TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

INSTITUT FÜR MECHANIK UND MECHATRONIK
Mechanics & Mechatronics

Diplomarbeit

Interpolation of System Dynamics

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines Diplom-Ingenieurs
unter der Leitung von

Univ.Prof. Dr. Stefan Jakubek
Institut für Mechanik und Mechatronik
E325 A5

eingereicht an der Technischen Universität Wien

Fakultät für Maschinenwesen und Betriebswissenschaften

von

Elvira Thonhofer
Matr.Nr.: 0425278
Untere Donaustraße 9/2/20
1020 Wien

Wien, am 18. Juli

Name der Diplomandin

Eidesstattliche Erklärung

Ich erkläre eidesstattlich, dass ich die Arbeit selbständig angefertigt, keine anderen als die angegebenen Hilfsmittel benutzt und alle aus ungedruckten Quellen, gedruckter Literatur oder aus dem Internet im Wortlaut oder im wesentlichen Inhalt übernommenen Formulierungen und Konzepte gemäß den Richtlinien wissenschaftlicher Arbeiten zitiert, durch Fußnoten gekennzeichnet bzw. mit genauer Quellenangabe kenntlich gemacht habe.

Wien, am 18. Juli

Name der Diplomandin

Acknowledgements

I would like to thank Univ. Prof. Dr. Stefan Jakubek for providing support and guidance throughout the entire thesis work. Special thanks to DI Christian Mayr for the countless hours during which we discussed new ideas and results, for his guidance and advice during the writing-phase and the fun and friendly atmosphere he and his fellow colleagues created.

I would like to express gratitude to the numerous teachers I had the pleasure or burden to work with. Above all, my primary school teachers Andrea and Martina deserve gratitude for making my first encounter with school such a pleasure and providing the best possible platform from which to start exploring.

I am indebted to my student colleagues for all the fun we had studying. Special thanks to Farhan, Johannes, Christin, Marie and Martin for making my ten months in Göteborg a great time both on and off campus.

I would also like to thank my friends, Nancy, Zelle and David for their support during the thesis work, for asking the right questions at the right time, for fun distraction when needed, for encouragement and inspiration and for bearing with me, when I just could not stop talking about my work.

Last but not least I want to thank my entire extended family. My parents, especially my mom who is always there for me. Together with my sister, my step-father and my aunts they all provided a great environment for me to grow up. My girlfriend with her view of life in general and her knowledge of the close relation of courage and chance that, in a not-so-secrete mixture, lead to what others call luck encouraged me to reach beyond what I thought possible.

Kurzfassung

Das Problem der System-Matrix Interpolation ergibt sich aus der Schwierigkeit, nichtlineare Modelle mit Methoden der linearen Regelungstechnik zu behandeln. Es ist deshalb üblich nichtlineare Systeme in Arbeitspunkten zu linearisieren und anschliessend ein Scheduling-Problem zu lösen, anstatt die Nichtlinearität direkt zu behandeln. Ein häufig angewandter Scheduling-Ansatz besteht darin, die systembeschreibenden Matrizen linear, elementweise zu interpolieren.

Ziel dieser Arbeit war die Entwicklung einer Interpolationsmethode für systembeschreibende Matrizen. Dabei sollen dynamische und statische Systemcharakteristika linear interpoliert werden. Systeme, welche nicht als Zustandsraumsystem sondern als Übertragungsfunktion gegeben sind können ebenso behandelt werden. Übertragungsfunktionen können leicht in Regelungsnormalform umgeschrieben werden. Die Methode erreicht, dass Systemcharakteristika wie Eigenfrequenz und Dämpfung linear interpoliert werden und stellt gleichzeitig sicher, dass das errechnete System stabil ist, wenn alle ursprünglich gegebenen Systeme stabil sind.

Weil die dynamischen Systemeigenschaften in den Eigenwerten und Eigenvektoren der Systemmatrix definiert sind, basiert die Interpolationsmethode auf einer modalen Zerlegung der Systemmatrix. Um die Stabilität der gegebenen Systeme im interpolierten System zu erhalten, werden die Eigenwerte linear interpoliert. Die Interpolation der Eigenvektoren teilt sich in zwei Schritte: Die Interpolation der Länge und die Interpolation der Lage im Raum. Die Länge der Eigenvektoren wird linear interpoliert. Die Lage der Vekoren im Zustandsraum wird geometrisch interpretiert. Konjugiert komplexe Eigenvektorpaare spannen Oszillationsebenen im Zustandsraum auf, welche über geometrische Algebra (GA) interpoliert werden. Reelle Eigenvektoren werden so interpoliert, dass ihre relative Lage zur Oszillationsebene linear interpoliert wird. Eingangsvektoren werden über die Zeilen der Eingangsmatrix in den Zustandsraum gemappt und ergeben die Anregung des Systems. Die Ausgangsmatrix mappt den Zustandsvektor auf den Ausgangsvektor. Deshalb werden sowohl Input- als auch Outputmatrix linear, elementweise interpoliert.

Die entwickelte Interpolationsmethode wird an zwei Beispielen demonstriert. Die Ergebnisse werden verglichen mit Ergebnissen die sich aus Interpolationsmethode am aktuellen Stand der Technik, der Matrixinterpolation, ergeben.

Abstract

The problem of system matrix interpolation arises from non-linear plants and the difficulty of treating non-linearities with methods of linear control. The usual process is to linearize the plant at operating points and deal with a scheduling problem rather than the non-linearity itself. A very common approach to scheduling is to interpolate system matrices.

In this work a method for the interpolation of state space systems is introduced, which is targeted to interpolate the system characteristics linearly. Systems which are denoted as transfer functions can also be interpolated using the proposed method. Transfer functions can be easily transformed to state-space notation, e.g. controllability canonical form. On the one hand the *system characteristics*, such as damping ratio and eigenfrequency, are linearly interpolated and on the other hand the stability of the resulting system is ensured if the original systems are stable.

Since the system characteristics are encoded in the eigenvalues and eigenvectors the introduced method is based on eigenvalue decomposition of the system matrix. To ensure stability of the resulting system the eigenvalues are linearly interpolated. The interpolation of the eigenvectors is split into two parts: their length and orientation. The length is interpolated linearly, the orientation of the eigenvectors is geometrically interpreted. Conjugate complex pairs define oscillation planes in the state-space which are interpolated using Geometric Algebra (GA). Real valued eigenvectors are interpolated, so that their relative attitude to the vectors that form the oscillation plane is interpolated linearly. The input vector is mapped onto the state space via the rows of the input matrix, yielding the excitation of the respective states. The output matrix maps the state space dimensions onto the output (vector). Consequently both are linearly interpolated.

The introduced method is tested on demonstrative examples. The results are compared with results generated by the state-of-the-art matrix coefficient interpolation.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

The interpolation of linear systems is a common task in control engineering. It arises from non-linear plants and the difficulty of treating non-linearities with methods of linear control. The usual process is to linearize the plant at operating points and deal with a scheduling (i.e. interpolation) problem rather than the non-linearity itself. Independent of the system architectures the interpolation of system characteristics is vital.
The problem has been addressed via gain scheduling [2], [3], for fuzzy systems [4] and local model networks [5]. A very common approach to scheduling is to linearly interpolate system matrices or coefficients of transfer functions. Unfortunately this approach may yield unstable interpolated systems, even if all contributing original systems are stable [6]. Additionally, the system characteristics are interpolated in an undesirable way. System characteristics such as natural frequency, oscillation planes, damping ratio and speed of particular modes are encoded in the eigenvalues and eigenvectors of the system matrix. Interpolating coefficients of the system matrix linearly implies, that the eigenvalues and eigenvectors of the matrix shift in an unpredictable way.
An algorithm is required, that interpolates system characteristics linearly and guarantees stable interpolated systems as long as all contributing original systems are stable.

1.2 Problem Definition

The core task of this work is to develop an interpolation algorithm for dynamic systems. The non-linear system is represented by a set of linear local models. Among the first non-linear problems of interest were missile flight controllers with only two controlling variables [7]. A scheduling law by which the set of linearized models was summarized needed to be found.

The desired specifications of such a scheduling law and the controller based on it were

- a continuous scheduled transfer function in the entire domain
- a controller that stabilizes the plant at every point in the domain

The concept of interpolation has since then been expanded. Due to computational power more complicated models can be treated. This in turn makes scheduling complicated, since the method requires a significant amount of intuition when choosing scheduling variables. Hence, a flexible, automated method for system interpolation is sought. Additionally, the interpolation should capture the system's dynamic characteristics as well.

1.2.1 Requirements for System Interpolation

The interpolation process should ideally (linearly) interpolate *all* of the following characteristics simultaneously:

- **Natural Frequency**
- **Damping Ratio**
- **Attitude of the oscillation planes** in the state space
- **Orientation of the oscillation planes** in the state space
- **Internal dynamics** of non-flat systems
- **Steady State Values** of the state vector

and retain stability of the interpolated system if the adjoining systems are stable, too. The introduced method meets all the above requirements, albeit linearity of interpolation of *all* factors simultaneously is generally not given. Also, since only differentially flat systems are investigated the requirement of correctly interpolating internal dynamics is not tested. Differential flatness is introduced in section 1.5. For SISO Systems all requirements are met simultaneously.
MISO Systems cause problems. A trade-off between dynamic and static (the steady state values) requirements exists. In section 3.8 the nature of this trade-off is explained.
However, it remains to be clarified whether linear interpolation is justified from a practical (or physical) point of view.

1.3 System Architecture

The system architecture defined here represents a general structure that does not limit the method's range of application. Transfer functions can be transformed to state space notation, more precisely, they can be transformed to controllability canonical form [8]. In analogy with [9] an ordered set for the indices of the local models is defined:

$$\mathcal{I} = (i \in \mathbb{N} | 1 \leq i \leq I) \tag{1.1}$$

where I denotes the number of local linear models. The state space notation of local model networks investigated here is defined as a weighted sum of linear, time-invartiant, local system descriptions defined by

$$\begin{aligned} d\boldsymbol{x}(t) &= \boldsymbol{A}_{int}\boldsymbol{x}(t) + \boldsymbol{B}_{int}\boldsymbol{u}(t) \\ y(t) &= \boldsymbol{C}_{int}\boldsymbol{x}(t), \end{aligned} \tag{1.2}$$

where

$$d\boldsymbol{x}(t) = \begin{cases} \dot{\boldsymbol{x}}(t), & \text{for continuous time systems} \\ \boldsymbol{x}(t+1), & \text{for discrete time systems} \end{cases}$$

The interpolation can be generally defined as function f of the local matrices $\boldsymbol{A}_i,\ \boldsymbol{B}_i,\ \boldsymbol{C}_i$ and the corresponding local weight Φ_i:

$$\begin{aligned} \boldsymbol{A}_{int} &= \boldsymbol{f}(\boldsymbol{A}_{\mathcal{I}}, \Phi_{\mathcal{I}}),\ \boldsymbol{A}_{int} \in \mathbb{R}^{n \times n} \\ \boldsymbol{B}_{int} &= \boldsymbol{f}(\boldsymbol{B}_{\mathcal{I}}, \Phi_{\mathcal{I}}),\ \boldsymbol{B}_{int} \in \mathbb{R}^{n \times q} \\ \boldsymbol{C}_{int} &= \boldsymbol{f}(\boldsymbol{C}_{\mathcal{I}}, \Phi_{\mathcal{I}}),\ \boldsymbol{C}_{int} \in \mathbb{R}^{m \times n} \end{aligned} \tag{1.3}$$

where n denotes the system dimension, q denotes the number of inputs and m denotes the number of outputs.
The local weights are constrained:

$$\sum_{\mathcal{I}} \Phi_i = 1 \tag{1.4}$$

$$0 \leq \Phi_i \leq 1,\ \forall i \in \mathcal{I} \tag{1.5}$$

1.4 Introduction to matrix interpolation

Matrix interpolation [10] is widely used and serves as the benchmark method in this work. It is defined by

$$\boldsymbol{A}_{int} = \sum_{\mathcal{I}} \Phi_i \boldsymbol{A}_i, \quad \boldsymbol{B}_{int} = \sum_{\mathcal{I}} \Phi_i \boldsymbol{B}_i, \quad \text{and} \quad \boldsymbol{C}_{int} = \sum_{\mathcal{I}} \Phi_i \boldsymbol{C}_i$$

so that

$$\begin{aligned} d\boldsymbol{x}(t) &= \sum_{\mathcal{I}} \Phi_i (\boldsymbol{A}_i \boldsymbol{x}(t) + \boldsymbol{B}_i \boldsymbol{u}(t)) \\ y(t) &= \sum_{\mathcal{I}} \Phi_i \boldsymbol{C}_i \boldsymbol{x}(t), \end{aligned} \tag{1.6}$$

where Φ_i are the interpolation weights.

The immediate advantage of matrix interpolation is a computationally cheap method. However, the main drawback of this ad hoc solution is, that for systems of order $n \geq 3$ the resulting system may become unstable, even if all original systems are stable. According to the Schur-Cohn-Jury criteria the stability regions are not convex for third and higher order systems [6]. This means that linear interpolation of matrix coefficients shifts the eigenvalues of the matrices in a non-linear way. This can cause intermediate pole configurations to be unstable, while the original pole configurations are stable.

Example 1.1: Unstable Intermediate Pole Configuration

Two systems of dimension $n = 6$ are defined by their transfer functions

$$G_{o,1} = \frac{1}{s^6 - 1.1s^5 - 0.8s^4 + 0.89s^3 + 0.6609s^2 - 0.4752s - 0.1135} \tag{1.7}$$

and

$$G_{o,2} = \frac{1}{s^6 + 0.2s^5 + 0.95s^4 + 0.22s^3 + 0.3679s^2 + 0.1066s - 0.07994} \tag{1.8}$$

The poles of the two systems are located at

$$\boldsymbol{P}_1 = \begin{bmatrix} 0.9 \\ 0.9 + 0.4i \\ 0.9 - 0.4i \\ -0.2 \\ -0.7 + 0.4i \\ -0.7 - 0.4i \end{bmatrix}, \quad \boldsymbol{P}_2 = \begin{bmatrix} 0.3 \\ 0.3 + 0.8i \\ 0.3 - 0.8i \\ -0.5 \\ -0.3 + 0.8i \\ -0.3 - 0.8i \end{bmatrix}$$

Matrix interpolation yields intermediate interpolated systems. Fig. 1.1 shows the pole configuration of the two stable systems, with interpolated systems being unstable. Note, that despite the interpolation step = 0.1 the poles are unequally spaced throughout the interpolation range of $\Phi_1 \in [0, 0.1, \ldots, 1]$ (and accordingly $\Phi_2 \in [1, 0.9, \ldots, 0]$).

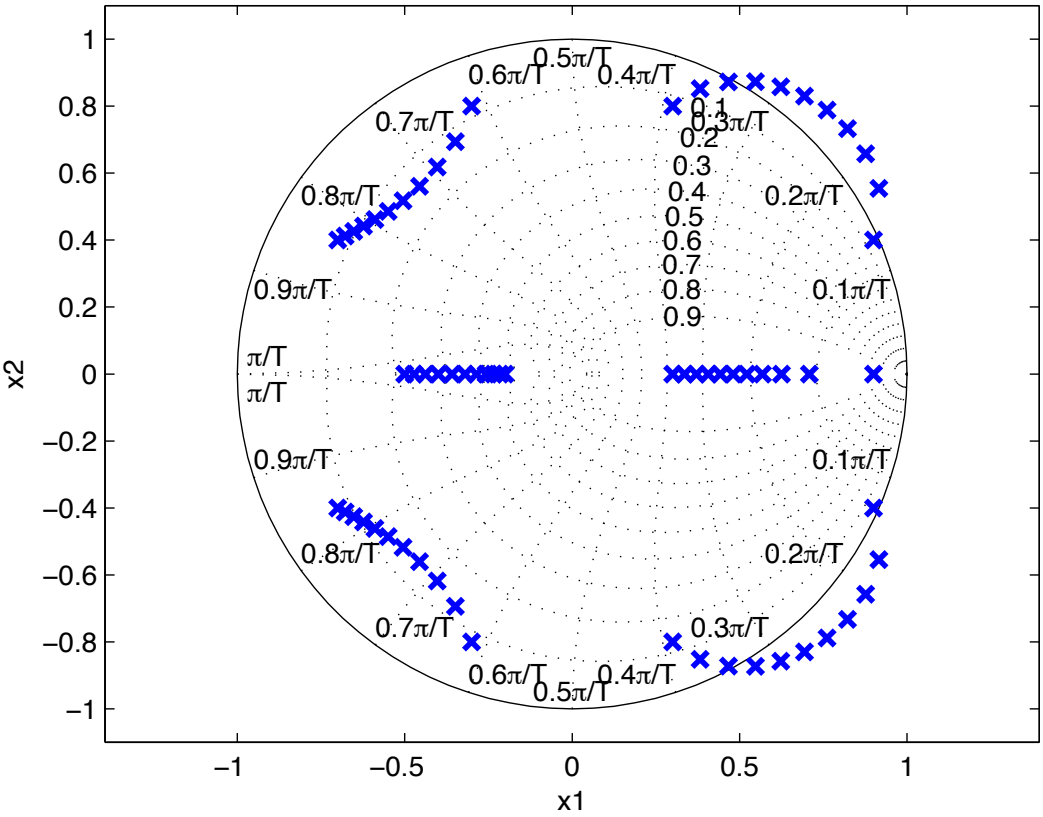


Figure 1.1: Matrix Interpolation applied to two stable discrete-time systems of dimension $n = 6$. Interpolated systems become unstable, poles (blue crosses) are located outside the unit circle.

This risk is minimized by only interpolating relatively "close" system matrices. That means, that the operating points by which the sets of matrices are defined must be fairly close so as to ensure that the coefficients of the matrices differ only a little. Hence, the eigenvalues of the system matrices at different operating points are close and the non-linear shift is somewhat under control, since the possible interval of travel is small.
Matrix interpolation is applied when eigenvalues of the original matrices do not differ much. If this pre-condition is satisfied matrix interpolation delivers plausible results at very low computational cost.

1.5 Differential Flatness and Internal Dynamics

A system is differentially flat if a set of outputs (equal in number to the number of inputs) can be found, such that all states and inputs can be determined from these outputs without integration. Precisely, a system with the states $\boldsymbol{x} \in \mathbb{R}^n$ and inputs $\boldsymbol{u} \in \mathbb{R}^m$ is flat if a set of outputs $\boldsymbol{y} \in \mathbb{R}^m$ of the form

$$\boldsymbol{y} = \boldsymbol{y}(\boldsymbol{x}, \boldsymbol{u}, \dot{\boldsymbol{u}}, \ldots, \boldsymbol{u}^{(p)}) \tag{1.9}$$

can be found, such that

$$\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{y}, \dot{\boldsymbol{y}}, \ldots, \boldsymbol{y}^{(p)}) \tag{1.10}$$

and

$$\boldsymbol{u} = \boldsymbol{u}(\boldsymbol{y}, \dot{\boldsymbol{y}}, \ldots, \boldsymbol{y}^{(p)}) \tag{1.11}$$

are fulfilled [11]. The integer $p \in \mathbb{N}^+$ indicates a finite number of derivatives with respect to time. Hence, it is possible to plan trajectories of plants such as unmanned vehicles [12] and helicopters [13].

Systems with internal dynamics are non-flat. A system that exhibits zero-dynamics, which is a special case of the more general formulation of internal dynamics, shows a zero-output in the open loop despite being excited by a non-zero input signal. Such a system is described in example 1.2. This is due to cancellation of one or more input terms with terms of the transfer function of the plant. Since internal dynamics are not observable it is vital to determine whether a system can exhibit internal dynamics at all. If this is the case, the system is non-flat.

In this work only flat systems are treated. However, the introduced interpolation scheme is applicable to general systems. Knowing the potential internal dynamics of a system enables the application of the interpolation method to this part of the system dynamics as well.

Example 1.2: Zero Dynamics

> A plant with a continuous-time transfer function $G(s) = \frac{s^2+1}{(s+1)(s+2)(s+5)}$ is excited by a sine wave $U(s) = \frac{1}{s^2+1}$. Then the output $Y(s) = \frac{1}{(s+1)(s+2)(s+5)}$ and the steady state output is zero. Fig. 1.2 shows the output (blue) and the input (red) of the excited system.

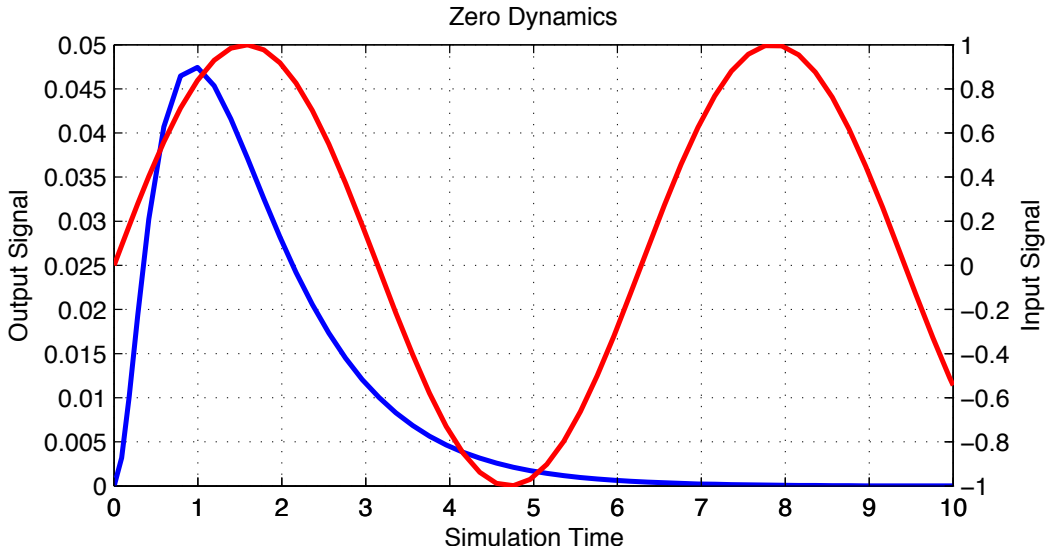


Figure 1.2: The output of a continuously excited system that exhibits zero-dynamics.

1.6 State Space Systems and Geometric Algebra

The state space representation of systems is a compact description that is based on coefficient matrices. One major advantage is that the extraction of eigenvectors and eigenvalues of the system matrix is easily accomplished and allows a straight forward geometrical interpretation of the system characteristics. Complex eigenvectors represent the system's oscillation planes, eigenvalues hint on how "fast" the system reacts to and excitation in the direction of the corresponding eigenvector.

To capture every element of a system response to an arbitrary signal the entire set of describing matrices must be considered. The approach presented here focuses on the geometric interpretation of system characteristics and the concluding "geometric" interpolation of the system matrix.

Geometric algebra [1], [14] provides an alternative to conventional 3-dimensional vector algebra and generally, to vector calculus. GA is a coordinate independent geometry based on Clifford Algebra [15]. In contrast to vector calculus it is based on subspaces. A subspace contains more information than just the object dimension by which it is spanned. Further computation with that subspace automatically considers relative orientation for example.

Geometric Algebra is successfully applied in mathematical physics [16], computer graphics [17] and engineering problems, as e.g. with neural networks [5].

Chapter 2

Geometric Algebra

2.1 Basics

Geometric Algebra (GA) is defined in spaces $\mathbb{R}^n$, with $n \in \mathbb{N}^+$. The space $\mathbb{R}^n$ contains all objects of dimension $m \leq n$, each representing a subspace of dimension m. Hence, in a 3-dimensional space $\mathbb{R}^3$ the maximum dimension of a subspace is 3.
Each space is fully defined by a set of unit elements. The attribute *unit* defines that the objects magnitude is = 1. These standard elements are of dimension $\leq n$.
For general spaces $\mathbb{R}^n$ with a basis of unit vectors $[\boldsymbol{e_1}, \boldsymbol{e_2}, \ldots \ , \boldsymbol{e_n}]$ the basic elements are

$$
\begin{aligned}
[\underbrace{1}_{\text{0-blade}}, &\underbrace{\boldsymbol{e_1}, \boldsymbol{e_2}, \ldots \boldsymbol{e_n}}_{\text{1-blades}}, \underbrace{\boldsymbol{e_1} \wedge \boldsymbol{e_2}, \boldsymbol{e_2} \wedge \boldsymbol{e_3}, \boldsymbol{e_3} \wedge \boldsymbol{e_1}}_{\text{2-blades}}, \\
&\underbrace{\boldsymbol{e_1} \wedge \boldsymbol{e_2} \wedge \boldsymbol{e_3}, \boldsymbol{e_2} \wedge \boldsymbol{e_3} \wedge \boldsymbol{e_4}, \ldots, \boldsymbol{e_{n-1}} \wedge \boldsymbol{e_n} \wedge \boldsymbol{e_1}}_{\text{3-blades}}, \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdots \\
&\ldots, \underbrace{\boldsymbol{e_1} \wedge \boldsymbol{e_2} \wedge \ldots \wedge \boldsymbol{e_n}}_{\text{n-blade or pseudoscalar}}]
\end{aligned}
$$

In $\mathbb{R}^3$ these elements are reduced to

$$
[\underbrace{1}_{\text{scalar}}, \underbrace{\boldsymbol{e_1}, \boldsymbol{e_2}, \boldsymbol{e_3}}_{\text{1-blades or vectors}}, \underbrace{\boldsymbol{e_1} \wedge \boldsymbol{e_2}, \boldsymbol{e_2} \wedge \boldsymbol{e_3}, \boldsymbol{e_3} \wedge \boldsymbol{e_1}}_{\text{2-blades or bivectors}}, \underbrace{\boldsymbol{e_1} \wedge \boldsymbol{e_2} \wedge \boldsymbol{e_3}}_{\text{3-blade or trivector}}] \tag{2.1}
$$

where the $\wedge$ operator denotes the outer product, see section 2.3.1. The trivector in $\mathbb{R}^3$ represents the basic, oriented volume element, also denoted by $\boldsymbol{I}_3$.
Arbitrary objects in $\mathbb{R}^n$ can be represented as a linear combination of scalar-weighted standard elements. Note that an object may consist of elements of various dimensions. Every object of a single (pure) dimension can be represented with conventional coordinates and

vice versa. This is of great use wherever part of the computation is based on vector calculus or in case the result of a geometric algebra computation is required for further processing in conventional vector calculus. In the following, objects of pure dimension will be called *blades*. An object of dimension n is an n-blade.
For illustration all figures presented in this chapter are generated with GABLE - a Geometric Algebra toolbox for MATLAB ®[1]. The entire chapter on Geometric Algebra is based on the GABLE Toolbox tutorial [1] and on [14]. Both cover basic and more advanced mathematical aspects. While the GABLE tutorial focusses, naturally, on GABLE application, [14] presents a more general approach and provides application examples and ideas in the field of computer graphics.
The GABLE toolbox is developed for $\mathbb{R}^3$, a 3-dimensional euclidian space. It uses an orthogonal basis [$\boldsymbol{e_1}$, $\boldsymbol{e_2}$, $\boldsymbol{e_3}$] similar to a conventional coordinate system with 3 orthogonal axes. The immediate advantage of a 3D space is that all objects can easily be displayed and operation results can be verified. The entire set of unit elements that GABLE is based on is introduced in section 2.2.
In this section the following notation will be used: Scalars will be assigned lower case greek letters (α, β, ...), 1-blades will be assigned bold lower case roman letters ($\mathbf{u}$, $\mathbf{v}$, ...) and objects of higher grade will be assigned bold, capital roman letters ($\mathbf{A}$, $\mathbf{B}$, ...). The lower case $\mathbf{i}$ denotes a unit bivector. The upper case $\boldsymbol{I_3}$ denotes a unit trivector.

2.2 Objects

2.2.1 Scalars, 0-dimensional Objects

A scalar is a 0-dimensional blade. It is best understood as a weighted point in the origin. Its attribute is

- magnitude: such as the mass of a point-mass

2.2.2 Vectors, 1-dimensional Objects

A vector is a 1-dimensional direction element, a 1-blade. Its attributes are

- magnitude ${}_m\mu$: a measure of length, similar to a vector's 2-norm
- attitude ${}_a\mu$: equivalent to a vector definition, the sum of scalar weighted standard elements

A vector in common geometry is mathematically equivalent to a 1-blade in GA. Arbitrary 1-blades in $\mathbb{R}^n$ can be considered a sum of scalar-weighted standard 1-blades

$$\boldsymbol{u} = \sum_n \alpha_i \boldsymbol{e_i} \tag{2.2}$$

where n denotes the dimension of the space.
In $\mathbb{R}^3$ the general sum is reduced to a maximum of three elements

$$\boldsymbol{u} = \alpha_1 \boldsymbol{e_1} + \alpha_2 \boldsymbol{e_2} + \alpha_3 \boldsymbol{e_3}$$

The relation between a vectors magnitude and attitude in space is defined by

$${}_m\mu = \sqrt{\alpha_1^2 + \alpha_2^2 + \alpha_3^2} \quad \text{and}$$

$${}_a\mu = \frac{\alpha_1}{{}_m\mu}\boldsymbol{e_1} + \frac{\alpha_2}{{}_m\mu}\boldsymbol{e_2} + \frac{\alpha_3}{{}_m\mu}\boldsymbol{e_3}$$

so that

$$\boldsymbol{u} = {}_m\mu \cdot {}_a\mu \tag{2.3}$$

To normalize an element's magnitude to 1 it is devided by its norm, similar to a vector being normalized to unit length by dividing it by its norm.
Fig. 2.1 depicts the two vectors $\boldsymbol{u} = \boldsymbol{e_1} + \boldsymbol{e_2}$ and $\boldsymbol{v} = \boldsymbol{e_3}$ (blue) as well as the standard 1-blades $\boldsymbol{e_1}, \boldsymbol{e_2}$ and $\boldsymbol{e_3}$ (black).

2.2.3 Bivectors, 2-dimensional Objects

A bivector is a directed area element, a 2-blade. It evolves from the outer product denoted by the $\wedge$ operator, which is defined in section 2.3.1, performed on two 1-blades. Its attributes are

- magnitude ${}_m\nu$: scalar value of its area, as if spanned by two conventional vectors
- attitude ${}_a\nu$: equivalent to a plane definition (for example by a parameter equation requiring a fixed point and two vectors starting at that point)
- ***orientation*** ${}_o\nu$: sense of rotation from first defining 1-blade to the second

The definition of a bivector's magnitude it related to the cross-product in $\mathbb{R}^3$. The cross (or vector)-product of **a** and **b** represents a vector perpendicular to **a** and **b** with magnitude equal to the area of the parallelogram spanned by these two vectors. The magnitude of a bivecor is just the scalar value of its virtual spanned area. The *shape* of the area-element

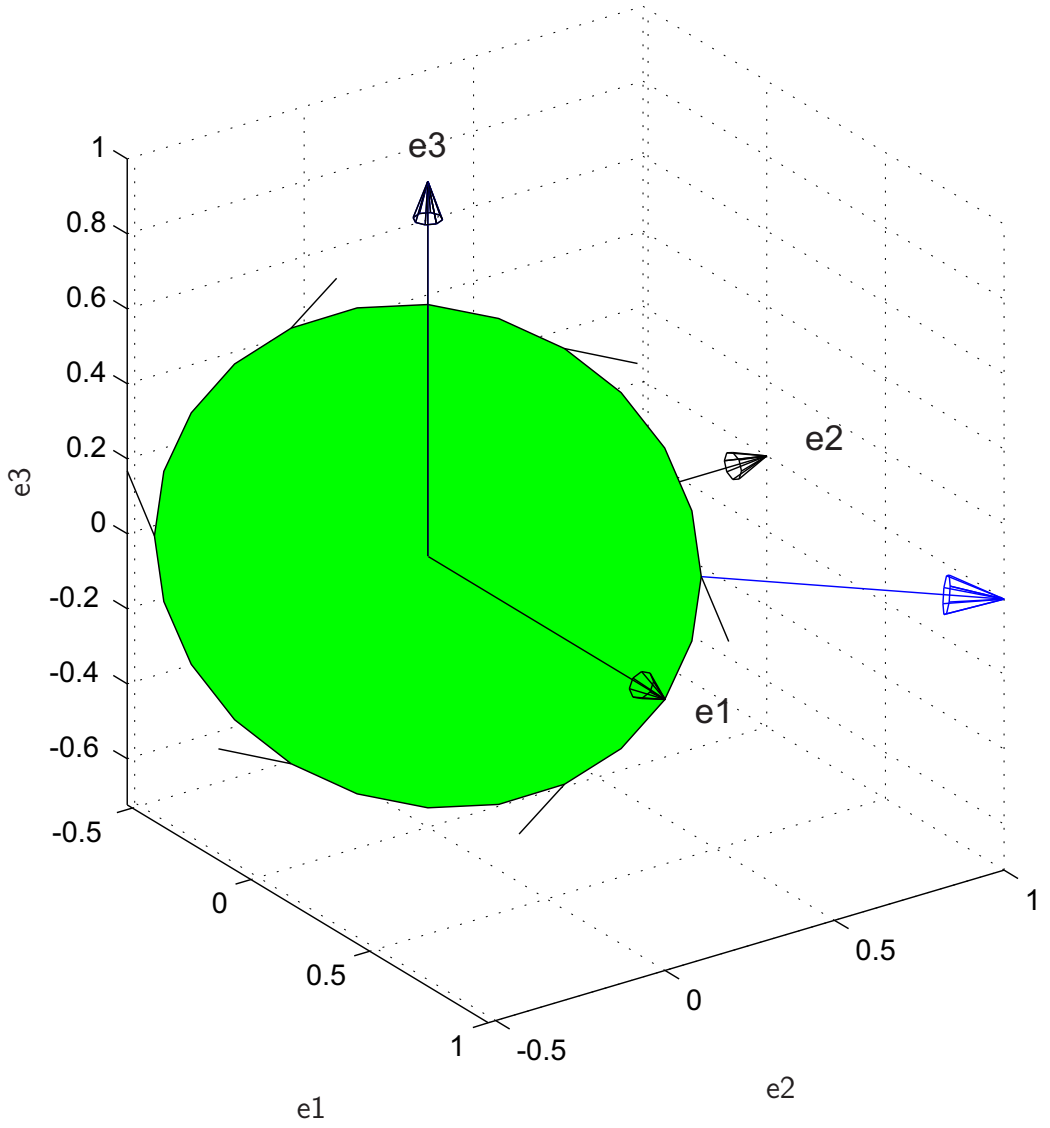


Figure 2.1: A bivector spanned by the two vectors $\boldsymbol{u} = \boldsymbol{e_1} + \boldsymbol{e_2}$ and $\boldsymbol{v} = \boldsymbol{e_3}$.

is not defined and irrelevant. In this work bivectors are displayed as colored circles. To underline that two 1-blades define the bivector a parallelogram can be used, as in Fig.2.7. The attitude in space is similar to the information of plane definition in linear algebra. The important characteristic is its orientation - the 2-blade *contains* this information. Operations on bivectors automatically capture the entire set of attributes. This is especially beneficial in high dimension, where vector calculus becomes complicated, if all attributes need to be captured.

Any 2-blade can be expressed as a linear combination of scalar-weighted *standard* bivectors. A standard bivector is a bivector generated from 2 vectors of the (orthogonal) basis of $\mathbb{R}^n$:

$$\boldsymbol{e_i} \wedge \boldsymbol{e_j} \text{ with } i, j \in [1, \dots n] \text{ and } i \neq j$$

These bivectors form the (orthogonal) bivector basis given in (2.1).
An arbitrary bivector is defined as

$$\boldsymbol{u} \wedge \boldsymbol{v} = \underbrace{|\boldsymbol{u}||\boldsymbol{v}| \sin(\theta)}_{({}_o\nu)\,{}_m\nu} \boldsymbol{i} \tag{2.4}$$

with

$$\mathbf{i} = \sum_n \frac{\alpha_i}{{}_m\nu_i} (\boldsymbol{e_i} \wedge \boldsymbol{e_j}) \text{ and } i \neq j \tag{2.5}$$

being the unit element of the plane spanned by $\boldsymbol{u}$ and $\boldsymbol{v}$. θ is the angle between the two 1-blades. Hence, each bivector can be rewritten as the sum of scalar weighted basic bivectors given in (2.1).
A short example will clarify (2.4).

Example 2.1: Characteristics of a Bivector

Consider two 1-blades $\boldsymbol{u} = \boldsymbol{e_1} + \boldsymbol{e_2}$ and $\boldsymbol{v} = \boldsymbol{e_3}$ as depicted in Fig. 2.1. Then $|\boldsymbol{u}| = \sqrt{2}$ and $|\boldsymbol{v}| = 1$. The angle between the two 1-blades $\theta = \pi/2$. Hence, (2.4) reads

$$\boldsymbol{u} \wedge \boldsymbol{v} = +\sqrt{2} \cdot 1 \cdot 1 \cdot (\frac{1}{\sqrt{2}} \boldsymbol{e_1} \wedge \boldsymbol{e_3} + \frac{1}{\sqrt{2}} \boldsymbol{e_2} \wedge \boldsymbol{e_3}) \tag{2.6}$$

$$\boldsymbol{u} \wedge \boldsymbol{v} = 1 \cdot \boldsymbol{e_1} \wedge \boldsymbol{e_3} + 1 \cdot \boldsymbol{e_2} \wedge \boldsymbol{e_3} \tag{2.7}$$

$$\boldsymbol{u} \wedge \boldsymbol{v} := \begin{cases} {}_o\nu & = + \\ {}_m\nu & = \sqrt{2} \\ {}_a\nu & = \boldsymbol{e_1 e_3} + \boldsymbol{e_2 e_3} \end{cases} \tag{2.8}$$

The magnitude can not be read directly from (2.7) but must be computed with (2.4). The orientation and attitude in space can be read from the terms and their respective signs in (2.7).

The orientation ${}_o\nu$ and the attitude ${}_a\nu$ of a bivector are coupled. The bivector $\boldsymbol{e_1} \wedge \boldsymbol{e_3}$ is oriented *from* $\boldsymbol{e_1}$ *to* $\boldsymbol{e_3}$ and is identical to the bivector $-\boldsymbol{e_3} \wedge \boldsymbol{e_1}$, which is oriented *against* the direction from $\boldsymbol{e_3}$ to $\boldsymbol{e_1}$.

$$\boldsymbol{e}_1 \wedge \boldsymbol{e}_3 = -\boldsymbol{e}_3 \wedge \boldsymbol{e}_1 \tag{2.9}$$

The bivector's characteristics call for the application of GA, when oriented plane elements need to be interpolated. Oscillation planes are an example of oriented plane elements. The oscillation plane is completely described by the bivector, see Fig. 2.1, as the bivector contains both magnitude and attitude and additionally the orientation. Hence, the interpolation specification stated in section 1.2.1 concerning the oscillation plane can be met. The physical meaning of the oscillation plane's attributes can be explained as follows: The state vector's

trajectory spirals in the oscillation plane as indicated by the orientation of the bivector. Hence it is vital to correctly interpolate this particular characteristic of the oscillation plane. Fig. 2.2 shows the exemplary trajectory of a state vector and the corresponding oscillation plane.

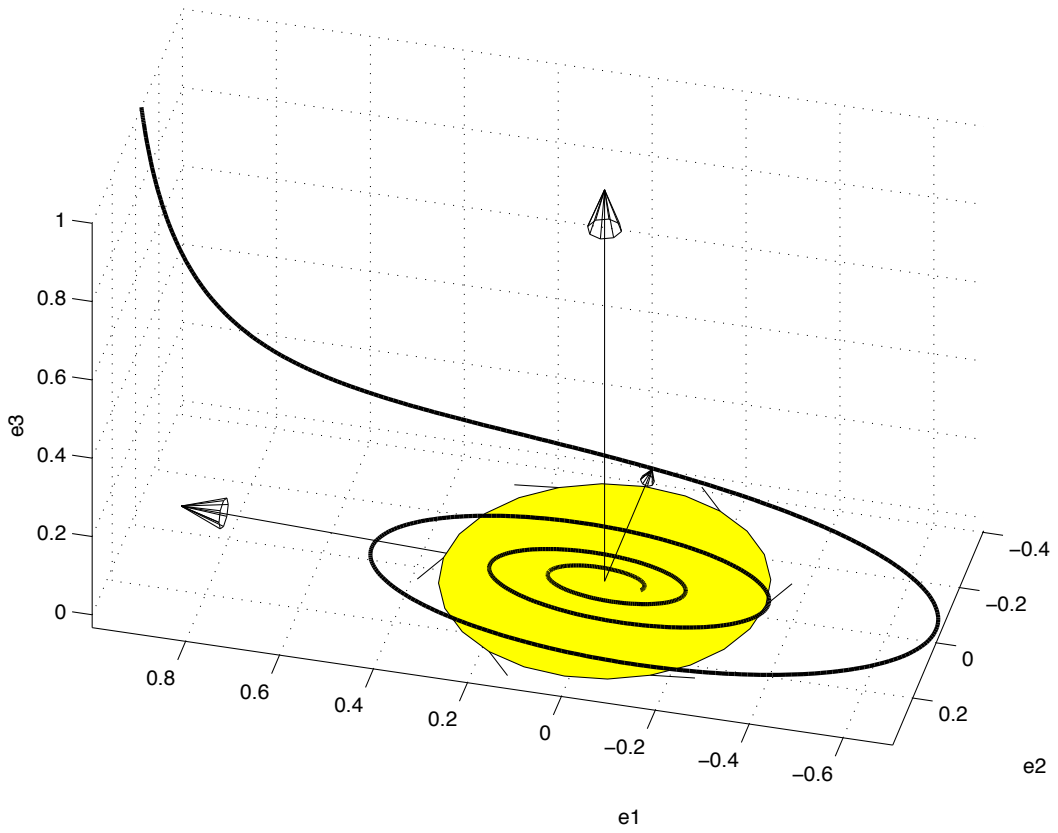


Figure 2.2: A state vector's trajectory, the eigenvector-triple, and the oscillation plane, spanned by the conjugate complex pair of eigenvectors.

2.2.4 Trivectors, 3-dimensional Objects

A trivector is a directed volume element, or 3-blade. It evolves from the outer product performed on three 1-blades. A trivectors characteristics are

- magnitude: absolute (scalar) value of its volume
- attitude of the volume element in space, defined by the three 1-blades
- orientation: righthanded or lefthanded, defined by the *first*, *second* and *third* defining 1-blade

A 3-dimensional (sub)space $\mathbb{R}^3$ is fully represented by the unit-trivector

$$\boldsymbol{I_3} \equiv \boldsymbol{e_1} \wedge \boldsymbol{e_2} \wedge \boldsymbol{e_3} \tag{2.10}$$

Note that $\boldsymbol{e_3} \wedge \boldsymbol{e_2} \wedge \boldsymbol{e_1} = -\boldsymbol{I_3}$, it is oriented lefthanded.
The definition of the orientation involves three basis elements as opposed to the bivector, the orientation of which comprises only two basic elements. Arbitrary trivectors in $\mathbb{R}^n$ are defined as

$$\boldsymbol{A} = \sum_n \alpha_i(\boldsymbol{e_i} \wedge \boldsymbol{e_j} \wedge \boldsymbol{e_k}) \text{ with } i, j, k \in [1, \dots n] \text{ and } i \neq j \neq k$$

Any trivector in $\mathbb{R}^3$ can be represented as a scalar multiple of *the* unit trivector $\boldsymbol{I_3}$. There is only one unit trivector in a 3-dimensional space since permutations only affect its sign. Hence a scalar value, be it positive or negaitve in sign, fully defines the trivector. Consequently, a trivector is also called pseudoscalar in $\mathbb{R}^3$. Note that this relation only holds for (sub)spaces of $\mathbb{R}^3$ but is handy when applicable.
Note, again, that the *shape* of the volume element is not defined and irrelevant. GABLE provides two different display options. The intuitive option is to represent trivectors as the spade spanned by the three defining 1-blades, indication first, second and third to capture orientation. The general option prints a sphere (rendered as line elements) with the orientation captured as surface-orthogonal lines pointing inwards or outwards. The volume represents the trivector's magnitude. One may use any shape that captures all three attributes. Fig. 2.3 depicts a trivector as the spade spanned by three 1-blades.

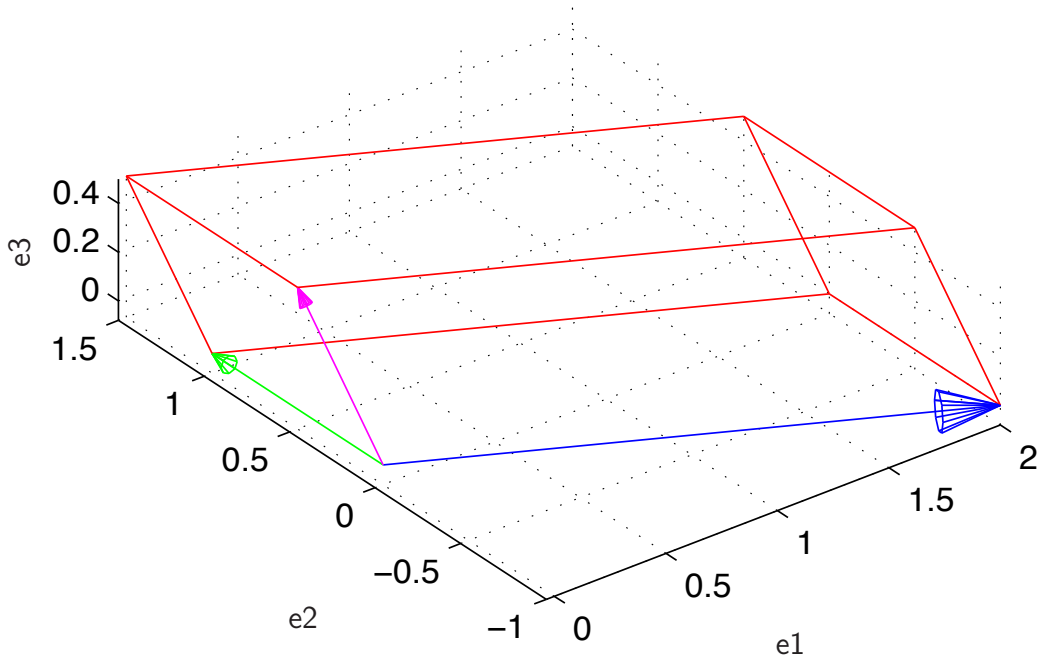


Figure 2.3: The trivector $(2\boldsymbol{e}_1 - \boldsymbol{e}_2) \wedge 1\boldsymbol{e}_2 \wedge 0.5(\boldsymbol{e}_2 + \boldsymbol{e}_3)$ oriented righthanded, from the blue, to the green, to the magenta 1-blade, or, from its first, to its second to its third component.

2.2.5 Multi-dimensional Objects

Objects of mixed dimension evolve from operations performed on blades of different grade. A mixed-grade-blade can contain elements of every possible subspace-dimension. Each object may contain the full number of scalar weighted standard elements. Hence, computations with multidimensional objects can become complicated and computationally expensive. Multidimensional objects can not easily be displayed, nor can such objects be physically interpreted. The application of the introduced interpolation method does not generate such objects under correct use.

2.2.6 Dual and Representation of an Object as a Pseudoscalar

The dual of an object $\mathbf{A}$ in $\mathbb{R}^n$ is defined to be

$$\text{dual}\boldsymbol{A} \equiv \boldsymbol{A}/\boldsymbol{I_3} = -\boldsymbol{A}\boldsymbol{I_3}, \tag{2.11}$$

it is the *orthogonal complement* of $\mathbf{A}$. In $\mathbb{R}^3$ the cross-product of two vectors returns their orthogonal complement with a magnitude of the scalar value of the parallelogram spanned by the two vectors.

$$\text{dual}(\boldsymbol{u} \wedge \boldsymbol{v}) = \boldsymbol{u} \times \boldsymbol{v} \text{ in } \mathbb{R}^3 \tag{2.12}$$

The dual of a plane in $\mathbb{R}^3$ is its surface normal. Hence, this plane is fully characterized with knowing its normal vector (its dual). Note that the dual of a plane in $\mathbb{R}^4$ is a plane, too. Knowing a two-dimensional plane's dual does not explicitly define the plane in $\mathbb{R}^4$.

2.3 Operations

The three basic operations in geometric algebra closely relate to the outer and inner product known in vector calculus. The major difference in GA is, that its operations are not limited to pure blades but can be applied to any object defined in the space considered. Knowing this, the result of these operations is not necessarily pure-dimensioned. Thus, results may not always be easily visualized.

2.3.1 Outer Product

The outer product in geometric algebra has the properties of *anti-symmetry*, *linearity* and *associativity*. Its assigned operator is the wedge $\wedge$. For arbitrary 1-blades (vectors) $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{w}$ the definition reads

$$\begin{aligned} \textit{anti-symmetry} \quad & \boldsymbol{v} \wedge \boldsymbol{w} = -\boldsymbol{w} \wedge \boldsymbol{v} \\ \textit{linearity} \quad & \boldsymbol{u} \wedge (\boldsymbol{v} + \boldsymbol{w}) = \boldsymbol{u} \wedge \boldsymbol{v} + \boldsymbol{u} \wedge \boldsymbol{w} \\ \textit{associativity} \quad & \boldsymbol{u} \wedge (\boldsymbol{v} \wedge \boldsymbol{w}) = (\boldsymbol{u} \wedge \boldsymbol{v}) \wedge \boldsymbol{v} \end{aligned}$$

With $\boldsymbol{v} \wedge \boldsymbol{v} = 0$ the outer product of parallel (here identical) objects is defined. The outer product of a vector $\boldsymbol{v}$ and a scalar α, or of two scalars α and β are

$$\begin{aligned} \alpha \wedge \boldsymbol{v} &= \alpha \boldsymbol{v} \\ \alpha \wedge \beta &= \alpha\beta \end{aligned}$$

The outer product of two 1-blades results in a 2-blade, a bivector. The outer product of two 1-blades in 3-dimensional geometric algebra relates to the cross product of two vectors in conventional vector calculus via the dual, see section 2.2.6.

2.3.2 Inner Product

The inner product can be applied to elements of any grade. It is neither *associative* nor *symmetric* but it is *linear*. For blades of different grade with r being the first argument's grade and s being the second argument's grade:

- if $r < s$, their inner product is of grade $(s - r)$, lies in the subspace of the second argument and is perpendicular to the first argument
- if $r = s$, their inner product is of grade $(s - r) = 0$, a scalar.
- if $r > s$, their inner product $= 0$.

It is a *contraction* because its arguments grades are reduced. Hence, if the first arguments grade is larger than the second arguments grade the inner product is zero.
In the special case of a 3-dimensional GA the inner product is exactly the inner (or dot) product known in conventional vector calculus. In $\mathbb{R}^3$, when performed on two 1-blades it is *symmetric* and *linear*. It is then defined

$$\boldsymbol{u} \cdot \boldsymbol{v} = \boldsymbol{v} \cdot \boldsymbol{u}$$

$$(\alpha\boldsymbol{u} + \beta\boldsymbol{v}) \cdot \boldsymbol{w} = \alpha(\boldsymbol{u} \cdot \boldsymbol{w}) + \beta(\boldsymbol{v} \cdot \boldsymbol{w})$$

$$\alpha \cdot \boldsymbol{u} = \alpha\boldsymbol{u} \text{ but } \boldsymbol{u} \cdot \alpha = 0$$

The inner product of 1-blades is a scalar and a measure for perpendicularity. The inner product of two unit vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ returns the projected length of $\boldsymbol{u}$ onto the axis of $\boldsymbol{v}$.

$$\boldsymbol{u} \cdot \boldsymbol{v} = |\boldsymbol{u}||\boldsymbol{v}| \cos\theta \tag{2.14}$$

2.3.3 Geometric Product

The geometric product is a combination of outer and inner product. It is defined to be *linear* and *associative* in its arguments and *distributive*. The exact definition for arbitrary multi-grade objects can be found in [1, section 2.5]. It is the only operation that captures *all* geometric relations between its arguments, not only those, that can be depicted easily. When performed on two blades of grade r and s its result is a multi-blade containing objects of grade $[|r - s|, |r - s| + 2, \ldots r + s - 2, r + s]$.
For 1-blades it is defined as

$$\boldsymbol{uv} = \boldsymbol{u} \wedge \boldsymbol{v} + \boldsymbol{u} \cdot \boldsymbol{v}$$

$$\boldsymbol{vu} = \boldsymbol{v} \wedge \boldsymbol{u} + \boldsymbol{u} \cdot \boldsymbol{v} = -\boldsymbol{u} \wedge \boldsymbol{v} + \boldsymbol{u} \cdot \boldsymbol{v}$$

Note here, that the geometric product is neither fully symmetric nor fully antisymmetric. The inner and outer product can now be defined as its symmetric respectively antisymmetric part.

$$\boldsymbol{u} \cdot \boldsymbol{v} = \tfrac{1}{2}(\boldsymbol{uv} + \boldsymbol{vu}) \tag{2.15}$$

$$\boldsymbol{u} \wedge \boldsymbol{v} = \tfrac{1}{2}(\boldsymbol{uv} - \boldsymbol{vu}) \tag{2.16}$$

With equations (2.4) and (2.14) the geometric product can be rewritten as

$$\begin{aligned} \boldsymbol{uv} &= \boldsymbol{u} \wedge \boldsymbol{v} + \boldsymbol{u} \cdot \boldsymbol{v} \\ &= |\boldsymbol{u}||\boldsymbol{v}|(\sin\theta \boldsymbol{i} + \cos\theta) \\ &= |\boldsymbol{u}||\boldsymbol{v}| \exp \boldsymbol{i}\theta \end{aligned} \tag{2.17}$$

2.3.4 Invertibility of blades

In geometric algebra of Euclidian space, subspaces (blades of pure grade) have an inverse. In a general space not all blades have an inverse. The inverse of an arbitrary pure blade $\boldsymbol{A} \neq 0$ is defined

$$\boldsymbol{A}\boldsymbol{A}^{-1} = 1$$

The inverse of 1-blades and 2-blades are defined to be

$$\boldsymbol{u}^{-1} = \frac{\boldsymbol{u}}{\boldsymbol{u} \cdot \boldsymbol{u}} \text{ and } \boldsymbol{A}^{-1} = \frac{\boldsymbol{A}}{\boldsymbol{A} \cdot \boldsymbol{A}}$$

Scalar multiples of blades are treated as in general calculus.

$$(\alpha \boldsymbol{I}_3)^{-1} = \alpha^{-1} \frac{\boldsymbol{I}_3}{\boldsymbol{I}_3 \cdot \boldsymbol{I}_3} = \alpha^{-1} \frac{\boldsymbol{I}_3}{-1} = -\boldsymbol{I}_3/\alpha$$

2.4 Projection and rejection

The *projection* of an object $\boldsymbol{u}$ onto another object $\boldsymbol{v}$ results in the part of $\boldsymbol{u}$ that is parallel to $\boldsymbol{v}$ *and* lies within the subspace $\boldsymbol{v}$. However, $\boldsymbol{u}$ is usually not within the subspace of $\boldsymbol{v}$. The grades of the two arguments may differ, but the grade of the first argument must be lower than or equal to the grade of the second argument upon which it is projected.
The *rejection* of an object $\boldsymbol{u}$ and another object $\boldsymbol{v}$ results in the part of $\boldsymbol{u}$ that is perpendicular to $\boldsymbol{v}$ and therefore *not* within the subspace of $\boldsymbol{v}$. Again, the rejection will usually not be within the subspace $\boldsymbol{u}$.
Both properties describe geometric relations between blades (or subspaces) and both can be computed with geometric algebra. Given an arbitrary 1-blade $\boldsymbol{v}$ and an arbitrary 2-blade $\boldsymbol{M}$ (a vector and a plane, for easy display) one can derive: The vector can be rewritten as $\boldsymbol{v} = \boldsymbol{v}_\perp + \boldsymbol{v}_\parallel$, a component $\boldsymbol{v}_\perp$ that is perpendicular to the subspace $\boldsymbol{M}$ and a component $\boldsymbol{v}_\parallel$ that is parallel to $\boldsymbol{M}$. With the inner and outer product $\boldsymbol{v}_\perp \cdot \boldsymbol{M} = 0$ and $\boldsymbol{v}_\parallel \wedge \boldsymbol{M} = 0$

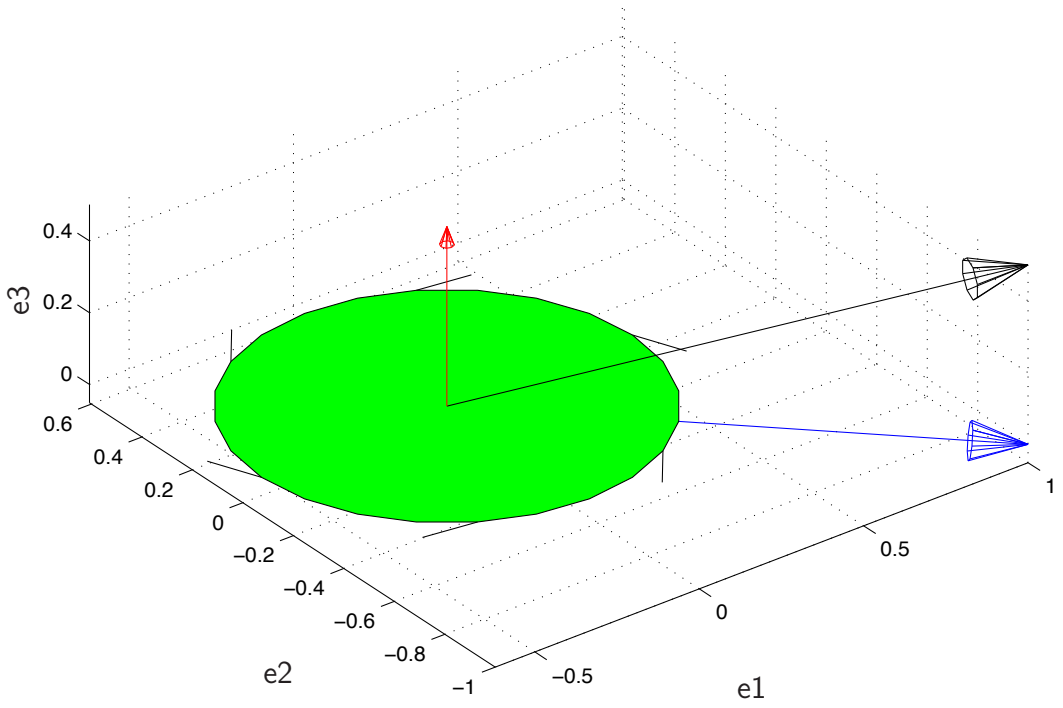


Figure 2.4: A vector $\boldsymbol{u} = 1\boldsymbol{e}_1 - 1\boldsymbol{e}_2 + 0.5\boldsymbol{e}_3$ (black) is projected onto the bivector $\boldsymbol{e}_1 \wedge (\boldsymbol{e}_1 + \boldsymbol{e}_2)$ (green). The Rejection $\boldsymbol{v}_\perp$ of the vector (red) with respect to the bivector is its component orthogonal to the bivector. The projection of the vector (blue) lies within the subspace of the bivector.

follows. Thus

$$\begin{aligned} \boldsymbol{v}_\perp \boldsymbol{M} &= \boldsymbol{v}_\perp \cdot \boldsymbol{M} + \boldsymbol{v}_\perp \wedge \boldsymbol{M} \\ &= \boldsymbol{v}_\perp \wedge \boldsymbol{M} \\ &= \boldsymbol{v}_\perp \wedge \boldsymbol{M} + \boldsymbol{v}_\parallel \wedge \boldsymbol{M} \\ &= \boldsymbol{v} \wedge \boldsymbol{M} \end{aligned} \tag{2.18}$$

Deviding (2.18) by $\boldsymbol{M}$ from the right hand side yields

$$\boldsymbol{v}_\perp = (\boldsymbol{v} \wedge \boldsymbol{M})/\boldsymbol{M} \tag{2.19}$$

In a similar fashion one derives

$$\boldsymbol{v}_\parallel = (\boldsymbol{v} \cdot \boldsymbol{M})/\boldsymbol{M} \tag{2.20}$$

From the above derivation another important statement is gathered: Dividing a space $\boldsymbol{B}$ by a subspace $\boldsymbol{A}$ yields the orthogonal complement to $\boldsymbol{A}$ in $\boldsymbol{B}$.

2.5 Reflection

An arbitrary blade $\boldsymbol{v} = \boldsymbol{v}_{\parallel} + \boldsymbol{v}_{\perp}$ is reflected on a subspace $\boldsymbol{M}$ and yields

$$_{\top}\boldsymbol{v} = \boldsymbol{v}_{\parallel} - \boldsymbol{v}_{\perp}$$

The component parallel to $\boldsymbol{M}$ remains unchanged while the component orthogonal to $\boldsymbol{M}$ is reflected relative to $\boldsymbol{M}$.
Generally an object's reflection on a blade is defined as

$$_{\top}\boldsymbol{v} = (-1)^{s+1}\boldsymbol{M}\boldsymbol{v}\boldsymbol{M}^{-1} \tag{2.21}$$

where s denotes the grade of $\boldsymbol{M}$.

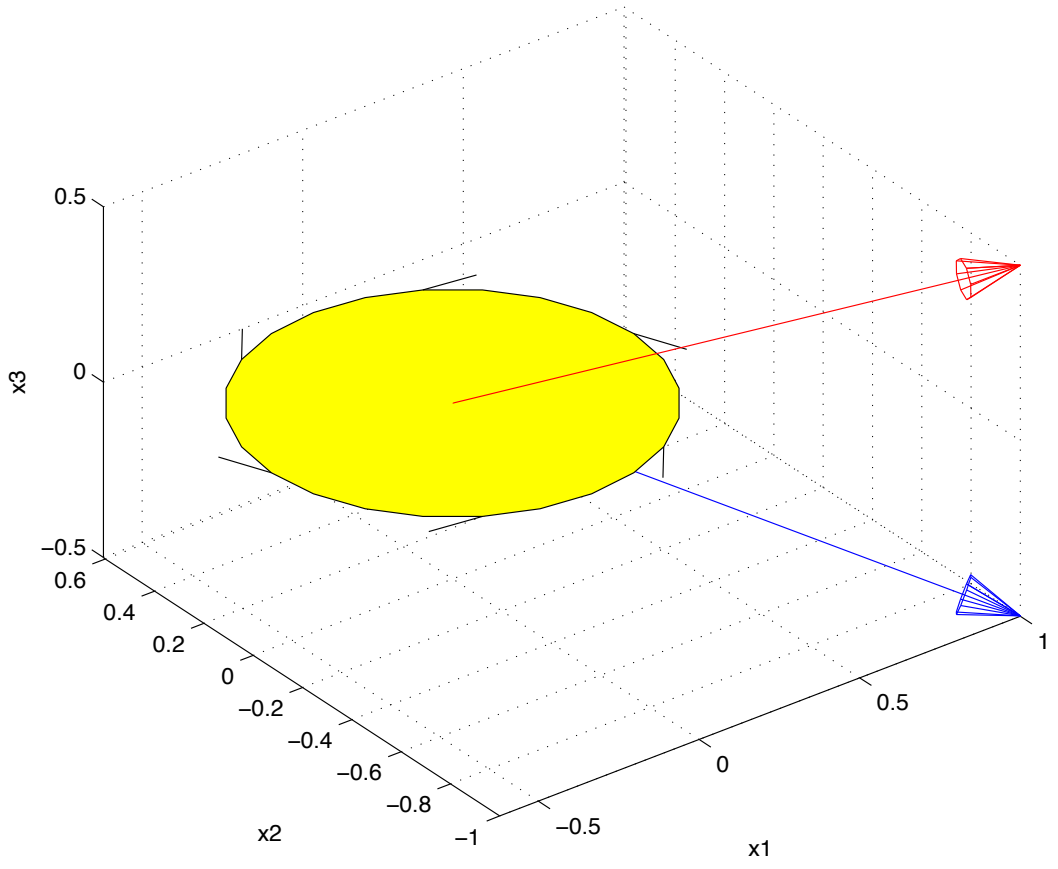


Figure 2.5: A vector $\boldsymbol{u} = 1\boldsymbol{e}_1 - 1\boldsymbol{e}_2 + 0.5\boldsymbol{e}_3$ is reflected on the bivector $\boldsymbol{e}_1 \wedge \boldsymbol{e}_2$. [1]

2.6 Spinors, representation of orientation and rotation

The rotation of an object $\boldsymbol{u}$ can be interpreted as two reflections performed in a row.

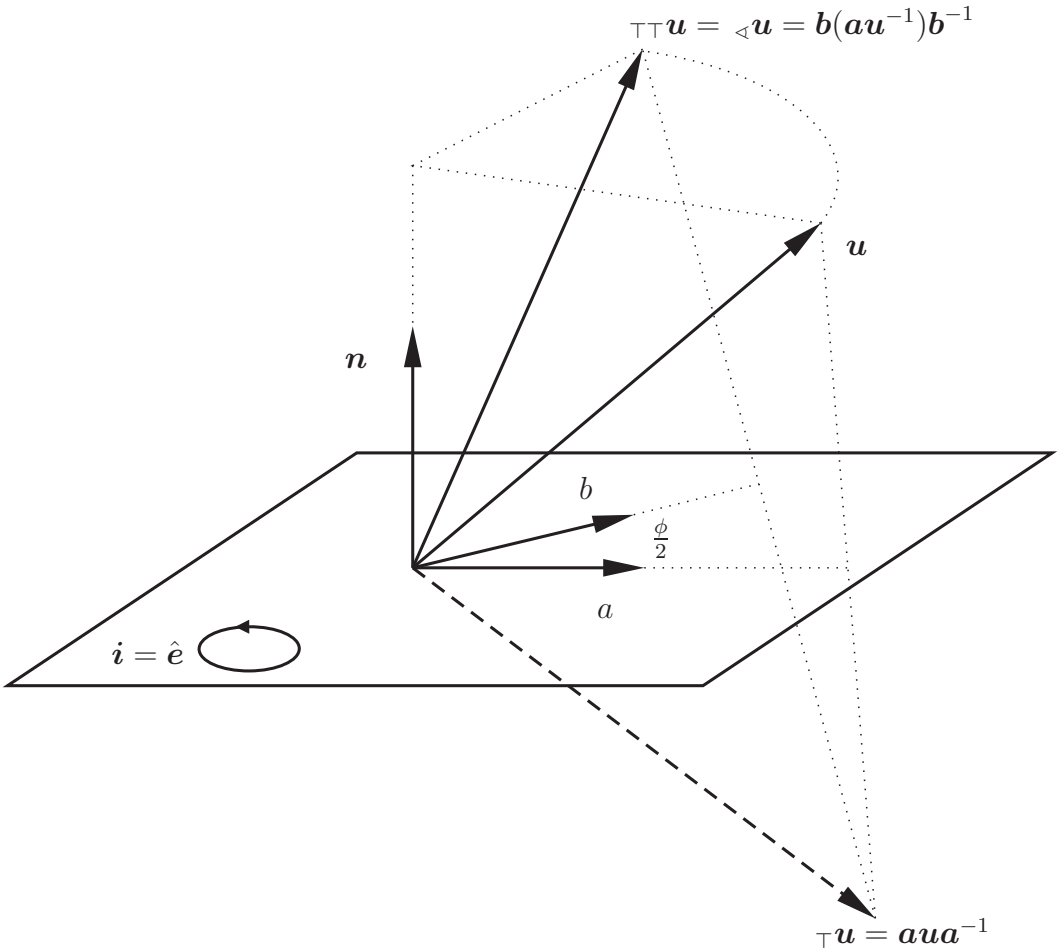


Figure 2.6: A vector $\boldsymbol{u}$ is rotated by ϕ along the bivector $\boldsymbol{i}$ [1].

First, the object $\boldsymbol{u}$ is reflected in a 1-blade $\boldsymbol{a}$ yielding $_{\top}\boldsymbol{u} = \boldsymbol{a}\boldsymbol{u}\boldsymbol{a}^{-1}$. In a second step the new vector $_{\top}\boldsymbol{u}$ is reflected in $\boldsymbol{b}$ yielding $_{\top\top}\boldsymbol{u} = \boldsymbol{b}(_{\top}\boldsymbol{u})\boldsymbol{b}^{-1}$. The pre-subscript $\top$ indicates that the vector is being reflected. Consequently the pre-subscript $\top\top$ indicates that the vector is being reflected twice. Since reflecting a vector twice is equivalent to rotating it, the pre-subscript $\lhd$ is equivalent to $\top\top$. Both $\boldsymbol{a}$ and $\boldsymbol{b}$ are chosen to be unit vectors. This and equation (2.17) lead to the following

$$_{\top\top}\boldsymbol{u} = {}_{\lhd}\boldsymbol{u} = \boldsymbol{b}(\boldsymbol{a}\boldsymbol{u}\boldsymbol{a}^{-1})\boldsymbol{b}^{-1} = (\boldsymbol{b}\boldsymbol{a})\boldsymbol{u}(\boldsymbol{b}\boldsymbol{a})^{-1} = e^{-\boldsymbol{i}\phi/2}\boldsymbol{u}e^{\boldsymbol{i}\phi/2}$$

where $\boldsymbol{i}$ denotes a unit element of the plane that is spanned by $\boldsymbol{a}$ and $\boldsymbol{b}$ and $\frac{\phi}{2}$ is the angle between $\boldsymbol{a}$ and $\boldsymbol{b}$. The bivector $\boldsymbol{i}$ represents the unit-rotation-plane spanned by the two unit vectors $\boldsymbol{a}$ and $\boldsymbol{b}$. Note that the object $\boldsymbol{u}$ is rotated *by* the angle ϕ measured in $\boldsymbol{i}$. The rotation is hence characterized by left- and right-multiplication with $\boldsymbol{R} = e^{-\boldsymbol{i}\phi/2}$, the *spinor* or *rotor*. For arbitrary blades the rotation with spinors is defined

$$_{\lhd}\boldsymbol{X} = \boldsymbol{R}\boldsymbol{X}\boldsymbol{R}^{-1} \tag{2.22}$$

A spinor characterizes a rotation in $\mathbb{R}^n$. Consecutive rotations are easily computed

$$\boldsymbol{R}_2(\boldsymbol{R}_1\boldsymbol{X}\boldsymbol{R}_1^{-1})\boldsymbol{R}_2^{-1} = (\boldsymbol{R}_2\boldsymbol{R}_1)\boldsymbol{X}(\boldsymbol{R}_2\boldsymbol{R}_1)^{-1}$$

Where $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$ are two spinors applied one after the other. Note here, that the operation defined by spinors is *not* commutative:

$$e^{-\boldsymbol{i_2}\phi_2/2}e^{-\boldsymbol{i_1}\phi_1/2} \neq e^{-(\boldsymbol{i_2}\phi_2/2+\boldsymbol{i_1}\phi_1/2)}$$

The exponent of the spinor contains the rotation information - the bivector $\boldsymbol{i}$ that defines the rotation plane and the angle of rotation ϕ with respect to that plane. To move back and forth between the spinor and its exponent the inverse of the exponential - the logarithm - is defined as well. Because

$$\boldsymbol{R} = e^{(-\boldsymbol{i}\phi/2)} = e^{(-\boldsymbol{i}\phi/2+2\pi\boldsymbol{i})}$$

the logarithm is *not* unique. It is convenient to chose the value of the logarithm from the interval $[-\pi\boldsymbol{i}, \pi\boldsymbol{i}]$.

2.7 Interpolation of Orientation with Spinors

In GA an object has a certain orientation with respect to the basic elements that are defined in the space considered. The orientation information is inherent to the object. Every object can be interpreted as a standard object of the same dimension rotated by a certain angle with respect to a certain object (a plane, in many cases). With the logarithm the argument of an objects exponential description, see (2.17), can be extracted. It contains the plane and angle of rotation that transforms a basic element of given dimension onto the object of interest. It follows that every object can be described by its spinor, or in other words, by its orientation. Knowing that a spinor encodes orientation, the orientation itself can be interpolated elegantly.

Two objects $\boldsymbol{A}$ and $\boldsymbol{B}$ are characterized by their orientations $\boldsymbol{R}_A$ and $\boldsymbol{R}_B$ with respect to the basis elements. A smooth transition from orientation $\boldsymbol{R}_A$ to $\boldsymbol{R}_B$ is achieved by n identical rotation operators $\boldsymbol{R}$ applied to $\boldsymbol{R}_A$ so as to finally reach $\boldsymbol{R}_B$. The subsequent rotations are

$$\boldsymbol{R}_0 = \boldsymbol{R}_A; \quad \boldsymbol{R}_{i+1} = \boldsymbol{R}\boldsymbol{R}_i; \quad \boldsymbol{R}_n = \boldsymbol{R}_B$$

The total rotation from $\boldsymbol{R}_A$ to $\boldsymbol{R}_B$ is $\boldsymbol{R}_B\boldsymbol{R}_A^{-1}$. Since the entire rotation is covered in n steps one defines

$$\boldsymbol{R}_B\boldsymbol{R}_A^{-1} = \boldsymbol{R}^n = e^{-\boldsymbol{I_3}\boldsymbol{a}\phi/2}$$

$$\boldsymbol{R} = (e^{-\boldsymbol{I_3 a}\phi/2})^{(1/n)} = e^{-\boldsymbol{I_3 a}\phi/(2n)}$$

The object $\boldsymbol{a}$ denotes the rotation axis, the dual of the rotation plane $\boldsymbol{i}$. It is defined by

$$\boldsymbol{a} \equiv \text{dual}(\boldsymbol{i})$$

Hence, with the definition of the dual given by equation 2.11: $\boldsymbol{i}\frac{\phi}{2} \leftrightarrow \boldsymbol{I_3 a}\frac{\phi}{2}$ in $\mathbb{R}^3$ because a unique dual of the plane $\boldsymbol{i}$ exists.
The angle ϕ denotes the rotation angle. Note that the unique representation with duals is only possible in $\mathbb{R}^3$. In $\mathbb{R}^n$ a rotation planes dual is of dimension $(n-2)$.

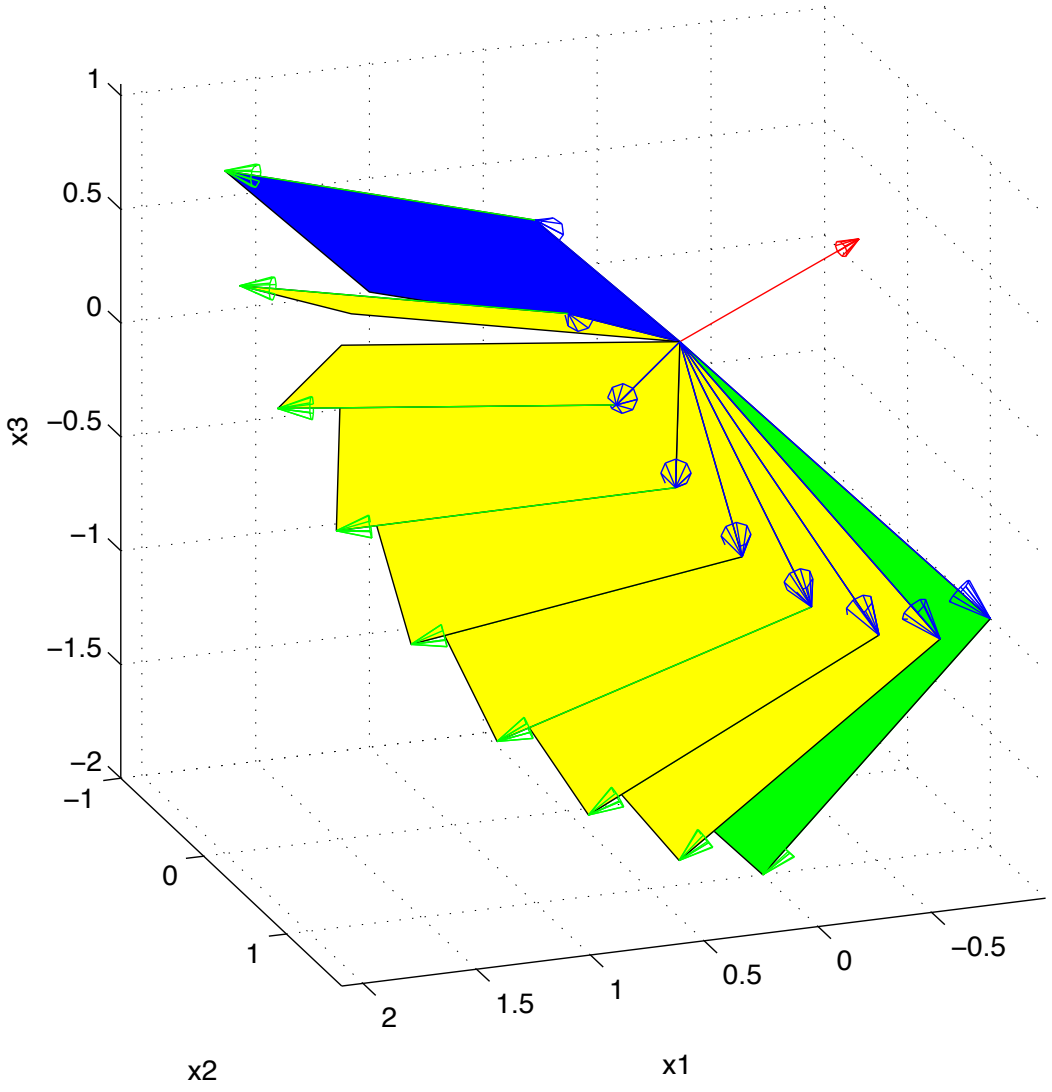


Figure 2.7: The interpolation of bivectors (blue and green planes) and intermediate steps (yellow planes). The red vector indicates the effective axis of "rotation" in the interpolation procedure.

Chapter 3

Interpolation of State Space Systems

The interpolation method can be applied to both discrete-time and continuous-time systems. Explanations in the following chapters will point out the differences between discrete- and continuous-time systems.
The physical interpretation of state space matrices helps to determine the quality of the interpolation. Of the desired interpolation characteristics stated in section 1.2.1 all except the last are related to the system matrix $\boldsymbol{A}$.

3.1 The System Matrix

The system matrix $\boldsymbol{A}$ is of dimension [n $\times$ n]. It encodes the dynamic characteristics of the system: natural frequency, damping ratio and the general "speed" of the system. Also, the attitude of the oscillation plane in the state space as well as its orientation are encoded in the system matrix.
All of the above values are encoded in the eigenvalues and eigenvectors of the system matrix. Eigenvectors and eigenvalues are computed as the solutions of the eigenvalue problem given by (3.1). As explained in section 3.1.1 the solutions are classified by the eigenvalues and form three groups.

3.1.1 System Characteristics, Eigenvectors and Eigenvalues

The eigenvectors and eigenvalues of the system matrix $\boldsymbol{A}$ characterize the system dynamics and are computed as a solution of the eigenvalue problem

$$(\boldsymbol{A} - \lambda_n \boldsymbol{I})\boldsymbol{u}_n = \boldsymbol{0} \quad \boldsymbol{A} \in \mathbb{R}^{n \times n} \tag{3.1}$$

where λ_n denotes the eigenvalues and $\boldsymbol{u}_n$ denotes the corresponding eigenvectors. The solutions can be classified by three general cases:

λ is real valued, multiplicity $= 1$: Its corresponding eigenvector is also real valued. If the state vector $\boldsymbol{x}$ aligns with the eigenvector its trajectory will keep that direction. Depending on the eigenvalue the trajectory will proceed towards the origin or away from it, making this particular mode either stable or unstable. For continuous-time systems negative eigenvalues correspond to a stable mode, while positive eigenvalues correspond to an unstable mode. Large negative values make a mode "fast" while low negative values (closer to the origin) make a mode "slow". For discrete-time systems eigenvalues inside the unit circle correspond to stable modes, eigenvalues outside the unit circle correspond to unstable modes.

λ is a conjugate complex pair: Its corresponding eigenvectors are also a conjugate complex pair, spanning a plane in the state space. This plane is this particular mode's oscillation plane. If the initial state $\boldsymbol{x}(\boldsymbol{t}_0)$ (or $\boldsymbol{x}(k = 0)$) lies on that plane, the state vector's trajectory will remain on that plane and the system will oscillate.

λ is real valued, multiplicity > 1: This case represents aperiodic oscillation. A pole with multiplicity $= 2$ marks a separation point in the root locus of the system.

3.1.2 Eigenvalue λ is Real Valued with Multiplicity $= 1$

Eq. (3.1) is rewritten

$$\boldsymbol{A}\boldsymbol{u}_n = \lambda_n \boldsymbol{u}_n \tag{3.2}$$

Depending on λ a particular mode is either stable, critically stable or unstable and at the same time fast or slow relative to other modes of the system. The "speed" of a mode determines, whether the state vector that aligns with the eigenvecor at some point travels towards the origin (equilibrium point) on a rather short path or a longer path. For discrete-time systems eigenvalues inside the unit circle correspond to stable modes, eigenvalues outside the unit circle correspond to unstable modes.

3.1.3 Eigenvalue λ is a Conjugate Complex Pair

Conjugate complex pairs denote oscillatory modes. Eq. (3.1) is rewritten

$$\boldsymbol{A}(\boldsymbol{u} \pm i\boldsymbol{v})_n = (\mu \pm i\nu)_n(\boldsymbol{u} \pm i\boldsymbol{v})_n \tag{3.3}$$

where $(\boldsymbol{u} + i\boldsymbol{v})_n$ is the conjugate complex eigenvector that corresponds to the conjugate complex eigenvalue $(\mu + i\nu)_n$ and $(\boldsymbol{u} - i\boldsymbol{v})_n$ is the conjugate complex eigenvector that cor-

responds to the conjugate complex eigenvalue $(\mu - i\nu)_n$. While μ indicates the "speed" of the mode, the complex component ν indicates the oscillation frequency of the mode.
Splitting (3.3) into a real valued part and a complex valued part results in

$$\boldsymbol{A}\boldsymbol{u} = \mu\boldsymbol{u} - \nu\boldsymbol{v} = \begin{bmatrix} \mu & -\nu \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{v} \end{bmatrix} \tag{3.4}$$

$$\boldsymbol{A}\boldsymbol{v} = \mu\boldsymbol{v} + \nu\boldsymbol{u} = \begin{bmatrix} \nu & \mu \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{v} \end{bmatrix} \tag{3.5}$$

The oscillation plane is spanned by the components of the eigenvector $\boldsymbol{w} = \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{v} \end{bmatrix}$. The trajectory of a stable oscillating system travels towards the oscillation plane and, once on the oscillation plane, spirals into the equilibrium point. The spiralling direction is encoded in $\boldsymbol{w}$: from the first component $\boldsymbol{u}$ to the second component $\boldsymbol{v}$.
It is important to note that $\boldsymbol{w}$ is defined by (3.3). Both the sign and the order of the components are explicitly defined. There is no need to manually pick the eigenvectors that define the oscillation plane.

3.1.4 Eigenvalue λ is Real Valued with Multiplicity = 2

This case denotes the aperiodic oscillation. If the system matrix can be transformed to diagonal form (3.2) yields two linear-independent eigenvectors. Otherwise, if it can be transformed to Jordan canonical form, only one linear independent eigenvector exists. However, a generalized eigenvector can be constructed. If the system matrix is of general form and can not be transformed into any of the above stated standard forms only one linear independent eigenvector exists. Hence, the dimension n of the space spanned by the eigenvectors is ≤ 2.
In control engineering the eigenvalue constellation of a system matrix indicates qualitative system behavior. Looking at a root locus plot, this case represents a separation point of the branches of the root locus of a system.

Example 3.1: The system matrix defines dynamic characteristics

> This example serves to underline the practical relevance of the characteristics encoded by the system matrix $\boldsymbol{A}$. A simple system with one real valued eigenvalue and a conjugate complex pair of eigenvalues is sufficient.
>
> $$\boldsymbol{A}_1 = \begin{bmatrix} 0 & 1 & 0 \\ -b1 & -a1 & 0 \\ 0 & 0 & -c1 \end{bmatrix}, \boldsymbol{A}_2 = \begin{bmatrix} 0 & 1 & 0 \\ -b2 & -a2 & 0 \\ 0 & 0 & -c2 \end{bmatrix}$$

The system matrix is of a special structure, so that the natural frequency, damping ratio and speed of the system are encoded by the values a, b and c.
To emphasize the influence of the real valued eigenvalue the following explicit numbers are chosen: $a1 = a2 = 0.1, \quad b1 = b2 = 0.2, \quad c1 = -1, c2 = -5$.
The corresponding transfer functions compute to

$$\boldsymbol{G_{o,1}} = \frac{1s^2 + 0.9s - 0.1}{s^3 + 1.1s^2 + 0.3s + 0.2} \tag{3.6}$$

and

$$\boldsymbol{G_{o,2}} = \frac{1s^2 + 4.9s - 0.5}{s^3 + 5.1s^2 + 0.7s + 1} \tag{3.7}$$

Fig. 3.1 depicts trajectories of state vectors of the two systems with different real valued eigenvalues but otherwise identical eigenvalues. The oscillation plane is depicted as a yellow circle, the arrow elements along the circumfence denote the orientation and indicate the spiralling direction of the trajectory. The vector triple consists of the eigenvectors of the first system.

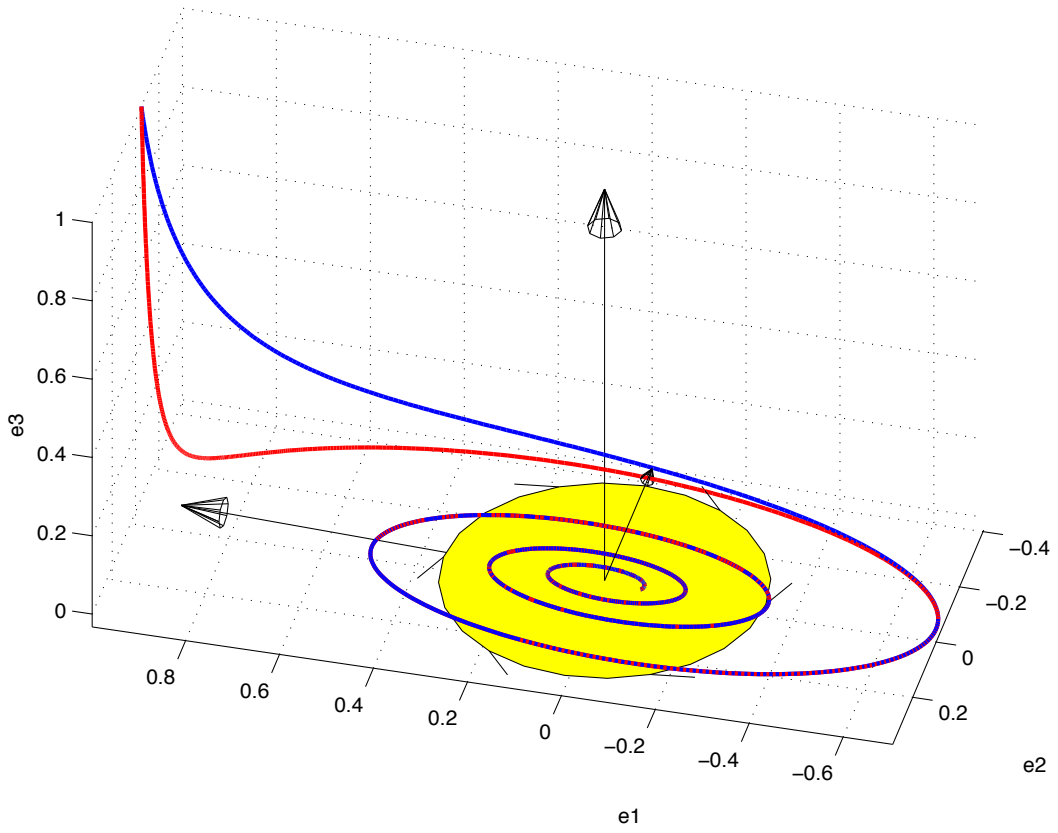


Figure 3.1: State vector trajectories of a system with varying real valued poles. The faster system (red, solid) reaches the oscillation plane well before the slower system (blue, solid).

Example 3.2: The system matrix defines dynamic characteristics - cont'd

To emphasize the influence of the damping ratio the following explicit numbers are chosen: a1 = 0.1, a2 = 0.3, b1 = b2 = 0.2, c1 = c2 = -1.
The corresponding transfer functions compute to

$$\boldsymbol{G_{o,1}} = \frac{1s^2 + 0.9s - 0.1}{s^3 + 1.1s^2 + 0.3s + 0.2} \tag{3.8}$$

and

$$\boldsymbol{G_{o,2}} = \frac{1s^2 + 1.1s + 1}{s^3 + 1.3s^2 + 0.5s + 0.2} \tag{3.9}$$

Fig. 3.2 depicts trajectories of state vectors of the two systems with different damping ratios but otherwise identical characteristics. The system $\boldsymbol{A}_1$ with less damping (blue, solid) oscillates longer until it eventually reaches the equilibrium point, while the system $\boldsymbol{A}_2$ with larger damping (red, solid) reaches the equilibrium point within the simulation time.

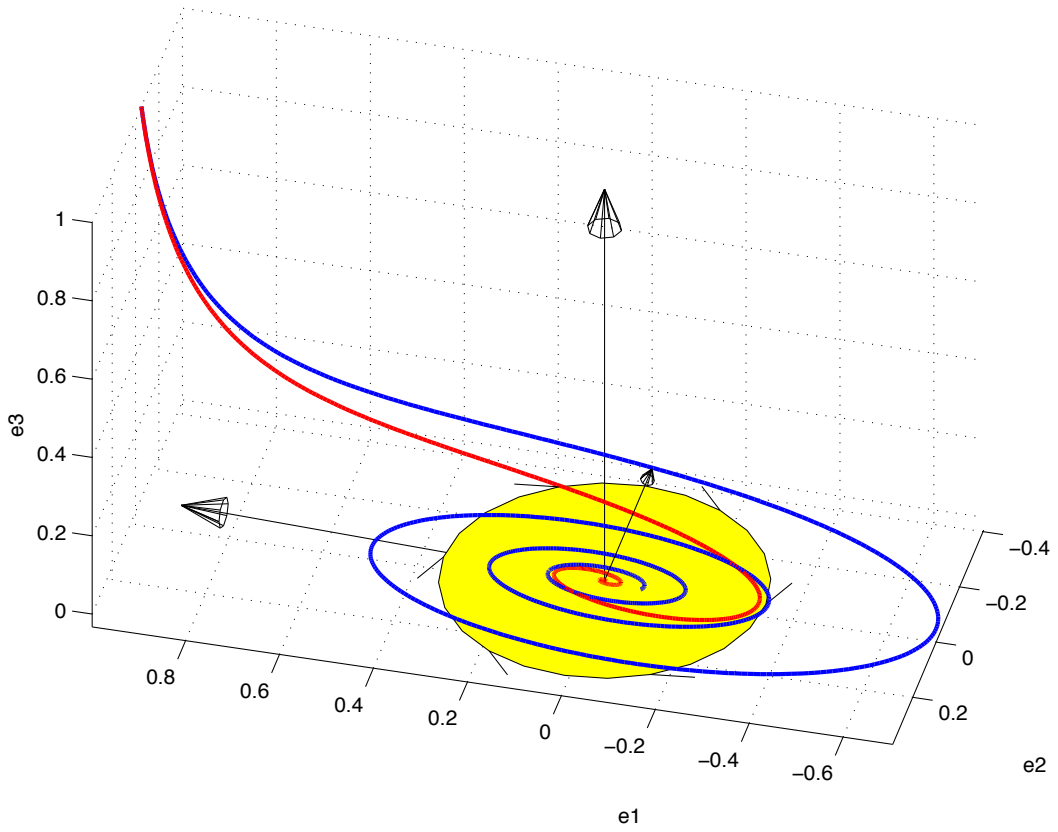


Figure 3.2: State vector trajectories of a system with varying damping ratio. The system with larger damping (red) reaches the equilibrium point within the simulation time, while the system with less damping (blue, solid) still oscillates until it eventually reaches the origin.

Example 3.3: The system matrix defines dynamic characteristics - cont'd

To emphasize the influence of the natural frequency the following explicit numbers are chosen: a1 = 0.1, a2 = a1$\sqrt{b2/b1}$, b1 = 0.2, b2 = 0.4 c1 = c2 = -1.
The corresponding transfer functions compute to

$$\boldsymbol{G_{o,1}} = \frac{1s^2 + 0.9s - 0.1}{s^3 + 1.1s^2 + 0.3s + 0.2} \tag{3.10}$$

and

$$\boldsymbol{G_{o,2}} = \frac{1s^2 + 0.5732s - 0.4268}{s^3 + 1.1732s^2 + 0.7732s + 0.6} \tag{3.11}$$

Fig. 3.3 depicts trajectories of state vectors of the two systems with different natural frequencies but otherwise identical characteristics. The system $\boldsymbol{A}_1$ with the lower natural frequency (blue, solid) oscillates fewer times within the simulation time, while the system $\boldsymbol{A}_2$ with higher natural frequency (red, solid) oscillates more often within the simulation time.

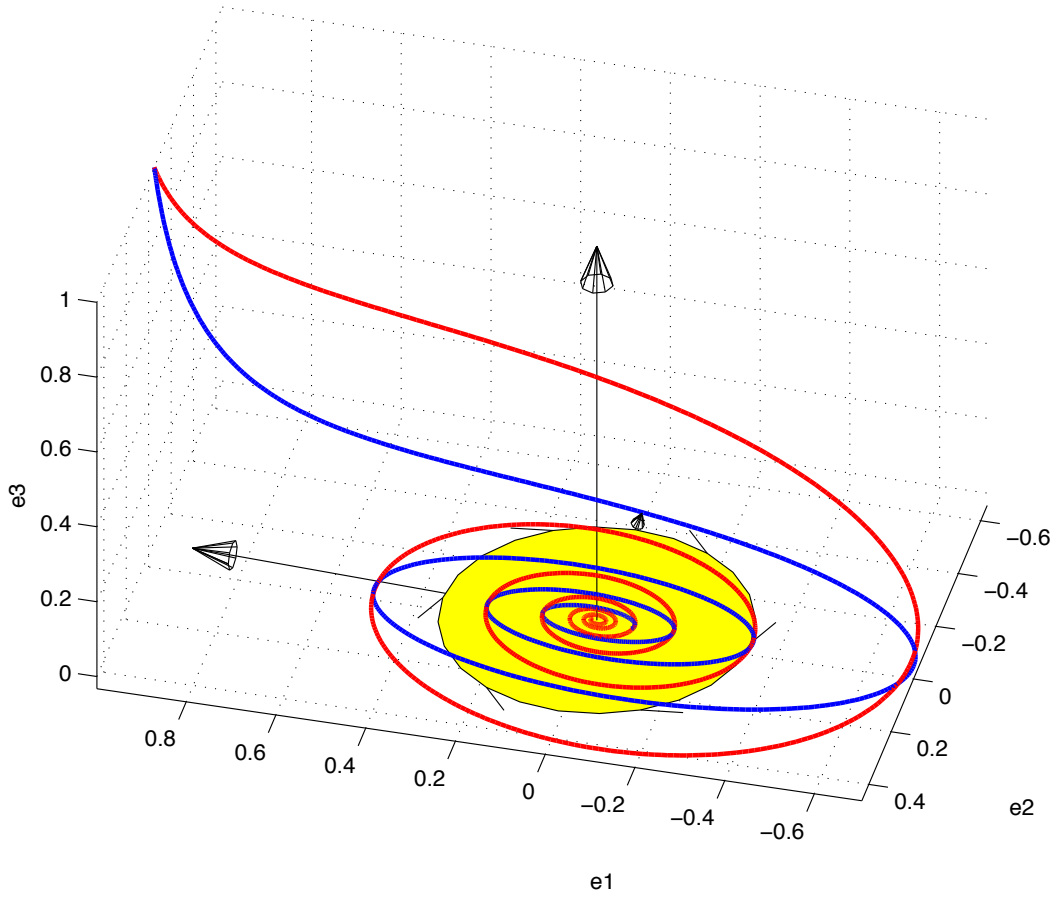


Figure 3.3: State vector trajectories of a system with varying natural frequency. The system with higher natural frequency (red) oscillates more often, compared to the system with lower natural frequency (blue).

3.2 Eigenvalue Constellation Combinations

Independent of the system dimension the introduced interpolation method can be applied by breaking down the problem to eigenvalue-eigenvector combinations with a maximum order of two. Every possible combination of the above introduced general cases is discussed here. For demonstration $n = 2$ is sufficient and only two systems are considered. The first index i denotes the system, and the second index denotes the eigenvalue. The distinguished combinations are

- $\lambda_{i,1}$ **and** $\lambda_{i,2}$ **are real valued, multiplicity** $= 1$**:** The eigenvalue determines, whether a particular mode is slow or fast. Thus, a linear interpolation of eigenvalues results in linear interpolation of the speed of a system's mode.

- $\lambda_{i,1}$ **and** $\lambda_{i,2}$ **form a complex conjugate pair:** This case represents the most challenging constellation and is covered in detail in this work. Conjugate complex pairs of eigenvalues contain the damping ratio and natural frequency of the particular mode. The eigenvalues are interpolated linearly. Thereby damping ratio and natural frequency are interpolated in a physically reasonable way. The corresponding conjugate complex pair of eigenvectors spans the oscillation plane of the mode. The oscillation planes, as part of characteristics of the system, must be interpolated as well. The attitude and orientation of the oscillation planes is interpolated using geometric algebra and, more specifically, the bivector introduced in section 2.2.3.

- $\lambda_{i,1}$ **and** $\lambda_{i,2}$ **are identical, with a multiplicity of two:** Considering a root locus, this case means, that two uncanceled poles of a system coincide and mark the separation point. Interpolating eigenvectors physically means that the new system's separation point is between the two given points. The speed of the new system is interpolated, while the (aperiodic) oscillation characteristics are retained.

- **mixed cases:** The introduced method can be applied to mixed cases with limitations.
  - The interpolation between two different real valued eigenvalues and a real eigenvalue with multiplicity $= 2$ is possible.
  - The interpolation between a conjugate complex pair and a real eigenvalue with multiplicity $= 2$ is possible.
  - The interpolation between a conjugate complex pair and two different real valued eigenvalues is *not yet* possible. The fact that the transition from eigenvalues in the complex plane to eigenvalues on the real axis is not defined causes problems when computing the corresponding eigenvectors. This is the only case to which the introduced method can not be reliably applied yet and requires further research.

The three pure interpolation cases (the first three in the above list, in contrast to the mixed cases) are depicted in Fig. 3.4. Two exemplary eigenvalue constellations (crosses and plus signs) of systems of dimension $n = 5$ are interpolated. Corresponding eigenvalues are interpolated linearly and yield the eigenvalues (triangles) at the interpolation weight $\Phi_1 = \Phi_2 = 0.5$.

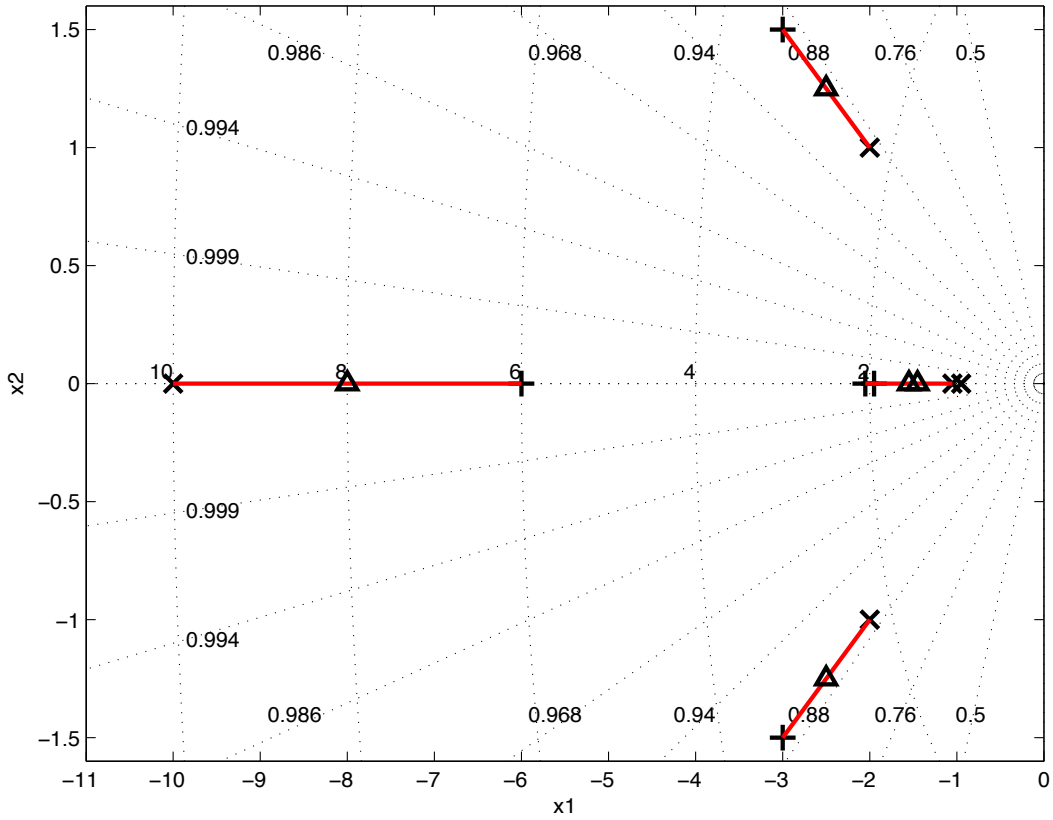


Figure 3.4: The interpolation of eigenvalues in pure cases. Eigenvalues with multiplicity = 2 are marked with two symbols. The eigenvalues of the first system (crosses) and of the second system (plus signs) yield the eigenvalues of the interpolated system (triangles) at $\Phi_1 = \Phi_2 = 0.5$.

Fig. 3.5 depicts an exemplary eigenvalue constellation of three systems of dimension $n = 2$.

1. System $\boldsymbol{A}_1$: A system with two different, real valued eigenvalues which are marked by crosses. The corresponding interpolation weight is Φ_1.

2. System $\boldsymbol{A}_2$: A system with a conjugate complex couple of eigenvalues, marked by plus signs. The corresponding interpolation weight is Φ_2.

3. System $\boldsymbol{A}_3$: A system with one real valued pole with multiplicity = 2, marked by triangles. The corresponding interpolation weight is Φ_3.

Interpolation between the first system $\boldsymbol{A}_1$ (crosses) with two different real valued eigenvalues and the third system $\boldsymbol{A}_3$ (triangles) with one real valued eigenvalue with multiplicity = 2

(separation point) yields a constellation with two real valued eigenvalues located between the original eigenvalues, on either side of the separation point somewhere along the red line, depending on the interpolation weights Φ_1 and Φ_3.
Interpolation between the second system $\boldsymbol{A}_2$ (plus signs) with a conjugate complex pair of eigenvalues and the third system $\boldsymbol{A}_3$ yields a conjugate complex pair of eigenvalues along the red line, depending on the interpolation weights Φ_2 and Φ_3.
The third mixed case is represented by the interpolation between system one (crosses) and three (plus signs). This particular case demands attention insofar as at some point the interpolated eigenvalues coincide with the separation point. Since it is not clear at which interpolation weight combination the interpolated eigenvalues coincide and additionally, the corresponding eigenvectors are not known it is not possible to interpolate across this transition point. This interpolation case is not part of this study and requires further research.

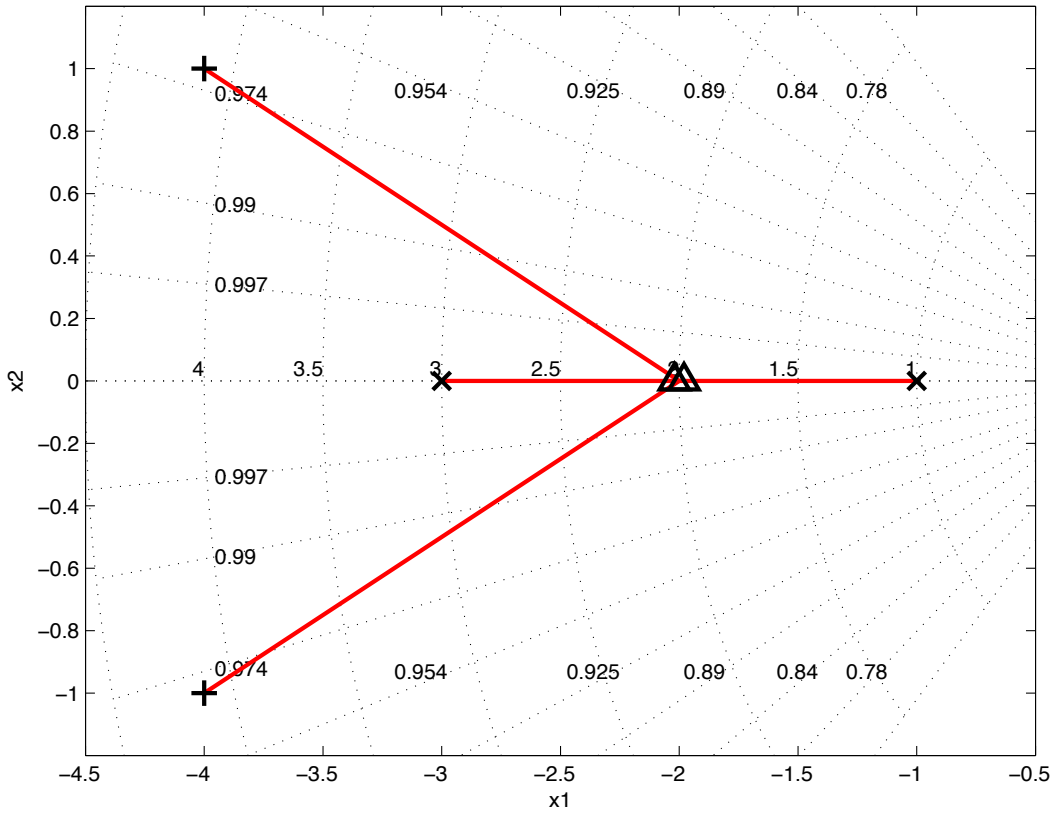


Figure 3.5: The interpolation of eigenvalues for mixed cases. Eigenvalues with multiplicity are marked with two symbols. The eigenvalues of the first system (crosses), eigenvalues of the second system (plus signs) and the eigenvalues of the third system (triangles) contain all three possible mixed cases.

Every system matrix interpolation can be broken down into interpolating the set of modes of the system matrix $\boldsymbol{A}$. This process is based on eigenvalue decomposition [18] of the system

matrix.
Thereby the problem is reduced to the six constellation combinations introduced above.
Except oscillation planes, which are spanned by complex conjugate pairs of eigenvectors, the interpolation can be accomplished by common vector algebra. For the interpolation of oscillation planes Geometric Algebra is best suited. Geometric Algebra provides an object, the bivector, see section 2.2.3, that naturally incorporates the rotational direction that characterizes an oscillation plane.

3.3 Mode Tracking

Breaking the entire system matrix $\boldsymbol{A}$ down to a number of modes, namely oscillating modes of dimension $n = 2$ and non-oscillating modes of dimension $n = 1$ can be seen as a preparation step. The process of interpolating modes demands that only corresponding modes are interpolated. Otherwise, the results will not reflect the actual system behavior in the interpolated operating point. The problem of mode tracking has not been addressed in this work. A solution is not yet readily available and is subject to further research. However, ad-hoc methods may be applicable, see section 3.3.1 and 3.3.2.

3.3.1 Mode Tracking by Eigenvalue

In case the number and nature of modes of the system matrices to be interpolated is identical it may be suitable to sort modes by the value of the corresponding eigenvalue. This approach can not be used if one or more modes appear only in one of the system matrices to be interpolated. This is the case, when the number of real modes and oscillatory modes is not identical in all systems to be interpolated Also, it will not be reliable if eigenvalues differ significantly throughout the system matrices. In general, this primitive approach to mode tracking may only be applied to system matrices of small dimension.

3.3.2 Mode Tracking by Pole Path Observation

A more reasonable approach is to observe which poles of one system (or operating point) travel towards which poles in the next system (or operating point) if the interpolation weight for one system is gradually increased and consequently the interpolation weight of the other system is decreased. Note, that the actual path of the poles is irrelevant, only the correspondence is of interest. Hence, matrix interpolation, see section 1.4, is a well suited method due to its low computational cost. Plotting the path of the poles while varying interpolation weights enables the user to assign respective correspondence and thereby prepares the system matrices for the introduced interpolation method, see section 3.8.

This approach to mode tracking employes conventional matrix interpolation. However, only the correspondence-information is sought. The application of conventional matrix interpolation is not a prerequisite to the introduced GA interpolation method, instead, it can be used optionally to find correspondence of modes.
Fig. 3.6 shows such a plot of the path of poles. The blue crosses show the poles of the interpolated system computed with matrix interpolation. Despite intermediate systems being unstable, the correspondence between the poles of the original systems (black crosses and plus-signs) is correct. The poles computed with the method introduced in this work are depicted as red crosses. It is obvious that all intermediate systems are stable.

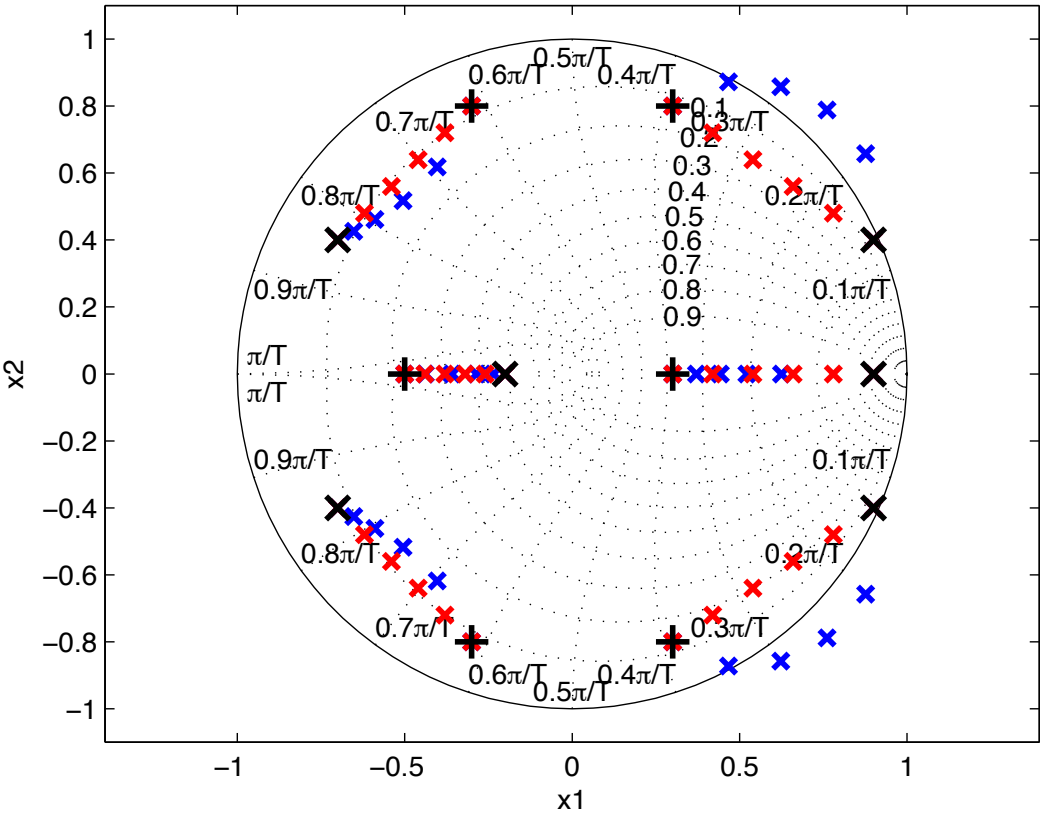


Figure 3.6: Poles of the original systems (black crosses and plus signs), the GA interpolated system (red crosses) and the poles of the systems computed via matrix interpolation (blue crosses) are depicted. The interpolation weight is defined as $\Phi_1 \in [1, 0]$. Based on such a plot mode tracking can be carried out.

3.4 The Input Matrix

The input matrix $\boldsymbol{B}$ is of dimension [n $\times$ r] where r is the number of inputs. It defines the excitation of the states. The input vector is mapped onto the state space via the rows of the

input matrix, yielding the contribution of the input to the change of the state vector $\dot{\boldsymbol{x}}$ (or $\boldsymbol{x}(k+1)$ for discrete time systems).

3.5 The Output Matrix

The output matrix $\boldsymbol{C}$ is of dimension [m × n] where m is the number of outputs. The output matrix $\boldsymbol{C}$ can be interpreted as row vectors upon which the state vector is projected, yielding the output values $\boldsymbol{y}$.

3.6 The Direct Input-Output Matrix

Since the direct input-output matrix passes signals to the output directly, it by-passes the system. Hence, a linear interpolation of state space systems with non-zero direct input-output matrices suggests that the direct input-output matrices are interpolated linearly.

3.7 Stability Considerations

Stability of a linear, time-invariant state space (LTI) system is a property encoded in the eigenvalues, i.e. the poles, of the system matrix $\boldsymbol{A}$. A continuous-time system is

stable if all eigenvalues of $\boldsymbol{A}$ are located in the left half-plane, see Fig. 3.7a

critically (or marginally) stable if all eigenvalues of $\boldsymbol{A}$ are located in the left half-plane, except one real valued eigenvalue at the origin *or* a conjugate complex pair on the imaginary axis, see Fig. 3.8a

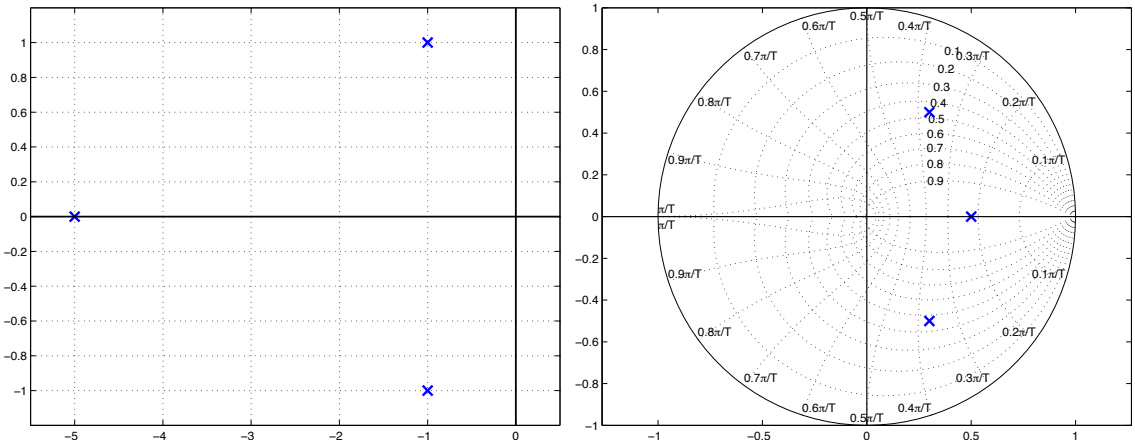
unstable if one or more eigenvalues of $\boldsymbol{A}$ are located in the right half-plane *or* eigenvalues with multiplicity > 2 are located on the imaginary axis, see Fig. 3.9a

A discrete-time system is

stable if all eigenvalues of $\boldsymbol{A}$ are located inside the unit circle, see Fig. 3.7b

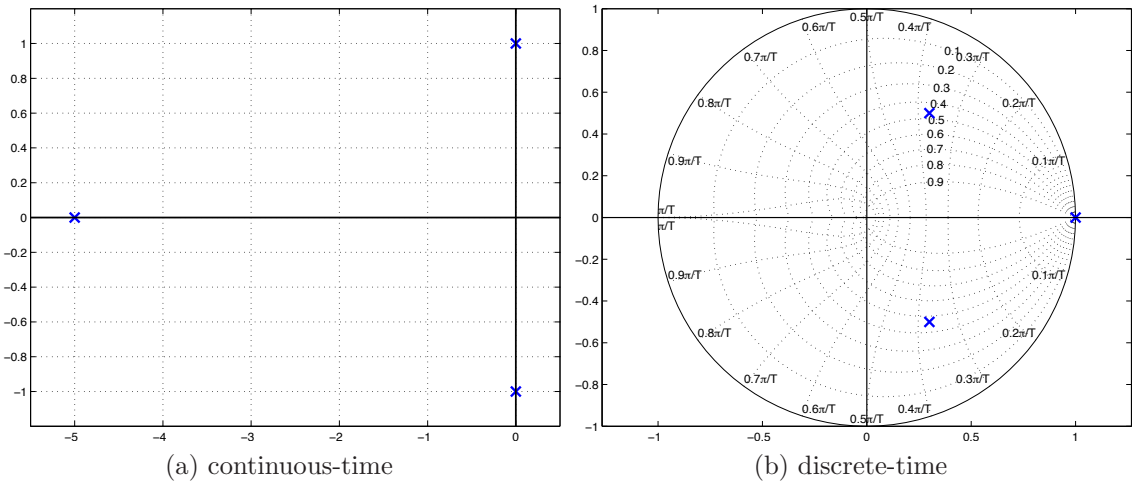
critically stable if one or more eigenvalues of $\boldsymbol{A}$ are located on the unit circle (circumfence), but no multiple pole. (A pole at distance = 1 from the origin.) No pole is located outside the unit circle. See Fig. 3.8b

unstable if one or more eigenvalues of $\boldsymbol{A}$ are located outside the unit circle *or* poles with multiplicity > 2 are located on the unit circle, see Fig. 3.9b



(a) continuous-time (b) discrete-time

Figure 3.7: Stable pole configuration.

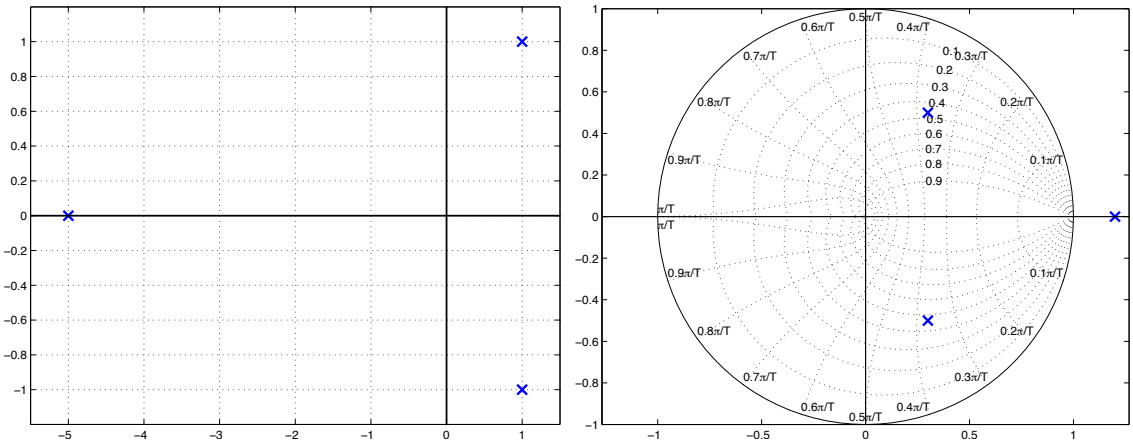


(a) continuous-time (b) discrete-time

Figure 3.8: Critically stable pole configuration.

The figures 3.7, 3.8 and 3.9 show exemplary stable, critically stable and unstable pole configurations respectively.

The presented interpolation method linearly interpolates eigenvalues and thereby *retains* (critical) stability, if all original systems are (critically) stable, in contrast to commonly used matrix coefficient interpolation, where stability of the resulting system is not guaranteed, even if all contributing systems are stable. This is due to the non-linear relation between the matrix coefficients and the eigenvalues of the matrix. Linear interpolation of matrix coefficients shifts eigenvalues in a non-linear and unobserved way, thereby making it possible, that eigenvalues "slip" out of the stable region (left half plane for continuous-time systems, unit circle for discrete-time systems).



(a) continuous-time (b) discrete-time

Figure 3.9: Unstable pole configuration.

3.8 Interpolation Method

3.8.1 Decomposition of the Original System Matrices

The first step of the interpolation method is to compute eigenvalues and eigenvectors of $\boldsymbol{A}$ matrices with (3.1). Second, the corresponding modes need to be assigned via mode tracking, see section 3.3. These two steps prepare the model for the interpolation process. Setting correct relations between the modes is vital for the result. If this step is omitted or carelessly performed, results will not reflect the true system behavior in the intended operating point. Once the modes are correctly related and the matrices are accordingly rearranged the interpolation process can be initiated.

Conjugate complex pairs and real valued eigenvalues and eigenvectors form two groups and are treated differently. The notation below assumes a system of dimension $n = 3$ so as to avoid crowding the equations with indices. However, the method is *not* restricted to three dimensions. Irrespective of the system dimension, all conjugate complex pairs are treated as denoted in section 3.8.2 and all real valued eigenvectors and eigenvalues are treated as denoted in section 3.8.2.

3.8.2 Eigenvector Interpolation

As described in section 3.2 three pure and three mixed eigenvalue constellation combinations are distinguished. Since eigenvectors and eigenvalues are strongly connected, the introduced cases must be distinguished here as well.

The three pure cases and the first two mixed cases are treated as described below. The third mixed case, where the separation point (or aperiodic oscillation point) must be “passed”

during the interpolation, can not yet be treated.

Interpolation of conjugate complex pairs

The conjugate complex pairs of eigenvectors span oscillation planes in the state space, where

$$\boldsymbol{\xi} = \text{Re}\ (\boldsymbol{u}_1) \quad \text{and} \quad \boldsymbol{\zeta} = \text{Im}\ (\boldsymbol{u}_1) \tag{3.12}$$

represent the first and the second defining 1-blade, respectively. Multiplying the two 1-blades as defined in (2.4) yields a 2-blade - the bivector defining the oscillation plane

$$\boldsymbol{\xi} \wedge \boldsymbol{\zeta} = {}_o\nu_i \cdot {}_m\nu_i \cdot {}_a\nu_i = \Pi_i, \quad \forall i \in \mathcal{I} \tag{3.13}$$

where Π_i represents the oscillation plane of a local model as a bivector. The next step is to normalize all planes, that means, each plane's magnitude ${}_m\nu_i = 1$.

$$\Pi_i \rightarrow {}_m\Pi_i = 1 \tag{3.14}$$

If this step is omitted, ${}_m\nu$ represents an additional "interpolation weight" and thereby interferes with the intended interpolation weights Φ_i.
The interpolation itself is defined by the following equation:

$$\Pi_{int} = ({}_o\nu \cdot 1 \cdot {}_a\nu) = \sum_{\mathcal{I}} \Phi_i \Pi_i \tag{3.15}$$

where the index *int* denotes the interpolated system. This seemingly simple equation not only correctly interpolates the respective attitudes in space but also takes each planes orientation into account.
Now that the interpolated oscillation plane is known a set of new eigenvectors must be extracted, that define exactly this bivector with ${}_o\nu_{int}$ and ${}_a\nu_{int}$. Note here, that two arbitrary vectors in that plane that yield a bivector of magnitude one may define the interpolated plane as a bivector completely. A particular set of two bivectors must be extracted from an infinite set of possible combinations of vectors.

Interpolation of Relative Attitude

From a modeling point of view the relative position of eigenvectors that form a plane affects the resulting system behavior. Or, putting it the other way round, the system behavior defines the relative position of eigenvectors. It is hence important to extract eigenvectors that fulfill the condition of interpolating the relative position (attitude). The set of eigenvectors

of each local model is gathered to define a coordinate-triple.

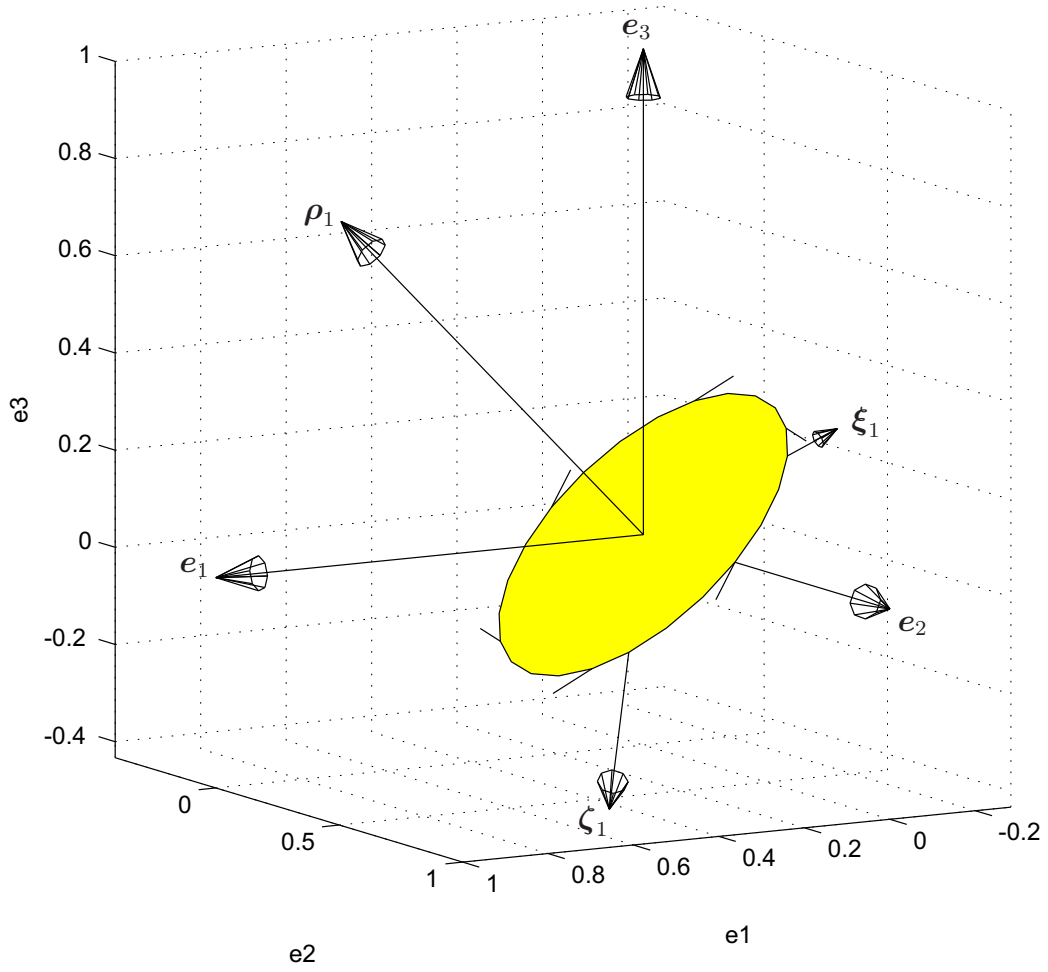


Figure 3.10: A local triple of eigenvectors and the oscillation plane spanned by the conjugate complex pair.

Fig. 3.10 shows the local triple of eigenvectors in the general orthogonal coordinate system $\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3$.

Consequently, the idea is to rotate each of the contributing systems' eigenvector-triple into the interpolated plane, so that the two conjugate complex eigenvectors lie in the interpolated plane. Their relative attitudes can now be interpolated, knowing that the resulting vector will be in the interpolated plane, too.

In analogy to (3.14) the vectors magnitude is set to 1 per definition, so as not to interfere with the intended interpolation weights.

$$\{\boldsymbol{\xi}_i, \boldsymbol{\zeta}_i, \boldsymbol{\rho}_i\} \rightarrow \{{}_m\boldsymbol{\xi}_i = 1,\ {}_m\boldsymbol{\zeta}_i = 1,\ {}_m\boldsymbol{\rho}_i = 1\} \tag{3.16}$$

The real valued eigenvector is denoted by $\boldsymbol{\rho} = \boldsymbol{u}_3$. The directions of the eigenvectors are

geometrically interpolated as defined by

$$\{\boldsymbol{\xi}_{int}, \boldsymbol{\zeta}_{int}\} = (+1 \cdot {}_a\mu) = \sum_{\mathcal{I}} \Phi_i \{\boldsymbol{\xi}_i, \boldsymbol{\zeta}_i\} \tag{3.17}$$

Note here, that only the directions of the eigenvectors are interpolated.

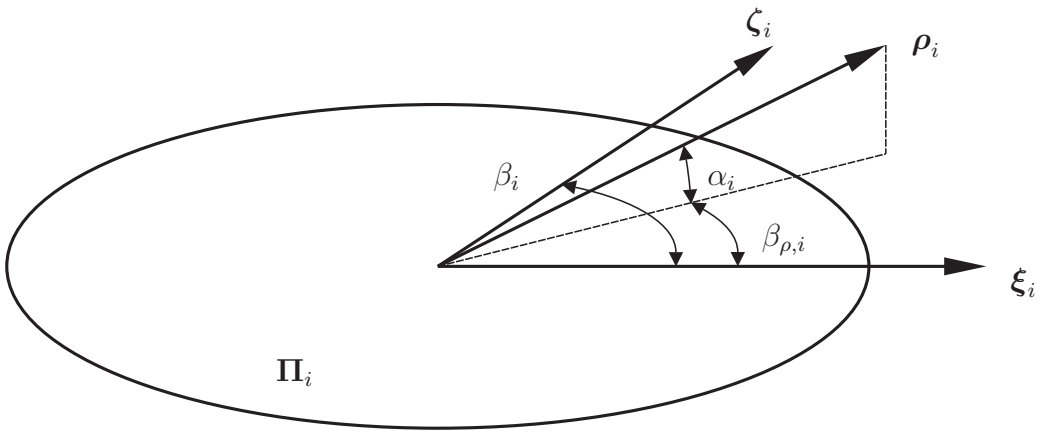


Figure 3.11: The relative position of the real valued eigenvector with respect to its conjugate complex pair

The real valued eigenvector $\boldsymbol{\rho}$ is treated separately. To correctly interpolate the system dynamics it is vital to interpolate its relative position to $\boldsymbol{\xi}$ and $\boldsymbol{\zeta}$. In Fig. 3.11 a general configuration is depicted. The angle β_i denotes the angle between $\boldsymbol{\xi}_i$ and $\boldsymbol{\zeta}_i$. The angle $\beta_{\rho,i}$ denotes the angle between $\boldsymbol{\xi}_i$ and the projection of $\boldsymbol{\rho}_i$ onto $\boldsymbol{\Pi}_i$. The angle α_i denotes the angle of ρ_i above the plane $\boldsymbol{\Pi}_i$. All three angles are interpolated as defined by

$$\beta_{int} = \sum_{\mathcal{I}} \Phi_i \, \beta_i \tag{3.18}$$

$$\alpha_{int} = \sum_{\mathcal{I}} \Phi_i \, \alpha i \tag{3.19}$$

$$\frac{\beta_{int}}{\beta_{\rho,int}} = \sum_{\mathcal{I}} \Phi_i \, \frac{\beta_i}{\beta_{\rho,i}} \tag{3.20}$$

Then $\boldsymbol{\rho}$ is a unit-vector for which $\beta = \beta_{int}$, $\alpha = \alpha_{int}$ and $\beta_\rho = \beta_{\rho,int}$ with respect to $\boldsymbol{\xi}_{int}$ and $\boldsymbol{\zeta}_{int}$.
Since all vectors are interpolated as unit elements with ${}_m\mu = 1$ the resulting vector is also of magnitude 1. The final magnitude is interpolated in a second step, using the original magnitudes of all eigenvectors:

$${}_m\mu_{n,int} = \sum_{\mathcal{I}} \Phi_i \, {}_m\mu_{i,n} \tag{3.21}$$

The $n = 3$ resulting new vectors are assembled as defined by

$$\{\boldsymbol{\xi}_{int}, \boldsymbol{\zeta}_{int}, \boldsymbol{\rho}_{int}\} = \{ {}_m\mu_{n,int} \cdot {}_a\mu_{n,int}\} \tag{3.22}$$

using the results of (3.17) and (3.21). The eigenvectors are reassembled using (3.12)

$$\boldsymbol{u}_{1,int} = \boldsymbol{\xi}_{int} - \boldsymbol{\zeta}_{int} i \tag{3.23}$$

$$\boldsymbol{u}_{2,int} = \boldsymbol{\xi}_{int} + \boldsymbol{\zeta}_{int} i \tag{3.24}$$

$$\boldsymbol{u}_{3,int} = \boldsymbol{\rho}_{int} \tag{3.25}$$

3.8.3 Eigenvalue Interpolation

As described in section 3.2 three pure and three mixed eigenvalue constellation combinations are distinguished. For the three pure cases the eigenvalues λ_n are linearly interpolated.

$$\lambda_{n,int} = \sum_{\mathcal{I}} \Phi_i \lambda_{i,n} \tag{3.26}$$

Fig. 3.4 illustrates the equation. The first mixed case, interpolation between two different real valued eigenvalues and one real valued eigenvalue with multiplicity = 2, and second mixed case, interpolation between a conjugate complex pair and one real valued eigenvalue with multiplicity = 2 are treated identical to the pure cases. In these two cases the eigenvalues are interpolated with (3.26).
Interpolation of the third mixed case is not part of this work.
The matrix of eigenvectors $\boldsymbol{U}_{n,int}$ and the eigenvalue matrix $\boldsymbol{U}_{n,int}$ are set up.

$$\boldsymbol{U}_{int} = \begin{bmatrix} \boldsymbol{u}_{1,int} & \boldsymbol{u}_{2,int} & \boldsymbol{u}_{3,int} \end{bmatrix}, \ \boldsymbol{U}_{int} \in \mathbb{R}^{3\times3} \tag{3.27}$$

$$\boldsymbol{\Lambda}_{int} = \begin{bmatrix} \lambda_{1,int} & 0 & 0 \\ 0 & \lambda_{2,int} & 0 \\ 0 & 0 & \lambda_{3,int} \end{bmatrix}, \ \boldsymbol{\Lambda}_{int} \in \mathbb{R}^{3\times3} \tag{3.28}$$

3.8.4 Assembling of the Interpolated System Matrix

The interpolated $\boldsymbol{A}$ matrix is reassembled using the eigendecomposition [18].

$$\boldsymbol{A}_{int} = \boldsymbol{U}_{int} \Lambda_{int} \boldsymbol{U}_{int}^{-1}, \ \boldsymbol{A}_{int} \in \mathbb{R}^{3\times3} \tag{3.29}$$

When reassembling the new system matrix it is vital to sort the eigenvector-eigenvalue-sets of $\boldsymbol{A}_i$ so that the corresponding real valued entities are in line and the conjugate complex pairs are in line. In this work, the proposed (but arbitrary) manner of sorting is: [complex,

complex, real]. This helps to avoid confusion and makes it impossible to accidentally relate a real valued eigenvalue to a conjugate complex coupled eigenvector and vice versa.

3.8.5 Interpolation of Input and Output Matrices

The interpolation of input and output matrices is not a core task of this work. A number of new approaches have been investigated, but none has delivered entirely convincing results. In addition all investigated problems in this work are flat systems, which can be seen as a restriction for the applicability of the method. Since flatness depends on the input and output matrices of a system as well as the system matrix itself the interpolation of all three matrices is critical. It remains to be clarified how a *correct* interpolation should be defined. Further research is required on this part.
An interpolation approach that allows for the application of defined conditions concerning the state vector and the output signal is applied here. The $\boldsymbol{B}_i$ and $\boldsymbol{C}_i$ matrices are interpolated linearly

$$\boldsymbol{B}_{int} = \sum_{\mathcal{I}} \Phi_i \boldsymbol{B}_i \tag{3.30}$$

$$\boldsymbol{C}_{int} = \sum_{\mathcal{I}} \Phi_i \boldsymbol{C}_i \tag{3.31}$$

thereby retaining the general structure of the matrices in case a special structure was given in the first place. The interpolated system is defined by the matrices $\boldsymbol{A}_{int}$, $\boldsymbol{B}_{int}$ and $\boldsymbol{C}_{int}$. Note that it is possible to generate a $\boldsymbol{B}_{int}$ (or a $\boldsymbol{C}_{int}$) matrix such that the stationary value of $\boldsymbol{x}_i$ is linearly interpolated. Unfortunately, when altering those matrices, the system dynamics are altered or the output is scaled differently and thereby step response-plots are skewed.

3.8.6 Tweaking of Input- and Output Matrices

Computation of input- and output matrices other than stated in section 3.8.5 is not implemented in the method due to the disadvantages introduced there. However, a number of approaches have been investigated and may be of interest for special applications.

Requirements on $\boldsymbol{B}$ for linear interpolation of steady state value

To interpolate the steady state value of the *state vector* linearly, a condition for the input matrix $\boldsymbol{B}$ can be derived:

$$0 = \dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}$$

$$\boldsymbol{x}_{i,\infty} = -\boldsymbol{A}_i^{-1}\boldsymbol{B}_i\boldsymbol{u}$$

and

$$\boldsymbol{x}_{int} = -\boldsymbol{A}_{int}^{-1}\boldsymbol{B}_{int}\boldsymbol{u} = \sum_{\mathcal{I}} \Phi_i[-\boldsymbol{A}_i^{-1}\boldsymbol{B}_i\boldsymbol{u}] \tag{3.32}$$

With (3.32) a condition for $\boldsymbol{B}_{int}$ follows

$$\boldsymbol{B}_{int} = \boldsymbol{A}_{int}\left[\sum_{\mathcal{I}} \Phi_i \boldsymbol{A}_i^{-1}\boldsymbol{B}_i\right] \tag{3.33}$$

Analogous for discrete-time systems

$$\boldsymbol{B}_{int} = (\boldsymbol{I} - \boldsymbol{A}_{int})\left[\sum_{\mathcal{I}} \Phi_i (\boldsymbol{I} - A_i)^{-1}\boldsymbol{B}_i\right] \tag{3.34}$$

where $\boldsymbol{I}$ is the identity matrix. The interpolated system matrix $\boldsymbol{A}_{int}$ is computed by the method introduced above in section 3.8. The only unknown $\boldsymbol{B}_{int}$ is fully defined by (3.33) and (3.34) respectively.
Judging output signals is hard insofar as the output matrix $\boldsymbol{C}$ greatly affects the result.

Requirements on $\boldsymbol{C}$ for linear interpolation of the output vector $\boldsymbol{y}$

It is possible to generate an output matrix $\boldsymbol{C}$ such that the output vectors are interpolated linearly, irrespective of the interpolation method applied on the system matrix $\boldsymbol{A}$ and input matrix $\boldsymbol{B}$. The condition demands the matrix dimensions to fit the resulting equation, hence, it is limited to square input matrices of dimension $n \times n$. Also, numerical difficulties may arise with matrix inversion and the conditioning (or even singularity) of the respective inverse. The defining equation, assuming that $\boldsymbol{x}$ is a steady state value, is derived:

$$\boldsymbol{y}_i = \boldsymbol{C}_i^T\boldsymbol{x}_i = \boldsymbol{C}_i^T(-\boldsymbol{A}_i^{-1}\boldsymbol{B}_i\boldsymbol{u}) \tag{3.35}$$

and with (3.32)

$$\boldsymbol{y}_{int} = \boldsymbol{C}_{int}^T\boldsymbol{x}_{int} = -\boldsymbol{C}_{int}^T\boldsymbol{A}_{int}^{-1}\boldsymbol{B}_{int}\boldsymbol{u} \tag{3.36}$$

By requiring that

$$\boldsymbol{y}_{int} = \boldsymbol{C}_{int}^T\boldsymbol{x}_{int} = \sum_{\mathcal{I}} \Phi_i[-\boldsymbol{C}_i^T\boldsymbol{A}_i^{-1}\boldsymbol{B}_i\boldsymbol{u}] \tag{3.37}$$

holds, $\boldsymbol{C}_{int}$ is found as

$$\boldsymbol{C}_{int}^T = \left[\sum_{\mathcal{I}} \Phi_i[-\boldsymbol{C}_i^T\boldsymbol{A}_i^{-1}\boldsymbol{B}_i\boldsymbol{u}]\right]\left(-\boldsymbol{A}_{int}\boldsymbol{B}_{int}^{-1}\right) \tag{3.38}$$

Analogous for discrete-time systems

$$\boldsymbol{C}_{int}^T = \left[\sum_{\mathcal{I}} \Phi_i[-\boldsymbol{C}_i^T(\boldsymbol{I}-\boldsymbol{A}_i)^{-1}\boldsymbol{B}_i\boldsymbol{u}]\right]\left((\boldsymbol{I}-\boldsymbol{A}_{int})^{-1}\boldsymbol{B}_{int}\right)^{-1} \tag{3.39}$$

For SISO Systems equations (3.38) and (3.39) can be simplified and solved. The idea is to define an arbitrary input signal, $\boldsymbol{u} = 1$ for example, and compute $\boldsymbol{B}_{int}$ so that the above conditions defined by (3.33) and (3.34) respectively, are met. Thereby the excitation of the states by the input signal is linearly interpolated.
The output signals for each original system must be computed with (3.35) and interpolated.

$$\boldsymbol{y}_{int} = \sum_{\mathcal{I}} \Phi_i \boldsymbol{y}_i \tag{3.40}$$

Then, the preliminary interpolated output signal is computed with a preliminary output matrix $\boldsymbol{C}_{prel}^T$ as defined by (3.31).

$$\boldsymbol{y}_{prel} = \sum_{\mathcal{I}} \Phi_i[-\boldsymbol{C}_{prel}^T\boldsymbol{A}_i^{-1}\boldsymbol{B}_i\boldsymbol{u}] \tag{3.41}$$

For SISO systems (3.40) and (3.41) yield scalar values. Then the following relation holds,

$$y_{int} = \alpha_{\text{corr}} y_{prel} \tag{3.42}$$

where α_{corr} is a scalar factor. With (3.40) and (3.41) equation (3.42) can be evaluated and yields α_{corr}.
Consequently, the *correct* $\boldsymbol{C}_{int}^T$ is defined by

$$\boldsymbol{C}_{int}^T = \alpha_{\text{corr}}\boldsymbol{C}_{prel}^T \tag{3.43}$$

This implies, that in order to meet the output value condition defined by equation (3.37), the steady state value condition defined by equation (3.32) on the input matrix must be fulfilled too. Note that the simplification can only be applied to SISO systems.

Chapter 4

Results and Validation of Concept

The GA (Geometric Algebra) interpolation method is applied to two examples so as to underline its advantages and to illustrate the procedure. Conventional matrix interpolation, see section 1.4 serves to compare and judge the results.

4.1 Orthogonal Planes

The first demonstrative example is an interpolation of two continuous-time systems with orthogonal oscillation planes defined by (1.2) and

$$\boldsymbol{A}_1 = \begin{bmatrix} 0 & 1 & 0 \\ -b1 & -a1 & 0 \\ 0 & 0 & -3 \end{bmatrix}, \boldsymbol{A}_2 = \begin{bmatrix} -3 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -b2 & -a2 \end{bmatrix},$$

$$\boldsymbol{B}_1 = \boldsymbol{B}_2 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \ \boldsymbol{c}_1^T = \boldsymbol{c}_2^T = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$$

$$\Phi_1 = \Phi_2 = 0.5, \quad a1 = a2 = 0.1, \quad b1 = b2 = 2$$

The advantage of using this particular structure is, that damping ratio and oscillation frequency are directly set by the factors $a1$, $a2$ and $b1$, $b2$ respectively. The influence of these numbers on the system behavior is outlined in example 3.1. Furthermore, interpolating orthogonal planes at $\Phi_1 = \Phi_2 = 0.5$ results in a large angular difference between the given and the computed oscillation planes. The input and output matrices are identical so that their influence on the result is as small as possible. Thereby, the introduced interpolation method for the system matrix is accentuated. The example serves to show that the proposed GA interpolation method performs well on problems, where the common matrix interpolation performs poorly.

Table 4.1: The bivector specifications of oscillation planes

	Π_1	Π_2	Π_{int}
${}_o\nu$	-	-	-
${}_m\nu$	0.4711	0.4711	1
${}_a\nu$	$\boldsymbol{e_1 \wedge e_2}$	$\boldsymbol{e_2 \wedge e_3}$	$\boldsymbol{e_1 \wedge e_2 + e_2 \wedge e_3}$

The eigenvectors and eigenvalues of both systems are

$$\boldsymbol{U}_1 = \begin{bmatrix} -0.0204 - 0.5770i & -0.0204 + 0.5770i & 0 \\ 0.8165 & 0.8165 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

$$\boldsymbol{U}_2 = \begin{bmatrix} 0 & 0 & 1 \\ -0.0204 - 0.5770i & -0.0204 + 0.5770i & 0 \\ -0.8165 & -0.8165 & 0 \end{bmatrix},$$

$$\boldsymbol{\Lambda}_1 = \boldsymbol{\Lambda}_2 = \begin{bmatrix} -0.05 + 1.4133i & 0 & 0 \\ 0 & -0.05 - 1.4133i & 0 \\ 0 & 0 & -3.0 \end{bmatrix}$$

The coordinate triples $\boldsymbol{\xi}, \boldsymbol{\zeta}$ and $\boldsymbol{\rho}$ defined by their respective eigenvectors are

$$\boldsymbol{\xi}_1 = \begin{bmatrix} -0.0204 \\ 0.8165 \\ 0 \end{bmatrix}, \boldsymbol{\zeta}_1 = \begin{bmatrix} 0.5770 \\ 0 \\ 0 \end{bmatrix}, \boldsymbol{\rho}_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}.$$

and

$$\boldsymbol{\xi}_2 = \begin{bmatrix} 0 \\ -0.0204 \\ -0.8660 \end{bmatrix}, \boldsymbol{\zeta}_2 = \begin{bmatrix} 0 \\ 0.5770 \\ 0 \end{bmatrix}, \boldsymbol{\rho}_2 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}.$$

where $\boldsymbol{\rho}$ denotes the eigenvector associated to the real eigenvalue.

Fig. 4.1 shows both oscillation planes and the real valued eigenvectors of both systems. Tab. 4.1 summarizes the systems 2-blades data. Note that, because the magnitudes of $\boldsymbol{\xi}_i < 1$ and the magnitudes of $\boldsymbol{\zeta}_i < 1$ the magnitudes ${}_m\Pi_i < 1$, too. Before interpolating the bivectors, their magnitude is set to 1, as in (3.14). Hence, ${}_m\Pi_{int} = 1$.

In Fig. 4.2 both oscillation planes with their original magnitude and the interpolated plane are depicted. Note here, that the attitude of the interpolated plane is defined *because* the

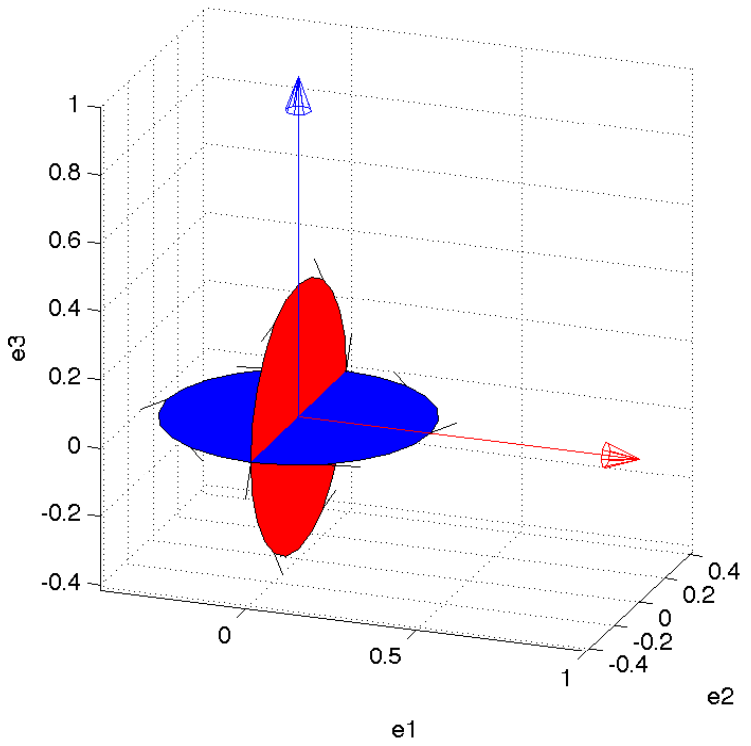


Figure 4.1: The oscillation planes and the real valued eigenvector of system one (blue) and system two (red).

orientation information is taken into account when using GA. Otherwise there would have been two possible solutions, each tilted by 45° between the two given planes.
Fig. 4.3 shows the interpolated plane and the two coordinate triples as well as the interpolated triple as a result of the method described in 3.8.2. The interpolated triple is an orthogonal triple because both given triples were orthogonal.
Hence, the interpolated coordinate triple is

$$\boldsymbol{\xi}_{int} = \begin{bmatrix} -0.4185 \\ 0.5629 \\ 0.4185 \end{bmatrix}, \boldsymbol{\zeta}_{int} = \begin{bmatrix} 0.2885 \\ 0.4080 \\ -0.2885 \end{bmatrix}, \boldsymbol{\rho}_{int} = \begin{bmatrix} 0.7071 \\ 0 \\ 0.7071 \end{bmatrix}.$$

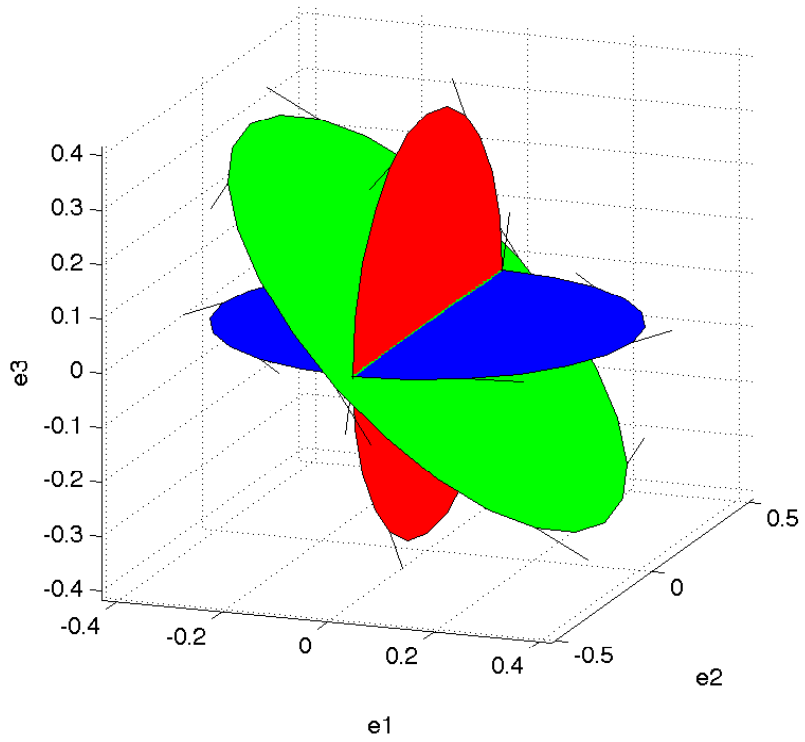


Figure 4.2: The oscillation planes and the interpolated plane.

The eigenvectors are assembled with (3.12) and $\mathbf{\Lambda}_1 = \mathbf{\Lambda}_2 = \mathbf{\Lambda}_{int}$ so that

$$\boldsymbol{U}_{int} = \begin{bmatrix} 0.4185 - 0.2885i & 0.4185 + 0.2885i & 0.7071 \\ 0.5629 - 0.4080i & 0.5629 + 0.4080i & 0 \\ 0.4185 + 0.2885i & 0.4185 - 0.2885i & 0.7071 \end{bmatrix},$$

$$\mathbf{\Lambda}_{int} = \begin{bmatrix} -0.05 + 1.4133i & 0 & 0 \\ 0 & -0.05 - 1.4133i & 0 \\ 0 & 0 & -3.0 \end{bmatrix}$$

Finally, the $\boldsymbol{A}$ matrix is assembled according to (3.29)

$$\boldsymbol{A}_{int} = \begin{bmatrix} -1.2750 & 1.0960 & -1.7250 \\ -1.0253 & -0.5500 & 1.0253 \\ -1.7250 & -1.0960 & -1.2750 \end{bmatrix} \tag{4.1}$$

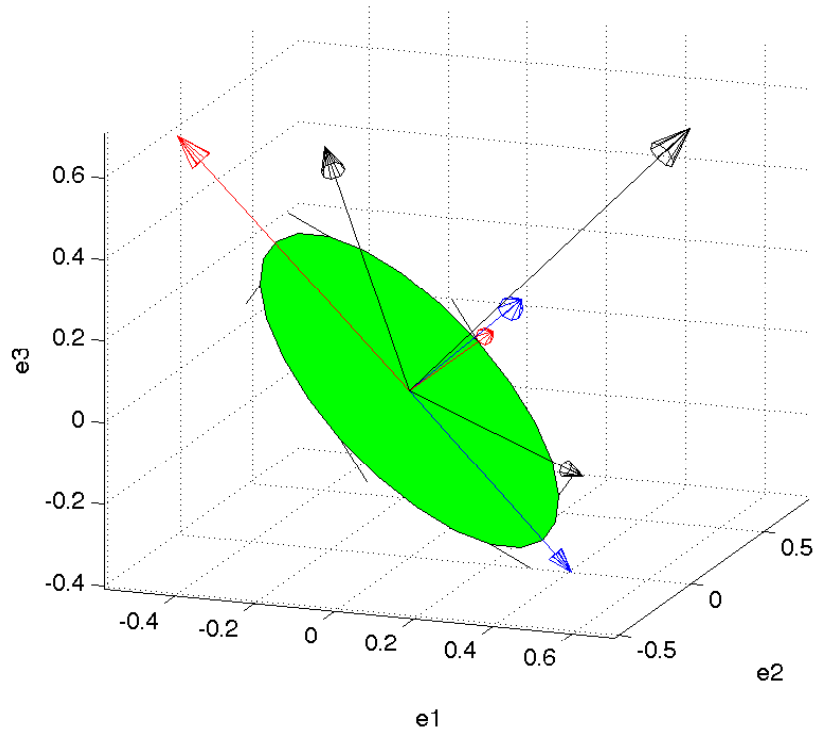


Figure 4.3: The rotated coordinate triples of both systems in the interpolated plane and the interpolated coordinate triple (black).

For comparison, the $\boldsymbol{A}$ matrix computed by matrix interpolation is found as

$$\boldsymbol{A}_{int,\,\text{matrix}} = \begin{bmatrix} -1.5000 & 0.5000 & 0 \\ -1.0000 & -0.0500 & 0.5000 \\ 0 & -1.0000 & -1.5500 \end{bmatrix} \tag{4.2}$$

Fig. 4.4 depicts the state vectors trajectories starting at the initial condition $x_0 = [1, 1, 1]$. The oscillation planes correspond to the bivector representation in Fig. 4.2. Note that both the orientation and the damping behavior are correctly interpolated with the GA interpolation method. In contrast, the matrix interpolation results in a non-oscillating system whose trajectory reaches the origin almost directly.

Fig. 4.5 depicts the location of the poles in the s-plane (the eigenvalues of the system matrices). Since the eigenvalues of both original systems are identical (black, large cross), the introduced GA interpolation leaves them unchanged. Thereby the system dynamics are interpolated correctly. In contrast, the matrix interpolation alters, depending on Φ_i, the location of the interpolated system's poles (blue and red, smaller cross). In this particular case, the real valued pole (blue crosses) travels towards the origin, making the system slower

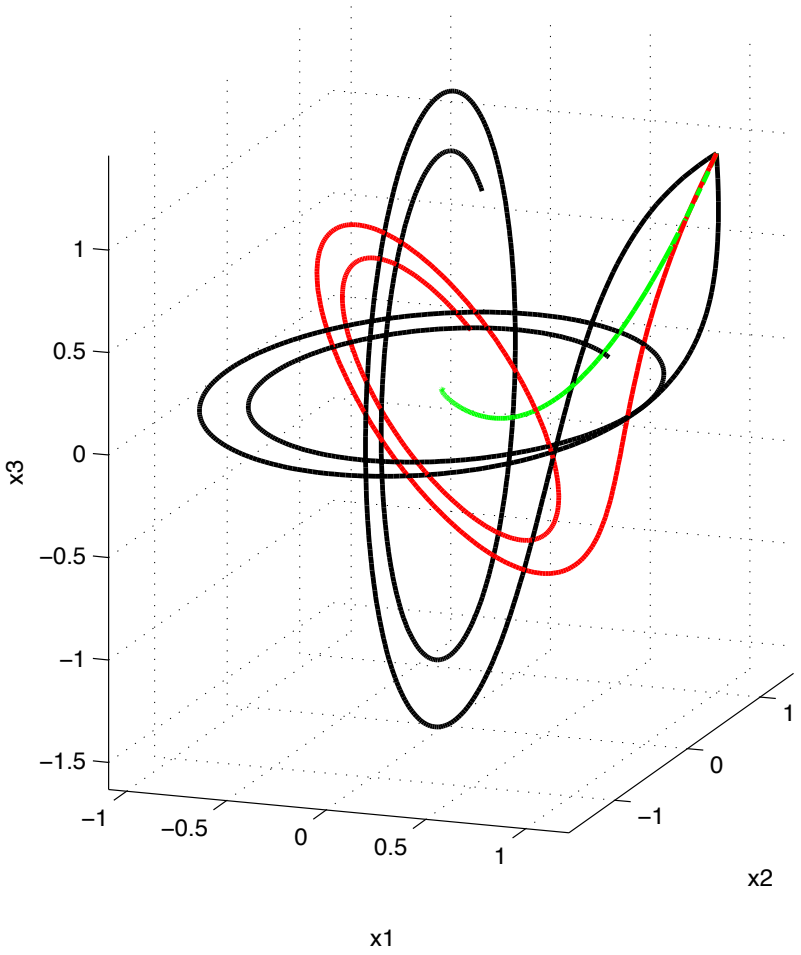


Figure 4.4: The trajectories of the state vectors of Systems 1 and 2 (solid, black), the GA interpolated system (solid, red, tilted) and the reference system interpolated by conventional matrix interpolation (solid, green) with common initial conditions $x_0 = [1, 1, 1]$.

and the conjugate complex pair travels from the original low damping ratio of $\zeta \approx 0.05$ at $\Phi_1 = 1$ to a maximum of $\zeta \approx 0.76$ at $\Phi_{1,2} = 0.5$.

Fig. 4.6 shows the step responses of the two given systems and the interpolated systems as well as step response of system computed with common matrix interpolation.

Fig. 4.7 shows the step responses of each state separately. Note here, only the second state oscillates in both given systems. The GA interpolation provides an accurate interpolation, where both the frequency and damping ratio are reasonable. Even the states where one system oscillates while the other system does not are intuitively correctly interpolated. Standard matrix interpolation results in a non-oscillating system.

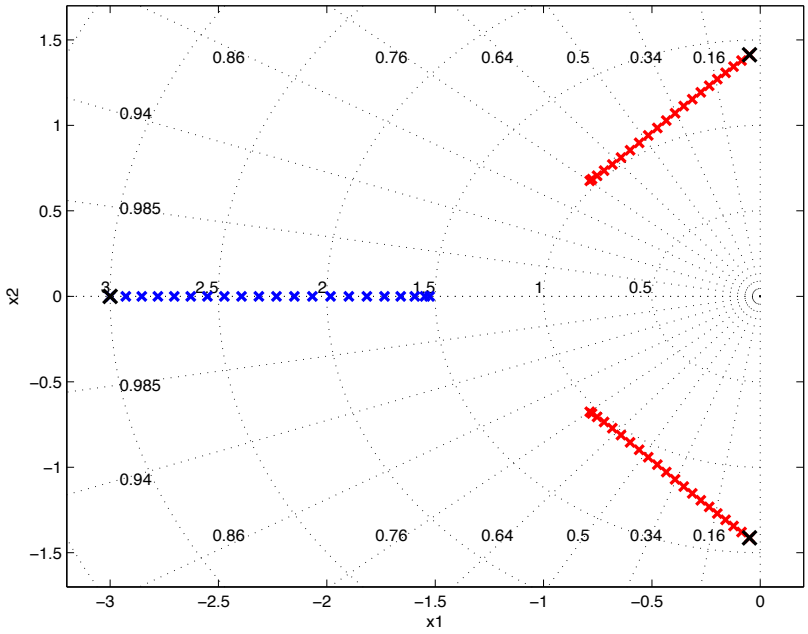


Figure 4.5: Poles of the original systems and the GA interpolated system (black, large cross, identical). The poles of the systems computed via matrix interpolation (blue and red, smaller cross) at varying interpolation weights $\Phi_1 \in [1, 0.5]$ change position in the s-plane.

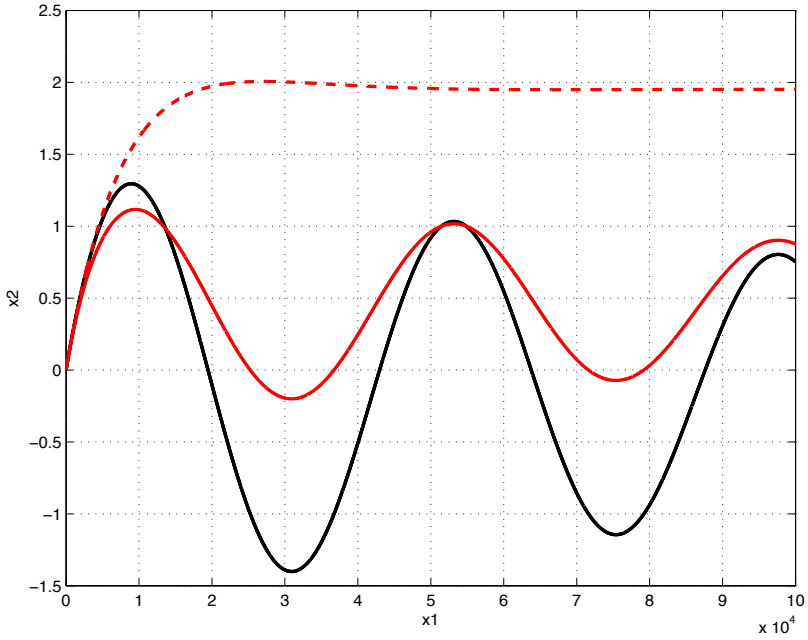


Figure 4.6: The step responses of Systems 1 and 2 (solid, black, identical), the GA interpolated system (solid, red) and the reference system interpolated by matrix interpolation (dashed, red).

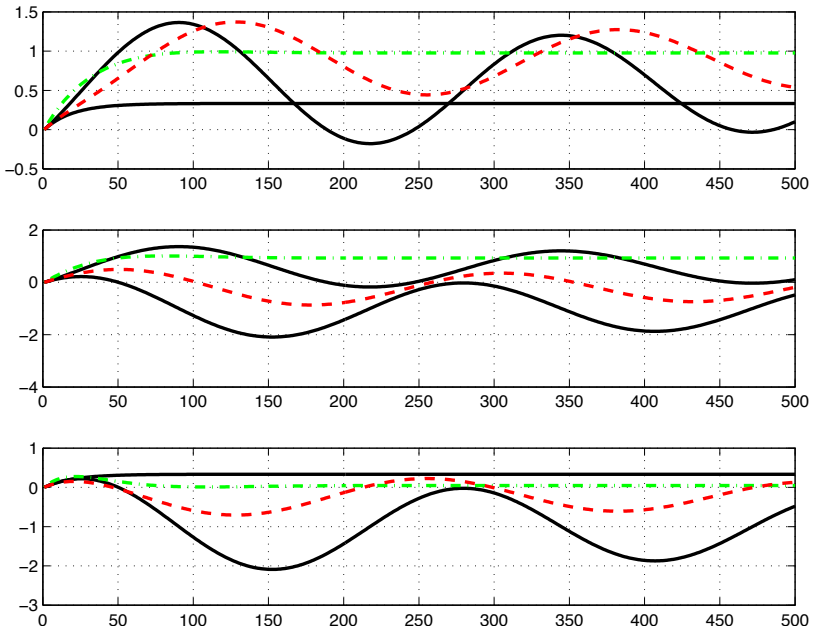


Figure 4.7: The step responses of Systems 1 and 2 (solid, black), the GA interpolated system (dashed, red) and the reference system interpolated matrix interpolation (dash-dotted, green) for each state.

4.1.1 Influence of Input and Output Matrices

Active steady state value condition

The application of the derived conditions for input and output matrices in section 3.8.5 is connected to the trade-off between the interpolation of static and dynamic characteristics. Fig. 4.8 depicts the step response for each state of a system with active steady state value condition. That means, the Input matrix is computed as defined by (3.33). The output matrix is linearly interpolated.
The new $\boldsymbol{B}$ matrix computes to

$$\boldsymbol{B}_{int,\text{ enforced}} = \begin{bmatrix} 0.2347 \\ 0.6709 \\ 0.0903 \end{bmatrix} \tag{4.3}$$

It is obvious that the steady state values for each state are interpolated linearly. At the

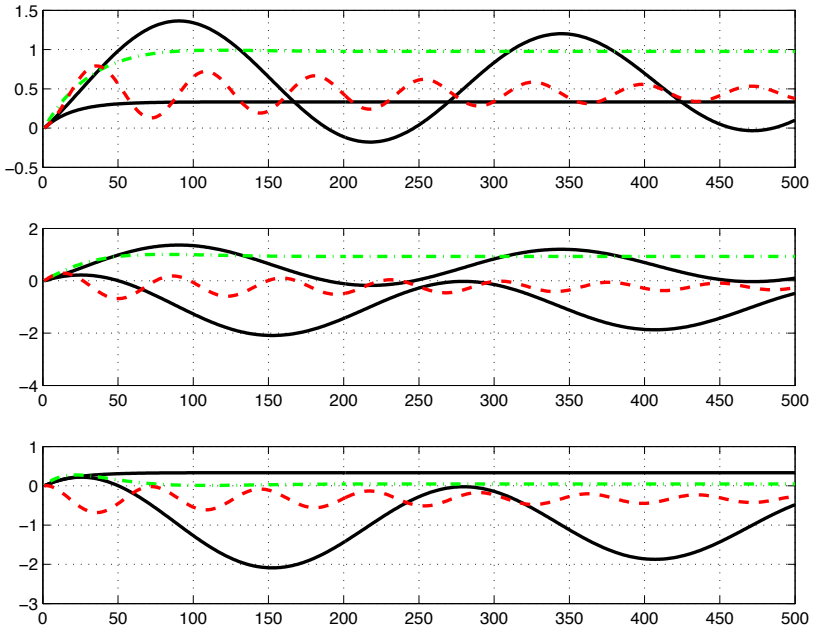


Figure 4.8: The step responses of Systems 1 and 2 (solid, black), the GA interpolated system (dashed, red) and the reference system interpolated by matrix interpolation (dash-dotted, green) for each state. Active steady state value condition.

same time the dynamic characteristics of the step response have changed dramatically. The oscillation frequency seems to have increased.

Influence of the output matrix

Due to the fact that these systems are not SISO systems the condition derived for the output matrix can not be applied. The equations can not be solved.

4.1.2 Variation: Different Damping Ratios

The system matrices are of the same structure as above. The damping ratio of system two is increased as given by the coefficients: $a1 = 0.1,\ a2 = 0.4$ and $b1 = b2 = 2$. Hence, the defining matrices are

$$\boldsymbol{A}_{1,\text{ damping}} = \begin{bmatrix} 0 & 1 & 0 \\ -2 & -0.1 & 0 \\ 0 & 0 & -3 \end{bmatrix},\ \boldsymbol{A}_{2,\text{ damping}} = \begin{bmatrix} -3 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -2 & -0.4 \end{bmatrix},$$

$$\boldsymbol{B}_{1,\text{ damping}} = \boldsymbol{B}_{2,\text{ damping}} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix},$$

$$\boldsymbol{c}^T_{1,\text{ damping}} = \boldsymbol{c}^T_{2,\text{ damping}} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$$

$$\Phi_1 = \Phi_2 = 0.5,$$

The interpolation algorithm is run with the altered matrices and yields the following results. The new $\boldsymbol{A}$ matrix is assembled according to (3.29)

$$\boldsymbol{A}_{int,\text{ damping}} = \begin{bmatrix} -1.3113 & 1.1482 & -1.6887 \\ -0.9716 & -0.6275 & 0.9716 \\ -1.6887 & -1.1482 & -1.3113 \end{bmatrix} \tag{4.4}$$

Note the differences compared to the $\boldsymbol{A}$ matrix given by (4.1).
For comparison, the $\boldsymbol{A}$ matrix computed by conventional matrix interpolation is found as

$$\boldsymbol{A}_{int,\text{ matrix}} = \begin{bmatrix} -1.5000 & 0.5000 & 0 \\ -1.0000 & -0.0500 & 0.5000 \\ 0 & -1.0000 & -1.7000 \end{bmatrix} \tag{4.5}$$

Fig. 4.9 depicts the state vectors trajectories starting at the initial condition $x_0 = [1, 1, 1]$ as in the above example. Again, both the orientation and the damping behavior are correctly interpolated with the GA interpolation method. In contrast, the matrix interpolation results in a non-oscillating system whose trajectory reaches the origin almost directly.
Fig. 4.10 shows the step responses of the two given systems and the interpolated systems as

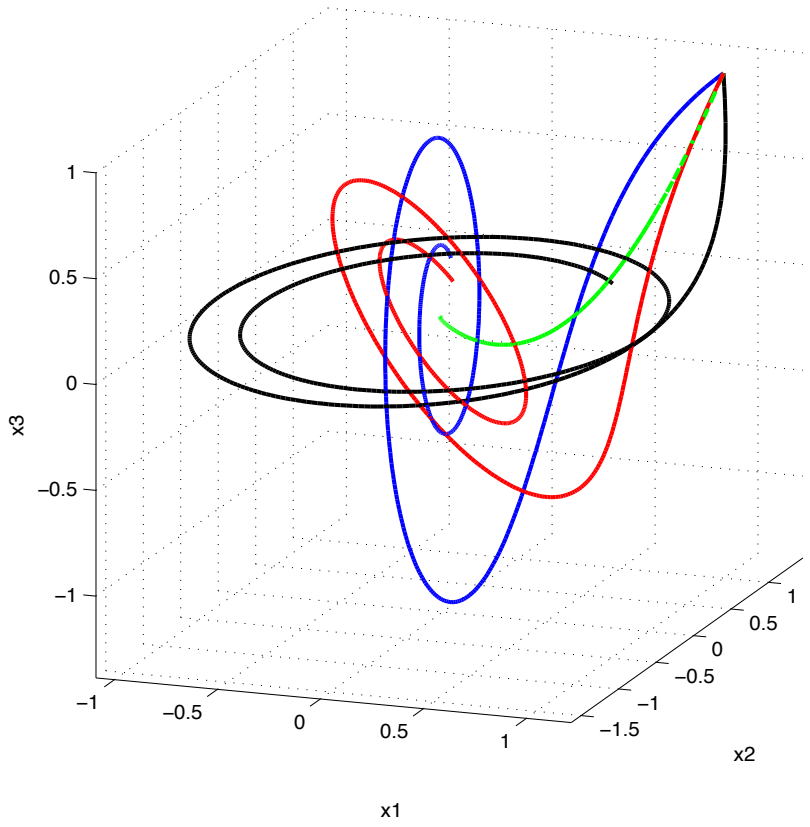


Figure 4.9: The trajectories of the state vectors of Systems 1 (solid, black) and 2 (solid, blue) with different damping ratios, the GA interpolated system (solid, red, tilted) and the reference system interpolated by conventional matrix interpolation (solid, green) with common initial conditions $x_0 = [1, 1, 1]$.

well as the step response of the system computed with common matrix interpolation. Note that the damping behavior is correctly interpolated. At the same time the natural frequency is correctly interpolated as well: since the natural frequency of the two original systems is identical, the natural frequency of the interpolated system is identical to the original natural frequency, too.

Fig. 4.11 shows the step responses of each state separately. Note here, only the second state oscillates in both given systems. The GA interpolation provides an accurate interpolation, where both the frequency and damping ratio are reasonable. Even the states where one system oscillates while the other system does not are intuitively correctly interpolated.

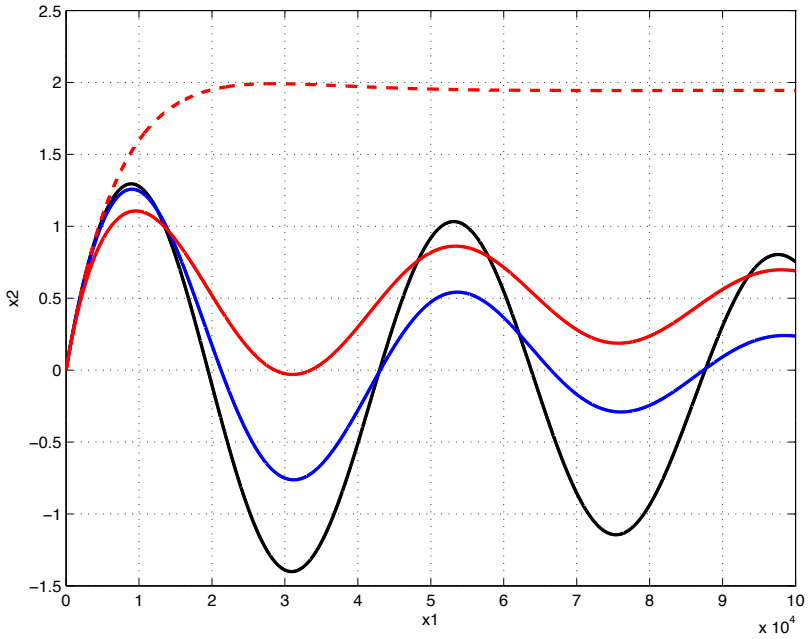


Figure 4.10: The step responses of Systems 1 (solid, black) and 2 (solid, blue) with different damping ratios, the GA interpolated system (solid, red) and the reference system interpolated by matrix interpolation (dashed, red).

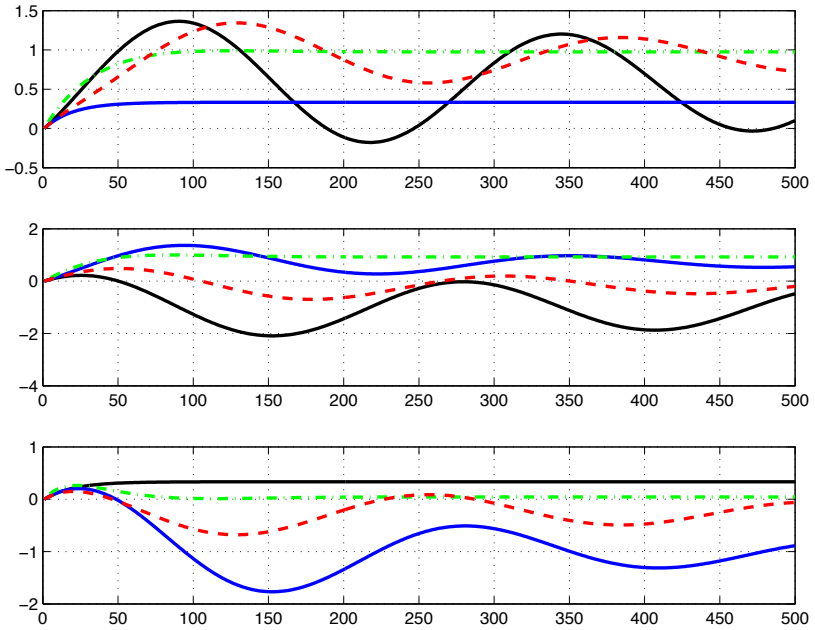


Figure 4.11: The step responses of Systems 1 (solid, black) and 2 (solid, blue) with different damping ratios, the GA interpolated system (dashed, red) and the reference system interpolated matrix interpolation (dash-dotted, green) for each state.

4.1.3 Variation: Different Natural Frequencies

The system matrices are of the same structure as above. The natural frequency of system two is increased as given by the coefficients: $a1 = 0.1,\ a2 = a1 * \sqrt{\frac{b2}{b1}}$ and $b1 = 2, b2 = 4$. The damping ratio is kept constant via the choice of the coefficient $a2$. Hence, the defining matrices are

$$\boldsymbol{A}_{1,\text{ natfreq}} = \begin{bmatrix} 0 & 1 & 0 \\ -2 & -0.1 & 0 \\ 0 & 0 & -3 \end{bmatrix},\ \boldsymbol{A}_{2,\text{ natfreq}} = \begin{bmatrix} -3 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -4 & -0.1414 \end{bmatrix},$$

$$\boldsymbol{B}_{1,\text{ natfreq}} = \boldsymbol{B}_{2,\text{ natfreq}} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix},$$

$$\boldsymbol{c}^T_{1,\text{ natfreq}} = \boldsymbol{c}^T_{2,\text{ natfreq}} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$$

$$\Phi_1 = \Phi_2 = 0.5,$$

The interpolation algorithm is run with the altered matrices and yields the following results. The new $\boldsymbol{A}$ matrix is assembled according to (3.29)

$$\boldsymbol{A}_{int,\text{ natfreq}} = \begin{bmatrix} -1.0731 & 1.4123 & -1.9269 \\ -1.3263 & -0.9746 & 1.3263 \\ -1.9269 & -1.4123 & -1.0731 \end{bmatrix} \tag{4.6}$$

Note the differences compared to the $\boldsymbol{A}$ matrix given by (4.1) and (4.6).
For comparison, the $\boldsymbol{A}$ matrix computed by conventional matrix interpolation is found as

$$\boldsymbol{A}_{int,\text{ matrix}} = \begin{bmatrix} -1.5000 & 0.5000 & 0 \\ -1.0000 & -0.0500 & 0.5000 \\ 0 & -1.0000 & -1.7000 \end{bmatrix} \tag{4.7}$$

Fig. 4.12 depicts the state vectors trajectories starting at the initial condition $x_0 = [1, 1, 1]$ as in the above example. Again, the orientation, the damping behavior and the natural frequencies are correctly interpolated with the GA interpolation method. In contrast, the matrix interpolation results in a non-oscillating system whose trajectory reaches the origin almost directly.
Fig. 4.13 shows the step responses of the two given systems and the interpolated systems as well as the step response of the system computed with common matrix interpolation. Note that the natural frequency is correctly interpolated. At the same time the damping behavior

is correctly interpolated as well: since the damping ratios of the two original systems is identical, the damping ratio of the interpolated system is identical to the original damping ratio, too.

Fig. 4.14 shows the step responses of each state separately. Note here, only the second state oscillates in both given systems. The GA interpolation provides an accurate interpolation, where both the frequency and damping ratio are reasonable. Even the states where one system oscillates while the other system does not are intuitively correctly interpolated.

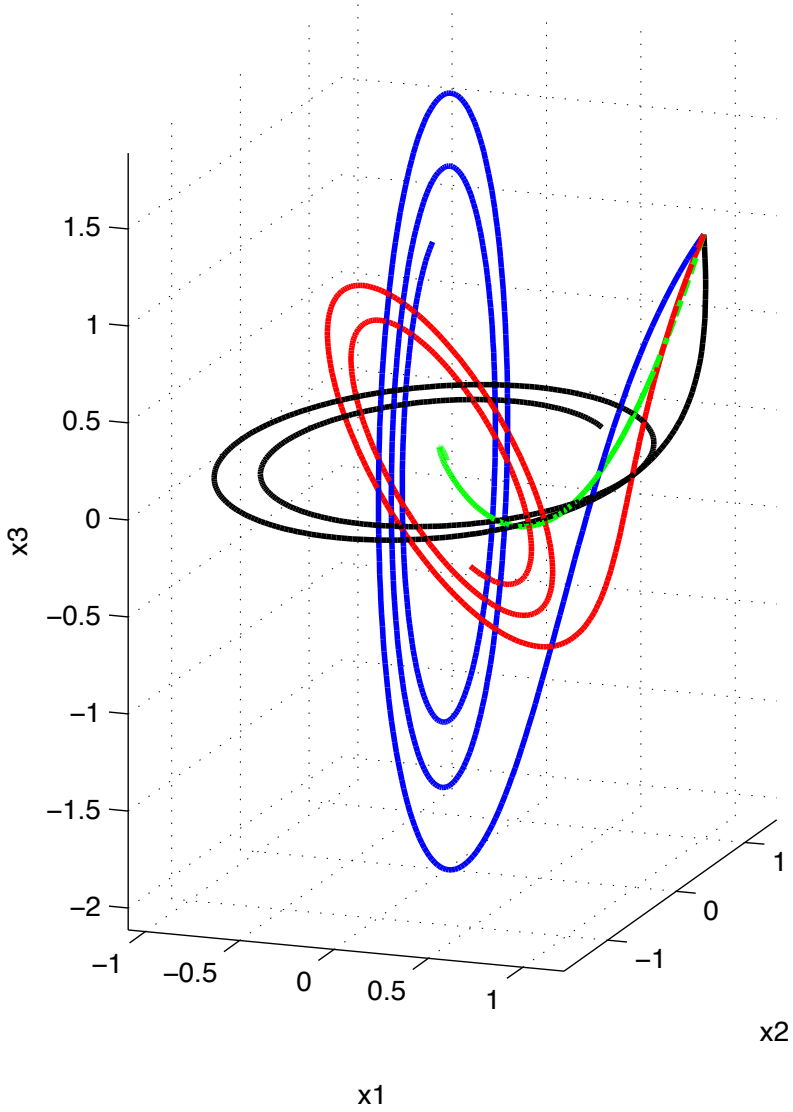


Figure 4.12: The trajectories of the state vectors of Systems 1 (solid, black) and 2 (solid, blue) with different natural frequencies, the GA interpolated system (solid, red, tilted) and the reference system interpolated by conventional matrix interpolation (solid, green) with common initial conditions $x_0 = [1, 1, 1]$.

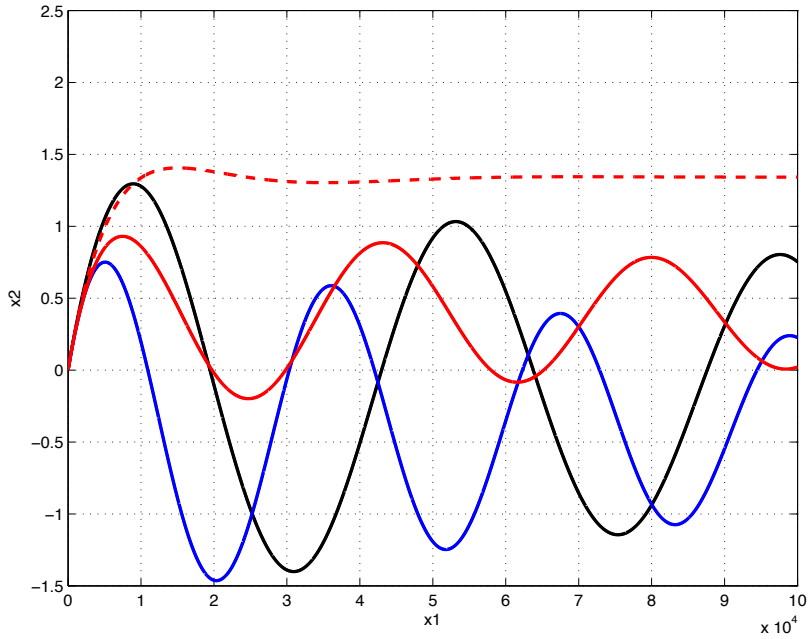


Figure 4.13: The step responses of Systems 1 (solid, black) and 2 (solid, blue) with different natural frequencies, the GA interpolated system (solid, red) and the reference system interpolated by matrix interpolation (dashed, red).

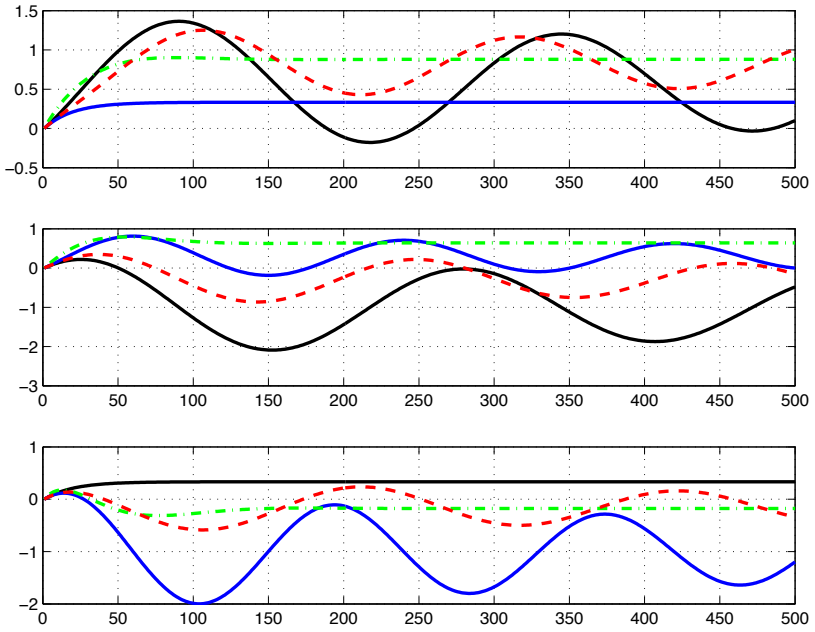


Figure 4.14: The step responses of Systems 1 (solid, black) and 2 (solid, blue) with different natural frequencies, the GA interpolated system (dashed, red) and the reference system interpolated matrix interpolation (dash-dotted, green) for each state.

4.1.4 Variation: Angular shift of the second oscillation plane

This variation of the above example serves to show that the GA interpolation method delivers reasonable results for increasing angular difference between the interpolated oscillation plane. At the same time the quality of the result of conventional matrix interpolation declines rapidly.

The above defined oscillation plane of system one is the basis for this extension. The alternative intermediate planes are generated by multiplication of the original bivector representing the first oscillation plane with a spinor. The axis of rotation and the angular "steps" need to be specified and, together, define the spinor. Here, the second oscillation plane will be rotated over a range of $\Psi = \frac{2\pi}{3}$ rad $= 120°$ in $n = 6$ steps of $\Psi_{\text{step}} = \frac{\pi}{9}$ rad $= 20°$ so that it will step-by-step fill the gap between the first original and second original oscillation plane. Note here, that the axis of rotation is chosen to be

$$\boldsymbol{u}_{\text{rotation}} = \boldsymbol{e}_1 + \boldsymbol{e}_2 + \boldsymbol{e}_3 \tag{4.8}$$

hence, the first plane including its defining eigenvectors is rotated onto the second plane to match it exactly in the sixth rotation step. The interpolated oscillation planes angular shift will range from 45 when interpolating the two original oscillation planes to $\approx 8°$ when interpolating the first original oscillation plane with the first alternative oscillation plane.

Figure 4.15 shows the original constellation and the additional oscillation planes (yellow).

From these rotated alternative oscillation planes the "corresponding" system matrices are computed (in order to perform conventional matrix interpolation for comparison of results only) with the original eigenvalues and equation (3.29).

Finally, the GA interpolation process is run with every one of these alternative second planes separately, yielding six alternative interpolated systems.

For evaluation the initial responses of the computed systems is chosen. Initial responses take only the system matrix $\boldsymbol{A}$ and the output matrix $\boldsymbol{C}$ of a system into account. When computing initial responses with identical initial conditions for all systems results become readily comparable.

Since the system characteristics of system one and system two are identical all alternative second systems exhibit identical system dynamics, too. Figure 4.16 shows the initial responses of the original systems one and the identical alternative system one and the initial response GA interpolated system as well as the conventional matrix interpolated system. Since the systems that are being interpolated are identical, all interpolated systems are identical to the original systems, too. Consequently, the initial response of all systems is identical and only one plotted line is visible.

Figure 4.17 shows the initial responses of the original systems and the alternative systems

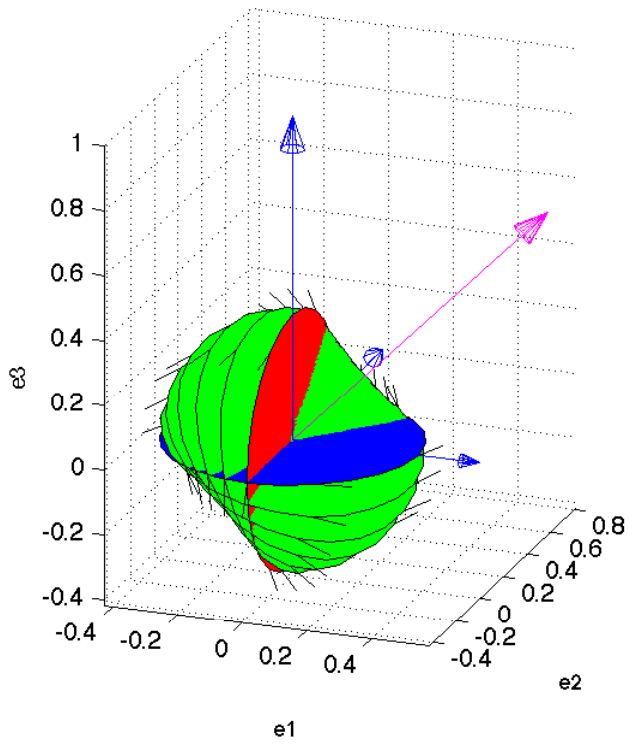


Figure 4.15: The original oscillation planes of system one (blue), system two (red), and the additional six rotated versions of the second oscillation plane (green). The magenta vector represents the axis of rotation.

with tilted oscillation planes but otherwise identical system characteristics (identical, solid, black). The initial response of the systems computed with GA interpolation are shown in colors (solid) and the initial response of the systems computed with matrix interpolation are shown in matching colors (dashed).

The slight difference in the GA interpolated results is due to numerical inaccuracies. The ideal solution would be identical to the original system response. That means, the ideal initial response plot would show only one solid line representing all original and GA interpolated systems. Note how conventional matrix interpolation fails to interpolate the system characteristics of any tested set of systems (with different oscillation planes).

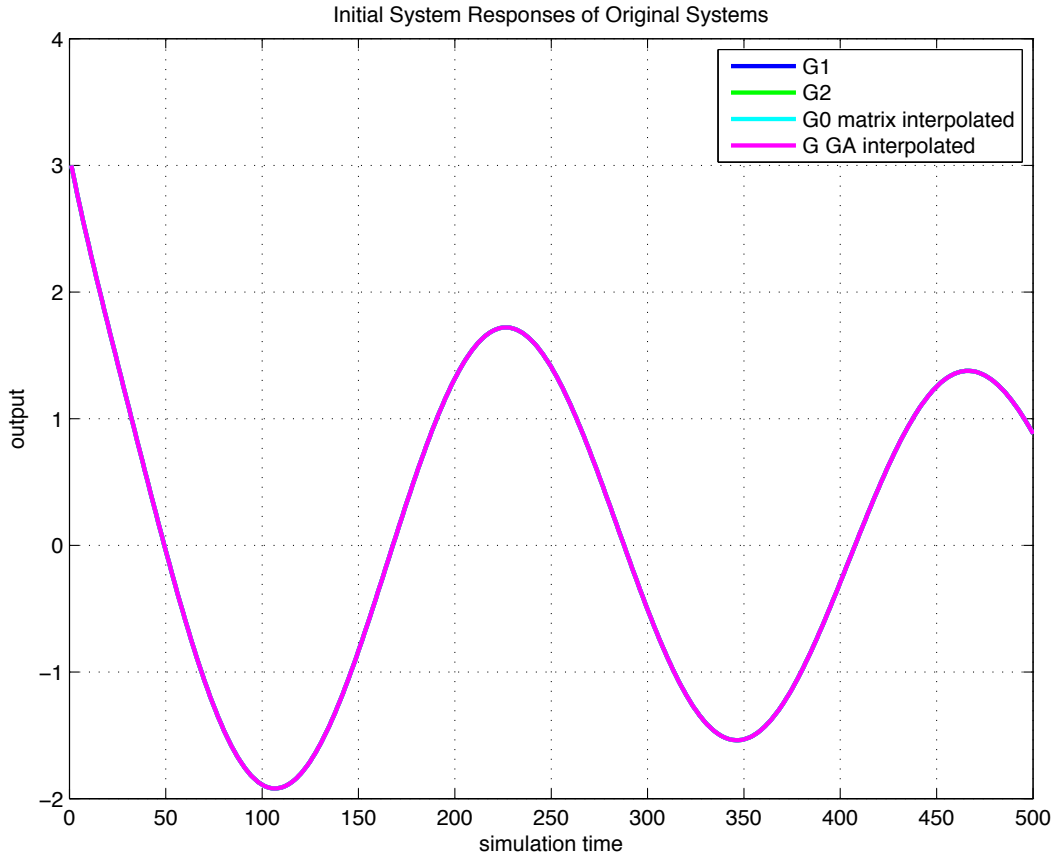


Figure 4.16: The initial responses of system one (blue), identical system two (green) and the initial responses of the systems interpolated by GA interpolation (magenta) and conventional matrix interpolation (cyan).

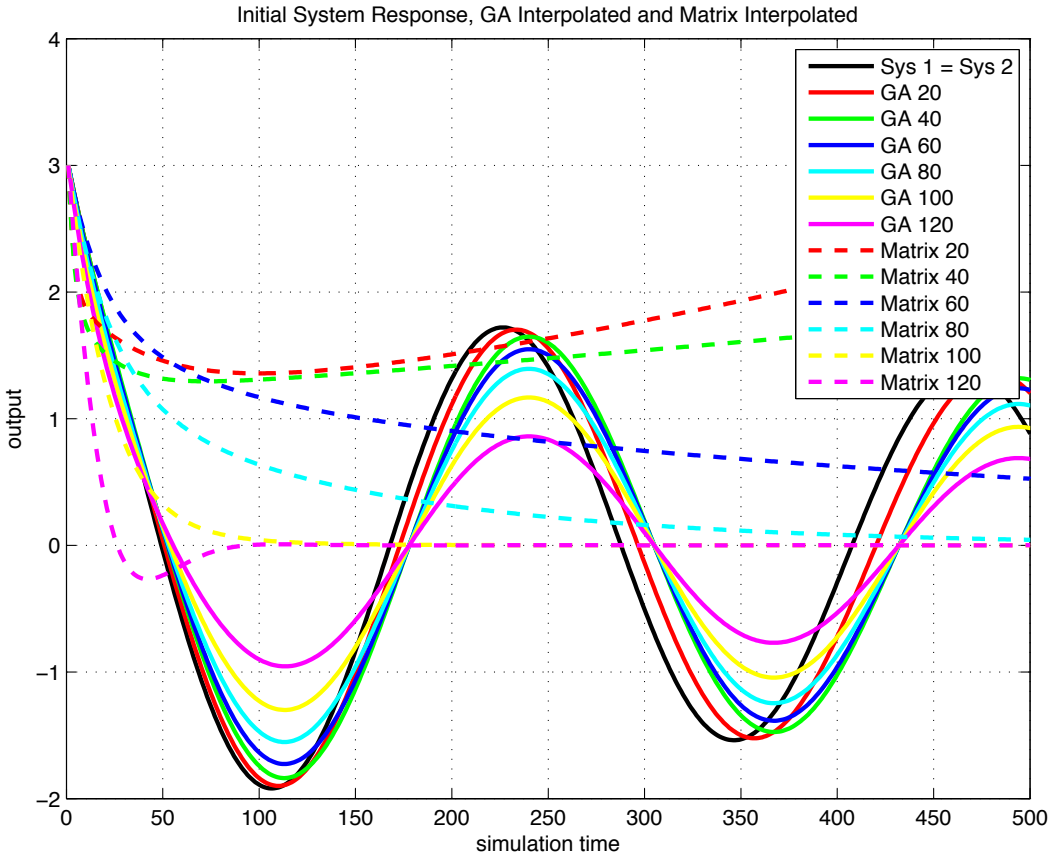


Figure 4.17: The initial responses of the original and tilted systems (identical, black), the initial responses of the systems interpolated by GA interpolation (colors, solid) and conventional matrix interpolation (matching colors, dashed).

4.2 General Example

The second demonstrative example is an interpolation of two arbitrary, discrete-time systems, defined by (1.2) and

$$\boldsymbol{A}_1 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0.4618 & -1.7535 & 2.2682 \end{bmatrix}, \boldsymbol{A}_2 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0.6137 & -1.9999 & 2.3474 \end{bmatrix},$$

$$\boldsymbol{B}_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \boldsymbol{B}_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix},$$

$$\boldsymbol{c}_1^T = \begin{bmatrix} 0 & 0 & 0.0660 \end{bmatrix}, \boldsymbol{c}_2^T = \begin{bmatrix} 0 & 0 & 0.0634 \end{bmatrix}$$

$$\Phi_1 = \Phi_2 = 0.5, \quad Ts = 0.01$$

These systems are fairly similar, the angular difference between the oscillation planes is small compared to the previous example. The example serves to show that the introduced GA interpolation performs well on problems, where matrix interpolation is commonly applied and performs well, too.

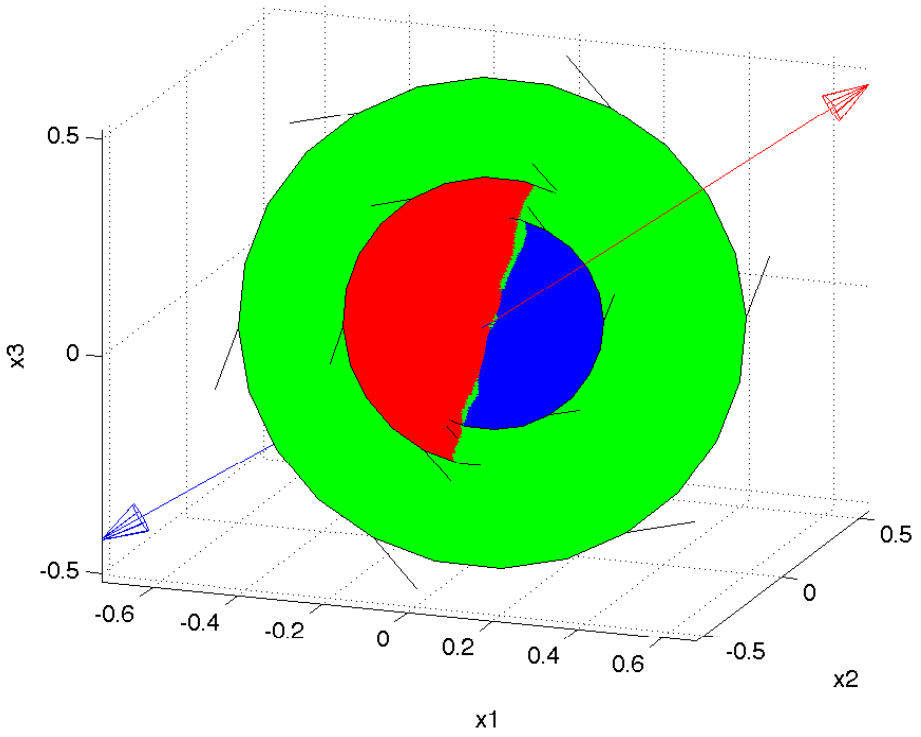


Figure 4.18: The oscillation planes and the interpolated plane.

Fig. 4.18 shows the both given oscillation planes as well as the interpolated plane.
The interpolated eigenvalues and the matrix of eigenvectors compute to

$$\boldsymbol{\Lambda}_{int} = \begin{bmatrix} 0.7561 + 0.3010i & 0 & 0 \\ 0 & 0.7561 - 0.3010i & 0 \\ 0 & 0 & 0.7956 \end{bmatrix},$$

$$\boldsymbol{U}_{int} = \begin{bmatrix} 0.6878 & 0.6878 & -0.6135 \\ 0.5193 + 0.2042i & 0.5193 - 0.2042i & 0.2018 \\ 0.3229 + 0.3087i & 0.3229 - 0.3087i & 0.7635 \end{bmatrix}$$

Thus, the $\boldsymbol{A}_{int}$ matrix reassembled to

$$\boldsymbol{A}_{int} = \begin{bmatrix} -10.2146 & 23.9737 & -15.1845 \\ -8.7891 & 19.7941 & -12.0846 \\ -5.8898 & 12.6114 & -7.2717 \end{bmatrix}$$

Note here, that the matrix is generally a non-sparse matrix. Since both given $\boldsymbol{A}$ matrices are of controllability canonical form a transformation step is performed:

$$\boldsymbol{A}_{int,\,\text{trans}} = \boldsymbol{T}\boldsymbol{A}_{int}\boldsymbol{T}^{-1} \tag{4.9}$$

where $\boldsymbol{T}$ is a suitable transformation matrix. The general structure of the $\boldsymbol{A}$ matrix is thereby retained and computes to

$$\boldsymbol{A}_{int,\,\text{trans}} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0.5269 & -1.8654 & 2.3078 \end{bmatrix}$$

In this particular example, where the oscillation planes and eigenvalues of the system matrices are almost identical the conventional matrix interpolation method performs well too. The interpolated system matrix via matrix interpolation computes to

$$\boldsymbol{A}_{int,\,\text{MatrixInt}} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0.5377 & -1.8767 & 2.3078 \end{bmatrix}$$

The resulting system matrices $\boldsymbol{A}_{int,\,\text{trans}}$ and $\boldsymbol{A}_{int,\,\text{MatixInt}}$ are almost identical.
Note that both $\boldsymbol{B}$ and $\boldsymbol{C}$ matrices are interpolated linearly as defined by (3.30) and (3.31).
Fig. 4.19 depicts the state vectors trajectories starting at the initial conditions $x_0 = [1, 1, 0]$.

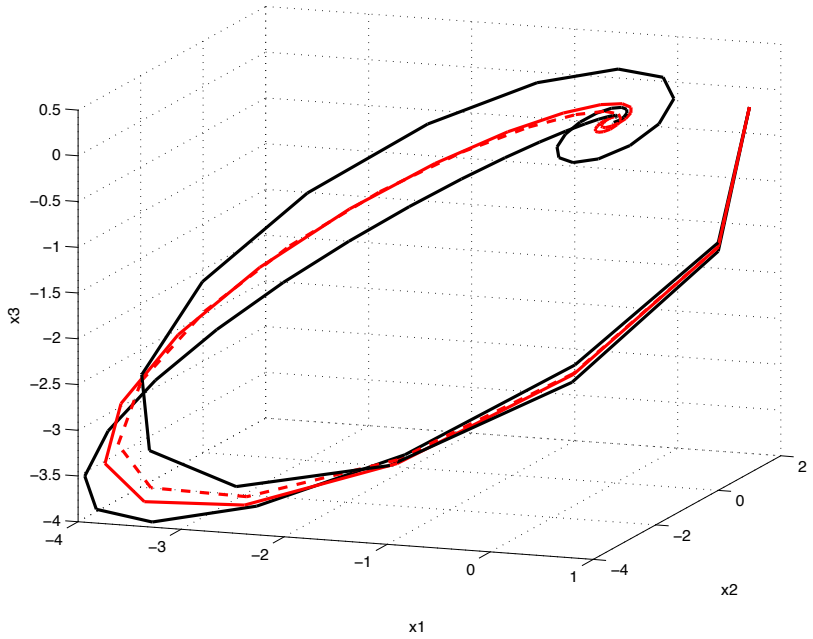


Figure 4.19: The trajectories of the state vectors of System 1 and 2 (solid, black), the GA interpolated system (solid, red) and the reference system interpolated by matrix interpolation (dashed, red) with common initial conditions $x_0 = [1, 1, 0]$.

It can easily be verified that the result is plausible, the interpolation generates a system with adequately interpolated dynamics.

Fig. 4.20 shows step responses of both given systems and the interpolated system as well as the step responses of the matrix interpolation method for comparison. In this case the matrix interpolation works well, too.

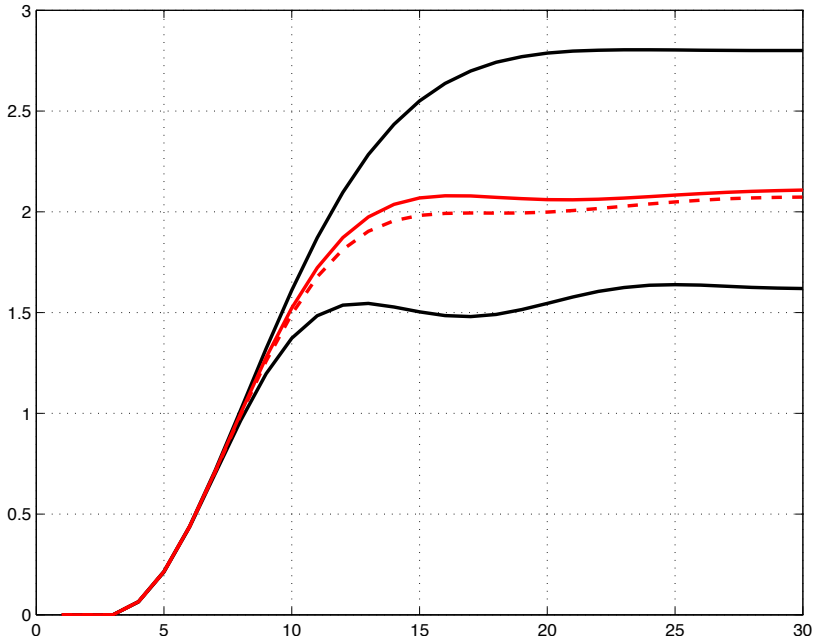


Figure 4.20: The step responses of Systems 1 and 2 (solid, black), the GA interpolated system (solid, red) and the reference system interpolated by matrix interpolation (dashed, red).

4.2.1 Influence of Input and Output Matrices

Active steady state value condition

Fig. 4.21 depicts the step response of a system with active steady state value condition. That means, the Input matrix is computed as defined by (3.33). The output matrix is linearly interpolated.
The new $\boldsymbol{B}$ matrix computes to

$$\boldsymbol{B}_{int,\text{ enforced}} = \begin{bmatrix} 0 \\ 0 \\ 1.0478 \end{bmatrix} \tag{4.10}$$

It is almost identical to the input matrix computed with linear interpolation, hence the difference of the step response is small and can hardly be identified on the plot. No significant difference between the system generated by the developed algorithm and the system with the extra condition on the steady-state values is noticeable. This is due to the similarity of the interpolated systems.

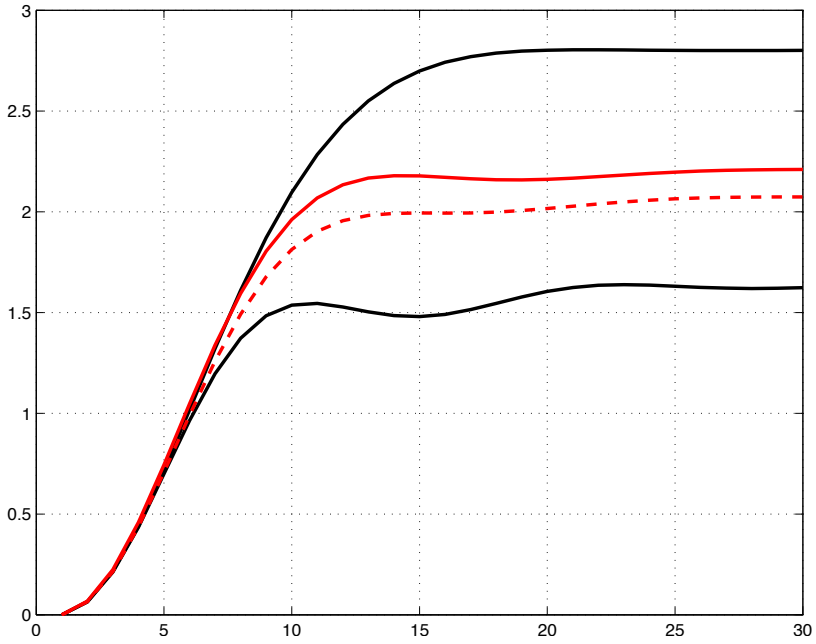


Figure 4.21: The step responses of Systems 1 and 2 (solid, black), the GA interpolated system (solid, red) and the reference system interpolated by matrix interpolation (dashed, red). Active steady state value condition.

Active steady state value and output condition

Fig. 4.22 depicts the step response of the system with active steady state value condition and active output condition. That means, the Input matrix is computed as defined by (3.33) and the output matrix is computed as defined by (3.43) The new $\boldsymbol{C}$ matrix computes to

$$\boldsymbol{C}^T_{int,\text{ enforced}} = \begin{bmatrix} 0 & 0 & 0.0650 \end{bmatrix}$$

Note that the original output matrices are

$$\boldsymbol{c}_1^T = \begin{bmatrix} 0 & 0 & 0.0660 \end{bmatrix}, \boldsymbol{c}_2^T = \begin{bmatrix} 0 & 0 & 0.0634 \end{bmatrix}$$

The step response is generated with $\boldsymbol{B}_{int,\text{ enforced}}$ and $\boldsymbol{C}_{int,\text{ enforced}}$.

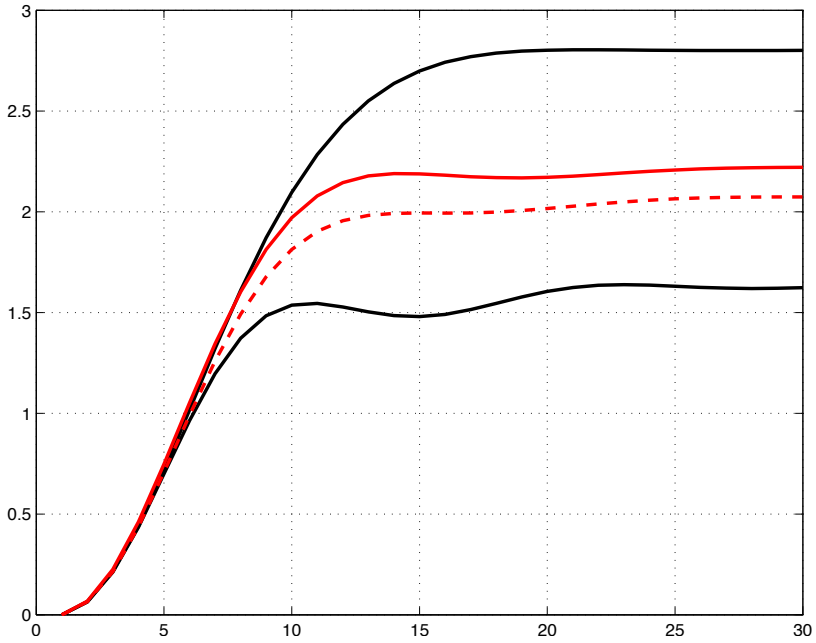


Figure 4.22: The step responses of Systems 1 and 2 (solid, black), the GA interpolated system (solid, red) and the reference system interpolated by matrix interpolation (dashed, red). Active steady state value condition and active output condition.

Chapter 5

Summary and Outlook

5.1 Accomplishments

An interpolation algorithm for system dynamics has been developed. The algorithm is based on the state space representation of local model systems. There is no dimension limit imposed. The method interpolates dynamic system characteristics simultaneously:

- **Natural Frequency**
- **Damping Ratio**
- **Attitude of the oscillation planes** in the state space
- **Orientation of the oscillation planes** in the state space
- **Steady State Values** of the state vector of SISO systems

The steady state value of the state vector is generally not linearly interpolated simultaneously. For arbitrary systems its simultaneous linear interpolation can be enforced with an additional condition on the input matrix. Additionally the output matrix must be adjusted, to finally make the correctly interpolated steady state value visible in the output. A condition for the output matrix can be derived but poses numerical difficulties. For SISO systems the condition can be significantly simplified and solved. Other systems generally can not satisfy the condition. In some cases working with numerical tricks can circumnavigate the matrix-inversion problem.

5.2 Fields of Further Research

5.2.1 Mode Tracking

In order to apply the interpolation algorithm a correlation between the modes of the original systems need to be established. This task is addressed by mode tracking. Mode tracking is not part of this work, hence only an ad hoc approach is presented. This approach demands that a user manually establishes the correct correlation between modes. Due to its simplicity it might as well be implemented into a software routine. Nevertheless, further research is required on mode tracking and the physical interpretation of mode correlation.

5.2.2 Interpolation between a Conjugate Complex Pair and two Real Valued Poles

In section 3.2 the six possible interpolation cases that appear during the interpolation process are introduced. All except the last case can be handled by the interpolation algorithm. The last case, where a conjugate complex couple of eigenvalues is interpolated with two different real valued poles is difficult. The fact that the transition from eigenvalues in the complex plane to eigenvalues on the real axis is not defined causes problems when computing the corresponding eigenvectors. At one point the poles must coincide and form a single real valued pole with multiplicity = 2. At that point neither the value of the pole nor the eigenvectors corresponding to that double pole are defined. Hence, this case is excluded in the algorithm. Further research is required on this special interpolation case.

5.2.3 Internal Dynamics of non-flat Systems

In this work only differentially flat systems have been investigated. These systems do not exhibit internal dynamics. Non-flat systems may exhibit internal dynamics as part of their system dynamics. It might be worth considering the idea of treating the internal dynamics analog to the system dynamics of flat systems. The idea seems plausible, since internal dynamics are a part of the system's dynamics and the interpolation process ideally captures the *entire* system dynamics.
Since flatness depends on both the input and output matrix of a system as well as the system matrix itself an interpolation scheme for the input and output matrices of non-flat systems must be developed, too.

Bibliography

[1] Leo Dorst, Stephen Mann, and Tim Bouma. *GABLE: A Matlab Tutorial for Geometric Algebra*, 1.3 edition, December 2002.

[2] Wilson J. Rugh and Jeff S. Shamma. Research on gain scheduling. *Automatica*, 2000.

[3] Wilson J. Rugh. *Analytical Framework for Gain Scheduling.* IEEE Control Systems, Jan 1991.

[4] T. Takagi and M. Sugeno. Fuzzy Identification of Systems and its Applications to Modeling and Control. *IEEE transactions on systems*, 1985.

[5] Eduardo Bayro-Corrochano, Sven Buchholz, and Gerald Sommer. A new self-organizing neural network using geometric algebra. *IEEE Proceedings of ICPR*, 1996.

[6] Oliver Nelles. *Nonlinear System Identification: From Classical Approaches to Neural Networks and Fuzzy Models.* Springer Verlag GmBH, 2001.

[7] Robert A. Nichols, Robert T. Reichert, and Wilson J. Rugh. *Gain Scheduling for $H\infty$ Controllers: A Flight Control Example.* IEEE Transactions on Control Systems Technology, 1, no. 2 edition, June 1993.

[8] Gene F. Franklin, J David Powell, and Michael L. Workman. Digital control of dynamic systems. page 760, Jan 2002.

[9] Christian H. Mayr, Christoph Hametner, Martin Kozek, and Stefan Jakubek. Relaxed Fuzzy Lyapunov Approach for Dynamic Local Model Networks. *Proceedings of the IEEE International Conference on Fuzzy Systems 2011*, 2011.

[10] Gilles Ferreres. Computation of a Flexible Aircraft LPV/LFT Model Using Interpolation. *IEEE Transactions on Control Systems Technology*, 19(1):132 – 140, January 2011.

[11] Richard M. Murray, Muruhan Rathinam, and Willem Sluis. Differential flatness of mechanical control systems: A catalog of prototype systems. *ASME International Mech. Eng. Congress and Expo*, November 1995.

[12] Robert Murphey, Sunil K. Agrawal, Stephen T. Pledgie, Yongxing Hao, and Armando M. Ferreira. Groups of unmanned vehicles: Differential flatness, trajectory planning, and control. *IEEE Proceedings of the International Conference of Robotics & Control*, May 2002.

[13] T. John Koo and Shankar Sastry. Differential flatness based full authority helicopter control design. *Proceedings of 38th Conference on Decision and Control*, December 1999.

[14] Leo Dorst and Stephen Mann. *Geometric Algebra: A Computational Framework for Geometrical Applications: Part 1 and 2*. IEEE Computer Graphics and Applications, May/June 2002.

[15] D. Hestenes and G. Sobezyk. Clifford algebra to geometric calculus: a unified language for mathematics and physics. *Kluwer Academic*, 1986.

[16] M. Castilla, J.C. Bravo, M. Ordonez, J.C. Montano, A. Lopez, D. Borras, and J. Gutierrez. The geometric algebra as a power theory analysis tool. *International School on Nonsinusoidal Currents and Compensation*, 2008.

[17] Jorge Rivera-Rovelo and Eduardo Bayro-Corrochano. Medical image segmentation using a self-organizing neural network and clifford geometric algebra. *2006 International Joint Conference on Neural Networks*, July 2006.

[18] G. H. Golub and C. F. VanLoan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, 3 edition, 1996.

Elvira Thonhofer

Curriculum Vitae

☎ *+43 699 125 78 766*
✉ *e0425278@student.tuwien.ac.at*

Personal Information

Date of Birth	October 14, 1984
Place of Birth	Vienna, Austria
Nationality	Austrian
TU Wien Univ. ID	e0425278

Education

2004 – now **M.Sc. in Mechanical Engineering.**, *Inst. for Mechanics and Mechatronics, Division of Control and Process Automation*, Technical University Vienna, Austria.
Expected in Spring 2011

– Master Thesis

title *Interpolation of System Dynamics*

supervisors Univ.Prof. Dipl.-Ing. Dr.techn. Stefan Jakubek, Projektass. Dipl.-Ing. Christian Mayr

description Algorithms for solving 3-dimensional system interpolation. A matlab toolbox is developed for computation and visualisation of results. Matlab code based GABLE, a matlab toolbox for Geometric Algebra.

2008 – 2009 **CHALMERS TU via ERASMUS**, *M.Sc. Program in Naval Architecture*, CHALMERS TU, Göteborg, Sweden.
Academic exchange year

Aug 2007 **Summer School at TU Uppsala, Sweden**, *Aerospace Engineering*, TU Uppsala, Sweden.
Scientific satellite design, orbit calculation, relevant control systems.

1999 – 2004 **Secondary College for Industrial Chemistriy** , *Department of Biochemistriy and Gene Technology*, Vienna, Austria.
- Graduated with distinction
- authored a scientific thesis on a subject related to the Department as part of graduation

Academic Backgrouond

Control Engineering	Digital control, MIMO Systems, Adaptive and Predictive Control, Neural Networks, System Identificaion, ...
Mathematics	Calculus, Linear Algebra, Geometric Algebra, ODE, PDE, Analysis, FEM, ...
Naval Design, Mechanical Engineering	Ship Structures, Class Rules, Composite Materials and Lightweight Design, Buckling Analysis, Strength and Stiffness Requirements, Hydrostatics and -dynamics, simulation, ...

Experience

Working

Sept 2009 – now **Pleasure Yacht and Boat Design Projects**, *Acico Yachts resp. Naval Design by Christian Bolinger and Bootsbaumeisterin*, Enkhuizen, Netherlands and Vienna, Austria.
- Structural Design for a pleasure Yacht, solar catamaran with ocean classification
- Design of a Venetian Gondola, supervising the production process

July – Sept 2009 **Junior Designer for Ship Structure**, *Naval Design by Christian Bolinger*, Udligenswil, Lucerne, Switzerland.
- Structural Design for pleasure Yachts up to 20m
- Communication with Class Rule Societies, Deck-plan Layout, Stability Calculations and Supervision of manufacturing process

2007 – Jan 2009 **Junior Structural Designer for Transmission Towers**, *Ziviltechnikerbüro Schelmberger*, Vienna, Austria.
- Full time during summers, part time during semester schedule
- Structural Design for transmission towers and aerial masts
- Structural refitting for hosting more/different antennas
- Re-computation of old existing power-line structures with up-to-date methods

Research

2003 – 2004 **Graduation Project**, *University of Vienna, Max F. Perutz Laboratories*, Vienna, Austria.
- During school schedule, replacing part of regular Laboratory Classes and extending to extracurricular research.
- Independent scientific research, completed with a Graduation Thesis

– Graduation Thesis

title *Studies on Development and Characterisation of metal nano-cluster biochips based on anomalous absorption as a new assaying principle*

supervisor Univ.Prof. Dr.techn. Fritz Pittner, Mag. Peter Altrichter, Jakob Haglmüller

description Student teams of 2 perform in an individual scientific research project. As part of the graduation exam the project is presented, followed by an oral exam similar to academic graduation processes.

Teaching

Jan – Feb 2007 **Tutor**, *Department of computer-aided Engineering and Design*, TU Vienna, Austria.
- Supervising students during their engineering project, designing combustion engines and gear boxes

Languages

German **Native**

English **Fluent** *Level C2 - C1*

Swedish **Basic** *Classes taken during ERASMUS exchange*

Computer skills

OS Windows, Mac OSX, Ubuntu

scientific Matlab, FLUENT, ShipFlow

typography, office LaTeX, MS Office, OpenOffice

Design Rhino3D, VRay, ProE, MaxSurf, AutoShip

Updated

July 15, 2011