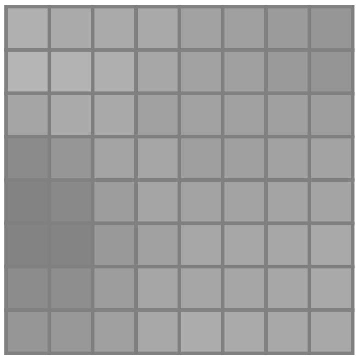
Mathematics 227
Lab 3, Due: October 31, 2018

Instructions: The exercises here should be completed in groups of 2 or 3 students with one write-up submitted from each group.

The JPEG compression algorithm begins by breaking an image into 8×8 blocks of pixels. Each pixel is represented as a color using three coordinates in the luminance-chrominance color model (YC_bC_r). For instance, here are the luminance values in one 8×8 block.



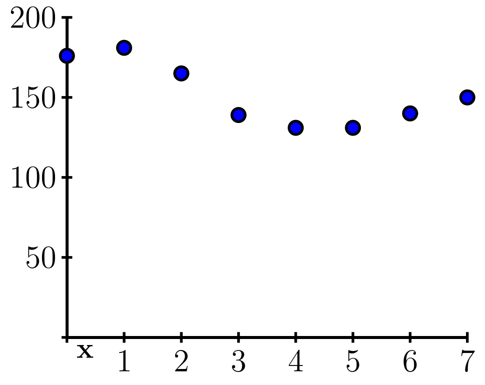
176	170	170	169	162	160	155	150
181	179	175	167	162	160	154	149
165	170	169	161	162	161	160	158
139	150	164	166	159	160	162	163
131	137	157	165	163	163	164	164
131	132	153	161	167	167	167	169
140	142	157	166	166	166	167	169
150	152	160	168	172	170	168	168

There are 64 values here, each represented by 8 bytes of computer memory. The compression algorithm works by finding a way of representing this information using only a fraction of the 64 values.

1. Let's look at the entries in the first column, which can be represented as a vector in $\mathbb{R}^8$:

$$\mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} = \begin{bmatrix} 176 \\ 181 \\ 165 \\ 139 \\ 131 \\ 131 \\ 140 \\ 150 \end{bmatrix}.$$

We can represent this graphically as:



Two things are important:

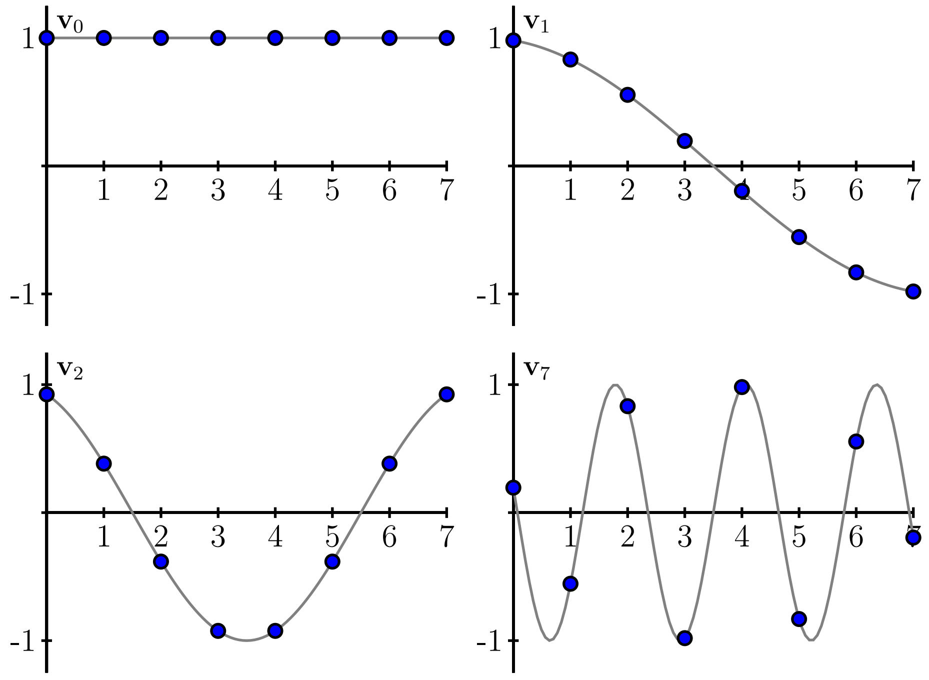
- There are not wide variations in these numbers across the column.
- If some of the values were to change slightly, our eyes would probably not notice because the pixels, when represented on a screen, are so small.

This suggests that we might be able to store the information more compactly using a different basis.

The *Fourier transform* uses a new basis $\mathcal{C}$ for $\mathbb{R}^8$ using the vectors

$$\mathbf{v}_0 = \begin{bmatrix} \cos\left(\frac{(2\cdot0+1)0\pi}{16}\right) \\ \cos\left(\frac{(2\cdot1+1)0\pi}{16}\right) \\ \cos\left(\frac{(2\cdot2+1)0\pi}{16}\right) \\ \vdots \\ \cos\left(\frac{(2\cdot7+1)0\pi}{16}\right) \end{bmatrix}, \mathbf{v}_1 = \begin{bmatrix} \cos\left(\frac{(2\cdot0+1)1\pi}{16}\right) \\ \cos\left(\frac{(2\cdot1+1)1\pi}{16}\right) \\ \cos\left(\frac{(2\cdot2+1)1\pi}{16}\right) \\ \vdots \\ \cos\left(\frac{(2\cdot7+1)1\pi}{16}\right) \end{bmatrix}, \ldots, \mathbf{v}_7 = \begin{bmatrix} \cos\left(\frac{(2\cdot0+1)7\pi}{16}\right) \\ \cos\left(\frac{(2\cdot1+1)7\pi}{16}\right) \\ \cos\left(\frac{(2\cdot2+1)7\pi}{16}\right) \\ \vdots \\ \cos\left(\frac{(2\cdot7+1)7\pi}{16}\right) \end{bmatrix}$$

OK, this looks a little scary, but we can understand the vectors by looking at them graphically:



Notice that $\mathbf{v}_0$ is constant, $\mathbf{v}_1$ varies slowly, $\mathbf{v}_2$ varies a little faster, and $\mathbf{v}_7$ varies quite rapidly.

Representing a vector in these coordinates, we obtain

$$\{\mathbf{x}\}_{\mathcal{C}} = \mathbf{f} = \begin{bmatrix} F_0 \\ F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \\ F_6 \\ F_7 \end{bmatrix},$$

where components of $\mathbf{f}$ are called the *Fourier coefficients* of the vector $\mathbf{x}$. It turns out that F_0 is the average of the components of $\mathbf{x}$.

Visit the page `http://gvsu.edu/s/0Jd` where you can see the effect of changing three Fourier coefficients.

What is the effect of changing F_0?

How is the effect of changing F_3 different from the effect of changing F_7?

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

2. Now go to the page `gvsu.edu/s/0RF`, which is a page of Sage cells. Evaluating the first cell will load some data and useful commands that are relevant to this lab. After you evaluate the first cell, don't reuse it again.

The first cell defines a matrix `C` composed of the basis vectors $\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_7$ and its inverse `Cinv`. The vector `x` given above is also defined. Find the vector $\mathbf{f}$ of Fourier coefficients of $\mathbf{x}$.

What do you notice about the relative size of the Fourier coefficients?

Form an approximation $\mathbf{f}_{\text{approx}}$ to $\mathbf{f}$ by rounding the Fourier coefficients to the nearest integer and then setting F_6 and F_7 to 0.

Now find the vector $\mathbf{x}_{\text{approx}}$ such that $\{\mathbf{x}_{\text{approx}}\}_{\mathcal{C}} = \mathbf{f}_{\text{approx}}$.

Compare this vector to the original vector $\mathbf{x}$. What is the largest difference in the components of $\mathbf{x}_{\text{approx}}$ and $\mathbf{x}$?

This is the central idea behind the compression algorithm: we have thrown away two of the eight Fourier coefficients so that we save only 75% of the Fourier data. When we reconstruct $\mathbf{x}$, we have a very good approximation using only 75% of the data.

3. Let's return to our 8×8 block of luminance values. Each column of luminance values forms a vector in $\mathbb{R}^8$. Convert each column into its corresponding vector of Fourier coefficients and record the top row, consisting of the F_0 components, below.

   Notice that the Fourier coefficients also don't change very much as you move across a row. The next step in the compression algorithm is to view each row as a vector in $\mathbb{R}^8$ and represent that vector in terms of its Fourier coefficients. This means that you are computing the Fourier coefficients of the Fourier coefficients.

   You can use the function `double_Fourier_transform(luminance)` to do this. Note the values in the upper-left 2×2 block below.

   Next, we will throw away some of the data contained in the Fourier coefficients. Set any entry in this matrix less than 2 to 0 and round the rest of them to the nearest integer. You can perform this operation using the function

   `round_matrix(double_Fourier_transform(luminance), 2)`. Record the upper 2×2 block below and count how many nonzero entries there are in this matrix.

   You started with 64 pieces of data and now you are left with a much smaller number. What fraction of the data have you thrown away? This is the compression ratio.

If you call the rounded matrix `ft_approx`, you can recover the approximation to the luminance values with `inverse_double_Fourier_transform(ft_approx)`. Write the four approximate luminance values from the upper-left 2×2 corner and compare them to the original luminance values. By how much do they differ?

Compare the original luminance values to the approximated luminance values. What is the largest difference that you find (Sage can help you do this)?

Once again, this is the idea behind the compression algorithm. We are saving only a small fraction of the original data yet we can reconstruct the colors with a fairly small error, which is probably less than our eyes would notice anyway.

Above, we set the Fourier coefficients to 0 if they were below 2. If you choose a higher number, more of the Fourier coefficients will be set to 0. This means you will be saving a smaller amount of data (the compression ratio goes up), but the errors in the reconstruction will be higher.