A Provably Stable, Non-iterative Domain Decomposition Technique for the Advection-Diffusion Equation

Oskar Ålund and Jan Nordström

LiTH-MAT-R--2016/15--SE
A PROVABLY STABLE, NON-ITERATIVE DOMAIN DECOMPOSITION TECHNIQUE FOR THE ADVECTION–DIFFUSION EQUATION

OSKAR ÅLUND AND JAN NORDSTRÖM

Abstract. We describe an efficient, non-iterative domain decomposition approach for the one-dimensional advection–diffusion equation based on the Summation-by-Parts technique in both time and space. A fully discrete multidomain analogue of the continuous equation is formulated and a linear system consisting only of the solution components involved in the coupling between the subdomain interfaces is isolated. Once the coupling system is solved, the full solution is found by computing linear combinations of known vectors, weighted by the coupling components. Both stability and invertibility of the discrete scheme is proved using standard Summation-by-Parts procedures.

In a numerical study we show that perfunctory implementations of monodomain Summation-by-Parts based time integration can be improved upon significantly. Using our proposed method we are able to reduce execution time and memory footprint by up to 80% and 95% respectively. Similar improvements in execution time is shown also when compared against explicit Runge–Kutta time integration.

Key words. domain decomposition, stability, summation-by-parts

AMS subject classifications. 65M55, 65M06

1. Introduction. A common domain decomposition procedure involves formulating a general class of partial differential equations on a single domain, followed by an equivalent multidomain formulation. The multidomain formulation is then used to construct an iterative scheme which can be employed using any discretization method. Notable contributors include [12] and [10]. Various approaches exist depending on if the subdomains overlap [3] or not [6], but as a rule, the methods are iterative. There are exceptions, such as a finite difference based domain decomposition algorithm [2], and the explicit-implicit domain decomposition methods [13].

Our approach is similar in principle to the one used in [2]. It is non-iterative and uses non-overlapping subdomains, and subdomain intercommunication is limited to the problem of computing interface components, whence interior and boundary points may be computed in parallel. Key differences arise in the treatment of boundary and initial conditions, which in our schemes is done weakly through penalty terms. Also, our time integration is fully implicit, whereas [2] uses explicit time-stepping for the interface components.

Since our schemes are formulated in terms of general discrete differential operators known as Summation-by-Parts (SBP) operators, we gain most of the convenience commonly associated with them. For example, it is trivial to adjust the order of the derivative approximations in our schemes by simply switching the operators. Furthermore, the theoretical properties of SBP operators – augmented with Simultaneous Approximation Terms (SATs) for weakly enforcing boundary conditions – provide a general and straightforward way to prove stability for a multitude of discretized problems by mimicking continuous energy estimates [1, 8].

Traditionally, the SBP-SAT technique has been used in space to formulate high-order semi-discrete schemes. Such schemes typically take the form of a linear system of ordinary differential equations, which is integrated in time using explicit methods. The groundwork for employing SBP-SAT also as a method of time integration was laid in [9]. However, naive usage of SBP in time produces schemes that, while provably stable and high order accurate, lead to large systems which are difficult to solve...
efficiently in multiple dimensions.

This article is an initial attempt at combining SBP in time and domain decomposition in order to address this efficiency problem. We consider provably stable SBP based domain decomposition methods for the one-dimensional advection and advection–diffusion equations, where local solutions are coupled at the subdomain interfaces using SATs. The coupling procedure follows the ideas in [1], with adjustments to account for the use of SBP in time [9, 7]. Our main result involves isolating a linear system consisting only of solution components at the subdomain interfaces. The system itself, as well as the systems we need to solve in order to construct it, are orders of magnitudes smaller than the original, full system.

In section 2 we outline some of the basic concepts and notation used in the report. Section 3 introduces the SBP-SAT based multidomain formulation in the context of the one-dimensional advection equation. It is used as a stepping stone for understanding the more involved advection–diffusion problem described in section 4. In section 5 we present numerical studies, comparing our proposed scheme both to a single domain scheme based on SBP time integration, and to a traditional SBP-SAT scheme based on explicit Runge-Kutta time integration. Section 6 contains a brief summary of our work and possible future research directions.

2. Preliminaries. In order for this text to retain both mathematical and intuitive clarity it is necessary to familiarize the reader with concepts and notations that will be used throughout the coming sections. A common drawback when working with discretized versions of continuous functions is reliance on heavy notation. Sometimes this is unavoidable, such as in the context of a stability proof, where it is necessary that we know the exact structure of involved vectors and operators. However, in other contexts, heavy notation serves only to obfuscate intuition. We therefore introduce a notational framework which is both intuitive and mathematically precise.

2.1. Discretization. Let $T$ and $L$ be positive real numbers and suppose $u : [0, T] \times [0, L] \to \mathbb{R}$ is a function of time and space. Let $\mathcal{T}$ and $\mathcal{X}$ be discretizations of the temporal and spatial intervals $[0, T]$ and $[0, L]$, such that $\mathcal{T} = \{t_0, t_1, \ldots, t_M\}$ and $\mathcal{X} = \{x_0, x_1, \ldots, x_N\}$, where

\begin{equation}
\begin{aligned}
0 &= t_0 < t_1 < \cdots < t_M = T \\
0 &= x_0 < x_1 < \cdots < x_N = L.
\end{aligned}
\end{equation}

We define the discrete version $\vec{u}$ of $u$ by expanding it first in time and next in space,

\[
\vec{u} = \begin{bmatrix}
    u_1 \\
    u_2 \\
    \vdots \\
    u_M
\end{bmatrix}, \quad \text{where} \quad u_i = \begin{bmatrix}
    u(t_i, x_0) \\
    u(t_i, x_1) \\
    \vdots \\
    u(t_i, x_N)
\end{bmatrix}.
\]

Note that this yields an intuitive map between $\vec{u}$ and $u$ since $u_{ij} = u(t_i, x_j)$.

Similarly, functions of only time or space, $g : [0, T] \to \mathbb{R}$, $f : [0, L] \to \mathbb{R}$, have discrete counterparts $\vec{f}$, $\vec{g}$:

\[
\vec{f} = \begin{bmatrix}
    f(x_0) \\
    f(x_1) \\
    \vdots \\
    f(x_N)
\end{bmatrix}, \quad \vec{g} = \begin{bmatrix}
    g(t_0) \\
    g(t_1) \\
    \vdots \\
    g(t_M)
\end{bmatrix}.
\]
We will generally skip the vector notation since it is almost always clear from the context what is meant (e.g., the symbol \( f \) will be used to refer to both the continuous and discrete version).

Discretized functions are ubiquitous throughout the text and we define the spaces

\[
F_T, \quad F_X, \quad F_{T,X},
\]

consisting of discretized functions of \( t, x \) and \((t, x)\) respectively (note that as sets these spaces are just \( \mathbb{R}^{M+1}, \mathbb{R}^{N+1} \) and \( \mathbb{R}^{(M+1)(N+1)} \); the new notation is introduced to clarify how the elements in the sets should be interpreted and indexed). It is often convenient to be able to regard a discretized function \( \bar{u} \in F_{T,X} \) as a function of only time for a fixed spatial component. Similar to the notation \( u(\cdot, x_j) \) in the continuous case, we denote by

\[
\bar{u}_j = \begin{bmatrix} u_{0j} & u_{1j} & \cdots & u_{Mj} \end{bmatrix}^\top \in F_T,
\]

and by \( \bar{u}_i \) the vector

\[
\bar{u}_i = \begin{bmatrix} u_{i0} & u_{i1} & \cdots & u_{iN} \end{bmatrix}^\top \in F_X.
\]

### 2.2. Discrete operators and Kronecker products.

A convenient way to apply the SBP framework in multiple dimensions is by using Kronecker products. This allows one to operate on different dimensions separately. To illustrate this, let \( D_t \) and \( D_x \) be SBP operators (these are discrete differential operators – for details on the properties of SBP operators, see [1, p. 343]) corresponding to the discretizations (1) and let \( I_t \) and \( I_x \) be identity matrices of size \( M + 1 \) and \( N + 1 \) respectively. Then the discrete analogue of differentiation with respect to \( t \) and \( x \) is multiplication by

\[
D_t := D_t \otimes I_x \quad \text{and} \quad D_x := I_t \otimes D_x
\]

respectively. I.e., if \( \bar{u} \in F_{T,X} \) is a discretized version of some differentiable function \( u \), then \((D_t \bar{u})_{ij} \approx \frac{\partial u}{\partial t}(t_i, x_j)\) and \((D_x \bar{u})_{ij} \approx \frac{\partial u}{\partial x}(t_i, x_j)\).

We extend the above convention to arbitrary matrices operating on only the spatial or temporal dimension. Given matrices \( A_t \) and \( B_t \), we define

\[
A_t := A_t \otimes I_x \quad \text{and} \quad B_t := I_t \otimes B_x.
\]

The SAT part of an SBP-SAT formulation works by comparing appropriate components of the numerical solution to data. Selecting specific components of a vector \( \bar{u} \in F_{T,X} \) is typically done using selection operators defined as follows. Let \( e^0_t, e^1_t, \ldots, e^M_t \) be the standard basis in \( \mathbb{R}^{M+1} \):

\[
e^i_t = (0, 0, \ldots, 1_{\text{pos. } i}, 0, 0, \ldots, 0)^\top, \quad i = 0, 1, \ldots, M.
\]

Then the selection operators on \( F_T \) are

\[
E^ij_t = e^i_t (e^j_t)^\top, \quad 0 \leq i, j \leq M.
\]

Note that \( E^ij_t \) is just the zero matrix with a single \( 1 \) at position \((i, j)\). The selection operators extract a given component and repositions it, while zeroing out all other
components:
\[ \vec{g} = (g_0, g_1, \ldots, g_M)^\top \mapsto E_i^j \vec{g} = (0, 0, \ldots, g_j, 0, \ldots, 0)^\top. \]

We define the operators \( E_i^j \) on \( \mathbb{F}_X \) in the same manner. The selection operators on \( \mathbb{F}_T \) and \( \mathbb{F}_X \) induce selection operators \( E_i^j = E_i^j \otimes I_x \) and \( E_i^j = I_t \otimes E_i^j \) on \( \mathbb{F}_{T,X} \) (these operators simply do the repositioning and zeroing for each grid point in space and time respectively).

2.3. \( \mathbb{F}_T, \mathbb{F}_X, \mathbb{F}_{T,X} \) as inner product spaces. Given discretizations \( T \) and \( X \) as in (1), define corresponding SBP operators

\begin{align*}
D_t &= P_t^{-1} Q_t, & D_x &= P_x^{-1} Q_x.
\end{align*}

Here \( P_t \) and \( P_x \) are symmetric positive definite, and the matrices \( Q_x \) and \( Q_t \) satisfy

\[ Q_x + Q_x^\top = \text{diag}(-1, 0, \ldots, 0, 1) \quad \text{and} \quad Q_t + Q_t^\top = \text{diag}(-1, 0, \ldots, 0, 1). \]

The spaces \( \mathbb{F}_T, \mathbb{F}_X \) and \( \mathbb{F}_{T,X} \) form inner product spaces with inner products induced by \( P_t, P_x \) and \( \mathbf{P} := P_t \otimes P_x \) respectively:

\begin{align*}
\langle g, \tilde{g} \rangle_{P_t} &= g^\top P_t \tilde{g}, \\
\langle f, \tilde{f} \rangle_{P_x} &= f^\top P_x \tilde{f}, \\
\langle u, \tilde{u} \rangle_{\mathbf{P}} &= \langle u, \tilde{u} \rangle_{P_t \otimes P_x} = u^\top (P_t \otimes P_x) \tilde{u},
\end{align*}

The above inner products are discrete analogues of integration in the sense that

\begin{align*}
\langle g, \tilde{g} \rangle_{P_t} &\approx \int_0^T g(t) \tilde{g}(t) dt, \\
\langle f, \tilde{f} \rangle_{P_x} &\approx \int_0^L f(x) \tilde{f}(x) dx, \\
\langle u, \tilde{u} \rangle_{\mathbf{P}} &\approx \int_0^T \int_0^L u(t,x) \tilde{u}(t,x) dx dt.
\end{align*}

Note that while we do not use the vector notation from subsection 2.1, the left-hand sides in (6) contain discrete vector versions of the functions appearing on the right-hand sides.

The key strength of the SBP operators (4) comes from the fact that they provide a natural discrete analogue of integration by parts. This is central since energy estimates are most commonly obtained in the continuous setting by using integration by parts in some form. The fact that we can mimic this procedure for discretized problems means that stability proofs can follow the same procedure that was used to obtain the continuous energy estimates. I.e., if we can find an energy estimate for a continuous problem, it is typically also possible to prove stability of an SBP-SAT based discretization of the problem.

In the continuous setting we have, for any differentiable functions \( f, \tilde{f} : [0, L] \to \mathbb{R} \), the familiar integration by parts formula,

\[ \int_0^L f(x) \tilde{f}'(x) dx = f(L) \tilde{f}(L) - f(0) \tilde{f}(0) - \int_0^L f'(x) \tilde{f}(x) dx. \]
Similar formulas hold in the discrete case. The matrix $Q_x$ appearing in the SBP operator (4) satisfies $Q_x + Q_x^T = \text{diag}(-1, 0, 0, \ldots, 1)$. Hence, for any $f, \tilde{f} \in \mathbb{F}_X$, 
\[
\langle f, D_x \tilde{f} \rangle_{P_x} = f^\top Q_x \tilde{f} \\
= f^\top (Q_x + Q_x^T) \tilde{f} \\
= f_N \tilde{f}_N - f_0 \tilde{f}_0 - f^\top Q_x^T \tilde{f} \\
= f_N \tilde{f}_N - f_0 \tilde{f}_0 - \langle D_x f, \tilde{f} \rangle_{P_x}.
\]

Similarly, for any $g, \tilde{g} \in \mathbb{F}_T$, 
\[
\langle g, D_t \tilde{g} \rangle_{P_t} = g_M \tilde{g}_M - g_0 \tilde{g}_0 - \langle D_t g, \tilde{g} \rangle_{P_t}.
\]

Furthermore, for any $u, \tilde{u} \in \mathbb{F}_{T,X}$, 
\begin{align*}
\langle u, D_x \tilde{u} \rangle_{P} &= \langle u_N, \tilde{u}_N \rangle_{P_x} - \langle u_0, \tilde{u}_0 \rangle_{P_x} - \langle D_x u, \tilde{u} \rangle_{P} , \\
\langle u, D_t \tilde{u} \rangle_{P} &= \langle u_M, \tilde{u}_M \rangle_{P_t} - \langle u_0, \tilde{u}_0 \rangle_{P_t} - \langle D_t u, \tilde{u} \rangle_{P} .
\end{align*}

3. The single domain advection equation. Consider the one-dimensional advection problem:

\begin{align*}
u_t + u_x &= F(t, x), \quad (t, x) \in [0, T] \times [0, L] \\
u(t, 0) &= g(t), \quad t \in [0, T] \\
u(0, x) &= f(x), \quad x \in [0, L].
\end{align*}

(8)

It can be shown that (8) is well-posed and it is straightforward to formulate a fully discrete SBP-SAT analogue of this problem. Let $T$ and $X$ be discretizations of the temporal and spatial domains, with corresponding SBP operators $D_t = P_t^{-1}Q_t$ and $D_x = P_x^{-1}Q_x$. Furthermore, let $u \in \mathbb{F}_{T,X}$ denote the vector approximating the exact solution to (8), and define discrete versions $g \in \mathbb{F}_T$, $f \in \mathbb{F}_X$ and $F \in \mathbb{F}_{T,X}$ of the data by evaluating their continuous counterparts in the discretization points.

Then the SBP-SAT analogue of (8) is

\begin{equation}
D_t u + D_x u = F + \alpha P_t^{-1}E^0_t (u - e^0_t f) + \beta P_x^{-1}E^0_x (u - e^0_x g).
\end{equation}

(9)

Excluding the SAT penalties $\alpha P_t^{-1}E^0_t (u - e^0_t f)$ and $\beta P_x^{-1}E^0_x (u - e^0_x g)$, the equation looks similar to the continuous case. The purpose of the penalty terms is to enforce the boundary and initial conditions. The constants $\alpha, \beta \in \mathbb{R}$ can be chosen such that (9) is stable. The matrices $e^0_t = e^0_t \otimes I_x$ and $e^0_x = I_t \otimes e^0_x$ map the initial and boundary data vectors $f$ and $g$ into $\mathbb{F}_{T,X}$ so that they become structurally compatible with $u$. The restructured initial data $e^0_t f \in \mathbb{F}_{T,X}$ is zero everywhere except in positions corresponding to the initial time $t_0$, and the restructured boundary data $e^0_x g \in \mathbb{F}_{T,X}$ is zero everywhere except in positions corresponding to the left spatial boundary $x_0$. More precisely,

\[
\begin{cases}
(e^0_t f)_0 = f \\
(e^0_t f)_i = 0, \quad i > 0
\end{cases}
\quad \text{and} \quad
\begin{cases}
(e^0_x g)_0 = g \\
(e^0_x g)_j = 0, \quad j > 0
\end{cases}
\]

The selection operators $E^0_t$ and $E^0_x$ then select the components of $u - e^0_t f$ and $u - e^0_x g$ which should be penalized in accordance with the initial and boundary conditions.
In principle, (9) is ready to be solved using a suitable linear solver. In multiple dimensions, however, this idea quickly becomes infeasible due to memory constraints. With grid refinement one will end up with a system too large to be solved efficiently on modern computers, and it becomes necessary to explore ways to reduce the system size. In this report we will focus on SBP-SAT based multidomain formulations and take advantage of the fact that the solution on a particular subdomain only depends on solutions on the remaining subdomains through a small number of coupling components.

3.1. Spatial multidomain formulation. In this section we outline the basic domain decomposition ideas that will be used for the advection–diffusion equation later. We split the spatial domain into three equally sized subdomains $B_L, B_I, B_R$, and associate to each subdomain an equidistant discretization

$$X_L = \{x_0, x_1, \ldots, x_N\}, \quad X_I = \{y_0, y_1, \ldots, y_N\}, \quad X_R = \{z_0, z_1, \ldots, z_N\},$$

such that $x_N = y_0$ and $y_N = z_0$ (see Figure 1). Note that the subdomains share an interface point with their neighboring subdomain(s). To approximate the exact solution on any particular subdomain we must use data from adjacent subdomains. The solutions are glued together using a SAT treatment similar to the one used to enforce boundary and initial conditions in the previous section.

Let $F_k \in \mathbb{F}_{T,X_k}$, $f_k \in \mathbb{F}_{X_k}$ ($k \in \{L, I, R\}$) and $g \in \mathbb{F}_{T}$ be discrete versions of the data. We look for $u \in \mathbb{F}_{T,X_L}$, $v \in \mathbb{F}_{T,X_I}$, $w \in \mathbb{F}_{T,X_R}$ such that

$$D_t u + D_x u = F_L + \alpha P_t^{-1} E_{t}^{00}(u - e_0^0 f_L)$$
$$+ \beta P_x^{-1} E_{x}^{00}(u - e_0^0 g)$$
$$+ \sigma_R P_x^{-1}(E_{x}^{NN} u - E_{x}^{N0} v)$$

$$D_t v + D_x v = F_I + \alpha P_t^{-1} E_{t}^{00}(v - e_0^0 f_I)$$
$$+ \sigma_I P_x^{-1}(E_{x}^{00} v - E_{x}^{0N} u)$$
$$+ \sigma_R P_x^{-1}(E_{x}^{NN} v - E_{x}^{N0} w)$$

$$D_t w + D_x w = F_R + \alpha P_t^{-1} E_{t}^{00}(w - e_0^0 f_R)$$
$$+ \sigma_I P_x^{-1}(E_{x}^{00} w - E_{x}^{0N} v)$$

(10)

The boundary and initial conditions are dealt with just as in the previous section. Similarly, differences at the interfaces between neighboring solutions are penalized using SATs to enforce continuity of the full solution. The interface SATs are built using selection operators to extract and reposition interface components which need to be compared. This formulation is readily extended to an arbitrary number of subdomains by repeating the middle equation.

It can be shown that the scheme (10) is stable and conservative for $\alpha = \beta = -1$, $\sigma_R \leq 1/2$ and $\sigma_L = \sigma_R - 1$. Note that the scheme (10) penalizes differences in the solution at the interfaces, i.e. $u_N - u_0$ and $v_N - v_0$. The components involved in the coupling between the local solutions are central in our algorithm, warranting the following definition.

**Definition 1.** Any solution components involved in the coupling between local solutions are called coupling components.
Assuming that where the advection–diffusion case) the following system is equivalent to (11)

\[ \begin{align*}
    A_L = D_t + D_x + P_t^{-1}E_t^{00} + P_x^{-1}E_x^{00} - \sigma_R P_x^{-1}E_x^{NN} \\
    b_L = F_L + P_t^{-1}e_t^0 f_L + P_x^{-1}e_x^g \\
    A_I = D_t + D_x + P_t^{-1}E_t^{00} - \sigma_L P_x^{-1}E_x^{00} - \sigma_R P_x^{-1}E_x^{NN} \\
    b_I = F_I + P_t^{-1}e_t^0 f_I \\
    A_R = D_t + D_x + P_t^{-1}E_t^{00} - \sigma_L P_x^{-1}E_x^{00} \\
    b_R = F_R + P_t^{-1}e_t^0 f_R \\
    L_x = \sigma_L P_x^{-1}E_x^{NN} \\
    R_x = \sigma_R P_x^{-1}E_x^{NN}.
\end{align*} \]

Assuming that \( A_L, A_I, \) and \( A_R \) are non-singular, (this is proven in subsection 4.4 for the advection–diffusion case) the following system is equivalent to (11)

\[ \begin{pmatrix}
    L_x & A_I & R_x \\
    A_I & L_x & A_I \\
    A_R & L_x & I
\end{pmatrix}
\begin{pmatrix}
    u \\
    v \\
    w
\end{pmatrix} = \begin{pmatrix}
    b_L \\
    b_I \\
    b_R
\end{pmatrix}
\]

We will see soon that (12) can be reduced to involve only coupling components.

Let us discuss the construction of (12) from a computational perspective. To construct the right-hand side we must compute the three vectors

\[ \begin{align*}
    \tilde{b}_L &= A_L^{-1}b_L, \\
    \tilde{b}_I &= A_I^{-1}b_I, \\
    \tilde{b}_R &= A_R^{-1}b_R
\end{align*} \]

(these quantities are independent and can be computed in parallel). Furthermore we must evaluate the matrix products appearing in \( C \). The matrices \( L_x \) and \( R_x \) have \( M+1 \) nonzero columns, where \( M+1 \) is the number of gridpoints in time. Therefore, evaluating the products \( A_k^{-1}L_x \) and \( A_k^{-1}R_x \), \( k \in \{L, I, R\} \), amounts to computing the following vectors for \( n = 0, 1, \ldots, M \):

\[ \begin{align*}
    \tilde{r}_n^L = A_L^{-1}r_n, \\
    \tilde{r}_n^I = A_I^{-1}r_n, \\
    \tilde{r}_n^R = A_R^{-1}r_n,
\end{align*} \]
where \( r_n \) is the \( n \):th nonzero column of \( R_x \), and \( l_n \) is the \( n \):th nonzero column of \( L_x \). All quantities in (13) are independent and can be computed in parallel. The vectors (13), henceforth known as coupling vectors, are important because they, together with the coupling components and \( b \), contain all the information needed to construct the full solution.

The coupling vectors depend only on the SBP operators and on the form of the boundary conditions, but not on the data itself. Thus, the construction of the left-hand side of (12) can be thought of as the discretization step. It can be done ahead of simulation and need not be redone unless we make changes to the grid or to the SBP operators, or if we change the form of the boundary conditions.

Remark 2. Even if we increase the number of subdomains, the number of systems we need to solve in the discretization step remains unchanged if we use the same grid size and SBP operators (and consequently, the same \( A_I \)) for all interior subdomains. The interior coupling vectors \( \tilde{r}_n \) and \( \tilde{l}_n \) in (13) can then be used for all interior subdomains.

The final step is to reduce the size of (12), such that it can be solved efficiently. By taking advantage of the fact that (12) describes linear relationships between local solutions through coupling components, it is possible to solve for the coupling components separately. Once the coupling components are known they can be used to compute the remaining solution components.

Let us study an artificial but illustrative example by considering the case with a single gridpoint \( t_0 \) in time and three gridpoints in space per subdomain. Obviously a single gridpoint in time is nonsensical; however, reducing the size of (12) is purely an exercise in linear algebra, and the procedure is analogous for an arbitrary number of gridpoints in time and space. With \( M + 1 = 1 \) and \( N + 1 = 3 \), (12) reads

\[
\begin{bmatrix}
1 & 0 & 0 & \tilde{r}_{00}^L & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & \tilde{r}_{01}^L & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & \tilde{r}_{02}^L & 0 & 0 & 0 & 0 \\
0 & 0 & \tilde{r}_{00}^I & 1 & 0 & 0 & \tilde{r}_{00}^I & 0 \\
0 & 0 & \tilde{r}_{01}^I & 0 & 1 & 0 & \tilde{r}_{01}^I & 0 \\
0 & 0 & \tilde{r}_{02}^I & 0 & 0 & 1 & \tilde{r}_{02}^I & 0 \\
0 & 0 & 0 & 0 & \tilde{r}_{00}^R & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & \tilde{r}_{01}^R & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & \tilde{r}_{02}^R & 0 & 0 & 1 \\
\end{bmatrix}
\begin{bmatrix}
u_{00} \\
u_{01} \\
u_{02} \\
v_{00} \\
v_{01} \\
v_{02} \\
w_{00} \\
w_{01} \\
w_{02} \\
\end{bmatrix}
= 
\begin{bmatrix}
\tilde{b}_L^0 \\
\tilde{b}_L^1 \\
\tilde{b}_L^2 \\
\tilde{b}_I^0 \\
\tilde{b}_I^1 \\
\tilde{b}_I^2 \\
\tilde{b}_R^0 \\
\tilde{b}_R^1 \\
\tilde{b}_R^2 \\
\end{bmatrix}.
\]

The coupling components in the system are \( u_{02}, v_{00}, v_{02}, \) and \( w_{00} \). Note that all other solution components are expressed in terms of the coupling components. This means that all rows and columns corresponding to non-coupling components can be removed without losing solvability of the system (this is proved in subsection 4.4 for the advection–diffusion case).
Removing the aforementioned rows results in the following system:

\[
\begin{bmatrix}
0 & 0 & 1 & \tilde{r}_{02}^L & 0 & 0 \\
0 & 0 & l_{00}^I & 1 & 0 & 0 \\
0 & 0 & l_{02}^I & 0 & 1 & \tilde{r}_{02}^I \\
0 & 0 & 0 & 0 & 0 & l_{00}^R \\
\end{bmatrix}
\begin{bmatrix}
0 \\
u_0 \\
u_2 \\
v_0 \\
v_2 \\
w_0 \\
w_{02} \\
\end{bmatrix}
= 
\begin{bmatrix}
\tilde{b}_{0}^L \\
\tilde{b}_{0}^R \\
\tilde{b}_{1}^L \\
\tilde{b}_{1}^R \\
\tilde{b}_{2}^L \\
\tilde{b}_{2}^R \\
\end{bmatrix}
\]

Next we remove the zero columns corresponding to non-coupling components to end up with the coupling system:

\[
\begin{bmatrix}
1 & \tilde{r}_{02}^L & 0 \\
l_{00}^I & 1 & 0 \\
l_{02}^I & 0 & 1 \\
0 & 0 & l_{00}^R \\
\end{bmatrix}
\begin{bmatrix}
u_0 \\
v_0 \\
v_2 \\
w_0 \\
\end{bmatrix}
= 
\begin{bmatrix}
\tilde{b}_{0}^L \\
\tilde{b}_{0}^R \\
\tilde{b}_{1}^L \\
\tilde{b}_{1}^R \\
\tilde{b}_{2}^L \\
\tilde{b}_{2}^R \\
\end{bmatrix}
\]

Hence, once we have solved (15) for the coupling components, everything appearing on the right-hand sides in (16) is known, and the full solution can be computed. From (14) we see that the full solutions \( u, v, \) and \( w \) take the form

\[(16)
\begin{align*}
u &= \tilde{b}_{0}^L - v_{00} l_{00}^L, \\
v &= \tilde{b}_{0}^L - u_{02} l_{00}^I - w_{00} l_{02}^I, \\
w &= \tilde{b}_{0}^R - v_{02} l_{00}^R.
\end{align*}
\]

The procedure is analogous for an arbitrary number of gridpoints, resulting in a coupling system where we solve for the coupling components \( u_N, v_0, v_N, \) and \( w_0 \). The general coupling system can be constructed algorithmically as follows. Use single indices for \( u \) and \( b \) (defined in (12)) so that \( u = ( u_1, u_2, \ldots, u_{3(M+1)(N+1)})^T \) and \( b = ( b_1, b_2, \ldots, b_{3(M+1)(N+1)})^T \). Let \( I_c \) be the index set consisting of single indices corresponding to coupling components. Remove any components of \( u \) and \( b \) with an index not in \( I_c \) to form \( \hat{u} \) and \( \hat{b} \). Remove all rows and columns from the matrix \( C \) in (12) except the ones belonging to \( I_c \) to form \( \tilde{C} \). This yields a system

\[(17)
\hat{C} \hat{u} = \hat{b}
\]

consisting only of coupling components. Once it is solved and all coupling components are known, the full solutions take the form

\[(18)
\begin{align*}
u &= \tilde{b}_{L} - \sum_{i=0}^{M} v_{i0} l_{i}^L, \\
v &= \tilde{b}_{L} - \sum_{i=0}^{M} u_{iN} l_{i}^I - \sum_{i=0}^{M} w_{i0} l_{i}^I, \\
w &= \tilde{b}_{R} - \sum_{i=0}^{M} v_{iN} l_{i}^R.
\end{align*}
\]

Our algorithm can now be succinctly described as follows:

- Compute the coupling vectors (13).
- Compute the right-hand sides \( \tilde{b}_k = A_k^{-1} b_k, k \in \{ L, I, R \} \).
- Construct and solve the coupling system (17).
- Compute the full solution using (18).
Note that while we have restricted ourselves to three subdomains for convenience, all of the above is readily extended to an arbitrary number of subdomains. Additional interior subdomains are treated just like the single interior subdomain \( B_1 \). It is also simple to extend the method to multiple domains in time, using the multiblock form presented in [7, p. 97].

We end this section by comparing the computational work of the above approach to that of solving the monodomain system (9). Assume a total number of \( N \) gridpoints in space and \( M \) gridpoints in time. To find the solution \( u \) in the monodomain case we must solve a linear system of size \( NM \times NM \). In the multidomain approach, assuming we use \( K \) subdomains with \( N = N/K \) spatial gridpoints each, we must first compute \( 4M \) systems of size \( NM \times NM \) for the coupling vectors (however, as discussed, this is a one time cost and can be done in parallel). Then we compute \( K \) additional systems of size \( NM \times NM \) for the right-hand sides \( \tilde{b}^k \). Finally we solve a coupling system of size \( 2(K-1)M \times 2(K-1)M \). Clearly the sizes of the subproblems (involving the coupling vectors and the right-hand sides \( \tilde{b}^k \)) decreases with \( K \), while the size of the coupling system increases with \( K \). The optimal number of subdomains is a topic of further research.

4. The advection–diffusion equation. Consider the problem

\[
\begin{align*}
  u_t + u_x - \epsilon u_{xx} &= F(t,x), \quad (t,x) \in [0,T] \times [0,L] \\
  u(t,0) - \epsilon u_x(t,0) &= g_0(t), \quad t \in [0,T] \\
  \epsilon u_x(t,L) &= g_1(t), \quad t \in [0,T] \\
  u(0,x) &= f(x), \quad x \in [0,L].
\end{align*}
\]

(19)

It can be shown that this problem is well-posed. We proceed, as in the advection case, by formulating a fully discrete (and provably stable) SBP-SAT monodomain analogue of (19). Let the discrete versions of the data be defined as in subsection 2.1. An SBP-SAT analogue of (19) is to find \( u \in F_{T,x} \) such that:

\[
(D_t + D_x - \epsilon D_x^2)u = F + \alpha P_t^{-1}E^{00}_t (u - e^0_t f) + \beta P_x^{-1}E^{00}_x (u - \epsilon D_x u - e^0_x g_0) + \gamma P_x^{-1}E^{NN}_x (\epsilon D_x u - e^N_x g_1),
\]

(20)

where \( \alpha, \beta, \gamma \) are derived in subsection 4.1 to ensure stability of the scheme.

Again, the resulting system grows quickly with mesh refinement and we will try to mitigate computational inefficiency by partitioning the space-time domain.

4.1. Single domain stability. In the continuous case we can find a bound on the \( L^2 \)-norm of the solution to (19) by multiplying the differential equation by \( u \) and integrating in space and time. The analogy for the discrete case is to apply the inner product (5), induced by \( P = P_t \otimes P_x \), to (20). Setting \( F = 0 \) this gives

\[
\langle u, D_t u \rangle_P + \langle u, D_x u \rangle_P = \epsilon \langle u, D_x^2 u \rangle_P + \alpha \langle u, P_t^{-1}E^{00}_t (u - e^0_t f) \rangle_P + \beta \langle u, P_x^{-1}E^{00}_x (u - \epsilon D_x u - e^0_x g_0) \rangle_P + \gamma \langle u, P_x^{-1}E^{NN}_x (\epsilon D_x u - e^N_x g_1) \rangle_P.
\]

(21)
Using the summation-by-parts property (7) on the advection–diffusion terms we find
\begin{align}
\tag{22}
\langle u, D_x u \rangle_P &= \frac{1}{2} (\|u_M\|_{P_1}^2 - \|u_0\|_{P_1}^2), \\
\tag{23}
\langle u, D_x^2 u \rangle_P &= \frac{1}{2} (\|u_N\|_{P_1}^2 - \|u_0\|_{P_1}^2), \\
\tag{24}
\langle u, D_x^2 u \rangle_P &= \langle u_N, (D_x u)_N \rangle_{P_1} - \langle u_0, (D_x u)_0 \rangle_{P_1} - \|D_x u\|_{P_1}^2.
\end{align}

Furthermore, by the definition of the inner product, the first penalty term can be written
\begin{align}
\tag{25}
\langle u, P_t^{-1} E_t^{00} (u - c_t^0 f) \rangle_P &= u^T (E_t^{00} \otimes P_x) (u - (c_t^0 \otimes I_x) f) \\
&= \langle u_0, u_0 - f \rangle_{P_1}.
\end{align}

Similar expressions hold for the remaining penalty terms,
\begin{align}
\tag{26}
\langle u, P_x^{-1} E_x^{00} (u - \epsilon D_x u - \epsilon^0 g_0) \rangle_P &= \langle u_0, u_0 - \epsilon (D_x u)_0 - g_0 \rangle_{P_1}, \\
\tag{27}
\langle u, P_x^{-1} E_x^{NN} (\epsilon (D_x u) - \epsilon^N g_1) \rangle_P &= \langle u_N, \epsilon (D_x u)_N - g_1 \rangle_{P_1}.
\end{align}

Inserting (22)–(27) into (21) and multiplying by two gives an expression of the energy of the solution at the final time,
\begin{align}
\|u_M\|_{P_1}^2 + 2\epsilon \|D_x u\|_{P_1}^2 &= (2\alpha + 1)\|u_0\|_{P_1}^2 - 2\alpha \langle u_0, f \rangle_{P_1} \\
+ &\ (2\beta + 1)\|u_0\|_{P_1}^2 - 2\beta \langle u_0, g_0 \rangle_{P_1} \\
- &\ \|u_N\|_{P_1}^2 + 2\gamma \langle u_N, g_0 \rangle_{P_1} \\
- &\ 2\epsilon (\beta + 1)\langle u_0, (D_x u)_0 \rangle_{P_1} \\
+ &\ 2\epsilon (\gamma + 1)\langle u_N, (D_x u)_N \rangle_{P_1}.
\end{align}

To eliminate the indefinite terms \(\langle u_0, (D_x u)_0 \rangle_{P_1}\) and \(\langle u_N, (D_x u)_N \rangle_{P_1}\) we choose \(\beta = \gamma = -1\), so that
\begin{align}
\|u_M\|_{P_1}^2 + 2\epsilon \|D_x u\|_{P_1}^2 &= (2\alpha + 1)\|u_0\|_{P_1}^2 - 2\alpha \langle u_0, f \rangle_{P_1} \\
- &\ \|u_0\|_{P_1}^2 + 2\langle u_0, g_0 \rangle_{P_1} \\
- &\ \|u_N\|_{P_1}^2 + 2\langle u_N, g_1 \rangle_{P_1}.
\end{align}

Furthermore, if \(\epsilon \in \mathbb{R}\), the following identity holds for arbitrary vectors \(v, w\), and any inner product and its associated norm:
\begin{align}
- \|v\|^2 + c \langle v, w \rangle &= \left(\frac{c}{2}\right)^2 \|w\|^2 - \|v - \frac{c}{2} w\|^2.
\end{align}

Repeated use of this formula in equation (29) yields
\begin{align}
\|u_M\|_{P_1}^2 + 2\epsilon \|D_x u\|_{P_1}^2 &= - \frac{\alpha^2}{2\alpha + 1} \|f\|_{P_1}^2 + (2\alpha + 1)\|u_0\|_{P_1}^2 - \frac{\alpha}{2\alpha + 1} f\|_{P_1}^2 \\
+ &\ \|g_0\|_{P_1}^2 - \|u_0 - g_0\|_{P_1}^2 \\
+ &\ \|g_1\|_{P_1}^2 - \|u_N - g_1\|_{P_1}^2.
\end{align}

From here it is clear that we must choose \(\alpha < -\frac{1}{2}\). The fraction \(-\frac{\alpha^2}{2\alpha + 1}\) is minimized for \(\alpha = -1\), giving the optimal stability estimate
\begin{align}
\|u_M\|_{P_1}^2 + 2\epsilon \|D_x u\|_{P_1}^2 &= \|f\|_{P_1}^2 - \|u_0 - f\|_{P_1}^2 \\
+ &\ \|g_0\|_{P_1}^2 - \|u_0 - g_0\|_{P_1}^2 \\
+ &\ \|g_1\|_{P_1}^2 - \|u_N - g_1\|_{P_1}^2.
\end{align}
4.2. Spatial multidomain formulation. The structure of this section follows that of subsection 3.1 closely, with a few differences due to the slightly more complicated nature of the equation. We will again restrict ourself to three subdomains (see Figure 1) since all arguments are easily extended to the general case.

Combining the interface treatment described in [1] and the penalty terms in (20) we formulate the following multidomain scheme. Find \( u \in \mathbb{F}_{T,x_L}, \ v \in \mathbb{F}_{T,x_I}, \ w \in \mathbb{F}_{T,x_R} \) such that

\[
(D_t + D_x - \epsilon D_x^2)u = F_L - P_t^{-1}E_t^{00}(u - e_t^0f_L) - P_x^{-1}E_x^{00}(u - \epsilon D_xu - e_x^0g_0) + \sigma_L^I P_x^{-1}(E_x^{NN}u - E_x^{N0}v) + \sigma_L^V P_x^{-1}(E_x^{NN}D_xu - E_x^{N0}D_xv) \\
(D_t + D_x - \epsilon D_x^2)v = F_I - P_t^{-1}E_t^{00}(v - e_t^0f_I) + \sigma_L^I P_x^{-1}(E_x^{00}v - E_x^{0N}u) + \sigma_L^V P_x^{-1}(E_x^{00}D_xv - E_x^{0N}D_xu) + \sigma_R^V P_x^{-1}(E_x^{NN}D_xv - E_x^{N0}D_xw) \\
(D_t + D_x - \epsilon D_x^2)w = F_R - P_t^{-1}E_t^{00}(w - e_t^0f_R) + \sigma_L^I P_x^{-1}(E_x^{00}w - E_x^{0N}v) + \sigma_L^V P_x^{-1}(E_x^{00}D_xw - E_x^{0N}D_xv) - P_x^{-1}E_x^{NN}(\epsilon D_xw - e_x^Ng_t)
\]

where the constants \( \sigma_L^I, \sigma_L^V, \sigma_R^V, \sigma_R^L \) must be determined to yield stability (see subsection 4.3). In the SATs used to enforce the boundary and boundary and initial conditions we use the penalty coefficients derived in the previous section.

Just like in subsection 3.1, the interface SATs are constructed using selection operators to extract and reposition interface components (both in \( u, v, w \) themselves and in their derivatives \( D_xu, D_xv, D_xw \)). To ensure stability it is necessary to include penalties at the interfaces also in terms of derivatives. The number of coupling components (recall Definition 1) in the above system therefore depends on the accuracy of the SBP operator \( D_x \). Typically, the boundary stencil of a \((2p,p)\)-accurate [11, p. 7] SBP operator is \( 2p \) wide. For example, assume that \( D_x \) is \((2,1)\)-accurate (all arguments are easily adjusted for any accuracy). This means that to approximate the spatial derivatives at an interface, we need two points from the neighboring local solutions. Thus, the coupling components in the scheme (31) would be

\[
\begin{align*}
&u_{N-1}, \ u_N, \ v_0, \ v_1, \ v_{N-1}, \ v_N, \ w_0, \ w_1.
\end{align*}
\]

By rearranging the terms in (31) we can write it compactly as

\[
\begin{bmatrix}
A_L & R_x \\
L_x & A_I & R_x \\
L_x & A_R
\end{bmatrix}
\begin{bmatrix}
u \\
w
\end{bmatrix}
= 
\begin{bmatrix}
b_L \\
b_I \\
b_R
\end{bmatrix}
\]
where

\[ A_L = D_t + D_x - cD^2_x + P^{-1}_t E_{t}^{00} + P^{-1}_x E_{x}^{00} - cP^{-1}_x E_{x}^{00}D_x \]

\[ - \sigma^L_P P^{-1}_x E_{x}^{NN} - \sigma^V_P P^{-1}_x E_{x}^{NN}D_x \]

\[ b_L = F_L + P^{-1}_t e^0_t f_L + P^{-1}_x e^0_x g_0 \]

\[ A_I = D_t + D_x - cD^2_x + P^{-1}_t E_{t}^{00} \]

\[ - \sigma^L_P P^{-1}_x E_{x}^{00} - \sigma^V_P P^{-1}_x E_{x}^{00}D_x \]

\[ b_I = F_I + P^{-1}_t e^0_t f_I \]

\[ A_R = D_t + D_x - cD^2_x + P^{-1}_t E_{t}^{00} + cP^{-1}_x E_{x}^{NN}D_x \]

\[ - \sigma^L_P P^{-1}_x E_{x}^{00} - \sigma^V_P P^{-1}_x E_{x}^{00}D_x \]

\[ b_R = F_R + P^{-1}_t e^0_t f_R + P^{-1}_x e^0_x g_1 \]

\[ R_x = \sigma^L_P P^{-1}_x E_{x}^{00} + \sigma^V_P P^{-1}_x E_{x}^{00}D_x \]

\[ L_x = \sigma^L_P P^{-1}_x E_{x}^{NN} + \sigma^V_P P^{-1}_x E_{x}^{NN}D_x \]

Once again, assuming that \( A_L, A_I, \) and \( A_R \) are non-singular (proved in subsection 4.4 below), the system (33) is equivalent to

\[
\begin{pmatrix}
I & A_L^{-1}R_x & 0 \\
A_I^{-1}L_x & I & A_I^{-1}R_x \\
A_R^{-1}L_x & 0 & I
\end{pmatrix}
\begin{pmatrix}
u \\
w \\
0
\end{pmatrix}
= C
\begin{pmatrix}u \\
w \\
b
\end{pmatrix}
\]

The only difference in form between (39) and its advection counterpart (12) comes from the fact that \( R_x \) and \( L_x \) have two (assuming a second order SBP operator) nonzero columns per gridpoint in time, rather than one. We will see, however, that this has a negligible effect on the amount of computation needed to evaluate the matrix products \( A_k^{-1}R_x \) and \( A_k^{-1}L_x \).

As an example, consider a single gridpoint \( t_0 \) in time, five gridpoints in space per subdomain (i.e. \( M + 1 = 1 \) and \( N + 1 = 5 \)) and a (2,1)-accurate SBP operator in space. Again, a single gridpoint in time does not make sense, but this degenerate case holds all the information necessary to understand the general case. We choose five gridpoints in space per subdomain to ensure that \( v \) contains some non-coupling components. Recall from (32) that the coupling components in this case are \( u_3, u_4, v_0, v_1, v_3, v_4, w_0, \) and \( w_1 \).

Since \( M + 1 = 1 \) we have, with \( D_x = (d_{ij}) \) and \( P_x = \text{diag}(p_0, \ldots, p_N) \),

\[ R_x = R_x = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ r_0 & r_1 & 0 & 0 & 0 \end{bmatrix} \]

and

\[ L_x = L_x = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \]

where

\[ r_0 = \frac{\sigma^L_P}{p_N} + \frac{\sigma^V_P d_{00}}{p_N}, \quad r_1 = \frac{\sigma^V_P d_{01}}{p_N}, \quad l_0 = \frac{\sigma^V_P d_{N,N-1}}{p_0}, \quad l_1 = \frac{\sigma^L_P}{p_0} + \frac{\sigma^V_P d_{NN}}{p_0}. \]
This follows immediately from the definitions (37)–(38) of $R_x$ and $L_x$. Note that no additional constants appear in the general case, because adding gridpoints in time simply means that $R_x$ and $L_x$ become block-diagonal with the matrices in (40) repeated along the diagonal.

To compute the products $A^{-1}_k R_x$, $k \in \{L, I\}$, and $A^{-1}_k L_x$, $k \in \{I, R\}$, we only need the coupling vectors $\tilde{r}_0^k = A^{-1}_k e^4$ and $\tilde{l}_0^k = A^{-1}_k e^0$, where $e^0 = (1, 0, 0, 0, 0)^T$ and $e^4 = (0, 0, 0, 0, 1)^T$. Then

\begin{equation}
A^{-1}_k R_x = \begin{bmatrix} r_0 \tilde{r}_0^L & r_1 \tilde{r}_0^L & 0 & 0 & 0 \end{bmatrix} \quad \text{and} \quad A^{-1}_k L_x = \begin{bmatrix} 0 & 0 & 0 & l_0^R & l_1^R \end{bmatrix}.
\end{equation}

Note that the zero subscripts on the vectors $\tilde{r}_0^k$ and $\tilde{l}_0^k$ are there to indicate that they belong to the gridpoint $t_0$. In general we must compute coupling vectors for each gridpoint in time as in (13):

\begin{align}
\tilde{r}_n^L &= A^{-1}_L e^{n(N+1)+N} \\
\tilde{r}_n^I &= A^{-1}_I e^{n(N+1)+N} \\
\tilde{l}_n^I &= A^{-1}_I e^{n(N+1)} \\
\tilde{l}_n^R &= A^{-1}_R e^{n(N+1)}
\end{align}

for $n = 0, 1, \ldots, M$.

Inserting (42) into (39) we see that the solutions take the form

\begin{align}
u &= A^{-1}_L b_L - (r_0 v_{00} + r_1 v_{01}) r_0^L \\
v &= A^{-1}_I b_I - (l_0 u_{03} + l_1 u_{04}) l_0^I - (r_0 w_{00} + r_1 w_{01}) r_0^I \\
w &= A^{-1}_R b_R - (l_0 v_{03} + l_1 v_{04}) l_0^R
\end{align}

Or, for an arbitrary number of gridpoints,

\begin{align}
u &= A^{-1}_L b_L - \sum_{i=0}^M (r_0 v_{i0} + r_1 v_{i1}) r_i^L \\
v &= A^{-1}_I b_I - \sum_{i=0}^M (l_0 u_{iN-1} + l_1 u_{iN}) l_i^I - (r_0 w_{i0} + r_1 w_{i1}) r_i^I \\
w &= A^{-1}_R b_R - \sum_{i=0}^M (l_0 v_{iN-1} + l_1 v_{iN}) l_i^R.
\end{align}

Reducing (39) to a coupling system follows the same procedure as in subsection 3.1 and in our example this leads to the following system.

\[ \begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
l_{00}^L & l_{00}^I & 1 & 0 & 0 \\
l_{03}^L & l_{03}^I & 0 & 1 & 0 \\
l_{04}^L & l_{04}^I & 0 & 0 & 1 \\
0 & 0 & 0 & l_{00}^R & l_{00}^I \\
0 & 0 & 0 & l_{03}^R & l_{03}^I \\
0 & 0 & 0 & l_{04}^R & l_{04}^I
\end{bmatrix} \begin{bmatrix}
u_{03} \\
u_{04} \\
u_{00} \\
u_{03} \\
u_{04} \\
u_{00} \\
u_{01} \\
u_{04} \\
u_{01} \\
u_{04} \\
u_{01} \\
u_{04} \\
u_{01}
\end{bmatrix} = \begin{bmatrix}
\tilde{l}_0^L \\
\tilde{l}_0^I \\
\tilde{l}_0^R \\
\tilde{l}_0^I \\
\tilde{l}_0^R \\
\tilde{l}_0^I \\
\tilde{l}_0^R \\
\tilde{l}_0^I \\
\tilde{l}_0^R \\
\tilde{l}_0^I \\
\tilde{l}_0^R \\
\tilde{l}_0^I
\end{bmatrix}. \]
Once this system has been solved, the full solutions can be computed according to (44). In the general case, a coupling system

\[ \hat{\mathbf{C}} \hat{\mathbf{u}} = \mathbf{b} \]

is constructed just like in subsection 3.1, and the full solutions are computed using (45). Again, the only difference between the advection–diffusion case and the pure advection case is the number of coupling components. The algorithm is otherwise identical:

- Compute the coupling vectors (43).
- Compute the right-hand sides \( \tilde{b}^k = A_k^{-1} b_k, k \in \{L, I, R\} \).
- Construct and solve the coupling system (46).
- Compute the full solution using (45).

The discussion at the end of subsection 3.1 holds true also for the advection–diffusion case, with the only difference being the size of the coupling system. Since we need to compare derivatives at the interfaces, the number of coupling components is larger in the advection–diffusion case. More precisely, assuming that the derivative approximations at the interfaces need \( p \) solution components, and that we use \( M \) gridpoints in time, the coupling system will be of size \( 2p(K - 1)M \times 2p(K - 1)M \) (compared to \( 2(K - 1)M \times 2(K - 1)M \) in the advection case), where \( K \) is the number of subdomains.

### 4.3. Multidomain stability

In order for the domain decomposition approach described above to be viable, we need a number of theoretical results. We start by proving stability of the scheme.

Let \( \xi = p_0 = p_N \), where \( p_0 \) and \( p_N \) is the first and last element in the diagonal of \( P_x = \text{diag}(p_0, \ldots, p_N) \) (the weights \( p_0 \) and \( p_N \) are equal since we are using equidistant grids). Then

**Proposition 3.** The multidomain scheme (31) is stable if

\[ \sigma^I_L = \sigma^I_R - 1, \quad \sigma^V_L = \sigma^V_R + 1, \quad \sigma^I_R \leq \frac{1}{2} - \epsilon \left( \frac{(\sigma^V_R)^2 + (\sigma^V_L)^2}{4\xi} \right). \]

**Proof.** Just as in subsection 4.1 we set \( F_L = F_I = F_R = 0 \) and apply the inner product induced by \( P = P_t \otimes P_x \) to (31) to get

\[
\begin{align*}
(u, D_t u)_P + (u, D_x u)_P &= \epsilon (u, D_x^2 u)_P \\
&\quad - (u_0, u_0, - f_L)_{P_x} \\
&\quad - (u_0, u_0 - \epsilon (D_x u)_0 - g_0)_{P_t} \\
&\quad + \sigma^I_R (u, P_t^{-1} (E^{NN}_x u - E^{NO}_x v))_P \\
&\quad + \sigma^V_R \epsilon (u, P_t^{-1} (E^{NN}_x D_x u - E^{NO}_x D_x v))_P.
\end{align*}
\]

15
from subsection 4.1 we find

\[
\langle v, D_v v \rangle_p + \langle v, D_x v \rangle_p = \epsilon \langle v, D_x^2 v \rangle_p \\
- \langle v_0, v_0 - f_t \rangle_p \\
+ \sigma_L^I \langle v, P_x^{-1}(E_x^0 v - E_x^{0N} u) \rangle_p \\
+ \sigma_L^Y \langle v, P_x^{-1}(E_x^0 D_x v - E_x^{0N} D_x u) \rangle_p \\
+ \sigma_R^I \langle v, P_x^{-1}(E_x^{0N} u - E_x^{0N} v) \rangle_p \\
+ \sigma_R^Y \langle v, P_x^{-1}(E_x^{0N} D_x v - E_x^{0N} D_x u) \rangle_p
\]

(49)

\[
\langle w, D_v w \rangle_p + \langle w, D_x w \rangle_p = \epsilon \langle w, D_x^2 w \rangle_p \\
- \langle w_0, w_0 - f_t \rangle_p \\
+ \sigma_L^I \langle w, P_x^{-1}(E_x^0 w - E_x^{0N} v) \rangle_p \\
+ \sigma_L^Y \langle w, P_x^{-1}(E_x^0 D_x w - E_x^{0N} D_x v) \rangle_p \\
- \langle w_0, \epsilon(D_x w)_N - g_t \rangle_p.
\]

(50)

Here we have used the identities from subsection 4.1 on the boundary and initial condition penalty terms. Similar identities hold also for the interface penalty terms. For any \( \phi, \psi \),

\[
\langle \phi, P_x^{-1}(E_x^{0N} \phi - E_x^{0N} \psi) \rangle_p = \langle \phi, N, \phi, N - \psi_0 \rangle_p, \\
\langle \phi, P_x^{-1}(E_x^{0N} D_x \phi - E_x^{0N} D_x \psi) \rangle_p = \langle \phi, N, (D_x \phi)_N - (D_x \psi)_0 \rangle_p, \\
\langle \psi, P_x^{-1}(E_x^{0N} \psi - E_x^{0N} \phi) \rangle_p = \langle \psi_0, \phi_0 - \phi, N \rangle_p, \\
\langle \psi, P_x^{-1}(E_x^{0N} D_x \psi - E_x^{0N} D_x \phi) \rangle_p = \langle \psi_0, (D_x \psi)_0 - (D_x \phi)_N \rangle_p.
\]

Recall also that

\[
\langle \phi, D_t \phi \rangle_p = \frac{1}{2} \left( \| \phi_M \|_{P_x}^2 - \| \phi_0 \|_{P_x}^2 \right), \\
\langle \phi, D_x \phi \rangle_p = \frac{1}{2} \left( \| \phi, N \|_{P_x}^2 - \| \phi_0 \|_{P_x}^2 \right), \\
\langle \phi, D_x^2 \phi \rangle_p = \langle \phi, N, (D_x \phi)_N \rangle_p - \langle \phi_0, (D_x \phi)_0 \rangle_p - \| D_x \phi \|_{P_x}^2.
\]

By using the above formulas in equations (48)–(50) and repeating the arguments from subsection 4.1 we find

\[
\| u_M \|_{P_x}^2 = \| f_L \|_{P_x}^2 - \| u_0 - f_t \|_{P_x}^2 + \| g_0 \|_{P_x}^2 - \| u_0 - g_0 \|_{P_x}^2 + \mathcal{R}(u, v)
\]

(51)

\[
\| v_M \|_{P_x}^2 = \| f_L \|_{P_x}^2 - \| v_0 - f_t \|_{P_x}^2 + \mathcal{L}(v, u) + \mathcal{R}(v, w)
\]

(52)

\[
\| w_M \|_{P_x}^2 = \| f_R \|_{P_x}^2 - \| w_0 - f_t \|_{P_x}^2 + \| g_1 \|_{P_x}^2 - \| v_N - g_1 \|_{P_x}^2 + \mathcal{L}(w, v)
\]

(53)

where, for any \( \phi, \psi \),

\[
\mathcal{R}(\phi, \psi) = (2\sigma_L^I - 1)\| \phi, N \|_{P_x}^2 + 2\epsilon(\sigma_R^Y + 1)\langle \phi, N, (D_x \phi)_N \rangle_p \\
- 2\sigma_L^I \langle \phi, N, \psi_0 \rangle_p - 2\epsilon\sigma_R^Y \langle \phi, N, (D_x \psi)_0 \rangle_p - 2\epsilon\| D_x \phi \|_{P_x}^2
\]

and

\[
\mathcal{L}(\psi, \phi) = (2\sigma_L^I + 1)\| \psi_0 \|_{P_x}^2 + 2\epsilon(\sigma_R^Y - 1)\langle \psi_0, (D_x \psi)_0 \rangle_p \\
- 2\sigma_L^I \langle \psi_0, \phi, N \rangle_p - 2\epsilon\sigma_R^Y \langle \psi_0, (D_x \phi)_N \rangle_p - 2\epsilon\| D_x \psi \|_{P_x}^2.
\]

16
Summing equations (51)–(53) gives

\[
\|u_M \|^2_{P_x} + \|v_M \|^2_{P_x} + \|w_M \|^2_{P_x} = \|f_L \|^2_{P_x} + \|f_I \|^2_{P_x} + \|f_R \|^2_{P_x} + \|g_0 \|^2_{P_x} + \|g_1 \|^2_{P_x} - \|u_0 - f_L \|^2_{P_x} - \|v_0 - f_I \|^2_{P_x} - \|w_0 - g_R \|^2_{P_x}
\]

If we show that the quantity \( R(\phi, \psi) + L(\psi, \phi) \) is non-positive for any \( \phi, \psi \), then the above equality bounds the energy of the solution in terms of data and the proof is complete. To this end, the quantities \( \|D_x \phi\|^2_{P} \) and \( \|D_x \psi\|^2_{P} \) can be written

\[
\|D_x \phi\|^2_{P} = \|D_x \phi\|^2_{P} + \xi \|D_x \phi\|^2_{P}, \quad \|D_x \psi\|^2_{P} = \|D_x \psi\|^2_{P} + \xi \|D_x \psi\|^2_{P},
\]

where

\[
\|D_x \phi\|^2_{P} := \sum_{j=0}^{N-1} P_x(j, j) \sum_{i=0}^{M} P_t(i, i)(D_x \phi)_{j,i}^2,
\]

\[
\|D_x \psi\|^2_{P} := \sum_{j=1}^{N} P_x(j, j) \sum_{i=0}^{M} P_t(i, i)(D_x \psi)_{j,i}^2.
\]

It follows that

\[
R(\phi, \psi) + L(\psi, \phi) = \Phi^T (B \otimes P_t) \Phi - 2\epsilon \|D_x \phi\|^2_{P} - 2\epsilon \|D_x \psi\|^2_{P},
\]

where

\[
B = \begin{bmatrix}
2\sigma_R^I - 1 & -(\sigma_R^L + \sigma_I^L) & \epsilon(\sigma_R^V + 1) & -\epsilon\sigma_R^V \\
-(\sigma_R^L + \sigma_I^L) & 2\sigma_I^L + 1 & -\epsilon\sigma_I^V & \epsilon(\sigma_I^V - 1) \\
\epsilon(\sigma_R^V + 1) & -\epsilon\sigma_I^V & 2\epsilon\xi & 0 \\
-\epsilon\sigma_R^V & \epsilon(\sigma_I^V - 1) & 0 & -2\epsilon\xi
\end{bmatrix}, \quad \Phi = \begin{bmatrix}
\phi_N \\
\psi_0 \\
(D_x \phi)_{N} \\
(D_x \psi)_{0}
\end{bmatrix}.
\]

The matrix \( B \) is derived in [1, p. 346], and shown to be negative semi-definite (Theorem 3.1 ibid.) under the conditions (47), leading to \( R(\phi, \psi) + L(\psi, \phi) \leq 0 \).

### 4.4. Invertibility

In this section we discuss the following three invertibility results.

- Invertibility of \( A_L, A_I, \) and \( A_R \) defined in (34)–(36).
- Invertibility of the matrix (33) defined by the multidomain scheme.
- Invertibility of the coupling matrix \( C \) in (46).

To construct the multidomain system (39) it is necessary that the matrices \( A_L, A_I, \) and \( A_R \) are non-singular. It is not immediately clear that the stable choices (47) of \( \sigma_{L,R}^{I,V} \) in (31) imply invertibility of these matrices, so let us convince ourselves of this fact.

The matrices \( A_L, A_I, \) and \( A_R \) can be written as sums of temporal and spatial parts:

\[
A_L = A_I \otimes I_x + I_I \otimes A_{Lx} = A_I + A_{Lx}
\]

\[
A_I = A_I \otimes I_x + I_I \otimes A_{Ix} = A_I + A_{Ix}
\]

\[
A_R = A_I \otimes I_x + I_I \otimes A_{Rx} = A_I + A_{Rx}
\]

(54)
where

\[ A_t = D_t + P_t^{-1}E_{t0} \]

\[ A_{Lx} = D_x - \epsilon D_x^2 + P_x^{-1}E_{x0} - \epsilon P_x^{-1}E_{x0}D_x - \sigma_x^I P_x^{-1}E_{NN} - \sigma_x^V P_x^{-1}E_{NN} D_x \]

\[ A_{Ix} = D_x - \epsilon D_x^2 - \sigma_x^I P_x^{-1}E_{x0} - \sigma_x^V P_x^{-1}E_{x0}D_x - \sigma_x^I P_x^{-1}E_{NN} - \sigma_x^V P_x^{-1}E_{NN} D_x \]

\[ A_{Rx} = D_x - \epsilon D_x^2 + \epsilon P_x^{-1}E_{NN} D_x - \sigma_x^I P_x^{-1}E_{x0} - \sigma_x^I P_x^{-1}E_{NN} D_x . \]

The matrix \( A_t \) has been studied extensively, both numerically and theoretically, due to its common appearance in SBP-SAT based schemes. Its spectral properties are particularly relevant in connection to the invertibility of schemes using SBP-SAT based time integration. The following conjecture is needed.

**Conjecture 4.** The eigenvalues of the matrix \( A_t = D_t + P_t^{-1}E_{t0} \) have strictly positive real parts.

Conjecture 4 has been proved in the second order case [9, p. 490]. There is no direct proof in the general case, however a multitude of SBP operators used in practice have this property and, to our knowledge, no counter example has been found.

In order to prove that \( A_L, A_I, \) and \( A_R \) are invertible we will use Conjecture 4 together with the following lemma.

**Lemma 5.** Under the stability conditions (47), the eigenvalues of the matrices \( A_{kx}, k \in \{L, I, R\} \), have non-negative real parts.

**Proof.** A main implication of [1] is that the semi-discrete problem

\[
\begin{bmatrix}
  u \\
  v \\
  w
\end{bmatrix}_t +
\begin{bmatrix}
  A_{Lx} & R_x \\
  L_x & A_{Ix} & R_x \\
  L_x & A_{Rx}
\end{bmatrix}
\begin{bmatrix}
  u \\
  v \\
  w
\end{bmatrix} = 0
\]

is stable under the stability conditions (47). Equivalently

\[
\begin{bmatrix}
  u \\
  v \\
  w
\end{bmatrix}^T
\begin{bmatrix}
  P_xA_{Lx} & P_xR_x \\
  P_xA_{Ix} & P_xR_x \\
  P_xA_{Rx}
\end{bmatrix}
+ \begin{bmatrix}
  A_{Lx}^T P_x & L_x^T P_x \\
  R_x^T P_x & A_{Ix}^T P_x \\
  R_x^T P_x & A_{Rx}^T P_x
\end{bmatrix}
\begin{bmatrix}
  u \\
  v \\
  w
\end{bmatrix} \geq 0
\]

for all \( u, v, w \). It follows that \( P_xA_{kx} + A_{kx}^T P_x \) is positive semi-definite for \( k \in \{L, I, R\} \). To see this, assume for example that \( P_xA_{Lx} + A_{Lx}^T P_x \) is not positive semi-definite. Then there is a vector \( \bar{u} \) such that \( \bar{u}^T (P_xA_{Lx} + A_{Lx}^T P_x) \bar{u} < 0 \). But this contradicts (56) if we set \( u = \bar{u}, v = 0, \) and \( w = 0 \). The other cases follow in the same manner.

We now show that the positive semi-definiteness of \( P_xA_{kx} + A_{kx}^T P_x \) implies that the eigenvalues of \( A_{kx} \) have non-negative real parts. Note that if \((\lambda, z)\) is an eigenpair for \( A_{kx} \), then

\[ A_{kx}z = \lambda z \Rightarrow P_xA_{kx}z = \lambda P_xz \Rightarrow z^* P_xA_{kx}z = \lambda z^* P_xz . \]

By adding the conjugate transpose of the rightmost equation above it follows that

\[ z^* \left[ P_xA_{kx} + (P_xA_{kx})^T \right] z = 2\text{Re}(\lambda) z^* P_xz . \]

Therefore, if the matrix \( P_xA_{kx} + (P_xA_{kx})^T \) is positive semi-definite, then \( \lambda \) has non-negative real part. Hence, the eigenvalues of \( A_{kx}, k \in \{L, I, R\} \) have non-negative real parts. \( \Box \)
Combining Lemma 5 and Conjecture 4 we can prove the desired result.

**Proposition 6.** The stability conditions (47) imply that $A_L$, $A_I$, and $A_R$ are non-singular.

**Proof.** According to Conjecture 4, all eigenvalues of $A_t$ have strictly positive real parts. The eigenvalues of the Kronecker product $A_t \otimes I_x$ are products of the eigenvalues of $A_t$ and $I_x$. Therefore the eigenvalues of $A_t \otimes I_x$ have strictly positive real parts. Furthermore, since the matrices $A_t \otimes I_x$ and $I_t \otimes A_{kx}$ commute, the eigenvalues of the sum $A_k = A_t \otimes I_x + I_t \otimes A_{kx}$ are sums of the eigenvalues of $A_t \otimes I_x$ and $I_t \otimes A_{kx}$ [4, p. 117]. Hence, by Lemma 5, all eigenvalues of $A_k$ have strictly positive real parts. It follows that $A_L$, $A_I$, and $A_R$ are non-singular.

The invertibility of the multidomain system (33) is a similar consequence of Conjecture 4.

**Proposition 7.** The stability conditions (47) imply that the matrix

$$D = \begin{bmatrix} A_L & R_x \\ L_x & A_I & R_x \\ L_x & A_R \end{bmatrix}$$

is invertible.

**Proof.** Using (54) we have

$$D = \begin{bmatrix} A_t & A_t \\ A_t & A_t \end{bmatrix} + \begin{bmatrix} A_{Lx} & R_x \\ L_x & A_{Ix} & R_x \\ L_x & A_{Rx} \end{bmatrix} =: D_1 + D_2.$$

The matrix $D_2$ is the blockwise Kronecker product of $I_t$ and the matrix in (55). By the proof of Lemma 5, the eigenvalues of the matrix in (55) have non-negative real parts. It follows that the eigenvalues of $D_2$ have non-negative real parts [5, p. 178]. Furthermore, by Conjecture 4, the eigenvalues of $D_1$ have strictly positive real parts. Since $D_1$ and $D_2$ commute (this is a consequence of the fact that $A_t = A_t \otimes I_x$ commutes with any matrix of the form $I_t \otimes \cdot$), the eigenvalues of the sum $D_1 + D_2$ have strictly positive real parts and thus $D = D_1 + D_2$ is invertible.

Finally, the invertibility of the coupling matrix is a simple consequence of Proposition 6 and Proposition 7.

**Proposition 8.** The coupling matrix $\hat{C}$ in (46) is invertible.

**Proof.** By Proposition 6, the matrix

$$\begin{bmatrix} A_L & A_I \\ A_I & A_R \end{bmatrix}$$

is invertible. It follows from Proposition 7 that the matrix

$$C = \begin{bmatrix} I & A_{L}^{-1}R_x \\ A_{L}^{-1}L_x & I \\ A_{R}^{-1}L_x & I \end{bmatrix}$$

is invertible. But then, by construction of the coupling matrix $\hat{C}$ in (46), we know that the coupling system $\hat{C}\hat{u} = \hat{b}$ has at least one solution (namely the one we get
4.5. Space-time multidomain formulation. Extending the ideas in subsection 4.2 to a multidomain formulation in both space and time is fairly straightforward. We partition the space-time domain using the same discretization and SBP operators on each subdomain (see Figure 2). For the initial subdomains (the left-most column in Figure 2) the formulation is identical to the one in subsection 4.2. For the $m$:th subdomain column $B_{m1}, B_{m2}, \ldots, B_{mK}$ we get initial data from adjacent subdomains. Let $u_{mk}$ denote the numerical solution on the subdomain $B_{mk}$. The space-time multidomain scheme is

$$(D_t + D_x - \epsilon D_x^2)u^{m1} = F^{m1} - P_t^{-1}(E^{00}_t u^{m1} - E^{0M}_t u^{m-1,1})$$

$$- P_x^{-1}E^{00}_x (u^{m1} - \epsilon D_x u^{m1} - e_0^0 g_0)$$

$$+ \sigma_I P_x^{-1}(E^{NN}_x u^{m1} - e^{N0}_x u^{m2})$$

$$+ \sigma_R P_x^{-1}(E^{NN}_x D_x u^{m1} - e^{N0}_x D_x u^{m2})$$

$$:$$

$$(D_t + D_x - \epsilon D_x^2)u^{mk} = F^{mk} - P_t^{-1}(E^{00}_t u^{mk} - E^{0M}_t u^{m-1,k})$$

$$+ \sigma_I P_x^{-1}(E^{00}_x u^{mk} - E^{0N}_x u^{m,k-1})$$

$$+ \sigma_L P_x^{-1}(E^{NN}_x u^{mk} - E^{NN}_x u^{m,k-1})$$

$$+ \sigma_R P_x^{-1}(E^{NN}_x D_x u^{mk} - E^{N0}_x D_x u^{m,k+1})$$

$$+ \sigma_R P_x^{-1}(E^{NN}_x D_x u^{mk} - E^{N0}_x D_x u^{m,k+1})$$

$$:$$

$$(D_t + D_x - \epsilon D_x^2)u^{mK} = F^{mK} - P_t^{-1}(E^{00}_t u^{mk} - E^{0M}_t u^{m-1,K})$$

$$+ \sigma_I P_x^{-1}(E^{00}_x u^{mk} - E^{0N}_x u^{m,K-1})$$

$$+ \sigma_L P_x^{-1}(E^{NN}_x u^{mk} - E^{NN}_x u^{m,K-1})$$

$$- P_x^{-1}E^{NN}_x (\epsilon D_x u^{mK} - e^{N1}_x g_1)$$

The simplest way to take advantage of this is to wait until we have computed $u^{m-1,k}$ (the initial solutions $u^{1k}$ for $k = 1, 2, \ldots, K$ are computed exactly as in subsection 4.2. Then, since $u^{m-1,k}$ is known, the problem of solving for $u^{mk}$, $k = 1, 2, \ldots, K$ is identical to the problem in subsection 4.2. That is, for each column of subdomains in Figure 2 we solve the coupling system (46) with a right-hand side depending on the solution on the previous subdomain column. This is the method we used in section 5.
5. Numerical experiments. To test the efficiency of the space-time multidomain scheme we solve the advection–diffusion equation (19), with $\epsilon = 0.01$ and $T = L = 1$, using data from the manufactured solution

\begin{equation}
  u(t, x) = \frac{1 - \exp\left(\frac{x - 1}{\epsilon}\right)}{1 - \exp\left(-\frac{1}{\epsilon}\right)} + \exp(-3t)\sin(8\pi(x - t)).
\end{equation}

5.1. Multidomain vs single domain SBP time integration. We begin with a small comparative study of the single domain method against the space-time multidomain method. Note that this is a comparison based on very straightforward Matlab implementations. The bulk of the computational work – solving the full system (20) in the single domain case, and solving the coupling system (46) in the multidomain case – is done using an UMFPACK-solver selected by the Matlab routine \texttt{mldivide}. For all computations done with the space-time multidomain method we use penalty coefficients $\sigma_{V_R} = 0$ and $\sigma_{V_L} = 1$. The coefficient $\sigma_{I_R}$ is chosen depending on the grid size, such that (47) is satisfied.

To compare our implementations we fix the total number of gridpoints in time to 256 and compute solutions on increasingly fine spatial grids using (4,2)-accurate SBP operators [11, p. 7] in both time and space. For the multidomain method, each spacetime block (see Figure 2) is of size 32-by-32 (i.e., the temporal domain is split into $\frac{256}{32} = 8$ subdomains, while the number of spatial subdomains depends on the spatial refinement level). Note that this choice of block size is not necessarily optimal since selecting optimal discretization parameters (number of subdomains and gridpoints per subdomain) is beyond the scope of this article. Our choices for the purposes of these experiments are ad hoc and should be viewed as a proof of concept.

Our multidomain implementation outperforms our single domain implementation both in terms of execution time and accuracy (see Figure 3 and Figure 4). Additionally, the amount of memory used by the UMFPACK-solver is significantly reduced in the multidomain implementation (see Figure 5), allowing us to compute solutions on finer grids without running out of memory.

Fig. 2: An illustration of the space-time block structure
Fig. 3: The $L^2$-errors of numerical solutions at $T = 1$, plotted as functions of the total number of spatial gridpoints. Dashed line: Single domain scheme. Solid line: Multidomain scheme.

Fig. 4: Execution times of single domain and multidomain implementations, plotted as functions of the total number of spatial gridpoints.

5.2. Multidomain vs Runge-Kutta time integration. Next we compare our implementation of the multidomain scheme to a Runge-Kutta based solver. We use the Matlab function \texttt{ode45} for time integration and a (4, 2)-accurate SBP operator together with SATs for spatial discretization. The multidomain method is set up as in subsection 5.1. We compute solutions for increasingly fine spatial grids and compare errors and execution times. Our implementation of the multidomain scheme consistently outperforms our Runge-Kutta based implementation both in terms of execution time and accuracy. The results can be seen in Figure 6 and Figure 7. As we refine the spatial grid, the Runge-Kutta solver requires much smaller timesteps than the multidomain solver in order to remain stable. In this case the execution time of the Runge-Kutta solver grows superlinearly with the number of grid points, while the execution time of the multidomain solver enjoys linear growth throughout the investigated grid sizes.
Fig. 5: Peak memory usage reported by the Matlab function `mldivide` in our single domain implementation versus our multidomain implementation.

Fig. 6: The $L^2$-errors of numerical solutions at $T = 1$, plotted as functions of the total number of spatial gridpoints. Dashed line: Runge-Kutta scheme. Solid line: Multidomain scheme.

Fig. 7: Execution times of Runge-Kutta and multidomain implementations, plotted as functions of the total number of spatial gridpoints.
6. Conclusions. We have formulated a fully discrete and provably stable multidomain scheme for the one-dimensional advection–diffusion equation using SBP-SAT in space and time. The structure of the interface SATs made it possible to isolate a linear system involving only solution components which are used in the subdomain couplings. The full solution could then be formed as a linear combination of a certain vector set, where the coefficients depend on the coupling components.

The stability of the multidomain scheme was used, together with a common assumption about the spectral properties of SBP operators, to prove invertibility of both the full system and the coupling system.

Significant computational efficiency was gained compared to solving the full system directly, and compared to an explicit Runge-Kutta based solver.

The fact that we are able to prove both stability and invertibility of our multidomain scheme suggests that the SBP-SAT framework can be successfully combined with domain decomposition methods to produce schemes that retain desirable theoretical properties while at the same time mitigating the computational inefficiencies commonly associated with SBP in time.

The ideas presented here provide a path for further research into the connection between SBP-SAT based discretizations and domain decomposition. In particular we will investigate a similar procedure in higher spatial dimensions, and if the procedure extends to more complicated settings like the Navier–Stokes equations.

REFERENCES