Optim Lett (2013) 7:1047–1059
DOI 10.1007/s11590-012-0491-7

ORIGINAL PAPER

An asymptotically optimal gradient algorithm for quadratic optimization with low computational cost

Anatoly Zhigljavsky · Luc Pronzato · Elena Bukina

Received: 5 October 2011 / Accepted: 27 April 2012 / Published online: 17 May 2012

Abstract We consider gradient algorithms for minimizing a quadratic function in $\mathbb{R}^n$ with large n. We suggest a particular sequence of step-sizes and demonstrate that the resulting gradient algorithm has a convergence rate comparable with that of Conjugate Gradients and other methods based on the use of Krylov spaces. When the matrix is large and sparse, the proposed algorithm can be more efficient than the Conjugate Gradient algorithm in terms of computational cost, as k iterations of the proposed algorithm only require the computation of $O(\log k)$ inner products.

Keywords Quadratic optimization · Gradient algorithms · Conjugate gradient · Arcsine distribution · Fibonacci numbers · Estimation of leading eigenvalues

The work of E. Bukina was partially supported by the EU through a Marie-Curie Fellowship (EST-SIGNAL program: http://est-signal.i3s.unice.fr) under the contract Nb. MEST-CT-2005-021175. Part of this work was accomplished while the first two authors were invited at the Isaac Newton Institute for Mathematical Sciences, Cambridge, UK; the support of the INI and of CNRS is gratefully acknowledged.

A. Zhigljavsky (✉)
School of Mathematics, Cardiff University, Cardiff CF24 4YH, UK
e-mail: ZhigljavskyAA@cardiff.ac.uk

L. Pronzato · E. Bukina
Laboratoire I3S, CNRS-UNS, 2000 route des Lucioles, B.P. 121, 06903 Sophia Antipolis, France
e-mail: pronzato@i3s.unice.fr

E. Bukina
e-mail: bukina@i3s.unice.fr

1 Introduction

Consider the problem of minimizing a quadratic function $f(\cdot)$ defined on $\mathbb{R}^n$ by

$$f(x) = \tfrac{1}{2}(Ax, x) - (x, b), \tag{1}$$

where $(\cdot, \cdot)$ denotes the inner product. We assume that A is a symmetric positive-definite matrix. Let x^* be the solution of the minimization problem of (1), $m = \lambda_1 \leq \cdots \leq \lambda_n = M$ be the eigenvalues of A (with $0 < m < M < \infty$) and $\{q_1, \ldots, q_n\}$ be the set of corresponding orthonormal eigenvectors. We do not assume that any information is available about the eigenvalues λ_i and eigenvectors q_i, $i = 1, \ldots, n$, and suppose that the condition number $\rho = M/m$ may be large.

Consider a general gradient algorithm with iterations of the form $x_{k+1} = x_k - \gamma_k g_k$, $(k = 0, 1, \ldots)$, where $x_0 \in \mathbb{R}^n$ is a starting point, $\gamma_k > 0$, the step-size at iteration k, is determined by some rule and $g_k = \nabla f(x_k) = Ax_k - b$ is the gradient of the objective function $f(\cdot)$ at point x_k. These iterations can be rewritten in terms of the gradients as

$$g_{k+1} = g_k - \gamma_k A g_k. \tag{2}$$

It is sometimes convenient to rewrite the basic iteration (2) as $g_k = P_k(A)g_0$, where P_k denotes the polynomial $P_k(\lambda) = (1 - \gamma_{k-1}\lambda)(1 - \gamma_{k-2}\lambda)\ldots(1 - \gamma_0\lambda)$.

We are interested in situations where the dimension n can be very large; it could even be infinite, with A a self-adjoint operator in a Hilbert space, but we shall restrict the presentation to the finite dimensional situation. We assume that n is much larger than the number of iterations k_* needed to achieve the precision required; formally, $k_* = o(n)$ with $k_* \to \infty$ in asymptotic considerations.

We shall define the rate of convergence at iteration j by $r_j = (g_{j+1}, g_{j+1})/(g_j, g_j)$. The rate of convergence of the algorithm (2) after k iterations is then $R_k = \prod_{j=0}^{k-1} r_j = (g_k, g_k)/(g_0, g_0)$ and the asymptotic rate can naturally be defined as $R = \lim_{k\to\infty} R_k^{1/k}$, if the limit exists. It is easy to see that R does not depend on g_0 as long as g_0 is not contained in the subspace spanned by $n' < n$ eigenvectors of A. Other rates of convergence towards x^*, which are asymptotically equivalent to r_k, could be considered as well, see Th. 6 in [7]. In particular, the rate $r'_k = (f(x_{k+1}) - f(x^*))/(f(x_k) - f(x^*))$ is of great interest in optimization theory.

The method of Steepest-Descent (SD) chooses γ_k in (2) that minimizes r'_k; direct calculation gives $\gamma_k = \gamma_k^{SD} = (g_k, g_k)/(Ag_k, g_k)$. The method of Minimum Residues [5] (shortly, MR) chooses γ_k that minimizes r_k, which gives $\gamma_k = \gamma_k^{MR} = (Ag_k, g_k)/(A^2 g_k, g_k)$. Both methods only look one-step forward. The method of Conjugate Gradients (CG in what follows) minimizes $R'_k = \prod_{j=0}^{k-1} r'_j$ with respect to the sequence $\gamma_0, \gamma_1, \ldots, \gamma_{k-1}$ and the method of Conjugate Residuals [3, p.547] (CR) does the same with R_k; we shall denote $R_k^{CR} = \min_{\gamma_0,\ldots,\gamma_{k-1}} R_k$. The construction of CG and CR does not require the direct minimization of R'_k or R_k with respect to the step-sizes γ_j, but an implementation of the Gram-Schmidt orthogonalization process leads to the construction of the optimal polynomial $P_k(\lambda)$ which minimizes either R'_k

or R_k. In this paper, we shall use the convergence rate R_k as measure of performance and therefore CR (rather than CG) will be our competitor.

Using the expression (5) below for the rate R_k, we have $R_k \le \max_{\lambda\in[m,M]} Q_k^2(\lambda)$, where $Q_k(\lambda)$ is a polynomial of degree k satisfying $Q_k(0) = 1$ (the set of polynomials $Q_k(\lambda)$ includes the polynomials $P_k(\lambda)$ introduced above). The right-hand side of the last inequality is minimized for $Q_k = P_k^*$, where

$$P_k^*(\lambda) = c_k\, T_k\left(\frac{2\lambda - (M+m)}{M-m}\right), \quad c_k = T_k^{-1}\left(\frac{\rho+1}{\rho-1}\right) \tag{3}$$

and $T_k(\cdot)$ is the k-th Chebyshev polynomial of the first kind; for $t \in [-1, 1]$ these polynomials are defined by $T_k(t) = \cos[k \arccos(t)]$. Hence, the rate R_k^{CR} of the CR algorithm satisfies $R_k^{CR} \le R_k^*$, where

$$R_k^* = c_k^2 = \frac{1}{4}(R_\infty^{k/2} + R_\infty^{-k/2})^{-2} \quad \text{and} \quad R_\infty = \lim_{k\to\infty} (R_k^*)^{1/k} = \left(\frac{\sqrt{\rho}-1}{\sqrt{\rho}+1}\right)^2. \tag{4}$$

There are no iterative methods of the form (2) that achieve smaller values of the rate R_k than R_k^{CR}, the rate of CR within k iterations. Therefore, our competition with CR cannot be in terms of R_k only and, in addition to the rate of convergence, we shall consider the complexity of the methods, which will be measured by the number of inner products computed after k iterations. The purpose of the paper is to present a method that requires less computations than CR and at the same time achieves asymptotically the same rate as the worst-case convergence rate of CR when stopped at some iteration $k < n$. As will be seen in (18), for any $k > 1$ the proposed method requires no more than $C \log k + 4$ computations of inner products ($C \simeq 8.31$) in k iterations which, for large k, is much smaller than the $2k$ computations of inner products required by CR and CG. This makes our method competitive when the matrix A in (1) is sparse and inner products yield the main contribution to the computational cost. For instance, in parallel computing the computation of inner products requires a global communication between processors, the transfer time between these processors increases as their number grows, whereas matrix by vector multiplications only require local communication between neighboring processors, see [12, Sect. 4.4].

The paper is organized as follows. In Sect. 2 we introduce some mathematical formalism and formulate several auxiliary statements needed to explain the construction and properties of the proposed method. We also explain how $\max_{x_0,A}(R_k)^{1/k}$ can be made arbitrary close to its lower bound by using step-sizes γ_k that are suitably distributed. In Sect. 3 we construct an auxiliary sequence $\{z_j\}$ which is one of the essential ingredients for the construction of the main algorithm in Sect. 4. In Sect. 3 and 4 we also explain the algorithm and validate its properties. Numerical examples are discussed in Sect. 5.

2 Preliminary results

Decompose the initial vector g_0 in the basis of eigenvectors $\{q_1, \dots, q_n\}$: $g_0 = \sum_{i=1}^n \alpha_i q_i$.Then for all $k \ge 1$ we have the decompositions$g_k = \sum_{i=1}^n \alpha_i P_k(\lambda_i) q_i$.

The squared L_2-norm of g_0 is $\|g_0\|^2 = (g_0, g_0) = \sum_{i=1}^n \alpha_i^2$ and the squared L_2-norm of g_k is$\|g_k\|^2 = (g_k, g_k) = \sum_{i=1}^n \alpha_i^2 P_k^2(\lambda_i)$. Using these representations we can write the rate R_k as

$$R_k = \frac{(g_k, g_k)}{(g_0, g_0)} = \left[\sum_{i=1}^n \alpha_i^2 P_k^2(\lambda_i)\right] \Big/ \sum_{i=1}^n \alpha_i^2 = \sum_{i=1}^n p_i^{(0)} P_k^2(\lambda_i), \tag{5}$$

where $p_i^{(0)} = \alpha_i^2 \big/ \sum_{j=1}^n \alpha_j^2 \geq 0$ and $\sum_{i=1}^n p_i^{(0)} = 1$. Without loss of generality all α_i^2 can be assumed to be strictly positive. Indeed, if $\alpha_i = 0$ for some i then the matrix $A = \sum_{j=1}^n \lambda_j q_j q_j^T$ can be replaced with $\tilde{A} = \sum_{j \neq i} \lambda_j q_j q_j^T$. The equality $\alpha_i = 0$ would mean that the algebraic (i.e. signed) length of the projection of $\lambda_i\, x_0$ (and therefore of all $\lambda_i\, x_k$, $k \geq 0$) to the vector q_i is equal to (b, q_i); that is, the algebraic length of the projection of b to q_i.

The algebraic length of the projection of the vector g_k on the eigenvector q_i is $(g_k, q_i) = \alpha_i P_k(\lambda_i)$. We define $p_i^{(k)} = (g_k, q_i)^2/(g_k, g_k) = \alpha_i^2 P_k^2(\lambda_i)/\sum_{j=1}^n \alpha_j^2 P_k^2 (\lambda_j)$ and interpret this as a point mass at λ_i. Then, the measure ν_k defined by its masses $\nu_k(\lambda_i) = p_i^{(k)}$ at $\lambda = \lambda_i (i = 1, \ldots, n)$ characterizes the normalized gradient $g_k/\|g_k\|$ (up to the signs of (g_k, q_i) which are irrelevant for the rate of convergence of the algorithm).

For any real α, define $\mu_\alpha^{(k)}$ as the α-th moment of the probability measure ν_k : $\mu_\alpha^{(k)} = \sum_{i=1}^n \lambda_i^\alpha p_i^{(k)} = (A^\alpha g_k, g_k)/(g_k, g_k)$. Using the main iteration (2), we obtain the following updating formula for the measure ν_{k+1}:

$$\begin{aligned} p_i^{(k+1)} = \nu_{k+1}(\lambda_i) &= \frac{\alpha_i^2 P_{k+1}^2(\lambda_i)}{(g_{k+1}, g_{k+1})} = \frac{\alpha_i^2 (1 - \gamma_k \lambda_i)^2 P_k^2(\lambda_i)}{(g_{k+1}, g_{k+1})} \\ &= \frac{(1 - \gamma_k \lambda_i)^2 p_i^{(k)}}{r_k}, \end{aligned} \tag{6}$$

where

$$\begin{aligned} r_k = \frac{(g_{k+1}, g_{k+1})}{(g_k, g_k)} &= \frac{(g_k, g_k) - 2\gamma_k (A g_k, g_k) + \gamma_k^2 (A^2 g_k, g_k)}{(g_k, g_k)} \\ &= 1 - 2\gamma_k \mu_1^{(k)} + \gamma_k^2 \mu_2^{(k)}. \end{aligned} \tag{7}$$

The following two results will be useful for the construction of the main algorithm and for the evaluation of its performance.

Proposition 1 (Monotonicity) *Let $\beta_k = 1/\gamma_k$. The condition $r_k < 1$ is equivalent to*

$$\beta_k > \mu_2^{(k)}/(2\mu_1^{(k)}). \tag{8}$$

Proof Follows from (7). □

Proposition 2 (Moment inequalities) *Let ν be any probability measure on $[m, M]$ and denote by μ_α the α-th moment of ν: $\mu_\alpha = \mu_\alpha(\nu) = \int_m^M t^\alpha \, \nu(dt)$. Then*

$$m \le \mu_1 \le \frac{\mu_2}{\mu_1} \le \frac{\mu_3}{\mu_2} \le \frac{\mu_4}{\mu_3} \le \cdots \le M. \tag{9}$$

Proof The Cauchy–Schwarz inequality implies $\mu_{\alpha+2}\mu_\alpha \ge (\mu_{\alpha+1})^2$ for any α. Moreover, for all $t \in [m, M]$ we have $t(M - t) \ge 0$ so that $\int_m^M t^\alpha (M - t)\, \nu(dt) = M\mu_\alpha - \mu_{\alpha+1} \ge 0$; that is, $\mu_{\alpha+1}/\mu_\alpha \le M$. Similarly, $m \le \mu_{\alpha+1}/\mu_\alpha$. This implies (9). □

For any choice of the step-size sequence $\{\gamma_k\}$ in (2) the rate R_k satisfies $\max_{x_0, A} R_k \ge R_k^*$ for $k < n$, with $(R_k^*)^{1/k}$ decreasing to R_∞ as $k \to \infty$, see (4). Our objective is to construct simple gradient algorithms such that $R_k^{1/k}$ gets close to the bound R_∞ for large k. If the values m and M were known, the results of [9] show that this can easily be achieved using some predefined sequences of step-sizes γ_k, without any need for computation of inner products. For example, one can use $\beta_j = 1/\gamma_j$ as the roots of the polynomials (3), but the degree of the polynomial must then be pre-specified (and be equal to the number of iterations to be performed); the ordering of the roots is important too.

One of the main points in our argumentation uses the fact that by a suitable choice of the coefficients $\gamma_0, \gamma_1, \ldots$ we can easily force the asymptotic relation $\nu_k(\lambda) \to 0$ as $k \to \infty$ for all $\lambda \in (m, M)$. That is, for large k we can force the projections of the normalized gradient $g_k/\|g_k\|$ to all eigenvectors except for q_1 and q_n to be negligible. This happens, for example, if the sequence $\{\beta_k\}$ is concentrated in an interval $[m', M']$ with $m < m' < M' < M$ and the values of $\{\beta_k\}$ are well-spread over $[m', M']$. More precisely, the following result is a particular case of Theorem 1 in [9].

Proposition 3 (Convergence to the set of measures supported at two points) *Assume that $\beta_k \in [m', M']$ with $m < m' \le M' < M$ for all k and that the sequence $\{\beta_k\}$ has an asymptotic distribution function $F(t)$ such that $\int \log(t - \lambda)^2 \, dF(t) < \max\{\int \log(M - t)^2 \, dF(t), \int \log(t - m)^2 \, dF(t)\}$ for all $\lambda \in (m, M)$. Then the algorithm (2) associated with the sequence $\{\beta_k\}$ is such that $\lim_{k\to\infty} \nu_k(\lambda_i) = 0$ for all $i = 2, \ldots, n - 1$.*

The following statement is a simple consequence of Proposition 3.

Proposition 4 (Attraction to one-point measures) *Under the conditions of Proposition 3, the sequence of measures ν_k attracts to the set of two-point measures supported on $\{m, M\}$. If, moreover, the asymptotic distribution of the sequence $\{\beta_k\}$ is biased towards m (respectively M), then the sequence $\{\nu_k\}$ tends to the one-point measure concentrated at M (respectively m). This happens in particular when the sequence $\{\beta_k\}$ is generated by pairs as in Proposition 5 with $\beta_{2j+1} = M' + m' - \beta_{2j}$ for all $j \ge j_0$, $m' = m + \varepsilon_m$, $M' = M - \varepsilon_M$ and $\varepsilon_m \ne \varepsilon_M$.*

Once the attraction of the sequence $\{\nu_k\}$ to the set of measures supported at the two-point set $\{m, M\}$ is obtained, it is enough to consider the rate of convergence r_k obtained for the two-point measures and the following is true, see [9, Th. 2].

Proposition 5 (Asymptotic rate) *Assume that the conditions of Proposition 3 are satisfied and the values $\beta_k = 1/\gamma_k$ are generated by symmetric pairs for large k; that is, $\beta_{2j+1} = M + m - \beta_{2j}$ for all $j \geq j_0$. Then the asymptotic rate R is* $R = \exp\{2 \int [\log (t - m) - \log t]\, dF(t)\}$.

If the limiting distribution function $F(\cdot)$ has the arcsine density on $[m', M']$

$$p(t) = \frac{1}{\pi \sqrt{(t - m')(M' - t)}}, \quad m' \leq t \leq M', \tag{10}$$

with $m' = m + \varepsilon$ and $M' = M - \varepsilon$ for some small $\varepsilon \geq 0$, then, as shown in [9], the asymptotic rate of convergence is $R_\varepsilon = R_\infty \left(1 + 4\sqrt{\varepsilon(M - m)}\right) + O(\varepsilon)$, $\varepsilon \to 0$.

Numerical results in [9] confirm the observation that if the asymptotic distribution of the sequence $\{\beta_k\}$ is close to the distribution with the arcsine density on $[m, M]$, then the asymptotic rate of the algorithm (2) is close to R_∞. Furthermore, we observed that for the Barzilai–Borwein (BB) [1] as well as many other gradient algorithms we have investigated (see, e.g., [4]), when the rate $R_k^{1/k}$ approached R_∞ then the distribution of the sequence $\{\beta_k\}$ was close to the distribution with the arcsine density on $[m, M]$. This phenomenon can be related to the fact that, roughly speaking, for large k the roots of all orthogonal polynomials behave similarly. In particular, if a probability measure ν_0 has a density $p_0(\cdot)$ with respect to the Lebesgue measure on $[m, M]$, with $p_0(t) > 0$ except for a set of zero Lebesgue measure, then the roots of the orthogonal polynomials P_k^* with respect to ν_0 are distributed asymptotically according to the arcsine density on $[m, M]$, see [2, Th. XIII]. As any efficient gradient algorithm should not deviate much from the behaviour of the optimal algorithm (which locates β_k at the roots of the polynomials P_k^*), the asymptotic distribution of $\{\beta_k\}$ for this algorithm should be close to the arcsine distribution on $[m, M]$.

3 Numerical sequence for generating step-sizes

The method presented below generates step-sizes γ_k such that the inverse sequence $\{\beta_k\} = \{1/\gamma_k\}$ has the arcsine distribution on a given interval $[m', M']$, where m' and M' can be chosen as $m' = m + \varepsilon$ and $M' = M - \varepsilon$, with small $\varepsilon \geq 0$. In practice m and M are unknown and we shall use estimates $m' = \widehat{m}_k$ and $M' = \widehat{M}_k$ of m and M, with $m < \widehat{m}_k < \widehat{M}_k < M$.

We shall initiate the algorithm (2) by using two iterations of MR, that is with $\beta_0 = \mu_2^{(0)}/\mu_1^{(0)}$ and $\beta_1 = \mu_2^{(1)}/\mu_1^{(1)}$; this provides us with $\widehat{m}_2 = \min\{\beta_0, \beta_1\}$ and $\widehat{M}_2 = \max\{\beta_0, \beta_1\}$, the initial estimates of m and M.

The following iterations will use the sequence $\{z_j\}$ in $[0, 1]$ introduced below: to compute β_k we rescale the values of z_j to $[m', M']$ through

$$\beta_k = m' + (M' - m')z_j, \quad \text{for } k = 2, 3, \ldots \quad \text{and} \quad \text{some } j = j(k). \tag{11}$$

Note that if m' and M' are fixed, $\{z_j\}$ has the asymptotic density

$$q(z) = 1/[\pi \sqrt{z(1 - z)}], \quad 0 \leq z \leq 1, \tag{12}$$

$j \le k$ and $j/k \to 1$ as $k \to \infty$, then the sequence $\{\beta_k\}$ constructed by (11) has the asymptotic density (10).

A discussion concerning the choice of $\{z_j\}$ with density (12) is contained in [9]. We have experimented with a number of sequences and provide below one of the best if not the best.

The sequence $\{\mathbf{z_j}\}$. *For all $j \ge 0$, we define $v_j = \{\varphi(j+1)\}$, where $\varphi = (\sqrt{5}+1)/2 \simeq 1.61803\ldots$ and $\{a\}$ denotes the fractional part of a. For $j = 0, 1, \ldots$, set*

$$z_j = (1+\cos(\pi u_j))/2, \quad \text{where } u_{2j} = \min\{v_j, 1-v_j\}, u_{2j+1} = \max\{v_j, 1-v_j\}. \tag{13}$$

All three sequences above, $\{v_j\}$, $\{u_j\}$ and $\{z_j\}$, belong to [0, 1]. The sequence $\{v_j\}$ consists of the fractional parts of multiples of the golden ratio φ, the sequence $\{u_j\}$ is a symmetrized version of $\{v_j\}$. The asymptotic distribution of $\{v_j\}$ and $\{u_j\}$ is uniform while the sequence $\{z_j\}$ has the required arcsine density (12). The reasons behind the proposed choice of the sequence $\{z_j\}$ are the following:

- simplicity of construction and analysis;
- required asymptotic distribution (with empirical distribution converging fast to the limiting distribution — a consequence of the good uniformity of $\{v_j\}$);
- symmetry $z_{2j+1} = 1 - z_{2j}$ and $z_{2j} > \frac{1}{2}$ for all j.

A benefit of using symmetric points is that by choosing the largest z_j (and therefore the largest β_k) first in each symmetric pair one makes the behavior of the proposed algorithm more monotonic. Indeed, $\beta_k \ge (M+m)/2$ implies $\beta_k > M/2 \ge \mu_2^{(k)}/(2\mu_1^{(k)})$, see (9), and therefore $r_k < 1$, see (8). Moreover, (6) indicates that when $\beta_k = 1/\gamma_k$ is large, the measure ν_{k+1} allocates more weight than ν_k to small eigenvalues λ_i, so that $\mu_2^{(k+1)}/(2\mu_1^{(k+1)})$ tends to be small and the next iteration is likely to be monotonic too according to (8).

For the sequence $z_0, z_1, z_2, \ldots$ constructed above we define the sequences of record moments $L_{\min} = \{L_{\min}(j)\}_{j=0}^{\infty}$ and $L_{\max} = \{L_{\max}(j)\}_{j=0}^{\infty}$ as follows: $L_{\min}(0) = L_{\max}(0) = 0$ and $L_{\min}(j+1) = \min\{k > L_{\min}(j) : z_k < z_{L_{\min}(j)}\}$, $L_{\max}(j+1) = \min\{k > L_{\max}(j) : z_k > z_{L_{\max}(j)}\}$ for $j \ge 0$. These two sequences of record moments are $L_{\min} = \{0, 1, 3, 5, 9, 15, \ldots\}$ and $L_{\max} = \{0, 2, 4, 8, 14, \ldots\}$ with $L_{\min}(j+1) = L_{\max}(j)+1$ for $j = 0, 1, \ldots$

The following statement expresses the sequences $L_{\min}$ and $L_{\max}$ in terms of the Fibonacci numbers. Recall the sequence of Fibonacci numbers $\{F_N\}_{N=1}^{\infty} = \{1, 1, 2, 3, 5, 8, 13, 21, 34\ldots\}$ and the exact formula $F_N = (\varphi^N - \widehat{\varphi}^N)/\sqrt{5}$, where φ is the golden ratio and $\widehat{\varphi} = -1/\varphi$.

Proposition 6 *We have $L_{\max}(j) = 2(F_{j+2} - 1)$ for $j = 0, 1, \ldots$ and $L_{\min}(j) = 2F_{j+1} - 1$ for $j = 1, 2, \ldots$ with $L_{\min}(0) = 0$, where F_i is the i-th Fibonacci number.*

Proof The statement is a direct consequence of the following two classical results of the theory of Diophantine approximations: (i) for the sequence $\{k\alpha\}$ with any irrational α, the successive minimal and maximal values occur when $k = q$ in the denominator of a convergent p/q for α in the standard continued fraction expansion of α, see [11]; (ii) the convergents of $\alpha = \varphi - 1$ are F_j/F_{j+1} for $j > 1$. □

Proposition 7 *Let $\{z_j\}$ be the sequence constructed in (13) and δ_j be the number of upper record moments appearing among $z_0, z_1, \dots, z_j$; that is,*

$$\delta_j = \{\textit{number of } i \geq 0 \textit{ s.t. } L_{\max}(i) \leq j\} = 1 + \max\{i \geq 0 : \textit{s.t. } L_{\max}(i) \leq j\}. \tag{14}$$

Then for all $j > 1$ we have the inequalities $c \log j - 2 < \delta_j < c \log j + 1$ and therefore the asymptotic relation $\delta_j = c \log j + O(1)$ as $j \to \infty$, where $c = 1/\log(\varphi) \simeq 2.078$.

Proof Denote $\vartheta = \sqrt{5}/(2\varphi^2) = (3\sqrt{5}-5)/4 \simeq 0.427$. We have $L_{\max}(i) = 2(F_{i+2} - 1)$ for $i = 0, 1, \dots$ and therefore for all $j > 1$ we obtain

$$\begin{aligned}\delta_j - 1 &= \max\{i : 2(F_{i+2} - 1) \leq j\} = \max\{i : F_{i+2} \leq j/2 + 1\}\\ &= \max\{i : \varphi^{i+2} - \widehat{\varphi}^{i+2} \leq \sqrt{5}(j/2 + 1)\}\\ &= \max\{i : \varphi^i \leq \sqrt{5}(j/2 + 1)/\varphi^2 + \widehat{\varphi}^{i+4}\}.\end{aligned}$$

This yields $C_\gamma < \delta_j - 1 < C_1$, where $\gamma = \sqrt{5}/\phi^2 - 1/\phi^5 = 3 - \sqrt{5} \simeq 0.764$ and $C_\delta = \max\{i : \varphi^i \leq \vartheta j + \delta\} = \max\{i : i \leq c \log(\vartheta j + \delta)\}$ for $\delta > 0$. Consequently, for all $j > 1$ we obtain

$$\begin{aligned}\delta_j - 1 < C_1 \leq c \log(\vartheta j + 1) &= c\left[\log j + \log(\vartheta + 1/j)\right] < c \log j,\\ \delta_j > C_\gamma + 1 \geq c \log(\vartheta j + \gamma) &> c\left[\log j + \log \vartheta\right] > c \log j - 2.\end{aligned}$$

□

4 The algorithm

Since we assume that the smallest and largest eigenvalues $\lambda_1 = m$ and $\lambda_n = M$ of the matrix A are unknown, these values have to be estimated. Our estimators will be based on the inequalities (9) and we shall use $\widehat{m}_{k+1} = \min\{\mu_1^{(i)}, i \in I_k\}$ and $\widehat{M}_{k+1} = \max\{\mu_{\alpha+1}^{(i)}/\mu_\alpha^{(i)}, i \in I_k\}$ where in the algorithm below $\alpha = 3$ and I_k is a subset of $\{0, 1, \dots, k\}$ which will be determined by the algorithm. The case $\alpha = 0$ has been considered in [9]. The resulting algorithms are efficient but not efficient enough to compete with CR and CG. The value $\alpha = 3$ leads to much better estimates of M and hence to much more cost-efficient algorithms which use very thin sets I_k^m and I_k^M and as a consequence few inner products to compute. Notice that from Proposition 2 we have $m \leq \widehat{m}_k$ and $\widehat{M}_k \leq M$ for all k, as required by Proposition 3.

Using good estimators of M is important for the following reason. Generating the $\beta_k = 1/\gamma_k$ by pairs and using the largest first in each pair favors the estimation of m against that of M. Indeed, the measure ν_k attracts to the set of two-point measures supported at $\{m, M\}$. For a measure ν_{2i} in this set, two iterations of (6) with $\beta_{2i+1} = M + m - \beta_{2i}$ give $\nu_{2i+2} = \nu_{2i}$ and thus $\mu_\alpha^{(2i+2)} = \mu_\alpha^{(2i)}$ for all α. This means that asymptotically the estimation of m and M can be improved for even values of j only, where j is the index of the sequence $\{z_j\}$. Using the largest β first in a pair yields

$\beta_{2i} > (M+m)/2$ resulting in a small $\mu_1^{(2i+1)}$ that may improve the current estimation of m when β_{2i} is close to M. In view of Proposition 4, this better estimation of m than M results in the concentration of ν_k at M. Since in this case the ratio $\mu_2^{(k)}/(2\mu_1^{(k)})$ becomes close to $M/2$ when M/m is large, the monotonicity condition (8) would be violated frequently. In the algorithm proposed below we avoid this by using a good estimator of M.

Monotonicity can be imposed by forcing the measures ν_k to be concentrated more at m rather than at M when needed. Indeed, the condition (8) is always satisfied when $\mu_2^{(k)}/(2\mu_1^{(k)})$ is close to $m/2$ since β_k is larger than $\widehat{m}_k > m$. This movement of the masses is achieved by using a step with large β_k when ν_{k-1} is close to the delta measure at M. In practice, we simply use $\beta_k = \widehat{M}_k$ when we observe $\widehat{M}_k > \widehat{M}_{k-1}$.

The calculation of $\mu_1^{(k)}$ needed for the computation of $\widehat{m}_{k+1}$ does not require the calculation of Ag_k at step k. Indeed, allowing a delay of one step in the estimation of m, (2) gives $Ag_k = \beta_k(g_k - g_{k+1})$, so that

$$\mu_1^{(k)} = \frac{(Ag_k, g_k)}{(g_k, g_k)} = \beta_k \left[1 - \frac{(g_k, g_{k+1})}{(g_k, g_k)}\right]. \tag{15}$$

Allowing a delay of two iterations, we get $A^2 g_{k-1} = \beta_{k-1}\beta_k(g_{k+1}-g_k)+\beta_{k-1}^2(g_{k-1}-g_k)$ and $Ag_{k-1} = \beta_{k-1}(g_{k-1} - g_k)$ when using (2). This gives

$$\begin{aligned}\frac{\mu_4^{(k-1)}}{\mu_3^{(k-1)}} &= \frac{(A^2 g_{k-1}, A^2 g_{k-1})}{(A^2 g_{k-1}, Ag_{k-1})}\\ &= \beta_{k-1} + \beta_k \frac{(\beta_k(g_{k+1}-g_k) + \beta_{k-1}(g_{k-1}-g_k), g_{k+1}-g_k)}{(\beta_k(g_{k+1}-g_k) + \beta_{k-1}(g_{k-1}-g_k), g_{k-1}-g_k)}. \end{aligned}\tag{16}$$

This avoids the use of additional matrix-vector multiplications while computing the estimates of m and M. The calculations of inner products required for the update of the estimators of m and M can be done in parallel. Moreover, these estimators need to be updated at iteration k only when the value z used at previous iteration was larger than the maximum over all previous z_i. The updating rule is then

$$\widehat{m}_{k+1} = \min\{\widehat{m}_k, \mu_1^{(k)}\} \quad \text{and} \quad \widehat{M}_{k+1} = \max\{\widehat{M}_k, \mu_4^{(k-1)}/\mu_3^{(k-1)}\} \tag{17}$$

where $\mu_1^{(k)}$ and $\mu_4^{(k-1)}/\mu_3^{(k-1)}$ are computed by (15) and (16) respectively.

We are now ready to formulate the main algorithm; its MATLAB implementation is available at http://www.i3s.unice.fr/~pronzato/Matlab/goldenArcsineQ.m

The Algorithm.

Stage I (initialization)

I.1 Choose x_0 and compute $g_0 = Ax_0 - b$.
I.2 Choose $\epsilon > 0$ used in the stopping rule.
I.3 Set $L_{\max} = \{2F_{i+2} - 2 : i = 0, 1, \ldots\} = \{0, 2, 4, 8, 14, \ldots\}$.
I.4 For $k = 0$ and 1, set $x_{k+1} = x_k - (1/\beta_k)g_k$ and $g_{k+1} = Ax_{k+1} - b$, where $\beta_k = (Ag_k, Ag_k)/(Ag_k, g_k)$.

```
I.5 Set $\widehat{m}_2 = \min\{\beta_0, \beta_1\}$ and $\widehat{M}_1 = \widehat{M}_2 = \max\{\beta_0, \beta_1\}$.
I.6 Set $k = 2$ and $j = 0$.
```

Stage II (iterations)

```
II.1 If $\widehat{M}_k > \widehat{M}_{k-1}$ then set $\beta_k = \widehat{M}_k$. Otherwise set $\beta_k = \widehat{m}_k + (\widehat{M}_k - \widehat{m}_k)z_j$
     and $j \leftarrow j + 1$.
II.2 Set $x_{k+1} = x_k - (1/\beta_k)g_k$ and $g_{k+1} = Ax_{k+1} - b$.
II.3 If $j-2 \in L_{\max}$ then compute $\widehat{m}_{k+1}$ and $\widehat{M}_{k+1}$ using formula (17) and check
     the stopping rule $(g_k, g_k) \le \epsilon$. Otherwise set $\widehat{m}_{k+1} = \widehat{m}_k$, $\widehat{M}_{k+1} = \widehat{M}_k$.
II.4 Set $k \leftarrow k + 1$ and return to Step II.1.
```

The algorithm uses the common stopping rule $(g_k, g_k) < \epsilon$ for some given ϵ. The values of (g_k, g_k) are available at the iterations k such that $j - 2 \in L_{\max}$. At these iterations, we can check the stopping rule directly. In addition, as we have an (updated) under-estimate for ρ and hence and under-estimate of R_∞ given by (4), we can estimate the number of iterations still remaining to achieve the required precision.

The proposed algorithm only requires one matrix-vector multiplication per iteration (used to calculate the gradient $g_k = Ax_k - b$), like other gradient methods, and Krylov-space based algorithms, like CR and CG. If A is sparse then matrix-vector multiplications can be cheap and therefore the computation of inner products will significantly contribute to the total computational cost. When using parallel computing, it may even yield the main contribution to the computational cost, see [12, Sect. 4.4].

The standard formulation of CG (and also CR) requires the computation of two inner products per iteration. In some sophisticated versions of CG, these two inner products can be computed in parallel, at the possible cost of a slight increase of storage and maybe reduced numerical stability, see for instance [6,10].

The following theorem establishes the upper bound and the asymptotical expression for the total number of inner products computed by the proposed algorithm.

Proposition 8 *Let N_k be the total number of inner products computed within $k+1$ steps of the proposed algorithm. Then for all $k > 0$ we have*

$$\begin{aligned} &N_k < 4 + 4\log k/\log(\varphi) \simeq 4 + 8.31\log k \;\; \textit{and} \\ &N_k = 4\log k/\log(\varphi) + O(1) \;\; \textit{as } k \to \infty. \end{aligned} \tag{18}$$

Proof The proposed algorithm requires the computation of four inner products in the initial two iterations and four inner products each time the estimates $\widehat{m}_k$ and $\widehat{M}_k$ are updated. This is done when $j - 2 \in L_{\max}$. Therefore, the total number of inner products computed within $k+1$ steps of the proposed algorithm is equal to $N_k = 4 + 4\delta_j$ where $j = j(k)$ is defined by the algorithm and δ_j is defined in (14). Proposition 7 implies that $j(k) < k$, $\delta_j < \log j/\log(\varphi)$ for all $j > 1$ and $\delta_j = \log j/\log(\varphi) + O(1)$ as $j \to \infty$. This and the fact $k/j(k) \to 1$ as $k \to \infty$ imply (18). □

5 Numerical performance and comparison with other methods

In order to illustrate the efficiency of the proposed algorithm we consider two test problems. In both problems, we set $m = 1$, $M = 1{,}000$, $n = 1{,}000$, $b = Ac$ with c

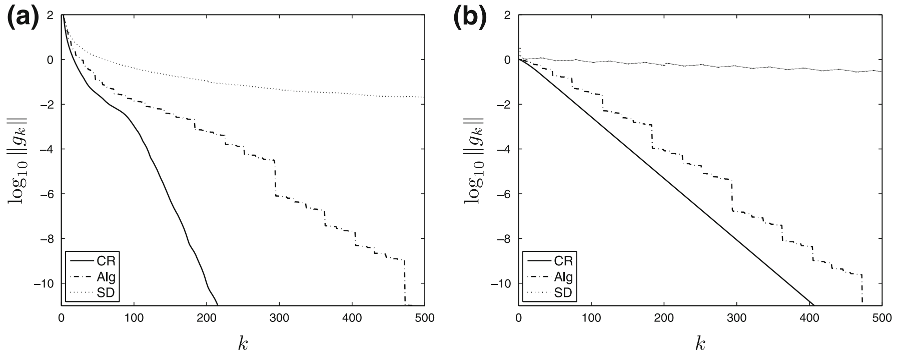

Fig. 1 $\text{Log}_{10}\|g_k\|$ against the number of iterations k for Problem 1 (*left*) and Problem 2 (*right*)

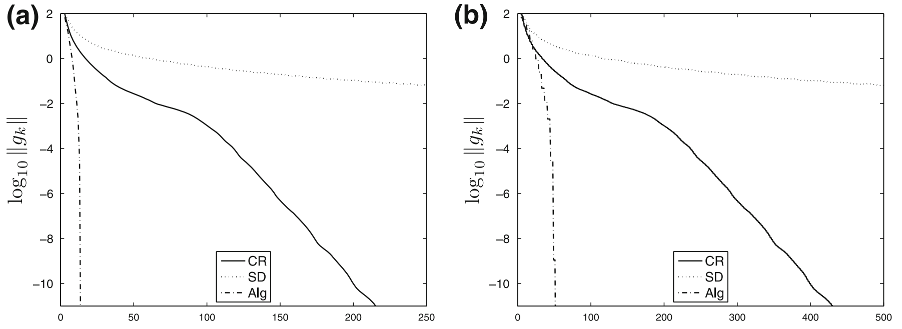

Fig. 2 Problem 1: $\log_{10}\|g_k\|$ against the number of iterations where inner products need to be computed (*left*) and the total number of inner products (*right*)

uniformly distributed on the unit sphere. Notice that any increase of n would make the situation more favorable to the algorithm we propose relative to CR and CG. Indeed, for given m and M, the behaviors of CR and CG depend on the particular values taken by the eigenvalues of A (and generally the rates of CR and CG deteriorate when n increases) whereas the proposed algorithm behaves similarly whatever the dimension n and the spectrum of A.

In Problem 1, x_0 is uniformly distributed on the unit sphere and the n normalized eigenvalues are distributed according to the Marchenko–Pastur density $p_c(x) = \sqrt{(b-x)(x-a)}/(2\pi x c^2)$, $a = (1-c)^2 < x < b = (1+c)^2$, where we chose a 'neutral' value $c = 1/2$. These random eigenvalues are then scaled from the interval $[a, b] = [1/4, 9/4]$ back to $[m, M] = [1, 1{,}000]$. The Marchenko–Pastur density describes the asymptotic behavior of eigenvalues of suitably randomized large positive definite matrices. Problem 2 corresponds to the worst-case situation for $n-1$ steps of CR: the eigenvalues of the matrix A are $\lambda_i = \frac{M+m}{2} + \frac{M-m}{2}\cos\left(\frac{\pi i}{n-1}\right)$ for $i = 1, \ldots, n$ and the initial point x_0 is such that the α_i^2 in the decomposition $g_0 = \sum_i \alpha_i q_i$ are proportional to $\alpha_1^2 = 1/2\lambda_1$, $\alpha_n^2 = 1/2\lambda_n$ and $\alpha_j^2 = 1/\lambda_j$ for $j = 2, \ldots, n-1$, see [8] for details.

In Figs. 1 and 2 for typical runs of SD, CR and the proposed algorithm (abbreviated as 'Alg') we plot the evolution of the decimal logarithm of the L_2-norm $\|g_k\| = \sqrt{(g_k, g_k)}$ of the gradients.

CR is optimal (and thus is better than CG) in terms of the rate of convergence R_k after k iterations, the proposed algorithm cannot therefore do better with respect to this criterion. It indeed does worse, see Fig. 1. Nevertheless, although the proposed algorithm requires more iterations than CR to obtain a similar value of $\|g_k\|$, it is very competitive in terms of computational cost. To illustrate this, consider again Problem 1 as in Fig. 1a. Figure 2a shows $\log_{10}\|g_k\|$ against the number of iterations where inner products are computed (possibly in parallel). Figure 2b shows $\log_{10}\|g_k\|$ against the total number of inner products calculated. For CR, the plot on Fig. 2a exactly matches that on Fig. 1a as we assumed that the computations of inner products are parallelized at each iteration; the decrease of $\log_{10}\|g_k\|$ for CR is two times slower on Fig. 2b than on Fig. 1a since CR uses two inner products per iteration.

Note that in the run of 500 iterations of the proposed algorithm, the estimators for m and M were updated 12 times (in addition to the computation of the estimators at Stage I) and 3 (resp. 5) iterations with $\beta_k = \widehat{M}_k$ were used for Problem 1 (resp. Problem 2). The total number of inner products computed was $52 = 4 + 4 \cdot 12$.

Conclusion

We proposed a gradient algorithm for optimizing a quadratic function $(1/2)(Ax, x) - (b, x)$, where A is a symmetric positive-definite matrix of size $n \times n$. We have demonstrated that if n is large and the distribution of the eigenvalues of the matrix A is dense on the spectrum, then the convergence rate of the algorithm we propose is not much worse than the rate achieved by the Conjugate-Gradient algorithm, although the iterations are simpler. For sparse matrices, the proposed algorithm can significantly outperform the Conjugate-Gradient algorithm in terms of computational cost: only $O(\log k)$ inner products need to be computed in k iterations.

References

1. Barzilai, J., Borwein, J.: Two-point step size gradient methods. IMA J. Numer. Anal. **8**, 141–148 (1988)
2. Erdös, P., Turan, P.: On interpolation. III. Interpolation theory of polynomials. Ann. Math. **41**(3), 510–553 (1940)
3. Golub, G., Loan, C.V.: Matrix Computations, 3rd edn. Johns Hopkins University Press, Baltimore (1996)
4. Haycroft, R., Pronzato, L., Wynn, H., Zhigljavsky, A.: Studying convergence of gradient algorithms via optimal experimental design theory. In: Optimal Design and Related Areas in Optimization and Statistics, pp. 13–37. Springer, Berlin (2009)
5. Krasnosel'skii, M.A., G.Krein, S.: An iteration process with minimal residues. Mat. Sb. **31**(4), 315–334 (1952)
6. Meurant, G.: The block preconditioned conjugate gradient method on vector computers. BIT **24**, 623–633 (1984)
7. Pronzato, L., Wynn, H., Zhigljavsky, A.: Asymptotic behaviour of a family of gradient algorithms in $\mathbb{R}^d$ and Hilbert spaces. Math. Program. A **107**(3), 409–438 (2006)
8. Pronzato. L., Wynn. H.P., Zhigljavsky. A.: A dynamical-system analysis of the optimum s-gradient algorithm. In: Optimal Design and Related Areas in Optimization and Statistics, pp. 39–80. Springer, Berlin (2009)
9. Pronzato, L., Zhigljavsky, A.: Gradient algorithms for quadratic optimization with fast convergence rates. Comput. Optim. Appl. **50**(3), 597–617 (2011)

10. Saad, Y.: Practical use of polynomial preconditionings for the conjugate gradient method. SIAM J. Sci. Stat. Comp. **6**(4), 865–881 (1985)
11. Slater, B.: Gaps and steps for the sequence $n\theta$ mod 1. Math. Proc. Camb. Phil. Soc. **63**, 1115–1123 (1967)
12. van der Vorst, H.A. (2002) Iterative methods for large linear systems. Tech. rep., Math. Inst. Utrecht Univ.