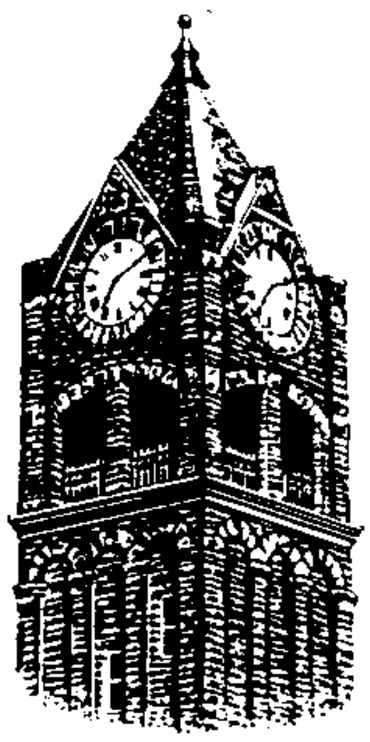
PROCEEDINGS OF THE FIFTEENTH ANNUAL CONFERENCE ON APPLIED MATHEMATICS



UNIVERSITY OF CENTRAL OKLAHOMA

University of Central Oklahoma
Edmond, Oklahoma
February 12-13, 1999

PERFORMANCE OF SOLVERS FOR SYSTEMS OF NONLINEAR ALGEBRAIC EQUATIONS

Deborah Dent and Marcin Paprzycki
School of Mathematical Sciences
University of Southern Mississippi
Hattiesburg, MS 39406-5106
{deborah.dent, m.paprzycki}@usm.edu

Anna Kucaba-Pietal
Department of Fluid Mechanics and Aerodynamics
Technical University of Rzeszów
anpietal@ewa.prz.rzeszow.pl

Abstract

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 rising each year. In the paper we attempt a comparison between existing solvers applied to a number of popular test problems. These problems have been used by various authors to study the performance of their nonlinear solvers and have been collected from a number of papers and from the Internet. We pursue the comparisons with two goals in mind. First, to establish the quality of particular solvers, we are interested in the speed of convergence, as well as the selection of the starting vector. Second, we use the experimental data to assess the usefulness of test problems themselves. Work discussed here represents a step in the direction of development of a unified library of test problems to be used when new methods for the solution of nonlinear systems are proposed.

1. INTRODUCTION

The aim of this note is to report on the progress of our work in the direction of building a library of nonlinear solvers combined with a set of test problems. We are utilizing a software engineering approach and focus on the software development with reuse. This study effort resulted from a need to solve a real world problem originating from avionics (Kucaba-Pietal 1995, Laudanski 1996). In this case, a large system of nonlinear algebraic equations has to be solved. Due to the lack of convergence, the programs and methods used by Kucaba-Pietal and Laudanski were unable to solve systems of more than 64 equations. At the same time it was estimated that to solve the engineering problem a system of about 1000 equations would have to be solved. This lead us on a path of finding a globally convergent algorithm (converging from almost any initial starting vector, regardless of its distance to the root) capable of solving large systems of nonlinear algebraic 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 (Rheinboldt 1998). For systems of linear equations, robust libraries of solvers have been developed (see e.g. Anderson 1994). These libraries may not provide the most efficient way of solving a particular special problem, but in most cases they allow to effectively solve a broad class of problems. In addition, sets of test cases have been

developed which can be used as benchmarks against which the newly developed software is to be tested (this is especially the case for linear systems with large sparse matrices; see e.g. Anon 1996). These benchmarks allow for establishing the quality of the new approach and a comparison with existing software aiming at the same problem.

Unfortunately, for systems of nonlinear algebraic equations we are far from a complete understanding of the solution process. Not only do we lack a library of solvers, but also no standard sets of test problems exist (different researchers use different test problems with only a minimal overlap). In this context one should observe that, until recently, in the engineering practice, only systems with relatively few equations have been solved. The increasing computer power resulted in a renewed interest in engineering and optimization problems consisting of a large (and very large) number of equations (Hostetter 1991). This explains one of the problems of existing "popular" test cases. Most of them have a very small number of equations (2-4) 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. It also should be mentioned that in engineering computing (e.g. in electrical engineering) problems arise which involve non-smooth functions (e.g. functions with the absolute value). These problems are also not represented in the test sets we encountered. However, since our original interest is in solving large systems of equations, the non-smooth cases will be omitted from our considerations at this stage.

One of the major goals of this research effort is development of a unified library of test problems and solvers. In order to produce a high-quality product, we are utilizing a software engineering approach. The end product will be expected to obtain all of the characteristics of an well-engineered software system, which include maintainability, reliability, efficiency, and an appropriate user interface. Software engineering models offer several approaches for the development of a system which include exploratory programming, developing a working system as quickly as possible and then modifying that system until it performs in an adequate way; prototyping, developing a program for user experiment followed by development based on user input; and assembly from reusable components, assembling a system from components which already exist (Sommerville 1996). We will take two approaches in constructing the library. First, we will use throwaway prototypes during the investigation of the methods. At the same time, we can also take the exploratory programming and reuse route, which will allow us to explore existing systems that meet our requirements and add them to our library. If we build the software by combining pre-existing proven parts, we will be able to reduce the time needed to build and verify new software (Ghezzi 1991). Not only will we save code development time, but also the reliability of the software is increased and organizational standard can be embodied in the reusable components (Sommerville 1996). When designing the library we will follow the lead of the design effort that led to the creation of the LAPACK library of linear algebraic solvers (Anderson 1994).

The remaining parts of the paper are organized as follows. Section 2 briefly describes various algorithms used to solve systems of nonlinear algebraic equations. In section 3 we summarize the algorithm implementations used in our study and introduce the test problems. Section 4 summarizes the results of our numerical experiments.

2. ALGORITHMS FOR THE SOLUTION OF SYSTEMS OF NONLINEAR ALGEBRAIC EQUATIONS

This section contains a brief summary of algorithms behind nonlinear solvers used in our experiments (in all cases the references cited and (Rheinboldt 1998) should be consulted for the details). We assume that a system of n nonlinear algebraic equations $f(\underline{x}) = \underline{0}$ is to be solved where $\underline{x}$ is n-dimensional vector and $\underline{0}$ is the zero vector.

2.1 Newton's method

The Newton's method for a system of equations is a natural extension of the Newton's method for a single equation (Hostetter 1991). Let us assume that the function G is defined by

$$G(\underline{x}) = \underline{x} - J(\underline{x})^{-1} f(x),$$

and the functional iteration procedure is: select starting vector x_0 and generate a series of vectors

$$\underline{x}_k = G(\underline{x}_{k-1}) = \underline{x}_{k-1} - J(\underline{x}_{k-1})^{-1} f(\underline{x}_{k-1})$$

where $J(\underline{x})$ is the Jacobian matrix. The convergence rate for this method is fast, but the success of the method depends on a good starting vector x_0.

2.2 Brown's method

Brown's method is a modification of Newton's method (Brown 1969). Here, we replace the Jacobian matrix with its difference quotient approximation. This method handles the functions, $f(x) = 0$, one at a time so that information obtained from working with f_1 can be incorporated when working with f_2. A successive substitution scheme is used rather than the simultaneous treatment of the f_i characteristic of Newton's method. Brown's method is derivative-free; moreover, second order convergence has been proven by Brown and Dennis (Brown 1971). There is also an optimal ordering strategy for the system of equations. The equations should be preordered so that the linear ones, or most nearly linear ones, come first and then the equations become progressively more nonlinear (as say measured by their degree). As in case of Newton's method, the convergence is fast, but it requires a good starting vector x_0.

2.3 Steepest Descent Method

In the Steepest Descent Method, the problem of solving the system of nonlinear algebraic equations is transformed into a minimization problem (Burden 1993). A relative minimum x^* of the function $f(x,y)$ with known partial derivatives $g=\frac{\partial f}{\partial x}$, $h=\frac{\partial f}{\partial y}$ is located starting from a given initial guess (x_0, y_0). Using a given step size c, a sequence of steps is generated according to:

$$\underline{x}_n = \underline{x}_n - \frac{c\,\underline{g}}{\sqrt{\underline{g}^2 + \underline{h}^2}},$$

$$\underline{y}_n = \underline{y}_n - \frac{c\,\underline{h}}{\sqrt{\underline{g}^2 + \underline{h}^2}}.$$

The stopping criteria is met when the relative minimum x^* is located and

$$\sqrt{\underline{g}^2 + \underline{h}^2} < \varepsilon.$$

where ε is a tolerance greater than zero. The convergence rate of the steepest descent method is slow, but it works for any choice of starting vectors.

2.4 Trust Region

In the Trust Region method, we replace the Jacobian matrix with an approximation (Rheinboldt 1998). Then we calculate

$$min\ \{||f(\underline{x}_k)+B_k\delta|| : ||D_k\delta||_2 <= \Delta_k\},$$

where D_k is a scaling matrix, δ is the step size factor and Δ_k is the trust region radius. The solution to this minimization problem is an approximate solution to the original problem. Stopping criteria is met when

$$\rho_k = \frac{\| f(\underline{x}_k) \| - \| f(\underline{x}_k + \delta k) \|}{\| f(\underline{x}_k) \| - \| f(\underline{x}_k + Bk\delta k) \|}$$

is smaller than some constant ε_k (typically .0001). Otherwise, we decrease the radius of the trust region and re-solve the minimization problem. The convergence rate of this method is slow, but it can use an arbitrary starting solution vector and still converge.

2.5 Continuation Method

The Continuation Method is designed to be able to target more complicated problems and is the subject of various research efforts (Allgowerr 1990, Stoer 1993). 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

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

where x_0 is a given point in $\mathbb{R}^n$ The problem

$$h(\underline{x}, \lambda) = 0$$

is then solved for values of λ between 0 and 1. When $\lambda=0$, the solution is clearly $x= x_0$. When $\lambda= 1$, we have that

$$h(\underline{x}, 1) = f(\underline{x}),$$

and the solution of $h(\underline{x}, \lambda)$ coincides with the solution of the original problem $f(\underline{x}) = 0$. The convergence rate of the Continuation Methods varies, but the method does not require a good choice of the initial vector $\underline{x}_0$.

2.6 Homotopy Method

The Homotopy (Burden 1993) and Continuation methods are closely related. In the Homotopy method, a given problem

$$f(x) = 0$$

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

$$G(x, \lambda) = f(x) + (1 - \lambda)F(x_0) = 0, \ 0 \leq \lambda \leq 1,$$

for fixed $\underline{x_0} \in \mathbb{R}^n$ forms a homotopy. When $\lambda=0$, the solution is $x(\lambda=1)$. Similarly to the Continuation method, the convergence rate of the homotopy method varies. Homotopy also does not require a good choice of the initial vector.

3. EXPERIMENTAL SETUP

In an effort to obtain a general picture of the existing solvers for systems of nonlinear equations we have experimented with implementations of all the above-described algorithms:

- Brown's Method,
- Hybrid (combination of Trust Region, Steepest Descent, and Newton's methods),
- Continuation Method,
- Homotopy Method.

During the evolutionary phase of our research, we developed two throwaway prototypes, which have assisted us in understanding the methods (Sommerville 1996). The Brown's Method prototype implementation was based on (Prosnak 1993). After experimenting with the Brown code (reference to FSCC 1999), we decided to reuse a well-tested mature implementation of the Brown's method (code SOS), that was obtained from the NETLIB Repository (http: liblist). Our second throwaway prototype, code QUASI_A, is a hybrid algorithm based on (Powell 1970).

A reusable system that uses a similar method was also obtained from NETLIB (code HYBRD1). HYBRD1's design is based on a combination of the Trust Region and Steepest Descent concept. It finds a zero of a system of N nonlinear functions in N variables by a modification of the Powell hybrid method (a modified Newton method).

Codes representing the Continuation (code Contin) and the Homotopy (code Hompack) methods were also obtained from NETLIB. CONTIN, also know as PITCON, computes a sequence of solution points along a one-dimensional manifold of a system of nonlinear equations $f(x)=0$ involving NVAR-1 equations and an NVAR dimensional unknown vector X. HOMPACK (Watson 1987) is a suite of subroutines for solving nonlinear systems of equations by Homotopy methods. The Homotopy method is

carried out via three qualitatively different algorithms: ODE-based (code FIXPDF), normal flow (code FIXPNF), and augmented Jacobian (code FIXPQF).

All codes were Fortran-based implementations and were run in double precision on a PC with a Pentium Pro 200 MHz processor. Some of the codes contain various options that allow the user to tailor the code to fit certain problems. During this phase of this research effort, we chose to utilize default values only. Since the software products and elements already written, tested and used cannot be plugged directly into a new application without some analysis to assure that the product meets all of the actual requirements (Behforooz 1996), a new interface was developed for use with all of the codes. Each code was tested and validated with the test set that accompanied the codes. The Contin and Hompack codes were more complex than the other codes and required some option changes just to get the codes to execute. More sophisticated option settings will be experimented with in future tests.

For the numerical tests we have used 22 problems found in (More 1990, More 1994, Weimann 1991). These problems come from the three collections of test problems for the solution of systems of nonlinear algebraic equations. While some of the problems come from the real life applications, others are artificially generated with properties not typical for real life applications. Table 1 contains a list of the problems used. Detailed descriptions can be found in the appendix of (FSCC 1999).

Table 1. Test problems.

1. Rosenbrock's function (More 1994)	12. Variably dimensioned function (More 1994)
2. Powell singular function (More 1994)	13. Broyden tridiagonal function (More 1994)
3.Powell badly scaled function (More 1994)	14. Broyden banded function (More 1994)
4. Wood function (More 1994)	15. Exponential/Sine Function (Weimann 1991)
5. Helical valley function (More 1994)	16. The Freudenstein-Roth function (More 1990)
6. Watson function (More 1994)	17. Semiconductor Bound. Cond. (Weimann 1991)
7. Chebyquad function (More 1994)	18. Brown Badly Scaled (More 1994)
8. Brown almost-linear function (More 1994)	19. Powell singular Extended(More 1990)
9. Discrete boundary value function (More 1994)	20. Rosenbrock Extended (More 1994)
10. Discrete integral equation function (More 1994)	21. Matrix Square Root Problem (More 1990)
11. Trigonometric function (More 1994)	22. Dennis, Gay, Vu Problem (More 1990)

4. EXPERIMENTAL RESULTS

In our experiments we have found that some problems appearing among the 22 tests studied are relatively easy to solve. In these cases, codes usually converged immediately for random data used in the starting vector or only a minimal amount of work was required for the convergence to be obtained. Table 2 summarizes the results. Here, # denotes problem number (see above), N number of equations, IT number of iterations, and FC number function evaluations.

Table 2. Problems that converged for all codes.

		QUASI_A		HYBRD1		SOS		CONTIN		Hompack's FIXPDF		Hompack's FIXPNF		Hompack's FIXPQF	
#	N	IT	FC	IT	FC	IT	FC	IT	FC	IT	FC	IT	FC	IT	FC
1	2	6	14	6	9	1	10	2	5	23	50	3	6	5	6
3	2	9	16	9	11	5	28	10	74	47	99	11	29	6	8
9	10	6	20	6	16	2	140	3	98	30	63	3	15	4	10
13	10	11	27	11	21	3	205	4	158	25	54	5	15	5	6
14	10	20	41	20	30	4	270	2	48	35	75	6	20	5	6

It can be observed that the Homotopy codes (FIXPNF and FIXPQF) outperform the remaining approaches in terms of number of function evaluations (the real measure of cost) except for in problem 3 where the hybrid codes perform the best even. The Brown method (SOS code) uses a smaller number of iterations in most cases. The Continuation method can require a smaller number of iterations, however it uses a larger number of function evaluations. The in-house hybrid prototype is slightly less efficient than its NETLIB-based counterpart.

Table 3 summarizes the results of test cases for which one or more codes were unable to converge. As previously, # denotes problem number, N number of equations, IT number of iterations, and FC number function evaluations (for problems 6 and 7 we varied the number of equations N). In addition, code *nc*, found among Quasi_A, HYBRD and HOMPACK results indicates that the execution terminated before convergence due to iterations not making good progress as measured by the improvement from the last five Jacobian evaluations. Code *nd*, found among the SOS results indicates that the execution terminated before convergence due to the iterative scheme judged to be diverging (the residual norms and solution increment norms increased over several consecutive iterations). The codes indicating abnormal termination for CONTIN are *nf*, *ne*, and *ni*; *nf* indicates the solver failed with a zero pivot error, *ni* means that the code did not reach a point of interest, *ne* specifies the problem terminated due to an error. In each case a number of starting points have been tried and the best IT and FC results are reported. This is to indicate a crude estimate on the best possible speed of convergence.

Table 3. Results of test problems with lack of convergence.

		QUASI_A		HYBRD1		SOS		CONTIN		Hompack's FIXPDF		Hompack's FIXPNF		Hompack's FIXPQF	
#	N	IT	FC	IT	FC	IT	FC	IT	FC	IT	FC	IT	FC	IT	FC
2	4	nc	nc	nc	nc	1	28	10	446	nc	nc	nc	nc	nc	nc
4	4	52	142	52	94	38	536	2	43	nc	nc	nc	nc	nc	nc
5	3	14	38	14	27	19	174	2	68	nc	nc	nc	nc	nc	nc
6	6	38	127	38	96	nd	nd	6	269	131	267	29	79	14	23
6	9	54	180	57	132	nd	nd	9	575	334	674	43	123	26	42
7	5	11	24	11	17	3	50	nf	nf	38	80	10	27	6	10
7	6	10	34	10	25	7	111	nf	nf	nc	nc	nc	nc	nc	nc
7	7	11	27	11	20	3	84	nf	nf	nc	nc	nc	nc	nc	nc
7	9	19	57	19	43	nd	nd	nf	nf	116	243	38	107	66	114
8	10	8	37	8	31	5	280	4	50	nc	nc	nc	nc	nc	nc
10	10	6	20	6	16	2	140	ni	ni	21	46	3	10	3	4
11	10	23	103	33	81	4	270	ne	ne	nc	nc	nc	nc	nc	nc
12	10	25	62	23	46	nd	nd	5	254	nc	nc	nc	nc	nc	nc
15	2	8	11	8	16	nd	nd	1	16	nc	nc	nc	nc	nc	nc
16	2	nc	nc	nc	nc	8	42	3	82	45	94	12	26	13	14
17	6	3	9	2	10	8	42	3	82	nc	nc	nc	nc	nc	nc
18	2	16	18	15	25	1	10	ni	ni	nc	nc	nc	nc	nc	nc
19	12	nc	nc	nc	nc	15	1440	ni	ni	nc	nc	nc	nc	nc	nc
20	10	6	17	6	22	2	195	3	97	nc	nc	nc	nc	nc	nc
21	4	413	847	413	1249	19	260	ni	ni	nc	nc	nc	nc	nc	nc
22	6	41	108	41	140	13	378	ni	ni	nc	nc	nc	nc	nc	nc

It can be observed that even though each code had at least one failure, the hybrid codes recorded the smallest overall number of them. It is rather surprising to see that the performance of the Continuation and Homotopy codes is rather disappointing. Since these codes are more complex than the others, it is clear that the default options do not apply to most problems and more effort must be placed in setting up the codes. When the codes converged, the Hompack (FIXPNF and FIXPQF) codes appear to outperform all other codes. As previously, the prototype Hybrid code is outperformed by its counterpart as far as the number of iterations and function evaluations are concerned. However, they behave quite similarly as far as the potential for solving a given problem is concerned. As previously, the hybrid methods require fewer function evaluations (while they require more iterations) than the implementations of the Brown's method (which require fewer iterations, but substantially more function evaluations).

In all experiments we have observed extreme sensitivity of the solvers to the selection of the starting vector $\underline{x}_0$. This problem becomes more pronounced as the number of equations increases.

5. CONCLUSIONS AND FUTURE WORK

In this note we have reported on our experiments comparing performance of solvers for systems of nonlinear algebraic equations on a number of test problems. We have found that methods based on similar algorithms behave similarly and the implementation details have a relatively small impact on performance. All methods, regardless of their underlying algorithm, showed high sensitivity to the starting vector and this sensitivity increased as the number of equations in the system increased. Out of the methods tested, the hybrid method appeared to be most robust and capable of solving largest number of problems. However, since not a single approach was capable of solving all test cases, there seems to be a message here for the practitioners. None of the approaches can be trusted and thus multiple approaches should be used to improve a chance of finding a correct answer.

The test problems that we are currently using do not point to the existence of "one best solver." In addition, numbers of equations are too small to add a timing component into the evaluation method. We were able to find that the popular test problems can be divided into two groups: a set of "relatively easy problems," where even the least powerful methods were capable of converging, and a set of "tough problems" where convergence is difficult to obtain. It should be stressed again that even though the tests used here cover a wide spectrum of functions they clearly do not exhaust the possibilities arising in practical engineering applications. First, such applications can result in systems of 100's of nonlinear algebraic equations and none of the test cases used in the current research seems to belong to this category (only few of the examples found can be extended to this many equations). In addition, none of the examples belongs to the class of non-smooth functions (e.g. functions with an absolute value).

In the near future we plan to proceed as follows. We will expand the test set (our literature and Internet searches have located additional test cases) and experiment with the above codes on these test cases. We will experiment with various option settings for the Continuation and Homotopy codes. We will also continue to add codes to our experiments, first by reuse only and in the future development of new functions. The sensitivity of the codes and test sets to the selection of starting vectors will be investigated. This should allow us to develop new robust methods for finding starting vectors. Finally, an attempt at solving large, engineering based systems will be made.

Acknowledgements. Work of Deborah Dent was sponsored by the US Army Corps of Engineers, Waterways Experiment Station, Vicksburg, Mississippi, and USA.

REFERENCES

Allgowerr, E., and Georg, K. (1990) Numerical Continuation Methods: An Introduction (Springer-Verlag, Berlin), 365.

Anderson, E., Bai,, Z., Bischof, C., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., Ostrouchov, S., and Sorensen, D. (1994) LAPACK Users' Guide, (SIAM, Philadelphia).

Anon. (1996) Harwell Subroutine Library: A Catalogue of Subroutines (AEA Technology, Oxfordshire).

Brown, K.M. (1969) "Quadratically Convergent Newton-Like Method Based Upon Gaussian Elimination," SIAM J. Num. Anal., **6**, 560-569.

Brown, K.M. and Dennis, J.E., (1971) "On the Second Order Convergence of Brown's Derivative-Free method for Solving Simultaneous Nonlinear Equations," Technical Report 71-7 (Yale University Department of Computer Science, New Haven, Conn., United States).

Behforooz, Ali and Hudson, Frederick J. (1996), Software Engineering Fundamentals (Oxford University Press, Oxford) 41.

Burden, R.L. and Faries, J.D. (1993), Numerical Analysis (PWS-Kent Publishing Company, Boston), 575-576.

Dent, Deborah, Paprzycki, Marcin and Kucaba-Pietal, Anna (1999) "Testing Convergence of Nonlinear System Solvers," FSCC, **1**, http://pax.st.usm.edu/cmi/fscc98_html/processed/.

Ghezzi, Carlo, Jazugeri, Mehdi, and Mandrioli, Dino (1991), Fundamentals of Software Engineering, (Prentice Hall, Englewood Cliffs) 591.

Hostetter, G.H., Santina, M.S., and D'Capio-Montalvo, P. (1991) Analytical, Numerical and Computational Methods for Science and Engineering, (Prentice Hall, Englewood Cliffs).

http://www.netlib.org/liblist.html.

Kucaba-Pietal A., Laudanski, L. (1995) "Modeling Stationary Gaussian Loads," Scientific Papers of Silesian Technical University, Mechanics ,**121,** 173-181.

Laudanski, L. (1996) "Designing Random Vibration Tests," Int. J. Non-Linear Mechanics, **31(5),** 563-572.

More, J.J. (1990) "A Collection of Nonlinear Model Problems," Amer. Math. Soc., **26**, 723-762.

More, J.J., Garbow, B.S., Hillstrom, K.E. (1994) "Algorithm 566", ACM Trans, Math. Software, **20(3),** 282-285.

Powell, M.J.D. (1970), Methods for Nonlinear Algebraic Equations, (Gordon and Breach, Rabinowitz).

Prosnak, W. (1993), Introduction to Numerical Fluid Mechanics, in Polish, (Ossolineum Gdansk).

Rheinboldt, W.C. (1998), Methods for Solving System of Nonlinear Equations, (SIAM, Philadelphia).

Sommerville, Ian (1996), Software Engineering, (Addison-Wesley Publishing Company, Reading), 39
398.

Stoer, J., Bulirsh, R. (1993), Introduction to Numerical Analysis (Springer, New York), 521.

Weimann, U.N. (1991) "A Family of Newton Codes for Systems of Highly Nonlinear Equations," ZIB Technical Report TR-91-10 (ZIB, Berlin, Germany).