QUARTERLY JOURNAL OF THE ROYAL METEOROLOGICAL SOCIETY
Q. J. R. Meteorol. Soc. **135**: 1573–1585 (2009)
Published online in Wiley InterScience
(www.interscience.wiley.com) DOI: 10.1002/qj.477

An observation-space formulation of variational assimilation using a restricted preconditioned conjugate gradient algorithm

Serge Gratton[a,b]* and Jean Tshimanga[a]
[a]*CERFACS, Toulouse, France*
[b]*CNES, Toulouse, France*

ABSTRACT: We consider parameter estimation problems involving a set of m physical observations, where an unknown vector of n parameters is defined as the solution of a nonlinear least-squares problem. We assume that the problem is regularized by a quadratic penalty term. When solution techniques based on successive linearization are considered, as in the incremental four-dimensional variational (4D-Var) techniques for data assimilation, a sequence of linear systems with particular structure has to be solved. We exhibit a subspace of dimension m that contains the solution of these linear systems, and derive a variant of the conjugate gradient algorithm that is more efficient in terms of memory and computational costs than its standard form, when m is smaller than n. The new algorithm, which we call the Restricted Preconditioned Conjugate Gradient (RPCG), can be viewed as an alternative to the so-called Physical-space Statistical Analysis System (PSAS) algorithm, which is another approach to solve the linear problem. In addition, we show that the non-monotone and somehow chaotic behaviour of PSAS algorithm when viewed in the model space, experimentally reported by some authors, can be fully suppressed in RPCG.

Moreover, since preconditioning and re-orthogonalization of residuals vectors are often used in practice to accelerate convergence in high-dimension data assimilation, we show how to reformulate these techniques within subspaces of dimension m in RPCG. Numerical experiments are reported, on an idealized data assimilation system based on the heat equation, that clearly show the effectiveness of our algorithm for large-scale problems. Copyright © 2009 Royal Meteorological Society

KEY WORDS reduced dimension; 4D-Var minimization; limited memory

Received 27 March 2009; Revised 12 June 2009; Accepted 22 June 2009

1. Introduction

The aim of four-dimensional variational (4D-Var) data assimilation is to find a *model* state $\mathbf{x}(t_0)$ at an initial time t_0, such that some distance between the *trajectory* of the model and a set of *observations* $\mathbf{y}_j^{\mathrm{o}}$ at times t_j, $j = 0, 1, \ldots, p$, is minimized, subject to $\mathbf{x}(t_0)$ remaining close to a prior estimate $\mathbf{x}^{\mathrm{b}}$, also known as a *background* field. Mathematically, this can be expressed as a nonlinear weighted least-squares problem, for which we wish to find the model state $\mathbf{x}(t_0)$ that minimizes the 4D-Var functional

$$\mathcal{J}[\mathbf{x}(t_0)] = \frac{1}{2}\{\mathbf{x}(t_0) - \mathbf{x}^{\mathrm{b}}\}^{\mathrm{T}}\mathbf{B}^{-1}\{\mathbf{x}(t_0) - \mathbf{x}^{\mathrm{b}}\} + \frac{1}{2}\sum_{j=0}^{p}\{\mathcal{H}_j[\mathbf{x}(t_j)] - \mathbf{y}_j^{\mathrm{o}}\}^{\mathrm{T}}\mathbf{R}_j^{-1}\{\mathcal{H}_j[\mathbf{x}(t_j)] - \mathbf{y}_j^{\mathrm{o}}\}, \tag{1}$$

where the state vector $\mathbf{x}(t_j)$ satisfies the discrete nonlinear model of evolution $\mathbf{x}(t_j) = \mathcal{M}_{j,0}[\mathbf{x}(t_0)]$. Here, the matrices $\mathbf{B}$ and $\mathbf{R}_j$ are respectively the background-error and observation-error covariance matrices, and $\mathcal{H}_j$ is an operator that maps the model field at time t_j to the observation space. Moreover, the assumption that observation errors are uncorrelated has been made.

For the problem of numerical weather prediction, the size n of the state vector is very large and situations where $n = 2 \times 10^8$ are addressed on a daily basis (Rabier, 2005). We focus in this paper on the case where m, the number of physical observations, is *significantly smaller* than n, and propose a *new algorithmic* approach to take this feature into account. Our approach is attractive because it significantly reduces the memory storage needed by the minimization algorithm, *without alteration* of its convergence properties.

Due to problem size, efficient methods are needed to minimize the function $\mathcal{J}$ in Equation (1). A common way of treating this problem is by a technique known as the *incremental method* (Courtier *et al.*, 1994), which has been shown to be equivalent to applying a *truncated* Gauss–Newton iteration to minimize a nonlinear least-squares function (Gratton *et al.*, 2007). This algorithm

*Correspondence to: Serge Gratton, CERFACS, 42 avenue Gaspard Coriolis, 31057 Toulouse Cedex 01, France.
E-mail: serge.gratton@cerfacs.fr

can be represented by the inner-outer Algorithm 1, in which a sequence $\{\mathbf{x}^{(k)}(t_0)\}_{k=0,\dots}$ of approximations of the solution is obtained by solving a sequence of quadratic minimization problems. Increment vectors $\delta\mathbf{x}_0^{(k)}$ are obtained as (approximate) minimizers of the quadratic functions $J^{(k)}$ defined by

$$J^{(k)}[\delta\mathbf{x}_0] = \frac{1}{2}\{\delta\mathbf{x}_0 - [\mathbf{x}^\mathrm{b} - \mathbf{x}^{(k)}(t_0)]\}^\mathrm{T}\mathbf{B}^{-1}\{\delta\mathbf{x}_0 - [\mathbf{x}^\mathrm{b} - \mathbf{x}^{(k)}(t_0)]\} + \frac{1}{2}\sum_{j=0}^{p}(\mathbf{H}_j^{(k)}\delta\mathbf{x}_j - \mathbf{d}_j^{(k)})^\mathrm{T}\mathbf{R}_j^{-1}(\mathbf{H}_j^{(k)}\delta\mathbf{x}_j - \mathbf{d}_j^{(k)}), \quad (2)$$

and used to perform the update

$$\mathbf{x}^{(k+1)}(t_0) = \mathbf{x}^{(k)}(t_0) + \delta\mathbf{x}_0^{(k)}.$$

In the above definition of $J^{(k)}$, we have used the following notation:

- $\mathbf{H}_j^{(k)}$ is a (possibly approximate) linearization of the observation operator $\mathcal{H}_j$ around the state $\mathbf{x}^{(k)}(t_j)$;
- the misfit vector is defined by
$$\mathbf{d}_j^{(k)} = \mathbf{y}_j^\mathrm{o} - \mathcal{H}_j[\mathbf{x}^{(k)}(t_j)];$$
- $\mathbf{M}_{j,0}^{(k)}$ is a (possibly approximate) linearization of the model $\mathcal{M}_{j,0}$ around $\mathbf{x}_0 = \mathbf{x}^{(k)}(t_0)$;
- $\delta\mathbf{x}_j = \mathbf{M}_{j,0}^{(k)}\,\delta\mathbf{x}_0$.

Each iteration of the steps 3, 4, 5 and 6 of Algorithm 1 is known as an *outer loop*. Within each outer iteration, we must solve the minimization problem involving $J^{(k)}$ at step 5 by an iterative procedure. (Direct methods based on matrix factorizations are not practicable in general, due to the size and the lack of sparsity of the matrices involved in the problem.) This procedure is known as the *inner loop* minimization. An important question for this algorithm is the choice of stopping criteria for the inner and outer iterations that enables us to obtain a good minimizer $\mathbf{x}^{(k)}(t_j)$ of the 4D-Var functional with the smallest computational time, see Lawless and Nichols (2006) for a discussion. Finally, the analysis field at the initial time is then given by $\mathbf{x}_0^\mathrm{a} = \mathbf{x}_0^{(k)}$, where k is the outer-loop index where the iterations are stopped.

The present paper presents in section 2 the primal and dual minimization approaches that are commonly used for the inner loop. Section 3 is the principal section of this paper, where we show how to derive a conjugate-gradient-like method that is mathematically equivalent to the primal approach, while making the quadratic minimization in a space of dimension $\Re^m$. This new approach can therefore be seen as a new algorithm (depicted in Algorithm 5 and Algorithm 8, for different starting points) that possesses the same convergence properties as the primal approach while keeping the memory cost similar to that of the dual approach. This new algorithm also allows preconditioning and re-orthogonalization which make it suitable for large-scale applications. The re-orthogonalization issue is illustrated in section 3.6. In section 3.7 we present an extension of the new algorithm and numerical experiments are reported in section 4 to illustrate the behaviour of the algorithm. Conclusions are finally given in section 5.

Algorithm 1. Incremental 4D-Var

```
1: For the first iteration, $k = 0$, we choose $\mathbf{x}_0^{(0)} = \mathbf{x}^\mathrm{b}$, the background state.
2: **while** convergence criterion not satisfied **do**
3:   Run the nonlinear model to calculate $\mathbf{x}^{(k)}(t_j)$ at each time step $t_j$ from $\mathbf{x}^{(k)}(t_0)$.
4:   For each observation, calculate the innovation vectors $\mathbf{d}_j^{(k)} = \mathbf{y}_j^\mathrm{o} - \mathcal{H}_j[\mathbf{x}^{(k)}(t_j)]$.
5:   Find the value of $\delta\mathbf{x}_0^{(k)}$ that approximately minimizes the incremental cost function defined in (2).
6:   Increment $k$ and perform the update
       $\mathbf{x}^{(k+1)}(t_0) = \mathbf{x}^{(k)}(t_0) + \delta\mathbf{x}_0^{(k)}$.   (3)
7: **end while**
```

2. Inner loop minimization: Primal and dual formulation

To clarify the presentation of the inner loop algorithms, we simplify the notation. Since we are primarily interested in the computations performed in a *single inner loop*, we suppress the outer loop superscript k. In addition, we concatenate all observations into one vector and we introduce the following notation:

$$\mathbf{d} = \begin{bmatrix} \mathbf{d}_0 \\ \mathbf{d}_1 \\ \vdots \\ \mathbf{d}_\mathrm{N} \end{bmatrix} \in \Re^m, \quad \mathbf{H} = \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1\mathbf{M}_{1,0} \\ \vdots \\ \mathbf{H}_\mathrm{p}\mathbf{M}_{\mathrm{p},0} \end{bmatrix} \in \Re^{n\times m},$$

and $\mathbf{R} = \mathrm{diag}(\mathbf{R}_0, \mathbf{R}_1, \dots, \mathbf{R}_\mathrm{p}) \in \Re^{m\times m}$ is the block diagonal matrix whose ℓth diagonal block is $\mathbf{R}_\ell$. The quadratic function J of step 5 can then be written more compactly as

$$\begin{aligned} J[\delta\mathbf{x}_0] &= \tfrac{1}{2}\{\delta\mathbf{x}_0 - [\mathbf{x}^\mathrm{b} - \mathbf{x}_0]\}^\mathrm{T}\mathbf{B}^{-1}\{\delta\mathbf{x}_0 - [\mathbf{x}^\mathrm{b} - \mathbf{x}_0]\} \\ &\quad + \tfrac{1}{2}(\mathbf{H}\delta\mathbf{x}_0 - \mathbf{d})^\mathrm{T}\mathbf{R}^{-1}(\mathbf{H}\delta\mathbf{x}_0 - \mathbf{d}). \end{aligned} \quad (4)$$

Equating to zero the derivative of J with respect to $\delta\mathbf{x}_0$ yields the following linear system, which represents the first-order necessary and sufficient optimality condition for the minimization of J:

$$(\mathbf{B}^{-1} + \mathbf{H}^\mathrm{T}\mathbf{R}^{-1}\mathbf{H})\delta\mathbf{x}_0 = \mathbf{B}^{-1}(\mathbf{x}^\mathrm{b} - \mathbf{x}_0) + \mathbf{H}^\mathrm{T}\mathbf{R}^{-1}\mathbf{d}. \quad (5)$$

It is well known that the preconditioned conjugate gradient method (PCG; Hestenes and Stiefel, 1952) is one of the most successful approach for solving linear systems with a symmetric positive definite matrix like Equation (5). When suitably preconditioned, and for a well-chosen starting point $\delta\mathbf{x}_0^0$, it often provides an acceptable approximation of the solution within a few steps (van der Sluis and van der Vorst, 1986; Golub and Van Loan, 1996). The method is therefore commonly used in data assimilation systems, either in its standard form (Hestenes and Stiefel, 1952), or in equivalent formulations based on either the Lanczos procedure (CONGRAD in Fisher, 1998), or quasi-Newton approaches (M1QN3 in Gilbert and Lemaréchal, 1989).

The standard PCG merely requires the storage of four n-dimensional vectors. In terms of computational cost per iteration, only few vector–vector operations – two scalar products and three vector sums (Nocedal and Wright, 1999, p 111) – are performed in addition to one matrix–vector product involving the matrix $\mathbf{B}^{-1}+\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{H}$; this matrix is never stored, but its action on a vector is evaluated.

In *large-scale* data assimilation problems, where evaluating this matrix–vector product is the major computational task, any additional technique that can accelerate, even by a few steps, the convergence of PCG is therefore of interest. Two such techniques are commonly used in operational systems: limited-memory preconditioning (Fisher 1998; Morales and Nocedal, 2000; Tshimanga *et al.*, 2008) and re-orthogonalization (Fisher, 1998). In limited-memory preconditioning (Tshimanga, 2007, provides an overview), *a set of vectors* of size n is retained from past PCG computations, and used together with $\mathbf{B}$, to derive an approximation of the inverse of $\mathbf{B}^{-1}+\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{H}$; this approximation is then used to precondition the PCG iterations on the next outer iteration. Re-orthogonalization techniques stem from the observation that, because of computing round-off errors, the PCG residuals vectors tend to loose their conjugacy properties, thereby degrading the convergence rate (van der Vorst, 2003, p 123). A remedy to this phenomenon is to re-conjugate the current residual with respect to the previous residuals using, for example, Gram–Schmidt procedures. Re-orthogonalization requires the storage of the residuals from past iterations (Fisher, 1998). Both acceleration techniques (preconditioning and re-orthogonalization) therefore require the storage of several vectors of size n, which is expected to become a serious problem for future data assimilation systems. In particular, the dimension n of the problem is expected to increase significantly with the advent of so-called weak constraint 4D-Var problems (Trémolet, 2007, 2006) which include additional variables to the control vector to account for the model error. The main idea that has been used to alleviate the memory cost is to exploit the quadratic structure on the inner-loop problem to replace a minimization in a space of dimension n with a minimization in the space of dimension m. This approach relies on the fact that the solution of Equation (5) can be written

$$\mathbf{x}^{\mathrm{b}}-\mathbf{x}_0 \\ +(\mathbf{B}^{-1}+\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{H})^{-1}\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\{\mathbf{d}-\mathbf{H}(\mathbf{x}^{\mathrm{b}}-\mathbf{x}_0)\}, \quad (6)$$

or equivalently, using the Sherman–Morrison–Woodbury formula (Golub and Van Loan, 1996, p 50; Conn *et al.*, 2000, p 57),

$$\mathbf{x}^{\mathrm{b}}-\mathbf{x}_0 \\ +\mathbf{B}\mathbf{H}^{\mathrm{T}}(\mathbf{R}+\mathbf{H}\mathbf{B}\mathbf{H}^{\mathrm{T}})^{-1}\{\mathbf{d}-\mathbf{H}(\mathbf{x}^{\mathrm{b}}-\mathbf{x}_0)\}. \quad (7)$$

A data assimilation algorithm based on Equation (7) is described in Courtier (1997) in which the solution of the m-order symmetric and positive definite linear system $(\mathbf{R}+\mathbf{H}\mathbf{B}\mathbf{H}^{\mathrm{T}})\delta\widetilde{\mathbf{x}}=\mathbf{d}-\mathbf{H}(\mathbf{x}^{\mathrm{b}}-\mathbf{x}_0)$ is iteratively approximated (in our case we could use PCG for this task). If $\delta\widetilde{\mathbf{x}}_i\in\Re^m$ stands for the iterate corresponding to iteration i where PCG is stopped, the approximate solution to the inner-loop problem is recovered readily from $\mathbf{x}^{\mathrm{b}}-\mathbf{x}_0+\mathbf{B}\mathbf{H}^{\mathrm{T}}\delta\widetilde{\mathbf{x}}_i\in\Re^n$ using Equation (7). In the present paper, the approach described in Courtier (1997) is referred to as the *dual* approach, and we use the term *primal* approach for the corresponding approach based on Equation (6).

Comparing the dual and primal approaches in general is not an easy task. Many points should be taken into account for a fair comparison, including the availability of good preconditioners, the choice of reasonable starting points for the quadratic minimizations, and the existence of efficient parallel implementations on modern computers. Such a comparison is out of the scope of the present paper and we just provide now a brief overview of the principal well-known facts concerning the two approaches. All vectors in the iterative part of the dual approach are in $\Re^m$, which is advantageous in terms of storage or computational cost if $m \ll n$. In addition, concerning condition numbers, which are known to be important for describing the convergence of iterative algorithms for linear systems, the following result holds: when $\mathbf{R}^{-1}$ is used as a preconditioner in the dual approach, the same condition number is obtained as in the corresponding primal approach preconditioned by $\mathbf{B}$ (Courtier, 1997).

Let us consider now the merits of the approximate solutions obtained when the primal and dual inner loops are truncated before full convergence. (If the iterative solvers used for the linear systems could be pushed to full convergence, the two approaches would give the same result.) It has been seen that, when a zero initial guess is taken and when few steps are performed within the inner loop, the approximate solution obtained from the dual approach may be a very bad approximation of the solution of Equation (5) (Auroux, 2007; El Akkraoui *et al.*, 2008). When PCG is considered in the inner loop, we shall show an example in section 4 where the incremental cost function J minimized at step 5 of the incremental algorithm may even *increase* during the initial iterations of the inner loop of the dual algorithm. This situation is theoretically excluded when the primal approach is considered, because PCG is precisely minimizing J on an increasing sequence of nested Krylov subspaces (Nocedal and Wright 1999). A mathematical study of the behaviour of the dual iterates versus the primal iterates is however not an easy task; we therefore rely on numerical experiments in (Auroux, 2007; El Akkraoui *et al.*, 2008) and section 4 of the present work for the comparison of the two approaches. All these results explain why the dual approach may not be convenient for cases where performing a large number of iterations is not allowed due, for example, to the unacceptable computational cost that would then be implied (Lawless *et al.*, 2005a,b; Gratton *et al.*, 2007).

3. A restricted preconditioned conjugate gradient algorithm (RPCG)

Let us begin with a short review of the preconditioned conjugate gradient algorithm for solving large symmetric positive definite linear systems. It is widely recognized that there is no universal way to design a preconditioner for all types of problems (Benzi, 2002). For a symmetric and positive-definite linear system $\mathbf{Ax} = \mathbf{b}$, a preconditioning matrix $\mathbf{F}$ for CG is a symmetric and positive definite matrix that accelerates the convergence. Efficient preconditioners often have some of the following properties:

1. The condition number of $\mathbf{FA}$ is smaller than that of $\mathbf{A}$;
2. The eigenvalues of $\mathbf{FA}$ are more clustered than those of $\mathbf{A}$;
3. $\mathbf{F}$ is cheap to compute and straightforward to apply.

It should be stressed that designing preconditioners is a very broad and active area of research (e.g. van der Vorst, 2003).

3.1. The preconditioned conjugate gradient algorithm in exact arithmetic

Algorithm 2 is the PCG method (Golub and Van Loan, 1996; van der Vorst 2003), with preconditioning matrix $\mathbf{F}$, for the solution of an n-by-n symmetric and positive definite system $\mathbf{Av} = \mathbf{b}$. This iterative algorithm starts from an initial guess $\mathbf{v}_0$ and generates a sequence of iterates $\mathbf{v}_i$ converging to the solution of the linear system in less than n steps.

We mention, for further use, the following useful properties (Axelsson, 1996, pp 472–473; Theorem 11.5) of the PCG algorithm:

1. The iterate $\mathbf{v}_i$ minimizes the quadratic function $\mathbf{v}^{\mathrm{T}}\mathbf{Av}/2 - \mathbf{v}^{\mathrm{T}}\mathbf{b} + \mathbf{c}$ over the subspace $\mathbf{v}_0+$ span $(\mathbf{r}_0, \mathbf{AFr}_0, \ldots, (\mathbf{AF})^{i-1}\mathbf{r}_0)$, where $\mathbf{r}_0 = \mathbf{Av}_0 - \mathbf{b}$;
2. The residuals vectors $\mathbf{r}_i = \mathbf{Av}_i - \mathbf{b}$ are mutually *conjugate* w.r.t. $\mathbf{F}$. This means that $\mathbf{r}_i^{\mathrm{T}}\mathbf{Fr}_j = 0$ whenever $i \neq j$.
3. The descent directions $\mathbf{p}_i$ are mutually *conjugate* w.r.t. $\mathbf{A}$, i.e. $\mathbf{p}_i^{\mathrm{T}}\mathbf{Ap}_j = 0$ whenever $i \neq j$.

Properties 2 and 3 may not hold when the method is used in the presence of round-off errors. This is why a re-orthogonalization may be implemented to cope with this problem.

Now, we adapt the PCG to Equation (5) to take into account the structure of the problem. This iterative algorithm generates at step i of the inner loop an approximate of the solution of Equation (5) which we simply denote by $\mathbf{v}_i$. Our goal is of course to write the recurrences of PCG in terms of vectors of $\Re^m$ to minimize as much as possible the memory space needed by the algorithm.

3.2. Derivation of the algorithm

Assume that Algorithm 2 is applied to solve Equation (5) using a preconditioning matrix $\mathbf{F}$. We obtain Algorithm 3 in which, at each iteration i, the vectors $\mathbf{v}_i$, $\mathbf{r}_i$, $\mathbf{q}_i$, $\mathbf{z}_i$ and $\mathbf{p}_i$ belong to $\Re^n$.

To further transform the algorithm we assume that there exists a matrix $\mathbf{G} \in \Re^{m\times m}$ such that the preconditioner satisfies

$$\textbf{Assumption 1}: \quad \mathbf{BH}^{\mathrm{T}}\mathbf{G} = \mathbf{FH}^{\mathrm{T}}. \tag{8}$$

Note that if $\mathbf{F}$ is given, the m^2 entries of $\mathbf{G}$ are defined in Equation (8) as a solution of a linear system with nm equations. Consequently, if $m < n$, the system is overdetermined, and may not have any solution. However, we shall see in section 3.4 that Assumption 1 is not overly restrictive, since it is satisfied e.g. by the limited memory preconditioners used in some large-scale applications (Fisher, 1998). Note also that for $\mathbf{F} = \mathbf{B}$ (which corresponds to the preconditioning by the background-error covariance matrix), a possible choice that obeys Assumption 1 is simply $\mathbf{G} = \mathbf{I}_m$, where $\mathbf{I}_m$ is the identity matrix of order m.

Our approach to keep the conjugate gradient vectors in spaces of dimension m is to detect when $\mathbf{H}^{\mathrm{T}}$ can be factored in the vectors generated by PCG. We shall see now that this can be easily done if the following holds:

$$\textbf{Assumption 2}: \quad \mathbf{v}_0 = \mathbf{x}^{\mathrm{b}} - \mathbf{x}_0. \tag{9}$$

Before going any further, let us make a comment on this assumption on $\mathbf{v}_0$. We see that for this particular value, the initial background term in Equation (4) is

$$\frac{1}{2}(\mathbf{v}_0 - [\mathbf{x}^{\mathrm{b}} - \mathbf{x}_0])^{\mathrm{T}}\mathbf{B}^{-1}(\mathbf{v}_0 - [\mathbf{x}^{\mathrm{b}} - \mathbf{x}_0]) = 0.$$

Algorithm 2. Preconditioned Conjugate Gradient

```
1:   $\mathbf{v}_0 =$ initial estimate
2:   $\mathbf{r}_0 = \mathbf{Av}_0 - \mathbf{b}$
3:   $\mathbf{z}_0 = \mathbf{Fr}_0$
4:   $\mathbf{p}_0 = -\mathbf{z}_0$
5:   $i = 0$
6:   **while** stopping criterion not satisfied **do**
7:   $i = i + 1$
8:   $\mathbf{q}_{i-1} = \mathbf{Ap}_{i-1}$
9.   $\alpha_{i-1} = \mathbf{r}_{i-1}^{\mathrm{T}}\mathbf{z}_{i-1}/\mathbf{q}_{i-1}^{\mathrm{T}}\mathbf{p}_{i-1}$
10:  $\mathbf{v}_i = \mathbf{v}_{i-1} + \alpha_{i-1}\mathbf{p}_{i-1}$
11:  $\mathbf{r}_i = \mathbf{r}_{i-1} + \alpha_{i-1}\mathbf{q}_{i-1}$
12:  $\mathbf{z}_i = \mathbf{Fr}_i$
13:  $\beta_i = \mathbf{r}_i^{\mathrm{T}}\mathbf{z}_i/\mathbf{r}_{i-1}^{\mathrm{T}}\mathbf{z}_{i-1}$
14:  $\mathbf{p}_i = -\mathbf{z}_i + \beta_i\mathbf{p}_{i-1}$
15:  **endwhile**.
```

Therefore, with this assumption the PCG iterations are started with a *zero background* term. This choice is, for example, made in Weaver *et al.* (2003) and others. Note that using the background as the starting point of the minimization is often a reasonable choice: if a better approximation of the solution was known before starting the assimilation, it would likely be used as the background. We shall however see that this assumption can be relaxed in section 3.7, using a more complicated derivation of the algorithm.

For the rest of this section, we suppose that Assumptions 1 and 2 hold. Letting

$$\begin{aligned}\widehat{\mathbf{r}}_0 &= \mathbf{R}^{-1}\mathbf{H}\mathbf{v}_0 - \mathbf{R}^{-1}\mathbf{d},\\ \widehat{\mathbf{z}}_0 &= \mathbf{G}\widehat{\mathbf{r}}_0,\\ \widehat{\mathbf{p}}_0 &= -\widehat{\mathbf{z}}_0\\ \widehat{\mathbf{v}}_0 &= 0,\end{aligned}$$

we then have

$$\mathbf{r}_0 = \mathbf{H}^{\mathrm{T}}\widehat{\mathbf{r}}_0, \tag{10}$$

$$\mathbf{p}_0 = \mathbf{B}\mathbf{H}^{\mathrm{T}}\widehat{\mathbf{p}}_0, \tag{11}$$

$$\mathbf{z}_0 = \mathbf{B}\mathbf{H}^{\mathrm{T}}\widehat{\mathbf{z}}_0. \tag{12}$$

In addition, from

$$\begin{aligned}\mathbf{q}_0 &= (\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{H} + \mathbf{B}^{-1})\mathbf{p}_0\\ &= (\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{H} + \mathbf{B}^{-1})\mathbf{B}\mathbf{H}^{\mathrm{T}}\widehat{\mathbf{p}}_0\\ &= \mathbf{H}^{\mathrm{T}}(\mathbf{R}^{-1}\mathbf{H}\mathbf{B}\mathbf{H}^{\mathrm{T}} + \mathbf{I}_m)\widehat{\mathbf{p}}_0,\end{aligned}$$

we also have

$$\mathbf{q}_0 = \mathbf{H}^{\mathrm{T}}\widehat{\mathbf{q}}_0,$$

where $\widehat{\mathbf{q}}_0 = (\mathbf{R}^{-1}\mathbf{H}\mathbf{B}\mathbf{H}^{\mathrm{T}} + \mathbf{I}_m)\widehat{\mathbf{p}}_0$.

Suppose now that, as just seen for $i = 0$, vectors $\widehat{\mathbf{r}}_i$, $\widehat{\mathbf{p}}_i$, $\widehat{\mathbf{v}}_i$, $\widehat{\mathbf{z}}_i$ and $\widehat{\mathbf{q}}_i$ exist for a given i, such that

$$\mathbf{r}_i = \mathbf{H}^{\mathrm{T}}\widehat{\mathbf{r}}_i, \tag{13}$$

$$\mathbf{p}_i = \mathbf{B}\mathbf{H}^{\mathrm{T}}\widehat{\mathbf{p}}_i, \tag{14}$$

$$\mathbf{v}_i = \mathbf{v}_0 + \mathbf{B}\mathbf{H}^{\mathrm{T}}\widehat{\mathbf{v}}_i, \tag{15}$$

$$\mathbf{z}_i = \mathbf{B}\mathbf{H}^{\mathrm{T}}\widehat{\mathbf{z}}_i, \tag{16}$$

$$\mathbf{q}_i = \mathbf{H}^{\mathrm{T}}\widehat{\mathbf{q}}_i \tag{17}$$

hold. We will see now that it is possible to define quantities at step $i+1$ such that relations in Equations (13)–(17) hold with the index 'i' replaced by '$i+1$'. Indeed, using the PCG algorithm, we obtain for the vector $\mathbf{v}_{i+1}$ that

$$\begin{aligned}\mathbf{v}_{i+1} = \mathbf{v}_i + \alpha_i\mathbf{p}_i &= \mathbf{v}_0 + \mathbf{B}\mathbf{H}^{\mathrm{T}}\widehat{\mathbf{v}}_i + \alpha_i\mathbf{B}\mathbf{H}^{\mathrm{T}}\widehat{\mathbf{p}}_i\\ &= \mathbf{v}_0 + \mathbf{B}\mathbf{H}^{\mathrm{T}}(\widehat{\mathbf{v}}_i + \alpha_i\widehat{\mathbf{p}}_i)\\ &= \mathbf{v}_0 + \mathbf{B}\mathbf{H}^{\mathrm{T}}\widehat{\mathbf{v}}_{i+1},\end{aligned}$$

where we set $\widehat{\mathbf{v}}_{i+1} = \widehat{\mathbf{v}}_i + \alpha_i\widehat{\mathbf{p}}_i$. For the vector $\mathbf{r}_{i+1}$, we see that

$$\begin{aligned}\mathbf{r}_{i+1} &= \mathbf{r}_i + \alpha_{i-1}\mathbf{q}_i\\ &= \mathbf{H}^{\mathrm{T}}\widehat{\mathbf{r}}_i + \alpha_{i-1}\mathbf{H}^{\mathrm{T}}\widehat{\mathbf{q}}_i\\ &= \mathbf{H}^{\mathrm{T}}\widehat{\mathbf{r}}_{i+1},\end{aligned}$$

where $\widehat{\mathbf{r}}_{i+1} = \widehat{\mathbf{r}}_i + \alpha_{i-1}\widehat{\mathbf{q}}_i$. Concerning the vector $\mathbf{z}_{i+1}$, we have

$$\begin{aligned}\mathbf{z}_{i+1} &= \mathbf{F}\mathbf{r}_{i+1}\\ &= \mathbf{F}\mathbf{H}^{\mathrm{T}}\widehat{\mathbf{r}}_{i+1}\\ &= \mathbf{B}\mathbf{H}^{\mathrm{T}}\mathbf{G}\widehat{\mathbf{r}}_{i+1}\\ &= \mathbf{B}\mathbf{H}^{\mathrm{T}}\widehat{\mathbf{z}}_{i+1},\end{aligned}$$

where $\widehat{\mathbf{z}}_{i+1} = \mathbf{G}\widehat{\mathbf{r}}_{i+1}$. And finally for the vector $\mathbf{p}_{i+1}$, from

$$\begin{aligned}\mathbf{p}_{i+1} &= -\mathbf{z}_{i+1} + \beta_{i+1}\mathbf{p}_i\\ &= -\mathbf{B}\mathbf{H}^{\mathrm{T}}\widehat{\mathbf{z}}_{i+1} + \beta_{i+1}\mathbf{B}\mathbf{H}^{\mathrm{T}}\widehat{\mathbf{p}}_1\\ &= \mathbf{B}\mathbf{H}^{\mathrm{T}}(-\widehat{\mathbf{z}}_{i+1} + \beta_{i+1}\widehat{\mathbf{p}}_1),\end{aligned}$$

we get

$$\mathbf{p}_{i+1} = \mathbf{B}\mathbf{H}^{\mathrm{T}}\widehat{\mathbf{p}}_{i+1},$$

where $\widehat{\mathbf{p}}_{i+1} = -\widehat{\mathbf{z}}_{i+1} + \beta_{i+1}\widehat{\mathbf{p}}_i$. With these transformations, the matrix–vector product in step 8 of Algorithm 3 reads

$$\begin{aligned}\mathbf{q}_i &= (\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{H} + \mathbf{B}^{-1})\mathbf{B}\mathbf{H}^{\mathrm{T}}\widehat{\mathbf{p}}_i\\ &= \mathbf{H}^{\mathrm{T}}(\mathbf{R}^{-1}\mathbf{H}\mathbf{B}\mathbf{H}^{\mathrm{T}} + \mathbf{I}_m)\widehat{\mathbf{p}}_i,\end{aligned}$$

which is consistent with the equality

$$\widehat{\mathbf{q}}_i = (\mathbf{R}^{-1}\mathbf{H}\mathbf{B}\mathbf{H}^{\mathrm{T}} + \mathbf{I}_m)\widehat{\mathbf{p}}_i.$$

Consequently, we have just proved by induction that it is possible to define quantities $\widehat{\mathbf{r}}_i$, $\widehat{\mathbf{p}}_i$, $\widehat{\mathbf{v}}_i$, $\widehat{\mathbf{z}}_i$, and $\widehat{\mathbf{q}}_i$ such that relations in Equations (13)–(17) hold for any $i \geq 0$. Putting together these results gives our first modified version of the preconditioned conjugate gradient algorithm applied to Equation (5) and that we describe in Algorithm 4. Note that, as originally planned, all vectors used in the iterations now belong to $\Re^m$.

Algorithm 3. Preconditioned Conjugate Gradient on Equation (5)

```
1:  $\mathbf{v}_0$ = initial estimate
2:  $\mathbf{r}_0 = (\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{H} + \mathbf{B}^{-1})\mathbf{v}_0 - \mathbf{B}^{-1}(\mathbf{x}_{\mathrm{b}} - \mathbf{x}_0) - \mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{d}$
3:  $\mathbf{z}_0 = \mathbf{F}\mathbf{r}_0$
4:  $\mathbf{p}_0 = -\mathbf{z}_0$
5:  $i = 0$
6:  **while** stopping criterion not satisfied **do**
7:  $i = i + 1$
8:  $\mathbf{q}_{i-1} = (\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{H} + \mathbf{B}^{-1})\mathbf{p}_{i-1}$
9:  $\alpha_{i-1} = \mathbf{r}_{i-1}^{\mathrm{T}}\mathbf{z}_{i-1}/\mathbf{q}_{i-1}^{\mathrm{T}}\mathbf{p}_{i-1}$
10: $\mathbf{v}_i = \mathbf{v}_{i-1} + \alpha_{i-1}\mathbf{p}_{i-1}$
11: $\mathbf{r}_i = \mathbf{r}_{i-1} + \alpha_{i-1}\mathbf{q}_{i-1}$
12: $\mathbf{z}_i = \mathbf{F}\mathbf{r}_i$
13: $\beta_i = \mathbf{r}_i^{\mathrm{T}}\mathbf{z}_i/\mathbf{r}_{i-1}^{\mathrm{T}}\mathbf{z}_{i-1}$
14: $\mathbf{p}_i = -\mathbf{z}_i + \beta_i\mathbf{p}_{i-1}$
15: **endwhile**.
```

Each iteration of Algorithm 4 contains *five* matrix–vector products involving $\mathbf{H}\mathbf{B}\mathbf{H}^{\mathrm{T}}$. It turns out that, at the cost of introducing *two* new auxiliary vectors $\mathbf{w}$ and $\mathbf{t}$, the number of matrix–vector products with $\mathbf{HBH}^{\mathrm{T}}$ can be reduced down to *one* per iteration, as shown below. Consider $\mathbf{w}$ and $\mathbf{t}$ defined as

$$\mathbf{w}_i = \mathbf{HBH}^{\mathrm{T}}\widehat{\mathbf{z}}_i \quad \text{and} \quad \mathbf{t}_i = \mathbf{HBH}^{\mathrm{T}}\widehat{\mathbf{p}}_i.$$

From the definitions above, it follows that multiplying lines 4 and 14 of Algorithm 4 by $\mathbf{HBH}^{\mathrm{T}}$ gives

$$\mathbf{t}_i = \begin{cases} -\mathbf{w}_0 & \text{if } i = 0, \\ -\mathbf{w}_i + \beta_i \mathbf{t}_{i-1} & \text{if } i > 0, \end{cases}$$

for $i = 0, 1, \ldots$ Then we can incorporate these relations and after a slight algebraic arrangement, to transform Algorithm 4 into Algorithm 5.

We summarize here the main features of Algorithm 5:

- It is derived under Assumption 1 on the preconditioner and Assumption 2 on the initial guess of the iterations.
- It is mathematically equivalent to Algorithm 3 in the sense that, in exact arithmetic, both algorithms generate exactly the same iterates.
- It contains a single occurrence of the matrix–vector products by $\mathbf{B}$, $\mathbf{H}$, $\mathbf{H}^{\mathrm{T}}$ and $\mathbf{R}^{-1}$ per iteration. Note that, contrary to some implementations where a change of variable is done using $\mathbf{B}^{1/2}$ (e.g. Weaver *et al.*, 2003), no matrix square root is needed by the algorithm. Indeed, when the background matrix is used as a preconditioner, i.e. $\mathbf{F} = \mathbf{B}$, we simply have $\mathbf{G} = \mathbf{I}_m$ in Algorithm 5.
- In the case where each operator $\mathbf{H}$, $\mathbf{B}$, $\mathbf{H}^{\mathrm{T}}$ and $\mathbf{R}^{-1}$ is implemented by a separate routine in the chosen language, implementing RPCG results only in calling those operators in a different order. More work may be needed in other situations.

Algorithm 4. Preliminary modified PCG on Equation (5)

```
1:   $\widehat{\mathbf{v}}_0 = 0$
2:   $\widehat{\mathbf{r}}_0 = \mathbf{R}^{-1}\mathbf{H}\mathbf{v}_0 - \mathbf{R}^{-1}\mathbf{d}$
3:   $\widehat{\mathbf{z}}_0 = \mathbf{G}\widehat{\mathbf{r}}_0$
4:   $\widehat{\mathbf{p}}_0 = -\widehat{\mathbf{z}}_0$
5:   $i = 0$
6:   while stopping criterion not satisfied do
7:       $i = i + 1$
8:       $\widehat{\mathbf{q}}_{i-1} = (\mathbf{R}^{-1}\mathbf{HBH}^{\mathrm{T}} + \mathbf{I}_m)\widehat{\mathbf{p}}_{i-1}$
9:       $\alpha_{i-1} = \widehat{\mathbf{r}}_{i-1}^{\mathrm{T}}\mathbf{HBH}^{\mathrm{T}}\widehat{\mathbf{z}}_{i-1}/\widehat{\mathbf{q}}_{i-1}^{\mathrm{T}}\mathbf{HBH}^{\mathrm{T}}\widehat{\mathbf{p}}_{i-1}$
10:      $\widehat{\mathbf{v}}_i = \widehat{\mathbf{v}}_{i-1} + \alpha_{i-1}\widehat{\mathbf{p}}_{i-1}$
11:      $\widehat{\mathbf{r}}_i = \widehat{\mathbf{r}}_{i-1} + \alpha_{i-1}\widehat{\mathbf{q}}_{i-1}$
12:      $\widehat{\mathbf{z}}_i = \mathbf{G}\widehat{\mathbf{r}}_i$
13:      $\beta_i = \widehat{\mathbf{r}}_i^{\mathrm{T}}\mathbf{HBH}^{\mathrm{T}}\widehat{\mathbf{z}}_i/\widehat{\mathbf{r}}_{i-1}^{\mathrm{T}}\mathbf{HBH}^{\mathrm{T}}\widehat{\mathbf{z}}_{i-1}$
14:      $\widehat{\mathbf{p}}_i = -\widehat{\mathbf{z}}_i + \beta_i\widehat{\mathbf{p}}_{i-1}$
15:  endwhile.
16:  The solution is recovered from $\mathbf{v}_i = \mathbf{v}_0 + \mathbf{BH}^{\mathrm{T}}\widehat{\mathbf{v}}_i$
```

Algorithm 5. The Restricted Preconditioned Conjugate Gradient (RPCG) under Assumption 2

```
1:   $\widehat{\mathbf{v}}_0 = 0$
2:   $\widehat{\mathbf{r}}_0 = \mathbf{R}^{-1}\mathbf{H}\mathbf{v}_0 - \mathbf{R}^{-1}\mathbf{d}$
3    $\widehat{\mathbf{z}}_0 = \mathbf{G}\widehat{\mathbf{r}}_0$
4:   $\widehat{\mathbf{p}}_0 = -\widehat{\mathbf{z}}_0$
5:   $\mathbf{w}_0 = \mathbf{HBH}^{\mathrm{T}}\widehat{\mathbf{z}}_0$
6:   $\mathbf{t}_0 = -\mathbf{w}_0$
7:   $i = 0$
8:   while stopping criterion not satisfied do
9:       $i = i + 1$
10:      $\widehat{\mathbf{q}}_{i-1} = \mathbf{R}^{-1}\mathbf{t}_{i-1} + \widehat{\mathbf{p}}_{i-1}$
11       $\alpha_{i-1} = \mathbf{w}_{i-1}^{\mathrm{T}}\widehat{\mathbf{r}}_{i-1}/\widehat{\mathbf{q}}_{i-1}^{\mathrm{T}}\mathbf{t}_{i-1}$
12       $\widehat{\mathbf{v}}_i = \widehat{\mathbf{v}}_{i-1} + \alpha_{i-1}\widehat{\mathbf{p}}_{i-1}$
13:      $\widehat{\mathbf{r}}_i = \widehat{\mathbf{r}}_{i-1} + \alpha_{i-1}\widehat{\mathbf{q}}_{i-1}$
14:      $\widehat{\mathbf{z}}_i = \mathbf{G}\widehat{\mathbf{r}}_i$
15:      $\mathbf{w}_i = \mathbf{HBH}^{\mathrm{T}}\widehat{\mathbf{z}}_i$
16:      $\beta_i = \mathbf{w}_i^{\mathrm{T}}\widehat{\mathbf{r}}_i/\mathbf{w}_{i-1}^{\mathrm{T}}\widehat{\mathbf{r}}_{i-1}$
17:      $\widehat{\mathbf{p}}_i = -\widehat{\mathbf{z}}_i + \beta_i\widehat{\mathbf{p}}_{i-1}$
18:      $\mathbf{t}_i = -\mathbf{w}_i + \beta_i\mathbf{t}_{i-1}$
19:  endwhile.
20:  The solution is recovered from $\mathbf{v}_i = \mathbf{v}_0 + \mathbf{BH}^{\mathrm{T}}\widehat{\mathbf{v}}_i$
```

3.3. Recovering the increment, the gradient and the cost function for diagnostics

To recover vectors in the model space using information in Algorithm 5, one relies on the transformation in Equation (13). This should be the case for the *increment* and the *gradient* vectors, $\mathbf{v}_i$ and $\mathbf{r}_i$, respectively. The increment $\mathbf{v}_i$ is computed using

$$\mathbf{v}_i = \mathbf{v}_0 + \mathbf{BH}^{\mathrm{T}}\widehat{\mathbf{v}}_i,$$

while from Equation (4), the gradient $\nabla J[\mathbf{v}_i]$ is obtained by

$$\begin{aligned}\nabla J[\mathbf{v}_i] &= (\mathbf{B}^{-1} + \mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{H})\mathbf{v}_i - \mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{d} - \mathbf{B}^{-1}\mathbf{v}_0 \\ &= \mathbf{H}^{\mathrm{T}}\{\mathbf{Q}\widehat{\mathbf{v}}_i + \mathbf{R}^{-1}(\mathbf{H}\mathbf{v}_0 - \mathbf{d})\},\end{aligned}$$

with $\mathbf{Q} = (\mathbf{I}_m + \mathbf{R}^{-1}\mathbf{HBH}^{\mathrm{T}})$.

Denote the Hessian matrix $\mathbf{B}^{-1} + \mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{H}$ of the *quadratic* cost function J by $\nabla^2 J$. Let $\mathbf{P} \in \Re^{n\times q}$, $q < n$, be a given full-rank matrix and define $\mathbf{v} = \mathbf{v}_0 + \mathbf{Pz}$ for some vector $\mathbf{s} \in \Re^{q}$. Recall that $\mathbf{r}_0 = \nabla J[\mathbf{v}_0]$ and consider the Taylor-series expansion

$$\begin{aligned}J[\mathbf{v}] &= J[\mathbf{v}_0] + (\mathbf{v} - \mathbf{v}_0)^{\mathrm{T}}\nabla J[\mathbf{v}_0] \\ &\quad + \frac{1}{2}(\mathbf{v} - \mathbf{v}_0)^{\mathrm{T}}\nabla^2 J(\mathbf{v} - \mathbf{v}_0) \\ &= J[\mathbf{v}_0] + \mathbf{s}^{\mathrm{T}}\mathbf{P}^{\mathrm{T}}\mathbf{r}_0 + \frac{1}{2}\mathbf{s}^{\mathrm{T}}\mathbf{P}^{\mathrm{T}}\nabla^2 J\mathbf{P}\mathbf{s}, \qquad (18)\end{aligned}$$

of $J[\mathbf{v}]$ about $\mathbf{v}_0$. Consider now the *particular* case where the columns of $\mathbf{P}$ are formed with descent directions $\mathbf{p}_0, \ldots, \mathbf{p}_{i-1}$ from PCG (Algorithms 2 or 3). We know that, starting with $\mathbf{v}_0$ as the initial iterate, PCG will minimize $J[\mathbf{v}]$ over $\mathbf{v}_0+$ span$(\mathbf{p}_0, \ldots, \mathbf{p}_{i-1})$, and that the corresponding minimizer amounts to $\mathbf{v}_0 + [\mathbf{p}_0, \ldots, \mathbf{p}_{i-1}]\mathbf{s}^*$ with

$$\mathbf{s}^* = (\alpha_0, \ldots, \alpha_{i-1})^{\mathrm{T}},$$

where $\alpha_0, \ldots, \alpha_{i-1}$ are also from PCG. Since $\mathbf{P}^{\mathrm{T}}\nabla^2 J\mathbf{P}$ is symmetric and positive definite, $\mathbf{s}^*$ is also the solution of the equation

$$\mathbf{P}^{\mathrm{T}}\nabla^2 J\mathbf{P}\mathbf{s} = -\mathbf{P}^{\mathrm{T}}\mathbf{r}_0, \qquad (19)$$

obtained by nullifying the gradient of (18) with respect to $\mathbf{s}$. It follows that, putting $\mathbf{s}^*$ in (19) and then substituting the result in Equation (18), we find the corresponding minimum (also Tshimanga *et al.*, 2008, p 767),

$$J[\mathbf{v}_i] = J[\mathbf{v}_0] + \frac{1}{2}\mathbf{r}_0^{\mathrm{T}}\mathbf{P}\mathbf{s}^*.$$

Now, if we define $\widehat{\mathbf{P}} = [\widehat{\mathbf{p}}_0, \ldots, \widehat{\mathbf{p}}_{i-1}]$, using lines 5 and 15 of Algorithm 5 and Equations (10) and (13), we get

$$\begin{aligned} J[\mathbf{v}_i] &= J[\mathbf{v}_0] + \frac{1}{2}\widehat{\mathbf{r}}_0^{\mathrm{T}}\mathbf{H}\mathbf{B}\mathbf{H}^{\mathrm{T}}\widehat{\mathbf{P}}\mathbf{s}^* \\ &= J[\mathbf{v}_0] + \frac{1}{2}\sum_{j=0}^{i-1}\alpha_j\widehat{\mathbf{r}}_0^{\mathrm{T}}\mathbf{t}_j. \end{aligned}$$

Finally, observe that

$$J[\mathbf{v}_0] = \frac{1}{2}(\mathbf{H}\mathbf{v}_0 - \mathbf{d})^{\mathrm{T}}\mathbf{R}^{-1}(\mathbf{H}\mathbf{v}_0 - \mathbf{d}),$$

with $\mathbf{v}_0 = \mathbf{x}^{\mathrm{b}} - \mathbf{x}_0$ from Assumption 2.

3.4. Preconditioning issues

We restrict ourselves to show that the spectral preconditioner, which is used in Fisher (1998) as well as in Weaver *et al.* (2003), satisfies Assumption 1. It is the object of forthcoming work to generalize the approach to the Limited Memory Preconditioner (LMP) class introduced in Tshimanga *et al.* (2008). In data assimilation, the preconditioning is often performed in two steps. First, a preconditioning by the background matrix is implemented as a change of variable from $\mathbf{v}$ to $\bar{\mathbf{v}}$ via the square-root matrix $\mathbf{B}^{1/2}$, that is,

$$\mathbf{v} = \mathbf{B}^{1/2}\bar{\mathbf{v}}.$$

Thus, the matrix of the preconditioned system reads

$$\mathbf{B}^{1/2}(\mathbf{B}^{-1} + \mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{H})\mathbf{B}^{1/2} = \mathbf{I}_n + \mathbf{B}^{1/2}\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{H}\mathbf{B}^{1/2}. \qquad (20)$$

Second, suppose that $(\lambda_j, \mathbf{u}_j)$, $j = 1, \ldots, l$, are eigenpairs of the symmetric matrix in (20), i.e.

$$\mathbf{u}_j^{\mathrm{T}}\mathbf{u}_i = \delta_{ji} \text{ and } (\mathbf{I}_n + \mathbf{B}^{1/2}\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{H}\mathbf{B}^{1/2})\mathbf{u}_j = \lambda_j\mathbf{u}_j, \qquad (21)$$

or, equivalently

$$\mathbf{u}_j^{\mathrm{T}}\mathbf{u}_i = \delta_{ji} \text{ and } (\lambda_j - 1)\mathbf{u}_j = \mathbf{B}^{1/2}\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{H}\mathbf{B}^{1/2}\mathbf{u}_j, \qquad (22)$$

where δ_{ji} is the Kronecker symbol defined by $\delta_{ii} = 1$, $\delta_{ij} = 0$ for $i \neq j$. Then, the so-called spectral LMP is defined by

$$\mathbf{K} = \mathbf{I}_n + \sum_{j=1}^{l}\left(\frac{1}{\lambda_j} - 1\right)\mathbf{u}_j\mathbf{u}_j^{\mathrm{T}}.$$

It is well known that performing a preconditioned CG is equivalent to implementing a change of variable (Demmel, 1997, pp 317–318; Nocedal and Wright, 1999, p 118). It follows that the whole preconditioning procedure, consisting in a preconditioning of $\mathbf{B}^{-1} + \mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{H}$ by the background matrix $\mathbf{B}$ followed by the spectral preconditioning with $\mathbf{K}$, is equivalent to using PCG on the system matrix $\mathbf{B}^{-1} + \mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{H}$ preconditioned with the product $\mathbf{F} = \mathbf{B}^{1/2}\mathbf{K}\mathbf{B}^{1/2}$. From

$$\begin{aligned} \mathbf{F} &= \mathbf{B}^{1/2}\mathbf{K}\mathbf{B}^{1/2} \\ &= \mathbf{B} + \sum_{j=1}^{l}\left(\frac{1}{\lambda_j} - 1\right)\mathbf{B}^{1/2}\mathbf{u}_j\mathbf{u}_j^{\mathrm{T}}\mathbf{B}^{1/2} \\ &= \mathbf{B} - \sum_{j=1}^{l}\frac{1}{\lambda_j}\mathbf{B}^{1/2}(\lambda_j - 1)\mathbf{u}_j\mathbf{u}_j^{\mathrm{T}}\mathbf{B}^{1/2}, \end{aligned}$$

we obtain, using (22), that

$$\begin{aligned} \mathbf{F} &= \mathbf{B} - \sum_{j=1}^{l}\frac{1}{\lambda_j}\mathbf{B}\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{H}\mathbf{B}^{1/2}\mathbf{u}_j\mathbf{u}_j^{\mathrm{T}}\mathbf{B}^{1/2} \\ &= \mathbf{B} - \mathbf{B}\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\sum_{j=1}^{l}\frac{1}{\lambda_j}\mathbf{H}\mathbf{B}^{1/2}\mathbf{u}_j\mathbf{u}_j^{\mathrm{T}}\mathbf{B}^{1/2}. \end{aligned}$$

Therefore, we get

$$\mathbf{F}\mathbf{H}^{\mathrm{T}} = \mathbf{B}\mathbf{H}^{\mathrm{T}}\left(\mathbf{I}_m - \mathbf{R}^{-1}\sum_{j=1}^{l}\frac{1}{\lambda_j}\mathbf{H}\mathbf{B}^{1/2}\mathbf{u}_j\mathbf{u}_j^{\mathrm{T}}\mathbf{B}^{1/2}\mathbf{H}^{\mathrm{T}}\right),$$

which corresponds to Assumption 1 with

$$\begin{aligned} \mathbf{G} &= \mathbf{I}_m - \mathbf{R}^{-1}\sum_{j=1}^{l}\frac{1}{\lambda_j}\mathbf{H}\mathbf{B}^{1/2}\mathbf{u}_j\mathbf{u}_j^{\mathrm{T}}\mathbf{B}^{1/2}\mathbf{H}^{\mathrm{T}} \\ &= \mathbf{I}_m - \mathbf{R}^{-1}\sum_{j=1}^{l}\frac{1}{\lambda_j}\underline{\mathbf{u}}_j\underline{\mathbf{u}}_j^{\mathrm{T}}, \end{aligned} \qquad (23)$$

where $\underline{\mathbf{u}}_j = \mathbf{H}\mathbf{B}^{1/2}\mathbf{u}_j \in \Re^m$. Thus, it follows that the spectral preconditioner satisfies Assumption 1.

Multiplying both sides of the second equality of Equation (21) by $\mathbf{B}^{1/2}$ yields

$$(\mathbf{I}_n + \mathbf{B}\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{H})\mathbf{B}^{1/2}\mathbf{u}_j = \lambda_j\mathbf{B}^{1/2}\mathbf{u}_j,$$

which, letting $\bar{\mathbf{u}}_j = \mathbf{B}^{1/2}\mathbf{u}_j$, is equivalent to

$$(\mathbf{I}_n + \mathbf{B}\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{H})\bar{\mathbf{u}}_j = \lambda_j \bar{\mathbf{u}}_j. \tag{24}$$

Relation (24) tells us that $(\lambda_j, \bar{\mathbf{u}}_j)$, $j = 1, \ldots, l$ are *right eigenpairs* of the nonsymmetric matrix $\mathbf{BA} = \mathbf{I}_n + \mathbf{B}\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{H}$, where $\mathbf{A}$ stands for $\mathbf{B}^{-1} + \mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{H}$. Thus, one can use right eigenpairs of $\mathbf{BA}$, if available, together with $\mathbf{R}^{-1}$ to built $\mathbf{G}$ following Equation (23).

3.5. Loss (and recovery) of orthogonality

We recalled in the introduction that current conjugate gradient-based quadratic solvers used in data assimilation use re-orthogonalization techniques, to alleviate the convergence delay due to loss of orthogonality that may arise when round-off errors occur. For Algorithm 3, these techniques require the storage of vectors of $\Re^n$ against which orthogonality is restored, which can be a serious problem when large problems are considered.

This section explores how the modifications of PCG we propose can be used to implement an orthogonalization scheme involving only vectors belonging to $\Re^m$. In PCG, the residuals are mutually orthogonal with respect to the inner product induced by the preconditioner, which can be expressed as $\mathbf{r}_i^{\mathrm{T}}\mathbf{F}\mathbf{r}_j = 0$ if $i \neq j$. In the presence of round-off errors, a (modified) Gram–Schmidt re-orthogonalization scheme is written as

$$\mathbf{r}_i \longleftarrow \prod_{j=1}^{i-1}\left(\mathbf{I}_n - \frac{\mathbf{r}_j\mathbf{r}_j^{\mathrm{T}}\mathbf{F}}{\mathbf{r}_j^{\mathrm{T}}\mathbf{F}\mathbf{r}_j}\right)\mathbf{r}_i.$$

(See Golub and Van Loan (1996, p 231) and Higham (2002, p 370) for the description of the algorithm that can be used to restore orthogonality.) Observe that vectors $\mathbf{F}\mathbf{r}_j = \mathbf{z}_j$ and scalars $\mathbf{r}_j^{\mathrm{T}}\mathbf{F}\mathbf{r}_j = \mathbf{z}_j^{\mathrm{T}}\mathbf{r}_j$ are by-products of Algorithm 3. This leads to

$$\mathbf{r}_i \longleftarrow \prod_{j=1}^{i-1}\left(\mathbf{I}_n - \frac{\mathbf{r}_j\mathbf{z}_j^{\mathrm{T}}}{\mathbf{r}_j^{\mathrm{T}}\mathbf{z}_j}\right)\mathbf{r}_i.$$

Now, using $\mathbf{r}_i = \mathbf{H}^{\mathrm{T}}\hat{\mathbf{r}}_i$, $\mathbf{z}_i = \mathbf{B}\mathbf{H}^{\mathrm{T}}\hat{\mathbf{z}}_i$ and $\mathbf{w}_i = \mathbf{H}\mathbf{B}\mathbf{H}^{\mathrm{T}}\mathbf{z}_i$ (see expression in line 5 of Algorithm 5), we obtain that for $i \neq j$, the relation $\mathbf{r}_j^{\mathrm{T}}\mathbf{F}\mathbf{r}_i = \mathbf{z}_j^{\mathrm{T}}\mathbf{r}_i = 0$ implies that $\hat{r}_j^{\mathrm{T}}\mathbf{H}\mathbf{B}\mathbf{H}^{\mathrm{T}}\hat{\mathbf{r}}_i = \hat{\mathbf{r}}_j^{\mathrm{T}}\mathbf{w}_i = 0$. It follows that the vectors $\hat{\mathbf{r}}_j$ are orthogonal with respect to the inner product induced by $\mathbf{H}\mathbf{B}\mathbf{H}^{\mathrm{T}}$. Consequently, we suggest the following re-orthogonalization scheme

$$\hat{\mathbf{r}}_i \longleftarrow \prod_{j=1}^{i-1}\left(\mathbf{I}_m - \frac{\hat{\mathbf{r}}_j\mathbf{w}_j^{\mathrm{T}}}{\hat{\mathbf{r}}_j^{\mathrm{T}}\mathbf{w}_j}\right)\hat{\mathbf{r}}_i. \tag{25}$$

Note that the total number of pairs to be stored can be reduced if selective re-orthogonalization is performed instead of the complete one. We refer the reader to Golub and Van Loan (1996, p 483) for a discussion about the concept of selective (re-)orthogonalization.

3.6. Numerical experiments

We show on a particular example the performance of the re-orthogonalization scheme (25). The test consists in solving a linear system of form given in Equation (5). We choose $n = 200$ and $m = 40$. Then we randomly generate symmetric positive definite matrices $\mathbf{B} \in \Re^{n\times n}$, $\mathbf{R} \in \Re^{m\times m}$, and $\mathbf{H} \in \Re^{m\times n}$ and vectors $\mathbf{x}^{\mathrm{b}}$ and $\mathbf{x}_0$ both in $\Re^n$. For simplicity, We use matrix $\mathbf{F} = \mathbf{B}$ as preconditioner. Therefore the corresponding matrix s$\mathbf{G}$ is the identity matrix $\mathbf{I}_m$. We compare the performance of Algorithm 3 and Algorithm 5 with and without re-orthogonalization. We base our comparison on the values of the quadratic cost function associated with the system (5).

The following observations can be made on Figure 1. Without re-orthogonalization, Algorithm 3 and Algorithm 5 give similar results that cannot be distinguished on the plot. The orthogonalization process improves both algorithms equally since the symbols $\circ$ and $+$ coincide. Therefore, we see that the re-orthogonalization scheme of Algorithm 5 is interesting because it is cheaper in memory and computational costs than that of Algorithm 3, and gives the same efficiency.

3.7. Dealing with arbitrary initial guess in the inner loop

A conjugate gradient algorithm working with vectors in $\Re^m$ has been derived in Section 3.2 under Assumption 2. This assumption takes an initial guess that corresponds to a zero background term. Since in the conjugate gradient method, it is always desirable to start the iterations with a guess that is the closest possible to the solution, Assumption 2 may be considered as restrictive, in situations where a *better estimate* of the solution is available. We therefore consider, in the present section, the case where the conjugate gradient iterations are

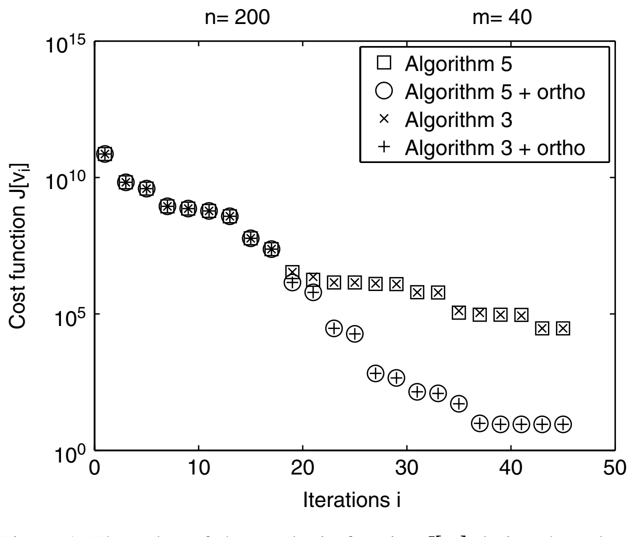

Figure 1. The value of the quadratic function $J[\mathbf{v}_i]$ during the solver iterations i. Here, $m = 40$ and $n = 200$. Results are displayed for algorithm 3 with ($\circ$) and without ($\square$) orthogonalization and for algorithm 5 with ($+$) and without ($\times$) re-orthogonalization. Note that the values of the cost function at the last iteration are 9 and 3×10^4, respectively.

initiated with a vector $\mathbf{v}_0$ that does not necessarily satisfy Assumption 2.

The derivation of section 3.2 does not hold any more, because it is not possible to factor by $\mathbf{H}^{\mathrm{T}}$ the expression of the initial residual, since we now have

$$\mathbf{r}_0 = (\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{H} + \mathbf{B}^{-1})\mathbf{v}_0 - \mathbf{B}^{-1}(\mathbf{x}^{\mathrm{b}} - \mathbf{x}_0) - \mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{d}.$$

However, if we now introduce the (augmented) matrices $\widetilde{\mathbf{H}} \in \Re^{(m+1)\times n}$, $\widetilde{\mathbf{R}}^{\mathrm{I}} \in \Re^{(m+1)\times(m+1)}$, and vector $\mathbf{d}_{\mathbf{R}^{\mathrm{I}}} \in \Re^{m+1}$, defined by

$$\widetilde{\mathbf{H}} = \begin{bmatrix} \mathbf{H} \\ (\mathbf{x}^b - \mathbf{x}_0 - \mathbf{v}_0)^T\mathbf{B}^{-1} \end{bmatrix},$$
$$\widetilde{\mathbf{R}}^{\mathrm{I}} = \begin{bmatrix} \mathbf{R}^{-1} & \\ & 0 \end{bmatrix},$$
$$\mathbf{d}_{\mathbf{R}^{\mathrm{I}}} = \begin{bmatrix} \mathbf{R}^{-1}\mathbf{d} \\ 1 \end{bmatrix},$$

the initial residual can now be factored as

$$\mathbf{r}_0 = \widetilde{\mathbf{H}}^{\mathrm{T}}\widetilde{\mathbf{R}}^{\mathrm{I}}\widetilde{\mathbf{H}}\mathbf{v}_0 - \widetilde{\mathbf{H}}^{\mathrm{T}}\mathbf{d}_{\mathbf{R}^{\mathrm{I}}} = \widetilde{\mathbf{H}}^{\mathrm{T}}(\widetilde{\mathbf{R}}^{\mathrm{I}}\widetilde{\mathbf{H}}\mathbf{v}_0 - \mathbf{d}_{\mathbf{R}^{\mathrm{I}}}),$$

and using the relation

$$(\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{H} + \mathbf{B}^{-1}) = (\widetilde{\mathbf{H}}^{\mathrm{T}}\widetilde{\mathbf{R}}^{\mathrm{I}}\widetilde{\mathbf{H}} + \mathbf{B}^{-1}),$$

we see that Algorithm 3 becomes Algorithm 6.

Since $\widetilde{\mathbf{H}}^{\mathrm{T}}$ can now be factored in $\mathbf{r}_0$, we can use directly the result of section 3.2 and obtain Algorithm 7 that is, once more, mathematically equivalent to Algorithm 6, under the assumption

$$\textbf{Assumption } \widetilde{1} \; : \; \mathbf{B}\widetilde{\mathbf{H}}^{\mathrm{T}}\mathbf{G} = \mathbf{F}\widetilde{\mathbf{H}}^{\mathrm{T}}. \qquad (26)$$

Algorithm 6. PCG with arbitrary starting point

```
1:  $\mathbf{v}_0$ = initial estimate
2:  $\mathbf{r}_0 = \widetilde{\mathbf{H}}^{\mathrm{T}}(\widetilde{\mathbf{R}}^{\mathrm{I}}\widetilde{\mathbf{H}}\mathbf{v}_0 - \mathbf{d}_{\mathbf{R}^{\mathrm{I}}})$
3:  $\mathbf{z}_0 = \mathbf{F}\mathbf{r}_0$
4:  $\mathbf{p}_0 = -\mathbf{z}_0$
5:  $i = 0$
6:  while stopping criterion not satisfied do
7:      $i = i + 1$
8.      $\mathbf{q}_{i-1} = (\widetilde{\mathbf{H}}^{\mathrm{T}}\widetilde{\mathbf{R}}^{\mathrm{I}}\widetilde{\mathbf{H}} + \mathbf{B}^{-1})\mathbf{p}_{i-1}$
9:      $\alpha_{i-1} = \mathbf{r}_{i-1}^{\mathrm{T}}\mathbf{z}_{i-1}/\mathbf{q}_{i-1}^{\mathrm{T}}\mathbf{p}_{i-1}$
10:     $\mathbf{v}_i = \mathbf{v}_{i-1} + \alpha_{i-1}\mathbf{p}_{i-1}$
11:     $\mathbf{r}_i = \mathbf{r}_{i-1} + \alpha_{i-1}\mathbf{q}_{i-1}$
12:     $\mathbf{z}_i = \mathbf{F}\mathbf{r}_i$
13:     $\beta_i = \mathbf{r}_i^{\mathrm{T}}\mathbf{z}_i/\mathbf{r}_{i-1}^{\mathrm{T}}\mathbf{z}_{i-1}$
14:     $\mathbf{p}_i = -\mathbf{z}_i + \beta_i\mathbf{p}_{i-1}$
15: endwhile.
```

Algorithm 7. RPCG with arbitrary starting point and augmented matrices

```
1:  $\mathbf{v}_0$ = initial estimate
2:  $\widehat{\mathbf{v}}_0 = 0$
3:  $\widehat{\mathbf{r}}_0 = \widetilde{\mathbf{R}}^{\mathrm{I}}\widetilde{\mathbf{H}}\mathbf{v}_0 - \mathbf{d}_{\mathbf{R}^{\mathrm{I}}}$
4:  $\widehat{\mathbf{z}}_0 = \mathbf{G}\widehat{\mathbf{r}}_0$
5:  $\widehat{\mathbf{p}}_0 = -\widehat{\mathbf{z}}_0$
6:  $\mathbf{w}_0 = \widetilde{\mathbf{H}}\mathbf{B}\widetilde{\mathbf{H}}^{\mathrm{T}}\widehat{\mathbf{r}}_0$
7:  $\mathbf{t}_0 = -\mathbf{w}_0$
8:  $i = 0$
9:  while stopping criterion not satisfied do
10:     $i = i + 1$
11:     $\widehat{\mathbf{q}}_{i-1} = \widetilde{\mathbf{R}}^{\mathrm{I}}\mathbf{t}_{i-1} + \widehat{\mathbf{p}}_{i-1}$
12:     $\alpha_{i-1} = \mathbf{w}_{i-1}^{\mathrm{T}}\widehat{\mathbf{z}}_{i-1}/\widehat{\mathbf{q}}_{i-1}^{\mathrm{T}}\mathbf{t}_{i-1}$
13:     $\widehat{\mathbf{v}}_i = \widehat{\mathbf{v}}_{i-1} + \alpha_{i-1}\widehat{\mathbf{p}}_{i-1}$
14:     $\widehat{\mathbf{r}}_i = \widehat{\mathbf{r}}_{i-1} + \alpha_{i-1}\widehat{\mathbf{q}}_{i-1}$
15:     $\widehat{\mathbf{z}}_i = \mathbf{G}\widehat{\mathbf{r}}_i$
16:     $\mathbf{w}_i = \widetilde{\mathbf{H}}\mathbf{B}\widetilde{\mathbf{H}}^{\mathrm{T}}\widehat{\mathbf{r}}_i$
17:     $\beta_i = \mathbf{w}_i^{\mathrm{T}}\widehat{\mathbf{z}}_i/\mathbf{w}_{i-1}^{\mathrm{T}}\widehat{\mathbf{z}}_{i-1}$
18:     $\widehat{\mathbf{p}}_i = -\widehat{\mathbf{z}}_i + \beta_i\widehat{\mathbf{p}}_{i-1}$
19:     $\mathbf{t}_i = -\mathbf{w}_i + \beta_i\mathbf{t}_{i-1}$
20: endwhile.
21: The solution is recovered from $\mathbf{v}_i = \mathbf{v}_0 + \mathbf{B}\widetilde{\mathbf{H}}^{\mathrm{T}}\widehat{\mathbf{v}}_i$
```

Note that the vectors used in Algorithm 7 are not in $\Re^m$ but in $\Re^{m+1}$, because we had to consider the augmented matrices $\widetilde{\mathbf{R}}^{\mathrm{I}}$ and $\widetilde{\mathbf{H}}$ to be able to perform the suitable transformations on Algorithm 6. Note also that Assumption $\widetilde{1}$ is again satisfied when $\mathbf{F} = \mathbf{B}$ and $\mathbf{G} = \mathbf{I}_m$.

To avoid explicit reference to the augmented operators in the algorithm, we introduce

$$\mathbf{c} = \mathbf{B}^{-1}(\mathbf{x}^{\mathrm{b}} - \mathbf{x}_0 - \mathbf{v}_0),$$
$$\mathbf{s} = \mathbf{H}\mathbf{B}\mathbf{c} = \mathbf{H}(\mathbf{x}^{\mathrm{b}} - \mathbf{x}_0 - \mathbf{v}_0)$$
$$\text{and } \sigma = \mathbf{c}^{\mathrm{T}}\mathbf{B}\mathbf{c} = (\mathbf{x}^{\mathrm{b}} - \mathbf{x}_0 - \mathbf{v}_0)^{\mathrm{T}}\mathbf{B}^{-1}(\mathbf{x}^{\mathrm{b}} - \mathbf{x}_0 - \mathbf{v}_0),$$

and the following block equalities hold:

$$\widetilde{\mathbf{H}}\mathbf{B}\widetilde{\mathbf{H}}^{\mathrm{T}} = \begin{bmatrix} \mathbf{H} \\ \mathbf{c}^{\mathrm{T}} \end{bmatrix}\mathbf{B}[\mathbf{H}^{\mathrm{T}}, \mathbf{c}] = \begin{bmatrix} \mathbf{H}\mathbf{B}\mathbf{H}^{\mathrm{T}} & \mathbf{s} \\ \mathbf{s}^{\mathrm{T}} & \sigma \end{bmatrix}. \qquad (27)$$

In addition, for any vector (here we chose $\mathbf{r}$ as an example), we introduce the notation

$$\widehat{\mathbf{r}}_i = \begin{bmatrix} \widehat{\mathbf{r}}_i(1:m) \\ \widehat{\mathbf{r}}_i(m+1) \end{bmatrix},$$

where the $(m+1)$-dimensional vector $\widehat{\mathbf{r}}_i$ is decomposed into a m-dimensional vector $\widehat{\mathbf{r}}_i(1:m)$ augmented with the scalar $\widehat{\mathbf{r}}_i(m+1)$. Using this notation, all the operations involving the augmented matrices in lines 3, 6, 11, 16 and 21 of Algorithm 7 are decomposed in Algorithm 8 using (27).

Let us make some comments on the implementation. The Algorithm 8 operates on nine $(m+1)$-dimensional vectors namely, $\mathbf{s}_i, \widehat{\mathbf{v}}_i, \mathbf{d}_i, \widehat{\mathbf{r}}_i, \widehat{\mathbf{p}}_i, \widehat{\mathbf{z}}_i, \mathbf{w}_i, \mathbf{t}_i$, and $\widehat{\mathbf{q}}_i$, together with three linear operators from $\Re^m$ to $\Re^m$, respectively,

$\mathbf{R}^{-1}$, $\mathbf{G}$ and $\mathbf{HBH}^{\mathrm{T}}$. In terms of scalars, apart from α_i and β_i the algorithm needs σ and two other auxiliary scalars to be defined in order to handle $\mathbf{w}_y^{\mathrm{T}}\widehat{\mathbf{z}}_i$ and $\widehat{\mathbf{q}}_i^{\mathrm{T}}\mathbf{t}_i$.

Algorithm 8. RPCG

```
1:  $\mathbf{v}_0$ = initial estimate
2:  Compute $\mathbf{s} = \mathbf{H}(\mathbf{x}^{\mathrm{b}} - \mathbf{x}_0 - \mathbf{v}_0)$
3:  Compute $\sigma = (\mathbf{x}^{\mathrm{b}} - \mathbf{x}_0 - \mathbf{v}_0)^{\mathrm{T}}\mathbf{B}^{-1}(\mathbf{x}^{\mathrm{b}} - \mathbf{x}_0 - \mathbf{v}_0)$
4:  $\widehat{\mathbf{v}}_0 = \mathbf{0}$
5:  $\widehat{\mathbf{r}}_0(1:m) = \mathbf{R}^{-1}\mathbf{H}\mathbf{v}_0 - \mathbf{R}^{-1}\mathbf{d}$
6:  $\widehat{\mathbf{r}}_0(m+1) = -1$
7:  $\widehat{\mathbf{z}}_0 = \mathbf{G}\widehat{\mathbf{r}}_0$
8:  $\widehat{\mathbf{p}}_0 = -\widehat{\mathbf{z}}_0$
9:  $\mathbf{w}_0(1:m) = \mathbf{HBH}^{\mathrm{T}}\widehat{\mathbf{r}}_0(1:m) + \widehat{\mathbf{r}}_0(m+1)\mathbf{s}$
10: $\mathbf{w}_0(m+1) = \mathbf{s}^{\mathrm{T}}\widehat{\mathbf{r}}_0(1:m) + \sigma\widehat{\mathbf{r}}_0(m+1)$
11: $\mathbf{t}_0 = -\mathbf{w}_0$
12: $i = 0$
13: while stopping criterion not satisfied do
14:     $i = i + 1$
15:     $\widehat{\mathbf{q}}_{i-1}(1:m) = \mathbf{R}^{-1}\mathbf{t}_{i-1}(1:m) + \widehat{\mathbf{p}}_{i-1}(1:m)$
16:     $\widehat{\mathbf{q}}_{i-1}(m+1) = \widehat{\mathbf{p}}_{i-1}(m+1)$
17:     $\alpha_{i-1} = \mathbf{w}_{i-1}^{\mathrm{T}}\widehat{\mathbf{z}}_{i-1}/\widehat{\mathbf{q}}_{i-1}^{\mathrm{T}}\mathbf{t}_{i-1}$
18:     $\widehat{\mathbf{v}}_i = \widehat{\mathbf{v}}_{i-1} + \alpha_{i-1}\widehat{\mathbf{p}}_{i-1}$
19:     $\widehat{\mathbf{r}}_i = \widehat{\mathbf{r}}_{i-1} + \alpha_{i-1}\widehat{\mathbf{q}}_{i-1}$
20:     $\widehat{\mathbf{z}}_i = \mathbf{G}\widehat{\mathbf{r}}_i$
21:     $\mathbf{w}_i(1:m) = \mathbf{HBH}^{\mathrm{T}}\widehat{\mathbf{r}}_i(1:m) + \widehat{\mathbf{r}}_i(m+1)\mathbf{s}$
22:     $\mathbf{w}_i(m+1) = \mathbf{s}^{\mathrm{T}}\widehat{\mathbf{r}}_i(1:m) + \sigma\widehat{\mathbf{r}}_i(m+1)$
23:     $\beta_i = \mathbf{w}_i^{\mathrm{T}}\widehat{\mathbf{z}}_i/\mathbf{w}_{i-1}^{\mathrm{T}}\widehat{\mathbf{z}}_{i-1}$
24:     $\widehat{\mathbf{p}}_i = -\widehat{\mathbf{z}}_i + \beta_i\widehat{\mathbf{p}}_{i-1}$
25:     $\mathbf{t}_i = -\mathbf{w}_i + \beta_i\mathbf{t}_{i-1}$
26: endwhile.
27: The solution is recovered from $\mathbf{v}_i = \mathbf{v}_0 + \mathbf{BH}^{\mathrm{T}}\widehat{\mathbf{v}}_i(1:m) + (\mathbf{x}^{\mathrm{b}} - \mathbf{x}_0 - \mathbf{v}_0)\widehat{\mathbf{v}}_i(m+1)$
```

As shown in line 27 of Algorithm 8, the increment at each iteration i is:

$$\mathbf{v}_i = \mathbf{v}_0 + \mathbf{BH}^{\mathrm{T}}\widehat{\mathbf{v}}_i(1:m) + (\mathbf{x}^{\mathrm{b}} - \mathbf{x}_0 - \mathbf{v}_0)\widehat{\mathbf{v}}_i(m+1).$$

According to the analogy with discussions and formulae in section 3.3, the corresponding cost function and gradient are written in a similar manner.

4. Numerical experiments

The aim of this section is to provide a comparison of the behaviour of RPCG and PSAS algorithms on an idealized and academic test-case. Experimental data are produced using twin experiments, where the data are perturbed according to the given covariance diagonal matrices **B** and **R**. Implementing RPCG in a realistic system, involving real-life models, observations, and correlated errors, will be considered in a future work.

4.1. The test problem

The problem consists in finding the 'best' initial temperature distribution (on a grid over an unit square $\Omega = (0,1)^2$) using an approximate initial temperature distribution as well as a given set of approximate temperature distributions at different times. The evolution model to be considered is the numerical integration of a nonlinear heat equation of the form

$$\begin{aligned}\frac{\partial x}{\partial t} - \frac{\partial^2 x}{\partial u^2} - \frac{\partial^2 x}{\partial v^2} + f[x] &= 0 \text{ in } \Omega \times (0,\infty),\\ x[u,v,t] &= 0 \text{ on } \partial\Omega \times (0,\infty),\end{aligned}$$

where the (continuous) temperature variable $x[u,v,t]$ depends on both time $t > 0$ and the position given by spatial coordinates u and v. The function $f[x]$ is defined by

$$f[x] = \exp[\eta x],$$

with $\eta = 4.2$. The nonlinear and linear *model* operators required in our incremental formulation (2) are produced as follows. We use an uniform 5-point finite-difference scheme with $n = 32 \times 32 = 1024$ nodes and, consequently, a space step length $h = 1/33$. The Euler scheme is employed for the time integration with a time step length $\tau = 2 \times 10^{-4}$. Let us consider vector $\mathbf{x}(t_j) \in \Re^n$, with $t_j = j\tau$ and $j = 0, 1, \ldots$, the *discretized* temperature field. The *nonlinear model*

$$\mathcal{M}_{j,0} : \Re^n \longrightarrow \Re^n : \mathcal{M}_{j,0}[\mathbf{x}(t_0)] = \mathbf{x}(t_j)$$

is defined by the recurrence below. Starting from the state vector $\mathbf{x}(t_0)$, perform

$$\mathbf{x}(t_{j+1}) = \left(\mathbf{I}_n - \frac{\tau}{h^2}\mathbf{Q}\right)^{-1} \{\mathbf{x}(t_j) - \tau\mathbf{f}^{(j)}\}.$$

Here, $\mathbf{I}_n$ is the order n identity matrix and $\mathbf{Q}$ corresponds to 5-point discrete negative *scaled* Laplacian matrix, a $(s \times s)$ by $(s \times s)$ block tridiagonal

$$\mathbf{Q} = \mathrm{tridiag}(-\mathbf{I}_s, \mathbf{T}_s, -\mathbf{I}_s),$$

where $\mathbf{I}_s$ is the identity matrix and $\mathbf{T}_s = \mathrm{tridiag}(-1, 4, -1)$, both of order s, with $s^2 = n$. Note that the vector $\mathbf{f} \in \Re^n$ results from the discretization of the function $f[u]$. The *linear model* operator $\mathbf{M}_{j,0} \in \Re^{n\times n} : \delta\mathbf{x}(t_j) = \mathbf{M}_{j,0}\delta\mathbf{x}(t_0)$ is given by the corresponding recurrence. Starting from the increment $\delta\mathbf{x}(t_0)$, perform

$$\delta\mathbf{x}(t_{j+1}) = \left(\mathbf{I}_n - \frac{\tau}{h^2}\mathbf{Q}\right)^{-1} (\mathbf{I}_n - \tau\mathbf{F}^{(j)})\delta\mathbf{x}(t_j).$$

The matrix $\mathbf{F} = \mathrm{diag}(\mathbf{f}'_u)$ is the Jacobian of the discretized function $\mathbf{f}$.

Concerning the generation of the *background* and *observation* vectors, we begin by constructing the 'true' initial discrete temperature distribution over the square grid as

$$x_0^{\mathrm{true}}(u_q, v_r) = \gamma u_q(1-u_q)v_r(1-v_r),$$

that we reshape into the vector $\mathbf{x}_0^{\mathrm{true}} \in \Re^n$ with $\gamma = 25$. Then we use random vectors with normal distribution to create pertubations as follows.

The vector $\mathbf{x}_0^{\text{true}}$ is slightly perturbed leading to the *background vector*

$$\mathbf{x}^{\text{b}} = \mathbf{x}_0^{\text{true}} + \mathbf{e}^{\text{b}},$$

where the n-vector $\mathbf{e}^{\text{b}} \sim \mathcal{N}_n(\mathbf{0}, \beta^2\mathbf{I}_n)$, with $\beta = 10^{-1}$, represents the (artificial) noise field. Here and in the sequel, $\mathbf{g} \sim \mathcal{N}_p(\mathbf{m}, \Sigma)$ stands for a normally distributed random vector from $\Re^p$ with the mean equal to $\mathbf{m} \in \Re^p$, and the covariance matrix equal to $\Sigma \in \Re^{p\times p}$.

Before we produce the observations, we construct the operator to map the model field to the observation space. Here, we choose a restriction operator. So, defining $\mathbf{e}_\ell$ as the ℓth column of the identity matrix of order n, the *observation operators* is built as the $m_j \times n$ matrix

$$\mathcal{H}_j = \mathbf{H}_j = \mathbf{C}[\mathbf{e}_1, \mathbf{e}_{17}, \mathbf{e}_{33}, \ldots, \mathbf{e}_{1009}]^{\text{T}},$$

with $m_j = 64$ and where $\mathbf{C}$ is the diagonal matrix of the eigenvalues of the 5-point discrete Laplacian of order m_j. This means that the observations are obtained by selecting 1 point every 16 points on the discretization grid and apply the operator $\mathbf{C}$. We (artificially) design the observations vectors $\mathbf{y}_j^{\text{o}} \in \Re^{m_j}$ by

$$\mathbf{y}_j^{\text{o}} = \mathbf{H}_j\mathcal{M}[t_o, t_j, \mathbf{x}_0^{\text{true}}] + \mathbf{e}_j^{\text{o}},$$

where the noise vector is defined by $\mathbf{e}_j^{\text{o}} \sim \mathcal{N}_{m_j}(\mathbf{0}, \rho^2\mathbf{I}_{m_j})$, with $\rho = 10^{-2}$. We assume that observations are available at times

$$t_j = j\tau, \text{ with } j = 0, 1, 2, 3, 4.$$

Consequently, we have $m = 5 \times m_j = 320 < n = 1024$, that is, the number of observations is less that the problem size to agree with the context of our study.

Finally, according to the perturbation definition of $\mathbf{e}^{\text{b}}$ and $\mathbf{e}_j^{\text{o}}$, the error covariance matrices are chosen to be multiple of the respective adequate identity matrix. We set $\mathbf{B} = \beta^2\ \mathbf{I}_n$ and $\mathbf{R} = \rho^2\ \mathbf{I}_m$.

4.2. Results

We give two illustrations of the comparison of the RPCG and the PSAS approaches used in the incremental inner-outer Algorithm 1. In this application matrix $\mathbf{G}$ is set to $\mathbf{I}_m$ (this corresponds to a CG minimization with a preconditioning by matrix $\mathbf{B}$) whereas the PSAS algorithm uses a preconditioning by $\mathbf{R}^{-1}$.

Illustration 1 The first example is devoted to the first outer iteration. This means that we do not take into account the nonlinear minimization process at this stage: we restrict our attention to the first quadratic minimization. To solve the linear system we consider and compare the PSAS and the RPCG. For both methods, the starting point (first guess) is equal to the background (i.e. Assumption 2). Figure 2 presents the 'true' initial state distribution and the background values over the square spatial grid.

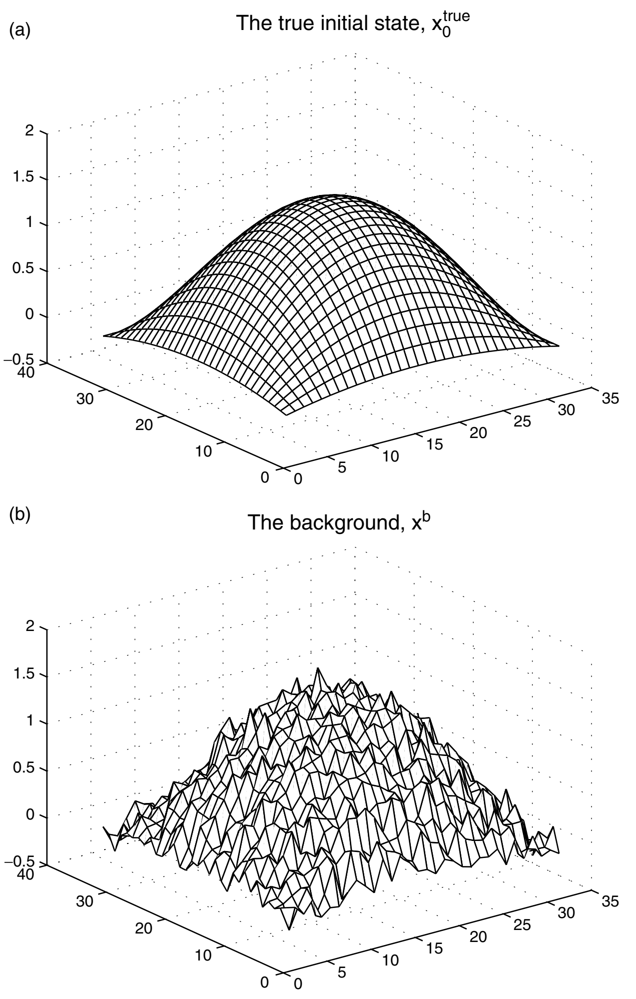

Figure 2. The 'true' initial state distribution (top): $x_0^{\text{true}}(u_q, v_r) = \gamma u_q(1-u_q)v_r(1-v_r)$, and the background distribution (bottom): $x^{\text{b}}(u_q, v_r) = x_0^{\text{true}}(u_q, v_r) + \beta$ randn $(n, 1)$, with $\gamma = 25$, $\beta = 10^{-1}$ and $0 < u_q, v_r < 1$.

The corresponding cost functions J are given by Figure 3.

On the one hand, we observe that in the PSAS approach the cost function appears to behave *erratically* during the iterations. The convergence is reached only around iteration 80.

On the other hand, the RPCG converges *monotonically* from the first iteration and reaches the convergence quicker than the PSAS approach, that is, at about iteration 40. This is no surprise since RPCG is minimizing J over an increasing sequence of nested Krylov subspaces.

Illustration 2 Now, we give the second example. The goal of these experiments is to study and compare the effect of increasing the number of inner iterations specially on the convergence of the *nonlinear function* in both the RPCG and the PSAS approaches. The results are shown in Figure 4 (20 inner iterations), Figure 6 (40 inner iterations) and Figure 6 (60 inner iterations) where the circle markers represent the nonlinear cost function values at outer iterations. The maximum number of outer iterations is set to 3 in these experiments. The starting

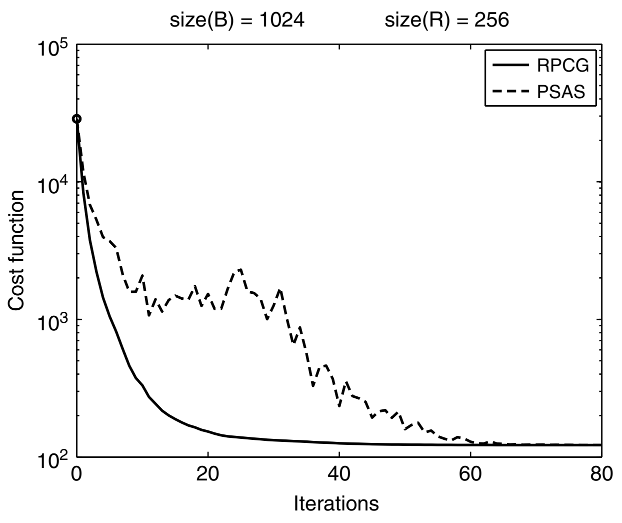

Figure 3. The cost function versus the inner iterations for one Gauss–Newton iteration.

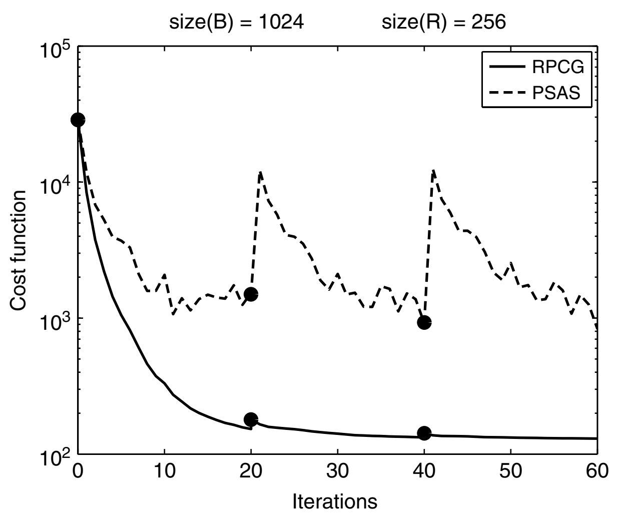

Figure 4. The cost function versus the inner iterations. The experiment consists of three Gauss–Newton iterations with 20 inner iterations.

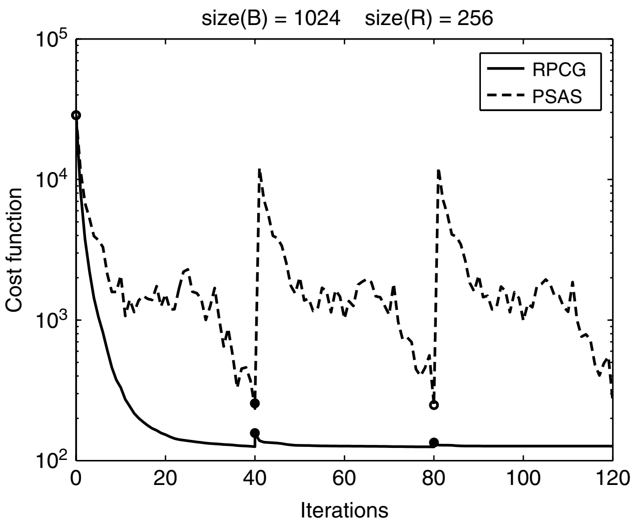

Figure 5. As Figure 4, but for 40 inner iterations.

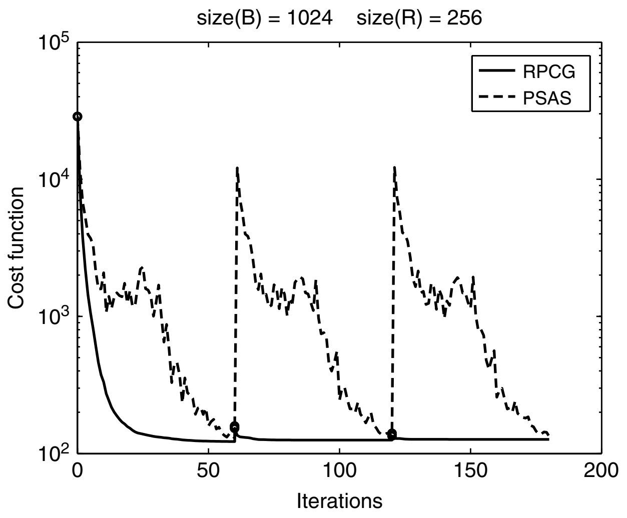

Figure 6. As Figure 4, but for 60 inner iterations.

point of the first outer iteration is again the background vector.

In this example, RPCG reduces the nonlinear cost function at each outer iteration. On the contrary, the nonlinear cost function may not converge when the number of inner iterations is 'small' and when using the PSAS approach, even starting the inner iterations 'close' to the solution (Figure 4). Nevertheless, when we increase the inner iterations number, we observe that the nonlinear cost function in the PSAS approach tends to behave as the one in the RPCG approach (Figures 5 and 6), and consequently it decreases at each outer iteration. This is consistent with the fact that if both algorithms were pushed to convergence they should find the same iterate in the incremental algorithm 1.

It is however known that neither PSAS nor RPCG would converge to a solution in the case of highly nonlinear functions unless a suitable *globalization strategy* (such as the linesearch or the trust-region techniques Nocedal and Wright (1999); Conn *et al.* (2000); Kelley (1999)) is adopted.

5. Conclusion

We have proposed a reformulation of the PCG when applied to linear inverse problems regularized by a quadratic background term of the kind

$$(\mathbf{B}^{-1} + \mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{H})\delta\mathbf{x}_0 = \mathbf{B}^{-1}(\mathbf{x}^{\mathrm{b}} - \mathbf{x}_0) + \mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{d}.$$

This new algorithm, which we call restricted preconditioned conjugate gradient algorithm (RPCG), is mathematically equivalent to the standard Hesteness–Stiefel preconditioned conjugate gradient method. It exploits the fact that in the standard method, all vectors involved in the method lie in a subspace of dimension m (or $m+1$ depending on the starting point), where $\mathbf{H} \in \Re^{m\times n}$. The RPCG algorithm is therefore naturally *cheaper* than PCG in terms of memory cost and even computational cost when m is significantly smaller than n, which is already an important point. This may be the case in the modern weak-constraint 4D-Var (e.g. Trémolet, 2006, 2007). However, we believe that the main interest of RPCG for data assimilation is when PCG is preconditioned with a Limited Memory Preconditioner (Tshimanga *et al.*, 2008) or when a re-orthogonalization scheme is used to alleviate the effect of round-off errors. In these cases PCG would require the additional storage of vectors of $\Re^n$, whereas the same techniques can be implemented with RPCG with vectors belonging to $\Re^m$, leading again to important memory savings when m is significantly smaller than n. In addition, as with PCG, RPCG requires exactly one call to the matrix–vector products by $\mathbf{B}$, $\mathbf{R}^{-1}$, $\mathbf{H}$ and $\mathbf{H}^{\mathrm{T}}$ per iteration. This is an important feature of our algorithm in applications where the matrix–vector product represents a dominant part of the computational cost. To illustrate the theoretical results, we have shown academic numerical results that illustrate the monotonic convergence obtained with RPCG, compared to the PSAS algorithm, which also involves vectors belonging to $\Re^m$.

The new algorithm raises many questions. Its convergence is well known in exact arithmetic from the equivalence with standard PCG. However, the behaviour of the method in the presence of round-off error needs further investigation; in particular, we would like to assess whether the fact that we thoroughly take into account the system structure gives better behaviour with respect to round-off errors. At least, we show that full re-orthogonalization is now affordable (in terms of memory) when m is significantly smaller than n, showing the practical importance of RPCG. Another important fact is to find efficient preconditioners $\mathbf{F}$ that satisfy the relation $\mathbf{FH}^{\mathrm{T}} = \mathbf{BH}^{\mathrm{T}}\mathbf{G}$ for some $\mathbf{G}$; this condition is sufficient for these preconditioners to be used in RPCG.

Finally, we would like to implement the technique in a real-life data assimilation system that uses a weak constraint formulation to account for model errors. In these systems, the large size n of the state variables makes RPCG an interesting alternative to the PCG algorithm.

Acknowledgements

The authors would like to thank Anthony Weaver for stimulating discussions and for suggesting a number of improvements on this paper. Part of this work was supported by the foundation RTRA/STAE (Réseau Thématique de Recherche Avancée/Sciences et Technologies pour l'Aéronautique et l'Espace).

References

Auroux D. 2007. Generalization of the dual variational data assimilation algorithm to a nonlinear layered quasi-geotrophic ocean model. *Inverse Problems* **23**: 2485–2503.

Axelsson O. 1996. *Iterative Solution Methods*. Cambridge University Press: Cambridge, UK.

Conn AR, Gould NIM, Toint PL. 2000. *Trust-Region Methods*. Number 01 in MPS-SIAM Series on Optimization. SIAM: Philadelphia, USA.

Courtier P. 1997. Dual formulation of four-dimensional variational assimilation. *Q. J. R. Meteorol. Soc.* **123**: 2449–2461.

Courtier P, Thépaut J-N, Hollingsworth A. 1994. A strategy for operational implementation of 4D-Var using an incremental approach. *Q. J. R. Meteorol. Soc.* **120**: 1367–1388.

Demmel JW. 1997. *Applied numerical linear algebra*. SIAM: Philadelphia, USA.

El Akkraoui A, Gauthier P, Pellerin S, Buis S. 2008. Intercomparison of the primal and dual formulations of variational data assimilation. *Q. J. R. Meteorol. Soc.* **134**: 1015–1025.

Fisher M. 1998. 'Minimization algorithms for variational data assimilation'. Pp 364–385 in Proceedings of Seminar on Recent Developments in Numerical Methods for Atmospheric Modelling, 7–11 Sep 1998. ECMWF: Reading, UK.

Gilbert JC, Lemaréchal C. 1989. Some numerical experiments with variable-storage quasi-Newton algorithms. *Math. Prog. Series B* **45**: 407–436.

Golub GH, Van Loan CF. 1996. *Matrix Computations*. Johns Hopkins University Press: Baltimore, USA.

Gratton S, Lawless AS, Nichols NK. 2007. Approximate Gauss–Newton methods for nonlinear least-squares problems. *SIAM J. Optimization* **18**: 106–132.

Hestenes MR, Stiefel E. 1952. Methods of conjugate gradients for solving linear systems. *J. Res. Nat. Bureau Standards* **49**: 409–436.

Higham NJ. 2002. *Accuracy and Stability of Numerical Algorithms*. SIAM: Philadelphia, USA.

Kelley CT. 1999. *Iterative Methods for Optimization*. SIAM: Philadelphia, USA.

Lawless AS, Nichols NK. 2006. Inner-loop stopping criteria for incremental four-dimensional variational data assimilation. *Mon. Weather Rev.* **134**: 3425–3435.

Lawless AS, Gratton S, Nichols NK. 2005a. Approximate iterative methods for variational data assimilation. *Int. J. Numer. Methods Fluids* **47**: 1129–1135.

Lawless AS, Gratton S, Nichols NK. 2005b. An investigation of incremental 4D-Var using non-tangent linear models. *Q. J. R. Meteorol. Soc.* **131**: 459–476.

Morales JL, Nocedal J. 2000. Automatic preconditioning by limited memory quasi-Newton updating. *SIAM J. Optimization* **10**: 1079–1096.

Nocedal J, Wright SJ. 1999. *Numerical Optimization. Series in Operations Research*. Springer Verlag: Heidelberg, Berlin, New York.

Rabier F. 2005. Overview of global data assimilation developments in numerical weather-prediction centres. *Q. J. R. Meteorol. Soc.* **131**: 3215–3233.

Trémolet Y. 2006. Accounting for an imperfect model in 4D-Var. *Q. J. R. Meteorol. Soc.* **132**: 2483–2504.

Trémolet Y. 2007. Model-error estimation in 4D-Var. *Q. J. R. Meteorol. Soc.* **133**: 1267–1280.

Tshimanga J. 2007. 'On a class of limited memory preconditioners for large-scale nonlinear least-squares problems (with application to variational ocean data assimilation)'. PhD thesis, Department of Mathematics, University of Namur, Belgium.

Tshimanga J, Gratton S, Weaver AT, Sartenaer A. 2008. Limited-memory preconditioners with application to incremental four-dimensional variational data assimilation. *Q. J. R. Meteorol. Soc.* **134**: 751–769.

van der Sluis A, van der Vorst HA. 1986. The rate of convergence of conjugates gradients. *Numerische Mathematik* **48**: 543–560.

van der Vorst HA. 2003. *Iterative Krylov methods for large linear systems*. Cambridge University Press: Cambridge, UK.

Weaver AT, Vialard J, Anderson DLT. 2003. Three- and four-dimensional variational assimilation with a general circulation model of the tropical Pacific Ocean. Part I: Formulation, internal diagnostics, and consistency checks. *Mon. Weather Rev.* **131**: 1360–1378.