Mini Project #2: Octave-spaced FIR filter banks

Mr. Dan Jacobellis and Prof. Brian Evans

Version 1.2. Assigned on Monday, October 27, 2025.
Due on Monday, November 3, 2025, by 11:59 pm via Gradescope submission

Late submission is subject to a penalty of two points per minute late.

Reading: McClellan, Schafer and Yoder, *Signal Processing First*, 2003, Chapters 5-7. Errata.
Companion Web site with demos and other supplemental information: http://dspfirst.gatech.edu/
Web site contains solutions to selected homework problems from *DSP First*.

E-mail Mr. Dan Jacobellis (TA) at danjacobellis@utexas.edu. Please consider posting questions on Ed Discussion, which can be answered by anyone in the class. You can post anonymously.

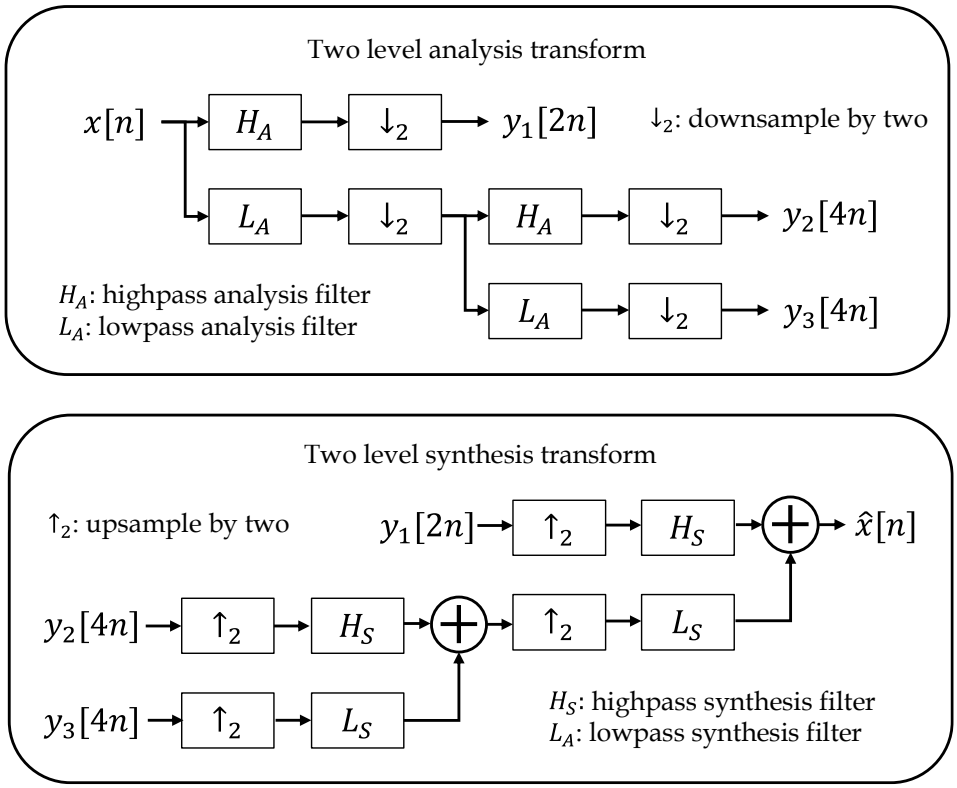
Lecture and office hours for Mr. Jacobellis and Prof. Evans follow. Prof. Evans also holds office hours in person in EER 6.882 and online on Zoom.

Time Slot	*Monday*	*Tuesday*	*Wednesday*	*Thursday*	*Friday*
11:00 am		**Evans (EER 1.516)**		**Evans (EER 1.516)**	
11:30 am		**Evans (EER 1.516)**		**Evans (EER 1.516)**	
12:00 pm		**Evans (EER 1.516)**		**Evans (EER 1.516)**	
12:30 pm		**Jacobellis (EER 1.810)**			
1:00 pm		**Jacobellis (EER 1.810)**			
1:30 pm		**Jacobellis (EER 1.810)**			
2:00 pm	**Evans (EER 6.882)**	**Jacobellis (EER 1.810)**			
2:30 pm	**Evans (EER 6.882)**				
3:00 pm	**Evans (EER 6.882)**			**Jacobellis (EER 7.702)**	
3:30 pm			**Evans (EER 6.882)**	**Jacobellis (EER 7.702)**	
4:00 pm			**Evans (EER 6.882)**	**Jacobellis (EER 7.702)**	
4:30 pm			**Evans (EER 6.882)**	**Jacobellis (EER 7.702)**	
5:00 pm				**Jacobellis (EER 7.702)**	**Jacobellis (EER 1.810)**
5:30 pm					**Jacobellis (EER 1.810)**
6:00 pm					**Jacobellis (EER 1.810)**

1.0 Introduction

An octave is a musical interval corresponding to a doubling of frequency. In the western musical scale, each octave is divided into twelve notes, with each note's frequency being $2^{1/12}$ higher than the previous note. See Mini-project #1 from Fall 2024 for more information.

In this project, you will design and implement a discrete-time analysis filter bank that separates an audio signal into octaves, and a discrete-time synthesis filter bank that recovers the signal from its components. The filter bank consists of a cascade of high-pass and low-pass filters combined with downsampling and upsampling operations. For an overview of downsampling and upsampling, see **Appendix A** and the mini-project #1 hints. An example using two levels is shown below. Please see the MATLAB code to implement the single and two level transforms `one_level.m` and `two_level_octave.m` in Appendices C and D.



In the two level analysis transform, $y_1[2n]$ corresponds to the octave containing frequencies $\frac{f_s}{4} < f < \frac{f_s}{2}$, $y_2[4n]$ corresponds to the octave containing frequencies $\frac{f_s}{8} < f < \frac{f_s}{4}$, and $y_3[4n]$ corresponds to $f < \frac{f_s}{8}$. More levels can be applied by recursively applying the process to the bottommost branch of the analysis transform, allowing the signal to be separated into a greater number of octaves.

When the filters satisfy certain properties, it is possible to achieve perfect reconstruciton, meaning that $\hat{x}[n] = x[n]$; this process is also known as a discrete wavelet transform. For additional reading, see Lecture 9: Multirate, Polyphase, and Wavelet Filter Banks from Stanford University Center for Computer Research in Music and Acoustics (CCRMA).

2.0 Complex sinusoidal response of FIR filters cascaded with downsampling and upsampling

1. Consider the system consisting of an FIR filter with frequency response $H_1(e^{j\hat{\omega}})$ followed by downsampling by a factor of two:

$$x[n] \longrightarrow \boxed{H_1(e^{j\hat{\omega}})} \longrightarrow \boxed{\downarrow_2} \longrightarrow y_1[2n]$$

Let $x[n] = e^{j\hat{\omega}n}$. Show that $y_1[2n] = H_1(e^{j\hat{\omega}})e^{j(2\hat{\omega})n}$.

2. Consider the cascade of upsampling by a factor of two followed by an FIR filter with frequency response $H_2(e^{j\hat{\omega}})$:

$$x[n] \longrightarrow \boxed{\uparrow_2} \longrightarrow \boxed{H_2(e^{j\hat{\omega}})} \longrightarrow y_1[n/2]$$

Let $x[n] = e^{j\hat{\omega}n}$. Show that $y_2[n/2] = \frac{1}{2}H_2(e^{j\hat{\omega}/2})e^{j\left(\frac{\hat{\omega}}{2}\right)n} + \frac{1}{2}H_2(e^{j(\hat{\omega}/2-\pi)})e^{j\left(\frac{\hat{\omega}}{2}-\pi\right)n}$

Hint: Setting odd-indexed samples of a discrete signal equal to zero is equivalent to multiplying by $\cos^2\left(\frac{\pi n}{2}\right) = \frac{1}{2} + \frac{1}{2}\cos(\pi n) = \{1,0,1,0,\dots\}$. Therefore, upsampling a discrete signal $x[n]$ by a factor of two is the same as evaluating $x[m]$ at $m = n/2$ and multiplying by $\cos^2\left(\frac{\pi n}{2}\right)$:

$$x[n/2] = \uparrow_2 \{x[n]\} = \begin{cases} x[m]|_{m=n/2} & n \text{ even} \\ 0 & n \text{ odd} \end{cases} = \cos^2\left(\frac{\pi n}{2}\right) x[m]|_{m=n/2}$$

3. Consider the cascade of the systems in part (2) and part (3).

$$x[n] \longrightarrow \boxed{H_1(e^{j\hat{\omega}})} \longrightarrow \boxed{\downarrow_2} \longrightarrow \boxed{\uparrow_2} \longrightarrow \boxed{H_2(e^{j\hat{\omega}})} \longrightarrow \hat{x}[n]$$

Let $x[n] = e^{j\hat{\omega}n}$. $\hat{x}[n]$ will contain multiple frequencies. Show that, for the output component at the original frequency $\hat{\omega}$, the effective frequency response is

$$\mathrm{H}_{\mathrm{eff}}(e^{j\hat{\omega}}) = \frac{1}{2}H_1(e^{j\hat{\omega}})H_2(e^{j\hat{\omega}})$$

4. Consider the following cascade:

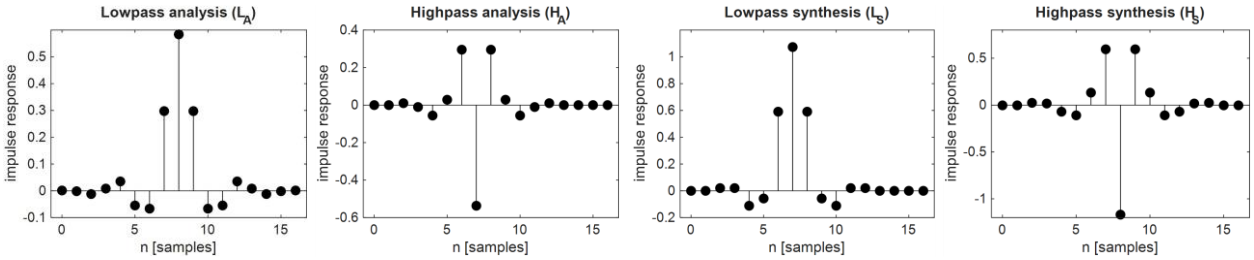
$$x[n] \longrightarrow \boxed{H_1(e^{j\hat{\omega}})} \longrightarrow \boxed{\downarrow_2} \longrightarrow \boxed{H_2(e^{j\hat{\omega}})} \longrightarrow \boxed{\downarrow_2} \longrightarrow \boxed{\uparrow_2} \longrightarrow \boxed{H_3(e^{j\hat{\omega}})} \longrightarrow \boxed{\uparrow_2} \longrightarrow \boxed{H_4(e^{j\hat{\omega}})} \longrightarrow \hat{x}[n]$$

Let $x[n] = e^{j\hat{\omega}n}$. $\hat{x}[n]$ will contain multiple frequencies. Show that, for the output component at the original frequency $\hat{\omega}$, the effective frequency response is

$$H_{\mathrm{eff}}(e^{j\hat{\omega}}) = \frac{1}{4}H_1(e^{j\hat{\omega}})H_2(e^{2j\hat{\omega}})H_3(e^{j2\hat{\omega}})H_4(e^{j\hat{\omega}})$$

3.0 Effective frequency response of individual branches.

Four FIR filters are used as building blocks in `two_level_octave.m`: a lowpass analysis filter (L_A), highpass analysis filter (H_A), lowpass synthesis filter (L_S), and highpass synthesis filter (H_S). The impulse responses are provided in the MATLAB code and plotted below.



1. In MATLAB, use `freqz` to calculate the frequency responses for each of the four filters.
2. Plot the magnitude responses for L_A and H_A as a function of $\hat{\omega}$ from $0 \le \hat{\omega} < \pi$. Put both magnitude responses on the same graph.
3. Using the results from 2.3 and 2.4, calculate the effective frequency responses for each of the three branches in the filter bank. Plot the magnitude of each as a function of $\hat{\omega}$ from $0 \le \hat{\omega} < \pi$. Put all three magnitude responses on the same graph .
    a. Upper octave branch: $(H_A \circ \downarrow_2 \circ \uparrow_2 \circ H_S)$

$$H_1(e^{j\omega}) = \frac{1}{2} H_{HA}(e^{j\hat{\omega}}) H_{HS}(e^{j\hat{\omega}})$$

    b. Middle octave branch: $(L_A \circ \downarrow_2 \circ H_A \circ \downarrow_2 \circ \uparrow_2 \circ H_S \circ \uparrow_2 \circ L_S)$

$$H_2(e^{j\hat{\omega}}) = \frac{1}{4} H_{LA}(e^{j\hat{\omega}}) H_{HA}(e^{2j\hat{\omega}}) H_{HS}(e^{j2\hat{\omega}}) H_{LS}(e^{j\hat{\omega}})$$

    c. Low frequency branch: $(L_A \circ \downarrow_2 \circ L_A \circ \downarrow_2 \circ \uparrow_2 \circ L_S \circ \uparrow_2 \circ L_S)$

$$H_3(e^{j\hat{\omega}}) = \frac{1}{4} H_{LA}(e^{j\hat{\omega}}) H_{LA}(e^{2j\hat{\omega}}) H_{LS}(e^{j2\hat{\omega}}) H_{LS}(e^{j\hat{\omega}})$$

4.0 Extending to more octaves.

More levels of the transform can be applied by recursively decomposing the low frequency branch, thus dividing the signal into a greater number of octaves. The highest octave covers the range from $\frac{f_S}{4} < f < \frac{f_S}{2}$. The next highest octave covers the range from $\frac{f_S}{8} < f < \frac{f_S}{4}$, and so on.

1. Assume a sampling rate of $f_S = 8134$ Hz so that the highest octave corresponds to the range 2033.5 Hz $< f <$ 4067 Hz (C7 to B7 on the western scale). Extend the implementation in `two_level_octave.m` to four levels so that middle C (261.62 Hz) is included in an octave-spaced branch instead of the low frequency branch.
2. Resample the violin-C4 recording to 8134 Hz, then apply the new four-level filter bank. Ensure that the output from the synthesis transform approximately matches the input.
3. Reconstruct each of the five branches independently. Create a spectrogram for each branch:
    a. "4-line" octave branch (C7 to B7)
    b. "3-line" octave branch (C6 to B6)
    c. "2-line" octave branch (C5 to B5)
    d. "1-line" octave branch (C4 to B4)
    e. Low frequency branch (0 Hz to B3)

5.0 Playback and use as compression system

1. Using MATLAB, load an audio signal of your choice whose length is at least 16 seconds. Resample the audio signal to 8134 Hz. If the audio file is stereo (two channel) convert it to mono (single channel) by averaging the two channels together. Truncate the signal to exactly 130144 samples (16 seconds).
2. Using the filter bank from part 4.0, divide the signal into 5 bands (low frequency + four octaves). How many samples are needed to represent each band at the output of the analysis transform prior to upsampling? Compute the compression ratio of each band assuming that the same precision is used for storage.[1]
3. Reconstruct each of the five bands independently and play them back using soundsc. Describe the sound of each band compared to the original signal.
4. Compute the PSNR[2] of each band compared to the original signal. Plot the PSNR vs the compression ratio from part 5.2. Which output band provides the best trade-off between PSNR and compression ratio?

[1] The compression ratio for the ith band is $\mathrm{CR}_i = \frac{\text{Number of output samples for } i\text{th band}}{\text{Number of samples in the original audio signal}}$

[2] For a pair of signals in the range [-1,1] with length L, the PSNR is:

$$\begin{aligned}\mathrm{PSNR}(x[n], y[n]) &= 20\log_{10}\left(\frac{(I_{\max} - I_{\min})^2}{\mathrm{MSE}(x[n], y[n])}\right) \\ &= 20\log_{10} 2 - 10\log_{10}\big(\mathrm{MSE}(x[n], y[n])\big) \\ &= -10\log_{10}\big(\mathrm{MSE}(x[n], y[n])\big) + 6.02 \text{ dB}\end{aligned}$$

where

$$\mathrm{MSE}(x[n], y[n]) = \frac{1}{L}\sum_{0}^{L}(x[n] - y[n])^2$$

In MATLAB:

```
PSNR = -10*log10(mean(abs(x - y).^2)) + 6.02
```

asd

Appendix A: Downsampling and Upsampling

One way to modify a continuous-time signal $x(t)$ is by scaling the time axis, (e.g. $x(2t)$ or $x\left(\frac{t}{2}\right)$). Downsampling and upsampling are the discrete-time versions of scaling the time axis.

When a discrete-time signal $x[n]$ is downsampled by an integer factor of M (denoted $\downarrow_M$), samples are discarded following a regular pattern, making the signal M times shorter.

$$\downarrow_M \{x[n]\} = x[Mn]$$

This can be performed in MATLAB using `downsample(x,M)`.

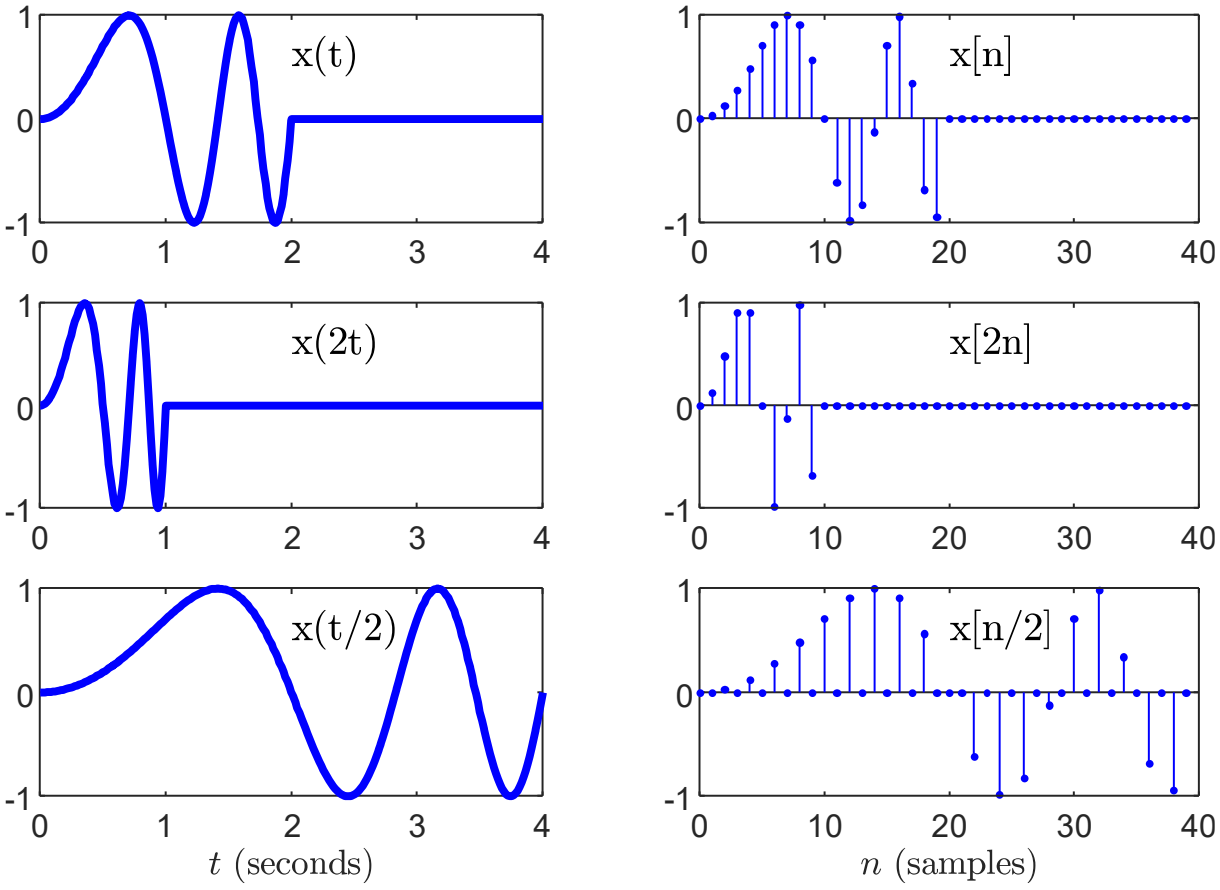
Upsampling a signal by an integer factor L (denoted $\uparrow_L$) makes the signal L times longer by inserting zeros following a regular pattern.

$$\uparrow_L \{x[n]\} = \begin{cases} x[n/L] & \text{if } n/L \text{ is an integer} \\ 0 & \text{otherwise} \end{cases}$$

If we assume the convention that a discrete-time signal is zero-valued for non-integer values of n, then we can simply write $\uparrow_L \{x[n]\} = x[n/L]$. This cane be performed in MATLAB using `upsample(x,L)`.

Upsampling and downsampling are linear operations, but not time-invariant.

Example: Consider a continuous-time signal $x(t) = \sin(\pi t^2)\, \mathbf{1}_{[0,2]}(t)$. Sampling $x(t)$ at a rate $f_s = 10$ Hz produces $x[n] = \sin\left(\frac{\pi}{100} n^2\right) \mathbf{1}_{[0,2]}(n)$. The plots of $\downarrow_2 \{x[n]\} = x[2n]$ and $\uparrow_2 \{x[n]\} = x[n/2]$ are shown.



Appendix B: Homework and Mini-Project Guidelines

Here are some things you should follow for all assignments.

Amount of work to show:

1. An explanation should be given for every single answer. Answers written without explanation will lose two-thirds of the points allotted for that part.
2. Only "standard" formulas (like Euler's formula, trigonometric formulas, etc.) can be used without a reference. If you're using something non-standard, then please put a reference to the formula number in the book, or whatever source you got it from. Just using the final result of a similar problem done in the class, and omitting the intermediate steps, is not okay. You have to show your work.
3. There shouldn't be big jumps in logic from one step to the next.
4. For everything, expect to show at least one intermediate step between the first line and the answer. Even if it seems unnecessary to you, please err on the side of caution. Things that seem obvious to you when you're writing the solution are not quite so obvious for someone reading it.
5. If you're in any doubt about how much work to show, please ask the instructor or the teaching assistant.

MATLAB source code guidelines:

1. Put a comment before the solution of each part, telling the question number of the solution.

2. If you're using complicated logic, leave a comment telling what that block of code is supposed to do.
3. Use variable names that related to their meaning/use.
4. Avoid using two different variables for the same thing.
5. Try to avoid using "magic numbers" in the code. If you're using a number, write a comment telling me how you derived it.
6. Make sure that your code will compile & run in a clean workspace; i.e., one without any variables present. Use a clear all; at least once before submitting it.
7. No marks will be deducted based on the efficiency of the code unless the problem asks you to write efficient code.

Technical points:

1. Merge all the files together into one PDF file.
2. Please adjust the contrast, exposure etc., to get a good scan quality so that the TA can easily read what you write. Take extra care to get a good scan for parts written in pencil.
3. For the MATLAB code you write for an assignment, please copy the code into Word or include a screenshot showing the code. Do not submit handwritten code.

Other things:

1. All plots must have axis labels, with units.
2. Final answers must be boxed, or underlined or otherwise differentiated from the rest of the solution.
3. All final answers must have units, if they exist.
4. Read the questions carefully.
5. Try to answer all parts of a question together. If the solution to some parts of a question is written elsewhere, then leave a note telling the reader where to find it.

Organization of a mini-project report:

Please write a self-contained narrative report. The audience is someone who has taken the equivalent of this class. The report should provide references to the textbook and other sources as needed. Please refer to the hints above, which apply to homework assignments and mini-project reports, as well as the following additional guidelines for the mini-project.

Here are example mini-project #1 reports written by the instructors:

- "FM Synthesis for Musical Instruments" (2018)
- "Sinusoidal Speech Synthesis" (2021)
- "Music Synthesis" (2023)

Please see the homework hints page for specific guidelines for this project.

Appendix C: Matlab code for one_level.m

```
% Code accompanying fall 2025 mini project #2 to apply one level of the DWT
% Originally written by Dan Jacobellis 10/24/2025

% The coefficients below for a dyadic perfect reconstruction filterbank
% are based on the Cohen-Daubechies-Feauveau wavelet (bior6.8)
% If the wavelet toolbox is installed, you can also use the following code:
% [LA, HA, LS, HS] = wfilters('bior6.8');
% LA = LA(2:end)/sqrt(2); HA(2:end) = HA/sqrt(2);
% LS= sqrt(2)*LS(2:end); HS = sqrt(2)*HS(2:end);
coeffs = [
       0.00134974786501001                         0                         0      -0.00269949573002003
      -0.00135360470301001                         0                         0      -0.00270720940602003
       -0.0120141966670801        0.0102009221870399        0.0204018443740798        0.0240283933341602
       0.00843901203981008       -0.0102300708193699        0.0204601416387398        0.0168780240796202
        0.0351664733065404       -0.0556648607799594        -0.111329721559919        -0.0703329466130807
       -0.0546333136825205        0.0285444717151497       -0.0570889434302994         -0.109266627365041
       -0.0665099006248407         0.295463938592917         0.590927877185834         0.133019801249681
         0.297547906345713        -0.536628801791565          1.07325760358313         0.595095812691426
         0.584015752240756         0.295463938592917         0.590927877185834         -1.16803150448151
         0.297547906345713        0.0285444717151497       -0.0570889434302994         0.595095812691426
       -0.0665099006248407       -0.0556648607799594        -0.111329721559919         0.133019801249681
       -0.0546333136825205       -0.0102300708193699        0.0204601416387398         -0.109266627365041
        0.0351664733065404        0.0102009221870399        0.0204018443740798        -0.0703329466130807
       0.00843901203981008                         0                         0        0.0168780240796202
       -0.0120141966670801                         0                         0        0.0240283933341602
      -0.00135360470301001                         0                         0      -0.00270720940602003
       0.00134974786501001                         0                         0      -0.00269949573002003
];
LA = coeffs(:,1); % Lowpass Analysis
HA = coeffs(:,2); % Highpass Analysis
LS = coeffs(:,3); % Lowpass Synthesis
HS = coeffs(:,4); % Highpass Synthesis

% Example audio file built into matlab.
audiodata = load('handel.mat');
x = audiodata.y(1:end-1); fs = audiodata.Fs;

% Analysis
L1 = downsample(conv(x,LA),2);
H1 = downsample(conv(x,HA),2);

% Synthesis
xrec = conv(upsample(L1,2),LS) + conv(upsample(H1,2),HS);

% Account for delay
single_filter_delay = (length(coeffs)-1)/2;
num_cascaded_filters = 2;
total_delay = num_cascaded_filters*single_filter_delay;
xrec = xrec(total_delay:end-total_delay-1);

% verify that xrec is the same as x (accounting for small numerical error)
assert( max(abs(x-xrec)) < 1e-10 )

% create spectrograms for original signal and components
figure; spectrogram(x,1024,0,1024,'yaxis',fs);
clim([-70,0]); title('Original signal')

low_component = conv(upsample(L1,2),LS);
figure; spectrogram(low_component,1024,0,1024,'yaxis',fs);
clim([-70,0]); title('Low frequency component');

high_component = conv(upsample(H1,2),HS);
figure; spectrogram(high_component,1024,0,1024,'yaxis',fs);
clim([-70,0]); title('High frequency component');
```

Appendix D: Matlab code for two_level_octave.m

```
% Code accompanying fall 2025 mini project #2 to apply two level DWT
% (discrete wavelet transform) and inverse
% Originally written by Dan Jacobellis 10/24/2025

% The coefficients below for a dyadic perfect reconstruction filterbank
% are based on the Cohen-Daubechies-Feauveau wavelet (bior6.8)
% If the wavelet toolbox is installed, you can also use the following code:
% [LA, HA, LS, HS] = wfilters('bior6.8');
% LA = LA(2:end)/sqrt(2); HA(2:end) = HA/sqrt(2);
% LS= sqrt(2)*LS(2:end); HS = sqrt(2)*HS(2:end);
coeffs = [
       0.00134974786501001                         0                         0      -0.00269949573002003
      -0.00135360470301001                         0                         0      -0.00270720940602003
       -0.0120141966670801        0.0102009221870399        0.0204018443740798        0.0240283933341602
       0.00843901203981008       -0.0102300708193699        0.0204601416387398        0.0168780240796202
        0.0351664733065404       -0.0556648607799594        -0.111329721559919       -0.0703329466130807
       -0.0546333136825205        0.0285444717151497       -0.0570889434302994        -0.109266627365041
       -0.0665099006248407         0.295463938592917         0.590927877185834         0.133019801249681
         0.297547906345713        -0.536628801791565          1.07325760358313         0.595095812691426
         0.584015752240756         0.295463938592917         0.590927877185834         -1.16803150448151
         0.297547906345713        0.0285444717151497       -0.0570889434302994         0.595095812691426
       -0.0665099006248407       -0.0556648607799594        -0.111329721559919         0.133019801249681
       -0.0546333136825205       -0.0102300708193699        0.0204601416387398        -0.109266627365041
        0.0351664733065404        0.0102009221870399        0.0204018443740798       -0.0703329466130807
       0.00843901203981008                         0                         0        0.0168780240796202
       -0.0120141966670801                         0                         0        0.0240283933341602
      -0.00135360470301001                         0                         0      -0.00270720940602003
       0.00134974786501001                         0                         0      -0.00269949573002003
];
LA = coeffs(:,1); % Lowpass Analysis
HA = coeffs(:,2); % Highpass Analysis
LS = coeffs(:,3); % Lowpass Synthesis
HS = coeffs(:,4); % Highpass Synthesis

% Example audio file built into matlab.
audiodata = load('handel.mat');
x = audiodata.y(1:end-1); fs = audiodata.Fs;

% Analysis (level 1)
L1 = downsample(conv(x,LA),2);
H1 = downsample(conv(x,HA),2);

% Analysis (level 2)
L1L2 = downsample(conv(L1,LA),2);
L1H2 = downsample(conv(L1,HA),2);

% Synthesis
delay = length(coeffs)-1;

L1rec = conv(upsample(L1L2,2),LS) + conv(upsample(L1H2,2),HS);
L1rec = L1rec(delay:end-delay-1);

xrec = conv(upsample(L1rec,2),LS) + conv(upsample(H1,2),HS);
xrec = xrec(delay:end-delay-1);

% verify that xrec is the same as x (accounting for small numerical error)
assert( max(abs(x-xrec)) < 1e-10 )

% create spectrograms for original signal and components
figure; spectrogram(x,1024,0,1024,'yaxis',fs);
clim([-70,0]); title('Original signal')

highest_octave = conv(upsample(H1,2),HS);
figure; spectrogram(highest_octave,1024,0,1024,'yaxis',fs);
clim([-70,0]); title('Highest octave component');

middle_octave = conv(upsample(L1H2,2),HS);
middle_octave = conv(upsample(middle_octave,2),LS);
figure; spectrogram(middle_octave,1024,0,1024,'yaxis',fs);
clim([-70,0]); title('Middle octave component');

low_component = conv(upsample(L1,2),LS);
figure; spectrogram(low_component,1024,0,1024,'yaxis',fs);
clim([-70,0]); title('Low frequency component');
```