Subspace Techniques for Nonlinear Optimization*

Ya-xiang Yuan
LSEC, ICMSEC, Academy of Mathematics and System Sciences,
Chinese Academy of Sciences, Beijing 100080, China.
E-mail: yyx@lsec.cc.ac.cn

Abstract

In this paper, we review various subspace techniques that are used in constructing of numerical methods for nonlinear optimization. The subspace techniques are getting more and more important as the optimization problems we have to solve are getting larger and larger in scale. The applications of subspace techniques have the advantage of reducing both computation cost and memory size. Actually in many standard optimization methods (such as conjugate gradient method, limited memory quasi-Newton method, projected gradient method, and null space method) there are ideas or techniques that can be viewed as subspace techniques. The essential part of a subspace method is how to choose the subspace in which the trial step or the trust region should belong. Model subspace algorithms for unconstrained optimization and constrained optimization are given respectively, and different proposals are made on how to choose the subspaces. As an example, we also present an interior point method based on subspace techniques.

1 Introduction

Nonlinear optimization is to minimize an objective function subject to some equality and inequality constraints, which can be written as follows.

$$\min_{x\in\Re^n} \quad f(x) \tag{1.1}$$

$$\text{subject to} \quad c_i(x) = 0, \qquad i = 1, ..., m_e \tag{1.2}$$

$$c_i(x) \geq 0, \qquad i = m_e + 1, ..., m, \tag{1.3}$$

where m and m_e are two non-negative integers satisfying $m \geq m_e$. Normally, $f(x)$ and $c_i(x) (i = 1, ..., m)$ are continuously differentiable functions. We use the notations $g(x) = \nabla f(x)$ and $a_i(x) = \nabla c_i(x)$.

*The authors are partly supported by NSF of China (No. 10231060).

Large scale optimization problems (n is very large) have attracted much attention from researchers in the recent years. A very good recent review paper was given by Gould, Orban and Toint[5], where the authors discuss the main approaches for handling large scale problems. The step computation, active set, gradient projection, and interior point method are discussed there.

One of the straightforward technique for large scale problems is to solve the large scale subproblems efficiently by approximate methods. The overall cost for building a house will be reduced if each brick is cheaper. For example, the truncated conjugate gradient (CG) method is a very successful algorithm for the single ball trust region subproblem:

$$\min_{d\in\Re^n} \quad Q(d) = g^T d + \frac{1}{2} d^T B d \tag{1.4}$$

$$s. \quad t. \quad \|d\|_2 \le \Delta. \tag{1.5}$$

The truncated CG method proposed independently by Steihaug[8] and Toint[11] defines an approximate solution by applying the standard conjugate gradient method until reaching the trust region bound or obtaining the exact solution. It was proved by Yuan[17] that :

$$Q(0) - Q(d_{TCG}) \ge \frac{1}{2}[Q(0) - Q(d^*)] \tag{1.6}$$

where d_{TCG} is the solution obtained by the truncated conjugate gradient method and d^* is the solution of (1.4)-(1.5).

Active set is also an efficient approach to large scale problems, particularly for problems having many redundant constraints. In a sequential quadratic programming (SQP) algorithm, the subproblem at the k-th iteration is

$$\min_{d\in\Re^n} \quad Q_k(d) = \nabla f(x_k)^T d + \frac{1}{2} d^T B_k d \tag{1.7}$$

$$s.t. \quad c_i(x_k) + a_i^T(x_k)^T d = 0, \qquad i = 1, ..., m_e, \tag{1.8}$$

$$c_i(x_k) + a_i^T(x_k)^T d \ge 0, \qquad i = m_e + 1, ..., m. \tag{1.9}$$

By active set technique, instead of solving the above subproblem, we can solve the following subproblem:

$$\min_{d\in\Re^n} \quad Q_k(d) = \nabla f(x_k)^T d + \frac{1}{2} d^T B_k d \tag{1.10}$$

$$s.t. \quad c_i(x_k) + a_i(x_k)^T d = 0, \qquad i \in \mathcal{A}_k, \tag{1.11}$$

where $\mathcal{A}_k$ is an active set which is a subset of $\{1, 2, ..., m\}$ and is an approximation to the active set at the solution. When the number of elements of $\mathcal{A}$ is significantly smaller that m, the active set subproblem should be much easier to solve than the original subproblem.

One of our motivations for suggesting subspace algorithms for nonlinear optimization is the unbalance property shared by most line search algorithms. Consider any line search method having the form:

$$x_{k+1} = x_k + \alpha_k d_k, \tag{1.12}$$

where d_k is the search direction and $\alpha_k > 0$ is the step-length computed by certain line search technique. The search direction d_k is usually computed by solving a subproblem which is an approximation to the original nonlinear problem. Thus, each iteration of a line search algorithm composites of two parts, one is to find a d_k in the whole n dimensional space, while the other part is to search a suitable step-length in the fixed one dimensional space spanned by the computed d_k. Thus, the overall algorithm swings between n dimensional search and one dimensional search alternately.

Another motivation for us to consider subspace algorithms is that quite a few well known existing algorithms essentially have certain subspace features. For example, the conjugate gradient method uses a search direction in a two dimensional subspace spanned by the steepest descent direction and the previous step, the dog-leg method computes a step that is a convex combination of the steepest descent direction and the Newton's direction, and limited memory quasi-Newton algorithms produce search directions spanned in lower dimensional spaces.

This paper is organized as follows. In the next section, we will give some examples of algorithms that have certain subspace structures. In section 3, a model algorithm using subspace approach for unconstrained optimization is given and some possibilities for choices of the subspaces are also discussed. In section 4, subspace techniques for constrained optimization are presented, together with a subspace interior point trust region algorithm for box constrained optimization as an example. Finally, a brief discussion is given.

2 Examples of Subspace Approaches

It is well-known that nonlinear conjugate gradient methods use a linear combination of the steepest descent direction $-g_k$ and the previous search direction d_{k-1} to define the new search direction:

$$d_k = -g_k + \beta_k d_{k-1}. \tag{2.1}$$

Thus, one of the central tasks of nonlinear conjugate gradient methods has been how to define the suitable β_k based on certain conjugate principles. The four leading contenders are the FR, HS, PRP and DY methods[7].

Instead of the conjugate property, Stoer and Yuan[9] suggests to look at the conjugate gradient method from the subspace point of view. As a conjugate gradient method uses a β_k to define d_k and then a step-size α_k to set $x_{k+1} = x_k + \alpha_k d_k$, no matter whatever β_k and α_k are used, the increment in the iterative point will be a linear combination of $-g_k$ and d_{k-1}. Hence it is natural to ask how to find a best point in the 2-dimensional subspace spanned by $-g_k$ and d_{k-1}. Namely, we can consider a model subproblem

$$\min_{d \in Span\{-g_k, s_{k-1}\}} Q_k(d) \approx f(x_k + d). \tag{2.2}$$

Let d_k be the solution of the above 2-dimensional subproblem. Stoer and Yuan[9] presents a successive 2-dimensional search algorithm, which is an example of algorithms using subspace models.

Another famous optimization technique has the subspace nature is the limited memory quasi-Newton method. Quasi-Newton updates (for example, see [3] and [10]) have the form

$$B_k = U(B_{k-1}, s_{k-1}, y_{k-1}) \tag{2.3}$$

which satisfies

$$B_k s_{k-1} = y_{k-1}, \tag{2.4}$$

where $s_{k-1} = x_k - x_{k-1}$ and $y_{k-1} = \nabla f(x_{k-1} + s_{k-1}) - \nabla f(x_{k-1})$. An example is the famous BFGS method

$$B_k = B_{k-1} - \frac{B_{k-1} s_{k-1} s_{k-1}^T B_{k-1}}{s_{k-1}^T B_{k-1} s_{k-1}} + \frac{y_{k-1} y_{k-1}^T}{s_{k-1}^T y_{k-1}}. \tag{2.5}$$

Limited memory quasi-Newton updates the approximate Hessian repeatedly:

$$B_k^{(i)} = U(B_k^{(i-1)}, s_{k-m-1+i}, y_{k-m-1+i}), \quad i = 1, ..., m, \tag{2.6}$$

with $B_k^{(0)} = \sigma_k I$. For example, see [6]. There are different formulae for σ_k with one particular choice being $s_{k-1}^T y_{k-1} / y_{k-1}^T y_{k-1}$. It can be shown that the limited memory quasi-Newton matrix can be written as

$$B_k = B_k^{(m)} = \sigma_k I + [S_k \quad Y_k] T_k \begin{bmatrix} S_k^T \\ Y_k^T \end{bmatrix} \tag{2.7}$$

where T_k is a $2m \times 2m$ matrix and

$$[S_k \quad Y_k] = [s_{k-1}, s_{k-2}, ... s_{k-m}, y_{k-1}, y_{k-2}, ..., y_{k-m}] \in \Re^{n \times 2m} \tag{2.8}$$

In a line search type method, we have that $s_k = \alpha_k d_k = -\alpha_k B_k^{-1} g_k$, while for a trust region type algorithm one would have $s_k = -(B_k + \lambda_k I)^{-1} g_k$. Therefore, in either case, it follows that

$$\begin{aligned} s_k &= -\left(\rho_k I + [S_k \quad Y_k] T_k \begin{bmatrix} S_k^T \\ Y_k^T \end{bmatrix}\right)^{-1} g_k \\ &\in Span\{g_k, s_{k-1}, ..., s_{k-m}, y_{k-1}, ..., y_{k-m}\}. \end{aligned} \tag{2.9}$$

Consequently, we have shown that limited memory quasi-Newton algorithms no matter with line search or trust region will always produce a step in the subspace $Span\{g_k, s_{k-1}, ..., s_{k-m}, y_{k-1}, ..., y_{k-m}\}$. Indeed, a trust region algorithm using this subspace is given by Wang, Wen and Yuan[13].

Even for the standard quasi-Newton updates, we can also establish subspace property results. The following is a result for trust region type algorithms.

Lemma 2.1. *(Wang and Yuan, [14]) Suppose $B_1 = \sigma I$, $\sigma > 0$. The matrix updating formula is any one chosen from amongst SR1, PSB and Broyden family, and B_k is the k-th updated matrix. s_k is the solution of*

$$\min_{d \in \Re^n} \; g_k^T d + \frac{1}{2} d^T B_k d \tag{2.10}$$

$$s.t. \; \|d\|_2 \le \Delta_k, \tag{2.11}$$

$x_{k+1} = x_k + s_k$, $g_k = \nabla f(x_k)$. Let $\mathcal{G}_k = Span\{g_1, g_2, \cdots, g_k\}$. Then

$$s_k \in \mathcal{G}_k, \tag{2.12}$$

and we have

$$B_k z \in \mathcal{G}_k, \quad B_k u = \sigma u \tag{2.13}$$

for any $z \in \mathcal{G}_k$, $w \in \mathcal{G}_k^{\perp}$.

The above lemma is an extension of similar results for line search type algorithms, which were discussed by Gill and Leonard[4] and Vlcek and Luksan[12].

A modification to the standard trust region subproblem (1.4)-(1.5) is given by Burdakov and Yuan[1]. Assume that the matrix B is generated by limited memory quasi-Newton updates. Namely we have

$$B = \sigma I + PDP^T, \qquad P \in \Re^{n \times l}. \tag{2.14}$$

Instead of using the Euclidean norm as in (1.5), Burdakov and Yuan[1] suggested the following norm

$$\|d\|_{P,\infty} = \max\{\|P^T d\|_\infty, \; \|P_\perp^T d\|_2\}, \tag{2.15}$$

where $P_\perp^T$ is a matrix such that $P_\perp^T P = 0$ and $(P, P_\perp)$ is nonsingular. Such a choice of the norm makes the subproblem

$$\min_{\|d\|_{P,\infty} \le \Delta} \quad Q(d) \tag{2.16}$$

easy to solve.

3 A model subspace algorithms

Based on the above observations, we can suggest a model subspace algorithm for unconstrained optimization, which is a slightly modification of the standard trust region algorithm for unconstrained optimization (for example, see Yuan[16]).

Algorithm 3.1. *(A model subspace algorithm for unconstrained optimization)*

```
Step 1  Given $x_1$, Define $\mathcal{S}_1$, $\epsilon > 0$, $k := 1$.
Step 2  Solve a subspace subproblem:
            $$\min_{d \in \mathcal{S}_k} Q_k(d) = g_k^T d + \frac{1}{2} d^T B_k d \qquad (3.1)$$
        obtaining $s_k$. If $\|s_k\| \le \epsilon$ then stop.
Step 3  Define
            $$x_{k+1} = \begin{cases} x_k + s_k & \text{if } f(x_k + s_k) < f(x_k) \\ x_k & \text{otherwise} \end{cases} \qquad (3.2)$$
Step 4  Generate $\mathcal{S}_{k+1}$ and $Q_{k+1}(d)$.
Step 1  $k := k + 1$, Go to Step 2.
```

The main difference between the above algorithm and standard whole space algorithms is the constraint for the step s_k to be in the subspace $\mathcal{S}_k$. Thus, the key issue here is how to choose the subspace $\mathcal{S}_k$.

One obvious choice for the subspace $\mathcal{S}_k$ is a generalization of the 2-dimensional subspace studied by Stoer and Yuan[9], namely $\mathcal{S}_k = Span\{-g_k, s_{k-1}, ..., s_{k-m}\}$. As all the points in $\mathcal{S}_k$ can be expressed by

$$d = \alpha g_k + \sum_{i=1}^{m} \beta_i s_{k-i},$$

using the following approximations

$$s_{k-i}^T \nabla^2 f(x_k) s_{k-j} \approx s_{k-i}^T y_{k-j}, \quad s_{k-i}^T \nabla^2 f(x_k) g_k \approx y_{k-i}^T g_k \tag{3.3}$$

we can write the quadratic function $Q_k(d) \approx f(x_k + d)$ in the subspace by a quadratic function of $\alpha, \beta_1, ..., \beta_m$:

$$\bar{Q}_k(\alpha, \beta_1, ..., \beta_m) = (-\|g_k\|^2, g_k^T s_{k-1}, ..., g_k^T s_{k-m})^T \begin{pmatrix} \alpha \\ \beta_1 \\ \vdots \\ \beta_m \end{pmatrix}$$

$$+\frac{1}{2} \begin{pmatrix} \alpha \\ \beta_1 \\ \vdots \\ \beta_m \end{pmatrix}^T \begin{pmatrix} \rho_k & -g_k^T y_{k-1} & \cdots & -g_k^T y_{k-m} \\ -g_k^T y_{k-1} & y_{k-1}^T s_{k-1} & \cdots & y_{k-m}^T s_{k-1} \\ \vdots & \vdots & \ddots & \vdots \\ -g_k^T y_{y-m} & y_{k-m}^T s_{k-1} & \cdots & y_{k-m}^T s_{k-m} \end{pmatrix} \begin{pmatrix} \alpha \\ \beta_1 \\ \vdots \\ \beta_m \end{pmatrix}, \quad (3.5)$$

where ρ_k should be an approximation to $g_k^T \nabla^2 f(x_k) g_k$. There are many possible ways to approximate it. First, we can write this value in a different form:

$$g_k^T \nabla^2 f(x_k) g_k = \frac{1}{\cos^2 \theta_k} \frac{(s_{k-1}^T \nabla^2 f(x_k) g_k)^2}{s_{k-1}^T \nabla^2 f(x_k) s_{k-1}} \approx \frac{1}{\cos^2 \theta_k} \frac{(y_{k-1}^T g_k)^2}{s_{k-1}^T y_{k-1}}, \quad (3.6)$$

where θ_k is the angle between $(\nabla^2 f(x_k))^{1/2} s_{k-1}$ and $(\nabla^2 f(x_k))^{1/2} g_k$. Similar to [9], due to the mean value to $cos^2\theta = 1/2$, we can let

$$\rho_k = 2 \frac{(y_{k-1}^T g_k)^2}{s_{k-1}^T y_{k-1}}. \quad (3.7)$$

Because $g_k^T \nabla^2 f(x_k) g_k$ is the magnitude of the Hessian matrix in the direction g_k, which can be estimated by the average over the m directions $s_{k-i} (i = 1, ..., m)$, it is reasonable to let

$$\rho_k = \frac{\|g_k\|_2^2}{m} \sum_{i=1}^{m} \frac{s_{k-i}^T y_{k-i}}{s_{k-i}^T s_{k-i}}. \quad (3.8)$$

Of course, we can use a limited memory matrix B_k and set $\rho_k = g_k^T B_k g_k$. If an extra function value at $x_k + t g_k$ is calculated, one can also let

$$\rho_k = \frac{2(f(x_k + t g_k) - f(x_k) - t\|g_k\|_2^2)}{t^2}. \quad (3.9)$$

Another possible choice for $\mathcal{S}_k$ is $Span\{-g_k, y_{k-1}, ..., y_{k-m}\}$. If we express the solution of (3.1) as

$$d = \alpha g_k + \sum_{i=1}^{m} \beta_i y_{k-i} = W_k \begin{pmatrix} \alpha \\ \beta_1 \\ \cdots \\ \beta_m \end{pmatrix}, \quad (3.10)$$

the coefficients $\alpha, \beta_1, ..., \beta_m$ should be

$$-\left[W_k^T \nabla^2 f(x_k) W_k\right]^{-1} W_k^T \nabla f(x_k).$$

One approach to estimate the above vector is to replace $\left[W_k^T \nabla^2 f(x_k) W_k\right]^{-1}$ by $\left[W_k^T (\nabla^2 f(x_k))^{-1} W_k\right]$. In this way, all we need to do is to estimate $g_k \nabla^2 f(x_k)^{-1} g_k$. Estimations for such a value can be obtained by similar techniques that were just discussed above for estimating $g_k^T \nabla^2 f(x_k) g_k$.

Let P_k be a matrix whose columns spanned $\mathcal{S}_k$. The solution of (3.1) d_k can be written as $P_k z_k$, where z_k solves

$$P_k^T g_k + P_k^T B_k P_k z = 0. \tag{3.11}$$

A generalization of the subspace subproblem (3.1) is to find a step d such that

$$\bar{P}_k^T (g_k + B_k d) = 0, \quad d \in \mathcal{S}_k, \tag{3.12}$$

where $\bar{P}_k^T$ is a projection from $\Re^n$ to a lower dimensional subspace $\bar{\mathcal{S}}_k$. Normally (3.12) exists a solution if the dimension of $\bar{\mathcal{S}}_k$ is not larger than that of $\mathcal{S}_k$. (3.1) is a special case of (3.12) when $\bar{\mathcal{S}}_k = \mathcal{S}_k$. It is easy to see that the Gauss-Seidel method for linear equations is just the case when $\mathcal{S}_k = \bar{\mathcal{S}}_k$ are the one dimensional subspaces spanned by the co-ordinate directions.

4 Subspace techniques for Constrained Optimization

Now we discuss subspace techniques for constrained optimization. For simplicity, we only consider equality constrained problems here, as the techniques discussed below can be easily extended to general inequality constraints. The equality constrained optimization has the form:

$$\min_{x \in \Re^n} \; f(x) \tag{4.1}$$

$$s.t. \;\; c(x) = 0. \tag{4.2}$$

An SQP subproblem for the above problem is

$$\min_{d \in \Re^n} \; Q_k(d) \tag{4.3}$$

$$s.t. \;\; c(x_k) + A_k^T d = 0 \tag{4.4}$$

The null space approach obtains a solution of the above subproblem by computing a Range Space step (vertical step) v_k and a Null Space step (horizontal step) h_k and setting $d_k = h_k + v_k$. For example, let the Q-R factorization of A_k be

$$A_k = [Y_k \quad Z_k] \begin{bmatrix} R_k \\ 0 \end{bmatrix}, \tag{4.5}$$

we can let $v_k = -Y_k R_k^{-T} c_k$ and $h_k = Z_k z$ with z being the solution of

$$\min Q_k(v_k + Z_k z). \tag{4.6}$$

One of the nice properties of the SQP subproblem is that its solution d_k is a superlinearly convergent step, namely

$$x_k + d_k - x^* = o(\|x_k - x^*\|) \tag{4.7}$$

if B_k is a good approximate to the Hessian of the Lagrangian. Unfortunately, Marotos effect may happen. That is, even when relation (4.7) holds, it is possible that

$$f(x_k + d_k) > f(x_k), \qquad \|c(x_k + d_k)\| > \|c(x_k)\|. \tag{4.8}$$

The increases in both the objective function and constraint violation will make all the standard globalization techniques such as line search, trust region, and filter reject the new point $x_k + d_k$!

A remedy for the Marotos effect is the second order correction step technique which solves another subproblem

$$\min_{d \in \Re^n} \quad Q_k(d_k + d) \tag{4.9}$$

$$s.t. \quad c(x_k + d_k) + A_k^T d = 0 \tag{4.10}$$

to obtain an additional step $\hat{d}_k$. Under certain conditions, it can be shown that $\hat{d}_k = O(\|d_k\|^2)$, therefore we call $\hat{d}_k$ a second order correction step. Such a second order step would make the new point $x_k + d_k + \hat{d}_k$ acceptable.

Though the second order correct step is a very good technique having both strong theoretical properties and nice computational performance, it has an undesirable feature explained as follows. From the subspace point of view, the standard QP step d_k is the sum of two steps, a range space step v_k and a null space step h_k. When Marotos effect happens, it can be proved that the second order correction step $\hat{d}_k$ is almost a range space step as well. The range space steps v_k and $\hat{d}_k$ are quadratical steps, because basically the Newton-Raphson method is used to compute the range space step. The null space step h_k is normally a quasi-Newton step as B_k is usually updated by quasi-Newton formulae. Though the quasi-Newton step is superlinearly convergent, its Q-order of convergence can ba arbitrary close to 1 (see [15]). Thus, the convergence rate of the null space step can be much slower than the range space step. This is not surprising as it is quite normal to observe iterate points sinking into the feasible region. Consequently, the second order correction step technique leads to two range space steps and one null space step. There exists a strong unbalance, because in the range space we apply a faster algorithm (Newton's method) and make two iterations while in the null space we apply a slower algorithm for only one iteration. Just suppose there were a chariot with a larger wheel on one side and a smaller wheel on the other side. Obviously, such a chariot would be far from perfect. What the second order correction technique would make it even worse is to install a gearbox to this chariot so that the smaller wheel turns only one round while the larger wheel turns two rounds. Such a chariot has been driven on the road of optimization for two decades without being noticed.

The above paragraph shows that subspace analysis will provide us additional insight into an algorithm. At least, for null space type algorithms, we need to give another think on the null space step and the range space step.

Similar to Algorithm 3.1, we can give a subspace algorithm for constrained problem as follows.

Algorithm 4.1. *(A model subspace method for equality constrained optimization)*

Step 1 *Given* x_1, *Define* $\mathcal{S}_1$, σ_1, $\epsilon > 0$, $k = 1$.

Step 2 *Solve a subspace subproblem:*

$$\min \ Q_k(d) \tag{4.11}$$

$$s.t. \ c_k + A_k^T d = 0, \qquad d \in \mathcal{S}_k. \tag{4.12}$$

obtaining s_k. *If* $\|s_k\| \le \epsilon$ *then stop.*

Step 3 *Define*

$$x_{k+1} = \begin{cases} x_k + s_k & \text{if } P(x_k + s_k, \sigma_k) < P(x_k, \sigma_k) \\ x_k & \text{otherwise} \end{cases} \tag{4.13}$$

where $P(x, \sigma)$ *is a penalty function.*

Step 4 *Generate* σ_{k+1}, $\mathcal{S}_{k+1}$ *and* $Q_{k+1}(d)$.

Step 1 $k := k + 1$, *Go to Step 2.*

Similar to the unconstrained case, there are many choices for $\mathcal{S}_k$. For example, a possible choice for $\mathcal{S}_k$ is $Span\{-g_k, s_{k-1}, ..., s_{k-m}, -\nabla c_{k_i}(x_k)\}$, where $|c_{k_i}(x_k)| = \|c(x_k)\|_\infty$.

Another subspace approach is try to find a subspace solution to the linear system:

$$\begin{bmatrix} B_k & A_k \\ A_k^T & 0 \end{bmatrix} \begin{bmatrix} d \\ \lambda \end{bmatrix} = - \begin{bmatrix} g_k \\ c_k \end{bmatrix} \tag{4.14}$$

Namely we use a subspace $\mathcal{S}_k$ to compute d_k by either solving

$$\min_{d \in \mathcal{S}_k} \left\| \begin{bmatrix} B_k & A_k \\ A_k^T & 0 \end{bmatrix} \begin{bmatrix} d \\ \lambda \end{bmatrix} + \begin{bmatrix} g_k \\ c_k \end{bmatrix} \right\|_2^2 . \tag{4.15}$$

or solving

$$\min_{d \in \mathcal{S}_k} \left\| P_k \left(\begin{bmatrix} B_k & A_k \\ A_k^T & 0 \end{bmatrix} \begin{bmatrix} d \\ \lambda \end{bmatrix} + \begin{bmatrix} g_k \\ c_k \end{bmatrix} \right) \right\|_2^2 , \tag{4.16}$$

where P_k is a projection from $\Re^{n+m}$ to some lower dimensional subspace.

For box constrained optimization, Yuan[18] presents a trust region interior point method based on subspace technique, which is an generalization of the algorithm given by Coleman and Li[2].

Consider the box constrained problem

$$\min \ f(x) \tag{4.17}$$
$$s.t. \ \ l \le x \le u. \tag{4.18}$$

Denote the current iterate point at the k-th iteration by x_k, and let

$$\Omega_k = Diag((\omega_k)_1, ..., (\omega_k)_n), \tag{4.19}$$

where

$$(\omega_k)_i = \min\{(x_k)_i - l_i, u_i - (x_k)_i\} \tag{4.20}$$

and $\Delta_k = \min_{1 \le i \le m} (\omega_k)_i = Dist(x_k, \Gamma)$.

The subspace subproblem given by Yuan[18] is defined by

$$\min \ \phi_k(d) = g_k^T d + \frac{1}{2} d^T B_k d \tag{4.21}$$
$$s.t. \ \ d^T \Omega_k^{-1} d \le \rho_k \Delta_k, \qquad d \in \mathcal{S}_k. \tag{4.22}$$

where $\rho_k \in (0,1)$ and the subspace $\mathcal{S}_k$ updated from iteration to iteration.

Algorithm 4.2. *(An Interior Point Method with Subspace Techniques)*

```
Step 1  Given an interior point $x_1$, $\epsilon > 0$, given a subspace $\mathcal{S}_1$, $k := 1$.
Step 2  Solve (4.21)-(4.22) obtaining $s_k$.
        if $\|s_k\| \le \epsilon$ then stop.
```

Step 3 Let

$$x_{k+1} = \begin{cases} x_k + s_k & \text{if } r_k > 0; \\ x_k & \text{otherwise} \end{cases} \tag{4.23}$$

$$\rho_{k+1} = \begin{cases} \frac{\rho_k + 1}{2} & \text{if } r_k \geq 0.1; \\ \frac{\rho_k}{2} & \text{otherwise} \end{cases} \tag{4.24}$$

Step 4 Define the subspace $\mathcal{S}_{k+1}$ for the next interation.
$k := k + 1$, go to Step 2.

Yuan[18] shows that the above algorithm converges to a stationary point of (4.17)-(4.18) if $\Omega_k g_k \in \mathcal{S}_k$ for all k.

5 Discussions

Various subspace techniques for constructing numerical methods for nonlinear optimization have been discussed in the paper. Subspace techniques are suitable for large scale problems, particularly when function and gradient values are difficult to compute and when functions are highly nonlinear. For constrained optimization, Subspace depending on constraints are not easy to be defined. We believe that subspace methods will attract more and more attention in the future.

References

[1] O. Burdakov and Y. Yuan, Limited memory trust region methods for solving large scale unconstrained optimization, Preprint, 2002.

[2] T. Coleman and Y.Y. Li, An interior point trust region approach for nonlinear minimization subject to bounds, SIAM J. Optimization, 6(1996) 418-445.

[3] R. Fletcher, Practical Methods of Optimization, Second Edition, (Jhon Wiley and Sons, 1987).

[4] P.E. Gill and M.W. Leonard, Reduced-Hessian quasi-Newton methods for unconstrained optimization, SIAM J. Optim. , 12(2001), pp.209-237.

[5] N.I.M. Gould, D. Orban and Ph.L. Toint, Numerical methods for large-scale nonlinear optimization, Acta Numerica (2005), pp. 299-361.

[6] D.C. Liu and J. Nocedal, On the limited memory BFGS method for large scale optimization, Mathematical Programming 45(1989), pp. 503-528.

[7] J.L. Nazareth, Conjugate-gradient methods, in C.A. Floudas and P.M. Pardalos, eds., Encyclopedia of Optimizaiton, Vol 1. (Kluwer, 2001) pp.319-323.

[8] T. Steihaug, The conjugate gradient method and trust regions in large scale optimization, SIAM J. Numerical Analysis, 20(1983), pp. 626–637.

[9] J. Stoer and Y. Yuan, A subspace study on conjugate gradient algorithms, ZAMM Z. angew. Math. Mech., 75(1995),pp. 69–77.

[10] W.Y. Suan and Y. Yuan, Optimization Theory and Mehtods: Nonlinear Programming, Springer Optimization and Its Application, Vol. 1, (Springer, 2006).

[11] Ph.L. Toint, Towards an efficient sparsity exploiting Newton method for minimization, in: I. Duff, ed. Sparse Matrices and Their Uses, (Academic Press, 1981), pp. 57-88.

[12] J. Vlček and L. Lukšan, New variable metric methods for unconstrained minimization covering the large-scale case, Technical report No. V876, October 2002, Institue of Computer Science, Academy of Sciences of the Czech Republic.

[13] Z.H. Wang, Z.W. Wen and Y. Yuan, A subspace trust region method for large scale unconstrained optimization, in; Y. Yuan, ed. Numerical Linear Algebra and Optimization (Science Press, 2004) pp. 265-274.

[14] Z.H. Wang and Y. Yuan, A subspace implementation of a quasi-Newton trust region method for unconstrained optimization, Report, ICMSEC, AMSS, Chinese Academy of Sciences, 2004.

[15] Y. Yuan, On the least Q-order of convergence of variable metric algorithms, IMA J. Numerical Analysis 4(1984) 233-239.

[16] Y. Yuan, A review of trust region algorithms for optimization, in ICM99: Proceedings of the Fourth International Congress on Industrial and Applied Mathematics, J. M. Ball and J. C. R. Hunt, eds., Oxford University Press,Oxford, 2000, pp. 271–282.

[17] Y. Yuan, On the truncated conjugate gradient method, Math. Prog., 87(2000), pp. 561–571.

[18] Y. Yuan, Subspace techniques for nonlinear optimization, talk presented at the International Conference on High Performance Scientific Computing, March 6-10, 20006.