Lyapunov and Sylvester equations: 
A quadrature framework

Christian Bertram*, Heike Faßbender*

*Institut Computational Mathematics/ AG Numerik, TU Braunschweig, Universitätsplatz 2, 38106 Braunschweig, Germany

Abstract

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. As the general approach suggested here leads to complex valued computation even for real problems, we present a general realification approach based on similarity transformation.

Key words: Lyapunov equation, Sylvester equation, alternating direction method, Runge-Kutta method

2000 MSC: 65F30, 15A24, 93A15, 65L06, 34A26

1. Introduction

The numerical approximation of the solution of the continuous Lyapunov equation

\[ AP + PA^T + BB^T = 0 \]  \hspace{1cm} (1.1)

has been considered to great extend in the literature, see, e.g. the recent survey [30] and the references therein. Here a new framework based on methods for the numerical integration of ordinary differential equations (ODEs) is presented. 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 \( P \) 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

\[ P = \int_0^\infty e^{At}BB^T e^{A^Tt} \, dt \in \mathbb{R}^{n \times n}, \]  \hspace{1cm} (1.2)

*Corresponding author, Email ch.bertram@tu-braunschweig.de

Preprint March 14, 2019
Lyapunov equations play an important role in control and systems theory, see, e.g., [1, 8, 11]. In the context of linear time-invariant systems \( \dot{x} = Ax + Bu \), the solution \( P \) of (1.1) is called the controllability Gramian. It measures the energy transfer in the system.

We will also consider Sylvester equations

\[
AY - YB - FG^T = 0
\]

(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 \( Y \) is unique when the spectra of \( A \) and \( B \) are disjoint, i.e., \( \sigma(A) \cap \sigma(B) = \emptyset \), see, e.g., [21].

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 \( P \) of (1.1) can be approximated by a low rank approximation in the sense that \( P \approx ZZ^T \) with a rectangular \( n \times N \) matrix \( Z, N \ll n \). \( Z \) 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 \( Y \approx \hat{Z}\hat{Z}^T \) with rectangular \( n \times N \) matrices \( \hat{Z}, \hat{Z}, N \ll n \) and a diagonal matrix \( \Gamma \), see, e.g., [6, 13].

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 [30].

A popular algorithm for deriving low rank Cholesky factors for Lyapunov and Sylvester equations is the alternate directions implicit (ADI) method. It was developed to solve linear systems of equations in [26] and modified to approximate the solution \( P \) of Lyapunov equations in [22] (see [3, Sec. 7] for a short or [20, Chp. 3.1-3.2.1] for a more detailed derivation). In [22] the iteration was reformulated such that low rank approximations \( Z_tZ_t^T \) to the solution \( P \) 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 \( L(P_j) = AP_j + P_jA^T + BB^T \) will remain. This residual can be used to determine convergence of the iterative process. In [20, Alg. 3.2], the ADI iteration was further manipulated in order to allow for a fast evaluation of the residual norm \( \| L(P_j) \| \). This is known as the residual-based ADI method. An alternative derivation of this formulation utilizing Krylov subspaces can be found in [32]. The ADI iteration was also adapted to Sylvester equations, see [6, 20, Chp. 3.3].

Another type of methods for the solution of Lyapunov equations is making use of empirical Gramians [24]. The empirical Gramian essentially involves a sum approximation of the integral (1.2) \( P = \sum_j \delta_j g(t_j) \) for \( g(t) = e^{At}BB^Te^{At} \), arbitrary times \( t_j \) and appropriate quadrature weights \( \delta_j \). Usually, the identity \( g(t) = h(t)h(t)^T \) 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) \( \frac{d}{dt} h(t) = Ah(t) \) with initial value \( h(0) = B \). Any numerical integration scheme can be used to do so. Related quadrature approaches are discussed, e.g., in [29, 51]. Empirical Gramians are mainly used in model order reduction via balanced proper orthogonal decomposition (POD), see [28].
Krylov subspaces and moment matching, see [10, 25, 32]. In [25] 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

\[ P(t) = \int_0^t e^{At}BB^T e^{At} d\tau. \]

For \( t \to \infty \), \( P(t) \) will approximate the solution \( P \) 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 \( L(P(t)) = AP(t) + P(t)A^T + BB^T = h(t)h(t)^T \) 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. [18, Chp. 5], [14]. 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.

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 a 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 \( L(P(T)) \). 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. Finally, in Section 3.7 the choice of (complex-valued) shifts is discussed, while in Section 3.8 the realification of the potentially complex arithmetic involving algorithm is considered. 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 equa-
tions. 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., [17, 12] 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

$$(X \otimes Y)(V \otimes W) = XV \otimes YW$$
$$P(X \otimes Y)Q^T = Y \otimes X$$

(2.1)

for suitable $V, W$ and the perfect shuffle permutation matrices $P$ and $Q$, see, e.g. [12, Chp. 1.3.6]. In particular, for square matrices $X$ and $Y$ (that is, $r = s$ and $p = q$),
we have $P = Q = I_{rp}[(1:p rp) \ (2:p rp) \ \cdots \ (p:p rp)]$ where $I_{rp}$ denotes the $rp \times rp$
identity matrix and MATLAB\textsuperscript{®} 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: let $s = 1$, then $P(X \otimes Y)I_q = Y \otimes X$.

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

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

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

$$\text{vec}(XYZ) = (Z^T \otimes X) \text{vec}(Y)$$

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

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

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 $Y$ exists for all $FG^T$ if and only if $A$ and $B$ have no common
eigenvalues, see, e.g., [17, 21].
2.1. Numerical integration

There are numerous methods for the numerical solution of ordinary differential equations of the type
\[
\frac{dy}{dt}(t) = f(t, y(t)), \quad y(0) = y_0.
\] (2.2)
see, e.g., [15, 16]. Here \( f : \mathbb{R} \times \mathbb{R}^n \rightarrow \mathbb{R}^n \) is a given function, \( y_0 \in \mathbb{R}^n \) is a given initial value and one is interested in computing the function \( y : \mathbb{R} \rightarrow \mathbb{R}^n \) in the interval \([0, t_{\text{end}}]\).

Single-step methods make use of the fact that
\[
y(t_j) = y(t_{j-1}) + \int_{t_{j-1}}^{t_j} f(t, y(t)) \, dt
\]
holds for \( j = 1, 2, \ldots, N \) 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., [7, 14, 15, 16]) which are defined via
\[
y_j = y_{j-1} + \omega_j \sum_{i=1}^{s} \beta_i k_i^{(j)}, \quad j = 1, \ldots, N, \quad (2.3)
\]
\[
k_i^{(j)} = f(t_{j-1} + \gamma_i \omega_j, y_{j-1} + \omega_j \sum_{\ell=1}^{s} \lambda_{i\ell} k_{\ell}^{(j)}), \quad i = 1, \ldots, s, \quad (2.4)
\]
for certain \( \beta_i \in \mathbb{C}, \gamma_i \in \mathbb{R}, i = 1, \ldots, s \) and \( \lambda_{i\ell} \in \mathbb{C}, i, \ell = 1, \ldots, s \). Please note that we allow for complex-valued \( \lambda_{i\ell} \) and \( \beta_i \) unlike the standard definition of Runge-Kutta methods for the solution of (2.2). Moreover, \( \omega_j := t_j - t_{j-1} > 0, j = 1, \ldots, N \), denotes the time step size. Often Runge-Kutta methods are given in short hand by the so-called Butcher tableau
\[
\begin{array}{c|cccccc}
\gamma_1 & \lambda_{11} & \lambda_{12} & \cdots & \lambda_{1s} \\
\gamma_2 & \lambda_{21} & \lambda_{22} & \cdots & \lambda_{2s} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\gamma_s & \lambda_{s1} & \lambda_{s2} & \cdots & \lambda_{ss} \\
\hline 
\beta_1 & \beta_2 & \cdots & \beta_s
\end{array}
\] (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 \( \Lambda \) 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 \( \Lambda \) is a lower triangular matrix [19]. This uncouples the system of equations to be solved into a sequence of \( s \) systems.
The function
\[ R(z) = 1 + z\beta^T (I_s - z\Lambda)^{-1} \mathbb{1}_s = \frac{\det(I_s - z(\Lambda - \mathbb{1}_s\beta^T))}{\det(I_s - z\Lambda)} \] (2.6)
is called the stability function of the Runge-Kutta method given by [2.5]. Here, \( \mathbb{1}_s \) denotes the vector \( \mathbb{1}_s = (1, \ldots, 1)^T \) 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 \( \text{Re}(z) < 0 \) 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 [20, 5]. The goal is to approximate the \( n \times n \) solution \( P \) of the Lyapunov equation (1.1) in factored form \( ZZ^T \approx P \), 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
\[
\begin{align*}
X_0 &= 0, \\
(A + \alpha_j I_n)X_j &= -BB^T - X_{j-1}(A^T - \alpha_j I_n), \\
(A + \alpha_j I_n)X_j^H &= -BB^T - X_j^{H\frac{1}{2}} (A^T - \alpha_j I_n),
\end{align*}
\]
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 \( P \). Reformulating this iteration into a single step, writing the iterates \( X_j = Z_jZ_j^H \) in factored form and applying some algebraic manipulations (see [20 Chp. 3.2] for the details) one obtains the iteration
\[
\begin{align*}
V_1 &= (A + \alpha_1 I_n)^{-1}B, Z_1 = \sqrt{-2\text{Re}(\alpha_1)}V_1, \\
V_j &= V_{j-1} - (\alpha_j + \overline{\alpha_{j-1}})(A + \alpha_j I_n)^{-1}V_{j-1}, \\
Z_j &= [Z_{j-1}, \sqrt{-2\text{Re}(\alpha_j)}V_j].
\end{align*}
\]

With the findings from [20 Chp. 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.

---

**Algorithm 1** Low rank ADI iteration [20, Alg. 3.2, \( E = I_n \)]

**Input:** \( A \in \mathbb{R}^{n \times n} \) stable, \( B \in \mathbb{R}^{n \times m} \), parameters \( \{\alpha_1, \ldots, \alpha_N\} \in \mathbb{C}_- \)

**Output:** \( Z \in \mathbb{C}^{n \times mN} \) with \( ZZ^H \approx P \)

1. initialize \( W_0 = B, Z_0 = [ ] \)
2. for \( j = 1, \ldots, N \) do
   3. solve \((A + \alpha_j I_n)V_j = W_{j-1}\) for \( V_j \)
   4. \( W_j = W_{j-1} - 2\text{Re}(\alpha_j)V_j \)
   5. update \( Z_j = [Z_{j-1}, \sqrt{-2\text{Re}(\alpha_j)}V_j] \)
3. end for
4. \( Z = Z_N \)

---

As the shift parameters \( \alpha_j \) 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.

We will briefly present the realification approach from [4] in the revised form of [20, Chp. 4.1.4]. It is denoted M4 in [20, Chp. 4.1.5]. Let $\alpha_j$ be a complex shift and $\alpha_{j+1} = \overline{\alpha_j}$. Instead of performing two separate steps, one with the shift $\alpha_j$ and one with $\alpha_{j+1}$, a double step involving $\alpha_j$ and $\alpha_{j+1}$ will be used. That is, the block $\hat{Z} = \sqrt{-2 \text{Re}(\alpha_j)} [V_j \quad V_{j+1}]$ of $2m$ columns is added to the current iterate $Z_{j-1}$. This is still a complex matrix, but it can be replaced by a real one. For this, note that

$$V_{j+1} = V_j + \frac{2 \text{Re}(\alpha_j)}{\text{Im}(\alpha_j)} \text{Im}(V_j).$$

Let

$$J = \begin{bmatrix} 1 & 1 \\ i & \text{Re}(\alpha_j) \text{Im}(\alpha_j) - i \end{bmatrix} \quad \text{and} \quad L = \sqrt{2} \begin{bmatrix} 1 & 0 \\ \text{Re}(\alpha_j) \text{Im}(\alpha_j) & \sqrt{\text{Re}(\alpha_j)^2 + 1} \end{bmatrix}. \quad (2.7)$$

Then the block $\hat{Z}$ of $2m$ columns that is added to the factor $Z_{j-1}$ can be written as

$$\hat{Z} = \sqrt{-2 \text{Re}(\alpha_j)} \left[ \text{Re}(V_j) \quad \text{Im}(V_j) \right] (J \otimes I_m).$$

Observing that

$$J J^H = \begin{bmatrix} 2 \text{Re}(\alpha_j) \text{Im}(\alpha_j) & 2 \text{Re}(\alpha_j) \text{Im}(\alpha_j)^2 + 2 \\ 2 \text{Re}(\alpha_j) \text{Im}(\alpha_j)^2 - 2 \text{Re}(\alpha_j) \text{Im}(\alpha_j)^2 + 2 \end{bmatrix} = LL^H,$$

we can use $\hat{Z} = \sqrt{-2 \text{Re}(\alpha_j)} \left[ \text{Re}(V_j) \quad \text{Im}(V_j) \right] (L \otimes I_m)$ instead of $\hat{Z}$, as $\hat{Z} \hat{Z}^H = \hat{Z} \hat{Z}^H$.

Altogether, to keep the iterates real, the columns

$$\sqrt{-2 \text{Re}(\alpha_j)} \cdot \sqrt{2} \left[ \text{Re}(V_j) + \frac{\text{Re}(\alpha_j)}{\text{Im}(\alpha_j)} \text{Im}(V_j) \sqrt{\text{Re}(\alpha_j)^2 + 1} \cdot \text{Im}(V_j) \right]$$

are added to the iterate $Z_{j-1}$, yielding the same approximation $Z_{j+1}$ as two steps of Algorithm [1] with shifts $\alpha_j$ and $\alpha_{j+1} = \overline{\alpha_j}$, but with real approximate Cholesky factors.

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., [20] for a detailed description of a residual based variant or [30] 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 $P$ (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 $B = [b_1, \ldots, b_m] \in \mathbb{R}^{n \times m}$
\( \mathbb{R}^{n \times m} \) can be reduced to
\[
P = \int_0^\infty e^{At} B B^T e^{At} \, dt = \sum_{i=1}^m \int_0^\infty e^{At} b_i b_i^T e^{At} \, dt. \tag{3.1}
\]

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
\[
P(t) := \int_0^t e^{A\tau} b b^T e^{A\tau} \, d\tau = \int_0^t h(\tau) h(\tau)^T \, d\tau
\]
where \( h(t) := e^{A t} b. \) The functions \( P(t) \) and \( h(t) \) are the solutions of the system of ODEs
\[
\frac{d}{dt} P(t) = h(t) h(t)^T, \quad P(0) = 0, \quad \frac{d}{dt} h(t) = A h(t), \quad h(0) = b. \tag{3.2}
\]

Vectorizing (3.2) allows to write the system in the form (2.2)
\[
\frac{d}{dt} \begin{bmatrix} \text{vec}(P(t)) \vline h(t) \end{bmatrix} = \begin{bmatrix} \text{vec}(h(t) h(t)^T) \\ A h(t) \end{bmatrix} = f(t, y(t)) \tag{3.3}
\]
for the high dimensional solution function \( y(t) = [\text{vec}(P(t))^T \ h(t)^T]^T \), \( y: \mathbb{R} \to \mathbb{R}^{n^2 + n} \) which needs to be determined. Clearly, \( y(0) = [0^T \ b^T]^T \).

### 3.1. Approximating \( P \) 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 \( y(t) = [v(t)^T \ h(t)^T]^T \), that is, \( v(t) = \text{vec}(P(t)) \).

Moreover, the vector \( k^{(j)}_t \) (2.4) is written in block form
\[
k^{(j)}_t = \begin{bmatrix} \hat{k}^{(j)}_t \\ \tilde{k}^{(j)}_t \end{bmatrix}
\]
corresponding to the two blocks of \( y(t) = [v(t)^T \ h(t)^T]^T \). We obtain
\[
v_j = v_{j-1} + \omega_j \sum_{i=1}^s \beta_i \hat{k}^{(j)}_{i}, \quad v_0 = 0 \in \mathbb{R}^{n^2},
\]
\[
h_j = h_{j-1} + \omega_j \sum_{i=1}^s \beta_i \tilde{k}^{(j)}_{i}, \quad h_0 = b \in \mathbb{R}^n,
\]
with
\[
k^{(j)}_t \begin{bmatrix} \hat{k}^{(j)}_t \\ \tilde{k}^{(j)}_t \end{bmatrix} = f \left( t, h_{j-1} + \omega_j \sum_{i=1}^s \lambda_i \tilde{k}^{(j)}_{i} \right) = \begin{bmatrix} \text{vec} \left( \left( h_{j-1} + \omega_j \sum_{i=1}^s \lambda_i \tilde{k}^{(j)}_{i} \right) (h_{j-1} + \omega_j \sum_{i=1}^s \lambda_i \tilde{k}^{(j)}_{i})^T \right) \\ A h_{j-1} + \omega_j \sum_{i=1}^s \lambda_i \tilde{A} \tilde{k}^{(j)}_{i} \end{bmatrix}. \tag{3.5}
\]
Please note the use of $H$ instead of $T$ in the expression above due to the possibly complex valued $\lambda_{ij}$. 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 $\Lambda$ and $\beta$ 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 $v_j = v_{j-1} + \omega_j \sum_{i=1}^{s} \beta_i \hat{k}_i^{(j)}$. The matrix $\Lambda$ does not appear as the right hand side of $P'(t) = h(t)h(t)^T$ 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 $\hat{\beta} = \beta$.

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

$$P_j = P_{j-1} + \omega_j \sum_{i=1}^{s} \beta_i b_i^{(j)}(b_i^{(j)})^H, \quad j = 1, \ldots, N$$

(3.6)

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

$$b_i^{(j)} = h_{j-1} + \omega_j \sum_{\ell=1}^{s} \lambda_{i\ell} \hat{k}_\ell^{(j)},$$

$$\hat{k}_i^{(j)} = Ah_{j-1} + \omega_j \sum_{\ell=1}^{s} \lambda_{i\ell} Ak_\ell^{(j)} = Ah_i^{(j)}$$

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

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

$$K_j = [Ah_{j-1}, \ldots, Ah_{j-1}] + \omega_j AK_j h_{j-1}. \quad (3.7)$$

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

$$(I_{ns} - \omega_j (\Lambda \otimes A)) \text{vec}(K_j) = [(Ah_{j-1})^T, \ldots, (Ah_{j-1})^T]^T \quad (3.8)$$

(3.9)

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

$$\mu_p \neq \frac{1}{\omega_j \lambda_q} \quad (3.10)$$

for all $p = 1, \ldots, s$ and $q = 1, \ldots, n$. Thus, in case we require $\Lambda$ 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 $\mathcal{H}_j = [b_1^{(j)}, \ldots, b_s^{(j)}]$. This implies that

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

(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 $I_{n s} - \omega_j (\Lambda \otimes A)$ as above. This system of equations can be solved uniquely if and only if the eigenvalues of $\Lambda$ and $A$ satisfy (3.10). In particular, note that, as for (3.8), we can not choose $\Lambda$ arbitrarily, at least (3.10) has to hold.

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

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

$$h_j = h_{j-1} + \omega_j K_j \beta,$$

(3.12)

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

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

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

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

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

$$Z_j Z_j^H = Z_{j-1} Z_{j-1}^H + \omega_j \mathcal{H}_j \text{diag}(\beta) \mathcal{H}_j^H$$

(3.13)

$$= \left[Z_{j-1}, \mathcal{H}_j \text{diag}(\omega_j \beta) \frac{1}{2}\right] \left[Z_{j-1}, \mathcal{H}_j \text{diag}(\omega_j \beta) \frac{1}{2}\right]^H$$

where $\text{diag}(\omega_j \beta) \frac{1}{2} \in \mathbb{C}^{s \times s}$ 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

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

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 that $\beta \in \mathbb{R}_+^s$, such that $P_j$ is positive semi-definite and that the eigenvalues of $\Lambda$ 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.

10
Algorithm 2 Low rank solution to (1.1) via an s-stage Runge-Kutta method

\textbf{Input:} $A \in \mathbb{R}^{n \times n}$ stable, $b \in \mathbb{R}^n$, positive time step sizes $\{\omega_1, \ldots, \omega_N\}$ and a Butcher tableau (2.5) with $\Lambda \in \mathbb{C}^{s \times s}$ and $\beta \in \mathbb{R}_+^s$ which satisfies (3.10)

\textbf{Output:} $Z \in \mathbb{C}^{n \times sN}$ with $ZZ^H \approx \mathcal{P}$

\begin{enumerate}
  \item initialize $h_0 = b$, $Z_0 = []$
  \item for $j = 1, \ldots, N$ do
    \begin{enumerate}
      \item solve $K_j = [Ah_{j-1}, \ldots, Ah_{j-1}] + \omega_j A K_j A^T$ for $K_j \in \mathbb{C}^{n \times s}$
      \item $H = [h_{j-1}, \ldots, h_{j-1}] + \omega_j K_j A^T$
      \item update $Z_j = [Z_{j-1}, H \ diag(\omega_j \beta)]$
      \item $h_j = h_{j-1} + \omega_j K_j \beta$
    \end{enumerate}
  \item $Z = Z_N$
\end{enumerate}

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

$$L(t) = AP(t) + P(t)A^T + bb^T.$$ 

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

$$\frac{d}{dt} P(t) = h(t) h(t)^T$$

$$= h(0) h(0)^T + \int_0^t \frac{d}{d\tau} h(\tau) h(\tau)^T \ d\tau$$

$$= bb^T + \int_0^t (Ah(\tau) h(\tau)^T + h(\tau) h(\tau)^T A^T) \ d\tau$$

$$= bb^T + A \int_0^t h(\tau) h(\tau)^T \ d\tau + \int_0^t h(\tau) h(\tau)^T A^T$$

$$= bb^T + AP(t) + P(t)A^T.$$ 

Thus, we have

$$L(t) = AP(t) + P(t)A^T + bb^T = h(t) h(t)^T,$$ 

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 $P_j = Z_j Z_j^H$ and $h_j$ satisfy the algebraic invariant (3.14) in the sense

$$AP_j + P_j A^T + bb^T = h_j h_j^H.$$ 

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).

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

$$\mathcal{M} \coloneqq \{(P, h) \mid P \in \mathbb{C}^{n \times n}, h \in \mathbb{C}^{n \times 1}, AP + PA^T + bb^T - hh^H = 0\}.$$ 

11
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 \( P_N \approx P \) is good when the iterate \( h_N \) is small, because then the tuple \( (P_N, h_N) \in \mathcal{M} \) is located close to \( (P, 0) \in \mathcal{M} \).

**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

\[
AP_j + P_j A^T + bb^T = h_j h_j^H + \sum_{i=1}^{j} \omega_i^2 K_i \left( \text{diag}(\beta)\overline{X} + \Lambda^T \text{diag}(\beta) - \beta \beta^T \right) K_i^H
\]

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

\[
\text{diag}(\beta)\overline{X} + \Lambda^T \text{diag}(\beta) - \beta \beta^T = 0
\]

holds.

**Proof.** For \( j = 0 \) the statement is obviously true as \( P_0 = Z_0 Z_0^T = 0 \) and \( h_0 = b \).

For \( j \in \mathbb{N} \) we first use (3.13) and then inductively (3.16) for \( j - 1 \)

\[
AP_j + P_j A^T + bb^T = AP_{j-1} + P_{j-1} A^T + bb^T + \omega_j A H_j \text{diag}(\beta) H_j^H + \omega_j H_j \text{diag}(\beta) H_j^H A^T
\]

\[
= h_{j-1} h_{j-1}^H + \sum_{i=1}^{j-1} \omega_i^2 K_i \left( \text{diag}(\beta)\overline{X} + (\text{diag}(\beta)A)^T - \beta \beta^T \right) K_i^H + \omega_j A H_j \text{diag}(\beta) H_j^H + \omega_j H_j \text{diag}(\beta) H_j^H A^T.
\]

Figure 1: Solution of \( \dot{h} = Ah, \dot{P} = hh^T \) and iterates from Algorithm 2 evolving on the rank-one residual manifold \( \mathcal{M} \).
Using $A H_j = K_j$ and expanding $H_j$ as in (3.11) we obtain

$$A H_j \text{ diag}(\beta) H_j^T = K_j \text{ diag}(\beta) (W_{j-1} + \omega_j K_j \Lambda^T)^H$$

$$= K_j \beta h_{j-1}^H + \omega_j K_j \text{ diag}(\beta) \bar{A} K_j^H.$$ 

Inserting this in (3.18) and adding a zero we find

$$AP_j + P_j A^T + b b^T$$

$$= h_{j-1} h_j^H - 1 + \sum_{i=1}^{j-1} \omega_i^2 K_i \left( \text{diag}(\beta) \bar{A} + \text{diag}(\beta) \Lambda^T \right) - \beta \beta^T \right) K_i^H$$

$$+ \omega_j K_j \beta h_{j-1}^H + \omega_j h_{j-1} \beta^T K_j^H + \omega_j^2 K_j \beta \beta^T K_j^H$$

$$= h_{j-1} h_j^H - 1 + \omega_j K_j \beta h_{j-1}^H + \omega_j h_{j-1} \beta^T K_j^H + \omega_j^2 K_j \beta \beta^T K_j^H$$

$$+ \omega_j^2 K_j \left( \text{diag}(\beta) \bar{A} + \text{diag}(\beta) \Lambda^T \right) - \beta \beta^T \right) K_j^H$$

$$+ \sum_{i=1}^{j-1} \omega_i^2 K_i \left( \text{diag}(\beta) \bar{A} + \text{diag}(\beta) \Lambda^T \right) - \beta \beta^T \right) K_i^H$$

$$= h_j h_j^H + \sum_{i=1}^{j} \omega_i^2 K_i \left( \text{diag}(\beta) \bar{A} + \text{diag}(\beta) \Lambda^T \right) - \beta \beta^T \right) K_i^H,$$

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 $i = 1, \ldots, j$, the second statement is immediate. This concludes the proof.

There are numerous Butcher tableaus for which (3.17) holds. First, observe that the diagonal entries of the equation $\text{diag}(\beta) \bar{A} + \text{diag}(\beta) \Lambda^T - \beta \beta^T = 0$ imply that

$$\beta_j \bar{A}_{jj} + \beta_j \Lambda_{jj} - \beta_j^2 = \beta_j (2 \text{Re}(\Lambda_{jj}) - \beta_j) = 0$$

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

$$\beta_j = 2 \text{Re}(\Lambda_{jj}), \quad j = 1, \ldots, s.$$

As $\beta_j \in \mathbb{R}_+$ is required, this implies that the diagonal elements of $\Lambda$ have to be in $\mathbb{C}_+$ whenever (3.17) is required.

The simplest 1-stage Butcher tableaus satisfying (3.17) are given for an arbitrary (complex) number $\mu \in \mathbb{C}$ by

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

The implicit midpoint rule

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

is a prominent example of such a 1-stage tableau. A simple 2-stage tableau which satisfies (3.17) is the 2-stage implicit Runge-Kutta method

$$\Lambda = \left[ \begin{array}{ccc} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{array} \right], \quad \beta = \left[ \begin{array}{c} \frac{1}{2} \\ \frac{1}{2} \end{array} \right].$$
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 (3.17), see [18, Lemma 5.3] and the subsequent corollary, where the matrix $M$ corresponds to the left-hand side of (3.17).

Another family of methods for which (3.17) holds is given by DIRK methods of the form

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

with $\mu_1, \ldots, \mu_s \in \mathbb{C}_+$. In this case, it is easy to verify whether the necessary and sufficient condition (3.10) is satisfied for all $p = 1, \ldots, s$ and $q = 1, \ldots, n$. As $\lambda_q$ is an eigenvalue of the stable matrix $A$, we have $\text{Re}(\lambda_q) < 0$. Thus, for stable $A$ any DIRK method with a tableau (3.19) 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 $\Lambda, \beta$ in every iteration step. The time step sizes $\omega_j$ for $j = 1, \ldots, N$ 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

$$
y_j = y_{j-1} + \omega_j \sum_{i=1}^s \beta_i^{(j)} k_i^{(j)}, \quad j = 1, \ldots, N,
$$

$$
k_i^{(j)} = f \left( t_{j-1} + \gamma_i^{(j)}, y_{j-1} + \omega_j \sum_{\ell=1}^s \Lambda_{i\ell}^{(j)} k_{\ell}^{(j)} \right), \quad i = 1, \ldots, s,
$$

instead of (2.3) and (2.4).

Clearly, we would like to choose $\Lambda(j), \beta(j)$ such that (3.17) is satisfied. In that case, $\omega_j \Lambda(j), \omega_j \beta(j)$ also satisfy (3.17). 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 $j = 1, \ldots, N$, whenever we allow for different $\Lambda(j), \beta(j)$ in every iteration step,
Algorithm 2 has to be modified accordingly. In step 3 and 4 the term $\omega_jA$ has to be replaced by $\Lambda^{(i)}$, while in step 5 and 6 the term $\omega_j\beta$ has to be replaced by $\beta^{(i)}$. Apparently Theorem 2 remains true even when different tableaus are used in every step as long as (3.17) holds for $\Lambda^{(i)}$ and $\beta^{(i)}$, $i = 1, \ldots, j$. In its proof the tableaus $\Lambda$, $\beta$ 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 $\Lambda^{(1)} = \mu_j \in \mathbb{C}_+$ and $\beta^{(1)} = 2 \text{Re}(\mu_j) \in \mathbb{R}_+$. With the condition $\text{Re}(\mu_j) > 0$ we make sure (3.10) is satisfied and all linear systems solves in Algorithm 2 will have a unique solution.

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

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

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 $\{\mu_1, \ldots, \mu_N\} \subset \mathbb{C}_+$

**Output:** factor $Z \in \mathbb{C}^{n \times N}$ with $ZZ^H \approx P$

1: initialize $h_0 = b$, $Z_0 = [ ]$
2: for $j = 1, \ldots, N$ do
3: solve $(I - \mu_j A) K_j = Ah_{j-1}$ for $K_j \in \mathbb{C}^{n \times 1}$
4: $H_j = h_{j-1} + \mu_j K_j$
5: update $Z_j = [Z_{j-1}, \sqrt{2 \text{Re}(\mu_j) H_j}]$
6: $h_j = h_{j-1} + 2 \text{Re}(\mu_j) K_j$
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 $j = 1, \ldots, N$. Let $W_0 = b$ and $V_j$, $j = 1, \ldots, N$ be determined by Algorithm 1. Let $h_0 = b$ and $h_j$, $j = 1, \ldots, N$ be determined by Algorithm 3. Then $W_j = c_j h_j$ and $\sqrt{-2 \text{Re}(\alpha_j)V_j} = d_j \sqrt{2 \text{Re}(\mu_j) H_j}$ holds for some constants $c_j, d_j \in \mathbb{C}$ with $|c_j| = |d_j| = 1$. Thus the approximation $Z_j Z_j^H \approx P$ 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 $\omega_j A = \mu_j$ we find $H_j = h_{j-1} + \mu_j A H_j$ and thus

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

Making use of step 4 in Algorithm 3 we obtain

$$(I_n - \mu_j A)^{-1} h_{j-1} = H_j = h_{j-1} + \mu_j K_j. \quad (3.20)$$
Via induction we find from $W_{j-1} = c_{j-1}h_{j-1}$ for $W_j$ as in Algorithm 3,
\[ W_j = W_{j-1} - 2 \text{Re}(\alpha_j)V_j \]
\[ = W_{j-1} - 2 \text{Re}(\alpha_j)(A + \alpha_jI_n)^{-1}W_{j-1} \]
\[ = c_{j-1}h_{j-1} - 2 \text{Re}(-\mu_j^{-1})(A - \mu_j^{-1}I_n)^{-1}c_{j-1}h_{j-1} \]
\[ = c_{j-1}(h_{j-1} + 2 \text{Re}(\mu_j)^{-1})(I_n - \mu_jA)^{-1}h_{j-1}) \]
\[ = c_{j-1}(h_{j-1} + 2 \text{Re}(-\mu_j)^{-1}c_{j-1}h_{j-1}) + 2 \text{Re}(\mu_j)K_j) \]
\[ = c_{j-1}(1 - 2 \text{Re}(\mu_j^{-1})h_{j-1} - 2 \text{Re}(\mu_j)^{-1})c_{j-1}h_{j-1} + 2 \text{Re}(\mu_j)K_j) \]
\[ = -c_{j-1} \frac{\mu_j}{\mu_j} (h_{j-1} + 2 \text{Re}(\mu_j)K_j) \]
\[ = c_jh_j \]
with $1 - 2 \text{Re}(\mu_j^{-1})\mu_j = -\frac{\mu_j}{\mu_j^2}, -2 \text{Re}(\mu_j^{-1})\mu_j^2 = -\frac{\mu_j}{\mu_j^2}2 \text{Re}(\mu_j)$ and $c_j := -c_{j-1}\frac{\mu_j}{\mu_j^2}$.

Further we find for $V_j$ as in Algorithm 1,
\[ \sqrt{-2 \text{Re}(\alpha_j)V_j} = \sqrt{-2 \text{Re}(\alpha_j)(A + \alpha_jI_n)^{-1}W_{j-1}} \]
\[ = \sqrt{2 \text{Re}(\mu_j^{-1})\mu_jA - I_n)^{-1}W_{j-1}} \]
\[ = -\sqrt{2 \text{Re}(\mu_j)|\mu_j|^{-2}h_{j-1} + 2 \text{Re}(\mu_j)} \]
\[ = d_j \sqrt{2 \text{Re}(\mu_j)h_{j-1}} \]
with $d_j := -\frac{\mu_j}{|\mu_j|}c_{j-1}$. The observation $c_j\overline{v_j} = d_j\overline{a_j} = 1$ concludes the proof.\[ \square \]

As Algorithm 3 is equivalent to Algorithm 1, all properties of Algorithm 1 hold for Algorithm 3 as well. In particular, complex valued iterates can be avoided if the set of parameters $\{\mu_j\}_{j=1}^N$ is proper. See Section 2.2, 274, 272, Chp. 4.3, 44 or 20, Chp. 4.1.4 for more.

In order to derive Algorithm 3, we have used different $1 \times 1$-tableaux in each iteration step. In general, larger tableaux 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 tableaux can be reduced to Algorithm 3. In particular, $s$ steps with different $1 \times 1$-tableaux $\Lambda^{(i)} = \mu_i$ and $\beta^{(i)} = 2 \text{Re}(\mu_i), i = 1, \ldots, s$ (as in Algorithm 3, $N = s$) are equivalent to one step with a particular $s$-stage DIRK tableau with $\Lambda$ and $\beta$ (as in Algorithm 2). For $s$ steps of Algorithm 3 we have from step 6
\[ h_s = h_{s-1} + 2 \text{Re}(\mu_s)K_s = h_0 + \sum_{i=1}^s 2 \text{Re}(\mu_i)K_i \]
\[ = h_0 + [K_1, \ldots, K_s] \begin{bmatrix} 2 \text{Re}(\mu_1) \\ \vdots \\ 2 \text{Re}(\mu_s) \end{bmatrix} \]
(3.21)
with
\[ K_i = Ah_{i-1} + \mu_i AK_i = Ah_0 + \sum_{j=1}^{i-1} 2 \operatorname{Re}(\mu_j) AK_j + \mu_i AK_i \in \mathbb{C}^{n \times 1} \] (3.22)

for \(i = 1, \ldots, s\). Merging (3.22) for \(i = 1, \ldots, s\) into one equation yields
\[
[K_1, \ldots, K_s] = [Ah_0, \ldots, Ah_0] + [AK_1, \ldots, AK_s]
\]

(3.23)

Hence (3.21) and (3.22) are identical to step 6 and step 3 of Algorithm 2 with \(\omega_1 = 1\), where \([K_1, \ldots, K_s] \in \mathbb{C}^{n \times s}\) from (3.23) corresponds to the \(n \times s\) matrix \(K_1\) in step 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 (3.19).

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 \(\mu_j \in \mathbb{C}_+\), \(j = 1, \ldots, s\) be given. Let \(\Lambda \in \mathbb{C}^{s \times s}\) and \(\beta \in \mathbb{R}^s_+\) be as in (3.19). One step of Algorithm 2 with \(\Lambda, \beta\) 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 step 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 \(\Lambda\) and \(\beta\) 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
\[
K_j^\beta = ((\beta(j))^T \otimes I_n) \text{vec}(K_j)
\]
\[
= ((\beta(j))^T \otimes I_n)(I_{ns} - \Lambda(j) \otimes A)^{-1}(I_s \otimes A)(I_s \otimes I_n)h_{j-1}
\]
\[
= ((\beta(j))^T \otimes I_n) \left\{ (I_s^{-1} \otimes A^{-1})(I_{ns} - \Lambda(j) \otimes A) \right\}^{-1} (I_s \otimes I_n)h_{j-1}
\]
\[
= ((\beta(j))^T \otimes I_n) \left\{ (I_s^{-1} \otimes A^{-1}) - \Lambda(j) \otimes I_n \right\}^{-1} (I_s \otimes I_n)h_{j-1}
\]
\[
= ((\beta(j))^T \otimes I_n) \left\{ (I_{ns} - \Lambda(j) \otimes A)(I_s^{-1} \otimes A^{-1}) \right\}^{-1} (I_s \otimes I_n)h_{j-1}
\]
\[
= ((\beta(j))^T \otimes I_n)(I_s \otimes A)(I_{ns} - \Lambda(j) \otimes A)^{-1}(I_s \otimes I_n)h_{j-1}
\]
\[
= ((\beta(j))^T \otimes A)(I_{ns} - \Lambda(j) \otimes A)^{-1}(I_s \otimes I_n)h_{j-1}.
\]

With (2.1) we have \( \Lambda(j) \otimes A = Q(A \otimes \Lambda(j))Q^T \) for a perfect shuffle permutation matrix \( Q \in \mathbb{R}^{ns \times ns} \). Moreover, as \( Q \) is orthogonal, it holds that \( I_n \otimes I_s = I_{ns} = QQ^T = Q(I_s \otimes I_n)Q^T \). Hence,
\[
K_j^\beta = ((\beta(j))^T \otimes A) \left\{ Q(I_n \otimes I_s - A \otimes \Lambda(j))Q^T \right\}^{-1} (I_s \otimes I_n)h_{j-1}
\]
\[
= I_n ((\beta(j))^T \otimes A)Q \left\{ I_n \otimes I_s - A \otimes \Lambda(j) \right\}^{-1} Q^T(I_s \otimes I_n)h_{j-1}
\]
\[
= (A \otimes (\beta(j))^T) \left( I_{ns} - A \otimes \Lambda(j) \right)^{-1} (I_n \otimes I_s)h_{j-1}
\]
as the perfect shuffle matrix for a vector is the identity.

Thus the iterate \( h_j \) is obtained from \( h_{j-1} \) via
\[
h_j = M_jh_{j-1} \tag{3.24}
\]
with the iteration matrix
\[
M_j := M_j(A) = I_n + (A \otimes (\beta(j))^T) \left( I_{ns} - A \otimes \Lambda(j) \right)^{-1} (I_n \otimes I_s). \tag{3.25}
\]
The matrix valued function \( M(z) = I_n + (z \otimes \beta^T)(I_{ns} - z \otimes \Lambda)^{-1} (I_n \otimes I_s) \) can be viewed as a generalization of the stability function \( R(z) = 1 + z\beta^T(I - z\Lambda)^{-1} I_s \) 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 \( D = \text{diag}(\lambda_1, \ldots, \lambda_n) \) containing the eigenvalues of \( A \) on the diagonal, the iteration matrix \( M_j \) from (3.25) simplifies considerably. We define the stability function \( R_{(j)}(z) = 1 + z(\beta(j))^T(I - z\Lambda(j))^{-1} I_s \) for a Runge-Kutta method with
\( M_j \) and \( \beta^{(j)} \) and see

\[
M_j = I_n + (A \otimes (\beta^{(j)})^T) \left( I_n - (A \otimes \Lambda^{(j)}) \right)^{-1} (I_n \otimes I_s)
\]

\[
= I_n + V (D \otimes (\beta^{(j)})^T) (V^{-1} \otimes I_s)
\]

\[
= I_n + V (D \otimes (\beta^{(j)})^T) \left( I_n \otimes I_s - (D \otimes \Lambda^{(j)}) (V^{-1} \otimes I_s) \right)^{-1} (I_n \otimes I_s) V^{-1}
\]

\[
= V \left( I_n + \begin{bmatrix}
\lambda_1 (\beta^{(j)})^T & \cdots & \lambda_n (\beta^{(j)})^T
\end{bmatrix}
\begin{bmatrix}
I_s - \lambda_1 \Lambda^{(j)} & \cdots & \cdots & I_s - \lambda_n \Lambda^{(j)}
\end{bmatrix}^{-1}
\begin{bmatrix}
1_s & \cdots & 1_s
\end{bmatrix}
\right) V^{-1}
\]

\[
= V \left( 1 + \lambda_1 (\beta^{(j)})^T (I_s - \lambda_1 \Lambda^{(j)})^{-1} 1_s
\right)
\]

\[
= V \left( R^{(j)}(\lambda_1) \right)
\]

\[
= V \left( \cdots \right)
\]

\[
= V \left( R^{(j)}(\lambda_n) \right)
\]

\[
= V^{-1},
\]

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, \( M_j \)—a composition of continuous functions—depends continuously on \( A \) and 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 (3.26) for each of the different methods. In case the tableaus satisfy (3.15), then

\[
AZ_j Z_j^H + Z_j Z_j^H A^T + (bb^T - h_j h_j^H) = 0
\]

holds, and \( Z_j Z_j^H \) (but not \( Z_j \)) is determined by \( h_j \) via (3.27). Thus, all tableaus with the same stability function yield the same approximation \( P_j = Z_j Z_j^H \) (as long as the tableaus satisfy (3.10), (3.17) and \( \beta \in \mathbb{R}^+ \)).

3.6. Runge-Kutta methods satisfying (3.17)

We now investigate when two \( s \)-stage Runge-Kutta methods have the same stability function. Assume that \( \Lambda \) and \( \beta \) are as in Theorem 2 and satisfy (3.17), \( 0 = \text{diag}(\beta) \Lambda + \Lambda^T \text{diag}(\beta) - \beta \beta^T \). Then

\[
-\Lambda = \text{diag}(\beta)^{-1} (\Lambda^T - \beta I_s^T) \text{diag}(\beta)
\]

as \( \beta_j > 0, j = 1, \ldots, s \), proving that the matrices \( -\Lambda \) and \( \Lambda - I_s \beta^T \) are similar, i.e.

\[
-\Lambda \sim \Lambda - I_s \beta^T.
\]

(3.28)
Let $\Lambda$ have eigenvalues $\mu_1, \ldots, \mu_s$. Then with the determinant based characterization of the stability function \((2.6)\) we find that
\[
R(z) = \frac{\det(I + z(\Lambda^T - \Lambda))}{\det(I - z\Lambda)} = \frac{\det(I + z\tilde{\Lambda})}{\det(I - z\Lambda)}
\]
\[
= \frac{(1 + \mu_1 z) \cdots (1 + \mu_s z)}{(1 - \mu_1 z) \cdots (1 - \mu_s z)}
\]
holds. This is just the stability function of a DIRK method as given in \((3.19)\). Thus any method based on a Butcher tableau with $\Lambda, \beta$ such that $\Lambda$ has the eigenvalues $\mu_1, \ldots, \mu_s$ and $\beta \in \mathbb{R}^n_+$ satisfies \((3.17)\) is equivalent to a DIRK method \((3.19)\). Note that obviously the order of the parameters $\mu_i$ is irrelevant. We summarize our findings in the following theorem.

**Theorem 5.** Let $\Lambda \in \mathbb{R}^{n \times n}$ 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 \((3.17)\) be given. Let $\Lambda$ have eigenvalues $\mu_1, \ldots, \mu_s \in \mathbb{C}_+$, so \((3.10)\) is guaranteed. Moreover, \((3.28)\) holds. Then the method based on this tableau is equivalent to the DIRK method given in \((3.19)\) and therefore equivalent to $s$ steps with the $1$-stage tableaus $\Lambda^{(j)} = \mu_j$ and $\beta^{(j)} = 2\text{Re}(\mu_j)$ 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 $\tilde{\Lambda} \in \mathbb{C}^{s \times s}$, $\tilde{\beta} \in \mathbb{R}^s_+$ be given such that $\Lambda$ and $\tilde{\Lambda}$ have the same eigenvalues, $\sigma(\Lambda) = \tilde{\sigma}(\tilde{\Lambda})$. Further assume that $\Lambda, \beta$ are chosen such that \((3.17), (3.10)\) and \((3.28)\) hold. Please note that Theorem 5 does not imply that the method based on the Butcher tableau with $\Lambda, \tilde{\Lambda}$ is equivalent to the one based on $\Lambda, \beta$. Only in case $\Lambda, \tilde{\Lambda}$ also satisfies \((3.17), (3.10)\) and \((3.28)\), 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 $\Lambda^{(1)} = \mu$, $\beta^{(1)} = 2\text{Re}(\mu)$ and $\Lambda^{(2)} = \pi$, $\beta^{(2)} = 2\text{Re}(\mu)$ where $\mu \in \mathbb{C}$ is chosen such that $\text{Re}(\mu) \neq 0$ and \((3.10)\) holds. This is equivalent to one step of the method based on the DIRK method with
\[
\Lambda = \begin{bmatrix}
\mu & 0 \\
2\text{Re}(\mu) & \beta
\end{bmatrix} \in \mathbb{C}^{2 \times 2}, \quad \beta = \begin{bmatrix}
2\text{Re}(\mu) \\
2\text{Re}(\mu)
\end{bmatrix} \in \mathbb{R}^2.
\]

The matrix $\Lambda$ is similar to
\[
\tilde{\Lambda} = \begin{bmatrix}
\pi & 0 \\
0 & \mu
\end{bmatrix} = V^{-1}\Lambda V \in \mathbb{C}^{2 \times 2}
\]
as well as to
\[
\tilde{\Lambda} = \begin{bmatrix}
\text{Re}(\mu) & -\text{Im}(\mu) \\
\text{Im}(\mu) & \text{Re}(\mu)
\end{bmatrix} \in \mathbb{R}^{2 \times 2}
\]
as $\tilde{\Lambda} = V^{-1}\Lambda V$ and $\tilde{\Lambda} = W^{-1}AW$ with $V = \begin{bmatrix} 0 & 1 \\ i & -i \end{bmatrix}$, $W = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & i \\ 0 & 2i \end{bmatrix}$ and $z = \frac{\text{Re}(\mu)}{\text{Im}(\mu)}$. A quick check reveals that for $\delta \in \mathbb{C}^2$
\[
0 = \text{diag}(\delta)\tilde{\Lambda} + \tilde{\Lambda}^T\text{diag}(\bar{\delta}) - \delta\delta^T
\]
as well as

\[ 0 = \text{diag}(\delta) \tilde{\Lambda} + \tilde{\Lambda}^T \text{diag}(\delta) - \delta^T \]

is satisfied only if either \( \delta_1 = 0 \) or \( \delta_2 = 0 \). Thus no method based on a Butcher tableau with \( \tilde{\Lambda} \) or \( \tilde{\Lambda} \) is equivalent to the method based on the Butcher tableau with \( \Lambda, \beta \).

Another matrix similar to \( \Lambda \) is given by

\[
\tilde{\Lambda} = \begin{bmatrix}
\text{Re}(\mu) & \text{Re}(\mu) + \varphi|\mu| \\
\text{Re}(\mu) - \varphi|\mu|
\end{bmatrix} \in \mathbb{R}^{2 \times 2}
\]

with \( \varphi = \text{sgn}(\text{Im}(\mu)) = \frac{\text{Im}(\mu)}{|\text{Im}(\mu)|} \). \( \tilde{\Lambda} \) is similar to \( \Lambda \) as \( \tilde{\Lambda} = S^{-1} \tilde{\Lambda} S \) with

\[
S = \frac{1}{\sqrt{2}} \begin{bmatrix}
1 & 1 \\
1 & -1
\end{bmatrix} \sqrt{2} \begin{bmatrix}
1 & z \\
0 & \sqrt{z^2 + 1}
\end{bmatrix}
\]

and thus similar to \( \tilde{\Lambda} = T^{-1} \tilde{\Lambda} T \) with \( T = WS^{-1} = \frac{1}{2\sqrt{2}} \frac{1}{\sqrt{z^2 + 1}} \begin{bmatrix}
\sqrt{z^2 + 1} & \sqrt{z^2 + 1} \\
-2z & -2z
\end{bmatrix} \).

Please note that \( L \) in (3.32) is essentially the same as \( L \) in (2.7). Choosing

\[
\tilde{\beta} = \begin{bmatrix}
2\text{Re}(\mu) \\
2\text{Re}(\mu)
\end{bmatrix}
\]

we find that \( \tilde{\Lambda}, \tilde{\beta} \) satisfy (3.17). Due to the choice of \( \mu \), (3.10) and (3.28) also hold. Thus the methods based on \( \Lambda, \beta \) and on \( \tilde{\Lambda}, \tilde{\beta} \) are equivalent.

Finally, please note, that given \( \Lambda, \beta \) satisfying (3.17) and \( \tilde{\Lambda} = U^{-1} \Lambda U \) with a regular matrix \( U \in \mathbb{C}^{2 \times 2} \) the condition \( 0 = \text{diag}(\beta) \tilde{\Lambda} + \tilde{\Lambda}^T \text{diag}(\beta) - \beta\beta^T \) can not be used to determine whether or not \( \tilde{\beta} \) exists such that (3.17) is satisfied for \( \tilde{\Lambda}, \tilde{\beta} \). We have

\[
0 = (U^T \text{diag}(\beta) U) (U^{-1} \Lambda U) + (U^T \Lambda U^{-T}) (U^T \text{diag}(\beta) U) - U^T \beta\beta^T U
\]

where \( Y = U^T \text{diag}(\beta) U \). Unfortunately, in general \( Y \) will not be diagonal (this is readily checked for \( U = V \) and \( \tilde{\Lambda} = \tilde{\Lambda} \) or \( U = T \) and \( \tilde{\Lambda} = \tilde{\Lambda} \)). Moreover, in general, \( U^T \beta \neq \beta^T U \). Thus, considering the above transformation of (3.17) does not help in order to determine an appropriate \( \tilde{\beta} \) such that \( \tilde{\Lambda}, \tilde{\beta} \) satisfy (3.17). One needs to check \( 0 = \text{diag}(\beta) \tilde{\Lambda} + \tilde{\Lambda}^T \text{diag}(\beta) - \beta\beta^T \) 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 \( h_N h_N^T \). Hence for a small residual the norm \( \|h_N\| \) has to be small.
Consider Algorithm 3, that is, \( R(j)(z) = \frac{1 + \mu_j z}{1 - \mu_j z} \). Then with (3.24) and (3.26) we find

\[
h_N = \left( \prod_{j=1}^{N} M_j \right) b
= V \left( \prod_{j=1}^{N} \begin{bmatrix} R(j)(\lambda_1) & \cdots & R(j)(\lambda_n) \end{bmatrix} \right) V^{-1} b
= V \left( \prod_{j=1}^{N} \begin{bmatrix} \frac{1 + \mu_j \lambda_1}{1 - \mu_j \lambda_1} & \cdots & \frac{1 + \mu_j \lambda_n}{1 - \mu_j \lambda_n} \end{bmatrix} \right) V^{-1} b.
\]

Taking the 2-norm this implies

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

To minimize this error bound the parameters \( \mu_j \) have to be chosen such that the problem

\[
\min_{\mu_1,\ldots,\mu_N} \max_{i=1,\ldots,n} \left( \prod_{j=1}^{N} \left| \frac{1 + \mu_j \lambda_i}{1 - \mu_j \lambda_i} \right| \right)
\]

is solved. It is equivalent to the rational min-max problem [5, Sec. 2.2]. For \( N = n \) the choice \( \mu_j = -\lambda_j^{-1} \), \( j = 1, \ldots, N \) yields \( h_N = 0 \) and thus \( P_N = P \). 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 \( \lambda_i \) of \( A \), i.e. \( \mu_j \approx -\lambda_i^{-1} \).

As \( \omega_j = 1 \) in Algorithm 3 and due to (3.10), the choice of \( \mu_j \) 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. Realification

We show that when using a proper set of shifts in Algorithm 3 we can avoid complex arithmetic in Algorithm 3 forced by the use of complex parameters \( \mu_j \). We follow the idea discussed in Section 2.2 and combine two (complex) iteration steps of the algorithm to a real one. Thereto Theorem 5 and the discussion thereafter is utilized. That is instead of two steps with the 1-stage tableaus \( \Lambda(j) = \mu, \beta(j) = 2 \text{Re}(\mu) \) and \( \Lambda(j+1) = \mu, \beta(j+1) = 2 \text{Re}(\mu) \) for \( \mu \in \mathbb{C}_+ \) only one step of Algorithm 3 is performed with a suitable real 2-stage Butcher tableau with \( \Lambda, \beta \) such that \( \sigma(\Lambda) = \{\mu, \mu^{-1}\} \) and meeting the requirements of
Theorem 5. As discussed in Section 3.6 the real 2-stage tableau with \( \tilde{\Lambda} \) and \( \tilde{\beta} \) as in (3.31) and (3.33) can be used here. We will do so in the following discussion.

In order to explain our realification idea, we will make use of the iteration based on

\[
H_j = [h_{j-1}, h_{j-1}] + A\tilde{H}_j\tilde{\Lambda}^T
\]  

(3.34)

as in (3.11). The ideas transfer to the iteration based on \( K_j \) from (3.7) which is used in Algorithm 2 easily. We further assume that all previous iteraes are real valued, in particular, \( h_{j-1} \in \mathbb{R}^n \) and \( P_{j-1} \in \mathbb{R}^{n \times n} \).

The iterate \( H_j = [b_1^{(j)}, b_2^{(j)}] \) can be obtained from (3.34) via vectorization

\[
\begin{bmatrix}
I_n - \text{Re}(\mu)A & -(\text{Re}(\mu) + \varphi|\mu|)A \\
- (\text{Re}(\mu) - \varphi|\mu|)A & I_n - \text{Re}(\mu)A
\end{bmatrix}
\begin{bmatrix}
b_1^{(j)} \\
b_2^{(j)}
\end{bmatrix}
= \begin{bmatrix}
h_{j-1} \\
h_{j-1}
\end{bmatrix}.
\]

(3.35)

Thus, instead of solving two complex \( n \times n \) systems in Algorithm 3 for the steps with \( \Lambda^{(j)} = \mu \) and \( \Lambda^{(j+1)} = \pi \) it is possible to solve one real \( 2n \times 2n \) system (3.35). As the system matrix as well as the right hand side is real, this gives \( H_j \in \mathbb{R}^{n \times n} \). This realification approach is comparable to the approaches \( M2^* \) and \( M4^* \) presented in [20] Chp. 4.1.5. However, it was shown in [20] that these approaches based on \( 2n \times 2n \) real linear systems are mostly outperformed by the approach \( M4 \) which is based on the solution of one complex \( n \times n \) system of linear equations. We summarized this approach in Section 2.2.

We continue our discussion with a modification of our approach such that instead of the \( 2n \times 2n \) real system (3.35) the solution of just one complex \( n \times n \) system of linear equations is employed. The idea is to compose the iterate \( H_j = [b_1^{(j)}, b_2^{(j)}] \) of the real and imaginary part of the solution of this complex system. For this purpose we intend to use the connection between a complex system of linear equations and its augmented real version, following the arguments in [9]. Consider real vectors \( v_1, v_2, y_1, y_2 \in \mathbb{R}^n \). Then the \( n \times n \) complex system

\[
(I_n - \mu A)(v_1 + iv_2) = y_1 + iy_2
\]

(3.36)

is equivalent to the \( 2n \times 2n \) real system

\[
\begin{bmatrix}
I_n - \text{Re}(\mu)A & \text{Im}(\mu)A \\
- \text{Im}(\mu)A & I_n - \text{Re}(\mu)A
\end{bmatrix}
\begin{bmatrix}
v_1 \\
v_2
\end{bmatrix}
= \begin{bmatrix}
y_1 \\
y_2
\end{bmatrix}.
\]

(3.37)

We now want to find a Butcher tableau leading to an iteration which is equivalent to the one with \( \tilde{\Lambda} \) and \( \tilde{\beta} \) and for which the equivalence of (3.36) and (3.37) can be used. Clearly, \( \tilde{\Lambda} \in \mathbb{R}^{2 \times 2} \) from (3.30) will lead to (3.37) when \( \tilde{H}_j = [h_{j-1}, h_{j-1}] + A\tilde{H}_j\tilde{\Lambda}^T \) (3.11) is vectorized. But as shown in Section 3.6 there is no appropriate \( \tilde{\beta} \) such that the requirements of Theorem 5 are met. Thus, we cannot claim that this is equivalent to the iteration with \( \tilde{\Lambda} \).

However, from (3.34) we obtain with \( \tilde{\Lambda} = S^{-1}\tilde{\Lambda}S \) and \( S \) as in (3.32)

\[
\tilde{H}_j' = [h_{j-1}, h_{j-1}]S^{-T} + A\tilde{H}_j'\tilde{\Lambda}^T
\]

(3.38)

for \( H_j' = [v_1, v_2] \in \mathbb{R}^{n \times 2} \) defined by \( \tilde{H}_j = H_j' S^T \). The vectorization of (3.38) gives the \( 2n \times 2n \) real linear system (3.37) with \( [y_1, y_2] = [h_{j-1}, h_{j-1}]S^{-T} \in \mathbb{R}^{n \times 2} \), which is
equivalent to the $n \times n$ complex system (3.36). Thus it is suggested to solve (3.36) for the right-hand side defined by $[y_1, y_2] = [h_{j-1}, h_{j-1}]S^{-T}$ in order to obtain $H_j = [v_1, v_2]S^T \in \mathbb{R}^{n \times 2}$.

Our discussion in the last paragraph is based on the choice $\hat{\Lambda} \in \mathbb{R}^{2 \times 2}$ as in (3.31) and the fact that it is similar to $\hat{\Lambda}$. Alternatively, any other Butcher tableau with $\hat{\Lambda}$ satisfying (3.17), (3.10) and (3.28) and with $\hat{\Lambda}$ similar to $\hat{\Lambda}$ can be chosen. This leads to other similarity transformations than $S$ to obtain $\Lambda$ and thus produces different realifications.

We conclude our discussion by taking a closer look at our approach with $\hat{\Lambda}$ from (3.31). As for $S$ as in (3.32) $[y_1, y_2] = [h_{j-1}, h_{j-1}]S^{-T} = [h_{j-1}, 0]$ holds, the complex system (3.36) to solve reduces to

$$(I_n - \mu A)(v_1 + i v_2) = h_{j-1}.$$ 

This system is just (3.11) for the choice $\Lambda^{(j)} = \mu$. It is also the same as step 3 in Algorithm 1 for $\alpha_j = -\mu^{-1}$. Next $H_j = [v_1, v_2]S^T$ has to be calculated for $S^T = LQ$ as in (3.32) (see also (2.7)). We note that multiplying the iterate $H_j$ from the right with the orthogonal matrix $Q \in \mathbb{R}^{2 \times 2}$ does not alter the approximation $P_j$, $(H_j Q) \text{diag}(\tilde{\beta})(H_j Q)^H = H_j \text{diag}(\tilde{\beta})H_j^H$,

as $\text{diag}(\tilde{\beta}) = 2 \text{Re}(\mu)I_2$. Therefore due to (3.12) and (3.13) we obtain the next approximate Cholesky factor $Z_j$ by appending

$$\sqrt{2 \text{Re}(\mu)}H_j^T = \sqrt{2 \text{Re}(\mu)}H_j^T S^T Q^T = \sqrt{2 \text{Re}(\mu)}H_j LQ^T = \sqrt{2 \text{Re}(\mu)}H_j L$$

$$= \sqrt{2 \text{Re}(\mu)} \sqrt{2 \begin{bmatrix} v_1 + \frac{\text{Re}(\mu)}{\text{Im}(\mu)} v_2, v_2 \end{bmatrix}} + \frac{\text{Re}(\mu)^2}{\text{Im}(\mu)^2} + 1 \cdot v_2$$

to $Z_{j-1}$. This assembly of the real and imaginary part has the same structure as in (2.8).

All in all we have presented two approaches to keep the iterates real. The former one directly uses Theorem 5 to replace two 1-stage complex tableaus by one real 2-stage tableau. The latter one exploits the connection between a complex system of linear equations and its augmented real version, employing similarity transformations.

4. Sylvester equation

In this section we consider the Sylvester equation

$$AY - YB = FG^T$$  \hspace{1cm} (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 $AP + PA^T = -bb^T$ 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 \( A\mathcal{Y}_i - \mathcal{Y}_i B = f_i g_i^T, i = 1, \ldots, r \).

In analogy to (3.2) we consider the system of ODEs
\[
\begin{align*}
\frac{d}{dt} Y(t) &= h(t) h(t)^T, \\
\frac{d}{dt} \hat{h}(t) &= Ah(t), \\
\frac{d}{dt} \hat{\alpha}(t) &= -B^T \hat{h}(t),
\end{align*}
\]
(4.2)

Please note that for \( t \to \infty \) in general we will have \( Y(t) \not\to \mathcal{Y} \). 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
\[
\begin{align*}
\frac{d^2}{dt^2} Y(t) &= \frac{d}{dt} \hat{h}(t) h(t)^T \\
&= A\hat{h}(t) h(t)^T - \hat{h}(t) h(t)^T B \\
&= A \left( \frac{d}{dt} Y(t) \right) - \left( \frac{d}{dt} Y(t) \right) B.
\end{align*}
\]

By integrating both sides over the interval [0, \( t \)] we find
\[
\frac{d}{dt} Y(t) = AY(t) - Y(t) B - fg^T.
\]

Thus the solution of the system of ODEs (4.2) does not satisfy (4.1) exactly, a rank one residual does remain
\[
AY(t) - Y(t) B - fg^T = \hat{h}(t) \hat{h}(t)^T.
\]

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 \( \hat{\alpha}(t) \) 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 \( \hat{\alpha}(t) \) with \( \hat{\Lambda}^{(j)} \in \mathbb{C}^{s \times s} \) and \( \hat{\beta}^{(j)} \in \mathbb{C}^s \). As before, the upper index \( j \) is used to denote the \( j \)th 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 \( \hat{\alpha}(t) \) lead to the iterations
\[
\hat{h}_j = \hat{h}_{j-1} + \hat{K}_j \hat{\beta}^{(j)},
\]
(4.3)
\[
\hat{\alpha}_j = \hat{\alpha}_{j-1} + \hat{K}_j \hat{\beta}^{(j)},
\]
(4.4)

25
with \( \hat{h}_0 = f, \hat{h}_0 = -g \) (see (3.12)) and
\[
\hat{K}_j = \begin{bmatrix} \hat{A}_{h_{j-1}}, \ldots, \hat{A}_{h_{j-1}} \end{bmatrix} + A\hat{K}_j(\hat{\Lambda}^{(j)})^T, \quad (4.5)
\]
\[
\check{K}_j = \begin{bmatrix} -B^T\check{h}_{j-1}, \ldots, -B^T\check{h}_{j-1} \end{bmatrix} - B^T\check{K}_j(\check{\Lambda}^{(j)})^T, \quad (4.6)
\]
(see (3.7)) where \( \hat{K}_j = A\hat{H}_j \) and additionally \( \check{K}_j = -B^T\check{H}_j \) holds, with
\[
\hat{H}_j = \begin{bmatrix} \hat{h}_{j-1}, \ldots, \hat{h}_{j-1} \end{bmatrix} + \hat{K}_j(\hat{\Lambda}^{(j)})^T, \\
\check{H}_j = \begin{bmatrix} \check{h}_{j-1}, \ldots, \check{h}_{j-1} \end{bmatrix} + \check{K}_j(\check{\Lambda}^{(j)})^T
\]
(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
\[
\check{\mu}_k \neq \check{\lambda}_\ell^{-1}
\]
for all \( \check{\mu}_k \in \sigma(\check{\Lambda}^{(j)}) \) and all \( \check{\lambda}_\ell \in \sigma(B) \) (see (3.10)).

The \( j \)th iterate \( Y_j \) approximating the function \( Y(t) \) is then given by
\[
Y_j = Y_{j-1} + \hat{H}_j \text{diag}(\beta^{(j)}) \check{H}_j^H
\]
where \( Y_0 = 0 \) (see the derivations leading to (3.12), in particular note that as in (3.5), \( \Lambda \) does not appear as the right hand side of \( Y'(t) = \hat{h}(t)\hat{h}(t)^T \) 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
\[
Y_j = \hat{Z}_j \Gamma_j \check{Z}_j^H
\]
with
\[
\hat{Z}_j = \begin{bmatrix} \hat{Z}_{j-1}, \hat{H}_j \end{bmatrix}, \quad \check{Z}_j = \begin{bmatrix} \check{Z}_{j-1}, \check{H}_j \end{bmatrix}, \quad \Gamma_j = \text{diag}(\Gamma_{j-1}, \beta^{(j)}).
\]
The solution of the Sylvester equation is neither symmetric nor positive definite, so the factors \( \hat{Z}_j \) and \( \check{Z}_j \) need not be equal and \( \Gamma_j \) 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
\[
AY_j - Y_j B - fg^T = \hat{h}_j \check{h}_j^T. \quad (4.7)
\]
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 \( Y \) when the residual \( h_j h_j^T \) 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 $\hat{h}_1$ into (4.7). For the left hand side we obtain

\[
AY_1 - Y_1 B - fg^T = A(Y_0 + \hat{\mathbf{R}}_1 \text{diag}(\beta^{(1)}))\hat{Y}_1^\mathbf{H} - (Y_0 + \hat{\mathbf{R}}_1 \text{diag}(\beta^{(1)}))\hat{Y}_1^\mathbf{H} B - fg^T \\
= A\hat{\mathbf{R}}_1 \text{diag}(\beta^{(1)})\hat{Y}_1^\mathbf{H} - \hat{\mathbf{R}}_1 \text{diag}(\beta^{(1)})\hat{Y}_1^\mathbf{H} B - fg^T \\
= \hat{K}_1 \text{diag}(\beta^{(1)})\hat{Y}_1^\mathbf{H} + \hat{\mathbf{R}}_1 \text{diag}(\beta^{(1)})\hat{K}_1^\mathbf{H} - fg^T \\
= \hat{K}_1 \text{diag}(\beta^{(1)})\left(\hat{h}_0, \ldots, \hat{h}_0\right) + \hat{K}_1(\hat{\Lambda}^{(1)})^\mathbf{T} + \hat{K}_1 \text{diag}(\beta^{(1)})\hat{Y}_1^\mathbf{H} - fg^T \\
+ \hat{h}_0(\beta^{(1)})^\mathbf{T} \hat{K}_1^\mathbf{H} + \hat{K}_1(\hat{\Lambda}^{(1)})^\mathbf{T} \text{diag}(\beta^{(1)})\hat{K}_1^\mathbf{H} - fg^T,
\]

while for the right hand side

\[
\hat{h}_1\hat{h}_1^\mathbf{H} = \left(\hat{h}_0 + \hat{K}_1 \hat{\beta}^{(1)}\right)\left(\hat{h}_0 + \hat{K}_1 \hat{}\beta^{(1)}\right)^\mathbf{H} \\
= -fg^\mathbf{H} + \hat{h}_0(\beta^{(1)})^\mathbf{H} \hat{K}_1^\mathbf{H} \\
+ \hat{K}_1 \hat{\beta}^{(1)}\hat{h}_0^\mathbf{H} + \hat{K}_1 \hat{\beta}^{(1)}(\hat{\beta}^{(1)})^\mathbf{H} \hat{K}_1^\mathbf{H}
\]

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

\[
\hat{K}_1 \beta^{(1)}\hat{h}_0^\mathbf{H} + \hat{K}_1 \text{diag}(\beta^{(1)})(\hat{\Lambda}^{(1)})\hat{K}_1^\mathbf{H} + \hat{h}_0(\beta^{(1)})^\mathbf{T} \hat{K}_1^\mathbf{H} + \hat{K}_1(\hat{\Lambda}^{(1)})^\mathbf{T} \text{diag}(\beta^{(1)})\hat{K}_1^\mathbf{H}
\]

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

\[
0 = \hat{h}_0 \left(\hat{\beta}^{(1)} - \hat{\beta}^{(1)}\right)^\mathbf{H} \hat{K}_1^\mathbf{H} + \hat{K}_1 \left(\hat{\beta}^{(1)} - \beta^{(1)}\right)\hat{h}_0^\mathbf{H} \\
+ \hat{K}_1 \left(\hat{\beta}^{(1)}(\hat{\beta}^{(1)})^\mathbf{H} - \text{diag}(\beta^{(1)})(\hat{\Lambda}^{(1)})^\mathbf{T} \text{diag}(\beta^{(1)})\right)\hat{K}_1^\mathbf{H},
\]

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

\[
\text{diag}(\beta^{(1)})(\hat{\Lambda}^{(1)})^\mathbf{T} + (\hat{\Lambda}^{(1)})^\mathbf{T} \text{diag}(\beta^{(1)}) - \beta^{(1)}(\beta^{(1)})^\mathbf{T} = 0.
\]

Via induction this leads to the conditions

\[
0 = \text{diag}(\beta^{(1)})(\hat{\Lambda}^{(1)})^\mathbf{T} + (\hat{\Lambda}^{(1)})^\mathbf{T} \text{diag}(\beta^{(1)}) - \beta^{(1)}(\beta^{(1)})^\mathbf{T},
\]

\[
\hat{\beta}^{(j)} = \beta^{(j)}, \qquad \hat{\beta}^{(j)} = \beta^{(j)}.
\]

Please note that $\hat{\Lambda}^{(j)}$, $\hat{\beta}^{(j)}$ and $\hat{\Lambda}^{(j)}$, $\hat{\beta}^{(j)}$ from the Butcher tableaus for approximating $h(t)$ 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

\[
\hat{\Lambda} = \begin{bmatrix} \hat{\mu}_1 & 0 & \cdots & 0 \\ \beta_1 & \hat{\mu}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \beta_1 & \beta_2 & \cdots & \hat{\mu}_s \end{bmatrix}, \quad \bar{\Lambda} = \begin{bmatrix} \tilde{\mu}_1 & 0 & \cdots & 0 \\ \tilde{\beta}_1 & \tilde{\mu}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{\beta}_1 & \tilde{\beta}_2 & \cdots & \tilde{\mu}_s \end{bmatrix}, \quad \beta = \begin{bmatrix} \tilde{\mu}_1 + \tilde{\mu}_1 \\ \tilde{\beta}_1 + \tilde{\beta}_2 \\ \vdots \\ \tilde{\beta}_1 + \tilde{\beta}_2 \end{bmatrix}. \tag{4.9}
\]

### 4.3. Multiplicative update formulae for \( \hat{h}_j \) and \( \hat{h}_j \)

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 \( B = \tilde{V} \tilde{D} \tilde{V}^{-1} \) with \( \hat{D} = \text{diag}(\hat{\lambda}_1, \ldots, \hat{\lambda}_n) \) and \( \tilde{D} = \text{diag}(\tilde{\lambda}_1, \ldots, \tilde{\lambda}_m) \). Then, as in Section 3.5 we find that the iterations (4.3) and (4.4) can be written in multiplicative form as in (3.26)

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

\[
\tilde{h}_j = \tilde{V}^{-T} \begin{bmatrix} \tilde{R}_j(-\tilde{\lambda}_1) & \tilde{R}_j(-\tilde{\lambda}_2) & \cdots & \tilde{R}_j(-\tilde{\lambda}_m) \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{R}_j(\tilde{\lambda}_1) & \tilde{R}_j(\tilde{\lambda}_2) & \cdots & \tilde{R}_j(\tilde{\lambda}_m) \end{bmatrix} \tilde{V}^T \tilde{h}_{j-1}. \tag{4.11}
\]

with the stability functions

\[
\hat{R}_j(z) = 1 + z(\beta^{(j)})^T(I - z\hat{\Lambda}^{(j)})^{-1}1_s, \\
\tilde{R}_j(z) = 1 + z(\beta^{(j)})^T(I - z\tilde{\Lambda}^{(j)})^{-1}1_s.
\]

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

\[
0 = \text{diag}(\beta^{(j)})\hat{\Lambda}^{(j)} \text{diag}(\beta^{(j)})^{-1} + (\hat{\Lambda}^{(j)})^T - \beta^{(j)}1_s^T, \\
0 = \tilde{\Lambda}^{(j)} + \text{diag}(\beta^{(j)})^{-1}(\tilde{\Lambda}^{(j)})^T \text{diag}(\beta^{(j)}) - 1_s(\beta^{(j)})^T.
\]

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

\[
\hat{\Lambda}^{(j)} - 1_s(\beta^{(j)})^T \sim -\hat{\Lambda}^{(j)}, \\
\tilde{\Lambda}^{(j)} - 1_s(\beta^{(j)})^T \sim -\tilde{\Lambda}^{(j)}.
\tag{4.12}
\]

Let \( \hat{\Lambda}^{(j)} \) have eigenvalues \( \hat{\mu}_1, \ldots, \hat{\mu}_s \) and let \( \tilde{\Lambda}^{(j)} \) have eigenvalues \( \tilde{\mu}_1, \ldots, \tilde{\mu}_s \). Now
with (2.6) and (4.12) we have (similar to (3.29))

\[
\begin{align*}
\hat{R}_j(z) &= \frac{\det(I - z\tilde{\Lambda}(j) - \mathbb{I}_s(\beta(j)^T))}{\det(I - z\tilde{\Lambda}(j))} = \det(I + z\tilde{\Lambda}(j)) = \prod_{i=1}^{s} \frac{(1 + z\mu_i)}{(1 - z\mu_i)}, \\
\tilde{R}_j(z) &= \frac{\det(I - z\tilde{\Lambda}(j) - \mathbb{I}_s(\tilde{\beta}(j)^T))}{\det(I - z\tilde{\Lambda}(j))} = \det(I + z\tilde{\Lambda}(j)) = \prod_{i=1}^{s} \frac{(1 + z\mu_i)}{(1 - z\mu_i)}.
\end{align*}
\]

(4.13)

Thus, if (4.17) is enforced, then the stability functions of the Runge-Kutta method for \(\hat{h}(t)\) and for \(\tilde{h}(t)\) are not independent of each other. Both depend on the eigenvalues of \(\tilde{\Lambda}(j)\) and \(\hat{\Lambda}(j)\). Thus, also the iterates \(\hat{h}_j\) (4.10) and \(\tilde{h}_j\) (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. \(\tilde{\Lambda}(j) = \tilde{\mu}_j \in \mathbb{C}\) and \(\hat{\Lambda}(j) = \hat{\mu}_j \in \mathbb{C}\) with \(\beta(j) = \beta(j) = \hat{\mu}_j + \tilde{\mu}_j \in \mathbb{C}\) and \(\beta(j) = \beta(j) = \hat{\mu}_j + \tilde{\mu}_j \in \mathbb{C}\).

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 [20, Alg. 3.4] with \(\alpha_j = -\tilde{\mu}_j^{-1}\) 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 \(B = -A^T\) and \(g = -f = b\) the Sylvester equation (4.1) reduces to the Lyapunov equation (1.1). If additionally for the parameters \(\tilde{\mu}_j = \hat{\mu}_j, j = 1, \ldots, N\), holds, then Algorithm 4 reduces to Algorithm 3 and we find \(\tilde{\mu}_j + \hat{\mu}_j = 2 \text{Re} \tilde{\mu}_j \in \mathbb{R}\).

### Algorithm 4 Low rank solution to (1.1) 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 \(\{\mu_1, \ldots, \mu_N\} \subset \mathbb{C}\) and \(\{\hat{\mu}_1, \ldots, \hat{\mu}_N\} \subset \mathbb{C}\)

**Output:** matrices \(\hat{Z} \in \mathbb{C}^{n \times n}, \tilde{Z} \in \mathbb{C}^{m \times n}\) and \(\Gamma \in \mathbb{C}^{N \times N}\) with \(\hat{Z}^T \hat{Z}^m \approx \mathcal{Y}\)

1. initialize \(\hat{h}_0 = f, \hat{h}_0 = -g, \hat{Z}_0 = \hat{Z}_0 = \Gamma_0 = [\ ]\)
2. for \(j = 1, \ldots, N\) do
3. solve \((I - \hat{\mu}_j A)\hat{K}_j = \hat{h}_{j-1}\) for \(\hat{K}_j\)
4. solve \((I + \hat{\mu}_j B^T)\tilde{K}_j = -B^T\hat{h}_{j-1}\) for \(\tilde{K}_j\)
5. \(\hat{R}_j = \hat{h}_{j-1} + \hat{\mu}_j\hat{K}_j\)
6. \(\tilde{R}_j = \tilde{h}_{j-1} + \tilde{\mu}_j\tilde{K}_j\)
7. update \(\hat{Z}_j = [\hat{Z}_{j-1}, \hat{R}_j]\)
8. update \(\tilde{Z}_j = [\tilde{Z}_{j-1}, \tilde{R}_j]\)
9. update \(\Gamma_j = \text{diag}(\Gamma_{j-1}, (\hat{\mu}_j + \tilde{\mu}_j)I_r)\)
10. \(\tilde{h}_j = \tilde{h}_{j-1} + (\mu_j + \tilde{\mu}_j)\tilde{K}_j\)
11. \(\hat{h}_j = \hat{h}_{j-1} + (\hat{\mu}_j + \tilde{\mu}_j)\hat{K}_j\)
12. end for
13. \(\hat{Z} = \hat{Z}_N, \tilde{Z} = \tilde{Z}_N, \Gamma = \Gamma_N\)

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. A realification approach based on similarity transformations was presented.

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.

References

[1] Antoulas, A.C., 2005. Approximation of Large-scale Dynamical Systems. Society for Industrial and Applied Mathematics.
[2] Benner, P., Breiten, T., 2013. Low rank methods for a class of generalized Lyapunov equations and related issues. Numerische Mathematik 124, 441–470. doi:10.1007/s00211-013-0521-0
[3] Benner, P., Breiten, T., Feng, L., 2015. Matrix Functions and Matrix Equations. World Scientific. chapter Matrix Equations and Model Reduction. pp. 50–75.
[4] Benner, P., Kürschner, P., Saak, J., 2012. Efficient handling of complex shift parameters in the low-rank Cholesky factor ADI method. Numerical Algorithms 62, 225–251. doi:10.1007/s11075-012-9669-7
[5] Benner, P., Kürschner, P., Saak, J., 2013. An improved numerical method for balanced truncation for symmetric second-order systems. Mathematical and Computer Modelling of Dynamical Systems 19, 593–615. doi:10.1080/13873954.2013.794353
[6] Benner, P., Li, R.C., Truhar, N., 2009. On the ADI method for Sylvester equations. Journal of Computational and Applied Mathematics 233, 1035 – 1045. doi:https://doi.org/10.1016/j.cam.2009.08.108
[7] Butcher, J., 2016. Numerical Methods for Ordinary Differential Equations. Wiley.
[8] Datta, B., 1994. Linear and numerical linear algebra in control theory: Some research problems. Linear Algebra and its Applications 197/198, 755–790.
[9] Day, D., Heroux, M.A., 2001. Solving complex-valued linear systems via equivalent real formulations. SIAM Journal on Scientific Computing 23, 480–498. doi:10.1137/s1064827500372262
[10] Druskin, V., Knizhnerman, L., Simoncini, V., 2011. Analysis of the rational Krylov subspace and ADI methods for solving the Lyapunov equation. SIAM Journal on Numerical Analysis 49, 1875–1898. doi:10.1137/100813287
[11] Gajić, Z., Qureshi, M., 1995. Lyapunov Matrix Equation in System Stability and Control. Math. in Science and Engineering, Academic Press, San Diego, CA.
[12] Golub, G.H., van Loan, C.F., 2013. Matrix Computations. 4th ed., Johns Hopkins University Press.
[13] Grasedyck, L., 2004. Existence of a low rank or \(H\)-matrix approximant to the solution of a Sylvester equation. Numerical Linear Algebra with Applications 11, 371–389. doi:10.1002/nla.366
[14] Hairer, E., Lubich, C., Wanner, G., 2006. Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations. 2nd ed., Springer.
[15] Hairer, E., Nørsett, S., Wanner, G., 1993. Solving Ordinary Differential Equations I. 2nd ed., Springer.
[16] Hairer, E., Wanner, G., 1996. Solving Ordinary Differential Equations II. 2nd ed., Springer.
[17] Horn, R., Johnson, C., 1991. Topics in Matrix Analysis. Cambridge University Press, Cambridge.
[18] Iserles, A., 2008. A First Course in the Numerical Analysis of Differential Equations. Cambridge Texts in Applied Mathematics. 2nd ed., Cambridge University Press.

[19] Kennedy, C.A., Carpenter, M.H., 2016. Diagonally Implicit Runge-Kutta Methods for Ordinary Differential Equations. A Review. Technical Memorandum NASA-TM-2016-21973. National Aeronautics and Space Administration Langley Research Center. Hampton, Virginia 23681-2199.

[20] Kürschner, P., 2016. Efficient Low-Rank Solution of Large-Scale Matrix Equations. Ph.D. thesis. OvGU Magdeburg.

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

[22] Li, J., White, J., 2002. Low rank solution of Lyapunov equations. SIAM Journal on Matrix Analysis and Applications 24, 260–280. doi:10.1137/S0895479801384937

[23] Lu, A., Wachspress, E., 1991. Solution of Lyapunov equations by alternating direction implicit iteration. Computers & Mathematics with Applications 21, 43–58. doi:10.1016/0898-1221(91)90124-7

[24] Moore, B., 1981. Principal component analysis in linear systems: Controllability, observability, and model reduction. IEEE Trans. Automat. Control AC-26, 17–32.

[25] Opmeer, M., 2012. Model order reduction by balanced proper orthogonal decomposition and by rational interpolation. IEEE Transactions on Automatic Control AC-57, 472–477.

[26] Peaceman, D., Rachford, Jr., H., 1955. The numerical solution of parabolic and elliptic differential equations. Journal of the Society for Industrial and Applied Mathematics 3, 17–32. doi:10.1137/0103003

[27] Penzl, T., 1999. A cyclic low-rank smith method for large sparse Lyapunov equations. SIAM Journal on Scientific Computing 21, 1401–1418. doi:10.1137/S1064827598347666

[28] Rowley, C.W., 2005. Model reduction for fluids, using balanced proper orthogonal decomposition. J. Bifurcation and Chaos 15, 997–1013. doi:10.1142/S0218127405012429

[29] Saad, Y., Gv, X.V., 1990. Numerical solution of large Lyapunov equations, in: Signal Processing, Scattering and Operator Theory, and Numerical Methods, Proc. MTNS-89, Birkhauser. pp. 503–511.

[30] Simoncini, V., 2016. Computational methods for linear matrix equations. SIAM Review 58, 377–441. doi:10.1137/130912839

[31] Singler, J.R., 2011. Convergent snapshot algorithms for infinite-dimensional Lyapunov equations. IMA Journal of Numerical Analysis 31, 1468–1496. doi:10.1093/imanum/drr029

[32] Wolf, T., Panzer, H., 2016. The ADI iteration for Lyapunov equations implicitly performs H2 pseudo-optimal model order reduction. International Journal of Control 89, 481–493. doi:10.1080/00207179.2015.1081985