Fast additive sound synthesis for real-time simulation of ocean surface

Matthias Robine, Jocelyn Fréchot

Fast Additive Sound Synthesis for Real-time Simulation of Ocean Surface

M. Robine and J. Fréchot
LaBRI, University of Bordeaux, France
{robine|frechot}@labri.fr

Keywords: additive sound synthesis, real-time image synthesis, natural phenomena simulation

Abstract - The fastest additive sound synthesis methods are presented and used to compute and animate ocean surface. These sound methods are commonly used to efficiently sum a lot of sinusoidal components of the sound called partials. We consider the ocean waves as partials of a sound by shifting from the frequency and time domains to the wavenumber and space ones. Sound methods can therefore be used to compute coordinates of ocean surface points, and allow the rendering of realistic ocean surfaces in real-time.

1. INTRODUCTION

Simulation of ocean surfaces requires a lot of visual details to be realistic. In particular, the presence of hundreds of wave trains leads to complex and nice looking surfaces. While rendering with texture effects is very efficient, it still lacks geometry details due to wave animations. This problem has been handled by the use of the Fast Fourier Transform (FFT), but this method is not flexible and can hardly produce realistic ocean scenes. On the other hand, classical Gerstner equations can easily be used to simulate wide and complex surfaces. By using oceanographic methods to get wave parameters, as amplitude and frequency distribution, these parametric equations allow the reproduction of ocean surface with respect to a given sea state. But since they rely on classical trigonometric functions, they cannot handle simultaneously fast and high-quality rendering.

We have recently presented a new additive sound synthesis method called PASS [1]. Particularly efficient for low frequency signals, this method seems to be suited to synthesize ocean waves. In this paper, we propose to adapt fast sound synthesis methods to the simulation of ocean waves. Since these methods are devoted to the fast evaluation of sums of sine functions, they are perfectly suitable for Gerstner equations computations. While not reaching FFT performances, they allow us to use hundreds of waves. Using an adaptive mesh method to handle the surface, we achieve real-time realistic ocean simulation.

This paper is organized as follows: the principles of classic additive synthesis are reviewed in section 2, as well as the methods proposed for real-time implementations. Section 3 presents the existing methods for the simulation of ocean surface in computer graphics, and explains how we apply sound methods to wave simulation. We give some results in section 4, and perspectives in section 5.

2. ADDITIVE SOUND SYNTHESIS METHODS

We present here the principles of additive sound synthesis, and its fastest implementations.

2.1. Additive Sound Synthesis

Additive sound synthesis is the original spectrum modeling technique (eg. [2]). It is rooted in Fourier's theorem, which states that any periodic function can be modeled as a sum of sinusoids at various amplitudes and harmonic frequencies. For stationary pseudo-periodic sounds, these amplitudes and frequencies evolve slowly with time, controlling a set of pseudo-sinusoidal oscillators commonly called *partials*. This is the well-known McAulay-Quatieri representation [3] for speech signals, also used by Serra [4] in the context of musical signals. As they evolve slowly in time, we can consider the frequencies and amplitudes as constant for a short length. An audio signal s can be calculated from the sum of the partials using:

$$s(t) = \sum_{i=1}^{N} a_i \sin\left(2\pi f_i t + \phi_i\right) \qquad (1)$$

where N is the number of partials in the sound and the parameters of the model are f_i, a_i, and ϕ_i, which are respectively the frequency, amplitude, and initial phase of the partial number i. This equation is valid if the frequency is constant. However, for practical sound examples, both the frequency and the amplitude must be updated regularly. Equation (1) then holds for each sound segment between two update times.

In the general approach derived from (1), the partials are processed separately for each sample, and then summed. The complexity of the method is therefore proportional to the product of the number of partials and the sample rate. Fast additive synthesis methods propose to reduce the computation time of the synthesis, while keeping the control of all the parameters of the sound partials in time.

2.2. Inverse Fourier Transform

In order to efficiently synthesize many sinusoids simultaneously, Freed, Rodet, and Depalle propose in [5] to use the inverse Fourier transform. The idea is to reconstruct the short-term spectrum of the sound at time t, by adding the band-limited contribution of each partial, then to apply the Inverse Fast Fourier Transform (IFFT or FFT^{-1}) in order to obtain the temporal representation of the sound, and finally to repeat the same computation further in time. Complexity

decreases when the number of oscillators is large in comparison to the number of samples to compute at each frame. This approach is very interesting, because its complexity is no more the product of the number of partials and the sampling rate. However, the control of the additive parameters is more complex.

2.3. Digital Resonator

The most straightforward way to calculate a partial contribution is to use the sine function, but it consumes a lot of computation time. We can prefer the use of the digital resonator method (see for example [6, 7]), which computes the samples of each separate partial with an optimal number of operations. In this method, the sine is calculated with an incremental algorithm that avoids computing the sine function for every sample. Marchand and Strandh [8, 9] proposed the use of the digital resonator with floating point arithmetic for fast additive synthesis. For each partial the resonator is initialized as (2) shows, with F_s the sampling rate of the synthesis, a, f, and ϕ respectively the amplitude, frequency, and initial phase of the partial, and Δ_ϕ the phase increment. The incremental computation of each oscillator sample requires only 1 multiplication and 1 addition.

$$\begin{cases} \Delta_\phi & = & \frac{2\pi f}{F_s} \\ s[0] & = & a\sin(\phi_0) \\ s[1] & = & a\sin(\phi_0 + \Delta_\phi) \\ C & = & 2\cos(\Delta_\phi) \\ s[n+1] & = & C \cdot s[n] - s[n-1] \end{cases} \quad (2)$$

The complexity C_{DR} of the digital resonator method is naturally given by:

$$C_{\text{DR}} = O(NF_s\Delta_t) \quad (3)$$

with F_s the sampling rate, N the number of partials in the sound, and Δ_t the duration of the synthesis.

2.4. Polynomial Additive Sound Synthesis (PASS)

We proposed in [1] the PASS method, which uses polynomials to replace the sine function. It consists of first calculating a set of polynomial coefficients for each partial of the sound. Evaluation of polynomials computed with these coefficients approximate the signal of the partials on a part of their period. The classic approach would evaluate the polynomial associated to each oscillator, and then sum up the results, which is quite inefficient. The idea is yet to sum the coefficients in a polynomial generator, then to evaluate the resulting polynomial only once. Indeed, summing polynomials leads to another polynomial of the same degree. The sound samples can be computed from this single resulting polynomial, with a fairly low degree, independent of the number of partials to synthesize. The general process is illustrated by Figure 1.

The polynomial approximation of a partial is valid only on a part of the period of the sine function. Thus, the polynomial coefficients must be regularly updated, with a rate that depends on the frequency of the partial. However, as the function to approximate is periodic, we only need to compute sets of coefficients for one period, as long as the amplitude and the frequency of the partial are constant. A polynomial generator is regularly updated with these changes of coefficients and is evaluated each time a sound sample must be produced.

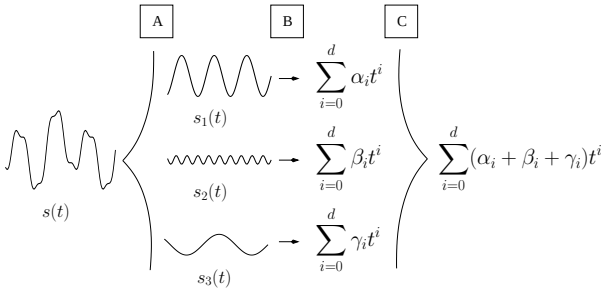

Figure 1: PASS. *Step A:* A periodic signal can be divided into sinusoidal components (Fourier's theorem). *Step B:* Computing polynomial coefficients to approximate the signal for each partial. *Step C:* A polynomial generator is obtained by summing the coefficients from the polynomials of the partials. The degree of the generator is the same as the degree used for partials. Values computed by the generator are the samples of the sound signal.

Thus, two types of event occur during time. An update event consists of updating the coefficients of the generator, due to the change of the polynomial coefficients of one partial. An evaluating event occurs when a value is computed from the polynomial generator to produce a sample of the sound. The generator is therefore regularly updated and computed during time. The complexity of PASS is dominated by the management of the update events from individual partials. [1] uses a binary heap as priority queue to manage these events. Thus, the general complexity of PASS is:

$$C_{\text{PASS}} = O\left(\alpha \sum_{i=1}^{N} f_i \log N\Delta_t + dF_s\Delta_t\right) \quad (4)$$

where f_i is the frequency parameter of the partial number i, d the polynomial degree of the generator and α a constant which is architecture-dependent.

3. APPLICATION TO OCEAN RENDERING

We propose in this section to use the additive sound methods described above to synthesize ocean surface.

3.1. Synthesis of Ocean Surface

Ocean waves can be described by several parameters. The most commonly used for wave simulation are amplitude A, wavelength λ, frequency F, wavenumber $k = 2\pi/\lambda$ and angular frequency $\omega = 2\pi F$ (in deep water, $\omega^2 = gk$, where g is the standard acceleration of gravity). A wave vector $\vec{k}$ is a vector with magnitude k and direction of the propagation of the wave.

The common wave model used for ocean simulation is based on the Gerstner equations and was introduced in computer graphics by Fournier and Reeves [10]. This Lagrangian model describes the trajectory of water particles at the ocean surface, due to a set of wave trains. For each wave, particles follow a vertical circle around their position at rest, leading

to a trochoid wave shape:

$$\begin{cases} \vec{x}(\vec{x}_0,t) = \vec{x}_0 + \sum_i \hat{\vec{k}}_i\, A_i\, \sin(\vec{k}_i \cdot \vec{x}_0 - \omega_i t + \Phi_i) \\ z(\vec{x}_0,t) = z_0 - \sum_i \quad A_i \cos(\vec{k}_i \cdot \vec{x}_0 - \omega_i t + \Phi_i) \end{cases} \tag{5}$$

where $\vec{x} = (x, y)$ is the horizontal particle position at time t, z is its vertical position, $\vec{x}_0 = (x_0, y_0)$, (x_0, y_0, z_0) is the position of the particle at rest, $\hat{\vec{k}}$ is the unit vector of $\vec{k}$ and Φ is a random phase term. Using a regular grid of points, this model allows procedural animations of ocean waves. The reader can refer to [11] for a short overview of the related work.

3.2. Sound Methods for Ocean Surface Synthesis

We choose to not use the inverse Fourier transform, which leads to a loss of control of the additive parameters. Digital resonator and PASS methods allow to sum a lot of sinusoidal components to synthesize a signal as a time function. As we want to synthesize a grid of points of an ocean surface, we propose to apply sound methods line by line on the grid. The synthesis is done as function of length on a line of the grid, instead as function of time. To compute the x coordinate of a point of the line, we use the wave parameters $\vec{k}$, A, w and Φ, respectively the wave vector, the amplitude, the angular frequency and the phase of a wave. Sound parameters a, f and ϕ, respectively the amplitude, the frequency and the initial phase of a partial are given by:

$$\begin{cases} a &= A\hat{k_x} \\ f &= \frac{k}{2\pi} \cos(\vec{k}, \vec{L}) \\ \phi &= \vec{k} \cdot \vec{x_0} - \omega t + \Phi \end{cases} \tag{6}$$

with $\hat{k_x}$ defined by $\hat{\vec{k}} = (\hat{k_x}, \hat{k_y})$, and $\vec{L}$ the direction vector of the line.

We can now perform a sound synthesis along a line of the grid, using distance Δ_L instead of time. Δ_L is the distance from the first point of the line, along this line. The value of the x coordinate is therefore given by:

$$x = x_0 + \sum_{i=1}^{N} a_i \sin\left(2\pi f_i \Delta_L + \phi_i\right) \tag{7}$$

We do the same for all the coordinates, and again for the derivatives used to compute the normals. Only the initial phase parameter varies between two lines.

We can choose to compute the sum of sinusoidal components directly with the sine function, but it is very time consuming. The digital resonator can be used instead. It must be initialized by computing two sine functions by partial and by line. Then only 1 addition and 1 multiplication by partial are necessary to process the next point evaluated on the line. Using PASS method, we must choice the part of period on which we approximate the sine function, and the degree of polynomials. It has a big influence on the performance of the approximation and on the computation time. We have chosen to approximate sine functions on half-periods with 2-degree polynomials. It leads to a maximal error of 4 percent of the signal. As the wave parameters do not change during a synthesis of an ocean surface line, we have modified the PASS algorithm to use a merge sort to manage the update events (see section 2.4), instead of a binary heap as priority queue. In our case, it leads to better performances.

3.3. Complexity Comparison

The complexities of digital resonator and PASS methods are respectively given by (3) and (4). If we apply the change of parameters from wave to partial as in (6), using L as the length of the synthesized ocean line and X as the number of points on this line, we obtain:

$$C_{\text{PASS}} = c_1 \sum_{i=1}^{N} f_i \log NL + c_2 dX \tag{8}$$

$$C_{\text{DR}} = c_3 NX \tag{9}$$

where c_1, c_2, c_3 are architecture-dependent constants.

We use an adaptive rendering [12] according to the view point to minimize the numbers of points needed in the rendering step. It reduces the time consuming by the synthesis. Moreover, we filter the waves according to their wave length to avoid aliasing. These methods have an impact on the performance of the synthesis. Improvements vary according to the synthesis method used.

Concerning the digital resonator method, which is linearly dependent on the number of waves and linearly dependent on the number of points on the line, the effect on the performance is easy to understand. We saw before that several parameters influence the complexity of the PASS method, principally the number of waves, the sum of the frequencies of their corresponding partials, and the length of the line to synthesize. If the impact of adaptive rendering is as clear as with digital resonator method, we can note that the anti-aliasing filtering affects waves with the highest frequencies. As the sum of frequencies is very sensitive to this, PASS method takes more advantage than the digital resonator from these rendering improvements.

A comparison of the complexities of the methods is hard to achieve, due to the architecture-dependent constants. However, all the tests we practiced with adaptive rendering have shown that the PASS method is the fastest, ever in extreme condition of wind speed, for different numbers of points or numbers of waves.

4. RESULTS AND DISCUSSION

We have implemented both digital resonator and PASS methods using C++. We used a regular grid for computation time comparisons while our main framework is the adaptive mesh model of [12]. Wave characteristics are found by adaptively sampling a JONSWAP spectrum, as in [11]. Thus, when the wind speed or fetch increases, sampling is focused on waves with low wavenumbers. For the PASS, this means the sum of frequencies $\sum_i f_i$ decreases (see section 3.3).

Comparisons of computation times on a 3 GHz Pentium 4 PC are shown in table 1. For the classical sine/cosine functions method, we used the sincos instruction present on modern PC. This allows the evaluation of the sine and cosine of a single value at the same time, reducing the evaluation

Figure 2: Adaptive mesh with a screen resolution of 1024×768 pixels, a 256×192 points mesh, 400 waves and a wind speed of $5\ \mathrm{m{\cdot}s^{-1}}$. The framerate is from 5 to 12 fps with PASS method (less than 1 fps with sincos).

time by about a factor 2. For the PASS method, computations took place along lines which are orthogonal to the direction of the wind. Since the majority of waves runs in a direction close to the direction of the wind, this decreases the sum of frequencies of the partials and advantages the method.

X × Y	N	L *m*	WS $m{\cdot}s^{-1}$	SC *ms*	DR *ms*	PASS *ms*
128 × 128	200	200	5	360	**100**	120
128 × 128	200	200	15	360	100	**30**
128 × 128	400	50	5	730	210	**105**
128 × 128	400	50	15	730	210	**50**
128 × 128	400	200	5	730	**210**	265
128 × 128	400	200	15	730	210	**70**
256 × 256	200	50	5	1610	600	**90**
256 × 256	200	50	15	1610	600	**50**
256 × 256	200	200	5	1610	600	**245**
256 × 256	200	200	15	1610	600	**65**

Table 1: Computation times in *ms* of a regular grid with the sincos (SC), digital resonator (DR) and PASS methods, for different numbers of points (X × Y), numbers of waves (N), grid widths (L) and wind speeds (WS).

Sound synthesis algorithms are always faster than the sincos instruction. The PASS gives generally better results than the digital resonator, but is very sensible to the width of the grid and to the wind speed. However, the improvement is noticeable and allows us to produce animations at an interactive rate, with many more waves and/or grid points than classical method. For comparison, computation of a grid of 128×128 points and 256×256 points with FFT method take respectively about 10 and 60 ms.

We have tested our adaptive mesh implementation with a Radeon 9200 graphic board, using a screen resolution of 1024×768 pixels, a 128×96 points mesh (i.e. mesh quads are 8 pixels wide), 400 waves and a wind speed of $5\ \mathrm{m{\cdot}s^{-1}}$. With the sincos method we got 2 to 5 fps, depending on the view point, and 5 to 15 fps with the digital resonator. We got 10 to 20 fps with the PASS, then 5 to 12 fps when increasing the mesh resolution by a factor 2 (figure 2). We tested also to render a high quality surface, with 800 waves and a mesh size of 1024×768. It took about 1 minute and 10 seconds with sincos instruction and less than 2 seconds with the PASS, increasing performances by a factor 35.

5. CONCLUSION AND FUTURE WORK

We have proposed a method that adapts the fastest additive sound synthesis algorithms to the realistic simulation of ocean waves. Combined with an adaptive mesh, we achieve the rendering of complex ocean surfaces at interactive rate. As improvements of our work, we could approach a trochoid curve with the PASS method instead of three sine functions. More generally, applications that need to sum a lot of periodic signals could take a big advantage by using methods initially suited to additive sound synthesis.

REFERENCES

[1] M. Robine, R. Strandh, and S. Marchand, "Fast Additive Sound Synthesis Using Polynomials," to appear in Proc. DAFx 2006.

[2] J. A. Moorer, "Signal Processing Aspects of Computer Music – A Survey," *Computer Music Journal*, vol. 1, no. 1, pp. 4–37, 1977.

[3] R. J. McAulay and T. F. Quatieri, "Speech analysis / synthesis based on a sinusoidal representation," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol. 34, no. 4, pp. 744–754, 1986.

[4] X. Serra and J. O. Smith, "Spectral modeling synthesis: A sound analysis/synthesis system based on a deterministic plus stochastic decomposition," *Computer Music Journal*, vol. 144, no. 4, pp. 12–24, 1990.

[5] A. Freed, X. Rodet, and P. Depalle, "Synthesis and Control of Hundreds of Sinusoidal Partials on a Desktop Computer without Custom Hardware," in *Proc. ICSPAT*, 1992.

[6] J. W. Gordon and J. O. Smith, "A Sine Generation Algorithm for VLSI Applications," in *Proc. ICMC*, 1985.

[7] J. O. Smith and P. R. Cook, "The Second-Order Digital Waveguide Oscillator," in *Proc. ICMC*, 1992, pp. 150–153.

[8] S. Marchand and R. Strandh, "InSpect and ReSpect: Spectral Modeling, Analysis and Real-Time Synthesis Software Tools for Researchers and Composers," in *Proc. ICMC*, 1999, pp. 341–344.

[9] S. Marchand, *Sound Models for Computer Music (Analysis, Transformation, and Synthesis of Musical Sound)*, Ph.D. thesis, University of Bordeaux 1, 2000.

[10] A. Fournier and W.T. Reeves, "A simple model of ocean waves," *SIGGRAPH Computer Graphics*, vol. 20, no. 4, pp. 75–84, 1986.

[11] J. Fréchot, "Realistic simulation of ocean surface using wave spectra," in *Proc. GRAPP*, 2006, pp. 76–83.

[12] D. Hinsinger, F. Neyret, and M.P. Cani, "Interactive animation of ocean waves," in *Symposium on Computer Animation*, 2002, pp. 161–166.