A Preconditioned Conjugate Gradient Approach to Linear Equality Constrained Minimization*

Thomas F. Coleman† Arun Verma†

July 1, 1998

Abstract

We propose a new framework for the application of preconditioned conjugate gradients in the solution of large-scale linear equality constrained minimization problems. This framework allows for the exploitation of structure and sparsity in the context of solving the reduced Newton system (despite the fact that the reduced system may be dense).

1 Introduction

We are concerned with large-scale quadratic minimization subject to linear equality constraints:

$$\min_{x\in\Re^n}\{c^Tx+\frac{1}{2}x^THx:\quad Ax=b\}, \tag{1}$$

where H is a symmetric matrix of order n, A is a m-by-n matrix of rank m, and $c\in\Re^n$. Of particular interest is a conjugate gradient (CG) approach to (1) and the issue of preconditioning.

If we remove the equality constraints from (1), and impose a positivity constraint on H, we are left with the well-studied positive definite linear equation problem; a preconditioned conjugate gradient approach (PCG) is given in Figure 1. Algorithm PCG is slightly nonstandard in that there is a test for negative curvature – if $\gamma\leq 0$ return (p,d) – useful in the optimization setting when the positivity of matrix H is sometimes uncertain. If negative curvature is discovered, i.e., $\gamma\leq 0$, the PCG output contains a direction of infinite descent. (A good place to start for references and background on conjugate gradient methods is the proceedings edited by Adams and Nazareth [1].)

The "art" of PCG is in the preconditioning step, $z\leftarrow\mathcal{P}(r)$: effective preconditioners are often tailored to the problem at hand. In the unconstrained case a common strategy is to approximate H with a sparser symmetric positive definite matrix, C, and then realize $z\leftarrow\mathcal{P}(r)$ by solving $Cz=-r$ using a sparse Cholesky factorization. There are many possible ways to choose C: most depend on sparsity or structure inherent in H.

For the linearly constrained problem (1), Coleman [2] suggests that algorithm PCG can be used if the preconditioning step, $z\leftarrow\mathcal{P}(r)$, is implemented in a special way. Specifically, solve

$$\begin{pmatrix} C & A^T \\ A & 0 \end{pmatrix}\begin{pmatrix} z \\ w \end{pmatrix}=\begin{pmatrix} r \\ 0 \end{pmatrix}, \tag{2}$$

where C is a "suitably sparse" positive definite approximation to the Hessian matrix H.

*Research 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-FG02-90ER25013.A000

†Computer Science Department and Cornell Theory Center, Cornell University, Ithaca NY 14850.

PCG

Initialize: Choose stopping tolerance tol, start point x_0,
set $\beta = 0$, $r = -(c + Hx_0)$,
$p = 0$, $d_- = 0$, and $z = \mathcal{P}(r)$

While $||z|| > tol$

1. $d \leftarrow z + \beta d_-, \qquad \gamma \leftarrow d^T H d$
2. If $\gamma \leq 0$ return (p, d)
   else $\alpha \leftarrow \frac{r^T z}{\gamma}, \quad p \leftarrow p + \alpha d, \quad r \leftarrow r - \alpha \bar{H} d$
3. $z \leftarrow \mathcal{P}(r)$
4. $\beta \leftarrow \frac{r^T z}{r_-^T z_-}$

Figure 1: PCG algorithm

While this approach can be very effective it suffers from two deficiencies. First, through accumulation of roundoff error, feasibility of vector p can be lost. (It is possible to correct for this with timely application of additional projections.) A second problem is that the 2-by-2 block matrix in (2), $M = \begin{pmatrix} C & A^T \\ A & 0 \end{pmatrix}$, requires the full original matrix A, else feasibility will be lost. Therefore, this approach is limited to problems where A is quite sparse (or m is small) so that the block matrix in (2) enjoys a sufficiently sparse (or inexpensive) factorization. Our new approach overcomes these two deficiencies. The cost is the need for a representation of the null space of A.

2 Reduced Preconditioned Conjugate Gradients

In theory a straightforward CG-approach to (1) is to introduce an n-by-$(n-m)$ matrix Z whose columns form a basis for null(A), and apply conjugate gradients to the reduced system

$$\bar{H}\bar{p} = -\bar{c} \tag{3}$$

where $\bar{H} = Z^T H Z, \bar{c} = Z^T c$. The full-dimensional (approximate) solution to (1) can then be recovered, $x \leftarrow Z\bar{p}$.

The corresponding reduced preconditioned conjugate gradient algorithm (RPCG), applied to equation (3), is shown in Figure 2.

The preconditioner is applied through $\bar{z} \leftarrow \mathcal{P}(\bar{r})$ which we think of as an easy-to-compute approximate solution to

$$\bar{H}\bar{z} = -\bar{r}. \tag{4}$$

This is our main challenge. Traditional preconditioning strategies for linear systems depend on a sparse matrix on the left-hand-side, matrix $\bar{H} = Z^T H Z$ in our case. However, even if the Hessian matrix H is quite sparse, it is unlikely that the reduced matrix $\bar{H}$ will be sufficiently sparse to be of practical use. Of course if m is large enough than this is not really an issue, since then matrix $\bar{H}$ will be small. However, for modest values of m the (lack of) sparsity in

RPCG

Initialize: Choose stopping tolerance tol, feasible starting point x_0.
Set $\beta = 0$, $r = -Z^T(c + Hx_0)$,
$p = 0$, $d_- = 0$, and $\bar{z} = \mathcal{P}(\bar{r})$

While $||\bar{z}|| > tol$

1. $\bar{d} \leftarrow \bar{z} + \beta \bar{d}_-, \qquad \gamma \leftarrow \bar{d}^T \bar{H} \bar{d}$
2. If $\gamma \leq 0$ return $(\bar{p}, \bar{d})$
   else $\alpha \leftarrow \frac{\bar{r}^T \bar{z}}{\gamma}, \quad \bar{p} \leftarrow \bar{p} + \alpha \bar{d}, \quad \bar{r} \leftarrow \bar{r} - \alpha \bar{H} \bar{d}$
3. $\bar{z} \leftarrow \mathcal{P}(\bar{r})$
4. $\beta \leftarrow \frac{\bar{r}^T \bar{z}}{\bar{r}_-^T \bar{z}_-}$

Figure 2: RPCG algorithm

$\bar{H}$ is a serious problem. Moreover, most traditional preconditioning strategies also require the explicit formulation of matrix $\bar{H}$ – this can be a non-trivial expense. (Nash and Sofers [10] have proposed an approach based on an approximation to (4).)

Our proposed method fits within the RPCG framework defined above. The output includes a vector $\bar{p}$, an approximate solution to (3) or a feasible direction of negative curvature. The full-dimensional representation can be computed, given $\bar{p}$, as $x \leftarrow x_0 + Z\bar{p}$. The defining features of our new approach are the use of a *fundamental* basis for null(A), and the form of the preconditioner $\bar{z} = \mathcal{P}(\bar{r})$.

A *fundamental basis* for the null space of matrix A, where A is m-by-n of rank m has the form

$$Z = P \begin{pmatrix} -A_1^{-1} A_2 \\ I_{n-m} \end{pmatrix} \tag{5}$$

where P is a permutation matrix and $AP = (A_1, A_2)$, A_1 is nonsingular. For additional discussions on fundamental bases, and alternatives, see [3, 4, 7]. For simplicity of this presentation we assume $P = I_n$ in (5).

We define, implicitly, the preconditioning operation $\bar{z} = \mathcal{P}(\bar{r})$ by an equation of the form

$$\begin{pmatrix} \tilde{H} & \tilde{A}^T \\ \tilde{A} & 0 \end{pmatrix} \begin{pmatrix} z \\ u \end{pmatrix} = \begin{pmatrix} 0 \\ -Z^T(Hx + c) = -\bar{r} \\ 0 \end{pmatrix} \tag{6}$$

where $\tilde{A} = (\tilde{A}_1, \tilde{A}_2)$ and $\tilde{A}_1$ is a nonsingular approximation to A_1, $\tilde{A}_2$ is an approximation to A_2, $\tilde{H}$ is a sparse symmetric positive definite approximation to H, and $z = \tilde{Z}\bar{z}$, where $\tilde{Z}$ is the corresponding fundamental basis for $\tilde{A}$. That is,

$$\tilde{Z} = \begin{pmatrix} -\tilde{A}_1^{-1} \tilde{A}_2 \\ I_{n-m} \end{pmatrix}. \tag{7}$$

It is now easy to see that solving for z via (6) is equivalent to solving

$$(\tilde{Z}^T \tilde{H} \tilde{Z})\bar{z} = -\bar{r}, \tag{8}$$

where $z = \tilde{Z}\bar{z}$. Clearly the inverse matrix $(\tilde{Z}^T\tilde{H}\tilde{Z})^{-1}$ can be viewed as an approximate inverse of Z^THZ; therefore, system (6) preconditions (4).

A key observation is that it is not necessary to compute $\tilde{Z}$ in order to obtain $\bar{z}$ or β (in RPCG). This is because $\tilde{Z}$ is a *fundamental* basis for null(A) and therefore $\bar{z} = (0, I_{n-m})z$, where z is computed using (6). Note that $\tilde{Z}$ does not appear on the right-hand-side of equation (6).

3 Constructing a Tilde Matrix, $\tilde{M}$

The framework developed above allows for many possible ways to choose the "tilde matrix" $\tilde{M}$,

$$\tilde{M} = \begin{pmatrix} \tilde{H} & \tilde{A^T} \\ \tilde{A} & 0 \end{pmatrix}, \tag{9}$$

in order to specify the preconditioning operation (6). Here we describe one possibility, with numerical results given in §4. We focus on the construction of $\tilde{A}$ since choosing a sparse symmetric positive definite approximation to H, $\tilde{H}$, is a well-studied problem with many possibilities. In our experiments we choose $\tilde{H}$ to be a positive diagonal matrix with $\tilde{H}(i,i)$ equal to the norm of column i, for $i = 1, ...n$.

The method we investigate uses the notion of a "drop tolerance", *droptl*. There are two basic steps to obtain $\tilde{A}$:

1. All rows of A are temporarily normalized to have unit norm.
2. The normalized matrix A is considered in two separate pieces, $A = (A_1, A_2)$, where A_1 is the leading nonsingular submatrix of A. Small nonzeroes of A_2 are "dropped", or set to zero, based on a tolerance. A similar approach is used on A_1 except that if the structural rank of A_1 decreases after dropping then the "zero tolerance" , *droptl*, for A_1 is increased and this process is repeated on A_1. The resulting matrix is $\tilde{A}$.

The MATLAB program which implements the above scheme is shown in Figure 3.

4 Computational Results

In this section we report on a number of computational experiments. All experiments are based on the "Brown" objective function [9], using a variety of linear constraint systems. In particular, our test problems have the form

$$\min\{c^Tx + \frac{1}{2}x^THx : \ Ax = b\} \tag{10}$$

with different choices for $A \in \Re^{m\times n}$, $b \in \Re^m$; c and H were chosen to be the gradient and the Hessian matrix for the "Brown" function,

$$y = \sum_{i=1}^{n-1} (x(i)^2)^{x(i+1)^2+1} + (x(i+1)^2)^{x(i)^2+1}.$$

at a given point, $x = (1, 1, ..., 1)^T$. Note that H is tridiagonal for all x. In our experiments we choose $\tilde{H}$ to be a positive diagonal matrix with $\tilde{H}(i,i)$ equal to the norm of column i of H, for $i = 1, ...n$.

The matrices A were chosen from public collections of matrices (see http://www.netlib.org/lp/data/). The CG tolerance was $tol = 10^{-3}$.

```
function A = gangstr(M,tol)

% Sparsify A
   [m,n] = size(M);
   if nargin < 2, droptl = 1e-2; end
%
% Normalize M
   Msqr=M.*M; X=sum(Msqr');
   X(find(X==0))=ones(length(find(X==0)),1); X=X(:);
   M=spdiags(1./sqrt(X),0,m,m)*M;
   A_1 = M(:,1:m); A_2 = M(:,m+1:n);

% Remove nonzeros from A_2 first
   [I_2, J_2, V_2]=find(A_2); absA_2=abs(A_2);
   maxvec=full(droptl*max(absA_2));
   tobekept=find(abs(V_2) > maxvec(J_2));
   A_2=sparse(I_2(tobekept),J_2(tobekept),V_2(tobekept),m,n-m);

% Remove nonzeros from A_1 making sure it has full structural rank
   dim = sprank(A_1);
   sprA_1 = 0;
   [I_1, J_1, V_1]=find(A_1); absA_1=abs(A_1);
   while sprA_1 < dim
         maxvec=full(droptl*max(absA_1));
         tobekept=find(abs(V_1) > maxvec(J_1));
         A_1=sparse(I_1(tobekept),J_1(tobekept),V_1(tobekept),m, m);
         sprA_1 = sprank(A_1);
         droptl = droptl-0.025;
   end

% Denormalize
   A=spdiags(sqrt(X),0,m,m)*[A_1   A_2];
```

Figure 3: Matlab Code to compute $\tilde{A}$ given $A = (A_1, A_2)$, A_1 nonsingular

Figure 4 illustrates the dependence of our method for choosing $\tilde{A}$ on the drop tolerance, *droptl*. We consider a particular example, test matrix qpleq7 with $n = 900$, $m = 358$. However, the dependence illustrated is typical over our test set. Efficiency, space and time, improve as *droptl* is increased from zero up to a point and then overall computing time begins to increase as the quality of the preconditoner degrades.

Plot (1,1) records the number of nonzeros in the preconditioner as a function of *droptl*. Plot (2,1) shows the combined sum of nonzeros in the LU factors of the preconditioner $\tilde{M}$, as a function of *droptl*. Note that while there is a general decrease in the number of nonzeroes in the LU factors, as *droptl* increases, the decrease is not monotone.

Plot (1,2) illustrates the dependence of the number of required CG-iterations as *droptl* is increased. As expected the number of iterations increases os *droptl* increases – consistent with the fact that the quality of the preconditioner correspondingly decreases. The increase in iterations is not monotone. Finally, and most interestingly, plot (2,2) shows the dependence of the overall computing time versus *droptl*. The behaviour illustrated is typical: as *droptl* increases from zero we observe a general decrease in computing time; a minimum is reached for some positive value of *droptl*, $droptl_*$, and a overall increase in computing time is observed beyond this point. (The increased sparsity of $\tilde{A}$ no longer compensates for the increasing number of required CG-iterations.)

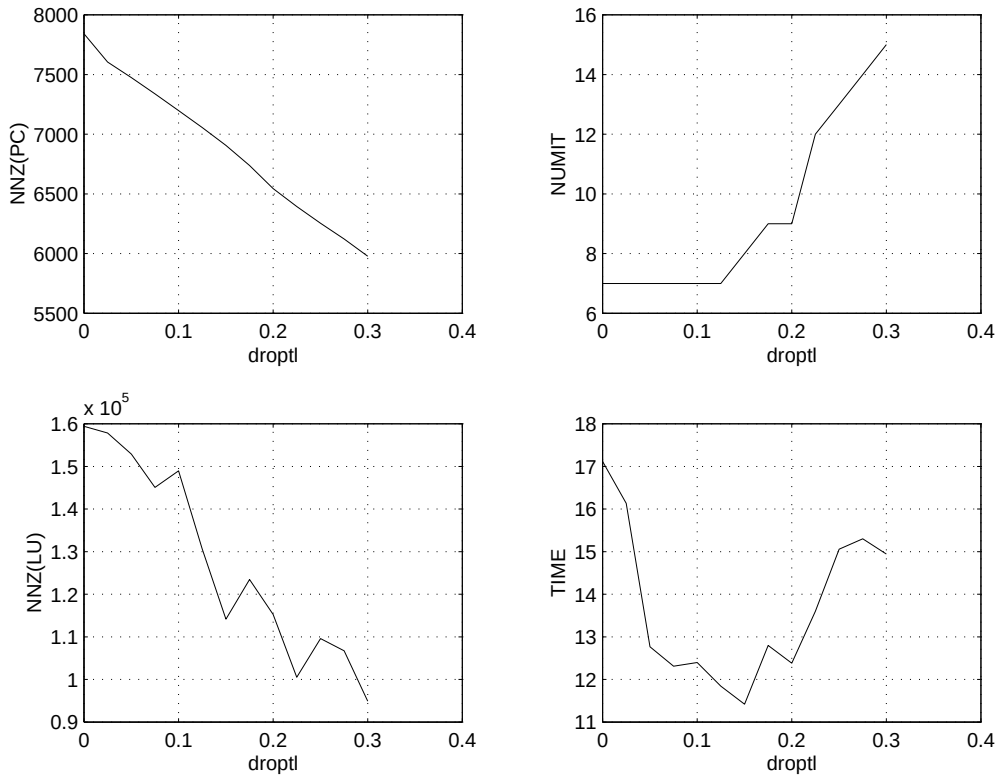

Figure 4: A sample preconditioner result

4.1 Observations on a Set of Constraint Matrices

The illustrative example above suggests that, for a given problem, there will be an optimal value of $droptl$, $droptl_*$, with respect to overall computing time to achieve a given accuracy. Unfortunately, the optimal value is not known a priori and it is far from clear how to efficiently compute the optimal value.

In our next experiment, in which we use a total of 39 different constraint matrices, we experimentally determine the "optimal" value of $droptl$ for each problem and then compare our approach to the alternatives of using a sparse direct solve ("backslash"), or, staying within the RPCG setting, choosing $droptl = 0$, i.e., $\tilde{A} = A$. The "optimal" value of $droptl$, for each problem (with given accuracy request $tol = .001$,) is determnined by applying RPCG using a discrete set of values for $droptl$: $droptl = 0, .025, ..., .5$, and chose the value of $droptl = droptl_*$ that corresponds to the best observed time.

Table 2 records the results: A is $m \times n$, $nnz(A)$ records the number of nonzeros in A, "backslash" denotes the time to directly solve the system(4). The next two columns record the CG timings for the two extremes, $droptl = 0$ and $droptl = droptl_*$. Finally we also record $droptl_*$, and the number of nonzeros in the optimal $\tilde{A}$.

On this test set the average value of $droptl_*$ is about 0.15. There is no case where $droptl_*$ is more than 0.45, and rarely is $droptl_*$ above 0.3.

A few cases are particularily noteworthy. For example, consider problem beaconfd where $nnz(\tilde{A}(droptl_*))$ is less than one-tenth of $nnz(A)$; typically, the reduction of nonzeroes is about 25%. Other examples of significant decrease in the number of nonzeroes include: grow15, grow7, and grow22.

Problem	$nnz(H)$	$Time(H)$	Backslash	Total	CG	Ratio1	Ratio2
qpleq7	293762	30.07	41.97	72.04	36.09	2.55	3.23
qpleq6	499623	28.45	68.52	96.97	40.57	3.36	3.43
pilot4	590179	31.55	87.21	118.76	63.62	3.76	13.04
fit1d	1050623	20.16	181.74	201.90	89.28	15.82	12.70
fleq2	739900	20.35	95.84	116.19	35.52	9.39	14.50

Table 1: Reduced system results

The typical improvement in timing between $droptl = 0$ and $droptl = droptl_*$ is about 25%. However, there are dramatic exceptions. Consider, for example, agg3, pilot4, boeing1, and etamarco.

Finally, the experiments confirm that if modest accuracy is required, i.e., $tol = .001$, then PCG is usually much faster than using the direct sparse solve (backslash). This difference clearly becomes more noticeable when $droptl = droptl_*$. Of course while our approach does allow for approximate solutions with respect to the reduced gradient, through the use of tol, high feasiblity accuracy is maintained (due to the use of Z).

We conclude this section with a brief comment on the comparison of our approach to the more straightforward technique of explicitly forming the reduced Hessian matrix and then solving the reduced Newton system, either directly or indirectly, via PCG. Table 1 shows some typical results using five of the constraint matrices from our collection.

Column "Time" indicates the time to actually form the reduced matrix whereas "Backslash" records the time to execute the MATLAB solve on the reduced system. "Total" is the sum of these two columns. "CG" records the time to obtain accuracy of $tol = .001$ using PCG on the reduced system with diagonal preconditioning. "Ratio1" is the total time to do the direct solve on the reduced system divided by the direct solve time on the extended system

$$\begin{pmatrix} C & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} s \\ w \end{pmatrix} = \begin{pmatrix} -c \\ b \end{pmatrix}.$$

"Ratio2" records the total time for applying PCG to the reduced system divided by the RPCG time with "optimal" choice of $droptl$.

Clearly the straightforward reduced approach does not appear to be competitive. Indeed, this small sample is consistent with our experiments on the entire test data set.

Finally we remark that feasibility is not an issue since RPCG works in the reduced space: on completion the full-space feasible solution is recovered, $x \leftarrow x_0 + Z\bar{p}$. In our experiments $\|Ax - b\|_2 \leq 10^{-12}$ over the complete test set.

5 Concluding Remarks.

Our proposed method is an attractive framework for implementing a preconditioned conjugate gradient approach to linearly constrained minimization. While the driving CG process is the *reduced* Newton system, often rather dense, the preconditioning step is based on the sparse (or structured) *extended* system. In this manner original structure or sparsity can be exploited in the design of the preconditioner. The special form of the right-hand-side of (6), and the restriction to the use of a *fundamental* basis, eliminates the need for a representation for null($\tilde{A}$).

The most obvious setting where our techniques will be of value is when H is sparse (and $m << n$). However, there are many practical contexts where H is structured, and not sparse, and this framework can still be of use. To illustrate, suppose that the Hessian matrix H has the form, $H = GE^{-1}G^T$, where E is sparse, symmetric, and positive definite; matrix G is sparse and of full row rank. This structure is common, e.g., [5, 6]. Clearly matrix H is dense due to the

Problem				Timing			Optim al parms	
Problem	m	n	nnz(A)	backslash	droptl = 0	droptl=droptl$_*$	droptl$_*$	nnz($\hat{A}_{droptl_*}$)
agg	488	615	2862	9.70	3.42	1.08	0.200	1250
agg2	516	758	4740	24.13	8.94	1.47	0.425	1252
agg3	516	758	4756	25.33	5.92	1.42	0.450	1241
bandm	305	472	2494	5.92	2.64	2.09	0.050	1966
beaconfd	173	295	3408	2.49	1.78	0.70	0.375	330
boeing1	351	726	3827	22.87	11.49	11.49	0.000	3827
capri	271	496	1965	4.31	2.05	1.94	0.025	1530
etamacro	400	816	2537	16.26	6.99	2.95	0.100	1977
finnis	497	1064	2760	17.77	5.45	4.81	0.025	2519
fit1d	24	1049	13427	12.76	8.69	7.03	0.025	13089
fleq1	20	1000	3480	15.41	3.35	2.78	0.150	2939
fleq2	50	1000	3617	12.37	2.91	2.45	0.025	3525
fleq2a	1217	2736	7887	83.89	17.21	12.89	0.300	6566
fleq6	20	920	3214	20.34	4.16	2.47	0.100	2886
fleq8	50	1000	2343	8.77	1.97	1.78	0.050	2305
fleq9	358	1000	3941	32.72	18.71	15.09	0.100	3564
forplan	161	492	4634	3.48	7.00	2.83	0.100	4167
ganges	1309	1706	6937	30.30	11.72	8.59	0.350	3469
grow15	300	645	5620	5.56	3.05	1.67	0.175	1119
grow22	440	946	8252	8.57	9.82	2.71	0.400	1387
grow7	140	301	2612	1.61	1.47	0.77	0.125	558
lotfi	153	366	1136	2.57	1.84	1.50	0.175	791
pilot4	410	1211	7342	31.58	15.66	4.88	0.200	2511
qpleq1	50	400	1442	2.06	0.98	0.88	0.100	1305
qpleq2a	10	900	2310	5.33	3.22	1.94	0.325	1646
qpleq2b	50	900	3221	11.53	2.00	1.98	0.125	2818
qpleq2c	300	900	4112	10.52	5.34	5.11	0.150	3449
qpleq3	100	512	2347	3.99	1.76	1.60	0.100	2077
qpleq6	193	900	5779	28.84	15.60	11.83	0.150	3464
qpleq7	358	900	3471	28.23	12.82	11.18	0.150	3003
qpleq8	50	900	1951	6.81	1.92	1.43	0.150	1899
recipe	91	204	687	0.66	0.54	0.43	0.275	568
scagr7	129	185	465	0.53	0.50	0.41	0.150	435
scfxm1	330	600	2732	4.04	2.61	2.03	0.075	2249
scfxm2	660	1200	5469	11.62	5.07	4.42	0.100	4244
scsd1	77	760	2388	6.60	2.06	1.91	0.250	2380
seba	515	1036	4360	21.92	5.61	4.85	0.050	4358
share1b	117	253	1179	1.21	0.94	0.85	0.050	1146
stair	356	620	4021	8.27	4.94	2.23	0.325	1174

Table 2: Preconditioner Results

application of E^{-1} (unless E is very special, e.g., diagonal). Direct preconditioning is a challenge due to the density of H. However, observe that the Newton step for $\min_x\{c^Tx + \frac{1}{2}x^THx\}$ can be written as the solution s to the system,

$$\begin{pmatrix} E & G^T \\ G & 0 \end{pmatrix} \begin{pmatrix} v \\ s \end{pmatrix} = \begin{pmatrix} 0 \\ c \end{pmatrix}. \tag{11}$$

Note that under our original assumptions the matrix in (11) is sparse. With respect to the RPCG approach to problem (1), it is now easy to see to that in this case the preconditioning step (6) can be modified: solve,

$$\begin{pmatrix} \tilde{E} & \tilde{G}^T & 0 \\ \tilde{G} & 0 & \tilde{A}^T \\ 0 & \tilde{A} & 0 \end{pmatrix} \begin{pmatrix} v \\ s \\ u \end{pmatrix} = \begin{pmatrix} 0 \\ \begin{pmatrix} 0 \\ -\bar{r} \end{pmatrix} \\ 0 \end{pmatrix}, \tag{12}$$

where $\tilde{E}$ is a sparse symmetric positive definite approximation to E, $\tilde{G}$ is a sparse (full row-rank) approximation to G, $\tilde{A}$ is a sparse (full row-rank) approximation to A. Note that under the positivity assumption on $\tilde{E}$, s is the solution to a reduced positive definite system of the form (4).

This is an example of how our proposed framework can be adapted to structured problems. A similar approach can be used for the general structures discussed in [5, 6].

Finally, we remark that our position in this paper is to consider a framework for the preconditioning of a conjugate gradient approach to (1). Restriction to conjugate gradient processes has the advantage that feasible descent directions are always generated - this is particularily important for nonlinear problems. Another iterative approach to (1) is to consider the (symmetric indefinite) system of equations defining the optimality conditions and to apply symmetric indefinite iterative techniques, e.g., [8]. We have not considered such methods here.

References

[1] L. Adams and J. Nazareth, *Linear and Nonlinear Conjugate Gradient-Related Methods*, SIAM, 1996.

[2] T. F. Coleman, *Linearly constrained optimization and projected preconditioned conjugate gradients*, in Proceedings of the Fifth SIAM Conference on Applied Linear Algebra, SIAM, Philadelphia, 1994, pp. 118–122.

[3] T. F. Coleman and A. Pothen, *The null space problem I: Complexity*, SIAM J. Alg. & Disc. Meth., 7 (1987), pp. 527–537.

[4] ——, *The null space problem II: Algorithms*, SIAM J. Alg. & Disc. Meth., 8 (1987), pp. 544–563.

[5] T. F. Coleman and A. Verma, *Structure and efficient Jacobian calculation*, in Computational Differentiation: Techniques, Applications, and Tools, M. Berz, C. Bischof, G. Corliss, and A. Griewank, eds., SIAM, Philadelphia, Penn., 1996, pp. 149–159.

[6] T. F. Coleman and A. Verma, *Structure and efficient Hessian calculation*, in Advances in Nonlinear Programming, Proceedings of the 1996 International Conference on Nonlinear Programming, Y.-X. Yuan, ed., Kluwer Academic Publishers, 1998.

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

[8] 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.

[9] J. J. Moré, B. S. Garbow, and K. H. Hillstrom, *Testing unconstrained optimization software*, ACM Trans. on Math. Software, 7 (1981), pp. 17–41.

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