Green's Function Method for Creating Accurate Stereo Sound Images: Simple Plucked String

David R. Clark, EXE Consulting
P.O. Box 450998 Garland, Texas 75045-0998
16 July 2004

Herein is presented an example which shows the Green's function method described in [1] and [2] applied to a simple, plucked-string model

As shown in [2], a solution for the one-dimensional (1-D) wave equation, for a source at (x'') and fixed reception point at (x):

$$\psi(x,t) = c^2 \frac{2u}{a} {\sum_n}' \left[\sin\left(\frac{n\pi x''}{a}\right) \sin\left(\frac{n\pi x}{a}\right) \int_{t'=-\infty}^{t} s(t') \frac{\sin(\omega_{nml}\tau)}{\omega_{nml}} dt' \right]$$

can be obtained by convolving f(t) with s(t') where:

$$f(t) = {\sum_n}' \frac{A_n}{\omega_n} \sin(\omega_n t)$$

$$A_n = c^2 \frac{2u}{a} \sin\left(\frac{n\pi x''}{a}\right) \sin\left(\frac{n\pi x}{a}\right)$$

which can be computed for all n. If we choose to incorporate a decay into f(t), it becomes:

$$f(t) = e^{-\alpha t} {\sum_n}' \frac{A_n}{\omega_n} \sin(\omega_n t)$$

For the case of a plucked string, we can use a truncated sawtooth function in time:

$$s(t') = (t' - t_1)\,[\theta(t' - t_1) - \theta(t' - t_2)]$$

where the actual amplitude will be comprehended by the constant u, and $\theta(t)$ is the Heaviside step function. The effect of this particular s(t') is to model the pulling on a string, for example the deflection of a guitar string by a pick, until it is suddenly released. The impulse response function is for a particular position, namely x'', so the pulling force is as if applied to a single point of the string. The solution will then be:

$$\psi(x,t) = f(t) * s(t) \equiv \int_{-\infty}^{\infty} f(t')\, s(t - t')\, dt' = \int_{-\infty}^{\infty} f(t - t')\, s(t')\, dt' \qquad \text{(x is fixed)}$$

As discussed in [1], the upper limit of integration is effectively t at any particular time t rather than infinity for the cases we will consider.

If we simply substitute s(t') into the original solution, we obtain:

$$\psi(x,t) = \int_{-\infty}^{\infty} f(\tau)\, s(t')\, dt' = \int_{-\infty}^{\infty} e^{-\alpha\tau} \sum_{n}{}' \frac{A_n}{\omega_n} \sin(\omega_n \tau)\, (t'-t_1)\, [\theta\,(t' - t_1) - \theta\,(t' - t_2)]\, dt'$$

Letting $t_1 = 0$:

$$\psi(x,t) = \int_{-\infty}^{\infty} e^{-\alpha\tau} \sum_{n}{}' \frac{A_n}{\omega_n} \sin(\omega_n \tau)\, (t')\, [\theta\,(t') - \theta\,(t' - t_2)]\, dt'$$

Because this particular integral can be integrated relatively easily, we can straightforwardly obtain a solution that does not require computation of an impulse response function and a convolution. However, the disadvantage of doing it this way is that it is necessary to re-solve the problem for each new type of source function s(t') rather than merely perform a convolution. With an impulse response function, s(t') could represent, for example, a series of different types of plucks which could in turn represent all of the notes played by a particular string of an instrument during a performance.

Nevertheless, results obtained from the evaluation of this integral may be compared to those obtained through computation of an impulse response function and a convolution. Towards solving the direct substitution, changing the integration variable to τ and interchanging the integral and summation gives:

$$\psi(x,t) = \sum_{n}{}' \frac{A_n}{\omega_n} \int_{t - t_2}^{t} (t - \tau)\, e^{-\alpha\tau} \sin(\omega_n \tau)\, d\tau$$

with solution:

$$\psi(x,t) = \sum_{n}{}' \frac{A_n}{\omega_n} \left[\frac{e^{-\alpha t}}{(\alpha^2+\omega_n^2)^2} \left[(\alpha^2-\omega_n^2) \sin(\omega_n t) + 2\alpha\, \omega_n \cos(\omega_n t) \right] \right.$$

$$- \frac{e^{-\alpha (t - t_2)}}{(\alpha^2+\omega_n^2)^2} \left[(\alpha^2-\omega_n^2) \sin\left[\omega_n (t - t_2) \right] + 2\alpha\, \omega_n \cos\left[\omega_n (t - t_2) \right] \right]$$

$$\left. + \frac{t_2 e^{-\alpha (t - t_2)}}{(\alpha^2+\omega_n^2)} \left[\alpha \sin\left[\omega_n (t - t_2) \right] + \omega_n \cos\left[\omega_n (t - t_2) \right] \right] \right]$$

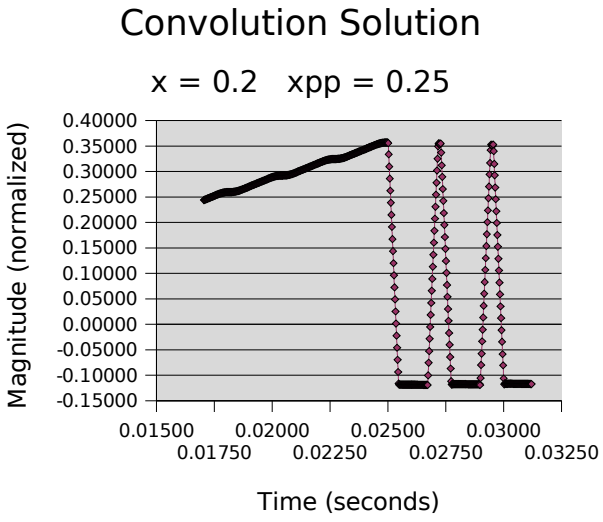
Using the same example as described in [2], the fundamental frequency is given by:

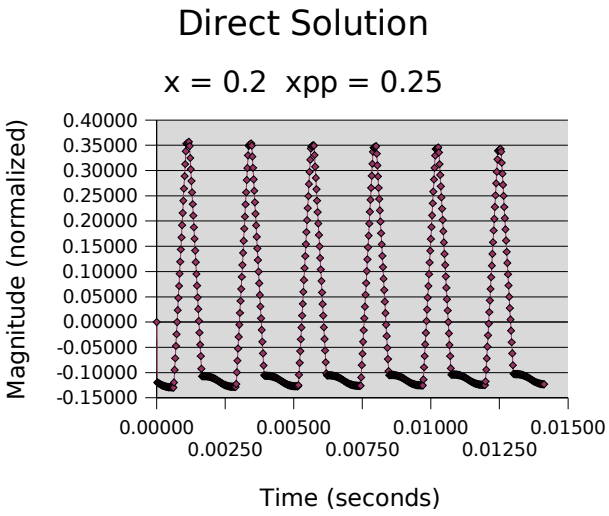
$$f_1 = 440\text{Hz}$$

For $\left(\frac{x}{a}\right) = \frac{1}{5}$ and $\left(\frac{x''}{a}\right) = \frac{1}{4}$:

$$\frac{A_n}{\omega_n} = \frac{1}{n} \sin\left(\frac{n\pi}{5}\right) \sin\left(\frac{n\pi}{4}\right)$$

Plotting versus time for both the convolution and direct solutions where no filtering has been applied:





The convolution solution shows the last portion of the motion of the string at position (x) due to the sawtooth waveform applied at (x'') as well as the motion due to release, followed by vibration of the string. The direct solution does not show the plucking motions. In an ideal solution, the string should remain at its displaced value (approximately -0.11) as it does in the convolution solution rather than move slightly as it does in the direct solution.

References and Notes

[1] David R. Clark. Green's function method for creating accurate stereo sound images. EXE Consulting, P.O. Box 450998, Garland, Texas 75045-0998, July 2004.

[2] David R. Clark. Green's function method for creating accurate stereo sound images: Simple 1-D Example. EXE Consulting, P.O. Box 450998, Garland, Texas 75045-0998, July 2004.

[3] Gabriel Barton. *Elements of Green's Functions and Propagation: Potentials, Diffusion, and Waves.* Oxford, 1989. Reprinted 1991.

Appendix

A simple C++ program for creating a plucked string data file using the convolution method is presented. Output is to standard out. The sound resembles a guitar at 440 Hz.

```
// One-dimensional pedagogical example for demonstrating application of
//    Green's function formulation.  No windowing or filtering is performed.
//    No FFT's.  (Exercise for intrepid readers.)
//    Plucked String
```

```
// Similar to electric guitar with strumming at one position, pickup at another

// Maximum frequency for spectra is one-half of sample rate
//    (e.g. 44.1 ksps -> 22.050 kHz)

// Dirac delta-function source at xpp = (x'' / length)
// Pickup at x = (x / length)
// Response convolved with sawtooth waveform to model plucking

// Compile:
//    $ g++ pluck.cc -o pluck

// WARNING: Be sure to REDIRECT OUTPUT:
//    $ pluck > pluck.out
// Use sox to convert text output to WAV file:
//    $ sox -t .dat pluck.out pluck.wav

// Be sure to try out different impulse and pickup positions because
//    these significantly change the timbre, just as with a real guitar.
//    When creating a guitar, try to use slightly different impulse and
//    pickup locations for each note to create more a more realistic sound.

// After generating WAV files, I recommend SPECIMEN for uploading into
//    a soundbank, then you should be able to play your instrument with
//    keyboard or MIDI file.  Although pitch-shifting in SPECIMEN will
//    work, I recommend creating individual notes or small groups for
//    more realistic and interesting sounds, varying the pluck and
//    pickup locations slightly.

// REDUCE amplitudes for full instruments.  Suggested values are HIGH.

// Placed in public domain.  No warranties expressed or implied.
//    By compiling this program, you agree to assume all risks associated
//    with this program and agree to indemnify author for all claims arising
//    from your use or misuse of this program, including derivatives, or
//    arising from your use or misuse of information contained herein.

#include <iomanip>
#include <iostream>
#include <cmath>

int main(int argc, char * argv[])   // One big, bad main...
{

// Get inputs:

   int i_sample_rate = 44100;
   std::cerr << "Sample rate (integer, suggest 48000 or 44100): ";
   std::cin >> i_sample_rate;

   double freq = 440.0;
   std::cerr << "Frequency (Hz, double, suggest ~440 for A): ";
   std::cin >> freq;

   double freq_max = double(i_sample_rate)/2.0;
   int N = (int)(rint(freq_max / freq));   // Close enough...

   int T = 2;
   std::cerr << "Length (seconds, integer, suggest 2): ";
   std::cin >> T;
   T = (int)(rint(2 * T * freq_max));

   double x = 0.2;
   std::cerr << "Pickup position (double, 0.0 < x < 1.0, suggest 0.1 to 0.2): ";
   std::cin >> x;
   double xpp = 0.25;
   std::cerr << "Impulse position (double, 0.0 < xpp < 1.0, suggest 0.25): ";
   std::cin >> xpp;
```

```
   double Amplitude = 4.5;
   std::cerr << "Amplitude (double, suggest 4.5): ";
   std::cin >> Amplitude;
   Amplitude = Amplitude * freq;

   int resolution = 16;   // Accuracy is actually double-precision, normalized
   double normalization = pow(2.0, double(resolution));

   double pi = 3.141592653589793;
   double * coeff = new double[N+1];
   double * f = new double[T+1];


// Sawtooth parameters:

   double saw_freq = 40.0;
   const double saw_amplitude = 1.0;
   double sample_rate = double(i_sample_rate);
   int saw_N = int(rint(sample_rate / saw_freq));
   double * sawtooth = new double[T+1];
   double * convolved = new double[T+1];


// A_n/omega_n:

   for (int n=1; n<=N; n++)   {
      coeff[n] = (1.0/double(n)) * sin(double(n) * pi * xpp) *
         sin(double(n) * pi * x);
   }

// f(t) with decay:

   double alpha = log(1e-3) / T;   // t60 for half-period (T)
   for (int t=1; t<=T; t++)   {
      f[t] = 0.0;
      for (int n=1; n<=N; n++)   {
         f[t] = f[t] + coeff[n] *
            sin(double(n) * pi * (2 * freq)  *  double(t)/sample_rate);
      }
      // decay:
      f[t] = f[t] * exp(alpha * double(t));
   }

// Create sawtooth

   for (int n=1; n<saw_N; n++)   {
      sawtooth[n] = (saw_amplitude * double(n) / double(saw_N));
   }
   for (int n=saw_N; n<=T; n++)   {
      sawtooth[n] = 0.0;   // Ensure zero
   }

// Convolve:

   std::cout << "; Sample Rate " << i_sample_rate << std::endl;
   std::cout << std::setw(20) << std::setprecision(20);
   std::cout << "0.0   0.0" << std::endl;
   for (int m=1; m<=T; m++)   {
      convolved[m] = 0;
      for (int t=1; t<=m; t++)   {
         convolved[m] = convolved[m] + (f[t] * sawtooth[m-t+1]);
      }
      std::cout << (double(m) / (sample_rate)) << "   "
         << Amplitude * convolved[m] / normalization << std::endl;
   }

   delete[] coeff;
   delete[] f;
```

```
     delete[] sawtooth;
     delete[] convolved;

     return(0);
  }

  /* Reference:

  [1] Gabriel Barton.  Elements of Green's Functions and Propagation:
      Potentials, Diffusion, and Waves.  Oxford,  1989.
*/
```