Linear Algebra and its Applications 622 (2021) 66–103

Contents lists available at ScienceDirect

Linear Algebra and its Applications

www.elsevier.com/locate/laa

A quadrature framework for solving Lyapunov and Sylvester equations

Christian Bertram [*], Heike Faßbender

Institute for Numerical Analysis, TU Braunschweig, Universitätsplatz 2, 38106 Braunschweig, Germany

A R T I C L E I N F O

Article history:
Received 15 March 2019
Accepted 20 March 2021
Available online 24 March 2021
Submitted by V. Mehrmann

MSC:
65F30
15A24
93A15

Keywords:
Lyapunov equation
Sylvester equation
Alternating direction method
Runge-Kutta method

A B S T R A C T

This paper introduces a novel framework for the solution of (large-scale) Lyapunov and Sylvester equations derived from numerical integration methods. Suitable systems of ordinary differential equations are introduced. Low rank approximations of their solutions are produced by Runge-Kutta methods. Appropriate Runge-Kutta methods are identified following the idea of geometric numerical integration to preserve a geometric property, namely a low rank residual. For both types of equations we prove the equivalence of one particular instance of the resulting algorithm to the well known ADI iteration.

© 2021 Elsevier Inc. All rights reserved.

1. Introduction

The numerical approximation of the solution of the continuous Lyapunov equation

$$A\mathcal{P} + \mathcal{P}A^\mathsf{T} + BB^\mathsf{T} = 0 \tag{1.1}$$

[*] Corresponding author.
E-mail address: ch.bertram@tu-braunschweig.de (C. Bertram).

https://doi.org/10.1016/j.laa.2021.03.029
0024-3795/© 2021 Elsevier Inc. All rights reserved.

has been considered to great extent in the literature, see, e.g. the recent survey [31] and the references therein. Here a new framework based on methods for the numerical integration of ordinary differential equations (ODEs) is presented. Our main focus is on providing new insights into well-established methods, not on developing competitive new numerical methods for solving (1.1). We will consider (1.1) for $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$, $m \leq n$. The equation (1.1) has a unique symmetric positive definite solution $\mathcal{P}$ if all eigenvalues of A are in the open left half-plane $\mathbb{C}_-$ (that is, if A is stable). In that case, the analytic solution can be written as

$$\mathcal{P} = \int_0^\infty e^{At}BB^\mathsf{T}e^{A^\mathsf{T}t}\,\mathrm{d}t \in \mathbb{R}^{n\times n}, \tag{1.2}$$

see, e.g., [22]. Lyapunov equations play an important role in control and systems theory, see, e.g. [1,10,12]. In the context of linear time-invariant systems $\dot{x} = Ax + Bu$, the solution $\mathcal{P}$ of (1.1) is called the controllability Gramian. It measures the energy transfer in the system.

We will also consider Sylvester equations

$$A\mathcal{Y} - \mathcal{Y}B - FG^\mathsf{T} = 0 \tag{1.3}$$

with given $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{m\times m}$, $F \in \mathbb{R}^{n\times r}$ and $G \in \mathbb{R}^{m\times r}$. The solution $\mathcal{Y}$ is unique when the spectra of A and B are disjunct, i.e. $\sigma(A) \cap \sigma(B) = \emptyset$, see, e.g. [22].

In particular, we will be concerned with (1.1) and (1.3) for large and sparse system matrices and a low rank initial residual, that is for the Lyapunov equation BB^T and for the Sylvester equation FG^T is of low rank. In case of (1.1) $m \ll n$ for large n will automatically yield a low rank residual. In that case, the symmetric positive definite solution $\mathcal{P}$ of (1.1) can be approximated by a low rank approximation in the sense that $\mathcal{P} \approx ZZ^\mathsf{T}$ with a rectangular $n \times N$ matrix Z and $N \ll n$, [23,2]. Z is often called a low rank Cholesky factor, even so Z is not a square lower triangular matrix. In a similar fashion, if FG^T is of low rank, the solution of the Sylvester equation can be approximated by $\mathcal{Y} \approx \hat{Z}\Gamma\check{Z}^\mathsf{T}$ with rectangular $n \times N$ matrices $\hat{Z}$, $\check{Z}$ with $N \ll n$ and a diagonal matrix Γ, see, e.g. [6,14].

In the following we give an overview of methods important or related to our later discussion. For a more exhaustive survey of methods for the solution of various linear matrix equations we refer to [31].

A popular algorithm for deriving low rank Cholesky factors for Lyapunov and Sylvester equations is the alternating direction implicit (ADI) method. It was developed to solve linear systems of equations in [28] and modified to approximate the solution $\mathcal{P}$ of Lyapunov equations in [24] (see [3, Sec. 7] for a short or [21, Ch. 3.1-3.2.1] for a more detailed derivation). In [23] the iteration was reformulated such that low rank approximations $Z_jZ_j^\mathsf{T}$ to the solution $\mathcal{P}$ are generated. This yields the computationally more efficient ADI-variant called Cholesky factor ADI (CF-ADI) algorithm. Clearly, an approximate solution P_j will not satisfy (1.1) exactly, a nonzero residual

$$\mathcal{L}(P_j) = AP_j + P_j A^\mathsf{T} + BB^\mathsf{T} \tag{1.4}$$

will remain. This residual can be used to determine convergence of the iterative process. In [21, Alg. 3.2], the ADI iteration was further manipulated in order to allow for a fast evaluation of the residual norm $\|\mathcal{L}(P_j)\|$. This is known as the residual-based ADI method. An alternative derivation of this formulation utilizing Krylov subspaces can be found in [33]. The ADI iteration was also adapted to Sylvester equations, see [6], [21, Ch. 3.3].

Another type of methods for the solution of Lyapunov equations is making use of empirical Gramians [25]. The empirical Gramian essentially involves a sum approximation of the integral (1.2) $\mathcal{P} = \sum_j \delta_j g(t_j)$ for $g(t) = e^{At}BB^\mathsf{T}e^{A^\mathsf{T}t}$, arbitrary times t_j and appropriate quadrature weights δ_j. Usually, the identity $g(t) = h(t)h(t)^\mathsf{T}$ for $h(t) = e^{At}B$ is used to determine $g(t_j)$. In doing so, $h(t)$ is not computed directly, but as the solution of the ordinary differential equation (ODE) $\frac{\mathrm{d}}{\mathrm{d}t}h(t) = Ah(t)$ with initial value $h(0) = B$. Any numerical integration scheme can be used to do so. Related quadrature approaches are discussed, e.g., in [30,32]. Empirical Gramians are mainly used in model order reduction via balanced proper orthogonal decomposition (POD), see [29].

The ADI iteration and the quadrature-based methods have been connected to rational Krylov subspaces and moment matching, see [11,27,33]. In [27] quadrature methods with complex time step sizes for the approximation of empirical Gramians were analyzed. Further the stability function of certain multi-stage implicit methods was connected to the (complex) interpolation points used in rational interpolation.

In this paper for the Lyapunov case we utilize the time-dependent Gramian

$$P(t) = \int_0^t e^{A\tau}BB^\mathsf{T}e^{A^\mathsf{T}\tau}d\tau.$$

For $t \to \infty$, $P(t)$ will approximate the solution $\mathcal{P}$ of the Lyapunov equation. We will make use of the fact that $P(t)$ can be interpreted as the solution of a certain system of ODEs. It turns out that also in the context of Sylvester equations we can state a useful system of ODEs. Runge-Kutta methods are employed to derive algorithms for the low rank approximation of the solution of Lyapunov and Sylvester equations. In the Lyapunov case, neither $P(t)$ nor the iterates P_j from the Runge-Kutta methods will exactly satisfy the Lyapunov equation. We will observe that $\mathcal{L}(P(t)) = AP(t) + P(t)A^\mathsf{T} + BB^\mathsf{T} = h(t)h(t)^\mathsf{T}$ for $h(t) = e^{At}B$. Our key idea is to use only those Runge-Kutta methods which lead to iterates P_j with conformable low rank Lyapunov residuals. Thus, we use ideas from geometric numerical integration, where qualitative properties (e.g. algebraic invariants) of the solution are preserved instead of fulfilling quantitative properties (e.g. small errors), cf. [19, Ch. 5], [15]. Herewith we derive a residual based iteration which turns out to be equivalent to the ADI iteration. By making use of the stability function of a Runge-Kutta method it will be shown further that these methods are equivalent to DIRK methods. Hence our approach results in a novel derivation of the ADI iteration and a new interpretation of the ADI iteration in terms of geometric numerical integration.

The paper is organized as follows. In the next section we introduce notation, review basic properties of Runge-Kutta methods for numerical integration and give a short introduction to the ADI iteration for the solution of Lyapunov equations. Section 3 deals with the Lyapunov equation and its numerical solution. In Section 3.1 we present the first algorithm for computing a low rank approximation to the time-dependent Gramian $P(t)$ by means of a Runge-Kutta method. Clearly, $P(t)$ will not exactly satisfy the Lyapunov equation. In Section 3.2, we derive an expression for the Lyapunov residual $\mathcal{L}(P(T))$. This turns out to be of low rank for $m \leq n$. We propose to use only those Runge-Kutta methods which yield iterates satisfying the same kind of Lyapunov residual as $P(t)$. Conditions for appropriate Runge-Kutta methods leading to such iterates are given. In Section 3.3 the usual approach of using the same Runge-Kutta method in each iteration step is relaxed in order to allow for the use of different Runge-Kutta methods in each iteration step. Section 3.4 deals with the equivalence of a certain instance of the resulting algorithm to the CF-ADI iteration. Next, in Section 3.5 the appropriate Runge-Kutta methods are further characterized by means of their stability functions. As discussed in Section 3.6, it turns out that the appropriate methods are essentially determined by s parameters. In Section 3.7 the choice of (complex-valued) shifts is discussed. A seemingly different approach to solve Lyapunov equations presented in [34] is connected to our geometric approach and the ADI iteration in Section 3.8. Finally, in Section 3.9, we demonstrate the effectiveness of our framework with numerical experiments. The ideas for the solution of Lyapunov equations are transferred to the Sylvester equation in Section 4. The paper ends with some concluding remarks in Section 5.

2. Preliminaries

In this section we will introduce some notation used in the following as well as briefly recall Runge-Kutta methods for the numerical integration of ordinary differential equations. Moreover, the ADI method for solving Lyapunov equations (1.1) is reviewed.

The set of complex numbers with positive (negative) real part will be denoted by $\mathbb{C}_+$ ($\mathbb{C}_-$). The positive (negative) real numbers will be denoted by $\mathbb{R}_+$ ($\mathbb{R}_-$).

We will frequently make use of the Kronecker product of two matrices as well as the vectorization of a matrix, see, e.g., [18,13] for a more complete discussion. If X is an $r \times s$ matrix and Y is a $p \times q$ matrix, then the Kronecker product $X \otimes Y$ is the $rp \times sq$ block matrix

$$X \otimes Y = \begin{bmatrix} x_{11}Y & \cdots & x_{1s}Y \\ \vdots & \ddots & \vdots \\ x_{r1}Y & \cdots & x_{rs}Y \end{bmatrix}.$$

If X and Y are regular, then the property

$$(X \otimes Y)^{-1} = X^{-1} \otimes Y^{-1}$$

holds. Other useful Kronecker product properties are

$$\begin{aligned}(X \otimes Y)(V \otimes W) &= XV \otimes YW,\\ P(X \otimes Y)Q^{\mathsf{T}} &= Y \otimes X,\end{aligned} \tag{2.1}$$

for suitable V, W and the perfect shuffle permutation matrices P and Q, see, e.g. [13, Ch. 1.3.6]. In particular, for square matrices X and Y (that is, $r = s$ and $p = q$), we have $P = Q = I_{rp}([(1{:}p{:}rp)\ (2{:}p{:}rp)\ \dots\ (p{:}p{:}rp)], :)$ where I_{rp} denotes the $rp \times rp$ identity matrix and MATLAB® colon notation is used to specify the arrangement of the rows of I_{rp}. In case X or Y is a vector, the corresponding perfect shuffle permutation matrix is the identity: $P(X \otimes Y)I_q = Y \otimes X$ for $s = 1$.

The vectorization of a matrix converts the matrix into a column vector. For a $r \times s$ matrix X, $\mathrm{vec}(X)$ denotes the $rs \times 1$ column vector obtained by stacking the columns of the matrix X on top of one another:

$$\mathrm{vec}(X) = [x_{11}, \dots, x_{r1}, x_{12}, \dots, x_{r2}, \dots, x_{1s}, \dots, x_{rs}]^{\mathsf{T}}.$$

The vectorization and the Kronecker product can be used to express matrix multiplication as a linear transformation on matrices. In particular,

$$\mathrm{vec}(XYZ) = (Z^{\mathsf{T}} \otimes X)\,\mathrm{vec}(Y)$$

for matrices X, Y, and Z of dimensions $r \times s$, $s \times t$, and $t \times v$. Thus we can rewrite the Sylvester equation (1.3) in the form

$$(I_n \otimes A - B^{\mathsf{T}} \otimes I_n)\,\mathrm{vec}(\mathcal{Y}) = \mathrm{vec}(FG^{\mathsf{T}}),$$

where I_n is the $n \times n$ identity matrix. In this form, the equation can be seen as a linear system of equations of dimension $n^2 \times n^2$. From this it is fairly straightforward to see that a unique solution $\mathcal{Y}$ exists for all FG^{T} if and only if A and B have no common eigenvalues, see, e.g., [18,22].

2.1. Numerical integration

There are numerous methods for the numerical solution of ordinary differential equations of the type

$$\frac{\mathrm{d}}{\mathrm{d}t} y(t) = f(t, y(t)), \quad y(0) = y_0, \tag{2.2}$$

see, e.g., [16,17]. Here $f\colon \mathbb{R}\times\mathbb{R}^n \to \mathbb{R}^n$ is a given function and $y_0 \in \mathbb{R}^n$ is a given initial value. One is interested in computing the function $y\colon \mathbb{R} \to \mathbb{R}^n$ in the interval $[0, t_{\text{end}}]$. Single-step methods make use of the fact that

$$y(t_j) = y(t_{j-1}) + \int_{t_{j-1}}^{t_j} f(t, y(t))\,\mathrm{d}t$$

holds for $j = 1, 2, \dots, N$ and $t_0 = 0 < t_1 < t_2 < \cdots < t_N = t_{\text{end}}$ in order to compute approximate solutions $y_j \approx y(t_j)$.

We will consider s-stage Runge-Kutta methods (see, e.g., [9,15–17]) which are defined via

$$y_j = y_{j-1} + \omega_j \sum_{i=1}^{s} \beta_i k_i^{(j)}, \qquad j = 1, \dots, N, \tag{2.3}$$

$$k_i^{(j)} = f\Big(t_{j-1} + \gamma_i \omega_j,\, y_{j-1} + \omega_j \sum_{\ell=1}^{s} \lambda_{i\ell} k_\ell^{(j)}\Big), \qquad i = 1, \dots, s, \tag{2.4}$$

for certain $\beta_i \in \mathbb{C}$, $\gamma_i \in \mathbb{R}$, $i = 1, \dots, s$ and $\lambda_{i\ell} \in \mathbb{C}$, $i, \ell = 1, \dots, s$. Please note that we allow for complex-valued $\lambda_{i\ell}$ and β_i unlike the standard definition of Runge-Kutta methods for the solution of (2.2). Moreover, $\omega_j := t_j - t_{j-1} > 0$, $j = 1, \dots, N$, denotes the time step size. Often Runge-Kutta methods are given in short hand by the so called Butcher tableau

$$\begin{array}{c|c} \gamma & \Lambda \\ \hline & \beta^{\mathsf{T}} \end{array} = \begin{array}{c|cccc} \gamma_1 & \lambda_{11} & \lambda_{12} & \dots & \lambda_{1s} \\ \gamma_2 & \lambda_{21} & \lambda_{22} & \dots & \lambda_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \gamma_s & \lambda_{s1} & \lambda_{s2} & \dots & \lambda_{ss} \\ \hline & \beta_1 & \beta_2 & \dots & \beta_s \end{array} \tag{2.5}$$

with $\Lambda \in \mathbb{C}^{s\times s}$, $\beta \in \mathbb{C}^s$ and $\gamma \in \mathbb{R}^s$.

If in the Butcher tableau Λ is a strict lower triangular matrix, then the $k_i^{(j)}$ can be calculated explicitly one after another. Otherwise they are only defined implicitly and a system of (in general nonlinear) equations with sn unknowns has to be solved to obtain them. For explicit Runge-Kutta methods the region of absolute stability is small and bounded. On the other hand, implicit Runge-Kutta methods have much larger regions of absolute stability and the time step size can be chosen based on the desired accuracy, not due to stability constraints. In order to avoid the high computational costs for general implicit Runge-Kutta methods, often so-called diagonal implicit Runge-Kutta (DIRK) methods are used, where Λ is a lower triangular matrix [20]. This uncouples the system of equations to be solved into a sequence of s systems.

The function

$$R(z) = 1 + z\beta^{\mathsf{T}}(I_s - z\Lambda)^{-1}\mathbb{1}_s = \frac{\det(I_s - z(\Lambda - \mathbb{1}_s\beta^{\mathsf{T}}))}{\det(I_s - z\Lambda)} \tag{2.6}$$

is called the stability function of the Runge-Kutta method given by (2.5). Here, $\mathbb{1}_s$ denotes the vector $\mathbb{1}_s = (1, \ldots, 1)^{\mathsf{T}}$ consisting of s ones. When a Runge-Kutta method is applied to the linear differential equation $y' = \lambda y$ the iteration is given by $y_k = R(z)y_{k-1}$ with $z = \omega\lambda$. The method is said to be A-stable if all z with $\operatorname{Re}(z) < 0$ are in the domain of absolute stability, that is the set of all $z = \omega\lambda$ with $|R(z)| < 1$.

2.2. The ADI method

Here we introduce the ADI iteration for the Lyapunov equation based on [21,5]. The goal is to approximate the $n \times n$ solution $\mathcal{P}$ of the Lyapunov equation (1.1) in factored form $ZZ^{\mathsf{T}} \approx \mathcal{P}$, where $Z \in \mathbb{R}^{n \times mN}$ with $mN \ll n$.

The ADI iteration for computing the solution of (1.1) is given by

$$\begin{aligned} X_0 &= 0, \\ (A + \alpha_j I_n)X_{j-\frac{1}{2}} &= -BB^{\mathsf{T}} - X_{j-1}(A^{\mathsf{T}} - \alpha_j I_n), \\ (A + \alpha_j I_n)X_j^{\mathsf{H}} &= -BB^{\mathsf{T}} - X_{j-\frac{1}{2}}^{\mathsf{H}}(A^{\mathsf{T}} - \alpha_j I_n), \end{aligned}$$

with complex shift parameters $\alpha_1, \ldots, \alpha_N \in \mathbb{C}_-$. For a certain choice of shift parameters the iterates X_j will converge to $\mathcal{P}$. Reformulating this iteration into a single step, writing the iterates $X_j = Z_j Z_j^{\mathsf{H}}$ in factored form and applying some algebraic manipulations (see [21, Ch. 3.2] for the details) one obtains the iteration

$$\begin{aligned} V_1 &= (A + \alpha_1 I_n)^{-1}B, \ Z_1 = \sqrt{-2\operatorname{Re}(\alpha_1)}V_1, \\ V_j &= V_{j-1} - (\alpha_j + \overline{\alpha_{j-1}})(A + \alpha_j I_n)^{-1}V_{j-1}, \\ Z_j &= \left[Z_{j-1}, \sqrt{-2\operatorname{Re}(\alpha_j)}V_j\right]. \end{aligned}$$

With the findings from [21, Ch. 3.2.4] respectively, the iteration results in Algorithm 1. Please note that Z_j grows in each iteration step by a block of m columns.

As the shift parameters α_j are complex numbers, the iterates V_j (and thus Z_j) are complex-valued matrices. This can be avoided if the set of shift parameters is proper, i.e. closed under complex conjugation such that complex shift parameters appear in pairs with their complex conjugated version, see, e.g., the realification approach from [4] and its revised form in [21, Ch. 4.1.4].

We refrain from stating the ADI iteration for solving the Sylvester equation (1.3) as we will not make explicit use of it. Please see, e.g., [21] for a detailed description of a

Algorithm 1 Low rank ADI iteration [21, Alg. 3.2, $E = I_n$].

```
Input: $A \in \mathbb{R}^{n\times n}$ stable, $B \in \mathbb{R}^{n\times m}$, parameters $\{\alpha_1, \dots, \alpha_N\} \in \mathbb{C}_-$
Output: $Z \in \mathbb{C}^{n\times mN}$ with $ZZ^{\mathsf{H}} \approx \mathcal{P}$
1: initialize $W_0 = B$, $Z_0 = [\ ]$
2: for $j = 1, \dots, N$ do
3:     solve $(A + \alpha_j I_n)V_j = W_{j-1}$ for $V_j$
4:     $W_j = W_{j-1} - 2\,\mathrm{Re}(\alpha_j)V_j$
5:     update $Z_j = [Z_{j-1}, \sqrt{-2\,\mathrm{Re}(\alpha_j)}V_j]$
6: end for
7: $Z = Z_N$
```

residual based variant or [31] for an outline of the development of the ADI iteration for Sylvester equations.

3. Lyapunov equation

In this section we will derive an ODE-based approximation of the solution $\mathcal{P}$ (1.2) of the Lyapunov equation (1.1). The ODE will be solved via a Runge-Kutta method. The equivalence of our method to the ADI method Algorithm 1 for certain special Runge-Kutta methods will be discussed.

For reasons of simplicity we will not consider (1.1) in full generality, only matrices $B = b \in \mathbb{R}^{n\times 1}$ with one column will be considered. The general case $B = [b_1, \dots, b_m] \in \mathbb{R}^{n\times m}$ can be reduced to

$$\begin{aligned}\mathcal{P} &= \int_0^\infty e^{At}BB^{\mathsf{T}}e^{A^{\mathsf{T}}t}\,\mathrm{d}t \\ &= \sum_{i=1}^{m}\int_0^\infty e^{At}b_ib_i^{\mathsf{T}}e^{A^{\mathsf{T}}t}\,\mathrm{d}t.\end{aligned} \tag{3.1}$$

Thus, our results extend to the case $m > 1$ easily.

We will make use of the time dependent Gramian $P(t)$ which is given by

$$P(t) := \int_0^t e^{A\tau}bb^{\mathsf{T}}e^{A^{\mathsf{T}}\tau}\,\mathrm{d}\tau = \int_0^t h(\tau)h(\tau)^{\mathsf{T}}\,\mathrm{d}\tau$$

where $h(t) := e^{At}b$. The functions $P(t)$ and $h(t)$ are the solutions of the system of ODEs

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}P(t) &= h(t)h(t)^{\mathsf{T}}, \quad & P(0) &= 0,\\ \frac{\mathrm{d}}{\mathrm{d}t}h(t) &= Ah(t), & h(0) &= b.\end{aligned} \tag{3.2}$$

Vectorizing (3.2) allows to write the system in the form (2.2)

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\operatorname{vec}(P(t))\\ h(t)\end{bmatrix} = \begin{bmatrix}\operatorname{vec}(h(t)h(t)^\mathsf{T})\\ Ah(t)\end{bmatrix} = f(t, y(t)) \tag{3.3}$$

for the high dimensional solution function $y(t) = [\operatorname{vec}(P(t))^\mathsf{T} \;\; h(t)^\mathsf{T}]^\mathsf{T}$, $y\colon \mathbb{R} \to \mathbb{R}^{n^2+n}$ which needs to be determined. Clearly, $y(0) = [0^\mathsf{T} \; b^\mathsf{T}]^\mathsf{T}$.

3.1. Approximating $\mathcal{P}$ by Runge-Kutta methods

In order to solve (3.3), we will make use of a Runge-Kutta method with tableau (2.5). For ease of notation, we will use $y(t) = [v(t)^\mathsf{T} \; h(t)^\mathsf{T}]^\mathsf{T}$, that is, $v(t) = \operatorname{vec}(P(t))$. Moreover, the vector $k_i^{(j)}$ (2.4) is written in block form

$$k_i^{(j)} = \begin{bmatrix}\tilde{k}_i^{(j)}\\ \hat{k}_i^{(j)}\end{bmatrix}$$

corresponding to the two blocks of $y(t) = [v(t)^\mathsf{T} \; h(t)^\mathsf{T}]^\mathsf{T}$. We obtain

$$\begin{aligned} v_j &= v_{j-1} + \omega_j \sum_{i=1}^{s} \beta_i \tilde{k}_i^{(j)}, \qquad v_0 = 0 \in \mathbb{R}^{n^2},\\ h_j &= h_{j-1} + \omega_j \sum_{i=1}^{s} \beta_i \hat{k}_i^{(j)}, \qquad h_0 = b \in \mathbb{R}^{n}, \end{aligned} \tag{3.4}$$

with

$$\begin{aligned} k_i^{(j)} = \begin{bmatrix}\tilde{k}_i^{(j)}\\ \hat{k}_i^{(j)}\end{bmatrix} &= f\left(t_{j-1} + \gamma_i\omega_j, \begin{bmatrix} v_{j-1} + \omega_j \sum_{\ell=1}^{s} \lambda_{i\ell}\tilde{k}_\ell^{(j)}\\ h_{j-1} + \omega_j \sum_{\ell=1}^{s} \lambda_{i\ell}\hat{k}_\ell^{(j)}\end{bmatrix}\right)\\ &= \begin{bmatrix}\operatorname{vec}\left((h_{j-1} + \omega_j \sum_{\ell=1}^{s} \lambda_{i\ell}\hat{k}_\ell^{(j)})(h_{j-1} + \omega_j \sum_{\ell=1}^{s} \lambda_{i\ell}\hat{k}_\ell^{(j)})^\mathsf{H}\right)\\ Ah_{j-1} + \omega_j \sum_{\ell=1}^{s} \lambda_{i\ell} A\hat{k}_\ell^{(j)}\end{bmatrix}. \end{aligned} \tag{3.5}$$

Please note the use of H instead of T in the expression above due to the possibly complex valued λ_{ij}. Moreover, note that the expression in (3.5) does not involve $\tilde{k}_i^{(j)}$.

Remark 1. Another idea to solve (3.2) is to employ two different s-stage Runge-Kutta methods, e.g., a Butcher tableau with Λ and β for $h(t)$ and a second Butcher tableau with $\hat{\Lambda}$ and $\hat{\beta}$ for $P(t)$. The only difference to (3.4) and (3.5) as above is that in (3.4) the equation for v_j changes to $v_j = v_{j-1} + \omega_j \sum_{i=1}^{s} \hat{\beta}_i \tilde{k}_i^{(j)}$. The matrix $\hat{\Lambda}$ does not appear as the right hand side of $P'(t) = h(t)h(t)^\mathsf{T}$ does not depend on $P(t)$. We do not consider this any further here, as it would turn out in Section 3.2 that $\hat{\beta}$ has to be chosen as $\hat{\beta} = \beta$ to preserve an algebraic invariant.

De-vectorizing the first equation of (3.4) we obtain the iteration

$$\begin{aligned} P_j &= P_{j-1} + \omega_j \sum_{i=1}^{s} \beta_i \mathfrak{h}_i^{(j)} (\mathfrak{h}_i^{(j)})^{\mathsf{H}}, \qquad j = 1, \dots, N, \\ h_j &= h_{j-1} + \omega_j \sum_{i=1}^{s} \beta_i \hat{k}_i^{(j)}, \end{aligned} \tag{3.6}$$

with $P_0 = 0 \in \mathbb{R}^{n \times n}$, $h_0 = b \in \mathbb{R}^n$ and

$$\begin{aligned} \mathfrak{h}_i^{(j)} &= h_{j-1} + \omega_j \sum_{\ell=1}^{s} \lambda_{i\ell} \hat{k}_\ell^{(j)}, \\ \hat{k}_i^{(j)} &= A h_{j-1} + \omega_j \sum_{\ell=1}^{s} \lambda_{i\ell} A \hat{k}_\ell^{(j)} = A \mathfrak{h}_i^{(j)}, \end{aligned}$$

for $i = 1, \dots, s$. Please note that due to $\Lambda \in \mathbb{C}^{s \times s}$ and $\beta \in \mathbb{C}^s$ all iterates $\mathfrak{h}_i^{(j)}$, h_j as well as $\hat{k}_i^{(j)}$ will be complex valued vectors, while $P_j \in \mathbb{C}^{n \times n}$, $j = 1, \dots, N$.

First all $\hat{k}_i^{(j)}$, $i = 1, \dots, s$ need to be determined for a given j. Then the corresponding iterates $\mathfrak{h}_i^{(j)}$, P_j and h_j can be computed. Let $K_j = [\hat{k}_1^{(j)}, \dots, \hat{k}_s^{(j)}]$. Then we have

$$K_j = [A h_{j-1}, \dots, A h_{j-1}] + \omega_j A K_j \Lambda^{\mathsf{T}}. \tag{3.7}$$

Via vectorization (3.7) is reformulated as a linear system of equations of size $ns \times ns$

$$\begin{aligned} (I_{ns} - \omega_j (\Lambda \otimes A)) \operatorname{vec}(K_j) &= \left[(A h_{j-1})^{\mathsf{T}}, \dots, (A h_{j-1})^{\mathsf{T}} \right]^{\mathsf{T}} && (3.8) \\ &= (I_s \otimes A)(\mathbb{1}_s \otimes I_n) h_{j-1}. && (3.9) \end{aligned}$$

Let $\mu_1, \dots, \mu_s$ and $\lambda_1, \dots, \lambda_n$ be the eigenvalues of Λ and A respectively. Then the eigenvalues of $I_{ns} - \omega_j (\Lambda \otimes A)$ are given by $1 - \omega_j \mu_p \lambda_q$, $p = 1, \dots, s$, $q = 1, \dots, n$. Thus the solution of (3.8) is unique if and only if

$$\mu_p \neq \frac{1}{\omega_j \lambda_q} \tag{3.10}$$

for all $p = 1, \dots, s$ and $q = 1, \dots, n$. Thus, in case we require Λ to be chosen such that $\mu_p \in \mathbb{C}_+$, (3.8) has an unique solution.

An alternative to determining K_j is given by the relation $K_j = A \mathcal{H}_j$ with $\mathcal{H}_j = [\mathfrak{h}_1^{(j)}, \dots, \mathfrak{h}_s^{(j)}]$. This implies that

$$\mathcal{H}_j = [h_{j-1}, \dots, h_{j-1}] + \omega_j A \mathcal{H}_j \Lambda^{\mathsf{T}} \tag{3.11}$$

needs to be solved. One way to realize this is, e.g., by vectorizing the equation to transform it into a linear system of equations with the same coefficient matrix $I_{ns} - \omega_j (\Lambda \otimes A)$

as above. This system of equations can be solved uniquely if and only if the eigenvalues of Λ and A satisfy (3.10). In particular, note that, as for (3.8), we can not choose Λ arbitrarily, at least (3.10) has to hold.

With the help of K_j and $\mathcal{H}_j$ (3.6) can be written as

$$\begin{aligned} P_j &= P_{j-1} + \omega_j \mathcal{H}_j \operatorname{diag}(\beta) \mathcal{H}_j^{\mathsf{H}}, \qquad j = 1, \ldots, N, \\ h_j &= h_{j-1} + \omega_j K_j \beta, \end{aligned} \tag{3.12}$$

where $\beta \in \mathbb{C}^s$ is as in (2.5). As

$$P_j^{\mathsf{H}} = P_{j-1}^{\mathsf{H}} + \omega_j \mathcal{H}_j \operatorname{diag}(\overline{\beta}) \mathcal{H}_j^{\mathsf{H}},$$

P_j is only Hermitian in case $\beta \in \mathbb{R}^s$. If in addition $\beta \in \mathbb{R}^s_+$, then we can decompose $\omega_j \operatorname{diag}(\beta)$ into $\operatorname{diag}(\omega_j\beta)^{\frac{1}{2}} \operatorname{diag}(\omega_j\beta)^{\frac{1}{2}}$ where $\operatorname{diag}(\omega_j\beta)^{\frac{1}{2}} \in \mathbb{R}^{s\times s}$ denotes the diagonal matrix with diagonal entries $\sqrt{\omega_j\beta_i}$, $i = 1, \ldots, s$. This allows to express $\omega_j \mathcal{H}_j \operatorname{diag}(\beta) \mathcal{H}_j^{\mathsf{H}}$ as

$$\omega_j \mathcal{H}_j \operatorname{diag}(\beta) \mathcal{H}_j^{\mathsf{H}} = \left(\mathcal{H}_j \operatorname{diag}(\omega_j\beta)^{\frac{1}{2}}\right) \left(\mathcal{H}_j \operatorname{diag}(\omega_j\beta)^{\frac{1}{2}}\right)^{\mathsf{H}}.$$

Thus, for $\beta \in \mathbb{R}^s_+$, P_j is by construction a positive semi-definite matrix and can be expressed as $P_j = Z_j Z_j^{\mathsf{H}}$ for some complex valued matrix Z_j. Hence we have

$$\begin{aligned} Z_j Z_j^{\mathsf{H}} &= Z_{j-1} Z_{j-1}^{\mathsf{H}} + \omega_j \mathcal{H}_j \operatorname{diag}(\beta) \mathcal{H}_j^{\mathsf{H}} \\ &= \left[Z_{j-1}, \mathcal{H}_j \operatorname{diag}(\omega_j\beta)^{\frac{1}{2}}\right] \left[Z_{j-1}, \mathcal{H}_j \operatorname{diag}(\omega_j\beta)^{\frac{1}{2}}\right]^{\mathsf{H}} \end{aligned} \tag{3.13}$$

where $\operatorname{diag}(\omega_j\beta)^{\frac{1}{2}} \in \mathbb{C}^{s\times s}$ denotes the diagonal matrix with diagonal entries $\sqrt{\omega_j\beta_i}$, $i = 1, \ldots, s$.

Instead of iterating on P_j as in (3.12), the above observation allows us in case $\beta \in \mathbb{R}^s_+$ to iterate on the low rank factor

$$Z_j = [Z_{j-1}, \mathcal{H}_j \operatorname{diag}(\omega_j\beta)^{\frac{1}{2}}] \in \mathbb{C}^{n\times js}$$

which gains s additional columns in every iteration step. The procedure to obtain the Gramian approximation described in this section is summarized in Algorithm 2. We need to require $\beta \in \mathbb{R}^s_+$ such that P_j is positive semi-definite and that the eigenvalues of Λ satisfy (3.10) in order to ensure that all linear system solves have a unique solution. In case N is such that sN is less than n, then Z_j are low rank factors of P_j, $j = 1, \ldots, N$.

3.2. Runge-Kutta methods which preserve an algebraic invariant

Next we will take a closer look at all possible Butcher tableaus in order to derive additional conditions for suitable methods.

Algorithm 2 Low rank solution to (1.1) via an s-stage Runge-Kutta method.

```
Input: $A \in \mathbb{R}^{n\times n}$ stable, $b \in \mathbb{R}^n$, positive time step sizes $\{\omega_1, \dots, \omega_N\}$ and a Butcher tableau (2.5) with
    $\Lambda \in \mathbb{C}^{s\times s}$ and $\beta \in \mathbb{R}^s_+$ which satisfies (3.10)
Output: $Z \in \mathbb{C}^{n\times sN}$ with $ZZ^{\mathsf{H}} \approx \mathcal{P}$
1: initialize $h_0 = b$, $Z_0 = [\,]$
2: for $j = 1, \dots, N$ do
3:     solve $K_j = [Ah_{j-1}, \dots, Ah_{j-1}] + \omega_j A K_j \Lambda^{\mathsf{T}}$ for $K_j \in \mathbb{C}^{n\times s}$
4:     $\mathcal{H}_j = [h_{j-1}, \dots, h_{j-1}] + \omega_j K_j \Lambda^{\mathsf{T}}$
5:     update $Z_j = [Z_{j-1}, \mathcal{H}_j \operatorname{diag}(\omega_j \beta)^{\frac{1}{2}}]$
6:     $h_j = h_{j-1} + \omega_j K_j \beta$
7: end for
8: $Z = Z_N$
```

First, observe that the time dependent Gramian $P(t)$ will not exactly satisfy (1.1), a residual, the so-called Lyapunov residual, will remain

$$\mathcal{L}(t) := AP(t) + P(t)A^{\mathsf{T}} + bb^{\mathsf{T}}.$$

Next, reconsider the derivative of $P(t)$ (3.2)

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}P(t) &= h(t)h(t)^{\mathsf{T}}\\ &= h(0)h(0)^{\mathsf{T}} + \int_0^t \frac{\mathrm{d}}{\mathrm{d}\tau}h(\tau)h(\tau)^{\mathsf{T}}\,\mathrm{d}\tau\\ &= bb^{\mathsf{T}} + \int_0^t \left(Ah(\tau)h(\tau)^{\mathsf{T}} + h(\tau)h(\tau)^{\mathsf{T}}A^{\mathsf{T}}\right)\mathrm{d}\tau\\ &= bb^{\mathsf{T}} + A\int_0^t h(\tau)h(\tau)^{\mathsf{T}}\,\mathrm{d}\tau + \int_0^t h(\tau)h(\tau)^{\mathsf{T}}\,\mathrm{d}\tau A^{\mathsf{T}}\\ &= bb^{\mathsf{T}} + AP(t) + P(t)A^{\mathsf{T}}.\end{aligned}$$

Thus we have

$$\mathcal{L}(t) = AP(t) + P(t)A^{\mathsf{T}} + bb^{\mathsf{T}} = h(t)h(t)^{\mathsf{T}}, \tag{3.14}$$

for all t. Obviously, due to the right-hand side, the Lyapunov residual is of rank one.

Our key idea is to use only those Butcher tableaus which guarantee that the iterates $P_j = Z_j Z_j^{\mathsf{H}}$ and h_j satisfy the algebraic invariant (3.14) in the sense

$$AP_j + P_jA^{\mathsf{T}} + bb^{\mathsf{T}} = h_j h_j^{\mathsf{H}}. \tag{3.15}$$

Please note that as $\Lambda \in \mathbb{C}^{s\times s}$ and $\beta \in \mathbb{C}^s$, our iterates P_j and h_j may be complex valued and thus we need to modify T in (3.14) to H in (3.15).

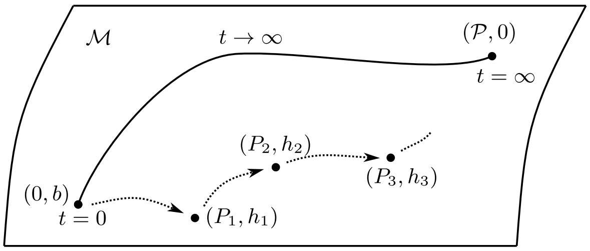

Fig. 1. Solution of $\dot{h} = Ah$, $\dot{P} = hh^{\mathsf{T}}$ and iterates from Algorithm 2 evolving on the rank-one residual manifold $\mathcal{M}$.

Before we discuss which Butcher tableaus allow for (3.15) let us give an interpretation of (3.14) and (3.15). Consider all tuples (P, h), for which the invariant (3.15) is satisfied. They are located on the manifold

$$\mathcal{M} := \left\{(P, h) \text{ with } P \in \mathbb{C}^{n\times n},\ h \in \mathbb{C}^{n\times 1} \mid AP + PA^{\mathsf{T}} + bb^{\mathsf{T}} - hh^{\mathsf{H}} = 0\right\}. \tag{3.16}$$

The solution $(P(t), h(t))$ of (3.2) lies on $\mathcal{M}$ for all times $t \in \mathbb{R}$ because of (3.14). As the Lyapunov residual (3.14) is of rank one we will call $\mathcal{M}$ the rank-one residual manifold. The time dependent Gramian evolves on the rank-one residual manifold with $h(t) \to 0$ for $t \to \infty$; see Fig. 1. Enforcing (3.15) for the iterates (P_j, h_j), in general we obtain P_j which do not approximate the trajectory of the time dependent Gramian $P(t)$ but which are located on $\mathcal{M}$. Therefore the approximation $P_N \approx \mathcal{P}$ is good when the iterate h_N is small, because then the tuple $(P_N, h_N) \in \mathcal{M}$ is located close to $(\mathcal{P}, 0) \in \mathcal{M}$.

Theorem 2. *Let $A \in \mathbb{R}^{n\times n}$ be stable, $b \in \mathbb{R}^n$ and $\omega_i > 0$, $i = 1, \ldots, j$. Consider a Butcher tableau (2.5) with $\Lambda \in \mathbb{C}^{s\times s}$ and $\beta \in \mathbb{R}^s_+$ which satisfies (3.10). After j steps of Algorithm 2 the equation*

$$\begin{aligned} &AP_j + P_jA^{\mathsf{T}} + bb^{\mathsf{T}} \\ &\quad = h_jh_j^{\mathsf{H}} + \sum_{i=1}^{j} \omega_i^2 K_i \left(\operatorname{diag}(\beta)\overline{\Lambda} + \Lambda^{\mathsf{T}} \operatorname{diag}(\beta) - \beta\beta^{\mathsf{T}}\right) K_i^{\mathsf{H}} \end{aligned} \tag{3.17}$$

holds for $P_j = Z_jZ_j^{\mathsf{H}}$. Thus for $K_i \neq 0$, $i = 1, \ldots, j$, the iterates P_j and h_j satisfy (3.15) if and only if

$$\operatorname{diag}(\beta)\overline{\Lambda} + \Lambda^{\mathsf{T}} \operatorname{diag}(\beta) - \beta\beta^{\mathsf{T}} = 0 \tag{3.18}$$

holds.

Proof. For $j = 0$ the statement is obviously true as $P_0 = Z_0Z_0^{\mathsf{T}} = 0$ and $h_0 = b$.

For $j \in \mathbb{N}$ we first use (3.13) and then inductively (3.17) for $j-1$

$$\begin{aligned} &AP_j + P_jA^\mathsf{T} + bb^\mathsf{T} \\ &= AP_{j-1} + P_{j-1}A^\mathsf{T} + bb^\mathsf{T} + \omega_j A\mathcal{H}_j \operatorname{diag}(\beta)\mathcal{H}_j^\mathsf{H} + \omega_j \mathcal{H}_j \operatorname{diag}(\beta)\mathcal{H}_j^\mathsf{H} A^\mathsf{T} \\ &= h_{j-1}h_{j-1}^\mathsf{H} + \sum_{i=1}^{j-1} \omega_i^2 K_i \left(\operatorname{diag}(\beta)\overline{\Lambda} + (\operatorname{diag}(\beta)\Lambda)^\mathsf{T} - \beta\beta^\mathsf{T}\right) K_i^\mathsf{H} \\ &\quad + \omega_j A\mathcal{H}_j \operatorname{diag}(\beta)\mathcal{H}_j^\mathsf{H} + \omega_j \mathcal{H}_j \operatorname{diag}(\beta)\mathcal{H}_j^\mathsf{H} A^\mathsf{T}. \end{aligned} \tag{3.19}$$

Using $A\mathcal{H}_j = K_j$ and expanding $\mathcal{H}_j$ as in (3.11) we obtain

$$\begin{aligned} A\mathcal{H}_j \operatorname{diag}(\beta)\mathcal{H}_j^\mathsf{H} &= K_j \operatorname{diag}(\beta) \left([h_{j-1}, \dots, h_{j-1}] + \omega_j K_j \Lambda^\mathsf{T}\right)^\mathsf{H} \\ &= K_j \beta h_{j-1}^\mathsf{H} + \omega_j K_j \operatorname{diag}(\beta)\overline{\Lambda} K_j^\mathsf{H}. \end{aligned}$$

Inserting this in (3.19) and adding a zero we find

$$\begin{aligned} &AP_j + P_jA^\mathsf{T} + bb^\mathsf{T} \\ &= h_{j-1}h_{j-1}^\mathsf{H} + \sum_{i=1}^{j-1} \omega_i^2 K_i \left(\operatorname{diag}(\beta)\overline{\Lambda} + (\operatorname{diag}(\beta)\Lambda)^\mathsf{T} - \beta\beta^\mathsf{T}\right) K_i^\mathsf{H} \\ &\quad + \omega_j K_j \beta h_{j-1}^\mathsf{H} + \omega_j^2 K_j \operatorname{diag}(\beta)\overline{\Lambda} K_j^\mathsf{H} + \omega_j h_{j-1}\beta^\mathsf{T} K_j^\mathsf{H} + \omega_j^2 K_j \Lambda^\mathsf{T} \operatorname{diag}(\beta) K_j^\mathsf{H} \\ &\quad + \omega_j^2 K_j \beta\beta^\mathsf{T} K_j^\mathsf{H} - \omega_j^2 K_j \beta\beta^\mathsf{T} K_j^\mathsf{H} \\ &= h_{j-1}h_{j-1}^\mathsf{H} + \omega_j K_j \beta h_{j-1}^\mathsf{H} + \omega_j h_{j-1}\beta^\mathsf{T} K_j^\mathsf{H} + \omega_j^2 K_j \beta\beta^\mathsf{T} K_j^\mathsf{H} \\ &\quad + \omega_j^2 K_j \left(\operatorname{diag}(\beta)\overline{\Lambda} + \Lambda^\mathsf{T} \operatorname{diag}(\beta) - \beta\beta^\mathsf{T}\right) K_j^\mathsf{H} \\ &\quad + \sum_{i=1}^{j-1} \omega_i^2 K_i \left(\operatorname{diag}(\beta)\overline{\Lambda} + (\operatorname{diag}(\beta)\Lambda)^\mathsf{T} - \beta\beta^\mathsf{T}\right) K_i^\mathsf{H} \\ &= h_j h_j^\mathsf{H} + \sum_{i=1}^{j} \omega_i^2 K_i \left(\operatorname{diag}(\beta)\overline{\Lambda} + (\operatorname{diag}(\beta)\Lambda)^\mathsf{T} - \beta\beta^\mathsf{T}\right) K_i^\mathsf{H}, \end{aligned}$$

as $h_j = h_{j-1} + \omega_j K_j \beta$. This proves the first statement. With $K_i \neq 0$ and $\omega_i > 0$ for $i = 1, \dots, j$, the second statement is immediate. This concludes the proof. □

There are numerous Butcher tableaus for which (3.18) holds. First, observe that the diagonal entries of the equation $\operatorname{diag}(\beta)\overline{\Lambda} + (\operatorname{diag}(\beta)\Lambda)^\mathsf{T} - \beta\beta^\mathsf{T} = 0$ imply that

$$\beta_j \overline{\lambda_{jj}} + \beta_j \lambda_{jj} - \beta_j^2 = \beta_j \left(2\operatorname{Re}(\lambda_{jj}) - \beta_j\right) = 0$$

for $j = 1, \dots, s$. Thus either $\beta_j = 0$, or

$$\beta_j = 2\,\mathrm{Re}(\lambda_{jj}), \qquad j = 1, \dots, s.$$

As $\beta_j \in \mathbb{R}_+$ is required, this implies that the diagonal elements of Λ have to be in $\mathbb{C}_+$ whenever (3.18) is required.

The simplest 1-stage Butcher tableaus satisfying (3.18) are given for an arbitrary (complex) number $\mu \in \mathbb{C}_+$ by

$$\Lambda = \mu, \ \beta = 2\,\mathrm{Re}(\mu).$$

The implicit midpoint rule

$$\Lambda = \frac{1}{2}, \ \beta = 1$$

is a prominent example of such a 1-stage tableau. A simple 2-stage tableau which satisfies (3.18) is the 2-stage implicit Runge-Kutta method

$$\Lambda = \begin{bmatrix} \frac{1}{4} & \frac{1}{4} - \frac{1}{6}\sqrt{3} \\ \frac{1}{4} + \frac{1}{6}\sqrt{3} & \frac{1}{4} \end{bmatrix}, \ \beta = \begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \end{bmatrix}.$$

The latter two methods belong to the family of Gauss-Legendre methods which are special s-stage implicit Runge-Kutta methods based on Gauss-Legendre quadrature. For $s \in \mathbb{N}$, the respective method is unique and satisfies (3.18), see [19, Lemma 5.3] and the subsequent corollary, where the matrix M corresponds to the left-hand side of (3.18).

Another family of methods for which (3.18) holds is given by DIRK methods of the form

$$\Lambda = \begin{bmatrix} \mu_1 & 0 & 0 & \cdots & \cdots & 0 \\ 2\,\mathrm{Re}(\mu_1) & \mu_2 & 0 & \cdots & \cdots & 0 \\ 2\,\mathrm{Re}(\mu_1) & 2\,\mathrm{Re}(\mu_2) & \mu_3 & \ddots & & \vdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \vdots & & \ddots & \mu_{s-1} & 0 \\ 2\,\mathrm{Re}(\mu_1) & 2\,\mathrm{Re}(\mu_2) & \cdots & \cdots & 2\,\mathrm{Re}(\mu_{s-1}) & \mu_s \end{bmatrix}, \ \beta = \begin{bmatrix} 2\,\mathrm{Re}(\mu_1) \\ 2\,\mathrm{Re}(\mu_2) \\ \vdots \\ 2\,\mathrm{Re}(\mu_s) \end{bmatrix} \tag{3.20}$$

with $\mu_1, \dots, \mu_s \in \mathbb{C}_+$. In this case, it is easy to verify whether the necessary and sufficient condition (3.10)

$$\mu_p \neq \frac{1}{\omega_j \lambda_q} = \frac{1}{\omega_j |\lambda_q|^2} \overline{\lambda_q},$$

is satisfied for all $p = 1, \dots, s$ and $q = 1, \dots, n$. As λ_q is an eigenvalue of the stable matrix A, we have $\mathrm{Re}(\overline{\lambda_q}) < 0$. Thus, for stable A any DIRK method with a tableau (3.20) satisfies (3.10).

3.3. Varying Butcher tableaus instead of one fixed Butcher tableau

Now we change the point of view on the Runge-Kutta methods. So far we have used the same tableau with Λ, β in every iteration step. The time step sizes ω_j for $j = 1, \dots, N$ may vary. In the following we allow for varying tableaus with $\Lambda^{(j)} \in \mathbb{C}^{s \times s}$, $\beta^{(j)} \in \mathbb{C}^s$ during the iteration, in particular the matrices $\Lambda^{(j)}$ do not need to have the same eigenvalues. This implies the iteration

$$\begin{aligned} y_j &= y_{j-1} + \omega_j \sum_{i=1}^{s} \beta_i^{(j)} k_i^{(j)}, && j = 1, \dots, N, \\ k_i^{(j)} &= f\Big(t_{j-1} + \gamma_i^{(j)}, \, y_{j-1} + \omega_j \sum_{\ell=1}^{s} \lambda_{i\ell}^{(j)} k_\ell^{(j)}\Big), && i = 1, \dots, s, \end{aligned}$$

instead of (2.3) and (2.4).

Clearly, we would like to choose $\Lambda^{(j)}$, $\beta^{(j)}$ such that (3.18) is satisfied. In that case, $\omega_j \Lambda^{(j)}$, $\omega_j \beta^{(j)}$ also satisfy (3.18). Thus, there is no need for choosing a different time step size in every iteration step, this is in a sense already dealt with by allowing different $\Lambda^{(j)}$, $\beta^{(j)}$ in every iteration step. Therefore, we set $\omega_j = 1$ for $j = 1, \dots, N$, whenever we allow for different $\Lambda^{(j)}$, $\beta^{(j)}$ in every iteration step,

$$\begin{aligned} y_j &= y_{j-1} + \sum_{i=1}^{s} \beta_i^{(j)} k_i^{(j)}, && j = 1, \dots, N, \\ k_i^{(j)} &= f\Big(t_{j-1} + \gamma_i^{(j)}, \, y_{j-1} + \sum_{\ell=1}^{s} \lambda_{i\ell}^{(j)} k_\ell^{(j)}\Big), && i = 1, \dots, s. \end{aligned}$$

Algorithm 2 has to be modified accordingly. In line 3 and 4 the term $\omega_j \Lambda$ has to be replaced by $\Lambda^{(j)}$, while in line 5 and 6 the term $\omega_j \beta$ has to be replaced by $\beta^{(j)}$. Apparently Theorem 2 remains true even when different tableaus are used in every step as long as (3.18) holds for $\Lambda^{(i)}$ and $\beta^{(i)}$, $i = 1, \dots, j$. In its proof the tableaus Λ, β have to be replaced as described above for Algorithm 2.

3.4. 1-stage Runge-Kutta methods

Let us consider the case $s = 1$ and the Butcher tableau with $\Lambda^{(j)} = \mu_j \in \mathbb{C}_+$ and $\beta^{(j)} = 2\,\mathrm{Re}(\mu_j) \in \mathbb{R}_+$ in iteration step j. With the condition $\mathrm{Re}(\mu_j) > 0$ we make sure (3.10) is satisfied and all linear system solves in Algorithm 2 will have a unique solution.

For $s = 1$, in line 3 of Algorithm 2 the $n \times n$ linear system of equations

$$(I - \mu_j A)\, K_j = A h_{j-1}$$

has to be solved to obtain K_j. This approach is summarized in Algorithm 3.

Algorithm 3 Low rank solution to (1.1) via 1-stage Runge-Kutta methods.

```
Input: $A \in \mathbb{R}^{n\times n}$ stable, $b \in \mathbb{R}^n$, parameters $\{\mu_1,\dots,\mu_N\} \subset \mathbb{C}_+$
Output: factor $Z \in \mathbb{C}^{n\times N}$ with $ZZ^{\mathsf{H}} \approx \mathcal{P}$
1: initialize $h_0 = b$, $Z_0 = [\ ]$
2: for $j = 1,\dots,N$ do
3:     solve $(I - \mu_j A)K_j = Ah_{j-1}$ for $K_j \in \mathbb{C}^{n\times 1}$
4:     $\mathcal{H}_j = h_{j-1} + \mu_j K_j$
5:     update $Z_j = [Z_{j-1}, \sqrt{2\,\mathrm{Re}(\mu_j)}\mathcal{H}_j]$
6:     $h_j = h_{j-1} + 2\,\mathrm{Re}(\mu_j)K_j$
7: end for
8: $Z = Z_N$
```

Algorithm 3 is equivalent to Algorithm 1 (with $m = 1$) as shown in the following theorem.

Theorem 3. *Let $A \in \mathbb{R}^{n\times n}$ be stable and $B = b \in \mathbb{R}^{n\times 1}$ (that is, $m = 1$ in Algorithm 1). Let the parameters in Algorithm 3 and Algorithm 1 be chosen such that $\alpha_j = -\mu_j^{-1} \in \mathbb{C}_-$ for $j = 1,\dots,N$. Let $W_0 = b$ and W_j, V_j, $j = 1,\dots,N$ be determined by Algorithm 1. Let $h_0 = b$ and h_j, $\mathcal{H}_j$, $j = 1,\dots,N$ be determined by Algorithm 3. Then $W_j = c_j h_j$ and $\sqrt{-2\,\mathrm{Re}(\alpha_j)}V_j = d_j\sqrt{2\,\mathrm{Re}(\mu_j)}\mathcal{H}_j$ holds for some constants $c_j, d_j \in \mathbb{C}$ with $|c_j| = |d_j| = 1$. Thus the approximation $Z_jZ_j^{\mathsf{H}} \approx \mathcal{P}$ to the Gramian is the same in every step of both algorithms.*

Proof. The initialization of the algorithms gives us $W_0 = b = h_0$ and thus $c_0 = 1$. From (3.11) with $s = 1$ and $\omega_j\Lambda = \mu_j$ we find $\mathcal{H}_j = h_{j-1} + \mu_j A\mathcal{H}_j$ and thus

$$\mathcal{H}_j = (I_n - \mu_j A)^{-1}h_{j-1}.$$

Making use of line 4 in Algorithm 3 we obtain

$$(I_n - \mu_j A)^{-1}h_{j-1} = \mathcal{H}_j = h_{j-1} + \mu_j K_j. \tag{3.21}$$

Via induction we find from $W_{j-1} = c_{j-1}h_{h-1}$ for W_j as in Algorithm 1

$$\begin{aligned}
W_j &= W_{j-1} - 2\,\mathrm{Re}(\alpha_j)V_j \\
&= W_{j-1} - 2\,\mathrm{Re}(\alpha_j)(A + \alpha_j I_n)^{-1}W_{j-1} \\
&= c_{j-1}h_{j-1} - 2\,\mathrm{Re}(-\mu_j^{-1})(A - \mu_j^{-1}I_n)^{-1}c_{j-1}h_{j-1} \\
&= c_{j-1}\left(h_{j-1} + 2\,\mathrm{Re}(\mu_j^{-1})(-\mu_j)(I_n - \mu_j A)^{-1}h_{j-1}\right) \\
&\overset{(3.21)}{=} c_{j-1}\left(h_{j-1} - 2\,\mathrm{Re}(\mu_j^{-1})\mu_j(h_{j-1} + \mu_j K_j)\right) \\
&= c_{j-1}\left((1 - 2\,\mathrm{Re}(\mu_j^{-1})\mu_j)h_{j-1} - 2\,\mathrm{Re}(\mu_j^{-1})\mu_j^2 K_j)\right) \\
&= -c_{j-1}\frac{\mu_j}{\overline{\mu}_j}\left(h_{j-1} + 2\,\mathrm{Re}(\mu_j)K_j\right) \\
&= c_j h_j
\end{aligned}$$

with $1-2\operatorname{Re}(\mu_j^{-1})\mu_j = -\frac{\mu_j}{\overline{\mu}_j}$, $-2\operatorname{Re}(\mu_j^{-1})\mu_j^2 = -\frac{\mu_j}{\overline{\mu}_j}2\operatorname{Re}(\mu_j)$ and $c_j := -c_{j-1}\frac{\mu_j}{\overline{\mu}_j}$. Further we find for V_j as in Algorithm 1

$$\begin{aligned}\sqrt{-2\operatorname{Re}(\alpha_j)}V_j &= \sqrt{-2\operatorname{Re}(\alpha_j)}(A+\alpha_j I_n)^{-1}W_{j-1}\\ &= \sqrt{2\operatorname{Re}(\mu_j^{-1})}\mu_j(\mu_j A - I_n)^{-1}W_{j-1}\\ &= -\sqrt{2\operatorname{Re}(\mu_j)|\mu_j|^{-2}}\mu_j(I_n-\mu_j A)^{-1}c_{j-1}h_{j-1}\\ &\overset{(3.21)}{=} -\sqrt{2\operatorname{Re}(\mu_j)}\frac{\mu_j}{|\mu_j|}c_{j-1}\mathcal{H}_j\\ &= d_j\sqrt{2\operatorname{Re}(\mu_j)}\mathcal{H}_j\end{aligned}$$

with $d_j := -\frac{\mu_j}{|\mu_j|}c_{j-1}$. The observation $c_j\overline{c_j} = d_j\overline{d_j} = 1$ concludes the proof. □

In order to derive Algorithm 3 we have used different 1×1-tableaus in each iteration step. In general, larger tableaus allow for more accurate quadrature rules. Hence, their use might lead to an approximation of $\mathcal{P}$ which is more accurate than the one obtained by Algorithm 3. However, as we show next, an algorithm using larger DIRK tableaus can be reduced to Algorithm 3. In particular, s steps with different 1×1-tableaus $\Lambda^{(i)} = \mu_i$ and $\beta^{(i)} = 2\operatorname{Re}(\mu_i)$, $i = 1,\dots,s$ (as in Algorithm 3, $N = s$) are equivalent to one step with a particular s-stage DIRK tableau with Λ and β (as in Algorithm 2). For s steps of Algorithm 3 we have from line 6

$$\begin{aligned}h_s = h_{s-1} + 2\operatorname{Re}(\mu_s)K_s &= h_0 + \sum_{i=1}^{s} 2\operatorname{Re}(\mu_i)K_i\\ &= h_0 + [K_1,\dots,K_s]\begin{bmatrix}2\operatorname{Re}(\mu_1)\\ \vdots\\ 2\operatorname{Re}(\mu_s)\end{bmatrix}\end{aligned}\tag{3.22}$$

with

$$K_i = Ah_{i-1} + \mu_i AK_i = Ah_0 + \sum_{j=1}^{i-1} 2\operatorname{Re}(\mu_j)AK_j + \mu_i AK_i \in \mathbb{C}^{n\times 1}\tag{3.23}$$

for $i = 1,\dots,s$. Merging (3.23) for $i = 1,\dots,s$ into one equation yields

$$[K_1,\dots,K_s] = [Ah_0,\dots,Ah_0] + [AK_1,\dots,AK_s]\begin{bmatrix}\mu_1 & 2\operatorname{Re}(\mu_1) & \cdots & 2\operatorname{Re}(\mu_1)\\ 0 & \mu_2 & \cdots & 2\operatorname{Re}(\mu_2)\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \cdots & \mu_s\end{bmatrix}.\tag{3.24}$$

Hence (3.22) and (3.23) are identical to line 6 and line 3 of Algorithm 2 with $\omega_1 = 1$, where $[K_1, \dots, K_s] \in \mathbb{C}^{n\times s}$ from (3.24) corresponds to the $n \times s$ matrix K_1 in line 3 of Algorithm 2. Thus Algorithm 3 is equivalent to one step of Algorithm 2 with $s = N$, time step size $\omega_1 = 1$ and the Butcher tableau (3.20).

In summary, Algorithm 3 is equivalent to the ADI method (Algorithm 1) as well as equivalent to one step of Algorithm 2 for a DIRK method with $s = N$.

Theorem 4. *Let $A \in \mathbb{R}^{n\times n}$ be stable and $b \in \mathbb{R}^n$. Let $\mu_j \in \mathbb{C}_+$, $j = 1, \dots, s$ be given. Let $\Lambda \in \mathbb{C}^{s\times s}$ and $\beta \in \mathbb{R}^s_+$ be as in (3.20). One step of Algorithm 2 with Λ, β is equivalent to s steps of Algorithm 3 with $N = s$ and equivalent to the ADI method (Algorithm 1) with $\alpha_j = -\mu_j^{-1}$.*

Please note that due to the observation (3.1) we have considered only a vector $b \in \mathbb{R}^n$ in our discussion and, in particular, in Algorithm 3. However, Algorithm 3 can easily be adapted to a problem with $B \in \mathbb{R}^{n\times m}$. The vector b in Algorithm 3 just has to be replaced by a matrix $B \in \mathbb{R}^{n\times m}$.

3.5. A multiplicative update formula for h_j

We now focus on the iterate $h_j = h_{j-1} + \omega_j K_j \beta$ as computed in line 6 of Algorithm 2 from a Butcher tableau with $\Lambda \in \mathbb{C}^{s\times s}$ and $\beta \in \mathbb{C}^s$, where K_j is as in (3.7). As before, we will assume that $\omega_j = 1$ by moving the time step size into the Butcher tableau such that instead of one Λ and β we are now using N different $\Lambda^{(j)} \in \mathbb{C}^{s\times s}$ and $\beta^{(j)} \in \mathbb{C}^s$. Thus we consider $h_j = h_{j-1} + K_j\beta^{(j)}$. Our goal is to rewrite the update rule as a multiplicative one; $h_j = M_j h_{j-1}$.

With (3.9) we find

$$\begin{aligned}
K_j\beta^{(j)} &= ((\beta^{(j)})^\mathsf{T} \otimes I_n)\operatorname{vec}(K_j)\\
&= ((\beta^{(j)})^\mathsf{T} \otimes I_n)(I_{ns} - \Lambda^{(j)} \otimes A)^{-1}(I_s \otimes A)(\mathbb{1}_s \otimes I_n)h_{j-1}\\
&= ((\beta^{(j)})^\mathsf{T} \otimes I_n)\left\{(I_s^{-1} \otimes A^{-1})(I_{ns} - \Lambda^{(j)} \otimes A)\right\}^{-1}(\mathbb{1}_s \otimes I_n)h_{j-1}\\
&= ((\beta^{(j)})^\mathsf{T} \otimes I_n)\left\{(I_s^{-1} \otimes A^{-1}) - \Lambda^{(j)} \otimes I_n\right\}^{-1}(\mathbb{1}_s \otimes I_n)h_{j-1}\\
&= ((\beta^{(j)})^\mathsf{T} \otimes I_n)\left\{(I_{ns} - \Lambda^{(j)} \otimes A)(I_s^{-1} \otimes A^{-1})\right\}^{-1}(\mathbb{1}_s \otimes I_n)h_{j-1}\\
&= ((\beta^{(j)})^\mathsf{T} \otimes I_n)(I_s \otimes A)(I_{ns} - \Lambda^{(j)} \otimes A)^{-1}(\mathbb{1}_s \otimes I_n)h_{j-1}\\
&= ((\beta^{(j)})^\mathsf{T} \otimes A)(I_{ns} - \Lambda^{(j)} \otimes A)^{-1}(\mathbb{1}_s \otimes I_n)h_{j-1}.
\end{aligned}$$

With (2.1) we have $\Lambda^{(j)} \otimes A = Q(A \otimes \Lambda^{(j)})Q^\mathsf{T}$ for a perfect shuffle permutation matrix $Q \in \mathbb{R}^{ns\times ns}$. Moreover, as Q is orthogonal, it holds that $I_n \otimes I_s = I_{ns} = QQ^\mathsf{T} = Q(I_s \otimes I_n)Q^\mathsf{T}$. Hence,

$$
\begin{aligned}
K_j\beta^{(j)} &= ((\beta^{(j)})^\mathsf{T} \otimes A)\left\{Q(I_n \otimes I_s - A \otimes \Lambda^{(j)})Q^\mathsf{T}\right\}^{-1}(\mathbb{1}_s \otimes I_n)h_{j-1}\\
&= I_n((\beta^{(j)})^\mathsf{T} \otimes A)Q\left\{I_n \otimes I_s - A \otimes \Lambda^{(j)}\right\}^{-1}Q^\mathsf{T}(\mathbb{1}_s \otimes I_n)I_n h_{j-1}\\
&= (A \otimes (\beta^{(j)})^\mathsf{T})\left(I_{ns} - A \otimes \Lambda^{(j)}\right)^{-1}(I_n \otimes \mathbb{1}_s)h_{j-1}
\end{aligned}
$$

as the perfect shuffle matrix for a vector is the identity.

Thus the iterate h_j is obtained from h_{j-1} via

$$h_j = M_j h_{j-1} \tag{3.25}$$

with the iteration matrix

$$M_j := M_j(A) = I_n + (A \otimes (\beta^{(j)})^\mathsf{T})\left(I_{ns} - A \otimes \Lambda^{(j)}\right)^{-1}(I_n \otimes \mathbb{1}_s). \tag{3.26}$$

The matrix valued function $M(z) = I_n + (z \otimes \beta^\mathsf{T})(I_{ns} - z \otimes \Lambda)^{-1}(I_n \otimes \mathbb{1}_s)$ can be viewed as a generalization of the stability function $R(z) = 1 + z\beta^\mathsf{T}(I - z\Lambda)^{-1}\mathbb{1}_s$ of the corresponding Runge-Kutta method.

For a diagonalizable system matrix $A = VDV^{-1}$ with the matrix of right eigenvectors V and the diagonal matrix $D = \operatorname{diag}(\lambda_1, \dots, \lambda_n)$ containing the eigenvalues of A on the diagonal, the iteration matrix M_j from (3.26) simplifies considerably. We define the stability function $R_{(j)}(z) = 1 + z(\beta^{(j)})^\mathsf{T}(I - z\Lambda^{(j)})^{-1}\mathbb{1}_s$ for a Runge-Kutta method with $\Lambda^{(j)}$ and $\beta^{(j)}$ and see

$$
\begin{aligned}
M_j &= I_n + (A \otimes (\beta^{(j)})^\mathsf{T})\left(I_{ns} - (A \otimes \Lambda^{(j)})\right)^{-1}(I_n \otimes \mathbb{1}_s)\\
&= I_n + V(D \otimes (\beta^{(j)})^\mathsf{T})(V^{-1} \otimes I_s)\\
&\qquad \cdot \left\{I_n \otimes I_s - (V \otimes I_s)(D \otimes \Lambda^{(j)})(V^{-1} \otimes I_s)\right\}^{-1}(I_n \otimes \mathbb{1}_s)\\
&= I_n + V(D \otimes (\beta^{(j)})^\mathsf{T})\left(I_n \otimes I_s - (D \otimes \Lambda^{(j)})\right)^{-1}(I_n \otimes \mathbb{1}_s)V^{-1}\\
&= V\left(I_n + \begin{bmatrix} \lambda_1(\beta^{(j)})^\mathsf{T} & & \\ & \ddots & \\ & & \lambda_n(\beta^{(j)})^\mathsf{T}\end{bmatrix}\begin{bmatrix} I_s - \lambda_1\Lambda^{(j)} & & \\ & \ddots & \\ & & I_s - \lambda_n\Lambda^{(j)}\end{bmatrix}^{-1}\begin{bmatrix} \mathbb{1}_s & & \\ & \ddots & \\ & & \mathbb{1}_s\end{bmatrix}\right)V^{-1}\\
&= V\begin{bmatrix} 1+\lambda_1(\beta^{(j)})^\mathsf{T}(I_s - \lambda_1\Lambda^{(j)})^{-1}\mathbb{1}_s & & \\ & \ddots & \\ & & 1+\lambda_n(\beta^{(j)})^\mathsf{T}(I_s - \lambda_n\Lambda^{(j)})^{-1}\mathbb{1}_s\end{bmatrix}V^{-1}\\
&= V\begin{bmatrix} R_{(j)}(\lambda_1) & & \\ & \ddots & \\ & & R_{(j)}(\lambda_n)\end{bmatrix}V^{-1},
\end{aligned} \tag{3.27}
$$

i.e. the iteration matrix is determined by the stability function $R_{(j)}(z)$ corresponding to the Butcher tableau with $\Lambda^{(j)}$, $\beta^{(j)}$ as well as by the eigenvalues and eigenvectors of the system matrix A. For non-diagonalizable system matrices we have no explicit formula in terms of the stability function. However, M_j can be interpreted as a composition of continuous functions. As such it depends continuously on A. The diagonalizable matrices are dense in the set of all matrices. Thus M_j is (implicitly) determined by the stability function of the utilized Runge-Kutta method for non-diagonalizable matrices A, too.

Tableaus with the same stability function yield the same approximation h_j as $h_j = M_j h_{j-1}$ holds with M_j as in (3.27) for each of the different methods. In case the tableaus satisfy (3.15), then

$$AZ_jZ_j^{\mathsf{H}} + Z_jZ_j^{\mathsf{H}}A^{\mathsf{T}} + (bb^{\mathsf{T}} - h_jh_j^{\mathsf{H}}) = 0 \tag{3.28}$$

holds, and $Z_jZ_j^{\mathsf{H}}$ (but not Z_j) is determined by h_j via (3.28). Thus, all tableaus with the same stability function yield the same approximation $P_j = Z_jZ_j^{\mathsf{H}}$ (as long as the tableaus satisfy (3.10), (3.18) and $\beta \in \mathbb{R}^s_+$).

3.6. Runge-Kutta methods satisfying (3.18)

We now investigate when two s-stage Runge-Kutta methods have the same stability function. Assume that Λ and β are as in Theorem 2 and satisfy (3.18), $0 = \operatorname{diag}(\beta)\overline{\Lambda} + \Lambda^{\mathsf{T}}\operatorname{diag}(\beta) - \beta\beta^{\mathsf{T}}$. Then

$$-\overline{\Lambda} = \operatorname{diag}(\beta)^{-1}(\Lambda^{\mathsf{T}} - \beta\mathbb{1}_s^{\mathsf{T}})\operatorname{diag}(\beta)$$

as $\beta_j > 0$, $j = 1, \dots, s$, proving that the matrices $-\overline{\Lambda}$ and $\Lambda - \mathbb{1}_s\beta^{\mathsf{T}}$ are similar, i.e.

$$-\overline{\Lambda} \quad \sim \quad \Lambda - \mathbb{1}_s\beta^{\mathsf{T}}. \tag{3.29}$$

Let Λ have eigenvalues $\mu_1, \dots, \mu_s$. Then with the determinant based characterization of the stability function (2.6) we find that

$$\begin{aligned} R(z) &= \frac{\det(I + z(\mathbb{1}_s\beta^{\mathsf{T}} - \Lambda))}{\det(I - z\Lambda)} = \frac{\det(I + z\overline{\Lambda})}{\det(I - z\Lambda)} \\ &= \frac{(1 + \overline{\mu_1}z)\cdots(1 + \overline{\mu_s}z)}{(1 - \mu_1 z)\cdots(1 - \mu_s z)} \end{aligned} \tag{3.30}$$

holds. This is just the stability function of a DIRK method as given in (3.20). Thus any method based on a Butcher tableau with Λ, β such that Λ has the eigenvalues $\mu_1, \dots, \mu_s$ and $\beta \in \mathbb{R}^s_+$ satisfies (3.18) is equivalent to a DIRK method (3.20). Note that obviously the order of the parameters μ_i is irrelevant. We summarize our findings in the following theorem.

Theorem 5. *Let $A \in \mathbb{R}^{n\times n}$ be stable and $b \in \mathbb{R}^n$. Let a Butcher tableau (2.5) with $\Lambda \in \mathbb{C}^{s\times s}$ and $\beta \in \mathbb{R}^s_+$ satisfying (3.18) be given. Let Λ have eigenvalues $\mu_1, \ldots, \mu_s \in \mathbb{C}_+$, so (3.10) is guaranteed. Moreover, (3.29) holds. Then the method based on this tableau is equivalent to the DIRK method given in (3.20) and therefore equivalent to s steps with the 1-stage tableaus $\Lambda^{(j)} = \mu_j$ and $\beta^{(j)} = 2\operatorname{Re}(\mu_j)$ for $j = 1, \ldots, s$, i.e., equivalent to s steps of Algorithm 3.*

Let two different Butcher tableaus with $\Lambda \in \mathbb{C}^{s\times s}$, $\beta \in \mathbb{R}^s_+$ and $\widetilde{\Lambda} \in \mathbb{C}^{s\times s}$, $\widetilde{\beta} \in \mathbb{R}^s_+$ be given such that Λ and $\widetilde{\Lambda}$ have the same eigenvalues, $\sigma(\Lambda) = \sigma(\widetilde{\Lambda})$. Further assume that Λ, β are chosen such that (3.18), (3.10) and (3.29) hold. Please note that Theorem 5 does not imply that the method based on the Butcher tableau with $\widetilde{\Lambda}$, $\widetilde{\beta}$ is equivalent to the one based on Λ, β. Only in case $\widetilde{\Lambda}$, $\widetilde{\beta}$ also satisfies (3.18), (3.10) and (3.29), the two methods are equivalent. In the following we demonstrate this statement with particular tableaus.

Consider 2 steps with the method based on the 1-stage tableaus $\Lambda^{(1)} = \mu$, $\beta^{(1)} = 2\operatorname{Re}(\mu)$ and $\Lambda^{(2)} = \overline{\mu}$, $\beta^{(2)} = 2\operatorname{Re}(\mu)$ where $\mu \in \mathbb{C}$ is chosen such that $\operatorname{Re}(\mu) \neq 0$ and (3.10) holds. This is equivalent to one step of the method based on the DIRK method with

$$\Lambda = \begin{bmatrix} \mu & 0 \\ 2\operatorname{Re}(\mu) & \overline{\mu} \end{bmatrix} \in \mathbb{C}^{2\times 2},\ \beta = \begin{bmatrix} 2\operatorname{Re}(\mu) \\ 2\operatorname{Re}(\mu) \end{bmatrix} \in \mathbb{R}^2.$$

The matrix Λ is similar to

$$\widetilde{\Lambda} = \begin{bmatrix} \overline{\mu} & 0 \\ 0 & \mu \end{bmatrix} = V^{-1}\Lambda V \in \mathbb{C}^{2\times 2}$$

as well as to

$$\widehat{\Lambda} = \begin{bmatrix} \operatorname{Re}(\mu) & -\operatorname{Im}(\mu) \\ \operatorname{Im}(\mu) & \operatorname{Re}(\mu) \end{bmatrix} \in \mathbb{R}^{2\times 2}.$$

A quick check reveals that for $\delta \in \mathbb{C}^2$

$$0 = \operatorname{diag}(\delta)\overline{\widetilde{\Lambda}} + \widetilde{\Lambda}^{\mathsf{T}} \operatorname{diag}(\delta) - \delta\delta^{\mathsf{T}}$$

as well as

$$0 = \operatorname{diag}(\delta)\overline{\widehat{\Lambda}} + \widehat{\Lambda}^{\mathsf{T}} \operatorname{diag}(\delta) - \delta\delta^{\mathsf{T}}$$

is satisfied only if either $\delta_1 = 0$ or $\delta_2 = 0$. Thus no method based on a Butcher tableau with $\widetilde{\Lambda}$ or $\widehat{\Lambda}$ is equivalent to the method based on the Butcher tableau with Λ, β.

Another matrix similar to Λ is given by

$$\breve{\Lambda} = \begin{bmatrix} \operatorname{Re}(\mu) & \operatorname{Re}(\mu) + \varphi|\mu| \\ \operatorname{Re}(\mu) - \varphi|\mu| & \operatorname{Re}(\mu) \end{bmatrix} \in \mathbb{R}^{2\times 2}$$

with $\varphi = \operatorname{sgn}(\operatorname{Im}(\mu)) = \frac{\operatorname{Im}(\mu)}{|\operatorname{Im}(\mu)|}$. Choosing

$$\breve{\beta} = \begin{bmatrix} 2\operatorname{Re}(\mu) \\ 2\operatorname{Re}(\mu) \end{bmatrix}$$

we find that $\breve{\Lambda}$, $\breve{\beta}$ satisfy (3.18). Due to the choice of μ, (3.10) and (3.29) also hold. Thus the methods based on Λ, β and on $\breve{\Lambda}$, $\breve{\beta}$ are equivalent.

The iteration with Λ and β produces complex valued iterates. Using the equivalent iteration with $\breve{\Lambda}$ and $\breve{\beta}$ allows to keep the iterates real valued. For a more detailed description of this realification approach we refer to [8].

Finally, please note that given Λ, β satisfying (3.18) and $\breve{\Lambda} = U^{-1}\Lambda U$ with a regular matrix $U \in \mathbb{C}^{2\times 2}$ the condition $0 = \operatorname{diag}(\beta)\overline{\Lambda} + \Lambda^\mathsf{T}\operatorname{diag}(\beta) - \beta\beta^\mathsf{T}$ can not be used to determine whether or not $\breve{\beta}$ exists such that (3.18) is satisfied for $\breve{\Lambda}, \breve{\beta}$. We have

$$\begin{aligned} 0 &= \left(U^\mathsf{T}\operatorname{diag}(\beta)\overline{U}\right)\left(\overline{U}^{-1}\overline{\Lambda U}\right) + \left(U^\mathsf{T}\Lambda^\mathsf{T}U^{-\mathsf{T}}\right)\left(U^\mathsf{T}\operatorname{diag}(\beta)\overline{U}\right) - U^\mathsf{T}\beta\beta^\mathsf{T}\overline{U} \\ &= Y\overline{\breve{\Lambda}} + \breve{\Lambda}^\mathsf{T}Y - U^\mathsf{T}\beta\beta^\mathsf{T}\overline{U} \end{aligned}$$

where $Y = U^\mathsf{T}\operatorname{diag}(\beta)\overline{U}$. Unfortunately, in general Y will not be diagonal. Moreover, in general, $U^\mathsf{T}\beta \neq \beta^\mathsf{T}\overline{U}$. Thus, considering the above transformation of (3.18) does not help in order to determine an appropriate $\breve{\beta}$ such that $\breve{\Lambda}, \breve{\beta}$ satisfy (3.18). One needs to check $0 = \operatorname{diag}(\breve{\beta})\overline{\breve{\Lambda}} + \breve{\Lambda}^\mathsf{T}\operatorname{diag}(\breve{\beta}) - \breve{\beta}\breve{\beta}^\mathsf{T}$ directly.

3.7. Choice of parameters in Algorithm 3

Here we discuss the choice of the parameters $\mu_1, \ldots, \mu_N$ to obtain a good approximation to the Gramian. We know that the Lyapunov residual for iterates which fulfill the invariant (3.15) is given by $h_N h_N^\mathsf{H}$. Hence for a small residual the norm $\|h_N\|$ has to be small.

Consider Algorithm 3, that is, $R_{(j)}(z) = \frac{1+\overline{\mu_j}z}{1-\mu_j z}$. Then with (3.25) and (3.27) we find

$$\begin{aligned} h_N &= \left(\prod_{j=1}^{N} M_j\right) b \qquad (3.31) \\ &= V\left(\prod_{j=1}^{N}\begin{bmatrix} R_{(j)}(\lambda_1) & & \\ & \ddots & \\ & & R_{(j)}(\lambda_n)\end{bmatrix}\right)V^{-1}b \\ &= V\left(\prod_{j=1}^{N}\begin{bmatrix} \frac{1+\overline{\mu_j}\lambda_1}{1-\mu_j\lambda_1} & & \\ & \ddots & \\ & & \frac{1+\overline{\mu_j}\lambda_n}{1-\mu_j\lambda_n}\end{bmatrix}\right)V^{-1}b. \end{aligned}$$

Taking the 2-norm this implies

$$\|h_N\| \leq \|V\| \max_{i=1,\ldots,n} \left(\prod_{j=1}^{N} \frac{|1+\overline{\mu_j}\lambda_i|}{|1-\mu_j\lambda_i|} \right) \|V^{-1}\| \|b\|.$$

To minimize this error bound the parameters μ_j have to be chosen such that the problem

$$\min_{\mu_1,\ldots,\mu_N} \max_{i=1,\ldots,n} \prod_{j=1}^{N} \frac{|1+\overline{\mu_j}\lambda_i|}{|1-\mu_j\lambda_i|}$$

is solved. It is equivalent to the rational min-max problem [5, Sec. 2.2]. For $N = n$ the choice $\mu_j = -\overline{\lambda_j}^{-1}$, $j = 1, \ldots, N$ yields $h_N = 0$ and thus $P_N = \mathcal{P}$. However, in that case the final iterate Z_N of Algorithm 3 is of size $n \times n$. Thus, this would not be a low rank approximation. Still this suggests that the parameters should somehow approximate the negative conjugated inverse of the eigenvalues λ_i of A, i.e. $\mu_j \approx -\overline{\lambda_i}^{-1}$.

As we consider a stable system matrix A, all eigenvalues λ_i have negative real part. Therefore, to obtain a factor with modulus smaller one, i.e. $\frac{|1+\overline{\mu_j}\lambda_i|}{|1-\mu_j\lambda_i|} < 1$, the parameters μ_j must have positive real parts.

As $\omega_j = 1$ in Algorithm 3 and due to (3.10), the choice of μ_j with positive real parts guarantees the uniqueness of the solution K of (3.8). As Algorithm 3 is equivalent to the ADI iteration, we refer the reader to [5] for a discussion of different strategies for choosing the parameters.

3.8. Shifted Legendre polynomials

In [34] approximate Cholesky factors of the Gramian are obtained via an approximation of the impulse response $h(t) = e^{At}B$ by shifted Legendre polynomials. With a fixed scalar $s \in \mathbb{N}$, $D = \operatorname{diag}(1, \sqrt{3}, \ldots, \sqrt{2s-1})$ and $\omega \in \mathbb{R}_+$ the resulting Gramian approximation is given by $\mathcal{P} \approx FF^\mathsf{T}$ with $F = \widetilde{F} \cdot D^{-\frac{1}{2}} \cdot \sqrt{\omega}$. The matrix $\widetilde{F} \in \mathbb{R}^{n\times s}$ is determined via the ns-dimensional block tridiagonal system of linear equations

$$(I_{ns} - \omega(\Lambda' \otimes A)) \operatorname{vec}(\widetilde{F}) = B \otimes e_1 \tag{3.32}$$

with

$$\Lambda' = \begin{bmatrix} \frac{1}{2} & -\frac{1}{6} & & & \\ \frac{1}{2} & 0 & -\frac{1}{10} & & \\ & \frac{1}{6} & 0 & \ddots & \\ & & \ddots & \ddots & -\frac{1}{2(2s+1)} \\ & & & \frac{1}{2(2s-1)} & 0 \end{bmatrix}.$$

Here we show that the approximation FF^T to the Gramian is the same as with one step of Algorithm 2 using the Gauß-Legendre tableau of size s. It is thus equivalent to Algorithm 3 and therefore equivalent to the ADI iteration, see Theorem 2 and Theorem 3. Let $\mathcal{H}' = \widetilde{F} \cdot D^{-1}$. De-vectorizing (3.32) yields

$$\mathcal{H}'D - \omega A\mathcal{H}'D(\Lambda')^\mathsf{T} = [B, 0, \dots, 0]$$

and multiplication with D^{-1} from the right results in

$$\mathcal{H}' = [B, 0, \dots, 0] + \omega A\mathcal{H}'(D^{-1}\Lambda'D)^\mathsf{T}. \tag{3.33}$$

We next show that $D^{-1}\Lambda'D = X_\mathrm{G}$ holds for the matrix

$$X_\mathrm{G} = \begin{bmatrix} \frac{1}{2} & -\xi_1 & & \\ \xi_1 & 0 & \ddots & \\ & \ddots & \ddots & -\xi_{s-1} \\ & & \xi_{s-1} & 0 \end{bmatrix} \tag{3.34}$$

from [9, Lem. 359A] with $\xi_i = \frac{1}{2\sqrt{4i^2-1}}$. Multiplication of Λ' with D^{-1} from the left means that the ith row is scaled with the factor $\frac{1}{\sqrt{2i-1}}$ while multiplication with D from the right means scaling of the ith column with $\sqrt{2i-1}$. Thus we find for the upper and lower diagonal entries of $D^{-1}\Lambda'D$ that

$$(D^{-1}\Lambda'D)_{i,i+1} = \frac{\sqrt{2(i+1)-1}}{\sqrt{2i-1}} \cdot \frac{-1}{2(2i+1)} = \frac{-1}{2\sqrt{4i^2-1}},$$
$$(D^{-1}\Lambda'D)_{i+1,i} = \frac{\sqrt{2i-1}}{\sqrt{2(i+1)-1}} \cdot \frac{1}{2(2i-1)} = \frac{1}{2\sqrt{4i^2-1}}$$

holds for $i = 1, \dots, s-1$, which proves (3.34).

Let Λ, β and γ be as in the Butcher tableau for the Gauß-Legendre method of size s. Define the generalized Vandermonde matrix as in [9, Sec. 359]

$$W := \begin{bmatrix} P_0(\gamma_1) & \cdots & P_{s-1}(\gamma_1) \\ \vdots & \ddots & \vdots \\ P_0(\gamma_s) & \dots & P_{s-1}(\gamma_s) \end{bmatrix}$$

with the normalized Legendre polynomials P_k of degree k for $k = 0, \dots, s-1$ on the interval $[0, 1]$ so that

$$\sum_{i=1}^{s} \beta_i P_k(c_i) P_l(c_i) = \delta_{kl}$$

holds. This implies orthonormality of the matrix $\operatorname{diag}(\beta)^{\frac{1}{2}}W$ as well as the equations

$$\begin{aligned} W^\top &= (\operatorname{diag}(\beta)W)^{-1}, \\ e_1^\top &= \mathbb{1}_s^\top \operatorname{diag}(\beta)W, \end{aligned}$$

so that

$$\begin{aligned} [B, 0, \ldots, 0] &= e_1^\top \otimes B \\ &= (\mathbb{1}_s^\top \cdot \operatorname{diag}(\beta)W) \otimes (B \cdot 1) \\ &= (\mathbb{1}_s^\top \otimes B) \cdot (\operatorname{diag}(\beta)W \otimes 1) \\ &= [B, \ldots, B] \operatorname{diag}(\beta)W \end{aligned}$$

holds. Further, due to [9, Lem. 359A], we have

$$X_{\mathrm{G}} = W^\top \operatorname{diag}(\beta)\Lambda W.$$

Herewith (3.33) becomes

$$\mathcal{H}' = [B, \ldots, B] \operatorname{diag}(\beta)W + \omega A\mathcal{H}'W^\top\Lambda^\top \operatorname{diag}(\beta)W. \tag{3.35}$$

Multiplication of (3.35) with $W^\top = (\operatorname{diag}(\beta)W)^{-1}$ from the right and setting $\mathcal{H} = \mathcal{H}'W^\top$ yields

$$\mathcal{H} = [B, \ldots, B] + \omega A\mathcal{H}\Lambda^\top.$$

Due to line 5 in Algorithm 2 the approximate Cholesky factor using the Gauß-Legendre method is given by $Z = \mathcal{H} \operatorname{diag}(\omega\beta)^{\frac{1}{2}}$. Therefore, because of

$$\begin{aligned} ZZ^\top &= \mathcal{H} \operatorname{diag}(\omega\beta)\mathcal{H}^\top \\ &= \omega\mathcal{H}' \underbrace{W^\top \operatorname{diag}(\beta)W}_{=I_s}(\mathcal{H}')^\top \\ &= \omega\widetilde{F}D^{-\frac{1}{2}}(\widetilde{F}D^{-\frac{1}{2}})^\top \\ &= FF^\top, \end{aligned}$$

the approximation $FF^\top$ based on Legendre polynomials and the approximation $ZZ^\top$ based on the Gauß-Legendre method is the same. An efficient way to compute the approximation is given by Algorithm 3 with parameters $\mu_1, \ldots, \mu_s$ chosen as the eigenvalues of the matrix ωX_{G} from (3.33), which is similar to $\omega\Lambda$ from the Gauß-Legendre tableau.

3.9. Numerical experiments

We present numerical experiments to illustrate some properties of the geometric integration method. For the experiments the implementation of Algorithm 2 was modified such that generalized Lyapunov equations

$$A\mathcal{P}E^{\mathsf{T}} + E\mathcal{P}A^{\mathsf{T}} + BB^{\mathsf{T}} = 0$$

with $B \in \mathbb{R}^{n\times m}$ could be handled efficiently.

To treat E line 4 of Algorithm 2 was modified to

$$E\mathcal{H}_j = [h_{j-1}, \ldots, h_{j-1}] + \omega_j A\mathcal{H}_j\Lambda^{\mathsf{T}}. \tag{3.36}$$

To solve for $\mathcal{H}_j$ efficiently a Schur decomposition $(\Lambda')^{\mathsf{T}} = S\Lambda^{\mathsf{T}}S^{-1}$ with an upper triangular matrix $(\Lambda')^{\mathsf{T}}$ and an orthonormal matrix S is used. Define $\mathcal{H}'_j = \mathcal{H}_jS^{-1}$, then (3.36) is equivalent to

$$E\mathcal{H}'_j = [\alpha_1 h_{j-1}, \ldots, \alpha_s h_{j-1}] + \omega_j A\mathcal{H}'_j(\Lambda')^{\mathsf{T}} \tag{3.37}$$

with $[\alpha_1, \ldots, \alpha_s] = \mathbb{1}_s S^{-1}$. As $(\Lambda')^{\mathsf{T}}$ is upper triangular, solving for $\mathcal{H}'_j = [\mathfrak{h}'^{(j)}_1, \ldots, \mathfrak{h}'^{(j)}_s]$ means solving

$$(E - \omega_j\Lambda'_{ii}A)\mathfrak{h}'^{(j)}_i = \alpha_i h_{j-1} + \omega_j \sum_{l=1}^{i-1} \Lambda'_{il}A\mathfrak{h}'^{(j)}_l$$

for $i = 1, \ldots, s$. In case $B \in \mathbb{R}^{n\times m}$ has multiple columns Λ and Λ' in (3.36) and (3.37) have to be replaced by $\Lambda \otimes I_m$ and $\Lambda' \otimes I_m$. Line 6 of Algorithm 2 becomes $h_j = h_{j-1} + \omega_j A\mathcal{H}_j(\beta \otimes I_m)$ with $h_j \in \mathbb{C}^{n\times m}$. Further the relation $\mathcal{H}_j = \mathcal{H}'_j(S \otimes I_m)$ has to be used.

In order to assess different Runge-Kutta methods, the change of the Lyapunov residual (1.4) and the iterate h_j is examined. Both quantities are coupled through (3.17) and a small residual indicates a good approximate solution. The relative Lyapunov residual norm $\|\mathcal{L}(Z_jZ_j^{\mathsf{H}})\|_{\mathrm{F}}/\|BB^{\mathsf{T}}\|_{\mathrm{F}}$ and the relative norm $\|h_j\|_{\mathrm{F}}/\|B\|_{\mathrm{F}}$ are determined after each iteration step. As $\mathcal{L}(Z_jZ_j^{\mathsf{H}})$ is of size $n \times n$ and n is large, we can not simply evaluate the norm. For a fast calculation of $\|\mathcal{L}(Z_jZ_j^{\mathsf{H}})\|_{\mathrm{F}}$ we make use of (3.17), which can be rewritten as follows

$$\mathcal{L}(ZZ^{\mathsf{H}}) = \underbrace{[h_N, K_1, \ldots, K_N]}_{=:V}\Bigg(\underbrace{\begin{bmatrix}1 & \\ & \operatorname{diag}(\omega_1^2, \ldots, \omega_N^2)\otimes M\end{bmatrix}\otimes I_m}_{=:M_0}\Bigg)[h_N, K_1, \ldots, K_N]^{\mathsf{H}},$$

with the Hermitian matrix $M = \operatorname{diag}(\beta)\overline{\Lambda} + \Lambda^{\mathsf{T}}\operatorname{diag}(\beta) - \beta\beta^{\mathsf{T}}$. Recall that Z is a low rank approximation of dimension $n \times (1 + Ns)m$ with $(1 + Ns)m \ll n$. Thus M_0 is a small matrix of size $(1 + Ns)m$. To evaluate the residual norm we use

$$\begin{aligned}\|\mathcal{L}(ZZ^{\mathsf{H}})\|_{\mathrm{F}} = \|VM_0V^{\mathsf{H}}\|_{\mathrm{F}} &= \mathrm{trace}(\underbrace{VM_0V^{\mathsf{H}}VM_0V^{\mathsf{H}}}_{n\times n})^{\frac{1}{2}} \\ &= \mathrm{trace}(\underbrace{M_0V^{\mathsf{H}}VM_0V^{\mathsf{H}}V}_{(1+Ns)m\times(1+Ns)m})^{\frac{1}{2}}, \end{aligned} \tag{3.38}$$

which holds because of the invariance of the trace under cyclic permutations of its argument. The trace of the small matrix in (3.38) can be calculated rapidly. The slowest part is the calculation of $V^{\mathsf{H}}V$. As V is constructed column by column only the scalar products with the new entries should be calculated and stored in every step of the iteration. If $M = 0$ holds, then the residual norm evaluation simplifies considerably to

$$\|\mathcal{L}(ZZ^{\mathsf{H}})\|_{\mathrm{F}} = \|h_N h_N^{\mathsf{H}}\|_{\mathrm{F}} = \|h_N^{\mathsf{H}} h_N\|_{\mathrm{F}}.$$

The two examples used here both are from the Oberwolfach model reduction benchmark collection.[1] The first example represents a finite element model of a chip cooled by convection. The system matrices are $A, E \in \mathbb{R}^{20082\times 20082}$ and $B \in \mathbb{R}^{20082\times 1}$ (ID 1428, [26]). The second one comes from the semi-discretization of a heat transfer process for optimal cooling of steel profiles. Due to different mesh widths used in the semi-discretization matrices of several dimensions are available. We chose the largest example with $A, E \in \mathbb{R}^{79841\times 79841}$ and $B \in \mathbb{R}^{79841\times 7}$ (ID 1445, [7]). All numerical experiments were executed using MATLAB 2019a on an Intel® Core™ i7-5600U CPU @ 2.60 GHz with 12 GB RAM.

The Radau IA method with the Butcher tableau

$$\begin{array}{c|ccc} 0 & \frac{1}{9} & \frac{-1-\sqrt{6}}{18} & \frac{-1+\sqrt{6}}{18} \\ \frac{3}{5} - \frac{\sqrt{6}}{10} & \frac{1}{9} & \frac{11}{45} + \frac{7\sqrt{6}}{360} & \frac{11}{45} - \frac{43\sqrt{6}}{360} \\ \frac{3}{5} + \frac{\sqrt{6}}{10} & \frac{1}{9} & \frac{11}{45} + \frac{43\sqrt{6}}{360} & \frac{11}{45} - \frac{7\sqrt{6}}{360} \\ \hline & \frac{1}{9} & \frac{4}{9} + \frac{\sqrt{6}}{36} & \frac{4}{9} - \frac{\sqrt{6}}{36} \end{array} \tag{3.39}$$

of size 3, denoted by *Radau IA (3)*, was used in the experiments. It was chosen as Λ has only nonzero eigenvalues, no entry in β is zero and the iterates do not lie on the manifold $\mathcal{M}$ from (3.16), because (3.18) is not satisfied. Other methods like Radau IIA and Lobatto IIIC were also tested and showed a similar behavior as the Radau IA method. The second method used is a DIRK method (3.20) where the diagonal elements μ_1, μ_2, μ_3 are chosen to be the eigenvalues of Λ from (3.39), so, in contrast to the Radau IA method, the iterates lie on the manifold $\mathcal{M}$. This method is denoted by *$\mathcal{M}$-Radau IA (3)*. Further, the Gauß-Legendre method with Butcher tableau

[1] https://sparse.tamu.edu/Oberwolfach.

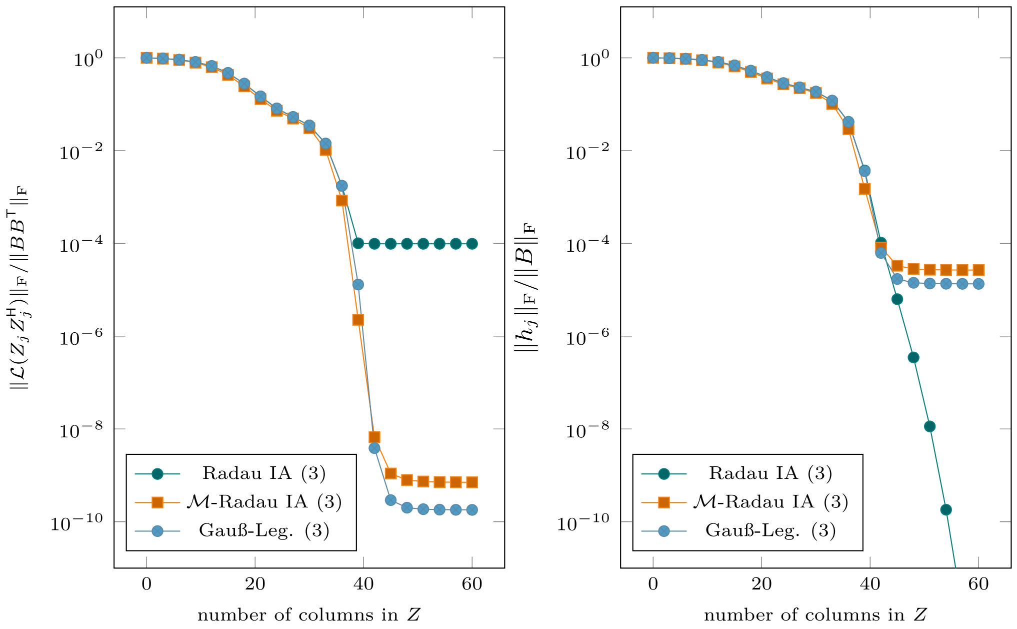

Fig. 2. The chip example with $n = 20082$, $m = 1$ and time step sizes `logspace(-5,2,20)`. Relative norms of the residual (left) and relative norms of the iterates h_j (right). (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article. Radau IA behaves differently from the other methods.)

$$\begin{array}{c|ccc}
\frac{1}{2} - \frac{\sqrt{15}}{10} & \frac{5}{36} & \frac{2}{9} - \frac{\sqrt{15}}{15} & \frac{5}{36} - \frac{\sqrt{15}}{30} \\
\frac{1}{2} & \frac{5}{36} + \frac{\sqrt{15}}{24} & \frac{2}{9} & \frac{5}{36} - \frac{\sqrt{15}}{24} \\
\frac{1}{2} + \frac{\sqrt{15}}{10} & \frac{5}{36} + \frac{\sqrt{15}}{30} & \frac{2}{9} + \frac{\sqrt{15}}{15} & \frac{5}{36} \\
\hline
 & \frac{5}{18} & \frac{4}{9} & \frac{5}{18}
\end{array}$$

is used, denoted by *Gauß-Leg. (3)*. As mentioned in Section 3.2 the iterates lie on the manifold $\mathcal{M}$. We note that due to different eigenvalues of Λ in *Gauß-Leg. (3)* and *$\mathcal{M}$-Radau IA (3)* the methods are not equivalent.

In Fig. 2 the relative norm of the Lyapunov residual and the relative norm of the iterate h_j is displayed for the chip example. As this example has single input and the Butcher tableaus are of size 3, after 20 steps of Algorithm 2 the approximate Cholesky factor Z consists of 60 columns. The convergence for the methods *$\mathcal{M}$-Radau IA (3)* and *Gauß-Leg. (3)* is comparable and stagnates after 48 columns at 10^{-9} while the residual obtained through the *Radau IA (3)* method stagnates after 39 columns at 10^{-4}. On the right hand side the relative norm of the iterate h_j is displayed. While for the two methods with iterates on $\mathcal{M}$ the relative norm of h_j correlates with the relative residual norm on the left hand side in accordance with $\|\mathcal{L}(Z_j Z_j^{\mathsf{H}})\|_{\mathrm{F}} = \|h_j h_j^{\mathsf{H}}\|_{\mathrm{F}} = \|h_j\|_{\mathrm{F}}^2$, this is not the case for the method *Radau IA (3)*. This behavior is in agreement with (3.15), which for iterates on $\mathcal{M}$ connects the Lyapunov residual and $h_j h_j^{\mathsf{H}}$. On the other hand,

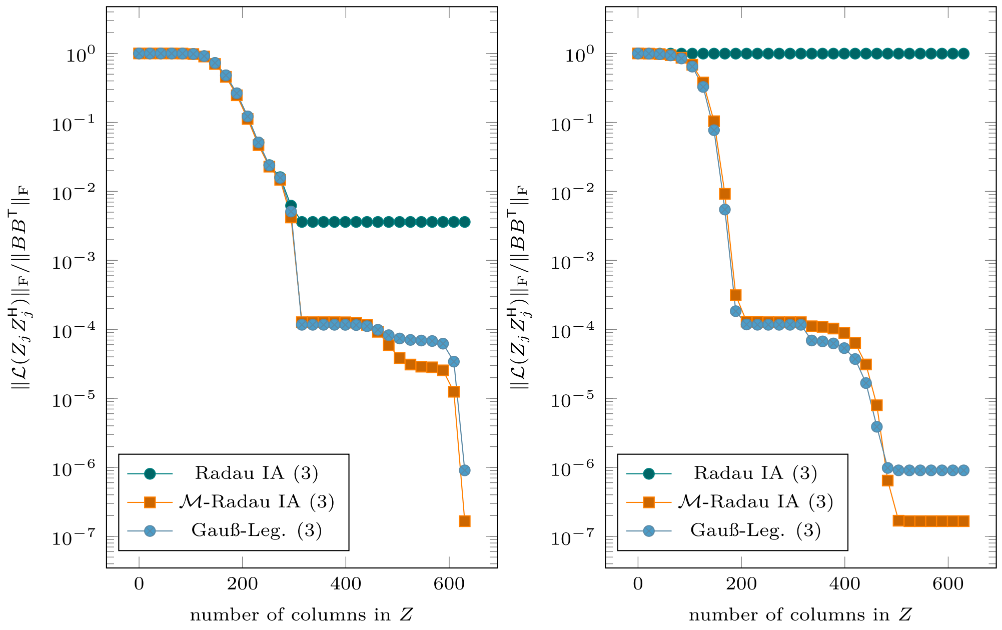

Fig. 3. Residual norms for the rail example with $n = 79841$, $m = 7$. Time step sizes `[logspace(-6,5,15), logspace(-6,5,15)]` (left) and `[logspace(5,-6,15), logspace(5,-6,15)]` (right).

for iterates not lying on $\mathcal{M}$, we have to apply (3.17). That is, the Lyapunov residual consists of $h_jh_j^{\mathsf{H}}$ and terms depending on K_i from line 3 of Algorithm 2.

The results for the rail example are displayed in Fig. 3. With 7 inputs and Butcher tableaus of size 3, after 30 steps of Algorithm 2 the approximate Cholesky factor Z is made up of 630 columns. We investigated the sensitivity of the methods to the choice of the time step sizes. Therefore 15 steps with increasing (decreasing) time step sizes on the left (right) hand side were performed, then the same 15 time step sizes were used again. For the increasing time step sizes on the left hand side the first half of the residual norm plot shows the same behavior as for the previous example. Using the identical 15 time step sizes a second time, it can be seen that the residual norms for the two methods with iterates on $\mathcal{M}$ decrease further, while for the *Radau IA (3)* method it stagnates at the same level as after the first 15 steps. On the right hand side we observe that starting with larger time step sizes is advantageous for the methods with iterates on $\mathcal{M}$, as the decay of the residual norm is faster at the beginning (but results in the same final residual due to (3.31)). For the Radau IA (3) method however we do not observe any convergence.

The first numerical experiment shows the effects of Theorem 2. In the methods which satisfy (3.18), a direct connection between $\mathcal{L}(Z_jZ_j^{\mathsf{H}})$ and h_j exists, while for other methods the K_i from (3.17) are relevant, too. The second experiment was used to demonstrate that the methods with iterates on $\mathcal{M}$ are robust regarding the order choice of the time step sizes. That is, the order of the time step sizes does not alter the final approximation, in agreement with (3.31). On the other hand, in methods whose iterates do not lie on $\mathcal{M}$ no convergence takes place if too large time step sizes are chosen.

As stated in Theorem 4 the ADI iteration is equivalent to Runge-Kutta methods with iterates on $\mathcal{M}$. Our experiments indicate that these methods have properties which are advantageous over other Runge-Kutta methods for the approximate solution of the Lyapunov equation.

4. Sylvester equation

In this section we consider the Sylvester equation

$$A\mathcal{Y} - \mathcal{Y}B = FG^\mathsf{T} \tag{4.1}$$

with the system matrices $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{m\times m}$, $F \in \mathbb{R}^{n\times r}$ and $G \in \mathbb{R}^{m\times r}$, $r \leq n, m$. We assume that the spectra of A and B are disjoint, $\sigma(A) \cap \sigma(B) = \emptyset$, as then (4.1) has a unique solution.

Similar to the Lyapunov case where we only considered Lyapunov equations $A\mathcal{P} + \mathcal{P}A^\mathsf{T} = -bb^\mathsf{T}$ with rank-one right-hand side, we will restrict our discussion to the case $r = 1$, $F = f$, $G = g$ here. The case $r > 1$ with $F = [f_1, \cdots, f_r]$ and $G = [g_1, \cdots, g_r]$ can be reduced to $\mathcal{Y} = \sum_{i=1}^r \mathcal{Y}_i$ with $A\mathcal{Y}_i - \mathcal{Y}_i B = f_i g_i^\mathsf{T}, i = 1, \ldots, r$.

In analogy to (3.2) we consider the system of ODEs

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}Y(t) &= \hat{h}(t)\breve{h}(t)^\mathsf{T}, & Y(0) &= 0, \\ \frac{\mathrm{d}}{\mathrm{d}t}\hat{h}(t) &= A\hat{h}(t), & \hat{h}(0) &= f, \\ \frac{\mathrm{d}}{\mathrm{d}t}\breve{h}(t) &= -B^\mathsf{T}\breve{h}(t), & \breve{h}(0) &= -g. \end{aligned} \tag{4.2}$$

Please note that for $t \to \infty$ in general we will have $Y(t) \nrightarrow \mathcal{Y}$. Nonetheless, as we will see, (4.2) is useful in order to derive approximations to $\mathcal{Y}$.

Due to the product rule the second derivative of Y satisfies

$$\begin{aligned} \frac{\mathrm{d}^2}{\mathrm{d}t^2}Y(t) &= \frac{\mathrm{d}}{\mathrm{d}t}\hat{h}(t)\breve{h}(t)^\mathsf{T} \\ &= A\hat{h}(t)\breve{h}(t)^\mathsf{T} - \hat{h}(t)\breve{h}(t)^\mathsf{T}B \\ &= A\left(\frac{\mathrm{d}}{\mathrm{d}t}Y(t)\right) - \left(\frac{\mathrm{d}}{\mathrm{d}t}Y(t)\right)B. \end{aligned}$$

By integrating both sides over the interval $[0, t]$ we find

$$\frac{\mathrm{d}}{\mathrm{d}t}Y(t) = AY(t) - Y(t)B - fg^\mathsf{T}.$$

Thus the solution of the system of ODEs (4.2) does not satisfy (4.1) exactly, a rank one residual does remain

$$AY(t) - Y(t)B - fg^{\mathsf{T}} = \hat{h}(t)\breve{h}(t)^{\mathsf{T}}.$$

4.1. Approximating $\mathcal{Y}$ by Runge-Kutta methods

Unlike in the previous section, we propose here to solve the partitioned system of ODEs (4.2) in two main steps. First, the latter two equations for $\hat{h}(t)$ and $\breve{h}(t)$ are solved, then the equation for $Y(t)$ is solved. Each of the three ODEs will be solved by a different method. We will make use of three different s-stage Runge-Kutta methods: The function $Y(t)$ is approximated using the Butcher tableau with $\Lambda^{(j)} \in \mathbb{C}^{s\times s}$ and $\beta^{(j)} \in \mathbb{C}^{s}$, the function $\hat{h}(t)$ with $\hat{\Lambda}^{(j)} \in \mathbb{C}^{s\times s}$ and $\hat{\beta}^{(j)} \in \mathbb{C}^{s}$ and the function $\breve{h}(t)$ with $\breve{\Lambda}^{(j)} \in \mathbb{C}^{s\times s}$ and $\breve{\beta}^{(j)} \in \mathbb{C}^{s}$. As before, the upper index j is used to denote the jth step of the iteration and w.l.o.g. the time step sizes are all set to $\omega_j = 1$.

In analogy to the derivation in the Lyapunov case we find that the application of the Runge-Kutta methods for $\hat{h}(t)$ and $\breve{h}(t)$ lead to the iterations

$$\hat{h}_j = \hat{h}_{j-1} + \hat{K}_j\hat{\beta}^{(j)}, \tag{4.3}$$

$$\breve{h}_j = \breve{h}_{j-1} + \breve{K}_j\breve{\beta}^{(j)}, \tag{4.4}$$

with $\hat{h}_0 = f$, $\breve{h}_0 = -g$ (see (3.12)) and

$$\hat{K}_j = \left[A\hat{h}_{j-1}, \dots, A\hat{h}_{j-1}\right] + A\hat{K}_j(\hat{\Lambda}^{(j)})^{\mathsf{T}}, \tag{4.5}$$

$$\breve{K}_j = \left[-B^{\mathsf{T}}\breve{h}_{j-1}, \dots, -B^{\mathsf{T}}\breve{h}_{j-1}\right] - B^{\mathsf{T}}\breve{K}_j(\breve{\Lambda}^{(j)})^{\mathsf{T}}, \tag{4.6}$$

(see (3.7)) where $\hat{K}_j = A\hat{\mathcal{H}}_j$ and additionally $\breve{K}_j = -B^{\mathsf{T}}\breve{\mathcal{H}}_j$ holds, with

$$\hat{\mathcal{H}}_j = \left[\hat{h}_{j-1}, \dots, \hat{h}_{j-1}\right] + \hat{K}_j(\hat{\Lambda}^{(j)})^{\mathsf{T}},$$

$$\breve{\mathcal{H}}_j = \left[\breve{h}_{j-1}, \dots, \breve{h}_{j-1}\right] + \breve{K}_j(\breve{\Lambda}^{(j)})^{\mathsf{T}}$$

(see (3.11)). The solution of (4.5) is unique if and only if

$$\hat{\mu}_p \neq \hat{\lambda}_q^{-1},$$

for all $\hat{\mu}_p \in \sigma(\hat{\Lambda}^{(j)})$ and all $\hat{\lambda}_q \in \sigma(A)$, while the solution of (4.6) is unique if and only if

$$\breve{\mu}_k \neq -\breve{\lambda}_\ell^{-1}$$

for all $\breve{\mu}_k \in \sigma(\breve{\Lambda}^{(j)})$ and all $\breve{\lambda}_\ell \in \sigma(B)$ (see (3.10)).

The jth iterate Y_j approximating the function $Y(t)$ is then given by

$$Y_j = Y_{j-1} + \hat{\mathcal{H}}_j \operatorname{diag}(\beta^{(j)}) \breve{\mathcal{H}}_j^{\mathsf{H}}$$

where $Y_0 = 0$ (see the derivations leading to (3.12), in particular note that as in (3.5), Λ does not appear as the right hand side of $Y'(t) = \hat{h}(t)\breve{h}(t)^{\mathsf{T}}$ is independent of $Y(t)$).

Following the ideas from the previous section, we rewrite Y_j in terms of two low rank factors and a diagonal matrix

$$Y_j = \hat{Z}_j \Gamma_j \breve{Z}_j^{\mathsf{H}}$$

with

$$\hat{Z}_j = \left[\hat{Z}_{j-1}, \hat{\mathcal{H}}_j\right], \quad \breve{Z}_j = \left[\breve{Z}_{j-1}, \breve{\mathcal{H}}_j\right], \quad \Gamma_j = \operatorname{diag}(\Gamma_{j-1}, \beta^{(j)}).$$

The solution of the Sylvester equation is neither symmetric nor positive definite, so the factors $\hat{Z}_j$ and $\breve{Z}_j$ need not be equal and Γ_j may contain arbitrary complex valued entries.

Employing our main idea from the previous section we will only consider Runge-Kutta methods whose iterates preserve the low rank property of the Sylvester residual, that is

$$AY_j - Y_j B - fg^{\mathsf{T}} = \hat{h}_j \breve{h}_j^{\mathsf{T}}. \tag{4.7}$$

Thus, following the arguments in Section 3.7, the iterates Y_j will not necessarily approximate the function $Y(t)$ very well, but Y_j should be a good approximation to $\mathcal{Y}$ when the residual $\hat{h}_j \breve{h}_j^{\mathsf{T}}$ is small.

4.2. Runge-Kutta methods which preserve (4.7)

To characterize the tableaus which ensure that all iterates fulfill (4.7), we insert the first iterates $Y_1, \hat{h}_1$ and $\breve{h}_1$ into (4.7). For the left hand side we obtain

$$\begin{aligned}
AY_1 - Y_1 B - fg^{\mathsf{T}} &= A(Y_0 + \hat{\mathcal{H}}_1 \operatorname{diag}(\beta^{(1)}) \breve{\mathcal{H}}_1^{\mathsf{H}}) - (Y_0 + \hat{\mathcal{H}}_1 \operatorname{diag}(\beta^{(1)}) \breve{\mathcal{H}}_1^{\mathsf{H}}) B - fg^{\mathsf{T}} \\
&= A\hat{\mathcal{H}}_1 \operatorname{diag}(\beta^{(1)}) \breve{\mathcal{H}}_1^{\mathsf{H}} - \hat{\mathcal{H}}_1 \operatorname{diag}(\beta^{(1)}) \breve{\mathcal{H}}_1^{\mathsf{H}} B - fg^{\mathsf{T}} \\
&= \hat{K}_1 \operatorname{diag}(\beta^{(1)}) \breve{\mathcal{H}}_1^{\mathsf{H}} + \hat{\mathcal{H}}_1 \operatorname{diag}(\beta^{(1)}) \breve{K}_1^{\mathsf{H}} - fg^{\mathsf{T}} \\
&= \hat{K}_1 \operatorname{diag}(\beta^{(1)}) \left([\breve{h}_0, \ldots, \breve{h}_0] + \breve{K}_1 (\breve{\Lambda}^{(1)})^{\mathsf{T}}\right)^{\mathsf{H}} \\
&\quad + \left([\hat{h}_0, \ldots, \hat{h}_0] + \hat{K}_1 (\hat{\Lambda}^{(1)})^{\mathsf{T}}\right) \operatorname{diag}(\beta^{(1)}) \breve{K}_1^{\mathsf{H}} - fg^{\mathsf{T}} \\
&= \hat{K}_1 (\beta^{(1)}) \breve{h}_0^{\mathsf{H}} + \hat{K}_1 \operatorname{diag}(\beta^{(1)}) \overline{\breve{\Lambda}^{(1)}} \breve{K}_1^{\mathsf{H}} \\
&\quad + \hat{h}_0 (\beta^{(1)})^{\mathsf{T}} \breve{K}_1^{\mathsf{H}} + \hat{K}_1 (\hat{\Lambda}^{(1)})^{\mathsf{T}} \operatorname{diag}(\beta^{(1)}) \breve{K}_1^{\mathsf{H}} - fg^{\mathsf{T}},
\end{aligned}$$

while for the right hand side

$$\begin{aligned}\hat{h}_1\breve{h}_1^{\mathsf{H}} &= \left(\hat{h}_0 + \hat{K}_1\hat{\beta}^{(1)}\right)\left(\breve{h}_0 + \breve{K}_1\breve{\beta}^{(1)}\right)^{\mathsf{H}}\\ &= -fg^{\mathsf{H}} + \hat{h}_0(\breve{\beta}^{(1)})^{\mathsf{H}}\breve{K}_1^{\mathsf{H}}\\ &\quad + \hat{K}_1\hat{\beta}^{(1)}\breve{h}_0^{\mathsf{H}} + \hat{K}_1\hat{\beta}^{(1)}(\breve{\beta}^{(1)})^{\mathsf{H}}\breve{K}_1^{\mathsf{H}}\end{aligned}$$

holds. As g is real $g^{\mathsf{T}} = g^{\mathsf{H}}$ and so the equation

$$\begin{aligned}&\hat{K}_1\beta^{(1)}\breve{h}_0^{\mathsf{H}} + \hat{K}_1\operatorname{diag}(\beta^{(1)})\overline{\breve{\Lambda}^{(1)}}\breve{K}_1^{\mathsf{H}} + \hat{h}_0(\beta^{(1)})^{\mathsf{T}}\breve{K}_1^{\mathsf{H}} + \hat{K}_1(\hat{\Lambda}^{(1)})^{\mathsf{T}}\operatorname{diag}(\beta^{(1)})\breve{K}_1^{\mathsf{H}}\\ &\overset{!}{=} \hat{h}_0(\breve{\beta}^{(1)})^{\mathsf{H}}\breve{K}_1^{\mathsf{H}} + \hat{K}_1\hat{\beta}^{(1)}\breve{h}_0^{\mathsf{H}} + \hat{K}_1\hat{\beta}^{(1)}(\breve{\beta}^{(1)})^{\mathsf{H}}\breve{K}_1^{\mathsf{H}}\end{aligned}$$

has to hold in order to satisfy (4.7). This implies

$$\begin{aligned}0 &= \hat{h}_0\left(\breve{\beta}^{(1)} - \overline{\beta^{(1)}}\right)^{\mathsf{H}}\breve{K}_1^{\mathsf{H}} + \hat{K}_1\left(\hat{\beta}^{(1)} - \beta^{(1)}\right)\breve{h}_0^{\mathsf{H}}\\ &\quad + \hat{K}_1\left(\hat{\beta}^{(1)}(\breve{\beta}^{(1)})^{\mathsf{H}} - \operatorname{diag}(\beta^{(1)})\overline{\breve{\Lambda}^{(1)}} - (\hat{\Lambda}^{(1)})^{\mathsf{T}}\operatorname{diag}(\beta^{(1)})\right)\breve{K}_1^{\mathsf{H}},\end{aligned}$$

which is satisfied for $\hat{\beta}^{(1)} = \beta^{(1)}$, $\breve{\beta}^{(1)} = \overline{\beta^{(1)}}$, and

$$\operatorname{diag}(\beta^{(1)})\overline{\breve{\Lambda}^{(1)}} + (\hat{\Lambda}^{(1)})^{\mathsf{T}}\operatorname{diag}(\beta^{(1)}) - \beta^{(1)}(\beta^{(1)})^{\mathsf{T}} = 0.$$

Via induction this leads to the conditions

$$\begin{aligned}0 &= \operatorname{diag}(\beta^{(j)})\overline{\breve{\Lambda}^{(j)}} + (\hat{\Lambda}^{(j)})^{\mathsf{T}}\operatorname{diag}(\beta^{(j)}) - \beta^{(j)}(\beta^{(j)})^{\mathsf{T}},\\ \hat{\beta}^{(j)} &= \beta^{(j)},\\ \breve{\beta}^{(j)} &= \overline{\beta^{(j)}}.\end{aligned} \tag{4.8}$$

Please note that $\breve{\Lambda}^{(j)}, \breve{\beta}^{(j)}$ and $\hat{\Lambda}^{(j)}, \hat{\beta}^{(j)}$ from the Butcher tableaus for approximating $\breve{h}(t)$ and $\hat{h}(t)$ are relevant here, as well as $\beta^{(j)}$ from the Butcher tableau for $Y(t)$, but as mentioned above not the matrix $\Lambda^{(j)}$. DIRK tableaus which fulfill (4.8) are given by

$$\hat{\Lambda} = \begin{bmatrix}\hat{\mu}_1 & 0 & \cdots & 0\\ \beta_1 & \hat{\mu}_2 & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ \beta_1 & \beta_2 & \cdots & \hat{\mu}_s\end{bmatrix},\ \breve{\Lambda} = \begin{bmatrix}\overline{\breve{\mu}_1} & 0 & \cdots & 0\\ \overline{\beta_1} & \breve{\mu}_2 & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ \overline{\beta_1} & \overline{\beta_2} & \cdots & \breve{\mu}_s\end{bmatrix},\ \beta = \begin{bmatrix}\hat{\mu}_1 + \overline{\breve{\mu}_1}\\ \hat{\mu}_2 + \overline{\breve{\mu}_2}\\ \vdots\\ \hat{\mu}_s + \overline{\breve{\mu}_s}\end{bmatrix}. \tag{4.9}$$

4.3. Multiplicative update formulae for $\hat{h}_j$ and $\breve{h}_j$

Let us assume that (4.8) holds with $\beta_k^{(j)} \neq 0$, $k = 1, \ldots, s$. Further assume that A and B are diagonalizable, that is, $A = \hat{V}\hat{D}\hat{V}^{-1}$ and $B = \breve{V}\breve{D}\breve{V}^{-1}$ with $\hat{D} = \operatorname{diag}(\hat{\lambda}_1, \ldots, \hat{\lambda}_n)$ and $\breve{D} = \operatorname{diag}(\breve{\lambda}_1, \ldots, \breve{\lambda}_m)$. Then, as in Section 3.5 we find that the iterations (4.3) and (4.4) can be written in multiplicative form as in (3.27)

$$\hat{h}_j = \hat{V} \begin{bmatrix} \hat{R}_j(\hat{\lambda}_1) & & & \\ & \hat{R}_j(\hat{\lambda}_2) & & \\ & & \ddots & \\ & & & \hat{R}_j(\hat{\lambda}_n) \end{bmatrix} \hat{V}^{-1}\hat{h}_{j-1}, \tag{4.10}$$

$$\breve{h}_j = \breve{V}^{-\mathsf{T}} \begin{bmatrix} \breve{R}_j(-\breve{\lambda}_1) & & & \\ & \breve{R}_j(-\breve{\lambda}_2) & & \\ & & \ddots & \\ & & & \breve{R}_j(-\breve{\lambda}_m) \end{bmatrix} \breve{V}^{\mathsf{T}}\breve{h}_{j-1}, \tag{4.11}$$

with the stability functions

$$\begin{aligned} \hat{R}_j(z) &= 1 + z(\beta^{(j)})^{\mathsf{T}}(I - z\hat{\Lambda}^{(j)})^{-1}\mathbb{1}_s, \\ \breve{R}_j(z) &= 1 + z(\overline{\beta^{(j)}})^{\mathsf{T}}(I - z\breve{\Lambda}^{(j)})^{-1}\mathbb{1}_s. \end{aligned}$$

As in the Lyapunov case, these stability functions only depend on the eigenvalues of the tableaus $\hat{\Lambda}^{(j)}$ and $\breve{\Lambda}^{(j)}$. In order to see this, first observe that for $\beta_i^{(j)} \neq 0$, $i = 1, \ldots, s$, we find from (4.8) by multiplying with $\operatorname{diag}(\beta^{(j)})^{-1}$ from the right (respectively left)

$$\begin{aligned} 0 &= \operatorname{diag}(\beta^{(j)})\overline{\breve{\Lambda}^{(j)}}\operatorname{diag}(\beta^{(j)})^{-1} + (\hat{\Lambda}^{(j)})^{\mathsf{T}} - \beta^{(j)}\mathbb{1}_s^{\mathsf{T}}, \\ 0 &= \overline{\breve{\Lambda}^{(j)}} + \operatorname{diag}(\beta^{(j)})^{-1}(\hat{\Lambda}^{(j)})^{\mathsf{T}}\operatorname{diag}(\beta^{(j)}) - \mathbb{1}_s(\beta^{(j)})^{\mathsf{T}}. \end{aligned}$$

As any matrix is similar to its transpose, these equations imply the similarities

$$\begin{aligned} \hat{\Lambda}^{(j)} - \mathbb{1}_s(\beta^{(j)})^{\mathsf{T}} &\sim -\overline{\breve{\Lambda}^{(j)}}, \\ \breve{\Lambda}^{(j)} - \mathbb{1}_s(\beta^{(j)})^{\mathsf{H}} &\sim -\overline{\hat{\Lambda}^{(j)}}. \end{aligned} \tag{4.12}$$

Let $\hat{\Lambda}^{(j)}$ have eigenvalues $\hat{\mu}_1, \ldots, \hat{\mu}_s$ and let $\breve{\Lambda}^{(j)}$ have eigenvalues $\breve{\mu}_1, \ldots, \breve{\mu}_s$. Now with (2.6) and (4.12) we have (similar to (3.30))

$$\begin{aligned} \hat{R}_j(z) &= \frac{\det(I - z(\hat{\Lambda}^{(j)} - \mathbb{1}_s(\beta^{(j)})^{\mathsf{T}}))}{\det(I - z\hat{\Lambda}^{(j)})} = \frac{\det(I + z\overline{\breve{\Lambda}^{(j)}})}{\det(I - z\hat{\Lambda}^{(j)})} = \prod_{i=1}^{s}\frac{(1 + z\overline{\breve{\mu}_i})}{(1 - z\hat{\mu}_i)}, \\ \breve{R}_j(z) &= \frac{\det(I - z(\breve{\Lambda}^{(j)} - \mathbb{1}_s(\beta^{(j)})^{\mathsf{H}}))}{\det(I - z\breve{\Lambda}^{(j)})} = \frac{\det(I + z\overline{\hat{\Lambda}^{(j)}})}{\det(I - z\breve{\Lambda}^{(j)})} = \prod_{i=1}^{s}\frac{(1 + z\overline{\hat{\mu}_i})}{(1 - z\breve{\mu}_i)}. \end{aligned} \tag{4.13}$$

Thus, if (4.7) is enforced, then the stability functions of the Runge-Kutta method for $\hat{h}(t)$ and for $\breve{h}(t)$ are not independent of each other. Both depend on the eigenvalues of $\hat{\Lambda}^{(j)}$ and $\breve{\Lambda}^{(j)}$. Thus, also the iterates $\hat{h}_j$ (4.10) and $\breve{h}_j$ (4.11) depend on those eigenvalues.

The stability functions corresponding to the s-stage DIRK tableaus as in (4.9) will have the form (4.13). As in the previous section, we can restrict ourselves to the use of several different 1-stage Butcher tableaus satisfying (4.8), i.e. $\hat{\Lambda}^{(j)} = \hat{\mu}_j \in \mathbb{C}$ and $\breve{\Lambda}^{(j)} = \breve{\mu}_j \in \mathbb{C}$ with $\beta^{(j)} = \hat{\beta}^{(j)} = \hat{\mu}_j + \overline{\breve{\mu}_j} \in \mathbb{C}$ and $\overline{\beta^{(j)}} = \breve{\beta}^{(j)} = \overline{\hat{\mu}_j} + \breve{\mu}_j \in \mathbb{C}$.

The resulting iteration for a low rank approximation to the solution of the Sylvester equation (4.1) is summarized in Algorithm 4. It is equivalent to [21, Alg. 3.4] with $\alpha_j = -\breve{\mu}_j^{-1}$ and $\beta_j = \hat{\mu}_j^{-1}$ (and $E = I_n$, $C = I_m$, $r = 1$), in analogy to Theorem 3.

In the special case $B = -A^{\mathsf{T}}$ and $g = -f = b$ the Sylvester equation (4.1) reduces to the Lyapunov equation (1.1). If additionally for the parameters $\hat{\mu}_j = \breve{\mu}_j$, $j = 1, \dots, N$, holds, then Algorithm 4 reduces to Algorithm 3 and we find $\hat{\mu}_j + \overline{\breve{\mu}}_j = 2\,\mathrm{Re}(\hat{\mu}_j) \in \mathbb{R}$.

Algorithm 4 Low rank solution to (4.1) via 1-stage Runge-Kutta methods.

```
Input: $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{m\times m}$, $f \in \mathbb{R}^{n\times 1}$, $g \in \mathbb{R}^{m\times 1}$, parameters $\{\hat{\mu}_1, \dots, \hat{\mu}_N\} \subset \mathbb{C}$ and $\{\breve{\mu}_1, \dots, \breve{\mu}_N\} \subset \mathbb{C}$
Output: matrices $\hat{Z} \in \mathbb{C}^{n\times N}$, $\breve{Z} \in \mathbb{C}^{m\times N}$ and $\Gamma \in \mathbb{C}^{N\times N}$ with $\hat{Z}\Gamma\breve{Z}^{\mathsf{H}} \approx \mathcal{Y}$
 1: initialize $\hat{h}_0 = f$, $\breve{h}_0 = -g$, $\hat{Z}_0 = \breve{Z}_0 = \Gamma_0 = [\,]$
 2: for $j = 1, \dots, N$ do
 3:     solve $(I - \hat{\mu}_j A)\hat{K}_j = A\hat{h}_{j-1}$ for $\hat{K}_j$
 4:     solve $(I + \breve{\mu}_j B^{\mathsf{T}})\breve{K}_j = -B^{\mathsf{T}}\breve{h}_{j-1}$ for $\breve{K}_j$
 5:     $\hat{\mathcal{H}}_j = \hat{h}_{j-1} + \hat{\mu}_j \hat{K}_j$
 6:     $\breve{\mathcal{H}}_j = \breve{h}_{j-1} + \breve{\mu}_j \breve{K}_j$
 7:     update $\hat{Z}_j = [\hat{Z}_{j-1}, \hat{\mathcal{H}}_j]$
 8:     update $\breve{Z}_j = [\breve{Z}_{j-1}, \breve{\mathcal{H}}_j]$
 9:     update $\Gamma_j = \mathrm{diag}(\Gamma_{j-1}, (\hat{\mu}_j + \overline{\breve{\mu}}_j) I_r)$
10:     $\hat{h}_j = \hat{h}_{j-1} + (\hat{\mu}_j + \overline{\breve{\mu}}_j)\hat{K}_j$
11:     $\breve{h}_j = \breve{h}_{j-1} + (\overline{\hat{\mu}}_j + \breve{\mu}_j)\breve{K}_j$
12: end for
13: $\hat{Z} = \hat{Z}_N$, $\breve{Z} = \breve{Z}_N$, $\Gamma = \Gamma_N$
```

5. Conclusion

In this paper we presented a new derivation of residual based low rank iterations for the solution of (large-scale) linear matrix equations. For the Lyapunov equation a quadrature approach is evident, as for a stable system matrix the solution has an integral representation (1.2). We applied a Runge-Kutta method parameterized by an arbitrary and possibly complex valued Butcher tableau to approximate the solution of the system of ODEs (3.2), resulting in Algorithm 2. The Lyapunov residual which remains after each iteration step is determined. Runge-Kutta methods that preserve the rank of the initial residual were characterized. By making use of the stability function of a Runge-Kutta method it was shown that these methods are equivalent to DIRK methods and thus equivalent to Algorithm 3, which is itself equivalent to the ADI iteration.

All ideas are applied in slightly modified form to the Sylvester equation in Section 4. Although the solution of the system of ODEs (4.2) does not converge to the solution of the Sylvester equation, the application of Runge-Kutta methods which preserve the initial rank of the residual yields a sound approximation, justified by (4.10) and (4.11). That is, by retaining a geometric, qualitative property during the quadrature the ADI equivalent Algorithm 4 for the approximation of the solution of a Sylvester equation was derived from a system of ODEs.

Declaration of competing interest

There is no competing interest.

References

[1] A.C. Antoulas, Approximation of Large-Scale Dynamical Systems, Society for Industrial and Applied Mathematics, 2005.

[2] P. Benner, T. Breiten, Low rank methods for a class of generalized Lyapunov equations and related issues, Numer. Math. 124 (2013) 441–470, https://doi.org/10.1007/s00211-013-0521-0.

[3] P. Benner, T. Breiten, L. Feng, Matrix Functions and Matrix Equations, chapter: Matrix Equations and Model Reduction, World Scientific, 2015, pp. 50–75.

[4] P. Benner, P. Kürschner, J. Saak, Efficient handling of complex shift parameters in the low-rank Cholesky factor ADI method, Numer. Algorithms 62 (2012) 225–251, https://doi.org/10.1007/s11075-012-9569-7.

[5] P. Benner, P. Kürschner, J. Saak, An improved numerical method for balanced truncation for symmetric second-order systems, Math. Comput. Model. Dyn. Syst. 19 (2013) 593–615, https://doi.org/10.1080/13873954.2013.794363.

[6] P. Benner, R.C. Li, N. Truhar, On the ADI method for Sylvester equations, J. Comput. Appl. Math. 233 (2009) 1035–1045, https://doi.org/10.1016/j.cam.2009.08.108.

[7] P. Benner, J. Saak, A semi-discretized heat transfer model for optimal cooling of steel profiles, in: P. Benner, D.C. Sorensen, V. Mehrmann (Eds.), Dimension Reduction of Large-Scale Systems, Springer, Berlin, Heidelberg, 2005, pp. 353–356.

[8] C. Bertram, H. Faßbender, Lyapunov and Sylvester equations: a quadrature framework, arXiv:1903.05383, 2019.

[9] J. Butcher, Numerical Methods for Ordinary Differential Equations, Wiley, 2016.

[10] B. Datta, Linear and numerical linear algebra in control theory: some research problems, Linear Algebra Appl. 197/198 (1994) 755–790, https://doi.org/10.1016/0024-3795(94)90512-6.

[11] V. Druskin, L. Knizhnerman, V. Simoncini, Analysis of the rational Krylov subspace and ADI methods for solving the Lyapunov equation, SIAM J. Numer. Anal. 49 (2011) 1875–1898, https://doi.org/10.1137/100813257.

[12] Z. Gajić, M. Qureshi, Lyapunov Matrix Equation in System Stability and Control, Math. in Science and Engineering, Academic Press, San Diego, CA, 1995.

[13] G.H. Golub, C.F. van Loan, Matrix Computations, 4th ed., Johns Hopkins University Press, 2013.

[14] L. Grasedyck, Existence of a low rank or $\mathcal{H}$-matrix approximant to the solution of a Sylvester equation, Numer. Linear Algebra Appl. 11 (2004) 371–389, https://doi.org/10.1002/nla.366.

[15] E. Hairer, C. Lubich, G. Wanner, Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations, 2nd ed., Springer, 2006.

[16] E. Hairer, S. Norsett, G. Wanner, Solving Ordinary Differential Equations I, 2nd ed., Springer, 1993.

[17] E. Hairer, G. Wanner, Solving Ordinary Differential Equations II, 2nd ed., Springer, 1996.

[18] R. Horn, C. Johnson, Topics in Matrix Analysis, Cambridge University Press, Cambridge, 1991.

[19] A. Iserles, A First Course in the Numerical Analysis of Differential Equations, 2nd ed., Cambridge Texts in Applied Mathematics, Cambridge University Press, 2008.

[20] C.A. Kennedy, M.H. Carpenter, Diagonally Implicit Runge-Kutta Methods for Ordinary Differential Equations. A Review. Technical Memorandum NASA-TM-2016-219173. National Aeronautics and Space Administration Langley Research Center, Hampton, Virginia 23681-2199, 2016.

[21] P. Kürschner, Efficient Low-Rank Solution of Large-Scale Matrix Equations, Ph.D. thesis, Shaker Verlag, Aachen, ISBN 978-3-8440-4385-3, 2016.

[22] P. Lancaster, L. Rodman, The Algebraic Riccati Equation, Oxford University Press, Oxford, 1995.

[23] J. Li, J. White, Low rank solution of Lyapunov equations, SIAM J. Matrix Anal. Appl. 24 (2002) 260–280, https://doi.org/10.1137/S0895479801384937.

[24] A. Lu, E. Wachspress, Solution of Lyapunov equations by alternating direction implicit iteration, Comput. Math. Appl. 21 (1991) 43–58, https://doi.org/10.1016/0898-1221(91)90124-M.

[25] B. Moore, Principal component analysis in linear systems: controllability, observability, and model reduction, IEEE Trans. Autom. Control 26 (1981) 17–32, https://doi.org/10.1109/TAC.1981.1102568.

[26] C. Moosmann, E.B. Rudnyi, A. Greiner, J.G. Korvink, Model Order Reduction for Linear Convective Thermal Flow, Sophia Antipolis, France, 2004.
[27] M. Opmeer, Model order reduction by balanced proper orthogonal decomposition and by rational interpolation, IEEE Trans. Autom. Control 57 (2012) 472–477, https://doi.org/10.1109/tac.2011.2164018.
[28] D. Peaceman, H. Rachford Jr., The numerical solution of parabolic and elliptic differential equations, J. Soc. Ind. Appl. Math. 3 (1955) 28–41, https://doi.org/10.1137/0103003.
[29] C.W. Rowley, Model reduction for fluids, using balanced proper orthogonal decomposition. I, Int. J. Bifurc. Chaos 15 (2005) 997–1013, https://doi.org/10.1142/S0218127405012429.
[30] Y. Saad, Numerical solution of large Lyapunov equations, in: Signal Processing, Scattering and Operator Theory, and Numerical Methods, Proc. MTNS-89, Birkhauser, 1990, pp. 503–511.
[31] V. Simoncini, Computational methods for linear matrix equations, SIAM Rev. 58 (2016) 377–441, https://doi.org/10.1137/130912839.
[32] J.R. Singler, Convergent snapshot algorithms for infinite-dimensional Lyapunov equations, IMA J. Numer. Anal. 31 (2011) 1468–1496, https://doi.org/10.1093/imanum/drq028.
[33] T. Wolf, H. Panzer, The ADI iteration for Lyapunov equations implicitly performs H2 pseudo-optimal model order reduction, Int. J. Control 89 (2016) 481–493, https://doi.org/10.1080/00207179.2015.1081985.
[34] Z.H. Xiao, Y.L. Jiang, Z.Z. Qi, Finite-time balanced truncation for linear systems via shifted Legendre polynomials, Syst. Control Lett. 126 (2019) 48–57, https://doi.org/10.1016/j.sysconle.2019.03.004.