LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-JRNL-865529

Stage-Local Partitioned Two-Step Runge-Kutta Methods for Large Systemsof Ordinary Differential Equations

P. Tranquilli, T. Haut, L. Ricketson, J. Hittinger

June 12, 2024

BIT Numerical Mathematics

Disclaimer

BIT manuscript No.
(will be inserted by the editor)

Stage-Local Partitioned Two-Step Runge-Kutta Methods for Large Systems of Ordinary Differential Equations

Paul Tranquilli · Lee Ricketson · Terry Haut · Jeffrey Hittinger

Received: date / Accepted: date

Abstract We introduce stage-local partitioned two-step Runge-Kutta methods are an extension of standard two-step Runge-Kutta methods, which are an alternative to the standard additive two-step Runge-Kutta methods currently existing in the literature. These new schemes are designed with an eye towards truly N-partitioned systems and leverage local stage approximations to make several computationally interesting approximations viable. Specifically, the focus on local stage approximations makes possible the construction of truly asynchronous schemes, in the parallel sense, possible. In addition, we show that an implicit-explicit approach to these schemes can lead to methods that require the inversion of only local nonlinear systems.

Keywords Two-Step Runge-Kutta · Time Integration · Asynchronous

1 Introduction

We consider the numerical solution of large initial value problems (IVPs) of the form

$$y' = f(y), \quad y(t_0) = y_0 \in \mathbb{R}^d, \quad f : \mathbb{R}^d \mapsto \mathbb{R}^d, \quad t_0 \leq t \leq t_f. \tag{1.1}$$

Center for Applied Scientific Computing
Lawrence Livermore National Laboratory
7000 East Avenue
Livermore, CA 94550

P. Tranquilli
E-mail: tranquilli1@llnl.gov

L. Ricketson
E-mail: ricketson1@llnl.gov

T. Haut
E-mail: haut3@llnl.gov

J. Hittinger
E-mail: hittinger1@llnl.gov

Such systems arise from the spatial discretization of important time-dependent physical systems that describe processes such as the dynamics of the atmosphere and oceans, the behavior of materials, or the evolution of plasmas. As the scope and magnitude of these simulations grow, so too does the complexity of the computational resources required for their solution.

Modern algorithms for solving such large-scale systems are highly distributed, making data communication costs a significant factor in time-to-solution. This situation is expected to become more extreme in next-generation supercomputers. Even in the presence of perfect load balancing, the latency and unpredictability of communication can reduce parallel efficiency. As a result, there is increasing interest in finding ways to eliminate the need for bulk synchronous communication, or to hide the communication cost by overlapping productive computation with the communication.

It is natural to consider using data from a previous time-step when necessary data at the current time-step has not yet arrived, thus reducing synchrony and masking communication overhead. Doing this naively has accuracy and stability consequences, so most previous approaches have explored the modification of spatial stencils to accommodate time-lagged data [4, 1]. The derivation of such stencils has thus far been limited to relatively simple PDEs (typically the heat equation) [10], and their behavior has been studied with low-order, explicit time-stepping methods (typically forward Euler)[1]. Future HPC architectures favor high-order methods, however.

In contrast, we propose to build into the time integration method itself tolerance for the time-lagged data. In order to construct reasonable approximations with potentially time-lagged information, we look to a class of methods that spans across multiple time-steps, specifically, Two-Step Runge-Kutta (TSRK) methods [7]. Additionally, the ability to time-lag only some data requires the use of an additive type TSRK scheme. We present a new class of TSRK schemes based on an N-partitioning of the right-hand side f, according to the parallel distribution of the right-hand side computation.

Consider an ODE whose state data is distributed across N processing elements (PEs). We refer to the data owned by a given PE as *local*, while the data owned by other PEs is *non-local*. In the context of PDEs, these two data sets are often called "valid data" and "ghost data", respectively. We propose an additive TSRK formulation in which distinct TSRK methods are used to construct local approximations for the valid and ghost data on each PE. The resulting scheme is a stage-local partitioned two-step Runge-Kutta method (SLP-TSRK). We provide a general order condition framework for schemes of arbitrary order, and show that preconsistency is the only required coupling condition of the distinct TSRK schemes. This leniency in coupling condition allows for relatively straightforward development of new schemes.

There are two natural sub-families of SLP-TSRK methods. In the first, the local TSRK method is explicit: computation of stage s requires knowledge of stages $\{1,\dots,s-1\}$, while the non-local methods computation of stage s is *super-explicit*, requiring only stages $\{1,\dots,s-2\}$. A given PE that knows its valid data up to stage $s-1$ and its ghost data up to $s-2$ may thus perform the computation of its local stage approximation simultaneously with its receipt of stage-$(s-1)$ ghost data. This construction enables overlapping at least some communication with computation. We will refer to such a time-stepping scheme as *asynchronous*.

In the second sub-family, the local method is implicit, while the non-local method is (super-)explicit. An iterative solver may thus be used to update valid data, but communication is only required **once per stage**, rather than once per iteration. This version is referred to as *locally implicit*. There is tremendous potential for such a method to mask communication costs behind non-linear, implicit solves that would be performed entirely locally. This local-only inversion will inevitably impact the stability of the overall method; however, lightly implicit methods, that leverage inexact approximations of the (non)linear system solutions, have been shown to be a valuable approach to a wide range of problems [15, 12, 3].

Of course, stability is a concern in both sub-families. One cannot expect the asynchronous scheme to have the same stability region as a well-optimized, synchronous explicit scheme, just as a locally implicit scheme cannot match the stability of a fully implicit scheme. We employ the framework of [8, 9, 13] to understand stability of the new methods in simple contexts, and show that efficient schemes can be constructed to balance the tradeoffs of step size versus computational cost.

We structure our presentation as follows. We begin in Section 2 by reviewing standard TSRK methods and then define the new class of SLP-TSRK methods and construct an order condition theory. Next, in Section 3, we describe in more detail the two sub-families of SLP-TSRK introduced in Section 2. In Section 4, we describe a framework for studying stability and use it to derive practical SLP-TSRK methods with favorable stability properties. We verify and explore the properties of these new schemes on an advection-diffusion system in Section 6. Finally, in Section 7 we conclude and discuss directions for future work.

2 Two-Step Runge-Kutta Methods

A standard two-step Runge-Kutta method computes a numerical approximation to the solution of (1.1) on a uniform grid $t_n = t_{n-1} + h$ as

$$\begin{aligned} Y_i^{[n]} &= (1-u_i)\, y_{n-1} + u_i y_{n-2} \\ &\quad + h \sum_{j=1}^{s} \left(a_{i,j} K_j^{[n]} + b_{i,j} K_j^{[n-1]} \right), \quad i = 1, \ldots, s, && \text{(2.1a)} \\ K_j^{[n]} &= f(Y_j^{[n]}) && \text{(2.1b)} \\ y_n &= (1-\theta)\, y_{n-1} + \theta y_{n-2} + \\ &\quad h \sum_{j=1}^{s} \left(v_j K_j^{[n]} + w_j K_j^{[n-1]} \right). && \text{(2.1c)} \end{aligned}$$

where i is the stage number, s the number of stages, n the step number, and the specific TSRK scheme is defined by the method coefficients u_i, a_{ij}, b_{ij}, θ, v_j, and w_j. These

coefficients are generally presented in a Butcher tableau as

$$\begin{array}{c|c|c} \mathbf{u} & \mathbf{A} & \mathbf{B} \\ \hline \theta & \mathbf{v}^T & \mathbf{w}^T \end{array} = \begin{array}{c|ccccc|cccc} u_1 & a_{1,1} & & & & & b_{1,1} & b_{1,2} & \cdots & b_{1,s} \\ u_2 & a_{2,1} & a_{2,2} & & & & b_{2,1} & b_{2,2} & \cdots & b_{1,s} \\ \vdots & \vdots & \ddots & \ddots & & & \vdots & \vdots & & \vdots \\ u_s & a_{s,1} & \cdots & a_{s,s-1} & a_{s,s} & & b_{s,1} & b_{s,2} & \cdots & b_{s,s} \\ \hline \theta & v_1 & v_2 & \cdots & v_s & & w_1 & w_2 & \cdots & w_s \end{array}. \tag{2.2}$$

An explicit TSRK will have a zero upper triangle of $\mathbf{A}$, $a_{ij} = 0 \ \forall j \geq i$, while it is standard for diagonally implicit methods to have a single value along the diagonal, $a_{ii} = \lambda$, $a_{ij} = 0 \ \forall j > i$. Finally, to obtain asynchrony we employ the notion of a *super-explicit* method in which both the main diagonal and first sub-diagonal are zero, $a_{ii} = 0$ and $a_{i,i-1} = 0$. In addition to these coefficients, which appear explicitly in the TSRK formula, it is convenient to include the abscissa vector, $\mathbf{c} = (\mathbf{A} + \mathbf{B})\,\mathbf{e} - \mathbf{u}$, that describes the times where stage approximations $Y_i^{[n]} \approx y(t_n + c_i h)$ are computed. Here, and throughout this manuscript, we use the convention that $\mathbf{e} = [1, 1, \cdots, 1]^T$ and its dimension is determined by context.

A TSRK method (2.1) is said to have stage order q if the stage vectors $Y_i^{[n]}$ are q-order approximations of the true solution at $t_n + c_i h$,

$$Y_i^{[n]} = y(t_{n-1} + c_i h) + \mathscr{O}(h^{q+1}), \tag{2.3}$$

as $h \to 0$. Similarly, the TSRK method is of global order p if the final approximation y_{n_f}, with $n_f = t_f / h$, is of order p, that is to say that

$$y_{n_f} = y(t_f) + \mathscr{O}(h^p) \tag{2.4}$$

as $h \to 0$.

Order conditions for these methods are well known and are given in [7] when $q \geq p - 1$. These conditions are split into two pieces, the so-called "stage" conditions,

$$\frac{\mathbf{c}^\nu}{\nu!} - \frac{(-1)^\nu}{\nu!}\mathbf{u} - \frac{\mathbf{A}\mathbf{c}^{\nu-1}}{(\nu-1)!} - \frac{\mathbf{B}\,(\mathbf{c}-\mathbf{e})^{\nu-1}}{(\nu-1)!} = 0, \tag{2.5}$$

and "step" conditions,

$$\frac{1}{\nu!} - \frac{(-1)^\nu}{\nu!}\theta - \frac{\mathbf{v}^T\mathbf{c}^{\nu-1}}{(\nu-1)!} - \frac{\mathbf{w}^T\,(\mathbf{c}-\mathbf{e})^{\nu-1}}{(\nu-1)!} = 0, \tag{2.6}$$

where the exponentiation is applied component-wise. The method (2.1) is of order p iff (2.5) is satisfied for all $\nu \leq q$ and iff (2.6) is satisfied for all $\nu \leq p$ with $q \geq p - 1$.

2.1 Stage-Local Partitioned TSRK

We consider methods for solving (1.1) via the component N-partitioning,

$$\frac{dy^{\{m\}}}{dt} = f^{\{m\}}(y^{\{1\}},\ldots,y^{\{m\}},\ldots y^{\{N\}}), \tag{2.7}$$

$$y(t_0) = y_0, \quad t_0 \le t \le t_f, \quad m = 1,\ldots,N, \tag{2.8}$$

with $y^{\{m\}} \in \mathbb{R}^{d_m}$, $f^{\{m\}} : \mathbb{R}^d \mapsto \mathbb{R}^{d_m}$, and $\sum d_m = d$. Without loss of generality and to simplify the presentation, we assume throughout the rest of our discussion that each $y^{\{m\}}$ and $f^{\{m\}}$ are scalar, that is to say that $d_m = 1$ for all $m \in \{1,\ldots,N\}$.

We advance the numerical solution on a uniform grid according to

$$\begin{aligned} Y_i^{\{m,\ell\}} &= \left(1-u_i^{\{m,\ell\}}\right) y_{n-1}^{\{\ell\}} + u_i^{\{m,\ell\}} y_{n-2}^{\{\ell\}} + h\sum_{j=1}^{s} a_{i,j}^{\{m,\ell\}} K_j^{\{\ell\}[n]} \\ &\quad + h\sum_{j=1}^{s} b_{i,j}^{\{m,\ell\}} K_j^{\{\ell\}[n-1]}, \quad l = 1,\ldots,N, \end{aligned} \tag{2.9a}$$

$$K_j^{\{m\}[n]} = f^{\{m\}}\left(Y_j^{\{m,1\}}, Y_j^{\{m,2\}},\ldots,Y_j^{\{m,m\}},\ldots,Y_j^{\{m,N\}}\right), \tag{2.9b}$$

$$\begin{aligned} y_n^{\{m\}} &= \left(1-\theta^{\{m\}}\right) y_{n-1}^{\{m\}} + \theta^{\{m\}} y_{n-2}^{\{m\}} + h\sum_{j=1}^{s} v_j^{\{m\}} K_j^{\{m\}[n]} \\ &\quad + h\sum_{j=1}^{s} w_j^{\{m\}} K_j^{\{m\}[n-1]}, \quad m = 1,\ldots,N, \end{aligned} \tag{2.9c}$$

where $Y_i^{\{m,\ell\}}$ is the m-th partition's local approximation of the l-th partition's ghost-stage solution at time $t_n + c_i h$. In other words, $Y_i^{\{m,\ell\}} \approx y^{\{\ell\}}(t_n + c_i h)$ and is used exclusively in the construction of the m-th partition's local solution. Similarly, the valid stage solution at time $t_n + c_i h$ is approximated by the vector $Y_i^{\{m,m\}} \approx y^{\{m\}}(t_n + c_i h)$. Readers seeking a more explicit expression may refer to the unrolling of the form of the method, applied to a linear problem with $N = 2$, given in Equation 4.6. A SLP-TSRK method can be represented by the general Butcher tableau,

$$\begin{array}{cccc|cccc|cccc} \mathbf{u}^{\{1,1\}} & \mathbf{u}^{\{1,2\}} & \ldots & \mathbf{u}^{\{1,N\}} & \mathbf{A}^{\{1,1\}} & \mathbf{A}^{\{1,2\}} & \ldots & \mathbf{A}^{\{1,N\}} & \mathbf{B}^{\{1,1\}} & \mathbf{B}^{\{1,2\}} & \ldots & \mathbf{B}^{\{1,N\}} \\ \mathbf{u}^{\{2,1\}} & \mathbf{u}^{\{2,2\}} & \ldots & \mathbf{u}^{\{2,N\}} & \mathbf{A}^{\{2,1\}} & \mathbf{A}^{\{2,2\}} & \ldots & \mathbf{A}^{\{2,N\}} & \mathbf{B}^{\{2,1\}} & \mathbf{B}^{\{2,2\}} & \ldots & \mathbf{B}^{\{2,N\}} \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ \mathbf{u}^{\{N,1\}} & \mathbf{u}^{\{N,2\}} & \ldots & \mathbf{u}^{\{N,N\}} & \mathbf{A}^{\{N,1\}} & \mathbf{A}^{\{N,2\}} & \ldots & \mathbf{A}^{\{N,N\}} & \mathbf{B}^{\{N,1\}} & \mathbf{B}^{\{N,2\}} & \ldots & \mathbf{B}^{\{N,N\}} \\ \hline \theta^{\{1\}} & \theta^{\{2\}} & \ldots & \theta^{\{N\}} & \mathbf{v}^{\{1\}} & \mathbf{v}^{\{2\}} & \ldots & \mathbf{v}^{\{N\}} & \mathbf{w}^{\{1\}} & \mathbf{w}^{\{2\}} & \ldots & \mathbf{w}^{\{N\}} \end{array}. \tag{2.10}$$

Here and in the proofs that follow, each method $\left\{\mathbf{u}^{\{m,\ell\}}, \mathbf{A}^{\{m,\ell\}}, \mathbf{B}^{\{m,\ell\}}, \theta^{\{m\}}, \mathbf{v}^{\{m\}}, \mathbf{w}^{\{m\}}\right\}$, is considered to be entirely distinct; however, throughout our discussion, we will focus on the case where the partitioning of (1.1) into (2.7) corresponds to the problem's parallel distribution across some distributed computation environment. In this case, we generally only need to distinguish between the *diagonal method* that computes a

local approximation of valid data, with $\left\{\mathbf{u}^{\{m,m\}},\mathbf{A}^{\{m,m\}},\mathbf{B}^{\{m,m\}},\theta^{\{m\}},\mathbf{v}^{\{m\}},\mathbf{w}^{\{m\}}\right\}$, and the *off-diagonal method* that computes the local approximation of the ghost stage solutions, with $\left\{\mathbf{u}^{\{m,\ell\}},\mathbf{A}^{\{m,\ell\}},\mathbf{B}^{\{m,\ell\}}\right\}$ and $m \neq l$. Such a method would have a Butcher tableau of the form

$$\begin{array}{cccc|cccc|cccc}
\mathbf{u}^{\{*\}} & \mathbf{u}^{\{\dagger\}} & \dots & \mathbf{u}^{\{\dagger\}} & \mathbf{A}^{\{*\}} & \mathbf{A}^{\{\dagger\}} & \dots & \mathbf{A}^{\{\dagger\}} & \mathbf{B}^{\{*\}} & \mathbf{B}^{\{\dagger\}} & \dots & \mathbf{B}^{\{\dagger\}} \\
\mathbf{u}^{\{\dagger\}} & \mathbf{u}^{\{*\}} & \dots & \mathbf{u}^{\{\dagger\}} & \mathbf{A}^{\{\dagger\}} & \mathbf{A}^{\{*\}} & \dots & \mathbf{A}^{\{\dagger\}} & \mathbf{B}^{\{\dagger\}} & \mathbf{B}^{\{*\}} & \dots & \mathbf{B}^{\{\dagger\}} \\
\vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\
\mathbf{u}^{\{\dagger\}} & \mathbf{u}^{\{\dagger\}} & \dots & \mathbf{u}^{\{*\}} & \mathbf{A}^{\{\dagger\}} & \mathbf{A}^{\{\dagger\}} & \dots & \mathbf{A}^{\{*\}} & \mathbf{B}^{\{\dagger\}} & \mathbf{B}^{\{\dagger\}} & \dots & \mathbf{B}^{\{*\}} \\
\hline
\theta^{\{*\}} & \theta^{\{*\}} & \dots & \theta^{\{*\}} & \mathbf{v}^{\{*\}} & \mathbf{v}^{\{*\}} & \dots & \mathbf{v}^{\{*\}} & \mathbf{w}^{\{*\}} & \mathbf{w}^{\{*\}} & \dots & \mathbf{w}^{\{*\}}
\end{array}, \tag{2.11}$$

2.2 Order Conditions of SLP-TSRK

To obtain a p-th order SLP-TSRK each sub-method in the tableau (2.11) must satisfy independently the stage conditions (2.5) up to order $q \geq p-1$, the step conditions (2.6) up to order p, and all of the methods must be consistent, that is to say that $\mathbf{c}^{\{m,\ell\}} = \mathbf{c}$ for all $m,l \in \{1,\dots,N\}$.

Theorem 2.1 *Consider the class of Stage-Local Partitioned Two-Step Runge-Kutta methods* (2.9)*. If*

- *each individual method is zero stable,*
- *the starting values are of order p,*
- *each individual diagonal method,* $\left\{\mathbf{u}^{\{m,m\}},\mathbf{A}^{\{m,m\}},\mathbf{B}^{\{m,m\}},\theta^{\{m\}},\mathbf{v}^{\{m\}},\mathbf{w}^{\{m\}}\right\}$*, is of order p,*
- *each individual method,* $\left\{\mathbf{u}^{\{m,\ell\}},\mathbf{A}^{\{m,\ell\}},\mathbf{B}^{\{m,\ell\}}\right\}$*, is of stage order* $q \geq p-1$*,*
- *and, each method is consistent with every other:* $\mathbf{c}^{\{m,\ell\}} = \mathbf{c}$ *for all* $m,l \in \{1,\dots,N\}$*,*

then the numerical solution converges with order p,

$$y_n - y(t_n) = \mathscr{O}(h^p), \forall n.$$

Proof We follow the strategy in [7] and [16]. Specifically, we will define some solution super vector Z_n, and then write the method (2.9) as a so-called A-method

$$Z_n = \mathscr{A} Z_{n-1} + h\mathscr{B} F(Z_n). \tag{2.12}$$

First, we consider the following ordering and super-vectors

$$Y^{[n]} = \begin{bmatrix} Y^{\{1,1\}[n]} \\ \vdots \\ Y^{\{1,N\}[n]} \\ Y^{\{2,1\}[n]} \\ \vdots \\ Y^{\{N,N\}[n]} \end{bmatrix}, \quad Y^{\{m,\ell\}[n]} = \begin{bmatrix} Y_1^{\{m,\ell\}[n]} \\ \vdots \\ Y_s^{\{m,\ell\}[n]} \end{bmatrix} \quad y_n = \begin{bmatrix} y_n^{\{1\}} \\ \vdots \\ y_n^{\{N\}} \end{bmatrix}, \quad \text{and} \quad Z_n = \begin{bmatrix} Y^{[n-1]} \\ Y^{[n]} \\ y_{n-1} \\ y_n \end{bmatrix}, \tag{2.13}$$

with $Y^{[n]} \in \mathbb{R}^{(N^2\cdot s)\times 1}$, $Y^{\{m,\ell\}[n]} \in \mathbb{R}^{s\times 1}$, $y_n \in \mathbb{R}^{N\times 1}$, and $Z_n \in \mathbb{R}^{(2\cdot N^2\cdot s+2\cdot N)\times 1}$. We further define the matrices $U^{\{m\}}, I_{N\times N}\otimes \mathbf{e}_{s\times 1} \in \mathbb{R}^{(N\cdot s)\times N}$

$$U^{\{m\}} = \begin{bmatrix} \mathbf{u}^{\{m,1\}} & \mathbf{0} & \cdots \\ \mathbf{0} & \ddots & \vdots \\ \vdots & \cdots & \mathbf{u}^{\{m,N\}} \end{bmatrix}, \quad \text{and} \quad I_{N\times N}\otimes \mathbf{e}_{s\times 1} = \begin{bmatrix} \mathbf{e} & \mathbf{0} & \cdots \\ \mathbf{0} & \ddots & \vdots \\ \vdots & \cdots & \mathbf{e} \end{bmatrix},$$

as well as the super-matrices $U, E \in \mathbb{R}^{(N^2\cdot s)\times N}$

$$U = \begin{bmatrix} U^{\{1\}} \\ U^{\{2\}} \\ \vdots \\ U^{\{N\}} \end{bmatrix}, \quad E = \begin{bmatrix} I_{N\times N}\otimes \mathbf{e}_{s\times 1} \\ I_{N\times N}\otimes \mathbf{e}_{s\times 1} \\ \vdots \\ I_{N\times N}\otimes \mathbf{e}_{s\times 1} \end{bmatrix},$$

and the structurally similar $\Theta \in \mathbb{R}^{N\times N}$

$$\Theta = \begin{bmatrix} \theta^{\{1\}} & 0 & \cdots \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \theta^{\{N\}} \end{bmatrix}.$$

Then we have that the matrix

$$\mathscr{A} = \begin{bmatrix} \mathbf{0}_{(N^2\cdot s)\times(N^2\cdot s)} & I_{(N^2\cdot s)\times(N^2\cdot s)} & \mathbf{0}_{(N^2\cdot s)\times N} & \mathbf{0}_{(N^2\cdot s)\times N} \\ \mathbf{0}_{(N^2\cdot s)\times(N^2\cdot s)} & \mathbf{0}_{(N^2\cdot s)\times(N^2\cdot s)} & U & E-U \\ \mathbf{0}_{N\times(N^2\cdot s)} & \mathbf{0}_{N\times(N^2\cdot s)} & \mathbf{0}_{N\times N} & I_{N\times N} \\ \mathbf{0}_{N\times(N^2\cdot s)} & \mathbf{0}_{N\times(N^2\cdot s)} & \Theta & I_{N\times N}-\Theta \end{bmatrix}.$$

We can, similarly to (2.13), define the super-vectors $F\left(Y^{[n]}\right) \in \mathbb{R}^{(N\cdot s)\times 1}$, $F^{\{m\}}\left(Y^{[n]}\right) \in \mathbb{R}^{s\times 1}$, and $F(Z_n) \in \mathbb{R}^{(2\cdot N\cdot s+2\cdot N)\times 1}$ as

$$F\left(Y^{[n]}\right) = \begin{bmatrix} F^{\{1\}}\left(Y^{[n]}\right) \\ \vdots \\ F^{\{N\}}\left(Y^{[n]}\right) \end{bmatrix}, \quad F^{\{m\}}\left(Y^{[n]}\right) = \begin{bmatrix} f^{\{m\}}\left(Y_1^{\{m,1\}[n]},\cdots,Y_1^{\{m,N\}[n]}\right) \\ \vdots \\ f^{\{m\}}\left(Y_s^{\{m,1\}[n]},\cdots,Y_s^{\{m,N\}[n]}\right) \end{bmatrix}, \tag{2.14}$$

and

$$F(Z_n) = \begin{bmatrix} F\left(Y^{[n-1]}\right) \\ F\left(Y^{[n]}\right) \\ f(y_{n-1}) \\ f(y_n) \end{bmatrix}.$$

We now define the matrices $\mathbb{A}^{\{m\}}, \mathbb{B}^{\{m\}} \in \mathbb{R}^{(N\cdot s)\times(N\cdot s)}$

$$\mathbb{A}^{\{m\}} = \begin{bmatrix} \mathbf{A}^{\{m,1\}} & \mathbf{0}_{s\times s} & \cdots \\ \mathbf{0}_{s\times s} & \ddots & \vdots \\ \vdots & \cdots & \mathbf{A}^{\{m,N\}} \end{bmatrix}, \quad \mathbb{B}^{\{m\}} = \begin{bmatrix} \mathbf{B}^{\{m,1\}} & \mathbf{0}_{s\times s} & \cdots \\ \mathbf{0}_{s\times s} & \ddots & \vdots \\ \vdots & \cdots & \mathbf{B}^{\{m,N\}} \end{bmatrix},$$

and $\mathbb{A}, \mathbb{B} \in \mathbb{R}^{(N^2 \cdot s)\times(N\cdot s)}$

$$\mathbb{A} = \begin{bmatrix} \mathbb{A}^{\{1\}} \\ \vdots \\ \mathbb{A}^{\{N\}} \end{bmatrix}, \quad \mathbb{B} = \begin{bmatrix} \mathbb{B}^{\{1\}} \\ \vdots \\ \mathbb{B}^{\{N\}} \end{bmatrix}.$$

In addition, we construct the structurally similar matrices $\mathbb{V}, \mathbb{W} \in \mathbb{R}^{N\times(N\cdot s)}$:

$$\mathbb{V} = \begin{bmatrix} \mathbf{v}^{\{1\}} & \mathbf{0}_{1\times s} & \cdots \\ \mathbf{0}_{1\times s} & \ddots & \vdots \\ \vdots & \cdots & \mathbf{v}^{\{N\}} \end{bmatrix}, \quad \mathbb{W} = \begin{bmatrix} \mathbf{w}^{\{1\}} & \mathbf{0}_{1\times s} & \cdots \\ \mathbf{0}_{1\times s} & \ddots & \vdots \\ \vdots & \cdots & \mathbf{w}^{\{N\}} \end{bmatrix}.$$

Then, the matrix $\mathscr{B}$ is given by

$$\mathscr{B} = \begin{bmatrix} \mathbf{0}_{(N^2\cdot s)\times(N\cdot s)} & \mathbf{0}_{(N^2\cdot s)\times(N\cdot s)} & \mathbf{0}_{(N^2\cdot s)\times N} & \mathbf{0}_{(N^2\cdot s)\times N} \\ \mathbb{B} & \mathbb{A} & \mathbf{0}_{(N^2\cdot s)\times N} & \mathbf{0}_{(N^2\cdot s)\times N} \\ \mathbf{0}_{N\times(N\cdot s)} & \mathbf{0}_{N\times(N\cdot s)} & \mathbf{0}_{N\times N} & \mathbf{0}_{N\times N} \\ \mathbb{W} & \mathbb{V} & \mathbf{0}_{N\times N} & \mathbf{0}_{N\times N} \end{bmatrix}$$

Note that with these definitions of Z_n, $\mathscr{A}$, $\mathscr{B}$, and F, the method has the form of (2.12) and is thus an A-method.

We now construct the supervectors for the exact solution. Specifically, we construct the vectors $y^{\{k\}}(t_{n-1}+\mathbf{c}h) \in \mathbb{R}^{s\times 1}$, and $y(t_{n-1}+\mathbf{c}h) \in \mathbb{R}^{(N^2\cdot s)\times 1}$, that contains each $y^{\{k\}}$ N times:

$$y^{\{k\}}(t_{n-1}+\mathbf{c}h) = \begin{bmatrix} y^{\{k\}}(t_{n-1}+c_1 h) \\ \vdots \\ y^{\{k\}}(t_{n-1}+c_s h) \end{bmatrix}, \quad y(t_{n-1}+\mathbf{c}h) = \begin{bmatrix} y^{\{1\}}(t_{n-1}+\mathbf{c}h) \\ \vdots \\ y^{\{N\}}(t_{n-1}+\mathbf{c}h) \\ \vdots \\ \vdots \\ y^{\{1\}}(t_{n-1}+\mathbf{c}h) \\ \vdots \\ y^{\{N\}}(t_{n-1}+\mathbf{c}h) \end{bmatrix},$$

and $F^{\{k\}}(y(t_{n-1}+\mathbf{c}h)) \in \mathbb{R}^{s\times 1}$, and $F(y(t_{n-1}+\mathbf{c}h)) \in \mathbb{R}^{N\cdot s\times 1}$

$$F^{\{k\}}(y(t_{n-1}+\mathbf{c}h)) = \begin{bmatrix} f^{\{k\}}(y(t_{n-1}+c_1 h)) \\ \vdots \\ f^{\{k\}}(y(t_{n-1}+c_s h)) \end{bmatrix}, \quad F(y(t_{n-1}+\mathbf{c}h)) = \begin{bmatrix} F^{\{1\}}(y(t_{n-1}+\mathbf{c}h)) \\ \vdots \\ F^{\{N\}}(y(t_{n-1}+\mathbf{c}h)) \end{bmatrix},$$

as well as $z(t_n), F(z(t_n)) \in \mathbb{R}^{2\cdot N\cdot s+2\cdot N\times 1}$

$$z(t_1) = \begin{bmatrix} \mathbf{0} \\ y(t_0+\mathbf{c}h) \\ y(t_0) \\ y(t_1) \end{bmatrix}, \quad z(t_n) = \begin{bmatrix} y(t_{n-2}+\mathbf{c}h) \\ y(t_{n-1}+\mathbf{c}h) \\ y(t_{n-1}) \\ y(t_n) \end{bmatrix}, \quad F(z(t_n)) = \begin{bmatrix} F(y(t_{n-2}+\mathbf{c}h)) \\ F(y(t_{n-1}+\mathbf{c}h)) \\ F(y(t_{n-1})) \\ F(y(t_n)) \end{bmatrix}.$$

The local discretization error $h\mathbf{d}(t_n)$ is the residual obtained when the numerical approximation Z_n is replaced by the exact solution $z(t_n)$ in (2.12).

$$z(t_n) = \mathscr{A} z(t_{n-1}) + h\mathscr{B} F(z(t_n)) + h\mathbf{d}(t_n). \tag{2.15}$$

We partition $h\mathbf{d}$ into stage and full step residuals as follows:

$$h\mathbf{d}(t_n) = h \begin{bmatrix} \mathbf{d}^{(1)}_{\text{stage}}(t_n) \\ \mathbf{d}^{(2)}_{\text{stage}}(t_n) \\ \mathbf{d}^{(1)}_{\text{step}}(t_n) \\ \mathbf{d}^{(2)}_{\text{step}}(t_n) \end{bmatrix}$$

Expanding out the residuals we have that

$$\begin{aligned}
h\mathbf{d}^{(1)}_{\text{stage}}(t_n) &= y(t_{n-2} + \mathbf{c}h) - y(t_{n-2} + \mathbf{c}h) = \mathbf{0} \\
h\mathbf{d}^{(2)}_{\text{stage}}(t_n) &= y(t_{n-1} + \mathbf{c}h) - (E - U)\, y(t_{n-1}) \\
&\quad -h\mathbb{A} F\left(y\left(t_{n-1} + \mathbf{c}h\right)\right) - h\mathbb{B} F\left(y(t_{n-1} + (\mathbf{c} - \mathbf{e})\, h)\right) \\
h\mathbf{d}^{(1)}_{\text{step}}(t_n) &= y(t_{n-1}) - y(t_{n-1}) = \mathbf{0} \\
h\mathbf{d}^{(2)}_{\text{step}}(t_n) &= y(t_n) - (I - \Theta)\, y(t_{n-1}) - \Theta y(t_{n-2}) \\
&\quad -h\mathbb{V} F(y(t_{n-1} + \mathbf{c}h)) - h\mathbb{W} F(y(t_{n-1} + (\mathbf{c} - \mathbf{e})\, h))
\end{aligned}$$

Expanding in a Taylor series around t_{n-1}, we see that

$$h\mathbf{d}^{(2)}_{\text{stage}}(t_n) = \begin{bmatrix} \sum_{\nu \geq 1} C^{\{1,1\}}_{\text{stage},\nu} h^\nu \frac{d^\nu}{dt^\nu} y^{\{1\}}(t_{n-1}) \\ \vdots \\ \sum_{\nu \geq 1} C^{\{1,N\}}_{\text{stage},\nu} h^\nu \frac{d^\nu}{dt^\nu} y^{\{N\}}(t_{n-1}) \\ \sum_{\nu \geq 1} C^{\{2,1\}}_{\text{stage},\nu} h^\nu \frac{d^\nu}{dt^\nu} y^{\{1\}}(t_{n-1}) \\ \vdots \\ \vdots \\ \sum_{\nu \geq 1} C^{\{N,N\}}_{\text{stage},\nu} h^\nu \frac{d^\nu}{dt^\nu} y^{\{N\}}(t_{n-1}) \end{bmatrix} \quad \text{and} \quad h\mathbf{d}^{(2)}_{\text{step}}(t_n) = \begin{bmatrix} \sum_{\nu \geq 1} C^{\{1\}}_{\text{step},\nu} h^\nu \frac{d^\nu}{dt^\nu} y^{\{1\}}(t_{n-1}) \\ \vdots \\ \sum_{\nu \geq 1} C^{\{N\}}_{\text{step},\nu} h^\nu \frac{d^\nu}{dt^\nu} y^{\{N\}}(t_{n-1}) \end{bmatrix},$$

where the coefficients $C^{\{i,j\}}_{\text{stage},\nu}$ and $C^{\{i\}}_{\text{step},\nu}$ are given by

$$C^{\{i,j\}}_{\text{stage},\nu} = \frac{\mathbf{c}^\nu}{\nu!} - \frac{(-1)^\nu}{\nu!}\mathbf{u}^{\{i,j\}} - \frac{\mathbf{A}^{\{i,j\}}\mathbf{c}^{\nu-1}}{(\nu-1)!} - \frac{\mathbf{B}^{\{i,j\}}(\mathbf{c}-\mathbf{e})^{\nu-1}}{(\nu-1)!}, \tag{2.17}$$

$$C^{\{i\}}_{\text{step},\nu} = \frac{1}{\nu!} - \frac{(-1)^\nu}{\nu!}\theta^{\{i\}} - \frac{\left[\mathbf{v}^{\{i\}}\right]^T \mathbf{c}^{\nu-1}}{(\nu-1)!} - \frac{\left[\mathbf{w}^{\{i\}}\right]^T (\mathbf{c}-\mathbf{e})^{\nu-1}}{(\nu-1)!}. \tag{2.18}$$

These are the standard stage (2.5) and step (2.6) order conditions for a TSRK method, with no extra coupling conditions present.

The global error after n_s steps is due to the accumulation of local discretization errors (2.15). To analyze the error propagation, we subtract the discrete method (2.12) from its continuous analogue (2.15), which leads to the linear error recurrence relation,

$$\mathbf{q}(t_n) = \mathscr{A}\mathbf{q}(t_{n-1}) + h\mathscr{B}\mathbf{r}(t_n) + h\mathbf{d}(t_n), \quad n = 2,\dots,n_s, \tag{2.19a}$$

where

$$\mathbf{q}(t_1) = \begin{bmatrix} \mathbf{q}^{(1)}_{\text{stage}}(t_1) \\ \mathbf{q}^{(2)}_{\text{stage}}(t_1) \\ \mathbf{q}^{(1)}_{\text{step}}(t_1) \\ \mathbf{q}^{(2)}_{\text{step}}(t_1) \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ y(t_0 + \mathbf{c}h) - Y^{[1]} \\ y(t_0) - y_0 \\ y(t_1) - y_1 \end{bmatrix}, \tag{2.19b}$$

$$\mathbf{q}(t_n) = \begin{bmatrix} \mathbf{q}^{(1)}_{\text{stage}}(t_n) \\ \mathbf{q}^{(2)}_{\text{stage}}(t_n) \\ \mathbf{q}^{(1)}_{\text{step}}(t_n) \\ \mathbf{q}^{(2)}_{\text{step}}(t_n) \end{bmatrix} = \begin{bmatrix} y(t_{n-2} + \mathbf{c}h) - Y^{[n-1]} \\ y(t_n + \mathbf{c}h) - Y^{[n]} \\ y(t_{n-1}) - y_{n-1} \\ y(t_n) - y_n \end{bmatrix}, \quad n \geq 2,$$

and

$$\mathbf{r}(t_n) = \begin{bmatrix} \mathbf{r}^{(1)}_{\text{stage}}(t_n) \\ \mathbf{r}^{(2)}_{\text{stage}}(t_n) \\ \mathbf{r}^{(1)}_{\text{step}}(t_n) \\ \mathbf{r}^{(2)}_{\text{step}}(t_n) \end{bmatrix} = \begin{bmatrix} F(y(t_{n-2} + \mathbf{c}h)) - F\left(Y^{[n-1]}\right) \\ F(y(t_n + \mathbf{c}h)) - F\left(Y^{[n]}\right) \\ F(y(t_{n-1})) - F(y_{n-1}) \\ F(y(t_n)) - F(y_n) \end{bmatrix}, \quad n \geq 2.$$

The solution to (2.19) is given by

$$\mathbf{q}(t_n) = \mathscr{A}^{n-1}\mathbf{q}(t_1) + h\sum_{l=1}^{n-1} \mathscr{A}^{n-1-l}\mathbf{d}(t_{l+1}) + h\sum_{l=1}^{n-1} \mathscr{A}^{n-1-l}\mathscr{B}\mathbf{r}(t_{l+1}).$$

It is shown in Appendix **??** that, when the individual methods are zero-stable ($-1 < \theta \leq 1$), powers of $\mathscr{A}^{\mu}$ take the form

$$\mathscr{A}^{\mu} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \alpha^{(\mu)}_{1,1} & \alpha^{(\mu)}_{1,2} \\ \mathbf{0} & \mathbf{0} & \alpha^{(\mu)}_{2,1} & \alpha^{(\mu)}_{2,2} \\ \mathbf{0} & \mathbf{0} & \beta^{(\mu)}_{1,1} & \beta^{(\mu)}_{1,2} \\ \mathbf{0} & \mathbf{0} & \beta^{(\mu)}_{2,1} & \beta^{(\mu)}_{2,2} \end{bmatrix},$$

where $\alpha^{(\mu)}_{k,l}$ and $\beta^{(\mu)}_{k,l}$ are uniformly bounded. Following the arguments in [7] and assuming that the starting values are approximations of order p and that the method has order of consistency p, we find that the global error of the method after n steps can be written

$$\begin{aligned} q_{\text{step}} = {} & h\mathbb{W}\mathbf{r}^{(2)}_{\text{stage}}(t_{n-1}) + h\mathbb{V}\mathbf{r}^{(2)}_{\text{stage}}(t_n) \\ & + h(1-\Theta)\mathbb{V}\mathbf{r}^{(2)}_{\text{stage}}(t_{n-1}) + h(1-\Theta)\mathbb{W}\mathbf{r}^{(2)}_{\text{stage}}(t_{n-2}) \\ & + h\sum_{l=1}^{n-3} \beta^{(n-1-l)}_{2,2}\left(\mathbb{W}\mathbf{r}^{(2)}_{\text{stage}}(t_l) + \mathbb{V}\mathbf{r}^{(2)}_{\text{stage}}(t_{l+1})\right), \end{aligned}$$

as $h \to 0$.

The assumption that the method has order of consistency p, is equivalent to (2.18); if we in addition require (2.17), then $\mathbf{q}^{(2)}_{\text{step}} = \mathscr{O}(h^{q+1}) = \mathscr{O}(h^p)$ as $h \to 0$.

Remark 2.1 (Startup procedure for SLP-TSRK schemes.) It is not enough to only define the coefficients of the integrator, absent some strategy for obtaining the starting values with appropriate accuracy. Fortunately, since we consider only schemes for which $q \geq p-1$, there are two well known, and straightforward, approaches to obtaining such values [14,6,7]. The first and most efficient, is the application of a continuous output Runge-Kutta scheme of appropriate order. A single timestep of the continuous Runge-Kutta method can compute from y_0, both y_1 and all Y_i. A simple application of the RHS to the Y_i's to produce K_i's is sufficient to start the SLP-TSRK scheme. Alternatively, a discrete Runge-Kutta method can be repeatedly applied to compute y_i and the Y_i's from y_0.

3 Subfamilies of SLP-TSRK

The leniency in coupling conditions allows for a tremendous amount of freedom in the construction of an SLP-TSRK. Here we outline two computationally relevant families of SLP-TSRK schemes. The first are referred to as asynchronous SLP-TSRK in which the diagonal method is either explicit or implicit, while the off-diagonal scheme is super-explicit, requiring only time-lagged ghost data. The second is the so-called locally-implicit SLP-TSRK. This scheme employs an implicit diagonal method, along with a (super-)explicit off-diagonal scheme, requiring only (non-)linear solves depending on valid data.

3.1 Asynchronous SLP-TSRK

An asynchronous SLP-TSRK on an individual processing element is given as Move paragraph below to here

$$Y_i^{\{\dagger\}} = \left(1-u_i^{\{\dagger\}}\right) y_{n-1}^{\{\dagger\}} + u_i^{\{\dagger\}} y_{n-2}^{\{\dagger\}} + h\sum_{j=1}^{i-2} a_{i,j}^{\{\dagger\}} K_j^{\{\dagger\}[n]} + h\sum_{j=1}^{s} b_{i,j}^{\{\dagger\}} K_j^{\{\dagger\}[n-1]}, \tag{3.1a}$$

$$Y_i^{\{*\}} = \left(1-u_i^{\{*\}}\right) y_{n-1}^{\{*\}} + u_i^{\{*\}} y_{n-2}^{\{*\}} + h\sum_{j=1}^{i-1} a_{i,j}^{\{*\}} K_j^{\{*\}[n]} + h\sum_{j=1}^{s} b_{i,j}^{\{*\}} K_j^{\{*\}[n-1]}, \tag{3.1b}$$

$$K_i^{\{*\}[n]} = f^{\{*\}}\left(Y_i^{\{*\}}, Y_i^{\{\dagger\}}\right) \quad \text{for } i = 1,\ldots,s, \tag{3.1c}$$

$$y_n^{\{*\}} = \left(1-\theta^{\{*\}}\right) y_{n-1}^{\{*\}} + \theta^{\{*\}} y_{n-2}^{\{*\}} + h\sum_{j=1}^{s} v_j^{\{*\}} K_j^{\{*\}[n]} + h\sum_{j=1}^{s} w_j^{\{*\}} K_j^{\{*\}[n-1]}, \tag{3.1d}$$

$$\tag{3.1e}$$

where once again $*$ denotes a valid (Data owned by PE $*$ and computed on PE $*$) solution, $\dagger$ denotes a ghost (Data owned by PE $\dagger$ and computed on PE $*$) solution, and the computation is applied to all processing elements simultaneously. We note that the quantities $y^{\{\dagger\}}$ and $K_j^{\{\dagger\}}$ are computed on some other PE and communicated to the local PE, where the reduced internal stage dependency in (3.1a) versus (3.1b) guarantees we have an entire stage calculation to obscure the cost of that communication. This discrepancy is essential to the underlying strategy of overlapping computation with communication. Each local approximation of the ghost data requires only information that is an entire stage behind the approximation being computed for the valid solution. In this way, we have an entire stage worth of computation with which to obscure the communication costs. In order to guarantee asynchrony, in addition to time-lagging by a single stage we must also require that $u_1^{\{\dagger\}} = 1$ and $b_{1s}^{\{\dagger\}} = 0$, so that the first valid stage approximation does not depend on ghost data of the current timestep or the final stage of the previous timestep. Such a scheme would have a Butcher tableau of the form

$$\mathbf{u}^{\{\dagger\}} \left| \mathbf{A}^{\{\dagger\}} \right| \mathbf{B}^{\{\dagger\}} = \begin{array}{c|ccc} 1 & & & \\ u_2 & & & \\ u_3 & a_{3,1} & & \\ \vdots & \vdots & \ddots & \\ u_s & a_{s,1} & \cdots & a_{s,s-2} \end{array} \left| \begin{array}{cccc} b_{1,1} & b_{1,2} & \cdots & 0 \\ b_{2,1} & b_{2,2} & \cdots & b_{2,s} \\ b_{3,1} & b_{3,2} & \cdots & b_{3,s} \\ \vdots & \vdots & & \vdots \\ b_{s,1} & b_{s,2} & \cdots & b_{s,s} \end{array} \right., \tag{3.2}$$

and its implementation is illustrated in detail in Algorithm 1.

Algorithm 1 A single step of an asynchronous SLP-TSRK

for each stage $i \in \{1,\dots,s-1\}$ **do**
  ▷ Wait for required (time lagged) information
  if $i == 1$ **then**
    MPI_WAIT() for $K_{s-1}^{\{\dagger\}[n-1]}$
  else if $i == 2$ **then**
    MPI_WAIT() for $K_s^{\{\dagger\}[n-1]}$ and $y_{n-1}^{\{\dagger\}}$
  else if $i > 2$ **then**
    MPI_WAIT() for $K_{i-2}^{\{\dagger\}[n]}$
  end if
  ▷ Compute ghost stage approximation
  $Y_i^{\{\dagger\}} = \left(1-u_i^{\{\dagger\}}\right) y_{n-1}^{\{\dagger\}} + u_i^{\{\dagger\}} y_{n-2}^{\{\dagger\}} + h\sum_{j=1}^{i-2} a_{i,j}^{\{\dagger\}} K_j^{\{\dagger\}[n]} + h\sum_{j=1}^{s} b_{i,j}^{\{\dagger\}} K_j^{\{\dagger\}[n-1]}$
  ▷ Compute valid stage approximation
  $Y_i^{\{*\}} = \left(1-u_i^{\{*\}}\right) y_{n-1}^{\{*\}} + u_i^{\{*\}} y_{n-2}^{\{*\}} + h\sum_{j=1}^{i-1} a_{i,j}^{\{*\}} K_j^{\{*\}[n]} + h\sum_{j=1}^{s} b_{i,j}^{\{*\}} K_j^{\{*\}[n-1]}$
  ▷ Compute local stage vector
  $K_i^{\{*\}[n]} = f^{\{*\}}\left(Y_i^{\{*\}}, Y_i^{\{\dagger\}}\right)$
  ▷ Send valid data to appropriate non-local processing elements
  MPI_Isend$\left(K_i^{\{*\}[n]}\right)$
  MPI_Irecv$\left(K_i^{\{\dagger\}[n]}\right)$
end for
▷ Compute final stage and new step
MPI_WAIT() for $K_{s-2}^{\{\dagger\}[n]}$
$Y_s^{\{\dagger\}} = \left(1-u_i^{\{\dagger\}}\right) y_{n-1}^{\{\dagger\}} + u_i^{\{\dagger\}} y_{n-2}^{\{\dagger\}} + h\sum_{j=1}^{s-2} a_{i,j}^{\{\dagger\}} K_j^{\{\dagger\}[n]} + h\sum_{j=1}^{s} b_{i,j}$
$Y_s^{\{*\}} = \left(1-u_i^{\{*\}}\right) y_{n-1}^{\{*\}} + u_i^{\{*\}} y_{n-2}^{\{*\}} + h\sum_{j=1}^{s-1} a_{i,j}^{\{*\}} K_j^{\{*\}[n]} + h\sum_{j=1}^{s} b_{i,j}^{\{*\}} K_j^{\{*\}[n-1]}$
$K_s^{\{*\}[n]} = f^{\{*\}}\left(Y_i^{\{*\}}, Y_i^{\{\dagger\}}\right)$
$y_n^{\{*\}} = \left(1-\theta^{\{*\}}\right) y_{n-1}^{\{*\}} + \theta^{\{*\}} y_{n-2}^{\{*\}} + h\sum_{j=1}^{s} v_j^{\{*\}} K_j^{\{*\}[n]} + h\sum_{j=1}^{s} w_j^{\{*\}} K_j^{\{*\}[n-1]}$
▷ Send valid data to appropriate non-local processing elements
MPI_Isend$\left(K_s^{\{*\}[n]}\right)$, MPI_Isend$\left(y_n^{\{*\}}\right)$
MPI_Irecv$\left(K_s^{\{\dagger\}[n]}\right)$, MPI_Irecv$\left(y_n^{\{\dagger\}}\right)$

3.2 Locally-Implicit SLP-TSRK

A locally-implicit SLP-TSRK for an individual processing element is given as

$$\begin{aligned} Y_i^{\{\dagger\}} &= \left(1-u_i^{\{\dagger\}}\right) y_{n-1}^{\{\dagger\}} + u_i^{\{\dagger\}} y_{n-2}^{\{\dagger\}} + h\sum_{j=1}^{i-1} a_{i,j}^{\{\dagger\}} K_j^{\{\dagger\}[n]} \\ &\quad + h\sum_{j=1}^{s} b_{i,j}^{\{\dagger\}} K_j^{\{\dagger\}[n-1]}, \end{aligned} \tag{3.3a}$$

$$\begin{aligned} Y_i^{\{*\}} &= \left(1-u_i^{\{*\}}\right) y_{n-1}^{\{*\}} + u_i^{\{*\}} y_{n-2}^{\{*\}} + h\sum_{j=1}^{i-1} a_{i,j}^{\{*\}} K_j^{\{*\}[n]} \\ &\quad + h\sum_{j=1}^{s} b_{i,j}^{\{*\}} K_j^{\{*\}[n-1]} + h a_{i,i}^{\{*\}} f^{\{*\}}\left(Y_i^{\{*\}}, Y_i^{\{\dagger\}}\right), \end{aligned} \tag{3.3b}$$

$$K_i^{\{*\}[n]} = f^{\{*\}}\left(Y_i^{\{*\}}, Y_i^{\{\dagger\}}\right) \quad \text{for } i = 1,\dots,s, \tag{3.3c}$$

$$\begin{aligned} y_n^{\{*\}} &= \left(1-\theta^{\{*\}}\right) y_{n-1}^{\{*\}} + \theta^{\{*\}} y_{n-2}^{\{*\}} + h\sum_{j=1}^{s} v_j^{\{*\}} K_j^{\{*\}[n]} \\ &\quad + h\sum_{j=1}^{s} w_j^{\{*\}} K_j^{\{*\}[n-1]}, \end{aligned} \tag{3.3d}$$

(3.3e)

where the computation of the above method is applied to all processing elements simultaneously, and $K^\dagger$ values are computed as K^* on non-local PEs and then communicated to the current PE as $K^\dagger$. We note that the local stage approximation of the ghost data $Y_i^{\{\dagger\}}$ is explicit, while the local stage approximation of the valid data is implicit, though only in the valid data $Y_i^{\{*\}}$ itself. This arrangement provides tremendous computational advantage since the cost of standard implicit methods is dominated by the need for global communication at each iteration of a nonlinear solver. In addition to any parallel benefits, it also leverages the time-honored tradition in numerical analysis of saving computational effort by solving many small systems instead of a single large system.

Remark 3.1 (Relative cost of SLP-TSRK versus standard TSRK) We note here that explicit SLP-TSRK schemes are only slightly more expensive than standard explicit TSRK schemes in both storage and computation. Specifically, explicit SLP-TSRK schemes require computing – and storing – local approximations of ghost data (that would be computed nonlocally and communicated to the current PE, in a standard TSRK) on each PE. The amount of extra computation and storage is problem dependent, relying entirely on the choice of partitioning in Equation (2.7). Fortunately, these methods benefit from keeping the ratio of ghost to valid data small, a feature that is already sought in the parallel decomposition of most PDEs. Additionally, implicit SLP-TSRK schemes benefit from their need to solve only N small, locally-implicit nonlinear systems, versus the standard implicit schemes single large implicit solve. Putting everything together, an SLP-TSRK scheme is about as costly per-timestep as a standard TSRK scheme, but its potentially asynchronous nature will lead to improved parallel scaling.

4 Stability of SLP-TSRK

A standard approach to the stability of a time integrator looks at the initial value problem (1.1) from which we started and reduces this problem to the much simpler linear ODE

$$y' = Ly, \quad y(t_0) = y_0 \in \mathbb{R}^d, \quad L \in \mathbb{R}^{d\times d}, \quad t_0 \le t \le t_f. \tag{4.1}$$

We then further assume that L can be diagonalized, and then that the action of this diagonalizable matrix can be approximated in some sense by a single scalar value. This string of assumptions leads us to the standard linear, scalar valued Dahlquist test problem

$$y' = \lambda y. \tag{4.2}$$

Unfortunately, here we are interested in the stability of a method that inherently depends on a component partitioning of the test ODE. This structure leads us to make the choice that our test problem should be vector valued, since the scalar problem is not reasonable. Instead, we once again reduce our test problem from non-linear to

linear; then for simplicity, consider a scheme with only two partitions so that our linear operator can be written in block form as

$$L = \begin{bmatrix} L_{1,1} & L_{1,2} \\ L_{2,1} & L_{2,2} \end{bmatrix}. \tag{4.3}$$

Finally, we make our last, and potentially least reasonable, assumption that the action of each submatrix can be in some sense approximated by a single scalar. This line of reasoning leads to the stability problem we consider here,

$$y' = \Lambda y = \begin{bmatrix} \lambda_{1,1} & \lambda_{1,2} \\ \lambda_{2,1} & \lambda_{2,2} \end{bmatrix} y, \tag{4.4}$$

where we partition the system according to the two components of y.

4.1 Solving the stability problem

We seek here a solution of the stability problem, specifically of the form

$$y_n = M y_{n-1}.$$

In this way, we can determine the stability of the system, where the system is stable if $\rho(M) \leq 1$ and unstable otherwise. We build the supervectors

$$Y^{\{1,1\}} = \begin{bmatrix} Y_1^{\{1,1\}} \\ \vdots \\ Y_s^{\{1,1\}} \end{bmatrix}, \quad Y^{\{1,2\}} = \begin{bmatrix} Y_1^{\{1,2\}} \\ \vdots \\ Y_s^{\{1,2\}} \end{bmatrix}, \quad Y^{\{2,1\}} = \begin{bmatrix} Y_1^{\{2,1\}} \\ \vdots \\ Y_s^{\{2,1\}} \end{bmatrix}, \quad Y^{\{2,2\}} = \begin{bmatrix} Y_1^{\{2,2\}} \\ \vdots \\ Y_s^{\{2,2\}} \end{bmatrix}. \tag{4.5}$$

so that when we apply the method (2.9) to the test problem we obtain

$$
\begin{aligned}
Y^{\{1,1\}[n]} &= \left(\mathbf{e}-\mathbf{u}^{\{1,1\}}\right) y_{n-1}^{\{1\}} + \mathbf{u}^{\{1,1\}} y_{n-2}^{\{1\}} + h\mathbf{A}^{\{1,1\}} \left(\lambda_{1,1} Y^{\{1,1\}[n]} + \lambda_{1,2} Y^{\{1,2\}[n]}\right) \\
&\quad + h\mathbf{B}^{\{1,1\}} \left(\lambda_{1,1} Y^{\{1,1\}[n-1]} + \lambda_{1,2} Y^{\{1,2\}[n-1]}\right) \qquad (4.6a) \\
Y^{\{1,2\}[n]} &= \left(\mathbf{e}-\mathbf{u}^{\{1,2\}}\right) y_{n-1}^{\{2\}} + \mathbf{u}^{\{1,2\}} y_{n-2}^{\{2\}} + h\mathbf{A}^{\{1,2\}} \left(\lambda_{2,1} Y^{\{2,1\}[n]} + \lambda_{2,2} Y^{\{2,2\}[n]}\right) \\
&\quad + h\mathbf{B}^{\{1,2\}} \left(\lambda_{2,1} Y^{\{2,1\}[n-1]} + h\lambda_{2,2} Y^{\{2,2\}[n-1]}\right) \qquad (4.6b) \\
Y^{\{2,1\}[n]} &= \left(\mathbf{e}-\mathbf{u}^{\{2,1\}}\right) y_{n-1}^{\{1\}} + \mathbf{u}^{\{2,1\}} y_{n-2}^{\{1\}} + h\mathbf{A}^{\{2,1\}} \left(\lambda_{1,1} Y^{\{1,1\}[n]} + \lambda_{1,2} Y^{\{1,2\}[n]}\right) \\
&\quad + h\mathbf{B}^{\{2,1\}} \left(\lambda_{1,2} Y^{\{1,1\}[n-1]} + \lambda_{1,2} Y^{\{1,2\}[n-1]}\right) \qquad (4.6c) \\
Y^{\{2,2\}[n]} &= \left(\mathbf{e}-\mathbf{u}^{\{2,2\}}\right) y_{n-1}^{\{2\}} + \mathbf{u}^{\{2,2\}} y_{n-2}^{\{2\}} + h\mathbf{A}^{\{2,2\}} \left(\lambda_{2,1} Y^{\{2,1\}[n]} + \lambda_{2,2} Y^{\{2,2\}[n]}\right) \\
&\quad + h\mathbf{B}^{\{2,2\}} \left(\lambda_{2,1} Y^{\{2,1\}[n-1]} + \lambda_{2,2} Y^{\{2,2\}[n-1]}\right) \qquad (4.6d) \\
y_n^{\{1\}} &= \left(1-\theta^{\{1\}}\right) y_{n-1}^{\{1\}} + \theta^{\{1\}} y_{n-2}^{\{1\}} + h\left(\mathbf{v}^{\{1\}}\right)^T \left(\lambda_{1,1} Y^{\{1,1\}[n]} + \lambda_{1,2} Y^{\{1,2\}[n]}\right) \\
&\quad + h\left(\mathbf{w}^{\{1\}}\right)^T \left(\lambda_{1,1} Y^{\{1,1\}[n-1]} + \lambda_{1,2} Y^{\{1,2\}[n-1]}\right) \qquad (4.6e) \\
y_n^{\{2\}} &= \left(1-\theta^{\{2\}}\right) y_{n-1}^{\{2\}} + \theta^{\{2\}} y_{n-2}^{\{2\}} + h\left(\mathbf{v}^{\{2\}}\right)^T \left(\lambda_{2,1} Y^{\{2,1\}[n]} + \lambda_{2,2} Y^{\{2,2\}[n]}\right) \\
&\quad + h\left(\mathbf{w}^{\{2\}}\right)^T \left(\lambda_{2,1} Y^{\{2,1\}[n-1]} + \lambda_{2,2} Y^{\{2,2\}[n-1]}\right). \qquad (4.6f)
\end{aligned}
$$

If we define the vector

$$
\mathscr{Y}_n = \begin{bmatrix} y_n^{\{1\}} \\ y_n^{\{2\}} \\ y_{n-1}^{\{1\}} \\ y_{n-1}^{\{2\}} \\ Y^{\{1,1\}[n]} \\ Y^{\{1,2\}[n]} \\ Y^{\{2,1\}[n]} \\ Y^{\{2,2\}[n]} \end{bmatrix}, \qquad (4.7)
$$

then it is possible to write (4.6a),(4.6b), (4.6c), and (4.6d) as

$$
A \begin{bmatrix} Y^{\{1,1\}[n]} \\ Y^{\{1,2\}[n]} \\ Y^{\{2,1\}[n]} \\ Y^{\{2,2\}[n]} \end{bmatrix} = B\mathscr{Y}_{n-1}, \quad A \in \mathbb{R}^{4s\times 4s}, B \in \mathbb{R}^{4s\times(4s+4)}, \qquad (4.8)
$$

with

$$
A = \begin{bmatrix} \left(I - h\lambda_{1,1}\mathbf{A}^{\{1,1\}}\right) & -h\lambda_{1,2}\mathbf{A}^{\{1,1\}} & 0 & 0 \\ 0 & I & -h\lambda_{2,1}\mathbf{A}^{\{1,2\}} & -h\lambda_{2,2}\mathbf{A}^{\{1,2\}} \\ -h\lambda_{1,1}\mathbf{A}^{\{2,1\}} & -h\lambda_{1,2}\mathbf{A}^{\{2,1\}} & I & 0 \\ 0 & 0 & -h\lambda_{2,1}\mathbf{A}^{\{2,2\}} & \left(I - h\lambda_{2,2}\mathbf{A}^{\{2,2\}}\right) \end{bmatrix}, \qquad (4.9)
$$

and

$$B=\begin{bmatrix} \left(\mathbf{e}-\mathbf{u}^{\{1,1\}}\right) & 0 & \mathbf{u}^{\{1,1\}} & 0 & h\lambda_{1,1}\mathbf{B}^{\{1,1\}} & h\lambda_{1,2}\mathbf{B}^{\{1,1\}} & 0 & 0 \\ 0 & \left(\mathbf{e}-\mathbf{u}^{\{1,2\}}\right) & 0 & \mathbf{u}^{\{1,2\}} & 0 & 0 & h\lambda_{2,1}\mathbf{B}^{\{1,2\}} & h\lambda_{2,2}\mathbf{B}^{\{1,2\}} \\ \left(\mathbf{e}-\mathbf{u}^{\{2,1\}}\right) & 0 & \mathbf{u}^{\{2,1\}} & 0 & h\lambda_{1,1}\mathbf{B}^{\{2,1\}} & h\lambda_{1,2}\mathbf{B}^{\{2,1\}} & 0 & 0 \\ 0 & \left(\mathbf{e}-\mathbf{u}^{\{2,2\}}\right) & 0 & \mathbf{u}^{\{2,2\}} & 0 & 0 & h\lambda_{2,1}\mathbf{B}^{\{2,2\}} & h\lambda_{2,2}\mathbf{B}^{\{2,2\}} \end{bmatrix}. \tag{4.10}$$

The solution to (4.8) can then be written as

$$\begin{bmatrix} Y^{\{1,1\}[n]} \\ Y^{\{1,2\}[n]} \\ Y^{\{2,1\}[n]} \\ Y^{\{2,2\}[n]} \end{bmatrix} = C\mathscr{Y}_{n-1}, \quad C=A^{-1}B\in\mathbb{R}^{4s\times(4s+4)}. \tag{4.11}$$

Let us now label the 4, s-row, components of C so that

$$C=\begin{bmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \end{bmatrix}, \tag{4.12}$$

and

$$Y^{\{1,1\}[n]}=C_1\mathscr{Y}_{n-1}, \quad Y^{\{1,2\}[n]}=C_2\mathscr{Y}_{n-1}, \quad Y^{\{2,1\}[n]}=C_3\mathscr{Y}_{n-1}, \quad Y^{\{2,2\}[n]}=C_4\mathscr{Y}_{n-1}. \tag{4.13}$$

From this it is clear that we can rewrite the solution to (4.6) as

$$\mathscr{Y}_n=\mathbf{M}(\lambda_{1,1},\lambda_{1,2},\lambda_{2,1},\lambda_{2,2},h)\mathscr{Y}_{n-1}, \quad \mathbf{M}\in\mathbb{R}^{(4s+4)\times(4s+4)}, \tag{4.14}$$

where

$$\mathbf{M}=\begin{bmatrix} \mathbf{M}_1 \\ C \end{bmatrix}+\begin{bmatrix} \mathbf{M}_2 \\ \mathbf{0} \end{bmatrix} \tag{4.15}$$

with

$$\mathbf{M}_1=\begin{bmatrix} \left(1-\theta^{\{1\}}\right) & 0 & \theta^{\{1\}} & 0 & h\lambda_{1,1}\left(\mathbf{w}^{\{1\}}\right)^T & h\lambda_{1,2}\left(\mathbf{w}^{\{1\}}\right)^T & \mathbf{0} & \mathbf{0} \\ 0 & \left(1-\theta^{\{2\}}\right) & 0 & \theta^{\{2\}} & \mathbf{0} & \mathbf{0} & h\lambda_{2,1}\left(\mathbf{w}^{\{2\}}\right)^T & h\lambda_{2,2}\left(\mathbf{w}^{\{2\}}\right)^T \\ 1 & 0 & 0 & 0 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ 0 & 1 & 0 & 0 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{4.16}$$

and

$$\mathbf{M}_2=\begin{bmatrix} h\left(\mathbf{v}^{\{1\}}\right)^T\left(\lambda_{1,1}C_1+\lambda_{1,2}C_2\right) \\ h\left(\mathbf{v}^{\{2\}}\right)^T\left(\lambda_{2,1}C_3+\lambda_{2,2}C_4\right) \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}. \tag{4.17}$$

The matrix $\mathbf{M}$ is the SLP-TSRK iteration matrix, and the method is stable iff

$$\rho(\mathbf{M}(\Lambda,h))\leq 1. \tag{4.18}$$

The stability result is equally valid for implicit or explicit methods, and for asynchronous or locally-implicit methods.

4.2 Leveraging the stability problem

Here, we follow the approach presented in [8,9,13] to leverage (4.4) to gain an understanding of the relative stability properties of our new methods. We restrict our interest to the case where $\lambda_{1,1}, \lambda_{2,2} < 0$ and

$$\delta = \frac{\lambda_{1,2}\lambda_{2,1}}{\lambda_{1,1}\lambda_{2,2}} < 1,$$

so that the eigenvalues of the matrix Λ have negative real parts. Then as in [8], the parameter δ can be considered a measure of the coupling between the partitions, and

$$\kappa = \frac{\lambda_{2,2}}{\lambda_{1,1}} \geq 1$$

can represent the relative stiffness of the partitions. In [13] it is shown for the numerical method they are studying that the iteration matrix $\mathbf{M}$ depends only on the quantities $h\lambda_{1,1}$, $h\lambda_{2,2}$, and $\det(h\Lambda)$, and so the stability region can be considered as depending only on the quantities $\kappa \geq 1$ and

$$\xi = \frac{h\lambda_{1,1}}{1 - h\lambda_{1,1}} \in (-1, 0) \quad \text{and} \quad \eta = \frac{\delta}{2 - \delta} \in (-1, 1). \tag{4.19}$$

Unfortunately, here we do not have access to the same theoretical result; however, empirically it appears to be the case. This is confirmed by the fact that we obtain the same results as in Figures 5.1, 5.2, and 5.3 independent of our choice of how to generate a Λ and h from the triplet (κ, ξ, η).

5 Constructing Practical SLP-TSRK

The order conditions of SLP-TSRK methods are fairly straightforward. It is possible to take any collection of existing internally consistent TSRK schemes and combine them to obtain a convergent SLP-TSRK scheme. Here we seek to do more. Specifically, we seek to construct the two-partition schemes from Section 3. To do so we find pairs of TSRK schemes that satisfy equations (2.17) and (2.18) while optimizing for stability as discussed in Section 4.

To construct such an optimized pair we must first build an objective function over which to optimize. We begin by considering the stability test problem (4.4), as well as a set of triplets (κ, ξ, η). Then for each triplet (κ, ξ, η), we construct a corresponding matrix, Λ, and stepsize, h. The objective function for a given scheme is then evaluated as the total number of triplets (κ, ξ, η) at which the scheme is stable. In addition to optimizing our objective function the new scheme is subject to the constraints of the order conditions (2.17) and (2.18).

Note that these order conditions are linear as long as the abscissa is fixed, and so are straightforward to solve. We therefore choose the abscissa among our optimization parameters, and employ a direct solve to obtain the rest of the method coefficients. The optimization is then performed using the genetic algorithm Python

$$\mathbf{c} = \begin{bmatrix} 0.19357073 & 0.50352658 & 0.97750613 \end{bmatrix}$$

$$\mathbf{u}^{\{*\}} = \begin{bmatrix} 1.86133177 \\ 1.74652867 \\ 1.429326 \end{bmatrix} \quad \mathbf{A}^{\{*\}} = \begin{bmatrix} 0 & 0 & 0 \\ 0.3258515912186877 & 0 & 0 \\ 0.27635351287871057 & 0.5142499678827194 & 0 \end{bmatrix}$$

$$\mathbf{B}^{\{*\}} = \begin{bmatrix} 0.7503417276510276 & 0.5854449264336774 & 0.7191158460866666 \\ 0.8006509363957107 & 0.292478352224931 & 0.8310743757572607 \\ 0.5771618722770031 & 0.12769222141061487 & 0.9113745608482877 \end{bmatrix}$$

$$\theta^{\{*\}} = 0.34725408186734763 \quad \mathbf{v}^{\{*\}} = \begin{bmatrix} 0.4317772 \\ 0.30848125 \\ 0.26559022 \end{bmatrix}, \quad \mathbf{w}^{\{*\}} = \begin{bmatrix} 0.06333613 \\ 0.24224691 \\ 0.03582237 \end{bmatrix}$$

$$\mathbf{u}^{\{\dagger\}} = \begin{bmatrix} 1 \\ 1.44566481 \\ 1.20800457 \end{bmatrix} \quad \mathbf{A}^{\{\dagger\}} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1.0104785400466718 & 0 & 0 \end{bmatrix}$$

$$\mathbf{B}^{\{\dagger\}} = \begin{bmatrix} -0.3591187218675605 & 1.5526894489850318 & 0 \\ 0.23169899162496854 & 0.7818501907175646 & 0.9356422138568928 \\ 0.10951478068520126 & 0.4421260380601868 & 0.6233913464486004 \end{bmatrix}$$

Table 5.1: Asynchronous SLP-TSRK of order 3 with stage order 2.

package PyGAD [5]. Tables 5.1 and 5.2 give specific values for both an asynchronous and a locally-implicit asynchronous SLP-TSRK scheme of method order three with stage order two. The coefficients with only eight decimals of precision were chosen by the optimizer, while those with 16 decimals of precision were determined by the preceding choices.

It is possible to leverage this optimization process to derive SLP-TSRK schemes that are advantaged for specific problems. For instance, if one seeks a scheme that is extremely stable for diffusion type problems, one might only consider symmetric Λ's or Λ's in Equation (4.4) that have strictly negative real eigenvalues. Here we considered a sequence of Λ's arising from the triplets (κ, ξ, η), where $\kappa \in 1, 10, 100$ and ξ and η come from an equispaced grid of 150 points on $(-1, 0)$ and $(-1, 1)$ respectively.

Figures 5.1, 5.2, and 5.3 show the stability plots for a 3rd order method of Jackiewicz [7, §5.6.1], as well as our new asynchronous explicit and locally implicit methods. We note that our new asynchronous scheme is slightly less stable then the existing scheme by Jackiewicz, but has the advantage of potential asynchronicity. This reduced stability is unsurprising since the off-diagonal methods are super-explicit and so extrapolate solutions over larger time distances. More interestingly, the locally-implicit method shows improved stability over the explicit methods, with its stability depending only weakly on the relative stiffness κ.

$$\mathbf{c} = \begin{bmatrix} 0.15265147 & 0.90761924 & 0.02019072 \end{bmatrix}$$

$$\mathbf{u}^{\{*\}} = \begin{bmatrix} 2.02516075 \\ 2.46213121 \\ 2.01005038 \end{bmatrix} \quad \mathbf{A}^{\{*\}} = \begin{bmatrix} 0.7172893606610329 & 0 & 0 \\ 0.8354777291752487 & 0.7172893606610329 & 0 \\ 0.2975360872161591 & -0.23506310031758615 & 0.7172893606610329 \end{bmatrix}$$

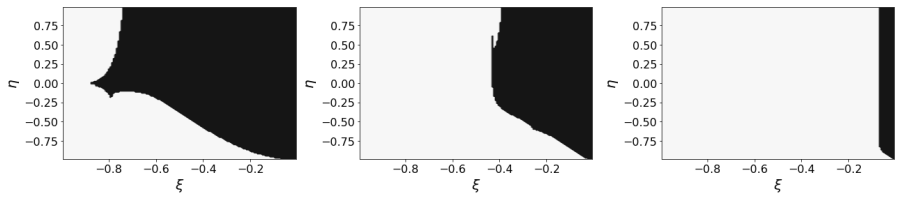
$$\mathbf{B}^{\{*\}} = \begin{bmatrix} 0.9193994118623637 & 0.22404633133217688 & 0.3170771210792589 \\ 0.7277487605615025 & 0.09708637871388748 & 0.992148222147617 \\ 0.932350179294266 & 0.2821133184774526 & 0.031220858701790255 \end{bmatrix}$$

$$\theta^{\{*\}} = 0.3192982358106409, \quad \mathbf{v}^{\{*\}} = \begin{bmatrix} 0.18727826 \\ 0.44043672 \\ 0.37155417 \end{bmatrix}, \quad \mathbf{w}^{\{*\}} = \begin{bmatrix} 0.00983038 \\ 0.24427559 \\ 0.06592312 \end{bmatrix}$$

$$\mathbf{u}^{\{\dagger\}} = \begin{bmatrix} 1 \\ -1.68698949 \\ 0.79801812 \end{bmatrix}, \quad \mathbf{A}^{\{\dagger\}} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ -0.2881064983723251 & 0 & 0 \end{bmatrix}$$

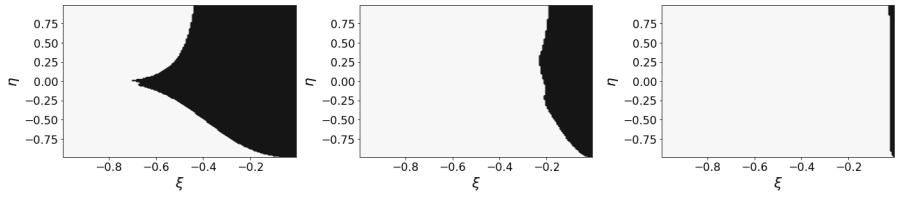
$$\mathbf{B}^{\{\dagger\}} = \begin{bmatrix} 0.5058042991717454 & 0.6468471713514119 & 0 \\ 0.7684348382234261 & 0.43942628779667814 & -1.9872313768468894 \\ 0.2979157830695197 & 0.7771786982185928 & 0.031220858701790255 \end{bmatrix}$$

Table 5.2: Locally-Implicit Asynchronous SLP-TSRK of order 3 with stage order 2.



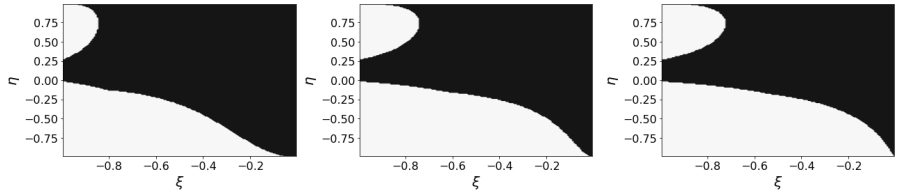
(a) $\kappa = 1$ (b) $\kappa = 10$ (c) $\kappa = 100$

Fig. 5.1: Stability regions (dark) for Jackiewicz 3rd order TSRK for $\kappa = 1, 10, 100$.



(a) $\kappa = 1$ (b) $\kappa = 10$ (c) $\kappa = 100$

Fig. 5.2: Stability regions (dark) for 3rd order asynchronous SLP-TSRK with stage order 2, for $\kappa = 1, 10, 100$.



(a) $\kappa = 1$ (b) $\kappa = 10$ (c) $\kappa = 100$

Fig. 5.3: Stability regions (dark) for 3rd order locally-implicit asynchronous SLP-TSRK with stage order 2, for $\kappa = 1, 10, 100$.

6 Numerical Results

Here we present some numerical experiments that verify the order condition theory presented in section 2.2 and illustrate the stability results of Section 4.

6.1 Verifying order of accuracy via Lorenz 96 with 2 partitions

We make use of the Lorenz-96 model [11] to verify our order condition theory on a standard ODE test problem. This chaotic model has $N = 40$ variables, periodic boundary conditions, and is described by the following equations:

$$\begin{aligned} &\frac{dy_j}{dt} = -y_{j-1}\left(y_{j-2} - y_{j+1}\right) - y_j + F\,, \quad j = 1,\ldots,N\,, \\ &y_{-1} = y_{N-1}\,, \quad y_0 = y_N\,, \quad y_{N+1} = y_1\,. \end{aligned} \tag{6.1}$$

The forcing term is $F = 8.0 + 4\cos(4\pi t)$, with $t \in [0,\ 1.5]$, and $y_i(0) = -2 + 4/39 \cdot (i-1)$. Here, we separate the system into two partitions, each with 20 variables, so that $y^{\{1\}} = \{y_1, \ldots, y_{20}\}$ and $y^{\{2\}} = \{y_{21}, \ldots, y_{40}\}$.

Figure 6.1 shows the convergence of both the asynchronous explicit and locally-implicit SLP-TSRK methods given in Tables 5.1 and 5.2. Both schemes obtain the expected order.

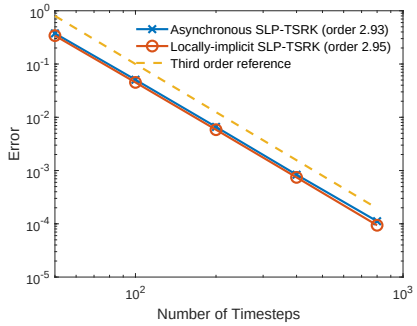


Fig. 6.1: Precision diagram for Lorenz 96, showing convergence order of methods using 2 partitions.

6.2 Locally-implicit SLP-TSRK schemes exhibit better than explicit stability.

Here we consider the 2-dimensional advection-diffusion system with periodic boundary conditions

$$\frac{\partial u}{\partial t} + \mathbf{a} \cdot \nabla u = \sigma \nabla^2 u \tag{6.2}$$

solved via MFEM [2]. Specifically, we use a discontinuous Galerkin discretization with third-order polynomial basis functions in each element. The advection term is treated via standard upwinding, and the diffusion term is treated via a symmetric interior penalty discretization. The system is separated into partitions corresponding to MFEM's choice of data distribution across the available processing elements.

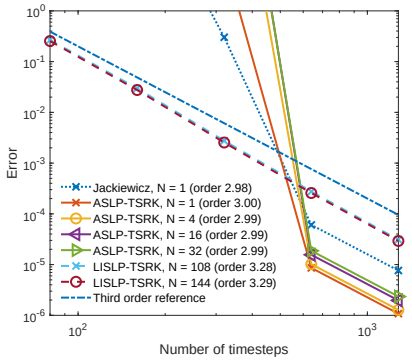


Fig. 6.2: Precision diagram for the advection-diffusion equation with $\sigma = 0.00005$ and a variable number of partitions.

Figure 6.2 shows a convergence diagram for SLP-TSRK schemes applied to the above problem with $\sigma = 0.00005$, $\mathbf{a} = (\sqrt{2/3}, \sqrt{1/3})$, $\mathbf{x} \in [-1,1] \times [-1,1]$,

$t \in [0, 0.125]$, 147,456 unknowns and periodic boundary conditions. These schemes obtain the full, expected order of convergence regardless of the number of partitions, once again verifying the formal order condition theory of section 2.2. In addition, Figure 6.2 illustrates the relative stiffness of this problem, where the explicit methods are unable to obtain low-accuracy (large timestep) solutions due to stability constraints, as well as demonstrates the ability of the locally-implicit scheme to obtain stable solutions with larger than explicit timesteps, thus verifying the stability results of section 4, even while employing a relatively large number of partitions. Recall that as the number of partitions is increased the relative stability of the locally-implicit schemes should decrease, since a larger proportion of the system is being handled explicitly. Considering only the ASLP-TSRK results in Figure **??** we note a modest rise in the local error as the partition count is increased.

7 Conclusion

We have constructed a new class of N-partitioned, additive, two-step Runge-Kutta methods called stage-local partitioned two-step Runge-Kutta (SLP-TSRK) schemes, as well as a corresponding order condition theory. We presented two families of SLP-TSRK methods: asynchronous schemes that allow for the communication of ghost stage data to be masked behind the computation of the valid stage data; and locally implicit schemes that require the solution of only local (non-)linear systems and so avoid costly communication in the solver. We discussed the stability of SLP-TSRK schemes, and presented both an asynchronous and locally implicit SLP-TSRK that have been optimized for stability in some sense. We have further verified numerically, the theoretical properties of these schemes.

Future work will focus on specific implementation details, such as a variable step-size formulation and continuous output, as well as study the computational impact of the local implicitness and asynchronicity of the methods in a large scale parallel setting.

Acknowledgements LLNL-JRNL-865529 – This work was performed under the auspices of the U.S. Department of Energy by LLNL under contracts DE-AC52-07NA27344, and was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration, as well as the DOE Office of Applied Scientific Computing Research (ASCR) Mathematical Multifaceted Integrated Capabilities Center (MMICCs) Program under Grant DE-SC0023164. The authors are thankful to the anonymous referees for their comments and streamlining of Theorem A.1.

Appendix A Bounding Powers of $\mathscr{A}$

Theorem A.1 *Consider the matrix*

$$\mathscr{A} = \begin{pmatrix} 0 & I & 0 & 0 \\ 0 & 0 & U & V \\ 0 & 0 & 0 & I \\ 0 & 0 & \Theta & I-\Theta \end{pmatrix}, \tag{A.1}$$

where $\Theta \in \mathbb{R}^{N\times N}$ is a diagonal matrix with entries $\theta_j \in (-1,1]$, then powers of $\mathscr{A}^\mu$ take the form

$$\mathscr{A}^\mu = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \alpha_{1,1}^{(\mu)} & \alpha_{1,2}^{(\mu)} \\ \mathbf{0} & \mathbf{0} & \alpha_{2,1}^{(\mu)} & \alpha_{2,2}^{(\mu)} \\ \mathbf{0} & \mathbf{0} & \beta_{1,1}^{(\mu)} & \beta_{1,2}^{(\mu)} \\ \mathbf{0} & \mathbf{0} & \beta_{2,1}^{(\mu)} & \beta_{2,2}^{(\mu)} \end{bmatrix},$$

with $\alpha_{k,l}^{(\mu)}$ and $\beta_{k,l}^{(\mu)}$ uniformly bounded. (The streamlined version of the proof below was suggested by one of the anonymous referees.)

Proof Consider the partitioning of $\mathscr{A}$ as

$$\mathscr{A} = \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix}, \tag{A.2}$$

with

$$A_{11} = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix}, \quad A_{12} = \begin{bmatrix} 0 & 0 \\ U & V \end{bmatrix}, \quad A_{22} = \begin{bmatrix} 0 & I \\ \Theta & I-\Theta \end{bmatrix}, \tag{A.3}$$

then it can be shown inductively that

$$\mathscr{A}^\mu = \begin{bmatrix} 0 & (A_{11}A_{12} + A_{12}A_{22})A_{22}^{\mu-1} \\ 0 & A_{22}^\mu \end{bmatrix}, \tag{A.4}$$

for integer $\mu > 1$, where

$$A_{22} = \begin{bmatrix} S_\mu & T_\mu \\ S_{\mu+1} & T_{\mu+1} \end{bmatrix}, \quad \begin{cases} S_\mu &= (I+\Theta)^{-1}\left[\Theta + (-\Theta)^\mu\right] \\ T_\mu &= (I+\Theta)^{-1}\left[I - (-\Theta)^\mu\right] \end{cases} \tag{A.5}$$

Powers of A_{22} and thus $\mathscr{A}$, are bounded whenever those of Θ are, i.e., if $\theta_j \in (-1,1]$.

Compliance with Ethical Standards

The authors have no relevant financial or non-financial interests to disclose.

References

1. Aditya, K., Donzis, D.A.: High-order asynchrony-tolerant finite difference schemes for partial differential equations. Journal of Computational Physics **350**, 550–572 (2017)
2. Anderson, R., Andrej, J., Barker, A., Bramwell, J., Camier, J.S., Dobrev, J.C.V., Dudouit, Y., Fisher, A., Kolev, T., Pazner, W., Stowell, M., Tomov, V., Akkerman, I., Dahm, J., Medina, D., Zampini, S.: MFEM: A modular finite element methods library. Computers & Mathematics with Applications **81**, 42–74 (2021). DOI 10.1016/j.camwa.2020.06.009
3. Chung, W.T., Ma, P.C., Ihme, M.: Examination of diesel spray combustion in supercritical ambient fluid using large-eddy simulations. International Journal of Engine Research **21**(1), 122–133 (2020)
4. Donzis, D.A., Aditya, K.: Asynchronous finite-difference schemes for partial differential equations. Journal of Computational Physics **274**, 370–392 (2014)
5. Gad, A.F.: PyGAD: An intuitive genetic algorithm python library. arXiv preprint arXiv:2106.06158 (2021)

6. Hairer, E., Wanner, G.: Order conditions for general two-step Runge-Kutta methods. SIAM Journal on Numerical Analysis **34**(6), 2087–2089 (1997). URL http://www.jstor.org/stable/2951940
7. Jackiewicz, Z.: General linear methods for ordinary differential equations. Wiley Online Library (2009)
8. Kuhn, K., Lang, J.: Comparison of the asymptotic stability for multirate Rosenbrock methods. Journal of Computational and Applied Mathematics **262**, 139–149 (2014). DOI https://doi.org/10.1016/j.cam.2013.07.030. URL https://www.sciencedirect.com/science/article/pii/S0377042713003749. Selected Papers from NUMDIFF-13
9. Kværnø, A.: Stability of multirate Runge-Kutta schemes. International Journal of Differential Equations and Applications [electronic only] **1** (2000)
10. Lee, K., Bhattacharya, R., Gupta, V.: A switched dynamical system framework for analysis of massively parallel asynchronous numerical algorithms. In: 2015 American Control Conference (ACC), pp. 1095–1100. IEEE (2015)
11. Lorenz, E.N.: Predictability – a problem partly solved. In: Predictability of Weather and Climate. Cambridge University Press (2006). URL http://dx.doi.org/10.1017/CBO9780511617652.004
12. Sarshar, A., Tranquilli, P., Pickering, B., McCall, A., Roy, C.J., Sandu, A.: A numerical investigation of matrix-free implicit time-stepping methods for large cfd simulations. Computers & Fluids **159**, 53–63 (2017)
13. Savcenco, V.: Comparison of the asymptotic stability properties for two multirate strategies. Journal of Computational and Applied Mathematics **220**(1), 508–524 (2008). DOI https://doi.org/10.1016/j.cam.2007.09.005. URL https://www.sciencedirect.com/science/article/pii/S0377042707004839
14. Tracogna, S., Welfert, B.: Two-step Runge-Kutta: Theory and practice. BIT Numerical Mathematics (2000). DOI 10.1023/A:1022352704635
15. Tranquilli, P., Sandu, A.: Rosenbrock–Krylov methods for large systems of differential equations. SIAM Journal on Scientific Computing **36**(3), A1313–A1338 (2014)
16. Zharovsky, E., Sandu, A., Zhang, H.: A class of implicit-explicit two-step Runge–Kutta methods. SIAM Journal on Numerical Analysis **53**(1), 321–341 (2015). DOI 10.1137/130937883. URL https://doi.org/10.1137/130937883