Numerical Methods for Unconstrained Optimization and Nonlinear Equations

J. E. Dennis, Jr.
Rice University
Houston, Texas

Robert B. Schnabel
University of Colorado
Boulder, Colorado

siam
Society for Industrial and Applied Mathematics
Philadelphia

Contents

References

Aasen, J. O. (1971), "On the reduction of a symmetric matrix to tridiagonal form," *BIT* **11**, 233–242.

Aho, A. V., J. E. Hopcroft, and J. D. Ullman (1974). *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, Mass.

Allgower, E., and K. Georg (1980), "Simplicial and continuation methods for approximating fixed points and solutions to systems of equations," *SIAM Review* **22**, 28–85.

Armijo, L., (1966), "Minimization of functions having Lipschitz-continuous first partial derivatives," *Pacific J. Math.* **16**, 1–3.

Avriel, M. (1976), *Nonlinear Programming: Analysis and Methods*, Prentice-Hall, Englewood Cliffs, N.J.

Bard, Y. (1970), *Nonlinear Parameter Estimation*, Academic Press, New York.

Barnes, J. (1965), "An algorithm for solving nonlinear equations based on the secant method," *Comput. J.* **8**, 66–72.

Bartels, R., and A. Conn. (1982), "An approach to nonlinear l_1 data fitting," in *Numerical Analysis, Cocoyoc* 1981, ed. by J. P. Hennart, Springer Verlag. Lecture Notes in Math. 909, 48–58.

Bates, D. M., and D. G. Watts (1980), "Relative curvature measures of nonlinearity," *J. Roy. Statist. Soc. Ser. B 42*, 1–25, 235.

Beale, E. M. L. (1977), "Integer programming," in *The State of the Art in Numerical Analysis*, D. Jacobs, ed., Academic Press, London, 409–448.

Beaton, A. E., and J. W. Tukey (1974), "The fitting of power series, meaning polynomials, illustrated on hand-spectroscopic data," *Technometrics* **16**, 147–192.

Boggs, P. T., and J. E. Dennis, Jr. (1976), "A stability analysis for perturbed nonlinear iterative methods," *Math. Comp.* **30**, 1–17.

Brent, R. P. (1973), *Algorithms For Minimization Without Derivatives*, Prentice-Hall, Englewood Cliffs, N.J.

Brodlie, K. W. (1977), "Unconstrained minimization," in *The State of the Art in Numerical Analysis*, D. Jacobs, ed., Academic Press, London, 229–268.

Broyden, C. G. (1965), "A class of methods for solving nonlinear simultaneous equations," *Math. Comp.* **19**, 577–593.

Broyden, C. G. (1969), "A new double-rank minimization algorithm," *AMS Notices* **16**, 670.

Broyden, C. G. (1970), "The covergence of a class of double-rank minimization algorithms," Parts I and II, *J.I.M.A.* **6**, 76–90, 222–236.

Broyden, C. G. (1971), "The convergence of an algorithm for solving sparse nonlinear systems," *Math. Comp.* **25**, 285–294.

Broyden, C. G., J. E. Dennis, Jr., and J. J. Moré (1973), "On the local and superlinear covergence of quasi-Newton methods," *J.I.M.A.* **12**, 223–246.

Bryan, C. A. (1968), "Approximate solutions to nonlinear integral equations," *SIAM J. Numer. Anal.* **5**, 151–155.

Buckley, A. G. (1978), "A combined conjugate gradient quasi-Newton minimization algorithm," *Math. Prog.* **15**, 200–210.

Bunch, J. R., and B. N. Parlett (1971), "Direct methods for solving symmetric indefinite systems of linear equations," *SIAM J. Numer. Anal.* **8**, 639–655.

Cline, A. K., C. B. Moler, G. W. Stewart, and J. H. Wilkinson (1979), "An estimate for the condition number of a matrix," *SIAM J. Numer. Anal.* **16**, 368–375.

Coleman, T. F., and J. J. Moré (1983), "Estimation of sparse Jacobian matrices and graph coloring problems," *SIAM J. Numer. Anal.* **20**, 187–209.

Coleman, T. F., and J. J. Moré (1982), "Estimation of sparse Hessian matrices and graph coloring problems," Argonne National Labs, Math–C. S. Div. TM-4.

Conte, S. D., and C. de Boor (1980), *Elementary Numerical Analysis: An Algorithmic Approach*, 3d ed., McGraw-Hill, New York.

Curtis, A., M. J. D. Powell, and J. K. Reid (1974), "On the estimation of sparse Jacobian matrices," *J.I.M.A.* **13**, 117–120.

Dahlquist, G., A. Björck, and N. Anderson (1974), *Numerical Methods*, Prentice-Hall, Englewood Cliffs, N.J.

Davidon, W. C. (1959), "Variable metric methods for minimization," Argonne National Labs Report ANL-5990.

Davidon, W. C. (1975), "Optimally conditioned optimization algorithms without line searches," *Math. Prog.* **9**, 1–30.

Dembo, R. S., S. C. Eisenstat, and T. Steihaug (1982). "Inexact Newton methods," *SIAM J. Numer. Anal.* **19**, 400–408.

Dennis, J. E., Jr. (1971), "Toward a unified convergence theory for Newton-like methods," in *Nonlinear Functional Analysis and Applications*, L. B. Rall, ed., Academic Press, New York, 425–472.

Dennis, J. E., Jr. (1977), "Nonlinear least squares and equations," in *The State of the Art in Numerical Analysis*, D. Jacobs, ed., Academic Press, London, 269–312.

Dennis, J. E., Jr. (1978), "A brief introduction to quasi-Newton methods," in Numerical Analysis, G. H. Golub and J. Oliger, eds., AMS, Providence, R.I., 19–52.

Dennis, J. E., Jr., D. M. Gay, and R. E. Welsch (1981a), "An adaptive nonlinear least-squares algorithm," *TOMS* **7**, 348–368.

Dennis, J. E., Jr., D. M. Gay, and R. E. Welsch (1981b), "Algorithm 573 NL2SOL—An adaptive nonlinear least-squares algorithm [E4]," *TOMS* **7**, 369–383.

Dennis, J. E., Jr., and E. S. Marwil (1982), "Direct secant updates of matrix factorizations," *Math. Comp.* **38**, 459–474.

Dennis, J. E., Jr., and H. H. W. Mei (1979), "Two new unconstrained optimization algorithms which use function and gradient values," *J. Optim. Theory Appl.* **28**, 453–482.

Dennis, J. E., Jr., and J. J. Moré (1974), "A characterization of superlinear convergence and its application to quasi-Newton methods," *Math. Comp.* **28**, 549–560.

Dennis, J. E., Jr., and J. J. Moré (1977), "Quasi-Newton methods, motivation and theory," *SIAM Review* **19**, 46–89.

Dennis, J. E., Jr., and R. B. Schnabel (1979), "Least change secant updates for quasi-Newton methods," *SIAM Review* **21**, 443–459.

Dennis, J. E., Jr., and R. A. Tapia (1976), "Supplementary terminology for nonlinear iterative methods," *SIGNUM Newsletter* **11**:4, 4–6.

Dennis, J. E., Jr., and H. F. Walker (1981), "Convergence theorems for least-change secant update methods," *SIAM J. Numer. Anal.* **18**, 949–987, **19**, 443.

Dennis, J. E., Jr., and H. F. Walker (1983), "Sparse secant update methods for problems with almost sparse Jacobians," in preparation.

Dixon, L. C. W. (1972a), "Quasi-Newton family generate identical points," Parts I and II, *Math. Prog.* **2**, 383–387, 2nd *Math. Prog.* **3**, 345–358.

Dixon, L. C. W. (1972b), "The choice of step length, a crucial factor in the performance of variable metric algorithms," in *Numerical Methods for Non-linear Optimization*, F. Lootsma, ed., Academic Press, New York, 149–170.

Dixon, L. C. W., and G. P. Szegö (1975, 1978), *Towards Global Optimization*, Vols. 1, 2, North-Holland, Amsterdam.

Dongarra, J. J., J. R. Bunch, C. B. Moler, and G. W. Stewart (1979), *LINPACK Users Guide*, SIAM Publications, Philadelphia.

Fletcher, R. (1970), "A new approach to variable metric algorithms," *Comput. J.* **13**, 317–322.

Fletcher, R. (1980), *Practical Methods of Optimization*, Vol. 1, *Unconstrained Optimization*, John Wiley & Sons, New York.

Fletcher, R., and M. J. D. Powell (1963), "A rapidly convergent descent method for minimization," *Comput. J.* **6**, 163–168.

Ford, B. (1978), "Parameters for the environment for transportable numerical software," *TOMS* **4**, 100–103.

Fosdick, L. ed. (1979), *Performance Evaluation of Numerical Software*, North-Holland, Amsterdam.

Frank, P., and R. B. Schnabel (1982), "Calculation of the initial Hessian approximation in secant algorithms," in preparation.

Gander, W. (1978), "On the linear least squares problem with a quadratic constraint," Stanford Univ. Computer Science Tech. Rept. STAN-CS-78-697, Stanford, Calif.

Gander, W. (1981), "Least squares with a quadratic constraint," *Numer. Math.* **36**, 291–307. [This is an abbreviated version of Gander (1978).]

Garfinkel, R. S., and G. L. Nemhauser (1972), *Integer Programming*, John Wiley & Sons, New York.

Gay, D. M. (1979), "Some convergence properties of Broyden's method," *SIAM J. Numer. Anal.*. **16**, 623–630.

Gay, D. M. (1981), "Computing optimal locally constrained steps," *SIAM J. Sci. Stat. Comp.* **2**, 186–197.

Gay, D. M., and R. B. Schnabel (1978), "Solving systems of nonlinear equations by Broyden's method with projected updates," in *Nonlinear Programming 3*, O. Mangasarian, R. Meyer, and S. Robinson, eds., Academic Press, New York, 245–281.

Gill, P. E., G. H. Golub, W. Murray, and M. A. Saunders (1974), "Methods for modifying matrix factorizations," *Math. Comp.* **28**, 505–535.

Gill, P. E., and W. Murray (1972), "Quasi-Newton methods for unconstrained optimization," *J.I.M.A.* **9**, 91–108.

Gill, P. E., W. Murray and M. H. Wright (1981), *Practical Optimization*, Academic Press, London.

Goldfarb, D. (1976), "Factorized variable metric methods for unconstrained optimization," *Math. Comp.* **30**, 796–811.

Goldfarb, D. (1970), "A family of variable metric methods derived by variational means," *Math. Comp.* **24**, 23–26.

Goldfeldt, S. M., R. E. Quandt, and H. F. Trotter (1966), "Maximization by quadratic hill-climbing," *Econometrica* **34**, 541–551.

Goldstein, A. A. (1967), *Constructive Real Analysis*, Harper & Row, New York.

Golub, G. H., and V. Pereyra (1973), "The differentiation of pseudo-inverse and non-linear least squares problems whose variables separate," *SIAM J. Numer. Anal.* **10**, 413–432.

Golub, G. H., and C. Van Loan (1983), *Matrix Computations*, the Johns Hopkins University Press.

Greenstadt, J. L. (1970), "Variations on variable-metric methods," *Math. Comp.* **24**, 1–22.

Griewank, A. O., (1982), "A short proof of the Dennis-Schnabel theorem," *B.I.T.* *22*, 252–256.

Griewank, A. O., and Ph. L. Toint (1982a), "Partitioned variable metric updates for large sparse optimization problems," *Numer. Math.* **39**, 119–37.

Griewank, A. O., and Ph. L. Toint (1982b), "Local convergence analysis for partitioned quasi-Newton updates in the Broyden class," to appear in *Numer. Math.*

Griewank, A. O., and Ph. L. Toint (1982c), "On the unconstrained optimization of partially separable functions," in *Nonlinear Optimization*, M. J. D. Powell, ed., Academic Press, London.

Hamming, R. W. (1973), *Numerical Methods for Scientists and Engineers*, 2 ed., McGraw-Hill, New York.

Hebden, M. D. (1973), "An algorithm for minimization using exact second derivatives," Rept. TP515, A.E.R.E., Harwell, England.

Hestenes, M. R. (1980), *Conjugate-Direction Methods In Optimization*, Springer Verlag, New York.

Hiebert, K. L. (1982), "An evaluation of mathematical software that solves systems of nonlinear equations," *TOMS* **8**, 5–20.

Huber, P. J. (1973), "Robust regression: asymptotics, conjectures, and Monte Carlo," *Anals of Statistics* **1**, 799–821.

Huber, P. J. (1981), *Robust Statistics*, John Wiley & Sons, New York.

Johnson, G. W., and N. H. Austria (1983), "A quasi-Newton method employing direct secant updates of matrix factorizations," *SIAM J. Numer. Anal.* **20**, 315–325.

Kantorovich, L. V. (1948), "Functional analysis and applied mathematics," *Uspehi Mat. Nauk.* **3**, 89–185; transl. by C. Benster as N.B.S. Rept. 1509, Washington, D. C., 1952.

Kaufman, L. C. (1975), "A variable projection method for solving separable nonlinear least squares problems," *BIT* **15**, 49–57.

Lawson, C. L., R. J. Hanson, D. R. Kincaid, and F. T. Krogh (1979), "Basic linear algebra subprograms for Fortran usage," *ACM TOMS* **5**, 308–323.

Levenberg, K. (1944), "A method for the solution of certain problems in least squares," *Quart. Appl. Math.* **2**, 164–168.

Marquardt, D. (1963), "An algorithm for least-squares estimation of nonlinear parameters," *SIAM J. Appl. Math.* **11**, 431–441.

Marwil, E. S. (1978), "Exploiting sparsity in Newton-type methods," Cornell Applied Math. Ph.D. Thesis.

Marwil, E. S. (1979), "Convergence results for Schubert's method for solving sparse nonlinear equations," *SIAM J. Numer. Anal.* **16**, 588–604.

Moré, J. J. (1977), "The Levenberg-Marquardt algorithm: implementation and theory," in *Numerical Analysis*, G. A. Watson, ed., Lecture Notes in Math. 630, Springer Verlag, Berlin, 105–116.

Moré, J. J., B. S. Garbow, and K. E. Hillstrom (1980), "User guide for MINPACK-1," Argonne National Labs Report ANL-80-74.

Moré, J. J., B. S. Garbow, and K. E. Hillstrom (1981a), "Testing unconstrained optimization software," *TOMS* 7, 17–41.

Moré, J. J., B. S. Garbow, and K. E. Hillstrom (1981b), "Fortran subroutines for testing unconstrained optimization software," *TOMS* **7**, 136–140.

Moré, J. J., and D. C. Sorensen (1979), "On the use of directions of negative curvature in a modified Newton method," *Math. Prog.* **16**, 1–20.

Murray, W. (1972), *Numerical Methods for Unconstrained Optimization*, Academic Press, London.

Murray, W., and M. L. Overton (1980), "A projected Lagrangian algorithm for nonlinear minimax optimization," *SIAM J. Sci. Statist. Comput.* **1**, 345–370.

Murray, W., and M. L. Overton (1981), "A projected Lagrangian algorithm for nonlinear l_1 optimization," *SIAM J. Sci. Statist. Comput.* **2**, 207–224.

Nelder, J. A., and R. Mead (1965), "A simplex method for function minimization," *Comput. J.* **7**, 308–313.

Oren, S. S., (1974), "On the selection of parameters in self-scaling variable metric algorithms," *Math. Prog.* **7**, 351–367.

Ortega, J. M., and W. C. Rheinboldt (1970), *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York.

Osborne, M. R. (1976), "Nonlinear least squares—the Levenberg algorithm revisited," *J. Austral. Math. Soc.* **19** (Series B), 343–357.

Powell, M. J. D. (1970a), "A hybrid method for nonlinear equations," in *Numerical Methods for Nonlinear Algebraic Equations*, P. Rabinowitz, ed., Gordon and Breach, London, 87–114.

Powell, M. J. D. (1970b), "A new algorithm for unconstrained optimization," in *Nonlinear Programming*, J. B. Rosen, O. L. Mangasarian, and K. Ritter, eds., Academic Press, New York, 31–65.

Powell, M. J. D. (1975), "Convergence properties of a class of minimization algorithms," in *Nonlinear Programming* **2**, O. Mangasarian, R. Meyer, and S. Robinson, eds. Academic Press, New York, 1–27.

Powell, M. J. D. (1976), "Some global convergence properties of a variable metric algorithm without exact line searches," in *Nonlinear Programming*, R. Cottle and C. Lemke, eds., AMS, Providence, R. I., 53–72.

Powell, M. J. D. (1981), "A note on quasi-Newton formulae for sparse second derivative matrices," *Math. Prog.* **20**, 144–151.

Powell, M. J. D., and Ph. L. Toint (1979), "On the estimation of sparse Hessian matrices," *SIAM J. Numer. Anal.* **16**, 1060–1074.

Pratt, J. W. (1977), "When to stop a quasi-Newton search for a maximum likelihood estimate," Harvard School of Business WP 77-16.

Reid, J. K. (1973), "Least squares solution of sparse systems of non-linear equations by a modified Marquardt algorithm," in *Proc. NATO Conf. at Cambridge, July 1972*, North-Holland, Amsterdam, 437–445.

Reinsch, C. (1971), "Smoothing by spline functions, II," *Numer. Math.* **16**, 451–454.

Schnabel, R. B. (1977), "Analysing and improving quasi-Newton methods for unconstrained optimization," Ph.D. Thesis, Cornell Computer Science TR-77-320.

Schnabel, R. B. (1982), "Convergence of quasi-Newton updates to correct derivative values," in preparation.

Schnabel, R. B., B. E. Weiss, and J. E. Koontz (1982), "A modular system of algorithms for unconstrained minimization," Univ. Colorado Computer Science, TR CU-CS-240-82.

Schubert, L. K. (1970), "Modification of a quasi-Newton method for nonlinear equations with a sparse Jacobian," *Math. Comp.* **24**, 27–30.

Shanno, D. F. (1970), "Conditioning of quasi-Newton methods for function minimization," *Math. Comp.* **24**, 647–657.

Shanno, D. F. (1978), "Conjugate-gradient methods with inexact searches," *Math. of Oper. Res.* **3**, 244–256.

Shanno, D. F. (1980), "On the variable metric methods for sparse Hessians," *Math. Comp.* **34**, 499–514.

Shanno, D. F., and K. H. Phua (1978a), "Matrix conditioning and nonlinear optimization," *Math. Prog.* **14**, 145–160.

Shanno, D. F., and K. H. Phua (1978b), "Numerical comparison of several variable metric algorithms," *J. Optim. Theory Appl.* **25**, 507–518.

Shultz, G. A., R. B. Schnabel, and R. H. Byrd (1982), "A family of trust region based algorithms for unconstrained minimization with strong global convergence properties," Univ. Colorado Computer Science TR CU-CS-216-82.

Sorensen, D. C. (1977), "Updating the symmetric indefinite factorization with applications in a modified Newton's method," Ph.D. Thesis, U. C. San Diego, Argonne National Labs Report ANL-77-49.

Sorensen, D. C. (1982), "Newton's method with a model trust region modification," *SIAM J. Numer. Anal.* **19**, 409–426.

Steihaug, T. (1981), "Quasi-Newton methods for large scale nonlinear problems," Ph.D. Thesis, Yale University.

Stewart, G. W., III (1967), "A modification of Davidon's method to accept difference approximations of derivatives," *J. ACM* **14**, 72–83.

Stewart, G. W., III (1973), *Introduction to Matrix Computations*, Academic Press, New York.

Strang, G. (1976), *Linear Algebra and Its Applications*, Academic Press, New York.

Toint, Ph. L. (1977), "On sparse and symmetric matrix updating subject to a linear equation," *Math. Comp.* **31**, 954–961.

Toint, Ph. L. (1978), "Some numerical results using a sparse matrix updating formula in unconstrained optimization," *Math. Comp.* **32**, 839–851.

Toint, Ph. L. (1981), "A sparse quasi-Newton update derived variationally with a non-diagonally weighted Frobenius norm," *Math. Comp.* **37**, 425–434.

Vandergraft, J. S. (1978), *Introduction to 'Numerical Computations*, Academic Press, New York.

Wilkinson, J. H. (1963), *Rounding Errors in Algebraic Processes*, Prentice-Hall, Englewood Cliffs, N.J.

Wilkinson, J. H. (1965), *The Algebraic Eigenvalue Problem*, Oxford University Press, London.

Wolfe, P. (1969), "Convergence conditions for ascent methods," *SIAM Review* **11**, 226–235.

Wolfe, P. (1971), "Convergence conditions for ascent methods. II: Some corrections," *SIAM Review* **13**, 185–188.

Zirilli, F. (1982), "The solution of nonlinear systems of equations by second order systems of o.d.e. and linearly implicit A-stable techniques," *SIAM J. Numer. Anal.* **19**, 800–815.

Stopping, Scaling, and Testing

this chapter we discuss three issues that are peripheral to the basic math-
atical considerations in the solution of nonlinear equations and mini-
ization problems, but essential to the computer solution of actual problems.
e first is how to adjust for problems that are badly scaled in the sense that
e dependent or independent variables are of widely differing magnitudes.
e second is how to determine when to stop the iterative algorithms in
ite-precision arithmetic. The third is how to debug, test, and compare non-
ear algorithms.

1 SCALING

important consideration in solving many "real-world" problems is that
me dependent or independent variables may vary greatly in magnitude. For
ample, we might have a minimization problem in which the first indepen-
nt variable, x_1, is in the range $[10^2, 10^3]$ meters and the second, x_2, is in the
nge $[10^{-7}, 10^{-6}]$ seconds. These ranges are referred to as the *scales* of the
spective variables. In this section we consider the effect of such widely dis-
rate scales on our algorithms.

One place where scaling will effect our algorithms is in calculating terms
ch as $\|x_+ - x_c\|_2$, which we used in our algorithms in Chapter 6. In the
ove example, any such calculation will virtually ignore the second (time)

variable. However, there is an obvious remedy: *rescale the independent variables*; that is, change their units. For example, if we change the units of x_1 to kilometers and x_2 to microseconds, then both variables will have range $[10^{-1}, 1]$ and the scaling problem in computing $\|x_+ - x_c\|_2$ will be eliminated. Notice that this corresponds to changing the independent variable to $\hat{x} = D_x x$, where D_x is the diagonal scaling matrix

$$D_x = \begin{bmatrix} 10^{-3} & 0 \\ 0 & 10^6 \end{bmatrix}. \tag{7.1.1}$$

This leads to an important question. Say we transform the units of our problem to $\hat{x} = D_x x$, or more generally, transform the variable space to $\hat{x} = Tx$, where $T \in \mathbb{R}^{n \times n}$ is nonsingular, calculate our global step in the new variable space, and then transform back. Will the resultant step be the same as if we had calculated it using the same globalizing strategy in the old variable space? The surprising answer is that the Newton step is unaffected by this transformation but the steepest-descent direction is changed, so that a line-search step in the Newton direction is unaffected by a change in units, but a trust region step may be changed.

To see this, consider the minimization problem and let us define $\hat{x} = Tx$, $\hat{f}(\hat{x}) = f(T^{-1}\hat{x})$. Then it is easily shown that

$$\nabla\hat{f}(\hat{x}) = T^{-T}\,\nabla f(x), \qquad \nabla^2\hat{f}(\hat{x}) = T^{-T}\,\nabla^2 f(x)T^{-1},$$

so that the Newton step and steepest-descent direction in the new variable space are

$$\hat{s}^N = -(T^{-T}\,\nabla^2 f(x)T^{-1})^{-1}(T^{-T}\,\Delta f(x)) = -T\,\nabla^2 f(x)^{-1}\,\nabla f(x),$$
$$\hat{s}^{SD} = -T^{-T}\,\nabla f(x),$$

or, in the old variable space,

$$s^N = T^{-1}\hat{s}^N = -\nabla^2 f(x)^{-1}\,\nabla f(x),$$
$$s^{SD} = T^{-1}\hat{s}^{SD} = -T^{-1}T^{-T}\,\nabla f(x) = -(T^T T)^{-1}\,\nabla f(x).$$

These conclusions are really common sense. The Newton step goes to the lowest point of a quadratic model, which is unaffected by a change in units of x. (The Newton direction for systems of nonlinear equations is similarly unchanged by transforming the independent variable.) However, determining which direction is "steepest" depends on what is considered a unit step in each direction. The steepest-descent direction makes the most sense if a step of one unit in variable direction x_i has about the same relative length as a step of one unit in any other variable direction x_j.

For these reasons, we believe the preferred solution to scaling problems is for the user to choose the units of the variable space so that each component of x will have roughly the same magnitude. However, if this is troublesome, the *equivalent* effect can be achieved by a transformation in the algorithm of the variable space by a corresponding diagonal scaling matrix D_x. This is the scaling strategy on the independent variable space that is implemented in our algorithms. All the user has to do is set D_x to correspond to the desired change in units, and then the algorithms operate as if they were working in the transformed variable space. The algorithms are still written in the original variable space, so that an expression like $\| x_+ - x_c \|_2$ becomes $\| D_x(x_+ - x_c) \|_2$ and the steepest-descent and hook steps become

$$x_+ := x_c - \lambda D_x^{-2} \nabla f(x_c),$$
$$s(\mu) := -(H_c + \mu D_x^2)^{-1} \nabla f(x_c),$$

respectively (see Exercise 3). The Newton direction is unchanged, however, as we have seen.

The positive diagonal scaling matrix D_x is specified by the user on input by simply supplying n values typx_i, $i = 1, \ldots, n$, giving "typical" magnitudes of each x_i. Then the algorithm sets $(D_x)_{ii} = (\text{typ}x_i)^{-1}$, making the magnitude of each transformed variable $\hat{x}_i = (D_x)_{ii}\, x_i$ about 1. For instance, if the user inputs typ$x_1 = 10^3$, typ$x_2 = 10^{-6}$ in our example, then D_x will be (7.1.1). If no scaling of x_i is considered necessary, typx_i should be set to 1. Further instructions for choosing typx_i are given in Guideline 2 in the appendix. Naturally, our algorithms do not store the diagonal matrix D_x, but rather a vector S_x (S stands for scale), where $(S_x)_i = (D_x)_{ii} = (\text{typ}x_i)^{-1}$.

The above scaling strategy is not always sufficient; for example, there are rare cases that need dynamic scaling because some x_i varies by many orders of magnitude. This corresponds to using D_x exactly as in all our algorithms, but recalculating it periodically. Since there is little experience along these lines, we have not included dynamic scaling in our algorithms, although we would need only to add a module to periodically recalculate D_x at the conclusion of an iteration of Algorithm D6.1.1 or D6.1.3.

An example illustrating the importance of considering the scale of the independent variables is given below.

EXAMPLE 7.1.1 A common test problem for minimization algorithms is the Rosenbrock banana function

$$f(x) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2, \tag{7.1.2}$$

which has its minimum at $x_* = (1, 1)^T$. Two typical starting points are $x_0 = (-1.2, 1)^T$ and $x_0 = (6.39, -0.221)^T$. This problem is well scaled, but if $\alpha \neq 1$, then the scale can be made worse by substituting $\alpha\hat{x}_1$ for x_1, and $\hat{x}_2/\alpha$ for x_2 in (7.1.2), giving

$$\hat{f}(\hat{x}) = f\left(\alpha\hat{x}_1, \frac{\hat{x}_2}{\alpha}\right) = 100\left((\alpha\hat{x}_1)^2 - \frac{\hat{x}_2}{\alpha}\right)^2 - (1 - \alpha\hat{x}_1)^2$$

$$\hat{x}_* = \left(\frac{1}{\alpha}, \alpha\right)^T.$$

This corresponds to the transformation

$$\hat{x} = \begin{bmatrix} \frac{1}{\alpha} & 0 \\ 0 & \alpha \end{bmatrix} x.$$

If we run the minimization algorithms found in the appendix on $\hat{f}(\hat{x})$, starting from $\hat{x}_0 = (-1.2/\alpha, \alpha)^T$ and $\hat{x}_0 = (6.39/\alpha, \alpha(-0.221))^T$, use exact derivatives, the "hook" globalizing step, and the default tolerances, and neglect the scale by setting typx_1 = typx_2 = 1, then the number of iterations required for convergence with various values of α are as follows (the asterisk indicates failure to converge after 150 iterations):

α	Iterations from $x_0 = (-1.2/\alpha, \alpha)^T$	Iterations from $x_0 = (6.39/\alpha, \alpha(-0.221))^T$
0.01	150 + *	150 + *
0.1	94	47
1	24	29
10	52	48
100	150 + *	150 + *

However, if we set typx_1 = $1/\alpha$, typx_2 = α, then the output of the program is exactly the same as for $\alpha = 1$ in all cases, except that the x values are multiplied by

$$\begin{bmatrix} \frac{1}{\alpha} & 0 \\ 0 & \alpha \end{bmatrix}.$$

It is also necessary to consider the scale of the dependent variables. In minimization problems, the scale of the objective function f really only matters in the stopping conditions, discussed in Section 7.2. In all other calculations, such as the test $f(x_+) < f(x) + 10^{-4}\, \nabla f(x)^T(x_+ - x)$, a change in the units of f is of no consequence.

On the other hand, in solving systems of nonlinear equations, differing sizes among the component functions f_i can cause the same types of problems as differing sizes among the independent variables. Once again, the Newton step is independent of this scaling (see Exercise 4). However, the globalizing strategy for nonlinear equations requires a decrease in $\|F\|_2$, and it is clear that if the units of two component functions of $F(x)$ are widely different, then the smaller component function will be virtually ignored.

For this reason, our algorithms also use a positive diagonal scaling matrix D_F on the dependent variable $F(x)$, which works as D_x does on x. The diagonal matrix D_F is chosen so that all the components of $D_F F(x)$ will have about the same typical magnitude at points not too near the root. D_F is then used to scale F in all the modules for nonlinear equations. The affine model becomes $D_F M_c$, and the quadratic model function for the globalizing step becomes $\hat{m}_c = \frac{1}{2} \| D_F M_c \|_2^2$. All our interfaces and algorithms are implemented like this, and the user just needs to specify D_F initially. This is done by inputting values typf_i, $i = 1, \ldots, n$, giving typical magnitudes of each f_i at points not too near a root. The algorithm then sets $(D_F)_{ii} = \text{typ} f_i^{-1}$. [Actually it stores $S_F \in \mathbb{R}^n$, where $(S_F)_i = (D_F)_{ii}$.] Further instructions on choosing typf_i are given in Guideline 5 in the appendix.

7.2 STOPPING CRITERIA

In this section we discuss how to terminate our algorithms. The stopping criteria are the same common-sense conditions discussed in Section 2.5 for one-dimensional problems: "Have we solved the problem?" "Have we ground to a halt?" or "Have we run out of money, time, or patience?" The factors that need consideration are how to implement these tests in finite-precision arithmetic, and how to pay proper attention to the scales of the dependent and independent variables.

We first discuss stopping criteria for unconstrained minimization. The most important test is "Have we solved the problem?" In infinite precision, a necessary condition for x to be the exact minimizer of f is $\nabla f(x) = 0$, but in an iterative and finite-precision algorithm, we will need to modify this condition to $\nabla f(x) \cong 0$. Although $\nabla f(x) = 0$ can also occur at a maximum or saddle point, our globalizing strategy and our strategy of perturbing the model Hessian to be positive definite make convergence virtually impossible to maxima and saddle points. In our context, therefore, $\nabla f(x) = 0$ is considered a necessary and sufficient condition for x to be a local minimizer of f.

To test whether $\nabla f \cong 0$, a test such as

$$\| \nabla f(x_+) \| \le \varepsilon \tag{7.2.1}$$

is inadequate, because it is strongly dependent on the scaling of both f and x. For example, if $\varepsilon = 10^{-3}$ and f is always in $[10^{-7}, 10^{-5}]$, then it is likely that any value of x will satisfy (7.2.1); conversely if $f \in [10^5, 10^7]$, (7.2.1) may be overly stringent. Also, if x is inconsistently scaled—for example, $x_1 \in [10^6, 10^7]$ and $x_2 \in [10^{-1}, 1]$—then (7.2.1) is likely to treat the variables unequally. A common remedy is to use

$$| \nabla f(x_+)^T \, \nabla^2 f(x_+)^{-1} \, \nabla f(x_+) | \le \varepsilon. \tag{7.2.2}$$

Inequality (7.2.2) is invariant under any linear transformation of the independent variables and thus is independent of the scaling of x. However, it is still dependent on the scaling of f. A more direct modification of (7.2.1) is to define the relative gradient of f at x by

$$\text{relgrad}\,(x)_i = \frac{\text{relative rate of change in } f}{\text{relative rate of change in } x_i} = \lim_{\delta \to 0} \frac{\dfrac{f(x+\delta e_i) - f(x)}{f(x)}}{\dfrac{\delta}{x_i}} = \frac{\nabla f(x)_i\, x_i}{f(x)} \tag{7.2.3}$$

and test

$$\|\,\text{relgrad}\,(x_+)\|_\infty \le \text{gradtol}. \tag{7.2.4}$$

Test (7.2.4) is independent of any change in the units of f or x. It has the drawback that the idea of relative change in x_i or f breaks down if x_i or $f(x)$ happen to be near zero. This problem is easily fixed by replacing x_i and f in (7.2.3) by max $\{|x_i|, \text{typ}x_i\}$ and max $\{|f(x)|, \text{typ}f\}$, respectively, where typ$f$ is the user's estimate of a typical magnitude of f. The resulting test,

$$\max_{1 \le i \le n} \left| \frac{\nabla f(x)_i \max\{|(x_+)_i|, \text{typ}x_i\}}{\max\{|f(x_+)|, \text{typ}f\}} \right| \le \text{gradtol}, \tag{7.2.5}$$

is the one used in our algorithms.

It should be mentioned that the problem of measuring relative change when the argument z is near zero is commonly addressed by substituting $(|z| + 1)$ or max $\{|z|, 1\}$ for z. It is apparent from the above discussion that both these substitutions make the implicit assumption that z has scale around 1. They may also work satisfactorily if $|z|$ is much larger than 1, but they will be unsatisfactory if $|z|$ is always much smaller than 1. Therefore, if a value of typ$z$ is available, the substitution max $\{|z|, \text{typ}z\}$ is preferable.

The other stopping tests for minimization are simpler to explain. The test for whether the algorithm has ground to a halt, either because it has stalled or converged, is

$$\|\,\text{relative change in successive values of } x\,\|_\infty \le \text{steptol}. \tag{7.2.6}$$

Following the above discussion, we measure the relative change in x_i by

$$\text{rel}x_i = \frac{|(x_+)_i - (x_c)_i|}{\max\{|(x_+)_i|, \text{typ}x_i\}}. \tag{7.2.7}$$

Selection of steptol is discussed in Guideline 2; basically, if p significant digits of x_* are desired, steptol should be set to 10^{-p}.

As in most iterative procedures, we quantify available time, money, and patience by imposing an iteration limit. In real applications this limit is often governed by the cost of each iteration, which can be high if function evaluation is expensive. During debugging, it is a good idea to use a low iteration limit so that an erroneous program won't run too long. In a minimization algorithm one should also test for divergence of the iterates x_k, which can occur if f is unbounded below, or asymptotically approaches a finite lower bound from above. To test for divergence, we ask the user to supply a maximum step length, and if five consecutive steps are this long, the algorithm is terminated. (See Guideline 2.)

The stopping criteria for systems of nonlinear equations are similar. We first test whether x_+ approximately solves the problem—that is, whether $F(x_+) \cong 0$. The test $\| F(x_+) \| \le \varepsilon$ is again inappropriate, owing to problems with scaling, but since $(D_F)_{ii} = 1/\text{typ}f_i$ has been selected so that $(D_F)_{ii} F_i$ should have magnitude about 1 at points not near the root, the test

$$\| D_F F \|_\infty \le \text{fntol}$$

should be appropriate. Suggestions for fntol are given in Guideline 5; values around 10^{-5} are typical.

Next one tests whether the algorithm has converged or stalled at x_+, using the test (7.2.6–7.2.7). The tests for iteration limit and divergence are also the same as for minimization, though it is less likely for an algorithm for solving $F(x) = 0$ to diverge.

Finally, it is possible for our nonlinear equations algorithm to become stuck by finding a local minimum of the associated minimization function $f = \frac{1}{2}\| D_F F \|_2^2$ at which $F \ne 0$ (see Figure 6.5.1). Although convergence test (7.2.6–7.2.7) will stop the algorithm in this case, we prefer to test for it explicitly by checking whether the gradient of f at x_+ is nearly zero, using a relative measure of the gradient analogous to (7.2.5). If the algorithm has reached a local minimum of $\| D_F F \|_2^2$ at which $F \ne 0$, all that can be done is to restart the algorithm in a different place.

Algorithms A7.2.1 and A7.2.3 in the appendix contain the stopping criteria for unconstrained minimization and nonlinear equations, respectively. Algorithms A7.2.2 and A7.2.4 are used before the initial iteration to test whether the starting point x_0 is already a minimizer or a root, respectively. Guidelines 2 and 5 contain advice for selecting all the user-supplied parameters. In our software that implements these algorithms [Schnabel, Weiss, and Koontz (1982)], default values are available for all the stopping and scaling tolerances.

7.3 TESTING

Once a computer program for nonlinear equations or minimization has been written, it will presumably be tested to see whether it works correctly and how it compares with other software that solves the same problem. It is important to discuss two aspects of this testing process: (1) how should the software be tested and (2) what criteria should be used to evaluate its performance? It is perhaps surprising that there is no consensus on either of these important questions. In this section we indicate briefly some of the leading ideas.

The first job in testing is to see that the code is working correctly. By "correctly" we currently mean a general idea that the program is doing what it should, as opposed to the computer scientist's much more stringent definition of "correctness." This is certainly a nontrivial task for any program the size of those in this book. We strongly recommend a modular testing procedure, testing first each module as it is written, then the pieces the modules form, and finally the entire program. Taking the approach of testing the entire program at once can make finding errors extremely difficult. The difficulty with modular testing is that it may not be obvious how to construct input data to test some modules, such as the module for updating the trust region. Our advice is to start with data from the simplest problems, perhaps one or two dimensions with identity or diagonal Jacobians or Hessians, since it should be possible to hand-check the calculations. Then it is advisable to check the module on more complex problems. An advantage of this modular testing is that it usually adds to our understanding of the algorithms.

Once all the components are working correctly, one should test the program on a variety of nonlinear problems. This serves two purposes: to check that the entire program is working correctly, and then to observe its performance on some standard problems. The first problems to try are the simplest ones: linear systems in two or three dimensions for a program to solve systems of nonlinear equations, positive definite quadratics in two or three variables for minimization routines. Then one might try polynomials or systems of equations of slightly higher degree and small (two to five) dimension. When the program is working correctly on these, it is time to run it on some standard problems accepted in this field as providing good tests of software for nonlinear equations or minimization. Many of them are quite difficult. It is often useful to start these test problems from 10 or 100 times further out on the ray from the solution x_* to the standard starting point x_0, as well as from x_0; Moré, Garbow, and Hillstrom (1981) report that this often brings out in programs important differences not indicated from the standard starting points.

Although the literature on test problems is still developing, we provide some currently accepted problems in Appendix B. We give a nucleus of standard problems for nonlinear equations or minimization sufficient for class projects or preliminary research results and provide references to additional problems that would be used in a thorough research study. It should be noted that most of these problems are well scaled; this is indicative of the lack of attention that has been given to the scaling problem. The dimensions of the test problems in Appendix B are a reflection of the problems currently being solved. The supply of medium (10 to 100) dimensional problems is still inadequate, and the cost of testing on such problems is a significant factor.

The difficult question of how to evaluate and compare software for minimization or nonlinear equations is a side issue in this book. It is complicated by whether one is primarily interested in measuring the efficiency and reliability of the program in solving problems, or its overall quality as a piece of software. In the latter case, one is also interested in the interface between the software and its users (documentation, ease of use, response to erroneous input, robustness, quality of output), and between the software and the computing environment (portability). We will comment only on the first set of issues; for a discussion of all these issues, see, e.g., Fosdick (1979).

By reliability, we mean the ability of the program to solve successfully the problems it is intended for. This is determined first by its results on test problems, and ultimately by whether it solves the problems of the user community. For the user, efficiency refers to the computing bill incurred running the program on his or her problems. For minimization or nonlinear equations problems, this is sometimes measured by the running times of the program on test problems. Accurate timing data is difficult to obtain on shared computing systems, but a more obvious objection is the inherent assumption that the test problems are like those of the user. Another common measure of efficiency is the number of function and derivative evaluations the program requires to solve test problems. The justification for this measure is that it indicates the cost on those problems that are inherently expensive, namely those for which function and derivative evaluation is expensive. This measure is especially appropriate for evaluating secant methods (see Chapters 8 and 9), since they are used often on such problems. In minimization testing, the number of function and gradient evaluations used sometimes are combined into one statistic,

$$\begin{aligned}&\text{number of equivalent function evaluations}\\&\qquad = \text{number of } f\text{-evaluations} + n\,(\text{number of } \nabla f\text{-evaluations}).\end{aligned}$$

This statistic indicates the number of function evaluations that would be used if the gradients were evaluated by finite differences. Since this is not always the case, it is preferable to report the function and gradient totals separately.

Some other possible measures of efficiency are number of iterations required, computational cost per iteration, and computer storage required. The number of iterations required is a simple measure, but is useful only if it is correlated to the running time of the problem, or the function and derivative evaluations required. The computational cost of an iteration, excluding function and derivative evaluations, is invariably determined by the linear algebra and is usually proportional to n^3, or n^2 for secant methods. When multiplied by the number of iterations required, it gives an indication of the running time for a problem where function and derivative evaluation is very inexpensive. Computer storage is usually not an issue for problems of the size discussed in this book; however, storage and computational cost per iteration become crucially important for large problems.

Using the above measures, one can compare two entirely different programs for minimization or nonlinear equations, but often one is interested only in comparing two or more versions of a particular segment of the algorithm—for example, the line search. In this case it may be desirable to test the alternative segments by substituting them into a modular program such as ours, so that the remainder of the program is identical throughout the tests. Such controlled testing reduces the reliance of the results on other aspects of the programs, but it is possible for the comparison to be prejudiced if the remainder of the program is more favorable to one alternative.

Finally, the reader should realize that we have discussed the evaluation of computer programs, not algorithms, in this section. The distinction is that a computer program may include many details that are crucial to its performance but are not part of the "basic algorithm." Examples are stopping criteria, linear algebra routines, and tolerances in line-search or trust region algorithms. The basic algorithm may be evaluated using measures we have already discussed: rate of local convergence, global convergence properties, performance on special classes of functions. When one tests a computer program, however, as discussed above, one must realize that a particular software implementation of a basic algorithm is being tested, and that two implementations of the same basic algorithm may perform quite differently.

7.4 EXERCISES

1. Consider the problem

$$\underset{x \in \mathbb{R}^2}{\text{minimize}} \quad (x_1 - 10^6)^4 + (x_1 - 10^6)^2 + (x_2 - 10^{-6})^4 + (x_2 - 10^{-6})^2.$$

What problems might you encounter in applying an optimization algorithm without scaling to this problem? (Consider steepest-descent directions, trust regions, stopping criteria.) What value would you give to typx_1, typx_2 in our algorithms in order to alleviate these problems? What change might be even more helpful?

2. Let $f\colon \mathbb{R}^n \longrightarrow \mathbb{R}$, $T \in \mathbb{R}^{n \times n}$ nonsingular. For any $x \in \mathbb{R}^n$, define $\hat{x} = Tx$, $\hat{f}(\hat{x}) = f(T^{-1}\hat{x}) = f(x)$. Using the chain rule for multivariable calculus, show that

$$\nabla \hat{f}(\hat{x}) = T^{-T}\, \nabla f(x),$$

$$\nabla^2 \hat{f}(\hat{x}) = T^{-T}\, \nabla^2 f(x) T^{-1}.$$

3. Let $f \in R$, $g \in \mathbb{R}^n$, $H \in \mathbb{R}^{n \times n}$, H symmetric and positive definite, $D \in \mathbb{R}^{n \times n}$, D a positive diagonal matrix. Using Lemma 6.4.1, show that the solution to

$$\underset{s \in \mathbb{R}^n}{\text{minimize}} \quad f + g^T s + \tfrac{1}{2} s^T H s$$

$$\text{subject to} \quad \|Ds\|_2 \le \delta$$

is given by

$$s(\mu) = -(H + \mu D^2)^{-1} g$$

for some $\mu \geq 0$. [*Hint*: Make the transformation $\hat{s} = Ds$, use Lemma 6.4.1, and transform back.]

4. Let $F : \mathbb{R}^n \longrightarrow \mathbb{R}^n$, $T_1, T_2 \in \mathbb{R}^{n \times n}$ nonsingular. For any $x \in \mathbb{R}^n$ define $\hat{x} = T_1 x$, $\hat{F}(\hat{x}) = T_2 F(T_1^{-1}\hat{x}) = T_2 F(x)$. Show that the Jacobian matrix of $\hat{F}$ with respect to $\hat{x}$ is given by

$$\hat{J}(\hat{x}) = T_2 J(x) T_1^{-1}.$$

What is the Newton step in the $\hat{x}$ variable space? If this step is transformed back to the original variable space, how does it compare to the normal Newton step in the original variable space?

5. What are some situations in which the scaling strategy of Section 7.1 would be unsatisfactory? Suggest a dynamic scaling strategy that would be successful in these situations. Now give a situation in which your dynamic strategy would be unsuccessful.

6. Suppose our stopping test for minimization finds that $\nabla f(x_k) \approx 0$. How could you test whether x_k is a saddle point (or maximizer)? If x_k is a saddle point, how could you proceed in the minimization algorithm?

7. Write a program for unconstrained minimization or solving systems of nonlinear equations using the algorithms in Appendix A (and using exact derivatives). Choose one of globalizing strategies of Sections 6.3 and 6.4 to implement in your program. Debug and test your program as discussed in Section 7.3.