

# Using Vector Divisions in Solving Nonlinear Systems of Equations

Yixun Shi<sup>1</sup>

Department of Mathematics, Computer Science and Statistics  
Bloomsburg University of Pennsylvania  
Bloomsburg, PA 17815, USA  
yshi@bloomu.edu

## Abstract

With the motivation of avoiding the computation of the inverse of Jacobian matrix, which is involved in Newton's method, we consider the use of vector divisions in solving a nonlinear system of equations  $F(x) = 0$ . Vector divisions are applied to form the secant method formulas. Based on that, a globally convergent hybrid algorithm for solving the nonlinear system of equations is proposed in this paper.

**Mathematics Subject Classification:** 65H10

**Keywords:** nonlinear system of equations, vector division, global convergence, Newton's method, secant method, step length

## 1 Introduction

Consider a nonlinear system of equations

$$F(x) = 0 \tag{1}$$

where  $F : R^n \rightarrow R^n$ ,  $F = (f_1, f_2, \dots, f_n)^T$  is well defined and has continuous partial derivatives on an open set of  $R^n$ . We use  $J(x)$  to denote the Jacobian matrix of  $F$ , i.e.

$$J(x) \equiv F'(x) \equiv \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(x) & \cdots & \frac{\partial f_1}{\partial x_n}(x) \\ \cdots & \cdots & \cdots \\ \frac{\partial f_n}{\partial x_1}(x) & \cdots & \frac{\partial f_n}{\partial x_n}(x) \end{bmatrix}. \tag{2}$$

---

<sup>1</sup>This work is supported by the 2007–2008 PASHEE FPDC grant.

The classical Newton's method for solving the problem (1) follows the formula

$$x_{k+1} = x_k - J(x_k)^{-1}F(x_k) \quad (3)$$

It is well known that Newton's method has superlinear local convergence. Although the global convergence is not guaranteed, in many cases the convergence region is much larger than predicted by local convergence theorems, and the method converges even if the initial point is not very close to the true solution of the system.

However, Newton's method requires the computation of  $J(x)^{-1}$  in each iteration, and therefore is not beneficial in terms of computational cost.

Many variations of Newton's method, such as quasi-Newton methods, have been explored with the motivation of avoiding the calculation of  $J(x)^{-1}$ . In this paper, we consider the use of vector divisions with the secant method in place of the Newton's method. Based on that, we propose a globally convergent hybrid algorithm for solving the nonlinear system (1).

## 2 Secant Formulas Using Vector Division

In case  $n = 1$ , the secant formula is

$$x_{k+1} = x_k - \frac{F(x_k)}{F[x_k, x_{k-1}]} \quad (4)$$

with

$$F[x_k, x_{k-1}] = \frac{F(x_k) - F(x_{k-1})}{x_k - x_{k-1}}.$$

When  $n > 1$ ,  $x_k$ ,  $x_{k-1}$ ,  $F(x_k)$  and  $F(x_{k-1})$  are all vectors. Therefore, in order to extend the formula (4) to the case when  $n > 1$ , we need to consider the division of vectors.

One way to define the division of vectors is described in [11], based on the definition of the Samelson inverse of a vector

$$\frac{1}{a} = \frac{a}{a^T a} \quad (5)$$

for any non-zero vector  $a \in R^n$ . Definition (5) implies that

$$a^T \left( \frac{1}{a} \right) = 1 \quad (6)$$

It also implies that  $a$  and  $\frac{1}{a}$  are colinear.

At least two secant formulas may be derived based on definition (5). One formula is

$$\begin{aligned}
 x_{k+1} &= x_k - \frac{F(x_k)}{\frac{F(x_k) - F(x_{k-1})}{x_k - x_{k-1}}} \\
 &= x_k - \frac{F(x_k)}{\left(\frac{1}{x_k - x_{k-1}}\right)^T (F(x_k) - F(x_{k-1}))} \\
 &= x_k - \frac{F(x_k)}{\frac{(x_k - x_{k-1})^T (F(x_k) - F(x_{k-1}))}{(x_k - x_{k-1})^T (x_k - x_{k-1})}} \\
 &= x_k - \frac{(x_k - x_{k-1})^T (x_k - x_{k-1})}{(x_k - x_{k-1})^T (F(x_k) - F(x_{k-1}))} F(x_k) \tag{7}
 \end{aligned}$$

Another formula can be similarly derived as

$$\begin{aligned}
 x_{k+1} &= x_k - \frac{F(x_k)}{F(x_k) - F(x_{k-1})} (x_k - x_{k-1}) \\
 &= x_k - \frac{(F(x_k) - F(x_{k-1}))^T F(x_k)}{(F(x_k) - F(x_{k-1}))^T (F(x_k) - F(x_{k-1}))} (x_k - x_{k-1}) \tag{8}
 \end{aligned}$$

Note that formula (7) is of the form

$$x_{k+1} = x_k + u_k$$

with

$$u_k = -\frac{(x_k - x_{k-1})^T (x_k - x_{k-1})}{(x_k - x_{k-1})^T (F(x_k) - F(x_{k-1}))} F(x_k) \tag{9}$$

and the formula (8) can be written as

$$x_{k+1} = x_k + v_k$$

with

$$v_k = -\frac{(F(x_k) - F(x_{k-1}))^T F(x_k)}{(F(x_k) - F(x_{k-1}))^T (F(x_k) - F(x_{k-1}))} (x_k - x_{k-1}) \tag{10}$$

Combining them we may have a formula

$$x_{k+1} = x_k + s_k \tag{11}$$

with

$$s_k = \alpha_k u_k + (1 - \alpha_k) v_k \tag{12}$$

where  $\alpha_k \in R$ . Note that when  $\alpha_k = 1$  we have  $s_k = u_k$ ; and when  $\alpha_k = 0$  we have  $s_k = v_k$ .

### 3 Selection of $\alpha_k$

One strategy of selecting the coefficient  $\alpha_k$  is to choose  $\alpha_k$  to make  $s_k$  a descent direction for minimizing the square-sum

$$f(x) = (1/2)F(x)^T F(x). \quad (13)$$

If the Jacobian matrix  $J(x)$  is nonsingular, then a local minimizer of  $f$  is also a global minimizer and thus a solution of the system (1). Hence a descent direction for minimizing  $f(x)$  is appropriate for our purpose.

When  $x = x_k$ , the steepest descent direction for minimizing  $f(x)$  is

$$d_k = -\nabla f(x_k) = -J(x_k)^T F(x_k) \quad (14)$$

We aim to choose  $\alpha_k$  so that  $s_k^T d_k > 0$ . That is

$$\begin{aligned} s_k^T d_k &= (\alpha_k u_k + (1 - \alpha_k)v_k)^T d_k \\ &= \alpha_k(u_k - v_k)^T d_k + v_k^T d_k > 0 \end{aligned} \quad (15)$$

There are three possible cases here.

Case 1:  $(u_k - v_k)^T d_k \neq 0$ . In this case, any  $\alpha_k > \frac{-v_k^T d_k}{(u_k - v_k)^T d_k}$  will make  $s_k$  a descent direction for minimizing  $f(x)$ . We may choose  $\alpha_k$  such that  $\alpha_k$  maximizes the value of  $\text{Cos}[s_k, d_k] = \frac{s_k^T d_k}{\|s_k\| \|d_k\|}$ ;

Case 2:  $(u_k - v_k)^T d_k = 0$  and  $v_k^T d_k > 0$ . In this case, any value of  $\alpha_k$  will make  $s_k$  a descent direction. Indeed in this case  $u_k^T d_k = v_k^T d_k$ . Therefore we may simply choose  $\alpha_k = 1/2$  to averagely combine  $u_k$  and  $v_k$ ;

Case 3:  $(u_k - v_k)^T d_k = 0$  and  $v_k^T d_k \leq 0$ . In this case, no value of  $\alpha_k$  will make  $s_k$  a descent direction. Therefore we may use the steepest descent direction  $d_k$  in place of  $s_k$  in this iteration.

## 4 A Globally Convergent Hybrid Algorithm

To ensure the global convergence, two additional strategies can be used. One is to apply a line search along the direction  $s_k$  to determine the "step length" in each iteration. The other is to insert a steepest descent direction step as a "spacer" after every few iterations. A hybrid algorithm can then be established as the following

*Algorithm 1*

*Step 0.* Determine three parameters  $M, \rho, \sigma$  such that  $M$  is a positive integer,  $0 < \rho < 1/2$ , and  $\rho < \sigma < 1$ ;

*Step 1: (initialization)* Select two points  $x_0$  and  $x_1 \in R^n$ ;

*Step 2.* for  $k = 1, 2, \dots$  until termination, do the following:

*2.1* compute the steepest descent direction  $d_k = -J(x_k)^T F(x_k)$ ;

*2.2* if  $k$  equals a multiple of  $M$ , then insert a steepest descent direction step, that is, let  $s_k = d_k$  and go to step 2.7;

*2.3* compute  $u_k$  and  $v_k$  using (9) and (10), respectively;

*2.4* if  $(u_k - v_k)^T d_k \neq 0$ , then choose  $\alpha_k > \frac{-v_k^T d_k}{(u_k - v_k)^T d_k}$  such that  $\alpha_k$  maximizes the value of  $\text{Cos}[s_k, d_k] = \frac{s_k^T d_k}{\|s_k\| \|d_k\|}$ . Set  $s_k = \alpha_k u_k + (1 - \alpha_k)v_k$  and go to step 2.7;

*2.5* if  $(u_k - v_k)^T d_k = 0$  and  $v_k^T d_k > 0$ , then set  $s_k = 1/2(u_k + v_k)$  and go to step 2.7;

*2.6* if  $(u_k - v_k)^T d_k = 0$  and  $v_k^T d_k \leq 0$ , then set  $s_k = d_k$  and go to step 2.7;

*2.7* take a line search along the direction  $s_k$  to determine the step length  $\gamma$  such that

$$f(x_k + \gamma s_k) \leq f(x_k) - \gamma \rho d_k^T s_k \quad (16)$$

and

$$\nabla f(x_k + \gamma s_k)^T s_k \geq -\sigma d_k^T s_k; \quad (17)$$

*2.8* set  $x_{k+1} = x_k + \gamma s_k$  and go to next iteration.

The line search rule (16)—(17) is based on the work of Armijo [1], Goldstein [5]–[6], Wolfe [10], and Powell [9]. It is one of the most popular rules for

accepting a step length along a descent direction. Other line search rules may also be used. See [3], [4], [8], and so on.

Note that step 2.2 serves as a spacer with a steepest descent direction step. Thus the spacer step theorem (see, for example, [7], pp 230–231) and the line search theorem regarding the rule (16)—(17) (see [3], p29) ensure the global convergence of the above hybrid algorithm.

## References

- [1] L. Armijo, Minimization of functions having Lipschitz continuous first partial derivatives, *Pacific J. Math.* 16, 1(1966), 1—3.
- [2] Jr.J.E. Dennis and R.B. Schnabel, *Numerical methods for unconstrained optimization and nonlinear equations*, Prentice–Hall, Inc., Englewood Cliffs, New Jersey, 1983.
- [3] R. Fletcher, *Practical methods of optimization*, John Wiley & Sons, 1987.
- [4] E.P. Gill, W. Murray, A.M. Saunders and H.M. Wright, A note on a sufficient–decrease criterion for a non–derivative step–length procedure, *Math. Programming* 23(1982), 349—352.
- [5] A.A. Goldstein, On steepest descent, *SIAM J. On Control* 3(1965), 147—151.
- [6] A.A. Goldstein, *Constructive real analysis*, Harpers & Row, New York, 1967.
- [7] D.G. Luenberger, *Linear and nonlinear programming*, Addison–Wesley Publishing, 1984.
- [8] F.A. Potra and Y. Shi, An efficient line search algorithm for unconstrained optimization, *J. Optimization Theory and Application*, 85(1995), 677 — 704.
- [9] M.J.D. Powell, Some global convergence properties of a variable metric algorithm for minimization without exact line searches, *SIAM–AMS Proceedings*

Vol. IX (Eds. R.W. Cottle and C.E. Lemke). SIAM Publications, Philadelphia, 1976.

[10] P. Wolfe, Convergence conditions for ascent methods, *SIAM Review* 11(1968), 226—235.

[11] P. Wynn, Acceleration techniques for iterated vector and matrix, *Math. Comp.*, 16(1962), 301 — 322.

**Received: November 27, 2007**