Lubin Vulkov  Jerzy Waśniewski
Plamen Yalamov (Eds.)

# Numerical Analysis and Its Applications

Second International Conference, NAA 2000
Rousse, Bulgaria, June 11-15, 2000
Revised Papers

Springer

# Solvers for Systems of Nonlinear Algebraic Equations – Their Sensitivity to Starting Vectors

Deborah Dent[1], Marcin Paprzycki[1], and Anna Kucaba-Pietal[2]

[1] School of Mathematical Sciences, University of Southern Mississippi
Hattiesburg, MS 39406-5106
[2] Department of Fluid Mechanics and Aerodynamics,
Rzeszow University of Technology
Rzeszow, W.Pola 2, Poland

**Abstract.** In this note we compare the sensitivity of six advanced solvers for systems of nonlinear algebraic equations to the choice of starting vectors. We will report on results of our experiments in which, for each test problem, the calculated solution was used as the center from which we have moved away in various directions and observed the behavior of each solver attempting to find the solution. We are particularly interested in determining the best global starting vectors. Experimental results are presented and discussed.

## 1 Introduction

Recently we can observe a growing interest in engineering problems resulting in large systems of nonlinear algebraic equations. For instance, in a real-world problem originating from avionics [11,12] a realistic model would require solution of 500+ equations, but due to the lack of convergence, the programs and methods used by the authors were unable to solve systems of more than 64 equations.

The mathematical theory and computational practice are well established when a system of linear algebraic equations or a single nonlinear equation is to be solved [18]. This is clearly not the case for systems of nonlinear algebraic equations. Our current research has shown both a lack of libraries of solvers and standard sets of test problems (different researchers use different test problems with only a small overlap). In this context we have to remember that, until recently, in the engineering practice, only systems with relatively few equations have been solved. This explains one of the problems of existing "popular" test cases. Most of them have a very small number of equations (2-10) and only very few are defined so that they can reach 100 equations. In our earlier work [6,7,8,9] we have reported on our efforts to collect most of the existing solvers and apply them to up to 22 of standard test problems with the number of equations ranging from 2 to 200. We were able to locate solvers based on *Newton*'s method and its modifications, *Brown's*, *bisection*, *continuation*, *hybrid* algorithms, and the *homotopy*, and *tensor* methods and applied each solver to the test problems collected from the literature and the Internet. We were able to conclude that we



can exclude the simple algorithms and in-house implementations from further testing, that methods like *homotopy* and *continuation* cannot be used as a black-box approach without more work and the *tensor* method seemed to be the most robust.

When the test problems were considered, we were able to find that five of them are easily solvable by all approaches and thus they are useless for testing purposes. The results of the remaining test problems allowed us to observe that proper choice of the starting vector has a strong effects on the solution process (bad selection of the starting vector can result in lack of convergence). Because the likelihood of convergence depends on the solution method and the problem to be solved, we decided to perform a behavior comparison of six advanced solvers for the test problems identified earlier.

In this note, we will report on results of our experiments in which, for each test problem, the perturbed solution, and initial starting vectors of all ones, zeros and random numbers were used to observe the behavior of each solver attempting to find the solution. Based on these experiments we will try to establish which solvers can handle global convergence.

The paper will be organized as follows. Section 2 briefly describes the solvers that used in our work. In section 3 we introduce the test problems. Section 4 will summarize the results of our numerical experiments followed by a concluding remarks and description of future work.

## 2 Solvers and Algorithms for Systems of Nonlinear Algebraic Equations

As mentioned above, in our earlier work, we have found that only more sophisticated algorithms are capable of solving test systems of nonlinear algebraic equations (outside of the group of five easy ones). We are now focusing on non-commercial versions of codes based on a *hybrid* algorithm and the *Brown's*, *homotopy*, *continuation*, and *tensor* methods. These algorithms are all documented in ACM TOMS and briefly reviewed in [15]. Their implementations were obtained from the NETLIB repository [17]. It is appropriate to use [15] as the reference where brief descriptions of the code are given. Further, in the subsections of §2 the original works where the methods were proposed have to be referred. We have thus modified (to handle up to 200 equations) the following software packages: 1) HYBRD, 2) SOS, 3) CONTIN, 4) HOMPACK, and 5) TENSOLVE. Recently we have also discovered and added to this list the LANCELOT package, which is a part of the NEOS environment [16].

We will now briefly summarize these algorithms and the solvers (in all cases the references cited and [18] should be consulted for the details). We assume that a system of $n$ nonlinear algebraic equations $f(\mathbf{x}) = \mathbf{0}$ is to be solved where $\mathbf{x}$ is n-dimensional vector and $\mathbf{0}$ is the zero vector.

### 2.1 HYBRD

HYBRD is part of the MINPACK-1 suite of codes [13,14]. HYBRD's design is based on a combination of a modified *Newton* method and the *trust region* method. Termination occurs when the estimated relative error less than or equal the defined by the user tolerance (we used the suggested default value of the square root of the machine precision).

### 2.2 SOS

SOS is a part of the SLATEC suites of codes [10]. SOS solves a system of N simultaneous nonlinear equations in N unknowns. It solves the problem $f(\mathbf{x}) = \mathbf{0}$ where $\boldsymbol{x}$ is a vector with components $x(1),...,x(N)$ and $\boldsymbol{f}$ is a vector of nonlinear functions. This code is based on an iterative method called the *Brown*'s method [2] which is a variation of Newton's method using Gaussian elimination in a manner similar to the Gauss-Seidel process. All partial derivatives required by the algorithm are approximated by first difference quotients.. The convergence behavior of this code is affected by the ordering of the equations, and it is advantageous to place linear and mildly nonlinear equations first in the ordering. Convergence is roughly quadratic. This method requires a good choice for the starting vector $\mathbf{x}_0$.

### 2.3 CONTIN

CONTIN, also know as PITCON [19] implements a continuation algorithm with an adaptive choice of a local coordinate. A *continuation* method is designed to be able to target more complicated problems and is the subject of various research efforts [1,20]. This method is expected to be slower than *linesearch* and the *trust region* methods, but it is to be useful on difficult problems for which a good starting point is difficult to establish. The method defines an easy problem for which the solution is known along with a path between the easy problem and the hard problem that is to be solved. The solution of the easy problem is gradually transformed to the solution of the hard problem by tracing this path. The path may be defined as by introducing an addition scalar parameter $\lambda$ into the problem and defining a function

$$h(\mathbf{x}, \lambda) = f(\mathbf{x}) - (1 - \lambda) f(\mathbf{x}_0) \tag{1}$$

where $x_0 \in \mathbf{R}^n$. The problem $h(\mathbf{x}, \lambda) = \mathbf{0}$ is then solved for values of $\lambda$ between *0* and *1*. When $\lambda=0$, the solution is clearly $\mathbf{x} = \mathbf{x}_0$. When $\lambda=1$, we have that $h(\mathbf{x},1)=f(\mathbf{x})$, and the solution of $h(\mathbf{x},\lambda)$ coincides with the solution of the original problem $f(\mathbf{x})=\mathbf{0}$. The algorithm for constructing the path is given in [19]. The convergence rate of the *continuation* methods varies, but according to documentation, the method does not require a good choice of the initial vector $\mathbf{x}_0$.



### 2.4 HOMPACK

HOMPACK [21] is a suite of subroutines for solving nonlinear systems of equations by *homotopy* methods [4]. The *homotopy* and *continuation* methods are closely related. In the *homotopy* method, a given problem $f(\mathbf{x})=\mathbf{0}$ is embedded in a one-parameter family of problems using a parameter $\lambda$ assuming values in the range $[0, \ldots, 1]$. Like the *continuation* method, the solution of an easy problem is gradually transformed to the solution of the hard problem by tracing a path. There are three basic path-tracking algorithms for this method: ordinary differential equation based (code FIXPDF), normal flow (code FIXPNF), and quasi *Newton* augmented Jacobian matrix (code FIXPQF). The code is available in both Fortran 77 and Fortran 90 [21]. The Fortran 77 version was used in our test. We tested all three approaches and since the results were very close, we will report FIXPDF results only.

### 2.5 TENSOLVE

TENSOLVE [3] is a modular software package for solving systems of nonlinear equations and nonlinear least-square problems using the *tensor* method. It is intended for small to medium-sized problems (up to 100 equations and unknowns) in cases where it is reasonable to calculate the Jacobian matrix or its approximations. This solver provides two different strategies for global convergence; a line search approach (default) and a two-dimensional trust region approach. The stopping criteria is meet when the relative size of $\mathbf{x}_{k+1} - \mathbf{x}_k$ is less than the $macheps^{\frac{2}{3}}$, or $\|f(x_{k+1})\|\infty$ is less than $macheps^{\frac{2}{3}}$, or the relative size of $f'(\mathbf{x}_{k+1})^T f(\mathbf{x}_{k+1})$ is less than $macheps^{\frac{1}{3}}$ and unsuccessfully if the iteration limit is exceeded.

### 2.6 LANCELOT

LANCELOT is one of the solvers available on the NEOS Web-based environment [5,16]. The NEOS environment is a high speed, socket-based interface for UNIX workstations that provide easy access to all the optimization solvers available on the NEOS Server. This tool allows users to submit problems to the NEOS Server directly from their local networks. Results are displayed on the screen. LANCELOT is a standard Fortran 77 package for solving large-scale nonlinearly constrained optimization problems. The areas covered by Release A of the package are: unconstrained optimization problems, constrained optimization problems, the solution of systems of nonlinear equations, and nonlinear least-squares problems.

The software combines a trust region approach adapted to handle the bound constraints, projected gradient techniques, and special data structures to exploit the (group partially separable) structure of the underlying problem. It additionally provides direct and iterative linear-solvers (for *Newton* equations), a variety of preconditioning and scaling algorithms for more difficult problems, quasi-*Newton* and *Newton* methods, provision for analytical and finite-difference gradients.

## 3 Test Cases for Systems of Nonlinear Algebraic Equations

In previous studies we were able to classify several of the test problems as easily solvable by all methods. These included the Rosenbrock's, Discrete Boundary Value, Broyden Tridiagonal, Broyden Banded and the Freudenstein-Roth functions [17]. Since the fact that a solver is capable of solving them introduces no new information we have decided to remove them from further considerations. In our search for test problems we have come across problems of least squares type as well as constrained and unconstrained optimization. We have decided to concentrate out attention strictly on systems of nonlinear algebraic equations and Table 1 contains the list of test problems used in our work.

**Table 1.** Test problems

| | |
|---|---|
| 1.Powell singular function [17] | 10. Variably dimensioned function[17] |
| 2.Powell badly scaled function [17] | 11. Exponential/Sine Function[22] |
| 3. Wood function [17] | 12. Semiconductor Boundary Condition[22] |
| 4. Helical valley function[17] | 13. Gulf Research and Development[17] |
| 5. Watson function[17] | 14. Extended Powell Singular[17] |
| 6. Chebyquad function[17] | 15. Extended Rosenbrock[17] |
| 7. Brown almost-linear function[17] | 16. Dennis, Gay and VU[17] |
| 8. Discrete integral equation [17] | 17. Matrix Square Root [17] |
| 9. Trigonometric function[17] | |

All codes are implemented in Fortran 77 and were run in double precision on a PC with a Pentium Pro 200 MHz processor. When applying the five solvers we have kept the default settings of all parameters as suggested in the implementation (which matches our assumption of the solver being treated like black-box software).

### 3.1 Simple Test Case

In this study we examined the 17 test problems summarized in Table 1 by studying the sensitivity of the starting vectors. For each problem we used the default vector, all ones, all zeros and random numbers as our initial starting vectors. We used the default number of equations for each problem, which ranged from 2 to 10 equations.

We were able to observe behavior patterns from the problems the were able to converge which helped us determine which solvers are more adapt for global convergence. The results for each problem were typical to that of problem 8, the Brown Almost-linear function. Figure 1 shows the number of iterations required for convergence for each of the various testing methods used on this problem. This problem shows that it is easy to converge with any solver as long as the



initial starting vector is in a certain range of the solution vector but once outside of that range, convergence did not occur for HYBRD, SOS, CONTIN, HOMPACK, and LANCELOT. We applied the same test to the other problems and found the pattern set by these problems to be consistent - solvability of the test problems depends on the solver and the starting vector except for TENSOLVE. It appears that TENSOLVE seems to be more robust and was able to achieve convergence regardless of the starting vectors.

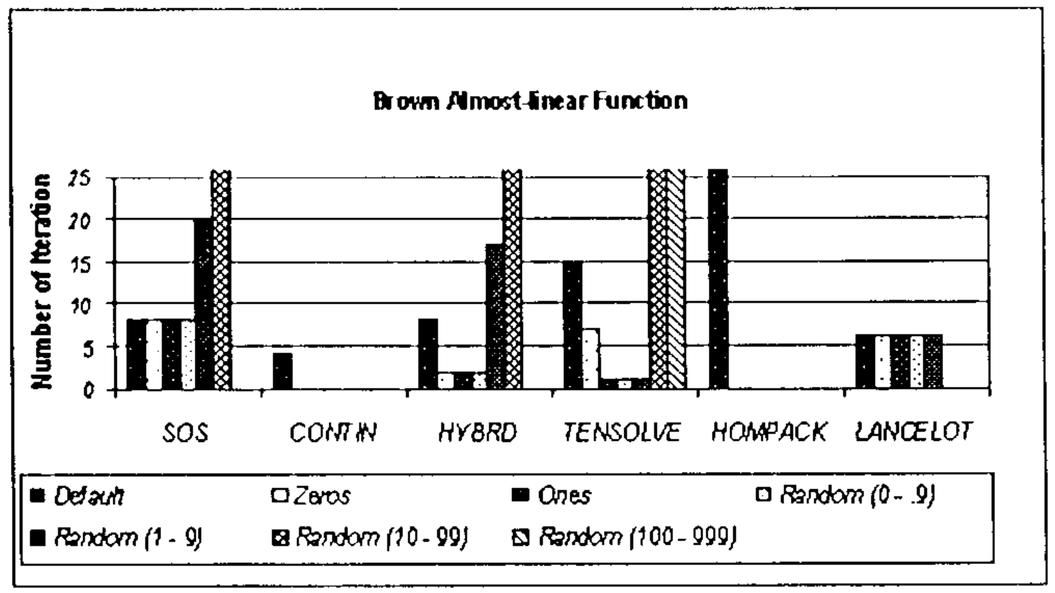


**Fig. 1.** Various Initial Starting Vectors for Problem 8

### 3.2 More Difficult Test Cases

We ran a set of test on all 17 problems. The starting vector were defined by adding percentages to a known solution in increments of 10%. For example, the solution set for Problem 8 is [1.0, . . . , 1.0] for $n=10$. We ran the problem 10 times using the initial values sets of ([1.1, . . . , 1.1], [1.2, . . . , 1.2],. . . , [2.0,. . . ,2.0]). Next we repeated the process subtracting percentages from a known solution in increments of 10%. We then recorded the point when there was non-convergence in the positive direction (adding percentages) and the negative direction (subtracting percentages). If there was always a convergence, we recorded the results as 100%, otherwise, we record the exact percentage away from the exact solution that non-convergence occurred. We then noted that the behavior was similar for all the test problems and the convergence rate above and below the exact solution for each problem per solver was less than 20% as shown in Figure 2.

The results are rather interesting as they show that, in the experimental setup used in our experiments; the *tensor* method outperforms the other solvers (with the *combination trust region/projected gradient* technique coming second and
*hybrid* method third) when looking at global convergence. The TENSOLVE, LANCELOT, HYBRD and SOS codes appear to have been better designed to handle various initial starting vectors.

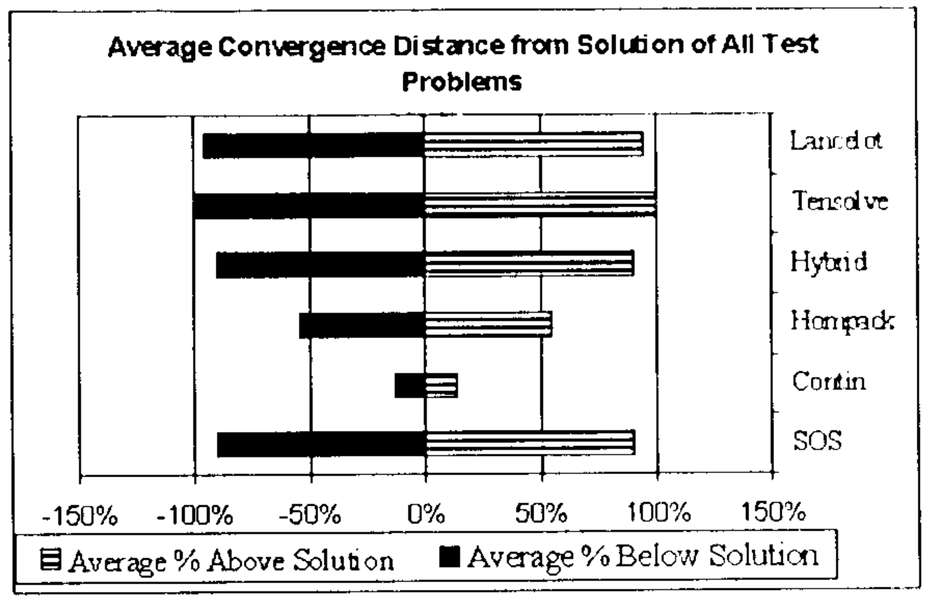


**Fig. 2.** Average Percentage Convergence in Dependence of Starting Vector of the Methods

## 4 Conclusions and Future Work

In this note we have briefly reported on our experiments analyzing the sensitivity of the initial starting vectors for standard test systems of up to 10 nonlinear algebraic equations solved by five advanced solvers. We have established a pattern with previous works and this set of experiments and has found that:

- solvability of the test problems depends on the solver and the starting vector,
- problems which are not solvable using one method may be solvable by another method, and
- of the solvers tested, the *tensor* method based solver appeared to be most robust.

Similar results were achieved for large number of equations in a previous publication [6].

Our future work will concentrate on expanding the *tensor* method based solver (as the most promising one) to handle very large systems. We will also continue our search for solvers that can handle medium to large systems of nonlinear algebraic equations as well as new interesting test problems that can be recommended to study the robustness of the nonlinear solvers. We will apply these solvers to the original avionics problem and observe their performance.

## References

1. Allgowerr, E. and George, K.: Numerical Continuation Methods: An Introduction, Springer-Verlag, Berlin (1990) 365
2. Brown, K. M.: A quadratically convergent Newton-like method based upon Gaussian elimination, SIAM Journal on Numerical Analysis, 6 (1969) 560-569
3. Bouaricha, A., Schnabel, R.: Algorithm 768: TENSOLVE: A Software Package For Solving Systems Of Nonlinear Equations And Nonlinear Least-Squares Problems Using Tensor Methods. ACM Trans. Math. Software, **23**, 2 (1997), 174-195



4. Burden, R. L., Faries, J. D.: Numerical Analysis. PWS-Kent Publishing Company, Boston, (1993) 575-576
5. Conn, A. R., Gould, N. I. M and Toint, Ph. L.: LANCELOT: A Fortran package for large-scale nonlinear optimization (Release A), Springer Series in Computiational Mathematics 17, Springer-Vergag, (1992)
6. Dent, D., Paprzycki, M., Kucaba-Pietal, A.: Comparing Solvers for Large Systems of Nonlinear Algebraic Equations. Proceedings of the 16th IMACS World Congress, to be published
7. Dent, D., Paprzycki, M., Kucaba-Pietal, A,: Performance of Solvers for Systems of Nonlinear Algebraic Equations. Proceedings of 15th Annual Conf. on Applied Math (1999) 67-77
8. Dent, D., Paprzycki, M., Kucaba-Pietal, A,: Studying the Numerical Properties of Solvers for Systems of Nonlinear Equations. Proceedings Of The Ninth International Colloquium On Differential Equations (1999), 113-118
9. Dent, D., Paprzycki, M., Kucaba-Pietal, A,: Testing Convergence of Nonlinear System Solvers. Proceedings of the First Southern Symposium on Computing, (1998)
10. Fong, K. W., Jefferson, T. H., Suyehiro, T.,Walton, L.: Guide to the SLATEC Common Mathematical Library, Argonne, Ill., (1990)
11. Kucaba-Pietal, A., Laudanski, L.:Modeling Stationary Gaussian Loads. Scientific Papers of Silesian Technical University, Mechanics, **121** (1995) 173-181
12. Laudanski, L.: Designing Random Vibration Tests. Int. J. Non-Linear Mechanics, **31,** 5 (1996) 563-572
13. More, J. J., Garbow, B. S., Hillstrom, K. E.: User Guide for MINPACK-1, Argonne National Laboratory Report ANL-80-74, Argonne, Ill., (1980)
14. More, J. J., Sorensen, D. C., Hillstrom, K. E., Garbow, B. S.: The MINPACK Project, in Sources and Development of Mathematical Software, W. J. Cowell, ed., Prentice-Hall (1984)
15. NEOS Guide (1996) http://www-fp.mcs.anl.gov/otc/Guide/
16. NEOS Solvers (2000) http://www-neos.mcs.anl.gov/neos/server-solvers.html
17. Netlib Repository (1999) http://www.netlib.org/liblist.html
18. Rheinboldt, W. C.: Methods for Solving System of Nonlinear Equations. SIAM, Philadelphia (1998)
19. Rheinboldt, W. C., Burkardt, J.: Algorithm 596: A Program For A Locally Parameterized Continuation Process. ACM Trans. Math. Software, **9** (1983) 236-241
20. Stoer, J., Bulirsh, R.: Introduction to Numerical Analysis. Springer, New York, (1993) 521
21. Watson, L. T., Sosonkina, M., Melville, R. C., Morgan, A. P., Walker, H. F.: Algorithm 777:HOMPACK 90: Suite Of Fortran 90 Codes For Globally Convergent Homotopy Algorithms. ACM Trans. Math. Software **23**, 4 (1997), 514 – 549
22. Weimann, U. N.: A Family of Newton Codes for Systems of Highly Nonlinear Equations. ZIB Technical Report TR-91-10, ZIB, Berlin, Germany (1991)