1990

Software Performance Evaluation Papers in TOMS, Volumes 1-15

John R. Rice
Purdue University, jrr@cs.purdue.edu

Report Number:
90-1026

https://docs.lib.purdue.edu/cstech/28

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries. Please contact epubs@purdue.edu for additional information.
SOFTWARE PERFORMANCE EVALUATION
PAPERS IN TOMS, VOLUMES 1-15

J.R. Rice*

Computer Sciences Department
Purdue University
Technical Report CSD-TR-1026
CAPO Report CER-90-39
October, 1990

* This report is part of the project on numerical software performance evaluation sponsored by IFZP WG2.5 (Working Group on Numerical Software).
SOFTWARE PERFORMANCE EVALUATION
PAPERS IN TOMS, VOLUMES 1-15

John R. Rice*

Computer Sciences Department
Purdue University
Technical Report CSD-TR-1026
CAPO Report CER-90-39
October, 1990

ABSTRACT

This report contains: 1) A list of 45 papers in the Association of Computing Machinery Transactions on Mathematical Software (TOMS) which consider the performance evaluation of software, either in general or for specific classes of applications. 2) The abstract and list of references for each of these papers. Some papers from Volume 16 (1990) are included but this list is not complete as the complete contents of this volume is not yet known.

* This report is part of the project on numerical software performance evaluation sponsored by IFZP WG2.5 (Working Group on Numerical Software).
Principles for Testing Polynomial Zerofinding Programs.

M.A. Jenkins and J.F. Traub, 26–34.

The state of the art in polynomial zerofinding algorithms and programs is briefly summarized, with emphasis on the principles for testing such programs. The authors view testing as requiring four stages: (1) testing program robustness, (2) testing for convergence difficulties, (3) testing for specific weakness of the algorithms, (4) assessment of program performance by statistical testing. It is emphasized that the statistical testing must be done with care. There are many ways to generate "random" polynomials, but two classes of random polynomials which have been widely used are of only limited usefulness in terms of evaluating reliability or performance because they produce polynomials with very similar characteristics. Classes of random polynomials which should be used are discussed.

References
Comments on the Nature of Automatic Quadrature Routines.


General properties of automatic quadrature routines are described, and the conflict between reliability and economy which arises when one constructs such a routine is discussed in detail. The concept of the “performance profile” is introduced; this serves as an aid in understanding how such routines may fail for some problems and why the cost is so much higher than in a nonautomatic routine. The relationship (often tenuous) between analytic results from numerical analysis relating to quadrature rules and the actual strategy used in these routines is discussed. Attention is drawn to the fact that an automatic quadrature routine is based on an unreliable exact arithmetic algorithm and, unlike simpler routines, is not based on a finite decision process. In this paper, those aspects of automatic quadrature routines which are relevant in the context of testing routines are gathered together as an introduction to forthcoming specific testing techniques.

References

A Comparison of Algorithms for Solving Symmetric Indefinite Systems of Linear Equations,

V. Barwell and A. George, 242–251.

A brief description of numerically stable methods for solving symmetric indefinite systems of linear equations $Ax = b$ is presented. The computer implementation of the methods is described and some numerical experiments are reported, comparing program performance on two different computers. The comparison leads to the conclusion that the algorithms proposed by Aasen and Bunch-Kaufman appear to be substantially superior to alternate methods.

References

A Comparison of Several Bandwidth and Profile Reduction Algorithms,

Large sparse matrix problems arise in many application areas such as structural engineering, fluid dynamics, and network analysis. Many algorithms for automatically reordering the rows and columns of a matrix in order to reduce the bandwidth and profile have been published. In this paper analysis and test results are reported for new algorithms of Cheng, Collins, Gibbs-Poole-Stockmeyer, and Wang; these algorithms are compared with those of Cuthill-McKee and King, which appeared most promising in a 1971 Cuthill paper. Test problems include (1) a collection of 19 sparse matrices derived from structural engineering problems and (2) several families of matrices whose adjacency graphs are rectangles and cylinders with quadrilateral and triangular elements. Set (1) was used to try to test the behavior of the various algorithms when applied to representative application problems. Set (2) was used to try to assess the asymptotic behavior of the algorithms as the order of the matrices increased.

References
A Comparison of Three Algorithms for Linear Zero-One Programs,
A. Mahendrarajah and F. Fiala, 331–334.

Three algorithms for the linear zero-one programming problem, Algorithms 341 and 449 of *Communications of the ACM* and Lawler and Bell’s algorithm, are compared and computational experience is summarized.

References


A Comparison of Algorithms for the Exact Solution of Linear Equations,
M.T. McClellan, 147–158.

A computing-time study is presented of several algorithms for the exact solution of
dense systems of linear equations with integer or dense polynomial coefficients. The
analytical computing times for rational Gauss elimination, exact division elimination (one
step and two step), and the modular algorithm are summarized and supplemented. Extensive empirical studies illustrate the superiority of the modular algorithm for completely
dense problems, in agreement with the analytical results. All algorithms were pro­
grammed in Fortran IV for the SAC-1 System, and all cases were run on a Univac 1108.

References
1. BAREISS, E.H. Sylvester's identity and multistep integer-preserving Gaussian elimination. Math
Comput. 22, 103 (July 1968), 565–578.
Appl. 10 (1972), 68–104.
3. BAREISS, E.H., and MAZUKELI, D. Multistep elimination over commutative rings. Rep. ANL-
4. BROWN, W.S. On Euclid's algorithm and the computation of polynomial greatest common divisors.
5. CABAY, S., and LAM, P.L. Congruence techniques for the solution of integer systems of linear equa­
tions. Submitted to a technical journal.
Madison, Wis., March 1971.
7. COLLINS, G.E. The SAC-1 System: An introduction and survey. Proc. 2nd Symp. Symbolic and
515–532.
12. GRISS, M.L. The algebraic solution of large sparse systems of linear equations using REDUCE 2.
Software 2, 1 (March 1976), 31–49.
14. HOWELL, J.A. and GREGORY, R.T. An algorithm for solving linear algebraic equations using res­
15. HOROWITZ, E. and SAHNl, S. On computing the exact determinant of matrices with polynomial


A Simulation Test Approach to the Evaluation of Nonlinear Optimization


A simulation test methodology has been developed to evaluate unconstrained nonlinear optimization computer algorithms. The test technique simulates problems optimization algorithms encounter in practice by employing a repertoire of problems representing various topographies (descending curved valleys, saddle points, ridges, etc.), dimensions, degrees of nonlinearity (e.g. linear to exponential) and minima, addressing them from various randomly generated initial approximations to the solution and recording their performances in the form of statistical summaries. These summaries, consisting of categorized results and statistical averages, are generated for each algorithm as tested over members of the problem set. The individual tests are composed of a series of runs from random starts over a member of the problem set. Descriptions of the test technique, test problem, and test results are provided.

References


Design and Testing of a Generalized Reduced Gradient Code for Nonlinear Programming,


An algorithm for solving nonlinear optimization problems is described along with its implementation as a Fortran program. Computational results are provided which compare its performance with that of other algorithms. The generalized reduced gradient (GRG) algorithm used is a nonlinear extension of the simplex method for linear programming. It is shown that the implementation is both robust and efficient.

References:


An Investigation of Romberg Quadrature,
G. Fairweather and P. Keast, 316–322.

A summary is made of a recent investigation of Romberg quadrature routines. In that study the performance of several existing Romberg routines on a set of test problems was examined and three new routines, constructed to exhibit the effects of sequence choice and cautious extrapolation, were described. Conclusions are reached concerning the form of an acceptable nonadaptive Romberg routine.

References:
15. PIESSENS, R. Remark on Algorithm 400. Modified Havie Integration. Comm. ACM 17, 6 (June 1974), 324.
Vol. 5

On Reporting Computational Experiments with Mathematical Software.

Many papers appearing in journals reporting computational experiments use computer generated evidence to compare or rank competing mathematical software techniques. Unfortunately, to date there have been no standards or guidelines indicating how computer experiments should be conducted or how the results should be presented. An initial attempt is made to rectify this situation, and a summary of important points which should be considered when writing or evaluating a paper in which computational results are reported is provided.

References
Computational Comparison of Eight Methods for the Maximum Network Flow Problem,

T-Y Cheung, 1-16.

There exist two approaches for solving the maximum network flow problem. In the first approach, flow is augmented along paths and is always conserved at the vertices. In the second approach, flow is augmented through layers and is sometimes not conserved at some of the vertices.

Many methods based on these two approaches are first briefly reviewed. A computational comparison is then presented of eight of the methods—depth-first search, breadth-first search, largest-augmentation, Dinic, layer-updating, Karzanov, Dinic-Karzanov, and Kinariwala-Rao. Problems with up to 1500 vertices and 7960 arcs have been tested. Computational results show that Dinic’s method is better than all of the other methods. However, for small-sized problems (up to 25 vertices and 200 arcs), the performances of the depth-first and breadth-first methods are comparable to Dinic’s method. Hence, with programming simplicity and memory space requirements taken into consideration, the former also seem to be a good choice for small problems.

References

A Comparative Study of Two Methods for Staircase Linear Programs,
J.K. Ho and E. Loute, 17-30.

This paper considers the important class of large-scale staircase linear programs arising from dynamic or multistage models. The two major approaches to large LPs—problem decomposition and basis factorization—are discussed. Two methods for staircase LPs, one from each approach, that have recently been developed and implemented are compared in both algorithmic and data-structural aspects. Computational results of an empirical comparison are presented. The study demonstrates that both special techniques can be more efficient than the direct simplex approach. It also identifies certain classes of problems for which a particular technique is especially promising.

References

A Note About Overhead Costs in ODE Solvers,

G.K. Gupta, 319–326.

It is shown that the number of function evaluations by itself is not an adequate measure of cost of solving ordinary differential equations because of the large overhead costs in Adams codes except when the function is very expensive to evaluate or when the codes being compared are similar. Because of these large overhead costs in Adams codes, it may be more economical to use a good Runge-Kutta code than an Adams code in many application environments.

References
A Test Problem Generator for Discrete Linear $L_1$ Approximation Problems,

Described here are the theoretical development and computer implementation of a
procedure that generates test problems for $L_1$ estimation of the linear model $y = X\beta + u$.
The generation procedure allows the user flexibility in specifying the problem dimensions,
the $L_1$ solution vector $\beta^*$, the distribution of the observed residuals $e$, as well as the
column rank, row repetitions, and degree of degeneracy of the matrix $X$. The user can
also specify the distributional form, mean, and variance for each independent variable.
An important feature of the generator is that any problem it creates is guaranteed to have
a unique solution $\beta$ whenever $X$ has full rank.

References
Algorithm 564: A Test Problem Generator for Discrete Linear $L_1$ Approximation Problems.


The algorithm given here is a complement to [1], where its theoretical development and implementation are described.

References

An Evaluation of Mathematical Software that Solves Nonlinear Least Squares Problems,

K.L. Hiebert, 1–16.

In this paper the "state of the art" of mathematical software that solves nonlinear least squares problems is evaluated. The evaluation is based on a comparison of 12 readily available FORTRAN codes. Theoretical and software aspects of the codes, as well as their performance on a carefully designed set of test problems, are used in the comparison.

References
A Comparison of Combination Generation Methods.
S.G. Akl. 42-45.

It is empirically shown that the algorithm of Mifsud is the fastest existing combination generator. This contradicts recent efficiency claims concerning other algorithms.

References
Testing Unconstrained Optimization Software,


Much of the testing of optimization software is inadequate because the number of test functions is small or the starting points are close to the solution. In addition, there has been too much emphasis on measuring the efficiency of the software and not enough on testing reliability and robustness. To address this need, we have produced a relatively large but easy-to-use collection of test functions and designed guidelines for testing the reliability and robustness of unconstrained optimization software.

References


Algorithm 566: Fortran Subroutines for Testing Unconstrained Optimization Software.
J.J. Moré, B.S. Garbow and K.E. Hillstrom, 136–140.
Abstract unavailable.
Evaluation of a Test Set for Stiff ODE Solvers,

In 1975 Enright, Hull, and Lindberg published a set of problems for the testing of codes for stiff ordinary differential equation (ODE) problems. It has been widely used since then. In this paper a critical evaluation of the test set is made, and changes are recommended that make the set a more useful tool. Comments about the difficulties of testing software in this area are made which may lead to more thoughtful use of the test set.

References


An Evaluation of Mathematical Software That Solves Systems of Nonlinear Equations,


The "state of the art" of mathematical software that solves systems of nonlinear equations is evaluated. The evaluation is based on a comparison of eight readily available FORTRAN codes. Theoretical and software aspects of the codes, as well as their performance on a carefully designed set of test problems, are used in the evaluation.

References
(Note. References [2] and [13] are not cited in text.)
1. BARRERA, P., and DENNIS, J. Private communications.
8. MINPACK Documentation Applied Mathematics Division, Argonne National Laboratory, Argonne, Ill.
Exploratory Data Analysis in a Study of the Performance of Nonlinear Optimization Routines,


Investigations into the comparative performances of mathematical software often involve collection and analysis of data under circumstances where the behavior of the software is not understood and unexpected results are likely to arise. The recently developed statistical techniques of exploratory data analysis are well suited to exposing important regularities of such data. Several of these techniques are explained and illustrated.

Nonlinear optimization algorithms are complicated by nature, and data on the performance of some of them provide an opportunity to apply exploratory techniques and other techniques of modern data analysis. The focus of this study is on one relatively small body of data from selected measurements by K.E. Hillstrom on five nonlinear optimization routines in solving one test problem, starting from each of twenty randomly chosen starting points.

The variability of performance across optimizers is described, and the effect of starting points is exposed. The concluding discussion examines some aspects of the design issues in studying behavior of mathematical software and related data analysis problems.

References
(Note. References [1, 5, 15, 20] are not cited in the text.)


15. RICE, J.R., and WHITE, J.S. Norms for smoothing and estimation. SIAM Rev. 6, 3 (July 1964), 243-256.


Vol. 9

Controlling the Defect in Existing Variable-Order Adams Codes for Initial-Value Problems,

P.M. Hanson and W.H. Enright, 71–97.

Variable-order, variable-step multistep methods based on Adams formulas have proved very effective in the numerical solution of nonstiff systems of ordinary differential equations. The user specifies an accuracy parameter and the method attempts to produce a solution consistent with this accuracy requirement. The relationship between the global error in the numerical solution and the prescribed accuracy requirement is problem dependent and very difficult to quantify precisely. On the other hand, most variable-order Adams methods provide the user with a piecewise polynomial approximation to the solution and use this to provide intermediate solutions at specified output points. Since the piecewise polynomial is differentiable at all but a finite number of points, one can define the defect of the numerical solution as the amount by which the piecewise polynomial fails to satisfy the differential equation. The maximum magnitude of the defect can then be used as a measure of the accuracy of the solution and, since the defect can be shown to be directly related to the local error, it provides a measure of accuracy that is relatively insensitive to the specific problem. In this study, the underlying piecewise polynomial approximates associated with two of the popular Adams codes are analyzed and assessed, and it is shown how the magnitude of the defect is related to the prescribed accuracy parameter. That current methods can keep the maximum magnitude of the defect bounded by a small multiple of the tolerance is also shown. Numerical results that verify the analysis on a variety of problems are presented.

References
(Note. Reference [8] is not mentioned in the text.)


Modeling Languages Versus Matrix Generators for Linear Programming,

R. Fourer, 143–183.

Linear optimization problems (linear programs) are expressed in one kind of form for human modelers, but in a quite different form for computer algorithms. Translation from the modeler’s form to the algorithm’s form is thus an unavoidable task in linear programming. Traditionally, this task of translation has been divided between human and computer, through the writing of computer programs known as matrix generators.

An alternative approach leaves almost all of the work of translation to the computer. Central to such an approach is a computer-readable modeling language that expresses a linear program in much the same way that a modeler does. It is argued that modeling languages should lead to more reliable application of linear programming at lower overall cost.

References

[Note. See also Appendix A for references that pertain to particular implementations of LP modeling-language features.]


18. JARVIS, R.J., and PAPACONSTADPOULOS, C. A Methodology for designing high level computer input systems for mathematical programming models. Reports Series, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, 1978.
30. ORCHARD-HAYS, W. Problems of symbology and recent experience (or, where improvements won't and may come from). In Large-Scale Linear Programming, G.B. Dantzig, M.A.H. Dempster, and M.J. Kallio (Eds.), International Institute of Applied Systems Analysis, Laxenburg, Austria, 1981, pp. 889-906.
A Computational Study of a Multiple-Choice Knapsack Algorithm,

The results from the computational testing of an algorithm designed to solve large-scale multiple-choice knapsack problems are described. Data list structures, sorting techniques, and fathoming criteria are investigated. It is determined that considerable storage space can be eliminated from the original implementation of the algorithm without affecting execution time. Also the insertion of a heap sort in the algorithm does provide a substantial reduction in computing time when solving the larger problems. Theoretical results relating to the multiple-choice knapsack are presented.

References
(Note. References [2, 14] are not cited in the text.)
The Multifrontal Solution of Indefinite Sparse Symmetric Linear Equations,
I.S. Duff and J.K. Reid. 302–325.

We extend the frontal method for solving linear systems of equations by permitting more than one front to occur at the same time. This enables us to develop code for general symmetric systems. We discuss the organization and implementation of a multifrontal code which uses the minimum-degree ordering and indicate how we can solve indefinite systems in a stable manner. We illustrate the performance of our code both on the IBM 3033 and on the CRAY-1.

References


A Performance Evaluation of Some Fortran Subroutines for the Solution of Stiff Oscillatory Ordinary Differential Equations.

P.W. Gaffney, 58–72.

This paper considers the problems that may arise when some existing FORTRAN codes are applied to the numerical solution of stiff oscillatory ordinary differential equations. By applying a selection of suitable software to a sequence of model problems, it is possible to identify the difficulties that may be encountered when the codes are to be used in more realistic applications. This paper highlights these difficulties and pays particular attention to the effects of using the software on different computers.

References

5. EHLE, B.L. A comparison of numerical methods for solving certain stiff ordinary differential equations. Report Number 70, University of Victoria, Victoria, B.C., Canada (1972).


25. WOLFBRANDT, A. A study of Rosenbrock processes with respect to order conditions and stiff stability. Rep. 77,01 R, Chalmers University of Technology and the University of Göteborg, Department of Computer Sciences (1977).
Randomly Generated Test Problems for Positive Definite Quadratic Programming,
M.L. Lenard and M. Minkoff, 86–96.

A procedure is described for randomly generating positive definite quadratic pro-
gramming test problems. The test problems are constructed in the form of linear least-
squares problems subject to linear constraints. The probability measure for the problems
so generated is invariant under orthogonal transformations. The procedure allows the
user to specify the size of the least-squares problem (number of unknown parameters,
number of observations, and number of constraints), the relative magnitude of the residu-
als, the condition number of the Hessian matrix of the objective function, and the struc-
ture of the feasible region (the number of equality constraints and of inequalities which
will be active at the feasible starting point and at the optimal solution). An example is
given illustrating how these problems can be used to evaluate the performance of a
software package.

References
   (ZAMP) 30 (1979), 148-158.
2. CLARK, N.W. Applied Mathematics Division system/360 library subroutine RANF: ANL G5525.
   Argonne National Laboratory, Argonne, Ill., 1967.
   sium on Mathematical Programming, H.W. Kuhn (Ed.), Princeton University Press, Princeton, N.J.,
4. CRANE, R.L., GARBOW, B.S., HILLSTROM, K.E., and MINKOFF, M. LCLSQ: An implementa-
   tion of an algorithm for linearly constrained linear least-squares problems. Mathematics and comput-
5. CROWDER, H.P., DEMBO, R.S., and MULVEY, J.M. Reporting computational experiments in
7. HANDSON, R.J. and HASKELL, K.H. Algorithm 5XX. Two algorithms for the linearly constrained
8. HOCK, W. and SCHITTOKOWSKI, K. Test Examples for Nonlinear Programming Codes. Lecture
9. LASDON, L.S., WAREN, A.D., JAIN, A., and RATNER, M. Design and testing of a generalized
10. LAWSON, C.L. and HANSON, R.J. Solving Least Squares Problems. Prentice-Hall, Englewood
11. LENARD, M.L. A computational study of active set strategies in nonlinear programming with linear
    249-264.
13. MICHAELS, W.M. and O'NEILL, R.P. A mathematical programming generator MPGENR. ACM
    Trans. Math. Softw. 6, 1 (March 1980), 31-44.
14. MINKOFF, M. Methods for evaluating nonlinear programming software. In Nonlinear Programming
    519-548.
    ACM 8, 3 (March 1965), 113.


A Test Problem Generator for Large-Scale Unconstrained Optimization,

A test problem generator for large-scale unconstrained optimization is described. It permits
the generation of poorly or well-conditioned problems of arbitrary size, derived from nonlinear
network flow models. An eigenvalue analysis provides bounds on the condition number of the
Hessian of the objective function and an example of an efficient preconditioner, using these
bounds, is outlined.

Reference
2. DEMBO, R.S. A set of geometric programming test problems and their solutions. Math. Program. 10
(1976), 192–214.
3. DEMBO, R.S., and STEIHAUGH, T. Truncated-Newton algorithms for large-scale unconstrained optimiza-
4. FIACCO, A.V., and MCCORMICK, G.P. Nonlinear Programming: Sequential Unconstrained Minimization
6. SCHITTOWSKI, K. Nonlinear Programming Codes—Information Tests and Performance. Lecture Notes
An Evaluation of Some New Cyclic Linear Multistep Formulas for Stiff ODEs,


We evaluate several sets of cyclic linear multistep formulas (CLMFs). One of these sets was derived by Tischer and Sacks-Davis. Three new sets of formulas have been derived and we present their characteristics.

The formulas have been evaluated by comparing the performance of four versions of a code which implements CLMFs. The four versions are very similar and each version implements one of the sets of CLMFs being studied. We compare the performance of these codes with that of a widely used code, LSODE. One of the new sets of CLMFs is not only much more efficient in solving stiff problems that have a Jacobian with eigenvalues close to the imaginary axis but is almost as efficient as LSODE in solving other problems. This is a significant improvement over the only other CLMF code available, STINT from Tendler, Bickart, and Pice!.

References
Inverse Pairs of Test Matrices.

W.S. Ericksen, 302–304.

Algorithms that are readily programmable are provided for constructing inverse pairs of matrices with elements in a field, a division ring, or a ring.

References
Fixed versus Variable Order Runge-Kutta,

Popular codes for the numerical solution of nonstiff ordinary differential equations (ODEs) are based on a (fixed order) Runge-Kutta method, a variable order Adams method, or an extrapolation method. Extrapolation can be viewed as a variable order Runge-Kutta method. It is plausible that variation of order could lead to a much more efficient Runge-Kutta code, but numerical comparisons have been contradictory.

We reconcile previous comparisons by exposing differences in testing methodology and incompatibilities of the implementations tested. An experimental Runge-Kutta code is compared to a state-of-the-art extrapolation code. With some qualifications, the extrapolation code shows no advantage. Extrapolation does not appear to be a particularly effective way to vary the order of Runge-Kutta methods. Although an acceptable way to solve nonstiff problems, our tests raise the question as to whether there is any point in pursuing it as a separate method.

References


Algorithm 645: Subroutines for Testing Programs that Compute the Generalized Inverse of a Matrix,


Abstract unavailable.

Reference
Two FORTRAN Packages for Assessing Initial Value Methods,

We present a discussion and description of a collection of FORTRAN routines designed to aid in the assessment of initial value methods for ordinary differential equations. Although the overall design characteristics are similar to those of earlier testing packages [2, 6] that were used for the comparison of methods [5, 7], the details and objectives of the current collection are quite different. Our principal objective is the development of testing tools that can be used to assess the efficiency and reliability of a standard numerical method without requiring significant modifications to the method and without the tools themselves affecting the performance of the method.

References
Algorithm 648: NSDTST and STDTST: Routines for Assessing the Performance of Initial Value Solvers,


Abstract unavailable.

References


Generation of Large-Scale Quadratic Programs for Use as Global Optimization Test Problems.

P.M. Pardalos, 133–137.

A method is presented for the generation of test problems for global optimization algorithms. Given a bounded polyhedron in $\mathbb{R}^n$ and a vertex, the method constructs non-convex quadratic functions (concave or indefinite) whose global minimum is attained at the selected vertex. The construction requires only the use of linear programming and linear systems of equations.

References


Sparse Matrix Test Problems,


We describe the Harwell-Boeing sparse matrix collection, a set of standard test matrices for sparse matrix problems. Our test set comprises problems in linear systems, least squares, and eigenvalue calculations from a wide variety of scientific and engineering disciplines. The problems range from small matrices, used as counter-examples to hypotheses in sparse matrix research, to large test cases arising in large-scale computation. We offer the collection to other researchers as a standard benchmark for comparative studies of algorithms. The procedures for obtaining and using the test collection are discussed. We also describe the guidelines for contributing further test problems to the collection.

References

Performance Evaluation of Programs for Certain Bessel Functions,


This paper presents methods for performance evaluation of the $K$ Bessel functions. Accuracy estimates are based on comparisons involving the multiplication theorem. Some ideas for checking robustness are also given. The techniques used here are easily extended to the $Y$ Bessel functions and, with a little more effort, to the $I$ and $J$ functions. Details on a specific implementation for testing the $K$ Bessel functions are included.

References


2. CAMPBELL, J.B. A FORTRAN IV subroutine for the modified bessel function sof the third kind of real order and real argument. Repor NRC/ERB-925, National Research Council, Canada, 1980.


Numerical Experience with Sequential Quadratic Programming Algorithms for Equality Constrained Nonlinear Programming,


Computational experience is given for a sequential quadratic programming algorithm when Lagrange multiplier estimates, Hessian approximations, and merit functions are varied to test for computational efficiency. Indications of areas for further research are given.

References


A Comparison of Adaptive Refinement Techniques for Elliptic Problems.

W.F. Mitchell, 326–347.

Adaptive refinement has proved to be a useful tool for reducing the size of the linear system of equations obtained by discretizing partial differential equations. We consider techniques for the adaptive refinement of triangulations used with the finite element method with piecewise linear functions. Several such techniques that differ mainly in the method for dividing triangles and the method for indicating which triangles have the largest error have been developed. We describe four methods for dividing triangles and eight methods for indicating errors. Angle bounds for the triangle division methods are compared. All combinations of triangle divisions and error indicators are compared in a numerical experiment using a population of eight test problems with a variety of difficulties (peaks, boundary layers, singularities, etc.). The comparison is based on the L-infinity norm of the error versus the number of vertices. It is found that all of the methods produced asymptotically optimal grids and that the number of vertices needed for a given error rarely differs by more than a factor of two.

References


Accurate and Efficient Testing of the Exponential and Logarithm Functions,


Table-driven techniques can be used to test highly accurate implementations of EXP and LOG. The largest error observed in EXP and LOG accurately to within 1/500 unit in the last place are reported in our tests. Methods to verify the tests' reliability are discussed. Results of applying the tests to our own as well as to a number of other implementations of EXP and LOG are presented.

References

Performance Evaluation of Programs for the Error and Complementary Error Functions,

W.J. Cody.

This paper presents methods for performance evaluation of computer programs for the functions $\text{erf}(x)$, $\text{erfc}(x)$, and $e^{x^2}\text{erfc}(x)$. Accuracy estimates are based on comparisons using power series expansions and an expansion in the repeated integrals of $\text{erfc}(x)$. Some suggestions for checking robustness are also given. Details of a specific implementation of a test program are included.

References


We describe a FORTRAN implementation, and some related problems, of Talbot’s method which numerically solves the inversion problem of almost arbitrary Laplace transforms by means of special contour integration.

The basic idea is to take into account computer precision to derive a special contour where integration will be carried out.

References
Numerical Comparisons of Explicit Runge-Kutta Pairs of Orders Four through Eight,

P.W. Sharp.

Abstract unavailable.