Karlsruhe Institute of Technology

wave phenomena
analysis and numerics

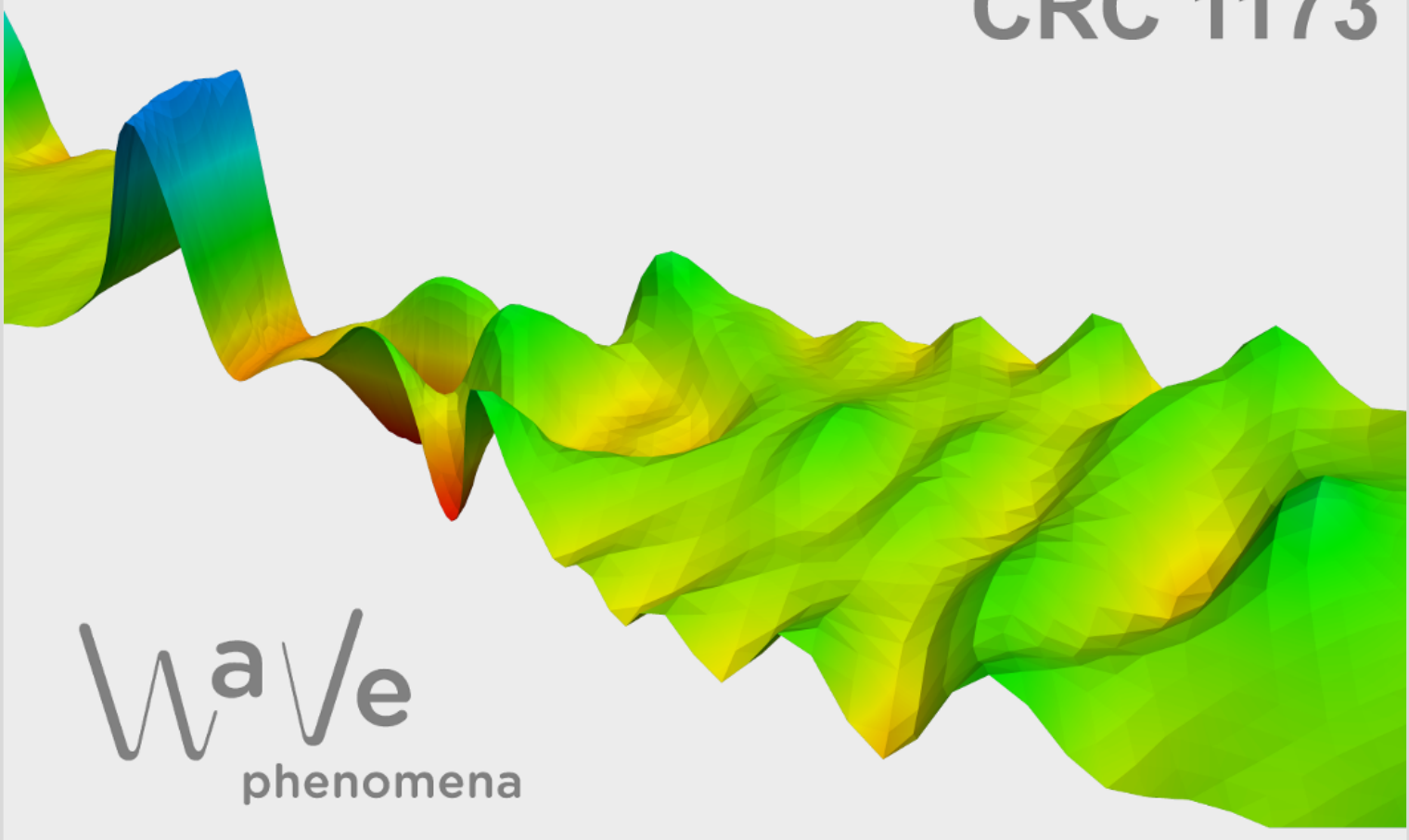
Multi-level local time-stepping methods of Runge-Kutta type for wave equations

Martin Almquist, Michaela Mehlin

CRC Preprint 2016/16, July 2016

KARLSRUHE INSTITUTE OF TECHNOLOGY

CRC 1173



Participating universities

Universität Stuttgart

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Funded by

ISSN 2365-662X

MULTI-LEVEL LOCAL TIME-STEPPING METHODS OF RUNGE–KUTTA TYPE FOR WAVE EQUATIONS *

MARTIN ALMQUIST † AND MICHAELA MEHLIN ‡

Abstract. Local mesh refinement significantly influences the performance of explicit time-stepping methods for numerical wave propagation. Local time-stepping (LTS) methods improve the efficiency by using smaller time-steps precisely where the smallest mesh elements are located, thus permitting a larger time-step in the coarser regions of the mesh without violating the stability condition. However, when the mesh contains nested patches of refinement, any local time-step will be unnecessarily small in some regions. To allow for an appropriate time-step at each level of mesh refinement, multi-level local time-stepping (MLTS) methods have been proposed. Starting from the Runge–Kutta-based LTS methods derived by Grote et al. [17], we propose explicit MLTS methods of arbitrarily high accuracy. Numerical experiments with finite difference and continuous finite element spatial discretizations illustrate the usefulness of the novel MLTS methods and show that they retain the high accuracy and stability of the underlying Runge–Kutta methods.

Key words. finite element methods, SBP-SAT finite differences, explicit time integration, local time-stepping, multi-level, multirate methods, hyperbolic problems

AMS subject classifications. 65M60, 65L06

1. Introduction.

Wave type phenomena are common in many fields of science, such as seismology, acoustics, and electromagnetics. The propagation of waves is often modeled by partial differential equations (PDEs), for which it is important to have accurate and efficient numerical solvers. In the presence of small geometric features or re-entrant corners in the spatial domain [33], locally refined meshes around the obstacles allow for accurate simulation without introducing too many spatial unknowns and are thus computationally efficient. Local mesh refinement, however, comes at a high price for explicit time integration as the stability restriction on the time-step depends on the smallest mesh-size. If the locally refined region is small compared to the entire computational domain, the costs of using a tiny time-step or an implicit scheme everywhere are too high. Instead, one might hope to use different time-steps or a combination of implicit and explicit schemes. Such strategies are not new in the community of ordinary differential equations (ODEs). Multirate methods [35, 15, 19, 9, 23, 10] and so-called IMEX (implicit-explicit) schemes [2, 24] are established methods when coming across naturally split systems of equations, where the different components evolve on different time-scales.

Over the last decades various local time-stepping (LTS) methods have been developed in the PDE community to deal with the time-step restriction caused by local mesh refinement. LTS methods utilize that the ODE in time comes from the spatial discretization of a PDE by dividing the spatial mesh into two distinct regions, "fine" and "coarse". One class of LTS methods is *locally implicit schemes*, which use an implicit scheme in the fine part of the mesh while remaining explicit in the coarse part. Thus, they retain one global time-step dictated by the mesh-size in the coarse region. Popular second order schemes and their analysis can be found in [34, 39, 12, 21], and

*M. Mehlin gratefully acknowledges financial support by the Deutsche Forschungsgemeinschaft (DFG) through CRC 1173 and by the Swiss National Science Foundation.

†Division of Scientific Computing, Department of Information Technology, Uppsala University, SE-751 05 Uppsala, Sweden, (martin.almquist@it.uu.se).

‡Institute for Applied and Numerical Analysis, Karlsruhe Institute of Technology, 76128 Karlsruhe, Germany,(michaela.mehlin@kit.edu).

higher order methods are presented in [25, 6]. Another LTS approach is fully *explicit local time-stepping methods*, which overcome the severe stability restriction caused by the locally refined mesh without solving linear systems on the fine part. Instead, they use a smaller time-step precisely where the small elements are located [8, 34, 13, 16].

In the presence of hierarchical meshes with several levels of refinement, the distinction between coarse and fine elements is no longer sufficient. If, for example, a locally refined region contains a subregion with even smaller grid sizes, it is no longer clear how to divide the unknowns into coarse and fine. Placing mesh elements with medium size diameter in the fine region increases the cost there as they are advanced with an unnecessarily small time-step. Counting those unknowns as coarse, however, will require an inappropriately small time-step in the coarse region. Ideally, one would like to use more than one local time-step. Based on the arbitrary high-order derivatives the DG ADER approach [38] allows every element to have its own time-step dictated by its mesh size. When considering other spatial discretization techniques than discontinuous Galerkin (DG) methods, multi-level schemes appeared in recent years, where the mesh is sorted into several levels with an appropriate time-step on each. In [1] Angulo et al. proposed a causal-path LTS technique for both the second-order leapfrog (LF) and a fourth-order low-storage (LS) explicit Runge–Kutta (RK) method [5] applied to Maxwell's equations, which sorts elements in tiers according to their size and computes required intermediate values recursively. Recently, Diaz and Grote extended their energy-conserving LTS-LF scheme [13] for the acoustic wave equation into a multi-level version [14]. In [36] Rietmann et al. derived a new LTS method based on the Newmark scheme, which also can be extended to accommodate multiple levels of mesh refinement. It is further shown how performance and scalability compare between different partitioning tools on CPU and GPU clusters using examples from computational seismology.

Herein we propose multi-level methods based on the LTS-RK schemes derived by Grote et al. in [17]. The multi-level LTS-RK (MLTS-RK) schemes allow for arbitrarily many local time-steps. They maintain the one-step nature of the underlying RK method and thus require no starting procedure and allow for adaptivity in time. This novel approach further retains the explicitness and accuracy of the underlying RK scheme.

The rest of the paper is structured as follows. As a model problem we consider in Section 2 the damped second order wave equation. Following the method-of-lines approach, we discretize in space using both continuous finite element methods (FEM) with mass-lumping and Summation-By-Parts–Simultaneous Approximation Term finite difference (SBP-SAT FD) methods. We then briefly recall the LTS-RK method in Section 3 and combine it with SBP-SAT FD schemes. In Section 4, we extend the LTS-RK scheme to cases with several levels of mesh refinement and describe how to implement the MLTS-RK algorithm. In Section 5 we prove that, like the LTS-RK scheme, the multi-level variant retains the accuracy of the underlying RK method. We also discuss the computational cost of the developed algorithm. We conclude this article by presenting numerical experiments where we combine the MLTS-RK methods with both continuous FEM and SBP-SAT FD in one and two space dimensions.

2. Spatial Discretization.

We consider the wave equation,

$$
\begin{aligned}
&u_{tt} + \sigma u_t - \nabla \cdot \left(c^2 \nabla u\right) = f, && \text{in } \Omega \times (0,T),\\
&u = g, && \text{on } \partial\Omega \times (0,T),\\
&u(\cdot,0) = u_0, \quad u_t(\cdot,0) = v_0, && \text{in } \Omega,
\end{aligned}
\tag{1}
$$

as a model problem. Here Ω is a bounded domain in $\mathbb{R}^d$, $f = f(\mathbf{x}, t)$ is a source term, and u_0 and v_0 are initial data. At the boundary, $\partial\Omega$, we impose a Dirichlet condition; g is the boundary data. The wave speed $c = c(\mathbf{x})$ is piecewise smooth and strictly positive ($c(\mathbf{x}) \geq c_0 > 0$). The damping coefficient $\sigma = \sigma(\mathbf{x})$ is assumed non-negative ($\sigma \geq 0$). If $\sigma = 0$ in all of Ω, then (1) coincides with the classical (undamped) wave equation.

We will follow the method-of-lines strategy by discretizing (1) in space while leaving time continuous. This leads to a system of ODEs that we subsequently wish to solve using a time-stepping algorithm. In this chapter we present two different methods for the spatial discretization of (1): continuous FE with mass-lumping and SBP-SAT FD. We argue that both approaches are viable for the wave equation. High-order FD methods are generally considered efficient for wave propagation problems on simple to moderately complex domains, where it is feasible to generate high-quality structured grids. In highly complex geometries however, mesh generation is difficult enough that the ability to use an unstructured mesh becomes a large advantage, making FEM an attractive approach.

2.1. SBP-SAT finite difference discretization. To introduce the SBP-SAT finite difference method [37] we consider (1) in one spatial dimension,

$$
\begin{aligned}
&u_{tt} + \sigma u_t - \left(c^2 u_x\right)_x = f, && x \in [x_L, x_R], && t \in (0, T),\\
&u = g_L, && x = x_L, && t \in (0, T),\\
&u = g_R, && x = x_R, && t \in (0, T),\\
&u(\cdot, 0) = u_0, \quad u_t(\cdot, 0) = v_0, && x \in [x_L, x_R]\,. &&
\end{aligned} \tag{2}
$$

We first discretize the interval $[x_L, x_R]$ using the N equidistant grid points,

$$
x_j = x_L + (j-1)h, \quad j = 1, \ldots, N, \quad h = \frac{x_R - x_L}{N-1}\,. \tag{3}
$$

We introduce the discrete solution vector $\mathbf{u}(t) = [u_1(t), \ldots, u_N(t)]^T$, where $u_j(t) \approx u(x_j, t)$. To discretize the spatial derivative we use a narrow-stencil summation-by-parts operator $D_2^{(c^2)}$, which approximates $\frac{\partial}{\partial x} c^2 \frac{\partial}{\partial x}$. $D_2^{(c^2)}$ is said to have the SBP property if [31]

$$
D_2^{(c^2)} = H^{-1}(-M_{c^2} + \bar{B}_{c^2} S) \tag{4}
$$

where $H = H^T > 0$, $M_{c^2} = M_{c^2}^T \geq 0$, S approximates the first derivative at the boundaries, and $\bar{B}_{c^2} = \operatorname{diag}\left(-c^2(x_1), 0, \ldots, 0, c^2(x_N)\right)$.

The SBP-SAT discretization of (2) can be written as

$$
\begin{aligned}
\frac{\mathrm{d}^2\mathbf{u}}{\mathrm{d}t^2} + \Lambda_\sigma \frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} - D_2^{(c^2)}\mathbf{u} = \mathbf{f} &+ H^{-1}\left[(\bar{B}_{c^2} S)^T - \frac{\Gamma c^2(x_1)}{\alpha h}\right]\mathbf{e}_1(\mathbf{e}_1^T\mathbf{u} - g_L)\\
&+ H^{-1}\left[(\bar{B}_{c^2} S)^T - \frac{\Gamma c^2(x_N)}{\alpha h}\right]\mathbf{e}_N(\mathbf{e}_N^T\mathbf{u} - g_R)\,,
\end{aligned} \tag{5}
$$

where $\mathbf{f}(t) = [f(x_1, t), \ldots, f(x_N, t)]^T$, $\Lambda_\sigma = \operatorname{diag}(\sigma(x_1), \ldots, \sigma(x_N))$, and $\{\mathbf{e}_i\}_{i=1}^N$ is the standard basis in $\mathbb{R}^N$. The SAT penalty terms that impose the Dirichlet boundary conditions are written in the right-hand side of (5). Here α depends on the order of the SBP operator (but not on h) and Γ is a penalty parameter. Mattsson et. al [30] showed that the scheme (5) is stable if $\Gamma \geq 1$. Their numerical experiments further

indicate that $\Gamma = 1.2$ is a suitable choice when considering both accuracy and stiffness; hence we use $\Gamma = 1.2$ in this paper.

The scheme (5) can be written as

$$\frac{\mathrm{d}^2\mathbf{u}}{\mathrm{d}t^2}(t) + \mathbf{D}\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t}(t) + \mathbf{A}\mathbf{u}(t) = \mathbf{R}(t) \tag{6}$$

with

$$\mathbf{D} = \Lambda_\sigma,$$
$$\mathbf{A} = -D_2^{(c^2)} - H^{-1}\left[(\bar{B}_{c^2}S)^T - \frac{\Gamma c^2(x_1)}{\alpha h}\right]\mathbf{e}_1\mathbf{e}_1^T - H^{-1}\left[(\bar{B}_{c^2}S)^T - \frac{\Gamma c^2(x_N)}{\alpha h}\right]\mathbf{e}_N\mathbf{e}_N^T,$$
$$\mathbf{R} = \mathbf{f} - H^{-1}\left[(\bar{B}_{c^2}S)^T - \frac{\Gamma c^2(x_1)}{\alpha h}\right]\mathbf{e}_1 g_L - H^{-1}\left[(\bar{B}_{c^2}S)^T - \frac{\Gamma c^2(x_N)}{\alpha h}\right]\mathbf{e}_N g_R\,.$$

It is straightforward to extend the scheme (5) to rectangular domains in n dimensions using tensor products. The extension to general curvilinear coordinates is more involved [40], but the resulting semi-discrete problem can still be written in the form of (6).

One way to introduce local grid refinement is to divide the computational domain into blocks and use different grid sizes in each block. The SBP-SAT method for the wave equation was first extended to multi-block domains in [29]. At the block interfaces, SAT penalty terms similar to the ones that impose the Dirichlet boundary conditions in (5) are used to impose interface conditions (continuity of the solution and its derivative across the interface) in a stable manner. In one dimension this is relatively straightforward, but in higher dimensions there is the additional complication that the grids do not conform at the interfaces between coarse and fine blocks. Treating non-conforming grid interfaces requires special operators that transfer the solution between coarse and fine grids. The *interpolation operators* constructed in [28] and the *projection operators* in [27] are both provably stable for first order hyperbolic systems. It was recently shown [41] that the projection and interpolation operators both lead to stable discretizations also of the second order wave equation, in combination with SBP operators with up to 4th order accurate interior stencils. In this paper we use the interpolation operators and do not consider higher than 4th order spatial accuracy; thus the spatial discretizations are stable. When employing the interpolation operators the semi-discrete system can still be written in the form (6). Thus, for our purposes it suffices to focus on the time-integration of (6).

2.2. Continuous finite element methods. Various FEM are available for the spatial discretization of (1). For instance, the standard H^1-conforming FEM with mass-lumping starts from the weak formulation: Find $u : [0,T] \to H_0^1(\Omega)$ such that

$$\begin{aligned} (u_{tt}, v) + (\sigma u_t, v) + (c\nabla u, c\nabla v) &= (f, v) && \forall\, v \in H_0^1(\Omega),\ t \in (0,T), \\ u|_{t=0} &= u_0 && \text{in } \Omega, \\ u_t|_{t=0} &= v_0 && \text{in } \Omega, \end{aligned} \tag{7}$$

where $(\cdot,\cdot)$ denotes the standard inner product on $L^2(\Omega)$

$$(w, v) = \int_\Omega w(\mathbf{x})v(\mathbf{x})\,d\mathbf{x}.$$

We assumed here that $g = 0$, i.e. we deal with homogeneous Dirichlet boundary conditions. The extension to nonzero g is, however, not difficult. For $\sigma = 0$ (7)

reduces to the variational formulation of the classical wave equation. Next, we consider a family of shape-regular meshes $\{\mathcal{T}_h\}_h$ that each partition Ω into disjoint elements K, i.e. $\overline{\Omega} = \cup_{K\in\mathcal{T}_h} \overline{K}$; for simplicity, we assume that Ω is polygonal. The diameter of element K, a triangle or a quadrilateral in two space dimensions, and a tetrahedron or hexahedron in three dimensions, is denoted by h_K; hence, the mesh size, h, is given by $h = \max_{K\in\mathcal{T}_h} h_K$. We also let $V_h \subset H_0^1(\Omega)$ denote the finite dimensional subspace

$$V_h = \{v \in H_0^1(\Omega) \,:\, v|_K \in \mathcal{S}^\ell(K),\ \forall K \in \mathcal{T}_h\}, \qquad \ell \geq 1,$$

where $\mathcal{S}^\ell(K)$ corresponds to the space $\mathcal{P}^\ell(K)$ of polynomials of total degree at most ℓ, if K is a triangle or tetrahedron, or the space $\mathcal{Q}^\ell(K)$ of polynomials of maximal degree ℓ in each variable, if K is a quadrilateral or hexahedron.

The semi-discrete Galerkin approximation, $u^h(t) \in V_h$, is then defined for $0 \leq t < T$ by the restriction of (7) to V_h: Find $u^h : [0,T] \to V^h$ such that

$$\begin{aligned} (u^h_{tt}, v) + (\sigma u^h_t, v) + (c\nabla u^h, c\nabla v) &= (f, v) \qquad \forall\, v \in V^h\,, \quad t \in (0,T)\,, \\ u^h(\cdot,0) = \Pi_h u_0\,,\ u^h_t(\cdot,0) &= \Pi_h v_0\,. \end{aligned} \tag{8}$$

Here, Π_h denotes the L^2-projection onto V^h.

Let $\mathbf{u}(t) \in \mathbb{R}^N$ denote the coefficients of $u^h(t)$ with respect to the standard Lagrangian basis $\{\phi_i\}_{i=1}^N$ of V_h. Then (8) is equivalent to the second-order system of ODEs

$$\begin{aligned} &\mathbf{M}\,\frac{\mathrm{d}^2\mathbf{u}}{\mathrm{d}t^2}(t) + \mathbf{M}_\sigma\,\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t}(t) + \mathbf{K}\,\mathbf{u}(t) = \widetilde{\mathbf{R}}(t)\,, \qquad t \in (0,T), \\ &\mathbf{M}\mathbf{u}(0) = u_0^h, \quad \mathbf{M}\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t}(0) = v_0^h, \end{aligned} \tag{9}$$

where u_0^h, v_0^h are suitable approximations of the initial conditions. Moreover, the mass matrix, $\mathbf{M}$, and the stiffness matrix, $\mathbf{K}$, are given by

$$\mathbf{M}_{ij} = (\phi_j, \phi_i), \qquad \mathbf{K}_{ij} = (c\nabla\phi_j, c\nabla\phi_i);$$

the matrix $\mathbf{M}_\sigma$ also corresponds to a mass matrix with weight σ. The matrix $\mathbf{M}$ is sparse, symmetric and positive definite, whereas the matrices $\mathbf{K}$ and $\mathbf{M}_\sigma$ are sparse, symmetric but, in general, only positive semi-definite.

Even though explicit numerical time integration may be applied directly to (9), every time-step then requires the solution of a linear system involving $\mathbf{M}$ or the multiplication with the inverse of $\mathbf{M}$, which need no longer be sparse. To avoid that computational work, various mass-lumping techniques have been developed [7, 32], which replace $\mathbf{M}$ by a diagonal approximation without ruining the accuracy [3]. Alternatively, the spectral element method [4, 26] and the symmetric interior penalty DG method [18] both waive the need for mass-lumping altogether: The former inherently leads to a diagonal mass matrix, whereas the latter leads to a block-diagonal mass matrix with block size equal to the number of degrees of freedom per element. Thus, both mentioned alternative discretizations also lead to (9) with an essentially diagonal mass matrix $\mathbf{M}$.

Since $\mathbf{M}$ in (9) is diagonal, the matrix $\mathbf{M}^{-1}$ is immediately available. Then, multiplication of (9) by $\mathbf{M}^{-1}$ yields

$$\frac{\mathrm{d}^2\mathbf{u}}{\mathrm{d}t^2}(t) + \mathbf{D}\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t}(t) + \mathbf{A}\mathbf{u}(t) = \mathbf{R}(t),$$

with

$$\mathbf{D} = \mathbf{M}^{-1}\mathbf{M}_\sigma, \quad \mathbf{A} = \mathbf{M}^{-1}\mathbf{K}, \quad \mathbf{R}(t) = \mathbf{M}^{-1}\widetilde{\mathbf{R}}(t),$$

which is of the same form as (6).

In order to apply an RK type LTS scheme we first need to rewrite (6) as a first-order system

$$\begin{aligned} \frac{\mathrm{d}\mathbf{y}}{\mathrm{d}t}(t) &= \mathbf{B}\mathbf{y}(t) + \mathbf{F}(t)\,, \qquad t \in (0,T)\,, \\ \mathbf{y}(0) &= \mathbf{y}_0\,, \end{aligned} \tag{10}$$

where we have introduced

$$\mathbf{y}(t) = \begin{pmatrix} \mathbf{u}(t) \\ \dfrac{\mathrm{d}\mathbf{u}}{\mathrm{d}t}(t) \end{pmatrix}, \qquad \mathbf{B} = \begin{pmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{A} & -\mathbf{D} \end{pmatrix}, \qquad \mathbf{F}(t) = \begin{pmatrix} \mathbf{0} \\ \mathbf{R}(t) \end{pmatrix}.$$

3. LTS Methods of Runge-Kutta Type. Before deriving the multilevel algorithm in Section 4 we recall the LTS-RK methods proposed in [17]. We introduce a slightly modified notation, which may seem overly complicated here but will simplify the extension of the LTS scheme to multiple refinement levels later. In [17], continuous FE and nodal DG discretizations were considered and it was shown that the LTS-RK methods retain the optimal CFL-condition when combined with these spatial discretizations. In Section 3.2 we investigate the stability properties of the LTS-RK schemes when combined with SBP-SAT FD discretizations. The observations from this section will be used later when we combine SBP-SAT FD with the MLTS-RK schemes in Section 6.

3.1. Explicit LTS-RK schemes. The LTS-RK methods are based on the classical RK schemes [20]. Let $\mathbf{y}_n$ denote the numerical approximation of the exact solution $\mathbf{y}(t_n)$ at time $t_n = n\Delta t$. A general explicit RK method with s stages (hereafter denoted an RKs method) applied to (10) yields

$$\begin{aligned} \mathbf{k}_1 &= \mathbf{B}\mathbf{y}_n + \mathbf{F}(t_n)\,, \\ \mathbf{k}_2 &= \mathbf{B}(\mathbf{y}_n + \Delta t\, a_{21}\mathbf{k}_1) + \mathbf{F}(t_n + c_2\Delta t)\,, \\ &\vdots \\ \mathbf{k}_s &= \mathbf{B}\left(\mathbf{y}_n + \Delta t \sum_{i=1}^{s-1} a_{si}\mathbf{k}_i\right) + \mathbf{F}(t_n + c_s\Delta t)\,, \\ \mathbf{y}_{n+1} &= \mathbf{y}_n + \Delta t \sum_{i=1}^{s} b_i\mathbf{k}_i\,. \end{aligned} \tag{11}$$

The constants a_{ij}, b_i, and c_i, with $c_1 = 0$, uniquely identify the RK method. Let further $\{\widetilde{c}_1, \ldots, \widetilde{c}_{s_0}\} \subset \{c_1, \ldots, c_s\}$ be the maximal subset of all coefficients c_i such that no two are identical, i.e. $\widetilde{c}_i \neq \widetilde{c}_j$ if $i \neq j$.

To derive LTS-RK methods for the efficient numerical solution of (10) Grote et al. [17] introduce a modified problem, which is subsequently solved from t_n to t_{n+1} by applying the RK scheme (11) with a smaller time-step. To derive the modified problem they introduce a projection matrix $\mathbf{P}_1$, which is diagonal with diagonal entries equal

to one exactly where the fine unknowns are located and zeros otherwise. The exact solution and right-hand side of (10) are then partitioned into coarse and fine parts,

$$\mathbf{y}(t) = (\mathbf{I} - \mathbf{P}_1)\mathbf{y}(t) + \mathbf{P}_1\mathbf{y}(t)\,, \qquad \mathbf{F}(t) = (\mathbf{I} - \mathbf{P}_1)\mathbf{F}(t) + \mathbf{P}_1\mathbf{F}(t)\,,$$

where $(\mathbf{I} - \mathbf{P}_1)\mathbf{y}(t)$ is zero in the fine region and $\mathbf{P}_1\mathbf{y}(t)$ is zero in the coarse region. Based on this partitioning one can derive the modified equation

$$\begin{aligned} \frac{\mathrm{d}\mathbf{y}^{[1]}}{\mathrm{d}t^{[1]}}(t^{[1]}) &= \mathbf{B}(\mathbf{I} - \mathbf{P}_1)\left[\sum_{j=0}^{s-1} \alpha_j \left(t^{[1]}\right)^j \left(\mathbf{B}^j\mathbf{y}_n + \sum_{\ell=1}^{j} \mathbf{B}^{j-\ell}\mathbf{q}_0^{(\ell-1)}(t_n)\right)\right] \\ &\quad + (\mathbf{I} - \mathbf{P}_1)\mathbf{q}_0(t_n + t^{[1]}) + \mathbf{B}\mathbf{P}_1\,\mathbf{y}^{[1]}(t^{[1]}) + \mathbf{P}_1\mathbf{F}(t_n + t^{[1]})\,, \\ \mathbf{y}^{[1]}(0) &= \mathbf{y}_n\,, \end{aligned} \tag{12}$$

where

$$\alpha_j = \frac{j+1}{j!}\sum_{i=1}^{s} b_i c_i^j.$$

Here $\mathbf{q}_0(t)$ denotes the interpolation polynomial of degree $s_0 - 1$ going through the points $(t_n + c_i\Delta t, \mathbf{F}(t_n + c_i\Delta t))$, $i = 1, \ldots, s$. We can write (12) in the form of (10)

$$\begin{aligned} \frac{\mathrm{d}\mathbf{y}^{[1]}}{\mathrm{d}t^{[1]}}(t^{[1]}) &= \mathbf{B}_1\mathbf{y}^{[1]}(t^{[1]}) + \mathbf{F}_1(t^{[1]})\,, \qquad t^{[1]} \in (0, \Delta t)\,, \\ \mathbf{y}^{[1]}(0) &= \mathbf{y}_n\,, \end{aligned} \tag{13}$$

with matrix and right-hand side defined as

$$\begin{aligned} \mathbf{B}_1 &= \mathbf{B}\mathbf{P}_1\,, \\ \mathbf{F}_1(t^{[1]}) &= \sum_{j=0}^{s-1} \left(t^{[1]}\right)^j \mathbf{w}_j^{[1]} + (\mathbf{I} - \mathbf{P}_1)\mathbf{q}_0(t_n + t^{[1]}) + \mathbf{P}_1\mathbf{F}(t_n + t^{[1]})\,, \end{aligned}$$

where

$$\mathbf{w}_j^{[1]} = \alpha_j\mathbf{B}(\mathbf{I} - \mathbf{P}_1)\left(\mathbf{B}^j\mathbf{y}_n + \sum_{\ell=1}^{j} \mathbf{B}^{j-\ell}\mathbf{q}_0^{(\ell-1)}(t_n)\right).$$

To advance the numerical solution from time t_n to $t_{n+1} = t_n + \Delta t$, the LTS-RK algorithm solves the modified equation (13) with the underlying RK method but with a smaller time-step $\Delta t^{[1]} = \Delta t/p_1$, where $p_1 > 1$ is an integer. If the smallest mesh size in the fine region is $\widetilde{p}_1$ times smaller than the smallest mesh size in the coarse region, one picks $p_1 = \lceil \widetilde{p}_1 \rceil$, to ensure that the CFL condition is not violated. The most challenging case for the LTS-RK method is when $\widetilde{p}_1$ is an integer so that $p_1 = \widetilde{p}_1$. In this case, both the global and local time-steps are exactly on the CFL stability limit. In the numerical experiments in Section 3.2 we shall only consider this case; hence we do not distinguish between p_1 and $\widetilde{p}_1$. For details on the derivation of (12) and on the algorithm we refer to [17].

REMARK 1. *Note that the CFL condition for the classical RK2 scheme is* $\Delta t \leq Ch^{4/3}$ *[11], which is stronger than for most higher order RKs methods. For the LTS-RK2 method to conform to this stability restriction we have to choose* $\Delta t^{[1]} = \Delta t/\lceil \widehat{p} \rceil$ *with* $\widehat{p} = p_1^{4/3}$.

3.2. LTS-RK combined with SBP-SAT FD. We now study the combination of SBP-SAT FD in space and LTS-RK methods in time. We consider (1) with $c = 1$, $\sigma = 0.1$, and $f = 0$ on the interval $\Omega = [0, 6]$. We divide Ω into three equal parts and use the grid spacing h^{coarse} in the left and right intervals, $[0, 2]$ and $[4, 6]$, referred to as the coarse region. In the fine region $[2, 4]$ we use the grid spacing $h^{\text{fine}} = h^{\text{coarse}}/p_1$. We combine the LTS approach based on the RK2 method with the second-order accurate SBP-SAT discretization and the RK3 and RK4 type LTS methods with the fourth-order accurate SBP-SAT discretization. Butcher tableaus of the methods used here can be found in Table 4.

To determine the stability range of the LTS schemes we rewrite them as one-step schemes,

$$\mathbf{y}_{n+1} = \mathbf{C}_{LTS-RKs}\mathbf{y}_n \tag{14}$$

and monitor the spectral radius $\rho(\mathbf{C}_{LTS-RKs})$ for different time-steps Δt. The method is stable if $\rho(\mathbf{C}_{LTS-RKs}) \leq 1$. We denote the maximal permissible time-step by Δt_{p_1}. We say that the CFL condition of the LTS method is optimal if $\Delta t_{p_1} = \Delta t_{RKs}$, where Δt_{RKs} denotes the largest permissible time-step of the underlying RK-method on an equidistant grid of size h^{coarse}. In [17] both continuous FE and nodal DG discretizations led to optimal CFL conditions for the LTS-RK methods.

Figure 1 (a) shows $\rho(\mathbf{C}_{LTS-RK4})$ for different time-steps, when $p_1 = 2$. We observe that the scheme is unstable for $\Delta t/\Delta t_{RK4} > 0.8$. That is, we are restricted to approximately 80% of the optimal time-step, which is unsatisfactory. To overcome this restriction we consider the technique of *overlapping*, which was successfully used in [13] for LTS methods based on the "leap-frog" scheme for second order wave equations. By overlapping we mean that we slightly enlarge the region where the small time-step is used by setting the corresponding entries in $\mathbf{P}_1$ to one. Thus, the local time-step is applied also to a few coarse grid points adjacent to the fine region, see Figure 2. When using one grid point of overlap, we observe in Figure 1 (b) that we obtain the optimal CFL condition. Figure 3 shows $\rho(\mathbf{C}_{LTS-RK2})$ and $\rho(\mathbf{C}_{LTS-RK3})$, using one grid point of overlap. Again, the CFL conditions are optimal.

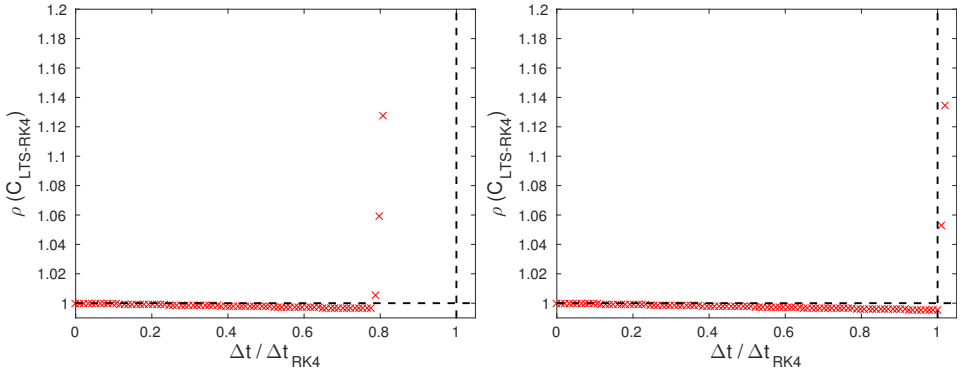


FIG. 1. *The spectral radius of* $\mathbf{C}_{LTS-RK4}$ *vs.* $\Delta t/\Delta t_{RK4}$ *using (a) no overlap and (b) one grid of point overlap.*

To investigate the effect of a larger p_1, Table 1 shows $\Delta t_{p_1}/\Delta t_{RKs}$ for $s = 2, 3, 4$ and various p_1, again using one grid point of overlap. The LTS-RK2 and LTS-RK3

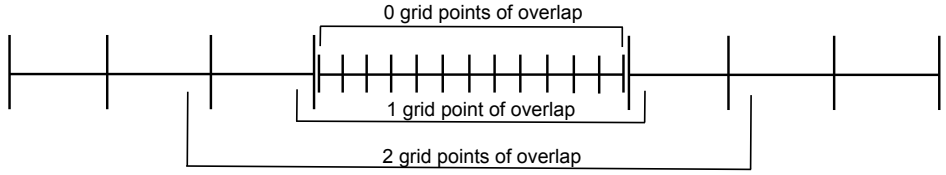
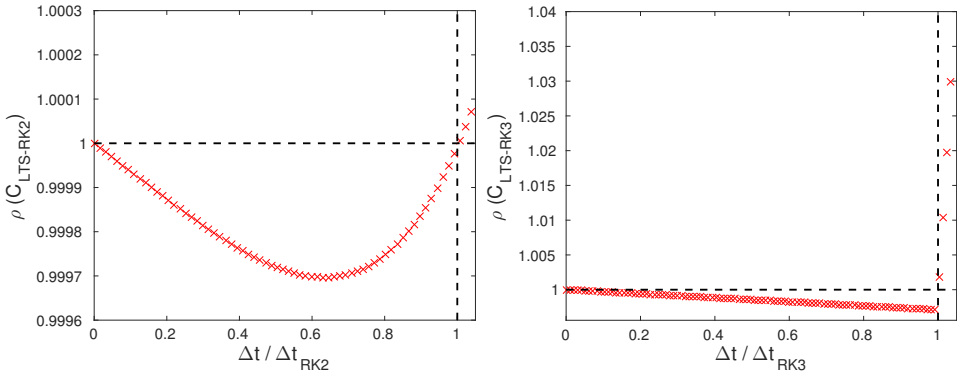


FIG. 2. *By overlapping, we mean that the local time-step is used also in a few grid points in the coarse region, adjacent to the fine region. Notice that the discrete solution is duplicated at the interface between coarse and fine regions, i.e., there are two grid points at the same physical location. Here they are visualized with a small separation for illustrative purposes.*



(a) (b)

FIG. 3. *The spectral radius of (a) $\mathbf{C}_{LTS-RK2}$ vs. $\Delta t/\Delta t_{RK2}$ and (b) $\mathbf{C}_{LTS-RK3}$ vs. $\Delta t/\Delta t_{RK3}$, using one grid point of overlap.*

methods retain the optimal CFL condition regardless of p_1. The LTS-RK4 method, however, exhibits a decrease in CFL number with increasing p_1. The solution to this problem is to use more overlap. Table 2 shows the ratio $\Delta t_{p_1}/\Delta t_{RK4}$ corresponding to different amounts of overlap. We observe that 2 grid points of overlap yields the optimal CFL condition up to $p_1 = 5$ and 3 grid points of overlap yields the optimal CFL condition up to $p_1 = 30$. Based on this study we conclude that, unless extreme refinement ratios are considered, suitable overlaps are 1 grid point for LTS-RK2 and LTS-RK3 and 3 grid points for LTS-RK4. We will use these overlaps in subsequent numerical experiments with the multilevel schemes.

To illustrate the effect of damping on the stability we present the ratio $\Delta t_2/\Delta t_{RKs}$ $(s = 2, 3, 4)$ for varying σ in Table 3. All LTS schemes yield the optimal CFL condition independently of $\sigma > 0$. When $\sigma = 0$ the equation (1) is undamped, which makes energy-conserving time-marching schemes such as the standard leap-frog scheme more attractive than RK methods. High-order energy-conserving MLTS methods based on the leap-frog scheme were derived in [14].

Numerical experiments in [17] confirmed the expected convergence rates in space and time for the LTS-RK schemes when combined with both continuous and discontinuous FEM. Our experiments show the expected rates also for the combination of LTS-RK and SBP-SAT FD, but we omit the results here. Convergence results for the MLTS-RK schemes are presented in Section 6.

	$p_1 = 2$	$p_1 = 3$	$p_1 = 5$	$p_1 = 11$	$p_1 = 15$	$p_1 = 20$	$p_1 = 30$
LTS-RK2	1.0	1.0	1.0	1.0	1.0	1.0	1.0
LTS-RK3	1.0	1.0	1.0	1.0	1.0	1.0	1.0
LTS-RK4	1.0	1.0	0.73	0.40	0.34	0.54	0.38

TABLE 1

Stability of the LTS-RKs schemes for $s = 2, 3, 4$: the ratio $\Delta t_{p_1}/\Delta t_{RKs}$ is shown for varying p_1, using an overlap of 1 grid point.

Overlap	$p_1 = 2$	$p_1 = 3$	$p_1 = 5$	$p_1 = 11$	$p_1 = 15$	$p_1 = 20$	$p_1 = 30$
1 grid point	1.0	1.0	0.73	0.40	0.34	0.54	0.38
2 grid points	1.0	1.0	1.0	0.89	0.77	0.85	0.77
3 grid points	1.0	1.0	1.0	1.0	1.0	1.0	1.0

TABLE 2

Stability of the LTS-RK4 scheme, using 1, 2, or 3 grid points of overlap: the ratio $\Delta t_{p_1}/\Delta t_{RK4}$ is shown for varying p_1.

4. Multi-level LTS-RK. The LTS-RK methods in [17] are limited to one local time-step size. While this is sufficient when all mesh elements are naturally categorized as either coarse or fine, there are situations where it is suboptimal. If, for example, a locally refined region itself contains a locally refined region, any combination of global time-step Δt and local time-step $\Delta t^{[1]}$ will correspond to an unnecessarily small time-step in some part of the domain. This can severely affect the efficiency of the LTS-RK scheme. Here, we shall derive MLTS schemes that allow for an appropriate time-step on each level of refinement. We present the derivation in Section 4.1 and discuss in Section 4.2 some details that are useful when implementing the MLTS-RK method.

4.1. Derivation of the MLTS-RK scheme. From now on we assume that we can distinguish $L_{\max} + 1$ levels of refinement in our mesh. That is, we can divide the mesh into $L_{\max} + 1$ levels or tiers of like-sized elements. We number the levels from 0 to $L_{\max}$, with 0 corresponding to the coarsest level. To each refined level we can associate a refinement ratio p_L, $L = 1, \ldots, L_{\max}$, where p_L is determined by the ratio between the smallest mesh sizes of levels $L - 1$ and L. Thus, the mesh size on the finest level is approximately $p_1 \cdot p_2 \cdot \ldots \cdot p_{L_{\max}}$ times smaller than the mesh size on the coarsest level. We further introduce projection matrices $\mathbf{P}_L$, $L = 0, \ldots, L_{\max}$, where $\mathbf{P}_L$ selects all unknowns that belong to the levels $L, L+1, \ldots, L_{\max}$. Thus, the projection matrices have the property

$$\mathbf{P}_L \mathbf{P}_{L+i} = \mathbf{P}_{L+i} \qquad \text{if } i \geq 0. \tag{15}$$

By definition $\mathbf{P}_0$ selects all elements, i.e. $\mathbf{P}_0 = \mathbf{I}$.

REMARK 2. *Similar to p_1 in the LTS case, the p_i all need to be integers, whereas the mesh size ratios $\widetilde{p}_i$ are not, in general. To avoid violating the CFL condition, one picks $p_i = \lceil \widetilde{p}_i \rceil$. For simplicity we shall avoid distinguishing between p_i and $\widetilde{p}_i$ in what follows.*

To derive MLTS-RK methods, we use LTS-RK methods as a starting point. LTS-RK methods rely on constructing the modified ODE (13), i.e.,

$$\begin{aligned} \frac{\mathrm{d}\mathbf{y}^{[1]}}{\mathrm{d}t^{[1]}}(t^{[1]}) &= \mathbf{B}_1 \mathbf{y}^{[1]}(t^{[1]}) + \mathbf{F}_1(t^{[1]}), \qquad t^{[1]} \in (0, \Delta t), \\ \mathbf{y}^{[1]}(0) &= \mathbf{y}_n, \end{aligned} \tag{16}$$

σ	LTS-RK2	LTS-RK3	LTS-RK4
0.001	1.0	1.0	1.0
0.01	1.0	1.0	1.0
0.1	1.0	1.0	1.0
1	1.0	1.0	1.0
10	1.0	1.0	1.0

TABLE 3

Stability of the LTS-RKs schemes for $s = 2, 3, 4$ and $p_1 = 2$, using 1 grid point overlap: the ratio $\Delta t_2/\Delta t_{RKs}$ is shown for varying σ.

and solving this modified problem using p_1 time steps of size $\Delta t^{[1]}$ with the underlying RKs method, as described in Section 3. However, in the case of multiple levels of refinement the time-step $\Delta t^{[1]}$ is of course too large on the finer levels $2, \ldots, L_{\max}$. Recall that we split the unknowns with respect to $\mathbf{P}_1$, which selects the degrees of freedom not only on level 1, but also on all finer levels. To derive the MLTS-RKs method, we shall therefore apply the LTS-RK approach to the modified problem (16). Because our aim is to use an even smaller time-step on the elements belonging to level 2, we split the unknowns with respect to $\mathbf{P}_2$ this time. Since (16) is of the same form as (10), we can use (13) to infer that the modified problem corresponding to (16) is

$$\begin{aligned} \frac{\mathrm{d}\mathbf{y}^{[2]}}{\mathrm{d}t^{[2]}}(t^{[2]}) &= \mathbf{B}_2\mathbf{y}^{[2]}(t^{[2]}) + \mathbf{F}_2(t^{[2]})\,, \qquad t^{[2]} \in (0, \Delta t^{[1]})\,, \\ \mathbf{y}^{[2]}(0) &= \mathbf{y}^{[1]}_{n_1}\,, \end{aligned} \tag{17}$$

where

$$\begin{aligned} \mathbf{B}_2 &= \mathbf{B}_1\mathbf{P}_2\,, \\ \mathbf{F}_2(t^{[2]}) &= \sum_{j=0}^{s-1} \left(t^{[2]}\right)^j \mathbf{w}_j^{[2]} + (\mathbf{I} - \mathbf{P}_2)\mathbf{q}_1(t^{[1]}_{n_1} + t^{[2]}) + \mathbf{P}_2\mathbf{F}_1(t^{[1]}_{n_1} + t^{[2]})\,, \end{aligned}$$

with

$$\mathbf{w}_j^{[2]} = \alpha_j \mathbf{B}_1(\mathbf{I} - \mathbf{P}_2)\left(\mathbf{B}_1^j\mathbf{y}^{[1]}_{n_1} + \sum_{\ell=1}^{j} \mathbf{B}_1^{j-\ell}\mathbf{q}_1^{(\ell-1)}(t^{[1]}_{n_1})\right).$$

Here $\mathbf{q}_1(t^{[1]})$ denotes the interpolation polynomial of degree $s_0 - 1$ going through the points $\left(t^{[1]}_{n_1} + c_i\Delta t^{[1]}, \mathbf{F}_1\left(t^{[1]}_{n_1} + c_i\Delta t^{[1]}\right)\right)$, $i = 1, \ldots, s$, where $t^{[1]}_{n_1} = n_1\Delta t^{[1]}$ is the current time on level 1 and $\mathbf{y}^{[1]}_{n_1}$ the corresponding numerical solution. Notice that the modified problem for $\mathbf{y}^{[2]}$ is posed on the time interval $(0, \Delta t^{[1]})$. In the case $L_{\max} = 2$, level 2 is the finest level. To advance the solution on levels 1 and 2 one time-step of size $\Delta t^{[1]}$, we solve (17) using p_2 time steps of size $\Delta t^{[2]} = \Delta t^{[1]}/p_2$. We repeat that p_1 times to advance the solution of the original ODE (10) from t_n to t_{n+1}.

For the case $L_{\max} > 2$, we note that the problem (17) is again of the same form as (10). We may thus split the unknowns with respect to $\mathbf{P}_3$ and derive a new modified problem, and so on. Repeating this process L times in total gives us the modified equation corresponding to level L,

$$
\begin{aligned}
\frac{\mathrm{d}\mathbf{y}^{[L]}}{\mathrm{d}t^{[L]}}(t^{[L]}) &= \mathbf{B}_L \mathbf{y}^{[L]}(t^{[L]}) + \mathbf{F}_L(t^{[L]}), \qquad t^{[L]} \in (0, \Delta t^{[L-1]}), \\
\mathbf{y}^{[L]}(0) &= \mathbf{y}^{[L-1]}_{n_{L-1}},
\end{aligned} \tag{18}
$$

where

$$
\begin{aligned}
\mathbf{B}_L &= \mathbf{B}_{L-1}\mathbf{P}_L, \\
\mathbf{F}_L(t^{[L]}) &= \sum_{j=0}^{s-1} \left(t^{[L]}\right)^j \mathbf{w}_j^{[L]} + (\mathbf{I} - \mathbf{P}_L)\mathbf{q}_{L-1}(t^{[L-1]}_{n_{L-1}} + t^{[L]}) \\
&\quad + \mathbf{P}_L \mathbf{F}_{L-1}(t^{[L-1]}_{n_{L-1}} + t^{[L]}),
\end{aligned} \tag{19}
$$

with

$$
\mathbf{w}_j^{[L]} = \alpha_j \mathbf{B}_{L-1}(\mathbf{I} - \mathbf{P}_L) \left(\mathbf{B}_{L-1}^j \mathbf{y}^{[L-1]}_{n_{L-1}} + \sum_{\ell=1}^{j} (\mathbf{B}_{L-1})^{j-\ell} \, \mathbf{q}_{L-1}^{(\ell-1)}(t^{[L-1]}_{n_{L-1}}) \right),
$$

and $\mathbf{q}_{L-1}$ is the interpolation polynomial of $\mathbf{F}_{L-1}$ in the quadrature points

$$
\left(t^{[L-1]}_{n_{L-1}} + c_i \Delta t^{[L-1]}, \mathbf{F}_{L-1}\left(t^{[L-1]}_{n_{L-1}} + c_i \Delta t^{[L-1]} \right) \right), \; i = 1, \ldots, s.
$$

Here, $t^{[L-1]}_{n_{L-1}}$ denotes the current time on level $L-1$, with the convention that $t^{[0]}_{n_0} = t_n$. The corresponding numerical solution $\mathbf{y}^{[L-1]}_{n_{L-1}}$ is used as initial condition for the new modified ODE (18).

The multi-level approach reduces to solving (18) at the final level $L = L_{\max}$ with the underlying RKs method using p_L fine time-steps of size $\Delta t^{[L]}$. This gives us the update of the numerical solution at level $L-1$. If we repeat that process p_{L-1} times in total, we get an update of the solution on level $L-2$, etc. Due to the inherent recursion over the levels, the MLTS-RK algorithms can be formulated as recursive functions – see Algorithm 1. The function call $\mathbf{y}^{[L]}_{n_L+1} = \text{MLTS-RK}(\Delta t^{[L]}, L+1, \mathbf{y}^{[L]}_{n_L}, \mathbf{B}_L, \mathbf{F}_L)$ advances the numerical solution on levels $L, L+1, \ldots, L_{\max}$ one step of size $\Delta t^{[L]}$. Thus, the call $\mathbf{y}_{n+1} = \text{MLTS-RK}(\Delta t, 1, \mathbf{y}_n, \mathbf{B}, \mathbf{F})$ advances all unknowns one global time-step of size Δt, via recursive calls to the MLTS-RK function. The recursion terminates when the finest level is reached, i.e. when $L = L_{\max}$. In this case, the MLTS-RK function calls the RK function in Algorithm 2, to apply the underlying RK method to the modified problem for $\mathbf{y}^{[L_{\max}]}$.

A schematic description of the MLTS-RK algorithm in the special case $L_{\max} = 2$, $p_1 = 3$, $p_2 = 2$ is given in Figure 4. To advance all unknowns one global time-step of size Δt, steps 1-10 must be completed. Figure 4 illustrates the state that the unknowns are in when only steps 1-6 have been completed: the unknowns in the finest level (level 2) have been advanced 3 time-steps of size $\Delta t^{[2]} = \Delta t/6$, the unknowns in the intermediate level (level 1) have been advanced one step of size $\Delta t^{[1]} = \Delta t/3$, and the coarsest unknowns (level 0) have not yet been advanced at all. The colors black and grey indicate whether the solution at that time-step has been computed or not. Whenever the unknowns $\mathbf{y}^{[L]}_{n_L}$ on level $L < L_{\max}$ have been advanced, one uses (19) to update $\mathbf{B}_L$ and $\mathbf{F}_L$. Notice, however, that $\mathbf{B}_L$ does not change in between time-steps and thus one could make the algorithm more efficient by precomputing the $\mathbf{B}_L$ once and for all.

Algorithm 1 $\mathbf{y} = \text{MLTS-RK}(\Delta t, L, \mathbf{y}, \mathbf{B}, \mathbf{F})$

Compute new $\mathbf{B}$ and $\mathbf{F}$ with (19).
if $L < L_{\max}$ **then**
  for $n_L = 0, \dots, p_L - 1$ **do**
    $\mathbf{y} = \text{MLTS-RK}(\Delta t/p_L, L+1, \mathbf{y}, \mathbf{B}, \mathbf{F})$
  end for
else
  for $n_L = 0, \dots, p_L - 1$ **do**
    $\mathbf{y} = \text{RK}(\Delta t/p_L, \mathbf{y}, \mathbf{B}, \mathbf{F})$
  end for
end if
return $\mathbf{y}$

Algorithm 2 $\mathbf{y} = \text{RK}\ (\Delta t, \mathbf{y}, \mathbf{B}, \mathbf{F})$

for $r = 1, \dots, s$ **do**
  $\mathbf{k}_r = \mathbf{B}\left(\mathbf{y} + \Delta t \sum_{i=1}^{r-1} a_{ri}\mathbf{k}_i\right) + \mathbf{F}(t_{n_{L_{\max}}}^{[L_{\max}]} + c_r \Delta t)$
end for
Compute $\mathbf{y} = \mathbf{y} + \Delta t \sum_{i=1}^{s} b_i \mathbf{k}_i$.

4.2. Details on implementation. To implement the MLTS-RK method efficiently, it is important to utilize the sparsity of the matrices with which one needs to perform matrix-vector multiplications. We shall in this subsection expand the terms appearing in (18) to reveal the sparsity patterns of the matrices. We will make use of the following notation:

$$T_{\lambda,L} = \sum_{\ell=\lambda}^{L} t_{n_\ell}^{[\ell]}, \quad \beta_{ki} = \frac{k!}{(k-i+1)!}\,.$$

Due to property (15), we have

$$\mathbf{B}_L = \mathbf{B}_{L-1}\mathbf{P}_L = \mathbf{B}_{L-2}\mathbf{P}_{L-1}\mathbf{P}_L = \mathbf{B}\mathbf{P}_1 \cdots \mathbf{P}_L = \mathbf{B}\mathbf{P}_L\,. \tag{20}$$

By straightforward but tedious algebra, one can also show (see Appendix B) that

$$\begin{aligned} F_L(t^{[L]}) = &\sum_{j=0}^{s-1}\sum_{\ell=0}^{L-1}\left(t^{[L]} + T_{\ell+1,L-1}\right)^j \mathbf{w}_j^{[\ell+1]} + \\ &+ \mathbf{P}_L\mathbf{F}\left(t^{[L]} + T_{0,L-1}\right) + \sum_{\ell=0}^{L-1}(\mathbf{P}_\ell - \mathbf{P}_{\ell+1})\mathbf{r}_\ell\left(t^{[L]} + T_{0,L-1}\right), \end{aligned} \tag{21}$$

where the $\mathbf{r}_\ell$ are interpolation polynomials of $\mathbf{F}$ in the quadrature points

$$\left(T_{0,\ell} + c_i\Delta t^{[\ell]}, \mathbf{F}\left(T_{0,\ell} + c_i\Delta t^{[\ell]}\right)\right),\ i = 1, \dots, s\,,$$

and

$$\begin{aligned} \mathbf{w}_j^{[\ell+1]} = \alpha_j \mathbf{B}(\mathbf{P}_\ell - \mathbf{P}_{\ell+1})\Bigg[&(\mathbf{B}\mathbf{P}_\ell)^j \mathbf{y}_{n_\ell}^{[\ell]} + \sum_{\lambda=1}^{j}(\mathbf{B}\mathbf{P}_\ell)^{j-\lambda}\mathbf{r}_\ell^{(\lambda-1)}(T_{0,\ell}) \\ &+ \sum_{i=1}^{j}(\mathbf{B}\mathbf{P}_\ell)^{j-i}\sum_{k=i-1}^{s-1}\beta_{ki}\sum_{\lambda=1}^{\ell}(T_{\lambda,\ell})^{k-i+1}\mathbf{w}_k^{[\lambda]}\Bigg]\,. \end{aligned} \tag{22}$$

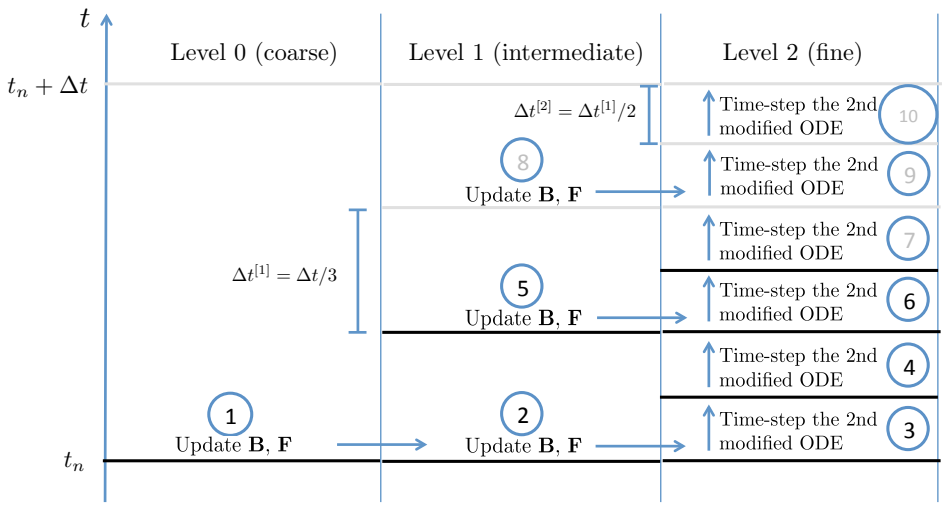


FIG. 4. *The multi-level algorithm in the case* $L_{\max} = 2$, $p_1 = 3$, $p_2 = 2$. *Arrows indicate the flow of information. Completing steps 1-10 corresponds to advancing all unknowns one global time-step of size* Δt.

Based on the modified equation (18) and the relations (20)-(22), we present a more detailed version of the MLTS-RK algorithm in Algorithm 3. As in Algorithm 1, it is natural to use recursion over the number of levels. The function call $\mathbf{y}^{[L]}_{n_L+1} = \text{MLTS-RK}(\Delta t^{[L]}, L+1, \mathbf{y}^{[L]}_{n_L})$ advances the numerical solution on levels $L, L+1, \ldots, L_{\max}$ one step of size $\Delta t^{[L]}$. The MLTS-RK function accomplishes this by calling itself recursively. The recursion terminates when the finest level is reached, i.e. when $L = L_{\max}$. In this case, the MLTS-RK function calls the RK function in Algorithm 4, to advance the finest level in time. Recall that time-stepping the unknowns on level $L_{\max}$ is performed by applying the underlying RK method to the modified problem for $\mathbf{y}^{[L_{\max}]}$, which is precisely what Algorithm 4 does.

At level L in the MLTS-RK algorithm, one first computes a polynomial $\mathbf{r}_L$ that approximates $\mathbf{F}$ in the nodes

$$\left(T_{0,L} + c_i \Delta t^{[L]}, \mathbf{F}\left(T_{0,L} + c_i \Delta t^{[L]}\right)\right), \; i = 1, \ldots, s\,.$$

Next, one computes the $\mathbf{w}_j^{[L+1]}$, $j = 0, \ldots, s-1$. In this step, the work corresponding to multiplications by $\mathbf{B}$ is performed. Notice that the factor $(\mathbf{P}_L - \mathbf{P}_{L+1})$ in (22) zeros all matrix elements not corresponding to the unknowns on or directly next to level L. Thus, multiplications by $\mathbf{B}(\mathbf{P}_L - \mathbf{P}_{L+1})$ are much cheaper than multiplications by $\mathbf{B}$, for instance. This shows that operations corresponding to the time-step of size $\Delta t^{[L]}$ only affect the unknowns on or directly adjacent to level L, which is precisely the property that an efficient MLTS algorithm should have. Roughly speaking, computing the $\mathbf{w}_j^{[L+1]}$ for $j = 0, \ldots, s-1$ corresponds to advancing the unknowns on level L one time-step of size $\Delta t^{[L]}$. The $\mathbf{w}$:s and the $\mathbf{r}$:s are then passed on to finer levels and used in subsequent calculations. For simplicity we have assumed that $L_{\max}$, the $\mathbf{P}_L$, the p_L, the $\mathbf{r}_L$, and the $\mathbf{w}_j^{[L]}$ are declared as global variables in the algorithms, but they could as well be passed as function parameters.

Algorithm 3 $\mathbf{y}_{new} = \text{MLTS-RK}(\Delta t, L, \mathbf{y}_{old})$

Set $\mathbf{y}_{new} = \mathbf{y}_{old}$
Compute interpolation polynomial $\mathbf{r}_L(t^{[L]})$ going through
$(T_{0,L} + c_i\Delta t, \mathbf{F}\left((T_{0,L} + c_i\Delta t)\right)$, $i = 1, \ldots, s$.
for $j = 0, \ldots, s-1$ **do**
Compute

$$\mathbf{w}_j^{[L+1} = \alpha_j \mathbf{B}(\mathbf{P}_L - \mathbf{P}_{L+1}) \left[(\mathbf{B}\mathbf{P}_L)^j \mathbf{y}_{old} + \sum_{\lambda=1}^{j} (\mathbf{B}\mathbf{P}_L)^{j-\lambda} \mathbf{r}_L^{(\lambda-1)}\left(T_{0,L}\right) + \sum_{i=1}^{j} (\mathbf{B}\mathbf{P}_L)^{j-i} \sum_{k=i-1}^{s-1} \beta_{ki} \sum_{\lambda=1}^{L} \left(T_{\lambda,L}\right)^{k-i+1} \mathbf{w}_k^{[\lambda]} \right]$$

end for
if $L < L_{\max}$ **then**
for $n_L = 0, \ldots, p_L - 1$ **do**
$\mathbf{y}_{new} = \text{MLTS-RK}(\Delta t/p_L, L+1, \mathbf{y}_{new})$
end for
else
for $n_L = 0, \ldots, p_L - 1$ **do**
$\mathbf{y}_{new} = \text{RK}\ (\Delta t/p_L, L, \mathbf{y}_{new})$
end for
end if
return $\mathbf{y}_{new}$

Algorithm 4 $\mathbf{y}_{new} = \text{RK}\ (\Delta t, L, \mathbf{y}_{old})$

for $r = 1, \ldots, s$ **do**

$$\mathbf{k}_r = \sum_{j=0}^{s-1} \sum_{\ell=0}^{L-1} \left(c_r\Delta t + T_{\ell+1,L}\right)^j \mathbf{w}_j^{[\ell+1]} + \sum_{\ell=0}^{L-1} (\mathbf{P}_\ell - \mathbf{P}_{\ell+1}) \mathbf{r}_\ell\left(c_r\Delta t + T_{0,L}\right) + \mathbf{P}_L \mathbf{F}\left(c_r\Delta t + T_{0,L}\right) + \mathbf{B}\mathbf{P}_L \left(\mathbf{y}_{old} + \Delta t \sum_{i=1}^{r-1} a_{ri}\mathbf{k}_i \right), \quad r = 1, \ldots, s,$$

$$\mathbf{y}_{new} = \mathbf{y}_{old} + \Delta t \sum_{i=1}^{s} b_i \mathbf{k}_i.$$

end for

5. Analysis of Accuracy and Cost. In this section we present some theoretical results for the multi-level methods. In Section 5.1 we prove that all MLTS-RKs methods are accurate of order k, which is the order of the underlying RKs scheme. In Section 5.2 we calculate the computational cost of the MLTS-RKs methods and show that the extra cost compared to an ideal method is negligible if the number of unknowns N is large.

5.1. Accuracy. Consider our model problem

$$\begin{aligned} \frac{\mathrm{d}\mathbf{y}}{\mathrm{d}t}(t) &= \mathbf{B}\mathbf{y}(t) + \mathbf{F}(t)\,, \qquad t \in (0,T)\,, \\ \mathbf{y}(0) &= \mathbf{y}_0\,. \end{aligned} \tag{23}$$

To prove accuracy we shall need some general assumptions. Again let $\{\widetilde{c}_1, \dots, \widetilde{c}_{s_0}\} \subset \{c_1, \dots, c_s\}$ be the maximal subset of all coefficients c_i such that no two are identical, i.e. $\widetilde{c}_i \neq \widetilde{c}_j$ if $i \neq j$. We assume that $s_0 \geq k-1$, i.e. we have enough distinct notes, a condition fulfilled by standard explicit RK methods. We further assume that $\mathbf{F} \in C^{s_0}([0,T])$.

To prove that the MLTS-RK methods are accurate of order k we shall use the accuracy of the LTS-RK schemes, proven in [17]:

PROPOSITION 1. *Let $\mathbf{y} \in C^{k+1}(0,T)$ be the solution of (23) and $\mathbf{y}_{n+1}$ the solution obtained with an LTS-RK method given the exact solution at time t_n (defined by Algorithm 1 in [17] with $\mathbf{y}_n = \mathbf{y}(t_n)$). Then $\|\mathbf{y}(t_{n+1}) - \mathbf{y}_{n+1}\|_\infty = \mathcal{O}(\Delta t^{k+1})$, i.e., the LTS-RK method is kth-order accurate.*

We are now ready to show that the MLTS-RK methods are accurate of order k, independently of the number of levels L. From now on MLTS-RK(L) denotes the multilevel scheme with L refinement levels.

THEOREM 2. *Let $\mathbf{y} \in C^{k+1}(0,T)$ be the exact solution of (23) and $\mathbf{y}_{n+1}$ the numerical solution obtained with an MLTS-RK(L) method given the exact solution at time t_n, i.e., $\mathbf{y}_{n+1} = \text{MLTS-RK}(\Delta t, 1, \mathbf{y}(t_n), \mathbf{B}, \mathbf{F})$ (see Algorithm 1). Then $\|\mathbf{y}(t_{n+1}) - \mathbf{y}_{n+1}\|_\infty = \mathcal{O}(\Delta t^{k+1})$, i.e., the MLTS-RK($L$) method is kth-order accurate.*

Proof. The proof relies on induction over L. In the induction hypothesis we assume that the result holds for $L-1$ levels, i.e., the MLTS-RK($L-1$) method is kth order accurate.

As in [17] we split the error into two parts

$$\|\mathbf{y}(t_{n+1}) - \mathbf{y}_{n+1}\|_\infty \leq \|\mathbf{y}(t_{n+1}) - \mathbf{y}^{[1]}(\Delta t)\|_\infty + \|\mathbf{y}^{[1]}(\Delta t) - \mathbf{y}_{n+1}\|_\infty\,, \tag{24}$$

where $\mathbf{y}(t_{n+1})$ denotes the exact solution of (23) at time t_{n+1} and $\mathbf{y}^{[1]}(\Delta t)$ the exact solution of the first modified equation (16) at time Δt. The estimate for the first error term,

$$\|\mathbf{y}(t_{n+1}) - \mathbf{y}^{[1]}(\Delta t)\|_\infty = \mathcal{O}(\Delta t^{k+1})\,, \tag{25}$$

follows directly from the proof of Proposition 1. To estimate the second term, we recall that $\mathbf{y}_{n+1}$ is the numerical solution of (23) with the MLTS-RK(L) method. Equivalently, we can interpret it as the numerical solution of the modified equation (16) with the MLTS-RK($L-1$) method. Thus, by the induction hypothesis

$$\|\mathbf{y}^{[1]}(\Delta t) - \mathbf{y}_{n+1}\|_\infty = \mathcal{O}(\Delta t^{k+1})\,. \tag{26}$$

Hence,

$$\|\mathbf{y}(t_{n+1}) - \mathbf{y}_{n+1}\|_\infty \leq \mathcal{O}(\Delta t^{k+1}) + \mathcal{O}(\Delta t^{k+1}) = \mathcal{O}(\Delta t^{k+1})\,, \tag{27}$$

i.e., the L-level method is kth order accurate.

Now consider the base case $L = 1$. For $L = 1$ the multilevel algorithm reduces to the LTS-RK method and the estimate (26) follows as in [17], i.e., $\mathbf{y}_{n+1}$ is the numerical solution of (16) with the underlying kth order RK method and thus (26) holds. By induction, it follows that the MLTS-RK(L) method is kth order accurate for $L = 1, 2, \dots$. ∎

5.2. Computational cost. In this section we analyze the computational cost of the methods derived in Section 4. The theoretical speed-up of the LTS-RKs method over a standard RKs method with a small time-step in the entire computational domain was discussed in [17] for the case where $\mathbf{F}(t) = 0$. Here, we extend the calculations to nonzero $\mathbf{F}(t)$. We further investigate how the degrees of freedom in the border region between coarse and fine elements contribute to the computational cost. For simplicity we shall ignore the cost of vector additions and scalar-vector multiplications. We also assume that the spatial discretization is local in the sense that in each multiplication by $\mathbf{B}$, each unknown is only affected by other unknowns within a radius a. This is the case for both FD and FE methods. In one spatial dimension, a equals the half-bandwidth of $\mathbf{A}$ in (6). We will first consider only one level of refinement, i.e., the LTS-RK algorithm, and then move on to the multilevel case.

A single refinement level

We introduce the notation:

- p_1: refinement ratio
- N: number of degrees of freedom
- r: percentage of fine elements
- bn: cost of multiplying an $n \times 1$ vector by $\mathbf{B}$
- a: the number of elements that information can travel in one multiplication by $\mathbf{B}$
- fn: cost of evaluating $\mathbf{F}$ at n points
- d: number of spatial dimensions

To advance the solution one time-step, an RKs method requires s multiplications by $\mathbf{B}$ and s_0 evaluations of $\mathbf{F}$. An *ideal* time-stepping method based on RKs would require s multiplications by $\mathbf{B}$ and s_0 evaluations of $\mathbf{F}$ for the $(1-r)N$ unknowns in the coarse region. The cost of one multiplication by $\mathbf{B}$ is $b(1-r)N$ and the cost of one evaluation of $\mathbf{F}$ is $f(1-r)N$. Thus, the total cost corresponding to the coarse region is $sb(1-r)N + s_0 f(1-r)N$. For the rN unknowns in the fine region there would be $p_1 s$ multiplications by $\mathbf{B}$ and $p_1 s_0$ evaluations of $\mathbf{F}$. Thus, the total cost of an ideal time-stepping method would be

$$C_{ideal} = sb(1-r)N + s_0 f(1-r)N + p_1 sbrN + p_1 s_0 frN\,. \tag{28}$$

We shall calculate the cost of the LTS-RKs algorithm and compare it with C_{ideal}. Since we expect any LTS scheme based on an RKs method to be at least as expensive as the ideal method, we will say that the LTS-RKs algorithm is efficient if the extra cost compared to C_{ideal} is small.

In the LTS-RKs algorithm we need to compute

$$\mathbf{w}_j^{[1]} = \alpha_j \mathbf{B}(\mathbf{I} - \mathbf{P}_1)\left(\mathbf{B}^j \mathbf{y}_n + \sum_{l=1}^{j} \mathbf{B}^{(j-l)} \mathbf{q}_0^{(l-1)}(t_n)\right), \quad j = 0, \ldots, s-1\,. \tag{29}$$

While the factor of $(\mathbf{I} - \mathbf{P}_1)$ zeros all entries corresponding to the fine region, the factor $\mathbf{B}^j$ will make information spread from the fine region to the coarse. Thus, the matrices $\alpha_j \mathbf{B}(\mathbf{I} - \mathbf{P}_1)\mathbf{B}^j$ are nonzero in the coarse region and in a border region adjacent to the coarse region, which complicates the analysis of the computational cost. Furthermore, it is not obvious how to compute the $\mathbf{w}$:s most efficiently. We therefore start by deriving an efficient algorithm, and then analyze the computational

cost of that algorithm. We introduce a diagonal projection matrix $\mathbf{Z}$ with ones and zeros on the diagonal, such that

$$(\mathbf{I}-\mathbf{P}_1)\mathbf{B}^j\mathbf{Z} = (\mathbf{I}-\mathbf{P}_1)\mathbf{B}^j\,, \quad j=0,1,\ldots,s-1\,. \tag{30}$$

More precisely, we define $\mathbf{Z}$ as the matrix with the least number of ones that satisfies (30). The entries corresponding to the coarse region and a border region around the interface between the coarse and fine regions will be set to 1, the others to 0. There are $(1-r)N$ unknowns in the coarse region and the size of the border region is proportional to $(s-1)aN^{\frac{d-1}{d}}$. Notice that the dimension of the border region is $d-1$, due to the locality of the spatial discretization. The size of the border region can thus be bounded from above by $\gamma(s-1)aN^{\frac{d-1}{d}}$, where the constant γ is independent of N and is related to the size of the interface between coarse and fine elements. In 1D γ simply equals the number of interfaces between coarse and fine regions, i.e. if there is one locally refined region surrounded by coarse region, as in Figure 2, then $\gamma = 2$. In higher dimensions, however, γ depends on the size of the interface surface. The number of ones in $\mathbf{Z}$ is thus approximately $(1-r)N + \gamma(s-1)aN^{\frac{d-1}{d}}$. We further define

$$\widetilde{\mathbf{Z}} = \mathbf{Z} - (\mathbf{I}-\mathbf{P}_1)\,. \tag{31}$$

The matrix $\widetilde{\mathbf{Z}}$ is very sparse, with only $\gamma(s-1)aN^{\frac{d-1}{d}}$ nonzero entries corresponding to the border region. Roughly speaking, the nonzero entries in $\widetilde{\mathbf{Z}}$ correspond to extra computational cost; an ideal LTS method would have $\widetilde{\mathbf{Z}} = 0$. Recall that (in one dimension) a is the half-bandwidth of the spatial discretization matrix, which typically increases with the order of accuracy. Thus, the extra cost of the LTS-RKs method relative to an ideal LTS method typically increases slightly with the order of the spatial discretization.

REMARK 3. *It is possible to derive a more efficient algorithm – and thus a sharper bound on the computational cost – by introducing matrices* $\mathbf{Z}_j$ *such that* $(\mathbf{I}-\mathbf{P}_1)\mathbf{B}^j\mathbf{Z}_j = (\mathbf{I}-\mathbf{P}_1)\mathbf{B}^j, \quad j=0,1,\ldots,s-1$. *However, since the gain is very slight we opt not to do this, for ease of notation.*

We are now in a position to derive an algorithm that computes the $\mathbf{w}$:s efficiently by using $\mathbf{w}_j^{[1]}$ to compute $\mathbf{w}_{j+1}^{[1]}$, etc. Due to the properties of $\mathbf{Z}$, we have

$$\begin{aligned}
\mathbf{w}_j^{[1]} &= \alpha_j\mathbf{B}(\mathbf{I}-\mathbf{P}_1)\left(\mathbf{B}^j\mathbf{y}_n + \sum_{l=1}^{j}\mathbf{B}^{(j-l)}\mathbf{q}_0^{(l-1)}(t_n)\right)\\
&= \alpha_j\mathbf{B}(\mathbf{I}-\mathbf{P}_1)\left((\mathbf{B}\mathbf{Z})^j\mathbf{y}_n + \sum_{l=1}^{j}(\mathbf{B}\mathbf{Z})^{j-l}\mathbf{Z}\mathbf{q}_0^{(l-1)}(t_n)\right)\\
&= \alpha_j\mathbf{B}(\mathbf{I}-\mathbf{P}_1)\mathbf{v}_j
\end{aligned} \tag{32}$$

where we have defined

$$\mathbf{v}_j = (\mathbf{B}\mathbf{Z})^j\mathbf{y}_n + \sum_{l=1}^{j}(\mathbf{B}\mathbf{Z})^{j-l}\mathbf{Z}\mathbf{q}_0^{(l-1)}(t_n)\,. \tag{33}$$

We find that the relation between $\mathbf{v}_{j+1}$ and $\mathbf{v}_j$ is

$$\begin{aligned}\mathbf{v}_{j+1} &= (\mathbf{BZ})^{j+1}\mathbf{y}_n + \sum_{l=1}^{j+1}(\mathbf{BZ})^{j+1-l}\mathbf{Z}\mathbf{q}_0^{(l-1)}(t_n) \\ &= \mathbf{BZ}(\mathbf{BZ})^{j}\mathbf{y}_n + \mathbf{BZ}\sum_{l=1}^{j}(\mathbf{BZ})^{j-l}\mathbf{Z}\mathbf{q}_0^{(l-1)}(t_n) + \mathbf{Z}\mathbf{q}_0^{(j)}(t_n) \\ &= \mathbf{BZ}\mathbf{v}_j + \mathbf{Z}\mathbf{q}_0^{(j)}(t_n)\,.\end{aligned} \tag{34}$$

We also observe that with (31)

$$\mathbf{w}_j^{[1]} = \alpha_j\mathbf{B}(\mathbf{I}-\mathbf{P}_1)\mathbf{v}_j = \alpha_j\mathbf{B}(\mathbf{Z}-\widetilde{\mathbf{Z}})\mathbf{v}_j = \alpha_j\left(\mathbf{v}_{j+1} - \mathbf{Z}\mathbf{q}_0^{(j)}(t_n) - \mathbf{B}\widetilde{\mathbf{Z}}\mathbf{v}_j\right)\,, \tag{35}$$

where (34) was used in the last step. Solving (35) for $\mathbf{v}_{j+1}$ yields

$$\mathbf{v}_{j+1} = \alpha_j^{-1}\mathbf{w}_j^{[1]} + \mathbf{Z}\mathbf{q}_0^{(j)}(t_n) + \mathbf{B}\widetilde{\mathbf{Z}}\mathbf{v}_j\,. \tag{36}$$

The relations (32) and (36) lead to an efficient algorithm, presented in Algorithm 5, that computes $\mathbf{w}_j^{[1]}$ for $j = 0, \ldots, s-1$. There are s multiplications by $\mathbf{B}(\mathbf{I}-\mathbf{P}_1)$, which

Algorithm 5 Algorithm to compute **w**:s

```
Set v_0 = y_n
Compute w_0^[1] = α_0 B(I − P_1)v_0
for j = 0, ..., s − 2 do
    Compute v_{j+1} = α_j^{-1} w_j^[1] + Z q_0^(j)(t_n) + B Z̃ v_j
    Compute w_{j+1}^[1] = α_{j+1} B(I − P_1)v_{j+1}
end for
```

require $sb(1-r)N$ work in total. The work corresponding to the $s-1$ multiplications by $\mathbf{B}\widetilde{\mathbf{Z}}$ is

$$(s-1)\cdot b\gamma(s-1)aN^{\frac{d-1}{d}} = (s-1)^2 b\gamma a N^{\frac{d-1}{d}}\,.$$

We further need to construct the polynomials $\mathbf{Z}\mathbf{q}_0^{(j)}(t)$. This requires s_0 evaluations of $\mathbf{F}$ for $(1-r)N + \gamma(s-1)aN^{\frac{d-1}{d}}$ unknowns, i.e. $s_0 f\left((1-r)N + \gamma(s-1)aN^{\frac{d-1}{d}}\right)$ work.

Once the **w**:s have been computed, it remains to solve the modified ODE (13) using p_1 time steps with the RKs method. In each time step there are s multiplications by $\mathbf{BP}$, with total cost $sbrN$. There are also s_0 evaluations of $\mathbf{F}_1 = \mathbf{P}_1\mathbf{F}$, which cost s_0frN in total. The cost for p_1 time-steps in the fine region is $p_1sbrN + p_1s_0frN$. Thus, the total cost for the LTS-RKs algorithm is

$$\begin{aligned}C_{LTS-RK} = {} & s_0 f\left((1-r)N + \gamma(s-1)aN^{\frac{d-1}{d}}\right) + sb(1-r)N + (s-1)^2 b\gamma a N^{\frac{d-1}{d}} \\ & + p_1 sbrN + p_1 s_0 frN\,.\end{aligned} \tag{37}$$

The extra cost relative to the ideal method is

$$\begin{aligned}\frac{C_{LTS-RK} - C_{ideal}}{C_{ideal}} &= \frac{s_0(s-1)f\gamma aN^{\frac{d-1}{d}} + \gamma(s-1)^2 baN^{\frac{d-1}{d}}}{s_0 f(1-r)N + sb(1-r)N + p_1 sbrN + p_1 s_0 frN} \\ &= \frac{\gamma a}{N^{1/d}}\cdot\frac{s_0(s-1)f + (s-1)^2 b}{s_0 f(1-r) + sb(1-r) + p_1 sbr + p_1 s_0 fr}\end{aligned} \tag{38}$$

The extra cost compared to the ideal method is negligible if $\gamma a << N^{1/d}$. Recall that s_0, s, f, a, γ, and b are constants that do not depend on the mesh size. Thus, the extra cost tends to zero as the mesh is refined.

Multiple levels of refinement

To analyze the multilevel case we introduce the notation

- r_ℓ: percentage of unknowns belonging to level ℓ
- p_ℓ: refinement ratio between level ℓ and $\ell - 1$, where $p_0 = 1$
- $\mathtt{p}_\ell = p_1 p_2 \cdots p_\ell$
- γ_ℓ: constant such that size of the border region corresponding to the interface between levels ℓ and $\ell + 1$ does not exceed $\gamma_\ell (s-1) a N^{\frac{d-1}{d}}$

The cost of an ideal time-marching scheme in the presence of $L_{\max}$ levels of refinement would be

$$C_{ideal} = \sum_{\ell=0}^{L_{\max}} \mathtt{p}_\ell s_0 f r_\ell N + \mathtt{p}_\ell s b r_\ell N \,, \tag{39}$$

where $r_\ell N$ is the number of unknowns on level ℓ and $\mathtt{p}_\ell$ is the number of local time-steps performed on level ℓ during one global time-step.

For simplicity we shall assume in this section that the different levels are *separated*, which means that elements belonging to level ℓ are not directly adjacent to elements on level k if $|\ell - k| > 1$. We further assume that the levels are *well separated*, which means that

$$\mathbf{P}_\ell \mathbf{B} (\mathbf{P}_{\ell-2} - \mathbf{P}_{\ell-1}) = 0 \,, \quad \ell = 2, ..., L_{\max} \,. \tag{40}$$

If the levels are *separated* they will become *well separated* eventually as the mesh is refined, which motivates the second assumption. We would like to stress that the method in no way relies on the above assumptions – they merely simplify the analysis of the computational cost. If the levels are well separated the different levels in the MLTS-RK method decouple and the method becomes simply a recursive application of the LTS-RK method. The cost corresponding to $L_{\max}$ levels of refinement is thus easily deduced from (37):

$$\begin{aligned} C_{MLTS-RK} = & \sum_{\ell=0}^{L_{\max}} \mathtt{p}_\ell s_0 f r_\ell N + \mathtt{p}_\ell s b r_\ell N \\ & + \sum_{\ell=0}^{L_{\max}-1} \mathtt{p}_\ell (s-1)^2 b \gamma_\ell a N^{\frac{d-1}{d}} + \mathtt{p}_\ell s_0 f (s-1) \gamma_\ell a N^{\frac{d-1}{d}} \,. \end{aligned} \tag{41}$$

Similar to the case with one refinement, the extra cost compared to the ideal method is negligible if $\gamma_\ell a << N^{1/d}$, $\ell = 0, \ldots, L_{\max} - 1$.

Having derived an expression for the cost, we shall now investigate the theoretical speed-up of the MLTS-RK method over the naïve approach of using the underlying RK scheme with the tiny time-step $\Delta t / \mathtt{p}_{L_{\max}} = \Delta t / (p_1 \cdot \ldots \cdot p_{L_{\max}})$ in the entire computational domain. The naïve approach requires $\mathtt{p}_{L_{\max}}$ time-steps of size $\Delta t / \mathtt{p}_{L_{\max}}$ to integrate (10) from t_n to t_{n+1}. Thus, the cost is

$$C_{RK} = \mathtt{p}_{L_{\max}} (s_0 f N + s b N) \,. \tag{42}$$

To simplify further calculations we assume that $\mathbf{F} = 0$. In this case, by (41) and (42), the ratio of $C_{MLTS-RK}$ over C_{RK} is given by

$$Q = \frac{\sum_{\ell=0}^{L_{\max}} \mathtt{p}_\ell s r_\ell + \sum_{\ell=0}^{L_{\max}-1} \mathtt{p}_\ell (s-1)^2 \gamma_\ell a N^{-1/d}}{\mathtt{p}_{L_{\max}} s}. \tag{43}$$

We assume that N, a, s, and $L_{\max}$ are fixed and investigate how Q varies with p_i and r_i. For simplicity we set $L_{\max} = 2$ and consider 1 spatial dimension, i.e. $d = 1$. We assume that the locally refined regions are contiguous such that $\gamma_0 = \gamma_1 = 2$. We choose the multilevel scheme based on RK3, which has three stages, and thus $s = 3$. If we combine this time integration with mass-lumped $\mathcal{P}^2$ FEM in space to keep the convergence order 3, we can assume that $a = 2$. We further choose $N = 100$. If N increases, the effect of the terms $\mathtt{p}_\ell(s-1)^2\gamma_\ell a N^{-1/d}$ decreases, as mentioned before, which means that the efficiency of the MLTS-RK method only improves.

In a first test we fix r_1 and r_2, and thus $r_0 = 1 - r_1 - r_2$, and study how Q varies with p_1 and p_2. We see in Fig. 5 that when $p_1 = p_2 = 1$, we actually have $Q > 1$,

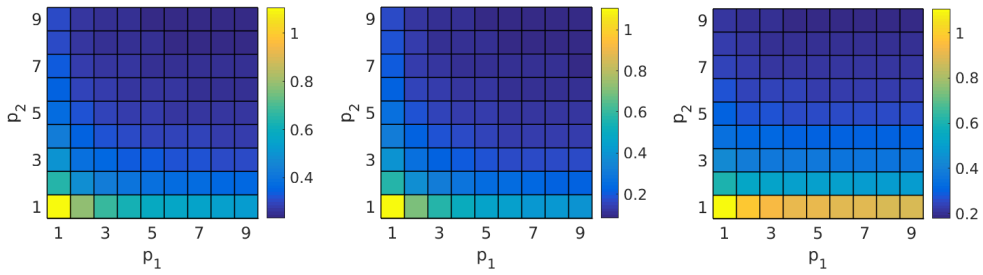


FIG. 5. *Q for fixed $(r_1, r_2) = (0.2, 0.2)$, $(0.2, 0.05)$, $(0.7, 0.1)$.*

which would imply that the MLTS scheme is less efficient than the naïve approach. However, recall that when $p_1 = p_2 = 1$ there is no mesh refinement, and hence no reason to use an MLTS method. The extra cost associated with the second sum in (43) makes the MLTS method slightly more expensive than the standard approach in this case. However, whenever $p_1 > 1$ or $p_2 > 1$, we observe $Q < 1$, which means that the MLTS method is more efficient than the naïve approach. The extra cost relative to an ideal MLTS method is thus small enough that there is something to gain in using the MLTS method, even when N is as small as 100. As p_1 and p_2 increase, so does the gain in using the MLTS method, for all values of r_1 and r_2.

In the second test we fix p_1 and p_2 and investigate how Q varies with r_1 and r_2. The results are presented in Figure 6. Since $r_0 + r_1 + r_2 = 1$, we have $r_1 + r_2 \leq 1$. It is clear that the MLTS scheme is efficient whenever r_1 and r_2 are small, i.e., when most of the elements are coarse. As r_1 and r_2 increase, the gain in using the MLTS method decreases. In particular, as r_2 approaches one the MLTS method and the naïve approach are approximately equally efficient, which is to be expected since when r_2 is close to one, most of the mesh elements are fine and using the small time-step everywhere is a reasonable strategy.

6. Numerical Results. In this section we first consider a model problem with four different levels of refinement in one spatial dimension. We discretize in space using both SBP-SAT and FEM, and verify the accuracy of the MLTS-RKs schemes by comparing the results with an exact solution. To show the versatility of the

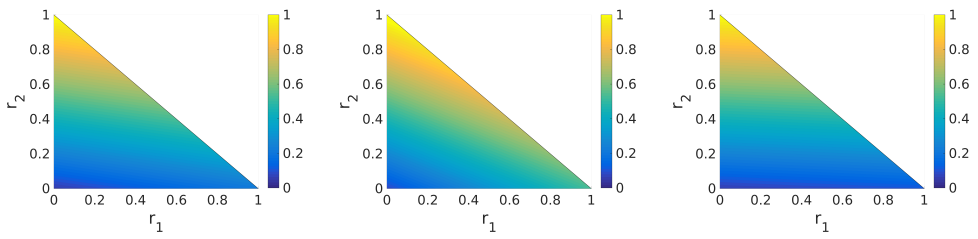


FIG. 6. *Q for fixed $(p_1, p_2) = (5, 5)$, $(10, 2)$, $(2, 10)$.*

MLTS-RKs methods in more realistic applications, we present in Section 6.2 two 2-D examples where local mesh refinement is key for efficient simulation. The Butcher tableaus of the RKs schemes used in the numerical experiments are displayed in Table 4.

6.1. Experiments in 1D. To demonstrate that the MLTS-RKs methods retain the convergence order of the corresponding RKs scheme when combined with SBP-SAT FD (second and fourth order) and continuous $\mathcal{P}^{s-1}$ mass-lumped FEM we consider (1) with $c = 1$ and $\sigma = 0.1$ on the interval $\Omega = [0, 12]$. The initial data $u_0(x) = \sin \pi x$, $v_0(x) = 0$, homogeneous Dirichlet boundary conditions, and the source

$$f(x,t) = \sin \pi x \left((\pi^2 c^2 - \omega^2) \cos \omega t - \sigma \omega \sin \omega t \right) \tag{44}$$

yield the exact solution

$$u(x,t) = \sin \pi x \cos \omega t \,. \tag{45}$$

In the experiments presented here we choose $\omega = 5$. To generate the computational grid we start with an initial mesh with mesh size h^{coarse} on the entire interval $[0, 12]$. The intervals $[0, 2]$ and $[8, 10]$ remain coarse while we refine all the other intervals in a first step by a factor of p_1. We then further refine the intervals $[4, 8]$ and $[10, 12]$ by a factor of p_2 and finally only the interval $[6, 8]$ with factor p_3. The mesh size in $[6, 8]$ is thus $p_1 p_2 p_3$ times smaller than the mesh size in the coarse part. Fig. 7 illustrates what such a mesh could look like. This mesh was designed to test several different aspects of the multilevel scheme. Notice in particular that the levels are not separated (as was assumed in the computational cost analysis in Section 5.2), as level 3 is directly adjacent to level 0 at $x = 8$.

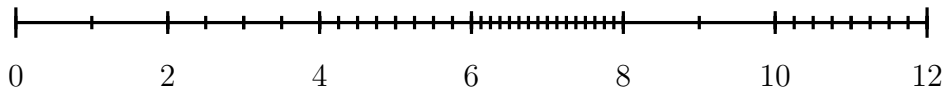


FIG. 7. *Grid with different refined regions in 1D.*

We discretize (1) in space using both SBP-SAT FD (second and fourth order) and continuous $\mathcal{P}^{s-1}$ mass-lumped FEM. Both discretization approaches result in a system of ODEs of the form (10). To advance (10) from 0 to the final time T we take $\mathtt{p}_3 = p_1 p_2 p_3$ local time-steps of size $\Delta t^{[L_{\max}]} = \Delta t / \mathtt{p}_3$ for every global time-step Δt. Here, Δt is the optimal time-step of the underlying RKs scheme on an equidistant mesh of size h^{coarse} and thus independent of the refinement ratios p_ℓ.

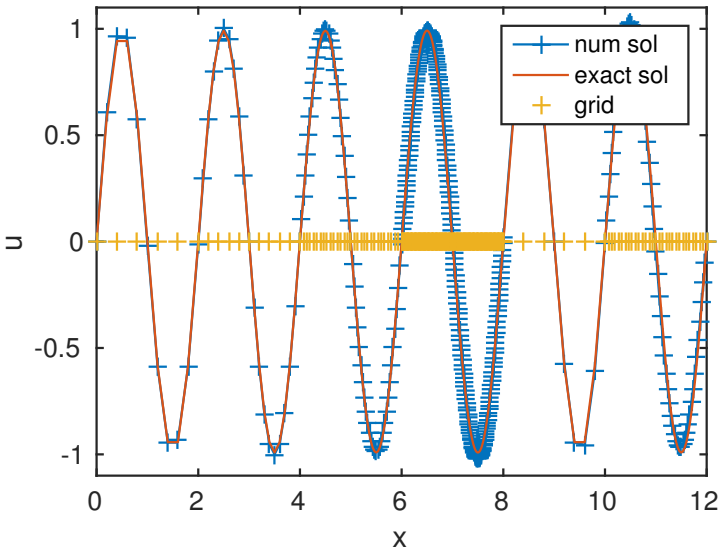


FIG. 8. *Numerical and exact solution for MLTS-RK3 combined with $\mathcal{P}^2$ FE with $h^{coarse} = 0.4$ at time $T = 5$.*

In Fig. 8 we present the exact and the numerical solution at time $T = 5$ on a mesh with $h^{\text{coarse}} = 0.4$, $p_1 = 2$, $p_2 = 3$, and $p_3 = 4$. Here we opt for a spatial discretization with $\mathcal{P}^2$ mass-lumped FE and combine this with the MLTS-RK3 scheme to achieve overall third order convergence. We then simultaneously reduce h^{coarse} and Δt and monitor the L^2-error $\|u(\cdot, T) - u_h^N(\cdot)\|_{L^2(\Omega)}$ (for FEM) or its discrete counterpart, the l^2-error (for SBP-SAT), at the final time T. The convergence rates at time $T = 5$ for the MLTS-RK schemes based on RK2, RK3 and RK4 combined with suitable spatial discretizations to match their accuracy are shown in Figure 9. When using SBP-SAT in space we use one grid point of overlap for MLTS-RK2 and MLTS-RK3, and 3 grid points of overlap for MLTS-RK4, cf. Section 3.2. We observe the expected convergence rates in all cases, which corroborates the accuracy analysis in Section 5.1. Furthermore, we used the optimal time-step in all calculations without experiencing instabilities, i.e., the MLTS-RKs methods show optimal stability behaviour, just as their 1-level variant LTS-RKs – see [17].

6.2. Experiments in 2D. As a first 2D example we consider (1) with homogeneous Dirichlet conditions on the rectangular domain $[-6, 6] \times [-24, 24]$, with a square and a circular inclusion. We set $f = 0$, $c = 1$, and $\sigma = 0.01$. As initial data we use two Gaussian pulses centered at $(0, 10)$ and $(0, -10)$. We here use the 4th order SBP-SAT discretization combined with MLTS-RK4. Figure 10 shows a coarse grid. The leftmost picture highlights the coarsest grid cells, where we use the time-step Δt. The second picture from the left highlights grid cells that are up to 3 times smaller than the coarsest cells; in these cells we use the local time-step $\Delta t/3$. The cells highlighted in the third picture are 3-6 times smaller than the coarsest cells; here we use the local time-step $\Delta t/6$. Finally, the cells highlighted in the fourth picture are 6-12 times smaller than the coarsest cells; here we use the local time-step $\Delta t/12$. That is, we have $L_{\max} = 3$, $p_1 = 3$, $p_2 = 2$, and $p_3 = 2$. The computations were performed

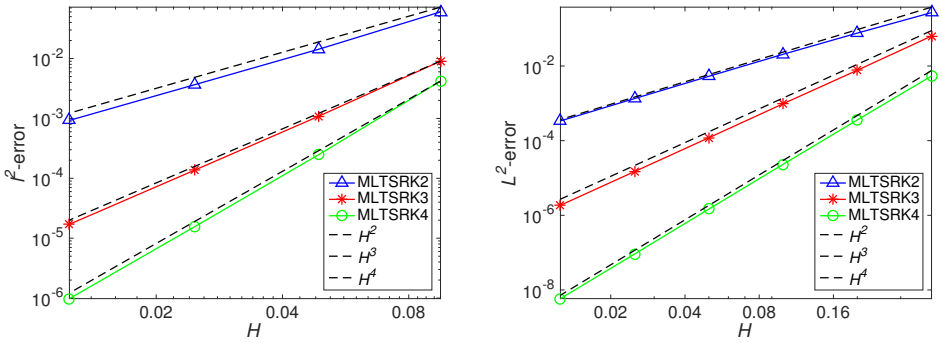


FIG. 9. *Errors of the MLTS-RK schemes at time $T = 5$ as functions of $H = h^{\text{coarse}}$, combined with SBP-SAT (left) and FEM (right).*

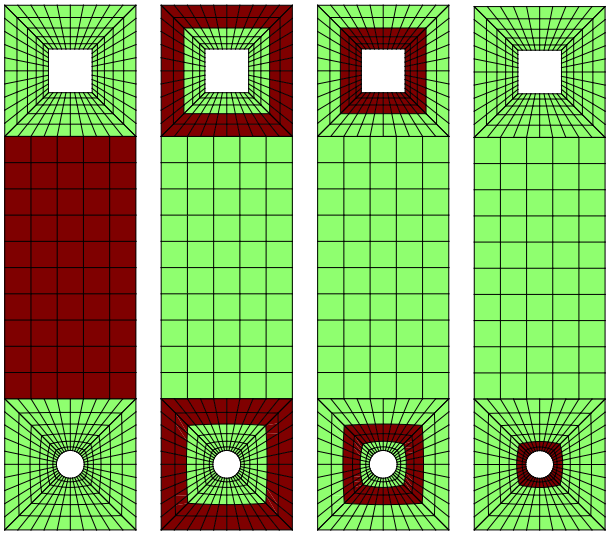


FIG. 10. *The different levels. In the red regions we use, from left to right, the (local) time steps Δt, $\Delta t/3$, $\Delta t/6$, and $\Delta t/12$. Around the square there are 3 levels, and around the circle there are 4 levels.*

on a much finer grid with approximately 260000 grid points, where the coarsest grid spacing was $h^{\text{coarse}} \approx 0.15$. Figure 11 shows the time evolution of the solution from $t = 0$ to $t = 15$.

To show the versatility of the multilevel approach in the presence of a more complex geometry, we consider (1) combined with homogeneous Neumann boundary conditions on the rectangle $[-0.5, 2.5] \times [-0.5, 1.5]$ with two enclosures forming a narrow gap inside. We set $f = 0$, $c = 1$, $\sigma = 0.01$, and final time $T = 2$. As initial data we choose a Gaussian pulse centered at $(0.5, 0.5)$. To resolve the small features in the gap and the initial localized pulse properly, we refine the mesh locally in those two areas. This results in a mesh with 3 levels of refinement – see Fig. 12 for the initial mesh and for zooms on refined regions. The elements in blue are up to 2 times smaller than the coarse elements. The elements in green are up to 6 times finer and

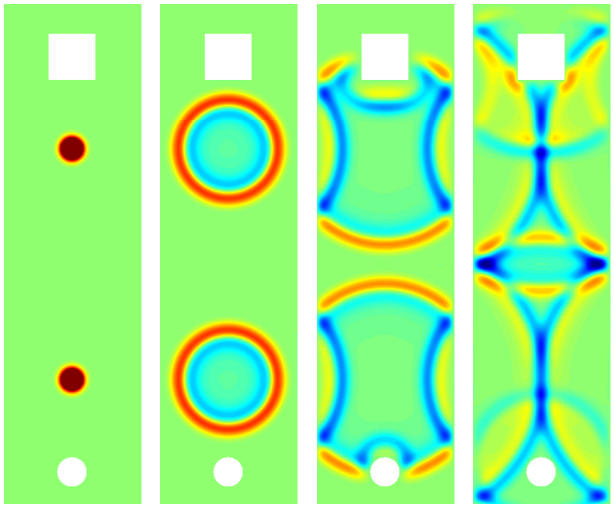


FIG. 11. *Snapshots of the solution at times* $t = 0,\ 5,\ 10,\ 15$.

finally the red ones are around 12 times smaller than the coarsest elements, i.e. we have $p_1 = 2$, $p_2 = 3$ and $p_3 = 2$. For a more accurate simulation we refine the initial mesh in Fig. 12 three times, each time dividing every element into four. Thus, the final mesh consists of about 220000 elements.

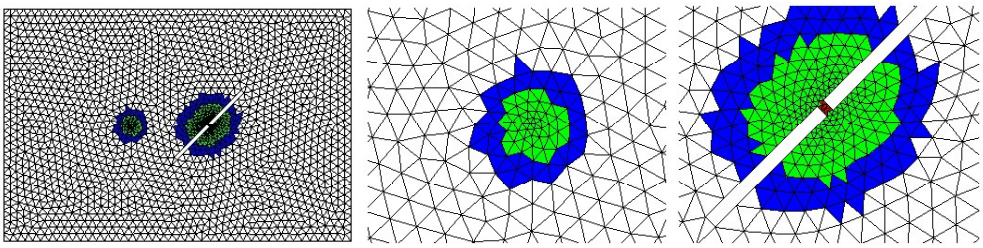


FIG. 12. *Initial mesh and zoom on refined regions of initial mesh*

We use $\mathcal{P}^2$ mass-lumped finite elements to discretize in space. To keep the 3rd order convergence we solve the resulting ODE with the MLTS-RK3 scheme. The initial condition excites a circular wave, which impinges on the obstacle. When the wave reaches the narrow gap another circular wave is generated, which then interacts with the propagating wave front. Snapshots of the solution at intermediate times can be seen in Fig. 13.

7. Concluding Remarks. Starting from the Runge–Kutta-based local time-stepping (LTS-RK) methods in [17], we have derived multi-level local time-stepping (MLTS-RK) methods. If the spatial mesh contains nested patches of refinement, the MLTS-RK methods apply the same multi-level structure to the time-stepping. Thus, the appropriate time-step dictated by the CFL condition can be used within each region of like-sized elements.

Suppose that we can distinguish $L_{\max} + 1$ levels of refinement in our mesh. That

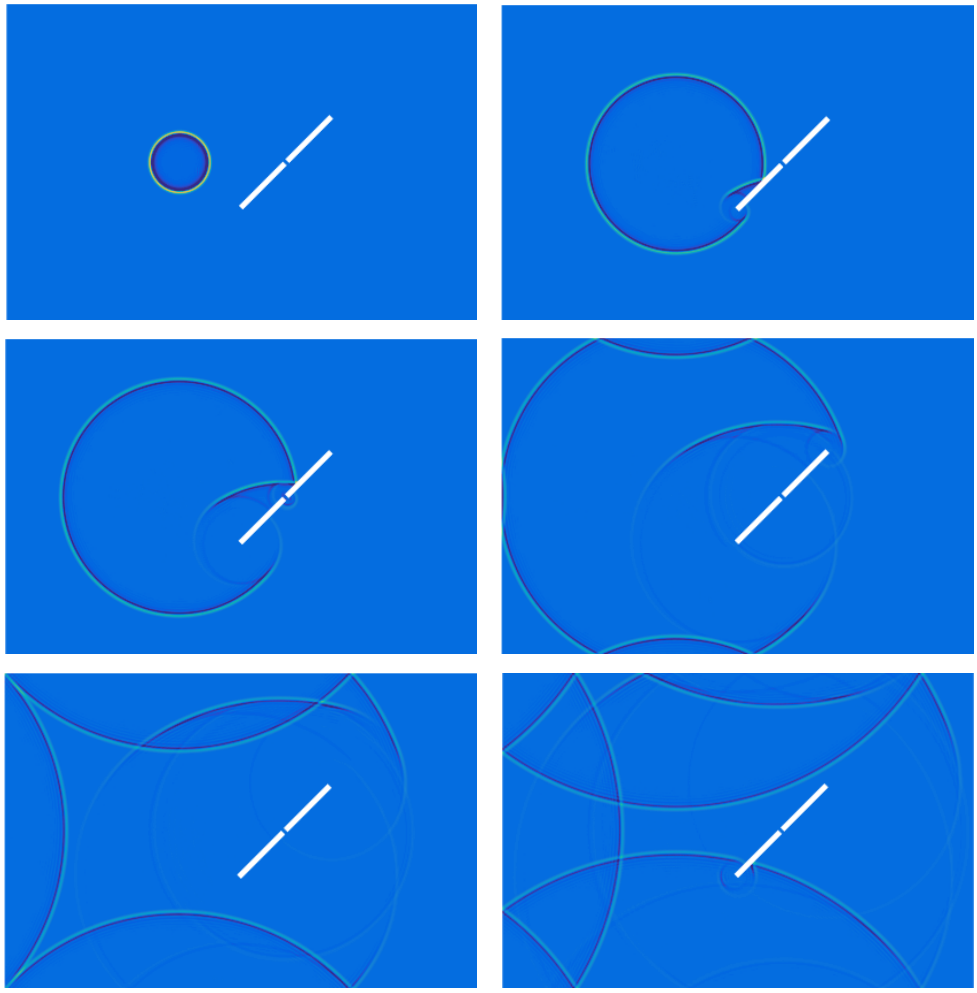


FIG. 13. *Two-dimensional example: the solution is shown at times t =0.19, 0.56, 0.74, 1.12, 1.49 and 1.86.*

is, we can divide the mesh into $L_{\max}+1$ levels or tiers of like-sized elements. We number the levels from 0 to $L_{\max}$, with 0 corresponding to the coarsest level. To each refined level we can associate a refinement ratio p_L, $L = 1, \ldots, L_{\max}$, where p_L is determined by the ratio between the smallest mesh sizes of levels $L-1$ and L. Let Δt denote the time-step dictated by the CFL condition in the coarsest part of the mesh. During every global time-step of size Δt, the MLTS-RK methods take p_1 local time steps of size $\Delta t^{[1]} = \Delta t/p_1$ with the degrees of freedom associated with level 1. During every local time-step of size $\Delta t^{[1]}$, the unknowns associated with level 2 are advanced using p_2 steps of size $\Delta t^{[2]} = \Delta t^{[1]}/p_2$, and so on. Thus, during every global time-step, the MLTS-RK method takes $p_1 \cdots p_L$ local time-steps of size $\Delta t/(p_1 \cdots p_L)$ with the unknowns associated with level L. The MLTS-RK methods are given by Algorithm 1. Due to the inherent recursion over the levels, the algorithms can be formulated recursively. Nonetheless, they remain fully explicit and thus highly parallel.

If the underlying RK method has order k, we have proved that the corresponding

MLTS-RK method retains the same accuracy, independently of the number of levels and the refinement ratios. Our numerical experiments indicate that if an MLTS-RK method of order k is combined with a $\mathcal{P}^{q-1}$ finite element or qth order finite difference spatial discretization, the numerical solution converges to the true solution with optimal rate $\mathcal{O}(h^q + \Delta t^k)$ as $h, \Delta t \to 0$. Furthermore, our numerical experiments suggest that the MLTS-RK methods preserve the optimal CFL condition at each level of refinement.

The derivation of the MLTS-RK methods applies to a general explicit RK method of arbitrary order, including e.g. low-storage RK methods [5] and the low-dispersion low-dissipation RK methods [22].

Acknowledgments. We thank Marcus J. Grote for helpful discussions during early stages of this work.

Appendix A. Runge-Kutta methods. In Table 4 we list the coefficients of the classical s-stage explicit RK methods.

TABLE 4
Coefficients of the classical RKs methods.

0		
1	1	
	$\frac{1}{2}$	$\frac{1}{2}$

(a) RK2 (order 2)

0			
$\frac{1}{2}$	$\frac{1}{2}$		
1	-1	2	
	$\frac{1}{6}$	$\frac{4}{6}$	$\frac{1}{6}$

(b) RK3 (order 3)

0				
$\frac{1}{2}$	$\frac{1}{2}$			
$\frac{1}{2}$	0	$\frac{1}{2}$		
1	0	0	1	
	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{6}$

(c) RK4 (order 4)

Appendix B. Derivation of modified ODE. The first modified equation was derived in [17],

$$\begin{aligned} \frac{\mathrm{d}\mathbf{y}^{[1]}}{\mathrm{d}t^{[1]}}(t^{[1]}) &= \mathbf{B}_1\mathbf{y}^{[1]}(t^{[1]}) + \mathbf{F}_1(t^{[1]}) \qquad t^{[1]} \in (0, \Delta t)\,, \\ \mathbf{y}^{[1]}(0) &= \mathbf{y}_n\,, \end{aligned} \tag{46}$$

with matrix and right-hand side defined as

$$\mathbf{B}_1 = \mathbf{B}\mathbf{P}_1\,, \tag{47}$$

$$\mathbf{F}_1(t^{[1]}) = \sum_{j=0}^{s-1} \left(t^{[1]}\right)^j \mathbf{w}_j^{[1]} + (\mathbf{I} - \mathbf{P}_1)\mathbf{q}_0(t_n + t^{[1]}) + \mathbf{P}_1\mathbf{F}(t_n + t^{[1]})\,, \tag{48}$$

where

$$\mathbf{w}_j^{[1]} = \alpha_j \mathbf{B}(\mathbf{I} - \mathbf{P}_1)\left(\mathbf{B}^j\mathbf{y}_n + \sum_{\ell=1}^{j} \mathbf{B}^{j-\ell}\mathbf{q}_0^{(\ell-1)}(t_n)\right). \tag{49}$$

Here $\mathbf{q}_0(t)$ denotes the interpolation polynomial of degree $s_0 - 1$ going through the points $(t_n + c_i\Delta t, \mathbf{F}(t_n + c_i\Delta t))$, $i = 1, \ldots, s$. Similarly, the second modified ODE is

$$\begin{aligned} \frac{\mathrm{d}\mathbf{y}^{[2]}}{\mathrm{d}t^{[2]}}(t^{[2]}) &= \mathbf{B}_2\mathbf{y}^{[2]}(t^{[2]}) + \mathbf{F}_2(t^{[2]})\,, \qquad t^{[2]} \in (0, \Delta t^{[1]})\,, \\ \mathbf{y}^{[2]}(0) &= \mathbf{y}_{n_1}^{[1]}\,, \end{aligned} \tag{50}$$

where

$$\mathbf{B}_2 = \mathbf{B}_1\mathbf{P}_2 = \mathbf{B}\mathbf{P}_2\,, \tag{51}$$

$$\mathbf{F}_2(t^{[2]}) = \sum_{j=0}^{s-1} \left(t^{[2]}\right)^j \mathbf{w}_j^{[2]} + (\mathbf{I}-\mathbf{P}_2)\mathbf{q}_1(t_{n_1}^{[1]} + t^{[2]}) + \mathbf{P}_2\mathbf{F}_1(t_{n_1}^{[1]} + t^{[2]})\,, \tag{52}$$

$$\mathbf{w}_j^{[2]} = \alpha_j \mathbf{B}_1(\mathbf{I}-\mathbf{P}_2)\left(\mathbf{B}_1^j \mathbf{y}_{n_1}^{[1]} + \sum_{\ell=1}^{j} \mathbf{B}_1^{j-\ell}\mathbf{q}_1^{(\ell-1)}(t_{n_1}^{[1]})\right). \tag{53}$$

Here $\mathbf{q}_1$ is an interpolation polynomial of $\mathbf{F}_1$, going through the quadrature points

$$(t_{n_1}^{[1]} + c_i\Delta t^{[1]}, \mathbf{F}_1(t_{n_1}^{[1]} + c_i\Delta t^{[1]})),\ i = 1,\dots,s\,.$$

We notice, however, that $\mathbf{F}_1$ is of the form (48) and thus consists mainly of polynomial terms. Instead of approximating the entire $\mathbf{F}_1$ in the coarse part by an interpolation polynomial we choose $\mathbf{q}_1$ as

$$\mathbf{q}_1(t^{[1]}) = \sum_{j=0}^{s-1} (t^{[1]})^j \mathbf{w}_j^{[1]} + (\mathbf{I}-\mathbf{P}_1)\mathbf{q}_0(t_n + t^{[1]}) + \mathbf{P}_1\mathbf{r}_1(t_n + t^{[1]})\,,$$

where $\mathbf{r}_1$ is the unique interpolation polynomial of $\mathbf{F}$ of degree $s_0 - 1$ going through the points

$$\left(t_n + t_{n_1}^{[1]} + c_i\Delta t^{[1]}, \mathbf{F}(t_n + t_{n_1}^{[1]} + c_i\Delta t^{[1]})\right),\ i = 1,\dots,s\,.$$

Hence $\mathbf{q}_1$ is also unique and the derivatives are given by

$$\begin{aligned}\mathbf{q}_1^{(\ell-1)}(t_{n_1}^{[1]}) = &\sum_{k=\ell-1}^{s-1} \frac{k!}{(k-\ell+1)!}(t_{n_1}^{[1]})^{k-\ell+1}\mathbf{w}_k^{[1]} + (\mathbf{I}-\mathbf{P}_1)\mathbf{q}_0^{(\ell-1)}(t_n + t_{n_1}^{[1]}) \\ &+ \mathbf{P}_1\mathbf{r}_1^{(\ell-1)}(t_n + t_{n_1}^{[1]}).\end{aligned} \tag{54}$$

Remark 4. *For $s = s_0$ the polynomial $\mathbf{q}_1$ coincides with the interpolation polynomial in Section 4.1.*

Inserting (54) into (53) and using that $(\mathbf{P}_1 - \mathbf{P}_2)(\mathbf{I} - \mathbf{P}_1) = 0$ yields

$$\begin{aligned}\mathbf{w}_j^{[2]} = \alpha_j \mathbf{B}(\mathbf{P}_1-\mathbf{P}_2)\Bigg((\mathbf{B}\mathbf{P}_1)^j\mathbf{y}_{n_1}^{[1]} + \sum_{\ell=1}^{j}(\mathbf{B}\mathbf{P}_1)^{j-\ell}\Bigg[\sum_{k=\ell-1}^{s-1}\beta_{k\ell}(t_{n_1}^{[1]})^{k-\ell+1}\mathbf{w}_k^{[1]} \\ +\mathbf{P}_1\mathbf{r}_1^{(\ell-1)}(t_n + t_{n_1}^{[1]})\Bigg]\Bigg)\,,\end{aligned} \tag{55}$$

where we have introduced the notation

$$\beta_{k\ell} = \frac{k!}{(k-\ell+1)!}\,.$$

Using equations (51) and 52 we can write the ODE in (50) as

$$\begin{aligned}\frac{\mathrm{d}\mathbf{y}^{[2]}}{\mathrm{d}t^{[2]}}(t^{[2]}) = &\sum_{j=0}^{s-1}\left((t_{n_1}^{[1]}+t^{[2]})^j\mathbf{w}_j^{[1]}+(t^{[2]})^j\mathbf{w}_j^{[2]}\right)+\mathbf{B}\mathbf{P}_2\mathbf{y}^{[2]}(t^{[2]})\\ &+(\mathbf{I}-\mathbf{P}_1)\mathbf{q}_0(t_n+t_{n_1}^{[1]}+t^{[2]})+(\mathbf{P}_1-\mathbf{P}_2)\mathbf{r}_1(t_n+t_{n_1}^{[1]}+t^{[2]})\\ &+\mathbf{P}_2\mathbf{F}(t_n+t_{n_1}^{[1]}+t^{[2]}).\end{aligned} \tag{56}$$

We can repeat the above procedure to derive the Lth modified ODE,

$$\begin{aligned}\frac{\mathrm{d}\mathbf{y}^{[L]}}{\mathrm{d}t^{[L]}}(t^{[L]}) = &\sum_{j=0}^{s-1}\sum_{\ell=0}^{L-1}\left(t^{[L]}+T_{\ell+1,L-1}\right)^j\mathbf{w}_j^{[\ell+1]}+\mathbf{B}_L\mathbf{y}^{[L]}(t^{[L]})\\ &+\mathbf{P}_L\mathbf{F}\left(t^{[L]}+T_{0,L-1}\right)+\sum_{\ell=0}^{L-1}(\mathbf{P}_\ell-\mathbf{P}_{\ell+1})\mathbf{r}_\ell\left(t^{[L]}+T_{0,L-1}\right),\\ \mathbf{y}^{[L]}(0) = &\,\mathbf{y}_{n_{L-1}}^{[L-1]},\end{aligned} \tag{57}$$

for $t^{[L]}\in(0,\Delta t^{[L-1]})$. The $\mathbf{r}_\ell$ are the interpolation polynomials of $\mathbf{F}$ in the quadrature points

$$\left(T_{0,\ell}+c_i\Delta t^{[\ell]},\mathbf{F}\left(T_{0,\ell}+c_i\Delta t^{[\ell]}\right)\right),\ i=1,\dots,s,$$

and

$$\begin{aligned}\mathbf{w}_j^{[\ell+1]} = \alpha_j\mathbf{B}(\mathbf{P}_\ell-\mathbf{P}_{\ell+1})\Bigg[&(\mathbf{B}\mathbf{P}_\ell)^j\mathbf{y}_{n_\ell}^{[\ell]}+\sum_{\lambda=1}^{j}(\mathbf{B}\mathbf{P}_\ell)^{j-\lambda}\mathbf{r}_\ell^{(\lambda-1)}\left(T_{0,\ell}\right)\\ &+\sum_{i=1}^{j}(\mathbf{B}\mathbf{P}_\ell)^{j-i}\sum_{k=i-1}^{s-1}\beta_{ki}\sum_{\lambda=1}^{\ell}\left(T_{\lambda,\ell}\right)^{k-i+1}\mathbf{w}_k^{[\lambda]}\Bigg].\end{aligned} \tag{58}$$

REFERENCES

[1] L.D. Angulo, J. Alvarez, F.L. Teixeira, M.F. Pantoja and S.G. Garcia, *Causal-path local time-stepping in the discontinuous Galerkin method for Maxwell's equation*, J. Comput. Phys., 256 (2014), pp. 678–695.

[2] U.M. Ascher, S.J. Ruuth and B. Wetton, *Implicit-explicit methods for time-dependent partial differential equations*, SIAM J. Numer. Anal., 32 (1995), pp. 797–823.

[3] G.A. Baker and V.A. Dougalis, *The effect of quadrature errors on finite element approximations for second order hyperbolic equations*, SIAM J. Numer. Anal., 13 (1976), pp. 577–598.

[4] C. Canuto, M.Y. Hussaini, A. Quateroni and T.A. Zang, *Spectral Methods: Fundamentals in Single Domains*, Springer, New York, 2006

[5] M.H. Carpenter and C.A. Kennedy, *Fourth-Order 2N-Storage Runge-Kutta Schemes*, NASA report TM-109112, NASA Langley Research Center, 1994.

[6] J. Chabassier and S. Imperiale, *Introduction and study of fourth order theta schemes for linear wave equations*, J. Comput. Appl. Math., 245 (2013), pp. 194–212.

[7] G. Cohen, P. Joly, J. Roberts and N. Tordjman, *Higher order triangular finite elements with mass lumping for the wave equation*, SIAM J. Numer. Anal., 38 (2001), pp. 2047–2078.

[8] F. Collino, T. Fouquet and P. Joly, *A conservative space-time mesh refinement method for the 1-D wave equation. I. Construction*, Numer. Math., 95 (2003), pp. 197–221.

[9] E. Constantinescu and A. Sandu, *Multirate time stepping methods for hyperbolic conservation laws*, J. Sci. Comput., 33 (2007), pp. 239–278.

[10] A. Demirel, J. Niegemann, K. Busch and M. Hochbruck, *Efficient Multiple Time-Stepping Algorithms of Higher Order*, J. Comput. Phys., 285 (2015), pp. 133–148.

[11] E. Deriaz, *Stability conditions for the numerical solution of convection-dominated problems with skew-symmetric discretizations*, SIAM J. Numer. Anal., 50 (2012), pp. 1058–1085.

[12] S. Descombes, S. Lanteri and L. Moya, *Locally implicit time integration strategies in a discontinuous Galerkin method*, J. Sci. Comput., 56 (2013), pp. 190–218.
[13] J. Diaz and M.J. Grote, *Energy conserving explicit local time-stepping for second-order wave equations*, SIAM J. Sci. Comput., 31 (2009), pp. 1985–2014.
[14] ———, *Multi-level explicit local time-stepping methods for second-order wave equations*, Comp. Meth. Appl. Mech. Engin., 291 (2015), pp. 240–265.
[15] C.W. Gear and D.R. Wells, *Multirate linear multistep methods*, 24 (1984), BIT, pp. 484–502.
[16] M.J. Grote and T. Mitkova, *High-order explicit local time-stepping methods for damped wave equations*, J. Comput. Appl. Math., 239 (2013), pp. 270–289.
[17] M.J. Grote, M. Mehlin and T. Mitkova, *Runge–Kutta-Based Explicit Local Time-Stepping Methods for Wave Propagation*, SIAM J. Sci. Comput., 37 (2015), pp. A747–A775.
[18] M.J. Grote, A. Schneebeli and D. Schötzau, *Discontinuous Galerkin finite element method for the wave equation*, SIAM J. Numer. Anal. 44 (2006), pp. 2408–2431.
[19] M. Günther, A. Kværnø and P. Rentrop, *Multirate partitioned Runge-Kutta methods*, BIT, 41 (2001), pp. 504–514.
[20] E. Hairer, S. Nørsett and G. Wanner, *Solving Ordinary Differential Equations I: Nonstiff Problems*, Springer, 2000.
[21] M. Hochbruck and A. Sturm, *Error analysis of a second order locally implicit method for linear Maxwell's equations*, CRC 1173-Preprint 2015/1, Karlsruhe Institute of Technology, 2015.
[22] F.Q. Hu, M.Y. Hussaini and J.L. Manthey, *Low-dissipation and low-dispersion Runge–Kutta schemes for computational acoustics*, J. Comput. Phys., 124 (1996), pp. 177–191
[23] W. Hundsdorfer, A. Mozartova, V. Savcenco, *Monotonicity Conditions for Multirate and Partitioned Explicit Runge–Kutta Schemes*, Recent Developments in the Numerics of Nonlinear Hyperbolic Conservation Laws, Notes Numer. Fluid Mech. Multidiscip. Des. 120, 2013, pp. 177195.
[24] W. Hundsdorfer and J.G. Verwer, *Numerical solution of time-dependent advection-diffusion-reaction equations*, Springer Series in Computational Mathematics, Springer, Berlin, 33 (2003).
[25] A. Kanevsky, M.H. Carpenter, D. Gottlieb and J.S. Hesthaven, *Application of implicit-explicit high order Runge-Kutta methods to discontinuous-Galerkin schemes*, J. Comput. Phys., 225 (2007), pp. 1753–1781
[26] G.E. Karniadakis and S.J. Sherwin, *Spectral/hp Element for CFD*, 2nd ed., Oxford Univ. Press, Oxford, 2005.
[27] J.E. Kozdon and L.C. Wilcox, *Stable Coupling of Nonconforming, High-Order Finite Difference Methods*, SIAM J. Sci. Comput., 38 (2016), pp. A923–A952.
[28] K. Mattsson and M.H. Carpenter, *Stable and Accurate Interpolation Operators for High-Order Multiblock Finite Difference Methods*, SIAM J. Sci. Comput., 32 (2010), pp. 2298–2320.
[29] K. Mattsson, F. Ham and G. Iaccarino, *Stable and Accurate Wave-Propagation in Discontinuous Media*, J. Comput. Phys., 227 (2008), pp. 8753–8767.
[30] ———, *Stable Boundary Treatment for the Wave Equation on Second-Order Form*, J. Sci. Comput., 41 (2009), pp. 366–383.
[31] K. Mattsson, *Stable Boundary Treatment for the Wave Equation on Second-Order Form*, J. Sci. Comput., 51 (2011), pp. 650–682.
[32] W.A. Mulder, *Higher-order mass-lumped finite elements for the wave equation*, J. Comput. Acoust., 09 (2001), pp. 671–680.
[33] F.L. Müller and C. Schwab, *Finite Elements with mesh refinement for wave equations in polygons*, J. Comput. Appl. Math., 283 (2015), pp. 163–181.
[34] S. Piperno, *Symplectic local time-stepping in non-dissipative DGTD methods applied to wave propagation problems*, Modél. Math. Anal. Numér., 40 (2006), pp. 815–841.
[35] J.R. Rice, *Split Runge-Kutta method for simultaneous equations*, J. of Res. Nat. Bureau of Standards-B, 64B (1960), pp. 151–170.
[36] M. Rietmann, D. Peter, O. Schenk, B. Uar, and M.J. Grote, *Load-balanced local time stepping for large-scale wave propagation*, In Parallel and Distributed Processing Symposium (IPDPS), 2015 IEEE International, pp. 925–935.
[37] M. Svärd and J. Nordström, *Review of summation-by-parts schemes for initialboundary-value problems*, J. Comput. Phys., 268 (2014), pp. 17–38.
[38] A. Taube, M. Dumbser, C.-D. Munz and R. Schneider, *A high-order discontinuous Galerkin method with time-accurate local time stepping for the Maxwell equations*, Int. J. Numer. Model., 22 (2009), pp. 77–103.
[39] J.G. Verwer, *Component splitting for semi-discrete Maxwell equations*, BIT Numer. Math.,

51 (2010), pp. 427–445.
[40] K. Virta and K. Mattsson, *Acoustic Wave Propagation in Complicated Geometries and Heterogeneous Media*, J. Sci. Comput., 61 (2014), pp. 90–118.
[41] S. Wang, K. Virta and G. Kreiss, *High order finite difference methods for the wave equation with non-conforming grid interfaces*, J. Sci. Comput., (2016), pp. 1–27.