VirtualBow v0.7

User Manual

Contents

1 Introduction

This is the user manual for VirtualBow, a software tool for bow and arrow physics simulation.

What is it intended for?

What kinds of bows is it geared towards

Reference the Theory Manual

Encouraging Feedback and Support

Components and filetypes

Compatibility of files, automatic overwriting in new versions

For the latest version of the software and this manual visit the project's website at
http://www.virtualbow.org/.

2   Model Editor

2.1   Overview

The bow editor is the main window of the application. Here you can load, edit and save bow models and start static and dynamic simulations.
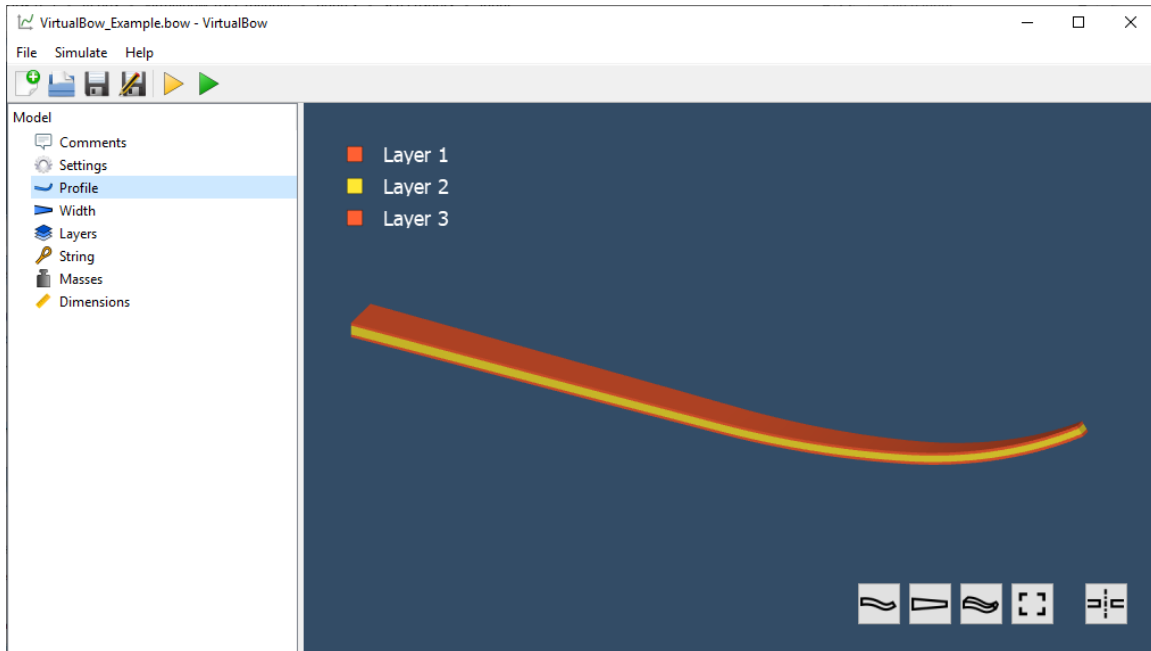

Figure 1: Bow Editor

Double-click any of the items on the left to edit the respective model properties. The 3D view on the right shows the current limb geometry. Use the mouse to rotate (left mouse button), shift (middle mouse button) and zoom the view. More view options are available through the buttons on the bottom-right corner.

2.2 Parameters

2.2.1 Profile

The profile curve is the geometric centerline of the bow limbs in unbraced state. Use the table on the left to edit the parameters. The result is shown on the plot on the right.
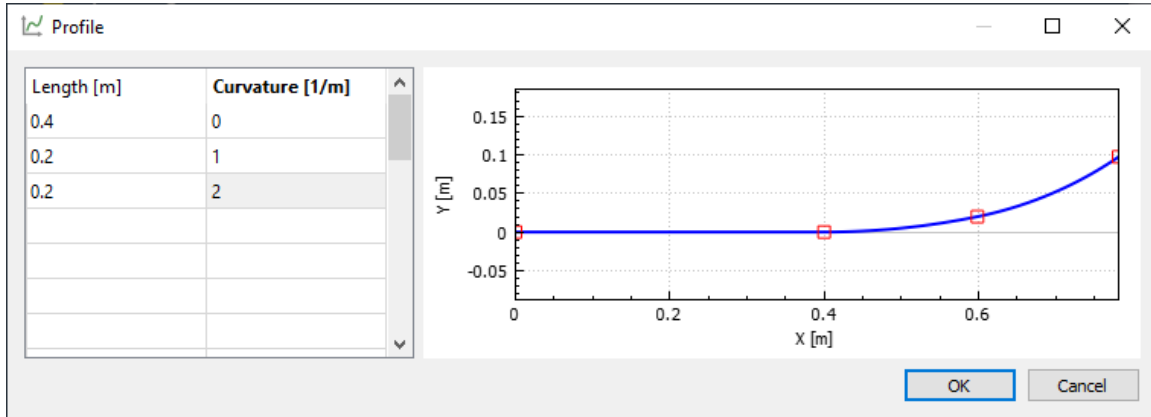

Figure 2: Profile dialog

The profile curve consists of a series of arc segments that each have a certain length and curvature. Each row in the table defines one such segment.

Note: Mathematically, the curvature κ of a segment is the reciprocal of its radius r, so you can calculate the curvature via $\kappa = \frac{1}{r}$ if you know the radius and vice versa.

2.2.2 Width

The width dialog is used to define the limb's width along its profile curve. This width is the same for all layers of the bow.
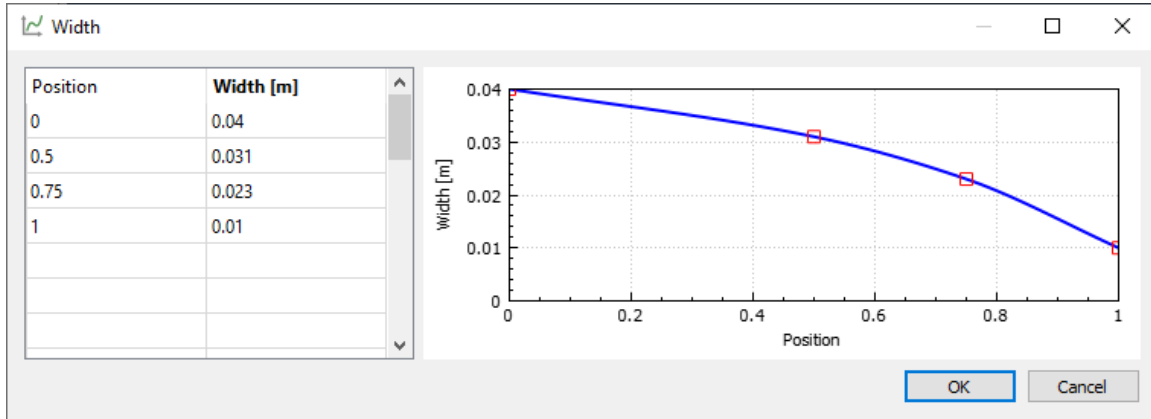
Figure 3: Width dialog

On the table on the left you can specify values for the width at certain relative positions (between 0 and 1) along the limb's profile curve. This definition of cross section properties relative to the total length of the profile curve makes it possible to change the profile without having to adjust any cross sections.

The actual width distribution of the limb is constructed as a smooth curve (monotonic cubic spline) passing through the supplied values as shown on the plot on the right.

2.2.3 Layers

With the layer dialog you can create any number of layers and specify their height/thickness and material properties.
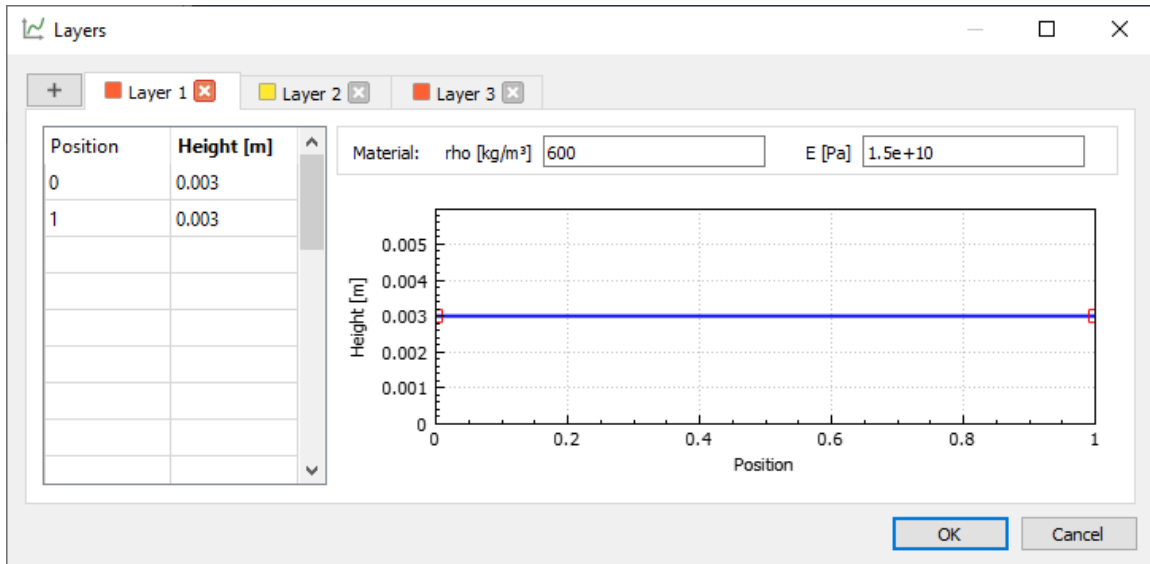

Figure 4: Layer dialog

Click the plus button on the top left to add layers. You can later rearrange them by dragging the tabs. Double-click on a tab to rename a layer.

The table on the left sets the height distribution of the layer. It works the same way as the limb's width: You specify a number of values at different relative positions and the program creates an interpolating curve.

The layer material is specified by the following two constants,

- **rho:** Density (Mass per unit volume)

- **E:** Elastic modulus (Measure for the stiffness of a solid material)

For manufactured materials like e.g. fiberglass or steel you might find those numbers in a datasheet provided by the manufacturer.

Wood is a bit more difficult, because the properties can vary quite a bit even within the same type. You can find average numbers on the internet, for example at `http://www.wood-database.com`. Those are probably good enough as a first reference. However, in order to be really sure about a specific piece of wood there is no other way than to test it. One possibility is a bending test as shown in Appendix D.

2.2.4 String

Here you can define the mechanical properties of the string by providing data for the string material and the number of strands being used.
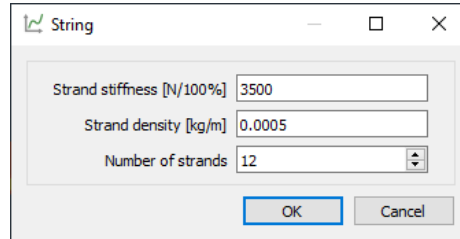

Figure 5: String dialog

- **Strand density:** Linear density of the strands (mass per unit length)
- **Strand stiffness:** Stiffness of the strands against elongation (force per unit strain)
- **Number of strands:** Total number of strands in the string

The linear density of a string material can be easily determined with a kitchen scale (weight divided by length), but the stiffness is much more difficult to obtain. Table 1 shows some reference values taken from the SuperTiller V6.6 Excel spreadsheet by Alan Case[1].

Note: The stiffness of the string material is an important parameter in dynamic analysis. The static results however aren't affected very much by it as long as it is high enough to prevent significant elongation.
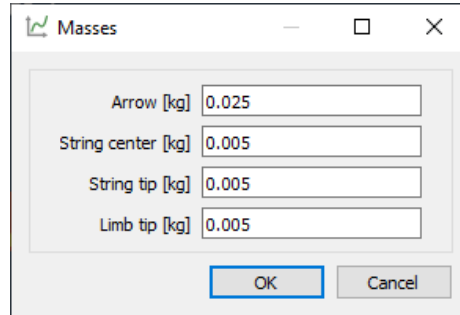
Material	Stiffness $[\text{N}/_{100\%}]$	Density $[^{kg}/_m]$	Breaking strength [N]	Source/Comment
Dacron B50	3113.76	0.000333	217.96	BCY assuming 7% elongation at break (linearized)
Fast Flight	14086.04	0.000182	422.58	BCY assuming 3% elongation at break (linearized)
Dyneema	18860.46	0.000160	578.27	Calculated assuming linear stress-strain (to break)
Linen 40/3	1668.08	0.0642	49.38	Maurice Taylor, Archery The Technical Side, 1947
Silk	1026.51	0.0930	65.83	Maurice Taylor, Archery The Technical Side, 1947

Table 1: Properties of common bowstring materials according to SuperTiller V6.6

---

[1] http://www.buildyourownbow.com/build-alongs/how-to-use-supertiller-build-along/

2.2.5 Masses

This dialog is used to define the various masses of the bow. All of them except the arrow mass are optional.

Figure 6: Masses dialog

- **Arrow:** Mass of the arrow
- **String center:** Additional mass at the string center (serving, nocking point)
- **String tip:** Additional mass at the ends of the string (serving, silencers)
- **Limb tip:** Additional mass at the limb tip (nocks, overlays)

2.2.6 Damping

Damping can be used to adjust for the energy a bow loses by dissipation, for example due to internal friction/hysteresis of the materials being used. This dialog therefore allows setting a damping ratio for the limbs and string, respectively.
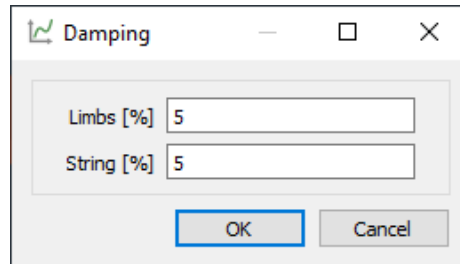
Figure 7: Damping dialog

Note: The damping ratio characterizes how quickly oscillations in a system decay over time. A system with a damping ratio of 0% is undamped, it doesn't dissipate any energy and just keeps going with a constant amplitude. The higher the damping ratio the faster the amplitudes decay over time, losing energy with every oscillation. Once the damping ratio reaches 100% the system stops oscillating (no more overshoot), this is called critical damping.
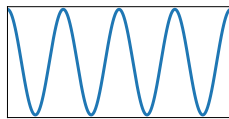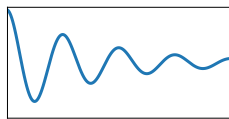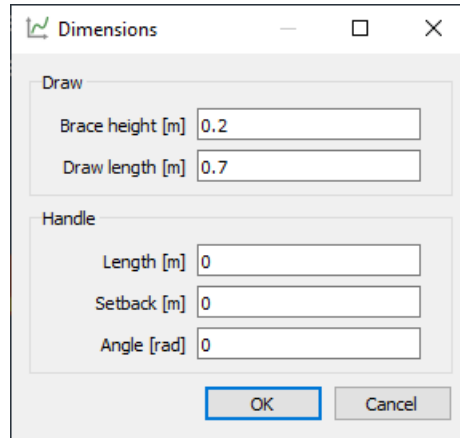
Damping ratio	Amplitude
0%	
10%	
100%	

Table 2: Oscillations with increasing damping ratio

The damping ratios for a bow's limbs and string are mostly empirical. Realistic values are probably in the range of 0 - 20%, but there is still more practical experience needed.

2.2.7 Dimensions

The parameters listed in this dialog define the lengths and angles of the bow and its optional middle section/grip. See also figure 9 for a visual explanation of those parameters.

Figure 8: Dimensions dialog

- Draw

  - **Brace height:** Distance of the string center to the coordinate origin in braced state

  - **Draw length:** Distance of the string center to the coordinate origin in fully drawn state

- Handle

  - **Length:** Length of the riser or stiff middle section

  - **Setback:** Depth of the riser or stiff middle section

  - **Angle:** Angle of the riser or stiff middle section
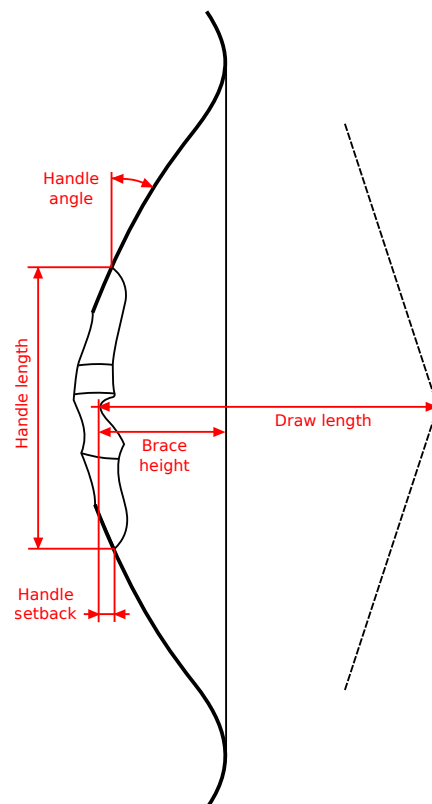
Figure 9: Dimensions of the bow

2.2.8 Comments

The comments are meant for documenting the bow model. Any notes about the bow and the simulation results can be added here.
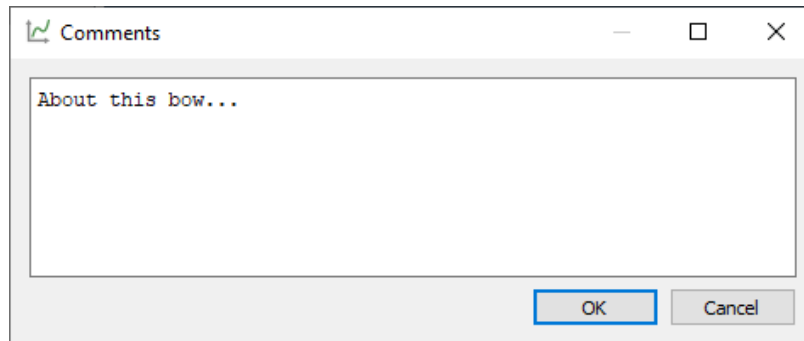
Figure 10: Comments dialog

2.2.9 Settings

These are numerical settings that can be used to tweak the simulation. Most of the time the default values should be fine though. However, as the default values favor accuracy and realiability over performance there might be use cases where finding faster settings is worth it. Think about running a large number of simulations with a script for example.
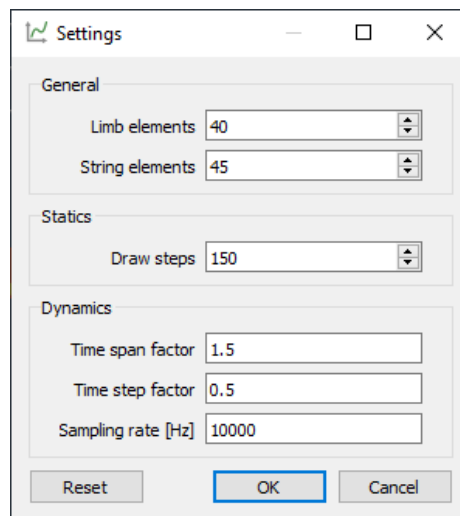
Figure 11: Settings dialog

These are the individual settings:

- **General**

  - **Limb elements:** Number of finite elements that are used to approximate the limb. More elements increase the accuracy but also the computing time.

  - **String elements:** Same as above.

- **Statics**

  - **Draw steps:** Number of steps that are performed by the static simulation from brace height to full draw. This determines the resolution of the static results. You can usually decrease this value to speed up the simulation.

- **Dynamics**

  - **Time span factor:** This controls the time period that is simulated. A value of 1 corresponds to the time at which the arrow reaches brace height. The default value is larger than that in order to capture some of the things that occur after the arrow left the bow (maximum dynamic loads on limb and string).

  - **Time step factor:** When carrying out the dynamic simulation the program will repeatedly use the current state of the bow at time t to calculate the next state at time $t + \Delta t$ where Δt is some small timestep. This timestep has to be chosen small enough to get an accurate and stable solution but also as large as possible to keep the number of steps low. The program can estimate this, but to be on the safe side the estimation is multiplied with a safety factor between 0 and 1 that you can choose here.

  - **Sampling rate:** Limits the time resolution of the output data. This is done because the dynamic simulation usually produces much finer grained data than is actually useful. Not including all of that in the final output saves time and memory.

3 Result Viewer

3.1 Overview

General: Slider, jumping to events, switching statics/dynamics

You can start static or dynamic simulations by using the yellow and green toolbar buttons or the simulation menu. A new window with the results will open once the simulation has finished.
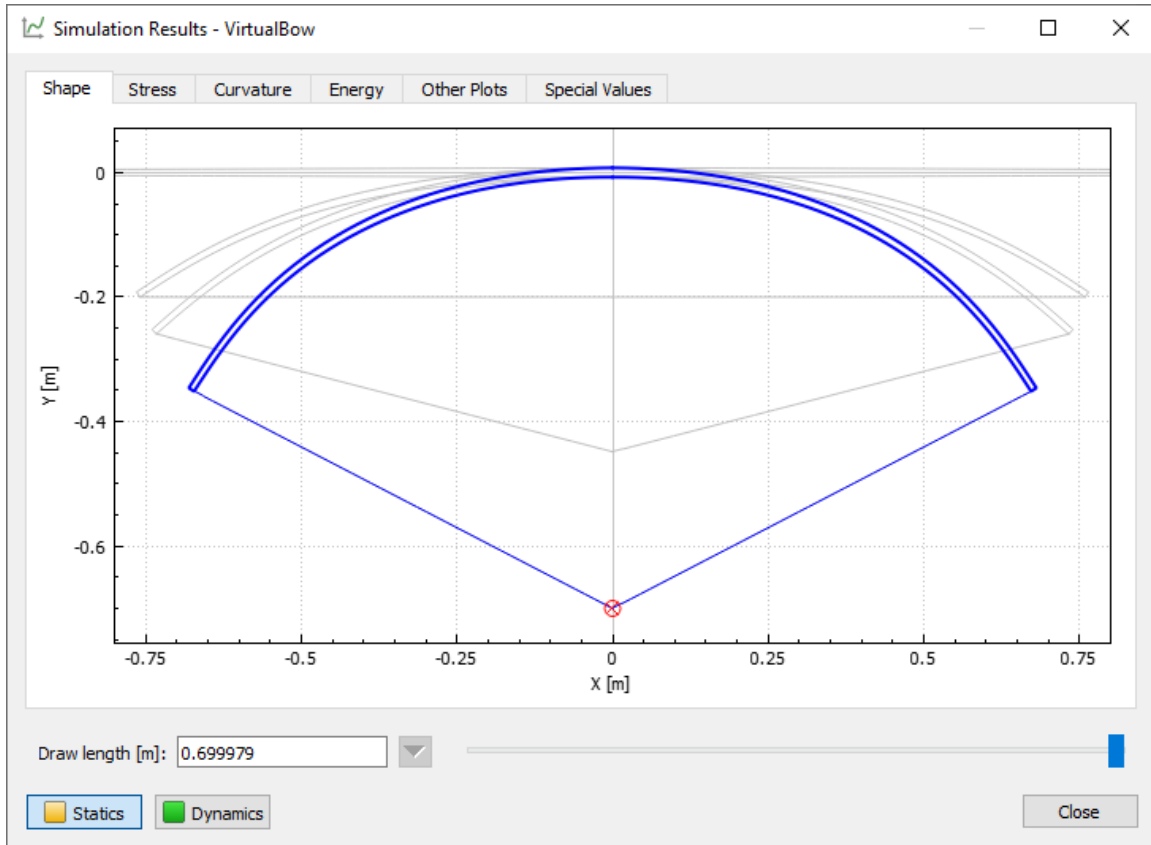
Figure 12: Simulation results

With the buttons on the bottom left you can switch between the static and dynamic (if available) results. The results themselves are organized in different tabs. At the bottom of the window there is a slider where you can change the current draw length (statics) or time (dynamics). This value applies to all of the result tabs.

These are the contents of the individual result tabs:

- **Shape:** Shows the shape of the limb and string as well as the position of the arrow at different stages of either the draw (statics) or the shot (dynamics)

- **Stress:** Shows the distribution of material stress along the length of the limb for the back and belly of each layer

- **Curvature:** Shows the curvature of the limb (difference to unbraced state)

- **Energy:** Shows how potential and kinetic energy of the different parts of the bow develop during the simulation

- **Other Plots:** Here you can combine arbitrary simulation results and plot them together, e.g. things like the draw curve of the bow or the velocity of the arrow.

- **Special Values (Statics):**

  - **String length:** Initial length of the string such that the bow meets the specified brace height

  - **Final draw force:** Draw force in fully drawn state

  - **Drawing work:** Total work done by drawing the bow.

  - **Storage ratio:** This is an indicator of the bow's capability to store energy and is defined (/made up by the author) as

    $$\text{storage\_ratio} = \frac{\text{drawing\_work}}{1/2 \cdot \text{draw\_force} \cdot (\text{draw\_length} - \text{brace\_height})}.$$

    It describes the amount of energy stored by the bow's draw curve in relation to a fictious linear draw curve with the same final draw force.

- **Special Values (Dynamics):**

  - **Arrow velocity:** Velocity of the arrow when leaving the bow

  - **Arrow energy:** Kinetic energy of the arrow when leaving the bow

  - **Efficiency:** Degree of efficiency of the bow. Useful energy output (kinetic energy of the arrow) divided by energy input (static drawing work).

3.2 Results

3.2.1 Characteristics

3.2.2 Shape

3.2.3 Stress

3.2.4 Curvature

3.2.5 Energy

3.2.6 Other Plots

4 Command Line Interface

The command line interface can be used to start simulations in batch mode, without opening the GUI. This way VirtualBow can be called from other programs for performing more advanced computations.

The command line parameters are as follows:

> **virtualbow [input] [output] [options]**

- **input:** Path to an input file (`.bow`)
- **output:** Path for the output file (`.dat`)
- **options:** Simulation options, `--static` or `--dynamic`

All of the arguments are optional. Calling VirtualBow with either no arguments or only an input file will open the GUI. If in addition to that either an output file or simulation options are provided, the simulation is carried out silently and the results are written to disk. If not specified, a default output file named after the input file is created.

Note: To use the command line interface on Windows you have to either specify the complete path to the `virtualbow.exe` executable or add the installation directory to your `PATH` environment variable.

Input and Output Files

VirtualBow's input files use the JSON[1] format, a human readable text format that stores different types of data in a hierarchical way. The output files use MessagePack[2], a more compact binary format that is otherwise very similar to JSON. Both are very common formats with implementations available in many programming languages. An example for using VirtualBow with Python can be found in Appendix C.

The exact definition and layout of the data contained in the input and output files is documented in Appendix A and B, respectively. Please note that those aren't stable yet. Future releases are very likely to introduce breaking changes.

---

[1]`http://json.org`
[2]`http://msgpack.org`

A   Input File Structure

Field	Type	Unit	Description
meta			
version	string	–	Internally used version string
comments	string	–	User comments
settings			
n_limb_elements	integer	–	Number of limb elements
n_string_elements	integer	–	Number of string elements
n_draw_steps	integer	–	Number of steps for the static simulation
time_span_factor	double	–	Factor for modifying total simulation time
time_step_factor	double	–	Factor for modifying simulation time steps
sampling_rate	double	Hz	Time resolution for the dynamic output
profile	double[][]	m, 1/m	Table with segment lengths and curvatures
width	double[][]	–, m	Table with positions and widths
layers			
{			
name	string	–	Name of the layer
height	double[][]	–, m	Table with positions and heights
rho	double	kg/m^3	Density of the layer material
E	double	Pa	Elastic modulus of the layer material
}			
{ ... }			
string			
strand_stiffness	double	N	Stiffness of the string material
strand_density	double	kg/m	Density of the string material
n_strands	integer	–	Number of strands
masses			
arrow	double	kg	Mass of the arrow
string_center	double	kg	Additional mass at string center
string_tip	double	kg	Additional mass at string tips
limb_tip	double	kg	Additional mass at limb tips
dimensions			
brace_height	double	m	Brace height
draw_length	double	m	Draw length
handle_length	double	m	Handle length
handle_setback	double	m	Handle setback
handle_angle	double	m	Handle angle

B Output File Structure

P: Number of limb nodes
Q: Number of string nodes
R: Number of layer nodes

Field	Type	Unit	Description
limb_properties			
length	double[P]	m	Arc lengths of the limb nodes (unbraced)
angle	double[P]	rad	Orientation angles of the limb nodes (unbraced)
x_pos	double[P]	m	X coordinates of the limb nodes (unbraced)
y_pos	double[P]	m	Y coordinates of the limb nodes (unbraced)
width	double[P]	m	Cross section width
height	double[P]	m	Cross section height (total)
rhoA	double[P]	kg/m	Linear density of the cross sections
Cee	double[P]	N	Longitudinal stiffness of the cross sections
Ckk	double[P]	Nm^2	Bending stiffness of the cross sections
Cek	double[P]	Nm	Coupling between bending and elongation
layers			
{			
length	double	m	Arc lengths of the layer nodes
He_back	double[R][P]	N/m^2	Stress evaluation matrix[1] (back)
Hk_back	double[R][P]	N/m	Stress evaluation matrix[1] (back)
He_belly	double[R][P]	N/m^2	Stress evaluation matrix[1] (belly)
Hk_belly	double[R][P]	N/m	Stress evaluation matrix[1] (belly)
}			
{ ... }			
statics			
states			
{ ... }			Sequence of bow states (see table below)
string_length	double	m	Initial length of the string
final_draw_force	double	N	Final draw force
drawing_work	double	J	Drawing work
storage_ratio	double	–	Storage ratio
dynamics			
states			
{ ... }			Sequence of bow states (see table below)
final_arrow_velocity	double	m/s	Final velocity of the arrow
final_arrow_energy	double	J	Final energy of the arrow
efficiency	double	–	Degree of efficiency

N: Number of simulation steps
P: Number of limb nodes
Q: Number of string nodes

Field	Type	Unit	Description
states			
time	`double[N]`	s	Time
draw_length	`double[N]`	m	Draw length
draw_force	`double[N]`	N	Draw force
string_force	`double[N]`	N	String force (total)
strand_force	`double[N]`	N	String force (strand)
grip_force	`double[N]`	N	Grip force
pos_arrow	`double[N]`	m	Arrow position
vel_arrow	`double[N]`	m/s	Arrow velocity
acc_arrow	`double[N]`	m/s^2	Arrow acceleration
x_pos_limb	`double[N][P]`	m	X coordinates of the limb nodes
y_pos_limb	`double[N][P]`	m	Y coordinates of the limb nodes
angle_limb	`double[N][P]`	rad	Rotation angles of the limb nodes
epsilon	`double[N][P]`	–	Longitudinal strain at the limb nodes
kappa	`double[N][P]`	m	Bending curvature at the limb nodes
x_pos_string	`double[N][Q]`	m	X coordinates of the string nodes
y_pos_string	`double[N][Q]`	m	Y coordinates of the string nodes
e_pot_limbs	`double[N]`	J	Potential energy of the limbs
e_kin_limbs	`double[N]`	J	Kinetic energy of the limbs
e_pot_string	`double[N]`	J	Potential energy of the string
e_kin_string	`double[N]`	J	Kinetic energy of the string
e_kin_arrow	`double[N]`	J	Kinetic energy of the arrow

[1]**Note:** For space efficiency reasons, the stresses for each layer aren't stored directly in the output data. Instead they can be calculated as needed by multiplying the layer's stress evaluation matrices with the strain and curvature of the limb at the given bow state,

$$\texttt{sigma\_back} = \texttt{He\_back} \cdot \texttt{epsilon} + \texttt{Hk\_back} \cdot \texttt{kappa}$$
$$\texttt{sigma\_belly} = \texttt{He\_belly} \cdot \texttt{epsilon} + \texttt{Hk\_belly} \cdot \texttt{kappa}$$

The result is a vector of stresses corresponding to the nodes of the layer.

C Python Scripting Example

The code example below shows how simulations can be automated with Python. It loads, modifies and saves an input file, runs a static simulation with it, loads the output results and calculates the maximum stress of the first layer at full draw.

Two external packages are needed: `msgpack` for reading the output files and `numpy` for evaluating stresses (matrix multiplication). They can be installed via

```
pip install msgpack
pip install numpy
```

Reading and writing the input file is possible out of the box by using Python's `json` standard library module. VirtualBow is called as an external process via `subprocess.call`.

```python
import json        # Loading and saving input files
import msgpack     # Loading output files
import numpy       # Evaluating stresses
import subprocess  # Runnig the simulation

# Load input file
with open("input.bow", "r") as file:
    input = json.load(file)

# Modify input
input["string"]["n_strands"] += 1

# Save modified input
with open("input.bow", "w") as file:
    json.dump(input, file)

# Run a static simulation
subprocess.call(["virtualbow", "input.bow", "output.dat", "--static"])

# Load the output file
with open("output.dat", "rb") as file:
    output = msgpack.unpack(file, raw=False)

# Calculate maximum stress for layer 0 at full draw
He_back = numpy.array(output["limb_properties"]["layers"][0]["He_back"])
Hk_back = numpy.array(output["limb_properties"]["layers"][0]["Hk_back"])
epsilon = numpy.array(output["statics"]["states"]["epsilon"][-1])
kappa   = numpy.array(output["statics"]["states"]["kappa"][-1])

sigma_back = He_back.dot(epsilon) + Hk_back.dot(kappa)
print(sigma_back.max())
```

D A Simple Bending Test

A bending test is an easy way to determine the elastic modulus of a material. It can be done without any special equipment. Figure 13 shows the setup. A test piece with length l is clamped on one side and subjected to a vertical force F at its free end. The deflection s due to this load is measured.
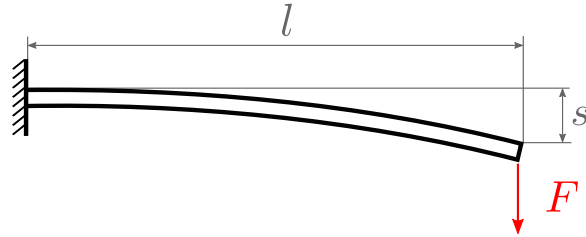

Figure 13: Experimental setup

The elastic modulus can then be calculated depending on the cross section geometry using the equations in Table 3.

Test geometry	Elastic Modulus
	$E = \dfrac{4}{wh^3} \dfrac{Fl^3}{s}$
	$E = \dfrac{12 \ln(h_l\, l) + 6}{w\,(h_l - h_0)^3} \dfrac{Fl^3}{s}$

Table 3: Elastic modulus for different test geometries

Here are a few practical considerations:

- The precision of the cross sections is very important, especially the height.

- The equations above hold for slender beams and small deflections. The test setup should be chosen accordingly. As a rule of thumb: $h, s < l/15$.

- A simple way to apply a defined force is to hang a mass m onto the beam tip and use $F = m \cdot g$, with $g = 9.81 \, \mathrm{m/s^2}$.

- If there is some small initial deflection due to gravity, then s is simply the difference in deflection after application of the force.