AGSynth - Granular Synthetizer for Automotive

Edoardo Di Pietrantonio and Giorgio Povegliano
edoardo.dipietrantonio@studenti.unipd.it
giorgio.povegliano@studenti.unipd.it

1 Introduction

Welcome to our guide! We will explain you every feature of the current version of our Granular Synth for Automotive. Before getting into the commands, it is necessary to give an overview of what is granular synthesis and show the general structure of the synthesizer.

1.1 Requirements

Here is a list of necessary software to run the application:

- MatLab 2023b
- Simulink

Additionally, some MatLab and Simulink Add-Ons are required:

- Audio Toolbox
- Control System Toolbox
- DSP HDL Toolbox
- DSP System Toolbox
- MatLab Coder and MatLab Compiler
- Simulink Coder and Simulink Compiler
- Vehicle Network Toolbox
- Signal Processing Toolbox
- MATLAB Support for MinGW-w64 C/C++/Fortran Compiler
- Fixed-Point Designer

To install Add-Ons you can use MatLab Add-On explorer.

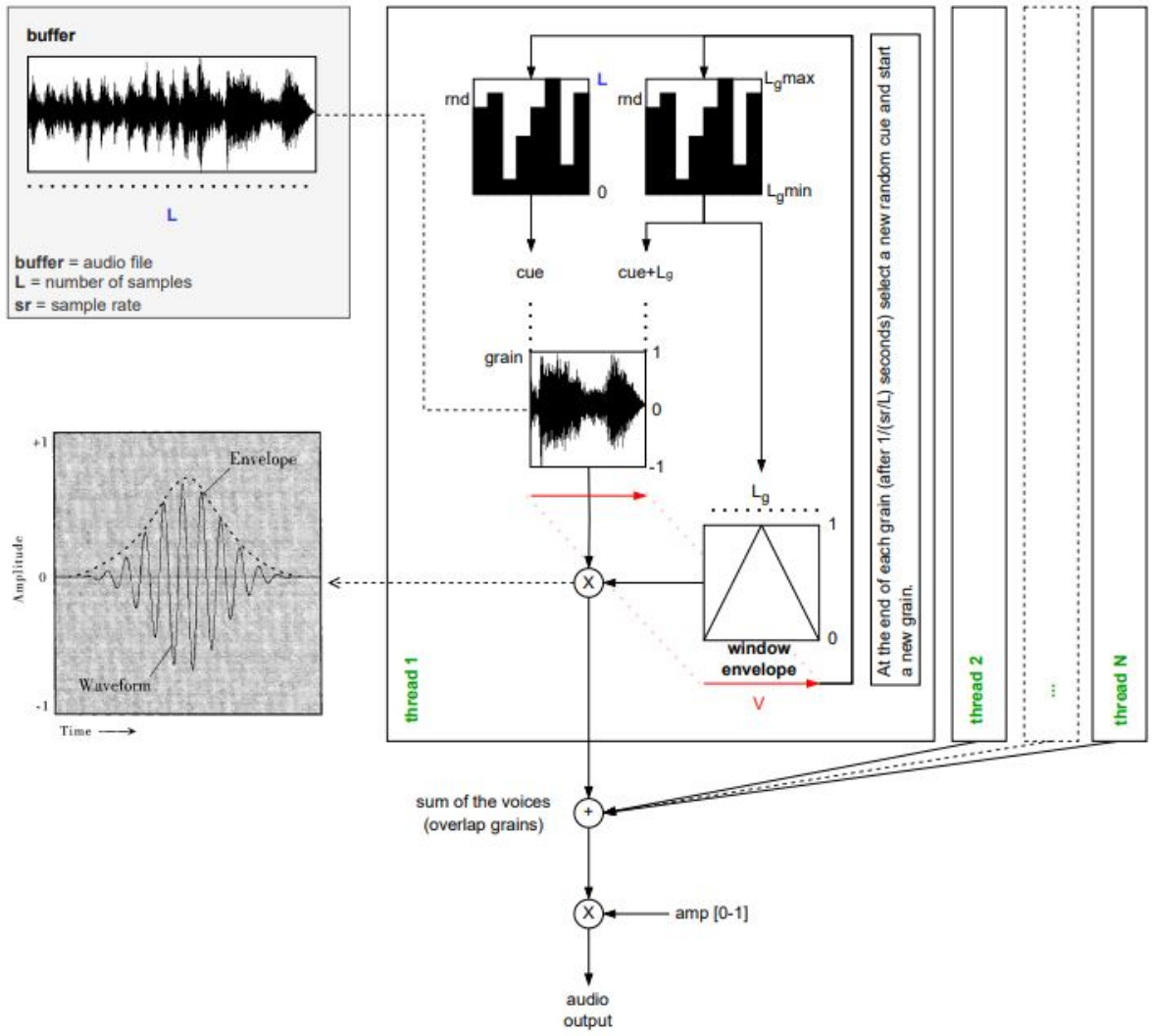


Figure 1: Granular Synthesis Implementation

1.2 Our implementation of Granular Synthesis

In order to fully understand how to interact with the application, it is needed a general and broad explanation of our approach to granular synthesis. The following Figure 1 illustrates the general structure of the algorithm.

As we can see, the synthesizer is composed of multiple threads (in our case, three), each with its own buffer. Each buffer contains an entire audio file, from which a collection of samples will be randomly extracted and processed during each iteration. This collection of samples is a structure that we commonly refer to as a *grain*. The randomization occurs in both the starting point and length of each grain. As depicted in the image, each time a new grain is extracted, the algorithm generates two random numbers indicating the starting point of the grain within the entire audio file and its length. After the grain is extracted, a window envelope of the same length as the grain is generated and applied to it. At the end of the iteration the grain will look like the one depicted in Figure 2.

Then, a new iteration begins, and another grain is extracted, continuing the process. After several iterations, the final output of each thread will resemble the example depicted in Figure

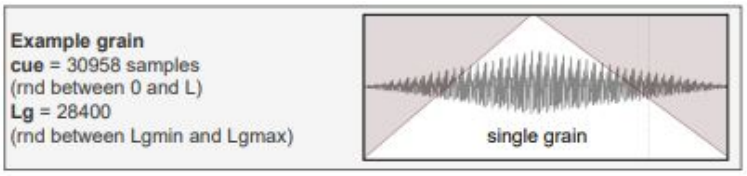


Figure 2: Grain Example

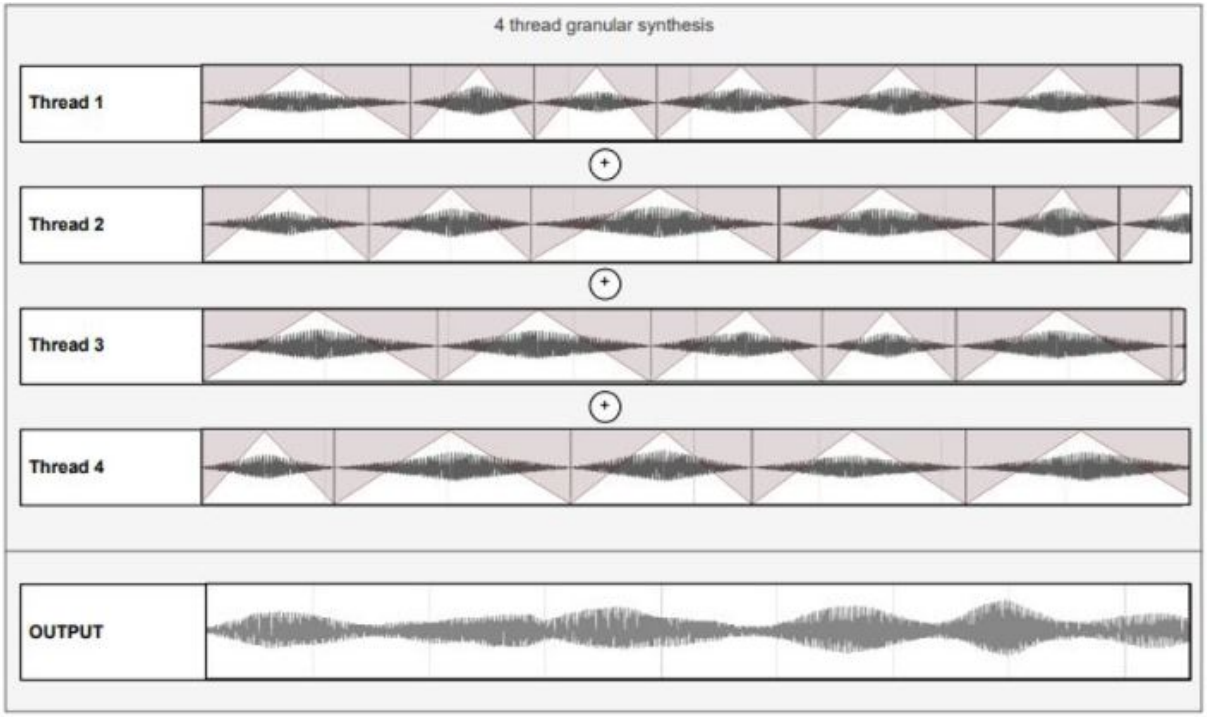


Figure 3: Final Output Example

3. The outputs of each thread will then be summed to represent the final output of the algorithm.

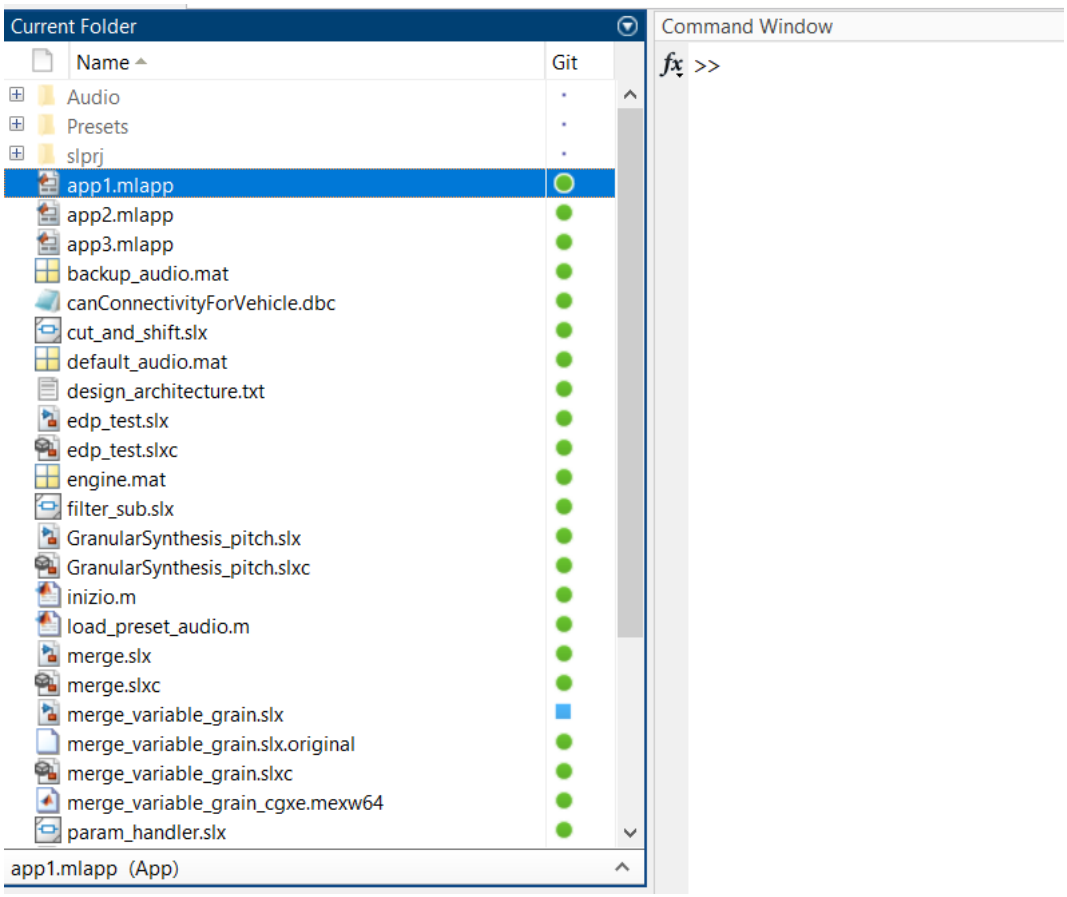
1.3 Granular Synthesis Algorithm Implementation

We now illustrate the structure of the synth, The synth is composed of 3 threads, each operating independently. Each thread has its own tunable parameters and operates starting from a different audio file provided as input by the user.

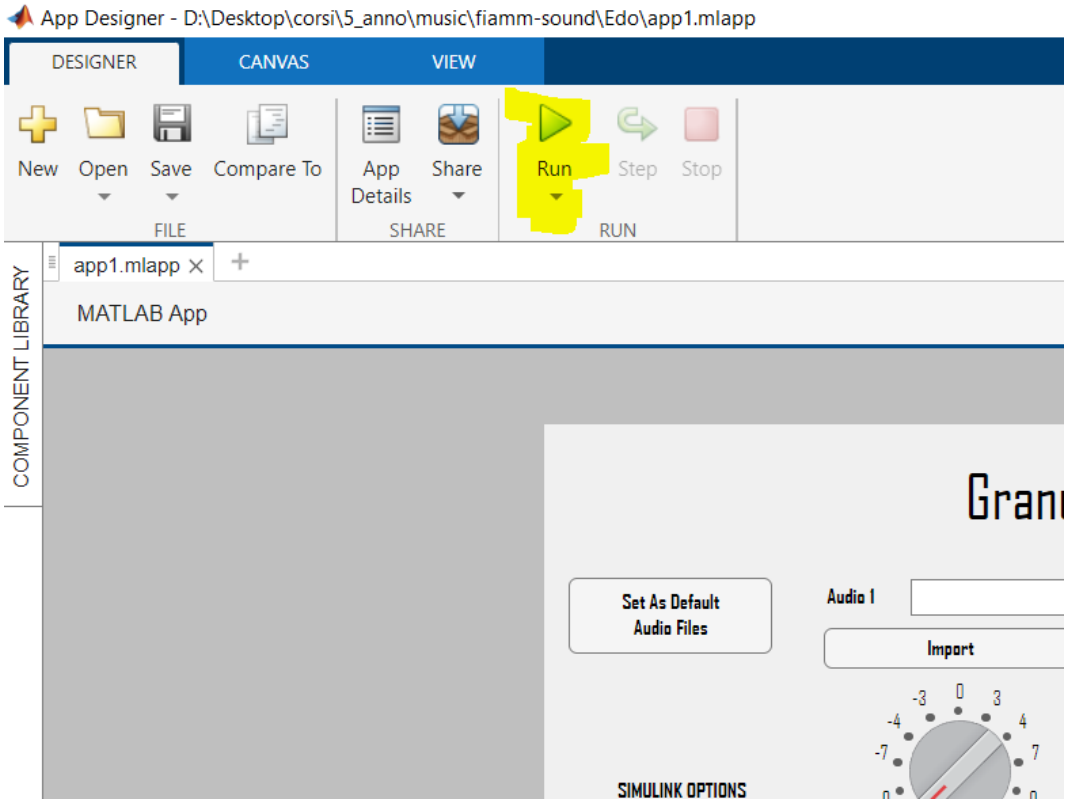
2 How to open the application

To access the application you need to follow the following steps:

1. Open Matlab and select the application folder path where all program files are located
2. From Matlab File Explorer double click on app1.mlapp and wait for the Matlab app designer to open

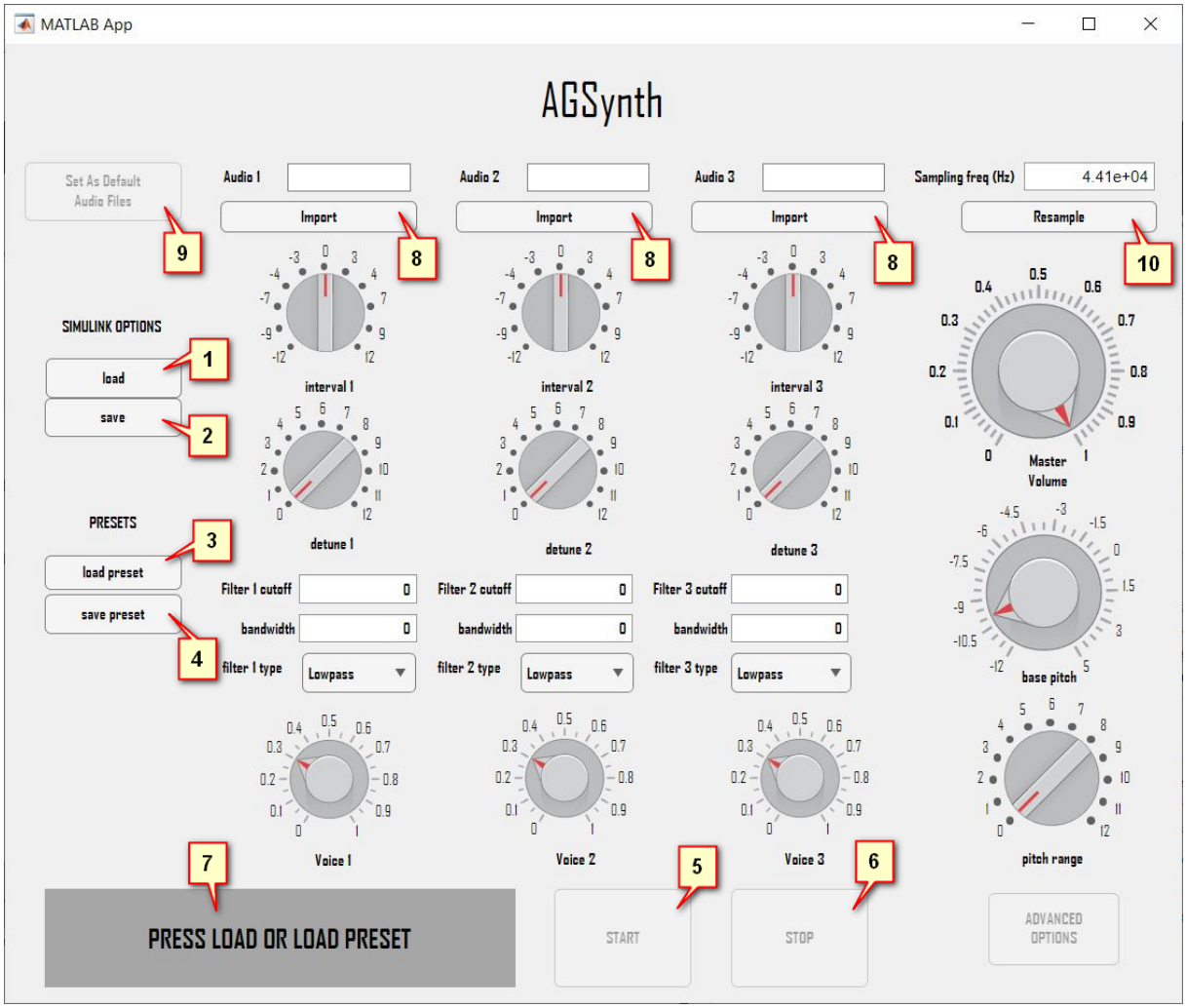


3. From Matlab app designer interface click on Run to open the app



4. Close the app before running a new app instance again. DO NOT USE TWO INSTANCES OF THE APPLICATION AT THE SAME TIME.

3 Basic commands



In this section we will explain the basic commands needed to understand the proper workflow.

1. Load button: it loads the Simulink model and runs the initial scripts. It may require several seconds on first usage of the app. After loading we will have available on our interface the current Simulink parameters.

2. Save button: it saves the simulink model and its parameters (equivalent of CTRL+S in Simulink).

3. Load Preset button: it allows the user to select a preset from the Preset folder and it loads all the parameters into the Simulink model

4. Save Preset button: it allows the user to save the current state of the app into a preset file in the Preset folder. The presets are saved in a numerical file whit .m extension. These files are not standalone and can only be imported using the application.

5. Start button: It initiates the simulation of the model. It may require some time during the first usage as the compilation of the model is time consuming. IMPORTANT: This

button is only available after the model has been correctly loaded.

6. Stop button: it stops the simulation
7. Audio text box: lets the user check the current audio file loaded for the specific thread.
8. Import button: lets the user select the desired audio file from the Audio folder.
   IMPORTANT NOTES ON IMPORTING AUDIO:
   - In order for the algorithm to work we always need 3 audio files imported
   - While not necessary, it's advised to use audio files that have a similar or identical sampling frequency. This helps to minimize the occurrence of artifacts and unwanted noise in the final audio.
   - ALL AUDIO SAMPLES MUST BE IN THE "Audio" FOLDER. You can find the folder in the project path.
9. Set as default audio files button: Allows the user to designate the current set of audio files as the default set. This default set will be automatically loaded when pressing the Load button.
10. Resampling button: Allows the user to resample all the audio files to the same sampling frequency. We recall that by default audio samples are always resampled at the frequency specified in the sampling frequency box. Sampling frequency must be specified in Hz, default value is 44100 Hz.

Next up, we want to give a possible example of workflow:

1. Click on "Load" or "Load Preset" before changing the parameters
2. Modify the parameters as desired
3. Click "Start" and wait for compilation to finish. Always remember to check if the volume of both the synth and your device is high enough!
4. Modify the parameters during live simulation and play with the synth!
5. Click "Stop"
6. Click "Save Preset" if you are satisfied by the resulting sound

NOTE ON EXECUTION AND COMPILING TIME: load and start operation may require some time to be executed especially if it's the first time running the simulation. Furthermore simulation performances are highly influenced by computational power: so, boost your settings if the simulation is not performing good.

Finally we want to point out that this app is still a demo and some small issues may occur during the usage. In that case please write a detailed email to the developers so they can fix the problem as fast as possible!

3.1 Common error

If the app is unable to load the model after pressing the load button, usually this is caused by the default audio files. To solve the problem import manually the audio files and click on "Set as Default Audio Files", the error should disappear from now on.

3.2 Audio drivers

Selected audio drivers are not accessible from the GUI due to Simulink limitations. In order to change the audio drivers (and their settings) you need to open the model named **merge_variable_grain.slx** and double click the block with the speaker image (see figure 4)

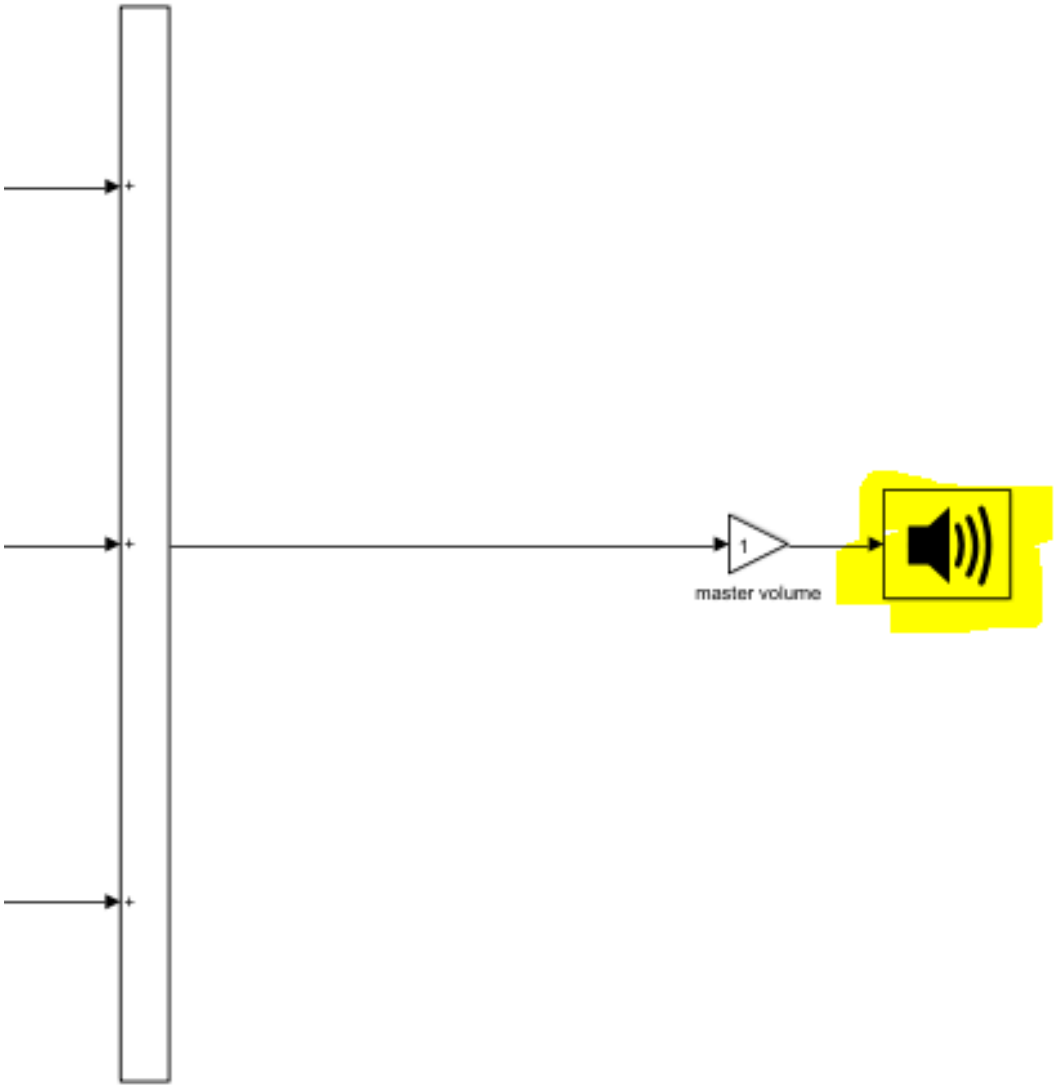
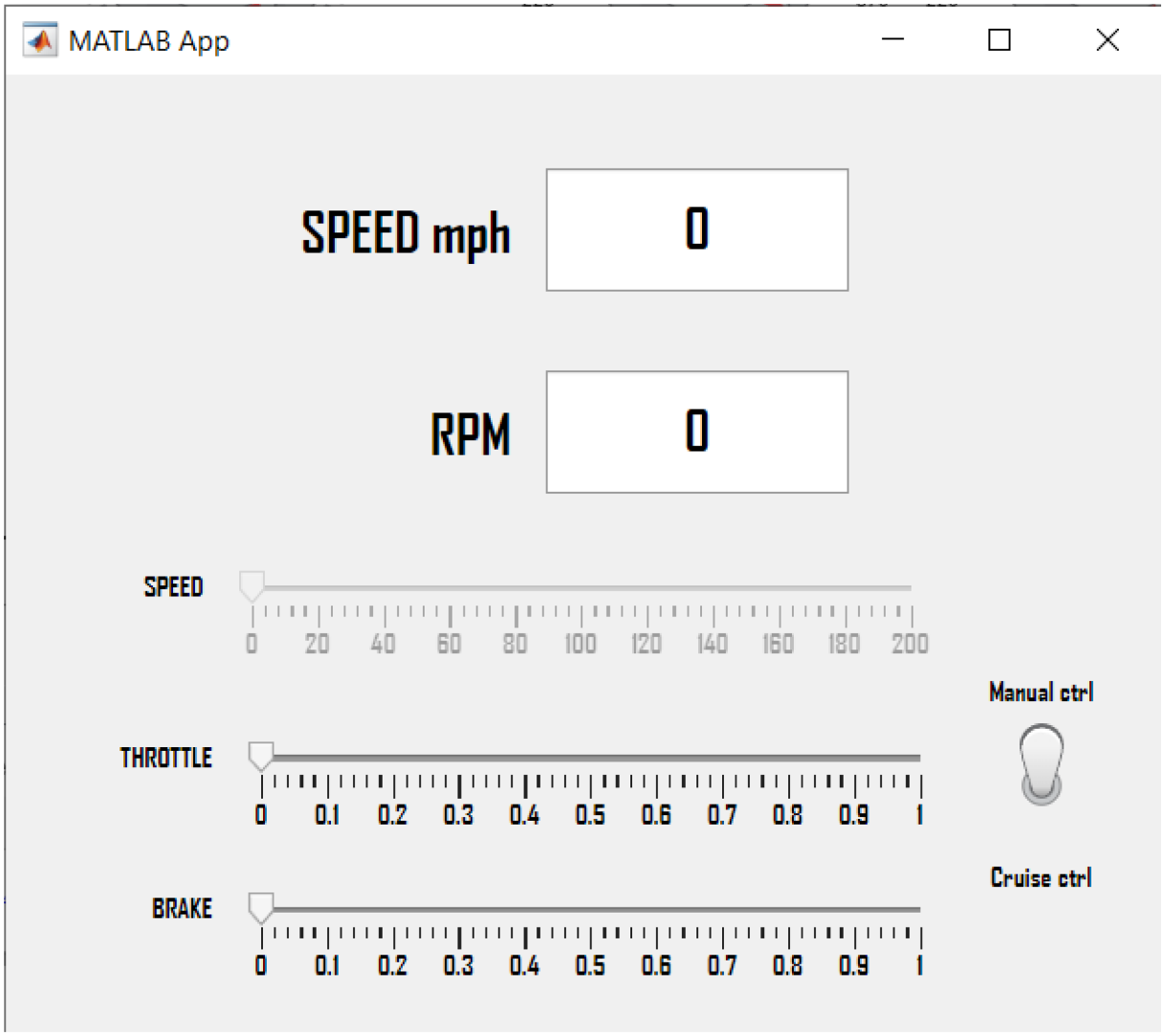


Figure 4: Audio device block

4 Control Panel



The control panel is not always accessible, it pops up when starting the simulation and closes when it stops. So, don't panic if you don't immediately see it.

This synthetizer is designed to be directly controlled by the imaginary driver, so the control is over the throttle/brake or the speed. Let's start describing the interface:

1. Switch Control Type: allows you to choose between Cruise Control mode or Manual Control:
   - in Cruise Control mode you have control over the desired speed you want the vehicle to reach
   - in Manual Control mode you have control over the virtual pedals. You can even press the pedals together
2. Speed Slider: active only in Cruise Control mode, allows you to choose the desired speed.
3. Throttle and Brake Sliders: active only in Manual Control mode, allow you to press the imaginary pedals, a value of 1 means that the pedal is completely pressed.

4. Speed Display: shows the current car speed in mph

5. RPM Display: shows the current engine RPMs

5 Granular Synthesis

This section is dedicated to explaining the various tunable parameters for using the granular synthesizer. If you haven't already, please refer to Chapter 1 to understand how our implementation of granular synthesis works. This understanding is essential for fully exploiting the functionalities of the algorithm and comprehending the function of each knob.

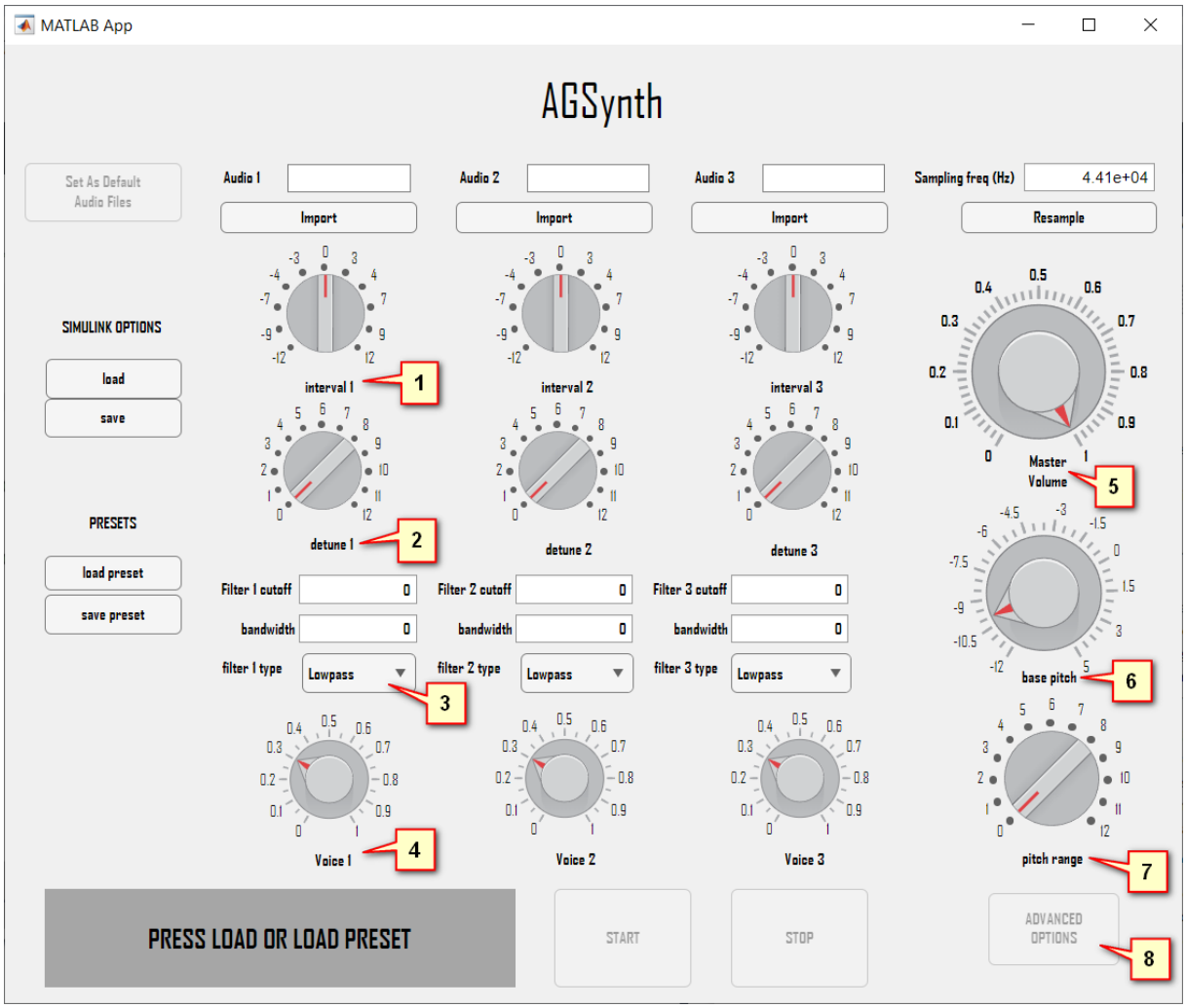


Figure 5: Threads and General Parameters

As evident from the numbers 1, 2, 3, and 4 in Figure 5, the control knobs for each thread are organized in columns. Let's begin by explaining the function of each knob:

1. Interval: enables you to adjust the pitch of the input audio file for each thread. The intervals are measured in semitones and span two octaves, ranging from one below to one above the original pitch of the sound.

2. Detune: allows you to select the amount of dynamic pitch shifting in semitones for each thread. The detuning reaches its maximum value when the car is at its highest RPM. For example, if Detune 1 is set to 7 semitones, Thread 1 will maintain its original pitch (selected by Interval 1) when the car is at its lowest RPM. As the car accelerates to reach its maximum speed and the engine reaches its maximum RPM, the pitch will then shift upward by $+7$ semitones.

3. Filter: these 3 blocks allow to control the parameters of the filters.

4. Voice Volume: this knob represents the volume of each thread.

We now explain the control knobs on the right, represented in Figure 5 by the numbers 5, 6, and 7. These parameters regulate the general settings of all the threads. In other words, by adjusting these parameters, it is possible to alter the settings of every thread simultaneously:

1. Master Volume: allows you to set the gain of all the threads at the final stage.

2. Base Pitch: simultaneously regulates the initial pitch of all threads, functioning similarly to the Interval knobs but applying to all threads at once. It's important to note that the effect of this parameter is combined with that of Interval. When both are used, the final pitch shift in semitones for each thread will be the sum of the two parameters.

3. Pitch Range: simultaneously regulates the dynamic pitch shift of all threads, functioning similarly to the Detune knobs but applying to all threads at once. Like with Base Pitch and Interval, the effect of this parameter is combined with that of Detune, so the amount of pitch shift reached at the highest rpm and speed will be the sum of the two parameters (for each thread).

4. Advanced Options: by pushing this button a new screen will appear, allowing to adjust some additional parameters that we will now present.

In the Advanced Options tab, you can find all the parameters related to the Granular Synthesizer, as well as a control for pitch dynamics. It's important to note that these parameters are closely linked to the behavior of the car, its speed, and the engine's rpm:

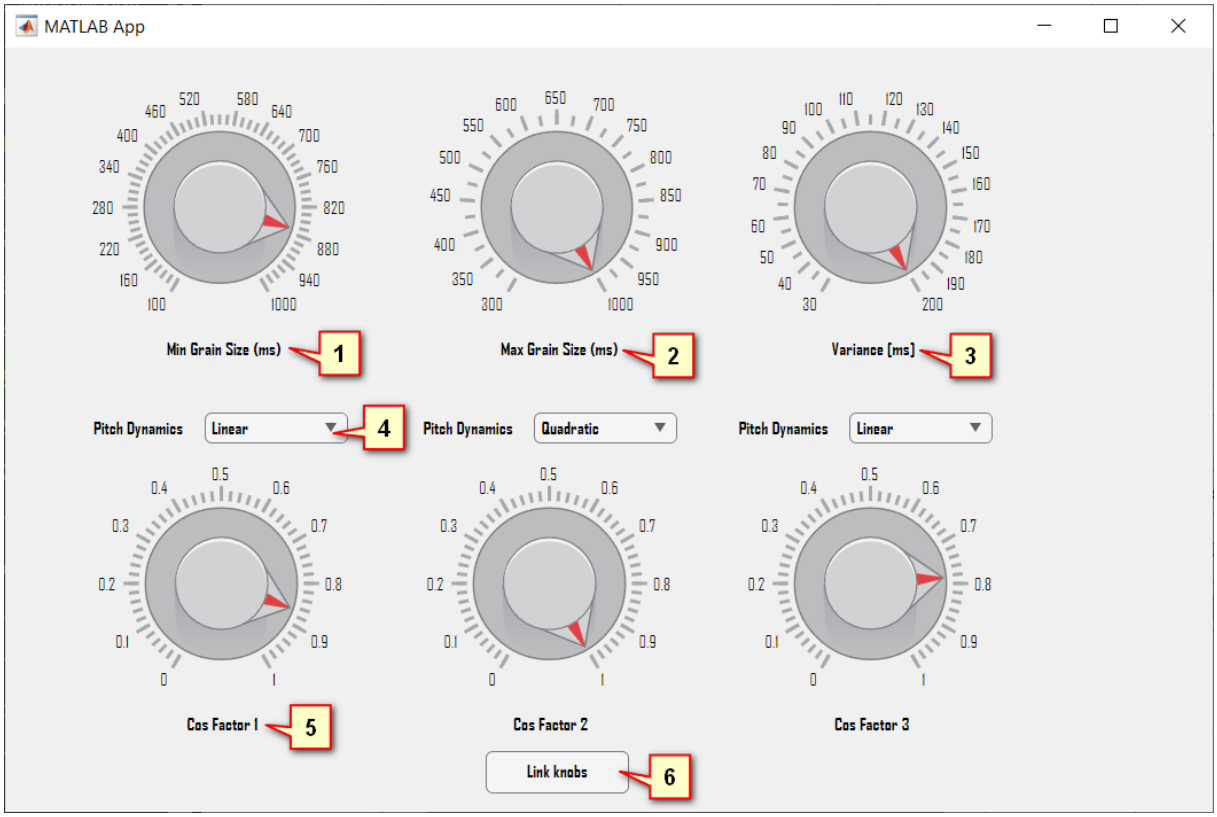


Figure 6: Advanced Options

1. Min Grain Size: represents the average size of the grain when the car reaches its maximum speed (and highest rpm).

2. Max Grain Size: Represents the average size of the grain when the car is stopped, i.e., when the engine is at its lowest rpm.

   In the range between the maximum reachable speed and the lowest speed, an average grain value is computed. In other words, as the car accelerates, the average grain size changes smoothly from the maximum (selected using knob 2) value to the minimum value (knob 1). The opposite occurs when the car brakes to come to a full stop; the grain size varies from the minimum value to the maximum value.

3. Variance: represents the variation in the average grain size at each iteration. As explained earlier, to ensure the algorithm's effectiveness, a degree of randomization is necessary. This knob allows the user to select the maximum difference between the average duration of the grain and the duration randomly extracted by the algorithm. Ultimately, the final duration of the grain will fall within the range of Average Grain Size $\pm$ Variance.

   *Note that the Average Grain Size has Value between Max Grain Size and Min Grain Size.

4. Pitch Dynamics: this setting controls the exponent of the pitch dynamic. Selecting 'Linear' provides the default result, maintaining a steady pitch progression and reaching the final pitch set in Pitch Range. Choosing 'Quadratic' and 'Cubic' allows the pitch to

rise faster and reach higher levels, resulting in a more pronounced pitch variation over time.

5. Cosine Factor: allows you to adjust the attack of each grain. As explained in Section 1.2, a windowing process is applied to each grain. In our algorithm, we use the cosine window, represented in Figure 7. Varying the Cosine Factor α affects the attack duration of the window, ranging from a rectangular window ($\alpha = 0$) to a Hann window ($\alpha = 1$). Each thread has its own independent Cosine Factor.

6. Link knobs: by pressing this button, the user can link the Cosine Factor of threads 2 and 3 to the Cosine Factor of thread 1. This allows the user to adjust the attacks of each thread simultaneously using the first knob on the left.

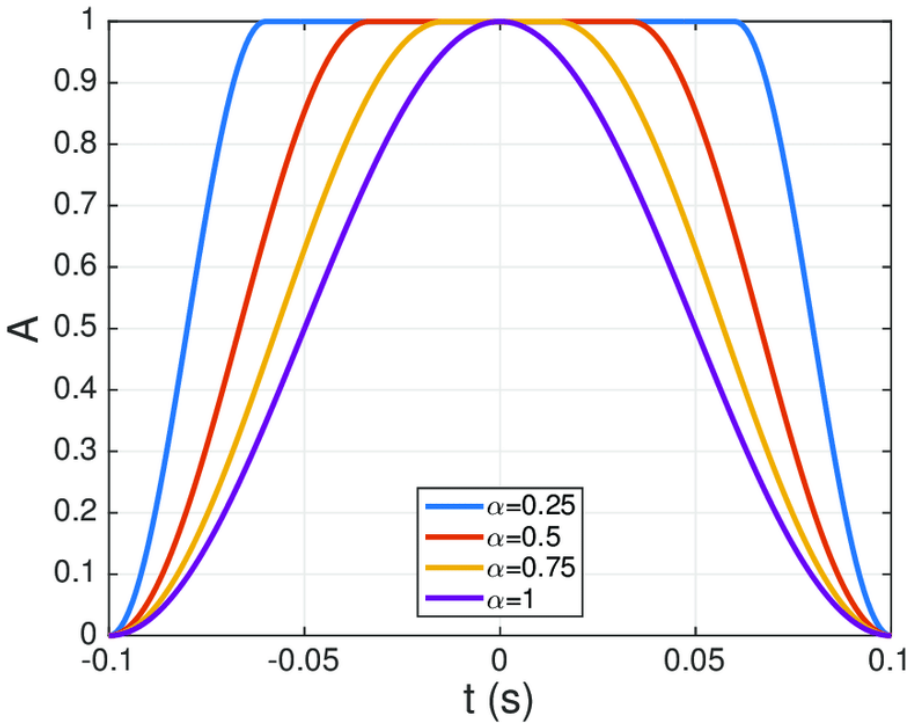


Figure 7: Cosine Window

PRACTICAL NOTE: It's possible that when using a high grain length (around one second), the device may struggle to keep up and compute and apply the window within the clock time, especially if using a Cosine Factor of 1. In such cases, the output could be laggy. Make sure to allocate the necessary computational resources.