Linearly Constrained Optimization and Projected Preconditioned Conjugate Gradients*

Thomas F. Coleman†

Abstract

We consider how to apply the preconditioned conjugate gradient process to generate feasible iterates for large-scale nonlinear minimization in the presence of linear constraints, including non-negativity conditions. A new and useful viewpoint is proffered; important computational linear algebra issues are discussed.

1 Introduction

We are concerned with using preconditioned conjugate gradient (PCG) technology to solve continuous large-scale minimization problems with linear constraints. Our objective in this note is to consider the use of *projected* PCG techniques, whilst generating *feasible* iterates, and highlight some important linear algebra issues; we do not present complete optimization algorithms in this missive.

An alternative approach, based on an indefinite iterative scheme applied to the optimality conditions, is given in [10]. Our approach is different: we are concerned with the direct application of the PCG process to the linearly constrained setting.

We consider two situations. First, we examine the nonlinear minimization problem with linear equality constraints,

$$\min\{f(x) : Ax = b\}, \tag{1}$$

where A is an m-by-n matrix of rank m, and $f : \Re^n \rightarrow \Re$ is twice continuously-differentiable. Our second problem adds non-negativity constraints to (1):

$$\min\{f(x) : Ax = b,\ x \geq 0\}. \tag{2}$$

We illustrate how the ideas we develop for problem (1) can also be applied to (2) with an appropriate class of preconditioners (within the context of an interior Newton method).

We begin with a cursory glance at the unconstrained problem. Superlinear methods for solving large-scale unconstrained minimization problems can be obtained by applying PCG to the linear (Newton) system

$$H_k p = -g_k, \tag{3}$$

*This research was partially supported by the Applied Mathematical Sciences Research Program (KC-04-02) of the Office of Energy Research of the U.S. Department of Energy under grant DE-FG02-86ER25013.A000, and in part by NSF, AFOSR, and ONR through grant DMS-8920550, and by the Advanced Computing Research Institute, a unit of the Cornell Theory Center which receives major funding from the National Science Foundation and IBM Corporation, with additional support from New York State and members of its Corporate Research Institute.

†Computer Science Department and Center for Applied Mathematics, Cornell University, Ithaca, New York, 14853.

in each iteration k, where $H_k = \nabla^2 f(x_k)$, and $g_k = \nabla f(x_k)$. PCG iterations can be applied to (3), always "improving" p until the residual, $H_k p + g_k$, is sufficiently small or until a direction, d, of non-positive curvature is discovered, i.e., $d^T H_k d \leq 0$. In the latter case d can be used to help further decrease the objective function f, e.g., Steihaug [12].

In algorithm PCG below, initialize scalar $\beta = 0$; initialize vectors $r = -g_k$, $p = 0$, $d_- = 0$, and $z = \mathcal{P}_k(r)$. *The function $\mathcal{P}_k$ is the preconditioning operation; typically, this involves a (sparse) symmetric positive definite matrix C_k approximating H_k, and the solution of the system: $C_k z = r$.*

Algorithm PCG

While $\|z\|$ is not "small enough"

1. $d = z + \beta d_-$, $\gamma = d^T H_k d$
2. if $\gamma \leq 0$, return(p, d)
   else $\alpha = \frac{r^T z}{\gamma}$, $p = p + \alpha d$, $r = r - \alpha H_k d$, end
3. $z = \mathcal{P}_k(r)$
4. $\beta = \frac{r^T z}{r_-^T z_-}$

return(p)

The notation v_-, where v is a vector, refers to the value of v in the previous PCG iteration.

2 Linear Constraints

How can Algorithm PCG be adapted to problem (1)? In principle there is a straightforward answer. Assume x_k is feasible. Let the columns of a matrix $Z \in \Re^{n \times (n-m)}$ form a basis for the null space of A and, using PCG, (approximately) solve the reduced system,

$$(Z^T H_k Z)\bar{p} = -Z^T g_k \tag{4}$$

and then assign $p = Z\bar{p}$. Algorithm PCG can be directly applied to (4) by identifying H_k and g_k in Algorithm PCG with $Z^T H_k Z$ and $Z^T g_k$ in (4) respectively. Despite the apparent simplicity of this approach, there are problems when applied in the large-scale setting.

First, is the reduced matrix $Z^T H_k Z$ formed explicitly? If so, it is imperative that both Z and $Z^T H_k Z$ be sparse when $n - m$ is large. The problem of finding a sparse null basis Z has been studied with some success, e.g., [4, 9]; however, the problem of determining a basis Z for the null space of A where both Z and $Z^T H_k Z$ are sparse is unchartered territory. Moreover, we expect it is unlikely that a good general purpose strategy is possible, though certainly problems with cooperative sparsity structures will be amenable to this approach.

Second, if the reduced matrix $Z^T H_k Z$ is not formed, two concerns remain. There is still the requirement that a sparse (or low-storage) null basis Z be determined, and how do we precondition system (4) when the matrix elements are not accessible? Nash and Sofer [11] make some preconditioning suggestions based on an approximation to (4). However, it is clear that preconditioning strategies that require sparsity and direct access to the matrix elements are not viable in this situation.

We can circumvent these problems with an elegant and general solution. Let C_k be a positive definite approximation to the current Hessian matrix H_k, where C_k is "suitably sparse". Then algorithm PCG can be used to solve (4) provided $z = \mathcal{P}_k(r)$ is defined by the augmented system:

$$\begin{pmatrix} C_k & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} z \\ w \end{pmatrix} = \begin{pmatrix} r \\ 0 \end{pmatrix} \tag{5}$$

We call this method the Projected Preconditioned Conjugate Gradient (PPCG) algorithm.

The structure of system (5) allows for three basic solution techniques. We discuss each in turn. Each approach involves a different factorization: it is important to realize, in the context of Algorithm PPCG, that the factorization is done just once for each (outer) index k. Subsequent PPCG iterations, within a fixed major iteration k, will involve only the solution of triangular systems.

A Full-Space Approach: This approach is the least restrictive with regard to the choice of C_k. The idea is to merely treat (5) as a sparse symmetric system and use a general sparse factorization/solver for such systems, e.g. [5]. Certainly C_k must be sparse to allow for a sparse factorization of the augmented system but there are no further restrictions on the sparsity of C_k. The main drawback is dimension: the augmented system is of order $m+n$ and so the factorization can be expensive.

A Range-Space Approach: Consider a block reduction of the matrix in (5) to introduce a zero "element" in the lower left block, i.e., position $(2,1)$. This leads to the use of the Schur-complement and ultimately,

$$z = [C_k^{-1} - C_k^{-1}A^T(AC_k^{-1}A^T)^{-1}AC_k^{-1}]r. \tag{6}$$

If $C_k = L_k L_k^T$ then (6) can be written[1], $z = L_k^{-T} P_{AL_k^{-T}} L_k^{-1} r$, where $P_{AL_k^{-T}}$ is the orthogonal projector onto the null space of AL_k^{-T}.

The computation of z, following (6), can be implemented as follows:

Algorithm Range-Project

1. Set $\tilde{A}_k = AL_k^{-T}$,
2. Solve $L_k w = r$, solve $L_k^T v = w$,
3. Solve $\tilde{A}_k^T u \stackrel{ls}{=} w$, solve $L_k^T y = \tilde{A}_k^T u$,
4. Set $z = v - y$.

The remaining freedom, given L_k, is how to solve the least-squares problem in step 3, $\tilde{A}_k^T u \stackrel{ls}{=} w$. Bear in mind that *Algorithm Range-Project* will be used several times each major iteration, i.e., with fixed matrices A, L_k. Therefore, a suitable factorization of $\tilde{A}_k$ can be performed once, each major iteration k, and the appropriate factors saved for subsequent applications of *Algorithm Range-Project* within the major iteration. One possibility is to compute a sparse QR-factorization of $\tilde{A}_k^T$, when $\tilde{A}_k$ is formed, and save the factor R_k. Numerous serial and parallel algorithms for computing a sparse QR-factorization exist, e.g., [8, 13]. However, it is usually not feasible to explicitly compute or save Q_k. Therefore, subsequent applications of *Algorithm Range-Project*, within a major iteration, may need to rely on normal equations to do the least squares solve, step 3, using the upper-triangular matrix R_k (saved from the sparse QR-factorization of $\tilde{A}_k^T$). That is, solve $R_k^T R_k u = \tilde{A}_k w$; the quality of the computed solution to this system can sometimes be improved with an iterative refinement technique.

The effectiveness of this approach is correlated to the sparsity/structure of $\tilde{A}_k$ which, in turn, is affected by the choice of preconditioner $C_k = L_k L_k^T$. If C_k is diagonal then $\tilde{A}_k$ has

[1] For simplicity we supress the role of permutation matrices, typically used to reduce fill in sparse matrix factorizations. In practise they are important and must be used. See George and Liu [7] or Duff, Erisman, and Reid [6] for an introduction to this important area.

the same sparsity as A; this choice yields the Carpenter/Shanno technique [1], used within an interior-point method for positive definite quadratic programming. Of course restricting the preconditioner to be diagonal is limiting – in some cases a diagonal preconditioner may not be effective. This leaves us with a challenge: How can we choose a preconditioner $C_k = L_k L_k^T$ so that C_k is both an effective preconditioner and yet the sparsity/structure of $\tilde{A}_k = AL_k^{-T}$ is acceptable?

A Null-Space Approach: If the columns of $Z \in \Re^{n\times(n-m)}$ form a basis for the null space of A the vector z can be computed by solving,

$$(Z^T C_k Z)\bar{z} = Z^T r \tag{7}$$

and then assigning $z = Z\bar{z}$. Equation (7) can be solved, for example, using either a sparse Cholesky factorization of $Z^T C_k Z$ or using a least-squares computation: $\tilde{Z}_k \bar{z} \overset{ls}{=} L_k^{-1} r$, where $\tilde{Z}_k = L_k^T Z$, i.e., $\tilde{Z}_k$ is a basis for the null space of AL_k^{-T}. Note that Z need be computed only once for the entire minimization process. The challenge of this approach is the determination of a sparse matrix Z such that the solution of (7) is economical. This is similar to our original difficulty with (4) except in this case we have additional flexibility - the choice of the structure of matrix C_k.

3 Non-negativity Constraints

The preconditioning ideas described above can also facilitate the solution of large-scale versions of (2).

If the current x-iterate is strictly feasible, it is possible to differentiate the complementary slackness condition for problem (2) to obtain an approximate Newton system. The complementary slackness condition for (2) can be written $g .* x = 0$, where $g = \nabla f + A^T\lambda$, λ is a vector of "multiplier estimates", and " $.*$" indicates component-wise multiplication. Assume x_k is a (strictly) feasible point in a neighbourhood of a non-degenerate optimal point. Taking into account the strict "sign-condition" that must hold at a non-degenerate optimal point, i.e., $x_i = 0 \Rightarrow g_i > 0$, then differentiation of the complementary slackness condition, along with $Ap = 0$, leads to system (4) with

$$H_k = \nabla^2 f(x_k) + diag(|g_k| \,./\, x_k). \tag{8}$$

In (8), "./" indicates component-wise division and $|g_k|$ is the vector satisfying $|g_k|_i = |(g_k)_i|$. After (approximately) solving (4), using the definitions of H_k and g_k above, assign $p = Z\bar{p}$. (To update x_k and remain feasible it is necessary to introduce a line search parameter α_k such that $x_{k+1} = x_k + \alpha_k p_k > 0$ and $\alpha_k \to 1$ sufficiently fast [2, 3].)

How can we effectively use the projected preconditioning ideas discussed above to compute p_k, when it is clear that H_k, as defined in (8), is unbounded as $x_k \to x_*$? A good answer is to define a preconditioner that isolates the source of the ill-conditioning, and then apply Algorithm PPCG. For example, let $C_k = D_k^{-1}\tilde{C}_k D_k^{-1}$ where $D_k^2 = diag(x_k)$ and $\tilde{C}_k = \tilde{L}_k \tilde{L}_k^T$ is a sparse positive definite approximation to

$$M_k = D_k \nabla^2 f(x_k) D_k + diag(|g_k|).$$

Note that matrix M_k is well-behaved: as $\{x_k\} \to x_*$, where x_* is a strong local minimizer of (2), $\{\|M_k\|\}$ is bounded; the reduction of M_k to the null space of A is positive definite for all k sufficiently large. Finally, we remark that under reasonable assumptions, Algorithm PPCG can be used, in conjunction with a line step procedure, described in [2, 3], to generate a strictly feasible sequence $\{x_k\}$ converging to x_* with a local superlinear rate.

4 Concluding Remarks

In summary, algorithm PPCG allows for the direct application of preconditioned conjugate gradient ideas to large-scale linearly constrained optimization, including problems with non-negativity constraints. Considering the three main approaches to (5), there does not appear to be an overall "best way". The Full-Space Approach is probably the most robust and is the most lenient with respect to the choice of C_k; however, it requires the factorization of a matrix of order $m + n$. The Range-Space Approach can work well when a diagonal preconditioner C_k is effective. Other choices of preconditioner for the Range-Space Approach are possible but the sparsity of AL_k^{-T} must be controlled. The Null-Space Approach requires that C_k and Z be chosen so that both Z and $Z^T C_k Z$ are sparse, and of course C_k must be effective as a preconditioner: Algorithms are needed.

5 Acknowledgements

Thanks to my ACRI colleagues for their assistance and advice.

References

[1] T. Carpenter and D. Shanno, *An interior point method for quadratic programs based on conjugate gradients*, Computational Optimization and Applications, 2 (1993), pp. 5–28.

[2] T. F. Coleman and Y. Li, *An interior trust region approach for nonlinear minimization subject to bounds*, Tech. Rep. TR 93-1342, Computer Science Department, Cornell University, 1993.

[3] T. F. Coleman and J. Liu, *An interior Newton method for quadratic programming*, Tech. Rep. 93-1388, Computer Science Department, Cornell University, 1993.

[4] T. F. Coleman and A. Pothen, *The null space problem II: Algorithms*, SIAM J. Alg. & Disc. Meth., 8 (1987), pp. 544–563.

[5] I. Duff, N. I. M. Gould, J. K. Reid, J. A. Scott, and K. Turner, *The factorization of sparse symmetric indefinite matrices*, IMA Journal of Numerical Analysis, 11 (1991), pp. 181–204.

[6] I. S. Duff, A. Erisman, and J. K. Reid, *Direct methods for sparse matrices*, Clarendon Press, Oxford UK, 1986.

[7] A. George and J. W.-H. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, 1981.

[8] J. George and M. Heath, *Solution of sparse least squares problems using Givens rotations*, SIAM J. Numer. Anal., 34 (1980), pp. 69–83.

[9] J. R. Gilbert and M. Heath, *Computing a sparse basis for the nullspace*, SIAM J. Alg. & Disc. Meth., 8 (1987), pp. 446–459.

[10] P. Gill, W. Murray, D. Ponceleon, and M. Saunders, *Preconditioners for indefinite systems arising in optimization*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 292 –311.

[11] S. G. Nash and A. Sofer, *Preconditioning of reduced matrices*, Tech. Rep. 93-01, Dept. of Operations Research and Engineering, George Mason University, 1993.

[12] T. Steihaug, *The conjugate gradient method and trust regions in large scale optimization*, SIAM J. Numer. Anal., 20 (1983), pp. 626–637.

[13] C. Sun, *Parallel sparse orthogonal factorizations on distributed-memory multiprocessors.* In preparation, 1993.