Music 206: Computational Acoustic Modeling for Sound Synthesis
Winter 2020

Tamara Smyth, trsmyth@ucsd.edu
Department of Music,
University of California, San Diego (UCSD)

January 28, 2020

Course Information

Meeting Time and Place

Lecture: Tuesday 2:00PM - 4:50PM, CPMC 367.

Office hours: CPMC 233 (my office) by appointment.

Course Description

This seminar introduces methods for discrete-time modelling of musical acoustic systems and delay-based audio effects. Covered topics, including filters, delay lines, flanging, aritificial reverb, sampled traveling waves, musical instrument acoustics/modeling, and acoustic measurement. Theory will be consolidated with practical programming assignments in Matlab.

Prerequisites

Music 170, 171 and 172, or equivalent, or permission by instructor.

Grading

- Assignments (6 total, 10% each) 60%
- Class participation 10%
- Paper presentation 10%

- Project 20%

Required Textbooks

- Music 206 on-line notes.

Resources

Matlab is available in the Music Lab, the Library, the Price Center, and any number of labs across the campus managed by ACMS. A list of the general access labs on campus is available here.

Reference

- The Physics of Musical Instruments Second Edition, by Neville Fletcher and Thomas Rossing.

Schedule and Online Lecture Notes (subject to change)

- Week 1
  - Class intro/admin;
  - Intro to Physical Modeling;
  - Filters and Delay I; (simple lowpass filter, delay line, comb filters);
  - Matlab tutorials 1 and 2
- Week 2
  - Filters and Delay II
- Week 3
  - Filters and Delay II cont.
  - Delay Effects: flanging, fractional delay (linear interpolation), multi-tap delay line, chorus, reverberation, dispersion (allpass filters), Karplus-Strong.
- Week 4
  - Digital Waveguides
  - wave equation, d'Alembert's solution, the digital waveguide, boundary conditions, the plucked string.
- Week 5

  - Paper presentations
- Week 6
  - Modeling Acoustic Tubes and Wind Instrument Bores/Bells
  - digital waveguide cont.; propagation loss; termination and scattering; change in cross-section area; two-port scattering junction; open-end reflection;
- Week 7
  - Mechanical Vibration mass-spring systems, simple harmonic motion, the equation of motion, discretization and the bilinear transform,
  - The pressure-controlled valve.
- Week 8
  - Filters III
- Week 9
  - Acoustic (impulse response) measurement
  - The Waveguide model in the frequency domain
  - Loose ends (other?)
- Week 10
  - Project presentations.

Assignments

- Assignment 1 (**due Tuesday January 14, 2020**):
  1. Read Smith and Jaffe's paper found here.
  2. Implement a parametric circular delay line in Matlab, where the single parameter M is the integer length of the effective delay.
     - your delayline should accommodate a number of *integer* delay values (this is the *parameter*) using a single delayline;
     - implement the integer sample delay using a *read* and *write* pointer to a delay line having a length that's set according to the longest expected delay (lowest frequency);
     - to make sure you implemented correctly, check its *impulse response*: the output in response to an *impulse* as input.
  3. Use your delay line to create a *feedforward* comb filter with gain coefficient g.

     - how does your output compare to the output of Matlab's filter function?

```
Matlab M-sample delay feedforward comb filter

M = ...;                % sample delayn
g = ...;                % comb filter coefficient
N = ...;                % signal length
x = [1; zeros(N-1,1)]; % input impulse

filter([1; zeros(M-1); g], 1, x);
```

  Solutions:
  - cdl.m
  - cdlffcomb.m
- Assignment 2 (due Tuesday January 21, 2020):

  Beginning with your circular delay line:
  1. implement fractional delay using linear interpolation
  2. implement a flanger with a variable length delay that oscillates between some maximum and minimum value (with linear interpolation for fractional delay).
  3. implement a chorus effect with a multi-tap delay line.
  4. implement a feedback comb filter with feedback coefficient g

  Solutions:
  - flanger.m (flanger with linear interpolation)
- Assignment 3 (due Tuesday January 28, 2020):

  Beginning with your (non-interpolating) circular delay line:
  - implement a feedback comp filter, setting feedback coefficient g given note duration (T60) and integer sample delay M.
  - implement the karplus strong for an integer delay using a noise burst for excitation and the simple lowpass filter in the feedback loop.

  Solutions:
  - karplus.m (simple Karplus-Strong string, with integer delay)
- Assignment 4 (due Tuesday February 4, 2020):

  Beginning with your Karplus-Strong pluck string model, extend using
  - allpass fraction delay for tuning

- decay length (note duration) shortening for low frequencies
- BONUS: decay length stretching so you can specify a note duration (this will create a change in the phase delay of the lowpass filter and the overall loop duration and sounding frequency).

• Assignment 5 (due Tuesday Febrary 18, 2020):

Implement a 1-D digital waveguide (bi-directional delay line) of a plucked string.

- Use your non-interpolating delay line to create your wavegude structure: begin with inversions but no losses at each end.
- **Check 1**: Input and impulse at one end. If tapped at the same location as the input, the impulse response should be a sequence of 1s seperated by N-1 zeros (where N is the round-trip delay, and N/2 is the delay for each delay line).
- **Check 2**: make sure the sum of right and left traveling waves is always zero at the boundaries.
- Add lowpass filter. Is it beginning to sound string like?
- Choose a plucking position and initialize the string to a triangle shape (as shown in class and in notes).
- Choose a pickup point and sum upper and lower delaylines at that point to produce the output waveform. Listen!

• Assignment 6 (due Tuesday February 25, 2020): Implement a sequence of N 2-port scattering junctions and use it to model the human vocal tract when vocalizing any of the following vowel sounds:

- 3-bird
- A-bart
- ae-bat
- E-bet
- i-beet
- I-bit
- O-ball
- u-food
- U-foot
- V-but

where each linked file is a sequence of cross-sectional areas.

1. BEFORE using the files, test with a simple vector of cross-sectional areas that are all the same,


```
S = [1 1 1 1];
Nj = length(S) - 1;
k = (S(1:end-1) - S(2:end)) ./ (S(1:end-1) + S(2:end));
```

as this should produce the same output as a purely cylindrical waveguide model.

2. LOAD files into Matlab:

```
load 'A-bart44100.txt';
```

Try your model with the following source as input:

- source.wav