Abstract

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 through a model of friction. Several friction models in the literature have been proposed, from simple velocity dependent to more accurate ones. Similarly, a highly accurate technique to simulate a stiff string is the use of finite-difference time-domain (FDTD) methods. As these models are generally computationally heavy, implementation in real-time is challenging. This paper presents a real-time implementation of the combination of a complex friction model, namely the elasto-plastic friction model, and a stiff string simulated using FDTD methods. We show that it is possible to keep the CPU usage of a single bowed string below 6 percent. For real-time control of the bowed string, the Sensel Morph is used.

1 Introduction

In physical modelling sound synthesis applications, the simulation of a bowed string is a challenging endeavour. This is mainly due to the strongly non-linear relationship between the bow and the string, through a model of friction. Such friction models can be categorised as static or dynamic; models of the latter type have only recently seen a major effort. As opposed to static friction models, where friction depends only on the relative velocity of the two bodies in contact, dynamic models describe the friction force through a differential equation.

A recently popular dynamic model is the elasto-plastic model, first proposed in [1]. The model assumes that the friction between the two objects in contact is caused by a large ensemble of bristles, each of which contributes to the total friction force. The average bristle deflection is used as an extra independent variable for calculating the friction force. As shown in [2], the elasto-plastic model can be applied to a bowed string simulation and it exhibits a hysteresis loop in the force versus velocity plane due to this multivariable dependency. This is consistent with measurements performed using a bowing machine in [3]. The elasto-plastic model has been thoroughly investigated in a musical context by Serafin et al. in [2, 4, 5].

Regarding bowed string simulations, the first musical non-linear systems, including bowed strings, were presented by McIntyre, et al. in [6]. Smith published the first real-time implementation of the bowed string using a digital waveguide (DW) for the string and a look-up table for the friction model in [7]. Simultaneously, Florens, et al. published a real-time implementation using mass-spring systems for the string and a static friction model for the bow in [8].

The dynamics of musical instruments are generally described by systems

of partial differential equations (PDEs). Specialised synthesis methods such as DWs [9] and modal synthesis [10] are derived from particular solutions. Mainstream time-stepping methods such as finite-difference time-domain (FDTD) methods were first proposed in [11, 12, 13], and developed subsequently [14, 15]. In [16] the authors adapted the thermal model proposed by Woodhouse in [3] for real-time applications using a DW for the string implementation and a combination of the DW and FDTD methods for the bowing interaction. In [17, 18], Desvages used FDTD methods for the implementation of the string in two polarizations and a static double exponential friction model introduced in [19]. This was, however, not implemented in real-time. To the best of the authors' knowledge, the only known real-time implementation of any bow model applied to complete FDTD strings was presented in [20] where the soft exponential friction function presented in [14] was used. The current work can be considered an extension of this work.

We are interested in bridging the gap between highly accurate physical models and efficient implementations so that these models can be played in real-time. 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 [21].

This paper is structured as follows. In Section 2, the elasto-plastic bow model in conjunction with a PDE model for a stiff string is described. Discretisation is covered in Section 3, and implementation details appear in Section 4. In Section 5, simulated results are presented and discussed. Some concluding remarks appear in Section 6.

2 Elasto-Plastic Bow Model

Consider a linear model of transverse string vibration in a single polarization, where $u(x,t)$ represents string displacement as a function of time $t \geq 0$, in s, and coordinate $x \in [0,\, L]$ (in m) for some string length L (in m). Using the subscripts t and x to denote differentiation with respect to time and space respectively, a partial differential equation describing the dynamics of the damped stiff string is [14]

$$u_{tt} = c^2 u_{xx} - \kappa^2 u_{xxxx} - 2\sigma_0 u_t + 2\sigma_1 u_{txx}. \tag{1}$$

Here, $c = \sqrt{T/\rho A}$ is the wave speed (in m/s) with tension T (in N), material density ρ (in kg·m^{-3}) and cross-sectional area A (in m^2). Furthermore, $\kappa = \sqrt{EI/\rho A}$ is the stiffness coefficient (in m^2/s) with Young's Modulus E (in Pa) and area moment of inertia I (in m^4). For a string of circular cross section we have radius r (in m), cross-sectional area $A = \pi r^2$ and area moment of inertia

$I = \pi r^4/4$. Lastly, $\sigma_0 \geq 0$ (in s^{-1}) and $\sigma_1 \geq 0$ (in m^2/s) are coefficients allowing for frequency-independent and frequency-dependent damping respectively.

In our implementation we assume simply supported boundary conditions, which are defined as

$$u = u_{xx} = 0 \quad \text{where} \quad x = 0, L\,. \tag{2}$$

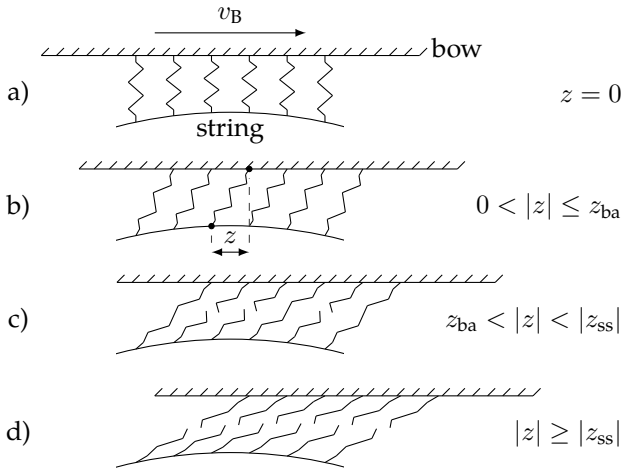

Fig. 1: *Microscopic displacements of the bristles between the bow and the string. The bow moves right with a velocity of v_B. a) The initial state is where the average bristle displacement $z = 0$. b) The bow has moved right relative to the string. The purely elastic, or presliding regime is entered (stick). c) After the break-away displacement z_{ba}, more and more bristles start to 'break'. This is defined as the elasto-plastic regime. d) After all bristles have 'broken', the steady state (slip) is reached and the purely plastic regime is entered.*

As mentioned in the introduction, the elasto-plastic bow model assumes that the friction between the bow and the string is due to a large ensemble of bristles, each of which contributes to the total friction force. See Figure 1 for a graphical representation of this. The bristles are assumed to be damped stiff springs and can 'break' after a given break-away displacement threshold. An extra term can be added to (1) to include the bowing interaction

$$u_{tt} = \ldots - \delta(x - x_\mathrm{B}) f(v, z)/\rho A. \tag{3}$$

Here, the spatial Dirac delta function $\delta(x - x_\mathrm{B})$ (in m^{-1}) allows for the pointwise application of the force f (in N) at externally supplied bowing position $x_\mathrm{B}(t)$ (in m).

In the following we will use the definitions found in [1]. The force f is defined in terms of the relative velocity v (in m/s) and average bristle displace-

ment z (in m) (see Figure 1) as

$$f(v, z) = s_0 z + s_1 \dot{z} + s_2 v + s_3 w, \tag{4}$$

where

$$v = u_t(x_{\mathrm{B}}) - v_{\mathrm{B}}, \tag{5}$$

where $v_{\mathrm{B}}(t)$ is an externally supplied bow velocity, s_0 is the bristle stiffness (in N/m), s_1 is the damping coefficient (in kg/s), s_2 is the viscous friction (in kg/s) and s_3 is a dimensionless noise coefficient multiplied onto pseudorandom function $w(t)$ (in N) as done in [4] and adds noise to the friction force. Here, $\dot{z}$ indicates a time derivative of z, and is related to v through

$$\dot{z} = r(v, z) = v\left[1 - \alpha(v, z)\frac{z}{z_{\mathrm{ss}}(v)}\right], \tag{6}$$

where z_{ss} is the steady-state function

$$z_{\mathrm{ss}}(v) = \frac{\mathrm{sgn}(v)}{s_0}\left[f_{\mathrm{C}} + (f_{\mathrm{S}} - f_{\mathrm{C}})e^{-(v/v_{\mathrm{S}})^2}\right], \tag{7}$$

with Stribeck velocity v_{S} (in m/s), Coulomb force $f_{\mathrm{C}} = f_{\mathrm{N}}\mu_{\mathrm{C}}$ and stiction force $f_{\mathrm{S}} = f_{\mathrm{N}}\mu_{\mathrm{S}}$ (both in N). Here μ_{C} and μ_{S} are the dynamic and static friction coefficient respectively and $f_{\mathrm{N}}(t)$ is the normal force (in N) which is, like $v_{\mathrm{B}}(t)$, externally supplied. See Figure 2 for a plot of (7).

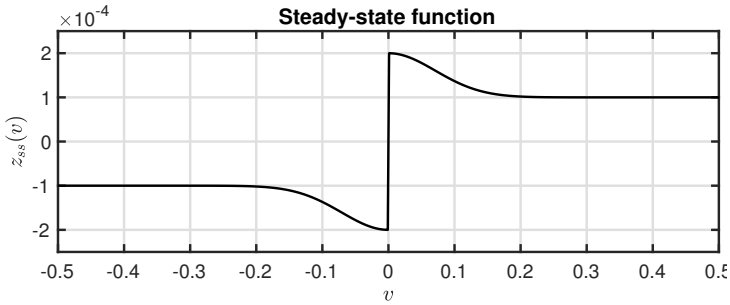

Fig. 2: *A plot of the steady-state function $z_{ss}(v)$ with a force of 5 N.*

Furthermore, the adhesion map between the bow and the string is defined as

$$\alpha(v, z) = \begin{cases} \left.\begin{array}{ll} 0 & |z| \le z_{\mathrm{ba}} \\ \alpha_{\mathrm{m}}(v, z) & z_{\mathrm{ba}} < |z| < |z_{\mathrm{ss}}(v)| \\ 1 & |z| \ge |z_{\mathrm{ss}}(v)| \end{array}\right\} & \text{if } \mathrm{sgn}(v) = \mathrm{sgn}(z) \\ 0 & \text{if } \mathrm{sgn}(v) \neq \mathrm{sgn}(z), \end{cases} \tag{8}$$

where the transition between the elastic and plastic behaviour is defined as

$$\alpha_{\mathrm{m}} = \frac{1}{2}\left[1 + \mathrm{sgn}(z)\sin\left(\pi\frac{z - \mathrm{sgn}(z)\frac{1}{2}(|z_{\mathrm{ss}}(v)| + z_{\mathrm{ba}})}{|z_{\mathrm{ss}}(v)| - z_{\mathrm{ba}}}\right)\right], \tag{9}$$

with break-away displacement z_{ba}, i.e., where the bristles start to break (see Figure 1 c)). A plot of the adhesion map can be found in Figure 3.[1]

One of the difficulties in working with this model is that, due to the many approximations, the notion of an energy balance, relating the rate of stored energy in the system to power input and loss is not readily available. Such energy methods are used frequently in the context of physical modeling synthesis and virtual analog modeling as a means of arriving at numerical stability conditions for strongly nonlinear systems, as is the present case. See, e.g., [14]. This means that we do not have a means of ensuring numerical stability in the algorithm development that follows. This does not mean, however, that an energy balance is not available.

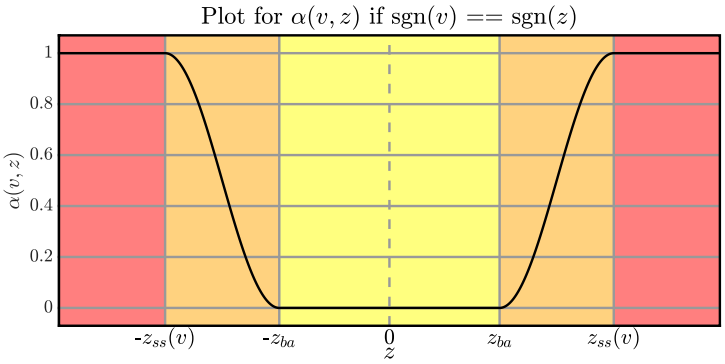

Fig. 3: *A plot of the adhesion map $\alpha(v,z)$ plotted against z when the signs of v and z are the same. The different regions of the map are shown with the coloured areas and correspond to Figure 1 according to: yellow - a) & b), orange - c) and red - d).*

3 Discretisation

Finite-difference schemes for the stiff string in isolation are covered by various authors [13, 14].

Equation (1) can be discretised at times $t = nk$, with sample $n \in \mathbb{N}$ and time-step $k = 1/f_{\mathrm{s}}$ (in s) with sample-rate f_{s} (in Hz) and locations $x = lh$,

[1] It is interesting to note is that in the literature on this topic such as [1, 2, 4, 5], a few inaccuracies can be found in the definition of $\alpha(v,z)$: 1) all uses of z_{ss} in (8) and (9) lack the absolute value operator, 2) the multiplications with $\mathrm{sgn}(z)$ in (9) are excluded, 3) $\alpha(v,z)$ is undefined for $|z| = z_{\mathrm{ba}}$ and $|z| = |z_{\mathrm{ss}}(v)|$ (correct in the original paper by Dupont et al. [1]). It can be shown that only with the definitions presented here, is it possible to obtain the curve shown in Figure 3.

where grid spacing h (in m) needs to abide the following condition [14]

$$h \geq h_{\text{min}} = \sqrt{\frac{c^2k^2 + 4\sigma_1 k + \sqrt{(c^2k^2 + 4\sigma_1 k)^2 + 16\kappa^2 k^2}}{2}} \tag{10}$$

and grid points $l \in [0, ..., N]$, where $N = \text{floor}(L/h)$ and $N + 1$ is the total number of grid points. It is important to note that the closer h is to h_{min}, the more accurate the scheme will be. Approximations for the derivatives found in (1) are described in the following way [14]:

$$u_t \approx \delta_{t\cdot} u_l^n = \frac{1}{2k}\big(u_l^{n+1} - u_l^{n-1}\big), \tag{11a}$$

$$u_{tt} \approx \delta_{tt} u_l^n = \frac{1}{k^2}\big(u_l^{n+1} - 2u_l^n + u_l^{n-1}\big), \tag{11b}$$

$$u_{xx} \approx \delta_{xx} u_l^n = \frac{1}{h^2}\big(u_{l+1}^n - 2u_l^n + u_{l-1}^n\big), \tag{11c}$$

$$u_{txx} \approx \delta_{t-}\delta_{xx} u_l^n = \frac{1}{hk^2}\big(u_{l+1}^n - 2u_l^n + u_{l-1}^n - u_{l+1}^{n-1} + 2u_l^{n-1} - u_{l-1}^{n-1}\big), \tag{11d}$$

$$u_{xxxx} \approx \delta_{xxxx} u_l^n = \frac{1}{h^4}\big(u_{l+2}^n - 4u_{l+1}^n + 6u_l^n - 4u_{l-1}^n + u_{l-2}^n\big), \tag{11e}$$

with grid function u_l^n denoting a discretised version of $u(x,t)$ at the nth time step and the lth point on the string. Note that in (11d), the backwards time difference operator is used to keep (12) explicit and thus computationally cheaper to update. Using the approximations shown in (11), (3) can be discretised to

$$\begin{aligned} \delta_{tt} u_l^n = {} & c^2 \delta_{xx} u_l^n - \kappa^2 \delta_{xxxx} u_l^n - 2\sigma_0 \delta_{t\cdot} u_l^n \\ & + 2\sigma_1 \delta_{t-}\delta_{xx} u_l^n - J(x_{\text{B}}^n) f(v^n, z^n)/\rho A, \end{aligned} \tag{12}$$

where the relative velocity described in (5) can be discretised as

$$v^n = I(x_{\text{B}}^n)\delta_{t\cdot} u_l^n - v_{\text{B}}^n. \tag{13}$$

Here, $I(x_{\text{B}}^n)$ and $J(x_{\text{B}}^n)$ are weighting functions where the former interpolates the string displacement and velocity and the latter distributes the bowing term around time-varying bowing position x_{B}^n (see Figure 4 and [14] for more details on this). Furthermore,

$$f(v^n, z^n) = s_0 z^n + s_1 r^n + s_2 v^n + s_3 w^n \tag{14}$$

is the discrete counterpart of (4) where

$$r^n = r(v^n, z^n) = v^n \left[1 - \alpha(v^n, z^n)\frac{z^n}{z_{\text{ss}}(v^n)}\right] \tag{15}$$

is the discrete counterpart of (6).

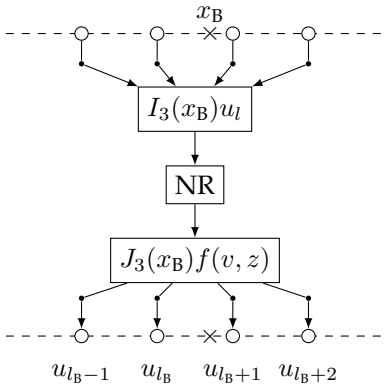

Fig. 4: *Cubic interpolation at bowing point x_B. The interpolator I retrieves the values of four grid points which are then used in the Newton-Raphson (NR) solver. This outputs the force function $f(v,z)$ that the spreading function J in turn distributes over the same four grid points. This process happens every single sample.*

At the bowing point we need to iteratively solve for two unknown variables: the relative velocity between the bow and the string v^n and the mean bristle displacement z^n of the bow at sample n. We can solve (12) at x_B^n using (13) and identity [14]

$$\delta_{tt} u_l^n = \frac{2}{k}\big(\delta_{t\cdot} u_l^n - \delta_{t-} u_l^n\big) \tag{16}$$

resulting in

$$I(x_\mathrm{B}^n) J(x_\mathrm{B}^n) f(v^n, z^n)/\rho A + \Big(\frac{2}{k} + 2\sigma_0\Big) v^n + b^n = 0, \tag{17}$$

where

$$\begin{aligned} b^n = {} & \frac{2}{k} v_\mathrm{B}^n - \frac{2}{k} I(x_\mathrm{B}^n)\delta_{t-} u_l^n - c^2 I(x_\mathrm{B}^n)\delta_{xx} u_l^n + \kappa^2 I(x_\mathrm{B}^n)\delta_{xxxx} u_l^n \\ & + 2\sigma_0 v_\mathrm{B}^n - 2\sigma_1 I(x_\mathrm{B}^n)\delta_{t-}\delta_{xx} u_l^n \end{aligned} \tag{18}$$

and can be pre-computed as its terms are not dependent on v^n or z^n. Recalling (4), this can be rewritten to

$$I(x_\mathrm{B}^n) J(x_\mathrm{B}^n) \left(\frac{s_0 z^n + s_1 r^n + s_2 v^n + s_3 w^n}{\rho A} \right) + \Big(\frac{2}{k} + 2\sigma_0\Big) v^n + b^n = 0. \tag{19}$$

To obtain the values of v^n and z^n, multivariate Newton-Raphson (NR) is

used. If (19) is defined to be $g_1 = g_1(v^n, z^n)$ and

$$g_2(v^n, z^n) = r^n - a^n = 0, \tag{20}$$

with

$$a^n = (\mu_{t-})^{-1} \delta_{t-} z^n \tag{21}$$

(where the operators applied to z^n denote the trapezoid rule [14]) we obtain the following iteration

$$\begin{bmatrix} v^n_{(i+1)} \\ z^n_{(i+1)} \end{bmatrix} = \begin{bmatrix} v^n_{(i)} \\ z^n_{(i)} \end{bmatrix} - \begin{bmatrix} \frac{\partial g_1}{\partial v} & \frac{\partial g_1}{\partial z} \\ \frac{\partial g_2}{\partial v} & \frac{\partial g_2}{\partial z} \end{bmatrix}^{-1} \begin{bmatrix} g_1 \\ g_2 \end{bmatrix}, \tag{22}$$

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

4 Implementation

In this section, we will elaborate on the implementation; the parameters used and the system architecture. The real-time implementation of the discrete-time model shown in the previous section has been done using C++ together with the JUCE framework [22]. The application is shown in Figure 5. The parameters we used can be found in Table 1, most of which are based on implementations by Serafin in [4]. These parameters will be static, i.e., are not user-controlled (except for z_{ba} and s_3 which rely on f_{N}). A demonstrative video can be found in [23]. We use the passivity condition proposed by

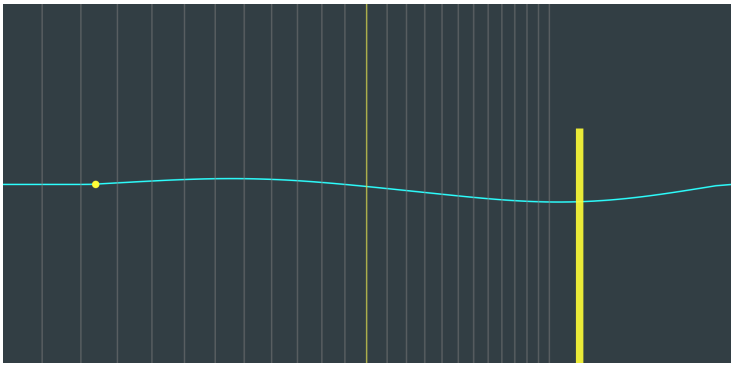

Fig. 5: *The elasto-plastic bowed string application. The bow is shown as a yellow rectangle, moves on interaction and its opacity depends on the finger force. The state* $\mathbf{u}^n$ *is visualised using the cyan curve and stopping-finger position is shown as a yellow circle. The grey lines show the 'frets' corresponding to semi-tones as a visual reference for the stopping position and do not influence the model.*

[24] for our choices of different parameter-values. As this condition applies to

Table 1: *Parameter values. Values for the fundamental frequency f_0 can be found in Section 5.*

Parameter	Symb. (unit)	Value (notes)
Material Density	ρ (kg·m^{-3})	7850
Radius	r (m)	$5 \cdot 10^{-4}$
String length	L (m)	1
Wave speed	c (m/s)	$2f_0/L$
Young's modulus	E (Pa)	$2 \cdot 10^{11}$
Freq. indep. damping	σ_0 (s^{-1})	1
Freq. dep. damping	σ_1 (m^2/s)	$5 \cdot 10^{-3}$
Coulomb friction	μ_C (-)	0.3 ($< \mu_S$)
Static friction	μ_S (-)	0.8 ($> \mu_C$)
Normal force	f_N (N)	10
Bow velocity	v_B (m/s)	0.1
Bow position	x_B (m)	0.25
Stribeck velocity	v_S (m/s)	0.1
Bristle stiffness	s_0 (N/m)	10^4
Bristle damping	s_1 (kg/s)	$0.001\sqrt{s_0}$
Viscous friction	s_2 (kg/s)	0.4
Noise coefficient	s_3 (-)	$0.02f_N$
Pseudorandom func.	w (N)	$-1 < w < 1$
Break-away disp.	z_{ba} (m)	$0.7f_C/s_0$ ($< f_C/s_0$)
Sample rate	f_s (Hz)	44,100
Time step	k (s)	$1/f_s$

the LuGre model first proposed in [25, 26] from which the elasto-plastic model evolved, further investigation is required to conclude whether these conditions are identical for the elasto-plastic model.

4.1 Sensel Morph

As mentioned in Section 1, the Sensel Morph (or Sensel for short) is used as an interface to control the bowed string (see Figure 6). The Sensel is a highly sensitive touch controller containing ca. 20,000 pressure sensitive sensors that allow for expressive control of the implementation [21].

4.2 Interaction

The first finger the Sensel registers is linked to the following parameters: the normal force of the bow f_N (finger pressure), the bowing velocity v_B (vertical finger velocity) and bowing position x_B (horizontal finger position). The parameters are limited by the following conditions: $0 \leq f_N \leq 10$, $-0.3 \leq v_B \leq 0.3$ and $0 < x_B < L$. The second finger acts as a stopping finger on the string. As done in [20], for a string stopped at location $x_f \in [0, L]$ and $l_f = \text{floor}(x_f/h)$ we

Fig. 6: *The Sensel Morph: an expressive touch sensitive controller used for controlling the real-time elasto-plastic bowed string implementation.*

use

$$u_l^n = \begin{cases} 0, & l = l_\mathrm{f} - 1 \vee l = l_\mathrm{f} \\ (1 - \alpha_\mathrm{f}^\epsilon) u_l^n, & l = l_\mathrm{f} + 1 \\ u_l^n, & \text{otherwise} \end{cases} \tag{23}$$

where $\alpha_\mathrm{f} = x_\mathrm{f}/h - l_\mathrm{f}$ and $\epsilon = 7$ is a heuristic value that has been found to most linearly alter pitch between grid points.

4.3 System Architecture

Implementation of the scheme shown in (12) starts by expanding the operators shown in (11) and solving for the state at the next sample $\mathbf{u}^{n+1}$ where $\mathbf{u}$ is a vector containing the values for all grid points $l \in [0, ..., N]$.

An overview of the system architecture can be found in Figure 7. The three main components of the application are the Sensel 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 black arrows indicate instructions that one of these components can give to another and the hollow arrows indicate data flows. Moreover, the arrows are accompanied by coloured boxes, depicting what thread the instruction or data flow is associated with and at what rate this runs.

The graphics thread has the lowest priority, is denoted by the green boxes and runs at 15 Hz. The redraw instruction merely retrieves the current string state $\mathbf{u}^n$ and bow and finger position and visualises this as shown in Figure 5.

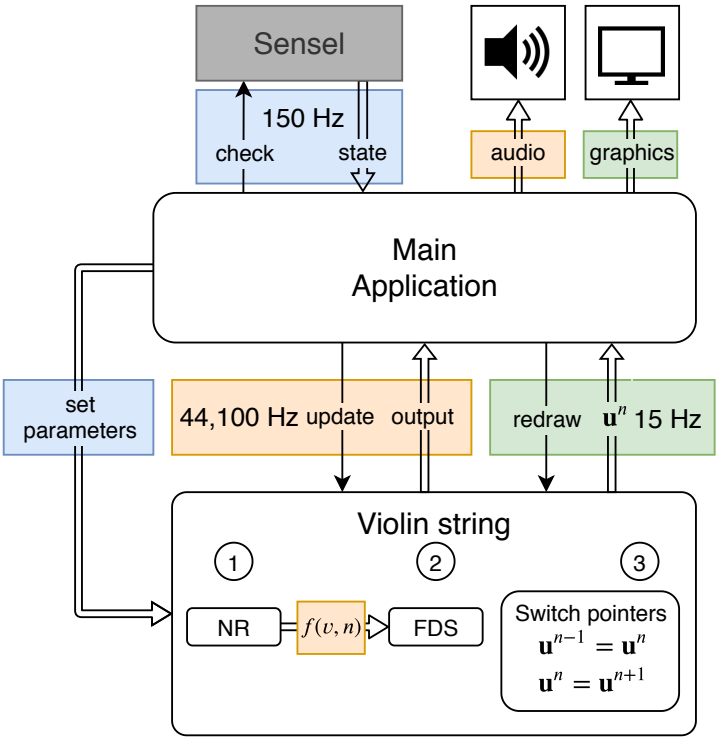

Fig. 7: *The system architecture. See Section 4.3 for a thorough explanation.*

The thread checking and receiving data from the Sensel runs at 150 Hz and is denoted by the blue boxes. The parameters that the user interacts with (bowing force, velocity and position) are also updated at this rate.

The highest priority thread is the audio thread denoted by the orange boxes and runs at 44,100 Hz. The violin string class gets updated at this rate and performs operations in the order shown in Algorithm 1.

5 Results and Discussion

Figure 8 shows the output waveforms for a string with $f_0 = 440$ Hz at different points along the string. The bowing parameters are $f_\mathrm{N} = 5$ N and $v_\mathrm{B} = 0.1$ m/s. The figure shows the traditional Helmholtz motion, which is the characteristic motion of a bowed string.

To test whether the implementation exhibits a hysteresis loop, the force vs. relative velocity plane was visualised. In Figure 9, this plot can be found for which the same parameters have been used. The figure shows values for 500 samples around $t = 0.5 f_\mathrm{s}$. As can be seen from the figure, the hysteresis loop

for *t = 1:lengthSound* **do**
  calculate computable part b^n (Eq. (18))
  $\epsilon = 1$
  $i = 0$
  while $\epsilon < tol \wedge i < 50 \wedge f_C > 0$ **do**
    calculate..
    1. $z_{\text{ss}}(v^n_{(i)})$ (Eq. (7) in discrete-time)
    2. $\alpha(v^n_{(i)}, z^n_{(i)})$ (Eq. (8) in discrete-time)
    3. $r(v^n_{(i)}, z^n_{(i)})$ (Eq. (15))
    4. g_1, g_2 (Eqs. (19) and (20))
    5.–9. Compute derivatives of 1.–4. in the same order.
    10. Perform vector NR to obtain $v^n_{(i+1)}$ and $z^n_{(i+1)}$
    11. Calculate ϵ: $\epsilon = \left\| \begin{bmatrix} v^n_{(i+1)} \\ z^n_{(i+1)} \end{bmatrix} - \begin{bmatrix} v^n_{(i)} \\ z^n_{(i)} \end{bmatrix} \right\|$
    12. Increment i: $i = i + 1$
  end
  Repeat 1.–3. using the values for v^n and z^n from the NR iteration.
  Calculate $f(v^n, z^n)$ (Eq. (14))
  Calculate $\mathbf{u}^{n+1}$ (Eq. (12) expanded)
  $\mathbf{u}^{n-1} = \mathbf{u}^n$
  $\mathbf{u}^n = \mathbf{u}^{n+1}$
end

Algorithm 1: Pseudocode showing the order of calculations.

is achieved and is similar to the one observed in [19]. The group of values around $v = 0$ are due to the sticking behaviour, and the others (the loop on the left) to the slipping behaviour.

For testing the speed of the algorithm, a MacBook Pro with a 2.2 GHz Intel Core i7 processor was used. The algorithm was tested using different frequencies according to the violin tuning of empty strings: $f_0 = 196.0$ (G3), 293.66 (D4), 440.0 (A4) and 659.26 (E5) Hz corresponding to $N = 95$, 71, 49, and 33 grid points respectively. The results can be seen in Table 2. When the total number of strings is smaller than 4, always the lowest frequency strings are used.

From Table 2 it can be observed that for one string, the CPU usage is $< 6\%$ with the graphics thread disabled. This is a great result, given the fact that both the bow and the string model are computationally complex. Empirical

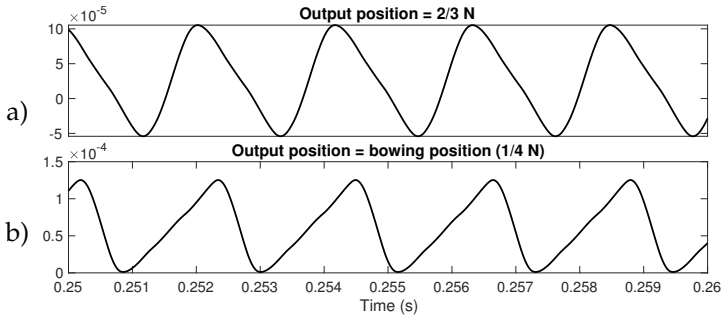

Fig. 8: *Output waveforms of the simulation at different positions along the string where N denotes the number of points of the string ($f_0 = 440$ Hz, $f_N = 5$ N and $v_B = 0.1$ m/s).*

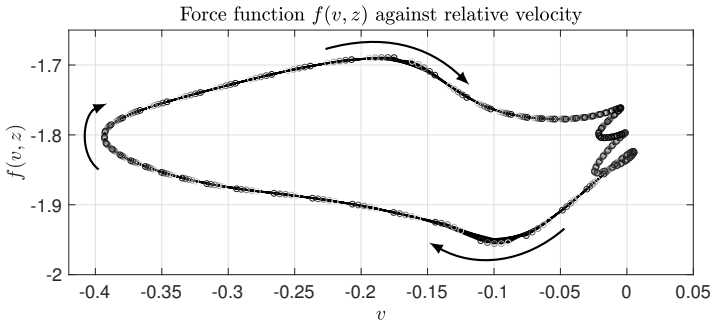

Fig. 9: *Hysteresis loop showing 500 values. The values around $v = 0$ are due to sticking behaviour and the loop on the left is due to slipping behaviour.*

investigation shows that the NR algorithm converges after ca. 3-4 iterations and the capping of 50 iterations never has to be used. A single string (but also more) could thus safely be used as an audio plugin in parallel to others without the user having to worry about auditory dropouts.

6 Conclusions

In this paper, we presented a real-time implementation of an elasto-plastic friction model with applications to a bow exciting a string, discretised using a finite-difference approach.

With a single string we are able to keep the CPU usage down to $< 6\%$ making for an efficient implementation that could be used in parallel with other virtual instruments or plugins.

Future work includes parameter design and including an instrument body for more realistic sounding results, as well as listening tests to verify the perceivable differences between simpler friction models versus the elasto-plastic

Table 2: *CPU usage for different amounts of strings. The values are averages over a 10 s period both for the enabled and disabled graphics thread. All strings are bowed simultaneously (polyphonically).*

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

model.

7 Acknowledgments

Many thanks to the anonymous reviewers for giving their valuable input. This work is supported by NordForsk's Nordic University Hub Nordic Sound and Music Computing Network NordicSMC, project number 86892.

8 References

[1] P. Dupont, V. Hayward, B. Armstrong, and F. Altpeter, "Single state elasto-plastic friction models," *IEEE Transactions on Automatic Control*, vol. 47, no. 5, pp. 787–792, 2002.

[2] S. Serafin, F. Avanzini, and D. Rocchesso, "Bowed string simulation using an elasto-plastic friction model," *SMAC 03*, pp. 95–98, 2003.

[3] J. Woodhouse, "Bowed string simulation using a thermal friction model," *Acta Acustica united with Acustica*, vol. 89, pp. 355–368, 2003.

[4] S. Serafin, "The sound of friction: Real-time models, playability and musical applications," Ph.D. dissertation, CCRMA, 2004.

[5] F. Avanzini, S. Serafin, and D. Rocchesso, "Interactive simulation of rigid body interaction with friction-induced sound generation," *IEEE Transactions on Speech and Audio Processing*, vol. 13, no. 5, pp. 1073–1080, 2005.

[6] M. McIntyre, R. Schumacher, and J. Woodhouse, "On the oscillations of musical instruments," *Journal of the Acoustical Society of America*, vol. 74, no. 5, pp. 1325–1345, 1983.

[7] J. Smith, "Efficient simulation of the reed bore and bow string mechanics," *ICMC 86*, pp. 275–280, 1986.

[8] J.-L. Florens, A. Razafindrakoto, A. Luciani, and C. Cadoz, "Optimized real time simulation of objects for musical synthesis and animated image synthesis," *ICMC 86*, pp. 65–70, 1986.

[9] J. Smith, "Physical modeling using digital waveguides," *Computer Music Journal*, vol. 16, no. 4, pp. 74–91, 1992.

[10] J. Morrison and J.-M. Adrien, "Mosaic: A framework for modal synthesis," *Computer Music Journal*, vol. 17, no. 1, pp. 45–56, 1993.

[11] L. Hiller and P. Ruiz, "Synthesizing musical sounds by solving the wave equation for vibrating objects: Part I," *Journal of the Audio Engineering Society*, vol. 19, no. 6, pp. 462–470, 1971.

[12] ——, "Synthesizing musical sounds by solving the wave equation for vibrating objects: Part II," *Journal of the Audio Engineering Society*, vol. 19, no. 7, pp. 542–550, 1971.

[13] A. Chaigne and A. Askenfelt, "Numerical simulations of struck strings. I. a physical model for a struck string using finite difference methods," *JASA*, vol. 95, no. 2, pp. 1112–1118, 1994.

[14] S. Bilbao, *Numerical Sound Synthesis*. J. Wiley & Sons, 2009.

[15] S. Bilbao, R. Hamilton, B and. Harrison, and A. Torin, "Finite-difference schemes in musical acoustics: A tutorial," in *Springer Handbook of Systematic Musicology*. Springer, 2018, ch. 19, pp. 349–384.

[16] E. Maestre, C. Spa, and J. Smith, "A bowed string physical model including finite-width thermal friction and hair dynamics," *Proceedings ICMC|SMC|2014*, pp. 1305–1311, 2014.

[17] C. G. M. Desvages, "Physical modelling of the bowed string and applications to sound synthesis," Ph.D. dissertation, The University of Edinburgh, 2017.

[18] C. Desvages and S. Bilbao, "Two-polarisation finite difference model of bowed strings with nonlinear contact and friction forces," *Proceedings of the International Conference on Digital Audio Effects*, 2015.

[19] J. Smith and J. Woodhouse, "The tribology of rosin," *Journal of the Mechanics and Physics of Solids*, vol. 48, pp. 1633–1681, 2000.

[20] S. Willemsen, N. Andersson, S. Serafin, and S. Bilbao, "Real-time control of large-scale modular physical models using the sensel morph," *Proc. of the 16th Sound and Music Computing Conference*, pp. 275–280, 2019.

[21] S. Inc., "Sensel Morph," accessed April 01, 2019, available at https://sensel.com/.

[22] J. ROLI, "JUCE," accessed April 04, 2019, available at https://juce.com/.

[23] S. Willemsen, "Elasto-Plastic Bow Model acting on a Finite-Difference Stiff String," accessed April 06, 2019, available at https://www.youtube.com/watch?v=-5bDebCW1Qg.

[24] K. J. Åström and C. Canudas de Wit, "Revisiting the lugre friction model," *IEEE Control Systems Magazine*, vol. 28, no. 6, pp. 101–114, 2008.

[25] C. Canudas de Wit, H. Olsson, K. J. Åström, and P. Lischinsky, "Dynamic friction models and control design," *Proceedings of the 1993 American Control Conference*, pp. 1920–1926, 1993.

[26] ——, "A new model for control of systems with friction," *IEEE Transactions on Automatic Control*, vol. 40, no. 3, pp. 419–425, 1995.