PREPARED PIANO SOUND SYNTHESIS

Stefan Bilbao
Music
University of Edinburgh
United Kingdom
sbilbao@staffmail.ed.ac.uk

John ffitch
Department of Computer Science
University of Bath
United Kingdom
jpff@codemist.co.uk

ABSTRACT

A sound synthesis algorithm which simulates and extends the behaviour of the acoustic prepared piano is presented. The algorithm is based on a finite difference approximation to multiple stiff string vibration, including an excitation method (a hammer) as well as several connected preparation elements, modeled as lumped nonlinearities. Numerical issues and implementation details are discussed, and sound examples are presented.

1. INTRODUCTION

The use of physical modeling principles in sound synthesis leads, eventually, to certain thorny questions. By their very nature, such methods allow very high-fidelity modeling of instrument timbres—and yet, by increasingly close attention to detail in the physics of a musical instrument, one can obtain a result which is, for musical purposes, useless: in the limiting case, an exact reproduction of the sound of an existing musical instrument. Though such sound synthesis methods are undeniably extremely important for purposes of checking the validity of a given musical instrument model, and for commercial sound synthesis applications, the focus here is on the creation of sounds which retain some of the natural texture of those produced by acoustic instruments, but which are, in some sense, new. In this regard, the use of physical models, though admirably suited to the task, is something of a balancing act.

One approach is to begin from a well-defined musical instrument, and to perform variations on its structure. This is essentially what was done early on, in the mechanical setting, in the case of prepared piano (probably best known through the work of John Cage [1], but still in extensive use up to the present day). Objects such as screws, rubber erasers, and washers (among others) are placed in proximity to the strings of a piano, yielding timbres sometimes quite foreign to the familiar piano tone—strongly percussive and bell-like tones may be generated in this way. Such an approach can be easily translated to physical modeling sound synthesis approaches, and leaves open the door to various types of preparation which are unrealizable in an actual piano (at least, not realizable without causing permanent damage to the instrument). In this study, string vibration will be simulated using finite difference schemes [2, 3, 4, 5], and various connecting or exciting elements, such as a standard hammer [2], and abstractions of preparing objects, are dealt with as lumped objects (though this is not strictly necessary). This approach is reminiscent of the mass-spring network type models used in the past [6], but the use of a fully distributed model of the string is far less unwieldy. Digital waveguides [7], [8] could also potentially be employed, but their efficiency advantage is substantially reduced under stiff string conditions, and when a string is broken up by various elements in contact with it (i.e., with a scattering junction required at each such connecting points, as well as the accompanying fractional delay machinery [9]). Another more important reason for making use of a direct simulation approach such as finite differences is that any results can easily be extended to deal with systems far more complex than simply strings.

In Section 2, a model of the stiff string, suitable for piano sound synthesis is presented, followed in Section 3 by definitions of the hammer interaction and various preparing elements. In Section 4, a finite difference scheme for multiple strings is given, including a discussion of boundary termination and numerical stability. Implementation details, particularly with regard to the csound music programming language [10] follows in Section 5. Numerical results appear in Section 6.

2. A STIFF STRING MODEL

A useful general purpose model of string vibration may be written as [3]:

$$\ddot{u} = c^2 u'' - \kappa^2 u'''' - \sigma \dot{u} + b \dot{u}'' \tag{1}$$

Here, $u(x,t)$ represents transverse string displacement in a single polarization, as a function of time $t \geq 0$ and $x \in [0,1]$. Dots and primes indicate differentiation with respect to time and space, respectively. The first term on the right-hand side is the usual wave equation term, and the parameter c is related to the fundamental frequency f_0 of the string by $f_0 = c/2$. (Note that in (1), for simplicity, the spatial variable has been non-dimensionalized, so that c does not have dimensions of velocity.) The second term represents a contribution due to string stiffness, leading to inharmonicity of partials, parameterized by the constant κ; in the limit as κ becomes large, Eq. (1) represents the behaviour of a vibrating bar. The final two terms model loss: the first, parameterized by $\sigma \geq 0$, describes the global decay rate of the string, and the second, parameterized by $b \geq 0$, approximates frequency-dependent loss characteristic of string vibration.

There are various boundary conditions which can be used to terminate the string equation (1) at either end point $x = 0$ or $x = 1$; two are required at any terminating point. In the case of strings, perhaps most useful are fixed end conditions of the clamped, i.e.,

$$u = u' = 0 \tag{2}$$

or simply supported type, i.e.,

$$u = u'' = 0 \tag{3}$$

Under low-stiffness conditions, the audible difference between the two types of conditions is negligible. If the equation (1) is intended

to represent the behaviour of a bar, then free boundary conditions may be implemented as well.

Eq. (1), though arguably suitable for piano sound synthesis [11], is by no means the most general musically useful string model; various nonlinear models, of varying degrees of complexity, allow rendering of more interesting effects such as pitch glides among others [12], [13], [14]. Any such model could easily be substituted here, with very little effect on the resulting development in this article (particularly with regard to the construction of finite difference schemes, which handle nonlinear distributed systems quite easily).

2.1. Multiple Strings and Excitation

The character of a single piano note is determined by the vibration of several strings, usually very similar in fundamental frequency, and identical in other respects. In the implementation described here, N_s such strings are coupled, each described by an equation of the form

$$\ddot{u}_{(q)} = c_{(q)}^2 u''_{(q)} - \kappa^2 u''''_{(q)} - \sigma \dot{u}_{(q)} + b\dot{u}''_{(q)} + \delta(x_0)\phi_{(q)} \quad (4)$$

where here q is an index which runs from $q = 1$ to $q = N_s$; notice that of the defining parameters, only $c_{(q)}$ has any dependence on q. Also, a set of excitations $\phi_{(q)}$, acting at $x = x_0$ on each string (through the Dirac delta function $\delta(x_0)$) has been introduced. The excitations represent the behaviour of a coupling element, such as a piano hammer, or one of various types of string preparation, which will be discussed shortly. This description of excitation or coupling is a lumped one—it is by no means necessary to make this assumption, but for many musically meaningful cases, the spatial extent of the excitation is smaller than the shortest audible wavelength in the string, for physically reasonable material sound speeds. See Section 4 for more discussion of this point.

3. EXCITATION AND PREPARATION

In this section, the hammer interaction, as well as various interactions with preparing elements are described; in each case, the element interacts with the entire set of strings. Though single elements are described here, it is simple to generalize to the case of connections to many of these objects simultaneously.

3.1. Hammers

A simple, but effective model of a hammer strike upon multiple strings from below [2] is easily described in terms of the displacement $u_H(t)$ of the hammer, and that of an individual string q at the strike location x_0, $u_{(q)}(x_0, t)$, or rather the difference $w_{(q)}(t) = u_H(t) - u_{(q)}(x_0, t)$:

$$\phi_{(q)}(w_{(q)}) = \begin{cases} \mu\gamma^2 w_{(q)}^\alpha, & w_{(q)} > 0 \\ 0, & w_{(q)} \leq 0 \end{cases}$$

In other words, a nonlinear force is exerted when the hammer and string are in contact, with the nonlinearity controlled by the exponent α (a value between 1 and 3 is generally chosen). The parameter $\mu \geq 0$ represents the relative mass density of the hammer relative to the string, and γ is a parameter which scales the stiffness of the hammer. The hammer position itself is governed by Newton's laws, i.e.,

$$\ddot{u}_H = -\frac{1}{\mu} \sum_{q=1}^{N_s} \phi_{(q)} \quad (5)$$

The hammer itself must be initialized, typically with an initial position $u_H(0)$ and an initial velocity $\dot{u}_H(0)$.

3.2. Traps

One type of string preparation is to vertically "trap" it, at a given point x_0, using nonlinear springs. In this case, the functions $\phi_{(q)}$ are given by

$$\phi_{(q)}(u_{(q)}(x_0, t)) = -\gamma^2 (u_{(q)}(x_0, t))^\alpha \quad (6)$$

where here, γ controls the spring stiffness and α is the nonlinearity exponent.

3.3. Rubber Stoppers

In order to model a rubber stopper (such as an eraser) wedged underneath a set of strings, the following model is a useful first approximation. If $u_E(t)$ is the position of the top of the rubber element, then in terms of $w_{(q)}(t) = u_E(t) - u_{(q)}(x_0, t)$, one has, similarly to the case of the hammer,

$$\phi_{(q)}(w_{(q)}) = \begin{cases} \mu\gamma^2 w_{(q)}, & w_{(q)} > 0 \\ 0, & w_{(q)} \leq 0 \end{cases}$$

In contrast to the hammer, however, the rubber stopper is anchored, and its dynamics may be written as

$$\ddot{u}_E = -\sigma_E \dot{u}_E - \gamma^2 \left(u_E + \frac{1}{\mu\gamma^2} \sum_{q=1}^{N_s} \phi_{(q)} \right) \quad (7)$$

where σ_E is a damping constant (typically quite large), and γ, in this case, can be viewed as the fundamental frequency of the rubber element. μ, in (4), is again the mass density ratio of the rubber element to the string.

An even simpler model of such a stopper involves the use of a simple damping term, i.e.,

$$\phi_{(q)}(u_{(q)}(x_0, t)) = -\sigma_E \dot{u}_{(q)}(x_0, t) \quad (8)$$

which may be used in conjunction with the trap discussed above.

3.4. Rattling Elements

A rattling element may be modeled, in the simplest case, as a two point-mass "dumbbell," of length ϵ, on which a force acts only when one of the two masses is in contact with the string. In this case, the coupling may be framed in terms of the displacement $w_{(q)}(t) = u_R(t) - u_{(q)}(x_0, t)$, where u_R is the vertical midpoint position of the rattling element. The force relationship is given by

$$\phi_{(q)}(w_{(q)}) = \begin{cases} \mu\gamma^2 \left(w_{(q)} - \epsilon/2 \right), & w_{(q)} > \epsilon/2 \\ 0, & -\epsilon/2 \leq w_{(q)} \leq \epsilon/2 \\ \mu\gamma^2 \left(w_{(q)} + \epsilon/2 \right), & w_{(q)} < -\epsilon/2 \end{cases}$$

The contact interaction here is piecewise linear, but could easily be made more strongly nonlinear through the introduction of an

exponent, as in the case of the hammer. The rattle position is again governed by Newton's laws, i.e.,

$$\ddot{u}_R = -\frac{1}{\mu}\sum_{q=1}^{N_s}\phi_{(q)} - g \tag{9}$$

where here, gravitational acceleration g has been introduced, as it will have particular importance in determining the collision dynamics of the rattle. Again, γ represents the fundamental frequency of the rattling elements, and μ in (4) the relative linear mass density of the rattle to the string.

4. A FINITE DIFFERENCE MODEL

For one-dimensional systems such as strings, finite difference schemes are a good choice of numerical method; though not as efficient as digital waveguides [7], they are simple to program, much more general, and allow a complete transparency of the entire state of the simulated system, which is especially useful if one is interested in, say, dynamically changing the position of an excitation or preparing element. As mentioned earlier, they extend easily to the case of fully nonlinear strings.

Considering the qth string, Eq. (4) may be discretized, at an interior point (to be defined presently) as

$$\begin{aligned} u^{n+1}_{(q),i} &= \sum_{l=-2}^{2}\alpha_{(q),|l|}u^{n}_{(q),i+l} + \sum_{l=-1}^{1}\beta_{(q),|l|}u^{n-1}_{(q),i+l} \\ &\quad + \frac{k^2}{h_{(q)}}\delta_{i_0}\mu\gamma^2\phi^n_{(q)} \end{aligned} \tag{10}$$

where here, $u^n_{(q),i}$ is a grid function representing an approximation to $u_{(q)}(x,t)$ at $x = ih_{(q)}$, and $t = nk$, and $\phi^n_{(q)}$ approximates $\phi_{(q)}(t)$ at $t = nk$, and at grid location $i = i_0$ corresponding to $x_0 = ih_{(q)}$ (hence the use of the Kronecker delta). $h_{(q)}$ is the spacing between adjacent grid points, and k is the time step ($1/k$ is the sample rate, generally set a priori). The scheme coefficients $\alpha_{(q),l}$ and $\beta_{(q),l}$ are given by

$$\begin{aligned} \alpha_{(q),0} &= \frac{2 - \frac{2c^2_{(q)}k^2}{h^2_{(q)}} - \frac{6\kappa^2k^2}{h^4_{(q)}} - \frac{2bk}{h^2_{(q)}}}{1+\sigma k/2} \\ \alpha_{(q),1} &= \frac{\frac{c^2_{(q)}k^2}{h^2_{(q)}} + \frac{4\kappa^2k^2}{h^4_{(q)}} + \frac{bk}{h^2_{(q)}}}{1+\sigma k/2} \\ \alpha_{(q),2} &= -\frac{\kappa^2k^2}{h^4_{(q)}(1+\sigma k/2)} \\ \beta_{(q),0} &= \frac{\sigma k/2 - 1 + \frac{2bk}{h^2_{(q)}}}{1+\sigma k/2} \\ \beta_{(q),1} &= \frac{-\frac{bk}{h^2_{(q)}}}{1+\sigma k/2} \end{aligned}$$

A necessary stability condition, which follows from von Neumann (Fourier) analysis of scheme (10) under zero input conditions [15], is

$$h_{(q)} \geq h_{(q),min}$$

with

$$h_{(q),min} = \sqrt{\frac{c^2_{(q)}k^2 + 2bk + \sqrt{(c^2_{(q)}k^2+2bk)^2 + 16\kappa^2k^2}}{2}}$$

In other words, for a given sample rate and set of material parameters, a minimum spacing between adjacent points must be respected. This spacing relates to the highest audible wavelength in the string. A slight additional complication comes from the fact that the scheme for each string will possess its own separate stability condition, giving rise, potentially, to different grid spacings on each string, which is inconvenient from a programming standpoint; it is useful to make use of a global spacing h over all strings, bounded by

$$h \geq \max_q h_{(q),min} \tag{11}$$

Use of such a global spacing is advisable only when the fundamental frequencies of the strings are relatively close; otherwise, one may be far from the ideal bound in at least some of the strings, and numerical dispersion [15] will play a significant role, leading to a numerical inharmonicity in partials.

Eq. (10) is implemented as a two-step recursion; it may be initialized at zero. It is also clear that any given grid function $u^n_{(q),i}$, which is defined over a set of grid points indexed by $i = 0, \ldots, N$, where $N = 1/h$, will require access to values outside the problem domain at points near the boundary, according to (10). For fixed termination, as is the case presented here, one may always set $u^n_{(q),0} = u^n_{(q),N} = 0$; in other words, the terminal grid points are permanently set to zero, and need not be computed, satisfying the first of conditions (2) or (3). To satisfy the second of conditions (2) at, say, the left end of the string, one may set $u^n_{-1} = 0$. To satisfy the second of conditions (3), one may set $u^n_{-1} = -u^n_1$. Many other methods of setting the boundary conditions are feasible, and details may be found elsewhere; these particular settings are simple, and provably numerically stable.

4.1. Updating for the Connecting Elements

The various connecting elements described in Section 3 are also, for the most part, described by differential equations, and require numerical integration. Though in the case of nonlinear elements, there are obviously many ways of performing this integration, simple methods are presented here. In the case of the hammer, (5) may be updated as

$$u^{n+1}_H = 2u^n_H - u^{n-1}_H - \frac{k^2}{\mu}\sum_{q=1}^{N_s}\phi^n_{(q)}$$

The rattling element, defined by (9) may be dealt with similarly as

$$u^{n+1}_R = 2u^n_R - u^{n-1}_R - \frac{k^2}{\mu}\sum_{q=1}^{N_s}\phi^n_{(q)} - k^2g$$

For the rubber stopper, one may use

$$\begin{aligned} u^{n+1}_E &= 2u^n_E - u^{n-1}_E - \sigma_E k(u^n_E - u^{n-1}_E) \\ &\quad -\gamma^2k^2\left(u^n_E + \sum_{q=1}^{N_s}\phi^n_{(q)}\right) \end{aligned}$$

The trapping element requires no additional state, and to approximate (6), one may simply use

$$\phi_{(q)}(u^n_{(q),i_0}) = -\gamma^2 (u^n_{(q),i_0})^\alpha$$

If loss is introduced, one may use, as an approximation to (8),

$$\phi_{(q)}(u^n_{(q),i_0}) = -\frac{\sigma_E}{k}(u^n_{(q),i_0} - u^{n-1}_{(q),i_0})$$

5. IMPLEMENTATION DETAILS

5.1. Operation Count

The bulk of the calculation in this prepared piano simulator is carried out in performing the update of the string state; hammers and various connections are of negligible computational cost. For a single string, of fundamental frequency f_0, and under low inharmonicity conditions, at sample rate f_s the number of multiplies per second is approximately $2f_s^2/f_0$; the computational cost obviously scales inversely with the fundamental frequency.

5.2. csound Implementation

This algorithm was prototyped in the Matlab programming language, but the aim is to make the sound available to composers. Ideally such an implementation should be playable in real time, or at least close to real time, and from the point of view of practicality there needs to be some attempt to minimise the memory use. For composers, the flexibility of the operation is also significant; csound [10] would appear to be a good choice of programming environment.

The internal operations naturally rely on three arrays to represent the values $u^{n-1}_{(q),i}$, $u^n_{(q),i}$ and $u^{n+1}_{(q),i}$, for each string. At each time step these are cycled by pointer assignment, taking care to correct for the boundary conditions. Similar arrangements are taken to maintain the state of the preparations.

In Csound, as in all MusicV descended systems, the operations are split between an initialisation method and then a perform method that creates a short section of audio. Much of the speed of csound comes from this batching of operations, with its attendant good cache behaviour and the less frequent adjustment of parameters. In the case of the prepared piano there are twelve fundamental parameters covering basics like fundamental frequency and the number of strings, as well as stiffness, loss parameters and the characteristics of the hammer, most of which cannot reasonably be adjusted after the initialisation (although there is a future action to investigate this more closely). The harder problem is how to specify the various preparations. Our implementation uses two f-tables, one to define the characteristics of rubber preparations and one to do likewise for the rattles. Either of these can of course be omitted.

The other design question is how to handle multiple hammer strokes. Our implementation does this by having one stroke for each call, but it is possible to skip the initialisation phase, and so keep the previous state of the system. There is also an issue on control of the amplitude of the output. As in many physical models the parameters determine the inputs rather than the resulting amplitude. The code performs some scaling based on experience.

```
<CsoundSynthesizer>
<CsInstruments>
nchnls = 1;
instr 1
;; fund NS detune stiffness decay loss (bndry)
;; (hammer) scan prep
aa prepiano 60, 3, 10, p4, 3, 0.002, 2, 2, 1,
5000, -0.01, p5, p6, 0, 1, 2
out aa
endin
</CsInstruments>
<CsScore>
f1 0 8 2 1 0.6 10 100 0.001 ;; 1 rattle
f2 0 8 2 1 0.7 50 500 1000  ;; and 1 rubber
i1 0.0 0.5  1 0.09 20
i1 0.5  .  -1 0.09 40  ;; 2nd strike keeping state
i1 1.0  .  -1 0.09 60
i1 1.5   . -1 0.09 80
i1 2.0 1.8 -1 0.09 100 ;; last strike until silence
</CsScore>
</CsoundSynthesizer>
```

Figure 1: *Simple csound example.*

5.3. Performance

The csound opcode has been tested *in situ* in two simple cases, with one rattle and one rubber stopper, and with no preparations. Clearly the processor speed is important when we are concerned with how close to real time the code runs. The tests were run on two widely differing systems both running Csound5.01; a 1.4GHz Centrino laptop with Linux with floating-point samples, and an AMD Athlon64 3400+ (2.4GHz) with Linux, and double-precision samples. The actual test program is shown in Figure 1, where a very low frequency string ($f_0 = 60$ Hz) has been chosen, as a "worst case."

Machine	Time	% real	Time	% real
Laptop	8.143	47%	8.239	46%
Workstation	0.591	643%	0.595	639%

The preliminary timings suggest that the cost of the preparations is small. The workstation is clearly showing usable performance while on the laptop the speed is too slow for real time, but not too bad for some uses. These results are preliminary, and there may be opportunities for further optimisation, such as moving some calculations from the sample-rate loop to the control rate.

6. NUMERICAL RESULTS

In this section, several illustrative plots of numerical output from the string model, under various types of preparation, are presented; all were created at the audio sample rate of 44.1 kHz. The effect of brightening with strike velocity for a nonlinear hammer of the type presented here has been discussed elsewhere [2]. Considering the case of a rattling element, in Figure 2 are shown output waveforms generated, as well as spectrograms, in the case of a struck string at fundamental frequency 600 Hz, both with and without a rattling element present. Though it is difficult to show the precise time-dependent effects of the rattling element, at the crudest level, one sees a considerable disruption of the output waveform, as well as a substantial presence of high frequencies in the case of the rattle.

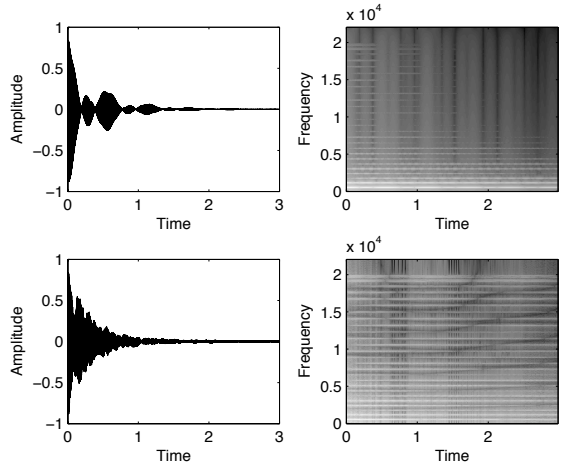

Figure 2: *Top: output waveform and spectrogram, for a struck piano tone at* 600 *Hz. Bottom: output waveform and spectrogram for a struck piano tone at* 600 *Hz, including the effects of a rattling element placed* 0.3 *of the way along the string from the left end.*

Consider the case of a trapping element, in the case of a string of fundamental frequency 200 Hz; a spectrogram of the output waveform is given in Figure 3. The string is struck successively with blows of initial velocity 5, 10, 50 and 100 m/s. Notice in particular the increase in high frequency energy with strike velocity (due to the effect of both the nonlinear hammer interaction, and the trapping element), as well as the rather complex effects of modulation visible in the upper harmonics of the tone under high amplitude striking velocity.

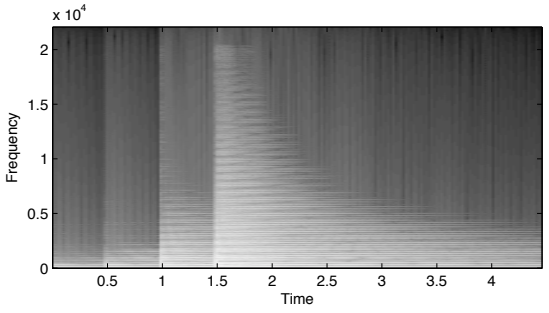

Figure 3: *Top: output spectrogram, for a struck piano tone at* 200 *Hz, with a trap placed* 0.3 *of the way along the string from the left end, under strikes of increasing amplitude.*

Under the same conditions as the example above, but with the addition of a rubber stopper (in this case at the same location as the trapping element), the output sounds become, predictably, shorter and more percussive (see Figure 4) with the additional feature that, under high amplitude conditions, a new, higher fundamental frequency emerges. The effect of a prepared piano sounding at a pitch different from that of the note struck is, of course, a well-known effect.

It is rather difficult to give the full flavor of the sound examples produced by presenting plots of waveforms and spectrograms; the reader is directed to a series of sound examples, created in Matlab, which are available on the internet [16].

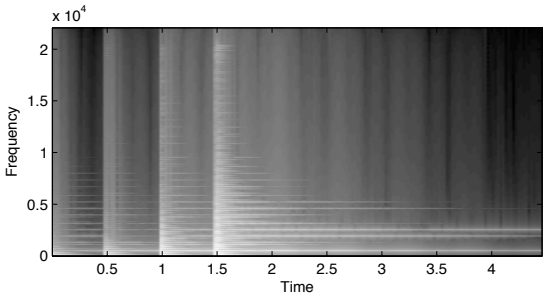

Figure 4: *Top: output spectrogram, for a struck piano tone at* 200 *Hz, with a trap accompanied by a rubber stopper placed* 0.3 *of the way along the string from the left end, under strikes of increasing amplitude.*

7. CONCLUSION

In this short article, a brute force numerical simulation of a prepared piano has been presented. As noted in Section 5, though more computationally demanding than techniques such as digital waveguides, these methods are already capable of real time performance.

One significant problem with the algorithm as presented here has to do with numerical stability concerns, with regard to the nonlinear interactions with the hammer and various preparing elements. The condition (11), which is derived using Fourier principles applied to the string scheme under linear conditions is merely necessary. One may develop energy-conserving or dissipating algorithms to deal with the various nonlinearities (indeed, in all cases, the model, including preparing elements, may be shown to be strictly dissipative) [17], but such algorithms may lead to implicit schemes, necessarily requiring the use of iterative methods, without a guarantee of convergence. Another approach would be to make use of so-called symplectic methods [18], but this approach is not strictly applicable to problems involving dissipation. The question is open as to how to construct robust (i.e. provably numerically stable) numerical methods for complex nonlinear systems such as that presented here.

It is worth noting that although the algorithm can be used to simulate the behaviour of a piano-like instrument, the model equation, as defined by (1) is quite general, and can be used, without modification to simulate the behaviour of various other percussion instruments, including bar-based instruments such as xylophones or marimbas, again under prepared conditions. One element which is missing here is a physical model of a soundboard, which can be easily modeled through the use of a finite difference plate model, connected to the string models mentioned here. Preparation of such a soundboard itself is, of course, a desirable option.

8. ACKNOWLEDGEMENTS

This work was supported in part by the Engineering and Physical Sciences Research Council UK, under grant number C007328/1.

9. REFERENCES

[1] J. Cage and D. Charles, *For The Birds: John Cage in Conversation with Daniel Charles*. Marion Boyers, 1981.

[2] A. Chaigne and A. Askenfelt, "Numerical simulations of struck strings. I. A physical model for a struck string using finite difference methods," *J. Acoust. Soc. Am.*, vol. 95, no. 2, pp. 1112–8, Feb. 1994.

[3] J. Bensa, S. Bilbao, R. Kronland-Martinet, and J. O. Smith III, "The simulation of piano string vibration: From physical models to finite difference schemes and digital waveguides," *J. Acoust. Soc. Am.*, vol. 114, no. 2, pp. 1095–1107, 2003.

[4] P. Ruiz, "A technique for simulating the vibrations of strings with a digital computer," Master's thesis, University of Illinois, 1969.

[5] L. Hiller and P. Ruiz, "Synthesizing musical sounds by solving the wave equation for vibrating objects," *J. Audio Eng. Soc.*, vol. 19, pp. 462–72, 542–51, 1971.

[6] C. Cadoz, A. Luciani, and J.-L. Florens, "CORDIS-ANIMA: a modeling and simulation system for sound and image synthesis – The General formalism," *Computer Music J.*, vol. 17, no. 1, pp. 19–29, 1993.

[7] J. O. Smith III, *Physical Audio Signal Processing*. Stanford, CA: draft version, 2004, [Online] http://ccrma.stanford.edu/jos/~pasp04/.

[8] T. Tolonen, V. Välimäki, and M. Karjalainen, "Modeling of tension modulation nonlinearity in plucked strings," *IEEE Trans. Speech and Audio Proc.*, vol. 8, pp. 300–310, May 2000.

[9] T. I. Laakso, V. Välimäki, M. Karjalainen, and U. K. Laine, "Splitting the unit delay: Tools for fractional delay filter design," *IEEE Sig. Proc. Magazine*, vol. 13, no. 1, pp. 30–60, Jan. 1996.

[10] R. Boulanger, *The Csound Book: Perspectives in Software Synthesis, Sound Design, Signal Processing and Programming*. Cambridge, Massachusetts, USA: MIT Press, 2000.

[11] B. Bank and L. Sujbert, "Generation of longitudinal vibrations in piano strings: From physics to sound synthesis," *J. Acoust. Soc. Am.*, vol. 117, no. 4, pp. 2268–2278, Apr. 2005.

[12] C. Vallette, "The mechanics of vibrating strings," in *Mechanics of Musical Instruments*, A. Hirschberg, J. Kergomard, and G. Weinreich, Eds. New York: Springer, 1995, pp. 116–183.

[13] S. Bilbao and J. O. Smith III, "Energy conserving finite difference schemes for nonlinear strings," *Acta Acustica united with Acustica*, vol. 91, pp. 299–311, 2005.

[14] S. Bilbao, "Conservative numerical methods for nonlinear strings," *J. Acoust. Soc. Am.*, vol. 118, no. 5, pp. 3316–3327, 2005.

[15] J. Strikwerda, *Finite Difference Schemes and Partial Differential Equations*. Pacific Grove, Calif.: Wadsworth and Brooks/Cole Advanced Books and Software, 1989.

[16] S. Bilbao, Retrieved June 29th, 2006, [Online] Plate sound synthesis examples at http://www.music.ed.ac.uk/Contacts/DrStefanBilbao_soundExamples.htm.

[17] ——, "Robust physical modeling sound synthesis for nonlinear systems: Direct numerical simulation and the energy method," 2006, under review, IEEE Signal Processing Magazine. Invited article.

[18] J. Sanz-Serna, "Symplectic integrators for hamiltonian problems: An overview," *Acta Numerica*, vol. 1, pp. 243–286, 1991.