Jitka Křížková; Petr Vaněk

Explicit conjugate gradient method with preconditioning

Applications of Mathematics, Vol. 39 (1994), No. 4, 309–316

Persistent URL: `http://dml.cz/dmlcz/134260`

EXPLICIT CONJUGATE GRADIENT METHOD WITH PRECONDITIONING

Jitka Křížková, Petr Vaněk, Plzeň

(Received February 5, 1993)

Summary. An algorithm of the preconditioned conjugate gradient method in which the solution of an auxiliary system is replaced with multiplication by the matrix $M = I - \omega A$ for suitably chosen ω is presented.

Keywords: Conjugate gradient method, preconditioning

AMS classification: 65F10

1. Introduction

The preconditioned conjugate gradient method requires to solve an auxiliary system of linear algebraic equations in each step. The aim of this paper is to describe an algorithm of the preconditioned conjugate gradient method in which the solution of an auxiliary system is replaced with multiplication by the matrix $M = I - \omega A$. Convergence properties are analysed and acceleration of the process in comparison with the standard conjugate gradient method is shown. Generalization based on the multiple usage of M is presented, too.

2. Algorithm

We will solve the system of linear algebraic equations

$$Ax = b$$

where A is a positive definite real matrix of order n. We will denote by (x, y) the usual scalar product in R_n, the norm in R_n being $\|x\| = (x, x)^{\frac{1}{2}}$.

For an arbitrary linear operator L on R_n, $\|L\|$ denotes the operator norm of L defined by the vector norm $\|x\|$ and $\|L\|_A$ denotes the operator norm of L defined by the vector norm $\|x\|_A = (Ax, x)^{\frac{1}{2}}$.

For an arbitrary positive definite operator B on R_n let us define the so called condition number

$$\mathrm{cond}(B) = \frac{\lambda_{\max}(B)}{\lambda_{\min}(B)},$$

where $\lambda_{\max}(B)$ denotes the maximal and $\lambda_{\min}(B)$ the minimal eigenvalue of B. Let us denote by $\sigma(B)$ the spectrum of B. Let C be a positive definite operator on R_n. Usage of the preconditioned conjugate gradient method algorithm follows the scheme:

Step 1. Given $\varepsilon > 0$, $x_0 = 0$, let $k = 0$ and

$$\begin{aligned} r_0 &= b - Ax_0 = b, \\ h_0 &= C^{-1} r_0, \\ p_0 &= h_0. \end{aligned}$$

Step 2. Do

$$\begin{aligned} \alpha_k &= \frac{(p_k, r_k)}{(Ap_k, p_k)}, \\ x_{k+1} &= x_k + \alpha_k p_k, \\ r_{k+1} &= r_k - \alpha_k A p_k, \\ h_{k+1} &= C^{-1} r_{k+1}, \\ \beta_k &= \frac{(r_{k+1}, h_{k+1})}{(r_k, h_k)}, \\ p_{k+1} &= h_{k+1} + \beta_k p_k. \end{aligned} \tag{2.1}$$

Step 3. Let c_k be an estimate of $\mathrm{cond}(C^{-1}A)$. If $c_k \frac{(r_k, h_k)}{(r_0, h_0)} \leqslant \varepsilon^2$, exit else $k = k + 1$ and go to Step 2—see [1].

Let us define the error $e(x)$ by $e(x) = x - \widehat{x}$, where $\widehat{x} = A^{-1} b$.

Then for the error of the preconditioned conjugate gradient method the following formula can be derived—see [3]

$$\|e(x_i)\|_A \leqslant 2 \left(\frac{\sqrt{\mathrm{cond}(C^{-1}A)} - 1}{\sqrt{\mathrm{cond}(C^{-1}A)} + 1} \right)^i \|e(x_0)\|_A. \tag{2.2}$$

Let us note that for $C = I$ the algorithm above is just the standard conjugate gradient algorithm (i.e. without preconditioning). It is evident that the smaller

$\operatorname{cond}(C^{-1}A)$, the better estimate (2.2) we get. Now let us define $C = (I - \omega A)^{-1}$, where ω is positive, not greater than $[\lambda_{\max}(A)]^{-1}$. Therefore $C^{-1} = I - \omega A$ and the solution of the auxiliary system of linear algebraic equations in (2.1) becomes multiplication by the matrix $M = I - \omega A$. In the estimate (2.2) the number $\operatorname{cond}(C^{-1}A)$ will be replaced by $\operatorname{cond}(MA)$. Our goal is to estimate $\operatorname{cond}(MA)$ and not only to show that $\operatorname{cond}(MA) < \operatorname{cond}(A)$ but to get a quantitative estimate of this fact. Let us note that $\operatorname{cond}(A)$ occurs in (2.2) when using the standard conjugate gradient algorithm (i.e. $C = I$).

Theorem 1. *Let A be a positive definite real matrix of order n, let $0 < \lambda_1 \leqslant \lambda_2 \ldots \leqslant \lambda_n$ be the eigenvalues of A, let $\overline{\omega} = (\lambda_n + \lambda_1)^{-1}$, $M = I - \omega A$, where $\omega \in [0, \overline{\omega}]$. Then*

$$\operatorname{cond}(MA) \leqslant f(\omega) \operatorname{cond}(A) \tag{2.3}$$

where

$$f(\omega) = \begin{cases} \dfrac{1 - \omega\lambda_n}{1 - \omega\lambda_1} & \text{for } \omega \in \left[0, \dfrac{1}{2\lambda_n}\right] \\ \dfrac{1}{4\lambda_n\omega(1 - \omega\lambda_1)} & \text{for } \omega \in \left(\dfrac{1}{2\lambda_n}, \overline{\omega}\right]. \end{cases}$$

The function $f(\omega)$ is continuous and decreasing in $[0, \overline{\omega}]$,

$$f(0) = 1, \quad f\left(\frac{1}{2\lambda_n}\right) = \frac{\lambda_n}{2\lambda_n - \lambda_1}, \quad f(\overline{\omega}) = \frac{1}{4}\left(1 + \frac{\lambda_1}{\lambda_n}\right)^2.$$

Remark 2.1. For $\omega \in [0, \frac{1}{2\lambda_n}]$ the inequality (2.3) is in fact the equality.

Proof. All eigenvalues of MA are of the form

$$\mu_i = \lambda_i(1 - \omega\lambda_i), \quad i = 1, 2, \ldots, n$$

and $\mu_i > 0$ as MA is positive definite. If we choose $\omega \leqslant (\lambda_i + \lambda_1)^{-1}$ for all i then $\mu_1 = \lambda_1(1 - \omega\lambda_1)$ is the smallest eigenvalue of MA, i.e. $\mu_1 \leqslant \mu_i$ for all i. For $\omega \in [0, \overline{\omega}]$ obviously $\omega \leqslant (\lambda_i + \lambda_1)^{-1}$ and therefore $\mu_1 = \min \lambda_i(1 - \omega\lambda_i)$. Hence there exists a j such that

$$\operatorname{cond}(MA) = \frac{\lambda_j(1 - \omega\lambda_j)}{\lambda_1(1 - \omega\lambda_1)}.$$

Let us denote by p the quadratic polynomial

$$p(t) = t(1 - \omega t).$$

The minimal value of p equals $\frac{1}{4\omega}$ and it is attained for $t = \frac{1}{2\omega}$. It is evident that

$$\operatorname{cond}(MA) = \frac{\max\limits_{t \in \sigma(A)} p(t)}{p(\lambda_1)} \leqslant \frac{\max\limits_{t \in [\lambda_1, \lambda_n]} p(t)}{p(\lambda_1)}.$$

For $\omega \in (\frac{1}{2\lambda_n}, \overline{\omega}]$, i.e. ω satisfying $\lambda_1 \leqslant \frac{1}{2\omega} \leqslant \lambda_n$ we get

$$\operatorname{cond}(MA) \leqslant \frac{1}{4\omega p(\lambda_1)} = \frac{1}{4\lambda_n\omega(1-\omega\lambda_1)} \operatorname{cond}(A),$$

for $\omega \in [0, \frac{1}{2\lambda_n}]$ the correspoding formula reads

$$\operatorname{cond}(MA) = \frac{1-\omega\lambda_n}{1-\omega\lambda_1} \operatorname{cond}(A).$$

The above inequalities prove (2.3). The remaining properties of $f(\omega)$ are easy to prove. □

3. Generalization

Generalization of the above algorithm will be presented in this section.

Generalization follows the following idea: Instead of preconditioning by the matrix $C = M^{-1}$ we will use preconditioning by the matrix $C = (M'M)^{-1}$, where $M' = I - \omega' A'$, $A' = MA$ for ω' suitably chosen. To give estimates for condition numbers we start with some notation.

Definition 3.1. Let us define sequences $\{A_i\}$, $\{M_i\}$ as follows:

$$A_0 = A, \quad \omega_0 = \omega, \quad M_0 = I - \omega_0 A_0, \quad f_0(\omega_0) = f(\omega),$$
$$A_{i+1} = M_i A_i, \quad M_{i+1} = I - \omega_i A_i,$$

where ω_i is a positive real number.

Theorem 2. *Let A be a positive definite real matrix of order n. Let $\{A_i\}$, $\{M_i\}$ be sequences from the previous definition. Let us denote by $\lambda_1(A_i)$ the minimal and by $\lambda_n(A_i)$ the maximal eigenvalue of A_i for all i. Let us set $\overline{\omega}_i = (\lambda_n(A_i) + \lambda_1(A_i))^{-1}$. For $\omega_i \in [0, \overline{\omega}_i]$*

$$\operatorname{cond}(A_{i+1}) \leqslant f_i(\omega_i) \operatorname{cond}(A_i),$$

where

$$f_i(\omega_i) = \begin{cases} \dfrac{1-\omega_i\lambda_n(A_i)}{1-\omega_i\lambda_1(A_i)} & \text{for } \omega_i \in \left[0, \dfrac{1}{2\lambda_n(A_i)}\right] \\[2ex] \dfrac{1}{4\lambda_n(A_i)\omega_i(1-\omega_i\lambda_1(A_i))} & \text{for } \omega_i \in \left(\dfrac{1}{2\lambda_n(A_i)}, \overline{\omega}_i\right], \end{cases}$$

f_i being continuous and decreasing.

Proof. The proof follows from Theorem 1 immediately. □

Corollary 3.3. *For large* $\mathrm{cond}(A_i)$ *we have*

$$f_i\Big(\frac{1}{2\lambda_n(A_i)}\Big) \approx \frac{1}{2},$$

$$f_i(\overline{\omega}_i) \approx \frac{1}{4},$$

f_i *being decreasing in* $[\frac{1}{2\lambda_n(A_i)}, \overline{\omega}_i]$.

Proof. The proof follows from the fact that

$$f_i\Big(\frac{1}{2\lambda_n(A_i)}\Big) = \frac{\lambda_n(A_i)}{2\lambda_n(A_i) - \lambda_1(A_i)},$$

$$f_i(\overline{\omega}_i) = \frac{1}{4}\Big(1 + \frac{\lambda_1(A_i)}{\lambda_n(A_i)}\Big)^2.$$

□

4. Practical remarks and numerical experiments

The full scheme for determining $C^{-1}x$ is the following:

$$C^{-1}x = \prod_{i=0}^{k} M_i x,$$

where

$$M_i = I - \omega_i A_i,$$
$$A_i = M_{i-1}A_{i-1},$$
$$A_0 = A.$$

The number k is chosen. Let us note that this scheme can be very easily programmed using recursion. For large k the prevailing operations are those for determining Ax, which can be done parallelly. When using the algorithm described above we must choose ω_i for $i = 1, 2 \dots k$. We will start with l_0, L_0 chosen so that

$$l_0 \geqslant \lambda_1(A_0), \quad L_0 \geqslant \lambda_n(A_0), \quad l_0 + L_0 \leqslant 2\lambda_n(A_0).$$

Generation of ω_i, $i = 0, 1 \dots$ follows the scheme:

$$\omega_i = \frac{1}{l_i + L_i},$$

$$L_{i+1} = \frac{1}{4\omega_i},$$
$$l_{i+1} = l_i(1 - \omega_i l_i).$$

It is not difficult to see that ω_i just defined fulfil the condition

$$\omega_i \in \left[\frac{1}{2\lambda_n}, \overline{\omega}_i\right].$$

To test the algorithm we consider the model problem

$$-\Delta u = f(x,y) \text{ on } \Omega = [0,1] \times [0,1]$$
$$u = 0 \text{ on } \partial\Omega,$$

where

$$f(x,y) = x^2\sqrt{y} + \sqrt{x}ye^{5xy}.$$

Using the discretization on the square grid by the finite differences method leads to the system of linear algebraic equations $Ax = b$. Computations were carried out for the initial choice $l_0 = 0.1$, $L_0 = 8$. The results can be found in the following table:

n	k	iterations	time of computing
3600	0	263	47s
	1	141	31s
	2	73	20s
	3	39	14s
2500	0	233	29s
	1	119	18s
	2	61	11s
	3	31	8s
625	0	119	3.90s
	1	62	2.48s
	2	36	1.76s
	3	20	1.38s

Required accuracy was 10^{-13}, n denotes the number of equations.

```
Unit Precondition;
Interface
const
     n   = 2500; {dimension of the problem}
     k   = 3;    {number of recursive steps}
type
     Vector = array[1..N] of real;
Procedure SetOmega(l1,l2 : Real);
Procedure Prec(var x,y : Vector; k :  integer);
Procedure Ax(var x,y : Vector);
implementation
var
    om  :  array[0..k] of Real;

Procedure Ax(var x,y  :  Vector);
begin
 (Implementation of y = Ax )
end;
Procedure SetOmega(l1,l2 : Real);
var i  :  integer;
begin
 for i := 0 to k do
  begin
   om[i] := 1/(l1 + l2);
   l1 := l1 * (1 − om[i] * l1);
   l2 := 1/(4 * om[i]);
   end;
  end;
Procedure M(var x,y  :  Vector; k :  integer); Forward;
Procedure A(varx,y  :  Vector; k  :  integer);
var x1  :  Vector;
begin
 if k  =  0
  then Ax(x,y)
  else
   begin
    M(x,x1,k − 1);
    A(x1,y,k − 1)
   end;
end;
Procedure M(var x,y  :  Vector; k  :  integer);
var
   x1  :  Vector;
   i  :  integer;
```

```
begin
 A(x,x1,k);
 for i := 1 to n do
  y[i] := x[i] - om[k] * x1[i];
end;
Procedure Prec(var x,y  :  Vector;  k  :  integer);
var x1  :  Vector;
begin
 if k = 0
  then y := x
  else
   begin
    M(x,x1,k - 1);
    Prec(x1,y,k - 1)
   end;
end;
end.
```

References

[1] *J. Mandel*: Adaptive Iterative Solvers in Finite Elements. To appear.

[2] *J. Mandel*: Balancing Domain Decomposition. Communications in Numerical Methods in Engineering *9* (1993).

[3] *G. Luenberger*: Introduction to Linear and Nonlinear Programming. Addison-Wesley, New York, 1973.

[4] *A.van der Sluis and H.A.van der Vorst*: The Rate of Convergence of Conjugate Gradients. Numer. Math. *48* (1986).

[5] *W. Hackbusch*: Multigrid Methods and Applications. Springer Verlag, 1985.

[6] *O. Axelsson, V. A. Barker*: Finite Element Solution of Boundary Value Problems. Academic Press, 1984.

Author's address: *Jitka Křížková, Petr Vaněk*, Západočeská universita, Americká 42, 306 14 Plzeň, ČR.