OPTIMIZATION TECHNOLOGY CENTER
Argonne National Laboratory and Northwestern University

NUMERICAL EXPERIENCE WITH A REDUCED HESSIAN METHOD FOR LARGE SCALE CONSTRAINED OPTIMIZATION

by

Lorenz T. Biegler[1], *Jorge Nocedal*[2], *Claudia Schmid*[1] *and David Ternet*[1]

May 9, 1997

[1] Chemical Engineering Department, Carnegie Mellon University, Pittsburgh, PA 15213. These authors were supported by the Engineering Design Research Center, an NSF Center at Carnegie Mellon as well as through a grant from Bayer Corporation

[2] Department of Electrical and Computer Engineering, Northwestern University, Evanston Il 60208. Email: nocedal@ece.nwu.edu. This author was supported by National Science Foundation Grant CCR-9625613, and by Department of Energy Grant DE-FG02-87ER25047

NUMERICAL EXPERIENCE WITH A REDUCED HESSIAN METHOD FOR LARGE SCALE CONSTRAINED OPTIMIZATION

by

Lorenz T. Biegler, Jorge Nocedal, Claudia Schmid and David Ternet

ABSTRACT

The reduced Hessian SQP algorithm presented in [2] is developed in this paper into a practical method for large-scale optimization. The novelty of the algorithm lies in the incorporation of a correction vector that approximates the cross term $Z^T W Y p_{\mathrm{Y}}$. This improves the stability and robustness of the algorithm without increasing its computational cost. The paper studies how to implement the algorithm efficiently, and presents a set of tests illustrating its numerical performance. An analytic example, showing the benefits of the correction term, is also presented.

Key words: Successive Quadratic Programming, reduced Hessian methods, constrained optimization, Quasi-Newton method, large-scale optimization.

Abbreviated title: Numerical Experience with a Reduced Hessian Method

1. Introduction

This paper studies the implementation and numerical performance of the reduced Hessian algorithm described in Biegler, Nocedal and Schmid [2]. The algorithm is designed to solve large equality constrained optimization problems of the form

$$\min_{x \in \mathbf{R}^n} f(x) \tag{1.1}$$

$$\text{subject to } \; c(x) = 0, \tag{1.2}$$

where $f : \mathbf{R}^n \to \mathbf{R}$ and $c : \mathbf{R}^n \to \mathbf{R}^m$ are smooth functions. The convergence analysis given in [2] shows that the reduced Hessian method has a superlinear rate of convergence and is globally convergent under certain assumptions. This paper completes the exploration of the new algorithm by addressing the important implementation issues that make it effective in practice. We study how to balance the conflicting goals of robustness of the iteration and economy of computation, in the context of large-scale optimization. This paper also presents an analytic example illustrating the difficulties that the new reduced Hessian method is designed to overcome.

We begin with a brief outline of the Successive Quadratic Programming (SQP) method upon which our algorithm is based. At every iterate x_k, a search direction d_k is obtained by solving the quadratic subproblem

$$\min_{d\in\mathbf{R}^n} \; g(x_k)^T d + \frac{1}{2} d^T W(x_k) d \tag{1.3}$$

$$\text{subject to } \; c(x_k) + A(x_k)^T d = 0. \tag{1.4}$$

Here g denotes the gradient of f, W denotes the Hessian (with respect to x) of the Lagrangian function $L(x,\lambda) = f(x) + \lambda^T c(x)$, and A stands for the $n \times m$ matrix of constraint gradients,

$$A(x) = [\nabla c_1(x), ..., \nabla c_m(x)]. \tag{1.5}$$

The new iterate is given by $x_{k+1} = x_k + \alpha_k d_k$, where α_k is a steplength parameter that provides sufficient reduction in the ℓ_1 merit function

$$\phi_\mu(x) = f(x) + \mu\|c(x)\|_1, \qquad \mu > 0. \tag{1.6}$$

To compute the search direction d_k, we use the null-space approach (see e.g. [17]). We write the solution d_k of (1.3)-(1.4) as

$$d_k = Y_k p_{\rm Y} + Z_k p_{\rm Z}, \tag{1.7}$$

where Z_k is an $n \times (n-m)$ matrix spanning the null space of A_k^T, Y_k is an $n \times m$ matrix spanning the range of A_k, and $p_{\rm Y}$ and $p_{\rm Z}$ are vectors in $\mathbf{R}^m$ and $\mathbf{R}^{n-m}$, respectively. (We will assume throughout the paper that A_k has full column rank for all k.) Since $A_k^T Z_k = 0$, the linear constraints (1.4) become

$$c_k + A_k^T Y_k p_{\rm Y} = 0.$$

Therefore $p_{\rm Y}$ is given by

$$p_{\rm Y} = -[A_k^T Y_k]^{-1} c_k, \tag{1.8}$$

showing that d_k has the form

$$d_k = -Y_k[A_k^T Y_k]^{-1} c_k + Z_k p_{\rm Z}. \tag{1.9}$$

To determine the component $p_{\rm Z}$, we substitute (1.9) into (1.3), giving rise to the unconstrained subproblem

$$\min_{p_{\rm Z}\in\mathbf{R}^{n-m}} (Z_k^T g_k + Z_k^T W_k Y_k p_{\rm Y})^T p_{\rm Z} + \frac{1}{2} {p_{\rm Z}}^T (Z_k^T W_k Z_k) p_{\rm Z}, \tag{1.10}$$

where we have omitted constant terms involving $p_{\rm Y}$. Assuming that $Z_k^T W_k Z_k$ is positive definite, the solution of (1.10) is

$$p_{\rm Z} = -(Z_k^T W_k Z_k)^{-1}[Z_k^T g_k + Z_k^T W_k Y_k p_{\rm Y}]. \tag{1.11}$$

This determines the search direction d_k.

The most expensive parts of this algorithm are: (i) the computation of the null space and range space matrices Z and Y; (ii) the solution of the linear systems in (1.8) and (1.11); (iii) the evaluation of f, g, c, A and of the Hessian of the Lagrangian W. We now discuss each of these computations.

There are many possible choices for the basis matrices Z and Y. By computing a QR factorization of A, we can define Z and Y so as to have orthonormal columns. This gives a well conditioned representation of the null space and range space of A, but can be very expensive when the number of variables is large. We therefore make use of a more economical alternative in which Z and Y are defined by simple elimination of variables [12], [17]. To do this we group the components of x into m basic or dependent variables (which without loss of generality are assumed to be the first m variables) and $n-m$ nonbasic or control variables, and group the columns of A accordingly,

$$A(x)^T = [C(x)\ N(x)]. \tag{1.12}$$

The $m \times m$ *basis matrix* $C(x)$ is assumed to be nonsingular. Then we define

$$Z(x) = \begin{bmatrix} -C(x)^{-1}N(x) \\ I \end{bmatrix}, \quad \text{and} \quad Y(x) = \begin{bmatrix} I \\ 0 \end{bmatrix}; \tag{1.13}$$

we refer to this choice of Z and Y as *coordinate bases.*

Let us now consider the step computation (1.7), (1.8) and (1.11). Due to our choice (1.13) of Y, the component $p_{\rm Y}$ takes the simple form

$$p_{\rm Y} = -C_k^{-1}\, c_k. \tag{1.14}$$

The computation (1.11) of $p_{\rm Z}$ requires careful consideration. When the number of variables n is large and the number $n-m$ of *degrees of freedom* is small, it is attractive to approximate the reduced Hessian $Z_k^T W_k Z_k$ by means of a variable metric formula, such as BFGS, because this eliminates the computational work required to evaluate W_k and form $Z_k^T W_k Z_k$. For the same reason, it is appealing to avoid forming the matrix $Z_k^T W_k Y_k$, and some reduced Hessian methods [8, 13, 24, 19, 30, 14] simply omit the "cross term" $Z_k^T W_k Y_k p_{\rm Y}$ in (1.11). Deleting this term often works well when Z and Y are orthonormal [24, 19], but can lead to poor search directions when Z and Y are the coordinate basis (1.13) [29]. An example illustrating this phenomenon is given in §4. To ensure that good search directions are always generated, the algorithm presented in Biegler, Nocedal and Schmid [2] approximates the cross term $[Z_k^T W_k Y_k]p_{\rm Y}$ by a vector w_k,

$$[Z_k^T W_k Y_k]p_{\rm Y} \approx w_k, \tag{1.15}$$

without computing the matrix $Z_k^T W_k Y_k$. Two options are given for the correction vector w_k: the first computes a finite-difference approximation of $Z_k^T W_k Y_k$ along p_Y, which requires an extra evaluation of the gradient of the objective function f and constraints c; the second option updates a quasi-Newton approximation S_k to $Z_k^T W_k$, and defines $w_k = S_k Y_k p_Y$. The finite-difference option provides more accurate information but is more expensive than the quasi-Newton approximation. To obtain economy of computation, we define w_k via the quasi-Newton approximation S_k as often as possible, and only resort to finite-differences when this is required to ensure a superlinear rate of convergence [2].

To summarize, the algorithm does not require the computation of the Hessian of the Lagrangian W_k, and only makes use of first derivatives of f and c. The reduced Hessian matrix $Z_k^T W_k Z_k$ is approximated by a positive definite quasi-Newton matrix B_k, using the BFGS formula, and the cross term $Z_k^T W_k Y_k p_Y$ is approximated by a vector w_k, which is computed either by means of a finite-difference formula or via a quasi-Newton approximation. The algorithm is therefore well-suited for large problems with relatively few degrees of freedom. The novelty of the approach lies in the use of the correction term w_k. Because we wish to focus on the effects of this correction term, we only consider in this paper the simpler case when all constraints are equalities. We should note, however, that the ideas presented here can be used within an active set method for inequality constrained optimization such as that implemented in the SNOPT code [16]. For other work on reduced or full Hessian SQP methods for large-scale optimization see [1, 3, 10, 22, 25, 27, 23, 15].

2. The Implemented Algorithm

The broad overview just given of the reduced Hessian algorithm does not include a description of various devices introduced in [2] to globalize the iteration and ensure a fast rate of convergence. Other aspects of the algorithm that were not reviewed include the definition of Lagrange multiplier estimates and quasi-Newton updating formulae. The reader is advised to read §§1-3 of [2] before proceeding; the starting point of our discussion will be Algorithm II in that paper. We first describe some simplifications and modifications to Algorithm II that are designed to improve its efficiency in practice.

The computation of the Lagrange multiplier estimates (see equation (83) in [2]) takes the following simple form due to our choice (1.13) of Y,

$$\lambda_k = -C_k^{-T} g_k^c, \tag{2.1}$$

where g^c denotes the first m components of the vector g. Using coordinate bases also allows one to simplify the quasi-Newton update formula given in [2], which make use of the vectors

$$\begin{aligned} y_k &= Z_k^T[\nabla L(x_{k+1}, \lambda_{k+1}) - \nabla L(x_k, \lambda_{k+1})] - \bar{w}_k & (2.2) \\ \bar{y}_k &= Z_k^T[\nabla L(x_{k+1}, \lambda_{k+1}) - \nabla L(x_k, \lambda_{k+1})]; & (2.3) \end{aligned}$$

see (46) and (85) in [2]. As shown by Orozco [26], the equations $\nabla L(x,\lambda) = g(x) + A(x)\lambda$, $A(x)^T Z(x) = 0$, (1.13) and (2.1) imply that for any points $\bar{x}$ and $\tilde{x}$ for which $C(\bar{x})$ and $C(\tilde{x})$ are nonsingular,

$$Z(\tilde{x})^T \nabla L(\bar{x}, \lambda(\bar{x})) = Z(\bar{x})^T g(\bar{x}). \tag{2.4}$$

This and the equation $Z_k^T A_k = 0$ allow us to rewrite (2.2)-(2.3) as

$$y_k = Z_{k+1}^T g_{k+1} - Z_k^T g_k - \bar{w}_k, \qquad \bar{y}_k = Z_{k+1}^T g_{k+1} - Z_k^T g_k. \tag{2.5}$$

Our next observation concerns the calculation of the correction term $\bar{w}_k$ via finite differences (see (38) in [2]). This takes place after a new iterate x_{k+1} has been computed, and is given by

$$\bar{w}_k = Z_k^T [\nabla L(x_k + \alpha_k Y_k p_{\rm Y}, \lambda_{k+1}) - \nabla L(x_k, \lambda_{k+1})], \tag{2.6}$$

where α_k is the steplength used to ensure a sufficient reduction in the merit function. The drawback of this formula is that it requires an additional evaluation of the gradient of the Lagrangian if $\alpha_k \neq 1$; see §3.1 of [2]. To avoid this cost, we will redefine (2.6) as

$$\bar{w}_k = \alpha_k Z_k^T [\nabla L(x_k + Y_k p_{\rm Y}, \lambda_{k+1}) - \nabla L(x_k, \lambda_{k+1})]. \tag{2.7}$$

These two expressions are not equivalent, but one can show that in a neighborhood of a solution point they have very similar effects on the algorithm. More specifically one can show that the only part of the analysis given in [2] that is affected by the new definition (2.7) is Lemma 5.5 of that paper, and that the proof of that lemma can easily be modified to accept the new definition. We should stress, however, that away from the solution (2.7) may be a poor approximation to (2.6), and to ensure good performance of the algorithm we will only activate the finite-difference option when the algorithm appears to be approaching a solution point. This is a significant departure from Algorithm II in [2] and is motivated by the desire to reduce the cost of the iteration: finite-difference corrections are not particularly valuable compared to quasi-Newton approximations away from the solution – and are expensive in terms of gradient evaluations.

Next we should emphasize that the computation of the null space variable $p_{\rm Z}$ will be exactly as in [2],

$$p_{\rm Z} = -B_k^{-1} [Z_k^T g_k + \zeta_k w_k]. \tag{2.8}$$

The *damping parameter* $0 < \zeta_k \leq 1$ is one of the devices mentioned earlier on. Its purpose is to ensure that the search direction d_k is always a descent direction for the merit function; see §3.4 of [2]. It is shown in §6 of [2] that, as the algorithm approaches the solution, ζ_k always takes the value 1, so that (2.8) reduces to (1.11) asymptotically.

We are now ready to state the full algorithm, which is a modification of Algorithm II in [2]. Throughout the paper $\|\cdot\|$ denotes the Euclidean norm; we will also make use of the infinity norm $\|\cdot\|_\infty$.

Algorithm RHC. (Reduced Hessian Algorithm with Cross Term)

1. Choose constants Γ, γ_{fd}, Δ and sequences γ_k and $\bar{\gamma}_k$; see Table 1 for suggested values for these parameters. Set $k := 1$, choose a starting point x_1, and initialize the penalty parameter of the merit function (1.6) as $\mu_1 = 1$. Set the initial reduced Hessian approximation to $B_1 = I$ and initialize the Broyden approximation to the cross term as $S_1 = [0 \;\; I]$. (Postmultiplying this matrix by Z (as in $SZ \approx Z^T W Z \approx B$) and recalling the definition (1.13) of Z shows that this initialization is in agreement with the choice $B_1 = I$.)

2. Evaluate f_1, g_1, c_1 and A_1 at x_1, and compute Y_1 and Z_1 as defined by (1.13).

3. Set *findiff* = *false* and compute p_{Y} by solving the system

$$C_k p_{\mathrm{Y}} = -c_k. \tag{2.9}$$

4. Calculate w_k using Broyden's method,

$$w_k = S_k Y_k p_{\mathrm{Y}}, \tag{2.10}$$

and then set

$$w_k := \begin{cases} w_k & \text{if } \|w_k\| \le \Gamma \|p_{\mathrm{Y}}\|^{1/2} \\ w_k \frac{\Gamma \|p_{\mathrm{Y}}\|^{1/2}}{\|w_k\|} & \text{otherwise} \end{cases} \tag{2.11}$$

5. Choose the damping parameter ζ_k from

$$\zeta_k = \begin{cases} 1 & \text{if } g_k^T Z_k B_k^{-1} w_k \ge 0 \\ \min\left(\frac{-0.1 g_k^T Z_k B_k^{-1} Z_k^T g_k}{g_k^T Z_k B_k^{-1} w_k}, 1 \right) & \text{otherwise.} \end{cases} \tag{2.12}$$

and compute p_{Z} from

$$B_k p_{\mathrm{Z}} = -[Z_k^T g_k + \zeta_k w_k]. \tag{2.13}$$

6. Calculate $\sigma_k = \|Z_k^T g_k\| + \|c_k\|$. If $\tilde{\sigma} = \max\{\|Z_k^T g_k\|_\infty, \|c_k\|_\infty\} \le \Delta$ and if both

$$\|p_{\mathrm{Y}}\| \le \frac{\gamma_{fd} \|p_{\mathrm{Z}}\|}{\sigma_k^{1/2}} \tag{2.14}$$

and

$$\|p_{\mathrm{Y}}\| > \gamma_k^2 \|p_{\mathrm{Z}}\| \tag{2.15}$$

are satisfied, set *findiff* = *true* and recompute w_k from

$$w_k = Z_k^T [\nabla L(x_k + Y_k p_{\mathrm{Y}}, \lambda_k) - g(x_k)]. \tag{2.16}$$

(Unlike Algorithm II in [2], the calculation of the finite difference correction occurs only after the KKT error has been decreased beyond a certain threshold Δ, in order to avoid additional gradient evaluations far away from the solution.)

7. If *findiff* = *true* use this new value of w_k to choose the damping parameter ζ_k from equation (2.12) and recompute p_Z from equation (2.13).

8. Define the search direction by
$$d_k = Y_k p_Y + Z_k p_Z \tag{2.17}$$
and set $\alpha_k = 1$.

9. Test the line search condition
$$\phi_{\mu_k}(x_k + \alpha_k d_k) \leq \phi_{\mu_k}(x_k) + 0.1\alpha_k D\phi_{\mu_k}(x_k; d_k), \tag{2.18}$$
where
$$\phi_{\mu_k}(x_k) = f_k + \mu_k \|c_k\|_1. \tag{2.19}$$

10. If (2.18) is not satisfied, choose a new α_k from
$$\alpha_k = \max\left\{ \frac{-0.5 D\phi_{\mu_k}(x_k; d_k)\alpha_k^2}{\phi_{\mu_k}(x_k + \alpha_k d_k) - \phi_{\mu_k}(x_k) - \alpha_k D\phi_{\mu_k}(x_k; d_k)}, 0.1 \right\} \tag{2.20}$$
and go to 9; otherwise set
$$x_{k+1} = x_k + \alpha_k d_k. \tag{2.21}$$

11. Evaluate f_{k+1}, g_{k+1}, c_{k+1}, A_{k+1} at x_{k+1}, and compute Y_{k+1} and Z_{k+1}.

12. Compute the Lagrange multiplier estimate
$$\lambda_{k+1} = -C_{k+1}^{-T} g_{k+1}^c \tag{2.22}$$
and update μ_k by
$$\mu_{k+1} = \max(1.001 + \|\lambda_{k+1}\|_\infty, (3\mu_k + \|\lambda_{k+1}\|_\infty)/4, 10^{-6}) \tag{2.23}$$

13. Update S_{k+1} using the Broyden update
$$S_{k+1} = S_k + \frac{(\bar{y}_k - S_k \bar{s}_k)\bar{s}_k^T}{\bar{s}_k^T \bar{s}_k} \tag{2.24}$$
where
$$\bar{y}_k = Z_{k+1}^T g_{k+1} - Z_k^T g_k, \qquad \bar{s}_k = z_{k+1} - z_k \tag{2.25}$$
If *findiff* = *false* calculate $\bar{w}_k$ by Broyden's method through
$$\bar{w}_k = \alpha_k S_{k+1} Y_k p_Y \tag{2.26}$$
and set
$$\bar{w}_k := \begin{cases} \bar{w}_k & \text{if } \|\bar{w}_k\| \leq \alpha_k \|p_Y\|/\gamma_k \\ \bar{w}_k \frac{\alpha_k \|p_Y\|}{\gamma_k \|\bar{w}_k\|} & \text{otherwise.} \end{cases} \tag{2.27}$$

If *findiff* $=$ *true* calculate $\bar{w}_k$ by

$$\bar{w}_k = \alpha_k Z_k^T[\nabla L(x_k + Y_k p_{\rm Y}, \lambda_{k+1}) - g(x_k)] \tag{2.28}$$

and

$$\bar{w}_k := \begin{cases} \bar{w}_k & \text{if } \|\bar{w}_k\| \le \alpha_k \|p_{\rm Y}\|/\bar{\gamma}_k \\ \bar{w}_k \frac{\alpha_k \|p_{\rm Y}\|}{\bar{\gamma}_k \|\bar{w}_k\|} & \text{otherwise.} \end{cases} \tag{2.29}$$

(Another departure from Algorithm II in [2] is the addition of the bound (2.29) on the correction term computed by finite-differences; in our previous paper only the Broyden correction was safeguarded by (2.27). Since $\bar{\gamma}_k < \gamma_k$ (see Table 1) this bound is weaker than that imposed on the Broyden correction, but still gives greater stability to the algorithm.)

14. If $s_k^T y_k \le 0$ or if (2.14) is not satisfied, set $B_{k+1} = B_k$. Else, compute

$$\begin{aligned} s_k &= \alpha_k p_{\rm Z} \\ y_k &= Z_{k+1}^T g_{k+1} - Z_k^T g_k - \bar{w}_k \end{aligned}$$

and update B_{k+1} by the BFGS formula

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}. \tag{2.30}$$

15. If $\tilde{\sigma}_k = \max\{\|Z_k^T g_k\|_\infty, \|c_k\|_\infty\} \le tol$, STOP; else set $k := k+1$, and go to 3.

The convergence tolerance *tol* used to terminate the algorithm is set by the user. The numerical values for the other parameters used to obtain the results in § 5 are given in Table 1; $n - m$ is the number of independent (or control) variables of the problem and k is the iteration count. Our numerical testing suggests that the values in Table 1 give good performance.

As in [2] we retain σ_k in the decision rules for choosing the correction factor as this leads to a smooth measure of the KKT error. However, in order to determine the region in the neighborhood of the solution where we invoke the finite difference correction (and later) the watchdog line search, we prefer to use $\tilde{\sigma}_k$ as this is independent of the problem size.

Parameter	Reference	Suggested Value
Γ	2.11	20
γ_{fd}	2.14	10
γ_k	2.27	$0.1(n-m)^{0.25}k^{-1.1}$
$\bar{\gamma}_k$	2.29	$0.01(n-m)^{0.25}k^{-1.1}$
Δ	step 6	0.1

Table 1. Suggested value for parameters in Algorithm RHC.

3. Further Details of Implementation

Even though our description of the new reduced Hessian method has been quite extensive, it leaves out three important details of implementation: the choice of basic or control variables, the adjustment to the quasi-Newton matrices when the basis has changed, and the use of the watchdog technique to permit a fast rate of convergence. We now discuss these algorithmic features which have an important effect on performance.

3.1. Selection of the Basis

The algorithm requires the partition of the variables into dependent and independent variables. This determines the choice of basis C_k in (1.12) and the definitions of Z_k and Y_k. While many different selections are often possible, it is necessary that the basis matrix C_k be non-singular and not too ill-conditioned.

Let us begin by discussing the choice of initial basis. In some areas of application, such as optimal control, the choice is straightforward since the control variables constitute a suitable set of independent variables. But for most other optimization problems the selection is not easy, and a procedure that does this automatically is required. Our strategy is to supply the rectangular matrix A to the sparse linear equations solver MA28 from the Harwell library [20], and let this code select a nonsingular basis. (MA48, the successor of MA28 could also be used.) MA28 computes also sparse LU factors of the basis matrix C_k, thereby providing all the information needed to define Z_k and Y_k. We should note, however, that even though MA28 usually provides a good choice for the basis matrix, there is no guarantee of this. As described below, our code includes a test to determine if the basis is unsuitable, and a procedure for constructing a new one.

Once the initial basis has been determined, we attempt to keep the same choice of basic variables during all subsequent iterations because the quasi-Newton formulae that update B_k and S_k require that Z changes smoothly. It is necessary, however, to monitor the conditioning of the basis matrices C_k, and make a different selection if they are considered too ill-conditioned. Ideally the monitoring should be done by computing the condition number of C_k at every iteration, but this is too expensive when the number of variables is large. Instead we use the following heuristic.

At the current iterate x_k, compute

$$\beta_k = \max_{i,j}\{|(C_k^{-1}N_k)_{i,j}|\};$$

see (1.12) to recall the definition of N. W request a new basis from the sparse linear solver MA28 if

$$\beta_k > 10 \times \beta_{k-1}$$

or if

$$\alpha_{k-1} < 10^{-3} \qquad \text{and} \qquad \beta_k > \beta_{k-1},$$

where α_{k-1} is the steplength used at iteration $k-1$.

This heuristic attempts to determine whether the inverse of C_k is growing rapidly or whether a poor choice of basis has produced an excessively large search direction that forced the line search procedure to generate a very small steplength.

To request a new basis from MA28 we provide it with the rectangular matrix A_k and invoke MA28's analyze phase, which returns a nonsingular basis matrix. There is no guarantee, however, that MA28 will return a different basis. In this (unlikely) case our code continues the iteration with this choice of basis, in spite of the possibility of ill-conditioning. Constructing a more exhaustive procedure that will force MA28 to consider another basis is a refinement that our code does not yet include.

3.2. Adjusting the Quasi-Newton Matrices

Suppose that at iterate x_{k+1} we have decided to make a new selection of basic and independent variables. The new partitioning affects the definitions (1.13) of the null space and range space basis Z and Y, which will therefore change abruptly from x_k to x_{k+1}. This will have a detrimental effect in the quasi-Newton updating formulae (2.24), (2.30) which involve differences in Z. The simplest strategy to overcome this difficulty would be to reinitialize the BFGS and Broyden matrices B_{k+1} and S_{k+1} as in step 1 of Algorithm RHC. But this is not always desirable since it leads to a loss of the curvature information gathered in these matrices – information that makes a rapid rate of convergence possible. Therefore we have developed the following strategy, which is similar to that used by Xie [30], for modifying the previous matrices B_k and S_k (so as to reflect the new choice of basis) before the update leading to B_{k+1} and S_{k+1} is performed.

To simplify the discussion that follows, we drop the iteration subscript k from all matrices. The constraint Jacobian A_k, given the partition of the variables at x_k, will be written as $A = [C \quad N]$. Let $\bar{A}^T = [\bar{C} \quad \bar{N}]$ be the matrix obtained by permuting some of the columns of A so that the columns corresponding to the new set of basic variables appear in the first m positions. We can therefore write

$$\bar{A}^T = A^T P \tag{3.1}$$

where P is a permutation matrix. The null space bases for the old and new choice of basis are given by

$$Z = \begin{bmatrix} -C^{-1}N \\ I \end{bmatrix} \qquad \text{and} \qquad \bar{Z} = \begin{bmatrix} -\bar{C}^{-1}\bar{N} \\ I \end{bmatrix}; \tag{3.2}$$

thus $A^T Z = 0$ and $\bar{A}^T \bar{Z} = 0$. Using (3.1) we have

$$\bar{A}^T \bar{Z} = A^T (P\bar{Z}) = 0, \tag{3.3}$$

showing that the columns of the matrix $P\bar{Z}$ also lie in the null space of A^T. Therefore there exists a square nonsingular matrix R of dimension $n - m$ such that

$$P\bar{Z} = ZR. \tag{3.4}$$

Now, the Hessian of the Lagrangian with respect to the current ordering of the variables has been denoted by W. When the selection of basic and nonbasic variables changes, the Hessian of the Lagrangian must reflect this new ordering of the variables. If we denote this new Hessian by $\bar{W}$, we have that

$$\bar{W} = P^T W P.$$

Since the current reduced Hessian matrix B approximates $Z^T W Z$, our goal is to find a new matrix $\bar{B}$ that approximates $\bar{Z}^T \bar{W} \bar{Z}$. Using (3.4) we have

$$\begin{aligned} \bar{B} \simeq \bar{Z}^T \bar{W} \bar{Z} &= \bar{Z}^T P^T W P \bar{Z} \\ &= R^T Z^T W Z R. \end{aligned} \tag{3.5}$$

This suggests that $\bar{B}$ be defined by $\bar{B} = R^T B R$. Nevertheless computing R is in general very expensive, and we therefore eliminate it by means of the following operations. Let $T = [0 \;\; I]$ be an $(n-m) \times n$ matrix partitioned so that $TZ = I$. Using (3.5) and (3.4) we obtain

$$\bar{Z}^T \bar{W} \bar{Z} = R^T Z^T T^T Z^T W Z T Z R \tag{3.6}$$
$$= (ZR)^T T^T (Z^T W Z) T (ZR) \tag{3.7}$$
$$= \bar{Z}^T P^T T^T (Z^T W Z) T P \bar{Z}. \tag{3.8}$$

This suggests the formula

$$\bar{B} = \bar{Z}^T P^T T^T B T P \bar{Z} \tag{3.9}$$

for computing the reduced Hessian approximation using the new null space basis $\bar{Z}$.

A similar derivation shows that a good choice of the Broyden matrix at x_k, for the the new selection of basis variables, is given by

$$\bar{S} = \bar{Z}^T P^T T^T S P. \tag{3.10}$$

Having redefined B_k and S_k as $\bar{B}_k$ and $\bar{S}_k$ we can now safely apply the updating formula (2.24), (2.30) to obtain B_{k+1} and S_{k+1}. Our numerical experience indicates that this strategy for adjusting the quasi-Newton matrices after a change of basis is beneficial compared to the simple alternative of resetting the matrices to their default values.

3.3. The Watchdog Technique

One of the requirements for 1-step superlinear convergence of Algorithm RHC is that the line search technique must allow full steps ($\alpha_k = 1$) close to the solution [2]. When a smooth merit function such as Fletcher's differentiable function [11] is used, it is not difficult to show that near the solution unit steplengths give a sufficient reduction in the merit function and will be accepted. However, the nondifferentiable exact penalty function (2.19) used in our algorithm may reject unit steplengths as the iterates approach the solution; this is commonly referred to as the Maratos effect. To circumvent this problem we use the non-monotone line search (or watchdog technique) of Chamberlain et al [7], as described in Byrd and Nocedal [6].

Even though the idea behind the watchdog technique is conceptually simple, a full description can be complicated. Therefore we will now only outline the main steps and refer the reader to [6] for a more details.

At the current iterate x_k we compute the search direction d_k (step 8 of Algorithm RHC). If $x_k + d_k$ gives sufficient reduction in the merit function, as measured by (2.18), we set $x_{k+1} = x_k + d_k$ and go to step 11 of Algorithm RHC. Otherwise we initiate the watchdog procedure. At $\hat{x} = x_k + d_k$ we compute a new search direction $\hat{d}$ and the new point $x' = \hat{x} + \hat{\alpha}\hat{d}$, where $\hat{\alpha}$ is a steplength giving a sufficient reduction of the merit function along $\hat{d}$. If x' gives sufficient reduction *with respect to* x_k, then we set $x_{k+1} = x'$ and terminate the watchdog procedure. Otherwise we compare the merit function values at x' and x_k. If x' has a lower merit function value, we compute yet another search direction from x', perform a line search along it enforcing the sufficient decrease condition (2.18), and define the new iterate as x_{k+1}. On the other hand, if the merit function value at x_k is smaller than at x', we fall back to x_k and perform a line search along d_k to obtain the new iterate x_{k+1} giving a sufficient reduction in the merit function. This terminates the watchdog procedure.

As other authors have observed (see [28] and the references therein) it is not advantageous to use a non-monotone line search in the first few iterations of an algorithm, before the correct scale of the problem has been determined. In the tests reported in §5, we activate the watchdog technique only after the KKT error $\tilde{\sigma}_k$ is below a user-specified tolerance.

4. An Analytic Example

We present a simple example that shows that for a practical choice of basis Z, Y, the reduced Hessian algorithm can produce very poor steps if the cross term $Z^T WYp_Y$ is omitted. The example also shows that this undesirable behavior disappears if the term $Z^T WYp_Y$ is included.

Let us write $x^T = [u, v]$ and consider the problem

$$\min f(x) = \frac{1}{2}(u^2 + v^2) \tag{4.1}$$

$$\text{subject to } \; c(x) = u(v-1) - \theta v = 0, \tag{4.2}$$

where θ is an adjustable parameter. It is easy to see that $u_* = v_* = \lambda_* = 0$ is a solution, and that

$$\nabla^2_{xx} L(x_*, \lambda_*) = I. \tag{4.3}$$

To simplify the calculations, we will not update B_k by the BFGS formula, and instead define it as $B_k = Z_k^T Z_k$; due to (4.3), this is a very good approximation of the true reduced Hessian $Z_k^T W_k Z_k$ near the solution.

Let us first apply Algorithm RHC, with no correction terms w_k and $\overline{w}_k$, but using *orthogonal* bases satisfying $Z_k^T Y_k = 0$, $Z_k^T Z_k = I_{n-m}$, $Y_k^T Y_k = I_m$. If the current iterate is of the form $x_k^T = [u_k, v_k] = [\delta, \delta]$, we find (see Appendix) that

$$\lim_{\delta \to 0} \frac{\|x_k + d_k\|}{\|x_k\|^2} = \frac{1}{2(\theta^2 + 1)^{1/2}}. \tag{4.4}$$

Thus we obtain a quadratic contraction in the error, if x_k is near the solution (note that since $x_* = 0$ the error $x_k - x_*$ is given by x_k).

In contrast, if we apply Algorithm RHC using the **coordinate** basis (1.13), we find that for the same iterate x_k

$$\lim_{\delta \to 0} \frac{\|x_k + d_k\|}{\|x_k\|} = \frac{\theta(1+\theta)}{\sqrt{2(1+\theta^2)}}.$$

This ratio will be large when θ is large, showing that the step from x_k to $x_k + d_k$ can be very poor. The algorithm may not accept this point, since for at least some values of θ the merit function increases at $x_k + d_k$, and the line search will be forced to compute a small steplength – at the cost of several function evaluations. Now, if the point $x_k + d_k$ were to be accepted, then at the next iterate we would have (see Appendix)

$$\lim_{\delta \to 0} \frac{\|x_{k+2}\|}{\|x_k\|} = 0,$$

which is consistent with the property of 2-step Q-superlinear convergence of this method demonstrated by Byrd [5] and Yuan [31]. But it would be too risky to enforce descent in the merit function only every two iterations because, as the example shows, the first step can be arbitrarily bad.

Finally, let us consider Algorithm RHC with a **coordinate** basis and using the **correction term** w_k. (The term $\overline{w}_k$ is not needed since BFGS updating is not used.) Using the same starting point one can show (see Appendix) that the behavior of the algorithm is very similar to that obtained with an orthogonal basis, and that quadratic contraction (4.4) takes place. Thus the correction term has a dramatic effect on the method in this case.

5. Numerical Results

As noted in the introduction, the novelty of Algorithm RHC lies in the use of the correction term, whose goal is to give stability to the iteration when the bases Y and Z do not provide a well conditioned representation of $\mathbf{R}^n$. The correction term cannot be expected to improve the performance of the algorithm when the basis matrices are adequate, but should prevent poor performance from taking place. We will begin with some experiments that test the algorithm under rather unfavorable circumstances. Then we will observe its overall behavior on some problems from the well known CUTE and Hock-Schittkowski collections.

We tested Algorithm RHC and two variants of it. The standard implementation of Algorithm RHC computes the correction vector by the Broyden or finite-difference approaches. In the first variation, labeled "No Correction", the correction term was completely eliminated from the algorithm, which then resembles a classical reduced Hessian method. In the version labeled "Broyden Correction", the correction vector was always computed via Broyden's method. These variants were introduced to better understand the practical behavior of the algorithm. The convergence tolerance in Step 15 was set to $tol = 10^{-5}$. The watchdog and finite difference corrections were activated when $\tilde{\sigma}_k < 0.1$. All tests were performed on a DEC-ALPHA 3000-400, in double precision FORTRAN.

The first two test problems are generalizations of problem (4.1)-(4.2) of §4: in Example 2 we increase the problem size but maintain the number of degrees of freedom at 1, and in Example 3 both the number of variables and the number of degrees of freedom are increased. The starting point and solution points are denoted by x^0 and x^*, respectively, and in the following description subscripts refer to components of a vector.

Example 2 (1 degree of freedom)

$$\begin{aligned} &\min \tfrac{1}{2} \textstyle\sum_{i=1}^{n} x_i^2 \\ \text{s.t.}\quad & x_1(x_{j+1} - 1) - 10x_{j+1} = 0 \quad j = 1, ..., n-1 \\ & x_j^0 = 0.1 \qquad x_j^* = 0. \end{aligned}$$

Example 3 (Number of degrees of freedom = $n/2$)

$$\begin{aligned} &\min \tfrac{1}{2} \textstyle\sum_{i=1}^{n} x_i^2 \\ \text{s.t.}\quad & x_j(x_{n/2+j} - 1) - 10x_{n/2+j} = 0, \quad j = 1, ..., n/2 \\ & x_j^0 = 0.1 \qquad x_j^* = 0. \end{aligned}$$

For these two test problems we dictated the choice of basis, and the algorithm was not allowed to change this choice. The analysis in the previous section indicates how to choose good and bad bases.

Number of variables	No Correction	Broyden Correction	Algorithm RHC
	Ind. variable $= x_1$ (Good choice)		
80	9 (9/ 9)	9 (9/ 9)	8 (8/11)
200	12 (13/12)	10 (11/10)	9 (10/13)
	Independent variable $= x_2$ (Poor choice)		
80	19 (34/19)	9 (12/ 9)	8 (11/10)
200	12 (19/12)	7 (11/ 7)	7 (11/ 9)

Table 2. Results for Example 2.
No. of iterations (No. function eval./No. of gradient eval.)

Number of variables	No Correction	Broyden Correction	Algorithm RHC
	Ind. variable $= x_1$ (Good choice)		
80	6 (6/6)	6 (6/6)	6 (6/6)
200	6 (6/6)	6 (6/6)	6 (6/6)
	Independent variable $= x_2$ (Poor choice)		
80	27 (39/27)	19 (28/19)	17 (21/18)
200	25 (36/25)	19 (26/19)	18 (22/19)

Table 3. Results for Example 3
No. of iterations (No. function eval./No. of gradient eval.)

For the good choice of basis, inclusion of the correction term has little effect on performance in both examples, but for the poor choice of basis the correction is highly beneficial. These results are consistent with the analysis given in [2] which shows that the correction term improves the local convergence rate only when the path followed by the iterates is not tangential to the constraints.

In the next set of tests, given in Table 4, we report the performance of Algorithm RHC on some of the problems from the Hock-Schittkowski collection [21]. Since these problems are of small dimension, we do not report execution times. In these and the rest of experiments reported in this paper, the algorithm chooses the basis freely, as described in §3.

Problem	N/M	No Correction	Broyden Correction	Algorithm RHC
HS 80	5/3	19(25/ 19)	11(11/ 11)	9(9/ 15)
HS 81	5/3	24(38/ 24)	11(11/ 11)	9(9/ 15)
HS 99	7/2	15(18/ 15)	16(28/ 17)	16(28/ 19)
HS100	7/4	21(29/ 21)	20(29/ 20)	20(29/ 22)
HS101	7/2	54(93/ 54)	43(69/ 43)	43(69/ 47)
HS102	7/3	102(118/105)	105(173/106)	101(167/107)
HS103	7/4	96(208/ 99)	119(221/121)	117(218/129)
HS104	8/4	34(87/ 36)	29(70/ 31)	29(70/ 39)
HS111	10/3	59(75/ 61)	48(55/ 49)	49(57/ 67)
HS112	10/3	36(66/ 36)	33(60/ 33)	33(60/ 33)
HS113	10/6	13(16/ 13)	15(19/ 15)	15(19/ 17)

Table 4. Results on several problems from Hock and Schittkowski [21].
No. of iterations (No. function eval./No. of gradient eval.)

In these problems the standard implementation of Algorithm RHC outperforms the option that does not make use of the correction term, in terms of iterations and function evaluations. But taking into account the number of gradient evaluations, the option that always computes the correction by Broyden's method, appears to be quite efficient.

We now consider some challenging test problems from the CUTE collection [4]. They were selected for their difficulty and variety (see [22]), and allow us to test most of the devices of Algorithm RHC. The results are given in Table 5.

Problem	N/M	No Correction	Broyden Correction	Algorithm RHC
	30/15	32(57/32/0.9)	32(59/32/1.0)	32(59/39/1.0)
EIGENC2	56/28	49(91/49/1.5)	58(116/59/2.1)	58(116/72/2.3))
	90/45	60(114/61/3.1)	67(125/67/3.5)	67(125/84/3.9)
	30/15	32(56/32/0.9)	33(57/33/1.0)	33(57/41/1.1)
EIGENCCO	56/28	58(117/59/2.3)	44(80/45/2.0)	45(81/56/2.2)
	90/45	69(133/70/4.1)	64(123/65/4.3)	65(122/88/4.9)
	37/16	110(262/113/ 2.5)	91(189/ 92/ 2.7)	91(189/97/ 2.7)
ORTHREGA	133/64	185(417/191/ 36.0)	308(608/313/ 72.3)	308(608/322/ 72.6)
	517/256	341(864/358/1294.5)	298(681/312/1370.6)	298(681/338/1379.9)
	205/100	118(239/122/ 30.1)	51(92/ 52/ 19.4)	49(84/ 65/ 19.8)
ORTHREGC	305/150	81(135/ 82/ 43.0)	90(185/ 93/ 81.2)	89(183/137/ 86.9)
	405/200	79(107/ 81/ 96.0)	123(181/126/196.2)	123(181/182/207.2)
	505/250	144(297/149/258.6)	108(193/109/246.0)	107(185/170/261.0)
	23/10	23(26/23/ 0.2)	20(24/20/ 0.3)	25(30/40/ 0.3))
ORTHREGD	103/50	28(35/28/ 1.4)	24(30/24/ 2.0)	29(38/48/ 2.0))
	203/100	33(42/33/ 6.5)	28(36/28/10.1)	23(27/37/10.3)
	303/150	26(30/26/14.0)	23(26/23/27.9)	33(41/55/28.1)

Table 5. Results on several problems from the CUTE collection [4].
No. of iterations (No. function eval./No. of gradient eval./CPU secs)

Problems EIGENC2 and EIGENCCO, have a quadratic objective function and quadratic constraints; they are reformulations of symmetric eigenvalue problems as a systems of nonlinear equations. These runs require a change of basis to avoid poorly conditioned bases, and our strategy for handling bases changes performed effectively. The initial point in these problems satisfies the equality constraints; hence the correction terms are relatively small and the performance of all options of the algorithm is similar.

ORTHREGA, ORTHREGC and ORTHREGD are orthogonal regressions problems [18] where the objective is to fit orthogonally a cardioid to a set of points in the plane. Frequent basis changes were necessary to avoid ill-conditioned Jacobian matrices. Here the correction vectors are not small relative to the step, and larger differences in performance are observed among the three options. In these problems, the number of degrees of freedom is large relative to the size of the problem (about half the variables are independent variables). Even though this leads to large Broyden matrices we observe that computing time is not greatly increased by the Broyden approximations. Note that the number of gradient evaluations is the least for the pure Broyden option.

Taken as a whole these numerical tests indicate, first of all, that Algorithm RHC is robust and efficient, and capable of dealing well with ill-conditioned problems. The results also indicate that the inclusion of the correction term provides stability to the iteration, and does not give rise to a significant increase in function evaluations, or in CPU time. The results also suggest that if the evaluation of gradients is expensive, it may be advantageous to compute the correction term always by Broyden's method. We conclude that a correction term developed in this paper may be a very useful feature in any reduced Hessian SQP method for constrained optimization.

6. Conclusions

This research was motivated by numerical tests on chemical process control problems performed, a few years ago, by the first author. The algorithm was of the reduced Hessian SQP type and used non-orthogonal bases. He observed that the algorithm was sometimes very inefficient, and an examination of the results revealed that poor search directions were being generated. This suggested that an approximation to the cross term $Z^TWYp_{\rm Y}$ might resolve the difficulties, and the result of this investigation was the algorithm analyzed in [2] and developed here into a practical method for large-scale equality constrained optimization. We have introduced various features in the algorithm that ensure its robustness and speed, while at the same time keeping the computational cost and evaluations of gradients at the same level as that of a standard SQP method.

7. Appendix: Derivation of the Results of §4

We have already noted that $u_* = v_* = \lambda_* = 0$ is a solution of (4.1)-(4.2) for any θ, and that $\nabla^2_{xx}L(x_*, \lambda_*) = I$. We also have that

$$A(x) = \begin{bmatrix} v-1 \\ u-\theta \end{bmatrix}, \quad A_* = \begin{bmatrix} -1 \\ -\theta \end{bmatrix}.$$

Let us first choose

$$Z(x) = \begin{bmatrix} \theta - u \\ v-1 \end{bmatrix} \qquad Y(x) = \begin{bmatrix} 1 \\ (u-\theta)/(v-1) \end{bmatrix}. \tag{7.1}$$

Note that Z is a smooth function, and that Y and Z are mutually orthogonal (i.e. $Y(x)^TZ(x) = 0$), but the the columns of Y and Z are not of norm 1.

Since near the solution $W_k \approx I$, we have that $Z_k^TW_kY_k \approx 0$, it is appropriate to define the correction term w_k to be zero for all k. The matrix B_k will not updated by the BFGS formula, but will be given instead by $B_k = Z_k^TZ_k$; therefore the correction $\bar{w}_k$ need not be defined. From (1.8) and (2.8) we have

$$p_{\rm Y} = \frac{(v_k-1)(\theta v_k - u_k(v_k-1))}{(\theta-u_k)^2 + (v_k-1)^2}$$

$$p_{\rm Z} = -\frac{(\theta-u_k)u_k + (v_k-1)v_k}{(\theta-u_k)^2+(v_k-1)^2}, \tag{7.2}$$

and therefore (1.9) gives

$$d_k = \frac{1}{(\theta - u_k)^2 + (v_k - 1)^2} \begin{bmatrix} ((v_k - 1) - (\theta - u_k)^2)\, u_k \\ (\theta(u_k - \theta) - (v_k - 1)^2)\, v_k \end{bmatrix}.$$

Let x_k be of the form $x_k^T = [u_k, v_k] = [\delta, \delta]$. Then

$$x_{k+1} = x_k + d_k = \frac{1}{(\theta - \delta)^2 + (\delta - 1)^2} \begin{bmatrix} \delta^2(\delta - 1) \\ \delta^2(\delta - \theta) \end{bmatrix},$$

and we see that for any $\theta \neq \delta$,

$$\lim_{\delta \to 0} \frac{\|x_k + d_k\|}{\|x_k\|^2} = \frac{1}{2(\theta^2 + 1)^{1/2}}. \tag{7.3}$$

Thus, near the solution, and when Y and Z are mutually orthogonal we obtain a quadratic contraction in the error.

Let us now suppose that Z is defined by (7.1), but that Y is given by

$$Y(x)^T = [1, 0].$$

We no longer have that $Y(x)^T Z(x) = 0$, and in fact, for large values of θ, this vector Y is almost orthogonal to the one defined in (7.1). The null-space component p_Z is still given by (7.2), but now

$$p_Y = \frac{\theta v - u(v - 1)}{v - 1}. \tag{7.4}$$

Substituting into (1.9) gives

$$d_k = \begin{bmatrix} -u \\ 0 \end{bmatrix} + \frac{1}{(v-1)^3 + (v-1)(\theta - u)^2} \begin{bmatrix} (\theta - u)^2(u + v(\theta - u)) + uv(v-1)^2 \\ -(\theta - u)(v-1)^2 u - (v-1)^3 v \end{bmatrix}$$

so that $x_{k+1} = x_k + d_k$ is given by

$$x_{k+1} = \frac{1}{(v-1)^3 + (v-1)(\theta - u)^2} \begin{bmatrix} (\theta - u)^2(u + v(\theta - u)) + uv(v-1)^2 \\ (\theta - u)(v-1)[(\theta - u)v - (v-1)u] \end{bmatrix}. \tag{7.5}$$

Now letting $x_k^T = [u_k, v_k] = [\delta, \delta]$ we obtain

$$x_{k+1} = \frac{1}{(\delta - 1)^3 + (\delta - 1)(\theta - \delta)^2} \begin{bmatrix} (\theta - \delta)^2(1 + \theta - \delta)\delta + \delta^2(\delta - 1)^2 \\ (\theta - \delta)^2(\delta - 1)\delta - (\theta - \delta)(\delta - 1)^2\delta \end{bmatrix}$$

or

$$x_{k+1} \equiv \delta \begin{bmatrix} a(\delta) \\ b(\delta) \end{bmatrix},$$

where $a(\delta)$ and $b(\delta)$ are rational functions of δ. Using (7.5), and abbreviating $a = a(\delta)$ and $b = b(\delta)$, we have at the next iteration

$$x_{k+2} = \frac{1}{(b\delta-1)^3 + (b\delta-1)(\theta-a\delta)^2} \begin{bmatrix} (\theta-a\delta)^2(a\delta+\delta b(\theta-a\delta)) + \delta^2 ab(b\delta-1)^2 \\ (\theta-a\delta)^2(b\delta-1)b\delta - (\theta-a\delta)(b\delta-1)^2 a\delta \end{bmatrix}.$$

Letting $\delta \to 0$ and recalling that a and b are functions of δ, we obtain

$$\lim_{\delta\to 0} \frac{\|x_{k+2}\|}{\|x_k\|} = \frac{1}{\sqrt{2}(1+\theta^2)} \left\| \begin{bmatrix} -\theta^2(a(0)+\theta b(0)) \\ \theta(a(0)+\theta b(0)) \end{bmatrix} \right\|$$

Now since

$$\begin{bmatrix} a(0) \\ b(0) \end{bmatrix} = \lim_{\delta\to 0} \begin{bmatrix} a(\delta) \\ b(\delta) \end{bmatrix} = \frac{1}{(1+\theta^2)} \begin{bmatrix} -\theta^2(1+\theta) \\ \theta(1+\theta) \end{bmatrix}$$

we have that

$$\lim_{\delta\to 0} \frac{\|x_{k+2}\|}{\|x_k\|} = 0,$$

which is consistent with the property of 2-step Q-superlinear property demonstrated in [5, 31]. In contrast

$$\lim_{\delta\to 0} \frac{\|x_{k+1}\|}{\|x_k\|} = \frac{\theta(1+\theta)}{\sqrt{2(1+\theta^2)}},$$

showing that the step from x_k to x_{k+1} is poor.

Finally, we now consider algorithm RHC with the coordinate range space basis matrix $\tilde{Y}(x)$ and with a correction term of the form $w_k = Z_k^T W_k \tilde{Y}_k p_{\rm Y}$. As with the uncorrected method, we have:

$$p_{\rm Y} = -(A_k^T \tilde{Y}_k)^{-1} c_k = \frac{\theta v - u(v-1)}{v-1}.$$

However, $p_{\rm Z}$ is now given by

$$p_{\rm Z} = -(Z_k^T \tilde{W}_k Z_k)^{-1}(Z_k^T g_k + Z_k^T \tilde{W}_k \tilde{Y}_k p_{\rm Y})$$

or

$$p_{\rm Z} = -\frac{\theta v_k(\theta-u_k) + v_k(v_k-1)^2}{(v_k-1)(\theta-u_k)^2 + (v_k-1)^3}.$$

Evaluation of the step $d_k = Z_k p_{\rm Z} + \tilde{Y}_k p_{\rm Y}$ leads to the following equation for x_{k+1}

$$x_{k+1} = \frac{1}{(\theta-u_k)^2 + (v_k-1)^2} \begin{bmatrix} u_k v_k(v_k-1) \\ u_k v_k(u_k-\theta) \end{bmatrix} \tag{7.6}$$

which, for this example, is identical to the result obtained for the reduced Hessian method with orthogonal bases.

8. *

References

[1] J.T. Betts and P.D. Frank, *A sparse nonlinear optimization algorithm*, JOTA, 82, (1994), pp. 519-541.

[2] L. T. Biegler, J. Nocedal, and C. Schmid, *A reduced Hessian method for large-scale constrained optimization*, SIAM J. Optimization, 5, 2, (1995), pp. 314–347.

[3] P.T. Boggs, J.W. Tolle, and A.J. Wang, *A practical algorithm for general large scale nonlinear optimization problems*, Internal Report, National Institute of Standards, 1994.

[4] I. Bongartz, A.R. Conn, N.I.M. Gould, and Ph.L. Toint, *CUTE: constrained and unconstrained testing environment*, Research Report, IBM T.J. Watson Research Center, Yorktown Heights, NY, 1993.

[5] R. H. Byrd, *An example of irregular convergence in some constrained optimization methods that use the projected Hessian*, Math. Programming, 32 (1985), pp. 232–237.

[6] R. H. Byrd and J. Nocedal, *An analysis of reduced Hessian methods for constrained optimization*, Math. Programming, 49 (1991), pp. 285–323.

[7] R. M. Chamberlain, C. Lemarechal, H. C. Pedersen, and M. J. D. Powell, *The watchdog technique for forcing convergence in algorithms for constrained optimization*, Math. Programming Studies, 16 (1982), pp. 1–17.

[8] T. F. Coleman and A. R. Conn, *On the local convergence of a quasi-Newton method for the nonlinear programming problem*, SIAM J. Numer. Anal., 21 (1984), pp. 755–769.

[9] I.S. Duff, A.M. Erisman, and J.K. Reid, *Direct Methods for Sparse Matrices*, Clarendon Press, Oxford, 1986.

[10] S.K. Eldersveld, *Large-scale sequential quadratic programming algorithms*, Ph.D. thesis, Department of Operations Research, Stanford University, Stanford, CA., 1991.

[11] R. Fletcher, *An exact penalty for nonlinear programming with inequalities*, Math. Programming, 5 (1973), pp. 129–150.

[12] R. Fletcher, *Practical Methods of Optimization*, (second edition), John Wiley and Sons, Chichester, 1987.

[13] D. Gabay, *Reduced quasi-Newton methods with feasibility improvement for nonlinearly constrained optimization*, Math. Programming Studies, 16 (1982), pp. 18–44.

[14] J. C. Gilbert, *On the local and global convergence of a reduced quasi-Newton method*, Optimization, 20 (1989), pp. 421–450.

[15] J. C. Gilbert, *Maintaining the positive definiteness of the matrices in reduced Hessian methods for equality constrained optimization*, Math. Programming, 50 (1991), pp. 1–28.

[16] P.E. Gill, W. Murray, and M. Saunders, *An SQP algorithm for large scale optimization*, working paper, EESOR Department, Stanford University, 1996.

[17] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*, Academic Press, London, 1981.

[18] M. Gulliksson, *Algorithms for nonlinear least squares with applications to orthogonal regression*, UMINF-178.90, University of Umea, Sweden, 1990.

[19] C. B. Gurwitz and M. L. Overton, *SQP methods based on approximating a projected Hessian matrix*, SIAM. J. Sci. Stat. Comp., 10 (1989), pp. 631–653.

[20] Harwell Subroutine Library. *A catalogue of subroutines (release 12).* AEA Technology, Harwell, Oxfordshire, England, 1995.

[21] W. Hock and K. Schittkowski, *Test examples for nonlinear programming codes*, Lecture Notes in Economics and Mathematical Systems 187, Springer Verlag, Berlin, 1981.

[22] M. Lalee, J. Nocedal, and T. Plantenga, *On the implementation of an algorithm for large-scale equality constrained optimization*, submitted to SIAM J. Optimization, December, 1993.

[23] W. Murray and F. J. Prieto, *A sequential quadratic programming algorithm using an incomplete solution of the subproblem*, Tech Report, Department of Operations Research, Stanford University, 1992.

[24] J. Nocedal and M. L. Overton, *Projected Hessian updating algorithms for nonlinearly constrained optimization*, SIAM J. Numer. Anal., 22 (1985), pp. 821–850.

[25] E.O. Omojokun,*Trust region algorithms for optimization with nonlinear equality and inequality constraints*, PhD dissertation, University of Colorado, 1991.

[26] C.E. Orozco, *Large-scale shape optimization: numerical methods, parallel algorithms and applications to aerodynamic design*, Ph.D. Dissertation, Carnegie Mellon University, 1993.

[27] E.R. Panier and A.L. Tits, *On combining feasibility, descent and superlinear convergence in inequality constrained optimization*, Mathematical Programming, 59 (1993), 261-276.

[28] Ph.L. Toint, *An assessment of non-monotone line search techniques for unconstrained optimization.* SIAM Journal on Scientific and Statistical Computing, 17 (3), (1996) pp. 725-739.

[29] S. Vasantharajan and , L. T. Biegler, *Large-scale decomposition for successive quadratic programming*, Comp. Chem. Engr., 12, 11 (1988) p. 1087

[30] Y. Xie, *Reduced Hessian algorithms for solving large-scale equality constrained optimization problems*, Ph.D. dissertation, Department of Computer Science, University of Colorado, Boulder, 1991.

[31] Y. Yuan, *An only 2-step Q-superlinear convergence example for some algorithms that use reduced Hessian approximations*, Math. Programming, 32 (1985), pp. 224–231.