Chair of Automatic Control

Technical University of Munich

Paul Rötzer

# Implementation of Model Reduction of Bilinear Systems and Extension of the Multipoint Volterra Series Interpolation

Bachelor's Thesis

15 November 2018

Supervisors:

Maria Cruz Varona, M.Sc.
Prof. Dr.-Ing. habil. Boris Lohmann

## Erklärung

Hiermit erkläre ich, die vorliegende Arbeit selbstständig durchgeführt zu haben und keine weiteren Hilfsmittel und Quellen als die angegebenen genutzt zu haben. Mit ihrer unbefristeten Aufbewahrung und Bereitstellung in der Lehrstuhlbibliothek erkläre ich mich einverstanden.

Garching bei München, den 15. November 2018 ______________________ (Paul Rötzer)

☐ Ich stelle die Software, die ich im Rahmen dieser Arbeit entwickelt habe, dem Lehrstuhl für Regelungstechnik unter den Bedingungen der 3-Klausel-BSD-Lizenz zur Verfügung. Ich behalte dabei die sämtlichen Urheber- sowie Nutzungsrechte und werde als Autor namentlich genannt.

Garching bei München, den 15. November 2018 ______________________ (Paul Rötzer)

Chair of Automatic Control (Prof. Dr.-Ing. habil. Boris Lohmann)
Technical University of Munich
Boltzmannstraße 15
85748 Garching bei München
Germany

Lehrstuhl für Regelungstechnik (Prof. Dr.-Ing. habil. Boris Lohmann)
Technische Universität München
Boltzmannstraße 15
85748 Garching bei München
Deutschland

# Task Description

## Problemstellung

Um die numerische Simulation großer Systeme effizienter zu machen, versucht man mittels Verfahren der Modellreduktion ein reduziertes Modell mit geringerer Ordnung $r \ll n$ zu bekommen, das das Übertragungsverhalten des Originalmodells möglichst gut approximiert. Für lineare Systeme ist die Modellreduktion gut untersucht und etabliert. Leider sind die meisten praxisbezogenen dynamischen Systemen jedoch *nichtlinear* oder *polynomial-nichtlinear*.

Bilineare Systeme stellen eine polynomial-nichtlineare Systemklasse dar, die eng mit den linearen Systemen zusammenhängt. Diese enge Verbindung erlaubt – unter Verwendung der sog. Volterra-Theorie – die Übertragung vieler systemtheoretischer Konzepte (z.B. Übertragungsfunktion, Gramsche, $\mathcal{H}_2$-Norm) sowie bestehender linearer Modellreduktionsverfahren (z.B. balanciertes Abschneiden, Krylow-Unterraummethoden, $\mathcal{H}_2$-optimale Reduktion) auf den bilinearen Fall.
Im Rahmen dieser Arbeit sollen, nach einer Einarbeitung in die Thematik, verschiedene existierenden Verfahren der bilinearen Systemtheorie und Modellreduktion in MATLAB® entwickelt und untersucht werden. Ziel ist also, möglichst vollständige und automatisierte Reduktionsalgorithmen für bilineare MIMO Systeme in MATLAB® zu implementieren und diese anschließend zu validieren, um somit eine MATLAB®-Toolbox für bilineare Systeme zu generieren.

## Arbeitsprogramm

1. Einarbeitung in die bilineare Modellreduktion, insbesondere in die Krylow-Unterraummethoden für lineare und bilineare Systeme [2, 6, 10, 11, 20]

2. Systematische Implementierung von Algorithmen zur Reduktion bilinearer MIMO Systeme. Folgende Schritte wären sinnvoll:

    - Weiterentwicklung der Moment Matching Algorithmen (*Volterra Series Intepolation*): `volterraBrk`, `volterraBarnoldi`, `b_irka`.
    - Integration von `bilyapchol` innerhalb von `bnorm` zur Berechnung von $\mathcal{H}_2$-Normen. Verbesserung und Entwicklung weiterer Funktionen.

3. Durchführung von Tests anhand von bilinearen Benchmarks (Fokker Plank equation, bilinear heat transfer model) und Evaluation der Reduktionsergebnisse

4. Dokumentation der Arbeit und der Ergebnisse

# Abstract

Bilinear systems are a special class of nonlinear systems. With the bilinear part in the state-equation, bilinear systems are more suitable than linear systems for the approximation of nonlinear systems. However, a series approach connects bilinear systems strongly to linear systems. This enables the possibility to adapt the system theory and model reduction methods for linear systems to bilinear systems.

In this sense, we start by investigating the *Volterra* series representation of bilinear systems and develop a criterion for convergence. Through this kind of representation we are able to define transfer functions for bilinear systems.
Subsequent, we discuss system-theoretic concepts of bilinear systems. Thereby, we especially examine bounded-input bounded-output (BIBO) stability, controllability, observability and bilinear system norms. This establishes the needed requirements for model reduction.
Regarding model reduction, we first discuss the basic concepts as well as subsystem interpolation. Following, we gain in-depth knowledge about *Volterra* series interpolation. In this context, we discuss different aspects of implementation of this framework. After that, we extend the multipoint *Volterra* series interpolation to match higher order moments and upgrade the previously introduced implementations for this. As a last point of model reduction we discuss an algorithm, which computes an $\mathcal{H}_2$-optimal model. Concluding, we test the theoretic concepts using bilinear benchmark models.

## Kurzfassung

Bilineare Systeme sind eine spezielle Systemklasse der nichtlinearen Systeme. Durch den bilinearen Anteil in der Zustandsgleichung eignen sich bilineare Systeme besser als lineare Systeme zur Approximation von nichtlinearen Systemen. Der Vorteil von bilinearen Systemen ist, dass sie durch einen Reihenansatz stark mit linearen verknüpft sind, was es ermöglicht Systemtheorie bzw. Modellreduktion für lineare Systeme auf bilineare Systeme zu übertragen.

In diesem Sinne wird die Reihendarstellung untersucht und ein Konvergenzkriterium erarbeitet. Mit dieser Reihendarstellung ist es möglich Übertragungsfunktionen für bilineare Systeme zu definieren.
Daraufhin werden die systemtheoretischen Konzepte zu bilinearen Systemen besprochen, wobei besonders auf BIBO Stabilität, Steuerbarkeit, Beobachtbarkeit und Systemnormen eingegangen wird. Damit schaffen wir das nötige Vorwissen zur Modellreduktion.
Im Bezug auf Modellreduktion werden zuerst Grundkonzepte sowie die Subsysteminterpolation besprochen. Anschließend wird vertieft auf die *Volterra* Reihen Interpolation eingegangen. Dabei wird ein tiefgreifendes Verständnis geschaffen und verschiedene Implementierungsaspekte werden diskutiert. Im Anschluss daran, wird die *Volterra* Reihen Interpolation für das Abgleichen höherer Momente erweitert und die zuvor präsentierten Implementierungen dafür angepasst. Als letzter Punkt der Modellreduktion wird ein Algorithmus besprochen, der ein $\mathcal{H}_2$-optimales, reduziertes Modell findet. Abschließend werden die theoretischen Konzepte an bilinearen Benchmarkmodellen getestet.

# Contents

# Chapter 1

# Introduction

## 1.1 Motivation

Today's demand for better and less expensive technology requires the use of high fidelity dynamical models, in other words, mathematical models which represent a real world process very well. The mentioned dynamic models are partial differential equations (PDE), ordinary differential equations (ODE) or differential algebraic equations (DAE), which allow us to avoid time-consuming and expensive experiments since we are able to simulate them. To be able to capture all dynamics of the system and in this sense increase the accuracy of the model, a high model complexity is often induced. Within this context high complexity means that the number of equations or rather the state-dimension $n$ of the underlying mathematical model increases significantly. Indeed, one often deals with so-called large-scale systems of size $n \sim 10^6$. Obviously, it is not possible to simulate these large-scale systems with conventional computers. Considering the following linear system of differential equations

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t)$$

where $\boldsymbol{A} \in \mathbb{R}^{10^6 \times 10^6}$. Storing the matrix $\boldsymbol{A}$ as a double-precision array on a 64-bit operating system requires a total amount of 7450 GB of storage. Assuming that we could store $\boldsymbol{A}$, it would take over 5 days on a moderate machine to simulate 10 seconds by use of *Euler's* method and a step size of 0.01.
This example shows, that there is a drastic need to reduce the complexity of those dynamic models to reduce the computational effort or rather at least make it possible to simulate them. This motivates model reduction, whose goal it is to approximate a dynamical model with a much smaller amount of differential equations. Apparently, the main objective is to reduce the error of the approximation.
Another aspect of the complexity of a dynamic model is that most technical processes do not follow a linear behavior which means that the differential equations are nonlinear. Since a linearized nonlinear system is mostly bad in capturing nonlinear dynamics we consider a special class of nonlinear systems: bilinear systems. They combine the two major advantages of better capturing nonlinear dynamics as well as the possibility to apply linear system theory and linear model reduction techniques.

## 1.2 Goals

The first goal of this thesis is to obtain in-depth understanding of bilinear systems. Therefore, we consider different representations of bilinear systems as well as bilinear system theory.
After that we draw our attention to model reduction of bilinear system. In this sense we point out different algorithms for *Volterra* series interpolation. We also want to extend the existing multipoint *Volterra* series interpolation framework to multiple-input multiple-output (MIMO) systems as well as for higher order moments which we consequently call multimoment *Volterra* series interpolation.
Finally, we want to discuss the bilinear rational *Krylov* algorithm (BIRKA).

## 1.3 Outline

We begin this thesis by explaining bilinear systems generally in Chapter 2. Consequently, we discuss their origin and different forms to represent them. On this occasion we introduce the *Volterra* series representation and point out convergence criteria. As a last part of this chapter we deal with transfer functions.
Chapter 3 is completely related to bilinear system theory. Hence, we start with the pole-residue and the diagonal form of a bilinear system. Following, we discuss BIBO stability in depth. In addition we obtain criteria for controllability and observability and in this sense introduce the *Gramians*. Finally, we show the connection between the bilinear system norms and the *Gramians*.
In Chapter 4 we discuss interpolation-based reduction methods for bilinear systems. Therefore, we start by explaining basic concepts of model reduction. After that, we briefly explain the subsystem interpolation framework. Following, we introduce multipoint *Volterra* series interpolation, explain its connection to *Sylvester* equations and give different algorithms which yield the projection matrices. In addition to that, we expand the framework to be able to match higher order moments which we consequently call multimoment *Volterra* series interpolation. In this sense, we also explain different ways to compute the projection matrices. Then we discuss other special cases of *Volterra* series interpolation like block *Krylov*, *Markov Parameters* and complex expansion points. Finally, we introduce BIRKA, an algorithm to obtain an $\mathcal{H}_2$-optimal reduced bilinear system.
Rounding off this thesis, we provide numerical examples in Chapter 5 corresponding to the previously introduced algorithms followed by the conclusion in Chapter 6.

Note that for better readability we highlight important equations blue. Since one often looses sight of the actual goal during a derivation, we highlight the goal red. If the derivation yields the highlighted goal, we mark the formula green.

# Chapter 2

# Bilinear Systems

Bilinear systems are a special class of nonlinear systems. In state-space representation they only differ from linear systems by having an additional bilinear term. A generalized single-input single-output (SISO) bilinear system $\zeta$ is given by

$$\zeta : \begin{cases} \boldsymbol{E}\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{N}\boldsymbol{x}(t)u(t) + \boldsymbol{b}u(t), \quad \boldsymbol{x}(0) = \boldsymbol{x}_0, \\ y(t) = \boldsymbol{c}^\mathsf{T}\boldsymbol{x}(t), \end{cases} \tag{2.1}$$

where $\boldsymbol{E}, \boldsymbol{A} \in \mathbb{R}^{n\times n}$ with $\boldsymbol{E}$ being regular, $\boldsymbol{N} \in \mathbb{R}^{n\times n}$, $\boldsymbol{b} \in \mathbb{R}^{n\times 1}$ and $\boldsymbol{c}^\mathsf{T} \in \mathbb{R}^{1\times n}$. Looking at a MIMO bilinear system things often get complicated due to a bilinear term for every input. In literature, the bilinear term is often written in *Kronecker* notation $\bar{\boldsymbol{N}}(\boldsymbol{I} \otimes \boldsymbol{x}(t))\boldsymbol{u}(t)$ where $\bar{\boldsymbol{N}} = [\boldsymbol{N}_1, \ldots, \boldsymbol{N}_m] \in \mathbb{R}^{n\times nm}$. This makes it possible to write it brief, but it is not intuitive to read. Making things easier to understand, we will use the equivalent sum notation which means that in the MIMO case $\boldsymbol{N}\boldsymbol{x}(t)u(t)$ becomes $\sum_{j=1}^{m} \boldsymbol{N}_j\boldsymbol{x}(t)u_j(t)$. Hence, a MIMO bilinear system with $m$ inputs and $p$ outputs results in

$$\boldsymbol{\zeta} : \begin{cases} \boldsymbol{E}\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \sum_{j=1}^{m} \boldsymbol{N}_j\boldsymbol{x}(t)u_j(t) + \boldsymbol{B}\boldsymbol{u}(t), \quad \boldsymbol{x}(0) = \boldsymbol{x}_0, \\ \boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t), \end{cases} \tag{2.2}$$

where $\boldsymbol{E}, \boldsymbol{A} \in \mathbb{R}^{n\times n}$ with $\boldsymbol{E}$ regular, $\boldsymbol{N}_j \in \mathbb{R}^{n\times n}$, $\boldsymbol{B} \in \mathbb{R}^{n\times m}$ and $\boldsymbol{C} \in \mathbb{R}^{p\times n}$.
As bilinear systems are introduced, we will deal with their origin in the subsequent part. Consequently, we will illustrate two ways to receive a bilinear system as well as provide examples.
After that we have to consider a different representation to be able to later apply similar theories as for linear model reduction. Hence, we introduce ways to gain the so-called *Volterra* series representation of a bilinear system.
Finally, we show two possibilities of transfer functions: transfer functions with triangular kernels and with regular ones.

## 2.1 Origin

While dealing with bilinear systems, one may ask oneself where they come from. Basically there are two possible options. First, one could receive a bilinear system by modeling a technical system which has bilinear properties. Second, it is possible to obtain a bilinear system out of a nonlinear system by use of the *Carleman* linearization.

### 2.1.1 Physical Modelling

Bilinear systems appear in a variety of disciplines. A simple biological example is the equation which describes the population of a species. Similar to that, the neutron population in nuclear fission is also described by a bilinear equation. Another physical example is a mechanical brake, where the approximation of the frictional force has bilinear characteristics. Although all this processes sound interesting due to their small order they are not relevant for this thesis as our goal is to apply model reduction which only makes sense for large-scale systems [16].
However, to provide a relevant example, we take a look at the *Itô-type* linear stochastic differential equations of the form

$$\begin{aligned}
\mathrm{d}\boldsymbol{x}(t) &= \boldsymbol{A}\boldsymbol{x}(t)\,\mathrm{d}t + \sum_{j=1}^{m} \boldsymbol{A}_0^j \boldsymbol{x}(t)\,\mathrm{d}w_j(t) + \boldsymbol{B}\boldsymbol{u}(t)\,\mathrm{d}t, \quad \boldsymbol{x}(0) = \boldsymbol{x}_0, \\
\boldsymbol{y}(t) &= \boldsymbol{C}\boldsymbol{x}(t)
\end{aligned}$$

where $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, $\boldsymbol{A}_0^j \in \mathbb{R}^{n\times n}$, $\boldsymbol{B} \in \mathbb{R}^{n\times m}$ and $\boldsymbol{C} \in \mathbb{R}^{p\times n}$ and the $\mathrm{d}w_j(t)$ are *white noise* processes associated with *Wiener* processes $w_j(t)$. One can clearly determine the bilinear structure. Later on, we will introduce an *Itô-type* equation, the *Fokker-Planck* equation, and use it as a benchmark for our model reduction implementations.

### 2.1.2 Carleman Bilinearization

Another option to derive large-scale bilinear systems is to apply the *Carleman* linearization to a nonlinear system. Replacing the approximation of a nonlinear system with a bilinear system instead of a linear system improves the accuracy significantly.

#### Kronecker Product

Initially, we have to define the *Kronecker* product. Unlike other operators, the application of the *Kronecker* product does not depend on the dimensions of the matrices. Thus, it is an operation on two matrices of arbitrary size.

**Definition 2.1** (*Kronecker* Product)**.** Let the matrix $\boldsymbol{A}$ be of size $m \times n$ and the matrix $\boldsymbol{B}$ of size $p \times q$. Applying the *Kronecker* product as follows results in a matrix with dimensions $mp \times nq$

$$\boldsymbol{A} \otimes \boldsymbol{B} = \begin{bmatrix} a_{1,1}\boldsymbol{B} & \dots & a_{1,n}\boldsymbol{B} \\ \vdots & \ddots & \vdots \\ a_{m,1}\boldsymbol{B} & \dots & a_{m,n}\boldsymbol{B} \end{bmatrix}.$$

▲

For later use in this chapter, we also have to define the *Kronecker* exponential which we will indicate with round brackets.

**Definition 2.2** (*Kronecker* Exponential)**.** The $k$-th *Kronecker* exponential of a matrix $\boldsymbol{A}$ results in $k$ *Kronecker* products of $\boldsymbol{A}$ with itself.

$$\boldsymbol{A}^{(k)} = \underbrace{\boldsymbol{A} \otimes \ldots \otimes \boldsymbol{A}}_{k-\text{times}}$$

▲

## Bilinearization

Considering the following state-space formulation of the MIMO nonlinear system

$$\begin{aligned}\boldsymbol{E}\dot{\boldsymbol{x}}(t) &= \boldsymbol{a}(\boldsymbol{x}(t)) + \sum_{j=1}^{m} \boldsymbol{b}_j(\boldsymbol{x}(t))u_j(t),\\ \boldsymbol{y}(t) &= \boldsymbol{C}\boldsymbol{x}(t)\end{aligned} \tag{2.3}$$

where $\boldsymbol{x}(t) \in \mathbb{R}^{n\times 1}$ denotes the state vector, $\boldsymbol{u}(t) = [u_1(t), \ldots, u_m(t)]^\mathsf{T} \in \mathbb{R}^{m\times 1}$ contains the inputs and $\boldsymbol{y}(t) \in \mathbb{R}^{p\times 1}$ contains the outputs. Our objective is to approximate this nonlinear system via the *Carleman* linearization. For better readability we will drop the time argument. We start by expanding the nonlinear function $\boldsymbol{a}(\boldsymbol{x})$ at the equilibrium point $\boldsymbol{x}_0$ as a *Taylor* series

$$\boldsymbol{a}(\boldsymbol{x}) = \boldsymbol{a}(\boldsymbol{x}_0) + \underbrace{\frac{\partial \boldsymbol{a}(\boldsymbol{x}_0)}{\partial \boldsymbol{x}}}_{\boldsymbol{A}_1}(\boldsymbol{x}-\boldsymbol{x}_0) + \underbrace{\frac{1}{2!}\frac{\partial^2 \boldsymbol{a}(\boldsymbol{x}_0)}{\partial \boldsymbol{x}^2}}_{\boldsymbol{A}_2}(\boldsymbol{x}-\boldsymbol{x}_0)^{(2)} + \underbrace{\frac{1}{3!}\frac{\partial^3 \boldsymbol{a}(\boldsymbol{x}_0)}{\partial \boldsymbol{x}^3}}_{\boldsymbol{A}_3}(\boldsymbol{x}-\boldsymbol{x}_0)^{(3)} + \ldots .$$

While setting $\boldsymbol{x}_0 = \boldsymbol{0}$ and assuming $\boldsymbol{a}(\boldsymbol{x}_0) = \boldsymbol{0}$ we can rewrite the *Taylor* series with Kronecker notation for $\boldsymbol{x}$.

$$\begin{aligned}\boldsymbol{a}(\boldsymbol{x}) &= \boldsymbol{A}_1\boldsymbol{x} + \boldsymbol{A}_2\overbrace{(\boldsymbol{x}\otimes\boldsymbol{x})}^{\boldsymbol{x}^{(2)}} + \boldsymbol{A}_3\overbrace{(\boldsymbol{x}\otimes\boldsymbol{x}\otimes\boldsymbol{x})}^{\boldsymbol{x}^{(3)}} + \ldots\\ &= \sum_{k=1}^{\infty} \boldsymbol{A}_k\boldsymbol{x}^{(k)} \approx \sum_{k=1}^{N} \boldsymbol{A}_k\boldsymbol{x}^{(k)}.\end{aligned} \tag{2.4}$$

Here $\boldsymbol{A}_1 \in \mathbb{R}^{n\times n}$ is the first partial derivative also called Jacobian of $\boldsymbol{a}(\boldsymbol{x})$. Generally, $\boldsymbol{A}_i \in \mathbb{R}^{n\times n^i}$ identifies the $i$-th partial derivative of $\boldsymbol{a}(\boldsymbol{x})$ with respect to $\boldsymbol{x}$ evaluated at $\boldsymbol{x}_0$. We expand each $\boldsymbol{b}_j(\boldsymbol{x})$ in the same way but now assuming that $\boldsymbol{b}_j(\boldsymbol{x}_0) \neq \boldsymbol{0}$

$$\begin{aligned}\boldsymbol{b}_j(\boldsymbol{x}) &= \boldsymbol{B}_{0,j} + \boldsymbol{B}_{1,j}\boldsymbol{x} + \boldsymbol{B}_{2,j}(\boldsymbol{x}\otimes\boldsymbol{x}) + \boldsymbol{B}_{3,j}(\boldsymbol{x}\otimes\boldsymbol{x}\otimes\boldsymbol{x}) + \ldots\\ &= \sum_{k=0}^{\infty} \boldsymbol{B}_{k,j}\boldsymbol{x}^{(k)} \approx \sum_{k=0}^{N-1} \boldsymbol{B}_{k,j}\boldsymbol{x}^{(k)}.\end{aligned} \tag{2.5}$$

Note that $\boldsymbol{B}_{0,j} \in \mathbb{R}^{n\times 1}$ represents $\boldsymbol{b}(\boldsymbol{x})$ evaluated at $\boldsymbol{x}_0$. As above, $\boldsymbol{B}_i \in \mathbb{R}^{n\times n^i}$ identifies the $i$-th partial derivative of $\boldsymbol{b}(\boldsymbol{x})$ with respect to $\boldsymbol{x}$ evaluated at $\boldsymbol{x}_0$. Taking (2.4) and (2.5) into account we can rewrite (2.3) as follows

$$\begin{aligned}\boldsymbol{E}\dot{\boldsymbol{x}} &= \sum_{k=1}^{N} \boldsymbol{A}_k\boldsymbol{x}^{(k)} + \sum_{j=1}^{m}\sum_{k=0}^{N-1} \boldsymbol{B}_{k,j}\boldsymbol{x}^{(k)}u_j,\\ \boldsymbol{y} &= \boldsymbol{C}\boldsymbol{x}.\end{aligned} \tag{2.6}$$

Basically one could stop right here and use (2.6) to approximate the nonlinear system (2.3). As our goal is to obtain a bilinear structure as follows

$$\boldsymbol{E}\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \sum_{j=1}^{m} \boldsymbol{N}_j \boldsymbol{x} u_j + \boldsymbol{B}\boldsymbol{u} \tag{2.7}$$

we have to continue with the bilinearization step. It is possible to write (2.6) with the structure of (2.7) by defining a new state-vector

$$\tilde{\boldsymbol{x}} := \begin{bmatrix} \boldsymbol{x}^{(1)} \\ \vdots \\ \boldsymbol{x}^{(N)} \end{bmatrix}.$$

To specify the first $N$ state-equations corresponding to (2.6), we develop a differential equation for $\boldsymbol{x}^{(2)}$ by neglecting the terms of degree higher than $N$.

$$\begin{aligned} \boldsymbol{E}^{(2)}\dot{\boldsymbol{x}}^{(2)} &= \frac{\mathrm{d}}{\mathrm{d}t}(\boldsymbol{E}\boldsymbol{x} \otimes \boldsymbol{E}\boldsymbol{x}) = \boldsymbol{E}\dot{\boldsymbol{x}} \otimes \boldsymbol{E}\boldsymbol{x} + \boldsymbol{E}\boldsymbol{x} \otimes \boldsymbol{E}\dot{\boldsymbol{x}} \\ &= \underbrace{\left( \sum_{k=0}^{N} \boldsymbol{A}_{\mathrm{k}} \boldsymbol{x}^{(k)} + \sum_{j=1}^{m} \sum_{k=0}^{N-1} \boldsymbol{B}_{k,j} \boldsymbol{x}^{(k)} u_j \right)}_{\boldsymbol{E}\dot{\boldsymbol{x}}} \otimes \boldsymbol{E}\boldsymbol{x} \\ &\quad + \boldsymbol{E}\boldsymbol{x} \otimes \underbrace{\left( \sum_{k=0}^{N} \boldsymbol{A}_{k} \boldsymbol{x}^{(k)} + \sum_{j=1}^{m} \sum_{k=0}^{N-1} \boldsymbol{B}_{k,j} \boldsymbol{x}^{(k)} u_j \right)}_{\boldsymbol{E}\dot{\boldsymbol{x}}} \\ &= \sum_{k=1}^{N-1} \left( \boldsymbol{A}_k \otimes \boldsymbol{E} + \boldsymbol{E} \otimes \boldsymbol{A}_k \right) \boldsymbol{x}^{(k+1)} + \\ &\quad \sum_{j=1}^{m} \sum_{k=0}^{N-2} \left( \boldsymbol{B}_{k,j} \otimes \boldsymbol{E} + \boldsymbol{E} \otimes \boldsymbol{B}_{k,j} \right) \boldsymbol{x}^{(k+1)} u_j, \quad \boldsymbol{x}^{(2)}(\boldsymbol{0}) = \boldsymbol{0}. \end{aligned} \tag{2.8}$$

As we can see in the last part of (2.8), $\boldsymbol{x}^{(2)}$ satisfies a differential equation with the same structure as $\boldsymbol{x}^{(1)} = \boldsymbol{x}$ in (2.6). Therefore, we can generalize the above equation as follows

$$\boldsymbol{E}^{(i)}\dot{\boldsymbol{x}}^{(i)} = \sum_{k=1}^{N-i+1} \boldsymbol{A}_{i,k} \boldsymbol{x}^{(k+i-1)} + \sum_{j=1}^{m} \sum_{k=0}^{N-i} \boldsymbol{B}_{i,k,j} \boldsymbol{x}^{(k+i-1)} u_j, \quad \boldsymbol{x}^{(i)}(\boldsymbol{0}) = \boldsymbol{0} \tag{2.9}$$

where $\boldsymbol{A}_{1,k} = \boldsymbol{A}_k$ and for $i > 1$

$$\boldsymbol{A}_{i,k} = \boldsymbol{A}_k \otimes \boldsymbol{E} \otimes \ldots \otimes \boldsymbol{E} + \boldsymbol{E} \otimes \boldsymbol{A}_k \otimes \boldsymbol{E} \otimes \ldots \otimes \boldsymbol{E} + \ldots + \boldsymbol{E} \otimes \ldots \otimes \boldsymbol{E} \otimes \boldsymbol{A}_k.$$

For the matrices $\boldsymbol{B}_{i,k,j}$ we use similar notation. Finally, we gain a bilinear system like (2.7) making use of (2.9)

$$\begin{aligned} \tilde{\boldsymbol{E}}\dot{\tilde{\boldsymbol{x}}} &= \tilde{\boldsymbol{A}}\tilde{\boldsymbol{x}} + \sum_{j=1}^{m} \tilde{\boldsymbol{N}}_j \tilde{\boldsymbol{x}} u_j + \tilde{\boldsymbol{B}}\boldsymbol{u}, \quad \tilde{\boldsymbol{x}}(0) = \tilde{\boldsymbol{x}}_0, \\ \boldsymbol{y} &= \tilde{\boldsymbol{C}}\tilde{\boldsymbol{x}} \end{aligned} \tag{2.10}$$

where

$$\tilde{\boldsymbol{E}} = \begin{bmatrix} \boldsymbol{E} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{E}^{(2)} & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \boldsymbol{E}^{(N)} \end{bmatrix}, \qquad \tilde{\boldsymbol{A}} = \begin{bmatrix} \boldsymbol{A}_{1,1} & \boldsymbol{A}_{1,2} & \dots & \boldsymbol{A}_{1,N} \\ \mathbf{0} & \boldsymbol{A}_{2,1} & \dots & \boldsymbol{A}_{2,N-1} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \boldsymbol{A}_{N,1} \end{bmatrix},$$

$$\tilde{\boldsymbol{N}}_j = \begin{bmatrix} \boldsymbol{B}_{1,1,j} & \boldsymbol{B}_{1,2,j} & \dots & \boldsymbol{B}_{1,N,j} & \mathbf{0} \\ \boldsymbol{B}_{2,0,j} & \boldsymbol{B}_{2,1,j} & \dots & \boldsymbol{B}_{2,N-2,j} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{B}_{3,0,j} & \dots & \boldsymbol{B}_{3,N-3,j} & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \dots & \boldsymbol{B}_{N,0,j} & \mathbf{0} \end{bmatrix}, \quad \tilde{\boldsymbol{B}} = \begin{bmatrix} \boldsymbol{B}_{1,0,j} \\ \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix}, \quad \tilde{\boldsymbol{x}}_0 = \begin{bmatrix} \boldsymbol{x}_0 \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix},$$

$$\tilde{\boldsymbol{C}} = \begin{bmatrix} \boldsymbol{C} & \mathbf{0} & \dots & \mathbf{0} \end{bmatrix}.$$

According to the bilinear structure of (2.10) we call this procedure *Carleman* bilinearization. Let us not loose sight of the fact that our bilinear approximation in (2.10) is truncated since we only consider $N$ terms with the *Taylor* series. Note that each differential equation of $\boldsymbol{x}^{(i)}$ contains $i-1$ *Kronecker* products which heavily increases the dimensions $\tilde{n} = \sum_{k=1}^{N} n^k$ of our new state-vector $\tilde{\boldsymbol{x}}$. This is the crucial disadvantage of the bilinearization approach [5] [20].

*Example* 2.1 (Bilinearizing a simple nonlinear system). We can approximate the system

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{a}(\boldsymbol{x}(t)) + \boldsymbol{b}u(t), \\ y(t) &= \boldsymbol{c}^{\mathsf{T}}\boldsymbol{x}(t) \end{aligned} \tag{2.11}$$

with the truncated *Carleman* bilinearization for $N = 2$. Above formulations yield the matrices

$$\tilde{\boldsymbol{x}}(t) = \begin{bmatrix} \boldsymbol{x}(t) \\ \boldsymbol{x}(t) \otimes \boldsymbol{x}(t) \end{bmatrix}, \qquad \tilde{\boldsymbol{b}} = \begin{bmatrix} \boldsymbol{b} \\ \mathbf{0} \end{bmatrix}, \qquad \tilde{\boldsymbol{c}}^{\mathsf{T}} = \begin{bmatrix} \boldsymbol{c}^{\mathsf{T}} & \mathbf{0} \end{bmatrix},$$

$$\tilde{\boldsymbol{A}} = \begin{bmatrix} \boldsymbol{A}_1 & \boldsymbol{A}_2 \\ \mathbf{0} & \boldsymbol{A}_1 \otimes \mathbf{I} + \mathbf{I} \otimes \boldsymbol{A}_1 \end{bmatrix}, \quad \tilde{\boldsymbol{N}} = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \boldsymbol{b} \otimes \mathbf{I} + \mathbf{I} \otimes \boldsymbol{b} & \mathbf{0} \end{bmatrix}.$$

Hence, the bilinear approximation of (2.11) results in

$$\begin{aligned} \dot{\tilde{\boldsymbol{x}}}(t) &= \tilde{\boldsymbol{A}}\tilde{\boldsymbol{x}}(t) + \tilde{\boldsymbol{N}}\tilde{\boldsymbol{x}}(t)u(t) + \tilde{\boldsymbol{b}}u(t), \\ y(t) &= \tilde{\boldsymbol{c}}^{\mathsf{T}}\tilde{\boldsymbol{x}}(t). \end{aligned} \tag{2.12}$$

As mentioned, the order of the bilinear system increases significantly with higher order terms. In this example the nonlinear state representation has the dimension $n$ whereas the bilinear state representation has size $(n + n^2)$. △

## 2.2 Volterra Series Representation

To apply linear model reduction concepts to bilinear systems, we need a different representation. Thus, we introduce the concepts of *Volterra* series. Our goal is to describe a bilinear system $\boldsymbol{\zeta}$ with the following sequence of cascaded linear systems $\boldsymbol{\Sigma}_k$

$$\begin{aligned} \boldsymbol{\Sigma}_1 : \quad & \boldsymbol{E}\dot{\boldsymbol{x}}_1(t) = \boldsymbol{A}\boldsymbol{x}_1(t) + \boldsymbol{B}\boldsymbol{u}(t), \qquad \boldsymbol{x}_1(0) = \boldsymbol{x}_0, \\ \boldsymbol{\Sigma}_k : \quad & \boldsymbol{E}\dot{\boldsymbol{x}}_k(t) = \boldsymbol{A}\boldsymbol{x}_k(t) + \sum_{j=1}^{m} \boldsymbol{N}_j \boldsymbol{x}_{k-1}(t) u_j(t), \qquad \boldsymbol{x}_k(0) = \boldsymbol{0} \quad \text{for } k \geq 2 \end{aligned} \tag{2.13}$$

for which we are able to compute each solution $\boldsymbol{x}_k(t)$. The solution of the bilinear system will then be the sum over all solutions $\boldsymbol{x}(t) = \sum_{k=1}^{\infty} \boldsymbol{x}_k(t)$.

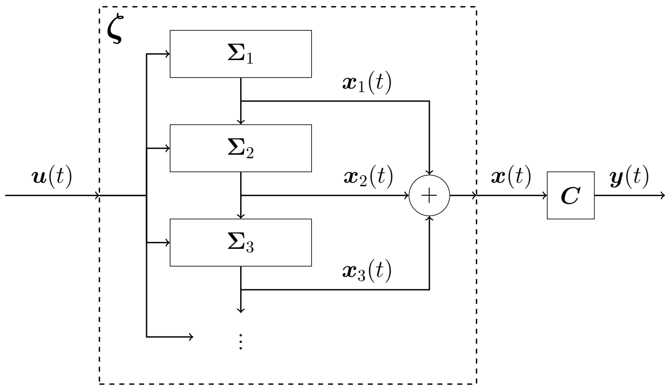


Figure 2.1: *Volterra* series representation of a bilinear system $\boldsymbol{\zeta}$

As we can see in (2.13) and Fig. 2.1, each so-called subsystem $\boldsymbol{\Sigma}_k$ includes the input and a unique state-vector $\boldsymbol{x}_k(t)$. For $k > 1$ every subsystem $\boldsymbol{\Sigma}_k$ also depends on the solution $\boldsymbol{x}_{k-1}(t)$ of the previous subsystem $\boldsymbol{\Sigma}_{k-1}$. We will discover that the importance of a subsystem decreases while $k$ increases.
To gain this *Volterra* series representation, there are two possible approaches. On the one hand, it exists the variational equation approach, on the other hand one could perform the *Picard* iteration.

### 2.2.1 Variational Equation Approach

The variational equation approach is a straightforward method to obtain the subsystem representation. Its ansatz inherits the idea of *Volterra* series. Let us assume that a MIMO bilinear system

$$\boldsymbol{\zeta} : \begin{cases} \boldsymbol{E}\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \sum_{j=1}^{m} \boldsymbol{N}_j \boldsymbol{x}(t) u_j(t) + \boldsymbol{B}\boldsymbol{u}(t), \quad \boldsymbol{x}(0) = \boldsymbol{x}_0, \\ \boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t), \end{cases}$$

is forced by an input which looks as follows

$$\boldsymbol{u}(t) = \alpha \boldsymbol{u}(t).$$

Hereby $\alpha$ denotes an arbitrary scalar. Let us also suppose that we can rewrite the state-vector as a sum of state-vectors weighted by powers of $\alpha$

$$\boldsymbol{x}(t) = \sum_{k=1}^{\infty} \alpha^k \boldsymbol{x}_k(t) = \alpha \boldsymbol{x}_1(t) + \alpha^2 \boldsymbol{x}_2(t) + \alpha^3 \boldsymbol{x}_3(t) + \dots ,$$
$$\dot{\boldsymbol{x}}(t) = \sum_{k=1}^{\infty} \alpha^k \dot{\boldsymbol{x}}_k(t) = \alpha \dot{\boldsymbol{x}}_1(t) + \alpha^2 \dot{\boldsymbol{x}}_2(t) + \alpha^3 \dot{\boldsymbol{x}}_3(t) + \dots \ .$$

Inserting the ansatz in the state-equation of $\boldsymbol{\zeta}$ results in

$$\begin{aligned} \boldsymbol{E}(\alpha \dot{\boldsymbol{x}}_1(t) + \alpha^2 \dot{\boldsymbol{x}}_2(t) + \dots) &= \boldsymbol{A}(\alpha \boldsymbol{x}_1(t) + \alpha^2 \boldsymbol{x}_2(t) + \dots) \\ &\quad + \sum_{j=1}^{m} \boldsymbol{N}_j (\alpha \boldsymbol{x}_1(t) + \alpha^2 \boldsymbol{x}_2(t) + \dots) \alpha u_j(t) + \boldsymbol{B} \alpha \boldsymbol{u}(t). \end{aligned} \tag{2.14}$$

Comparing terms with equal powers of $\alpha$ yields a state-equation for each $\boldsymbol{x}_k(t)$ that must hold for all $\alpha$. The previous yields [8]

$$\begin{aligned} \alpha \ &: & \boldsymbol{E}\dot{\boldsymbol{x}}_1(t) &= \boldsymbol{A}\boldsymbol{x}_1(t) + \boldsymbol{B}\boldsymbol{u}(t)\,, & \boldsymbol{x}_1(0) &= \boldsymbol{x}_0\,, \\ \alpha^2 &: & \boldsymbol{E}\dot{\boldsymbol{x}}_2(t) &= \boldsymbol{A}\boldsymbol{x}_2(t) + \sum_{j=1}^{m} \boldsymbol{N}_j \boldsymbol{x}_1(t) u_j(t)\,, & \boldsymbol{x}_2(0) &= \boldsymbol{0}\,, \\ \alpha^3 &: & \boldsymbol{E}\dot{\boldsymbol{x}}_3(t) &= \boldsymbol{A}\boldsymbol{x}_3(t) + \sum_{j=1}^{m} \boldsymbol{N}_j \boldsymbol{x}_2(t) u_j(t)\,, & \boldsymbol{x}_3(0) &= \boldsymbol{0}\,, \\ & & &\vdots \end{aligned}$$

which we can generalize. By assuming that $\alpha = 1$ we obtain the following subsystem representation of $\boldsymbol{\zeta}$

$$\boldsymbol{\zeta} : \begin{cases} \boldsymbol{E}\dot{\boldsymbol{x}}_1(t) = \boldsymbol{A}\boldsymbol{x}_1(t) + \boldsymbol{B}\boldsymbol{u}(t)\,, & \boldsymbol{x}_1(0) = \boldsymbol{x}_0, \\ \boldsymbol{E}\dot{\boldsymbol{x}}_k(t) = \boldsymbol{A}\boldsymbol{x}_k(t) + \sum_{j=1}^{m} \boldsymbol{N}_j \boldsymbol{x}_{k-1}(t)\, u_j(t)\,, & \boldsymbol{x}_k(0) = \boldsymbol{0}\,, \quad k \geq 2 \end{cases}$$

which is equal to (2.13). Note that only the first subsystem contains the initial condition $\boldsymbol{x}_0$. All subsystems for $k \geq 2$ will also hold $\boldsymbol{x}_0$ since they contain the first subsystem due to cascading.

### 2.2.2 Picard Iteration Approach

The variational equation approach does not answer questions concerning convergence of the *Volterra* series. Thus, we take a look at the *Picard* iteration approach. We will discover criteria for convergence which then allow us to truncate the series. The truncation is necessary to numerically compute $\boldsymbol{x}(t)$ via the sum over the subsystems. To apply the *Picard* iteration to bilinear systems we discuss the underlying theory first.

#### Picard Iteration

Generally, the *Picard* iteration is a fixed point iteration to approximate functions like

$$z(t) = f(t, z(t))$$

with iterative calculations of the sequence

$$z_k(t) = f(t, z_{k-1}(t)).$$

To ensure convergence, $f(z)$ needs to be a contraction, which means that there exists a constant $K \in [0, 1)$ so that

$$\|f(t, z_1(t)) - f(t, z_2(t))\| \leq K\|z_1(t) - z_2(t)\|.$$

Therefore, a contraction means that the images of two points are closer than the two points themselves [1].
To apply this theory to our problem, we have to use a modified *Picard* iteration. The starting value problem

$$\dot{z}(t) = f(t, z(t)), \qquad z(t_0) = z_0 \tag{2.16}$$

can also be written as

$$z(t) = z_0 + \int_{t_0}^{t} f(\tau, z(\tau))\,\mathrm{d}\tau.$$

As the *Picard-Lindelöf* theorem shows, a starting value problem like (2.16) has a unique solution within $t \in [0, T]$ if $f(t, z(t))$ satisfies a *Lipschitz* condition for the second variable. Similar to above, this means there exists a variable $L \geq 0$ so that

$$\|f(t, z_1(t)) - f(t, z_2(t))\| \leq L\|z_1(t) - z_2(t)\|, \text{with } t \in [0, T] \tag{2.17}$$

is fulfilled [25]. If $f(t, z(t))$ satisfies (2.17) we can approximate (2.16) with a small enough $\varepsilon > 0$ by iteratively computing

$$\begin{aligned} z_0(t) &= z_0 \\ z_k(t) &= z_0 + \int_{t_0}^{t} f(\tau, z_{k-1}(\tau))\,\mathrm{d}\tau, \qquad t \in [t_0, t_0 + \varepsilon]. \end{aligned}$$

For infinite iterations the solution $z_{k\to\infty}(t)$ converges against the analytic solution $z(t)$ [22].

### Picard Iteration Applied To Bilinear Systems

To not loose sight of our actual goal we briefly recapitulate that we try to find a sequence of cascaded linear systems with which we can represent our bilinear system $\boldsymbol{\zeta}$

$$\boldsymbol{\zeta} : \begin{cases} \boldsymbol{\Sigma}_1 : \quad \boldsymbol{E}\dot{\boldsymbol{x}}_1(t) = \boldsymbol{A}\boldsymbol{x}_1(t) + \boldsymbol{B}\boldsymbol{u}(t), \qquad \boldsymbol{x}_1(0) = \boldsymbol{x}_0, \\ \boldsymbol{\Sigma}_k : \quad \boldsymbol{E}\dot{\boldsymbol{x}}_k(t) = \boldsymbol{A}\boldsymbol{x}_k(t) + \sum_{j=1}^{m} \boldsymbol{N}_j \boldsymbol{x}_{k-1}(t) u_j(t), \qquad \boldsymbol{x}_k(0) = \boldsymbol{0} \quad \text{for } k \geq 2. \end{cases}$$

Making things more understandable we start applying the *Picard* iteration to a SISO bilinear system $\zeta$

$$\zeta : \begin{cases} \boldsymbol{E}\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{N}\boldsymbol{x}(t)u(t) + \boldsymbol{b}u(t), \quad \boldsymbol{x}(t_0) = \boldsymbol{x}_0, \\ \quad y(t) = \boldsymbol{c}^{\mathsf{T}}\boldsymbol{x}(t). \end{cases}$$

To gain a structure as in (2.16) we need to perform a change in the variable $\boldsymbol{x}(t)$ as follows [8]

$$\begin{aligned}\boldsymbol{x}(t) &= \mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}t}\boldsymbol{z}(t),\\ \dot{\boldsymbol{x}}(t) &= \mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}t}\dot{\boldsymbol{z}}(t) + \boldsymbol{E}^{-1}\boldsymbol{A}\mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}t}\boldsymbol{z}(t).\end{aligned}$$

Applying the transformation to $\zeta$ results in the equivalent representation

$$\begin{gathered}\mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}t}\dot{\boldsymbol{z}}(t) + \boldsymbol{E}^{-1}\boldsymbol{A}\,\mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}t}\boldsymbol{z}(t) = \boldsymbol{E}^{-1}\boldsymbol{A}\,\mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}t}\boldsymbol{z}(t) + \boldsymbol{E}^{-1}\boldsymbol{N}\,\mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}t}\boldsymbol{z}(t)\,u(t) + \boldsymbol{E}^{-1}\,\boldsymbol{b}\,u(t),\\ \boldsymbol{z}(t_0) = \boldsymbol{x}_0.\end{gathered}$$

Making use of $\hat{\boldsymbol{N}}(t) = \mathrm{e}^{-\boldsymbol{E}^{-1}\boldsymbol{A}t}\boldsymbol{E}^{-1}\boldsymbol{N}\mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}t}$, $\hat{\boldsymbol{b}}(t) = \mathrm{e}^{-\boldsymbol{E}^{-1}\boldsymbol{A}t}\boldsymbol{E}^{-1}\boldsymbol{b}$ and being aware that $\boldsymbol{E}^{-1}\boldsymbol{A}\,\mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}t}\boldsymbol{z}(t)$ cancels out, we can write

$$\dot{\boldsymbol{z}}(t) = \hat{\boldsymbol{N}}(t)\boldsymbol{z}(t)u(t) + \hat{\boldsymbol{b}}(t)u(t)\,, \qquad \boldsymbol{z}(t_0) = \boldsymbol{x}_0. \tag{2.18}$$

Note that we do not apply the transformation to the output equation as we will later change back to the original variable.
Assuming that $\boldsymbol{z}(t_0) = \boldsymbol{0}$ as well as the right-hand side of the first equation of (2.18) fulfills the necessary *Lipschitz* condition we can apply the *Picard* iteration. The solution of (2.18) can then be constructed by

$$\begin{aligned}\boldsymbol{z}_0(t) &= \boldsymbol{z}_0 = \boldsymbol{0}\\ \boldsymbol{z}_1(t) &= \underbrace{\boldsymbol{z}_0}_{=\boldsymbol{0}} + \underbrace{\int_{\tau_1=t_0}^{t} \hat{\boldsymbol{N}}(\tau_1)\boldsymbol{z}_0(\tau_1)u(\tau_1)\,\mathrm{d}\tau_1}_{=\boldsymbol{0}} + \int_{\tau_1=t_0}^{t} \hat{\boldsymbol{b}}(\tau_1)u(\tau_1)\,\mathrm{d}\tau_1\\ \boldsymbol{z}_2(t) &= \underbrace{\boldsymbol{z}_0}_{=\boldsymbol{0}} + \int_{\tau_2=t_0}^{t} \hat{\boldsymbol{N}}(\tau_2)\boldsymbol{z}_1(\tau_2)u(\tau_2)\,\mathrm{d}\tau_2 + \int_{\tau_2=t_0}^{t} \hat{\boldsymbol{b}}(\tau_2)u(\tau_2)\,\mathrm{d}\tau_2\\ &= \int_{\tau_2=t_0}^{t} \hat{\boldsymbol{N}}(\tau_2)\underbrace{\int_{\tau_1=t_0}^{\tau_2} \hat{\boldsymbol{b}}(\tau_1)u(\tau_1)\,\mathrm{d}\tau_1}_{=\boldsymbol{z}_1(\tau_2)} u(\tau_2)\,\mathrm{d}\tau_2 + \int_{\tau_2=t_0}^{t} \hat{\boldsymbol{b}}(\tau_2)u(\tau_2)\,\mathrm{d}\tau_2\\ &= \int_{\tau_2=t_0}^{t}\int_{\tau_1=t_0}^{\tau_2} \hat{\boldsymbol{N}}(\tau_2)\hat{\boldsymbol{b}}(\tau_1)u(\tau_1)u(\tau_2)\,\mathrm{d}\tau_1\,\mathrm{d}\tau_2 + \int_{\tau_2=t_0}^{t} \hat{\boldsymbol{b}}(\tau_2)u(\tau_2)\,\mathrm{d}\tau_2\\ &\vdots\end{aligned}$$

After we apply the iteration infinite times and make use of the fact that $\boldsymbol{z}_{k\to\infty}(t) = \boldsymbol{z}(t)$ we can write the solution $\boldsymbol{z}(t)$ of (2.18) as follows

$$\boldsymbol{z}(t) = \sum_{k=1}^{\infty}\int_{\tau_1=0}^{t}\cdots\int_{\tau_k=0}^{\tau_{k-1}} \hat{\boldsymbol{N}}(\tau_1)\cdots\hat{\boldsymbol{N}}(\tau_{k-1})\hat{\boldsymbol{b}}(\tau_k)u(\tau_k)\cdots u(\tau_1)\,\mathrm{d}\tau_k\cdots\mathrm{d}\tau_1. \tag{2.19}$$

Obviously it is not possible to iterate infinite times. Thus, we have to truncate the series in (2.19). Nevertheless, we want to provide error measures and proof that the series in (2.19) converges so that we can truncate it. Therefore, we follow the idea presented in [11] and write the first equation in (2.18) in its integrated form [8]

$$\boldsymbol{z}(t) = \int_{\tau_1=0}^{t} \hat{\boldsymbol{N}}(\tau_1)\boldsymbol{z}(\tau_1)u(\tau_1)\,\mathrm{d}\tau_1 + \int_{\tau_1=0}^{t} \hat{\boldsymbol{b}}(\tau_1)u(\tau_1)\,\mathrm{d}\tau_1.$$

Clearly, we cannot compute the solution of the first integral as we do not know what the value of $\boldsymbol{z}(\tau_1)$ is. Thus, we use the definition of $\boldsymbol{z}(t)$ and write

$$\boldsymbol{z}(\tau_1) = \int_{\tau_2=0}^{\tau_1} \hat{\boldsymbol{N}}(\tau_2)\boldsymbol{z}(\tau_2)u(\tau_2)\,\mathrm{d}\tau_2 + \int_{\tau_2=0}^{\tau_1} \hat{\boldsymbol{b}}(\tau_2)u(\tau_2)\,\mathrm{d}\tau_2.$$

Now we substitute $\boldsymbol{z}(\tau_1)$ and obtain the following

$$\begin{aligned}
\boldsymbol{z}(t) = \int_{\tau_1=0}^{t} \hat{\boldsymbol{N}}(\tau_1) &\underbrace{\left(\int_{\tau_2=0}^{\tau_1} \hat{\boldsymbol{N}}(\tau_2)\boldsymbol{z}(\tau_2)u(\tau_2)\,\mathrm{d}\tau_2 + \int_{\tau_2=0}^{\tau_1} \hat{\boldsymbol{b}}(\tau_2)u(\tau_2)\,\mathrm{d}\tau_2\right)}_{\boldsymbol{z}(\tau_1)} u(\tau_1)\,\mathrm{d}\tau_1 \\
&+ \int_{\tau_1=0}^{t} \hat{\boldsymbol{b}}(\tau_1)u(\tau_1)\,\mathrm{d}\tau_1 \\
&= \int_{\tau_1=0}^{t}\int_{\tau_2=0}^{\tau_1} \hat{\boldsymbol{N}}(\tau_1)\hat{\boldsymbol{N}}(\tau_2)\boldsymbol{z}(\tau_2)u(\tau_2)u(\tau_1)\,\mathrm{d}\tau_2\,\mathrm{d}\tau_1 \\
&+ \int_{\tau_1=0}^{t}\int_{\tau_2=0}^{\tau_1} \hat{\boldsymbol{N}}(\tau_1)\hat{\boldsymbol{b}}(\tau_2)u(\tau_2)u(\tau_1)\,\mathrm{d}\tau_2\,\mathrm{d}\tau_1 + \int_{\tau_1=0}^{t} \hat{\boldsymbol{b}}(\tau_1)u(\tau_1)\,\mathrm{d}\tau_1.
\end{aligned}$$

Since we need $\boldsymbol{z}(\tau_2)$ let us define every $\boldsymbol{z}(\tau_k)$ analogously to $\boldsymbol{z}(\tau_1)$

$$\boldsymbol{z}(\tau_k) = \int_{\tau_{k+1}=0}^{\tau_k} \hat{\boldsymbol{N}}(\tau_{k+1})\boldsymbol{z}(\tau_{k+1})u(\tau_{k+1})\,\mathrm{d}\tau_{k+1} + \int_{\tau_{k+1}=0}^{\tau_k} \hat{\boldsymbol{b}}(\tau_{k+1})u(\tau_{k+1})\,\mathrm{d}\tau_{k+1}.$$

After $N$ substitutions we acquire

$$\boldsymbol{z}(t) = \int_{\tau_1=0}^{t}\cdots\int_{\tau_N=0}^{\tau_{N-1}} \hat{\boldsymbol{N}}(\tau_1)\cdots\hat{\boldsymbol{N}}(\tau_N)\boldsymbol{z}(\tau_N)u(\tau_N)\cdots u(\tau_1)\,\mathrm{d}\tau_N\cdots\mathrm{d}\tau_1 \tag{2.20a}$$

$$+ \sum_{k=1}^{N}\int_{\tau_1=0}^{t}\cdots\int_{\tau_k=0}^{\tau_{k-1}} \hat{\boldsymbol{N}}(\tau_1)\cdots\hat{\boldsymbol{N}}(\tau_{k-1})\hat{\boldsymbol{b}}(\tau_k)u(\tau_k)\cdots u(\tau_1)\,\mathrm{d}\tau_k\cdots\mathrm{d}\tau_1. \tag{2.20b}$$

As above, it is not possible to compute the first term (2.20a) as it still depends on $\boldsymbol{z}(\tau_N)$. To be able to drop (2.20a) we have to quantify its importance. As it is shown in [7] we can assume that $\hat{\boldsymbol{N}}(t), \boldsymbol{z}(t)$ and $u(t)$ are bounded on $t \in [0,T]$ which let us find an upper limit for each of them

$$\begin{aligned}
\max \sup_{0<t<T} \|\hat{\boldsymbol{N}}(t)\| &< M, \\
\max \sup_{0<t<T} \|\boldsymbol{z}(t)\| &< Z, \\
\max \sup_{0<t<T} \|u(t)\| &< U.
\end{aligned} \tag{2.21}$$

If we make use of (2.21) we are able to estimate how much weight (2.20a) adds to the solution $\boldsymbol{z}(t)$. It follows

$$\begin{aligned}
&\underbrace{\| \int_{\tau_1=0}^{t}\cdots\int_{\tau_N=0}^{\tau_{N-1}} \hat{\boldsymbol{N}}(\tau_1)\cdots\hat{\boldsymbol{N}}(\tau_N)\boldsymbol{z}(\tau_N)u(\tau_N)\cdots u(\tau_1)\,\mathrm{d}\tau_N\cdots\mathrm{d}\tau_1 \|}_{(2.20a)} \\
&\leq \| \int_{\tau_1=0}^{t}\cdots\int_{\tau_N=0}^{\tau_{N-1}} M\cdots MZU\cdots U\,\mathrm{d}\tau_N\cdots\mathrm{d}\tau_1 \| \\
&\leq M^N Z\, U^N \| \int_{\tau_1=0}^{t}\cdots\int_{\tau_N=0}^{\tau_{N-1}} \cdots\mathrm{d}\tau_N\cdots\mathrm{d}\tau_1 \| \\
&\leq M^N Z\, U^N \frac{t^N}{N!} \underset{\max t=T}{\underset{\uparrow}{\leq}} \frac{(MUT)^N}{N!} Z.
\end{aligned} \tag{2.22}$$

The last inequality in (2.22) shows that for $N \to \infty$ (2.20a) disappears as faculties grow faster than exponentials. Finally, we are able to write the solution

$$\boldsymbol{z}(t) = \sum_{k=1}^{\infty} \int_{\tau_1=0}^{t} \cdots \int_{\tau_k=0}^{\tau_{k-1}} \hat{\boldsymbol{N}}(\tau_1) \cdots \hat{\boldsymbol{N}}(\tau_{k-1}) \hat{\boldsymbol{b}}(\tau_k) u(\tau_k) \cdots u(\tau_1) \,\mathrm{d}\tau_k \cdots \mathrm{d}\tau_1$$

which obviously is equal to (2.19). By changing back to the original variables we receive

$$\begin{aligned} \boldsymbol{x}(t) = \sum_{k=1}^{\infty} \int_{\tau_1=0}^{t} \cdots \int_{\tau_k=0}^{\tau_{k-1}} & \mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}(t-\tau_1)} \boldsymbol{E}^{-1}\boldsymbol{N} \cdots \mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}(\tau_{k-2}-\tau_{k-1})} \boldsymbol{E}^{-1}\boldsymbol{N} \\ & \times \mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}(\tau_{k-1}-\tau_k)} \boldsymbol{E}^{-1} \boldsymbol{b} u(\tau_k) \cdots u(\tau_1) \,\mathrm{d}\tau_k \cdots \mathrm{d}\tau_1 \end{aligned} \tag{2.23}$$

which apparently yields a *Volterra* series representation in its integrated form.
Even though the equations for MIMO systems get quite messy we still apply the *Picard* iteration resulting in a more general subsystem representation which certainly holds for SISO systems as well. Therefore, we start with a MIMO system $\boldsymbol{\zeta}$

$$\boldsymbol{\zeta} : \begin{cases} \boldsymbol{E}\,\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\,\boldsymbol{x}(t) + \sum_{j=1}^{m} \boldsymbol{N}_j\,\boldsymbol{x}(t)\,u_j(t) + \boldsymbol{B}\,\boldsymbol{u}(t), \quad \boldsymbol{x}(0) = \boldsymbol{x}_0, \\ \boldsymbol{y}(t) = \boldsymbol{C}\,\boldsymbol{x}(t). \end{cases}$$

We can make use of the same change in the variable as for the SISO systems

$$\begin{aligned} \boldsymbol{x}(t) &= \mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}t} \boldsymbol{z}(t), \\ \dot{\boldsymbol{x}}(t) &= \mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}t} \dot{\boldsymbol{z}}(t) + \boldsymbol{E}^{-1}\boldsymbol{A}\mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}t} \boldsymbol{z}(t). \end{aligned}$$

Defining $\hat{\boldsymbol{N}}_j(t) = \mathrm{e}^{-\boldsymbol{E}^{-1}\boldsymbol{A}t} \boldsymbol{E}^{-1}\boldsymbol{N}_j \mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}t}$, $\hat{\boldsymbol{B}}(t) = \mathrm{e}^{-\boldsymbol{E}^{-1}\boldsymbol{A}t}\boldsymbol{E}^{-1}\boldsymbol{B}$ yields

$$\dot{\boldsymbol{z}}(t) = \sum_{j=1}^{m} \hat{\boldsymbol{N}}_j(t)\boldsymbol{z}(t)u_j(t) + \hat{\boldsymbol{B}}(t)\boldsymbol{u}(t)\,, \qquad \boldsymbol{z}(t_0) = \boldsymbol{x}_0.$$

Again, assuming that $\boldsymbol{x}_0 = \boldsymbol{0}$ we can write $\boldsymbol{z}(t)$ as follows

$$\boldsymbol{z}(t) = \int_{\tau_1=0}^{t} \sum_{j=1}^{m} \hat{\boldsymbol{N}}_j(\tau_1)\boldsymbol{z}(\tau_1)u_j(\tau_1)\,\mathrm{d}\tau_1 + \int_{\tau_1=0}^{t} \hat{\boldsymbol{B}}(\tau_1)\boldsymbol{u}(\tau_1)\,\mathrm{d}\tau_1.$$

After $N$ substitutions we receive

$$\begin{aligned} \boldsymbol{z}(t) = & \sum_{j_1=1}^{m} \cdots \sum_{j_k=1}^{m} \int_{\tau_1=0}^{t} \cdots \int_{\tau_k=0}^{\tau_{k-1}} \hat{\boldsymbol{N}}_{j_1}(\tau_1) \cdots \hat{\boldsymbol{N}}_{j_k}(\tau_k)\boldsymbol{z}(\tau_k)u_{j_k}(\tau_k) \cdots u_{j_1}(\tau_1)\,\mathrm{d}\tau_k \cdots \mathrm{d}\tau_1 \\ & + \sum_{k=1}^{N} \sum_{j_1=1}^{m} \cdots \sum_{j_{k-1}=1}^{m} \int_{\tau_1=0}^{t} \cdots \int_{\tau_k=0}^{\tau_{k-1}} \hat{\boldsymbol{N}}_{j_1}(\tau_1) \cdots \hat{\boldsymbol{N}}_{j_{k-1}}(\tau_{k-1}) && \text{(2.24a)} \\ & \qquad \times \hat{\boldsymbol{B}}(\tau_k)\boldsymbol{u}(\tau_k)u_{j_{k-1}}(\tau_{k-1}) \cdots u_{j_1}(\tau_1)\,\mathrm{d}\tau_k \cdots \mathrm{d}\tau_1 && \text{(2.24b)} \end{aligned}$$

where also $k = 1, \ldots, N$ for (2.24a). At this point we can make the same assumptions as in the SISO case to drop (2.24a) and obtain a similar *Volterra* series representation in original variables

$$\begin{aligned} \boldsymbol{x}(t) = \sum_{k=1}^{\infty} \sum_{j_1=1}^{m} \cdots \sum_{j_{k-1}=1}^{m} \int_{\tau_1=0}^{t} \cdots \int_{\tau_k=0}^{\tau_{k-1}} & \mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}(t-\tau_1)} \boldsymbol{E}^{-1}\boldsymbol{N}_{j_1} \cdots \mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}(\tau_{k-2}-\tau_{k-1})} \boldsymbol{E}^{-1}\boldsymbol{N}_{j_{k-1}} \\ & \times \mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}(\tau_{k-1}-\tau_k)} \boldsymbol{E}^{-1}\boldsymbol{B}\boldsymbol{u}(\tau_k)u_{j_{k-1}}(\tau_{k-1}) \cdots u_{j_1}(\tau_1)\,\mathrm{d}\tau_k \cdots \mathrm{d}\tau_1. \end{aligned} \tag{2.25}$$

Finally, we want to obtain the state-space equation for each subsystem. Hence, we make use of $\boldsymbol{x}(t) = \sum_{k=1}^{\infty} \boldsymbol{x}_k(t)$ and try to derive an equation for each $\boldsymbol{x}_k(t)$. Since we first have to solve the innermost integral in (2.25) we treat it as our first subsystem which is given by

$$\boldsymbol{x}_1(t) = \int_{\tau=0}^{t} \mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}(t-\tau)} \boldsymbol{E}^{-1} \boldsymbol{B} \boldsymbol{u}(\tau) \, \mathrm{d}\tau. \tag{2.26}$$

All the other subsystems are defined for $k \geq 2$ by

$$\boldsymbol{x}_k(t) = \sum_{j=1}^{m} \int_{\tau=0}^{t} \mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}(t-\tau)} \boldsymbol{E}^{-1} \boldsymbol{N}_j \boldsymbol{x}_{k-1}(t) u_j(t) \, \mathrm{d}\tau. \tag{2.27}$$

As we know from linear system theory, we can rewrite the formulations of $\boldsymbol{x}_1(t)$ and $\boldsymbol{x}_k(t)$ in state-space which yields equal equations to (2.13)

$$\begin{aligned} \boldsymbol{E}\dot{\boldsymbol{x}}_1(t) &= \boldsymbol{A}\boldsymbol{x}_1(t) + \boldsymbol{B}\boldsymbol{u}(t), \qquad \boldsymbol{x}_1(0) = \boldsymbol{0} \\ \boldsymbol{E}\dot{\boldsymbol{x}}_k(t) &= \boldsymbol{A}\boldsymbol{x}_k(t) + \sum_{j=1}^{m} \boldsymbol{N}_j \boldsymbol{x}_{k-1}(t) u_j(t), \qquad \boldsymbol{x}_k(0) = \boldsymbol{0} \quad \text{for } k \geq 2. \end{aligned} \tag{2.28}$$

*Remark* 2.1. As one can see in (2.28), the initial condition for the first subsystem is set to zero which is different compared to (2.13) where the first subsystem holds the initial condition $\boldsymbol{x}_0$. Applying the *Picard* iteration to a bilinear system while considering an arbitrary initial condition makes the equations more complex and does not contribute to understanding the process. In addition, it does not affect the convergence of the *Volterra* series as long as the eigenvalues of $\boldsymbol{E}^{-1}\boldsymbol{A}$ have negative real part so that the initial condition disappears with growing $t$. Nevertheless, due to the linear character of the first subsystem one could easily write a generalized integral formulation for the solution of $\boldsymbol{x}_1(t)$ as follows

$$\boldsymbol{x}_1(t) = \int_{\tau=0}^{t} \mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}(t-\tau)} \boldsymbol{E}^{-1} \boldsymbol{B} \boldsymbol{u}(\tau) \, \mathrm{d}\tau + \mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}t} \boldsymbol{x}_0.$$

$\triangle$

## 2.3 Transfer Functions

Generally, transfer functions describe the input-output-behavior of a system. Due to the bilinear term in the state-equation (2.2), we are not able to directly derive a transfer function for bilinear systems. Thus, we have to use the *Volterra* series representation that contains only linear systems for which we know how to obtain transfer functions.
We begin with triangular transfer functions in time domain as we derive them through the *Picard* iteration. Then we transform them in frequency domain. In this sense we introduce the multidimensional *Laplace* transform, since the kernel of the $k$-th subsystem is dependent on $k$ different variables. Hereby, the $k$-th kernel describes the impulse response, in other words the multi-variable transfer function of the $k$-th subsystem.
Finally, we take a look at regular transfer functions which are equivalent to triangular ones by a change of variables. Similar to triangular transfer functions, we start by discussing the time domain and after that the frequency domain. Note that for later use, regular transfer functions are of major importance, since we will use them for bilinear system theory and also for model reduction.

### 2.3.1 Triangular Transfer Functions

We obtain triangular transfer functions directly from the *Volterra* series representation by integrating the state-equations for each subsystem (which is exactly what we derive through the *Picard* iteration).

#### Time Domain

Our goal is to find a generalized time-domain equation for triangular transfer functions like follows

$$\begin{aligned}\boldsymbol{g}_{k,\triangle}^{(j_1,\dots,j_k)}(t_1,\dots,t_k) = \boldsymbol{C}\mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}t_k}\boldsymbol{E}^{-1}\boldsymbol{N}_{j_k}\mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}(t_{k-1}-t_k)}\boldsymbol{E}^{-1}\boldsymbol{N}_{j_{k-1}}\cdots\\ \times\boldsymbol{E}^{-1}\boldsymbol{N}_{j_2}\mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}(t_1-t_2)}\boldsymbol{E}^{-1}\boldsymbol{b}_{j_1}.\end{aligned}$$

We approach this problem by starting with the outcome of the *Picard* iteration. Therefore we briefly recapitulate (2.26) and (2.27) with which we can describe the solution $\boldsymbol{x}(t) = \sum_{k=1}^{\infty}\boldsymbol{x}_k(t)$ of the state-vector of a MIMO bilinear system

$$\boldsymbol{\zeta}:\begin{cases}\boldsymbol{E}\,\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\,\boldsymbol{x}(t) + \sum_{j=1}^{m}\boldsymbol{N}_j\,\boldsymbol{x}(t)\,u_j(t) + \boldsymbol{B}\,\boldsymbol{u}(t), \quad \boldsymbol{x}(0) = \boldsymbol{x}_0,\\ \boldsymbol{y}(t) = \boldsymbol{C}\,\boldsymbol{x}(t)\end{cases}$$

as follows

$$\begin{aligned}\boldsymbol{x}_1(t) &= \int_{\tau=0}^{t}\mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}(t-\tau)}\boldsymbol{E}^{-1}\boldsymbol{B}\boldsymbol{u}(\tau)\,\mathrm{d}\tau + \mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}t}\boldsymbol{x}_0\\ \boldsymbol{x}_k(t) &= \sum_{j=1}^{m}\int_{\tau=0}^{t}\mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}(t-\tau)}\boldsymbol{E}^{-1}\boldsymbol{N}_j\boldsymbol{x}_{k-1}(t)u_j(t)\,\mathrm{d}\tau \quad \text{for } k\geq 2.\end{aligned}$$

For more clarity we first consider a SISO ($m = 1$) system, for which $\boldsymbol{x}_1(t)$ and $\boldsymbol{x}_k(t)$ become

$$\begin{aligned}\boldsymbol{x}_1(t) &= \int_{\tau=0}^{t}\mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}(t-\tau)}\boldsymbol{E}^{-1}\boldsymbol{b}u(\tau)\,\mathrm{d}\tau + \mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}t}\boldsymbol{x}_0\\ \boldsymbol{x}_k(t) &= \int_{\tau=0}^{t}\mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}(t-\tau)}\boldsymbol{E}^{-1}\boldsymbol{N}\boldsymbol{x}_{k-1}(t)u(t)\,\mathrm{d}\tau \quad \text{for } k\geq 2.\end{aligned}$$

We gain the solution for $\boldsymbol{x}_2(t)$ by substituting $\boldsymbol{x}_1(t)$ which gives us

$$\begin{aligned}\boldsymbol{x}_2(t) &= \int_{\tau_1=0}^{t}\mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}(t-\tau_1)}\boldsymbol{E}^{-1}\boldsymbol{N}u(\tau_1)\boldsymbol{x}_1(\tau_1)\,\mathrm{d}\tau_1\\ &= \int_{\tau_1=0}^{t}\mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}(t-\tau_1)}\boldsymbol{E}^{-1}\boldsymbol{N}u(\tau_1)\Bigg(\underbrace{\int_{\tau_2=0}^{\tau_1}\mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}(\tau_1-\tau_2)}\boldsymbol{E}^{-1}\boldsymbol{b}u(\tau_2)\,\mathrm{d}\tau_2 + \mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}\tau_1}\boldsymbol{x}_0}_{\boldsymbol{x}_1(\tau_1)}\Bigg)\mathrm{d}\tau_1\\ &= \int_{\tau_1=0}^{t}\int_{\tau_2=0}^{\tau_1}\mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}(t-\tau_1)}\boldsymbol{E}^{-1}\boldsymbol{N}u(\tau_1)\mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}(\tau_1-\tau_2)}\boldsymbol{E}^{-1}\boldsymbol{b}u(\tau_2)\,\mathrm{d}\tau_2\,\mathrm{d}\tau_1\\ &+ \int_{\tau_1=0}^{t}\mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}(t-\tau_1)}\boldsymbol{E}^{-1}\boldsymbol{N}u(\tau_1)\mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}\tau_1}\boldsymbol{x}_0\,\mathrm{d}\tau_1.\end{aligned}$$

Any $\boldsymbol{x}_k(t)$ can be obtained by sequent substitution of all previous solutions. After $k$ substitutions this results in

$$\begin{aligned}\boldsymbol{x}_k(t) = &\int_{\tau_1=0}^{t} \cdots \int_{\tau_k=0}^{\tau_{k-1}} \mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}(t-\tau_1)}\boldsymbol{E}^{-1}\boldsymbol{N}u(\tau_1)\cdots \\ &\times \mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}(\tau_{k-2}-\tau_{k-1})}\boldsymbol{E}^{-1}\boldsymbol{N}u(\tau_{k-1})\mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}(\tau_{k-1}-\tau_k)}\boldsymbol{E}^{-1}\boldsymbol{b}u(\tau_k)\,\mathrm{d}\tau_k\cdots\mathrm{d}\tau_1 \\ &+ \int_{\tau_1=0}^{t} \cdots \int_{\tau_{k-1}=0}^{\tau_{k-2}} \mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}(t-\tau_1)}\boldsymbol{E}^{-1}\boldsymbol{N}u(\tau_1)\cdots \\ &\times \mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}(\tau_{k-2}-\tau_{k-1})}\boldsymbol{E}^{-1}\boldsymbol{N}u(\tau_{k-1})\mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}\tau_{k-1}}\boldsymbol{x}_0\,\mathrm{d}\tau_{k-1}\cdots\mathrm{d}\tau_1.\end{aligned} \tag{2.29}$$

To make things less confusing and improve readability let $\tilde{\boldsymbol{A}} := \boldsymbol{E}^{-1}\boldsymbol{A}$, $\tilde{\boldsymbol{N}} := \boldsymbol{E}^{-1}\boldsymbol{N}$, $\tilde{\boldsymbol{b}} := \boldsymbol{E}^{-1}\boldsymbol{b}$. Hence, we can rewrite (2.29) as follows

$$\begin{aligned}\boldsymbol{x}_k(t) = &\int_{\tau_1=0}^{t} \cdots \int_{\tau_k=0}^{\tau_{k-1}} \mathrm{e}^{\tilde{\boldsymbol{A}}(t-\tau_1)}\tilde{\boldsymbol{N}}u(\tau_1)\cdots\mathrm{e}^{\tilde{\boldsymbol{A}}(\tau_{k-2}-\tau_{k-1})}\tilde{\boldsymbol{N}}u(\tau_{k-1})\mathrm{e}^{\tilde{\boldsymbol{A}}(\tau_{k-1}-\tau_k)}\tilde{\boldsymbol{b}}u(\tau_k)\,\mathrm{d}\tau_k\cdots\mathrm{d}\tau_1 \\ &+ \int_{\tau_1=0}^{t} \cdots \int_{\tau_{k-1}=0}^{\tau_{k-2}} \mathrm{e}^{\tilde{\boldsymbol{A}}(t-\tau_1)}\tilde{\boldsymbol{N}}u(\tau_1)\cdots\mathrm{e}^{\tilde{\boldsymbol{A}}(\tau_{k-2}-\tau_{k-1})}\tilde{\boldsymbol{N}}u(\tau_{k-1})\mathrm{e}^{\tilde{\boldsymbol{A}}\tau_{k-1}}\boldsymbol{x}_0\,\mathrm{d}\tau_{k-1}\cdots\mathrm{d}\tau_1.\end{aligned} \tag{2.30}$$

We are only interested in the input-output behavior. Therefore, we can set $\boldsymbol{x}_0 = \boldsymbol{0}$ without loosing general relevance. With this assumption we can drop the second term in (2.30). The definitions for $y(t)$ and the *Volterra* series let us write the output as a series

$$y(t) = \boldsymbol{c}^\mathsf{T}\,\boldsymbol{x}(t) = \boldsymbol{c}^\mathsf{T}\left(\sum_{k=1}^{\infty}\boldsymbol{x}_k(t)\right) = \sum_{k=1}^{\infty}\boldsymbol{c}^\mathsf{T}\boldsymbol{x}_k(t) = \sum_{k=1}^{\infty}y_k(t).$$

Considering (2.30) and $\boldsymbol{x}_0 = \boldsymbol{0}$ yields following equation

$$\begin{aligned}y_k(t) = &\int_{\tau_1=0}^{t} \cdots \int_{\tau_k=0}^{\tau_{k-1}} \boldsymbol{c}^\mathsf{T}\mathrm{e}^{\tilde{\boldsymbol{A}}(t-\tau_1)}\tilde{\boldsymbol{N}}u(\tau_1)\cdots \\ &\times \mathrm{e}^{\tilde{\boldsymbol{A}}(\tau_{k-2}-\tau_{k-1})}\tilde{\boldsymbol{N}}u(\tau_{k-1})\mathrm{e}^{\tilde{\boldsymbol{A}}(\tau_{k-1}-\tau_k)}\tilde{\boldsymbol{b}}u(\tau_k)\,\mathrm{d}\tau_k\cdots\mathrm{d}\tau_1\end{aligned} \tag{2.31}$$

Unlike in [20] we are not defining the kernels directly out of (2.31). Instead, we are following [9, 8] and perform a change in the integration variables to obtain kernels without reflected arguments $-\tau_1, \dots, -\tau_k$ in the exponential functions. However, a change in variables always affects the integration limits. To prevent that, we make use of the *Heaviside* step function

$$\sigma(t) := \begin{cases} 1 & t \geq 0 \\ 0 & t < 0 \end{cases}$$

which allows us to expand the integration limits to infinity. This results in an equivalent equation for $y_k(t)$

$$\begin{aligned}y_k(t) = &\int_{\tau_1=-\infty}^{\infty} \cdots \int_{\tau_k=-\infty}^{\infty} \boldsymbol{c}^\mathsf{T}\,\underbrace{\mathrm{e}^{\tilde{\boldsymbol{A}}(t-\tau_1)}\sigma(t-\tau_1)}_{:=\mathrm{e}^{\tilde{\boldsymbol{A}}(t-\tau_1)}}\tilde{\boldsymbol{N}}\,\underbrace{u(\tau_1)\sigma(\tau_1)}_{:=u(\tau_1)}\cdots\mathrm{e}^{\tilde{\boldsymbol{A}}(\tau_{k-2}-\tau_{k-1})}\sigma(\tau_{k-2}-\tau_{k-1}) \\ &\times \tilde{\boldsymbol{N}}u(\tau_{k-1})\sigma(\tau_{k-1})\mathrm{e}^{\tilde{\boldsymbol{A}}(\tau_{k-1}-\tau_k)}\sigma(\tau_{k-1}-\tau_k)\tilde{\boldsymbol{b}}u(\tau_k)\sigma(\tau_k)\,\mathrm{d}\tau_k\cdots\mathrm{d}\tau_1.\end{aligned} \tag{2.32}$$

We only consider positive times since negative times are generally not relevant for engineering applications. By defining one-sided matrix exponentials $\mathrm{e}^{\tilde{\boldsymbol{A}}} := \mathrm{e}^{\tilde{\boldsymbol{A}}}\sigma(t)$, there is no need to further write (2.32) with *Heaviside* functions. This results in

$$y_k(t) = \int_{\tau_1=-\infty}^{\infty} \cdots \int_{\tau_k=-\infty}^{\infty} \boldsymbol{c}^{\mathsf{T}} \mathrm{e}^{\tilde{\boldsymbol{A}}(t-\tau_1)} \tilde{\boldsymbol{N}} u(\tau_1) \cdots \mathrm{e}^{\tilde{\boldsymbol{A}}(\tau_{k-2}-\tau_{k-1})} \tilde{\boldsymbol{N}} u(\tau_{k-1}) \mathrm{e}^{\tilde{\boldsymbol{A}}(\tau_{k-1}-\tau_k)} \tilde{\boldsymbol{b}} u(\tau_k)\, \mathrm{d}\tau_k \cdots \mathrm{d}\tau_1.$$

As a last step we want to get rid of the reflected arguments $-\tau_1, \dots, -\tau_k$. Hence, we perform following change in variables

$$\left.\begin{array}{lll} \tilde{\tau}_k = t-\tau_1, & \tilde{\tau}_{k-1} = t-\tau_2, \ \dots, & \tilde{\tau}_1 = t-\tau_k, \\ \tau_1 = t-\tilde{\tau}_k, & \tau_2 = t-\tilde{\tau}_{k-1}, \ \dots, & \tau_k = t-\tilde{\tau}_1, \end{array}\right\} \text{ with } \left\{ \begin{array}{rcl} \tau_1-\tau_2 & = & \tilde{\tau}_{k-1}-\tilde{\tau}_k, \\ & \vdots & \\ \tau_{k-1}-\tau_k & = & \tilde{\tau}_1-\tilde{\tau}_2. \end{array}\right. \tag{2.33}$$

After that, we redefine $\tilde{\tau}_k$ as $\tau_k$, …, $\tilde{\tau}_1$ as $\tau_1$ which finally yields the suitable expression

$$y_k(t) = \int_{\tau_1=-\infty}^{\infty} \cdots \int_{\tau_k=-\infty}^{\infty} g_k^{\triangle}(\tau_1, \dots, \tau_k) u(t-\tau_k) \cdots u(t-\tau_1)\, \mathrm{d}\tau_k \cdots \mathrm{d}\tau_1,$$

with the triangular kernels

$$g_k^{\triangle}(t_1,\dots,t_k) = \begin{cases} \boldsymbol{c}^{\mathsf{T}} \mathrm{e}^{\tilde{\boldsymbol{A}} t_k} \tilde{\boldsymbol{N}} \mathrm{e}^{\tilde{\boldsymbol{A}}(t_{k-1}-t_k)} \tilde{\boldsymbol{N}} \cdots \tilde{\boldsymbol{N}} \mathrm{e}^{\tilde{\boldsymbol{A}}(t_1-t_2)} \tilde{\boldsymbol{b}}, & 0 < t_k < \ldots < t_1 \\ \text{not yet defined}, & \text{on the surface} \\ 0, & \text{else.} \end{cases} \tag{2.34}$$

Note that the previously defined $\tilde{\boldsymbol{A}} := \boldsymbol{E}^{-1}\boldsymbol{A}$, $\tilde{\boldsymbol{N}} := \boldsymbol{E}^{-1}\boldsymbol{N}$ and $\tilde{\boldsymbol{b}} := \boldsymbol{E}^{-1}\boldsymbol{b}$ are still valid.

It is definitely possible to apply all these concepts to MIMO bilinear systems. Since the derivation would be the same, we only want to review the result. To obtain an equation for $\boldsymbol{y}_k(t)$ we can simply replace the following in (2.31)

$$\begin{aligned} y_k(t) &\to \boldsymbol{y}_k(t), \\ \boldsymbol{c}^{\mathsf{T}} &\to \boldsymbol{C}, \\ \tilde{\boldsymbol{N}} u(t) = \boldsymbol{E}^{-1}\boldsymbol{N} u(t) &\to \sum_{j=1}^{m} \boldsymbol{E}^{-1}\boldsymbol{N}_j u_j(t) = \sum_{j=1}^{m} \tilde{\boldsymbol{N}}_j u_j(t), \\ \tilde{\boldsymbol{b}} u(t) = \boldsymbol{E}^{-1}\boldsymbol{b} u(t) &\to \sum_{j=1}^{m} \boldsymbol{E}^{-1}\boldsymbol{b}_j u_j(t) = \sum_{j=1}^{m} \tilde{\boldsymbol{b}}_j u_j(t). \end{aligned}$$

Considering unique indexes $j_k$ for each summation over the inputs we obtain

$$\begin{aligned} \boldsymbol{y}_k(t) &= \int_{\tau_1=0}^{t} \cdots \int_{\tau_k=0}^{\tau_{k-1}} \boldsymbol{C} \mathrm{e}^{\tilde{\boldsymbol{A}}(t-\tau_1)} \left( \sum_{j_1=1}^{m} \tilde{\boldsymbol{N}}_{j_1} u_{j_1}(\tau_1) \right) \cdots \\ &\quad \times \mathrm{e}^{\tilde{\boldsymbol{A}}(\tau_{k-2}-\tau_{k-1})} \left( \sum_{j_{k-1}=1}^{m} \tilde{\boldsymbol{N}}_{j_{k-1}} u_{j_{k-1}}(\tau_{k-1}) \right) \mathrm{e}^{\tilde{\boldsymbol{A}}(\tau_{k-1}-\tau_k)} \left( \sum_{j_k=1}^{m} \tilde{\boldsymbol{b}}_{j_k} u_{j_k}(\tau_k) \right) \mathrm{d}\tau_k \cdots \mathrm{d}\tau_1 \\ &= \sum_{j_1=1}^{m} \cdots \sum_{j_k=1}^{m} \int_{\tau_1=0}^{t} \cdots \int_{\tau_k=0}^{\tau_{k-1}} \boldsymbol{C} \mathrm{e}^{\tilde{\boldsymbol{A}}(t-\tau_1)} \tilde{\boldsymbol{N}}_{j_1} u_{j_1}(\tau_1) \cdots \\ &\quad \times \mathrm{e}^{\tilde{\boldsymbol{A}}(\tau_{k-2}-\tau_{k-1})} \tilde{\boldsymbol{N}}_{j_{k-1}} u_{j_{k-1}}(\tau_{k-1}) \mathrm{e}^{\tilde{\boldsymbol{A}}(\tau_{k-1}-\tau_k)} \tilde{\boldsymbol{b}}_{j_k} u_{j_k}(\tau_k)\, \mathrm{d}\tau_k \cdots \mathrm{d}\tau_1. \end{aligned}$$

To come up with a similar equation as for the SISO case we can define

$$\boldsymbol{y}_k(t) = \sum_{j_1=1}^{m} \cdots \sum_{j_k=1}^{m} \boldsymbol{y}_k^{(j_1,\ldots,j_k)}(t)$$

where

$$\begin{aligned}\boldsymbol{y}_k^{(j_1,\ldots,j_k)}(t) = \int_{\tau_1=0}^{t} \cdots \int_{\tau_k=0}^{\tau_{k-1}} &\boldsymbol{C}\mathrm{e}^{\tilde{\boldsymbol{A}}(t-\tau_1)}\tilde{\boldsymbol{N}}_{j_1} u_{j_1}(\tau_1)\cdots \\ &\times \mathrm{e}^{\tilde{\boldsymbol{A}}(\tau_{k-2}-\tau_{k-1})}\tilde{\boldsymbol{N}}_{j_{k-1}} u_{j_{k-1}}(\tau_{k-1})\mathrm{e}^{\tilde{\boldsymbol{A}}(\tau_{k-1}-\tau_k)}\tilde{\boldsymbol{b}}_{j_k} u_{j_k}(\tau_k)\,\mathrm{d}\tau_k\cdots\mathrm{d}\tau_1.\end{aligned}$$

Modifying the integration limits and applying the same change in variables as in (2.33) yields

$$\boldsymbol{y}_k^{(j_1,\ldots,j_k)}(t) = \int_{\tau_1=-\infty}^{\infty} \cdots \int_{\tau_k=-\infty}^{\infty} \boldsymbol{g}_{k,\triangle}^{(j_1,\ldots,j_k)}(\tau_1,\ldots,\tau_k) u_{j_k}(t-\tau_k)\cdots u_{j_1}(t-\tau_1)\,\mathrm{d}\tau_k\cdots\mathrm{d}\tau_1 \tag{2.35}$$

and the triangular MIMO kernels

$$\boldsymbol{g}_{k,\triangle}^{(j_1,\ldots,j_k)}(t_1,\ldots,t_k) = \begin{cases} \boldsymbol{C}\mathrm{e}^{\tilde{\boldsymbol{A}}t_k}\tilde{\boldsymbol{N}}_{j_k}\mathrm{e}^{\tilde{\boldsymbol{A}}(t_{k-1}-t_k)} & 0 < t_k < \ldots < t_1 \\ \quad \times\tilde{\boldsymbol{N}}_{j_{k-1}}\cdots\tilde{\boldsymbol{N}}_{j_2}\mathrm{e}^{\tilde{\boldsymbol{A}}(t_1-t_2)}\tilde{\boldsymbol{b}}_{j_1}, & \\ \text{not yet defined,} & \text{on the surface} \\ \boldsymbol{0}, & \text{else.} \end{cases} \tag{2.36}$$

## Multidimensional Laplace Transform

As one can clearly see, $\boldsymbol{g}_{k,\triangle}^{(j_1,\ldots,j_k)}(t_1,\ldots,t_k)$ is dependent on $k$ different variables. To still be able to transform that in frequency domain, we have to introduce the multidimensional *Laplace* transform. First, we want to briefly recapitulate the single variable *Laplace* transform.

**Definition 2.3** (*Laplace* transform)**.** The *Laplace* transform of a one-sided, real-valued function $f(t)$ is given by

$$F(s) := \mathcal{L}\{f(t)\}(s) := \int_0^{\infty} f(t)\mathrm{e}^{-st}\,\mathrm{d}t$$

if the integral converges. That is the case if the complex variable $s$ is in the complex half-plane $H_\gamma$ such that

$$s \in H_\gamma = \{s \in \mathbb{C} |\ \mathrm{Re}(s) > \gamma\}\,.$$

▲

With linear systems we usually deal with exponential functions for which the *Laplace* transform always exists. Hence, we do not dig into detail with the convergence criterion. Generalization for multi-variable functions $f(t_1, \ldots, t_k)$ is straightforward.

**Definition 2.4** (Multidimensional *Laplace* transform)**.** The $k$-dimensional *Laplace* transform of a one-sided, real-valued function $f(t_1, \ldots, t_k)$ is given by

$$\begin{aligned} F(s_1, \ldots, s_k) &:= \mathcal{L}_k\{f(t_1, \ldots, t_k)\}(s_1, \ldots, s_k) \\ &:= \int_{t_1=0}^{\infty} \cdots \int_{t_k=0}^{\infty} f(t_1, \ldots, t_k) \mathrm{e}^{-s_1 t_1} \cdots \mathrm{e}^{-s_k t_k} \, \mathrm{d}t_k \cdots \mathrm{d}t_1 \end{aligned}$$

if the integrals converge. That is the case if the complex variables $s_1, \ldots, s_k$ are in the complex half-space $H_\gamma$ such that

$$\boldsymbol{s} = (s_1 \ \ \ldots \ \ s_k)^\mathsf{T} \in H_{\gamma_1, \ldots, \gamma_k} := H_\gamma = \left\{ \boldsymbol{s} \in \mathbb{C}^k \middle| \ \mathrm{Re}(s_i) > \gamma_i, \ i = 1, \ldots, k \right\}.$$

▲

Since we are dealing with a sum of linear systems, with which we represent a bilinear system, we are basically dealing exclusively with exponential functions for which, as mentioned, the integral of the *Laplace* transform converges. Therefore, we again do not bother much about the existence of $F(s_1, \ldots, s_k)$ [20].

## Frequency Domain

Now that we have the mathematical tools to apply the *Laplace* transform to multi-variable functions, we want to obtain a frequency domain triangular transfer function

$$\begin{aligned} \boldsymbol{G}_{k,\triangle}^{(j_1, \ldots, j_k)}(s_1, \ldots, s_k) = \boldsymbol{C}\big((s_1 + \ldots + s_k)\mathbf{I} - \boldsymbol{E}^{-1}\boldsymbol{A}\big)^{-1}\boldsymbol{E}^{-1}\boldsymbol{N}_{j_k} \cdots \\ \boldsymbol{E}^{-1}\boldsymbol{N}_{j_3}\big((s_1 + s_2)\mathbf{I} - \boldsymbol{E}^{-1}\boldsymbol{A}\big)^{-1}\boldsymbol{E}^{-1}\boldsymbol{N}_{j_2}\big(s_1\mathbf{I} - \boldsymbol{E}^{-1}\boldsymbol{A}\big)^{-1}\boldsymbol{E}^{-1}\boldsymbol{b}_{j_1}. \end{aligned}$$

Remembering the definitions of $\tilde{\boldsymbol{A}} := \boldsymbol{E}^{-1}\boldsymbol{A}$, $\tilde{\boldsymbol{N}}_j := \boldsymbol{E}^{-1}\boldsymbol{N}_j$, $\tilde{\boldsymbol{b}}_j := \boldsymbol{E}^{-1}\boldsymbol{b}_j$ while making use of (2.36) we can start determining the triangular transfer function in frequency domain as follows

$$\begin{aligned} \boldsymbol{G}_{k,\triangle}^{(j_1, \ldots, j_k)}(s_1, \ldots, s_k) &:= \mathcal{L}_k\{\boldsymbol{g}_{k,\triangle}^{(j_1, \ldots, j_k)}(t_1, \ldots, t_k)\}(s_1, \ldots, s_k) \\ &= \int_{t_1=-\infty}^{\infty} \cdots \int_{t_k=-\infty}^{\infty} \underbrace{\boldsymbol{C}\mathrm{e}^{\tilde{\boldsymbol{A}}t_k}\tilde{\boldsymbol{N}}_{j_k}\mathrm{e}^{\tilde{\boldsymbol{A}}(t_{k-1}-t_k)}\tilde{\boldsymbol{N}}_{j_{k-1}} \cdots \tilde{\boldsymbol{N}}_{j_2}\mathrm{e}^{\tilde{\boldsymbol{A}}(t_1-t_2)}\tilde{\boldsymbol{b}}_{j_1}}_{\boldsymbol{g}_{k,\triangle}^{(j_1, \ldots, j_k)}(t_1, \ldots, t_k)} \\ &\qquad \times \mathrm{e}^{-s_1 t_1} \cdots \mathrm{e}^{-s_{k-1}t_{k-1}}\mathrm{e}^{-s_k t_k}\sigma(t_1) \cdots \sigma(t_k) \, \mathrm{d}t_k \cdots \mathrm{d}t_1 . \end{aligned}$$

Substituting the integration variables

$$\left.\begin{aligned} \tilde{t}_k = t_k, \quad \tilde{t}_{k-1} = t_{k-1} - t_k, \quad \ldots, \quad \tilde{t}_1 = t_1 - t_2, \\ t_k = \tilde{t}_k, \quad t_{k-1} = \tilde{t}_{k-1} + t_k, \quad \ldots, \quad t_1 = \tilde{t}_1 + t_2, \end{aligned}\right\} \text{ with } \left\{\begin{aligned} &t_{k-1} = \tilde{t}_{k-1} + t_k, \\ &t_{k-2} = \tilde{t}_{k-2} + t_{k-1} = \tilde{t}_{k-2} + \tilde{t}_{k-1} + t_k, \\ &\quad\vdots \\ &\quad t_1 = \tilde{t}_1 + t_2 = \tilde{t}_1 + \ldots + \tilde{t}_{k-1} + t_k \end{aligned}\right.$$

and redefining $\tilde{t}_k$ as $t_k$, ..., $\tilde{t}_1$ as $t_1$ leads to

$$\boldsymbol{G}_{k,\triangle}^{(j_1,\dots,j_k)}(s_1,\dots,s_k) = \int_{t_1=-\infty}^{\infty} \cdots \int_{t_k=-\infty}^{\infty} \boldsymbol{C}\mathrm{e}^{\tilde{\boldsymbol{A}}t_k}\tilde{\boldsymbol{N}}_{j_k}\mathrm{e}^{\tilde{\boldsymbol{A}}(t_{k-1}-t_k)}\tilde{\boldsymbol{N}}_{j_{k-1}}\cdots\tilde{\boldsymbol{N}}_{j_2}\mathrm{e}^{\tilde{\boldsymbol{A}}(t_1-t_2)}\tilde{\boldsymbol{b}}_{j_1}$$
$$\times\, \mathrm{e}^{-s_1(t_1+\ldots+t_k)}\mathrm{e}^{-s_2(t_2+\ldots+t_k)}\ldots\mathrm{e}^{-s_k t_k}\sigma(t_1+\ldots+t_k)\cdots\sigma(t_k)\,\mathrm{d}t_k\cdots\mathrm{d}t_1.$$

To be able to compute the integrals, we rearrange the exponential functions so that each $t_k$ gets separated

$$\boldsymbol{G}_{k,\triangle}^{(j_1,\dots,j_k)}(s_1,\dots,s_k) = \boldsymbol{C}\int_{t_k=0}^{\infty}\mathrm{e}^{\left(\tilde{\boldsymbol{A}}-(s_1+\ldots+s_k)\mathbf{I}\right)t_k}\,\mathrm{d}t_k\,\tilde{\boldsymbol{N}}_{j_k}\cdots$$
$$\times\,\tilde{\boldsymbol{N}}_{j_3}\int_{t_2=0}^{\infty}\mathrm{e}^{\left(\tilde{\boldsymbol{A}}-(s_1+s_2)\mathbf{I}\right)t_2}\,\mathrm{d}t_2\tilde{\boldsymbol{N}}_{j_2}\int_{t_1=0}^{\infty}\mathrm{e}^{\left(\tilde{\boldsymbol{A}}-s_1\mathbf{I}\right)t_1}\,\mathrm{d}t_1\,\tilde{\boldsymbol{b}}_{j_1}.$$

Evaluating the integrals finally yields the formula for the triangular transfer functions in frequency domain

$$\begin{aligned}&\boldsymbol{G}_{k,\triangle}^{(j_1,\dots,j_k)}(s_1,\dots,s_k)\\&=\boldsymbol{C}\big((s_1+\ldots+s_k)\mathbf{I}-\tilde{\boldsymbol{A}}\big)^{-1}\tilde{\boldsymbol{N}}_{j_k}\cdots\tilde{\boldsymbol{N}}_{j_3}\big((s_1+s_2)\mathbf{I}-\tilde{\boldsymbol{A}}\big)^{-1}\tilde{\boldsymbol{N}}_{j_2}\big(s_1\mathbf{I}-\tilde{\boldsymbol{A}}\big)^{-1}\tilde{\boldsymbol{b}}_{j_1}.\end{aligned}$$

*Remark* 2.2 (Triangular Input-Output Behavior). Even tough we claim that we have derived transfer functions, this is not completely true. $\boldsymbol{G}_{k,\triangle}^{(j_1,\dots,j_k)}(s_1,\dots,s_k)$ does not represent the input-output behavior of a bilinear system since we are not considering the input terms $u_j(t-\tau_k)$. $\boldsymbol{G}_{k,\triangle}^{(j_1,\dots,j_k)}(s_1,\dots,s_k)$ is rather just the *Laplace* transformed kernel. In other words, our definition for $\boldsymbol{G}_{k,\triangle}^{(j_1,\dots,j_k)}(s_1,\dots,s_k)$ does not yield $\boldsymbol{Y}(s)$ like in the linear case $\boldsymbol{Y}(s)=\boldsymbol{G}(s)\boldsymbol{U}(s)$. Following [9, 8] we can still provide a formula for $\boldsymbol{Y}(s)$ by transforming the whole equation (2.35) at once into the frequency domain. This reveals the actual *Laplace* transformed input-output behavior

$$\boldsymbol{Y}_k^{\triangle}(s_1,\dots,s_k)=\sum_{j_1=1}^{m}\cdots\sum_{j_k=1}^{m}\boldsymbol{Y}_{k,\triangle}^{(j_1,\dots,j_k)}(s_1,\dots,s_k)$$
$$=\sum_{j_1=1}^{m}\cdots\sum_{j_k=1}^{m}\underbrace{\boldsymbol{C}\big((s_1+\ldots+s_k)\mathbf{I}-\tilde{\boldsymbol{A}}\big)^{-1}\tilde{\boldsymbol{N}}_{j_k}\cdots\tilde{\boldsymbol{N}}_{j_2}\big(s_1\mathbf{I}-\tilde{\boldsymbol{A}}\big)^{-1}\tilde{\boldsymbol{b}}_{j_1}}_{\boldsymbol{G}_{k,\triangle}^{(j_1,\dots,j_k)}(s_1,\dots,s_k)}U_{j_k}(s_k)\cdots U_{j_1}(s_1).$$

△

## 2.3.2 Regular Transfer Functions

Regular transfer functions are generally used for model reduction. The multi-variable sums $s_1+\ldots+s_k$ within the triangular transfer functions are on the one hand difficult to handle and on the other hand numerically inefficient. Therefore, we aim to derive the regular transfer functions with single variable factors such that

$$\boldsymbol{G}_{k,\square}^{(j_1,\dots,j_k)}(s_1,\dots,s_k)=\boldsymbol{C}(s_k\mathbf{I}-\boldsymbol{E}^{-1}\boldsymbol{A})^{-1}\boldsymbol{E}^{-1}\boldsymbol{N}_{j_k}\cdots$$
$$\times\,\boldsymbol{E}^{-1}\boldsymbol{N}_{j_3}(s_2\mathbf{I}-\boldsymbol{E}^{-1}\boldsymbol{A})^{-1}\boldsymbol{E}^{-1}\boldsymbol{N}_{j_2}(s_1\mathbf{I}-\boldsymbol{E}^{-1}\boldsymbol{A})^{-1}\boldsymbol{E}^{-1}\boldsymbol{b}_{j_1}.$$

**Time Domain**

As mentioned, the triangular and regular representation are connected by a change in variables. Looking at (2.35) and (2.36) we can simplify the terms $\mathrm{e}^{\tilde{\boldsymbol{A}}(t_{k-1}-t_k)}$ with the following transformation in variables

$$\left.\begin{array}{l} \tilde{\tau}_k = \tau_k, \quad \tilde{\tau}_{k-1} = \tau_{k-1} - \tau_k, \quad \dots, \quad \tilde{\tau}_1 = \tau_1 - \tau_2, \\ \tau_k = \tilde{\tau}_k, \quad \tau_{k-1} = \tau_k + \tilde{\tau}_{k-1}, \quad \dots, \quad \tau_1 = \tau_2 + \tilde{\tau}_1, \end{array}\right\} \text{ with } \left\{\begin{array}{l} \tau_{k-1} = \tau_k + \tilde{\tau}_{k-1} = \tilde{\tau}_k + \tilde{\tau}_{k-1}, \\ \quad\vdots \\ \tau_1 = \tau_2 + \tilde{\tau}_1 = \tilde{\tau}_k + \ldots + \tilde{\tau}_1. \end{array}\right.$$

By redefining $\tilde{\tau}_k$ as $\tau_k, \dots, \tilde{\tau}_1$ as $\tau_1$ we can write (2.35) as

$$\boldsymbol{y}_k^{(j_1,\dots,j_k)}(t) = \int_{\tau_1=-\infty}^{\infty} \cdots \int_{\tau_k=-\infty}^{\infty} \boldsymbol{g}_{k,\square}^{(j_1,\dots,j_k)}(\tau_1,\dots,\tau_k) u_{j_k}(t-\tau_k) \cdots u_{j_1}(t-\tau_k-\ldots-\tau_1)\,\mathrm{d}\tau_k \cdots \mathrm{d}\tau_1$$

with the regular kernels

$$\boldsymbol{g}_{k,\square}^{(j_1,\dots,j_k)}(t_1,\dots,t_k) = \begin{cases} \boldsymbol{C}\mathrm{e}^{\tilde{\boldsymbol{A}}t_k}\tilde{\boldsymbol{N}}_{j_k}\mathrm{e}^{\tilde{\boldsymbol{A}}t_{k-1}}\tilde{\boldsymbol{N}}_{j_{k-1}}\cdots\tilde{\boldsymbol{N}}_{j_2}\mathrm{e}^{\tilde{\boldsymbol{A}}t_1}\tilde{\boldsymbol{b}}_{j_1}, & t_1,\dots,t_k > 0 \\ \text{not yet defined,} & \text{on the surface} \\ 0, & \text{else.} \end{cases}$$

Note that it still holds $\tilde{\boldsymbol{A}} := \boldsymbol{E}^{-1}\boldsymbol{A}$, $\tilde{\boldsymbol{N}}_j := \boldsymbol{E}^{-1}\boldsymbol{N}_j$, $\tilde{\boldsymbol{b}}_j := \boldsymbol{E}^{-1}\boldsymbol{b}_j$.

**Frequency Domain**

Like the triangular transfer functions we can also give a frequency domain representation of the regular kernels. Therefore, we again make use of the definition of the multidimensional *Laplace* transform

$$\begin{aligned} \boldsymbol{G}_{k,\square}^{(j_1,\dots,j_k)}(s_1,\dots,s_k) &:= \mathcal{L}_k\{\boldsymbol{g}_{k,\square}^{(j_1,\dots,j_k)}(t_1,\dots,t_k)\}(s_1,\dots,s_k) \\ &= \int_{t_1=-\infty}^{\infty} \cdots \int_{t_k=-\infty}^{\infty} \underbrace{\boldsymbol{C}\mathrm{e}^{\tilde{\boldsymbol{A}}t_k}\tilde{\boldsymbol{N}}_{j_k}\mathrm{e}^{\tilde{\boldsymbol{A}}t_{k-1}}\tilde{\boldsymbol{N}}_{j_{k-1}}\cdots\tilde{\boldsymbol{N}}_{j_2}\mathrm{e}^{\tilde{\boldsymbol{A}}t_1}\tilde{\boldsymbol{b}}_{j_1}}_{\boldsymbol{g}_{k,\square}^{(j_1,\dots,j_k)}(t_1,\dots,t_k)} \\ &\qquad \times \mathrm{e}^{-s_1t_1}\cdots\mathrm{e}^{-s_{k-1}t_{k-1}}\mathrm{e}^{-s_kt_k}\sigma(t_1)\cdots\sigma(t_k)\,\mathrm{d}t_k\cdots\mathrm{d}t_1\,. \end{aligned}$$

Here, we can directly resort the terms to be able to compute the integral which yields

$$\boldsymbol{G}_{k,\square}^{(j_1,\dots,j_k)}(s_1,\dots,s_k) = \boldsymbol{C}\int_{t_k=0}^{\infty} \mathrm{e}^{\left(\tilde{\boldsymbol{A}}-s_k\mathbf{I}\right)t_k}\,\mathrm{d}t_k\,\tilde{\boldsymbol{N}}_{j_k}\cdots\tilde{\boldsymbol{N}}_{j_2}\int_{t_1=0}^{\infty}\mathrm{e}^{\left(\tilde{\boldsymbol{A}}-s_1\mathbf{I}\right)t_1}\,\mathrm{d}t_1\,\tilde{\boldsymbol{b}}_{j_1}.$$

Therefore the *Laplace* transform of the regular kernels is given by

$$\boldsymbol{G}_{k,\square}^{(j_1,\dots,j_k)}(s_1,\dots,s_k) = \boldsymbol{C}(s_k\mathbf{I}-\tilde{\boldsymbol{A}})^{-1}\tilde{\boldsymbol{N}}_{j_k}\cdots\tilde{\boldsymbol{N}}_{j_3}(s_2\mathbf{I}-\tilde{\boldsymbol{A}})^{-1}\tilde{\boldsymbol{N}}_{j_2}(s_1\mathbf{I}-\tilde{\boldsymbol{A}})^{-1}\tilde{\boldsymbol{b}}_{j_1}.$$

*Remark* 2.3 (Efficient Handling Of $\boldsymbol{E}^{-1}$). In the large-scale setting it would not be very efficient to compute $\boldsymbol{E}^{-1}$ to obtain $\tilde{\boldsymbol{A}}$, $\tilde{\boldsymbol{N}}_j$, $\tilde{\boldsymbol{b}}_j$, since we have to come up with the inverse of $\boldsymbol{E}$ which sometimes is not possible. Writing $\mathbf{I} = \boldsymbol{E}^{-1}\boldsymbol{E}$ lets us transform $\boldsymbol{G}_{1,\square}^{(j_1)}(s)$ as follows

$$\begin{aligned} \boldsymbol{G}_1^{(j_1)}(s) &= \boldsymbol{C}(s\boldsymbol{E}^{-1}\boldsymbol{E} - \boldsymbol{E}^{-1}\boldsymbol{A})^{-1}\boldsymbol{E}^{-1}\boldsymbol{b}_{j_1} \\ &= \boldsymbol{C}\left(\boldsymbol{E}^{-1}(s\boldsymbol{E} - \boldsymbol{A})\right)^{-1}\boldsymbol{E}^{-1}\boldsymbol{b}_{j_1} \\ &= \boldsymbol{C}(s\boldsymbol{E} - \boldsymbol{A})^{-1}\boldsymbol{b}_{j_1}. \end{aligned}$$

This could be applied to each subsystem which yields the equivalent equation

$$\boldsymbol{G}_{k,\square}^{(j_1,\dots,j_k)}(s_1,\dots,s_k) = \boldsymbol{C}(s_k\boldsymbol{E} - \boldsymbol{A})^{-1}\boldsymbol{N}_{j_k}\cdots\boldsymbol{N}_{j_3}(s_2\boldsymbol{E} - \boldsymbol{A})^{-1}\boldsymbol{N}_{j_2}(s_1\boldsymbol{E} - \boldsymbol{A})^{-1}\boldsymbol{b}_{j_1}.$$

△

*Remark* 2.4 (Regular Input-Output Behavior)*.* Following the triangular transfer functions $\boldsymbol{G}_{k,\square}$ only outlines the *Laplace* transform of the regular kernels and does not represent the input-output behavior of a bilinear system. As shown in [9, 8], we can still provide a formula by transforming the whole output equation which yields the regular input-output behavior in frequency domain

$$\begin{aligned} \boldsymbol{Y}_k^{\square}(s_1,\dots,s_k) &= \sum_{j_1=1}^{m}\cdots\sum_{j_k=1}^{m}\boldsymbol{Y}_{k,\square}^{(j_1,\dots,j_k)}(s_1,\dots,s_k) \\ &= \sum_{j_1=1}^{m}\cdots\sum_{j_k=1}^{m}\underbrace{\boldsymbol{C}(s_k\boldsymbol{E} - \boldsymbol{A})^{-1}\boldsymbol{N}_{j_k}\cdots\boldsymbol{N}_{j_2}(s_1\boldsymbol{E} - \boldsymbol{A})^{-1}\boldsymbol{b}_{j_1}}_{\boldsymbol{G}_{k,\square}^{(j_1,\dots,j_k)}(s_1,\dots,s_k)} \\ &\qquad \times U_{j_k}(s_k - s_{k-1})\cdots U_{j_2}(s_2 - s_1)U_{j_1}(s_1). \end{aligned}$$

△

*Remark* 2.5 (*Kronecker* Notation For Transfer Functions)*.* Note that in literature usually the *Kronecker* notation is used to describe the transfer functions as follows

$$\begin{aligned} \boldsymbol{G}_k^{\square}(s_1,\dots,s_k) &= \boldsymbol{C}(s_k\boldsymbol{E} - \boldsymbol{A})^{-1}\bar{\boldsymbol{N}}(\mathbf{I}_m \otimes (s_{k-1}\boldsymbol{E} - \boldsymbol{A})^{-1})(\mathbf{I}_m \otimes \bar{\boldsymbol{N}})\cdots \\ &\quad \cdots(\underbrace{\mathbf{I}_m \otimes \cdots \otimes \mathbf{I}_m}_{k-2 \text{ times}} \otimes (s_2\boldsymbol{E} - \boldsymbol{A})^{-1})(\underbrace{\mathbf{I}_m \otimes \cdots \otimes \mathbf{I}_m}_{k-2 \text{ times}} \otimes \bar{\boldsymbol{N}}) \\ &\quad \times(\underbrace{\mathbf{I}_m \otimes \cdots \otimes \mathbf{I}_m}_{k-1 \text{ times}} \otimes (s_1\boldsymbol{E} - \boldsymbol{A})^{-1})(\underbrace{\mathbf{I}_m \otimes \cdots \otimes \mathbf{I}_m}_{k-1 \text{ times}} \otimes \boldsymbol{B}) \\ &= \boldsymbol{C}(s_k\boldsymbol{E} - \boldsymbol{A})^{-1}\bar{\boldsymbol{N}}(\mathbf{I}_m \otimes (s_{k-1}\boldsymbol{E} - \boldsymbol{A})^{-1}\bar{\boldsymbol{N}})\cdots \\ &\quad \cdots(\underbrace{\mathbf{I}_m \otimes \cdots \otimes \mathbf{I}_m}_{k-2 \text{ times}} \otimes (s_2\boldsymbol{E} - \boldsymbol{A})^{-1}\bar{\boldsymbol{N}}) \\ &\quad \times(\underbrace{\mathbf{I}_m \otimes \cdots \otimes \mathbf{I}_m}_{k-1 \text{ times}} \otimes (s_1\boldsymbol{E} - \boldsymbol{A})^{-1}\boldsymbol{B}) \in \mathbb{R}^{p\times m^k} \end{aligned}$$

where $\bar{\boldsymbol{N}} = [\boldsymbol{N}_1,\dots,\boldsymbol{N}_m] \in \mathbb{R}^{n\times nm}$. Hereby, the $k$-th order transfer function is a multi-variable matrix function $\boldsymbol{G}_k^{\square}(s_1,\dots,s_k) \in \mathbb{R}^{p\times m^k}$ which holds all combinations for $j_1,\dots,j_k$. With the abbreviation $\boldsymbol{G}_k^{\square} := \boldsymbol{G}_k^{\square}(s_1,\dots,s_k)$ it follows [8]

$$\boldsymbol{G}_k^{\square} = \left[\boldsymbol{G}_{k,\square}^{(1,\dots,1)},\ \dots,\ \boldsymbol{G}_{k,\square}^{(1,\dots,m)},\ \dots\quad \dots,\ \boldsymbol{G}_{k,\square}^{(m,\dots,1)},\ \dots,\ \boldsymbol{G}_{k,\square}^{(m,\dots,m)}\right].$$

Including all combinations of $j_1,\dots,j_k$ might result in a more compact way to describe the transfer functions. But as one can clearly see, the dimensions increase crucially with every subsystem. Since the sum notation is more enlightening we will continue using it. Transfer functions with *Kronecker* notation can be found e.g. in [5] [9]. △

# Chapter 3

# Bilinear System Theory

In the previous chapter we fundamentally discussed bilinear systems, the *Volterra Series* representation and found a way to describe the bilinear system transfer functions. To extend our knowledge even further, we now take a closer look at the bilinear system theory. This will provide some more tools which we later need for model reduction.
Carrying on where we ended the last chapter, we discuss the pole-residue formulation of transfer functions. In this context we show the invariance of a bilinear system against transformation. Following, we specify the BIBO stability of a bilinear system. Since we still make use of the *Volterra Series*, the BIBO stability is strongly connected to the convergence of the series. To gain information about observability and reachability, we analyze the *Gram* matrices for a bilinear system. Concluding, we find formulations for the $\mathcal{H}_2-$norm and the $\mathcal{L}_2-$norm for bilinear systems.

## 3.1 Pole-Residue Formulation

Similar to linear systems, we are able to obtain a pole-residue formulation for the transfer functions. This means that it is possible to write the transfer function as a sum of fractions where the numerators are not dependent on the function variables $s_1, \cdots, s_k$ and the denominators are polynomials whose roots are called poles. Since the bilinear transfer function for the $k$-th subsystem depends on $k$ different variables, each of the polynomials in one of those variables has $n$ roots. Thus, the pole-residue formulation is given by [8]

$$\boldsymbol{G}_{k,\square}^{(j_1,\ldots,j_k)}(s_1,\ldots,s_k) = \sum_{l_1=1}^{n} \cdots \sum_{l_k=1}^{n} \frac{\boldsymbol{\Phi}_{l_1,\ldots,l_k}^{(j_1,\ldots,j_k)}}{\prod\limits_{\ell=1}^{k}(s_\ell - \lambda_{l_\ell})}.$$

To be able to determine the above expression, we first look at the invariance of a bilinear system — or rather the corresponding transfer function — against transformation, which establishes the possibility to diagonalize a bilinear system. With this, we then are able to easily obtain the pole-residue formulation.

## Invariance And Diagonal-Form

Let the following bilinear system

$$\boldsymbol{\zeta}:\begin{cases}\boldsymbol{E}\dot{\boldsymbol{x}}(t)=\boldsymbol{A}\boldsymbol{x}(t)+\sum_{j=1}^{m}\boldsymbol{N}_j\boldsymbol{x}(t)u_j(t)+\boldsymbol{B}\boldsymbol{u}(t), \quad \boldsymbol{x}(0)=\boldsymbol{x}_0,\\ \boldsymbol{y}(t)=\boldsymbol{C}\boldsymbol{x}(t)\end{cases}$$

have the transfer function

$$\boldsymbol{G}_{k,\square}^{(j_1,\dots,j_k)}(s_1,\dots,s_k)=\boldsymbol{C}(s_k\mathbf{I}-\boldsymbol{E}^{-1}\boldsymbol{A})^{-1}\boldsymbol{E}^{-1}\boldsymbol{N}_{j_k}\cdots\boldsymbol{E}^{-1}\boldsymbol{N}_{j_2}(s_1\mathbf{I}-\boldsymbol{E}^{-1}\boldsymbol{A})^{-1}\boldsymbol{E}^{-1}\boldsymbol{b}_{j_1}.$$

By applying a change in the variable

$$\begin{aligned}\boldsymbol{x}(t)&=\boldsymbol{T}\boldsymbol{z}(t),\\ \dot{\boldsymbol{x}}(t)&=\boldsymbol{T}\dot{\boldsymbol{z}}(t)\end{aligned}$$

we gain a transformed bilinear system

$$\tilde{\boldsymbol{\zeta}}:\begin{cases}\boldsymbol{E}\boldsymbol{T}\dot{\boldsymbol{z}}(t)=\boldsymbol{A}\boldsymbol{T}\boldsymbol{z}(t)+\sum_{j=1}^{m}\boldsymbol{N}_j\boldsymbol{T}\boldsymbol{z}(t)u_j(t)+\boldsymbol{B}\boldsymbol{u}(t), \quad \boldsymbol{z}(0)=\boldsymbol{T}\boldsymbol{x}_0,\\ \boldsymbol{y}(t)=\boldsymbol{C}\boldsymbol{T}\boldsymbol{z}(t).\end{cases}$$

Looking at the transfer function $\boldsymbol{G}_{k,\square}^{(j_1,\dots,j_k)}(s_1,\dots,s_k)$ of the regular system and the transfer function $\tilde{\boldsymbol{G}}_{k,\square}^{(j_1,\dots,j_k)}(s_1,\dots,s_k)$ of the transformed system we can see, while writing $\mathbf{I}=\boldsymbol{T}^{-1}\mathbf{I}\boldsymbol{T}$, that the change in variable does not affect the input-output behavior. Consequently, we write [8]

$$\begin{aligned}\tilde{\boldsymbol{G}}_{k,\square}^{(j_1,\dots,j_k)}(s_1,\dots,s_k)&=\boldsymbol{C}\boldsymbol{T}(s_k\mathbf{I}-\boldsymbol{T}^{-1}\boldsymbol{E}^{-1}\boldsymbol{A}\boldsymbol{T})^{-1}\boldsymbol{T}^{-1}\boldsymbol{E}^{-1}\boldsymbol{N}_{j_k}\boldsymbol{T}\cdots\\ &\quad\times\boldsymbol{T}^{-1}\boldsymbol{E}^{-1}\boldsymbol{N}_{j_2}\boldsymbol{T}(s_1\mathbf{I}-\boldsymbol{T}^{-1}\boldsymbol{E}^{-1}\boldsymbol{A}\boldsymbol{T})^{-1}\boldsymbol{T}^{-1}\boldsymbol{E}^{-1}\boldsymbol{b}_{j_1}\\ &=\boldsymbol{C}\boldsymbol{T}(s_k\boldsymbol{T}^{-1}\mathbf{I}\boldsymbol{T}-\boldsymbol{T}^{-1}\boldsymbol{E}^{-1}\boldsymbol{A}\boldsymbol{T})^{-1}\boldsymbol{T}^{-1}\boldsymbol{E}^{-1}\boldsymbol{N}_{j_k}\boldsymbol{T}\cdots\\ &\quad\times\boldsymbol{T}^{-1}\boldsymbol{E}^{-1}\boldsymbol{N}_{j_2}\boldsymbol{T}(s_1\boldsymbol{T}^{-1}\mathbf{I}\boldsymbol{T}-\boldsymbol{T}^{-1}\boldsymbol{E}^{-1}\boldsymbol{A}\boldsymbol{T})^{-1}\boldsymbol{T}^{-1}\boldsymbol{E}^{-1}\boldsymbol{b}_{j_1}\\ &=\boldsymbol{C}\boldsymbol{T}\boldsymbol{T}^{-1}\Big((s_k\mathbf{I}-\boldsymbol{E}^{-1}\boldsymbol{A})^{-1}\boldsymbol{E}^{-1}\boldsymbol{N}_{j_k}\cdots\\ &\quad\times\boldsymbol{E}^{-1}\boldsymbol{N}_{j_2}(s_1\mathbf{I}-\boldsymbol{E}^{-1}\boldsymbol{A})^{-1}\Big)\boldsymbol{T}\boldsymbol{T}^{-1}\boldsymbol{E}^{-1}\boldsymbol{b}_{j_1}\\ &=\boldsymbol{C}(s_k\mathbf{I}-\boldsymbol{E}^{-1}\boldsymbol{A})^{-1}\boldsymbol{E}^{-1}\boldsymbol{N}_{j_k}\cdots\boldsymbol{E}^{-1}\boldsymbol{N}_{j_2}(s_1\mathbf{I}-\boldsymbol{E}^{-1}\boldsymbol{A})^{-1}\boldsymbol{E}^{-1}\boldsymbol{b}_{j_1}\\ &=\boldsymbol{G}_{k,\square}^{(j_1,\dots,j_k)}(s_1,\dots,s_k).\end{aligned}$$

With this knowledge we are able to diagonalize a bilinear system to later compute the inverse of $(s_k\mathbf{I}-\boldsymbol{E}^{-1}\boldsymbol{A})$ more easily. Since we want the pair $\boldsymbol{E}^{-1}\boldsymbol{A}$ to appear as a diagonal matrix we use the following link

$$\boldsymbol{E}^{-1}\boldsymbol{A}=\boldsymbol{X}\boldsymbol{\Lambda}\boldsymbol{X}^{-1}\Leftrightarrow\boldsymbol{X}^{-1}\boldsymbol{E}^{-1}\boldsymbol{A}\boldsymbol{X}=\boldsymbol{\Lambda}$$

where $\boldsymbol{\Lambda}$ is a diagonal matrix which holds the eigenvalues $\lambda_i$ of $\boldsymbol{E}^{-1}\boldsymbol{A}$ on its diagonal. The columns of $\boldsymbol{X}$ are the right eigenvectors of $\boldsymbol{E}^{-1}\boldsymbol{A}$. Choosing $\boldsymbol{T}=\boldsymbol{X}$ and defining

$$\hat{\boldsymbol{N}}=\boldsymbol{X}^{-1}\boldsymbol{E}^{-1}\boldsymbol{N}\boldsymbol{X},\quad\hat{\boldsymbol{b}}=\boldsymbol{X}^{-1}\boldsymbol{E}^{-1}\boldsymbol{b},\qquad\hat{\boldsymbol{c}}^{\mathsf{T}}=\boldsymbol{c}^{\mathsf{T}}\boldsymbol{X}\tag{3.1}$$

we obtain the following diagonal bilinear system

$$\zeta_{\text{diag}}: \begin{cases} \dot{\boldsymbol{z}}(t) = \boldsymbol{\Lambda}\boldsymbol{z}(t) + \sum_{j=1}^{m} \hat{\boldsymbol{N}}_j \boldsymbol{z}(t) u_j(t) + \hat{\boldsymbol{B}}\boldsymbol{u}(t), \quad \boldsymbol{z}(0) = \boldsymbol{X}^{-1}\boldsymbol{x}_0, \\ \boldsymbol{y}(t) = \hat{\boldsymbol{C}}\boldsymbol{z}(t) \end{cases}$$

with the following transfer function

$$\boldsymbol{G}^{(j_1,\dots,j_k)}_{\text{diag},k,\square}(s_1,\dots,s_k) = \hat{\boldsymbol{C}}(s_k\mathbf{I} - \boldsymbol{\Lambda})^{-1}\hat{\boldsymbol{N}}_{j_k}\cdots\hat{\boldsymbol{N}}_{j_2}(s_1\mathbf{I} - \boldsymbol{\Lambda})^{-1}\hat{\boldsymbol{b}}_{j_1}. \tag{3.2}$$

## Pole-Residue Formulation

To obtain the pole-residue formulation we make use of (3.2) and write every $(s_k\mathbf{I} - \boldsymbol{\Lambda})^{-1}$ explicitly since it is easy to compute the inverse due to its diagonal form. This happens via exclusively computing the inverse value of each diagonal element, which yields

$$\boldsymbol{G}^{(j_2,\dots,j_k)}_{\text{diag},k,\square}(s_1,\dots,s_k) = \hat{\boldsymbol{C}} \begin{bmatrix} \frac{1}{s_k-\lambda_1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{1}{s_k-\lambda_n} \end{bmatrix} \hat{\boldsymbol{N}}_{j_k}\cdots\hat{\boldsymbol{N}}_{j_2} \begin{bmatrix} \frac{1}{s_1-\lambda_1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{1}{s_1-\lambda_n} \end{bmatrix} \hat{\boldsymbol{B}}.$$

Note that we now use $\hat{\boldsymbol{B}}$ instead of $\hat{\boldsymbol{b}}_{j_1}$ to rather obtain the inputs to outputs transfer function matrix $\boldsymbol{G}^{(j_2,\dots,j_k)}_{\text{diag},k,\square} \in \mathbb{R}^{p\times m}$ than an input to outputs transfer function vector $\boldsymbol{G}^{(j_1,\dots,j_k)}_{\text{diag},k,\square} \in \mathbb{R}^{p\times 1}$. To make things easier we start with a diagonal SISO system where the transfer function is given by

$$G^{\square}_{\text{diag},k}(s_1,\dots,s_k) = \hat{\boldsymbol{c}}^{\mathsf{T}} \begin{bmatrix} \frac{1}{s_k-\lambda_1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{1}{s_k-\lambda_n} \end{bmatrix} \hat{\boldsymbol{N}}\cdots\hat{\boldsymbol{N}} \begin{bmatrix} \frac{1}{s_1-\lambda_1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{1}{s_1-\lambda_n} \end{bmatrix} \hat{\boldsymbol{b}}.$$

By evaluating all matrix products one could determine the following structure of the diagonal transfer function

$$G^{\square}_{\text{diag},k}(s_1,\dots,s_k) = \frac{\varphi_{1,\dots,1}}{(s_1-\lambda_1)\cdots(s_k-\lambda_1)} + \cdots + \frac{\varphi_{n,\dots,n}}{(s_1-\lambda_n)\cdots(s_k-\lambda_n)}. \tag{3.3}$$

Generally, we can write (3.3) in compact form and obtain

$$G^{\square}_{\text{diag},k}(s_1,\dots,s_k) = \sum_{l_1=1}^{n}\cdots\sum_{l_k=1}^{n} \frac{\varphi_{l_1,\dots,l_k}}{\prod_{\ell=1}^{k}(s_\ell - \lambda_{l_\ell})}$$

where the residues are given by

$$\varphi_{l_1,\dots,l_k} = \hat{c}_{l_k}\cdot\hat{n}_{l_k,l_{k-1}}\cdot\dots\cdot\hat{n}_{l_2,l_1}\cdot\hat{b}_{l_1} \qquad l_1 = 1,\dots,n,\ \cdots\ ,l_k = 1,\dots,n$$

and $\hat{c}, \hat{n}, \hat{b}$ describe elements of the transformed system matrices

$$\hat{\boldsymbol{c}}^{\mathsf{T}} = [\hat{c}_1,\cdots,\hat{c}_n], \quad \hat{\boldsymbol{b}} = \begin{bmatrix} \hat{b}_1 \\ \vdots \\ \hat{b}_n \end{bmatrix}, \quad \hat{\boldsymbol{N}} = \begin{bmatrix} \hat{n}_{1,1} & \cdots & \hat{n}_{1,n} \\ \vdots & \ddots & \vdots \\ \hat{n}_{n,1} & \cdots & \hat{n}_{n,n} \end{bmatrix}.$$

*Example* 3.1 (Pole-Residue Formulation For The First Subsystems). [8] To make the above more comprehensible, we want to provide some examples. Since the first transfer function is only dependent on one variable $s_1$ and does not contain the bilinear term, we can give the pole-residue formulation as follows

$$G_1(s_1) = \hat{\boldsymbol{c}}^\mathsf{T}(s_1\mathbf{I} - \boldsymbol{\Lambda})^{-1}\,\hat{\boldsymbol{b}} = \sum_{l_1=1}^{n} \frac{\hat{c}_{l_1}\cdot \hat{b}_{l_1}}{s_1 - \lambda_{l_1}}, \quad n^1 \text{ residues } \varphi_{l_1}.$$

As can be seen in the following, each additional subsystem yields another factor $\hat{n}$ in the residues, therefore contains $n$-times more residues and $n$-times more poles due to the additional variable.

$$G_2^{\square}(s_1,s_2) = \cdots = \sum_{l_1=1}^{n}\sum_{l_2=1}^{n} \frac{\hat{c}_{l_2}\cdot \hat{n}_{l_2,l_1}\cdot \hat{b}_{l_1}}{(s_1-\lambda_{l_1})\,(s_2-\lambda_{l_2})}, \quad n^2 \text{ residues } \varphi_{l_1,l_2}$$

$$G_3^{\square}(s_1,s_2,s_3) = \cdots = \sum_{l_1=1}^{n}\sum_{l_2=1}^{n}\sum_{l_3=1}^{n} \frac{\hat{c}_{l_3}\cdot \hat{n}_{l_3,l_2}\cdot \hat{n}_{l_2,l_1}\cdot \hat{b}_{l_1}}{(s_1-\lambda_{l_1})\,(s_2-\lambda_{l_2})\,(s_3-\lambda_{l_3})}, \; n^3 \text{ residues } \varphi_{l_1,l_2,l_3}$$

$$\vdots$$

△

Applying the same principle to a MIMO system yields similar to (3.3)

$$\boldsymbol{G}_{\mathrm{diag},k,\square}^{(j_2,\ldots,j_k)}(s_1,\ldots,s_k) = \frac{\boldsymbol{\Phi}_{1,\ldots,1}^{(j_2,\ldots,j_k)}}{(s_1-\lambda_1)\cdots(s_k-\lambda_1)} + \cdots + \frac{\boldsymbol{\Phi}_{n,\ldots,n}^{(j_2,\ldots,j_k)}}{(s_1-\lambda_n)\cdots(s_k-\lambda_n)}$$

which again could be generally written as [8]

$$\boldsymbol{G}_{\mathrm{diag},k,\square}^{(j_2,\ldots,j_k)}(s_1,\ldots,s_k) = \sum_{l_1=1}^{n}\cdots\sum_{l_k=1}^{n} \frac{\boldsymbol{\Phi}_{l_1,\ldots,l_k}^{(j_2,\ldots,j_k)}}{\prod_{\ell=1}^{k}(s_\ell - \lambda_{l_\ell})}.$$

This time the residues are matrices, given by

$$\boldsymbol{\Phi}_{l_1,\ldots,l_k}^{(j_2,\ldots,j_k)} = \hat{\boldsymbol{c}}_{l_k}\cdot \hat{n}_{j_{k}{}_{l_k,l_{k-1}}}\cdot\ldots\cdot \hat{n}_{j_2{}_{l_2,l_1}}\cdot \hat{\boldsymbol{b}}_{l_1}^\mathsf{T} \quad l_1,\ldots,l_k = 1,\ldots,n,$$

where

$$\hat{\boldsymbol{C}} = \boldsymbol{C}\boldsymbol{X} = [\hat{\boldsymbol{c}}_1,\cdots,\hat{\boldsymbol{c}}_n]\,,\;\hat{\boldsymbol{B}} \;= \boldsymbol{X}^{-1}\boldsymbol{E}^{-1}\boldsymbol{B} = \begin{bmatrix}\hat{\boldsymbol{b}}_1^\mathsf{T}\\ \vdots \\ \hat{\boldsymbol{b}}_n^\mathsf{T}\end{bmatrix},\; \hat{\boldsymbol{N}}_j = \boldsymbol{X}^{-1}\boldsymbol{E}^{-1}\boldsymbol{N}_j\boldsymbol{X} = \begin{bmatrix}\hat{n}_{j_{1,1}} & \cdots & \hat{n}_{j_{1,n}}\\ \vdots & \ddots & \vdots\\ \hat{n}_{j_{n,1}} & \cdots & \hat{n}_{j_{n,n}}\end{bmatrix}.$$

Hence, the pole-residue formulation for a diagonal bilinear system is directly given by evaluating the formula (3.2) for the transfer function [11].

*Remark* 3.1 (Pole-Residue Formulation From Transfer Function). It may be interesting to know that it is also possible to obtain the pole-residue formulation directly through the transfer functions. Since this method requires more complicated mathematical theorems and is not as straightforward as the one above we do not review it. If the reader is still interested we refer to [11]. △

## 3.2 BIBO Stability

In linear system theory one would start by discussing whether a system is asymptotically stable by checking if the eigenvalues of $\boldsymbol{A}$ are in the left complex half-plane. This is only possible because we know that the solution of a linear system is a matrix exponential function which disappears for $t \to \infty$ if the exponent is less than zero. Since we do not know the solution of the bilinear system without making use of the *Volterra* series we are not able to proof asymptotic stability. Nonetheless, to give some insight on stability of a bilinear system we check its BIBO stability. We know from (2.21) that the *Volterra* series converges for bounded $\boldsymbol{u}(t)$. We want to specify the constraints $\boldsymbol{u}(t)$ has to fulfill to ensure convergence and consequently BIBO stability. Therefore, we first consider a finite time interval as we did in (2.21). Finally, we look at an infinite interval to give a general constraint for the inputs $\boldsymbol{u}(t)$.
Preceding, we recapitulate the integral solutions for each subsystem while assuming $\boldsymbol{E} = \mathbf{I}$. For any bounded, regular $\boldsymbol{E}$ we do not loose general sense. Hence, we can write

$$\boldsymbol{x}_1(t) = \int_{\tau=0}^{t} \mathrm{e}^{\boldsymbol{A}(t-\tau)} \boldsymbol{B}\boldsymbol{u}(\tau)\,\mathrm{d}\tau + \mathrm{e}^{\boldsymbol{A}t}\boldsymbol{x}_0, \tag{3.4a}$$

$$\boldsymbol{x}_k(t) = \sum_{j=1}^{m} \int_{\tau=0}^{t} \mathrm{e}^{\boldsymbol{A}(t-\tau)} \boldsymbol{N}_j \boldsymbol{x}_{k-1}(t) u_j(t)\,\mathrm{d}\tau, \qquad k \geq 2. \tag{3.4b}$$

In the following we derive statements for which the state-vector $\boldsymbol{x}(t)$ is bounded but we do not provide similar conditions for $\boldsymbol{y}(t)$. Hence, one could call it bounded-input bounded-state stability. Note that as long as $\boldsymbol{C}$ and $\boldsymbol{x}(t)$ are bounded $\boldsymbol{y}(t)$ is also bounded due to the linear relation $\boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t)$. This would describe the bounded-state bounded-output stability. Since we assume that $\boldsymbol{C}$ is bounded we still call the following BIBO stability.

### 3.2.1 Finite Interval

We begin by examining a finite interval $t \in [t_0, T]$. Due to rougher assumptions for the exponential functions the estimation for $\boldsymbol{x}(t)$ will be easier to handle. Our goal is to find a bounded function which is an upper limit for $\boldsymbol{x}(t)$ as follows

$$\|\boldsymbol{x}(t)\| \leq \mathrm{e}^{(t-t_0)\Phi\Gamma K} \left((t-t_0)\Phi\beta K + \Phi\|\boldsymbol{x}_0\|\right).$$

#### Assumptions

Following [21] we make some assumptions. Let $K$ be the upper limit for all inputs

$$\|\boldsymbol{u}(t)\| \quad < \max_j \sup_{t\in\xi} \|u_j(t)\| < K \qquad \text{where} \qquad K < \infty. \tag{3.5}$$

Similar to that we can find limits $\alpha, \beta, \Gamma$ for the system matrices

$$\begin{aligned}
&\sup\|\boldsymbol{A}\| && \leq \alpha && \text{where} \quad \alpha < \infty, \\
&\sup\|\boldsymbol{B}\| && \leq \beta && \text{where} \quad \beta < \infty, \\
&\sup\|\boldsymbol{N}_j\| && \leq \eta_j && \text{where} \quad \eta_j < \infty, \\
&\Gamma = \sum_{j=1}^{m} \eta_j .
\end{aligned} \tag{3.6}$$

To avoid evaluating the exponential functions we can also assume that

$$\sup\|\mathrm{e}^{\boldsymbol{A}t}\| \quad \leq \Phi \qquad \text{where} \quad \Phi < \infty \tag{3.7}$$

while expecting all eigenvalues of $\boldsymbol{A}$ to have negative real part which results in $\mathrm{e}^{\boldsymbol{A}t} \to \boldsymbol{0}$ for $t \to \infty$.

## Condition

To come up with a formulation for $\boldsymbol{u}(t)$ we substitute above assumptions in (3.4a)

$$\begin{aligned}
\|\boldsymbol{x}_1(t)\| &\leq \int_{\tau=t_0}^{t} \|\mathrm{e}^{\boldsymbol{A}(t-\tau)}\boldsymbol{B}\boldsymbol{u}(\tau)\| \,\mathrm{d}\tau + \|\mathrm{e}^{\boldsymbol{A}(t-t_0)}\boldsymbol{x}_0\| \\
&\leq \int_{\tau=t_0}^{t} \Phi\beta K \,\mathrm{d}\tau + \Phi\|\boldsymbol{x}_0\| \\
&\leq (t-t_0)\Phi\beta K + \Phi\|\boldsymbol{x}_0\|.
\end{aligned}$$

For every subsystem where $k \geq 2$ we obtain the following

$$\begin{aligned}
\|\boldsymbol{x}_k(t)\| &\leq \int_{\tau=t_0}^{t} \|\mathrm{e}^{\boldsymbol{A}(t-\tau)} \left( \sum_{j=1}^{m} \boldsymbol{N}_j u_j(\tau) \right) \boldsymbol{x}_{k-1}(\tau)\| \,\mathrm{d}\tau \\
&\leq \int_{\tau=t_0}^{t} \Phi\Gamma K \|\boldsymbol{x}_{k-1}(\tau)\| \,\mathrm{d}\tau.
\end{aligned} \tag{3.8}$$

Using (3.8) and inserting the estimation for $\|\boldsymbol{x}_1(t)\|$ yields

$$\begin{aligned}
\|\boldsymbol{x}_2(t)\| &\leq \int_{\tau=t_0}^{t} \Phi\Gamma K \|\boldsymbol{x}_1(\tau)\| \,\mathrm{d}\tau && (3.9\text{a}) \\
&\leq \int_{\tau=t_0}^{t} \Phi\Gamma K \left[(\tau-t_0)\Phi\beta K + \Phi\|\boldsymbol{x}_0\|\right] \mathrm{d}\tau && (3.9\text{b}) \\
&\leq \frac{1}{2}(t-t_0)^2\Phi^2K^2\Gamma\beta + (t-t_0)\Phi^2\Gamma K\|\boldsymbol{x}_0\| && (3.9\text{c}) \\
&< (t-t_0)^2\Phi^2K^2\Gamma\beta + (t-t_0)\Phi^2\Gamma K\|\boldsymbol{x}_0\|. && (3.9\text{d})
\end{aligned}$$

Note that we write (3.9d) without the factor $\frac{1}{2}$ to later be able to interpret the sum over all $\boldsymbol{x}_k(t)$ as an exponential function. Estimating $\boldsymbol{x}_3(t)$ while also making another assessment similar to that from (3.9c) to (3.9d) leads to

$$\begin{aligned}
\|\boldsymbol{x}_3(t)\| &\leq \int_{\tau=t_0}^{t} \Phi\Gamma K \|\boldsymbol{x}_2(\tau)\| \,\mathrm{d}\tau \\
&\leq \int_{\tau=t_0}^{t} \Phi\Gamma K \left(\frac{1}{2}(\tau - t_0)^2(\Phi K)^2\Gamma\beta + (\tau - t_0)\Phi^2\Gamma K\|\boldsymbol{x}_0\|\right) \mathrm{d}\tau \\
&\leq \frac{1}{6}(t-t_0)^3\Phi^3K^3\Gamma^2\beta + \frac{1}{2}(t-t_0)^2\Phi^3\Gamma^2K^2\|\boldsymbol{x}_0\| \\
&< \frac{1}{2}(t-t_0)^3\Phi^3K^3\Gamma^2\beta + \frac{1}{2}(t-t_0)^2\Phi^3\Gamma^2K^2\|\boldsymbol{x}_0\|.
\end{aligned}$$

We observe in the last of $\|\boldsymbol{x}_2(t)\|$ and the last of $\|\boldsymbol{x}_3(t)\|$ that the estimations follow a certain structure. Generally, it is possible to say that every $\boldsymbol{x}_k$ is constrained by

$$\|\boldsymbol{x}_k(t)\| < \frac{1}{k-1}(t-t_0)\Phi\Gamma K\|\boldsymbol{x}_{k-1}(t)\|, \qquad k \geq 2$$

where we use the less-than sign on purpose to mark that we make another estimation by e.g. dropping the $\frac{1}{2}$ in (3.9d). To gain an expression for $\boldsymbol{x}(t)$ we make use of the *Volterra* series and write

$$\begin{aligned}
\|\boldsymbol{x}(t)\| &\leq \sum_{k=1}^{\infty}\|\boldsymbol{x}_k(t)\| \\
&< \|\boldsymbol{x}_1(t)\| + \underbrace{(t-t_0)\Phi\Gamma K\|\boldsymbol{x}_1(t)\|}_{\|\boldsymbol{x}_2(t)\|} + \underbrace{\frac{1}{2}(t-t_0)\Phi\Gamma K(t-t_0)\Phi\Gamma K\|\boldsymbol{x}_1(t)\|}_{\|\boldsymbol{x}_3(t)\|} + \dots \\
&< \sum_{\ell=0}^{\infty}\frac{[(t-t_0)\Phi\Gamma K]^{\ell}}{\ell!}\underbrace{((t-t_0)\Phi\beta K + \Phi\|\boldsymbol{x}_0\|)}_{\|\boldsymbol{x}_1(t)\|} \\
&< \mathrm{e}^{(t-t_0)\Phi\Gamma K}\left((t-t_0)\Phi\beta K + \Phi\|\boldsymbol{x}_0\|\right).
\end{aligned}$$

In the last step we apply the series representation of the exponential function $\mathrm{e}^c = \sum_{\ell=0}^{\infty}\frac{c^\ell}{\ell!}$. Concluding, we determined that $\boldsymbol{x}(t)$ is bounded by a bounded function as long as the input is bounded. This expression only holds for the finite interval $t \in [t_0, T]$ since the exponential function $\mathrm{e}^{(t-t_0)\Phi\Gamma K}$ grows to infinity otherwise. In other words: as long as all system matrices are bounded we can use any bounded input to gain BIBO stability on $t \in [t_0, T]$.

*Remark* 3.2. Comparing the result for $\|\boldsymbol{x}(t)\|$ in [21] and our result one may notice a slight difference. This follows from our additional estimation which we make to gain the exponential series expression. For sake of clarity we provide a brief explanation how to obtain the structure in [21]. Remembering the expressions for $\|\boldsymbol{x}_2(t)\|$ and $\|\boldsymbol{x}_3(t)\|$ without the additional assessment

$$\begin{aligned}
\|\boldsymbol{x}_2(t)\| &\leq \frac{1}{2}(t-t_0)^2\Phi^2K^2\Gamma\beta + (t-t_0)\Phi^2\Gamma K\|\boldsymbol{x}_0\| \\
\|\boldsymbol{x}_3(t)\| &\leq \frac{1}{6}(t-t_0)^3\Phi^3K^3\Gamma^2\beta + \frac{1}{2}(t-t_0)^2\Phi^3\Gamma^2K^2\|\boldsymbol{x}_0\|
\end{aligned}$$

we can write the *Volterra* series of $\boldsymbol{x}(t)$ as

$$\begin{aligned}\|\boldsymbol{x}(t)\| &\leq \sum_{k=1}^{\infty}\|\boldsymbol{x}_k(t)\| \\ &\leq \|\boldsymbol{x}_1(t)\| + \|\boldsymbol{x}_2(t)\| + \|\boldsymbol{x}_3(t)\| + \dots \\ &\leq (t-t_0)\Phi\beta K + \Phi\|\boldsymbol{x}_0\| + \frac{1}{2}(t-t_0)^2\Phi^2K^2\Gamma\beta + (t-t_0)\Phi^2\Gamma K\|\boldsymbol{x}_0\| + \\ &\frac{1}{6}(t-t_0)^3\Phi^3K^3\Gamma^2\beta + \frac{1}{2}(t-t_0)^2\Phi^3\Gamma^2K^2\|\boldsymbol{x}_0\| + \dots .\end{aligned}$$

While resorting the terms we write

$$\begin{aligned}\|\boldsymbol{x}(t)\| \leq{}& \Phi\|\boldsymbol{x}_0\| + (t-t_0)\Phi\beta K + (t-t_0)\Phi^2\Gamma K\|\boldsymbol{x}_0\| + \frac{1}{2}(t-t_0)^2\Phi^2K^2\Gamma\beta \\ &+ \frac{1}{2}(t-t_0)^2\Phi^3\Gamma^2K^2\|\boldsymbol{x}_0\| + \frac{1}{6}(t-t_0)^3\Phi^3K^3\Gamma^2\beta + \dots\end{aligned}$$

and make the crucial observation to finally gain the formulation as in [21], again using the exponential series definition

$$\begin{aligned}\|\boldsymbol{x}(t)\| &\leq \Phi\|\boldsymbol{x}_0\| + \left(\frac{\beta}{\Gamma} + \Phi\|\boldsymbol{x}_0\|\right)(t-t_0)\Phi\Gamma K + \frac{1}{2}\left(\frac{\beta}{\Gamma} + \Phi\|\boldsymbol{x}_0\|\right)(t-t_0)^2\Phi^2\Gamma^2K^2 + \dots \\ &\leq \Phi\|\boldsymbol{x}_0\| + \left(\frac{\beta}{\Gamma} + \Phi\|\boldsymbol{x}_0\|\right)\sum_{\ell=1}^{\infty}\frac{[(t-t_0)\Phi\Gamma K]^{\ell}}{\ell!} \\ &\leq \Phi\|\boldsymbol{x}_0\| + \left(\frac{\beta}{\Gamma} + \Phi\|\boldsymbol{x}_0\|\right)\left[\mathrm{e}^{(t-t_0)\Phi\Gamma K} - 1\right].\end{aligned}$$

△

### 3.2.2 Infinite Interval

Also following [21] we can find an expression which holds for infinite times $t \in [t_0, \infty)$. This time we try to find a more meaningful expression for $\boldsymbol{u}(t)$. Again, we make some assumptions and deduce a condition for the input afterwards such that $\boldsymbol{x}(t)$ is bounded if the following holds

$$\|\boldsymbol{u}(t)\| < \frac{a}{b\Gamma}.$$

#### Assumptions

Similar to (3.5) and (3.6) we use $K$, $\alpha$, $\beta$, $\Gamma$ as upper limits for $\boldsymbol{u}(t)$, $\boldsymbol{A}$, $\boldsymbol{B}$, $\sum_{j=1}^{m}\boldsymbol{N}_j$. The difference this time is, that we use a scalar exponential function to estimate $\mathrm{e}^{\boldsymbol{A}(t-\tau)}$ as follows

$$\|\mathrm{e}^{\boldsymbol{A}(t-\tau)}\| \leq b\,\mathrm{e}^{-a(t-\tau)}, \qquad t > \tau \tag{3.10}$$

where we again expect all eigenvalues of $\boldsymbol{A}$ in the left complex half-plane. Hereby $a, b$ are finite positive scalars.

**Condition**

With above assumptions and (3.10) we can make use of (3.4a) and are able to gain an estimation for $\boldsymbol{x}_1(t)$ similar to above

$$
\begin{aligned}
\|\boldsymbol{x}_1(t)\| &\leq \int_{\tau=t_0}^{t} \|\mathrm{e}^{\boldsymbol{A}(t-\tau)}\boldsymbol{B}\boldsymbol{u}(t)\| \,\mathrm{d}\tau + \|\mathrm{e}^{\boldsymbol{A}(t-t_0)}\boldsymbol{x}_0\| \\
&\leq \int_{\tau=t_0}^{t} b\,\mathrm{e}^{-a(t-\tau)}\beta K \,\mathrm{d}\tau + \|\boldsymbol{x}_0\| b\,\mathrm{e}^{-a(t-t_0)} \\
&\leq \frac{b\beta K}{a} - \frac{b\beta K}{a}\,\mathrm{e}^{-a(t-t_0)} + \|\boldsymbol{x}_0\| b\,\mathrm{e}^{-a(t-t_0)}.
\end{aligned}
$$

Following the same procedure as for the finite interval, we can write a formula for the upper limit for each subsystem where $k \geq 2$ as follows

$$
\begin{aligned}
\|\boldsymbol{x}_k(t)\| &\leq \int_{\tau=t_0}^{t} \|\mathrm{e}^{\boldsymbol{A}(t-\tau)} \left(\sum_{j=1}^{m} \boldsymbol{N}_j u_j(\tau)\right) \boldsymbol{x}_{k-1}(\tau)\| \,\mathrm{d}\tau \\
&\leq \int_{\tau=t_0}^{t} b\,\mathrm{e}^{-a(t-\tau)}\Gamma K \|\boldsymbol{x}_{k-1}(\tau)\| \,\mathrm{d}\tau.
\end{aligned}
\tag{3.11}
$$

Substituting $\|\boldsymbol{x}_1(t)\|$ yields an expression for the second subsystem

$$
\begin{aligned}
\|\boldsymbol{x}_2(t)\| &\leq \int_{\tau=t_0}^{t} b\,\mathrm{e}^{-a(t-\tau)}\Gamma K \|\boldsymbol{x}_1(\tau)\| \,\mathrm{d}\tau \\
&\leq \int_{\tau=t_0}^{t} b\,\mathrm{e}^{-a(t-\tau)}\Gamma K \underbrace{\left(\frac{b\beta K}{a} - \frac{b\beta K}{a}\,\mathrm{e}^{-a(\tau-t_0)} + \|\boldsymbol{x}_0\| b\,\mathrm{e}^{-a(\tau-t_0)}\right)}_{\|\boldsymbol{x}_1(t)\|} \mathrm{d}\tau \\
&\leq \int_{\tau=t_0}^{t} \frac{b^2K^2\beta\Gamma}{a}\,\mathrm{e}^{-a(t-\tau)} \,\mathrm{d}\tau + \int_{\tau=t_0}^{t} \left(b^2K\Gamma\|\boldsymbol{x}_0\| - \frac{b^2K^2\beta\Gamma}{a}\right) \mathrm{e}^{-a(t-t_0)} \,\mathrm{d}\tau \\
&\leq \frac{b^2K^2\beta\Gamma}{a} \int_{\tau=t_0}^{t} \mathrm{e}^{-a(t-\tau)} \,\mathrm{d}\tau + \left(b^2K\Gamma\|\boldsymbol{x}_0\| - \frac{b^2K^2\beta\Gamma}{a}\right) \mathrm{e}^{-a(t-t_0)} \int_{\tau=t_0}^{t} 1 \,\mathrm{d}\tau \\
&\leq \frac{b^2K^2\beta\Gamma}{a^2} - \frac{b^2K^2\beta\Gamma}{a^2}\mathrm{e}^{-a(t-t_0)} + \left(b^2K\Gamma\|\boldsymbol{x}_0\| - \frac{b^2K^2\beta\Gamma}{a}\right) \mathrm{e}^{-a(t-t_0)}\,(t-t_0) \\
&\leq b^2K\Gamma\|\boldsymbol{x}_0\|\mathrm{e}^{-a(t-t_0)}\,(t-t_0) + \frac{b^2K^2\beta\Gamma}{a^2}\left(1 - \mathrm{e}^{-a(t-t_0)} - a\mathrm{e}^{-a(t-t_0)}(t-t_0)\right).
\end{aligned}
$$

Since the preceding looks quite messy we enlighten the key steps. After substituting $\|\boldsymbol{x}_1(t)\|$ the integral gets split up into two parts. Hereby, it is important to see that $\tau$ cancels out in the second integral. Hence, for the second integral we only have to integrate one. Rearranging the terms lets us finally exclude the factors $b^2K\Gamma\|\boldsymbol{x}_0\|$ and $\frac{b^2K^2\beta\Gamma}{a^2}$. We can apply the same concept to gain an estimation for the third subsystem

$$\begin{aligned}
\|\boldsymbol{x}_3(t)\| &\leq \int_{\tau=t_0}^{t} b\,\mathrm{e}^{-a(t-\tau)}\Gamma K\|\boldsymbol{x}_2(\tau)\|\,\mathrm{d}\tau \\
&\leq \int_{\tau=t_0}^{t} b\,\mathrm{e}^{-a(t-\tau)}\Gamma K\Bigg(b^2K\Gamma\|\boldsymbol{x}_0\|\mathrm{e}^{-a(\tau-t_0)}\,(\tau-t_0) \\
&\quad + \frac{b^2K^2\beta\Gamma}{a^2}\Big(1-\mathrm{e}^{-a(\tau-t_0)}-a\mathrm{e}^{-a(\tau-t_0)}(\tau-t_0)\Big)\Bigg)\,\mathrm{d}\tau \\
&\leq \int_{\tau=t_0}^{t} b^3K^2\Gamma^2\|\boldsymbol{x}_0\|\mathrm{e}^{-a(t-t_0)}\,(\tau-t_0)\,\mathrm{d}\tau \\
&\quad + \int_{\tau=t_0}^{t} \frac{b^3K^3\beta\Gamma^2}{a^2}\Big(\mathrm{e}^{-a(t-\tau)}-\mathrm{e}^{-a(t-t_0)}-a\mathrm{e}^{-a(t-t_0)}(\tau-t_0)\Big)\,\mathrm{d}\tau \\
&\leq \frac{1}{2}b^3K^2\Gamma^2\|\boldsymbol{x}_0\|\mathrm{e}^{-a(t-t_0)}\,(\tau-t_0)^2+\frac{b^3K^3\beta\Gamma^2}{a^3}-\frac{b^3K^3\beta\Gamma^2}{a^3}\mathrm{e}^{-a(t-t_0)} \\
&\quad - \frac{b^3K^3\beta\Gamma^2}{a^2}\mathrm{e}^{-a(t-t_0)}(t-t_0)-\frac{1}{2}\frac{b^3K^3\beta\Gamma^2}{a}\mathrm{e}^{-a(t-t_0)}(t-t_0)^2 \\
&\leq \frac{1}{2}b^3K^2\Gamma^2\|\boldsymbol{x}_0\|\mathrm{e}^{-a(t-t_0)}\,(t-t_0)^2 \\
&\quad + \frac{b^3K^3\beta\Gamma^2}{a^3}\Big(1-\mathrm{e}^{-a(t-t_0)}-a\mathrm{e}^{-a(t-t_0)}(t-t_0)-\frac{1}{2}a^2\mathrm{e}^{-a(t-t_0)}(t-t_0)^2\Big)\,.
\end{aligned}$$

Note, it is again crucial that in some exponential functions the integration variable cancels out. Comparing the last big brackets in the estimation for $\|\boldsymbol{x}_2(t)\|$ and $\|\boldsymbol{x}_3(t)\|$, one can determine a certain structure of terms. Again, while making use of the *Volterra* series we can come up with an upper limit for $\boldsymbol{x}(t)$. To not get completely disorientated we comment on every crucial step. Let us begin with inserting our previous results for $\boldsymbol{x}_1(t)$, $\boldsymbol{x}_2(t)$ and $\boldsymbol{x}_3(t)$

$$\begin{aligned}
\|\boldsymbol{x}(t)\| &\leq \sum_{k=1}^{\infty}\|\boldsymbol{x}_k(t)\| \\
&\leq \|\boldsymbol{x}_1(t)\|+\|\boldsymbol{x}_2(t)\|+\|\boldsymbol{x}_3(t)\|+\dots \\
&\leq \frac{b\beta K}{a}-\frac{b\beta K}{a}\,\mathrm{e}^{-a(t-t_0)}+b\|\boldsymbol{x}_0\|\,\mathrm{e}^{-a(t-t_0)} \\
&\quad + b^2K\Gamma\|\boldsymbol{x}_0\|\mathrm{e}^{-a(t-t_0)}\,(t-t_0)+\frac{b^2K^2\beta\Gamma}{a^2}\Big(1-\mathrm{e}^{-a(t-t_0)}-a\mathrm{e}^{-a(t-t_0)}(t-t_0)\Big) \\
&\quad + \frac{1}{2}b^3K^2\Gamma^2\|\boldsymbol{x}_0\|\mathrm{e}^{-a(t-t_0)}\,(t-t_0)^2 \\
&\quad + \frac{b^3K^3\beta\Gamma^2}{a^3}\Big(1-\mathrm{e}^{-a(t-t_0)}-a\mathrm{e}^{-a(t-t_0})(t-t_0)-\frac{1}{2}a^2\mathrm{e}^{-a(t-t_0)}(t-t_0)^2\Big)+\dots\,.
\end{aligned}$$

From here we resort the terms by similar structure and show the continuing of each structure with dots. One could easily determine the two schemes in the following

$$
\begin{aligned}
&\le b\|\boldsymbol{x}_0\|\mathrm{e}^{-a(t-t_0)} + b\|\boldsymbol{x}_0\|\mathrm{e}^{-a(t-t_0)}bK\Gamma(t-t_0) + b\|\boldsymbol{x}_0\|\mathrm{e}^{-a(t-t_0)}\frac{b^2K^2\Gamma^2(t-t_0)^2}{2} + \dots \\
&+ \frac{\beta}{\Gamma}\frac{bK\Gamma}{a}\left(1-\mathrm{e}^{-a(t-t_0)}\right) + \frac{\beta}{\Gamma}\frac{b^2K^2\Gamma^2}{a^2}\left(1-\mathrm{e}^{-a(t-t_0)} - \mathrm{e}^{-a(t-t_0)}a(t-t_0)\right) \\
&+ \frac{\beta}{\Gamma}\frac{b^3K^3\Gamma^3}{a^3}\left(1-\mathrm{e}^{-a(t-t_0)} - \mathrm{e}^{-a(t-t_0)}a(t-t_0) - \mathrm{e}^{-a(t-t_0)}\frac{a^2(t-t_0)^2}{2}\right) + \dots .
\end{aligned}
$$

Writing each of the schemes as series yields

$$
\le b\|\boldsymbol{x}_0\|\mathrm{e}^{-a(t-t_0)}\sum_{\ell=0}^{\infty}\frac{[(t-t_0)b\Gamma K]^\ell}{\ell!} + \frac{\beta}{\Gamma}\sum_{m=1}^{\infty}\left(\frac{bK\Gamma}{a}\right)^m \underbrace{\left(1-\mathrm{e}^{-a(t-t_0)}\sum_{n=0}^{m-1}\frac{(a(t-t_0))^n}{n!}\right)}_{\le 1}.
$$

The last factor is always less than or equal to one, since we subtract only positive values. Dropping the last factor by setting it equal to one leads to a much more compact formula. While also expanding the second series from zero to infinity we can write

$$
\begin{aligned}
&< b\|\boldsymbol{x}_0\|\mathrm{e}^{-a(t-t_0)}\sum_{\ell=0}^{\infty}\frac{[(t-t_0)b\Gamma K]^\ell}{\ell!} + \frac{\beta}{\Gamma}\sum_{m=1}^{\infty}\left(\frac{bK\Gamma}{a}\right)^m \\
&= b\|\boldsymbol{x}_0\|\mathrm{e}^{-a(t-t_0)}\sum_{\ell=0}^{\infty}\frac{[(t-t_0)b\Gamma K]^\ell}{\ell!} + \frac{\beta}{\Gamma}\sum_{m=0}^{\infty}\left(\frac{bK\Gamma}{a}\right)^m - \frac{\beta}{\Gamma}.
\end{aligned}
$$

Finally we make use of definitions of the exponential series $\mathrm{e}^c = \sum_{\ell=0}^{\infty}\frac{c^\ell}{\ell!}$ as well as of the geometric series $\frac{1}{1-q} = \sum_{m=0}^{\infty} q^m$ where $\|q\| < 1$ to ensure convergence. Keeping the constraint of the geometric series ($\|\frac{bK\Gamma}{a}\| < 1$) in mind we can write

$$
\begin{aligned}
&= b\|\boldsymbol{x}_0\|\mathrm{e}^{-a(t-t_0)}\mathrm{e}^{(t-t_0)b\Gamma K} + \frac{\beta}{\Gamma}\left(\frac{1}{1-\frac{bK\Gamma}{a}} - 1\right) \\
&= b\|\boldsymbol{x}_0\|\mathrm{e}^{-(a-b\Gamma K)(t-t_0)} + \frac{\beta}{\Gamma}\frac{\frac{bK\Gamma}{a}}{1-\frac{bK\Gamma}{a}}.
\end{aligned}
$$

As mentioned the above expression only holds for the condition set by the geometric series. In addition to that, the exponent of the exponential function yields the same condition to ensure that the function decreases. Rewriting the inequality ($\frac{bK\Gamma}{a} < 1$) yields

$$
K < \frac{a}{b\Gamma}
$$

and since $K$ is the upper limit for our input the above reveals the requirement for BIBO stability

$$
\boxed{\|\boldsymbol{u}(t)\| < \frac{a}{b\Gamma}.}
$$

Summarizing, we obtained a condition for the maximum value of our inputs. As long as we find a scalar exponential function, which is for all times $t \in [t_0, \infty)$ greater than a norm of $\mathrm{e}^{\boldsymbol{A}(t-t_0)}$, we are able to compute the upper limit for $\boldsymbol{u}(t)$ [21].

## 3.3 Controllability And Observability Gramian

Other important system-theoretic properties are controllability and observability of a system. In simpler words: a system is called controllable if it is possible to reach a desired state starting from an arbitrary initial state with a finite input in a finite amount of time whereas a system is called observable if the state vector can be estimated from the measurement of the outputs. To describe this mathematically we first introduce the controllability *Gramian* and then the observability *Gramian* of a bilinear system. As we will discover the controllability *Gramian* can be obtained with the solution $\boldsymbol{P}$ of the following *Lyapunov* equation

$$\boldsymbol{A}\boldsymbol{P}\boldsymbol{E}^{\mathsf{T}} + \boldsymbol{E}\boldsymbol{P}\boldsymbol{A}^{\mathsf{T}} + \sum_{j=1}^{m} \boldsymbol{N}_j\boldsymbol{P}\boldsymbol{N}_j^{\mathsf{T}} = -\boldsymbol{B}\boldsymbol{B}^{\mathsf{T}}.$$

Similar to that, one could receive the observability *Gramian* $\boldsymbol{Q}$ by solving

$$\boldsymbol{A}^{\mathsf{T}}\boldsymbol{Q}\boldsymbol{E} + \boldsymbol{E}^{\mathsf{T}}\boldsymbol{Q}\boldsymbol{A} + \sum_{j=1}^{m} \boldsymbol{N}_j^{\mathsf{T}}\boldsymbol{Q}\boldsymbol{N}_j = -\boldsymbol{C}^{\mathsf{T}}\boldsymbol{C}.$$

Since we are interested in the solutions $\boldsymbol{P}$ and $\boldsymbol{Q}$ we will discuss how to solve the *Lyapunov* equations in the final part of this section.

**Controllability**

A linear system is called controllable if the matrix $\boldsymbol{Q}_c = [\boldsymbol{B}, \boldsymbol{A}\boldsymbol{B}, \cdots, \boldsymbol{A}^{n-1}\boldsymbol{B}]$ has full rank $n$. Note that in literature reachability is often used as a synonym for controllability. Since the bilinear term sometimes causes that the set of controllable states does not form a subspace of $\mathbb{R}^n$, we review the less strong span-controllability of a bilinear system. Similar to linear systems we can define $\boldsymbol{P}_1 = \boldsymbol{B}$ and $\boldsymbol{P}_i = [\boldsymbol{A}\boldsymbol{P}_{i-1},\ \boldsymbol{N}_1\boldsymbol{P}_{i-1},\ \ldots,\ \boldsymbol{N}_m\boldsymbol{P}_{i-1}]$ and $i = 1, \cdots, n$. Then a bilinear system is called span-controllable if $\boldsymbol{P}_n$ has full rank $n$ [11] [20].
Alternatively, we can follow [14] and review the controllability *Gramian*. According to the input-to-state transfer function we define

$$\begin{aligned}\boldsymbol{p}_1(t_1) &= \mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}t_1}\boldsymbol{E}^{-1}\boldsymbol{B},\\ \boldsymbol{p}_k^{(j_2,\ldots,j_k)}(t_1,\cdots,t_k) &= \mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}t_k}\boldsymbol{E}^{-1}\boldsymbol{N}_{j_k}\cdots\mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}t_2}\boldsymbol{E}^{-1}\boldsymbol{N}_{j_2}\mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}t_1}\boldsymbol{E}^{-1}\boldsymbol{B}.\end{aligned}$$

With this we can define the controllability *Gramian* as

$$\begin{aligned}\boldsymbol{P} &= \sum_{k=1}^{\infty}\boldsymbol{P}_k,\\ \boldsymbol{P}_k &= \int_{\tau_1=0}^{\infty}\cdots\int_{\tau_k=0}^{\infty}\sum_{j_2=1}^{m}\cdots\sum_{j_k=1}^{m}\boldsymbol{p}_k^{(j_2,\ldots,j_k)}(\tau_1,\ldots,\tau_k)\boldsymbol{p}_k^{(j_2,\ldots,j_k)}(\tau_1,\ldots,\tau_k)^{\mathsf{T}}\,\mathrm{d}\tau_k\cdots\mathrm{d}\tau_1.\end{aligned}$$

If $\boldsymbol{P}$ exists it is symmetric and positive semi-definite and it could be proven that $\boldsymbol{P}$ satisfies following bilinear *Lyapunov* equation

$$\boldsymbol{A}\boldsymbol{P}\boldsymbol{E}^{\mathsf{T}} + \boldsymbol{E}\boldsymbol{P}\boldsymbol{A}^{\mathsf{T}} + \sum_{j=1}^{m} \boldsymbol{N}_j\boldsymbol{P}\boldsymbol{N}_j^{\mathsf{T}} = -\boldsymbol{B}\boldsymbol{B}^{\mathsf{T}}. \tag{3.12}$$

In addition to this, every $\boldsymbol{P}_k$ solves the corresponding linear *Lyapunov* equation

$$\begin{aligned}\boldsymbol{A}\boldsymbol{P}_1\boldsymbol{E}^\mathsf{T} + \boldsymbol{E}\boldsymbol{P}_1\boldsymbol{A}^\mathsf{T} &= -\boldsymbol{B}\boldsymbol{B}^\mathsf{T},\\ \boldsymbol{A}\boldsymbol{P}_k\boldsymbol{E}^\mathsf{T} + \boldsymbol{E}\boldsymbol{P}_k\boldsymbol{A}^\mathsf{T} &= -\sum_{j=1}^{m}\boldsymbol{N}_j\boldsymbol{P}_{k-1}\boldsymbol{N}_j^\mathsf{T} \qquad k \geq 2.\end{aligned}$$

As pointed out in [24], even if all eigenvalues of $\boldsymbol{E}^{-1}\boldsymbol{A}$ have negative real part, it might happen that the integrals do not exist but the *Lyapunov* equation still yields a solution for $\boldsymbol{P}$. In this case, $\boldsymbol{P}$ does not correspond to the controllability *Gramian* and consequently the system is not controllable.

*Example* 3.2. [8] In order to enlighten the result above, we want to explicitly write the equations for the first two subsystems. To simplify things we assume that for $\boldsymbol{E} = \mathbf{I}$. Hence, the first *Gramian* is given by

$$\boldsymbol{P}_1 = \int_{\tau_1=0}^{\infty} \mathrm{e}^{\boldsymbol{A}\tau_1}\boldsymbol{B}\boldsymbol{B}^\mathsf{T}\mathrm{e}^{\boldsymbol{A}^\mathsf{T}\tau_1}\,\mathrm{d}\tau_1$$

and satisfies the linear *Lyapunov* equation

$$\boldsymbol{A}\boldsymbol{P}_1 + \boldsymbol{P}_1\boldsymbol{A}^\mathsf{T} = -\boldsymbol{B}\boldsymbol{B}^\mathsf{T}.$$

The controllability *Gramian* of the second subsystem

$$\begin{aligned}\boldsymbol{P}_2 &= \int_{\tau_1=0}^{\infty}\int_{\tau_2=0}^{\infty}\sum_{j=1}^{m}\mathrm{e}^{\boldsymbol{A}\tau_2}\boldsymbol{N}_j\mathrm{e}^{\boldsymbol{A}\tau_1}\boldsymbol{B}\boldsymbol{B}^\mathsf{T}\mathrm{e}^{\boldsymbol{A}^\mathsf{T}\tau_1}\boldsymbol{N}_j^\mathsf{T}\mathrm{e}^{\boldsymbol{A}^\mathsf{T}\tau_2}\,\mathrm{d}\tau_1\,\mathrm{d}\tau_2\\ &= \sum_{j=1}^{m}\int_{\tau_2=0}^{\infty}\mathrm{e}^{\boldsymbol{A}\tau_2}\boldsymbol{N}_j\underbrace{\int_{\tau_1=0}^{\infty}\mathrm{e}^{\boldsymbol{A}\tau_1}\boldsymbol{B}\boldsymbol{B}^\mathsf{T}\mathrm{e}^{\boldsymbol{A}^\mathsf{T}\tau_1}\,\mathrm{d}\tau_1}_{\boldsymbol{P}_1}\boldsymbol{N}_j^\mathsf{T}\mathrm{e}^{\boldsymbol{A}^\mathsf{T}\tau_2}\,\mathrm{d}\tau_2\end{aligned}$$

satisfies the linear *Lyapunov* equation

$$\boldsymbol{A}\boldsymbol{P}_2 + \boldsymbol{P}_2\boldsymbol{A}^\mathsf{T} = -\sum_{j=1}^{m}\boldsymbol{N}_j\boldsymbol{P}_1\boldsymbol{N}_j^\mathsf{T}.$$

△

**Observability**

Since derivations of observability are approached by a dual ansatz, the definition is quite similar to controllability. Hence, a linear system is called observable if the matrix $\boldsymbol{Q}_o = [\boldsymbol{C}^\mathsf{T}, \boldsymbol{A}^\mathsf{T}\boldsymbol{C}^\mathsf{T}, \cdots, (\boldsymbol{A}^\mathsf{T})^{n-1}\boldsymbol{C}^\mathsf{T}]^\mathsf{T}$ has full rank $n$. We are also able to define observability for bilinear systems. Consequently, a bilinear system is called observable if $\boldsymbol{Q}_n$ has full rank $n$. Hereby $\boldsymbol{Q}_1 = \boldsymbol{C}$ and $\boldsymbol{Q}_i = [\boldsymbol{A}^\mathsf{T}\boldsymbol{Q}_{i-1}^\mathsf{T},\ \boldsymbol{N}_1^\mathsf{T}\boldsymbol{Q}_{i-1}^\mathsf{T},\ \ldots,\ \boldsymbol{N}_m^\mathsf{T}\boldsymbol{Q}_{i-1}^\mathsf{T}]^\mathsf{T}$ where $i = 1, \cdots, n$ [11] [20].
Again we follow [14] to discuss the bilinear observability *Gramian*. According to the state-to-output transfer function we define

$$\begin{aligned}\boldsymbol{q}_1(t_1) &= \boldsymbol{C}\mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}t_1},\\ \boldsymbol{q}_k^{(j_2,\ldots,j_k)}(t_1,\cdots,t_k) &= \boldsymbol{C}\mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}t_k}\boldsymbol{E}^{-1}\boldsymbol{N}_{j_k}\cdots\mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}t_2}\boldsymbol{E}^{-1}\boldsymbol{N}_{j_2}\mathrm{e}^{\boldsymbol{E}^{-1}\boldsymbol{A}t_1}.\end{aligned}$$

This lets us define the observability *Gramian* as

$$\begin{aligned}\boldsymbol{Q} &= \sum_{k=1}^{\infty} \boldsymbol{Q}_k,\\ \boldsymbol{Q}_k &= \int_{\tau_1=0}^{\infty} \cdots \int_{\tau_k=0}^{\infty} \sum_{j_2=1}^{m} \cdots \sum_{j_k=1}^{m} \boldsymbol{q}_k^{(j_2,\ldots,j_k)}(\tau_1,\ldots,\tau_k)^{\mathsf{T}} \boldsymbol{q}_k^{(j_2,\ldots,j_k)}(\tau_1,\ldots,\tau_k)\,\mathrm{d}\tau_k \cdots \mathrm{d}\tau_1.\end{aligned}$$

Again, if $\boldsymbol{Q}$ exists, it is symmetric and positive semi-definite and satisfies following bilinear *Lyapunov* equation

$$\boldsymbol{A}^{\mathsf{T}}\boldsymbol{Q}\boldsymbol{E} + \boldsymbol{E}^{\mathsf{T}}\boldsymbol{Q}\boldsymbol{A} + \sum_{j=1}^{m} \boldsymbol{N}_j^{\mathsf{T}}\boldsymbol{Q}\boldsymbol{N}_j = -\boldsymbol{C}^{\mathsf{T}}\boldsymbol{C}.$$

Hereby, every $\boldsymbol{Q}_k$ is a solution of the linear *Lyapunov* equation

$$\begin{aligned}\boldsymbol{A}^{\mathsf{T}}\boldsymbol{Q}_1\boldsymbol{E} + \boldsymbol{E}^{\mathsf{T}}\boldsymbol{Q}_1\boldsymbol{A} &= -\boldsymbol{C}^{\mathsf{T}}\boldsymbol{C},\\ \boldsymbol{A}^{\mathsf{T}}\boldsymbol{Q}_k\boldsymbol{E} + \boldsymbol{E}^{\mathsf{T}}\boldsymbol{Q}_k\boldsymbol{A} &= -\sum_{j=1}^{m} \boldsymbol{N}_j^{\mathsf{T}}\boldsymbol{Q}_{k-1}\boldsymbol{N}_j \qquad k \geq 2.\end{aligned}$$

Similar to the controllability, even if all eigenvalues of $\boldsymbol{E}^{-1}\boldsymbol{A}$ have negative real part, it might appear that the integrals do not converge due to the bilinear character. In this case, the *Lyapunov* equation still yields a solution for $\boldsymbol{Q}$ which does not correspond to the observability *Gramian* $\boldsymbol{Q}$ and consequently the system is not observable [14].

*Example* 3.3. Comparing in the following $\boldsymbol{Q}_1$ and $\boldsymbol{Q}_2$ with $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$ one can even better determine the duality of the observability *Gramian* and the controllability *Gramian*. To again make things easier we assume that $\boldsymbol{E} = \mathbf{I}$. Following our definition we can write the first observability *Gramian* as

$$\boldsymbol{Q}_1 = \int_{\tau_1=0}^{\infty} \mathrm{e}^{\boldsymbol{A}^{\mathsf{T}}\tau_1}\boldsymbol{C}^{\mathsf{T}}\boldsymbol{C}\mathrm{e}^{\boldsymbol{A}\tau_1}\,\mathrm{d}\tau_1$$

which satisfies the linear *Lyapunov* equation

$$\boldsymbol{A}^{\mathsf{T}}\boldsymbol{Q}_1 + \boldsymbol{Q}_1\boldsymbol{A} = -\boldsymbol{C}^{\mathsf{T}}\boldsymbol{C}.$$

The observability Gramian of the second subsystem

$$\begin{aligned}\boldsymbol{Q}_2 &= \int_{\tau_1=0}^{\infty}\int_{\tau_2=0}^{\infty}\sum_{j=1}^{m} \mathrm{e}^{\boldsymbol{A}^{\mathsf{T}}\tau_2}\boldsymbol{N}_j^{\mathsf{T}}\mathrm{e}^{\boldsymbol{A}^{\mathsf{T}}\tau_1}\boldsymbol{C}^{\mathsf{T}}\boldsymbol{C}\mathrm{e}^{\boldsymbol{A}\tau_1}\boldsymbol{N}_j\mathrm{e}^{\boldsymbol{A}\tau_2}\,\mathrm{d}\tau_1\,\mathrm{d}\tau_2\\ &= \sum_{j=1}^{m}\int_{\tau_2=0}^{\infty} \mathrm{e}^{\boldsymbol{A}^{\mathsf{T}}\tau_2}\boldsymbol{N}_j^{\mathsf{T}}\int_{\tau_1=0}^{\infty} \mathrm{e}^{\boldsymbol{A}^{\mathsf{T}}\tau_1}\boldsymbol{C}^{\mathsf{T}}\boldsymbol{C}\mathrm{e}^{\boldsymbol{A}\tau_1}\,\mathrm{d}\tau_1\boldsymbol{N}_j\mathrm{e}^{\boldsymbol{A}\tau_2}\,\mathrm{d}\tau_2\end{aligned}$$

satisfies the linear *Lyapunov* equation

$$\boldsymbol{A}^{\mathsf{T}}\boldsymbol{Q}_2 + \boldsymbol{Q}_2\boldsymbol{A} = -\sum_{j=1}^{m}\boldsymbol{N}_j^{\mathsf{T}}\boldsymbol{Q}_1\boldsymbol{N}_j.$$

△

*Remark* 3.3 (Minimal Realization). A bilinear system is minimally realized if, and only if it is span controllable and observable [20]. △

### Solving Lyapunov Equations By Vectorization

Since we are interested in the solution of the preceding *Lyapunov* equations we somehow need a way to solve them. To be able to do this we need the definition of the vectorization operator vec$(\cdot)$ as well as the vectorization of a matrix product.

**Definition 3.1** (Vectorization)**.** Let $\boldsymbol{K} \in \mathbb{R}^{n\times n}$ be an arbitrary matrix. The vectorized form is then obtained by writing all columns underneath each other such that

$$\operatorname{vec}(\boldsymbol{K}) = \operatorname{vec}\left(\begin{bmatrix} k_{11} & \cdots & k_{1n} \\ \vdots & \ddots & \vdots \\ k_{n1} & \cdots & k_{nn} \end{bmatrix}\right) = \operatorname{vec}\left(\begin{bmatrix} \boldsymbol{k}_1, & \cdots, & \boldsymbol{k}_n \end{bmatrix}\right) = \begin{pmatrix} \boldsymbol{k}_1 \\ \vdots \\ \boldsymbol{k}_n \end{pmatrix} \quad \in \mathbb{R}^{n^2\times 1}.$$

▲

**Definition 3.2** (Vectorization Of A Matrix Product)**.** Let $\boldsymbol{K} \in \mathbb{R}^{m\times n}$, $\boldsymbol{L} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{M} \in \mathbb{R}^{n\times p}$ be arbitrary matrices multiplied like $\boldsymbol{KLM} \in \mathbb{R}^{m\times p}$. Vectorizing this product yields the following link

$$\operatorname{vec}(\boldsymbol{KLM}) = \left(\boldsymbol{M}^\mathsf{T} \otimes \boldsymbol{K}\right)\operatorname{vec}(\boldsymbol{L}) \quad \in \mathbb{R}^{mp\times 1}.$$

▲

This link makes it possible to solve the above *Lyapunov* equations. Vectorizing (3.12) yields

$$\operatorname{vec}\left(\boldsymbol{APE}^\mathsf{T}\right) + \operatorname{vec}\left(\boldsymbol{EPA}^\mathsf{T}\right) + \sum_{j=1}^{m} \operatorname{vec}\left(\boldsymbol{N}_j\boldsymbol{PN}_j^\mathsf{T}\right) = -\operatorname{vec}\left(\boldsymbol{BB}^\mathsf{T}\right).$$

With the above definition of a vectorized matrix product we obtain

$$(\boldsymbol{E}\otimes\boldsymbol{A})\operatorname{vec}(\boldsymbol{P}) + (\boldsymbol{A}\otimes\boldsymbol{E})\operatorname{vec}(\boldsymbol{P}) + \sum_{j=1}^{m}(\boldsymbol{N}_j\otimes\boldsymbol{N}_j)\operatorname{vec}(\boldsymbol{P}) = -\operatorname{vec}\left(\boldsymbol{BB}^\mathsf{T}\right).$$

Obviously we can exclude vec$(\boldsymbol{P})$ and solve the equation by inverting the left hand side such that

$$\operatorname{vec}(\boldsymbol{P}) = -\left(\boldsymbol{E}\otimes\boldsymbol{A} + \boldsymbol{A}\otimes\boldsymbol{E} + \sum_{j=1}^{m}\boldsymbol{N}_j\otimes\boldsymbol{N}_j\right)^{-1}\operatorname{vec}\left(\boldsymbol{BB}^\mathsf{T}\right)$$

where vec$(\boldsymbol{P}) \in \mathbb{R}^{n^2\times 1}$. Reshaping vec$(\boldsymbol{P})$ into a $n\times n$ matrix finally yields the solution $\boldsymbol{P}$. We can follow the exact same procedure and obtain the solution for $\boldsymbol{Q}$ by solving

$$\operatorname{vec}(\boldsymbol{Q}) = -\left(\boldsymbol{E}^\mathsf{T}\otimes\boldsymbol{A}^\mathsf{T} + \boldsymbol{A}^\mathsf{T}\otimes\boldsymbol{E}^\mathsf{T} + \sum_{j=1}^{m}\boldsymbol{N}_j^\mathsf{T}\otimes\boldsymbol{N}_j^\mathsf{T}\right)^{-1}\operatorname{vec}\left(\boldsymbol{C}^\mathsf{T}\boldsymbol{C}\right)$$

and reshaping vec$(\boldsymbol{Q})$ [11].

*Remark* 3.4 (Solving Large-Scale Bilinear Lyapunov Equations). [8] Obviously the direct method using the vectorization as shown above is only suitable for small-scale systems where the dimension $n$ of the system is small. Due to heavily increasing dimensions of the vectorized matrices ($n^2 \times n^2$) it is not possible to solve large-scale systems since commercial computers do not have enough storage. In a large-scale setting one should consider following options. As mentioned it is possible to gain the *Gramians* by solving linear *Lyapunov* equations and summing those solutions up. Assuming that the *Volterra* series converges, we can truncate the series and consider only the first $N$ subsystems. Hence, one could use existing routines (MATLAB® functions: `lyap()`, `lyapchol()`) for solving those first $N$ linear *Lyapunov* equations. The other option is to generalize low rank linear matrix equation solvers to be able to solve bilinear matrix equations which can be seen in [3]. △

## 3.4 Bilinear System Norms

The system norms are somehow the most important system theoretic concept for us, since they later give us the possibility to measure the difference between the output of two systems. We first discuss the $\mathcal{L}_2$-norm of a bilinear system. In this context we discover the link between the $\mathcal{L}_2$-norm and the *Gramians*. Following, we introduce the $\mathcal{H}_2$-norm for bilinear systems and its connection to the $\mathcal{L}_2$-norm. Finally, we show a possible procedure to compute the difference between the output of two systems without having to simulate them.

### 3.4.1 $\mathcal{L}_2$-Norm For Bilinear Systems

We can generalize the definition for the $\mathcal{L}_2$-norm by making use of the *Volterra* series. Therefore, we obtain the $\mathcal{L}_2$-norm for a MIMO bilinear system $\boldsymbol{\zeta}$ by evaluating the following

$$\|\boldsymbol{\zeta}\|_{\mathcal{L}_2}^2 = \sum_{k=1}^{\infty} \int_{\tau_1=0}^{\infty} \cdots \int_{\tau_k=0}^{\infty} \| \sum_{j_2=1}^{m} \cdots \sum_{j_k=1}^{m} \boldsymbol{g}_{k,\square}^{(j_2,\ldots,j_k)}(\tau_1,\ldots,\tau_k)\|_{\mathrm{F}}^2 \, \mathrm{d}\tau_1 \cdots \mathrm{d}\tau_k.$$

Hereby $\|\boldsymbol{g}_{k,\square}^{(j_2,\ldots,j_k)}(\tau_1,\ldots,\tau_k)\|_{\mathrm{F}}$ denotes the *Frobenius*-norm of the $k$-th order regular kernel. We can use the link between the *Frobenius*-norm and the trace of a matrix where $\|\boldsymbol{K}\|_{\mathrm{F}}^2 = \mathrm{tr}(\boldsymbol{K}\boldsymbol{K}^{\mathsf{T}}) = \mathrm{tr}(\boldsymbol{K}^{\mathsf{T}}\boldsymbol{K})$. Hence, we can also write the $\mathcal{L}_2$-norm of a bilinear system as follows

$$\begin{aligned}
&\|\boldsymbol{\zeta}\|_{\mathcal{L}_2}^2 = \\
&= \sum_{k=1}^{\infty} \int_{\tau_1=0}^{\infty} \cdots \int_{\tau_k=0}^{\infty} \mathrm{tr}\left( \sum_{j_2=1}^{m} \cdots \sum_{j_k=1}^{m} \boldsymbol{g}_{k,\square}^{(j_2,\ldots,j_k)}(\tau_1,\ldots,\tau_k) \boldsymbol{g}_{k,\square}^{(j_1,\ldots,j_k)}(\tau_1,\ldots,\tau_k)^{\mathsf{T}} \right) \mathrm{d}\tau_1 \cdots \mathrm{d}\tau_k \\
&= \sum_{k=1}^{\infty} \mathrm{tr}\left( \int_{\tau_1=0}^{\infty} \cdots \int_{\tau_k=0}^{\infty} \sum_{j_2=1}^{m} \cdots \sum_{j_k=1}^{m} \boldsymbol{g}_{k,\square}^{(j_2,\ldots,j_k)}(\tau_1,\ldots,\tau_k) \boldsymbol{g}_{k,\square}^{(j_1,\ldots,j_k)}(\tau_1,\ldots,\tau_k)^{\mathsf{T}} \, \mathrm{d}\tau_1 \cdots \mathrm{d}\tau_k \right).
\end{aligned}$$

Since we defined the controllability *Gramian* with the state to output transfer function we use that definition to write each $\boldsymbol{g}_{k,\square}^{(j_2,\ldots,j_k)}(\tau_1,\ldots,\tau_k)\boldsymbol{g}_{k,\square}^{(j_2,\ldots,j_k)}(\tau_1,\ldots,\tau_k)^{\mathsf{T}}$ as $\boldsymbol{C}\boldsymbol{P}_k\boldsymbol{C}$ which yields

$$
\begin{aligned}
&\|\zeta\|_{\mathcal{L}_2}^2 = \\
&= \sum_{k=1}^{\infty} \operatorname{tr}\left( \sum_{j_2=1}^{m} \cdots \sum_{j_k=1}^{m} \int_{\tau_1=0}^{\infty} \cdots \int_{\tau_k=0}^{\infty} \boldsymbol{g}_{k,\square}^{(j_2,\ldots,j_k)}(\tau_1,\ldots,\tau_k)^{\mathsf{T}} \boldsymbol{g}_{k,\square}^{(j_2,\ldots,j_k)}(\tau_1,\ldots,\tau_k)\,\mathrm{d}\tau_1\cdots\mathrm{d}\tau_k \right) \\
&= \operatorname{tr}\left( \boldsymbol{C} \underbrace{\int_{\tau_1=0}^{\infty} \mathrm{e}^{\boldsymbol{A}\tau_1}\boldsymbol{B}\boldsymbol{B}^{\mathsf{T}}\mathrm{e}^{\boldsymbol{A}^{\mathsf{T}}\tau_1}\,\mathrm{d}\tau_1}_{\boldsymbol{P}_1} \boldsymbol{C}^{\mathsf{T}} \right) \\
&\quad + \operatorname{tr}\left( \boldsymbol{C} \underbrace{\int_{\tau_1=0}^{\infty}\int_{\tau_2=0}^{\infty} \sum_{j_2=1}^{m} \mathrm{e}^{\boldsymbol{A}\tau_2}\boldsymbol{N}_{j_2}\mathrm{e}^{\boldsymbol{A}\tau_1}\boldsymbol{B}\boldsymbol{B}^{\mathsf{T}}\mathrm{e}^{\boldsymbol{A}^{\mathsf{T}}\tau_1}\boldsymbol{N}_{j_2}^{\mathsf{T}}\mathrm{e}^{\boldsymbol{A}^{\mathsf{T}}\tau_2}\,\mathrm{d}\tau_1\,\mathrm{d}\tau_2}_{\boldsymbol{P}_2} \boldsymbol{C}^{\mathsf{T}} \right) \\
&\quad + \ldots \\
&= \operatorname{tr}\left(\boldsymbol{C}\boldsymbol{P}\boldsymbol{C}^{\mathsf{T}}\right)
\end{aligned}
$$

Consequently we can compute the $\mathcal{L}_2$-norm with the controllability *Gramian*. In addition to that we can also compute the $\mathcal{L}_2$-norm by evaluating

$$\|\zeta\|_{\mathcal{L}_2}^2 = \operatorname{tr}\left(\boldsymbol{B}^{\mathsf{T}}\boldsymbol{Q}\boldsymbol{B}\right)$$

whose derivation is dual to above. Of course, computing the $\mathcal{L}_2$-norm with use of the *Gramians* implies that they exist [11].

*Example* 3.4 (Relation Between *Gramians* And Kernels). To provide better understanding for the equality between computing the $\mathcal{L}_2$-norm through the kernels and through the *Gramians* we take a closer look at the first subsystem. Again we assume that $\boldsymbol{E} = \mathbf{I}$. The first kernel which holds all inputs is defined by

$$\boldsymbol{g}_1(t) = \boldsymbol{C}\mathrm{e}^{\boldsymbol{A}t}\boldsymbol{B}.$$

Multiplying $\boldsymbol{g}_1(t_1)$ with its transposed yields

$$\boldsymbol{g}_1(t)\boldsymbol{g}_1(t)^{\mathsf{T}} = \boldsymbol{C}\mathrm{e}^{\boldsymbol{A}t}\boldsymbol{B}\boldsymbol{B}^{\mathsf{T}}\mathrm{e}^{\boldsymbol{A}^{\mathsf{T}}t}\boldsymbol{C}^{\mathsf{T}}.$$

As shown above we need to integrate the product of the kernel with its transposed to obtain the $\mathcal{L}_2$-norm. Hence, the $\mathcal{L}_2$-norm for the first subsystem $\boldsymbol{\Sigma}_1$ is defined by

$$\|\boldsymbol{\Sigma}_1\|_{\mathcal{L}_2}^2 = \operatorname{tr}\left(\int_{\tau=0}^{\infty} \boldsymbol{C}\mathrm{e}^{\boldsymbol{A}\tau}\boldsymbol{B}\boldsymbol{B}^{\mathsf{T}}\mathrm{e}^{\boldsymbol{A}^{\mathsf{T}}\tau}\boldsymbol{C}^{\mathsf{T}}\,\mathrm{d}\tau\right).$$

Comparing this with the definition of the controllability *Gramian* for the first subsystem

$$\boldsymbol{P}_1 = \int_{\tau=0}^{\infty} \mathrm{e}^{\boldsymbol{A}\tau}\boldsymbol{B}\boldsymbol{B}^{\mathsf{T}}\mathrm{e}^{\boldsymbol{A}^{\mathsf{T}}\tau}\,\mathrm{d}\tau$$

obviously reveals that we have to multiply $\boldsymbol{P}_1$ with $\boldsymbol{C}$ and its transposed to obtain the definition of the $\mathcal{L}_2$-norm through the transfer function

$$\operatorname{tr}\left(\boldsymbol{C}\boldsymbol{P}_1\boldsymbol{C}^{\mathsf{T}}\right) = \operatorname{tr}\left(\boldsymbol{C}\left(\int_{\tau=0}^{\infty} \mathrm{e}^{\boldsymbol{A}\tau}\boldsymbol{B}\boldsymbol{B}^{\mathsf{T}}\mathrm{e}^{\boldsymbol{A}^{\mathsf{T}}\tau}\,\mathrm{d}\tau\right)\boldsymbol{C}^{\mathsf{T}}\right) = \operatorname{tr}\left(\int_{\tau=0}^{\infty} \boldsymbol{C}\mathrm{e}^{\boldsymbol{A}\tau}\boldsymbol{B}\boldsymbol{B}^{\mathsf{T}}\mathrm{e}^{\boldsymbol{A}^{\mathsf{T}}\tau}\boldsymbol{C}^{\mathsf{T}}\,\mathrm{d}\tau\right).$$

△

### 3.4.2 $\mathcal{H}_2$-Norm For Bilinear Systems

Following [11] we could define the $\mathcal{H}_2$-norm of a bilinear system $\boldsymbol{\zeta}$ as

$$\|\boldsymbol{\zeta}\|_{\mathcal{H}_2}^2 = \sum_{k=1}^{\infty} \sup_{x_1>0,\ldots,x_k>0} \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \|\boldsymbol{G}_k^{\square}(x_1+\mathrm{i}\,y_1,\ldots,x_k+\mathrm{i}\,y_k)\|_{\mathrm{F}}^2\,\mathrm{d}y_1\cdots\mathrm{d}y_k. \quad (3.13)$$

Applying the link between the *Frobenius* norm and the trace of a matrix (3.13) and the *Phragmen-Lindelöf* principle reduces the $\mathcal{H}_2$-norm to

$$\|\boldsymbol{\zeta}\|_{\mathcal{H}_2}^2 = \sum_{k=1}^{\infty} \frac{1}{(2\pi)^k} \operatorname{tr}\Bigg(\sum_{j_2=1}^{m}\cdots\sum_{j_k=1}^{m}\int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty} \boldsymbol{G}_{k,\square}^{(j_2,\ldots,j_k)}(\mathrm{i}\,\omega_1,\ldots,\mathrm{i}\,\omega_k)$$
$$\times\, \boldsymbol{G}_{k,\square}^{(j_2,\ldots,j_k)}(-\mathrm{i}\,\omega_1,\ldots,-\mathrm{i}\,\omega_k)^{\mathsf{T}}\,\mathrm{d}\omega_1\cdots\mathrm{d}\omega_k\Bigg).$$

By applying *Plancherel's* theorem to $\|\boldsymbol{\zeta}\|_{\mathcal{L}_2}^2$ and then the *Phragmen-Lindelöf* principle yields the following equality [11, 8]

$$\|\boldsymbol{\zeta}\|_{\mathcal{L}_2[0,\infty)}^2 = \|\boldsymbol{\zeta}\|_{\mathcal{L}_2(\mathrm{i}\,\mathbb{R})}^2 = \|\boldsymbol{\zeta}\|_{\mathcal{H}_2}^2.$$

Hence, as long as the *Gramians* exists the $\mathcal{H}_2$-norm is equal to the $\mathcal{L}_2$-norm and we are able to compute the $\mathcal{H}_2$-norm as follows

$$\|\boldsymbol{\zeta}\|_{\mathcal{H}_2}^2 = \operatorname{tr}\left(\boldsymbol{C}\boldsymbol{P}\boldsymbol{C}^{\mathsf{T}}\right) = \operatorname{tr}\left(\boldsymbol{B}^{\mathsf{T}}\boldsymbol{Q}\boldsymbol{B}\right).$$

### 3.4.3 Difference Between Two Bilinear Systems

As pointed out previously we want to determine the difference $\|\boldsymbol{y}(t)-\tilde{\boldsymbol{y}}(t)\|_{\mathcal{H}_2}^2$ between the outputs of two bilinear systems $\boldsymbol{\zeta}$ and $\tilde{\boldsymbol{\zeta}}$. At the same time we want to avoid the simulation of the two systems, as this can be very time consuming. Therefore, we make use of the so-called error-system $\boldsymbol{\zeta}_{\mathrm{err}}$ to measure the error $E$ as follows

$$E = \|\boldsymbol{\zeta}_{\mathrm{err}}\|_{\mathcal{H}_2}^2 = \|\boldsymbol{\zeta}-\tilde{\boldsymbol{\zeta}}\|_{\mathcal{H}_2}^2.$$

Our goal is that the output of the error-system yields the difference between the output of $\boldsymbol{\zeta}$ and the output of $\tilde{\boldsymbol{\zeta}}$. Therefore we can define

$$\boldsymbol{y}_{\mathrm{err}}(t) = \underbrace{\begin{bmatrix}\boldsymbol{C} & -\tilde{\boldsymbol{C}}\end{bmatrix}}_{:=\boldsymbol{C}_{\mathrm{err}}} \underbrace{\begin{bmatrix}\boldsymbol{x}(t)\\ \tilde{\boldsymbol{x}}(t)\end{bmatrix}}_{:=\boldsymbol{x}_{\mathrm{err}}(t)}.$$

Hereby $\boldsymbol{x}_{\text{err}}(t)$ can be obtained by solving

$$\boldsymbol{E}_{\text{err}}\,\dot{\boldsymbol{x}}_{\text{err}}(t) = \boldsymbol{A}_{\text{err}}\,\boldsymbol{x}_{\text{err}}(t) + \sum_{j=1}^{m} \boldsymbol{N}_{\text{err},j}\,\boldsymbol{x}_{\text{err}}(t)\,u_j(t) + \boldsymbol{B}_{\text{err}}\,\boldsymbol{u}(t)$$

where

$$\boldsymbol{E}_{\text{err}} = \begin{bmatrix} \boldsymbol{E} & \boldsymbol{0} \\ \boldsymbol{0} & \tilde{\boldsymbol{E}} \end{bmatrix}, \quad \boldsymbol{A}_{\text{err}} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{0} \\ \boldsymbol{0} & \tilde{\boldsymbol{A}} \end{bmatrix}, \qquad \boldsymbol{N}_{\text{err},j} = \begin{bmatrix} \boldsymbol{N}_j & \boldsymbol{0} \\ \boldsymbol{0} & \tilde{\boldsymbol{N}}_j \end{bmatrix}, \qquad \boldsymbol{B}_{\text{err}} = \begin{bmatrix} \boldsymbol{B} \\ \tilde{\boldsymbol{B}} \end{bmatrix}.$$

Obviously it is only possible to compute the error-system for two systems with an equal number of inputs as well as an equal number of outputs. By evaluating the error-system's $\mathcal{H}_2$-norm as for an usual bilinear system we obtain the difference between the outputs of both systems. Assuming the *Gramians* exist, we can write the error as

$$E = \|\boldsymbol{\zeta}_{\text{err}}\|_{\mathcal{H}_2}^2 = \operatorname{tr}\left(\boldsymbol{C}_{\text{err}}\boldsymbol{P}_{\text{err}}\boldsymbol{C}_{\text{err}}^{\mathsf{T}}\right) = \operatorname{tr}\left(\boldsymbol{B}_{\text{err}}^{\mathsf{T}}\boldsymbol{Q}_{\text{err}}\boldsymbol{B}_{\text{err}}\right)$$

where $\boldsymbol{P}_{\text{err}}$ and $\boldsymbol{Q}_{\text{err}}$ are obtained by solving the *Lyapunov* equations for the error-systems [2] [24].

# Chapter 4

# Model Reduction Of Bilinear Systems

We now draw our attention towards model reduction. As mentioned previously, one could derive large-scale bilinear systems through the application of finite elements or by applying the *Carleman* bilinearization to a nonlinear system. Since a large-scale system, called full order model (FOM), is not efficiently solvable on a commercial computer, we try to reduce it to obtain an approximation, the so-called reduced order model (ROM). The main goals of reduction are:

- Reducing the error $E$ between the original output and the output of the reduced system such that

  $$\boldsymbol{y}_{\mathrm{r}}(t) \approx \boldsymbol{y}(t)$$

  and consequently approximating the FOM through the ROM as good as possible.

- Maintaining the system properties such as stability, structure, controllability/observability and more.

- Making the computation of the ROM numerically efficient, robust and stable.

The task modelreduction could be approached by several methods where none of them could perfectly achieve the above goals. The two main approaches are balanced-based model reduction and interpolation-based model reduction. Both are projective methods, which means that the state-vector gets reduced in size and approximated via a projection. Within this thesis we focus on interpolation-based methods. Therefore, we start by discussing the fundamentals like projection, *Krylov* subspaces, moments and moment matching. Following, we briefly introduce the subsystem interpolation framework and point out its huge disadvantage, the combinatoric problem. After that we focus on the *Volterra* Series interpolation framework, which we extend to match high-order moments as well as to support complex expansion points and *Markov Parameters*. Finally, we review a method to obtain $\mathcal{H}_2$-optimal ROMs. In this sense we present BIRKA.

## 4.1 Fundamentals

To provide better understanding of the following model reduction frameworks, let us begin by explaining the basic concepts. Since all presented methods yield the ROM by projection, we first discuss what a projection is and how to apply it to a bilinear system. After that, we continue with *Krylov* subspaces and finally introduce moments and moment matching.

### Projection

Generally, a projection is a linear transformation from a vector space $\mathcal{V}$ to itself or to a subspace of $\mathcal{V}$. This happens with use of a projector.

**Definition 4.1** (Projector)**.** A matrix $\boldsymbol{P} \in \mathbb{R}^{n\times n}$ is called projector onto the subspace $\mathcal{V} \subset \mathbb{R}^n$ if

$$\boldsymbol{P} = \boldsymbol{P}^2$$

and $\operatorname{im}(\boldsymbol{P}) = \mathcal{V}$, where $\operatorname{im}(\cdot)$ denotes the image of a matrix. ▲

Hence, applying the same projector twice yields the same result. If $\boldsymbol{P} = \boldsymbol{P}^\mathsf{T}$ holds then the projector is called orthogonal, otherwise oblique. A projector $\boldsymbol{P} = \boldsymbol{V}\boldsymbol{V}^\mathsf{T}$ is also orthogonal if $\boldsymbol{V} = \begin{bmatrix}\boldsymbol{v}_1, & \dots, & \boldsymbol{v}_r\end{bmatrix}$ is an orthonormal basis of $\mathcal{V}$. A projector $\boldsymbol{P} = \boldsymbol{V}\left(\boldsymbol{W}^\mathsf{T}\boldsymbol{V}\right)^{-1}\boldsymbol{W}^\mathsf{T}$ is oblique if $\boldsymbol{W} = \begin{bmatrix}\boldsymbol{w}_1, & \dots, & \boldsymbol{w}_r\end{bmatrix}$ is a basis for the subspace $\mathcal{W}$ with the same dimensions as $\mathcal{V}$.
Let us assume that we approximate the state vector $\boldsymbol{x}(t) \in \mathbb{R}^{n\times 1}$ of

$$\boldsymbol{\zeta} : \begin{cases} \boldsymbol{E}\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \sum\limits_{j=1}^{m} \boldsymbol{N}_j\boldsymbol{x}(t)u_j(t) + \boldsymbol{B}\boldsymbol{u}(t), \quad \boldsymbol{x}(0) = \boldsymbol{x}_0, \\ \boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t), \end{cases} \tag{4.1}$$

with a reduced state vector $\boldsymbol{x}_\mathrm{r}(t) \in \mathbb{R}^{r\times 1}$ where $r \ll n$. Hence, we write

$$\boldsymbol{x}(t) = \boldsymbol{V}\boldsymbol{x}_\mathrm{r}(t) + \boldsymbol{e}(t). \tag{4.2}$$

Hereby $\boldsymbol{e}(t)$ is the error we make in the approximation. Substituting the approximation for $\boldsymbol{x}(t)$ into (4.1) yields

$$\begin{aligned} \boldsymbol{E}\boldsymbol{V}\dot{\boldsymbol{x}}_\mathrm{r}(t) &= \boldsymbol{A}\boldsymbol{V}\boldsymbol{x}_\mathrm{r}(t) + \sum_{j=1}^{m} \boldsymbol{N}_j\boldsymbol{V}\boldsymbol{x}_\mathrm{r}(t)u_j(t) + \boldsymbol{B}\boldsymbol{u}(t) + \boldsymbol{r}(t), \\ \boldsymbol{y}(t) &= \boldsymbol{C}\boldsymbol{V}\boldsymbol{x}_\mathrm{r}(t), \end{aligned} \tag{4.3}$$

where the residual $\boldsymbol{r}(t) = \boldsymbol{A}\boldsymbol{e}(t) + \sum\limits_{j=1}^{m} \boldsymbol{N}_j\boldsymbol{e}(t)u_j(t) - \boldsymbol{E}\dot{\boldsymbol{e}}(t)$ holds the errors. Since the above representation is overdetermined ($n$ equations for $r$ variables) we project the whole equation onto the subspace $\operatorname{span}(\boldsymbol{E}\boldsymbol{V})$. Therefore, we use $\boldsymbol{W} \in \mathbb{R}^{n\times r}$ and assume that $\left(\boldsymbol{W}^\mathsf{T}\boldsymbol{E}\boldsymbol{V}\right)^{-1}$ exists. With that expression we are able to formulate the projector

$$\boldsymbol{P} = \boldsymbol{E}\boldsymbol{V}\left(\boldsymbol{W}^\mathsf{T}\boldsymbol{E}\boldsymbol{V}\right)^{-1}\boldsymbol{W}^\mathsf{T}.$$

Consequently we call $\boldsymbol{V}$ and $\boldsymbol{W}$ projection matrices. Applying the projector to the state-equation of (4.3) yields

$$\boldsymbol{PEV}\dot{\boldsymbol{x}}_\mathrm{r}(t) = \boldsymbol{PAV}\boldsymbol{x}_\mathrm{r}(t) + \sum_{j=1}^{m} \boldsymbol{PN}_j\boldsymbol{V}\boldsymbol{x}_\mathrm{r}(t)u_j(t) + \boldsymbol{PBu}(t) + \boldsymbol{Pr}(t). \tag{4.4}$$

With the *Petrov-Galerkin* condition ($\boldsymbol{r}(t) \perp \boldsymbol{W}$) it follows that

$$\boldsymbol{W}^\mathsf{T}\boldsymbol{r}(t) = \boldsymbol{0}.$$

Keeping the *Petrov-Galerkin* condition in mind we factor $\boldsymbol{EV}\left(\boldsymbol{W}^\mathsf{T}\boldsymbol{EV}\right)^{-1}$ in (4.4) out, which yields

$$\boldsymbol{0} = \boldsymbol{EV}\left(\boldsymbol{W}^\mathsf{T}\boldsymbol{EV}\right)^{-1}\left(\boldsymbol{W}^\mathsf{T}\boldsymbol{AV}\boldsymbol{x}_\mathrm{r}(t) + \sum_{j=1}^{m}\boldsymbol{W}^\mathsf{T}\boldsymbol{N}_j\boldsymbol{V}\boldsymbol{x}_\mathrm{r}(t)u_j(t) + \boldsymbol{W}^\mathsf{T}\boldsymbol{Bu}(t) - \boldsymbol{W}^\mathsf{T}\boldsymbol{EV}\dot{\boldsymbol{x}}_\mathrm{r}(t)\right).$$

Assuming that $\boldsymbol{EV}\left(\boldsymbol{W}^\mathsf{T}\boldsymbol{EV}\right)^{-1}$ is not a zero matrix, the above could only be fulfilled if the terms in brackets yield a zero matrix. Consequently, we write the above as follows

$$\boldsymbol{W}^\mathsf{T}\boldsymbol{EV}\dot{\boldsymbol{x}}_\mathrm{r}(t) = \boldsymbol{W}^\mathsf{T}\boldsymbol{AV}\boldsymbol{x}_\mathrm{r}(t) + \sum_{j=1}^{m}\boldsymbol{W}^\mathsf{T}\boldsymbol{N}_j\boldsymbol{V}\boldsymbol{x}_\mathrm{r}(t)u_j(t) + \boldsymbol{W}^\mathsf{T}\boldsymbol{Bu}(t).$$

Hence, we obtain the reduced bilinear system

$$\boldsymbol{\zeta}_\mathrm{r} : \begin{cases} \boldsymbol{E}_\mathrm{r}\dot{\boldsymbol{x}}_\mathrm{r}(t) = \boldsymbol{A}_\mathrm{r}\boldsymbol{x}_\mathrm{r}(t) + \sum_{j=1}^{m}\boldsymbol{N}_{\mathrm{r},j}\boldsymbol{x}_\mathrm{r}(t)u_j(t) + \boldsymbol{B}_\mathrm{r}\boldsymbol{u}(t), \quad \boldsymbol{x}_\mathrm{r}(0) = \left(\boldsymbol{W}^\mathsf{T}\boldsymbol{EV}\right)^{-1}\boldsymbol{W}^\mathsf{T}\boldsymbol{E}\boldsymbol{x}_0, \\ \boldsymbol{y}_\mathrm{r}(t) = \boldsymbol{C}_\mathrm{r}\boldsymbol{x}_\mathrm{r}(t) \end{cases}$$

where $\boldsymbol{E}_\mathrm{r} = \boldsymbol{W}^\mathsf{T}\boldsymbol{EV}$, $\boldsymbol{A}_\mathrm{r} = \boldsymbol{W}^\mathsf{T}\boldsymbol{AV}$, $\boldsymbol{N}_{\mathrm{r},j} = \boldsymbol{W}^\mathsf{T}\boldsymbol{N}_j\boldsymbol{V}$, $\boldsymbol{B}_\mathrm{r} = \boldsymbol{W}^\mathsf{T}\boldsymbol{B}$, $\boldsymbol{C}_\mathrm{r} = \boldsymbol{CV}$. Note that the number of inputs and outputs does not change but $\boldsymbol{y}_\mathrm{r}(t)$ is an approximation of $\boldsymbol{y}(t)$ [5] [18] [23].

### Krylov Subspaces

*Krylov* subspaces or rather *Krylov* subspace methods were originally developed to solve eigenvalue problems and linear systems of equation. In our case we try to approximate the state equation with *Krylov* subspace model reduction methods which means that we try to find projection matrices $\boldsymbol{V}$ and $\boldsymbol{W}$ which generate equality between the FOM transfer function and the ROM transfer function at certain interpolation points.

**Definition 4.2** (Krylov Subspace)**.** The $q$-th *Krylov* subspace $\mathcal{K}_q$ of a matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ and a vector $\boldsymbol{v} \in \mathbb{R}^{n\times 1}$ is defined by

$$\mathcal{K}_q\left(\boldsymbol{A}, \boldsymbol{v}\right) = \mathrm{span}\{\boldsymbol{v},\, \boldsymbol{Av},\, \boldsymbol{A}^2\boldsymbol{v},\, \dots,\, \boldsymbol{A}^{q-1}\boldsymbol{v}\} = \mathrm{span}(\boldsymbol{V}).$$

Consequently $\mathcal{K}_q$ spans a subspace of $\mathbb{R}^n$ with an infinite amount of bases $\boldsymbol{V} \in \mathbb{R}^{n\times q}$. ▲

In simpler words one could describe $\mathcal{K}_q$ as a subspace constructed by a matrix and a vector. Since it is possible to use a matrix instead of a vector we also provide the definition of the block *Krylov* subspace.

**Definition 4.3** (Block Krylov Subspace)**.** The $q$-th block *Krylov* subspace $\mathcal{K}_q$ of a matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ and a matrix $\boldsymbol{B} = \begin{bmatrix}\boldsymbol{b}_1, & \dots, & \boldsymbol{b}_m\end{bmatrix} \in \mathbb{R}^{n\times m}$ is defined by

$$\mathcal{K}_q\left(\boldsymbol{A},\boldsymbol{B}\right) = \operatorname{span}\{\boldsymbol{b}_1, \dots, \boldsymbol{b}_m,\ \boldsymbol{A}\boldsymbol{b}_1, \dots, \boldsymbol{A}\boldsymbol{b}_m,\ \dots,\ \boldsymbol{A}^{q-1}\boldsymbol{b}_1, \dots, \boldsymbol{A}^{q-1}\boldsymbol{b}_m\} = \operatorname{span}(\boldsymbol{V}).$$

Consequently $\mathcal{K}_q$ spans a subspace of $\mathbb{R}^n$ with an infinite amount of bases $\boldsymbol{V} \in \mathbb{R}^{n\times qm}$.
▲

Note that the bases this time are in $\mathbb{R}^{n\times qm}$ [19] [6].
Very important for us is the input *Krylov* subspace, defined by

$$\mathcal{K}_q\left(\tilde{\boldsymbol{A}},\tilde{\boldsymbol{B}}\right)$$

where

$$\tilde{\boldsymbol{A}} = \sum_{j_2=1}^{m}\cdots\sum_{j_k=1}^{m}\left(s_k\boldsymbol{E}-\boldsymbol{A}\right)^{-1}\boldsymbol{N}_{j_k}\cdots\boldsymbol{N}_{j_2}\left(s_1\boldsymbol{E}-\boldsymbol{A}\right)^{-1}$$
$$\tilde{\boldsymbol{B}} = \sum_{j_2=1}^{m}\cdots\sum_{j_k=1}^{m}\left(s_k\boldsymbol{E}-\boldsymbol{A}\right)^{-1}\boldsymbol{N}_{j_k}\cdots\boldsymbol{N}_{j_2}\left(s_1\boldsymbol{E}-\boldsymbol{A}\right)^{-1}\boldsymbol{B}$$

and the output *Krylov* subspace defined by

$$\mathcal{K}_q\left(\tilde{\boldsymbol{A}}^{\mathsf{T}},\tilde{\boldsymbol{C}}^{\mathsf{T}}\right)$$

where

$$\tilde{\boldsymbol{A}}^{\mathsf{T}} = \sum_{j_2=1}^{m}\cdots\sum_{j_k=1}^{m}\left(s_1\boldsymbol{E}-\boldsymbol{A}\right)^{-\mathsf{T}}\boldsymbol{N}_{j_2}^{\mathsf{T}}\cdots\boldsymbol{N}_{j_k}^{\mathsf{T}}\left(s_k\boldsymbol{E}-\boldsymbol{A}\right)^{-\mathsf{T}}$$
$$\tilde{\boldsymbol{C}}^{\mathsf{T}} = \sum_{j_2=1}^{m}\cdots\sum_{j_k=1}^{m}\left(s_1\boldsymbol{E}-\boldsymbol{A}\right)^{-\mathsf{T}}\boldsymbol{N}_{j_2}^{\mathsf{T}}\cdots\boldsymbol{N}_{j_k}^{\mathsf{T}}\left(s_k\boldsymbol{E}-\boldsymbol{A}\right)^{-\mathsf{T}}\boldsymbol{C}^{\mathsf{T}}.$$

Obviously, $\hat{\boldsymbol{B}}$ is the input-to-state transfer function and $\hat{\boldsymbol{C}}^{\mathsf{T}}$ is the transposed of the state-to-output transfer function.

### Moments

Starting from the linear case we can define a moment with a *Taylor* series expansion of the *Laplace* transformed impulse response $\boldsymbol{G}(s) = \boldsymbol{C}(s\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{B}$ at a complex expansion point $\sigma$, also called shift, such that

$$\boldsymbol{G}(s) = \sum_{i=0}^{\infty}\boldsymbol{M}_{(i)}(\sigma)(s-\sigma)^i.$$

Hereby, $\boldsymbol{M}_{(i)}(\sigma)$ describes the $i$-th derivative of $\boldsymbol{G}(s)$ evaluated at $\sigma$ and is also known as the $i$-th moment of $\boldsymbol{G}(s)$. Hence, the $i$-th moment corresponding to the expansion point $\sigma$ is is given by

$$\boldsymbol{M}_{(i)}(\sigma) = (-1)^i\,\boldsymbol{C}\left[(\sigma\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E}\right]^i(\sigma\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{B}$$

where the 0-th moment is equal to $\boldsymbol{G}(\sigma)$ [17] [23].
Since the $k$-th transfer function of a bilinear system depends on $k$ variables one has to make use of the *Neumann* expansion to write the $k$-th order transfer function

$$\boldsymbol{G}_{k,\square}^{(j_2,\ldots,j_k)}(s_1,\ldots,s_k) = \boldsymbol{C}(s_k\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{N}_{j_k}\cdots\boldsymbol{N}_{j_2}(s_1\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{B}$$

as a *Taylor* series [6] [15]. In addition to that we now expand every variable $s_k$ at a certain shift $\sigma_k$ which results in $k$ different shifts. This leads to

$$\boldsymbol{G}_{k,\square}^{(j_2,\ldots,j_k)}(s_1,\ldots,s_k) = \sum_{i_1=0}^{\infty}\cdots\sum_{i_k=0}^{\infty}\boldsymbol{M}_{k,(i_1,\cdots,i_k)}^{(j_2,\ldots,j_k)}(\sigma_1,\cdots,\sigma_k)\,(s_1-\sigma_1)^{i_1}\cdots(s_k-\sigma_k)^{i_k}$$

where the so-called multi-moments $\boldsymbol{M}_{k,(i_1,\cdots,i_k)}^{(j_2,\ldots,j_k)}(\sigma_1,\cdots,\sigma_k)$ of a bilinear system are given by

$$\begin{aligned}\boldsymbol{M}_{k,(i_1,\cdots,i_k)}^{(j_2,\ldots,j_k)}(\sigma_1,\cdots,\sigma_k) = (-1)^{i_1+\cdots+i_k}\,\boldsymbol{C}\left[(\sigma_1\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E}\right]^{i_1}(\sigma_1\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{N}_{j_k}\cdots\\ \times\boldsymbol{N}_{j_2}\left[(\sigma_k\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E}\right]^{i_k}(\sigma_k\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{B}.\end{aligned}$$

It is also possible to choose infinity as an expansion point. In this case the moments are called *Markov Parameters*, which are defined differently compared to moments expanded at finite expansion points. Expanding a transfer function of a linear system at $t \to 0$ in time domain is equal to expanding the frequency domained transfer function at $s \to \infty$ which yields the definition

$$\boldsymbol{M}_{\infty,(i)} = \boldsymbol{C}\left[(\boldsymbol{E}^{-1}\boldsymbol{A})^{-1}\right]^{i}\boldsymbol{E}^{-1}\boldsymbol{B}.$$

Similar to that, one could expand the $k$-variable transfer function of a bilinear systems in time domain at $t_1 \to 0$, …, $t_k \to 0$ and transform this into frequency domain. Consequently, the multi-moment *Markov Parameters* $\boldsymbol{M}_{k,\infty,(i_1,\cdots,i_k)}^{(j_2,\ldots,j_k)}$ are defined as follows

$$\boldsymbol{M}_{k,\infty,(i_1,\cdots,i_k)}^{(j_2,\ldots,j_k)} = \boldsymbol{C}\left[(\boldsymbol{E}^{-1}\boldsymbol{A})^{-1}\right]^{i_1}\boldsymbol{E}^{-1}\boldsymbol{N}_{j_k}\cdots\boldsymbol{E}^{-1}\boldsymbol{N}_{j_2}\left[(\boldsymbol{E}^{-1}\boldsymbol{A})^{-1}\right]^{i_k}\boldsymbol{E}^{-1}\boldsymbol{B}.$$

## Moment Matching

In the linear case the idea of moment matching is to ensure equality of the first $q$ moments at a certain shift $\sigma$ between the FOM and the ROM such that

$$\boldsymbol{M}_{(i)}(\sigma) \equiv \boldsymbol{M}_{\mathrm{r},(i)}(\sigma)$$

where $\boldsymbol{M}_{\mathrm{r},(i)}(\sigma)$ is the $i$-th moment of the ROM corresponding to $\sigma$ and $i = 1,\ldots,q$. Applying this to bilinear systems means ensuring equality between the first $q^k$ multi-moments of the FOM and the ROM at a certain shift combination $\sigma_1,\ldots,\sigma_k$ for all variables $s_1,\ldots,s_k$ such that

$$\boldsymbol{M}_{k,(i_1,\cdots,i_k)}^{(j_2,\ldots,j_k)}(\sigma_1,\cdots,\sigma_k) \equiv \boldsymbol{M}_{\mathrm{r},k,(i_1,\cdots,i_k)}^{(j_2,\ldots,j_k)}(\sigma_1,\cdots,\sigma_k)$$

where $\boldsymbol{M}_{\mathrm{r},k,(i_1,\cdots,i_k)}^{(j_2,\ldots,j_k)}(\sigma_1,\cdots,\sigma_k)$ is the $i_1,\ldots,i_k$-th multi-moment corresponding to the shift combination $\sigma_1,\ldots,\sigma_k$ and $i_1 = 1,\ldots,q \cdots i_k = 1,\ldots,q$. Of course, one could also match the FOM and the ROM at different expansion points.
As we will see it is possible to fulfill moment matching conditions by defining the projection matrices $\boldsymbol{V}$, $\boldsymbol{W}$ as input or output *Krylov* subspaces. This feature finds application in subsystem interpolation as well as in the *Volterra* series interpolation. Therefore, we call both methods *Krylov* subspace model reduction methods.

## 4.2 Subsystem Interpolation

Subsystem interpolation is a wide spread interpolation method for bilinear systems. It was first introduced by [18] for *Markov Parameters*. The most generalized definition can be found in [15] where MIMO systems as well as high-order moments are considered. We do not want to explain subsystem interpolation in every detail since our main focus lies on the *Volterra* series interpolation. Hence, we only introduce the multi-point SISO and MIMO case without considering high-order moments. Finally, we explain the combinatoric problem which appears within the subsystem interpolation framework.

### 4.2.1 SISO Subsystem Interpolation

Let us begin by explaining the basic concept of subsystem interpolation for a SISO bilinear system. The main goal is to match the 0-th multi-moments for the first $k$ subsystems at certain shift combinations $\sigma_i$ as follows

$$\begin{aligned} G_1(\sigma_{l_1}) &\equiv G_{1,\mathrm{r}}(\sigma_{l_1}), && l_1 = 1,\dots,r \\ G_2^{\square}(\sigma_{l_1},\sigma_{l_2}) &\equiv G_{2,\mathrm{r}}^{\square}(\sigma_{l_1},\sigma_{l_2}), && l_1,l_2 = 1,\dots,r \\ &\vdots \\ G_k^{\square}(\sigma_{l_1},\dots,\sigma_{l_k}) &\equiv G_{k,\mathrm{r}}^{\square}(\sigma_{l_1},\dots,\sigma_{l_k}), && l_1,\cdots,l_k = 1,\dots,r. \end{aligned} \tag{4.5}$$

Hereby, the number of different interpolation points as well as the number of considered subsystems define the order of the ROM. Note, that we write the moment matching conditions with transfer functions since the 0-th moment is equal to the transfer function. To be able to determine the difference between input and output subsystem interpolation we write the moment matching conditions with different expansion points $\mu$

$$\begin{aligned} G_1(\mu_{l_1}) &\equiv G_{1,\mathrm{r}}(\mu_{l_1}), && l_1 = 1,\dots,r \\ G_2^{\square}(\mu_{l_2},\mu_{l_1}) &\equiv G_{2,\mathrm{r}}^{\square}(\mu_{l_2},\mu_{l_1}), && l_1,l_2 = 1,\dots,r \\ &\vdots \\ G_k^{\square}(\mu_{l_k},\dots,\mu_{l_1}) &\equiv G_{k,\mathrm{r}}^{\square}(\mu_{l_k},\dots,\mu_{l_1}), && l_1,\cdots,l_k = 1,\dots,r. \end{aligned} \tag{4.6}$$

As mentioned previously, moment matching can be achieved by choosing $\boldsymbol{V}$ and $\boldsymbol{W}$ as an input and output *Krylov* subspace. Hence, we choose the projection matrices $\boldsymbol{V}^{(k)}$ for each subsystem $k$ as follows

$$\begin{aligned} \boldsymbol{V}^{(1)} &\subseteq \mathcal{K}_r\left((\sigma_{l_1}\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E}, (\sigma_{l_1}\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{b}\right) \in \mathbb{R}^{n\times r} \\ \boldsymbol{V}^{(2)} &\subseteq \mathcal{K}_r\left((\sigma_{l_2}\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E}, (\sigma_{l_2}\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{N}\boldsymbol{V}^{(1)}\right) \in \mathbb{R}^{n\times r^2} \\ &\vdots \\ \boldsymbol{V}^{(k)} &\subseteq \mathcal{K}_r\left((\sigma_{l_k}\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E}, (\sigma_{l_k}\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{N}\boldsymbol{V}^{(k-1)}\right) \in \mathbb{R}^{n\times r^k},\ k = 3,\dots,N \end{aligned}$$

and the projection matrices $\boldsymbol{W}^{(k)}$ for each subsystem $k$ are defined by

$$\begin{aligned}
\boldsymbol{W}^{(1)} &\subseteq \mathcal{K}_r\left((\mu_{l_1}\boldsymbol{E}-\boldsymbol{A})^{-\mathsf{T}}\boldsymbol{E}^{\mathsf{T}}, (\mu_{l_1}\boldsymbol{E}-\boldsymbol{A})^{-\mathsf{T}}\boldsymbol{c}\right) \in \mathbb{R}^{n\times r} \\
\boldsymbol{W}^{(2)} &\subseteq \mathcal{K}_r\left((\mu_{l_2}\boldsymbol{E}-\boldsymbol{A})^{-\mathsf{T}}\boldsymbol{E}^{\mathsf{T}}, (\mu_{l_2}\boldsymbol{E}-\boldsymbol{A})^{-\mathsf{T}}\boldsymbol{N}^{\mathsf{T}}\boldsymbol{W}^{(1)}\right) \in \mathbb{R}^{n\times r^2} \\
&\vdots \\
\boldsymbol{W}^{(k)} &\subseteq \mathcal{K}_r\left((\mu_{l_k}\boldsymbol{E}-\boldsymbol{A})^{-\mathsf{T}}\boldsymbol{E}^{\mathsf{T}}, (\mu_{l_k}\boldsymbol{E}-\boldsymbol{A})^{-\mathsf{T}}\boldsymbol{N}^{\mathsf{T}}\boldsymbol{W}^{(k-1)}\right) \in \mathbb{R}^{n\times r^k}, \ k = 3,\dots,N.
\end{aligned}$$

Finally, we can obtain the matrices $\boldsymbol{V}$ and $\boldsymbol{W}$ by concatenating the matrices of each subsystem

$$\begin{aligned}
\boldsymbol{V} &= \left[\boldsymbol{V}^{(1)}, \ \dots, \ \boldsymbol{V}^{(k)}\right] \in \mathbb{R}^{n\times r_{\text{tot}}} \\
\boldsymbol{W} &= \left[\boldsymbol{W}^{(1)}, \ \dots, \ \boldsymbol{W}^{(k)}\right] \in \mathbb{R}^{n\times r_{\text{tot}}}
\end{aligned} \tag{4.7}$$

where $r_{\text{tot}} = \sum_{k=1}^{N} r^k$. Note, that every $\boldsymbol{V}^{(k)}$ depends on the previous projection matrix $\boldsymbol{V}^{(k-1)}$ and every $\boldsymbol{W}^{(k)}$ as well on the matrix $\boldsymbol{W}^{(k-1)}$. This yields a greater reduced order with every additionally considered subsystem. It is possible to compute $\boldsymbol{V}$ with different expansion points compared to $\boldsymbol{W}$.
By reducing the system matrices of the FOM as shown in the fundamentals ($\boldsymbol{E}_{\text{r}} = \boldsymbol{W}^{\mathsf{T}}\boldsymbol{E}\boldsymbol{V}$, $\boldsymbol{A}_{\text{r}} = \boldsymbol{W}^{\mathsf{T}}\boldsymbol{A}\boldsymbol{V}$, $\boldsymbol{N}_{\text{r}} = \boldsymbol{W}^{\mathsf{T}}\boldsymbol{N}\boldsymbol{V}$, $\boldsymbol{b}_{\text{r}} = \boldsymbol{W}^{\mathsf{T}}\boldsymbol{b}$, $\boldsymbol{c}_{\text{r}}^{\mathsf{T}} = \boldsymbol{c}^{\mathsf{T}}\boldsymbol{V}$) we ensure the desired moment matching conditions from (4.5) and (4.6) [2].

*Example* 4.1 (Projection Matrix $\boldsymbol{V}$ For SISO Subsystem Interpolation). [8] Since we wrote above very general formulas, let us consider an example where we want to match the first and the second subsystem at two expansion points $\sigma_1$ and $\sigma_2$. Hence, the moment matching conditions are

$$\begin{aligned}
G_1(\sigma_1) &\equiv G_{1,\text{r}}(\sigma_1), & G_1(\sigma_2) &\equiv G_{1,\text{r}}(\sigma_2), \\
G_2^{\square}(\sigma_1,\sigma_1) &\equiv G_{2,\text{r}}^{\square}(\sigma_1,\sigma_1), & G_2^{\square}(\sigma_2,\sigma_1) &\equiv G_{2,\text{r}}^{\square}(\sigma_2,\sigma_1), \\
G_2^{\square}(\sigma_1,\sigma_2) &\equiv G_{2,\text{r}}^{\square}(\sigma_1,\sigma_2), & G_2^{\square}(\sigma_2,\sigma_2) &\equiv G_{2,\text{r}}^{\square}(\sigma_2,\sigma_2).
\end{aligned}$$

The conditions for the first subsystem are fulfilled if we construct the first projection matrix as follows

$$\boldsymbol{V}^{(1)} = \left[(\sigma_1\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{b}, (\sigma_2\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{b}\right].$$

The conditions for the second subsystem are fulfilled by constructing $\boldsymbol{V}^{(2)}$ as

$$\begin{aligned}
\boldsymbol{V}^{(2)} = \Big[&(\sigma_1\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{N}(\sigma_1\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{b}, (\sigma_1\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{N}(\sigma_2\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{b}, \\
&(\sigma_2\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{N}(\sigma_1\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{b}, (\sigma_2\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{N}(\sigma_2\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{b}\Big].
\end{aligned}$$

Concatenating $\boldsymbol{V}^{(1)}$ and $\boldsymbol{V}^{(2)}$ finally yields the projection matrix

$$\boldsymbol{V} = \left[\boldsymbol{V}^{(1)}, \boldsymbol{V}^{(2)}\right] \in \mathbb{R}^{n\times 6}.$$

△

### 4.2.2 MIMO Subsystem Interpolation

As mentioned, it is possible to expand this framework to reduce MIMO bilinear systems. To prohibit a large increase of the dimensions of the ROM, we match the moments along so-called tangential directions and along a sum over all input combinations. Since we match the moments via the input *Krylov* subspace as well as via the output *Krylov* subspace we need right tangential directions as well as left tangential directions. This yields the moment matching conditions for the input subspace [8]

$$\begin{aligned}
\boldsymbol{G}_1(\sigma_{l_1})\,\boldsymbol{r}_{l_1} &\equiv \boldsymbol{G}_{1,\mathrm{r}}(\sigma_{l_1})\,\boldsymbol{r}_{l_1}, && l_1 = 1,\dots,r\\
\sum_{j_2=1}^{m} \boldsymbol{G}_{2,\square}^{(j_2)}(\sigma_{l_1},\sigma_{l_2})\,\boldsymbol{r}_{l_1} &\equiv \sum_{j_2=1}^{m} \boldsymbol{G}_{2,\mathrm{r},\square}^{(j_2)}(\sigma_{l_1},\sigma_{l_2})\,\boldsymbol{r}_{l_1}, && l_1,l_2 = 1,\dots,r\\
&\vdots\\
\sum_{j_2=1}^{m}\cdots\sum_{j_k=1}^{m} \boldsymbol{G}_{k,\square}^{(j_2,\dots,j_k)}(\sigma_{l_1},\dots,\sigma_{l_k})\boldsymbol{r}_{l_1} &\equiv \sum_{j_2=1}^{m}\cdots\sum_{j_k=1}^{m} \boldsymbol{G}_{k,\mathrm{r},\square}^{(j_2,\dots,j_k)}(\sigma_{l_1},\dots,\sigma_{l_k})\boldsymbol{r}_{l_1},\\
& && l_1,\cdots,l_k = 1,\dots,r
\end{aligned}$$

as well as the moment matching conditions for the output subspace

$$\begin{aligned}
\boldsymbol{l}_{l_1}^{\mathsf{T}}\,\boldsymbol{G}_1(\mu_{l_1}) &\equiv \boldsymbol{l}_{l_1}^{\mathsf{T}}\,\boldsymbol{G}_{1,\mathrm{r}}(\mu_{l_1}), && l_1 = 1,\dots,r\\
\sum_{j_2=1}^{m} \boldsymbol{l}_{l_1}^{\mathsf{T}}\,\boldsymbol{G}_{2,\square}^{(j_2)}(\mu_{l_2},\mu_{l_1}) &\equiv \sum_{j_2=1}^{m} \boldsymbol{l}_{l_1}^{\mathsf{T}}\,\boldsymbol{G}_{2,\mathrm{r},\square}^{(j_2)}(\mu_{l_2},\mu_{l_1}), && l_1,l_2 = 1,\dots,r\\
&\vdots\\
\sum_{j_2=1}^{m}\cdots\sum_{j_k=1}^{m} \boldsymbol{l}_{l_1}^{\mathsf{T}}\,\boldsymbol{G}_{k,\square}^{(j_2,\dots,j_k)}(\mu_{l_k},\dots,\mu_{l_1}) &\equiv \sum_{j_2=1}^{m}\cdots\sum_{j_k=1}^{m} \boldsymbol{l}_{l_1}^{\mathsf{T}}\,\boldsymbol{G}_{k,\mathrm{r},\square}^{(j_2,\dots,j_k)}(\mu_{l_k},\dots,\mu_{l_1}),\\
& && l_1,\cdots,l_k = 1,\dots,r.
\end{aligned}$$

These conditions hold if we construct the projection matrix with the input and output *Krylov* subspace but this time also considering the tangential direction $\boldsymbol{r}_l$ and $\boldsymbol{l}_l^{\mathsf{T}}$. Hence, it yields [8]

$$\begin{aligned}
\boldsymbol{V}^{(1)} &\subseteq \mathcal{K}_r\left((\sigma_{l_1}\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E}, (\sigma_{l_1}\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{B}\,\boldsymbol{r}_{l_1}\right) \in \mathbb{R}^{n\times r}\\
\boldsymbol{V}^{(2)} &\subseteq \mathcal{K}_r\left((\sigma_{l_2}\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E}, \sum_{j=1}^{m}(\sigma_{l_2}\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{N}_j\boldsymbol{V}^{(1)}\right) \in \mathbb{R}^{n\times r^2}\\
&\vdots\\
\boldsymbol{V}^{(k)} &\subseteq \mathcal{K}_r\left((\sigma_{l_k}\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E}, \sum_{j=1}^{m}(\sigma_{l_k}\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{N}_j\boldsymbol{V}^{(k-1)}\right) \in \mathbb{R}^{n\times r^k},\ k = 3,\dots,N
\end{aligned}$$

and

$$
\begin{aligned}
\boldsymbol{W}^{(1)} &\subseteq \mathcal{K}_r\left((\mu_{l_1}\boldsymbol{E}-\boldsymbol{A})^{-\mathsf{T}}\boldsymbol{E}^{\mathsf{T}}, (\mu_{l_1}\boldsymbol{E}-\boldsymbol{A})^{-\mathsf{T}}\boldsymbol{C}^{\mathsf{T}}\boldsymbol{l}_{l_1}\right) \in \mathbb{R}^{n\times r}\\
\boldsymbol{W}^{(2)} &\subseteq \mathcal{K}_r\left((\mu_{l_2}\boldsymbol{E}-\boldsymbol{A})^{-\mathsf{T}}\boldsymbol{E}^{\mathsf{T}}, \sum_{j=1}^{m}(\mu_{l_2}\boldsymbol{E}-\boldsymbol{A})^{-\mathsf{T}}\boldsymbol{N}_j^{\mathsf{T}}\boldsymbol{W}^{(1)}\right) \in \mathbb{R}^{n\times r^2}\\
&\vdots\\
\boldsymbol{W}^{(k)} &\subseteq \mathcal{K}_r\left((\mu_{l_k}\boldsymbol{E}-\boldsymbol{A})^{-\mathsf{T}}\boldsymbol{E}^{\mathsf{T}}, \sum_{j=1}^{m}(\mu_{l_k}\boldsymbol{E}-\boldsymbol{A})^{-\mathsf{T}}\boldsymbol{N}_j^{\mathsf{T}}\boldsymbol{W}^{(k-1)}\right) \in \mathbb{R}^{n\times r^k},\ k=3,...,N.
\end{aligned}
$$

Similar to the SISO case we can construct $\boldsymbol{V}$ and $\boldsymbol{W}$ as

$$
\begin{aligned}
\boldsymbol{V} &= \left[\boldsymbol{V}^{(1)},\ \cdots,\ \boldsymbol{V}^{(k)}\right] \in \mathbb{R}^{n\times r_{\text{tot}}}\\
\boldsymbol{W} &= \left[\boldsymbol{W}^{(1)},\ \cdots,\ \boldsymbol{W}^{(k)}\right] \in \mathbb{R}^{n\times r_{\text{tot}}}
\end{aligned}
$$

where again $r_{\text{tot}} = \sum_{k=1}^{N} r^k$. The above moment matching conditions are once more fulfilled if we construct $\boldsymbol{V}$ and $\boldsymbol{W}$ as shown and apply them to the FOM to obtain the ROM whose system matrices are defined by $\boldsymbol{E}_{\text{r}} = \boldsymbol{W}^{\mathsf{T}}\boldsymbol{E}\boldsymbol{V}$, $\boldsymbol{A}_{\text{r}} = \boldsymbol{W}^{\mathsf{T}}\boldsymbol{A}\boldsymbol{V}$, $\boldsymbol{N}_{\text{r},j} = \boldsymbol{W}^{\mathsf{T}}\boldsymbol{N}_j\boldsymbol{V}$, $\boldsymbol{B}_{\text{r}} = \boldsymbol{W}^{\mathsf{T}}\boldsymbol{B}$, $\boldsymbol{C}_{\text{r}} = \boldsymbol{C}^{\mathsf{T}}\boldsymbol{V}$ [8][11] [15] .

*Example* 4.2 (Projection Matrix $\boldsymbol{V}$ For MIMO Subsystem Interpolation)*.* [8] Let us consider a MIMO bilinear system with two inputs. We want to match the FOM and the ROM at two shifts $\sigma_1$ and $\sigma_2$ and along the tangential directions $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$. Hence, we can give the interpolation conditions as follows

$$
\begin{aligned}
&\boldsymbol{G}_1(\sigma_1)\boldsymbol{r}_1 \equiv \boldsymbol{G}_{1,\text{r}}(\sigma_1)\boldsymbol{r}_1, && \boldsymbol{G}_1(\sigma_2)\boldsymbol{r}_2 \equiv \boldsymbol{G}_{1,\text{r}}(\sigma_2)\boldsymbol{r}_2,\\
&\sum_{j_2=1}^{2}\boldsymbol{G}_2^{(j_2)}(\sigma_1,\sigma_1)\boldsymbol{r}_1 \equiv \sum_{j_2=1}^{2}\boldsymbol{G}_{2,\text{r}}^{(j_2)}(\sigma_1,\sigma_1)\boldsymbol{r}_1, && \sum_{j_2=1}^{2}\boldsymbol{G}_2^{(j_2)}(\sigma_2,\sigma_1)\boldsymbol{r}_1 \equiv \sum_{j_2=1}^{2}\boldsymbol{G}_{2,\text{r}}^{(j_2)}(\sigma_2,\sigma_1)\boldsymbol{r}_1,\\
&\sum_{j_2=1}^{2}\boldsymbol{G}_2^{(j_2)}(\sigma_1,\sigma_2)\boldsymbol{r}_2 \equiv \sum_{j_2=1}^{2}\boldsymbol{G}_{2,\text{r}}^{(j_2)}(\sigma_1,\sigma_2)\boldsymbol{r}_2, && \sum_{j_2=1}^{2}\boldsymbol{G}_2^{(j_2)}(\sigma_2,\sigma_2)\boldsymbol{r}_2 \equiv \sum_{j_2=1}^{2}\boldsymbol{G}_{2,\text{r}}^{(j_2)}(\sigma_2,\sigma_2)\boldsymbol{r}_2.
\end{aligned}
$$

We can fulfill these conditions by constructing $\boldsymbol{V}^{(k)}$ for each subsystem as follows

$$
\begin{aligned}
\boldsymbol{V}^{(1)} &= \left[(\sigma_1\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{B}\,\boldsymbol{r}_1, (\sigma_2\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{B}\,\boldsymbol{r}_2\right]\\
\boldsymbol{V}^{(2)} &= \Big[(\sigma_1\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{N}_1(\sigma_1\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{B}\,\boldsymbol{r}_1 + (\sigma_1\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{N}_2(\sigma_1\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{B}\,\boldsymbol{r}_1,\\
&\quad (\sigma_1\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{N}_1(\sigma_2\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{B}\,\boldsymbol{r}_2 + (\sigma_1\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{N}_2(\sigma_2\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{B}\,\boldsymbol{r}_2,\\
&\quad (\sigma_2\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{N}_1(\sigma_1\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{B}\,\boldsymbol{r}_1 + (\sigma_2\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{N}_2(\sigma_1\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{B}\,\boldsymbol{r}_1,\\
&\quad (\sigma_2\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{N}_1(\sigma_2\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{B}\,\boldsymbol{r}_2 + (\sigma_2\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{N}_2(\sigma_2\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{B}\,\boldsymbol{r}_2\Big]
\end{aligned}
$$

and obtain the projection matrix $\boldsymbol{V}$ by concatenating $\boldsymbol{V}^{(1)}$ and $\boldsymbol{V}^{(2)}$ such that

$$\boldsymbol{V} = \left[\boldsymbol{V}^{(1)}, \boldsymbol{V}^{(2)}\right] \in \mathbb{R}^{n\times 6}.$$

$\triangle$

*Remark* 4.1 (Block Krylov Subsystem Interpolation). [8] Note, that it is also possible to match the transfer functions without tangential directions $\boldsymbol{r}$ and $\boldsymbol{l}^{\mathsf{T}}$. This could be achieved in this framework by choosing the tangential vectors as identity matrices such that $\boldsymbol{r} = \mathbf{I}_m$ and $\boldsymbol{l}^{\mathsf{T}} = \mathbf{I}_p$. In this case we call it block *Krylov* subsystem interpolation and the reduced order $r_{\text{tot}} = \sum_{k=1}^{N} mr^k$ or rather $r_{\text{tot}} = \sum_{k=1}^{N} pr^k$ gets greater by a factor $m$ or $p$. Hence, in the block *Krylov* case one obtains a greater dimension for the ROM with the same amount of interpolation points compared to the tangential subsystem interpolation. △

*Remark* 4.2 (Different Approach For MIMO Subsystem Interpolation). [8] In the above approach we used *Krylov* subspaces where we sum up over all inputs or rather outputs. According to our definition of the regular transfer function $\boldsymbol{G}_{k,\square}^{(j_1,\dots,j_k)}(s_1,\dots,s_k)$ it would be more straightforward to use the following *Krylov* subspaces

$$\begin{aligned}
\boldsymbol{V}^{(k)} &\subseteq \mathcal{K}_r\left((\sigma_{l_k}\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E}, (\sigma_{l_k}\boldsymbol{E}-\boldsymbol{A})^{-1}\bar{\boldsymbol{N}}(\mathbf{I}\otimes\boldsymbol{V}^{(k-1)})\right) \in \mathbb{R}^{n\times r^k m^{k-1}},\\
\boldsymbol{W}^{(k)} &\subseteq \mathcal{K}_r\left((\mu_{l_k}\boldsymbol{E}-\boldsymbol{A})^{-\mathsf{T}}\boldsymbol{E}^{\mathsf{T}}, (\mu_{l_k}\boldsymbol{E}-\boldsymbol{A})^{-\mathsf{T}}\bar{\boldsymbol{N}}^{\mathsf{T}}(\mathbf{I}\otimes\boldsymbol{W}^{(k-1)})\right) \in \mathbb{R}^{n\times r^k m^{k-1}}
\end{aligned}$$

where $\bar{\boldsymbol{N}} = [\boldsymbol{N}_1,\dots,\boldsymbol{N}_m]$ and $\bar{\boldsymbol{N}}^{\mathsf{T}} = [\boldsymbol{N}_1^{\mathsf{T}},\dots,\boldsymbol{N}_m^{\mathsf{T}}]$. With this *Krylov* subspaces we fulfill the following stronger moment matching conditions for input tangential subsystem interpolation

$$\begin{aligned}
\boldsymbol{G}_1(\sigma_{l_1})\,\boldsymbol{r}_{l_1} &\equiv \boldsymbol{G}_{1,\mathrm{r}}(\sigma_{l_1})\,\boldsymbol{r}_{l_1}, && l_1 = 1,\dots,r\\
\boldsymbol{G}_{2,\square}^{(j_2)}(\sigma_{l_1},\sigma_{l_2})\,\boldsymbol{r}_{l_1} &\equiv \boldsymbol{G}_{2,\mathrm{r},\square}^{(j_2)}(\sigma_{l_1},\sigma_{l_2})\,\boldsymbol{r}_{l_1}, && l_1,l_2 = 1,\dots,r\\
&\vdots\\
\boldsymbol{G}_{k,\square}^{(j_2,\dots,j_k)}(\sigma_{l_1},\dots,\sigma_{l_k})\boldsymbol{r}_{l_1} &\equiv \boldsymbol{G}_{k,\mathrm{r},\square}^{(j_2,\dots,j_k)}(\sigma_{l_1},\dots,\sigma_{l_k})\boldsymbol{r}_{l_1}, && l_1,\cdots,l_k = 1,\dots,r
\end{aligned}$$

and output tangential subsystem interpolation

$$\begin{aligned}
\boldsymbol{l}_{l_1}^{\mathsf{T}}\,\boldsymbol{G}_1(\mu_{l_1}) &\equiv \boldsymbol{l}_{l_1}^{\mathsf{T}}\,\boldsymbol{G}_{1,\mathrm{r}}(\mu_{l_1}), && l_1 = 1,\dots,r\\
\boldsymbol{l}_{l_1}^{\mathsf{T}}\,\boldsymbol{G}_{2,\square}^{(j_2)}(\mu_{l_2},\mu_{l_1}) &\equiv \boldsymbol{l}_{l_1}^{\mathsf{T}}\,\boldsymbol{G}_{2,\mathrm{r},\square}^{(j_2)}(\mu_{l_2},\mu_{l_1}), && l_1,l_2 = 1,\dots,r\\
&\vdots\\
\boldsymbol{l}_{l_1}^{\mathsf{T}}\,\boldsymbol{G}_{k,\square}^{(j_2,\dots,j_k)}(\mu_{l_k},\dots,\mu_{l_1}) &\equiv \boldsymbol{l}_{l_1}^{\mathsf{T}}\,\boldsymbol{G}_{k,\mathrm{r},\square}^{(j_2,\dots,j_k)}(\mu_{l_k},\dots,\mu_{l_1}), && l_1,\cdots,l_k = 1,\dots,r.
\end{aligned}$$

These moment matching conditions are stronger because we match every combination of $(j_2,\dots,j_k)$ independently. The dimensions $r_{\text{tot}}$ of the ROM within this framework are growing even faster with each additional considered subsystem since $r_{\text{tot}} = \sum_{k=1}^{N} r^k m^{k-1}$. △

*Remark* 4.3 (Higher Order Moments). Similar to the linear case it is also possible to match high-order moments as well as to combine this with *Markov Parameters* and shifts which are only matched at the 0-th moment. For more insight we refer to [15]. △

### 4.2.3 Combinatoric Problem

Throughout the provided examples it is quite obvious that on the one hand the order of the ROM grows by the power of considered subsystems and on the other hand that we have to fulfill even more combinations of shifts if we consider more subsystems. We illustrate the mentioned problem with a SISO system $\zeta$. Assume that we want to obtain a reduced bilinear system $\zeta_{\mathrm{r}}$ corresponding to $\zeta$ by interpolating the 0-th moment at $r = 2$ shifts $\sigma_1$ and $\sigma_2$. Hence, for the first subsystem, we have to match two moments

$$G_1(\sigma_1) \equiv G_{1,\mathrm{r}}(\sigma_1), \quad G_1(\sigma_2) \equiv G_{1,\mathrm{r}}(\sigma_2).$$

For the second subsystem we have to match four moments

$$\begin{aligned} &G_2^{\square}(\sigma_1,\sigma_1) \equiv G_{2,\mathrm{r}}^{\square}(\sigma_1,\sigma_1), \quad G_2^{\square}(\sigma_2,\sigma_1) \equiv G_{2,\mathrm{r}}^{\square}(\sigma_2,\sigma_1),\\ &G_2^{\square}(\sigma_1,\sigma_2) \equiv G_{2,\mathrm{r}}^{\square}(\sigma_1,\sigma_2), \quad G_2^{\square}(\sigma_2,\sigma_2) \equiv G_{2,\mathrm{r}}^{\square}(\sigma_2,\sigma_2)\end{aligned}$$

and for the third subsystem eight moments

$$\begin{aligned} &G_3^{\square}(\sigma_1,\sigma_1,\sigma_1) \equiv G_{3,\mathrm{r}}^{\square}(\sigma_1,\sigma_1,\sigma_1), \quad G_3^{\square}(\sigma_2,\sigma_1,\sigma_1) \equiv G_{3,\mathrm{r}}^{\square}(\sigma_2,\sigma_1,\sigma_1),\\ &G_3^{\square}(\sigma_1,\sigma_2,\sigma_1) \equiv G_{3,\mathrm{r}}^{\square}(\sigma_1,\sigma_2,\sigma_1) \quad G_3^{\square}(\sigma_2,\sigma_2,\sigma_1) \equiv G_{3,\mathrm{r}}^{\square}(\sigma_2,\sigma_2,\sigma_1),\\ &G_3^{\square}(\sigma_1,\sigma_1,\sigma_2) \equiv G_{3,\mathrm{r}}^{\square}(\sigma_1,\sigma_1,\sigma_2), \quad G_3^{\square}(\sigma_2,\sigma_1,\sigma_2) \equiv G_{3,\mathrm{r}}^{\square}(\sigma_2,\sigma_1,\sigma_2),\\ &G_3^{\square}(\sigma_1,\sigma_2,\sigma_2) \equiv G_{3,\mathrm{r}}^{\square}(\sigma_1,\sigma_2,\sigma_2) \quad G_3^{\square}(\sigma_2,\sigma_2,\sigma_2) \equiv G_{3,\mathrm{r}}^{\square}(\sigma_2,\sigma_2,\sigma_2).\end{aligned}$$

Consequently, we have to match $r^k$ moments for the $k$-th subsystem which obviously is more complicated to handle the more subsystems being considered. In addition to that, the total reduced order $r_{\mathrm{tot}} = \sum_{k=1}^{N} r^k$ is growing by the power of $k$. This makes it inefficient to increase the accuracy of the ROM by considering more subsystems. Considering one more subsystem means matching $r^k$ more moments and increasing the order of the ROM by $r^k$ in addition to the existing order.

*Remark* 4.4 (Workarounds For The Combinatoric Problem). [8] As mentioned, it is quite complicated and inefficient to compute all multi-moments for additional subsystems. Hence, we want to provide some ideas how to reduce the complexity.

- The most straightforward way is to simply use one interpolation point which completely eliminates the combinatoric problem. Of course this approach leads to less global accuracy of the ROM.
- As the importance of the subsystem decreases with greater $k$ one could choose less matched moments for higher subsystems such that the reduced order of additional subsystems decreases.

It is possible to find a lot more workarounds [8] as the subsystem interpolation framework is highly customizable. $\triangle$

## 4.3 Volterra Series Interpolation

The *Volterra* series interpolation framework was first introduced by *Flagg* and *Gugercin* in [10]. It turned out that the method was implicitly used by [24] and [2] for an $\mathcal{H}_2$-optimal model reduction algorithm called BIRKA. As one can tell from the name, this reduction method employs the underlying *Volterra* series representation. The basic idea is to match the previously introduced multi-moments at certain shift combinations along weighted sums, in other words: to match the whole *Volterra* series at once. With this summation feature the combinatoric problem gets eliminated [8]. We discover that it

is also possible to gain the projection matrices $\boldsymbol{V}$ and $\boldsymbol{W}$ through solving *Sylvester* equations.
Let us begin by introducing the framework in its original form: for multiple interpolation points. In this sense we explore the *Sylvester* equations and different ways to compute the projection matrices, which we illustrate in algorithms. After that, we extend the *Volterra* series interpolation to match high-order moments and show which constraints the interpolation data has to fulfill to obtain again a *Sylvester* equations representation. We also provide algorithms to construct the projection matrices for higher moments. Lastly, we discuss other special cases of this framework like the block *Krylov* case, *Markov Parameters* and complex expansion points.

### 4.3.1 Multipoint Volterra Series Interpolation

As mentioned, this method matches the whole *Volterra* series along weighted sums. While only considering the 0-th moment of each subsystem we can define the interpolation conditions for a SISO system as follows

$$\begin{gathered}\sum_{k=1}^{\infty}\sum_{l_1=1}^{r}\cdots\sum_{l_{k-1}=1}^{r}\eta_{l_1,\ldots,l_{k-1},i}\,G_k^{\square}(\sigma_{l_1},\ldots,\sigma_{l_{k-1}},\sigma_i)\\ \equiv\\ \sum_{k=1}^{\infty}\sum_{l_1=1}^{r}\cdots\sum_{l_{k-1}=1}^{r}\eta_{l_1,\ldots,l_{k-1},i}\,G_{k,\mathrm{r}}^{\square}(\sigma_{l_1},\ldots,\sigma_{l_{k-1}},\sigma_i)\end{gathered} \tag{4.8}$$

where each $\eta$ denotes a weight for a certain shift combination and where $G_{\mathrm{r},k}^{\square}$ is the $k$-th transfer function of the reduced system. Note that we will later specify the weights $\eta$ through entries of a matrix $\boldsymbol{U}_v$ which requires those indexes. For now, we assume that each $\eta_{l_1,\ldots,l_{k-1},i}$ is arbitrary. We again write the moments as the transfer functions, since the 0-th moment of the $k$-th subsystem is equal to the $k$-th transfer function. Note that we hold $\sigma_i$ throughout all sums. A necessary condition to be able to make use of the *Volterra* series interpolation is obviously the convergence of the series.

*Example* 4.3 (Volterra Series Interpolation Conditions). To enlighten the interpolation conditions a little more let us consider a simple example. Assuming that we want to match the FOM and the ROM at two ($r = 2$) interpolation points $\sigma_1$ and $\sigma_2$, we write the interpolation condition for the first shift $\sigma_1$ as

$$\begin{gathered}\sum_{k=1}^{\infty}\sum_{l_1=1}^{2}\cdots\sum_{l_{k-1}=1}^{2}\eta_{l_1,\ldots,l_{k-1},1}\,G_k^{\square}(\sigma_{l_1},\ldots,\sigma_{l_{k-1}},\sigma_1)=\\ G_1(\sigma_1)+\eta_{1,1}G_2^{\square}(\sigma_1,\sigma_1)+\eta_{2,1}G_2^{\square}(\sigma_2,\sigma_1)+\eta_{1,1,1}G_3^{\square}(\sigma_1,\sigma_1,\sigma_1)+\ldots\\ \equiv\\ \sum_{k=1}^{\infty}\sum_{l_1=1}^{2}\cdots\sum_{l_{k-1}=1}^{2}\eta_{l_1,\ldots,l_{k-1},1}\,G_{k,\mathrm{r}}^{\square}(\sigma_{l_1},\ldots,\sigma_{l_{k-1}},\sigma_1)=\\ G_{1,\mathrm{r}}(\sigma_1)+\eta_{1,1}G_{2,\mathrm{r}}^{\square}(\sigma_1,\sigma_1)+\eta_{2,1}G_{2,\mathrm{r}}^{\square}(\sigma_2,\sigma_1)+\eta_{1,1,1}G_{3,\mathrm{r}}^{\square}(\sigma_1,\sigma_1,\sigma_1)+\ldots\ .\end{gathered}$$

Similar to that, we write the interpolation condition for the second shift $\sigma_2$ as

$$\sum_{k=1}^{\infty}\sum_{l_1=1}^{2}\cdots\sum_{l_{k-1}=1}^{2}\eta_{l_1,\dots,l_{k-1},2}\,G_k^{\square}(\sigma_{l_1},\dots,\sigma_{l_{k-1}},\sigma_2) =$$
$$G_1(\sigma_2)+\eta_{1,2}G_2^{\square}(\sigma_1,\sigma_2)+\eta_{2,2}G_2^{\square}(\sigma_2,\sigma_2)+\eta_{1,1,2}G_3^{\square}(\sigma_1,\sigma_1,\sigma_2)+\dots$$
$$\equiv$$
$$\sum_{k=1}^{\infty}\sum_{l_1=1}^{2}\cdots\sum_{l_{k-1}=1}^{2}\eta_{l_1,\dots,l_{k-1},2}\,G_{k,\mathrm{r}}^{\square}(\sigma_{l_1},\dots,\sigma_{l_{k-1}},\sigma_2) =$$
$$G_{1,\mathrm{r}}(\sigma_2)+\eta_{1,2}G_{2,\mathrm{r}}^{\square}(\sigma_1,\sigma_2)+\eta_{2,2}G_{2,\mathrm{r}}^{\square}(\sigma_2,\sigma_2)+\eta_{1,1,2}G_{3,\mathrm{r}}^{\square}(\sigma_1,\sigma_1,\sigma_2)+\dots\ .$$

As one can clearly see, we combine all interpolation points with the current interpolation point. Since we sum these combinations up, we receive $r = 2$ interpolation conditions. △

Again it is also possible to match the moments with a projection matrix constructed through the output *Krylov* subspace. Therefore, we use different interpolation points $\mu$ and weights $\vartheta$. Hence we write the output interpolation conditions as follows

$$\begin{gathered}\sum_{k=1}^{\infty}\sum_{l_1=1}^{r}\cdots\sum_{l_{k-1}=1}^{r}\vartheta_{l_1,\dots,l_{k-1},i}\,G_k^{\square}(\mu_i,\mu_{l_{k-1}},\dots,\mu_{l_1})\\ \equiv\\ \sum_{k=1}^{\infty}\sum_{l_1=1}^{r}\cdots\sum_{l_{k-1}=1}^{r}\vartheta_{l_1,\dots,l_{k-1},i}\,G_{k,\mathrm{r}}^{\square}(\mu_i,\mu_{l_{k-1}},\dots,\mu_{l_1}).\end{gathered} \tag{4.9}$$

The conditions (4.8) and(4.9) hold if we reduce the FOM with the projection matrix $\boldsymbol{V}$ corresponding to the following input *Krylov* subspace

$$\boldsymbol{V}\subseteq\mathcal{K}_r\left((\sigma_i\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E},\boldsymbol{v}_i\right)\in\mathbb{R}^{n\times r}$$

where

$$\boldsymbol{v}_i=\sum_{k=1}^{\infty}\sum_{l_1=1}^{r}\cdots\sum_{l_{k-1}=1}^{r}\eta_{l_1,\dots,l_{k-1},i}(\sigma_i\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{N}(\sigma_{l_{k-1}}\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{N}\cdots\boldsymbol{N}(\sigma_{l_1}\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{b} \tag{4.10}$$

and the projection matrix $\boldsymbol{W}$ corresponding to the following output *Krylov* subspaces

$$\boldsymbol{W}\subseteq\mathcal{K}_r\left((\mu_i\boldsymbol{E}-\boldsymbol{A})^{-\mathsf{T}}\boldsymbol{E}^{\mathsf{T}},\boldsymbol{w}_i\right)\in\mathbb{R}^{n\times r}$$

where

$$\boldsymbol{w}_i=\sum_{k=1}^{\infty}\sum_{l_1=1}^{r}\cdots\sum_{l_{k-1}=1}^{r}\vartheta_{l_1,\dots,l_{k-1},i}(\mu_i\boldsymbol{E}-\boldsymbol{A})^{-\mathsf{T}}\boldsymbol{N}^{\mathsf{T}}(\mu_{l_{k-1}}\boldsymbol{E}-\boldsymbol{A})^{-\mathsf{T}}\boldsymbol{N}^{\mathsf{T}}\cdots\boldsymbol{N}^{\mathsf{T}}(\mu_{l_k}\boldsymbol{E}-\boldsymbol{A})^{-\mathsf{T}}\boldsymbol{c}. \tag{4.11}$$

Different to subsystem interpolation we obtain projection matrices in $\mathbb{R}^{n\times r}$, which means that our reduced order is equal to the number of shifts $r$. As one might have noticed, the first sum $\sum_{k=1}^{\infty}$ corresponds to the previously introduced *Volterra* series. Assuming that we interpolate at $r$ shifts $\sigma_1, \dots, \sigma_r$ where $\sigma_1 \neq \dots \neq \sigma_r$ we obtain the projection matrices by computing each column $\boldsymbol{v}_r$ corresponding to (4.10) and analogously for $r$ shifts $\mu_1, \dots, \mu_r$ where $\mu_1 \neq \dots \neq \mu_r$ by computing each column $\boldsymbol{w}_r$ corresponding to (4.11). This results in

$$\begin{aligned}\boldsymbol{V} &= \begin{bmatrix}\boldsymbol{v}_1, & \cdots, & \boldsymbol{v}_r\end{bmatrix} \in \mathbb{R}^{n\times r}\\ \boldsymbol{W} &= \begin{bmatrix}\boldsymbol{w}_1, & \cdots, & \boldsymbol{w}_r\end{bmatrix} \in \mathbb{R}^{n\times r}.\end{aligned}$$

*Example* 4.4 (Computing A Column Of $\boldsymbol{V}$). [8] Let us assume that we want to reduce a SISO bilinear system to order $r = 2$. Therefore, we use the interpolation points $\sigma_1$ and $\sigma_2$. We want to illustrate how the first column of $\boldsymbol{V}$ is computed. From above we know that

$$\boldsymbol{v}_1 = \sum_{k=1}^{\infty}\sum_{l_1=1}^{2}\cdots\sum_{l_{k-1}=1}^{2}\eta_{l_1,\dots,l_{k-1},1}(\sigma_1\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{N}(\sigma_{l_{k-1}}\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{N}\cdots\boldsymbol{N}(\sigma_{l_1}\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{b}.$$

To make things easier, we make use of the *Volterra* series and write $\boldsymbol{v}_1$ as follows

$$\boldsymbol{v}_1 = \sum_{k=1}^{\infty}\boldsymbol{v}_1^{(k)} = \boldsymbol{v}_1^{(1)} + \boldsymbol{v}_1^{(2)} + \boldsymbol{v}_1^{(3)} + \boldsymbol{v}_1^{(4)} + \cdots$$

where $\boldsymbol{v}_1^{(k)}$ denotes the first column of $\boldsymbol{V}^{(k)}$ corresponding to the subsystem $k$. Hence, $\boldsymbol{v}_1^{(1)}$ is given by

$$\boldsymbol{v}_1^{(1)} = (\sigma_1\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{b}.$$

Accordingly, we can compute $\boldsymbol{v}_1^{(2)}$ for the second subsystem

$$\begin{aligned}\boldsymbol{v}_1^{(2)} &= \sum_{l_1=1}^{2}\eta_{l_1,1}(\sigma_1\boldsymbol{E}-\boldsymbol{A})\boldsymbol{N}(\sigma_{l_1}\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{b}\\ &= \eta_{1,1}(\sigma_1\boldsymbol{E}-\boldsymbol{A})\boldsymbol{N}(\sigma_1\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{b} + \eta_{2,1}(\sigma_1\boldsymbol{E}-\boldsymbol{A})\boldsymbol{N}(\sigma_2\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{b}\end{aligned}$$

and $\boldsymbol{v}_1^{(3)}$ for the third subsystem

$$\boldsymbol{v}_1^{(3)} = \sum_{l_1=1}^{2}\sum_{l_2=1}^{2}\eta_{l_1,l_2,1}(\sigma_1\boldsymbol{E}-\boldsymbol{A})\boldsymbol{N}(\sigma_{l_2}\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{N}(\sigma_{l_1}\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{b}$$

Continuing for all subsystems finally yields $\boldsymbol{v}_1$ via summation over all subsystems.

△

Finally, we obtain the ROM analogously to the subsystem interpolation by applying the projection matrices to the system matrices as shown in the fundamentals ($\boldsymbol{E}_\mathrm{r} = \boldsymbol{W}^\top\boldsymbol{E}\boldsymbol{V}$, $\boldsymbol{A}_\mathrm{r} = \boldsymbol{W}^\top\boldsymbol{A}\boldsymbol{V}$, $\boldsymbol{N}_\mathrm{r} = \boldsymbol{W}^\top\boldsymbol{N}\boldsymbol{V}$, $\boldsymbol{b}_\mathrm{r} = \boldsymbol{W}^\top\boldsymbol{b}$, $\boldsymbol{c}_\mathrm{r}^\top = \boldsymbol{c}^\top\boldsymbol{V}$) [10].

It is also possible to apply this method to MIMO bilinear systems. Hereby, the moment matching conditions for an input *Krylov* subspace are given by

$$\begin{gathered}\sum_{k=1}^{\infty}\sum_{j_2=1}^{m}\cdots\sum_{j_k=1}^{m}\sum_{l_1=1}^{r}\cdots\sum_{l_{k-1}=1}^{r}\eta_{l_1,\dots,l_{k-1},i}^{j_2,\dots,j_k}\,\boldsymbol{G}_{k,\square}^{(j_2,\dots,j_k)}(\sigma_{l_1},\dots,\sigma_{l_{k-1}},\sigma_i)\,\boldsymbol{r}_{l_1}\\ \equiv \\ \sum_{k=1}^{\infty}\sum_{j_2=1}^{m}\cdots\sum_{j_k=1}^{m}\sum_{l_1=1}^{r}\cdots\sum_{l_{k-1}=1}^{r}\eta_{l_1,\dots,l_{k-1},i}^{j_2,\dots,j_k}\,\boldsymbol{G}_{k,\mathrm{r},\square}^{(j_2,\dots,j_k)}(\sigma_{l_1},\dots,\sigma_{l_{k-1}},\sigma_i)\,\boldsymbol{r}_{l_1}.\end{gathered} \tag{4.12}$$

with additional sums over all inputs. Similar to that, we can give the moment matching conditions for the output case

$$\begin{gathered}\sum_{k=1}^{\infty}\sum_{j_2=1}^{m}\cdots\sum_{j_k=1}^{m}\sum_{l_1=1}^{r}\cdots\sum_{l_{k-1}=1}^{r}\vartheta_{l_1,\dots,l_{k-1},i}^{j_2,\dots,j_k}\,\boldsymbol{l}_i^{\mathsf{T}}\,\boldsymbol{G}_{k,\square}^{(j_2,\dots,j_k)}(\mu_i,\mu_{l_{k-1}},\dots,\mu_{l_1})\\ \equiv \\ \sum_{k=1}^{\infty}\sum_{j_2=1}^{m}\cdots\sum_{j_k=1}^{m}\sum_{l_1=1}^{r}\cdots\sum_{l_{k-1}=1}^{r}\vartheta_{l_1,\dots,l_{k-1},i}^{j_2,\dots,j_k}\,\boldsymbol{l}_i^{\mathsf{T}}\,\boldsymbol{G}_{k,\mathrm{r},\square}^{(j_2,\dots,j_k)}(\mu_i,\mu_{l_{k-1}},\dots,\mu_{l_1}).\end{gathered} \tag{4.13}$$

Again, we use tangential directions $\boldsymbol{r}$ and $\boldsymbol{l}^{\mathsf{T}}$. Note that we still assume that the weights $\eta_{l_1,\dots,l_{k-1},i}^{j_2,\dots,j_k}$ and $\vartheta_{l_1,\dots,l_{k-1},i}^{j_2,\dots,j_k}$ are arbitrary while keeping in mind that we need the indexes to later define them through matrices $\boldsymbol{U}_{v,j}$ and $\boldsymbol{U}_{w,j}$. The MIMO moment matching conditions hold if we build the projection matrix $\boldsymbol{V}$ with use of the following input *Krylov* subspace

$$\boldsymbol{V}\subseteq\mathcal{K}_r\left((\sigma_i\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E},\boldsymbol{v}_i\right)\in\mathbb{R}^{n\times r}$$

where

$$\begin{aligned}\boldsymbol{v}_i=\sum_{k=1}^{\infty}\sum_{j_2=1}^{m}\cdots\sum_{j_k=1}^{m}\sum_{l_1=1}^{r}\cdots\sum_{l_{k-1}=1}^{r}&\eta_{l_1,\dots,l_{k-1},i}^{j_2,\dots,j_k}(\sigma_i\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{N}_{j_k}\\ &\times(\sigma_{l_{k-1}}\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{N}_{j_{k-1}}\cdots\boldsymbol{N}_{j_2}(\sigma_{l_1}\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{B}\,\boldsymbol{r}_{l_1}\end{aligned}$$

and the projection matrix $\boldsymbol{W}$ corresponding to the following output *Krylov* subspaces

$$\boldsymbol{W}\subseteq\mathcal{K}_r\left((\mu_i\boldsymbol{E}-\boldsymbol{A})^{-\mathsf{T}}\boldsymbol{E}^{\mathsf{T}},\boldsymbol{w}_i\right)\in\mathbb{R}^{n\times r}$$

where

$$\begin{aligned}\boldsymbol{w}_i=\sum_{k=1}^{\infty}\sum_{j_2=1}^{m}\cdots\sum_{j_k=1}^{m}\sum_{l_1=1}^{r}\cdots\sum_{l_{k-1}=1}^{r}&\vartheta_{l_1,\dots,l_{k-1},i}^{j_2,\dots,j_k}(\mu_i\boldsymbol{E}-\boldsymbol{A})^{-\mathsf{T}}\boldsymbol{N}_{j_k}^{\mathsf{T}}\\ &\times(\mu_{l_{k-1}}\boldsymbol{E}-\boldsymbol{A})^{-\mathsf{T}}\boldsymbol{N}_{j_{k-1}}^{\mathsf{T}}\dots\boldsymbol{N}_{j_2}^{\mathsf{T}}(\mu_{l_1}\boldsymbol{E}-\boldsymbol{A})^{-\mathsf{T}}\boldsymbol{C}^{\mathsf{T}}\,\boldsymbol{l}_i.\end{aligned}$$

If the ROM gets constructed by $\boldsymbol{E}_\mathrm{r}=\boldsymbol{W}^{\mathsf{T}}\boldsymbol{E}\boldsymbol{V}$, $\boldsymbol{A}_\mathrm{r}=\boldsymbol{W}^{\mathsf{T}}\boldsymbol{A}\boldsymbol{V}$, $\boldsymbol{N}_{\mathrm{r},j}=\boldsymbol{W}^{\mathsf{T}}\boldsymbol{N}_j\boldsymbol{V}$, $\boldsymbol{B}_\mathrm{r}=\boldsymbol{W}^{\mathsf{T}}\boldsymbol{B}$, $\boldsymbol{C}_\mathrm{r}=\boldsymbol{C}^{\mathsf{T}}\boldsymbol{V}$ and $\boldsymbol{V}$ and $\boldsymbol{W}$ as shown above then the moment matching conditions in (4.12) and (4.13) hold [10].

*Remark* 4.5 (Computing Only One Projection Matrix). It is possible to compute only one projection matrix either for the input *Krylov* subspace or for the output *Krylov* subspace. In this case, one could simply choose $\boldsymbol{W}=\boldsymbol{V}$ and vice versa to obtain the ROM. △

**Sylvester Equations**

Our goal now is to show that it is possible to obtain the projection matrix $\boldsymbol{V}$ by solving the following bilinear *Sylvester* equation

$$\boldsymbol{E}\,\boldsymbol{V}\,\boldsymbol{S}_v - \boldsymbol{A}\,\boldsymbol{V} - \sum_{j=1}^{m} \boldsymbol{N}_j\,\boldsymbol{V}\,\boldsymbol{U}_{v,j}^{\mathsf{T}} = \boldsymbol{B}\,\boldsymbol{R}$$

where $\boldsymbol{S}_v = \operatorname{diag}(\sigma_1, \dots, \sigma_r) \in \mathbb{C}^{r\times r}$ , $\boldsymbol{U}_{v,j} = \left\{u_{v,j}^{(a,b)}\right\} \in \mathbb{C}^{r\times r}$ and $\boldsymbol{R} = [\boldsymbol{r}_1, \dots, \boldsymbol{r}_r] \in \mathbb{C}^{m\times r}$. And accordingly, we want to show that we can obtain the projection matrix $\boldsymbol{W}$ by solving

$$\boldsymbol{E}^{\mathsf{T}}\,\boldsymbol{W}\,\boldsymbol{S}_w^{\mathsf{T}} - \boldsymbol{A}^{\mathsf{T}}\,\boldsymbol{W} - \sum_{j=1}^{m} \boldsymbol{N}_j^{\mathsf{T}}\,\boldsymbol{W}\,\boldsymbol{U}_{w,j}^{\mathsf{T}} = \boldsymbol{C}^{\mathsf{T}}\,\boldsymbol{L}$$

where $\boldsymbol{S}_w = \operatorname{diag}(\mu_1, \dots, \mu_r) \in \mathbb{C}^{r\times r}$, $\boldsymbol{U}_{w,j} = \left\{u_{w,j}^{(a,b)}\right\} \in \mathbb{C}^{r\times r}$ and $\boldsymbol{L} = [\boldsymbol{l}_1, \dots, \boldsymbol{l}_r] \in \mathbb{C}^{p\times r}$. Hereby $u_{v,j}^{(a,b)}$ denotes the entry in the $a$-th row and the $b$-th column of $\boldsymbol{U}_{v,j}$ and analogously $u_{w,j}^{(a,b)}$ denotes the entry in the $a$-th row and the $b$-th column of $\boldsymbol{U}_{w,j}$. As mentioned, it is necessary to define the weights $\eta_{l_1,\dots,l_{k-1},i}^{j_2,\dots,j_k}$ and $\vartheta_{l_1,\dots,l_{k-1},i}^{j_2,\dots,j_k}$ through entries of the matrices $\boldsymbol{U}_{v,j}$ and $\boldsymbol{U}_{w,j}$. Hence, we give the definition for the weights as follows

$$\begin{aligned}
\eta_{l_1,\dots,l_{k-1},i}^{j_2,\dots,j_k} &= u_{v,j_k}^{(i,l_{k-1})} u_{v,j_{k-1}}^{(l_{k-2},l_{k-3})} \cdots u_{v,j_2}^{(l_2,l_1)} && \text{for } k \geq 2\\
\vartheta_{l_1,\dots,l_{k-1},i}^{j_2,\dots,j_k} &= u_{w,j_k}^{(i,l_{k-1})} u_{w,j_{k-1}}^{(l_{k-2},l_{k-3})} \cdots u_{w,j_2}^{(l_2,l_1)} && \text{for } k \geq 2.
\end{aligned}$$

As this is not very enlightening, let us try to focus on every subsystem independently. Therefore, we make use of the *Volterra* series and write the projection matrix $\boldsymbol{V}$ as

$$\boldsymbol{V} = \sum_{k=1}^{\infty} \boldsymbol{V}^{(k)} = \boldsymbol{V}^{(1)} + \boldsymbol{V}^{(2)} + \boldsymbol{V}^{(3)} + \boldsymbol{V}^{(4)} + \cdots .$$

Starting with the first subsystem, the $i$-th column $\boldsymbol{v}_i^{(1)}$ of $\boldsymbol{V}^{(1)}$ is defined as

$$\boldsymbol{v}_i^{(1)} = (\sigma_i \boldsymbol{E} - \boldsymbol{A})^{-1} \boldsymbol{B} \boldsymbol{r}_i \tag{4.14}$$

where $i = 1, \dots, r$. While recalling Definition 3.2 and writing $\boldsymbol{V}^{(1)}$ vectorized

$$\begin{aligned}
\operatorname{vec}\left(\boldsymbol{V}^{(1)}\right) = \begin{bmatrix} \boldsymbol{v}_1^{(1)} \\ \vdots \\ \boldsymbol{v}_r^{(1)} \end{bmatrix} &= \begin{bmatrix} \sigma_1 \boldsymbol{E} - \boldsymbol{A} & & \\ & \ddots & \\ & & \sigma_r \boldsymbol{E} - \boldsymbol{A} \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{B} & & \\ & \ddots & \\ & & \boldsymbol{B} \end{bmatrix} \begin{bmatrix} \boldsymbol{r}_1 \\ \vdots \\ \boldsymbol{r}_r \end{bmatrix} \\
&= \left(\boldsymbol{S}_v^{\mathsf{T}} \otimes \boldsymbol{E} - \mathbf{I}_r \otimes \boldsymbol{A}\right)^{-1} (\mathbf{I}_r \otimes \boldsymbol{B}) \operatorname{vec}(\boldsymbol{R})
\end{aligned}$$

it is quite obvious that $\boldsymbol{V}^{(1)}$ is the solution of the following linear *Sylvester* equation

$$\boldsymbol{E}\,\boldsymbol{V}^{(1)}\,\boldsymbol{S}_v - \boldsymbol{A}\,\boldsymbol{V}^{(1)} = \boldsymbol{B}\,\boldsymbol{R}. \tag{4.15}$$

For the second subsystem the $i$-th column $\boldsymbol{v}_i^{(2)}$ of $\boldsymbol{V}^{(2)}$ is defined as

$$\begin{aligned}\boldsymbol{v}_i^{(2)} &= \sum_{j_2=1}^{m}\sum_{l_1=1}^{r} \eta_{l_1,i}^{j_2}(\sigma_i\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{N}_{j_2}\underbrace{(\sigma_{l_1}\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{B}\boldsymbol{r}_{l_1}}_{=\boldsymbol{v}_{l_1}^{(1)}}\\ &= \sum_{j_2=1}^{m}\sum_{l_1=1}^{r} u_{v,j_2}^{(i,l_1)}(\sigma_i\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{N}_{j_2}\boldsymbol{v}_{l_1}^{(1)}\end{aligned}$$

where $i = 1,\dots,r$. As we can see, $\boldsymbol{v}_i^{(2)}$ depends on the previously computed columns $\boldsymbol{v}_{l_1}^{(1)}$. While again writing $\boldsymbol{V}^{(2)}$ in its vectorized form

$$\begin{aligned}\begin{bmatrix}\boldsymbol{v}_1^{(2)}\\ \vdots\\ \boldsymbol{v}_r^{(2)}\end{bmatrix} &= \begin{bmatrix}\sigma_1\boldsymbol{E}-\boldsymbol{A} & & \\ & \ddots & \\ & & \sigma_r\boldsymbol{E}-\boldsymbol{A}\end{bmatrix}^{-1}\sum_{j=1}^{m}\begin{bmatrix}u_{v,j}^{(1,1)}\boldsymbol{N}_j & \cdots & u_{v,j}^{(1,r)}\boldsymbol{N}_j\\ \vdots & \ddots & \vdots\\ u_{v,j}^{(r,1)}\boldsymbol{N}_j & \cdots & u_{v,j}^{(r,r)}\boldsymbol{N}_j\end{bmatrix}\begin{bmatrix}\boldsymbol{v}_1^{(1)}\\ \vdots\\ \boldsymbol{v}_r^{(k)}\end{bmatrix}\\ &= \left(\boldsymbol{S}_v^{\mathsf{T}}\otimes\boldsymbol{E}-\mathbf{I}_r\otimes\boldsymbol{A}\right)^{-1}\sum_{j=1}^{m}(\boldsymbol{U}_v\otimes\boldsymbol{N}_j)\operatorname{vec}\left(\boldsymbol{V}^{(1)}\right)\end{aligned}$$

and remembering Definition 3.2 one can clearly see that $\boldsymbol{V}^{(2)}$ solves

$$\boldsymbol{E}\,\boldsymbol{V}^{(2)}\,\boldsymbol{S}_v-\boldsymbol{A}\,\boldsymbol{V}^{(2)}=\sum_{j=1}^{m}\boldsymbol{N}_j\,\boldsymbol{V}^{(1)}\,\boldsymbol{U}_{v,j}^{\mathsf{T}}.$$

We can continue this until the $k$-th subsystem where each column $\boldsymbol{v}_i^{(k)}$ of $\boldsymbol{V}^{(k)}$ depends on the columns of the previous projection matrix. Hence, we can write [8]

$$\begin{aligned}\boldsymbol{v}_i^{(k)} &= \sum_{j_2=1}^{m}\cdots\sum_{j_k=1}^{m}\sum_{l_1=1}^{r}\cdots\sum_{l_{k-1}=1}^{r}\eta_{l_1,\dots,l_{k-1},i}^{j_2,\dots,j_k}(\sigma_i\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{N}_{j_k}\\ &\qquad\times(\sigma_{l_{k-1}}\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{N}_{j_{k-1}}\cdots\boldsymbol{N}_{j_2}(\sigma_{l_1}\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{B}\,\boldsymbol{r}_{l_1}\\ &= \sum_{j_k=1}^{m}\sum_{l_{k-1}=1}^{r}u_{v,j_k}^{(i,l_{k-1})}(\sigma_i\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{N}_{j_k}\boldsymbol{v}_{l_{k-1}}^{(k-1)}. \end{aligned}\tag{4.16}$$

Thus, $\boldsymbol{V}^{(k)}$ solves the following linear *Sylvester* equation

$$\boldsymbol{E}\,\boldsymbol{V}^{(k)}\,\boldsymbol{S}_v-\boldsymbol{A}\,\boldsymbol{V}^{(k)}=\sum_{j=1}^{m}\boldsymbol{N}_j\,\boldsymbol{V}^{(k-1)}\,\boldsymbol{U}_{v,j}^{\mathsf{T}}\quad\text{for } k\geq 2.\tag{4.17}$$

By combining (4.15) and (4.17) and letting $k\to\infty$ we find a bilinear *Sylvester* equation whose solution is $\boldsymbol{V}$

$$\boldsymbol{E}\,\boldsymbol{V}\,\boldsymbol{S}_v-\boldsymbol{A}\,\boldsymbol{V}-\sum_{j=1}^{m}\boldsymbol{N}_j\,\boldsymbol{V}\,\boldsymbol{U}_{v,j}^{\mathsf{T}}=\boldsymbol{B}\,\boldsymbol{R}.$$

With an analog approach we can show that every $\boldsymbol{W}^{(k)}$ solves the following linear *Sylvester* equations

$$\begin{aligned}\boldsymbol{E}^{\mathsf{T}}\,\boldsymbol{W}^{(1)}\,\boldsymbol{S}_w^{\mathsf{T}}-\boldsymbol{A}^{\mathsf{T}}\,\boldsymbol{W}^{(1)}&=\boldsymbol{C}^{\mathsf{T}}\,\boldsymbol{L},\\ \boldsymbol{E}^{\mathsf{T}}\,\boldsymbol{W}^{(k)}\,\boldsymbol{S}_w^{\mathsf{T}}-\boldsymbol{A}^{\mathsf{T}}\,\boldsymbol{W}^{(k)}&=\sum_{j=1}^{m}\boldsymbol{N}_j^{\mathsf{T}}\,\boldsymbol{W}^{(k-1)}\,\boldsymbol{U}_{w,j}^{\mathsf{T}}\quad\text{for } k\geq 2.\end{aligned}$$

These linear *Sylvester* equations can also be combined to

$$\boldsymbol{E}^\mathsf{T}\,\boldsymbol{W}\,\boldsymbol{S}_w^\mathsf{T} - \boldsymbol{A}^\mathsf{T}\,\boldsymbol{W} - \sum_{j=1}^{m} \boldsymbol{N}_j^\mathsf{T}\,\boldsymbol{W}\,\boldsymbol{U}_{w,j}^\mathsf{T} = \boldsymbol{C}^\mathsf{T}\,\boldsymbol{L}.$$

*Remark* 4.6 (Solution Of Bilinear Sylvester Equation)*.* Note, that the bilinear *Sylvester* equations yield also solutions if the *Volterra* series does not converge. In this case, the solution of the bilinear *Sylvester* equation does not correspond to the sum over the solutions of linear the *Sylvester* equations of each subsystem such that $\boldsymbol{V} \neq \sum_{k=1}^{\infty} \boldsymbol{V}^{(k)}$. While recalling the definition of the moment matching conditions and the definition of the regular transfer functions, one can interpret the tangential directions and weights as the inputs of the bilinear system. Hence, the convergence of the *Sylvester* equations is strongly connected to BIBO stability conditions. △

*Remark* 4.7 (Invariance Of Sylvester Equations)*.* It may be interesting to transform the solutions $\boldsymbol{V}$ and $\boldsymbol{W}$ to e.g. obtain orthonormal matrices. This problem can be approached by multiplying a specific transformation matrix $\boldsymbol{T}_v$ and receive $\tilde{\boldsymbol{V}} = \boldsymbol{V}\boldsymbol{T}_v$. To let $\tilde{\boldsymbol{V}}$ be the solution of the same *Sylvester* equations we should adjust the interpolation data as follows

$$\tilde{\boldsymbol{S}}_v = \boldsymbol{T}_v^{-1}\boldsymbol{S}_v\boldsymbol{T}_v, \quad \tilde{\boldsymbol{U}}_{v,j} = \boldsymbol{T}_v^\mathsf{T}\boldsymbol{U}_{v,j}\boldsymbol{T}_v^{-\mathsf{T}}, \qquad \tilde{\boldsymbol{R}} = \boldsymbol{R}\boldsymbol{T}_v.$$

Inserting the transformed interpolation data and the transformed projection matrix in the *Sylvester* equation yields

$$\boldsymbol{E}\,\boldsymbol{V}\boldsymbol{T}_v\,\boldsymbol{T}_v^{-1}\boldsymbol{S}_v\boldsymbol{T}_v - \boldsymbol{A}\,\boldsymbol{V}\boldsymbol{T}_v - \sum_{j=1}^{m} \boldsymbol{N}_j\,\boldsymbol{V}\boldsymbol{T}_v\,\boldsymbol{T}_v^{-1}\boldsymbol{U}_{v,j}^\mathsf{T}\boldsymbol{T}_v = \boldsymbol{B}\,\boldsymbol{R}\boldsymbol{T}_v.$$

One can see $\boldsymbol{T}_v\,\boldsymbol{T}_v^{-1}$ cancels itself which yields

$$\boldsymbol{E}\,\boldsymbol{V}\,\boldsymbol{S}_v\boldsymbol{T}_v - \boldsymbol{A}\,\boldsymbol{V}\boldsymbol{T}_v - \sum_{j=1}^{m} \boldsymbol{N}_j\,\boldsymbol{V}\,\boldsymbol{U}_{v,j}^\mathsf{T}\boldsymbol{T}_v = \boldsymbol{B}\,\boldsymbol{R}\boldsymbol{T}_v.$$

By simply multiplying with $\boldsymbol{T}_v^{-1}$ from the right we obtain the original *Sylvester* equation. The same principle could be applied to the linear *Sylvester* equations for each subsystem as well as to the output *Sylvester* equation. If one transforms the output projection matrix such that $\tilde{\boldsymbol{W}} = \boldsymbol{W}\boldsymbol{T}_w$ one has to transform the interpolation data as follows

$$\tilde{\boldsymbol{S}}_w = \boldsymbol{T}_w^\mathsf{T}\boldsymbol{S}_w\boldsymbol{T}_w^{-\mathsf{T}}, \quad \tilde{\boldsymbol{U}}_{w,j} = \boldsymbol{T}_w^\mathsf{T}\boldsymbol{U}_{w,j}\boldsymbol{T}_w^{-\mathsf{T}}, \qquad \tilde{\boldsymbol{L}} = \boldsymbol{L}\,\boldsymbol{T}_w$$

to solve the same *Sylvester* equation. △

### Computing Projection Matrices

At this point we know the basics about multipoint *Volterra* series interpolation. Hence, we want to find algorithms which allow us to compute the projection matrices. Therefore, we start by giving an *Arnoldi* algorithm to compute each column of $\boldsymbol{V}$ explicitly. After that we concentrate on solving linear *Sylvester* equations and finally we discuss whether it is possible to obtain a projection matrix $\boldsymbol{V}$ which considers all subsystems by solving the bilinear *Sylvester* equation. Note, that we write all algorithms only for the input case

as one can easily obtain the output projection matrix by a dual call of the algorithms (see Remark 4.10).
Obviously, it is not possible to compute each column of $\boldsymbol{V}$ by considering an infinite sum [8]. As we have shown above, it is possible to compute the projection matrix for each subsystem independently and obtain the final projection matrix via summation. We also know that the importance of each additional subsystem decreases. Hence, we truncate the sum over all subsystems and mark the number of considered subsystems with the truncation index $N$. It follows that

$$\boldsymbol{V} \approx \sum_{k=1}^{N} \boldsymbol{V}^k.$$

Applying this to the formula for each column $\boldsymbol{v}_i$ of $\boldsymbol{V}$ yields

$$\boldsymbol{v}_i \approx \sum_{k=1}^{N} \boldsymbol{v}_i^{(k)}$$

where

$$\begin{aligned} \boldsymbol{v}_i^{(1)} &= (\sigma_i \boldsymbol{E} - \boldsymbol{A})^{-1} \boldsymbol{B} \boldsymbol{r}_i \\ \boldsymbol{v}_i^{(k)} &= \sum_{j=1}^{m} \sum_{l=1}^{r} u_{v,j}^{(i,l)} (\sigma_i \boldsymbol{E} - \boldsymbol{A})^{-1} \boldsymbol{N}_j \boldsymbol{v}_l^{(k-1)} \quad \text{for } k \geq 2. \end{aligned} \tag{4.18}$$

With this we can give the so-called *Arnoldi* algorithm 4.1 to compute an approximation of $\boldsymbol{V}$. Note that all following algorithms are written with MATLAB® syntax. Especially

**Algorithm 4.1 :** Efficient Arnoldi Algorithm For Computing Projection Matrices

```
Data : bilinear system $\boldsymbol{\zeta}$, interpolation points $\sigma_1, \dots, \sigma_r$, interpolation weights $\boldsymbol{U}_v$,
       tangential directions $\boldsymbol{R}$, truncation index $N$
Result : approximated projection matrix $\boldsymbol{V}$
1  for $i = 1 : r$ do
2      $[\boldsymbol{L}_i, \boldsymbol{U}_i] = \mathtt{lu}(\sigma_i \boldsymbol{E} - \boldsymbol{A})$ ;        ▷ Compute LU factors for each shift
3      $\boldsymbol{V}_{\text{old}}(:, i) = \boldsymbol{U}_i \backslash (\boldsymbol{L}_i \backslash (\boldsymbol{B}\boldsymbol{R}(:, i)))$ ;        ▷ Compute $\boldsymbol{V}^{(1)}$
4  $\boldsymbol{V} = \boldsymbol{V}_{\text{old}}$;
5  for $k = 2 : N$ do
6      for $i = 1 : r$ do
7          $\boldsymbol{v}_{\text{temp}} = \boldsymbol{0}$;
8          for $l = 1 : r$ do
9              for $j = 1 : m$ do
10                 $\boldsymbol{v}_{\text{temp}} = \boldsymbol{v}_{\text{temp}} + \boldsymbol{U}_{v,j}(i, l) \boldsymbol{N}_j \boldsymbol{V}_{\text{old}}(:, l)$ ;   ▷ $\boldsymbol{v}_{\texttt{temp}} = \sum_{j=1}^{m} u_{v,j}^{(i,l)} \boldsymbol{N}_j \boldsymbol{v}_l^{(k-1)}$
11         $\boldsymbol{V}_{\text{new}}(:, i) = \boldsymbol{U}_i \backslash (\boldsymbol{L}_i \backslash \boldsymbol{v}_{\text{temp}})$ ;        ▷ Compute $\boldsymbol{V}^{(k>1)}$
12     $\boldsymbol{V}_{\text{old}} = \boldsymbol{V}_{\text{new}}$;
13     $\boldsymbol{V} = \boldsymbol{V} + \boldsymbol{V}_{\text{new}}$;
```

for large-scale systems we recommend a computation of LU-factors since we have to

solve $N$-times for each $(\sigma_i \boldsymbol{E} - \boldsymbol{A})$. Also crucially for performance is evaluating the sums before solving the systems of equations as seen in Algorithm 4.1 Line 10. Otherwise one would solve the system of equations $ijl$-times instead of $i$-times.
Another option to approximate $\boldsymbol{V}$ is via using existing routines for solving matrix equations to solve the linear *Sylvester* equations for each subsystem. Recalling the truncation of the *Volterra* series yields

$$\boldsymbol{V} \approx \sum_{k=1}^{N} \boldsymbol{V}^{(k)}$$

where

$$\boldsymbol{V}^{(1)} \text{ is the solution of } \boldsymbol{E}\,\boldsymbol{V}^{(1)}\,\boldsymbol{S}_v - \boldsymbol{A}\,\boldsymbol{V}^{(1)} = \boldsymbol{B}\,\boldsymbol{R},$$
$$\boldsymbol{V}^{(k)} \text{ is the solution of } \boldsymbol{E}\,\boldsymbol{V}^{(k)}\,\boldsymbol{S}_v - \boldsymbol{A}\,\boldsymbol{V}^{(k)} = \sum_{j=1}^{m} \boldsymbol{N}_j\,\boldsymbol{V}^{(k-1)}\,\boldsymbol{U}_{v,j}^{\mathsf{T}} \quad \text{for } k \geq 2.$$

In Algorithm 4.2 we use the MATLAB® function `sylvester(`$\boldsymbol{A}$`, `$\boldsymbol{B}$`, `$\boldsymbol{C}$`)` which computes the solution $\boldsymbol{X}$ of the *Sylvester* equation $\boldsymbol{AX} + \boldsymbol{XB} - \boldsymbol{C} = \boldsymbol{0}$. Unfortunately this function does not support our type of *Sylvester* equation. Hence, we have to resolve $\boldsymbol{E}$ by inverting it, which might not be possible especially for large systems. Compared

**Algorithm 4.2 :** Computing Projection Matrices With Sylvester Equation Solver

```
Data : bilinear system $\boldsymbol{\zeta}$, interpolation points $\sigma_1, \dots, \sigma_r$, interpolation weights $\boldsymbol{U}_v$,
       tangential directions $\boldsymbol{R}$, truncation index $N$
Result : approximated projection matrix $\boldsymbol{V}$
1  for $j = 1 : m$ do
2      $\boldsymbol{N}_j = \boldsymbol{E}^{-1}\boldsymbol{N}_j$;                                  ▷ Resolve $\boldsymbol{E}$
3  $\boldsymbol{A} = \boldsymbol{E}^{-1}\boldsymbol{A}$;
4  $\boldsymbol{B} = \boldsymbol{E}^{-1}\boldsymbol{B}$ ;                                       ▷ Resolve $\boldsymbol{E}$
5  for $i = 1 : r$ do
6      $\boldsymbol{V}_{\text{old}}(:, i)$ = sylvester($-\boldsymbol{A}, \boldsymbol{S}_v, \boldsymbol{BR}$) ;          ▷ Compute $\boldsymbol{V}^{(1)}$
7  $\boldsymbol{V} = \boldsymbol{V}_{\text{old}}$;
8  for $k = 2 : N$ do
9      $\boldsymbol{V}_{\text{old, sum}} = \boldsymbol{0}$;
10     for $j = 1 : m$ do
11         $\boldsymbol{V}_{\text{old, sum}} = \boldsymbol{N}_j \boldsymbol{V}_{\text{old}} \boldsymbol{U}_{v,j}^{\mathsf{T}}$;      ▷ $\boldsymbol{V}_{\text{old, sum}} = \sum_{j=1}^{m} \boldsymbol{N}_j\,\boldsymbol{V}^{(k-1)}\,\boldsymbol{U}_{v,j}^{\mathsf{T}}$
12     $\boldsymbol{V}_{\text{new}}$ = sylvester($-\boldsymbol{A}, \boldsymbol{S}_v, \boldsymbol{V}_{\text{old, sum}}$) ;     ▷ Compute $\boldsymbol{V}^{(k>1)}$
13     $\boldsymbol{V}_{\text{old}} = \boldsymbol{V}_{\text{new}}$;
14     $\boldsymbol{V} = \boldsymbol{V} + \boldsymbol{V}_{\text{new}}$;
```

to the computation of each column independently, solving linear *Sylvester* equations does not require the diagonal structure of $\boldsymbol{S}_v$ or rather $\boldsymbol{S}_w$. Consequently, it is possible to apply transformations to the projection matrices after every iteration step or rather before using the algorithm.
The only way to consider all subsystems is by vectorizing the whole bilinear *Sylvester*

equation as seen for the *Lyapunov* equations and solve the system of equations. Hence, we could compute $\boldsymbol{V}$ by solving

$$\operatorname{vec}(\boldsymbol{V}) = \left(\boldsymbol{S}_v^\mathsf{T} \otimes \boldsymbol{E} - \mathbf{I}_r \otimes \boldsymbol{A} - \sum_{j=1}^{m} \boldsymbol{U}_{v,j} \otimes \boldsymbol{N}_j\right)^{-1} \operatorname{vec}(\boldsymbol{BR})$$

and reshaping $\operatorname{vec}(\boldsymbol{V}) \in \mathbb{R}^{nr\times 1}$ into $\boldsymbol{V} \in \mathbb{R}^{n\times r}$. Similar can be applied to the output *Sylvester* equation and we obtain $\boldsymbol{W}$ by solving

$$\operatorname{vec}(\boldsymbol{W}) = \left(\boldsymbol{S}_w \otimes \boldsymbol{E}^\mathsf{T} - \mathbf{I}_r \otimes \boldsymbol{A}^\mathsf{T} - \sum_{j=1}^{m} \boldsymbol{U}_{w,j} \otimes \boldsymbol{N}_j^\mathsf{T}\right)^{-1} \operatorname{vec}\left(\boldsymbol{C}^\mathsf{T}\boldsymbol{L}\right)$$

and reshaping $\operatorname{vec}(\boldsymbol{W}) \in \mathbb{R}^{nr\times 1}$ into $\boldsymbol{W} \in \mathbb{R}^{n\times r}$. As mentioned, this approach is only suitable for small-scale systems since we have to solve a system of equations in $\mathbb{R}^{nr\times nr}$. The biggest upside of this approach is that it is the only way which does not require a truncation and it therefore matches the whole *Volterra* series.

*Remark* 4.8 (Solving Bilinear Sylvester Equations). [8] Sometimes it might not be possible to solve the vectorized bilinear *Sylvester* equations due to the increasing dimensions caused by the vectorization and *Kronecker* products. Hence, one should consider generalizing low rank solvers for linear *Sylvester* equations to bilinear *Sylvester* equations. For more insight we refer to [3]. △

*Remark* 4.9 (Hermite Volterra Series Interpolation). Using equal shifts and weights to compute the input and the output projection matrix is called *Hermite* interpolation. For linear subsystem interpolation (which is included in the *Volterra* series framework if one only considers the first subsystem) this yields a special feature. Constructing $\boldsymbol{V}$ and $\boldsymbol{W}$ to match

$$\begin{aligned}\boldsymbol{G}(\sigma_i)\boldsymbol{r}_i &= \boldsymbol{G}_\mathrm{r}(\sigma_i)\boldsymbol{r}_i,\\ \boldsymbol{l}_i^\mathsf{T}\boldsymbol{G}(\sigma_i) &= \boldsymbol{l}_i^\mathsf{T}\boldsymbol{G}_\mathrm{r}(\sigma_i)\end{aligned}$$

accordingly also matches

$$\boldsymbol{l}_i^\mathsf{T}\boldsymbol{G}'(\sigma_i)\boldsymbol{r}_i = \boldsymbol{l}_i^\mathsf{T}\boldsymbol{G}_\mathrm{r}'(\sigma_i)\boldsymbol{r}_i.$$

Similar applies for bilinear systems. Hence, *Hermite* interpolation yields stronger moment matching conditions. △

*Remark* 4.10 (Computing Output Projection Matrix). As mentioned, it is possible to compute the output projection matrix with a dual call of the presented algorithms. Hence, we briefly discuss what the dual inputs look like. To compute the projection matrix $\boldsymbol{W}$ we have to use

$$\begin{aligned}&\boldsymbol{E}^\mathsf{T} \text{ for } \boldsymbol{E}, \quad \boldsymbol{A}^\mathsf{T} \text{ for } \boldsymbol{A}, \quad \boldsymbol{C}^\mathsf{T} \text{ for } \boldsymbol{B}, \quad \boldsymbol{N}_j^\mathsf{T} \text{ for } \boldsymbol{N}_j,\\ &\boldsymbol{U}_{w,j} \text{ for } \boldsymbol{U}_{v,j}, \quad \boldsymbol{L} \text{ for } \boldsymbol{R}, \quad \boldsymbol{S}_w^\mathsf{T} \text{ for } \boldsymbol{S}_v.\end{aligned}$$

△

*Remark* 4.11 (Increase Stability Of Arnoldi Algorithm). Algorithm 4.1 computes the projection matrices of each subsystem by use of the projection matrix of the previous subsystem. This dependency potentially yields big projection matrices for a badly conditioned bilinear system. To prevent that, one should consider an abort criterion which

prevents the projection matrices grow above a certain threshold. Note, that this threshold has to be less than your operating systems working precision.
In case of equal shifts for input and output interpolation, one might want to increase the efficiency of the algorithm by reusing the LU-factors for computation of the output projection matrices by transposing them. Our experiences have shown that this yields a small error. Since we solve the linear system of equations $N$ times, we make this error $N$ times and consequently it could grow significantly. With our benchmarks we were not able to achieve plausible results by reusing the LU-factors for computation of the output projection matrix. △

### 4.3.2 Multimoment Volterra Series Interpolation

Our goal is to extend the multipoint *Volterra* series interpolation to be able to match multiple high-order moments. Consequently, we call this framework multimoment *Volterra* series interpolation. For the sake of understanding we first discuss only the input case. Note that the following requires complicated indexes which we sometimes define through functions. This is motivated by later finding a *Sylvester* equation representation. Within this method we want to fulfill the following moment matching condition

$$\sum_{k=1}^{\infty}\sum_{j_2=1}^{m}\cdots\sum_{j_k=1}^{m}\sum_{l_1=1}^{\tilde{r}}\cdots\sum_{l_{k-1}=1}^{\tilde{r}}\underbrace{\sum_{i_1=0}^{n_{\tilde{\sigma}_{l_1}}-1}\cdots\sum_{i_{k-1}=0}^{n_{\tilde{\sigma}_{l_{k-1}}}-1}\sum_{i_k=0}^{n_{\tilde{\sigma}_i}-1}}_{\substack{\text{additional sums}\\ \text{for multimoments}}}\eta^{j_2,\dots,j_k}_{\substack{l_1,\dots,l_{k-1},i\\ i_1,\dots,i_{k-1}}}\boldsymbol{M}^{(j_2,\dots,j_k)}_{k,(i_1,\cdots,i_k)}(\tilde{\sigma}_{l_1},\dots,\tilde{\sigma}_{l_{k-1}},\sigma_i)\,\boldsymbol{r}_{r_{\text{col}}(l_1)}$$

$$\equiv$$

$$\sum_{k=1}^{\infty}\sum_{j_2=1}^{m}\cdots\sum_{j_k=1}^{m}\sum_{l_1=1}^{\tilde{r}}\cdots\sum_{l_{k-1}=1}^{\tilde{r}}\underbrace{\sum_{i_1=0}^{n_{\tilde{\sigma}_{l_1}}-1}\cdots\sum_{i_{k-1}=0}^{n_{\tilde{\sigma}_{l_{k-1}}}-1}\sum_{i_k=0}^{n_{\tilde{\sigma}_i}-1}}_{\substack{\text{additional sums}\\ \text{for multimoments}}}\eta^{j_2,\dots,j_k}_{\substack{l_1,\dots,l_{k-1},i\\ i_1,\dots,i_{k-1}}}\boldsymbol{M}^{(j_2,\dots,j_k)}_{k,\text{r},(i_1,\cdots,i_k)}(\tilde{\sigma}_{l_1},\dots,\tilde{\sigma}_{l_{k-1}},\sigma_i)\,\boldsymbol{r}_{r_{\text{col}}(l_1)}$$

for each $i = 1,\dots,r$ where

$$r_{\text{col}}(l) = 1 + \sum_{q=1}^{l-1} n_{\tilde{\sigma}_q}.$$

Hereby a tilde-shift $\tilde{\sigma}_i$ denotes the $i$-th unique shift, every $n_{\tilde{\sigma}_i}$ denotes the multiplicity of a certain shift $\tilde{\sigma}_i$ and $\tilde{r}$ denotes the number of unique shifts. Let us assume for now that the weights are arbitrary. We define them later again through entries of a matrix $\boldsymbol{U}_{v,j}$. Since finding a *Sylvester* equation requires $\boldsymbol{U}_{v,j}$ and $\boldsymbol{R}$ to have a certain structure, we are forced to define the weights with the indexes $i_1,\dots,i_{k-1}$ as well as the corresponding tangential directions with the function $r_{\text{col}}(l)$.

*Example* 4.5 (Indexes Of Multimoment *Volterra* Series Interpolation). Let us assume that we want to interpolate with the following $r = 5$ shifts $\sigma_1 = \sigma_2 \neq \sigma_3 = \sigma_4 = \sigma_5$. Since $\sigma_1 = \sigma_2$ and $\sigma_3 = \sigma_4 = \sigma_5$ we have $\tilde{r} = 2$ unique shifts $\tilde{\sigma}_1$ and $\tilde{\sigma}_2$. Hence, the multiplicity of the first shift $\tilde{\sigma}_1$ is $n_{\tilde{\sigma}_1} = 2$ and the multiplicity of the second shift $\tilde{\sigma}_2$ is $n_{\tilde{\sigma}_2} = 3$. Consequently, we have to match the 0-th and the 1-st moment for $\tilde{\sigma}_1$ and the 0-th, the 1-st and the 2-nd moment for $\tilde{\sigma}_2$. △

To enlighten the moment matching condition even more we write all moments with corresponding tangential directions of the first subsystem explicitly

$$\sum_{i_1=0}^{n_{\tilde{\sigma}_i}-1} \boldsymbol{M}_{1,(i_1)}(\sigma_i)\,\boldsymbol{r}_{\tilde{i}} = \sum_{i_1=0}^{n_{\tilde{\sigma}_i}-1} (-1)^{i_1}\,\boldsymbol{C}\left[(\sigma_i\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E}\right]^{i_1}(\sigma_i\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{B}\boldsymbol{r}_{\tilde{i}}.$$

Note that we later define $\tilde{i}$ in (4.19), again corresponding to the required structure of $\boldsymbol{R}$. One can clearly determine that we obtain all moments from the 0-th to the $(n_{\tilde{\sigma}_i}-1)$-th for a multiplicity $n_{\tilde{\sigma}_i}$ of the shift $\tilde{\sigma}_i$. Let us also write explicitly all moments for the second subsystem

$$\sum_{j_2=1}^{m}\sum_{l_1=1}^{\tilde{r}}\sum_{i_1=0}^{n_{\tilde{\sigma}_{l_1}}-1}\sum_{i_2=0}^{n_{\tilde{\sigma}_i}-1} \eta^{j_2,\dots,j_k}_{\substack{l_1,2\\ i_1}}\,\boldsymbol{M}^{(j_2)}_{2,(i_1,i_2)}(\tilde{\sigma}_{l_1},\sigma_i)\,\boldsymbol{r}_{r_{\text{col}}(l_1)} =$$

$$= \sum_{j_2=1}^{m}\sum_{l_1=1}^{\tilde{r}}\sum_{i_1=0}^{n_{\tilde{\sigma}_{l_1}}-1}\sum_{i_2=0}^{n_{\tilde{\sigma}_i}-1} \eta^{j_2}_{\substack{l_1,i\\ i_1}}\,(-1)^{i_1+i_2}\,\boldsymbol{C}\left[(\sigma_i\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E}\right]^{i_2}$$
$$\times(\sigma_i\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{N}_{j_2}\left[(\tilde{\sigma}_{l_1}\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E}\right]^{i_1}(\tilde{\sigma}_{l_1}\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{B}\boldsymbol{r}_{r_{\text{col}}(l_1)}.$$

By resorting the terms we can see that we combine all moments of the first subsystem with all additional moments of the second subsystem

$$= \sum_{j_2=1}^{m}\sum_{i_2=0}^{n_{\tilde{\sigma}_i}-1}(-1)^{i_2}\,\boldsymbol{C}\left[(\sigma_i\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E}\right]^{i_2}(\sigma_i\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{N}_{j_2}$$
$$\times\underbrace{\left(\sum_{l_1=1}^{\tilde{r}}\sum_{i_1=0}^{n_{\tilde{\sigma}_{l_1}}-1}\eta^{j_2}_{\substack{l_1,i\\ i_1}}(-1)^{i_1}\left[(\tilde{\sigma}_{l_1}\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E}\right]^{i_1}(\tilde{\sigma}_{l_1}\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{B}\boldsymbol{r}_{r_{\text{col}}(l_1)}\right)}_{\text{weighted combinations of moments of the first subsystem}}.$$

Since the moment matching condition is different form the multipoint *Volterra* series interpolation framework, we have to construct the projection matrix $\boldsymbol{V}$ with a slightly different *Krylov* subspace

$$\boldsymbol{V}\subseteq\mathcal{K}_r\left((\sigma_i\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E},\boldsymbol{v}_i\right)\in\mathbb{R}^{n\times r}$$

where

$$\boldsymbol{v}_i=\sum_{k=1}^{\infty}\sum_{j_2=1}^{m}\cdots\sum_{j_k=1}^{m}\sum_{l_1=1}^{\tilde{r}}\cdots\sum_{l_{k-1}=1}^{\tilde{r}}\sum_{i_1=0}^{n_{\tilde{\sigma}_{l_1}}-1}\cdots\sum_{i_{k-1}=0}^{n_{\tilde{\sigma}_{l_{k-1}}}-1}\eta^{j_2,\dots,j_k}_{\substack{l_1,\dots,l_{k-1},i\\ i_1,\dots,i_{k-1}}}(\sigma_i\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{N}_{j_k}$$
$$\times\left[(\tilde{\sigma}_{l_{k-1}}\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E}\right]^{i_{k-1}}(\tilde{\sigma}_{l_{k-1}}\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{N}_{j_{k-1}}\cdots$$
$$\times\boldsymbol{N}_{j_2}\left[(\tilde{\sigma}_{l_1}\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E}\right]^{i_1}(\tilde{\sigma}_{l_1}\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{B}\,\boldsymbol{r}_{r_{\text{col}}(l_1)}.$$

Taking a closer look at the definition of the above *Krylov* subspace reveals that it is the generalization of the multipoint *Krylov* subspace. The projection matrix $\boldsymbol{V}$ for

$$\underbrace{(\sigma_1=\sigma_2=\ldots=\sigma_{n_{\tilde{\sigma}_1}})}_{\tilde{\sigma}_1\text{ with multiplicity }n_{\tilde{\sigma}_1}}\neq\cdots\quad\cdots\neq\underbrace{(\sigma_{1+n_{\sigma_{\tilde{1}}}+\cdots+n_{\sigma_{\tilde{r}-1}}}=\ldots=\sigma_{n_{\tilde{\sigma}_1}+\cdots+n_{\tilde{\sigma}_r}})}_{\tilde{\sigma}_{\tilde{r}}\text{ with multiplicity }n_{\tilde{\sigma}_{\tilde{r}}}}$$

is given by

$$\boldsymbol{V} = \Big[\boldsymbol{v}_1\,,\ \ (\sigma_2\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E}\,\boldsymbol{v}_2,\ \ \cdots,\ \ \left[(\sigma_{n_{\tilde{\sigma}_1}}\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E}\right]^{n_{\tilde{\sigma}_1}-1}\boldsymbol{v}_{n_{\tilde{\sigma}_1}},\ \ \cdots$$
$$\cdots,\ \ \boldsymbol{v}_{1+n_{\tilde{\sigma}_1}+\cdots+n_{\tilde{\sigma}_{\tilde{r}-1}}},\ \ (\sigma_{2+n_{\tilde{\sigma}_1}+\cdots+n_{\tilde{\sigma}_{\tilde{r}-1}}}\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E}\,\boldsymbol{v}_{2+n_{\tilde{\sigma}_1}+\cdots+n_{\tilde{\sigma}_{\tilde{r}-1}}},$$
$$\cdots,\ \ \left[(\sigma_{n_{\tilde{\sigma}_1}+\cdots+n_{\tilde{\sigma}_{\tilde{r}}}}\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E}\right]^{n_{\tilde{\sigma}_{\tilde{r}}}-1}\boldsymbol{v}_{n_{\tilde{\sigma}_1}+\cdots+n_{\tilde{\sigma}_{\tilde{r}}}}\Big] \in \mathbb{R}^{n\times r}$$

where $r = n_{\tilde{\sigma}_1} + \cdots + n_{\tilde{\sigma}_{\tilde{r}}}$.

*Example* 4.6 (Multimoment Projection Matrix). Let us assume that we want to reduce a SISO bilinear system with $r = 3$ shifts $\sigma_1 = \sigma_2 \neq \sigma_3$. With our choice of shifts we get $\tilde{r} = 2$ unique shifts $\tilde{\sigma}_1$ with multiplicity $n_{\tilde{\sigma}_1} = 2$ and $\tilde{\sigma}_2$ with multiplicity $n_{\tilde{\sigma}_2} = 1$. Let us consider $N = 2$ subsystems. Hence, we have to construct the first column of the projection matrix $\boldsymbol{V}$ as follows

$$\begin{aligned}
\boldsymbol{v}_1 &= \sum_{k=1}^{2}\underbrace{\sum_{l_1=1}^{2}\sum_{i_1=0}^{n_{\tilde{\sigma}_{l_1}}-1}\eta^{j_2}_{\underset{i_1}{l_1,1}}(\sigma_1\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{N}\left[(\tilde{\sigma}_{l_1}\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E}\right]^{i_1}}_{=1 \text{ for } k=1}(\tilde{\sigma}_{l_1}\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{B}\,\boldsymbol{r}_{r_{\text{col}}(l_1)}\\
&= (\sigma_1\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{B}\,\boldsymbol{r}_{r_{\text{col}}(1)}\\
&\quad + \sum_{i_1=0}^{1}\eta_{\underset{i_1}{1,1}}(\sigma_1\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{N}\left[(\tilde{\sigma}_1\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E}\right]^{i_1}(\tilde{\sigma}_1\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{B}\,\boldsymbol{r}_{r_{\text{col}}(1)}\\
&\quad + \eta_{\underset{0}{2,1}}(\sigma_1\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{N}(\tilde{\sigma}_2\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{B}\,\boldsymbol{r}_{r_{\text{col}}(2)}.
\end{aligned}$$

Since the second column of $\boldsymbol{V}$ corresponds to the second interpolation point $\sigma_2$ which is equal to the shift corresponding to the previous column, we have to construct $\boldsymbol{v}_2$ with a high-order moment such that

$$\begin{aligned}
\boldsymbol{v}_2 &= \underbrace{(\sigma_2\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E}}_{\text{high-order moment}}\sum_{k=1}^{2}\sum_{l_1=1}^{2}\sum_{i_1=0}^{n_{\tilde{\sigma}_{l_1}}-1}\eta^{j_2}_{\underset{i_1}{l_1,2}}(\sigma_2\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{N}\\
&\qquad\times\left[(\tilde{\sigma}_{l_1}\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E}\right]^{i_1}(\tilde{\sigma}_{l_1}\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{B}\,\boldsymbol{r}_{r_{\text{col}}(l_1)}\\
&= (\sigma_2\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E}(\sigma_2\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{B}\,\boldsymbol{r}_{r_{\text{col}}(1)}\\
&\quad + (\sigma_2\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E}\sum_{i_1=0}^{1}\eta_{\underset{i_1}{1,2}}(\sigma_2\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{N}\\
&\qquad\times\left[(\tilde{\sigma}_1\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E}\right]^{i_1}(\tilde{\sigma}_1\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{B}\,\boldsymbol{r}_{r_{\text{col}}(1)}\\
&\quad + (\sigma_2\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E}\,\eta_{\underset{0}{2,2}}(\tilde{\sigma}_2\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{N}(\tilde{\sigma}_2\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{B}\,\boldsymbol{r}_{r_{\text{col}}(2)}.
\end{aligned}$$

Finally, we can compute the third column corresponding to $\sigma_3 = \tilde{\sigma}_2$ as follows

$$\begin{aligned}\boldsymbol{v}_3 &= \sum_{k=1}^{2}\sum_{l_1=1}^{2}\sum_{i_1=0}^{n_{\tilde{\sigma}_{l_1}}-1} \eta^{j_2}_{\substack{l_1,3\\ i_1}} (\sigma_3\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{N}\left[(\tilde{\sigma}_{l_1}\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E}\right]^{i_1}(\tilde{\sigma}_{l_1}\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{B}\,\boldsymbol{r}_{r_{\mathrm{col}}(l_1)}\\ &= (\sigma_3\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{B}\,\boldsymbol{r}_{r_{\mathrm{col}}(2)}\\ &\quad+\sum_{i_1=0}^{n_{\tilde{\sigma}_1}-1}\eta_{\substack{1,3\\ i_1}}(\sigma_3\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{N}\left[(\tilde{\sigma}_1\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E}\right]^{i_1}(\tilde{\sigma}_1\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{B}\,\boldsymbol{r}_{r_{\mathrm{col}}(1)}\\ &\quad+\eta_{\substack{2,3\\ 0}}(\sigma_3\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{N}(\tilde{\sigma}_2\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{B}\,\boldsymbol{r}_{r_{\mathrm{col}}(2)}.\end{aligned}$$

△

For sake of completeness we introduce the interpolation condition with output tangential directions

$$\sum_{k=1}^{\infty}\sum_{j_2=1}^{m}\cdots\sum_{j_k=1}^{m}\sum_{l_1=1}^{\tilde{r}}\cdots\sum_{l_{k-1}=1}^{\tilde{r}}\underbrace{\sum_{i_1=0}^{n_{\tilde{\mu}_{l_1}}-1}\cdots\sum_{i_{k-1}=0}^{n_{\tilde{\mu}_{l_{k-1}}}-1}\sum_{i_k=0}^{n_{\tilde{\mu}_i}-1}}_{\substack{\text{additional sums}\\ \text{for multimoments}}}\vartheta^{j_2,\ldots,j_k}_{\substack{l_1,\ldots,l_{k-1},i\\ i_1,\ldots,i_{k-1}}}\,\boldsymbol{l}^{\mathsf{T}}_{r_{\mathrm{col}}(l_1)}\boldsymbol{M}^{(j_2,\ldots,j_k)}_{k,(i_1,\cdots,i_k)}(\mu_i,\tilde{\mu}_{l_{k-1}},\ldots,\tilde{\mu}_{l_1})$$

$$\equiv$$

$$\sum_{k=1}^{\infty}\sum_{j_2=1}^{m}\cdots\sum_{j_k=1}^{m}\sum_{l_1=1}^{\tilde{r}}\cdots\sum_{l_{k-1}=1}^{\tilde{r}}\underbrace{\sum_{i_1=0}^{n_{\tilde{\mu}_{l_1}}-1}\cdots\sum_{i_{k-1}=0}^{n_{\tilde{\mu}_{l_{k-1}}}-1}\sum_{i_k=0}^{n_{\tilde{\mu}_i}-1}}_{\substack{\text{additional sums}\\ \text{for multimoments}}}\vartheta^{j_2,\ldots,j_k}_{\substack{l_1,\ldots,l_{k-1},i\\ i_1,\ldots,i_{k-1}}}\,\boldsymbol{l}^{\mathsf{T}}_{r_{\mathrm{col}}(l_1)}\boldsymbol{M}^{(j_2,\ldots,j_k)}_{\mathrm{r},k,(i_1,\cdots,i_k)}(\mu_i,\tilde{\mu}_{l_{k-1}},\ldots,\tilde{\mu}_{l_1})\,.$$

which we can fulfill if we build the projection matrix $\boldsymbol{W}$ with the following output *Krylov* subspace

$$\boldsymbol{W}\subseteq\mathcal{K}_r\left((\mu_i\boldsymbol{E}-\boldsymbol{A})^{-\mathsf{T}}\boldsymbol{E}^{\mathsf{T}},\boldsymbol{w}_i\right)\in\mathbb{R}^{n\times r}$$

where

$$\begin{aligned}\boldsymbol{w}_i=\sum_{k=1}^{\infty}\sum_{j_2=1}^{m}\cdots\sum_{j_k=1}^{m}\sum_{l_1=1}^{\tilde{r}}\cdots\sum_{l_{k-1}=1}^{\tilde{r}}\sum_{i_1=0}^{n_{\tilde{\mu}_{l_1}}-1}\cdots\sum_{i_{k-1}=0}^{n_{\tilde{\mu}_{l_{k-1}}}-1}\vartheta^{j_2,\ldots,j_k}_{\substack{l_1,\ldots,l_{k-1},i\\ i_1,\ldots,i_{k-1}}}(\mu_i\boldsymbol{E}-\boldsymbol{A})^{-\mathsf{T}}\boldsymbol{N}^{\mathsf{T}}_{j_k}\\ \times\left[(\tilde{\mu}_{l_{k-1}}\boldsymbol{E}-\boldsymbol{A})^{-\mathsf{T}}\boldsymbol{E}^{\mathsf{T}}\right]^{i_{k-1}}(\tilde{\mu}_{l_{k-1}}\boldsymbol{E}-\boldsymbol{A})^{-\mathsf{T}}\boldsymbol{N}^{\mathsf{T}}_{j_{k-1}}\cdots\\ \times\boldsymbol{N}^{\mathsf{T}}_{j_2}\left[(\tilde{\mu}_{l_1}\boldsymbol{E}-\boldsymbol{A})^{-\mathsf{T}}\boldsymbol{E}^{\mathsf{T}}\right]^{i_1}(\tilde{\mu}_{l_1}\boldsymbol{E}-\boldsymbol{A})^{-\mathsf{T}}\boldsymbol{C}^{\mathsf{T}}\,\boldsymbol{l}_{r_{\mathrm{col}}(l_1)}.\end{aligned}$$

The complete projection matrix $\boldsymbol{W}$ for

$$\underbrace{(\mu_1=\mu_2=\ldots=\mu_{n_{\mu_{\tilde{1}}}})}_{\mu_{\tilde{1}}\text{ with multiplicity }n_{\mu_{\tilde{1}}}}\neq\cdots\quad\cdots\neq\underbrace{(\mu_{1+n_{\mu_{\tilde{1}}}+\cdots+n_{\mu_{\tilde{r}-1}}}=\ldots=\mu_{n_{\tilde{\mu}_1}+\cdots+n_{\mu_{\tilde{r}}}})}_{\mu_{\tilde{r}}\text{ with multiplicity }n_{\tilde{\mu}_{\tilde{r}}}}$$

is given by

$$\begin{aligned}\boldsymbol{W}=\Big[\boldsymbol{w}_1,\ (\mu_2\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E}\,\boldsymbol{w}_2,\ \cdots,\ \left[(\mu_{n_{\tilde{\mu}_1}}\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E}\right]^{n_{\tilde{\mu}_1}-1}\boldsymbol{w}_{n_{\tilde{\mu}_1}},\ \cdots\\ \cdots\ \boldsymbol{w}_{1+n_{\mu_{\tilde{1}}}+\cdots+n_{\mu_{\tilde{r}-1}}},\ (\mu_{2+n_{\tilde{\mu}_1}+\cdots+n_{\tilde{\mu}_{\tilde{r}-1}}}\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E}\,\boldsymbol{w}_{2+n_{\tilde{\mu}_1}+\cdots+n_{\tilde{\mu}_{\tilde{r}-1}}},\\ \cdots,\ \left[(\mu_{n_{\tilde{\mu}_1}+\cdots+n_{\tilde{\mu}_{\tilde{r}}}}\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E}\right]^{n_{\tilde{\mu}_{\tilde{r}}}-1}\boldsymbol{w}_{n_{\tilde{\mu}_1}+\cdots+n_{\tilde{\mu}_{\tilde{r}}}}\Big]\in\mathbb{R}^{n\times r}\end{aligned}$$

where $r=n_{\tilde{\mu}_1}+\cdots+n_{\tilde{\mu}_{\tilde{r}}}$.

### Sylvester Equations

We now want to derive the projection matrix $\boldsymbol{V}$ by solving the following bilinear *Sylvester* equation

$$\boldsymbol{E}\,\boldsymbol{V}\,\boldsymbol{S}_v - \boldsymbol{A}\,\boldsymbol{V} - \sum_{j=1}^{m} \boldsymbol{N}_j\,\boldsymbol{V}\,\boldsymbol{U}_{v,j}^{\mathsf{T}} = \boldsymbol{B}\,\boldsymbol{R},$$

where $\boldsymbol{U}_{v,j} = \left\{u_{v,j}^{(a,b)}\right\} \in \mathbb{C}^{r\times r}$. And to show accordingly that we can obtain the projection matrix $\boldsymbol{W}$ by solving

$$\boldsymbol{E}^{\mathsf{T}}\,\boldsymbol{W}\,\boldsymbol{S}_w^{\mathsf{T}} - \boldsymbol{A}^{\mathsf{T}}\,\boldsymbol{W} - \sum_{j=1}^{m} \boldsymbol{N}_j^{\mathsf{T}}\,\boldsymbol{W}\,\boldsymbol{U}_{w,j}^{\mathsf{T}} = \boldsymbol{C}^{\mathsf{T}}\,\boldsymbol{L}$$

where $\boldsymbol{U}_{w,j} = \left\{u_{w,j}^{(a,b)}\right\} \in \mathbb{C}^{r\times r}$. As mentioned, we have to determine the structure of $\boldsymbol{S}_v \in \mathbb{C}^{r\times r}$, $\boldsymbol{U}_{v,j}$ and $\boldsymbol{R} \in \mathbb{C}^{m\times r}$ (as well as of $\boldsymbol{S}_w \in \mathbb{C}^{r\times r}$, $\boldsymbol{U}_{w,j}$ and $\boldsymbol{L} \in \mathbb{C}^{m\times r}$) to be able to represent the multimoment *Volterra* series framework as a *Sylvester* equation. Again $u_{v,j}^{(a,b)}$ denotes the entry in the $a$-th row and the $b$-th column of $\boldsymbol{U}_{v,j}$ and analogously $u_{w,j}^{(a,b)}$ denotes the entry in the $a$-th row and the $b$-th column of $\boldsymbol{U}_{w,j}$. As for the multipoint framework, it is necessary to define the weights with entries of the matrices $\boldsymbol{U}_{v,j}$ and $\boldsymbol{U}_{w,j}$. In the following we will discover that $\boldsymbol{U}_{v,j}$ and $\boldsymbol{U}_{w,j}$ have to have a certain structure. Hence, the definitions of the weights get more complex

$$\begin{aligned}
\eta^{j_2,\ldots,j_k}_{\substack{l_1,\ldots,l_{k-1},i\\ i_1,\ldots,i_{k-1}}} &= u_{v,j_k}^{(\tilde{i},u_{\text{col}}(l_{k-1},i_{k-1}))}\, u_{v,j_{k-1}}^{(u_{\text{row}}(l_{k-2}),u_{\text{col}}(l_{k-3},i_{k-3}))} \cdots u_{v,j_2}^{(u_{\text{row}}(l_2),u_{\text{col}}(l_1,i_1))} && \text{for } k>1\\
\vartheta^{j_2,\ldots,j_k}_{\substack{l_1,\ldots,l_{k-1},i\\ i_1,\ldots,i_{k-1}}} &= u_{w,j_k}^{(\tilde{i},u_{\text{col}}(l_{k-1},i_{k-1}))}\, u_{w,j_{k-1}}^{(u_{\text{row}}(l_{k-2}),u_{\text{col}}(l_{k-3},i_{k-3}))} \cdots u_{w,j_2}^{(u_{\text{row}}(l_2),u_{\text{col}}(l_1,i_1))} && \text{for } k>1.
\end{aligned}$$

with

$$\begin{aligned}
\tilde{i} &= 1 && \text{if} \quad i \le n_{\tilde{\sigma}_1},\\
\tilde{i} &= 1 + n_{\tilde{\sigma}_1} && \text{if} \quad n_{\tilde{\sigma}_1} < i \le n_{\tilde{\sigma}_1} + n_{\tilde{\sigma}_2},\\
&\;\;\vdots\\
\tilde{i} &= 1 + n_{\tilde{\sigma}_1} + \ldots + n_{\tilde{\sigma}_{\tilde{r}}} && \text{if} \quad n_{\tilde{\sigma}_1} + \ldots + n_{\tilde{\sigma}_{\tilde{r}-2}} < i \le n_{\tilde{\sigma}_1} + \ldots + n_{\tilde{\sigma}_{\tilde{r}-1}},\\
u_{\text{row}}(l) &= 1 + \sum_{q=1}^{l-1} n_{\tilde{\sigma}_q},\\
u_{\text{col}}(l,i) &= i + 1 + \sum_{q=1}^{l-1} n_{\tilde{\sigma}_q}.
\end{aligned} \tag{4.19}$$

Note that the definition of $\tilde{i}$ also holds throughout the whole chapter. Let us again try to find a linear *Sylvester* equation for each subsystem. Therefore, we once more write $\boldsymbol{V}$ as a series

$$\boldsymbol{V} = \sum_{k=1}^{\infty} \boldsymbol{V}^{(k)} = \boldsymbol{V}^{(1)} + \boldsymbol{V}^{(2)} + \boldsymbol{V}^{(3)} + \boldsymbol{V}^{(4)} + \cdots.$$

Starting with the first subsystem, the $i$-th column $\boldsymbol{v}_i^{(1)}$ of $\boldsymbol{V}^{(1)}$ is defined as

$$\boldsymbol{v}_i^{(1)} = \Big[(\sigma_i \boldsymbol{E} - \boldsymbol{A})^{-1}\boldsymbol{E}\Big]^{i_1} (\sigma_i \boldsymbol{E} - \boldsymbol{A})^{-1}\boldsymbol{B}\,\boldsymbol{r}_{\tilde{i}} \tag{4.20}$$

where $i = 1, \dots, r$. $i_1 = 0$ if the previous column was computed with a different shift and $i_1 = q$ if the $q$ previous columns where computed with the same shift. Let us write the matrix $\boldsymbol{V}^{(1)}$ in vectorized form while using the abbreviation $\boldsymbol{A}_{\sigma_i} := (\sigma_i \boldsymbol{E} - \boldsymbol{A})^{-1}$

$$\begin{bmatrix} \boldsymbol{v}_1^{(1)} \\ \\ \\ \\ \vdots \\ \\ \\ \\ \boldsymbol{v}_r^{(1)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}_{\tilde{\sigma}_1} & & & & & & & & \\ -\boldsymbol{E} & \boldsymbol{A}_{\tilde{\sigma}_1} & & & & & & & \\ & \ddots & \ddots & & & & & & \\ & & -\boldsymbol{E} & \boldsymbol{A}_{\tilde{\sigma}_1} & & & & & \\ & & & & \ddots & & & & \\ & & & & & \boldsymbol{A}_{\tilde{\sigma}_{\tilde{r}}} & & & \\ & & & & & -\boldsymbol{E} & \boldsymbol{A}_{\tilde{\sigma}_{\tilde{r}}} & & \\ & & & & & & \ddots & \ddots & \\ & & & & & & & -\boldsymbol{E} & \boldsymbol{A}_{\tilde{\sigma}_{\tilde{r}}} \end{bmatrix}^{-1} (\mathbf{I}_r \otimes \boldsymbol{B}) \begin{bmatrix} \boldsymbol{r}_1 \\ \boldsymbol{0} \\ \vdots \\ \boldsymbol{0} \\ \vdots \\ \boldsymbol{r}_{1+n_{\tilde{\sigma}_1}+\dots+n_{\tilde{\sigma}_{\tilde{r}}}} \\ \boldsymbol{0} \\ \vdots \\ \boldsymbol{0} \end{bmatrix}$$

$$= \left(\boldsymbol{S}_v^{\mathsf{T}} \otimes \boldsymbol{E} - \mathbf{I}_r \otimes \boldsymbol{A}\right)^{-1} (\mathbf{I}_r \otimes \boldsymbol{B})\,\mathrm{vec}\,(\boldsymbol{R})\,.$$

Under the constraint that every column of $\boldsymbol{V}^{(1)}$ corresponding to a high-order moment has null-vectors as tangential directions we are able to determine $\boldsymbol{V}^{(1)}$ by solving the following linear *Sylvester* equation

$$\boldsymbol{E}\,\boldsymbol{V}^{(1)}\,\boldsymbol{S}_v - \boldsymbol{A}\,\boldsymbol{V}^{(1)} = \boldsymbol{B}\,\boldsymbol{R} \tag{4.21}$$

where

$$\boldsymbol{S}_v = \begin{bmatrix} \tilde{\sigma}_1 & -1 & & & & & & & \\ & \tilde{\sigma}_1 & \ddots & & & & & & \\ & & \ddots & -1 & & & & & \\ & & & \tilde{\sigma}_1 & & & & & \\ & & & & \ddots & & & & \\ & & & & & \tilde{\sigma}_{\tilde{r}} & -1 & & \\ & & & & & & \tilde{\sigma}_{\tilde{r}} & \ddots & \\ & & & & & & & \ddots & -1 \\ & & & & & & & & \tilde{\sigma}_{\tilde{r}} \end{bmatrix}, \qquad . \tag{4.22}$$

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{r}_1 & \boldsymbol{0} & \cdots & \boldsymbol{0} & \cdots & \boldsymbol{r}_{1+n_{\tilde{\sigma}_1}+\dots+n_{\tilde{\sigma}_{\tilde{r}}}} & \boldsymbol{0} & \cdots & \boldsymbol{0} \end{bmatrix}$$

For the second subsystem the $i$-th column $\boldsymbol{v}_i^{(2)}$ of $\boldsymbol{V}^{(2)}$ is defined as

$$\begin{aligned}\boldsymbol{v}_i^{(2)} &= \left[(\sigma_i\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E}\right]^{i_2}\sum_{j_2=1}^{m}\sum_{l_1=1}^{\tilde{r}}\sum_{i_1=0}^{n_{\sigma_{l_1}}-1}\eta_{i_1,i}^{j_2}(\sigma_i\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{N}_{j_2}\\ &\qquad\times\left[(\tilde{\sigma}_{l_1}\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E}\right]^{i_1}(\tilde{\sigma}_{l_1}\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{B}\,\boldsymbol{r}_{r_{\mathrm{col}}(l_1)}\\ &= \left[(\sigma_i\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E}\right]^{i_2}\sum_{j_2=1}^{m}\sum_{l_1=1}^{r}u_{v,j_2}^{(\tilde{i},l_1)}(\sigma_i\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{N}_{j_2}\boldsymbol{v}_{l_1}^{(1)}\end{aligned}$$

where $i=1,\dots,r$. Note, that the sum with index $l_1$ changes its upper limit if we compute $\boldsymbol{v}_i^{(2)}$ by using the columns of the the projection matrix of the previous subsystem. Again, $i_2=0$ if the previous column was computed with a different shift and $i_2=q$ if the $q$ previous columns where computed with the same shift. It is only possible to write $\boldsymbol{V}^{(2)}$ as a solution of a linear *Sylvester* equation if all rows of each $\boldsymbol{U}_{v,j}$ which correspond to a higher moment are zero-rows. While keeping this constraint in mind we can write $\boldsymbol{V}^{(2)}$ in its vectorized form still using the abbreviation $\boldsymbol{A}_{\sigma_i} := (\sigma_i\boldsymbol{E}-\boldsymbol{A})^{-1}$

$$\begin{aligned}\begin{bmatrix}\boldsymbol{v}_1^{(2)}\\ \\ \vdots\\ \\ \boldsymbol{v}_r^{(2)}\end{bmatrix} &= \begin{bmatrix}\boldsymbol{A}_{\tilde{\sigma}_1} & & & & & & & \\ -\boldsymbol{E} & \boldsymbol{A}_{\tilde{\sigma}_1} & & & & & & \\ & \ddots & \ddots & & & & & \\ & & -\boldsymbol{E} & \boldsymbol{A}_{\tilde{\sigma}_1} & & & & \\ & & & & \ddots & & & \\ & & & & & \boldsymbol{A}_{\tilde{\sigma}_{\tilde{r}}} & & \\ & & & & & -\boldsymbol{E} & \boldsymbol{A}_{\tilde{\sigma}_{\tilde{r}}} & \\ & & & & & & \ddots & \ddots \\ & & & & & & -\boldsymbol{E} & \boldsymbol{A}_{\tilde{\sigma}_{\tilde{r}}}\end{bmatrix}^{-1}\\ &\quad\times\sum_{j=1}^{m}\begin{bmatrix}u_{v,j}^{(1,1)}\boldsymbol{N}_j & \cdots & \cdots & u_{v,j}^{(1,r)}\boldsymbol{N}_j\\ \boldsymbol{0} & \cdots & \cdots & \boldsymbol{0}\\ \vdots & & & \vdots\\ \boldsymbol{0} & \cdots & \cdots & \boldsymbol{0}\\ & & \cdots & \\ u_{v,j}^{(u_{\mathrm{ind}},1)}\boldsymbol{N}_j & \cdots & \cdots & u_{v,j}^{(u_{\mathrm{ind}},r)}\boldsymbol{N}_j\\ \boldsymbol{0} & \cdots & \cdots & \boldsymbol{0}\\ \vdots & & & \vdots\\ \boldsymbol{0} & \cdots & \cdots & \boldsymbol{0}\end{bmatrix}\begin{bmatrix}\boldsymbol{v}_1^{(1)}\\ \\ \vdots\\ \\ \boldsymbol{v}_r^{(1)}\end{bmatrix}\\ &= \left(\boldsymbol{S}_v^{\mathsf{T}}\otimes\boldsymbol{E}-\mathbf{I}_r\otimes\boldsymbol{A}\right)^{-1}\sum_{j=1}^{m}(\boldsymbol{U}_v\otimes\boldsymbol{N}_j)\,\mathrm{vec}\left(\boldsymbol{V}^{(1)}\right)\end{aligned}$$

where $u_{\text{ind}} = 1 + n_{\tilde{\sigma}_1} + \cdots + n_{\tilde{\sigma}_{\tilde{r}-1}}$. Hence, $\boldsymbol{V}^{(2)}$ solves

$$\boldsymbol{E}\,\boldsymbol{V}^{(2)}\,\boldsymbol{S}_v - \boldsymbol{A}\,\boldsymbol{V}^{(2)} = \sum_{j=1}^{m} \boldsymbol{N}_j\,\boldsymbol{V}^{(1)}\,\boldsymbol{U}_{v,j}^{\mathsf{T}}$$

only, if $\boldsymbol{S}_v$ and each $\boldsymbol{U}_{v,j}$ have following structure

$$\boldsymbol{S}_v = \begin{bmatrix} \tilde{\sigma}_1 & -1 & & & & & & & \\ & \tilde{\sigma}_1 & \ddots & & & & & & \\ & & \ddots & -1 & & & & & \\ & & & \tilde{\sigma}_1 & & & & & \\ & & & & \ddots & & & & \\ & & & & & \tilde{\sigma}_{\tilde{r}} & -1 & & \\ & & & & & & \tilde{\sigma}_{\tilde{r}} & \ddots & \\ & & & & & & & \ddots & -1 \\ & & & & & & & & \tilde{\sigma}_{\tilde{r}} \end{bmatrix}, \boldsymbol{U}_{v,j} = \begin{bmatrix} u_{v,j}^{(1,1)} & \cdots & \cdots & u_{v,j}^{(1,r)} \\ \boldsymbol{0} & \cdots & \cdots & \boldsymbol{0} \\ \vdots & & & \vdots \\ \boldsymbol{0} & \cdots & \cdots & \boldsymbol{0} \\ & & \cdots & \\ u_{v,j}^{(f,1)} & \cdots & \cdots & u_{v,j}^{(f,r)} \\ \boldsymbol{0} & \cdots & \cdots & \boldsymbol{0} \\ \vdots & & & \vdots \\ \boldsymbol{0} & \cdots & \cdots & \boldsymbol{0} \end{bmatrix}. \tag{4.23}$$

We can continue this until the $k$-th subsystem where each column $\boldsymbol{v}_i^{(k)}$ of $\boldsymbol{V}^{(k)}$ is defined as follows

$$\begin{aligned} \boldsymbol{v}_i^{(k)} &= \left[(\sigma_i \boldsymbol{E} - \boldsymbol{A})^{-1}\boldsymbol{E}\right]^{i_k} \\ &\times \sum_{j_2=1}^{m} \cdots \sum_{j_k=1}^{m} \sum_{l_1=1}^{\tilde{r}} \cdots \sum_{l_{k-1}=1}^{\tilde{r}} \sum_{i_1=0}^{n_{\sigma_{l_1}}-1} \cdots \sum_{i_{k-1}=0}^{n_{\sigma_{l_{k-1}}}-1} \eta^{j_2,\ldots,j_k}_{\substack{i_1,\ldots,i_{k-1},i \\ i_1,\ldots,i_{k-1}}} (\sigma_i \boldsymbol{E} - \boldsymbol{A})^{-1}\boldsymbol{N}_{j_k} \\ &\times \left[(\tilde{\sigma}_{l_{k-1}}\boldsymbol{E} - \boldsymbol{A})^{-1}\boldsymbol{E}\right]^{i_{k-1}} (\tilde{\sigma}_{l_{k-1}}\boldsymbol{E} - \boldsymbol{A})^{-1}\boldsymbol{N}_{j_{k-1}} \cdots \\ &\times \boldsymbol{N}_{j_2} \left[(\tilde{\sigma}_{l_1}\boldsymbol{E} - \boldsymbol{A})^{-1}\boldsymbol{E}\right]^{i_1} (\tilde{\sigma}_{l_1}\boldsymbol{E} - \boldsymbol{A})^{-1}\boldsymbol{B}\,\boldsymbol{r}_{r_{\text{col}}(l_1)} \end{aligned}$$

$$= \left[(\sigma_i \boldsymbol{E} - \boldsymbol{A})^{-1}\boldsymbol{E}\right]^{i_k} \sum_{j_k=1}^{m} \sum_{l_{k-1}}^{r} u_{v,j_k}^{(\tilde{i},l_{k-1})} (\sigma_i \boldsymbol{E} - \boldsymbol{A})^{-1}\boldsymbol{N}_{j_k}\boldsymbol{v}_{l_{k-1}}^{(k-1)} \tag{4.24}$$

Thus, $\boldsymbol{V}^{(k)}$ solves the following linear *Sylvester* equation

$$\boldsymbol{E}\,\boldsymbol{V}^{(k)}\,\boldsymbol{S}_v - \boldsymbol{A}\,\boldsymbol{V}^{(k)} = \sum_{j=1}^{m} \boldsymbol{N}_j\,\boldsymbol{V}^{(k-1)}\,\boldsymbol{U}_{v,j}^{\mathsf{T}} \quad \text{for } k \geq 2 \tag{4.25}$$

if $\boldsymbol{S}_v$ and $\boldsymbol{U}_{v,j}$ have the structure given in (4.23). While combining (4.21) and (4.25), letting $k \to \infty$ and complying structures for $\boldsymbol{S}_v$, $\boldsymbol{U}_{v,j}$, $\boldsymbol{R}$ as shown in (4.22), (4.23) we can find a bilinear *Sylvester* equation

$$\boldsymbol{E}\,\boldsymbol{V}\,\boldsymbol{S}_v - \boldsymbol{A}\,\boldsymbol{V} - \sum_{j=1}^{m} \boldsymbol{N}_j\,\boldsymbol{V}\,\boldsymbol{U}_{v,j}^{\mathsf{T}} = \boldsymbol{B}\,\boldsymbol{R}$$

whose solution is $\boldsymbol{V}$. With an analog approach we can show that every $\boldsymbol{W}^{(k)}$ solves the following linear *Sylvester* equations

$$\begin{aligned}
&\boldsymbol{E}^\top \boldsymbol{W}^{(1)} \boldsymbol{S}_w^\top - \boldsymbol{A}^\top \boldsymbol{W}^{(1)} = \boldsymbol{C}^\top \boldsymbol{L} \\
&\boldsymbol{E}^\top \boldsymbol{W}^{(k)} \boldsymbol{S}_w^\top - \boldsymbol{A}^\top \boldsymbol{W}^{(k)} = \sum_{j=1}^{m} \boldsymbol{N}_j^\top \boldsymbol{W}^{(k-1)} \boldsymbol{U}_{w,j}^\top \quad \text{for } k \geq 2
\end{aligned}$$

where $\boldsymbol{S}_w$, $\boldsymbol{U}_{w,j}$, $\boldsymbol{L}$ have to fulfill following structures

$$\boldsymbol{S}_w = \begin{bmatrix} \tilde{\mu}_1 & & & & & & & & \\ -1 & \tilde{\mu}_1 & & & & & & & \\ & \ddots & \ddots & & & & & & \\ & & -1 & \tilde{\mu}_1 & & & & & \\ & & & & \ddots & & & & \\ & & & & & \tilde{\mu}_{\tilde{r}} & & & \\ & & & & & -1 & \tilde{\mu}_{\tilde{r}} & & \\ & & & & & & \ddots & \ddots & \\ & & & & & & & -1 & \tilde{\mu}_{\tilde{r}} \end{bmatrix}, \ \boldsymbol{U}_{w,j} = \begin{bmatrix} u_{w,j}^{(1,1)} & \cdots & & \cdots & u_{w,j}^{(1,r)} \\ \boldsymbol{0} & \cdots & & \cdots & \boldsymbol{0} \\ \vdots & & & & \vdots \\ \boldsymbol{0} & \cdots & & \cdots & \boldsymbol{0} \\ & & \cdots & & \\ u_{w,j}^{(u_{\text{ind}},1)} & \cdots & & \cdots & u_{w,j}^{(u_{\text{ind}},r)} \\ \boldsymbol{0} & \cdots & & \cdots & \boldsymbol{0} \\ \vdots & & & & \vdots \\ \boldsymbol{0} & \cdots & & \cdots & \boldsymbol{0} \end{bmatrix},$$

$$\boldsymbol{L} = \begin{bmatrix} \boldsymbol{l}_1 & \boldsymbol{0} & \cdots & \boldsymbol{0} & \cdots & \boldsymbol{l}_{1+n_{\tilde{\mu}1}+\cdots+n_{\tilde{\mu}\tilde{r}}} & \boldsymbol{0} & \cdots & \boldsymbol{0} \end{bmatrix} \tag{4.26}$$

where $u_{\text{ind}} = 1 + n_{\tilde{\mu}1} + \cdots + n_{\tilde{\mu}\tilde{r}-1}$. If (4.26) holds one could combine the linear *Sylvester* equations to

$$\boldsymbol{E}^\top \boldsymbol{W} \boldsymbol{S}_w^\top - \boldsymbol{A}^\top \boldsymbol{W} - \sum_{j=1}^{m} \boldsymbol{N}_j^\top \boldsymbol{W} \boldsymbol{U}_{w,j}^\top = \boldsymbol{C}^\top \boldsymbol{L}.$$

Concluding, we can obtain a multimoment projection matrix by solving a *Sylvester* equation if:

- $\boldsymbol{S}_v$ or rather $\boldsymbol{S}_w^\top$ have *Jordan* blocks for equal shifts,
- $\boldsymbol{U}_v$ or rather $\boldsymbol{U}_w$ have null-rows for rows which correspond to a high-order column of $\boldsymbol{V}$ or rather $\boldsymbol{W}$,
- $\boldsymbol{R}$ or rather $\boldsymbol{L}$ have null-columns for columns which correspond to a high-order column of $\boldsymbol{V}$ or rather $\boldsymbol{W}$.

*Example* 4.7 (Multi-Moment SISO Sylvester Equation). Let us reduce a SISO bilinear system $\zeta$. Therefore, we use $r = 2$ shifts $\sigma_1 = \sigma_2$, the tangential directions $r_1 = 1, r_2 = 0$ and the weight matrix

$$\boldsymbol{U}_v = \begin{bmatrix} u_v^{(1,1)} & u_v^{(1,2)} \\ 0 & 0 \end{bmatrix}.$$

Hence, we can obtain the projection matrix $\boldsymbol{V}$ by solving

$$\boldsymbol{E}\boldsymbol{V}\begin{bmatrix}\sigma_1 & -1\\ 0 & \sigma_2\end{bmatrix} - \boldsymbol{A}\boldsymbol{V} - \boldsymbol{N}\boldsymbol{V}\begin{bmatrix}u_v^{(1,1)} & 0\\ u_v^{(1,2)} & 0\end{bmatrix} = \boldsymbol{b}\begin{bmatrix}1 & 0\end{bmatrix},$$

$$\boldsymbol{E}\boldsymbol{V}\boldsymbol{S}_v - \boldsymbol{A}\boldsymbol{V} - \boldsymbol{N}\boldsymbol{V}\boldsymbol{U}_v = \boldsymbol{b}\boldsymbol{r}^{\mathsf{T}}.$$

△

*Example* 4.8 (Multi-Moment And Multipoint SISO Sylvester Equation). Let us reduce a SISO bilinear system $\zeta$. Therefore, we use $r = 3$ shifts $\sigma_1 = \sigma_2 \neq \sigma_3$, the "tangential directions" $r_1 = 1, r_2 = 0, r_3 = 1$ and the weights matrix

$$\boldsymbol{U}_v = \begin{bmatrix}u_v^{(1,1)} & u_v^{(1,2)} & u_v^{(1,3)}\\ 0 & 0 & 0\\ u_v^{(3,1)} & u_v^{(3,2)} & u_v^{(3,3)}\end{bmatrix}.$$

Hence, we can obtain the projection matrix $\boldsymbol{V}$ by solving

$$\boldsymbol{E}\boldsymbol{V}\begin{bmatrix}\sigma_1 & -1 & 0\\ 0 & \sigma_2 & 0\\ 0 & 0 & \sigma_3\end{bmatrix} - \boldsymbol{A}\boldsymbol{V} - \boldsymbol{N}\boldsymbol{V}\begin{bmatrix}u_v^{(1,1)} & 0 & u_v^{(3,1)}\\ u_v^{(1,2)} & 0 & u_v^{(3,2)}\\ u_v^{(1,3)} & 0 & u_v^{(3,3)}\end{bmatrix} = \boldsymbol{b}\begin{bmatrix}1 & 0 & 1\end{bmatrix},$$

$$\boldsymbol{E}\boldsymbol{V}\boldsymbol{S}_v - \boldsymbol{A}\boldsymbol{V} - \boldsymbol{N}\boldsymbol{V}\boldsymbol{U}_v = \boldsymbol{b}\boldsymbol{r}^{\mathsf{T}}.$$

△

## Computing Projection Matrices

Let us extend the previously introduced *Arnoldi* algorithm (Algorithm 4.1) to be able to compute projection matrices for high-order moments. Note that solving linear *Sylvester* equations as well as solving the bilinear *Sylvester* equation is similar to the multipoint *Volterra* series interpolation as long as the conditions for $\boldsymbol{S}_v$, $\boldsymbol{U}_v$ and $\boldsymbol{R}$ hold.
While recalling that we can truncate the *Volterra* series and consequently write $\boldsymbol{V}$ as

$$\boldsymbol{V} \approx \sum_{k=1}^{N} \boldsymbol{V}^{(k)} \tag{4.27}$$

we can find a formula for each column of each $\boldsymbol{V}^{(k)}$

$$\boldsymbol{v}_i^{(1)} = \left[(\sigma_i\boldsymbol{E} - \boldsymbol{A})^{-1}\boldsymbol{E}\right]^{i_1}(\sigma_i\boldsymbol{E} - \boldsymbol{A})^{-1}\boldsymbol{B}\,\boldsymbol{r}_i$$

$$\boldsymbol{v}_i^{(k)} = \left[(\sigma_i\boldsymbol{E} - \boldsymbol{A})^{-1}\boldsymbol{E}\right]^{i_k}\sum_{j_k=1}^{m}\sum_{l_{k-1=1}}^{r} u_{v,j_k}^{(\tilde{i},l_{k-1})}(\sigma_i\boldsymbol{E} - \boldsymbol{A})^{-1}\boldsymbol{N}_{j_k}\boldsymbol{v}_{l_{k-1}}^{(k-1)} \quad \text{for } k \geq 2.$$

Each $i_k = 0$ if the previous column was computed with another shift and $i_k = q$ if the $q$ previous columns were computed with the same shift. With this we can give a modified *Arnoldi* algorithm 4.3 which inherits the following constraints:

- $\boldsymbol{U}_v$ has null-rows for rows which correspond to a high-order column of $\boldsymbol{V}$,
- $\boldsymbol{R}$ has null-columns for columns which correspond to a high-order column of $\boldsymbol{V}$.

Note, that it is very important to compute the LU-factors for each shift only once as done in Algorithm 4.3 Line 2. The index $l_i$ means that we choose the LU-factors corresponding to the current shift $\sigma_i$.

*Remark* 4.12 (Multimoment Volterra Series Interpolation With Arbitrary Weights). We do not recommend using arbitrary weights. One should recall the definition of the moments. The $q$-th moment is the $q$-th derivative of the transfer function defined by a *Taylor* expansion. Thus, for each subsystem we actually match a weighted *Taylor* series for a certain combination of interpolation points. If one chooses arbitrary weights and tangential directions such that corresponding multimoments are not weighted the same, one chooses terms from different weighted *Taylor* series. In other words: this would not weight the terms of the *Taylor* series equally. Since a *Taylor* series increases its accuracy with more considered terms, it is quite obvious that it is important to consider the whole series in order to increase accuracy. △

**Algorithm 4.3 :** Efficient Arnoldi Algorithm For Computing Projection Matrices Modification For Multimoments

```
Data : bilinear system $\boldsymbol{\zeta}$, sorted interpolation points $\sigma_1, \ldots, \sigma_r$, unique interpolation
       points $\sigma_{\tilde{1}}, \ldots, \sigma_{\tilde{r}}$, number of unique interpolation points $\tilde{r}$, interpolation weights
       $\boldsymbol{U}_v$, tangential directions $\boldsymbol{R}$, truncation index $N$
Result : approximated projection matrix $\boldsymbol{V}$
1  for $l = 1 : \tilde{r}$ do
2      $[\boldsymbol{L}_l, \boldsymbol{U}_l] = \mathtt{lu}(\tilde{\sigma}_l \boldsymbol{E} - \boldsymbol{A})$ ;        ▷ Compute LU factors for unique shifts
3  for $i = 1 : r$ do
4      if $i > 1$ && $\sigma_i \neq \sigma_{i-1}$ then
5          $\boldsymbol{V}_{\text{old}}(:, i) = \boldsymbol{U}_{l_i} \backslash (\boldsymbol{L}_{l_i} \backslash (\boldsymbol{B}\boldsymbol{R}(:, i)))$ ;        ▷ Compute $\boldsymbol{V}^{(1)}$
6      else
7          $\boldsymbol{V}_{\text{old}}(:, i) = \boldsymbol{U}_{l_i} \backslash (\boldsymbol{L}_{l_i} \backslash (\boldsymbol{E}\boldsymbol{V}_{\text{old}}(:, i-1)))$ ;        ▷ high-order moment
8  $\boldsymbol{V} = \boldsymbol{V}_{\text{old}}$;
9  for $k = 2 : N$ do
10     for $i = 1 : r$ do
11         if $i > 1$ && $\sigma_i \neq \sigma_{i-1}$ then
12             $\boldsymbol{v}_{\text{temp}} = \boldsymbol{0}$;
13             for $l = 1 : r$ do
14                 for $j = 1 : m$ do
15                     $\boldsymbol{v}_{\text{temp}} = \boldsymbol{v}_{\text{temp}} + \boldsymbol{U}_{v,j}(i, l)\boldsymbol{N}_j \boldsymbol{V}_{\text{old}}(:, l)$;
16             $\boldsymbol{V}_{\text{new}}(:, i) = \boldsymbol{U}_{l_i} \backslash (\boldsymbol{L}_{l_i} \backslash \boldsymbol{v}_{\text{temp}})$ ;        ▷ Compute $\boldsymbol{V}^{(k>1)}$
17         else
18             $\boldsymbol{V}_{\text{new}}(:, i) = \boldsymbol{U}_{l_i} \backslash (\boldsymbol{L}_{l_i} \backslash (\boldsymbol{E}\boldsymbol{V}_{\text{new}}(:, i-1)))$ ;        ▷ high-order moment
19     $\boldsymbol{V}_{\text{old}} = \boldsymbol{V}_{\text{new}}$;
20     $\boldsymbol{V} = \boldsymbol{V} + \boldsymbol{V}_{\text{new}}$;
```

### 4.3.3 Other Special Cases

To complete the *Volterra* series interpolation framework we discuss certain special cases. First we take a closer look at the Block *Krylov* case. After that, we show how we can obtain projection matrices for *Markov Parameters*. Finally, we determine which constraints the interpolation data has to fulfill to obtain real projection matrices while using complex expansion points.

#### Block Krylov

As mentioned in the fundamentals, it is possible to build a *Krylov* subspace with two matrices which is called block *Krylov*. Applied to the *Volterra* series interpolation this means not to consider tangential directions. Hence, the input block *Krylov* subspace for multipoint and multimoment *Volterra* series interpolation is defined as follows

$$\boldsymbol{V} \subseteq \mathcal{K}_r\left((\sigma_i\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E}, \boldsymbol{V}_i\right) \in \mathbb{R}^{n\times rm}$$

where

$$\begin{aligned}\boldsymbol{V}_i = \sum_{k=1}^{\infty}\sum_{j_2=1}^{m}\cdots\sum_{j_k=1}^{m}\sum_{l_1=1}^{\tilde r}\cdots\sum_{l_{k-1}=1}^{\tilde r}\sum_{i_1=0}^{n_{\tilde\sigma_{l_1}}-1}\cdots\sum_{i_{k-1}=0}^{n_{\tilde\sigma_{l_{k-1}}}-1}\eta^{j_2,\dots,j_k}_{\substack{l_1,\dots,l_{k-1},i\\ i_1,\dots,i_{k-1}}}\,(\sigma_i\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{N}_{j_k}\\ \times\left[(\tilde\sigma_{l_{k-1}}\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E}\right]^{i_{k-1}}(\tilde\sigma_{l_{k-1}}\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{N}_{j_{k-1}}\cdots\\ \times\boldsymbol{N}_{j_2}\left[(\tilde\sigma_{l_1}\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E}\right]^{i_1}(\tilde\sigma_{l_1}\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{B}.\end{aligned}$$

As we can see the projection matrix $\boldsymbol{V}$ is now $\in \mathbb{R}^{n\times rm}$. Since we do not want to introduce a new framework for block *Krylov* we try to derive a way to integrate block *Krylov* in the existing one. Therefore, we assume that we only consider **one subsystem**. With this assumption we can compute each part-matrix $\boldsymbol{V}_i$ of $\boldsymbol{V} = \left[\boldsymbol{V}_1, \cdots, \boldsymbol{V}_r\right]$ as follows

$$\boldsymbol{V}_i = \left[(\sigma_i\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E}\right]^{i_1}(\sigma_i\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{B}$$

where $i = 1, \dots, r$. $i_1 = 0$ if the previous matrix $\boldsymbol{V}_{i-1}$ was computed with a different shift and $i_1 = q$ if the $q$ previous matrices were computed with the same shift. Consequently, each column $\boldsymbol{v}_{i,j}$ of $\boldsymbol{V}_i$ is defined as

$$\boldsymbol{v}_{i,j} = \left[(\sigma_i\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E}\right]^{i_1}(\sigma_i\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{b}_j$$

where $\boldsymbol{b}_j$ denotes the $j$-th column of $\boldsymbol{B}$. To somehow apply this to our existing frameworks we can multiply the formula for $\boldsymbol{V}_i$ with $\boldsymbol{R}_{\tilde i} = \mathbf{I}_m$ from the left which yields

$$\boldsymbol{V}_i = \left[(\sigma_i\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E}\right]^{i_1}(\sigma_i\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{B}\boldsymbol{R}_{\tilde i}.$$

where $\tilde i$ is defined in (4.19). Hence, we can compute each column of $\boldsymbol{V}_i$ with

$$\boldsymbol{v}_{i,j} = \left[(\sigma_i\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E}\right]^{i_1}(\sigma_i\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{B}\boldsymbol{r}_{\tilde i,j} = \left[(\sigma_i\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E}\right]^{i_1}(\sigma_i\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{b}_j$$

where $\boldsymbol{r}_{i,j}$ denotes the $j$-th column of $\boldsymbol{R}_i$. To obtain a *Sylvester* equation representation we have to use one shift $\sigma_i$ $m$-times in the interpolation point matrix $\boldsymbol{S}_v$ without using

*Jordan* blocks. We also have to set $\boldsymbol{R}_i$ to zero if it corresponds to a high-order moment $\boldsymbol{V}_i$. Finally, we have to enlarge each $\boldsymbol{U}_{v,j}$ such that each weight gets repeated $m$-times in row direction and $m$-times in column direction such that the enlarged $\boldsymbol{U}_{v,j}$ is in $\mathbb{R}^{rm\times rm}$. To preserve the *Jordan* structure of $\boldsymbol{S}_v$ one could change the order of columns of $\boldsymbol{V}$ since this does not change the basis of $\boldsymbol{V}$. The following examples illustrate the enlargement of the matrices.

*Example* 4.9 (Multipoint Block *Krylov*). Let us assume we want to reduce a MIMO bilinear system with $m = 3$ inputs with $r = 2$ shifts $\sigma_1 \neq \sigma_2$. Therefore, we use the weight matrices

$$\boldsymbol{U}_{v,j} = \begin{bmatrix} u_{v,j}^{(1,1)} & u_{v,j}^{(1,2)} \\ u_{v,j}^{(2,1)} & u_{v,j}^{(2,2)} \end{bmatrix}$$

and no tangential directions. As mentioned we can use a unit matrix for each tangential direction such that

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{R}_1, & \boldsymbol{R}_2 \end{bmatrix}$$

where

$$\boldsymbol{R}_1 = \boldsymbol{R}_2 = \begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix}.$$

To get the correct block *Krylov* projection matrix we have to match each shift $m = 3$ times and extend the weights accordingly which results in the following $\boldsymbol{S}_v$ and $\boldsymbol{U}_{v,j}$ matrices

$$\boldsymbol{S}_v = \begin{bmatrix} \sigma_1 & & & & & \\ & \sigma_1 & & & & \\ & & \sigma_1 & & & \\ & & & \sigma_2 & & \\ & & & & \sigma_2 & \\ & & & & & \sigma_2 \end{bmatrix}, \boldsymbol{U}_{v,j} = \begin{bmatrix} u_{v,j}^{(1,1)} & u_{v,j}^{(1,1)} & u_{v,j}^{(1,1)} & u_{v,j}^{(1,2)} & u_{v,j}^{(1,2)} & u_{v,j}^{(1,2)} \\ u_{v,j}^{(1,1)} & u_{v,j}^{(1,1)} & u_{v,j}^{(1,1)} & u_{v,j}^{(1,2)} & u_{v,j}^{(1,2)} & u_{v,j}^{(1,2)} \\ u_{v,j}^{(1,1)} & u_{v,j}^{(1,1)} & u_{v,j}^{(1,1)} & u_{v,j}^{(1,2)} & u_{v,j}^{(1,2)} & u_{v,j}^{(1,2)} \\ u_{v,j}^{(2,1)} & u_{v,j}^{(2,1)} & u_{v,j}^{(2,1)} & u_{v,j}^{(2,2)} & u_{v,j}^{(2,2)} & u_{v,j}^{(2,2)} \\ u_{v,j}^{(2,1)} & u_{v,j}^{(2,1)} & u_{v,j}^{(2,1)} & u_{v,j}^{(2,2)} & u_{v,j}^{(2,2)} & u_{v,j}^{(2,2)} \\ u_{v,j}^{(2,1)} & u_{v,j}^{(2,1)} & u_{v,j}^{(2,1)} & u_{v,j}^{(2,2)} & u_{v,j}^{(2,2)} & u_{v,j}^{(2,2)} \end{bmatrix}.$$

△

*Example* 4.10 (Multipoint And Multimoment Block *Krylov*). Let us assume we want to reduce a MIMO bilinear system with $m = 2$ inputs with $r = 3$ shifts $\sigma_1 = \sigma_2 \neq \sigma_3$. Therefore, we use the weight matrices

$$\boldsymbol{U}_{v,j} = \begin{bmatrix} u_{v,j}^{(1,1)} & u_{v,j}^{(1,2)} & u_{v,j}^{(1,3)} \\ 0 & 0 & 0 \\ u_{v,j}^{(2,1)} & u_{v,j}^{(2,2)} & u_{v,j}^{(2,3)} \end{bmatrix}$$

and no tangential directions. As mentioned we can use a unit matrix for each tangential direction such that

$$\boldsymbol{R} = \begin{bmatrix}\boldsymbol{R}_1, & \boldsymbol{R}_2, & \boldsymbol{R}_3\end{bmatrix}$$

where

$$\boldsymbol{R}_1 = \boldsymbol{R}_3 \begin{bmatrix}1 & \\ & 1\end{bmatrix}, \ \boldsymbol{R}_2 = \mathbf{0}.$$

To get the correct block *Krylov* projection matrix we have to match each shift $m = 2$ times and extend the weights accordingly. Hence, it follows that

$$\boldsymbol{S}_v = \begin{bmatrix}\sigma_1 & & -1 & & & \\ & \sigma_1 & & -1 & & \\ & & \sigma_2 & & & \\ & & & \sigma_2 & & \\ & & & & \sigma_3 & \\ & & & & & \sigma_3\end{bmatrix}, \boldsymbol{U}_{v,j} = \begin{bmatrix}u_{v,j}^{(1,1)} & u_{v,j}^{(1,1)} & u_{v,j}^{(1,2)} & u_{v,j}^{(1,2)} & u_{v,j}^{(1,3)} & u_{v,j}^{(1,3)} \\ u_{v,j}^{(1,1)} & u_{v,j}^{(1,1)} & u_{v,j}^{(1,2)} & u_{v,j}^{(1,2)} & u_{v,j}^{(1,3)} & u_{v,j}^{(1,3)} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ u_{v,j}^{(2,1)} & u_{v,j}^{(2,1)} & u_{v,j}^{(2,2)} & u_{v,j}^{(2,2)} & u_{v,j}^{(2,3)} & u_{v,j}^{(2,3)} \\ u_{v,j}^{(2,1)} & u_{v,j}^{(2,1)} & u_{v,j}^{(2,2)} & u_{v,j}^{(2,2)} & u_{v,j}^{(2,3)} & u_{v,j}^{(2,3)}\end{bmatrix}.$$

With the above we obtain a projection matrix

$$\boldsymbol{V} = \begin{bmatrix}\boldsymbol{v}_1, & \boldsymbol{v}_2, & \boldsymbol{v}_3, & \boldsymbol{v}_4, & \boldsymbol{v}_5, & \boldsymbol{v}_6\end{bmatrix} \in \mathbb{R}^{n\times 6}. \tag{4.28}$$

To preserve the *Jordan* structure of $\boldsymbol{S}_v$ we could also use

$$\boldsymbol{R} = \begin{bmatrix}1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1\end{bmatrix}$$

and

$$\boldsymbol{S}_v = \begin{bmatrix}\sigma_1 & -1 & & & & \\ & \sigma_2 & & & & \\ & & \sigma_1 & -1 & & \\ & & & \sigma_2 & & \\ & & & & \sigma_3 & \\ & & & & & \sigma_3\end{bmatrix}, \boldsymbol{U}_{v,j} = \begin{bmatrix}u_{v,j}^{(1,1)} & u_{v,j}^{(1,1)} & u_{v,j}^{(1,2)} & u_{v,j}^{(1,2)} & u_{v,j}^{(1,3)} & u_{v,j}^{(1,3)} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ u_{v,j}^{(1,1)} & u_{v,j}^{(1,1)} & u_{v,j}^{(1,2)} & u_{v,j}^{(1,2)} & u_{v,j}^{(1,3)} & u_{v,j}^{(1,3)} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ u_{v,j}^{(2,1)} & u_{v,j}^{(2,1)} & u_{v,j}^{(2,2)} & u_{v,j}^{(2,2)} & u_{v,j}^{(2,3)} & u_{v,j}^{(2,3)} \\ u_{v,j}^{(2,1)} & u_{v,j}^{(2,1)} & u_{v,j}^{(2,2)} & u_{v,j}^{(2,2)} & u_{v,j}^{(2,3)} & u_{v,j}^{(2,3)}\end{bmatrix}$$

to compute $\tilde{\boldsymbol{V}}$. $\boldsymbol{V}$ and $\tilde{\boldsymbol{V}}$ are related by a change in columns such that

$$\tilde{\boldsymbol{V}} = \begin{bmatrix}\boldsymbol{v}_1, & \boldsymbol{v}_3, & \boldsymbol{v}_2, & \boldsymbol{v}_4, & \boldsymbol{v}_5, & \boldsymbol{v}_6\end{bmatrix} \in \mathbb{R}^{n\times 6}. \tag{4.29}$$

△

### Markov Parameters

As mentioned, the $k$-th *Markov Parameter* is the $k$-th transfer function expanded by a *Taylor* series at the expansion point $s \to \infty$. Let us recall the definition of the $k$-th *Markov Parameter*

$$\boldsymbol{M}^{(j_1,\ldots,j_k)}_{k,\infty,(i_1,\cdots,i_k)} = \boldsymbol{C}\left[(\boldsymbol{E}^{-1}\boldsymbol{A})^{-1}\right]^{i_1}\boldsymbol{E}^{-1}\boldsymbol{N}_{j_k}\cdots\boldsymbol{E}^{-1}\boldsymbol{N}_{j_2}\left[(\boldsymbol{E}^{-1}\boldsymbol{A})^{-1}\right]^{i_k}\boldsymbol{E}^{-1}\boldsymbol{B}.$$

Since the above formula does not consider choosing infinity together with finite expansions points we want to give a more general formula

$$\begin{aligned}\boldsymbol{M}^{(j_1,\ldots,j_k)}_{k,(i_1,\cdots,i_k)}(\sigma_1,\cdots,\sigma_k) = \boldsymbol{C}&\underbrace{\left[(\sigma_1\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E}\right]^{i_1}(\sigma_1\boldsymbol{E}-\boldsymbol{A})^{-1}}_{\substack{\text{if } \sigma_1=\infty \text{ then}\\ =[(\boldsymbol{E}^{-1}\boldsymbol{A})^{-1}]^{i_1}\boldsymbol{E}^{-1}}}\boldsymbol{N}_{j_k}\cdots\\ &\times\boldsymbol{N}_{j_2}\underbrace{\left[(\sigma_k\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E}\right]^{i_k}(\sigma_k\boldsymbol{E}-\boldsymbol{A})^{-1}}_{\substack{\text{if } \sigma_k=\infty \text{ then}\\ =[(\boldsymbol{E}^{-1}\boldsymbol{A})^{-1}]^{i_k}\boldsymbol{E}^{-1}}}\boldsymbol{B}.\end{aligned}$$

As one can see it is not possible to write the above in a consistent mathematical manner. Therefore, we waive defining the corresponding *Krylov* subspace as well as the interpolation conditions and continue with a heuristic approach to modify Algorithm 4.3 for *Markov Parameters*. In this sense we want to generalize the formulas for each column of $\boldsymbol{V}^{(1)}$ or rather $\boldsymbol{V}^{(k)}$ which yields

$$\begin{aligned}\boldsymbol{v}_i^{(1)} &= \underbrace{\left[(\sigma_i\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E}\right]^{i_1}(\sigma_i\boldsymbol{E}-\boldsymbol{A})^{-1}}_{\substack{\text{if } \sigma_i=\infty \text{ then}\\ =[(\boldsymbol{E}^{-1}\boldsymbol{A})^{-1}]^{i_1}\boldsymbol{E}^{-1}}}\boldsymbol{B}\,\boldsymbol{r}_i\\ \boldsymbol{v}_i^{(k)} &= \sum_{j_k=1}^{m}\sum_{l_{k-1}}^{r} u_{v,j_k}^{(i,l_{k-1})}\underbrace{\left[(\sigma_i\boldsymbol{E}-\boldsymbol{A})^{-1}\boldsymbol{E}\right]^{i_k}(\sigma_i\boldsymbol{E}-\boldsymbol{A})^{-1}}_{\substack{\text{if } \sigma_i=\infty \text{ then}\\ =[(\boldsymbol{E}^{-1}\boldsymbol{A})^{-1}]^{i_k}\boldsymbol{E}^{-1}}}\boldsymbol{N}_{j_k}\boldsymbol{v}_{l_{k-1}}^{(k-1)} \quad \text{for } k \geq 2.\end{aligned}$$

Hereby, $i_k = 0$ if the previous column $\boldsymbol{v}_{i-1}^k$ was computed with a different shift and $i_k = q$ if the $q$ previous columns were computed with the same shift. This allows us to give the generalized *Arnoldi* Algorithm 4.4 which is able to compute high-order moments as well as *Markov Parameters* and high-order *Markov Parameters*. This algorithm uses the same conditions for the interpolation data for multimoment *Markov Parameters* as for finite multimoments.

**Algorithm 4.4 :** Efficient Arnoldi Algorithm For Computing Projection Matrices Modification For Multimoments Including Markov Parameters

```
Data : bilinear system $\boldsymbol{\zeta}$, sorted $(\sigma_1,\dots,\sigma_r)$ and unique $\tilde{\sigma}_1,\dots,\tilde{\sigma}_r$ interpolation points,
       amount distinct shifts $\tilde{r}$, weights $\boldsymbol{U}_v$, tangential directions $\boldsymbol{R}$, truncation index $N$
Result : approximated projection matrix $\boldsymbol{V}$
1  for $l = 1 : \tilde{r}$ do
2      $[\boldsymbol{L}_l, \boldsymbol{U}_l] = \mathtt{lu}(\tilde{\sigma}_l \boldsymbol{E} - \boldsymbol{A})$ ;    ▷ Compute LU factors for unique shifts
3  if any$(\sigma_i == \infty)$ then
4      $[\boldsymbol{L}_\infty, \boldsymbol{U}_\infty] = \mathtt{lu}(\boldsymbol{E})$ ;    ▷ Compute LU factors for Markov Parameters
5      if $n_{\sigma_\infty} > 1$ then
6          $[\boldsymbol{L}_{\infty,\text{higher}}, \boldsymbol{U}_{\infty,\text{higher}}] = \mathtt{lu}(\boldsymbol{U}_\infty \backslash (\boldsymbol{L}_\infty \backslash \boldsymbol{A}))$;
7  for $i$ = 1 : $r$ do
8      if $i > 1$ && $\sigma_i \neq \sigma_{i-1}$ then
9          if $\sigma_i == \infty$ then
10             $\boldsymbol{V}_{\text{old}}(:,i) = \boldsymbol{U}_\infty \backslash (\boldsymbol{L}_\infty \backslash (\boldsymbol{B}\boldsymbol{R}(:,i)))$ ;    ▷ Compute Markov Parameter
11         else
12             $\boldsymbol{V}_{\text{old}}(:,i) = \boldsymbol{U}_{l_i} \backslash (\boldsymbol{L}_{l_i} \backslash (\boldsymbol{B}\boldsymbol{R}(:,i)))$ ;    ▷ Compute $\boldsymbol{V}^{(1)}$
13     else
14         if $\sigma_i == \infty$ then
15             $\boldsymbol{V}_{\text{old}}(:,i) = \boldsymbol{U}_{\infty,\text{higher}} \backslash (\boldsymbol{L}_{\infty,\text{higher}} \backslash (\boldsymbol{V}_{\text{old}}(:,i-1)))$ ;    ▷ high-order Markov
16         else
17             $\boldsymbol{V}_{\text{old}}(:,i) = \boldsymbol{U}_{l_i} \backslash (\boldsymbol{L}_{l_i} \backslash (\boldsymbol{E}\boldsymbol{V}_{\text{old}}(:,i-1)))$ ;    ▷ high-order moment
18 $\boldsymbol{V} = \boldsymbol{V}_{\text{old}}$;
19 for $k$ = 2 : $N$ do
20     for $i$ = 1 : $r$ do
21         if $i > 1$ && $\sigma_i \neq \sigma_{i-1}$ then
22             $\boldsymbol{v}_{\text{temp}} = \boldsymbol{0}$;
23             for $l$ = 1 : $r$ do
24                 for $j$ = 1 : $m$ do
25                     $\boldsymbol{v}_{\text{temp}} = \boldsymbol{v}_{\text{temp}} + \boldsymbol{U}_{v,j}(i,l)\boldsymbol{N}_j\boldsymbol{V}_{\text{old}}(:,l)$;
26             if $\sigma_i == \infty$ then
27                 $\boldsymbol{V}_{\text{new}}(:,i) = \boldsymbol{U}_\infty \backslash (\boldsymbol{L}_\infty \backslash \boldsymbol{v}_{\text{temp}})$ ;    ▷ Compute $\boldsymbol{V}^{(k>1)}$ for Markov
28             else
29                 $\boldsymbol{V}_{\text{new}}(:,i) = \boldsymbol{U}_{l_i} \backslash (\boldsymbol{L}_{l_i} \backslash \boldsymbol{v}_{\text{temp}})$ ;    ▷ Compute $\boldsymbol{V}^{(k>1)}$
30         else
31             if $\sigma_i == \infty$ then
32                 $\boldsymbol{V}_{\text{new}}(:,i) = \boldsymbol{U}_{\infty,\text{higher}} \backslash (\boldsymbol{L}_{\infty,\text{higher}} \backslash (\boldsymbol{V}_{\text{new}}(:,i-1)))$; ▷ high-order Markov
33             else
34                 $\boldsymbol{V}_{\text{new}}(:,i) = \boldsymbol{U}_{l_i} \backslash (\boldsymbol{L}_{l_i} \backslash (\boldsymbol{E}\boldsymbol{V}_{\text{new}}(:,i-1)))$ ;    ▷ high-order moment
35     $\boldsymbol{V}_{\text{old}} = \boldsymbol{V}_{\text{new}}$;
36     $\boldsymbol{V} = \boldsymbol{V} + \boldsymbol{V}_{\text{new}}$;
```

### Complex Expansion Points

As it turns out, it is important to consider complex expansion points e.g. for BIRKA. Using complex expansion points results in a complex projection matrix which then results in a reduced system with complex matrices. To avoid that, we try to find a transformation for $\boldsymbol{V}$ or rather $\boldsymbol{W}$ which allows us to obtain a real projection matrix $\tilde{\boldsymbol{V}} = \boldsymbol{V}\boldsymbol{T}_v$ or rather $\tilde{\boldsymbol{W}} = \boldsymbol{W}\boldsymbol{T}_w$. With the invariance of *Sylvester* equations we know that transforming the projection matrices comes with a transformation of the interpolation data. Hence, we address the problem of finding transformation matrices $\boldsymbol{T}_v$ and $\boldsymbol{T}_w$ by trying to make the shift matrices $\boldsymbol{S}_v$ and $\boldsymbol{S}_w$ real. If it then is possible to obtain real interpolation data by application of the transformation matrix, we can conclude that the transformed projection matrices also have to be real. Hence, if we find a transformation matrix, we can compute the projection matrix with the imaginary interpolation data and after that transform it to real numbers. Let us recall the transformed *Sylvester* equations and the corresponding interpolation data

$$\begin{aligned}
\boldsymbol{E}\tilde{\boldsymbol{V}}\tilde{\boldsymbol{S}}_v - \boldsymbol{A}\tilde{\boldsymbol{V}} - \sum_{j=1}^{m} \boldsymbol{N}_j \tilde{\boldsymbol{V}} \tilde{\boldsymbol{U}}_{w,j}^{\mathsf{T}} &= \boldsymbol{B}\tilde{\boldsymbol{R}}, \\
\boldsymbol{E}^{\mathsf{T}}\tilde{\boldsymbol{W}}\tilde{\boldsymbol{S}_w}^{\mathsf{T}} - \boldsymbol{A}^{\mathsf{T}}\tilde{\boldsymbol{W}} - \sum_{j=1}^{m} \boldsymbol{N}_j^{\mathsf{T}} \tilde{\boldsymbol{W}} \tilde{\boldsymbol{U}}_{w,j}^{\mathsf{T}} &= \boldsymbol{C}^{\mathsf{T}}\tilde{\boldsymbol{W}}
\end{aligned}$$

where

$$\begin{aligned}
\tilde{\boldsymbol{S}}_v &= \boldsymbol{T}_v^{-1}\boldsymbol{S}_v\boldsymbol{T}_v, & \tilde{\boldsymbol{U}}_{v,j} &= \boldsymbol{T}_v^{\mathsf{T}}\boldsymbol{U}_{v,j}\boldsymbol{T}_v^{-\mathsf{T}}, & \tilde{\boldsymbol{R}} &= \boldsymbol{R}\boldsymbol{T}_v, \\
\tilde{\boldsymbol{S}}_w &= \boldsymbol{T}_w^{\mathsf{T}}\boldsymbol{S}_w\boldsymbol{T}_w^{-\mathsf{T}}, & \tilde{\boldsymbol{U}}_{w,j} &= \boldsymbol{T}_w^{\mathsf{T}}\boldsymbol{U}_{w,j}\boldsymbol{T}_w^{-\mathsf{T}}, & \tilde{\boldsymbol{L}} &= \boldsymbol{L}\boldsymbol{T}_w.
\end{aligned}$$

The first step is to find a transformation matrix which eliminates the imaginary part in $\boldsymbol{S}_v$ or rather $\boldsymbol{S}_w$ so that $\tilde{\boldsymbol{S}}_v$ and $\tilde{\boldsymbol{S}}_w$ are real matrices. It is possible to show that the following transformation matrix

$$\boldsymbol{T} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2}\,\mathrm{i} & & & \\ \frac{1}{2} & -\frac{1}{2}\,\mathrm{i} & & & \\ & & \ddots & & \\ & & & \frac{1}{2} & \frac{1}{2}\,\mathrm{i} \\ & & & \frac{1}{2} & -\frac{1}{2}\,\mathrm{i} \end{bmatrix}, \qquad \boldsymbol{T}^{-1} = \begin{bmatrix} 1 & 1 & & & \\ -\mathrm{i} & \mathrm{i} & & & \\ & & \ddots & & \\ & & & 1 & 1 \\ & & & -\mathrm{i} & \mathrm{i} \end{bmatrix}$$

makes $\boldsymbol{S}_v$ and $\boldsymbol{S}_w$ real if both have only complex conjugated pairs on their diagonal such that

$$\boldsymbol{S}_v = \begin{bmatrix} \sigma_1 & & & & \\ & \bar{\sigma}_1 & & & \\ & & \ddots & & \\ & & & \sigma_r & \\ & & & & \bar{\sigma}_r \end{bmatrix}, \qquad \boldsymbol{S}_w = \begin{bmatrix} \mu_1 & & & & \\ & \bar{\mu}_1 & & & \\ & & \ddots & & \\ & & & \mu_r & \\ & & & & \bar{\mu}_r \end{bmatrix}.$$

Hereby $\bar{\sigma}$ denotes the complex conjugated of $\sigma$. Note that it is important that the first shift of a complex conjugated pair is always the one which has the negative imaginary part.

*Example* 4.11 (Making Shift Matrix Real). Let us assume we use $\sigma_1 = 1 - \mathrm{i}$ and $\sigma_2 = 1 + \mathrm{i}$ to reduce a bilinear system. Hence, the shift matrix $\boldsymbol{S}_v$ is given by

$$\boldsymbol{S}_v = \begin{bmatrix} 1-\mathrm{i} & \\ & 1+\mathrm{i} \end{bmatrix}.$$

The above implies that it is possible to obtain a real $\tilde{\boldsymbol{S}}_v = \boldsymbol{T}_v^{-1}\boldsymbol{S}_v\boldsymbol{T}_v$ if we use the following transformation matrix

$$\boldsymbol{T}_v = \begin{bmatrix} \frac{1}{2} & \frac{1}{2}\,\mathrm{i} \\ \frac{1}{2} & -\frac{1}{2}\,\mathrm{i} \end{bmatrix}, \qquad \boldsymbol{T}_v^{-1} = \begin{bmatrix} 1 & 1 \\ -\mathrm{i} & \mathrm{i} \end{bmatrix}.$$

Applying this to $\boldsymbol{S}_v$ yields

$$\begin{aligned} \tilde{\boldsymbol{S}}_v = \boldsymbol{T}_v^{-1}\boldsymbol{S}_v\boldsymbol{T}_v &= \begin{bmatrix} 1 & 1 \\ -\mathrm{i} & \mathrm{i} \end{bmatrix} \begin{bmatrix} 1-\mathrm{i} & \\ & 1+\mathrm{i} \end{bmatrix} \begin{bmatrix} \frac{1}{2} & \frac{1}{2}\,\mathrm{i} \\ \frac{1}{2} & -\frac{1}{2}\,\mathrm{i} \end{bmatrix} \\ &= \begin{bmatrix} 1 & 1 \\ -\mathrm{i} & \mathrm{i} \end{bmatrix} \begin{bmatrix} \frac{1}{2}-\frac{1}{2}\,\mathrm{i} & \frac{1}{2}+\frac{1}{2}\,\mathrm{i} \\ \frac{1}{2}+\frac{1}{2}\,\mathrm{i} & \frac{1}{2}-\frac{1}{2}\,\mathrm{i} \end{bmatrix} \\ &= \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}. \end{aligned} \tag{4.30}$$

△

As one might want to simultaneously use complex conjugated paired expansion points and real expansion points we can generalize the transformation matrix. Let the shift matrix have following structure

$$\boldsymbol{S}_v = \begin{bmatrix} \sigma_1 &&&&&&& \\ & \bar{\sigma}_1 &&&&&& \\ && \ddots &&&&& \\ &&& \sigma_{r_{\mathrm{imag}}} &&&& \\ &&&& \bar{\sigma}_{r_{\mathrm{imag}}} &&& \\ &&&&& \sigma_{r_{\mathrm{imag}}+1} && \\ &&&&&& \ddots & \\ &&&&&&& \sigma_r \end{bmatrix} \tag{4.31}$$

where $\sigma_1, \cdots, \sigma_{r_{\text{imag}}} \in \mathbb{C}$ with the corresponding complex conjugated pairs $\bar{\sigma}_1, \cdots, \bar{\sigma}_{r_{\text{imag}}} \in \mathbb{C}$ and $\sigma_{r_{\text{imag}}+1}, \cdots, \sigma_r \in \mathbb{R}$. To obtain a real $\tilde{\boldsymbol{S}}_v = \boldsymbol{T}_v^{-1}\boldsymbol{S}_v\boldsymbol{T}_v$ we have to use following projection matrix

$$\boldsymbol{T} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2}\mathrm{i} & & & & & & \\ \frac{1}{2} & -\frac{1}{2}\mathrm{i} & & & & & & \\ & & \ddots & & & & & \\ & & & \frac{1}{2} & \frac{1}{2}\mathrm{i} & & & \\ & & & \frac{1}{2} & -\frac{1}{2}\mathrm{i} & & & \\ & & & & & 1 & & \\ & & & & & & \ddots & \\ & & & & & & & 1 \end{bmatrix}, \quad \boldsymbol{T}^{-1} = \begin{bmatrix} 1 & 1 & & & & & & \\ -\mathrm{i} & \mathrm{i} & & & & & & \\ & & \ddots & & & & & \\ & & & 1 & 1 & & & \\ & & & -\mathrm{i} & \mathrm{i} & & & \\ & & & & & 1 & & \\ & & & & & & \ddots & \\ & & & & & & & 1 \end{bmatrix}.$$

Obviously, one could use the same projection matrices to obtain a real $\tilde{\boldsymbol{S}}_w = \boldsymbol{T}_w^{\mathsf{T}}\boldsymbol{S}_w\boldsymbol{T}_w^{-\mathsf{T}}$ where $\boldsymbol{S}_w$ has a similar structure as $\boldsymbol{S}_v$ shown in (4.31).
The second step is to check if it is possible to make the other interpolation data real with the above transformation matrix. It is possible to derive conditions concerning the structure of the tangential directions and weights to obtain real interpolation data. Since it is complicated to generalize these conditions to cover all cases (multipoint, multimoment), we use a more heuristic and straightforward approach: we simply transform the tangential directions and weights and check if the transformed data is real. If this is the case we can compute the projection matrices with the imaginary interpolation data and after that make them real as follows

$$\begin{aligned} \boldsymbol{V}_{\text{real}} &= \boldsymbol{V}\boldsymbol{T}_v, \\ \boldsymbol{W}_{\text{real}} &= \boldsymbol{W}\boldsymbol{T}_w. \end{aligned} \tag{4.32}$$

Hereby, $\boldsymbol{T}_v$ is the transformation matrix corresponding to the structure of $\boldsymbol{S}_v$ and $\boldsymbol{T}_w$ is the transformation matrix corresponding to the structure of $\boldsymbol{S}_w$. Note that it is not possible to compute the projection matrix with transformed interpolation data by use ot the presented *Arnoldi* algorithms as we loose the diagonal structure of $\boldsymbol{S}_v$ which is crucial for computing each column independently.

*Remark* 4.13 (Efficient Computing Of Columns Corresponding To Complex Conjugated Pairs). Looking at the transformation matrix we can see that the columns corresponding to complex conjugated interpolation points also have to be complex conjugated if it is possible to make the projection matrix real. Hence, it is possible to compute only one column of the projection matrix corresponding to two shifts which are complex conjugated. The other column could then simply be obtained by conjugating the previous column which is significantly faster than solving an additional system of equations. △

## 4.4 $\mathcal{H}_2$-Optimal Reduction

To complete this chapter, we want to discuss $\mathcal{H}_2$-optimal model reduction. Hereby, our goal is to reduce a bilinear system $\boldsymbol{\zeta}$ such that the reduced model $\boldsymbol{\zeta}_{\mathrm{r}}$ minimizes the following quality criterion

$$E = \|\boldsymbol{\zeta} - \boldsymbol{\zeta}_{\mathrm{r}}\|_{\mathcal{H}_2}^2.$$

To compute $\boldsymbol{\zeta}_{\mathrm{r}}$, we first have to define the necessary conditions which $\boldsymbol{\zeta}_{\mathrm{r}}$ has to fulfill for $\mathcal{H}_2$-optimality. After that we show that it is possible to obtain an $\mathcal{H}_2$-optimal ROM by using the *Volterra* series interpolation. Finally, we derive an iterative method which, assuming it converges, yields an $\mathcal{H}_2$-optimal ROM.

### Error System And Quality Criterion

To minimize our quality criterion $E$ we first have to find a meaningful expression for $E$. Therefore, we briefly recapitulate the error system and after that derive an equation for $E$ which depends on $\boldsymbol{\Lambda}_{\mathrm{r}}$, $\tilde{\boldsymbol{B}}$, $\tilde{\boldsymbol{C}}$ and $\tilde{\boldsymbol{N}}_{\mathrm{r},j}$, the system matrices of the diagonalized ROM. As mentioned, it is possible to write

$$E = \|\boldsymbol{\zeta} - \boldsymbol{\zeta}_{\mathrm{r}}\|_{\mathcal{H}_2}^2 = \|\boldsymbol{\zeta}_{\mathrm{err}}\|_{\mathcal{H}_2}^2 = \mathrm{tr}\left(\boldsymbol{C}_{\mathrm{err}}\boldsymbol{P}_{\mathrm{err}}\boldsymbol{C}_{\mathrm{err}}^{\mathsf{T}}\right).$$

Assuming that we can diagonalize the reduced system we write $\boldsymbol{E}_{\mathrm{r}}^{-1}\boldsymbol{A}_{\mathrm{r}} = \boldsymbol{X}_{\mathrm{r}}\boldsymbol{\Lambda}_{\mathrm{r}}\boldsymbol{X}_{\mathrm{r}}^{-1}$ where $\boldsymbol{X}_{\mathrm{r}}$ holds the right eigenvectors of the pair $\boldsymbol{E}_{\mathrm{r}}^{-1}\boldsymbol{A}_{\mathrm{r}}$ and $\boldsymbol{\Lambda}_{\mathrm{r}}$ the corresponding distinct eigenvalues. With this we can transform $\boldsymbol{\zeta}_{\mathrm{r}}$ in diagonal form and the matrices of the error system are given by

$$\boldsymbol{E}_{\mathrm{err}} = \begin{bmatrix}\boldsymbol{E} & \boldsymbol{0}\\ \boldsymbol{0} & \mathbf{I}_{\mathrm{r}}\end{bmatrix}, \quad \boldsymbol{A}_{\mathrm{err}} = \begin{bmatrix}\boldsymbol{A} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{\Lambda}_{\mathrm{r}}\end{bmatrix}, \quad \boldsymbol{N}_{\mathrm{err},j} = \begin{bmatrix}\boldsymbol{N}_j & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{X}_{\mathrm{r}}^{-1}\boldsymbol{E}_{\mathrm{r}}^{-1}\boldsymbol{N}_{\mathrm{r},j}\boldsymbol{X}_{\mathrm{r}}\end{bmatrix} = \begin{bmatrix}\boldsymbol{N}_j & \boldsymbol{0}\\ \boldsymbol{0} & \hat{\boldsymbol{N}}_{\mathrm{r},j}\end{bmatrix}$$

$$\boldsymbol{B}_{\mathrm{err}} = \begin{bmatrix}\boldsymbol{B}\\ \boldsymbol{X}_{\mathrm{r}}^{-1}\boldsymbol{E}_{\mathrm{r}}^{-1}\boldsymbol{B}_{\mathrm{r}}\end{bmatrix} = \begin{bmatrix}\boldsymbol{B}\\ \hat{\boldsymbol{B}}_{\mathrm{r}}\end{bmatrix}, \quad \boldsymbol{C}_{\mathrm{err}} = \begin{bmatrix}\boldsymbol{C} & -\boldsymbol{C}_{\mathrm{r}}\boldsymbol{X}_{\mathrm{r}}\end{bmatrix} = \begin{bmatrix}\boldsymbol{C} & -\hat{\boldsymbol{C}}_{\mathrm{r}}\end{bmatrix}.$$

While assuming that $\boldsymbol{\zeta}_{\mathrm{err}}$ is controllable $\boldsymbol{P}_{\mathrm{err}}$ follows by solving

$$\boldsymbol{A}_{\mathrm{err}}\boldsymbol{P}_{\mathrm{err}}\boldsymbol{E}_{\mathrm{err}}^{\mathsf{T}} + \boldsymbol{E}_{\mathrm{err}}\boldsymbol{P}_{\mathrm{err}}\boldsymbol{A}_{\mathrm{err}}^{\mathsf{T}} + \sum_{j=1}^{m}\boldsymbol{N}_{\mathrm{err},j}\boldsymbol{P}\boldsymbol{N}_{\mathrm{err},j}^{\mathsf{T}} = -\boldsymbol{B}_{\mathrm{err}}\boldsymbol{B}_{\mathrm{err}}^{\mathsf{T}}$$

with vectorization such that

$$\mathrm{vec}\left(\boldsymbol{P}_{\mathrm{err}}\right) = -\left(\boldsymbol{E}_{\mathrm{err}}\otimes\boldsymbol{A}_{\mathrm{err}} + \boldsymbol{A}_{\mathrm{err}}\otimes\boldsymbol{E}_{\mathrm{err}} + \sum_{j=1}^{m}\boldsymbol{N}_{\mathrm{err},j}\otimes\boldsymbol{N}_{\mathrm{err},j}\right)^{-1}\mathrm{vec}\left(\boldsymbol{B}_{\mathrm{err}}\boldsymbol{B}_{\mathrm{err}}^{\mathsf{T}}\right).$$

To obtain an expression for $E$ which contains the result for $\mathrm{vec}\left(\boldsymbol{P}_{\mathrm{err}}\right)$ we follow [5] and use the following properties of the trace of a matrix

$$\begin{aligned}\mathrm{tr}\left(\boldsymbol{ABC}\right) &= \mathrm{tr}\left(\boldsymbol{CAB}\right) = \mathrm{tr}\left(\boldsymbol{BCA}\right),\\ \mathrm{tr}\left(\boldsymbol{AC}\right) &= \mathrm{vec}\left(\boldsymbol{A}^{\mathsf{T}}\right)^{\mathsf{T}}\mathrm{vec}\left(\boldsymbol{C}\right).\end{aligned}$$

Hence, we write

$$E = \|\boldsymbol{\zeta}_{\mathrm{err}}\|_{\mathcal{H}_2}^2 = \mathrm{tr}\left(\boldsymbol{C}_{\mathrm{err}}\boldsymbol{P}_{\mathrm{err}}\boldsymbol{C}_{\mathrm{err}}^{\mathsf{T}}\right) = \mathrm{tr}\left(\boldsymbol{C}_{\mathrm{err}}^{\mathsf{T}}\boldsymbol{C}_{\mathrm{err}}\boldsymbol{P}_{\mathrm{err}}\right).$$

With the second property we obtain

$$E = \mathrm{tr}\left(\boldsymbol{C}_{\mathrm{err}}^{\mathsf{T}}\boldsymbol{C}_{\mathrm{err}}\boldsymbol{P}_{\mathrm{err}}\right) = \mathrm{vec}\left(\left(\boldsymbol{C}_{\mathrm{err}}^{\mathsf{T}}\boldsymbol{C}_{\mathrm{err}}\right)^{\mathsf{T}}\right)^{\mathsf{T}}\mathrm{vec}\left(\boldsymbol{P}_{\mathrm{err}}\right) = \mathrm{vec}\left(\boldsymbol{C}_{\mathrm{err}}^{\mathsf{T}}\boldsymbol{C}_{\mathrm{err}}\right)^{\mathsf{T}}\mathrm{vec}\left(\boldsymbol{P}_{\mathrm{err}}\right).$$

In the next step we substitute $\mathrm{vec}(\boldsymbol{P}_{\mathrm{err}})$ and consequently $E$ is given by

$$E = \mathrm{vec}\left(\boldsymbol{C}_{\mathrm{err}}^{\mathsf{T}}\boldsymbol{C}_{\mathrm{err}}\right)^{\mathsf{T}} \left(-\boldsymbol{E}_{\mathrm{err}} \otimes \boldsymbol{A}_{\mathrm{err}} - \boldsymbol{A}_{\mathrm{err}} \otimes \boldsymbol{E}_{\mathrm{err}} - \sum_{j=1}^{m} \boldsymbol{N}_{\mathrm{err},j} \otimes \boldsymbol{N}_{\mathrm{err},j}\right)^{-1} \mathrm{vec}\left(\boldsymbol{B}_{\mathrm{err}}\boldsymbol{B}_{\mathrm{err}}^{\mathsf{T}}\right).$$

With Definition 3.2 it is possible to exclude $\boldsymbol{C}_{\mathrm{err}}$ and $\boldsymbol{B}_{\mathrm{err}}$ from the vectorization operator such that

$$\begin{aligned}
\mathrm{vec}\left(\boldsymbol{C}_{\mathrm{err}}^{\mathsf{T}}\boldsymbol{C}_{\mathrm{err}}\right)^{\mathsf{T}} &= \mathrm{vec}\left(\boldsymbol{C}_{\mathrm{err}}^{\mathsf{T}}\,\mathbf{I}_p\boldsymbol{C}_{\mathrm{err}}\right)^{\mathsf{T}} = \left(\left(\boldsymbol{C}_{\mathrm{err}}^{\mathsf{T}} \otimes \boldsymbol{C}_{\mathrm{err}}^{\mathsf{T}}\right)\mathrm{vec}(\mathbf{I}_p)\right)^{\mathsf{T}} = \mathrm{vec}(\mathbf{I}_p)^{\mathsf{T}}\left(\boldsymbol{C}_{\mathrm{err}} \otimes \boldsymbol{C}_{\mathrm{err}}\right),\\
\mathrm{vec}\left(\boldsymbol{B}_{\mathrm{err}}\boldsymbol{B}_{\mathrm{err}}^{\mathsf{T}}\right) &= \mathrm{vec}\left(\boldsymbol{B}_{\mathrm{err}}\,\mathbf{I}_m\boldsymbol{B}_{\mathrm{err}}^{\mathsf{T}}\right) = \left(\boldsymbol{B}_{\mathrm{err}} \otimes \boldsymbol{B}_{\mathrm{err}}\right)\mathrm{vec}(\mathbf{I}_m)
\end{aligned}$$

with which we can rewrite $E$ as follows

$$\begin{aligned}
E &= \mathrm{vec}(\mathbf{I}_p)^{\mathsf{T}}\left(\boldsymbol{C}_{\mathrm{err}} \otimes \boldsymbol{C}_{\mathrm{err}}\right)\left(-\boldsymbol{E}_{\mathrm{err}} \otimes \boldsymbol{A}_{\mathrm{err}} - \boldsymbol{A}_{\mathrm{err}} \otimes \boldsymbol{E}_{\mathrm{err}} - \sum_{j=1}^{m} \boldsymbol{N}_{\mathrm{err},j} \otimes \boldsymbol{N}_{\mathrm{err},j}\right)^{-1}\\
&\qquad \times \left(\boldsymbol{B}_{\mathrm{err}} \otimes \boldsymbol{B}_{\mathrm{err}}\right)\mathrm{vec}(\mathbf{I}_m)\\
&= \mathrm{vec}(\mathbf{I}_p)^{\mathsf{T}}\left(\begin{bmatrix}\boldsymbol{C} & -\hat{\boldsymbol{C}}_{\mathrm{r}}\end{bmatrix} \otimes \begin{bmatrix}\boldsymbol{C} & -\hat{\boldsymbol{C}}_{\mathrm{r}}\end{bmatrix}\right)\Bigg(-\begin{bmatrix}\boldsymbol{E} & \mathbf{0}\\ \mathbf{0} & \mathbf{I}_{\mathrm{r}}\end{bmatrix} \otimes \begin{bmatrix}\boldsymbol{A} & \mathbf{0}\\ \mathbf{0} & \boldsymbol{\Lambda}_{\mathrm{r}}\end{bmatrix} - \begin{bmatrix}\boldsymbol{A} & \mathbf{0}\\ \mathbf{0} & \boldsymbol{\Lambda}_{\mathrm{r}}\end{bmatrix} \otimes \begin{bmatrix}\boldsymbol{E} & \mathbf{0}\\ \mathbf{0} & \mathbf{I}_{\mathrm{r}}\end{bmatrix}\\
&\qquad - \sum_{j=1}^{m}\begin{bmatrix}\boldsymbol{N}_j & \mathbf{0}\\ \mathbf{0} & \hat{\boldsymbol{N}}_{\mathrm{r},j}\end{bmatrix} \otimes \begin{bmatrix}\boldsymbol{N}_j & \mathbf{0}\\ \mathbf{0} & \hat{\boldsymbol{N}}_{\mathrm{r},j}\end{bmatrix}\Bigg)^{-1}\left(\begin{bmatrix}\boldsymbol{B}\\ \hat{\boldsymbol{B}}_{\mathrm{r}}\end{bmatrix} \otimes \begin{bmatrix}\boldsymbol{B}\\ \hat{\boldsymbol{B}}_{\mathrm{r}}\end{bmatrix}\right)\mathrm{vec}(\mathbf{I}_m).
\end{aligned} \tag{4.33}$$

With (4.33) we have determined an expression for $E$ which in fact depends on $\boldsymbol{\Lambda}_{\mathrm{r}}$, $\tilde{\boldsymbol{B}}$, $\tilde{\boldsymbol{C}}$ and $\tilde{\boldsymbol{N}}_{\mathrm{r},j}$ and makes it possible to obtain the necessary optimality conditions [5].

### First Order Necessary Conditions

Since we try to find the minimum of $E$ it is obvious that we have to derive $E$ with respect to the optimization parameters $\boldsymbol{\Lambda}_{\mathrm{r}}$, $\tilde{\boldsymbol{B}}$, $\tilde{\boldsymbol{C}}$ and $\tilde{\boldsymbol{N}}_{\mathrm{r},j}$. We do not want to show the derivation as it is illustrated in [5, Lemma 4.3.1] starting from (4.33). As a result $\boldsymbol{\zeta}_{\mathrm{r}}$ has to fulfill following four conditions to ensure $\mathcal{H}_2$-optimal reduction. The first condition follows by $\frac{\partial E}{\partial \tilde{\boldsymbol{C}}_{\mathrm{r}}^{(i,ii)}} = 0$ where $\tilde{\boldsymbol{C}}_{\mathrm{r}}^{(i,ii)}$ denotes the entry in the $i$-th row and $ii$-th column of $\tilde{\boldsymbol{C}}_{\mathrm{r}}$. Note, that in the following equations we use $\boldsymbol{e}_i$ which is a unit vector with one in the $i$-th position. To keep the notation simpler we suppose that the size of the vectors is clear by the context. Hence, the first condition is given by

$$\begin{gathered}
\mathrm{vec}(\mathbf{I}_p)^{\mathsf{T}}\left(\boldsymbol{e}_i\boldsymbol{e}_{ii}^{\mathsf{T}} \otimes \boldsymbol{C}\right)\left(-\boldsymbol{\Lambda}_{\mathrm{r}} \otimes \boldsymbol{E} - \mathbf{I}_{\mathrm{r}} \otimes \boldsymbol{A} - \sum_{j=1}^{m}\hat{\boldsymbol{N}}_{\mathrm{r},j} \otimes \boldsymbol{N}_j\right)^{-1}\left(\hat{\boldsymbol{B}}_{\mathrm{r}} \otimes \boldsymbol{B}\right)\mathrm{vec}(\mathbf{I}_m)\\
\equiv\\
\mathrm{vec}(\mathbf{I}_p)^{\mathsf{T}}\left(\boldsymbol{e}_i\boldsymbol{e}_{ii}^{\mathsf{T}} \otimes \hat{\boldsymbol{C}}_{\mathrm{r}}\right)\left(-\boldsymbol{\Lambda}_{\mathrm{r}} \otimes \mathbf{I}_{\mathrm{r}} - \mathbf{I}_{\mathrm{r}} \otimes \boldsymbol{\Lambda}_{\mathrm{r}} - \sum_{j=1}^{m}\hat{\boldsymbol{N}}_{\mathrm{r},j} \otimes \hat{\boldsymbol{N}}_{\mathrm{r},j}\right)^{-1}\left(\hat{\boldsymbol{B}}_{\mathrm{r}} \otimes \hat{\boldsymbol{B}}_{\mathrm{r}}\right)\mathrm{vec}(\mathbf{I}_m).
\end{gathered} \tag{4.34}$$

We obtain the second condition by $\frac{\partial E}{\partial \tilde{\boldsymbol{B}}_{\mathrm{r}}^{(i,ii)}} = 0$

$$\operatorname{vec}(\mathbf{I}_p)^{\mathsf{T}} \left(\hat{\boldsymbol{C}}_{\mathrm{r}} \otimes \boldsymbol{C}\right) \left(-\boldsymbol{\Lambda}_{\mathrm{r}} \otimes \boldsymbol{E} - \mathbf{I}_{\mathrm{r}} \otimes \boldsymbol{A} - \sum_{j=1}^{m} \hat{\boldsymbol{N}}_{\mathrm{r},j} \otimes \boldsymbol{N}_j\right)^{-1} \left(\boldsymbol{e}_i \boldsymbol{e}_{ii}^{\mathsf{T}} \otimes \boldsymbol{B}\right) \operatorname{vec}(\mathbf{I}_m)$$
$$\equiv$$
$$\operatorname{vec}(\mathbf{I}_p)^{\mathsf{T}} \left(\hat{\boldsymbol{C}}_{\mathrm{r}} \otimes \hat{\boldsymbol{C}}_{\mathrm{r}}\right) \left(-\boldsymbol{\Lambda}_{\mathrm{r}} \otimes \mathbf{I}_{\mathrm{r}} - \mathbf{I}_{\mathrm{r}} \otimes \boldsymbol{\Lambda}_{\mathrm{r}} - \sum_{j=1}^{m} \hat{\boldsymbol{N}}_{\mathrm{r},j} \otimes \hat{\boldsymbol{N}}_{\mathrm{r},j}\right)^{-1} \left(\boldsymbol{e}_i \boldsymbol{e}_{ii}^{\mathsf{T}} \otimes \hat{\boldsymbol{B}}_{\mathrm{r}}\right) \operatorname{vec}(\mathbf{I}_m). \tag{4.35}$$

The third condition is the result of $\frac{\partial E}{\partial \lambda_{\mathrm{r},i}} = 0$ where $\lambda_{\mathrm{r},i}$ denotes the eigenvalues of $\boldsymbol{\Lambda}_{\mathrm{r}}$

$$\begin{aligned}
&\operatorname{vec}(\mathbf{I}_p)^{\mathsf{T}} \left(\hat{\boldsymbol{C}}_{\mathrm{r}} \otimes \boldsymbol{C}\right) \left(-\boldsymbol{\Lambda}_{\mathrm{r}} \otimes \boldsymbol{E} - \mathbf{I}_{\mathrm{r}} \otimes \boldsymbol{A} - \sum_{j=1}^{m} \hat{\boldsymbol{N}}_{\mathrm{r},j} \otimes \boldsymbol{N}_j\right)^{-1} \\
&\quad \times \boldsymbol{e}_i \boldsymbol{e}_i^{\mathsf{T}} \otimes \boldsymbol{E} \left(-\boldsymbol{\Lambda}_{\mathrm{r}} \otimes \boldsymbol{E} - \mathbf{I}_{\mathrm{r}} \otimes \boldsymbol{A} - \sum_{j=1}^{m} \hat{\boldsymbol{N}}_{\mathrm{r},j} \otimes \boldsymbol{N}_j\right)^{-1} \left(\hat{\boldsymbol{B}}_{\mathrm{r}} \otimes \boldsymbol{B}\right) \operatorname{vec}(\mathbf{I}_m) \\
&\equiv \\
&\operatorname{vec}(\mathbf{I}_p)^{\mathsf{T}} \left(\hat{\boldsymbol{C}}_{\mathrm{r}} \otimes \hat{\boldsymbol{C}}_{\mathrm{r}}\right) \left(-\boldsymbol{\Lambda}_{\mathrm{r}} \otimes \mathbf{I}_{\mathrm{r}} - \mathbf{I}_{\mathrm{r}} \otimes \boldsymbol{\Lambda}_{\mathrm{r}} - \sum_{j=1}^{m} \hat{\boldsymbol{N}}_{\mathrm{r},j} \otimes \hat{\boldsymbol{N}}_{\mathrm{r},j}\right)^{-1} \\
&\quad \times \boldsymbol{e}_i \boldsymbol{e}_i^{\mathsf{T}} \otimes \mathbf{I}_{\mathrm{r}} \left(-\boldsymbol{\Lambda}_{\mathrm{r}} \otimes \mathbf{I}_{\mathrm{r}} - \mathbf{I}_{\mathrm{r}} \otimes \boldsymbol{\Lambda}_{\mathrm{r}} - \sum_{j=1}^{m} \hat{\boldsymbol{N}}_{\mathrm{r},j} \otimes \hat{\boldsymbol{N}}_{\mathrm{r},j}\right)^{-1} \left(\hat{\boldsymbol{B}}_{\mathrm{r}} \otimes \hat{\boldsymbol{B}}_{\mathrm{r}}\right) \operatorname{vec}(\mathbf{I}_m).
\end{aligned} \tag{4.36}$$

Finally, the fourth condition is derived from $\frac{\partial E}{\partial \tilde{\boldsymbol{N}}_{\mathrm{r},j}^{(i,ii)}} = 0$

$$\begin{aligned}
&\operatorname{vec}(\mathbf{I}_p)^{\mathsf{T}} \left(\hat{\boldsymbol{C}}_{\mathrm{r}} \otimes \boldsymbol{C}\right) \left(-\boldsymbol{\Lambda}_{\mathrm{r}} \otimes \boldsymbol{E} - \mathbf{I}_{\mathrm{r}} \otimes \boldsymbol{A} - \sum_{j=1}^{m} \hat{\boldsymbol{N}}_{\mathrm{r},j} \otimes \boldsymbol{N}_j\right)^{-1} \\
&\quad \times \boldsymbol{e}_i \boldsymbol{e}_{ii}^{\mathsf{T}} \otimes \boldsymbol{N}_j \left(-\boldsymbol{\Lambda}_{\mathrm{r}} \otimes \boldsymbol{E} - \mathbf{I}_{\mathrm{r}} \otimes \boldsymbol{A} - \sum_{j=1}^{m} \hat{\boldsymbol{N}}_{\mathrm{r},j} \otimes \boldsymbol{N}_j\right)^{-1} \left(\hat{\boldsymbol{B}}_{\mathrm{r}} \otimes \boldsymbol{B}\right) \operatorname{vec}(\mathbf{I}_m) \\
&\equiv \\
&\operatorname{vec}(\mathbf{I}_p)^{\mathsf{T}} \left(\hat{\boldsymbol{C}}_{\mathrm{r}} \otimes \hat{\boldsymbol{C}}_{\mathrm{r}}\right) \left(-\boldsymbol{\Lambda}_{\mathrm{r}} \otimes \mathbf{I}_{\mathrm{r}} - \mathbf{I}_{\mathrm{r}} \otimes \boldsymbol{\Lambda}_{\mathrm{r}} - \sum_{j=1}^{m} \hat{\boldsymbol{N}}_{\mathrm{r},j} \otimes \hat{\boldsymbol{N}}_{\mathrm{r},j}\right)^{-1} \\
&\quad \times \boldsymbol{e}_i \boldsymbol{e}_{ii}^{\mathsf{T}} \otimes \hat{\boldsymbol{N}}_{\mathrm{r},j} \left(-\boldsymbol{\Lambda}_{\mathrm{r}} \otimes \mathbf{I}_{\mathrm{r}} - \mathbf{I}_{\mathrm{r}} \otimes \boldsymbol{\Lambda}_{\mathrm{r}} - \sum_{j=1}^{m} \hat{\boldsymbol{N}}_{\mathrm{r},j} \otimes \hat{\boldsymbol{N}}_{\mathrm{r},j}\right)^{-1} \left(\hat{\boldsymbol{B}}_{\mathrm{r}} \otimes \hat{\boldsymbol{B}}_{\mathrm{r}}\right) \operatorname{vec}(\mathbf{I}_m).
\end{aligned} \tag{4.37}$$

Concluding, if a reduced bilinear system $\boldsymbol{\zeta}_{\mathrm{r}}$ fulfills the conditions (4.34), (4.35), (4.36) and (4.37) then $\boldsymbol{\zeta}_{\mathrm{r}}$ locally minimizes $E$ and vice versa.

## Constructing An $\mathcal{H}_2$-Optimal Reduced System

From the structure of the optimality conditions one might notice a link with the *Volterra* series interpolation framework. To clarify this link we expand the left-hand-side of (4.34) as a *Neumann* series. In addition to that we write $\left(\tilde{\boldsymbol{B}}_\mathrm{r} \otimes \boldsymbol{B}\right) \mathrm{vec}(\mathbf{I}_m)$ as $\boldsymbol{B}\tilde{\boldsymbol{B}}_\mathrm{r}^\mathsf{T}$ which yields

$$\begin{aligned}
&\mathrm{vec}(\mathbf{I}_p)^\mathsf{T} \left(\boldsymbol{e}_i \boldsymbol{e}_{ii}^\mathsf{T} \otimes \boldsymbol{C}\right) \left(-\boldsymbol{\Lambda}_\mathrm{r} \otimes \boldsymbol{E} - \mathbf{I}_\mathrm{r} \otimes \boldsymbol{A} - \sum_{j=1}^{m} \hat{\boldsymbol{N}}_{\mathrm{r},j} \otimes \boldsymbol{N}_j\right)^{-1} \left(\hat{\boldsymbol{B}}_\mathrm{r} \otimes \boldsymbol{B}\right) \mathrm{vec}(\mathbf{I}_m)\\
&= \sum_{k=0}^{\infty} \mathrm{vec}(\mathbf{I}_p)^\mathsf{T} \left(\boldsymbol{e}_i \boldsymbol{e}_{ii}^\mathsf{T} \otimes \boldsymbol{C}\right) \left((-\boldsymbol{\Lambda}_\mathrm{r} \otimes \boldsymbol{E} - \mathbf{I}_\mathrm{r} \otimes \boldsymbol{A})^{-1} \sum_{j=1}^{m} \hat{\boldsymbol{N}}_{\mathrm{r},j} \otimes \boldsymbol{N}_j\right)^k\\
&\qquad \times (-\boldsymbol{\Lambda}_\mathrm{r} \otimes \boldsymbol{E} - \mathbf{I}_\mathrm{r} \otimes \boldsymbol{A})^{-1} \boldsymbol{B}\hat{\boldsymbol{B}}_\mathrm{r}^\mathsf{T}.
\end{aligned}$$

If we define $\boldsymbol{R} = \hat{\boldsymbol{B}}^\mathsf{T}$ and write the matrices explicitly

$$\begin{aligned}
&= \sum_{k=0}^{\infty} \mathrm{vec}(\mathbf{I}_p)^\mathsf{T} \left(\boldsymbol{e}_i \boldsymbol{e}_{ii}^\mathsf{T} \otimes \boldsymbol{C}\right) \left(\begin{bmatrix} (-\lambda_1 \boldsymbol{E} - \boldsymbol{A})^{-1} & & \\ & \ddots & \\ & & (-\lambda_r \boldsymbol{E} - \boldsymbol{A})^{-1} \end{bmatrix} \sum_{j=1}^{m} \begin{bmatrix} \hat{n}_{\boldsymbol{r},j}^{(1,1)} \boldsymbol{N}_j & \cdots & \hat{n}_{\boldsymbol{r},j}^{(1,r)} \boldsymbol{N}_j \\ \vdots & & \vdots \\ \hat{n}_{\boldsymbol{r},j}^{(r,1)} \boldsymbol{N}_j & \cdots & \hat{n}_{\boldsymbol{r},j}^{(r,r)} \boldsymbol{N}_j \end{bmatrix}\right)^k\\
&\qquad \begin{bmatrix} (-\lambda_1 \boldsymbol{E} - \boldsymbol{A})^{-1} & & \\ & \ddots & \\ & & (-\lambda_r \boldsymbol{E} - \boldsymbol{A})^{-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{B}\boldsymbol{r}_1 \\ \vdots \\ \boldsymbol{B}\boldsymbol{r}_r \end{bmatrix}
\end{aligned}$$

we can determine the well-known structure from the vectorized *Sylvester* equations. In addition to that, if we look at each $i - ii$-combination and apply the vectorization backwards it follows that the above is equal to

$$\sum_{k=1}^{\infty} \sum_{j_2=1}^{m} \cdots \sum_{j_k=1}^{m} \sum_{l_1=1}^{r} \cdots \sum_{l_{k-1}=1}^{r} \eta_{l_1,\ldots,l_{k-1},i}^{j_2,\ldots,j_k} \, \boldsymbol{G}_{k,\square}^{(j_2,\ldots,j_k)}(-\lambda_{l_1}, \ldots, -\lambda_{l_{k-1}}, -\lambda_i)\, \boldsymbol{r}_{l_1}$$

which apparently is a weighted series over all transfer functions of the FOM evaluated at $-\lambda_i$ for $i = 1, \ldots, r$ and multiplied by the corresponding tangential vectors. Hereby the weights are defined by $\boldsymbol{U}_{v,j} = \hat{\boldsymbol{N}}_{\mathrm{r},j}$ and the tangential directions by $\boldsymbol{R} = \hat{\boldsymbol{B}}_\mathrm{r}^\mathsf{T}$. We follow the same procedure for the right hand side in (4.34) and since the transfer function of a diagonalized system is equal to the regular transfer function ($\boldsymbol{G}_{\mathrm{diag},k,\square}^{(j_2,\ldots,j_k)}(s_1, \ldots, s_k) = \boldsymbol{G}_{k,\square}^{(j_2,\ldots,j_k)}(s_1, \ldots, s_k)$) we can conclude that (4.34) yields the following *Volterra* series interpolation moment matching condition

$$\begin{aligned}
&\sum_{k=1}^{\infty} \sum_{j_2=1}^{m} \cdots \sum_{j_k=1}^{m} \sum_{l_1=1}^{r} \cdots \sum_{l_{k-1}=1}^{r} \eta_{l_1,\ldots,l_{k-1},i}^{j_2,\ldots,j_k} \, \boldsymbol{G}_{k,\square}^{(j_2,\ldots,j_k)}(-\lambda_{l_1}, \ldots, -\lambda_{l_{k-1}}, -\lambda_i)\, \boldsymbol{r}_{l_1}\\
&\equiv \qquad (4.38)\\
&\sum_{k=1}^{\infty} \sum_{j_2=1}^{m} \cdots \sum_{j_k=1}^{m} \sum_{l_1=1}^{r} \cdots \sum_{l_{k-1}=1}^{r} \eta_{l_1,\ldots,l_{k-1},i}^{j_2,\ldots,j_k} \, \boldsymbol{G}_{k,\mathrm{r},\square}^{(j_2,\ldots,j_k)}(-\lambda_{l_1}, \ldots, -\lambda_{l_{k-1}}, -\lambda_i)\, \boldsymbol{r}_{l_1}
\end{aligned}$$

for $i = 1, \dots, r$.
We can apply this analogously to (4.35) and receive the following output *Volterra* series interpolation moment matching condition

$$\begin{aligned}&\sum_{k=1}^{\infty}\sum_{j_2=1}^{m}\cdots\sum_{j_k=1}^{m}\sum_{l_1=1}^{r}\cdots\sum_{l_{k-1}=1}^{r}\vartheta_{l_1,\dots,l_{k-1},i}^{j_2,\dots,j_k}\,\boldsymbol{l}_{l_1}^{\mathsf{T}}\,\boldsymbol{G}_{k,\square}^{(j_2,\dots,j_k)}(-\lambda_i,-\lambda_{l_{k-1}},\dots,-\lambda_{l_1})\\ &\equiv \\ &\sum_{k=1}^{\infty}\sum_{j_2=1}^{m}\cdots\sum_{j_k=1}^{m}\sum_{l_1=1}^{r}\cdots\sum_{l_{k-1}=1}^{r}\vartheta_{l_1,\dots,l_{k-1},i}^{j_2,\dots,j_k}\,\boldsymbol{l}_{l_1}^{\mathsf{T}}\,\boldsymbol{G}_{k,\mathrm{r},\square}^{(j_2,\dots,j_k)}(-\lambda_i,-\lambda_{l_{k-1}},\dots,-\lambda_{l_1}).\end{aligned} \tag{4.39}$$

Hereby, $i = 1, \dots, r$, the interpolation points are the same reflected eigenvalues as for (4.38), $\boldsymbol{L} = \hat{\boldsymbol{C}}_\mathrm{r}$ and the weights are defined by $\boldsymbol{U}_{w,j} = \hat{\boldsymbol{N}}_{\mathrm{r},j}^{\mathsf{T}}$. As we know from the *Volterra* series interpolation, we can fulfill (4.38) and (4.39) by projecting the FOM with the projection matrices $\boldsymbol{V}$ and $\boldsymbol{W}$. This projection matrices could be computed by solving following *Sylvester* equations

$$\begin{aligned}\boldsymbol{E}\,\boldsymbol{V}\,(-\boldsymbol{\Lambda}_\mathrm{r}) - \boldsymbol{A}\,\boldsymbol{V} - \sum_{j=1}^{m}\boldsymbol{N}_j\,\boldsymbol{V}\,\hat{\boldsymbol{N}}_{\mathrm{r},j}^{\mathsf{T}} &= \boldsymbol{B}\,\hat{\boldsymbol{B}}_\mathrm{r}^{\mathsf{T}}\\ \boldsymbol{E}^{\mathsf{T}}\,\boldsymbol{W}\,(-\boldsymbol{\Lambda}_\mathrm{r})^{\mathsf{T}} - \boldsymbol{A}^{\mathsf{T}}\,\boldsymbol{W} - \sum_{j=1}^{m}\boldsymbol{N}_j^{\mathsf{T}}\,\boldsymbol{W}\,\hat{\boldsymbol{N}}_{\mathrm{r},j} &= \boldsymbol{C}^{\mathsf{T}}\,\hat{\boldsymbol{C}}_\mathrm{r}.\end{aligned} \tag{4.40}$$

To ensure $\mathcal{H}_2$-optimality we have to prove that the construction of the ROM with $\boldsymbol{V}$ and $\boldsymbol{W}$ additionally fulfills (4.36) and (4.37). Therefore, we write (4.36) as

$$\begin{aligned}&\underbrace{\mathrm{vec}(\mathbf{I}_p)^{\mathsf{T}}\left(\hat{\boldsymbol{C}}_\mathrm{r}\otimes\boldsymbol{C}\right)\left(-\boldsymbol{\Lambda}_\mathrm{r}\otimes\boldsymbol{E} - \mathbf{I}_\mathrm{r}\otimes\boldsymbol{A} - \sum_{j=1}^{m}\hat{\boldsymbol{N}}_{\mathrm{r},j}\otimes\boldsymbol{N}_j\right)^{-1}}_{=\mathrm{vec}(\boldsymbol{W})^{\mathsf{T}}}\\ &\times\,\boldsymbol{e}_i\boldsymbol{e}_i^{\mathsf{T}}\otimes\boldsymbol{E}\underbrace{\left(-\boldsymbol{\Lambda}_\mathrm{r}\otimes\boldsymbol{E} - \mathbf{I}_\mathrm{r}\otimes\boldsymbol{A} - \sum_{j=1}^{m}\hat{\boldsymbol{N}}_{\mathrm{r},j}\otimes\boldsymbol{N}_j\right)^{-1}\left(\hat{\boldsymbol{B}}_\mathrm{r}\otimes\boldsymbol{B}\right)\mathrm{vec}(\mathbf{I}_m)}_{=\mathrm{vec}(\boldsymbol{V})}\\ &\equiv\\ &\underbrace{\mathrm{vec}(\mathbf{I}_p)^{\mathsf{T}}\left(\hat{\boldsymbol{C}}_\mathrm{r}\otimes\hat{\boldsymbol{C}}_\mathrm{r}\right)\left(-\boldsymbol{\Lambda}_\mathrm{r}\otimes\mathbf{I}_\mathrm{r} - \mathbf{I}_\mathrm{r}\otimes\boldsymbol{\Lambda}_\mathrm{r} - \sum_{j=1}^{m}\hat{\boldsymbol{N}}_{\mathrm{r},j}\otimes\hat{\boldsymbol{N}}_{\mathrm{r},j}\right)^{-1}}_{=\mathrm{vec}(\boldsymbol{W}_\mathrm{r})^{\mathsf{T}}}\\ &\times\,\boldsymbol{e}_i\boldsymbol{e}_i^{\mathsf{T}}\otimes\mathbf{I}_\mathrm{r}\underbrace{\left(-\boldsymbol{\Lambda}_\mathrm{r}\otimes\mathbf{I}_\mathrm{r} - \mathbf{I}_\mathrm{r}\otimes\boldsymbol{\Lambda}_\mathrm{r} - \sum_{j=1}^{m}\hat{\boldsymbol{N}}_{\mathrm{r},j}\otimes\hat{\boldsymbol{N}}_{\mathrm{r},j}\right)^{-1}\left(\hat{\boldsymbol{B}}_\mathrm{r}\otimes\hat{\boldsymbol{B}}_\mathrm{r}\right)\mathrm{vec}(\mathbf{I}_m)}_{=\mathrm{vec}(\boldsymbol{V}_\mathrm{r})}\\ &\Rightarrow \mathrm{vec}\,(\boldsymbol{W})^{\mathsf{T}}\left(\boldsymbol{e}_i\boldsymbol{e}_i^{\mathsf{T}}\otimes\boldsymbol{E}\right)\mathrm{vec}\,(\boldsymbol{V}) \equiv \mathrm{vec}\,(\boldsymbol{W}_\mathrm{r})^{\mathsf{T}}\left(\boldsymbol{e}_i\boldsymbol{e}_i^{\mathsf{T}}\otimes\mathbf{I}_\mathrm{r}\right)\mathrm{vec}\,(\boldsymbol{V}_\mathrm{r})\end{aligned}$$

where we can see that the following link must hold to fulfill the equation

$$\mathrm{vec}\,(\boldsymbol{V}) = (\mathbf{I}_\mathrm{r}\otimes\boldsymbol{V}\boldsymbol{X})\,\mathrm{vec}\,(\boldsymbol{V}_\mathrm{r}), \tag{4.41}$$

$$\mathrm{vec}\,(\boldsymbol{W}) = \left(\mathbf{I}_\mathrm{r}\otimes\boldsymbol{W}\boldsymbol{E}_r^{-\mathsf{T}}\boldsymbol{X}_\mathrm{r}^{-\mathsf{T}}\right)\mathrm{vec}\,(\boldsymbol{W}_\mathrm{r}). \tag{4.42}$$

Let us write the equations for vec $(\boldsymbol{V}_\mathrm{r})$ and vec $(\boldsymbol{V})$ without inverting the terms in the brackets

$$\left(-\boldsymbol{\Lambda}_\mathrm{r} \otimes \mathbf{I}_\mathrm{r} - \mathbf{I}_\mathrm{r} \otimes \boldsymbol{\Lambda}_\mathrm{r} - \sum_{j=1}^{m} \hat{\boldsymbol{N}}_{\mathrm{r},j} \otimes \hat{\boldsymbol{N}}_{\mathrm{r},j}\right) \operatorname{vec}(\boldsymbol{V}_\mathrm{r}) = \left(\hat{\boldsymbol{B}}_\mathrm{r} \otimes \hat{\boldsymbol{B}}_\mathrm{r}\right) \operatorname{vec}(\mathbf{I}_m), \tag{4.43}$$

$$\left(-\boldsymbol{\Lambda}_\mathrm{r} \otimes \boldsymbol{E} - \mathbf{I}_\mathrm{r} \otimes \boldsymbol{A} - \sum_{j=1}^{m} \hat{\boldsymbol{N}}_{\mathrm{r},j} \otimes \boldsymbol{N}_j\right) \operatorname{vec}(\boldsymbol{V}) = \left(\hat{\boldsymbol{B}}_\mathrm{r} \otimes \boldsymbol{B}\right) \operatorname{vec}(\mathbf{I}_m). \tag{4.44}$$

Assuming that (4.41) holds, we substitute vec $(\boldsymbol{V})$ in (4.44) which yields

$$\left(-\boldsymbol{\Lambda}_\mathrm{r} \otimes \boldsymbol{E} - \mathbf{I}_\mathrm{r} \otimes \boldsymbol{A} - \sum_{j=1}^{m} \hat{\boldsymbol{N}}_{\mathrm{r},j} \otimes \boldsymbol{N}_j\right) (\mathbf{I}_\mathrm{r} \otimes \boldsymbol{V}\boldsymbol{X}) \operatorname{vec}(\boldsymbol{V}_\mathrm{r}) = \left(\hat{\boldsymbol{B}}_\mathrm{r} \otimes \boldsymbol{B}\right) \operatorname{vec}(\mathbf{I}_m).$$

Multiplying the above from the left with $\mathbf{I}_\mathrm{r} \otimes \boldsymbol{X}_\mathrm{r}^{-1}\boldsymbol{E}^{-1}\boldsymbol{W}^\top$ indeed yields (4.43) as shown in the following

$$\begin{aligned}
&\left(\mathbf{I}_\mathrm{r} \otimes \boldsymbol{X}_\mathrm{r}^{-1}\boldsymbol{E}^{-1}\boldsymbol{W}^\top\right)\Bigg( - \boldsymbol{\Lambda}_\mathrm{r} \otimes \boldsymbol{E} - \mathbf{I}_\mathrm{r} \otimes \boldsymbol{A} \\
&\qquad\qquad - \sum_{j=1}^{m} \hat{\boldsymbol{N}}_{\mathrm{r},j} \otimes \boldsymbol{N}_j\Bigg) (\mathbf{I}_\mathrm{r} \otimes \boldsymbol{V}) \operatorname{vec}(\boldsymbol{V}_\mathrm{r}) = \left(\hat{\boldsymbol{B}}_\mathrm{r} \otimes \boldsymbol{B}\right) \operatorname{vec}(\mathbf{I}_m) \\
\Rightarrow &\Bigg( - \boldsymbol{\Lambda}_\mathrm{r} \otimes \boldsymbol{X}_\mathrm{r}^{-1}\boldsymbol{E}^{-1}\boldsymbol{W}^\top\boldsymbol{E}\boldsymbol{V} - \mathbf{I}_\mathrm{r} \otimes \boldsymbol{X}_\mathrm{r}^{-1}\boldsymbol{E}^{-1}\boldsymbol{W}^\top\boldsymbol{A}\boldsymbol{V} \\
&\quad - \sum_{j=1}^{m} \hat{\boldsymbol{N}}_{\mathrm{r},j} \otimes \boldsymbol{X}_\mathrm{r}^{-1}\boldsymbol{E}^{-1}\boldsymbol{W}^\top\boldsymbol{N}_j\boldsymbol{V}\Bigg)\operatorname{vec}(\boldsymbol{V}_\mathrm{r}) = \left(\hat{\boldsymbol{B}}_\mathrm{r} \otimes \boldsymbol{X}_\mathrm{r}^{-1}\boldsymbol{E}^{-1}\boldsymbol{W}^\top\boldsymbol{B}\right) \operatorname{vec}(\mathbf{I}_m) \\
\Rightarrow &\left(-\boldsymbol{\Lambda}_\mathrm{r} \otimes \mathbf{I}_\mathrm{r} - \mathbf{I}_\mathrm{r} \otimes \boldsymbol{\Lambda}_\mathrm{r} - \sum_{j=1}^{m} \hat{\boldsymbol{N}}_{\mathrm{r},j} \otimes \hat{\boldsymbol{N}}_{\mathrm{r},j}\right) \operatorname{vec}(\boldsymbol{V}_\mathrm{r}) = \left(\hat{\boldsymbol{B}}_\mathrm{r} \otimes \hat{\boldsymbol{B}}_\mathrm{r}\right) \operatorname{vec}(\mathbf{I}_m).
\end{aligned}$$

Consequently, (4.41) holds. We could write the equations for vec $(\boldsymbol{W}_\mathrm{r})$ and vec $(\boldsymbol{W})$ without the inverse

$$\left(-\boldsymbol{\Lambda}_\mathrm{r}^\top \otimes \mathbf{I}_\mathrm{r}^\top - \mathbf{I}_\mathrm{r} \otimes \boldsymbol{\Lambda}_\mathrm{r}^\top - \sum_{j=1}^{m} \hat{\boldsymbol{N}}_{\mathrm{r},j}^\top \otimes \hat{\boldsymbol{N}}_{\mathrm{r},j}^\top\right) \operatorname{vec}(\boldsymbol{W}_\mathrm{r}) = \left(\hat{\boldsymbol{C}}_\mathrm{r}^\top \otimes \hat{\boldsymbol{C}}_\mathrm{r}^\top\right) \operatorname{vec}(\mathbf{I}_p),$$

$$\left(-\boldsymbol{\Lambda}_\mathrm{r}^\top \otimes \boldsymbol{E}^\top - \mathbf{I}_\mathrm{r} \otimes \boldsymbol{A}^\top - \sum_{j=1}^{m} \hat{\boldsymbol{N}}_{\mathrm{r},j}^\top \otimes \boldsymbol{N}_j^\top\right) \operatorname{vec}(\boldsymbol{W}) = \left(\hat{\boldsymbol{C}}_\mathrm{r}^\top \otimes \boldsymbol{C}^\top\right) \operatorname{vec}(\mathbf{I}_p).$$

Substituting vec $(\boldsymbol{W})$ and multiplying the resulting equation from the left with $\boldsymbol{X}^\top\boldsymbol{V}^\top$ shows that (4.42) holds. Concluding, the computation of the projection matrices by (4.40) fulfills the necessary condition (4.36). Analogously to that, we could proof that (4.37) is fulfilled.

### Iterative Algorithm To Obtain An $\mathcal{H}_2$-Optimal Reduced System

Finally, we come up with an algorithm which yields the $\mathcal{H}_2$-optimal ROM. As pointed out in [12] it is possible to interpret the search for the optimal ROM as a search for roots of

$$\boldsymbol{g}(\boldsymbol{\sigma}) = \boldsymbol{\lambda}(\boldsymbol{\sigma}) + \boldsymbol{\sigma}$$

such that $\boldsymbol{g}(\boldsymbol{\sigma}) = \mathbf{0}$. Hereby, $\boldsymbol{\sigma} = \{\sigma_1, \dots, \sigma_r\}$ is a set of interpolation points and $\boldsymbol{\lambda}(\boldsymbol{\sigma}) = \{\lambda_1, \dots, \lambda_r\}$ contains the eigenvalues of the resulting ROM constructed with $\boldsymbol{\sigma}$ for an input and output *Volterra* series interpolation such that all four $\mathcal{H}_2$-optimality conditions are fulfilled. We approach this root finding problem by using *Newton's* method. Thus we formulate

$$\boldsymbol{\sigma}^{(k+1)} = \boldsymbol{\sigma}^{(k)} - (\mathbf{I}_r + \boldsymbol{J})\left(\boldsymbol{\sigma}^{(k)} + \boldsymbol{\lambda}(\boldsymbol{\sigma}^{(k)})\right) \tag{4.45}$$

where $\boldsymbol{J}$ denotes the *Jacobian* of $\boldsymbol{\lambda}(\boldsymbol{\sigma})$ with respect to $\boldsymbol{\sigma}$. As proposed in [12] we set $\boldsymbol{J} = \mathbf{0}$ with the argumentation that the *Jacobian* is reasonably small in regions around optimal shifts $\boldsymbol{\sigma}$. Consequently, (4.45) results in

$$\boldsymbol{\sigma}^{(k+1)} = -\boldsymbol{\lambda}(\boldsymbol{\sigma}^{(k)}).$$

The simplified iteration yields the BIRKA algorithm. After making an initial guess we compute the projection matrices $\boldsymbol{V}$ and $\boldsymbol{W}$ such that the $\mathcal{H}_2$-optimality conditions are satisfied. Afterwards we construct the new ROM and compute its eigenvalues. Then we take the reflected eigenvalues as new interpolation points, use the eigenvectors of the corresponding ROM to compute the new interpolation data and with that we compute new projection matrices $\boldsymbol{V}$ and $\boldsymbol{W}$.

**Algorithm 4.5 :** BIRKA

```
Data : bilinear system $\boldsymbol{\zeta}$, reduced order r
Result : $\mathcal{H}_2$-optimal $\boldsymbol{\zeta}_\mathrm{r}$
$\boldsymbol{S}_v = \boldsymbol{S}_w = \mathtt{zeros}(r)$;    ▷ initial guess for interpolation data
$\boldsymbol{U}_{v,j} = \boldsymbol{U}_{w,j} = \mathtt{ones}(r)$;
$\boldsymbol{R} = \mathtt{ones}(m, r)$;
$\boldsymbol{L} = \mathtt{ones}(p, r)$;
while not converged do
    Solve    ▷ compute projection matrices
        $\boldsymbol{E}\,\boldsymbol{V}\,\boldsymbol{S}_v - \boldsymbol{A}\,\boldsymbol{V} - \sum_{j=1}^{m} \boldsymbol{N}_j\,\boldsymbol{V}\,\boldsymbol{U}_{v,j}^\mathsf{T} = \boldsymbol{B}\,\boldsymbol{R}$,
        $\boldsymbol{E}^\mathsf{T}\,\boldsymbol{W}\,\boldsymbol{S}_w^\mathsf{T} - \boldsymbol{A}^\mathsf{T}\,\boldsymbol{W} - \sum_{j=1}^{m} \boldsymbol{N}_j^\mathsf{T}\,\boldsymbol{W}\,\boldsymbol{U}_{w,j}^\mathsf{T} = \boldsymbol{C}^\mathsf{T}\,\boldsymbol{L}$
    with proposed methods (vectorization, Algorithm 4.1, Algorithm 4.2).
    $\boldsymbol{E}_\mathrm{r} = \boldsymbol{W}^\mathsf{T}\boldsymbol{E}\boldsymbol{V}$;    ▷ applying projection → new reduced system
    $\boldsymbol{A}_\mathrm{r} = \boldsymbol{W}^\mathsf{T}\boldsymbol{A}\boldsymbol{V}$;
    $\boldsymbol{N}_{\mathrm{r},j} = \boldsymbol{W}^\mathsf{T}\boldsymbol{N}_j\boldsymbol{V}$;
    $\boldsymbol{B}_\mathrm{r} = \boldsymbol{W}^\mathsf{T}\boldsymbol{B}$;
    $\boldsymbol{C}_\mathrm{r} = \boldsymbol{C}^\mathsf{T}\boldsymbol{V}$;
    $\boldsymbol{E}_\mathrm{r}^{-1}\boldsymbol{A}_\mathrm{r} = \boldsymbol{X}_\mathrm{r}\boldsymbol{\Lambda}_\mathrm{r}\boldsymbol{X}_\mathrm{r}^{-1}$ ;    ▷ eigenvalue decomposition
    $\boldsymbol{S}_v = \boldsymbol{S}_w = -\boldsymbol{\Lambda}_\mathrm{r}$;    ▷ new interpolation data
    $\boldsymbol{U}_{w,j}^\mathsf{T} = \boldsymbol{U}_{v,j} = \boldsymbol{X}_\mathrm{r}^\mathsf{T}\boldsymbol{E}_\mathrm{r}^{-1}\boldsymbol{N}_{\mathrm{r},j}\boldsymbol{X}_\mathrm{r}^{-\mathsf{T}}$;
    $\boldsymbol{R}^\mathsf{T} = \boldsymbol{X}_\mathrm{r}^{-1}\boldsymbol{E}_\mathrm{r}^{-1}\boldsymbol{B}_\mathrm{r}$;
    $\boldsymbol{L} = \boldsymbol{C}\boldsymbol{X}_\mathrm{r}$;
```

*Remark* 4.14 (Truncated BIRKA). Note that in literature most of the times BIRKA gets presented together with the truncated bilinear rational *Krylov* algorithm (TBIRKA).

This algorithm uses methods to compute the projection matrices which do not consider the whole *Volterra* series. Consequently, we should call Algorithm 4.5 TBIRKA if we do not solve the bilinear *Sylvester* equations and in this sense do not consider the whole *Volterra* series. △

*Remark* 4.15 (Convergence Criterion For BIRKA). As for every iterative algorithm we also need a convergence criterion for BIRKA. An obvious choice would be to check if $E$ is smaller than a certain tolerance. This requires solving a *Lyapunov* equation in $\mathbb{R}^{n+r\times n+r}$, which is not possible (or at least not useful) in the large-scale setting due to lack of storage. Hence, we want to propose alternatives. One could compute the relative $\mathcal{H}_2$-error of the previous ROM and the current ROM and check if it is lower than a certain tolerance. Similar to that, one could compute the relative difference between the current interpolation points and the previous ones. △

*Remark* 4.16 (Initial Starting Point For BIRKA). Another important question to answer is the question about the initial guess since the convergence of a *Newton's* method highly depends on the starting point. Hence, we want to present some ideas proposed in [12] for the linear case. As we might expect the eigenvalues of $\boldsymbol{A}_\mathrm{r}$ close to the eigenvalues of $\boldsymbol{A}$ we could choose values in the reflected range of the eigenvalues of $\boldsymbol{A}$. Another approach would be to choose those eigenvalues of $\boldsymbol{A}$ which correspond to the largest residues. Both approaches require a modal decomposition of the FOM which is a tough task in the large-scale domain. It turned out that an initial guess which is not to big most of the times yields convergence. Hence, in the implementation, we use zero as our initial shifts. △

# Chapter 5

# Numerical Examples

Within this chapter we test the previously discussed algorithms. Therefore, we first introduce our benchmark models: the *Fokker Plank* equation and a heat transfer model. After that, we show the results of our numerical tests. In this regard, we start by comparing multipoint and multimoment *Volterra* series interpolation. Following, we compare *Volterra* series interpolation with different truncation indexes. Finally, we investigate initial points of $\mathcal{H}_2$-optimal reduction and compare BIRKA and truncated BIRKA.

## 5.1 Benchmark Models

Let us introduce our benchmark models which we will use for all our tests. Taking up the *Itô-type* linear stochastic differential equations from Chapter 2, we present the *Fokker Plank* equation. After that, we outline a standard bilinear benchmark: a heat transfer model of a steel profile.

### Fokker Plank

The *Fokker Plank* equation generally describes the chronological sequence of a probability density function under the influence of drag. Let us briefly recapitulate on [13] where it is visualized how to obtain such equation. Considering a *Brownian* particle on the real line assuming states $x \in \mathbb{R}$ which is restricted by a double-well potential

$$W(x) = (x^2 - 1)^2.$$

Supposing, we want to drag the *Brownian* particle from one well to the other well which e.g. in atomic force microscopy is done by an optical tweezer. Then, the motion of the *Brownian* particle could be described by the following stochastic differential equation

$$\mathrm{d}X_t = -\nabla V(X_t, t)\,\mathrm{d}t + \sqrt{2\sigma}\,\mathrm{d}W_t$$

where $V(x,t) = W(x) = -ux$, $0 < \sigma < \frac{1}{2}$ and $X_t$ denotes the position of the particle at time $t$. The movement could equivalently be described by the particle's probability density function

$$\rho(x,t)\,\mathrm{d}x = \boldsymbol{P}[X_t \in [x, x + \mathrm{d}x)]$$

which is defined by the *Fokker Plank* equation

$$\frac{\partial \rho}{\partial t} = \sigma \Delta \rho + \nabla \cdot (\rho \nabla V).$$

Spatial discretization of this parabolic problem directly yields a bilinear system. Following [5] we use a finite element discretization which results in a SISO bilinear system of size $n = 500$. Since it is difficult to realize the whole probability distribution, we use the probability of the particle being in one specific well as an output.

### Heat Transfer

In [4] a boundary controlled heat transfer system is introduced. The dynamics are described by the heat equation subject to *Dirichlet* and *Robin* boundary conditions which could mathematically be expressed as follows

$$\begin{aligned} x_t &= \Delta x && \text{in } (0,1) \times (0,1) \\ n \cdot \nabla x &= 0.75 \cdot u_{1,2,3}(x-1)\Delta x && \text{on } \Gamma_1, \Gamma_2, \Gamma_3, \\ x &= u_4 && \text{on } \Gamma_4, \end{aligned}$$

where $\Gamma_1, \Gamma_2, \Gamma_3, \Gamma_4$ denote the boundaries of $\Omega$. The heat transfer coefficients $u_{1,2,3,4}$ can be interpreted, e.g. as spraying-intensities of a cooling-fluid on the corresponding boundaries. The boundary conditions and a finite discretization of the *Poisson* equation using $k$ grid points yields a bilinear system of dimension $n = k^2$ with four inputs and one output. We implemented the system with $k = 40$ grid points, resulting in system dimensions $n = 1600$, to prove that the algorithms also work in the larger-scale domain.

## 5.2 Examples

In the following we present the results of our benchmarks. First, we compare multimoment and multipoint *Volterra* series interpolation in terms of interpolation quality and duration. After that, we vary the amount of considered subsystems to illustrate the subsystem's influence. Lastly, we compare BIRKA to truncated BIRKA and investigate different starting point strategies.

### 5.2.1 Multipoint And Multimoment Volterra Series Interpolation

For our first test we start by reducing both benchmark models to order $r = 20$. Therefore, we perform multimoment, mixed (multimoment and multipoint) and multipoint input *Volterra* series interpolation. For both models we use the following interpolation points $s_0$:

- multimoment: $\boldsymbol{s}_0 = 10 * \texttt{ones(1,20)}$,
- mixed:

$$\begin{aligned} \boldsymbol{s}_0 = [&1,\ 1,\ 1,\ 1,\ 1,\ 10,\ 10,\ 10,\ 10,\ 10, \\ &100,\ 100,\ 100,\ 100,\ 100,\ 1000,\ 1000,\ 1000,\ 1000,\ 1000], \end{aligned}$$

- multipoint:

$$s_0 = [1,\ 2,\ 4,\ 6,\ 8,\ 10,\ 20,\ 40,\ 60,\ 80,\\ 100,\ 200,\ 400,\ 600,\ 800,\ 1000,\ 2000,\ 4000,\ 6000,\ 8000].$$

For the *Fokker Plank* model we use the following weight matrix, only containing ones

$$\boldsymbol{U}_v = \texttt{ones(20)}.$$

Since the heat transfer model is a system with four inputs we have to use four weight matrices and additionally tangential directions which we choose also as ones

$$\boldsymbol{U}_{v,1,\dots,4} = \texttt{ones(20)},\\ \boldsymbol{R} = \texttt{ones(4, 20)}.$$

Note, that in case of higher order moments, we set the corresponding columns and rows to zero. For the *Fokker Plank* equation we choose a *Heaviside* step function ($u(t) = \sigma(t)$) as an input and for all four inputs of the heat transfer model $u_{1,\dots,4}(t) = \sin(2t)$.

In Fig. 5.1 we can observe that the multimoment and mixed *Volterra* series interpolated models yield similar results compared to the FOM. The multipoint *Volterra* series ROM could not capture the right dynamics. Let us give a brief explanation for this phenomena. The multipoint *Volterra* series framework depends far more on the right interpolation data compared to matching higher order moments which causes zero rows in the weight matrices. The importance of the right interpolation data is shown by the result of the $\mathcal{H}_2$-optimal model. It captures the right dynamics and yields the smallest relative error.

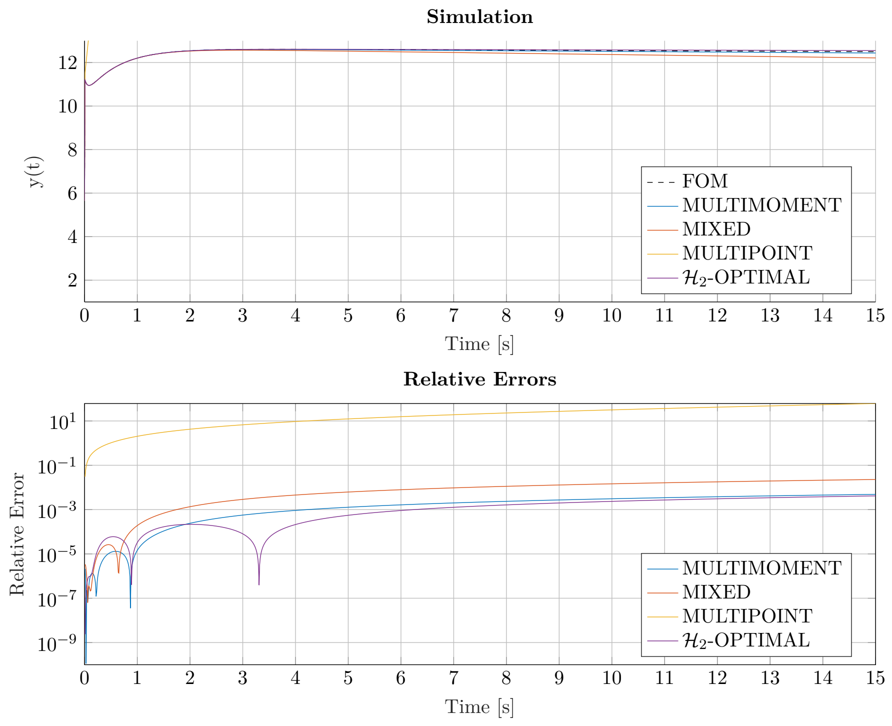


Figure 5.1: Multimoment compared to multipoint *Volterra* series interpolation of *Fokker Plank* with reduced order 20

In case of the heat transfer model, we can see in Fig. 5.2 that all frameworks yield a good approximation. All reduced systems could capture the dynamics and the mixture of multipoint and multimoment *Volterra* series interpolation performs slightly better than the other ones.

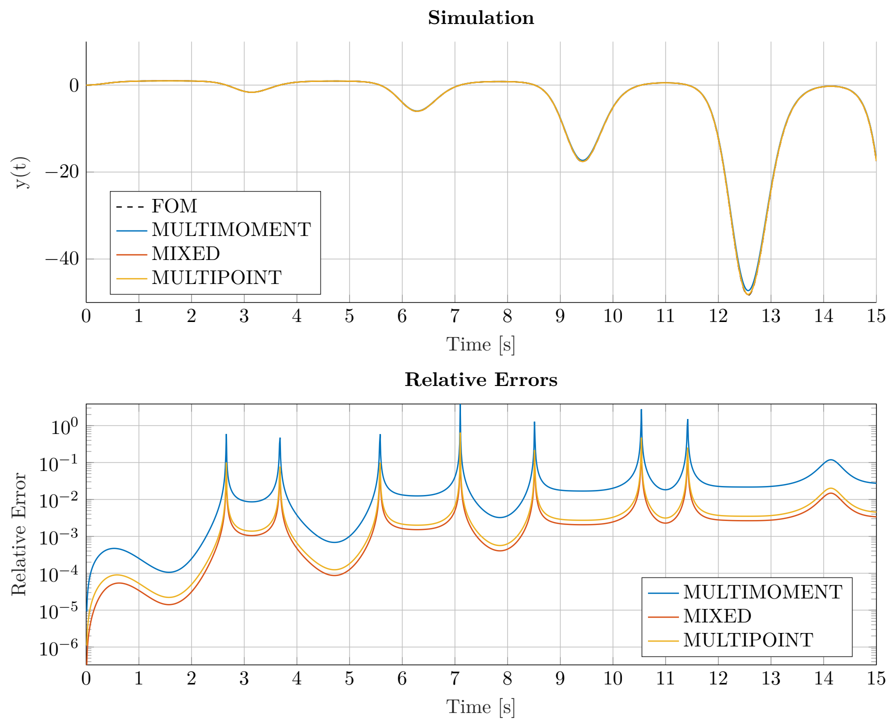


Figure 5.2: Multimoment compared to multipoint *Volterra* series interpolation of heat transfer model with reduced order 20

Upcoming, we reduce both systems to order $r = 40$. Therefore, we use the following interpolation points $s_0$ for both benchmark models:

- multimoment: $\boldsymbol{s}_0 = 10 * \texttt{ones(1,40)}$,
- mixed:

$$\boldsymbol{s}_0 = [1,\ 1,\ 1,\ 1,\ 1,\ 5,\ 5,\ 5,\ 5,\ 5,\ 10,\ 10,\ 10,\ 10,\ 10,\\ 50,\ 50,\ 50,\ 50,\ 50,\ 100,\ 100,\ 100,\ 100,\ 100,\ 500,\ 500,\ 500,\ 500,\ 500,\\ 1000,\ 1000,\ 1000,\ 1000,\ 1000,\ 10000,\ 10000,\ 10000,\ 10000,\ 10000],$$

- multipoint:

$$\boldsymbol{s}_0 = [1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9,\ 10,\ 20,\ 30,\ 40,\ 50,\ 60,\ 70,\ 80,\ 90,\ 100,\\ 200,\ 300,\ 400,\ 500,\ 600,\ 700,\ 800,\ 900,\ 1000,\ 1100,\\ 2000,\ 3000,\ 4000,\ 5000,\ 6000,\ 7000,\ 8000,\ 9000,\ 10000,\ 10100,\ 10200].$$

To reduce the *Fokker Plank* model we choose a matrix containing ones as weights

$$\boldsymbol{U}_v = \texttt{ones(40)}.$$

For the heat transfer model we use ones as weights and tangential directions. Hence, the matrices are given by

$$\begin{aligned}\boldsymbol{U}_{v,1,\ldots,4} &= \texttt{ones(40)},\\ \boldsymbol{R} &= \texttt{ones(4, 40)}.\end{aligned}$$

Again, we set the columns and rows in $\boldsymbol{R}$ and $\boldsymbol{U}_v$ corresponding to higher order moments to zero. We choose the same inputs as for the reduction to order $r = 20$.

In Fig. 5.3 we can see the results for the *Fokker Plank* equation. The multipoint *Volterra* series interpolation could not capture the right dynamics. This time, the mixture of both frameworks performs slightly better than multimoment *Volterra* series interpolation.

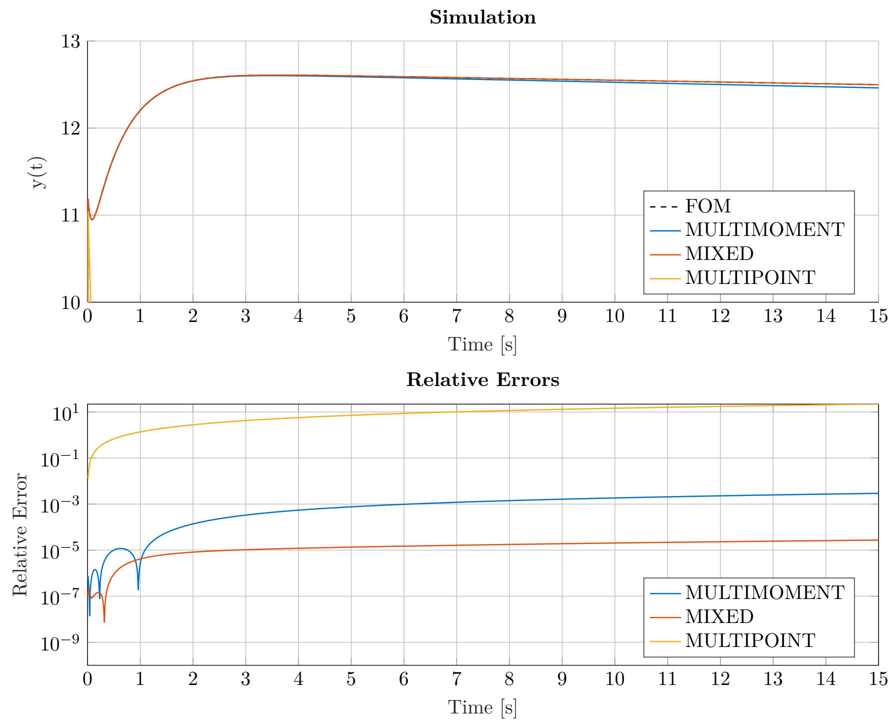
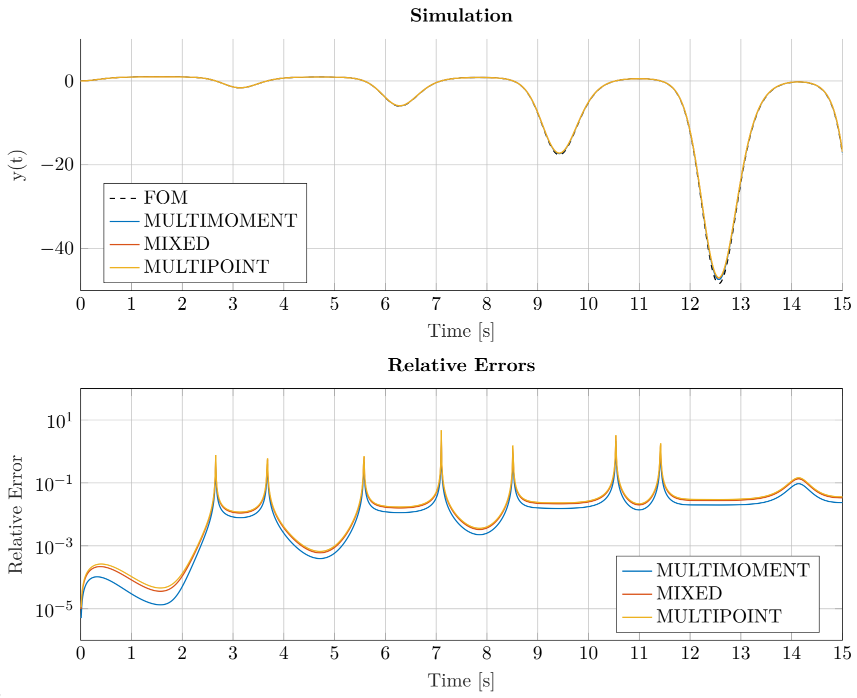


Figure 5.3: Multimoment compared to multipoint *Volterra* series interpolation of *Fokker Plank* with reduced order 40

The results for the heat transfer model are shown in Fig. 5.4. As expected, the accuracy of all frameworks increased due to the higher order of the ROM.



th t

Figure 5.4: Multimoment compared to multipoint *Volterra* series interpolation of heat transfer model with reduced order 40

### 5.2.2 Truncated Volterra Series Interpolation

In the following we present the importance of the amount of considered subsystems. Therefore, we reduce both benchmark models to order $r = 20$ using the same interpolation data as above for the mixed reduction. Consequently, we simulated the reduced system with the same inputs and computed the relative error of each reduced system. The results for the *Fokker Plank* model shown in Fig. 5.5 demonstrate that the accuracy of the reduced system increases significantly with a higher amount of considered subsystems.

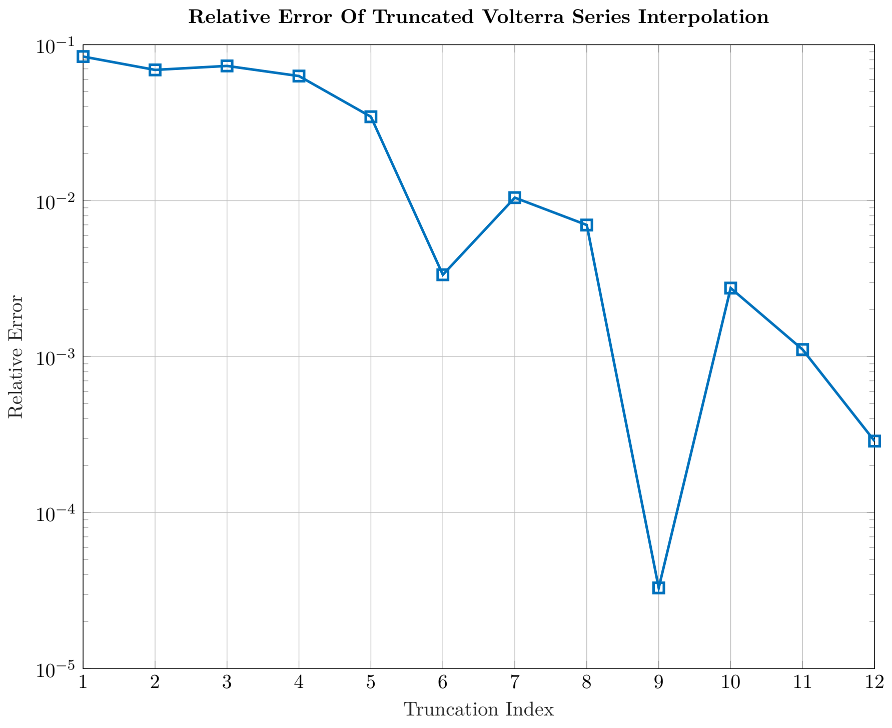


Figure 5.5: Truncated *Volterra* series interpolation of *Fokker Plank* with reduced order 20 and different amounts of considered subsystems

The results for the heat transfer system shown in Fig. 5.6 yield a different outcome. Considering more than two subsystems does not result in much better performance. This is owed to the small bilinear character (all $\boldsymbol{N}_j$ matrices have entries only on their diagonals) of this model. Concluding, it depends on the bilinear system how many subsystems are necessary to obtain the wished accuracy, but in general, the error decreases with a higher amount of considered subsystems.

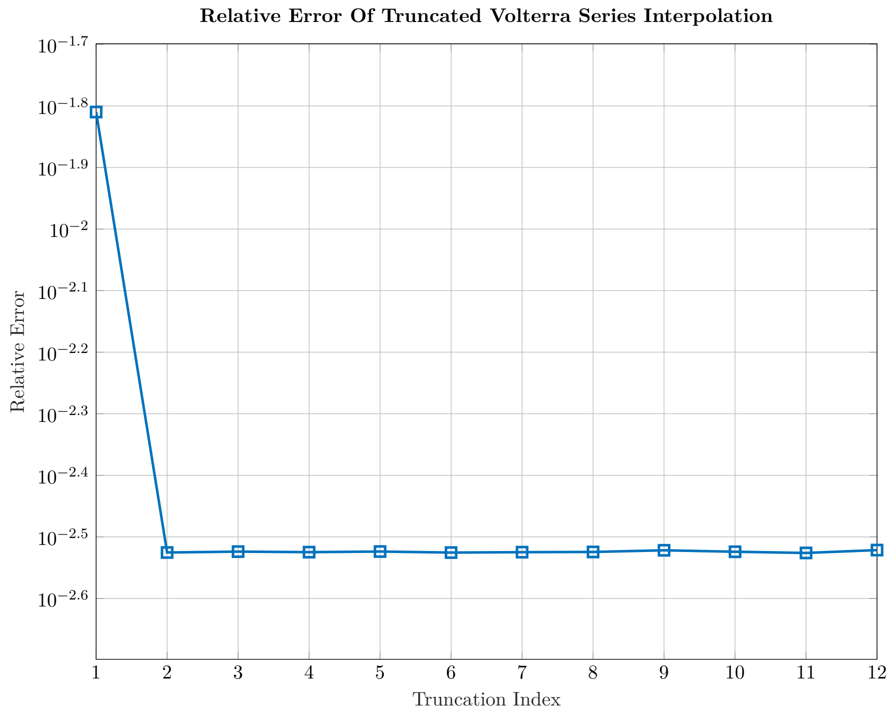


Figure 5.6: Truncated *Volterra* series interpolation of heat transfer model with reduced order 20 and different amount of considered subsystems

### 5.2.3 $\mathcal{H}_2$-Optimal Reduced Model

Finally, we want to investigate $\mathcal{H}_2$-optimal reduction. Therefore, we begin by comparing BIRKA to TBIRKA. After that, we discuss different initialization strategies.

#### BIRKA Compared To TBIRKA

We start by showing the difference between the quality of $\mathcal{H}_2$-optimal models gained with BIRKA and TBIRKA. In this sense, we reduce the *Fokker Plank* equation with BIRKA, TBIRKA considering $N = 5$ subsystems and TBIRKA considering $N = 15$ subsystems. As one can see in Fig. 5.7 BIRKA and TBIRKA considering 15 subsystems yield the same results. TBIRKA considering 5 subsystems yields even better results. Hereby, we do not want to forget that we look at relative errors. In this case the relative errors might differ a lot but the absolute errors do not.

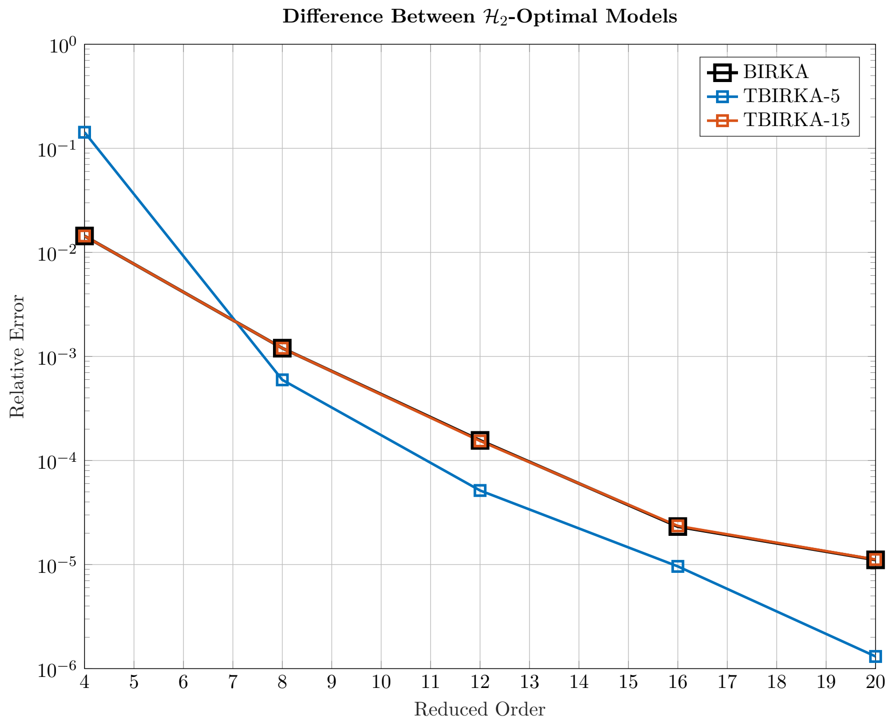


Figure 5.7: BIRKA compared to TBIRKA results for *Fokker Plank*

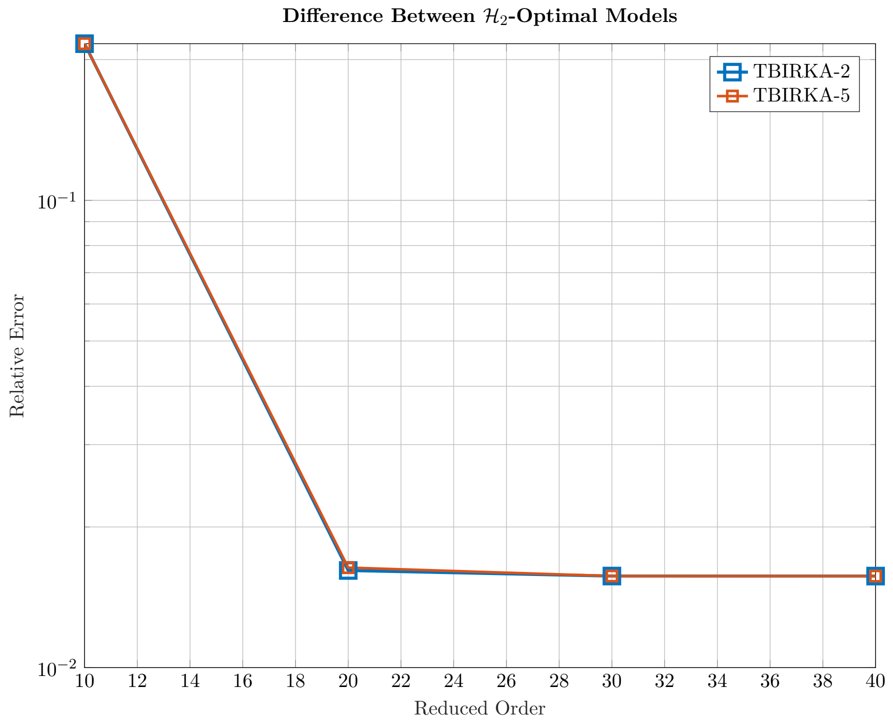


Figure 5.8: TBIRKA results compared for heat transfer model

For the heat transfer model we used BIRKA, TBIRKA considering $N = 2$ subsystems and TBIRKA considering $N = 5$ subsystems. BIRKA suffered crucially from convergence and could not yield a stable reduced model. This is due to the fact which we discussed in Remark 4.6. As we can see in Fig. 5.8, it does not make a big difference weather we consider $N = 2$ or $N = 5$ subsystems. This coincides with the results in Fig. 5.6.

**Initialization Strategies**

Finally, we want to investigate initialization strategies for BIRKA or rather TBIRKA. In this sense, we compare choosing arbitrary interpolation data, zeros as initial shifts and ones as weights or rather tangential directions and choosing interpolation data which corresponds to the FOM. We realize the last strategy by choosing the interpolation data as follows

$$\begin{aligned} [\boldsymbol{X},\ \boldsymbol{D}] &= \texttt{eigs}(\boldsymbol{A},\ \boldsymbol{E},\ r), & \boldsymbol{s}_0 &= -\texttt{diag}(\boldsymbol{D})\texttt{.'}, \\ \boldsymbol{U}_{v,j} &= \frac{1}{\texttt{norm}(\boldsymbol{E})}\texttt{norm}(\boldsymbol{N}_j)\,\texttt{ones}(r), & \boldsymbol{U}_{w,j} &= \boldsymbol{U}_{v,j}, \\ \boldsymbol{R} &= \frac{1}{\texttt{norm}(\boldsymbol{E})\,\texttt{norm}(\boldsymbol{X})}\texttt{norm}(\boldsymbol{B})\,\texttt{ones}(m,\ r), & \boldsymbol{L} &= \texttt{norm}(\boldsymbol{X})\,\texttt{norm}(\boldsymbol{C})\,\texttt{ones}(p,\ r). \end{aligned}$$

Note, that $\texttt{eigs}(\boldsymbol{A},\ \boldsymbol{E},\ r)$ computes the first $r$ eigenvalues and corresponding eigenvectors of $\boldsymbol{A}$ and $\boldsymbol{E}$. In the following we reduce both benchmark models to an order $r = 16$.

For the *Fokker Plank* model we use BIRKA to obtain the $\mathcal{H}_2$-optimal model. As we can see in Table 5.1, the results for choosing arbitrary initial interpolation data and ones and zeros are similar. Since choosing random interpolation data is non-deterministic, the outcomes vary a lot and the results should not get over interpreted. Choosing interpolation data which is related to the FOM did not yield any stable results.

Table 5.1: Different initialization strategies compared using the *Fokker Plank* system

| Strategies | Iterations | Relative Error |
|---|---|---|
| Random Interpolation Data | 52 | 1.4295e-4 |
| Zeros And Ones | 55 | 1.4294e-4 |
| Similar To FOM | no convergence | - |

For the heat transfer system we used TBIRKA considering two subsystems. Hereby, it was also possible to obtain a reduced order model by choosing random interpolation data. Setting all initial shifts to zero did not yield any convergence. Due to the properties of the system matrices and the fact that we match higher order moments in the first TBIRKA iteration the projection matrices in this step had hardly full rank at working precision. This resulted in close to singular system matrices of the first ROM. Hence, the eigenvalues have not been complex pairs anymore and the algorithm canceled the computation. Fortunately, it was possible to obtain a reduced order model with initial interpolation data, which is related to the FOM.

Table 5.2: Different initialization strategies compared using the heat transfer system

| Strategies | Iterations | Relative Error |
|---|---|---|
| Random Interpolation Data | 13 | 4.5653e-2 |
| Zeros And Ones | no convergence | - |
| Similar To FOM | 21 | 3.4475e-2 |

Summarizing, it crucially depends on system properties of the FOM which initialization yields good results. Concerning Table 5.1 and Table 5.2, one might think that it is always possible to obtain a ROM while using random initial interpolation data. As mentioned, the results for this approach varied a lot and sometimes did not result in a stable ROM, in other words BIRKA or rather TBIRKA did not converge.

# Chapter 6

# Conclusions And Outlook

## 6.1 Summary And Conclusions

The main contributions of this thesis are in-depth understanding of bilinear systems, insight in the implementation of the *Volterra* series framework and the extension of the *Volterra* series framework to MIMO systems as well as to match higher order moments.

While dealing with different representations of bilinear systems in Chapter 2 and the system theory of bilinear systems in Chapter 3 we discovered that convergence of the *Volterra* series and BIBO stability are strongly connected. In this context we found a specific constraint which the input has to fulfill to ensure convergence.
To improve the comprehensibility of the *Volterra* series framework, we illustrated the theory with many examples. With Algorithm 4.4 we proposed an implementation of the *Volterra* series framework which covers all special cases.
The multimoment *Volterra* series interpolation framework turned out to be very challenging to represent mathematically. Due to the obtained constraints for the interpolation data, a general formulation of the moment matching conditions was only possible with complex indexes. Nevertheless, this framework still is easy to implement as shown in Algorithm 4.4 since one only has to add another condition which captures higher order moments. Obviously, the multimoment framework requires less LU-decomposition which decreases computational effort. As the numerical examples have shown, a mixture of multipoint and multimoment *Volterra* series interpolation yields the the best results as one considers all globally relevant dynamics as well as local dynamics more precisely. Finally, multimoment *Volterra* series interpolation reduces the importance of the tangential directions and weights since columns and rows of the interpolation data corresponding with higher order moments are zero. This is, especially for unknown system properties, of major advantage.

## 6.2 Future Work

Within the toolbox, we implemented the proposed *Arnoldi* algorithm (Algorithm 4.4) and the vectorization of the bilinear *Sylvester* equations. As mentioned, one could customize low rank solvers for linear *Sylvester* equations to be able to solve bilinear *Sylvester* equations. Since Algorithm 4.4 tends to be unstable for badly conditioned problems, one should consider an implementation of a low rank solver which probably yields a more stable result.

As mentioned, we implemented BIRKA rather heuristically without computing the *Jacobian*. To improve convergence, one could generalize BIRKA to *Newton*-BIRKA as proposed for linear systems in [12]. Therefore, a way to compute the *Jacobian* has to be found.

We only considered the implementation of *Volterra* series interpolation and BIRKA. As mentioned there exist several other model reduction frameworks. Consequently, an implementation of subsystem interpolation and the *Gramian*-based approaches as balanced truncation are missing. Especially, for the *Gramian*-based approach one should use the existing solvers for bilinear *Lyapunov* equations as the dimensions grow crucially and the vectorization approach is not suitable due to the lack of storage.

# Appendix A

# Notation

## A.1 Special operators and symbols

### Trace of a Matrix

The trace of a matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ is defined as the sum over all diagonal entries $a_{ii}$ which yields

$$\operatorname{tr}(\boldsymbol{A}) = \sum_{i=1}^{n} a_{ii}. \tag{A.1}$$

## A.2 List of symbols

### Typographical symbols

■ end of a proof

△ end of a remark or example

▲ end of a definition

### Mathematical symbols

i imaginary unit

$\mathbf{I}_n$ identity matrix in $\mathbb{R}^{n\times n}$

$\mathbf{0}$ zero matrix

$\boldsymbol{T}$ transformation matrix

$\boldsymbol{\Lambda}$ matrix with eigenvalues on its diagonal

$\boldsymbol{X}$ matrix with right eigenvectors

$\lambda(\boldsymbol{A})$ eigenvalue of $\mathbf{A}$

$\mathcal{H}_p$ Hardy $p$-norm

$\mathcal{L}_p$ Lebesgue $p$-norm

$\times$ multiplication

## A.3 Abbreviations and acronyms

| | |
|---|---|
| **BIBO** | bounded-input bounded-output |
| **BIRKA** | bilinear rational *Krylov* algorithm |
| **DAE** | differential algebraic equations |
| **FOM** | full order model |
| **MIMO** | multiple-input multiple-output |
| **ODE** | ordinary differential equations |
| **PDE** | partial differential equations |
| **ROM** | reduced order model |
| **SISO** | single-input single-output |
| **TBIRKA** | truncated bilinear rational *Krylov* algorithm |

# List of Theorems and Other Statements

# List of Algorithms

# List of Figures

# List of Tables

# References

[1] Günter Bärwolff. *Numerik für Ingenieure, Physiker und Informatiker.* Springer Berlin Heidelberg, Berlin, Heidelberg, 2016. ISBN 978-3-662-48015-1 978-3-662-48016-8. doi: 10.1007/978-3-662-48016-8.

[2] Peter Benner and Tobias Breiten. Interpolation-Based $\mathcal{H}_2$-Model Reduction of Bilinear Control Systems. *SIAM Journal on Matrix Analysis and Applications*, 33(3), January 2012. ISSN 0895-4798, 1095-7162. doi: 10.1137/110836742.

[3] Peter Benner and Tobias Breiten. Low rank methods for a class of generalized Lyapunov equations and related issues. *Numerische Mathematik*, 124(3), July 2013. ISSN 0029-599X, 0945-3245. doi: 10.1007/s00211-013-0521-0.

[4] Peter Benner and Tobias Damm. Lyapunov Equations, Energy Functionals, and Model Order Reduction of Bilinear and Stochastic Systems. *SIAM Journal on Control and Optimization*, 49(2):686–711, January 2011. ISSN 0363-0129, 1095-7138. doi: 10.1137/09075041X.

[5] Tobias Breiten. *Interpolatory Methods for Model Reduction of Large-Scale Dynamical Systems.* PhD thesis, Otto-von-Guericke-Universität Magdeburg, May 2012.

[6] Tobias Breiten and Tobias Damm. Krylov subspace methods for model order reduction of bilinear control systems. *Systems & Control Letters*, 59(8):443–450, August 2010. ISSN 01676911. doi: 10.1016/j.sysconle.2010.06.003.

[7] C. Bruni, Gianni Di Pillo, and Giacomo Koch. On the mathematical models of bilinear systems. 2(i), January 1971.

[8] M. Cruz Varona. Bilinear systems theory and model order reduction. Unpublished internal document/monograph, Technische Universität München, 2018.

[9] M. Cruz Varona and R. Gebhart. Impulse response of bilinear systems based on Volterra series representation. March 2018. URL `https://arxiv.org/abs/1812.05360`.

[10] Garret Flagg and Serkan Gugercin. Multipoint Volterra Series Interpolation and H2 Optimal Model Reduction of Bilinear Systems. *SIAM Journal on Matrix Analysis and Applications*, 36(2), January 2015. ISSN 0895-4798, 1095-7162. doi: 10.1137/130947830.

[11] Garret M. Flagg. *Interpolation Methods for the Model Reduction of Bilinear Systems.* PhD thesis, Virginia Polytechnic Institute and State University, April 2012.

[12] S. Gugercin, A. C. Antoulas, and C. Beattie. $\mathcal{H}_2$ Model Reduction for Large-Scale Linear Dynamical Systems. June 2008.

[13] Carsten Hartmann, Anastasia Zueva, and Boris Schäfer-Bung. Balanced Model Reduction of Bilinear Systems with Applications to Positive Systems. May 2010.

[14] Alberto Isidori and Antonio Ruberti. Realization Theory of Bilinear Systems. In D. Q. Mayne and R. W. Brockett, editors, *Geometric Methods in System Theory*, pages 83–130. Springer Netherlands, Dordrecht, 1973. ISBN 978-94-010-2677-2 978-94-010-2675-8. doi: 10.1007/978-94-010-2675-8_3.

[15] Yiqin Lin, Liang Bao, and Yimin Wei. A model-order reduction method based on Krylov subspaces for MIMO bilinear dynamical systems. *Journal of Applied Mathematics and Computing*, 25(1-2):293–304, September 2007. ISSN 1598-5865, 1865-2085. doi: 10.1007/BF02832354.

[16] Ronald Mohler. Natural Bilinear Control Processes. *IEEE Transactions on Systems Science and Cybernetics*, 6(3):192–197, 1970. ISSN 0536-1567. doi: 10.1109/TSSC.1970.300341.

[17] Heiko Panzer. *Model order reduction by Krylov subspace methods with global error bounds and automatic choice of parameters*. PhD thesis, Technische Universität München, 2014.

[18] J. R. Phillips. Projection-based approaches for model reduction of weakly nonlinear, time-varying systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 22(2):171–187, February 2003. ISSN 0278-0070. doi: 10.1109/TCAD.2002.806605.

[19] Joel R. Phillips. Projection frameworks for model reduction of weakly nonlinear systems. In *Proceedings of the 37th Conference on Design Automation - DAC '00*, pages 184–189, Los Angeles, California, United States, 2000. ACM Press. ISBN 978-1-58113-187-1. doi: 10.1145/337292.337380.

[20] Wilson J. Rugh. *Nonlinear System Theory*. The Johns Hopkins University Press, 1981. ISBN O-8018-2549-0.

[21] Ta Siu and Martin Schetzen. Convergence of Volterra series representation and BIBO stability of bilinear systems. *International Journal of Systems Science*, 22(12):2679–2684, December 1991. ISSN 0020-7721, 1464-5319. doi: 10.1080/00207729108910824.

[22] Wolfgang Walter. *Ordinary Differential Equations*. Springer New York, New York, NY, 1998. ISBN 978-1-4612-0601-9. OCLC: 853271752.

[23] Thomas Wolf. *$\mathcal{H}_2$ Pseudo-Optimal Model Order Reduction*. PhD thesis, Technische Universität München, August 2014.

[24] Liqian Zhang and James Lam. On $\mathcal{H}_2$ model reduction of bilinear systems. *Automatica*, 38(2):205–216, February 2002. ISSN 00051098. doi: 10.1016/S0005-1098(01)00204-7.

[25] Walter Zulehner. *Numerische Mathematik: eine Einführung anhand von Differentialgleichungsproblemen. Bd. 2: Instationäre Probleme*. Mathematik kompakt. Birkhäuser, Basel, 2011. ISBN 978-3-7643-8428-9 978-3-7643-8429-6. OCLC: 255965613.