Studying the Performance Nonlinear Systems Solvers Applied to the Random Vibration Test

Deborah Dent, Marcin Paprzycki[1] and Anna Kucaba-Piętal, Ludomir Laudański[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, ul. W.Pola 2, Rzeszów, Poland

Abstract. In this paper we compare the performance of four solvers for systems of nonlinear algebraic equations applied to the random vibration test, which requires a solution of a system of 512 or more equations. Experimental results obtained for two test cases are presented and discussed.

1 Introduction

In the early 1990's, A. Kucaba-Piętal and L. Laudański studied digital simulation of samples of stationary Gaussian stochastic processes possessing multi modal spectra applied to dynamic loads arising in an airplane in gusty flying conditions [8]. In this problem, the numerical solution of the equations describing the disturbed motion of an elastic airframe result in a detailed description of the vertical displacement of any point chosen on the airplane under attention. Obtained results can also be transformed into a time history of stresses at the same place. There exist two possible approaches to this problem. Method of harmonics, developed by S. O. Rice and method of filtration developed by N. Wiener and, independently, Y. A. Kchinchin [8]. In the original study, the method of filtration was used and the impulsive characteristics of a nonrecursive filter $h(t)$ was related to the correlation function $K(\tau)$ of the output stochastic process $\{y(t)\}$ obtained via filtering of the input stochastic process $\{x(t)\}$. The resulting equation had the following form:

$$K(\tau) = E[\sum_{k=0}^{N} h(k)x(t-k)\sum_{n=0}^{N} h(n)x(t+\tau-n)]. \qquad (1)$$

Due to the fact that $\{x(t)\}$ is a white noise the problem was simplified to a system of nonlinear algebraic equations:

$$K(j) = \sum_{i=0}^{N} h(i)h(j) \ for \ j = 0, 1, ..., N, \qquad (2)$$

and expanded into its explicit form:

$$\begin{aligned}
K(0) &= h(0)h(0) + h(1)h(1) + h(2)h(2) + \ldots + h(N)h(N) \\
K(1) &= h(0)h(1) + h(1)h(2) + h(2)h(3) + \ldots + h(N-1)h(N) \\
K(2) &= h(0)h(2) + h(1)h(3) + h(2)h(4) + \ldots + h(N-2)h(N) \\
&\vdots \\
K(N-2) &= h(0)h(N-2) + h(1)h(N-1) + h(2)h(N) \\
K(N-1) &= h(0)h(N-1) + h(1)h(N) \\
K(N) &= h(0)h(N)
\end{aligned} \tag{3}$$

In 1994, the authors were able to solve this system for up to 64 equations (with the solution time of approximately 10 minutes on a desktop PC), while it was estimated that a minimum of 512 equations would be required, with a quality solution available for 1024 or more equations. The main reason for this relative failure (in addition to the weakness of computer hardware available at this time in Poland) was the fact that the proposed solution methods were not robust enough to handle the problem.

This earlier research has prompted us to investigate the state of the art in the area of solvers for nonlinear algebraic equations. The results of these investigations (which started in 1998) have been summarized in a number of papers (for more details see [4] and references quoted there). After completing the assessment stage we decided to go back to the original problem and see if we can solve it for the appropriately large system sizes by using the more sophisticated solution methods.

In this paper we summarize the results obtained so far. Section 2 contains a brief description of the preliminary work that lead us to the selection of four solvers and a summary of their functionalities. In Section 3 we describe the experimental data. Finally, in Section 5 we summarize our results and sktech future research directions.

2 Solver selection and descriptions

2.1 Preliminary work

We have investigated a number of algorithms and software packages designed to solve systems of nonlinear algebraic equations. In Table 1 we list the algorithms tested in our search to find “the best possible” solver.

Table 1: Algorithms for solution of systems of nonlinear algebraic equations

1. *Augmented Lagrangian* method	7. *Line Search* method
2. *Brown's* method	8. *Powell's* method
3. *Broyden's* method	9. *Reduced-gradient* method

Table 1: Algorithms for solution of systems of nonlinear algebraic equations

4. *Characteristic Bisection* method	10. *Tensor* method
5. *Continuation* method	11. *Trust Region* method
6. *Homotopy* method	

Algorithms listed in Table 1 have appeared as stand-alone, or as combinations, in a number of shareware packages found, among others, in the NETLIB repository and in the ACM TOMS. The list of solvers and algorithms constituting them is presented in Table 2.

Table 2: Software packages for solving systems of nonlinear algebraic equations

Package	**Algorithms**
1.CHABIS [13]	*Characteristic Bisection* method
2. CONTIN [11]	*Continuation* method
3. HOMPACK [12]	*Homotopy* method
4. LANCELOT [2]	*Augmented Lagrangian* method
5. MINOS [9]	*Reduced-Gradient* method
6. MINPACK'S HYBRID [10]	*Trust Region, Broyden's*, and *Powell's* methods
7. SLATEC's SOS [5]	*Brown's* method
8. TENSOLVE [1]	*Tensor*, *Trust Region* and *Line Search* methods

In previous papers, we have reported on the results of testing solvers listed in Table 2 (as well as additional, in house developed codes) on a set of 22 standard test problems [3]. Unfortunately we were not able to locate a single "best" solver. However, we were able to remove some solvers from further consideration and reduced the field to codes based on *hybrid Newton, homotopy, continuation, tensor, augmented Lagrangian* and *reduced-gradient* methods. Implementations of these methods were obtained from the NETLIB repository [15] and the NEOS server [14].

2.2 Test Problem I

In the original study [7] an artificial sample problem was developed with an integer answer-set and we have decided to use it as a starting point. We have taken $h(i) = i,\ for\ i = 1..N$ and substituted it to (3) and calculated the values of K to generate our Test Problem I. When attempting at a solution we have experimented with a number of possible starting vectors (including zero, one and random numbers) and found that the convergence was reached most often when the starting vector $h_0(i) = 1\ for\ i = 1..N$ was used. The aim of Test Problem I was to select the solvers for our final testing suite that would allow us to solve larger problems. We have found that only the following four solvers converged for more than $N = 64$ equations: HYBRID, TENSOLVE, LANCELOT, and MINOS.

It should be noted that the remaining solvers listed in Table 2 have also been tried for this and for the real-world data (see Test Problem II below) and the results were similar (no convergence beyond $N = 64$). We will now briefly describe the selected four solvers. We assume that a system of N nonlinear algebraic equations $f(x) = 0$ is to be solved where $\boldsymbol{x}$ is N-dimensional vector and $\mathbf{0}$ is the zero vector.

2.3 Solvers

HYBIRD is part of the MINPACK-1 suite of codes. HYBRID's design is based on a combination of a modified *Newton* method [10] and the *trust region* method [14]. 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).

TENSOLVE [1] 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.

LANCELOT (Large And Nonlinear Constrained Extended Lagrangian Optimization Techniques) is a package of standard Fortran subroutines and utilities for solving large-scale nonlinearly constrained optimization problems [2]. The LANCELOT package uses an *augmented Lagrangian* approach to handle all constraints other than simple bounds. The bounds are dealt with explicitly at the level of an outer-iteration sub-problem, where a bound-constrained nonlinear optimization problem is approximately solved at each iteration.

The algorithm for solving the bounded problem 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. The software 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, and an automatic decoder capable of reading problems expressed in Standard Input Format (SIF) or a Modeling Language for Mathematical Programming (AMPL). For our experiments, the Web-based NEOS version of this software was used. Each nonlinear problem was converted to an AMPL minimization problem.

MINOS [9] is a software package for solving large-scale optimization problems (linear and nonlinear programs). It is especially effective for linear programs and for problems with a nonlinear objective function and sparse linear constraints (e.g., quadratic programs). MINOS can also process large numbers of nonlinear constraints. The nonlinear functions should be smooth but need not be convex. For linear programs, MINOS uses a sparse implementation of the primal simplex method. For nonlinear objective functions (and linear constraints), MINOS uses a *reduced-gradient* method with quasi-Newton approximations to the reduced Hessian. For problems with nonlinear constraints, MINOS uses a sparse SLC algorithm (a projected Lagrangian method). It solves a sequence of sub-problems in which the constraints are linearized and the objective is an *augmented Lagrangian* (involving all nonlinear functions). Convergence is rapid near a solution.

MINOS makes use of nonlinear function and gradient values. The solution obtained will be a local optimum (which may or may not be a global optimum). If some of the gradients are unknown, they will be estimated by finite differences. If the linear constraints have no feasible solution, MINOS terminates as soon as infeasibility is confirmed. Infeasible nonlinear constraints are difficult to diagnose as with LANCELOT, the Web-based NEOS version of this software was used with AMPL input.

3 Experimental Results

3.1 Test Problem I

The results from Test Problem I are summarized in Table 3. Here, N denotes the number of equations, IC - the number of iterations required for convergence, FC – the number of function calls required for convergence, and *time/sec* – the number of CPU seconds used (on a 900 MHz Pentium III workstation) and NC - represents non-convergence.

Table 3: Results for the Test Problem I

	Hybrid			*Lancelot*		
N	*IC*	*FC*	*time/Sec*	*IC*	*FC*	*time/sec*
128	10	110	1	49	50	34.68
256	10	160	2	161	162	905.4
512	10	210	3	*NC*	-	-
1024	10	310	12	*NC*	-	-
	Minos			*Tensolve*		
N	*IC*	*FC*	*time/Sec*	*IC*	*FC*	*time/sec*
128	1977	4182	19.2	602	1367	2.44
256	7340	14771	564.07	1216	2646	11.75
512	24205	47580	8346.04	4591	9422	171.02
1024	*NC*	-	-	*NC*	-	-

Comparing the performance of the four solvers we can observe that:

- HYBRID converges for the largest number of equations, is the fastest, and is very accurate.
- LANCELOT converges only for up to $N = 256$ and is the slowest.
- MINOS converges for up to $N = 512$ and produces a different solution than the remaining three solvers.
- TENSOLVE converges for up to $N = 512$ and is not as accurate as LANCELOT and HYBRID.

As noted, HYBRID, LANCELOT, and TENSOLVE produced the same basic solution while MINOS produced an alternate solution. This result appears consistently since $N = 4$, where the three solvers produced the expected integer answer while MINOS produced a different result (see also [7]).

3.2 Test Problem II

The aim of Test Problem II was to apply real-world data requiring up to 512 equations to the solvers selected from Test Problem I. For this case the coefficient vector, $\boldsymbol{K(\tau)}$, consisted of the floating-point correlation data. For the starting vector we used $\boldsymbol{h0(i)} = \mathbf{1}$ (we have tried, again, a number of possible starting vectors and found that one most often results in convergence). The results from these tests are summarized in Table 4 (the meaning of all symbols is the same as in Table 3, above). For each of the calculated results we have calculated also the component-wise error. In Table 4 we depict the minimum and maximum component-error for the solution obtained by each of the solvers. It should be noted that HYBRID converged **only** when the solution vector calculated by TENSOLVE was used as its starting vector and this fact needs to be kept in mind while looking at the data.

Table 4: Results for the Test Problem II

	Hybrid			*Lancelot*		
N	*IC*	*FC*	*time/sec*	*IC*	*FC*	*time/sec*
128	10	110	1	161	162	111.5
256	10	160	2	164	165	633.09
512	21	2095	90	201	206	5140.05
	Hybrid			*Tenslove*		
N	*IC*	*FC*	*time/sec*	*IC*	*FC*	*time/sec*
128	1068	2737	19.22	28	3818	4
256	1383	3881	131.52	16	4381	20
512	2310	6508	1068.88	12	6692	117

Before we proceed with summarizing our observations let us note that we have obtained **three** separate answers. For obvious reasons HYBRID and TENSOLVE produced a similar solution vectors, however both MINOS and LANCELOT produced alternate solutions. We illustrate this in Table 5, where the initial and final components of solution vectors produced by all four solvers are presented.

Table 5: Partial View of Final Solution Vectors for $N = 512$

	LANCELOT	**MINOS**	**TENSOLVE**	**HYBRID**
i	***x(i)***	***x(i)***	***x(i)***	***x(i)***
1	-0.0596631	0.141907	-0.135745775	-0.137628574
2	-0.0336641	0.072495	-0.080370211	-0.079030645
3	-0.0176408	0.077313	-0.074520022	-0.075754926

Results presented in Tables 4 and 5 lead us to the following observations:

– HYBRID is the "best" solver for the Test Problem I but cannot handle the Test Problem II, but it can be used to improve accuracy of results produced by TENSOLVE.
– LANCELOT is the slowest of the three converging solvers.
– MINOS is faster than LANCELOT and more accurate than TENSOLVE.
– TENSOLVE produces an "approximate solution" and is the fastest of the three converging solvers.

4 Concluding Remarks

In this paper we have reported on our attempts at solution of an avionics-engineering problem. Using modern robust solvers we were able to solve a system of $N = 512$ nonlinear algebraic equations. As previously, we have found no “silver bullet” but rather, that the solution depends on the interplay between the problem, the solver and the starting vector. Here, even in the case of the same problem with different “right hand sides” the behavior of individual solvers can be very different; as illustrated by the HYBRID solver applied to Test Problems I and II.

The most interesting result seems to be that, in the case of Test Problem II, all three globally convergent solvers have produced different solution vectors. At this time we do not have an answer which of them (if any) has physical interpretation (or, maybe, if all of them represent physically feasible solutions). This lack of answer(s) should be viewed in the context of the avionics-engineering problem itself. Finding a solution to this problem consists of solving two sub-problems: (a) finding solutions to the system of nonlinear algebraic equations, and (b) interpretation of the results in terms of the physics of flight. In this note we have addressed the first sub-problem in a positive way. Addressing the second sub-problem is outside of the scope of this note, but will be included in the next step our research and we hope to be able to report on it shortly.

In addition to looking for physical interpretation of the results obtained from the numerical study, we have a few more items that we will investigate. First, we plan to solve the system of $N = 1024$ equations. Second, we plan to look into the ways in which the three solvers find their answers (since all three of them start from the same vector, it is interesting to see how they arrive into three separate answers). Third, we will apply to our problem commercial solvers from NAG and Visual Numerics libraries. Finally, we will apply to it a solver based on the interval approach (INTLIB) [6] either as a solver in its own right, or, as a verification tool for solutions located by other solvers.

References

1. Bouaricha, A. and 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
2. Conn, A.R., Gould, N.I.M. and Toint, P.L.: LANCELOT: A FORTRAN Package for Large-Scale Nonlinear Optimization (Release A), Springer-Verlag, New York, (1992)
3. Dent, D., Paprzycki, M. and Kucaba-Pietal, A.: Testing Convergence of Nonlinear System Solvers, Proceedings of the First Southern Symposium on Computing, Hattiesburg, MS, CD, (1999) file Dent-etal.ps
4. Dent, D., Paprzycki, M. and Kucaba-Pietal, A.: Solvers for systems of nonlinear algebraic equations - their sensitivity to starting vectors, Numerical Analysis and Its Applications, Second International Conference, NAA 2000, Rousse, Bulgaria, Spinger Verlag, (2000) 230-238
5. Fong, K. W., Jefferson,T.H. and T. Suyehiro, T.: Guide to the SLATEC Common Mathematical Library , Technical Report Energy Science and Technology Software Center, Oak Ridge, TN (1993)
6. Kearfott, R.B. and M, Novoa III, M.: Algorithm 681 INTLIB, a Portable Interval Newton/Bisection Package, ACM Transactions on Mathematical Software, **16**, 2, (1990) 152-157
7. Kucaba-Pietal, A. and Laudanski, L.: Modeling stationary Gaussian loads, Scientific Papers of Silesian Technical University, Mechanics, **121**, (1995) 173-181
8. Laudanski, L.: Designing random vibration tests, Int. J. Non-Linear Mechanics, **31**, 5, (1996) 563-572
9. Murtagh, B.A. and Saunders, M.A.: MINOS 5.4 USER'S GUIDE. Technical Report SOL83-20R, Department of Operations Research, Standford University, Standford, California 94305, USA, 1993, (Revised 1995)
10. Powell, M.J.D.: A hybrid method for nonlinear algebraic equations. Gordon and Breach, Rabinowitz, (1979)
11. Rheinboldt, W.C. and Burkardt, J.:, Algorithm 596: A program for a locally parameterized continuation process, ACM Trans. Math. Software, **9**, (1983) 236-241
12. Watson, L.T., Sosonkina, M., Melville, R.C., Morgan, A.P. and 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
13. VRAHATIS, M.N.: Algorithm 666: CHABIS: A Mathematical Software Package Systems of Nonlinear Equations, ACM Transactions on Mathematical Software, **14**, 4, (1988) 312-329
14. NEOS Guide, http://www-fp.mcs.anl.gov/otc/Guide/ (1996)
15. Netlib Repository, http://www.netlib.org/liblist.html (1999)