Conjugate Gradient Methods for Saddle Point Systems

Lukšan, Ladislav
1999

INSTITUTE OF COMPUTER SCIENCE

ACADEMY OF SCIENCES OF THE CZECH REPUBLIC

Conjugate Gradient Methods for Saddle Point Systems

Ladislav Lukšan Jan Vlček

Technical report No. 778

September 1999

Institute of Computer Science, Academy of Sciences of the Czech Republic
Pod vodárenskou věží 2, 182 07 Prague 8, Czech Republic
phone: (+4202) 6884244 fax: (+4202) 8585789
e-mail: uivt@uivt.cas.cz

INSTITUTE OF COMPUTER SCIENCE

ACADEMY OF SCIENCES OF THE CZECH REPUBLIC

Conjugate Gradient Methods for Saddle Point Systems[1]

Ladislav Lukšan Jan Vlček

Technical report No. 778
September 1999

Abstract

Conjugate gradient methods have proved to be very efficient for solving saddle point systems, especially with indefinite preconditioning. These systems are indefinite and have a 2×2 block structure. We study various structured conjugate gradient methods and prove their theoretical properties.

Keywords

Nonlinear programming, sparse problems, equality constraints, saddle point systems, indefinite systems, indefinite preconditioners, conjugate gradient methods.

[1]This work was supported under grant No. 201/96/0918 given by the Czech Republic Grant Agency

1 Introduction

Saddle point equations arise in many applications including equality constrained optimization problems [4], [2] and the Stokes medium flow problems [3]. Here we use the notation from the first class of problems. Consider the problem of finding a point $x^* \in R^n$, such that

$$x^* = \arg\min_{x \in \mathcal{F}} F(x), \tag{1.1}$$

where $\mathcal{F} \subset R^n$ is a feasible set defined by the system of equations

$$\mathcal{F} = \{x \in R^n : c_k(x) = 0, 1 \leq k \leq m\}. \tag{1.2}$$

where $m \leq n$ (in fact we consider only local minima). Here $F : R^n \to R$ and $c_k : R^n \to R$, $1 \leq k \leq m$, are twice continuously differentiable functions, whose gradients and Hessian matrices will be denoted by $\nabla F(x)$, $\nabla c_k(x)$, $1 \leq k \leq m$, and $\nabla^2 F(x)$, $\nabla^2 c_k(x)$, $1 \leq k \leq m$, respectively. Furthermore, we use the notation $c(x) = [c_1(x), \ldots, c_m(x)]^T$ and $A(x) = [a_1(x), \ldots, a_m(x)] = [\nabla c_1(x), \ldots, \nabla c_m(x)]$ and we suppose that matrix $A(x)$ has a full column rank. Then the solution $x^* \in R^n$ of the problem (1.1)-(1.2) satisfies the Karush-Kuhn-Tucker (KKT) conditions, i.e., there exists a vector $u^* \in R^m$, such that

$$\begin{aligned} \nabla_x L(x^*, u^*) &= \nabla F(x^*) + A(x^*)u^* = 0, & (1.3)\\ \nabla_u L(x^*, u^*) &= c(x^*) = 0, & (1.4) \end{aligned}$$

where

$$L(x, u) = F(x) + u^T c(x) \tag{1.5}$$

is the Lagrangian function, whose gradient and Hessian matrix will be denoted by

$$\begin{aligned} g(x, u) &\equiv \nabla_x L(x, u) = \nabla F(x) + \sum_{k=1}^{m} u_k \nabla c_k(x),\\ G(x, u) &\equiv \nabla_x^2 L(x, u) = \nabla^2 F(x) + \sum_{k=1}^{m} u_k \nabla^2 c_k(x), \end{aligned}$$

and $(x^*, u^*) \in R^{n+m}$ is the KKT pair (first order necessary conditions). Let $Z(x)$ be a matrix whose columns form an orthonormal basis in the null space of $A^T(x)$ so that $A^T(x)Z(x) = 0$ and $Z^T(x)Z(x) = I$. If, in addition to (1.3)-(1.4), matrix $Z^T(x^*)G(x^*, u^*)Z(x^*)$ is positive definite, then the point $x^* \in R^n$ is a solution of the problem (1.1)-(1.2) (second order sufficient conditions).

The equations (1.3)-(1.4) can be solved by using the inexact Newton method whose iteration step has the form

$$\begin{aligned} x^+ &= x + \alpha d_x,\\ u^+ &= u + \alpha d_u, \end{aligned}$$

where $(d_x, d_u) \in R^{n+m}$ is a direction pair obtained as a solution of the following saddle point system

$$\begin{bmatrix} G(x,u) & A(x) \\ A^T(x) & 0 \end{bmatrix} \begin{bmatrix} d_x \\ d_u \end{bmatrix} = - \begin{bmatrix} g(x,u) \\ c(x) \end{bmatrix} \tag{1.6}$$

and $\alpha > 0$ is a suitable stepsize. Here we omit globalization theories concerning the choice of α, which lead to the use of exact penalty functions. However, we concentrate on the solution of (1.6), which is a system of $n + m$ linear equations with $n + m$ unknowns $(d_x, d_u) \in R^{n+m}$, whose matrix is always indefinite. Matrix $G(x, u)$ can be also indefinite even singular in general. This fact leads to some difficulties when standard iterative methods with standard preconditioners are used.

The contribution is organized as follows. In Section 2, we study several conjugate gradient methods for saddle point systems. We prove the correctness and the linear convergence of the CG method which uses the vertical step initially (Algorithm 2) and we show an equivalence of this method with the CG method applied to the reduced system (Algorithm 3). In Section 3, we present the results of numerical experiments which confirm high efficiency of CG methods applied to saddle point systems.

2 Conjugate gradient algorithms

We will investigate the saddle point system

$$\begin{bmatrix} B & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} d_x \\ d_u \end{bmatrix} = \begin{bmatrix} b_x \\ b_u \end{bmatrix} \tag{2.1}$$

(in easier notation $Kd = b$). Assume that matrix A has a full column rank and matrix $Z^T B Z$ is positive definite (matrix Z is defined as above). In [4], the special preconditioner

$$C = \begin{bmatrix} D & A \\ A^T & 0 \end{bmatrix} \tag{2.2}$$

is used, where D is a positive definite approximation of B. This preconditioner, called a constraint preconditioner, is also studied in [3]. It is clear that the multiplication by C^{-1} can be expressed in the form

$$\begin{aligned} C^{-1}r &= \begin{bmatrix} D^{-1} - D^{-1}A(A^TD^{-1}A)^{-1}A^TD^{-1} & D^{-1}A(A^TD^{-1}A)^{-1} \\ (A^TD^{-1}A)^{-1}A^TD^{-1} & -(A^TD^{-1}A)^{-1} \end{bmatrix} \begin{bmatrix} r_x \\ r_u \end{bmatrix} \\ &= \begin{bmatrix} D^{-1}(r_x - At_u) \\ t_u \end{bmatrix}, \end{aligned} \tag{2.3}$$

where $t_u = (A^TD^{-1}A)^{-1}(A^TD^{-1}r_x - r_u)$.

Consider first the standard conjugate gradient method with preconditioner C applied to the system $Kd = b$, which, using (2.1) and (2.3), can be implemented by the following algorithm.

Algorithm 1

d_x – given (usually $d_x = 0$), $\quad d_u = 0$,
$r_x = b_x - Bd_x, \quad r_u = b_u - A^T d_x$,
$t_x = D^{-1}(r_x - At_u), \quad t_u = (A^T D^{-1} A)^{-1}(A^T D^{-1} r_x - r_u)$,
$p_x = t_x, \quad p_u = t_u$,
$\rho = r_x^T t_x + r_u^T t_u$,

while $\|r_x\| > \omega\|b_x\|$ or $\|r_u\| > \omega\|b_u\|$ **do**

$q_x = Bp_x + Ap_u, \quad q_u = A^T p_x$,
$\sigma = p_x^T q_x + p_u^T q_u, \quad \alpha = \rho/\sigma$,
$d_x^+ = d_x + \alpha p_x, \quad d_u^+ = d_u + \alpha p_u$,
$r_x^+ = r_x - \alpha q_x, \quad r_u^+ = r_u - \alpha q_u$,
$t_x^+ = D^{-1}(r_x^+ - At_u^+), \quad t_u^+ = (A^T D^{-1} A)^{-1}(A^T D^{-1} r_x^+ - r_u^+)$,
$\rho^+ = (r_x^+)^T t_x^+ + (r_u^+)^T t_u^+, \quad \beta = \rho^+/\rho$,
$p_x^+ = t_x + \beta p_x, \quad p_u^+ = t_u + \beta p_u$,
(delete +),

end while.

This algorithm has one nonstandard feature, namely the termination criterion $\|r_x\| \le \omega\|b_x\|$, $\|r_u\| \le \omega\|b_u\|$, where $0 \le \omega < 1$, which is required by the inexact Newton method applied to nonlinear KKT equations (1.3)-(1.4).

The above general algorithm has two possibilities to break down. Either $\sigma = p^T q$ can be equal to zero, since K is indefinite, or $\rho = r^T t$ can be equal to zero, since C is indefinite. In [4] it has been proved that these breakdowns cannot occur before the solution d_x^* is reached whenewer the initial approximation is chosen in such a way that $r_u = b_u - A^T d_x = 0$. Now we will investigate this case in more details.

Assume that the initial approximation is the vertical direction given by the formula $d_x = D^{-1}A(A^T D^{-1} A)^{-1} b_u$ so that $r_u = b_u - A^T d_x = 0$. If we use the following unique representations

$$
\begin{aligned}
&d_x = Zd_Z + D^{-1}Ad_A, \quad t_x = Zt_Z + D^{-1}At_A, \quad p_x = Zp_Z + D^{-1}Ap_A,\\
&r_x = DZ(Z^T DZ)^{-1} r_Z + Ar_A \;\Rightarrow\; r_Z = Z^T r_x, \quad r_A = (A^T D^{-1} A)^{-1} A^T D^{-1} r_x,\\
&q_x = DZ(Z^T DZ)^{-1} q_Z + Aq_A \;\Rightarrow\; q_Z = Z^T q_x, \quad q_A = (A^T D^{-1} A)^{-1} A^T D^{-1} q_x
\end{aligned}
$$

and the fact that $Z^T A = 0$, then the formulas in Algorithm 1 can be rewritten in the following theoretical form (we assume exact computations).

Algorithm 1a

$d_Z = 0, \quad d_A = (A^T D^{-1} A)^{-1} b_u, \quad d_u = 0$,
$r_Z = Z^T(b_x - BD^{-1}Ad_A), \quad r_A = (A^T D^{-1} A)^{-1} A^T D^{-1}(b_x - BD^{-1}Ad_A), \quad r_u = 0$,
$t_Z = (Z^T DZ)^{-1} r_Z, \quad t_A = 0, \quad t_u = r_A$,
$p_Z = t_Z, \quad p_A = 0, \quad p_u = t_u$,
$\rho = r_Z^T t_Z = r_Z^T (Z^T DZ)^{-1} r_Z$,

while $\|r_x\| > \omega\|b_x\|$ or $\|r_u\| > \omega\|b_u\|$ **do**

$q_Z = Z^T B Z p_Z, \quad q_A = (A^T D^{-1} A)^{-1} A^T D^{-1} B Z p_Z + p_u, \quad q_u = 0$
$\sigma = p_Z^T q_Z = p_Z^T Z^T B Z p_Z, \quad \alpha = \rho/\sigma,$
$d_Z^+ = d_Z + \alpha p_Z, \quad d_A^+ = d_A, \quad d_u^+ = d_u + \alpha p_u,$
$r_Z^+ = r_Z - \alpha q_Z, \quad r_A^+ = r_A - \alpha q_A, \quad r_u^+ = 0,$
$t_Z^+ = (Z^T D Z)^{-1} r_Z^+, \quad t_A^+ = 0, \quad t_u^+ = r_A^+,$
$\rho^+ = (r_Z^+)^T t_Z^+ = (r_Z^+)^T (Z^T D Z)^{-1} r_Z^+, \quad \beta = \rho^+/\rho,$
$p_Z^+ = t_Z^+ + \beta p_Z, \quad p_A^+ = 0, \quad p_u^+ = t_u + \beta p_u,$
(delete +),

end while.

See also the proof of Theorem 3.5 in [4]. Algorithm 1a implies several coclusions. The iterations for d_Z are the same as iterations of conjugate gradient method with preconditioner $Z^T D Z$ applied to the reduced system $Z^T B Z d_Z = b_Z$ where $b_Z = Z^T(b_x - B D^{-1} A d_A)$. Since Z has a full column rank and D is positive definite, this method cannot break down before solution d_Z^* is obtained. Therefore, Algorithm 1a cannot break down before the solution d_x^* is obtained either. Since d_A is the same in each iteration, we can write $d_x - d_x^* = Z(d_Z - d_Z^*)$ (Z is an orthogonal matrix) so that the convergence rate for $d_x \to d_x^*$ is the same as those for $d_Z \to d_Z^*$. Nevertheless, there is a serious difficulty. If $d_x = d_x^*$ then $d_u = d_u^*$ usually does not hold, but Algorithm 1a breaks down because $\rho = 0$. Fortunately, d_u^* can be easily found as a solution of equation $A(d_u^* - d_u) = r_x$ in this case. Taking this fact into account, Algorithm 1 can be slightly modified. Since $\|r_u\| = 0$ when $d_x = D^{-1} A (A^T D^{-1} A)^{-1} b_u$ is chosen initially, we can replace the termination criterion $\|r_x\| \le \omega\|b_x\|$ by $\rho \le \omega\overline{\rho}$, where $\overline{\rho}$ is the initial value of ρ. After termination we add $t_u = (A^T D^{-1} A)^{-1} A^T D^{-1} r_x$ to d_u. The following algorithm is the corresponding modification of Algorithm 1.

Algorithm 2

$d_x = D^{-1} A (A^T D^{-1} A)^{-1} b_u, \quad d_u = 0,$
$r_x = b_x - B d_x, \quad r_u = b_u - A^T d_x,$
$t_x = D^{-1}(r_x - A t_u), \quad t_u = (A^T D^{-1} A)^{-1}(A^T D^{-1} r_x - r_u),$
$p_x = t_x, \quad p_u = t_u,$
$\rho = r_x^T t_x + r_u^T t_u, \quad \overline{\rho} = \rho,$

while $\rho > \omega\overline{\rho}$ **do**

$q_x = B p_x + A p_u, \quad q_u = A^T p_x,$
$\sigma = p_x^T q_x + p_u^T q_u, \quad \alpha = \rho/\sigma,$
$d_x^+ = d_x + \alpha p_x, \quad d_u^+ = d_u + \alpha p_u,$
$r_x^+ = r_x - \alpha q_x, \quad r_u^+ = r_u - \alpha q_u,$
$t_x^+ = D^{-1}(r_x^+ - A t_u^+), \quad t_u^+ = (A^T D^{-1} A)^{-1}(A^T D^{-1} r_x^+ - r_u^+),$
$\rho^+ = (r_x^+)^T t_x^+ + (r_u^+)^T t_u^+, \quad \beta = \rho^+/\rho,$
$p_x^+ = t_x + \beta p_x, \quad p_u^+ = t_u + \beta p_u,$
(delete +),

end while,

set $d_u := d_u + t_u$.

Notice that Algorithm 2 only differs from Algorithm 1 by the initial choice of d_x, a different termination criterion and a final redefinition of d_u. The following theorem holds if the computations are exact.

Theorem 1. *Consider Algorithm 2 (with $\omega = 0$) applied to the saddle point system (2.1). Assume that matrix A has a full column rank and matrix Z^TBZ is positive definite. Then:*
(a) The solution pair (d_x^*, d_u^*) is found after $n - m$ iterations at most.
(b) The algorithm cannot break down.
(c) The error $\|d_x - d_x^*\|$ converges to zero at last R - linearly with the quotient $(\sqrt{\kappa} - 1)/(\sqrt{\kappa} + 1)$ where $\kappa = \kappa(Z^TBZ(Z^TDZ)^{-1})$ is a spectral condition number of matrix $Z^TBZ(Z^TDZ)^{-1}$.

Proof. Theorem 1 follows immediately from our above considerations (see also the proof of Theorem 3.5 in [4]).

Notice that $r_u = 0$ if the computations are exact, which could make Algorithm 2 easier. However, the proposed form is more stable with respect to the round-off errors.

Now let us compare Algorithm 2 with the preconditioned conjugate gradient method applied to the reduced system, which is proposed in [2]. Consider again the unique representation $d_x = Zd_Z + D^{-1}Ad_A$. Using the second equation in (2.1), we get $d_A = (A^TD^{-1}A)^{-1}b_u$ and the first one gives $BZd_Z = b_x - BD^{-1}Ad_A - Ad_u$ which after premultiplying by Z^T gives

$$Z^TBZd_Z = b_Z, \tag{2.4}$$

where $b_Z = Z^T(b_x - BD^{-1}Ad_A)$. Thus d_Z can be obtained by the conjugate gradient method with preconditioner Z^TDZ applied to equation (2.4). Unfortunately, matrix Z is not usually known (its computation is time-consuming and difficult for large sparse A because of fill-in). For this reason, Gould, Hribar and Nocedal [2] modify conjugate gradient iterations in such a way that they use vectors $d_z = Zd_Z$, $t_z = Zt_Z$, $p_z = Zp_Z$ and r_z, q_z, such that $r_Z = Z^Tr_z$, $q_Z = Z^Tq_z$. Then preconditioning $t_Z = (Z^TDZ)^{-1}r_Z$ can be expressed in the form

$$t_z = Z(Z^TDZ)^{-1}Z^Tr_z = (D^{-1} - D^{-1}A(A^TD^{-1}A)^{-1}A^TD^{-1})r_z \tag{2.5}$$

so that matrix Z need not be used explicitly. Since d_z does not influence formulas in conjugate gradient iterations directly, we can use $d_x = d_z + D^{-1}Ad_A$ instead of d_z. Since d_u cannot be obtained from conjugate gradient iterations, it has to be estimated in any way. The most natural choice is the weighted least-square minimization of the total residuum $r_z - Ad_u$. This way leads to formula $d_u = (A^TD^{-1}A)^{-1}A^TD^{-1}r_z$. The following algorithm summarizes the above considerations.

Algorithm 3

$d_x = D^{-1}A(A^TD^{-1}A)^{-1}b_u$
$r_z = b_x - Bd_x,$
$t_z = D^{-1}(r_z - At_u), \quad t_u = (A^TD^{-1}A)^{-1}A^TD^{-1}r_z$
$p_z = t_z,$
$\rho = r_z^Tt_z, \quad \overline{\rho} = \rho.$

while $\rho > \omega\overline{\rho}$ **do**

$q_z = Bp_z,$
$\sigma = p_z^Tq_z, \quad \alpha = \rho/\sigma,$
$d_x^+ = d_x + \alpha p_z,$
$r_z^+ = r_z - \alpha q_z,$
$t_z^+ = D^{-1}(r_z^+ - At_u^+), \quad t_u^+ = (A^TD^{-1}A)^{-1}A^TD^{-1}r_z^+$
$\rho^+ = (r_z^+)^Tt_z^+, \quad \beta = \rho^+/\rho,$
$p_z^+ = t_z^+ + \beta p_z,$
(delete +),

end while,

set $d_u := t_u$.

Theorem 2. *Algorithm 2 and Algorithm 3 are equivalent in the sense that both generate the same vectors d_x and give the same final pair (d_x, d_u).*

Proof. By definition, $d_z = Zd_Z$, $t_z = Zt_Z$, $p_z = Zp_Z$, where d_Z, t_Z, p_Z are vectors obtained by the conjugate gradient method with preconditioner Z^TDZ applied to the system (2.4). But vectors d_Z, t_Z, p_Z in Algorithm 1a have the same property. Since $t_A = 0$, $p_A = 0$ in Algorithm 1a, we can set $t_x = Zt_Z = t_z$, $p_x = Zp_Z = p_z$ in Algorithm 2. Moreover, the same initial vector $d_x = D^{-1}A(A^TD^{-1}A)^{-1}b_u$ is used in both Algorithm 2 and Algorithm 3. Therefore both algorithms generate the same vectors d_x. The final vector d_u is also the same in both algorithms, since it is the least-square solution of the equation $Bd_x + Ad_u = b_x$. □

Theorem 2 says that the conjugate gradient method with preconditioner (2.2) applied to original system (2.1) is equivalent to the conjugate gradient method with preconditioner Z^TDZ applied to reduced system (2.4), when the vertical direction $d_x = D^{-1}A(A^TD^{-1}A)^{-1}b_u$ is chosen initially.

3 Numerical experiments

The algorithms described above can be used for solving large saddle-point systems arising in equality constrained optimization. In this case, matrix $A^TD^{-1}A$ should be sparse. Unfortunately, this is not satisfied when matrix A contains dense rows. Therefore, some modifications of the algorithms are necessary. The first modification is based on the Woodbury theorem. Assume that $A^T = [A_s^T, A_d^T]$ where $A_s^TD_s^{-1}A_s$ is sparse and A_d contains dense columns. Then

$$
\begin{aligned}
(A^T D^{-1} A)^{-1} &= (A_s^T D_s^{-1} A_s + A_d^T D_d^{-1} A_d)^{-1} \\
&= (A_s^T D_s^{-1} A_s)^{-1} - (A_s^T D_s^{-1} A_s)^{-1} A_d^T M_d^{-1} A_d (A_s^T D_s^{-1} A_s)^{-1} \quad (3.1)
\end{aligned}
$$

where

$$
M_d = D_d + A_d (A_d^T D_d^{-1} A_d)^{-1} A_d^T
$$

is a (low-dimensional) dense matrix. Another modification is based on the sparse Bunch-Parlet decomposition. Since

$$
\begin{bmatrix} D & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} t_x \\ t_u \end{bmatrix} = \begin{bmatrix} r_x \\ r_u \end{bmatrix}, \quad (3.2)
$$

vectors t_x and t_u can be easily obtained after decomposition of the left-hand side matrix. Numerical stability of both of these possibilities is studied in [2].

Now we are in the position to introduce the results of our experiments. We have used a collection of 18 equality constrained problems with 1000 variables proposed in report [5], which can be downloaded (together with corresponding fortran codes) from the home-page `http://www.uivt.cas.cz/ ~luksan/`. The following table contains total numbers of outer iterations (numbers of saddle-point systems solved) NSP, total numbers of inner (conjugate-gradient) iterations NCG and total CPU times over all 18 problems. We have tested three above algorithms with two realizations, (3.1) and (3.2). Moreover, the results for trust region method [1] are added for comparison.

Algorithm	NSP	NCG	CPU
Algorithm 1 / (3.1)	306	1015	31.09
Algorithm 1 / (3.2)	306	1015	33.73
Algorithm 2 / (3.1)	330	738	34.66
Algorithm 2 / (3.2)	329	749	36.96
Algorithm 3 / (3.1)	311	598	29.60
Algorithm 3 / (3.2)	311	598	34.39
Trust region [1]	518	1452	39.88

Table 1

Table 1 shows high efficiency of the conjugate-gradient methods applied to saddle point systems. Surprisingly, Algorithm 1 (with the initial choice $d_x = 0$) is very efficient and competitive with Algorithm 3 although it can lead to a breakdown theoretically. In Algorithm 1, numbers ρ and σ are not guaranteed to be positive. They have sometimes acquired negative values in our computations.

Bibliography

[1] Dennis J.E., Vicente L.N.: On the convergence theory of trust-region-based algorithms for equality-constrained optimization. SIAM J. Optimization, Vol.7, 1997, pp.927-950.

[2] Gould N.I.M., Hribar M.E., Nocedal J.: On the solution of equality constrained quadratic programming problems arising in optimization. Technical Report RAL-TR-1998-069, Rutherford Appleton Laboratory, 1998.

[3] Keller C., Gould N.I.M., Wathen A.J.: Constraint preconditioning for indefinite linear systems. Technical Report RAL-TR-1999-016, Rutherford Appleton Laboratory, 1999.

[4] Lukšan L., Vlček J.: Indefinitely preconditioned truncated Newton method for large sparse equality constrained nonlinear programming problems. Numerical Linear Algebra with Applications, Vol.5, 1998, pp.219-247.

[5] Lukšan L., Vlček J.: Subroutines for testing large sparse and partially separable unconstrained and equality constrained optimization problems. Report No. 767, Institute of Computer Science, Academy of Sciences of the Czech Republic, Prague 1999.