Received: 7 August 2019 | Revised: 11 December 2019 | Accepted: 16 December 2019
DOI: 10.1002/nme.6302

RESEARCH ARTICLE

Two solution strategies to improve the computational performance of sequentially linear analysis for quasi-brittle structures

M. Pari[1] | W. Swart[2] | M.B. van Gijzen[2] | M.A.N. Hendriks[1,3] | J.G. Rots[1]

[1]Faculty of Civil Engineering and Geosciences, Delft University of Technology, GA Delft, The Netherlands
[2]Delft Institute of Applied Mathematics, Delft University of Technology, XE Delft, The Netherlands
[3]Norwegian University of Science and Technology (NTNU), Trondheim, Norway

Correspondence
M. Pari, Faculty of Civil Engineering and Geosciences, Delft University of Technology, P.O. Box 5048, 2600 GA Delft, The Netherlands.
Email: m.pari@tudelft.nl

Funding information
Nederlandse Aardolie Maatschappij B.V. (NAM), UI46268

Summary
Sequentially linear analysis (SLA), an event-by-event procedure for finite element (FE) simulation of quasi-brittle materials, is based on sequentially identifying a critical integration point in the FE model, to reduce its strength and stiffness, and the corresponding critical load multiplier (λ_{crit}), to scale the linear analysis results. In this article, two strategies are proposed to efficiently reuse previous stiffness matrix factorisations and their corresponding solutions in subsequent linear analyses, since the global system of linear equations representing the FE model changes only locally. The first is based on a direct solution method in combination with the Woodbury matrix identity, to compute the inverse of a low-rank corrected stiffness matrix relatively cheaply. The second is a variation of the traditional incomplete LU preconditioned conjugate gradient method, wherein the preconditioner is the complete factorisation of a previous analysis step's stiffness matrix. For both the approaches, optimal points at which the factorisation is recomputed are determined such that the total analysis time is minimised. Comparison and validation against a traditional parallel direct sparse solver, with regard to a two-dimensional (2D) and three-dimensional (3D) benchmark study, illustrates the improved performance of the Woodbury-based direct solver over its counterparts, especially for large 3D problems.

KEYWORDS
direct linear solver, iterative linear solver, low-rank matrix correction, nonlinear finite element analysis, sequentially linear analysis

1 | INTRODUCTION

Sequentially linear analysis (SLA), wherein the nonlinear response of a structure in a displacement-based finite element (FE) framework is approximated as a series of linear analyses*, has in the past few years been improved to enable

*Unlike NLFEA which is considered as *one analysis* containing several *steps*, SLA comprises several linear analysis which are referred herein interchangeably as *'analysis steps'* or *'steps'* as such.

structural-level applications in the civil engineering field. The procedure involves discretising the softening constitutive law with negative tangent stiffness into stepwise decreasing positive secant stiffness branches, and performing one linear analysis at a time to identify the critical stress point in the FE model and the corresponding load multiplier (λ_{crit}). Subsequently, the strength and stiffness of the critical point are reduced, and the stresses and strains of the FE model are scaled with the critical load multiplier.[1-3] The procedure is robust due to the combined use of positive secant stiffnesses and the event-by-event approach, thereby avoiding typical ill-conditioning problems encountered in Nonlinear FE analysis (NLFEA) wherein multiple integration points enter into softening simultaneously. It has been a proven alternative to traditional incremental-iterative methods for NLFEA of quasi-brittle structures. SLA, thus far, has had contributions in the contexts of mesh-objectivity,[3,4] saw-tooth laws for extremely brittle materials with snap-back at constitutive level like glass,[5] extension to non-proportional loading situations,[6-10] stepwise secant Coulomb friction laws,[4] creep-induced cracking,[11] combined incremental-total approaches like Non-Iterative Energy-based Method (NIEM) and the automatic method,[8] combining SLA with a crack tracking technique[12] and non-proportional loading strategies for three-dimensional (3D) stress states. [13]

Despite active contributions to this topic, the computational performance of SLA remains not very conducive to practical applications as has been pointed out previously.[4,14-16] It is the event-by-event nature of the SLA approach, which on the one hand instills robustness, contrarily makes the procedure computationally intensive. However, since only one element is effectively damaged at a time, the system of linear equations to be solved only changes locally between these analyses. Traditional direct solution techniques do not exploit this property and calculate a rather expensive stiffness matrix factorisation every linear analysis, resulting in high computational times per analysis step. Additionally, the need for a high number of linear analyses, to bring about an equivalent nonlinear response as in the traditional approaches, compounds the total analysis time. Inspired by remarks made in, for example[17,18] this motivated the need for a tailor-made solver for SLA. To address this issue, and efficiently make use of previous stiffness matrix factorisations and solutions, two solution strategies are proposed in this article. Alternative methods combining a traditional incremental-iterative technique and the total approach of SLA are also available in the literature addressing the need for a practical alternative,[8] but the focus of this work is to solely improve the performance of SLA with regards to solving the system of linear equations.

To begin with, the constitutive framework and work flow of SLA are briefed upon in Section 2. In Section 3, the aforementioned solution strategies are explained alongside the corresponding restarting ideas. Firstly, an adapted direct solution technique based on the Woodbury matrix identity is proposed. This identity, the generalisation of the Sherman-Morrison formula (to find the inverse of a rank-1 corrected matrix) to a rank-r correction, allows for cheaper numerical computation of the inverse of a low-rank corrected matrix by avoiding the matrix factorisation every analysis step. The old factorisation can be reused with some additional matrix and vector manipulations in order to solve a significantly smaller linear system of equations relatively efficiently. The optimal point of restarting, to start off again with a new factorisation is also deduced. Secondly, an improved preconditioner for the conjugate gradient (CG) method[19] is proposed. Instead of an incomplete LU factorisation (ILU) as a preconditioner, which is more commonly used for solving large systems of equations pertaining to structural applications, the complete factorisation (LU) of a previous analysis step is used as a preconditioner which reduces the number of required CG iterations significantly. The point at which too many CG iterations are required and a new factorisation is necessary, is determined using a restarting strategy similar to that of the first method. Subsequently, in Section 4 the proposed solution strategies are compared against a widely used parallel direct sparse solver (PARDISO),[19] from a performance perspective, using two real-life benchmarks. The first benchmark is that of a masonry wall tested for a quasi-static lateral load in combination with an overburden load[20] and the second is a reinforced concrete (RC) slab tested for a concentrated shear load, along with axial loads at the lateral faces.[21] From the numerical studies, it follows that both the proposed methods perform significantly better than the direct solution method, especially for large 3D problems. Results from the sensitivity studies performed for problem sizes and for the number of steps used to discretise the constitutive model, are also detailed in this section. Finally, Section 5 summarises the main findings of the presented work, ongoing investigations and the directions to future work.

2 | SLA: AN OVERVIEW

2.1 | Discretisation of the softening constitutive model

The crux of the method is in discretising the uniaxial softening constitutive relation into an equivalent stepwise secant material law, also known as the saw-tooth law. In principle, the material law is described as a series of successively reducing secant stiffnesses, starting from the initial elastic branch with the original Young's modulus of the material (E_0). Whenever there is breach of the stress limit in an integration point and it becomes the most critical in the FE model, the next secant relation with reduced strength and stiffness properties takes over from its previous secant branch. This process of reducing the stiffness upon attaining a stress limit is repeated until the stiffness of the structure has vanished, which corresponds to a state of complete damage. Constant stiffness/strength decrements, the ripple band method, and the improved ripple band methods are some of the approaches in use for saw-tooth approximations of typical tension and compression softening curves.[4] Figure 1A depicts the ripple bandwidth type saw-tooth law for a linear tension softening relation.

These uniaxial saw-tooth laws are used in the orthotropic fixed smeared crack framework, wherein as soon as the principal stress violates the allowable strength at an integration point, the isotropic stress strain relation $\boldsymbol{\sigma} = \boldsymbol{D\epsilon}$ transforms into an orthotropic relation as $\boldsymbol{\sigma}_{\mathbf{nst}} = \boldsymbol{D}_{\mathbf{nst}}\boldsymbol{\epsilon}_{\mathbf{nst}}$ with the *nst* cracked coordinate system. The primary principal stress direction's Young's modulus and strength are damaged according to the uniaxial saw-tooth law. In the event that normal stresses in the tangential directions (secondary or tertiary) violate the corresponding allowable strengths, caused by stress rotations or redistribution of stresses or application of another load non-proportionaly, damage is introduced in those directions similarly. So every integration point essentially requires two uniaxial saw-tooth laws each for tension and compression in the two-dimensional (2D) stress state and three in the case of a 3D stress state. This aside, the shear behaviour in the fixed cracking model is represented using a variable shear retention function that reduces with increasing damage in normal directions of the cracked plane.[22] Also, the Poisson's ratio is reduced at the same rate as the associated Young's modulus. For further information on the constitutive model used in SLA, the reader is referred to References 3,4,13,23.

2.2 | Work flow

Defining the saw-tooth laws is the first step in SLA's workflow, as shown in Figure 1B. Thereafter, the FE model is loaded by a unit value of the load to be actually applied and a linear analysis is performed. A load factor can be calculated for each integration point as the ratio of the allowable strength to that of the governing stress (considering the principal stresses for damage initiation in a smeared fixed crack approach) as shown in the following.

$$\lambda^j_{\text{crit},i} = \frac{f^j_i}{\sigma^j_{\text{gov},i}}, \tag{1}$$

where i and j denote an integration point number[†] and the analysis step, respectively, $\sigma^j_{\text{gov},i}$ is the governing stress component for integration point i, f^j_i is the peak stress limit as defined by the current secant branch of the saw-tooth law and $\lambda^j_{\text{crit},i}$ the associated load multiplier. To ensure that only one integration point reaches its peak stress limit, the linear analysis is scaled with the minimum of all load factors which is referred to as the critical load factor and is defined as

$$\lambda^j_{\text{crit}} = \min_i \left(\lambda^j_{\text{crit},i}\right) \quad \forall \quad \lambda^j_{\text{crit},i} > 0. \tag{2}$$

The strength f_t and stiffness E_0 of this critical integration point is then reduced based on the saw-tooth laws, with the discretisation factor p, as shown in Figure 1A. Eventually, the results of the linear analysis: the stresses, strains and displacements, are scaled using the critical load multiplier.

The workflow of SLA under non-proportional loading conditions is not as straightforward as elaborated above. Real-life loading conditions are rather complex, where the rate of change of all loads in not the same. The simplest and the most common situation is when there are constant loads like dead loads, precompression, overburden etc. on the structure, and subsequently variable loads like earthquake or wind loads act. In such situations, each global stress component is expressed as a superposition of the stresses due to the constant loads and the scaled variable loads. The load multiplier is then deduced for undamaged integration points by limiting the principal stresses to the allowable strength. Already damaged integration points yield load multipliers per direction of the orthogonal cracked system. The critical load multiplier is then identified based on a constrained maximisation approach in combination with a double-load multiplier

[†] *i* denotes an integration point for an undamaged situation and alternatively upon damage, denotes events corresponding to tension / compression failure criteria along the two or three fixed damage directions. depending on the stress state

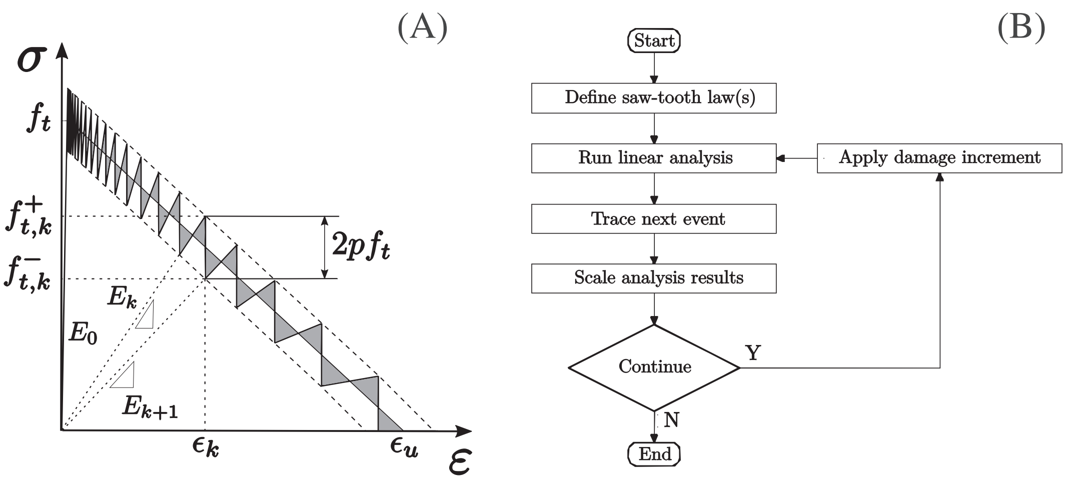


FIGURE 1 (A) Saw tooth constitutive laws—linear tension softening; (B) the work flow for proportional loading conditions

strategy.[4] Damage initiation involves solving quadratic (2D) or cubic equations (3D) of the load multiplier, as the stress state may be, while damage propagation involves solving linear equations. However, the subsequent process of applying damage and scaling results is analogous to that in the proportional loading case. For further information on the workflow for non-proportional loading situations, the reader is referred to References 4,6,13.

3 | METHODOLOGIES

3.1 | Motivation

The traditional direct solution method to solve the system of linear equations involves an expensive matrix factorisation of the stiffness matrix K and calculates the displacements $\mathbf{u}$ by forward and backward substitution.

$$K\mathbf{u} = \mathbf{f}, \tag{3}$$

Common factorisation techniques like the LU and Cholesky decomposition methods, become expensive if they have to be computed in every analysis step. As a result, for increasing problem sizes, solving the system of equations becomes the bottleneck for SLA and is illustrated by the CPU-time measurements for a 2D shear wall case-study reported in Reference 24, see Figure 2 for the results. The system of equations undergoes a low-rank correction of the global stiffness matrix per analysis step in SLA and therefore, there is a need for efficient solution strategies that exploit this feature, two of which are outlined in the following sections.

3.2 | Direct solver using Woodbury matrix identity

The inverse of a rank-1 corrected matrix $(A + \mathbf{uv}^T)$, subject to the inverse A^{-1} being known a priori, can be computed relatively easily using the well-known *Sherman-Morrison formula*[19] as against having to perform the inverse operation altogether anew. The rank correction for practical applications in FE analysis using the SLA, however, is generally of a higher order.

To this end, the *Woodbury matrix identity*,[19] which is a generalisation of the *Sherman-Morrison formula* for a rank-r correction of a matrix, is more suitable. The identity states that for matrices $A \in \mathbb{R}^{N\times N}, U \in \mathbb{R}^{N\times r}, C \in \mathbb{R}^{r\times r}, V \in \mathbb{R}^{r\times N}$, assuming A and C are invertible, the inverse of a low-rank corrected matrix is defined as : $(A + UCV)^{-1} = A^{-1} - A^{-1}U\left(C^{-1} + VA^{-1}U\right)^{-1}VA^{-1}$. Substituting $r = 1$, it follows directly that the Woodbury identity reduces to the Sherman-Morrison formula. In SLA, additionally, the low-rank correction is symmetric and therefore UCV can be written as UCU^T with C as a symmetric matrix. In this case, the expression simplifies to the following:

$$\left(A + UCU^T\right)^{-1} = A^{-1} - A^{-1}U\left(C^{-1} + U^TA^{-1}U\right)^{-1}U^TA^{-1}. \tag{4}$$

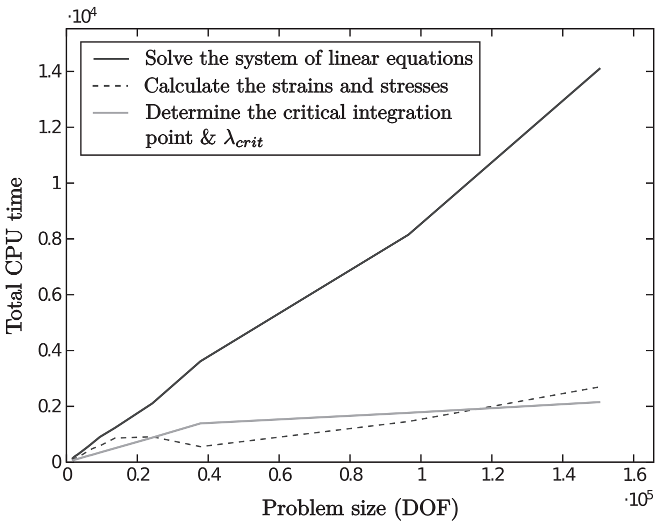


FIGURE 2 Total central processing unit (CPU) time (in seconds) of the most dominant processes of sequentially linear analysis in relation to the problem size (number of degrees of freedom (DOFs)) of a 2D shear wall case-study[24]

Assuming an element e_i is damaged in the nth analysis step, the low-rank corrected system stiffness matrix $K^{(n+1)}$, for the subsequent analysis step can be written as

$$\begin{aligned} K^{(n+1)} &= K^{(n)} + T_{e_i}\left(K_{e_i}^{(n+1)} - K_{e_i}^{(n)}\right)T_{e_i}^T \\ &:= K^{(n)} + T_{e_i}D_{e_i}^{(n)}T_{e_i}^T, \end{aligned} \tag{5}$$

where $D_{e_i}^{(n)}$ is the update to the stiffness matrix of element e_i and T_{e_i} is the transformation matrix which maps the local numbering of the element to the global numbering of the FE model. It is to be noted that the subscripts and superscripts refer to element numbers and the analysis step, respectively. Constructing the eigendecomposition of $D_{e_i}^{(n)}$ and substituting it in Equation (5) we obtain

$$K^{(n+1)} = K^{(n)} + T_{e_i}Q_{e_i}^{(n)}\Lambda_{e_i}^{(n)}\left(Q_{e_i}^{(n)}\right)^T T_{e_i}^T \tag{6}$$

$$\begin{aligned} &= K^{(n)} + \left(T_{e_i}Q_{e_i}^{(n)}\right)\Lambda_{e_i}^{(n)}\left(T_{e_i}Q_{e_i}^{(n)}\right)^T \\ &:= K^{(n)} + U^{(n)}C^{(n)}U^{(n)^T} \end{aligned} \tag{7}$$

wherein the matrix $C^{(n)} = \Lambda_{e_i}^{(n)}$ is a diagonal matrix whose elements are the eigenvalues of $D_{e_i}^{(n)}$, $Q_{e_i}^{(n)}$ contains the corresponding eigenvectors and $U^{(n)} = T_{e_i}Q_{e_i}^{(n)}$. The aforementioned eigendecomposition only considers sufficiently large eigenvalues corresponding to dominant features of the applied damage increment in an SLA step. The basis for this choice and its effect on the convergence of the solution is elaborated upon in Section 3.4. It is clear that the rank in every analysis step increases by the number of sufficiently large eigenvalues of the eigendecomposition of $D_{e_i}^{(n)}$, which regardless of the type of analysis (2D or 3D) is at most d, where d is the number of degrees of freedom (DOF) of the element e_i. Rewriting in a suitable form for Woodbury identity, Equation (7) is defined recursively in terms of the initial stiffness matrix $K^{(0)}$ which yields

$$\begin{aligned} K^{(n+1)} &= K^{(0)} + \sum_{j=1}^{n} U^{(j)}C^{(j)}U^{(j)^T} \\ &= K^{(0)} + \begin{bmatrix} U^{(1)} & \dots & U^{(n)} \end{bmatrix} \begin{bmatrix} C^{(1)} & & \\ & \ddots & \\ & & C^{(n)} \end{bmatrix} \begin{bmatrix} U^{(1)^T} \\ \vdots \\ U^{(n)^T} \end{bmatrix} \\ &:= K^{(0)} + U_n C_n U_n^T. \end{aligned} \tag{8}$$

Now, Equation (8) is of the form as required by Equation (4). Once the factorisation of $K^{(0)}$ is known and the above setup is performed, the solution to the system of equations $K^{(n+1)}\mathbf{u} = \mathbf{f}$ of the $(n+1)^{th}$ linear analysis step can be calculated by performing the steps in Algorithm 1. In summary, Woodbury identity helps achieve a cheaper computation of the inverse of a low-rank corrected matrix, by avoiding the calculation of an expensive new factorisation every analysis step. Thus, it enables the reuse of an old factorisation and subsequently the solution is obtained with additional matrix and vector multiplications. A significantly smaller system of equations is solved for effectively in step 5 of Algorithm 1, as opposed to the direct solution method.

Algorithm 1. Direct solution using the Woodbury matrix identity

```
1: Solve the system $K^{(0)}\mathbf{x} = \mathbf{f}$, for $\mathbf{x}$ by using the known factorisation of $K^{(0)}$
2: Solve the system $K^{(0)}Z = U_n$, for $Z$ by using the known factorisation of $K^{(0)}$
3: Calculate $E = C_n^{-1} + U_n^T Z$
4: Calculate $\mathbf{y} = U_n^T \mathbf{x}$
5: Solve the system $E\mathbf{z} = \mathbf{y}$, for $\mathbf{z}$ by calculating a factorisation of $E$ and applying subsequent forward and backward
   substitutions.
6: Solution to the system of equations: $\mathbf{u} = \mathbf{x} - Z\mathbf{z}$
```

3.3 | Preconditioned conjugate-gradient iterative solver

Krylov subspace methods, which belong to an iterative class of solution methods, in contrast to direct solution methods generate a sequence of approximate solutions $\mathbf{u}_i$ to Equation (3). This is done by solving a minimisation problem over the subspace $\mathcal{K}^k$ which contains the solution and is called the Krylov subspace as defined below in Equation (9), where $\mathbf{r}$ is the residual vector and $\mathbf{r_0} = \mathbf{f} - \mathbf{Ku_0}$ with $\mathbf{u_0}$ as the initial guess, and k is the number of iterations.

$$\mathcal{K}^k(K, \mathbf{r}_0) = \text{span}\left\{\mathbf{r}_0, K\mathbf{r}_0, \dots, K^{k-1}\mathbf{r}_0\right\} \tag{9}$$

The stiffness matrix is symmetric and positive definite (SPD) for the problems solved using SLA and for such cases, the CG method is the Krylov subspace method of choice. After N iterations the Krylov subspace spans $\mathbb{R}^N$ and therefore, CG terminates (in exact arithmetic) at the exact solution after at most N iterations. A stricter error bound using eigenvalues is also well known, which states that if K (or $P^{-1}K$ for a preconditioned problem) has ρ distinct eigenvalues, convergence is guaranteed in at most ρ iterations.[19] The extreme eigenvalues of K influence the convergence speed of *CG*, which in turn can be improved using preconditioning to obtain more favourable eigenvalues.

In the preconditioned CG (PCG) method, for a matrix P which is assumed to be non-singular, $P^{-1}K\mathbf{u} = P^{-1}\mathbf{f}$ is solved for, and is shown in Algorithm 2. Since in every iteration the linear system $P\mathbf{z}_{k+1} = \mathbf{r}_{k+1}$ is solved for, P should be chosen such that operations with P^{-1} are cheap to perform. Furthermore, the choice of the preconditioner must ensure that the eigenvalues of $P^{-1}K$ are clustered for a faster rate of convergence. Several choices for the preconditioner P exist such as the extremes $P = I$ and $P = K$, or the intermediate ILU factorisation.

Algorithm 2. PCG algorithm

```
1: Set $\mathbf{r}_0 = \mathbf{f} - K\mathbf{u}_0$, $\mathbf{z}_0 = P^{-1}\mathbf{r}_0$, $\mathbf{p}_0 = \mathbf{z}_0$.
2: for $k = 0, 1, \dots$ until convergence do
3:     $\alpha_k = \mathbf{r}_k^T\mathbf{z}_k / \mathbf{p}_k^T K\mathbf{p}_k$
4:     $\mathbf{u}_{k+1} = \mathbf{u}_k + \alpha_k\mathbf{p}_k$
5:     $\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k K\mathbf{p}_k$
6:     $\mathbf{z}_{k+1} = P^{-1}\mathbf{r}_{k+1}$
7:     $\beta_k = \mathbf{r}_{k+1}^T\mathbf{z}_{k+1} / \mathbf{r}_k^T\mathbf{z}_k$
8:     $\mathbf{p}_{k+1} = \mathbf{z}_{k+1} + \beta_k\mathbf{p}_k$
9: end for
```

Due to the event-by-event strategy of *SLA*, the choice $P = K^{(0)}$ would be appropriate especially when the factorisation is not performed every analysis, in order to have a balance between a cheap computation of P and a reasonable rate of convergence. This suggests the use of the complete factorisation of $K^{(0)}$ as a preconditioner. Taking $P = K^{(0)}$, it follows that in the first analysis step $P^{-1}K^{(0)} = I$, and the solution method basically becomes a direct solution method. However, subsequent analysis steps require considerably lesser iterations due to the event-by-event nature of SLA. That is, the first subsequent system needs at most $(r_1 + 1)$ iterations, with r_1 equal to the rank of the first update. The second system needs at most $(r_1 + r_2 + 1)$ iterations, since $K^{(2)}$ differs form $K^{(0)}$ by at most a rank $r_1 + r_2$ update. This argument can be repeated, which implies that after n SLA steps, at most $\left(\sum_{i=1}^{n} r_i + 1\right)$ CG-iterations are needed. The iterative scheme is repeated until the approximate solution is sufficiently converged, determined by a stopping criterion, one of which is shown below for some ϵ, where $\mathbf{r}$ is the residual vector.

$$\frac{\|\mathbf{r_k}\|}{\|\mathbf{f}\|} \leq \epsilon \tag{10}$$

Therefore, the optimal point at which a new factorisation is calculated should be determined such that the total computing time is minimised. In such an iterative approach, instead of factorisation, the matrix K is only involved in matrix-vector multiplications and the solution to Equation (3) is determined using inner-products, vector updates, scalar-vector and matrix-vector products, and back- and forward substitutions with the factors of $K^{(0)}$ (the most expensive operations). Krylov subspace methods, in general, require a relatively small amount of memory to solve the problem compared to direct solution methods, however, in SLA's context, more memory is required because of the factorisation being used as preconditioner.

3.4 | Restarting strategies for both approaches

The solution strategies presented in Sections 3.2 & 3.3 are similar to the end that both require an expensive factorisation step followed by a series of significantly faster steps. While solving large linear systems using these strategies, two parameters can be tuned. Firstly, the number of sufficiently large eigenvalues to be considered for the eigendecomposition of the update to the critical element stiffness matrix in Equation (6) has to be chosen, that is, a decision has to be made to find a balance between performance and accuracy. Secondly, it is also possible to restart which implies that a new factorisation of the stiffness matrix has to be computed resulting in the rank being set back to 0. The penalty in restarting is in having to recompute a costly matrix decomposition, while, on the other hand, the following analysis steps would be considerably cheaper. The restarting point, therefore, has be to be determined such that these effects are balanced.

Eigenvalue ratio. Numerically calculated eigenvalues of the update to critical element's stiffness matrix, using some iterative scheme, could contain rounding errors which may result in non-zero values and therefore influence the eventual results. To address this, the absolute value of all eigenvalues during every analysis step was compared to the largest eigenvalue as a ratio, the eigenvalue threshold $\epsilon = \lambda_i / \lambda_{\max}$, which in turn helped control the choice of dominant eigenvalues differing from the largest value by a certain order of magnitude. Parametric studies were performed on several test problems, without restarting as proposed in Section 3.2, to solely analyse the influence of the choice of an eigenvalue ratio on the accuracy of the solution. Thus, isolating the effect of only the eigenvalue threshold, a choice of $\varepsilon > 10^{-10}$ was made.[24] In general, the solution residuals were observed to increase faster with the Woodbury-based solution (without restarting) than those of the direct solution method, which is due to the rounding errors resulting from the numerous intermediate matrix and vector manipulations involved in the former approach. Nevertheless, the residuals of the Woodbury solution were in reasonable agreement to the direct solution method in terms of accuracy (one order of magnitude difference), especially considering the fact that restarting was not yet used.

Restarting strategy. After deciding on an eigenvalue threshold based on the performance of the method without loss of any accuracy (residuals differing by one order of magnitude), the next step of determining a restarting strategy was carried out. The optimal point when a new factorisation has to be recalculated for both Woodbury-based and PCG methods, such that the total analysis time is minimised, could possibly be determined using two approaches. Owing to the inherent similarity in the proposed Woodbury-based and PCG methods, the restarting ideas are presented using the Woodbury method as reference.

Firstly, a rank-correction-based approach was considered. Herein, the theoretical cost estimates for both the direct solution method and the Woodbury identity-based approach are derived using the theoretical flop counts for all necessary substeps within a linear analysis, which depends on the rank correction r, the lower and upper bandwidths of the stiffness matrix p, q and also its size $N \times N$. This rank-based optimal restarting strategy determines the point of restarting by minimising the cost function with respect to the rank r. However, since the approach relies heavily on the estimated bandwidths, which cannot be efficiently deduced for complex geometries, and the fact that the direct solution method (PARDISO) being considered here has a fill-in minimising reordering scheme (which does not necessarily minimise the bandwidth), it is highly unlikely that the prediction for the restarting point would be optimal indeed. Detailed information on the rank-based restarting strategy and the associated cost functions can be found in Reference.[24]

Secondly, a time-estimation-based approach was deduced. The computing times of the analysis steps are measured and an estimate is made for the expected total analysis time. The measured time for a direct solution analysis step (t_d) includes those for factorisation, back- and forward substitutions. The subsequent n_r analysis steps, where n_r denotes the next restarting analysis step, yield the solution using Woodbury method in time $t_w(i)$ where $i = 1, \dots, n_r$. There are two key assumptions to this restarting strategy:

- The maximum number of analysis steps m is known a priori.
- The total computing time is composed of a sequence of repeating measured patterns (times) after every restart.

Restarting after the n_r^{th} analysis step and assuming that the measured sequence of times $\{t_d, t_w(1), \dots, t_w(n_r)\}$ repeats until the end of the analysis, the total computing time of the analysis can be computed as shown below, where the second term is premultiplied by a typical indicator function (to adjust for the remaining analysis steps after the last of several restarts):

$$t(n_r) = \left\lfloor \frac{m}{n_r + 1} \right\rfloor \cdot \left(t_d + \sum_{i=1}^{n_r} t_w(i) \right) + \mathbb{1}_{\left\{m - \left\lfloor \frac{m}{n_r+1} \right\rfloor \cdot (n_r+1) \neq 0\right\}}(n_r) \cdot \left(t_d + \sum_{j=1}^{m - \left\lfloor \frac{m}{n_r+1} \right\rfloor \cdot (n_r+1) - 1} t_w(j) \right) \tag{11}$$

In order to corroborate the assumptions to be realistic, simple performance studies were carried out for a coarse mesh of a 3D RC slab problem, reported.[24] This RC slab subject to concentrated shear load was previously simulated using SLA[4] under proportional loading conditions. Figure 3A shows the patterns of elapsed time per linear analysis step/event for the standard PARDISO and, Woodbury Identity-based direct solvers (with and without the restarts) for 80 steps. Figure 3B shows the total time taken for these three cases up to 80 steps and emphasises the need for restarting. The maximum number of analysis steps m does not have any significant effect on the restarting point derived by optimising Equation (11) and this was also observed as closely spaced restarting points for varying values of m, as shown in Figure 4A. This is because m just appears as a multiplicative constant in Equation (11) and for very large cases, one could ignore the second term in Equation (11).

Furthermore, the problem was solved with restarting points a few steps before and after the optimal ones, for a finer mesh of the aforementioned case study. Figure 4B shows the total analysis times (for 10000 steps in all) for these three cases referred to as *Earlier, Later*, and *Optimal*. This study confirmed that continuing longer without restarting results in a performance penalty in the last analysis steps (total analysis time = 1289 mins) while restarting earlier results in too

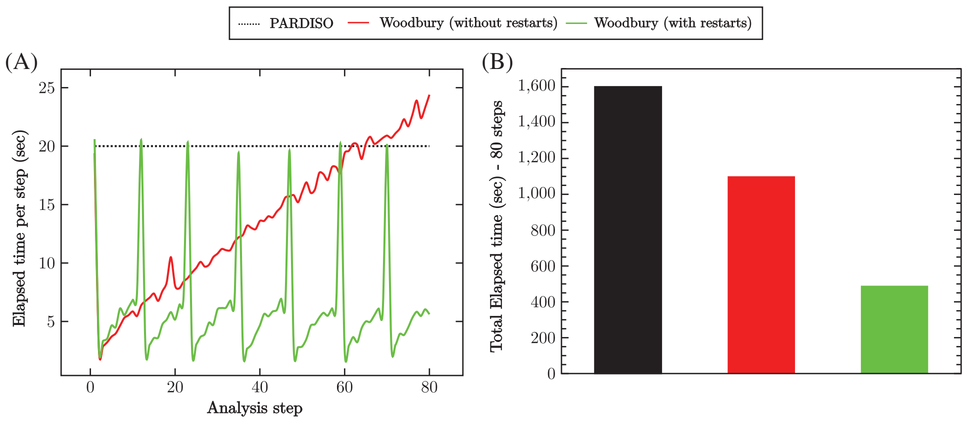


FIGURE 3 (A) Elapsed times per analysis step up to 80 steps of a three-dimensional RC-Slab simulation using the parallel direct sparse solver and Woodbury identity-based direct solution methods, with and without restarting at the optimal point; (B) Total elapsed time for the three cases up to 80 events

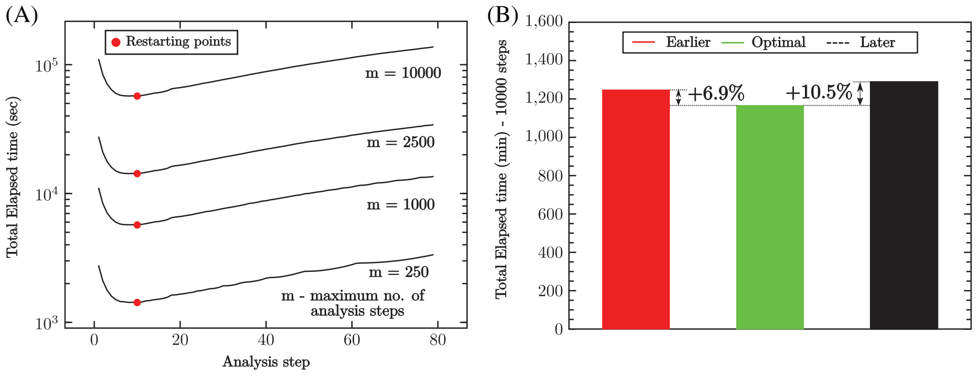


FIGURE 4 (A) Total elapsed times for different values of *m*, that is, the maximum number of analysis steps illustrating the proximity of restarting points and (B) the total elapsed times until the 10000th analysis step, for the three-dimensional reinforced concrete-slab simulation with a finer mesh, comparing the effect of earlier or delayed restarting with respect to the optimal point

many expensive factorisations (total analysis time = 1247 mins) as against the optimal one (total analysis time = 1166 mins). These simple studies validate the aforementioned assumptions and the restarting strategy by itself. In conclusion, the time-estimation-based restarting strategy seems a more reasonable option for the presented solution methods and is therefore used as reference for the validation and parametric studies presented in Section 4. Further information on the time-estimation-based restarting strategy can be found in Reference.[24]

4 | RESULTS AND DISCUSSION

In order to validate the proposed solution strategies, two experimental benchmarks are considered. Firstly, the structural response of both the benchmarks, as simulated using SLA, are briefly touched upon. Thereafter, the computational performance of these reference models (solved with the PARDISO) are compared against those solved with the Woodbury identity-based method and the PCG. Subsequently, parametric studies are presented in Section 4.2.

4.1 | Case studies

4.1.1 | Pushover analysis of a Shear Wall (2D)

The first benchmark considered is that of an unreinforced brick masonry wall, 1.35m x 1.1m in size and clamped along the top and bottom edges, firstly subject to an overburden/precompression of 0.6 MPa followed by a quasi-static lateral load. Although the test is cyclic in nature, the test can be used under monotonic loading as a benchmark for 2D (plane stress) SLA simulations by making qualitative comparisons between the response and the envelope of the experimental curve. Diagonal shear failure was observed in the experiment subsequent to reaching the peak force. Further details about the experiment can be found in Reference 20. The experimental setup and the results of the SLA simulations are shown in Figure 5. Modeling and material parameters are given in Table 1. Good agreement with the force-displacement curves are observed, however, since the focus of this study is more on the performance of the solver, further information on the simulation in terms of the 2D FE model, the agreement between the experimental crack patterns and those from SLA etc, can be found in Reference 25.

4.1.2 | Shear testing of a RC slab (3D)

The second benchmark is that of a RC slab (excluding shear reinforcements), $4\,\text{m} \times 2.6\,\text{m} \times 0.3\,\text{m}$ in size and simply supported on all four sides, firstly subject to an in-plane compressive axial load of 1.5 MPa followed by an out-of-plane concentrated load near one of the line supports. The axial loads were applied by means of 12 in-plane hydraulic jacks, while the concentrated load was applied at a distance of 560 mm from the line support using an out-of-plane hydraulic jack over a loading plate. The failure mechanism begins with flexural cracks that appeared at the bottom face along the transverse and longitudinal reinforcements, followed by cracks due to the two-way shear slab mechanism (punching shear

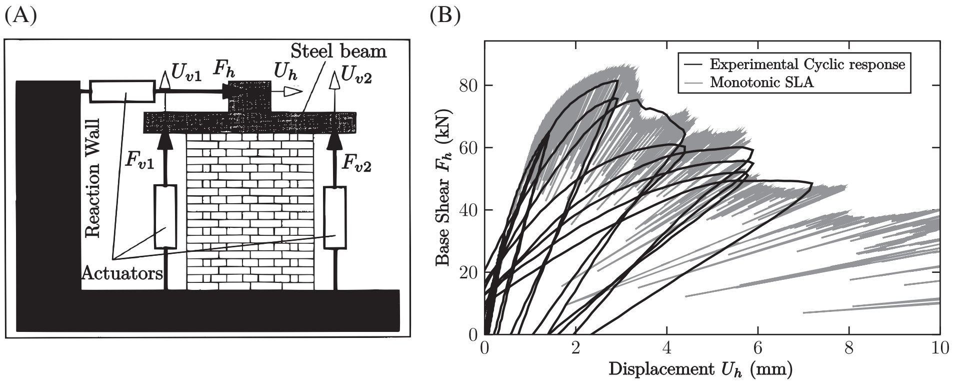


FIGURE 5 (A) Experimental setup of the quasi-static cyclic pushover test[20] and (B) the comparison of the two-dimensional monotonic sequentially linear analysis results to those of the backbone of the cyclic response[25]

TABLE 1 Modeling and material parameters

Material	Parameters	Shear Wall	Reinforced Concrete Slab
Masonry/concrete	Young's Modulus E_o (GPa)	1.491	15.375
	Poison's ratio v_o	0.15	0.15
	Tensile strength f_{tb} (MPa)	0.17	3.63
	Mode I fracture energy G_f^1 (N/mm)	0.1[a]	0.177
	Saw-teeth discretisation factor	0.1	0.15
	Tension softening relation	Linear	Linear
	Number of saw-teeth	32	9
	Compressive strength f_{tb} (N/mm^2)	6.2	33.3
	Compressive fracture energy G_c (N/mm)	40	44.25
	Compressive softening relation	parabolic	parabolic
	Number of saw-teeth	24	12
	Crack bandwidth h (mm)	Element size	Element size
	Shear retention factor β	10^{-2}	10^{-4}
Reinforcements & steel plates	Young's Modulus E_o (GPa)	—	210
	Poison's ratio v_o	—	0.3

[a]For the shear wall case, the bottom and top row of elements of the FEM model in Figure 5 has Mode I fracture energy of G_f^1 0.1 [N/mm] to simulate rocking failure. The rest of the elements are provided $G_f^1 = 0.15$ [N/mm] to mimic the larger dissipation of energy observed in a diagonal shear crack.

failure) with a perimeter crack surrounding the loading area, and eventually the pure shear failure along the line support which was quite brittle. Further information about the experiments pertaining to the experimental setup, material properties, crack patterns etc, can be found in Reference.[21] The experimental setup and the results of the 3D SLA simulations are shown in Figure 6. Reasonable agreement with the force-displacement curves and also the qualitative brittle behavior in terms of the simulated failure mechanism are observed. The experimental peak load is predicted quite well by the SLA simulation but the ductility is underestimated. However, this is not just a feature of SLA but the smeared crack approach in general as has been previously observed in another NLFEA simulation[26] and also using a plasticity-based approach in an explicit ABAQUS simulation.[27] Considering the scope of the study, further information on the simulation in terms of the FE model (3D), the agreement between the experimental crack patterns and those from SLA etc., can be found in Reference.[13]

4.1.3 | Discussion on performance of the solution methods

The central processing unit (CPU) used for this study is an AMD EPYC 7351 processor with 16 cores / 32 threads, with a base clock speed of 2.4 GHz and an all core boost speed of 2.9 GHz. All studies are run on single-threading, unless specified

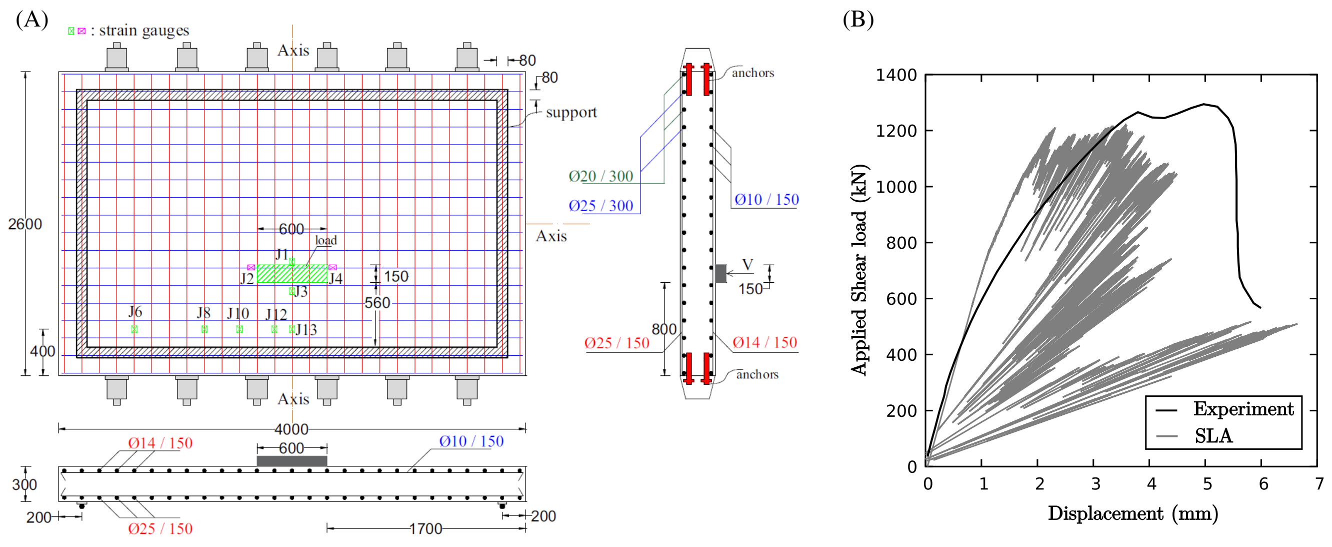


FIGURE 6 (A) Experimental setup of the shear test on reinforced concrete slab: Loading plate, reinforcement layout, and axial load application setup[21] and (B) comparison of the sequentially linear analysis and experimental force displacement curves[13] [Colour figure can be viewed at wileyonlinelibrary.com]

TABLE 2 Contributions of dominant processes to total elapsed analysis times on a single thread—Shear wall and reinforced concrete (RC) slab

Description	**Parallel Direct Sparse Solver**		**Woodbury**		**Preconditioned Conjugate Gradient**	
	Shear Wall	**RC Slab**	**Shear Wall**	**RC Slab**	**Shear Wall**	**RC Slab**
SOLVE - Solve system of linear equations	43.57 %	65.69 %	13.70 %	26.86 %	29.56 %	49.08 %
	(10.07 minutes)	(282.24 minutes)	(2.15 minutes)	(54.15 minutes)	(5.52 minutes)	(145.57 minutes)
STREAC - Calculate strains and stresses from displacement fields	20.15 %	11.49 %	30.42 %	24.67 %	24.95 %	17.12 %
	(4.66 minutes)	(49.39 minutes)	(4.77 minutes)	(49.32 minutes)	(4.69 minutes)	(50.76 minutes)
SLSCAL - Determine critical int. point (IP) and λ_{crit}, update stiffness of IP and scale results	32.54 %	22.01 %	50.06 %	46.98 %	39.98 %	32.49 %
	(7.52 minutes)	(94.58 minutes)	(7.85 minutes)	(94.72 minutes)	(7.51 minutes)	(96.35 minutes)
Total elapsed time	23.12 minutes	429.65 minutes	15.69 minutes	201.63 minutes	18.78 minutes	296.557 minutes

otherwise. CPU time is the exact amount of time spent in processing data by the CPU for a specific process, while the elapsed time refers to the total time taken for the completion of a process, which is the sum of the CPU and I/O times.

Total analysis times in each SLA step are composed of those for several operations like setting up the element stiffness matrices and assembling the global stiffness matrix (referred to as ELMATR hereon); solving the system of equations for unknown displacements (referred to as SOLVE hereon); calculation of stresses and strains from the displacement field (referred to as STREAC hereon); and determining the critical integration point and the load multiplier, scaling stresses and strains and finally updating the stiffness and strength of the critical integration point (referred to as SLSCAL hereon). Times per analysis step for all the aforementioned operations, except SOLVE, would approximately be the same for a simulation run using either the reference PARDISO or the proposed solutions methods presented in Sections 3.2 and 3.3. This is clear from the elapsed times of the Shear wall and RC Slab case studies run on a single-thread of a processor, as shown in Table 2.

Thus, the CPU time per analysis step to solve the system of equations (SOLVE) is chosen as the yardstick to compare the performance of the PARDISO, Woodbury and PCG solvers. The pattern of CPU times for the first 1000 analysis steps

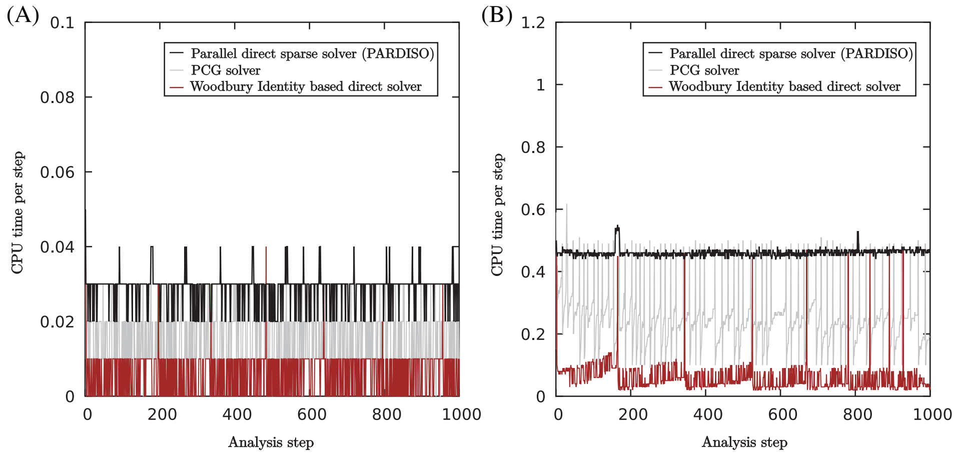


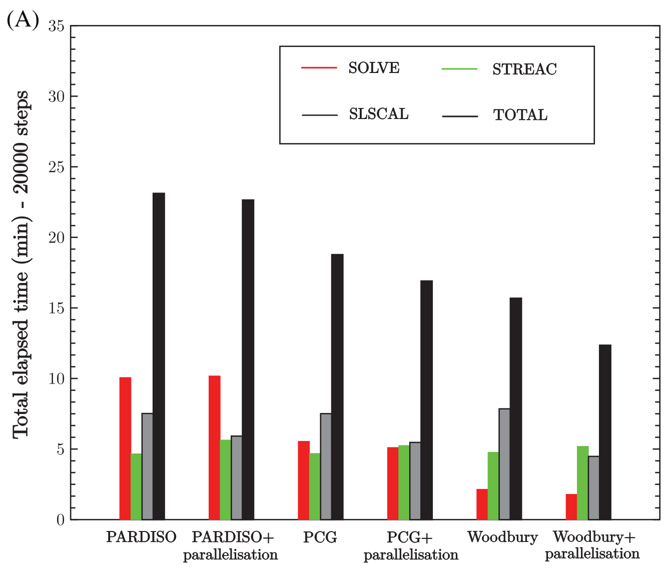
FIGURE 7 Comparison of CPU time taken per analysis step by the SOLVE block of the parallel direct sparse solver, preconditioned conjugate-gradient, and Woodbury solvers: for the Shear wall (A) and RC Slab (B) cases, shown for the first 1000 steps

is illustrated in Figure 7 for the shear wall and RC slab studies. On the one hand, the PARDISO cost as expected is roughly constant throughout the analysis since the stiffness matrix factorisation and the backward and forward eliminations to obtain the solution are repeated every analysis step. On the other hand, Woodbury and PCG gain speed by reusing the factorisation repeatedly until the optimal point of restart is reached. This pattern of an expensive analysis step followed by relatively cheaper steps is clearly seen in Figure 7B.

In the time patterns for the 2D shear wall example shown in Figure 7A, two types of fluctuations are observed regardless of the solver type. One, when there are other jobs simultaneously running on the same processor (higher anomalous peaks) while the other is when the measured time is so small, of the order of 0.01 seconds, that minor variations seem accentuated thereby giving an impression of rather unstable patterns. However, qualitatively, PARDISO gives a constant response at an average of 0.03 seconds per analysis step while PCG and Woodbury-based solvers take lower times. Whenever the Woodbury's identity is restarted (six times up-to 1000th analysis step - brown peaks), the time tends to be equivalent to that of PARDISO. After restarting, the times for the Woodbury approach are reduced to an average of 0.005 seconds. However, in comparison, the performance of PCG is poorer. This is attributed to the fact that the bandwidth of the stiffness matrix is relatively small for 2D problems. Consequently, backward and forward substitutions are relatively more expensive than a matrix factorisation. Since PCG uses these backward and forward substitutions every analysis step to apply the preconditioner, costs for the PCG-based method increase quicker in the intermediate steps thereby demanding more frequent restarts in comparison to the Woodbury-based solver.

In case of the 3D RC slab, a similar trend is observed in the time patterns of the three solver types, see Figure 7B. The fluctuations are not that apparent since each analysis step takes times in the order of 0.1 seconds. The only interesting point is that the performance of PCG is a little different compared to the 2D shear wall problem. The times for the intermediate steps do not increase as sharply as for the 2D problem because of the larger bandwidth of the stiffness matrix for 3D problems. This results in the back- and forward substitutions becoming relatively cheap compared to the matrix factorisation. However, over both cases, Woodbury outperforms PCG on a single-threading, because of the relatively lower rate of increase of time in the intermediate steps.

In summary, the total elapsed time to solve the system of equations (SOLVE) decreases by a factor of ~ 5 using the Woodbury solver as against PARDISO for both the shear wall and RC slab cases, whose model sizes are roughly 3400 DOFs and 11100 DOFs, respectively. These problem sizes are extremely small in reference to the range illustrated in Figure 2 and therefore, for bigger problem sizes, the gains would be significantly higher. A parametric study on problem sizes for the 3D RC slab is presented in Section 4.2.2. However, the improvement of PCG solver over PARDISO is only by a factor of ~ 2 for both case studies due to the aforementioned reason of PCG using back- and forward substitutions every analysis step to apply the preconditioner. An interesting point of observation, with regards to the Woodbury solution, is that when the rank-update is very high owing to complete loss of stiffness, the solution time exceeds the direct solution time step. To address this, restarts were prescribed for such unforeseen steps, wherein the rank update in one analysis step is large, that is, close to the rank of the critical element matrix, in addition to the time-estimation based restarting steps as detailed in Section 3.4.



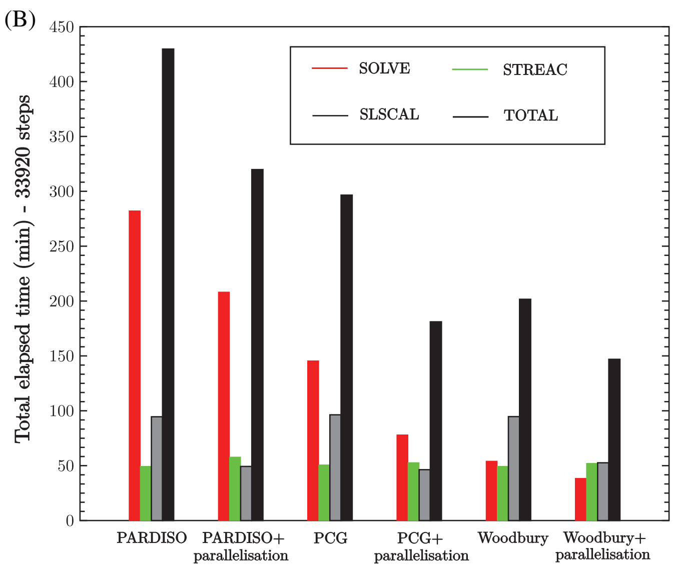


FIGURE 8 Performance of the parallel direct sparse solver, preconditioned conjugate-gradient-, and Woodbury-based direct solution methods, with and without parallelisation, in terms of total elapsed time for the two case studies: (A) Shear wall and (B) RC Slab (right)

Parallel computing. The performance of both the proposed methods have been illustrated, thus far, using 2D and 3D simulations run on a single-thread of a processor. Table 2, summarising the times for major operations in the work flow of SLA, indicates that with improved times for solving the system of equations (SOLVE), the SLSCAL and STREAC building blocks become the bottleneck. To address this and further improve the computational performance of SLA, the operations in SOLVE and SLSCAL have been parallelised. Furthermore, the calculation of stresses and strains in STREAC can also be computed in parallel but this is not taken into account in this study. The result of multi-threading on the performance of the PARDISO, PCG, and Woodbury solvers for the two case studies, using 4 threads of the AMD processor (Section 4.1.3), is shown in Figure 8. Firstly, for the 2D shear wall simulation, the SOLVE blocks of the Woodbury and PCG methods (with multi-threading) improve over PARDISO by the same factors of ~ 5 and ~ 2, respectively, as observed in single-threading. Additionally, PARDISO's performance with multi-threading is similar to its single-threaded counterpart since the problem is 2-dimensional and is small in size. While in the case of SLSCAL, times are reduced with multi-threading owing to the fact that many operations in this block are otherwise carried out sequentially for each integration point. Secondly, with regard to the 3D RC slab simulation, the SOLVE blocks of the Woodbury and PCG methods (with multi-threading) again improve over PARDISO by factors of ~ 5 and ~ 2 respectively, as observed in single-threading. The effective gains made in SOLVE due to multi-threading is greater for PCG in comparison to the Woodbury solver. This is because in PCG's case the number of restarts is greater and with multi-threading, the factorisation and repeated back- and forward substitutions become cheaper. Additionally, PARDISO's performance with multi-threading shows an improvement over its single-threaded counterpart as it is highly optimised for parallel computing. The improvement is more apparent compared to the 2D case because the problem is 3D and is a larger problem size. Furthermore, the effect of parallel computing on PARDISO is expected to increase with increasing problem sizes. In case of the SLSCAL block, all three solvers gain by a factor ~ 2 since the number of integration points in this case is higher than the 2D case and therefore the positive effect of multi-threading is greater. In summary, upon multi-threading, all four possible combinations (of Woodbury and PCG—with or without multi-threading) are an improvement over the traditional direct solution method (PARDISO) in terms of the total elapsed times.

4.2 | Parametric studies

4.2.1 | Effect of number of saw-teeth on solution methods

In order to understand the effect of refinement of the saw-tooth law (p-factor) (as shown in Figure 1A), on the performance of the presented solution strategies in Sections 3.2 and 3.3 and their corresponding restarting approaches, a parametric study is carried out. Only the 3D RC slab is considered for this study since the total computation time is higher than the 2D

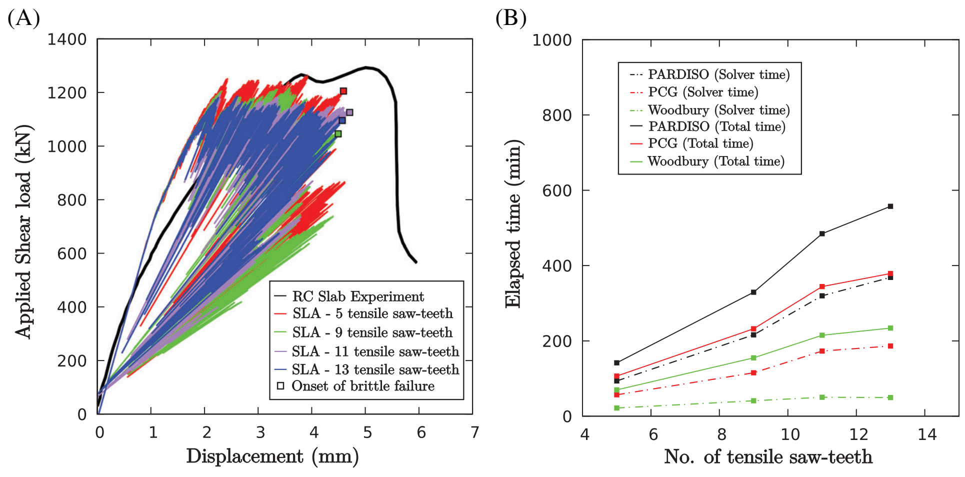


FIGURE 9 (A) Force-displacement curves of four finite element models with different number of saw-teeth for the tensile softening relation and their corresponding onsets of brittle failure and (B) the total and SOLVE-block elapsed times for these cases with regard to the three types of considered solvers

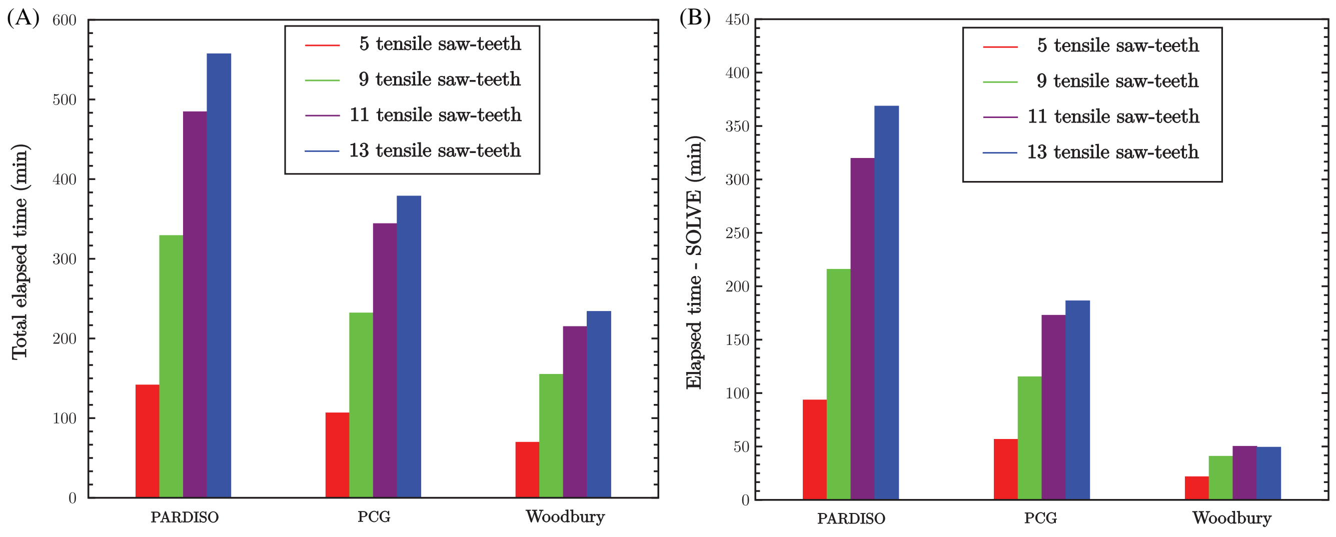


FIGURE 10 Saw-teeth parametric study using the parallel direct sparse solver, preconditioned conjugate-gradient, and Woodbury solvers showing (A) total and (B) SOLVE block elapsed times

example and is therefore more interesting. The single-threaded response elaborated in Sections 4.1.2 and 4.1.3 is treated as reference. By adjusting the saw-teeth discretisation parameter (p) for the number of saw-teeth in the tensile softening relation, four cases are considered: 5, 9 (reference), 11, and 13 saw-teeth, while the compressive softening relation is kept unchanged with respect to the reference case (since the compressive failure doesn't influence the failure mechanism).

Figure 9A shows the force displacement relation for the four considered cases, all run on single-threading. It is quite evident that with increase of the number of saw-teeth (or decrease in p-factor), the peak loads decrease by a small amount. This is attributed to the corresponding shift in strength properties based on the ripple band approach (Figure 1A). The tail part of the post-failure response, as seen in Figure 6B, is not shown here for the four cases because of the small differences between them. However, qualitatively, the onset of brittle failure begins approximately around the same displacement, and a similar ultimate load and failure mechanism are obtained in all the cases. Therefore, in order to objectively compare the computation times of the responses, the onset of brittle failure for each response is treated as the reference point. The number of analysis steps (events) to reach the reference points are 11233, 26202, 38000, and 43051 for the 5, 9 (reference), 11, and 13 tensile saw-teeth cases, respectively.

The total and SOLVE block elapsed times for all four responses, to reach their respective onsets of brittle failure, are illustrated in Figure 9B and additionally in Figure 10. When the saw-tooth model is rather coarse, the number of events required to reach the onset of brittle failure is lower as against a finer one. That is, regardless of the solver type, the total

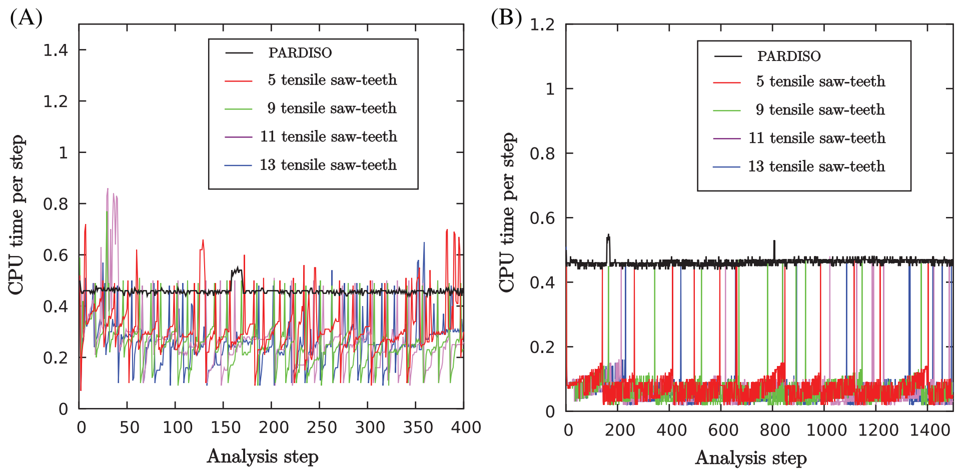


FIGURE 11 (A) Performance of preconditioned conjugate-gradient and (B) Woodbury solvers, with respect to parallel direct sparse solver for the four cases in the saw-teeth parametric study

number of SLA steps required is anyway lower for a coarser saw-tooth model and therefore the total analysis time also drops. This is reflected in the trends of Figure 10A,B and as expected, both PCG and Woodbury outperform PARDISO. However, the point of interest lies in the amount of gain that Woodbury or PCG make over PARDISO which increases with increasing number of saw-teeth and this is evident in both Figures 9B and 10. This is attributed to the inherent increase in the number of events required and also to the fact that the time-estimation based restarting strategy is indirectly related to the rank-update per analysis step in SLA. With increasing number of saw-teeth in the constitutive model, the rank update per analysis step decreases. In other words, the jump in rank is more abrupt for a coarser saw-tooth law as against a finer saw-tooth law. Therefore, the rate of increase of the times for intermediate steps and the proximity of the restarting steps is influenced by the saw-tooth discretisation which in turn affects the amount of gain.

The typical pattern of expensive steps followed by cheaper steps for all four cases, with Woodbury- and PCG-based solvers, is illustrated in Figure 11 and these patterns are identical to those presented in Figure 7B. Only the first few analysis steps are shown herein and the variations in the number of restarts needed for the different cases is evident in the performances of PCG- and the Woodbury-based solver (Figure 11) which in turn affects the aforementioned gain. The performance can be further improved using parallel computing, as detailed in Section 4.1.3, to reduce the total time shown in Figure 10A, but is not presented here owing to triviality.

4.2.2 | Effect of problem size on solution methods

In order to understand the effect of mesh refinement on the performance of the Woodbury and PCG solvers, another sensitivity study is carried out. Once again, only the 3D RC slab is considered for this study because of the higher computation times involved. The response elaborated in Sections 4.1.2 and 4.1.3 is treated as reference, wherein the average size of the 20-noded isoparametric solid brick element is approximately 150 mm. Three other cases with average element sizes of 100, 75, and 50 mm are considered. In terms of the total number of DOFs, the four cases translate to 11175 (reference), 31443, 56910, and 181182 DOFs. The three new simulations are run with the same parameters as in Table 1 except that the saw-teeth discretisation factor (p) is increased to 0.25 to reduce the total number of events, in order to avoid extremely higher computation times. The FE models of the four cases are shown in Figure 13A.

Figure 12A shows the force displacement relation for the four considered cases, all run on multithreading (four cores). It is observed that results are closer to the experimental peak load upon mesh refinement but the finest mesh overshoots the experimental peak load by 150 kN. A similar failure mechanism is observed for the finer meshes but the reference case clearly suffers from mesh objectivity problems, see Figure 13B. However, this is not a feature just of SLA and is also observed in NLFEA, since the brittle shear failure when simulated using 150 mm elements (two elements over the depth of the slab) instead of 100, 75, or 50 mm elements (three, four, and six elements over depth of slab, respectively) could be affected by mesh-directional bias, although the smeared cracking approach based on the traditional crack band theory limits element size dependency. The one way shear failure mechanism is captured better by the finer meshes as shown in Figure 13B. Therefore, in order to objectively compare the computation times of the responses considering the structural

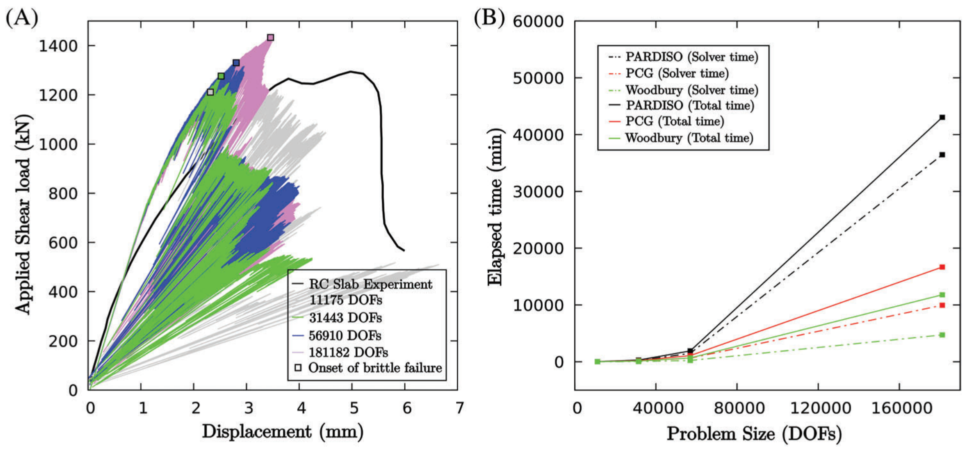


FIGURE 12 (A) Force-displacement curves of four finite element models with different mesh sizes and the corresponding reference points for comparison and (B) the total and SOLVE block elapsed times for these cases with regard to the three types of considered solvers

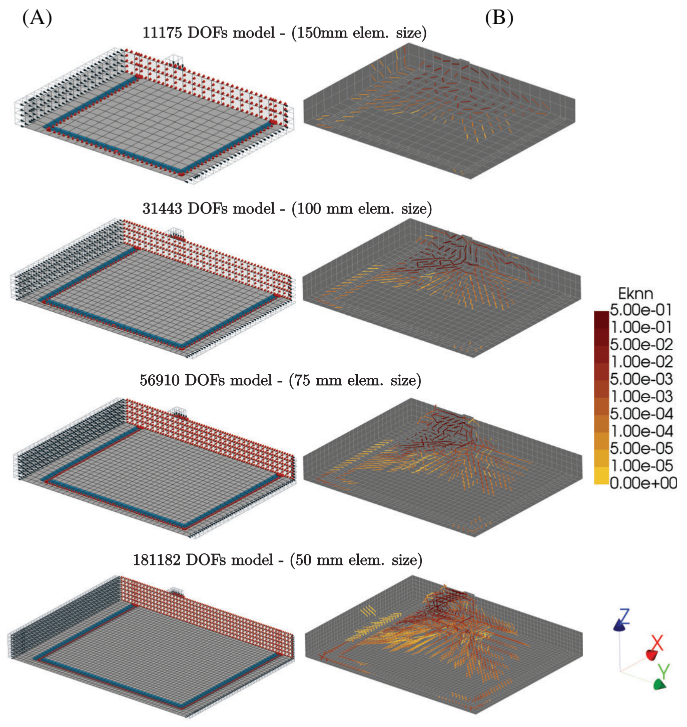

(A)
(B)
11175 DOFs model - (150mm elem. size)
31443 DOFs model - (100 mm elem. size)
56910 DOFs model - (75 mm elem. size)
181182 DOFs model - (50 mm elem. size)
Eknn
5.00e-01
1.00e-01
5.00e-02
1.00e-02
5.00e-03
1.00e-03
5.00e-04
1.00e-04
5.00e-05
1.00e-05
0.00e+00
Z
X
Y

FIGURE 13 (A) Finite element models (halves—using symmetry along global X) for the four cases with average element sizes of 150, 100, 75, and 50 mm: showing the axial loads (along global Y), concentrated shear loads (along global negative Z), and the boundary conditions (simple supports along blue steel plates and for symmetry along the mid-face) and (B) the corresponding crack pattern plots (Eknn denotes the normal crack strain) at the ends of the respective collapses

response, the peak load reached in each response (which is followed by the brittle collapse) is treated as the reference points (denoted in Figure 12A).

It is well known that for a band solver, the calculation of a matrix factorisation scales ($\mathcal{O}(Nb^2)$) and back- and forward substitutions scale ($\mathcal{O}(Nb)$) with respect to the problem size N and bandwidth b (given by ($\mathcal{O}(N^{\frac{(d-1)}{d}}$)), wherein d is the dimension of the problem).[19] Therefore, for 3D problems, factorisation and back/forward substitution scale ($\mathcal{O}(N^{\frac{7}{3}}$)) & ($\mathcal{O}(N^{\frac{5}{3}}$)) respectively. The reinforced slab problem is 3D and therefore, the bandwidth of the stiffness matrix increases significantly faster for increasing problem sizes. Consequently, the number of non-zeros within the bandwidth increases faster and the number of calculations to be performed on these non-zero elements increase. This manifests as the nonlinear trend observed in Figure 12B for all three solution methods. For smaller problems, the costs of factorisation

TABLE 3 Factors of improvement of Woodbury and preconditioned conjugate gradient (PCG) methods over parallel direct sparse solver's (PARDISO) performance treated as unity

		SOLVE			TOTAL		
Problem Size (DOFs)	**Analysis Steps (Events)**	**PARDISO**	**PCG**	**Woodbury**	**PARDISO**	**PCG**	**Woodbury**
11175	3999	1.0	1.80	4.38	1.0	1.29	1.58
31443	8049	1.0	1.98	5.08	1.0	1.54	2.30
56910	23 708	1.0	2.34	6.42	1.0	1.77	2.87
181182	93 328	1.0	3.66	7.72	1.0	2.58	3.65

TABLE 4 Contribution of the three dominant processes— Solving the system of linear equations (SOLVE); calculation of stresses and strains from the displacement field (STREAC); and Determination of the critical integration point (IP), updating its stiffness and strength, and scaling results (SLSCAL), in percentages, for the four cases of Section 4.2.2 run with the parallel direct sparse solver (PARDISO), preconditioned conjugate gradient (PCG) and Woodbury solvers.

Problem Size (degrees of freedom, DOFs)	**SOLVE**			**STREAC**			**SLSCAL**		
	PARDISO	**PCG**	**Woodbury**	**PARDISO**	**PCG**	**Woodbury**	**PARDISO**	**PCG**	**Woodbury**
11175	63.08	40.9	22.73	17.79	30.21	37.59	16.73	24.84	34.69
31443	73.33	57.15	33.23	14.08	21.47	34.61	11.20	18.28	27.30
56910	77.68	58.72	34.76	12.00	20.86	34.21	9.25	17.41	26.26
181182	84.70	59.71	40.08	7.59	19.02	28.08	7.02	18.52	27.72

and back- and forward substitutions are low. However, most of the performance that is gained with the solution methods is compensated by relatively expensive overhead costs for setting up Woodbury's identity and PCG which include allocating memory, creating arrays and initialising the PARDISO interface for the back- and forward substitutions. Since these costs do not depend on the problem size, with increasing problem sizes the influence of the overhead costs on the overall performance drops.

The gains made by the proposed solution methods compared to the PARDISO, all run with four threads of the AMD processor (Section 4.1.3), are summarised in Table 3 in terms of factors of improvement. The PARDISO performance is treated as unity. SOLVE improves by a factor of ~ 8 with Woodbury and ~ 4 with PCG for the largest case of 181182 DOFs. All simulations are run fully except the PARDISO ones, because of the enormous run-times, and the total times for PARDISO necessary to calculate the factors depicted in Table 3, are extrapolated based on the average of the first 3000 steps of the simulations for each of the cases.

In case of 2D problems, the factorisation scales ($\mathcal{O}(N^2)$) and back/forward substitutions scale ($\mathcal{O}(N^{\frac{3}{2}})$) for band solvers. Although the influence of problem sizes on the 2D case study is not illustrated here, previous studies[24] show an almost linear scaling of the direct solution, contrary to that of the 3D case. This can be attributed to the relatively small growth in bandwidth due to the problem being 2D. For 3D problems, the bandwidth of the matrix generally grows significantly faster due to the inherent numbering of the DOFs. Furthermore, as previously pointed out in Section 4.1.3, for 2D problems the back- and forward substitutions are relatively expensive and therefore, for increasing problem sizes, the performance of PCG was observed to remain largely equal to that of the direct solution method, while the Woodbury method gained. Detailed information on the performance of the methods for the 2D case with regard to problem sizes can be found in Reference.[24]

5 CONCLUSIONS & FUTURE WORK

This article addresses the high computational intensity of SLA, an alternative to NLFEA for civil engineering applications, through two solutions methods that use the favorable event-by-event strategy of the method. Since numerous linear analyses have to be solved, each requiring an expensive stiffness matrix factorisation that only changes locally, the proposed methods reuse the factorisation of a certain analysis step followed by steps involving small matrix and vector manipulations to solve a significantly smaller system of equations. The first method is that of direct solution method, wherein Woodbury's matrix identity is applied which allows for a numerically efficient computation of the inverse of a low-rank corrected matrix. The second method is a PCG, wherein instead of using an ILU preconditioner, the complete LU factorisation of the stiffness matrix is used as preconditioner for the CG. Due to the quality of this preconditioner, few iterations are required every analysis step. Furthermore, an optimal time-estimation-based restarting strategy is derived, for both the approaches, to determine the point at which a new factorisation should be calculated. The restarting strategy is additionally forced to have restarts when there are unforeseen high rank updates, resulting in times higher than that of the PARDISO, contrary to the assumed repeating patterns of time. Additionally, parallel computing is introduced for certain sections of SLA's algorithm (including those of the proposed solution methods), where there is need for calculations at integration point level, to further improve the computational performance.

Two benchmark cases, involving non-proportional loading (which further increases the computational intensity of SLA due to the additional need for solving quadratic and cubic equations for undamaged integration points), are chosen to elucidate the performance of the proposed solution methods compared against a traditional direct linear solver like PARDISO. The first one is of a masonry wall subject to overburden followed by a lateral load and the second, a prestressed RC slab subject to axial loads followed by a concentrated shear load, simulated using 2D-plane stress and 3D models, respectively. Both the proposed solution methods perform significantly better than PARDISO, especially for 3D problems, and the Woodbury identity-based solver seems the better choice of the two proposed methods. Furthermore, numerical experiments on the sensitivity of the proposed methods were performed for the 3D RC slab case. Firstly, the number of tensile saw-teeth in the constitutive model was varied and as the saw-teeth became finer, the gains made by both the proposed methods over the direct linear solver (PARDISO) increased. The finer the saw-tooth model, the larger the number of events that are required to bring about a similar mechanism as the response using a coarser saw-teeth model. The proposed time-based restarting strategy used by both methods relies indirectly on the rank update per linear analysis which in turn depended on the fineness of the saw-tooth model. Since the rank update per analysis step is smaller for finer cases, which results in a lower rate of increase in time for the intermediate steps, the effective number of restarts are lower and therefore the gains are significant. Secondly, the effect of problem size on solution methods was studied and both methods gained significantly over PARDISO for increasing problem sizes. SOLVE, the bottleneck as illustrated in Figure 2, for large problem sizes is not the constraint anymore as is shown in the drop of contribution to total times from about 85 % to about 40 % in the 181 182 DOFs case, see Table 4. However, the remaining two blocks now become equally intensive. There is further scope for improvement as multithreading is yet to be introduced in STREAC.

In conclusion, to achieve higher speeds for typical FE models used in SLA, the use of Woodbury identity-based solver is recommended, in combination with parallel processing. Furthermore, coarser saw-teeth are recommended for faster simulations and further research is required to find an optimum number of saw-teeth for the best performance with regard to both computational and mechanics aspects. Besides the major gains made within the scope of the linear solver and parallel processing, the computational performance of SLA could be further improved using strategies to allow for multiple failure events, such that the response does not deviate much from the equilibrium path, and these are currently being investigated.

ACKNOWLEDGEMENTS

This research was funded by Nederlandse Aardolie Maatschappij B.V. (NAM) under contract number UI46268 “Physical testing and modelling - masonry structures Groningen” as part of the NAM Hazard & Risk programme which is gratefully acknowledged. The implementation of the presented concept was carried out in the commercial package DIANA and the authors are thankful for the collaboration with DIANA FEA B.V. towards the development of SLA and the forthcoming new implementations.

ORCID

M. Pari https://orcid.org/0000-0002-9969-0843

REFERENCES

1. Rots JG. Sequentially linear continuum model for concrete fracture. *FraMCoS*. 2001;2:831-840.
2. Rots JG, Invernizzi S. Regularized sequentially linear saw-tooth softening model. *Int J Numer Anal Methods Geomech*. 2004;28:821-856. https://doi.org/10.1002/nag.371.
3. Hendriks MAN, Rots JG. Robust modeling of RC structures with an “event-by-event” strategy. *Eng Fract Mech*. 2008;75:590-614. https://doi.org/10.1201/9780203882955.ch76.

4. Van de Graaf Anne V. Sequentially linear analysis for simulating brittle failure [PhD thesis]. Delft University of Technology, 2017.
5. Invernizzi S, Trovato D, Hendriks MAN, Van de Graaf Anne V. Sequentially linear modelling of local snap-back in extremely brittle structures. *Eng Struct*. 2011;33(5):1617-1625. https://doi.org/10.1016/j.engstruct.2011.01.031.
6. DeJong MJ, Hendriks MAN, Rots JG. Sequentially linear analysis of fracture under non-proportional loading. *Eng Fract Mech*. 2008;75:5042-5056. https://doi.org/10.1016/j.engfracmech.2008.07.003.
7. Jan E, Petr Frantıćk, Miroslav V. Improved sequentially linear solution procedure. *Eng Fract Mech*. 2010;77(12):2263-2276. https://doi.org/10.1016/j.engfracmech.2010.05.018.
8. Graça-E-Costa R, Alfaiate J, Dias-Da-Costa D, Neto P, Sluys LJ. Generalisation of non-iterative methods for the modelling of structures under non-proportional loading. *Int J Fract*. 2013;182(1):21-38. https://doi.org/10.1007/s10704-013-9851-2.
9. Eliáš J. Generalization of load–unload and force-release sequentially linear methods. *Int J Damage Mech*. 2015;24(2):279-293. https://doi.org/10.1177/1056789514531001.
10. Alfaiate J, Sluys LJ. On the use of non-iterative methods in cohesive fracture. *Int J Fract*. 2018;210(1–2):167-186. https://doi.org/10.1007/s10704-018-0270-2.
11. Hendriks MAN, Rots JG. Simulation of creep induced cracking based on sequentially linear analysis. In: Tanabe T, Sakata K, Mihashi H, Sato R, Maekawa K, Nakamura H, eds. Paper presented at: Proceeding of the 8th International Conference on Creep, Shrinkage and Durability Mechanics of Concrete and Concrete Structures; 579–585; 2009.
12. Slobbe AT. Propagation and band width of smeared cracks [PhD thesis]. Delft University of Technology, 2014.
13. Manimaran P, Hendriks MAN, Rots Jan G. Non-proportional loading in sequentially linear analysis for 3D stress states. *Int J Numer Methods Eng*. 2019;119(6):506-531. https://doi.org/10.1002/nme.6060.
14. Vorel J, Boschoff WP. Computational modelling of real structures made of Strain-hardening Cement-based Composites. *Appl Math Comput*. 2015;267:562-570. https://doi.org/10.1016/j.amc.2015.01.056.
15. Waled A. Nonlinear finite element analysis of quasi-brittle materials [PhD thesis]. Cardiff University, 2016.
16. Abd SA-S, Laefer Debra F. Meshfree sequentially linear analysis of concrete. *J Comput Civ Eng*. 2015;30(2):04015009. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000474.
17. Eliáš J, Stang H. Lattice modeling of aggregate interlocking in concrete. *Int J Fract*. 2012;175(1):1-11. https://doi.org/10.1007/s10704-012-9677-3.
18. Jirásek M, Bažant ZP. Macroscopic fracture characteristics of random particle systems. *Int J Fract*. 1994;69:201-228. https://doi.org/10.1007/BF00034763.
19. Golub Gene H, Van Loan Charles F. *Matrix Computations*. 4th ed. Baltimore, MD: John Hopkins University Press; 2013.
20. Anthoine A, Magonette G, Magenes G. Shear-compression testing and analysis of brick masonry walls. Paper presented at: Proceedings of the 10th European Conference on Earthquake Engineering, 1995.
21. Bui TT, Nana WSA, Abouri S, Limam A, Tedoldi B, Roure T. Influence of uniaxial tension and compression on shear strength of concrete slabs without shear reinforcement under concentrated loads. *Constr Build Mater*. 2017;146:86-101. https://doi.org/10.1016/j.conbuildmat.2017.04.068.
22. Slobbe AT, Hendriks MAN, Rots JG. Sequentially linear analysis of shear critical reinforced concrete beams without shear reinforcement. *Finite Elem Anal Des*. 2012;50:108-124. https://doi.org/10.1016/j.finel.2011.09.002.
23. DeJong MJ, Belletti B, Hendriks MAN, Rots JG. Shell elements for sequentially linear analysis: lateral failure of masonry structures. *Eng Struct*. 2009;31:1382-1392. https://10.1016/j.engstruct.2009.02.007.
24. Swart W. Methods for improving the computational performance of sequentially linear analsysis [Master's thesis]. Delft University of Technology, 2018.
25. Manimaran P, Rots JG, Max H. Non-proportional loading for 3-D stress situations in sequentially linear analysis. In: Günther M, Bernhard P, Rots Jan G, eds. *Computational Modelling of Concrete Structures*. Bad Hofgastein, Austria: CRC Press; 2018:931-940.
26. Yanxin Z. Validation of nonlinear finite element analysis on reinforced concrete slab under concentrated out-of-plane load combined with uniaxial in-plane load based on case study [Master's thesis], Delft University of Technology, 2018.
27. Nana WSA, Bui TT, Limam A, Abouri S. Experimental and numerical modelling of shear behaviour of full-scale RC slabs under concentrated loads. *Structure*. 2017;10:96-116. https://10.1016/j.istruc.2017.02.004.

SUPPORTING INFORMATION

Additional supporting information may be found online in the Supporting Information section at the end of this article.

How to cite this article: Pari M, Swart W, van Gijzen M, Hendriks M, Rots J. Two solution strategies to improve the computational performance of sequentially linear analysis for quasi-brittle structures. *Int J Numer Methods Eng*. 2020;121:2128–2146. https://doi.org/10.1002/nme.6302