Mini Project #2: Octave-spaced FIR filter banks

Mr. Dan Jacobellis and Prof. Brian Evans

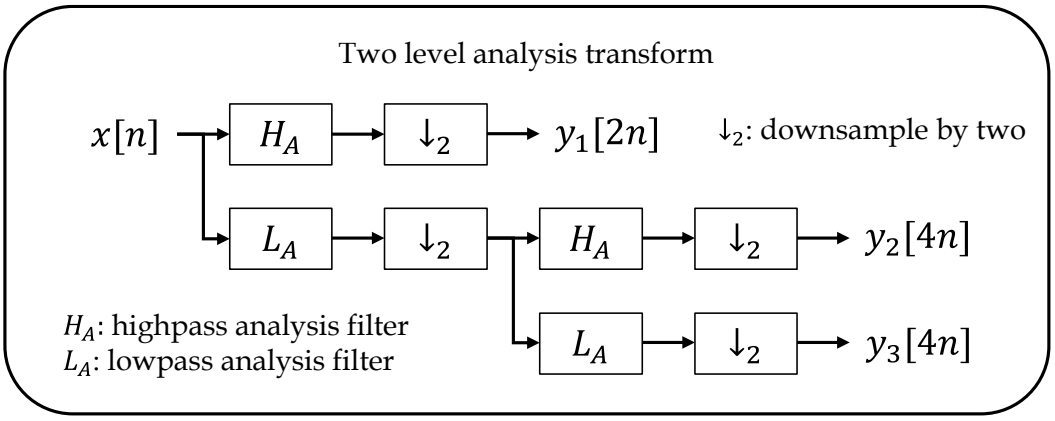
Solution Version 0.2

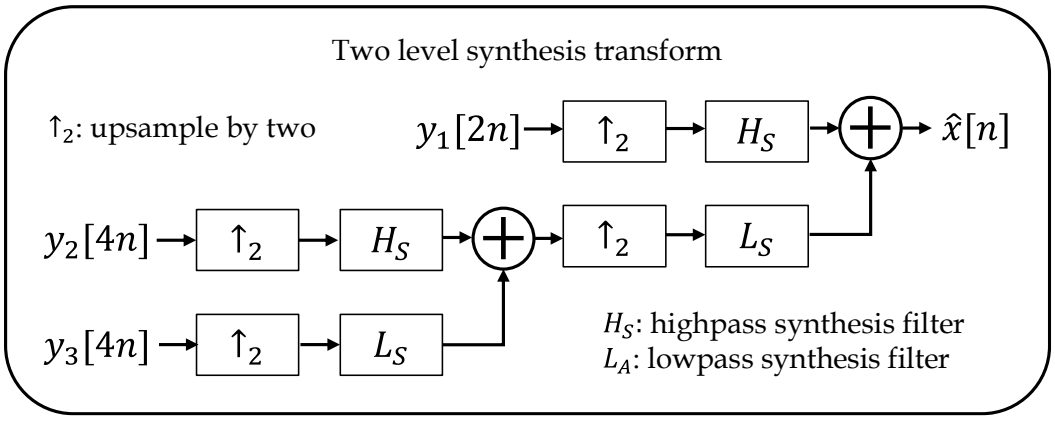
Reading: McClellan, Schafer and Yoder, *Signal Processing First*, 2003, Chapters 5-7. Errata.
Companion Web site with demos and other supplemental information: http://dspfirst.gatech.edu/
Web site contains solutions to selected homework problems from *DSP First*.

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, we 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**. An example using two levels is shown below. The MATLAB code for one, two, and four level transforms are presented in Appendices C, D, and E.





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

Consider sampling a continuous-time signal $x(t)$ at a rate $f_s = 1/T_s$, resulting in a discrete time signal $x[n] = x(t)|_{nT_s}$. In the time domain, downsampling by a factor of M is equivalent to sampling $x(t)$ at a lower rate $= \frac{f_s}{M}$. Upsampling by L is equivalent to sampling $x(t)$ at a higher rate Lf_s, then multiplying by an impulse train $\text{Ш}_L[n] = \sum_{k=-\infty}^{\infty} \delta[n - kL]$. In the special case that $L = 2$, $\text{Ш}_2[n] = \frac{1}{2} + \frac{1}{2}\cos(\pi n)$.

$\downarrow_M \{x[n]\} = x[n]$ downsampled by a factor of M $= x[k]\|_{k=Mn}$	$\uparrow_L \{x[n]\} = x[n]$ upsampled by a factor of L $= \text{Ш}_L[n]x[k]\|_{k=\frac{n}{L}}$

If a complex sinusoidal signal with frequency $-\pi \leq \hat{\omega} < \pi$ is downsampled by a factor of $M = 2$, the frequency is doubled:

$$\downarrow_2 \{e^{j\hat{\omega}n}\} = e^{j\hat{\omega}k}\big|_{k=2n} = e^{j(2\hat{\omega})n}.$$

If the original frequency $\hat{\omega}$ is larger than $\pi/2$ or smaller than $-\pi/2$, aliasing will occur.

Upsampling a complex sinusoidal signal with frequency $\hat{\omega}$ results in two frequency components: one at $\hat{\omega}/2$ and another at $\hat{\omega}/2 - \pi$.

$$\begin{aligned}
\uparrow_2 \{e^{j\hat{\omega}n}\} &= \left(\frac{1}{2} + \frac{1}{2}\cos(\pi n)\right) e^{j\hat{\omega}n/L} \\
&= \frac{1}{2} e^{j\hat{\omega}n/L} + \frac{1}{2} e^{j\hat{\omega}n/L} \cos(\pi n) \\
&= \frac{1}{2} e^{j\hat{\omega}n/L} + \frac{1}{4} e^{j\hat{\omega}n/L - \pi} + \frac{1}{4} e^{j\hat{\omega}n/L + \pi} \\
&= \frac{1}{2} e^{j\hat{\omega}n/L} + \frac{1}{4} e^{j\hat{\omega}n/L - \pi} + \frac{1}{4} e^{j\hat{\omega}n/L + \pi - 2\pi} \\
&= \frac{1}{2} e^{j\hat{\omega}n/L} + \frac{1}{2} e^{j\hat{\omega}n/L - \pi}
\end{aligned}$$

Upsampling and downsampling are linear operations, but not time-invariant. Systems that include upsampling and downsampling operations may not be LTI, but it is still useful to analyze the response of these systems to complex sinusoidal inputs.

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, and let $x[n] = e^{j\hat{\omega}n}$.

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

Complex sinusoids are eigenfunctions of LTI systems. When a complex sinusoid $x[n] = Ae^{j\hat{\omega}n+\phi}$ is input to an LTI system, the output is a complex sinusoid with the same frequency scaled by the frequency response i.e., $y[n] = H(e^{j\hat{\omega}})Ae^{j\hat{\omega}n+\phi}$. In this case, the output of the filter is

$$y_1[n] = x[n] * h_1[n] = H_1(e^{j\hat{\omega}})e^{j\hat{\omega}n}$$

The output of the filter-then-downsample cascade is:

$$y_1[2n] = y_1[m]|_{m=2n} = H_1(e^{j\hat{\omega}})e^{j(2\hat{\omega})n}$$

2. Consider the cascade of upsampling by a factor of two followed by an FIR filter with frequency response $H_2(e^{j\hat{\omega}})$, and let $x[n] = e^{j\hat{\omega}n}$.

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

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, the output of the upsampler is

$$\uparrow_2 \{e^{j\hat{\omega}n}\} = \frac{1}{2}e^{j\left(\frac{\hat{\omega}}{2}\right)n} + \frac{1}{2}e^{j\left(\frac{\hat{\omega}}{2}\right)n}\cos(\pi n) = \frac{1}{2}e^{j\left(\frac{\hat{\omega}}{2}\right)n} + \frac{1}{2}e^{j\left(\frac{\hat{\omega}}{2}-\pi\right)n}$$

The output of the upsample-then-filter cascade is:

$$\begin{aligned} y_2[n/2] &= h_2[n] * (\uparrow_2 \{e^{j\hat{\omega}n}\}) \\ &= h_2[n] * \left(\frac{1}{2}e^{j\left(\frac{\hat{\omega}}{2}\right)n} + \frac{1}{2}e^{j\left(\frac{\hat{\omega}}{2}-\pi\right)n}\right) \\ &= \left(h_2[n] * \frac{1}{2}e^{j\left(\frac{\hat{\omega}}{2}\right)n}\right) + \left(h_2[n] * \frac{1}{2}e^{j\left(\frac{\hat{\omega}}{2}-\pi\right)n}\right) \\ &= \frac{1}{2}H_2(e^{j\hat{\omega}/2})e^{j\left(\frac{\hat{\omega}}{2}\right)n} + \frac{1}{2}H_2(e^{j(\hat{\omega}/2-\pi)})e^{j\left(\frac{\hat{\omega}}{2}-\pi\right)n} \end{aligned}$$

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:

$$\uparrow_2 \{y_2[2n]\} = \frac{1}{2}H_1(e^{j\hat{\omega}})e^{j\hat{\omega}n} + \frac{1}{2}H_1(e^{j\hat{\omega}})e^{j\hat{\omega}n}\cos(\pi n) = \frac{1}{2}H_1(e^{j\hat{\omega}})e^{j\hat{\omega}n} + \frac{1}{2}H_1(e^{j\hat{\omega}})e^{j(\hat{\omega}-\pi)n}$$

$$\hat{x}[n] = \underbrace{\frac{1}{2}H_1(e^{j\hat{\omega}})H_2(e^{j\hat{\omega}})e^{j\hat{\omega}n}}_{\text{output component at original freuqency}} + \frac{1}{2}H_1(e^{j\hat{\omega}})H_2(e^{j(\hat{\omega}-\pi)})e^{j(\hat{\omega}-\pi)n}$$

Thus, 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:

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

Let $x[n] = e^{j\hat{\omega}n}$. $\hat{x}[n]$ will contain multiple frequencies:

$$y_1[2n] = H_1(e^{j\hat{\omega}})e^{j(2\hat{\omega})n}$$

$$y_2[4n] = H_1(e^{j\hat{\omega}})H_2(e^{j2\hat{\omega}})e^{j(4\hat{\omega})n}$$

$$h_3[n] *\uparrow_2 \{y_2[4n]\} = \frac{1}{2}H_1(e^{j\hat{\omega}})H_2(e^{j2\hat{\omega}})\left[H_3(e^{j2\hat{\omega}})e^{j(2\hat{\omega})n} + H_3(e^{j(2\hat{\omega}-\pi)})e^{j(2\hat{\omega}-\pi)n}\right]$$

$$\hat{x}[n] = \frac{1}{4}H_1(e^{j\hat{\omega}})H_2(e^{j2\hat{\omega}})\Big[H_3(e^{j2\hat{\omega}})\left(H_4(e^{j\hat{\omega}})e^{j\hat{\omega}n} + H_4(e^{j(\hat{\omega}-\pi)})e^{j(\hat{\omega}-\pi)n}\right) + H_3(e^{j2\hat{\omega}-\pi})\left(H_4\left(e^{j\left(\hat{\omega}-\frac{\pi}{2}\right)}\right)e^{j\left(\hat{\omega}-\frac{\pi}{2}\right)n} + H_4\left(e^{j\left(\hat{\omega}+\frac{\pi}{2}\right)}\right)e^{j\left(\hat{\omega}+\frac{\pi}{2}\right)n}\right)\Big]$$

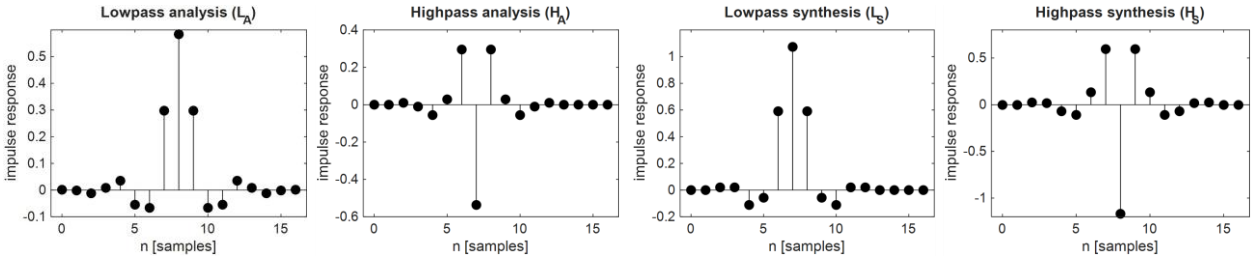
$$\begin{aligned}\hat{x}[n] = {} & \underbrace{\frac{1}{4}H_1(e^{j\hat{\omega}})H_2(e^{j2\hat{\omega}})H_3(e^{j2\hat{\omega}})H_4(e^{j\hat{\omega}})e^{j\hat{\omega}n}}_{\text{output component at original frequency}} \\ & + \frac{1}{4}H_1(e^{j\hat{\omega}})H_2(e^{j2\hat{\omega}})H_3(e^{j2\hat{\omega}})H_4(e^{j(\hat{\omega}-\pi)})e^{j(\hat{\omega}-\pi)n} \\ & + \frac{1}{4}H_1(e^{j\hat{\omega}})H_2(e^{j2\hat{\omega}})H_3(e^{j2\hat{\omega}-\pi})H_4\left(e^{j\left(\hat{\omega}-\frac{\pi}{2}\right)}\right)e^{j\left(\hat{\omega}-\frac{\pi}{2}\right)n} \\ & + \frac{1}{4}H_1(e^{j\hat{\omega}})H_2(e^{j2\hat{\omega}})H_3(e^{j2\hat{\omega}-\pi})H_4\left(e^{j\left(\hat{\omega}+\frac{\pi}{2}\right)}\right)e^{j\left(\hat{\omega}+\frac{\pi}{2}\right)n}\end{aligned}$$

Thus, for the output component at the original frequency $\hat{\omega}$, the effective frequency response is

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

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.



We can use freqz to calculate the frequency response of these filters. We can also use freqz to calculate the effective frequency response at the original frequency when the filters are cascaded with upsampling and downsampling operations, though other frequencies will also be created by upsampling. For the two level transform, the effective frequency responses of the three branches are:

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: $(L_A \circ \downarrow_2 \circ H_A \circ \downarrow_2 \circ \uparrow_2 \circ H_S \circ \uparrow_2 \circ L_S)$

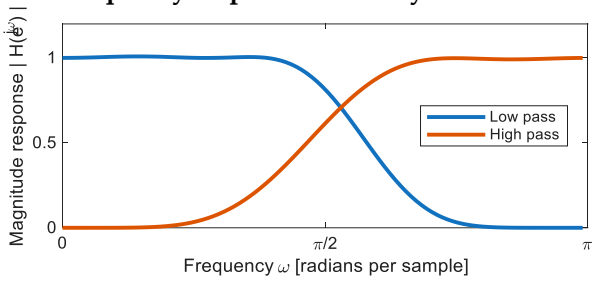
$$H_2(e^{j\hat{\omega}}) = \frac{1}{4} H_{LA}(e^{j\hat{\omega}}) H_{HA}(e^{2j\hat{\omega}}) H_{HS}(e^{j2\hat{\omega}}) H_{LS}(e^{j\hat{\omega}})$$

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

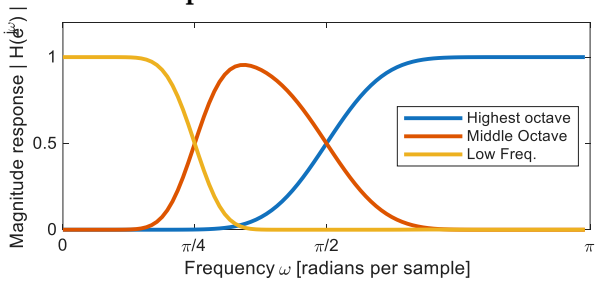
$$H_3(e^{j\hat{\omega}}) = \frac{1}{4} H_{LA}(e^{j\hat{\omega}}) H_{LA}(e^{2j\hat{\omega}}) H_{LS}(e^{j2\hat{\omega}}) H_{LS}(e^{j\hat{\omega}})$$

```
Nplot = 101;
[HLA, w] = freqz(LA,1,Nplot); [HHA, w] = freqz(HA,1,Nplot);
[HLS, w] = freqz(LS,1,Nplot); [HHS, w] = freqz(HS,1,Nplot);

figure; plot(w,abs(HLA),linewidth=2); hold on; plot(w,abs(HHA),linewidth=2);
xlabel('Frequency \omega [radians per sample]')
ylabel('Magnitude response | H(e^{j\omega}) |')
legend("Low pass","High pass")

H1 = 0.5*HHA.*HHS; figure; plot(w,abs(H1),linewidth=2)
HHA2 = downsample([HHA; flipud(conj(HHA))],2);
HHS2 = downsample([HHS; flipud(conj(HHS))],2);
H2 = 0.25*HLA.*HHA2.*HHS2.*HLS; hold on; plot(w,abs(H2),linewidth=2)
HLA2 = downsample([HLA; flipud(conj(HLA))],2);
HLS2 = downsample([HLS; flipud(conj(HLS))],2);
H3 = 0.25*HLA.*HLA2.*HLS2.*HLS; hold on; plot(w,abs(H3),linewidth=2)

xlim([0,pi]); set(gca,'XTick',[0,pi/4,pi/2,pi])
set(gca,'XTickLabels',["0", "\pi/4","\pi/2","\pi"])
xlabel('Frequency \omega [radians per sample]')
ylabel('Magnitude response | H(e^{j\omega}) |')
legend("Highest octave", "Middle Octave", "Low Freq.")
```

Frequency response for Analysis filters

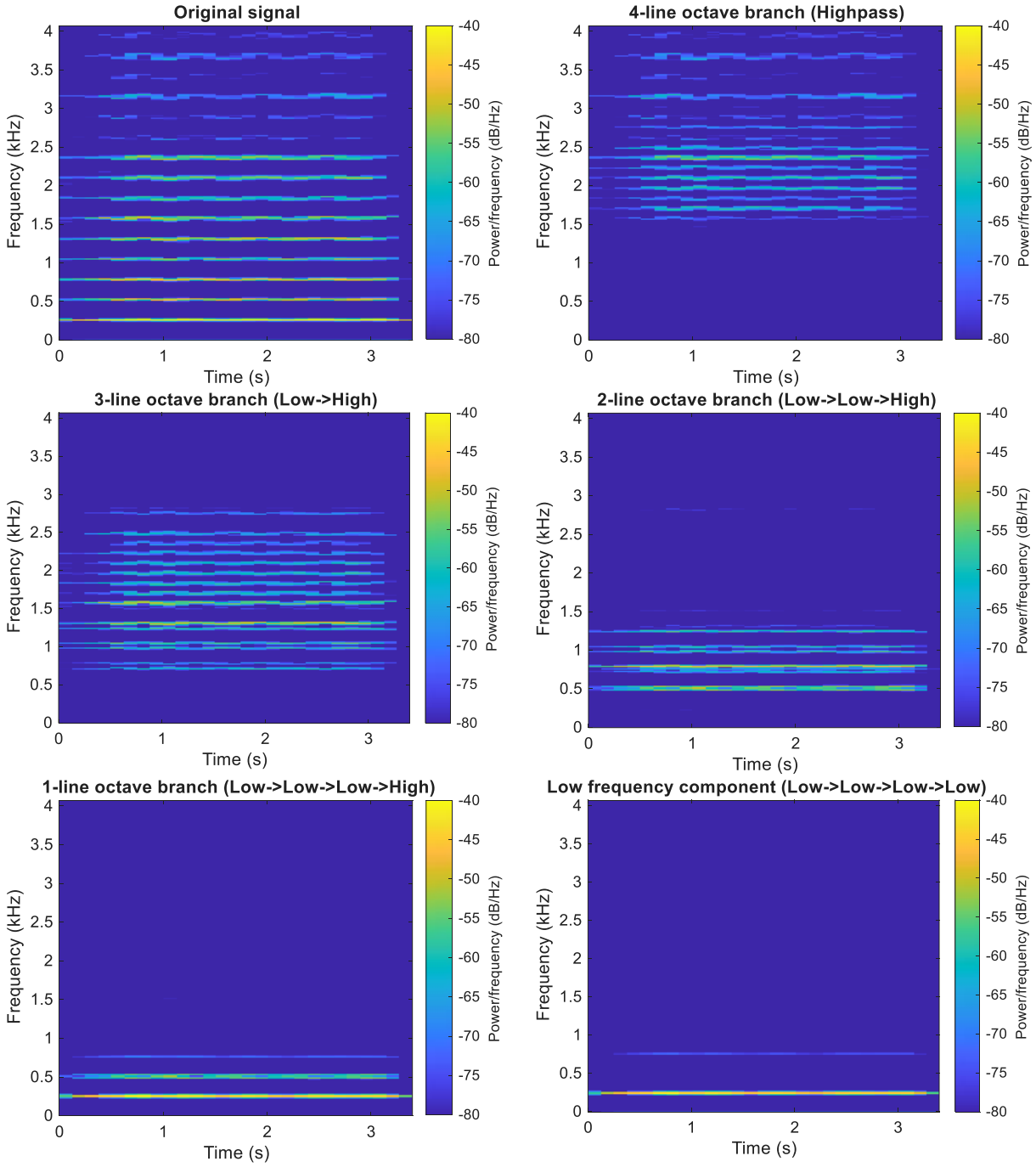


Effective responses for two level transform.



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. See the MATLAB code `four_level_octave.m` provided in Appendix E. We ensure that the output signal approximately matches the input using the line `assert( max(abs(x-xrec)) < 1e-10 )`. The spectrograms corresponding the outputs of individual branches is shown:



5.0 Playback and use as compression system

Using the filter bank from part 4.0, we divide the signal into 5 bands (low frequency + four octaves), Each branch is reconstructed independently and played as audio using soundsc. We compute the PSNR[1] of each band compared the original signal, and plot it vs the compression ratio.

```
four_level_octave; close all; % run the four_level_octave.m script first.
PSNR = @(x,y) -10*log10(mean(abs(x - y).^2)) + 6.02;
CR = @(x,y) length(x)/length(y);
PSNR_CR_table = [ PSNR(x, y1_rec), CR(x, y1)
                  PSNR(x, y2_rec), CR(x, y2)
                  PSNR(x, y3_rec), CR(x, y3)
                  PSNR(x, y4_rec), CR(x, y4)
                  PSNR(x, low_rec), CR(x, low_freq)];
figure;hold on;
plot(PSNR_CR_table(1,2), PSNR_CR_table(1,1), 'hexagramk',MarkerSize=10,
LineWidth=2);
plot(PSNR_CR_table(2,2), PSNR_CR_table(2,1), 'pentagramk',MarkerSize=10,
LineWidth=2);
plot(PSNR_CR_table(3,2), PSNR_CR_table(3,1), 'sk',MarkerSize=10, LineWidth=2);
plot(PSNR_CR_table(4,2), PSNR_CR_table(4,1), '^k',MarkerSize=10, LineWidth=2);
plot(PSNR_CR_table(5,2), PSNR_CR_table(5,1), 'ok',MarkerSize=10, LineWidth=2);
ylim([0,35]); xlim([1,17]); grid on; set(gca,'XTick',[1,2,4,8,16]);
title('PSNR vs Compression ratio when using single band for reconstruction')
xlabel('Compression Ratio')
ylabel('PSNR')
legend('4-line octave branch','3-line octave branch', ...
    '2-line octave branch','1-line octave branch', 'low frequency branch');
soundsc(y1_rec,fs); pause(16);
soundsc(y2_rec,fs); pause(16);
soundsc(y3_rec,fs); pause(16);
soundsc(y4_rec,fs); pause(16);
soundsc(low_rec,fs); pause(16);
```

---

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

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

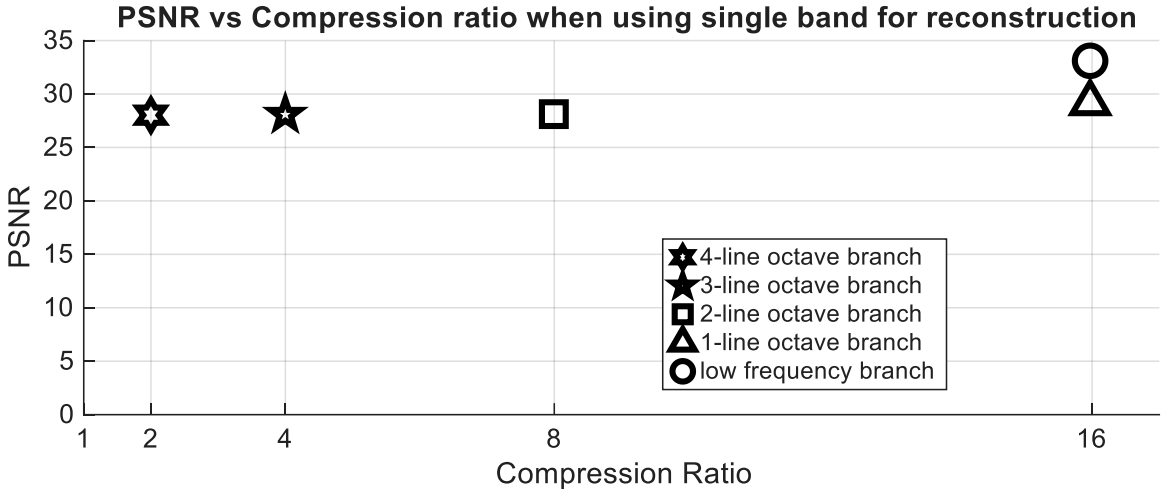
where

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

In MATLAB:

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

After each downsampling operation the number of samples is reduced by two. If the other bands are discarded, this leads to a doubling of the compression ratio[2]. If the group delay is not accounted for until after the synthesis transform, the compression ratio will be slightly lower than a power of two. Reconstructing individual bands provides roughly 28 dB PSNR, except for the lowest frequency band (33 dB). The low frequency component provides the best trade-off between PSNR and compression ratio.



Frequency band	PSNR (dB)	Comp. Ratio	Description of sound
4-line octave	28.02	2.000	The highest harmonics in the original signal are audible. Other audible artifacts are also present corresponding to frequencies that were not present in the original signal.
3-line octave	28.07	3.999	The treble range notes and harmonics in the original signal are audible. Artifacts and frequencies not in the original signal are also present.
2-line octave	28.12	7.993	The upper-mid range notes and harmonics in the original signal are audible. Artifacts and frequencies not in the original signal are also present.
1-line octave	29.15	15.97	The lower-mid range notes and harmonics in the original signal are audible. Artifacts and frequencies not in the original signal are also present.
Low frequency	33.10	15.97	The bass notes in the original signal are present, but only the fundamental frequency. The harmonics are significantly attenuated. Some other frequencies and artifacts are also audible.

[2] The compression ratio for the ith band is $\mathrm{CR}_i = \frac{\text{Number of samples in the original audio signal}}{\text{Number of output samples for } i\text{th band}}$. In the original mini project assignment, it was erroneously defined as the reciprocal. Either ratio can be used to analyze the efficacy of a compression system as long as it is interpreted appropriately.

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)`.

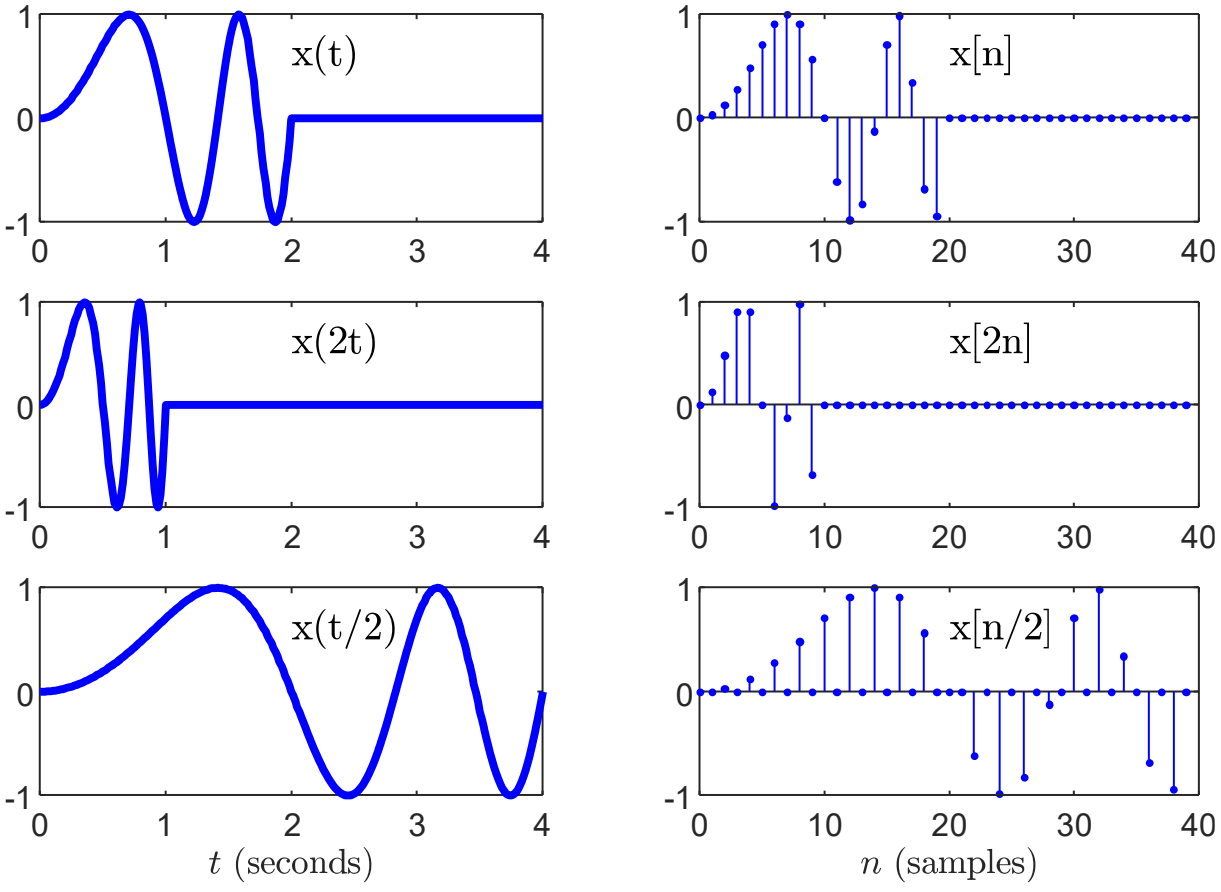
Upsampling a signal by an integer factor L (denoted $\uparrow_L$) makes the signal L times longer by inserting zeros following a regular pattern.

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

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');
```

Appendix E: Matlab code for four_level_octave.m

```
% Fall 2025 mini project #2 solution to apply four level DWT
% Originally written by Dan Jacobellis 11/3/2025
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); HA = coeffs(:,2); LS = coeffs(:,3); HS = coeffs(:,4);

%% Loading audio data
% for the violin clip (fs=11.05 kHz) in part 4, the resample by 166/225
% The signal is truncated to a multiple of 16 samples
x = audioread('violin-C4.wav'); x = resample(x,166,226); fs=8134; x = x(1:28272);
% for 44.1kHz audio in part 5, the resampling factor is 83/450
x = mean(audioread('sm_cello.mp3'),2); x = resample(x,83,450); fs=8134; x = x(1:130144);

%% Analysis transform
analysis_transform = @(x,h) downsample(conv(x,h),2);
low_freq = x;
y1 = analysis_transform(low_freq, HA); % 4-line octave
low_freq = analysis_transform(low_freq, LA);
y2 = analysis_transform(low_freq, HA);  % 3-line octave
low_freq = analysis_transform(low_freq, LA);
y3 = analysis_transform(low_freq, HA);  % 2-line octave
low_freq = analysis_transform(low_freq, LA);
y4 = analysis_transform(low_freq, HA);  % 1-line octave
low_freq = analysis_transform(low_freq, LA);

%% Synthesis transform
delay = length(coeffs)-1; % combined delay of analyis and synthesis
account_for_delay = @(x) x(delay:end-delay-1);
synthesis_transform = @(x,h) account_for_delay(conv(upsample(x,2),h));

y1_rec = synthesis_transform(y1, HS); % 4-line octave
y2_hs = synthesis_transform(y2, HS); % 3-line octave
y2_rec = synthesis_transform(y2_hs, LS);
y3_hs = synthesis_transform(y3, HS); % 2-line octave
y3_up1 = synthesis_transform(y3_hs, LS);
y3_rec = synthesis_transform(y3_up1, LS);
y4_hs = synthesis_transform(y4, HS); % 1-line octave branch
y4_up1 = synthesis_transform(y4_hs, LS);
y4_up2 = synthesis_transform(y4_up1, LS);
y4_rec = synthesis_transform(y4_up2, LS);
low_up1 = synthesis_transform(low_freq, LS); % Low frequency branch
low_up2 = synthesis_transform(low_up1, LS);
low_up3 = synthesis_transform(low_up2, LS);
low_rec = synthesis_transform(low_up3, LS);

% Full reconstruction
xrec = low_rec + y4_rec + y3_rec + y2_rec + y1_rec;
assert( max(abs(x-xrec)) < 1e-10 )
```