engineering applications require the user to find solutions to systems of nonlinear constraints over real numbers or to optimize a nonlinear function subject to nonlinear constraints. This includes applications such the modeling of chemical engineering processes and of electrical circuits, robot kinematics, chemical equilibrium problems, and design problems (e.g., nuclear reactor design). The field of global optimization is the study of methods to find all solutions to systems of nonlinear constraints and all global optima to optimization problems. Nonlinear problems raise many issues from a computation standpoint. On the one hand, deciding if a set of polynomial constraints has a solution is NP-hard. In fact, Canny [Canny, 1988] and Renegar [Renegar, 1988] have shown that the problem is in PSPACE and it is not known whether the problem lies in NP. Nonlinear programming problems can be so hard that some methods are designed only to solve problems up to, say, 20 variables. On the other hand, computing over real numbers raises numerical problems because of the finite nature of computers.

Numerica [Van Hentenryck *et al.*, 1997c] is a modeling language for global optimization which makes it possible to solve nonlinear problems written in a form close to the statements traditionally found in textbooks and scientific papers. In addition, and contrary to most nonlinear programming tools, Numerica provides many guarantees on its results (modulo implementation errors):

- **Correctness:** Numerica never produces any wrong solution;
- **Completeness:** Under reasonable assumptions, Numerica is guaranteed to isolate all solutions to nonlinear equation systems and all global optima to unconstrained and constrained optimization problems.
- **Finiteness:** Numerica is guaranteed to converge;
- **Certainty:** Numerica can prove the existence of solutions and the absence of solutions.

These functionalities should be contrasted with traditional numerical methods (e.g., quasi-Newton methods). Traditional methods are inherently local: they converge quickly when they are close to a solution or to a local optimum but it is outside the scope of these methods to find all solutions (or global optima) or to prove the existence or absence of solutions. Traditional methods may also fail to converge on hard problems.

The limitations of local methods come from their inability to obtain *global* information on nonlinear functions. There is no way to collect global information on a function by probing finitely many points. In contrast, Numerica has the ability to evaluate nonlinear functions over intervals, which provides global information on the value of the function on any point in the intervals. The global nature of this information makes it possible to bound numerical errors automatically and to prune away entire regions of search space. As a consequence, the use of intervals makes it possible to implement global search algorithms for nonlinear programming.

Of course, the use of intervals in numerical computations is hardly new, since it originated from Moore's thesis in 1966 [Moore, 1966] and is a very active research area (e.g., [Hammer *et al.*, 1993; Hansen, 1992; Hansen and Greenberg, 1983; Hansen and Sengupta, 1981; Hong and Stahl, 1994; Kearfott, 1990; 1991; 1997; Krawczyk, 1969; Moore, 1966; Neumaier, 1990; Rump, 1988]). What distinguishes the constraint-solving algorithm of Numerica is the combination of techniques from numerical analysis and artificial intelligence to obtain effective pruning techniques (for many problems). At a very abstract level, Numerica can be viewed as mapping continuous problems into discrete problems, which is exactly the opposite of traditional relaxation techniques (e.g., in integer programming [Garfinkel and Nemhauser, 1972]). Once nonlinear programming problems are viewed as discrete problems, it is natural to apply consistency techniques such as arc- and path-consistency (e.g., [Montanari, 1974; Mackworth, 1977; Mackworth and Freuder, 1985]) which have been successfully applied in many areas [Van Hentenryck and Saraswat, 1996].

Numerica, and its constraint-solving algorithm, does not aim at replacing local methods. Local methods are extremely effective tools when they apply and are probably the only way to approach large-scale nonlinear programming problems involving thousands of variables. However, there are many applications where the additional functionalities of NUMERICA are needed, either because of the nature of the application, or because the problem is too hard for local methods, or simply because the robustness of the approach simplifies the task. This is especially true for small-scale highly nonlinear problems as those found in chemical and electrical engineering where traditional methods are likely to diverge, are unable to locate all solutions or to prove the absence of solutions (a requirement in these problems). Reference [Gehrke and Marquardt, 1996] in fact indicates that progress in chemical engineering increases the need for these functionalities.

The rest of this extended abstract illustrates the advantages of NUMERICA and contrasts it with traditional methods. More information about NUMERICA can be found in [Van Hentenryck *et al.*, 1997c; 1997a; Michel and Van Hentenryck, 1997; Van Hentenryck *et al.*, 1997b].

2 What is Possible and What is Not?

Today's computers can manipulate and store only a finite amount of information. Since the solution of a nonlinear problem may be a real number that cannot be represented in finite space or displayed on a screen in finite time, the best we can hope for in general is a point close to a solution (preferably with some guarantee on its proximity to the solution) or an interval enclosing a solution.

Computer methods for solving nonlinear problems typically use floating-point numbers to approximate real numbers. Since there are only finitely many floating-point numbers, these methods are bound to make numerical errors. These errors, although probably small considered in isolation, may have fundamental implications on the results. Consider, for instance, Wilkinson's problem, which consists in finding all solutions to the equation

$$\prod_{i=1}^{20}(x+i) + px^{19} = 0$$

in the interval $[-20.4, -9.4]$. When $p = 0$, the equation obviously has 11 solutions. When $p = 2^{-23}$, it has no solution. Wilkinson's problem clearly indicates that a small numerical error (e.g., assume that p is the output of some numerical computation) can have fundamental implications for the results of an application. These numerical issues require users of numerical software to exercise great care when interpreting their results. With this in mind, consider the combustion problem, which consists in finding positive values for $x_i \quad (1 \leq i \leq 10)$ satisfying the equations

$$\begin{cases} x_2 + 2x_6 + x_9 + 2x_{10} = 10^{-5} \\ x_3 + x_8 = 3\ 10^{-5} \\ x_1 + x_3 + 2x_5 + 2x_8 + x_9 + x_{10} = 5\ 10^{-5} \\ x_4 + 2x_7 = 10^{-5} \\ 0.5140437\ 10^{-7}\ x_5 = x_1^2 \\ 0.1006932\ 10^{-6}\ x_6 = 2x_2^2 \\ 0.7816278\ 10^{-15}\ x_7 = x_4^2 \\ 0.1496236\ 10^{-6}\ x_8 = x_1 x_3 \\ 0.6194411\ 10^{-7}\ x_9 = x_1 x_2 \\ 0.2089296\ 10^{-14}\ x_{10} = x_1 x_2^2. \end{cases}$$

Using $(0.5, \ldots, 0.5)$ as starting point and the default setting of the system, a well-known commercial system produces a point, say a. In the same conditions but with the defaults set to obtain the highest numerical precision, the same commercial system produces another point, say b, and prints a warning that the machine precision is not sufficient to achieve the desired accuracy. It is not obvious in this case how to interpret these results in a meaningful way.

It is also interesting to mention the common belief that proving the existence or uniqueness of solutions is outside the scope of computer algorithms. For instance, Dennis and Schnabel in their excellent text [Dennis and Schnabel, 1983] present the three functions

$$\begin{aligned} f_1(x) &= x^4 - 12x^3 + 47x^2 - 60x \\ f_2(x) &= x^4 - 12x^3 + 47x^2 - 60x + 24 \\ f_3(x) &= x^4 - 12x^3 + 47x^2 - 60x + 24.1 \end{aligned}$$

and state

> *It would be wonderful if we had a general-purpose computer routine that would tell us: "The roots of $f_1(x)$ are $x = 0, 3, 4,$ and 5; the real roots of $f_2(x)$ are $x = 1$ and $x \cong 0.888$; $f_3(x)$ has no real roots."*
> *It is unlikely that there will ever be such a routine. In general, the questions of existence and uniqueness —does a problem have a solution and is it unique?— are beyond the capabilities one can expect of algorithms that solve nonlinear problems. In fact, we must readily admit that for any computer algorithm there exist nonlinear functions (infinitely continuously differentiable, if you wish) perverse enough to defeat the algorithm. Therefore, all a user can be guaranteed from any algorithm applied to a nonlinear problem is the answer, "An approximate solution to the problem is ..." or "No approximate solution to the problem was found in the allocated time."*

This statement is not correct in general and applies mainly to local methods. Such wonderful procedures in fact exist (within the limits imposed by the finite nature of computers) and one of them is used in NUMERICA.

Let us conclude this section by showing the behavior of NUMERICA on the above examples. On Wilkinson's

```
Pragma:
   precision = 1e-12;
Constant :
   range idx = [1..10];
Variable:
   x:array[1..10] in [0..1];
Body:  solve system all
   x[2] + 2 * x[6] + x[9] + 2*x[10] = 1e-5;
   x[3] + x[8] = 3e-5 ;
   x[1]+x[3]+2*x[5]+2*x[8]+x[9]+x[10]=5e-5;
   x[4] + 2 * x[7] = 1e-5;
   0.5140437e-7 * x[5] = x[1]^2;
   0.1006932e-6 * x[6] = 2 * x[2]^2;
   0.7816278e-15 * x[7] = x[4]^2;
   0.1496236e-6 * x[8] = x[1]*x[3];
   0.6194411e-7 * x[9] = x[1]*x[2];
   0.2089296e-14 * x[10] = x[1]*x[2]^2;
```

Figure 1: A Combustion Problem.

problem, NUMERICA returns $-20, -19, \ldots, -10$ as solutions and proves their existence when $p = 0$; it proves the absence of solutions when $p = 2^{-23}$. On the combustion problem whose statement is depicted in Figure 1, NUMERICA returns the unique positive solution and proves its existence in about 0.1 second. Solution b produced by the commercial system mentioned previously is close to being contained in this output box. On the functions

$$\begin{array}{l} f_1(x) = x^4 - 12x^3 + 47x^2 - 60x \\ f_2(x) = x^4 - 12x^3 + 47x^2 - 60x + 24 \\ f_3(x) = x^4 - 12x^3 + 47x^2 - 60x + 24.1 \end{array}$$

and for an initial range $[-10^8..10^9]$, NUMERICA returns the four ranges enclosing the solutions and proves the existence of a solution in each of them for f_1; it returns two ranges and proves the existence of a solution in each of them for f_2; it shows the absence of solutions for f_3. The computation times for these examples are negligible. More precisely, the NUMERICA statement

```
Variable:
   x in [0..1e8];
Body:
   solve system all
      x^4 - 12 * x^3 + 47 * x^2 - 60 * x + 24 = 0;
```

produces the following output boxes:

```
Solution:  1 [SAFE]
 x = 0.8883057790717 + [0.4e-13 , 0.6e-13]

Solution:  2 [SAFE]
 x = 1.0 + [-0.1e-13 , +0.1e-13]
```

3 Local Versus Global Optimum

Traditional globally convergent methods when applied to a minimization problem converge to a local optimum from almost all starting points. They are unable however to isolate all local optima and the global optima. This limitation is well illustrated by the minimization of the function

$$\sum_{i=1}^{5} i \ \cos((i+1) \ x \ + \ i))$$

and the maximization of the function

$$-(\sum_{i=1} 5i \ \cos((i-1) \ x_1 \ + \ i))(\sum_{i=1} 5i \ \cos((i+1) \ x_2 \ + \ i))$$

as well as by the minimization of the function $f(x_1, \ldots, x_n)$ defined as

$$10 \sin(\pi y_1)^2 + (y_n - 1)^2 + \sum_{i=1}^{n-1} (y_i - 1)^2 (1 + 10 \sin(\pi y_{i+1})^2).$$

These functions have many local minima. For instance, the last function has 10^{10} local minima when $n = 10$ but only a single global minimum. It is unlikely that a local method will converge towards a global minimum without external knowledge about these problems.[1] Also, a local method will never be able to prove that the global minimum has been found. In contrast, NUMERICA isolates all global optima to these functions without difficulty. Figure 2 is the NUMERICA statement for the last problem and it involves several of the features of the languages: input constant, minimization, function, and summation. In addition, it uses a trigonometric function `sin` and a predefined constant `pi`. NUMERICA seems to be essentially quadratic in the number of variables on this problem, as shown in Table 1.

```
Input:
   int n :  "Number of variables";
Constant:
   range idx = [1..n];
Variable:
   x :  array[idx] in [-10..10];
Function:
   y(i in idx) = 1 + 0.25 * (x[i]-1);
Body:
   minimize
      10 * sin(pi*y(1))^2 + (y(n) - 1)^2
      + Sum(i in [1..n-1])
          (y(i) -1)^2 * (1 + 10 * sin(pi*y(i+1))^2);
```

Figure 2: Unconstrained Optimization: Levy 8'.

4 Convergence

In addition to the above theoretical limitations, local methods also suffer from practical problems when implemented on a computer. One of the main problems is of course convergence. An interesting example in this context is the transistor modeling example of Ebers and

[1] Of course, there always exists a starting point that will converge towards the global optimum.

$$\begin{cases} (1-x_1x_2)x_3\ [\ exp(x_5(g_{1k}-g_{3k}x_7 10^{-3}-g_{5k}x_8 1e-3))-1\ ]-g_{5k}+g_{4k}x_2=0\ \ (1\leq k\leq 4) \\ (1-x_1x_2)x_3\ [\ exp(x_6(g_{1k}-g_{2k}-g_{3k}x_7 10^{-3}+g_{4k}x_9 10^{-3}))-1\ ]-g_{5k}x_1+g_{4k}=0\ \ (1\leq k\leq 4) \\ x_1x_3-x_2x_4=0 \end{cases}$$

Figure 3: The Transistor Model Problem

n	Solving Time (s)	Growth Factor
5	0.40	
10	1.20	3.00
20	4.30	3.58
40	27.10	6.30
80	136.60	5.04

Table 1: Performance Results on Levy 8'.

Moll [Ebers and Moll, 1954]. The problem is to find a solution to the system of nonlinear equations depicted in Figure 3 where the variables x_i must take their values in $[0, 10]$ and the constants are given by

0.485	0.752	0.869	0.982
0.369	1.254	0.703	1.455
5.2095	10.0677	22.9274	20.2153
23.3037	101.779	111.461	191.267
28.5132	111.8467	134.3884	211.4823

The article [Ratschek and Rokne, 1993] summarizes various attempts to find a solution to this problem using local methods and states

> *In 1974, Cutteridge [Cutteridge, 1971] combined local damped Newton-Raphson steps with the conjugate gradient method and a second-order gradient-descent method with eigenvalue determination where the two latter methods were applied to the least squares problem [...] Cutteridge emphasized that only the sophisticated combination of the three methods had led to a positive result, i.e., it did not suffice to only use the first two approaches mentioned above [...]*

Numerica finds the unique solution to the transistor modeling problem in the box $[0.10]^9$ and proves its existence and the absence of other solutions in less than 40 minutes. The previous interval solution required more than 14 months on a network of workstations.

Another important practical problem is convergence to an undesired solution, i.e., a solution that fails to satisfy some external constraints not included in the problem statement. Globally convergent algorithms are guaranteed to converge to some solution or some local minimum from almost any starting point, but they may fail to produce a given solution. For instance, a traditional quasi-Newton method applied to the transistor modeling problem almost always converges to a solution in which some variables have negative values. Solution a produced by the commercial system on the combustion problem has some negative components. Morgan [Morgan, 1987] also mentions that these undesired convergences are typical of chemical equilibrium systems:

n	Solving Time (ms)	Growth Factor
5	100	
10	500	5.00
20	1600	3.20
40	4200	2.65
80	9800	2.33
160	30700	3.13

Table 2: Performance Results on Broyden's Function.

> *The other day an electrochemist friend came by my office with a problem. He was trying to work out part of a battery-plate manufacturing process. He had set up a math model to determine the amounts of various metal compounds that would be present in the plating bath at various times. He had ended up with a system of 10 polynomial equations in 10 unknowns. His problem was that Newton's method kept converging to nonphysical solutions. [...] This incident has been repeated in various guises many times.*

5 Practicality

The functionalities of Numerica of course come at a price. The intractable nature of nonlinear programming precludes any guarantee on the computation times of interval methods. Conventional wisdom claims that interval methods are too slow to be of practical use and that their guarantees and ease of use come at too high a price. The performance of Numerica indicates that, for a rich collection of nonlinear problems, the price to pay is reasonable. Moreover, even when the full functionality of global methods is not needed, Numerica avoids the tedious work necessary to tune local methods and find suitable starting points. As a consequence, Numerica's ease of use and robustness frequently compensate for a longer running time and may even reduce the actual time to obtain a solution. In this context, it may be useful to mention that Numerica takes essentially linear time in the number of variables to isolate the zeros of the Broyden banded function, a traditional benchmark from numerical analysis, even when the initial range of the variable is as large as or larger than, say, $[-10^8, 10^8]$. See Figure 4 for a description of the

```
Input:
   int n :  "Number of variables:  ";
Constant:
   range idx = [1..n];
Set:
   J[i in idx] = { j in [max(1,i-5)..min(n,i+1)] | j <> i };
Variable:
   x :  array[idx] in [-10e8..10e8];
Body:
   solve system all
      [i in idx]:
          0 = x[i] * (2 + 5 * x[i]^2) + 1 - Sum(k in J[i]) x[k] * (1 +x[k]);
```

Figure 4: The Broyden Banded Function.

Broyden banded function in NUMERICA and Table 2 for experimental results.

In addition, NUMERICA compares well and frequently outperforms continuation methods on their benchmarks. This good performance comes from a novel combination of interval analysis methods (e.g., Hansen-Sengupta's operator) and constraint satisfaction techniques. The combination of these orthogonal techniques gives surprisingly good results on many problems, although understanding its strengths and limitations more formally requires further research.

Of course, there are also classes of problems for which interval methods are not appropriate at this point because interval evaluations may lose too much precision. For instance, nonlinear least-squares problems are not amenable to effective solution with the interval methods of which we are aware. Interval methods converge, of course, on these applications but they do not compare well in efficiency with local methods.

6 Challenges and Opportunities

There are many possible ways to improve global methods for nonlinear programming and we mention some of them without trying to be exhaustive. A particularly interesting research avenue (studied by F. Benhamou and D. Kapur for instance) is the combination of symbolic and numerical methods. New pruning techniques with a more global view of the problem is also of paramount importance to improve the pruning when far from a solution. Similarly, it would be interesting to study ways of collecting global information beyond intervals. Finally, constraint satisfaction techniques have been a driving force behind the development of NUMERICA but only a tiny fraction of the existing research is exploited in NUMERICA. It is an exciting field and it is likely to evolve substantially in the coming years.

Acknowledgment NUMERICA was developed jointly with Laurent Michel and Yves Deville. NUMERICA is also based on previous work on `Newton` with Frédéric Benhamou, Deepak Kapur, and David McAllester. Thanks to all of them for their invaluable contributions. Part of this research was supported by the Office of Naval Research under grant ONR Grant N00014-94-1-1153 and a NSF National Young Investigator Award.

References

[Canny, 1988] J. Canny. Some Algebraic and Geometric Computations in PSPACE. In *Proc. 20th ACM Symposium on the Theory of Computing*, pages 460–467, 1988.

[Cutteridge, 1971] O.P.D. Cutteridge. Powerful 2-part Program for Solution of Nonlinear Simultaneous Equations. *Electronics Letters*, 10, 1971.

[Dennis and Schnabel, 1983] J.E. Dennis and R.B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice Hall, Englewood Cliffs, New Jersey, 1983.

[Ebers and Moll, 1954] J.J. Ebers and J.L. Moll. Large-Scale Behaviour of Junction Transistors. *IEE Proc.*, 42:1761–1772, 1954.

[Garfinkel and Nemhauser, 1972] R.S Garfinkel and G.L Nemhauser. *Integer Programming*. John Wiley & Sons, New York, 1972.

[Gehrke and Marquardt, 1996] V. Gehrke and W. Marquardt. Direct Computation of All Singular Points of Chemical Engineering Models Using Interval Methods. In *International Conference on Interval Methods and Computer Aided Proofs in Science and Engineering*, Wuerzburg, Germany, 1996.

[Hammer *et al.*, 1993] R. Hammer, M. Hocks, M. Kulisch, and D. Ratz. *Numerical Toolbox for Verified Computing I – Basic Numerical Problems, Theory, Algorithms, and PASCAL-XSC Programs*. Springer-Verlag, Heidelberg, 1993.

[Hansen and Greenberg, 1983] E.R. Hansen and R.I. Greenberg. An Interval Newton Method. *Appl. Math. Comput.*, 12:89–98, 1983.

[Hansen and Sengupta, 1981] E.R. Hansen and S. Sengupta. Bounding Solutions of Systems of Equations Using Interval Analysis. *BIT*, 21:203–211, 1981.

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

[Hong and Stahl, 1994] H. Hong and V. Stahl. Safe Starting Regions by Fixed Points and Tightening. *Computing*, 53(3-4):323–335, 1994.

[Kearfott, 1990] R.B. Kearfott. Preconditioners for the Interval Gauss-Seidel Method. *SIAM Journal of Numerical Analysis*, 27, 1990.

[Kearfott, 1991] R.B. Kearfott. A Review of Preconditioners for the Interval Gauss-Seidel Method. *Interval Computations 1*, 1:59–85, 1991.

[Kearfott, 1997] R.B. Kearfott. A Review of Techniques in the Verified Solution of Constrained Global Optimization Problems. (to appear), 1997.

[Krawczyk, 1969] R. Krawczyk. Newton-Algorithmen zur Bestimmung von Nullstellen mit Fehlerschranken. *Computing*, 4:187–201, 1969.

[Mackworth and Freuder, 1985] A.K. Mackworth and E.C. Freuder. The Complexity of Some Polynomial Network Consistency Algorithms for Constraint Satisfaction Problems. *Artificial Intelligence*, 25:65–74, 1985.

[Mackworth, 1977] A.K. Mackworth. Consistency in Networks of Relations. *Artificial Intelligence*, 8(1):99–118, 1977.

[Michel and Van Hentenryck, 1997] L. Michel and P. Van Hentenryck. Helios: A Modeling Language for Global Optimization and its Implementation in Newton. *Theoretical Computer Science*, 173(1):3–48, February 1997.

[Montanari, 1974] U. Montanari. Networks of Constraints : Fundamental Properties and Applications to Picture Processing. *Information Science*, 7(2):95–132, 1974.

[Moore, 1966] R.E. Moore. *Interval Analysis.* Prentice-Hall, Englewood Cliffs, NJ, 1966.

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

[Neumaier, 1990] A. Neumaier. *Interval Methods for Systems of Equations.* PHI Series in Computer Science. Cambridge University Press, Cambridge, 1990.

[Ratschek and Rokne, 1993] H. Ratschek and J. Rokne. Experiments Using Interval Analysis for Solving a Circuit Design Problem. *Journal of Global Optimization*, 3:501–518, 1993.

[Renegar, 1988] J. Renegar. A Faster PSPACE Algorith for the Existential Theory of the Reals. In *Proc. 29th IEEE Symp. Foundations of Computer Science*, pages 291–295, 1988.

[Rump, 1988] S.M. Rump. Verification Methods for Dense and Sparse Systems of Equations. In J. Herzberger, editor, *Topics in Validated Computations*, pages 217–231. Elsevier, 1988.

[Van Hentenryck and Saraswat, 1996] P. Van Hentenryck and V. Saraswat. Strategic Directions in Constraint Programming. *ACM Computing Surveys*, 28(4), December 1996.

[Van Hentenryck *et al.*, 1997a] P. Van Hentenryck, D. McAllister, and D. Kapur. Solving Polynomial Systems Using a Branch and Prune Approach. *SIAM Journal on Numerical Analysis*, 34(2), 1997.

[Van Hentenryck *et al.*, 1997b] P. Van Hentenryck, L. Michel, and F. Benhamou. Newton: Constraint Programming over Nonlinear Constraints. *Science of Computer Programming*, 1997. (to appear).

[Van Hentenryck *et al.*, 1997c] P. Van Hentenryck, L. Michel, and Y. Deville. *Numerica: a Modeling Language for Global Optimization.* The MIT Press, Cambridge, Mass., 1997.