Semi-incremental model order reduction approach for fatigue damage computations

Shadi Alameddin, Amélie Fau, David Néron, Pierre Ladevèze, Udo Nackenhorst

A Semi-incremental Scheme for Cyclic Damage Computations

Shadi Alameddin, Amélie Fau, David Néron, Pierre Ladevèze and Udo Nackenhorst

Abstract High fidelity structural problems that involve nonlinear material behaviour, when subjected to cyclic loading, usually demand infeasible computational resources; this demonstrates the need for efficient model order reduction (MOR) techniques in order to shrink these demands to fit into the available means. The solution of cyclic damage problems in a model order reduction framework is investigated in this chapter. A semi-incremental framework based on a large time increment (LATIN) approach is proposed to tackle cyclic damage computations under variable amplitude and frequency loadings. The involved MOR approach provides a low-rank approximation in terms of proper generalised decomposition (PGD) of the solution. The generated PGD basis can be interpreted as a set of linear subspaces altered on the fly to the current problem settings. The adaptation of PGD to new settings is based on a greedy algorithm that may lead to a large-sized reduced order basis (ROB). Thus, different orthonormalisation and compression techniques were evaluated to ensure the optimality of the generated ROB in [1] and will be overviewed here. The proposed implementation and a displacement formulated finite element (FE) incremental framework are compared to illustrate their differences in terms of memory footprint and computational time. Numerical examples with variable loadings are discussed, and a typical implementation is provided as open-source code, available at https://gitlab.com/shadialameddin/romfem.

1 Introduction

Fatigue damage is one of the primary failure mechanisms of structural components where the material fails below its nominal strength, and it is often hard to tell with simple observations when such failure may occur. Therefore, this phenomenon is

S. Alameddin (✉) · A. Fau · U. Nackenhorst
IBNM, Leibniz Universität Hannover, Appelstraße 9a, 30167 Hannover, Germany
e-mail: shadi.alameddin@ibnm.uni-hannover.de

D. Néron · P. Ladevèze
LMT, ENS Paris-Saclay, CNRS, Université Paris Saclay,
61 Avenue du Président Wilson, 94235 Cachan, France

of great interest for many researchers and engineers [2]. The fatigue life can be expressed as the number of loading cycles required to initiate a macrocrack and cause its propagation up to a specific limit where the nucleation, growth and coalescence of the microscopic cavities lead to the initiation of a mesocrack [2]. As indicated by experimental observations, at high-stress amplitude, i.e., low-cycle fatigue, fatigue cracks start to develop in the early cycles, whereas at low-stress levels: high-cycle fatigue, the crack initiation period may consume a significant percentage of the usable fatigue life [3]. In metallic alloys of steel and nickel, for instance, crack initiation accounts for more than 70–80% of the total fatigue life of the structure [4, 5] and the corresponding critical damage value mostly belongs to [0.2–0.5] [2]. It was noted in [6] that a life-prediction rule for fatigue damage accumulation under variable-amplitude loading does not exist yet. Hence, it is of interest to develop numerical techniques that can predict fatigue damage [7].

Numerical simulations appeal as an attractive augmentation to experiments to design and analyse mechanical structures. Despite the recent developments in computational resources that makes it feasible to solve systems with a substantial number of degrees of freedom efficiently, it is of common interest to reduce the numerical cost of numerical models throughout model order reduction strategies [8]. The performance of MOR techniques has been shown in different fields such as their application to nonlinear problems [9, 10], real-time computations [11] or for performing cyclic, parametric or probabilistic computations in which the information provided by some queries can be efficiently reused to respond to other queries that exhibit some similarities [12, 13].

In MOR, posteriori techniques such as the proper orthogonal decomposition (POD) are built on a data extraction phase where the dominant characteristics of the high fidelity model are extracted, offline, to build a ROB [14]. Then, the linear solver is restricted to search for a solution in the space spanned by the given ROB, which leads to high savings in the numerical cost [8]. The generation of the ROB in the offline phase limits the model flexibility toward variabilities in the online stage. Hence, adaptive strategies that enrich generated ROB to tackle nonlinearities are required [15]. Another way to ensure flexibility is the usage of a priori MOR techniques such as the proper generalised decomposition (PGD) where a low-rank approximation of the quantities of interest is assumed a priori while the actual search for this approximation is done in the online phase [16]. The optimality of the PGD basis is traded for its flexibility, which might lead to large sized reduced order bases reaching hundreds or even thousands of modes [17, 18]. Hence, efficient compression stages are appealing to be coupled with PGD. However, classical compression steps are computationally expensive, which prohibit their direct implementation in online phases. Deterministic and probabilistic compression algorithms that exploit the low-rank assumption are investigated in this work within a PGD framework.

PGD does not have any prior knowledge of the system behaviour, but it builds and enriches the knowledge of the system just-in-time. A PGD method coupled with a semi-explicit finite element scheme was used to obtain a stabilised cycle for an elastoplastic material in [17]. Another technique that uses PGD, since its early stages, is the LATIN method [19]. LATIN is a linearisation scheme that simplifies

the introduction of PGD in nonlinear mechanical computations. A review of the LATIN-PGD method and some of its recent extensions to nonlinear solid mechanics problems can be found in [16, 20].

Contrary to the classical LATIN formulations [10, 20], a semi-incremental formulation is proposed here where the temporal domain is divided into intervals that are solved consecutively instead of simultaneously. Thus, time integration is tackled over small domains compared to the whole time history, and data recycling plays a vital role in the efficiency of the scheme. This work shall address sophisticated viscoplastic damage model subjected to variable loading. The goal of this contribution is to extend the LATIN-PGD framework suggested in [21] to tackle general nonlinear constitutive models subjected to variable loading that causes some convergence issues which shall be tackled.

This chapter is structured as follows. An overview of the utilised LATIN scheme is detailed in Sect. 2 followed by its semi-incremental extension in Sect. 3. After that, modal optimisation techniques are investigated in Sect. 4. At the end, different numerical examples are discussed, in Sect. 5, to illustrate the robustness and efficiency of the proposed scheme.

1.1 Notation

The notation, used in this work, for different tensor types is summarised in Table 1.

The symmetric second order tensors such as $\boldsymbol{\sigma}$ is identified by its corresponding six-dimensional vector $\underline{\sigma} \in \mathbb{R}^6$. The same applies to fourth order tensors, e.g., $\mathbb{C} \leftrightarrow \underline{\underline{C}} \in \mathbb{R}^{6\times 6}$. In a three-dimensional Euclidean space $\mathbb{E}^3$ with an orthonormal basis $\{\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3\}$, the dot product of two vectors is written as $\underline{x} \cdot \underline{y} = \underline{x}^{\mathsf{T}}\, \underline{y}$, and the following contractions are defined

Table 1 Symbols and their representation

Symbolic representation	Verbal representation
a, φ	Scalars: lowercase letters
$\boldsymbol{u}, \boldsymbol{x}$	First-order tensors: lowercase boldface letters
$\boldsymbol{I}, \boldsymbol{N}$	Second-order tensors: uppercase boldface letters
$\boldsymbol{\sigma}, \boldsymbol{\varepsilon}$	Second-order tensors: greek boldface letters
$\mathbb{C}, \mathbb{H}$	Fourth-order tensor: blackboard bold letters
$\underline{a}$	Column vector
$\underline{\underline{A}}$	Matrix

$$
\begin{aligned}
\boldsymbol{\sigma} : \boldsymbol{\varepsilon} &= \sum_{i,j=1}^{3} \sigma_{ij}\, \varepsilon_{ij}, \\
\mathbb{C} : \boldsymbol{\varepsilon} &= \sum_{i,j,k,l=1}^{3} C_{ijkl}\, \varepsilon_{kl}\, \boldsymbol{e}_i \otimes \boldsymbol{e}_j, \\
\boldsymbol{\sigma} \cdot \boldsymbol{n} &= \sum_{i,j=1}^{3} \sigma_{ij}\, n_j\, \boldsymbol{e}_i.
\end{aligned} \tag{1}
$$

The partial derivative operator ∇ is defined as $\nabla = \left[\frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2}, \frac{\partial}{\partial x_3}\right]^{\mathsf{T}}$. Then, the divergence differential operator of a vector field $\boldsymbol{u}(\boldsymbol{x})$ is denoted by $\nabla \cdot \boldsymbol{u} = \nabla_{\boldsymbol{x}} \cdot \boldsymbol{u}$ while the gradient operator is defined as a right gradient via

$$
\nabla \boldsymbol{u} = \boldsymbol{u} \otimes \nabla_{\boldsymbol{x}}. \tag{2}
$$

After that, a symmetric gradient operator is denoted by

$$
\nabla^{\mathrm{s}} \boldsymbol{u} = \frac{1}{2} \left(\boldsymbol{u} \otimes \nabla_{\boldsymbol{x}} + \nabla_{\boldsymbol{x}} \otimes \boldsymbol{u}\right) = \frac{1}{2} \left(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^{\mathsf{T}}\right). \tag{3}
$$

2 An Overview of the LATIN-PGD Method

Within an infinitesimal strain and isothermal quasi-static framework, the deformation of an isotropic solid material occupying the domain $\Omega \subset \mathbb{R}^d$, with the spatial dimension $d = 3$, is described in terms of the Cauchy stress tensor $\boldsymbol{\sigma} : \Omega \to \mathbb{R}^{d \times d}$, the strain tensor $\boldsymbol{\varepsilon} : \Omega \to \mathbb{R}^{d \times d}$, the volumetric force density $\boldsymbol{b} : \Omega \to \mathbb{R}^d$ and the displacement field $\boldsymbol{u} : \Omega \to \mathbb{R}^d$. The balance of linear and angular momentum lead to the definition of an initial boundary value problem (IBVP) which is stated in the strong formulation as [22]

$$
\begin{aligned}
\nabla \cdot \boldsymbol{\sigma}(\boldsymbol{x}, t) + \boldsymbol{b} &= \boldsymbol{0} && \forall (\boldsymbol{x}, t) \in \Omega \times \mathcal{I}, \\
\boldsymbol{u} &= \bar{\boldsymbol{u}} && \text{on } \partial\Omega_D \times \mathcal{I}, \\
\boldsymbol{\sigma} \cdot \boldsymbol{n} &= \bar{\boldsymbol{t}} && \text{on } \partial\Omega_N \times \mathcal{I}, \\
\boldsymbol{u}(\boldsymbol{x}, t)|_{t=0} &= \boldsymbol{u}_0(\boldsymbol{x}) && \text{in } \Omega \text{ at } t = 0,
\end{aligned} \tag{4}
$$

where $\boldsymbol{n}$ represents the outward unit normal, $\mathcal{I} = [0, T]$, $\partial\Omega_D \cup \partial\Omega_N = \partial\Omega$, $\partial\Omega_D \cap \partial\Omega_N = \emptyset$ and $\boldsymbol{\sigma} = \boldsymbol{\sigma}^{\mathsf{T}}$, $\boldsymbol{\varepsilon} = \nabla^{\mathrm{s}} \boldsymbol{u}$. The stress is related to displacement through a constitutive model $\boldsymbol{\sigma} = \tilde{f}(\boldsymbol{\varepsilon}(\boldsymbol{u}), \boldsymbol{q})$, $\boldsymbol{q}(\boldsymbol{x}, t)_{|t=0} = \boldsymbol{q}_0(\boldsymbol{x})$. Here, the Neumann boundary conditions are assumed to be homogeneous by taking $\bar{\boldsymbol{t}} = \boldsymbol{0}$ on $\partial\Omega_N$.

The equilibrium and kinematic equations, in LATIN, are solved simultaneously ahead of the evaluation of the constitutive relations, (evolution and state laws [23]), in the next step. Then, these two steps are repeated iteratively until a solution that satisfies all governing equations is found. Throughout the iterations, feedback between the two stages mentioned above is achieved via what is called search direction equations. With the LATIN scheme, the highest computational complexity is confined to the solution of the linear global equilibrium system that may benefit from MOR schemes such as PGD to produce an efficient scheme [16].

In PGD, the quantity of interest, the displacement field for instance, may be approximated by a finite sum of separated functions via

$$\boldsymbol{u}(\boldsymbol{x},t) \approx \sum_{j=1}^{N} \boldsymbol{v}_j(\boldsymbol{x}) \circ \boldsymbol{\lambda}_j(t), \qquad \boldsymbol{v}_j(\boldsymbol{x}) : \Omega \to \mathbb{R}^d,\ \boldsymbol{\lambda}_j(t) : \mathcal{I} \to \mathbb{R}^d, \tag{5}$$

where $N \in \mathbb{N}_+$ and $\circ$ is the entry-wise Hadamard product [16, 24]. Terms of Eq. (5) are generated on-the-fly throughout a greedy algorithm that aims at reducing the approximation error [25].

The viscoplastic damage model, utilised in this work, is based on the Helmholtz free energy density function $\psi(\boldsymbol{\varepsilon}, \boldsymbol{\varepsilon}^{\mathrm{p}}, \boldsymbol{q})$ and the dissipation potential $\phi(\boldsymbol{\sigma}, \boldsymbol{Q})$, where $\boldsymbol{\sigma}$ is the Cauchy stress, $\boldsymbol{\varepsilon}^{\mathrm{p}}$ denotes the plastic strain, $\boldsymbol{q}$ is a vector field of additional internal variables such as hardening and damage, and $\boldsymbol{Q}$ contains the conjugate variables of $\boldsymbol{q}$. The state equations are derived from the free energy as

$$\boldsymbol{\sigma} = f(\psi(\boldsymbol{\varepsilon}, \boldsymbol{\varepsilon}^{\mathrm{p}}, \boldsymbol{q})), \quad \boldsymbol{Q} = g(\psi(\boldsymbol{\varepsilon}, \boldsymbol{\varepsilon}^{\mathrm{p}}, \boldsymbol{q})), \tag{6}$$

while the evolution of the internal variables is linked to the dissipation potential via

$$\dot{\boldsymbol{\varepsilon}}^{\mathrm{p}} = \hat{f}(\phi(\boldsymbol{\sigma}, \boldsymbol{Q})), \quad \dot{\boldsymbol{q}} = \hat{g}(\phi(\boldsymbol{\sigma}, \boldsymbol{Q})). \tag{7}$$

Starting with an elastic initialisation, the LATIN scheme seeks a solution throughout two consecutive stages, the local and global ones. The solution at each step is denoted by $s = \{\boldsymbol{X}, \dot{\boldsymbol{Y}}\}$, where $\boldsymbol{X} = \{\boldsymbol{\sigma}, \boldsymbol{Q}\}$ contains the dynamic conjugate variables and $\dot{\boldsymbol{Y}} = \{\dot{\boldsymbol{\varepsilon}}^{\mathrm{p}}, \dot{\boldsymbol{q}}\}$ contains the evolution of the internal variables.

The boundary conditions are addressed in the elastic solution $s_0 = \{\boldsymbol{X}_0, \dot{\boldsymbol{Y}} = \boldsymbol{0}\}$ while the following iterations compute corrections to s_0.

2.1 Local Stage

The evolution and state equations, Eqs. (6) and (7), in addition to the additive split of the infinitesimal strain are solved within the local stage at every space-time point. The initial conditions for $\dot{\boldsymbol{Y}}$ in Ω at $t = 0$ are addressed here. The local search direction equation reads

$$(\hat{\boldsymbol{\sigma}}_i - \boldsymbol{\sigma}_i) + \mathbb{H}^+ : (\hat{\boldsymbol{\varepsilon}}_i - \boldsymbol{\varepsilon}_i) = \mathbf{0}, \tag{8}$$

where $\boldsymbol{\sigma}_i$ and $\boldsymbol{\varepsilon}_i$ come from the previous global stage, $\hat{\boldsymbol{\sigma}}_i$ and $\hat{\boldsymbol{\varepsilon}}_i$ are the outcome of the current local stage and $\mathbb{H}^+$ is the direction of ascent. The resulting approximation $\hat{s}_i$ is used in the following global step to produce s_{i+1}.

2.2 Global Stage

The admissibility equations are what is left to the global stage. Deriving the strain as the symmetric gradient of the displacement field $\boldsymbol{\varepsilon} = \nabla^{\mathrm{s}}\boldsymbol{u}$ with $\boldsymbol{u} = \bar{\boldsymbol{u}}$ on $\partial\Omega_D \times \mathcal{I}$ and $\boldsymbol{u}(\boldsymbol{x}, 0) = \mathbf{0}$ satisfies the kinematic admissibility while the static admissibility condition is obtained via applying the Hamilton law of varying action [26] on Eq. (4), leading to the following weak form

$$\int_{\Omega\times\mathcal{I}} \boldsymbol{\sigma} : \boldsymbol{\varepsilon}^* \, \mathrm{d}\Omega \, \mathrm{d}t = \int_{\Omega\times\mathcal{I}} \boldsymbol{b}\cdot\boldsymbol{u}^* \, \mathrm{d}\Omega \, \mathrm{d}t + \int_{\partial\Omega_N\times\mathcal{I}} \bar{\boldsymbol{t}}\cdot\boldsymbol{u}^* \, \mathrm{d}S \, \mathrm{d}t \qquad \forall \boldsymbol{u}^* \in \mathcal{U}_0, \tag{9}$$

where the test space is a Bochner space, defined as $\mathcal{U}_0 = \{\boldsymbol{u}(\boldsymbol{x}, t) \mid \boldsymbol{u}(\boldsymbol{x}, t) \in L^2(\mathcal{I}; \left[H_0^1(\Omega)\right]^d) \equiv L^2(\mathcal{I}) \otimes \left[H_0^1(\Omega)\right]^d, \ \boldsymbol{u} = \mathbf{0} \text{ on } \partial\Omega_D \times \mathcal{I}\}$. The corrections in each iteration, in terms of displacement, are defined as $\Delta\boldsymbol{u}_{i+1} = \boldsymbol{u}_{i+1} - \boldsymbol{u}_i$ where the subscripts refer to the previous and the current global stage.

The global search direction equation is defined as

$$(\boldsymbol{\sigma}_{i+1} - \hat{\boldsymbol{\sigma}}_i) - \mathbb{H}^- : (\boldsymbol{\varepsilon}_{i+1} - \hat{\boldsymbol{\varepsilon}}_i) = \mathbf{0}, \tag{10}$$

where $\hat{\boldsymbol{\sigma}}_i$ and $\hat{\boldsymbol{\varepsilon}}_i$ belong to the last local stage and $\mathbb{H}^-$ is the direction of descent. Here, $\mathbb{H}^-$ is assumed to be a scaled version of the elasticity tensor, i.e., $\mathbb{H}^- = \alpha\,\mathbb{C}$, where $\alpha \in [0, 1]$. The parameter α is taken as constant in the following derivations.

In order to reduce the computational cost of Eq. (9), a PGD representation of the displacement field is introduced as

$$\Delta\boldsymbol{u} = \boldsymbol{v}(\boldsymbol{x}) \circ \boldsymbol{\lambda}(t), \qquad \boldsymbol{v}(\boldsymbol{x}) : \Omega \to \mathbb{R}^d, \ \boldsymbol{\lambda}(t) : \mathcal{I} \to \mathbb{R}^d. \tag{11}$$

The temporal functions are chosen to be identical for all the spatial coordinates, i.e., $\lambda_x(t) = \lambda_y(t) = \lambda_z(t)$, leading to

$$\Delta\boldsymbol{u} = \boldsymbol{v}(\boldsymbol{x})\,\lambda(t), \qquad \Delta\boldsymbol{\varepsilon} = \nabla^{\mathrm{s}}\boldsymbol{v}(\boldsymbol{x})\,\lambda(t), \qquad \boldsymbol{v}(\boldsymbol{x}) : \Omega \to \mathbb{R}^d, \ \lambda(t) : \mathcal{I} \to \mathbb{R}. \tag{12}$$

The subscript $i + 1$ is dropped for the sake of compact notation and it is assumed that only one PGD term/pair is generated within one iteration. Following the derivations in [10, 27] by introducing the PGD scheme into the equilibrium equation and solving the resulting system via an alternated directions algorithm [16] lead to a spatial and

temporal problems. After incorporating a Galerkin finite element discretisation, for Ω and $\mathcal{I}$, the discrete spatial problem, with homogeneous boundary conditions, reads

$$\gamma \, \underline{\underline{K}} \, \underline{v} = \underline{f} \qquad \gamma \in \mathbb{R} \qquad \underline{\underline{K}} \in \mathbb{R}^{n \times n} \qquad \underline{v}, \underline{f} \in \mathbb{R}^{n} \qquad \text{with } \underline{v} = \underline{0} \text{ on } \partial\Omega_D. \quad (13)$$

The temporal problem, with zero initial conditions, is written as

$$a \, \underline{\lambda}^{\mathsf{T}} = \underline{b}^{\mathsf{T}} \qquad a \in \mathbb{R} \qquad \underline{\lambda}, \underline{b} \in \mathbb{R}^{n_t} \qquad \text{with } \underline{\lambda}_{|t=0} = 0, \quad (14)$$

where (n, n_t) are the spatial and temporal degrees of freedom and $(\underline{v}, \underline{\lambda})$ are the spatial and temporal functions. The stiffness matrix $\underline{\underline{K}} = \int_{\Omega} \underline{\underline{B}}^{\mathsf{T}} \, \underline{\underline{C}} \, \underline{\underline{B}} \, \mathrm{d}\Omega$, where the globally assembled matrix $\underline{\underline{B}} \in \mathbb{R}^{6n_{gp} \times n}$ contains the derivatives of the shape functions and $\underline{\underline{C}}$ is a sparse matrix with 6×6 diagonal blocks representing the elasticity tensor at each integration point. The scaling factor in front of the stiffness matrix is defined as $\gamma = \alpha \int_{\mathcal{I}} \underline{\lambda}^{\mathsf{T}} \underline{\lambda} \, \mathrm{d}t$. The right-hand side of Eq. (13) is given via $\underline{f} = -\int_{\Omega} \underline{\underline{B}}^{\mathsf{T}} \left(\int_{\mathcal{I}} \underline{\underline{\hat{f}}} \, \underline{\lambda} \, \mathrm{d}t \right) \mathrm{d}\Omega$, where $\underline{\underline{\hat{f}}}$ is obtained from the preceding local stage. The temporal problem is defined by $a = \alpha \int_{\Omega} (\underline{\underline{B}} \, \underline{v})^{\mathsf{T}} \, \underline{\underline{C}} \, (\underline{\underline{B}} \, \underline{v}) \, \mathrm{d}\Omega$ and $\underline{b}^{\mathsf{T}} = -\int_{\Omega} (\underline{\underline{B}} \, \underline{v})^{\mathsf{T}} \, \underline{\underline{\hat{f}}} \, \mathrm{d}\Omega$.

Using μ PGD modes at the $(i+1)$th iteration, the discrete displacement is approximated by

$$\underline{\underline{u}}_{i+1} = \underline{\underline{u}}_{0} + \sum_{j=1}^{\mu} \underline{v}_j \, \underline{\lambda}_j^{\mathsf{T}}, \quad (15)$$

with $\underline{\underline{u}}_{0}$ coming from the elastic solution.

The global stage is dominated by the computational demands of the spatial problem, Eq. (13). Hence, the global stage carries initially a POD-like step that reuses previously generated spatial modes and updates the temporal ones [21].

2.2.1 Temporal Modes Update

Starting with a certain number (μ) of previously generated PGD modes, the displacement correction reads

$$\Delta \underline{\underline{u}}_{i+1} = \sum_{j=1}^{\mu} \underbrace{\underline{v}_j}_{known} \, \Delta \underline{\lambda}_j^{\mathsf{T}}, \quad (16)$$

where $\Delta \underline{\lambda}_j(t)$ is the correction to $\underline{\lambda}_j(t)$. Introducing this correction into the temporal problem, Eq. (14), leads to

$$\underline{\underline{\tilde{A}}} \, \underline{\tilde{\lambda}}^{\mathsf{T}} = \underline{\tilde{B}} \qquad \underline{\underline{\tilde{A}}} \in \mathbb{R}^{\mu \times \mu} \qquad \underline{\tilde{B}} \in \mathbb{R}^{\mu \times n_t}, \quad (17)$$

where $\underline{\underline{\tilde{\Lambda}}} = [\Delta\underline{\lambda}_1, \ldots, \Delta\underline{\lambda}_\mu]$ and

$$\tilde{A}_{kl} = \alpha \int_\Omega (\underline{\underline{B}}\ \underline{v}_k)^\mathsf{T} \underline{\underline{C}}\ (\underline{\underline{B}}\ \underline{v}_l)\ \mathrm{d}\Omega, \qquad \tilde{B}_{kl} = -\int_\Omega (\underline{\underline{B}}\ \underline{v}_k)^\mathsf{T} \underline{\hat{f}}_{|t=t_l}\ \mathrm{d}\Omega. \tag{18}$$

The computational cost of this update depends only on the temporal discretisation n_t and the size of the ROB measured by μ.

Note that the displacement spatial modes $\{\underline{v}_j\}_{j=1}^{\mu}$ are not stored. However, the corresponding strain spatial modes $\{\underline{\underline{B}}\ \underline{v}_j\}_{j=1}^{\mu}$ are the ones stored and orthonormalised afterwards.

When the update step introduces a significant change to the original temporal modes, measured by $\left(\|\Delta\underline{\lambda}_j\|_I / \|\underline{\lambda}_j\|_I$, with $\|\underline{\lambda}\|_I = \sqrt{\int_T \underline{\lambda}^\mathsf{T}\underline{\lambda}\,\mathrm{d}t}\right)$, no expansion of the ROB is required and the algorithm returns from the global stage.

3 Variable Amplitude and Frequency Loading

A semi-incremental extension to the classical LATIN formulation is proposed here where the temporal domain is divided into intervals, and the solution is computed for each interval consecutively instead of solving all of them simultaneously.

The solution involving multiple cycles with variable loads starts by dividing the temporal domain into subintervals with an equal number of time points in each one, for simplicity. Then the linearity in the elastic initialisation is exploited by scaling the previous elastic solution without the need to assemble the stiffness matrix again or solve the elastic problem once more. Other information from previous cycles is recycled as well, such as the generated PGD pairs. In such case, the temporal modes have to be scaled (based on the new load) and shifted to ensure continuity of the obtained solutions across all cycles or time intervals, i.e., the modified and shifted final solution of cycle n is used as an initial guess s_0 for cycle $n+1$.

Starting from a given temporal mode λ_n, that belongs to a previously simulated cycle, the temporal mode of the next cycle λ_{n+1} has to be estimated. The load over two consecutive cycles is assumed cyclic, to simplify the following explanation, with different amplitudes A_n, A_{n+1} and different time periods T_n, T_{n+1}, respectively. In order to address variable frequencies, the cycles are assumed to be defined over a dimensionless time coordinate $\tilde{t} \in [0, 1]$. Then a simple way to ensure continuity and account for variable amplitudes is to define a new temporal mode as a scaled version of a previous one, with $m = A_{n+1}/A_n$, plus a linear function of time with appropriate boundary conditions, e.g.,

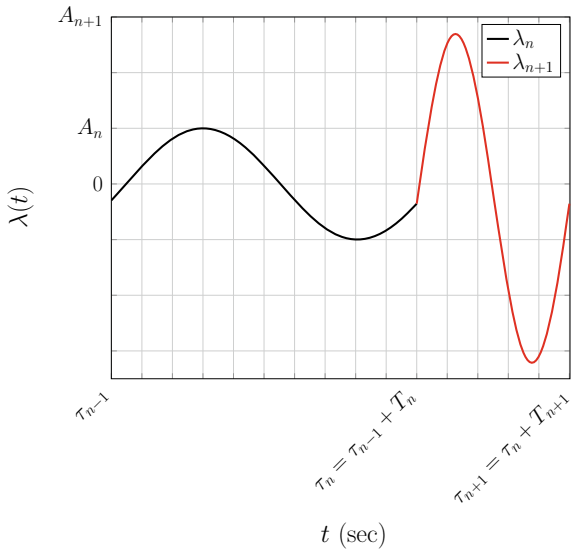

Fig. 1 Continuity of the temporal modes

$$\begin{aligned} &\tilde{\lambda}_{n+1}(\tilde{t}) = m\,\tilde{\lambda}_n(\tilde{t}) + g\,\tilde{t} + h \quad \text{with} \\ &\tilde{\lambda}_{n+1}(0) = \tilde{\lambda}_n(1), \quad \tilde{\lambda}_{n+1}(1) = \tilde{\lambda}_{n+1}(0). \end{aligned} \tag{19}$$

The new temporal mode can be scaled to the original time coordinate via $\lambda_{n+1}(\tau_n + T_{n+1}\,\tilde{t}) = \tilde{\lambda}_{n+1}(\tilde{t})$ to obtain the modified temporal mode, as illustrated in Fig. 1.

The previously introduced scheme renders a solution algorithm, for variable loading cases, that involves the assembly of the stiffness matrix only once. Besides, the computation of each new interval starts with an initial guess from the previous one. Note that the temporal discretisation does not enter linearly in the model complexity due to the semi-incremental scheme and the storage requirement is proportional to the temporal discretisation within one load cycle only and the number of generated modes.

3.1 Hybrid Search Direction Formulation

Initially the direction of ascent in Eq. (8) is considered to be horizontal, similar to [28], i.e., $\mathbb{H}^+$ is zero. Hence, due to the assumption of a given stress state, the local stage renders a straightforward nonlinear function evaluation rather than a nonlinear system of equations. However, as illustrated in [29], nonexisting or multiple solutions might be an issue with such formulation. On the other hand, if only the strain is prescribed from the global stage, with a vertical search direction, i.e., $\mathbb{H}^+$ is infinity, then the local stage becomes identical to the classical update of internal variables

usually implemented in incremental linearisation schemes leading to a demanding stage due to the involved nonlinear system of equations.

Based on the discussion above, a hybrid formulation that involves both horizontal and vertical search directions is proposed here. In order to ensure the computational efficiency of the ROM, the scheme uses the horizontal search direction always unless a convergence issue is detected, then the search direction is switched to the vertical one for one iteration only. Divergence may be detected based on the utilised error indicator or based on the damage value exceeding one, for instance. Contrary to other LATIN implementations, with the presented algorithm here, there is no need for a relaxation scheme to ensure convergence.

4 Optimality of the Generated ROB

The ith correction or solution of the LATIN algorithm, in an outer-product form [30], reads

$$\underline{\underline{\tilde{U}}} = \sum_{j=1}^{\mu} \underline{v}_j \, \underline{\lambda}_j^{\mathsf{T}} = \underline{\underline{V}} \, \underline{\underline{\Lambda}}^{\mathsf{T}} \in \mathbb{R}^{n \times n_t}, \tag{20}$$

where $\underline{\underline{V}} = [\underline{v}_1, \ldots, \underline{v}_\mu] \in \mathbb{R}^{n \times \mu}$ and $\underline{\underline{\Lambda}} = [\underline{\lambda}_1, \ldots, \underline{\lambda}_\mu] \in \mathbb{R}^{n_t \times \mu}$. Practically, the spatial functions $\underline{v}_j$ are orthonormalised via an othonormalisation scheme such as Gram–Schmidt (GS) procedure [10]. However, using a GS scheme with the LATIN algorithm requires generating many modes to obtain a low approximation error. Hence, it is not enough to have orthonormal spatial modes to ensure a small sized PGD ROB [9].

4.1 Randomised Singular Value Decomposition (RSVD) Compression of PGD Bases

Singular value decomposition provides an optimal, see [31] for details, and straightforward decomposition of the full solution by compressing spatial and temporal information into a minimal set of global functions. However, due to the high cost of applying a singular value decomposition (SVD) at each LATIN iteration a randomised SVD algorithm [32] is used to compress the PGD expansion. An overview

of the randomised SVD algorithm applied to a quantity of interest $\tilde{U}$ defined over $\Omega \times \mathcal{I}$ is briefed in Algorithm 1.

Algorithm 1: Randomised singular value decomposition (RSVD)

Data: given quantity of interest $\underline{\underline{\tilde{U}}} \in \mathbb{R}^{n \times n_t}$

Result: tuncated SVD of $\underline{\underline{\tilde{U}}}$ with size k

Use a a random matrix $\underline{\underline{Q}} \in \mathbb{R}^{n_t \times k}$ to obtain $\underline{\underline{E}} = \underline{\underline{\tilde{U}}}\ \underline{\underline{Q}} \in \mathbb{R}^{n \times k}$

Use GS to orthonormalise the columns of $\underline{\underline{E}}$

Compute a restriction of $\underline{\underline{\tilde{U}}}$ via $\underline{\underline{S}} = \underline{\underline{E}}^{\mathsf{T}} \underline{\underline{\tilde{U}}} \in \mathbb{R}^{k \times n_t}$

Compute a truncated SVD $\underline{\underline{S}} \approx \underline{\underline{S}}^{(k)} = \underline{\underline{\tilde{\tilde{V}}}}\ \underline{\underline{\tilde{S}}}\ \underline{\underline{\tilde{\Lambda}}}^{\mathsf{T}}$

Expand $\underline{\underline{S}}$ via $\underline{\underline{\tilde{U}}} \approx \underline{\underline{E}}\ \underline{\underline{S}}^{(k)} = \underline{\underline{E}}\ \underline{\underline{\tilde{V}}}\ \underline{\underline{\tilde{S}}}\ \underline{\underline{\tilde{\Lambda}}}^{\mathsf{T}} = \underline{\underline{\tilde{V}}}\ \underline{\underline{\tilde{S}}}\ \underline{\underline{\tilde{\Lambda}}}^{\mathsf{T}}$

Taking the outer-product form of the PGD into account results in an efficient orthonormalisation scheme [30]; details are provided in Algorithm 2.

Algorithm 2: Orthonormalisation scheme that exploits the PGD expansion

Previously generated modes $\{\underline{v}_j, \underline{\lambda}_j\}\ (j = 1, \ldots, \mu)$

Data: New pair of modes $\{\underline{v}_{\mu+1}, \underline{\lambda}_{\mu+1}\}$

Required number of modes / truncation threshold $k \leq \mu + 1,\ \epsilon_{\text{tol}}$

Result: Enriched basis $\{\underline{v}_j, \underline{\lambda}_j\}\ (j = 1, \ldots, k)$ with $\underline{v}_l^{\mathsf{T}}\ \underline{v}_m = \delta_{lm}$

Use a QR-decomposition to obtain $\underline{\underline{V}} = \underline{\underline{Q}}_v\ \underline{\underline{R}}_v, \underline{\underline{\Lambda}} = \underline{\underline{Q}}_\lambda\ \underline{\underline{R}}_\lambda$

Compute $\underline{\underline{R}} = \underline{\underline{R}}_v\ \underline{\underline{R}}_\lambda^{\mathsf{T}} \in \mathbb{R}^{(\mu+1)\times(\mu+1)}$ and approximate it as $\sum_{j=1}^{k} \tilde{s}_j\ \tilde{v}_j\ \tilde{\lambda}_j^{\mathsf{T}}$

Recover the outer-product representation:

$\underline{\underline{V}} \leftarrow \underline{\underline{Q}}_v\ \underline{\underline{\tilde{V}}} \in \mathbb{R}^{n \times k}$

$\underline{\underline{\Lambda}} \leftarrow \underline{\underline{Q}}_\lambda\ \underline{\underline{\tilde{\Lambda}}}\ \underline{\underline{\tilde{S}}}^{\mathsf{T}} \in \mathbb{R}^{n_t \times k}$

Further details of deterministic and randomised orthonormalisation algorithms may be found in [32–34]. However, the aim at this stage is to illustrate the applicability of the presented algorithms to be combined with the LATIN scheme.

5 Numerical Results

The analysis is carried out on a three-dimensional plate with a central groove. One-eighth of the plate with symmetric boundary conditions is shown in Fig. 2. The plate geometry is defined by its length, width and depth being (40, 20, 2) mm while the length and width of the groove are (10, 4) mm. This plate is subjected to a uniformly distributed displacement field of the form $U_d = U_0 \sin\left(\frac{2\pi}{T} t\right)$ with t and T being the time and the time period, respectively. The material used in the following examples is Cr-Mo steel at 580 °C.

Three examples are shown below. Firstly, a comparison with a standard incremental scheme for a cyclic load is presented where the error is computed for the

quantity of interest (damage) rather than displacement. Then, a comparison between an SVD and RSVD schemes is provided to illustrate the advantages of randomised algorithms. At the end, the proposed scheme is analysed on a periodic loading with variable amplitudes and frequencies.

5.1 Model Verification

The analysis of the plate, shown in Fig. 2, is carried out on a mesh that consists of 387 hexahedron elements, with eight integration points in each element, resulting in 1884 spatial displacement degrees of freedom. The model is subjected to a uniformly distributed displacement field with an amplitude of $U_0 = 0.004$ mm. The time discretisation is chosen such that the temporal domain of each cycle is discretised into 200 time steps. Since the whole temporal domain of each cycle is computed at once, a total of 376,800 degrees of freedom are being sought. The convergence and the enrichment criteria are taken as $tol_1 = 10^{-14}$ and $tol_2 = 10^{-1}$, respectively. The simulation of the plate was carried out using a Newton-Raphson incremental scheme and the LATIN-PGD scheme.

The prescribed load is illustrated in Fig. 3a. The corresponding damage evolution curve for the integration point with maximum damage as well as the corresponding error with respect to the incremental scheme are illustrated in Fig. 3b.

A total of four PGD modes, see Fig. 4a, were enough to maintain a relative error, in the maximum damage, below 0.2%. This error might be an outcome of using different temporal integration schemes between the incremental and semi-incremental algorithms. The convergence behaviour of the proposed algorithm is illustrated in Fig. 4b where the number of iterations required to converge in each cycle and their corresponding error indicator values are depicted. The initial slow decrease in the error indicator, in the second cycle in Fig. 4b, is due to the update of the temporal functions before enriching the generated basis.

In these simulations, a speedup factor of 20, in favour of the semi-incremental scheme is obtained through the provided prototype implementation. The change in the memory footprint is negligible in the given example. However, it is worth noting

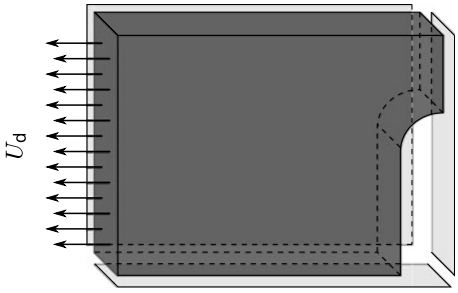

Fig. 2 A plate with a central groove subjected to cyclic loading

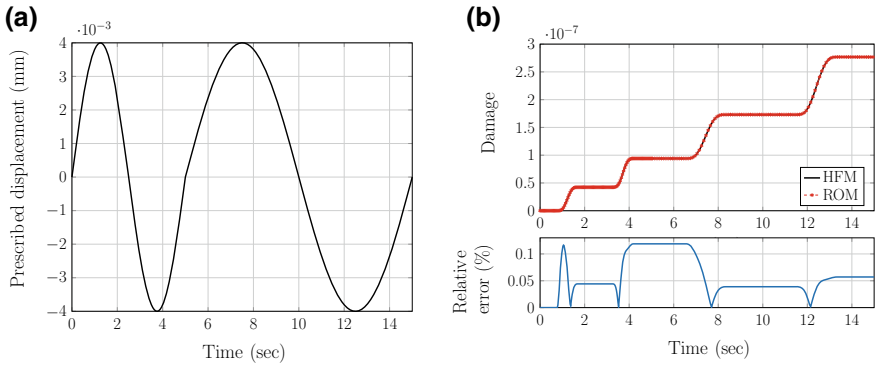

Fig. 3 Verification with respect to an incremental scheme. **a** The prescribed load; **b** damage evolution w.r.t. time along with the corresponding error

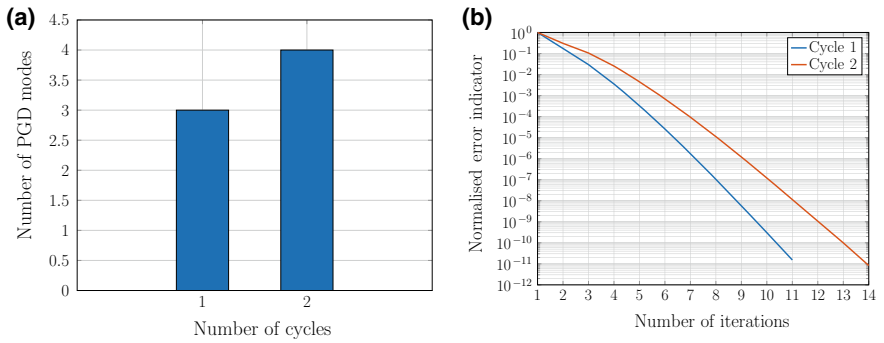

Fig. 4 Required number of modes and the corresponding error indicator. **a** Number of generated PGD modes w.r.t. number of cycles; **b** error indicator and number of iterations for each cycle

that all the fields are stored over the whole temporal domain to ease the comparison and postprocessing procedures.

5.2 Comparison Between Deterministic and Randomised SVD Schemes

Following is an example that illustrates the difference between an SVD scheme and an RSVD one, as presented in Algorithm 1, in case of two given quantities of interest stored in the matrices $\underline{\underline{M}}_1 \in \mathbb{R}^{10^6 \times 10^2}$ and $\underline{\underline{M}}_2 \in \mathbb{R}^{10^6 \times 10^3}$. Note that this example reproduces results from the literature to give a feeling on possible speedups using the following MATLAB® implementation.

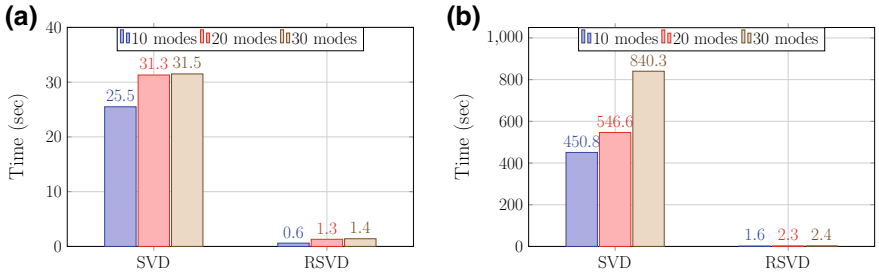

Fig. 5 The required time to extract the first 10, 20 and 30 singular vectors using SVD and RSVD. **a** Timing of modal extraction from $\underline{\underline{M}}_1$; **b** timing of modal extraction from $\underline{\underline{M}}_2$

RSVD implementation in MATLAB®

```
function [V, S, L] = randomised_svd(A, k)
    n = size(A, 2);
    random_matrix = randn(n, min(k+5, n));
    approximate_basis = A * random_matrix;
    orthonormalised_basis = orth(approximate_basis);
    A_restricted = transpose(orthonormalised_basis) * A;
    [V_restricted, S, L] = svds(A_restricted, k);
    V = orthonormalised_basis * V_restricted;
end
```

The cost to extract the first 10, 20 and 30 singular vectors is recorded and summarised in Fig. 5.

It is seen that when a low-rank approximation is sought, RSVD is very efficient compared to an SVD, and its cost does not increase linearly with the number of sought singular values. It is known that the provided timing are resources dependent. However, their relative performance is expected not to change. The RSVD algorithm is implemented in MATLAB® and uses its built-in SVD routine.

5.3 Variable Amplitude and Frequency Loading

This example illustrates the capabilities of the proposed scheme and its numerical efficiency to simulate cyclic loads with variable amplitudes and frequencies. An increasing-decreasing load is investigated in this example where the higher the load amplitude, the shorter the time period, as seen in Fig. 6a. The chosen load amplitudes

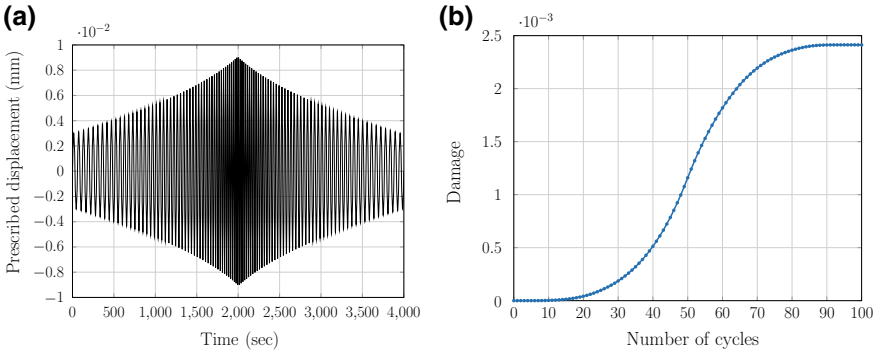

Fig. 6 Variable amplitude and frequency loading scenario. **a** Prescribed load; **b** damage evolution w.r.t. the number of cycles

belong to $[30, 90] \times 10^{-4}$ mm while the time periods vary in $[20, 60]$ s. The time discretisation is chosen such that the temporal domain of each cycle is discretised into 40 time steps. The convergence and the enrichment criteria are taken as $tol_1 = 10^{-3}$ and $tol_2 = 10^{-1}$, respectively.

The damage evolution for the integration point with maximum damage is depicted in Fig. 6. In order to better investigate the damage response in Fig. 6, the damage increment in each load cycle is demonstrated in Fig. 7b. It can be seen that Fig. 7b is not exactly symmetric due to the nonlinear response, which is driven by different initial conditions in each cycle and different plastic strain rates. As expected, the maximum damage increment corresponds to the cycle with maximum load amplitude.

A maximum of eight PGD pairs is enough to obtain the response above. The number of PGD modes with respect to the number of cycles is shown in Fig. 7a. It can be seen that the ROB size does not grow linearly with respect to the number of cycles and after simulating almost half of the cycles and facing all the different amplitudes and frequencies for the first time, the basis size does not grow any more.

An (R)SVD compression does not only provides optimality of the ROB; it also leads to not wasting resources in the update of the temporal functions. When the (R)SVD scheme is invoked at each LATIN iteration, regardless of whether it is an update or enrichment step as in Fig. 8b, is referred to in the sequel as excessive (R)SVD and abbreviated by e(R)SVD.

After illustrating the interests of using an optimal orthonormalisation scheme, the problem is solved again but with finer spatial discretisation to show the benefits of such a scheme when the problem gets more challenging. The plate model is discretised into 13,812 hexahedron elements, with eight integration points in each element, resulting in 50,547 spatial displacement degrees of freedom. The temporal discretisation consists of 33 time steps in each cycle resulting in 1,668,051 degrees of freedom in each cycle. The plate is subjected to a uniformly distributed displacement field with a uniformly distributed random amplitudes in the range of $[18, 22] \times$

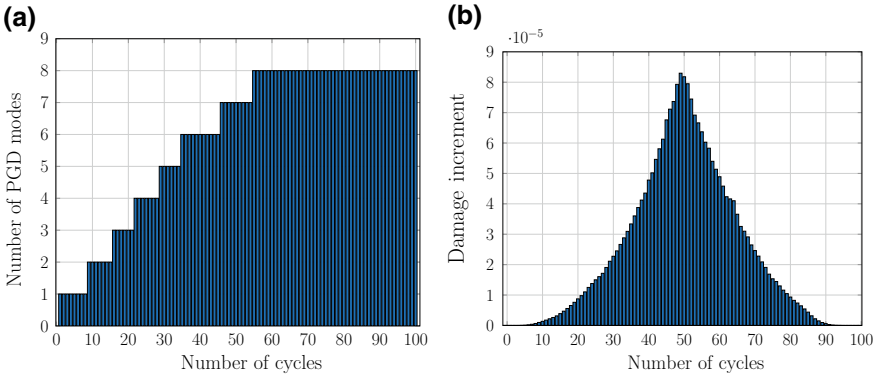

Fig. 7 ROB size and damage increment with respect to the number of cycles. **a** The growth of the PGD basis; **b** damage increment w.r.t. the number of cycles

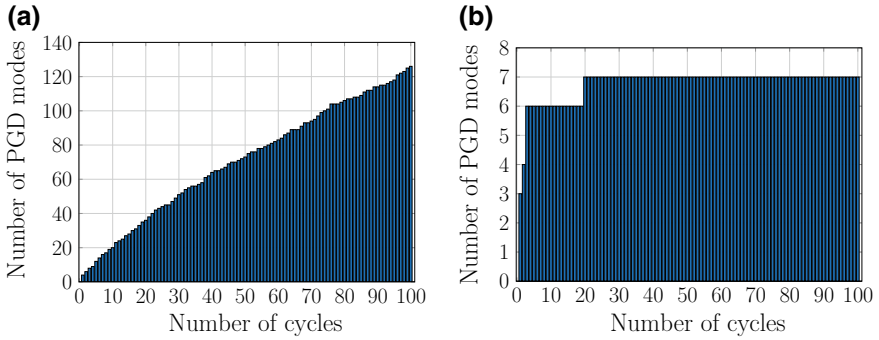

Fig. 8 ROB size using different orthonormalisation algorithms. **a** Number of PGD modes using GS; **b** number of PGD modes using e(R)SVD

10^{-5} mm and a time period $T = 10$ s. The convergence criterion is considered to be 10^{-4}.

The growth of the ROB size using GS and SVD algorithms is depicted in Fig. 8. The usage of a GS algorithm allows the ROB to grow fast and contain 126 modes while e(R)SVD algorithms do not allow this size to exceed seven modes.

Due to the smaller sized ROB in case of the SVD schemes, the computational requirements, in terms of time, to update of the temporal functions is significantly decreased when compared with the GS case; a profiler summary is provided in Fig. 9.

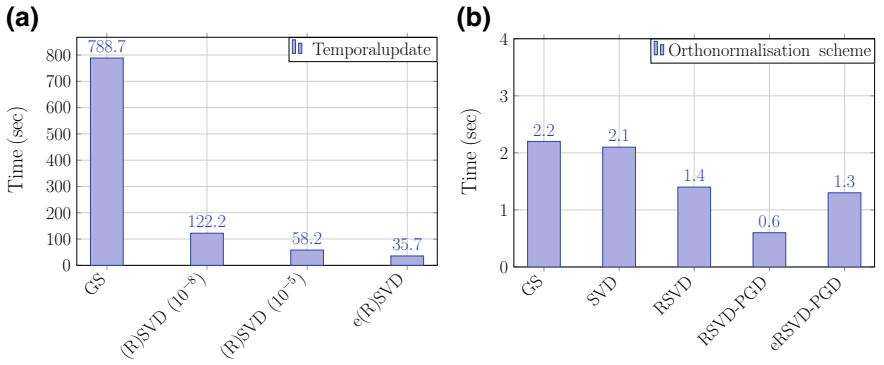

Fig. 9 The required time to perform the temporal update and the orthonormalisation steps. **a** Timing of the temporal functions update; **b** timing of orthonormalisation schemes

6 Conclusions

A semi-incremental scheme to perform damage computations in case of variable amplitude and frequency loading scenarios has been presented in this work along with an optimal proper generalised decomposition. The proposed scheme tackles variable loadings in a flexible manner, where the temporal domain is divided into intervals which allow for more data recycling possibilities between these intervals instead of seeking a solution of the whole time history. The temporal modes had to be treated carefully to ensure providing a "good" initial guess when carrying out simulations over new intervals. The flexibility and optimality of the low-rank approximation have been illustrated through examples with a varying number of degrees of freedom. The optimality of the proper generalised decomposition is ensured throughout an orthonormalisation scheme that exploits the fact that the quantity of interest is sought in an outer-product format; this optimality of the ROB reduces the basis enrichment to a minimum.

Acknowledgements This research was funded by the German Research Foundation/Deutsche Forschungsgemeinschaft (DFG) through the International Research Training Group (IRTG) 1627.

References

1. Alameddin, S., Fau, A., Néron, D., Ladevèze, P., & Nackenhorst, U. (2019). Toward optimality of proper generalised decomposition bases. *Mathematical Computational Applications*, *24*(1), 30.
2. Lemaitre, J., & Desmorat, R. (2005). *Engineering damage mechanics*. Berlin, Heidelberg: Springer.
3. Sobczyk, K., & Spencer, B, Jr. (1992). *Random fatigue: From data to theory*. Cambridge: Academic Press.

4. Wang, Q., Berard, J., Rathery, S., & Bathias, C. (1999). High-cycle fatigue crack initiation and propagation behaviour of high-strength spring steel wires. *Fatigue and Fracture of Engineering Materials and Structure*, *22*(8), 673–677.
5. Deng, G., Tu, S., Zhang, X., Wang, Q., & Qin, C. (2015). Grain size effect on the small fatigue crack initiation and growth mechanisms of nickel-based superalloy GH4169. *Engineering Fracture Mechanics*, *134*, 433–450.
6. Schijve, J., & Yarema, S. (2003). Fatigue of structures and materials in the 20th century and the state of the art. *Materials Science*, *39*(3), 307–333.
7. Schijve, J. (2001). *Fatigue of structures and materials*. Berlin: Springer.
8. F. Chinesta, A. Huerta, G. Rozza, K. Willcox. (2018). *Encyclopedia of computational mechanics* (Vol. 3, chapter Model Redu). Wiley, New York.
9. Giacoma, A., Dureisseix, D., Gravouil, A., & Rochette, M. (2015). Toward an optimal a priori reduced basis strategy for frictional contact problems with LATIN solver. *Computer Methods in Applied Mechanics and Engineering*, *283*, 1357–1381.
10. Bhattacharyya, M., Fau, A., Nackenhorst, U., Néron, D., & Ladevèze, P. (2018). A LATIN-based model reduction approach for the simulation of cycling damage. *Computer Mechanics*, *62*(4), 725–743.
11. Niroomandi, S., González, D., Alfaro, I., Bordeu, F., Leygue, A., Cueto, E., et al. (2013). Real-time simulation of biological soft tissues: A PGD approach. *International Journal for Numerical Methods in Biomedical Engineering*, *29*(5), 586–600.
12. Heyberger, C., Boucard, P., & Néron, D. (2013). A rational strategy for the resolution of parametrized problems in the PGD framework. *Computer Methods in Applied Mechanics and Engineering*, *259*, 40–49.
13. Bhattacharyya, M., Fau, A., Nackenhorst, U., Néron, D., & Ladevèze, P. (2018). A multi-temporal scale model reduction approach for the computation of fatigue damage. *Computer Methods in Applied Mechanics and Engineering*, *340*, 630–656.
14. Cline, A., & Dhillon, I. (2013). *Computation of the singular value decomposition*. Boca Raton: CRC Press.
15. Kerfriden, P., Gosselet, P., Adhikari, S., & Bordas, S. (2011). Bridging proper orthogonal decomposition methods and augmented Newton-Krylov algorithms: An adaptive model order reduction for highly nonlinear mechanical problems. *Computer Methods in Applied Mechanics and Engineering*, *200*(5–8), 850–866.
16. Chinesta, F., & Ladevèze, P. (2014). Separated rrepresentations and PGD-based model reduction. In *CISM International Centre for Mechanical Sciences* (Vol. 554). Vienna: Springer.
17. Nasri, M., Robert, C., Ammar, A., El Arem, S., & Morel, F. (2018). Proper generalized decomposition (PGD) for the numerical simulation of polycrystalline aggregates under cyclic loading. *Comptes Rendus Mécanique*, *346*(2), 132–151.
18. El Halabi, F., González, D., Sanz-Herrera, J., & Doblaré, M. (2016). A PGD-based multiscale formulation for non-linear solid mechanics under small deformations. *Computer Methods in Applied Mechanics and Engineering*, *305*, 806–826.
19. Ladevèze, P. (1999). *Nonlinear computational structural mechanics*., Mechanical Engineering Series New York: Springer.
20. Ladevèze, P. (2016). On reduced models in nonlinear solid mechanics. *European Journal of Mechanics—A/Solids*, *60*, 227–237.
21. Bhattacharyya, M. (2018). *A model reduction technique in space and time for fatigue simulation* (Ph.D. thesis). Leibniz Universität Hannover, Université Paris Saclay.
22. Holzapfel, G. (2000). *Nonlinear solid mechanics: A continuum approach for engineering*. New York: Wiley.
23. Lemaitre, J. (1992). *A course on damage mechanics*. Berlin, Heidelberg: Springer.
24. Cueto, E., González, D., & Alfaro, I. (2016). *Proper generalized decompositions*. Springer International Publishing.
25. Chinesta, F., Keunings, R., & Leygue, A. (2014). *The proper generalized decomposition for advanced numerical simulations: A primer*. Springer International Publishing.

26. Gellin, S., & Pitarresi, J. (1988). Nonlinear analysis using temporal finite elements. *Engineering Analysis*, *5*(3), 126–132.
27. Allix, O., Ladevèze, P., Gilletta, D., & Ohayon, R. (1989). A damage prediction method for composite structures. *International Journal for Numerical Methods in Engineering*, *27*(2), 271–283.
28. Allix, O., & Vidal, P. (2002). A new multi-solution approach suitable for structural identification problems. *Computer Methods in Applied Mechanics and Engineering*, *191*(25–26), 2727–2758.
29. Vandoren, B., De Proft, K., Simone, A., & Sluys, L. (2013). A novel constrained LArge Time INcrement method for modelling quasi-brittle failure. *Computer Methods in Applied Mechanics and Engineering*, *265*, 148–162.
30. Bebendorf, M. (2008). *Hierarchical matrices: A means to efficiently solve elliptic boundary value problems*. Berlin, Heidelberg: Springer.
31. Eckart, C., & Young, G. (1936). The approximation of one matrix by another of lower rank. *Psychometrika*, *1*(3), 211–218.
32. Halko, N., Martinsson, P., & Tropp, J. (2011). Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, *53*(2), 217–288.
33. Golub, G., & Van Loan, C. (1996). *Matrix computations* (3rd ed.). Baltimore: The Johns Hopkins University Press.
34. Bach, C., Ceglia, D., Song, L., & Duddeck, F. (2019). Randomized low-rank approximation methods for projection-based model order reduction of large nonlinear dynamical problems. *International Journal for Numerical Methods in Engineering*, *118*(4), 209–241.