Proceedings of the Southern Conference on Computing
The University of Southern Mississippi, October 26-28, 2000

COMPARING SOLVERS FOR LARGE SYSTEMS OF NONLINEAR ALGEBRAIC EQUATIONS

Deborah Dent[*] Marcin Paprzycki[*] Anna Kucaba-Piętal[H]

Abstract. The solution of systems of nonlinear algebraic equations is a problem which complexity grows as the number of equations increases. The number of engineering problems, which led to these systems, is increasing each year. In this paper we compare the performance of four advanced solvers for systems of nonlinear algebraic equations which have been collected from Internet repositories. We are particularly interested in their performance when the number of equations is large (100+). We have collected test problems from literature and the Internet and found that most popular problems consist of systems of 2-4 equations but a few of the popular test problems are defined in such a way that they can be extended to a large system of nonlinear algebraic equations.

In the paper we report on results of our experiments in applying the nonlinear solvers to these test problems for an increasing number of equations. All of the solvers that we are using were designed to handle small and medium sized (up to 100) problems. Since the source code was available we were able to extend their functionality to handle problems with up to 200 equations and test the solvers' efficiency by comparing their convergence rates and sensitivity to starting vectors.

In the paper, we first describe the solvers and the algorithms on which the solvers are based. This is followed by the presentation of our experimental results. We end our discussion with conclusions and a description of future research directions.

1. Introduction.

The motivation for this work stems from a real-world avionics problem that requires a solution to a large system of nonlinear algebraic equations [7,8]. Due to the lack of convergence, algorithms applied there were unable to solve systems of more than 64 equations. At the same time it was estimated that, to model the engineering problem at hand, a system of about 500-1000 equations would have to be solved. This prompted us to search for existing software capable of solving our problem. In our earlier work [4,5,6], we have reported on our attempts at comparing the performance of non-commercial nonlinear system solvers based on *Newton's* method and its modifications, *bisection*, *continuation*, *hybrid*, *homotopy* and *tensor* methods were applied to 22 test problems collected from literature and the Internet. We have found that: (a) the simple algorithms (which work well in the case of a single nonlinear equation) like *bisection* or *Newton's* method and its modifications perform poorly when applied to systems of equations, (b) in-house developed simple implementations of known algorithms are only slightly less efficient than state of the art library codes. We have also found a few facts related to the test problems. (1) While some test problems are relatively popular, different researchers use different problems to test their algorithms and their implementations. (2) Typical test problems (in their default formulation) consist of systems of 2-4 equations and only very few reach 10 equations. In our literature and Internet search we have not located tests corresponding to the real-life engineering problems of 100+ equations (e.g. the avionics problem mentioned above). (3) Test problems involving non-smooth functions (e.g. arising in electrical engineering) are not represented in the test sets we encountered (however, since interest is directed toward solving large systems of equations, the non-smooth cases have been omitted from our considerations). (4) Out of the 22 popular test problems (in their default formulation) five are easily solvable by all solvers (including the simplest) and thus have no real value when used to test

[*] School of Mathematical Sciences, University of Southern Mississippi, Hattiesburg, MS 39406-5106, {deborah.dent, m.paprzycki}@usm.edu

[H] Department of Fluid Mechanics and Aerodynamics, Technical University of Rzeszow, Anpietal@ewa.prz.rzeszow.pl

algorithms. (5) Only nine of the 22 collected test problems can be used to generate large systems of nonlinear algebraic equations.

This note will report results of our experiments in applying four non-commercial advanced nonlinear solvers, to the latter group of test problems, for an increasing number of equations (up to 200). These solvers were applied to the nine test problems in a way that is to resemble an engineer approaching a solution to a real-life problem. It was thus assumed that the potential user is not a highly trained numerical analyst and/or programmer who is able and/or willing to invest time into studying intricacies of methods and their implementations (e.g. working toward finding a proper homotopy map, which is the necessary step to assure the most efficient work of the *homotopy* method based solver). Rather, we envision the user as someone who will be applying the codes as more or less black-box solvers, likely to follow the default settings provided by the solvers and aiming at finding a verifiable solution to the problem. (This assumption should be kept in mind when reading the remaining parts of this note.)

The solvers' efficiency as well as their numerical properties was assessed by comparing their convergence rates (measured in number of function evaluations and iterations) and sensitivity to the starting vectors. All four solvers were designed to handle small and medium sized (up to 100 equations) problems (and this observation applies all solvers that we have located). Since the source codes were available we were able to extend their functionality above this limit and attempt at solving problems with up to 200 equations.

This note is organized as follows. Section 2 briefly describes the solvers and the algorithms on which the solvers are based. In section 3, we summarize the results of numerical experiments. The paper concludes with a description of future research directions.

2. Solvers and Algorithms for Systems of Nonlinear Algebraic Equations. As mentioned above, 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 have thus excluded codes based on *Newton's* method and its modification (e.g. Brown's method) and *bisection* from further considerations. This left us non-commercial versions of codes based on the *hybrid* algorithm, *homotopy*, *continuation*, and *tensor* methods. These algorithms are all documented in ACM TOMS and their implementations were obtained from the NETLIB repository [11]. We have thus modified (to handle up to 200 equations) the following software packages:

- HYBRD, a combination of *trust region* and *Powell's (modified Newton)* method,
- CONTIN, an implementation of the *continuation* method,
- HOMPACK, an implementation of the *homotopy* method,
- TENSOLVE, an implementation of the t*ensor* method combined with the *trust region* and *line-search* options.

We will now briefly summarize these algorithms and the solvers (in all cases the references cited and [13] 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. HYBRD1's design is based on a combination of a modified *Newton* method [12] and the *trust region* method [10]. 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 CONTIN. CONTIN [10], also know as PITCON [14] implements a continuation algorithm with an adaptive choice of a local coordinate. The *continuation* method is designed to be able to target more complicated problems and is the subject of various research efforts [1,15,10]. This method is expected to be slower than *line-search* 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 λ into the problem and defining a function

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

where $\mathbf{x}_0\in$ |ⁿ. The problem $h(\mathbf{x},\lambda)=\mathbf{0}$ is then solved for values of λ 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 convergence rate of the *continuation* methods varies, but the method does not require a good choice of the initial vector $\mathbf{x}_0$.

2.3 HOMPACK. HOMPACK [17,18] is a suite of subroutines for solving nonlinear systems of equations by *homotopy* methods [3,18]. 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 λ assuming values in [0,1]. 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, normal flow, and quasi Newton augmented Jacobian matrix. The original problem corresponds to $\lambda=1$ and a problem with a known solution corresponds to $\lambda=0$. For example, the set of problems

(2.2) $$G(\mathbf{x},\lambda) = f(\mathbf{x}) + (1-\lambda) f(\mathbf{x}_0) = \mathbf{0}, \;\; 0 \le \lambda \le 1,$$

for fixed $\mathbf{x}_0\in$ |ⁿ forms a homotopy. When $\lambda=0$, the solution is $\mathbf{x}(\lambda=1)$. Similarly to the *continuation* method, the vector but proper implementation of this method involves defining the homotopy $h(\mathbf{z},t)$ and finding a numerical method for tracking the paths defined by $h(\mathbf{z},t)=\mathbf{0}$.

The *homotopy* method is carried out via three qualitatively different algorithms: ODE-based (code FIXPDF), normal flow (code FIXPNF), and augmented Jacobian (code FIXPQF). The code is available in both Fortran 77 [18] and Fortran 90 [17]. 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.

2.4 TENSOLVE. TENSOLVE [2] 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. These two methods are described in the following sections. The stopping criteria is meet when the relative size of $\mathbf{x}_{k+1} - \mathbf{x}_k$ is less than the $macheps^{2/3}$, or $||F(\mathbf{x}_{k+1})||_\infty$ is less than $macheps^{2/3}$, or the relative size of $J(\mathbf{x}_{k+1})^{TF}(\mathbf{x}_{k+1})$ is less than $macheps^{1/3}$ and unsuccessfully if the iteration limit is exceeded.

3. Experiments and Results. All codes are implemented in Fortran 77 and were run in double precision on a PC with a Pentium Pro 200 MHz processor. As specified above, out of the test problems collected in [9,19] we have found only nine for which the number of equations *n* could be increased to generate large systems of equations: (1) Watson function, (2) Chebyquad function, (3) Brown almost-linear function, (4) Discrete boundary value function, (5) Discrete integral equation function, (6) Trigonometric function, (7) Variably dimensioned function, (8) Broyden tridiagonal function, (9) Broyden banded function. When applying the four 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).

In our experiments we have used four different starting vectors (which correspond to what an engineer could try to use in cases where the solution is unknown). First, for each of the test problems (as they were described in the literature) a starting vector is provided as a part of their definition and we have utilized this data (results denoted as ***default***). We first attempted to expand the size of each problem to a maximum of 200 equations. Depending on the solver, for the ***default*** starting vector, some of the problems could not be solved beyond a specific numbers of equations and this fact is illustrated in Table 1 where the largest number of equations for which the solution was found is reported.

Table 1. Maximum Number of Equations Solvable by Each Solver for the ***default*** Starting Values

Problem # **Solver**	**#1**	**#2**	**#3**	**#4**	**#5**	**#6**	**#7**	**#8**	**#9**
CONTIN	6	2	200	11	4	23	4	200	200
HYBRD	14	9	22	200	200	40	42	200	200
HOMPACK	9	9	9	9	100	1	1	100	100
TENSOLVE	30	18	200	200	200	200	200	200	200

The results are rather interesting as they show that, in the experimental setup used in our experiments (codes used as black-box software and default parameter setup is applied); the *tensor* method outperforms the other solvers (with the *hybrid* method coming second). This result illustrates the overall relative weakness of the *continuation* method and the need for a more hands-on approach when the *homotopy* method based solver is applied. In addition, it can be easily observed that problems #8 and #9 are the easiest to solve. Thus their usability as the test cases is limited not only when they are used in their default setup [4,5,6] but also when the number of equations is large. The results indicate that, out of the nine tested, problems #1 and #2 appear to be most difficult to solve.

We next wanted to observe the behavior of the solvers when the number of equations (in a given problem) is increasing. Since only problems #8 and #9 were "solvable" by most solvers for up to n = 200 equations we have decided to use them in our experiment, and the results for the n = 50, 100, 150 and 200 equations are displayed in Table 2. We recorded the number of function calls (FC) and the number of iterations (IT) for each solver. (Usually, solution time is important in analyzing performance, but due to fast processors the time to complete the task for all of the problems was to small to record, so we had to omit it). We were unsuccessful in modifying HOMPACK to handle more than 100 equations; therefore, the results of HOMPACK for n=150 and 200 were reported as N/A.

Table 2. Comparison of Performance of Four Solvers;
Problems #8 and #9; n = 50, 100, …, 200 Equations

		***n*=50**		***n*=100**		***n*=150**		***n*=200**	
Problem #	**Solvers**	**FC**	**IT**	**FC**	**IT**	**FC**	**IT**	**FC**	**IT**
#8	CONTIN	626	4	1226	4	1826	4	2426	4
	HYBRD1	61	11	111	11	161	11	211	11
	TENSOLVE	204	3	404	3	604	3	804	3
	HOMPACK	215	104	355	172	N/A	N/A	N/A	N/A
#9	CONTIN	886	4	1736	4	2586	4	3436	4
	HYBRD1	69	19	119	19	169	19	219	19
	TENSOLVE	255	4	505	4	755	4	1005	4
	HOMPACK	256	123	424	205	N/A	N/A	N/A	N/A

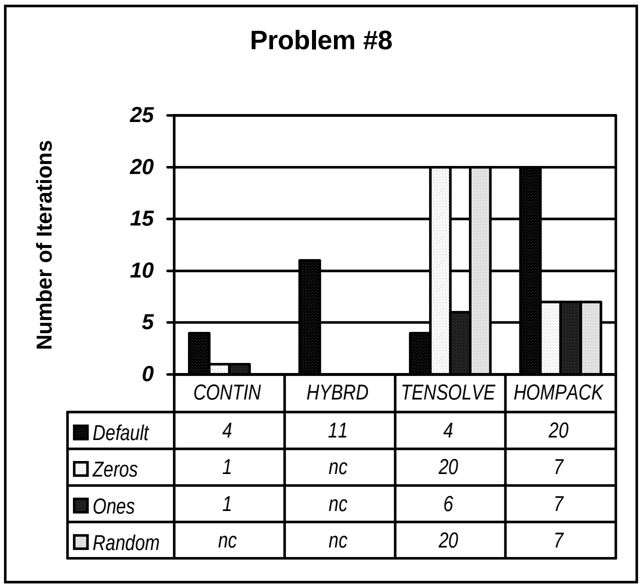
We can observe that as the number of equations increases the number of function evaluations increases as well, but the number of iterations remains constant. While this effect is clearly problem dependent, in our experiments we have found that out of the nine problems studied, only for problem #1 as the number of equations increased the number of iterations increased as well. Since this problem is solvable only for up to n = 7 equations, in Table 3 we report the performance of the four solvers for n = 1, 2, …, 7 and for the ***default*** starting vector. Here, it can be observed that the number of iterations can fluctuate (does not always increase steadily) i.e. observe performance of TENSOLVE for n = 5, 6, and

7. Note also, that HOMPACK did not converge for *n* = 4 equations. We observed a similar effect sporadically also in other experiments.

Table 3. Comparison of Performance of Four Solvers; Problem #1; n = 1, 2, …, 7 Equations

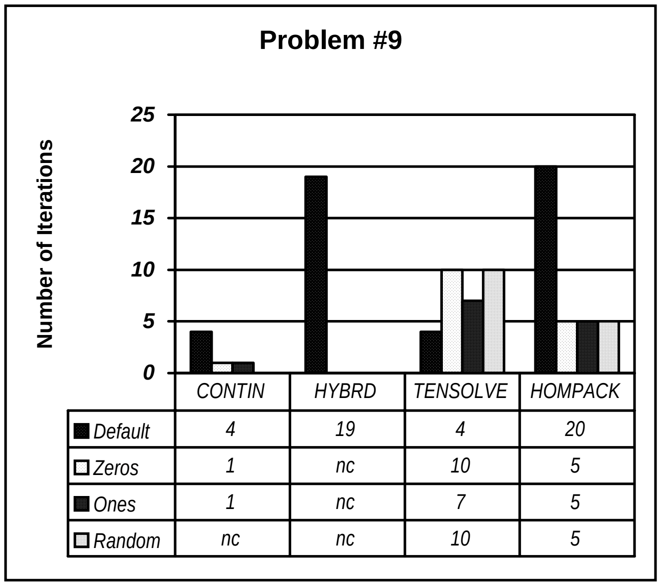
		***n*=2**	***n*=3**	***n*=4**	***n*=5**	***n*=6**	***n*=7**
Problem #	**Solvers**	**IT**	**IT**	**IT**	**IT**	**IT**	**IT**
#1	CONTIN	5	5	5	5	6	8
	HYBRD1	12	15	19	29	38	53
	TENSOLVE	4	6	10	23	22	42
	HOMPACK	39	40	*nc*	63	97	141

In the final series of experiments we have observed the effect of various starting vectors on the performance of the four solvers. Here we have tried a few "obvious" selections for starting vectors: vectors of all zeroes (denoted ***zeroes***), vector of ones (denoted ***ones***) and vectors of random numbers from the interval [0, 1] (denoted ***random***) (in addition to the ***default*** values specified above). Figures 1 and 2 show the fluctuation in the resulting number of iterations required for convergence when the different starting vectors were applied to problems #8 and #9 for *n* = 100 equations. In both figures lack of convergence denoted is by *nc* (and the appropriate bar is removed from the graph). We can easily see that CONTIN worked better with the initial values set equal to ***ones*** and ***zeroes*** and did not respond to ***random*** at all (we have no explanation of this behavior). We can also observe that HYBRD is very sensitive to the starting vector and does not respond well to starting vectors other than ***default***. TENSOLVE converged with all attempts but responded best to the ***default*** starting vector. HOMPACK responded better to vectors other than ***default***, which coincides with the authors' comments in [16] that the ***default*** starting vectors defined for these test sets were not well suited for HOMPACK.



	CONTIN	HYBRD	TENSOLVE	HOMPACK
■ *Default*	*4*	*11*	*4*	*20*
□ *Zeros*	*1*	*nc*	*20*	*7*
■ *Ones*	*1*	*nc*	*6*	*7*
□ *Random*	*nc*	*nc*	*20*	*7*

Figure 1. Sensitivity Analysis of Problem #8



	CONTIN	HYBRD	TENSOLVE	HOMPACK
■ *Default*	*4*	*19*	*4*	*20*
□ *Zeros*	*1*	*nc*	*10*	*5*
■ *Ones*	*1*	*nc*	*7*	*5*
□ *Random*	*nc*	*nc*	*10*	*5*

Figure 2. Sensitivity Analysis of Problem #9

4. Conclusions and Future Work. In this paper we have reported on our experiments comparing performance of solvers for large (up to $n = 200$) systems of nonlinear algebraic equations. We have found that:

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

For the collected test problems we have found that:

1) problems #8 and #9 are relatively easy and thus do not provide useful information when used to compare the solver performance,
2) the remaining problems are hard and neither of the solvers seems to be ready to easily solve them; especially as the number of equations increases. In particular, problems #1 and #2 seems to be very hard to solve and thus can be recommended as real benchmarks for the robustness of new solvers that will be developed in the future.

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] E. Allgowerr and K. George, *Numerical Continuation Methods*, *An Introduction*, Springer-Verlag, Berlin, 1990, p.365.
[2] A. Bouaricha and R. Schnabel, *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), pp. 174-195.
[3] R. L. Burden and J. D. Faries, Numerical Analysis, PWS-Kent Publishing Company, Boston, 1993, pp. 575-576.
[4] D. Dent, M. Paprzycki, and A. Kucaba-Pietal, *Performance of Solvers for Systems of Nonlinear Algebraic Equations*, Proceedings of 15th Annual Conf. on Applied Math (1999), pp. 67-77.
[5] D. Dent, M. Paprzycki, and A. Kucaba-Pietal, *Studying The Numerical Properties of Solvers for Systems of Nonlinear Equations*, Proceedings of the Ninth International Colloquium on Differential Equations (1999), pp. 113-118.
[6] D. Dent, M. Paprzycki, and A. Kucaba-Pietal, *Testing Convergence of Nonlinear System Solvers*, FSSC, 1 (1999), http://pax.st.usm.edu/cmi/fscc98_html/processed/.
[7] A. Kucaba-Pietal and L. Laudanski, L., *Modeling Stationary Gaussian Loads*, Scientific Papers of Silesian Technical University, Mechanics, 121 (1995), pp. 173-181.
[8] L. Laudanski, *Designing Random Vibration Tests*, Int. J. Non-Linear Mechanics, 31, 5 (1996), pp. 563-572.
[9] J. J. More, B. S. Garbow and K. E. Hillstrom, *Algorithm 566*, ACM Trans, Math. Software, 20, 3 (1994), 282-285.
[10] _______, NEOS Guide, http://www-fp.mcs.anl.gov/otc/Guide/, 1996.
[11] _______, Netlib Repository, http://www.netlib.org/liblist.html, 1999.
[12] M. J. D. Powell, *A Hybrid Method for Nonlinear Algebraic Equations, in Polish*. Gordon and Breach, Rabinowitz, 1979.
[13] W. C. Rheinboldt, *Methods for Solving System of Nonlinear Equations*, SIAM, Philadelphia, 1998.
[14] W. C. Rheinboldt and J. Burkardt, *Algorithm 596: A Program for A Locally Parametrized Continuation Process*, ACM Trans. Math. Software, 9 (1983), pp. 236-241.
[15] J. Stoer and R. Bulirsh, *Introduction to Numerical Analysis*, Springer, New York, 1993, p. 521.
[16] L. T. Watson, personal communication.
[17] L. T. Watson, M. Sosonkina, R. C. Melville, A. P. Morgan and H. F. Walker, *Algorithm 777:HOMPACK 90: Suite of Fortran 90 Codes for Globally Convergent Homotopy Algorithms*, ACM Trans. Math. Software 23, 4 (1997), pp. 514 – 549.
[18] L. T. Watson, S. C. Billups and A. P. Morgan, *Algorithm 652:HOMPACK: A Suite of Codes for Globally Convergent Homotopy Algorithms*, ACM Trans. Math. Software 13 (1987), pp. 281 – 310.
[19] U. N. Weimann, *A Family of Newton Codes for Systems of Highly Nonlinear Equations*, ZIB Technical Report TR-91-10, ZIB, Berlin, Germany, 1991.