



5-1983


An Interval Newton Method

E R. Hansen

R I. Greenberg

Loyola University Chicago, Rgreen@luc.edu

Follow this and additional works at: https://ecommons.luc.edu/cs_facpubs

 Part of the [Numerical Analysis and Computation Commons](#), and the [Numerical Analysis and Scientific Computing Commons](#)

Recommended Citation

Hansen, E R. and Greenberg, R I.. An Interval Newton Method. *Applied Mathematics and Computation*, 12, 2-3: 89-98, 1983. Retrieved from Loyola eCommons, Computer Science: Faculty Publications and Other Works, [http://dx.doi.org/10.1016/0096-3003\(83\)90001-2](http://dx.doi.org/10.1016/0096-3003(83)90001-2)

This Article is brought to you for free and open access by the Faculty Publications at Loyola eCommons. It has been accepted for inclusion in Computer Science: Faculty Publications and Other Works by an authorized administrator of Loyola eCommons. For more information, please contact ecommons@luc.edu.



This work is licensed under a [Creative Commons Attribution-NonCommercial-No Derivative Works 3.0 License](#).
Copyright © 1983 Published by Elsevier Inc.

AN INTERVAL NEWTON METHOD

E.R. Hansen

Lockheed Missiles and Space Co., Sunnyvale, California

R.I. Greenberg

Department of Systems Science and Mathematics,
Washington University, St. Louis, Missouri

Prepublication Manuscript.

Published in Applied Mathematics and Computation
12: 89-98 (1983).

Abstract

We introduce an interval Newton method for bounding solutions of systems of nonlinear equations. It entails three sub-algorithms. The first is a Gauss-Seidel type step. The second is a real (non-interval) Newton iteration. The third solves the linearized equations by elimination. We explain why each sub-algorithm is desirable and how they fit together to provide solutions in as little as one third or one quarter the time required by Krawczyk's method [7] in our implementations.

1. Introduction

Interval Newton methods of various types have been published for finding and bounding solutions of systems of nonlinear equations. For example, see [2], [5], [7]- [10]. We combine various features of these methods into a single algorithm of greater efficiency.

Consider a continuously differentiable function $f: \mathbb{R}^n \rightarrow \mathbb{R}^n$ with Jacobian J . Let y be a zero of f . Using Taylor's Theorem and expanding $f(y)$ about x yields

$$f(x) + J(\xi)(y-x) = f(y) = 0.$$

If X is a box (an interval vector) containing x and y , then the points indicated by the notation ξ are contained in X . Thus, if we replace $J(\xi)$ by the interval matrix $J(X)$, y is contained in the set Z of points z satisfying

$$f(x) + J(X)(z-x) = 0. \tag{1.1}$$

Interval Newton methods find an interval vector $N(x, X)$ containing Z . The process is iterated. Given a box $X^{(0)}$, a new iterate is obtained as $X^{(k+1)} = X^{(k)} \cap N(x^{(k)}, X^{(k)})$. In practice, the intersection is best done for a given component of $N(x^{(k)}, X^{(k)})$ as soon as it is obtained.

Hansen and Smith [6] showed that to solve (1.1), it is best to first multiply

by an approximate inverse of J^c , where J^c denotes the center of $J(X)$. That is, for each i and j , J_{ij}^c is the midpoint of $J_{ij}(X)$. If we let B denote our approximate inverse, the multiplication yields

$$M(z-x)=b \tag{1.2}$$

where $M=BJ(X)$ and $b=-Bf(x)$.

Interval Newton methods differ in how (1.2) is solved. As we point out below, they also differ in how the arguments of $J(X)$ are chosen. A method due to Krawczyk [7] does not attempt to obtain Z , but only a bound for Z . Hansen and Sengupta [5] introduced a superior method which uses a single step of the Gauss-Seidel iteration. This also provides only a bound on Z . With both ^{of} to these methods, even if $x \in X$ is a zero of f , so that Z is the single point x , the new box obtained is not x (although it contains x).

A method which does yield x as output in one step (with exact interval arithmetic) when x is a zero is to solve (1.2) by Gaussian elimination as suggested by Hansen [2].

We shall refer to this as the elimination method. Unfortunately, this method cannot always be employed since M may contain a singular matrix. More importantly, intervals tend to grow during the elimination process, and poor results may be obtained. If this interval growth did not occur, elimination would be preferable to the Krawczyk and Hansen-Sengupta methods because it seeks Z , and not just a bound for Z .

The algorithm proposed in this paper takes advantage of the strengths of both the elimination and the Hansen-Sengupta methods. In order to evade the growing intervals in the elimination method, a real (non-interval) iteration technique is employed to improve the value of x before elimination is attempted. In this technique, the approximate inverse B is used to obtain a point x at which $\|f(x)\|$ is small. This can also be of some help in the Hansen-Sengupta method.

Wolfe [10] introduced an "inner iteration" into the Krawczyk method to reduce the number of times $J(X)$ (as well as the corresponding B and the interval matrix

product $BJ(X)$ must be computed. He repeated the Krawczyk step a fixed number of times using the same interval matrix J . We have adopted this idea in performing some repetition of the elimination and Hansen-Sengupta steps before recalculating J . The real iteration may also be thought of as an inner iteration since it is performed repetitively without recalculating J (or B).

We now describe more fully our composite interval Newton algorithm.

2. Initial Hansen-Sengupta Step

For a reason to become apparent later, we perform one Hansen-Sengupta step immediately after calculating J and B , and multiplying through by B . Since in obtaining M , we multiplied J by an approximate inverse of its center, M should ideally approximate the identity matrix. Even so, a diagonal element of M may contain zero. If so, then we use extended interval arithmetic (see [5]) to solve the i -th equation of $M(z-x)=b$ for the i -th variable. That is

$$Y_i = x_i + R_i / M_{ij} \quad (2.1a)$$

where

$$R_i = b_i - \sum_{\substack{j=1 \\ j \neq i}}^n M_{ij} (X_j - x_j). \quad (2.1b)$$

Here we have replaced z_j by X_j for all $j \neq i$. The interval Y_i contains every solution z_i in X_i . We perform the calculations specified by (2.1) first for those i such that $0 \notin M_{ii}$, and in each case, we replace X_i by

$$X_i' = X_i \cap Y_i. \quad (2.2)$$

If the intersection is empty, we conclude that there is no solution in the box X . When the intersection is not empty, we use these improved X_i 's to calculate the Y_i 's for those i such that $0 \in M_{ii}$.

If both $0 \in R_i$ and $0 \in M_{ii}$, then $Y_i = [-\infty, \infty]$, and we gain no useful information. If $0 \notin R_i$ and $0 \in M_{ii}$ then Y_i consists of two semi-infinite intervals which exclude a gap (an open interval). The intersection $Y_i \cap X_i$ may be empty or consist of one or two intervals. We can deal with the first two cases

as we did when $0 \notin M_{ii}$. In the third case, where two new intervals arise, we would have to split the box X . For simplicity, we wish to do this only once during a given step of the outer iteration.

Thus, we find the largest gap which would split the box and do not use any gap in the other

Y_i 's which would cause a split. We also do not use the widest gap right away, but save it for later use after we apply other techniques to narrow the box.

Ironically, the extended interval arithmetic calculations provide a wider gap when x is a poor approximation for a zero of f . It is for that reason that we find a gap first before attempting to improve x , which is our next step.

3. Real Iteration

We attempt to find an improved approximation x for a zero of f in X (if one exists) by starting at $x^{(0)} = m(X)$, the center of the box X , and computing

$$x^{(k+1)} = x^{(k)} - Bf(x^{(k)})$$

for $k=0,1,2,\dots,s$. If $x^{(k+1)}$ is not in X , we find the point on the boundary of X which is on the line connecting $x^{(k)}$ to $x^{(k+1)}$ and use this as a replacement for $x^{(k+1)}$. The value of s is determined implicitly in that we continue until we find an $x^{(s+1)}$ such that $\|f(x^{(s+1)})\| > \frac{1}{2}\|f(x^{(s)})\|$. The norm we use is the Euclidean norm. We set x to whichever of $x^{(s)}$ and $x^{(s+1)}$ yields the smaller

norm of f . This x is then used for elimination or Hansen-Sengupta iteration. The real iteration is also stopped if $\|f(x^{(s+1)})\| < 10^{-3}$.

4. Elimination Iteration

If we have succeeded in finding an x such that $\|f(x)\|$ is sufficiently small, we attempt to perform an LU decomposition of M . If successful, we calculate $Bf(x)$, and we perform forward and back substitution to solve (1.2). Then we intersect our solution with X and calculate a new x at the center of the new X . Each time the width of X (the width of its widest component) decreases significantly (to .9 of the previous width, say), we repeat the elimination step, starting at the calculation of $Bf(x)$ with the new x .

5. Additional Hansen-Sengupta Iteration

If, for any reason, we are unable to complete an elimination step, we perform the Hansen-Sengupta iteration instead. Since we have already done our best to find a wide gap in the box X , we perform the operations specified by (2.1) and (2.2) for only those i such that $0 \notin M_{ii}$. Each time X improves significantly (as described in Section 4), we calculate a new x at its center and repeat the Hansen-Sengupta step.

6. Splitting the Box

If we have found a gap using the initial Hansen-Sengupta step described earlier, we make use of that gap to split the box (if it still does so; it may not if the box has been narrowed by the other steps performed). If we have found no gap, and we have not managed to improve the box width significantly at any stage, we split the box at the center of the widest component. When the box is split, one part is placed on a stack for later processing, and the other part becomes the current box. (We return to process the most recently stacked box whenever the current box is narrowed to the acceptable tolerance or is found to contain no solution.) Whether or not we split the box, we now return to the beginning of the outer step, the calculation of the Jacobian matrix.

7. Summary of Algorithm

1. Calculate $J(X)$.
2. Calculate B , the approximate inverse of J^c .
3. Calculate $M=BJ(X)$.
4. Set x equal to the center of X .
5. Perform a Hansen-Sengupta step for those i such that $0 \notin M_{ii}$.
6. Perform a Hansen-Sengupta step for those i such that $0 \in M_{ii}$, and save the largest gap.
7. Perform real iteration to improve x , starting at the center of X .

8. If $\|f(x)\|$ is not sufficiently small, skip to step 14.
9. Perform the LU decomposition of M if possible; otherwise skip to step 14.
10. Calculate $Bf(x)$ and solve for Z by forward and back substitution.
11. Replace X by $X \cap Z$.
12. If the width of X improved significantly, set x equal to the center of the new X, and return to step 10.
- 13 Skip to step 16.
14. Perform a Hansen-Sengupta step for those components such that $0 \notin M_{ii}$.
15. If X improved significantly, set x equal to the center of X, and return to step 14.
16. If we have a gap saved from step 6, use it. If the box splits, put one part on the stack, and keep the other part as the new X. Return to step 1.
17. If the box did not improve significantly in steps 5 and 6, in step 11, or in step 14, then split at the center of the widest component. Save one half on the stack, and use the other half as the new X.
18. Return to step 1.

If in steps 5 and 6, in step 11, or in step 14, the box is narrowed to within the acceptable tolerance, or is found not to contain a solution, the most recently stacked box becomes the new X, and we return to step 1. A sufficiently small box can be output and dropped internally. When we encounter an empty stack, we are done.

8. Experimental Results

Several problems were solved using various versions of the algorithm described above and, for comparison, various versions of the Krawczyk method. Sometimes, certain components of the algorithm we have introduced were left out; sometimes we placed fixed limits on the number of times to perform the inner iterations. We settled on the algorithm as described above with variability of the factor representing significant improvement of the box width. This allows the user to apply some control of how much inner iteration is performed, according to how

difficult it is to calculate the Jacobian, the inverse of its center, and the necessary interval arithmetic matrix product. If, for example, we consider reducing the box width to .9 of its old width to be significant, we shall be performing more inner iterations than if we use a factor of .6. The higher the dimensionality, or the more complicated the Jacobian, the higher this factor should be chosen. Various examples were used.

Shown below are representative ~~numerical~~ results for the Broyden banded function [1]

$$f_i(x) = x_i(2+5x_i^2) + 1 - \sum_{j \in J_i} x_j(1+x_j) \quad (i=1, \dots, n)$$

where $J_i = \{j : j \neq i, \max(1, i-5) \leq j \leq \min(n, i+1)\}$.

This function was chosen for easy programmability in arbitrary dimension. In each run the initial box was $[-1,1]$ in each component. There is one solution in this box. It was required that the solution box have a width less than 10^{-8} .

We now define the parameters used in the tabular results below:

s = the significant box width improvement factor

n_1 = the number of outer steps = the number of Jacobian evaluations

= the number of matrix inversions = the number of interval arithmetic matrix-matrix products.

n_2 = the number of interval arithmetic function evaluations

= the number of interval arithmetic matrix-vector products

= n_3 or $n_6 + n_7$,

n_3 = the number of Krawczyk steps,

n_4 = the number of real iterations = the number of real function evaluations,
= the number of real matrix-vector products

n_5 = the number of interval arithmetic LU decomposition attempts,

n_6 = the number of elimination steps,

n_7 = the number of Hansen-Sengupta steps for those i with $0 \notin M_{ii}$,

n_8 = the number of Hansen-Sengupta steps for those i with $0 \in M_{ii}$,

The time is minutes:seconds. The experiments were run on a slow computer: the HP 98-45.

Table 1. n=3, Krawczyk method with inner iteration

<u>s</u>	<u>n₁</u>	<u>n₂</u>	<u>n₃</u>	<u>time</u>
.6	57	80	80	8:34
.7	46	74	74	7:30
.8	44	114	114	9:52
.9	36	153	153	11:48

Table 2. n=3, Hansen-Sengupta method without inner iteration

<u>s</u>	<u>n₁</u>	<u>n₂</u>	<u>n₇</u>	<u>n₈</u>	<u>time</u>
.8	21	21	21	16	2:37

Table 3. n=3, new method

<u>s</u>	<u>n₁</u>	<u>n₂</u>	<u>n₄</u>	<u>n₅</u>	<u>n₆</u>	<u>n₇</u>	<u>n₈</u>	<u>time</u>
.6	13	27	18	2	3	24	13	2:31
.8	13	26	17	2	3	23	12	2:29
.9	12	27	17	2	3	24	12	2:28
.99	12	27	17	2	3	24	12	2:28

Table 4. n=5, Krawczyk method with inner iteration

<u>s</u>	<u>n₁</u>	<u>n₂</u>	<u>n₃</u>	<u>time</u>
.9	194	234	234	80:14

Table 5. n=5, Hansen-Sengupta method without inner iteration

<u>s</u>	<u>n₁</u>	<u>n₂</u>	<u>n₇</u>	<u>n₈</u>	<u>time</u>
.9	80	80	80	45	30:03

Table 6. n=5, new method

<u>s</u>	<u>n₁</u>	<u>n₂</u>	<u>n₄</u>	<u>n₅</u>	<u>n₆</u>	<u>n₇</u>	<u>n₈</u>	<u>time</u>
.9	46	88	47	2	6	82	38	23:17

9. Other Forms of J

It was shown in [2] and [4] that certain arguments used in evaluating the Jacobian J could be real rather than interval. The real arguments are values of x , the point about which f is expanded to get equation (1.1). If a real inner iteration is used (see Section 3), then this improved form of J changes because x changes.

Thus if we use a real inner iteration and use the better form of J (with some real arguments), we must recompute J and B after completing the real iterative steps. We have the competing options of use of a better Jacobian vs. use of a real inner iteration.

Obtaining the better Jacobian requires more sophisticated analysis and extra programming. In the few experiments we have done on problems of low dimension, it appears that it is somewhat more efficient to use the better Jacobian.

10. Perturbed Problems

Consider a problem in which parameters occur which are subject to variation or error. If they are represented by intervals bounding the possible values, the problem usually becomes one with a set of solutions. Generally, interval Newton methods do not yield convergence to the best interval bounding the solution set. We shall illustrate this fact by counter example and describe how we have tried to overcome this difficulty.

Consider the one-dimensional problem

$$f(x) = x^2 - [4,9].$$

The solution is obviously $X = [2,3]$ or $[-3,-2]$. Let $X^{(0)} = [0.1,4.9]$ and let $x^{(0)} = 2.5$, the midpoint of $X^{(0)}$. Then $f'(X^{(0)}) = 2X^{(0)} = [0.2,9.8]$. The one-dimensional Newton iterate $N(x^{(k)}, X^{(k)}) = x^{(k)} - f(x^{(k)})/f'(X^{(k)})$ ($k=0,1,2,\dots$) yields $N(x^{(0)}, X^{(0)}) = [-8.75,16.25]$ so that $X^{(1)} = X^{(0)} \cap N(x^{(0)}, X^{(0)}) = X^{(0)}$. That is, no improvement of $X^{(0)}$ occurs so the iteration stops; and the solution $[2,3]$ is not obtained sharply.

A multidimensional counterexample is possible even when f is linear.

Consider the set of equations from [3]:

$$f(x) = A^I x - b^I = 0$$

where

$$A^I = \begin{bmatrix} [2,3] & [0,1] \\ [1,2] & [2,3] \end{bmatrix}, \quad b^I = \begin{bmatrix} [0,120] \\ [60,240] \end{bmatrix}.$$

As shown in [3], the interval vector with narrowest interval components containing the solution set is

$$\begin{bmatrix} [-120,90] \\ [-60,240] \end{bmatrix}.$$

Suppose we choose $X^{(0)}$ so that it contains the solution set. Consider the case in which the left endpoint of each component of $X^{(0)}$ is less in magnitude than the corresponding right endpoint. For the Krawczyk method, $X^{(0)}$ can be as large as

$$\frac{1}{11} \begin{bmatrix} [-1545, 1845] \\ [-1800, 2940] \end{bmatrix} \doteq \begin{bmatrix} [-140.45, 167.73] \\ [-163.64, 267.27] \end{bmatrix}$$

and no further improvement occurs.

For the Hansen-Sengupta method, $X^{(0)}$ can be as large as

$$\begin{bmatrix} [-2865/22, 1845/11] \\ [-8040/77, 2940/11] \end{bmatrix} \doteq \begin{bmatrix} [-130.23, 167.73] \\ [-104.42, 267.27] \end{bmatrix}$$

without further improvement.

In [5], it was argued that the Hansen-Sengupta method can be expected to converge faster than the Krawczyk method. This same argument indicates that the Hansen-Sengupta method can be expected to converge to a sharper result for perturbed problems. This is born out by the above example.

In the one-dimensional case, it is possible to modify the Newton method so as to obtain the true solution set as a limit. We simply let $x^{(k)}$ be the left and right endpoints of $X^{(k)}$, alternately. Thus if $X^{(k)} = [X_L^{(k)}, X_R^{(k)}]$, then

$$X^{(2k+1)} = X^{(2k)} \cap N(X_L^{(2k)}, X^{(2k)}) \quad (k = 0, 1, \dots)$$

and
$$X^{(2k)} = X^{(2k-1)} \cap N(X_R^{(2k-1)}, X^{(2k-1)}) \quad (k = 1, 2, \dots)$$

It is not difficult to prove that this algorithm always converges to the solution interval.

In the n -dimensional case, there are 2^n corners of the region defined by an interval vector. It is not practical to let $x^{(k)}$ cycle through all of them when n is large. We choose to let $x^{(k)}$ alternate between the two extreme corners formed by the left endpoints and by the right endpoints of all the components of $X^{(k)}$. We do not know whether or not this assures convergence to the solution interval vector.

References

- [1] BROYDEN, C. G., The convergence of an algorithm for solving sparse non-linear systems, *Math. Comp.*, 25, 285-294, (1971).
- [2] HANSEN, E. R., On solving systems of equations using interval arithmetic, *Math. Comp.*, 22, 374-384, (1968).
- [3] HANSEN, E. R., On linear algebraic equations with interval coefficients, in *Topics in interval analysis*, Oxford Press, London, 1969 (edited by E. Hansen).
- [4] HANSEN, E. R., Interval forms of Newton's method, *Computing*, 20, 153-163, (1978).
- [5] HANSEN, E. R. & SENGUPTA, S., Bounding solutions of systems of equations using interval analysis, *BIT*, 21, 203-211, (1981).
- [6] HANSEN, E. R. & SMITH, R. R., Interval arithmetic in matrix computations, part II, *SIAM Jour. Numer. Anal.*, 4, 1-9, (1967).
- [7] KRAWCZYK, R., Newton-Algorithmen zur Bestimmung von Nullstellen mit Fehlerschranken, *Computing*, 4, 187-201, (1969).
- [8] MOORE, R. E., *Interval Analysis*, Prentice-Hall, Englewood Cliffs, 1966.
- [9] MOORE, R. E., *Methods and Applications of Interval Analysis*, SIAM, Philadelphia, 1979.
- [10] WOLFE, M. A., A modification of Krawczyk's algorithm, *SIAM Jour. Numer. Anal.*, 17, 376-379, (1980).