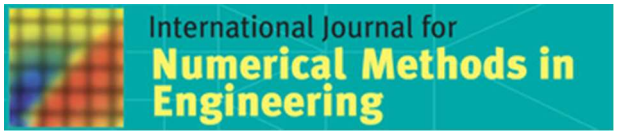

A Reduced Order Model-based preconditioner for the efficient solution of transient diffusion equations

Journal:	*International Journal for Numerical Methods in Engineering*
Manuscript ID	NME-Mar-16-0146
Wiley - Manuscript type:	Research Article
Date Submitted by the Author:	04-Mar-2016
Complete List of Authors:	Pasetto, Damiano; Ecole Polytechnique Federale de Lausanne, Laboratory of Ecohydrology Ferronato, Massimiliano; University of Padova, Department of Civil, Environmental and Architectural Engineering (DICEA) Putti, Mario; University of Padua, Dept. of Mathematics
Keywords:	Model Order Reduction, Proper Orthogonal Decomposition, Preconditioning, Iterative methods, Diffusion Equation

INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING

Int. J. Numer. Meth. Engng 0000; **00**:2–**??**

Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/nme

A reduced order model-based preconditioner for the efficient solution of transient diffusion equations

Damiano Pasetto[1]*, Massimiliano Ferronato[2], Mario Putti[3]

[1]*Laboratory of Ecohydrology, École polytechnique fédérale de Lausanne, Switzerland*

[2]*Department of Civil, Architectural and Environmental Engineering, University of Padova, Italy*

[3]*Department of Mathematics, University of Padova, Italy*

SUMMARY

This paper presents a novel class of preconditioners for the iterative solution of the sequence of symmetric positive definite linear systems arising from the numerical discretization of transient parabolic and self-adjoint partial differential equations. The preconditioners are obtained by nesting appropriate projections of reduced order models into the classical iteration of the preconditioned conjugate gradient (PCG). The main idea is to employ the reduced order solver to project the residual associated with the conjugate gradient iterations onto the space spanned by the reduced bases. This approach is particularly appealing for transient systems where the full model solution has to be computed at each time step. In these cases the natural reduced space is the one generated by full-model solutions at previous time steps. When increasing the size of the projection space, the proposed methodology highly reduces the system conditioning number and the number of PCG iterations at every time step. The cost of the application of the preconditioner linearly increases with the size of the projection basis, and a trade-off must be found to effectively reduce the PCG computational cost. The quality and efficiency of the proposed approach is finally tested in the solution of groundwater flow models. Copyright © 0000 John Wiley & Sons, Ltd.

Received . . .

KEY WORDS: Model Order Reduction, Proper Orthogonal Decomposition, Preconditioning, Iterative methods, Diffusion equation

1. INTRODUCTION

The numerical solution of three-dimensional (3D) parabolic partial differential equations (PDEs) is a common task in several advanced engineering applications. Numerical discretization is typically performed via the method of lines combining finite element or finite difference techniques in space with a time marching scheme. For self-adjoint problems this approach produces a sequence of large size, sparse and symmetric positive definite (SPD) linear systems whose solution generally constitutes the most time- and memory-demanding task in a transient simulation. The Preconditioned Conjugate Gradient (PCG) scheme is typically the method of choice for the solution of such systems, with several different general-purpose preconditioners already available to accelerate the convergence [1]. Among the most traditional preconditioners employed for the solutions of SPD systems we mention here the Jacobi preconditioner and the Incomplete Cholesky factorization with zero fill-in (IC0) [2, 3, 4]. These algebraic approaches can be easily implemented as a black box in the discrete solver of the transient equation, since they are directly computed from the system matrix and do not depend upon the particular physics of the problem. However, general-purpose algebraic preconditioners are not always the optimal choice for the specific case at hand, especially when approximate solutions can be obtained at a relatively low computational cost. For example, several authors [5, 6, 7, 8] have explored the possibility of computing suitable preconditioners starting from a simplification of the spatial differential operators governing the physical equations. In this paper we present a variation of this approach based on the use of model order reduction techniques.

Model order reduction methods, such as Proper Orthogonal Decomposition (POD), are approaches designed to yield low-cost approximations to the solution of linear PDEs [9, 10, 11, 12, 13]. The main underlying idea is the Galerkin projection of the PDE onto the space generated by a few spatially distributed and linearly independent basis vectors (forward step). The resulting system of linear ordinary differential equations has a low dimension and its analytical or numerical

*Correspondence to: Bâtiment GR, EPFL, Station 2, CH-1015 Lausanne (Switzerland)

Prepared using nmeauth.cls http://mc.manuscriptcentral.com/nme DOI: 10.1002/nme

solution is available at a negligible computational cost. The backward projection of this low-dimensional solution onto the high-dimensional full-model space (backward step) results in an approximation of the solution of the original linear PDE. The expensive computation of the basis vectors typically occurs offline, starting from a limited number of snapshots, i.e., solutions of the full model at given times. The accuracy and the efficiency of the reduced model procedure depend on the number and quality of the basis vectors employed in the reduction. The addition of basis vectors reduces the approximation errors, but increases the computational cost of the offline forward and backward steps. Model order reduction techniques are frequently applied to obtain a fast solution to computationally complex and parameter-dependent problems such as model calibration [14, 15], Monte Carlo simulations [16, 17, 18] and experimental design [19].

One of the main issues arising when applying model order reduction to transient problems is the need of updating the basis functions during the transient simulation. In fact, the distance between the space generated by the basis functions and the system solution might increase along the model simulation, thus causing a deterioration in the quality of the reduced model. For this reason model order reduction techniques have been recently applied to improve the convergence of iterative methods. Markovinovìc et al. [20] employ the reduced order model solution as initial guess for the iterative method obtaining an acceleration of the convergence. Astrid et al. [21] show the benefits of replacing the algebraic multigrid method with a two stage iterative method based on a Constrained Pressure Residual solver in combination with POD for the solution of a two-phase reservoir model. Jiang [22] introduces a reduced order model preconditioner in the stationary Richardson iteration.

The present paper describes how a POD model obtained from a few snapshots can be employed as a preconditioner for the PCG algorithm in the solution of a transient diffusion PDE. The idea is based on the fact that the POD solution resembles the full model numerical solution. Hence, the application of the POD operator can be regarded as an approximation of the application of the inverse of the full model matrix but in a smaller vector space. From this starting point, we derive a purely algebraic approach by suitably projecting a small number of appropriate snapshots. The overall algorithmic framework resembles from the classical two-grid procedure used in Algebraic

Prepared using nmeauth.cls http://mc.manuscriptcentral.com/nme DOI: 10.1002/nme

Multigrid (AMG) methods [23, 24, 25] where the system error arising from the application of a smoothing matrix is projected back and forth on a coarser level. In our procedure the restriction and prolongation AMG steps are replaced by the forward and backward POD operators at each PCG iteration. The resulting algorithm is compared to standard general-purpose preconditioners in the solution of large size 3D diffusion problems. Such an approach can be easily generalized to other applications governed by similar PDEs.

2. PROBLEM EQUATIONS

The solution of a transient diffusion problem defined on a three dimensional domain Ω is governed in space and time by the parabolic PDE:

$$S_s(\mathbf{x})\frac{\partial s(t,\mathbf{x})}{\partial t} - \nabla \cdot [D(\mathbf{x})\nabla s(t,\mathbf{x})] = f(t,\mathbf{x}), \quad t \in [0,T], \quad \mathbf{x} \in \Omega \subset \mathbb{R}^3, \tag{1}$$

where $s \in \mathbb{R}$ is the unknown solution, $S_s \in \mathbb{R}$ is a storage coefficient, $D \in \mathbb{R}^{3\times3}$ is the diffusion tensor, and $f \in \mathbb{R}$ represents the source/sink term. With no loss of generality, the initial solution is $s(0,\mathbf{x}) = 0$ for any $\mathbf{x} \in \Omega$ with homogeneous boundary conditions, i.e., $s(t,\mathbf{x}) = 0$ for $\mathbf{x}$ in the Dirichlet boundary Γ_D and $D(\mathbf{x})\nabla s(t,\mathbf{x}) \cdot \mathbf{n}(\mathbf{x}) = 0$ for $\mathbf{x}$ in the Neumann boundary Γ_N, where $\mathbf{n} \in \mathbb{R}^3$ is the outward unit normal.

We use the method of lines to solve eq. (1). Spatial discretization is obtained by Galerkin finite elements with piecewise linear basis functions using a grid formed by n nodes and r tetrahedral elements to discretize the domain Ω. Other discretization methods can be used, such as higher order or mixed FEM, finite volumes/differences, or pseudo-spectral methods, without changing the purpose and results of the present study. All these approaches entail solving a system of first order differential equations that can be written as:

$$H\mathbf{s} + M\frac{d\mathbf{s}}{dt} + \mathbf{q} = 0 \tag{2}$$

Prepared using nmeauth.cls http://mc.manuscriptcentral.com/nme DOI: 10.1002/nme

where $H \in \mathbb{R}^{n\times n}$ is the SPD stiffness matrix, $M \in \mathbb{R}^{n\times n}$ is the (possibly lumped) mass matrix with strictly positive elements, $\mathbf{s} \in \mathbb{R}^n$ is the vector of unknowns, and $\mathbf{q} \in \mathbb{R}^n$ represents the discretized forcing term. Using a backward Euler time marching scheme, the vector $\mathbf{s}^{j+1}$ at the $(j+1)$-th time step is obtained by solving the linear system:

$$\left(H + \frac{M}{\Delta t_{j+1}}\right)\mathbf{s}^{j+1} = \frac{M}{\Delta t_{j+1}}\mathbf{s}^{j} - \mathbf{q} \tag{3}$$

where $\Delta t_{j+1} = t_{j+1} - t_j$ is the variable integration step in time.

The solution $\mathbf{s}^{j+1}$ can be also obtained as $\mathbf{s}^{j+1} = \mathbf{s}^{j} + \mathbf{y}^{j+1}$, where $\mathbf{y}^{j+1}$ is the correction with respect to the previous step. From (3), $\mathbf{y}^{j+1}$ is the solution of the system:

$$\left(H + \frac{M}{\Delta t_{j+1}}\right)\mathbf{y}^{j+1} = -H\mathbf{s}^{j} - \mathbf{q}. \tag{4}$$

At any time step the system (4) can be rewritten as

$$A\mathbf{u} = \mathbf{f} \tag{5}$$

where the system matrix A reads:

$$A = \left(H + \frac{M}{\Delta t_{j+1}}\right), \tag{6}$$

and the solution and right-hand side vectors are $\mathbf{u} = \mathbf{y}^{j+1}$ and $\mathbf{f} = -H\mathbf{s}^{j} - \mathbf{q}$, respectively.

The matrix $A \in \mathbb{R}^{n\times n}$ is SPD, sparse, and typically with a large size, so that the use of the PCG method is virtually mandatory. The choice of the preconditioner is a decisive factor for the efficiency and the convergence rate of PCG. Here, we propose a preconditioner based on a model order reduction of system (4).

Prepared using nmeauth.cls http://mc.manuscriptcentral.com/nme DOI: 10.1002/nme

3. PROJECTION-BASED REDUCED ORDER MODELS

In model order reduction methods the solution vector $\mathbf{s}^{j+1}$ is approximated by the vector $\tilde{\mathbf{s}}^{j+1} \in \mathbb{R}^n$ obtained as a linear combination of a small set of accurately chosen basis vectors $\mathbf{p}_1, \dots, \mathbf{p}_m \in \mathbb{R}^n$ with $m << n$, plus a reference solution $\mathbf{s}^*$ (see, e.g. Vermeulen et al. [26]):

$$\mathbf{s}^{j+1} \approx \tilde{\mathbf{s}}^{j+1} = \mathbf{s}^* + \sum_{i=1}^{m} \mathbf{p}_i a_i^{j+1} = \mathbf{s}^* + P\mathbf{a}^{j+1} \tag{7}$$

where $P \in \mathbb{R}^{n \times m}$ is the matrix $P = [\mathbf{p}_1, \dots, \mathbf{p}_m]$ and $\mathbf{a}^{j+1} \in \mathbb{R}^m$ is the coefficient vector at time step $j+1$, $\mathbf{a}^{j+1} = \left[a_1^{j+1}, \dots, a_m^{j+1}\right]^T$. Hence, given P, the m coefficients $a_1, \dots, a_m$ are the only unknowns to be determined at every time step, while the basis vectors can be computed in an offline stage. Note that the full spatial variability of the solution is described by the basis vectors $\mathbf{p}_1, \dots, \mathbf{p}_m$. The computation of the coefficient vector $\mathbf{a}^{j+1}$ is performed using a Galerkin projection onto the space $\mathcal{P} = \text{span}\{\mathbf{p}_1, \dots, \mathbf{p}_m\}$. To this aim, the solution $\mathbf{s}^{j+1}$ in (3) is replaced with the approximation $\tilde{\mathbf{s}}^{j+1}$ given in (7) and the resulting residual $\tilde{\mathbf{r}}^{j+1}$ is orthogonalized with respect to the basis vectors $\mathbf{p}_i$. In other words, the following Galerkin problem is solved:

$$\text{Find } \tilde{\mathbf{s}}^{j+1} \in \mathbf{s}^* + \mathcal{P}, \text{ such that } \tilde{\mathbf{r}}^{j+1} \perp \mathcal{P}$$

We set the reference solution equal to the solution at the previous time step, $\mathbf{s}^* = \mathbf{s}^j$, i.e.,

$$\mathbf{y}^{j+1} \approx \tilde{\mathbf{y}}^{j+1} = P\mathbf{a}^{j+1} \tag{8}$$

so that the innovation vector $\tilde{\mathbf{y}}^{j+1} = \tilde{\mathbf{s}}^{j+1} - \mathbf{s}^j \in \mathcal{P}$. The resulting reduced $(m \times m)$ linear system becomes:

$$\tilde{A}\tilde{\mathbf{a}} = \tilde{\mathbf{f}}, \tag{9}$$

where

$$\tilde{A} = \left(P^T H P + \frac{P^T M P}{\Delta t_{j+1}}\right), \quad \tilde{\mathbf{a}} = \mathbf{a}^{j+1}, \quad \tilde{\mathbf{f}} = P^T H P \mathbf{a}^j - P^T \mathbf{q},$$

with matrix $\tilde{A}$ being SPD since both H and M are so. If $m \ll n$, the solution of (9) by a direct method is computationally inexpensive. Equation (9) is addressed as the Reduced Order Model (ROM). In contrast, we denote as Full System Model (FSM) system (4) in the full dimensional space.

The construction of the orthogonal basis vectors $\mathbf{p}_i$ is the most important and computationally expensive task of the model order reduction procedure. The basis vectors are typically computed from a set of full model system solutions, called snapshots. Let $Y \in \mathbb{R}^{n \times z}$ be the matrix of z snapshots, $Y = [\mathbf{y}^{r_1}, \ldots, \mathbf{y}^{r_z}]$, where r_i, $i = 1, \ldots, z$ are the time steps at which the solutions are collected. In the POD approach, the basis vectors are the m eigenvectors corresponding to the m largest eigenvalues of the matrix YY^T [26, 13]. This procedure resembles the Principal Component Analysis and for this reason the basis vectors are also called principal components. Alternatively, the Reduced Basis (RB) approach [14, 27] proposes a different set of basis vectors. Given a ROM of size $l < m$, the RB method increments the space $\mathcal{P}$ by a Gram-Schmidt orthogonalization of the snapshot that maximizes the error between FSM and ROM, following a standard greedy algorithm. The error between FSM and ROM is typically evaluated using a posteriori error estimations. In both cases, the ROM quality and computational efficiency depends upon the number of basis vectors used in the reduction. The employment of a small number of basis vectors may jeopardize the accuracy of the reduced model solution and a compromise between cost and accuracy must be achieved. In this work we restrict ourself to the POD method for its simplicity and computational efficiency when applied to parabolic problems.

4. ROM-BASED PRECONDITIONER

The objective of preconditioning an SPD linear system by K^{-1} is to reduce the condition number κ associated to the system matrix A, i.e., $\kappa(K^{-1}A) << \kappa(A)$. This accelerates, or in finite precision even allows for, the convergence of the PCG method with the aim of reducing the overall computational cost. To comply with this task the preconditioning matrix K^{-1} should be such that

Prepared using nmeauth.cls http://mc.manuscriptcentral.com/nme DOI: 10.1002/nme

Algorithm 1 Two-grid AMG algorithm for the solution of $A\mathbf{v} = \mathbf{w}$

Set ν_1, ν_2, $\mathbf{v}_0$, S, and P
for $k = 0, 1, \ldots, \nu_1 - 1$ **do**
    $\mathbf{v}_{k+1} = (I - SA)\mathbf{v}_k + S\mathbf{w}$
end for
$\mathbf{r} = \mathbf{w} - A\mathbf{v}_{\nu_1}$
$\tilde{\mathbf{r}} = P^T\mathbf{r}$
$\tilde{\mathbf{e}} = \left(P^T AP\right)^{-1}\tilde{\mathbf{r}}$
$\mathbf{e} = P\tilde{\mathbf{r}}$
$\mathbf{v}_{\nu_1} \leftarrow \mathbf{v}_{\nu_1} + \mathbf{e}$
for $k = \nu_1, \ldots, \nu_1 + \nu_2 - 1$ **do**
    $\mathbf{v}_{k+1} = (I - SA)\mathbf{v}_k + S\mathbf{w}$
end for
$\mathbf{v} = \mathbf{v}_{\nu_1+\nu_2}$

$K^{-1}A$ is spectrally close to I, though preserving a large sparsity to render its application to a vector computationally efficient.

With the ROM approach the solution of system (5) is approximated by (7), i.e.,

$$\mathbf{u} \approx \tilde{\mathbf{u}} = P\tilde{A}^{-1}\tilde{\mathbf{f}} \approx P\tilde{A}^{-1}P^T\mathbf{f}. \tag{10}$$

This means that $P\tilde{A}^{-1}P^T$ can be regarded as a low-rank approximation of A^{-1}. However, simply setting $K^{-1} = P\tilde{A}^{-1}P^T$ is not viable as this matrix has rank $m < n$ and thus prevents the expansion of the Krylov subspace beyond dimension m. To avoid this problem, the following approach can be used. Any PCG iteration requires one application of the preconditioning matrix for computing $\mathbf{e} = K^{-1}\mathbf{r}$, $\mathbf{r}$ being the current residual of system (5). This can be regarded as an approximation of the error associated to the current solution $\mathbf{u}$. Our idea is to apply the reduced order model for the computation of $\mathbf{e}$. From an algebraic point of view, the ROM approach can be regarded as restricting the space $\mathbb{R}^n$ to a subspace of size m, finding the exact solution in $\mathbb{R}^m$ and prolongating the reduced solution back to $\mathbb{R}^n$. This is very similar to a two-grid correction using an AMG approach. To find an approximate solution of the linear system $A\mathbf{v} = \mathbf{w}$, the classical algorithm for a two-grid AMG step runs as follows.

Prepared using nmeauth.cls http://mc.manuscriptcentral.com/nme DOI: 10.1002/nme

1. Perform ν_1 applications of a smoother, S, to the native system $A\mathbf{v} = \mathbf{w}$:

$$\mathbf{v}_{k+1} = (I - SA)\mathbf{v}_k + S\mathbf{w}, \qquad k = 1, \ldots, \nu_1.$$

   The smoother S is typically an operator that applies a stationary iteration, e.g., Jacobi or symmetric Gauss-Seidel for SPD problems.
2. Compute the residual $\mathbf{r}$ after ν_1 iterations of the smoother, $\mathbf{r} = \mathbf{w} - A\mathbf{v}_{\nu_1}$.
3. Apply the restriction operator P^T to project $\mathbf{r}$ into the reduced space of dimension m, $\tilde{\mathbf{r}} = P^T\mathbf{r}$.
4. Compute the error in the reduced space, $\tilde{\mathbf{e}} = \tilde{A}^{-1}\tilde{\mathbf{r}}$.
5. Apply the prolongation operator P to extend $\tilde{\mathbf{e}}$ to the larger space, $\mathbf{e} = P\tilde{\mathbf{e}}$.
6. Update the solution obtained after the smoothing iterations (point 1.) with $\mathbf{e}$.
7. Perform ν_2 applications of S starting from the last corrected solution. If $\nu_1 = \nu_2$, the overall preconditioner is symmetric.

This procedure is summarized in Algorithm 1. A first preconditioner can be derived from this algorithm by setting $\nu_1 = \nu_2 = 1$ and $\mathbf{v}_0 = 0$. In this case we have:

$$\mathbf{v} = (2I - SA)\, S\mathbf{w} + (I - SA)\, P\tilde{A}^{-1}P^T\, (I - SA)^T\, \mathbf{w} \tag{11}$$

and the matrix

$$G_m^{-1} = 2S - SAS + (I - SA)\, P\tilde{A}^{-1}P^T\, (I - SA)^T \tag{12}$$

is an approximation of A^{-1} that can be used as a preconditioner in the PCG algorithm. The subindex m indicates the dependence of the preconditioner on the selected number of basis vectors m. If both A and S are SPD, so is G_m^{-1}. Moreover, consistency is ensured as in the limit of $m \to n$ we have that $P\tilde{A}^{-1}P^T \to A^{-1}$ and

$$\lim_{m \to n} G_m^{-1} = 2S - SAS + A^{-1} - 2S + SAS = A^{-1}.$$

The application of this preconditioner, however, can be quite expensive, depending on the choice of the matrix S. A way to reduce the computational cost can be turning off the post-smoothing step in Algorithm 1, i.e., setting $\nu_2 = 0$ to reduce the number of matrix-vector products. If smoothing is unbalanced, it is well-known that the preconditioner becomes non-symmetric and the standard convergence theory of PCG is no longer valid. However, nonsymmetric preconditioning of the PCG algorithm does not always prevent convergence [28, 29, 30, 31]. In particular, Bouwmeester et al. [32] have recently proved that turning off post-smoothing in a V-cycle AMG preconditioner causes the loss of the PCG global optimality property, but convergence is still ensured because the method is locally optimal, i.e., the residual norm keeps decreasing during the iterations not slower than the Preconditioned Steepest Descent method. From a computational point of view, this approach can be advantageous if the PCG convergence is preserved with a cheaper cost for the preconditioner application.

If we set $\nu_1 = 1$, $\nu_2 = 0$, and $\mathbf{v}_0 = 0$, Algorithm 1 now reads:

$$\mathbf{v} = S\mathbf{w} + P\tilde{A}^{-1}P^T(\mathbf{w} - AS\mathbf{w}), \tag{13}$$

so that matrix

$$K_m^{-1} = S + P\tilde{A}^{-1}P^T(I - AS) \tag{14}$$

is another approximation of A^{-1} that can be used as a preconditioner. We denote K_m^{-1} in (14) as the ROM preconditioner, and matrix S is called support matrix for the ROM preconditioner. As support matrix, we elect to use either the diagonal Jacobi matrix or the incomplete Cholesky factorization with zero fill-in (IC0), though other choices are obviously possible. Consistency of K_m^{-1} in (14) is trivially still ensured. Although K_m^{-1} is non-symmetric, the convergence of PCG is guaranteed under non-restrictive assumptions with a number of iterations bounded by the number of iterations of S-preconditioner PCG. To see this we write the ROM-preconditioned matrix as:

$$K_m^{-1}A = SA + P\tilde{A}^{-1}P^TA - P\tilde{A}^{-1}P^TASA = SA + R\,(I - SA) = I - E_S E_P \tag{15}$$

Prepared using nmeauth.cls http://mc.manuscriptcentral.com/nme DOI: 10.1002/nme

where $R = P\tilde{A}^{-1}P^T A$ and

$$E_S = I - SA \tag{16}$$

$$E_P = I - R \tag{17}$$

Matrices E_S and E_P can be regarded as error matrices measuring the quality of S and $P\tilde{A}^{-1}P^T$ as approximations of A^{-1}. The eigenvalues of the preconditioned matrix are therefore:

$$\lambda(K_m^{-1}A) = 1 - \lambda(E_S E_P) \tag{18}$$

and their distance from the unity is bounded by:

$$\left|\lambda(K_m^{-1}A) - 1\right| = |\lambda(E_S E_P)| \leq \|E_S\| \, \|E_P\| \tag{19}$$

As already observed, matrix $P\tilde{A}^{-1}P^T$ is the prolongation in $\mathbb{R}^{n\times n}$ of the ROM matrix $\tilde{A}^{-1} \in \mathbb{R}^{m\times m}$. Thus, matrix $R \in \mathbb{R}^{n\times n}$ is of rank m and the eigenvalues of E_P can be written as:

$$|\lambda(E_P)| = \begin{cases} 1 & \text{with multiplicity } n-m \\ \epsilon_i & i = 1, \ldots, m \end{cases} \tag{20}$$

If the reduced order model is a good approximation of the full system model we expect the non-zero eigenvalues of R to be close to 1, so that $\epsilon_i \approx 0$. Assuming that this is true, then $\|E_P\| = 1$ and inequality(19) reads:

$$\left|\lambda(K_m^{-1}A) - 1\right| \leq \|E_S\| \tag{21}$$

which shows that, under the above assumption, the spectral properties of $K_m^{-1}A$ are not worse than SA.

Prepared using nmeauth.cls http://mc.manuscriptcentral.com/nme DOI: 10.1002/nme

Algorithm 2 Application of the ROM preconditioner: $\mathbf{v} = K_m^{-1}\mathbf{w}$

1: Set S and P
2: $Q = AP$
3: $\tilde{A} = P^T Q$
4: $\mathbf{v} = S\mathbf{w}$
5: $\tilde{\mathbf{r}} = P^T\mathbf{w} + Q^T\mathbf{v}$
6: $\tilde{\mathbf{e}} = \tilde{A}^{-1}\tilde{\mathbf{r}}$
7: $\mathbf{e} = P\tilde{\mathbf{e}}$
8: $\mathbf{v} \leftarrow \mathbf{v} + \mathbf{e}$

4.1. Computational cost for preconditioner application

In this section we analyze the computational cost associated with the application and construction of the ROM preconditioner. For the computation of the reduced order matrix, it is useful to define the projected matrix $Q = AP = Q_H + Q_M/\Delta t_{k+1}$, $Q \in \mathbb{R}^{n\times m}$, where $Q_H = HP \in \mathbb{R}^{n\times m}$ and $Q_M = MP \in \mathbb{R}^{n\times m}$. Since Q_H and Q_M do not depend on the time step, they are computed at the beginning of the simulation. The preconditioning matrix (14) can be also written as:

$$K_m^{-1} = S + P\tilde{A}^{-1}(P^T - Q^T S) \tag{22}$$

The application of (22) to a vector $\mathbf{w}$ consists of the following operations (Algorithm 2).

1. Application of the support matrix S:

$$\mathbf{v} = S\mathbf{w}. \tag{23}$$

   The cost of this operation depends on the choice of S. For example, if S is the Jacobi preconditioner, the cost is one scalar product of $\mathbb{R}^n$. Instead, using IC0 as support matrix, the cost of (23) is σ scalar products, where σ is the average number of non-zero entries in each row of A, i.e., a measure of its sparsity.

2. Application of matrices P^T and Q^T:

$$\tilde{\mathbf{r}} = P^T\mathbf{w} + Q^T\mathbf{v}. \tag{24}$$

   The cost is equal to $2m$ scalar products.

3. Computation of the error in the low dimensional space:

$$\tilde{\mathbf{e}} = \tilde{A}^{-1}\tilde{\mathbf{r}} \tag{25}$$

As the size m is much smaller than n, it does not effect the overall computational cost of the preconditioner application. The cost for computing equation (25) is $O(m^3)$ and negligible.

4. Prolongation to the large dimension m:

$$\mathbf{e} = P\tilde{\mathbf{e}}. \tag{26}$$

The cost is equal to m scalar products.

5. Final computation of $\mathbf{v}$ by a vector update:

$$\mathbf{v} \leftarrow \mathbf{v} + \mathbf{e}. \tag{27}$$

The additional cost incurred by the application of the ROM preconditioner with respect to the application of the support matrix S alone is equal to $3m$ scalar products. Recalling that each PCG iteration involves 7 scalar products and one matrix-vector product, which costs σ scalar products, the total computational cost of each ROM-preconditioned PCG iteration is: (a) $(8 + \sigma + 3m)$ scalar products if S=diag$(A)^{-1}$, or (b) $(7 + 2\sigma + 3m)$ scalar products if S=IC0. For example, with $m = 7$ and $\sigma = 14$ each iteration performs twice the number of scalar products of the standard PCG with IC0. With smaller values of m, say 1 or 2, the additional cost due to the use of the ROM preconditioner is almost negligible.

As already observed, the ROM preconditioner of equation (14) is not symmetric. To restore the symmetry and the global PCG optimality at a smaller cost than using equation (12) the following

Prepared using nmeauth.cls http://mc.manuscriptcentral.com/nme DOI: 10.1002/nme

variant of the ROM preconditioner can be used:

$$K_{S,m}^{-1} = S + P\tilde{A}^{-1}P^T - \frac{P\tilde{A}^{-1}Q^TS + S^TQ\tilde{A}^{-1}P^T}{2}$$
$$= S + P\tilde{A}^{-1}(P^T - \frac{1}{2}Q^TS) - \frac{1}{2}S^TQ\tilde{A}^{-1}P^T. \quad (28)$$

Notice that $K_{S,m}^{-1}$ arises from a purely algebraic symmetrization of K_m^{-1} and does not restore the symmetry of the underlying two-grid cycle. We name $K_{S,m}^{-1}$ the Symmetric ROM (SROM) preconditioner. To avoid the double application of S at every iteration, it is useful to compute the matrix $S^TQ \in \mathbb{R}^{n\times m}$, at the beginning of every time step. The computational cost of the SROM preconditioner (28) increases by m scalar products with respect to the ROM preconditioner of equation (14). Table I summarizes the number of scalar products for each iteration of the PCG algorithm with different combinations of the preconditioner.

4.2. Basis vectors and preconditioner construction

The application of K_m^{-1} and $K_{S,m}^{-1}$ requires the computation of:

1. the m basis vectors (matrix P);
2. the reduced model matrix, $\tilde{A} = P^TAP$;
3. the matrices Q_H and Q_M;
4. the support matrix S.

Once P is known, steps 2 and 3 are well defined. The construction of the support matrix S is a standard operation in PCG. Thus we focus next on two possible procedures for the computation of the basis vectors.

4.2.1. Optimal basis vectors. Using the idea of POD [9], the reduced order model is constructed employing as basis vectors the principal components of a set of z snapshots, i.e., the set obtained collecting the full model solutions at z time steps. We represent this set as $Y = [\mathbf{y}^{r_1}, \ldots, \mathbf{y}^{r_z}]$. The basis vectors $\mathbf{p}_1, \ldots, \mathbf{p}_m$ are the eigenvectors corresponding to the largest eigenvalues of the matrix

$YY^T \in \mathbb{R}^{n\times n}$. This eigenvalue analysis is performed in the reduced dimension computing the eigenvalues/eigenvectors of the matrix $Y^TY \in \mathbb{R}^{z\times z}$. With this approach the matrix P is evaluated offline. Also matrices $Q_H = HP$, $\tilde{H} = P^TQ_H$, $Q_M = MP$ and $\tilde{M} = P^TQ_M$ are computed offline with a computational cost of $2m \times (\sigma + 1)$ scalar products. The reduced order matrix $\tilde{A}$ is then inexpensively computed for any time-step as $\tilde{A} = \tilde{H} + \tilde{M}/\Delta t$.

This procedure requires the solution of the full system model at the times of the snapshots, entailing a possibly large computational effort for the offline phase. To reduce this cost, in many applications the snapshots are collected only at the beginning of the temporal simulation, e.g., as in Vermeulen et al. [26], but this approach does not guarantee that the snapshots are representative of the full model solution for the rest of the simulation. Siade et al. [13] proposes an optimal distribution of the snapshots times during the simulation, thus avoiding the computation of the full system model at each time step. In this work POD is based on the set of all snapshots evaluated by the full-model during the transient simulation over the entire time interval. This approach is obviously computationally expensive but produces an optimal m-th order reduced model that is used as benchmark preconditioner. In fact, given the optimality of the POD basis [9], the ROM solution obtained using POD on the full set of snapshots Y minimizes the error $\sum_{j=1}^{z} \|\mathbf{y}^j - \tilde{\mathbf{y}}^j\|$.

4.2.2. Updated basis vectors. The classical collection of the snapshots in the offline procedure is useful when the final goal is to replace the full system model with the reduced order model. However, since in our case the reduced order model is used for preconditioning purposes, the FSM solution is available at every time step. This suggests that the space generated by the basis vectors, $\mathcal{P}$, may be augmented at every time step with the newly available FSM solution. At the time t_j, the new FSM solution $\mathbf{y}^j$ is added to the previously computed basis vectors $P^{j-1} = [\mathbf{p}_1, \ldots, \mathbf{p}_{j-1}]$ and orthogonalization is achieved through the modified Gram-Schmidt method. In this procedure the principal components are computed online, with a computational cost of j scalar products at every time step t_j. Since at every time step we only add a new basis vector, the reduced order matrix $\tilde{H}^j = P^{j,T}HP^j$ (or $\tilde{M}^j = P^{j,T}MP^j$) is simply computed inserting the vector $P^{j,T}H\mathbf{p}_j$

Prepared using nmeauth.cls http://mc.manuscriptcentral.com/nme DOI: 10.1002/nme

(or $P^{j,T} M \mathbf{p}_j$) and its transpose as the j-th column and row, respectively, of the matrix $\tilde{H}^{j-1}$ (or $\tilde{M}^{j-1}$).

To control the maximum number of basis vectors employed in the reduction, m, when $j > m$ the orthonormalization procedure is applied only to the m basis vectors $\mathbf{p}_{j-m+1}, \ldots, \mathbf{p}_j$. The reduced order matrix $\tilde{H}^j$ (or $\tilde{M}^j$) is obtained discarding the first row and column of matrix $\tilde{H}^{j-1}$ (or $\tilde{M}^{j-1}$) and then inserting $P^{j,T} H \mathbf{p}_j$ (or $P^{j,T} M \mathbf{p}_j$) and its transpose as the $m-th$ column and row. The computation of $Q_H^j = HP^j$ and $Q_M^j = MP^j$ proceeds in the same way.

Algorithm 3 Updated basis vectors

1. Set m and S
2. $P^0 = [\,]$, $\tilde{H}^0 = [\,]$, $\tilde{M}^0 = [\,]$

for $j = 1, \ldots, m$ **do**

3. Compute $\mathbf{y}^j$ with PCG
4. $\mathbf{p}_j = \mathbf{y}^j$ and orthonormalize with respect to P^{j-1}
5. $P^j = [P^{j-1}, \mathbf{p}_j]$
6. $\tilde{H}^j_{1:j-1,1:j-1} = \tilde{H}^{j-1} \quad ; \quad \tilde{H}^j_{:,j} = P^{j,T} H \mathbf{p}_j \quad ; \quad \tilde{H}^j_{j,:} = \tilde{H}^{j,T}_{:,j}$
7. $\tilde{M}^j_{1:j-1,1:j-1} = \tilde{M}^{j-1} \quad ; \quad \tilde{M}^j_{:,j} = P^{j,T} M \mathbf{p}_j \quad ; \quad \tilde{M}^j_{j,:} = \tilde{M}^{j,T}_{:,j}$

end for

for $j = m+1, \ldots$ **do**

8. Compute $\mathbf{y}^j$ with PCG
9. $\mathbf{p}_j = \mathbf{y}^j$ and orthonormalize with respect to $[\mathbf{p}_{j-m+1}, \ldots, \mathbf{p}_{j-1}]$
10. $P^j = [\mathbf{p}_{j-m}, \ldots, \mathbf{p}_j]$
11. $\tilde{H}^j_{1:m-1,1:m-1} = \tilde{H}^{j-1}_{2:m,2:m} \quad ; \quad \tilde{H}^j_{:,m} = P^{j,T} H \mathbf{p}_j \quad ; \quad \tilde{H}^j_{m,:} = \tilde{H}^{j,T}_{:,m}$
12. $\tilde{M}^j_{1:m-1,1:m-1} = \tilde{M}^{j-1}_{2:m,2:m} \quad ; \quad \tilde{M}^j_{:,m} = P^{j,T} M \mathbf{p}_j \quad ; \quad \tilde{M}^j_{m,:} = \tilde{M}^{j,T}_{:,m}$

end for

Algorithm 3 summarizes the main operations necessary to compute the updated principal components. The reduced order model constructed with this update procedure is based only on the m full model solutions computed at the previous time steps. It is expected that these basis functions are better suited to extrapolate the solution at t_j with respect to the global basis vectors computed with the principal components evaluated on the whole set of snapshots.

In what follows we use a fixed maximum number m of basis vectors so as to assess its influence on the solver efficiency. In real applications, however, m may be adaptively changed in time according to the transient solution. Markoninovic̀ et al. [20] use a similar algorithm for updating the basis vectors of the reduced order model, which is then used to evaluate the initial guess for PCG.

5. NUMERICAL RESULTS AND DISCUSSION

As test problem, we consider the transient diffusion taking place in a 3D domain with horizontal dimension 100×100 m^2 and 20 m depth. The synthetic domain is characterized by the two materials M1 and M2 shown in Figure 1a. Material M1 is located at the top (from 0 to 5 m depth) and at the bottom (from 15 to 20 m depth) of the domain. The diffusion coefficients in material M1 are $D_x = D_y = 5.0$ m/d, $D_z = 10^{-4}$ m/d. Material M2 constitutes the central layer of the domain (from 5 to 15 m depth) and has the following properties: $D_x = 10^{-7}$ m/d, $D_y = 10^3$ m/d, $D_z = 1.0$ m/d. The domain is characterized by a homogeneous storage coefficient ($S_s = 0.1$ m^{-1}). The extreme values of the anisotropy are used to yield a severe ill-conditioning in the linear systems that will be used to test the behavior of the proposed preconditioner.

A homogeneous Dirichlet condition is prescribed at the boundaries $x = 0$ m and $x = 100$ m. No-flux condition is assumed on the other 4 boundaries. A sink is located at the center of the domain ($x = y = 50$ m, $z = -10$ m) with a withdrawal rate of 500 m^3/d. With these conditions the system reaches the steady state in about 10^4 days.

To explore the performance of the ROM preconditioner with respect to the system size, we consider four test cases (TC1-TC4) with different spatial discretizations of the domain. The mesh of the first test case (TC1) has $n = 1880$ nodes and $r = 8520$ tetrahedral elements (Figure 1a). The associated system matrix has 13149 non-zero elements with an average $\sigma = 12.98$ non-zero entries per row. The second, third and fourth test cases (TC2-TC4) are obtained by regularly refining the grid used in TC1, with the total number of nodes that increases up to $755,073$ in TC4.

A final test case (TC5) considers the same spatial discretization as TC4 and a random distribution of the diffusion coefficient D (Figure 1b). D_x is a second-order random field with lognormal distribution ($Y = \ln(D_x)$, mean $\mu_Y = -3.45$ log(m/d), standard deviation $sd_Y = 1.53$ log(m/d)) and exponential covariance function (correlation length $\lambda_x = \lambda_y = \lambda_z = 5$ m). The diffusion in each grid element is horizontally isotropic ($D_y = D_x$) and anisotropic along the vertical direction ($D_z = D_x/50$). The main properties of the five test problems are summarized in Table II.

Prepared using nmeauth.cls http://mc.manuscriptcentral.com/nme DOI: 10.1002/nme

The numerical results are obtained using IC0 as support matrix for the construction of both the ROM and SROM preconditioners. Moreover, we consider two sets of possible basis vectors, i.e., either the vectors arising from the PCA on the snapshots or those obtained with Algorithm 3. Quite intuitively, the use of a different support matrix does not qualitatively modify the results that will be presented in the sequel.

5.1. Reduced Order Model

The reduced order model constructed performing the PCA on the set of snapshots can significantly reduce the computational cost of the full system model, as it considers a problem with size $m << n$. However, the use of few basis functions may lead to large errors in the system resolution (see, e.g., Pasetto et al. [17]). Recent techniques have been advanced to assess how many basis vectors should be considered in the reduced model by relating the norm of the residual to the associated error [14, 16].

In our application of the reduction strategy in the preconditioner for the PCG, first we have to assess how many basis vectors are needed to obtain a prescribed accuracy. Figure 2 shows the Euclidean norm of the errors between the ROM and the FSM solutions at different times for $m = \{1, 2, 5, 10, 20, 30\}$. As expected, the error decreases when we increase the number of basis vectors. Moreover, the ROM is particularly accurate when the system approaches the steady state, i.e., for large values of time, independently of the number of basis vectors. This means that the steady state solution is well captured also with a small value of m. However, at the beginning of the transient phase the ROM error does not decrease when more than 20 basis vectors are employed, suggesting that further improvements the ROM accuracy are difficult to obtain.

5.2. Condition number and PCG convergence

Let us investigate the effectiveness of the ROM preconditioner by analyzing the condition number of the preconditioned matrix. Figure 3 compares the condition numbers of $K^{-1}A$, using the IC0 and the SROM preconditioners with $m = \{1, 10, 30\}$. For the sake of simplicity and with no loss of

Prepared using nmeauth.cls http://mc.manuscriptcentral.com/nme DOI: 10.1002/nme

generality, the results are presented only for the smallest test case TC1. Note that the ill-conditioning is more severe for larger values of the time step size, which in our simulations is increased geometrically as time progresses. The SROM preconditioner reduces the condition number more than IC0 using both the PCA of snapshots and the updated basis vectors, and especially for large time values. The condition number of SROM with m= 30 is more than two orders of magnitude smaller than the condition number obtained with IC0. In Figure 3a the projection matrix P corresponds to the principal components of the snapshots collected at each time step. As a consequence, the condition number is only slightly dependent on the time step size when using a large number of basis vectors ($m = \{10, 30\}$), meaning that the basis vectors display a global optimality. In Figure 3b the projection matrix P is updated at each time step as described in Algorithm 3. In this case, the reduction on the condition number is less effective than in Figure 3a at the beginning of the simulation, i.e., when only few basis vectors are available. However, after few time steps the updated basis vectors seem to be much more suited for the construction of the SROM preconditioner since they provide a significant reduction of the condition number. Notice that in this case increasing m over 10 is still effective, while it is not with the PCA.

To explore the efficiency of the proposed algorithm in the PCG convergence, we stored the system matrix and the preconditioner at the 30th time step of the simulation (time $t \approx 1500$ d), i.e., when 30 basis vectors have been collected for the construction of the SROM preconditioner with $m = 30$. Figure 4 depicts the total set of the eigenvalues of matrices A and $K^{-1}A$ using IC0 and the SROM preconditioners. As expected, the application of the preconditioner has a significant impact in the reduction of the largest eigenvalues of A, and thus in the reduction of the conditioning number. The main difference between the IC0 and SROM preconditioners is the magnitude of the smallest eigenvalues, that are larger for the SROM preconditioning matrix than for IC0, resulting in an overall eigenvalue distribution closer to 1. This clustering effect, which is already present with $m = 1$, is more pronounced when a larger number of basis vectors are employed in the construction of the SROM preconditioner (e.g. $m = 30$), and when these vectors are updated with Algorithm 3 (*upd*).

Prepared using nmeauth.cls http://mc.manuscriptcentral.com/nme DOI: 10.1002/nme

This property of the SROM preconditioner is fundamental for the fast convergence of the PCG, as highlighted in Figures 5 and 6.

Figures 5 shows the PCG convergence results in terms of error and residual of a linear system with an assigned right hand side (obtained from the IC0 simulation at the 30th time-step). The small dimension of the model in TC1 allowed us to solve the system by a direct method at the machine precision, thus computing the errors at each PCG iteration. In Figure 5 we can see that IC0 requires several iterations before decreasing the residual norm and achieving asymptotic conditions. Both panels (a) and (b) show that the PCG convergence with the SROM preconditioner is significantly faster than IC0, especially in the initial iterations. For example, after about 15 iterations the error norm using the SROM preconditioner with $m = 30$ is about 10^{-9} while with IC0 it is several orders of magnitude larger ($\approx 10^{-3}$). This beneficial behavior is due to the increase of the smallest eigenvalues of the preconditioned matrix with respect to IC0 (see Figure 3). We can also notice that asymptotically the convergence rate is roughly similar for the IC0 and SROM preconditioners. Similar conclusions are obtained analyzing Figure 6, which shows the PCG convergence profiles of the residual norm at the 30th time step of TC1, TC2 and TC3.

It is well known that the PCG convergence rate is faster when the solution error is orthogonal to a large number of eigenvectors of the preconditioned system matrix. To further investigate the reasons of the SROM behavior, in the following we consider the scalar products among the eigenvectors of the preconditioned matrix and the iteration error. We indicate with $r_{k,j}$ the scalar product between the j-th eigenvector associated to the eigenvalue λ_j and the error at iteration k. Figure 7 shows the values of $r_{k,j}$ for the first 40 iterations of the PCG using IC0 and SROM preconditioner with $m = 30$. In these iterations the PCG preconditioned with IC0 seeks a solution whose error is mainly orthogonal to the eigenvectors associated with the largest eigenvalues ($j = 1878, 1879, 1880$). Instead, the error produced by the use of the SROM preconditioner seems to rapidly get orthogonal to the complete set of eigenvectors, with both large and small eigenvalues.

Figure 8 shows the number of eigenvectors that are orthogonal to the error at each iteration. With IC0 the initial error is orthogonal only to a small number of eigenvectors. This number increases

Prepared using nmeauth.cls http://mc.manuscriptcentral.com/nme DOI: 10.1002/nme

significantly only after 90 iterations, i.e., when the PCG reaches the asymptotic convergence. Slightly larger numbers are recorded when the SROM preconditioner with $m = 1$ is employed. When using a large number of basis vectors ($m = 30$) or the updated basis vectors (Algorithm 3) the PCG errors result immediately orthogonal to a sizeable number of eigenvectors of the preconditioned matrix.

5.3. Computational cost

To generalize previous results to the complete transient simulation, Figure 9 depicts the number of PCG iterations necessary to achieve convergence at every time step in the TC2, TC3 and TC4 test cases. The results are presented for an exit PCG tolerance $\tau = 10^{-6}$ on the relative residual norm. For the sake of brevity, in the following only the results associated to the updated basis vectors are presented, as this strategy appears to be more favorable and easier to be applied in practice since it does not require an expensive offline stage.

Figure 9 shows that the convergence rate of the SROM preconditioner outperforms IC0 in all test cases. Using few basis functions ($m = \{1, 2\}$) the number of iterations of SROM is up to 2/3 that of IC0, while with a large number of basis functions ($m = \{20, 30\}$) this ratio becomes 1/3 in TC3 and 1/8 in TC4. This suggests that the SROM preconditioner reduces the number of PCG iterations also employing a small number of basis vectors ($m = \{1, 2\}$), i.e., at a relatively marginal computational cost.

As discussed in Section 4.1, the computational cost of the application of the SROM preconditioner increases with the number of basis vectors. We indicate with n^{SROM} and n^{IC0} the total number of scalar products per time step using the SROM and the IC0 preconditioners, respectively. Figure 10 shows the relative cost c_r of SROM with respect to IC0, i.e., $c_r = (n^{SROM} - n^{IC0})/n^{IC0}$. A negative value for c_r means that the PCG preconditioned with SROM is less expensive than with IC0. The results obtained show that for the majority of time steps the SROM preconditioners are computationally advantageous with respect to IC0. The SROM performance is slightly different in test cases TC2, TC3 and TC4 and depends upon the number of basis vectors employed in the

Prepared using nmeauth.cls http://mc.manuscriptcentral.com/nme DOI: 10.1002/nme

projection. In TC2 the SROM preconditioner appears to be more competitive with few basis vectors, reducing the number of scalar products up to 30% for several time steps. In TC3, the use of the SROM preconditioner reduces the computational cost up to 50% with $m = 10$ and 45% with $m = 1$. In TC4 the best preconditioners have $m = 30$ basis vectors with a maximum gain of 60% at time $t = 100$ d. The proposed algorithm proved more expensive than the IC0 approach at few time steps only, with the maximum loss of 20% computed with $m = 30$ in TC2. Finally note that the SROM preconditioner with $m = 1$ consistently outperformed IC0 in all the test cases.

Tables III, IV, and V summarize the results of the numerical simulation showing the total relative cost C of the new preconditioners with respect to IC0 in the complete transient simulation, i.e.:

$$C = \frac{\sum (n^R - n^{IC0})}{\sum n^{IC0}}. \tag{29}$$

where n^R is either n^{SROM} or n^{ROM}. Table III shows the total relative costs of the SROM preconditioners for test cases TC2, TC3 and TC4 and for two values of the tolerance on the residual norm ($\tau = 10^{-6}$ and $\tau = 10^{-10}$). The results do not show a clear trend with the increase of m and the size of the domain. This is basically due to rounding errors, that are also the main cause for the oscillatory behavior with time shown in Figure 10. We can see that the largest reductions of the computational costs are achieved in TC4 ($m = 20$ and 30 and $\tau = 10^{-6}$) with advantages higher than 40%. The SROM preconditioners always outperforms IC0 with the tolerance $\tau = 10^{-6}$, while this is not the case with tolerance $\tau = 10^{-10}$, that is actually rarely required in practical applications. This is a direct consequence of the particular convergence profiles associated to the SROM preconditioners, as shown in Figure 6. However, note that the results obtained with $m = 1$ and 2 are always reducing the costs of IC0 in all the explored test cases.

Table IV is the same as Table III but for the non-symmetric ROM preconditioner. At a first glance we can see that the savings obtained with the ROM preconditioner are slightly larger that those of SROM. This is due to the fact that the per-iteration cost of PCG with the ROM preconditioner is m scalar products lower than the analogous cost of PCG with SROM. Although

Prepared using nmeauth.cls http://mc.manuscriptcentral.com/nme DOI: 10.1002/nme

the ROM preconditioner is non-symmetric, in these applications it can be more convenient from a computational point of view.

Finally, Table V reports the results obtained in the fully heterogeneous scenario TC5 with tolerance $\tau = 10^{-6}$. In this case the new preconditioner is competitive with IC0 when using few basis vectors. We can see that both the ROM and SROM preconditioners improve the computational costs by about 10% with $m = 1$ and 5% with $m = 2$. The ROM preconditioner seems to be slightly more efficient than the SROM for $m = 1$ and $m = 2$. Note that, convergence of ROM preconditioned PCG is not achieved for larger values of m. This is due to the fact that the PCG with this non-symmetric preconditioner is more prone to errors related to numerical round-off and the global optimality is lost. Table V also shows the computational costs associated to the ROM and SROM preconditioners when using the Jacobi preconditioner as support matrix ($S = \text{diag}(A)^{-1}$). We can see that this procedure is never convenient with respect to IC0, increasing the costs of about 50% with $m = 1$. However, this is anyway a remarkable improvement with respect to the cost of the Jacobi preconditioner alone, which for this application is ten times larger than IC0. Roughly speaking, the SROM global cost with $S = \text{diag}(A)^{-1}$ is about 1/7 the cost of the Jacobi preconditioner alone. Remembering that the application of the Jacobi preconditioner can be efficiently parallelized while IC0 cannot, the combination of the SROM with Jacobi preconditioner and parallel computing represents a potentially promising alternative to IC0 to obtain a fast solution to large-size SPD linear systems. Similarly, an almost perfectly parallel implementation is also expected when an approximate inverse is used as support matrix, e.g., as those proposed in [33, 34, 35] for SPD linear systems.

6. CONCLUSIONS

The present work proposes a new class of preconditioners based on the POD technique for the efficient PCG solution of SPD linear systems arising from the numerical discretization of parabolic PDEs. Starting with a suitable support matrix (e.g., Jacobi preconditioner or IC0), the

Prepared using nmeauth.cls http://mc.manuscriptcentral.com/nme DOI: 10.1002/nme

new preconditioning strategy uses the forward and backward projection steps of POD in each PCG iteration, with the goal of improving the spectral properties of the preconditioned system matrix. The new preconditioner has a non symmetric (ROM preconditioner (14)) and a symmetric version (SROM preconditioner (28)). Although under unrestrictive assumptions both versions guarantee convergence of the underlying PCG method, iterations employing the ROM preconditioner are less expensive than those employing SROM, but are more sensitive to the propagation of round-off errors. The choice of the projection space for the POD reduction and its dimension (m) are of crucial importance for the methodology. Two possible strategies are considered for the construction of suitable spaces: the snapshot technique, which is largely used with POD and computes the principal components of the complete collection of model solutions, and an updating algorithm (Algorithm 3), which continuously updates the projection space with the model solutions computed in the last few time steps. The performance of the new preconditioners are compared with respect to the performance of the support matrix alone, IC0 in this case, in terms of reduction of condition number of the preconditioned matrix, total number of PCG iterations to achieve convergence, and computational cost. Five test cases with different grid sizes and parameter distributions are considered to verify the performance and robustness of the developed technique at varying degree of ill-conditioning.

Numerical results prompt the following major conclusions.

- The condition number of the SROM preconditioned matrix is significantly reduced with respect to the condition number of the IC0 preconditioned matrix, both for the optimal reduced space preconditioner calculated using snapshots collected during the entire transient simulation or when POD is based on the full model solutions calculated at the m previous time steps. This entails a faster orthogonalization of the PCG error with respect the eigenvectors associated to the small eigenvalues of the preconditioned matrix, strongly improving the convergence of the PCG during the initial iterations.
- Although the number of PCG iterations decreases using ROM and SROM preconditioners with respect to IC0 in all the considered scenarios, the computational cost of the new

Prepared using nmeauth.cls http://mc.manuscriptcentral.com/nme DOI: 10.1002/nme

preconditioner increases with m and a trade-off is necessary. The application of the SROM and ROM preconditioners is therefore suggested with small values of m. In our test cases the performance was optimized for m =1 or 2, where the system solver was always more efficient than IC0-preconditioned conjugate gradient.

- The proposed preconditioner is particularly suited for parallel computing, since the forward and backward projections only involve matrix-vector products. The application of the preconditioner becomes totally parallel when also the support matrix S can be applied directly, e.g. when S is the Jacobi preconditioner. This can be highly effective in reducing the computational burden of transient PDEs. In fact, the cost of the numerical solution of the linear system on a serial computer for a spatially heterogeneous diffusion coefficient (TC5) using the SROM preconditioner with $m = 1$ or 2 together with the Jacobi support matrix is comparable to the analogous cost of the IC0-preconditioned conjugate gradient. Important improvements can be expected in a parallel environment.

REFERENCES

REFERENCES

1. Ferronato M. Preconditioning for sparse linear systems at the dawn of the 21st century: history, current developments, and future perspective. *ISRN Appl. Math.* 2012; **2012**:Article ID 127 647, doi:10.5402/2012/127647.
2. Varga RS. Factorization and normalized iterative methods. *Boundary Problems in Differential Equations*, vol. 26, Langer RE (ed.). University of Wisconsin Press: Madison, Wis, USA, 1960; 121–142.
3. Meijerink JA, vad der Vorst HA. An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. *Math. Comp.* 1977; **31**(137):148–162, doi:10.1090/S0025-5718-1977-0438681-4.
4. Kershaw DS. The incomplete Cholesky-conjugate gradient method for the iterative solution of systems of linear equations. *J. Comp. Phys.* 1978; **26**(1):43–65, doi:10.1016/0021-9991(78)90098-0.
5. Parter SV, Rothma EE. Preconditioning Legendre spectral collocation approximations to elliptic problems. *SIAM J. Num. Anal.* 1995; **32**(2):333–385, doi:10.1137/0732015.

6. Mousseau VA, Knoll DA, Rider WJ. Physics-based preconditioning and the Newton-Krylov method for non-equilibrium radiation diffusion. *J. Comp. Phys.* 2000; **160**(2):743–765, doi:10.1006/jcph.2000.6488.
7. Kim SD. Piecewise bilinear preconditioning of high-order finite element methods. *Electronic Trans. Num. Anal.* 2007; **26**:228–242.
8. Kwon JK, Ryu S, Kim P, Kim SD. Finite element preconditioning on spectral element discretizations for coupled elliptic equations. *J. Appl. Math.* 2012; **2012**:1–16, doi:10.1155/2012/245051.
9. Kunisch K, Volkwein S. Galerkin proper orthogonal decomposition methods for parabolic problems. *Numer. Math.* 2001; **90**(1):117–148, doi:10.1007/s002110100282.
10. Kunisch K, Volkwein S. Galerkin proper orthogonal decomposition methods for a general equation in fluid dynamics. *SIAM J. Num. Anal.* 2002; **40**:492–515, doi:10.1137/S0036142900382612.
11. Haasdonk B, Ohlberger M. Efficient reduced models and a-posteriori error estimation for parametrized dynamical systems by offline/online decomposition. *Math. Comp. Model. Dyn.* 2011; **17**(2):145–161, doi:10.1080/13873954.2010.514703.
12. Hasenauer J, Löhning M, Khammash M, Allgöwer F. Dynamical optimization using reduced order models: a method to guarantee performance. *J. Process Contr.* 2012; **22**(8):1490–1501, doi:10.1016/j.jprocont.2012.01.017.
13. Siade AJ, Putti M, Yeh WWG. Snapshot selection for groundwater model reduction using proper orthogonal decomposition. *Water Resour. Res.* 2010; **46**:W08 539, doi:10.1029/2009WR008792.
14. Grepl MA, Patera AT. A posteriori error bounds for reduced-basis approximations of parametrized parabolic partial differential equations. *ESAIM-Math. Model. Num.* 2005; **39**(1):157–181, doi:10.1051/m2an:2005006.
15. Siade AJ, Putti M, Yeh WWG. Reduced order parameter estimation using quasilinearization and quadratic programming. *Water Resour. Res.* 2012; **48**:W06 502, doi:10.1029/2011WR011471.
16. Pasetto D, Putti M, Yeh WWG. A reduced order model for groundwater flow equation with random hydraulic conductivity: Application to Monte Carlo methods. *Water Resour. Res.* 2013; **49**:1–49, doi:10.1003/wrcr.20136.
17. Pasetto D, Guadagnini A, Putti M. A reduced-order model for Monte Carlo simulations of stochastic groudwater flow. *Comput. Geosci.* 2014; **18**:157–169, doi:10.1007/s10596-013-9389-4.
18. Boyce SE, Yeh WWG. Parameter-independent model reduction of transient groundwater flow models: Application to inverse problems. *Adv. Water Resour.* 2014; **69**:168–180, doi:10.1016/j.advwatres.2014.04.009.
19. Ushijima TT, Yeh WWG. Experimental design for estimating unknown groundwater pumping using genetic algorithm and reduced order model. *Water Resour. Res.* 2013; **49**(10):6688–6699, doi:10.1002/wrcr.20513.
20. Markovinovič R, Jansen JD. Accelerating iterative solution methods using reduced-order models as solution predictors. *Int. J. Numer. Methods Engrg.* 2006; **68**(5):525–541, doi:10.1002/nme.1721.
21. Astrid P, Papaioannou G, Vink JC, Jansen JD. Pressure preconditioning using proper orthogonal decomposition. *SPE Reservoir Simulation Symposium, The Woodlands, Texas*, Society of Petroleum Engineers, 2011; SPE paper 141 992, doi:10.2118/141922-MS.
22. Jiang R. Pressure preconditioning using proper orthogonal decomposition. Master's Thesis, Stanford University 2013.

549 23. Brandt A, McCormick SF, Ruge JW. Algebraic multigrid (AMG) for sparse matrix equations. *Sparsity and Its*
550 *Applications*, Evans DJ (ed.). Cambridge University Press: Cambridge, MA, 1984; 257–284.
551 24. Ruge JW, Stüben K. Algebraic multigrid (AMG). *Multigrid Methods. Frontiers in Applied Mathematics*, vol. 3, F
552 MS (ed.). SIAM: Philadelphia, PA, 1987; 73–130.
553 25. Falgout RD, Vassilevski PS. On generalizing the algebraic multigrid framework. *SIAM J. Num. Anal.* 2004;
554 **42**:1669–1693, doi:10.1137/S0036142903429742.
555 26. Vermeulen PTM, Heemink AW, Stroet CBMT. Reduced models for linear groundwater flow models using empirical
556 orthogonal functions. *Adv. Water Resour.* 2004; **27**(1):57 – 69, doi:10.1016/j.advwatres.2003.09.008.
557 27. Quarteroni A, Rozza G. Numerical solution of parametrized Navier-Stokes equations by reduced basis methods.
558 *Num. Meth. PDE* 2007; **23**(4):923–948, doi:10.1002/num.20249.
559 28. Axelsson O. *Iterative Solution Methods*. Cambridge University Press: Cambridge, 1994.
560 29. Blaheta R. GPCG-generalized preconditioned CG method and its use with non-linear and non-symmetric
561 displacement decomposition preconditioners. *Numer. Lin. Alg. Appl.* 2002; **9**:527–550, doi:10.1002/nla.295.
562 30. Notay Y. Flexible conjugate gradients. *SIAM J. Sci. Comput.* 2000; **22**:1444–1460, doi:10.1137/
563 S1064827599362314.
564 31. Vassilevski PS. *Multilevel Block Factorization Preconditioners*. Springer: New York, NY, 2008.
565 32. Bouwmeester H, Dougherty A, Knyazev AV. Nonsymmetric preconditioning for the conjugate gradient and steepest
566 descent methods. *Procedia Comput. Sci.* 2015; **51**:276–285, doi:10.1016/j.procs.2015.05.241.
567 33. Ferronato M, Janna C, Pini G. Shifted FSAI preconditioners for the efficient parallel solution of non-linear
568 groundwater flow models. *Int. J. Numer. Methods Engrg.* 2012; **89**:1707–1719, doi:10.1002/nme.3309.
569 34. Janna C, Ferronato M, Gambolati G. The use of supernodes in factored sparse approximate inverse preconditioning.
570 *SIAM J. Sci. Comput.* 2015; **37**:C72–C94, doi:10.1137/140956026.
571 35. Janna C, Ferronato M, Sartoretto F, Gambolati G. FSAIPACK: A software package for high-perfomance factored
572 sparse approximate inverse preconditioning. *ACM Trans. Math. Softw.* 2015; **41**:Article 10, 26 pages, doi:
573 10.1145/2629475.

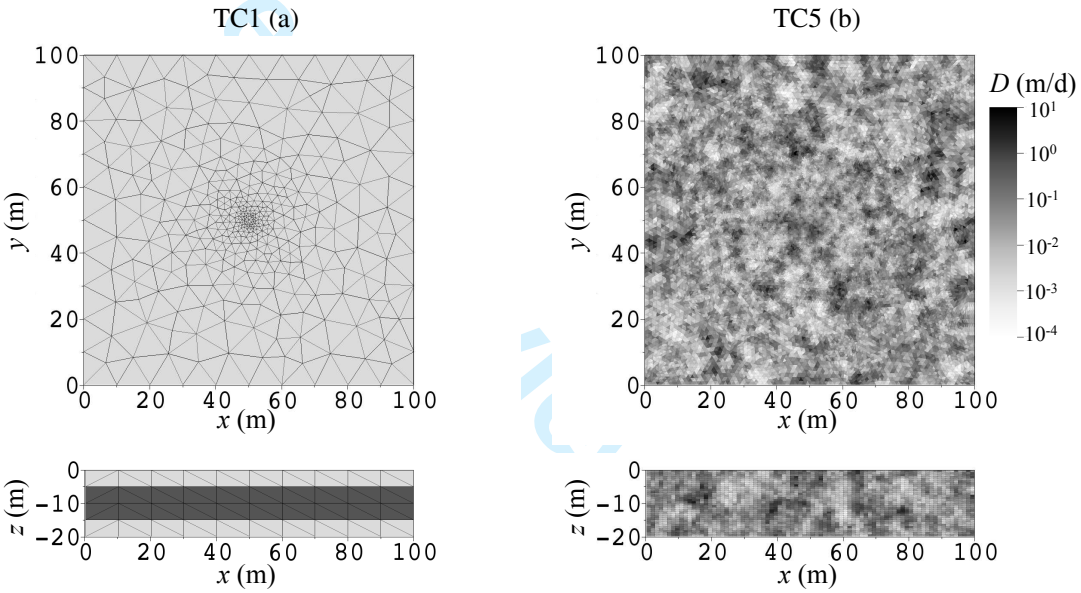

Figure 1. Horizontal and vertical sections of the spatial distribution of the diffusion coefficient D and of the discretizations of the domain used in TC1 (panel (a)) and TC5 (panel (b)). The dark and light grey colors in in TC1 (a) correspond to materials M1 and M2, respectively.

Prepared using nmeauth.cls
http://mc.manuscriptcentral.com/nme
DOI: 10.1002/nme

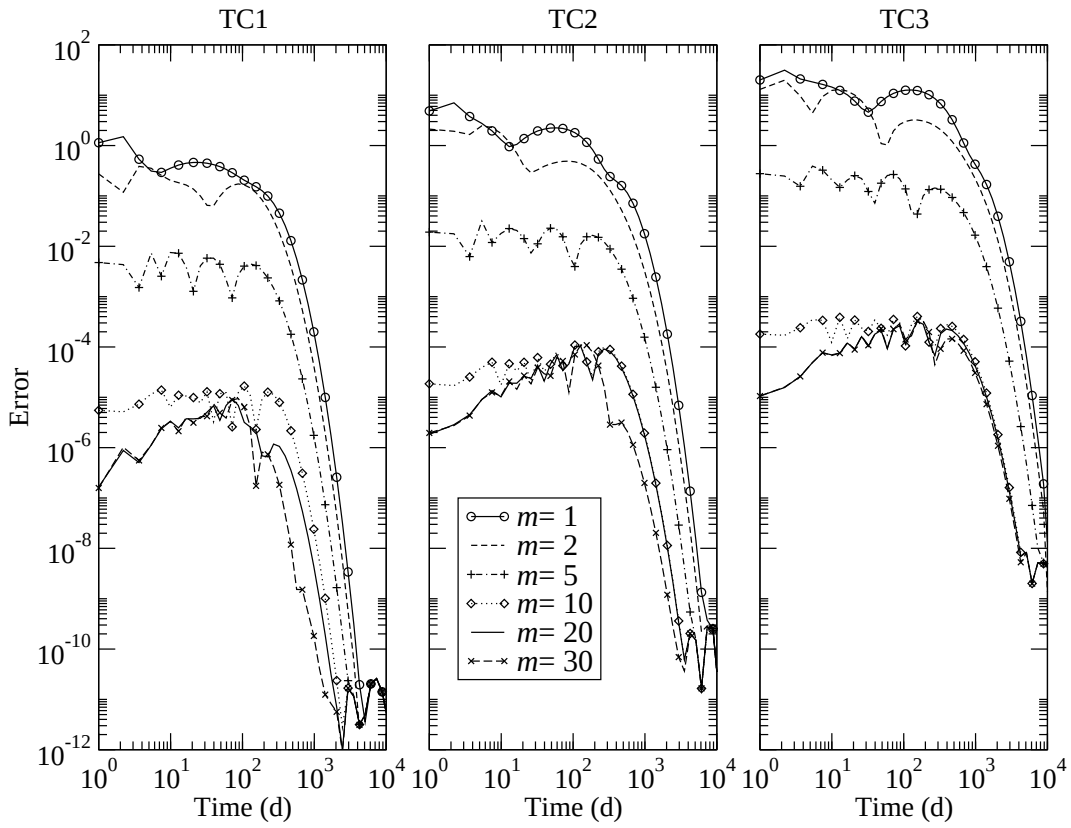

Figure 2. 2-Norm of the error between the ROM and the FSM solutions at every time step. Results for TC1, TC2 and TC3.

Prepared using nmeauth.cls
http://mc.manuscriptcentral.com/nme
DOI: 10.1002/nme

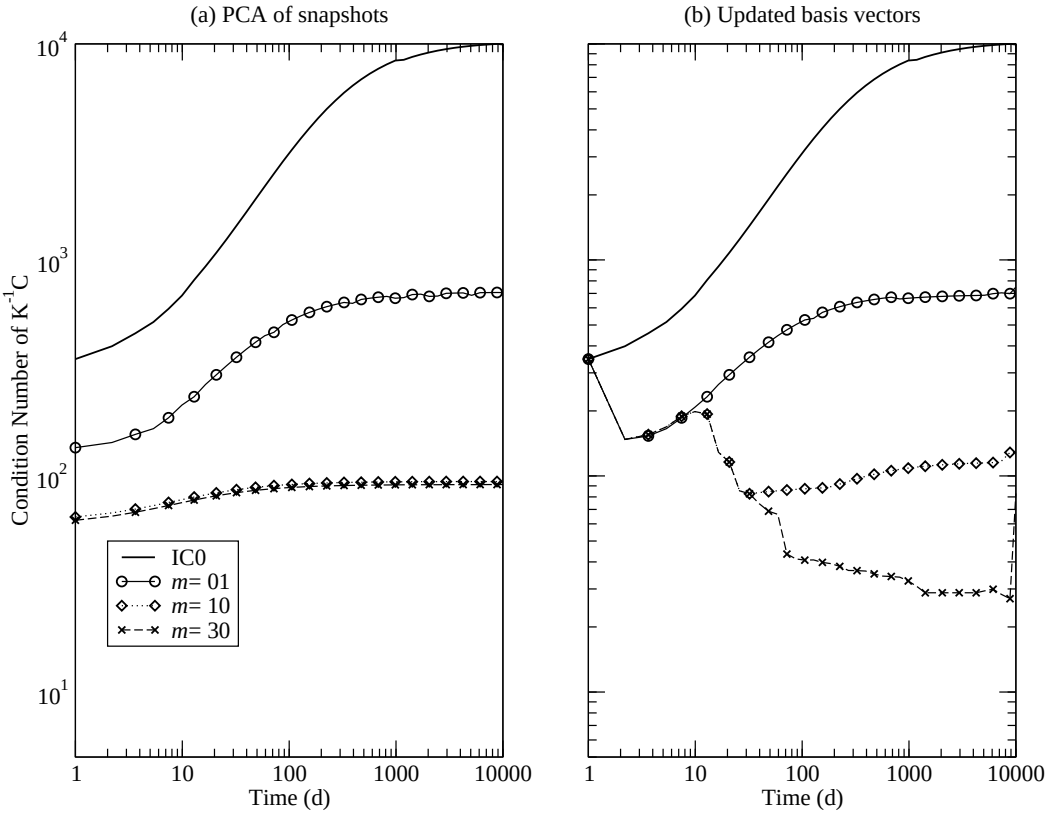

Figure 3. Comparison between the condition numbers associated to the preconditioned system matrix using the IC0 preconditioner and the SROM preconditioner. The basis vectors arise from the PCA on the snapshots (Panel a) and from Algorithm 3 (Panel b). Results for TC1.

Prepared using nmeauth.cls
http://mc.manuscriptcentral.com/nme
DOI: 10.1002/nme

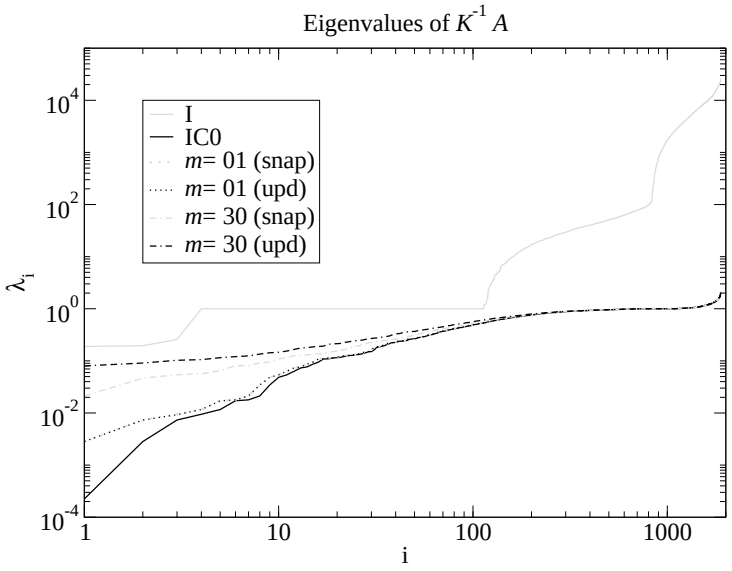

Figure 4. Comparison between the complete set of eigenvalues of the preconditioned system matrix $K^{-1}A$ using the identity matrix I, IC0 and SROM with different values of m as preconditioners. The basis vectors arise from the PCA on the snapshots (*snap*) or from Algorithm 3 (*upd*). Results for TC1 at the 30th time step.

Prepared using nmeauth.cls http://mc.manuscriptcentral.com/nme DOI: 10.1002/nme

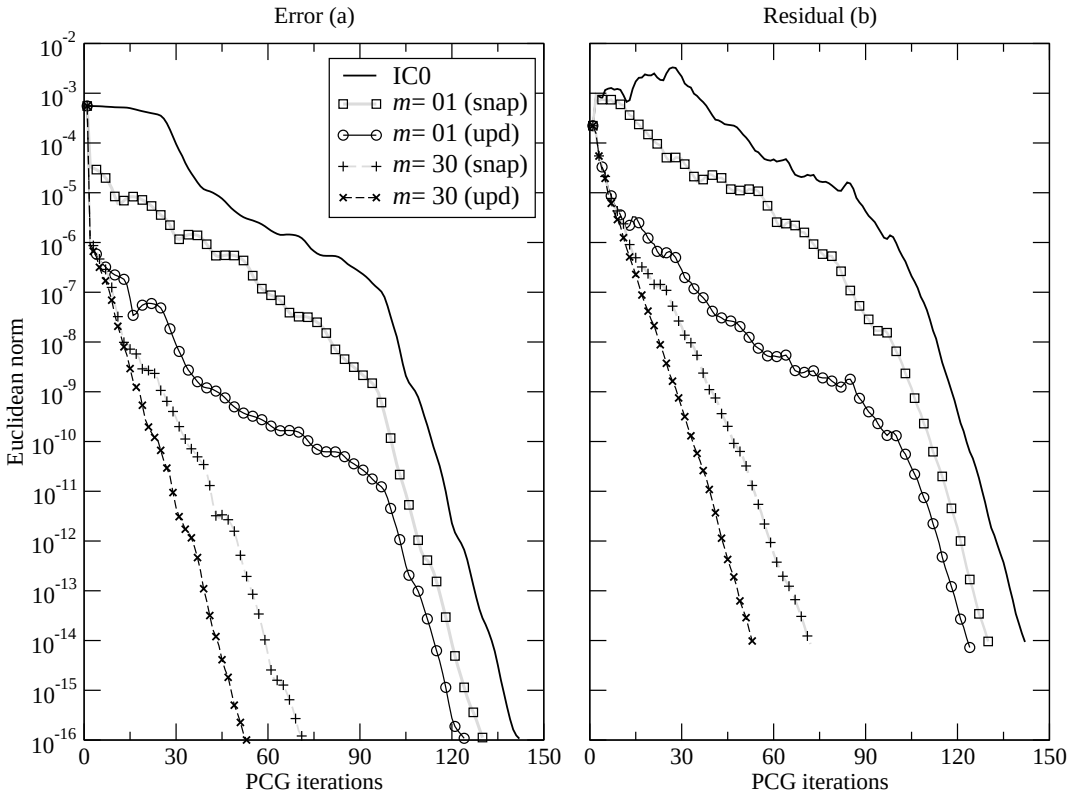

Figure 5. Convergence profile of the system error (left) and residual (right) 2-norms at the 30th time step for TC1. Comparison between the IC0 and the SROM preconditioners. The basis vectors arise from the PCA on the snapshots (*snap*) or from Algorithm 3 (*upd*).

Prepared using nmeauth.cls http://mc.manuscriptcentral.com/nme DOI: 10.1002/nme

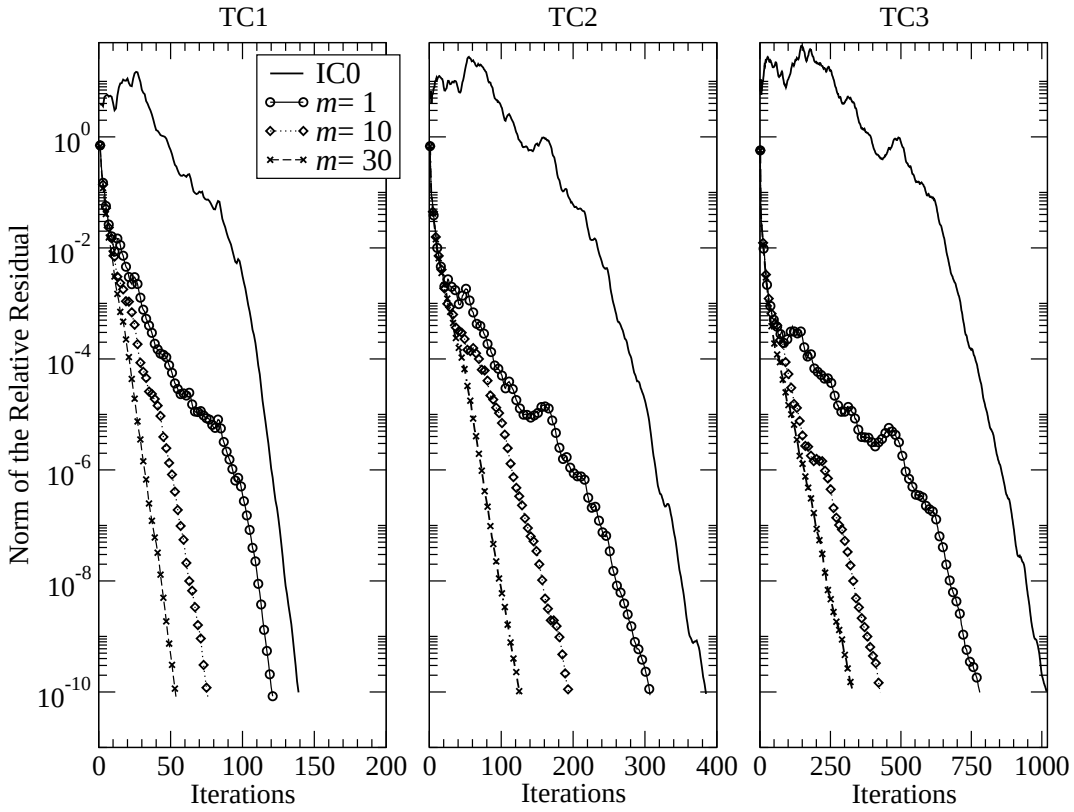

Figure 6. Convergence profile of the system residual at the 30th time step. Comparison between the IC0 and the SROM preconditioners. The basis vectors of the SROM preconditioner arise from Algorithm 3. Note that the x-scales are different in the three test cases.

Prepared using nmeauth.cls

DOI: 10.1002/nme

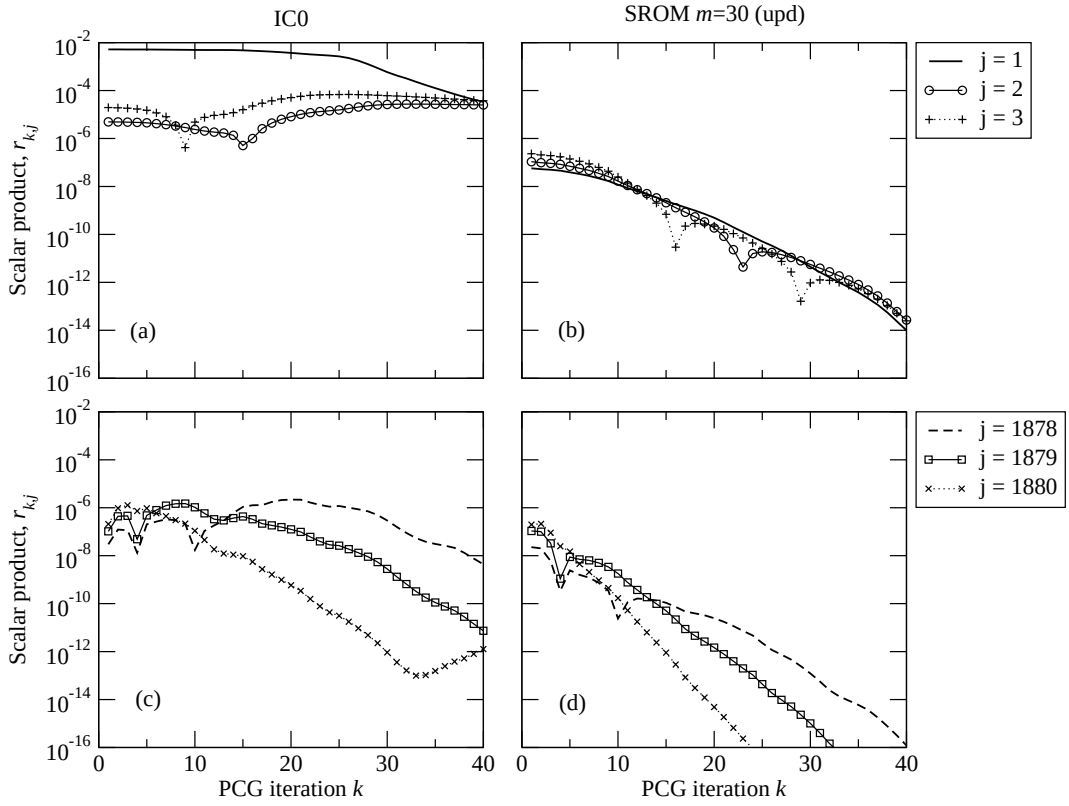

Figure 7. Values of $r_{k,j}$ for the first 40 PCG iterations using IC0 (panels (a) and (c)) and SROM preconditioner (panels (b) and (d)) with $m = 30$. Panels (a) and (b) show the scalar products associated to the three smallest eigenvalues of $K^{-1}A$. Panels (c) and (d) show the scalar products associated to the three largest eigenvalues of $K^{-1}A$.

Prepared using nmeauth.cls http://mc.manuscriptcentral.com/nme DOI: 10.1002/nme

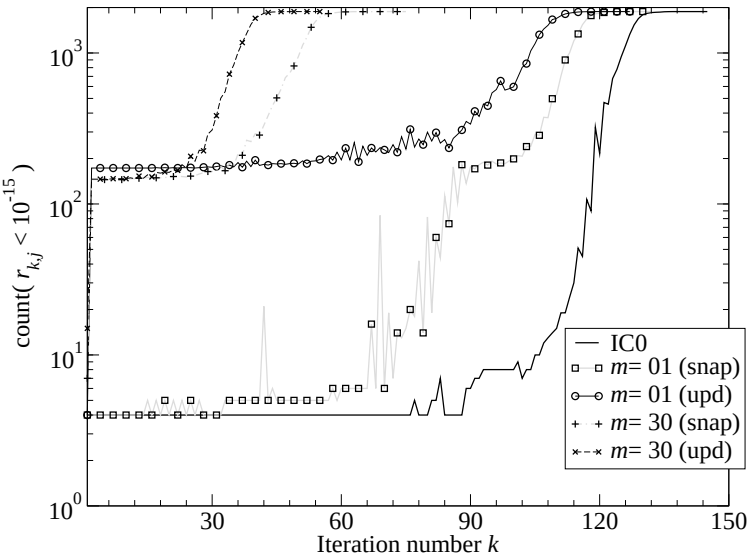

Figure 8. Number of scalar products $r_{k,j}$ which are smaller that 10^{-15} at each PCG iteration.

Prepared using nmeauth.cls http://mc.manuscriptcentral.com/nme DOI: 10.1002/nme

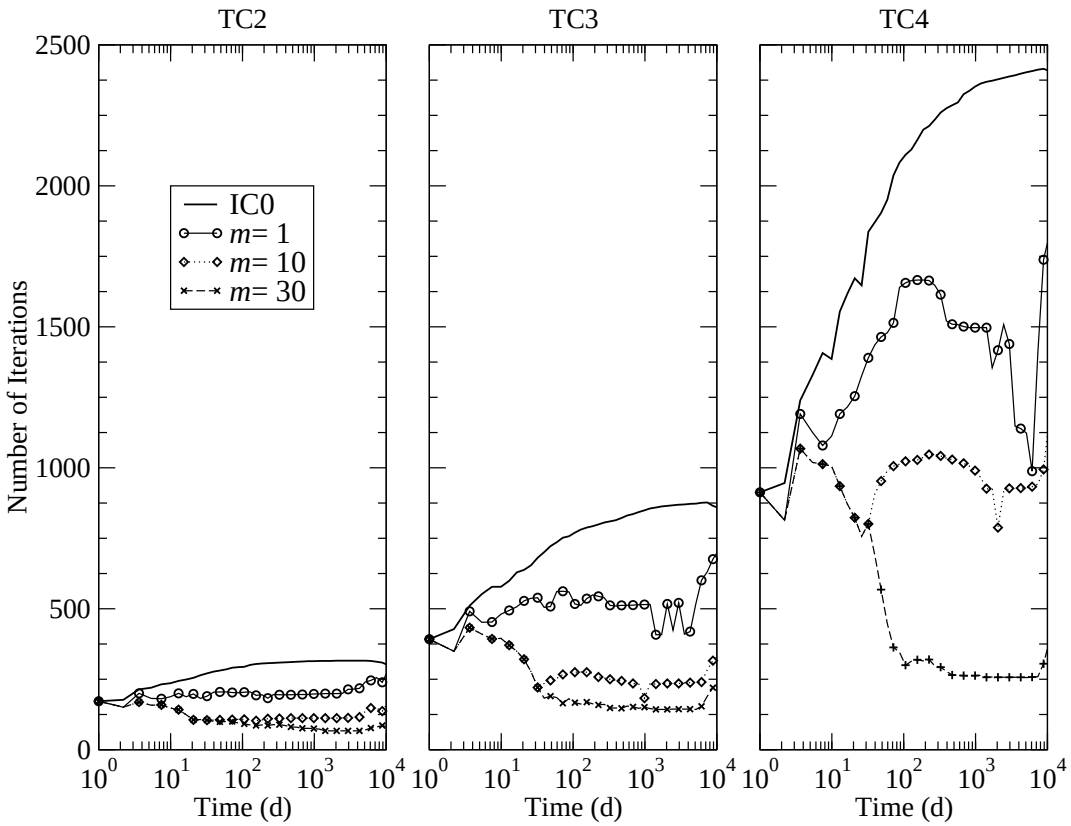

Figure 9. Number of iterations of the PCG at every time step using the IC0 and the SROM preconditioner. The basis vectors of the SROM preconditioner are generated by Algorithm 3.

Prepared using nmeauth.cls http://mc.manuscriptcentral.com/nme DOI: 10.1002/nme

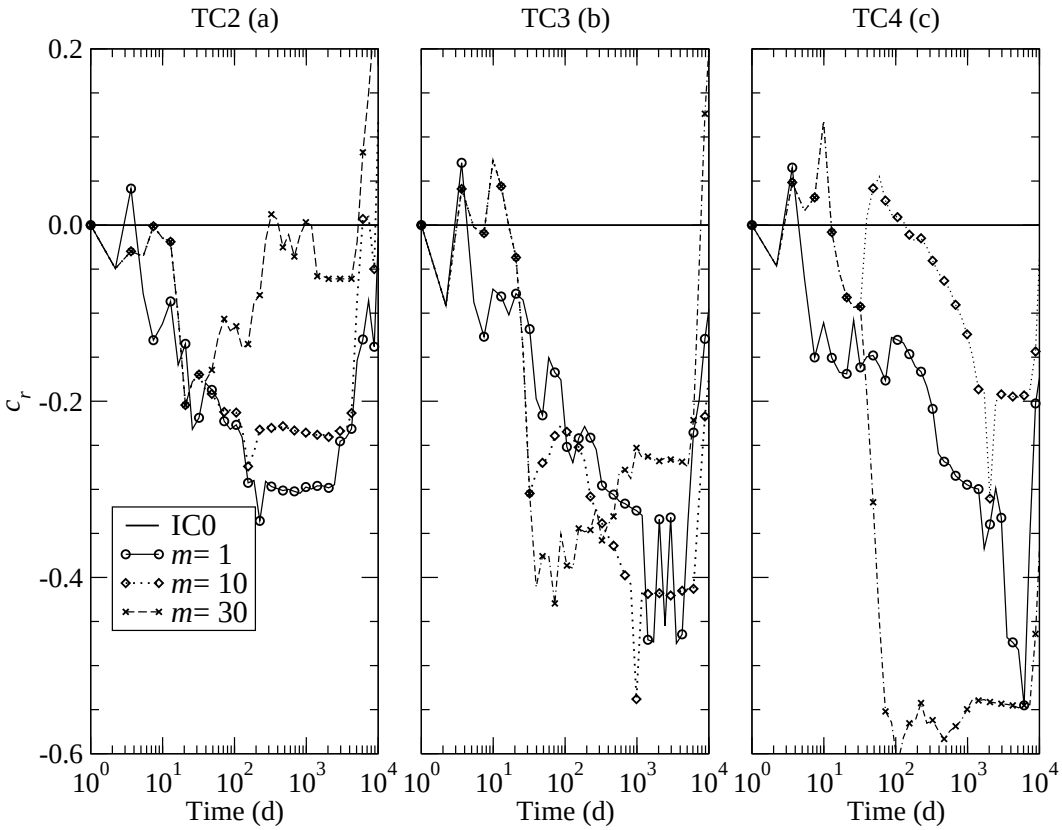

Figure 10. Relative cost c_r of the SROM preconditioner with respect the IC0 in terms of number of scalar products per time step. The basis vectors of the SROM preconditioner arise from Algorithm 3.

Prepared using nmeauth.cls
http://mc.manuscriptcentral.com/nme
DOI: 10.1002/nme

Table I. Number of scalar products per iteration of the PCG using the ROM/SROM preconditioners with either S=diag$(A)^{-1}$ or S=IC0 .

	S=diag$(A)^{-1}$	S=IC0
PCG	$8+\sigma$	$7+2\sigma$
ROM PCG	$8+\sigma+3m$	$7+2\sigma+3m$
SROM PCG	$8+\sigma+4m$	$7+2\sigma+4m$

Table II. Properties of the 3-D grids used in TC1-TC5.

	TC1	TC2	TC3	TC4	TC5
Nodes, n	1,880	13,149	97,937	755,073	755,073
Tetrahedra, r	8,529	68,160	545,280	4,362,240	4,362,240
Non-zeros	13,149	97,937	755,073	5,928,065	5,928,065
Sparsity, σ	12.98	13.89	14.41	14.70	14.70
Diffusion D	zones M1, M2				random

Table III. SROM preconditioner: total relative cost C (equation (29)) during the complete simulation. The basis vectors arise from Algorithm 3.

	$\tau=10^{-6}$			$\tau=10^{-10}$		
	TC2	TC3	TC4	TC2	TC3	TC4
m=1	-0.22	-0.25	-0.24	-0.09	-0.10	-0.09
m=2	-0.17	-0.19	-0.17	-0.03	-0.04	-0.02
m=5	-0.14	-0.10	-0.10	0.06	0.04	0.08
m=10	-0.17	-0.28	-0.08	0.13	-0.01	0.12
m=20	-0.07	-0.29	-0.45	0.33	0.26	0.08
m=30	-0.04	-0.23	-0.44	0.35	0.31	0.09

Table IV. ROM preconditioner: total relative cost C (equation (29)) during the complete simulation. The basis vectors arise from Algorithm 3.

	$\tau=10^{-6}$			$\tau=10^{-10}$		
	TC2	TC3	TC4	TC2	TC3	TC4
m=1	-0.20	-0.24	-0.25	-0.10	-0.12	-0.10
m=2	-0.16	-0.19	-0.20	-0.07	-0.08	-0.06
m=5	-0.17	-0.16	-0.16	0.06	-0.03	0.01
m=10	-0.24	-0.35	-0.18	0.09	-0.10	0.01
m=20	-0.15	-0.36	-0.48	0.18	0.09	-0.1
m=30	-0.09	-0.31	-0.48	0.21	0.14	-0.08

Table V. Total relative cost C (equation (29)) in TC5. 'n.c.' means that the PCG solver could not achieve convergence ($\tau=10^{-6}$). The basis vectors arise from Algorithm 3. Note that the relative cost of the Jacobi preconditioner ($diag(A)^{-1}$) is about 10 times the cost of IC0 in this application, so the ROM-based preconditioners are highly improving the Jacobi performance.

	SROM $S=$ IC0	ROM $S=$ IC0	SROM $S=diag(A)^{-1}$	ROM $S=diag(A)^{-1}$
m=1	-0.10	-0.12	0.58	0.52
m=2	-0.01	-0.05	0.81	0.70
m=5	0.21	n.c.	1.42	n.c.

Prepared using nmeauth.cls http://mc.manuscriptcentral.com/nme DOI: 10.1002/nme