Proceedings of the First Southern Symposium on Computing
The University of Southern Mississippi, December 4-5, 1998

TESTING CONVERGENCE OF NONLINEAR SYSTEM SOLVERS

DEBORAH DENT*, MARCIN PAPRZYCKI* AND ANNA KUCABA-PIĘTAL†

Abstract. Solution of systems of nonlinear algebraic equations is a relatively complicated problem with few definite answers. The situation becomes even more complex when the number of equations increases. In this note we compare the performance of several solvers applied to a number of popular test cases. We concentrate our attention on the robustness of the solvers and the properties of the test problems.

1. Introduction. The solution of systems of algebraic equations has a well-developed mathematical and computational theory when the equations are linear, or when a single nonlinear equation is to be solved [15]. When systems of linear equations are considered, libraries of solvers have been developed (see e.g. [5]). These libraries may not provide the most efficient way of solving a particular special problem, but they should be robust enough 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. [2]). These benchmarks allow for establishing the quality of the new approach and a comparison with existing software aiming at the same problem.

The situation is much more complicated for systems of nonlinear algebraic equations. Both the mathematical theory and computational practice are far from a complete understanding of the solution process. No only we lack a library of solvers, but also no agreed upon sets of test problems exist (different researchers use different test problems with only a minimal overlap). It should be also pointed out 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 [8]. This points out to a potential problem with the existing "popular" test cases. Most of them are characterized by 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 test cases corresponding to the real-life engineering problems of 100+ equations. Finally, it may be worth mentioning that in the engineering computing (e.g. in electrical engineering) practical 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 interest is in solving large systems of equations, the non-smooth cases will be omitted from our considerations.

The aim of this note is to report on the progress of our investigations in the direction of building a library of nonlinear solvers and compiling a set of test problems. A survey of methods, software implementations, and standard test problems is presented and the results of experiments discussed. Section 2 briefly describes various algorithms for solving nonlinear systems of algebraic equations. In section 3 we describe the engineering origins of our research, summarize the algorithm implementations used in our study and introduce the test problems used. Section 4 presents and discusses the results of our numerical experiments. The Appendix contains complete definitions of all test problems used.

2. Algorithms for the solution of systems of nonlinear algebraic equations. This section contains a brief summary of algorithms behind nonlinear solvers that we have experimented with (in all cases the references cited and the book by Rheinboldt [15] should be consulted for the details). We assume that a

* School of Mathematical Sciences, University of Southern Mississippi, Hattiesburg, MS 39406-5106.
deborah.dent@usm.edu, m.paprzycki@usm.edu.
† Department of Fluid Mechanics and Aerodynamics, Technical University of Rzeszów.
anpietal@ewa.prz.rzeszow.pl.

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 [8]. Let us assume that the function G is defined by

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

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 [3]. Here, we replace the Jacobian matrix with its difference quotient approximation. For each iteration, only one function from the system at a time is evaluated (function f_i for use with f_{i+1}). A successive substitution scheme is used for treatment of f_i. As in case of Newton's method, the convergence is fast, but it requires a good starting vector x_0.

2.3. Characteristic Bisection Method. A generalized bisection, known as the characteristic bisection method [19], can be applied to systems of equations. It is based on the topological degree theory, which locates at least one root of the system within an n-dimensional polyhedron. The algorithm only makes use of the algebraic sign of $\boldsymbol{F}(\boldsymbol{X})$ where $\boldsymbol{F}(\boldsymbol{X}) = \Theta = (0,0,\ldots,0) \in \mathbb{R}^n$, and $\boldsymbol{F}$ is a given continuous mapping of a bounded region $\boldsymbol{D}$ in $\mathbb{R}^n$ into $\mathbb{R}^n$, in a polyhedron $\boldsymbol{P}$ in $\boldsymbol{D}$. One method for obtaining at least one root of a system is to compute the topological degree of $\boldsymbol{F}$ at Θ relative to a $\boldsymbol{P}$. This method is very time-consuming and cannot be accurately achieved unless the modulus of continuity of $\boldsymbol{F}$ on $\boldsymbol{P}$ is known. Another concept allows the use of what is called a characteristic n-polyhedron, $\boldsymbol{CP}$, by which all calculations concerning the topological degree can be avoided. This method requires the bisection of $\boldsymbol{CP}$ in such a way that the new refined n-polyhedron is also a characteristic one. First, a vector of signs of $\boldsymbol{F}$ is computed. Next we let $\langle X_i, X_j \rangle$ be a proper 1-simplex of $\boldsymbol{CP}$, and let $B=(X_i+X_j)/2$ be its midpoint. The characteristic bisection is repeatedly applied to the diagonals, starting with $\langle X_k, X_l \rangle$ $<X_k X_l>$ until sign $F((X_k + X_l)/2)$ becomes different from sign $F(X_k)$ and from sign $F(x_l)$. If a point, AS, is obtained so that $\|F(AS)\| \le 10^{-8}$ then AS is considered the final approximate solution. Otherwise the process is repeated until a solution if found or the maximum number of iterations are exceeded.

The Bisection method, though conceptually clear, has significant drawbacks. It is very slow in converging (that is, n may become quite large before the convergence criteria is met) and, moreover, a good intermediate approximation may be inadvertently discarded. However, the method has the important property that it always converges to a solution (if the computational limits, e.g. number of allowed iterations, are set high enough) and is often used as a "starter" for other methods [4].

2.4. Steepest Descent Method. In the Steepest Descent Method the problem of solving the system of nonlinear algebraic equations is transformed into a minimization problem [4]. A relative minimum x^* of the function $f(x,y)$ with known partial derivatives $g=\frac{\partial f}{\partial x}$, $h=\frac{\partial f}{\partial x}$ 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

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

The convergence rate of the steepest descent method is slow, but it works for any choice of starting vectors.

2.5. Trust Region. In the trust region method we replace the Jacobian matrix with an approximation [6]. Then we calculate

$$\min\left\{\left\| f(\underline{x}_k) + B_k\delta \right\| : \left\| D_k\delta \right\|_2 \le \Delta_k \right\}$$

where D_k is a scaling matrix 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

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

is smaller than some constant σ_0 (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.

2.6. Continuation Method. The Continuation Method is supposed to be able to target more complicated problems and is the subjects of current research [1, 17]. This method expected to be slower than line-search and 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 λ=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 Continuation Methods varies, but the method does not require a good choice of the initial vector x_0.

3. Experimental setup. The initial aim of our research was to find an efficient solution to an engineering problem arising in avionics [9, 10]. After initial experiments we have found out that with the increasing number of equations (64, 128, ...) it was extremely difficult to obtain solution. We have thus decided to obtain a general picture of the existing software that can be possibly used to solve the original problem. Thus far we have experimented with the following methods:

- implementation of the Bisection Method,
- implementation of the Brown's Method,
- implementation of combination of Trust Region, Steepest Descent, and Newton's methods,
- implementation of the Continuation Method.

For the bisection method we used the CHABIS software package [18]. We used two versions of Brown's Method; an in-house implementation based on [14] (code BROWN) and a well-tested mature implementation of the Brown's method (code SOS). We have also used two versions of the hybrid algorithm. An in-house version based on [13] (code QUASI_A) and an another well-tested mature implementation, (code HYBRDI). For the Continuation Method we used the software package, CONTIN [16]. The codes for CHABIS, CONTIN, HYBRDI and SOS were all obtained from the NETLIB Repository [7]. All algorithms and codes were Fortran-based implementations and were run in double precision on a PC with a Pentium Pro 200 MHz processor.

For the numerical tests we have used 24 problems found in [11,12,19]. 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. The problems used were (see the references and the Appendix for the complete formulation):

1. Rosenbrock's function [12]	13. Broyden tridiagonal function[12]
2. Powell singular function [12]	14. Broyden banded function[12]
3.Powell badly scaled function [12]	15. Exponential/Sine Function[20]
4. Wood function [12]	16. The Freudenstein-Roth function[11]
5. Helical valley function[12]	17. Semiconductor Boundary Condition[20]
6. Watson function[12]	18. Gulf Research and Development[11]
7. Chebyquad function[12]	19. Brown Badly Scaled[11]
8. Brown almost-linear function[12]	20. Beale Function[11]
9. Discrete boundary value function[12]	21. Jennrich and Sampson[11]
10. Discrete integral equation function[12]	22. Bard[13]
11. Trigonometric function[12]	23. Gaussian[11]
12. Variably dimensioned function[12]	24. Linear Full Rank[11]

4. Experimental results. In our experiments we were able to find out that some of the problems appearing among the 24 test problems 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 1 summarizes the results. Here, # denotes problem number (see above), N number of equations, IT number of iterations, and FC number function evaluations.

		HYBRDI		QUASI_A		SOS		BROWN		CONTIN		CHABIS	
#	N	IT	FC	IT	FC	IT	FC	IT	FC	IT	FC	IT	FC
1	2	6	9	6	14	1	10	5	20	2	5	3	62
3	2	9	11	9	16	5	28	7	30	10	74	9	91
5	3	14	27	14	38	19	174	8	63	2	68	35	866
14	10	20	30	20	41	4	270	7	390	2	48	33	20709
16	6	3	9	3	11	1	54	5	108	3	82	34	6814
18	3	2	5	2	7	1	9	5	36	3	7	1	2

TABLE 1. *Problems that converged for all codes*

It can be observed that the hybrid methods outperform the remaining approaches in terms of number of function evaluations (the real measure of cost). They also use a considerably smaller number of iterations than the Bisection and Brown codes. The continuation method requires a smaller number of iterations than the hybrid methods, however it uses a larger number of function evaluations. The two in-house codes are less efficient than their NETLIB-based counterparts. The bisection method was not expected to perform well when compared to the other methods. With the exception of problem 18 (which had a very good starting vector), our test proved this expectation to be correct.

Table 2 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 CHABIS, a Quasi_A and HYBRDI result indicates that the execution terminated before convergence. Quasi_A and HYBRDI terminated before convergence due to iterations not making good progress as measured by the improvement from the last five Jacobian evaluations. CHABIS terminated due to the maximum number of iterations being exceeded. Code *nd*, found among the SOS and Brown 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 has 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.

		HYBRDI		QUASI A		SOS		BROWN		CONTIN		CHABIS	
#	N	IT	FC	IT	FC	IT	FC	IT	FC	IT	FC	FC	IT
2	4	*nc*	*nc*	*nc*	*nc*	1	28	4	42	10	446	*nc*	*nc*
4	4	52	94	52	142	38	536	37	504	2	43	*nc*	*nc*
6	6	38	96	38	127	*Nd*	*nd*	*nd*	*nd*	6	269	*nc*	*nc*
6	9	57	132	54	180	*Nd*	*nd*	*nd*	*nd*	9	575	*nc*	*nc*
7	5	11	17	11	24	3	50	10	180	*nf*	*nf*	*nc*	*nc*
7	6	10	25	10	34	7	111	20	665	*nf*	*nf*	*nc*	*nc*
7	7	11	20	11	27	3	84	nd	Nd	*nf*	*nf*	*nc*	*nc*
7	9	19	43	19	57	*Nd*	*nd*	18	918	*nf*	*nf*	*nc*	*nc*
8	10	8	31	8	37	5	280	9	520	4	50	*nc*	*nc*
9	10	6	16	6	20	2	140	6	325	3	98	*nc*	*nc*
13	10	11	21	11	27	3	205	8	455	4	158	*nc*	*nc*
10	10	6	20	6	16	2	140	5	260	*ni*	*ni*	*nc*	*nc*
11	10	33	81	23	103	4	270	8	455	*ne*	*ne*	*nc*	*nc*
12	10	23	46	25	62	*Nd*	*nd*	*nd*	*nd*	5	254	*nc*	*nc*
15	2	8	11	8	16	*Nd*	*nd*	*nd*	*nd*	1	16	*nc*	*nc*
17	3	2	5	1	7	*Nd*	*nd*	5	36	2	295	35	315
19	2	16	18	15	25	1	10	5	20	*ne*	*ne*	*nc*	*nc*
20	2	18	28	18	40	24	119	104	515	*ni*	*ni*	*nc*	*nc*
21	2	1	28	*nc*	*nc*	*Nd*	*nd*	*nd*	*nd*	*ni*	*ni*	*nc*	*nc*
22	3	11	8	16	8	54	7	39	4	67	4	*nc*	*nc*
23	3	35	75	35	111	36	3	30	2	*ni*	*ni*	*nc*	*nc*
24	10	3	29	4	34	35	6	130	1	*ni*	*ni*	*nc*	*nc*

TABLE 2. *Results of test problems with lack of convergence*

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 code is rather disappointing (it fails in more cases than the Brown Method implementations). As previously, the in-house codes are outperformed by their counterparts as far as the number of function evaluations and iterations 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 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 hybrid algorithms 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.

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 even 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. The Homotopy [8] code will be used on all test problems and we will continue to add codes to our experiments. 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

[1] ALLGOWERR, E., AND K. GEORG. *Numerical Continuation Methods: An Introduction*. Springer-Verlag, Berlin (1990).

[2] ANON. 1996. Harwell Subroutine Library. *A Catalogue of Subroutines* (*Release* 12). AEA Technology, Harwell Laboratory, Oxford shire, England.

[3] K.M. BROWN, *Quadratically Convergent Newton-Like Method Based Upon Gaussian Elimination*, SIAM J. Num Anal., Vol. 6, 1969, pp. 560-569.

[4] R.L. BURDEN AND J.D. FARIES, *Numerical Analysis*, PWS-Kent Publishing Company, Boston (1993).

[5] `http://spcons.cnuce.cnr.it/software/lapack.html`

[6] `http://www-c.mcs.anl.gov/home/otc/Guide/OptWeb/continuous/unconstrained/nonlineareq/trls.html`

[7] `http://www.netlib.org/liblist.html`

[8] G.H. HOSTETTER, M.S. SANTINA AND P.D'CAPIO-MONTALVO, *Analytical, Numerical and Computational Methods for Science and Engineering*, Prentice Hall, New Jersey (1991).

[9] KUCABA-PIĘTAL, L. LAUDAŃSKI, Designing Random Vibration Tests, Proceedings of I Symposium of Dynamics of Mechanical Systems, Warsaw (1994).

[10] KUCABA-PIĘTAL, L. LAUDAŃSKI, Modeling Stationary Gaussian Loads, *Zeszyty* Naukowe Politechniki Slaskiej, Seria Mechanika 121, pp. 173-181 (1995).

[11] J.J. MORE, A Collection Of Nonlinear Model Problems. Preprint MCS-P60-0289, Mathematics and Computer Science Division, Argonne National Laboratory (1989).

[12] J.J. MORE, B.S. GARBOW, K.E. HILLSTROM, Testing Unconstrained Optimization Software, ACM Trans, Math. Software, 7(1), pp. 17-41 (1981).

[13] M.J.D. POWELL, A Hybrid Method for Nonlinear Algebraic Equations, in: P. Rabinowitz (ed.), Gordon and Breach (1970).

[14] W. PROSNAK, *Introduction to Fluid Mechanics* (in Polish), Ossolineum. Wroclaw (1993).

[15] W.C. RHEINBOLDT, *Methods for Solving System of Nonlinear Equations*, Society for Industrial and Applied Mathematics, Philadelphia (1998).

[16] W.C. RHEINBOLDT, W.C., J. BURKARDT, Algorithm 596: A Program For A Locally Parameterized Continuation Process, ACM Trans. Math. Software, 9, pp. 236-241 (1980).

[17] J. STOER, R. BULIRSH, *Introduction to Numerical Analysis*. Springer, New York (1993).

[18] M.N. VRAHATIS, Algorithm 666: CHABIS: A Mathematical Software Package Systems of Nonlinear Equations, ACM Transactions on Mathematical Software, Vol. 14, No.4, pp. 312-329 (1988).

[19] M.N. VRAHATIS, Solving Systems of Nonlinear Equations Using the Nonzero Value of the Topological Degree, ACM Transactions on Mathematical Software, Vol. 14, No.4, pp. 330-336 (1988).

[20] U.N. WEIMANN, A Family of Newton Codes for Systems of Highly Nonlinear Equations, ZIB Technical Report TR-(1-Berlin (1991), http://weyl.zib-berlin.de/Documents/tr91-10/sc90-10.html

[21] D. ZWILLINGER, *CRC Standard Mathematical Tables and Formulae*, CRC Press, Boca Raton (1996).

APPENDIX

PROBLEM 1. *Rosenbrock Function*

$$F(x_1) = 1 - x_1$$
$$F(x_2) = 10(x_2 - x_1)^2$$

PROBLEM 2. *Powell Singular Function*

$$F(x_1) = x_1 + 10x_2$$
$$F(x_2) = \sqrt{5}(x_3 - x_4)$$
$$F(x_3) = (x_2 - 2x_3)^2$$
$$F(x_4) = \sqrt{10}(x_1 - x_2)^2$$

PROBLEM 3. *Powell Badly Scaled Function*

$$F(x_1) = 1000x_1x_2 - 1$$
$$F(x_2) = e^{-x1} + e^{-x2} - 1.001$$

PROBLEM 4. *Wood Function*

$$F(x_1) = -(200x_1(x_2 - x_1^2)) - (1 - x1)$$
$$F(x_2) = 200(x_2 - x_11^2) - 20.2(x_2 - 1) + 19.8(x_4 - 1)$$
$$F(x_3) = -(.25x_3(x_4 - x_3^2)) - (1 - x3)$$
$$F(x_4) = 180(x_4 - x_3^2) - 20.2(x_4 - 1) + 19.8(x_2 - 1)$$

PROBLEM 5. *Helical Valley Function*

$$\Lambda = \frac{a\tan(\frac{x_2}{x_1})}{8a\tan(1)}, x_1 > 0,$$

$$\Lambda = \frac{a\tan(\frac{x_2}{x_1})}{8a\tan(1)} + .4, x_1 > 0,$$

$$\Lambda = sign(.25, x_2), x_1 = 0$$

$$F(x_1) = 10(x_3 - 10\Lambda)$$

$$F(x_2) = 10(\sqrt{x_1^2 + x_2^2} - 1)$$

$$F(x_3) = x_3$$

PROBLEM 6. *Watson Function*

$$\Lambda_i = (\sum_{j=2}^{n}(j-1)\frac{i}{29}^{j-2} x_j) - (\sum_{k=1}^{n}\frac{i}{29}^{k-1} x_k)^2 - 1$$

$$B_i = \frac{2i}{29}\sum_{k=1}^{n}\frac{i}{29}^{k-1} x_k$$

$$F(x_i) = \sum_{m=1}^{n}\frac{29}{i}((m-1) - B_i)\Lambda_i, i = 1,29$$

$$F(x_1) = F(x_i) + x_1(1 - 2(x_2 - x_1^2 - 1)$$

$$F(x_2) = F(x_2) + x_2 - x_1^2 - 1$$

PROBLEM 7. *Chebyquad Function*

$$F(x_{2i}) = \frac{1}{2i^2 - 1}, i = 1, n$$

$$For -> j = 1, n$$

$$\Lambda_0 = 1$$

$$\Lambda_1 = 2x$$

$$\Lambda_i = 2(2x_j - 1)\Lambda_{i-1} - \Lambda_{i-2},$$

$$F(x_i) = \frac{1}{n}A_i, i = 2, n$$

PROBLEM 8. *Brown Almost-Linear Function*

$$F(x_i) = x_i + (\sum_{j=1}^{n} x_j), i = 1, n-1$$

$$F(x_n) = \prod_{j=1}^{n} x_j - 1$$

PROBLEM 9. *Discrete Boundary Value Function*

$$F(x_i) = 2x_i - x_{i-1} - x_{i+1} + \frac{(x_i + \frac{i}{n+1} + 1)^3}{2}, i = 1, n$$

PROBLEM 10. *Discrete Integral Equation Function*

$$A_i = \sum_{j=1}^{n} \frac{j}{n+1}(x_j + \frac{j}{n+1} + 1)^3$$

$$B_i = \sum_{j=1}^{n} (1 - \frac{j}{n+1})(x_j + \frac{j}{n+1} + 1)^3$$

$$F(x_i) = x_i + \frac{1}{n+1}\frac{(1 - \frac{i}{n+1})A_i + \frac{i}{n+1}B_i}{2}, i = 1, n$$

PROBLEM 11. *Trigonometric Function*

$$F(x_i) = n + i - \sin(x_i) - \sum_{j=1}^{n} \cos(x_j) - ix_i, i = 1, n$$

PROBLEM 12. *Variably Dimensioned Function*

$$A = \sum_{j=1}^{n} j(x_j - 1)$$

$$B = A(1 - 2A^2)$$

$$F(x_i) = x_i - 1 + iB_i, i = 1, n$$

PROBLEM 13. *Broyden Tridiagonal Function*

$$A = x_{i-1}, i > 1, otherwise = 0$$
$$B = x_{i+1}, i > 1, otherwise = 0$$
$$F(x_i) = (3 - 2x_i^2) - A - 2B + 1, i = 1, n$$

PROBLEM 14. *Broyden Banded Function*

$$A_i = \sum_{j=1}^{n} x_j(1 + x_j), j \neq k, otherwise = 0$$
$$F(x_i) = x_i(2 + 5x_i^2) + 1 - A_i, i = 1, n$$

PROBLEM 15. *Artificial Test Problem*

$$F(x_1) = e^{x_1^2} + x_2^2 - 3$$
$$F(x_2) = x_1 + x_2 + \sin(3x_1 + x_2)$$

PROBLEM 16. *Semiconductor Device Simulation*

$$F(x_1) = e^{\alpha(x_3 - x_1)} - e^{\alpha(x_1 - x_2)}$$
$$F(x_2) = x_2$$
$$F(x_3) = x_3$$
$$F(x_4) = e^{\alpha(x_6 - x_4)} - e^{\alpha(x_4 - x_5)}$$
$$F(x_5) = x_5 - 100$$
$$F(x_6) = x_6 - 100$$

PROBLEM 17. *Freudenstein And Roth*

$$F(x_1) = -13 + x_1 + ((x_2^3 - 5x_2^2 + 2x_2)$$
$$F(x_2) = -29 + x_1 + ((x_2^3 + x_2^2 + 14x_2)$$

PROBLEM 18. *Gulf Research And Development*

$$F(x_i) = e^{\left|-\alpha^{x_3}\right|^{x_1}} - .01i, i = 1,3$$

Where $\alpha = 25 + (-50\log(.01i)^{\frac{2}{3}})$

PROBLEM 19. *Brown Badly Scaled*

$$F(x_1) = x_1 - 10^6$$
$$F(x_2) = x_2 - 2 * 10^{-6}$$
$$F(x_3) = x_1 x_2 - 2$$

PROBLEM 20. *Beale Function*

$$F(x_1) = 1.5 - x_1(1 - x_2)$$
$$F(x_2) = 2.25 - x_1(1 - x_2{}^2)$$
$$F(x_3) = 2.62 - x_1(1 - x_2{}^3)$$

PROBLEM 21. *Jennrich and Sampson*

$$F(x_1) = 4 - (e^{x_1} + e^{x_2})$$
$$F(x_2) = 6 - (e^{2x_1} + e^{2x_2})$$

PROBLEM 22. *Bard*

$$y = \left\{\begin{matrix} 0.14 \\ 0.18 \\ 0.22 \\ 0.25 \\ 0.29 \\ 0.32 \\ 0.35 \\ 0.39 \\ 0.37 \\ 0.58 \\ 0.73 \\ 0.96 \\ 1.34 \\ 2.10 \\ 4.39 \end{matrix}\right\}, F(x_i) = x_1 + \frac{i}{(16 - i) + \min(i, 16 - i)} x_3 - y_i, i = 1, n$$

PROBLEM 23. *Gauss*

$$y = \begin{Bmatrix} .0009 \\ .0044 \\ .0175 \\ .0540 \\ .1295 \\ .2420 \\ .3521 \\ .3989 \\ .3521 \\ .2420 \\ .1295 \\ .0540 \\ .0175 \\ .0044 \\ .0009 \end{Bmatrix}, F(x_i) = x_1 e^{\frac{-x_2(\alpha - x_3)^2}{2}} - y_i, i = 1, n$$

PROBLEM 24. *Linear – Full Rank*

$$F(x_i) = x_i - \frac{2}{n} \sum_{j=1}^{n} x_j - 1$$