Real-time Implementation of an Elasto-Plastic Friction Model applied to Stiff Strings using Finite-Difference Schemes

Silvin Willemsen[1], Stefan Bilbao[2], Stefania Serafin[1]

[1] Multisensory Experience Lab, CREATE, Aalborg University Copenhagen, Denmark
[2] Acoustics and Audio Group, University of Edinburgh, UK
sil@create.aau.dk

Introduction

- The simulation of a **bowed string** is challenging due to the strongly **non-linear relationship** between the bow and the string.
- This relationship can be described by a **model of friction**.
- A recently popular and accurate friction model is the **elasto-plastic** model.
- This can be applied to a string implemented with **FDTD methods**, which are also focused on accuracy.
- We are interested in **bridging the gap** between highly **accurate** physical models and **efficient** implementations.
- In this work, we present an implementation of the elasto-plastic friction model in conjunction with a finite-difference implementation of the damped stiff string.
- Furthermore, we show that it is possible to play the string in **real-time** using the Sensel Morph controller [1].

Elasto-Plastic Bow Model

- The elasto-plastic friction model assumes that the friction between objects in contact is caused by a large **ensemble of bristles** (see Fig. 1).
- Next to the relative velocity v between the bow and the string, the **average bristle displacement** z is introduced as a second independent variable.

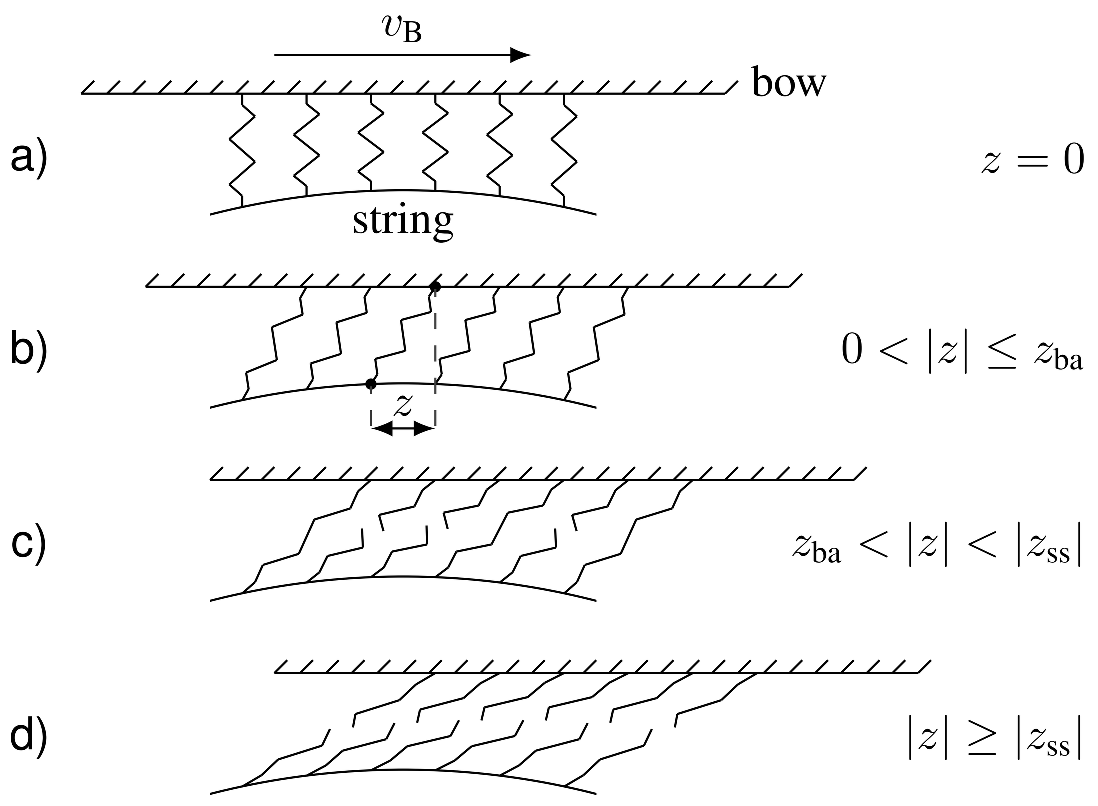

Fig. 1: Microscopic displacements of the bristles between the bow and the string. The bow moves right with a velocity of v_B.
a) **Initial** state. The average bristle displacement $z = 0$.
b) The purely **elastic**, or presliding regime (STICK).
c) The **elasto-plastic** regime.
d) The purely **plastic** regime (SLIP).

Applying to stiff string

Using the subscripts t and x to denote differentiation, we can write the equation for the bow-excited **linear damped stiff string**

$$u_{tt} = c^2 u_{xx} - \kappa^2 u_{xxxx} - 2\sigma_0 u_t + 2\sigma_1 u_{txx} - \delta(x - x_\mathrm{B}) f(v, z)/\rho A \tag{1}$$

with **force function**

$$f(v, z) = s_0 z + s_1 \dot{z} + s_2 v + s_3 w, \tag{2}$$

and **relative velocity** with bowing point x_B and bowing velocity v_B is defined as

$$v = u_t(x_\mathrm{B}) - v_\mathrm{B}. \tag{3}$$

Elasto-Plastic Bow Model cont.

The **time-derivative of** z is defined as $\dot{z}$ and is related to v through

$$\dot{z} = r(v, z) = v\left[1 - \alpha(v, z)\frac{z}{z_\mathrm{ss}(v)}\right], \tag{4}$$

with **steady-state function** z_ss (see Fig. 2),

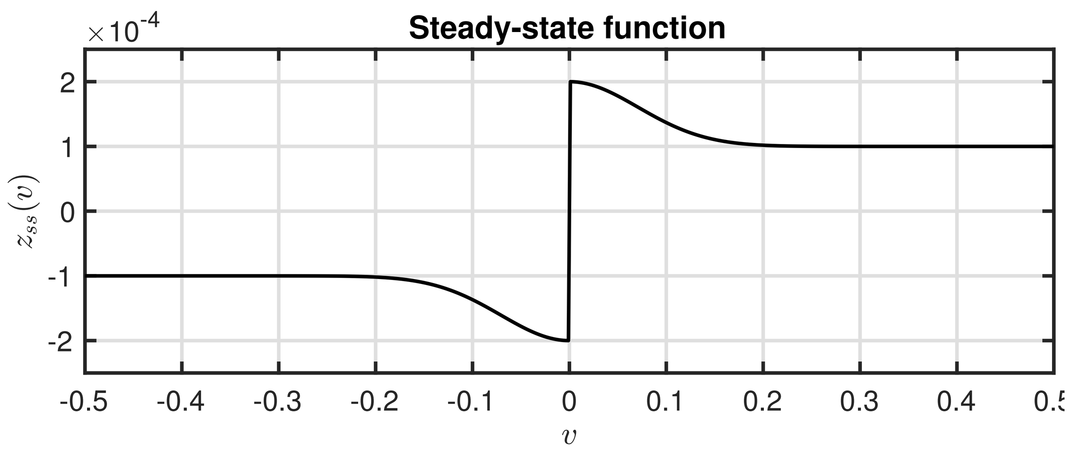

Fig. 2: A plot of the steady-state function $z_\mathrm{ss}(v)$ with a force of 5 N.

and the **adhesion map** between the bow and the string $\alpha(v, z)$, which is defined as (see Fig. 3)

$$\alpha(v, z) = \begin{cases} \left.\begin{cases} 0 & |z| \leq z_\mathrm{ba} \\ \alpha_\mathrm{m} & z_\mathrm{ba} < |z| < |z_\mathrm{ss}(v)| \\ 1 & |z| \geq |z_\mathrm{ss}(v)| \end{cases}\right\} & \text{if } \operatorname{sgn}(v) = \operatorname{sgn}(z) \\ 0 & \text{if } \operatorname{sgn}(v) \neq \operatorname{sgn}(z) \end{cases}$$

where α_m is the transition between the elastic and plastic behaviour.

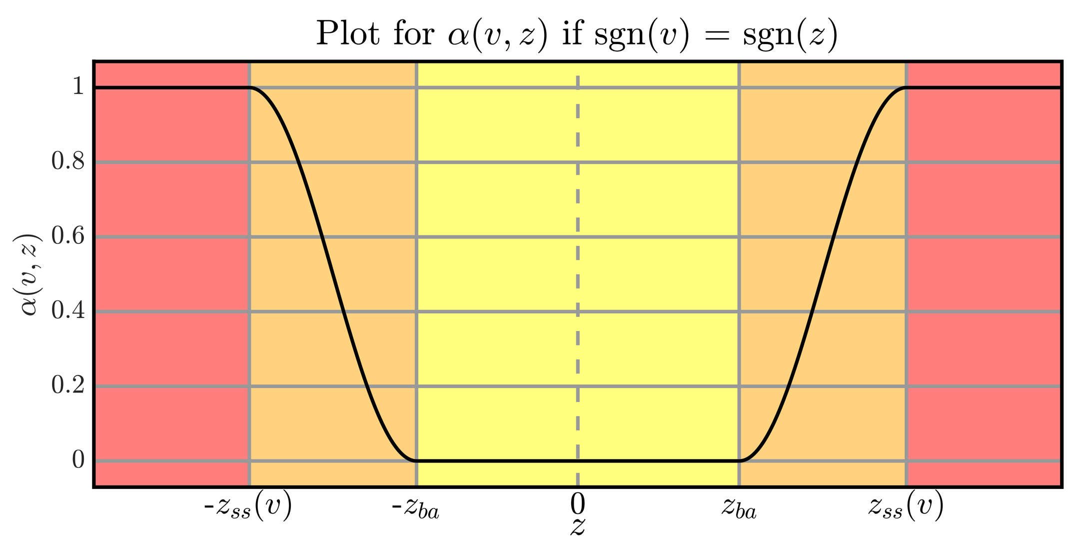

Fig. 3: A plot of the adhesion map $\alpha(v, z)$ plotted against z when the signs of v and z are the same. The different regions of the map are shown with the coloured areas and correspond to Fig. 1 according to: yellow - a) & b), orange - c) and red - d).

Discretisation

At the bowing point we need to iteratively solve for relative velocity v^n and average bristle displacement z^n at sample n using **multivariate Newton-Raphson**. We can rewrite (1) (in discrete-time) to

$$g_1(v^n, z^n) = \frac{s_0 z^n + s_1 r^n + s_2 v^n + s_3 w^n}{\rho A h} + \left(\frac{2}{k} + 2\sigma_0\right) v^n + b^n = 0, \tag{5}$$

where b^n is not dependent on v^n and z^n and can be pre-computed. Furthermore, using the **discrete counterpart of** (4) and defining a^n as the **trapezoid rule applied to** z, we define

$$g_2(v^n, z^n) = r^n - a^n, \tag{6}$$

and obtain the following iteration

$$\begin{bmatrix} v^n_{(i+1)} \\ z^n_{(i+1)} \end{bmatrix} = \begin{bmatrix} v^n_{(i)} \\ z^n_{(i)} \end{bmatrix} - \begin{bmatrix} \frac{\partial g_1}{\partial v} & \frac{\partial g_1}{\partial z} \\ \frac{\partial g_2}{\partial v} & \frac{\partial g_2}{\partial z} \end{bmatrix}^{-1} \begin{bmatrix} g_1 \\ g_2 \end{bmatrix}, \tag{7}$$

where i is the iteration number capped by 50 iterations, and the convergence threshold is set to 10^{-7}.

Implementation

- The real-time implementation has been done using **C++** and the **JUCE** framework [2].

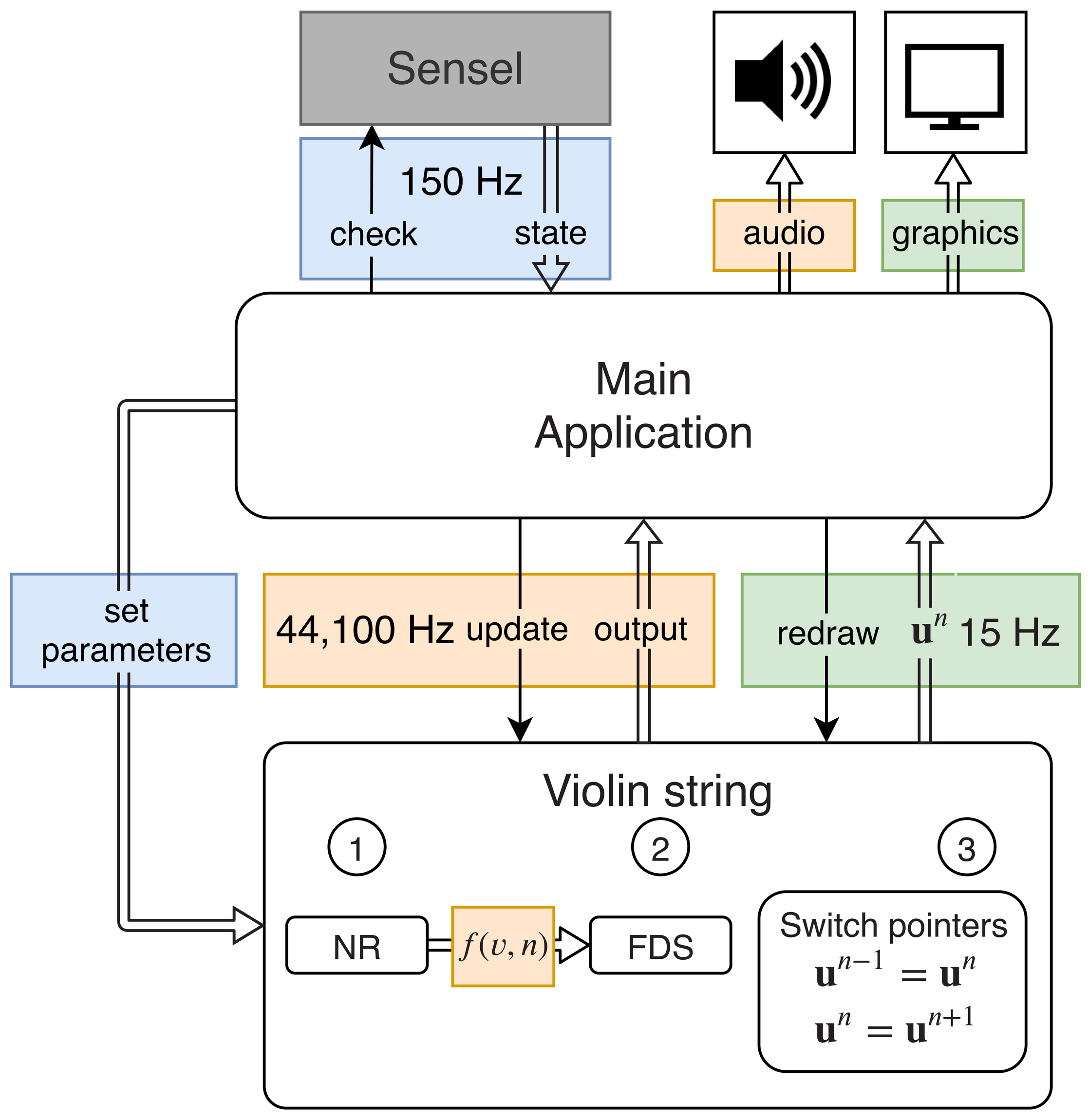

Fig. 4: The system architecture. Black arrows indicate instructions and hollow arrows indicate data flows.

- The three main **components** of the application are:
  - the **Sensel** for controlling the application,
  - the **violin string class** that performs the simulation, and
  - the **main application class** that moderates between these and the auditory and visual outputs.
- The three main threads running are (color in Fig. 4, frequency):
  - the **Graphics** thread (green, 15 Hz)
  - the **Sensel** thread (blue, 150 Hz)
  - the **Audio** thread (orange, 44,100 Hz)

Results and Discussion

- The algorithm was tested with different numbers of strings according to the violin tuning of empty strings.

Table 1: Average CPU usage for different amounts of strings. All strings are bowed simultaneously (polyphonically).

# strings	Graphics (%)	No graphics (%)
1	44.8	5.95
2	47.7	9.54
3	52.8	12.1
4	60.9	17.9

Conclusion

- With a single string we are able to keep the **CPU usage under 6%**.
- Future work includes parameter design and including an instrument body for a more realistic sound.

References

[1] Sensel Inc., "Sensel Morph," Available at https://sensel.com/, accessed April 01, 2019.

[2] JUCE ROLI, "JUCE," Available at https://juce.com/.