VirtualBow v0.9.0

Theory Manual

Contents

Chapter 1

Introduction

This is the theory manual for VirtualBow, an open-source software tool for simulating bow and arrow physics. For the latest version of the software and this document visit

http://www.virtualbow.org/.

This documentation is about the theoretical foundations and technical details of the software. It is still a work in progress and is meant as a reference for developers and interested users.

Chapter 2

The Bow Model

A scientific model is a simplification and abstraction of reality, often formulated in a mathematical way. A good model reduces the complexity of a real system by including only its most significant aspects and disregarding less significant ones. Analyzing this simplified model can then lead to conclusions about the real system that wouldn't have been possible otherwise.

What aspects of reality a model has to reflect depends on the kinds of questions it seeks to answer. The first step for developing a bow model is therefore to clarify its scope and intended application.

2.1 Scope of the Model

In the case of VirtualBow the purpose of the bow model is to give users the ability to evaluate the bow designs they come up with. So what characteristics make a bow design a good one and what are their implications for the bow model? Many things come to mind, but we limit ourselves to the following three categories:

1. **Viability**: Can the design be realised? Do the materials withstand the arising loads when the bow is being used? To be able to judge this the model has to accurately predict the distribution of stress across the limbs as well as the tensile forces in the string that arise when drawing and shooting the bow.
2. **Performance**: Important performance characteristics are the arrow's velocity, kinetic energy and the bow's degree of efficiency, i.e. the percentage of the stored elastic energy that is converted into kinetic energy of the arrow during the shot. To capture these things the model has to reflect all major mechanisms of efficiency loss in a bow. We will later see what those are when we look at scientific research.
3. **Comfort**: Viability and raw performance aren't everything. A bow design will also be judged by it's influence on the archer, like the draw force, the shape of the draw

curve (no "stacking", i.e. increasing stiffness towards the end of draw) and the dynamic forces exerted to the shooter ("hand shock").

Any physical effects in archery that don't (significantly) affect one of those categories above are outside the scope of VirtualBow and the bow model doesn't need to include them. Exterior ballistics is an example for this, another one is the bending motion of the arrow (Archer's Paradox).

Now the next question is: How does a bow model have to look like in order to capture the things listed above? That's not easy to answer, but luckily there has been done quite some scientific research about bow and arrow physics in the past, so we don't have to start completely from scratch. The next section therefore examines some of that research before we define our own model.

2.2 Literature Review

Research about the mechanics of bow and arrow goes back to at least the 1930s, when C.N. Hickman, P.E. Klopstek and others performed systematic experiments and developed simple mathematical models. Without modern computers available these early models were still limited in their complexity by what could be solved analytically. The focus was therefore more on qualitative insight rather than accurate prediction. One of those early models is the *Hickman Model* [1], published in 1937. It assumes two rigid limbs that are connected to the stationary rest of the bow via elastic hinges as shown in figure 2.1. Even though this is a huge simplification, this model can already be used to investigate the efficiency of a bow, i.e. how it transfers its stored elastic energy to an arrow.

An important modification to the Hickman model was suggested by W.C. Marlow in 1980 [2]. It was already well-known that the original Hickman model with an inextensible string severely overpredicted a bow's degree of efficiency. (Predicted efficiencies typically over 90% while measurements suggest values between 70% and 85% for most bows.) Popular explanations for this discrepancy were effects like hysteresis and air resistance. Marlow however found another reason: With an inextensible string (as in Hickman's original model) the arrow exits the bow exactly at brace height, where the limbs have zero velocity. Only the kinetic energy of the string remains in the bow, the rest is transferred to the arrow. Marlow then showed that accounting for the elasticity of the string shifts the point of exit for the arrow away from the braced position, such that the limbs still have substantial kinetic energy left. In his numerical example the modification reduces the predicted efficiency to 78% as opposed to 92% previously. Of the energy remaining in the bow, 11% were the kinetic energy of the limbs, 9% the kinetic energy of the string and 2% the potential energy of limbs and string.

This leads to an important realization: The efficiency of a bow is mainly determined by the kinetic energy that stays in the system due to mass and elasticity of the string and the mass distribution of the limbs. The contribution of dissipative effects like air resistance

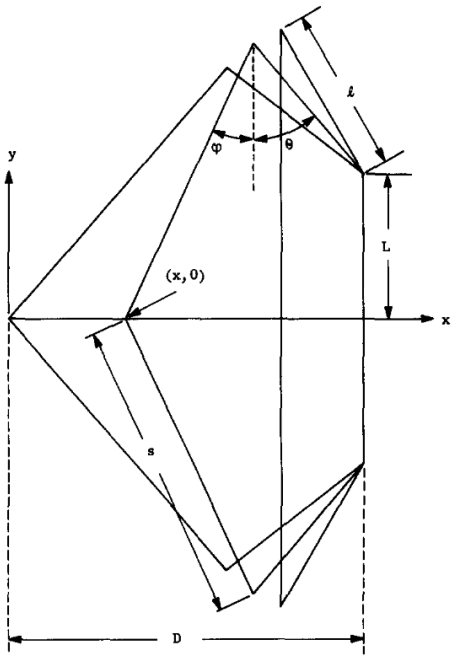

Figure 2.1: Hickman model as used by W.C. Marlow [2]

and hysteresis to the overall efficiency was estimated by Marlow to be less than 2% and 3% respectively.

The widespread availability of digital computers eventually made it possible to simulate much more detailed models and therefore make more accurate predictions. Much notable work in this area has been done by Dr. Bob Kooi and his co-authors. Over the course of the 1980s and 90s they published a good amount of papers about the mechanics of bow and arrow in which they utilize numerical methods to solve increasingly complex mathematical models.

The first one of those papers is titled *On the static deformation of a bow* [3] and investigates the storage of deformation energy in bows with and without recurve. Figure 2.2 shows the model used there. The bow is assumed to be symmetric and has an inextensible string. The limbs are represented by an Euler-Bernoulli beam, which is an infinitely thin, elastic line with a certain (varying) bending stiffness for which the Euler-Bernoulli assumptions hold. These are assumptions about the kinematics of the deformation, namely that the cross sections of the beam always stay flat and perpendicular to its centerline. This is typically a good approximation for slender beams, that is beams whose length is much larger than their cross section dimensions. Kooi uses the general version of the Euler-Bernoulli equation that accounts for arbitrarily large deformation. Combining this with geometric constraints for the string and some boundary conditions leads to the bow's equations of equilibrium, a free-boundary value problem for a set of ordinary differential equations. The equations of equilibrium are solved numerically for specific draw lengths of the bow.

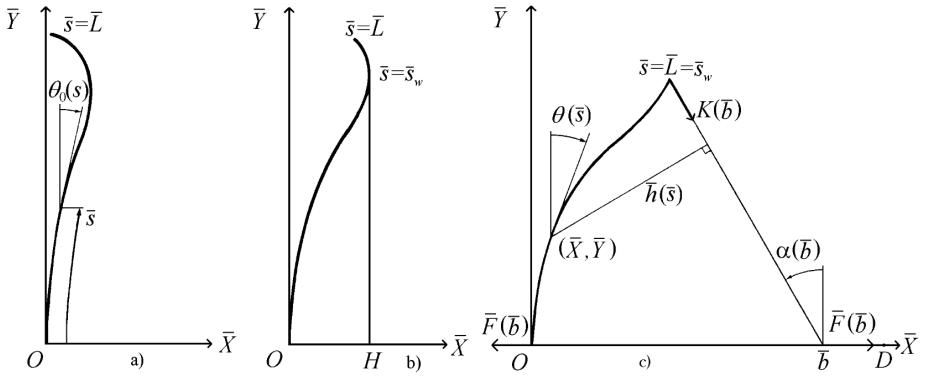

Figure 2.2: Bow model used by Bob Kooi [3]

In subsequent papers titled *On the mechanics of the bow and arrow* [4] and *On the mechanics of the modern working-recurve bow* [5] Kooi also considers the dynamic case. Like Marlow he uses a linear-elastic string and assumes the arrow as a concentrated point mass. The equations of motion for the system form a moving-boundary value problem for a set of partial differential equations. They are solved numerically by using a finite-difference method with complete discretization. (Both space and time are approximated by a discrete "grid" of values. The partial derivatives in the governing equations are then approximated by finite difference quotients operating on those grid values. The resulting large number of difference equations can then be solved with a computer program.)

In the paper *On the Mechanics of the Arrow: Archer's Paradox* [6], Kooi refines his model even more by including an elastic arrow with a sliding contact to the bow's grip. This makes it possible to analyze the Archer's Paradox: The phenomenon of arrows going around the grip of a bow by performing a bending motion in the horizontal plane. The dynamics of the arrow are however treated partly decoupled from the bow's dynamics: Only the influence of the bow on the arrow's motion is considered, the other way around is neglected. This is in line with our assumption from section 2.1 that the bending motion of the arrow doesn't contribute much to a bow's overall performance and is therefore out of scope for our model.

2.3 Model Definition

We have now learned enough in order to define our own bow model. First of all a few words on the general modeling approach. As we saw in the previous section, any decently complex bow model cannot be solved analytically but has to be combined with some sort of numerical method for calculating an approximate solution. Models based on continuum mechanics (e.g. Euler-Bernoulli beam theory) always have an infinite number of degrees of freedom and will eventually have to be approximated by a discrete system with finitely many degrees of freedom in order to be solvable by a computer program. This step is called discretization. Kooi for example, like a true mathematician, formulates

the complete equations of motion for his continuous bow model and only then starts to employ a discretization technique (in this case a finite difference method) to solve those equations. This results in a clear separation between the mathematical model and the numerical solution methods.

We're going to use more of an engineering approach here instead and start with a discrete system in the first place. We do this by constructing the continuous parts of the bow from a large number of so-called finite elements. A finite element is an approximation of a little part of a continuous structure, e.g. a beam or a bar, but with only finitely many degrees of freedom, namely the displacements of the nodes at which the elements are connected to each other. The more elements are used, the better the approximation gets and the closer the results will be to the exact solution.

The finite element method leads to a (potentially large) system of ordinary differential equations as the equations of motion, which can be solved by standard numerical methods as we will see later.

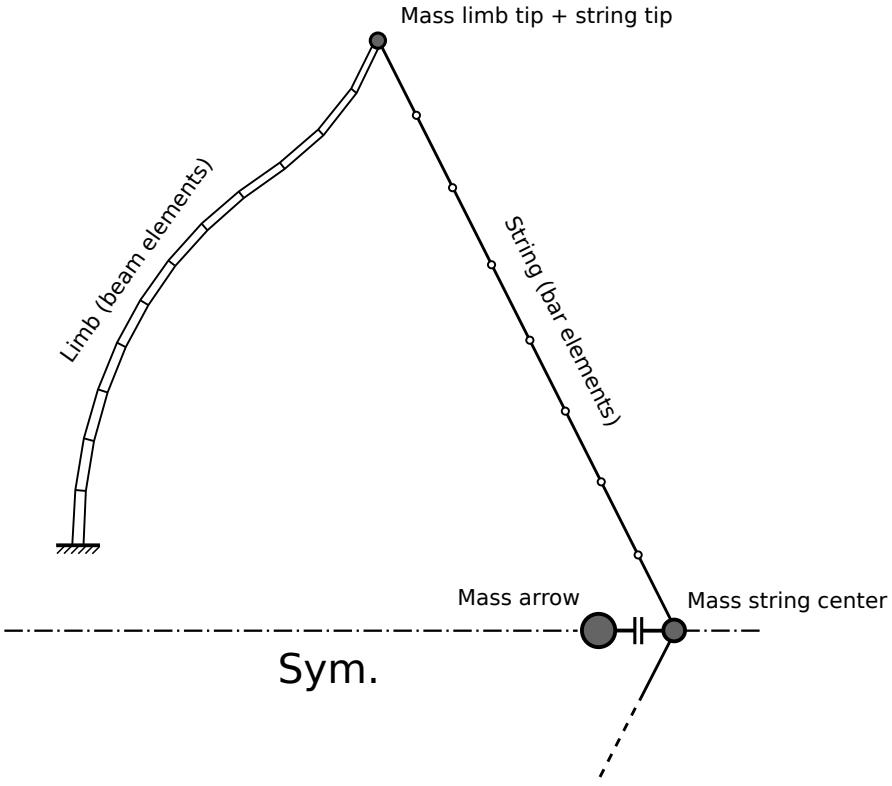

Figure 2.3: Finite element model of the bow

Figure 2.3 shows the finite element approximation of our bow model. Like Kooi's model it is symmetric with respect to the horizontal plane, assumes the limbs to satisfy the Euler-Bernoulli assumptions and neglects any dissipative effects like air resistance and hysteresis. The limbs are therefore made up of Euler-Bernoulli beam elements. A concentrated point

mass is placed at the tips to account for any added weight, like overlays for example.

The string, as opposed to the limbs, only transfers longitudinal forces and is therefore approximated by linear-elastic bar elements. We limit ourselves to linear elasticity for pragmatic reasons: Obtaining the exact nonlinear stress-strain curve of a particular string material is pretty difficult. Instead, any material nonlinearity (which can be the case especially for modern synthetic materials) will have to be accounted for by using an "average" stiffness. In the previous section we saw that the mass properties of the string are fairly important, so in addition to the mass of the string itself there are two additional point masses at the tips and center to account for things like servings, nocking point, etc.

The arrow is another point mass, as already justified earlier. It is connected to the string by an ideal, frictionless contact without any release force. This means that the arrow will be released as soon as the string doesn't push against/accelerate it anymore.

Not shown in figure 2.3 are the contact elements that prevent the string nodes from passing through the limb, thereby making it possible to simulate recurves. These are also ideal contacts without any friction.

All in all, there are four different types of elements needed:

- Beam Elements (Limbs)
- Bar Elements (String)
- Mass Elements (Arrow + additional masses)
- Contact Elements (String to limb contact)

All of them have to account for arbitrarily large displacements and are therefore geometrically nonlinear (even though all the materials are linear-elastic).

In the next chapters we will first show how the equations of motion for a finite element system can be assembled from the equations of motion for the single elements. We will then derive the equations of motion for the element types listed above. Finally we're going to discuss the numerical solution of those equations for the static and dynamic case and how this is implemented in VirtualBow.

Chapter 3

Equations of Motion

Consider a single, unconstrained finite element i with n degrees of freedom. Its configuration at time t will be described by the displacement vector $\boldsymbol{u}_i(t) \in \mathbb{R}^n$ containing all positions and rotation angles of the nodes. The equation of motion for the kind of elements we're concerned with takes the form

$$\boldsymbol{M}_i\, \ddot{\boldsymbol{u}}_i + \boldsymbol{q}_i(\boldsymbol{u}_i,\, \dot{\boldsymbol{u}}_i) = \boldsymbol{p}_i(t). \tag{3.1}$$

This is a nonlinear ordinary differential equation of second order for the displacements. $\boldsymbol{M}_i \in \mathbb{R}^{n\times n}$ is called the element's mass matrix, $\boldsymbol{q}_i \in \mathbb{R}^n$ is the vector of internal forces depending on the displacements and their associated velocity (e.g. elastic forces, damping forces, ...) and $\boldsymbol{p}_i \in \mathbb{R}^n$ is the vector of externally applied loads.

Now consider a finite element system that consists of a number of such elements that are connected to each other in a certain way. If the overall system has a total of N degrees of freedom, it's configuration can be described by the global displacement vector $\boldsymbol{u}(t) \in \mathbb{R}^N$. The local element displacements $\boldsymbol{u}_i$ are now no longer independent but instead directly related to the global displacements of the system they are part of. In our case these relationships can be written as

$$\boldsymbol{u}_i = \boldsymbol{T}_i\, \boldsymbol{u}, \tag{3.2}$$

where $\boldsymbol{T_i} \in \mathbb{R}^{n\times N}$ is a constant transformation matrix for the respective element. Using the local equations of motion (3.1), the kinematic relationship (3.2) and some analytical mechanics it can be shown [add citation] that the equation of motion of the overall system takes the similar form

$$\boldsymbol{M}\, \ddot{\boldsymbol{u}} + \boldsymbol{q}(\boldsymbol{u},\, \dot{\boldsymbol{u}}) = \boldsymbol{p}(t). \tag{3.3}$$

where the global mass matrix $\boldsymbol{M} \in \mathbb{R}^{N \times N}$, internal forces $\boldsymbol{q} \in \mathbb{R}^N$ and external forces $\boldsymbol{p} \in \mathbb{R}^N$ are calculated from the local element matrices as

$$\boldsymbol{M} = \sum_i \boldsymbol{T}_i^\mathsf{T} \, \boldsymbol{M}_i \, \boldsymbol{T}_i, \tag{3.4}$$

$$\boldsymbol{q} = \sum_i \boldsymbol{T}_i^\mathsf{T} \boldsymbol{q}_i(\boldsymbol{T}_i \, \boldsymbol{u}, \, \boldsymbol{T}_i \, \dot{\boldsymbol{u}}), \tag{3.5}$$

$$\boldsymbol{p} = \sum_i \boldsymbol{T}_i^\mathsf{T} \, \boldsymbol{p}_i. \tag{3.6}$$

In practice, because of the way the element displacements are related to the overall displacements, the transformation matrices $\boldsymbol{T}_i$ only contain ones and zeros, effectively describing a selection of entries. Therefore it is not actually necessary to carry out all the matrix multiplications occurring above. Instead, the entries of the local matrices can be directly added to the corresponding entries of the global matrices.

In order to solve the equations of motion for equilibrium we're later going to need the derivative of the internal forces with respect to the displacements, which is called the tangent stiffness matrix $\boldsymbol{K} \in \mathbb{R}^{N \times N}$ of the system,

$$\boldsymbol{K}(\boldsymbol{u}) = \frac{\partial \boldsymbol{q}}{\partial \boldsymbol{u}} \tag{3.7}$$

It can be calculated from the element's local tangent stiffness matrices $\boldsymbol{K}_i = \frac{\partial \boldsymbol{q}_i}{\partial \boldsymbol{u}_i}$ like this,

$$\begin{aligned} \boldsymbol{K}(\boldsymbol{u}) = \frac{\partial \boldsymbol{q}}{\partial \boldsymbol{u}} = \sum_i \boldsymbol{T}_i^\mathsf{T} \frac{\partial \boldsymbol{q}_i}{\partial \boldsymbol{u}} = \sum_i \boldsymbol{T}_i^\mathsf{T} \frac{\partial \boldsymbol{q}_i}{\partial \boldsymbol{u}_i} \frac{\partial \boldsymbol{u}_i}{\partial \boldsymbol{u}} = \sum_i \boldsymbol{T}_i^\mathsf{T} \frac{\partial \boldsymbol{q}_i}{\partial \boldsymbol{u}_i} \boldsymbol{T}_i \\ = \sum_i \boldsymbol{T}_i^\mathsf{T} \boldsymbol{K}_i \boldsymbol{T}_i. \end{aligned}$$

For now this is all we have to know about the equations of motion of the finite element system. What's going to be needed next are the local equations of motion for the different element types. Those are going to be derived in the next sections.

3.1 Mass Element

The point mass is the most simple element. Figure 3.1 shows its definition: A mass m is placed in a cartesian coordinate system. Its position is described by the displacements $u_0(t)$ and $u_1(t)$ along the coordinate axes. The external forces $p_0(t)$ and $p_1(t)$ are acting in the direction of the respective displacements.

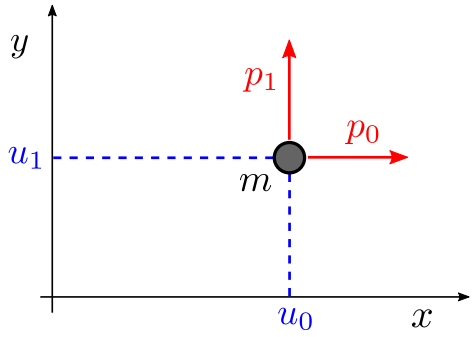

Figure 3.1: Point mass element

The equation of motion for this system can be written down directly by using Newton's second law of motion:

$$\underbrace{\begin{bmatrix} m & 0 \\ 0 & m \end{bmatrix}}_{\boldsymbol{M}} \underbrace{\begin{bmatrix} \ddot{u}_0 \\ \ddot{u}_1 \end{bmatrix}}_{\ddot{\boldsymbol{u}}} + \underbrace{\begin{bmatrix} 0 \\ 0 \end{bmatrix}}_{\boldsymbol{q}(\boldsymbol{u},\dot{\boldsymbol{u}})} = \underbrace{\begin{bmatrix} p_0 \\ p_1 \end{bmatrix}}_{\boldsymbol{p}(t)} \tag{3.8}$$

This immediately gives us the element's mass matrix, internal forces (which are zero) and external forces. Because the internal forces are zero, the tangent stiffness matrix is zero as well:

$$\boldsymbol{K} = \frac{\partial \boldsymbol{q}}{\partial \boldsymbol{u}} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}. \tag{3.9}$$

3.2 Bar Element

A bar only transfers forces in longitudinal direction, it has no bending stiffness. Figure 3.2 shows a bar with an initial length of L that is subjected to a normal force N which causes it to elongate by ΔL.

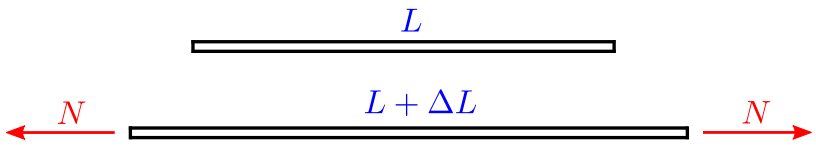

Figure 3.2: Elongation of a bar under load

In order to simulate the damping properties of the string, a viscoelastic material model of the form

$$\sigma = E\,\varepsilon + \eta\,\dot{\varepsilon} \tag{3.10}$$

is chosen. Here E is the elastic modulus and η the viscosity of the material. This model is called the Kelvin-Voigt model and combines linear elastic and linear viscous behaviour. The normal force in the bar given a constant cross section area A is therefore

$$\begin{aligned} N &= \sigma\, A \\ &= EA\,\epsilon + \eta A\,\dot{\varepsilon} \\ &= \frac{EA}{L}\,\Delta L + \frac{\eta A}{L}\,\Delta \dot{L}. \end{aligned} \tag{3.11}$$

The product EA is also called the longitudinal stiffness. The bar is now placed between the two nodes A and B as shown in figure 3.3. Its configuration is described by the nodal displacements $\boldsymbol{u} = (u_0,\, u_1,\, u_2,\, u_3)^{\mathsf{T}}$.

The position vectors of the two nodes are given by

$$\boldsymbol{r}_A = \begin{bmatrix} u_0 \\ u_1 \end{bmatrix}, \quad \boldsymbol{r}_B = \begin{bmatrix} u_2 \\ u_3 \end{bmatrix}. \tag{3.12}$$

The mass of the element is assumed to be concentrated at the nodes, so that each of them is carrying half of the total mass. This is called a lumped mass approach and will later lead to a constant, diagonal mass matrix. For a bar with density ρ the mass of a node is therefore $\frac{1}{2}\rho AL$. With this in mind we can write down Newton's second law of

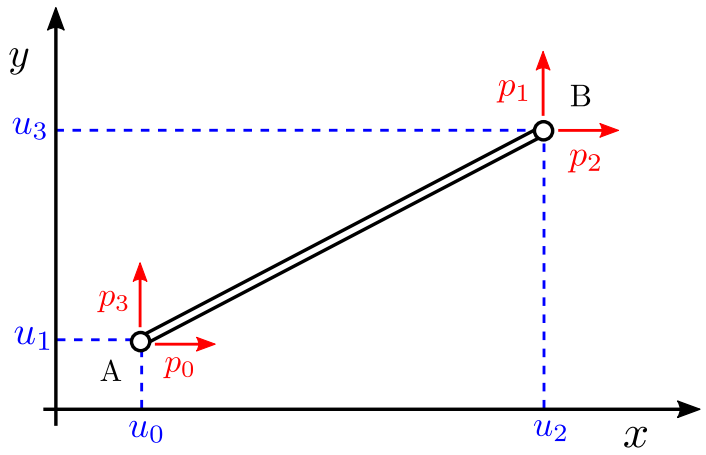

Figure 3.3: Bar element

motion for each of the two nodes,

$$\frac{\rho AL}{2}\,\ddot{\boldsymbol{r}}_A = \boldsymbol{q}_A + \boldsymbol{p}_A, \tag{3.13}$$

$$\frac{\rho AL}{2}\,\ddot{\boldsymbol{r}}_B = \boldsymbol{q}_B + \boldsymbol{p}_B. \tag{3.14}$$

Here, $\boldsymbol{q}_A$, $\boldsymbol{q}_B$ are the elastic forces that the bar exerts to the respective nodes and $\boldsymbol{p}_A$, $\boldsymbol{p}_B$ are the externally applied loads. The elastic forces can be obtained by multiplying the scalar normal force N in the bar (3.11) with the tangent vector of the connecting line between the two nodes to account for the direction,

$$\boldsymbol{q}_A = N\,\frac{\boldsymbol{r}_B - \boldsymbol{r}_A}{|\boldsymbol{r}_B - \boldsymbol{r}_A|}, \quad \boldsymbol{q}_B = -\boldsymbol{q}_A. \tag{3.15}$$

The elongation ΔL can be calculated by subtracting the initial length of the element from its actual length,

$$\Delta L = |\boldsymbol{r}_B - \boldsymbol{r}_A| - L, \tag{3.16}$$

$$\Delta \dot{L} = \frac{d}{dt}\,|\boldsymbol{r}_B - \boldsymbol{r}_A| = \frac{\boldsymbol{r}_B - \boldsymbol{r}_A}{|\boldsymbol{r}_B - \boldsymbol{r}_A|}\,(\dot{\boldsymbol{r}}_B - \dot{\boldsymbol{r}}_A). \tag{3.17}$$

Combining equations (3.13), (3.14) with (3.15), (3.16) and (3.17) leads to the element's equation of motion,

$$\frac{\rho AL}{2}\begin{bmatrix}\ddot{\boldsymbol{r}}_A\\ \ddot{\boldsymbol{r}}_B\end{bmatrix} + \frac{\eta A}{L}\begin{bmatrix}\dot{\boldsymbol{r}}_A - \dot{\boldsymbol{r}}_B\\ \dot{\boldsymbol{r}}_B - \dot{\boldsymbol{r}}_A\end{bmatrix} + \frac{EA}{L}\left(1 - \frac{L}{|\boldsymbol{r}_B - \boldsymbol{r}_A|}\right)\begin{bmatrix}\boldsymbol{r}_A - \boldsymbol{r}_B\\ \boldsymbol{r}_B - \boldsymbol{r}_A\end{bmatrix} = \begin{bmatrix}\boldsymbol{p}_A\\ \boldsymbol{p}_B\end{bmatrix}. \tag{3.18}$$

With (3.12) and the abbreviations $\Delta x = u_2 - u_0$, $\Delta y = u_3 - u_1$ this expands to

$$\underbrace{\frac{\rho AL}{2}\begin{bmatrix}1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1\end{bmatrix}}_{\boldsymbol{M}}\underbrace{\begin{bmatrix}\ddot{u}_0\\ \ddot{u}_1\\ \ddot{u}_2\\ \ddot{u}_3\end{bmatrix}}_{\ddot{\boldsymbol{u}}}+\underbrace{\frac{\eta A}{L}\begin{bmatrix}-\Delta\dot{x}\\ -\Delta\dot{y}\\ \Delta\dot{x}\\ \Delta\dot{y}\end{bmatrix}+\frac{EA}{L}\left(1-\frac{L}{\sqrt{\Delta x^2+\Delta y^2}}\right)\begin{bmatrix}-\Delta x\\ -\Delta y\\ \Delta x\\ \Delta y\end{bmatrix}}_{\boldsymbol{q}(\boldsymbol{u},\dot{\boldsymbol{u}})}=\underbrace{\begin{bmatrix}p_0\\ p_1\\ p_2\\ p_3\end{bmatrix}}_{\boldsymbol{p}}. \tag{3.19}$$

Note that the vector $\boldsymbol{q}$ of internal forces depends on the displacements in a nonlinear way despite the bar itself being considered linear-elastic. The nonlinearity arises only from the geometry of the arbitrarily large nodal displacements. This is called geometric nonlinearity as opposed to material nonlinearity.

Deriving the internal forces with respect to $\boldsymbol{u}$ results in the tangent stiffness matrix

$$\boldsymbol{K}=\frac{\partial\boldsymbol{q}}{\partial\boldsymbol{u}}=\frac{EA}{L}\left(1-\frac{L}{\sqrt{\Delta x^2+\Delta y^2}}\right)\begin{bmatrix}1 & 0 & -1 & 0\\ 0 & 1 & 0 & -1\\ -1 & 0 & 1 & 0\\ 0 & -1 & 0 & 1\end{bmatrix}$$
$$+\frac{EA}{(\sqrt{\Delta x^2+\Delta y^2})^3}\begin{bmatrix}\Delta x^2 & \Delta x\Delta y & -\Delta x^2 & -\Delta x\Delta y\\ \Delta x\Delta y & \Delta y^2 & -\Delta x\Delta y & -\Delta y^2\\ -\Delta x^2 & -\Delta x\Delta y & \Delta x^2 & \Delta x\Delta y\\ -\Delta x\Delta y & -\Delta y^2 & \Delta x\Delta y & \Delta y^2\end{bmatrix}.$$

Similarly, deriving the internal forces with respect to $\dot{\boldsymbol{u}}$ results in the tangent damping matrix

$$\boldsymbol{D}=\frac{\partial\boldsymbol{q}}{\partial\dot{\boldsymbol{u}}}=\frac{\eta A}{L}\begin{bmatrix}1 & 0 & -1 & 0\\ 0 & 1 & 0 & -1\\ -1 & 0 & 1 & 0\\ 0 & -1 & 0 & 1\end{bmatrix}.$$

3.3 Beam Element

The beam element will be developed in several steps. First of all we will consider a standard linear Euler-Bernoulli element for small deformations. Next we're going to see how laminated cross sections can be accounted for. Finally, the linear beam element will be used as the base for a fully geometrically nonlinear element by means of a floating frame of reference approach.

3.3.1 Linear Euler-Bernoulli Beam

Consider the beam element shown in figure 3.4. In the undeformed reference state it is aligned with the x-axis and has an initial length of L. The deformed state of the element is described by the (small) longitudinal and transversal displacements $v(x,t)$ and $w(x,t)$, which are continuous functions of both time and the position along the x-axis.

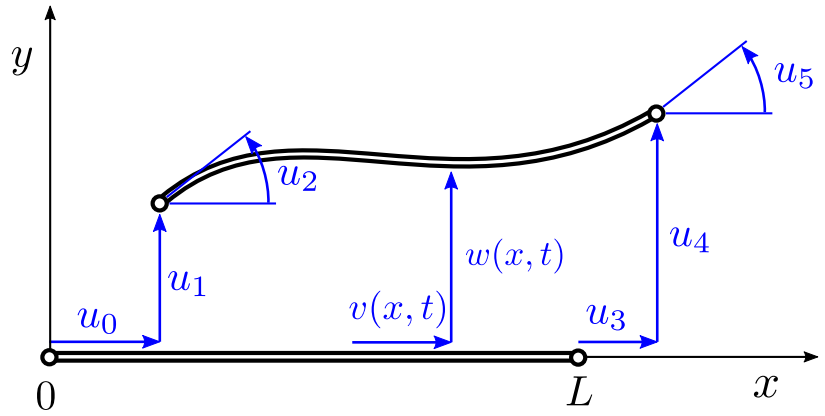

Figure 3.4: Beam element in reference and deformed configurations

The finite element approach involves discretising this continuous displacement field by a finite number of degrees of freedom. This is achieved by choosing the nodal dispacements $\boldsymbol{u}(t) = \left[u_0(t), \ldots, u_5(t)\right]^\mathsf{T}$ as the degrees of freedom and interpolating $v(x,t)$ and $w(x,t)$ as

$$v(x,t) \approx \boldsymbol{V}(x)\,\boldsymbol{u}(t), \tag{3.20}$$

$$w(x,t) \approx \boldsymbol{W}(x)\,\boldsymbol{u}(t), \tag{3.21}$$

with the shape functions $\boldsymbol{V}(x)$ and $\boldsymbol{W}(x)$. As can be seen from figure 3.4 the shape functions have to be chosen such that the following boundary conditions in regards to the nodal displacements are satisfied,

$$
\begin{aligned}
v(0,t) &= u_0(t), & w(0,t) &= u_1(t), & w'(0,t) &= u_2(t),\\
v(L,t) &= u_3(t), & w(L,t) &= u_4(t), & w'(L,t) &= u_5(t).
\end{aligned}
\tag{3.22}
$$

As the displacements were said to be small, the rotation angles of the nodes have been assumed to be equal to the derivative w' of the bending line here (small angle approximation). Most frequently, for this standard type of beam element, cubic polynomials are used for $\boldsymbol{W}(x)$ and a linear ones for $\boldsymbol{V}(x)$. Coincidentally that is also the exact analytical solution for the static deformation of the beam. The free constants in those polynomials are determined by the boundary conditions (3.22). The results are given by [11] as

$$
\boldsymbol{W}(x) = \begin{bmatrix} 0 \\ (1-\frac{x}{L})^2\,(1+2\,\frac{x}{L}) \\ x\,(1-\frac{x}{L})^2 \\ 0 \\ \frac{x^2}{L^2}\,(3-2\,\frac{x}{L}) \\ -\frac{x^2}{L}\,(1-\frac{x}{L}) \end{bmatrix}^{\mathrm{T}}, \qquad \boldsymbol{V}(x) = \begin{bmatrix} 1-\frac{x}{L} \\ 0 \\ 0 \\ \frac{x}{L} \\ 0 \\ 0 \end{bmatrix}^{\mathrm{T}}. \tag{3.23}
$$

We now have a kinematic description of the beam element that has only a finite number of degrees of freedom. The vector of elastic forces $\boldsymbol{q}(\boldsymbol{u})$ for the system can be obtained by differentiating the system's total potential energy [13], which in turn can be calculated by integrating the product of forces and deformations over the element length,

$$
\boldsymbol{q}(\boldsymbol{u}) = \frac{\partial}{\partial \boldsymbol{u}}\, V(\boldsymbol{u}) = \frac{\partial}{\partial \boldsymbol{u}} \left(\frac{1}{2} \int_0^L \begin{bmatrix} N \\ M \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \varepsilon \\ \kappa \end{bmatrix} \mathrm{d}x \right). \tag{3.24}
$$

Here, N and M are the normal force and the bending moment within the beam's cross sections, while ε and κ are the corresponding longitudinal strain and curvature of the beam's centerline. They are related to each other by the constitutive equation

$$
\begin{bmatrix} N \\ M \end{bmatrix} = \begin{bmatrix} C_{\varepsilon\varepsilon} & C_{\varepsilon\kappa} \\ C_{\varepsilon\kappa} & C_{\kappa\kappa} \end{bmatrix} \begin{bmatrix} \varepsilon \\ \kappa \end{bmatrix}. \tag{3.25}
$$

The constants $C_{\varepsilon\varepsilon}$, $C_{\kappa\kappa}$ and $C_{\varepsilon\kappa}$ are linear-elastic constants of the beam's cross section and will be determined later when we look at laminated beams. Combining the constitutive equation (3.25) with (3.24), the elastic forces turn into

$$
\boldsymbol{q}(\boldsymbol{u}) = \frac{\partial}{\partial \boldsymbol{u}} \left(\frac{1}{2} \int_0^L \begin{bmatrix} \varepsilon \\ \kappa \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} C_{\varepsilon\varepsilon} & C_{\varepsilon\kappa} \\ C_{\varepsilon\kappa} & C_{\kappa\kappa} \end{bmatrix} \begin{bmatrix} \varepsilon \\ \kappa \end{bmatrix} \mathrm{d}x \right). \tag{3.26}
$$

The strain ε and curvature κ for this beam can be calculated as

$$\varepsilon(x,t) = u'(x,t) = \boldsymbol{V}'(x)\,\boldsymbol{u}(t), \tag{3.27}$$
$$\kappa(x,t) = w''(x,t) = \boldsymbol{W}''(x)\,\boldsymbol{u}(t). \tag{3.28}$$

Substituting these into equation (3.26) finally leads to the following expression for the elastic forces,

$$\begin{aligned}\boldsymbol{q}(\boldsymbol{u}) &= \frac{\partial}{\partial \boldsymbol{u}} \left(\frac{1}{2}\,\boldsymbol{u}^{\mathrm{T}}(t) \underbrace{\left(\int_0^L \begin{bmatrix} \boldsymbol{V}'(x) \\ \boldsymbol{W}''(x) \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} C_{\varepsilon\varepsilon} & C_{\varepsilon\kappa} \\ C_{\varepsilon\kappa} & C_{\kappa\kappa} \end{bmatrix} \begin{bmatrix} \boldsymbol{V}'(x) \\ \boldsymbol{W}''(x) \end{bmatrix} \mathrm{d}x \right)}_{\boldsymbol{C}} \boldsymbol{u}(t) \right), \\ &= \boldsymbol{C}\,\boldsymbol{u}(t). \end{aligned} \tag{3.29}$$

The internal forces depend linearly on the displacements in a way that is determined by the stiffness matrix $\boldsymbol{C}$. Carrying out the integration yields

$$\boldsymbol{C} = \begin{bmatrix} \frac{1}{L}C_{\varepsilon\varepsilon} & 0 & \frac{1}{L}C_{\varepsilon\kappa} & -\frac{1}{L}C_{\varepsilon\varepsilon} & 0 & -\frac{1}{L}C_{\varepsilon\kappa} \\ 0 & \frac{12}{L^3}C_{\kappa\kappa} & \frac{6}{L^2}C_{\kappa\kappa} & 0 & -\frac{12}{L^3}C_{\kappa\kappa} & \frac{6}{L^2}C_{\kappa\kappa} \\ \frac{1}{L}C_{\varepsilon\kappa} & \frac{6}{L^2}C_{\kappa\kappa} & \frac{4}{L}C_{\kappa\kappa} & -\frac{1}{L}C_{\varepsilon\kappa} & -\frac{6}{L^2}C_{\kappa\kappa} & \frac{2}{L}C_{\kappa\kappa} \\ -\frac{1}{L}C_{\varepsilon\varepsilon} & 0 & -\frac{1}{L}C_{\varepsilon\kappa} & \frac{1}{L}C_{\varepsilon\varepsilon} & 0 & \frac{1}{L}C_{\varepsilon\kappa} \\ 0 & -\frac{12}{L^3}C_{\kappa\kappa} & -\frac{6}{L^2}C_{\kappa\kappa} & 0 & \frac{12}{L^3}C_{\kappa\kappa} & -\frac{6}{L^2}C_{\kappa\kappa} \\ -\frac{1}{L}C_{\varepsilon\kappa} & \frac{6}{L^2}C_{\kappa\kappa} & \frac{2}{L}C_{\kappa\kappa} & \frac{1}{L}C_{\varepsilon\kappa} & -\frac{6}{L^2}C_{\kappa\kappa} & \frac{4}{L}C_{\kappa\kappa} \end{bmatrix}. \tag{3.30}$$

The next step will be to determine the elastic constants $C_{\varepsilon\varepsilon}$, $C_{\kappa\kappa}$ and $C_{\varepsilon\kappa}$ for the kind of cross sections we're interested in.

3.3.2 Laminated Cross Sections

The following derivations have been adapted from [10] and [12]. Consider the laminated cross section shown in figure 3.5. It is symmetric with respect to the y-axis, bends/rotates around the z-axis and consists of a number of rectangular layers. Each layer has its own density ρ_i and elastic modulus E_i. The geometry of the layers is given by their width w_i, height h_i and center position y_i.

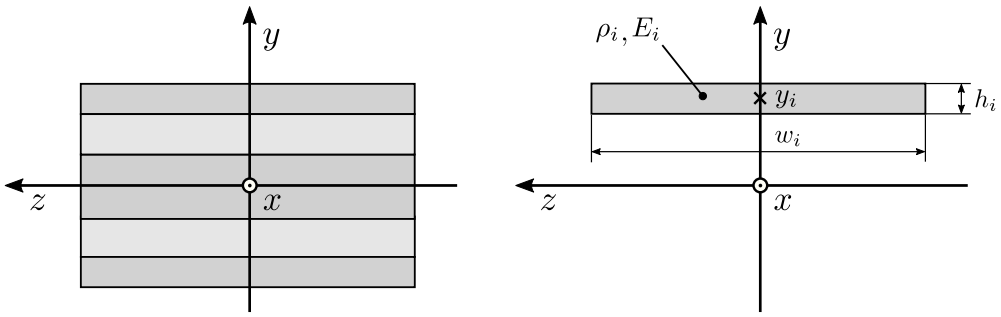

Figure 3.5: Composite cross section and single layer

In Euler-Bernoulli beam theory the cross sections are assumed to stay flat and perpendicular to the beam axis during deformation. We assume this to be true for our laminated cross section as well. The distribution of longitudinal strain over the section is therefore given by the continuous function

$$\overline{\varepsilon}(y) = \varepsilon - \kappa\, y, \tag{3.31}$$

where ε is the strain of the beam's centerline in x-direction and κ its curvature around the z-axis. The stress distribution however is not necessarily continuous, because each layer can have a different elastic modulus. The stress within a layer is given by hooke's law as

$$\sigma_i(y) = E_i \cdot \overline{\varepsilon}(y), \quad y \in [y_i - \frac{h_i}{2},\, y_i + \frac{h_i}{2}]. \tag{3.32}$$

In order to determine the elastic constants in the constitutive equation (3.25) we calculate the normal force N and bending moment M acting on the cross section by integrating the stresses (3.32) over the section's area,

$$
\begin{aligned}
N &= \int_A \sigma \,\mathrm{d}A = \sum_i \int_{A_i} \sigma_i \,\mathrm{d}A_i, \\
&= \sum_i \int_{A_i} E_i \,\overline{\varepsilon}(y) \,\mathrm{d}A_i = \sum_i E_i \int_{A_i} (\varepsilon - \kappa\, y) \,\mathrm{d}A_i, \\
&= \left(\sum_i E_i \int_{A_i} \mathrm{d}A_i \right) \varepsilon - \left(\sum_i E_i \int_{A_i} y \,\mathrm{d}A_i \right) \kappa, \\
&= \underbrace{\left(\sum_i E_i \, A_i \right)}_{C_{\varepsilon\varepsilon}} \varepsilon - \underbrace{\left(\sum_i E_i \, A_i \, y_i \right)}_{C_{\varepsilon\kappa}} \kappa.
\end{aligned} \tag{3.33}
$$

The same can be done for the bending moment,

$$
\begin{aligned}
M &= -\int_A y \,\sigma \,\mathrm{d}A = -\sum_i \int_{A_i} y \,\sigma_i \,\mathrm{d}A_i, \\
&= -\sum_i \int_{A_i} y \, E_i \,(\varepsilon - \kappa\, y) \,\mathrm{d}A = -\sum_i \int_{A_i} (E_i \,\varepsilon \, y - E_i \,\kappa \, y^2) \,\mathrm{d}A, \\
&= -\left(\sum_i E_i \int_{A_i} y \, dA_i \right) \varepsilon + \left(\sum_i E_i \int_{A_i} y^2 \, dA_i \right) \kappa, \\
&= \underbrace{-\left(\sum_i E_i \, A_i \, y_i \right)}_{C_{\varepsilon\kappa}} \varepsilon + \underbrace{\left(\sum_i E_i \, I_i \right)}_{C_{\kappa\kappa}} \kappa.
\end{aligned} \tag{3.34}
$$

The elastic constants describing the relationship between forces and deformation for the laminated cross section are therefore

$$
C_{\varepsilon\varepsilon} = \sum_i E_i \, A_i, \quad C_{\kappa\kappa} = \sum_i E_i \, I_i, \quad C_{\varepsilon\kappa} = -\sum_i E_i \, A_i \, y_i. \tag{3.35}
$$

In the case of the rectangular layers shown above, area A_i and second moment of inertia I_i evaluate to

$$
A_i = w_i \, h_i, \tag{3.36}
$$

$$
I_i = A_i \left(\frac{h_i^2}{12} + y_i^2 \right). \tag{3.37}
$$

Later we're also going to need the linear density (mass per unit length) of the laminated beam. It can be calculated by summing the the densities of the individual layers,

$$\overline{\rho A} = \sum_i \rho_i \, A_i. \tag{3.38}$$

Pretty straightforward.

3.3.3 Floating Frame of Reference

The floating frame of reference formulation is an easy way to construct a geometrically nonlinear finite element by reusing linear-elastic results. The beam element is going to use a floating frame of reference approach based on [7] for the elastic forces, together with a lumped mass matrix according to [8]. Key component is a floating frame of reference that is attached to the element and translates/rotates along with it. The total motion of the element is then described as a superposition of two parts:

- The arbitrarily large, nonlinear rigid body motion (translation, rotation) of the reference frame
- Small, linear-elastic deformations of the element with respect to its floating reference frame

Therefore the elastic deformation of the element can be treated as linear (because it's small), but on the other hand the element is still allowed to undergo arbitrarily large, nonlinear translations and rotations. Note that we only assume the elastic deformations of the single elements to be small with respect to their frame of reference. A structure composed of many such elements can still undergo large deformations as long as that assumption is valid. (And it can always be made valid by using enough elements.)

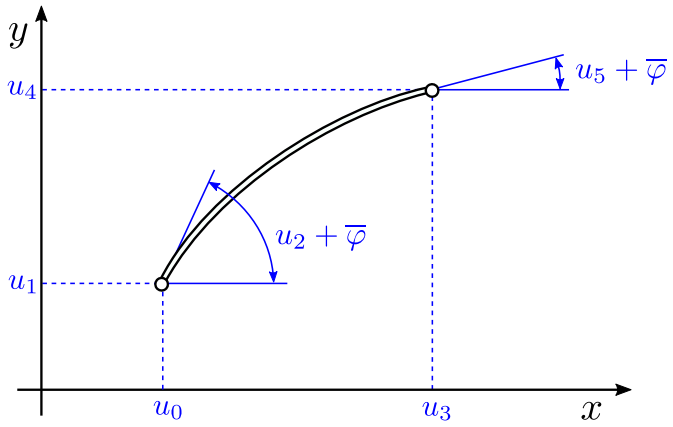

Figure 3.6: Nodal displacements of the beam element

Let's apply this approach to the beam element shown in figure 3.6. It has two nodes with the (arbitrarily large) nodal displacements $\boldsymbol{u} = (u_0, \ldots, u_5)$ measured against the $\{x, y\}$ coordinate system. The angle $\overline{\varphi}$ is a simple offset so that the rotation angles u_2 and u_5 don't necessarily have to correspond to the beam axis. This helps in practice when two adjacent elements share a node but are not tangential to each other.

We now introduce a floating frame of reference $\{x', y'\}$ as shown in figure 3.7. The x'-axis passes through both of the nodes (secant system) and the coordinate origin is placed at the left node. Within this reference system we now describe the elastic deformation

with the three displacements $\boldsymbol{e} = (e_0,\, e_1,\, e_2)^{\mathsf{T}}$, where e_0 is the longitudinal extension of the beam and e_1, e_2 are the rotation angles of the nodes with respect to the x'-axis.

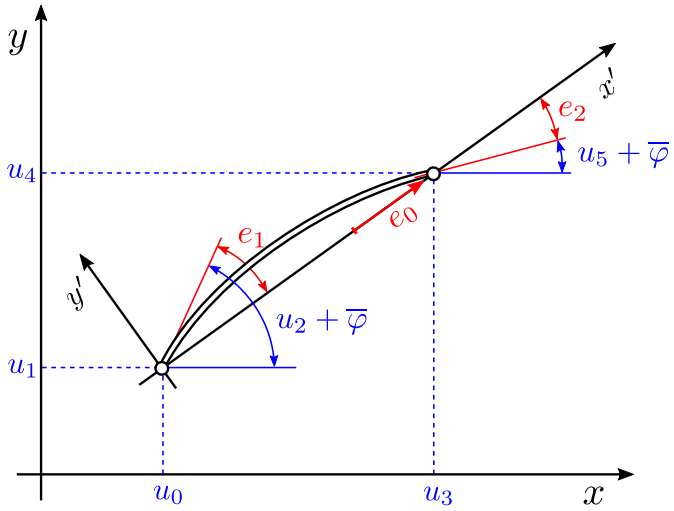

Figure 3.7: Floating frame of reference and elastic deformations

Now obviously the elastic displacements $\boldsymbol{e}$ depend on the overall displacements $\boldsymbol{u}$ in some way, so the first thing we'll do is express this kinematic relationship. We introduce the auxiliary variables

$$\Delta x = u_3 - u_0, \tag{3.39}$$
$$\Delta y = u_4 - u_1, \tag{3.40}$$
$$\varphi = \operatorname{atan2}\left(\frac{\Delta y}{\Delta x}\right). \tag{3.41}$$

Here, φ is the rotation angle of the floating reference frame. Using these abbreviations we can calculate the elastic deformations by using some elementary geometry.

Calculating e_0:

$$e_0 = \sqrt{\Delta x^2 + \Delta y^2} - L \tag{3.42}$$

Calculating e_1:

$$\begin{aligned}
\sin(e_1) &= \sin(u_2 + \overline{\varphi} - \varphi) = \sin(u_2)\cos(\overline{\varphi} - \varphi) + \cos(u_2)\sin(\overline{\varphi} - \varphi) \\
\cos(e_1) &= \cos(u_2 + \overline{\varphi} - \varphi) = \cos(u_2)\cos(\overline{\varphi} - \varphi) - \sin(u_2)\sin(\overline{\varphi} - \varphi) \\
e_1 &= \arctan\left(\frac{\sin(e_1)}{\cos(e_1)}\right)
\end{aligned} \tag{3.43}$$

Calculating e_2:

$$\begin{aligned}
\sin(e_2) &= \sin(u_5 + \overline{\varphi} - \varphi) = \sin(u_5)\cos(\overline{\varphi} - \varphi) + \cos(u_5)\sin(\overline{\varphi} - \varphi) \\
\cos(e_2) &= \cos(u_5 + \overline{\varphi} - \varphi) = \cos(u_5)\cos(\overline{\varphi} - \varphi) - \sin(u_5)\sin(\overline{\varphi} - \varphi) \\
e_2 &= \arctan\left(\frac{\sin(e_2)}{\cos(e_2)}\right)
\end{aligned} \tag{3.44}$$

Putting all those together, the elastic deformations can be written as

$$\boldsymbol{e}(\boldsymbol{u}) = \begin{bmatrix} \sqrt{\Delta x^2 + \Delta y^2} - L \\ \arctan\left(\frac{\sin(u_2)\cos(\overline{\varphi}-\varphi)+\cos(u_2)\sin(\overline{\varphi}-\varphi)}{\cos(u_2)\cos(\overline{\varphi}-\varphi)-\sin(u_2)\sin(\overline{\varphi}-\varphi)}\right) \\ \arctan\left(\frac{\sin(u_5)\cos(\overline{\varphi}-\varphi)+\cos(u_5)\sin(\overline{\varphi}-\varphi)}{\cos(u_5)\cos(\overline{\varphi}-\varphi)-\sin(u_5)\sin(\overline{\varphi}-\varphi)}\right) \end{bmatrix}. \tag{3.45}$$

Having this out of the way, the vector of elastic forces $\boldsymbol{q}(\boldsymbol{u})$ for the element will be obtained by differentiating the total potential energy V with respect to the displacements $\boldsymbol{u}$,

$$\boldsymbol{q}(\boldsymbol{u}) = \frac{\partial V}{\partial \boldsymbol{u}}. \tag{3.46}$$

Now because we consider the beam as linear-elastic within its reference frame we can express its elastic energy as the quadratic expression

$$V = \boldsymbol{e}^{\mathsf{T}} \boldsymbol{C}\, \boldsymbol{e}, \tag{3.47}$$

where $\boldsymbol{C}$ is the stiffness matrix of the beam. Its entries are a subset of the stiffness matrix (3.30) that we derived earlier for the linear beam element,

$$\boldsymbol{C} = \frac{1}{L}\begin{bmatrix} C_{\varepsilon\varepsilon} & -C_{\varepsilon\kappa} & C_{\varepsilon\kappa} \\ -C_{\varepsilon\kappa} & 4\,C_{\kappa\kappa} & 2\,C_{\kappa\kappa} \\ C_{\varepsilon\kappa} & 2\,C_{\kappa\kappa} & 4\,C_{\kappa\kappa} \end{bmatrix}. \tag{3.48}$$

Using this expression for the beam's energy, equation (3.46) becomes

$$\begin{aligned}
\boldsymbol{q}(\boldsymbol{u}) = \frac{\partial V}{\partial \boldsymbol{u}} &= \frac{\partial V}{\partial \boldsymbol{e}}\,\frac{\partial \boldsymbol{e}}{\partial \boldsymbol{u}} = \left(\frac{\partial \boldsymbol{e}}{\partial \boldsymbol{u}}\right)^{\mathsf{T}} \boldsymbol{C}\, \boldsymbol{e}(\boldsymbol{u}) \\
&= \boldsymbol{J}(\boldsymbol{u})^{\mathsf{T}} \boldsymbol{C}\, \boldsymbol{e}(\boldsymbol{u}),
\end{aligned} \tag{3.49}$$

with the jacobian matrix $\boldsymbol{J}(\boldsymbol{u}) = \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{u}}$. Differentiating (3.45) gives us

$$\boldsymbol{J}(\boldsymbol{u}) = \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{u}} = \begin{bmatrix} -J_0 & -J_1 & 0 & J_0 & J_1 & 0 \\ -J_3 & J_2 & 1 & J_3 & -J_2 & 0 \\ -J_3 & J_2 & 0 & J_3 & -J_2 & 1 \end{bmatrix}, \tag{3.50}$$

$$\begin{aligned} J_0 &= \frac{\Delta x}{\sqrt{\Delta x^2 + \Delta y^2}}, \quad J_1 = \frac{\Delta y}{\sqrt{\Delta x^2 + \Delta y^2}}, \\ J_2 &= \frac{\Delta x}{\Delta x^2 + \Delta y^2}, \quad J_3 = \frac{\Delta y}{\Delta x^2 + \Delta y^2}. \end{aligned} \tag{3.51}$$

As always, the element's tangent stiffness matrix is obtained by differentiating the internal forces:

$$\begin{aligned} \boldsymbol{K}(\boldsymbol{u}) &= \frac{\partial \boldsymbol{q}}{\partial \boldsymbol{u}} = \frac{\partial}{\partial \boldsymbol{u}} \left(\boldsymbol{J}^{\mathsf{T}}(\boldsymbol{u})\, \boldsymbol{C} \right) \boldsymbol{e}(\boldsymbol{u}) + \boldsymbol{J}^{\mathsf{T}}(\boldsymbol{u})\, \boldsymbol{C}\, \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{u}} \\ &= \left[\frac{\partial \boldsymbol{J}^{\mathsf{T}}(\boldsymbol{u})}{\partial u_1} \boldsymbol{C}\boldsymbol{e}(\boldsymbol{u}), \ldots \frac{\partial \boldsymbol{J}^{\mathsf{T}}(\boldsymbol{u})}{\partial u_6} \boldsymbol{C}\boldsymbol{e}(\boldsymbol{u}) \right] + \boldsymbol{J}^{\mathsf{T}}(\boldsymbol{u})\, \boldsymbol{C}\, \boldsymbol{J}(\boldsymbol{u}). \end{aligned} \tag{3.52}$$

The partial derivatives of the jacobian matrix $\boldsymbol{J}$ are:

$$\begin{aligned} \frac{\partial \boldsymbol{J}}{\partial u_0} &= \begin{bmatrix} -b_0 & -b_2 & 0 & b_0 & b_2 & 0 \\ -b_5 & b_3 & 0 & b_5 & -b_3 & 0 \\ -b_5 & b_3 & 0 & b_5 & -b_3 & 0 \end{bmatrix}, \quad \frac{\partial \boldsymbol{J}}{\partial u_2} = 0, \quad \frac{\partial \boldsymbol{J}}{\partial u_4} = -\frac{\partial \boldsymbol{J}}{\partial u_1}, \\ \frac{\partial \boldsymbol{J}}{\partial u_1} &= \begin{bmatrix} -b_2 & -b_1 & 0 & b_2 & b_1 & 0 \\ -b_4 & b_5 & 0 & b_4 & -b_5 & 0 \\ -b_4 & b_5 & 0 & b_4 & -b_5 & 0 \end{bmatrix}, \quad \frac{\partial \boldsymbol{J}}{\partial u_3} = -\frac{\partial \boldsymbol{J}}{\partial u_0}, \quad \frac{\partial \boldsymbol{J}}{\partial u_5} = 0. \end{aligned} \tag{3.53}$$

With the abbreviations

$$\begin{aligned} a_0 &= \frac{1}{(\Delta x^2 + \Delta y^2)^{\frac{1}{2}}}, \quad a_1 = \frac{1}{\Delta x^2 + \Delta y^2} \\ a_2 &= \frac{1}{(\Delta x^2 + \Delta y^2)^{\frac{3}{2}}}, \quad a_3 = \frac{1}{(\Delta x^2 + \Delta y^2)^2} \end{aligned} \tag{3.54}$$

$$\begin{aligned} b_0 &= a_2 \Delta x^2 - a_0, \quad b_1 = a_2 \Delta y^2 - a_0, \quad b_2 = a_2 \Delta x \Delta y \\ b_3 &= 2a_3 \Delta x^2 - a_1, \quad b_4 = 2a_3 \Delta y^2 - a_1, \quad b_5 = 2a_3 \Delta x \Delta y \end{aligned} \tag{3.55}$$

The mass matrix is constructed by lumping the total mass and inertia of the element to the nodes, like before. The total mass is $\overline{\rho A}\, L$, using the linear density (3.38) of the laminated cross sections. Each translational degree of freedom therefore carries half of that mass. But the nodes also have a rotational degree of freedom each. As it turns out, distributing the rotational inertia of a beam cannot be done in such a straightforward way. There is simply no single "solution", only different approximations. One possibility, according to [8], is to use the mass matrix

$$\boldsymbol{M} = \overline{\rho A}\, L \begin{bmatrix} \frac{1}{2} & & & & & & \\ & \frac{1}{2} & & & & & \\ & & \alpha L^2 & & & & \\ & & & \frac{1}{2} & & & \\ & & & & \frac{1}{2} & & \\ & & & & & \alpha L^2 \end{bmatrix}, \tag{3.56}$$

where $\alpha \in \,]0,\, 1/50]$ is a tunable parameter. Tests with the bow simulation suggest that the choice of α barely has any influence on the results. Therefore the maximum value $\alpha = 1/50$ has been chosen for numerical reasons.

3.4 Contact element

The contact element is used to treat contact between a point and a surface. It uses a simple penalty approach: If the distance between the point and the surface is positive, there is no contact and nothing happens. But if the distance is negative, a contact force proportional to the penetration is applied that pushes the point outwards. By selecting a reasonably high contact stiffness the penetration can be kept small enough for all practical purposes.

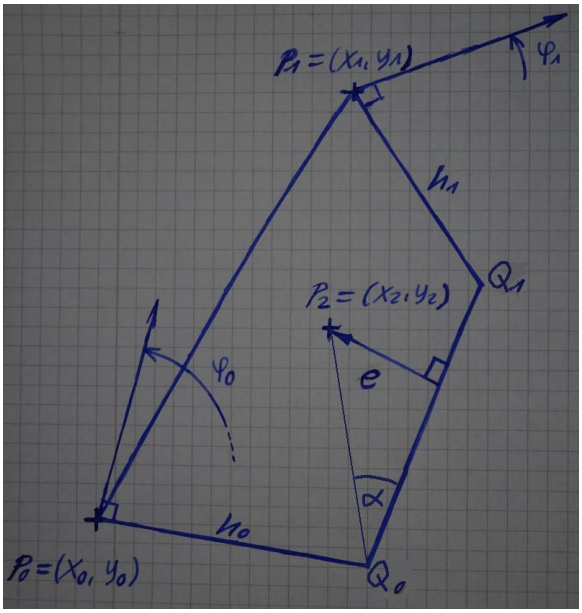

Figure 3.8: Contact element

3.4.1 Contact Detection

Contact detection 1: Bounding box test

$$
\begin{aligned}
x_{\min} &= \min\{x_0 - h_0,\ x_1 - h_1\} \\
y_{\min} &= \min\{y_0 - h_0,\ y_1 - h_1\} \\
x_{\max} &= \max\{x_0 + h_0,\ x_1 + h_1\} \\
y_{\max} &= \max\{y_0 + h_0,\ y_1 + h_1\}
\end{aligned}
$$

Necessary conditions for P_2 intersecting the segment:

$$
\begin{aligned}
x_{\min} \leq x_{\mathrm{P2}} \leq x_{\max} \\
y_{\min} \leq y_{\mathrm{P2}} \leq y_{\max}
\end{aligned}
$$

Contact detection 2: Exact contact detection

Coordinates of points Q_0 and Q_1:

$$Q_0 = \begin{bmatrix} x_0 + h_0 \sin(\varphi_0) \\ y_0 - h_0 \cos(\varphi_0) \end{bmatrix}, \quad Q_1 = \begin{bmatrix} x_1 + h_1 \sin(\varphi_1) \\ y_1 - h_1 \cos(\varphi_1) \end{bmatrix}.$$

Define the four triangles

$$T_0 : \{P_2, P_0, Q_0\}, \quad T_1 : \{P_2, P_1, P_0\}, \quad T_2 : \{P_2, Q_0, Q_1\}, \quad T_3 : \{P_2, Q_1, P_1\}$$

P_2 lies inside the square if and only if all of the four triangles are oriented counter-clockwise. [Citation needed]

A triangle $\{A, B, C\}$ is oriented counter-clockwise if

$$\begin{aligned} & (B - A) \times (C - A) > 0, \\ \Leftrightarrow \quad & (x_B - x_A)(y_C - y_A) - (y_B - y_A)(x_C - x_A) > 0. \end{aligned}$$

Contact detection 3: Penetration

$$\sin(\alpha) = \frac{e}{|P_2 - Q_0|} = \frac{(Q_1 - Q_0) \times (P_2 - Q_0)}{|Q_1 - Q_0||P_2 - Q_0|}$$

$$\begin{aligned} \Leftrightarrow \; e &= \frac{(Q_1 - Q_0) \times (P_2 - Q_0)}{|Q_1 - Q_0|} \\ &= \frac{(x_{\mathrm{Q1}} - x_{\mathrm{Q0}})(y_{\mathrm{P2}} - y_{\mathrm{Q0}}) - (y_{\mathrm{Q1}} - y_{\mathrm{Q0}})(x_{\mathrm{P2}} - x_{\mathrm{Q0}})}{\sqrt{(x_{\mathrm{Q1}} - x_{\mathrm{Q0}})^2 + (y_{\mathrm{Q1}} - y_{\mathrm{Q0}})^2}} \\ &= \frac{a_1 a_4 - a_2 a_3}{\sqrt{a_1^2 + a_2^2}} \end{aligned}$$

With the abbreviations

$$\begin{aligned} a_1 &= x_1 - x_0 - h_0 \sin(\varphi_0) + h_1 \sin(\varphi_1) \\ a_2 &= y_1 - y_0 + h_0 \cos(\varphi_0) - h_1 \cos(\varphi_1) \\ a_3 &= x_2 - x_0 - h_0 \sin(\varphi_0) \\ a_4 &= y_2 - y_0 + h_0 \cos(\varphi_0) \end{aligned}$$

3.4.2 Contact Forces

Elastic forces

The potential energy of the system can be determined as

$$V = \int_0^e F(s)\, ds$$

which yields the vector of elastic forces

$$\boldsymbol{q}(\boldsymbol{u}) = \frac{\partial V}{\partial \boldsymbol{u}} = \frac{\partial V}{\partial e}\frac{\partial e}{\partial \boldsymbol{u}} = F(e)\,\frac{\partial e}{\partial \boldsymbol{u}}. \tag{3.57}$$

Taking the derivatives:

$$\begin{aligned}
\frac{\partial e}{\partial \boldsymbol{u}} &= \frac{\partial}{\partial \boldsymbol{u}}\left(\frac{a_1 a_4 - a_2 a_3}{\sqrt{a_1^2 + a_2^2}}\right) \\
&= \frac{\partial}{\partial \boldsymbol{u}}\left(\frac{1}{\sqrt{a_1^2 + a_2^2}}\right)(a_1 a_4 - a_2 a_3) + \frac{1}{\sqrt{a_1^2 + a_2^2}}\,\frac{\partial}{\partial \boldsymbol{u}}\,(a_1 a_4 - a_2 a_3) \\
&= \underbrace{\frac{1}{\sqrt{a_1^2 + a_2^2}}}_{b_1}\underbrace{\left(a_4\frac{\partial a_1}{\partial \boldsymbol{u}} - a_3\frac{\partial a_2}{\partial \boldsymbol{u}} - a_2\frac{\partial a_3}{\partial \boldsymbol{u}} + a_1\frac{\partial a_4}{\partial \boldsymbol{u}}\right)}_{\boldsymbol{v}_1} + \underbrace{\frac{a_2 a_3 - a_1 a_4}{(a_1^2 + a_2^2)^{\frac{3}{2}}}}_{b_2}\underbrace{\left(a_1\frac{\partial a_1}{\partial \boldsymbol{u}} + a_2\frac{\partial a_2}{\partial \boldsymbol{u}}\right)}_{\boldsymbol{v}_2} \\
&= b_1\boldsymbol{v}_1 + b_2\boldsymbol{v}_2
\end{aligned}$$

$$\frac{\partial a_1}{\partial \boldsymbol{u}} = \begin{bmatrix} -1 \\ 0 \\ -h_0\cos(\varphi_0) \\ 1 \\ 0 \\ h_1\cos(\varphi_1) \\ 0 \\ 0 \end{bmatrix}, \frac{\partial a_2}{\partial \boldsymbol{u}} = \begin{bmatrix} 0 \\ -1 \\ -h_0\sin(\varphi_0) \\ 0 \\ 1 \\ h_1\sin(\varphi_1) \\ 0 \\ 0 \end{bmatrix}, \frac{\partial a_3}{\partial \boldsymbol{u}} = \begin{bmatrix} -1 \\ 0 \\ -h_0\cos(\varphi_0) \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \frac{\partial a_4}{\partial \boldsymbol{u}} = \begin{bmatrix} 0 \\ -1 \\ -h_0\sin(\varphi_0) \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Tangent stiffness matrix

$$\boldsymbol{K}(\boldsymbol{u}) = \frac{\partial \boldsymbol{q}}{\partial \boldsymbol{u}} = \frac{\partial F}{\partial \boldsymbol{u}} \left(\frac{\partial e}{\partial \boldsymbol{u}}\right)^{\mathrm{T}} + F(e)\,\frac{\partial^2 e}{\partial \boldsymbol{u}^2} = \frac{\partial F}{\partial e}\,\frac{\partial e}{\partial \boldsymbol{u}} \left(\frac{\partial e}{\partial \boldsymbol{u}}\right)^{\mathrm{T}} + F(e)\,\frac{\partial^2 e}{\partial \boldsymbol{u}^2}$$

$$\begin{aligned}
\frac{\partial^2 e}{\partial \boldsymbol{u}^2} &= \frac{\partial}{\partial \boldsymbol{u}}\left(b_1 \boldsymbol{v}_1 + b_2 \boldsymbol{v}_2\right) = \frac{\partial b_1}{\partial \boldsymbol{u}}\,\boldsymbol{v}_1^{\mathrm{T}} + \frac{\partial b_2}{\partial \boldsymbol{u}}\,\boldsymbol{v}_2^{\mathrm{T}} + b_1\,\frac{\partial \boldsymbol{v}_1}{\partial \boldsymbol{u}} + b_2\,\frac{\partial \boldsymbol{v}_2}{\partial \boldsymbol{u}} \\
\frac{\partial b_1}{\partial \boldsymbol{u}} &= -\frac{1}{\left(a_1^2 + a_2^2\right)^{\frac{3}{2}}}\left(a_1\,\frac{\partial a_1}{\partial \boldsymbol{u}} + a_2\,\frac{\partial a_2}{\partial \boldsymbol{u}}\right) \\
\frac{\partial b_2}{\partial \boldsymbol{u}} &= \frac{\sqrt{a_1^2 + a_2^2}}{a_1^6 + 3\,a_1^4\,a_2^2 + 3\,a_1^2\,a_2^4 + a_2^6}\Big(\left(2\,a_1^2\,a_4 - a_2^2\,a_4 - 3\,a_1\,a_2\,a_3\right)\frac{\partial a_1}{\partial \boldsymbol{u}} \\
&\quad + \left(a_1^2\,a_3 - 2\,a_2^2\,a_3 + 3\,a_1\,a_2\,a_4\right)\frac{\partial a_2}{\partial \boldsymbol{u}} + \left(a_2^3 + a_1^2\,a_2\right)\frac{\partial a_3}{\partial \boldsymbol{u}} + \left(-a_1^3 - a_1\,a_2^2\right)\frac{\partial a_4}{\partial \boldsymbol{u}}\Big)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial \boldsymbol{v}_1}{\partial \boldsymbol{u}} &= \frac{\partial a_4}{\partial \boldsymbol{u}}\left(\frac{\partial a_1}{\partial \boldsymbol{u}}\right)^{\mathrm{T}} - \frac{\partial a_3}{\partial \boldsymbol{u}}\left(\frac{\partial a_2}{\partial \boldsymbol{u}}\right)^{\mathrm{T}} - \frac{\partial a_2}{\partial \boldsymbol{u}}\left(\frac{\partial a_3}{\partial \boldsymbol{u}}\right)^{\mathrm{T}} + \frac{\partial a_1}{\partial \boldsymbol{u}}\left(\frac{\partial a_4}{\partial \boldsymbol{u}}\right)^{\mathrm{T}} \\
&\quad + a_4\,\frac{\partial^2 a_1}{\partial \boldsymbol{u}^2} - a_3\,\frac{\partial^2 a_2}{\partial \boldsymbol{u}^2} - a_2\,\frac{\partial^2 a_3}{\partial \boldsymbol{u}^2} + a_1\,\frac{\partial^2 a_4}{\partial \boldsymbol{u}^2} \\
\frac{\partial \boldsymbol{v}_2}{\partial \boldsymbol{u}} &= \frac{\partial a_1}{\partial \boldsymbol{u}}\left(\frac{\partial a_1}{\partial \boldsymbol{u}}\right)^{\mathrm{T}} + \frac{\partial a_2}{\partial \boldsymbol{u}}\left(\frac{\partial a_2}{\partial \boldsymbol{u}}\right)^{\mathrm{T}} + a_1\,\frac{\partial^2 a_1}{\partial \boldsymbol{u}^2} + a_2\,\frac{\partial^2 a_2}{\partial \boldsymbol{u}^2}
\end{aligned}$$

Second derivatives:

$$\left(\frac{\partial^2 a_1}{\partial \boldsymbol{u}^2}\right)_{ij} = \begin{cases} h_0\,\sin(\varphi_0) & ij = (3,3) \\ -h_1\,\sin(\varphi_1) & ij = (6,6) \\ 0 & \text{otherwise} \end{cases} \qquad \left(\frac{\partial^2 a_2}{\partial \boldsymbol{u}^2}\right)_{ij} = \begin{cases} -h_0\,\cos(\varphi_0) & ij = (3,3) \\ h_1\,\cos(\varphi_1) & ij = (6,6) \\ 0 & \text{otherwise} \end{cases}$$

$$\left(\frac{\partial^2 a_3}{\partial \boldsymbol{u}^2}\right)_{ij} = \begin{cases} h_0\,\sin(\varphi_0) & ij = (3,3) \\ 0 & \text{otherwise} \end{cases} \qquad \left(\frac{\partial^2 a_4}{\partial \boldsymbol{u}^2}\right)_{ij} = \begin{cases} -h_0\,\cos(\varphi_0) & ij = (3,3) \\ 0 & \text{otherwise} \end{cases}$$

3.4.3 Broadphase Algorithm

TODO

Chapter 4

Solving the Equations of Motion

4.1 Static Solution

Static solution in context of bow: Bracing and equilibrium path, necessity for load control and displacement control

A System with the equation of motion

$$\boldsymbol{M}\,\ddot{\boldsymbol{u}} + \boldsymbol{q}(\boldsymbol{u},\,\dot{\boldsymbol{u}}) = \boldsymbol{p} \tag{4.1}$$

is in static equilibrium if $\dot{\boldsymbol{u}},\,\ddot{\boldsymbol{u}} = \boldsymbol{0}$, leading to the static equilibrium condition

$$\boldsymbol{q}(\boldsymbol{u},\,\boldsymbol{0}) = \boldsymbol{p} \tag{4.2}$$

for the displacements $\boldsymbol{u}$ and the external loads $\boldsymbol{p}$. This is a nonlinear system of equations and can be interpreted as the internal elastic forces $\boldsymbol{q}$ being in balance with the external loads.

4.1.1 The Basic Newton-Raphson Method

Most frequently, when solving static problems, the external loads are given and the task is to calculate the corresponding equilibrium displacements. Equation (4.2) is then a nonlinear system of equations in $\boldsymbol{u}$. Solving it can only be done numerically except for very simple examples.

An often used method for this kind of problem is the Newton-Raphson method. The basic idea behind it is to linearise the equilibrium equation (4.2) around a known displacement vector $\boldsymbol{u}_i$ and solve the resulting linear equation to obtain an improved approximation $\boldsymbol{u}_{i+1}$.

Starting with an initial value $\boldsymbol{u}_0$ the procedure is repeated until the solution is accurate enough by some convergence criterion.

Linearising of (4.2) yields

$$\boldsymbol{q}(\boldsymbol{u}_i) + \underbrace{\left(\left. \frac{\partial \boldsymbol{q}}{\partial \boldsymbol{u}} \right|_{\boldsymbol{u}_i} \right)}_{\boldsymbol{K}(\boldsymbol{u}_i)} \underbrace{(\boldsymbol{u}_{i+1} - \boldsymbol{u}_i)}_{\Delta \boldsymbol{u}_i} = \boldsymbol{p} \tag{4.3}$$

Solving for $\Delta \boldsymbol{u}_i$ leads to the iterative procedure

$$\Delta \boldsymbol{u}_i = \boldsymbol{K}^{-1}(\boldsymbol{u}_i) \left(\boldsymbol{p} - \boldsymbol{q}(\boldsymbol{u}_i) \right), \tag{4.4}$$

$$\boldsymbol{u}_{i+1} = \boldsymbol{u}_i + \Delta \boldsymbol{u}_i. \tag{4.5}$$

In the implementation the tangent stiffness matrix is not actually inverted, because that's expensive and potentially inaccurate. Instead, $\Delta \boldsymbol{u}_i$ is obtained by solving the equivalent linear system of equations using a decomposition of the stiffness matrix. The symmetry of the tangent stiffness can be used here. (It isn't necessarily positive definite though, like the stiffness matrices of linear mechanical systems.)

Convergence criteria are usually formulated in terms of change in displacement $\Delta \boldsymbol{u}$, the residuum $\boldsymbol{p} - \boldsymbol{q}$ or their product $\Delta \boldsymbol{u}^{\mathsf{T}}(\boldsymbol{p} - \boldsymbol{q})$ (which has the dimension of an energy). We will use the energy criterion

$$\Delta \boldsymbol{u}_i^{\mathsf{T}} (\boldsymbol{p} - \boldsymbol{q}(\boldsymbol{u}_i)) < \varepsilon$$

The basic Newton-Raphson method can be improved and extended in many ways, for example by combining it with a line search or by avoiding the calculation of the tangent stiffness matrix at every iteration (simplified Newton-Raphson method, BFGS method [Source]).

4.1.2 Constrained Newton-Raphson method

In the previous section the Newton-Raphson method was applied to the problem of calculating the equilibrium displacements of a system when all applied forces are known. This method is used in VirtualBow for finding the braced equilibrium state of the bow (see section [Reference]). However, when calculating the draw curve we don't actually know the draw forces, just the displacement of the string center. Also: Load decrease.

A solution to this is displacement control, where we prescribe a value for one of the displacement components and instead treat the corresponding external forces as a variable that is determined during solution.

The following derivation has been adapted from [**?**]. Displacement control is a special case of a wider class of methods where the equilibrium equations are modified as

$$\boldsymbol{q}(\boldsymbol{u}) - \lambda\, \boldsymbol{p} = 0, \tag{4.6}$$

$$c(\boldsymbol{u},\, \lambda) = 0. \tag{4.7}$$

Here the external forces are scaled by the unknown load factor λ. To compensate for this additional unknown, a scalar constraint $c(\boldsymbol{u},\, \lambda) = 0$ on the displacements and the load parameter is introduced. Depending on the choice of constraint the method can be turned into load control, displacement control or other more sophisticated schemes like the arc-length method. We will carry out the derivation for the general case and specify the constraints later. This is also how these methods are implemented in VirtualBow: A general implementation for arbitrary constraints that is then specialized into load controlled and displacement controlled methods by providing the necessary constraints.

The system of nonlinear equations (4.6), (4.7) has to be solved for the displacements $\boldsymbol{u}$ and the load parameter λ. Like before, the Newton-Raphson method is used: The equations are linearised around a given approximation $\{\boldsymbol{u}_i,\, \lambda_i\}$ and solved to obtain an improved approximiation $\{\boldsymbol{u}_{i+1},\, \lambda_{i+1}\}$. This is repeated iteratively until displacements and load parameter satisfy the equilibrium equations (4.6) as well as the constraint equation (4.7) reasonably well as determined by some convergence criterion.

Carrying out the linearisation leads to the two equations

$$\begin{aligned}
\boldsymbol{q}(\boldsymbol{u}) - \lambda\, \boldsymbol{p} \;&\approx\; \boldsymbol{q}(\boldsymbol{u_i}) - \lambda_i\, \boldsymbol{p} + \underbrace{\frac{\partial \boldsymbol{q}}{\partial \boldsymbol{u}}(\boldsymbol{u_i})}_{\boldsymbol{K_i}} \underbrace{(\boldsymbol{u_{i+1}} - \boldsymbol{u_i})}_{\Delta \boldsymbol{u_i}} - \boldsymbol{p}\, \underbrace{(\lambda_{i+1} - \lambda_i)}_{\Delta \lambda_i} \\
&\approx\; \boldsymbol{q}(\boldsymbol{u_i}) - \lambda_i\, \boldsymbol{p} + \boldsymbol{K_i}\, \Delta \boldsymbol{u_i} - \boldsymbol{p}\, \Delta \lambda_i
\end{aligned} \tag{4.8}$$

$$\begin{aligned}
c(\boldsymbol{u},\, \lambda) \;&\approx\; c(\boldsymbol{u_i},\, \lambda_i) + \frac{\partial c}{\partial \boldsymbol{u}}(\boldsymbol{u_i},\, \lambda_i)(\boldsymbol{u_{i+1}} - \boldsymbol{u_i}) + \frac{\partial c}{\partial \lambda}(\boldsymbol{u_i},\, \lambda_i)(\lambda_{i+1} - \lambda_i) \\
&\approx\; c(\boldsymbol{u_i},\, \lambda_i) + \frac{\partial c}{\partial \boldsymbol{u}}(\boldsymbol{u_i},\, \lambda_i)\, \Delta \boldsymbol{u_i} + \frac{\partial c}{\partial \lambda}(\boldsymbol{u_i},\, \lambda_i)\, \Delta \lambda_i
\end{aligned} \tag{4.9}$$

Equations (4.8) and (4.9) can written as the single linear system of equations

$$\begin{bmatrix} \boldsymbol{K_i} & -\boldsymbol{p} \\ \frac{\partial c}{\partial \boldsymbol{u}}(\boldsymbol{u_i},\, \lambda_i) & \frac{\partial c}{\partial \lambda}(\boldsymbol{u_i},\, \lambda_i) \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{u_i} \\ \Delta \lambda_i \end{bmatrix} = \begin{bmatrix} \lambda_i\, \boldsymbol{p} - \boldsymbol{q}(\boldsymbol{u_i}) \\ -\boldsymbol{c}(\boldsymbol{u_i},\, f_i) \end{bmatrix}. \tag{4.10}$$

Solving this for the increments in displacement and load factor $\{\Delta \boldsymbol{u_i},\, \Delta \lambda_i\}$ allows us to calculate the next values as

$$\boldsymbol{u_{i+1}} = \boldsymbol{u_i} + \Delta \boldsymbol{u_i} \tag{4.11}$$

$$\lambda_{i+1} = \lambda_i + \Delta \lambda_i. \tag{4.12}$$

However, directly solving (4.10) is not numerically efficient because the augmented stiffness matrix is no longer symmetric. A much better way to solve this is block elimination. We use the first row of (4.10) to express $\Delta \boldsymbol{u_i}$ as

$$\Delta \boldsymbol{u_i} = \boldsymbol{\alpha} + \boldsymbol{\beta}\, \Delta\lambda_i. \tag{4.13}$$

With the auxiliary vectors

$$\boldsymbol{\alpha} = \boldsymbol{K_i^{-1}}(\lambda_i\, \boldsymbol{p} - \boldsymbol{q}(\boldsymbol{u_i})) \tag{4.14}$$

$$\boldsymbol{\beta} = \boldsymbol{K_i^{-1}}\, \boldsymbol{p} \tag{4.15}$$

These can be computed efficiently by exploiting the symmetry of the stiffness matrix $\boldsymbol{K_i}$. To calculate $\Delta\lambda_i$, we substitute (4.13) into the second row of (4.10) which gives us

$$\begin{aligned}
&\frac{\partial c}{\partial \boldsymbol{u}}(\boldsymbol{u_i},\, \lambda_i)\Delta \boldsymbol{u_i} + \frac{\partial c}{\partial \lambda}(\boldsymbol{u_i},\, \lambda_i)\, \Delta\lambda_i = -c(\boldsymbol{u_i},\, \lambda_i) \\
&\frac{\partial c}{\partial \boldsymbol{u}}(\boldsymbol{u_i},\, \lambda_i)(\boldsymbol{\alpha} + \boldsymbol{\beta}\, \Delta\lambda_i) + \frac{\partial c}{\partial \lambda}(\boldsymbol{u_i},\, \lambda_i)\, \Delta\lambda_i = -c(\boldsymbol{u_i},\, \lambda_i) \\
&\quad \vdots \\
&\Delta\lambda_i = -\frac{c(\boldsymbol{u_i},\, \lambda_i) + \frac{\partial c}{\partial \boldsymbol{u}}(\boldsymbol{u_i},\, \lambda_i)\, \boldsymbol{\alpha}}{\frac{\partial c}{\partial \boldsymbol{u}}(\boldsymbol{u_i},\, \lambda_i)\, \boldsymbol{\beta} + \frac{\partial c}{\partial \lambda}(\boldsymbol{u_i},\, \lambda_i)}.
\end{aligned} \tag{4.16}$$

The Newton-Raphson iteration is now complete. To summarize: First, calculate $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ by solving (4.14) and (4.15). Then calculate the increment of the load parameter by using (4.16). The increment in displacements is then obtained by equation (4.13).

4.1.3 Constraints for load- and displacement control

Finally, the constraint function and the parametrisation of the load vector have to be chosen. To get a displacement controlled method as initially stated we choose the constraint function as

$$\boldsymbol{c}(\boldsymbol{u},\, \lambda) = u^{(k)} - \overline{u} \tag{4.17}$$

where u^k is the k-th component of the displacement vector that we want to control and $\overline{u}$ is the prescribed value for it. The partial derivatives are

$$\frac{\partial c}{\partial \boldsymbol{u}} = \boldsymbol{e}_k, \quad \frac{\partial c}{\partial \lambda} = 0,$$

with $\boldsymbol{e}_k$ being the k-th unit vector. If a load controlled method is needed instead, the constraint becomes

$$\boldsymbol{c}(\boldsymbol{u},\, \lambda) = \lambda - \overline{\lambda}, \tag{4.18}$$

with the desired load factor $\overline{\lambda}$. Partial derivatives:

$$\frac{\partial c}{\partial \boldsymbol{u}} = \boldsymbol{0}, \quad \frac{\partial c}{\partial \lambda} = 1,$$

4.1.4 Line search

There is one last extension to the Newton-Raphson method that is needed for our purposes and that is line search.

Todo: Explain line search and why it helps with contact handling

4.1.5 Finding the Braced Equilibrium State

Length of the string...

4.2 Dynamic Solution

The equations of motion together with the initial displacements $\boldsymbol{u}_0$ and velocities $\boldsymbol{v}_0$ of the system form the second-order initial value problem

$$\boldsymbol{M}\ddot{\boldsymbol{u}} + \boldsymbol{q}(\boldsymbol{u}, \ddot{\boldsymbol{u}}) = \boldsymbol{p}, \quad \boldsymbol{u}(0) = \boldsymbol{u}_0, \ \ddot{\boldsymbol{u}}(0) = \boldsymbol{v}_0. \tag{4.19}$$

for the displacements $\boldsymbol{u}(t)$.

Numerical solution methods for initial value problems can be grouped into explicit and implicit ones. Explicit methods use the current state of the system at time t to calculate the next state at $t+\Delta t$. Implicit methods instead find the next state by solving an equation that involves the current and the next state. In the context of finite element analysis, implicit methods require equilibrium iterations at every time step involving the system's tangent stiffness matrix, while explicit methods only require simple vector operations. Implicit methods therefore have a higher computational cost per step, but can make up for it by requiring less steps overall, as their better stability allows them to use much larger step sizes than explicit methods.

Explicit methods are easier to implement and are a good choice for small time scales, like impact and wave propagation problems. They are most effective when the computation of the time steps is fast, so they work well with constant, diagonal (lumped) mass matrices and simple elements with low-order shape functions.

Implicit methods in contrast are well suited for larger time scales, where explicit algorithms would need too many steps. Examples are structural response problems or numerically stiff systems, i.e. systems whose eigenfrequencies range from very small to very large.

VirtualBow currently uses the central difference method [15], which is an explicit method often used in FEM. Based on it's properties and the advantages of explicit methods it seems like a good choice. However, implicit methods have not yet been tested with VirtualBow. The Newmark-beta method Source for example might be worth a try.

VirtualBow previously used the Runge-Kutta-Fehlberg-7-8 method provided by the the odeint library. It performed worse than the central difference method, probably because of the high number of function evaluations carried out by an order 7/8 Runge-Kutta scheme. The big advantage however was the timestep control based on error estimation, which meant that absolutely no user input regarding the timestep was necessary.

4.2.1 The Central Difference Method

The continuous solution $\boldsymbol{u}(t)$ is approximated by the discrete values $\boldsymbol{u}_i = \boldsymbol{u}(t_i)$ at the equidistant time steps $t_i = i \cdot \Delta t, \;\; i \in \mathbb{N}$. The acceleration $\ddot{\boldsymbol{u}}_i$ can be approximated by a central difference quotient as

$$\ddot{\boldsymbol{u}}_i = \frac{\boldsymbol{u}_{i+1} - 2\,\boldsymbol{u}_i + \boldsymbol{u}_{i-1}}{\Delta t^2} \tag{4.20}$$

Solving for $\boldsymbol{u}_{i+1}$ and calculating the accelerations $\ddot{\boldsymbol{u}}_i$ from the equation of motion (4.19) leads to

$$\begin{aligned} \boldsymbol{u}_{i+1} &= 2\,\boldsymbol{u}_i - \boldsymbol{u}_{i-1} + \Delta t^2\,\ddot{\boldsymbol{u}}_i \\ &= 2\,\boldsymbol{u}_i - \boldsymbol{u}_{i-1} + \Delta t^2\,\boldsymbol{M}^{-1}(\boldsymbol{p}_i - \boldsymbol{q}_i). \end{aligned} \tag{4.21}$$

This expression is already the core of the method, it allows the calculation of the next displacements $\boldsymbol{u}_{i+1}$ from the current and previous system states. It only involves cheap vector operations as the mass matrix is diagonal and doesn't have to be "actually" inverted.

Only the velocity is still missing. For this the central difference quotients of the displacements $\boldsymbol{u}_{i+1}$, $\boldsymbol{u}_i$ and $\boldsymbol{u}_{i-1}$ are used to calculate the intermediate velocities

$$\dot{\boldsymbol{u}}_{i+\frac{1}{2}} = \frac{\boldsymbol{u}_{i+1} - \boldsymbol{u}_i}{\Delta t}, \tag{4.22}$$

$$\dot{\boldsymbol{u}}_{i-\frac{1}{2}} = \frac{\boldsymbol{u}_i - \boldsymbol{u}_{i-1}}{\Delta t}. \tag{4.23}$$

The actual new velocity $\dot{\boldsymbol{u}}_{i+1}$ is linearly extrapolated from those,

$$\begin{aligned} \dot{\boldsymbol{u}}_{i+1} &= \dot{\boldsymbol{u}}_{i+\frac{1}{2}} + \frac{1}{2}\left(\dot{\boldsymbol{u}}_{i+\frac{1}{2}} - \dot{\boldsymbol{u}}_{i-\frac{1}{2}}\right), \\ &= \frac{1}{\Delta t}\left(\frac{3}{2}\,\dot{\boldsymbol{u}}_{i+1} - 2\,\dot{\boldsymbol{u}}_i + \frac{1}{2}\,\dot{\boldsymbol{u}}_{i-1}\right). \end{aligned} \tag{4.24}$$

This last step is actually a custom modification as far as I can tell. All descriptions of the central difference method that I've seen end with the calculation of $\dot{\boldsymbol{u}}_{i+\frac{1}{2}}$.

The initialisation of the procedure needs some special attention, as the method uses the previous displacements $\boldsymbol{u}_{i-1}$ that don't exist yet at time $t = 0$. Instead, $\boldsymbol{u}_{-1}$ is estimated by using finite difference approximations of the system's initial velocity and acceleration:

$$\dot{\boldsymbol{u}}_0 = \frac{\boldsymbol{u}_1 - \boldsymbol{u}_{-1}}{2\,\Delta t}, \tag{4.25}$$

$$\ddot{\boldsymbol{u}}_0 = \frac{\boldsymbol{u}_1 - 2\,\boldsymbol{u}_0 + \boldsymbol{u}_{-1}}{\Delta t^2}, \tag{4.26}$$

$$\Rightarrow \boldsymbol{u}_{-1} = \boldsymbol{u}_0 - \Delta t\,\dot{\boldsymbol{u}}_0 + \frac{\Delta t^2}{2}\,\ddot{\boldsymbol{u}}_0. \tag{4.27}$$

A last but very important question is how to choose the timestep Δt. The timestep influences the computational effort, the accuracy of the solution but most importantly the stability of the algorithm. A too large time step will lead to a solution that "explodes" numerically. For linear, conservative mechanical systems of the form

$$\boldsymbol{M}\ddot{\boldsymbol{u}} + \boldsymbol{K}\boldsymbol{u} = \boldsymbol{p} \tag{4.28}$$

the stability condition is

$$\Delta t \leq \frac{2}{\omega_{max}} \tag{4.29}$$

where ω_{max} is the maximum natural frequency [15] of the system. Unfortunately there is no such analytical bound on Δt that guarantees stability for nonlinear systems. A common thing to do is to linearise the system around the current configuration and apply (4.29) anyway. A safety factor $0 < \beta < 1$ is introduced to account for the uncertainties and the timestep is set to

$$\Delta t = \beta \frac{2}{\omega_{max}}. \tag{4.30}$$

This doesn't guarantee stability in a strict sense, but it's okay for practical purposes. A common choice is $\beta = 0.9$ but VirtualBow uses a lower value for the sake of robustness. The frequency ω_{max} is sometimes estimated by using the bound

$$\omega_{max} \leq \max_i \{\omega_{max,i}\} \tag{4.31}$$

where $\omega_{max,i}$ is the maximum natural frequency of the respective finite element i. But this leads to larger time steps than necessary and actually causes more implementation effort than just solving the generalised eigenvalue problem

$$\left(\boldsymbol{M} - \omega^2 \boldsymbol{K}\right) \bar{\boldsymbol{u}} = 0 \tag{4.32}$$

which is what VirtualBow does.

No damping here. For a version with damping see the LS-Dyna theoretical manual of 2006.

4.2.2 Stopping Criterion and Progress Estimation

The dynamic simulation is carried out until some stopping criterion is satisfied. This criterion has to be chosen such that the simulated time interval contains all the information needed by the users while also keeping it as short as possible. It has to include the separation of the arrow from the string plus some additional time, because some things like e.g. the maximum force on the string tend to happen only after the arrow left the bow. A related requirement on the stopping criterion is that it must allow estimating the simulation progress, so that it can be shown to the user.

The simplest solution would be to simulate until a fixed time is reached. This makes the progress estimation very easy, but it also has some drawbacks. It would require users to specify the simulation time, which can only be found by trial and error and might have to be adjusted for different bows.

The current approach is to simulate the time interval $[0,\, \alpha\, T]$ where T is the time for the arrow to reach brace height and α is an arbitrary positive factor. The default value is $\alpha = 1.5$. The simulation progress at time $\bar{t}$ can now be written as

$$\text{progress} = \frac{\bar{\text{t}}}{\alpha\, \text{T}} \cdot 100\%. \tag{4.33}$$

But only formally, because T is not yet known as long as $\bar{t} < T$, i.e. as long as the arrow has not yet actually reached brace height. For this first part of the simulation T is estimated by using the current time $\bar{t}$, arrow position $\bar{u}$ and arrow velocity $\bar{v}$ and extrapolating to the time when the arrow will reach the braced position u_T. Assuming constant velocity, which is a very crude approximation, we can extrapolate the arrow position as

$$u(t) = \bar{u} + \bar{v}(t - \bar{t}). \tag{4.34}$$

Now T can be calculated from by using the condition $u(T) = u_T$, with results in

$$T = \frac{u_T - \bar{u}}{\bar{v}} + \bar{t}. \tag{4.35}$$

This approximation is far from exact, but it is good enough for progress estimation and gets better the closer the arrow is to the braced position. After the arrow has reached the braced position, the actual value of T is known and used for the rest of the simulation.

4.3 Model setup

4.3.1 Finding the damping parameters

The damping parameter ηA of the string is determined analytically such that the damping ratio associated with the first (longitudinal) natural frequency matches the user supplied value. From Appendix A it follows that

$$\eta A = \frac{4L}{\pi}\sqrt{\rho A\, EA}\,\zeta \tag{4.36}$$

in order for this to be true, where L is half the total length of the string to match the boundary conditions used.

4.3.2 Distributing string elements

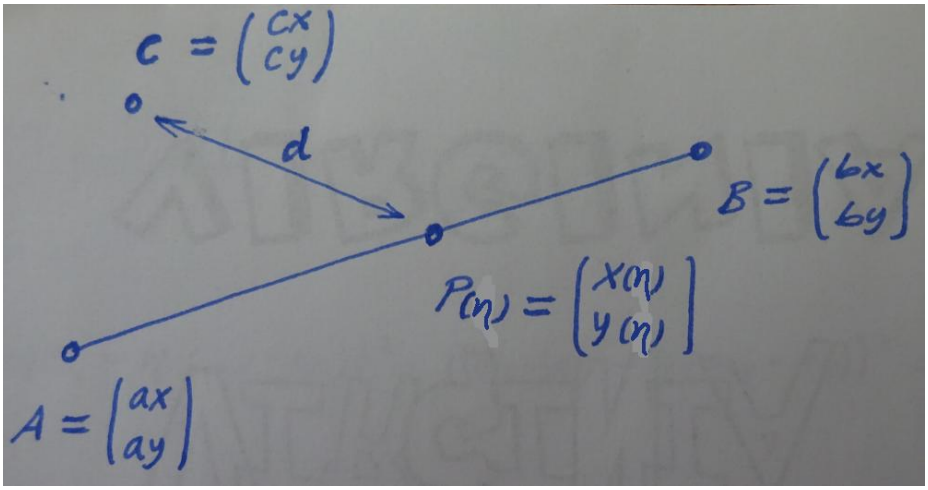

Figure 4.1: Problem

Problem: Find a point P on the line between A and B which has the euclidean distance d to point C.

Parametrisation of the points P:

$$P(\eta) = A + \eta\,(B - A), \quad \eta \in [0,\, 1]$$

Distance condition:

$$
\begin{aligned}
&\|C - P(\eta)\| = d \\
\Leftrightarrow\ &\| \begin{bmatrix} c_x - a_x - \eta\,(b_x - a_x) \\ c_y - a_y - \eta\,(b_y - a_y) \end{bmatrix} \| = d \\
\Leftrightarrow\ &\left(c_x - a_x - \eta\,(b_x - a_x)\right)^2 + \left(c_y - a_y - \eta\,(b_y - a_y)\right)^2 = d^2 \\
\Leftrightarrow\ &\left((b_x - a_x)^2 + (b_y - a_y)^2\right)\,\eta^2 - 2\left((c_x - a_x)(b_x - a_x) + (c_y - a_y)(b_y - a_y)\right)\,\eta \\
&+ (c_x - a_x)^2 + (c_y - a_y)^2 - d^2 = 0
\end{aligned}
$$

This is a quadratic equation for η. Take smallest solution in $[0,\,1]$.

Chapter 5

Model Validation

A very important task in the development of this software is to make sure that the results obtained by simulation agree reasonably well with real world examples. This section shows the validation efforts made so far. It is still very sparse, so if you have used this program for a real world application, let me know about your results and they will be added here.

Statics of a straight steel bow

In this experiment the draw curve and limb shapes of a small steel bow with a constant cross section have been measured and compared to version 2014.4 of Bow Simulation Tool (now VirtualBow). The bow is shown in figure 5.1 and has been made from an old saw blade. Steel is a good material for this kind of test because of its homogenous mechanical properties. The elastic modulus was assumed to be $E = 210\ \mathrm{GPa}$ which is a good estimate for most types of steel.

Figure 5.1: Steel bow. Cross section: 16.85 $\times$ 0.75mm. Length: 269mm. Brace heighth: 49.8mm

The experiment was carried out by hanging a plastic bag at the string center. The draw force was then gradually applied by counting steel balls with a known mass into the bag. After every load step the draw length was measured and a photo of the bow was taken.

Figure 5.2 shows a comparison between the measured and the simulated draw curve and figure 5.3 compares the pictures of the limb against the simulated limb shapes.

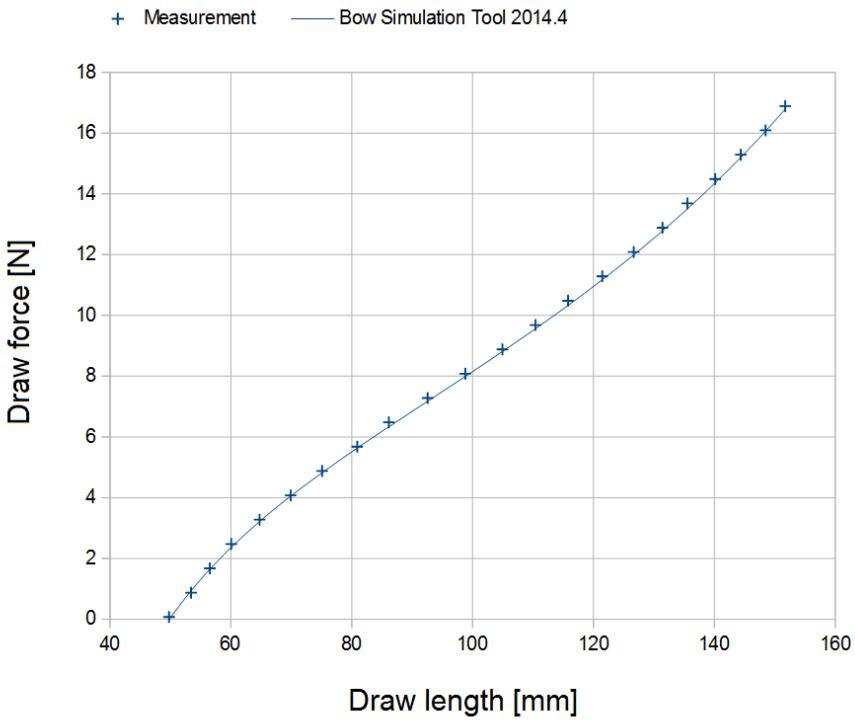

Figure 5.2: Experimental and simulated draw curves

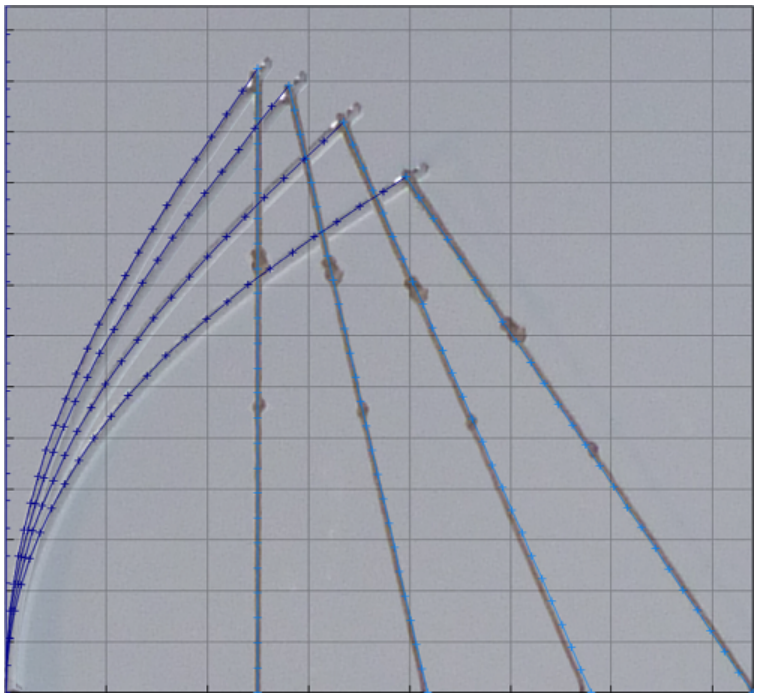

Figure 5.3: Experimental and simulated limb shapes

The agreement between experiment and simulation is surprisingly good here. It really shows the potential of such kinds of simulations, provided that the material properties are well known and a the bow can be built exactly as simulated, with low tolerances.

But this is still a very simple example. The next step would be to repeat this experiment with bows that have varying cross sections and non-straight profiles. Another open question are the dynamic simulation results. It's unclear if they can match experiments as good as the static results do, because there are much more uncertainties involved.

Appendix A

Vibrations of a viscoelastic bar

Consider the elastic bar shown in figure A.1. It is fixed on the left hand side and free on the right, has a length of L and can undergo longitudinal motion as described by the function $u(x, t)$. We want to calculate its (undamped) natural frequencies ω_0 and associated damping ratios ζ.

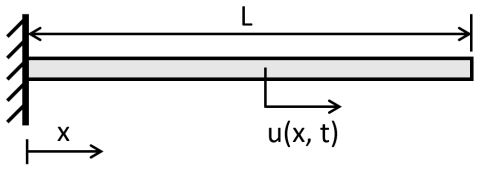

Figure A.1: Kinematics and boundary conditions of the bar

The material of the bar is assumed to behave according to the viscoelastic Kelvin-Voigt model, which expresses the stress σ in the bar as

$$\sigma = E\,\varepsilon + \eta\,\dot{\varepsilon} \tag{A.1}$$

where $\varepsilon = \frac{\partial u}{\partial x}$ is the longitudinal strain, E the elastic modulus and η the viscosity of the material. The acceleration at every point in the bar is given by Source?

$$\rho\,\frac{\partial^2 u}{\partial t^2} = \frac{\partial \sigma}{\partial x} \tag{A.2}$$

where ρ is the bar's density. Combining the equations (A.1) and (A.2) and multiplying with the cross section area A results in the equation of motion

$$\rho A\,\frac{\partial^2 u}{\partial t^2} = EA\,\frac{\partial^2 u}{\partial x^2} + \eta A\,\frac{\partial^3 u}{\partial x^2\,\partial t}. \tag{A.3}$$

This partial differential equation can be solved by separation of variables. The solution is assumed as $u(x,\,t) = X(x)\,T(t)$, a product of two functions depending on x and t respectively. Inserting this into the equation of motion (A.3) and separating terms depending on x from terms depending on t leads to

$$\begin{aligned} \rho A\, X(x)\,\ddot{T}(t) &= EA\, X''(x)\,T(t) + \eta A\, X''(x)\,\dot{T}(t) \\ \frac{X''(x)}{X(x)} &= \frac{\ddot{T}(t)}{\frac{EA}{\rho A}\,T(t) + \frac{\eta A}{\rho A}\,\dot{T}(t)} =: -\frac{\rho A}{EA}\,\omega_0^2 \end{aligned} \tag{A.4}$$

Apply the usual argument: As one side of the equation depends on position and the other on time they have to be constant in order to be equal. We choose that constant to be $-\frac{\rho A}{EA}\,\omega_0^2$ and obtain the two ordinary differential equations

$$\ddot{T}(t) + \frac{\eta A}{EA}\,\omega_0^2\,\dot{T}(t) + \omega_0^2\,T(t) = 0, \tag{A.5}$$

$$X''(x) + \frac{\rho A}{EA}\,\omega_0^2\,X(x) = 0. \tag{A.6}$$

The time equation (A.5) describes a damped harmonic oscillator with undamped natural frequency ω_0 and damping ratio

$$\zeta = \frac{1}{2}\,\frac{\eta\,A}{EA}\,\omega_0 \tag{A.7}$$

as determined by the coefficients of the equation. The last remaining task is to determine the natural frequencies ω_0 which solve the second differential equation (A.6) while also satisfying the boundary conditions of the bar. The general solution of (A.6) is

$$X(x) = C \cdot \sin\left(\sqrt{\frac{\rho A}{EA}}\,\omega_0\,x - \varphi\right). \tag{A.8}$$

Boundary condition on the fixed end:

$$\begin{aligned} u(0,\,t) = 0: \qquad X(0)\,T(t) &= 0 \\ C \cdot \sin(-\varphi) &= 0 \\ \varphi &= 0 \end{aligned} \tag{A.9}$$

The case $T(t) = 0$ was ignored here because it is of no interest. The boundary condition for the free end is

$$\begin{aligned} \sigma(L,\,t) = 0: \qquad E\,\varepsilon(L,\,t) + \eta\,\dot{\varepsilon}(L,\,t) &= 0 \\ EA\,\frac{\partial u}{\partial x}(L,\,t) + \eta A\,\frac{\partial^2 u}{\partial x\,\partial t}(L,\,t) &= 0 \\ EA\,X'(L)\,T(t) + \eta A\,X'(L)\,\dot{T}(t) &= 0 \\ (EA\,T(t) + \eta A\,\dot{T}(t))\,X'(L) &= 0 \\ \ddot{T}(t)\,X'(L) &= 0 \end{aligned}$$

And therefore the solution for the natural frequencies is

$$X'(L) = C \cdot \sqrt{\frac{\rho A}{EA}}\,\omega_0 \cos\left(\sqrt{\frac{\rho A}{EA}}\,\omega_0\, L\right) = 0,$$

$$\omega_0 = \frac{\pi(2k-1)}{2L} \cdot \sqrt{\frac{EA}{\rho A}}, \quad k \in \mathbb{Z}. \tag{A.10}$$

And finally, with equations (A.7) and (A.10) the damping ratio follow as

$$\zeta = \frac{\pi(2k-1)}{4L} \cdot \frac{\eta A}{\sqrt{\rho A \cdot EA}}, \quad k \in \mathbb{Z}. \tag{A.11}$$

Appendix B

Cubic Spline Interpolation

Cubic splines are used to define some of the geometric features of the bow, namely the profile curve, width and layer heights.

B.1 Cubic C^2 Spline

Consider n points $(x_0,\, y_0) \ldots (x_{n-1},\, y_{n-1})$ with increasing values of x. We define the differences $\Delta x_i = x_{i+1} - x_i$ and $\Delta y_i = y_{i+1} - y_i$ as well as the slope $\delta_i = \Delta y_i / \Delta x_i$ of the secant line between two successive points.

For each interval $x \in [x_i,\, x_{i+1}]$ the data is interpolated by the cubic polynomial

$$f_i(x) = h_{00}(t)\, y_i + h_{10}(t) \Delta x_i\, m_i + h_{01}(t)\, y_{i+1} + h_{11}(t) \Delta x_i\, m_{i+1} \tag{B.1}$$

$$f_i'(x) = \frac{h_{00}'(t)}{\Delta x_i}\, y_i + h_{10}'(t)\, m_i + \frac{h_{01}'(t)}{\Delta x_i}\, y_{i+1} + h_{11}'(t)\, m_{i+1} \tag{B.2}$$

$$f_i''(x) = \frac{h_{00}''(t)}{\Delta x_i^2}\, y_i + \frac{h_{10}''(t)}{\Delta x_i}\, m_i + \frac{h_{01}''(t)}{\Delta x_i^2}\, y_{i+1} + \frac{h_{11}''(t)}{\Delta x_i}\, m_{i+1} \tag{B.3}$$

where y_i, y_{i+1} and m_i, m_{i+1} are the values and slopes at the interval boundaries, respectively and $h_{00}(t) \ldots h_{11}(t)$ are the hermite basis functions, defined as

$$\begin{array}{lll} h_{00}(t) = 2\,t^3 - 3\,t^2 + 1 & h_{00}'(t) = 6\,t^2 - 6\,t & h_{00}''(t) = 12\,t - 6 \\ h_{10}(t) = t^3 - 2\,t^2 + t & h_{10}'(t) = 3\,t^2 - 4\,t + 1 & h_{10}''(t) = 6\,t - 4 \\ h_{01}(t) = -2\,t^3 + 3\,t^2 & h_{01}'(t) = -6\,t^2 + 6\,t & h_{01}''(t) = -12\,t + 6 \\ h_{11}(t) = t^3 - t^2 & h_{11}'(t) = 3\,t^2 - 2\,t & h_{11}''(t) = 6\,t - 2 \end{array} \tag{B.4}$$

with $t = (x - x_i)/(x_{i+1} - x_i)$. The resulting spline by definition already interpolates the given function values and is C^1 continuous since adjacent segments share the same value and slope at the interval boundaries. To determine the n unknown slopes m_i we require the second derivatives at the connection points to match as well, making the curve C^2

continuous. This gives us $n-2$ continuity conditions, leaving two more conditions to make the problem uniquely determined. Those additional conditions are obtained from boundary considerations, e.g. by specifying the spline's first and/or second derivatives at the left and right endpoints.

Continuity conditions $f''_{i-1}(x_i) = f''_i(x_i)$ for $i = 1 \ldots n-2$:

$$\begin{aligned}
&\frac{h''_{00}(1)}{\Delta x_{i-1}^2}\, y_{i-1} + \frac{h''_{10}(1)}{\Delta x_{i-1}}\, m_{i-1} + \frac{h''_{01}(1)}{\Delta x_{i-1}^2}\, y_i + \frac{h''_{11}(1)}{\Delta x_{i-1}}\, m_i = \frac{h''_{00}(0)}{\Delta x_i^2}\, y_i + \frac{h''_{10}(0)}{\Delta x_i}\, m_i + \frac{h''_{01}(0)}{\Delta x_i^2}\, y_{i+1} + \frac{h''_{11}(0)}{\Delta x_i}\, m_{i+1} \\
&\frac{-2}{\Delta x_i}\, m_{i+1} + \left(\frac{-4}{\Delta x_i} - \frac{4}{\Delta x_{i-1}}\right) m_i - \frac{2}{\Delta x_{i-1}}\, m_{i-1} = \frac{6\, y_{i-1} + -6\, y_i}{\Delta x_{i-1}^2} - \frac{-6\, y_i + 6\, y_{i+1}}{\Delta x_i^2} \\
&\frac{1}{\Delta x_i}\, m_{i+1} + 2\left(\frac{1}{\Delta x_i} + \frac{1}{\Delta x_{i-1}}\right) m_i + \frac{1}{\Delta x_{i-1}}\, m_{i-1} = 3\left(\frac{\Delta\, y_i}{\Delta x_i^2} + \frac{\Delta y_{i-1}}{\Delta x_{i-1}^2}\right) \\
&\Delta x_{i-1}\, m_{i+1} + 2\left(\Delta x_i + \Delta x_{i-1}\right)\, m_i + \Delta x_i\, m_{i-1} = 3\left(\delta_i\, \Delta x_{i-1} + \delta_{i-1}\, \Delta x_i\right)
\end{aligned} \tag{B.5}$$

Boundary condition $f''_0(0) = y''_0$:

$$\begin{aligned}
\frac{h''_{00}(0)}{\Delta x_0^2}\, y_0 + \frac{h''_{10}(0)}{\Delta x_0}\, m_0 + \frac{h''_{01}(0)}{\Delta x_0^2}\, y_1 + \frac{h''_{11}(0)}{\Delta x_0}\, m_1 &= y''_0 \\
\frac{-6}{\Delta x_0^2}\, y_0 + \frac{-4}{\Delta x_0}\, m_0 + \frac{6}{\Delta x_0^2}\, y_1 + \frac{-2}{\Delta x_0}\, m_1 &= y''_0 \\
4\, m_0 + 2\, m_1 &= 6\, \delta_0 - \Delta x_0\, y''_0
\end{aligned} \tag{B.6}$$

Boundary condition $f''_{n-2}(1) = y''_{n-1}$:

$$\begin{aligned}
\frac{h''_{00}(1)}{\Delta x_{n-2}^2}\, y_{n-2} + \frac{h''_{10}(1)}{\Delta x_{n-2}}\, m_{n-2} + \frac{h''_{01}(1)}{\Delta x_{n-2}^2}\, y_{n-1} + \frac{h''_{11}(1)}{\Delta x_{n-2}}\, m_{n-1} &= y''_{n-1} \\
\frac{6}{\Delta x_{n-2}^2}\, y_{n-2} + \frac{2}{\Delta x_{n-2}}\, m_{n-2} + \frac{-6}{\Delta x_{n-2}^2}\, y_{n-1} + \frac{4}{\Delta x_{n-2}}\, m_{n-1} &= y''_{n-1} \\
2\, m_{n-2} + 4\, m_{n-1} &= 6\, \delta_{n-2} + \Delta x_{n-2}\, y''_{n-1}
\end{aligned} \tag{B.7}$$

Boundary condition $f'_0(0) = y'_0$:

$$m_0 = y'_0 \tag{B.8}$$

Boundary condition $f'_{n-1}(1) = y'_{n-1}$:

$$m_{n-1} = y'_{n-1} \tag{B.9}$$

Combining the continuity and boundary conditions gives us the system of equations

$$\begin{bmatrix} a_0 & b_0 & & & & \\ \ddots & \ddots & \ddots & & & \\ & \Delta x_{i-1} & 2(\Delta x_i + \Delta x_{i-1}) & \Delta x_i & & \\ & & \ddots & \ddots & \ddots & \\ & & & a_{n-1} & b_{n-1} \end{bmatrix} \begin{bmatrix} m_0 \\ \vdots \\ m_i \\ \vdots \\ m_{n-1} \end{bmatrix} = \begin{bmatrix} c_0 \\ \vdots \\ 3\left(\delta_i \, \Delta x_{i-1} + \delta_{i-1} \, \Delta x_i\right) \\ \vdots \\ c_{n-1} \end{bmatrix} \tag{B.10}$$

where a, b and c depend on the choice of boundary conditions. Since the coefficient matrix has a tridiagonal structure it can be solved very efficiently by using the Thomas algorithm.

B.2 Monotonicity

Sometimes it is desirable for an interpolating curve to preserve monotonicity of the input data, i.e. not to introduce any local minima or maxima between the given data points. In VirtualBow this is used for the width and layer heights, not least to ensure that those quantities are always positive if the user entered positive values.

One popular method for monotonic cubic interpolation is the Fritsch–Carlson method[17], which operates on a cubic hermite spline like the one we defined above and adjusts the slopes according to some simple rules in order to make the spline monotonic within each interval. The only difference is that Fritsch and Carlson use finite differences to determine the initial slopes for their spline while we used the condition of C^2 continuity.

After the initial slopes were computed, the first necessary condition for monotonicity in an interval is

$$\operatorname{sgn}(m_i) = \operatorname{sgn}(m_{i+1}) = \operatorname{sgn}(\delta_i) \tag{B.11}$$

which means that the slopes m_i and m_{i+1} must have the same direction as the slope δ_i of the secant. Any slopes that violate this condition are set to zero. With the slopes modified in this way, define $\alpha_i = m_i/\delta i$ and $\beta_i = m_{i+1}/\delta i$. A sufficient condition for the segment to be monotone is

$$\alpha_i^2 + \beta_i^2 \leq 9 \tag{B.12}$$

i.e. the curve must be restricted to a circle of radius 3 in the α-β-plane. If this condition is violated in any segment, define the factor

$$\tau_i = \frac{3}{\sqrt{\alpha_i^2 + \beta_i^2}} \tag{B.13}$$

and rescale the slopes as

$$m_i = \tau_i \, \alpha_i \, \delta_i, \quad m_{i+1} = \tau_i \, \beta_i \, \delta_i \tag{B.14}$$

After these modifications the spline will be monotonous within each interval, but may no longer be C^2 continuous. Also the chosen boundary conditions may no longer be fulfilled.

Bibliography

[1] C.N. Hickman: *The Dynamics of A Bow and Arrow.* Journal of Applied Physics, Vol. 8, No. 6, page 404, 1937.

[2] W. C. Marlow: *Bow and arrow dynamics.* The Perkin-Elmer Corporation, Ridgefield, Connecticut 06877, 1980. (`http://sci-toys.com/bow_and_arrow.pdf`)

[3] B.W. Kooi, J.A. Sparenberg: *On the static deformation of a bow.* Journal of Engineering Mathematics, Vol. 14, No. 1, pages 27-45, 1980. (`http://www.bio.vu.nl/thb/users/kooi/kosp80.pdf`)

[4] B.W. Kooi: *On the mechanics of the bow and arrow.* Journal of Engineering Mathematics, Vol. 15, No. 2, pages 119-145, 1981. (`http://www.bio.vu.nl/thb/users/kooi/kooi81.pdf`)

[5] B.W. Kooi: *On the Mechanics of the Modern Working-Recurve Bow.* Computational Mechanics, Vol. 8, No. 5, pages 291-304, 1991. (`http://www.bio.vu.nl/thb/users/kooi/kooi91.pdf`)

[6] B.W. Kooi: *On the Mechanics of the Arrow: Archer's Paradox.* Journal of Engineering Mathematics, Vol. 31, No. 4, pages 285-303, 1997. (`http://www.bio.vu.nl/thb/users/kooi/kosp97.pdf`)

[7] Louie L. Yaw: *2D Corotational Beam Formulation.* Walla Walla University, 30.11.2009. (`http://people.wallawalla.edu/~louie.yaw/Co-rotational_docs/2Dcorot_beam.pdf`)

[8] *Standard Mass Matrices For Plane Beam Elements.* Matrix Finite Element Methods in Dynamics, Chapter 18, University of Colorado at Boulder. (Link)

[9] E. Marotta, P. Salvini: *Analytical Stiffness Matrix for Curved Metal Wires.* Procedia Structural Integrity, Vol. 8, pages 43-55, 2018. (Link)

[10] D. Gross, W. Hauger, J. Schröder, W.A. Wall: *Technische Mechanik 2 - Elastostatik.* (In German.) Springer-Verlag Berlin Heidelberg, 2011.

[11] D. Gross, W. Hauger, P. Wriggers: *Technische Mechanik 4 - Hydromechanik, Elemente der Höheren Mechanik, Numerische Methoden.* (In German.) Springer-Verlag Berlin Heidelberg, 2011.

[12] Wikipedia: *Engineering sandwich beam theory.* (`https://en.wikipedia.org/wiki/Sandwich_theory#Engineering_sandwich_beam_theory`). Accessed on 25th December 2017.

[13] Wikipedia: *Lagrangian mechanics.* (`https://en.wikipedia.org/wiki/Lagrangian_mechanics`). Accessed on 3rd January 2018.

[14] Klaus-Jürgen Bathe: *Finite Element Procedures for Solids and Structures - Solution of the Nonlinear Finite Element Equations in Static Analysis - Part I*. MIT OpenCourseWare. (Link)

[15] Klaus-Jürgen Bathe: *Finite Element Procedures for Solids and Structures - Solution of Nonlinear Dynamic Response - Part I.* MIT OpenCourseWare. (Link)

[16] D. Kuhl, G. Meschke: *Finite Elemente Methoden I & II. Lehrstuhl für Statik und Dynamik.* (In German.), Ruhr-Universität Bochum, 2002.

[17] F.N. Fritsch, R.E. Carlson: *Monotone Piecewise Cubic Interpolation*. SIAM Journal on Numerical Analysis, Vol 17, No. 2, pages 238-246, 1980.