Karlsruhe Institute of Technology

wave phenomena
analysis and numerics

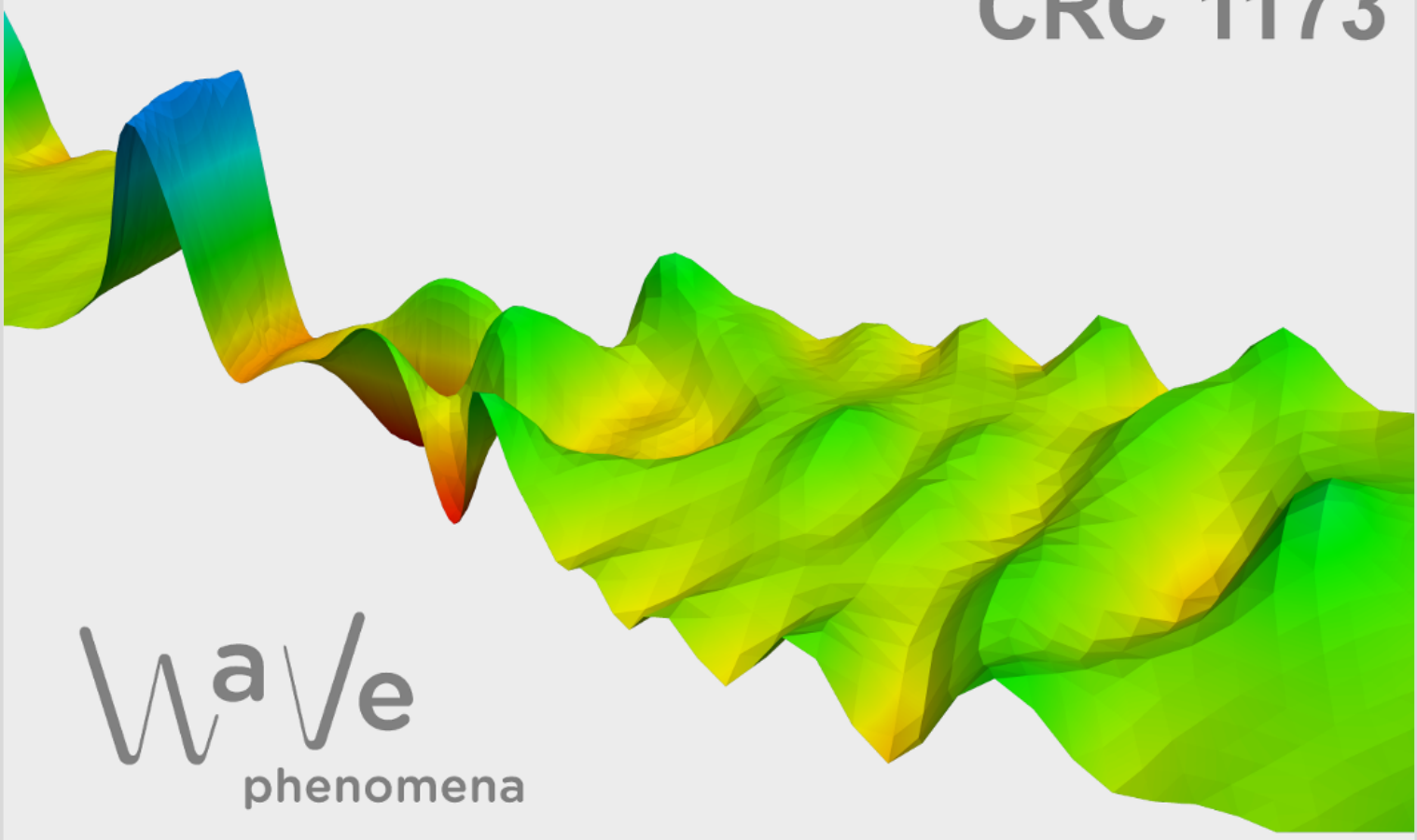
A non-iterative domain decomposition time integrator combined with discontinuous Galerkin space discretizations for acoustic wave equations

Tim Buchholz, Marlis Hochbruck

CRC Preprint 2025/51 (revised), April 2026

KARLSRUHE INSTITUTE OF TECHNOLOGY

CRC 1173



Participating universities

Funded by

ISSN 2365-662X

A non-iterative domain decomposition time integrator combined with discontinuous Galerkin space discretizations for acoustic wave equations

Tim Buchholz[0009−0000−0389−0983] and
Marlis Hochbruck[0000−0002−5968−0480]

Abstract We propose a novel non-iterative domain decomposition time integrator for acoustic wave equations using a discontinuous Galerkin discretization in space. It is based on a local Crank-Nicolson approximation combined with a suitable local prediction step in time. In contrast to earlier work using linear continuous finite elements with mass lumping, the proposed approach enables higher-order approximations and also heterogeneous material parameters in a natural way.

1 Introduction

We construct a novel non-iterative domain decomposition time integrator for acoustic wave equations which uses a discontinuous Galerkin (DG) space discretization. Employing a DG discretization offers two key advantages. First, it easily allows us to use higher-order polynomials on the mesh elements and second, it works very well for spatially varying material parameters, e.g., piecewise constant material coefficients modeling composite materials. The construction is inspired by the work of [2, 6] for parabolic problems and [3], where we proposed and analyzed this method for linear acoustic wave equations using a space discretization based on linear finite elements combined with mass lumping.

The linear acoustic wave equation is posed on an open, bounded, and polyhedral domain $\Omega \subset \mathbb{R}^d$ with a non-empty Dirichlet boundary $\Gamma_D \subseteq \partial\Omega$ and Neumann boundary $\Gamma_N = \partial\Omega \setminus \Gamma_D$. The material coefficient $\kappa \in L^\infty(\Omega)$ satisfies $\alpha < \kappa(x) < \beta$ almost everywhere for some constants $\alpha, \beta > 0$, and may in particular be piecewise

Tim Buchholz
Institute for Applied and Numerical Mathematics, Karlsruhe Institute of Technology, Englerstr. 2, 76131 Karlsruhe, Germany, e-mail: tim.buchholz@kit.edu

Marlis Hochbruck
Institute for Applied and Numerical Mathematics, Karlsruhe Institute of Technology, Englerstr. 2, 76131 Karlsruhe, Germany e-mail: marlis.hochbruck@kit.edu

constant. Let $Lu = \nabla \cdot (\kappa \nabla u)$ be the differential operator applied to a function u in the domain $D(L) = H^1(\Omega) \cap \{ u \in L^2(\Omega) \mid Lu \in L^2(\Omega) \}$. Given initial data $u^0 \in D(L)$ and $v^0 \in H^1(\Omega)$, the linear acoustic wave equation is given by

$$\partial_t u = v, \qquad \partial_t v = Lu + f, \quad \text{in } \Omega \times (0,T] \tag{1a}$$

$$u(x,0) = u^0(x), \quad v(x,0) = v^0(x), \quad \text{in } \Omega \tag{1b}$$

$$u = g, \qquad \text{on } \Gamma_D \times (0,T] \tag{1c}$$

$$\partial_n u = 0, \qquad \text{on } \Gamma_N \times (0,T] \tag{1d}$$

where $T > 0$ denotes the final time. For the inhomogeneity f we assume $f \in C([0,T], H^1(\Omega))$. However, the precise necessary conditions on g to get well-posedness of the problem are quite delicate, and lie outside the scope of this work. We just demand $g \in C^2([0,T], C^2(\Gamma_D))$, which is sufficient to lift the problem into one with homogeneous mixed boundary conditions. Moreover, we assume compatibility of g at the transition between Γ_D and Γ_N, as well as $u^0\big|_{\Gamma_D} = g\big|_{t=0}$.

The remainder of the paper is structured as follows. In Section 2, we review relevant preliminaries, including the space discretization, global time integrators, and cell extensions within a mesh. Section 3 presents the construction and details of the proposed domain splitting method. In Section 4, we highlight some key aspects to be considered for an efficient implementation. Finally, Section 5 presents numerical experiments that demonstrate the method's performance.

2 Preliminaries

2.1 Discretization in space

For the spatial discretization, we consider a shape- and contact-regular, matching simplicial mesh $\mathcal{T}_h = \mathcal{T}_h(\Omega)$ of the domain Ω, cf. [8, Definition 8.11]. We denote by $\mathcal{F}_h = \mathcal{F}_h(\Omega)$ the set of all mesh faces, decomposed into the set of boundary faces $\mathcal{F}_h^{\text{bnd}} = \mathcal{F}_h^{\text{bnd}}(\Omega)$ and the set of interior faces $\mathcal{F}_h^{\text{int}} = \mathcal{F}_h^{\text{int}}(\Omega)$. Further, we assume that $\partial\Omega$ is the distinct union of the Dirichlet boundary Γ_D and the Neumann boundary Γ_N and that all faces are contained either in Γ_D or in Γ_N. The wave propagation speed κ is piecewise constant, and the mesh $\mathcal{T}_h$ is matched to κ, i.e., $\kappa\big|_K$ is constant for every element $K \in \mathcal{T}_h$. For each element $K \in \mathcal{T}_h$, let h_K denote its diameter and h the maximal element diameter in the mesh.

For any subset $\widehat{\mathcal{T}_h} \subset \mathcal{T}_h$ of the mesh, we define the corresponding spatial domain

$$\widehat{\Omega} = \operatorname{dom}(\widehat{\mathcal{T}_h}) \coloneqq \operatorname{int} \bigcup_{K \in \widehat{\mathcal{T}_h}} \overline{K} \subset \Omega. \tag{2}$$

If a domain $\widehat{\Omega}$ is defined in this way as a union of cells, then the set of mesh elements belonging to $\widehat{\Omega}$ is denoted by $\mathcal{T}_h(\widehat{\Omega}) = \{K \in \mathcal{T}_h \mid K \subset \widehat{\Omega}\}$. We further denote by $\mathcal{F}_h(\widehat{\Omega})$ the set of all faces in $\mathcal{T}_h(\widehat{\Omega})$, which we split into the interior faces $\mathcal{F}_h^{\text{int}}(\widehat{\Omega})$

and boundary faces $\mathcal{F}_h^{\mathrm{bnd}}(\widehat{\Omega})$. The Dirichlet and Neumann parts of the boundary of $\widehat{\Omega}$ are denoted by $\widehat{\Gamma}_D$ and $\widehat{\Gamma}_N$, respectively. Analogously, for a generic interface $\widehat{\Gamma}$ we denote its associated set of faces by $\mathcal{F}_h(\widehat{\Gamma}) \subset \mathcal{F}_h$. As defined in [7, Definition 1.18] we denote the face normals of a given face $F \in \mathcal{F}_h$ by n_F.

Next, we introduce the discrete function spaces. For each element $K \in \mathcal{T}_h$, let $\mathbb{P}_k(K)$ denote the set of all polynomials in d variables of total degree at most k. The associated broken polynomial space on a domain $\widehat{\Omega}$ is then defined as

$$\mathcal{P}_k^b(\mathcal{T}_h(\widehat{\Omega})) \coloneqq \{\, \psi_h \in L^2(\widehat{\Omega};\mathbb{R}) \mid \psi_h\big|_K \in \mathbb{P}_k(K),\ \forall K \in \mathcal{T}_h(\widehat{\Omega}) \,\}.$$

This broken space serves as the approximation space in the discontinuous Galerkin method, allowing discontinuities of the functions across element interfaces. We denote the standard L^2 inner product on $\widehat{\Omega}$ by $(\cdot,\cdot)_{\widehat{\Omega}} = (\cdot,\cdot)_{L^2(\widehat{\Omega})}$. The corresponding L^2 projection $\Pi_{\widehat{\Omega}} : L^2(\widehat{\Omega}) \to \mathcal{P}_k^b(\mathcal{T}_h(\widehat{\Omega}))$ is defined for any $\phi \in L^2(\widehat{\Omega})$ by

$$\big(\Pi_{\widehat{\Omega}}\phi - \phi, \psi_h\big)_{\widehat{\Omega}} = 0, \quad \forall \psi_h \in \mathcal{P}_k^b(\mathcal{T}_h(\widehat{\Omega})). \tag{3}$$

We follow the symmetric weighted interior penalty (SWIP) approach as described in [7, Section 4.5.2.3], where jumps are denoted by $[\![\cdot]\!]$ and weighted averages by $\{\!\{\cdot\}\!\}^{\omega}$ with weights defined as in [7, Definition 4.46]. In this context, we also recall the definition of the local length scale h_F, cf. [7, Definition 4.5], the κ-dependent penalty parameter $\gamma_{\kappa,F}$ introduced in [7, below eq. (4.64)], and the penalty parameter $\eta > 0$. Denoting the broken gradient from [7, Definition 1.21] by ∇_h the SWIP bilinear form on $\widehat{\Omega}$ is given by

$$\begin{aligned} a_{\widehat{\Omega}}\big(\phi_h,\psi_h\big) = &\sum_{K\in\mathcal{T}_h(\widehat{\Omega})} \int_K \kappa \nabla_h\phi_h \nabla_h\psi_h \,\mathrm{d}x + \sum_{F\in\mathcal{F}_h(\widehat{\Omega})} \eta \frac{\gamma_{\kappa,F}}{h_F} \int_F [\![\phi_h]\!][\![\psi_h]\!] \,\mathrm{d}\sigma \\ &- \sum_{F\in\mathcal{F}_h(\widehat{\Omega})} \int_F \{\!\{\kappa\nabla_h\phi_h\}\!\}^{\omega} \cdot n_F [\![\psi_h]\!] + \{\!\{\kappa\nabla_h\psi_h\}\!\}^{\omega} \cdot n_F [\![\phi_h]\!] \,\mathrm{d}\sigma, \end{aligned}$$

for $\phi_h, \psi_h \in \mathcal{P}_k^b(\mathcal{T}_h(\widehat{\Omega}))$. We write $\|\cdot\|_{a,\widehat{\Omega}}^2$ for the associated SWIP norm, cf. [7, eq. (4.69)]. We now introduce the linear operator $\mathcal{L}_{h,\widehat{\Omega}} : \mathcal{P}_k^b(\mathcal{T}_h(\widehat{\Omega})) \to \mathcal{P}_k^b(\mathcal{T}_h(\widehat{\Omega}))$ associated with the SWIP bilinear form. It is defined by

$$\big(\mathcal{L}_{h,\widehat{\Omega}}\phi_h, \psi_h\big)_{\widehat{\Omega}} = a_{\widehat{\Omega}}\big(\phi_h,\psi_h\big), \quad \forall \phi_h, \psi_h \in \mathcal{P}_k^b(\mathcal{T}_h(\widehat{\Omega})). \tag{4}$$

For the case $\widehat{\Omega} = \Omega$, we write simply $\mathcal{L}_h = \mathcal{L}_{h,\Omega}$. For the treatment of inhomogeneous Dirichlet boundary conditions we refer to [7, Section 4.2.2]. Specifically, the boundary data g on a set of faces $\widehat{\Gamma}$ is weakly enforced by an additional term on the right-hand side of the discrete problem: We thus define

$$\big(\mathcal{G}_{\widehat{\Gamma}}(g), \psi_h\big)_{\widehat{\Omega}} = \sum_{F\in\mathcal{F}_h(\widehat{\Gamma})} \eta \frac{\gamma_{\kappa,F}}{h_F} \int_F g\psi_h \,\mathrm{d}\sigma - \int_F g\kappa\nabla_h\psi_h \cdot n \,\mathrm{d}\sigma, \tag{5}$$

for all $\psi_h \in \mathcal{P}_k^b(\mathcal{T}_h(\widehat{\Omega}))$. We abbreviate $\mathcal{G}_{\widehat{\Gamma}}(g^n) \coloneqq \mathcal{G}_{\widehat{\Gamma}}(g(t_n))$ for a given $t_n \in [0,T]$.

The initial values and the right-hand side are approximated using the L^2 projection onto the broken polynomial space. Specifically, on a domain $\widehat{\Omega}$ we set

$$u^0_{h,\widehat{\Omega}} = \Pi_{\widehat{\Omega}} u^0, \qquad v^0_{h,\widehat{\Omega}} = \Pi_{\widehat{\Omega}} v^0, \qquad f_{h,\widehat{\Omega}}(t) = \Pi_{\widehat{\Omega}} f(t). \tag{6}$$

Whenever the domain is clear from the context, we omit the subscript $\widehat{\Omega}$, and we also abbreviate $f^n_h = f_{h,\widehat{\Omega}}(t_n)$ for a given $t_n \in [0,T]$. With these definitions, the semi-discrete problem in $\mathcal{P}^b_k(\mathcal{T}_h(\widehat{\Omega})) \times \mathcal{P}^b_k(\mathcal{T}_h(\widehat{\Omega}))$ reads

$$\partial_t \begin{pmatrix} u_h \\ v_h \end{pmatrix} = \begin{pmatrix} 0 & I \\ -\mathcal{L}_{h,\widehat{\Omega}} & 0 \end{pmatrix} \begin{pmatrix} u_h \\ v_h \end{pmatrix} + \begin{pmatrix} 0 \\ f_{h,\widehat{\Omega}}(t) \end{pmatrix} + \begin{pmatrix} 0 \\ \mathcal{G}_{\widehat{\Gamma}_D}(g(t)) \end{pmatrix}. \tag{7}$$

2.2 Time integrators on $\widehat{\Omega}$

We consider a generic subdomain $\widehat{\Omega}$ and a given time-step size $\tau > 0$ with

$$t_n = n\tau, \qquad n = 0, \dots, N_T, \quad T = N_T \tau.$$

Starting from the semi-discrete problem (7), we present two standard second-order accurate time integration methods to obtain full-discretizations on $\widehat{\Omega} \times [0,T]$. As described in Section 2 the boundary data $g(t_n)$ is incorporated weakly through the operator $\mathcal{G}_{\widehat{\Gamma}_D}$ defined in (5). The **Crank-Nicolson** discretization of (7) is given by

$$u^{n+1}_h = u^n_h + \frac{\tau}{2}\big(v^{n+1}_h + v^n_h\big), \tag{8a}$$

$$\begin{aligned} v^{n+1}_h = v^n_h - \frac{\tau}{2}\mathcal{L}_{h,\widehat{\Omega}}\big(u^{n+1}_h + u^n_h\big) + \frac{\tau}{2}\big(f^{n+1}_h + f^n_h\big) \\ + \frac{\tau}{2}\big(\mathcal{G}_{\widehat{\Gamma}_D}(g^{n+1}) + \mathcal{G}_{\widehat{\Gamma}_D}(g^n)\big). \end{aligned} \tag{8b}$$

Moreover, we also consider the **leapfrog** method, which reads

$$v^{n+1/2}_h = v^n_h - \frac{\tau}{2}\mathcal{L}_{h,\widehat{\Omega}} u^n_h + \frac{\tau}{2} f^n_h + \frac{\tau}{2}\mathcal{G}_{\widehat{\Gamma}_D}(g^n) \tag{9a}$$

$$u^{n+1}_h = u^n_h + \tau v^{n+1/2}_h \tag{9b}$$

$$v^{n+1}_h = v^{n+1/2}_h - \frac{\tau}{2}\mathcal{L}_{h,\widehat{\Omega}} u^{n+1}_h + \frac{\tau}{2} f^{n+1}_h + \frac{\tau}{2}\mathcal{G}_{\widehat{\Gamma}_D}(g^{n+1}). \tag{9c}$$

In the discontinuous Galerkin setting, the leapfrog method has the advantage that the resulting mass matrix is block-diagonal. There, each block in the mass matrix corresponds to the degrees of freedom in a single cell $K \in \mathcal{T}_h$.

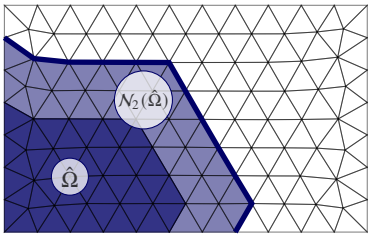


Fig. 1 Extension $\mathcal{N}_2(\widehat{\Omega})$ by $\ell = 2$ layers of a subdomain $\widehat{\Omega} \subset \Omega$ colored in dark blue.

2.3 Cell extensions

To define overlapping subdomains and cell neighborhoods, we introduce the concept of cell extensions. For a given subdomain $\widehat{\Omega} \subset \Omega$ and a fixed integer $\ell \geq 1$, we define its extension by ℓ layers of neighboring cells recursively as

$$\begin{aligned} \mathcal{N}_0(\widehat{\Omega}) &\coloneqq \mathcal{T}_h(\widehat{\Omega}), \\ \mathcal{N}_j(\widehat{\Omega}) &\coloneqq \{\, K \in \mathcal{T}_h(\Omega) \mid \exists K_\star \in \mathcal{N}_{j-1}(\widehat{\Omega}) \;:\; \overline{K} \cap \overline{K}_\star \neq \emptyset \,\}, \end{aligned} \tag{10}$$

for $j = 1, \ldots, \ell$, see also Figure 1. Similarly, for a generic interface $\widehat{\Gamma}$, we define the interface cell extension as

$$\mathcal{N}(\widehat{\Gamma}) \coloneqq \{\, K \in \mathcal{T}_h(\widehat{\Omega}) \mid \exists F_\star \in \mathcal{F}_h(\widehat{\Gamma}) \;:\; \overline{K} \cap F_\star \neq \emptyset \,\}, \tag{11}$$

which contains all elements which share a face or a corner with $\widehat{\Gamma}$.

3 Domain splitting method

To construct our new method, we first decompose the spatial domain Ω into $\mathcal{I}$ distinct, non-overlapping subdomains Ω_i, i.e.

$$\overline{\Omega} = \bigcup_{i=1}^{\mathcal{I}} \overline{\Omega}_i, \qquad \Omega_i \cap \Omega_j = \emptyset \quad \text{for } i \neq j \ . \tag{12}$$

Based on the definitions (2) and (10), we then introduce overlapping subdomains Ω_i^ℓ by extending each Ω_i with ℓ layers of elements

$$\Omega_i^\ell \coloneqq \operatorname{dom} \mathcal{N}_\ell(\Omega_i), \quad i = 1, \ldots \mathcal{I}. \tag{13}$$

Similarly, using (11), we define domains based on the cells around the interface $\Gamma_i^\ell = \partial\Omega_i^\ell \cap \Omega$

$$\Omega_{\Gamma,i}^\ell \coloneqq \operatorname{dom} \mathcal{N}(\Gamma_i^\ell), \quad i = 1, \ldots \mathcal{I}, \tag{14}$$

which we will refer to as prediction domain for the interface Γ_i^ℓ of Ω_i^ℓ. An overview of this subdomain notation is given in Figure 2. Let δ be the minimal physical width of the overlap $\Omega_i^\ell \setminus \overline{\Omega}_i$, which satisfies $\delta \sim \ell h$, given the underlying mesh $\mathcal{T}_h$ is

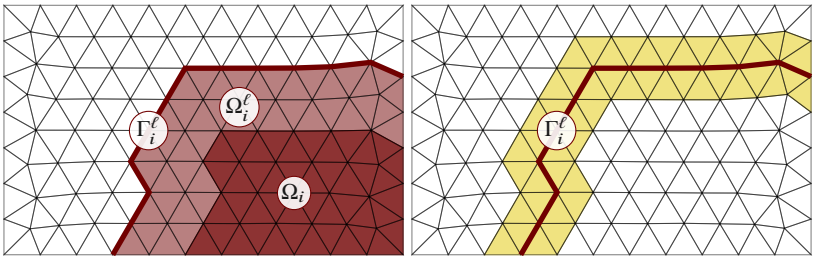


Fig. 2 Overlapping subdomain Ω_i^ℓ (left, dark and light red area) and prediction domain $\Omega_{\Gamma,i}^\ell$ (right, yellow area). The interface Γ_i^ℓ is shown in dark red in both pictures.

shape- and contact-regular. We denote by $\Gamma_{i,D}^\ell$ the portion of the boundary of Ω_i^ℓ that coincides with the Dirichlet boundary of Ω, i.e.,

$$\Gamma_{i,D}^\ell = \Gamma_D \cap \overline{\Omega}_i^\ell .$$

Next, we describe the construction of the domain-splitting approximations

$$x_{\mathrm{DS}}^n = \begin{pmatrix} u_{\mathrm{DS}}^n \\ v_{\mathrm{DS}}^n \end{pmatrix}, \qquad \text{for } n = 0, \dots, N_T. \tag{15}$$

For the initial values we use L^2 projections

$$u_{\mathrm{DS}}^0 = u_h^0 = \Pi_\Omega u^0, \qquad v_{\mathrm{DS}}^0 = v_h^0 = \Pi_\Omega v^0. \tag{16}$$

Note, that we can also replace this by local L^2 projections on the subdomains

$$u_{\mathrm{DS}}^0\big|_{\overline{\Omega}_i^\ell} = \big(\Pi_\Omega u^0\big)\big|_{\overline{\Omega}_i^\ell} = \Pi_{\Omega_i^\ell} u^0 ,$$

since we are using a discontinuous Galerkin discretization. Given an approximation x_{DS}^n at time t_n, the method advances to the next time step as described below:

First, we loop over $i = 1, \dots, \mathcal{I}$ and perform for each subdomain:

Prediction on strip $\Omega_{\Gamma,i}^\ell$ of the interface Γ_i^ℓ

A leapfrog step is carried out on $\Omega_{\Gamma,i}^\ell$ to compute an approximation $u_\star^{n+1}$:

$$v_\star^{n+1/2} = v_{\mathrm{DS}}^n\big|_{\overline{\Omega}_{\Gamma,i}^\ell} - \frac{\tau}{2}\mathcal{L}_{h,\Omega_{\Gamma,i}^\ell} u_{\mathrm{DS}}^n\big|_{\overline{\Omega}_{\Gamma,i}^\ell} + \frac{\tau}{2} f_h^n\big|_{\overline{\Omega}_{\Gamma,i}^\ell} + \frac{\tau}{2}\mathcal{G}_{\Gamma_D \cap \overline{\Omega}_{\Gamma,i}^\ell}(g^n) \tag{17a}$$

$$u_\star^{n+1} = u_{\mathrm{DS}}^n\big|_{\overline{\Omega}_{\Gamma,i}^\ell} + \tau v_\star^{n+1/2} \tag{17b}$$

Using $u_\star^{n+1}$ we define a boundary term $\mathcal{G}_{\Gamma_i^\ell}(\{\!\{u_\star^{n+1}\}\!\}^\omega)$ on the interface Γ_i^ℓ.

Local calculation on Ω_i^ℓ

On Ω_i^ℓ, we perform a Crank-Nicolson step with weakly imposed interface and boundary data. This results in a boundary term of the form

$$\mathcal{G}_{\Omega_i^\ell}^{n+1,n} \coloneqq \frac{\tau}{2}\Big(\mathcal{G}_{\Gamma_i^\ell}(\{\!\{u_\star^{n+1}\}\!\}^\omega) + \mathcal{G}_{\Gamma_{i,D}^\ell}(g^{n+1}) + \mathcal{G}_{\Gamma_i^\ell}(\{\!\{u_{\mathrm{DS}}^n\}\!\}^\omega) + \mathcal{G}_{\Gamma_{i,D}^\ell}(g^n)\Big). \tag{18a}$$

The resulting subdomain approximations x_i^{n+1} on $\overline{\Omega}_i^\ell$ are then given by

$$u_i^{n+1} = u_{\mathrm{DS}}^n\big|_{\overline{\Omega}_i^\ell} + \frac{\tau}{2}\big(v_i^{n+1} + v_{\mathrm{DS}}^n\big|_{\overline{\Omega}_i^\ell}\big), \tag{18b}$$

$$v_i^{n+1} = v_{\mathrm{DS}}^n\big|_{\overline{\Omega}_i^\ell} - \frac{\tau}{2}\mathcal{L}_{h,\Omega_i^\ell}\big(u_i^{n+1} + u_{\mathrm{DS}}^n\big|_{\overline{\Omega}_i^\ell}\big) + \frac{\tau}{2}\big(f_h^{n+1} + f_h^n\big)\big|_{\overline{\Omega}_i^\ell} + \mathcal{G}_{\Omega_i^\ell}^{n+1,n}. \tag{18c}$$

Note, that (17) and (18) do not couple between subdomains and can thus both be performed in parallel across all subdomains. Then, after completion of the loop over all subdomains we exchange data among neighboring subdomains:

Exchange approximations in overlap regions

If two subdomains Ω_i, Ω_j are adjacent, i.e., $S_{ij} = \Omega_i^\ell \cap \Omega_j^\ell \neq \emptyset$, then the local approximations are exchanged across the overlap. For the efficiency, it is important that this does not require global communication. Specifically, we replace

$$x_i^{n+1}\big|_{S_{ij}\cap\overline{\Omega}_j} \quad \text{by} \quad x_j^{n+1}\Big|_{S_{ij}\cap\overline{\Omega}_j} \qquad \text{and} \qquad x_j^{n+1}\Big|_{S_{ij}\cap\overline{\Omega}_i} \quad \text{by} \quad x_i^{n+1}\big|_{S_{ij}\cap\overline{\Omega}_i}\,.$$

We store these updated values in

$$x_{\mathrm{DS}}^{n+1}\big|_{\overline{\Omega}_i^\ell} \quad \text{and} \quad x_{\mathrm{DS}}^{n+1}\big|_{\overline{\Omega}_{\Gamma,i}^\ell} \tag{19}$$

locally on each subdomain for $i = 1, \dots, \mathcal{I}$ for the next time step.

These local updates are equivalent to assembling the global approximation via

$$x_{\mathrm{DS}}^{n+1} \coloneqq \sum_{i=1}^{\mathcal{I}} x_i^{n+1}\big|_{\overline{\Omega}_i}, \tag{20}$$

where we sum up the local approximations restricted to $\overline{\Omega}_i$. Note however, that it is not necessary to act globally here. Moreover, unlike in the mass-lumped finite element setting (cf. [3]), no averaging at the non-overlapping interfaces is required. We summarize the method in Algorithm 3.1.

Algorithm 3.1 Domain splitting for DG discretizations (one time step)

{given $x^n_{\mathrm{DS}}|_{\overline{\Omega}_i^\ell}$ and $x^n_{\mathrm{DS}}|_{\overline{\Omega}_{\Gamma,i}^\ell}$}
for $i = 1 : \mathcal{I}$ **do**
  {on each subdomain}
  calculate prediction $u_\star^{n+1}$ by leapfrog step (17) on $\Omega_{\Gamma,i}^\ell$
  calculate x_i^{n+1} by Crank-Nicolson step (18) on Ω_i^ℓ with boundary term $\mathcal{G}_{\Omega_i^\ell}^{n+1,n}$ (18a)
end for
update overlap regions with values from $x_i^{n+1}|_{\overline{\Omega}_i}$, $i = 1, \dots, \mathcal{I}$
→ prepare $x^{n+1}_{\mathrm{DS}}|_{\overline{\Omega}_i^\ell}$ and $x^{n+1}_{\mathrm{DS}}|_{\overline{\Omega}_{\Gamma,i}^\ell}$ for the next step
{only if desired build a global approximation x^{n+1}_{DS} on Ω by (20)}

4 Implementation

The method is implemented within the FEniCSx framework [1], using its PETSc interface via petsc4py [5]. The global mesh is generated with Gmsh [9], which is subsequently partitioned into non-overlapping subdomains using PT-Scotch [4], cf. Figure 6. For each subdomain, we construct the corresponding local meshes of the overlapping subdomain Ω_i^ℓ and the leapfrog prediction domain $\Omega_{\Gamma,i}^\ell$, as illustrated in Figure 2. The implementation is consistently based on these local meshes $\mathcal{T}_h(\Omega_i^\ell)$ and $\mathcal{N}(\Gamma_i^\ell)$, while direct access to the global mesh $\mathcal{T}_h(\Omega)$ is avoided whenever possible.

Communication between subdomains is performed using point-to-point MPI routines. Since a subdomain can only communicate with one neighbor at a time, the exchanges must be organized into rounds in order to avoid deadlocks. To structure these rounds, we construct a weighted undirected graph whose nodes correspond to subdomains and whose edges represent the messages (value exchanges) between neighboring subdomains; the weights encode the number of values to be transmitted, see Figure 3.

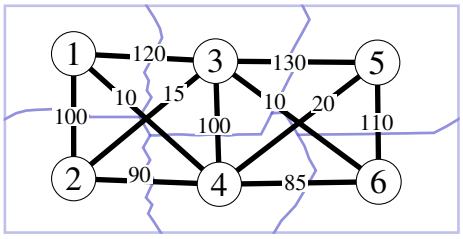


Fig. 3 Example for a weighted undirected communication graph for $\mathcal{I} = 6$ subdomains. Weights correspond to the number of values, which need to get exchanged between two subdomains.

The sequence of communication rounds is determined by a greedy algorithm with two objectives: minimizing the total number of rounds and balancing the message sizes within each round. To this end, the edges are first sorted by message size and then by the maximal node degree (i.e., the largest number of edges incident to either endpoint). Rounds are then built by selecting edges from the front of the sorted list, ensuring that no node appears more than once per round. This procedure yields a communication schedule that is both efficient and well-balanced.

For the example graph in Figure 3, the algorithm produces the following sequence of communication rounds:

$$
\begin{aligned}
1: &\quad (3,5,130), \quad (2,4,90)\\
2: &\quad (1,3,120), \quad (4,6,85)\\
3: &\quad (3,4,100), \quad (5,6,110), \ (1,2,100)\\
4: &\quad (4,5,20), \quad (2,3,15)\\
5: &\quad (1,4,10), \quad (3,6,10)
\end{aligned}
$$

Here, a tuple (i,j,M) denotes an exchange of M values between subdomains i and j.

The exchange of values in the overlap regions requires particular care in the implementation. To organize this process, we construct a dofmap that associates local DoFs with their corresponding global DoFs in $\mathcal{T}_h(\Omega)$, see Figure 4. This mapping allows us to identify exactly which DoFs in a subdomain require updated values from which neighbor. Moreover, it enables us to directly relate the DoFs of one subdomain to the DoFs of another subdomain, so that the communication can be set up locally and efficiently between neighboring subdomains.

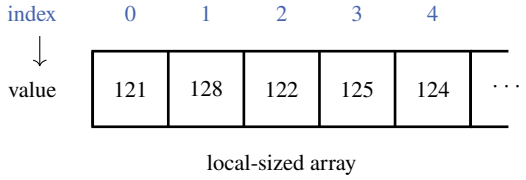


Fig. 4 For each local mesh we build a 'dofmap', which maps the local DoFs to the global DoFs in $\mathcal{T}_h(\Omega)$. Here the index denotes the local DoF, while the value stores the global DoF. Note, that the size of the map is only related to the local mesh.

These preparatory steps — global meshing, partitioning, dofmap setup, and scheduling — are carried out once in a preprocessing phase. During the actual simulation, communication then proceeds directly between neighboring subdomains, without any further global operations on $\mathcal{T}_h(\Omega)$, except when a global reconstruction as in (20) is explicitly required.

5 Numerical experiments

We simulate the propagation of a linear wave crossing a triangular inclusion, representing the cross-section of a prism, in two dimensions. The example features a spatially varying coefficient $\kappa(x)$ and mixed inhomogeneous boundary conditions. The computational domain is $\overline{\Omega} = [0,8] \times [0,4] \subset \mathbb{R}^2$, with a piecewise constant coefficient κ taking values $\kappa_i = 1.0$ inside of the prism and $\kappa_o = 1.5$ outside, corresponding to a refractive index of 1.5 which is typical for glass. An inhomogeneous Dirichlet condition is imposed on the inflow boundary $\Gamma_D = \partial\Omega\big|_{x=0}$ to generate an incoming wave of frequency $\omega = 0.0125$, described by

$$g(x, y, t) = \sin\left(\frac{t}{\omega}\right) W(y),$$

where $W \in C^2(\mathbb{R})$ satisfies $W(y) = 1$ for $y \in [1, 3]$ and $W(y) = 0$ for $y \in [0, 0.5] \cup [3.5, 4.0]$. Homogeneous Neumann conditions $\partial_n u = 0$ are applied on $\Gamma_N = \partial\Omega \setminus \Gamma_D$.

The mesh $\mathcal{T}_h(\Omega)$ is generated in Gmsh [9] with a local refinement (factor 2) around the prism and partitioned into eight subdomains using PT-Scotch [4], cf. Figures 5 and 6. The simulation runs over $[0, T]$ with $T = 3.0$.

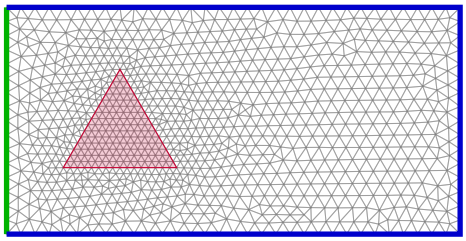


Fig. 5 Example grid on $[0.8] \times [0, 4]$ for the prism example with quite coarse h. Red area $\kappa(x) = \kappa_i$, white area $\kappa = \kappa_o$. Dirichlet boundary Γ_D (green) and Neumann boundary Γ_N (blue). For simulations much finer h were used.

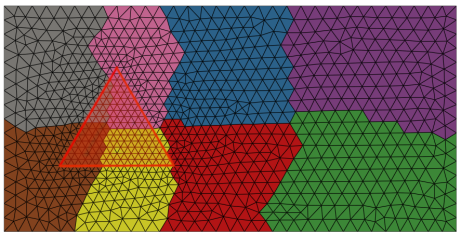


Fig. 6 Mesh from Figure 5 partitioned into 8 non-overlapping subdomains using the PT-Scotch partitioner [4]. The red area depicts the prism, where $\kappa(x) = \kappa_i$.

To evaluate the accuracy of the domain splitting method, we compare its results with a reference solution obtained from a leapfrog simulation on a refined mesh with 1,163,020 cells. A DG discretization with polynomial degree $p = 2$ yields 6,978,120 DoFs per component for the reference mesh. The domain splitting method is applied on a coarser mesh with 812,298 cells (4,873,788 DoFs per component, $h_{\min} = 0.00446$) using 8 subdomains. We consider different overlap parameters $\ell \in \{2, 4, 8\}$, denoted by DS_2, DS_4, and DS_8, respectively. For comparison, a global Crank-Nicolson (CN) simulation is performed on the same mesh. The relative L^2-error of the u-component, $\|u_{\mathrm{DS}} - u_{\mathrm{lf}}\|_{L^2(\Omega)} / \|u_{\mathrm{lf}}\|_{L^2(\Omega)}$ measured at $T = 3.0$, is shown in Figure 7. The results indicate that the CFL condition is linear dependent on ℓ analogously to the findings in [3]. However, this has to be rigorously analyzed in future work.

Next, we compare the domain splitting approximation directly to the global Crank-Nicolson (CN) solution. Both methods are run on the same mesh with 4,873,788 DoFs and $h_{\min} = 0.00446$. For the CN reference, we employ a Cholesky decomposition (chol), while the domain splitting systems are solved iteratively using a conjugate gradient (cg) method preconditioned with an incomplete Cholesky (icc) factorization. For this problem size, no speedup was observed when solving the global CN

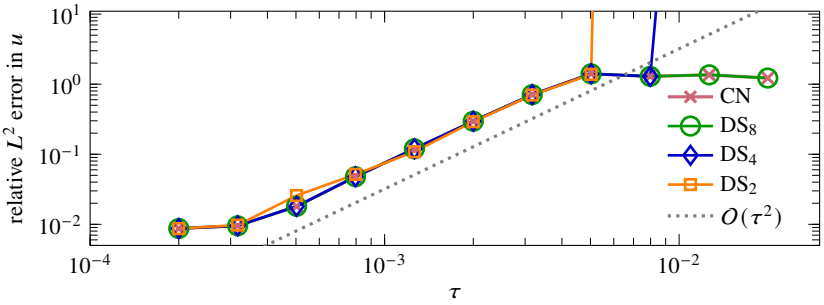
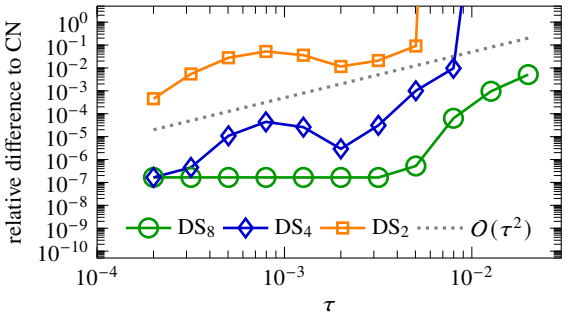


Fig. 7 Relative L^2-error of u at final time $T = 3.0$ for the domain splitting method (DS$_\ell$) with different overlap parameter ℓ, compared to a reference leapfrog solution on a refined mesh. The Crank-Nicolson method (CN) is shown for comparison.

system iteratively. The relative difference between both solutions is measured in the combined $\|\cdot\|_{a,\Omega} \times \|\cdot\|_{L^2(\Omega)}$ norm; see Figure 8.

Fig. 8 Difference of domain splitting approximation to Crank-Nicolson approximation at the end time $T = 3.0$. We measure the first component in $\|\cdot\|_{a,\Omega}$ and the second component in $\|\cdot\|_{L^2(\Omega)}$.



Finally, Table 1 reports run times and solver performance for DS$_4$ and the global Crank-Nicolson (CN) method, parallelized via the MPI interfaces of FEniCSx and petsc4py. All simulations were carried out on the same workstation using 8 cores, a time step $\tau = 0.001$, and the mesh from Figure 8. A visualization of both solutions at $T = 3.0$ is given in Figure 9. The results indicate that the proposed domain splitting method attains accuracy comparable to global time integration while providing structural benefits for parallelization.

	DS$_4$ (chol)	DS$_4$ (cg + icc)	CN (chol)
rel. L^2 error in u against ref	7.545e-2	7.545e-2	7.547e-2
rel. difference to CN (chol)	3.639e-5	3.639e-5	–
time for meshing	94.9 s	94.9 s	70.8 s
setup time for solvers	112.2 s	0.17 s	47.9 s
wall time per time step	0.94 s	1.21 s	1.52 s
total simulation time	3021 s	3739 s	4669 s

Table 1 Detailed comparison between the domain splitting method with different solver configurations and the global (parallelized) Crank-Nicolson method for a simulation with 3000 time steps.

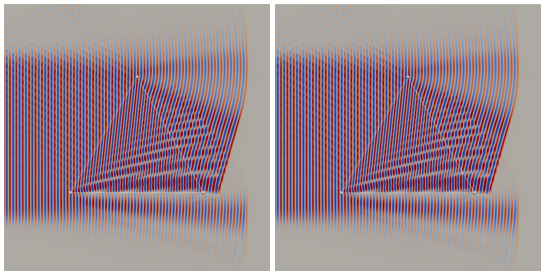


Fig. 9 Snapshots of the domain splitting approximation DS_4 (chol, left) and Crank-Nicolson approximation (chol, right) at the end time $T = 3.0$ on the subarea $[0, 4] \times [0, 4]$.

A more detailed theoretical study and large-scale experiments are planned as part of future work. The code corresponding to this section is made publicly available at

https://github.com/tim-buchholz/dsdg-acoustic-wave.git.

Acknowledgements This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) — Project-ID 258734477 — CRC 1173. The authors acknowledge support by the state of Baden-Württemberg through bwHPC.

Competing Interests The authors have no conflicts of interest to declare that are relevant to the content of this chapter.

References

1. Baratta, I.A., Dean, J.P., Dokken, J.S., Habera, M., Hale, J.S., Richardson, C.N., Rognes, M.E., Scroggs, M.W., Sime, N., Wells, G.N.: DOLFINx: the next generation FEniCS problem solving environment. preprint (2023)
2. Blum, H., Lisky, S., Rannacher, R.: A domain splitting algorithm for parabolic problems. Computing **49**(1), 11–23 (1992)
3. Buchholz, T., Hochbruck, M.: A non-iterative domain decomposition time integrator for linear wave equations. preprint (2025)
4. Chevalier, C., Pellegrini, F.: PT-Scotch: a tool for efficient parallel graph ordering. Parallel Comput. **34**(6-8), 318–331 (2008)
5. Dalcin, L.D., Paz, R.R., Kler, P.A., Cosimo, A.: Parallel distributed computing using python. Adv. Water Resour. **34**(9), 1124 – 1139 (2011)
6. Dawson, C.N., Dupont, T.F.: Explicit/implicit conservative Galerkin domain decomposition procedures for parabolic problems. Math. Comput. **58**(197), 21–34 (1992)
7. Di Pietro, D.A., Ern, A.: Mathematical aspects of discontinuous Galerkin methods, *Mathématiques & Applications*, vol. 69. Springer, Heidelberg (2012)
8. Ern, A., Guermond, J.L.: Finite elements I—Approximation and interpolation, *Texts in Applied Mathematics*, vol. 72. Springer, Cham (2021)
9. Geuzaine, C., Remacle, J.F.c.: Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities. Int. J. Numer. Methods Eng. **79**(11), 1309–1331 (2009)