Minimizing convex quadratics with variable precision Krylov methods

Gratton, Serge; Simon, Ehouarn; Toint, Philippe

Published in:
Numerical Linear Algebra with Applications

Publication date:
2020

Document Version
Early version, also known as pre-print

Minimizing convex quadratics with variable precision Krylov methods

S. Gratton [*], E. Simon [†], and Ph. L. Toint [‡]

11 July 2018

Abstract

Iterative algorithms for the solution of convex quadratic optimization problems are investigated, which exploit inaccurate matrix-vector products. Theoretical bounds on the performance of a Conjugate Gradients and a Full-Orthormalization methods are derived, the necessary quantities occurring in the theoretical bounds estimated and new practical algorithms derived. Numerical experiments suggest that the new methods have significant potential, including in the steadily more important context of multi-precision computations.

Keywords: quadratic optimization, positive-definite linear systems, variable accuracy, multi-precision arithmetic, high-performance computing.

1 The problem

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

$$\min_{x\in\mathbb{R}^n} q(x) \stackrel{\rm def}{=} \tfrac{1}{2}x^T Ax - b^T x \tag{1.1}$$

where A is an $n \times n$ symmetric positive definite matrix and b a given vector in $\mathbb{R}^n$. Such problems are at the centre of efficient methods for a large variety of domains in applied mathematics, the most proeminent being large-scale numerical nonlinear optimization and the solution of large symmetric positive-definite linear systems (often derived from applications in partial differential equations). It is thus critical to make the solution of (1.1) as efficient as possible, especially when the problem size grows. Since the cost of most iterative methods for solving (1.1) is often dominated by the (potentially many) computations of products of the form Ap for some vector p, it is then of interest to investigate if efficiency gains may be obtained for this 'core' operation. This is the object of this paper.

Two different but converging contexts motivate the analysis presented here. The first is the frequent occurence of problems involving a hierachy of model representations themselves

[*]Université de Toulouse, INP, IRIT, Toulouse, France. Email: serge.gratton@enseeiht.fr

[†]Université de Toulouse, INP, IRIT, Toulouse, France. Email: ehouarn.simon@enseeiht.fr

[‡]NAXYS, University of Namur, Namur, Belgium. Email: philippe.toint@unamur.be. Partially supported by ANR-11-LABX-0040-CIMI within the program ANR-11-IDEX-0002-02.

resulting in the ability to use approximate versions of A to compute the product Ap. This occurs for instance in discretized applications, possibly in a multi-resolution framework, or in inexactly weighted linear and nonlinear least-squares where the product itself is obtained by applying an iterative procedure[(1)]. The second is the increasing importance of computations in multi-precision arithmetic on the new generations of high-performance computers (see [12] and the many references therein), in which the use of varying levels of accuracy is seen as a key ingredient in obtaining state-of-the-art energy-efficient computer architectures. In both cases, using 'inexact' products Ap (while controlling their inexactness) within Krylov-based iterative methods is natural option. In what follows, we focus on the analysis of this choice from the point of view of ensuring a prescribed decrease in the objective function q.

Although using inexact product in Krylov-based iterative methods has already been investigated (see [17, 18, 19, 10] for example), the proposed analysis typically focus on the Euclidean norm of the residual but none of them, to the best of our knowledge, considers its effect on assessing decrease in the objective function of the associated optimization problem. This point of view is however important. In optimization, monitoring the evolution of the 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 or inexact Newton methods. In applications arising from elliptic partial differential applications, several authors have argued that monitoring the energy norm of the error leads to better subproblem termination rules, avoiding under- or over-solving (see [1, 3, 5, 2]).

As it turns out, monitoring the residual of the linear system

$$Ax = b \tag{1.2}$$

provides a handle for monitoring the error in the quadratic q, provided it is considered in the appropriate norm. Indeed, if x_* is the solution of (1.2) and $r(x) \stackrel{\rm def}{=} Ax - b$, then

$$\begin{array}{rcl} \frac{1}{2}\|r(x)\|^2_{A^{-1}} & = & \frac{1}{2}(Ax-b)^TA^{-1}(Ax-b) \\ & = & \frac{1}{2}(x-x_*)^TA(x-x_*) \\ & = & \frac{1}{2}[x^TAx - 2x^TAx_* + x_*^TAx_*] \\ & = & q(x) - q(x_*). \end{array} \tag{1.3}$$

This approach however requires first that $r(x)$ or a sufficiently good approximation thereof is available and, second, that its A^{-1}-norm can be estimated, both of which are non-trivial if the products Ap are inexact.

The contributions of this paper are the derivation of theoretical residual bounds ensuring that the error on $\|r(x)\|_{A^{-1}}$ remains suitably small in the presence of inexact products (Section 2), the traduction of these results into computable estimates and the definition of the associated algorithms (Section 3), and the demonstration, in Section 4, that very significant efficiency gains can be obtained by the use of these algorithms, both in the case where the accuracy of Ap can be varied continuously and in the case where it is bound to prescribed levels (as is the case in multi-precision arithmetic). We finally provide some conclusions and perspectives in Section 5.

Notations. In the sequel of this paper, $\|\cdot\|_2$ denotes the standard Euclidean norm for vectors and its induced (spectral) norm for matrices. In addition, if M is symmetric positive

[(1)] Our starting point was a nonlinear weighted least-squares problem occuring in large-scale data assimilation for weather forecasting [9], where the inverse of the weighting matrix can not be computed. It use thus requires the iterative solution of an innermost linear system.

definite and x is a vector, $\|x\|_M = \|M^{1/2}x\|_2$. The dual norm of $\|.\|_M$ with respect to the Euclidean inner product is the norm $\|.\|_{M^{-1}}$. $\mathrm{Tr}(M)$ is the trace of the matrix M and e_i is the i-th vector of the canonical basis of $\mathbb{R}^n$.

2 A rigorous error analysis

Standard Krylov-based methods (we will consider here the full orthonormalization (FOM) method [16] and the conjugate-gradients (CG) algorithm [11] with and without reorthogonalization, both initialized with the zero vector, i.e. $x_0 = 0$ and $r_0 = -b$) do provide recurrences for the residual $r_k = r(x_k)$, where x_k is the approximate solution at iteration k. However, these recurrences rely on the assumption that the products Ap computed in the course of the FOM or CG iterations are exact. In our context, where we aim at using inexact products, we therefore need to bound the *residual gap*, that is the difference between the residuals r_k within FOM or CG at iteration k and the true residual $r(x_k)$, where x_k is the approximate solution at this iteration. If we manage to do this and, at the same time, make the computed residual r_k small (as is typical in the application of FOM or CG), we therefore obtain the following desirable property.

Lemma 2.1 Suppose that, at some iterate x_k of either FOM or CG, one has that for any r_k

$$\max\left[\|r_k - r(x_k)\|_{A^{-1}}, \|r_k\|_{A^{-1}}\right] \leq \frac{\sqrt{\epsilon}}{2}\|b\|_{A^{-1}}. \tag{2.1}$$

Then

$$|q(x_k) - q(x_*)| \leq \epsilon |q(x_*)| \tag{2.2}$$

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

$$|q(x_*)| = -q(x_*) = \tfrac{1}{2}\|b\|^2_{A^{-1}} = \tfrac{1}{2}\|x_*\|^2_A = \tfrac{1}{2}|b^T x_*|. \tag{2.3}$$

Using this identity, (1.3), the triangle inequality and (2.1) we deduce that

$$\begin{aligned} |q(x_k) - q(x_*)| &= \tfrac{1}{2}\|r(x_k)\|^2_{A^{-1}} \\ &\leq \tfrac{1}{2}\left(\|r(x_k) - r_k\|_{A^{-1}} + \|r_k\|_{A^{-1}}\right)^2 \\ &\leq \tfrac{1}{2}\left(\sqrt{\epsilon}\|b\|_{A^{-1}}\right)^2 \\ &= \tfrac{1}{2}\epsilon\|b\|^2_{A^{-1}} \\ &= \epsilon|q(x_*)|. \end{aligned} \tag{2.4}$$

□

Thus the decrease in the quadratic q obtained at x_k is at least $(1-\epsilon)$ times the maximum obtainable decrease. This is exactly the type of result we wish if we are to terminate the quadratic minimization, for instance in a trust-region context (see [6, Theorem 6.3.5]). Because we expect FOM or CG to make r_k small eventually, the rest of the section is now devoted to analyzing how to enforce the part of (2.1) related to the residual gap, that is the condition that $\|r_k - r(x_k)\|_{A^{-1}} \leq \frac{\sqrt{\epsilon}}{2}\|b\|_{A^{-1}}$.

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 to consider the primal-dual matrix norm defined by

$$\|E\|_{A^{-1},A} \stackrel{\rm def}{=} \sup_{x\neq 0} \frac{\|Ex\|_{A^{-1}}}{\|x\|_A} = \|A^{-1/2}EA^{-1/2}\|_2. \tag{2.5}$$

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

$$\|A\|_{A^{-1},A} = 1. \tag{2.6}$$

We first derive a useful observation.

Lemma 2.2 Let A be a symmetric and positive definite matrix and E be any symmetric perturbation. Then, if $\|E\|_{A^{-1},A} < 1$, the matrix $A + E$ is symmetric positive definite.

Proof. Assume that $\|E\|_{A^{-1},A} < 1$. From the definition of the norm $\|\cdot\|_{A^{-1},A}$ we deduce that

$$\|E\|^2_{A^{-1},A} = \sup_{x\neq 0} \frac{x^TE^TA^{-1}Ex}{x^TAx} = \sup_{u\neq 0} \frac{u^TA^{-1/2}E^TA^{-1}EA^{-1/2}u}{u^Tu} < 1, \tag{2.7}$$

which shows that $\|A^{-1/2}EA^{-1/2}\|_2 < 1$. Using

$$\lambda_{\min}(I + A^{-1/2}EA^{-1/2}) \geq \lambda_{\min}(I) - |\lambda_{\max}(A^{-1/2}EA^{-1/2})| > 0,$$

we obtain that $I + A^{-1/2}EA^{-1/2}$ is symmetric and positive definite. Sylvester's inertia theorem then yields that $A^{1/2}(I + A^{-1/2}EA^{-1/2})A^{1/2}$ is symmetric and positive definite, which completes the proof. □

2.1 Inexact FOM

We first focus on deriving conditions on the FOM algorithm for which the analysis is simpler. A first version of the inexact FOM algorithm (initialized with the zero vector) can be stated as follows.

Theoretical inexact FOM algorithm

1. Set $\beta = \|b\|_2$, and $v_1 = [b/\beta]$,
2. For k=1, 2, . . . , do
3. $w_k = (A + E_k)v_k$
4. For $i = 1, \ldots, k$ do
5. $h_{i,k} = v_i^Tw_k$
6. $w_k = w_k - h_{i,k}v_i$
7. EndFor
8. $h_{k+1,k} = \|w_k\|_2$
9. $y_k = H_k^{-1}(\beta e_1)$
10. if $|h_{k+1,k}e_k^Ty_k|$ is small enough then go to 13
11. $v_{k+1} = w_k/h_{k+1,k}$
12. EndFor
13. $x_k = V_ky_k$

In this description, we use the notation $H_k = [h_{i,j}]_{i,j=1}^{k}$. It is also convenient to define

$$\tilde{H}_k = \begin{bmatrix} H_k \\ h_{k+1,k} e_k^T \end{bmatrix} \quad \text{and} \quad V_k = [v_1, \ldots, v_k].$$

It is well-known [16, 13] that both H_k and $\tilde{H}_k$ are upper Hessenberg matrices and that, if $E_k = 0$ for all k, the Arnoldi relation

$$AV_k y_k = V_{k+1} \tilde{H}_k y_k \tag{2.8}$$

holds. Note that, again for $E_k = 0$, the approximation x_k, the residual r_k and q_k, the estimated value of the quadratic objective at x_k, are not directly recurred by FOM, but can easily be retrieved from

$$x_k = V_k y_k, \quad r_k = V_{k+1} \tilde{H}_k y_k - b = Ax_k - b = r(x_k) \quad \text{and} \quad q_k = -\tfrac{1}{2} b^T x_k = q(x_k). \tag{2.9}$$

Note that, unless no error is made in the products, q_k might differ from $q(x_k)$, just as r_k may differ from $r(x_k)$. Also note that, because of Step 9 of the algorithm, $\|r_{k+1}\|_2 = |h_{k+1,k} e_k^T y_k|$ and thus Step 10 branches out of the outer loop as soon as $\|r_{k+1}\|_2$ is small enough. When the error matrices E_k are nonzero, it is easy to verify that (2.8) becomes

$$AV_k y_k + G_k V_k y_k = V_{k+1} \tilde{H}_k y_k \tag{2.10}$$

where

$$G_k = (E_1 v_1, \ldots, E_k v_k)\, V_k^T = \sum_{j=1}^{k} E_j v_j v_j^T. \tag{2.11}$$

We now derive conditions on the error matrices E_j which ensure that the relative residual gap (measured in dual norm) remains suitably small.

Lemma 2.3 Let $\epsilon_\pi > 0$ and let $\phi \in \mathbb{R}^k$ be a positive vector such that

$$\sum_{j=1}^{k} \frac{1}{\phi_j} \leq 1. \tag{2.12}$$

Suppose furthermore that

$$\|E_j\|_{A^{-1},A} \leq \omega_j^{\text{FOM}} \stackrel{\text{def}}{=} \min \left[1, \frac{\epsilon_\pi \|b\|_{A^{-1}}}{\phi_j \|v_j\|_A \|H_k^{-1}\|_2 \|r_{j-1}\|_2} \right] \quad \text{for all} \quad j \in \{1, \ldots, k\}. \tag{2.13}$$

Then

$$\|r(x_k) - r_k\|_{A^{-1}} \leq \epsilon_\pi \|b\|_{A^{-1}}.$$

Proof. Using (2.11), (2.10), (2.9) and the triangle inequality, we find that

$$\|r(x_k) - r_k\|_{A^{-1}} = \|G_k V_k y_k\|_{A^{-1}} = \|\sum_{j=1}^{k} E_j v_j e_j^T y_k\|_{A^{-1}} \leq \sum_{j=1}^{k} \|E_j\|_{A^{-1},A} \|v_j\|_A |e_j^T y_k|. \tag{2.14}$$

Furthermore,

$$e_j^T y_k = e_j^T H_k^{-1}(\beta e_1)$$

because of the definition of y_k in Step 9 of FOM. Introducing $y_{j-1}^0 = [y_{j-1}, 0]^T \in \mathbb{R}^k$ and $\tilde{H}_{j-1}^0 = \begin{bmatrix} \tilde{H}_{j-1} & 0 \\ 0 & 0 \end{bmatrix} \in \mathbb{R}^{k\times k}$, one has that

$$e_j^T H_k^{-1} \tilde{H}_{j-1}^0 y_{j-1}^0 = e_j^T H_k^{-1} H_k y_{j-1}^0 = 0.$$

It then follows that

$$e_j^T y_k = e_j^T H_k^{-1}(\beta e_1 - \tilde{H}_{j-1}^0 y_{j-1}^0) = e_j^T H_k^{-1} \begin{bmatrix} \beta\tilde{e}_1 - \tilde{H}_{j-1} y_{j-1} \\ 0 \end{bmatrix}$$

where $\tilde{e}_1$ is the first vector of the canonical basis of $\mathbb{R}^j$. Thus, using the second part of (2.9),

$$\begin{aligned} e_j^T y_k &= e_j^T H_k^{-1} \begin{bmatrix} V_j^T(\beta V_j \tilde{e}_1 - V_j \tilde{H}_{j-1} y_{j-1}) \\ 0 \end{bmatrix} \\ &= e_j^T H_k^{-1} \begin{bmatrix} V_j^T(b - V_j \tilde{H}_{j-1} y_{j-1}) \\ 0 \end{bmatrix} \\ &= -e_j^T H_k^{-1} \begin{bmatrix} V_j^T r_{j-1} \\ 0 \end{bmatrix}. \end{aligned}$$

The Cauchy-Schwarz inequality then implies that

$$|e_j^T y_k| \leq \|e_j\|_2 \|H_k^{-1}\|_2 \|V_j^T r_{j-1}\|_2 \leq \|H_k^{-1}\|_2 \|r_{j-1}\|_2$$

because the columns of V_j are orthonormal. Thus we deduce from (2.14) and (2.13) that

$$\frac{\|r(x_k)_k - r_k\|_{A^{-1}}}{\|b\|_{A^{-1}}} \leq \|H_k^{-1}\|_2 \frac{\sum_{j=1}^k \omega_j \|v_j\|_A \|r_{j-1}\|_2}{\|b\|_{A^{-1}}}.$$

Now the definition of $\omega_j^{\rm FOM}$ in (2.13) gives that

$$\frac{\|r(x_k) - r_k\|_{A^{-1}}}{\|b\|_{A^{-1}}} \leq \epsilon_\pi \sum_{j=1}^k \frac{1}{\phi_j} \leq \epsilon_\pi,$$

as requested. □

We now combine all the above results in a single theorem.

Theorem 2.4 Let $\epsilon > 0$ and suppose that, at iteration $k > 0$ of the FOM algorithm,

$$\|r_k\|_{A^{-1}} \leq \tfrac{1}{2}\sqrt{\epsilon}\, \|b\|_{A^{-1}} \tag{2.15}$$

and the product error matrices E_j satisfy (2.13) with $\epsilon_\pi = \frac{1}{2}\sqrt{\epsilon}$ for some positive vector $\phi \in \mathbb{R}^k$ satisfying (2.12). Then (2.2) holds.

Proof. Directly results from Lemmas 2.1 and 2.3. □

2.2 Inexact CG

We now turn to the conjugate gradient algorithm and derive an analog of Theorem 2.4 for this case. We first state the algorithm itself.

Theoretical inexact CG algorithm	
1.	Set $x_0 = 0$, $\beta_0 = \|b\|_2^2$, $r_0 = -b$ and $p_0 = r_0$
2.	For k=0, 1, ..., do
3.	$c_k = (A + E_k)p_k$
4.	$\alpha_k = \beta_k / p_k^T c_k$
5.	$x_{k+1} = x_k + \alpha_k p_k$
6.	$r_{k+1} = r_k + \alpha_k c_k$
7.	if r_{k+1} is small enough then stop
8.	$\beta_{k+1} = r_{k+1}^T r_{k+1}$
9.	$p_{k+1} = -r_{k+1} + (\beta_{k+1}/\beta_k)p_k$
10.	EndFor

Observe that, at variance with FOM, the values of the iterates x_k and recurred residuals r_k are explicitly available. The value of q_k may again be retrieved from (2.9).

We next restate, for clarity, a simple result relating the residual gap to the error matrices for the conjugate gradient algorithm (see [18]).

Lemma 2.5 The residual gap in the inexact CG satisfies

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

Proof. We proceed inductively. Observe that $r(x_0) - r_0 = 0$ and $r(x_1) - r_1 = -\alpha_0 E_0 p_0$. Suppose now that the result is true for iterations $1, \ldots, k$. We then have that

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

□

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

Lemma 2.6 Let $\epsilon_\pi > 0$ and let $\phi \in \mathbb{R}^k$ be a positive vector satisfying (2.12). Suppose furthermore that

$$\|E_j\|_{A^{-1},A} \leq \omega_j^{\rm CG} \stackrel{\rm def}{=} \frac{\epsilon_\pi \|b\|_{A^{-1}} \|p_j\|_A}{\phi_{j+1}\|r_j\|_2^2 + \epsilon_\pi \|b\|_{A^{-1}} \|p_j\|_A}, \quad \text{for all} \quad j \in \{0, \ldots, k-1\}. \quad (2.16)$$

Then

$$\|r(x_k) - r_k\|_{A^{-1}} \leq \epsilon_\pi \|b\|_{A^{-1}}.$$

Proof. First note that (2.16) ensures that $\omega_j^{\rm CG} \in (0,1)$. Lemma 2.5, the triangle inequality and (2.5) imply that

$$\|r(x_k) - r_k\|_{A^{-1}} \leq \sum_{j=0}^{k-1} \|\alpha_j E_j p_j\|_{A^{-1}} \leq \sum_{j=0}^{k-1} |\alpha_j| \|E_j\|_{A^{-1},A} \|p_j\|_A \leq \sum_{i=0}^{k-1} |\alpha_j| \omega_j^{\rm CG} \|p_j\|_A. \quad (2.17)$$

Now, using (2.16),

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

Substituting this bound in (2.17) and using (2.16) again, we obtain that

$$\|r(x_k) - r_k\|_{A^{-1}} \leq \sum_{j=0}^{k-1} \frac{\omega_j^{\rm CG}}{1-\omega_j^{\rm CG}} \frac{\|r_j\|_2^2}{\|p_j\|_A}. \quad (2.18)$$

But the definition of ω_j in (2.16) gives that

$$\frac{\omega_j^{\rm CG}}{1-\omega_j^{\rm CG}} = \frac{(\epsilon_\pi \|b\|_{A^{-1}} \|p_j\|_A)(\phi_{j+1}\|r_j\|_2^2 + \epsilon_\pi \|b\|_{A^{-1}} \|p_j\|_A)}{(\phi_{j+1}\|r_j\|_2^2 + \epsilon_\pi \|b\|_{A^{-1}} \|p_j\|_A)\phi_{j+1}\|r_j\|_2^2} = \frac{\epsilon_\pi \|b\|_{A^{-1}} \|p_j\|_A}{\phi_{j+1}\|r_j\|_2^2},$$

so that (2.18) becomes

$$\|r(x_k) - r_k\|_{A^{-1}} \leq \sum_{j=0}^{k-1} \frac{\epsilon_\pi \|b\|_{A^{-1}}}{\phi_{j+1}} \leq \epsilon_\pi \|b\|_{A^{-1}}. \quad (2.19)$$

□

As above, we summarize the results for the CG algorithm.

Theorem 2.7 Let $\epsilon > 0$ and suppose that (2.15) holds at iteration $k > 0$ of the CG algorithm, and that the product error matrices E_j satisfy (2.16) with $\epsilon_\pi = \frac{1}{2}\sqrt{\epsilon}$ for some positive vector $\phi \in \mathbb{R}^k$ satisfying (2.12). Then (2.2) holds.

Some comments are in order at this stage.

1. Both (2.13) and (2.16) assume 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, it makes these requirements difficult to verify as such in other cases.

2. Even discounting that potential difficulty, (2.13) and (2.16) remain unfortunately impractical, since they involve quantites, such as $\|b\|_{A^{-1}}$, $\|v_j\|_A$ or $\|p_j\|_A$, which cannot be computed readily in the course of the FOM or CG algorithm. Moreover, $\|r_k\|_{A^{-1}}$ in (2.15) is also unavailable in practice.

3. Observe that (2.13) allows a growth of the error in $\|r_j\|^{-1}$ while (2.16) allows a growth of the order of $\|r_j\|^{-2}\|p_j\|_A$ instead.

4. In (2.13), the minimum with one is taken to ensure that the relative error remains meaningful (while at the same time allowing to apply Lemma 2.2). This guarantee is automatic in (2.16).

5. The ϕ_j appearing in (2.13) and (2.16) may be considered as an "error management strategy", in that they potentially allow for mitigating the direct effect of the residual norm in the definition of the error bound. A simple choice is to define $\phi_j = n$ for all j, which obviously satisfies (2.12). We will show below that they can be used to further advantage.

2.3 Bounding the error of the computed quadratic value

Neither the FOM or CG algorithm updates recursively the value of the quadratic objective function at the current iterate. However, tracking this value may be important in practice (as we will see in Section 3), and, if there is no error in the products Ap, can be done easily without computing additional products with A. Indeed, since x_k is the minimizer of $q(\cdot)$ in the direction x_k, one verifies that

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

In the presence of errors on Ap, this property may no longer hold, and it is of interest to analyze how much $q_k = -\frac{1}{2}b^T x_k$ differs from $q(x_k)$. This is 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 this decrease is a key ingredient in the management of the trust-region radius (see [6, Chapter 6]).

In order to provide an error estimate, we first prove the following backward error property.

Lemma 2.8 Let A be a symmetric positive-definite $n \times n$ matrix and b a vector in $\mathbb{R}^n$. Then, for any $x \in \mathbb{R}^n$,

$$\min_E \left\{ \frac{\|E\|_{A^{-1},A}}{\|A\|_{A^{-1},A}} \mid x^T(A+E)x = x^T b \right\} = \frac{|x^T r(x)|}{\|x\|_A^2}. \tag{2.20}$$

Proof. Using (2.5) and (2.6), we rewrite the optimization problem as

$$\min_E \left\{ \|A^{-1/2} E A^{-1/2}\|_2 \mid x^T E x = x^T (b - Ax) \right\}$$

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

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

But, for any symmetric matrix M and scalar γ,

$$\min_M \left\{ \|M\|_2 \mid y^T M y = \gamma \right\} = \gamma$$

is attained for $M = \gamma y y^T / \|y\|_2^4$. Thus

$$E = -(x^T r(x)) \frac{A x x^T A}{\|x\|_A^4} \quad \text{and} \quad \|E\|_{A^{-1},A} = \|A^{-1/2} E A^{-1/2}\|_2 = \frac{|x^T r(x)|}{\|x\|_A^2}.$$

□

Consider x the result of applying the FOM or CG methods with inexact products, such that (2.15) and and either (2.13) or (2.16) holds. We deduce from the preceding lemma that there exists a quadratic

$$\hat{q}(x) = \tfrac{1}{2} x^T (A + E) x - b^T x \tag{2.21}$$

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

$$\begin{aligned} |q(x) - q| &= |q(x) - \hat{q}(x)| \\ &= \tfrac{1}{2} |x^T E x| \\ &\leq \tfrac{1}{2} \|E\|_{A^{-1},A} \|x\|_A^2 \\ &= \tfrac{1}{2} |x^T r(x)| \\ &\leq \tfrac{1}{2} \frac{\|r(x)\|_{A^{-1}}}{\|x_*\|_A} \|x_*\|_A \|x\|_A, \end{aligned} \tag{2.22}$$

we successively used (2.21), the Cauchy-Schwarz inequality, Lemma 2.8, the Cauchy-Schwarz inequality again and (2.3). But, using the triangle inequality, Lemma 2.3 or (2.19), (2.3) and (2.15), we obtain that

$$\begin{aligned} \|x\|_A &\leq \|x_*\|_A + \|x - x_*\|_A \\ &= \|x_*\|_A + \|r(x)\|_{A^{-1}} \\ &\leq \|x_*\|_A + \|r(x) - r\|_{A^{-1}} + \|r\|_{A^{-1}} \\ &\leq (1 + 2\epsilon_\pi) \|x_*\|_A \end{aligned}$$

and hence, using (2.15) and (2.3) again,

$$|q(x) - q| \leq \tfrac{1}{2} \frac{\|r(x)\|_{A^{-1}}}{\|x\|_A} \|x\|_A^2 \leq 2\epsilon_\pi (1 + 2\epsilon_\pi) |q(x_*)|.$$

We summarize the above discussion in the following theorem.

Theorem 2.9 Let x be the result of applying the FOM or CG algorithm with inexact products and suppose that (2.15) and either (2.13) or (2.16) holds with $\epsilon_\pi = \frac{1}{2}\sqrt{\epsilon}$. Then

$$\frac{|q(x) - q|}{|q(x_*)|} \leq \sqrt{\epsilon}(1 + \sqrt{\epsilon}). \tag{2.23}$$

Remembering that $\epsilon_\pi = \frac{1}{2}\sqrt{\epsilon}$, we see that this bound is considerably weaker than (2.2) but is likely to be pessimistic, as we have not taken into account in (2.22) the fact that the angle between x and $r(x)$ is expected to be small. This will be numerically confirmed in Section 4.

3 Computable bounds and practical inexact algorithms

The first two comments at the end of Section 2.2 might, at first sight, suggest that the above approach has to remain of purely theoretical interest. However, the potential of allowing inexact products revealed by (2.13) and (2.16) prompts us to estimate the unavailable quantities, from which practical algorithms may then be derived. We consider these questions in this section, and demonstrate in the next section that substantial benefits may result.

3.1 Managing the inaccuracy budget

An important ingredient of a computable inexact FOM or CG algorithm is the choice of ϕ in (2.13) or (2.16). We note that (2.12) limits the sum of the inverse of ϕ_j over all iteration *preceding termination*. Of course, choosing $\phi_j = n$ is adequate (assuming FOM or CG will not use more than n iterations!), but is often suboptimal. Indeed, the expected number of iterations to termination is very commonly, for not too-ill-conditioned or correctly preconditioned problems, much lower than the problem's dimension. This can be exploited to make the ϕ_j smaller, for instance by choosing ϕ_j equal to this expected iteration number. This last number may be obtained from the maximum of iterations for FOM or CG ($k_{\max}^{\text{user}}$) typically imposed by the user, or from

$$k_{\max}^{\text{spectral}} = \frac{\log(\epsilon)}{\log(\rho)} \quad \text{where} \quad \rho \stackrel{\text{def}}{=} \frac{\sqrt{\lambda_{\max}/\lambda_{\min}} - 1}{\sqrt{\lambda_{\max}/\lambda_{\min}} + 1} \tag{3.1}$$

where $\lambda_{\min}$ and $\lambda_{\max}$ approximate $\lambda_{\min}(A)$ and $\lambda_{\max}(A)$, respectively (see [8, Theorem 10.2.6] for example). Moreover, it is possible to adjust the ϕ_j adaptively in the case where the user is able to specify a bound on $\|E_j\|_{A^{-1},A}$ when it is smaller than ω_j. As we will see below, this typically happens when the accuracy of the product Ap cannot be chosen continuously, but is bound to prescribed levels (such as arithmetic precision). Suppose therefore that, besides the product $(A + E)p$, one knows a value $\hat{\omega}_j \leq \omega_j$ such that $\|E_j\|_{A^{-1},A} \leq \hat{\omega}_j$. We may then decide to add the (potential) "unused inaccuracy" to the available inaccuracy budget for the remaining iterations, thereby allowing larger subsequent errors. Indeed, knowing $\hat{\omega}_j$, one may first compute $\hat{\phi}_j$ as the root of the (linear) equation $\hat{\omega}_j = \omega_j(\phi_j)$, that is

$$\hat{\phi}_j = \frac{\epsilon_\pi \|b\|_{A^{-1}}}{\hat{\omega}_j \|v_j\|_A \|H_k^{-1}\|_2 \|r_{j-1}\|_2} \quad \text{for FOM, and} \quad \hat{\phi}_j = \frac{(1 - \hat{\omega}_j)\epsilon_\pi \|b\|_{A^{-1}} \|p_j\|_A}{\hat{\omega}_j \|r_j\|_2^2} \quad \text{for CG,} \tag{3.2}$$

and then reset

$$\phi_i = \frac{k_{\max} - j}{1 - \sum_{p=1}^{j} \hat{\phi}_p^{-1}} \quad (i = j+1, \ldots, k_{\max}).$$

In practice, this allows maintaining only single running values for ϕ_{j+1} and $\Phi_j \stackrel{\rm def}{=} 1 - \sum_{p=1}^{j} \hat{\phi}_p^{-1}$ for j ranging from 0 to $k_{\max} - 1$.

3.2 Estimations

We now attempt to estimate unavailable quantities in the theoretical FOM and CG algorithms. We first consider that $\|E\|_{A^{-1},A}$ may not be available from the application context and note that, from (2.7),

$$\|E\|_{A^{-1},A} = \|A^{-1/2} E A^{-1/2}\|_2 \geq \lambda_{\min}(A)^{-1} \|E\|_2, \tag{3.3}$$

so that abound on $\|E\|_2$ can be used provided one knows (an approximation of) the smallest eigenvalue of A. We also have to compute $\|u\|_A$ for a variety of vectors u (the v_j for FOM and the p_j for CG). We can also use the fact that

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

to derive suitable bounds. However, for ill-conditioned problem, these bounds are likely to be often very loose. Another approach is to choose

$$\|u\|_A \approx \frac{1}{n} \mathrm{Tr}(A) \|u\|_2$$

as an approximation (this would be the average value for vectors u with random independent components). Finding an estimation of $\|b\|_{A^{-1}}$ is slighlty more difficult, since this is the size of value of the quadratic at the solution, the quantity we seek to compute. As it turns out[(2)], the best available approximation is the absolute value of current value of the quadratic at the current iterate, and thus we choose

$$\|b\|_{A^{-1}} \approx |q(x_k)| \approx |q_k| = \tfrac{1}{2} |b^T x_k|. \tag{3.4}$$

(At $x = 0$; which occurs at the first iteration of both FOM and CG), we choose the even more approximate value $\|b\|_2 / \sqrt{\|A\|_2}$.). Finally, (2.13) requires an approximation of $\|H_k^{-1}\|_2$. Unfortunately, this value depends on the H_k matrix *at termination* of FOM and its estimation from the current H_i turns out to be quite poor. We have therefore chosen the safe option to use

$$\|H_k^{-1}\|_2 = \frac{1}{\lambda_{\min}(H_k)} \leq \frac{1}{\lambda_{\min}(A)}. \tag{3.5}$$

Combining the above approximations, we suggest to approximate (2.13) by the condition

$$\|E_j\|_2 \leq \min \left[1, \frac{\epsilon_\pi \, n |q_j| \lambda_{\min}(A)}{\phi_j \|r_{j-1}\|_2 \mathrm{Tr}(A) \|v_j\|_2} \right] \quad \text{for all} \quad j \in \{2, \ldots, k\}, \tag{3.6}$$

and (2.16) by

$$\|E_j\|_2 \leq \frac{\epsilon_\pi \, |q_j| \mathrm{Tr}(A) \|p_j\|_2}{n \phi_{j+1} \|r_j\|_2^2 + \epsilon_\pi \, |q_j| \mathrm{Tr}(A) \|p_j\|_2}, \quad \text{for all} \quad j \in \{1, \ldots, k-1\}. \tag{3.7}$$

[(2)] Despite the weak bound of Theorem 2.9.

(The formulae are similar for $j = 1$ in FOM and $j = 0$ in CG, with $|q_j|$ replaced by $\|b\|_2 \lambda_{\max}$).

It also remains to replace (2.15) by a computable estimate. We have chosen to follow ideas in [4] ignoring pathological convergence instances and to estimate

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

where d is a small integer (10, in our case) specifying a "stabilization delay" in the curve of decreasing quadratic values. Using this estimate then suggests replacing (2.15) by the test

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

which, given that $q(x_k)$ is unavailable, is itself approximated by the test

$$q_{k-d} - q_k \leq \tfrac{1}{4}\epsilon |q_k|. \tag{3.8}$$

We conclude this paragraph by stressing the somewhat *ad hoc* nature of the above approximations, which rely in particular on an estimate of the smallest and (less crucially) largest eigenvalues of A. As for the theory presented in Section 2, their justification essentially results from the numerical results presented in Section 4.

3.3 Resulting practical algorithms

We now consolidate our approximations within the body of the FOM and CG algorithms in order to obtain practical versions of these only relying on computable quantities. We note that our definitions of these quantities nevertheless require the user to provide a (potentially very rough, see Section 4) approximate values for the smallest and largest eigenvalues of A. We start by the FOM algorithm on the following page.

Steps 1 to 3 include the determination of a bound on the expected number of iteration and the initialization of ϕ and the inaccuracy budget Φ. The requested accuracy ω_k and the associated inexact product are computed in Steps 5 and 6. The running quadratic q_k value is computed in Step 13 and used for possibly terminating the iteration in Step 14, using the "delay" test (3.8). The formula in Step 13 uses the fact that $q_k = -\frac{1}{2} b^T x_k = -\frac{1}{2} b^T V_k y_k = -\frac{1}{2} z^T y_k$, the j-th component of z being set at iteration j to $v_j^T b$ (in Steps 3 and 19). Steps 15 to 21 introduce the management of the inaccuracy budget described in Section 3.1. Because $k_{\max}$ is based on an estimation of the maximum number of iterations that may be too low, the ϕ are not updated any longer when $k_{\max}$ is reached. This phenomenon occurs for very low condition number in our experiments (only for $\kappa(A) \sim 10^1$). Iterations with fixed accuracy then occur at the end of the minimization[(3)]

Given that the FOM algorithm uses the time and memory for orthonormalization, it seems natural to compare it with an version of the CG algorithm also incorporating reorthonormalization. We therefore include this option in our practical inexact CG method, along with the techniques just described in the FOM case for estimating and managing the inaccuracy budget and selecting the accuracy level for inexact products. This gives the algorithm on page 15.

It is important to note that the cost of applying any of the above algorithms is still dominated by that of the matrix vector products, provided the cost of reorthogonalization does not escalate, that is when the number of iterations remains a moderate fraction of the problem size.

[(3)] An other strategy could be to stop the algorithm after $k_{\max}$ iterations. Preliminary experiments suggest that it reduces the number of iterations when the targeted precision is low without significantly damaging the quality of the solution. For $\kappa(A) \geq 10^2$, the algorithms stop before the accumulated budget vanishes.

A practical inexact FOM algorithm

Given A, b, ϵ, $k_{\max}^{\text{user}}$, $\lambda_{\min}$, $\lambda_{\max}$ and d.

1. Compute $k_{\max}^{\text{spectral}}$ from $\lambda_{\min}$ and $\lambda_{\max}$ using (3.1)
2. $k_{\max} = \min[k_{\max}^{\text{user}}, k_{\max}^{\text{spectral}}]$
3. Set $q_0 = 0$, $\beta = \|b\|_2$, $v_1 = b/\beta$, $z_1 = \beta$, $\phi_1 = k_{\max}$ and $\Phi_1 = 1$
4. For $k = 1, \ldots, k_{\max}^{\text{user}}$, do
5. Determine ω_k from (3.6) with $\epsilon_\pi = \frac{1}{2}\sqrt{\epsilon}$
6. Compute the product $w_k = (A + E_k)v_k$ with $\|E\|_2 \leq \omega_k$, also returning $\hat{\omega}_k$
7. For $i = 1, \ldots, k$ do
8. $h_{i,k} = v_i^T w$
9. $w_k = w_k - h_{i,k} v_i$
10. EndFor
11. $h_{k+1,k} = \|w_k\|_2$
12. $y_k = H_k^{-1}(\beta e_1)$
13. $q_k = -\frac{1}{2} z^T y_k$
14. if $q_{k-d} - q_k \leq \frac{1}{4}\epsilon |q_k|$, then go to 25
15. Compute $\hat{\phi}_k$ from $\hat{\omega}_k$ using the second part of (3.2)
16. $\Phi_{k+1} = \Phi_k - \hat{\phi}_k^{-1}$
17. If $k_{\max} > k$
18. $\phi_{k+1} = (k_{\max} - k)/\Phi_{k+1}$
19. Else
20. $\phi_{k+1} = \phi_k$
21. EndIf
22. $v_{k+1} = w_k / h_{k+1,k}$
23. $z_{k+1} = v_{k+1}^T b$
24. EndFor
25. $x_k = V_k y_k$

4 Numerical experiments

In the numerical experiments we now present, we first to verify and illustrate the theory of Section 2 and then illustrate how performance is affected by the use of the (possibly very) approximate but computable quantities discussed in Section 3. We focus mainly on the reduction in the computational costs associated with the use of inexact matrix-vector products, the original aim of the paper. We use synthetic examples with varying conditionning and examples extracted from the NIST Matrix Market collection(4).

4.1 Continuously varying accuracy

We start with synthetic examples, where, for each example, we have constructed a random symmetric matrix of size $n = 1000$ whose eigenvalues are equidistant in logarithm (base 10) between prescribed $\lambda_{\min}(A)$ and $\lambda_{\max}(A) = 1$, and a random normalized right-hand side b. Note that eigenvalues are not clustered.

We start by considering the case where the accuracy of the products can be specified continuously, assuming that it is possible for the user to obtain the requested accuracy (i.e. $\hat{\omega}_j = \omega_j$ for all j). It can be verified that this implies that $\phi_j = \hat{\phi}_j = k_{\max}$ for all $j =$

(4) Available at https://math.nist.gov/MatrixMarket

A practical inexact CG algorithm

Given A, b, ϵ, $k_{\max}^{\rm user}$, $\lambda_{\min}$, $\lambda_{\max}$, d and reorth.

1. Compute $k_{\max}^{\rm spectral}$ from $\lambda_{\min}$ and $\lambda_{\max}$ using (3.1)
2. $k_{\max} = \min[k_{\max}^{\rm user}, k_{\max}^{\rm spectral}]$
3. Set $x_0 = 0$, $r_0 = -b$, $q_0 = 0$, $\beta_0 = \|b\|_2$, $u_1 = b/\beta_0$, $p_0 = b$, $\phi_0 = k_{\max}$ and $\Phi_0 = 1$
4. For $k = 0, \ldots, k_{\max}^{\rm user}$, do
5. Determine ω_k from (3.7) with $\epsilon_\pi = \frac{1}{2}\sqrt{\epsilon}$
6. Compute the product $c_k = (A + E_k)p_k$ with $\|E\|_2 \leq \omega_k$, also returning $\hat{\omega}_k$
7. $\alpha_k = \beta_k^2 / p_k^T c_k$
8. $x_{k+1} = x_k + \alpha_k p_k$
9. $q_{k+1} = \frac{1}{2} b^T x_{k+1}$
10. If $q_{k+1-d} - q_{k+1} \leq \frac{1}{4}\epsilon |q_{k+1}|$, then stop
11. Compute $\hat{\phi}_k$ from $\hat{\omega}_k$ using the first part of (3.2)
12. $\Phi_{k+1} = \Phi_k - \hat{\phi}_k^{-1}$
13. If $k_{\max} > k$
14. $\phi_{k+1} = (k_{\max} - k)/\Phi_{k+1}$
15. Else
16. $\phi_{k+1} = \phi_k$
17. EndIf
18. $r_{k+1} = r_k + \alpha_k c_k$
19. If (reorth)
20. For $i = 1, \ldots, k$ do
21. $r_{k+1} = r_{k+1} - (u_i^T r_{k+1}) u_i$
22. EndFor
23. $\beta_{k+1} = \|r_{k+1}\|_2$
24. $u_{k+1} = r_{k+1}/\beta_{k+1}$
25. Else
26. $\beta_{k+1} = \|r_{k+1}\|_2$
27. EndIf
28. $p_{k+1} = -r_{k+1} + (\beta_{k+1}/\beta_k)^2 p_k$
29. EndFor

$1, \ldots, k_{\max}$.

In order to illustrate the effective cost of inexact products, we assume that computing Ap is obtained by running a linearly convergent process, whose rate is given by ρ (as defined in (3.1)). This would be the case, for instance, if $A = J^T W^{-1} J$ and only W is known[(5)]. The cost of a full accuracy product is then given by

$$\frac{\log(\epsilon_M)}{\log(\rho)}$$

where ϵ_M is the machine precision, while that of an inexact product with accuracy requirement ω is then assumed to be

$$\frac{\log(\omega)}{\log(\rho)}.$$

These values are then summed in the course of the FOM/CG algorithm to represent an *equivalent number of full accuracy products.*

[(5)] This case occurs in approximately weighted nonlinear least-squares, for instance in data assimilation for weather forecasting.

In order to illustrate the theory of Section 2, we first run versions of FOM and CG where we use the exact tests (2.13) and (2.16) (rather than (3.6)[6] and (3.7)), the true $\|E_j\|_{A^{-1},A}$ and the original termination test (2.15). This is of course impractical but allows measuring the potential for inexactness provided by Theorems 2.4 and 2.7. We consider 6 algorithms:

FOM: the standard FOM with products computed in full machine accuracy,
iFOM: the practical inexact FOM algorithm on page 14.
CG: the standard CG with products computed in full machine accuracy,
CGR: the standard CG with reorthogonalization and products computed in full machine accuracy,
iCG: the practical inexact CG algorithm on the previous page without reorthogonalization,
iCGR: the practical inexact CG algorithm on the preceding page with reorthogonalization,

Tables 4.1-4.3 report the results obtained for three choices of accuracy ($\epsilon = 10^{-3}, 10^{-5}$ and 10^{-7}) and, for each ϵ, choosing the conditionings of A equal to 10^i for $i = 1, \ldots, 8$. 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 $\frac{1}{2}\|r(x) - r\|^2_{A^{-1}}/|q(x)|$,
"r.sol.err" is the relative error on the solution value $(q(x) - q(x_*))/|q(x_*)|$,
"r.val.err" is the relative error on the quadratic value $|q(x) - q|/|q(x_*)|$.

From (2.4) we have that "r.res.gap" + "r.sol.err" $\leq \epsilon$, representing the total error on the true optimal value, while "r.val.err" obeys (2.23).

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.
FOM	10^1	7	7.0e+00	3.5e-30	1.6e-04	1.1e-15	10^2	22	2.2e+01	9.5e-29	2.3e-04	7.4e-15
iFOM		7	2.0e+00	3.7e-30	1.6e-04	7.2e-06		22	9.0e+00	8.8e-29	2.3e-04	7.6e-07
CG		8	8.0e+00	5.0e-30	4.2e-05	2.2e-16		26	2.6e+01	6.2e-29	4.2e-05	1.2e-15
CGR		8	8.0e+00	5.9e-30	4.2e-05	1.1e-15		26	2.6e+01	7.1e-29	4.2e-05	3.5e-15
iCG		8	2.0e+00	1.9e-06	4.4e-05	6.5e-06		26	9.0e+00	2.6e-07	4.2e-05	4.1e-05
iCGR		8	2.0e+00	1.9e-06	4.3e-05	8.3e-05		26	9.0e+00	2.1e-07	4.3e-05	1.6e-05
FOM	10^3	69	6.9e+01	4.3e-27	2.3e-04	4.2e-14	10^4	192	1.9e+02	3.0e-25	2.5e-04	4.6e-13
iFOM		69	3.3e+01	4.7e-27	2.3e-04	2.4e-07		192	1.1e+02	3.0e-25	2.5e-04	1.0e-09
CG		88	8.8e+01	2.0e-27	1.9e-05	3.4e-11		290	2.9e+02	1.3e-25	7.1e-06	1.0e-07
CGR		88	8.8e+01	2.1e-27	1.9e-05	9.6e-15		250	2.5e+02	1.2e-25	6.8e-06	4.4e-15
iCG		89	3.4e+01	2.4e-08	1.8e-05	1.9e-05		301	1.2e+02	2.5e-09	7.1e-06	2.4e-06
iCGR		88	3.3e+01	2.0e-08	1.9e-05	1.1e-05		250	1.0e+02	3.3e-09	6.8e-06	2.4e-06
FOM	10^5	363	3.6e+02	1.9e-23	2.5e-04	6.7e-13	10^6	494	4.9e+02	1.7e-21	2.4e-04	7.7e-12
iFOM		363	2.5e+02	1.9e-23	2.5e-04	2.0e-10		494	3.9e+02	1.3e-21	2.4e-04	1.5e-10
CG		928	9.3e+02	1.0e-23	2.5e-06	1.1e-07		2983	3.0e+03	6.3e-22	8.5e-07	1.3e-08
CGR		433	4.3e+02	9.8e-24	2.4e-06	2.5e-13		565	5.6e+02	6.9e-22	8.2e-07	1.2e-12
iCG		993	4.5e+02	2.8e-10	2.5e-06	1.4e-06		3000	1.5e+03	3.7e-11	3.1e-06	3.3e-07
iCGR		433	2.1e+02	4.9e-10	2.4e-06	2.3e-07		565	3.0e+02	4.5e-10	8.2e-07	7.4e-07
FOM	10^7	583	5.8e+02	1.1e-19	2.5e-04	9.2e-12	10^8	650	6.5e+02	7.0e-18	2.4e-04	2.6e-10
iFOM		583	5.2e+02	9.4e-20	2.5e-04	2.2e-09		650	6.5e+02	6.7e-18	2.4e-04	1.2e-09
CG		3000	3.0e+03	5.4e-20	1.3e-02	6.0e-07		3000	3.0e+03	1.3e-18	3.0e-01	3.6e-06
CGR		656	6.6e+02	5.5e-20	2.7e-07	1.9e-11		721	7.2e+02	5.1e-18	8.6e-08	4.2e-10
iCG		3000	1.7e+03	1.6e-11	2.0e-02	1.3e-06		3000	1.9e+03	3.6e-12	3.5e-01	2.4e-06
iCGR		656	3.7e+02	6.6e-10	2.7e-07	2.1e-07		721	4.4e+02	7.0e-10	8.7e-08	1.0e-06

Table 4.1: Using exact bounds for $\epsilon = 10^{-3}$

These tables suggest the following conclusions.

1. Compared to their full-accuracy versions FOM, CG and CGR, *the inexact variants iFOM, iCG and iCGR exhibit very significant potential savings in the costs of the prod-*

[6] We still use (3.5) to approximate $\|H_k^{-1}\|_2$.

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.
FOM	10^1	11	1.1e+01	4.1e-30	8.3e-07	6.6e-16	10^2	34	3.4e+01	9.0e-29	2.0e-06	7.1e-15
iFOM		11	4.0e+00	3.3e-30	8.3e-07	2.0e-07		34	1.4e+01	1.0e-28	2.0e-06	1.1e-08
CG		11	1.1e+01	4.6e-30	8.3e-07	8.8e-16		38	3.8e+01	5.4e-29	3.7e-07	1.1e-15
CGR		11	1.1e+01	6.4e-30	8.3e-07	4.4e-16		38	3.8e+01	7.1e-29	3.7e-07	3.4e-15
iCG		11	4.0e+00	1.1e-08	8.5e-07	2.9e-07		38	1.4e+01	1.2e-09	3.7e-07	6.6e-07
iCGR		11	4.0e+00	1.1e-08	8.4e-07	4.8e-06		38	1.4e+01	1.1e-09	3.7e-07	7.9e-07
FOM	10^3	104	1.0e+02	4.6e-27	2.3e-06	4.4e-14	10^4	263	2.6e+02	2.6e-25	2.5e-06	4.5e-13
iFOM		104	5.3e+01	4.7e-27	2.3e-06	1.6e-08		263	1.6e+02	2.9e-25	2.5e-06	1.1e-10
CG		123	1.2e+02	2.2e-27	1.8e-07	1.5e-08		393	3.9e+02	1.5e-25	6.9e-08	8.9e-09
CGR		122	1.2e+02	2.3e-27	1.9e-07	5.8e-15		307	3.1e+02	1.6e-25	7.1e-08	6.1e-14
iCG		124	5.0e+01	1.3e-10	1.9e-07	6.2e-09		408	1.8e+02	1.5e-11	7.0e-08	2.0e-07
iCGR		122	4.9e+01	1.2e-10	1.9e-07	6.6e-07		307	1.4e+02	1.6e-11	7.1e-08	6.3e-08
FOM	10^5	433	4.3e+02	2.1e-23	2.4e-06	5.0e-13	10^6	554	5.5e+02	1.4e-21	2.3e-06	7.3e-12
iFOM		433	3.2e+02	2.3e-23	2.4e-06	1.1e-10		554	4.6e+02	1.2e-21	2.3e-06	1.7e-11
CG		1252	1.3e+03	1.0e-23	2.5e-08	1.6e-08		3000	3.0e+03	7.4e-22	8.0e-07	6.9e-08
CGR		486	4.9e+02	1.0e-23	2.5e-08	4.5e-13		606	6.1e+02	6.6e-22	8.1e-09	5.0e-12
iCG		1339	6.5e+02	1.6e-12	2.5e-08	1.1e-07		3000	1.6e+03	4.6e-13	2.5e-06	9.3e-08
iCGR		486	2.5e+02	5.2e-12	2.5e-08	4.5e-08		606	3.4e+02	6.9e-12	8.1e-09	1.1e-07
FOM	10^7	636	6.4e+02	1.1e-19	2.5e-06	6.4e-12	10^8	697	7.0e+02	6.8e-18	2.3e-06	4.4e-10
iFOM		636	6.1e+02	8.4e-20	2.5e-06	2.9e-10		697	7.5e+02	5.2e-18	2.3e-06	8.9e-11
CG		3000	3.0e+03	5.4e-20	1.3e-02	6.0e-07		3000	3.0e+03	1.3e-18	3.0e-01	3.6e-06
CGR		687	6.9e+02	4.9e-20	2.9e-09	2.2e-11		746	7.5e+02	4.1e-18	8.8e-10	6.7e-11
iCG		3000	1.9e+03	1.5e-13	1.9e-02	4.1e-06		3000	2.1e+03	3.5e-14	3.4e-01	4.8e-06
iCGR		687	4.2e+02	6.9e-12	2.8e-09	2.9e-08		747	5.0e+02	7.6e-12	5.9e-10	1.8e-07

Table 4.2: Using exact bounds for $\epsilon = 10^{-5}$

ucts Ap. This is especially the case when the conditioning of the problem is moderate (at most 10^4). As it could be argued that the methods discussed here should be applied on preconditioned systems, this restriction only moderately affect the practical applicability of the technique.

2. Among the inexact variants, *iCGR seems to promise the largest gains* in terms of product costs. The iCG variant, which may be preferable *a priori* since it does not involve the cost and memory requirements for reorthogonalization, suffers from this very feature for more ill-conditioned problems: it is adversely affected by rounding errors and may terminate prematurely when reaching its maximum number of iterations ($3n$ here).

3. The iFOM variant is still efficient and typically produces a more accurate residual gap that the two other inexact variants. Note that, although CGR and FOM are equivalent in exact arithmetic, this is no longer the case for iCGR and iFOM, because the error is made in the product with different vectors (p_j for the former and v_j for the latter).

4. The theoretical bounds (2.13) and (2.16) appear to be often much too restrictive. This is most likely due to the fact that they are based on norms, which take the worst case into account irrespective of the particular directions encountered in the course of the computation. The approximation (3.5), in particular, is often the most restrictive, which might explain the overly accurate residual gaps obtained with the iFOM algorithm.

5. We see that the accuracy obtained on the quadratic value ("r.val.err") is often comparable to the bound on optimality ("r.sol.err"), thereby confirming our comment after Theorem 2.9.

6. The validity of the theoretical bound (obtained in exact arithmetic) does not seem to suffer too much from the effect of rounding errors, even for large condition numbers and high accuracy.

We now show the effect of using the practical algorithms on page 14 and on page 15. In addition to using the approximate constants and tests described in Section 3, we also

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.
FOM	10^1	14	1.4e+01	4.0e-30	1.7e-08	6.6e-16	10^2	45	4.5e+01	1.0e-28	2.2e-08	6.5e-15
iFOM		14	5.0e+00	3.3e-30	1.7e-08	1.2e-07		45	2.0e+01	8.8e-29	2.2e-08	2.1e-08
CG		15	1.5e+01	4.8e-30	4.6e-09	0.0e+00		49	4.9e+01	5.7e-29	4.6e-09	3.1e-16
CGR		15	1.5e+01	5.6e-30	4.6e-09	1.1e-15		49	4.9e+01	7.0e-29	4.6e-09	4.3e-15
iCG		15	5.0e+00	7.3e-11	4.6e-09	1.4e-07		49	1.9e+01	7.1e-12	4.6e-09	2.7e-07
iCGR		15	5.0e+00	7.0e-11	4.7e-09	3.3e-07		49	1.9e+01	9.1e-12	4.6e-09	1.2e-07
FOM	10^3	136	1.4e+02	4.3e-27	2.5e-08	3.7e-14	10^4	320	3.2e+02	2.5e-25	2.4e-08	3.9e-13
iFOM		136	7.4e+01	5.7e-27	2.5e-08	2.3e-10		320	2.1e+02	2.5e-25	2.4e-08	1.3e-10
CG		157	1.6e+02	2.0e-27	1.9e-09	6.7e-09		499	5.0e+02	1.3e-25	7.0e-10	5.2e-10
CGR		155	1.6e+02	2.4e-27	1.7e-09	7.1e-15		356	3.6e+02	1.3e-25	6.9e-10	2.9e-14
iCG		160	6.8e+01	9.0e-13	1.8e-09	9.9e-08		520	2.4e+02	8.7e-14	6.8e-10	2.1e-08
iCGR		155	6.6e+01	7.7e-13	1.7e-09	1.5e-08		356	1.7e+02	8.4e-14	6.9e-10	8.4e-09
FOM	10^5	486	4.9e+02	2.1e-23	2.5e-08	8.3e-13	10^6	598	6.0e+02	1.1e-21	2.2e-08	3.5e-12
iFOM		486	3.8e+02	1.8e-23	2.5e-08	2.4e-11		598	5.4e+02	1.2e-21	2.2e-08	6.7e-11
CG		1559	1.6e+03	8.8e-24	2.5e-10	7.2e-10		3000	3.0e+03	7.4e-22	8.0e-07	6.9e-08
CGR		525	5.2e+02	8.6e-24	2.4e-10	3.0e-13		637	6.4e+02	6.4e-22	7.3e-11	1.6e-12
iCG		1662	8.6e+02	8.5e-15	2.5e-10	1.6e-09		3000	1.8e+03	3.3e-15	2.1e-06	8.5e-08
iCGR		525	2.9e+02	6.3e-14	2.4e-10	5.5e-09		637	3.9e+02	8.0e-14	7.4e-11	1.1e-08
FOM	10^7	674	6.7e+02	1.2e-19	2.3e-08	8.8e-12	10^8	729	7.3e+02	6.3e-18	2.3e-08	4.2e-10
iFOM		674	6.9e+02	8.8e-20	2.3e-08	2.4e-11		729	8.4e+02	7.1e-18	2.3e-08	2.9e-10
CG		3000	3.0e+03	5.4e-20	1.3e-02	6.0e-07		3000	3.0e+03	1.3e-18	3.0e-01	3.6e-06
CGR		713	7.1e+02	5.8e-20	4.4e-11	4.8e-12		769	7.7e+02	4.1e-18	1.0e-10	2.4e-10
iCG		3000	2.1e+03	1.6e-15	1.8e-02	1.8e-06		3000	2.2e+03	3.6e-16	3.3e-01	6.0e-07
iCGR		713	4.7e+02	8.3e-14	1.8e-11	1.5e-08		769	5.5e+02	8.6e-14	2.4e-10	1.9e-08

Table 4.3: Using exact bounds for $\epsilon = 10^{-7}$

perturbed the smallest and largest eigenvalues estimates 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 4.4 to 4.6, using the same conventions as for Tables 4.1 to 4.3,

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.
iFOM	10^1	17	5.0e+00	3.9e-30	6.9e-07	3.3e-05	10^2	32	1.2e+01	1.0e-28	4.3e-06	7.9e-07
iCG		17	5.0e+00	2.1e-06	2.1e-06	1.1e-05		32	1.1e+01	7.9e-07	5.2e-06	5.2e-05
iCGR		17	5.0e+00	2.2e-06	2.2e-06	4.4e-05		32	1.1e+01	9.0e-07	5.0e-06	4.0e-05
iFOM	10^3	75	3.3e+01	4.2e-27	1.1e-04	4.6e-07	10^4	187	9.7e+01	3.1e-25	3.3e-04	7.5e-08
iCG		75	2.7e+01	8.1e-07	1.2e-04	5.2e-05		199	7.7e+01	7.1e-07	4.8e-04	3.8e-05
iCGR		76	2.7e+01	7.4e-07	9.5e-05	1.6e-05		187	7.2e+01	9.0e-07	3.3e-04	1.8e-05
iFOM	10^5	351	2.1e+02	1.7e-23	4.4e-04	2.5e-08	10^6	490	3.4e+02	1.5e-21	3.0e-04	4.8e-08
iCG		523	2.2e+02	1.0e-06	1.6e-03	1.8e-05		1305	5.7e+02	1.5e-06	5.9e-03	7.5e-05
iCGR		351	1.5e+02	3.0e-06	4.4e-04	4.6e-05		490	2.2e+02	4.1e-05	3.4e-04	6.8e-05
iFOM	10^7	586	4.6e+02	8.7e-20	1.9e-04	2.3e-08	10^8	651	5.8e+02	7.2e-18	2.2e-04	7.4e-09
iCG		3000	1.4e+03	4.3e-06	2.4e-02	7.0e-05		3000	1.5e+03	1.6e-06	3.7e-01	9.3e-05
iCGR		587	2.8e+02	1.9e-04	3.6e-04	7.9e-04		651	3.3e+02	3.1e-04	5.2e-04	3.3e-03

Table 4.4: Using practical algorithms with $\epsilon = 10^{-3}$

This leads to the following comments.

1. The practical methods effectively provide significant gains in the cost of performing the matrix-vector products in FOM or CG.

2. The practical variants typically take slightly more iterations than their "impractical" versions discussed above, which is partly explained by the fact that the termination criterion is based on the delay d (10 in our case) to assess termination. This is obvious for the case $\kappa(A) = 10^1$ for which the algorithms implemented in full machine precision and the "impractical" algorithms converge in 7 to 15 iterations (depending on the targeted accuracy), which is the order of the delay. Experiments with a delay d equal to 5 led to a decrease in the iteration number, especially for matrices with low condition numbers (not shown). Thus, the tuning of the parameter d is problem dependent and should be adapted to the condition number.

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.
iFOM	10^1	21	7.0e+00	4.1e-30	3.3e-09	1.3e-06	10^2	44	1.7e+01	1.1e-28	3.3e-08	1.4e-08
iCG		21	6.0e+00	1.2e-08	1.2e-08	5.0e-06		49	1.7e+01	5.2e-09	9.5e-09	4.9e-06
iCGR		21	6.0e+00	1.1e-08	1.1e-08	3.8e-07		44	1.5e+01	5.7e-09	3.9e-08	2.1e-06
iFOM	10^3	111	5.2e+01	4.4e-27	9.1e-07	5.3e-08	10^4	266	1.5e+02	3.4e-25	2.0e-06	6.2e-09
iCG		123	4.6e+01	5.1e-09	2.3e-07	2.9e-06		307	1.3e+02	5.6e-09	5.4e-06	1.3e-06
iCGR		112	4.3e+01	5.3e-09	7.9e-07	1.7e-06		266	1.1e+02	9.0e-09	2.0e-06	2.3e-06
iFOM	10^5	436	2.8e+02	1.9e-23	1.9e-06	7.9e-09	10^6	558	4.1e+02	1.3e-21	1.6e-06	1.0e-09
iCG		867	3.8e+02	8.6e-09	1.6e-05	1.0e-06		2345	1.1e+03	4.0e-08	7.0e-05	1.3e-05
iCGR		436	1.9e+02	9.9e-08	2.1e-06	2.4e-06		558	2.6e+02	2.8e-06	4.6e-06	9.7e-05
iFOM	10^7	642	5.4e+02	8.0e-20	1.4e-06	1.2e-09	10^8	704	6.7e+02	7.3e-18	9.9e-07	9.6e-10
iCG		3000	1.5e+03	3.3e-07	2.3e-02	3.0e-05		3000	1.6e+03	3.1e-07	3.6e-01	4.7e-05
iCGR		643	3.2e+02	4.7e-05	4.7e-05	6.5e-04		704	3.7e+02	1.7e-04	1.7e-04	1.0e-03

Table 4.5: Using practical algorithms with $\epsilon = 10^{-5}$

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.
iFOM	10^1	32	1.0e+01	3.9e-30	2.3e-11	2.7e-08	10^2	56	2.3e+01	9.5e-29	2.3e-10	7.3e-09
iCG		23	8.0e+00	8.8e-11	8.8e-11	6.4e-07		51	1.9e+01	3.8e-11	2.0e-09	2.9e-07
iCGR		24	8.0e+00	8.2e-11	8.2e-11	7.5e-09		55	2.0e+01	3.9e-11	3.8e-10	2.3e-07
iFOM	10^3	145	7.2e+01	5.3e-27	6.9e-09	4.8e-09	10^4	324	1.9e+02	2.7e-25	1.7e-08	2.1e-10
iCG		145	5.9e+01	3.6e-11	1.2e-08	2.5e-07		387	1.7e+02	4.1e-11	1.7e-07	6.5e-07
iCGR		145	5.9e+01	4.0e-11	6.9e-09	8.1e-08		324	1.4e+02	7.4e-11	1.7e-08	6.7e-08
iFOM	10^5	491	3.4e+02	2.4e-23	1.4e-08	4.7e-10	10^6	605	4.8e+02	1.3e-21	9.1e-09	2.3e-10
iCG		1017	4.8e+02	6.1e-11	1.8e-06	4.3e-07		2672	1.3e+03	7.5e-10	1.3e-05	2.0e-06
iCGR		492	2.3e+02	1.8e-09	1.4e-08	4.3e-08		605	3.0e+02	7.7e-08	8.3e-08	2.4e-06
iFOM	10^7	683	6.1e+02	9.6e-20	5.4e-09	3.0e-10	10^8	738	7.6e+02	7.2e-18	4.5e-09	9.4e-10
iCG		3000	1.6e+03	5.0e-09	2.2e-02	4.1e-06		3000	1.7e+03	6.7e-09	3.5e-01	4.3e-06
iCGR		683	3.6e+02	2.2e-06	2.2e-06	7.1e-05		738	4.1e+02	4.2e-05	4.2e-05	3.8e-05

Table 4.6: Using practical algorithms with $\epsilon = 10^{-7}$

3. The relative ranking of the various algorithms is very consistent with that observed above for the "impractical" variants. The only qualitative difference observed is, not surprisingly[(7)], that the practical algorithms seem to suffer more from the rounding errors, in that the desired accuracy cannot be reached when the ratio of accuracy on conditioning ($\epsilon/\kappa(A)$) approaches machine precision.

We conclude this analysis of the case where the accuracy of the product Ap can be specified continuously by presenting some results of running the practical inexact algorithms on some examples from the NIST Matrix Market. Dimension, condition number and 2-norm of the matrices are given in Table (4.7). All of them are symmetric positive definite and are associated with PDE problems.

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 4.7: Properties of the Matrix Market matrices (sorted by increasing condition number)

[(7)] Because of the low accuracy of the estimates $\lambda_{\max}$ and $\lambda_{\min}$.

Again, we perturbed the smallest and largest eigenvalues estimates by a random relative perturbation of magnitude between 0 and 100%. We report the results of the corresponding runs in Tables 4.8 to 4.10, using the same conventions as for Tables 4.1 to 4.3.

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.
FOM	bcsstm02	7	7.0e+00	1.2e-31	3.5e-05	2.1e-16	nos4	45	4.5e+01	1.5e-28	2.4e-04	5.4e-15
iFOM		16	5.0e+00	1.3e-31	1.6e-06	2.2e-04		55	2.9e+01	1.0e-28	7.5e-07	6.9e-08
CG		7	7.0e+00	2.6e-32	3.5e-05	0.0e+00		50	5.0e+01	2.9e-29	8.4e-06	5.2e-12
CGR		7	7.0e+00	8.8e-32	3.5e-05	2.1e-16		50	5.0e+01	1.3e-28	8.4e-06	5.6e-15
iCG		16	5.0e+00	1.3e-05	1.3e-05	1.7e-04		56	2.2e+01	8.3e-09	8.4e-07	5.4e-05
iCGR		17	5.0e+00	1.3e-05	1.3e-05	5.1e-04		55	2.1e+01	1.4e-07	8.5e-07	4.2e-06
FOM	bcsstk09	89	8.9e+01	7.6e-27	2.5e-04	1.6e-13	bcsstk05	84	8.4e+01	1.1e-27	2.0e-04	4.6e-13
iFOM		92	7.3e+01	2.3e-26	2.0e-04	5.8e-12		93	6.4e+01	3.8e-27	4.6e-05	4.8e-10
CG		52	5.2e+01	1.8e-27	5.5e-02	2.6e-14		73	7.3e+01	2.4e-28	6.8e-03	2.0e-06
CGR		52	5.2e+01	2.4e-27	5.5e-02	5.4e-14		65	6.5e+01	8.7e-28	7.6e-03	2.1e-14
iCG		92	5.2e+01	3.1e-16	2.0e-04	2.6e-08		121	6.2e+01	1.2e-12	1.1e-04	4.3e-07
iCGR		92	5.2e+01	1.2e-15	2.0e-04	8.8e-09		93	4.7e+01	5.0e-12	4.6e-05	1.5e-07
FOM	bcsstk27	219	2.2e+02	4.7e-27	2.4e-04	4.9e-13	685_bus	130	1.3e+02	1.8e-26	2.5e-04	3.4e-12
iFOM		214	1.7e+02	3.9e-27	3.1e-04	1.8e-11		139	9.9e+01	3.2e-26	8.2e-05	1.8e-10
CG		154	1.5e+02	9.4e-30	1.0e-02	1.2e-06		225	2.2e+02	2.4e-27	7.2e-05	8.5e-09
CGR		137	1.4e+02	1.8e-27	1.1e-02	2.7e-15		141	1.4e+02	9.6e-27	7.0e-05	1.1e-13
iCG		238	1.3e+02	4.3e-14	1.0e-03	6.5e-07		267	1.4e+02	5.4e-11	4.0e-04	3.3e-06
iCGR		214	1.2e+02	1.0e-13	3.1e-04	4.0e-09		139	6.9e+01	9.2e-10	8.2e-05	1.1e-06
FOM	nos1	191	1.9e+02	2.2e-22	2.0e-04	1.4e-10	nos7	207	2.1e+02	6.1e-20	2.4e-04	2.1e-08
iFOM		200	2.6e+02	4.9e-22	4.8e-05	1.4e-10		181	2.5e+02	4.4e-18	3.0e-03	2.0e-08
CG		711	7.1e+02	3.6e-23	3.1e-01	6.8e-07		1497	1.5e+03	3.0e-20	1.8e-05	1.9e-07
CGR		182	1.8e+02	7.3e-24	4.7e-03	4.4e-12		240	2.4e+02	1.8e-19	1.9e-05	5.4e-10
iCG		711	4.8e+02	1.3e-13	3.9e-01	3.8e-06		529	3.3e+02	1.8e-06	1.5e-01	1.6e-04
iCGR		200	1.4e+02	7.2e-10	4.8e-05	2.6e-06		182	1.2e+02	4.9e-05	3.1e-03	1.9e-03

Table 4.8: Matrix Market: using practical algorithms with $\epsilon = 10^{-3}$

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.
FOM	bcsstm02	9	9.0e+00	4.9e-32	2.2e-06	4.2e-16	nos4	53	5.3e+01	2.6e-28	2.3e-06	1.9e-15
iFOM		18	6.0e+00	9.6e-32	5.9e-09	1.8e-05		63	3.5e+01	1.3e-28	2.7e-09	6.7e-09
CG		10	1.0e+01	2.4e-32	4.9e-07	0.0e+00		59	5.9e+01	3.2e-29	5.4e-08	1.9e-08
CGR		10	1.0e+01	1.1e-31	4.9e-07	0.0e+00		59	5.9e+01	1.4e-28	5.4e-08	3.0e-15
iCG		18	5.0e+00	7.2e-08	7.2e-08	5.8e-05		64	2.7e+01	1.3e-10	4.2e-09	8.2e-06
iCGR		19	6.0e+00	1.0e-07	1.0e-07	4.8e-07		63	2.5e+01	1.3e-09	3.9e-09	1.2e-05
FOM	bcsstk09	153	1.5e+02	8.6e-27	2.5e-06	1.4e-13	bcsstk05	119	1.2e+02	2.5e-27	2.2e-06	5.1e-13
iFOM		152	1.3e+02	7.1e-27	2.6e-06	7.2e-13		129	9.2e+01	1.8e-27	2.7e-09	9.2e-11
CG		80	8.0e+01	4.3e-27	6.1e-04	9.7e-14		113	1.1e+02	3.8e-28	7.8e-05	4.0e-07
CGR		80	8.0e+01	3.5e-27	6.1e-04	1.0e-14		89	8.9e+01	8.5e-28	6.9e-05	4.4e-15
iCG		152	8.9e+01	1.2e-18	2.7e-06	4.7e-09		179	9.6e+01	3.8e-15	1.1e-05	9.3e-08
iCGR		152	8.9e+01	6.1e-18	2.7e-06	3.6e-10		129	6.7e+01	3.6e-13	2.7e-09	6.7e-08
FOM	bcsstk27	302	3.0e+02	5.6e-27	2.4e-06	4.5e-13	685_bus	182	1.8e+02	7.4e-26	2.3e-06	3.6e-12
iFOM		293	2.4e+02	3.6e-27	4.1e-06	1.3e-12		188	1.4e+02	7.1e-26	1.0e-06	2.7e-11
CG		305	3.0e+02	1.8e-29	1.0e-04	2.5e-07		322	3.2e+02	4.8e-27	7.0e-07	4.4e-09
CGR		235	2.4e+02	1.9e-27	1.0e-04	2.3e-15		191	1.9e+02	1.7e-26	6.5e-07	3.2e-14
iCG		397	2.3e+02	3.5e-16	7.8e-06	7.0e-08		370	2.0e+02	2.7e-13	4.9e-06	2.8e-07
iCGR		293	1.7e+02	6.2e-16	4.1e-06	9.7e-10		188	9.8e+01	6.6e-12	1.0e-06	8.4e-07
FOM	nos1	220	2.2e+02	4.7e-22	2.1e-06	1.4e-10	nos7	270	2.7e+02	1.0e-18	1.8e-06	2.2e-08
iFOM		230	3.1e+02	1.7e-21	9.0e-09	1.4e-10		251	3.5e+02	1.0e-17	1.4e-05	2.1e-08
CG		711	7.1e+02	3.6e-23	3.1e-01	6.8e-07		2102	2.1e+03	1.5e-22	1.7e-07	2.7e-08
CGR		199	2.0e+02	1.1e-23	5.6e-05	1.7e-12		300	3.0e+02	1.2e-18	9.6e-08	1.0e-09
iCG		711	5.2e+02	9.1e-16	3.8e-01	1.4e-06		1361	8.8e+02	6.5e-08	3.0e-03	6.9e-05
iCGR		230	1.6e+02	5.9e-12	8.9e-09	2.1e-07		261	1.7e+02	1.4e-06	1.4e-05	2.4e-05

Table 4.9: Matrix Market: using practical algorithms with $\epsilon = 10^{-5}$

Similar comments to the synthetic cases can be made:

1. The practical variants tend to result in an increase of the iteration numbers compared to both the exact (FOM, CG and CG with reorthogonalization) and the "impractical" methods with inaccurate matrix-vector products (not shown for the latter ones). For instance, the practical CG with reorthogonalization requires almost twice the number of iterations of the method with exact matrix-vector products.

2. Despite these increases in the iteration number, the costs of the practical methods remain lower than those of the exact methods. As for the CG with reorthogonalization,

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.
FOM	bcsstm02	11	1.1e+01	7.1e-32	2.5e-09	2.1e-16	nos4	60	6.0e+01	4.0e-28	2.4e-08	1.2e-15
iFOM		24	8.0e+00	1.4e-31	3.9e-11	8.9e-07		70	4.1e+01	9.9e-29	3.5e-13	6.6e-10
CG		11	1.1e+01	3.0e-32	2.5e-09	0.0e+00		65	6.5e+01	3.9e-29	4.2e-10	1.0e-09
CGR		11	1.1e+01	1.1e-31	2.5e-09	2.1e-16		64	6.4e+01	1.7e-28	6.7e-10	4.7e-15
iCG		19	6.0e+00	7.5e-10	7.5e-10	2.6e-07		72	3.2e+01	5.6e-13	1.9e-12	6.8e-07
iCGR		21	7.0e+00	6.7e-10	6.7e-10	4.9e-07		70	3.0e+01	1.2e-11	1.2e-11	1.1e-06
FOM	bcsstk09	185	1.8e+02	7.3e-27	2.5e-08	1.6e-13	bcsstk05	127	1.3e+02	2.7e-27	2.3e-08	5.0e-13
iFOM		195	1.7e+02	1.0e-26	3.6e-09	9.0e-14		137	1.0e+02	3.3e-27	9.5e-12	9.0e-12
CG		140	1.4e+02	3.2e-27	5.9e-06	4.8e-08		193	1.9e+02	6.6e-28	7.6e-07	7.1e-08
CGR		140	1.4e+02	3.9e-27	5.7e-06	1.7e-14		121	1.2e+02	1.2e-27	7.2e-07	8.8e-15
iCG		195	1.2e+02	7.6e-21	4.9e-09	3.0e-09		232	1.3e+02	1.1e-16	2.1e-08	3.5e-08
iCGR		195	1.2e+02	2.0e-20	3.8e-09	4.0e-11		137	7.6e+01	4.4e-15	9.5e-12	1.7e-08
FOM	bcsstk27	377	3.8e+02	6.8e-27	2.4e-08	5.2e-13	685_bus	213	2.1e+02	3.1e-26	2.1e-08	3.5e-12
iFOM		375	3.3e+02	4.8e-27	2.7e-08	2.9e-13		222	1.8e+02	2.0e-26	3.8e-09	1.3e-11
CG		449	4.5e+02	2.6e-29	1.1e-06	3.7e-09		385	3.8e+02	3.1e-27	7.4e-09	2.1e-10
CGR		317	3.2e+02	1.9e-27	1.0e-06	3.7e-15		219	2.2e+02	8.5e-27	6.4e-09	2.7e-14
iCG		516	3.2e+02	2.9e-18	1.6e-07	5.4e-09		456	2.6e+02	8.1e-15	3.3e-08	7.3e-08
iCGR		375	2.3e+02	5.1e-18	2.7e-08	3.5e-11		222	1.2e+02	9.0e-14	3.8e-09	5.3e-08
FOM	nos1	226	2.3e+02	2.6e-22	1.9e-08	1.4e-10	nos7	315	3.2e+02	8.2e-19	2.0e-08	2.0e-08
iFOM		236	3.5e+02	6.8e-23	8.1e-12	1.4e-10		311	4.5e+02	4.8e-20	6.5e-08	2.3e-08
CG		711	7.1e+02	3.6e-23	3.1e-01	6.8e-07		2187	2.2e+03	6.7e-19	7.3e-08	5.4e-10
CGR		222	2.2e+02	2.8e-23	4.5e-07	6.3e-13		341	3.4e+02	1.6e-18	1.0e-09	1.5e-09
iCG		711	5.7e+02	2.2e-17	3.6e-01	1.0e-06		2002	1.4e+03	8.6e-10	5.3e-05	3.8e-06
iCGR		236	1.8e+02	7.1e-14	8.8e-12	2.9e-08		318	2.1e+02	2.5e-08	9.7e-08	5.0e-05

Table 4.10: Matrix Market: using practical algorithms with $\epsilon = 10^{-7}$

the practical variant results in costs that are 15% to 40% lower than those of the exact method depending on the matrix.

3. Again the practical algorithms seem more sensitive to rounding errors. For instance, the desired accuracy cannot be reached for the 685_bus and nos7 matrices without reorthogonalization. As observed in the synthetic matrices, it results in a decrease in the number of iterations of the practical methods, and so a damage of the quality of the solution, compared to the methods with accurate matrix-vector products.

4. We note significant increases of the iteration numbers for the practical algorithms with the matrix bcsstm02. Its condition number is very low ($\kappa(A) = 8.8$) and results in a fast convergence of the exact algorithms (less than 10 iterations). As for the synthetic matrices, the fact that the delay d (equal to 10) introduced in the stopping criterion is too large partly explains these increases.

Finally, it should be noted that the problems associated with the Matrix Market matrices seem to be "easier" than the synthetic case, in the sense that the CG, FOM and CGR methods are able to provide solutions for which the relative errors on the solution value are lower than the prescribed accuracy, even for ill-conditioned matrices. This is not the case for the synthetic matrices.

4.2 Multi-precision and discontinuous accuracy

While admissible in theory, the case where accuracy of the product can be continuously adapted may be unrealistic. Indeed the computational process by which the products Ap are computed may be constrained to produce results whose accuracy is fixed to one of several predetermined levels. This occurs in particular in the inaccuracy arises from the use of multi-precision arithmetic, or if the inaccuracies results from the use of hierachical models such as nested discretizations. Because of the current high interest in multi-precision computations, we present our results in this context.

Using single of half precision arithmetic is nowadays considered as crucial for allowing fast computations that are energy-wise efficient. The energy use of floating-points units is broadly proportional to the chip area used, which is itself dominated by the square of the length of the mantissa of the involved numbers. Thus half precision arithmetic is much more energy efficient than the corresponding double precision version. As a consequence, the use of specialized multi-precision floating points units is considered as critical in the context of very-high performance computing [14, 12].

In the following set of experiments with the practical FOM and CG algorithms, we assume that the products Ap can be computed in double, single or half precision (with corresponding accuracy level equal to machine precision, half machine precision or quarter machine precision). Thus, when the inexact algorithm specifies an accuracy level ω_j, this may not be attainable as such, but the lower of these three levels of accuracy is then chosen to perform the computation in (possibly moderately) higher accuracy than requested. In this case, we assume that it is possible to return the precision level effectively used to the algorithm by specifying $\hat{\omega}_j \leq \omega_j$.

To measure the gains obtainable in this context, we again need to define an "equivalent" cost of a product Ap in double, single or half precision. According [14, 12] and [7, 15], the gain in efficiency depends on the details of the architecture and is between 5 and 3 for each decrease from double to single precision, or from single to half. In our experiments, we have assigned a unit cost to a matrix product in double precison, a cost $\frac{1}{4}$ for a product in single precision and a cost $\frac{1}{16}$ to a product in half precision.

As in the previous paragraph, Tables 4.11-4.13 (resp. 4.14-4.16) report the results obtained when running the practical FOM and CG algorithms with synthetic matrices (resp. matrices from the NIST Matrix Market) and simulating the accuracy reduction to one of the three chosen precision levels, for three choices of final accuracy ($\epsilon = 10^{-3}, 10^{-5}$ and 10^{-7}). The "cost" column is now computed as just described.

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.
iFOM	10^1	17	1.2e+00	3.8e-30	4.0e-06	5.5e-05	10^2	32	5.2e+00	9.1e-29	4.3e-06	1.8e-06
iCG		17	1.1e+00	6.6e-06	6.7e-06	5.4e-06		32	2.9e+00	2.3e-06	6.6e-06	8.2e-05
iCGR		17	1.1e+00	7.3e-06	7.3e-06	3.8e-05		32	2.9e+00	2.4e-06	6.4e-06	4.5e-05
iFOM	10^3	75	2.1e+01	5.0e-27	1.1e-04	5.0e-07	10^4	187	1.3e+02	3.0e-25	3.3e-04	1.1e-07
iCG		75	1.1e+01	1.7e-06	1.2e-04	7.8e-05		200	4.9e+01	1.3e-06	4.6e-04	3.6e-05
iCGR		76	1.1e+01	1.6e-06	9.6e-05	1.7e-05		187	4.6e+01	1.6e-06	3.3e-04	2.9e-06
iFOM	10^5	351	3.5e+02	1.9e-23	4.4e-04	3.3e-08	10^6	490	4.9e+02	1.3e-21	3.0e-04	8.9e-08
iCG		524	1.4e+02	1.6e-06	1.5e-03	1.8e-05		1307	3.6e+02	2.1e-06	5.9e-03	7.2e-05
iCGR		351	9.2e+01	4.5e-06	4.4e-04	4.3e-05		490	1.4e+02	5.8e-05	3.5e-04	1.3e-06
iFOM	10^7	586	5.9e+02	6.1e-20	1.9e-04	5.1e-08	10^8	651	6.5e+02	5.9e-18	2.2e-04	5.7e-09
iCG		3000	8.4e+02	6.4e-06	2.4e-02	1.1e-04		3000	1.2e+03	2.2e-06	3.7e-01	1.4e-04
iCGR		587	1.9e+02	2.2e-04	3.9e-04	8.9e-04		651	3.1e+02	3.2e-04	5.3e-04	3.4e-03

Table 4.11: Using practical algorithms in multi-precision arithmetic with $\epsilon = 10^{-3}$

The results for the multi-precision case 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 using synthetic matrices, 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 then computed in full double precision. This is especially noticeable for the iFOM algorithm, due to the conservative nature of (2.13) involving $\|H_k\|$. However, we note efficiency gains occuring also for the ill-conditioned cases with the real matrices compared

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.
iFOM	10^1	23	2.2e+00	4.0e-30	1.3e-08	4.1e-07	10^2	44	8.4e+00	9.3e-29	3.3e-08	2.0e-07
iCG		21	1.9e+00	2.9e-08	2.9e-08	4.4e-06		52	6.4e+00	1.3e-08	1.4e-08	9.7e-06
iCGR		21	1.9e+00	3.0e-08	3.0e-08	8.3e-07		44	5.9e+00	1.5e-08	4.7e-08	3.1e-07
iFOM	10^3	111	4.5e+01	5.2e-27	9.1e-07	6.9e-08	10^4	266	2.2e+02	2.9e-25	2.0e-06	8.6e-09
iCG		132	2.3e+01	1.2e-08	8.1e-08	8.0e-07		304	8.0e+01	1.0e-08	6.3e-06	1.0e-06
iCGR		112	2.2e+01	1.1e-08	7.9e-07	8.9e-07		266	7.0e+01	1.8e-08	2.0e-06	1.6e-06
iFOM	10^5	436	4.4e+02	2.0e-23	1.9e-06	1.2e-08	10^6	558	5.6e+02	1.2e-21	1.6e-06	1.0e-09
iCG		870	2.4e+02	1.5e-08	1.5e-05	2.2e-06		2348	7.1e+02	9.1e-08	7.0e-05	2.5e-05
iCGR		436	1.3e+02	1.7e-07	2.1e-06	4.0e-07		558	2.0e+02	4.9e-06	6.5e-06	1.3e-04
iFOM	10^7	642	6.4e+02	9.7e-20	1.4e-06	3.2e-09	10^8	704	7.0e+02	6.4e-18	9.9e-07	3.8e-09
iCG		3000	1.2e+03	1.3e-06	2.3e-02	4.1e-05		3000	2.1e+03	8.9e-07	3.6e-01	5.4e-05
iCGR		643	3.0e+02	7.7e-05	7.7e-05	8.9e-04		704	4.3e+02	2.1e-04	2.1e-04	1.3e-03

Table 4.12: Using practical algorithms in multi-precision arithmetic with $\epsilon = 10^{-5}$

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.
iFOM	10^1	32	3.5e+00	4.0e-30	7.8e-11	1.4e-07	10^2	57	1.4e+01	9.1e-29	1.5e-10	1.1e-08
iCG		22	2.7e+00	2.2e-10	2.1e-10	9.0e-07		52	8.9e+00	8.7e-11	1.4e-09	2.1e-07
iCGR		24	2.8e+00	2.2e-10	2.2e-10	9.1e-08		55	9.1e+00	9.5e-11	4.4e-10	3.6e-07
iFOM	10^3	145	8.1e+01	3.9e-27	6.9e-09	6.7e-09	10^4	324	2.9e+02	2.6e-25	1.7e-08	1.9e-10
iCG		152	3.4e+01	7.7e-11	4.9e-09	3.2e-07		396	1.2e+02	7.6e-11	1.2e-07	8.8e-07
iCGR		145	3.4e+01	8.6e-11	6.9e-09	3.6e-08		324	1.0e+02	1.4e-10	1.7e-08	4.4e-08
iFOM	10^5	491	4.9e+02	1.8e-23	1.4e-08	6.0e-10	10^6	605	6.0e+02	1.2e-21	9.1e-09	6.7e-10
iCG		1030	4.2e+02	1.0e-10	1.5e-06	6.3e-07		2556	1.2e+03	3.1e-09	2.3e-05	2.1e-06
iCGR		492	2.0e+02	3.7e-09	1.6e-08	2.6e-07		605	3.3e+02	2.0e-07	2.1e-07	1.9e-05
iFOM	10^7	683	6.8e+02	9.6e-20	5.4e-09	9.9e-10	10^8	737	7.4e+02	6.4e-18	5.9e-09	3.6e-09
iCG		3000	2.1e+03	1.3e-07	2.2e-02	8.9e-06		3000	2.8e+03	2.1e-07	3.6e-01	1.1e-05
iCGR		683	4.4e+02	5.5e-06	5.5e-06	1.1e-04		738	5.6e+02	7.9e-05	7.9e-05	1.3e-04

Table 4.13: Using practical algorithms in multi-precision arithmetic with $\epsilon = 10^{-7}$

to the continuous varying accuracy case. This can be partly associated with the significant decreases in the number of iterations for the methods, and 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 multi-precision context.

5 Conclusions

We have considered the iterative solution of convex quadratic optimization problems and have proposed an analysis of variants of FOM and CG which allow and control 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 new practical algorithms that use them. We have finally discussed a fairly large set of numerical experiments, suggesting that significant gains in efficiency can be achieved by the use of variable precision 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 the control is limited to predefined levels. We have illustrated the latter in the important context of multi-precision computations.

In view of the promising potential of the new algorithms, it is now interesting to apply them in more general contexts, such as other optimization algorithms for instance involving nonquadratic and possibly nonconvex objective functions. Given the emerging concepts for new high-performance computer architecture and the needs resulting thereof, it is also worthwhile, in our opinion, to pursue experimentation with the new methods in the framework

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.
FOM	bcsstm02	7	7.0e+00	1.2e-31	3.5e-05	2.1e-16	nos4	45	4.5e+01	1.5e-28	2.4e-04	5.4e-15
iFOM		16	1.4e+00	9.0e-32	2.1e-05	6.5e-04		55	3.8e+01	1.2e-28	7.5e-07	9.4e-08
CG		7	7.0e+00	2.6e-32	3.5e-05	0.0e+00		50	5.0e+01	2.9e-29	8.4e-06	5.2e-12
CGR		7	7.0e+00	8.8e-32	3.5e-05	2.1e-16		50	5.0e+01	1.3e-28	8.4e-06	5.6e-15
iCG		16	1.2e+00	8.5e-05	8.5e-05	3.0e-04		56	1.4e+01	9.4e-09	8.4e-07	5.8e-05
iCGR		17	1.2e+00	8.7e-05	8.7e-05	4.0e-04		55	1.2e+01	2.0e-07	8.9e-07	7.0e-07
FOM	bcsstk09	89	8.9e+01	7.6e-27	2.5e-04	1.6e-13	bcsstk05	84	8.4e+01	1.1e-27	2.0e-04	4.6e-13
iFOM		92	2.2e+01	2.1e-26	2.0e-04	5.9e-12		93	1.9e+01	1.5e-27	4.6e-05	4.8e-10
CG		52	5.2e+01	1.8e-27	5.5e-02	2.6e-14		73	7.3e+01	2.4e-28	6.8e-03	2.0e-06
CGR		52	5.2e+01	2.4e-27	5.5e-02	5.4e-14		65	6.5e+01	8.7e-28	7.6e-03	2.1e-14
iCG		92	5.8e+00	3.0e-16	2.0e-04	2.6e-08		119	8.4e+00	2.4e-15	1.0e-04	4.4e-07
iCGR		92	5.8e+00	1.2e-15	2.0e-04	8.7e-09		93	5.8e+00	4.3e-12	4.6e-05	3.6e-07
FOM	bcsstk27	219	2.2e+02	4.7e-27	2.4e-04	4.9e-13	685_bus	130	1.3e+02	1.8e-26	2.5e-04	3.4e-12
iFOM		214	2.0e+02	4.0e-27	3.1e-04	1.3e-11		139	6.1e+01	2.1e-26	8.2e-05	3.0e-10
CG		154	1.5e+02	9.4e-30	1.0e-02	1.2e-06		225	2.2e+02	2.4e-27	7.2e-05	8.5e-09
CGR		137	1.4e+02	1.8e-27	1.1e-02	2.7e-15		141	1.4e+02	9.6e-27	7.0e-05	1.1e-13
iCG		236	5.5e+01	1.4e-16	1.0e-03	4.9e-08		266	3.0e+01	4.2e-11	3.5e-04	3.5e-06
iCGR		214	5.0e+01	1.9e-16	3.1e-04	1.8e-10		139	1.3e+01	1.0e-09	8.2e-05	7.0e-07
FOM	nos1	191	1.9e+02	2.2e-22	2.0e-04	1.4e-10	nos7	207	2.1e+02	6.1e-20	2.4e-04	2.1e-08
iFOM		200	1.7e+02	1.1e-21	4.8e-05	1.4e-10		181	1.8e+02	6.7e-19	3.0e-03	2.1e-08
CG		711	7.1e+02	3.6e-23	3.1e-01	6.8e-07		1497	1.5e+03	3.0e-20	1.8e-05	1.9e-07
CGR		182	1.8e+02	7.3e-24	4.7e-03	4.4e-12		240	2.4e+02	1.8e-19	1.9e-05	5.4e-10
iCG		711	1.6e+02	6.6e-18	3.6e-01	2.7e-07		258	3.3e+01	1.8e-07	5.4e-01	6.2e-05
iCGR		200	4.0e+01	3.0e-16	4.8e-05	6.4e-10		186	2.3e+01	6.1e-05	3.1e-03	1.7e-03

Table 4.14: Matrix Market: using practical algorithms in multi-precision with $\epsilon = 10^{-3}$

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.
FOM	bcsstm02	9	9.0e+00	4.9e-32	2.2e-06	4.2e-16	nos4	53	5.3e+01	2.6e-28	2.3e-06	1.9e-15
iFOM		17	2.0e+00	8.7e-32	1.5e-07	1.2e-04		63	5.2e+01	9.0e-29	2.7e-09	8.9e-09
CG		10	1.0e+01	2.4e-32	4.9e-07	0.0e+00		59	5.9e+01	3.2e-29	5.4e-08	1.9e-08
CGR		10	1.0e+01	1.1e-31	4.9e-07	0.0e+00		59	5.9e+01	1.4e-28	5.4e-08	3.0e-15
iCG		21	1.9e+00	6.7e-07	6.7e-07	2.2e-04		64	1.7e+01	1.5e-10	4.2e-09	8.6e-06
iCGR		19	1.8e+00	5.3e-07	5.3e-07	3.1e-05		63	1.6e+01	2.0e-09	4.6e-09	1.5e-05
FOM	bcsstk09	153	1.5e+02	8.6e-27	2.5e-06	1.4e-13	bcsstk05	119	1.2e+02	2.5e-27	2.2e-06	5.1e-13
iFOM		152	4.3e+01	4.4e-27	2.6e-06	8.7e-13		129	2.6e+01	4.0e-27	2.7e-09	9.1e-11
CG		80	8.0e+01	4.3e-27	6.1e-04	9.7e-14		113	1.1e+02	3.8e-28	7.8e-05	4.0e-07
CGR		80	8.0e+01	3.5e-27	6.1e-04	1.0e-14		89	8.9e+01	8.5e-28	6.9e-05	4.4e-15
iCG		152	1.1e+01	8.1e-19	2.7e-06	4.2e-09		179	1.2e+01	2.5e-15	1.0e-05	3.2e-10
iCGR		152	1.1e+01	4.5e-18	2.7e-06	3.2e-10		129	8.6e+00	2.7e-13	2.7e-09	5.8e-08
FOM	bcsstk27	302	3.0e+02	5.6e-27	2.4e-06	4.5e-13	685_bus	182	1.8e+02	7.4e-26	2.3e-06	3.6e-12
iFOM		293	2.7e+02	4.6e-27	4.1e-06	1.4e-13		188	1.2e+02	3.9e-26	1.0e-06	2.5e-11
CG		305	3.0e+02	1.8e-29	1.0e-04	2.5e-07		322	3.2e+02	4.8e-27	7.0e-07	4.4e-09
CGR		235	2.4e+02	1.9e-27	1.0e-04	2.3e-15		191	1.9e+02	1.7e-26	6.5e-07	3.2e-14
iCG		395	9.4e+01	2.5e-17	7.6e-06	6.4e-08		370	7.1e+01	2.6e-13	5.0e-06	3.4e-07
iCGR		293	6.8e+01	2.7e-17	4.1e-06	3.5e-10		188	2.9e+01	7.6e-12	1.0e-06	8.7e-07
FOM	nos1	220	2.2e+02	4.7e-22	2.1e-06	1.4e-10	nos7	270	2.7e+02	1.0e-18	1.8e-06	2.2e-08
iFOM		230	1.9e+02	2.0e-21	9.0e-09	1.4e-10		252	2.5e+02	4.2e-18	1.4e-05	2.0e-08
CG		711	7.1e+02	3.6e-23	3.1e-01	6.8e-07		2102	2.1e+03	1.5e-22	1.7e-07	2.7e-08
CGR		199	2.0e+02	1.1e-23	5.6e-05	1.7e-12		300	3.0e+02	1.2e-18	9.6e-08	1.0e-09
iCG		711	1.7e+02	1.1e-18	3.6e-01	7.7e-07		1097	1.5e+02	1.1e-07	1.0e-02	1.5e-04
iCGR		230	4.5e+01	2.4e-16	9.0e-09	6.4e-09		274	3.4e+01	1.7e-06	6.4e-06	9.9e-05

Table 4.15: Matrix Market: using practical algorithms in multi-precision with $\epsilon = 10^{-5}$

of multi-precision arithmetic, because the effective solution of symmetric positive-definite systems is obviously important beyond optimization.

While we have taken the point of view that the cost of matrix-vector products dominates the overall computational burden, a realistic assumption in large scale applications where this product often involves the application of several complicated operators (see [9] for example), the cost of orthogonalization (or re-orthogonalization) must not be neglected for algorithms that use it. Strategies to reduce this cost are therefore of interest. It is not the purpose of this paper to develop a rigorous analysis of variable accuracy orthogonalization techniques, but we defer this analysis to a future contribution.

Acknowledgment

The authors are indebted to Pr. S. Matsuoka (Riken) for an interesting conversation which confirmed their interest in multi-precision arithmetic in the context of very high performance computing, to Pr A. Podobas

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.
FOM	bcsstm02	11	1.1e+01	7.1e-32	2.5e-09	2.1e-16	nos4	60	6.0e+01	4.0e-28	2.4e-08	1.2e-15
iFOM		18	2.4e+00	1.5e-31	9.4e-10	8.7e-06		70	6.0e+01	1.4e-28	3.5e-13	8.6e-10
CG		11	1.1e+01	3.0e-32	2.5e-09	0.0e+00		65	6.5e+01	3.9e-29	4.2e-10	1.0e-09
CGR		11	1.1e+01	1.1e-31	2.5e-09	2.1e-16		64	6.4e+01	1.7e-28	6.7e-10	4.7e-15
iCG		20	2.4e+00	9.4e-09	9.4e-09	7.5e-06		72	2.3e+01	6.1e-13	2.0e-12	7.2e-07
iCGR		21	2.4e+00	8.8e-09	8.8e-09	2.2e-06		70	1.9e+01	1.7e-11	1.7e-11	1.3e-06
FOM	bcsstk09	185	1.8e+02	7.3e-27	2.5e-08	1.6e-13	bcsstk05	127	1.3e+02	2.7e-27	2.3e-08	5.0e-13
iFOM		195	8.6e+01	1.0e-26	3.6e-09	1.9e-13		137	4.0e+01	2.0e-27	9.5e-12	8.2e-12
CG		140	1.4e+02	3.2e-27	5.9e-06	4.8e-08		193	1.9e+02	6.6e-28	7.6e-07	7.1e-08
CGR		140	1.4e+02	3.9e-27	5.7e-06	1.7e-14		121	1.2e+02	1.2e-27	7.2e-07	8.8e-15
iCG		195	1.8e+01	7.6e-21	4.9e-09	3.0e-09		232	2.0e+01	1.1e-16	2.1e-08	3.5e-08
iCGR		195	1.8e+01	2.0e-20	3.8e-09	4.0e-11		137	1.1e+01	4.4e-15	9.5e-12	1.7e-08
FOM	bcsstk27	377	3.8e+02	6.8e-27	2.4e-08	5.2e-13	685_bus	213	2.1e+02	3.1e-26	2.1e-08	3.5e-12
iFOM		375	3.5e+02	3.4e-27	2.7e-08	3.7e-13		222	1.6e+02	1.3e-26	3.8e-09	1.6e-11
CG		449	4.5e+02	2.6e-29	1.1e-06	3.7e-09		385	3.8e+02	3.1e-27	7.4e-09	2.1e-10
CGR		317	3.2e+02	1.9e-27	1.0e-06	3.7e-15		219	2.2e+02	8.5e-27	6.4e-09	2.7e-14
iCG		519	1.2e+02	1.0e-18	1.5e-07	3.4e-09		445	9.8e+01	6.0e-15	7.7e-08	6.4e-08
iCGR		375	8.7e+01	1.4e-18	2.7e-08	3.1e-11		222	4.5e+01	1.2e-13	3.8e-09	1.1e-07
FOM	nos1	226	2.3e+02	2.6e-22	1.9e-08	1.4e-10	nos7	315	3.2e+02	8.2e-19	2.0e-08	2.0e-08
iFOM		236	2.0e+02	5.7e-22	9.7e-12	1.4e-10		311	3.1e+02	2.8e-18	6.3e-08	2.1e-08
CG		711	7.1e+02	3.6e-23	3.1e-01	6.8e-07		2187	2.2e+03	6.7e-19	7.3e-08	5.4e-10
CGR		222	2.2e+02	2.8e-23	4.5e-07	6.3e-13		341	3.4e+02	1.6e-18	1.0e-09	1.5e-09
iCG		711	1.7e+02	1.8e-18	3.6e-01	1.7e-06		2040	3.4e+02	3.7e-09	5.3e-05	6.7e-06
iCGR		236	4.8e+01	1.6e-16	1.0e-11	5.4e-09		331	6.0e+01	3.6e-08	5.7e-08	1.1e-05

Table 4.16: Matrix Market: using practical algorithms in multi-precision with $\epsilon = 10^{-7}$

(Tokyo Institute of Technology) for providing further pointers on computer architecture, and to Pr. M. Daydé (IRIT) for his continued and friendly support.

References

[1] M. Arioli. A stopping criterion for the conjugate gradient algorithm in a finite element framework. *Numerische Mathematik*, 97:1–24, 2004.

[2] M. Arioli. Generalized Golub-Kahan bidiagonalization and stopping criteria. *SIAM Journal on Matrix Analysis*, 34(2):571–592, 2013.

[3] M. Arioli, I. S. Duff, and D. Ruiz. Stopping criteria for iterative solvers. *SIAM Journal on Matrix Analysis and Applications*, 13(1):138–144, 1992.

[4] M. Arioli and S. Gratton. Linear regression models, least-squares problems, normal equations, and stopping criteria for the conjugate-gradient method. *Computer Physics Communications*, 183:2322–2336, 2012.

[5] M. Arioli, D. Loghin, and A. Wathen. Stopping criteria for iterations in finite element methods. *Numerische Mathematik*, 99(3):381–410, 2004.

[6] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. *Trust-Region Methods*. MPS-SIAM Series on Optimization. SIAM, Philadelphia, USA, 2000.

[7] S. Galal and M. Horowitz. Energy-efficient floating-point unit design. *IEEE Transactions on Computers*, 60(7), 2011.

[8] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, third edition, 1996.

[9] S. Gratton, S. Gürol, E. Simon, and Ph. L. Toint. On the use of the saddle formulation in weakly constrained 4D-Var data assimilation. arXiv:1709.06383, 2017.

[10] S. Gratton, Ph. L. Toint, and J. Tshimanga. Range-space variants and inexact matrix-vector products in Krylov solvers for linear systems arising from inverse problems. *SIAM Journal on Matrix Analysis*, 32(3):969–986, 2011.

[11] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of the National Bureau of Standards*, 49:409–436, 1952.

[12] N. J. Higham. The rise of multiprecision computations. Talk at SAMSI 2017, April 2017. `https://bit-ly/higham-samsi2017`.

[13] C. T. Kelley. *Iterative Methods for Linear and Nonlinear Equations*. Frontiers in Applied Mathematics. SIAM, Philadelphia, USA, 1995.

[14] S. Matsuoka. private communication, March 2018.

[15] J. Pu, S. Galal, X. Yang, O. Shacham, and M. Horowitz. FPMax: a 106GFLOPS/W at 217GFLOPS/mm2 single-precision FPU, and a 43.7 GFLOPS/W at 74.6 GFLOPS/mm2 double-precision FPU, in 28nm UTBB FDSOI. *Hardware Architecture*, 2016.

[16] Y. Saad. *Iterative Methods for Sparse Linear Systems*. PWS Publishing Company, Boston, USA, 1996.

[17] V. Simoncini and D. Szyld. Theory of inexact Krylov subspace methods and applications to scientific computing. *SIAM Journal on Scientific Computing*, 25(2):454–477, 2003.

[18] J. van den Eshof and G. L. G. Sleijpen. Inexact Krylov subspace methods for linear systems. *SIAM Journal on Matrix Analysis and Applications*, 26(1):125–153, 2004.

[19] J. van den Eshof, G. L. G. Sleijpen, and M. B. van Gijzen. Relaxation strategies for nested Krylov methods. *Journal of Computational and Applied Mathematics*, 177:347–365, 2005.