

Minimizing convex quadratics with variable precision conjugate gradients

Serge Gratton¹ | Ehouarn Simon¹  | David Tittley-Peloquin² | Philippe L. Toint³

¹Université de Toulouse, INP, IRIT, Toulouse, France

²McGill University, Montreal, Quebec, Canada

³NAXYS, University of Namur, Namur, Belgium

Correspondence

Ehouarn Simon, Université de Toulouse, INP, IRIT, 2 rue Camichel, BP 7122, Toulouse 31071, France.
Email: ehoularn.simon@toulouse-inp.fr

Funding information

Agence Nationale de la Recherche, Grant/Award Number: ANR-11-LABX-0040-CIMI within the program ANR-11-ID

Summary

We investigate the method of conjugate gradients, exploiting inaccurate matrix-vector products, for the solution of convex quadratic optimization problems. Theoretical performance bounds are derived, and the necessary quantities occurring in the theoretical bounds estimated, leading to a practical algorithm. Numerical experiments suggest that this approach has significant potential, including in the steadily more important context of multiprecision computations.

KEYWORDS

conjugate gradients, high-performance computing, multiprecision arithmetic, positive-definite linear systems, quadratic optimization, variable accuracy

1 | INTRODUCTION

We are interested in iterative methods for solving convex quadratic optimization problems

$$\min_{x \in \mathbb{R}^n} q(x) \stackrel{\text{def}}{=} \frac{1}{2} x^T A x - b^T x \quad (1)$$

and large symmetric positive-definite linear systems

$$A x = b, \quad (2)$$

where $A \in \mathbb{R}^{n \times n}$ is symmetric positive definite and $b \in \mathbb{R}^n$. Such problems are at the center of efficient methods in a large variety of domains in applied mathematics, the most prominent being large-scale numerical nonlinear optimization and the numerical solution of discretized partial differential equations. It is thus critical to make the solution of (1) and (2) as efficient as possible. The cost of most iterative methods for solving these problems is often dominated by the (potentially many) computations of products of the form $A p$ for some vector p . It is therefore of interest to investigate if efficiency gains may be obtained for this “core” operation. This is the object of the present article.

Two different contexts motivate the analysis presented here. The first is the frequent occurrence of problems involving a hierarchy of model representations, themselves providing approximations of A to compute the product $A p$. This occurs, for instance, in discretized applications, possibly in a multiresolution framework, or in inexact weighted linear

and nonlinear least-squares, where the product itself is obtained by applying an iterative procedure¹. The second is the increasing importance of computations in multiprecision arithmetic on the new generations of high-performance computers (see References 2-8 and the many references therein), in which the use of varying levels of floating point precision is a key ingredient for obtaining state-of-the-art energy-efficient computer architectures. In both cases, using inexact matrix-vector products (while controlling their inexactness) within the method of conjugate gradients (CG) of Hestenes and Stiefel⁹ is a natural option.

Although the use of inexact matrix-vector products in CG and other Krylov subspace iterative methods has already been investigated (see, e.g., References 10-14), the proposed analyses typically focus on bounding the Euclidean norm of the residual. This leads to criteria for controlling the inaccuracy of the matrix-vector products that are somewhat different from the one used here (for instance, see equation (4.4) from Reference 14 for the inexact CG, equations (5.8) and (5.9) from Reference 10 for the inexact FOM and GMRES). We refer to Section 3.5 for a discussion about the differences between these criteria. We further refer to [15, section 11], and the many references therein, for a review on inexact Krylov methods.

To the best of our knowledge, none considers the decrease in the objective function of the associated optimization problem (1), which is related to the energy norm of the error in (2). This point of view is, however, important. In optimization, monitoring the evolution of the nonlinear (and possibly nonconvex) objective function or of its model is an obvious concern: ensuring a fraction of the optimal decrease is, for instance, a standard convergence argument in trust-region and inexact Newton methods (see, e.g., [16, chapter 6]). In applications arising from elliptic partial differential equations, several authors have argued that monitoring the energy norm of the error leads to better termination rules, avoiding under- or over-solving (see References 17-23). Monitoring the energy norm of the error has been applied for CG with inexact preconditioning, where iterative methods can be embedded for the resolution of the system associated with the computation of the new directions. A stopping criterion in the resolution of the inner systems was suggested in Reference 24, that results in a decrease of the energy norm of the residual, and so a decrease of the associated quadratic. However, the accuracy level of the matrix-vector products is fixed.

Contributions. In Section 2, we derive theoretical bounds on the value of the quadratic in (1) in the presence of inexact matrix-vector products in CG. We then derive computable estimates required in these theoretical bounds in Section 3, leading to a practical inexact CG algorithm. In Section 4, we show that very significant efficiency gains can be obtained in CG by this approach, both in the case where the accuracy of Ap can be varied continuously and in the case where it is bound to discrete prescribed levels (as is the case in multiprecision arithmetic).

Notations. Throughout this article, $\|\cdot\|_2$ denotes the standard Euclidean norm for vectors and its induced (spectral) norm for matrices. If M is symmetric positive definite and x is a vector, $\|x\|_M = \|M^{1/2}x\|_2$. The dual norm of $\|\cdot\|_M$ with respect to the Euclidean inner product is $\|\cdot\|_{M^{-1}}$. $\text{Tr}(M)$ is the trace of the matrix M . If M is symmetric positive definite, $\lambda_{\min}(M)$ and $\lambda_{\max}(M)$ are its smallest and largest eigenvalue, respectively. e_i is the i th vector of the canonical basis of \mathbb{R}^n .

2 | ANALYSIS OF INEXACT CG

We start by stating the CG algorithm with inexact matrix-vector products, initialized with $x_0 = 0$.

In the above algorithm, the matrix E_k represents a perturbation of A and is the source of inexactness in the matrix-vector product at iteration k .

The name “inexact CG” can be viewed as an abuse of language since, due to the error in the matrix-vector product at Step 2 of the algorithm, the standard conjugacy of search directions and orthogonality of the residuals is lost. We nevertheless continue to use the designation “inexact CG” because of the very close similarity between the statements of the method and true CG.

It is known that the residual of the linear system (2) provides a handle for monitoring the error in the quadratic $q(x)$ in (1), provided it is considered in the appropriate norm. Indeed, if $x_* = A^{-1}b$ is the solution of (2) and $r(x) \stackrel{\text{def}}{=} Ax - b$, then

$$\begin{aligned} \frac{1}{2} \|r(x)\|_{A^{-1}}^2 &= \frac{1}{2} (Ax - b)^T A^{-1} (Ax - b) \\ &= \frac{1}{2} (x - x_*)^T A (x - x_*) \end{aligned}$$

¹Our starting point was a nonlinear weighted least-squares problem occurring in large-scale data assimilation for weather forecasting,¹ where the inverse of the weighting matrix cannot be computed. It thus requires the iterative solution of an innermost linear system.

$$\begin{aligned}
&= \frac{1}{2}(x^T Ax - 2x^T Ax_* + x_*^T Ax_*) \\
&= q(x) - q(x_*).
\end{aligned} \tag{3}$$

Algorithm 1. Theoretical inexact CG algorithm

Given symmetric positive definite $A \in \mathbb{R}^{n \times n}$, $b \in \mathbb{R}^n$,

(1) Set $x_0 = 0$, $\beta_0 = \|b\|_2^2$, $r_0 = -b$, and $p_0 = b$

for $k = 0, 1, \dots$, **do**

(2) $c_k = (A + E_k)p_k$

(3) $\alpha_k = \beta_k / p_k^T c_k$

(4) $x_{k+1} = x_k + \alpha_k p_k$

(5) $r_{k+1} = r_k + \alpha_k c_k$

if $\|r_{k+1}\|_{A^{-1}}$ is small enough **then**

(6) **Stop**

end if

(7) $\beta_{k+1} = r_{k+1}^T r_{k+1}$

(8) $p_{k+1} = -r_{k+1} + (\beta_{k+1} / \beta_k)p_k$

end for

However, monitoring $\|r(x_k)\|_{A^{-1}}$ in inexact CG requires that $r(x_k)$ or a sufficiently good approximation thereof be available, and that its A^{-1} -norm be computed or estimated, both of which are nontrivial.

When the products Ap are computed inexactly, the vector r_k recurred in CG is not the same as the true residual $r(x_k) = Ax_k - b$. Current literature (see References 10,11) focuses on bounding the *residual gap* measured in the Euclidean norm, $\|r(x_k) - r_k\|_2$. Because we are interested in the optimization problem (1), we need to bound the residual gap in the A^{-1} -norm, as motivated by (3) and the following lemma.

Lemma 1. Suppose that, at iteration k of the inexact CG algorithm,

$$\max [\|r(x_k) - r_k\|_{A^{-1}}, \|r_k\|_{A^{-1}}] \leq \frac{\sqrt{\epsilon}}{2} \|b\|_{A^{-1}} \tag{4}$$

for some $\epsilon > 0$. Then

$$|q(x_k) - q(x_*)| \leq \epsilon |q(x_*)|. \tag{5}$$

Proof. First, evaluating the quadratic $q(x)$ at $x = x_* = A^{-1}b$ gives a very useful identity, namely that

$$|q(x_*)| = -q(x_*) = \frac{1}{2} \|b\|_{A^{-1}}^2 = \frac{1}{2} \|x_*\|_A^2 = \frac{1}{2} |b^T x_*|. \tag{6}$$

Successively using (3), the triangle inequality, (4), and (6), we deduce that

$$\begin{aligned}
|q(x_k) - q(x_*)| &= \frac{1}{2} \|r(x_k)\|_{A^{-1}}^2 \\
&\leq \frac{1}{2} (\|r(x_k) - r_k\|_{A^{-1}} + \|r_k\|_{A^{-1}})^2 \\
&\leq \frac{1}{2} (\sqrt{\epsilon} \|b\|_{A^{-1}})^2 \\
&= \epsilon |q(x_*)|.
\end{aligned} \tag{7}$$

■

Note that (5) implies

$$q(x_*) \leq q(x_k) \leq (1 - \epsilon)q(x_*).$$

Thus, if (4) holds, the quadratic at x_k is within a factor $(1 - \epsilon)$ of its minimal value. In addition, because $q(x_0) = 0$,

$$|q(x_k) - q(x_0)| \geq (1 - \epsilon)|q(x_*) - q(x_0)|.$$

Thus, if (4) holds, the decrease of the quadratic $q(x)$ obtained at x_k is at least $(1 - \epsilon)$ times the maximum obtainable decrease. This is exactly the type of result required to terminate the minimization in a trust-region context (see [16, theorem 6.3.5]).

We assume that CG makes $\|r_k\|_{A^{-1}}$ small eventually, which it is not guaranteed for the inexact CG. However, if we introduce a reorthogonalization step on the internally recurred residuals, then r_k becomes zero after at most n steps. The rest of this article is devoted to analyzing how to enforce the part of (4) related to the residual gap, that is, the condition $\|r(x_k) - r_k\|_{A^{-1}} \leq \frac{\sqrt{\epsilon}}{2}\|b\|_{A^{-1}}$. Because this last condition measures the residual gap in the A^{-1} -norm (i.e., a norm in the dual space), it is natural to use the A -norm in the primal space and the primal-dual matrix norm defined by

$$\|E\|_{A^{-1},A} \stackrel{\text{def}}{=} \sup_{x \neq 0} \frac{\|Ex\|_{A^{-1}}}{\|x\|_A} = \|A^{-1/2}EA^{-1/2}\|_2. \quad (8)$$

We will thus use this norm to measure the size of the backward error E_k made in the matrix-vector product. Note for future reference that

$$\|A\|_{A^{-1},A} = 1 \quad (9)$$

and

$$\|Ex\|_{A^{-1}} \leq \|E\|_{A^{-1},A}\|x\|_A \quad (10)$$

for any vector x .

We first restate, for completeness, a simple result relating the residual gap to the error matrices E_j in inexact CG (see Reference 11).

Lemma 2. *The residual gap in the inexact CG algorithm satisfies*

$$r(x_k) - r_k = -\sum_{j=0}^{k-1} \alpha_j E_j p_j.$$

Proof. We proceed by induction. Observe that $r(x_0) - r_0 = 0$ and

$$\begin{aligned} r(x_1) - r_1 &= (Ax_1 - b) - r_1 \\ &= (\alpha_0 Ap_0 - b) - (r_0 + \alpha_0 c_0) = -\alpha_0 E_0 p_0. \end{aligned}$$

Suppose now that the result is true for iterations $j = 0, \dots, k$. From the recurrences for x_{k+1} and r_{k+1} we then have that

$$\begin{aligned} r(x_{k+1}) - r_{k+1} &= (Ax_{k+1} - b) - r_{k+1} \\ &= (Ax_k + \alpha_k Ap_k - b) - r_k + r_k - r_{k+1} \\ &= r(x_k) + \alpha_k Ap_k - r_k - \alpha_k (A + E_k) p_k \\ &= r(x_k) - r_k - \alpha_k E_k p_k, \end{aligned}$$

from which the result follows. ■

We are now in a position to derive suitable bounds on the error matrices.

Theorem 1. *Let $\epsilon > 0$ and let $\phi \in \mathbb{R}^k$ be a positive vector satisfying*

$$\sum_{j=1}^k \frac{1}{\phi_j} \leq 1. \quad (11)$$

Suppose furthermore that

$$\|E_j\|_{A^{-1},A} \leq \omega_j \stackrel{\text{def}}{=} \frac{\sqrt{\epsilon} \|b\|_{A^{-1}} \|p_j\|_A}{2 \phi_{j+1} \|r_j\|_2^2 + \sqrt{\epsilon} \|b\|_{A^{-1}} \|p_j\|_A} \quad (12)$$

for all $j \in \{0, \dots, k-1\}$. Then

$$\|r(x_k) - r_k\|_{A^{-1}} \leq \frac{\sqrt{\epsilon}}{2} \|b\|_{A^{-1}}.$$

If additionally

$$\|r_k\|_{A^{-1}} \leq \frac{\sqrt{\epsilon}}{2} \|b\|_{A^{-1}}, \quad (13)$$

then (4) and (5) both hold.

Proof. First note that (12) ensures that $\omega_j \in (0, 1)$. Lemma 2, the triangle inequality, and (10) imply that

$$\begin{aligned} \|r(x_k) - r_k\|_{A^{-1}} &\leq \sum_{j=0}^{k-1} \|\alpha_j E_j p_j\|_{A^{-1}} \\ &\leq \sum_{j=0}^{k-1} |\alpha_j| \|E_j\|_{A^{-1},A} \|p_j\|_A. \end{aligned} \quad (14)$$

Now, using (10) again and (12),

$$\alpha_j = \frac{\|r_j\|_2^2}{p_j^T (A + E_j) p_j} \leq \frac{\|r_j\|_2^2}{p_j^T A p_j - \|E_j\|_{A^{-1},A} \|p_j\|_A^2} \leq \frac{\|r_j\|_2^2}{(1 - \omega_j) \|p_j\|_A^2}.$$

Substituting this bound in (14) and using (12) again, we obtain

$$\|r(x_k) - r_k\|_{A^{-1}} \leq \sum_{j=0}^{k-1} \frac{\omega_j}{1 - \omega_j} \frac{\|r_j\|_2^2}{\|p_j\|_A}. \quad (15)$$

But the definition of ω_j in (12) gives

$$\frac{\omega_j}{1 - \omega_j} = \frac{\sqrt{\epsilon} \|b\|_{A^{-1}} \|p_j\|_A}{2 \phi_{j+1} \|r_j\|_2^2},$$

so that (15) becomes

$$\begin{aligned} \|r(x_k) - r_k\|_{A^{-1}} &\leq \sum_{j=0}^{k-1} \frac{\sqrt{\epsilon} \|b\|_{A^{-1}} \|p_j\|_A}{2 \phi_{j+1} \|r_j\|_2^2} \frac{\|r_j\|_2^2}{\|p_j\|_A} \\ &= \frac{\sqrt{\epsilon}}{2} \|b\|_{A^{-1}} \sum_{j=0}^{k-1} \frac{1}{\phi_{j+1}} \leq \frac{\sqrt{\epsilon}}{2} \|b\|_{A^{-1}}. \end{aligned} \quad (16)$$

The result then follows from Lemma 1. ■

Observe that (12) allows a perturbation in A whose norm $\|E_j\|_{A^{-1},A}$ depends on the ratio $\frac{\|r_j\|_2^2}{\|p_j\|_A}$. In particular, when this ratio decreases as the iterations proceed (although the decrease of $\|r_j\|_2$ is not generally monotonic in CG) the error in the matrix-vector products is permitted to grow.

Some additional comments on Theorem 1 are in order at this stage.

1. (12) assumes that the primal-dual norm is the natural norm for measuring the size of the error matrices E_j . While this may be true in certain applications, $\|E_j\|_{A^{-1},A}$ may be difficult to compute or estimate in practice.
2. Even discounting that potential difficulty, verifying conditions (12) and (13) remains impractical, as the quantities $\|p_j\|_A$, $\|b\|_{A^{-1}}$, and $\|r_k\|_{A^{-1}}$ are not readily available in the course of the inexact CG algorithm.
3. The ϕ_j appearing in (12) may be used as part of a global “error management strategy.” A simple choice that obviously satisfies (11) is to define $\phi_j = k_{\max}$ for all j , where k_{\max} is the maximum allowable number of iterations. In fact, the ϕ_j can be used to further advantage.

We shall address the above issues in the following section.

3 | A PRACTICAL INEXACT CG ALGORITHM

3.1 | Managing the inaccuracy budget

An important ingredient of a practical inexact CG algorithm is the choice of the ϕ_j in (12). Note that (11) constrains the ϕ_j over *all iterations until termination*. As mentioned earlier, choosing $\phi_j = k_{\max}$ is adequate but often suboptimal. Indeed, it is possible to adjust the ϕ_j adaptively, in particular when the inaccuracy of the product Ap cannot be varied continuously but is bound to prescribed levels (e.g., different levels of floating point precision).

Suppose that for a given ϕ_{j+1} , an inexactness $\omega_j(\phi_{j+1})$ in the matrix-vector product at step j is allowed by Theorem 1, but the actual error E_j satisfies $\|E_j\|_{A^{-1},A} = \hat{\omega}_j < \omega_j$. This implies that a larger value $\hat{\phi}_{j+1}$ could have been used instead of ϕ_{j+1} . Solving for $\hat{\phi}_{j+1}$ in the linear equation $\hat{\omega}_j = \omega_j(\hat{\phi}_{j+1})$ in (12) gives

$$\hat{\phi}_{j+1} = \frac{1 - \hat{\omega}_j}{\hat{\omega}_j} \frac{\sqrt{\epsilon} \|b\|_{A^{-1}} \|p_j\|_A}{2\|r_j\|_2^2} > \phi_{j+1}. \quad (17)$$

We may then distribute the unused inaccuracy $1 - \sum_{p=1}^{j+1} \hat{\phi}_p^{-1}$ evenly in the remaining $k_{\max} - j - 1$ iterations, by setting

$$\phi_i = \frac{k_{\max} - j - 1}{1 - \sum_{p=1}^{j+1} \hat{\phi}_p^{-1}}, i = j + 2, \dots, k_{\max}.$$

This leads to smaller values of ϕ_i (and therefore larger allowable errors) in subsequent iterations. The updated ϕ_i still satisfies (11), as shown below:

$$\begin{aligned} \sum_{i=1}^{k_{\max}} \phi_i^{-1} &= \sum_{i=1}^{j+1} \phi_i^{-1} + \sum_{i=j+2}^{k_{\max}} \phi_i^{-1} \\ &= \sum_{i=1}^{j+1} \phi_i^{-1} + (k_{\max} - j - 1) \frac{1 - \sum_{p=1}^{j+1} \hat{\phi}_p^{-1}}{k_{\max} - j - 1} \\ &= \sum_{i=1}^{j+1} \phi_i^{-1} + 1 - \sum_{i=1}^{j+1} \hat{\phi}_i^{-1} < 1, \end{aligned}$$

since $\hat{\phi}_i > \phi_i$. In practice, this allows maintaining only single running values for ϕ_{j+1} and

$$\Phi_j \stackrel{\text{def}}{=} 1 - \sum_{p=1}^j \hat{\phi}_p^{-1}$$

for j ranging from 0 to $k_{\max} - 1$.

3.2 | Computable estimates of $\|E_j\|_{A^{-1},A}$ and $\|p_j\|_A$

We now attempt to estimate the quantities required by Theorem 1 that are unavailable in the inexact CG algorithm.

We first consider that $\|E_j\|_{A^{-1},A}$ in (12) may not be available from the application context and note that, from (8),

$$\|E_j\|_{A^{-1},A} = \|A^{-1/2}E_jA^{-1/2}\|_2 \leq \lambda_{\min}(A)^{-1}\|E_j\|_2, \quad (18)$$

so that a bound on $\|E_j\|_2$ can be used provided one knows (an approximation of) the smallest eigenvalue of A . To estimate $\|p_j\|_A$, we can use the fact that

$$\lambda_{\min}(A)^{1/2}\|p_j\|_2 \leq \|p_j\|_A \leq \lambda_{\max}(A)^{1/2}\|p_j\|_2.$$

However, for ill-conditioned problem, the above bounds are likely to be very loose. Another approach is to choose

$$\|p_j\|_A \approx \sqrt{\frac{1}{n}\text{Tr}(A)}\|p_j\|_2. \quad (19)$$

This can be justified by the fact that each side of the above expression would have the same mean squared value if the entries of p_j were independent standard normal variables.

3.3 | A computable estimate of $\|b\|_{A^{-1}}$

Finding an estimate of $\|b\|_{A^{-1}}$ is more difficult, since this quantity is related to the value of the quadratic $q(x)$ at the solution x^* , see (6). Note from (3) that

$$|q(x_*)| \leq |q(x_k)| + \frac{1}{2}\|r(x_k)\|_{A^{-1}}^2.$$

To our knowledge, the best available approximation is the absolute value of the quadratic at the current iterate, and thus we choose

$$\|b\|_{A^{-1}} = \sqrt{2|q(x_*)|} \approx \sqrt{2|q(x_k)|}. \quad (20)$$

If there is no error in the products Ap_j in CG, assuming exact arithmetic, $r(x_k)$ is orthogonal to x_k and it follows that

$$q(x_k) = \frac{1}{2}x_k^T Ax_k - b^T x_k = \frac{1}{2}x_k^T (Ax_k - b) - \frac{1}{2}b^T x_k = -\frac{1}{2}b^T x_k.$$

In the presence of errors in the matrix-vector products, the above property may no longer hold, and it is of interest to analyze how much

$$q_k \stackrel{\text{def}}{=} -\frac{1}{2}b^T x_k \quad (21)$$

differs from $q(x_k)$. This is also important if the decrease in the quadratic objective function is used for other purposes, as is the case, for instance, in trust-region methods, where it is a key ingredient in the management of the trust-region radius (see [16, chapter 6]). To this end, we first prove the following backward error property.

Lemma 3. Let $A \in \mathbb{R}^{n \times n}$ be symmetric positive definite and $b \in \mathbb{R}^n$. For any nonzero $x \in \mathbb{R}^n$, with $r(x) = Ax - b$,

$$\min_E \left\{ \frac{\|E\|_{A^{-1},A}}{\|A\|_{A^{-1},A}} |x^T (A + E)x = x^T b| \right\} = \frac{|x^T r(x)|}{\|x\|_A^2}. \quad (22)$$

Proof. Using (8) and (9), we rewrite the optimization problem as

$$\min_E \{ \|A^{-1/2}EA^{-1/2}\|_2 |x^T Ex = x^T (b - Ax)| \}$$

which, since A is positive definite and setting $y = A^{1/2}x$, is itself equivalent to

$$\min_E \{ \|A^{-1/2}EA^{-1/2}\|_2 | y^T A^{-1/2}EA^{-1/2}y = -x^T r(x) \}.$$

But for any nonzero vector y and scalar γ ,

$$\min_M \{ \|M\|_2 | y^T My = \gamma \} = \frac{\gamma}{\|y\|_2^2}$$

and the minimum is attained by $M = \gamma y y^T / \|y\|_2^4$ with $\|M\|_2 = \gamma / \|y\|_2^2$. Thus, the minimum in (22) is

$$\|E\|_{A^{-1},A} = \|A^{-1/2}EA^{-1/2}\|_2 = \frac{|x^T r(x)|}{\|x\|_A^2}.$$

■

We can now show that q_k remains close to $q(x_k)$ despite the inexact matrix-vector products.

Theorem 2. *Let x_k be the result of applying the inexact CG algorithm and let $q_k = -\frac{1}{2}b^T x_k$. If the inexactness of the matrix-vector products is controlled as in (12), and (13) holds, then*

$$\frac{|q(x_k) - q_k|}{|q(x_*)|} \leq \frac{\sqrt{\epsilon}(1 + \sqrt{\epsilon})}{2}. \quad (23)$$

Proof. We deduce from Lemma 3 that there exists a quadratic

$$\hat{q}(x_k) = \frac{1}{2}x_k^T (A + E)x_k - b^T x_k \quad (24)$$

such that by construction $\hat{q}(x_k) = -\frac{1}{2}b^T x_k = q_k$ and

$$\begin{aligned} |q(x_k) - q_k| &= |q(x_k) - \hat{q}(x_k)| \\ &= \frac{1}{2}|x_k^T E x_k| \\ &\leq \frac{1}{2}\|E\|_{A^{-1},A} \|x_k\|_A^2 \\ &= \frac{1}{2}|x_k^T r(x_k)| \\ &\leq \frac{1}{2}\|r(x_k)\|_{A^{-1}} \|x_k\|_A, \end{aligned} \quad (25)$$

where we used (24), Lemma 3, and the Cauchy-Schwarz inequality. But by Theorem 1 and (6),

$$\begin{aligned} \|x_k\|_A &\leq \|x_*\|_A + \|x_k - x_*\|_A \\ &= \|x_*\|_A + \|r(x_k)\|_{A^{-1}} \\ &\leq \|x_*\|_A + \|r(x_k) - r_k\|_{A^{-1}} + \|r_k\|_{A^{-1}} \\ &\leq (1 + \sqrt{\epsilon})\|x_*\|_A \end{aligned}$$

and hence, using (13) and (6) again,

$$|q(x_k) - q_k| \leq \frac{1}{2}\|r(x_k)\|_{A^{-1}}(1 + \sqrt{\epsilon})\|x_*\|_A \leq \frac{1}{2}\sqrt{\epsilon}(1 + \sqrt{\epsilon})|q(x_*)|.$$

■

The above theorem shows that q_k in (21) remains close to $q(x_k)$. The bound is considerably weaker than (5), but it is likely to be pessimistic as we have not taken into account in (25) the fact that the angle between x_k and $r(x_k)$ is expected

to be small. (For CG in exact arithmetic with exact matrix-vector products, $x_k^T r_k = 0$.) This will be numerically confirmed in Section 4.

From (20) and Theorem 2, we use the approximation

$$\|b\|_{A^{-1}} \approx \sqrt{2|q_k|}, \quad (26)$$

for $k = 1, \dots, k_{\max}$ in our practical inexact CG algorithm. For $k=0$, because $x_0=0$, we use the rougher approximation $\|b\|_{A^{-1}} \approx \|b\|_2 / \sqrt{\lambda_{\max}(A)}$. This conservative estimate is likely to result with the computation of the first matrix-vector product in double precision. Different values could be naturally chosen in $\left[\frac{\|b\|_2}{\sqrt{\lambda_{\max}(A)}}, \frac{\|b\|_2}{\sqrt{\lambda_{\min}(A)}} \right]$.

3.4 | A computable estimate of $\|r_k\|_{A^{-1}}$

It is also necessary to estimate $\|r_k\|_{A^{-1}}$, in order to perform the termination test (13). Estimating the energy norm of the error in CG is a well-studied problem (see, e.g., References 25-29). We follow these ideas, ignoring pathological convergence instances, and estimate

$$\frac{1}{2} \|r_k\|_{A^{-1}}^2 \approx q(x_k) - q(x_*) \approx q(x_{k-d}) - q(x_k)$$

where d is a small integer ($d=10$ in our case). Using this estimate in (13) leads to

$$q(x_{k-d}) - q(x_k) \leq \frac{1}{4} \epsilon |q(x_k)|,$$

which, using (26), is itself approximated by

$$q_{k-d} - q_k \leq \frac{1}{4} \epsilon |q_k|. \quad (27)$$

3.5 | Resulting practical algorithm

We now consolidate our approximations in order to obtain a practical version of the inexact CG algorithm that relies only on computable quantities. We note that our definitions of these quantities nevertheless requires the user to provide a (potentially very rough, see Section 4) approximation of the smallest and largest eigenvalues of A .

Making use of (18), (19), and (26), we suggest to approximate (12) by

$$\frac{\|E_j\|_2}{\lambda_{\min}(A)} \leq \frac{\sqrt{\epsilon} \sqrt{|q_j|} \sqrt{\text{Tr}(A)} \|p_j\|_2}{\sqrt{2n} \phi_{j+1} \|r_j\|_2^2 + \sqrt{\epsilon} \sqrt{|q_j|} \sqrt{\text{Tr}(A)} \|p_j\|_2}, \quad (28)$$

for $j=1, \dots, k-1$. The formula is similar for $j=0$, with $\sqrt{|q_0|}$ replaced by $\sqrt{2}\|b\|_2/\sqrt{\lambda_{\max}(A)}$. We also replace the termination test (13) by its practical version (27).

Our practical bound differs from the one suggested by Du et al.¹⁴ and defined as

$$\|E_j\|_2 \leq \frac{\sigma_{\min}(A)}{2} \min \left(1, \frac{\sqrt{\epsilon} \|p_j\|_2}{m \|r_j\|_2^2} \right), \quad \forall j = 0, 1, \dots, m-1.$$

However, we note that the two bounds depend in the same manner of the smallest eigenvalue of A and the ratio $\frac{\|r_j\|_2^2}{\|p_j\|_2}$ when it is large. The decrease in $\frac{\|r_j\|_2^2}{\|p_j\|_2}$ allows the increase of the bounds of the error in the matrix-vector products, while the smallest eigenvalue of A may constrain the bound to remain tiny.

Algorithm 2. Practical inexact CG algorithm

Given symmetric positive definite $A \in \mathbb{R}^{n \times n}$, $b \in \mathbb{R}^n$, ϵ , k_{\max} , λ_{\min} , λ_{\max} , and reorth.

(1) Set $x_0 = 0$, $r_0 = -b$, $q_0 = 0$, $\beta_0 = \|b\|_2^2$, $u_1 = b/\beta_0$, $p_0 = b$, $\phi_0 = k_{\max}$, and $\Phi_0 = 1$

for $k = 0, 1, \dots, k_{\max}$ **do**

(2) Determine ω_k defined by the right-hand side of Equation (28)

(3) Compute the product $c_k = (A + E_k)p_k$ with $\|E_k\|_{A^{-1},A} \leq \omega_k$, also returning $\hat{\omega}_k$

(4) $\alpha_k = \beta_k/p_k^T c_k$

(5) $x_{k+1} = x_k + \alpha_k p_k$

(6) $q_{k+1} = \frac{1}{2} b^T x_{k+1}$

if $(q_{k+1-d} - q_{k+1}) \leq \frac{1}{4} \epsilon |q_{k+1}|$ **then**

(7) **Stop**

end if

(8) Compute $\hat{\phi}_k$ from $\hat{\omega}_k$ using the first part of Equation (17)

(9) $\Phi_{k+1} = \Phi_k - \hat{\phi}_k^{-1}$

if $k < k_{\max}$ **then**

(10) $\phi_{k+1} = (k_{\max} - k)/\Phi_{k+1}$

else

(11) $\phi_{k+1} = \phi_k$

end if

(12) $r_{k+1} = r_k + \alpha_k c_k$

if (reorth) **then**

for $i = 1, \dots, k$ **do**

(13) $r_{k+1} = r_{k+1} - (u_i^T r_{k+1}) u_i$

end for

(14) $\beta_{k+1} = r_{k+1}^T r_{k+1}$

(15) $u_{k+1} = r_{k+1}/\sqrt{\beta_{k+1}}$

else

(16) $\beta_{k+1} = r_{k+1}^T r_{k+1}$

end if

(17) $p_{k+1} = -r_{k+1} + (\beta_{k+1}/\beta_k) p_k$

end for

We include the option of reorthogonalization in our practical inexact CG method. It is well known that the residual vectors, though theoretically orthogonal, quickly lose their orthogonality in finite precision computations. The reorthogonalization step is applied to the sequence of the internally recurred residuals using the MGS algorithm prior to the computation of the new search direction. It means that the A -norm of the error is minimized over some expanding subspaces. To the limits of the machine precision, the iterative process is expected to converge in n steps at most, and guarantees that $\|r_k\|_{A^{-1}}$ will become lower than the prescribed tolerance.

However, it is well beyond the scope of this article to analyze the effects of finite precision and the loss of orthogonality on the convergence of CG with inexact matrix-vector products, but we do report on some numerical experiments with and without reorthogonalization in the following section.

4 | NUMERICAL EXPERIMENTS

We first provide some figures to illustrate typical behavior of the theoretical and practical inexact CG algorithms.

4.1 | Continuously varying precision

In our first example, A is a 100×100 diagonal matrix with entries logarithmically equally spaced between 1 and 10^{-4} , and $b = A[1, \dots, 1]^T$.

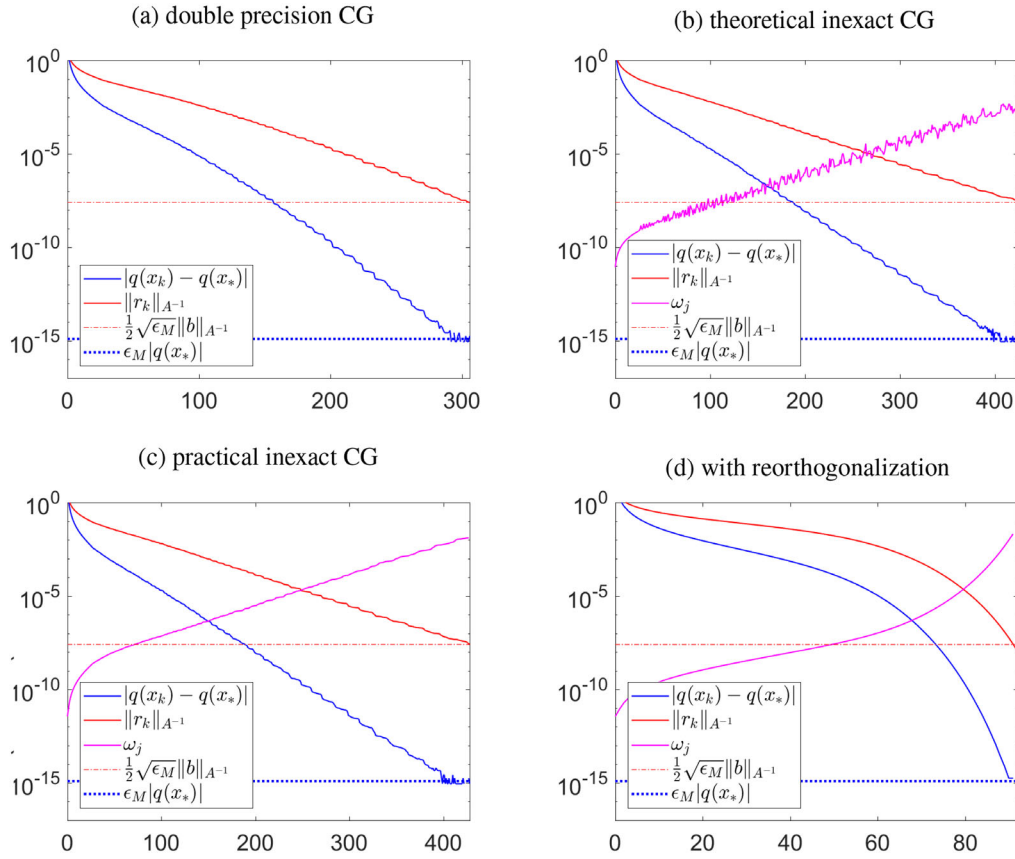


FIGURE 1 CG applied to $\text{diag}(\text{logspace}(-4, 0, 100))$. CG, conjugate gradients

We plot both the decrease in the quadratic and the A^{-1} norm of the residual. We terminate the iteration when (13) holds with $\epsilon = \epsilon_M = 2^{-52}$, the IEEE double machine precision. Lemma 1 then ensures that (5) holds.

In Figure 1a, the matrix-vector products are performed in full double precision arithmetic. In Figure 1b, they are performed inexactly, with a random perturbation E satisfying the theoretical condition (12). In Figure 1c, the matrix-vector products are once again performed inexactly, but with a random perturbation E satisfying the practical condition (28). Figure 1d is the same as Figure 1c, except CG is performed with full reorthogonalization.

In our second example, we repeat the same tests with A the matrix `nos1.mat` from the Matrix Market, scaled to have Euclidean norm 1. This is a 237×237 matrix with condition number approximately 10^8 . Results are shown in Figure 2.

In both examples, performing inexact matrix-vector products may lead to delays in convergence compared to double precision CG. The quantity ω_j in (12) can be quite oscillatory (see Figure 2b). When it is estimated as in (28), however, it seems to increase monotonically (see Figure 2c). In these examples, despite the heuristic nature of the estimates, the practical criterion (28) works just as well as the theoretical criterion (12) (compare Figure 2b,c). Without reorthogonalization, CG may require (much) more than n iterations to converge to the required tolerance, especially in the second ill-conditioned problem. In both these examples, however, reorthogonalization ensures convergence of the practical inexact CG algorithm in fewer than n iterations (see Figure 2d).

4.2 | Discrete precision levels

In the next example, we suppose that only IEEE double, single, and half precisions are available. We will refer to IEEE precisions when using the expressions double, single or half precisions until the end of the article. We switch to matrix-vector products in a lower precision once the error in the lower precision satisfies the practical condition (28). We modify the ϕ_j adaptively as explained in Section 3.1. In practice, the use of the single and half precisions is likely to trigger under- and over-flows both when converting numbers in the targeted format and computing the matrix-vector products. However, Reference 7 suggested an algorithm that prevent overflows when converting a matrix from double or single precision to

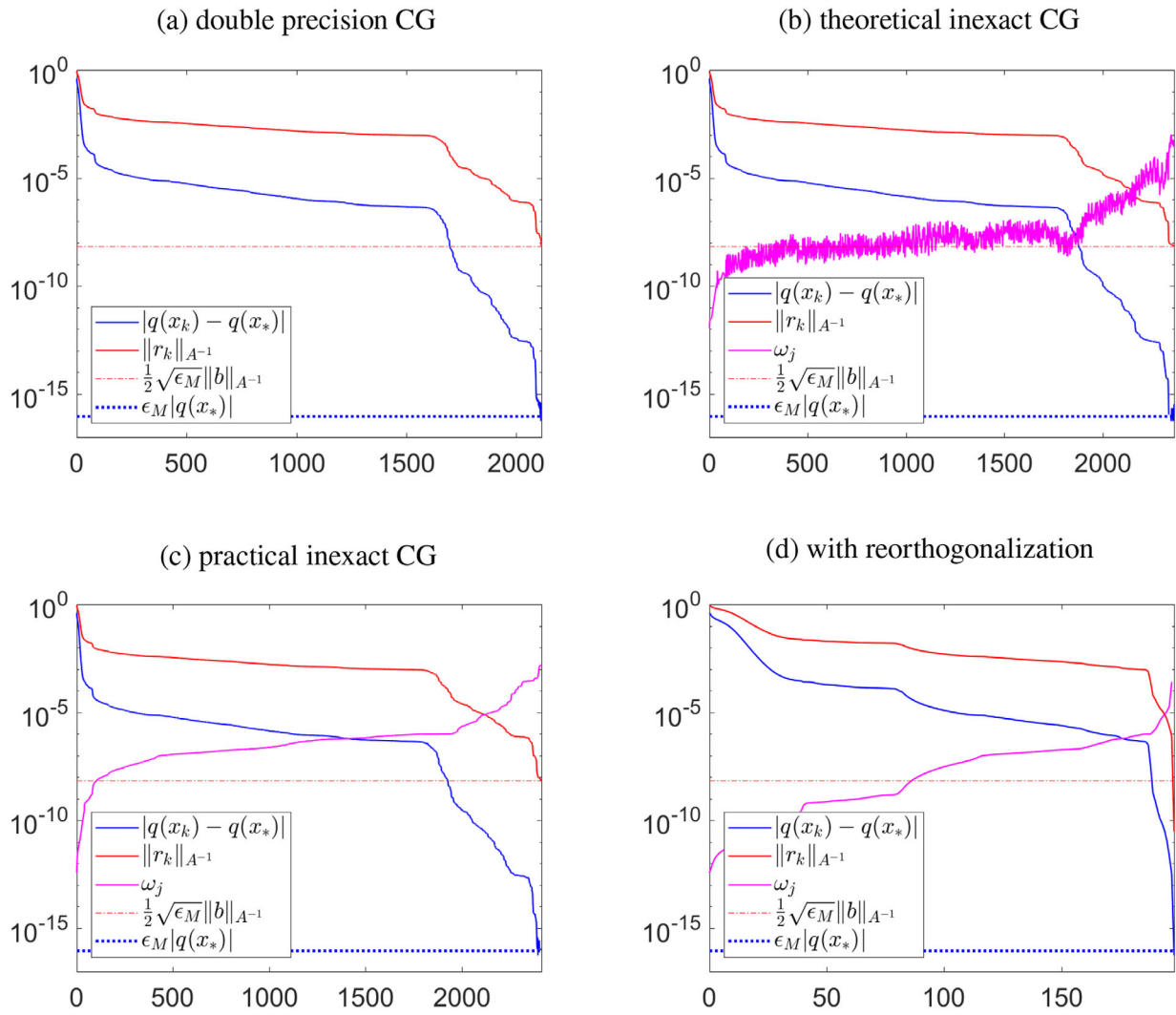


FIGURE 2 CG applied to nos1.mat. CG, conjugate gradients

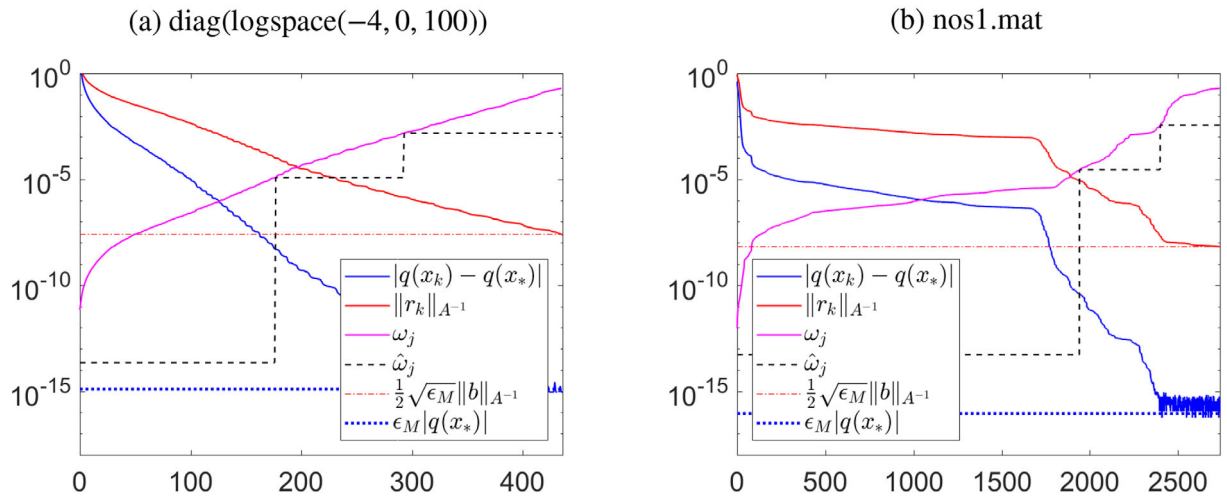


FIGURE 3 inexact CG in discrete precision levels. CG, conjugate gradients

half precision. Nevertheless if occurrence of such issues, the precision and the inaccuracy budget will have to be adapted accordingly (computation in simple or double precision). This would certainly prevent the use of the IEEE half precision depending on the application. We did not implement such strategy nor assess the occurrence of this issue in the following numerical experiments due to the use of emulated accuracy.

Results are shown in Figure 3. The convergence behavior is similar to that of the continuously varying precision case.

4.3 | Performance comparisons

We consider four algorithms:

CG:	the standard CG algorithm with products computed in double precision,
CGR:	the standard CG algorithm with reorthogonalization and products computed in double precision
iCG:	the inexact CG algorithm without reorthogonalization,
iCGR:	the inexact CG algorithm with reorthogonalization (computed in double precision).

Abbreviation: CG, conjugate gradients.

The reorthogonalization step introduced in CGR and iCGR corresponds to the Gram-Schmidt process with the standard inner product. In both algorithms, it is applied to the sequence of the recurred residual $\{r_k\}$ and is computed in double precision. iCGR differs from the flexible CG algorithm³⁰ in the sense that the Gram-Schmidt process is not used to prevent the loss of the A -conjugacy of the search directions $\{p_k\}$, but rather a loss of orthogonality of the recurred residual $\{r_k\}$. Therefore, the practical variant of iCGR does not require computing the matrix-vector products with A in double precision, at variance with the flexible CG algorithm.

In order compare the performance of these algorithms, we assume that the computational cost, and so the energy cost, is dominated by the matrix-vector products, and use the following two models of energy cost. For the continuously varying precision case, we suppose that computing Ap is performed by running a linearly convergent process, whose rate

TABLE 1 Synthetic examples: Exact bound in the continuously varying accuracy case

Method	$\kappa(A)$	n_{it}	cost	r.res.gap	r.sol.err	r.val.err.	$\kappa(A)$	n_{it}	cost	r.res.gap	r.sol.err	r.val.err.
CG	10^1	11	1.1e+01	4.6e-30	8.3e-07	8.8e-16	10^2	34	3.4e+01	6.0e-29	2.0e-06	4.6e-16
CGR		11	1.1e+01	6.4e-30	8.3e-07	4.4e-16		34	3.4e+01	8.6e-29	2.0e-06	3.2e-15
iCG		11	4.0e+00	1.1e-08	8.5e-07	2.9e-07		34	1.3e+01	1.0e-09	2.0e-06	9.1e-07
iCGR		11	4.0e+00	1.1e-08	8.4e-07	4.8e-06		34	1.3e+01	1.0e-09	2.0e-06	8.2e-07
CG	10^3	104	1.0e+02	2.6e-27	2.4e-06	2.1e-07	10^4	313	3.1e+02	1.4e-25	2.4e-06	4.0e-08
CGR		104	1.0e+02	2.7e-27	2.3e-06	4.1e-15		263	2.6e+02	1.3e-25	2.5e-06	2.5e-14
iCG		105	4.3e+01	9.8e-11	2.4e-06	7.7e-07		323	1.5e+02	9.9e-12	2.4e-06	2.1e-07
iCGR		104	4.3e+01	1.1e-10	2.3e-06	3.1e-07		263	1.2e+02	1.2e-11	2.5e-06	2.1e-07
CG	10^5	928	9.3e+02	1.0e-23	2.5e-06	1.1e-07	10^6	2764	2.8e+03	6.8e-22	2.5e-06	1.7e-08
CGR		433	4.3e+02	9.8e-24	2.4e-06	2.5e-13		554	5.5e+02	6.9e-22	2.3e-06	2.9e-12
iCG		983	5.0e+02	1.0e-12	2.5e-06	6.0e-08		3000	1.6e+03	3.5e-13	2.5e-06	9.4e-09
iCGR		433	2.2e+02	3.7e-12	2.4e-06	6.4e-08		554	3.2e+02	5.0e-12	2.3e-06	2.2e-08
CG	10^7	3000	3.0e+03	5.4e-20	1.3e-02	6.0e-07	10^8	3000	3.0e+03	1.3e-18	3.0e-01	3.6e-06
CGR		636	6.4e+02	4.6e-20	2.5e-06	1.8e-11		697	7.0e+02	3.4e-18	2.3e-06	2.8e-10
iCG		3000	1.9e+03	1.4e-13	1.9e-02	5.3e-08		3000	2.1e+03	3.2e-14	3.4e-01	6.2e-06
iCGR		636	4.0e+02	5.7e-12	2.5e-06	1.6e-07		697	4.7e+02	5.6e-12	2.3e-06	1.0e-07

Abbreviation: CG, conjugate gradients.

TABLE 2 Synthetic examples: Practical algorithms in the continuously varying accuracy case

Method	$\kappa(A)$	n_{it}	cost	r.res.gap	r.sol.err	r.val.err.	$\kappa(A)$	n_{it}	cost	r.res.gap	r.sol.err	r.val.err.
CG	10^1	11	1.1e+01	4.6e-30	8.3e-07	8.8e-16	10^2	34	3.4e+01	6.0e-29	2.0e-06	4.6e-16
CGR		11	1.1e+01	6.4e-30	8.3e-07	4.4e-16		34	3.4e+01	8.6e-29	2.0e-06	3.2e-15
iCG		21	6.0e+00	1.2e-08	1.2e-08	5.3e-06		46	1.6e+01	9.8e-10	1.6e-08	2.1e-06
iCGR		21	6.0e+00	1.2e-08	1.2e-08	4.6e-07		44	1.6e+01	1.1e-09	3.4e-08	3.5e-07
CG	10^3	104	1.0e+02	2.6e-27	2.4e-06	2.1e-07	10^4	313	3.1e+02	1.4e-25	2.4e-06	4.0e-08
CGR		104	1.0e+02	2.7e-27	2.3e-06	4.1e-15		263	2.6e+02	1.3e-25	2.5e-06	2.5e-14
iCG		112	4.6e+01	1.4e-10	9.9e-07	2.5e-07		307	1.4e+02	2.1e-11	5.0e-06	2.0e-07
iCGR		112	4.6e+01	1.3e-10	7.9e-07	4.2e-07		266	1.2e+02	3.5e-11	2.0e-06	1.6e-07
CG	10^5	928	9.3e+02	1.0e-23	2.5e-06	1.1e-07	10^6	2764	2.8e+03	6.8e-22	2.5e-06	1.7e-08
CGR		433	4.3e+02	9.8e-24	2.4e-06	2.5e-13		554	5.5e+02	6.9e-22	2.3e-06	2.9e-12
iCG		854	4.3e+02	4.6e-12	1.6e-05	7.7e-08		2314	1.3e+03	3.4e-12	5.6e-05	5.0e-07
iCGR		436	2.2e+02	6.5e-11	1.9e-06	1.9e-09		558	3.0e+02	6.5e-10	1.6e-06	4.2e-07
CG	10^7	3000	3.0e+03	5.4e-20	1.3e-02	6.0e-07	10^8	3000	3.0e+03	1.3e-18	3.0e-01	3.6e-06
CGR		636	6.4e+02	4.6e-20	2.5e-06	1.8e-11		697	7.0e+02	3.4e-18	2.3e-06	2.8e-10
iCG		3000	1.8e+03	3.4e-12	2.0e-02	1.6e-06		3000	2.0e+03	1.3e-12	3.4e-01	7.4e-07
iCGR		642	3.7e+02	4.0e-09	1.4e-06	2.6e-06		704	4.4e+02	2.4e-08	1.0e-06	1.5e-06

Abbreviation: CG, conjugate gradients.

is given by

$$\rho = \frac{\sqrt{\lambda_{\max}/\lambda_{\min}} - 1}{\sqrt{\lambda_{\max}/\lambda_{\min}} + 1}.$$

This would be the case, for instance, if $A = J^T W^{-1} J$ and only W is known². The energy cost of a full accuracy product is then given by $\log(\epsilon_M)/\log(\rho)$, where ϵ_M is the IEEE double machine precision, while that of an inexact product with accuracy requirement ω is $\log(\omega)/\log(\rho)$.

For the discrete precision levels case, we assume that the products Ap can be computed in double, single, or half precision. The gain in energy efficiency in this context depend on the details of the computer architecture. According to References 2,3,31,32, a gain between 3× and 5× can be achieved for each decrease from double to single, and from single to half. To model this, we assign a unit energy cost for a matrix-vector product in double precision, a cost $\frac{1}{4}$ for a product in single precision, and a cost $\frac{1}{16}$ for a product in half precision.

In both cases, we sum up the costs over all iterations to obtain a cost in number of equivalent full double precision matrix-vector products.

Tables 1 to 6 summarize our results for the accuracy level $\epsilon = 10^{-5}$. In these tables,

$\kappa(A)$	is the condition number of A ,
n_{it}	is the number of iterations required for termination,
cost	is the equivalent number of full accuracy products used,
r.res.gap	is the squared relative residual gap $\frac{1}{2} \ r(x) - r\ _{A^{-1}}^2 / q(x_*) $ at termination,
r.sol.err	is the relative error in the solution value $(q(x) - q(x_*)) / q(x_*) $ at termination,
r.val.err	is the relative error in the quadratic value $ q(x) - q / q(x_*) $ at termination.

Recall from (4) and (5) that $\text{r.res.gap} \leq \frac{1}{4}\epsilon$ and $\text{r.sol.err} \leq \epsilon$, while r.val.err obeys (23).

²This case occurs in approximately weighted nonlinear least-squares, for instance, in data assimilation for weather forecasting.

TABLE 3 Synthetic examples: Practical algorithms in the discrete precision levels case

Method	$\kappa(A)$	n_{it}	cost	r.res.gap	r.sol.err	r.val.err.	$\kappa(A)$	n_{it}	cost	r.res.gap	r.sol.err	r.val.err.
CG	10^1	11	1.1e+01	4.6e-30	8.3e-07	8.8e-16	10^2	34	3.4e+01	6.0e-29	2.0e-06	4.6e-16
CGR		11	1.1e+01	6.4e-30	8.3e-07	4.4e-16		34	3.4e+01	8.6e-29	2.0e-06	3.2e-15
iCG		21	1.9e+00	3.0e-11	3.2e-11	5.1e-08		44	6.7e+00	6.3e-14	3.3e-08	1.9e-08
iCGR		21	1.9e+00	3.0e-11	3.2e-11	3.9e-08		44	6.7e+00	6.7e-14	3.3e-08	1.6e-10
CG	10^3	104	1.0e+02	2.6e-27	2.4e-06	2.1e-07	10^4	313	3.1e+02	1.4e-25	2.4e-06	4.0e-08
CGR		104	1.0e+02	2.7e-27	2.3e-06	4.1e-15		263	2.6e+02	1.3e-25	2.5e-06	2.5e-14
iCG		112	2.7e+01	1.2e-16	9.1e-07	3.7e-08		302	9.6e+01	3.8e-20	4.9e-06	3.1e-07
iCGR		112	2.6e+01	1.3e-16	7.9e-07	7.9e-11		266	8.7e+01	6.1e-20	2.0e-06	1.7e-11
CG	10^5	928	9.3e+02	1.0e-23	2.5e-06	1.1e-07	10^6	2764	2.8e+03	6.8e-22	2.5e-06	1.7e-08
CGR		433	4.3e+02	9.8e-24	2.4e-06	2.5e-13		554	5.5e+02	6.9e-22	2.3e-06	2.9e-12
iCG		805	4.8e+02	2.2e-22	2.0e-05	3.5e-07		2067	1.7e+03	7.2e-22	8.9e-05	8.1e-07
iCGR		436	2.8e+02	2.1e-21	1.9e-06	4.9e-14		558	4.6e+02	1.1e-21	1.6e-06	2.0e-12
CG	10^7	3000	3.0e+03	5.4e-20	1.3e-02	6.0e-07	10^8	3000	3.0e+03	1.3e-18	3.0e-01	3.6e-06
CGR		636	6.4e+02	4.6e-20	2.5e-06	1.8e-11		697	7.0e+02	3.4e-18	2.3e-06	2.8e-10
iCG		3000	2.9e+03	4.4e-20	1.3e-02	5.9e-07		3000	3.0e+03	1.3e-18	3.0e-01	3.6e-06
iCGR		642	5.9e+02	5.8e-20	1.4e-06	2.1e-11		704	6.8e+02	3.8e-18	9.9e-07	1.3e-10

Abbreviation: CG, conjugate gradients.

4.3.1 | Synthetic examples

In order to illustrate the theory of Section 2, we first run versions of CG where we use the exact test (12) (rather than (28)), the true $\|E_j\|_{A^{-1},A}$, and the original termination test (13). This is of course impractical but allows measuring the potential for inexactness provided by Theorem 1. Table 1 reports the results obtained in the continuously varying accuracy case. Similar conclusions hold in the discrete precision levels case (not shown). First, we note that the double precision and inexact variants of CG and CGR lead to the same relative error on the solution value (r.sol.err). As targeted, this value is lower than $\epsilon = 10^{-5}$ for condition numbers lower than 10^7 without reorthogonalization. Compared with the double precision versions CG and CGR, the inexact variants iCG and iCGR exhibit very significant potential savings in the costs of the products Ap . This is especially the case when the conditioning of the problem is moderate (at most 10^5). As it could be argued that the methods discussed here should be applied on preconditioned systems, this restriction only moderately affects the practical applicability of the technique. Despite an increase in the number of iterations for iCG compared with CG for condition numbers larger 10^4 , the overall cost remains lower with iCG. Finally, the theoretical bound (12) can be too conservative, leading to smaller values of r.res.gap than necessary.

We now show the effect of using the practical algorithm outlined in Section 2. In addition to using the approximate constants and tests described in Section 3, we also use estimates of the smallest and largest eigenvalues obtained by perturbing the true eigenvalues by a random relative perturbation of magnitude between 0 and 100%, with the result that these estimates only hold in order, but typically have no exact digit. We report the results of the corresponding runs in Tables 2 (continuously varying accuracy case) and 3 (discrete precision levels case), using the same conventions as for Table 1. We also provide Figure 4 in order to better highlight the gain in costs obtained using the inexact variants iCG and iCGR for both cases. In these figures, we discarded the cases for which the double precision CG did not converge before reaching the maximum number of iterations (high condition numbers). For each variant, the black bar corresponds to the number of iterations and the gray bar to the modeled cost. (For the double precision CG and CGR, these numbers are equal and only the gray bar is visible.)

In the continuous-varying accuracy case, Table 2 shows that the practical variants of iCG and iCGR present similar performances to the theoretical inexact CG methods in terms of errors on the solution value, except for the practical iCG when the condition number is larger than 10^5 . For condition numbers in $\{10^5, 10^6\}$, we note that the number of iterations

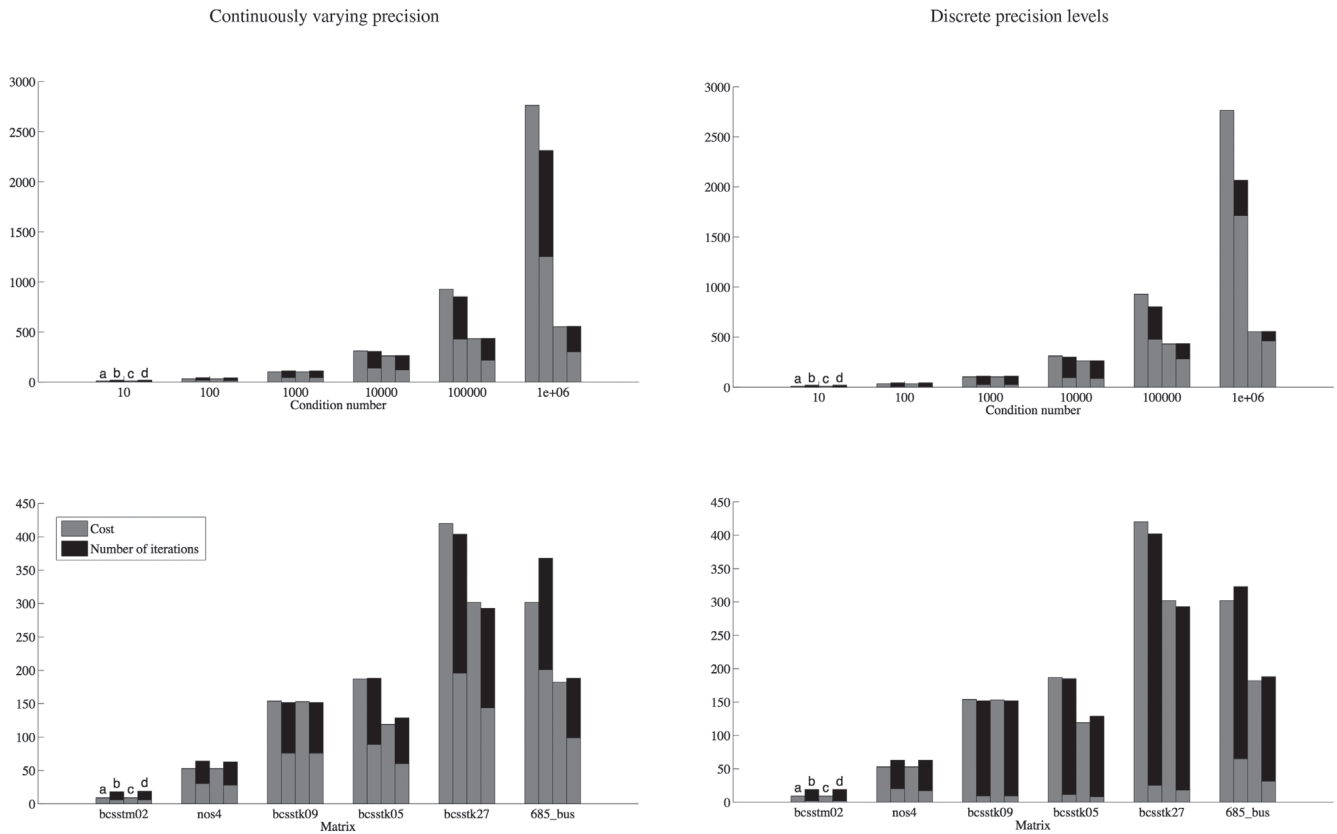


FIGURE 4 Number of iterations and associated costs for the continuously varying precision and discrete precision levels models. First row: synthetic matrices with varying logspace condition number. Second row: matrices from Matrix Market. (a) Double precision CG, (b) inexact practical CG, (c) double precision CG with reorthogonalization, (d) inexact practical CG with reorthogonalization. CG, conjugate gradients

of the practical iCG is lower than in double precision CG. The approximations involved in the stopping criterion (27) lead to early termination, which result in a solution error slightly larger than the targeted one ($16 \cdot 10^{-6}$ and $56 \cdot 10^{-6}$ instead of 10^{-5}). For condition numbers larger than 10^7 , both the double precision CG and inexact iCG reach the maximum number of iterations without having converged. Furthermore, the practical methods effectively provide significant gains in the cost of performing the matrix-vector products in CG, as highlighted in Table 2 and Figure 4. Finally, we note a slight increase in the number of iterations for the practical variants of iCG and iCGR at low to medium condition numbers, which is partly explained by the fact that the termination criterion is based on the delay d (10 in our case) to assess termination. The tuning of the parameter d is problem dependent and should be adapted to the condition number.

Similar conclusions hold for the discrete precision levels case. Furthermore, the results indicate that the management of the inaccuracy budget discussed in Section 3.1 is quite effective. We note that it leads to even more significant efficiency gains for moderately conditioned problems. The situation is, however, reversed for the more ill-conditioned cases, because ω then exceeds more quickly the accuracy threshold allowing single precision. While the small inaccuracy allowed by the bound can be exploited in the continuous case, this is no longer the case here and many products are computed in double precision.

4.3.2 | Examples from the matrix market

We conclude our experiments with some examples from the NIST Matrix Market. Properties of these matrices are given in Table 4. All of them are symmetric positive definite and result from discretization of PDEs. Again, in our computations, we use estimates of the smallest and largest eigenvalues obtained by perturbing the exact eigenvalues by a random relative perturbation of magnitude between 0 and 100%. We report the results of the corresponding runs in Tables 5

TABLE 4 Properties of the matrix market matrices (sorted by increasing condition number)

Matrix	Dimension	$\kappa_2(A)$	$\ A\ _2$
bcsstm02	66	8.8	0.17
nos4	100	1.5e03	0.85
bcsstk09	1083	9.5e03	6.8e07
bcsstk05	153	1.4e04	6.2e06
bcsstk27	1224	2.4e04	3.5e06
685_bus	685	4.2e05	2.6e04
nos1	237	2.0e07	2.5e09
nos7	729	2.4e09	9.9e06

TABLE 5 Matrix market: Practical algorithms in the continuously varying accuracy case

Method	Matrix	n_{it}	cost	r.res.gap	r.sol.err	r.val.err.	Matrix	n_{it}	cost	r.res.gap	r.sol.err	r.val.err.
CG	bcsstm02	9	9.0e+00	2.2e-32	2.2e-06	0.0e+00	nos4	53	5.3e+01	1.1e-28	2.3e-06	4.7e-11
CGR		9	9.0e+00	9.8e-32	2.2e-06	0.0e+00		53	5.3e+01	2.7e-28	2.3e-06	8.6e-15
iCG		18	5.0e+00	2.0e-08	2.0e-08	2.8e-05		64	2.9e+01	1.1e-12	4.0e-09	8.1e-07
iCGR		19	6.0e+00	3.0e-08	3.0e-08	2.8e-06		63	2.8e+01	1.3e-11	2.7e-09	1.2e-06
CG	bcsstk09	154	1.5e+02	4.3e-27	2.4e-06	1.7e-08	bcsstk05	187	1.9e+02	7.6e-28	2.5e-06	6.3e-08
CGR		153	1.5e+02	4.5e-27	2.5e-06	4.4e-14		119	1.2e+02	8.9e-28	2.2e-06	2.2e-14
iCG		152	7.6e+01	4.5e-13	2.8e-06	4.3e-07		188	8.8e+01	2.4e-11	1.0e-05	4.5e-06
iCGR		152	7.6e+01	2.3e-12	2.7e-06	5.8e-08		129	6.0e+01	2.0e-09	4.6e-09	8.0e-06
CG	bcsstk27	420	4.2e+02	2.1e-29	2.4e-06	2.4e-08	685_bus	302	3.0e+02	8.2e-27	2.5e-06	3.3e-08
CGR		302	3.0e+02	1.9e-27	2.4e-06	2.9e-15		182	1.8e+02	1.4e-26	2.3e-06	1.4e-14
iCG		404	2.0e+02	2.6e-12	7.7e-06	5.0e-08		368	2.0e+02	1.6e-13	4.8e-06	2.3e-07
iCGR		293	1.4e+02	4.0e-12	4.1e-06	7.5e-08		188	9.9e+01	3.9e-12	1.0e-06	4.3e-07
CG	nos1	711	7.1e+02	3.6e-23	3.1e-01	6.8e-07	nos7	1810	1.8e+03	1.3e-19	2.1e-06	2.6e-08
CGR		220	2.2e+02	2.5e-23	2.1e-06	6.1e-13		270	2.7e+02	2.9e-19	1.8e-06	4.0e-12
iCG		711	4.4e+02	2.2e-12	4.0e-01	7.3e-06		1031	7.1e+02	2.0e-09	1.0e-02	1.5e-05
iCGR		230	1.4e+02	1.9e-08	2.5e-08	2.5e-05		260	1.7e+02	8.3e-08	1.3e-05	2.1e-05

Abbreviation: CG, conjugate gradients.

(continuous-varying accuracy case) and 6 (discrete accuracy levels case), using the same conventions as for Table 1. The number of iterations and associated relative costs are also shown in Figure 4.

Similar comments to the synthetic cases can be made. Regarding the continuous-varying accuracy case, the practical iCG and iCGR tend to result in an increase of the iteration numbers compared with both the double precision CG and CGR, and the impractical methods with inaccurate matrix-vector products (not shown for the latter ones in Table 5 nor Figure 4). However, a tiny gain in iteration numbers may be obtained with the practical iCG and iCGR as observed for the matrix bcsstk27. Despite these increases in the iteration number, the costs of the practical methods remain lower than those of the methods in double precision. Again the practical algorithms seem more sensitive to rounding errors. For instance, the desired accuracy is not reached for matrices nos1 and nos7 without reorthogonalization. As observed in the synthetic matrices, it can result in a decrease in the number of iterations of the practical methods, and so a damage of the quality of the solution, compared with the methods with matrix-vector products in double precision. Finally, as for the synthetic matrices, the fact that the delay $d = 10$ introduced in the stopping criterion is too large partly explains the cost increases observed for matrices with a small condition number (bcsstm02 and nos4).

TABLE 6 Matrix market: Practical algorithms in the discrete precision levels case

Method	$\kappa(A)$	n_{it}	cost	r.res.gap	r.sol.err	r.val.err.	$\kappa(A)$	n_{it}	cost	r.res.gap	r.sol.err	r.val.err.
CG	bcsstm02	9	9.0e+00	2.2e-32	2.2e-06	0.0e+00	nos4	53	5.3e+01	1.1e-28	2.3e-06	4.7e-11
CGR		9	9.0e+00	9.8e-32	2.2e-06	0.0e+00		53	5.3e+01	2.7e-28	2.3e-06	8.6e-15
iCG		19	1.9e+00	2.4e-12	2.4e-12	8.7e-08		63	2.0e+01	5.3e-21	5.2e-09	6.8e-09
iCGR		19	1.9e+00	2.7e-12	2.7e-12	3.1e-08		63	1.7e+01	1.4e-20	2.7e-09	4.3e-11
CG	bcsstk09	154	1.5e+02	4.3e-27	2.4e-06	1.7e-08	bcsstk05	187	1.9e+02	7.6e-28	2.5e-06	6.3e-08
CGR		153	1.5e+02	4.5e-27	2.5e-06	4.4e-14		119	1.2e+02	8.9e-28	2.2e-06	2.2e-14
iCG		152	9.5e+00	2.9e-13	2.8e-06	4.0e-07		185	1.2e+01	1.2e-11	1.0e-05	2.9e-06
iCGR		152	9.5e+00	2.8e-12	2.7e-06	2.6e-07		129	8.1e+00	4.2e-11	2.7e-09	2.6e-06
CG	bcsstk27	420	4.2e+02	2.1e-29	2.4e-06	2.4e-08	685_bus	302	3.0e+02	8.2e-27	2.5e-06	3.3e-08
CGR		302	3.0e+02	1.9e-27	2.4e-06	2.9e-15		182	1.8e+02	1.4e-26	2.3e-06	1.4e-14
iCG		402	2.5e+01	6.5e-13	7.7e-06	4.3e-08		323	6.5e+01	7.5e-18	4.7e-06	4.6e-08
iCGR		293	1.8e+01	6.6e-13	4.1e-06	7.0e-08		188	3.1e+01	7.1e-16	1.0e-06	3.9e-10
CG	nos1	711	7.1e+02	3.6e-23	3.1e-01	6.8e-07	nos7	1810	1.8e+03	1.3e-19	2.1e-06	2.6e-08
CGR		220	2.2e+02	2.5e-23	2.1e-06	6.1e-13		270	2.7e+02	2.9e-19	1.8e-06	4.0e-12
iCG		711	4.5e+01	7.0e-15	3.9e-01	1.7e-06		957	2.1e+02	4.4e-15	3.0e-03	4.9e-08
iCGR		230	1.4e+01	2.5e-12	9.0e-09	6.1e-07		269	4.1e+01	9.6e-14	4.1e-06	1.3e-08

Abbreviation: CG, conjugate gradients.

Regarding the discrete precision levels case (see Table 6 and Figure 4), we note that it leads to even more significant efficiency gains for moderately conditioned problems and large efficiency gains for the ill-conditioned cases. This can be partly associated with the fact that these problems are easier than those obtained in the synthetic cases. However, it remains cases where the desired accuracy cannot be reached without reorthogonalization due to rounding errors. As a consequence, a good problem preconditioning is even more important in the multiprecision context.

5 | CONCLUSIONS

We have considered the iterative solution of convex quadratic optimization problems (1) and linear systems (2) using the CG algorithm with inaccurate matrix-vector products, with the aim of monitoring the decrease of the quadratic objective function. Circumventing the unavailability of some of the quantities involved in the theory, we have proposed estimates and derived a practical algorithm that use them. Our numerical experiments suggest that significant gains in energy efficiency can be achieved by the use of variable precision matrix-vector products. Such gains are most noticeable for problems that are reasonably well-conditioned, and occur both in the case where the accuracy of the products can be controlled continuously, and in the case where it is limited to discrete predefined levels. We have illustrated the latter in the important context of multiprecision computations. However, the potential speed-up in this context, as well as the gain in energy efficiency, are limited by the ability of the algorithm to run with precisions lower than the IEEE double precision. This depends both on the data (A, b) and the decrease in the quadratic that is targeted.

In view of the promising potential of this approach, it may be of interest to apply it in a more general context, for example, other optimization algorithms involving nonquadratic and possibly nonconvex objective functions. It is also worthwhile, in our opinion, to pursue experimentation with other methods beyond CG in the framework of multiprecision arithmetic.

While we focused our analysis on inexact matrix-vector products, a realistic assumption in large-scale applications where this product often involves the application of several complicated operators (see Reference 1, for example), the cost of inner-products involved in CG (and also potentially in reorthogonalization) may also be significant in some applications. Strategies to reduce this cost are therefore also of interest. It is not the purpose of this article to develop a rigorous analysis of CG with inexact inner products or reorthogonalization techniques, but we defer this analysis to a future contribution.

Finally, it would be interesting to investigate how this approach can be adapted to communication-avoiding algorithms like the s -step Krylov methods.^{33,34} In recent works on the s -step CG,³⁵ suggested a criterion for the adaptive selection of the parameter s (defining the size of the block of iterations) exploiting bounds on the residual gap. While combining both approaches would be of interest, this is out of the scope of the present work.

ACKNOWLEDGEMENTS

The authors thank the two anonymous referees for their constructive comments that leads to significant improvements in the presentation. Furthermore, the authors are indebted to Pr. S. Matsuoka (Riken) for an interesting conversation³¹ which confirmed their interest in multiprecision arithmetic in the context of very high-performance computing, to Pr A. Podobas (Tokyo Institute of Technology) for providing further pointers on computer architecture, and to Pr. M. Daydé (IRIT) for his continued and friendly support. Ph. Toint was partially supported by ANR-11-LABX-0040-CIMI within the program ANR-11-IDEX-0002-02.

ORCID

Ehouarn Simon  <https://orcid.org/0000-0003-3205-1649>

REFERENCES

1. Gratton S, Gürol S, Simon E, Toint PL. Guaranteeing the convergence of the saddle formulation for weakly-constrained 4D-VAR data assimilation. *QJR Meteorol Soc.* 2018;144:2592–2602.
2. Galal S, Horowitz M. Energy-efficient floating-point unit design. *IEEE Trans Comput.* 2011;60(7):913–922.
3. Pu J, Galal S, Yang X, Shacham O, Horowitz M. FPMAX: A 106GFLOPS/W at 217GFLOPS/mm² single-precision FPU, and a 43.7 GFLOPS/W at 74.6 GFLOPS/mm² double-precision FPU. 28nm UTBB FDSOI. *Hardware Architecture*; 2016.
4. Carson E, Higham NJ. Accelerating the solution of linear systems by iterative refinement in three precisions. *SIAM J Sci Comput.* 2018;40(2):A817–A847.
5. Fasi M, Higham NJ. Multiprecision algorithms for computing the matrix logarithm. *SIAM J Matrix Anal Appl.* 2018;39(1):472–491.
6. Haidar A, Tomov S, Dongarra J, and Higham NJ. Harnessing GPU tensor cores for fast FP16 arithmetic to speed up mixed-precision iterative refinement solvers. *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis. SC '18 (Dallas, TX). Piscataway, NJ, IEEE Press; 2018. p. 47:1–47:11.*
7. Higham NJ, Pranesh S, Zounon M. Squeezing a matrix into half precision, with an application to solving linear systems. *SIAM J Sci Comput.* 2019;41(4):A2536–A2551.
8. Higham NJ, Pranesh S. Simulating low precision floating-point arithmetic. *SIAM J Sci Comput.* 2019;41(5):C585–C602.
9. Hestenes MR, Stiefel E. Methods of conjugate gradients for solving linear systems. *J Nat Bureau Stand.* 1952;49:409–436.
10. Simoncini V, Szyld D. Theory of inexact Krylov subspace methods and applications to scientific computing. *SIAM J Sci Comput.* 2003;25(2):454–477.
11. van den Eshof J, Sleijpen GLG. Inexact Krylov subspace methods for linear systems. *SIAM J Matrix Anal Appl.* 2004;26(1):125–153.
12. van den Eshof J, Sleijpen GLG, van Gijzen MB. Relaxation strategies for nested Krylov methods. *J Comput Appl Math.* 2005;177:347–365.
13. Gratton S, Toint PL, Tshimanga J. Range-space variants and inexact matrix-vector products in Krylov solvers for linear systems arising from inverse problems. *SIAM J Matrix Anal.* 2011;32(3):969–986.
14. Du X, Haber E, Karampataki M, Szyld DB. Varying iteration accuracy using inexact conjugate gradients in control problems governed by PDE's. *Proceedings of the 2nd Annual International Conference on Computational Mathematics, Computational Geometry and Statistics (CMCGS 2013). Singapore, Global and Technology Forum; 2013. p. 29–38.*
15. Simoncini V, Szyld DB. Recent computational developments in Krylov subspace methods for linear systems. *Numer Linear Algebra Appl.* 2007;14:1–59.
16. Conn AR, Gould NIM, Toint PL. Trust-region methods. MPS-SIAM series on optimization. Philadelphia, PA: SIAM, 2000.
17. Arioli M, Duff IS, Ruiz D. Stopping criteria for iterative solvers. *SIAM J Matrix Anal Appl.* 1992;13(1):138–144.
18. Arioli M. A stopping criterion for the conjugate gradient algorithm in a finite element framework. *Numerische Mathematik.* 2004;97:1–24.
19. Arioli M, Loghin D, Wathen A. Stopping criteria for iterations in finite element methods. *Numerische Mathematik.* 2004;99(3):381–410.
20. Arioli M. Generalized Golub-Kahan bidiagonalization and stopping criteria. *SIAM J Matrix Anal.* 2013;34(2):571–592.
21. Arioli M, Liesen J, Midelar A, Strakos Z. Interplay between discretization and algebraic computation in adaptive numerical solution of elliptic PDE problems. *GAMM Mitteilungen.* 2013;36:102–119.
22. Jiranek P, Strakos Z, Vohralik M. A posteriori error estimates including algebraic error and stopping criteria for iterative solvers. *SIAM J Sci Comput.* 2010;32:1567–1590.
23. Papez J, Liesen J, Strakos Z. Distribution of the discretization and algebraic error in numerical solution of partial differential equations. *Linear Algebra Appl.* 2014;449:89–114.
24. Golub GH, Ye Q. Inexact preconditioned conjugate gradient method with inner-outer iteration. *SIAM J Sci Comput.* 1999;21:1305–1320.
25. Strakos Z, Tichy P. On error estimation in the conjugate gradient method and why it works in finite precision computations. *Electr Trans Numer Anal.* 2002;13:56–80.

26. Meurant G, Strakos Z. The Lanczos and conjugate gradient algorithms in finite precision arithmetic. *Acta Numerica*. 2006;15:471–542.
27. Jiranek P, Tittley-Peloquin D. Estimating the backward error in LSQR. *SIAM J Matrix Anal Appl*. 2010;31:2055–2074.
28. Arioli M, Gratton S. Linear regression models, least-squares problems, normal equations, and stopping criteria for the conjugate-gradient method. *Comput Phys Commun*. 2012;183:2322–2336.
29. Liesen J, Strakos Z. *Krylov subspace methods, principles and analysis*. Oxford, UK: Oxford University Press, 2000.
30. Notay Y. Flexible conjugate gradients. *SIAM J Sci Comput*. 2000;22:1444–1460.
31. Matsuoka S. Private communication; 2018.
32. Higham NJ. The rise of multiprecision computations, SAMSI 2017; April 2017. <https://bit-ly/higham-samsi2017>.
33. John van Rosendale. Minimizing inner product data dependencies in conjugate gradient iteration. *International Conference on Parallel Processing, ICPP'83*, Columbus, Ohio: IEEE Computer Society; 1983;44–46.
34. Chronopoulos A, Gear C. s-step iterative methods for symmetric linear systems. *J Comput Appl Math*. 1989;25:.
35. Carson E. The adaptive s-step conjugate gradient method. *SIAM J Matrix Anal Appl*. 2018;39(3):1318–1338.

How to cite this article: Gratton S, Simon E, Tittley-Peloquin D, Toint PL. Minimizing convex quadratics with variable precision conjugate gradients. *Numer Linear Algebra Appl*. 2020;e2337. <https://doi.org/10.1002/nla.2337>