DESIGN AND IMPLEMENTATION OF REAL-TIME PHYSICALLY-BASED VIRTUAL MUSICAL INSTRUMENTS: A BALANCING ACT

James Leonard
Univ. Grenoble Alpes, CNRS, Grenoble INP*
GIPSA-Lab, 38000 Grenoble, France
james.leonard@gipsa-lab.fr

Jerome Villeneuve
Univ. Grenoble Alpes, CNRS, Grenoble INP*
GIPSA-Lab, 38000 Grenoble, France
jerome.villeneuve@gipsa-lab.fr

ABSTRACT

In recent years, real-time physically-based solutions for sound synthesis have become a common practice both in academia and commercial applications, driven jointly by the increase in computing power and advances in modelling and simulation techniques. However, designing and implementing such instruments and the means for expressive interaction with them still poses several challenges, factors to be balanced by the instrument creator depending on his or her needs. This paper discusses these concerns through the prism of mass-interaction physical modelling. First, a short review on the topics of computation and expressive control of real-time physical models is proposed, followed by the introduction of the physical proxy, a new mass-interaction modelling element yielding new possibilities for expressive interaction via dynamically interpolated topological connections. Three case-studies of physically-based virtual musical instruments are then presented and discussed, particularly in regards to computational aspects, with benchmarks of various possible implementations. Finally, we offer some insight on the balance that one must consider between model and control complexity, and between software genericity and performance, when building physical models for real-time sound synthesis.

1. INTRODUCTION

Physical models have been used creatively for sound synthesis and musical composition purposes for over three decades [1], spanning from the simulation of large virtual worlds [2] to virtual musical instruments [3, 4]. While in the early 90s real-time simulation was mostly limited to waveguide modelling [5], today nearly all physically-based methods may be computed and controlled interactively for virtual musical instrument simulation. Indeed, advances in hardware and computing power and breakthroughs in modelling techniques now allow simulating fairly large-scale virtual acoustical structures, and to increasingly account for the non-linear dynamics at play in musical acoustics [6, 7]. A number of formalisms and software tools are now at the disposal of musicians, composers and virtual instrument designers, or indeed anyone interested in making sound through physical simulation.

This paper aims to discuss some questions that arise in the design and development process of real-time physical models for musical creation, starting with general concerns regarding computation and expressive control, and then analysing three case-studies of physically-based digital musical instruments, centred on the mass-interaction paradigm (although bearing strong similarities with finite difference approaches, as will be discussed). From this work, we then offer a discussion on the "balancing act" of developing physically based virtual instruments: balance between model complexity and speed, balance between dynamical richness of interaction and acoustical richness of simulated vibrating structures, balance between software genericity / clarity (high-level code, abstractions, etc.) and hand-tailored optimisation (low-level code). Finally, we conclude with perspectives on future work.

2. REAL-TIME CONTROL OF PHYSICALLY-BASED SOUND SYNTHESIS

2.1 Computational Aspects

Once primarily considered as a costly endeavour best suited for off-line acoustical simulation, physical modelling techniques are now deployed in a variety of academic and commercial real-time systems [8–10]. While digital waveguides (at least in 1D) offered a strikingly economical solution for real-time sound synthesis as soon as they appeared, other more computationally demanding techniques have only really grown into viable real-time solutions for complex physical models in the last 15 years, jointly due to efforts in simulation optimisation and the increase in computing power and dedicated hardware (including vectorisation, multithreading, GPU-based computation [11, 12]). Several works now allow for real-time sound-synthesis with mass-interaction models [13], modal synthesis, or finite difference time-domain schemes [6]. In particular, the latter have made impressive progress into real-time territory, through tailored optimisation techniques [14] and advances into efficient solvers for non-linear problems [15].

It is worth noting that while all physical-modelling sound synthesis techniques allow representing 1D, 2D or 3D physical topologies or spatial grids, the vast majority consider the discretised matter to move along a single degree of freedom (1-DoF). This approach, rooted in theoretical acoustics,

*Institute of Engineering, Univ. Grenoble Alpes.

has the advantage of yielding mostly linear systems that in turn allow using many convenient mathematical tools (such as modal decomposition & analysis). The extension of such formalisms to account for the numerous non-linear phenomena present in physical acoustical systems is very much an ongoing effort [15, 16], generally involving rather heavy mathematics and computation. In this regard, 3-DoF mass-interaction techniques [7] (drawing from the wide use of 3D mass-spring models in computer graphics) offer an interesting gateway into inherently non-linear territory, although their computational cost is certainly higher than simple 1-DoF systems.

2.2 Means for Expressive Control

Physical modelling is largely employed in Digital Musical Instruments, offering means for creative exploration [17] as well as multisensory interaction through force-feedback technologies [10, 18, 19]. Traditional MIDI often falls short in installing expressive interaction with sound-producing models (especially for continuous excitation such as bowing), whereas more sophisticated input devices such as the Sensel Morph [8] or Roli Seaboard [1] show promising results thanks to precise multi-DoF continuous control data.

A key component for expressive interaction with virtual instruments is to provide means to perform both *excitation* (e.g. striking, bowing) and *modification* gestures (e.g. moving a bow application point or a finger along a string, moving a pressure point across a membrane or plate). Dynamically constraining physical matter in this way generally sounds much more plausible than direct manipulation of physical model parameters, which understandably can lead to "non-physical" sounding effects. Interpolation strategies may be employed to account for interaction points located between two points of discretised space or matter [6]. To the best of our knowledge, common mass-interaction sound synthesis frameworks such as [20,21] lack such possibilities, hindering expressive control potential (a notable exception being ad hoc implementations by Alexandros Kontogeorgakopoulos for cross-fading between listening points to obtain physically-based audio-effects in [22]). Hereafter, we present a general solution in the form of dynamically interpolated topological connections based on standard interpolation schemes.

3. EXPRESSIVE CONTROL OF MASS INTERACTION MODELS

3.1 Spatial and Topological Considerations

Most mass-interaction toolkits operating on 1-DoF particles define each physical interaction statically between two punctual masses of the system. For instance, a collision between a hammer and a string is modelled as one (or possibly several) point-based interaction(s) between the hammer and statically defined string mass(es). This offers little flexibility: the string will always be struck at the same spot, yielding similar acoustic responses that are only modulated by the relative velocities of the masses. Such punctual connections prohibit modification gestures such as those mentioned in Section 2.2, often crucial in adding expressiveness and liveliness to synthesised sound.

[1] Examples of real-time MPE control include Physical Audio's coupled-bars plugin (https://physicalaudio.co.uk/PA3) and SWAM's bowed string models (https://roli.com/stories/swam-soundpacks).

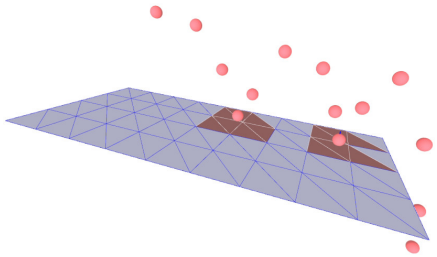

Figure 1. A 3D mass bouncing on a 3-DoF mesh with surface contact computation. Surfaces in red represent active point/plane collision detection and force projection when the mass is close or in contact with the mesh triangles.

2 or 3-dimensional mass-interaction models may alleviate this problem, as they possess geometrical properties: a hammer mass may potentially collide with any mass of a vibrating structure. However, handling these cases can be rather involved and computationally heavy, relying on 3D collision engines and possibly - if one wishes to go further than *mass-to-mass* interactions - resorting to surface meshes defined atop of the physical topology, allowing to calculate force projections onto the masses that compose their vertexes (cf. Figure 1). Given the cost of simulating 3D mass-interaction models at audio rate, this solution is currently ill-fitted for implementing large scale models with many degrees-of-freedom of control.

3.2 Interpolated Input, Output and Coupling Points by Means of Physical Proxies in 1-DoF Models

Interpolation strategies allowing to handle inputs, outputs, and coupling points that do not match the scattering junctions of a finite difference grid or a waveguide network (be it one, two or three-dimensional) have been studied in a number of works such as [6, 23, 24].

In an effort to integrate these concepts into the framework of 1-DoF mass-interaction networks, we introduce the physical *virtual proxy* as an economical way to represent and compute smooth interpolation between topological connections. In essence, a proxy enables to dynamically couple an interaction point to a macro-structure (for instance a string or a mesh with a known topological organisation). The interaction point can therefore be placed between actual physical masses, and move smoothly along the structures, including at audio rates.

Excluding zero order truncation or rounding methods, the most basic interpolation strategy for an interpolated point p located between two grid points m_1 and m_2 with a normalised location coefficient $0 < \alpha < 1$ as shown in Figure 2 is first order Lagrange (or linear) interpolation, which gives coefficients:

$$w_1 = (1 - \alpha) \quad , \quad w_2 = \alpha \tag{1}$$

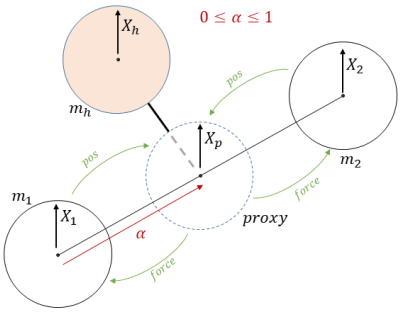

Figure 2. The first order 1D proxy, allowing for interpolated topological connections between two masses.

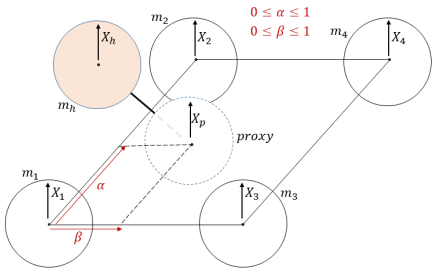

Figure 3. The 2-D "mesh" bilinear interpolation proxy, allowing interpolated topological connections between the four points of a rectangular mesh.

This extends to the bilinear transform in the 2D case [23], with masses $m_{1..4}$ and additional coefficient β defining the location along the second axis (cf. Figure 3):

$$\begin{aligned} w_1 &= (1-\alpha)(1-\beta) \quad , \quad w_2 = \alpha(1-\beta) \\ w_3 &= (1-\alpha)\beta \quad , \quad w_4 = \alpha\beta \end{aligned} \tag{2}$$

Certain limitations of linear interpolation are pointed out in [6], notably possible low-pass filtering effects that could be addressed using higher order interpolators such as cubic Lagrange, requiring a larger stencil of points. For simplicity, in the following we will assume first order Lagrange interpolation (which is also employed in the examples presented in Section 4).

Integration of the interpolators into the mass-interaction paradigm is as follows: the proxy p has no mass property, and its physical position X_{proxy} is dynamically calculated at each simulation step according to the "spatial" location given by the α and β parameters:

$$X_{proxy} = \frac{1}{N} \sum_{i=1}^{n} w_i . X_i \tag{3}$$

Interaction forces are then calculated normally between the proxy and the mass interacting with it (denoted m_h with position X_h) depending on the type of interaction (e.g. spring, collision, friction, etc.) :

$$\begin{aligned} F_{proxy} &= F_{interaction}(X_h, X_{proxy}) \\ F_h &= -F_{proxy} \end{aligned} \tag{4}$$

The equal and opposite interaction force is applied directly to the mass m_h, whereas F_{proxy} (the force applied to the proxy) is redistributed into forces applied to the actual masses using the same interpolated weighting :

$$\begin{aligned} F_{proxy} &= \sum_{i=1}^{n} F_i \\ F_i &= w_i . F_{proxy} \end{aligned} \tag{5}$$

In a static regime (fixed α, β parameters) the use of a proxy is mathematically equivalent to n weighted point-based interactions with the n-mass underlying structure. Displacement of the proxy results in smooth variation of physical interaction parameters and topological connections, either moving inside a group of masses, or moving from one group of masses to another.

As such, physical proxies constitute a means to approximate linear or surface interactions that may be obtained in 3D geometry (cf. Figure 1) at a fraction of the computational cost, by adding "artificial" Y and Z degrees of freedom to move connection points dynamically within a 1-DoF physical model. Hereafter, we will discuss the interest of this solution for the design of real-time large scale models offering means for expressive control.

4. DESIGN AND IMPLEMENTATION OF THREE PHYSICALLY-BASED VIRTUAL INSTRUMENTS

We now present three case-studies of virtual musical instruments developed over the last year using the mass-interaction paradigm. For each, we will present elements pertaining to the computational aspects (model complexity, optimisation, benchmarking) and discuss design choices and implementation of real-time control strategies. We will also try to highlight the expressive potential of these instruments through examples and/or by reporting their use in recent or upcoming artistic works.

4.1 A Bowed String Drone Instrument

This instrument was designed in collaboration with the French composer Eddy Jaeber and forms the basis of his piece entitled "Le Cosmos Pour Acoustique". It features several strings that may be bowed or plucked and whose pitch may be altered by sliding "fretting fingers".

4.1.1 Physical Model and Implementation

The mass-interaction model of the instrument is shown in Figure 4. It contains four 160-mass stiff strings, wherein each mass is connected to its immediate neighbours via spring dampers, and also to its second neighbours through springs with a dedicated "stiff" term. Bowing interactions are connected to each string through proxies and distributed over three application points (thus approximating non-null bow width). The fretting finger interactions clamp strings against a fretboard (located below) at a given length. Additional sliding damping interactions are placed behind the fretting point in order to dampen any residual vibrations in the string section located behind the finger.

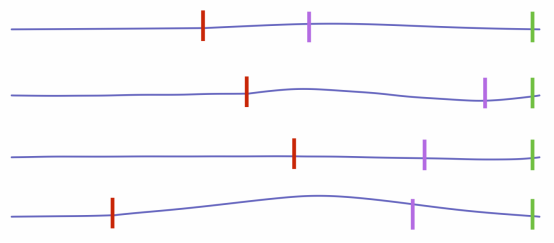

Figure 4. Bowed string real-time visualisation in Max/MSP. Blue lines show the motion of each string. Finger positions in red, bow positions in purple, listening points in green.

The instrument was implemented directly in Max/MSP using the mi-gen~ toolkit [25], driving the physical model with data-rate or audio-rate signals from within a patch that also handles visualisation processes (shown in Figure 4) and control logic such as voice allocation. The simulation uses approximately 55% of the CPU monitor in Max/MSP [2] which, although it is non-negligible for a model of this size, poses no problem for real-time audio computation.

4.1.2 Real-time control

A Sensel Morph device with the MIDI overlay is used to control the model, providing precise Polyphonic Expression (MPE) information, dynamically allocating voices to available strings. The mapping is as follows :

- Touch pressure is mapped to bowing pressure and bow velocity.
- MIDI notes and per-note pitch bend (lateral motion) control the finger position on the current string.
- Vertical motion is mapped to the bow position on the string.

In *pizzicato* mode, note-on messages and touch pressure are used to pull the strings back through a contact interaction, and note-off messages release the contact.

4.1.3 Use in the piece "Le Cosmos Pour Acoustique"

The composition is a sound construction built with multiple layers of audio from the instrument. It explores various playing modalities discovered by the player himself during preliminary experimentation, including some rather extreme cases : very short and high pitched strings, slowly evolving infra-bass drones, progressive shifting between string harmonics by moving the bow position, bowing extremely close to the bridge/finger (producing unstable "screeching" tones), etc. The resulting sounds do not mimic a specific bowed instrument reference (such as the violin) but they are highly controllable, aided by sonic and visual feedback.

4.2 A Tangible Bowed Mesh

The tangible bowed mesh came from the idea of providing many dynamically moving interaction points with a relatively simple, yet large, single acoustical structure. The model is a 2-dimensional mesh with closed boundary conditions, excited by up to ten bowing friction interactions at positions defined by "mesh" proxies.

[2] Test machine specifications are detailed in Section 4.2.1.

Implementation	**CPU Usage** (Max/MSP)
mi-gen~	100%
C++ MI basic	30%
C++ MI shaved down	22%
C++ MI autovector	20%
C++ Finite Difference	18%

Table 1. Benchmarking results for a 30 x 35 closed mesh (total of 1050 masses), simluated at 44.1 kHz.

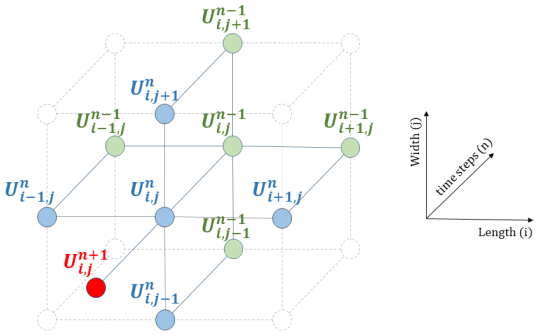

Figure 5. Equivalent finite difference scheme for the 2D mass-interaction mesh. Next value to compute (red) from current (blue) and previous (green) values.

4.2.1 Implementation and benchmarking

Performance rapidly becomes a concern when simulating meshes or plates. A Max external was therefore implemented directly in C++ using the *min-dev-kit* package [3] . The model is computed with a specifically-coded mass-interaction simulation engine using static pre-allocation of contiguous memory blocks for the model state data (positions, forces) in double-precision floating-point format.

Benchmarks for a *30 x 35* mesh were measured on a 6-core 2.60GHz Intel Core i7-8850H CPU, compiling the Max external with Visual Studio 2019 using the *02* optimisation and *\fp:fast* floating point model flags. The results are reported in Table 1 and discussed below:

- mi-gen~ performance is largely inferior to any of the hand coded C++ implementations, choking at 100%.
- Standard C++ mass-interaction algorithms lower this to 30%, and further "shaving" of the code (avoiding function calls, using pointer-based access, etc.) brings this down to 22%.
- Algorithms can be made autovectorisation [4] "friendly" to some extent by simplifying loop code and using the `restrict` keyword to avoid pointer aliasing.

[3] http://cycling74.github.io/min-devkit/

[4] an optimisation consisting in the simultaneous computation of data organised into vectors, typically of 128, 256 or possibly 512 bits.

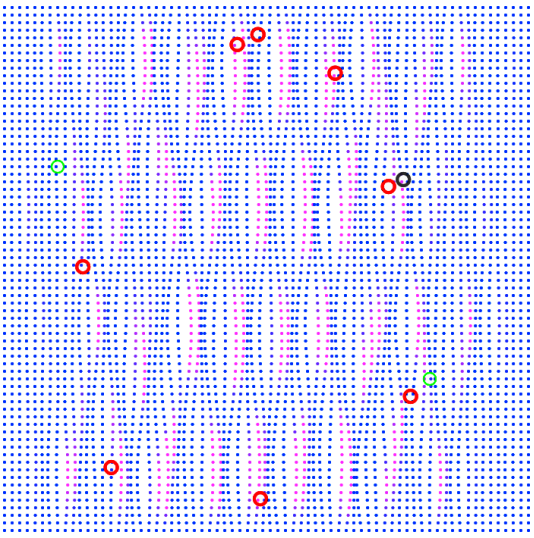

Figure 6. Bowed mesh model, showing active (red) and inactive (grey) bowing points and listening points (green). The mesh masses (in blue, turning to red as a function of displacement) are connected to top, down, left and right neighbours through spring dampers (not displayed).

This allows vectorising the mass updates, but not the interactions as they require non-contiguous access into array data. The gain is therefore quite minimal. Manually vectorised code for the mesh topology is not trivial, although it can be done [11, 14].

- Our mass-interaction mesh is equivalent to a ten-point stencil explicit finite difference scheme for the 2D wave equation (a standard 5-point Laplacian approximation for the undamped 2D wave as presented in [6], with 5 extra points induced by the damping terms at time $n-1$), shown in Figure 5. Therefore, the position update scheme may be expressed as a function of current and previous positions covered by the stencil and external forces (merging mass and interaction terms into one update). This offers yet another slight improvement, bringing the cost down to 18%. It is currently the most efficient option.

Using the finite-difference implementation, mesh size can be increased up to 70 x 70 (4900) masses, as shown in Figure 6.

4.2.2 Real-time control

As for the bowed string instrument, interaction with the model is handled via the Sensel Morph, this time directly using raw contact point data: each point acts as a bow positioned at the *X-Y* location on the device, with bowing velocity and pressure driven by applied contact pressure. A video showcasing the expressive possibilities of this dynamical multi-point interaction is provided [5] .

[5] Video demonstrations of all three virtual instruments can be found at the following address: http://mi-creative.eu/smc2020

4.3 A Virtual Classical Harp

The final presented instrument stems from an ongoing collaboration between Gipsa-Lab's Digital Arts and Sensory Immersions group, composer Arnaud Petit, and instrument manufacturer CAMAC Harps. The latter have developed an augmented electric diatonic harp, equiped with piezo sensors for each string, offering real time per-string audio channels as well as MIDI/OSC tracking. The piece *Orbis* explores the capabilities of this instrument, while confronting it with a virtual counterpart, playing with the limits of what is physically possible to perform on both the real and the virtual instruments.

4.3.1 Model and Implementation

Designing the virtual harp model is a matter of compromise, as the scale of the instrument alone makes real-time simulation a challenge. Among other things, simulating sufficient brightness in the lower register calls for tense and extremely long strings (approximately 505 masses for the low C). What's more, real plucked harp strings contain noticeable tension effects (pitch glides in the lower notes). The interactions between the tuning clamps operated by the pedals and the strings involve complex contacts that may generate (wanted or unwanted) noise on the instrument, including from strings that are not plucked, as they are lightly excited whenever the pedals are operated (each pedal changes all octaves of a given natural note up or down a semitone). And last but not least, the body of the instrument plays a large part in its acoustical output.

The necessity for a robust real-time solution integrated into a large musical piece alongside other audio processes leads to pragmatic solutions to these aspects:

- Ideal linear 1D strings, scaled and tuned by numerical resolution so that the highest mode of each string lies at approximately 11kHz.
- Tuning the strings up and down artificially by direct manipulation of the stiffness parameter when pedal-induced alterations occur [6] . Pitch glide may be accounted for by triggering additional exponentially decaying stiffness terms, raising the initial pitch of plucked strings by a velocity-dependant amount.
- Coupling via the body modelled by two "bridge" oscillators, forming common termination points for the strings. Inertia, stiffness and damping properties of these oscillators allow tuning sympathetic resonance throughout the instrument.

The resulting physical model contains 5500 masses for a total of approximately 13000 modules. It is triggered by MIDI or OSC data, either directly issued from the augmented harp or from score following patches that synchronise a score for the virtual harp with the score played on the real instrument.

4.3.2 Implementation challenges: AVX2 vectorisation

While the above model does run in real time using the same C++ framework as the bowed mesh model presented in

[6] The effect in altered string brightness is barely noticeable, if at all, for changes of the order of a semitone.

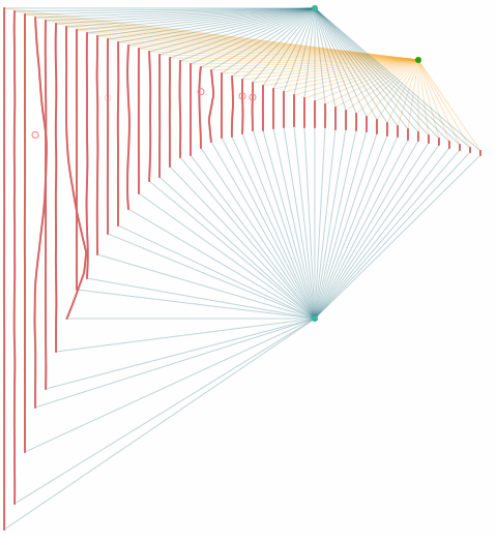

Figure 7. Mass-interaction virtual harp. Strings in red, bridge elements and bridge coupling springs in blue, optional string-barrier collision interactions in orange. Red dots represent finger-string interaction (OSC or MIDI-controlled excitation and damping).

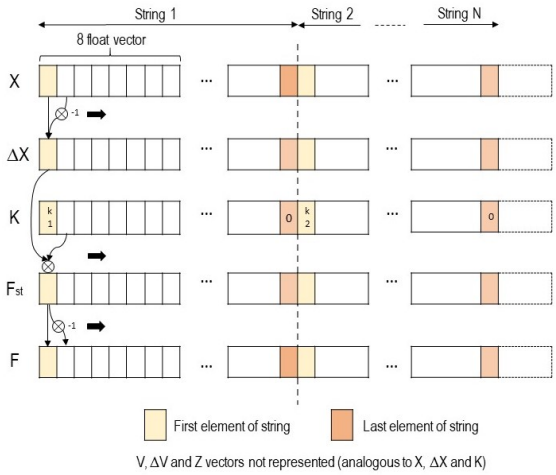

Figure 8. AVX2 vectorised computation scheme for the virtual harp model. Reading top-to-bottom, adjacent position values are shifted into an aligned position difference vector, which is multiplied by the stiffness term to generate spring forces, that are copied and opposite-shifted forwards to account to equal and opposite spring forces between adjacent masses.

the previous subsection, it lies dangerously close to 100% CPU usage and consequently causes severe glitches when running the low latency audio settings required for minimal delay between the real harp player's notes and triggered virtual harp notes.

A final possibility to reduce computational cost without sacrificing the model's sound qualities is explicit AVX2 vectorisation of the physical algorithms: while the compiler struggles to automatically vectorise code at compile-time, such vectorised operations can be defined manually by the user, using specific Intel AVX2 primitives to compute 4 *doubles* or 8 *floats* in a single operation. This involves letting go of genericity and modularity and writing low-level code tailored for speeding up a specific computation process. This is slightly less daunting for strings than for meshes (or worse, arbitrary topologies!) but it still requires some mental and computational gymnastics.

The entire system of strings (excluding bridge connections) is represented as a single string composed of `__m256` (*8 x float*) aligned memory blocks for position X, velocity V, force F, stiffness K and damping Z, with null stiffness and damping parameters separating one actual string from the next. A challenge for manual vectorisation resides in the fact that 256 bit vectorised operations only work on data that is aligned and iterates over 256 bits: while our data structure is suitably aligned for mass algorithms, interactions along a string compute forces between adjacent masses i and $i+1$, requiring a 32 bit offset that AVX instructions cannot handle. Our proposed solution, shown in Figure 8, works by:

1. creating aligned ΔX and ΔV vectors by substracting backward-shifted copies from X and V vectors (optimised using `_mm256_set_ps` operations and carry-over variables for data that shifts from one vector into the previous one).
2. performing vectorised computation of the string interaction forces on this aligned data F_{st}.
3. adding the string force vector F_{st} and a negative forward-shift of F_{st} into the mass force buffer vector F (symmetrically to 1.) to account for equal and opposite forces applied between adjacent masses.
4. performing vectorised computation of the new mass position vector X from the aligned X, V and F vectors (using combined multiplication and add/substract instructions whenever possible).

Since `__m256` arrays are float arrays in memory, keeping *float** pointers to the addresses of the state arrays allows to access string data for the computation of non-vectorised elements, such as the springs connecting both ends of each string to the bridges, damper mechanisms and so forth.

The computational gain of manual single-precision float AVX2 vectorisation is, to say the least, considerable: the vectorised virtual harp runs on a lean 34% in Max/MSP, allowing for steady ultra-low buffer-sizes running alongside other sound processes, and possibly giving leverage for enriching the initial model (e.g. increasing string length and stiffness for additional brightness).

These three examples highlight a number of design and implementation choices that must be confronted on a regular basis when conceiving physically-based virtual musical instruments, in terms of model and real-time control complexity, and software implementation techniques that ensue. In the next section, we discuss some of these aspects.

5. BALANCING MODEL, INTERACTION AND SOFTWARE CONSIDERATIONS

5.1 Balancing model and control complexity

Traditionally, complexity and richness of acoustical vibrating structures has been the dominant concern in physically-based sound-synthesis research, and rightly so, as it has lead to major breakthroughs as testified by many recent works concerning the simulation of strongly non-linear instrumental dynamics [9, 15, 16]. However, we argue that richness of control, and in particular real-time user interaction, is of equal importance in unleashing the full potential of physical models for sound production, as shown in works such as [8, 19]. Even simple models such as the bowed string system of Section 4.1 may yield surprisingly expressive sound through fine multi-DoF control of excitation and modification gestures, allowing users to progressively discover and assimilate the sonic possibilities of the instrument. As another example, the mesh of Section 4.2 sounds rather bland when simply struck, but reveals a completely different nature when driven by continuous control of the moving bow interactions, to the point where it is hard to believe that it is such a rudimentary linear acoustical structure.

5.2 Balancing software genericity and performance

The three instruments discussed in Section 4 also show that model complexity leads to important implementation decisions: while a modular and generic model coding environment such as mi-gen~ allows to focus on model features rather than implementation specifics, the performance bottleneck limits real-time use for large models. Specific C++ coding with static memory allocation for the model state and various algorithmic optimisations allows for a leap in performance, but can still be costly. And finally, low-level manual code vectorisation can allow for a quantitative boost, enabling very large models, but supposes completely re-thinking the algorithms for a specific model topology.

6. CONCLUSIONS AND PERSPECTIVES

Through the complementary questions of physical modelling techniques, computational cost and means for expressive control, as well as the analysis of three recently developed virtual musical instruments, this work offers some insight into various design and implementation considerations that arise when developing real-time physically-based virtual musical instruments.

The proxy elements discussed in Section 3.2 have been integrated into the mi-gen~ Max/MSP package [7] and can readily be used to create and creatively explore modular mass-interaction models with extensive real-time over physical parameters, inputs, outputs and topological connection points. Although side-effects of the first-order interpolators used in the example models were not perceived as particularly problematic or significantly audible during testing and playing phases, higher order cubic Lagrange interpolators will be implemented, as the additional computational cost is very moderate.

[7] https://github.com/mi-creative/mi-gen/

Given the difference in performance between automatically generated C++ (whether using Max/MSP's gen~ engine, as shown in the present results, or using the Faust environment as shown in previous work [26]) and hand-tailored C++ simulation code, ongoing development efforts aim to provide a high-level, modular yet efficient simulation engine based on the prototypes developed for the bowed plate and harp virtual instruments.

On a broader scope, the issue of efficient simulation for large scale physical models for sound synthesis is still a very active topic [9, 11, 14, 15]. This leads to yet another interrogation: is such modelling and computational complexity compatible with modular creative physical modelling tools, tailored for artists, musicians, composers? While the science of physical modelling sound synthesis propels forward at a sustained pace, one must remain vigilant not to place such advances beyond reach from its primary target audience. In this regard, proposing the right tools and environments allowing for mere mortals to harness the full power of physically-based models for sound creation remains a fundamental and exciting challenge.

Further comparison between finite difference schemes and mass-interaction models [27] would be worth exploring, in particular incorporating more advanced/implicit schemes into benchmarks. Models such as the bowed mesh could offer an interesting means to evaluate the audible effects of numerical dispersion in such schemes as well as integration methods for the non-linear bow interactions, both formally and through user experiments. Finally, a more extensive evaluation of the computational cost of 3-DoF versus 1-DoF mass-interaction models in efficient C++ implementation would provide a larger picture of how viable 3D models could be for complex physically-based sound-synthesis in the near future. Early results show exciting perspectives in leveraging audio-rate 3D simulation to capture inherently non-linear acoustical phenomena [7], however the computational requirements lead to believe that 1-DoF physics simulations for large scale virtual musical instruments still have good days ahead.

Acknowledgments

This work has been carried out with support and funding from the French Ministry of Culture, CNRS, Grenoble INP and Universite Grenoble Alpes.

7. REFERENCES

[1] C. Chafe, "Case studies of physical models in music composition," in *Proceedings of the 18th International Congress on Acoustics*, 2004.

[2] C. Cadoz, "The physical model as metaphor for musical creation:" pico.. tera", a piece entirely generated by physical model," in *ICMC 2002-International Computer Music Conference*. MPublishing, 2002, pp. 305–312.

[3] V. Välimäki and T. Takala, "Virtual musical instruments—natural sound using physical models," *Organised Sound*, vol. 1, no. 2, pp. 75–86, 1996.

[4] R. Rabenstein, S. Petrausch, A. Sarti, G. De Sanctis, C. Erkut, and M. Karjalainen, "Blocked-based physical modeling for digital sound synthesis," *IEEE Signal Processing Magazine*, vol. 24, no. 2, pp. 42–54, 2007.

[5] J. O. Smith, "Physical modeling using digital waveguides," *Computer music journal*, vol. 16, no. 4, pp. 74–91, 1992.

[6] S. D. Bilbao, *Numerical sound synthesis*. Wiley Online Library, 2009.

[7] J. Villeneuve and J. Leonard, "Mass-interaction physical models for sound and multi-sensory creation: Starting anew," in *International Conference on Sound and Music Computing*, 2019.

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

[9] S. Bilbao, M. Ducceschi, and C. Webb, "Large-scale real-time modular physical modeling sound synthesis," in *Proc. International Conference On Digital Audio Effects (DAFx 2019), Birmingham, UK*, 2019.

[10] J. Leonard, N. Castagné, C. Cadoz, and A. Luciani, *The MSCI Platform: A Framework for the Design and Simulation of Multisensory Virtual Musical Instruments*. Cham: Springer International Publishing, 2018, pp. 151–169.

[11] J. Perry, S. Bilbao, and A. Torin, "Hierarchical parallelism in a physical modelling synthesis code," *Parallel Computing: On the Road to Exascale*, vol. 27, p. 207, 2016.

[12] V. Zappi, A. Allen, and S. Fels, "Shader-based physical modelling for the design of massive digital musical instruments." in *New Instruments for Musical Expression*, 2017, pp. 145–150.

[13] 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 journal*, vol. 17, no. 1, pp. 19–29, 1993.

[14] C. J. Webb and S. Bilbao, "On the limits of real-time physical modelling synthesis with a modular environment," in *Proceedings of the International Conference on Digital Audio Effects*, 2015, p. 65.

[15] M. Ducceschi and S. Bilbao, "Non-iterative solvers for nonlinear problems: The case of collisions," in *22nd International Conference On Digital Audio Effects (DAFx 2019), Birmingham, UK*, 2019.

[16] C. Issanchou, S. Bilbao, J.-L. Le Carrou, C. Touzé, and O. Doaré, "A modal-based approach to the nonlinear vibration of strings against a unilateral obstacle: Simulations and experiments in the pointwise case," *Journal of Sound and Vibration*, vol. 393, pp. 229–251, 2017.

[17] S. Gelineck and S. Serafin, "A practical approach towards an exploratory framework for physical modeling," *Computer Music Journal*, vol. 34, no. 2, pp. 51–65, 2010.

[18] E. Berdahl, A. Pfalz, M. Blandino, and S. D. Beck, "Force-feedback instruments for the laptop orchestra of louisiana," in *Musical Haptics*. Springer, Cham, 2018, pp. 171–191.

[19] S. Serafin, M. Burtner, C. Nichols, and S. O'Modhrain, "Expressive controllers for bowed string physical models," in *Proceedings of the Digital Audio Effects Conference (DAFx 2001). Limerick, Ireland*, 2001.

[20] N. Castagné and C. Cadoz, "GENESIS : a Friendly Musician-Oriented Environment for Mass-Interaction Physical Modeling," in *ICMC 2002 - International Computer Music Conference*, Gothenburg, Sweden, Sep. 2002, pp. 330–337.

[21] E. Berdahl and J. Smith III, "An introduction to the synth-a-modeler compiler: Modular and open-source sound synthesis using physical models," in *Proceedings of the Linux Audio Conference*, 2012.

[22] A. Kontogeorgakopoulos and C. Cadoz, "Designing and synthesizing delay-based digital audio effects using the cordis anima physical modeling formalism," in *International Conference on Sound and Music Computing*, 2008.

[23] F. Fontana, L. Savioja, and V. Välimäki, "A modified rectangular waveguide mesh structure with interpolated input and output points." in *ICMC*, 2001.

[24] L. Savioja and V. Valimaki, "Improved discrete-time modeling of multi-dimensional wave propagation using the interpolated digital waveguide mesh," in *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 1. IEEE, 1997, pp. 459–462.

[25] J. Leonard and J. Villeneuve, "mi-gen˜: An efficient and accessible mass-interaction sound synthesis toolbox," in *International Conference on Sound and Music Computing*, 2019.

[26] J. Leonard, J. Villeneuve, R. Michon, Y. Orlarey, and S. Letz, "Formalizing Mass-Interaction Physical Modeling in Faust," in *17th Linux Audio Conference (LAC-19)*, Stanford, United States, Mar. 2019.

[27] P. J. Christensen and S. Serafin, "Graph based physical models for sound synthesis," in *International Conference on Sound and Music Computing*, 2019.