Comput Geosci
DOI 10.1007/s10596-017-9667-7

ORIGINAL PAPER

An MSSS-preconditioned matrix equation approach for the time-harmonic elastic wave equation at multiple frequencies

M. Baumann[1] · R. Astudillo[1] · Y. Qiu[2] · E. Y. M. Ang[3] · M. B. van Gijzen[1] · R.-É. Plessix[4]

Received: 20 September 2016 / Accepted: 25 May 2017

Abstract In this work, we present a new numerical framework for the efficient solution of the time-harmonic elastic wave equation at multiple frequencies. We show that multiple frequencies (and multiple right-hand sides) can be incorporated when the discretized problem is written as a matrix equation. This matrix equation can be solved efficiently using the preconditioned IDR(s) method. We present an efficient and robust way to apply a single preconditioner using MSSS matrix computations. For 3D problems, we present a memory-efficient implementation that exploits the solution of a sequence of 2D problems. Realistic examples in two and three spatial dimensions demonstrate the performance of the new algorithm.

Keywords Time-harmonic elastic wave equation · Multiple frequencies · Induced dimension reduction (IDR) method · Preconditioned matrix equations · Multilevel sequentially semiseparable (MSSS) matrices

✉ M. Baumann
m.m.baumann@tudelft.nl

[1] Delft University of Technology, Delft, The Netherlands

[2] Max Planck Institute for Dynamics of Complex Technical Systems, Magdeburg, Germany

[3] Nanyang Technological University, Singapore 639798, Singapore

[4] Shell Global Solutions International B.V., The Hague, The Netherlands

1 Introduction

The understanding of the earth subsurface is a key task in geophysics, and Full-Waveform Inversion (FWI) is a computational approach that matches the intensity of reflected shock waves (measurements) with simulation results in a least squares sense; cf. [44] and the references therein for an overview of state-of-the-art FWI algorithms. From a mathematical point of view, the problem of matching measurements with simulation results leads to a PDE-constrained optimization problem where the objective function is defined by the respective FWI approach, and the constraining partial differential equation (PDE) is the wave equation. Since the earth is an elastic medium, the elastic wave equation needs to be considered. In order to design an efficient optimization algorithm, a fast numerical solution of the elastic wave equation is required at every iteration of the optimization loop. This so-called *forward problem* is the focus of this work.

More recently, FWI has been considered for an equivalent problem formulated in the frequency-domain [22, 28]. The frequency-domain formulation of wave propagation has shown specific modeling advantages for both acoustic and elastic media. For the efficient FWI, notably the waveform tomography [27, 44], a fast numerical solution of the respective time-harmonic forward problem is required. More precisely, the forward problem requires the fast numerical solution of the discretized time-harmonic elastic wave equation at multiple wave frequencies and for multiple source terms. In this context, many efficient numerical solution methods have been proposed mostly for the (acoustic) Helmholtz equation [23, 25, 26, 33]. In this work, we present an efficient solver of the time-harmonic *elastic* wave equation that results from a finite element discretization, cf. [11, 15].

Especially for large 3D problems, the efficient numerical solution with respect to computation time and memory requirements is subject to current research. When an iterative Krylov method is considered, the design of efficient preconditioners for the elastic wave equation is required. In [1] a damped preconditioner for the elastic wave equation is presented. The authors of [34] analyze a multi-grid approach for the damped problem. Both works are extensions of the work of Erlangga et al. [33] for the acoustic case. The recent low-rank approach of the MUMPS solver [2] makes use of the hierarchical structure of the discrete problem and can be used as a preconditioner, cf. [3, 46]. When domain decomposition is considered, the sweeping preconditioner [42] is an attractive alternative.

In this work, we propose a *hybrid* method that combines the iterative Induced Dimension Reduction (IDR) method with an efficient preconditioner that exploits the multi-level sequentially semiseparable (MSSS) matrix structure of the discretized elastic wave equation on a Cartesian grid. Moreover, we derive a matrix equation formulation that includes multiple frequencies and multiple right-hand sides, and present a version of IDR that solves linear matrix equations at a low memory requirement. The paper is structured as follows: In Section 2, we derive a finite element discretization for the time-harmonic elastic wave equation with a special emphasis on the case when multiple frequencies are present. Section 3 presents the IDR(s) method for the iterative solution of the resulting matrix equation. We discuss an efficient preconditioner in Section 4 based on the MSSS structure of the discrete problem. We present different versions of the MSSS preconditioner for 2D and 3D problems in Sections 4.2 and 4.3, respectively. The paper concludes with extensive numerical tests in Section 5.

2 The time-harmonic elastic wave equation at multiple frequencies

In this section, we present a finite element discretization of the time-harmonic elastic wave equation with a special emphasis on the mathematical and numerical treatment when multiple frequencies (and possibly multiple right-hand sides) are present.

2.1 Problem description

The time-harmonic elastic wave equation describes the displacement vector $\mathbf{u} : \Omega \to \mathbb{C}^d$ in a computational domain $\Omega \subset \mathbb{R}^d, d \in \{2, 3\}$, governed by the following partial differential equation (PDE),

$$-\omega_k^2 \rho(\mathbf{x})\mathbf{u}_k - \nabla\cdot\sigma(\mathbf{u}_k) = \mathbf{s},\ \mathbf{x} \in \Omega \subset \mathbb{R}^d,\ k = 1, ..., N_\omega. \tag{1}$$

Here, $\rho(\mathbf{x})$ is the density of an elastic material in the considered domain Ω that can differ with $\mathbf{x} \in \Omega$ (inhomogeneity), $\mathbf{s}$ is a source term, and $\{\omega_1, ..., \omega_{N_\omega}\}$ are multiple angular frequencies that define N_ω problems in Eq. 1. The stress and strain tensor follow from Hooke's law for isotropic elastic media,

$$\sigma(\mathbf{u}_k) := \lambda(\mathbf{x})\,(\nabla\cdot\mathbf{u}_k\ I_d) + 2\mu(\mathbf{x})\epsilon(\mathbf{u}_k), \tag{2}$$

$$\epsilon(\mathbf{u}_k) := \frac{1}{2}\left(\nabla\mathbf{u}_k + (\nabla\mathbf{u}_k)^\mathsf{T}\right), \tag{3}$$

with λ, μ being the Lamé parameters (6). On the boundary $\partial\Omega$ of the domain Ω, we consider the following boundary conditions,

$$i\omega_k\rho(\mathbf{x})B\mathbf{u}_k + \sigma(\mathbf{u}_k)\hat{\mathbf{n}} = \mathbf{0}, \quad \mathbf{x} \in \partial\Omega_a, \tag{4}$$

$$\sigma(\mathbf{u}_k)\hat{\mathbf{n}} = \mathbf{0}, \quad \mathbf{x} \in \partial\Omega_r, \tag{5}$$

where Sommerfeld radiation boundary conditions at $\partial\Omega_a$ model absorption, and we typically prescribe a free-surface boundary condition in the north of the computational domain $\partial\Omega_r$, with $\partial\Omega_a \cup \partial\Omega_r = \partial\Omega$. In Eq. 4, B is a $d \times d$ matrix that depends on c_p and c_s, $B \equiv B(\mathbf{x}) := c_p(\mathbf{x})\hat{\mathbf{n}}\hat{\mathbf{n}}^\mathsf{T} + c_s(\mathbf{x})\hat{\mathbf{t}}\hat{\mathbf{t}}^\mathsf{T} + c_s(\mathbf{x})\hat{\mathbf{s}}\hat{\mathbf{s}}^\mathsf{T}$, with vectors $\{\hat{\mathbf{n}}, \hat{\mathbf{t}}, \hat{\mathbf{s}}\}$ being normal or tangential to the boundary, respectively; cf. [1] for more details. Note that the boundary conditions (4)–(5) can naturally be included in a finite element approach as explained in Section 2.2.

We assume the set of five parameters $\{\rho, c_p, c_s, \lambda, \mu\}$ in Eqs. 1–5 to be space-dependent. The Lamé parameters λ and μ are directly related to the density ρ and the speed of P-waves c_p and speed of S-waves c_s via,

$$\mu = c_s^2\rho, \quad \lambda = \rho\left(c_p^2 - 2c_s^2\right). \tag{6}$$

All parameter functions are assumed in $L^1(\Omega)$. More specifically, we interpolate data points using Q_1 basis functions.

2.2 Finite element (FEM) discretization

For the discretization of Eqs. 1–5, we follow a classical finite element approach using the following ansatz,

$$\mathbf{u}_k(\mathbf{x}) \approx \sum_{i=1}^{N} u_k^i \boldsymbol{\varphi}_i(\mathbf{x}), \quad \mathbf{x} \in \Omega \subset \mathbb{R}^d, \quad u_k^i \in \mathbb{C}. \tag{7}$$

In the numerical examples presented in Section 5, we restrict ourselves to Cartesian grids and basis functions $\boldsymbol{\varphi}_i$ that are B-splines of degree p as described for instance in [10, Chapter 2]. The number of degrees of freedom is, hence, given by

$$N = d \prod_{i\in\{x,y,z\}} (n_i - 1 + p), \quad d \in \{2, 3\},\ p \in \mathbb{N}^+, \tag{8}$$

with n_i grid points in the respective spatial direction (in Fig. 1 we illustrate the case where $d = 2$ and $n_x = 7$, $n_y = 5$).

Definition 1 (**Tensor notation,** [14]) The *dot product* between two vector-valued quantities $\mathbf{u} = (u_x, u_y), \mathbf{v} = (v_x, v_y)$ is denoted as, $\mathbf{u} \cdot \mathbf{v} := u_x v_x + u_y v_y$. Similarly, we define the *componentwise multiplication* of two matrices $U = [u_{ij}], V = [v_{ij}]$ as, $U : V := \sum_{i,j} u_{ij} v_{ij}$.

A Galerkin finite element approach applied to Eq. 1 yields the following weak form: Find $\boldsymbol{\varphi}_i \in [H^1(\Omega)]^d$ such that,

$$-\omega_k^2 \sum_{i=1}^{N} u_k^i \int_\Omega \rho(\mathbf{x}) \boldsymbol{\varphi}_i \cdot \boldsymbol{\varphi}_j \, d\Omega - \sum_{i=1}^{N} u_k^i \int_\Omega \nabla \cdot \sigma(\boldsymbol{\varphi}_i) \cdot \boldsymbol{\varphi}_j \, d\Omega = \int_\Omega \mathbf{s} \cdot \boldsymbol{\varphi}_j \, d\Omega, \quad \text{for all } \boldsymbol{\varphi}_j \in [H^1(\Omega)]^d,$$

$j = 1, ..., N$, and for all source functions $\mathbf{s} \in [L^1(\Omega)]^d$. We exploit the boundary conditions (4) –(5) in the following way,

$$\begin{aligned} \int_\Omega \nabla \cdot \sigma(\boldsymbol{\varphi}_i) \cdot \boldsymbol{\varphi}_j \, d\Omega &= \int_{\partial\Omega} \sigma(\boldsymbol{\varphi}_i) \boldsymbol{\varphi}_j \cdot \hat{\mathbf{n}} \, d\Gamma - \int_\Omega \sigma(\boldsymbol{\varphi}_i) : \nabla \boldsymbol{\varphi}_j \, d\Omega \\ &= -i\omega_k \int_{\partial\Omega_a} \rho(\mathbf{x}) B \boldsymbol{\varphi}_i \cdot \boldsymbol{\varphi}_j \, d\Gamma - \int_\Omega \sigma(\boldsymbol{\varphi}_i) : \nabla \boldsymbol{\varphi}_j \, d\Omega \end{aligned}$$

Note that the stress-free boundary condition (5) can be included naturally in a finite element discretization by excluding $\partial\Omega_r$ from the above boundary integral.

We summarize the finite element discretization of the time-harmonic, inhomogeneous elastic wave equation at multiple frequencies ω_k by,

$$(K + i\omega_k C - \omega_k^2 M)\mathbf{x}_k = \mathbf{b}, \quad k = 1, ..., N_\omega, \tag{9}$$

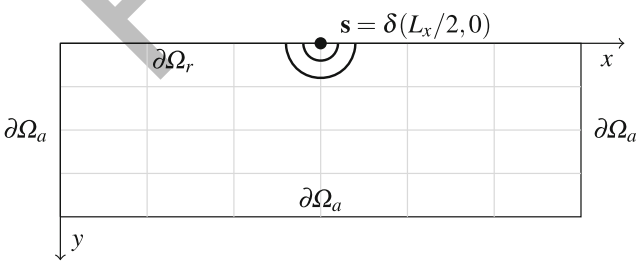


Fig. 1 Boundary conditions and source term for $d = 2$. For $d = 3$, the source is for instance located at $(L_x/2, L_y/2, 0)^\top$

with unknown vectors $\mathbf{x}_k := [u_k^1, ..., u_k^N]^\top \in \mathbb{C}^N$ consisting of the coefficients in Eq. 7, and mass matrix M, stiffness matrix K and boundary matrix C given by,

$$[K]_{ij} = \int_\Omega \sigma(\boldsymbol{\varphi}_i) : \nabla \boldsymbol{\varphi}_j \, d\Omega, \quad [M]_{ij} = \int_\Omega \rho \boldsymbol{\varphi}_i \cdot \boldsymbol{\varphi}_j \, d\Omega,$$
$$[C]_{ij} = \int_{\partial\Omega_a} \rho B \boldsymbol{\varphi}_i \cdot \boldsymbol{\varphi}_j \, d\Gamma, \quad [\mathbf{b}]_j = \int_\Omega \mathbf{s} \cdot \boldsymbol{\varphi}_j \, d\Omega.$$

In a 2D problem (see Fig. 1), the unknown $\mathbf{x}_k$ contains the x-components and the y-components of the displacement vector. When lexicographic numbering is used, the matrices in Eq. 9 have the block structure

$$K = \begin{bmatrix} K_{xx} & K_{xy} \\ K_{yx} & K_{yy} \end{bmatrix}, \quad C = \begin{bmatrix} C_{xx} & C_{xy} \\ C_{yx} & C_{yy} \end{bmatrix}, \quad M = \begin{bmatrix} M_{xx} & M_{xy} \\ M_{yx} & M_{yy} \end{bmatrix},$$

as shown in Fig. 3 (left) for $d = 2$, and Fig. 2 (top left) for $d = 3$. When solving (9) with an iterative Krylov method, it is necessary to apply a preconditioner. Throughout this document, we consider a preconditioner of the form

$$\mathscr{P}(\tau) = (K + i\tau C - \tau^2 M), \tag{10}$$

where τ is a single *seed frequency* that needs to be chosen with care for the range of frequencies $\{\omega_1, ...\omega_{N_\omega}\}$, cf. the considerations in [6, 38]. The efficient application of the preconditioner (10) for problems of dimension $d = 2$ and $d = 3$ on a structured domain is presented in Section 4, and the choice of τ is discussed in Section 5.2.

2.3 Reformulation as a matrix equation

We next describe a new approach to solve (9) at multiple frequencies. Therefore, we define the block matrix $\mathbf{X}$ consisting of all unknown vectors, $\mathbf{X} := [\mathbf{x}_1, ..., \mathbf{x}_{N_\omega}] \in \mathbb{C}^{N \times N_\omega}$, and note that (9) can be rewritten as,

$$\mathscr{A}(\mathbf{X}) := K\mathbf{X} + iC\mathbf{X}\Sigma - M\mathbf{X}\Sigma^2 = B, \tag{11}$$

where $\Sigma := \mathrm{diag}(\omega_1, ..., \omega_{N_\omega})$, and with block right-hand side $B := [\mathbf{b}, ..., \mathbf{b}]$. In Eq. 11, we also define the linear operator $\mathscr{A}(\cdot)$ which defines the matrix Eq. 11 in short-hand notation as $\mathscr{A}(\mathbf{X}) = B$. This reformulation gives rise to use an extension of the IDR(s) method to solve linear matrix equations [4].

Note that an alternative approach to efficiently solve (9) at multiple frequencies ($N_\omega > 1$) leads to the solution of shifted linear systems as presented in [6, Section 4.2] and the references therein. The memory-efficient approach followed by [6] relies on the shift-invariance property of the Krylov spaces belonging to different frequencies. Some

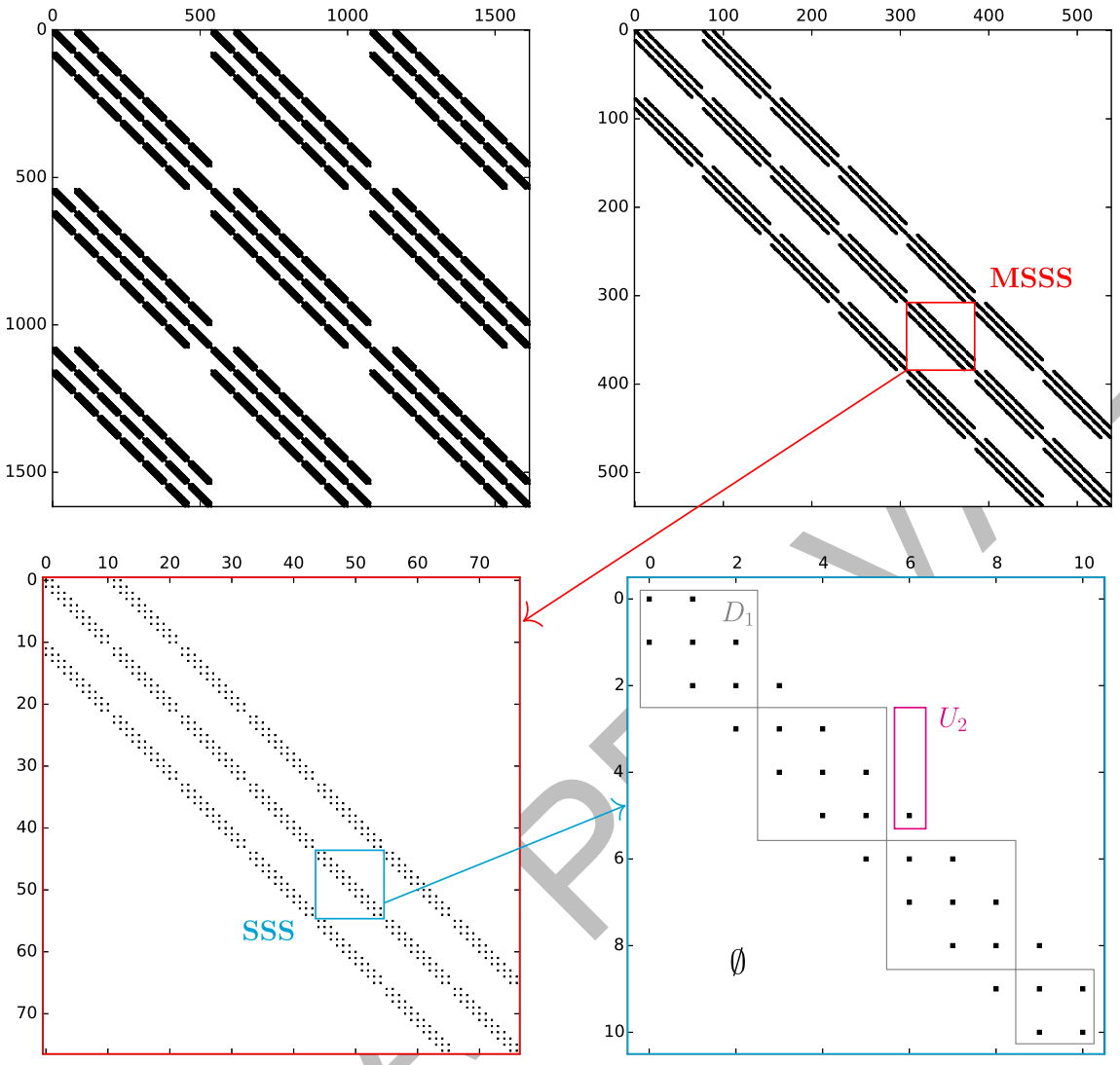


Fig. 2 A `spy` plot of Eq. 10 for a 3D elastic problem when linear basis functions ($p = 1$) are used: In the *top row*, we show the discretized problem for lexicographic (*top left*) and nodal-based ordering (*top right*). Appropriate zooming demonstrates the hierarchically repeating structure of the matrix on level 2 (*bottom left*) and level 1 (*bottom right*). For level 1, we indicate the SSS data structure used in Section 4.1. Here, the rank of U_2 equals one

restrictions of this approach like collinear right-hand sides in Eq. 9 and the difficulty of preconditioner design are, however, not present in the matrix equation setting (11).

3 The induced dimension reduction (IDR) method

Krylov subspace methods are an efficient tool for the iterative numerical solution of large-scale linear systems of equations [20]. In particular, the matrices K, C, M that typically are obtained from a discretization of the time-harmonic elastic wave Eq. 9 are ill-conditioned and have very large dimensions, especially when high frequencies are considered. For these reasons, the numerical solution is computationally challenging, and factors like memory consumption and computational efficiency have to be taken into account when selecting a suitable Krylov method.

The Generalized Minimum Residual (GMRES) method [37] is one of the most widely used Krylov method because of its rather simple implementation and optimal convergence property. Nevertheless, GMRES is a *long-recurrence* Krylov method, i.e., its requirements for memory and computation grow in each iteration which is unfeasible when solving linear systems arising from the elastic wave

equation. On the other hand, short-recurrence Krylov methods keep the computational cost constant per iteration; one of the most used method of this class is the Bi-conjugate gradient stabilized (BiCGStab) method [45].

In this work, we propose to apply an alternative short-recurrence Krylov method: the Induced Dimension Reduction (IDR) method [16, 41]. IDR(s) uses recursions of depth $s+1$, with $s \in \mathbb{N}^+$ being typically small, to solve linear systems of equations of the form,

$$A\mathbf{x} = \mathbf{b}, \quad A \in \mathbb{C}^{N\times N}, \quad \{\mathbf{x}, \mathbf{b}\} \in \mathbb{C}^N, \tag{12}$$

where the coefficient matrix A is a large, sparse, and in general non-Hermitian. We mention some important numerical properties of the IDR(s) method: First, finite termination of the algorithm is ensured with IDR(s) computing the exact solution in $N + \frac{N}{s}$ iterations in exact arithmetics. Second, BiCGStab and IDR(1) are mathematically equivalent [39]. Third, IDR(s) with $s > 1$ often outperforms BiCGStab for numerically difficult problems, for example, for convection-diffusion-reaction problems where the convection term is dominating, or problems with a large negative reaction term, cf. [41] and [16], respectively.

3.1 IDR(s) for linear systems

We present a brief introduction of the IDR(s) method that closely follows [41]. In Section 3.2, we explain how to use IDR(s) for solving (11) for multiple frequencies in a matrix equation setting. We introduce the basic concepts of the IDR(s) method. The IDR(s) algorithm is based on the following theorem.

Theorem 1 (The IDR(s) theorem) *Let A be a matrix in $\mathbb{C}^{N\times N}$, let $\mathbf{v}_0$ be any non-zero vector in $\mathbb{C}^N$, and let $\mathcal{G}_0$ be the full Krylov subspace, $\mathcal{G}_0 := \mathcal{K}_N(A, \mathbf{v}_0)$. Let $\mathcal{S}$ be a (proper) subspace of $\mathbb{C}^N$ such that $\mathcal{S}$ and $\mathcal{G}_0$ do not share a nontrivial invariant subspace of A, and define the sequence:*

$$\mathcal{G}_j := (I - \xi_j A)(\mathcal{G}_{j-1} \cap \mathcal{S}), \quad j = 1, 2, \ldots, \tag{13}$$

where ξ_j are nonzero scalars. Then it holds:

1. *$\mathcal{G}_{j+1} \subset \mathcal{G}_j$, for $j \geq 0$, and,*
2. *$dim(\mathcal{G}_{j+1}) < dim(\mathcal{G}_j)$, unless $\mathcal{G}_j \equiv \{\mathbf{0}\}$.*

Proof Can be found in [41]. □

Exploiting the fact that the subspaces $\mathcal{G}_j$ are shrinking and $\mathcal{G}_j = \{\mathbf{0}\}$ for some j, IDR(s) solves the problem (12) by constructing residuals $\mathbf{r}_{k+1}$ in the subspaces $\mathcal{G}_{j+1}$, while in parallel, it extracts the approximate solutions $\mathbf{x}_{k+1}$. In order to illustrate how to create a residual vector in the space $\mathcal{G}_{j+1}$, let us assume that the space $\mathcal{S}$ is the left null space of a full rank matrix $P := [\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_s]$, $\{\mathbf{x}_i\}_{i=k-(s+1)}^{k}$ are $s+1$ approximations to Eq. 12 and their corresponding residual vectors $\{\mathbf{r}_i\}_{i=k-(s+1)}^{k}$ are in $\mathcal{G}_j$. IDR(s) creates a residual vector $\mathbf{r}_{k+1}$ in $\mathcal{G}_{j+1}$ and obtains the approximation $\mathbf{x}_{k+1}$ using the following $(s+1)$-term recursions,

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \xi_{j+1}\mathbf{v}_k + \sum_{j=1}^{s} \gamma_j \Delta\mathbf{x}_{k-j},$$

$$\mathbf{r}_{k+1} = (I - \xi_{j+1}A)\mathbf{v}_k, \quad \mathbf{v}_k = \mathbf{r}_k - \sum_{j=1}^{s} \gamma_j \Delta\mathbf{r}_{k-j},$$

where $\Delta\mathbf{y}_k$ is the forward difference operator $\Delta\mathbf{y}_k := \mathbf{y}_{k+1} - \mathbf{y}_k$. The vector $\mathbf{c} = (\gamma_1, \gamma_2, \ldots, \gamma_s)^\mathsf{T}$ can be obtained imposing the condition $\mathbf{r}_{k+1} \in \mathcal{G}_{j+1}$ by solving the $s \times s$ linear system,

$$P^\mathsf{H}[\Delta\mathbf{r}_{k-1}, \Delta\mathbf{r}_{k-2}, \ldots, \Delta\mathbf{r}_{k-s}]\mathbf{c} = P^\mathsf{H}\mathbf{r}_k.$$

At this point, IDR(s) has created a new residual vector $\mathbf{r}_{k+1}$ in $\mathcal{G}_{j+1}$. However, using the fact that $\mathcal{G}_{j+1} \subset \mathcal{G}_j$, $\mathbf{r}_{k+1}$ is also in $\mathcal{G}_j$, IDR(s) repeats the above computation in order to create $\{\mathbf{r}_{k+1}, \mathbf{r}_{k+2}, \ldots, \mathbf{r}_{k+s+1}\}$ in $\mathcal{G}_{j+1}$. Once $s+1$ residuals are in $\mathcal{G}_{j+1}$, IDR(s) is able to sequentially create new residuals in $\mathcal{G}_{j+2}$.

3.2 Preconditioned IDR(s) for linear matrix equations

The IDR(s) Theorem 1 can be generalized to solve linear problems in any finite-dimensional vector space. In particular, IDR(s) has recently been adapted to solve linear matrix equations [4]. In this work, we use this generalization of the IDR(s) method to solve the time-harmonic elastic wave equation at multiple frequencies. Using the definition of the linear operator $\mathscr{A}(\cdot)$ in Eq. 11 yields a matrix equation in short-hand notation, $\mathscr{A}(\mathbf{X}) = B$, which is close to Eq. 12. Here, the block right-hand side B equals

$$B := \mathbf{b}[1, 1, \ldots, 1]_{N_\omega} \quad \text{or} \quad B := [\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_{N_\omega}]$$

depending whether we consider a constant source term for each frequency as in Eq. 1 or allow variations.

IDR(s) for solving (11) uses the same recursions described in Section 3.1 acting on block matrices. The

main differences with the original IDR(s) algorithm of [41] are the substitution of the matrix-vector product $A\mathbf{x}$ by the application of the linear operator $\mathscr{A}(\mathbf{X})$, and the use of Frobenius inner products, see Definition 2. Note that two prominent long-recurrence Krylov methods have been generalized to the solution of linear matrix equations in [17] using a similar approach. In Algorithm 1, we present IDR(s) for solving the matrix Eq. 11 with biorthogonal residuals (see details in [4, 16]). The preconditioner used in Algorithm 1 is described in the following Section.

Definition 2 (Frobenius inner product, [17]) The *Frobenius inner product* of two real matrices A, B of the same size is defined as $\langle A, B\rangle_F := \mathrm{tr}(A^{\mathsf{H}}B)$, where $\mathrm{tr}(\cdot)$ denotes the trace of the matrix $A^{\mathsf{H}}B$. The *Frobenius norm* is, thus, given by $\|A\|_F^2 := \langle A, A\rangle_F$.

4 Multilevel sequentially semiseparable preconditioning techniques

Semiseparable matrices [43] and the more general concept of sequentially semiseparable (SSS) matrices [8, 9] are structured matrices represented by a set of generators. Matrices that arise from the discretization of 1D partial differential equations typically have an SSS structure [31], and submatrices taken from the strictly lower/upper-triangular part yield generators of low rank. Multiple applications from different areas can be found [12, 18, 32] that exploit this structure. Multilevel sequentially semiseparable (MSSS) matrices generalize SSS matrices to the case when $d > 1$. Again, discretizations of higher-dimensional PDEs give rise to matrices that have an MSSS structure [29], and the multilevel paradigm yields a hierarchical matrix structure with MSSS generators that are themselves MSSS of a lower hierarchical level. This way, at the lowest level, generators are SSS matrices. The advantages of Cartesian grids in higher dimensions and the resulting structure of the corresponding discretization matrices depicted in Fig. 2 is directly exploited in MSSS matrix computations. For unstructured meshes we refer to [47] where hierarchically semiseparable (HSS) matrices are used. MSSS preconditioning techniques were first studied for PDE-constrained optimization problems in [29] and later extended to computational fluid dynamics problems [30]. In this work, we apply MSSS matrix computations to precondition the time-harmonic elastic wave equation. Appropriate splitting of the 3D elastic operator leads to a sequence of 2D problems in level-2 MSSS structure. An efficient preconditioner for 2D problems is based on model order reduction of level-1 SSS matrices.

Algorithm 1 Preconditioned IDR(s) for linear matrix equations [4]

procedure PIDR(s)
  Input: $\mathscr{A}$ as defined in Eq. 11, $B \in \mathbb{C}^{N\times N_\omega}$, $tol \in (0,\ 1)$, $s \in \mathbb{N}^+$, $P \in \mathbb{C}^{N\times(s\times N_\omega)}$, $\mathbf{X}_0 \in \mathbb{C}^{N\times N_\omega}$, preconditioner $\mathscr{P}$
  Output: $\mathbf{X}$ such that $\|B - \mathscr{A}(\mathbf{X})\|_F/\|B\|_F \le tol$
  $G = 0 \in \mathbb{C}^{N\times s\times N_\omega}$, $U = 0 \in \mathbb{C}^{N\times s\times N_\omega}$
  $M = I_s \in \mathbb{C}^{s\times s}$, $\xi = 1$
  $R = B - \mathscr{A}(\mathbf{X}_0)$
  while $\|R\|_F \le tol \cdot \|B\|_F$ **do**
    Compute $[\mathbf{f}]_i = \langle P_i,\ R\rangle_F$ for $i = 1, \ldots, s$
    for $k = 1$ to s **do**
      Solve $\mathbf{c}$ from $M\mathbf{c} = \mathbf{f}$, $(\gamma_1, \ldots, \gamma_s)^{\mathsf{T}} = \mathbf{c}$
      $V = R - \sum_{i=k}^{s} \gamma_i G_i$
      $V = \mathscr{P}^{-1}(V)$ ▷ Apply preconditioner, see Section 4
      $U_k = U\mathbf{c} + \xi V$
      $G_k = \mathscr{A}(U_k)$
      for $i = 1$ to $k-1$ **do**
        $\alpha = \langle P_i,\ G_k\rangle_F/[M]_{i,i}$
        $G_k = G_k - \alpha G_i$
        $U_k = U_k - \alpha U_i$
      end for
      $[M]_{ik} = \langle P_i,\ G_k\rangle_F$
      $\beta = [\mathbf{f}]_k/[M]_{k,k}$
      $R = R - \beta G_k$
      $\mathbf{X} = \mathbf{X} + \beta U_k$
      if $k+1 \le s$ **then**
        $[\mathbf{f}]_i = 0$ for $i = 1, \ldots, k$
        $[\mathbf{f}]_i = [\mathbf{f}]_i - \beta[M]_{i,k}$ for $i = k+1, \ldots, s$
      end if
      Overwrite k-th block of G, U by G_k and U_k
    end for
    $V = \mathscr{P}^{-1}(R)$ ▷ Apply preconditioner, see Section 4
    $T = \mathscr{A}(V)$
    $\xi = \langle T, R\rangle_F/\langle T, T\rangle_F$
    $\rho = \langle T, R\rangle_F/(\|T\|_F\|R\|_F)$
    if $|\rho| < \rho_0$ **then** ▷ $\rho_0 = 0.7$ recommended in [40]
      $\xi = \rho_0 \times \xi/|\rho|$
    end if
    $R = R - \xi T$
    $\mathbf{X} = \mathbf{X} + \xi V$
  end while
  return $\mathbf{X} \in \mathbb{C}^{N\times N_\omega}$
end procedure

4.1 Definitions and basic SSS operations

We present the formal definition of an SSS matrix used on 1D level in Definition 3.

Definition 3 (SSS matrix structure, [8]) Let A be an $n \times n$ block matrix in SSS structure such that A can be written in the following block-partitioned form,

$$A_{ij} = \begin{cases} U_i W_{i+1} \cdots W_{j-1} V_j^{\mathsf{H}}, & \text{if} \quad i < j, \\ D_i, & \text{if} \quad i = j, \\ P_i R_{i-1} \cdots R_{j+1} Q_j^{\mathsf{H}}, & \text{if} \quad i > j. \end{cases} \tag{14}$$

Here, the superscript 'H' denotes the conjugate transpose of a matrix. The matrices $\{U_s, W_s, V_s, D_s, P_s, R_s, Q_s\}_{s=1}^{n}$ are called *generators* of the SSS matrix A, with their respective dimensions given in Table 1. As a short-hand notation for Eq. 14, we use $A = SSS(P_s, R_s, Q_s, D_s, U_s, W_s, V_s)$.

The special case of an SSS matrix when $n = 4$ is presented in the Appendix.

In general, every matrix can be represented in SSS format. In Fig. 2 (bottom right) we show that the 1D level of the elastic operator is tridiagonal if $p = 1$. Therefore, diagonal blocks D_i are copies of the 1D operator, and off-diagonal blocks can, for instance, be represented by the product of rank-p matrices, $U_2 V_3^{\mathsf{H}}$, where the last element of U_2 is identical to the respective entry of the 1D operator and V_3 is the first unit vector. Basic operations such as addition, multiplication, and inversion are closed under SSS structure and can be performed in linear computational complexity if k_i and l_i in Table 1 are bounded by a constant. The rank of the off-diagonal blocks, formally defined as the *semiseparable order* in Definition 4, plays an important role in the computational complexity analysis of SSS matrix computations.

Definition 4 (Semiseparable order, [13]) Let A be an $n \times n$ block matrix in SSS structure satisfying Definition 3. We use a *colon-style* notation: $A(i : j, k : \ell)$ selects rows of blocks from i to j and columns of blocks from k to ℓ of the SSS matrix A, i.e. $A(2{:}2, 3{:}3) = U_2 V_3^{\mathsf{H}}$. Let

$$\text{rank } A(s{+}1{:}n, 1{:}s) =: l_s, \; s = 1, 2, \ldots, n{-}1, \text{ and let further,}$$

$$\text{rank } A(1{:}s, s+1{:}n) =: u_s, \; s = 1, 2, \ldots, n-1.$$

Setting $r^l := \max\{l_s\}$ and $r^u := \max\{u_s\}$, we call r^l the *lower semiseparable order* and r^u the *upper semiseparable order* of A, respectively.

If the upper and lower semiseparable order are bounded by say r^*, i.e., $\{r^l, r^u\} \leq r^*$, then the computational cost for the SSS matrix computations is of $\mathcal{O}((r^*)^3 n)$ complexity [8], where n is the number of blocks of the SSS matrix as introduced in Definition 3. We will refer to r^* as the maximum off-diagonal rank. Matrix-matrix operations are closed under SSS structure, but performing SSS matrix computations will increase the semiseparable order, cf. [8]. We use model order reduction in the sense of Definition 5 in order to bound the semiseparable order.

Using the aforementioned definition of semiseparable order, we next introduce the following lemma to compute the (exact) LU factorization of an SSS matrix.

Lemma 1 (LU factorization of an SSS matrix) *Let $A = SSS(P_s, R_s, Q_s, D_s, U_s, W_s, V_s)$ be given in generator form with semiseparable order (r^l, r^u). Then the factors of an LU factorization of A are given by the following generators representation,*

$$L = SSS(P_s, R_s, \hat{Q}_s, D_s^L, 0, 0, 0),$$
$$U = SSS(0, 0, 0, D_s^U, \hat{U}_s, W_s, V_s).$$

The generators of L and U are computed by Algorithm 2. Moreover, L has semiseparable order $(r^l, 0)$, and U has semiseparable order $(0, r^u)$.

Algorithm 2 LU factorization and inversion of an SSS matrix A [9, 43]

procedure INV_SSS(A)
  Input: $A = SSS(P_s, R_s, Q_s, D_s, U_s, W_s, V_s)$ in generator form
  // Perform LU factorization
  $D_1 =: D_1^L D_1^U$ ▷ LU factorization on generator level
  Let $\hat{U}_1 := (D_1^L)^{-1} U_1$, and $\hat{Q}_1 := (D_1^L)^{-\mathsf{H}} Q_1$
  for $i = 2 : n - 1$ **do**
    if $i = 2$ **then**
      $M_i := \hat{Q}_{i-1}^{\mathsf{H}} \hat{U}_{i-1}$
    else
      $M_i := \hat{Q}_{i-1}^{\mathsf{H}} \hat{U}_{i-1} + R_{i-1} M_{i-1} W_{i-1}$
    end if
    $\left(D_i - P_i M_i V_i^{\mathsf{H}}\right) =: D_i^L D_i^U$ ▷ LU factorization of generators
    Let $\hat{U}_i := (D_i^L)^{-1}(U_i - P_i M_i W_i)$, and
    let $\hat{Q}_i := (D_i^U)^{-\mathsf{H}}(Q_i - V_i M_i^{\mathsf{H}} R_i^{\mathsf{H}})$
  end for
  $M_n := \hat{Q}_{n-1}^{\mathsf{H}} \hat{U}_{n-1} + R_{n-1} M_{n-1} W_{n-1}$
  $\left(D_n - P_n M_n V_n^{\mathsf{H}}\right) =: D_n^L D_n^U$ ▷ LU factorization of generators
  // Perform inversion
  $L := SSS(P_s, R_s, \hat{Q}_s, D_s^L, 0, 0, 0)$
  $U := SSS(0, 0, 0, D_s^U, \hat{U}_s, W_s, V_s)$
  $A^{-1} = U^{-1} L^{-1}$ ▷ SSS inversion (Appendix A) & matrix-matrix multiplication (Appendix B)
end procedure

Table 1 Generators sizes for the SSS matrix A in Definition 3

U_i	W_i	V_i	D_i	P_i	R_i	Q_i
$m_i \times k_i$	$k_{i-1} \times k_i$	$m_i \times k_{i-1}$	$m_i \times m_i$	$m_i \times l_i$	$l_{i-1} \times l_i$	$m_i \times l_{i+1}$

Note that, for instance, $m_1 + ... + m_n$ equals the dimension of A

Definition 5 (Model order reduction of an SSS matrix) Let $A = SSS(P_s, R_s, Q_s, D_s, U_s, W_s, V_s)$ be an SSS matrix with lower order numbers l_s and upper order numbers u_s. The SSS matrix $\tilde{A} = SSS(\tilde{P}_s, \tilde{R}_s, \tilde{Q}_s, D_s, \tilde{U}_s, \tilde{W}_s, \tilde{V}_s)$ is called a reduced order approximation of A, if $\|A - \tilde{A}\|_2$ is small, and for the lower and upper order numbers it holds, $\tilde{l_s} < l_s, \tilde{u_s} < u_s$ for all $1 \leq s \leq n-1$.

4.2 Approximate block-LU decomposition using MSSS computations for 2D problems

Similar to Definition 3 for SSS matrices, the generators representation for MSSS matrices (level-k SSS matrices) is given in Definition 6.

Definition 6 (MSSS matrix structure, [29]) The matrix A is said to be a level-k SSS matrix if it has a form like (14) and all its generators are level-$(k-1)$ SSS matrices. The level-1 SSS matrix is the SSS matrix that satisfies Definition 3 We call A to be in MSSS matrix structure if $k > 1$.

Most operations for SSS matrices can directly be extended to MSSS matrix computations. In order to perform a matrix-matrix multiplication of two MSSS matrices in linear computational complexity, model order reduction which is studied in [8, 29, 30] is necessary to keep the computational complexity low. The preconditioner (10) for a 2D elastic problem is of level-2 MSSS structure. We present a block-LU factorization of a level-2 MSSS matrix in this Section. Therefore, model order reduction is necessary which results in an *approximate* block-LU factorization. This approximate factorization can be used as a preconditioner for IDR(s) in Algorithm 1. On a two-dimensional Cartesian grid, the preconditioner (10) has a 2×2 block structure as presented in Fig. 3 (left).

Definition 7 (Permutation of an MSSS matrix, [29]) Let $\mathscr{P}(\tau)$ be a 2×2 level-2 MSSS block matrix arising from the FEM discretization of (10) using linear B-splines ($p = 1$),

$$\mathscr{P}(\tau) = \begin{bmatrix} \mathscr{P}_{11} & \mathscr{P}_{12} \\ \mathscr{P}_{21} & \mathscr{P}_{22} \end{bmatrix} \in \mathbb{C}^{2n_x n_y \times 2n_x n_y}, \tag{15}$$

with block entries being level-2 MSSS matrices in generator form,

$$\mathscr{P}_{11} = MSSS(P_s^{11}, R_s^{11}, Q_s^{11}, D_s^{11}, U_s^{11}, W_s^{11}, V_s^{11}), \tag{16a}$$
$$\mathscr{P}_{12} = MSSS(P_s^{12}, R_s^{12}, Q_s^{12}, D_s^{12}, U_s^{12}, W_s^{12}, V_s^{12}), \tag{16b}$$
$$\mathscr{P}_{21} = MSSS(P_s^{21}, R_s^{21}, Q_s^{21}, D_s^{21}, U_s^{21}, W_s^{21}, V_s^{21}), \tag{16c}$$
$$\mathscr{P}_{22} = MSSS(P_s^{22}, R_s^{22}, Q_s^{22}, D_s^{22}, U_s^{22}, W_s^{22}, V_s^{22}), \tag{16d}$$

where $1 \leq s \leq n_x$. Note that all generators in Eqs. 16a–16d are SSS matrices of (fixed) dimension n_y. Let $\{m_s\}_{s=1}^n$ be the dimensions of the diagonal generators of such an SSS

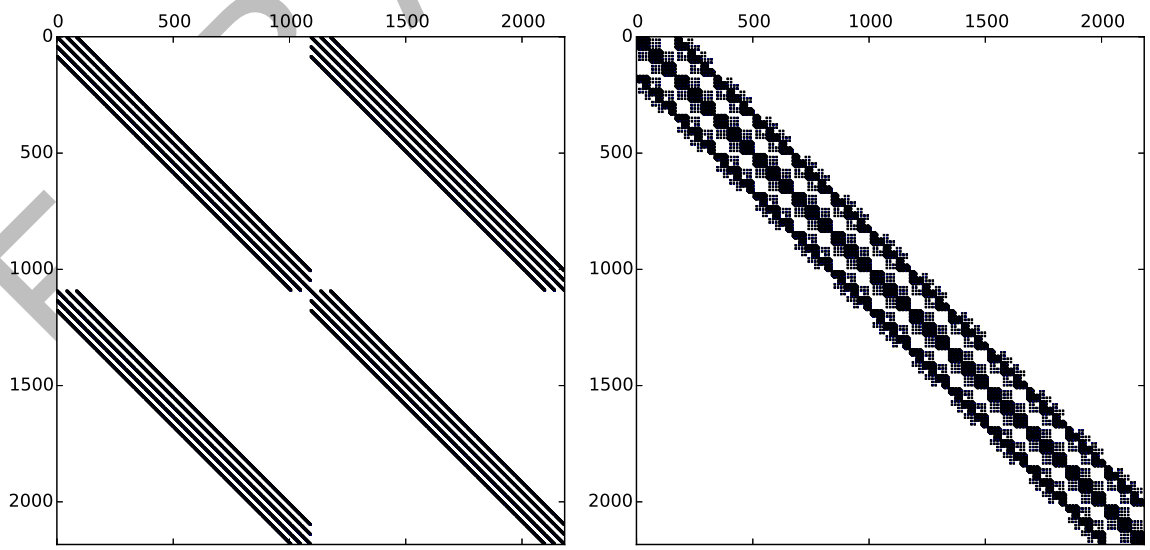


Fig. 3 A `spy` plot of $\mathscr{P}(\tau)$ for the `wedge` problem (*left*) and $\Psi^T \mathscr{P}(\tau) \Psi$ (*right*) for $d = p = 2$, and `nnz=100,587` in both cases. Clearly, the permutation leads to a reduction in bandwidth, and the permuted matrix is block tri-diagonal

matrix, cf. Table 1, with $\sum_{s=1}^{n} m_s = n_y$. Then, there exists a permutation matrix Ψ, $\Psi\Psi^{\mathsf{T}} = \Psi^{\mathsf{T}}\Psi = I$, given by

$$\Psi = \left[I_{n_x} \otimes \begin{bmatrix} \Psi_{1D} \\ 0 \end{bmatrix} \quad I_{n_x} \otimes \begin{bmatrix} 0 \\ \Psi_{1D} \end{bmatrix} \right], \tag{17}$$

where

$$\Psi_{1D} := \left[\text{blkdiag}\left(\begin{bmatrix} I_{m_s} \\ 0 \end{bmatrix} \right)_{s=1}^{n} \quad \text{blkdiag}\left(\begin{bmatrix} 0 \\ I_{m_s} \end{bmatrix} \right)_{s=1}^{n} \right],$$

such that $\mathscr{P}_{2D}(\tau) = \Psi^{\mathsf{T}}\mathscr{P}(\tau)\Psi$ is of global MSSS level-2 structure.

We illustrate the effect of the permutation matrix Ψ in Fig. 3. For a matrix (10) that results from a discretization of the 2D time-harmonic elastic wave equation, P_{2D} is of block tri-diagonal MSSS structure.

Corollary 1 (Block tri-diagonal permutation) *Consider in Definition 7 the special case that the block entries in* Eq. 15 *are given as,*

$$\mathscr{P}_{11} = MSSS\left(P_s^{11}, 0, \underline{I}, D_s^{11}, U_s^{11}, 0, \underline{I}\right), \tag{18a}$$

$$\mathscr{P}_{12} = MSSS\left(P_s^{12}, 0, \underline{I}, D_s^{12}, U_s^{12}, 0, \underline{I}\right), \tag{18b}$$

$$\mathscr{P}_{21} = MSSS\left(P_s^{21}, 0, \underline{I}, D_s^{21}, U_s^{21}, 0, \underline{I}\right), \tag{18c}$$

$$\mathscr{P}_{22} = MSSS\left(P_s^{22}, 0, \underline{I}, D_s^{22}, U_s^{22}, 0, \underline{I}\right), \tag{18d}$$

with rectangular matrix $\underline{I} = [I, 0]$. *Then the matrix* $\Psi^{\mathsf{T}}\mathscr{P}(\tau)\Psi$ *is of block tri-diagonal MSSS structure.*

Proof This result follows from formula (2.13) of Lemma 2.4 in the original proof [29] when generators $R_s^{ij} = W_s^{ij} \equiv 0$ for $i, j \in \{1, 2\}$. □

If the matrix (15) is sparse, it is advisable to use a sparse data structure on generator-level for Eqs. 18a–18d as well.

Because of Corollary 1, the permuted 2D preconditioner can be written as,

$$\mathscr{P}_{2D} = \Psi^{\mathsf{T}}\mathscr{P}(\tau)\Psi = \begin{bmatrix} P_{1,1} & P_{1,2} & & \\ P_{2,1} & P_{2,2} & P_{2,3} & \\ & \ddots & \ddots & \ddots \\ & & \ddots & P_{n_x,n_x} \end{bmatrix} \tag{19}$$

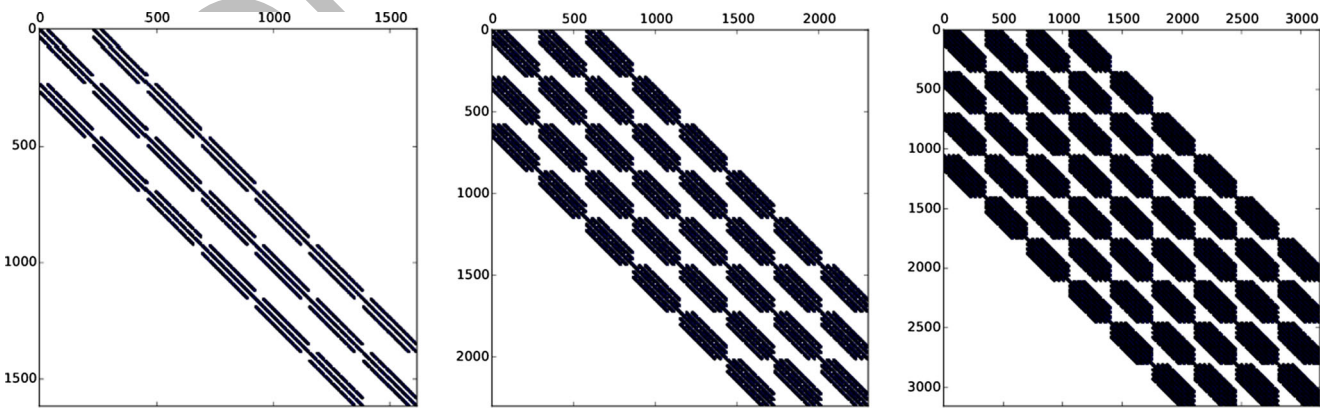
with block entries $P_{i,j}$ in SSS format according to Definition 3, compare Fig. 3 (right). We perform a block-LU factorization of the form $\mathscr{P}_{2D} = LSU$, with

$$L_{i,j} = \begin{cases} I & \text{if } i = j \\ P_{i,j}S_j^{-1} & \text{if } i = j+1 \end{cases}, \quad U_{i,j} = \begin{cases} I & \text{if } j = i \\ S_i^{-1}P_{i,j} & \text{if } j = i+1 \end{cases}, \tag{20}$$

and Schur complements given by

$$S_i = \begin{cases} P_{i,i} & \text{if } i = 1 \\ P_{i,i} - P_{i,i-1}S_{i-1}^{-1}P_{i-1,i} & \text{if } 2 \le i \le n_x. \end{cases} \tag{21}$$

The Schur complements in Eqs. 20–21 are SSS matrices and inverses can be computed with Algorithm 2. From Lemma 1, we conclude that this does not increase the respective off-diagonal ranks. However, in Eqs. 20–21, we also need to perform matrix-matrix multiplications and additions of SSS matrices which lead to an increase in rank, cf. [8] and Appendix B. Therefore, we apply model order reduction in the sense of Definition 5 at each step i of the recursion (21) in order to limit the off-diagonal rank. An algorithm that limits the off-diagonal ranks to a constant, say r^*, can be found in [29]. This leads to approximate Schur complements and, hence, an inexact LU factorization. In Experiment 1, we show that for small off-diagonal ranks, this approach results in a very good preconditioner for 2D elastic problems.



(a) $p = 1$ (b) $p = 2$ (c) $p = 3$

Fig. 4 Nodal-based discretization of $\mathscr{P}_{3D}(\tau)$ in 3D for different degrees p of FEM basis function

4.3 SSOR splitting using MSSS computations for 3D problems

For 3D problems, we consider a nodal-based FEM discretization of Eq. 10 with n_z being the outermost dimension, as shown in Fig. 4 for different order of B-splines. In order to derive a memory-efficient algorithm for 3D problems, we consider the matrix splitting,

$$\mathscr{P}_{3D}(\tau) = \underline{\mathrm{L}} + \hat{S} + \overline{U}, \quad \hat{S} = \mathrm{blkdiag}(\hat{S}_1, ..., \hat{S}_{n_z}), \tag{22}$$

where $\underline{\mathrm{L}}$ and $\overline{U}$ are the (sparse) strictly lower and strictly upper parts of $\mathscr{P}_{3D}(\tau)$, and $\hat{S}$ is a block-diagonal matrix with blocks $\hat{S}_i$ being in level-2 MSSS structure. This data structure is illustrated in Fig. 5a.

According to [36, Section 4.1.2], the SSOR preconditioner based on the splitting (22) is given by,

$$\mathscr{P}_{3D}(\tau) = \frac{1}{\eta(2-\eta)}(\eta\,\underline{\mathrm{L}} + \hat{S})\hat{S}^{-1}(\eta\,\overline{U} + \hat{S})$$

which for $\eta = 1$ equals,

$$\mathscr{P}_{3D}(\tau) = (\underline{\mathrm{L}}\hat{S}^{-1} + I)\hat{S}(\hat{S}^{-1}\overline{U} + I). \tag{23}$$

In Eq. 23, we note that this decomposition coincides with the 2D approach (20)–(21) when the term "$P_{i,i-1}S_{i-1}^{-1}P_{i-1,i}$" in the Schur complements (21) is neglected. This choice avoids a rank increase due to multiplication and addition, but yields a *worse* preconditioner than in 2D. The block entries $\hat{S}_i$,$i = 1, .., n_z$, are in level-2 MSSS structure and, hence, formula (20)-(21) can be applied sequentially for the inverses that appear in Eq. 23. In order to invert level-1 SSS matrices that recursively appear in (21), we use Algorithm 2. On the generator level, we use suitable `LAPACK` routines; cf. Table 2 for an overview of the different algorithms used at each level.

We illustrate the data structure of the preconditioner (23) in 3D for the case of linear B-splines ($p = 1$) in Fig. 5. On level-3, we use a mixed data format that is most memory-efficient for the splitting (22). Since only diagonal blocks need to be inverted, we convert those to level-2 MSSS format, and keep the off-diagonal blocks of $\underline{\mathrm{L}}$ and $\overline{U}$ in sparse format.

For $p > 1$, we apply the permutation of Definition 7 on each diagonal block of $\hat{S}$, cf. Fig. 6. This way, the Schur decomposition described in Section 4.2 can be applied for inverting block tri-diagonal level-2 MSSS matrices.

Table 2 Overview of algorithms applied at different levels for the (approximate) inversion of the preconditioner (23)

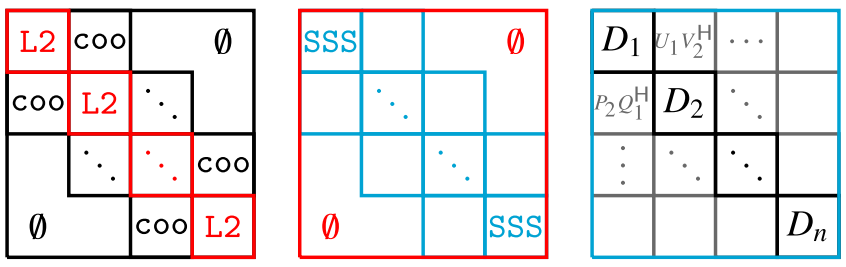
Level	Algorithm for $(\cdot)^{-1}$	Datatype
3D MSSS	SSOR decomposition (23)	`sparse` + `L2_SSS`
2D MSSS	Schur (20)-(21) & MOR	`tridiag.` `L2_SSS`
1D SSS	Algorithm 2	`L1_SSS` (14)
generator	`LAPACK` routines	set of sparse matrices

4.4 Memory analysis for 2D and 3D MSSS preconditioner

We finish our description of MSSS preconditioners with a memory analysis of the suggested algorithms described for 2D problems in Section 4.2, and for 3D problems in Section 4.3, respectively. The following Corollary 2 shows that in both cases, we obtain linear memory requirements in terms of the problem size (8).

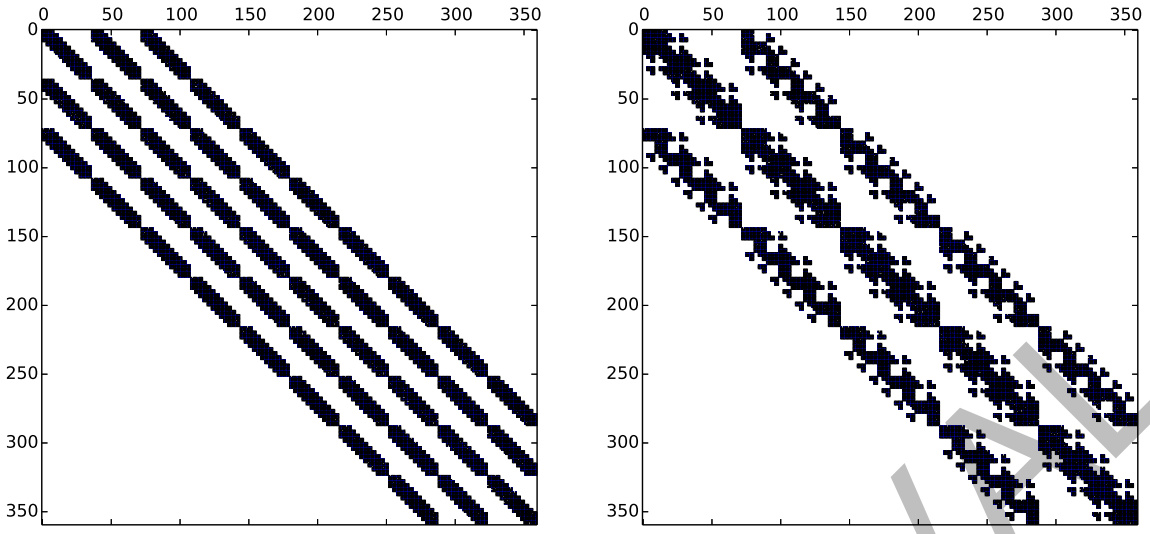
Corollary 2 (Linear memory requirement) *Consider $p = 1$ and a three-dimensional problem of size $n_x \times n_y \times n_z$. For simplicity, we assume on the generator-level $m_i \equiv m$, and the off-diagonal ranks of the inverse Schur complements S_i in* Eq. 21 *being limited by $k_i = l_i \equiv r^*$. The grid size in y-direction on level-1 implies n generators via $n = dn_y m^{-1}$, with m being a constant and $d \in \{2, 3\}$. The memory requirement of the preconditioners $\mathscr{P}_{2D}$ and $\mathscr{P}_{3D}$ presented in* Sections 4.2 and 4.3*, respectively, is linear in the respective problem dimension* (8).

Proof Consider the preconditioner $\mathscr{P}_{2D} = LSU$ given by Eqs. 20–21. Besides blocks of the original operator, an



(a) `L3_SSS` (b) `L2_SSS`, def. 6 (c) `SSS`, def. 3

Fig. 5 Nested data structure for the preconditioner (19) after permutation for $d = 3$ and $p = 1$. With `'coo'` we abbreviate the coordinate-based sparse data structure as used, for instance, in [35]



(a) $\hat{S}_i,\ 1 \le i \le n_x$ (b) $\Psi^\top \hat{S}_i \Psi$

Fig. 6 Permutation on level-2 leads to a block tri-diagonal level-2 MSSS matrix for $p > 1$

additional storage of n_x inverse Schur complements S_i^{-1} in SSS format is required,

$$\mathtt{mem}(\mathscr{P}_{2D}^{-1}, r^*) = \mathtt{mem}(\mathscr{P}_{2D}) + \sum_{i=1}^{n_x} \mathtt{mem}(S_i^{-1}, r^*) \in \mathscr{O}(n_x n_y).$$

The approximate Schur decomposition described in Section 4.2 allows dense, full rank diagonal generators $D_i, 1 \le i \le n$, of size $m \times m$, and limits the rank of all off-diagonal generators by r^* using model order reduction techniques:

$$\mathtt{mem}(S_i^{-1}, r^*) = \underbrace{n \cdot m^2}_{\sim D_i} + \underbrace{4(n-1)mr^*}_{\sim\{U_i, V_i, P_i, Q_i\}} + \underbrace{2(n-2)r^* r^*}_{\sim\{W_i, R_i\}} \in \mathscr{O}(n_y).$$

Concerning the memory requirement for storing $\mathscr{P}_{2D}$ in MSSS format, we first note that the permutation described in Corollary 1 does not affect the memory consumption. Since we use sparse generators in Eqs. 18a–18d, the memory requirement is of the same order as the original, sparse matrix (10) obtained from the FEM discretization.

For 3D problems, we suggest the usage of $\mathscr{P}_{3D}$ as in Eq. 23 based on the splitting (22). For the data structure, we keep the strictly lower and upper diagonal parts in sparse format and convert the diagonal blocks to level-2 MSSS format, cf. Fig. 7,

$$\begin{aligned}\mathtt{mem}(\mathscr{P}_{3D}^{-1}, r^*) \ &= n_z \cdot \mathtt{mem}(\mathscr{P}_{2D}^{-1}, r^*) + \mathtt{nnz}(\underline{\mathrm{L}}) + \mathtt{nnz}(\overline{U}) \\ &\in \mathscr{O}(n_x n_y n_z).\end{aligned}$$

□

Note that the case $p > 1$ also yields a linear memory requirement but is, for simplicity, not addressed here.

5 Numerical experiments

We present numerical examples[1] for the two-dimensional, elastic `Marmousi-II` model [21] as well as for a three-dimensional elastic `wedge` problem which has been inspired by the well-known acoustic test case introduced in [19, 26] for 2D and 3D, respectively. In the examples, we restrict ourselves to Cartesian grids with fixed discretization size $h \equiv h_x = h_y = h_z$. Depending on the specific problem parameters, the maximum frequency we allow is restricted by,

$$f_{\max} < \frac{\min_{\mathbf{x}\in\Omega}\{c_p, c_s\}}{ppw \cdot h}, \quad ppw = 20,$$

where in the following experiments a minimum of 20 points per wavelength (ppw) is guaranteed, and $\omega_k = 2\pi f_k$.

All numerical examples presented in this section have been implemented in FORTRAN 90 using the GNU/gfortran compiler running over GNU/Debian Linux, and executed on a computer with 4 CPUs Intel I5 with 32 GB of RAM.

[1] All test cases are publicly available from the author's github repository [5].

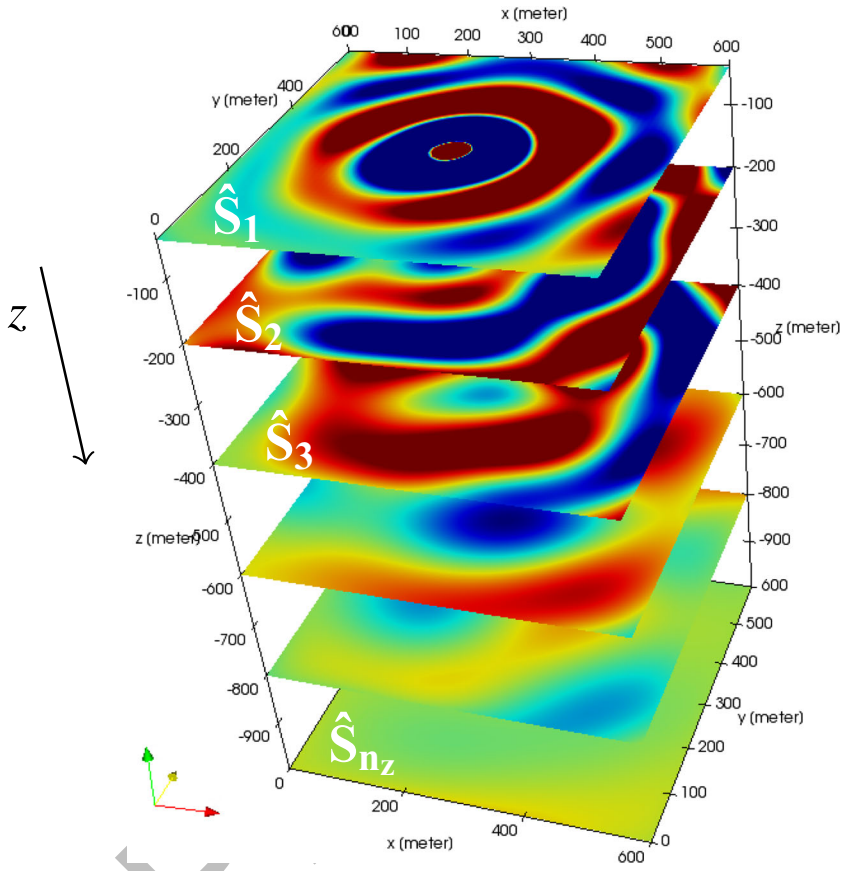


Fig. 7 Schematic illustration: The diagonal blocks of $\hat{S}$ in Eq. 22 correspond to a sequence of n_z 2D problems in the $xy-$plane. This algebraic decoupling affects the propagation of information along the z-axis and, hence, the quality of the preconditioner in 3D. Note that this effect is much less severe in the presence of damping, cf. [7]

5.1 Parameter studies

We begin our numerical tests with a sequence of experiments performed on an *academic* two-dimensional wedge problem described in Fig. 8. The aim of these first experiments is to prove the following concepts for the 2D algorithm introduced in Section 4.2:

- Demonstrate the dependency of the iterative solution method on the maximum off-diagonal rank, $r^* = \max\{r^l, r^u\}$. In Experiment 1 we show that a small value of r^* leads to a very good preconditioner in terms of number of Krylov iterations.
- Show that the 2D algorithm yields linear computational complexity when all problem parameters are unchanged and the grid size doubles (Experiment 2).
- In Experiments 3 and 4, we evaluate the frequency dependency of the MSSS-preconditioner (10) when $\tau \neq \omega$. This is in particular important when multiple

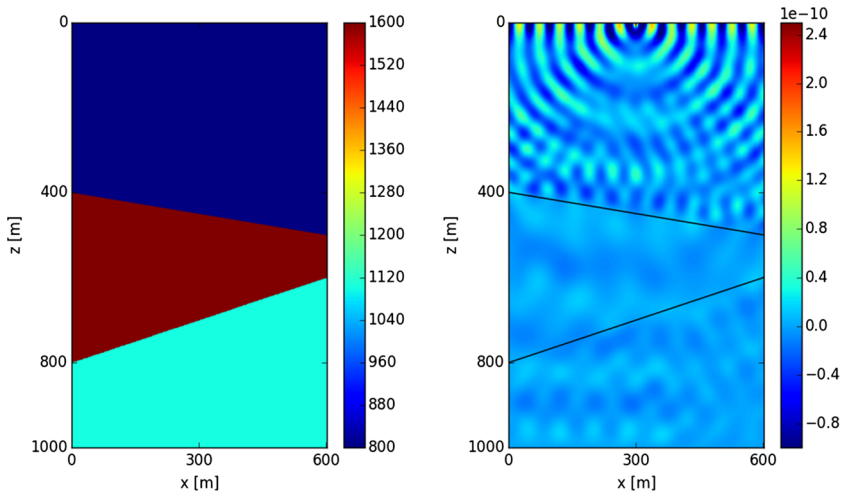


Fig. 8 2D elastic wedge problem used for parameter study: Speed of S-waves in m/s (*left*) and real part of z-component of displacement vector at $f = 16$ Hz (*right*)

frequencies in a matrix equation framework are considered in Section 5.2.

We perform parameter studies on a two-dimensional slice (xz-plane) of the `wedge` problem described in Fig. 14. The values of ρ, c_p and c_s in the respective layers are given in Table 3, and the considered computational domain $\Omega = [0, 600] \times [0, 1000]$ meters is shown in Fig. 8.

In the first set of experiments, we restrict ourselves to the single-frequency case, $N_\omega = 1$. The discrete problem is, thus, given by,

$$(K + i\omega C - \omega^2 M)\mathbf{x} = \mathbf{b},$$

with a preconditioner that approximates the original operator, $\mathscr{P}(\tau) \approx (K + i\tau C - \tau^2 M)$, $\tau = \omega$, by taking low-rank approximations in the block-LU factorization.

Experiment 1 (Off-diagonal rank) *This experiment evaluates the performance of the MSSS-preconditioner (19) for 2D problems when the maximal off-diagonal rank r^* is increased.*

In Experiment 1, we apply the approximate block-LU decomposition (20)–(21) as described in Section 4.2 to the 2D `wedge` problem at frequencies $f = 8$ Hz and $f = 16$ Hz. The maximum off-diagonal rank $r^* = \max\{r^l, r^u\}$ of the Schur complements (21) is restricted using model order reduction techniques, cf. [29]. The dimension of the diagonal constructors has been chosen to be $m_i = 40$, cf. Table 1. Figure 9 shows the convergence behavior of preconditioned IDR(4) (Algorithm 1 with $N_\omega = 1$) and preconditioned BiCGStab [45]. We note that even in the high-frequency case, an off-diagonal rank of $r^* = 10$ leads to a very efficient preconditioner, and an (outer) Krylov method that converges within at most 40 iterations to a residual tolerance `tol=10e-8`. Moreover, we observe that IDR(s) outperforms BiCGStab in the considered example when the same preconditioner is applied. For a rank $r^* > 15$, we observe convergence within very few iterations.

Experiment 2 (Computational complexity in 2D) *The inexact block-LU factorization yields linear computational complexity when applied as a preconditioner within MSSS-preconditioned IDR(s), demonstrated for the 2D wedge problem.*

Table 3 Parameter configuration of the elastic `wedge` problem

Parameter	Layer #1	Layer #2	Layer #3
$\rho[kg/m^3]$	1800	2100	1950
$c_p[m/s]$	2000	3000	2300
$c_s[m/s]$	800	1600	1100

The Lamé parameters can be computed via (6)

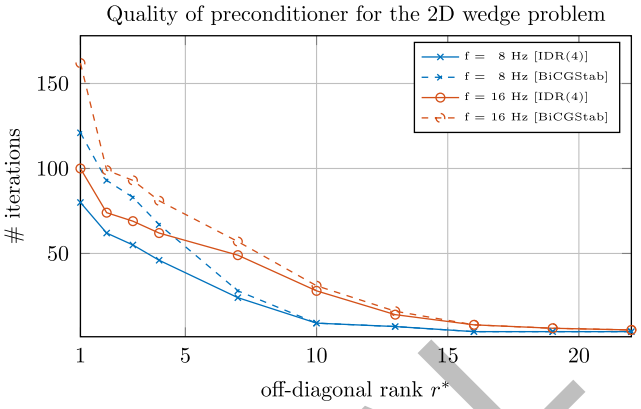


Fig. 9 Number of Krylov iterations when the maximum off-diagonal rank of the inverse Schur complements is restricted to r^*

In our second numerical experiment, the maximum off-diagonal rank is fixed to $r^* = 15$ such that very few IDR iterations are required, and the computational costs in Fig. 10 are dominated by the MSSS preconditioner. We solve the 2D `wedge` problem at frequency 8 Hz for different mesh sizes and a finite element discretization with B-splines of degree $p = \{1, 2\}$. In Fig. 10, the CPU time is recorded for different problem sizes: The mesh size h is doubled in both spatial directions such that the number of unknowns quadruples according to Eq. 8. From our numerical experiments we see that the CPU time increases by a factor of ~ 4 for both, linear and quadratic, splines. This gives strong numerical evidence that the 2D MSSS computations are performed in linear computational complexity.

Experiment 3 (Constant points per wavelength) *Convergence behavior of MSSS-preconditioned IRD(s) when the problem size and wave frequency are increased simultaneously.*

In the previous example, the wave frequency is kept constant while the problem size is increased which is of little

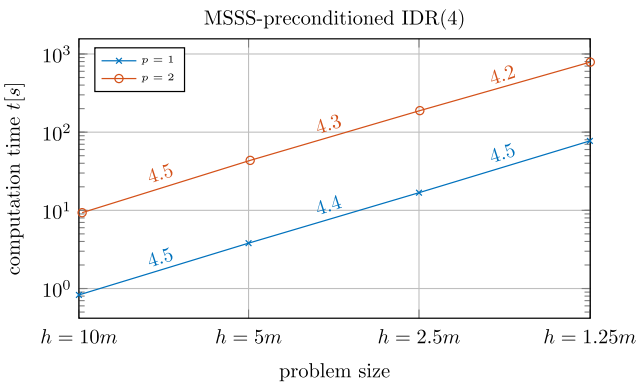


Fig. 10 Linear computational complexity of preconditioned IDR(4) for the 2D `wedge` problem at $f = 8$ Hz

Table 4 Performance of the MSSS preconditioner when problem size and frequency are increased simultaneously such that $ppw = 20$ and `tol=10e-8`: $\mathcal{O}(n^3)$ complexity

f	$h[m]$	r^*	MSSS	IDR(4)	Total CPU time
4 Hz	10.0	5	0.6 s	16 iter.	0.7 s
8 Hz	5.0	7	2.9 s	33 iter.	4.2 s
16 Hz	2.5	10	15.3 s	62 iter.	31.8 s
32 Hz	1.25	16	95.4 s	101 iter.	242.5 s

practical use due to oversampling. We next increase the wave frequency *and* the mesh size simultaneously such that a constant number of points per wavelength, $ppw = 20$, is guaranteed.

In Table 4, we use the freedom in choosing the maximum off-diagonal rank parameter r^* such that the overall preconditioned IDR(s) algorithm converges within a total number of iterations that grows linearly with the frequency. This particular choice of r^* shows that the MSSS preconditioner has comparable performance to the multi-grid approaches in [24, 34] where the authors numerically prove $\mathcal{O}(n^3)$ complexity for 2D problems of size $n_x = n_y \equiv n$.

The off-diagonal rank parameter r^* can on the other hand be used to tune the preconditioner in such a way that the number of IDR iterations is kept constant for various problem sizes. In Table 5, we show that a constant number of ~ 30 IDR iterations can be achieved by a moderate increase of r^* which yields an algorithm that is nearly linear.

Experiment 4 (Quality of $\mathscr{P}_{2D}(\tau)$ when $\tau \neq \omega$) *Single-fre- quency experiments when seed frequency differs from the original problem.*

This experiments bridges to the multi-frequency case. We consider single-frequency problems at $f \in \{2, 3, 4, 5\}$ Hz, and vary the parameter τ of the preconditioner (19). The off-diagonal rank r^* is chosen sufficiently large such that fast convergence is obtained when $\tau = \omega$. From Fig. 11 we

Table 5 Performance of the MSSS preconditioner when problem size and frequency are increased simultaneously such that $ppw = 20$ and `tol=10e-8`: Constant number of iterations

f	$h[m]$	r^*	MSSS	IDR(4)	Total CPU time
4 Hz	10.0	3	0.5 s	29 iter.	0.8 s
8 Hz	5.0	7	2.9 s	33 iter.	4.2 s
16 Hz	2.5	11	16.9 s	27 iter.	24.5 s
32 Hz	1.25	18	107.1 s	33 iter.	163.2 s

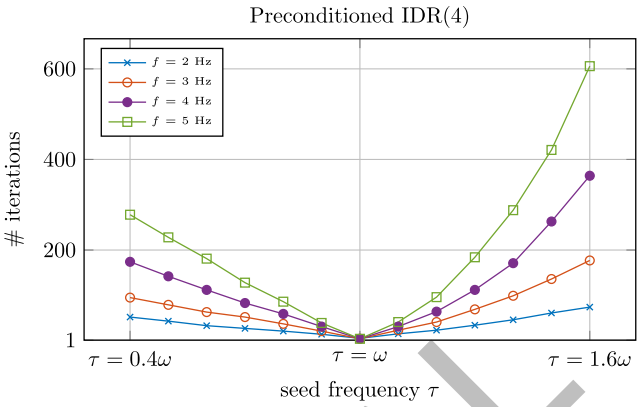


Fig. 11 Number of iterations of preconditioned IDR(s) when $\tau \neq \omega$ in (19). We perform the experiment for different frequencies, and keep a constant grid size $h = 5m$ and residual tolerance `tol = 10e-8`

conclude that the quality of the preconditioner heavily relies on the seed frequency, and a fast convergence of preconditioned IDR(4) is only guaranteed when τ is close to the original frequency.

5.2 The elastic `Marmousi-II` model

We now consider the case when $N_\omega > 1$, and the matrix equation,

$$K\mathbf{X} + iC\mathbf{X}\Sigma - M\mathbf{X}\Sigma^2 = B, \quad \mathbf{X} \in \mathbb{C}^{N \times N_\omega}, \tag{24}$$

is solved. Note that this way we can incorporate multiple wave frequencies in the diagonal matrix $\Sigma = \text{diag}(\omega_1, ..., \omega_{N_\omega})$, and different source terms lead to a block right-hand side of the form $B = [\mathbf{b}_1, ..., \mathbf{b}_{N_\omega}]$. When multiple frequencies are present, the choice of seed frequency τ is crucial as we demonstrate for the `Marmousi-II` problem in Experiment 6. We solve the matrix Eq. 24 arising from the realistic `Marmousi-II` problem [21]. We consider a subset of the computational domain, $\Omega = [0, 4000] \times [0, 1850]m$, as suggested in [34], cf. Fig. 12.

Experiment 5 (`Marmousi-II` at multiple right-hand sides) *Performance of the MSSS-preconditioned IDR(s) method for the two-dimensional `Marmousi-II` problem when multiple source locations are present.*

We consider the `Marmousi-II` problem depicted in Fig. 12 at $h = 5$m and frequency $f = 2$ Hz. We present the performance of MSSS-preconditioned IDR(4) for N_ω equally-spaced source locations (right-hand sides) in Table 6. The CPU time required for the preconditioner as well as the iteration count is constant when $N_\omega > 1$ because

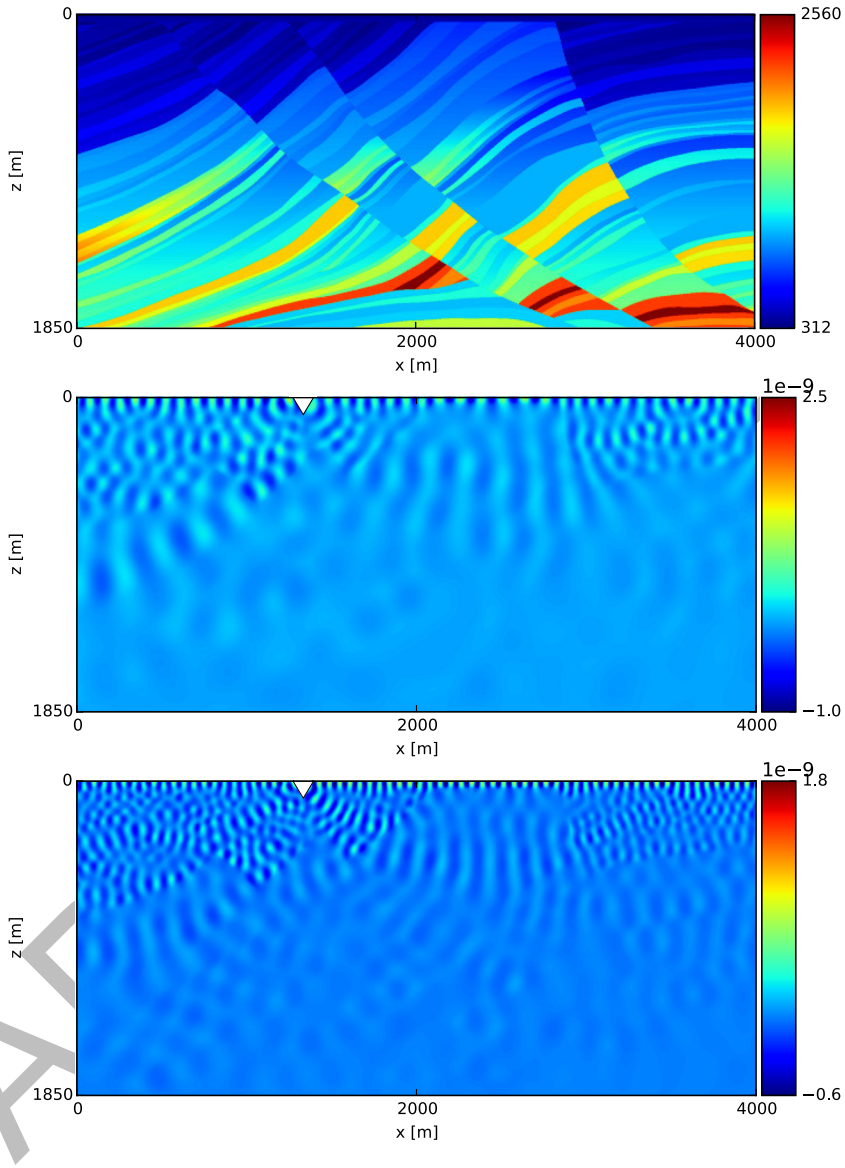


Fig. 12 Speed of S-waves in m/s (*top*), and real part of the z-component of the displacement vector in frequency-domain at $f = 4$ Hz (*middle*) and $f = 6$ Hz (*bottom*) for the `Marmousi-II` model, cf. [21] for a complete parameter set. The source location is indicated by the symbol '$\triangledown$'. In the present setting, the water layer of the `Marmousi-II` model has been removed, and we place the source term at $L_x/3$

we consider a single frequency. The overall wall clock time, however, scales better than N_ω due to the efficient implementation of block matrix-vector products in the IDR algorithm. The experiment for $N_\omega = 20$ shows that there is an optimal number of right-hand sides for a single-core algorithm.

Table 6 Numerical experiments for the `Marmousi-II` problem at $f = 2$ Hz using a maximum off-diagonal rank of $r^* = 15$

# RHSs	MSSS factorization	PIDR(s = 4)
1	60.2 s	8.2 s (8 iter.)
5	60.2 s	25.0 s (8 iter.)
10	60.1 s	43.5 s (8 iter.)
20	60.3 s	108.3 s (8 iter.)

Experiment 6 (`Marmousi-II` at multiple frequencies) *Performance of MSSS-preconditioned IDR(s) for the two-dimensional `Marmousi-II` problem at multiple frequencies.*

In Experiment 6, we consider a single source term located at $(L_x/2, 0)^\mathsf{T}$ and N_ω frequencies equally-spaced in the intervals $f_k \in [2.4, 2.8]$ Hz and $f_k \in [2.0, 4.0]$ Hz. The seed frequency is chosen at $\tau = (1 - 0.5i)\omega_{max}$ for which we recorded optimal convergence behavior. When the number of frequencies is increased, we observe an improved performance compared to an extrapolation of the $N_\omega = 2$

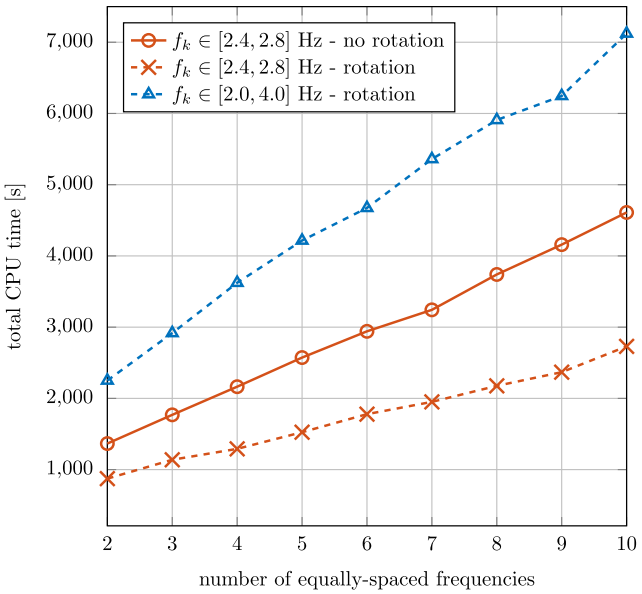


Fig. 13 Total CPU times of MSSS-IDR(4) for $N_\omega > 1$ frequencies equally-spaced within a fixed range. Additional scaling (*dashed lines*) following [7] improves convergence, and allows for larger frequency ranges

case. We also observed that the size of interval in which the different frequencies range is crucial for the convergence behavior. In [7], we describe how the convergence of global GMRES [17] can be improved by scaling the k-th column of the block unknown $\mathbf{X}$ by $e^{-i\varphi_k}$. Spectral anlysis shows that the angles φ_k can be chosen such that the spectrum of the preconditioned operator is rotated and convergence is improved, cf. [7]. In the present case of global IDR(s) (Algorithm 1) combined with an inexact MSSS preconditioner (19), we record a reduction to 60% of the CPU time when spectral rotation is applied to the $N_\omega = 10$ case, cf. Fig. 13.

5.3 A three-dimensional elastic `wedge` problem

The `wedge` problem with parameters presented in Table 3 is extended to a third spatial dimension, resulting in $\Omega = [0, 600] \times [0, 600] \times [0, 1000] \subset \mathbb{R}^3$.

Experiment 7 (A 3D elastic wedge problem) *A three-dimensional, inhomogeneous elastic wedge problem with physical parameters specified in* Table 3 *is solved using the SSOR-MSSS preconditioner described in* Section 4.3.

Similar to Experiment 3, we consider a constant number of 20 points per wavelength, and increase the wave frequency from 2Hz to 4Hz while doubling the number of grid points in each spatial direction (in Fig. 14 we exemplify setup and numerical solution at 4 Hz). In Fig. 15 we observe a factor of ~ 4 which numerically indicates a complexity of

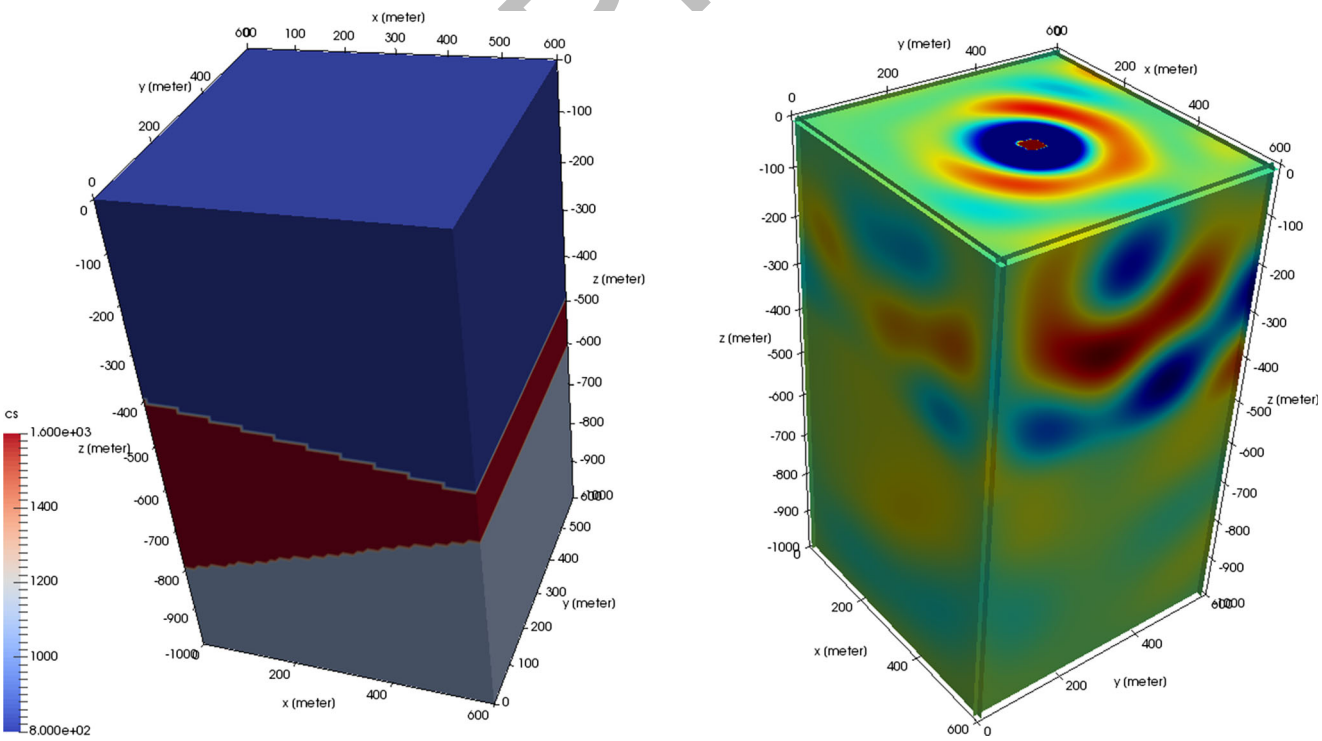


Fig. 14 *Left*: Parameter configuration of the elastic `wedge` problem for $d = 3$ according to Table 3. *Right*: Numerical solution of $\Re(\mathbf{u}_z)$ at $f = 4Hz$

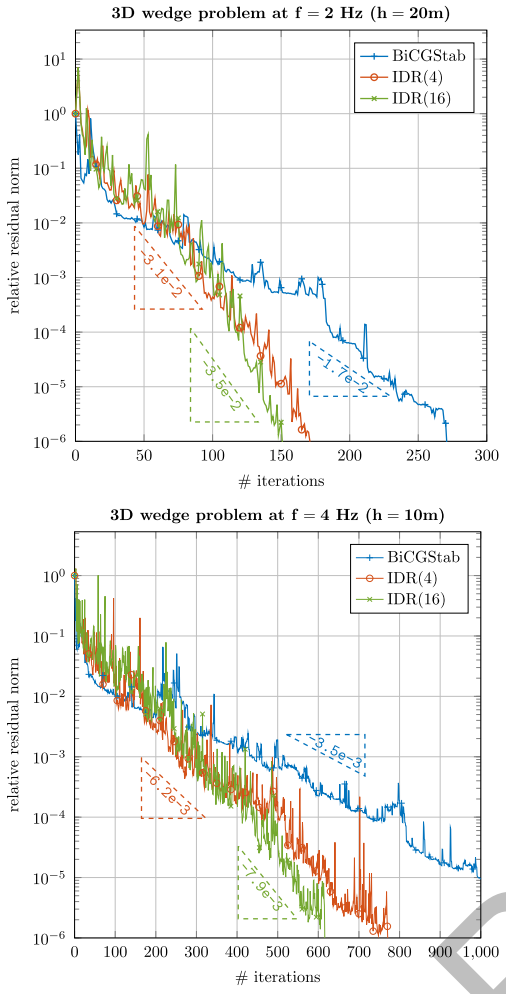


Fig. 15 Convergence history of different Krylov methods preconditioned with the SSOR-MSSS preconditioner (23) for the 3D `wedge` problem of Fig. 14. We indicate (approximate) slopes based on a linear fit of the convergence curves

$\mathcal{O}(n^5)$ for 3D problems. Moreover, we note that IDR outperforms BiCGStab in terms of number of iterations. The corresponding CPU times are presented in Table 7: From the previous analysis, a factor of ~ 32 for the overall CPU times is expected since the number of unknowns in three spatial directions is doubled (linear complexity yields a factor of 8), and Fig. 15 motivates an additional factor of 4 in iteration numbers.

Table 7 Total CPU times in seconds corresponding to the convergence plots in Fig. 15

frequency	BiCGStab	IDR(4)	IDR(16)
$f = 2$ Hz	144.5	95.5	91.3
$f = 4$ Hz	4430.7	3536.4	3100.5

Note that BiCGStab at $f = 4$ Hz is stopped after 1,000 iterations, cf. Fig. 15

6 Conclusions

We present an efficient *hybrid* method for the numerical solution of the inhomogeneous time-harmonic elastic wave equation. We use an incomplete block-LU factorization based on MSSS matrix computations as a preconditioner for IDR(s). The presented framework further allows to incorporate multiple wave frequencies and multiple source locations in a matrix equation setting (11). The suggested MSSS preconditioner is conceptional different for 2D and 3D problems:

- We derive an MSSS permutation matrix (17) that transforms the 2D elastic operator into block tridiagonal level-2 MSSS matrix structure. This allows the application of an approximate Schur factorization (20)-(21). In order to achieve linear computational complexity, the involved SSS operations (level-1) are approximated using model order reduction techniques that limit the off-diagonal rank.
- A generalization to 3D problems is not straight-forward because no model order reduction algorithms for level-2 MSSS matrices are currently available [29]. We therefore suggest the SSOR splitting (23) where off-diagonal blocks are treated as sparse matrices and diagonal blocks resemble a sequence of 2D problems in level-2 MSSS structure.

We present a series of numerical experiments on a 2D elastic `wedge` problem (Fig. 8) that prove theoretical concepts. In particular, we have numerically shown that a small off-diagonal rank $r^* \sim 10$ yields a preconditioner such that IDR(s) converges within very few iterations (Experiment 1).

Further numerical experiments for 2D elastic problems are performed on the realistic `Marmousi-II` data set. The newly derived matrix equation approach shows computational advantages when multiple right-hand sides (Experiment 5) and multiple frequencies (Experiment 6) are solved simultaneously.

In Corollary 2, we prove that the MSSS preconditioner has linear memory requirements for 2D and 3D problems. The overall computational complexity is investigated for the case of a constant number of wavelength, i.e., the number of grid points n in one spatial direction in linearly increased with the wave frequency. Numerical experiments show $\mathcal{O}(n^3)$ complexity for 2D (Experiment 3) and $\mathcal{O}(n^5)$ complexity for 3D (Experiment 7) problems. The 3D preconditioner solves a sequence of 2D problems and can be parallelized in a straight forward way.

Acknowledgments We would like to thank Joost van Zwieten, co-developer of the open source project `nutils`[2] for helpful discussions concerning the finite element discretization described in Section 2.2. Shell Global Solutions International B.V. is gratefully acknowledged for financial support of the first author.

[2] http://www.nutils.org/

Appendix

The appendix serves two purposes: We illustrate two basic SSS matrix operations used at 1D level by means of an example computation. At the same time, we complete Algorithm 2. For simplicity, we consider the case $n = 4$ in Definition 3,

$$A = \begin{bmatrix} D_1 & U_1V_2^{\mathsf{H}} & U_1W_2V_3^{\mathsf{H}} & U_1W_2W_3V_4^{\mathsf{H}} \\ P_2Q_1^{\mathsf{H}} & D_2 & U_2V_3^{\mathsf{H}} & U_2W_3V_4^{\mathsf{H}} \\ P_3R_2Q_1^{\mathsf{H}} & P_3Q_2^{\mathsf{H}} & D_3 & U_3V_4^{\mathsf{H}} \\ P_4R_3R_2Q_1^{\mathsf{H}} & P_4R_3Q_2^{\mathsf{H}} & P_4Q_3^{\mathsf{H}} & D_4 \end{bmatrix},$$

and refer to standard literature for the more general case.

A Inversion of a lower/upper diagonal SSS matrix

A lower diagonal SSS matrix in generator form is given by

$$L = SSS(P_s, R_s, Q_s, D_s, 0, 0, 0), \quad 1 \le s \le n, \tag{25}$$

and we denote L^{-1} via,

$$L^{-1} = SSS(\underline{\mathrm{P}}_s, \underline{\mathrm{R}}_s, \underline{\mathrm{Q}}_s, \underline{\mathrm{D}}_s, 0, 0, 0), \quad 1 \le s \le n.$$

Clearly, for $n = 4$, the matrix (25) yields,

$$L = \begin{bmatrix} D_1 & 0 & 0 & 0 \\ P_2Q_1^{\mathsf{H}} & D_2 & 0 & 0 \\ P_3R_2Q_1^{\mathsf{H}} & P_3Q_2^{\mathsf{H}} & D_3 & 0 \\ P_4R_3R_2Q_1^{\mathsf{H}} & P_4R_3Q_2^{\mathsf{H}} & P_4Q_3^{\mathsf{H}} & D_4 \end{bmatrix},$$

and we immediately conclude $\underline{\mathrm{D}}_s = D_s^{-1}$, $s = 1, ..., 4$, for all diagonal generators of L^{-1}. In Lemma 1, we claim that L^{-1} can be computed without increase of the off-diagonal rank, and we illustrate this fact by computing the generators at entry (2, 1):

$$P_2Q_1^{\mathsf{H}}\underline{\mathrm{D}}_1 + D_2\underline{\mathrm{P}}_2\underline{\mathrm{Q}}_1^{\mathsf{H}} = 0 \quad \Leftrightarrow \quad \underline{\mathrm{P}}_2\underline{\mathrm{Q}}_1^{\mathsf{H}} \equiv \left(-D_2^{-1}P_2\right)\left(D_1^{-\mathsf{H}}Q_1\right)^{\mathsf{H}}.$$

The computation of U^{-1} in Algorithm 2 can be done analogously, and we refer to [9, Lemma 2] for the complete algorithm and the case $n \neq 4$.

B Matrix-matrix multiplication in SSS structure

In the final step of Algorithm 2, we perform the matrix-matrix multiplication $A^{-1} = U^{-1} \cdot L^{-1}$ with U^{-1} and L^{-1} given in upper/lower diagonal SSS format, cf. Appendix A. In this section, we illustrate how to perform the SSS matrix-matrix multiplication $C = A \cdot B$ when $n = 4$ and A and B are given as,

$$A = \begin{bmatrix} D_1^A & U_1V_2^{\mathsf{H}} & U_1W_2V_3^{\mathsf{H}} & U_1W_2W_3V_4^{\mathsf{H}} \\ 0 & D_2^A & U_2V_3^{\mathsf{H}} & U_2W_3V_4^{\mathsf{H}} \\ 0 & 0 & D_3^A & U_3V_4^{\mathsf{H}} \\ 0 & 0 & 0 & D_4^A \end{bmatrix}, \quad \text{and}$$

$$B = \begin{bmatrix} D_1^B & 0 & 0 & 0 \\ P_2Q_1^{\mathsf{H}} & D_2^B & 0 & 0 \\ P_3R_2Q_1^{\mathsf{H}} & P_3Q_2^{\mathsf{H}} & D_3^B & 0 \\ P_4R_3R_2Q_1^{\mathsf{H}} & P_4R_3Q_2^{\mathsf{H}} & P_4Q_3^{\mathsf{H}} & D_4^B \end{bmatrix}.$$

The SSS matrix C can then be computed by appropriate block multiplications of the respective generators. For example, the (3, 2) entry of the product yields,

$$\begin{aligned} C_{32} &= 0 \cdot D_2^B + D_3^A P_3 Q_2^{\mathsf{H}} + U_3 V_4^{\mathsf{H}} P_4 R_3 Q_2^{\mathsf{H}} \\ &= \left(D_3^A P_3 + U_3 V_4^{\mathsf{H}} P_4 R_3\right) Q_2^{\mathsf{H}} \equiv P_3^C \left(Q_2^C\right)^{\mathsf{H}} \end{aligned}$$

The above computation illustrates on the one hand that the off-diagonal rank does not increase due to the lower/upper diagonal SSS structure of the matrices A and B. On the other hand, we note that in general the off-diagonal rank of C will increase due to the non-vanishing term that contains the full-rank generator D_2^B. Matrix-matrix multiplication in SSS form is presented in [9, Theorem 1].

References

1. Airaksinen, T., Pennanen, A., Toivanen, J.: A damping preconditioner for time-harmonic wave equations in fluid and elastic material. J. Comput. Phys. **228**(5), 1466–1479 (2009)
2. Amestoy, P., Ashcraft, C., Boiteau, O., Buttari, A., L'Excellent, J.Y., Weisbecker, C.: Improving multifrontal methods by means of block low-rank representations. SIAM J. Sci. Comput. **37**, A1451–A1474 (2015)
3. Amestoy, P., Brossier, R., Buttari, A., L'Excellent, J.Y., Mary, T., Métivier, L., Miniussi, A., Operto, S.: Fast 3D frequency-domain full-waveform inversion with a parallel block low-rank multifrontal direct solver: Application to OBC data from the North Sea. Geophysics **81**(6), R363–R383 (2016)
4. Astudillo, R., van Gijzen, M.B.: Induced dimension reduction method for solving linear matrix equations. Procedia Computer Science **80**, 222–232 (2016)
5. Baumann, M.: Two benchmark problems for the time-harmonic elastic wave equation in 2D and 3D. https://github.com/ManuelMBaumann/elastic_benchmarks (Sept. 2016). doi:10.5281/zenodo.154700

6. Baumann, M., van Gijzen, M.B.: Nested Krylov methods for shifted linear systems. SIAM J. Sci. Comput. **37**(5), S90–S112 (2015)
7. Baumann, M., van Gijzen, M.B.: An efficient two-level preconditioner for multi-frequency wave propagation problems. Tech. rep., DIAM Report 17-03 Delft University of Technology (2017)
8. Chandrasekaran, S., Dewilde, P., Gu, M., Pals, T., Sun, X., van der Veen, A., White, D.: Some fast algorithms for sequentially semiseparable representations. SIAM J. Matrix Anal. Appl. **27**(2), 341–364 (2005)
9. Chandrasekaran, S., Dewilde, P., Gu, M., Pals, T., van der Veen, A.J.: Fast Stable Solvers for Sequentially Semi-separable Linear Systems of Equations. Tech. rep., Lawrence Livermore National Laboratory (2003)
10. Cottrell, J.A., Hughes, T.J.R., Bazilevs, Y.: Isogeometric Analysis. Towards integration of CAD and FEA. John Wiley & Son Ltd. (2009)
11. De Basabe, J.: High-order Finite Element Methods for Seismic Wave Propagation. Ph.D. thesis The University of Texas at Austin (2009)
12. Dewilde, P., Van der Veen, A.: Time-Varying Systems and Computations. Kluwer Academic Publishers, Boston (1998)
13. Eidelman, Y., Gohberg, I.: On generators of quasiseparable finite block matrices. Calcolo **42**(3), 187–214 (2005)
14. Elman, H., Silvester, D., Wathen, A.: Finite elements and fast iterative solvers: With applications in incompressible fluid dynamics. Numerical Mathematics and Scientific Computation. Oxford University Press (2014)
15. Etienne, V., Chaljub, E., Virieux, J., Glinsky, N.: An hp-adaptive discontinuous Galerkin finite-element method for 3-D elastic wave modelling. Geophys. J. Int. **183**(2), 941–962 (2010)
16. van Gijzen, M.B., Sonneveld, P.: Algorithm 913: An Elegant IDR(s) Variant that Efficiently Exploits Bi-orthogonality Properties. ACM Trans. Math. Software **38**(1), 5:1–5:19 (2011)
17. Jbilou, K., Messaoudi, A., Sadok, H.: Global FOM and GMRES algorithms for matrix equations. Appl. Numer. Math. **31**, 49–63 (1999)
18. Kavcic, A., Moura, J.: Matrices with banded inverses: inversion algorithms and factorization of Gauss-Markov processes. IEEE Trans. Inform. Theory **46**(4), 1495–1509 (2000)
19. Knibbe, H., Vuik, C., Oosterlee, C.W.: Reduction of computing time for least-squares migration based on the Helmholtz equation by graphics processing units. Comput. Geosci. **20**(2), 297–315 (2016)
20. Liesen, J., Strakos, Z.: Krylov subspace methods: Principles and analysis. Numerical mathematics and scientific computation OUP Oxford (2013)
21. Martin, G.S., Marfurt, K.J., Larsen, S.: Marmousi-2: an Updated Model for the Investigation of AVO in Structurally Complex Areas. In: 72Nd Annual International Meeting, SEG, Expanded Abstract, pp. 1979–1982 (2002)
22. Mulder, W.A., Plessix, R.E.: How to choose a subset of frequencies in frequency-domain finite-difference migration. Geophys. J. Int. **158**, 801–812 (2004)
23. Petrov, P.V., Newman, G.A.: Three-dimensional inverse modelling of damped elastic wave propagation in the Fourier domain. Geophys. J. Int. **198**, 1599–1617 (2014)
24. Plessix, R.E.: A Helmholtz iterative solver for 3D seismic-imaging problems. Geophysics **72**(5), SM185–SM194 (2007)
25. Plessix, R.E.: Three-dimensional frequency-domain full-waveform inversion with an iterative solver. Geophysics **74**, WCC149–WCC157 (2009)
26. Plessix, R.E., Mulder, W.A.: Seperation-of-variables as a preconditioner for an iterative Helmholtz solver. Appl. Numer. Math. **44**, 385–400 (2004)
27. Plessix, R.E., Pérez Solano, C.A.: Modified surface boundary conditions for elastic waveform inversion of low-frequency wide-angle active land seismic data. Geophys. J. Int. **201**, 1324–1334 (2015)
28. Pratt, R.: Seismic waveform inversion in the frequency domain, Part 1: Theory and verification in a physical scale mode. Geophysics **64**(3), 888–901 (1999)
29. Qiu, Y., van Gijzen, M.B., van Wingerden, J.W., Verhaegen, M., Vuik, C.: Efficient Preconditioners for PDE-constrained Optimization Problems with a Multilevel Sequentially SemiSeparable Matrix Structure. Electron. Trans. Numer. Anal. **44**, 367–400 (2015)
30. Qiu, Y., van Gijzen, M.B., van Wingerden, J.W., Verhaegen, M., Vuik, C.: Evaluation of multilevel sequentially semiseparable preconditioners on CFD benchmark problems using incompressible flow and iterative solver software. Math. Methods Appl. Sci. **38** (2015)
31. Rice, J.: Efficient Algorithms for Distributed Control: a Structured Matrix Approach. Ph.D. thesis, Delft University of Technology (2010)
32. Rice, J., Verhaegen, M.: Distributed control: a sequentially Semi-Separable approach for spatially heterogeneous linear systems. IEEE Trans. Automat. Control **54**(6), 1270–1283 (2009)
33. Riyanti, C.D., Erlangga, Y.A., Plessix, R.E., Mulder, W.A., Vuik, C., Osterlee, C.: A new iterative solver for the time-harmonic wave equation. Geophysics **71**, E57–E63 (2006)
34. Rizzuti, G., Mulder, W.: Multigrid-based 'shifted-Laplacian' preconditioning for the time-harmonic elastic wave equation. J. Comput. Phys. **317**, 47–65 (2016)
35. Saad, Y.: SPARSEKIT: a Basic Tool Kit for Sparse Matrix Computations. Tech. Rep. University of Minnesota, Minneapolis (1994)
36. Saad, Y.: Iterative methods for sparse linear systems: Second edition. Society for Industrial and Applied Mathematics (2003)
37. Saad, Y., Schultz, M.: GMRES: A generalized minimal residual algorithm for solving nonsymetric linear systems. SIAM J. Sci. Stat. Comput. **7**, 856–869 (1986)
38. Saibaba, A., Bakhos, T., Kitanidis, P.: A flexible Krylov solver for shifted systems with application to oscillatory hydraulic tomography. SIAM J. Sci. Comput. **35**, 3001–3023 (2013)
39. Sleijpen, G.L.G., Sonneveld, P., van Gijzen, M.B.: BiCGStab as an induced dimension reduction method. Appl. Numer. Math. **60**, 1100–1114 (2010)
40. Sleijpen, G.L.G., van der Vorst, H.A.: Maintaining convergence properties of BiCGStab methods in finite precision arithmetic. Numer. Algorithms **10**, 203–223 (1995)
41. Sonneveld, P., van Gijzen, M.B.: IDR(S): a family of simple and fast algorithms for solving large nonsymmetric linear systems. SIAM J. Sci. Comput. **31**(2), 1035–1062 (2008)
42. Tsuji, P., Poulson, J., Engquist, B., Ying, L.: Sweeping preconditioners for elastic wave propagation with spectral element methods. ESAIM Math. Model. Numer. Anal. **48**(2), 433–447 (2014)
43. Vandebril, R., Van Barel, M., Mastronardi, N.: Matrix Computations and Semiseparable Matrices: Linear Systems. Johns Hopkins University Press, Baltimore (2007)
44. Virieux, J., Operto, S.: An overview of full-waveform inversion in exploration geophysics. Geophysics **73**(6), VE135–VE144 (2009)
45. van der Vorst, H.A.: BiCGStab: A Fast and Smoothly Converging Variant of bi-CG for the Solution of Nonsymmetric Linear Systems. SIAM J. Sci. Stat. Comput. **13**(2), 631–644 (1992)
46. Wang, S., de Hoop, M.V., Xia, J., Li, X.: Massively parallel structured multifrontal solver for time-harmonic elastic waves in 3-D anisotropic media. Geophys. J. Int. **191**(1), 346–366 (2012)
47. Xia, J.: Efficient structured multifrontal factorization for general large sparse matrices. SIAM J. Sci. Comput. **35**(2), A832–A860 (2013)