International Journal of Information Technology and Knowledge Management
July-December 2010, Volume 2, No. 2, pp. 585-589

A Physical Modeling Approach for PC Based Softsynth- A Low Cost Solution

Jagruti Sanghavi[1], N.G.Bawane[2] & Akansha Pilley[3]

A softsynth is a computer program for digital audio generation. Softsynths can be cheaper and more portable than dedicated hardware. In this paper we proposed a scheme in which a dedicated hardware musical keyboard is developed which sends only MIDI notes. A keyboard is interfaced to PC through USB.A synthesizing of musical notes is done by PC to produced sound. Synthesis is done by comparing different physical modeling technique. This is the low cost solution towards dedicated hardware synthesizers. The paper aims towards using existing use of processing power of PC with existing memory system to process MIDI notes to play different instruments through general purpose PC. Also it makes easy for keyboard player to play music with dedicated hardware keyboard. Proposed system aimed at developing a synthesizer system using computer devices where all the processing done by computer system. Main idea behind project is to avoid using readymade synthesizer IC and utilization of computer processing to get maximum computer power. The physical model is usually formulated as a partial differential equation resulting from a mechanical analysis. The resulting synthesis algorithms consist of a parallel arrangement of second order digital filters [2]. Their coefficients are obtained by analytic expressions directly from the parameters of the physical model. More elaborate computational models include nonlinearities and excitation mechanisms.
Keywords: MIDI, Softsynth, Physical Modeling Synthesis, Digital Waveguide Synthesis.

1. Introduction

A music synthesizer makes sounds by using an electrical circuit as an oscillator to create and vary the frequency of sounds in order to produce different pitches. As long as the pitch is within the range of frequency that can be heard by a human ear, it's known as a "musical pitch". You can use a keyboard to vary these pitches at discrete intervals that correspond to the notes on the musical scale. If you put several oscillators together, you can combine several pitches to create a chord. The tone of particular pitch is done by playing a given pitch with waveforms of different shapes. Since the harmonic structure of these waveforms differs, our ears interpret them as different tones. The sound you will hear can also be modified by voltage-controlled oscillators and voltage controlled filters .Synthesizers are able to only mimic the sounds of non-synthetic instruments, but also to create sounds that absolutely cannot be played by anything but a music synthesizer. That is because a music synthesizer is well-suited to delicate manipulations of its oscillators.

Synthesizing using physical modeling in software. Many sound synthesis methods like sampling, frequency modulation (FM) synthesis, and additive and subtractive synthesis model sound. This is good for creating new sounds, but has several disadvantages in reproducing sounds of real acoustic instruments. The most important disadvantage is that the musician does not have the physical based variability he has with real musical instruments. Therefore it is difficult to phrase a melody with these methods. Because of these disadvantages there are various methods for sound synthesis based on physical models that do not model the sound but the sound production mechanism. They all start from physical models in form of partial differential equations (PDEs).

2. Present Work

Below are various physical-model representations:

- Ordinary Differential Equations (ODE);
- Partial Differential Equations (PDE);
- Difference Equations (DE);
- Finite Difference Schemes (FDS);
- Transfer Functions (between physical signals).

ODEs and PDEs are purely mathematical descriptions (being differential equations), but they can be readily digitized to obtain computational physical models. Difference equations are simply digitized differential equations. That is, digitizing ODEs and PDEs produces DEs. A DE may also be called a finite difference scheme. A

[1,2,3]Department of Computer Science and Engineering, G.H. Raisoni College of Engineering, Nagpur University, INDIA

Email: jagrutijai@rediffmail.com, narenbawane@rediffmail.com, Akansha.Pilley@gmail.com

discrete-time state-space model is a special formulation of a DE in which a vector of state variables is defined and propagated in a systematic way (as a vector first-order finite-difference scheme). A linear difference equation with constant coefficients—the Linear, Time-Invariant (LTI) case—can be reduced to a collection of transfer functions, one for each pairing of input and output signals (or a single transfer function matrix can relate a vector of input signal z transforms to a vector of output signal z transforms).

An LTI state-space model can be diagonalized to produce a so-called modal representation, yielding a computational model consisting of a parallel bank of second-order digital filters. Digital waveguide networks can be viewed as highly efficient computational forms for propagating solutions to PDEs allowing wave propagation. They can also be used to compress the computation associated with a sum of quasi harmonically tuned second-order resonators.

2.1. Ordinary Differential Equations (ODEs)

Ordinary Differential Equations (ODEs) typically result directly from Newton's laws of motion, restated here as follows:

$$f(t) = m\ddot{x}(t) \tag{1}$$

The initial position x(0) and velocity v(0) of the mass comprise the initial state of mass, and serve as the boundary conditions for the ODE. The boundary conditions must be known in order to determine the two constants of integration needed when computing x(t) for t > 0.

If the applied force f (t) is due to a spring with spring-constant k, then we may write the

$$x(t) + m\ddot{x}(t) = 0 \tag{2}$$

2.2. PDEs (A Partial Differential Equation)

A partial differential equation (PDE) extends ODEs by adding one or more independent variables (usually spatial variables). For example, the wave equation for the ideal vibrating string adds one spatial dimension x (along the axis of the string) and may be written as follows:

$$Ky''(x,t) = \epsilon \ddot{y}(t) \tag{3}$$

(Restoring Force = Inertial Force)

Where y(x, t) denotes the transverse displacement of the string at position x along the string and time t and

$$y'(x,t) \triangleq \frac{\partial y(x,t)}{\partial x} \tag{4}$$

denotes the partial derivative of y with respect to x.

The physical parameters in this case are string tension k and string mass-density ϵ.

2.3. Difference Equations (Finite Difference Schemes)

There are many methods for converting ODEs and PDEs to difference equations. One method is to replace each derivative with a finite difference:

$$\dot{x}(t) \triangleq \frac{d}{dt}(x,t) \triangleq \lim_{\delta \to 0} x(t) - \frac{x(t-\delta)}{\delta} \approx x(nt) - x(n-1)T\,/\,T \tag{5}$$

Consider a mass m driven along a frictionless surface by a driving force f (t), as in Fig.1, and suppose we wish to know the resulting velocity of the mass v (t), assuming it starts out with position and velocity 0 at time 0. Then, from Newton's

$$f = ma \tag{6}$$

relation, the ODE is $f(t) = mv(t)$ (7)

and the difference equation resulting from the backward-difference substitution is

$$f(n, t) = m\frac{v(nt) - v\lfloor (n-1)\rfloor T]}{T}, n = 0,1,2 \tag{8}$$

Solving for v (nT) yields the following finite difference scheme:

$$v(nt) = v[(n-1)T] + \frac{T}{mf(nt)}, n = 0,1,2 \tag{9}$$

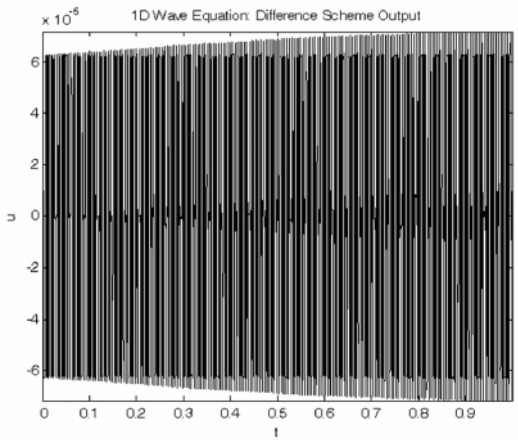
Finite difference scheme in explicit form can be implemented in real time as a causal digital filter. There are also implicit finite-difference schemes which may correspond to non-causal digital filters .When we consider finite difference scheme, it can be implemented in matlab as follows

1D wave equation the matlab code is as follows:

```
% matlab script waveeq1dfd.m
% finite difference scheme for the 1D wave equation
% *fixed boundary conditions
% *raised cosine initial conditions
%%%%%% begin global parameters
SR = 44100; % sample rate (Hz)
f0 = 200; % fundamental frequency (Hz)
TF = 1; % duration of simulation (s)
ctr = 0.4; % center location of excitation (0-1)
wid = 0.1; % width of excitation
u0 = 0; % maximum initial displacement
v0 = 1; % maximum initial velocity
lambda = 1; % Courant number
rp = 0.5; % position of readout (0-1)
%%%%%% end global parameters
%%%%%% begin derived parameters
gamma = 2*f0; % wave equation free parameter
```

```
    k = 1/SR; % time step
    NF = floor(SR*TF); % duration of simulation
(samples)
    h = gamma*k/lambda; % grid spacing
    N = floor(1/h); % number of subdivisions
of spatial domain
    h = 1/N; % reset h
    lambda = gamma*k/h; % reset Courant number
    s0 = 2*(1-lambda^2); s1 = lambda^2; %
scheme parameters
    rp_int = 1+floor(N*rp); % rounded grid
index for readout
    rp_frac = 1+rp/h-rp_int; % fractional part
of readout location
    %%%%%% initialize grid functions and output
    u = zeros(N+1,1); u1 = zeros(N+1,1); u2 =
zeros(N+1,1);
    out = zeros(NF,1);
    %%%%%% create raised cosine
    rc = zeros(N+1,1);
    for qq=1:N+1
     pos = (qq-1)*h; dist = ctr-pos;
     if(abs(dist)<=wid/2)
     rc(qq) = 0.5*(1+cos(2*pi*dist/wid));
     end
    end
    %%%%%% set initial conditions
    u2 = u0*rc; u1 = (u0+k*v0)*rc;
    %%%%%% start main loop
    for n=3:NF
    u(2:N) = -u2(2:N)+s0*u1(2:N)+ s1*(u1(1:N-
1)+u1(3:N+1)); % scheme calculation
    out(n) = (1-rp_frac)*u(rp_int)+
rp_frac*u(rp_int+1); % readout
    u2 = u1; u1 = u; % update of grid variables
    end
    %%%%% end main loop
    % plot output waveform
    plot([0:NF-1]*k, out, 'k');
    xlabel('t'); ylabel('u'); title('1D Wave
Equation: Difference Scheme Output');
    axis tight
    % play sound
     soundsc(out,SR);
```

The output for the 1D wave equation is as follows:



2.4. Transfer Functions

A discrete-time transfer function is the z transform of the impulse response of a linear, time-invariant (LTI) system. In a physical modeling context, we must specify the input and output signals we mean for each transfer function to be associated with the LTI model. For example, if the system is a simple mass sliding on a surface, the input signal could be an external applied force, and the output could be the velocity of the mass in the direction of the applied force. In systems containing many masses and other elements, there are many possible different input and output signals. It is worth emphasizing that a system can be reduced to a set of transfer functions only in the LTI case, or when the physical system is at least nearly linear and only slowly time-varying (compared with its impulse-response duration).

3. CONVERTING PHYSICAL MODELS TO WAVE DIGITAL FILTERS

The idea of wave digital filters is to digitize RLC circuits (and certain more general systems) as follows:

1. Determine the ODEs describing the system (PDEs also workable).
2. Express all physical quantities (such as force and velocity) in terms of traveling-wave components. The traveling wave components are called wave variables. For example, the force f(n) on a mass is decomposed as $f(n) = f^{+}(n) + f^{-}(n)$, where $f^{+}(n)$ is regarded as a traveling wave propagating toward the mass, while $f^{-}(n)$ is seen as the traveling component propagating away from the mass. A traveling wave view of force mediation is actually much closer to underlying physical reality than any instantaneous model.
3. Next, digitize the resulting traveling-wave system using the bilinear transform. The bilinear transform is equivalent in the time domain to the trapezoidal rule for numerical integration.
4. Connect N elementary units together by means of N -port scattering junctions. There are two basic types of scattering junction, one for parallel, and one for series connection. The theory of scattering junctions is introduced in the digital waveguide context.

4. REASON FOR SELECTION OF PROJECT

Proposed system is a combination of computer system and MIDI keyboard to generate synthesizer instrumental frequency by synthesizing sound using physical modeling using digital loop filter method. Whenever we install device driver for sound card in computer system it will install logical driver input device, output device and synthesizer

device that is MIDI device. Computer sound card is capable of playing different instrumental frequency using synthesizer device. By developing software application we can utilize these synthesizer capabilities of sound card. As we are developing software application we can make it dynamic at any extend by proving power of input and output device available in the system. Only problem with such a software application if lack of flexibility in playing piano like keys. It is not possible to play music using computer keyboard and hence many artist avoid such a software system. So to overcome this problem proposed system is developed in such a manner that it will provide power of computer processing and the flexibility of piano keyboard. The project is further divided in different module like processing unit, piano keyboard interfacing with computer system and output devices. So the main reason for developing this project is to provide maximum PC power to synthesizer. Many synthesizer ICs available in the market and will these are build for specific task to generate different instrumental frequency on speaker system. In many recording studio artist use PC system for further processing.

5. RESEARCH METHODOLOGY TO BE EMPLOYED

5.1. API Technology

Windows APIs are dynamic link libraries (DLLs) that are part of the Windows operating system. You use them to perform tasks when it is difficult to write equivalent procedures of your own. . Windows dynamic-link libraries (DLLs) represent a special category of interoperability. Windows APIs do not use managed code, do not have built-in type libraries, and use data types that are different than those used with Visual Studio .NET. Because of these differences, and because Windows APIs are not COM objects, interoperability with Windows APIs and the .NET Platform is performed using platform invoke, or PInvoke. Platform invoke is a service that enables managed code to call unmanaged functions implemented in DLLs. For more information, see Consuming Unmanaged DLL Functions. You can use PInvoke in Visual Basic .NET by using either the Declare statement or applying the DllImport attribute to an empty procedure.

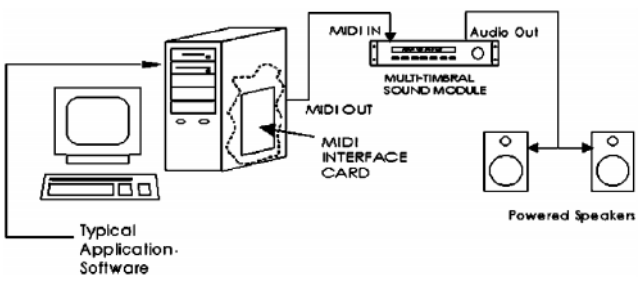


Fig. 1: A PC-Based MIDI System

6. CONCLUSION

Proposed systems aims towards implementing low cost synthesizers using existing PC power .The inputs are provided by only buttons and processing is done at PC. Many sound synthesis methods like sampling, frequency modulation (FM) synthesis, and additive and subtractive synthesis model sound. This is good for creating new sounds, but has several disadvantages in reproducing sounds of real acoustic instruments. The most important disadvantage is that the musician does not have the physical based variability he has with real musical instruments. Therefore it is difficult to phrase a melody with these methods. Because of these disadvantages there are various methods for sound synthesis based on physical models that do not model the sound but the sound production mechanism. They all start from physical models in form mathematical equations such as of partial differential equations (PDEs). They can be obtained by applying the first principles of physics. These mathematical equations are then realised into equivalent circuits by digital wave guide method to get real melody of instrument.

REFERENCES

[1] Farshad Arvin, and Shyamala Doraisamy, "A Real-Time Signal Processing Technique for MIDI Generation", WASET Journal, 50, pp:593-597, 2009.

[2] Braun, J.P.; Gosbell, V.J.; Perera, S "Power Quality Waveform Generator based on the CSound Software Sound Synthesizer", 11th IEEE International Conference on Digital Object Identifier, 12, Issue 15, Sept, pp.391 – 396, 2004.

[3] Victor Lazzarini "Distortion Synthesis" A Tutorial with Csound Examples Issue 11, July 15. (Summer 2009).

[4] Aaron Hechmer, Adam Tindale, George Tzanetakis, "LogoRhythms: Introductory Audio Programming for Computer Musicians in a Functional Language Paradigm", 36th ASEE/IEEE Frontiers in Education Conference, pp-1-6, (October 28 – 31, 2006).

[5] Lazzarini, V., J. Timoney and T. Lysaght, "Split-Sideband Synthesis". Proceedings of the ICMC 2008, Belfast, UK, 2008.

[6] Lazzarini, V., J. Timoney and T. Lysaght, "Adaptive FM Synthesis". Proceedings of the 10th Intl. Conference on Digital Audio Effects (DAFx07). Bordeaux: University of Bordeaux: 21-26, 2007.

[7] Lazzarini, V., J. Timoney and T. Lysaght,"The Generation of Natural-Synthetic Spectra by Means of Adaptive Frequency Modulation". Computer Music Journal, 32 (2): 12-22,2008.

[8] Lazzarini, V., J. Timoney and T. Lysaght, "Asymmetric Methods for Adaptive FM Synthesis". Proceedings of the International Conference on Digital Audio Effects, Helsinki, Finland, 2008.

[9] Dave Phillips " Computer Music and the Linux Operating System a Report from the Front" Computer Music Journal, 27:4, pp. 27–42, (Winter 2003).

[10] Mikael Laurson Mika Kuuskankare Vesa Norilo, (Spring 2009) "An Overview of PWGL, a Visual Programming Environment for Music" ProjectMuse-Computer Music . . . . . , 33, Number 1, pp. 19-31.

[11] Lazzarini, V., J. Timoney, 2008, "New Perspectives on Distortion Synthesis for Virtual Analogue Oscillators". Submitted to Computer Music Journal.

[12] Alexander Müller and Rudolf Rabenstein, (September 1-4, 2009) "Physical Modeling for Spatial Sound Synthesis" 12th Int. Conference on Digital Audio Effects Como, Italy, pp:1-8.

[13] Karjalainen, M. (July 2008) " Efficient Realization of Wave Digital Components for Physical Modeling and Sound Synthesis" IEEE Transactions on Audio, Speech, and Language Processing, 16, Issue: 5 on page(s): 947-956.

[14] Bilbao, S. (March 2007) "Robust Physical Modeling Sound Synthesis for Nonlinear Systems" Signal Processing Magazine, IEEE, 24, Issue : 2, On page(s): 32 - 41.

[15] Vesa Välimäki Henri Penttinen, Jonte Knif Mikael Laurson Cumhur Erkut, (January 2004) "Sound Synthesis of the Harpsichord using a Computationally Efficient Physical Model" EURASIP Journal on Applied Signal Processing, 2004, Pages: 934 – 948.

[16] Trautmann, L. Petrausch, S. Rabenstein, R, "Physical Modeling of Drums by Transfer Function Methods" Acoustics, Speech, and Signal Processing, 2001, 5, pp: 3385 – 3388.