Int. J. Contemp. Math. Sci., Vol. 2, 2007, no. 15, 737 - 744

Modified Quasi-Newton Methods for Solving Systems of Linear Equations

Yixun Shi

Department of Mathematics, Computer Science and Statistics
Bloomsburg University of Pennsylvania
Bloomsburg, PA 17815, USA

Abstract

Quasi-Newton methods for unconstrained optimization problems are considered for solving a system of linear equations $Ax = b$ where $A \in R^{n\times n}$, Rank$(A) = n$, $b \in R^n$, and $x \in R^n$ is the vector of unknowns. This problem can be converted into an equivalent quadratic optimization problem. Based on the observation that if $H \approx (A^TA)^{-1} = A^{-1}(A^T)^{-1}$ then $\bar{x} = HA^Tb$ can be taken as an approximate solution of the problem, we propose a modification to the Quasi-Newton method. The modified algorithm incorporates the above observation. Global convergence is ensured by adding the steepest descent direction into the combination. Numerical experiments are also reported.

Mathematics Subject Classification: 65H10

Keywords: Quasi-Newton method, steepest descent direction, conjugate gradient method

1 Introduction

In this paper, we consider the system of linear equations

$$Ax = b \tag{1}$$

where $A \in R^{n\times n}$, Rank$(A) = n$, $b \in R^n$, and $x \in R^n$ is the vector of unknowns. Since Rank$(A) = n$, the problem (1) has a unique solution in R^n.

System of linear equations arise in many areas and often the value of n can be large. Much research has been devoted to achieve their numerical solutions effectively and efficiently. See [1], [2], [3], [4], [5], [6], for example.

One of the approaches to solve the system (1) is to convert it into a quadratic optimization problem

$$\text{minimize} \quad f(x) = \frac{1}{2}(Ax-b)^T(Ax-b) \tag{2}$$

and then use iterative methods to solve this quadratic optimization problem. If we use $g(x)$ and $G(x)$ to denote the gradient $\bigtriangledown f(x)$ and the Hessian matrix (matrix of the second order partial derivatives) of $f(x)$, respectively, then we have

$$g(x) = A^T(Ax-b)$$

and

$$G(x) = A^TA.$$

Since $G(x)$ is symmetric and positive definite, the problem (2) has a unique solution x that satisfies

$$g(x) = 0$$

or equivalently

$$A^TAx = A^Tb \tag{3}$$

Since $\text{Rank}(A^T) = \text{Rank}(A) = n$, (3) is equivalent to (1). Therefore, x is the solution of (2) if and only if it is the solution of (1).

A description of Quasi-Newton method for solving general unconstrained optimization problems of the form

$$\text{minimize} \quad F(x), \quad x \in R^n \tag{4}$$

can be found, for example, in [4]. These methods are very effecive when applied to the quadratic problem (2). In general, a quasi-Newton method can be described as the following Algorithm 1. Again, we use $g(x)$ and $G(x)$ to denote the gradient $\bigtriangledown F(x)$ and the Hessian matrix of $F(x)$, respectively.

Algorithm 1

Step 1.1: (initialization) Select an initial point $x_1 \in R^n$ and a symmetric positive definite matrix $H_1 \in R^{n\times n}$.

Step 1.2: (iteration) For $k = 1, 2, \ldots$ until termination, do the following:

1.2.1 Set direction $s_k = -H_kg(x_k)$

1.2.2 Conduct a line search along the direction s_k to determine a step length α_k.

1.2.3 Set $x_{k+1} = x_k + \alpha_k s_k$

1.2.4 Update H_k *to get* H_{k+1}.

Note that if in Algorithm 1 we let $H_k = [G(x_k)]^{-1}$, then this algorithm will become exactly the Newton's method with line search. However, the main idea here is to use a symmetric positive definite matrix H_k to approximate $[G(x_k)]^{-1}$ so that the computational cost can be reduced. Different formulas used in step 1.2.4 for updating H_k yield different quasi-Newton methods, but the H_k's generated by those formulas all satisfy the following quasi-Newton condition:

$$H_{k+1}\gamma_k = \delta_k \tag{5}$$

where $\gamma_k = g(x_{k+1}) - g(x_k)$ and $\delta_k = x_{k+1} - x_k$. The initial matrix H_1 can be any symmetric positive definite matrix. Often H_1 is chosen as the $n \times n$ identity matrix I_n.

A variety of formulas for updating H_k are discussed in [4]. Among those, a one-parameter family of rank two formulas form the family of Broyden methods. The following theorem indicates that the Broyden methods are quite effective when applied to solve the quadratic optimization problem (2).

Theorem 1.1 (see [4], p64)
A Broyden method with exact line search finds the solution of problem (2) after $m \leq n$ *iterations. Here exact line search means that in step 1.2.2 the step length* α_k *needs to be the exact minimum of the function* $\bar{F}(\alpha) = F(x_k + \alpha s_k)$

The purpose of this paper is to propose a modification to the quasi-Newton method (Algorithm 1) when it is applied to solve problem (2). The modification is motivated by the observation that if $H_k \approx [G(x_k)]^{-1} = (A^T A)^{-1} = A^{-1}(A^T)^{-1}$ then $\bar{x} = H_k A^T b$ can be considered an approximate solution of (3), and therefore an approximate solution of (1) and (2). In next section, we will present a modified quasi-Newton method which incorperates this observation into the algorithm. The steepest descent direction is used to ensure the global convergence. Some numerical experiments comparing the modified algorithms with Algorithm 1 are reported in Section 3.

2 Modified Quasi-Newton Method

In this section we propose a modified quasi-Newton method for solving problem (2), which can be described as the following Algorithm 2.

Algorithm 2

Step 2.1: (initialization) Select an initial point $x_1 \in R^n$ *and a symmetric positive definite matrix* $H_1 \in R^{n\times n}$.

Step 2.2: (iteration) For $k = 1, 2, \ldots$ *until termination, do the following:*

2.2.1 Set two direction $s_k = -H_k g(x_k)$ *and* $t_k = H_k A^T b - x_k$

2.2.2 Find values of α_k *and* β_k *such that* (α_k, β_k) *is the exact minimum of the function*

$$\bar{f}(\alpha, \beta) = \frac{1}{2}[A(x_k + \alpha s_k + \beta t_k) - b]^T [A(x_k + \alpha s_k + \beta t_k) - b]$$

2.2.3 Set $x_{k+1} = x_k + \alpha_k s_k + \beta_k t_k$

2.2.4 Update H_k *to get* H_{k+1}.

As mentioned in Section 1, Algorithm 2 is a modification of Algorithm 1 motivated by the observation that $H_k A^T b$ can be considered an approximate solution of (2) since $H_k \approx (A^T A)^{-1}$. This observation implies that $t_k = H_k A^T b - x_k$ may be used as a "direction to move" from x_k towards the solution of the problem. Therefore, in steps 2.2.2 and 2.2.3 the two directions s_k and t_k are combined to generate the next point x_{k+1}.

Note that when Algorithm 1 with exact line search is used to solve the problem (2), the point x_{k+1} generated in step 1.2.3 is the minimum of $f(x)$ along the line $\{x = x_k + \alpha s_k \ \ |\alpha \in R\}$. With Algorithm 2, x_{k+1} generated in step 2.2.3 is the minimum of $f(x)$ in the plane $\{x = x_k + \alpha s_k + \beta t_k \ \ |\alpha, \beta \in R\}$.

The values of α_k and β_k in step 2.2.2 can be easily found by solving a 2-by-2 linear system

$$\begin{aligned} (s_k^T A^T A s_k)\alpha + (s_k^T A^T A t_k)\beta &= s_k^T A^T (b - A x_k) \\ (s_k^T A^T A t_k)\alpha + (t_k^T A^T A t_k)\beta &= t_k^T A^T (b - A x_k) \end{aligned} \tag{6}$$

In case s_k and t_k are linearly dependent, Algorithm 2 becomes Algorithm 1 with exact line search. In this case, steps 2.2.2 and 2.2.3 become the same as steps 1.2.2 and 1.2.3. The only coefficient α_k in step 1.2.2 will be the solution of the linear equation

$$(s_k^T A^T A s_k)\alpha = s_k^T A^T (b - A x_k) \tag{7}$$

When Broyden methods are used for updating H_k, Theorem 1.1 ensures that Algorithm 1 with exact line search will find the solution of (2) after $m \leq n$

iterations. When Broyden methods are used with Algorithm 2, the combination with t_k provides more reduction on $f(x)$ at each individual iteration. Therefore the number of iterations needed may be reduced although the global convergence is now not guaranteed.

In order to ensure the global convergence and further reduce the number of iterations, we may add the steepest descent direction into the combination. This further modification is described as the following Algorithm 3.

Algorithm 3

Step 3.1: (initialization) Select an initial point $x_1 \in R^n$ *and a symmetric positive definite matrix* $H_1 \in R^{n\times n}$.

Step 3.2: (iteration) For $k = 1, 2, \ldots$ *until termination, do the following:*

3.2.1 Set three direction $s_k = -H_k g(x_k)$, $t_k = H_k A^T b - x_k$, *and* $d_k = -\bigtriangledown f(x_k) = A^T(b - Ax_k)$

3.2.2 Find values of α_k, β_k, *and* θ_k *such that* $(\alpha_k, \beta_k, \theta_k)$ *is the exact minimum of the function*

$$\bar{f}(\alpha, \beta, \theta) = \frac{1}{2}[A(x_k + \alpha s_k + \beta t_k + \theta d_k) - b]^T [A(x_k + \alpha s_k + \beta t_k + \theta d_k) - b]$$

3.2.3 Set $x_{k+1} = x_k + \alpha_k s_k + \beta_k t_k + \theta_k d_k$

3.2.4 Update H_k *to get* H_{k+1}.

Note that the straightforward steepest descent direction method with exact line search yields

$$\bar{x}_{k+1} = x_k + c_k d_k \tag{8}$$

where the value of c_k is chosen so that $\bar{x}_{k+1}$ is the minimum of $f(x)$ along the line $\{x = x_k + cd_k \ \ | c \in R\}$. Comparing with x_{k+1} obtained in step 3.2.3 we have

$$f(x_{k+1}) \leq f(\bar{x}_{k+1}) \tag{9}$$

Therefore, if we imaginarily view step 3.2.3 as $x_k \longrightarrow \bar{x}_{k+1} \longrightarrow x_{k+1}$, then the part $x_k \longrightarrow \bar{x}_{k+1}$ serves as a steepest descent spacer and the part $\bar{x}_{k+1} \longrightarrow x_{k+1}$ offers a further improvement. Thus the spacer step theorem (see, for example, [5], pp 230–231) ensures the global convergence of Algorithm 3.

3 Numerical Experiments

In this section we report some numerical experiments comparing Algorithms 2 and 3 with Algorithm 1 with exact line search. The formula for updating H_k used in our experiments is the following BFGS formula, which is a typical member of the Broyden family (see, for example, [4] p55):

$$H_{k+1} = H_k + (1 + \frac{\gamma_k^T H_k \gamma_k}{\delta_k^T \gamma_k}) \frac{\delta_k \delta_k^T}{\delta_k^T \gamma_k} - (\frac{\delta_k \gamma_k^T H_k + H_k \gamma_k \delta_k^T}{\delta_k^T \gamma_k}) \tag{10}$$

where

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

and

$$\gamma_k = \bigtriangledown f(x_{k+1}) - \bigtriangledown f(x_k) = A^T A(x_{k+1} - x_k) \tag{12}$$

In our experiment, the initial matrix H_1 is chosen as $n \times n$ identity matrix I_n, and the initial point x_1 is chosen as

$$x_1 = \frac{b^T(AA^T)b}{b^T(AA^T)(AA^T)b} A^T b \tag{13}$$

which is the same as the x_1 used with the conjugate gradient method for solving problem (2). It is proven (see [4], p65) that with such H_1 and x_1, Algorithm 1 with BFGH formula (10) and exact line search is equivalent to the Fletcher–Reeves conjugate gradient method when applied to solve problem (2). Therefore, we are in fact also comparing Algorithms 2 and 3 with the conjugate gradient method.

Three systems of linear equations $Ax = b$ are used as test problems. The termination criteria for all three algorithms is $\|Ax - b\|_2 \leq \epsilon$ with $\epsilon = 10^{-10}$. The numerical experiments were conducted on a Pentium III Gateway Intel computer using Fortran/77 on a Linux operating system. Double precision was used. The number of iterations used by each algorithm is reported in the following Table 3.1.

For problem 1, $n = 100$ and

$$A = \begin{pmatrix} 101 & 1 & 1 & 1 & \dots & 1 \\ -1 & 102 & 1 & 1 & \dots & 1 \\ -1 & -1 & 103 & 1 & \dots & 1 \\ & & \dots & \dots & & \\ -1 & -1 & -1 & \dots & -1 & 200 \end{pmatrix} \qquad b = \begin{pmatrix} 200 \\ 199 \\ 198 \\ \dots \\ 101 \end{pmatrix}$$

The true solution of this problem is $x_* = (1, 1, \dots, 1)^T$. All three algorithms work equally well on this problem as shown in Table 3.1.

Problem 2 has $n = 32$ and

$$A = \begin{pmatrix} 31 & 1 & 1 & 1 & \dots & 1 & 0 & 0 \\ -1 & 32 & 1 & 1 & \dots & 1 & 0 & 0 \\ -1 & -1 & 33 & 1 & \dots & 1 & 0 & 0 \\ & & \dots & \dots & & & & \\ -1 & -1 & -1 & \dots & -1 & 60 & 0 & 0 \\ 0 & 0 & \dots & \dots & & 0 & 100 & -101 \\ 0 & 0 & \dots & \dots & & 0 & -1001 & 1000 \end{pmatrix} \qquad b = \begin{pmatrix} 60 \\ 59 \\ 58 \\ \dots \\ 31 \\ -1 \\ -1 \end{pmatrix}$$

The true solution of this problem is $x_* = (1, 1, \dots, 1)^T$ (with 32 elements). In this case, Algorithms 2 and 3 have worked slightly better than Algorithm 1.

Problem 3 is a modification of Problem 1, with $n = 102$ and

$$A = \begin{pmatrix} 101 & 1 & 1 & 1 & \dots & 1 & 0 & 0 \\ -1 & 102 & 1 & 1 & \dots & 1 & 0 & 0 \\ -1 & -1 & 103 & 1 & \dots & 1 & 0 & 0 \\ & & \dots & \dots & & & & \\ -1 & -1 & -1 & \dots & -1 & 200 & 0 & 0 \\ 0 & 0 & \dots & \dots & & 0 & 300 & -301 \\ 0 & 0 & \dots & \dots & & 0 & -301 & 300 \end{pmatrix} \qquad b = \begin{pmatrix} 200 \\ 199 \\ 198 \\ \dots \\ 101 \\ -1 \\ -1 \end{pmatrix}$$

The true solution of this problem is also $x_* = (1, 1, \dots, 1)^T$ (with 102 elements).

Note that for Problem 3, the condition number of the matrix $A^T A$ is very large (greater than or equal to $601^2 = 361201$). This is because $A^T A v_1 = 601^2 v_1$ and $A^T A v_2 = v_2$ with $v_1 = (0, \dots, 0, -1, 1)^T$ and $v_2 = (0, \dots, 0, 1, 1)^T$. Hence 601^2 and 1 are two of the eigenvalues of $A^T A$. Therefore, the conjugate gradient method is not expected to converge very fast with this problem. Table 3.1 shows that in this case Algorithm 2 takes more iterations than Algorithm 1 does, displaying a case where "gaining more at individual iterations may not necessarily result in a better global performance". However, Algorithm 3, the combination of all three directions, does provide a significant reduction of the number of iterations in this case.

Table 3.1 Number of Iterations

Problem	Algorithm 1	Algorithm 2	Algorithm 3
1	27	27	27
2	30	29	29
3	61	82	46

References

[1] Atkinson K. : *An introduction to numerical analysis*, John Wiley & Sons. New York, 1989.

[2] Bertsekas, D.P.: *Nonlinear programming*, Athena Scientific, Belmont, Massachusetts, 1995.

[3] Dennis, Jr.J.E. and Schnabel, R.B.: *Numerical methods for unconstrained optimization and nonlinear equations*, Prentice–Hall, Inc., Engleword Cliffs, New Jersey, 1983.

[4] Fletcher, R.: *Practical methods of optimization*, John Wiley & Sons, New York, 1987.

[5] Luenberger, D.G.: *Linear and nonlinear programming*, Addison–Wesley Publishing, 1984.

[6] Shi, Y.: Solving linear systems involved in constrained optimization, *Linear Algebra And Its Applications*, 229, 1995, pp. 175—189.

[7] Shi, Y.: Some computational aspects of interior point methods for linear, quadratic, and convex programmings, In G. Liu et al., editor, *Optimization techniques and applications*, World Scientific Publishing Co., 1995.

Received: September 14, 2006