Audio Signal Processing Laboratory #2: Schroeder Reverb

Intro Video: https://youtu.be/z6Jop3EGBdw

1

What to Submit:

- Your code and Simulink block diagrams. Make sure you are saving all Simulink models to your Current Folder in MATLAB
- A report with answers, figures, and explanations
- Any sound files that are asked for
- When to Hand It In:
  - It is due immediately at the beginning of the day (i.e., 0:00AM) specified on the course calendar. **Late assignments will receive a grade deduction of 20% of the full grade each day.**

2

How to Submit:

- Put all your files in one folder. Compress this folder and name it <firstname>_<lastname>_HW6 .zip , ie “Zhiyao_Duan_HW6.zip”.
- Submit via blackboard to the appropriate entry. You can make multiple attempts but only the last attempt will be graded.
- **New to these labs:** You are provided a script that allows you to demonstrate your real-time Simulink audio. There is a sample demonstration in the ”Problem 5 Deliverables” slide on the last page of this assignment.

If you don’t have a computer with a built-in microphone or an external microphone you can use, let us know ASAP

3

Problem 1: Read Papers, Set up Code Directory

While there is no code involved in this problem, it is still vital to complete this problem and answer questions in your report

4

Part A) Read Schroeder Reverb Papers (1 Point)

Read “Schroeder_1961.pdf” and “Schroeder_1962.pdf”. In your report, use your own words to explain the main benefits of using/adding a series of all-pass filters in a reverberator implementation.

http://www2.ece.rochester.edu/~zduan/teaching/ece472/reading/Schroeder_1961.pdf

http://www2.ece.rochester.edu/~zduan/teaching/ece472/reading/Schroeder_1962.pdf

5

Part B) Put code in folder from last time (0 Points)

The classes we will be constructing in this lab are dependent on the classes we made in the previous assignment. Therefore, move all the skeleton files provided in the folder ”Additional Codes HW7” into your MATLAB directory from the last assignment.

Do not just put the provided folder into the old folder unless you know how to add sub-folders into your working directory (and even then, you need to do it each time you launch MATLAB). Simply click and drag or copy and paste the new files into your existing directory.

6

Problem 1 Deliverables

- A summary in your own words why all-pass filters are used in Schroeder's reverb structure (1 point)
- Your directory should be set up correctly (0 points)

7

Problem 2: CombFilter and AllPassFilter Classes

This problem involves the implementation of two helper classes that will be involved in the overall reverb structure's process function

Video Tutorial: https://youtu.be/hi52vTZE-nE

8

Part A) Implement CombFilter (1 point)

From Schroeder 1962, the following represents the overall reverb structure we'll be implementing in this lab assignment:

These highlighted items are block diagrams for a comb filter structure. Therefore, just like last lab, we can implement a process function to process one sample through the filter at a time

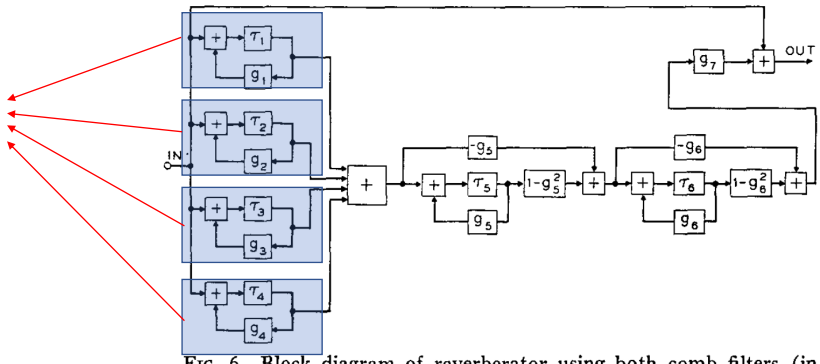
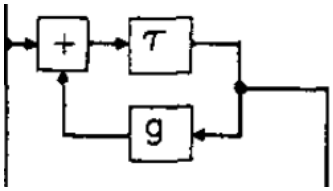


FIG. 6. Block diagram of reverberator using both comb filters (in parallel) and all-pass filters (in series). This type of reverberator allows the widest choice of specifications such as mixing ratios, delay of reverberated sound and kind of decay.

9

Fill in the blanks in the provided *CombFilter.m*, which implements the following structure:

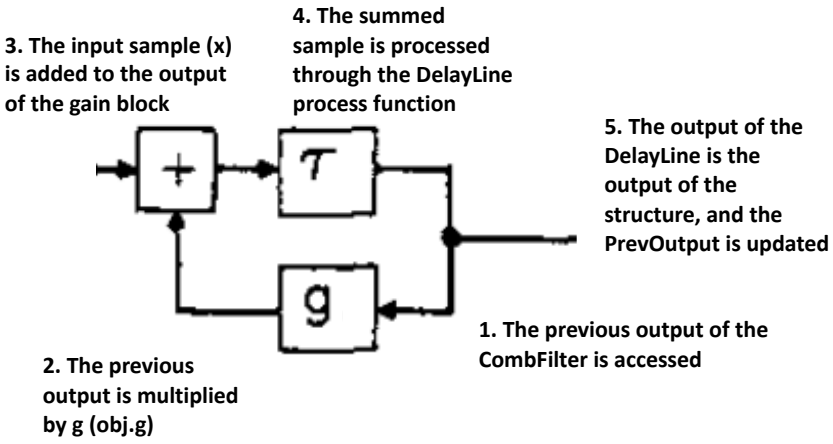


The properties (tau, g, previous output, and comb_delay) are provided, as well as the constructor. Notice that, in the constructor, the property comb_delay is made a (integer) DelayLine with a buffer the size of input parameter tau_in (which also becomes obj.tau). This DelayLine in the above block diagram is the block labeled τ (or tau). Recall that this block has its own process function, accessible by obj.comb_delay.process(…). The block labeled 'g' is simply a gain, ie, the output of the block labeled 'g' is (input * g).

The desired process function for this comb filter is illustrated in the next slide:

10

Implementation: Simply walk through the block diagram, reproduced below:



The process function that's been provided is started but incomplete. The steps are broken out into different lines of code, such that a running 'output' value is overwritten at each step. This is one strategy of sequential programming languages like MATLAB

11

Part B) Implement AllPassFilter

(1 point)

These highlighted items are block diagrams for an all pass filter structure. Therefore, just like last lab, we can implement a process function to process one sample through the filter at a time

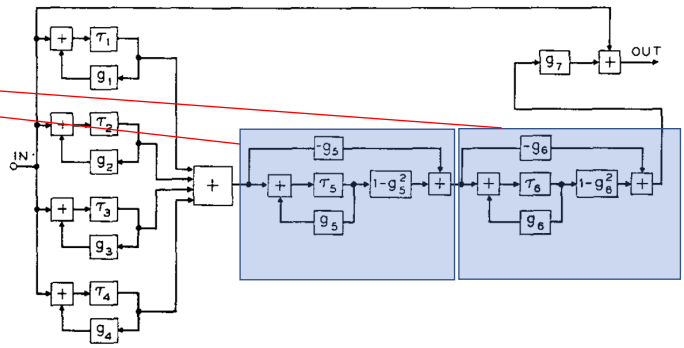


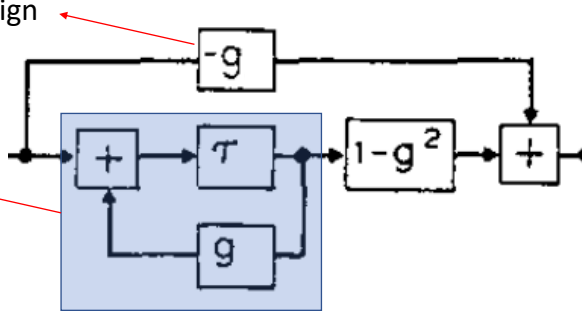
FIG. 6. Block diagram of reverberator using both comb filters (in parallel) and all-pass filters (in series). This type of reverberator allows the widest choice of specifications such as mixing ratios, delay of reverberated sound and kind of decay.

12

Fill in the blanks in the provided *AllPassFilter.m*, which implements the following structure:

Notice the minus sign

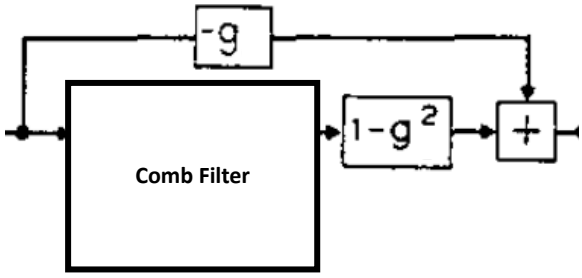
Notice, this is the comb filter structure that we just implemented



The properties (tau, g, and allpass_comb) are provided, as well as the constructor. Notice that, in the constructor, the property allpass_comb is made to be an instance of a comb filter, the class written in the previous part. Recall that this block has its own process function, accessible by obj.allpass_comb.process(…).

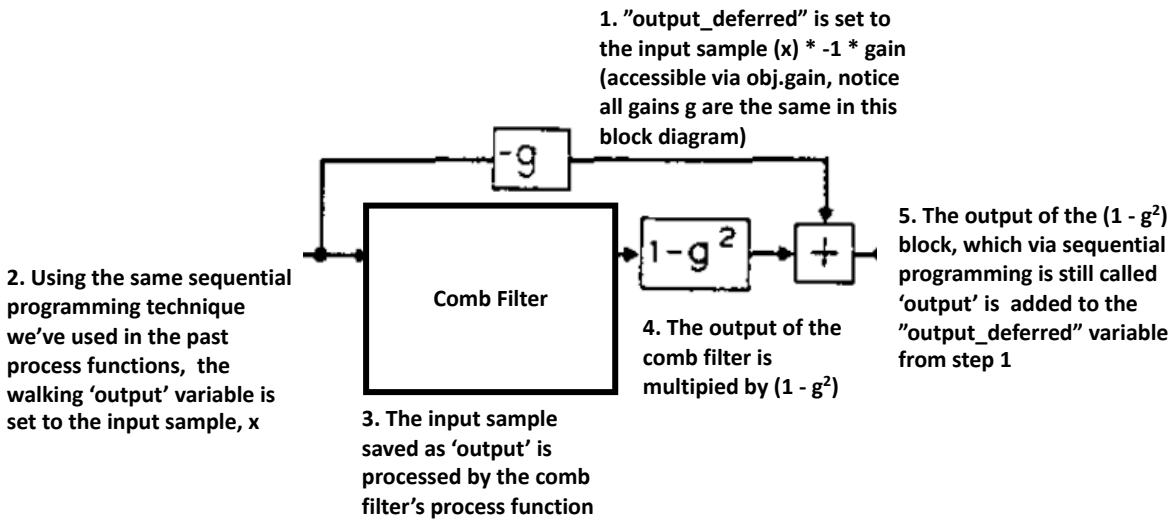
13

Therefore, we can re-write the block diagram of this all pass filter as follows:



The desired process function for this filter is illustrated in the next slide:

14



The process function that's been provided is started but incomplete. The steps are broken out into different lines of code, such that a running 'output' value is overwritten at each step.

15

Problem 2 Deliverables

- Implemented CombFilter.m (1 point)
- Implemented AllPassFilter.m (1 point)
- **In your report:** make sure you answer all questions posed in this PowerPoint, and give an explanation of what you did and why you did it

16

Problem 3: ReverbSchroeder Class

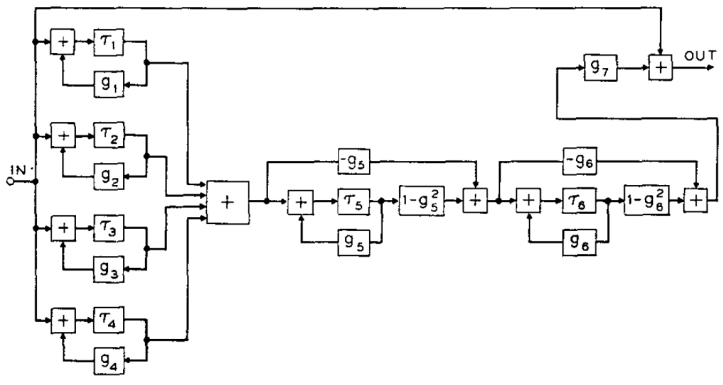
In this problem, the Schroeder Reverb block diagram will be created as it's own class with it's own process function

Video Tutorial: https://youtu.be/GtcPW2o3_0Q

17

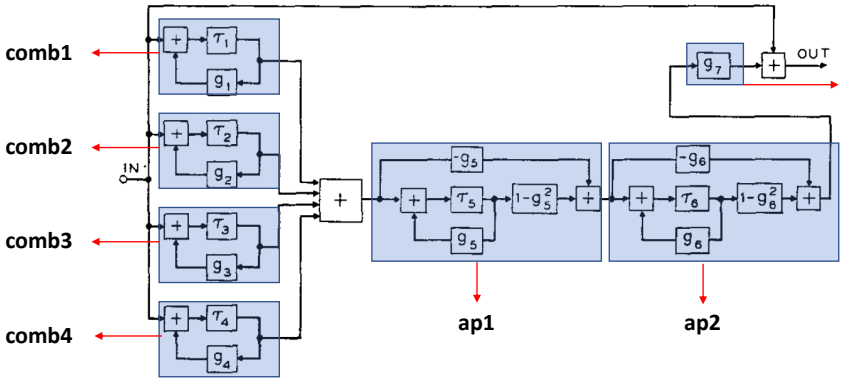
Part A) ReverbSchroeder (2 points)

Fill in blanks in *ReverbSchroeder.m*. This class implements the Schroeder reverb structure reproduced below:



18

The properties and constructor have been provided for you, you must finish writing the process function. There are seven total properties in this class, comb filters 1 through 4, all pass filters 1 and 2, and gain g7. These are all labeled below:



g7 notice how the output of the comb and all pass filters is being multiplied by g7, and then *mixed with the original input audio sample* through the next summing junction. For this reason, g7 should be thought of as controlling the *wet/dry ratio*

19

Quick Note on g7

- While we say that g7 controls the wet/dry ratio, please note that it is not directly the wet/dry ratio. Note that we have no real control over how loud the original input sample is at the output branch, we can only control the reverb portion via g7. Therefore, g7 isn't directly the proportion of wet to dry audio but changing g7 changes our perception of how 'wet' the mix is.

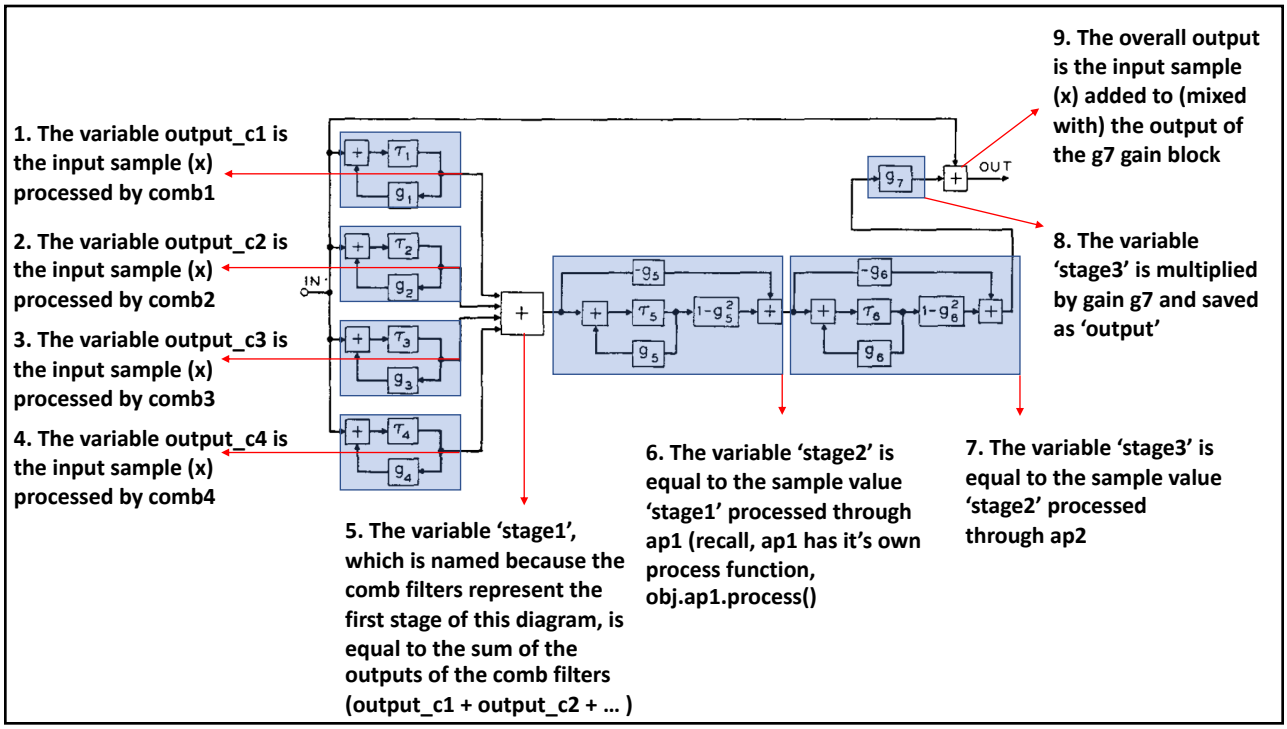
20

The Constructor: Notice in the provided code that there are many input arguments, because we need to initialize four comb filters and two all pass filters (which each contain their own comb filter as a property).

In the next problem, the method to calculated the comb-filter delayline lengths tau1 through tau6 will be discussed, as well as filter gains g1 though g6. g7 is a separate input argument. As this controls the wet-dry ratio, this gain should be played around with similarly to how this parameter is played around with in your DAW until a well-mixed sound is attained.

The Process Method: You must finish writing the process method. The desired process for this filter is illustrated in the next slide:

21



22

Problem 3 Deliverables

- Implemented ReverbSchroeder.m (2 points)
- **In your report:** make sure you answer all questions posed in this PowerPoint, and give an explanation of what you did and why you did it

23

Problem 4: Calculating Tau and Gain values for Four Different Rooms

Now that we have a working reverb class, we can apply an instance of this class to process audio as it would sound in several different rooms

Video Tutorial: https://youtu.be/Cikr7sTF-Lc

24

How to calculate T_{60} Values:

T_{60} is a measure of how long it takes for the energy in a room from an acoustic impulse to decay by 60 dB. The 'lingering energy' is the reverberant audio bouncing off walls within the space. Therefore, the larger the T_{60}, the longer the reverb trail, as the energy of the reverberant audio lasts longer.

To calculate T_{60}, sometimes called just T, we know the damping coefficient α of each wall in the space and the size of each wall. We can use this equation, reproduced from the lecture notes:

$$T_{60} = 0.163 \frac{V}{\alpha S} = 0.163 \frac{V}{\sum_n \alpha_n S_n}$$

25

How to calculate T_{60} Values: *(continued)*

$$T_{60} = \boxed{0.163 \frac{V}{\alpha S}} = 0.163 \frac{V}{\sum_n \alpha_n S_n}$$

In the above equation, V is the volume of the room in meters^3, α_n is the damping coefficient of the n^{th} wall in the room which has surface area S_n.

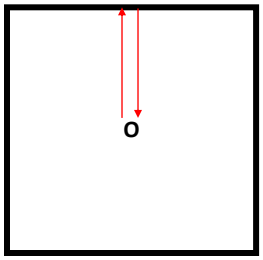
For the problems in this homework assignment, all surfaces will have the same damping coefficient α. Therefore, we don't need to use the summation shown, and only use the highlighted portion of the formula, where S is the total surface area in the room (which includes all four walls, the ceiling, and the floor). Therefore, T_{60} can be calculated for any room with known dimensions and the damping coefficient α of the walls in the room. *Note, $0 < \alpha < 1$*

26

How to calculate tau Values:

The physical interpretation of the tau1 through tau4 values that they *capture the early reflections in the room*. Remember, tau corresponds to the length of a delayline, so in order to get a reverberant sound, we're using these delaylines to simulate reflection times within the room.

Therefore, we say by convention that tau1 is the value of the earliest reflection in the room.



The total path length is thus equal to the 2*(0.5 * length of the shortest dimension), or just the length of the shortest dimension

27

How to calculate tau Values: *(continued)*

Recall that the speed of sound in air (c) is roughly 343 m/s, so we can figure out how long it takes sound to travel the shortest path back to the speaker. Since the path length is the same as the smallest room dimension, we can calculate the number of seconds it takes for sound to travel this path, which is essentially calculating the number of milliseconds that pass before the first reflection is heard.

Time until first reflection (milliseconds) = (Path Length / Speed of Sound) * 1000

The multiplication by 1000 is to convert from seconds to milliseconds. For acoustic path lengths, it is often more convenient to use milliseconds (ms) due to the scale of such times.

28

How to calculate tau Values: *(continued)*

Now that we have the time until the first reflection, we can calculate tau1. Recall that tau1 is the number of samples in an integer delay line, therefore, we need to convert the number of milliseconds until the first reflection to the number of samples using the sampling rate fs. We must also round it, as this is an integer delayline.

tau1 = ms_to_samples(Time until first reflection (milliseconds))

In just a moment, we will make the helper function ms_to_samples.

29

How to calculate tau Values: *(continued)*

Schroeder's 1962 paper instructs that tau1:tau4 be spread out on a scale of 1.0:1.5. This means that the tau1 through tau4 values should range from 1.0*tau1 to 1.5*tau1. Splitting this range evenly yields:

*tau1 = 1.0*tau1*
*tau2 = 1.16*tau1*
*tau3 = 1.32*tau1*
*tau4 = 1.5*tau1*

Therefore, once we calculate tau1 from the earliest reflection, we can calculate the values of tau2, tau3, and tau4.

30

How to calculate tau Values: *(continued)*

Schroeder's 1962 paper also instructs that tau5 and tau6 be fixed a delay of 5 ms and 1.7 ms, respectively. Therefore, we can use the function we're about to write, ms_to_samples, to get the following:

tau5 = ms_to_samples(5 ms)
tau6 = ms_to_samples(1.7 ms)

Thus, for each room, we can obtain all tau values. We must now calculate all the gain values, g1 – g7

31

How to calculate Gain Values:

Schroeder's 1962 paper gives the following formula for n = 1:4

$$\mathrm{T60} = 3\tau_n/(-\log_{10} g_n)$$

Once we've calculated T_{60} and tau1 through tau4, we can re-arrange this formula to get g1 through g4:

$$g_n = 10\text{^}((-3 * \tau_n)/\mathrm{T60})$$

The above formula can be used to calculate g1 through g4, we can again refer to the paper for a suggestion on g5 and g6. The paper instructs to set g5 and g6 to both equal to 0.7

Note: tau_n in these formulas is the tau value in SI units, in this case, SECONDS. Therefore, you must convert your tau value to seconds to calculate gain

32

Setting g7:

Once again, g7 is to be interpreted as the control for the wet/dry ratio for the reverb. For the following problems, g7 should be played with until a good mix is achieved to demonstrate your reverb. In your real-time implementation in Simulink, a slider will be implemented to control g7. For this problem, set **g7 = 0.1**.

33

Part A) ms_to_samples (0 points)

Fill in the blanks in *ms_to_samples.m*. This function should take in a value in milliseconds and a sampling frequency and determine the amount of samples it takes to represent the milliseconds value.

For the purposes of this function, **do not round the output**. We will round everything when we call the constructor function to make our ReverbSchroeder instances.

34

Part B) Small Bright Room (0.5 points)

Fill in the blanks in *smallbrightroom.m*.

Step 1: Start by calculating on pencil and paper the desired T_{60} for a small bright room (such as an Eastman practice room) with the following parameters:

Dimensions 4*2.5*3 meters with 15% ($\alpha = 0.15$) sound absorbtion at each wall

Put your answer for T_{60} of this room in your report.

35

Part B) Small Bright Room *(continued)*

Step 2: Use the smallest room dimension and your helper function ms_to_samples to calculate the value integer of tau1. Then, calculate tau2_ms, tau3_ms, and tau4_ms using the previously presented ratios, and convert them to samples as well. Recall the fixed values of tau5 and tau6. **State your value for tau1_ms in your report**

Step 3: Use the gain formula to calculate g1 through g4 from the values of tau1 through tau4. Do this in-line in your code. Recall that g5 and g6 are fixed to 0.7, and g7 is fixed to 0.1 **State your value for g1 in your report**

Step 4: Call the constructor function to create an instance of ReverbSchroeder.

Step 5: Sample by sample, process the provided audio file "violin_44100.wav". Save your output as "smallbrightroom.wav". Make sure you listen to the audio, and make sure the reverb sounds like it should in a small bright room. **Submit the output wav file**

36

Part C) Small Dead Room (0.5 points)

Fill in the blanks in *smalldeadroom.m*.

Step 1: Start by calculating on pencil and paper the desired T_{60} for a small dead room (such as a dorm room full of stuff) with the following parameters:

Dimensions 4*2.5*3 meters with 70% ($\alpha = 0.70$) sound absorbtion at each wall

Put your answer for T_{60} of this room in your report.

37

Part C) Small Dead Room *(continued)*

Step 2: Use the smallest room dimension and your helper function ms_to_samples to calculate the value integer of tau1. Then, calculate tau2_ms, tau3_ms, and tau4_ms using the previously presented ratios, and convert them to samples as well. Recall the fixed values of tau5 and tau6. **State your value for tau1_ms in your report**

Step 3: Use the gain formula to calculate g1 through g4 from the values of tau1 through tau4. Do this in-line in your code. Recall that g5 and g6 are fixed to 0.7, and g7 is fixed to 0.1 **State your value for g1 in your report**

Step 4: Call the constructor function to create an instance of ReverbSchroeder.

Step 5: Sample by sample, process the provided audio file “violin_44100.wav”. Save your output as ”smalldeadroom.wav”. Make sure you listen to the audio, and make sure the reverb sounds like it should in a small bright room. **Submit the output wav file**

38

Part D) Large Bright Room (0.5 points)

Fill in the blanks in *largebrightroom.m*.

Step 1: Start by calculating on pencil and paper the desired T_{60} for a large bright room (such as a church) with the following parameters:

Dimensions 30*50*10 meters with 15% ($\alpha = 0.15$) sound absorbtion at each wall

Put your answer for T_{60} of this room in your report.

39

Part D) Large Bright Room *(continued)*

Step 2: Use the smallest room dimension and your helper function ms_to_samples to calculate the value integer of tau1. Then, calculate tau2_ms, tau3_ms, and tau4_ms using the previously presented ratios, and convert them to samples as well. Recall the fixed values of tau5 and tau6. **State your value for tau1_ms in your report**

Step 3: Use the gain formula to calculate g1 through g4 from the values of tau1 through tau4. Do this in-line in your code. Recall that g5 and g6 are fixed to 0.7, and g7 is fixed to 0.1 **State your value for g1 in your report**

Step 4: Call the constructor function to create an instance of ReverbSchroeder.

Step 5: Sample by sample, process the provided audio file "violin_44100.wav". Save your output as "largebrightroom.wav". Make sure you listen to the audio, and make sure the reverb sounds like it should in a small bright room. **Submit the output wav file**

40

Part E) Large Dead Room (0.5 points)

Fill in the blanks in *largedeadroom.m*.

Step 1: Start by calculating on pencil and paper the desired T_{60} for a large bright room (such as a concert hall with full audience) with the following parameters:

Dimensions 30*50*10 meters with 70% ($\alpha = 0.70$) sound absorbtion at each wall

Put your answer for T_{60} of this room in your report.

41

Part E) Large Dead Room *(continued)*

Step 2: Use the smallest room dimension and your helper function ms_to_samples to calculate the value integer of tau1. Then, calculate tau2_ms, tau3_ms, and tau4_ms using the previously presented ratios, and convert them to samples as well. Recall the fixed values of tau5 and tau6. **State your value for tau1_ms in your report**

Step 3: Use the gain formula to calculate g1 through g4 from the values of tau1 through tau4. Do this in-line in your code. Recall that g5 and g6 are fixed to 0.7, and g7 is fixed to 0.1 **State your value for g1 in your report**

Step 4: Call the constructor function to create an instance of ReverbSchroeder.

Step 5: Sample by sample, process the provided audio file “violin_44100.wav”. Save your output as ”largedeadroom.wav”. Make sure you listen to the audio, and make sure the reverb sounds like it should in a small bright room. **Submit the output wav file**

42

Problem 4 Deliverables

- Filled in ms_to_samples function (0 points)
  - While this helper function is worth 0 points, it is required to carry out the rest of this problem
- Filled in scripts smallbrightroom, smalldeadroom, largebrightroom, and largedeadroom (2 points)
  - **In your report,** report all T_{60}, tau1_ms, and g1 calculations
  - Submit all required audio files
  - Each room is worth 0.5 points

43

Problem 5: Real Time Implementation of Reverb

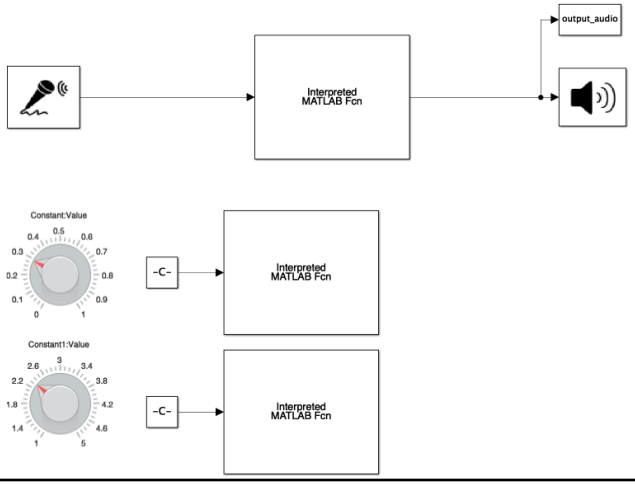
In this problem, we'll use Simulink to create a real time version of adding reverb to input audio.

NOTE: If you don't have a microphone on your computer or a microphone you can use, or headphones, let us know ASAP

Video Tutorial: https://youtu.be/4-ge8XIIYXY

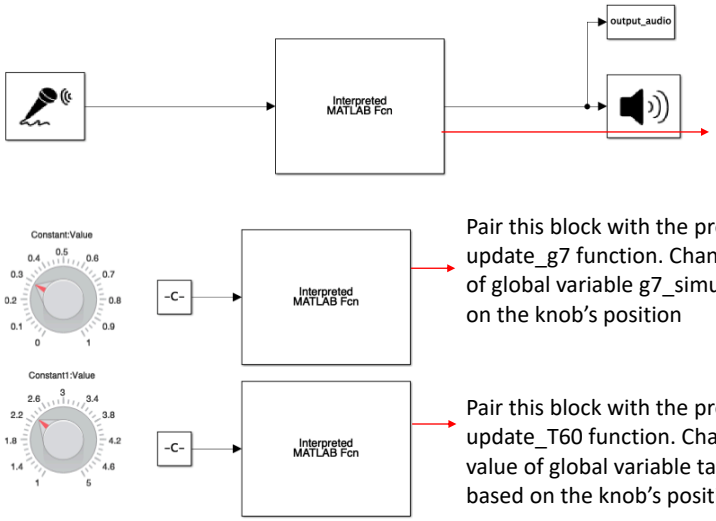
44

Objective: Use this Simulink model to process audio from your microphone through a ReverbSchroeder object. Make functions so that the ReverbSchroeder object can have it's wet/dry ratio (g7), it's T60, and it's tau1 (**in milliseconds**) in real time. The following Simulink model is provided. You will be implementing a third knob to control tau1_ms



45

You'll be using the given starter Simulink model reverb_simulink.slx, which is reproduced below. Each MATLAB Fcn block is described as follows:



Pair this block with the provided function process_reverb. Process the incoming audio frame through a Schroeder reverb object's process method sample by sample.

Pair this block with the provided update_g7 function. Change the value of global variable g7_simulink based on the knob's position

Pair this block with the provided update_T60 function. Change the value of global variable tau1_simulink based on the knob's position

46

Part A) Initialization

Fill in the blanks in the provided *init_reverb.m* function. Our goal in this script is to define our global variables that will be accessible in both MATLAB and Simulink. The following shows the script:

```
% Define global variables to pass by reference:
global T60_simulink;
global g7_simulink;
global tau1_ms_simulink

% Provide initial values for
T60_simulink =  % FILL IN;   % Calculated T60 from the large bright room (church)
tau1_ms_simulink = % FILL IN    % Calculated tau1 from the large bright room (church)
g7_simulink = 0.1;

% call a script that will generate (and re-generate) a reverb object
gen_rev;
```

All you must do here is fill in the initial 'T60_simulink' and "tau1_ms_simulink" values from your large bright room calculation. We will implement the gen_rev script in the next part

47

Part B) Reverb Generation

Fill in the blanks in the provided *gen_rev.m* function. Here, we input our global variables by reference, and then follow the same algorithm as in Problem 4 to create a global reverb object, which we will call 's_rev' for Schroeder reverb. As you will see in the script, it's laid out with the same steps as the rooms from the previous project. Make sure you use the same values as your largebrightroom script. When you calculate the new g1-g4 values, make sure you use the global T60_simulink value as T60. This, and the generation of s_rev, are shown in the snippet below:

```
% Use these tau values in seconds to compute the value of g1-g4 using the
% equation from the assignment. Recall g5-g7 are given
g1 = %FILL IN, use T60_simulink
g2 = %FILL IN, use T60_simulink
g3 = %FILL IN, use T60_simulink
g4 = %FILL IN, use T60_simulink
g5 = %FILL IN
g6 = %FILL IN
g7 = g7_simulink;

% STEP 4: Create a global instance of ReverbSchroeder:
global s_rev;
s_rev = ReverbSchroder( % FILL IN
```

48

Part C) interupt_reverb

Fill in the blanks in *interupt_reverb.m*, the 'interrupt function' in your Simulink diagram that leads to the processing of input audio frames sample by sample. This algorithm will look very similar to the previous homeworks string processing interrupt function, except this time we want to use the input audio instead of just using the input audio frames for timing.

```
function [out_frame] = interupt_reverb(in_frame)

global s_rev; % pass by reference

in_len = length(in_frame);
out_frame = zeros(size(in_frame));


for idx = 1:in_len
   out_frame(idx) = % FILL IN, remember, we're processing through s_rev
end



end
```

49

Part D) update_g7

Fill in the blanks in update_*g7.m*, the 'interrupt function' in your Simulink diagram that leads to the updating of the g7_simulink global variable. We want to create a new s_rev object **only when** the knob controlling g7_simulink changes.

```
function update_g7(g_in)

% pass by reference:
global g7_simulink;

% Check to make sure g7 has changed:
if g7_simulink ~= % FILL IN
    g7_simulink = % FILL IN

    % Call the generator function with the new g7_simulink value, which is
    % already passed by reference within gen_rev
    gen_rev;
end


end
```

50

Part E) update_T60

Fill in the blanks in update_*T60.m*, the 'interrupt function' in your Simulink diagram that leads to the updating of the T60_simulink global variable. We want to create a new s_rev object **only when** the knob controlling T60_simulink changes.

```
function update_T60(T60_in)

% Pass by reference
global T60_simulink;


if T60_simulink ~= % FILL IN
    T60_simulink = % FILL IN

    % Call the generator function with the new T60_simulink value, which is
    % already passed by reference within gen_rev
    gen_rev;
end

end
```

51

Part F) Add a knob in Simulink for controlling the tau1_ms variable

In your existing Simulink file, make a new knob to control the tau1_ms variable. To do this, remember to launch the Library Browser and search for "knob". Then, add a constant block by searching for "constant". In the same fashion as in Lab #1, pair the knob with the constant value. Make the knob have a linear scale, **starting at 5 and ending at 500**. Remember, these values will be millisecond values. **Turn the knob to about your initial tau1_ms value before starting the simulation**, as the global initialization ocassionally gets overwritten with knob value

Now add an "Interpreted MATLAB Function" block. Make it call the function "update_tau1_ms" function that we'll write in the next part. Make it's output dimensions equal to 0.

If you need further assistance adding to a Simulink diagram, watch the video from the previous homework for the Simulink problem.

52

Part G) update_tau1_ms

Fill in the blanks in *update_tau1_ms.m*, the 'interrupt function' in your Simulink diagram that leads to the updating of the tau1_ms_simulink global variable. We want to create a new s_rev object **only when** the knob controlling tau1_ms_simulink changes.

```
function update_tau1_ms(tau1_ms_in)

% pass by reference:
global tau1_ms_simulink;

% Check to make sure g7 has changed:
if tau1_ms_simulink ~= % FILL IN
    tau1_ms_simulink =  % FILL IN

    % Call the generator function with the new g7_simulink value, which is
    % already passed by reference within gen_rev
    gen_rev;
end

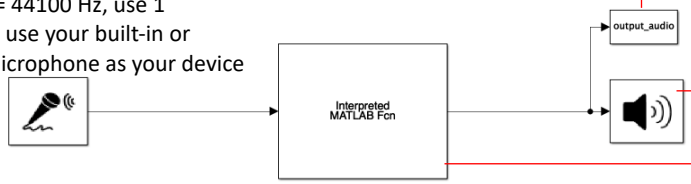

end
```

53

Part H) Set up Simulink

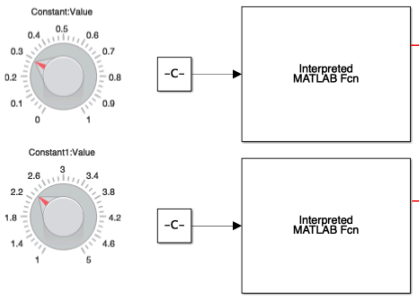
Use frame size of 1024 samples, make sure your fs = 44100 Hz, use 1 channel, and use your built-in or plugged in microphone as your device

This block will output your demonstration to the MATLAB workspace, more on this in the next slide



Plug in headphones and make this block output to your headphones, use fs = 44100 Hz.

Make sure this block is paired with interupt_reverb



Make sure this block is paired with update_g7

Make sure this block is paired with update_T60

Make sure your tau1_ms_update knob, constant, and FCN block is also set up correctly.

54

Important Note:

Make sure you call the script init_reverb in the MATLAB command window before hitting “play” on your Simulink model. This script (along with gen_rev, which it calls) creates the instance of s_rev that is used in the Simulink FCN blocks

55

Part I) Do your Demo

From the previous slide, you’re the “output_audio” block will save a variable called “out” to your MATLAB workspace, after you hit the “stop” button on Simulink to end your demonstration. You should call the provided MATLAB script **demo_audio AFTER** you stop you Simulink demonstration. This will save the audio file “reverb_demo.wav” to your directory. **Make sure you submit this audio file**. This file will contain all the output audio from your Simulink diagram from the time you start to the time you stop your demonstration.

Make sure to verbally announce what you’re doing (ie “I’m about to change my T60 value so you should hear the time it takes for the reverb to decay get longer” or “I’m about to change by wet/dry ratio, so you should hear less direct audio”, etc).

An example demonstration file is provided here:
https://youtu.be/tTN_n4vAFbc

56

Problem 5 Deliverables

- All initialization, generation, interrupt, and update functions are completed (1 points)
- The Simulink model is verified and runs as expected (2 points)
  - The way we will verify this is through your demo recording, so make sure you submit **reverb_demo.wav** as part of your submission.
  - An example demonstration is provided here: https://youtu.be/tTN_n4vAFbc

    KEEP IN MIND: we won't be able to see a video of you moving the knobs, make sure you explain what you're doing verbally as you do it

  - Make sure you briefly comment on what you hear as you move the knobs in your **report**

57