b. Find $A\mathbf{v}_1$ and $A\mathbf{v}_2$.

c. Use what you found in the previous part of this problem to find $\{A\mathbf{v}_1\}_{\mathcal{B}}$ and $\{A\mathbf{v}_2\}_{\mathcal{B}}$.

d. If $\{\mathbf{x}\}_{\mathcal{B}} = \begin{bmatrix} 1 \\ -5 \end{bmatrix}$, find $\{A\mathbf{x}\}_{\mathcal{B}}$.

e. Find a matrix D such that $\{A\mathbf{x}\}_{\mathcal{B}} = D\{\mathbf{x}\}_{\mathcal{B}}$.

You should find that the matrix D is a very simple matrix, which means that this basis $\mathcal{B}$ is well suited to study the effect of multiplication by A. This observation is the central idea of the next chapter.

a. The vectors are linearly independent and span $\mathbb{R}^2$.

b. We compute that $A\mathbf{v}_1 = 3\mathbf{v}_1$ and $A\mathbf{v}_2 = \mathbf{v}_2$.

c. $\{A\mathbf{v}_1\}_{\mathcal{B}} = \begin{bmatrix} 3 \\ 0 \end{bmatrix}$ and $\{A\mathbf{v}_2\}_{\mathcal{B}} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$.

d. We know that $\mathbf{x} = \mathbf{v}_1 - 5\mathbf{v}_2$, which means that $A\mathbf{x} = 3\mathbf{v}_1 - 5\mathbf{v}_2$. Therefore, $\{A\mathbf{x}\}_{\mathcal{B}} = \begin{bmatrix} 3 \\ -5 \end{bmatrix}$.

e. If $\{\mathbf{x}\}_{\mathcal{B}} = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}$, then $\mathbf{x} = c_1\mathbf{v}_1 + c_2\mathbf{v}_2$ and $A\mathbf{x} = 3c_1\mathbf{v}_1 + c_2\mathbf{v}_2$. Therefore, $D\begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} 3c_1 \\ c_2 \end{bmatrix}$, which says that $D = \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix}$.

a. The vectors are linearly independent and span $\mathbb{R}^2$.

b. $A\mathbf{v}_1 = 3\mathbf{v}_1$ and $A\mathbf{v}_2 = \mathbf{v}_2$.

c. $\{A\mathbf{v}_1\}_{\mathcal{B}} = \begin{bmatrix} 3 \\ 0 \end{bmatrix}$ and $\{A\mathbf{v}_2\}_{\mathcal{B}} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$.

d. $\{A\mathbf{x}\}_{\mathcal{B}} = \begin{bmatrix} 3 \\ -5 \end{bmatrix}$.

e. $D = \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix}$.

3.3 Image compression

Digital images, such as the photographs taken on your phone, are displayed as a rectangular array of pixels. For example, the photograph in Figure 1 is 1440 pixels wide and 1468 pixels high. If we were to zoom in on the photograph, we would be able to see individual pixels, such as those shown on the right.

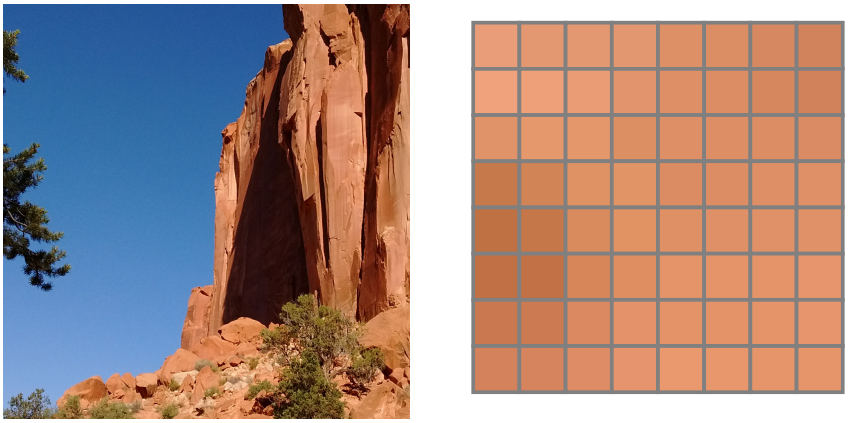


Figure 3.3.1: An image stored as a 1440×1468 array of pixels along with a close up of a smaller 8×8 array.

A lot of data is required to display this image. A quantity of digital data is frequently measured in bytes, where one byte is the amount of storage needed to record an integer between 0 and 255. As we will see shortly, each pixel requires three bytes to record that pixel's color. This means the amount of data required to display this image is $3 \times 1440 \times 1468 = 6,341,760$ bytes or about 6.3 megabytes.

Of course, we would like to store this image on a phone or computer and perhaps transmit it through our data plan to share it with others. If possible, we would like to find a way to represent this image using a smaller amount of data so that we don't run out of memory on our phone and quickly exhaust our data plan.

As we will see in this section, the JPEG compression algorithm provides a means for doing just that. This image, when stored in the JPEG format, requires only 467,359 bytes of data, which is about 7

Preview Activity 3.3.1. Since we will be using various bases and the coordinate systems they define, let's review how we translate between coordinate systems.

a. Suppose that we have a basis $\mathcal{B} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m\}$ for $\mathbb{R}^m$. Explain what we mean by the representation $\{\mathbf{x}\}_\mathcal{B}$ of a vector $\mathbf{x}$ in the coordinate system defined by $\mathcal{B}$.

b. If we are given the representation $\{\mathbf{x}\}_\mathcal{B}$, how can we recover the vector $\mathbf{x}$?

c. If we are given the vector $\mathbf{x}$, how can we find $\{\mathbf{x}\}_\mathcal{B}$?

d. Suppose that

$$\mathcal{B} = \left\{ \begin{bmatrix} 1 \\ 3 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\}$$

is a basis for $\mathbb{R}^2$. If $\{\mathbf{x}\}_\mathcal{B} = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$, find the vector $\mathbf{x}$.

e. If $\mathbf{x} = \begin{bmatrix} 2 \\ -4 \end{bmatrix}$, find $\{\mathbf{x}\}_\mathcal{B}$.

a. The components of the vector $\{\mathbf{x}\}_{\mathcal{B}}$ are the weights that express $\mathbf{x}$ as a linear combination of the basis vectors; that is, $\{\mathbf{x}\}_{\mathcal{B}} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \end{bmatrix}$ if $\mathbf{x} = c_1\mathbf{v}_1 + c_2\mathbf{v}_2 + \ldots + c_m\mathbf{v}_m$.

b. If we form the matrix $C_{\mathcal{B}} = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \ldots & \mathbf{v}_m \end{bmatrix}$, then $\mathbf{x} = C_{\mathcal{B}}\{\mathbf{x}\}_{\mathcal{B}}$.

c. As before, $\{\mathbf{x}\}_{\mathcal{B}} = C_{\mathcal{B}}^{-1}\mathbf{x}$.

d. We find $\mathbf{x} = C_{\mathcal{B}}\{\mathbf{x}\}_{\mathcal{B}} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$.

e. We find $\{\mathbf{x}\}_{\mathcal{B}} = C_{\mathcal{B}}^{-1}\mathbf{x} = \begin{bmatrix} -3 \\ 5 \end{bmatrix}$.

3.3.1 Color models

A color is represented digitally by a vector in $\mathbb{R}^3$. There are different ways in which we can represent colors, however, depending on whether a computer or a human will be processing the color. We will describe two of these representations, called *color models*, and demonstrate how they are used in the JPEG compression algorithm.

Digital displays typically create colors by blending together various amounts of red, green, and blue. We can therefore describe a color by putting its constituent amounts of red, green, and blue into a vector $\begin{bmatrix} R \\ G \\ B \end{bmatrix}$. The quantities R, G, and B are stored with one byte of information so they are allowed to vary between 0 and 255. This is called the RGB color model.

We define a basis $\mathcal{B} = \{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3\}$ where

$$\mathbf{v}_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \mathbf{v}_2 = \begin{bmatrix} 0 \\ -0.34413 \\ 1.77200 \end{bmatrix}, \mathbf{v}_3 = \begin{bmatrix} 1.40200 \\ -0.71414 \\ 0 \end{bmatrix}$$

to define a new coordinate system with coordinates we denote Y, C_b, and C_r:

$$\left\{ \begin{bmatrix} R \\ G \\ B \end{bmatrix} \right\}_{\mathcal{B}} = \begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix}.$$

The coordinate Y is called *luminance* while C_b and C_r are called blue and red *chrominance*, respectively. In this coordinate system, luminance will vary from 0 to 255, while the chrominances vary between -127.5 and 127.5. This is known as the YC_bC_r color model. (To be completely accurate, we should add 127.5 to the chrominance values so that they lie between 0 and 255, but we won't worry about that here.)

Activity 3.3.2. In this activity, we will explore the difference between these two coordinate systems.

a. First, we will explore the RGB color model.

The diagram available at the top of http://gvsu.edu/s/0Jc enables you to create colors using various amounts of red, green, and blue. For each of these three quantities, the slider varies between 0 and 255.

i. What happens when $G = 0$, $B = 0$ (pushed all the way to the left), and R is allowed to vary?

ii. What happens when $R = 0$, $G = 0$, and B is allowed to vary?

iii. How can you create black in this color model?

iv. How can you create white?

b. Next, we will explore the YC_bC_r color model.

The diagram available in the middle of http://gvsu.edu/s/0Jc enables you to create colors using various amounts of luminance Y, blue chrominance C_b, and red chrominance C_r. The lumninance slider moves between 0 and 255 while the chrominance sliders move between -127.5 and 127.5.

i. What happens when $C_b = 0$ and $C_r = 0$ (kept in the center) and Y is allowed to vary?

ii. What happens when $Y = 0$ (pushed to the left), $C_r = 0$ (kept in the center), and C_b is allowed to increase between 0 and 127.5?

iii. What happens when $Y = 0$, $C_b = 0$, and C_r is allowed to increase between 0 and 127.5?

iv. How can you create black in this color model?

v. How can you create white?

c. Verify that $\mathcal{B}$ is a basis for $\mathbb{R}^3$.

d. Find the matrix $C_{\mathcal{B}}$ that converts from $\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix}$ coordinates into $\begin{bmatrix} R \\ G \\ B \end{bmatrix}$ coordinates. Then find the matrix $C_{\mathcal{B}}^{-1}$ that converts from $\begin{bmatrix} R \\ G \\ B \end{bmatrix}$ coordinates back into $\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix}$ coordinates.

e. Find the $\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix}$ coordinates for the following colors and check, using the diagrams above, that the two representations agree.

i. Pure red is $\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 255 \\ 0 \\ 0 \end{bmatrix}$.

ii. Pure green is $\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 0 \\ 255 \\ 0 \end{bmatrix}$.

iii. Pure blue is $\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 255 \end{bmatrix}$.

iv. Pure white is $\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 255 \\ 255 \\ 255 \end{bmatrix}$.

v. Pure black is $\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$.

f. Find the $\begin{bmatrix} R \\ G \\ B \end{bmatrix}$ coordinates for the following colors and check, using the diagrams above, that the two representations agree.

i. $\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 128 \\ 0 \\ 0 \end{bmatrix}$.

ii. $\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 128 \\ 60 \\ 0 \end{bmatrix}$.

iii. $\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 128 \\ 0 \\ 60 \end{bmatrix}$.

g. Write an expression for

i. The luminance Y as it depends on R, G, and B.

ii. The blue chrominance C_b as it depends on R, G, and B.

iii. The red chrominance C_r as it depends on R, G, and B.

Explain how these quantities can be roughly interpreted by stating that

i. the luminance represents the brightness of the color.

ii. the blue chrominance measures the amount of blue in the color.

iii. the red chrominance measures the amount of red in the color.

These two color models provide us with two ways to represent colors, each of which is useful in a certain context. Digital displays, such as those in phones and computer monitors, create colors by combining differing amounts of red, green, and blue. The RGB model is therefore most relevant in digital applications.

By contrast, the YC_bC_r color model was created based on research into human vision and aims to concentrate the most visually important data into a single coordinate, the luminance, to which our eyes are most sensitive. Of course, any basis of $\mathbb{R}^3$ must have three vectors so we need two more coordinates, blue and red chrominance, if we want to represent all colors.

To see this explicitly, shown in Figure 2 is the original image and the image as rendered with only the luminance. That is, on the right, the color of each pixel is represented by only byte, which is the luminance. This image essentially looks like a grayscale version of the original image with all its visual detail. In fact, before digital television became the standard, television signals

were broadcast using the YC_bC_r color model. When a signal was displayed on a black-and-white television, the luminance was displayed and the two chrominance values simply ignored.

Figure 3.3.2: The original image rendered with only the luminance values.

For comparison, shown in Figure 3 are the corresponding images created using only the blue chrominance and the red chrominance. Notice that the amount of visual detail is considerably less in these images.

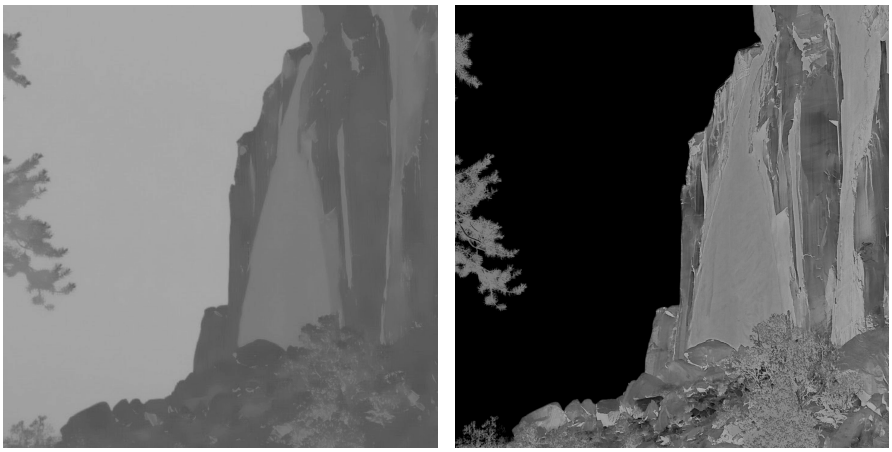


Figure 3.3.3: The original image rendered, on the left, with only blue chrominance and, on the right, with only red chrominance.

In the JPEG compression algorithm, we are interested in representing an image using the smallest amount of data possible. By converting from the RGB color model to the YC_bC_r color model, we are concentrating the most visually important data into a single quantity. This is helpful because we can safely ignore some of the data in the chrominance values since that data is not as visually important.

3.3.2 The JPEG compression algorithm

The key to representing the image using a smaller amount of data is to detect redundancies in the data. For this reason, we will break the image, which is composed of 1440×1468 pixels, into small 8×8 blocks of pixels. For example, we will consider the 8×8 block of pixels outlined in green in the original image, shown on the left of Figure 4. The image on the right zooms in on the block.

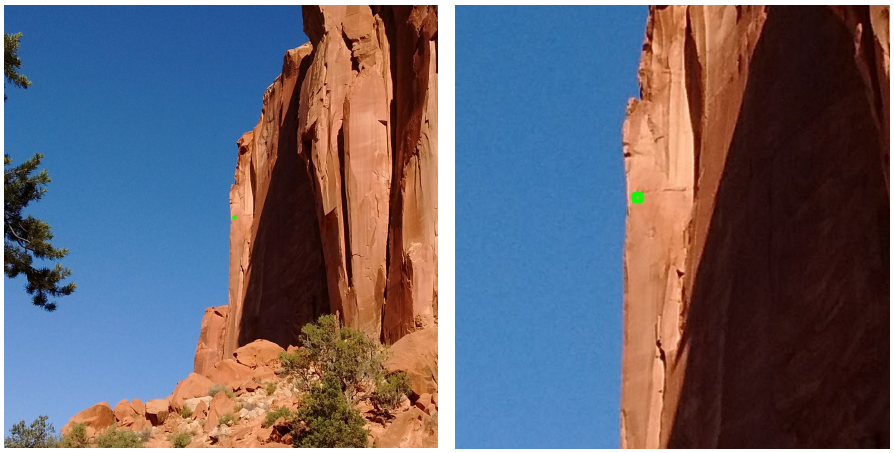


Figure 3.3.4: An 8×8 block of pixels outlined in green in the original image on the left. We see the same block on a smaller scale on the right.

Notice that this block, as seen in the original image, is very small. If we were to change some of the colors in this block slightly, our eyes would probably not notice.

Here we see a close up of the block. The important point here is that the colors do not change too much over this block. In fact, we expect this to be true for most of the blocks. There will, of course, be some blocks that contain dramatic changes, such as where the sky and rock intersect, but they will be the exception.

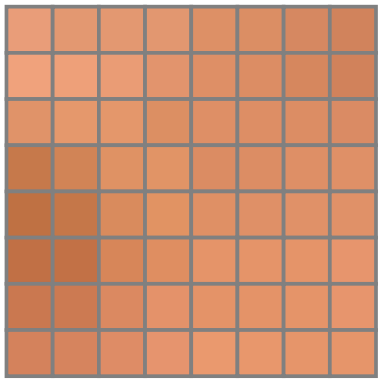
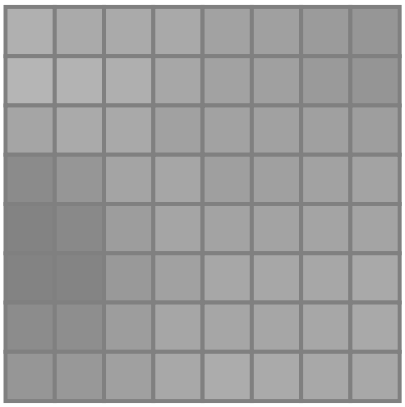


Figure 3.3.5: A close up of the 8×8 block we are considering.

Following our earlier work, we will change the representation of colors from the RGB color model to the YC_bC_r model. This separates the colors into luminance and chrominance values that we will consider separately. In Figure 6, we see the luminance values of this block. Again, notice how these values do not vary significantly over the block.



176	170	170	169	162	160	155	150
181	179	175	167	162	160	154	149
165	170	169	161	162	161	160	158
139	150	164	166	159	160	162	163
131	137	157	165	163	163	164	164
131	132	153	161	167	167	167	169
140	142	157	166	166	166	167	169
150	152	160	168	172	170	168	168

Figure 3.3.6: The luminance values in this block.

Our strategy in the compression algorithm is to perform a change of basis to take advantage of the fact that the luminance values do not change significantly over the block. Rather than recording the luminance of each of the pixels, this change of basis will allow us to record the average luminance along with some information about how the individual colors vary from the average.

Let's look at the first column of luminance values, which is a vector in $\mathbb{R}^8$:

$$\mathbf{x} = \begin{bmatrix} 176 \\ 181 \\ 165 \\ \vdots \\ 150 \end{bmatrix}.$$

We will perform a change of basis so that we can describe this vector by the average of the luminance values and information about variations from the average.

The JPEG compression algorithm uses the *Discrete Fourier Transform*, which is defined using the basis $\mathcal{C}$ whose basis vectors are

$$\mathbf{v}_0 = \begin{bmatrix} \cos\left(\frac{(2\cdot 0+1)\cdot 0\pi}{16}\right) \\ \cos\left(\frac{(2\cdot 1+1)\cdot 0\pi}{16}\right) \\ \cos\left(\frac{(2\cdot 2+1)\cdot 0\pi}{16}\right) \\ \vdots \\ \cos\left(\frac{(2\cdot 7+1)\cdot 0\pi}{16}\right) \end{bmatrix}, \mathbf{v}_1 = \begin{bmatrix} \cos\left(\frac{(2\cdot 0+1)\cdot 1\pi}{16}\right) \\ \cos\left(\frac{(2\cdot 1+1)\cdot 1\pi}{16}\right) \\ \cos\left(\frac{(2\cdot 2+1)\cdot 1\pi}{16}\right) \\ \vdots \\ \cos\left(\frac{(2\cdot 7+1)\cdot 1\pi}{16}\right) \end{bmatrix},$$

$$\ldots, \mathbf{v}_6 = \begin{bmatrix} \cos\left(\frac{(2\cdot 0+1)\cdot 6\pi}{16}\right) \\ \cos\left(\frac{(2\cdot 1+1)\cdot 6\pi}{16}\right) \\ \cos\left(\frac{(2\cdot 2+1)\cdot 6\pi}{16}\right) \\ \vdots \\ \cos\left(\frac{(2\cdot 7+1)\cdot 6\pi}{16}\right) \end{bmatrix}, \mathbf{v}_7 = \begin{bmatrix} \cos\left(\frac{(2\cdot 0+1)\cdot 7\pi}{16}\right) \\ \cos\left(\frac{(2\cdot 1+1)\cdot 7\pi}{16}\right) \\ \cos\left(\frac{(2\cdot 2+1)\cdot 7\pi}{16}\right) \\ \vdots \\ \cos\left(\frac{(2\cdot 7+1)\cdot 7\pi}{16}\right) \end{bmatrix}.$$

On first glance, this probably looks intimidating, but we can make sense of it by looking at these vectors graphically. Shown in Figure 7 are four of these basis vectors. Notice that $\mathbf{v}_0$ is constantly 1, $\mathbf{v}_1$ is relatively slowly varying, $\mathbf{v}_2$ varies a little more rapidly, and $\mathbf{v}_7$ varies quite rapidly. This is the main observation: the basis vectors vary at different rates with the first vectors varying relatively slowly.

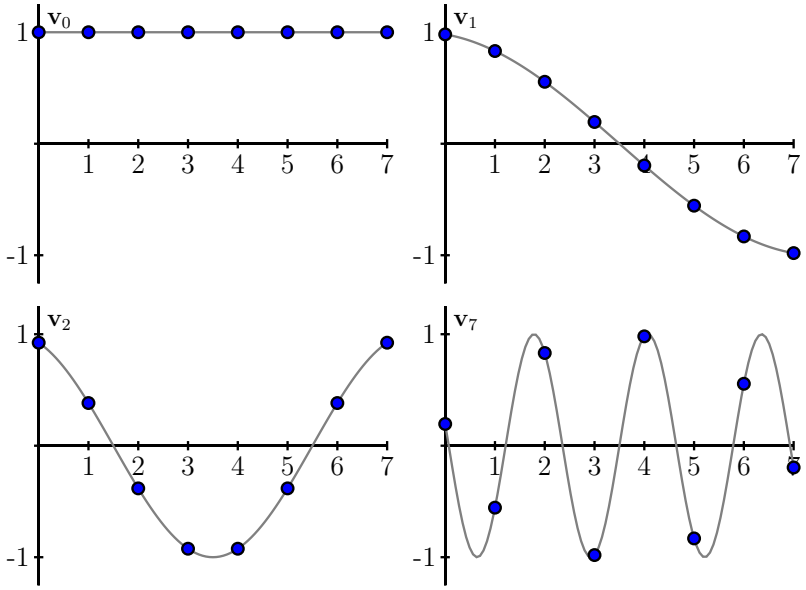
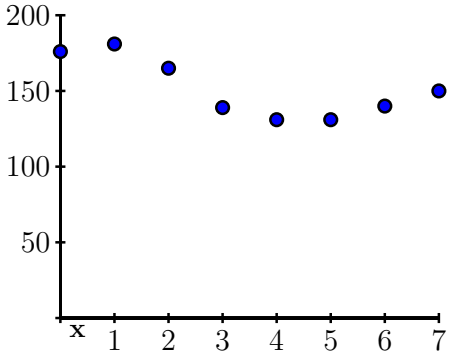


Figure 3.3.7: Four of the basis vectors $\mathbf{v}_0$, $\mathbf{v}_1$, $\mathbf{v}_2$, and $\mathbf{v}_7$.

These vectors form the basis $\mathcal{C}$ for $\mathbb{R}^8$. Remember that $\mathbf{x}$ is the vector of luminance values in the first column as seen on the right. We will write $\mathbf{x}$ in the new coordinates

$$\{\mathbf{x}\}_{\mathcal{C}} = \begin{bmatrix} F_0 \\ F_1 \\ F_2 \\ \vdots \\ F_7 \end{bmatrix}.$$

The coordinates F_j are called the *Fourier coefficients* of the vector $\mathbf{x}$.



Activity 3.3.3. We will explore the influence that the Fourier coefficients have on the vector $\mathbf{x}$.

The diagram available at http://gvsu.edu/s/0Jd enables you to vary three of the Fourier coefficients F_0, F_3, and F_7 and observe the effect on $\mathbf{x}$.

a. Describe the effect on the vector $\mathbf{x}$ when you vary F_0.

b. Now observe the effect on $\mathbf{x}$ when F_3 and F_7 are varied. Compare the effect of F_0, F_3, and F_7.

c. If the vector $\mathbf{x}$ shows only small variations, what would you expect to be true of the Fourier coefficients F_j?

d. The Sage cell below will construct the vector $C_{\mathcal{B}}$, which is denoted `C`, and its inverse $C_{\mathcal{B}}^{-1}$, which is denoted `Cinv`. Evaluate this Sage cell and notice that it prints the matrix $C_{\mathcal{B}}^{-1}$.

```
mat = []
for i in range(8):
    for j in range(8):
        mat.append(cos((2*i+1)*j*pi/16))
C = matrix(8,8, mat).numerical_approx()
Cinv = C.inverse()
print Cinv.numerical_approx(digits=3)
```

Now look at the form of $C_{\mathcal{B}}^{-1}$ and explain why F_0 is the average of the luminance values in the vector $\mathbf{x}$.

e. The Sage cell below defines the vector $\mathbf{x}$, which is the vector of luminance values in the first column, as seen in Figure 6. Use the cell below to find the vector $\mathbf{f}$ of Fourier coefficients $F_0, F_1, \ldots, F_7$. If you have evaluated the cell above, you will still be able to refer to `C` and `Cinv` in this cell.

```
x = vector([176,181,165,139,131,131,140,150])
#  find the vector of Fourier coefficients f below
f =
print f.numerical_approx(digits=4)
```

Write the Fourier coefficients and discuss the relative sizes of the coefficients.

f. We see that the coefficients F_6 and F_7, which correspond to rapid variations in the luminance values, are quite small. Let's see what happens when we ignore them. Form a new vector of Fourier coefficients by rounding the coefficients to the nearest integer and setting F_6 and F_7 to zero. This is an approximation to $\mathbf{f}$, the vector of Fourier coefficients. Use the approximation to $\mathbf{f}$ to form an approximation of the vector $\mathbf{x}$.

```
# define fapprox below and then find xapprox
fapprox =
xapprox =
print "x␣␣␣␣␣␣=", x
print "xapprox=", xapprox.numerical_approx(digits=3)
```

How much does your approximation differ from the actual vector $\mathbf{x}$?

g. When we ignore the Fourier coefficients corresponding to rapidly varying basis elements, we see that the vector $\mathbf{x}$ that we reconstruct is very close to the original one. In fact, the luminance values in the approximation differ by at most one or two from the actual luminance values. Our eyes are not sensitive enough to detect this difference.

So far, we have concentrated on only one column in our 8×8 block of luminance values. Let's now consider all of the columns. The following Sage cell defines a matrix called `luminance`, which is the 8×8 matrix of luminance values. Find the 8×8 matrix F whose columns are the Fourier coefficients of the columns of luminance values.

```
luminance = matrix(8,8, [176, 170, 170, 169, 162, 160,
    155, 150, 181,
179, 175, 167, 162, 160, 154, 149, 165, 170, 169, 161,
    162, 161, 160,
158, 139, 150, 164, 166, 159, 160, 162, 163, 131, 137,
    157, 165, 163,
163, 164, 164, 131, 132, 153, 161, 167, 167, 167, 169,
    140, 142, 157,
166, 166, 166, 167, 169, 150, 152, 160, 168, 172, 170,
    168, 168])
# define your matrix F below
F =
print F.numerical_approx(digits=3)
```

h. Notice that the first row of this matrix consists of the Fourier coefficient F_0 for each of the columns. Just as we saw before, the entries in this row do not change significantly as we move across the row. In the Sage cell below, write these entries in the vector $\mathbf{y}$ and find the corresponding Fourier coefficients.

```
# define the vector y as the entries in the first row
    of F
y =
y_fourier =
print y_fourier.numerical_approx(digits=3)
```

Up to this point, we have been working with the luminance values in one 8×8 block of our image. We formed the Fourier coefficients for each of the columns of this block. Once we notice that the Fourier coefficients across a row are relatively constant, it seems reasonable to find the Fourier coefficients of the rows of the matrix of Fourier coefficients. Doing so leads to the matrix

$$\begin{bmatrix} 160.6 & -4.0 & -4.8 & -1.7 & 0.0 & 0.9 & 0.8 & 0.3 \\ 2.7 & 14.7 & 3.8 & 1.1 & -1.6 & -0.3 & -0.3 & -0.4 \\ 3.8 & 7.0 & 2.1 & 2.9 & 0.8 & -0.2 & -0.3 & -0.3 \\ -2.4 & -3.9 & -1.9 & 0.1 & 1.2 & 1.2 & 0.7 & 0.1 \\ -0.6 & -1.4 & -1.5 & -0.9 & 0.2 & 0.6 & -0.2 & -0.5 \\ -0.7 & -1.6 & 0.0 & -1.1 & 0.0 & 0.3 & -0.1 & -0.2 \\ -0.0 & -1.4 & 0.4 & 0.9 & 0.1 & -0.5 & 0.0 & 0.5 \\ 0.0 & 0.2 & 0.3 & 0.3 & 0.0 & -0.0 & -0.2 & 0.0 \end{bmatrix}.$$

If we were to look inside a JPEG image file, we would see lots of matrices like this. For each 8×8 block, there would be three matrices of Fourier coefficients of the rows of Fourier coefficients, one matrix for each of the luminance, blue chrominance, and red chrominance values. However, we store these Fourier coefficients as integers inside the JPEG file so we need to round off coefficients to the nearest integer, as shown here:

$$\begin{bmatrix} 161 & -4 & -5 & -2 & 0 & 1 & 1 & 0 \\ 3 & 15 & 4 & 1 & -2 & 0 & 0 & 0 \\ 4 & 7 & 2 & 3 & 1 & 0 & 0 & 0 \\ -2 & -4 & -2 & 0 & 1 & 1 & 1 & 0 \\ -1 & -1 & -1 & -1 & 0 & 1 & 0 & 0 \\ -1 & -2 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

There are many zeroes in this matrix, and this observation is where we can save space when creating a JPEG image file: we will only record the *nonzero* Fourier coefficients.

In fact, when a JPEG file is created, there is a "quality" parameter that can be set, such as that shown in Figure 8. When the quality parameter is high, we will store many of the Fourier coefficients; when it is low, we will ignore more of them.

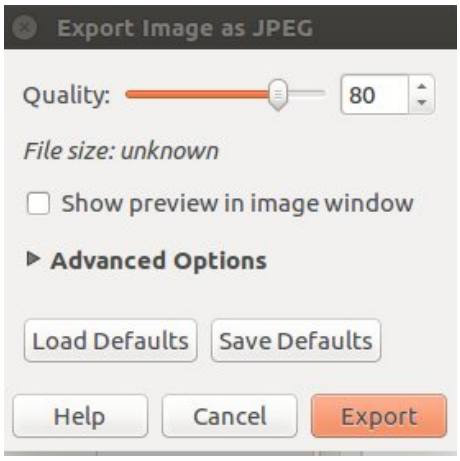


Figure 3.3.8: When creating a JPEG file, we choose a value of the "quality" parameter.

To see how this works, suppose the quality setting is relatively high. After rounding off the Fourier coefficients, we will set all of the coefficients whose absolute value is less than 2 to zero, which creates the matrix:

$$\begin{bmatrix} 161 & -4 & -5 & 0 & 0 & 0 & 0 & 0 \\ 3 & 15 & 4 & 0 & 0 & 0 & 0 & 0 \\ 4 & 7 & 2 & 3 & 0 & 0 & 0 & 0 \\ -2 & -4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Notice that there are 12 Fourier coefficients, out of 64, that we need to record. Consequently, we only record $12/64 \approx 19\%$ of the data.

If instead, the quality setting is relatively low, we set all of the Fourier coefficients whose absolute value is less than 4 to zero, creating the matrix:

$$\begin{bmatrix} 161 & -4 & -5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 15 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 7 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Notice that there are only 5 nonzero Fourier coefficients that we need to record

now, meaning we record only $5/64 \approx 8\%$ of the data. This will result in a smaller JPEG file describing the image.

With a lower quality setting, we have thrown away more information about the Fourier coefficients so the image will not be reconstructed as accurately. To see this, we can reconstruct the luminance values from the Fourier coefficients by converting back into the standard coordinate system. Rather than showing the luminance values themselves, we will show the difference in the original luminance values and the reconstructed luminance values. When the quality setting was high and we stored 12 Fourier coefficients, we find this difference to be

$$\begin{bmatrix} -7 & -7 & -1 & 3 & -2 & -1 & 0 & -1 \\ 4 & 4 & 4 & -1 & -3 & 0 & -1 & -3 \\ 1 & 3 & 0 & -7 & -3 & 1 & 3 & 3 \\ -7 & -3 & 3 & 1 & -5 & -2 & 1 & 2 \\ 0 & -3 & 4 & 4 & -1 & -1 & -1 & -2 \\ 2 & -5 & 3 & 1 & 1 & -1 & -1 & 1 \\ 1 & -2 & 4 & 3 & -4 & -6 & -2 & 3 \\ 0 & -1 & 2 & 1 & -1 & -4 & -1 & 5 \end{bmatrix}.$$

When the quality setting is lower and we store only 5 Fourier coefficients, the difference is

$$\begin{bmatrix} 3 & -3 & -2 & 0 & 0 & 7 & 10 & 10 \\ 14 & 11 & 6 & -1 & -1 & 3 & 4 & 4 \\ 7 & 10 & 5 & -5 & -3 & -1 & 2 & 3 \\ -10 & -3 & 5 & 2 & -8 & -7 & -3 & -1 \\ -12 & -11 & 2 & 2 & -5 & -7 & -6 & -6 \\ -11 & -15 & -2 & -2 & -2 & -4 & -5 & -2 \\ -3 & -6 & 2 & 3 & -2 & -5 & -4 & -1 \\ 6 & 3 & 4 & 5 & 4 & 0 & -1 & 0 \end{bmatrix}.$$

This demonstrates the trade off. With a high quality setting, we require more storage to save more of the data, but the reconstructed image is closer to the original. With the lower quality setting, we require less storage, but the reconstructed image differs more from the original.

If we remember that the visual information stored by the blue and red chrominance values is not as important as that contained in the luminance values, we feel safer in discarding more of the Fourier coefficients for the chrominance values resulting in a greater savings.

Shown in Figure 9 is the original image compared to a version stored with a very low quality setting. If you look carefully, you can individual 8×8 blocks.

Figure 3.3.9: The original image and the result of storing the image with a low quality setting.

This description of the JPEG compression algorithm is meant to convey the ideas that underlie its construction. There are a few details, most notably about the rounding of the Fourier coefficents, that are not strictly accurate. The actual implementation is a little more complicated, but the presentation here conveys the spirit of the algorithm.

We have described the JPEG compression algorithm, which allows us to store image files using only a fraction of the data. Similar ideas are used to efficiently store digital music and video files.

3.3.3 Summary

This section has explored how appropriate changes in bases help us reconstruct an image using only a fraction of its data. This is known as image compression.

- There are several ways of representing colors, all of which use vectors in $\mathbb{R}^3$. We explored the RGB color model, which is appropriate in digital applications, and the YC_bC_r model, in which the most important visual information is conveyed by the Y coordinate, known as lumninance.

- We also explored a change of basis called the Discrete Fourier Transform. In the coordinate system that results, the first coordinate measures the average of the components of a vector. Subsequent components measure deviations from the average.

- We put both of these ideas to use in demonstrating the JPEG compression algorithm. An image is broken into 8×8 blocks, and the colors into luminance, blue chrominance, and red chrominance. Applying the Discrete Fourier Transform allowed us to reconstruct a good approximation of the image using only a small number of Fourier coefficients.

3.3.4 Exercises

1. Consider the vector $\mathbf{x} = \begin{bmatrix} 103 \\ 94 \\ 91 \\ 92 \\ 103 \\ 105 \\ 105 \\ 108 \end{bmatrix}$.

a. In the Sage cell below is a copy of the change of basis matrices that define the Fourier transform. Find the Fourier coefficients of $\mathbf{x}$.

```
mat = []
for i in range(8):
    for j in range(8):
        mat.append(cos((2*i+1)*j*pi/16))
C = matrix(8,8, mat).numerical_approx()
Cinv = C.inverse()
print Cinv.numerical_approx(digits=3)
```

b. We will now form the vector $\mathbf{y}$, which is an approximation of $\mathbf{x}$. To do this, round all the Fourier coefficients of $\mathbf{x}$ to the nearest integer to obtain $\{\mathbf{y}\}_{\mathcal{C}}$. If a coefficient has an absolute value less than one, set it equal to zero. Now find the vector $\mathbf{y}$ and compare this approximation to $\mathbf{x}$. What is the error in this approximation?

c. Repeat the last part of this problem, but set the rounded Fourier coefficients to zero if they have an absolute value less than five. Use it to create a second approximation of $\mathbf{x}$. What is the error in this approximation?

d. Compare the number of nonzero Fourier coefficients that you have in the two approximations and compare the accuracy of the approximations. Using a few sentences, discuss the comparisons that you find.

2. There are several steps to the JPEG compression algorithm. The following questions examine the motivation behind some of them.

a. What is the overall goal of the JPEG compression algorithm?

b. Why do we convert colors from the the RGB color model to the YC_bC_r model?

c. Why do we decompose the image into a collection of 8×8 arrays of pixels?

d. What role does the Discrete Fourier Transform play in the JPEG compression algorithm?

e. Why is the information conveyed by the rapid-variation Fourier coefficients, generally speaking, less important than the slow-variation coefficients?

3. The Fourier transform that we used in this section is often called the Discrete Fourier Cosine Transform because it is defined using a basis $\mathcal{C}$ consisting of cosine functions. There is also a Fourier Sine Transform defined using a basis $\mathcal{S}$ consisting of sine functions. For instance, in $\mathbb{R}^4$, the basis vectors of $\mathcal{S}$

are

$$\mathbf{v}_1 = \begin{bmatrix} \sin\left(\frac{1\cdot 1\pi}{8}\right) \\ \sin\left(\frac{3\cdot 1\pi}{8}\right) \\ \sin\left(\frac{5\cdot 1\pi}{8}\right) \\ \sin\left(\frac{7\cdot 1\pi}{8}\right) \end{bmatrix}, \mathbf{v}_2 = \begin{bmatrix} \sin\left(\frac{1\cdot 2\pi}{8}\right) \\ \sin\left(\frac{3\cdot 2\pi}{8}\right) \\ \sin\left(\frac{5\cdot 2\pi}{8}\right) \\ \sin\left(\frac{7\cdot 2\pi}{8}\right) \end{bmatrix},$$

$$\mathbf{v}_3 = \begin{bmatrix} \sin\left(\frac{1\cdot 3\pi}{8}\right) \\ \sin\left(\frac{3\cdot 3\pi}{8}\right) \\ \sin\left(\frac{5\cdot 3\pi}{8}\right) \\ \sin\left(\frac{7\cdot 3\pi}{8}\right) \end{bmatrix}, \mathbf{v}_4 = \begin{bmatrix} \sin\left(\frac{1\cdot 4\pi}{8}\right) \\ \sin\left(\frac{3\cdot 4\pi}{8}\right) \\ \sin\left(\frac{5\cdot 4\pi}{8}\right) \\ \sin\left(\frac{7\cdot 4\pi}{8}\right) \end{bmatrix}.$$

We can think of these vectors graphically, as shown in Figure 10.

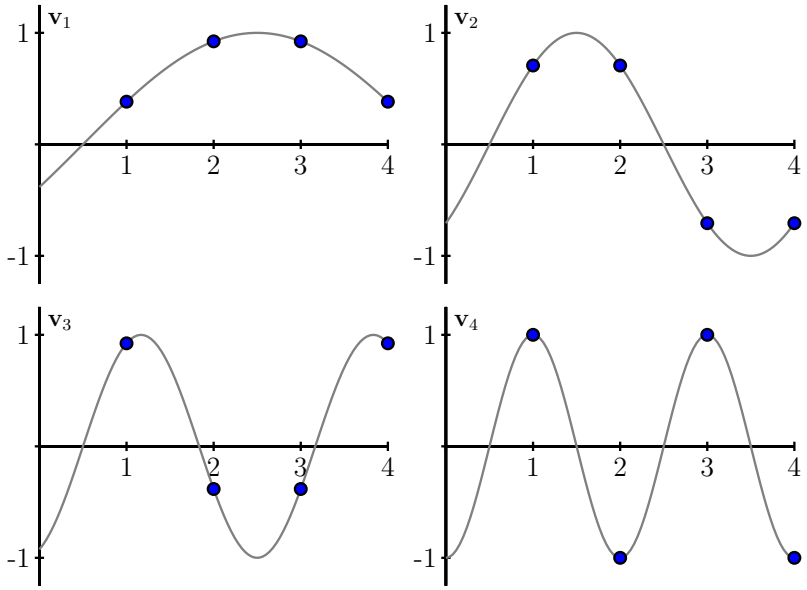


Figure 3.3.10: The vectors $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4$ that form the basis $\mathcal{S}$.

a. The Sage cell below defines the matrix `S` whose columns are the vectors in the basis $\mathcal{S}$ as well as the matrix `C` whose columns form the basis $\mathcal{C}$ used in the Fourier Cosine Transform.

```
sinmat = []
cosmat = []
for i in range(4):
    for j in range(1,5):
        sinmat.append(sin((2*i+1)*j*pi/8.0))
        cosmat.append(cos((2*i+1)*(j-1)*pi/8.0))
S = matrix(4,4,sinmat).numerical_approx()
C = matrix(4,4,cosmat).numerical_approx()
```

In the 8×8 block of luminance values we considered in this section, the first column begins with the four entries 176, 181, 165, and 139, as seen in

Figure 6. These form the vector $\mathbf{x} = \begin{bmatrix} 176 \\ 181 \\ 165 \\ 139 \end{bmatrix}$. Find both $\{\mathbf{x}\}_{\mathcal{S}}$ and $\{\mathbf{x}\}_{\mathcal{C}}$.

b. Write a sentence or two comparing the values for the Fourier Sine coefficients $\{\mathbf{x}\}_{\mathcal{S}}$ and the Fourier Cosine coefficients $\{\mathbf{x}\}_{\mathcal{C}}$.

c. Suppose now that $\mathbf{x} = \begin{bmatrix} 100 \\ 100 \\ 100 \\ 100 \end{bmatrix}$. Find the Fourier Sine coefficients $\{\mathbf{x}\}_{\mathcal{S}}$ and the Fourie Cosine coefficients $\{\mathbf{x}\}_{\mathcal{C}}$.

d. Write a few sentences explaining why we use the Fourier Cosine Transform in the JPEG compression algorithm rather than the Fourier Sine Transform.

4. In Example 3.2.7, we looked at a basis for $\mathbb{R}^4$ that we called the Haar wavelet basis. The basis vectors are

$$\mathbf{v}_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \mathbf{v}_2 = \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix}, \mathbf{v}_3 = \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0 \end{bmatrix}, \mathbf{v}_4 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ -1 \end{bmatrix},$$

which may be understood graphically as in Figure 11. We will denote this basis by $\mathcal{W}$.

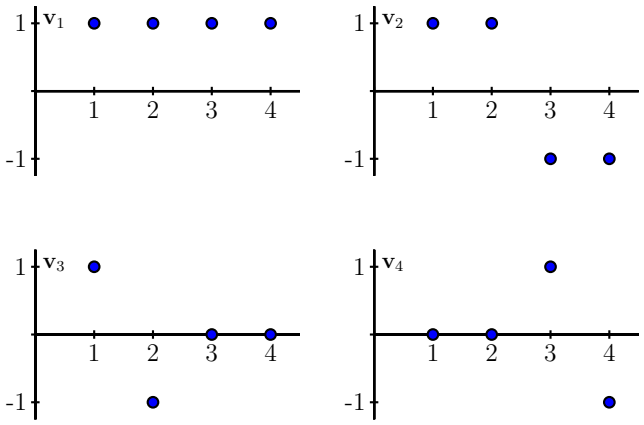


Figure 3.3.11: The Haar wavelet basis represented graphically.

The change of coordinates from a vector $\mathbf{x}$ in $\mathbb{R}^4$ to $\{\mathbf{x}\}_{\mathcal{W}}$ is called the Haar *wavelet transform* and we write

$$\{\mathbf{x}\}_{\mathcal{W}} = \begin{bmatrix} H_1 \\ H_2 \\ H_3 \\ H_4 \end{bmatrix}.$$

The coefficients H_1, H_2, H_3, H_4 are called wavelet coefficients.

Let's work with the 4×4 block of luminance values in the upper left corner of our larger 8×8 block:

$$\begin{bmatrix} 176 & 170 & 170 & 169 \\ 181 & 179 & 175 & 167 \\ 165 & 170 & 169 & 161 \\ 139 & 150 & 164 & 166 \end{bmatrix}.$$

a. The following Sage cell defines the matrix `W` whose columns are the basis vectors in $\mathcal{W}$. If $\mathbf{x}$ is the first column of luminance values in the 4×4 block above, find the wavelet coefficients $\{\mathbf{x}\}_{\mathcal{W}}$.

```
W = matrix(4,4,[1,1,1,0,1,1,-1,0,1,-1,0,1,1,-1,0,-1])
```

b. Notice that H_1 gives the average value of the components of $\mathbf{x}$ and H_2 describes how the averages of the first two and last two components differ from the overall average. The coefficients H_3 and H_4 describe small-scale variations between the first two components and last two components, respectively.

   If we set the last wavelet coefficients $H_3 = 0$ and $H_4 = 0$, we obtain the wavelet coefficients $\{\mathbf{y}\}_{\mathcal{W}}$ for a vector $\mathbf{y}$ that approximates $\mathbf{x}$. Find the vector $\mathbf{y}$ and compare it to the original vector $\mathbf{x}$.

c. What impact does the fact that $H_3 = 0$ and $H_4 = 0$ have on the form of the vector $\mathbf{y}$? Explain how setting these coefficients to zero ignores the behavior of $\mathbf{x}$ on a small scale.

d. In the JPEG compression algorithm, we looked at the Fourier coefficients of all the columns of luminance values and then performed a Fourier transform on the rows. The Sage cell below will perform the same operation using the wavelet transform; that is, it will first find the wavelet coefficients of each of the columns and then perform the wavelet transform on the rows. You only need to evaluate the cell to find the wavelet coefficients obtained in this way.

```
luminance = matrix(4,4,[176, 170, 170, 169, 181, 179,
    175, 167, 165,
170, 169, 161, 139, 150,  164, 166])
Winv = W.inverse()
wavelet_transform =
    (Winv*(Winv*luminance).transpose()).transpose()
print wavelet_transform.numerical_approx(digits=3)
```

e. Now set all the wavelet coefficients equal to zero except those in the upper left 2×2 block and use them to define the matrix `coeffs` in the Sage cell below. This has the effect of ignoring all of the small-scale differences. Evaluating this cell will recover the approximate luminance values.

```
# define the matrix of coefficients below
coeffs =
# this code will undo the wavelet transform
approx_luminance =
    W*((W*(coeffs.transpose())).transpose())
print approx_luminance.numerical_approx(digits=3)
```

f. Explain how the wavelet transform and this approximation can be used to create a lower resolution version of the image.

This kind of wavelet transform is the basis of the JPEG 2000 compression algorithm, which is an alternative to the usual JPEG algorithm.

5. In this section, we looked at the RGB and YC_bC_r color models. In this exercise, we will look at the HSV color model where H is the hue, S is the saturation, and V is the value of the color. All three quantities vary between 0 and 255.
The diagram available at the bottom of http://gvsu.edu/s/0Jc enables you to vary the three parameters H, S, and V in the HSV color model.

a. If you leave S and V at some fixed values, what happens when you change the value of H?

b. Move the value V to the right and keep it fixed. Describe what happens when you vary the saturation S using a fixed hue H and value V.

c. Describe what happens when H and S are fixed and V varies.

d. How can you create white in this color model?

e. How can you create black in this color model?

f. Find an approximate range of hues that correspond to blue.

g. Find an approximate range of hues that correspond to green.

The YC_bC_r color model concentrates the most important visual information in the luminance coordinate, which roughly measures the brightness of the color. The other two coordinates describe the hue of the color. By contrast, the HSV color model concentrates all the information about the hue in the H coordinate.
This is useful in computer vision applications. For instance, if we want a robot to detect a blue ball in its field of vision, we can specify a range of hue values to search for. If the lighting changes in the room, the saturation and value may change, but the hue will not. This increases the likelihood that the robot will still detect the blue ball across a wide range of lighting conditions.

3.4 Determinants

In this chapter, we have been concerned with bases and related questions about the invertibility of square matrices. We saw that a square matrix is invertible if and only if it is row equivalent to the identity matrix. In this section, we will develop a numerical criterion that tells us whether a square matrix is invertible. This criterion will prove useful in the next chapter.

To begin, let's consider a 2×2 matrix A whose columns are vectors $\mathbf{v}_1$ and $\mathbf{v}_2$. We have frequently drawn the vectors and considered the linear combinations they form through a figure such as Figure 1.