Experiment 1

Signal Generation using Sinusoids and Wave-shaping

The objective of this experiment is to get some insight about frequency domain representation of discrete periodic signals. You will use your knowledge of Fourier series to construct synthetic sound waveforms.

Prelab

P1) Review of Fourier series

In order to design a system or study the behavior of a system to different stimuli, the system should first be modeled. According to the models adopted for the systems, the systems are categorized into linear and nonlinear. Linear systems can be modeled with linear operators such as multiplication by a constant, integration and differentiation. The linear systems are easier to analyze and superposition holds for them.

In the study of linear systems, if the stimulus signal is a sinusoid, the output of the system would also be a sinusoid with the same frequency. The only parameters that would change are the amplitude and the phase. Since superposition holds for linear systems, sum of a number of sinusoids would also be a sum of sinusoids with the same frequencies. It is thus very easy to analyze linear systems if the input or stimulus is a sum of sinusoids.

Now if the input to a system is, say a sawtooth waveform, is it a sum of sinusoids and how could one know what the output of the system would be? It is proved that almost any periodic signal $x(t)$ can be written as a sum of sinusoids as:

$$x(t)=\sum_{n=0}^{\infty} a_n \cos(2\pi n\omega t)+b_n \sin(2\pi n\omega t) \qquad \textbf{(*)}$$

where, $T=1/2\pi\omega$ is the fundamental period of $x(t)$ and:

$$a_n=\frac{2}{T}\int_T x(t)\cos(2\pi n\omega t)\,dt,\ b_n=\frac{2}{T}\int_T x(t)\sin(2\pi n\omega t)\,dt$$

Equation (*) can also be written as:

$$x(t)=\sum_{n=0}^{\infty} a_n \cos(2\pi n\omega t)+b_n \sin(2\pi n\omega t)=\sum_{n=0}^{\infty}\sqrt{a_n^2+b_n^2}\cos\left(2\pi n\omega t+\tan^{-1}\left(\frac{b_n}{a_n}\right)\right)$$

$$=\sum_{n=0}^{\infty} c_n \cos(2\pi n\omega t+\phi_n)$$

This representation of $x(t)$, greatly simplifies the analysis of the system.

P2) Generating Synthetic Sounds

In order to see the power of Fourier series, it is interesting to see how it can be used in generating synthetic sounds. The simplest form of sound that can be generated is a simple sinusoid waveform with a frequency in the hearing range of human’s ear. Our ears can hear the sounds with frequency in the range 20Hz to 20KHz. Thus, if a sinusoid waveform with a frequency in the hearing range and with sufficiently large amplitude is present in the air, our ears will "hear" it as a sound.

Basic sound synthesis could be accomplished through construction of a sinusoid from its samples. More complicated periodic waveforms can be generated as in previous section by adding some harmonics. The sounds from musical instruments however, are much more complicated to generate. The main problem is that the waveforms change over time. For instance, when natural sounds start, they build up from zero amplitude to a certain level in a time lapse, i.e. they don't suddenly start at their full amplitude. It also takes some time till they fade out, i.e. they don't suddenly stop.

Though their waveforms are not periodic, yet some periodicity can be observed in them. In fact one can approximate the sounds from many musical instruments by a periodic waveform that is multiplied in an envelope as in Fig. P1. Many sounds like the sound of piano can be synthetically generated. One of the simple envelope models is the ADSR model which stands for Attack, Decay, Sustain and Release (See Fig. P2). The Attack is the first portion of the sound, from when it starts to build up to the point it reaches its peak. Decay is the part of the envelope where the signal drops from its peak value to an almost constant value which would be held during Sustain. The Release is the final part where the sound fades out.

As an example when you press a key on the piano, the sound will hit in (Attack), then fall fast (Decay) to a rather constant tone (Sustain) and when you take your finger off, the sound fades away (Release).

In this experiment you will use Fourier series to generate some waveforms and then you will shape them to sound more realistically using the ADSR envelope.

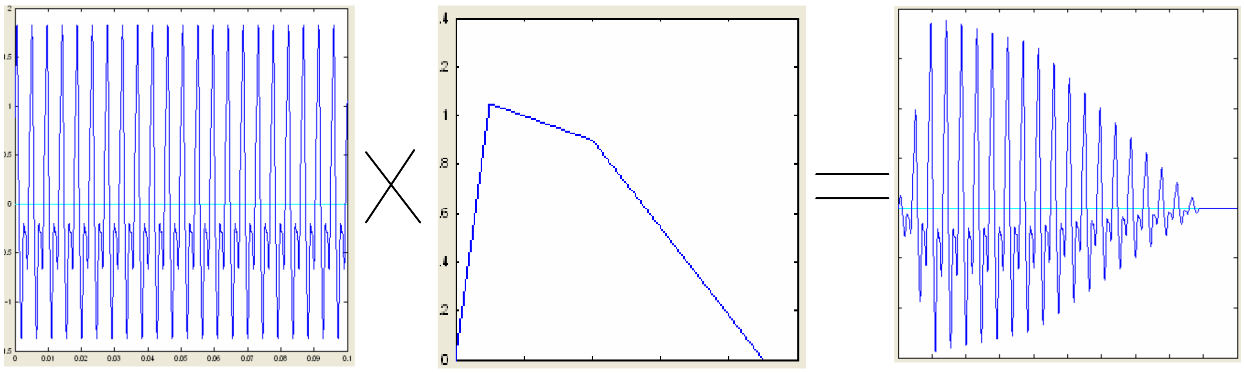


Figure P1 – Multiplying the synthetic sound with an envelope to generate more realistic sound

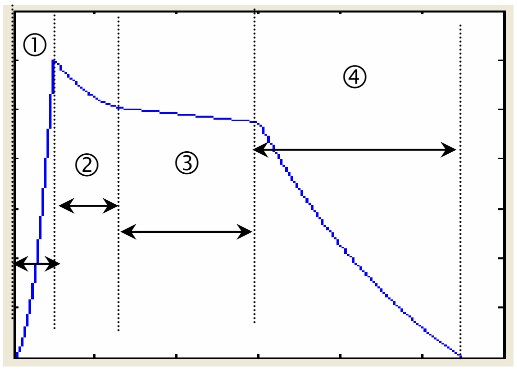
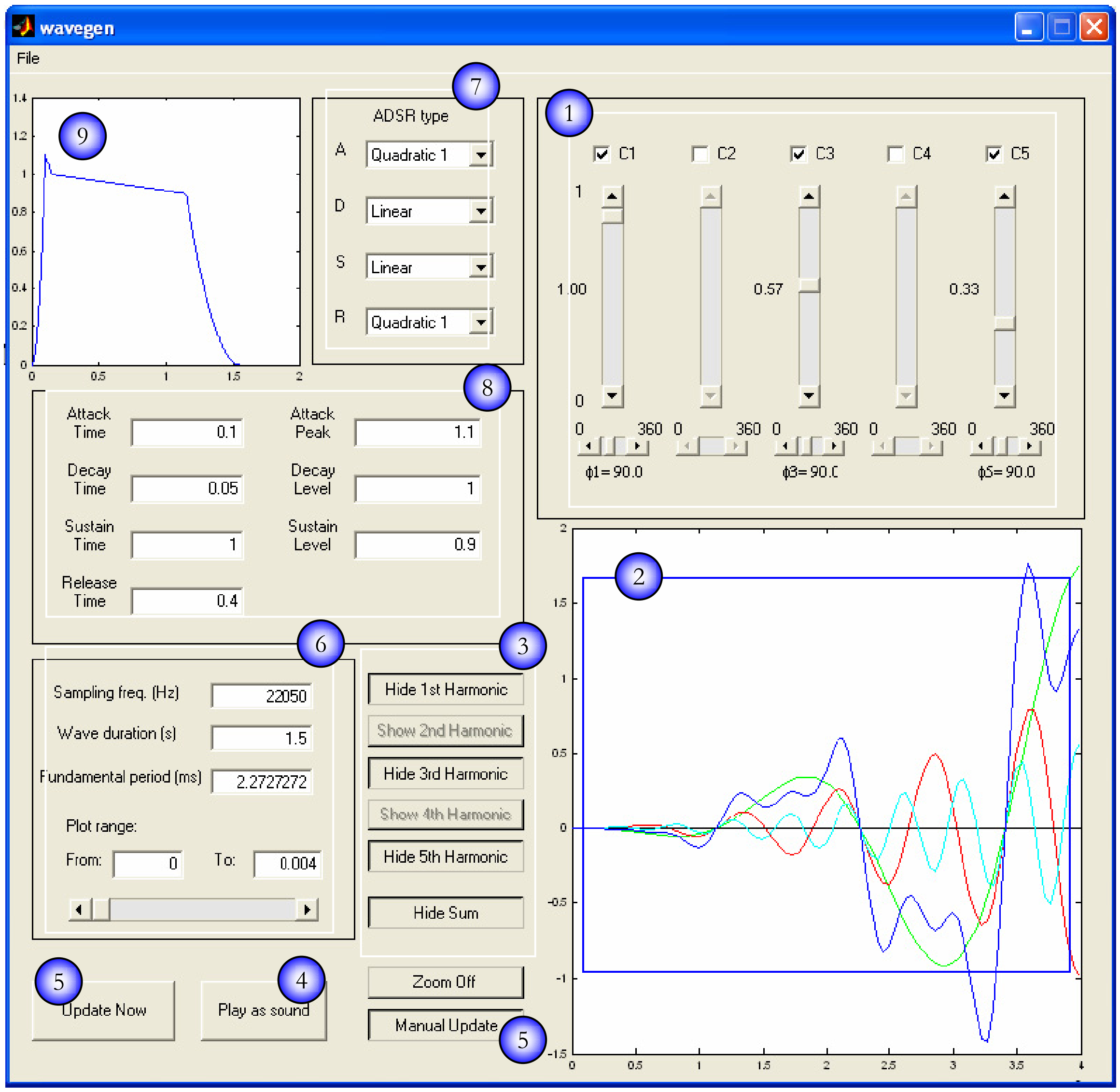


Figure P2 – The ADSR envelope: 1) Attack, 2) Decay, 3) Sustain and 4) Release

1. The wavegen wave generator



ADSR type
A Quadratic 1
D Linear
S Linear
R Quadratic 1

Attack Time	0.1	Attack Peak	1.1
Decay Time	0.05	Decay Level	1
Sustain Time	1	Sustain Level	0.9
Release Time	0.4		

Sampling freq. (Hz)	22050
Wave duration (s)	1.5
Fundamental period (ms)	2.2727272

Plot range:
From: 0 To: 0.004

☑ C1 ☐ C2 ☑ C3 ☐ C4 ☑ C5

1.00 0.57 0.33

ϕ1= 90.0 ϕ3= 90.0 ϕ5= 90.0

Figure 1 – A screen-shot of wavegen: 1. Fourier coefficients panel, 2. Generated waveform axes, 3. View selector panel, 4. Play button, 5. Plot update buttons, 6. Waveform and view range settings panel, 7. ADSR type panel, 8. ADSR settings panel, 9. ADSR envelope axes.

The wavegen GUI, would let you interactively construct the waveforms of your choice. In this experiment you will learn how to work with the program and generate the waveforms that you wish. You would also write some portions of the code, wavegen uses to generate the sound and envelope.

In the following section you would learn about the different panels and controls of this tool. Fig. 1 shows the various controls of the program.

1) Fourier Coefficients Panel

As you saw in the introduction, a periodic signal $x(t)$ can be represented in terms of a sum of sinusoids:

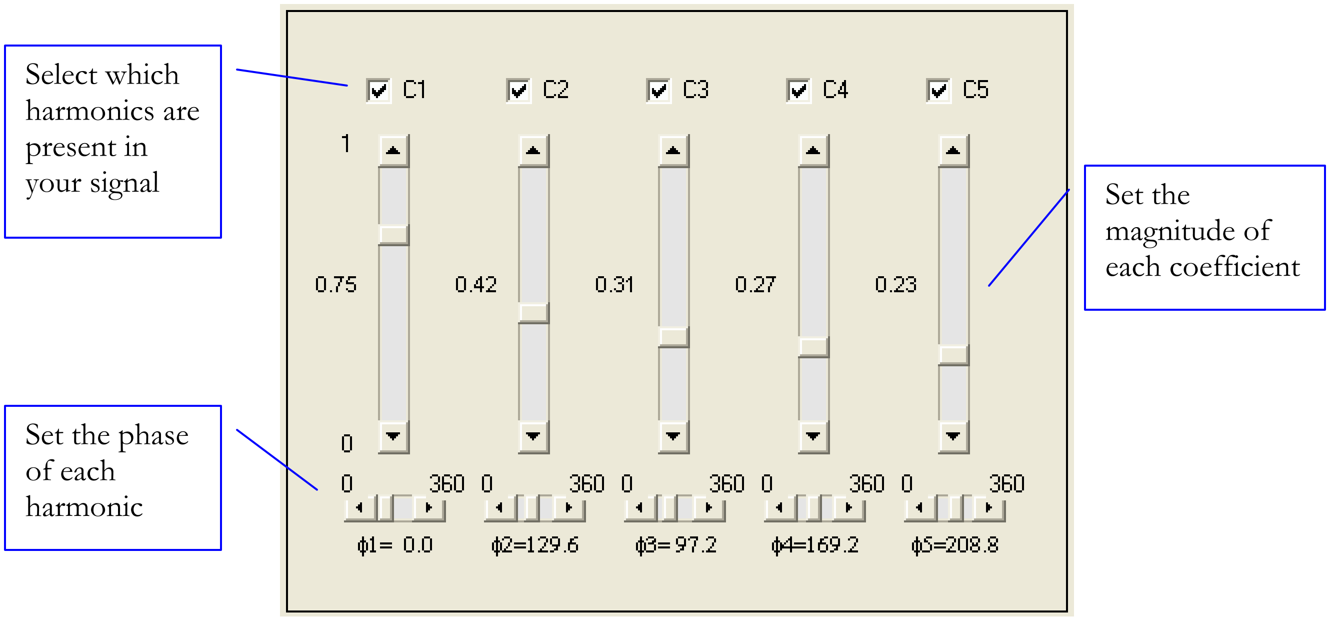
$$x(t) = \sum_{n=0}^{\infty} c_n \cos(2\pi n\omega t + \phi_n) \tag{1}$$

Thus $x(t)$ can be expressed in terms of $\{c_n\}_{n=0}^{\infty}$, $\{\phi_n\}_{n=0}^{\infty}$ and ω.

The coeffiecients c_n decrease with n for all practical signals $x(t)$, so the series in equation (1) could be approximated by a few terms with greater contribution, i.e. bigger coefficients. Many signals could be approximated using only the first 5 harmonics. In fact in many applications like in electronics and power systems, the approximation with the first 5 harmonics gives predicts the behavior of the system with a very good degree of precision.

In this panel you may select which of the first 5 harmonics are present in your signal by checking the box beside each coefficient from c_1 to c_5. The sliders under the checkboxes set the normalized amplitude of each harmonic ($|c_n|$), and the phases (ϕ_n) in degrees for each harmonic. (Fig. 2)

Note: The amplitude values are in the range [0, 1], so you should normalize your coefficient values by dividing all by the magnitude of the largest coefficient.


Select which harmonics are present in your signal
Set the magnitude of each coefficient
Set the phase of each harmonic
1
0
0
360

☑ C1 ☑ C2 ☑ C3 ☑ C4 ☑ C5

0.75 0.42 0.31 0.27 0.23

ϕ1= 0.0 ϕ2=129.6 ϕ3= 97.2 ϕ4=169.2 ϕ5=208.8

Figure 2 – Coefficients panel

2) Generated Waveform Axes

This is where the generated waveform is plotted. The controls for what to be plotted and other settings for the plots are described in the subsequent sections. The sum signal is plotted in blue, and the five harmonics are plotted in green, magenta, red, black, and cyan.

3) View Selector Panel

This panel consists of six buttons. The first five buttons set which harmonics should be viewed in the plot. The sixth button controls the appearance of the waveform resulting from summing up the five harmonics, i.e. the generated waveform. The caption of each button signifies what action would be taken if each button is clicked on. The buttons would be disabled if their associated harmonic is not present in the signal.

4) Play Button

Pressing this button you could listen to the sound of the waveform you have generated. The sound will be played for the duration specified in the waveform and view range settings panel.

Warning: Do not the play button again when the sound is being played, this may cause MATLAB crash.

5) Plot Update Buttons

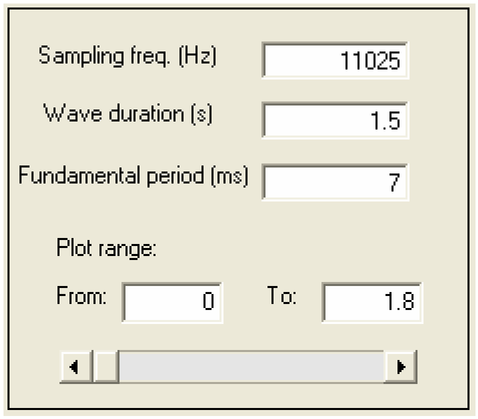
The update now button causes the plots to be regenerated with current settings. The real-time update button causes the plots be updated whenever you change a setting.

Note: When in real update mode make sure the plot range is small since large ranges would slow down the system due to excessive processing load.

6) Waveform and View Range Settings Panel

In this panel you can set the basic parameters of the generated waveform including: fundamental period, wave duration, and sampling frequency. Note that the fundamental period should be given in milliseconds.

The plot range settings at the bottom of the panel, set the start and end position of the window on the whole waveform to be plotted. For instance you may want to generate a waveform with duration of 3 seconds, having a fundamental frequency of 420Hz. Viewing the plot of the whole waveform, you would see a solid plot. To see the details of the plot you could set the plot range to show you only 6ms of the waveform, say from 0 to 0.06. To move around and see different parts of the plot, you could either enter the new starting and ending points, or use the slider to move the window to the position you like.



Sampling freq. (Hz) 11025
Wave duration (s) 1.5
Fundamental period (ms) 7
Plot range:
From: 0 To: 1.8

Figure 3 – Waveform and view range settings panel

7) ADSR Type

In this panel you can set the type of each of attack, decay, sustain and release sections of the envelope. The four types are as follows:

a) Linear: the shape is a line passing through the starting and ending points
b) Quadratic 1: This is a quadratic function curving upwards
c) Quadratic 2: This is a quadratic function curving downwards
d) Exponential: This is an exponential function

The different settings can be seen in Fig. 4. Other combinations are possible by selecting different types for each part of the ADSR envelope.

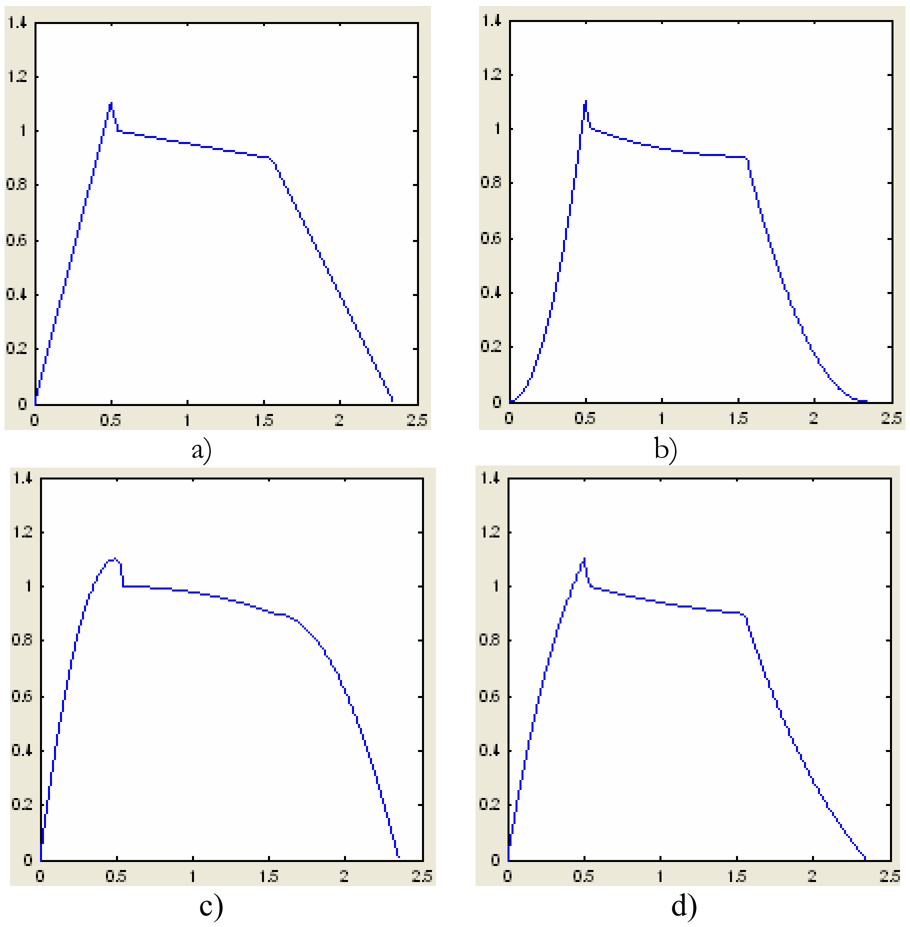


Figure 4 – Different types of ADSR: a) linear, b) quadratic 1, c) quadratic 2, and d) exponential

8) ADSR Settings Panel

You can set different settings of the ADSR envelope in this panel. These settings are shown in Fig. 5. If you want to remove any of the A, D, S or R from the envelope, simply enter a very small time for its duration.

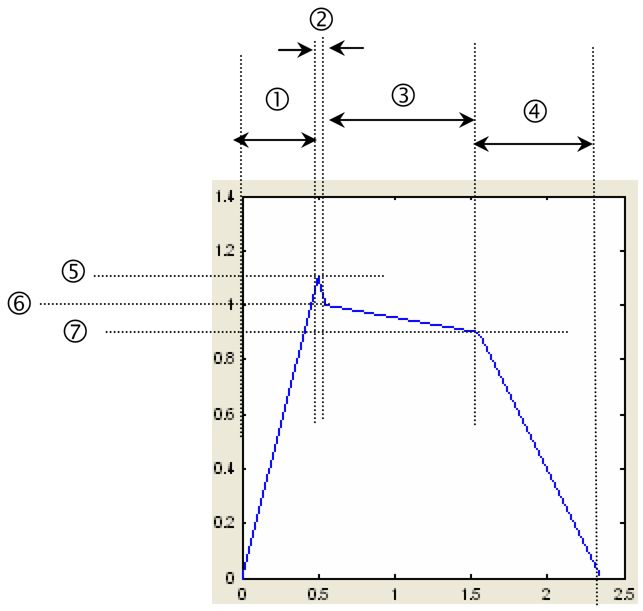


Figure 5 – ADSR envelope parameters: 1) Attack time, 2) Decay time, 3) Sustain time, 4) Release time, 5) Attack peak, 6) Decay level, 7) Sustain level

9) ADSR Envelope Axes

This is where the ADSR envelope shape is plotted.

2. Generating Simple Waveforms

Generating A Sinusoid Waveform

In this section you will generate a number of simple waveforms using `wavegen` and write some functions in MATLAB needed to accomplish this task. The ear of human is sensitive to sounds with frequencies between 20Hz – 20KHz. As a first step let's begin with a simple sinusoid waveform. Write a function named `gensin` as follows:

Listing 1. Function gensin

```
function [wave, t] = gensin(amplitude, frequency, phase, window_start, window_end, f_s)
%--------------------------------------------------------------------------
%
%    Function to generate a sinusoid waveform
%
%     Output:
%
%       wave: The vector containing the time samples of the generated sinusoid
%             waveform
%       t   : The vector containg the sampled time points
%
%     Input:
%
%       amplitude: The amplitude of sinusoid, i.e. A in  A cos (wt + phi)
%       frequency: In Hz, is the frequency of the sinusoid
%       phase    : In "Degrees", Specifies the phase of the sinusoid at t = 0,
%                  i.e. phi in  A cos (wt + phi)
%    window_start: Specifies the point in time from which the samples
%                  should start to generate
%      window_end: Specifies the end of the interval in which the samples
%                  are generated
%             f_s: The sampling frequency in Hz
%
%--------------------------------------------------------------------------
```

This function would generate both the points of time at which the sinusoid has been sampled and the samples at those points. The output could thus be readily plotted with a `plot` command. `wavegen` uses this function as a basic tool to generate the waveforms that are combination of sinusoids.

1. Launch `wavegen` by typing its name at MATLAB command prompt.
2. Check the C1 box in the Fourier coefficients panel.
3. Set the fundamental period so that the signal would have a frequency of 440Hz, this would generate an A tone.
4. Set the wave duration to 2 seconds.
5. Set the sampling frequency to 22050 Hz.
6. Press the show 1st harmonic button and update the plot by either of the update buttons.
7. Play the sound of the sinusoid waveform you have generated.
8. Change the frequency of the sinusoid to 880 Hz and 220 Hz and listen to the sound, what is your deduction?

Generating Square and Triangular Waveforms

Now try to approximate a square wave by its first five harmonics.

1. Compute the first five Fourier series coefficients and phases, namely $c_1 - c_5$ and $\phi_1 - \phi_5$.

Note: Normalize the coefficients by dividing them with the value of their largest.

2. Check the C1 box in the Fourier coefficients panel and set the magnitude and phase of the 1st harmonic.

3. Set the wave duration, sampling frequency and fundamental period as in steps 3-5 from last section.
4. Turn real-time update on.
5. Turn show 1st harmonic on.
6. Turn show sum on.
7. Play the sound.
8. One by one enable the other coefficients starting from C2, set their corresponding harmonic magnitude and phase.
9. Note the change in the shape of the sum as you enable more harmonics. Listen to the sound after adding each harmonic to the sum and note the difference in sound.

Repeat these steps for a triangular waveform.

3. Shaping the Envelope

As seen in the pre-lab, in order to construct more realistic sounds one may use an envelope shaping function. In this section you would write two functions which let you create many different types of envelopes. These functions would then be called by `wavegen` to let you see the effect of envelope shaping on the sounds you generated before.

As mentioned before, the ADSR envelope consists of four pieces: Attack, Decay, Sustain and Release. So here you will write a program that will generate one piece at a time, your second program will put the pieces together and make the complete envelope.

Listing 2. Function piecegen

```
function segment = piecegen(type, y1, y2, dur, f_s)

%----------------------------------------------------------------------
%
%    Function to generate a one piece of the ADSR envelope
%
%     Output:
%
%         segment: The vector containing the time samples of the generated piece
%
%     Input:
%
%            type: The type of the curve:
%                  2: linear
%                  3: Quadratic 1
%                  4: Quadratic 2
%                  5: Exponential
%              y1: Amplitude at the beginning of the piece
%              y2: Amplitude at the end of the piece
%             dur: Duration of the piece
%             f_s: The sampling frequency in Hz
%
%----------------------------------------------------------------------
```

In order to generate each piece, you need to find its equation according to its type. The equations would be of the following forms: a) linear: $y = at + b$, b) quadratic: $y = at^2 + bt + c$, c) exponential (rising): $y = a(1 - e^{-t/duration}) + b$, d) exponential (falling): $y = ae^{-t/duration} + b$. Use `switch/case` to implement the different cases. Note that for quadratic types the quadratic is at its extremum at one end, i.e. its derivative is zero. Also note that one end of the piece is at $t = 0$. As an example the code for the exponentials is shown here:

For a falling exponential: $a = \frac{y_1 - y_2}{1 - e^{-1}}$ and $b = y_1 - \frac{y_1 - y_2}{1 - e^{-1}}$, thus

```
case 5           % Exponential
        if (y1 > y2)              %falling
            segment = y1 - (y1 - y2)/(1 - exp(-1)) + (y1 - y2)*exp(-t/dur)/(1 - exp(-1));
        else                      %rising
            segment =                        % <- Fill this blank!
        End
```

In order to test the code you have written you plot the segment you have generated at MATLAB command prompt or you may continue with the assembling function which puts the segments together to make the complete envelope:

Listing 3. Function ADSRgen

```
function [ADSR, t] = ADSRgen(Atype, Dtype, Stype, Rtype, Atime, Dtime, Stime, Rtime,
Apeak, Dlevel, Slevel, f_s)

%---------------------------------------------------------------------------
%
%    Function to generate the ADSR envelope
%
%     Output:
%
%            ADSR: The vector containing the time samples of the generated
%                  envelope
%               t: The vector contating time points at which the samples
%                  are taken
%
%     Input:
%
%           Atype: The type of Attack
%           Dtype: The type of Decay
%           Stype: The type of Sustain
%           Rtype: The type of Release
%           Atime: The duration of Attack (in seconds)
%           Dtime: The duration of Decay (in seconds)
%           Stime: The duration of Sustain (in seconds)
%           Rtime: The duration of Release (in seconds)
%           Apeak: Attack peak
%          Dlevel: Decay level
%          Slevel: Sustain level
%             f_s: The sampling frequency in Hz
%
%---------------------------------------------------------------------------

A = piecegen(Atype, 0, Apeak, Atime, f_s);
D =                          % <- Fill this blank!
S =                          % <- Fill this blank!
R =                          % <- Fill this blank!

ADSR = [A, D, S, R];
```

This function is used by wavegen to perform the envelope shaping.

1. Run wavegen now and select a linear Attack type, note that the rest of D, S, and R will also be set to linear. See if your codes work correctly.
2. Select the other types for Attack and Release. Since one is ascending and the other is descending and both are long enough you may see if there is any problem with your code.
3. If there is any problem don't close wavegen, simply modify you code in MATLAB editor and reselect one of the ADSR type settings to see if the code is corrected.
4. After making sure that your code works fine, regenerate the rectangular wave as in the previous section.
5. Select linear type for all A, D, S, and R.
6. Set the durations for A, D, S and R to : 0.05, 0.05, 0.5, 0.4 respectively.
7. Set wave duration to 1s

8. Play the sound with envelope. Note the difference.
9. If you have time see the effect of changing the duration and other settings of the ADSR envelope. If you made a sound you liked take a note of its settings.

4. Approximating a Piano Tone

In this section you will try to approximate a piano tone by setting the appropriate parameters in `wavegen`. As the first step you should read the file containing the sound and plot the waveform to extract its parameters.

The tone is stored in .wav format which is one of the most popular file formats for storing sounds (it was probably *the* most popular before MP3 came into being!).

1. Type at MATLAB command prompt: `[tone, FS, NBITS] = wavread('tone.wav')` , this would read the file and put the samples of the sound in the variable `tone`. The sampling frequency is returned in `FS`. (You may learn more about this command by typing `help wavread`).
2. Listen to the sound using MATLAB `soundsc` command a couple of times.
3. Plot the waveform of the sound. Note the resemblance of the envelope to the simple ADSR model.
4. Zoom a part of the plot, where the amplitude is almost constant. Zoom enough to see only 2 or 3 cycles of the signal.
5. Estimate the fundamental period of the sound from the plot.
6. Run wavegen and set the fundamental period.
7. Set the amplitude and phase of the harmonics as follows:

$$c = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{bmatrix} = \begin{bmatrix} 1 \\ 0.76 \\ 0.07 \\ 0.04 \\ 0.15 \end{bmatrix}, \quad \phi = \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \\ \phi_5 \end{bmatrix} \simeq \begin{bmatrix} 340 \\ 276 \\ 130 \\ 155 \\ 105 \end{bmatrix}$$

(If you are curious to know how these values are computed, you may use MATLAB `fft` command to compute the Discrete Fourier Transform of the portion with constant amplitude that you cut out.)

The waveform would look smoother than the original because the higher order harmonics are discarded.

8. Try to set the ADSR parameters so that the envelope would look similar to the envelope of the piano tone.
9. Check to see how close your synthetic sound matches the original.