This equation, if used with i ranging over the roots already polished, will prevent a tentative root from spuriously hopping to another one's true root. It is an example of so-called *zero suppression* as an alternative to true deflation.

Muller's method, which was described above, can also be useful at the polishing stage.

CITED REFERENCES AND FURTHER READING:

Acton, F.S. 1970, *Numerical Methods That Work*; 1990, corrected edition (Washington: Mathematical Association of America), Chapter 7. [1]

Peters G., and Wilkinson, J.H. 1971, *Journal of the Institute of Mathematics and its Applications*, vol. 8, pp. 16–35. [2]

IMSL Math/Library Users Manual (IMSL Inc., 2500 CityWest Boulevard, Houston TX 77042). [3]

Ralston, A., and Rabinowitz, P. 1978, *A First Course in Numerical Analysis*, 2nd ed. (New York: McGraw-Hill), §8.9–8.13. [4]

Adams, D.A. 1967, *Communications of the ACM*, vol. 10, pp. 655–658. [5]

Johnson, L.W., and Riess, R.D. 1982, *Numerical Analysis*, 2nd ed. (Reading, MA: Addison-Wesley), §4.4.3. [6]

Henrici, P. 1974, *Applied and Computational Complex Analysis*, vol. 1 (New York: Wiley).

Stoer, J., and Bulirsch, R. 1980, *Introduction to Numerical Analysis* (New York: Springer-Verlag), §§5.5–5.9.

9.6 Newton-Raphson Method for Nonlinear Systems of Equations

We make an extreme, but wholly defensible, statement: There are *no* good, general methods for solving systems of more than one nonlinear equation. Furthermore, it is not hard to see why (very likely) there *never will be* any good, general methods: Consider the case of two dimensions, where we want to solve simultaneously

$$\begin{aligned} f(x,y) &= 0 \\ g(x,y) &= 0 \end{aligned} \tag{9.6.1}$$

The functions f and g are two arbitrary functions, each of which has zero contour lines that divide the (x,y) plane into regions where their respective function is positive or negative. These zero contour boundaries are of interest to us. The solutions that we seek are those points (if any) that are common to the zero contours of f and g (see Figure 9.6.1). Unfortunately, the functions f and g have, in general, no relation to each other at all! There is nothing special about a common point from either f's point of view, or from g's. In order to find all common points, which are the solutions of our nonlinear equations, we will (in general) have to do neither more nor less than map out the full zero contours of both functions. Note further that the zero contours will (in general) consist of an unknown number of disjoint closed curves. How can we ever hope to know when we have found all such disjoint pieces?

For problems in more than two dimensions, we need to find points mutually common to N unrelated zero-contour hypersurfaces, each of dimension $N-1$. You

Sample page from NUMERICAL RECIPES IN FORTRAN 77: THE ART OF SCIENTIFIC COMPUTING (ISBN 0-521-43064-X)
Copyright (C) 1986-1992 by Cambridge University Press. Programs Copyright (C) 1986-1992 by Numerical Recipes Software.
Permission is granted for internet users to make one paper copy for their own personal use. Further reproduction, or any copying of machine-readable files (including this one) to any server computer, is strictly prohibited. To order Numerical Recipes books or CDROMs, visit website
http://www.nr.com or call 1-800-872-7423 (North America only), or send email to directcustserv@cambridge.org (outside North America).

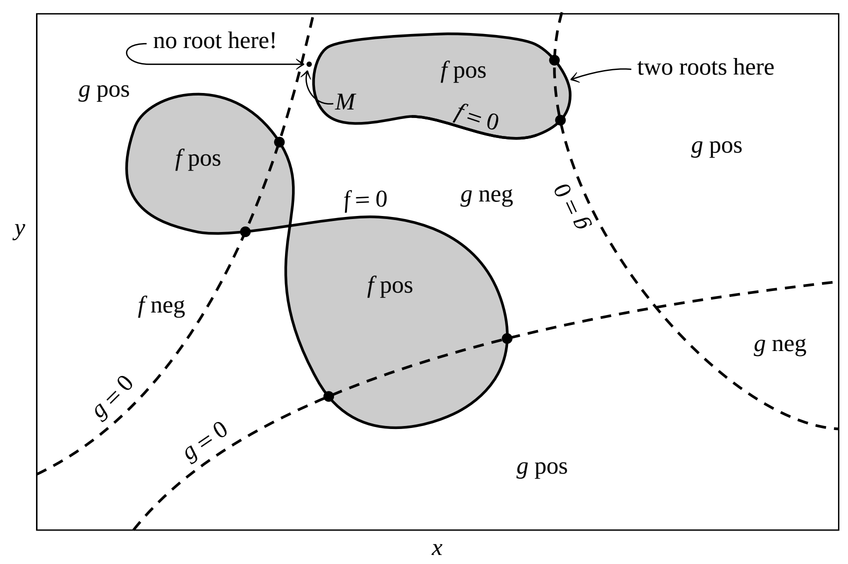

Figure 9.6.1. Solution of two nonlinear equations in two unknowns. Solid curves refer to $f(x, y)$, dashed curves to $g(x, y)$. Each equation divides the (x, y) plane into positive and negative regions, bounded by zero curves. The desired solutions are the intersections of these unrelated zero curves. The number of solutions is *a priori* unknown.

see that root finding becomes virtually impossible without insight! You will almost always have to use additional information, specific to your particular problem, to answer such basic questions as, "Do I expect a unique solution?" and "Approximately where?" Acton [1] has a good discussion of some of the particular strategies that can be tried.

In this section we will discuss the simplest multidimensional root finding method, Newton-Raphson. This method gives you a very efficient means of converging to a root, if you have a sufficiently good initial guess. It can also spectacularly fail to converge, indicating (though not proving) that your putative root does not exist nearby. In §9.7 we discuss more sophisticated implementations of the Newton-Raphson method, which try to improve on Newton-Raphson's poor global convergence. A multidimensional generalization of the secant method, called Broyden's method, is also discussed in §9.7.

A typical problem gives N functional relations to be zeroed, involving variables $x_i, i = 1, 2, \ldots, N$:

$$F_i(x_1, x_2, \ldots, x_N) = 0 \qquad i = 1, 2, \ldots, N. \tag{9.6.2}$$

We let $\mathbf{x}$ denote the entire vector of values x_i and $\mathbf{F}$ denote the entire vector of functions F_i. In the neighborhood of $\mathbf{x}$, each of the functions F_i can be expanded in Taylor series

$$F_i(\mathbf{x} + \delta\mathbf{x}) = F_i(\mathbf{x}) + \sum_{j=1}^{N} \frac{\partial F_i}{\partial x_j} \delta x_j + O(\delta\mathbf{x}^2). \tag{9.6.3}$$

Sample page from NUMERICAL RECIPES IN FORTRAN 77: THE ART OF SCIENTIFIC COMPUTING (ISBN 0-521-43064-X)
Copyright (C) 1986-1992 by Cambridge University Press. Programs Copyright (C) 1986-1992 by Numerical Recipes Software.
Permission is granted for internet users to make one paper copy for their own personal use. Further reproduction, or any copying of machine-readable files (including this one) to any server computer, is strictly prohibited. To order Numerical Recipes books or CDROMs, visit website http://www.nr.com or call 1-800-872-7423 (North America only), or send email to directcustserv@cambridge.org (outside North America).

The matrix of partial derivatives appearing in equation (9.6.3) is the *Jacobian* matrix **J**:

$$J_{ij} \equiv \frac{\partial F_i}{\partial x_j}. \tag{9.6.4}$$

In matrix notation equation (9.6.3) is

$$\mathbf{F}(\mathbf{x} + \delta\mathbf{x}) = \mathbf{F}(\mathbf{x}) + \mathbf{J} \cdot \delta\mathbf{x} + O(\delta\mathbf{x}^2). \tag{9.6.5}$$

By neglecting terms of order $\delta\mathbf{x}^2$ and higher and by setting $\mathbf{F}(\mathbf{x} + \delta\mathbf{x}) = 0$, we obtain a set of linear equations for the corrections $\delta\mathbf{x}$ that move each function closer to zero simultaneously, namely

$$\mathbf{J} \cdot \delta\mathbf{x} = -\mathbf{F}. \tag{9.6.6}$$

Matrix equation (9.6.6) can be solved by LU decomposition as described in §2.3. The corrections are then added to the solution vector,

$$\mathbf{x}_{\text{new}} = \mathbf{x}_{\text{old}} + \delta\mathbf{x} \tag{9.6.7}$$

and the process is iterated to convergence. In general it is a good idea to check the degree to which both functions and variables have converged. Once either reaches machine accuracy, the other won't change.

The following routine `mnewt` performs `ntrial` iterations starting from an initial guess at the solution vector `x` of length `n` variables. Iteration stops if either the sum of the magnitudes of the functions F_i is less than some tolerance `tolf`, or the sum of the absolute values of the corrections to δx_i is less than some tolerance `tolx`. `mnewt` calls a user supplied subroutine `usrfun` which must return the function values **F** and the Jacobian matrix **J**. If **J** is difficult to compute analytically, you can try having `usrfun` call the routine `fdjac` of §9.7 to compute the partial derivatives by finite differences. You should not make `ntrial` too big; rather inspect to see what is happening before continuing for some further iterations.

```
      SUBROUTINE mnewt(ntrial,x,n,tolx,tolf)
      INTEGER n,ntrial,NP
      REAL tolf,tolx,x(n)
      PARAMETER (NP=15)                  Up to NP variables.
C     USES lubksb,ludcmp,usrfun
         Given an initial guess x for a root in n dimensions, take ntrial Newton-Raphson steps to
         improve the root. Stop if the root converges in either summed absolute variable increments
         tolx or summed absolute function values tolf.
      INTEGER i,k,indx(NP)
      REAL d,errf,errx,fjac(NP,NP),fvec(NP),p(NP)
      do 14 k=1,ntrial
        call usrfun(x,n,NP,fvec,fjac)    User subroutine supplies function values at x in fvec
        errf=0.                              and Jacobian matrix in fjac.
        do 11 i=1,n                      Check function convergence.
          errf=errf+abs(fvec(i))
        enddo 11
        if(errf.le.tolf)return
        do 12 i=1,n                      Right-hand side of linear equations.
          p(i)=-fvec(i)
        enddo 12
        call ludcmp(fjac,n,NP,indx,d)    Solve linear equations using LU decomposition.
        call lubksb(fjac,n,NP,indx,p)
```

Sample page from NUMERICAL RECIPES IN FORTRAN 77: THE ART OF SCIENTIFIC COMPUTING (ISBN 0-521-43064-X)
Copyright (C) 1986-1992 by Cambridge University Press. Programs Copyright (C) 1986-1992 by Numerical Recipes Software.
Permission is granted for internet users to make one paper copy for their own personal use. Further reproduction, or any copying of machine-readable files (including this one) to any server computer, is strictly prohibited. To order Numerical Recipes books or CDROMs, visit website
http://www.nr.com or call 1-800-872-7423 (North America only), or send email to directcustserv@cambridge.org (outside North America).

```
        errx=0.                         Check root convergence.
        do 13 i=1,n                     Update solution.
            errx=errx+abs(p(i))
            x(i)=x(i)+p(i)
        enddo 13
        if(errx.le.tolx)return
    enddo 14
    return
    END
```

Newton's Method versus Minimization

In the next chapter, we will find that there *are* efficient general techniques for finding a minimum of a function of many variables. Why is that task (relatively) easy, while multidimensional root finding is often quite hard? Isn't minimization equivalent to finding a zero of an N-dimensional gradient vector, not so different from zeroing an N-dimensional function? No! The components of a gradient vector are not independent, arbitrary functions. Rather, they obey so-called integrability conditions that are highly restrictive. Put crudely, you can always find a minimum by sliding downhill on a single surface. The test of "downhillness" is thus one-dimensional. There is no analogous conceptual procedure for finding a multidimensional root, where "downhill" must mean simultaneously downhill in N separate function spaces, thus allowing a multitude of trade-offs, as to how much progress in one dimension is worth compared with progress in another.

It might occur to you to carry out multidimensional root finding by collapsing all these dimensions into one: Add up the sums of squares of the individual functions F_i to get a master function F which (i) is positive definite, and (ii) has a global minimum of zero exactly at all solutions of the original set of nonlinear equations. Unfortunately, as you will see in the next chapter, the efficient algorithms for finding minima come to rest on global and local minima indiscriminately. You will often find, to your great dissatisfaction, that your function F has a great number of local minima. In Figure 9.6.1, for example, there is likely to be a local minimum wherever the zero contours of f and g make a close approach to each other. The point labeled M is such a point, and one sees that there are no nearby roots.

However, we will now see that sophisticated strategies for multidimensional root finding can in fact make use of the idea of minimizing a master function F, by *combining* it with Newton's method applied to the full set of functions F_i. While such methods can still occasionally fail by coming to rest on a local minimum of F, they often succeed where a direct attack via Newton's method alone fails. The next section deals with these methods.

CITED REFERENCES AND FURTHER READING:

Acton, F.S. 1970, *Numerical Methods That Work*; 1990, corrected edition (Washington: Mathematical Association of America), Chapter 14. [1]

Ostrowski, A.M. 1966, *Solutions of Equations and Systems of Equations*, 2nd ed. (New York: Academic Press).

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

Sample page from NUMERICAL RECIPES IN FORTRAN 77: THE ART OF SCIENTIFIC COMPUTING (ISBN 0-521-43064-X)
Copyright (C) 1986-1992 by Cambridge University Press. Programs Copyright (C) 1986-1992 by Numerical Recipes Software.
Permission is granted for internet users to make one paper copy for their own personal use. Further reproduction, or any copying of machine-readable files (including this one) to any server computer, is strictly prohibited. To order Numerical Recipes books or CDROMs, visit website http://www.nr.com or call 1-800-872-7423 (North America only), or send email to directcustserv@cambridge.org (outside North America).

9.7 Globally Convergent Methods for Nonlinear Systems of Equations

We have seen that Newton's method for solving nonlinear equations has an unfortunate tendency to wander off into the wild blue yonder if the initial guess is not sufficiently close to the root. A *global* method is one that converges to a solution from almost any starting point. In this section we will develop an algorithm that combines the rapid local convergence of Newton's method with a globally convergent strategy that will guarantee some progress towards the solution at each iteration. The algorithm is closely related to the quasi-Newton method of minimization which we will describe in §10.7.

Recall our discussion of §9.6: the Newton step for the set of equations

$$\mathbf{F}(\mathbf{x}) = 0 \tag{9.7.1}$$

is

$$\mathbf{x}_{\text{new}} = \mathbf{x}_{\text{old}} + \delta\mathbf{x} \tag{9.7.2}$$

where

$$\delta\mathbf{x} = -\mathbf{J}^{-1} \cdot \mathbf{F} \tag{9.7.3}$$

Here **J** is the Jacobian matrix. How do we decide whether to accept the Newton step $\delta\mathbf{x}$? A reasonable strategy is to require that the step decrease $|\mathbf{F}|^2 = \mathbf{F} \cdot \mathbf{F}$. This is the same requirement we would impose if we were trying to minimize

$$f = \frac{1}{2}\mathbf{F} \cdot \mathbf{F} \tag{9.7.4}$$

(The $\frac{1}{2}$ is for later convenience.) Every solution to (9.7.1) minimizes (9.7.4), but there may be local minima of (9.7.4) that are not solutions to (9.7.1). Thus, as already mentioned, simply applying one of our minimum finding algorithms from Chapter 10 to (9.7.4) is *not* a good idea.

To develop a better strategy, note that the Newton step (9.7.3) is a *descent direction* for f:

$$\nabla f \cdot \delta\mathbf{x} = (\mathbf{F} \cdot \mathbf{J}) \cdot (-\mathbf{J}^{-1} \cdot \mathbf{F}) = -\mathbf{F} \cdot \mathbf{F} < 0 \tag{9.7.5}$$

Thus our strategy is quite simple: We always first try the full Newton step, because once we are close enough to the solution we will get quadratic convergence. However, we check at each iteration that the proposed step reduces f. If not, we *backtrack* along the Newton direction until we have an acceptable step. Because the Newton step is a descent direction for f, we are guaranteed to find an acceptable step by backtracking. We will discuss the backtracking algorithm in more detail below.

Note that this method essentially minimizes f by taking Newton steps designed to bring **F** to zero. This is *not* equivalent to minimizing f directly by taking Newton steps designed to bring ∇f to zero. While the method can still occasionally fail by landing on a local minimum of f, this is quite rare in practice. The routine `newt` below will warn you if this happens. The remedy is to try a new starting point.

Sample page from NUMERICAL RECIPES IN FORTRAN 77: THE ART OF SCIENTIFIC COMPUTING (ISBN 0-521-43064-X)
Copyright (C) 1986-1992 by Cambridge University Press. Programs Copyright (C) 1986-1992 by Numerical Recipes Software.
Permission is granted for internet users to make one paper copy for their own personal use. Further reproduction, or any copying of machine-readable files (including this one) to any server computer, is strictly prohibited. To order Numerical Recipes books or CDROMs, visit website http://www.nr.com or call 1-800-872-7423 (North America only), or send email to directcustserv@cambridge.org (outside North America).