An Energy Stable Discontinuous Galerkin Discretization Approach for the Geometrically Exact Intrinsic Beam Model

Christian Bleffert

Master's Thesis Mathematics

Universität zu Köln

Department Mathematik/Informatik

November 10, 2022

Advisors: Prof. Dr. Gregor Gassner,
Lukas Dreyer

Contents

1 Introduction

In this thesis we are concerned with the *Geometrically Exact, Intrinsic Model for Dynamics of Curved and Twisted Anisotropic Beams* which was derived by Hodges [20]. Particular emphasis is placed on its discretization in space using an *Energy Stable Discontinuous Galerkin Approach.*

Attempts at modelling flexible beams have been made for over 500 years. As early as the 16th century, already Leonardo da Vinci was interested in this field [4]. Since then, several theories have been developed, extended and improved. Very well known representatives are the *Euler-Bernoulli model* and the *Timoschenko model.* The Euler-Bernoulli model was developed in the mid 18th century while the works founding the Timoschenko beam theory were published over 150 years later in 1916 [13]. Both models assume that the underlying beam follows linear elastic behaviour. As a consequence, they are only valid for small deformations [6, 32].

The Geometrically Exact, Intrinsic Model for Dynamics of Curved and Twisted Anisotropic Beams as it is considered here was developed in 2003 by Hodges [20]. It will in the following be referred to as *Intrinsic Beam Model*, or *Intrinsic Beam Equations*, which we use synonymously. Unlike the Euler-Bernoulli and the Timoschenko model, it is *geometrically exact*, which means that it contains non-linearities. Moreover, it considers intrinsic variables only, meaning that neither displacement nor rotation variables appear in the corresponding governing equations.

Geometrically exact formulations allow us to model beams undergoing large deformations, making them particularly interesting for applications regarding highly flexible structures. Modern applications include for example highly flexible wings of aircraft [1, 36], robotic arms [48] or wind turbine blades [33, 53].

This thesis was written in cooperation with the German Aerospace Center (DLR). Their software framework *VAST* (Versatile Aeromechanic Simulation Tool) [52] is used to simulate the dynamics of flying helicopters. In order to increase the accuracy of the simulation, the helicopter rotor blades and their dynamical behaviour have to be simulated as precisely as possible. The intrinsic beam model is a promising candidate for this purpose. With this application in mind, we focus ourselves on general beams that are clamped at one end and free swinging at the other.

The governing equations that represent the intrinsic beam model are a system of partial differential equations (PDEs). Modelled beams are idealized by a one dimensional reference line, leading to a problem in one space dimension and a time dimension. For a well posed problem, we need to provide boundary conditions at the boundaries of the spatial domain and an initial condition at the start time. Our first goal will be to formulate a problem with suitable boundary, and initial conditions that describes the setup of the clamped beam. Considerations regarding boundary conditions that describe clamped beams have been made for example in [37, 49] from a more physical or engineering point of view. We want to verify the admissibility of such boundary conditions in a mathematical point of view. Similar investigations of mathematical appropriate boundary conditions for the intrinsic beam model, concerning networks of beams, have been made in [42, 43].

As it is often the case for complex partial differential equations modelling physical phenomenons, an analytical solution of the intrinsic beam equation is not known in general. Nonetheless, in order to simulate flexible beams modelled by the intrinsic beam model, a solution, or at least an approximation to a solution, has to be found. Several methods to solve PDEs numerically are available. Each of them does have different properties. Therefore, different numerical approaches can be more or less suitable to solve certain types of equations. Before choosing a numerical approach that suits the problem well, the considered PDE has to be investigated regarding its properties. We will show that the governing system of equations of the intrinsic beam model can be classified as a *system of linear hyperbolic balance laws.* This has also been shown in [42, 43].

Numerical approaches, that are widely used to determine approximate solutions of hyperbolic balance laws are the so called *Discontinuous Galerkin (DG) Approaches.* They are used to discretize the considered problem. The discretized problem can then be solved numerically to obtain an approximate solution of the original problem. DG discretization approaches for hyperbolic balance laws are for example used in [18, 26, 27, 54].

The basic idea behind the theory of DG discretization is to approximate the solution by piece wise high order polynomials. That is, while the resulting numerical solution is continuous locally, it is not from a global perspective. As a result, there occur jumps in the global numerical solution that have to be dealt with in the discretization process. To do so, the concept of so called *numerical fluxes* is used which will also be described in this thesis.

When it comes to the derivation of DG discretization schemes for certain problems, one aspect is of special interest: the *stability* (cf. for example [18, 26]). Therefore, to prepare the discretization, the potential solution of the original problem is analyzed regarding its *energy*. Often, the energy is either the squared Lebesgue norm or the square of a so called *energy norm* of the solution that is induced by an inner product on the base of a positive definite matrix.

In the case of the intrinsic beam equation, the latter definition of energy will be used, and we will see that this is not only an abstract mathematical energy but also a measure of actual mechanical energy of the modelled beam. A necessary condition for the mathematical well-posedness of a problem is, that if there exists a solution of the problem, its energy growth can be bounded by a "nicely" behaving function. That is, the energy cannot increase at an arbitrary rate in time. A solution of the problem whose growth of energy can be bounded, is called *energy stable*. This property can be transferred to the numerical solution resulting from the DG discretization approach. In particular, a bound for the rate at which the discrete energy of the resulting numerical solution increases has to be found, analogously to the investigations into the energy of the original problem. If such a bound for the numerical solution can be found a priori, the DG discretization scheme is called *energy stable* as well.

The main goal of this thesis is to derive an energy stable Discontinuous Galerkin scheme to discretize the governing equations of the intrinsic beam model with respect to the spatial variable. Energy stability statements for the solution of the original problem have been made by other authors, assuming that the modelled beam is not exposed to any external forces and moments, for instance in [44]. Although we will make some restrictions on external forces and moments, too, we will show that energy stability can be obtained even for non-zero external forces and moments. To the best of our knowledge, such a statement has not been proven yet. Furthermore, it will be shown that this statement can be transferred to the discrete energy of the numerical solution.

In the literature, different numerical approaches have been used for the discretization of the intrinsic beam model. In [20] for instance, Hodges himself suggests a finite difference scheme. In [37], a discontinuous finite element discretization with penalty terms is presented to numerically solve the intrinsic beam equations. However, to the best of the author's knowledge, the here derived DG scheme in its generality is new.

Beyond the above described theoretical aspects, we want to implement the developed discretization scheme using the simulation framework `Trixi.jl` [46]. The implementation will be used to simulate different configurations of the intrinsic beam model and to analyze the resulting numerical solutions. In particular, the simulation results will be investigated with regard to their discrete energy to verify the theoretical stability statements we derive.

The structure of the following sections is as follows: In section 2, we will examine the analytical aspects of the intrinsic beam model. We will introduce the original formulation derived by Hodges and then reformulate it as a linear hyperbolic balance law. Afterwards, mathematically appropriate boundary conditions will be derived and used to describe the clamped beam. The resulting *initial boundary value problem* will then be analyzed regarding its energy stability.

The initial boundary value problem will further be discretized in space, using a DG approach in section 3. We will derive a *weak formulation* of the original problem, which will then be discretized, resulting in a *semi discrete formulation*. This includes deriving a *stable numerical flux* and consistently implement the derived boundary conditions into the discretization scheme. The semi discrete formulation will then be analyzed regarding its energy, analogously to the energy analysis in section 2. Eventually, an ordinary differential equation in time, resulting from the discretization in space, will be derived. Section 3 will be completed by some remarks on how the ordinary differential equation can be solved numerically.

In section 4, the theoretical considerations from the previous sections are implemented into the simulation framework `Trixi.jl`. We will describe the most important steps that are necessary for the implementations and afterwards solve different configurations of the intrinsic beam model. In particular, we will perform a convergence analysis using the method of *manufactured solutions*, analyze the discrete energy of different simulations and complete the section by demonstrating how the simulation results can be post processed to determine and visualize the beam's position and deformation.

2 Intrinsic Beam Model

In this section, we will consider the theoretical aspects regarding the intrinsic beam model. We will start by introducing the governing equations in their classical formulation as they are derived in [20]. Afterwards, we will give an overview about the theory of linear hyperbolic balance laws. On the base of this, we will show that the governing equations of the intrinsic beam model can, in fact, be interpreted as a linear hyperbolic balance law. This reformulation is followed by the derivation of appropriate boundary conditions, which eventually leads to the formulation of an initial boundary value problem. Potential solutions of the initial boundary value problem are afterwards investigated with regard to their energy to complete this section.

2.1 Classical Formulation of the Equations

In the course of this section, we want to present the classical formulation of the intrinsic beam equations as they are derived in [19, 20]. We refrain from a derivation of the equations and refer to the mentioned works of Hodges, where a derivation can be read up. The fundamental details that are necessary to describe and understand the equations, especially the remarks on the body attached variables and the different coordinate systems, are taken from [42], where the results of [20] are prepared in detail.

First of all, the considerations we do in this thesis are in general with regard to an *initially curved and twisted anisotropic beam* of length ℓ that is clamped at one end and free swinging at the other, and we want to model it in a specific time interval $[0, T]$. The intrinsic beam model is a geometrically exact model, which means that it contains non-linearities, so that the model is able to represent large motions including large displacements of the centerline and large deformations of the beam's cross sections. *Intrinsic* means that neither displacement nor rotation variables appear in the governing system. Instead, only intrinsic variables, namely internal forces and moments and linear and angular velocities are considered.

The beam in its undeformed state is idealized by a *reference line*, also called *centerline*, that goes through the centroids of all cross sections of the beam and is denoted by $r : [0, \ell] \to \mathbb{R}^3$. The space variable x is the running length coordinate along r. The orientation of the cross sections of the undeformed beam is described by $\mathfrak{S} : [0, \ell] \to \mathbb{R}^{3\times 3}$, in the sense that for $x \in [0, \ell]$, the matrix $\mathfrak{S}(x)$ is a rotation matrix with columns $\{\mathfrak{S}_j(x)\}_{j=1,2,3}$, that form an orthogonal basis of the beam's cross section at the point x in space, where $\mathfrak{S}_1(x)$ is chosen to be tangent to r.

Similarly, the a priori unknown position of the centerline of the beam in its deformed state is denoted by $\boldsymbol{r} : [0, \ell] \times [0, T] \to \mathbb{R}^3$. Because we want to model a dynamic beam, the position of the deformed beam's centerline necessarily depends on the time. The orientation of the cross sections of the deformed beam is given by $\boldsymbol{\mathfrak{S}} : [0, \ell] \times [0, T] \to \mathbb{R}^{3\times 3}$, in the sense that for $x \in [0, \ell]$ and $t \in [0, T]$, the columns $\{\boldsymbol{\mathfrak{S}}_j(x, t)\}_{j=1,2,3}$ of the rotation matrix $\boldsymbol{\mathfrak{S}}(x, t)$ form an orthogonal basis of the cross sections of the deformed beam at the point (x, t) in space time.

Throughout this thesis, we will consider two different types of coordinate systems in $\mathbb{R}^3$. One is the *global coordinate system*, also called *reference frame*. It refers to the standard basis $\{e_j\}_{j=1,2,3}$ that is fixed in space and time. The other is the so called *body attached coordinate system*, that refers to the basis $\{\boldsymbol{\mathfrak{S}}_j\}_{j=1,2,3}$, that moves together with the deformation of the beam and therefore depends on both, the space variable x and the time variable t. A vector $z = (z_1, z_2, z_3)^T \in \mathbb{R}^3$ in the global coordinate system has the representation

$$z = \sum_{j=1}^{3} z_j e_j.$$

Let $\boldsymbol{z} = (\boldsymbol{z}_1, \boldsymbol{z}_2, \boldsymbol{z}_3)^T \in \mathbb{R}^3$ be the representation of z in the body attached coordinate system. Vectors that are given with respect to the body attached basis, i.e. in the body attached coordinate system, are referred to as *body attached variables*. As z and $\boldsymbol{z}$ represent the same vector in different coordinate systems, z can be represented by

$$z = \sum_{j=1}^{3} \boldsymbol{z}_j \boldsymbol{\mathfrak{S}}_j = \boldsymbol{\mathfrak{S}}\boldsymbol{z}.$$

That is, a vector that is known with respect to the body attached basis, can be transformed to a vector in the reference frame, by a multiplication with $\boldsymbol{\mathfrak{S}}$.

Let now $\Theta(x,t)$ denote the internal forces of the beam, $\Xi(x,t)$ the internal moments, $V(x,t)$ the linear velocities, $\Omega(x,t)$ the angular velocities, $k(x)$ the initial curvature of the beam, $\kappa(x,t)$ the curvature of the beam relative to k (which can also be interpreted as the angular strains), $\gamma(x,t)$ the linear strains, $P(x,t)$ the linear momentum, $H(x,t)$ the angular momentum, $f_{ext}(x,t)$ external forces acting along the beam and $m_{ext}(x,t)$ external moments. All defined quantities in this paragraph have values in $\mathbb{R}^3$ and are body attached variables. Most of the time, at least when misunderstandings are ruled out, we omit the dependencies on x and t, writing for example Θ instead of $\Theta(x,t)$.

To stress that the considered quantities are body attached, is very important for the understanding and application of the upcoming equations. If, for example, we want to model a beam under the effect of gravitational force, we would have to consider this force as external and therefore insert it into the vector f_{ext}. While in the global coordinate system, gravitation results in a constant force in one direction, this is not necessarily the case in the body attached coordinate system. There, the acting forces depend on the position and the deformation of the considered beam. Hence, assuming a deforming beam, gravitation does not act as a constant force from the f_{ext} point of view. Effects of this fact will be considered also in section 2.6.

Before we specify the equations of the intrinsic beam model, we introduce a notation for the *cross product matrix*. Let $w = (w_1, w_2, w_3)^T, z = (z_1, z_2, z_3)^T$ be two vectors in $\mathbb{R}^3$. Then the cross product of the two vectors is defined as

$$w \times z = \begin{pmatrix} w_2 z_3 - w_3 z_2 \\ w_3 z_1 - w_1 z_3 \\ w_1 z_2 - w_2 z_1 \end{pmatrix}.$$

The cross product matrix, $\tilde{w}$, of the vector w is defined by

$$\tilde{w} := \begin{pmatrix} 0 & -w_3 & w_2 \\ w_3 & 0 & -w_1 \\ -w_2 & w_1 & 0 \end{pmatrix} \tag{1}$$

and with this notation, we can write the cross product of w and z equivalently as

$$w \times z = \tilde{w} z.$$

An important property of the cross product matrix, which we will use multiple times in the course of this thesis, is that it is skew symmetric. This can be derived directly from its above definition, as $\tilde{w}^T = -\tilde{w}$. Some more useful properties of the cross product and the cross product matrix will be derived in section 2.6.

The governing system of equations of the intrinsic beam model that is derived in [20] can now be formulated as

$$\begin{aligned} \Theta' + (\tilde{k} + \tilde{\kappa})\Theta + f_{ext} &= \dot{P} + \tilde{\Omega} P \\ \Xi' + (\tilde{k} + \tilde{\kappa})\Xi + (\tilde{e_1} + \tilde{\gamma})\Theta + m_{ext} &= \dot{H} + \tilde{\Omega} H + \tilde{V} P \\ V' + (\tilde{k} + \tilde{\kappa})V + (\tilde{e_1} + \tilde{\gamma})\Omega &= \dot{\gamma} \\ \Omega' + (\tilde{k} + \tilde{\kappa})\Omega &= \dot{\kappa} \end{aligned} \tag{2}$$

for $x \in (0, \ell)$ and $t \in (0, T]$, where with $(\cdot)'$, we denote the (partial) derivative with respect to the spatial variable x and with $\dot{(\cdot)}$ the (partial) derivative with respect to the time variable t. The first two equations are referred to as the dynamic equations while the last two describe the kinematics of the beam. Together, these four equations are a system of partial differential equations (PDEs).

In order to have a well posed problem, that can be solved (numerically), it is essential to have appropriate boundary conditions at $x = 0$ and $x = \ell$ and an initial condition at $t = 0$. We will look into the derivation of such conditions in section 2.5 and for now consider (2) isolated from boundary and initial conditions.

2.2 Constitutive Laws

With (2), we have a system of PDEs, describing the evolution of internal forces Θ, internal moments Ξ and linear and angular velocities V and Ω. However, there appear more unknowns in the system,

including the generalized momenta P and H and linear and angular strains γ and κ. In this section, we present two *constitutive laws*. One of them creates a linear connection between the variables Θ, Ξ and γ, κ and the other creates a generalized momentum-velocity relation between V, Ω and P, H. These linear connections build on the definition of two matrices: the *mass matrix* and the *flexibility matrix*, that are very important for the rest of the thesis.

The constitutive laws as used for example in [20, 36, 42] read as follows:

$$\begin{pmatrix} P \\ H \end{pmatrix} = \mathbf{M} \begin{pmatrix} V \\ \Omega \end{pmatrix}, \tag{3}$$

$$\begin{pmatrix} \gamma \\ \kappa \end{pmatrix} = \mathbf{F} \begin{pmatrix} \Theta \\ \Xi \end{pmatrix}, \tag{4}$$

where $\mathbf{M} = \mathbf{M}(x)$ is called *mass matrix* and $\mathbf{F} = \mathbf{F}(x)$ is called *flexibility matrix*. Both matrices have values in $\mathbb{R}^{6\times 6}$ and depend on the material properties of the considered beam. In general, i.e. for an anisotropic beam, the material properties are not constant along the beam. For an isotropic beam on the other hand, the material properties are constant and therefore, $\mathbf{M}$ and $\mathbf{F}$ do not depend on x in this special case.

The first constitutive law, (3), is derived by Hodges in [19], while the second one, (4), builds on Hooke's generalized law for elastic solids. A more detailed look into the latter constitutive law is given for example in [29, ch. 5].

Although, we will not go into detail regarding the derivation of the two matrices, we want to give a brief overview about their structure. Afterwards, we will make some important assumptions about both, the flexibility and the mass matrix, that will hold for the rest of this thesis. For the following considerations about the two matrices, we are guided by [20] and especially section 2.1.2 in [42].

At some points, it will be useful to subdivide $\mathbf{M}$ into four block matrices of the size 3×3. More specifically, the mass matrix can be written as

$$\mathbf{M} = \begin{pmatrix} M_1 & M_2 \\ M_2^T & M_3 \end{pmatrix} = \begin{pmatrix} \mu \mathbf{I}_{3,3} & -\mu\tilde{\zeta} \\ \mu\tilde{\zeta} & I \end{pmatrix}. \tag{5}$$

Here and in the rest of this thesis, $\mathbf{I}_{i,j}$ denotes the identity in $\mathbb{R}^{i\times j}$. Similarly $\mathbf{0}_{i,j}$ and $\mathbf{0}_i$ will denote the zero in $\mathbb{R}^{i\times j}$ and $\mathbb{R}^i$, respectively. Furthermore, $\mu = \mu(x) \in \mathbb{R}$ is the beam's mass per unit length, $\zeta = \zeta(x) \in \mathbb{R}^3$ is the mass center offset and by $I = I(x) \in \mathbb{R}^{3\times 3}$, we denote the cross sectional inertia matrix, having the form

$$I = \begin{pmatrix} I_{22} + I_{23} & 0 & 0 \\ 0 & I_{22} & I_{23} \\ 0 & I_{23} & I_{33} \end{pmatrix}.$$

Taking advantage of the skew symmetry of the cross product matrix, we see that, without further assumptions, it holds $\mathbf{M}^T = \mathbf{M}$ and, hence, the mass matrix is symmetric. If for example the considered beam is prismatic and isotropic, the mass center offset ζ is zero and the off diagonal entries of the inertia matrix, I_{23}, are zero as well, meaning that in this specific case, the mass matrix is not only symmetric but diagonal.

Similar to the mass matrix, the flexibility matrix can also be subdivided into four 3×3 blocks. The notation is as follows:

$$\mathbf{F} = \begin{pmatrix} F_1 & F_2 \\ F_2^T & F_3 \end{pmatrix}, \tag{6}$$

where $F_1 = F_1(x), F_2 = F_2(x), F_3 = F_3(x)$ are the cross sectional flexibilities of the beam, going back to the generalized law of Hook. The matrices F_1 and F_3 are symmetric, meaning that the flexibility matrix is symmetric as well. In the above mentioned case of a prismatic and isotropic beam, we have $F_2 = \mathbf{0}_{3,3}$ and the matrices F_1 and F_3 reduce to diagonal matrices, so that $\mathbf{F}$ is also diagonal in this case. In particular, the flexibility matrix for a prismatic and isotropic beam takes the form

$$\mathbf{F} = \operatorname{diag}\left(\frac{1}{aE}, \frac{1}{a\mathfrak{K}_2 G}, \frac{1}{a\mathfrak{K}_3 G}, \frac{1}{\mathfrak{K}_1 (I_{22} + I_{33}) G}, \frac{1}{I_{22}E}, \frac{1}{I_{33}E} \right).$$

In the above matrix, $a > 0$ is the beam's cross section area, $G > 0$ is its shear modulus, $E > 0$ is its Young modulus, $\mathfrak{K}_2, \mathfrak{K}_3 > 0$ are shear correction factors and $\mathfrak{K}_1 > 0$ corrects the polar moment of area. For a more detailed look into the flexibility matrix from a mechanical point of view we refer to [15, 23, 29].

For the rest of this thesis, we will assume that the mass matrix and the flexibility matrix are symmetric positive definite. Because this assumption is essential for the following sections, we want to give an overview if other authors use the same assumption. Rodriguez et al. make the same assumptions about the flexibility and mass matrix as we do in [42, 43, 44]. In fact, these are the sources we were guided by, making these assumptions. Nevertheless, in all of these three works, it is mentioned that the mass matrix may only be positive semi definite. In [2, 3], Artola et al. state that the flexibility matrix has values in the set of positive semi definite matrices while the mass matrix is always strictly positive definite. In [7, 15] and especially [9], on the other hand, the flexibility matrix is assumed to be strictly positive definite. In the latter one, the same mechanical assumptions as here are made as the authors consider an initially curved and twisted anisotropic beam.

While all authors seem to agree that the mass and flexibility matrices are at least positive semi definite, there are different statements about their strict definiteness. Nevertheless, we orientate ourselves on [42, 43, 44] and assume that both, the flexibility and the mass matrix are positive definite. Note that the assumptions regarding these two matrices are implicit assumptions about the material properties of the underlying beam. Therefore, in applications for real beams, one might want to check the validity of this assumption and if this is a restriction for the beam's shape or material properties.

2.3 Theory of Linear Hyperbolic Balance Laws

The goal in the next sections will be to reformulate the governing equations of the intrinsic beam model as a system of *linear hyperbolic balance laws*. Beforehand, we would like to give an overview about the fundamental ideas behind the theory of this class of equations. Hyperbolic balance laws in general are the extension of *hyperbolic conservation laws*. To get a better understanding of the ideas behind the theory of conservation laws, we will briefly derive a scalar hyperbolic conservation law based on an example, irrespective of the intrinsic beam model. Afterwards, we will build up the concept of scalar balance laws based on conservation laws. Eventually, we will extend these considerations to non scalar equations, i.e. systems of hyperbolic balance laws. This section is guided by Chapter 2 in [31].

Hyperbolic conservation laws are widely used in the field of fluid dynamics. Although the intrinsic beam model is not a part of this field, it makes sense to consider an example of fluid dynamics to illustratively derive the prototype of a hyperbolic conservation law and get a better understanding of the basic ideas behind it. Let us therefore consider a liquid flowing through a one dimensional pipe. The scalar $u = u(x,t)$ denotes the density of the fluid, depending on the position in space x and time t. The mass within a section of the pipe at a specific point in time t can now be expressed as

$$\int_{x_a}^{x_b} u(x,t)\,\mathrm{d}x$$

for any points $x_a \leq x_b$ that are located inside the pipe. Assuming that, within the pipe, the fluid is neither created nor destroyed, the total mass can only change due to flow through the endpoints of the section, x_a and x_b. Let $f = f(x,t)$ denote the rate at which the liquid flows past a point x at time t, where if $f(x,t) > 0$, we say that the flow is to the right and if $f(x,t) < 0$, we say the flow is to the left. The function f is called the *flux function*. The change of mass in time within the section $[x_a, x_b]$ is then given by the difference of flows through the boundaries. In other words:

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{x_a}^{x_b} u(x,t)\,\mathrm{d}x = f(x_a,t) - f(x_b,t).$$

For the class of equations that we want to consider, the flux function f actually does not only depend on x and t, but also on u itself. Therefore, we write $f = f(u(x,t))$. Thus, the above

equation, describing the change of mass, can be represented by

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{x_a}^{x_b} u(x,t)\,\mathrm{d}x = -f(u(x,t))\Big|_{x_a}^{x_b}, \tag{7}$$

which, if we assume that u and f are sufficiently smooth, can be transformed to

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{x_a}^{x_b} u(x,t) = -\int_{x_a}^{x_b} \frac{\partial}{\partial x} f(u(x,t))\,\mathrm{d}x$$

or equivalently

$$\int_{x_a}^{x_b} \left(\frac{\partial}{\partial t}u(x,t) + \frac{\partial}{\partial x}f(u(x,t))\right)\,\mathrm{d}x = 0.$$

Recall that the above equality does hold for arbitrary $x_a \leq x_b$ within the pipe. To fulfill this, the integrand of the above integral on the left hand side has to be zero, which leads to

$$\frac{\partial}{\partial t}u(x,t) + \frac{\partial}{\partial x}f(u(x,t)) = 0. \tag{8}$$

Equation (8) is the prototype of a scalar hyperbolic conservation law. The flux function f can take many shapes. Let us for example suppose that the fluid flows through the pipe at a constant velocity $A \in \mathbb{R}$. Then the flux function f, that describes this flow, reads

$$f(u(x,t)) = Au(x,t).$$

This can be extended to a more general form, where the coefficient A is not constant, but depends on the position x. The function f then reads

$$f(x,u) = A(x)u(x,t). \tag{9}$$

An equation of type (8) with a flux of the form (9) is in general called a *linear* scalar hyperbolic conservation law, or a *linear advection equation.*

A linear advection equation as defined above describes the evolution of a conservative variable u. However, there are many use cases that are generally similar to the above considerations but include an additional source or sink. In the above example of the fluid flowing through a pipe, this would mean that the mass within a section of the pipe would not only change due to flows through the section's boundaries, but also due to the source or sink. Mathematically, such a setup can be modelled by extending the conservation law by a source term Q_{cons}, that can function as a source as well as a sink depending on its sign. For now, we assume that $Q_{cons} = Q_{cons}(x,t,u)$ can be an arbitrary function. The solution u of a conservation law that is extended by such a source term is no longer a conservative variable. Therefore, the resulting equation is not longer a conservation law but is called a hyperbolic *balance law* and formally reads as follows:

$$\frac{\partial}{\partial t}u(x,t) + \frac{\partial}{\partial x}f(u(x,t)) = Q_{cons}(x,t,u).$$

With the linear flux function, defined above, the balance law reads

$$\frac{\partial}{\partial t}u(x,t) + \frac{\partial}{\partial x}(A(x)u(x,t)) = Q_{cons}(x,t,u) \tag{10}$$

and is called linear hyperbolic balance law.

The above considerations can be transferred from scalar quantities to vector valued quantities. The scalar equation then becomes a system of equations, where u and $Q_{cons}(x,t,u)$ are vectors, and A is a matrix. In this case, A is called the *advection matrix.* For the rest of the thesis, we interpret equation (10) as a system of linear balance laws, to which we refer as *linear advection equation with source term* or simply *linear advection equation.* Furthermore, we use a more convenient notation for the partial derivatives, and omit the dependencies on x and t, writing

$$u_t + (Au)_x = Q_{cons}(u). \tag{11}$$

The system (11) is called *hyperbolic*, if the advection matrix A is diagonalizable and its eigenvalues are real. Note, that the advection matrix may depend on x and therefore A has to be diagonalizable at every point in space x.

A linear advection equation may have a different representation that is equivalent to (11). All of the following formulations can be found in the literature, namely for example [31, 50]. If we assume for example, that the advection matrix A is differentiable with respect to the spatial variable x, we can write (11) equivalently as

$$u_t + Au_x = Q_{cons}(u) - A_x u. \tag{12}$$

Equation (11) is called the *conservation form*, while equation (12) is called the *advection form* of the linear advection equation. In the latter formulation, the term $-A_x u$ can be interpreted as an additional source term. If we further define $Q_{adv}(u) := Q_{cons}(u) - A_x u$, the advection form reads

$$u_t + Au_x = Q_{adv}(u). \tag{13}$$

Another equivalent representation of the same equation is obtained by inserting the diagonalization of the matrix A with the corresponding diagonal matrix $\mathbf{\Lambda}$ into the conservation form. Remember that by definition, the matrix A, belonging to a hyperbolic balance law, is diagonalizable. It can be shown that an equivalent formulation can be found and reads

$$w_t + (\mathbf{\Lambda} w)_x = Q_{char}(w) \tag{14}$$

for so called *characteristic variables*, w. In section 2.4.3, we will describe the process to obtain the last representation in detail, including the definition of w and the derivation of Q_{char}.

The last representation of the linear advection equation, we want to consider, is obtained by multiplying both sides of the equation in advection form, (13), by a matrix $\Gamma = \Gamma(x)$. Assuming that Γ is an invertible matrix at every point x in space, the formulation

$$\Gamma u_t + \Pi u_x = Q_{cap}(u) \tag{15}$$

with $\Pi = \Gamma A$ and $Q_{cap}(u) := \Gamma Q_{adv}(u)$, is also an equivalent representation of the linear advection equation. For the rest of this thesis, we refer to an equation of the form (15) as *capacity form* of the linear advection equation. This is not a term that is used in the literature but – inspired by the name *capacity function* in front of the time derivative that is used in the literature [31] – we use the term capacity form to have a clear name for this formulation. This completes the general considerations on the theory of linear hyperbolic balance laws. In the next sections, we will resume investigating the intrinsic beam model.

2.4 Transfer to a Linear Hyperbolic Balance Law

In this section, we will transfer the governing equations of the intrinsic beam model to a hyperbolic system of balance laws, namely a linear advection equation with source term. In the previous section, we have seen that there are different representations of such a linear advection equation. In the following subsections, we will derive some of these representations for the intrinsic beam equation, namely the advection form, the capacity form and the formulation for characteristic variables with a diagonal advection matrix. Furthermore, we will show that the intrinsic beam equation is a hyperbolic equation in the sense that the advection matrix can be diagonalized and has real eigenvalues, only.

2.4.1 Formulation in Advection Form

In the following, we will reformulate the governing equations of the intrinsic beam model into a linear advection equation in advection form. First, we will write the system of PDEs (2) in matrix vector notation and afterwards use the constitutive laws to close the formulation of the intrinsic beam equation in its advection form. A similar presentation of the intrinsic beam equations is presented in the work of Rodriguez et al. [42, 43, 44].

We start with the classical form of the intrinsic beam equations that was presented in section 2.1 and is given by

$$\begin{aligned}
\Theta' + (\tilde{k} + \tilde{\kappa})\Theta + f_{ext} &= \dot{P} + \tilde{\Omega}P \\
\Xi' + (\tilde{k} + \tilde{\kappa})\Xi + (\tilde{e_1} + \tilde{\gamma})\Theta + m_{ext} &= \dot{H} + \tilde{\Omega}H + \tilde{V}P \\
V' + (\tilde{k} + \tilde{\kappa})V + (\tilde{e_1} + \tilde{\gamma})\Omega &= \dot{\gamma} \\
\Omega' + (\tilde{k} + \tilde{\kappa})\Omega &= \dot{\kappa}.
\end{aligned}$$

First, we rewrite this system of equations into a matrix-vector formulation:

$$\begin{aligned}
&\begin{pmatrix} \Theta \\ \Xi \\ V \\ \Omega \end{pmatrix}_x + \begin{pmatrix} \tilde{k} + \tilde{\kappa} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \tilde{e_1} + \tilde{\gamma} & \tilde{k} + \tilde{\kappa} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \tilde{k} + \tilde{\kappa} & \tilde{e_1} + \tilde{\gamma} \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \tilde{k} + \tilde{\kappa} \end{pmatrix} \begin{pmatrix} \Theta \\ \Xi \\ V \\ \Omega \end{pmatrix} \\
&\quad - \begin{pmatrix} P \\ H \\ \gamma \\ \kappa \end{pmatrix}_t - \begin{pmatrix} \tilde{\Omega} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \tilde{V} & \tilde{\Omega} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \end{pmatrix} \begin{pmatrix} P \\ H \\ \gamma \\ \kappa \end{pmatrix} + \begin{pmatrix} f_{ext} \\ m_{ext} \\ \mathbf{0}_3 \\ \mathbf{0}_3 \end{pmatrix} = \mathbf{0}_{12}.
\end{aligned} \tag{16}$$

Next, we make use of the constitutive laws that can equivalently to equations (3) and (4) be written as

$$\begin{pmatrix} P \\ H \\ \gamma \\ \kappa \end{pmatrix} = \begin{pmatrix} \mathbf{0}_{6,6} & \mathbf{M} \\ \mathbf{F} & \mathbf{0}_{6,6} \end{pmatrix} \begin{pmatrix} \Theta \\ \Xi \\ V \\ \Omega \end{pmatrix}.$$

By inserting the constitutive laws in the above form into the equation in its matrix-vector formulation, (16), rearranging terms and setting $u := (\Theta^T, \Xi^T, V^T, \Omega^T)^T$, we obtain

$$\begin{aligned}
&- \begin{pmatrix} \mathbf{0}_{6,6} & \mathbf{M} \\ \mathbf{F} & \mathbf{0}_{6,6} \end{pmatrix} u_t + u_x + \left(\begin{pmatrix} \tilde{k} + \tilde{\kappa} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \tilde{e_1} + \tilde{\gamma} & \tilde{k} + \tilde{\kappa} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \tilde{k} + \tilde{\kappa} & \tilde{e_1} + \tilde{\gamma} \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \tilde{k} + \tilde{\kappa} \end{pmatrix} \right. \\
&\qquad \left. - \begin{pmatrix} \tilde{\Omega} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \tilde{V} & \tilde{\Omega} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \end{pmatrix} \begin{pmatrix} \mathbf{0}_{6,6} & \mathbf{M} \\ \mathbf{F} & \mathbf{0}_{6,6} \end{pmatrix} \right) u + \begin{pmatrix} f_{ext} \\ m_{ext} \\ \mathbf{0}_3 \\ \mathbf{0}_3 \end{pmatrix} = \mathbf{0}_{12}.
\end{aligned}$$

Defining the matrix $A = A(x)$ with values in $\mathbb{R}^{12\times 12}$ as

$$A = - \begin{pmatrix} \mathbf{0}_{6,6} & \mathbf{M} \\ \mathbf{F} & \mathbf{0}_{6,6} \end{pmatrix}^{-1} = - \begin{pmatrix} \mathbf{0}_{6,6} & \mathbf{F}^{-1} \\ \mathbf{M}^{-1} & \mathbf{0}_{6,6} \end{pmatrix}, \tag{17}$$

and multiplying the last equation by A from the left, yields

$$\begin{aligned}
&u_t + Au_x + A \left(\begin{pmatrix} \tilde{k} + \tilde{\kappa} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \tilde{e_1} + \tilde{\gamma} & \tilde{k} + \tilde{\kappa} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \tilde{k} + \tilde{\kappa} & \tilde{e_1} + \tilde{\gamma} \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \tilde{k} + \tilde{\kappa} \end{pmatrix} \right. \\
&\qquad \left. + \begin{pmatrix} \tilde{\Omega} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \tilde{V} & \tilde{\Omega} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \end{pmatrix} A^{-1} \right) u + A \begin{pmatrix} f_{ext} \\ m_{ext} \\ \mathbf{0}_3 \\ \mathbf{0}_3 \end{pmatrix} = \mathbf{0}_{12}.
\end{aligned} \tag{18}$$

Note that the matrix A is well defined because by assumption, the flexibility and mass matrix are positive definite and therefore invertible. To point out the analogy to a linear advection equation

with source term, as it was presented in the previous section, we define the source term Q_{adv} as follows:

$$Q_{adv}(u) := -A \left(\begin{pmatrix} \tilde{k}+\tilde{\kappa} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \tilde{e_1}+\tilde{\gamma} & \tilde{k}+\tilde{\kappa} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \tilde{k}+\tilde{\kappa} & \tilde{e_1}+\tilde{\gamma} \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \tilde{k}+\tilde{\kappa} \end{pmatrix} + \begin{pmatrix} \tilde{\Omega} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \tilde{V} & \tilde{\Omega} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \end{pmatrix} A^{-1} \right) u - A \begin{pmatrix} f_{ext} \\ m_{ext} \\ \mathbf{0}_3 \\ \mathbf{0}_3 \end{pmatrix}$$

Then, equation (18) can be written compactly as

$$u_t + Au_x = Q_{adv}(u), \tag{19}$$

which is the advection form of a linear advection equation just as we defined it in (13).

2.4.2 Hyperbolicity of the Equation

In the following section, we will show that the linear advection equation representing the intrinsic beam equations is hyperbolic. Recall, that a linear advection equation of the form we are considering here is hyperbolic, if the advection matrix A can be diagonalized and has real eigenvalues only. We will constructively derive a transformation matrix, that diagonalizes the matrix A, and will show that the corresponding eigenvalues are real. The diagonalization is inspired by [44] and adjusted to the notation in this thesis.

We start by introducing a notation for the square root of a matrix. First, we note that for every positive definite matrix $\hat{A}$, there exists a positive definite matrix $\hat{B}$ with $\hat{B}^2 = \hat{A}$ (cf. for example [21]). Analogous to real numbers, we call $\hat{B}$ the square root of the matrix $\hat{A}$ and denote this by $\hat{A}^{1/2} = \hat{B}$. Note that the square root of a matrix is invertible because, by definition it is positive definite. We denote the inverse of a square root matrix by $(\hat{A}^{1/2})^{-1} = \hat{A}^{-1/2}$. Now recall that the matrix $\mathbf{F}$ is positive definite by assumption and therefore has a square root. Using the above notation, we define the matrix $\Psi = \Psi(x)$ with values in $\mathbb{R}^{6\times 6}$ as follows:

$$\Psi := \mathbf{F}^{-1/2}\,\mathbf{M}^{-1}\,\mathbf{F}^{-1/2}. \tag{20}$$

One can easily check, that Ψ is symmetric. Furthermore, as a consequence of the positive definiteness of $\mathbf{F}$ and $\mathbf{M}$, the matrix Ψ is positive definite itself. This is because for any non-zero $z \in \mathbb{R}^6$, we have

$$z^T\Psi z = z^T\mathbf{F}^{-1/2}\,\mathbf{M}^{-1}\,\mathbf{F}^{-1/2}z = (\mathbf{F}^{-1/2}z)^T\mathbf{M}^{-1}(\mathbf{F}^{-1/2}z) > 0.$$

Let us denote the eigenvalues of Ψ by $\lambda_i = \lambda_i(x)$ for $i = 1, \dots, 6$. Moreover, we define the diagonal matrix $\Lambda = \Lambda(x)$ by

$$\Lambda = \operatorname{diag}\left(\lambda_1^{1/2}, \dots, \lambda_6^{1/2}\right).$$

As Ψ is positive definite, all of its eigenvalues are positive, meaning that Λ is in fact a real diagonal matrix. The positive definiteness of Ψ also implicates that it can be diagonalized. More precisely, there exists an orthogonal matrix $\mathcal{X} = \mathcal{X}(x)$ with values in $\mathbb{R}^{6\times 6}$, such that

$$\Psi = \mathcal{X}^T\Lambda^2\,\mathcal{X} \tag{21}$$

holds.

Based on these considerations regarding the matrix Ψ and its eigenvalues, we will now specify a transformation matrix T and its inverse, from which we will show, it diagonalizes the matrix A. In particular, the matrices $T = T(x), T^{-1} = T^{-1}(x)$ with values in $\mathbb{R}^{12\times 12}$ are given by

$$T := \frac{1}{2}\begin{pmatrix} \mathbf{F}^{-1/2}\chi^T & \mathbf{F}^{-1/2}\mathcal{X}^T \\ \mathbf{F}^{1/2}\mathcal{X}^T\Lambda & -\mathbf{F}^{1/2}\mathcal{X}^T\Lambda \end{pmatrix}, \qquad T^{-1} = \begin{pmatrix} \mathcal{X}\mathbf{F}^{1/2} & \Lambda^{-1}\mathcal{X}\mathbf{F}^{-1/2} \\ \mathcal{X}\mathbf{F}^{1/2} & -\Lambda^{-1}\mathcal{X}\mathbf{F}^{-1/2} \end{pmatrix}.$$

Let us additionally define a 12×12 diagonal matrix, $\mathbf{\Lambda} = \mathbf{\Lambda}(x)$, in block form, consisting of Λ and $-\Lambda$ on its diagonal:

$$\mathbf{\Lambda} := \begin{pmatrix} -\Lambda & \mathbf{0}_{6,6} \\ \mathbf{0}_{6,6} & \Lambda \end{pmatrix}.$$

All in all, this results in the following transformation:

$$T\mathbf{\Lambda}T^{-1} = \frac{1}{2}\begin{pmatrix} \mathbf{0}_{6,6} & -2\mathbf{F}^{-1} \\ -2\mathbf{F}^{1/2}\mathcal{X}^T\Lambda^2\mathcal{X}\mathbf{F}^{1/2} & \mathbf{0}_{6,6} \end{pmatrix}.$$

Here, we use that the matrix Ψ is diagoanlized by a transformation with $\mathcal{X}$ as in (21) to obtain

$$T\mathbf{\Lambda}T^{-1} = \begin{pmatrix} \mathbf{0}_{6,6} & -\mathbf{F}^{-1} \\ -\mathbf{F}^{1/2}\Psi\mathbf{F}^{1/2} & \mathbf{0}_{6,6} \end{pmatrix}.$$

Finally, we use the definition of Ψ in (20) and solve it for $\mathbf{M}^{-1}$ to see that

$$T\mathbf{\Lambda}T^{-1} = \begin{pmatrix} \mathbf{0}_{6,6} & -\mathbf{F}^{-1} \\ -\mathbf{M}^{-1} & \mathbf{0}_{6,6} \end{pmatrix} = A. \tag{22}$$

We thereby showed that the matrix A is diagonalized by a transformation with T. In particular, the diagonalization reads

$$\mathbf{\Lambda} = T^{-1}AT. \tag{23}$$

Note that due to the definition of $\mathbf{\Lambda}$ this implicates that the eigenvalues of A are the square roots of the eigenvalues of Ψ and their negatives, as these are the entries on the diagonal of $\mathbf{\Lambda}$. The matrix A, therefore, has real eigenvalues only, meaning that the linear advection equation we derived in the previous section is hyperbolic, indeed.

Moreover, Ψ and $(\mathbf{FM})^{-1}$ have the same eigenvalues as they are similar matrices. The latter statement holds because we have that

$$(\mathbf{FM})^{-1} = \mathbf{F}^{1/2}\Psi\mathbf{F}^{-1/2}.$$

In general both, the flexibility and the mass matrix depend on the spatial variable x and, thus, also the eigenvalues of Ψ and therefore the ones of A depend on x. Nevertheless, the above considerations show that this does not have an impact on the sign of the eigenvalues. The first six eigenvalues of A will be negative for any x, while the last six eigenvalues will be positive. The realization of this property will be very important for the derivation of boundary conditions in section 2.5.

Remark 1 *As mentioned in the previous section, if an isotropic prismatic beam is considered, the flexibility and the mass matrix are diagonal. In that case, the matrix $\mathcal{X}$ is the identity matrix, which results in the following simplification of the transformation matrix and its inverse:*

$$T = \frac{1}{2}\begin{pmatrix} \mathbf{F}^{-1/2} & \mathbf{F}^{-1/2} \\ \mathbf{M}^{-1/2} & -\mathbf{M}^{-1/2} \end{pmatrix}, \qquad T^{-1} = \begin{pmatrix} \mathbf{F}^{1/2} & \mathbf{M}^{1/2} \\ \mathbf{F}^{1/2} & -\mathbf{M}^{1/2} \end{pmatrix}.$$

2.4.3 Formulation for Characteristic Variables

Regarding some parts of the analysis of linear advection equations, especially the derivation of boundary conditions, it is often useful to consider an equivalent formulation that includes the diagonal advection matrix $\mathbf{\Lambda}$. To find such a formulation, we define the so called *characteristic variables*

$$w := T^{-1}u.$$

In contrast to that, we call u *physical variables* in the following as it contains the physical quantities Θ, Ξ, V and Ω. Now, inserting $u = Tw$ into the original advection form of the intrinsic beam equation, namely (19), gives us the equivalent equation

$$Tw_t + A(Tw)_x = Q_{adv}(Tw).$$

In the previous section, we learned about the transformation (22) of the matrix A with T, that leads to the diagonal matrix $\mathbf{\Lambda}$. This particular transformation is now used to represent A in the above equation leading us to

$$Tw_t + T\mathbf{\Lambda}T^{-1}(Tw)_x = Q_{adv}(Tw).$$

The derivative in the second summand of the above expression can be calculated by applying the product rule to it. Explicitly, this results in

$$Tw_t + T\mathbf{\Lambda}w_x + T\mathbf{\Lambda}T^{-1}T_x w = Q_{adv}(Tw).$$

Multiplying the last result with the inverse transformation matrix, T^{-1}, from the left, then results in

$$w_t + \mathbf{\Lambda}w_x = Q_{char}(w),$$

where we define the source term

$$Q_{char}(w) := T^{-1}Q_{adv}(Tw) - \mathbf{\Lambda}T^{-1}T_x w.$$

The advantage of this representation is that the so called *characteristic speeds* explicitly appear on the diagonal of the matrix $\mathbf{\Lambda}$. As we mentioned initially in this section, this is especially helpful when deriving appropriate boundary conditions for the system. In section 2.5, we will give a more detailed description of the concept of characteristic speeds. Beforehand we will derive another equivalent representation of the intrinsic beam equation in the next section.

2.4.4 Formulation in Capacity Form

In this section, we would like to deduce a capacity form of the considered linear advection equation as we introduced it in section 2.3. Let therefore $\Gamma = \Gamma(x)$ be the matrix with values in $\mathbb{R}^{12\times 12}$ defined by

$$\Gamma := \begin{pmatrix} \mathbf{F} & \mathbf{0}_{6,6} \\ \mathbf{0}_{6,6} & \mathbf{M} \end{pmatrix}$$

and $\Pi \in \mathbb{R}^{12\times 12}$ be the matrix defined by the matrix product of Γ and A, that reads

$$\Pi := \Gamma A = -\begin{pmatrix} \mathbf{F} & \mathbf{0}_{6,6} \\ \mathbf{0}_{6,6} & \mathbf{M} \end{pmatrix} \begin{pmatrix} \mathbf{0}_{6,6} & \mathbf{F}^{-1} \\ \mathbf{M}^{-1} & \mathbf{0}_{6,6} \end{pmatrix} = -\begin{pmatrix} \mathbf{0}_{6,6} & \mathbf{I}_{6,6} \\ \mathbf{I}_{6,6} & \mathbf{I}_{6,6} \end{pmatrix}.$$

Note that, as an implication of the symmetry and positive definiteness of the flexibility and the mass matrix, the matrix Γ is likewise symmetric, positive definite. Furthermore, the matrix Π is constant and symmetric. These properties of the two matrices Γ and Π will be very helpful when deriving energy stability, as we will see in section 2.6.

Similar to the definition of Π, we define the source term Q_{cap} as the product of the source term Q_{adv} with the matrix Γ from the left:

$$Q_{cap}(u) := \Gamma Q_{adv}(u) = -\Gamma A \left(\begin{pmatrix} \tilde{k}+\tilde{\kappa} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \tilde{e_1}+\tilde{\gamma} & \tilde{k}+\tilde{\kappa} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \tilde{k}+\tilde{\kappa} & \tilde{e_1}+\tilde{\gamma} \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \tilde{k}+\tilde{\kappa} \end{pmatrix} + \begin{pmatrix} \tilde{\Omega} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \tilde{V} & \tilde{\Omega} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \end{pmatrix} A^{-1} \right) u - \Gamma A \begin{pmatrix} f_{ext} \\ m_{ext} \\ \mathbf{0}_3 \\ \mathbf{0}_3 \end{pmatrix}.$$

In the above term, we again find the matrix product of Γ and A, which we defined as Π earlier, so

that the term Q_{cap} can be written as

$$Q_{cap}(u) = -\Pi \left(\begin{pmatrix} \tilde{k}+\tilde{\kappa} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \tilde{e_1}+\tilde{\gamma} & \tilde{k}+\tilde{\kappa} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \tilde{k}+\tilde{\kappa} & \tilde{e_1}+\tilde{\gamma} \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \tilde{k}+\tilde{\kappa} \end{pmatrix} \right.$$
$$\left. + \begin{pmatrix} \tilde{\Omega} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \tilde{V} & \tilde{\Omega} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \end{pmatrix} A^{-1} \right) u - \Pi \begin{pmatrix} f_{ext} \\ m_{ext} \\ \mathbf{0}_3 \\ \mathbf{0}_3 \end{pmatrix}.$$

By multiplying the advection form of the equation, (19), with Γ from the left and using the above definitions of Π and Q_{cap}, we obtain the capacity form of the linear advection equation:

$$\Gamma u_t + \Pi u_x = Q_{cap}(u). \tag{24}$$

Comparing (24) to the definition (15) of general capacity forms for linear advection equations, we indeed have a capacity form of the linear advection equation for the intrinsic beam model. The positive definite matrix Γ represents the capacity function.

In the analysis of the solution's energy in section 2.6, the source term Q_{cap} will play an important role. To prepare this analysis, we take a more detailed look at the source term and bring it in a certain form, which will be helpful. Due to its special shape, the multiplications with the matrix Π in Q_{cap} can be seen as a rearrangement of 6×6 blocks, which in particular yields

$$Q_{cap}(u) = \left(\begin{pmatrix} \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \tilde{k}+\tilde{\kappa} & \tilde{e_1}+\tilde{\gamma} \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \tilde{k}+\tilde{\kappa} \\ \tilde{k}+\tilde{\kappa} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \tilde{e_1}+\tilde{\gamma} & \tilde{k}+\tilde{\kappa} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \end{pmatrix} \right.$$
$$\left. + \begin{pmatrix} \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \tilde{\Omega} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \tilde{V} & \tilde{\Omega} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \end{pmatrix} A^{-1} \right) u + \begin{pmatrix} \mathbf{0}_3 \\ \mathbf{0}_3 \\ f_{ext} \\ m_{ext} \end{pmatrix}.$$

The first matrix in the above expression can be split up into the sum of two matrices. One containing the cross product matrices of the initial curvature k and the vector e_1, the other containing the cross product matrices of the strains κ and γ. This results in

$$\begin{aligned} Q_{cap}(u) = & \begin{pmatrix} \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \tilde{k} & \tilde{e_1} \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \tilde{k} \\ \tilde{k} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \tilde{e_1} & \tilde{k} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \end{pmatrix} u + \left(\begin{pmatrix} \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \tilde{\kappa} & \tilde{\gamma} \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \tilde{\kappa} \\ \tilde{\kappa} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \tilde{\gamma} & \tilde{\kappa} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \end{pmatrix} \right. \\ & \left. + \begin{pmatrix} \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \tilde{\Omega} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \tilde{V} & \tilde{\Omega} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \end{pmatrix} A^{-1} \right) u + \begin{pmatrix} \mathbf{0}_3 \\ \mathbf{0}_3 \\ f_{ext} \\ m_{ext} \end{pmatrix}. \end{aligned} \tag{25}$$

We recall that κ and γ can be represented in terms of Θ and Ξ by applying the constitutive law

$$\begin{pmatrix} \gamma \\ \kappa \end{pmatrix} = \begin{pmatrix} F_1 & F_2 \\ F_2^T & F_3 \end{pmatrix} \begin{pmatrix} \Theta \\ \Xi \end{pmatrix}. \tag{26}$$

As $u = (\Theta^T, \Xi^T, V^T, \Omega^T)$, there occur non-linearities of u in Q_{cap}. The degree of non-linearity is in fact two, as it is already pointed out in [20] for the classical formulation of the intrinsic beam equations. The definition of the source term in (25) can, thus, be subdivided into a term that is linear in u, namely the first summand on the right hand side of (25), a term that is quadratic in u, namely the product of the expression in big brackets and u, and a term that does not depend on u but only on external forces and moments.

According to that separation depending on the degree in u, we write the source term as follows

$$Q_{cap}(u) = Bu + J(u)u + q_{ext},$$

where we define $B = B(x), J(u) = J(u,x)$ and $q_{ext} = q_{ext}(x,t)$ as

$$B := \begin{pmatrix} \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \tilde{k} & \tilde{e_1} \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \tilde{k} \\ \tilde{k} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \tilde{e_1} & \tilde{k} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \end{pmatrix},$$

$$J(u) := \begin{pmatrix} \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \tilde{\kappa} & \tilde{\gamma} \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \tilde{\kappa} \\ \tilde{\kappa} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \tilde{\gamma} & \tilde{\kappa} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \end{pmatrix} + \begin{pmatrix} \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \tilde{\Omega} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \tilde{V} & \tilde{\Omega} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \end{pmatrix} A^{-1},$$

$$q_{ext} := \begin{pmatrix} \mathbf{0}_3 \\ \mathbf{0}_3 \\ f_{ext} \\ m_{ext} \end{pmatrix}.$$

The matrix B can be brought into a more compact representation that will also be useful for the energy analysis. Therefore, we define the matrix $\mathcal{B} = \mathcal{B}(x)$ as

$$\mathcal{B} := \begin{pmatrix} \tilde{k} & \tilde{e_1} \\ \mathbf{0}_{3,3} & \tilde{k} \end{pmatrix}.$$

As an implication of the skew symmetry of the cross product matrices, the matrix B can then be represented by

$$B := \begin{pmatrix} \mathbf{0}_{6,6} & \mathcal{B} \\ -\mathcal{B}^T & \mathbf{0}_{6,6} \end{pmatrix}.$$

Recall, that the matrix A^{-1}, in terms of 6×6 blocks, can be written as

$$A^{-1} = \begin{pmatrix} \mathbf{0}_{6,6} & \mathbf{M} \\ \mathbf{F} & \mathbf{0}_{6,6} \end{pmatrix}$$

(cf. (17)) and that the flexibility and mass matrix can each be subdivided into four 3×3 blocks. The representation of A^{-1} in terms of 3×3 blocks, therefore becomes

$$A^{-1} = \begin{pmatrix} \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & M_1 & M_2 \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & M_2^T & M_3 \\ F_1 & F_2 & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ F_2^T & F_3 & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \end{pmatrix}.$$

Inserting this into the definition of $J(u)$ yields

$$J(u) = \begin{pmatrix} \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \tilde{\kappa} & \tilde{\gamma} \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \tilde{\kappa} \\ \tilde{\kappa} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \tilde{\gamma} & \tilde{\kappa} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \end{pmatrix} - \begin{pmatrix} \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \tilde{\Omega} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \tilde{V} & \tilde{\Omega} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \end{pmatrix} \begin{pmatrix} \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & M_1 & M_2 \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & M_2^T & M_3 \\ F_1 & F_2 & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ F_2^T & F_3 & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \end{pmatrix}.$$

After executing the matrix multiplication in the above second summand, this term reads

$$J(u) = \begin{pmatrix} \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \tilde{\kappa} & \tilde{\gamma} \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \tilde{\kappa} \\ \tilde{\kappa} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \tilde{\gamma} & \tilde{\kappa} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \end{pmatrix} - \begin{pmatrix} \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \tilde{\Omega} M_1 & \tilde{\Omega} M_2 \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \tilde{V} M_1 + \tilde{\Omega} M_2^T & \tilde{V} M_2 + \tilde{\Omega} M_3 \end{pmatrix}$$

To consistently operate in the variables Θ, Ξ, V and Ω, we make again use of the constitutive law (26), to replace κ and γ. The final representation of the term $J(u)$ then reads

$$J(u) = \begin{pmatrix} \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \widetilde{F_2^T\Theta + F_3\Xi} & \widetilde{F_1\Theta + F_2\Xi} \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \widetilde{F_2^T\Theta + F_3\Xi} \\ \widetilde{F_2^T\Theta + F_3\Xi} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \widetilde{F_1\Theta + F_2\Xi} & \widetilde{F_2^T\Theta + F_3\Xi} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \end{pmatrix} - \begin{pmatrix} \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \tilde{\Omega}M_1 & \tilde{\Omega}M_2 \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \tilde{V}M_1 + \tilde{\Omega}M_2^T & \tilde{V}M_2 + \tilde{\Omega}M_3 \end{pmatrix}.$$

2.5 Boundary Conditions

In the course of this section, we will derive and discuss boundary conditions for the intrinsic beam model. In the literature, namely for example [36, 49], a beam that is clamped at one end and free swinging at the other, is modeled by prescribing boundary values for internal forces and moments at the free swinging end and for linear and angular velocities at the clamped end. According to [49], boundary values at both ends can be zero or non-zero. A follower force at the tip of the beam is for example modeled by prescribing non-zero internal forces at the free swinging end. A rotating beam could be modeled by prescribing angular velocities at its clamped end.

Nevertheless, it is not necessarily ensured that these type of boundary conditions lead to a mathematically well posed problem. We will derive mathematically appropriate boundary conditions for the intrinsic beam equation. This will be done by transferring the results of Nordström in [34], Nordström and Wahlsten in [35] and Russell in chapter 3 of [45], concerning boundary conditions for general linear hyperbolic systems, to the intrinsic beam equation. Afterwards, we will show that this general class of boundary conditions can be used to obtain the boundary conditions that describe a clamped beam.

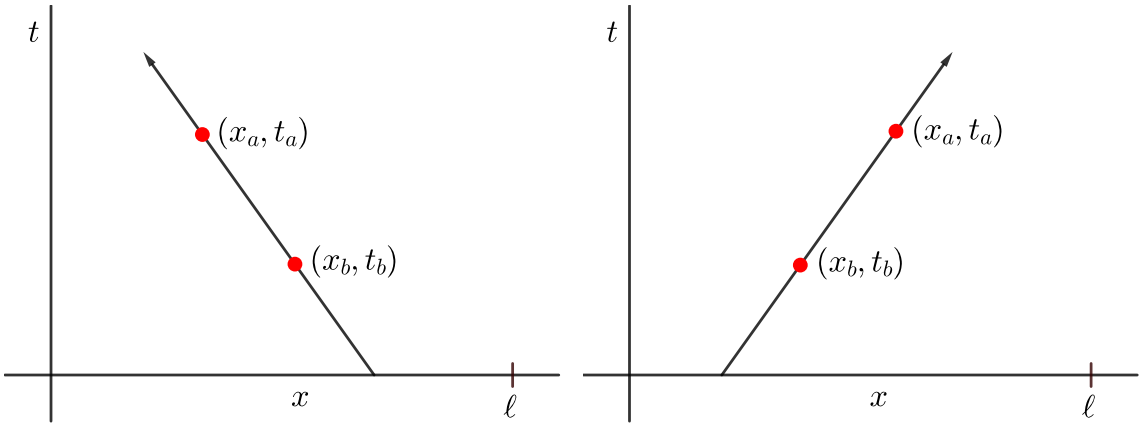
Before we start discussing the boundary conditions, we would like to give a brief overview about the concept of *characteristic curves*. Detailed information about the corresponding theory can be read up in [31, 50]. Roughly spoken, characteristic curves are curves in the x-t-plane on which the solution of a conservation law takes constant values. Thus, characteristic curves are a way of describing how the information of the initial condition of the problem propagates through the considered domain $[0, \ell]$ in time. *Characteristic speeds* are the gradients of characteristic curves and can therefore be interpreted as the speeds at which information is transported through the domain.

To illustrate this concept, in figure 1 two simple examples of characteristic curves for a scalar conservation law, irrespective of the intrinsic beam equation, are given. Figure 1a shows a straight line characteristic curve with a negative slope, i.e. a negative characteristic speed. Figure 1b shows a straight line characteristic curve with a positive slope. For any two points on the characteristic curve with the negative slope, (x_a, t_a), (x_b, t_b), with $t_a > t_b$, one has $x_a < x_b$, meaning that the more time passes, the further to the left of the domain the value of the corresponding PDE solution is transported. For the characteristic curve with a positive slope on the other hand, for any two points (x_a, t_a), (x_b, t_b), with $t_a > t_b$, one has $x_a > x_b$, meaning that in this case, the value of the corresponding PDE solution is transported to the right of the domain as time passes.

To summarize the above considerations: the sign of the slope of the characteristic curves and thus, the sign of the characteristic speed, indicates whether information of the solution propagates from left to right or from right to left within the domain $[0, \ell]$. Consequently, boundary conditions for the solution of such a PDE have to be specified at the right boundary in case of a negative characteristic speed and at the left boundary in case of a positive characteristic speed.

This can be transferred to non scalar linear balance laws, such as the intrinsic beam equation. In this case, the characteristic speeds are the eigenvalues of the advection matrix A, i.e. the diagonal entries of the matrix $\mathbf{\Lambda}$. In the formulation for characteristic variables, this means that the sign of the i-th entry in $\mathbf{\Lambda}$, namely $\mathbf{\Lambda}_{ii}$, indicates whether information in the i-th component of characteristic variables, w, is transported leftwards or rightwards. Hence, boundary conditions for characteristic variables associated with negative diagonal entries in $\mathbf{\Lambda}$, have to be specified at $x = \ell$. For characteristic variables associated with positive diagonal entries in $\mathbf{\Lambda}$, boundary

conditions have to be specified at $x = 0$. At the respective boundary, the components of w for which we need to specify boundary conditions, are also called *ingoing variables*. The components for which we do not need to specify boundary conditions, are called *outgoing variables* at the respective boundary.



(a) Characteristic curve with negative characteristic speed.

(b) Characteristic curve with positive characteristic speed.

Figure 1: Examples for characteristic curves of a scalar conservation law (cf. [50, Fig. 2.1.]).

The matrix $\mathbf{\Lambda}$, derived in section 2.4.2, containing the characteristic speeds for the intrinsic beam model, has the form

$$\mathbf{\Lambda} = \begin{pmatrix} -\Lambda & \mathbf{0}_{6,6} \\ \mathbf{0}_{6,6} & \Lambda \end{pmatrix},$$

with exclusively negative values in the upper six entries of its diagonal and positive ones in the lower six diagonal entries. According to that, we introduce a new notation to point out which components of w are associated with negative characteristic speeds and which ones are associated with positive characteristic speeds. The twelve dimensional vector of characteristic variables, w, is therefore subdivided into two six dimensional vectors w_- and w_+, where the subscript indicates the sign of the associated characteristic speed. The characteristic variables according to the entries of $\mathbf{\Lambda}$ are then

$$w = \begin{pmatrix} w_- \\ w_+ \end{pmatrix}.$$

At $x = 0$, this means that w_+ are the ingoing variables while w_- are the outgoing variables. At $x = \ell$, w_- are the ingoing variables and w_+ are the outgoing ones. Guided by [34, 35], we first choose a general class of boundary conditions, allowing the ingoing variables to depend linearly on the respective outgoing ones and some external data. Following the above argumentation, we have to specify boundary values for $w_+(0, t)$ and for $w_-(\ell, t)$. The general formulation of boundary conditions then reads

$$\begin{aligned} w_+(0, t) &= R_0 w_-(0, t) + g_0(t), \\ w_-(\ell, t) &= R_\ell w_+(\ell, t) + g_\ell(t), \end{aligned} \tag{27}$$

for functions g_0, g_ℓ and matrices $R_0, R_\ell \in \mathbb{R}^{6\times 6}$. Given (27), we now have a general class of boundary conditions, that lead to a sensible problem from a mathematical point of view. We now want to trace back these boundary conditions for characteristic variables to boundary conditions for physical variables, u. Furthermore, we want to find out, if the resulting boundary conditions for physical variables are suitable to describe actual physical problems. More precisely, we wish to use (27) to represent the initially in this section described boundary conditions for a clamped beam.

For better readability, we omit the solution's dependency on t in the notation for the rest of the section, writing for example $w_+(0)$ instead of $w_+(0, t)$. Furthermore we define $s := (\Theta^T, \Xi^T)^T$

and $y := (V^T, \Omega^T)^T$ and use this notation to subdivide the state vector u in the following way:

$$u = \begin{pmatrix} \Theta \\ \Xi \\ V \\ \Omega \end{pmatrix} = \begin{pmatrix} s \\ y \end{pmatrix}.$$

That is, s is the compound of inner forces and moments and y the compound of linear and angular velocities. Then by definition (cf. section 2.4.3), w can be written as

$$\begin{aligned} w = \begin{pmatrix} w_- \\ w_+ \end{pmatrix} = T^{-1}u &= \begin{pmatrix} \mathcal{X}\mathbf{F}^{1/2} & \Lambda^{-1}\mathcal{X}\mathbf{F}^{-1/2} \\ \mathcal{X}\mathbf{F}^{1/2} & -\Lambda^{-1}\mathcal{X}\mathbf{F}^{-1/2} \end{pmatrix} \begin{pmatrix} s \\ y \end{pmatrix} \\ &= \begin{pmatrix} \mathcal{X}\mathbf{F}^{1/2}s + \Lambda^{-1}\mathcal{X}\mathbf{F}^{-1/2}y \\ \mathcal{X}\mathbf{F}^{1/2}s - \Lambda^{-1}\mathcal{X}\mathbf{F}^{-1/2}y \end{pmatrix}. \end{aligned} \tag{28}$$

If we choose $R_0 = \mathbf{I}_{6,6}$, the boundary condition at $x = 0$ becomes

$$w_+(0) = w_-(0) + g_0.$$

To trace this back to physical variables at $x = 0$, we have to multiply $w(0)$ by the transformation matrix, as $u = Tw$. In particular, this yields

$$u(0) = Tw\big|_{x=0} = T \begin{pmatrix} w_- \\ w_+ \end{pmatrix}\Bigg|_{x=0}.$$

Inserting the definition of T as well as the boundary condition for w_+, and the relation between w and s and y, derived in (28), we obtain

$$\begin{aligned} u(0) &= \frac{1}{2} \begin{pmatrix} \mathbf{F}^{-1/2}\chi^T & \mathbf{F}^{-1/2}\mathcal{X}^T \\ \mathbf{F}^{1/2}\mathcal{X}^T\Lambda & -\mathbf{F}^{1/2}\mathcal{X}^T\Lambda \end{pmatrix} \begin{pmatrix} \mathcal{X}\mathbf{F}^{1/2}s + \Lambda^{-1}\mathcal{X}\mathbf{F}^{-1/2}y \\ \mathcal{X}\mathbf{F}^{1/2}s + \Lambda^{-1}\mathcal{X}\mathbf{F}^{-1/2}y + g_0 \end{pmatrix}\Bigg|_{x=0} \\ &= \frac{1}{2} \begin{pmatrix} 2s + 2\mathbf{F}^{-1/2}\mathcal{X}^T\Lambda^{-1}\mathcal{X}\mathbf{F}^{-1/2}y + \mathbf{F}^{-1/2}\mathcal{X}^T g_0 \\ -\mathbf{F}^{1/2}\mathcal{X}^T\Lambda g_0 \end{pmatrix}\Bigg|_{x=0}. \end{aligned} \tag{29}$$

Now, suppose that the desired velocities at the clamped end, i.e. at $x = 0$, are denoted by

$$y_0(t) := \begin{pmatrix} V_0(t) \\ \Omega_0(t) \end{pmatrix}.$$

Then setting the external boundary data g_0 to

$$g_0 := -2\Lambda^{-1}\mathcal{X}\mathbf{F}^{-1/2}y_0\big|_{x=0}$$

and inserting this into (29) yields

$$u(0) = \begin{pmatrix} s + \mathbf{F}^{-1/2}\mathcal{X}^T\Lambda^{-1}\mathcal{X}\mathbf{F}^{-1/2}(y - y_0) \\ y_0 \end{pmatrix}\Bigg|_{x=0}. \tag{30}$$

Remember that by definition, $u = (s^T, y^T)$, so that the above result implicates $y(0) = y_0$. This can be inserted into the first row of the above vector, so that all in all, we have

$$u(0) = \begin{pmatrix} s(0) \\ y_0 \end{pmatrix}.$$

Thus, we showed that for the particular choice of R_0 and g_0, the boundary conditions for characteristic variables at $x = 0$ are equivalent to Dirichlet boundary conditions for the linear and angular velocities at $x = 0$, while for $s(0)$ we do not need to specify any boundary conditions.

The same procedure can be done for the right boundary, $x = \ell$. First, we set $R_\ell = -\mathbf{I}_{6,6}$, so that the boundary condition for w_- at $x = \ell$ reads

$$w_-(\ell) = -w_+(\ell) + g_\ell.$$

We want to trace this condition back again to physical variables, using the transformation

$$u(\ell) = Tw\big|_{x=\ell} = T\begin{pmatrix} w_- \\ w_+ \end{pmatrix}\Bigg|_{x=\ell}.$$

Together with the definition of T, the boundary condition for w_- and the relation (28), we have

$$\begin{aligned} u(\ell) &= \frac{1}{2}\begin{pmatrix} \mathbf{F}^{-1/2}\chi^T & \mathbf{F}^{-1/2}\mathcal{X}^T \\ \mathbf{F}^{1/2}\mathcal{X}^T\Lambda & -\mathbf{F}^{1/2}\mathcal{X}^T\Lambda \end{pmatrix}\begin{pmatrix} -\mathcal{X}\mathbf{F}^{1/2}s + \Lambda^{-1}\mathcal{X}\mathbf{F}^{-1/2}y + g_\ell \\ \mathcal{X}\mathbf{F}^{1/2}s - \Lambda^{-1}\mathcal{X}\mathbf{F}^{-1/2}y \end{pmatrix}\Bigg|_{x=\ell} \\ &= \frac{1}{2}\begin{pmatrix} \mathbf{F}^{-1/2}\mathcal{X}^T g_\ell \\ 2y - 2\mathbf{F}^{1/2}\mathcal{X}^T\Lambda\mathcal{X}\mathbf{F}^{1/2}s + \mathbf{F}^{1/2}\mathcal{X}^T\Lambda g_\ell \end{pmatrix}\Bigg|_{x=\ell}. \end{aligned} \tag{31}$$

Suppose that the desired internal forces and moments at the free swinging end are denoted by

$$s_\ell(t) := \begin{pmatrix} \Theta_\ell(t) \\ \Xi_\ell(t) \end{pmatrix}. \tag{32}$$

Then the external boundary data, g_ℓ, can be chosen accordingly again. More precisely, we set

$$g_\ell = 2\mathcal{X}\mathbf{F}^{1/2}s_\ell\big|_{x=\ell}$$

and insert this into (31), which yields

$$u(\ell) = \begin{pmatrix} s_\ell \\ y - \mathbf{F}^{1/2}\mathcal{X}^T\Lambda\mathcal{X}\mathbf{F}^{1/2}(s - s_\ell) \end{pmatrix}\Bigg|_{x=\ell}. \tag{33}$$

By using $u = (s^T, y^T)^T$, we can follow $s(0) = s_\ell$. Inserting this into the second entry of the vector on the above right hand side, we obtain

$$u(\ell) = \begin{pmatrix} s_\ell \\ y(\ell) \end{pmatrix}.$$

Similar to the boundary $x = 0$, we can thereby follow that for the particular choices of R_ℓ and g_ℓ, the boundary conditions for characteristic variables at $x = \ell$ are equivalent to Dirichlet boundary conditions for the internal forces and moments at $x = \ell$, while for the velocities no boundary conditions are needed at this boundary.

Choosing an initial condition $u_0 = u_0(x)$ for $t = 0$, that fulfills the boundary conditions, we can now formulate the problem that we want to analyze and solve numerically. It is an *initial boundary value problem* and reads as follows: find $u = u(x,t) = (s(x,t)^T, y(x,t)^T)^T$, such that

$$\begin{cases} \Gamma u_t + \Pi u_x = Q_{cap}(u) & \text{for } (x,t) \in (0,\ell) \times (0,T], \\ y(0,t) = y_0(t) & \text{for } t \in (0,T], \\ s(\ell,t) = s_\ell(t) & \text{for } t \in (0,T], \\ u(x,0) = u_0(x) & \text{for } x \in [0,\ell]. \end{cases} \tag{34}$$

After deriving the initial boundary value problem with appropriate boundary conditions, we are now able to investigate the energy of a potential solution to this problem in the next section.

2.6 Energy Considerations

In the following section, we will analyze the energy of a potential solution of the initial boundary value problem (34), we derived in the previous sections. In the course of that, we well define a so called *energy norm*, induced by an inner product on the base of the positive definite matrix Γ. The squared energy norm of the solution will be referred to as the *energy* of the solution.

In order to obtain a statement about the energy, we will use the so called *energy method*, multiplying the capacity form of the linear advection equation with the transposed of the solution and thereby obtaining an equation describing the change of energy in time. The energy method for hyperbolic balance laws is for instance used in [26, 34, 35], which is also the literature, we are guided by for this section. We separate the section into two subsections because we want to derive two different statements about the energy.

First, we will show that the solution of the initial boundary value problem is *energy conserving*. That means, that in a modelled beam, there is no energy created or annihilated out of nothing. If there is an increasement or decreasement in energy, this can always be traced back to external influences: either through the ends of the beam and therefore the boundaries of the considered interval or through external sources like external forces and moments. This is an interesting property of the intrinsic beam model, which one may expect from a physical point of view, but from a mathematical point of view this is not clear at all because of the source term, which may mathematically very well be a source of energy along the beam. However, we will see that this source term has – up to the external forces and moments – no influence on the solution's energy.

In the second part of this section, we will use the result from the first part to go on and derive *energy stability* with some restrictions to boundary conditions and external sources. The difference between the two statements is that energy conservation is telling us where energy comes from and that it cannot be created or annihilated in an unphysical way along the beam. Nonetheless, through boundary conditions and external sources, the energy might increase at an arbitrary rate. In this case, we still have energy conservation but no energy stability because the norm of the solution cannot be controlled. To derive a stable numerical approach to solve the problem, we first need a problem that has a stable solution itself. Therefore, we want to bound the change of energy in the first place and then estimate the maximal rate of change in energy. We will see, that different boundary conditions lead to different statements about energy stability.

2.6.1 Energy Conservation

In this subsection, we want to use the energy method, that is also used e.g. in [26, 34, 35], to derive an equation that describes the change of energy of the solution. In order to do so, we first define an inner product and the induced norm. Let $g, h \in \mathbb{L}^2$ be two square integrable functions on $[0, \ell]$. Then the inner product, denoted by $\langle \cdot, \cdot \rangle_\Gamma$ and defined by

$$\langle g, h \rangle_\Gamma := \int_0^\ell g^T \Gamma h \, \mathrm{d}x,$$

induces a norm $\|\cdot\|_\Gamma$ in the following way:

$$\|g\|_\Gamma := \langle g, g \rangle_\Gamma^{1/2} = \left(\int_0^\ell g^T \Gamma g \, \mathrm{d}x \right)^{1/2}.$$

Note that the positive definiteness of Γ is essential for the definition of the inner product and the norm. The above defined norm is in the following referred to as *energy norm* and the squared energy norm of a function will be called the *energy* of that function. This is an abstract mathematical definition of energy, but we will see later in this section that in the case of the intrinsic beam model, this abstract norm does have a physical interpretation.

We now consider the linear advection equation in its capacity form and multiply both sides of the equation by the transposed of the solution, u^T, to obtain the following equation:

$$u^T \Gamma u_t + u^T \Pi u_x = u^T Q_{cap}(u).$$

Integrating this last equation over the domain $[0, \ell]$ gives us

$$\int_0^\ell u^T \Gamma u_t \, \mathrm{d}x + \int_0^\ell u^T \Pi u_x \, \mathrm{d}x = \int_0^\ell u^T Q_{cap}(u) \, \mathrm{d}x. \tag{35}$$

Next, we use the definition of the energy norm to see that with the above equation, we already have an equation that gives a statement about the time derivative of the solution's energy. The first term on the left hand side of (35) can be represented by

$$\int_0^\ell u^T \Gamma u_t \, \mathrm{d}x = \frac{1}{2} \frac{\mathrm{d}}{\mathrm{d}t} \int_0^\ell u^T \Gamma u \, \mathrm{d}x = \frac{1}{2} \frac{\mathrm{d}}{\mathrm{d}t} \|u\|_\Gamma^2 .$$

More precisely, equation (35) can now be written as

$$\frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}t}\|u\|_\Gamma^2 = -\int_0^\ell u^T \Pi u_x \,\mathrm{d}x + \int_0^\ell u^T Q_{cap}(u)\,\mathrm{d}x. \tag{36}$$

To get a better understanding of the energy's time evolution, we take a detailed look at the right hand side of (36). We treat the two integrals separately, beginning with the second one, that gives a measure of the source term's contribution to the energy. Let us therefore consider the integrand of the second integral. By definition, this is

$$u^T Q_{cap}(u) = u^T B u + u^T J(u) u + u^T q_{ext}.$$

First, we show that the matrix B is skew symmetric. When we derived the representation of the source term Q_{cap} in section 2.4.4, we defined B as

$$B = \begin{pmatrix} \mathbf{0}_{6,6} & \mathcal{B} \\ -\mathcal{B}^T & \mathbf{0}_{6,6} \end{pmatrix}.$$

Now the advantage of this notation becomes clear as it allows to directly follow that B is skew symmetric:

$$B^T = \begin{pmatrix} \mathbf{0}_{6,6} & -\mathcal{B} \\ \mathcal{B}^T & \mathbf{0}_{6,6} \end{pmatrix} = -B.$$

As an implication of the skew symmetry of the matrix B, we have that

$$u^T B u = -(Bu)^T u = -u^T B u,$$

which is equivalent to $u^T B u = 0$. In other words: the contribution of the linear part of the source term to the solution's energy equals zero.

We proceed, by investigating the contribution of the quadratic source term, $u^T J(u) u$, to the energy. In section 2.4.4, the term $J(u)$ was defined as

$$J(u) = \begin{pmatrix} \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \widetilde{F_2^T\Theta + F_3\Xi} & \widetilde{F_1\Theta + F_2\Xi} \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \widetilde{F_2^T\Theta + F_3\Xi} \\ \widetilde{F_2^T\Theta + F_3\Xi} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \widetilde{F_1\Theta + F_2\Xi} & \widetilde{F_2^T\Theta + F_3\Xi} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \end{pmatrix}$$
$$- \begin{pmatrix} \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \tilde{\Omega} M_1 & \tilde{\Omega} M_2 \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \tilde{V} M_1 + \tilde{\Omega} M_2^T & \tilde{V} M_2 + \tilde{\Omega} M_3 \end{pmatrix}.$$

Note that, again, because the cross product matrices are skew symmetric, the first summand of $J(u)$ is skew symmetric as well. This means that with the same argumentation as for the matrix B above, the contribution of this first summand of $J(u)$ to the energy is zero. Therefore, we have

$$u^T J(u) u = -\begin{pmatrix} \Theta \\ \Xi \\ V \\ \Omega \end{pmatrix}^T \begin{pmatrix} \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \tilde{\Omega} M_1 & \tilde{\Omega} M_2 \\ \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \tilde{V} M_1 + \tilde{\Omega} M_2^T & \tilde{V} M_2 + \tilde{\Omega} M_3 \end{pmatrix} \begin{pmatrix} \Theta \\ \Xi \\ V \\ \Omega \end{pmatrix}.$$

Multiplying the above terms, the contribution of the quadratic source term to the energy becomes

$$u^T J(u) u = -V^T \tilde{\Omega} M_1 V - \Omega^T \left(\tilde{V} M_1 + \tilde{\Omega} M_2^T\right) V - V^T \tilde{\Omega} M_2 \Omega - \Omega^T \left(\tilde{V} M_2 + \tilde{\Omega} M_3\right) \Omega. \tag{37}$$

In order to handle this last term, we first show some properties of the cross product matrix. Let therefore $v, y, z \in \mathbb{R}^3$ be arbitrary vectors. We already mentioned, that the cross product matrix is skew symmetric, i.e.

$$\tilde{v}^T = -\tilde{v}. \tag{i}$$

Another property that directly follows from the anticommutativity of the cross product itself and can also be shown easily, is that it holds

$$\tilde{y}z = -\tilde{z}y. \tag{ii}$$

Furthermore, a vector multiplied with itself in the cross product is zero. Together with statement (i), we can thereby follow

$$y^T\tilde{y}z = (-\tilde{y}y)^T z = \mathbf{0}_3. \tag{iii}$$

Properties (i) and (ii) let us derive one more symmetry property, that reads as follows:

$$v^T\tilde{y}z = (-\tilde{y}v)^T z = (\tilde{v}y)^T z = -y^T\tilde{v}z. \tag{iv}$$

Coming back now to the energy contribution of the quadratic source term (37), we use the properties (iii)-(iv) to simplify the expression. Taking advantage of (iv), it holds

$$-V^T\tilde{\Omega}M_1V - \Omega^T\tilde{V}M_1V = -V^T\tilde{\Omega}M_2\Omega - \Omega^T\tilde{V}M_2\Omega = 0.$$

And using (iii), we obtain

$$-\Omega^T\tilde{\Omega}M_2^TV = -\Omega^T\tilde{\Omega}M_3\Omega = 0,$$

meaning that the contribution of the second summand of the quadratic source term is zero as well and that, in total, we have

$$u^TJ(u)u = 0.$$

Putting this result together with the result for the contribution of the linear part of the source term, the total contribution of the source term to the energy becomes

$$\int_0^\ell u^TQ_{cap}(u)\,\mathrm{d}x = \int_0^\ell u^Tq_{ext}\,\mathrm{d}x. \tag{38}$$

Note that the above statement is independent of u being an exact solution to the intrinsic beam equation. Due to the skew symmetry of the linear source term and the first part of the quadratic source term and the special symmetry properties of the cross product matrices, u in (38) can be replaced by any vector in $\mathbb{R}^{12}$ and the equality will still hold. This is an important observation, we will take advantage of in the analysis of the discrete energy in section 3.5.

The result (38) for the contribution of the source term is completely in line with the expectations from a physical point of view, because it states that if we neglect external forces and moments, the source term has no impact on the solution's energy. That is because, except for external forces and moments, the source term in the linear advection formulation does not emerge from any actual sources that create or annihilate physical quantities along the beam out of nothing. It rather comes from a coupling that transforms some of the considered physical quantities into others and therefore should be balanced in terms of the energy. Thus, we can update the energy equation (36), to get

$$\frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}t}\|u\|_\Gamma^2 = -\int_0^\ell u^T\Pi u_x\,\mathrm{d}x + \int_0^\ell u^Tq_{ext}\,\mathrm{d}x. \tag{39}$$

After looking into the energy contribution of the source term, we now consider the remaining part of the right hand side in the updated energy equation (39), namely the first integral. Integration by parts for that term yields

$$-\int_0^\ell u^T\Pi u_x\,\mathrm{d}x = \int_0^\ell u^T\Pi u_x\,\mathrm{d}x - u^T\Pi u\Big|_0^\ell,$$

where the integrand of the first term on the right hand side is actually $(u^T\Pi)_x u$, but due to the constancy and symmetry of the matrix Π, this is the same as $u^T\Pi u_x$. Rearranging terms now equivalently results in

$$-\int_0^\ell u^T\Pi u_x \,\mathrm{d}x = -\frac{1}{2}u^T\Pi u\Big|_0^\ell,$$

meaning that this part of the energy equation can be traced back to boundary terms. The energy equation now reads

$$\frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}t}\|u\|_\Gamma^2 = -\frac{1}{2}u^T\Pi u\Big|_0^\ell + \int_0^\ell u^T q_{ext}\,\mathrm{d}x. \tag{40}$$

The above result for the change of energy can already be interpreted as energy conservation. We observe that the energy changes only due to external influences: either through the boundaries, i.e. the ends of the beam, or through the presence of external forces and moments along the beam. Before we come to the energy stability in the next section, we want to give a more physical interpretation of the derived energy conservation statement (40).

First, we execute the matrix vector multiplication in the boundary term in equation (40) to obtain

$$-\frac{1}{2}u^T\Pi u\Big|_0^\ell = \frac{1}{2}\,u^T\begin{pmatrix}\mathbf{0}_{6,6} & \mathbf{I}_{6,6}\\ \mathbf{I}_{6,6} & \mathbf{0}_{6,6}\end{pmatrix}u\Big|_0^\ell = \left(V^T\Theta + \Omega^T\Xi\right)\Big|_0^\ell. \tag{41}$$

Moreover, by definition of the external source term q_{ext}, the second part on the right hand side of the energy equation (40) can be represented by

$$\int_0^\ell u^T q_{ext}\,\mathrm{d}x = \int_0^\ell \left(V^T f_{ext} + \Omega^T m_{ext}\right)\,\mathrm{d}x. \tag{42}$$

And if we insert the results (41) and (42) into the energy equation (40), it becomes

$$\frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}t}\|u\|_\Gamma^2 = \left(V^T\Theta + \Omega^T\Xi\right)\Big|_0^\ell + \int_0^\ell \left(V^T f_{ext} + \Omega^T m_{ext}\right)\,\mathrm{d}x. \tag{43}$$

The last result is therefore just another representation of (40), but it allows a physical interpretation of the derived energy conservation. In fact, equation (43) is a result, Hodges already showed in [20] using a slightly different approach. According to that, the left hand side of (43) is the temporal change of total mechanical energy of the beam. As we already suggested earlier in this section, this means that the abstract energy we defined, is not only a mathematical energy but also gives a measure of the beam's actual mechanical energy. The right hand side of (43) on the other hand, is according to [20] the sum of the work done at the beam's ends (boundary term on the right hand side) and the work done along the beam (external sources term on the right hand side).

Thus, what we showed, using mathematical techniques and what Hodges already showed in his work, is that the total mechanical energy of a modelled beam changes only due to work done at its ends or along the beam, which is just the physical definition of energy conservation.

2.6.2 Energy Stability

The goal in this section, is to find a bound for the right hand side of the energy equation

$$\frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}t}\|u\|_\Gamma^2 = -\frac{1}{2}u^T\Pi u\Big|_0^\ell + \int_0^\ell u^T q_{ext}\,\mathrm{d}x, \tag{44}$$

we derived in the previous section. In the case where the energy can be bounded and cannot increase at an arbitrary rate, the solution of the problem is called energy stable. In order to find

such a bound, we have to estimate both, the boundary term and the term that emerges from external sources, which we do separately.

Estimating the boundary term is usually realized by applying the boundary conditions for u and then finding a bound for example by completing the square. Before we explicitly insert the boundary conditions, we would like to mention a result from the literature, namely [34, 35]. Let us therefore take the general boundary conditions

$$w_+(0,t) = R_0 w_-(0,t) + g_0(t),$$

$$w_-(\ell,t) = R_\ell w_+(\ell,t) + g_\ell(t),$$

for characteristic variables and the matrices R_0 and R_ℓ from section 2.5 into account again. Nordström showed in [34] that a sufficient condition to find an appropriate estimation for the boundary terms regarding characteristic variables is that the matrix

$$R_i \Lambda R_i - \Lambda$$

is negative definite for $i = 0, \ell$, where we transferred the general results of [34] for initial boundary value problems to our specific problem. Remember that the diagonal matrices $\pm\Lambda$ were defined in section 2.4.2 as the blocks, forming the matrix $\mathbf{\Lambda}$. From a mathematical point of view, this is already a result worth mentioning: For the intrinsic beam equation with general boundary conditions fulfilling this requirement, we can bound the boundary terms and have energy stability if we assume that we find an appropriate bound for the external sources as well. Nevertheless, for our application of a clamped beam, we would like to set $R_0 = \mathbf{I}_{6,6}$ and $R_\ell = -\mathbf{I}_{6,6}$. For this particular choice, the above sufficient condition is not fulfilled, but we have

$$R_i \Lambda R_i - \Lambda = \mathbf{0}_{6,6}$$

instead. The statement in [34, 35] for this special case is that energy stability can be obtained for zero external boundary data, i.e. $g_0 = g_\ell = \mathbf{0}_6$. For non-zero boundary data, there is not a statement made. That is why we take a detailed look into this case ourselves in the following.

The considerations in [34, 35] are for characteristic variables. The energy equation, we derived, concerns physical variables, but as $u = Tw$, we can switch between physical and characteristic variables equivalently. To be able to apply the argumentation of [34, 35], we will therefore represent the boundary term in (44) in terms of characteristic variables. In particular, by inserting $u = Tw$ into the boundary term, we obtain

$$-\frac{1}{2} u^T \Pi u \Big|_0^\ell = -\frac{1}{2} w^T T^T \Pi T w \Big|_0^\ell.$$

The matrix product can be calculated straight forward using the definitions of T and Π from section 2.4.2 so that the above term becomes

$$-\frac{1}{2} u^T \Pi u \Big|_0^\ell = -\frac{1}{4} w^T \mathbf{\Lambda} w \Big|_0^\ell. \tag{45}$$

Remember that $\mathbf{\Lambda}$ can be represented in block form

$$\mathbf{\Lambda} = \begin{pmatrix} -\Lambda & \mathbf{0}_{6,6} \\ \mathbf{0}_{6,6} & \Lambda \end{pmatrix}$$

and that the vector of characteristic variables can be subdivided into $w = (w_-^T, w_+^T)^T$. Therefore, the right hand side of (45) can be rewritten, so that we have

$$-\frac{1}{2} u^T \Pi u \Big|_0^\ell = -\frac{1}{4} \left(-w_-^T \Lambda w_- + w_+^T \Lambda w_+ \right) \Big|_0^\ell.$$

Evaluating the right hand side at $x = 0$ and $x = \ell$ gives

$$-\frac{1}{2} u^T \Pi u \Big|_0^\ell = \frac{1}{4} \Big(w_-^T \Lambda w_- \big|_\ell - w_+^T \Lambda w_+ \big|_\ell - w_-^T \Lambda w_- \big|_0 + w_+^T \Lambda w_+ \big|_0 \Big).$$

We proceed by inserting the boundary conditions for characteristic variables with $R_0 = \mathbf{I}_{6,6}$ and $R_\ell = -\mathbf{I}_{6,6}$:

$$-\frac{1}{2}u^T\Pi u\Big|_0^\ell = \frac{1}{4}\Big((g_\ell - w_+)^T\Lambda(g_\ell - w_+)\big|_\ell - w_+^T\Lambda w_+\big|_\ell - w_-^T\Lambda w_-\big|_0 + (g_0 + w_-)^T\Lambda(g_0 + w_-)\big|_0\Big)$$

and by gathering terms, we get

$$-\frac{1}{2}u^T\Pi u\Big|_0^\ell = \frac{1}{4}\Big(g_\ell^T\Lambda g_\ell\big|_\ell - 2g_\ell^T\Lambda w_+\big|_\ell + g_0^T\Lambda g_0\big|_0 + 2g_0^T\Lambda w_-\big|_0\Big). \tag{46}$$

To obtain energy stability, a bound for the right hand side of the last result has to be found. To the best of our knowledge, there is no literature that makes a general statement about a bound for this term for non-zero external boundary data. The appearance of mixed terms containing the external boundary data g_0 and g_ℓ and the evaluations of w_+ and w_- makes it difficult to find a bound because a priori, a statement about their respective sign cannot be made. Neither did we find an appropriate way of completing the squares or any other procedure to obtain an estimation. In the argumentation of Nordström, this difficulty arises from the circumstance that

$$R_i\Lambda R_i - \Lambda = \mathbf{0}_{6,6}. \tag{47}$$

At this point, we want to emphasize that neither the authors of [34, 35], nor us do state that there is no bound for the terms in (46). The argumentation used in [34, 35] can simply not be used to derive general statements for boundary conditions of linear advection equations with the property (47). This means that boundary conditions of this type have to be considered individually from case to case. In the future, there might well be found another argumentation that can be used to find an estimation for boundary conditions with non-zero external boundary data and the particular choices of R_0 and R_ℓ for the intrinsic beam model.

For zero external boundary data on the other hand, i.e. $g_0 = g_\ell = \mathbf{0}_6$, we get a bound easily because the right hand side of (46) vanishes in this case and thus, the contribution of the boundary terms to the energy is zero. The rate of change in the energy then becomes

$$\frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}t}\|u\|_\Gamma^2 = \int_0^\ell u^T q_{ext}\,\mathrm{d}x. \tag{48}$$

For zero external forces and moments, the above right hand side is zero. This is the first important stability result. If we assume zero external boundary data and zero external forces and moments, the energy of the solution is constant in time. This is also a direct implication of the energy conservation statement, we derived in the previous section. For the rest of the thesis, if not stated otherwise, the external boundary data g_0 and g_ℓ is assumed to be zero. Remember that, in order to transfer the boundary conditions from characteristic variables to physical variables, in section 2.5, g_0 and g_ℓ were set to

$$g_0 = -2\Lambda^{-1}\mathcal{X}\mathbf{F}^{-1/2}y_0\big|_{x=0},$$
$$g_\ell = 2\mathcal{X}\mathbf{F}^{1/2}s_\ell\big|_{x=\ell},$$

where y_0 and s_ℓ were the velocities and internal forces and moments, we want to prescribe at the respective boundary to describe the clamped beam. Allowing $g_0, g_\ell = \mathbf{0}_6$ only, means that in terms of boundary conditions for physical variables, we only allow zero velocities at the clamped end and zero internal forces and moments at the free swinging end.

Aside from the resulting energy stability for zero external forces and moments, we would like to show, that even for non-zero external forces and moments, the corresponding term in (48) can be bounded. At first glance, the handling of this term seems easy, because for bounded external forces and moments in the sense that $\|q_{ext}\|_{\mathbb{L}^2} \leq c_1$, for every point in time, the corresponding term can be estimated by

$$\int_0^\ell u^T q_{ext}\,\mathrm{d}x \leq c_1\|u\|_{\mathbb{L}^2}, \tag{49}$$

where we used the Cauchy-Schwarz inequality for the $\mathbb{L}^2$ inner product. In reality, to find an estimation for that term is a lot harder because of the body attached variables. As explained in section 2.1, external forces act on the beam not in a fixed coordinate system, but in a body attached coordinate system that deforms together with the beam's deformation. This means that even a constant external force (as seen from the global coordinate system) does not result in a constant norm of q_{ext} but somehow depends on the beam's deformation and therefore on the time. Finding an a priori estimation for the norm of q_{ext} that does not depend on the time variable is therefore difficult.

To the best of our knowledge, there is no energy estimation for the solution of the intrinsic beam equation that considers realistic external forces and moments. Rodriguez et al. for example neglect external forces and moments in [42, 43, 44]. In this thesis, we will only allow constant (in time) external forces and moments (as seen from the body attached coordinate system), i.e. $f_{ext}(x,t) = f_{ext}(x)$ and $m_{ext}(x,t) = m_{ext}(x)$.

Although, these constant external forces and moments can hardly be transferred to realistic physical applications, we do take this choice instead of neglecting external sources completely because we are interested in deriving an energy stability statement for the most general case.

With the above assumptions, inequality (49) holds and we would like to trace the $\mathbb{L}^2$-norm of u on the right hand side, back to the energy norm of u. To do so, we use *Rayleigh's min-max principle*. The theorem and a proof is for example to find in [21, pp. 235f.]. It states that for the symmetric matrix $\Gamma(x)$ with minimal eigenvalue $\mu_{\min}(x)$ and maximal eigenvalue $\mu_{\max}(x)$, it holds

$$\mu_{\min}(x) \leq \frac{z^T \Gamma(x) z}{z^T z} \leq \mu_{\max}(x)$$

for any non-zero vector $z \in \mathbb{R}^{12}$. A direct implication of this is that

$$z^T z \leq \frac{1}{\mu_{\min}(x)} z^T \Gamma(x) z$$

for any $z \in \mathbb{R}^{12}$. Due to the positive definiteness of Γ, its minimal eigenvalue, $\mu_{\min}(x)$, is always positive so that its reciprocal is well defined. Now, defining $c_2 := \max_{x\in[0,\ell]} {}^{1}\!/_{\mu_{\min}(x)}$, we can follow

$$\int_0^\ell u^T u \,\mathrm{d}x \leq c_2 \int_0^\ell u^T \Gamma u \,\mathrm{d}x,$$

which is the same as

$$\|u\|_{\mathbb{L}^2} \leq \sqrt{c_2}\, \|u\|_\Gamma \,.$$

Inserting the last result into (49), respective (48), we obtain

$$\frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}t} \|u\|_\Gamma^2 \leq c_3 \|u\|_\Gamma \,, \tag{50}$$

where $c_3 := c_1\sqrt{c_2}$. The above result is an ordinary differential inequality that can be solved using the *Theorem of Petrovitsch*, originating from [38]. We set $z(t) = \frac{1}{2}\|u(\cdot,t)\|_\Gamma^2$, which results in

$$\frac{\mathrm{d}}{\mathrm{d}t} z(t) \leq c_3 \sqrt{2z(t)}. \tag{51}$$

To apply Petrovitsch's theorem, we first need to find a solution of the following initial value problem:

$$\begin{cases} \frac{\mathrm{d}}{\mathrm{d}t}\hat{z}(t) & = c_3\sqrt{2\hat{z}(t)}, \\ \hat{z}(0) & = \frac{1}{2}\|u_0\|_\Gamma^2 \,, \end{cases} \tag{52}$$

where u_0 is the initial condition of the initial boundary value problem, i.e. $u(x,0) = u_0(x)$. One can easily check that the function

$$\hat{z}(t) = \frac{1}{2}\left(\|u_0\|_\Gamma + c_3 t\right)^2$$

is the solution of the initial value problem (52) and according to Petrovitsch's theorem, we can now follow that

$$z(t) \leq \hat{z}(t) \qquad \text{for } t > 0,$$

which is equivalent to

$$\|u(\cdot,t)\|_\Gamma^2 \leq \left(\|u_0\|_\Gamma + c_3 t\right)^2 .$$

That is, the energy can be bounded by the quadratic term on the right hand side. Thus, we showed that the energy of the solution cannot increase faster than quadratically in time. A direct implication for the energy norm of the solution is that

$$\|u(\cdot,t)\|_\Gamma \leq \|u_0\|_\Gamma + c_3 t,$$

meaning that for constant (in time) and bounded external forces and moments, the energy norm of the solution increases at maximum linearly in t. That is, under the assumption that the external boundary data g_0 and g_ℓ is zero and external forces and moments are constant in time and bounded in terms of their $\mathbb{L}^2$-norm, the solution of the intrinsic beam equation is energy stable.

After finding the desired bound and deriving energy stability for the solution of the initial boundary value problem, we are now able to proceed by discretizing the problem and derive a similar statement for the approximate solution in the next section.

3 Discretization

In the course of this section, we will discretize the initial boundary value problem, that resulted from the considerations in the previous sections. Special attention is thereby paid to the discretization in space, for which we will use a Discontinuous Galerkin approach. We will derive a weak formulation of the linear advection equation in capacity form and, afterwards, discretize the solution space of that weak formulation in space, resulting in a semi discrete formulation that still continuously depends on the time variable. Furthermore, we will introduce the concept of numerical fluxes and derive such a numerical flux for the discretization of the intrinsic beam equation. The spatial discretization is completed by discussing how the boundary conditions can be implemented into the scheme via the numerical flux.

The semi discretization will be investigated regarding its energy. We will find a definition of discrete energy of the numerical solution analogous to the energy from section 2. Using this measure of energy in the discrete context, we will then show that the numerical solution inherits the energy properties of the exact solution. This means, that we will analogously to the energy considerations in the previous section, derive statements about both, energy conservation and energy stability of the numerical solution. The semi discrete formulation will then be formulated as an ordinary differential equation in time and some remarks on the discretization in time will be made to complete the section.

3.1 Theoretical Preliminaries

In the following section, the most important theoretical basics are defined, which are needed for the discretization approach in the subsequent sections. This includes the definition of the Legendre polynomials, the Legendre-Gauß-Lobatto quadrature method to approximate integrals, the definition of the Lagrange polynomials as well as the introduction of an interpolation operator. Aside from the definition of the interpolation operator, the definitions in this section can also be found in [18, 24].

We will proceed as follows: first, we will define the Legendre polynomials. On the basis of this, we will define the Legendre-Gauß-Lobatto quadrature method whose quadrature points are the roots of the Legendre polynomials. Afterwards, we define the Lagrange polynomials and an interpolation operator that is based on the latter polynomials.

The Legendre polynomials are $\mathbb{L}^2$-orthogonal polynomials $\{P_j\}_{j=0,\dots,N}$ on the interval $[-1,1]$. For the purpose of this thesis, it is sufficient to define the polynomials by their recursion formula as done in the following definition.

Definition 1 *Let $N \in \mathbb{N}$. The* Legendre polynomials $\{P_j\}_{j=0,\dots,N}$ *are defined by the following recursion:*

$$P_0(\xi) = 1, \qquad P_1(\xi) = \xi$$

and

$$P_{j+1}(\xi) = \frac{2j+1}{j+1}\xi P_j(\xi) - \frac{j}{j+1}P_{j-1}(\xi)$$

for $j = 1, \dots, N-1$.

On the basis of this last definition, we are now able to specify a formula for the Legendre-Gauß-Lobatto quadrature.

Definition 2 *Let $N \in \mathbb{N}$. We define the quadrature nodes $\{\xi_j\}_{j=0,\dots,N}$, as follows: we set*

$$\xi_0 = -1, \qquad \xi_N = +1$$

and ξ_j as the zeros of the Legendre polynomial P_{N-1} for $j = 1, \dots, N-1$. Further, we define the quadrature weights $\{\omega_j\}_{j=0,\dots,N}$ as

$$\omega_j = \frac{2}{(N+1)N(P_N(\xi_j))^2}, \qquad \text{for } j = 0, \dots, N.$$

For a function $z = z(\xi)$, the Legendre-Gauß-Lobatto quadrature formula *(or short* LGL quadrature*) to approximate the integral of z over the interval $[-1, 1]$ then reads as follows:*

$$\int_{-1}^{1} z(\xi)\,\mathrm{d}\xi \approx \sum_{j=0}^{N} z(\xi_j)\omega_j.$$

An integral that is approximated by the LGL quadrature method will for the rest of the thesis also be denoted by

$$\int_{-1,N}^{1} z(\xi)\,\mathrm{d}\xi.$$

Note that the defined quadrature is exact for polynomials up to a degree of $2N - 1$.

Before we proceed with the discretization, we define another set of polynomials, namely the Lagrange polynomials:

Definition 3 *Let $N \in \mathbb{N}$ and consider the points $\{\xi_j\}_{j=0,\dots,N}$ from Definition 2. The* Lagrange polynomials *L_j for $j = 0, \dots, N$ are defined defined as the following polynomials:*

$$L_j(\xi) = \prod_{i=0, i\neq j}^{N} \frac{(\xi - \xi_i)}{(\xi_j - \xi_i)}.$$

Lagrange polynomials have the really useful property $L_j(\xi_i) = \delta_{ij}$ with the Kronecker delta δ_{ij}.

Note, that the $N + 1$ Lagrange polynomials, as defined above, form a basis for the space of polynomials of degree N on the interval $[-1, 1]$. Note also, that the points $\{\xi_j\}_{j=0,\dots,N}$ in the above definition can be replaced by any set of $N + 1$ disjoint points in $[-1, 1]$, that include the boundaries ± 1. However, in the following sections, the Lagrange polynomials will be used just as they are defined above.

In the following sections, with $\mathcal{P}_N([-1, 1])$, we will denote the space of multivariate polynomials that map from the interval $[-1, 1]$ onto the twelve dimensional space $\mathbb{R}^{12}$. Given definition 3, we have that the set

$$\mathcal{L} := \left\{\mathcal{L}_j^{\mathrm{m}}\right\}_{j=0,\dots,N}^{\mathrm{m}=1,\dots,12} := \left\{e_{\mathrm{m}} L_j\right\}_{j=0,\dots,N}^{\mathrm{m}=1,\dots,12}$$

forms a basis of $\mathcal{P}_N([-1, 1])$, where e_{m} is the m-th element of the standard basis of $\mathbb{R}^{12}$. To complete this section, we define an interpolation operator:

Definition 4 *Let N, $\{\xi_j\}_{j=0,\dots,N}$ be defined as before and let $z = z(\xi)$ be a vector valued or matrix valued function on $[-1, 1]$. The interpolation operator $I^N(\cdot)$ maps every component of z onto its unique interpolation polynomial of degree N at the interpolation points $\{\xi_j\}_{j=0,\dots,N}$. Let for example $z : [-1, 1] \to \mathbb{R}^{12}$ with*

$$z(\xi) = (z_1(\xi), \dots, z_{12}(\xi))^T.$$

Then $I^N(z) \in P_N([-1, 1])$ is defined as

$$\left(I^N(z)\right)(\xi) = \sum_{\mathrm{m}=1}^{12} \sum_{j=0}^{N} a_j^{\mathrm{m}} \mathcal{L}_j^{\mathrm{m}}(\xi),$$

or in the component wise representation

$$\left(I^N(z)\right)(\xi) = \left(\sum_{j=0}^{N} a_j^1 L_j(\xi), \dots, \sum_{j=0}^{N} a_j^{12} L_j(\xi)\right)^T,$$

for appropriate coefficients a_j^{m} and the Lagrange basis functions L_j from definition 3.

With these definitions, we now have the essential mathematical tools to derive a semi discrete formulation of the intrinsic beam equation in its linear advection formulation from section 2.

3.2 A Discontinuous Galerkin Approach for Discretization in Space

In this section the semi discrete formulation will be derived. The procedure is guided by [26]. We will subdivide the domain $[0, \ell]$ into cells and derive a cell wise weak formulation of the considered equation, which will then be discretized in space. The starting point for the discretization is the capacity form of the linear advection equation, we derived in section 2.4.4. It reads

$$\Gamma u_t + \Pi u_x = Q_{cap}(u). \tag{53}$$

As mentioned initially, the considered interval $[0, \ell]$ is subdivided into non overlapping *elements* or *cells* $\{C_\mathrm{k}\}_{\mathrm{k}=1\ldots,N_c}$, defined by

$$C_\mathrm{k} := \left[x_\mathrm{k} - \frac{\Delta x_\mathrm{k}}{2}, x_\mathrm{k} + \frac{\Delta x_\mathrm{k}}{2}\right] \subset [0, \ell],$$

meaning that for $\mathrm{k} = 1, \ldots, N_c$, the point x_k is the center and Δx_k is the width of the cell C_k. Since the solution of equation (53) fulfills the equation on $(0, \ell)$, it fulfills it on every cell C_k and the equation can be considered cell wise. To save computational operations and memory requirement, we transform every cell C_k to the same reference element R with

$$R := [-1, 1].$$

In particular, for every cell C_k, the transformation is a linear map. A point $x \in C_\mathrm{k}$ is mapped onto a point $\xi = \xi(x) \in R$ by

$$\xi(x) = \frac{2}{\Delta x_\mathrm{k}}(x - x_\mathrm{k}). \tag{54}$$

Inversely, a point $\xi \in R$ is mapped onto a point $x = x(\xi) \in C_\mathrm{k}$ by

$$x(\xi) = x_\mathrm{k} + \frac{\Delta x_\mathrm{k}}{2}\xi.$$

Using this transformation between the physical cells C_k, and the reference element R, the dependencies of the considered quantities on the space variable x can be interpreted as dependencies on ξ, as it holds for example $u(x,t)|_{C_\mathrm{k}} = u(x(\xi), t)|_R$ for every cell C_k. Thus, equation (53) can also be transformed from each cell to the reference element. Therefore, the derivatives with respect to x, that appear in (53) have to be transformed. In particular, we have

$$\Pi u_x = \Pi u_\xi \frac{\mathrm{d}\xi}{\mathrm{d}x} = \frac{2}{\Delta x_\mathrm{k}} \Pi u_\xi \tag{55}$$

for every cell C_k. Inserting the transformation of the derivative with respect to x into equation (53) and multiplying both sides by $\Delta x_\mathrm{k}/2$ afterwards, yields that on every cell C_k, it holds

$$\frac{\Delta x_\mathrm{k}}{2}\Gamma u_t + \Pi u_\xi = \frac{\Delta x_\mathrm{k}}{2} Q_{cap}(u). \tag{56}$$

On the basis of equation (56), we now want to derive the *weak formulation* of the linear advection equation for each cell C_k. We start by multiplying the equation by the transposed of an arbitrary *test function* φ that is smooth on the reference element R and integrating both sides over R:

$$\frac{\Delta x_\mathrm{k}}{2} \int_{-1}^{1} \varphi^T \Gamma u_t \,\mathrm{d}\xi + \int_{-1}^{1} \varphi^T \Pi u_\xi \,\mathrm{d}\xi - \frac{\Delta x_\mathrm{k}}{2} \int_{-1}^{1} \varphi^T Q_{cap}(u) \,\mathrm{d}\xi = 0.$$

The second summand on the left hand side of the above expression can be integrated by parts. Additionally, we exploit the symmetry and constancy of the matrix Π, which results in

$$\frac{\Delta x_\mathrm{k}}{2} \int_{-1}^{1} \varphi^T \Gamma u_t \,\mathrm{d}\xi - \int_{-1}^{1} (\Pi\varphi)_\xi^T u \,\mathrm{d}\xi - \frac{\Delta x_\mathrm{k}}{2} \int_{-1}^{1} \varphi^T Q_{cap}(u) \,\mathrm{d}\xi = -\varphi^T f\Big|_{-1}^{1}, \tag{57}$$

with the flux function $f(u) = \Pi u$. The test function φ was left arbitrary, so that (57) holds for every test function that is smooth in R.

The goal is now, to discretize the cell wise weak formulation of the intrinsic beam equation with respect to the spatial variable x, respectively ξ. Hence, we have to specify a finite dimensional function space from which we choose a numerical solution to approximate the exact solution of the problem. Likewise, the space of test functions has to be discretized and therefore restricted to a finite dimensional space of test functions. We will approximate both, the local solution in the cells and the test functions, by polynomials of degree N_p.

Because the polynomials of adjacent cells do not have to coincide, the numerical solution is a piecewise polynomial in a global point of view, i.e. on $[0, \ell]$. Thus, the discrete space of test functions and the global discrete solution space can be specified as

$$\mathcal{V}_{N_p} := \left\{ W \in \mathbb{L}^2([0,\ell]) \;\middle|\; W|_{C_\mathrm{k}} \in \mathcal{P}_{N_p}(C_\mathrm{k}) \right\}.$$

Consequently, there usually occur discontinuities in the numerical solution at the interfaces between the cells. This special property of the discretization approach results in its name *Discontinuous* Galerkin Approach. The global numerical solution and the test functions from the discrete set of test functions are denoted by $U, \phi \in \mathcal{V}_{N_p}$, respectively. For the local numerical solution in a specific cell C_k, we write $U^{(\mathrm{k})} = U|_{C_\mathrm{k}}$. Moreover, whenever we write $U^{(\mathrm{k})}(\xi, t)$, in the following we mean $U^{(\mathrm{k})}(x(\xi), t)$ with $x(\xi) \in C_\mathrm{k}$. Formally, $U^{(\mathrm{k})}$ is the interpolation polynomial of the exact solution u in the cell C_k. As interpolation points we choose the $N_p + 1$ LGL points $\{\xi_j\}_{j=0,\dots,N_p}$ on $R = [-1, 1]$ (cf. definition 2). Using the basis $\mathcal{L}$ of the polynomial space $\mathcal{P}_{N_p}$, we defined in the previous section, the numerical solution in the cell C_k can therefore be specified as

$$U^{(k)}(\xi, t) = \sum_{\mathrm{m}=1}^{12} \sum_{j=0}^{N_p} a_j^{\mathrm{m},(\mathrm{k})}(t) \mathcal{L}_j^{\mathrm{m}}(\xi)$$

for appropriate coefficients $a_j^{\mathrm{m},(\mathrm{k})}$ depending on the time variable. A more detailed look into these coefficients and how they can be determined will be given in section 3.6.

When we want to approximate equation (57) with the discretized quantities we defined above, the discontinuities at the interfaces have to be minded especially regarding the right hand side of the equation. There, the evaluations of the flux function $f(u) = \Pi u$ at the cell boundaries have to be approximated. The cell boundaries are just the points, where the numerical solution is discontinuous and therefore not unambiguously defined. The idea of the Discontinuous Galerkin approach is now to approximate these evaluations by a so called *numerical flux* function $\mathcal{F}^*$. This idea comes from the Finite Volume Methods, which can be interpreted as Discontinuous Galerkin methods with $N_p = 0$. A more detailed look into the ideas of the numerical flux is given in section 3.3. For now we just take it as an abstract concept and assume that $\mathcal{F}^*$ is an approximation of the flux function at the cell boundaries.

Summarizing the considerations on the discretization of the function spaces and on the numerical flux, we can specify an approximation of equation (57):

$$\frac{\Delta x_\mathrm{k}}{2} \int_{-1}^{1} \phi^T \Gamma U_t^{(\mathrm{k})} \,\mathrm{d}\xi - \int_{-1}^{1} (\Pi\phi)_\xi^T U^{(\mathrm{k})} \,\mathrm{d}\xi + \frac{\Delta x_\mathrm{k}}{2} \int_{-1}^{1} \phi^T Q_{cap}(U^{(\mathrm{k})}) \,\mathrm{d}\xi = -\phi^T \mathcal{F}^* \Big|_{-1}^{1}. \tag{58}$$

Further, we approximate the matrix Γ and the source term $Q_{cap}(U^{(\mathrm{k})})$ using the interpolating operator, we defined in definition 4 and denote the approximations by

$$\Gamma^{(\mathrm{k})} := I^{N_p}\left(\Gamma\big|_{C_\mathrm{k}}\right),$$

$$Q_{cap}^{(\mathrm{k})} := I^{N_p}\left(Q_{cap}(U^{(\mathrm{k})})\right).$$

Another approximation of equation (58) is therefore given by

$$\frac{\Delta x_\mathrm{k}}{2} \int_{-1}^{1} \phi^T \Gamma^{(\mathrm{k})} U_t^{(\mathrm{k})} \,\mathrm{d}\xi - \int_{-1}^{1} (\Pi\phi)_\xi^T U^{(\mathrm{k})} \,\mathrm{d}\xi + \frac{\Delta x_\mathrm{k}}{2} \int_{-1}^{1} \phi^T Q_{cap}^{(k)} \,\mathrm{d}\xi = -\phi^T \mathcal{F}^* \Big|_{-1}^{1}. \tag{59}$$

To obtain a formulation that is fully discrete with regard to the space variable, we need to replace the integrals in (59) by a numerical quadrature. In particular, we use the Legendre-Gauß-Lobatto quadrature method with nodes $\{\xi_j\}_{j=0,\dots,N_p}$ and weights $\{\omega_j\}_{j=0,\dots,N_p}$ as defined in definition 2. Note that the quadrature nodes $\{\xi_j\}_{j=0,\dots,N_p}$ coincide with the interpolation nodes of the operator I^{N_p}. This accordance is called *co-location* of quadrature and interpolation nodes, and we will see later that it leads to a very efficient method and brings some other advantages with it. The approximation of (59) with the integrals replaced by LGL quadratures reads

$$\frac{\Delta x_{\mathrm{k}}}{2} \int\limits_{-1,N_p}^{1} \phi^T \Gamma^{(\mathrm{k})} U_t^{(\mathrm{k})} \,\mathrm{d}\xi - \int\limits_{-1,N_p}^{1} (\Pi\phi)_\xi^T U^{(\mathrm{k})} \,\mathrm{d}\xi + \frac{\Delta x_{\mathrm{k}}}{2} \int\limits_{-1,N_p}^{1} \phi^T Q_{cap}^{(k)} \,\mathrm{d}\xi = -\phi^T \mathcal{F}^* \Big|_{-1}^{1}. \tag{60}$$

Let us, on the basis of the LGL quadrature, define a discrete inner product and the discrete norm that is induced by this inner product. Therefore, consider two polynomials $W, Z \in \mathcal{P}_{N_p}([-1,1])$. The discrete inner product is defined by

$$\langle W, Z \rangle_{N_p} := \int\limits_{-1,N_p}^{1} W^T Z \,\mathrm{d}\xi.$$

The norm that is induced by that inner product will be denoted by

$$\|W\|_{N_p} := \langle W, W \rangle_{N_p}^{1/2}.$$

It is the discrete analogue to the $\mathbb{L}^2$-norm. The semi discrete approximation of the weak formulation (60) of the intrinsic beam equation on the cell C_{k} finally reads:

$$\frac{\Delta x_{\mathrm{k}}}{2} \langle \phi, \Gamma^{(\mathrm{k})} U_t^{(\mathrm{k})} \rangle_{N_p} - \langle \Pi\phi_\xi, U^{(\mathrm{k})} \rangle_{N_p} - \frac{\Delta x_{\mathrm{k}}}{2} \langle \phi, Q_{cap}^{(\mathrm{k})} \rangle_{N_p} = -\phi^T \mathcal{F}^* \Big|_{-1}^{1}. \tag{61}$$

Remark 2 *Note that even though we formally approximate Γ and $Q_{cap}(U^{(\mathrm{k})})$ by their respective interpolant, in the context of the LGL quadrature, $\Gamma^{(k)}$ can be replaced by Γ. The same holds for Q_{cap} and its interpolant. This is because of the co-location of interpolation nodes and quadrature nodes that has the effect that evaluations only take place at interpolation nodes, where the interpolation is exact. In the following, we will therefore write Γ and $Q_{cap}(U^{(\mathrm{k})})$ instead of $\Gamma^{(\mathrm{k})}$ and $Q_{cap}^{(\mathrm{k})}$ in the context of discrete scalar products and norms.*

3.3 The Numerical Flux

In the following section, we want to give an overview about the principle idea that underlies the approximations of the surface flux, i.e. the evaluation of the flux function at the cell boundaries, by a numerical flux and, subsequently, specify the numerical flux that is used in the rest of this thesis. The part regarding the overview about the idea behind the numerical flux and the associated Riemann Problems is guided by [31, 50] while the choice of the numerical flux is inspired by [26].

To appropriately define the numerical flux later in this section, we will need some new notations, that are also guided by [26]. We start by introducing the notation for the absolute value of a diagonal matrix. Let therefore D be an arbitrary real diagonal matrix. Then its absolute value $|D|$ is determined by taking the absolute value of every diagonal entry, i.e.

$$|D| := \mathrm{diag}(|d_{11}|, |d_{22}|, \dots),$$

where d_{ii} are the diagonal entries of D. Furthermore, we define the matrices D_+ and D_- as follows:

$$D_+ := \frac{1}{2}(D + |D|),$$
$$D_- := \frac{1}{2}(D - |D|).$$

This results in D_+ containing the positive entries of D on its diagonal and negative ones being replaced by zero. Likewise, D_- contains the negative values of D and the positive ones are replaced by zero. Consequently, we have

$$D = D_+ + D_-,$$

which can also be checked easily using the definition of D_+ and D_-.

These definitions can be extended to non diagonal but diagonalizable matrices. In particular, let $\hat{A}$ be a real matrix that is diagonalized by a transformation with $\hat{T}$ and let D be the corresponding diagonal matrix containing the eigenvalues of $\hat{A}$. Then by $|\hat{A}|$, we denote the matrix

$$|\hat{A}| := \hat{T}^{-1}|D|\hat{T},$$

which can be interpreted as the matrix that results from replacing all eigenvalues of $\hat{A}$ by their absolute value. The matrices $\hat{A}_+$ and $\hat{A}_-$ can then be defined analogously to the diagonal case, meaning

$$\hat{A}_+ := \frac{1}{2}(\hat{A} + |\hat{A}|),$$
$$\hat{A}_- := \frac{1}{2}(\hat{A} - |\hat{A}|).$$

Note that the relations

$$\hat{A} = \hat{A}_+ + \hat{A}_-$$
$$\hat{A}_\pm = \hat{T}^{-1}D_\pm\hat{T}$$

hold.

Having completed these preparatory considerations, we can now come to the actual derivation of the numerical flux. As mentioned in the discretization procedure in the previous section, for a numerical solution that is discontinuous at the cell boundaries, the evaluation of the flux function f at these points is not defined. Hence, an appropriate approximation is needed in order to realize the discretization scheme as it was introduced in section 3.2. The numerical flux is used to obtain such an approximation. Similar to Finite Volume Methods, the idea is to interpret the two values at each side of an interface as initial condition of a local *Riemann Problem*. This idea goes back to Godunov [16].

A Riemann problem is an initial value problem with an initial condition that has a jump at one point. Since the derived partial differential equation holds in every cell, and the numerical solution usually jumps at an interface, in the form we defined it in the previous section, there emerges a local Riemann problem at every interface. To be precise, we have to distinguish between Riemann problems and *generalized* Riemann problems. The former consider initial conditions that are constant apart from the jump, while the latter also consider non-constant initial conditions with a jump. Because the numerical solution is non-constant in the cells, the local Riemann problems that emerge at the interfaces are generalized Riemann problems. Nevertheless, in the following we refer to them simply as Riemann problems.

Thus, to approximate the flux value at the interfaces, the local Riemann problem can be solved numerically to insert the solution into the flux function afterwards. This is why the numerical flux is often called *Riemann Solver* or more precisely *approximate Riemann Solver*. Theoretical considerations on how a Riemann problem can be solved (numerically) are for example given in [50]. In this thesis we will not look into the details on solving the Riemann problem itself but limit ourselves to specifying a widely used Riemann solver as numerical flux.

Therefore, we introduce another notation. Let us consider one arbitrary but fixed interface between the cells C_k and $C_{\mathrm{k}+1}$ for some $\mathrm{k} \in \{1, \dots, N_c - 1\}$. Then the numerical solution's value at this interface is $U^{(\mathrm{k})}(\xi = 1)$ as seen from the cell C_k and $U^{(\mathrm{k}+1)}(\xi = -1)$ as seen from the cell $C_{\mathrm{k}+1}$ (omitting the dependency on the time). As long as we are operating at one fixed interface, in the following we will denote the value of the numerical solution left, respectively right to the interface as

$$U_L := U^{(\mathrm{k})}(\xi = 1),$$
$$U_R := U^{(\mathrm{k}+1)}(\xi = -1).$$

The evaluation of a function at the respective interface is furthermore denoted by the superscript $(\cdot)^*$, meaning that we write for instance A^* for the evaluation of A at the interface. This notation will only be used for smooth functions, so that the value at the interface is always well defined. A simplified illustration of the definition of U_L, U_R and A^* is given in figure 2.

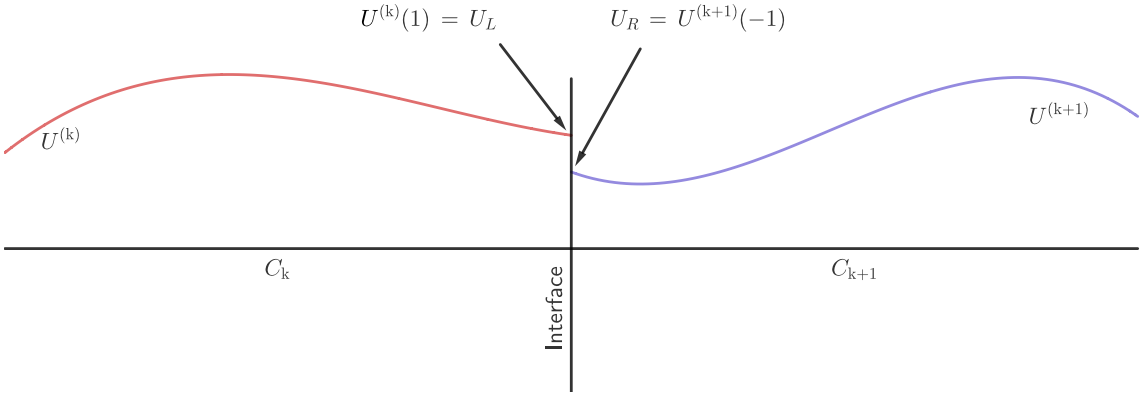


Figure 2: Simplified illustration of the definition of U_L and U_R. Here, for illustration purposes, U is an one dimensional quantity whereas the actual U is twelve dimensional. The matrix A^* is the evaluation of A at the interface between C_k and C_{k+1}.

Although we are guided by [26] in choosing the numerical flux, we have to slightly adjust the numerical flux that is chosen there. The reason for this is, that in this thesis, we discretize the capacity form of the advection equation while in [26] the advection form[1] is discretized. Nevertheless, the procedures can be adapted.

In [26], Gassner and Kopriva choose the so called *Lax-Friedrichs Flux* that originates from Lax's work [30] in 1954 and is still a widely used numerical flux for Discontinuous Galerkin discretization methods (cf. for example [8, 11, 14, 18, 26]). The discretized problem in [26] is the advection form[1] of a linear hyperbolic balance law with a symmetric advection matrix and a linear source term. Apart from the source term, the intrinsic beam equation for characteristic variables derived in section 2.4.3, is such an equation. For this formulation, we can thus specify the Lax-Friedrichs flux analogously to [26]. Adapted to our notation, it reads

$$\bar{\mathcal{F}}^*(W_L, W_R) := \frac{1}{2}\mathbf{\Lambda}^*(W_L + W_R) - \frac{\sigma}{2}|\mathbf{\Lambda}^*|(W_R - W_L),$$

where $\sigma \geq 0$ is a so called *upwind parameter* and W_L and W_R are the values of the numerical solution at the respective interface when discretizing the formulation for characteristic variables. For $\sigma = 0$ the flux is also called *central flux* as it computes the average of the flux for the two states left and right to an interface. For $\sigma = 1$ on the other hand, we have

$$\bar{\mathcal{F}}^*(W_L, W_R) := \mathbf{\Lambda}^*_+ W_L + \mathbf{\Lambda}^*_- W_R,$$

meaning that in this case, only values from the left that are associated with positive characteristic speeds and only values from the right that are associated with negative characteristic speeds, are taken into account. The numerical flux resulting from $\sigma = 1$ is therefore called *upwind flux*.

In [26] it is shown that the Discontinuous Galerkin discretization for the problem, considered there, is energy stable using the Lax-Friedrichs flux with $\sigma \geq 0$. After deriving this energy stable numerical flux for the symmetric system, Gassner and Kopriva also show that it can be transferred to a non symmetric system if the corresponding advection matrix is symmetrizable. This is done by transforming the numerical flux function just in the inverse way, the non symmetric system is transformed into the equivalent symmetric system (remember that the advection matrix was assumed to be symmetrizable).

The symmetrization process in [26] is the same as the diagonalization process in this thesis in section 2.4.3. There, we found a transformation matrix T, that diagonalizes (symmetrizes) the advection matrix, inserted $u = Tw$ into the non diagonal (non symmetric) system and multiplied both sides of the equation by T^{-1}. Applying the inverse transformation to the above Lax-Friedrich flux for the diagonal (symmetric) system, means inserting $W_L = (T^{-1})^* U_L$ respectively $W_R =$

[1] Actually a split form, i.e. a weighted average of the advection form and the conservation form is discretized in [26]. Nevertheless, the weight can be chosen in such a way that only the advection form is used.

$(T^{-1})^* U_R$ and multiplying by T^* from the left. In particular, this yields

$$\begin{aligned}\hat{\mathcal{F}}^*(U_L, U_R) &:= T^* \bar{\mathcal{F}}^*((T^{-1})^* U_L, (T^{-1})^* U_R) \\ &= \frac{1}{2}(T\mathbf{\Lambda}T^{-1})^*(U_L + U_R) - \frac{\sigma}{2}(T|\mathbf{\Lambda}|T^{-1})^*(U_R - U_L) \\ &= \frac{1}{2}A^*(U_L + U_R) - \frac{\sigma}{2}|A|^*(U_R - U_L).\end{aligned}$$

Eventually, in [26] it is shown that the discretization scheme for the non symmetric system and the transformed numerical flux $\hat{\mathcal{F}}^*$ is still energy stable.

Up to this point the proceedings are completely analogue to the ones in [26]. The additional step is now to transform the numerical flux for the non symmetric system in advection form to a numerical flux for the discretization of the system in capacity form. Just as in the first transformation, the same steps that are done to transfer the system from the advection form to the capacity form are done to the numerical flux. In particular this is the multiplication by the matrix Γ^* from the left, meaning that we set

$$\begin{aligned}\mathcal{F}^*(U_L, U_R) &:= \Gamma^* \hat{\mathcal{F}}^*(U_L, U_R) \\ &= \frac{1}{2}\Gamma^* A^*(U_L + U_R) - \frac{\sigma}{2}\Gamma^*|A|^*(U_R - U_L) \\ &= \frac{1}{2}\Pi(U_L + U_R) - \frac{\sigma}{2}(\Gamma|A|)^*(U_R - U_L).\end{aligned}$$

Note that $\mathcal{F}^*(U, U) = \Pi U = f(U)$, so that the numerical flux is consistent, which is the minimum requirement to a numerical flux. In section 3.5.2, we will show that the discretization scheme from the last section together with the above numerical flux leads to an energy stable numerical solution.

Analogous to the considerations on the upwind parameter in the numerical flux for the system for characteristic variables, earlier in this section, we refer to the numerical flux $\mathcal{F}^*$ with $\sigma = 0$, i.e.

$$\mathcal{F}^*(U_L, U_R) = \frac{1}{2}\Pi(U_L + U_R)$$

as central flux while the numerical flux that results from choosing $\sigma = 1$, i.e.

$$\mathcal{F}^*(U_L, U_R) = \frac{1}{2}\Pi(U_L + U_R) - \frac{1}{2}(\Gamma|A|)^*(U_R - U_L),$$

will be called upwind flux. Another representation of the upwind flux is obtained by inserting $\Pi = (\Gamma A)^*$ again and gathering terms. In particular, the upwind flux can be written as

$$\mathcal{F}^*(U_L, U_R) = (\Gamma A_+)^* U_L + (\Gamma A_-)^* U_R.$$

While the numerical results in section 4 will be computed by using either of these two special cases, for the theoretical considerations that follow, we will leave σ as a parameter. The only limitation we set ourselves is that at the boundaries, we always set $\sigma = 1$, i.e. take the upwind flux. This is analogue to the literature, e.g. [26].

The treatment of the boundaries is not only special with regard to the upwind parameter but also because the cells adjacent to the boundaries do not have two neighbours like the inner cells do. More precisely, the first cell does not have a left neighbour cell, whereas the last cell does not have a right neighbour cell. Nevertheless, the numerical flux does need a value from the left and from the right at the boundaries. The implementation of the boundary conditions into the discretization scheme will be discussed in the next section.

To complete this section, we introduce another notation, that emerges from the notation in the previous section. The right hand side of the semi discrete formulation, we derived there, reads

$$-\phi^T \mathcal{F}^* \Big|_{-1}^{1},$$

which is always in the context of a cell C_{k}. The evaluations of $\mathcal{F}^*$ at $\xi = \pm 1$ have therefore to be interpreted as the evaluations of the numerical flux function at the respective interface. In particular, at $\xi = -1$, we have

$$\mathcal{F}^* \big|_{-1} = \mathcal{F}^* \left(U^{(\mathrm{k}-1)}(1), U^{(\mathrm{k})}(-1) \right) \tag{62}$$

as in the notation U_L and U_R, at this particular interface, it holds

$$U_L = U^{(\mathrm{k}-1)}(1),$$
$$U_R = U^{(\mathrm{k})}(-1).$$

At $\xi = +1$ on the other hand, the evaluation has to be interpreted as

$$\mathcal{F}^*\big|_{+1} = \mathcal{F}^*\left(U^{(\mathrm{k})}(1), U^{(\mathrm{k}+1)}(-1)\right) \tag{63}$$

because U_L and U_R at this interface are defined as

$$U_L = U^{(\mathrm{k})}(1),$$
$$U_R = U^{(\mathrm{k}+1)}(-1).$$

3.4 Implementation of Boundary Conditions

In this section, we will discuss how the boundary conditions for the intrinsic beam model, we derived in section 2.5, can be implemented into the discretization approach. Therefore, the procedures in [18, 31, 50] for dealing with the boundary conditions are adapted.

In Discontinuous Galerkin discretizations, boundary conditions are typically implemented, using the numerical flux. This means that we will not set the coefficients of the numerical solution directly and reduce the degrees of freedom so that the boundary conditions are fulfilled exactly. Instead, we will define an outer value at the boundaries according to the boundary conditions, that can then be used to evaluate the numerical flux at the boundaries. This approach is also called *weak imposition of boundary conditions*.

The idea can be described as follows: At each interface (respectively boundary) the numerical flux we defined in the previous subsection, depends on a value from the left adjoined cell and from the right adjoined cell. At the boundaries, one of the values is not naturally given as for example the first cell C_1 does not have a neighbour to its left. Hence, the value on the respective outside of the domain, that is inserted into the numerical flux has to be set according to the boundary conditions. This procedure can be illustrated by introducing two fictitious cells C_0 and C_{N_c+1} that are attached to the domain $[0, \ell]$, as it is illustrated in figure 3.

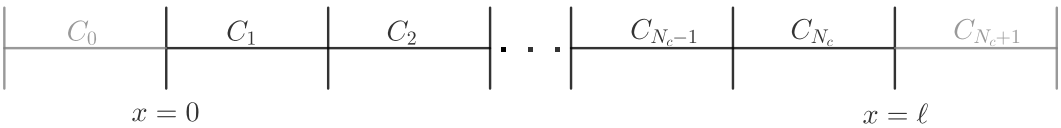


Figure 3: Illustration of the fictitious cells C_0 and C_{N_c+1} outside the interval $[0, \ell]$ inspired by Fig. 7.1. in [31].

The boundaries can now be interpreted as interfaces between the cells C_0 and C_1 or C_{N_c} and C_{N_c+1}, respectively. At the left boundary, i.e. $x = 0$, to evaluate the numerical flux we need the value $U_R = U^{(1)}(-1)$. Furthermore, a value, U_L from the left cell of the interface, namely the fictitious cell C_0 is needed. This is the value we need to set according to the boundary condition at $x = 0$. Analogously, at the right boundary, i.e. at $x = \ell$ an inner value $U_L = U^{(N_c)}(1)$ and an outer value, U_R, from the fictitious cell C_{N_c+1} is needed. The latter one is the one we have to set according to the boundary condition at $x = \ell$.

We have to set the outer values to be consistent on the one hand, meaning that if the numerical solution converges, the boundary values converge to the specified boundary values from section 2.5. On the other hand, the resulting boundary terms, i.e. the evaluation of the numerical flux at the boundaries must have a dissipative effect on the discrete energy. In this section we will focus on deriving consistent outer values. When deriving a discrete energy estimation for the numerical solution in section 3.5.2, we will then go into detail regarding the dissipative effect of the resulting boundary terms.

Although by assumption, the external boundary data y_0 and s_ℓ is zero, we will derive the discrete boundary conditions for the general case of non-zero boundary data and will then reduce

this general case to zero boundary data. Recall, that the boundary conditions for physical variables in general were derived as

$$u(0) = \begin{pmatrix} s(0) \\ y_0 \end{pmatrix}, \qquad u(\ell) = \begin{pmatrix} s_\ell \\ y(\ell) \end{pmatrix}.$$

To have a consistent notation, we adopt the subdivision of the exact solution u into s and y for the numerical solution. In particular, we set

$$U^{(k)} = \begin{pmatrix} S^{(k)} \\ Y^{(k)} \end{pmatrix}.$$

To impose the boundary conditions at $x = 0$, we set the outer value U_L in the fictitious cell C_0 as follows:

$$U_L = \begin{pmatrix} S_R + (\mathbf{F}^{-1/2}\mathcal{X}^T\Lambda^{-1}\mathcal{X}\mathbf{F}^{-1/2})^*(Y_R - y_0) \\ y_0 \end{pmatrix}, \tag{64}$$

where we take on the notation of the previous section, meaning that $(\,\cdot\,)^*$ indicates the evaluation at the considered interface, which is the boundary $x = 0$ in this case. Note, that if we replace S_R by $s(0)$ and Y_R by $y(0)$, this term already appeared in the derivation of the original boundary conditions in section 2.5 (cf. (30)). To ease the notation, we recall the definition of the matrix $\Psi = \mathbf{F}^{-1/2}\mathbf{M}^{-1}\mathbf{F}^{-1/2}$ from section 2.4.2 and especially the fact that it is diagonalized by the orthogonal matrix $\mathcal{X}$, as

$$\Psi = \mathcal{X}^T\Lambda^2\mathcal{X}$$

holds. Because Ψ was shown to be positive definite, it has a square root. In particular, this square root can be specified as

$$\Psi^{1/2} = \mathcal{X}^T\Lambda\mathcal{X},$$

which can be checked easily by multiplying $\Psi^{1/2}$ by itself. Moreover, the inverse of $\Psi^{1/2}$ is given by

$$\Psi^{-1/2} = \mathcal{X}^T\Lambda^{-1}\mathcal{X},$$

which can also be checked easily. Using these considerations on the matrix Ψ, the vector U_L in (64) can be written as

$$U_L = \begin{pmatrix} S_R + (\mathbf{F}^{-1/2}\Psi^{-1/2}\mathbf{F}^{-1/2})^*(Y_R - y_0) \\ y_0 \end{pmatrix}.$$

This choice of the outer value at the left boundary is consistent because, as the numerical solution converges to the exact solution of the problem, we have $S_R = s(0)$ and $Y_R = y(0) = y_0$ and, thus

$$U_L = \begin{pmatrix} s(0) \\ y_0 \end{pmatrix},$$

which is just the boundary condition, we derived in section 2.5 for the boundary $x = 0$. Note that for $y_0 = \mathbf{0}_6$, this reduces to

$$U_L = \begin{pmatrix} s(0) \\ \mathbf{0}_6 \end{pmatrix}.$$

Analogously, to impose the boundary conditions at the boundary where $x = \ell$, we set the outer value at this boundary, U_R, i.e. the value in the fictitious cell C_{N_c+1}, as

$$U_R = \begin{pmatrix} s_\ell \\ Y_L - (\mathbf{F}^{1/2}\mathcal{X}^T\Lambda\mathcal{X}\mathbf{F}^{1/2})^*(S_L - s_\ell) \end{pmatrix}.$$

By using the considerations on the matrix Ψ and its square root, U_R can be represented by

$$U_R = \begin{pmatrix} s_\ell \\ Y_L - (\mathbf{F}^{1/2}\Psi^{1/2}\mathbf{F}^{1/2})^*(S_L - s_\ell) \end{pmatrix}. \tag{65}$$

This choice for the outer value at the right boundary is also consistent. For a converging numerical solution, we have $S_L = s(\ell) = s_\ell$ and $Y_L = y(\ell)$, which if we insert it into (65), results in

$$U_R = \begin{pmatrix} s_\ell \\ y(\ell) \end{pmatrix}.$$

This is again just the boundary condition for the original problem at $x = \ell$. For the special case $s_\ell = \mathbf{0}_6$, the outer state U_R reduces to

$$U_R = \begin{pmatrix} \mathbf{0}_6 \\ y(\ell) \end{pmatrix}.$$

After deriving a possibility to impose the boundary conditions by using the numerical flux, with consistent choices of the outer values in the respective fictitious cells C_0 and C_{N_c+1}, we have now everything that is needed to analyze the discrete energy of the numerical solution. This will also include the verification that the boundary conditions, in the way we implemented them into the scheme, have a dissipative effect.

3.5 Discrete Energy Considerations

After deriving the semi discrete formulation of the equation, choosing a numerical flux and implementing the boundary conditions into the discretization scheme in the previous sections, we are now able to analyze the discrete energy of the numerical solution. We will proceed mostly analogous to the considerations on the energy of the solution of the continuous problem. In the first part of the section we will therefore derive a statement about energy conservation for the discrete solution. Afterwards, we will use the assumptions, we made to obtain energy stability in the continuous case, to derive an analogous statement about discrete energy stability.

3.5.1 Discrete Energy Conservation

Analogous to the considerations in the continuous case we define the discrete inner product of two polynomials W, Z within the cell C_k as

$$\langle W, Z\rangle_{N_p,\Gamma} := \langle W, \Gamma Z\rangle_{N_p} = \int_{-1,N_p}^{1} W^T \Gamma Z \,\mathrm{d}\xi$$

and the induced *discrete energy norm*

$$\|W\|_{N_p,\Gamma} := \langle W, W\rangle_{N_p,\Gamma}^{1/2}.$$

In the following, we will analyze the energy cell wise and therefore consider an arbitrary fixed cell C_k. The superscript (k) is omitted for a better readability and we will write $U = U^{(\mathrm{k})}$ for example. The semi discrete formulation of the intrinsic beam equation in the considered cell then reads

$$\frac{\Delta x_\mathrm{k}}{2}\big\langle \phi, \Gamma U_t\big\rangle_{N_p} - \big\langle \Pi\phi_\xi, U\big\rangle_{N_p} - \frac{\Delta x_\mathrm{k}}{2}\big\langle \phi, Q_{cap}(U)\big\rangle_{N_p} = -\phi^T \mathcal{F}^*\Big|_{-1}^{1}. \tag{66}$$

Equation (66) holds for every test function ϕ in the discrete space of test functions $\mathcal{V}_{N_p}$, which especially includes $\phi = U$. We can therefore insert $\phi = U$ into equation (66) to obtain

$$\frac{\Delta x_\mathrm{k}}{2}\big\langle U, \Gamma U_t\big\rangle_{N_p} - \big\langle \Pi U_\xi, U\big\rangle_{N_p} - \frac{\Delta x_\mathrm{k}}{2}\big\langle U, Q_{cap}(U)\big\rangle_{N_p} = -U^T \mathcal{F}^*\Big|_{-1}^{1}. \tag{67}$$

Analogous to the continuous energy analysis, the first term on the above left hand side can be represented in terms of the discrete energy of U:

$$\big\langle U, \Gamma U_t\big\rangle_{N_p} = \int_{-1,N_p}^{1} U^T \Gamma U_t \,\mathrm{d}\xi = \frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}t}\int_{-1,N_p}^{1} U^T \Gamma U \,\mathrm{d}\xi = \frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}t}\|U\|_{N_p,\Gamma}^2. \tag{68}$$

Furthermore we take advantage of the so called *summation by parts (SBP)* property of the LGL quadrature [25, 26]. It mimics integration by parts in the discrete context. Hence, the second summand in (67) can be written as

$$-\left\langle \Pi U_\xi, U \right\rangle_{N_p} = -U^T \Pi U \Big|_{-1}^{1} + \left\langle \Pi U, U_\xi \right\rangle_{N_p}. \tag{69}$$

Remember that Π is symmetric, which especially implicates that

$$\left\langle \Pi U, U_\xi \right\rangle_{N_p} = \left\langle \Pi U_\xi, U \right\rangle_{N_p}.$$

Thus, equation (69) can be equivalently transformed to

$$-\left\langle \Pi U_\xi, U \right\rangle_{N_p} = -\frac{1}{2} U^T \Pi U \Big|_{-1}^{1}.$$

Inserting the representation of the first summand by the discrete energy, (68), and the discrete integration by parts for the second summand, (69), into (67) and rearranging terms yields

$$\frac{\Delta x_\mathrm{k}}{4} \frac{\mathrm{d}}{\mathrm{d}t} \|U\|^2_{N_p,\Gamma} = \frac{\Delta x_\mathrm{k}}{2} \left\langle U, Q_{cap}(U) \right\rangle_{N_p} - \left(U^T \mathcal{F}^* - \frac{1}{2} U^T \Pi U \right) \Big|_{-1}^{1}, \tag{70}$$

which is an equation that describes the change of the discrete energy of the numerical solution in one cell. Analogous to the continuous case, we will now show that the term, emerging from the source term Q_{cap}, vanishes right down to external sources. The quadrature for this term reads

$$\left\langle U, Q_{cap}(U) \right\rangle_{N_p} = \sum_{j=0}^{N_p} U^T Q_{cap}(U) \big|_{\xi_j} \, \omega_j.$$

Note that the above sum actually contains the evaluations of the interpolant of $Q_{cap}(U)$ but they are the same as the evaluations of the exact $Q_{cap}(U)$ at the interpolation nodes (cf. remark 2). In section 2.6, we showed that for any $z \in \mathbb{R}^{12}$, it holds

$$z^T Q_{cap}(z) = z^T q_{ext}.$$

An implication of this is that especially

$$U^T Q_{cap}(U) \Big|_{\xi_j} = U^T q_{ext} \Big|_{\xi_j}$$

for all $j \in \{0, \ldots, N_p\}$ and therefore

$$\left\langle U, Q_{cap}(U) \right\rangle_{N_p} = \left\langle U, q_{ext} \right\rangle_{N_p}$$

holds. Thus, equation (70) reduces to

$$\frac{\Delta x_\mathrm{k}}{4} \frac{\mathrm{d}}{\mathrm{d}t} \|U\|^2_{N_p,\Gamma} = \frac{\Delta x_\mathrm{k}}{2} \left\langle U, q_{ext} \right\rangle_{N_p} - \left(U^T \mathcal{F}^* - \frac{1}{2} U^T \Pi U \right) \Big|_{-1}^{1}. \tag{71}$$

This is the cell wise and discrete analogue to the energy conservation statement, we derived in section 2.6.1 for the solution of the continuous problem. It states that energy in each cell only changes through the cell boundaries or due to the presence of external forces and moments, acting on the beam within the cell. In the next section the discrete energy considerations are extended to the global domain $[0, \ell]$, and a statement about discrete energy stability is derived.

3.5.2 Discrete Energy Stability

In the following section we want to derive energy stability for the numerical solution in the discrete energy norm. To obtain a result that is analogous to the considerations in the continuous case, we assume the external sources to be constant in time and bounded in the $\mathbb{L}^2$-norm. Furthermore, we consider zero external boundary data only.

The discrete energy conservation statement (71) from the previous section reads

$$\frac{\Delta x_{\mathrm{k}}}{4}\frac{\mathrm{d}}{\mathrm{d}t}\|U^{(\mathrm{k})}\|^2_{N_p,\Gamma} = \frac{\Delta x_{\mathrm{k}}}{2}\langle U^{(\mathrm{k})}, q_{ext}\rangle_{N_p} - \left((U^{(\mathrm{k})})^T\mathcal{F}^* - \frac{1}{2}(U^{(\mathrm{k})})^T\Pi U^{(\mathrm{k})}\right)\Big|_{-1}^{1}, \tag{72}$$

where we added the superscript (k) again to stress that the above equation holds within one cell. This means that each cell has a contribution to the total discrete energy on the interval $[0,\ell]$ of the form (72). Consequently, we have to sum up the contributions of each cell in order to measure the total discrete energy. The summation reads

$$\frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}t}\sum_{\mathrm{k}=1}^{N_c}\frac{\Delta x_{\mathrm{k}}}{2}\|U^{(\mathrm{k})}\|^2_{N_p,\Gamma} = \sum_{\mathrm{k}=1}^{N_c}\frac{\Delta x_{\mathrm{k}}}{2}\langle U^{(\mathrm{k})}, q_{ext}\rangle_{N_p} - \sum_{\mathrm{k}=1}^{N_c}\left((U^{(\mathrm{k})})^T\mathcal{F}^* - \frac{1}{2}(U^{(\mathrm{k})})^T\Pi U^{(\mathrm{k})}\right)\Big|_{-1}^{1}. \tag{73}$$

We define the *total discrete energy norm* of the global numerical solution U as

$$\|U\|_{N_p,\Gamma} := \left(\sum_{\mathrm{k}=1}^{N_c}\frac{\Delta x_{\mathrm{k}}}{2}\|U^{(\mathrm{k})}\|^2_{N_p,\Gamma}\right)^{1/2}.$$

Consequently, the *total discrete energy* of the global numerical solution is defined as

$$\|U\|^2_{N_p,\Gamma} = \sum_{\mathrm{k}=1}^{N_c}\frac{\Delta x_{\mathrm{k}}}{2}\|U^{(\mathrm{k})}\|^2_{N_p,\Gamma},$$

so that (73) can be interpreted as

$$\frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}t}\|U\|^2_{N_p,\Gamma} = \sum_{\mathrm{k}=1}^{N_c}\frac{\Delta x_{\mathrm{k}}}{2}\langle U^{(\mathrm{k})}, q_{ext}\rangle_{N_p} - \sum_{\mathrm{k}=1}^{N_c}\left((U^{(\mathrm{k})})^T\mathcal{F}^* - \frac{1}{2}(U^{(\mathrm{k})})^T\Pi U^{(\mathrm{k})}\right)\Big|_{-1}^{1}. \tag{74}$$

Before we analyze the total discrete energy, we take a more detailed look at the term

$$\sum_{\mathrm{k}=1}^{N_c}\left((U^{(\mathrm{k})})^T\mathcal{F}^* - \frac{1}{2}(U^{(\mathrm{k})})^T\Pi U^{(\mathrm{k})}\right)\Big|_{-1}^{1}, \tag{75}$$

that occurs on the right hand side of (74). Note that the evaluation of the k-th summand at $\xi = +1$ refers to the interface between the cells C_{k} and $C_{\mathrm{k}+1}$ and so does the evaluation of the (k + 1)-st summand at $\xi = -1$. The evaluations of the numerical flux in these summands are therefore equal (cf. definition of the evaluations of $\mathcal{F}^*$ at the interfaces in (62), (63)). This means that, if we arrange the sum in such a way that summands referring to the same interface follow each other, terms of the following form result:

$$\begin{aligned}\left(U^{(\mathrm{k})}(1) - U^{(\mathrm{k}+1)}(-1)\right)^T&\mathcal{F}^*(U_L,U_R)\\ &- \frac{1}{2}\left(\left(U^{(\mathrm{k})}(1)\right)^T\Pi U^{(\mathrm{k})}(1) - (U^{(\mathrm{k}+1)}(-1))^T\Pi U^{(\mathrm{k}+1)}(-1)\right),\end{aligned} \tag{76}$$

where U_L, U_R represent the values of the numerical solution left and right to the respective interface. To ease the notation, we define the jump of the solution at the interface between the cells C_{k} and $C_{\mathrm{k}+1}$ as

$$\left[U^{(\mathrm{k})}\right] := U^{(\mathrm{k})}(1) - U^{(\mathrm{k}+1)}(-1).$$

As long as it is clear, which interface we refer to, in the notation of section 3.3 this can also be written as

$$\left[U^{(\mathrm{k})}\right] = U_L - U_R.$$

This notation can analogously be transferred to $[(U^{(\mathrm{k})})^T\Pi U^{(\mathrm{k})}]$. The terms in (76) can then be written as

$$\left[U^{(\mathrm{k})}\right]^T\mathcal{F}^*(U_L,U_R) - \frac{1}{2}\left[(U^{(\mathrm{k})})^T\Pi U^{(\mathrm{k})}\right].$$

If we sum these terms up over all $\mathrm{k} = 1, \ldots, N_c - 1$, every summand from (75) can be found in the resulting sum except the two terms

$$b_L := -\left((U^{(1)})^T \mathcal{F}^* - \frac{1}{2}(U^{(1)})^T \Pi U^{(1)} \right)\Big|_{-1},$$

$$b_R := \left((U^{(\mathrm{N_c})})^T \mathcal{F}^* - \frac{1}{2}(U^{(\mathrm{N_c})})^T \Pi U^{(\mathrm{N_c})} \right)\Big|_{1},$$

which refer to the boundaries. In particular, b_L refers to the left boundary and b_R to the right one. Putting everything together, the sum (75) can be represented by

$$\sum_{\mathrm{k}=1}^{N_c} \left((U^{(\mathrm{k})})^T \mathcal{F}^* - \frac{1}{2}(U^{(\mathrm{k})})^T \Pi U^{(\mathrm{k})} \right)\Big|_{-1}^{1} = \sum_{\mathrm{k}=1}^{N_c-1} \left(\left[U^{(\mathrm{k})}\right]^T \mathcal{F}^*(U_L, U_R) - \frac{1}{2}\left[(U^{(\mathrm{k})})^T \Pi U^{(\mathrm{k})}\right] \right) + b_L + b_R.$$

This is now inserted into equation (72), so that the change of total discrete energy becomes

$$\frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}t}\|U\|^2_{N_p,\Gamma} = \sum_{\mathrm{k}=1}^{N_c} \frac{\Delta x_{\mathrm{k}}}{2}\left\langle U^{(\mathrm{k})}, q_{ext}\right\rangle_{N_p} - \sum_{\mathrm{k}=1}^{N_c-1} \left(\left[U^{(\mathrm{k})}\right]^T \mathcal{F}^*(U_L, U_R) - \frac{1}{2}\left[(U^{(\mathrm{k})})^T \Pi U^{(\mathrm{k})}\right] \right) - b_L - b_R. \tag{77}$$

The above right hand side can be interpreted as follows: The first sum is the contribution of the external source term to the discrete energy, the second sum represents the energy contribution at the interfaces between the cells within the domain $(0, \ell)$ and the last two terms, namely b_L and b_R are the contribution of the boundary terms to the total discrete energy. Every change in total discrete energy can be traced back to one of these terms. Similar to the continuous case, to obtain energy stability, we consider the right hand side of (77) term by term and try to find a bound for each, which we do in the following paragraphs.

Contribution of the interface terms First, we consider the contribution of the interface terms. In particular, we start with the term

$$\left[U^{(\mathrm{k})}\right]^T \mathcal{F}^*(U_L, U_R)$$

for a fixed k. According to the definition of the general numerical flux with upwind parameter σ, this can be expressed as

$$\left[U^{(\mathrm{k})}\right]^T \mathcal{F}^*(U_L, U_R) = \frac{1}{2}(U_L - U_R)^T \Pi (U_L + U_R) - \frac{\sigma}{2}(U_L - U_R)^T (\Gamma|A|)^* (U_R - U_L),$$

where we use the notation $U_L = U^{(\mathrm{k})}(1)$ and $U_R = U^{(\mathrm{k}+1)}(-1)$ again. Now, out multiplying the brackets in the first summand while using the symmetry of Π and changing the sign of the second summand by switching U_L and U_R in the last brackets yields

$$\left[U^{(\mathrm{k})}\right]^T \mathcal{F}^*(U_L, U_R) = \frac{1}{2}U_L^T \Pi U_L - U_R^T \Pi U_R + \frac{\sigma}{2}(U_L - U_R)^T (\Gamma|A|)^* (U_L - U_R).$$

The notation for the jump at an interface, $[\,\cdot\,]$, that we defined earlier in this section can be used to represented the last result by

$$\left[U^{(\mathrm{k})}\right]^T \mathcal{F}^*(U_L, U_R) = \frac{1}{2}\left[(U^{(\mathrm{k})})^T \Pi U^{(\mathrm{k})}\right] + \frac{\sigma}{2}\left[U^{(\mathrm{k})}\right]^T (\Gamma|A|)^* \left[U^{(\mathrm{k})}\right].$$

The contribution of one interface, i.e. the k-th summand of the corresponding sum in (77), is therefore

$$\left[U^{(\mathrm{k})}\right]^T \mathcal{F}^*(U_L, U_R) - \frac{1}{2}\left[(U^{(\mathrm{k})})^T \Pi U^{(\mathrm{k})}\right] = \frac{\sigma}{2}\left[U^{(\mathrm{k})}\right]^T (\Gamma|A|)^* \left[U^{(\mathrm{k})}\right].$$

The matrix $\Gamma|A|$ can be determined in terms of 6×6 blocks, which is done in detail in Appendix A.2. The result of this is

$$\Gamma|A| = \begin{pmatrix} \mathbf{F}^{1/2}\Psi^{1/2}\mathbf{F}^{1/2} & \mathbf{0}_{6,6} \\ \mathbf{0}_{6,6} & \mathbf{F}^{-1/2}\Psi^{-1/2}\mathbf{F}^{-1/2} \end{pmatrix}.$$

Remember that $\mathbf{F}^{\pm 1/2}$ and $\Psi^{\pm 1/2}$ are positive definite (cf. section 2.4.2) and, thus, the diagonal blocks of $\Gamma|A|$, are positive definite, which can be checked easily and is also shown Appendix A.2. This holds for the evaluation of $\mathbf{F}^{\pm 1/2}\Psi^{\pm 1/2}\mathbf{F}^{\pm 1/2}$ at any x. This means, that especially $(\Gamma|A|)^*$ is positive definite at every interface. Moreover, by definition $\sigma \geq 0$, so every summand of the interface contribution is non negative:

$$\frac{\sigma}{2}\left[U^{(\mathrm{k})}\right]^T (\Gamma|A|)^* \left[U^{(\mathrm{k})}\right] \geq 0. \tag{78}$$

Inserting this into the total contribution of the interfaces in (77) yields

$$-\sum_{\mathrm{k}=1}^{N_c-1} \left(\left[U^{(\mathrm{k})}\right]^T \mathcal{F}^*(U_L, U_R) - \frac{1}{2}\left[(U^{(\mathrm{k})})^T \Pi U^{(\mathrm{k})}\right] \right) \leq 0.$$

The change of total discrete energy (77) can, thus, be updated by bounding the interface terms by zero, so that it becomes

$$\frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}t}\|U\|^2_{N_p,\Gamma} \leq \sum_{\mathrm{k}=1}^{N_c} \frac{\Delta x_{\mathrm{k}}}{2}\left\langle U^{(\mathrm{k})}, q_{ext}\right\rangle_{N_p} - b_L - b_R. \tag{79}$$

That is, while on a local basis, i.e. in the cells, the energy conservation still holds in the discrete context, on a global basis, we have additional numerical dissipation at the interfaces. The amount of numerical dissipation, (78), depends on the choice of the numerical flux. The bigger the upwind parameter σ is chosen, the more numerical dissipation is generated at the interfaces. Note that for the central flux, i.e. $\sigma = 0$, the left hand side of (78) is exactly zero, which is why in this specific case, no numerical dissipation at the interfaces is generated and the inequality (79) actually becomes an equality.

Contribution of the boundary terms For the estimation of the contribution of the discrete boundary terms b_L and b_R, we start with the term b_L, referring to the left boundary. Note that this term occurs as $-b_L$ in (79), which is why we actually consider $-b_L$. In the course of this, we use the notation U_L, U_R from section 3.4 again. More precisely, U_L will denote the outer value in the fictitious cell C_0, given by the discrete boundary conditions and $U_R = U^{(1)}(-1)$. By definition, $-b_L$ can then be written as

$$-b_L = U_R^T \left(\mathcal{F}^*(U_L, U_R) - \frac{1}{2}\Pi U_R \right).$$

As mentioned in section 3.3, at the boundaries of the domain, we always use the upwind flux with the upwind parameter $\sigma = 1$. By inserting the definition of the upwind flux, we obtain

$$-b_L = U_R^T \left((\Gamma A_+)^* U_L + (\Gamma A_-)^* U_R - \frac{1}{2}\Pi U_R \right).$$

The matrices ΓA_+ and ΓA_- can be determined in terms of 6×6 blocks. In particular, they read

$$\begin{aligned} \Gamma A_+ &= \frac{1}{2}\begin{pmatrix} \mathbf{F}^{1/2}\Psi^{1/2}\mathbf{F}^{1/2} & -\mathbf{I}_{6,6} \\ -\mathbf{I}_{6,6} & \mathbf{F}^{-1/2}\Psi^{-1/2}\mathbf{F}^{-1/2} \end{pmatrix}, \\ \Gamma A_- &= \frac{1}{2}\begin{pmatrix} -\mathbf{F}^{1/2}\Psi^{1/2}\mathbf{F}^{1/2} & -\mathbf{I}_{6,6} \\ -\mathbf{I}_{6,6} & -\mathbf{F}^{-1/2}\Psi^{-1/2}\mathbf{F}^{-1/2} \end{pmatrix}. \end{aligned} \tag{80}$$

The derivation of the matrices can be read up in detail in Appendix A.1. Using this result and inserting the inner value

$$U_R = \begin{pmatrix} S_R \\ Y_R \end{pmatrix}$$

as well as the outer value

$$U_L = \begin{pmatrix} S_R + (\mathbf{F}^{-1/2}\Psi^{-1/2}\mathbf{F}^{-1/2})^* Y_R \\ \mathbf{0}_6 \end{pmatrix}$$

into the boundary term $-b_L$, finally yields

$$-b_L = -Y_R^T (\mathbf{F}^{-1/2}\Psi^{-1/2}\mathbf{F}^{-1/2})^* Y_R, \tag{81}$$

or in the global notation

$$-b_L = -\left(Y^{(1)}\right)^T \mathbf{F}^{-1/2}\Psi^{-1/2}\mathbf{F}^{-1/2} Y^{(1)}\Big|_{\xi=-1}.$$

A detailed derivation of the above right hand side is also given in Appendix A.1. The matrix $\mathbf{F}^{-1/2}\Psi^{-1/2}\mathbf{F}^{-1/2}$ is already known from the previous paragraph. There, it was mentioned that at any x, the matrix is positive definite, so especially at $x = 0$, which is why the expression on the right hand side of (81) is at maximum zero and we have

$$-b_L \leq 0.$$

The term b_R, referring to the right boundary, can be treated analogously. In (79) it appears with a negative sign as well, which is why we consider $-b_R$. First, we use again the notation U_L and U_R to represent $-b_R$ as

$$-b_R = -U_L^T \left((\Gamma A_+)^* U_L + (\Gamma A^-)^* U_R - \frac{1}{2}\Pi^* U_L \right),$$

where we already inserted the definition of the upwind numerical flux. The matrices ΓA_+ and ΓA_- are given in (80) and additionally inserting the inner value

$$U_L = \begin{pmatrix} S_L \\ Y_L \end{pmatrix}$$

as well as the outer value for the right boundary,

$$U_R = \begin{pmatrix} \mathbf{0}_6 \\ Y_L - (\mathbf{F}^{1/2}\Psi^{1/2}\mathbf{F}^{1/2})^* S_L \end{pmatrix},$$

yields

$$-b_R = -S_L^T (\mathbf{F}^{1/2}\Psi^{1/2}\mathbf{F}^{1/2})^* S_L,$$

or in the global notation

$$-b_R = -\left(S^{(N_c)}\right)^T \mathbf{F}^{1/2}\Psi^{1/2}\mathbf{F}^{1/2} S^{(N_c)}\Big|_{\xi=1}.$$

The detailed derivation of the last result can also be read up in Appendix A.1. With the same argumentation as above, the matrix $(\mathbf{F}^{1/2}\Psi^{1/2}\mathbf{F}^{1/2})^*$ is positive definite, which leads to the bound

$$-b_R \leq 0.$$

Hence, both boundary terms, or more precisely their negatives in (79) can be bounded by zero, which we use to bound the change of total discrete energy by

$$\frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}t}\|U\|^2_{N_p,\Gamma} \leq \sum_{\mathrm{k}=1}^{N_c} \frac{\Delta x_{\mathrm{k}}}{2} \left\langle U^{(\mathrm{k})}, q_{ext} \right\rangle_{N_p}. \tag{82}$$

Recall, that when we derived the outer states in the fictitious cells in section 3.4 to implement the boundary conditions, we required that they are consistent on the one hand and have a dissipative effect on the energy on the other hand. While the consistence property was shown immediately, the proof of the dissipation effect was postponed to this section. In fact, this second property has

now been shown by showing that the boundary terms $-b_L$ and $-b_R$ that appear in the discrete energy estimation, can be bounded by zero.

The inequality (82) implicates that for zero external forces and moments, we have that the total discrete energy of the numerical solution is decreasing in time. This statement will be validated by numerical experiments in section 4.3.

Moreover, inequality (82) is the discrete analogue to the energy equation (49) in the continuous energy analysis. There, we had an actual equality of the time derivative of the energy and the contribution of the external source term. Here, the inequality results because of the numerical dissipation at the interfaces and at the boundaries. While in the previous paragraph, we showed that the numerical dissipation at the interfaces can be controlled by the choice of the upwind parameter σ, the boundary terms are in general negative, meaning that at the boundaries, there is always numerical dissipation generated. However, with the determination of the boundary terms in this paragraph, the amount of numerical dissipation at the boundaries can be specified.

Contribution of external sources We proceed now by estimating the right hand side of (82). Most parts are analogue to the considerations on the contribution of the external source term in the continuous energy analysis in section 2.6.2. We will use the same theorems but have to slightly adjust the constants that were used to bound the contribution. We start by applying the Cauchy-Schwarz inequality to the discrete inner product $\langle \cdot, \cdot \rangle_{N_p}$ to obtain

$$\sum_{\mathrm{k}=1}^{N_c} \frac{\Delta x_{\mathrm{k}}}{2} \langle U^{(\mathrm{k})}, q_{ext} \rangle_{N_p} \leq \sum_{\mathrm{k}=1}^{N_c} \frac{\Delta x_{\mathrm{k}}}{2} \|U^{(\mathrm{k})}\|_{N_p} \|q_{ext}\|_{N_p}.$$

Remember that the external sources were assumed to be constant in time and that $\|q_{ext}\|_{N_p} = \|I^{N_p}(q_{ext})\|_{N_p}$, can be bounded by a constant in every cell. If we further take the maximum of all these bounds in the cells and denote it by $\hat{c}_1$, we can estimate the above term by

$$\sum_{\mathrm{k}=1}^{N_c} \frac{\Delta x_{\mathrm{k}}}{2} \langle U^{(\mathrm{k})}, q_{ext} \rangle_{N_p} \leq \hat{c}_1 \sum_{\mathrm{k}=1}^{N_c} \frac{\Delta x_{\mathrm{k}}}{2} \|U^{(\mathrm{k})}\|_{N_p}. \tag{83}$$

Similar to the continuous energy analysis, we would like to trace back the discrete N_p-norm of $U^{(\mathrm{k})}$ to the discrete energy norm. This is realized by applying Rayleigh's min-max principle again. In particular, by definition of the discrete N_p-norm, we have

$$\|U^{(\mathrm{k})}\|_{N_p}^2 = \sum_{j=0}^{N_p} \omega_j (U^{(\mathrm{k})})^T U^{(\mathrm{k})} \Big|_{\xi_j}.$$

Every summand can be estimated by Raleigh's min-max principle, similar to section 2.6.2, as

$$(U^{(\mathrm{k})})^T U^{(\mathrm{k})} \Big|_{\xi_j} \leq \frac{1}{\mu_{\min}(\xi_j)} (U^{(\mathrm{k})})^T \Gamma U^{(\mathrm{k})} \Big|_{\xi_j}$$

holds for the minimal eigenvalue $\mu_{\min}(\xi_j)$ of $\Gamma(\xi_j)$. Now defining $\hat{c}_2 := \max_{j=0,\dots,N_p} {}^1\!/_{\mu_{\min}(\xi_j)}$ lets us estimate the squared N_p-norm of $U^{(\mathrm{k})}$ by

$$\|U^{(\mathrm{k})}\|_{N_p}^2 \leq \hat{c}_2 \sum_{j=0}^{N_p} \omega_j (U^{(\mathrm{k})})^T \Gamma U^{(\mathrm{k})} \Big|_{\xi_j} = \hat{c}_2 \|U^{(\mathrm{k})}\|_{N_p,\Gamma}^2.$$

Taking the square root on both sides of the above inequality yields just the estimation of the N_p-norm of U in terms of the discrete energy norm of U, we were aiming for, namely

$$\|U^{(\mathrm{k})}\|_{N_p} \leq \sqrt{\hat{c}_2}\, \|U^{(\mathrm{k})}\|_{N_p,\Gamma}.$$

Note that $\hat{c}_2$ actually depends on the respective cell. If we additionally define the following constant $\hat{c}_3 := \max_{\mathrm{k}=1,\dots,N_c} \hat{c}_1 \sqrt{\hat{c}_2}$, and insert this last result into (83) , we obtain

$$\frac{1}{2} \frac{\mathrm{d}}{\mathrm{d}t} \|U\|_{N_p,\Gamma}^2 \leq \hat{c}_3 \sum_{\mathrm{k}=1}^{N_c} \frac{\Delta x_{\mathrm{k}}}{2} \|U^{(\mathrm{k})}\|_{N_p,\Gamma}. \tag{84}$$

Due to the equivalence of p-norms, there exists another constant $\hat{c}_4$, such that the sum on the above right hand side can be bounded by

$$\sum_{\mathrm{k}=1}^{N_c} \frac{\Delta x_{\mathrm{k}}}{2} \|U^{(\mathrm{k})}\|_{N_p,\Gamma} \leq \hat{c}_4 \left(\sum_{\mathrm{k}=1}^{N_c} \left(\frac{\Delta x_{\mathrm{k}}}{2} \|U^{(\mathrm{k})}\|_{N_p,\Gamma} \right)^2 \right)^{1/2}.$$

With $\hat{c}_5 := \max_{\mathrm{k}=1,\ldots,N_c} \frac{\Delta x_{\mathrm{k}}}{2}$, this can again be estimated by

$$\hat{c}_4 \left(\sum_{\mathrm{k}=1}^{N_c} \left(\frac{\Delta x_{\mathrm{k}}}{2} \|U^{(\mathrm{k})}\|_{N_p,\Gamma} \right)^2 \right)^{1/2} \leq \hat{c}_4 \sqrt{\hat{c}_5} \left(\sum_{\mathrm{k}=1}^{N_c} \frac{\Delta x_{\mathrm{k}}}{2} \|U^{(\mathrm{k})}\|^2_{N_p,\Gamma} \right)^{1/2} = \hat{c}_4 \sqrt{\hat{c}_5} \|U\|_{N_p,\Gamma},$$

where the last equality holds by definition of the discrete total energy norm. Putting these considerations together by inserting the last result into the inequality (84), we obtain

$$\frac{1}{2} \frac{\mathrm{d}}{\mathrm{d}t} \|U\|^2_{N_p,\Gamma} \leq \hat{c}_6 \|U\|_{N_p,\Gamma},$$

where we define $\hat{c}_6 = \hat{c}_3 \hat{c}_4 \sqrt{\hat{c}_5}$. From here on, the proceedings are completely analogous to the ones in section 2.6.2 from inequality (50) on. Using Petrovitsch's theorem, the solution of the above ordinary differential inequality is

$$\|U(\cdot,t)\|^2_{N_p,\Gamma} \leq \left(\|U_0\|_{N_p,\Gamma} + \hat{c}_6 t \right)^2, \tag{85}$$

where U_0 is the interpolation of the initial condition u_0. That is, with the assumption we made about the boundary conditions and external forces, we have a growth of total discrete energy that is quadratic in time at maximum, analogous to the statement we derived about the total continuous energy in section 2.6.2. This can again be transferred to the discrete energy norm of the numerical solution, as

$$\|U(\cdot,t)\|_{N_p,\Gamma} \leq \|U_0\|_{N_p,\Gamma} + \hat{c}_6 t \tag{86}$$

holds. The equivalent estimations (85) and (86) are the main results of this thesis. For zero boundary data and in time constant, bounded external forces and moments, we derived an energy stable Discontinuous Galerkin discretization approach to discretize the intrinsic beam model in space.

Before we complete this section, we will have a more detailed look into one specific case of the derived results that we want to validate numerically in section 4. Recall that for the central flux, we found (cf. (79))

$$\frac{1}{2} \frac{\mathrm{d}}{\mathrm{d}t} \|U\|^2_{N_p,\Gamma} = \sum_{\mathrm{k}=1}^{N_c} \frac{\Delta x_{\mathrm{k}}}{2} \left\langle U^{(\mathrm{k})}, q_{ext} \right\rangle_{N_p} - b_L - b_R.$$

If we assume zero external forces and moments, the corresponding terms on the above right hand side vanish, and we have

$$\frac{1}{2} \frac{\mathrm{d}}{\mathrm{d}t} \|U\|^2_{N_p,\Gamma} = -b_L - b_R. \tag{87}$$

The boundary terms were computed exactly in the previous paragraph. So instead of estimating $-b_L - b_R$ by zero, we could also integrate both sides of (87) over the time interval $[0,t]$ to obtain

$$\|U(\cdot,t)\|^2_{N_p,\Gamma} = \|U_0\|^2_{N_p,\Gamma} - 2 \int_0^t \left(b_L(\tau) + b_R(\tau) \right) \mathrm{d}\tau.$$

Recall that with the above assumptions, the energy of the exact solution was shown to be constant in time. Therefore, with the above equation, we have a formula to compute the amount of numerical dissipation explicitly for this special case. Namely, this is the integral value of

$$2 \int_0^t \left(b_L(\tau) + b_R(\tau) \right) \mathrm{d}\tau.$$

This result will be validated in section 4.3.

3.6 Formulation as Ordinary Differential Equation in Time

Up to this point, we described the steps that are necessary to discretize the intrinsic beam equation in space, using a Discontinuous Galerkin approach in general, but did not go into detail regarding the process of numerically solving the resulting semi discrete problem. In this section, we will explicitly derive an ordinary differential equation for the unknowns of the numerical solution, namely the time dependent coefficients $a_j^{\mathrm{m},(\mathrm{k})}(t)$ from the representation

$$U^{(\mathrm{k})}(\xi,t) = \sum_{\mathrm{m}=1}^{12}\sum_{j=0}^{N_p} a_j^{\mathrm{m},(\mathrm{k})}(t)\mathcal{L}_j^{\mathrm{m}}(\xi)$$

of the numerical solution for every cell C_{k}. As long as it is clear, that we consider the numerical solution in the fixed cell C_{k}, we omit the superscript (k) again, writing for example $U = U^{(\mathrm{k})}$ to ease the notation.

Due to the special property $L_j(\xi_i) = \delta_{ij}$ of the Lagrange polynomials, we have that $\mathcal{L}_j^{\mathrm{m}}(\xi_i) = \delta_{ij}e_{\mathrm{m}}$ at an arbitrary interpolation node ξ_i. The evaluation of the solution at an interpolation node therefore becomes

$$U(\xi_i,t) = \sum_{\mathrm{m}=1}^{12} a_i^{\mathrm{m}}(t)e_{\mathrm{m}},$$

meaning that $a_i^{\mathrm{m}}(t) = U_{\mathrm{m}}(\xi_i,t)$. Hence, the unknown coefficients that have to be determined are the evaluations of the numerical solution at the interpolation nodes.

To have a notation that lets us distinguish between indices and partial derivatives easily, we switch from the notation U_t to $\dot{U}$ for the time derivative and from U_ξ to U' for the derivative with respect to ξ. The semi discrete weak formulation (66) from section 3.2 then reads

$$\frac{\Delta x_{\mathrm{k}}}{2}\left\langle \phi, \Gamma\dot{U}\right\rangle_{N_p} - \left\langle \Pi\phi', U\right\rangle_{N_p} - \frac{\Delta x_{\mathrm{k}}}{2}\left\langle \phi, Q_{cap}(U)\right\rangle_{N_p} = -\phi^T\mathcal{F}^*\Big|_{-1}^{1}, \tag{88}$$

where ϕ was an arbitrary function in the discrete space of test functions $\mathcal{V}_{N_p}$. Within the cell C_{k}, ϕ can therefore be interpreted as a polynomial of degree N_p. The equation is therefore fulfilled for every test function, if and only if it is fulfilled for every basis function of a basis for the polynomial space $\mathcal{P}_{N_p}(R)$. That basis can again be $\mathcal{L}$, meaning that the semi discrete weak formulation is fulfilled if and only if

$$\frac{\Delta x_{\mathrm{k}}}{2}\left\langle \mathcal{L}_i^{\mathrm{m}}, \Gamma\dot{U}\right\rangle_{N_p} - \left\langle \Pi\big(\mathcal{L}_i^{\mathrm{m}}\big)', U\right\rangle_{N_p} - \frac{\Delta x_{\mathrm{k}}}{2}\left\langle \mathcal{L}_i^{\mathrm{m}}, Q_{cap}(U)\right\rangle_{N_p} = -\left(\mathcal{L}_i^{\mathrm{m}}\right)^T\mathcal{F}^*\Big|_{-1}^{1}, \tag{89}$$

for every $i = 0,\dots,N_p$ and $\mathrm{m} = 1,\dots,12$. The goal is now, to reformulate (89) as a linear equation system that can be solved for $\dot{U}$, resulting in a system of ODEs for U in time.

We derive the linear equation system, treating the above semi discrete formulation term by term. First, we write down the quadrature of the first term explicitly:

$$\left\langle \mathcal{L}_i^{\mathrm{m}}, \Gamma\dot{U}\right\rangle_{N_p} = \sum_{j=0}^{N_p}\omega_j\left(\mathcal{L}_i^{\mathrm{m}}\right)^T\Gamma\dot{U}\Big|_{\xi_j}.$$

Note that $\mathcal{L}_i^{\mathrm{m}}$ only has one non-zero entry in its m-th component, which is why the summands of the above quadrature simplify to

$$\left\langle \mathcal{L}_i^{\mathrm{m}}, \Gamma\dot{U}\right\rangle_{N_p} = \sum_{j=0}^{N}\omega_j L_i(\Gamma\dot{U})_{\mathrm{m}}\Big|_{\xi_j}.$$

Again, taking advantage of the properties of the Lagrange polynomials, all summands of the sum are zero except the i-th one, resulting in

$$\left\langle \mathcal{L}_i^{\mathrm{m}}, \Gamma\dot{U}\right\rangle_{N_p} = \omega_i(\Gamma\dot{U})_{\mathrm{m}}\Big|_{\xi_i}.$$

The last result is a direct consequence of the co-location. Only because the quadrature nodes coincide with the interpolation nodes, the evaluation of the Lagrange polynomials takes place at

the interpolation nodes. If we proceed like this for every basis function $\mathcal{L}_i^{\mathrm{m}}$, we obtain a vector whose entries can be determined as follows

$$\left(\left\langle \mathcal{L}_i^{\mathrm{m}}, \Gamma\dot{U}\right\rangle_{N_p}\right)_{i=0,\dots,N_p}^{\mathrm{m}=1,\dots,12} =$$

$$\begin{pmatrix} \omega_0\mathbf{I}_{12,12} & & & \\ & \omega_1\mathbf{I}_{12,12} & & \\ & & \ddots & \\ & & & \omega_{N_p}\mathbf{I}_{12,12} \end{pmatrix} \begin{pmatrix} \Gamma(\xi_0) & & & \\ & \Gamma(\xi_1) & & \\ & & \ddots & \\ & & & \Gamma(\xi_{N_p}) \end{pmatrix} \begin{pmatrix} \dot{U}(\xi_0,t) \\ \dot{U}(\xi_1,t) \\ \vdots \\ \dot{U}(\xi_{N_p},t) \end{pmatrix}, \tag{90}$$

where the order of the entries is chosen such that for every i, we run through the values of the index m before we increment i by one, i.e.

$$\begin{aligned} &\left(\left\langle \mathcal{L}_i^{\mathrm{m}}, \Gamma\dot{U}\right\rangle_{N_p}\right)_{i=0,\dots,N_p}^{\mathrm{m}=1,\dots,12} = \\ &\quad\left(\left\langle \mathcal{L}_1^1, \Gamma\dot{U}\right\rangle_{N_p}, \left\langle \mathcal{L}_1^2, \Gamma\dot{U}\right\rangle_{N_p}, \dots, \left\langle \mathcal{L}_1^{12}, \Gamma\dot{U}\right\rangle_{N_p}, \left\langle \mathcal{L}_2^1, \Gamma\dot{U}\right\rangle_{N_p}, \dots \left\langle \mathcal{L}_{N_p}^{12}, \Gamma\dot{U}\right\rangle_{N_p}\right)^T. \end{aligned} \tag{91}$$

Let us denote the vector with the same entries as above but in an order where m and i are swapped as

$$\begin{aligned} &\left(\left\langle \mathcal{L}_i^{\mathrm{m}}, \Gamma\dot{U}\right\rangle_{N_p}\right)_{\mathrm{m}=1,\dots,12}^{i=0,\dots,N_p} = \\ &\quad\left(\left\langle \mathcal{L}_1^1, \Gamma\dot{U}\right\rangle_{N_p}, \left\langle \mathcal{L}_2^1, \Gamma\dot{U}\right\rangle_{N_p}, \dots, \left\langle \mathcal{L}_{N_p}^1, \Gamma\dot{U}\right\rangle_{N_p}, \left\langle \mathcal{L}_1^2, \Gamma\dot{U}\right\rangle_{N_p}, \dots \left\langle \mathcal{L}_{N_p}^{12}, \Gamma\dot{U}\right\rangle_{N_p}\right)^T. \end{aligned} \tag{92}$$

Although it is convenient to derive the matrix vector formulation for the first term, ordered as in (91), for the remaining terms it will be a lot easier to use an order like in (92). This is why we introduce the permutation matrix $\mathfrak{P} \in \mathbb{R}^{12(N_p+1)\times 12(N_p+1)}$ with the following entries:

$$\mathfrak{P}_{ij} = \begin{cases} 1, & \text{if } j = \lceil {}^{i}/_{(N_p+1)} \rceil + 12\left((i-1) \mod (N_p+1)\right) \\ 0, & \text{else.} \end{cases}$$

It rearranges the entries of a vector in the sense that a multiplication of (91) by $\mathfrak{P}$ results in (92), i.e.

$$\mathfrak{P}\left(\left\langle \mathcal{L}_i^{\mathrm{m}}, \Gamma\dot{U}\right\rangle_{N_p}\right)_{i=0,\dots,N_p}^{\mathrm{m}=1,\dots,12} = \left(\left\langle \mathcal{L}_i^{\mathrm{m}}, \Gamma\dot{U}\right\rangle_{N_p}\right)_{\mathrm{m}=1,\dots,12}^{i=0,\dots,N_p}.$$

Applying this to the derived matrix vector notation of the first term of the semi discrete formulation, (90), we obtain

$$\mathfrak{P}\left(\left\langle \mathcal{L}_i^{\mathrm{m}}, \Gamma\dot{U}\right\rangle_{N_p}\right)_{i=0,\dots,N_p}^{\mathrm{m}=1,\dots,12} =$$

$$\mathfrak{P}\begin{pmatrix} \omega_0\mathbf{I}_{12,12} & & & \\ & \omega_1\mathbf{I}_{12,12} & & \\ & & \ddots & \\ & & & \omega_{N_p}\mathbf{I}_{12,12} \end{pmatrix} \begin{pmatrix} \Gamma(\xi_0) & & & \\ & \Gamma(\xi_1) & & \\ & & \ddots & \\ & & & \Gamma(\xi_{N_p}) \end{pmatrix} \begin{pmatrix} \dot{U}(\xi_0,t) \\ \dot{U}(\xi_1,t) \\ \vdots \\ \dot{U}(\xi_{N_p},t) \end{pmatrix}.$$

A more compact notation of the above expression is

$$\left(\left\langle \mathcal{L}_i^{\mathrm{m}}, \Gamma\dot{U}\right\rangle_{N_p}\right)_{\mathrm{m}=1,\dots,12}^{i=0,\dots,N_p} = \boldsymbol{\mathcal{M}}\boldsymbol{\Gamma}\dot{\boldsymbol{U}}$$

where we define $\boldsymbol{\mathcal{M}}, \boldsymbol{\Gamma} \in \mathbb{R}^{12(N_p+1)\times 12(N_p+1)}$ and $\boldsymbol{U} \in \mathbb{R}^{12(N_p+1)}$ as

$$\boldsymbol{\mathcal{M}} := \mathfrak{P} \begin{pmatrix} \omega_0 \mathbf{I}_{12,12} & & & \\ & \omega_1 \mathbf{I}_{12,12} & & \\ & & \ddots & \\ & & & \omega_{N_p} \mathbf{I}_{12,12} \end{pmatrix} \mathfrak{P}^{-1} = \operatorname{diag}(\omega_0, \dots, \omega_{12}, \omega_0, \dots, \omega_{12}),$$

$$\boldsymbol{\Gamma} := \mathfrak{P} \begin{pmatrix} \Gamma(\xi_0) & & & \\ & \Gamma(\xi_1) & & \\ & & \ddots & \\ & & & \Gamma(\xi_{N_p}) \end{pmatrix} \mathfrak{P}^{-1},$$

$$\boldsymbol{U} = \boldsymbol{U}(t) = \mathfrak{P}\Big(U(\xi_0, t), U(\xi_1, t), \dots, U(\xi_{N_p}, t)\Big)^T = \Big(U_{\mathrm{m}}(\xi_i, t)\Big)_{\mathrm{m}=1,\dots,12}^{i=0,\dots,N_p}.$$

We proceed by deriving the matrix vector notation for the second summand of the semi discrete formulation, namely

$$\big\langle \Pi\big(\mathcal{L}_i^{\mathrm{m}}\big)', U \big\rangle_{N_p}.$$

Remember that Π is a symmetric and constant matrix. The quadrature of the above term can therefore be written as

$$\big\langle \Pi\big(\mathcal{L}_i^{\mathrm{m}}\big)', U \big\rangle_{N_p} = \sum_{j=0}^{N_p} \omega_j ((\mathcal{L}_i^{\mathrm{m}})')^T \Pi U \Big|_{\xi_j}.$$

Because $(\mathcal{L}_i^{\mathrm{m}})'$ has again only one entry in its m-th component, we can write

$$\big\langle \Pi\big(\mathcal{L}_i^{\mathrm{m}}\big)', U \big\rangle_{N_p} = \sum_{j=0}^{N_p} \omega_j L_i'(\Pi U)_{\mathrm{m}} \Big|_{\xi_j}.$$

Another representation of this expression in matrix vector notation is

$$\big\langle \Pi\big(\mathcal{L}_i^{\mathrm{m}}\big)', U \big\rangle_{N_p} = \Big(L_i'(\xi_0), \dots, L_i'(\xi_{N_p})\Big) \begin{pmatrix} \omega_0 & & \\ & \ddots & \\ & & \omega_{N_p} \end{pmatrix} \begin{pmatrix} (\Pi U)_{\mathrm{m}}(\xi_0, t) \\ \vdots \\ (\Pi U)_{\mathrm{m}}(\xi_{N_p}, t) \end{pmatrix}. \tag{93}$$

Now, fixing the index m and running through the i yields the vector

$$\Big(\big\langle \Pi\big(\mathcal{L}_i^{\mathrm{m}}\big)', U \big\rangle_{N_p}\Big)^{i=0,\dots,N_p} = \begin{pmatrix} L_0'(\xi_0) & \dots & L_0'(\xi_{N_p}) \\ \vdots & \ddots & \vdots \\ L_{N_p}'(\xi_0) & \dots & L_{N_p}'(\xi_{N_p}) \end{pmatrix} \begin{pmatrix} \omega_0 & & \\ & \ddots & \\ & & \omega_{N_p} \end{pmatrix} \begin{pmatrix} (\Pi U)_{\mathrm{m}}(\xi_0, t) \\ \vdots \\ (\Pi U)_{\mathrm{m}}(\xi_{N_p}, t) \end{pmatrix}. \tag{94}$$

Let us further define the discrete differentiation matrix $\mathfrak{D} \in \mathbb{R}^{(N_p+1)\times(N_p+1)}$ as

$$\mathfrak{D} := \begin{pmatrix} L_0'(\xi_0) & \dots & L_{N_p}'(\xi_0) \\ \vdots & \ddots & \vdots \\ L_0'(\xi_{N_p}) & \dots & L_{N_p}'(\xi_{N_p}) \end{pmatrix}$$

and the resulting block diagonal matrix $\boldsymbol{\mathfrak{D}} \in \mathbb{R}^{12(N_p+1)\times 12(N_p+1)}$ with $\mathfrak{D}$ on its diagonal, i.e.

$$\boldsymbol{\mathfrak{D}} := \begin{pmatrix} \mathfrak{D} & & \\ & \ddots & \\ & & \mathfrak{D} \end{pmatrix}.$$

Additionally, by $\boldsymbol{\mathcal{F}} \in \mathbb{R}^{12(N_p+1)}$, we denote the discrete flux vector

$$\boldsymbol{\mathcal{F}} := \boldsymbol{\mathcal{F}}(t) = \Big((\Pi U)_{\mathrm{m}}(\xi_i, t)\Big)_{\mathrm{m}=1,\dots,12}^{i=0,\dots,N_p}.$$

Setting up the vector (94) for every $\mathrm{m} = 1, \dots, 12$, then results in the following expression

$$\Big(\big\langle \Pi(\mathcal{L}_i^{\mathrm{m}})', U\big\rangle_{N_p}\Big)_{\mathrm{m}=1,\dots,12}^{i=0,\dots,N_p} = \boldsymbol{\mathfrak{D}}^T \boldsymbol{\mathcal{M}} \boldsymbol{\mathcal{F}}.$$

The next step is to bring the term, emerging from the source term, namely

$$-\frac{\Delta x_{\mathrm{k}}}{2} \big\langle \mathcal{L}_i^{\mathrm{m}}, Q_{cap}(U)\big\rangle_{N_p}$$

into a similar matrix vector form. In order to do so, we write down the corresponding quadrature formula explicitly again:

$$\big\langle \mathcal{L}_i^{\mathrm{m}}, Q_{cap}(U)\big\rangle_{N_p} = \sum_{j=0}^{N_p} \omega_j (\mathcal{L}_i^{\mathrm{m}})^T Q_{cap}(U)\Big|_{\xi_j}.$$

As before, we take advantage of the fact that $\mathcal{L}_i^{\mathrm{m}}$ has only one non-zero entry and then use the Kronecker delta property of the Lagrange polynomials to simplify the quadrature to

$$\big\langle \mathcal{L}_i^{\mathrm{m}}, Q_{cap}(U)\big\rangle_{N_p} = \omega_i (Q_{cap}(U))_{\mathrm{m}}\big|_{\xi_i}.$$

We run through the indices in the same way as we did before. This means for a fixed m we first run through all the i, resulting in

$$\Big(\big\langle \mathcal{L}_i^{\mathrm{m}}, Q_{cap}(U)\big\rangle_{N_p}\Big)^{i=0,\dots,N_p} = \begin{pmatrix} \omega_0 & & \\ & \ddots & \\ & & \omega_{N_p} \end{pmatrix} \begin{pmatrix} Q_{cap}(U)_{\mathrm{m}}\big|_{\xi_0} \\ \vdots \\ Q_{cap}(U)_{\mathrm{m}}\big|_{\xi_{N_p}} \end{pmatrix}.$$

Following this procedure for every m, we can then write

$$\Big(\big\langle \mathcal{L}_i^{\mathrm{m}}, Q_{cap}(U)\big\rangle_{N_p}\Big)_{\mathrm{m}=1,\dots,12}^{i=0,\dots,N_p} = \boldsymbol{\mathcal{M}} \boldsymbol{Q},$$

where we define the vector $\boldsymbol{Q} \in \mathbb{R}^{12(N_p+1)}$ similar to $\boldsymbol{\mathcal{F}}$ by

$$\boldsymbol{Q} := \boldsymbol{Q}(t) = \Big(Q_{cap}(U)_{\mathrm{m}}\big|_{\xi_i}\Big)_{\mathrm{m}=1,\dots,12}^{i=0,\dots,N_p}.$$

Note, that the dependency of $\boldsymbol{Q}$ on t emerges from the dependency of U on t. What is now left to complete the matrix vector formulation of the semi discrete equation, is the cell boundary term

$$-\left(\mathcal{L}_i^{\mathrm{m}}\right)^T \mathcal{F}^*\Big|_{-1}^{1} = -L_i \mathcal{F}_{\mathrm{m}}^*\Big|_{-1}^{1}.$$

Note that ± 1 are in fact the first, respective last LGL nodes, i.e. $\xi_0 = -1$ and $\xi_{N_p} = 1$. This is why the above term is only non-zero, if $i = 0$ or $i = N_p$. In other words, if we run through all i for a fixed m, we have

$$-\left(\left(\mathcal{L}_i^{\mathrm{m}}\right)^T \mathcal{F}^*\Big|_{-1}^{1}\right)^{i=0,\dots,N_p} = -\begin{pmatrix} -1 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 0 \\ 0 & \dots & 0 & 1 \end{pmatrix} \begin{pmatrix} \mathcal{F}_{\mathrm{m}}^*|_{-1} \\ 0 \\ \vdots \\ 0 \\ \mathcal{F}_{\mathrm{m}}^*|_{1} \end{pmatrix}.$$

If we additionally define the above matrix as $\beta \in \mathbb{R}^{(N_p+1)\times(N_p+1)}$, meaning

$$\beta := \begin{pmatrix} -1 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 0 \\ 0 & \dots & 0 & 1 \end{pmatrix}$$

and introduce the block diagonal matrix $\boldsymbol{\beta} \in \mathbb{R}^{12(N_p+1)\times 12(N_p+1)}$ and the vector $\boldsymbol{\mathcal{F}}^* \in \mathbb{R}^{12(N_p+1)}$ as

$$\boldsymbol{\beta} := \begin{pmatrix} \beta & & \\ & \ddots & \\ & & \beta \end{pmatrix},$$

$$\boldsymbol{\mathcal{F}}^* := \left(\mathcal{F}_1^*\big|_{-1}, 0\ldots, 0, \mathcal{F}_1^*\big|_1, \mathcal{F}_2^*\big|_{-1}, \ldots, \mathcal{F}_{12}^*\big|_1\right)^T,$$

we have that

$$-\left(\left(\mathcal{L}_i^{\mathrm{m}}\right)^T \mathcal{F}^*\Big|_{-1}^{1}\right)_{\mathrm{m}=1,\ldots,12}^{i=0,\ldots,N_p} = -\boldsymbol{\beta}\boldsymbol{\mathcal{F}}^*.$$

Now putting the matrix vector formulation of all the terms together again, the semi discrete equation reads

$$\frac{\Delta x_{\mathrm{k}}}{2}\boldsymbol{\mathcal{M}}\boldsymbol{\Gamma}\dot{\boldsymbol{U}} = \boldsymbol{\mathfrak{D}}^T\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{F}} + \frac{\Delta x_{\mathrm{k}}}{2}\boldsymbol{\mathcal{M}}\boldsymbol{Q} - \boldsymbol{\beta}\boldsymbol{\mathcal{F}}^*.$$

This expression can be solved for the vector $\dot{\boldsymbol{U}}$. More precisely, we have

$$\dot{\boldsymbol{U}} = \frac{2}{\Delta x_{\mathrm{k}}}\boldsymbol{\Gamma}^{-1}\boldsymbol{\mathcal{M}}^{-1}\left(\boldsymbol{\mathfrak{D}}^T\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{F}} + \frac{\Delta x_{\mathrm{k}}}{2}\boldsymbol{\mathcal{M}}\boldsymbol{Q} - \boldsymbol{\beta}\boldsymbol{\mathcal{F}}^*\right). \tag{95}$$

The above equation is a system of ODEs in time, whose solution consists of the unknown coefficients that define the polynomial representing the numerical solution of the intrinsic beam equation. Remember that we omitted the superscript (k) when we derived this equation, meaning that the solution vector $\boldsymbol{U}$ actually only represents the coefficients of the local solution in the cell C_{k} and has therefore to be denoted by $\boldsymbol{U}^{(\mathrm{k})}$, to be precise. This holds true for all the matrices and vectors in (95) and is also indicated by the factor $\Delta x_{\mathrm{k}}/2$. Thus, in order to complete the global ODE formulation, one has to put together the equations (95) for $\mathrm{k} = 1, \ldots, N_c$ into one global system, with solution vector

$$\mathfrak{U} := \begin{pmatrix} \boldsymbol{U}^{(1)} \\ \vdots \\ \boldsymbol{U}^{(N_c)} \end{pmatrix}.$$

The derivation of the global system on the basis of equation (95) is straight forward, which is why we will not explicitly show it here and instead just denote the right hand side by $\mathfrak{R}$. The global system of ODEs that arises from that, thus, reads

$$\dot{\mathfrak{U}} = \mathfrak{R}(\mathfrak{U}, t). \tag{96}$$

The number of unknowns and thus the degrees of freedom (DOF) equals the number of entries in $\mathfrak{U}$, which is $12(N_p + 1)N_c$.

This ODE system can now be solved numerically by applying a numerical time integration scheme in order to obtain the coefficients in $\mathfrak{U}$ at certain points in time. We continue with some remarks about the discretization in time in the next section.

3.7 Remarks on Time Discretization

In the previous section, we showed that the discretization approach in space eventually leads us to a system of ordinary differential equations for the unknown coefficients of the numerical solution that still continuously depend on the time variable t. In order to find a numerical solution to the problem, this ODE has to be solved. Although we will not go into detail regarding the solution of the ODE in this thesis, we would like to mention the most important ideas behind it and give an example of a scheme that is suitable to numerically solve the ODE.

A class of numerical schemes that are commonly used to obtain a solution of the ODE emerging from the DG discretization of time dependent PDEs are *Runge-Kutta methods* (cf. for example [14,

18, 27]). A special Runge-Kutta method that will also be used to obtain numerical results in section 4 is the *low-storage five-stage fourth-order explicit Runge-Kutta method (LSERK)*, originating from [10] and recommended for the time integration following a spatial DG discretization for example in [18].

To apply the LSERK to our problem, first the time interval, $[0, T]$, is discretized by subdividing it into $N_t + 1$ points $0 =: t_0 < t_1 < \ldots < t_{N_t} := T$ and defining timesteps Δt_n for $n = 1, \ldots, N_t$ by

$$\Delta t_n := t_n - t_{n-1} > 0.$$

For the LSERK, we assume the timestep to be constant, i.e. $\Delta t := \Delta t_n$ for all n, meaning that

$$t_n = n\Delta t$$

for $n = 0, \ldots, N_t$. Some more detailed remarks on the timestep and especially how it has to be chosen in order to obtain a stable numerical approximation scheme will be made later in this section.

Let us now denote the numerical solution of (96) at the n-th timestep as $\mathfrak{U}^n$, i.e.

$$\mathfrak{U}^n \approx \mathfrak{U}(t_n).$$

Then the five stages of one step of the LSERK are given in algorithm 1, for which the coefficients a_i, b_i and c_i for $i = 1, \ldots, 5$ can be read up in table 1.

Algorithm 1 LSERK [18].

```
Set p^(0) = U^n

for i = 1 to 5 do
    k^(i) = a_i k^(i-1) + Δt R(p^(i-1), t_n + c_i Δt)
    p^(i) = p^(i-1) + b_i k^(i)
end for

U^(n+1) = p^(5)
```

The stability considerations we did up to this point have been made under the assumption that time integration is exact. Now, that the solution is no longer semi discrete but fully discrete, the numerical time integration can also affect the stability. This has to be minded when choosing the timestep Δt. More precisely, there is a sufficient condition for the timestep in order to have a stable time integration scheme. It relates the size of the timestep to the size of the grid size from the spatial discretization and is called the CFL-condition going back to [12] and its authors Courant, Friedrichs and Lewy. Adapted to the notations in this thesis, the CFL condition for a stable timepstep reads

$$\Delta t \leq \mathrm{CFL}\, \frac{1}{\max |\boldsymbol{\Lambda}_{ii}|} \min \Delta x. \tag{97}$$

In the simplified notation above, $\max |\boldsymbol{\Lambda}_{ii}|$ represents the maximal diagonal entry of the matrix $|\boldsymbol{\Lambda}|$, which is in fact the maximum of the absolute value of the matrix's A eigenvalues. Note that A and therefore also its maximal absolute eigenvalue depends on the spatial variable x. Furthermore, $\min \Delta x$ represents the minimum of all distances between spatial nodes. To determine this minimal distance, one has to mind that the cells C_k may be of different width (Δx_k), and that the LGL nodes are not equidistant. Finally, the CFL-number can be derived by determining the stability region of the chosen time integration scheme. A more detailed view on the stability of the time discretization and the CFL-number within the context of DG discretizations is for example given in [18].

For our purposes, it suffices to assume that $\mathrm{CFL} \leq 1$. Later, in section 4, when we actually determine a numerical solution, the CFL-number will be a tool to control the timestep. The smaller we choose CFL, the smaller the timestep and the more accurate the time integration will be, provided that the time integration scheme is convergent. This is espacially useful if we want to analyze the numerical solution with regard to the spatial discretization approach, because we can minimize the effects of the time discretization by choosing $\mathrm{CFL} \ll 1$.

i	a_i	b_i	c_i
1	0	$\frac{1432997174477}{9575080441755}$	0
2	$-\frac{567301805773}{1357537059087}$	$\frac{5161836677717}{13612068292357}$	$\frac{1432997174477}{9575080441755}$
3	$-\frac{2404267990393}{2016746695238}$	$\frac{1720146321549}{2090206949498}$	$-\frac{2526269341429}{6820363962896}$
4	$-\frac{3550918686646}{2091501179385}$	$\frac{3134564353537}{4481467310338}$	$\frac{2006345519317}{3224310063776}$
5	$-\frac{1275806237668}{842570457699}$	$\frac{2277821191437}{14882151754819}$	$\frac{2802321613138}{2924317926251}$

Table 1: LSERK coefficitents.

4 Numerical Results

In the following section we will numerically validate the theoretical results from the previous sections. To this end, we will use the numerical simulation framework `Trixi.jl`. We will start with a summary of the most important steps regarding an implementation of the intrinsic beam model into `Trixi.jl`. Subsequently, we will validate the correctness of our implementations by performing a convergence test using the method of manufactured solutions. Afterwards, we will pick an example configuration of the intrinsic beam model with zero external sources and moments, to show that the discrete energy that results from the corresponding simulation with `Trixi.jl` is non increasing, indeed. Finally, we will show how the solution resulting from a `Trixi.jl` simulation can be post processed to obtain and visualize the position line of the dynamic beam.

Throughout this section, we make some more assumptions in addition to the assumptions of the previous sections. Firstly, our implementations at the moment consider constant flexibility and mass matrices only, which is why we assume

$$\mathbf{F}, \mathbf{M} = \text{const.}$$

in the following investigations. As a consequence, for the matrices A, Γ and Ψ it also holds

$$A, \Gamma, \Psi = \text{const.}$$

Furthermore, as time integration scheme, we will use the LSERK, defined in section 3.7, throughout the whole section.

4.1 Implementation with `Trixi.jl`

In this section, we will describe the most important implementations that are needed in order to be able to use `Trixi.jl` for simulations of the intrinsic beam model. `Trixi.jl` is a numerical simulation framework for hyperbolic conservation laws. Detailed information about `Trixi.jl` can be read up in [40, 46, 47] and in the documentation [51]. Moreover, information about the underlying programming language, Julia, is to find in [22]. In this thesis, we will use `Trixi.jl` to numerically solve the intrinsic beam equation using a Discontinuous Galerkin approach as it was derived in the previous sections.

First of all, we need `Trixi.jl` to get to know the intrinsic beam equation. Different types of equations in `Trixi.jl` are represented by different composite data types, each of them being a subtype of an abstract data type `AbstractEquations{NDIMS, NVARS}`, where `NDIMS` specifies the number of spatial dimensions and `NVARS` specifies the number of primary variables. In the case of the intrinsic beam model, we have a spatial dimension of one, meaning `NDIMS` $= 1$ and the state variable u consists of twelve unknowns, i.e. `NVARS` $= 12$.

The data types representing the equations can contain optional fields, helping us to specify the parameters in specific setups. For the intrinsic beam equation, we create a new data type

```
IntrinsicBeamEquation <:  AbstractEquations{1, 12}
```

containing the following fields: the matrix A, the matrices A_+ and A_-, the matrix Ψ, the matrix Π, the matrix Γ and its inverse Γ^{-1}, the flexibility matrix $\mathbf{F}$ and the mass matrix $\mathbf{M}$, the initial curvature k as a function of x and the external forces f_{ext} and moments m_{ext} as functions of x and t. In the above definition, `<:` is the Julia syntax for `IntrinsicBeamEquation` being a subtype of `AbstractEquations{1, 12}`.

Further, we need a constructor function, also named `IntrinsicBeamEquation`, which creates an object of the type `IntrinsicBeamEquation` for given flexibility and mass matrices and given initial curvatures and external forces and moments. Note that every other field of `IntrinsicBeam Equation` can be determined out of these.

To implement the new data type for the intrinsic beam equation is the most important part of the implementations. For the rest, we can mostly use the existing `Trixi.jl` functions or extend them to solve specific setups. This is because `Trixi.jl` takes advantage of the multiple dispatch functionality in Julia. Generally spoken, this means that when a function is applied in Julia, the dispatch process chooses, which method has to be called based on the number of arguments given to that function and on all their types.

In the following, we will explain the implementation of a function that evaluates the flux function of the intrinsic beam equation in capacity form. In the course of this, the usage of

multiple dispatch will be explained, too. All conservation laws, respectively balance laws, have in common that they have a flux function. Whenever the flux needs to be computed in `Trixi.jl`, the function `flux(u, equations)` is applied. The argument `equations` is an object representing the conservation/balance law and `u` is a vector at which the flux function should be evaluated, e.g. the solution at a specific node.

Suppose now that we define a method, `flux(u, equations::IntrinsicBeamEquation)`, where the symbol `::` is the Julia syntax to specify the type of an argument. Then, whenever the `flux` function is called in `Trixi.jl`, the multiple dispatch process chooses just this method provided that the input argument `equations` is of the type `IntrinsicBeamEquation`. Explicitly, the flux function can be implemented as follows:

```
flux(u,equations::IntrinsicBeamEquation) = equations.Pi * u,
```

where the field `equations.Pi` contains the matrix Π.

Now that we have implemented a function computing the flux of the intrinsic beam equation, there are three remaining essentials, that need to be implemented: the numerical flux, derived in section 3.3, the source term as it was specified in section 2.4.4 and the discrete boundary conditions from section 3.4.

For the numerical flux, we write a function `flux_upwind`. From the sections before, we know that the numerical flux is always evaluated at interfaces (or at the boundaries). The arguments of the `flux_upwind` function therefore are a value of the solution left and right to the interface, `u_L` and `u_R`. Moreover, an input argument `equations` that is of the type `IntrinsicBeamEquation` is needed again. The latter argument contains fields, in which the matrices Γ, A_+ and A_- are stored. These can be used to straight forward implement the formula of the upwind numerical flux and return its evaluation at `u_L` and `u_R`.

The source term is implemented as a function `source_term_intrinsic_beam` with input arguments `u, x, t, equations`. The implementation is straight forward, inserting the state vector, `u`, and the point in space, `x`, as well as the point in time, `t`, into the source term Q_{cap}, as it is specified in section 2.4.4. The needed matrices and external forces and moments can be retrieved from the corresponding fields in `equations`. The product of cross product matrices and vectors that appear in the source term's definition are interpreted as cross products of vectors and computed by using the `cross` function of the Julia package `LinearAlgebra.jl`. The variable that is returned is a vector containing the value of the source term for the given input variables.

Another function, named `boundary_conditions_intrinsic_beam`, is needed to implement the boundary conditions. Its input arguments are `u_inner, direction, equations`. This function sets up the outer values at the respective boundary as we specified them in section 3.4, using the inner state of the current solution `u_inner`. The information about the location, i.e. which boundary has to be considered, is received in the argument `direction`. A value of 1 means that the left boundary is considered while a value of 2 indicates that the right boundary is considered. Afterwards, the numerical flux at the respective boundary is computed by inserting the outer value and the inner state of the solution into the formula of the upwind flux. Again, the quantities that are needed to set up the outer states and to evaluate the numerical flux are stored in the fields of the input argument `equations`. The output value is the evaluation of the boundary flux at the respective boundary.

Before we can actually solve a configuration of the intrinsic beam equation with `Trixi.jl`, we need to consider the following: The above functions assume that we discretize the capacity form of the linear advection equation in space. However, `Trixi.jl` expects that the discretization is for an equation without a factor in front of the time derivative, i.e. for an equation of the form

$$u_t + Au_x = Q(u).$$

This means that we have to slightly adjust the source code. The adjustment takes place in the function `rhs!` that sets up the right hand side of the ODE in time. For the Runge-Kutta discretization in time, the right hand side has to be evaluated at discrete points in time (cf. for example algorithm 1). The function `rhs!` performs all discretization steps we described when we derived the ODE in time, including for example the computation of the source term at the nodes and the numerical flux at the interfaces etc. This is done step by step, eventually leading to the evaluation of the right hand side of the ODE which is needed for the Runge-Kutta stages. For our purpose there is an additional step when setting up the right hand side. Namely, this is the

multiplication with the matrix $\mathbf{\Gamma}^{-1}$. This means we need to extend the function `rhs!` by adding a step at the end that multiplies the whole right hand side that is set up to that point by the matrix $\mathbf{\Gamma}^{-1}$. To do so, we define a new function `apply_gamma_inverse` and call it at the end of the `rhs!` function. Its input arguments include an object `equations` again. This means, that we can once more take advantage of the multiple dispatch, defining `apply_gamma_inverse` for `equation` of type `IntrinsicBeamEquation` and for `equation` of type `AbstractEquations`. The former does multiply the right hand side by the matrix $\mathbf{\Gamma}^{-1}$, while the latter does nothing, meaning that for any other equation type than `IntrinsicBeamEquation`, the right hand side is not affected. This ensures that our adjustment does not have an influence on the functionality of `Trixi.jl` for other equation types.

Now that we have implemented the essentials of the intrinsic beam equation, we can create different configurations of the intrinsic beam model and use `Trixi.jl` to solve them numerically. We follow the convention of the existing examples in `Trixi.jl` and name example scripts, that solve specific setups, *elixirs*. In the following, we want to give an overview of how an elixir for the intrinsic beam equation looks like in general. The `Trixi.jl` components that are mentioned in these explanations and their interaction are also shown schematically in figure 4.

First, we need to specify everything that is needed for the discretization of the considered spatial domain. This includes the interval boundaries 0 and ℓ and the number of cells in which the domain should be subdivided. With these information, an object `mesh` can be created that stores information about the discretization of the domain. In `Trixi.jl`, we have the possibility to create different types of meshes. For our purposes, we will only need the `TreeMesh` that simply subdivides the interval $[0, \ell]$ into a fixed number of cells of the same width. An implication of this is that the value of $\Delta x_\mathrm{k}/2$ that emerged from the transformation to the reference element, is the same for every cell C_k, i.e.

$$\frac{\Delta x_k}{2} = \text{const.}$$

Furthermore, we create an object called `solver`, using the `Trixi.jl` constructor function `DGSEM`, in which the spatial discretization approach, namely the DG approach, is specified. As input arguments, `DGSEM` requires the numerical flux function and the polynomial degree that should be used for the semi discretization. The numerical flux will either be `flux_upwind`, whose implementation is described above, or `flux_central`, which is already implemented in `Trixi.jl` and can also be used for our purposes. The resulting object `solver` stores for example the polynomial basis and the quadrature nodes and weights.

Afterwards, we create an object `equations` that is of the type `IntrinsicBeamEquation` using the constructor function of the same name. To do so, we first need to define a flexibility matrix, a mass matrix and functions for the initial curvature as well as for the external forces and moments. The object `equation` can then simply be created by calling the constructor function with these matrices and functions as input arguments.

Given the above objects, we can now use the `Trixi.jl` constructor `Semidiscretization Hyperbolic` to create a semi discretization according to the derivations in the previous sections. As input arguments, `SemidiscretizationHyperbolic` receives `mesh`, `equations`, `solver` as well as an initial condition (IC), the boundary conditions (BCs) and the source term. The implementation of the latter two is described earlier in this section and the initial condition is a simple function with input arguments `x` and `t` that evaluates the desired initial condition at the point `x` in space and the point `t` in time and returns the corresponding value. Note that actually, an initial condition does not depend on the time. Nevertheless, if an exact solution of the considered problem is known, it can be handed over as initial condition depending on the time. This can later for example be used to analyze the error of the numerical solution. If no exact solution is known, the input argument `t` in the initial condition simply remains unused.

Now, a time interval on which the numerical solution should be computed has to be specified. The time interval together with the semi discretization is given to the function `ode`, that creates an object of the type `ODEProblem`, containing the ordinary differential equation in time that arises from the above semi discretization.

Finally, the object of type `ODEProblem`, together with a specification of a time discretization scheme included in the package `OrdinaryDiffEq.jl` is handed over to the `solve` function, that computes the solution of the ODE in time using the specified time discretization scheme and, thus, returns the numerical solution of the intrinsic beam equation with the desired setup.

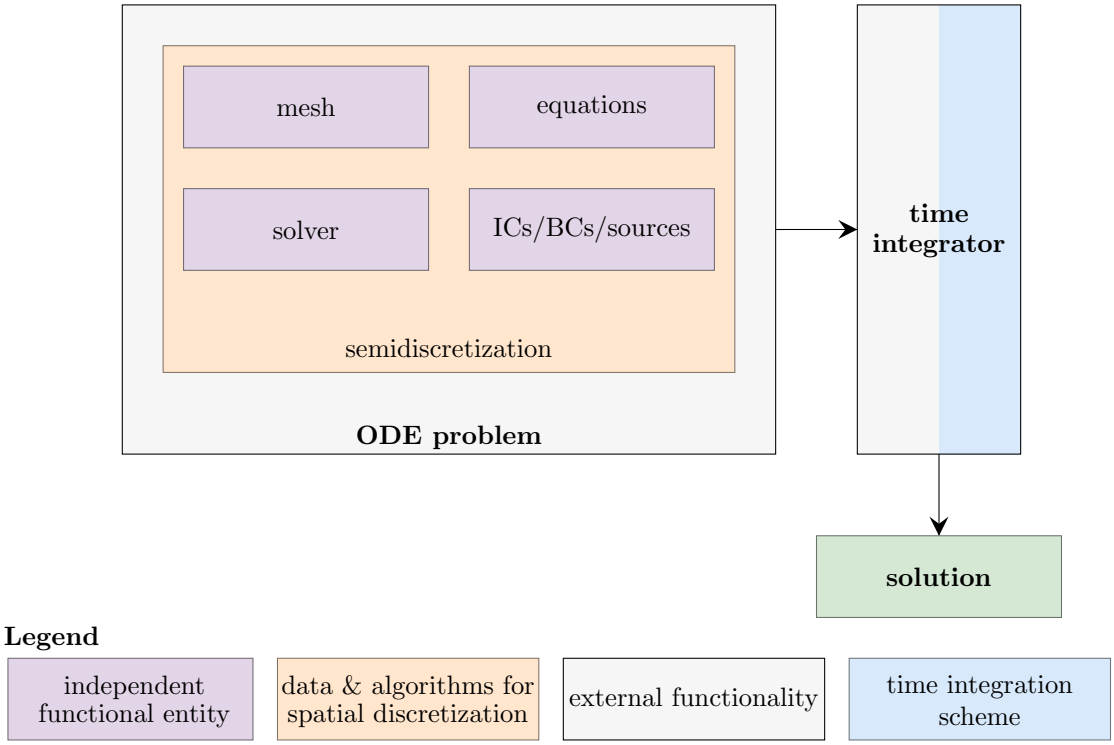


Figure 4: Schematic overview of the basic components of `Trixi.jl` and how they interact. This figure is a cutout of Fig. 4 in [40].

4.2 Convergence Tests

The purpose of this section is to check the correctness of the implementations in `Trixi.jl` we described in the previous section. The optimal convergence rate, i.e. the optimal rate at which the error of the numerical solution converges to zero as the spatial grid size converges to zero, is the order of the piecewise polynomials, $N_p + 1$. We will numerically solve a test problem using the method of *manufactured solutions* to determine the *experimental order of convergence (EOC)* and verify whether the optimal convergence rate is reached.

For a detailed derivation of the method of manufactured solutions and how it is used to verify code in general, we refer to [41]. The idea behind the method is the following: Usually, for a convergence analysis one would determine the numerical solution of a problem for different refinement levels in the discretization and compare this numerical solution to the exact solution of the problem to determine for example an $\mathbb{L}^2$-error. Afterwards the rate, at which the error is getting smaller when the refinement level is raised, can be computed. This results in an empirical order of convergence that can be compared to the theoretical order of convergence of the considered discretization approach. Regarding the problem that is considered in this thesis, the above procedure can not be implemented without further ado because, in general, an analytical solution of the intrinsic beam equation is not known. Instead, defining a function u and inserting it into the left hand side of the linear balance law will result in a residuum like term, $\mathfrak{r} = \mathfrak{r}(x,t)$. In particular, this is

$$\Gamma u_t + \Pi u_x = \mathfrak{r}(x,t). \tag{98}$$

Now remember that in the original problem, the right hand side of the balance law consists of the source term $Q_{cap}(u)$. To bring this into account, we add a zero on the right hand side of the above definition of the residuum, to obtain

$$\Gamma u_t + \Pi u_x = Q_{cap}(u) + \mathfrak{r}(x,t) - Q_{cap}(u). \tag{99}$$

That is, the function u satisfies the original balance law with an additional source term on its right hand side. Namely, the additional source term is

$$\mathfrak{r}(x,t) - Q_{cap}(u). \tag{100}$$

While analytically there is no difference in solving equation (98) or equation (99), for the `Trixi.jl` simulation it does make a difference. To solve (99), we can use the implementations described in the previous section and only need to implement the additional source term. If the additional source term is implemented in such a way, that it evaluates the term $Q_{cap}(u)$ for the exact function u, the two Q_{cap} terms on the right hand side will not cancel each other out exactly, but for a converging numerical solution, the simulated source term should also converge to the exact one. Thus, performing the convergence test for the version (99) of the equation can also be seen as a verification of the source term's correctness in the original implementation.

As we know from the previous section, in `Trixi.jl` a source term can be implemented easily. Therefore, a function `st_manufactured_solution`, that evaluates the additional source term (100) for specific values of x and t needs to be implemented. More precisely, the input arguments of the new function are `x, t, equations, solution, solution_dot, solution_prime`, where the latter three ones represent the function u and its derivatives with respect to t and x in that order. The input variables `x` and `t` represent the values of x and t, at which the additional source term should be evaluated and `equations` is again an object of the type `IntrinsicBeamEquation`. The residuum can then be evaluated at every x and t. Moreover, the evaluation of Q_{cap} at the exact u can be implemented straight forward, using the definition of Q_{cap}. This is similar to the implementation of the original source term `source_term_intrinsic_beam` in the previous section. The only difference is that now the evaluation of the exact u is determined within the function and inserted into the source term instead of the current state of the solution, handed over by `Trixi.jl` as in `source_term_intrinsic_beam`.

When setting up a `Trixi.jl` simulation, the source term that is handed over to the constructor of the semi discretization has then to be defined as the sum of the above introduced function and the function that evaluates the original source term. The result of the `Trixi.jl` simulation should then be the numerical solution of equation (99), of which we know the exact solution. In [41], Roache summarizes the idea of defining the desired solution first and then solving the resulting problem by citing the counsel "Only a fool starts at the beginning; the wise one starts at the end" which is originally formulated by Polya in [39].

Let us now implement this for a specific configuration of the intrinsic beam model. First we choose the flexibility and mass matrix as

$$\mathbf{F} = \begin{pmatrix} 13 & 6 & 5 & 7 & 8 & 5 \\ 6 & 6 & 4 & 3 & 5 & 4 \\ 5 & 4 & 5 & 3 & 5 & 4 \\ 7 & 3 & 3 & 7 & 4 & 3 \\ 8 & 5 & 5 & 4 & 9 & 5 \\ 5 & 4 & 4 & 3 & 5 & 5 \end{pmatrix}, \qquad \mathbf{M} = \begin{pmatrix} 12 & 4 & 5 & 5 & 10 & 3 \\ 4 & 4 & 2 & 3 & 4 & 1 \\ 5 & 2 & 8 & 6 & 6 & 2 \\ 5 & 3 & 6 & 8 & 7 & 3 \\ 10 & 4 & 6 & 7 & 13 & 5 \\ 3 & 1 & 2 & 3 & 5 & 4 \end{pmatrix}. \tag{101}$$

They are chosen randomly and suffice the positive definiteness requirement for the mass and the flexibility matrix but do not take realistic values that describe the material properties of any beam. For the simulations in this section, realistic values are not needed, because the results, we are aiming for are purely mathematical. The initial curvature k as well as the external forces f_{ext} and moments m_{ext} are set to zero, i.e.

$$k(x) \equiv f_{ext}(x,t) \equiv m_{ext}(x,t) \equiv \mathbf{0}_3.$$

Moreover, we define the function $u(x,t) = (u_{\mathrm{m}}(x,t))_{\mathrm{m}=1,\dots,12}$ as

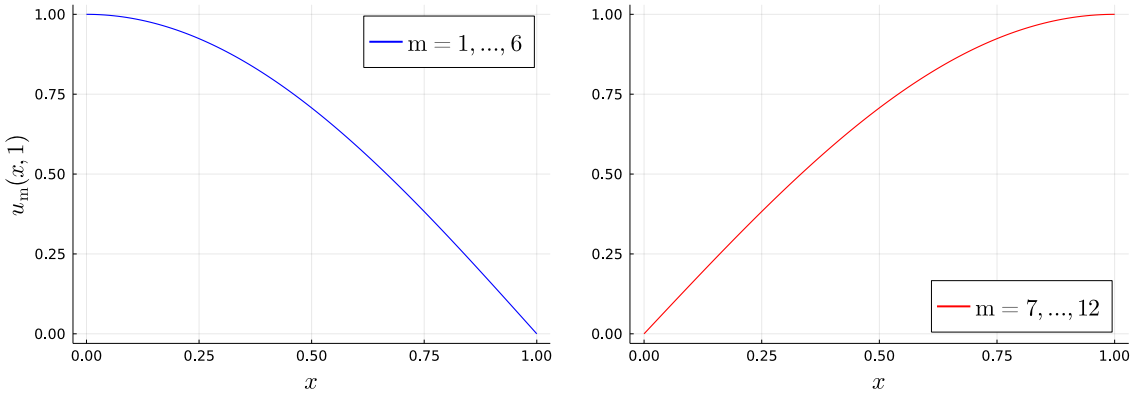
$$u_{\mathrm{m}}(x,t) = \begin{cases} t\sin\left(\frac{\pi}{2}(1-x)\right) & \text{for } \mathrm{m} = 1,\dots,6, \\ t\sin\left(\frac{\pi}{2}x\right) & \text{for } \mathrm{m} = 7,\dots,12. \end{cases} \tag{102}$$

The derivative of u with respect to x is then

$$(u_x)_{\mathrm{m}}(x,t) = \begin{cases} -\frac{\pi}{2}t\cos\left(\frac{\pi}{2}(1-x)\right) & \text{for } \mathrm{m} = 1,\dots,6, \\ \frac{\pi}{2}t\cos\left(\frac{\pi}{2}x\right) & \text{for } \mathrm{m} = 7,\dots,12, \end{cases}$$

while the derivative of u with respect to t can be calculated as

$$(u_t)_{\mathrm{m}}(x,t) = \begin{cases} \sin\left(\frac{\pi}{2}(1-x)\right) & \text{for } \mathrm{m} = 1,\dots,6, \\ \sin\left(\frac{\pi}{2}x\right) & \text{for } \mathrm{m} = 7,\dots,12. \end{cases}$$



(a) First six components of the solution.

(b) Last six components of the solution.

Figure 5: Plot of the exact solution of the problem (103).

This function can then be used to determine the residuum function

$$\mathfrak{r}(x,t) = \Gamma u_t + \Pi u_x$$

and the resulting additional source term (100). The problem we will consider for the rest of this section can now be specified as

$$\begin{cases} \Gamma u_t + \Pi u_x = Q_{cap}(u) + \mathfrak{r}(x,t) - Q_{cap}(u) & \text{for } (x,t) \in (0,1) \times (0,1], \\ u_{\mathrm{m}}(0,t) = 0 & \text{for } t \in [0,1],\ \mathrm{m} = 1\ldots,6, \\ u_{\mathrm{m}}(\ell,t) = 0 & \text{for } t \in [0,1],\ \mathrm{m} = 7,\ldots,12, \\ u_{\mathrm{m}}(x,0) = 0 & \text{for } x \in [0,\ell],\ \mathrm{m} = 1\ldots,12. \end{cases} \tag{103}$$

Note that the function u indeed fulfills the above initial and boundary conditions. This means that by construction the exact solution of the problem (103) is the function u, defined in (102). In figure 5, we can see a plot of the solution u at the time $t = T = 1$.

Having implemented the additional source term and what is needed for that implementation, the convergence analysis can now be performed easily taking advantage of the `Trixi.jl` function `convergence_test`. As input argument, this function expects a file path and an integer. The path is the location of an elixir that returns the solution of a `Trixi.jl` simulation. The `convergence_test` function then compares the resulting solution of the elixir to the initial condition that is defined within the elixir. In particular, it computes the $\mathbb{L}^2$-error. Note that for this to work, the initial condition has to be implemented as a function depending on t, i.e. as the exact solution. For the simulation itself `Trixi.jl` then uses this function evaluated at $t = 0$ as initial condition. For the convergence test, the solution is compared to this initial condition evaluated at the end time, i.e. at $t = 1$. This comparison is performed as many times as it is specified in the second argument of the `convergence_test` function, while in every iteration, the number of cells is increased by the factor of two. To minimize the impact of the time integrator, we choose a small CFL number of 0.1 in the following considerations.

For the resulting errors on different refinement levels of the spatial mesh, the EOC can now be determined. Let $\mathcal{E}_{c_1}$ be the $\mathbb{L}^2$-error of the numerical solution associated with a number of cells of N_{c_1} and let $\mathcal{E}_{c_2}$ be defined analogously with $N_{c_1} < N_{c_2}$. Then the EOC concerning the refinement levels c_1 and c_2 is defined as

$$\mathrm{EOC} = \frac{\log\left(\dfrac{\mathcal{E}_{c_1}}{\mathcal{E}_{c_2}}\right)}{\log\left(\dfrac{N_{c_2}}{N_{c_1}}\right)}.$$

As the `convergence_test` function doubles the refinement level in every iteration, we have that $N_{c_2} = 2N_{c_1}$ and, thus

$$\mathrm{EOC} = \frac{\log\left(\dfrac{\mathcal{E}_{c_1}}{\mathcal{E}_{c_2}}\right)}{\log(2)}.$$

In table 2, the resulting $\mathbb{L}_2$-error, and the corresponding EOC for a polynomial degree of $N_p = 3$ are listed. More precisely, the average $\mathbb{L}^2$-error of the first six components of the solution and the average for the last six components is specified for simulations with the upwind flux and with the central flux. In Appendix A.3, the individual errors and EOCs of each component can be looked up. The values do only differ on a small scale, which is why we only list the two averages in this section.

The optimal convergence rate for the spatial discretization approach is $N_p+1 = 4$. The results in table 2 show that this rate is approximately reached by the experiments using the upwind numerical flux. We observe that the average EOC in the first six as well as in the last six components is nearly 4. For the first two iterations ($N_c = 8 \to 16$ and $N_c = 16 \to 32$), the EOC is a bit lower than for the last two iterations ($N_c = 32 \to 64$ and $N_c = 64 \to 128$). Nevertheless, the EOC is constantly clear above a level of 3.5. At maximum it reaches a value of approximately 3.97 in the first six components and 3.96 in the last six components.

For the central flux, the optimal convergence rate is not reached. Overall, the EOC for these simulations is approximately 3, i.e. corresponds to the polynomial degree, N_p and not to $N_p + 1$. This is a well known phenomenon [18]. Numerical solutions that result from simulations, using the central numerical flux, usually converge at a rate of N_p for odd N_p and at a rate of $N_p + 1$ for even N_p. This will also be validated by our experiments for $N_p = 4$, later in this section.

The $\mathbb{L}^2$-error takes values from a magnitude of 10^{-7} to 10^{-11} for the upwind flux and of 10^{-7} to 10^{-10} for the central flux. Although the level of the error for the central flux is above the level of the error for the upwind flux all the time, in both cases, the exact solution is approximated well by the numerical solution even for the smallest refinement level of $N_c = 8$.

N_c	Num. flux	Avg. $\mathbb{L}^2$-error (m = 1, ..., 6)	Avg. EOC (m = 1, ..., 6)	Avg. $\mathbb{L}^2$-error (m = 7, ..., 12)	Avg. EOC (m = 7, ..., 12)
8	upwind	$5.31 \cdot 10^{-7}$	-	$4.80 \cdot 10^{-7}$	-
	central	$9.20 \cdot 10^{-7}$	-	$7.00 \cdot 10^{-7}$	-
16	upwind	$3.70 \cdot 10^{-8}$	3.84	$3.94 \cdot 10^{-8}$	3.65
	central	$9.92 \cdot 10^{-8}$	3.32	$7.66 \cdot 10^{-8}$	3.27
32	upwind	$2.58 \cdot 10^{-9}$	3.83	$2.68 \cdot 10^{-9}$	3.86
	central	$1.19 \cdot 10^{-8}$	3.12	$9.13 \cdot 10^{-9}$	3.11
64	upwind	$1.64 \cdot 10^{-10}$	3.97	$1.73 \cdot 10^{-10}$	3.94
	central	$1.47 \cdot 10^{-9}$	3.03	$1.13 \cdot 10^{-9}$	3.04
128	upwind	$1.04 \cdot 10^{-11}$	3.97	$1.11 \cdot 10^{-11}$	3.96
	central	$1.83 \cdot 10^{-10}$	3.01	$1.40 \cdot 10^{-10}$	3.01

Table 2: Average $\mathbb{L}^2$-error, computed by the `Trixi.jl` convergence test for the numerical solution of (103) with $N_p = 3, \mathrm{CFL} = 0.1$, compared to the exact solution (102) and corresponding average EOCs. Both measures are rounded to three significant figures.

Table 3 shows the $\mathbb{L}^2$-error of the `Trixi.jl` convergence test and the corresponding EOC, averaged analogously to the values in table 2 but for a polynomial degree of $N_p = 4$. Hence, the optimal convergence rate, one would expect is 5. Again, this is generally reached by the results. Apart from the last iteration, the EOCs constantly stay above a level of 4.7. This time the optimal convergence rate is reached for both, the upwind and the central numerical flux.

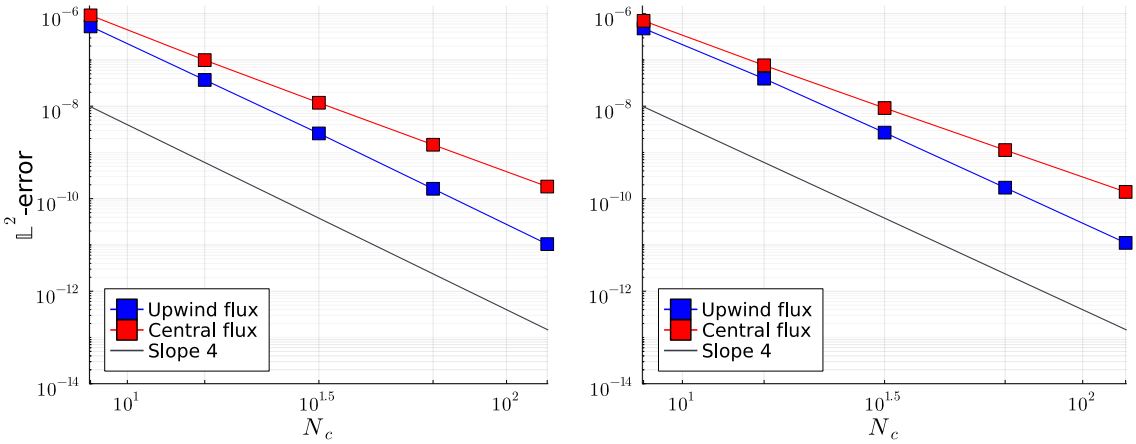
The $\mathbb{L}^2$-error takes values on a level of 10^{-9} for a polynomial degree of 4 and the lowest refinement level, $N_c = 8$. The smallest errors are reached for the maximal refinement level, $N_c = 128$, and take values on a level of 10^{-14}. In the last iteration the finite machine precision limits the convergence of the solution beyond a level of 10^{-14}, which is why the EOCs drop massively.

In figures 6 and 7, the results of tables 2 and 3 are visualized in convergence plots. They show the average $\mathbb{L}^2$-errors dependent on the respective refinement level for the different polynomial degrees and different numerical fluxes. The axes are scaled logarithmic, which is why the errors approximately appear linear with a slope corresponding to the respective EOC. The different convergence rates for the upwind and the central flux manifest in different slopes for $N_p = 3$, which can be well observed in figures 6. Figure 7 shows that for a polynomial degree of $N_p = 4$, the central flux performs slightly better than the upwind flux. Furthermore, in figure 7, the drop

N_c	Num. flux	Avg. $\mathbb{L}^2$-error (m = 1, ..., 6)	Avg. EOC (m = 1, ..., 6)	Avg. $\mathbb{L}^2$-error (m = 7, ..., 12)	Avg. EOC (m = 7, ..., 12)
8	upwind	$4.95 \cdot 10^{-9}$	-	$4.92 \cdot 10^{-9}$	-
	central	$5.28 \cdot 10^{-9}$	-	$4.84 \cdot 10^{-9}$	-
16	upwind	$1.77 \cdot 10^{-10}$	4.82	$1.85 \cdot 10^{-10}$	4.73
	central	$1.50 \cdot 10^{-10}$	5.14	$1.46 \cdot 10^{-10}$	5.03
32	upwind	$5.88 \cdot 10^{-12}$	4.91	$6.26 \cdot 10^{-12}$	4.89
	central	$4.07 \cdot 10^{-12}$	5.19	$3.92 \cdot 10^{-12}$	5.22
64	upwind	$1.95 \cdot 10^{-13}$	4.91	$2.06 \cdot 10^{-13}$	4.92
	central	$1.37 \cdot 10^{-13}$	4.89	$1.30 \cdot 10^{-13}$	4.92
128	upwind	$9.54 \cdot 10^{-14}$	1.03	$8.55 \cdot 10^{-14}$	1.30
	central	$9.02 \cdot 10^{-14}$	0.63	$8.06 \cdot 10^{-14}$	0.76

Table 3: Average $\mathbb{L}^2$-error, computed by the `Trixi.jl` convergence test for the numerical solution of (103) with $N_p = 4, \mathrm{CFL} = 0.1$, compared to the exact solution (102) and corresponding average EOCs. Both measures are rounded to three significant figures.

of the EOC in the last iteration, i.e. for the highest refinement level is clearly visible, as the slope drops from a value of approximately 5 to a a value below 1.



(a) Average $\mathbb{L}^2$-error in the first six components (b) Average $\mathbb{L}^2$-error in the last six components

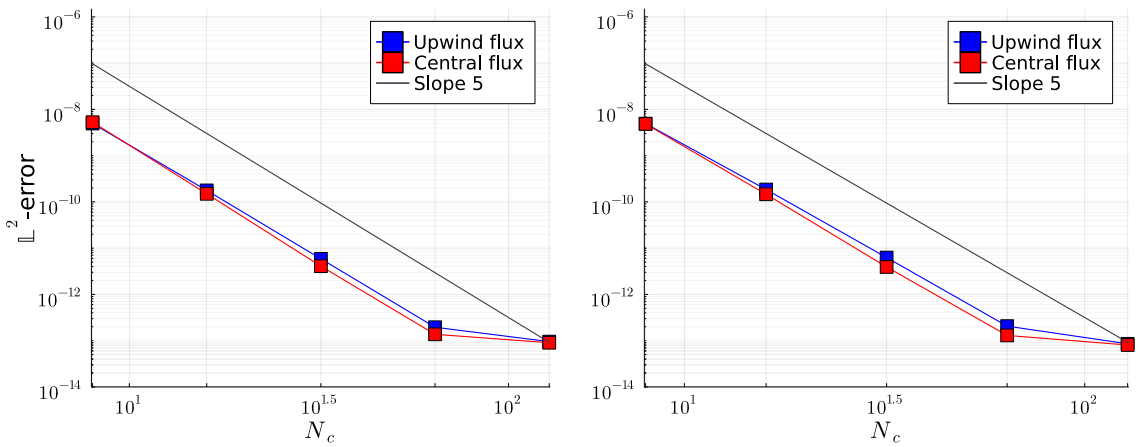
Figure 6: Plot of the average $\mathbb{L}^2$-error from tables 2 and 3 for different number of cells and $N_p = 3$.

The results verify the correctness of the implementations as the observed EOCs nearly reach the optimal convergence rate that could be expected for the approach. The lower convergence rates for the central flux and a polynomial degree of $N_p = 3$ are explained in the literature [18].

4.3 Energy Simulation

Our main result in section 3 was that the spatial discretization approach leads to an energy stable numerical solution. For zero boundary data and absent external forces and moments, it could be shown that the discrete energy is non increasing. Furthermore, we showed that in comparison to the central numerical flux, the upwind numerical flux has additional dissipation at the interfaces. In this section, we will validate the general statements about non increasing energy for zero external forces and moments. Additionally, we will investigate whether the additional dissipation of the upwind flux can be observed, when determining the discrete energy of two solutions whose setups differ only in the choice of the numerical flux.

In order to be able to perform the energy analysis, we need a method to actually compute the discrete energy of the solution that results from a `Trixi.jl` simulation. This is realized by writing a function `compute_energy`. In the following we will describe the idea behind the implementation



(a) Average of $\mathbb{L}^2$-error in the first six components. (b) Average of $\mathbb{L}^2$-error in the last six components.

Figure 7: Plot of the average $\mathbb{L}^2$-error from tables 2 and 3 for different number of cells and $N_p = 4$.

of this function. First, we recall the definition of the total discrete energy of the solution, which reads

$$\|U\|^2_{N_p,\Gamma} = \sum_{\mathrm{k}=1}^{N_c} \frac{\Delta x_{\mathrm{k}}}{2} \|U^{(\mathrm{k})}\|^2_{N_p,\Gamma}.$$

Suppose now that the solution `Trixi.jl` returns, is brought into the shape of a three dimensional array, `U`, of the size $12 \times N_x \times N_t$, where N_x is the total number of spatial nodes and N_t is the total number of timesteps executed by the ODE solver. Moreover, let the quadrature weights of the LGL quadrature in the different cells be ordered consecutively in an one dimensional array `weights` of size N_x. The term $\Delta x_k/2$ is still constant for all cells, because we use a uniform mesh.

In algorithm 2, the simplified code of the implementation of a function that computes the total discrete energy, given the above variables, is shown. In lines 3-7, the integrand that is defined by the discrete energy norm, is initialized and computed. In particular the two dimensional array `integrand` contains the integrand, evaluated at every LGL node in every cell and at every discrete time. Afterwards, in lines 10-15, the discrete energy is initialized and computed at every discrete point in time, using the LGL quadrature formula scaled by $\Delta x_k/2$. Finally, the total discrete energy is returned as an one dimensional array of size N_t.

Algorithm 2 Pseudo code of function to compute discrete energy.

```
1: function compute_energy(U, weights, equations::IntrinsicBeamEquation)
2:
3: integrand = zeros(N_x, N_t)
4: for j = 1 to N_t do
5:     for i = 1 to N_x do
6:         integrand[i,j] = U[:,i,j]^T * equations.Gamma * U[:,i,j]
7:     end for
8: end for
9:
10: energy = zeros(N_t)
11: for j = 1 to N_t do
12:     for i = 1 to N_x do
13:         energy[j] += Δx_k/2 * weights[i] * integrand[i,j]
14:     end for
15: end for
16:
17: return energy
```

To analyze the discrete energy, we can now write an elixir, that solves an arbitrary configuration of the intrinsic beam equation and afterwards computes the solution's energy using the above

Parameter	Value
ℓ	1.0
N_p	3
N_c	8, 16
T	4.0
CFL	0.1
IC (m $= 1, \dots, 6$)	$\sin\left(\frac{\pi}{2}(1-x)\right)$
IC (m $= 7, \dots, 12$)	$\sin\left(\frac{\pi}{2}x\right)$
$k(x)$	$\mathbf{0}_3$
$m_{ext}(x,t)$	$\mathbf{0}_3$
$f_{ext}(x,t)$	$\mathbf{0}_3$

Table 4: Setup used to generate the plots in figure 8. IC stands for initial condition and is meant component wise.

defined function. This energy can then be visualized by plotting it at the timesteps. We will use the same positive definite flexibility and mass matrices as in the convergence analysis, (101). As mentioned earlier, they should not be interpreted as values of flexibility and mass matrices describing the material properties of a realistic beam. The statement about energy stability holds for arbitrary flexibility and mass matrices as long as they are positive definite, and the purpose of the investigations in this section is to validate this statement numerically.

Figure 8 shows the total discrete energy of different numerical solutions associated with the setup given in table 4 at the discrete points in time. For figure 8a, the spatial domain was subdivided into 8 cells, i.e. $N_c = 8$ and for figure 8b we have $N_c = 16$. For each spatial refinement level, the numerical solution and the corresponding energy was determined for the central flux and for the upwind flux.

First of all, we can observe that the discrete energy is indeed decreasing in time. This holds for the upwind flux as well as for the central flux. As expected, the level of energy corresponding to the solution for the central flux is above the level of energy of the solution for the upwind flux all the time. This is due to the additional dissipation the upwind flux generates at the interfaces (cf. section 3.5.2). As we showed, the central flux does not have that dissipational effect but the corresponding solution still has a decreasing and not constant energy. This is caused by the boundary flux. At the boundaries, the numerical flux is always chosen as the upwind flux. As a consequence of that, there is always dissipation at the boundaries, which results in a decreasing energy even for the solution corresponding to the central flux.

What we also observe, comparing figure 8a to figure 8b, is that both, the energy of the solution for the upwind flux and the energy of the solution for the central flux, show less dissipation for a higher refinement level in space. This is associated with the convergence of the numerical solution. A better approximation of the exact solution means, that the energy of the exact solution is better approximated as well and for the energy of the exact solution, it was shown that it is constant for zero external forces and moments.

Before completing this section, we take one closer look at the energy that results from the simulation using the central numerical flux. Remember that for zero external forces and moments and the central flux, our result for the total discrete energy in section 3.5.2, was

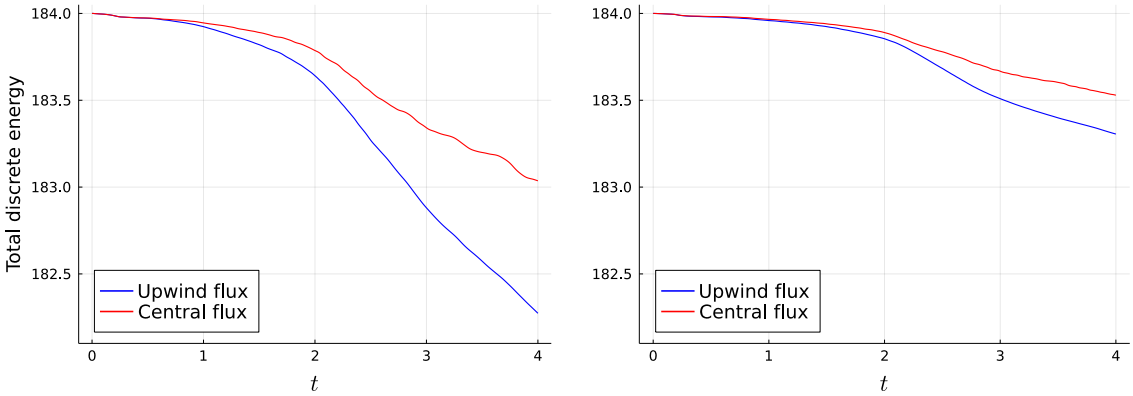
$$\|U(\cdot,t)\|^2_{N_p,\Gamma} = \|U_0\|^2_{N_p,\Gamma} - 2\int_0^t \big(b_L(\tau) + b_R(\tau)\big)\,\mathrm{d}\tau. \tag{104}$$

Moreover, for the boundary terms, we found

$$-b_L = -\big(Y^{(1)}\big)^T \mathbf{F}^{-1/2}\Psi^{-1/2}\mathbf{F}^{-1/2}Y^{(1)}\Big|_{\xi=-1},$$

$$-b_R = -\big(S^{(N_c)}\big)^T \mathbf{F}^{1/2}\Psi^{1/2}\mathbf{F}^{1/2}S^{(N_c)}\Big|_{\xi=1}.$$

After a simulation, $Y^{(1)}$ and $S^{(N_c)}$ are known approximately at every discrete point in time, t_n. Note that these are not the exact $Y^{(1)}$ and $S^{(N_c)}$, because the ODE in time representing the semi



(a) Energy for 8 cells. (b) Energy for 16 cells.

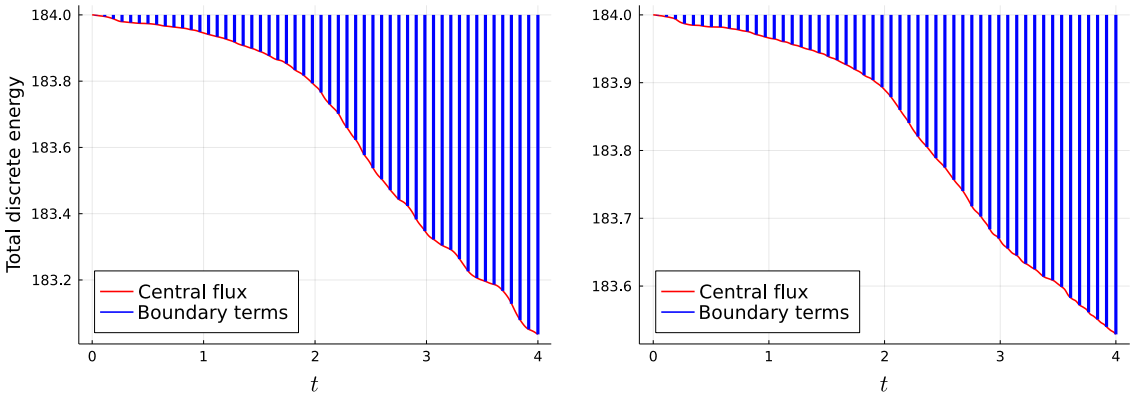
Figure 8: Total discrete energy of the solution with setup of table 4 and flexibility and mass matrix defined in (101).

discretization was solved numerically. Nevertheless, the boundary terms b_L, b_R can therefore be determined approximately and also the integral value on the right hand side of (104).

In particular, the integral values at the discrete points in time t_n can for example be approximated by

$$2\int_0^{t_n} \big(b_L(\tau) + b_R(\tau)\big)\,\mathrm{d}\tau \approx \sum_{i=1}^{n} (t_i - t_{i-1})\big(b_L(t_i) + b_R(t_i)\big) =: d_n \tag{105}$$

for $n = 1, \dots N_t$ and $d_0 = 0$. In figure 9, the approximated integral values d_n are visualized as bars together with the total discrete energy of the simulations with the central flux, which have already been plotted in figure 8. The origins of the bars are attached to the value of the energy at $t = 0$ and their length represents the value of the d_n and therefore the amount of dissipation, generated at the boundaries. For the simulation with 8 cells, there is a bar plotted at every 20-th timestep and for the simulation for 16 cells, the bars are plotted at every 40-th timestep.



(a) Energy and boundary dissipation for 8 cells. (b) Energy and boundary dissipation for 16 cells.

Figure 9: Total discrete energy for the central numerical flux of figure 8 and value of the dissipative boundary terms. (Note that the axes are scaled differently.)

It can be observed that the amount of dissipation at the boundaries matches the amount by which the discrete energy of the simulations with the central flux decreases. This validates the statement (104). In theory, there should be no difference between the left hand side of (104) and the right hand side. However, this is only the case if time integration is exact. In practice, the difference should be approximately zero. A measure for the difference of the two terms can be found via

$$\mathcal{E}^{diss}(t_n) := \Big| \|U(\cdot, t_n)\|^2_{N_p,\Gamma} - \|U_0\|_{N_p,\Gamma} - d_n \Big|$$

at every discrete point in time. For the simulation with 8 cells, the maximal difference is

$$\max_{n=0,\dots,N_t} \mathcal{E}^{diss}(t_n) \approx 1.33 \cdot 10^{-3},$$

while for the simulation with 16 cells, it amounts to

$$\max_{n=0,\dots,N_t} \mathcal{E}^{diss}(t_n) \approx 3.02 \cdot 10^{-4}.$$

Note that not only the time discretization of the semi discrete formulation, but also the approximation of the integral values with d_n in (105), has an impact on that difference. As the number of cells is increased, also the timesteps become smaller because of the CFL condition (cf. section 3.7). Therefore both, the numerical integration of the semi discrete formulation and the approximation d_n gets better and $\mathcal{E}^{diss}(t_n)$ is getting smaller for a higher number of cells.

Taking these considerations into account, the values of $\mathcal{E}^{diss}(t_n)$ for the two different simulations are sufficiently close to zero to validate the statement (104).

4.4 Determination of the Position Line

In this section, we want to demonstrate how the numerical solution of the intrinsic beam equation resulting from a `Trixi.jl` simulation can be used to visualize the corresponding beam. In particular, we will simulate a set up with realistic material parameters and constant external forces acting along the beam. The position line of the corresponding beam will be plotted and we will interpret the results. Note that the purpose of this section is not to obtain a physically exact solution but to show the post processing that is needed to determine the position line of the underlying beam out of the intrinsic variables.

Because the intrinsic beam model as it is considered here, does not include any damping mechanisms, it is difficult to simulate a setup in which a steady state solution is reached in finite time that could be compared to an exact steady state solution. Further, we did not find simulation results of other authors, that consider the same mechanical setup with the same assumptions as we do. In [5, 17, 48], there are numerical experiments for the intrinsic beam model and other geometrical exact models but they either do not consider "clamped-free" beams or use non-zero boundary conditions. We therefore limit ourselves to demonstrating how a modelled beam can be visualized and evaluating the sensibleness of the results in general.

The easiest way to obtain a setup for a dynamically behaving beam is to prescribe constant external forces, $f_{ext}(x,t) \equiv$ const., that act along the beam. As discussed in section 2.6.2, this cannot particularly be seen as a constant force acting along the beam as seen from the global coordinate system, because external forces are represented in the body attached coordinate system and therefore depend on the beam's deformation. Nevertheless, for deformations that are not too large, these external forces will at least be an approximation to a constant force in the global system. For the rest of this section, we consider a constant external force of

$$f_{ext}(x,t) \equiv (0,0,-10)^T.$$

Remember that the solution of the intrinsic beam model contains intrinsic variables only, i.e. internal forces and moments and linear and angular velocities. In order to obtain a visualization of the simulation, we therefore need some post processing to determine the deformation of the beam based on the numerical solution. We will see that the steps needed to determine the beam's deformation, include solving two ODEs.

First, we recall that the position of the deformed beam at any point x in $[0,\ell]$ and t in time, is denoted as $\boldsymbol{r}(x,t) \in \mathbb{R}^3$. Moreover, $\mathfrak{S}(x,t)$ was defined as the rotation matrix, that transforms vectors in the body attached coordinate system into their representation in the global coordinate system. According to [36], the position line of the deformed beam can be determined by solving the following ordinary differential equation

$$\boldsymbol{r}' = \mathfrak{S}(\gamma + e_1). \tag{106}$$

In order to find a solution of the above ODE, we first need to determine the matrix $\mathfrak{S}$, that is again obtained by solving an ODE, namely

$$(\mathfrak{S}^{-1})' = -(\tilde{\kappa} + \tilde{k})\mathfrak{S}^{-1}. \tag{107}$$

We will numerically solve these two ODEs, using the trapeze rule for the approximation of integrals, which eventually leads to a recursion formula for the beam's position at the discrete space time points (x_j, t_n). The idea behind this is the following: Let x_j and x_{j+1} be two adjacent spatial nodes. For the rest of the section, this is interpreted globally, meaning that x_0 is the first LGL node in the first cell, x_{N_p} is the last LGL node in the first cell, x_{N_p+1} is the first LGL node in the second cell and so on. Then, if we integrate (107) over the interval $[x_j, x_{j+1}]$, we get

$$\mathfrak{S}^{-1}(x_{j+1}, t) - \mathfrak{S}^{-1}(x_j, t) = -\int_{x_j}^{x_{j+1}} (\tilde{\kappa} + \tilde{k})\mathfrak{S}^{-1}\,\mathrm{d}x.$$

Both sides of the above equation are now evaluated at an arbitrary discrete point in time t_n and the integral on the right hand side is approximated by applying the trapeze rule. This results in

$$\begin{aligned}\mathfrak{S}^{-1}(x_{j+1}, t_n) - \mathfrak{S}^{-1}(x_j, t_n) \approx \\ -(x_{j+1} - x_j)\frac{(\tilde{k} + \tilde{\kappa})\mathfrak{S}^{-1}\big|_{(x_j,t_n)} + (\tilde{k} + \tilde{\kappa})\mathfrak{S}^{-1}\big|_{(x_{j+1},t_n)}}{2}.\end{aligned}$$

The equation can be solved for $\mathfrak{S}^{-1}(x_{j+1}, t_n)$. In particular, we have

$$\begin{aligned}&\mathfrak{S}^{-1}(x_{j+1}, t_n) \approx \\ &\left(\mathbf{I}_{3,3} + \frac{x_{j+1} - x_j}{2}(\tilde{k}(x_{j+1}) + \tilde{\kappa}(x_{j+1}, t_n))\right)^{-1}\left(\mathbf{I}_{3,3} - \frac{x_{j+1} - x_j}{2}(\tilde{k}(x_j) + \tilde{\kappa}(x_j, t_n))\right)\mathfrak{S}^{-1}(x_j, t_n).\end{aligned} \tag{108}$$

The numerical solution of the intrinsic beam equation is known at every LGL node and every discrete point in time after a `Trixi.jl` simulation. By applying the constitutive law (4), we can therefore determine the corresponding discrete values of κ. That is, we can interpret (108) as a recursive formula to determine the values of $\mathfrak{S}^{-1}$ at the LGL nodes for every timestep t_n. What is left to complete the recursion, is an initial value for the first node x_0 for every time t_n.

For our purposes, the initial value will simply be the identity matrix in $\mathbb{R}^{3\times 3}$, i.e.

$$\mathfrak{S}^{-1}(x_0, t_n) = \mathbf{I}_{3,3}$$

for every discrete time t_n. This is because we consider a clamped beam with zero velocities at the clamped end and therefore the origin of the beam does not rotate and is represented by the same point, no matter in which coordinate system we express it. For non-zero angular velocities, one would have to integrate the velocities at the first spatial node in $[0, t_n]$ in order to determine the angle the origin has rotated by. Then $\mathfrak{S}^{-1}(x_0, t_n)$ can be set as a rotation matrix for every point in time accordingly.

Inverting the discrete values of $\mathfrak{S}^{-1}$ from the recursion (108) gives us the values of $\mathfrak{S}$. These can then be used to determine the discrete values of the beam's position line $\boldsymbol{r}$ as approximate solution of the ODE (106). Therefore, the same approach as before is used, i.e. integrating the equation on both sides over $[x_j, x_{j+1}]$, approximating the right hand side at a discrete point in time t_n by the trapeze rule and then solving for $\boldsymbol{r}(x_{j+1}, t_n)$. This leads to the following recursion formula:

$$\begin{aligned}&\boldsymbol{r}(x_{j+1}, t_n) \approx \\ &\qquad \boldsymbol{r}(x_j, t_n) + \frac{x_{j+1} - x_j}{2}\big(\mathfrak{S}(x_{j+1}, t_n)(\gamma(x_{j+1}, t_n) + e_1) + \mathfrak{S}(x_j, t_n)(\gamma(x_j, t_n) + e_1)\big).\end{aligned}$$

Again, to complete the recursion, an initial value for $\boldsymbol{r}$ at the first node x_0 is needed. For this, we choose a point in $\mathbb{R}^3$, where we want the beam's origin to be located in space at the beginning of the simulation. This will be the point $(0, 0, 0)$. Because the beam is clamped and the external boundary data is assumed to be zero, the linear velocities are zero at the clamped end and the origin of the beam does not move. This means that the initial value of the iteration is $(0, 0, 0)$ for every t_n. In the case of non-zero linear velocities at the boundary, one would have to integrate the linear velocities at x_0 in $[0, t_n]$ and set the initial value of $\boldsymbol{r}(x_0, t_n)$ accordingly.

With the above recursion formulas, we are now able to determine the approximate position of the beam for every spatial node and every discrete point in time on the base of the discrete values

Parameter	Value
ℓ	4.0
N_p	3
N_c	16
CFL	1.0
IC	$\mathbf{0}_{12}$
$k(x)$	$\mathbf{0}_3$
$m_{ext}(x,t)$	$\mathbf{0}_3$
$f_{ext}(x,t)$	$(0,0,-10)^T$
NF	upwind

Table 5: Setup used to generate the plots in figures 10 - 12. IC stands for initial condition, NF for numerical flux.

of the intrinsic variables. After implementing the recursions, the deformation of a simulated beam can be plotted, which will be done in the following.

The flexibility and mass matrix, that were used to obtain the results in the following investigations are chosen as follows:

$$\mathbf{F} = \operatorname{diag}\left(10^4, 10^4, 10^4, 500, 500, 500\right)^{-1},$$

$$\mathbf{M} = \operatorname{diag}(1, 1, 1, 20, 10, 10).$$

These values are for example used for simulations of flexible beams in [17, 48]. The setup that was used to obtain the numerical results of this section, is given in table 5.

To demonstrate the dynamic behaviour of the beam, we show some snap shots of its position line at certain points in time in figure 10. Figure 10a shows the beam's position line at five points in time in the time interval [0, 1.28], whereas in figure 10b, we can see the position line at five points in time within the interval [1.28, 2.56].

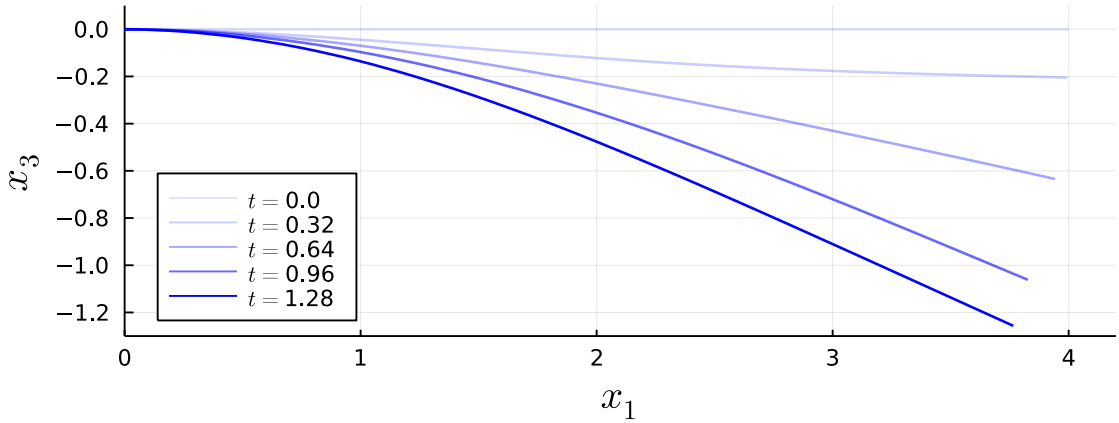
Although the results cannot be seen as simulation of a real application, we want to discuss whether they are generally reasonable in the considered context. First of all, note that because the external force we prescribe acts only in x_3-direction, deformations only take place in the x_1-x_3-plane and it suffices to plot the beam's position line in this plane. This has also been validated by investigating the x_2-component of the discrete values of $\boldsymbol{r}$. They are constantly zero.

We observe that the negative external force acting in x_3-direction leads to a beam deforming in negative x_3-direction as one would expect. For $t \in [0, 1.28]$, the beam's deflection is successively increasing until it reaches its maximum at $t = 1.28$. After that, for $t \in [1.28, 2.56]$, the deflection decreases until the position line seemingly reaches its initial position again at $t = 2.56$. In figure 11 the position of the beam's tip is shown in the time interval [0, 2.56] (figure 11a) but also for a longer simulation in the time interval [0, 36] (figure 11b). In particular, the plots show the x_3-coordinate of the tip position in time. It can be observed that the beam swings periodically. Nevertheless, the initial position is not quite reached after the first period of swinging. In particular, the amplitude at which the beam swings, is becoming smaller in the first half of the longer simulation, before it increases again in the second half.

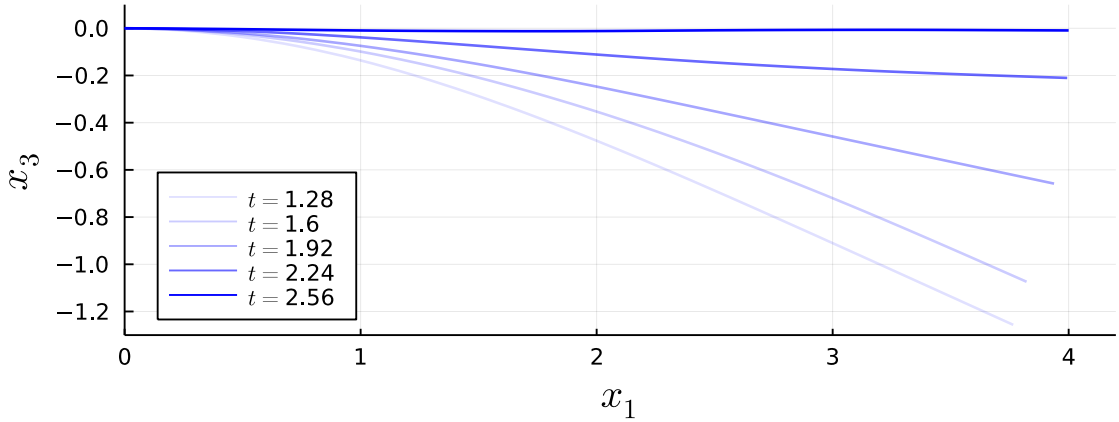
Figure 12 shows the energy of the beam for the shorter simulation as well as for the longer simulation. The results are in line with the observations for the deflection. The short term behaviour of the energy in figure 12a shows that the energy increases until the maximum deflection is reached at $t \approx 1.28$. Afterwards, we observe a decreasing energy until the beam returns to the minimal deflection state at $t \approx 2.56$. In the long term (figure 12b), the energy periodically increases and decreases with an amplitude that is getting smaller in the first half of the simulation and then getting bigger again in the second half.

In a realistic simulation with constant external forces (as seen from the global coordinate system), one would expect that the amplitude of the deflection decreases in time, whereas in our simulation, we observe a periodical swinging with an amplitude that slightly decreases for some time but then increases until the initial amplitude is reached again.

This unphysical behaviour might on the one hand be caused by the external forces, we chose. It was discussed earlier that they cannot particularly be interpreted as constant forces as seen from



(a) Deformation in $[0, 1.28]$.

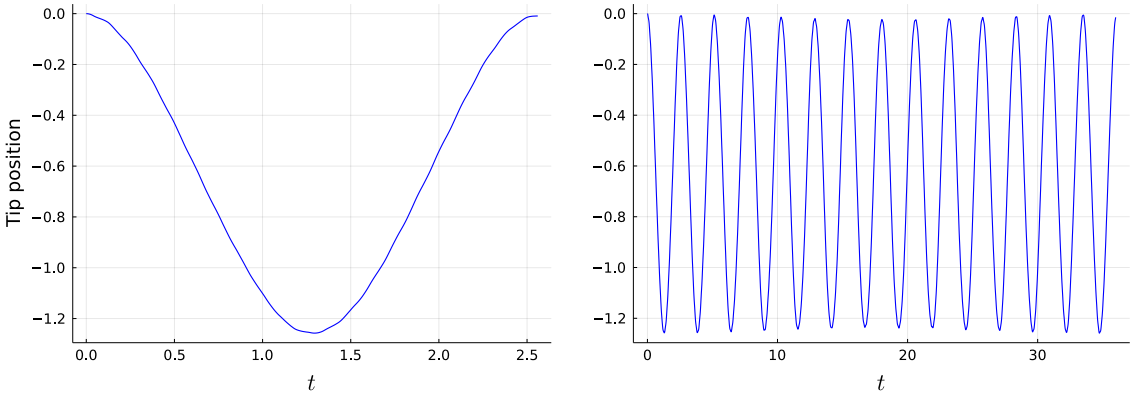


(b) Deformation in $[1.28, 2.56]$.

Figure 10: Deformation of a beam undergoing constant external forces.

the global coordinate system. On the other hand, we do not consider any physical damping mechanisms, that would simulate the decreasing amplitude which one would expect from a realistically swinging beam.

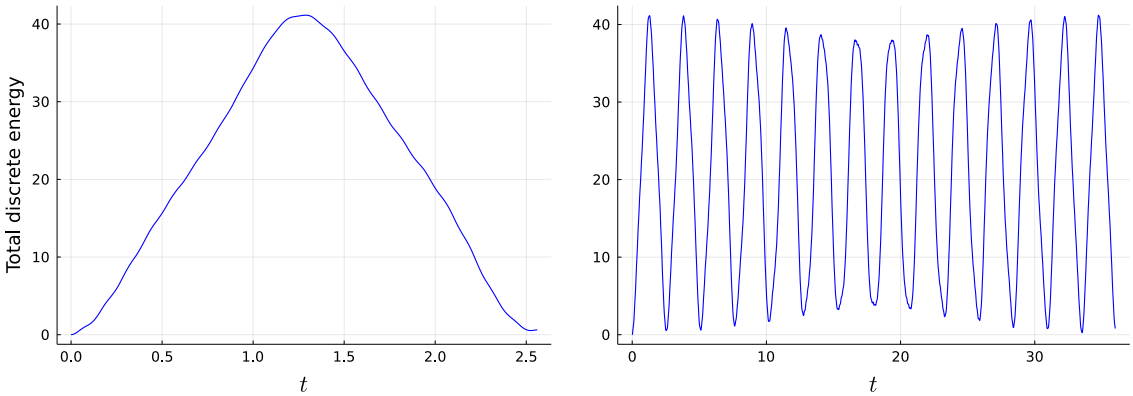
Under the circumstances considered here, we classify the results as sensible and we showed that the intrinsic variables can be post processed to determine the position line of the simulated beam. However, the results of this section do not allow us to evaluate the applicability of the DG approach in combination with the intrinsic beam model with regard to simulations of realistic beams. This will be discussed more detailed in the outlook in section 5.2.



(a) Tip position in time interval [0, 2.56].

(b) Tip position in time interval [0, 36].

Figure 11: x_3-coordinate of the position of the deforming beam's tip.



(a) Energy in the time interval [0, 2.56].

(b) Energy in the time interval [0, 36].

Figure 12: Discrete energy of the deforming beam.

5 Conclusion and Outlook

In the following two subsections we want to recapitulate the investigations that were made in this thesis. In the conclusion, we will summarize our results from the previous sections and evaluate them with regard to the objectives we formulated initially. In the outlook, we want to suggest some further investigations for the future.

5.1 Conclusion

The main objective we formulated at the beginning of the thesis was to derive an energy stable Discontinuous Galerkin approach for the discretization of the intrinsic beam model in space. In order to obtain such a discretization scheme, the intrinsic beam model was reformulated and classified as a system of linear hyperbolic balance laws. Different equivalent formulations were derived. The formulation for characteristic variables enabled us to provide the governing equations with mathematically appropriate boundary conditions and an initial condition. These were then brought into a form that suits the modelling of a clamped beam that is free swinging at one side. Moreover, we derived a capacity form of the intrinsic beam equation. Together with the initial and boundary conditions, this resulted in the formulation of an initial boundary value problem which could be investigated with regard to the energy of a potential solution.

We were able to show that the energy of a solution of the initial boundary value problem behaves in an expected way from a physical point of view, in the sense that the total mechanical energy of the modelled beam is conserved. Moreover, we showed that even for non-zero external forces and moments the solution of the problem is energy stable. To this end, we had to make some additional assumptions. In particular, we derived energy stability for zero external boundary data and in time constant and bounded external forces and moments and showed that with these assumptions the energy norm of the solution can increase linearly in time at maximum.

Having completed the analytical considerations, we used the techniques of DG methods to derive a discretization scheme for the spatial discretization of the initial boundary value problem. This included introducing a stable numerical flux and implementing the boundary conditions into the discretization scheme. The result was a semi discrete formulation of the initial boundary value problem, that still continuously depended on the time. Energy statements, analogous to the energy statements for the solution of the continuous problem, could be made for the numerical solution of the semi discrete formulation. More precisely, it was shown that energy conservation does hold cell wise in the discrete context. On a global point of view, the discrete energy has additional numerical dissipation that is generated at the interfaces between the cells and by the boundary terms. The dissipation at the interfaces can be controlled by the choice of the numerical flux, whereas the boundary terms always generate numerical dissipation. Together with a bound for the contribution of external sources and moments that was analogue to the corresponding bound in the continuous analysis, this led to energy stability of the numerical solution. Similar to the result in the continuous energy analysis, it could be shown that the discrete energy norm of the numerical solution does not grow faster than linearly. The latter result was, in fact, the property we were aiming for: we showed that the Discontinuous Galerkin approach we used is indeed energy stable.

Our theoretical results have been validated by numerical experiments. We simulated different configurations and investigated the results with regard to their discrete energy. The investigations have shown that the theoretical predictions for the discrete energy are fulfilled. However, the numerical experiments have not been made for the most general case. We looked into constant flexibility and mass matrices only. This will also be addressed in the outlook.

All in all, the DG discretization for the intrinsic beam model yields excellent mathematical results. The convergence tests showed that an optimal convergence rate can be reached for test problems and the theoretical investigations as well as the numerical experiments confirm the stability of the approach.

5.2 Outlook

The results of this thesis indicate that, from a mathematical point of view, the derived DG approach suits the discretization of the intrinsic beam model very well. What remains to complete the validation of our theoretical results, is to extend the implementations in order to be able to simulate configurations with non-constant flexibility and mass matrices. The numerical experiments can

then be repeated for non-constant matrices. It will be interesting to see if the experimental order of convergence is influenced by this or if the optimal convergence rate is still approximately reached for such simulations. Aside from that, the energy statements that were derived theoretically should also be validated numerically for non-constant matrices.

Another point that will be investigated on the theoretical side, is a generalization of the energy stability statement for non-zero external boundary data – in the continuous context as well as in the discrete context. Such an extension is particularly interesting for the application on helicopter rotor blades as they naturally rotate and move in space. It is not ensured that such a generalization is possible, but future investigations could either look into an extension of the statement for more general external data, or even prove that energy stability cannot be obtained if the external boundary data is not zero. Either way, we suppose that a more detailed look into this field could bring interesting results.

Beyond these theoretical aspects, there are several possibilities of further practical investigations concerning realistic applications. The extension of the implementations to non-constant flexibility and mass matrices is not only important to validate the mathematical results, but also with regard to realistic simulations of beams. As mentioned in section 2, in real applications of anisotropic beams, material properties vary along the beam and the possibility to consider non-constant matrices will therefore be essential for their simulation. In the future, the theoretical and practical considerations could even be extended to take into account flexibility and mass matrices that are discontinuous, i.e. have jumps. This would open the possibility to simulate beams whose material properties do not change smoothly but abruptly, which is often the case in applications. The nature of DG methods, to allow jumps at interfaces, could very well be suitable for such simulations. The stability of DG approaches for the discretization of hyperbolic PDEs with coefficient matrices that have jumps is for example investigated in [28].

It would also be interesting to try applying the presented discretization approach to benchmark problems for the simulation of flexible beams and see how it performs in general, but also compared to other models and discretization approaches. In the course of this, it might be useful to add damping mechanisms, in order obtain simulations that reach steady states in finite time. Damping mechanisms for the intrinsic beam model are for example considered in [3].

To complete this section, we would like to mention another point that has not been taken into account yet. This is the efficiency of the scheme as it is presented here. In particular, this concerns the time stepping. Recall that in the simplified notation of section 3.7, the timestep was chosen according to the CFL-condition

$$\Delta t \leq \mathrm{CFL}\,\frac{1}{\max|\mathbf{\Lambda}_{ii}|}\min\Delta x,$$

where $\max|\mathbf{\Lambda}_{ii}|$ represents the maximal absolute eigenvalue of the advection matrix A. Moreover, recall that the eigenvalues of A were shown to be the square roots of the eigenvalues of Ψ and also of $(\mathbf{FM})^{-1}$ (cf. section 2.4.2). In realistic applications, beams are often flexible in one direction but almost rigid in another. If one thinks of a helicopter rotor blade for example, the blade is very flexible in the "up" and "down" direction but not in the "left" and "right" direction. Such material properties result in very small entries in the flexibility matrix. Note that the flexibility matrix with realistic parameters, that was used to plot the deforming beam in section 4.4, describes a relatively flexible beam, and has diagonal entries of 10^{-4}, nonetheless. The maximal absolute eigenvalue of A is 100 in this case. For the biggest possible CFL-number, CFL $= 1.0$, a polynomial degree of $N_p = 3$ and a number of cells $N_c = 16$ (which is the setup used in section 4.4), this results in a timestep of $\Delta t = 6.25\cdot 10^{-4}$. For beams whose flexibility is arbitrarily small in some directions, the eigenvalues of A can theoretically become arbitrarily big and the timestep in turn arbitrarily small. Simulations of such beams are therefore computationally very intensive. One could therefore also consider, for which beams the discretization approach is practicable in the way it is presented here, and for which it is not. There might be found a way to avoid such small timesteps by replacing almost zero entries in the flexibility matrix by zeros and reduce the degrees of freedom, but this has to be part of future investigations.

A Appendices

A.1 Computation of Boundary Terms

In this appendix, we will determine the value of the boundary terms b_L and b_R from section 3.5.2. In order to do so, we will firstly determine the matrices A_+ and A_-, secondly the matrices ΓA_+ and ΓA_-, thirdly execute the matrix vector multiplications in the boundary terms and fourthly apply the discrete boundary conditions and gather terms.

Before we actually start with the described procedure above, we recall the definition of the matrices needed:

$$
\begin{aligned}
T &= \frac{1}{2}\begin{pmatrix} \mathbf{F}^{-1/2}\mathcal{X}^T & \mathbf{F}^{-1/2}\mathcal{X}^T \\ \mathbf{F}^{1/2}\mathcal{X}^T\Lambda & -\mathbf{F}^{1/2}\mathcal{X}^T\Lambda \end{pmatrix}, & T^{-1} &= \begin{pmatrix} \mathcal{X}\mathbf{F}^{1/2} & \Lambda^{-1}\mathcal{X}\mathbf{F}^{-1/2} \\ \mathcal{X}\mathbf{F}^{1/2} & -\Lambda^{-1}\mathcal{X}\mathbf{F}^{-1/2} \end{pmatrix}, \\
\mathbf{\Lambda}_- &= \begin{pmatrix} -\Lambda & \mathbf{0}_{6,6} \\ \mathbf{0}_{6,6} & \mathbf{0}_{6,6} \end{pmatrix}, & \mathbf{\Lambda}_+ &= \begin{pmatrix} \mathbf{0}_{6,6} & \mathbf{0}_{6,6} \\ \mathbf{0}_{6,6} & \Lambda \end{pmatrix}, \\
\Gamma &= \begin{pmatrix} \mathbf{F} & \mathbf{0}_{6,6} \\ \mathbf{0}_{6,6} & \mathbf{M} \end{pmatrix}, & A &= -\begin{pmatrix} \mathbf{0}_{6,6} & \mathbf{F}^{-1} \\ \mathbf{M}^{-1} & \mathbf{0}_{6,6} \end{pmatrix} \\
\Pi &= -\begin{pmatrix} \mathbf{0}_{6,6} & \mathbf{I}_{6,6} \\ \mathbf{I}_{6,6} & \mathbf{0}_{6,6} \end{pmatrix} & \Psi &= \mathbf{F}^{-1/2}\mathbf{M}^{-1}\mathbf{F}^{-1/2}.
\end{aligned}
$$

The matrix Ψ is positive definite and can be diagonalized by the orthogonal matrix $\mathcal{X}$ (cf. section 2.4.2):

$$\Psi = \mathcal{X}^T\Lambda^2\mathcal{X}.$$

We can show easily that

$$\Psi^{1/2} = \mathcal{X}^T\Lambda\mathcal{X},$$

as

$$\Psi^{1/2}\Psi^{1/2} = \mathcal{X}^T\Lambda\mathcal{X}\mathcal{X}^T\Lambda\mathcal{X} = \mathcal{X}^T\Lambda^2\mathcal{X} = \Psi$$

holds. Also the inverses of Ψ and $\Psi^{1/2}$ are given by

$$
\begin{aligned}
\Psi^{-1} &= \mathbf{F}^{1/2}\mathbf{M}\mathbf{F}^{1/2}, \\
\Psi^{-1/2} &= \mathcal{X}^T\Lambda^{-1}\mathcal{X}.
\end{aligned}
$$

Another identity that will be useful in the upcoming computations is

$$\mathbf{M}\mathbf{F}^{1/2} = \mathbf{F}^{-1/2}\mathbf{F}^{1/2}\mathbf{M}\mathbf{F}^{1/2} = \mathbf{F}^{-1/2}\Psi^{-1}. \tag{A.1}$$

Now, the matrices A_+ and A_- are defined in section 3.3 as

$$
\begin{aligned}
A_+ &= T\mathbf{\Lambda}_+T^{-1}, \\
A_- &= T\mathbf{\Lambda}_-T^{-1}.
\end{aligned}
$$

And, thus, the matrix A_+ explicitly reads

$$
\begin{aligned}
A_+ &= \frac{1}{2}\begin{pmatrix} \mathbf{F}^{-1/2}\mathcal{X}^T & \mathbf{F}^{-1/2}\mathcal{X}^T \\ \mathbf{F}^{1/2}\mathcal{X}^T\Lambda & -\mathbf{F}^{1/2}\mathcal{X}^T\Lambda \end{pmatrix}\begin{pmatrix} \mathbf{0}_{6,6} & \mathbf{0}_{6,6} \\ \mathbf{0}_{6,6} & \Lambda \end{pmatrix}T^{-1} \\
&= \frac{1}{2}\begin{pmatrix} \mathbf{0}_{6,6} & \mathbf{F}^{-1/2}\mathcal{X}^T\Lambda \\ \mathbf{0}_{6,6} & -\mathbf{F}^{1/2}\mathcal{X}^T\Lambda^2 \end{pmatrix}\begin{pmatrix} \mathcal{X}\mathbf{F}^{1/2} & \Lambda^{-1}\mathcal{X}\mathbf{F}^{-1/2} \\ \mathcal{X}\mathbf{F}^{1/2} & -\Lambda^{-1}\mathcal{X}\mathbf{F}^{-1/2} \end{pmatrix} \\
&= \frac{1}{2}\begin{pmatrix} \mathbf{F}^{-1/2}\mathcal{X}^T\Lambda\mathcal{X}\mathbf{F}^{1/2} & -\mathbf{F}^{-1} \\ -\mathbf{F}^{1/2}\mathcal{X}^T\Lambda^2\mathcal{X}\mathbf{F}^{1/2} & \mathbf{F}^{1/2}\mathcal{X}^T\Lambda\mathcal{X}\mathbf{F}^{-1/2} \end{pmatrix} \\
&= \frac{1}{2}\begin{pmatrix} \mathbf{F}^{-1/2}\Psi^{1/2}\mathbf{F}^{1/2} & -\mathbf{F}^{-1} \\ -\mathbf{F}^{1/2}\Psi\mathbf{F}^{1/2} & \mathbf{F}^{1/2}\Psi^{1/2}\mathbf{F}^{-1/2} \end{pmatrix} \\
&= \frac{1}{2}\begin{pmatrix} \mathbf{F}^{-1/2}\Psi^{1/2}\mathbf{F}^{1/2} & -\mathbf{F}^{-1} \\ -\mathbf{M}^{-1} & \mathbf{F}^{1/2}\Psi^{1/2}\mathbf{F}^{-1/2} \end{pmatrix}.
\end{aligned}
$$

In a similar way, the matrix A_- can be determined:

$$\begin{aligned}
A_- &= \frac{1}{2}\begin{pmatrix} \mathbf{F}^{-1/2}\mathcal{X}^T & \mathbf{F}^{-1/2}\mathcal{X}^T \\ \mathbf{F}^{1/2}\mathcal{X}^T\Lambda & -\mathbf{F}^{1/2}\mathcal{X}^T\Lambda \end{pmatrix}\begin{pmatrix} -\Lambda & \mathbf{0}_{6,6} \\ \mathbf{0}_{6,6} & \mathbf{0}_{6,6} \end{pmatrix} T^{-1} \\
&= \frac{1}{2}\begin{pmatrix} -\mathbf{F}^{-1/2}\mathcal{X}^T\Lambda & \mathbf{0}_{6,6} \\ -\mathbf{F}^{1/2}\mathcal{X}^T\Lambda^2 & \mathbf{0}_{6,6} \end{pmatrix}\begin{pmatrix} \mathcal{X}\mathbf{F}^{1/2} & \Lambda^{-1}\mathcal{X}\mathbf{F}^{-1/2} \\ \mathcal{X}\mathbf{F}^{1/2} & -\Lambda^{-1}\mathcal{X}\mathbf{F}^{-1/2} \end{pmatrix} \\
&= \frac{1}{2}\begin{pmatrix} -\mathbf{F}^{-1/2}\mathcal{X}^T\Lambda\mathcal{X}\mathbf{F}^{1/2} & -\mathbf{F}^{-1} \\ -\mathbf{F}^{1/2}\mathcal{X}^T\Lambda^2\mathcal{X}\mathbf{F}^{1/2} & -\mathbf{F}^{1/2}\mathcal{X}^T\Lambda\mathcal{X}\mathbf{F}^{-1/2} \end{pmatrix} \\
&= \frac{1}{2}\begin{pmatrix} -\mathbf{F}^{-1/2}\Psi^{1/2}\mathbf{F}^{1/2} & -\mathbf{F}^{-1} \\ -\mathbf{F}^{1/2}\Psi\mathbf{F}^{1/2} & -\mathbf{F}^{1/2}\Psi^{1/2}\mathbf{F}^{-1/2} \end{pmatrix} \\
&= \frac{1}{2}\begin{pmatrix} -\mathbf{F}^{-1/2}\Psi^{1/2}\mathbf{F}^{1/2} & -\mathbf{F}^{-1} \\ -\mathbf{M}^{-1} & -\mathbf{F}^{1/2}\Psi^{1/2}\mathbf{F}^{-1/2} \end{pmatrix}.
\end{aligned}$$

The correctness of the previous results can be verified by checking if the equation $A = A_+ + A_-$ holds, which it does.

The multiplication of A_+ by the matrix Γ from the left now yields

$$\begin{aligned}
\Gamma A_+ &= \frac{1}{2}\begin{pmatrix} \mathbf{F} & \mathbf{0}_{6,6} \\ \mathbf{0}_{6,6} & \mathbf{M} \end{pmatrix}\begin{pmatrix} \mathbf{F}^{-1/2}\Psi^{1/2}\mathbf{F}^{1/2} & -\mathbf{F}^{-1} \\ -\mathbf{M}^{-1} & \mathbf{F}^{1/2}\Psi^{1/2}\mathbf{F}^{-1/2} \end{pmatrix} \\
&= \frac{1}{2}\begin{pmatrix} \mathbf{F}^{1/2}\Psi^{1/2}\mathbf{F}^{1/2} & -\mathbf{I}_{6,6} \\ -\mathbf{I}_{6,6} & \mathbf{M}\mathbf{F}^{1/2}\Psi^{1/2}\mathbf{F}^{-1/2} \end{pmatrix} \\
&= \frac{1}{2}\begin{pmatrix} \mathbf{F}^{1/2}\Psi^{1/2}\mathbf{F}^{1/2} & -\mathbf{I}_{6,6} \\ -\mathbf{I}_{6,6} & \mathbf{F}^{-1/2}\Psi^{-1/2}\mathbf{F}^{-1/2} \end{pmatrix}.
\end{aligned}$$

To show the last equality in the above calculation, one uses the identity (A.1). For A_-, the multiplication with Γ yields

$$\begin{aligned}
\Gamma A_- &= \frac{1}{2}\begin{pmatrix} \mathbf{F} & \mathbf{0}_{6,6} \\ \mathbf{0}_{6,6} & \mathbf{M} \end{pmatrix}\begin{pmatrix} -\mathbf{F}^{-1/2}\Psi^{1/2}\mathbf{F}^{1/2} & -\mathbf{F}^{-1} \\ -\mathbf{M}^{-1} & -\mathbf{F}^{1/2}\Psi^{1/2}\mathbf{F}^{-1/2} \end{pmatrix} \\
&= \frac{1}{2}\begin{pmatrix} -\mathbf{F}^{1/2}\Psi^{1/2}\mathbf{F}^{1/2} & -\mathbf{I}_{6,6} \\ -\mathbf{I}_{6,6} & -\mathbf{M}\mathbf{F}^{1/2}\Psi^{1/2}\mathbf{F}^{-1/2} \end{pmatrix} \\
&= \frac{1}{2}\begin{pmatrix} -\mathbf{F}^{1/2}\Psi^{1/2}\mathbf{F}^{1/2} & -\mathbf{I}_{6,6} \\ -\mathbf{I}_{6,6} & -\mathbf{F}^{-1/2}\Psi^{-1/2}\mathbf{F}^{-1/2} \end{pmatrix},
\end{aligned}$$

where (A.1) can be used to show the latter equation. Again, the correctness of the two matrix multiplications can be verified by checking if $\Gamma A_+ + \Gamma A_- = \Gamma A = \Pi$ holds, which it does.

We can now come to the computation of b_L and b_R. Starting with b_L, by definition we have

$$b_L = U_R^T\left((\Gamma A_+)^* U_L + (\Gamma A_-)^* U_R - \frac{1}{2}\Pi U_R\right).$$

In the following we will omit the superscript $(\,\cdot\,)^*$, to ease the notation. We consider the above boundary term summand by summand. For the first summand, we have

$$\begin{aligned}
U_R^T\Gamma A_+ U_L &= \frac{1}{2}\begin{pmatrix} S_R \\ Y_R \end{pmatrix}^T \begin{pmatrix} \mathbf{F}^{1/2}\Psi^{1/2}\mathbf{F}^{1/2} & -\mathbf{I}_{6,6} \\ -\mathbf{I}_{6,6} & \mathbf{F}^{-1/2}\Psi^{-1/2}\mathbf{F}^{-1/2} \end{pmatrix}\begin{pmatrix} S_L \\ Y_L \end{pmatrix} \\
&= \frac{1}{2}\left(S_R^T\mathbf{F}^{1/2}\Psi^{1/2}\mathbf{F}^{1/2}S_L - S_L^T Y_R - S_R^T Y_L + Y_R^T\mathbf{F}^{-1/2}\Psi^{-1/2}\mathbf{F}^{-1/2}Y_L\right)
\end{aligned}$$

Inserting the discrete boundary condition for S_L and Y_L with zero external boundary data, namely

$$\begin{pmatrix} S_L \\ Y_L \end{pmatrix} = \begin{pmatrix} S_R + \mathbf{F}^{-1/2}\Psi^{-1/2}\mathbf{F}^{-1/2}Y_R \\ \mathbf{0}_3 \end{pmatrix}$$

gives us

$$\begin{aligned} U_R^T \Gamma A_+ U_L &= \frac{1}{2}\big(S_R^T \mathbf{F}^{1/2}\Psi^{1/2}\mathbf{F}^{1/2} S_R + S_R^T Y_R - S_R^T Y_R - Y_R^T \mathbf{F}^{-1/2}\Psi^{-1/2}\mathbf{F}^{-1/2} Y_R \\ &\qquad - S_R^T \mathbf{0}_3 + Y_R^T \mathbf{F}^{-1/2}\Psi^{-1/2}\mathbf{F}^{-1/2}\mathbf{0}_3\big) \\ &= \frac{1}{2}\big(S_R^T \mathbf{F}^{1/2}\Psi^{1/2}\mathbf{F}^{1/2} S_R - Y_R^T \mathbf{F}^{-1/2}\Psi^{-1/2}\mathbf{F}^{-1/2} Y_R\big). \end{aligned}$$

The second summand is

$$\begin{aligned} U_R^T \Gamma A_- U_R &= \frac{1}{2}\begin{pmatrix} S_R \\ Y_R \end{pmatrix}^T \begin{pmatrix} -\mathbf{F}^{1/2}\Psi^{1/2}\mathbf{F}^{1/2} & -\mathbf{I}_{6,6} \\ -\mathbf{I}_{6,6} & -\mathbf{F}^{-1/2}\Psi^{-1/2}\mathbf{F}^{-1/2} \end{pmatrix} \begin{pmatrix} S_R \\ Y_R \end{pmatrix} \\ &= \frac{1}{2}\big(- S_R^T \mathbf{F}^{1/2}\Psi^{1/2}\mathbf{F}^{1/2} S_R - Y_R^T \mathbf{F}^{-1/2}\Psi^{-1/2}\mathbf{F}^{-1/2} Y_R - 2Y_R^T S_R\big). \end{aligned}$$

Finally, the last and third summand can be represented by

$$-\frac{1}{2} U_R^T \Pi U_R = \frac{1}{2}\begin{pmatrix} S_R \\ Y_R \end{pmatrix}^T \begin{pmatrix} \mathbf{0}_{6,6} & \mathbf{I}_{6,6} \\ \mathbf{I}_{6,6} & \mathbf{0}_{6,6} \end{pmatrix} \begin{pmatrix} S_R \\ Y_R \end{pmatrix} = Y_R^T S_R.$$

Putting all three summands together again, we obtain

$$\begin{aligned} b_L = \; & \frac{1}{2}\big(S_R^T \mathbf{F}^{1/2}\Psi^{1/2}\mathbf{F}^{1/2} S_R - Y_R^T \mathbf{F}^{-1/2}\Psi^{-1/2}\mathbf{F}^{-1/2} Y_R\big) \\ & + \frac{1}{2}\big(- S_R^T \mathbf{F}^{1/2}\Psi^{1/2}\mathbf{F}^{1/2} S_R - Y_R^T \mathbf{F}^{-1/2}\Psi^{-1/2}\mathbf{F}^{-1/2} Y_R - 2Y_R^T S_R\big) + Y_R^T S_R \\ = \; & -Y_R^T \mathbf{F}^{-1/2}\Psi^{-1/2}\mathbf{F}^{-1/2} Y_R. \end{aligned}$$

The same procedure can be used to determine the right boundary term b_R. By definition this is

$$b_R = U_L^T \left((\Gamma A_+)^* U_L + (\Gamma A_-)^* U_R - \frac{1}{2}\Pi U_L \right),$$

which we consider summand by summand. Again, we omit the superscript $(\,\cdot\,)^*$ for a better readability. The first summand is

$$\begin{aligned} U_L^T \Gamma A_+ U_L &= \frac{1}{2}\begin{pmatrix} S_L \\ Y_L \end{pmatrix}^T \begin{pmatrix} \mathbf{F}^{1/2}\Psi^{1/2}\mathbf{F}^{1/2} & -\mathbf{I}_{6,6} \\ -\mathbf{I}_{6,6} & \mathbf{F}^{-1/2}\Psi^{-1/2}\mathbf{F}^{-1/2} \end{pmatrix} \begin{pmatrix} S_L \\ Y_L \end{pmatrix} \\ &= \frac{1}{2}\left(S_L^T \mathbf{F}^{1/2}\Psi^{1/2}\mathbf{F}^{1/2} S_L + Y_L^T \mathbf{F}^{-1/2}\Psi^{-1/2}\mathbf{F}^{-1/2} Y_L - 2S_L^T Y_L\right). \end{aligned}$$

The second summand can be written as

$$\begin{aligned} U_L^T \Gamma A_- U_R &= \frac{1}{2}\begin{pmatrix} S_L \\ Y_L \end{pmatrix}^T \begin{pmatrix} -\mathbf{F}^{1/2}\Psi^{1/2}\mathbf{F}^{1/2} & -\mathbf{I}_{6,6} \\ -\mathbf{I}_{6,6} & -\mathbf{F}^{-1/2}\Psi^{-1/2}\mathbf{F}^{-1/2} \end{pmatrix} \begin{pmatrix} S_R \\ Y_R \end{pmatrix} \\ &= \frac{1}{2}\big(- S_L^T \mathbf{F}^{1/2}\Psi^{1/2}\mathbf{F}^{1/2} S_R - Y_L^T S_R - Y_R^T S_L - Y_L^T \mathbf{F}^{-1/2}\Psi^{-1/2}\mathbf{F}^{-1/2} Y_R\big) \end{aligned}$$

and applying the discrete boundary conditions this time for U_R respectively S_R and Y_R, namely

$$\begin{pmatrix} S_R \\ Y_R \end{pmatrix} = \begin{pmatrix} \mathbf{0}_3 \\ -\mathbf{F}^{1/2}\Psi^{1/2}\mathbf{F}^{1/2} S_L + Y_L \end{pmatrix}$$

yields

$$\begin{aligned} U_L^T \Gamma A_- U_R &= \frac{1}{2}\big(- S_L^T \mathbf{F}^{1/2}\Psi^{1/2}\mathbf{F}^{1/2}\mathbf{0}_3 - Y_L^T \mathbf{0}_3 + S_L^T \mathbf{F}^{1/2}\Psi^{1/2}\mathbf{F}^{1/2} S_L - Y_L^T S_L \\ &\qquad + Y_L^T S_L - Y_L^T \mathbf{F}^{-1/2}\Psi^{-1/2}\mathbf{F}^{-1/2} Y_L\big) \\ &= \frac{1}{2}\big(S_L^T \mathbf{F}^{1/2}\Psi^{1/2}\mathbf{F}^{1/2} S_L - Y_L^T \mathbf{F}^{-1/2}\Psi^{-1/2}\mathbf{F}^{-1/2} Y_L\big). \end{aligned}$$

The third and last summand is

$$-\frac{1}{2}U_L^T \Pi U_L = \frac{1}{2}\begin{pmatrix} S_L \\ Y_L \end{pmatrix}^T \begin{pmatrix} \mathbf{0}_{6,6} & \mathbf{I}_{6,6} \\ \mathbf{I}_{6,6} & \mathbf{0}_{6,6} \end{pmatrix} \begin{pmatrix} S_L \\ Y_L \end{pmatrix} = Y_L^T S_L.$$

Finally, by putting all three summands together again, we have

$$\begin{aligned} b_R = \quad & \frac{1}{2}\big(S_L^T \mathbf{F}^{1/2}\Psi^{1/2}\mathbf{F}^{1/2} S_L + Y_L^T \mathbf{F}^{-1/2}\Psi^{-1/2}\mathbf{F}^{-1/2} Y_L - 2S_L^T Y_L\big) \\ & + \frac{1}{2}\big(S_L^T \mathbf{F}^{1/2}\Psi^{1/2}\mathbf{F}^{1/2} S_L - Y_L^T \mathbf{F}^{-1/2}\Psi^{-1/2}\mathbf{F}^{-1/2} Y_L\big) + Y_L^T S_L \\ = \quad & S_L^T \mathbf{F}^{1/2}\Psi^{1/2}\mathbf{F}^{1/2} S_L. \end{aligned}$$

A.2 Computation of the Matrix $\Gamma|A|$

First, note that by definition, it holds

$$\begin{aligned} |A| = T|\mathbf{\Lambda}|T^{-1} &= \frac{1}{2}\left(2T|\mathbf{\Lambda}|T^{-1} + T\mathbf{\Lambda}T^{-1} - T\mathbf{\Lambda}T^{-1}\right) \\ &= \frac{1}{2}(A + |A|) - \frac{1}{2}(A - |A|) \\ &= A_+ - A_-, \end{aligned}$$

meaning that

$$\Gamma|A| = \Gamma A_+ - \Gamma A_-.$$

The matrices on the right hand side have been determined in Appendix A.1, so that the above expression can be calculated explicitly by

$$\begin{aligned} \Gamma|A| &= \frac{1}{2}\begin{pmatrix} \mathbf{F}^{1/2}\Psi^{1/2}\mathbf{F}^{1/2} & -\mathbf{I}_{6,6} \\ -\mathbf{I}_{6,6} & \mathbf{F}^{-1/2}\Psi^{-1/2}\mathbf{F}^{-1/2} \end{pmatrix} - \frac{1}{2}\begin{pmatrix} -\mathbf{F}^{1/2}\Psi^{1/2}\mathbf{F}^{1/2} & -\mathbf{I}_{6,6} \\ -\mathbf{I}_{6,6} & -\mathbf{F}^{-1/2}\Psi^{-1/2}\mathbf{F}^{-1/2} \end{pmatrix} \\ &= \begin{pmatrix} \mathbf{F}^{1/2}\Psi^{1/2}\mathbf{F}^{1/2} & \mathbf{0}_{6,6} \\ \mathbf{0}_{6,6} & \mathbf{F}^{-1/2}\Psi^{-1/2}\mathbf{F}^{-1/2} \end{pmatrix}. \end{aligned}$$

The matrices $\mathbf{F}^{\pm 1/2}, \Psi^{\pm 1/2}$ are positive definite and therefore also the matrices $\mathbf{F}^{1/2}\Psi^{1/2}\mathbf{F}^{1/2}$ and $\mathbf{F}^{-1/2}\Psi^{-1/2}\mathbf{F}^{-1/2}$, because for any non-zero $z \in \mathbb{R}^{6\times 6}$ we have

$$z^T \mathbf{F}^{\pm 1/2}\Psi^{\pm 1/2}\mathbf{F}^{\pm 1/2} z = \left(\mathbf{F}^{\pm 1/2} z\right)^T \Psi^{\pm 1/2}\left(\mathbf{F}^{\pm 1/2} z\right) > 0.$$

Therefore also $\Gamma|A|$ is positive definite.

A.3 Convergence Tables

N_c	$\mathbb{L}^2$-error					
	(m = 1)	(m = 2)	(m = 3)	(m = 4)	(m = 5)	(m = 6)
8	$4.73 \cdot 10^{-7}$	$3.03 \cdot 10^{-7}$	$5.93 \cdot 10^{-7}$	$7.19 \cdot 10^{-7}$	$5.09 \cdot 10^{-7}$	$5.87 \cdot 10^{-7}$
16	$3.66 \cdot 10^{-8}$	$2.50 \cdot 10^{-8}$	$3.77 \cdot 10^{-8}$	$4.61 \cdot 10^{-8}$	$4.94 \cdot 10^{-8}$	$2.69 \cdot 10^{-8}$
32	$2.44 \cdot 10^{-9}$	$1.81 \cdot 10^{-9}$	$2.83 \cdot 10^{-9}$	$3.21 \cdot 10^{-9}$	$3.12 \cdot 10^{-9}$	$2.06 \cdot 10^{-9}$
64	$1.45 \cdot 10^{-10}$	$1.22 \cdot 10^{-10}$	$1.86 \cdot 10^{-10}$	$2.03 \cdot 10^{-10}$	$1.93 \cdot 10^{-10}$	$1.35 \cdot 10^{-10}$
128	$9.01 \cdot 10^{-12}$	$8.07 \cdot 10^{-12}$	$1.17 \cdot 10^{-11}$	$1.29 \cdot 10^{-11}$	$1.22 \cdot 10^{-11}$	$8.63 \cdot 10^{-12}$

Table A.1: First six components of the $\mathbb{L}^2$-error, computed by the `Trixi.jl` convergence test for the numerical solution of (103) with $N_p = 3, \mathrm{CFL} = 0.1$ and the upwind numerical flux, compared to the exact solution (102).

N_c	$\mathbb{L}^2$-error					
	(m = 1)	(m = 2)	(m = 3)	(m = 4)	(m = 5)	(m = 6)
8	$7.50 \cdot 10^{-7}$	$9.25 \cdot 10^{-7}$	$1.20 \cdot 10^{-6}$	$1.61 \cdot 10^{-6}$	$4.77 \cdot 10^{-7}$	$5.53 \cdot 10^{-7}$
16	$7.26 \cdot 10^{-8}$	$1.09 \cdot 10^{-7}$	$1.32 \cdot 10^{-7}$	$1.99 \cdot 10^{-7}$	$4.04 \cdot 10^{-8}$	$4.17 \cdot 10^{-8}$
32	$8.44 \cdot 10^{-9}$	$1.35 \cdot 10^{-8}$	$1.61 \cdot 10^{-8}$	$2.47 \cdot 10^{-8}$	$4.24 \cdot 10^{-9}$	$4.20 \cdot 10^{-9}$
64	$1.04 \cdot 10^{-9}$	$1.68 \cdot 10^{-9}$	$2.01 \cdot 10^{-9}$	$3.09 \cdot 10^{-9}$	$5.04 \cdot 10^{-10}$	$4.93 \cdot 10^{-10}$
128	$1.29 \cdot 10^{-10}$	$2.1 \cdot 10^{-10}$	$2.50 \cdot 10^{-10}$	$3.86 \cdot 10^{-10}$	$6.22 \cdot 10^{-11}$	$6.06 \cdot 10^{-11}$

Table A.2: First six components of the $\mathbb{L}^2$-error, computed by the `Trixi.jl` convergence test for the numerical solution of (103) with $N_p = 3, \mathrm{CFL} = 0.1$ and the central numerical flux, compared to the exact solution (102).

N_c	$\mathbb{L}^2$-error					
	(m = 7)	(m = 8)	(m = 9)	(m = 10)	(m = 11)	(m = 12)
8	$6.05 \cdot 10^{-7}$	$5.30 \cdot 10^{-7}$	$4.53 \cdot 10^{-7}$	$4.50 \cdot 10^{-7}$	$3.88 \cdot 10^{-7}$	$4.51 \cdot 10^{-7}$
16	$5.07 \cdot 10^{-8}$	$3.88 \cdot 10^{-8}$	$3.88 \cdot 10^{-8}$	$4.41 \cdot 10^{-8}$	$1.98 \cdot 10^{-8}$	$4.41 \cdot 10^{-8}$
32	$3.17 \cdot 10^{-9}$	$2.97 \cdot 10^{-9}$	$2.51 \cdot 10^{-9}$	$2.68 \cdot 10^{-9}$	$1.57 \cdot 10^{-9}$	$3.19 \cdot 10^{-9}$
64	$1.90 \cdot 10^{-10}$	$2.05 \cdot 10^{-10}$	$1.7 \cdot 10^{-10}$	$1.48 \cdot 10^{-10}$	$1.16 \cdot 10^{-10}$	$2.11 \cdot 10^{-10}$
128	$1.17 \cdot 10^{-11}$	$1.35 \cdot 10^{-11}$	$1.10 \cdot 10^{-11}$	$8.80 \cdot 10^{-12}$	$8.04 \cdot 10^{-12}$	$1.34 \cdot 10^{-11}$

Table A.3: Last six components of the $\mathbb{L}^2$-error, computed by the `Trixi.jl` convergence test for the numerical solution of (103) with $N_p = 3, \mathrm{CFL} = 0.1$ and the upwind numerical flux, compared to the exact solution (102).

N_c	$\mathbb{L}^2$-error					
	(m = 7)	(m = 8)	(m = 9)	(m = 10)	(m = 11)	(m = 12)
8	$8.86 \cdot 10^{-7}$	$1.03 \cdot 10^{-6}$	$9.78 \cdot 10^{-7}$	$4.69 \cdot 10^{-7}$	$4.45 \cdot 10^{-7}$	$3.95 \cdot 10^{-7}$
16	$1.03 \cdot 10^{-7}$	$1.20 \cdot 10^{-7}$	$1.16 \cdot 10^{-7}$	$4.99 \cdot 10^{-8}$	$3.66 \cdot 10^{-8}$	$3.50 \cdot 10^{-8}$
32	$1.25 \cdot 10^{-8}$	$1.47 \cdot 10^{-8}$	$1.42 \cdot 10^{-8}$	$5.90 \cdot 10^{-9}$	$3.75 \cdot 10^{-9}$	$3.74 \cdot 10^{-9}$
64	$1.55 \cdot 10^{-9}$	$1.84 \cdot 10^{-9}$	$1.76 \cdot 10^{-9}$	$7.26 \cdot 10^{-10}$	$4.38 \cdot 10^{-10}$	$4.45 \cdot 10^{-10}$
128	$1.93 \cdot 10^{-10}$	$2.29 \cdot 10^{-10}$	$2.2 \cdot 10^{-10}$	$9.04 \cdot 10^{-11}$	$5.38 \cdot 10^{-11}$	$5.49 \cdot 10^{-11}$

Table A.4: Last six components of the $\mathbb{L}^2$-error, computed by the `Trixi.jl` convergence test for the numerical solution of (103) with $N_p = 3, \mathrm{CFL} = 0.1$ and the central numerical flux, compared to the exact solution (102).

N_c	Num. flux	EOC					
		(m = 1)	(m = 2)	(m = 3)	(m = 4)	(m = 5)	(m = 6)
8	upwind	-	-	-	-	-	-
	central	-	-	-	-	-	-
16	upwind	3.69	3.60	3.97	3.96	3.36	4.45
	central	3.37	3.08	3.18	3.02	3.56	3.73
32	upwind	3.91	3.79	3.74	3.84	3.98	3.71
	central	3.10	3.02	3.04	3.01	3.25	3.31
64	upwind	4.07	3.89	3.93	3.98	4.01	3.93
	central	3.03	3.01	3.01	3.00	3.07	3.09
128	upwind	4.01	3.91	3.98	3.98	3.98	3.96
	central	3.01	3.00	3.00	3.00	3.02	3.02

Table A.5: First six components of the EOC, corresponding to the $\mathbb{L}^2$-errors in tables A.1 and A.2 for $N_p = 3, \mathrm{CFL} = 0.1$.

N_c	Num. flux	EOC					
		(m = 7)	(m = 8)	(m = 9)	(m = 10)	(m = 11)	(m = 12)
8	upwind	-	-	-	-	-	-
	central	-	-	-	-	-	-
16	upwind	3.58	3.77	3.54	3.35	4.29	3.35
	central	3.11	3.10	3.08	3.23	3.60	3.50
32	upwind	4.00	3.71	3.95	4.04	3.66	3.79
	central	3.04	3.02	3.03	3.08	3.29	3.22
64	upwind	4.06	3.86	3.89	4.17	3.76	3.92
	central	3.01	3.01	3.01	3.02	3.10	3.07
128	upwind	4.02	3.93	3.94	4.07	3.85	3.97
	central	3.00	3.00	3.00	3.01	3.03	3.02

Table A.6: Last six components of the EOC, corresponding to the $\mathbb{L}^2$-errors in tables A.3 and A.4 for $N_p = 3, \mathrm{CFL} = 0.1$.

N_c	$\mathbb{L}^2$-error					
	(m = 1)	(m = 2)	(m = 3)	(m = 4)	(m = 5)	(m = 6)
8	$4.12 \cdot 10^{-9}$	$4.62 \cdot 10^{-9}$	$4.71 \cdot 10^{-9}$	$6.20 \cdot 10^{-9}$	$5.65 \cdot 10^{-9}$	$4.39 \cdot 10^{-9}$
16	$1.48 \cdot 10^{-10}$	$1.45 \cdot 10^{-10}$	$1.86 \cdot 10^{-10}$	$2.37 \cdot 10^{-10}$	$2.09 \cdot 10^{-10}$	$1.39 \cdot 10^{-10}$
32	$4.84 \cdot 10^{-12}$	$4.74 \cdot 10^{-12}$	$6.35 \cdot 10^{-12}$	$7.60 \cdot 10^{-12}$	$6.75 \cdot 10^{-12}$	$4.99 \cdot 10^{-12}$
64	$1.64 \cdot 10^{-13}$	$1.54 \cdot 10^{-13}$	$2.12 \cdot 10^{-13}$	$2.45 \cdot 10^{-13}$	$2.24 \cdot 10^{-13}$	$1.70 \cdot 10^{-13}$
128	$8.92 \cdot 10^{-14}$	$7.59 \cdot 10^{-14}$	$7.99 \cdot 10^{-14}$	$1.25 \cdot 10^{-13}$	$1.09 \cdot 10^{-13}$	$9.64 \cdot 10^{-14}$

Table A.7: First six components of the $\mathbb{L}^2$-error, computed by the `Trixi.jl` convergence test for the numerical solution of (103) with $N_p = 4, \mathrm{CFL} = 0.1$ and the upwind numerical flux, compared to the exact solution (102).

N_c	$\mathbb{L}^2$-error					
	(m = 1)	(m = 2)	(m = 3)	(m = 4)	(m = 5)	(m = 6)
8	$5.78 \cdot 10^{-9}$	$3.71 \cdot 10^{-9}$	$6.23 \cdot 10^{-9}$	$5.89 \cdot 10^{-9}$	$4.80 \cdot 10^{-9}$	$5.28 \cdot 10^{-9}$
16	$1.66 \cdot 10^{-10}$	$1.13 \cdot 10^{-10}$	$1.76 \cdot 10^{-10}$	$1.60 \cdot 10^{-10}$	$1.34 \cdot 10^{-10}$	$1.49 \cdot 10^{-10}$
32	$4.01 \cdot 10^{-12}$	$3.97 \cdot 10^{-12}$	$4.16 \cdot 10^{-12}$	$4.16 \cdot 10^{-12}$	$3.94 \cdot 10^{-12}$	$4.19 \cdot 10^{-12}$
64	$1.35 \cdot 10^{-13}$	$1.31 \cdot 10^{-13}$	$1.36 \cdot 10^{-13}$	$1.45 \cdot 10^{-13}$	$1.36 \cdot 10^{-13}$	$1.42 \cdot 10^{-13}$
128	$8.16 \cdot 10^{-14}$	$7.14 \cdot 10^{-14}$	$7.51 \cdot 10^{-14}$	$1.19 \cdot 10^{-13}$	$1.03 \cdot 10^{-13}$	$9.36 \cdot 10^{-14}$

Table A.8: First six components of the $\mathbb{L}^2$-error, computed by the `Trixi.jl` convergence test for the numerical solution of (103) with $N_p = 4, \mathrm{CFL} = 0.1$ and the central numerical flux, compared to the exact solution (102).

N_c	$\mathbb{L}^2$-error					
	(m = 7)	(m = 8)	(m = 9)	(m = 10)	(m = 11)	(m = 12)
8	$5.70 \cdot 10^{-9}$	$5.54 \cdot 10^{-9}$	$5.16 \cdot 10^{-9}$	$5.16 \cdot 10^{-9}$	$2.95 \cdot 10^{-9}$	$5.01 \cdot 10^{-9}$
16	$2.04 \cdot 10^{-10}$	$2.12 \cdot 10^{-10}$	$1.85 \cdot 10^{-10}$	$1.77 \cdot 10^{-10}$	$1.14 \cdot 10^{-10}$	$2.21 \cdot 10^{-10}$
32	$6.42 \cdot 10^{-12}$	$7.57 \cdot 10^{-12}$	$6.28 \cdot 10^{-12}$	$5.18 \cdot 10^{-12}$	$4.31 \cdot 10^{-12}$	$7.80 \cdot 10^{-12}$
64	$2.15 \cdot 10^{-13}$	$2.54 \cdot 10^{-13}$	$2.07 \cdot 10^{-13}$	$1.61 \cdot 10^{-13}$	$1.48 \cdot 10^{-13}$	$2.51 \cdot 10^{-13}$
128	$1.02 \cdot 10^{-13}$	$1.16 \cdot 10^{-13}$	$7.34 \cdot 10^{-14}$	$1.12 \cdot 10^{-13}$	$5.86 \cdot 10^{-14}$	$5.45 \cdot 10^{-14}$

Table A.9: Last six components of the $\mathbb{L}^2$-error, computed by the `Trixi.jl` convergence test for the numerical solution of (103) with $N_p = 4, \mathrm{CFL} = 0.1$ and the upwind numerical flux, compared to the exact solution (102).

N_c	$\mathbb{L}^2$-error					
	(m = 7)	(m = 8)	(m = 9)	(m = 10)	(m = 11)	(m = 12)
8	$5.98 \cdot 10^{-9}$	$4.49 \cdot 10^{-9}$	$6.07 \cdot 10^{-9}$	$4.31 \cdot 10^{-9}$	$4.15 \cdot 10^{-9}$	$4.07 \cdot 10^{-9}$
16	$1.64 \cdot 10^{-10}$	$1.35 \cdot 10^{-10}$	$1.52 \cdot 10^{-10}$	$1.41 \cdot 10^{-10}$	$1.46 \cdot 10^{-10}$	$1.40 \cdot 10^{-10}$
32	$3.95 \cdot 10^{-12}$	$3.89 \cdot 10^{-12}$	$4.00 \cdot 10^{-12}$	$3.95 \cdot 10^{-12}$	$3.82 \cdot 10^{-12}$	$3.90 \cdot 10^{-12}$
64	$1.38 \cdot 10^{-13}$	$1.32 \cdot 10^{-13}$	$1.29 \cdot 10^{-13}$	$1.3 \cdot 10^{-13}$	$1.23 \cdot 10^{-13}$	$1.24 \cdot 10^{-13}$
128	$9.27 \cdot 10^{-14}$	$1.10 \cdot 10^{-13}$	$6.77 \cdot 10^{-14}$	$1.07 \cdot 10^{-13}$	$1.03 \cdot 10^{-13}$	$5.04 \cdot 10^{-14}$

Table A.10: Last six components of the $\mathbb{L}^2$-error, computed by the `Trixi.jl` convergence test for the numerical solution of (103) with $N_p = 4, \mathrm{CFL} = 0.1$ and the central numerical flux, compared to the exact solution (102).

N_c	Num. flux	EOC (m = 1)	(m = 2)	(m = 3)	(m = 4)	(m = 5)	(m = 6)
8	upwind	-	-	-	-	-	-
	central	-	-	-	-	-	-
16	upwind	4.80	5.00	4.66	4.71	4.75	4.98
	central	5.12	5.04	5.15	5.20	5.16	5.15
32	upwind	4.93	4.93	4.87	4.96	4.96	4.80
	central	5.37	4.83	5.40	5.27	5.09	5.15
64	upwind	4.88	4.95	4.91	4.96	4.91	4.88
	central	4.89	4.92	4.93	4.84	4.85	4.88
128	upwind	0.88	1.02	1.41	0.97	1.05	0.82
	central	0.73	0.88	0.86	0.28	0.40	0.60

Table A.11: First six components of the EOC, corresponding to the $\mathbb{L}^2$-errors intables A.7 and A.8 for $N_p = 4, \mathrm{CFL} = 0.1$.

N_c	Num. flux	EOC (m = 7)	(m = 8)	(m = 9)	(m = 10)	(m = 11)	(m = 12)
8	upwind	-	-	-	-	-	-
	central	-	-	-	-	-	-
16	upwind	4.81	4.71	4.81	4.86	4.70	4.50
	central	5.19	5.05	5.32	4.94	4.83	4.86
32	upwind	4.99	4.81	4.88	5.10	4.73	4.82
	central	5.37	5.12	5.24	5.15	5.25	5.17
64	upwind	4.90	4.99	4.92	5.01	4.86	4.96
	central	4.84	4.88	4.95	4.92	4.95	4.98
128	upwind	1.08	1.13	1.50	0.52	1.34	2.20
	central	0.57	0.26	0.94	0.28	1.22	1.30

Table A.12: Last six components of the EOC, corresponding to the $\mathbb{L}^2$-errors in tables A.9 and A.10 for $N_p = 4, \mathrm{CFL} = 0.1$.

List of Figures

List of Tables

Nomenclature

General notations

$\mathbf{0}_{i,j}$	zero in $\mathbb{R}^{i\times j}$
$\mathbf{0}_i$	zero in $\mathbb{R}^i$
$\mathbf{I}_{i,j}$	identity in $\mathbb{R}^{i\times j}$
$\tilde{z}$	cross product matrix of a vector $z \in \mathbb{R}^3$

Frequently used variables

$\Theta = \Theta(x,t) \in \mathbb{R}^3$	internal forces
$\Xi = \Xi(x,t) \in \mathbb{R}^3$	internal moments
$V = V(x,t) \in \mathbb{R}^3$	linear velocities
$\Omega = \Omega(x,t) \in \mathbb{R}^3$	angular velocities
$k = k(x) \in \mathbb{R}^3$	initial curvature
$f_{ext} = f_{ext}(x,t) \in \mathbb{R}^3$	external forces
$m_{ext} = m_{ext}(x,t) \in \mathbb{R}^3$	external moments
$\mathbf{F} = \mathbf{F}(x) \in \mathbb{R}^{6\times 6}$	flexibility matrix, positive definite
$\mathbf{M} = \mathbf{M}(x) \in \mathbb{R}^{6\times 6}$	mass matrix, positive definite
$A = A(x) \in \mathbb{R}^{12\times 12}$	advection matrix for intrinsic beam balance law
$\Psi = \Psi(x) \in \mathbb{R}^{6\times 6}$	positive definite matrix
$\Lambda = \Lambda(x) \in \mathbb{R}^{6\times 6}$	positive definite diagonal matrix with square roots of eigenvalues of Ψ as entries
$\mathbf{\Lambda} = \mathbf{\Lambda}(x) \in \mathbb{R}^{12\times 12}$	diagonal matrix, consisting of $\pm\Lambda$ on diagonal
$\mathcal{X} = \mathcal{X}(x) \in \mathbb{R}^{6\times 6}$	transformation matrix that diagonalizes Ψ
$T = T(x) \in \mathbb{R}^{12\times 12}$	transformation matrix that diagonalizes A
$\Gamma = \Gamma(x) \in \mathbb{R}^{12\times 12}$	positive definite coefficient matrix in capacity form
$\Pi \in \mathbb{R}^{12\times 12}$	constant, symmetric coefficient matrix in capacity form
$u = u(x,t) \in \mathbb{R}^{12}$	state variable of intrinsic beam equation in linear advection form, physical variables
$w = w(x,t) \in \mathbb{R}^{12}$	characteristic variables
$Q_{cap} = Q_{cap}(u,x,t) \in \mathbb{R}^{12}$	source term for capacity form

References

[1] M. R. Amoozgar, H. Shahverdi, and A. S. Nobari. Aeroelastic stability of hingeless rotor blades in hover using fully intrinsic equations. *AIAA Journal*, 55(7):2450–2460, 2017.

[2] M. Artola, A. Wynn, and R. Palacios. A nonlinear modal-based framework for low computational cost optimal control of 3d very flexible structures. *18th European Control Conference (ECC)*, 2019.

[3] M. Artola, A. Wynn, and R. Palacios. Generalized kelvin–voigt damping for geometrically nonlinear beams. *AIAA Journal*, 59(1):356–365, 2021.

[4] R. Ballarini. The da vinci-euler-bernoulli beam theory? *Mechanical Engineering Magazine Online*, 2006.

[5] O. A. Bauchau, P. Betsch, A. Cardona, J. Gerstmayr, B. Jonker, P. Masarati, and V. Sonneville. Validation of flexible multibody dynamics beam formulations using benchmark problems. *Multibody System Dynamics*, 37(1):29–48, mar 2016.

[6] O. A. Bauchau and J. I. Craig. Euler-bernoulli beam theory. In *Structural Analysis*, pages 173–221. Springer Netherlands, 2009.

[7] O. A. Bauchau and S. Han. Advanced beam theory for multibody dynamics. *ASME 2013 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 2013.

[8] A. D. Beck, G. J. Gassner, and C.-D. Munz. On the effect of flux functions in discontinuous galerkin simulations of underresolved turbulence. In *Lecture Notes in Computational Science and Engineering*, pages 145–155. Springer International Publishing, 2013.

[9] M. Borri, G. L. Ghiringhelli, and T. Merlini. Linear analysis of naturally curved and twisted anisotropic beams. *Composites Engineering*, 2(5-7):433–456, 1992.

[10] M. H. Carpenter and C. A. Kennedy. Fourth-order 2n-storage runge-kutta schemes. *NASA Report TM 109112, NASA Langley Research Center*, 1994.

[11] Y. Cheng and C.-W. Shu. A discontinuous galerkin finite element method for directly solving the hamilton–jacobi equations. *Journal of Computational Physics*, 223(1):398–415, 2007.

[12] R. Courant, K. Friedrichs, and H. Lewy. Über die partiellen differenzengleichungen der mathematischen physik. *Mathematische Annalen*, 100(1):32–74, 1928.

[13] I. Elishakoff. Who developed the so-called timoshenko beam theory? *Mathematics and Mechanics of Solids*, 25(1):97–116, 2019.

[14] G. J. Gassner. A skew-symmetric discontinuous galerkin spectral element discretization and its relation to SBP-SAT finite difference methods. *SIAM Journal on Scientific Computing*, 35(3):A1233–A1253, 2013.

[15] V. Giurgiutiu. *Structural Health Monitoring of Aerospace Composites*. Elsevier, 2016.

[16] S. K. Godunov and I. Bohachevsky. Finite difference method for numerical computation of discontinuous solutions of the equations of fluid dynamics. *Matematičeskij sbornik*, 47(89)(3):271–306, 1959.

[17] H. Hesse and R. Palacios. Consistent structural linearisation in flexible-body dynamics with large rigid-body motion. *Computers & Structures*, 110-111:1–14, 2012.

[18] J. S. Hesthaven and T. Warburton. *Nodal Discontinuous Galerkin Methods*. Springer New York, 2008.

[19] D. H. Hodges. A mixed variational formulation based on exact intrinsic equations for dynamics of moving beams. *International Journal of Solids and Structures*, 26(11):1253–1273, 1990.

[20] D. H. Hodges. Geometrically exact, intrinsic theory for dynamics of curved and twisted anisotropic beams. *AIAA Journal*, 41(6):1131–1137, 2003.

[21] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

[22] Julia 1.8 documentation. `https://docs.julialang.org/en/v1/`. Accessed: 2022-11-10.

[23] M. J. Jweeg, M. Al-Waily, and K. K. Resan. *Energy Methods and Finite Element Techniques*. Elsevier, 2022.

[24] G. Karniadakis and S. Sherwin. *Spectral/hp Element Methods for Computational Fluid Dynamics*. Oxford University Press, 2005.

[25] D. A. Kopriva and G. J. Gassner. On the quadrature and weak form choices in collocation type discontinuous galerkin spectral element methods. *Journal of Scientific Computing*, 44(2):136–155, 2010.

[26] D. A. Kopriva and G. J. Gassner. An energy stable discontinuous galerkin spectral element discretization for variable coefficient advection problems. *SIAM Journal on Scientific Computing*, 36(4):A2076–A2099, 2014.

[27] D. A. Kopriva and G. J. Gassner. A split-form, stable CG/DG-SEM for wave propagation modeled by linear hyperbolic systems. *Journal of Scientific Computing*, 89(1), 2021.

[28] D. A. Kopriva, G. J. Gassner, and J. Nordström. Stability of discontinuous galerkin spectral element schemes for wave propagation when the coefficient matrices have jumps, 2020.

[29] W. M. Lai, D. Rubin, and E. Krempl. *Introduction to Continuum Mechanics*. Elsevier, 3 edition, 2010.

[30] P. D. Lax. Weak solutions of nonlinear hyperbolic equations and their numerical computation. *Communications on Pure and Applied Mathematics*, 7(1):159–193, 1954.

[31] R. J. LeVeque. *Finite Volume Methods for Hyperbolic Problems*. Cambridge University Press, 2002.

[32] C. K. Manoli, S. Papatzani, and D. E. Mouzakis. Exploring the limits of euler–bernoulli theory in micromechanics. *Axioms*, 11(3):142, 2022.

[33] A. Muñoz-Simón, A. Wynn, and R. Palacios. Unsteady and three-dimensional aerodynamic effects on wind turbine rotor loads. In *AIAA Scitech 2020 Forum*. American Institute of Aeronautics and Astronautics, jan 2020.

[34] J. Nordström. A roadmap to well posed and stable problems in computational physics. *Journal of Scientific Computing*, 71(1):365–385, 2016.

[35] J. Nordström and M. Wahlsten. Variance reduction through robust design of boundary conditions for stochastic hyperbolic systems of equations. *Journal of Computational Physics*, 282:1–22, 2015.

[36] M. J. Patil and D. H. Hodges. Flight dynamics of highly flexible flying wings. *Journal of Aircraft*, 43(6):1790–1799, 2006.

[37] M. J. Patil and D. H. Hodges. Variable-order finite elements for nonlinear, fully intrinsic beam equations. *Journal of Mechanics of Materials and Structures*, 6(1-4):479–493, 2011.

[38] M. Petrovitsch. Sur une manière d'étendre le théorème de la moyence aux équations différentielles du premier ordre. *Annals of Mathematics*, 54(3):417–436, 1901.

[39] G. Polya. *How to Solve It; A New Aspect of Mathematical Method*. Princeton University Press, 1957.

[40] H. Ranocha, M. Schlottke-Lakemper, A. R. Winters, E. Faulhaber, J. Chan, and G. J. Gassner. Adaptive numerical simulations with Trixi.jl: A case study of Julia for scientific computing. *Proceedings of the JuliaCon Conferences*, 1(1):77, 2022.

[41] P. J. Roache. Code verification by the method of manufactured solutions. *Journal of Fluids Engineering*, 124(1):4–10, 2001.

[42] C. Rodriguez. *Control and stabilization of geometrically exact beams.* PhD thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg, 2022.

[43] C. Rodriguez. Networks of geometrically exact beams: Well-posedness and stabilization. *Mathematical Control and related fields*, 12(1):49, 2022.

[44] C. Rodriguez and G. Leugering. Boundary feedback stabilization for the intrinsic geometrically exact beam model. *SIAM Journal on Control and Optimization*, 58(6):3533–3558, 2020.

[45] D. L. Russell. Controllability and stabilizability theory for linear partial differential equations: Recent progress and open questions. *SIAM Review*, 20(4):639–739, 1978.

[46] M. Schlottke-Lakemper, G. J. Gassner, H. Ranocha, A. R. Winters, and J. Chan. Trixi.jl: Adaptive high-order numerical simulations of hyperbolic PDEs in Julia. `https://github.com/trixi-framework/Trixi.jl`, 2021.

[47] M. Schlottke-Lakemper, A. R. Winters, H. Ranocha, and G. J. Gassner. A purely hyperbolic discontinuous Galerkin approach for self-gravitating gas dynamics. *Journal of Computational Physics*, 442:110467, 2021.

[48] J. C. Simo and L. Vu-Quoc. On the dynamics of flexible beams under large overall motions—the plane case: Part II. *Journal of Applied Mechanics*, 53(4):855–863, 1986.

[49] Z. Sotoudeh and D. H. Hodges. Modeling beams with various boundary conditions using fully intrinsic equations. *Journal of Applied Mechanics*, 78(3), 2011.

[50] E. F. Toro. *Riemann Solvers and Numerical Methods for Fluid Dynamics.* Springer Berlin Heidelberg, 2009.

[51] Trixi.jl documentation. `https://trixi-framework.github.io/Trixi.jl/stable/`. Accessed: 2022-11-10.

[52] VAST webpage. `https://www.dlr.de/sc/desktopdefault.aspx/tabid-12766/22301_read-51581/`. Accessed: 2022-11-10.

[53] L. Wang, X. Liu, N. Renevier, M. Stables, and G. M. Hall. Nonlinear aeroelastic modelling for wind turbine blades based on blade element momentum theory and geometrically exact beam theory. *Energy*, 76:487–501, 2014.

[54] Y. Xing and C.-W. Shu. High order well-balanced finite volume WENO schemes and discontinuous galerkin methods for a class of hyperbolic systems with source terms. *Journal of Computational Physics*, 214(2):567–598, 2006.