Name: _____

EE 193-03 Homework 2

Due via `provide` at 11:59pm, 1 October 2018

Submission

Zip up your code (i.e., the "code" subdirectory of the starter code package) and submit it via provide from one of the Linux machines:

```
provide ee193HIP hw2 <ZIP FILE>
```

You can also use the web interface at `http://www.ece.tufts.edu/ee/193HIP/provide.cgi`.

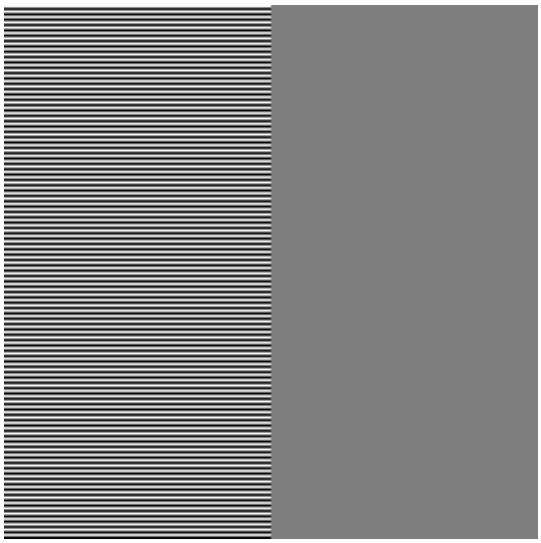
Please don't submit the data files. They're big, and I have those already.

Problem 1: Experimenting with gamma

Normally it's difficult to observe the gamma mapping that your computer screen is doing, because it all happens "magically" behind the scenes, and you don't have an easy way to measure the true light output of the screen.

However, if you create alternating rows of pixels with different gray levels and step back from your screen, the lines blur together to produce a gray which is the average luminance of the two. We can compare this to a solid gray, and find the value with matching luminance.

Write a MATLAB script in that creates a 200x200 pixel image like the one below. The left half should be alternating rows of gray values, while the right should be solid with an intermediate value.



Then use your script to find the sRGB gray value that matches the luminance of the average of 0 and 255 (#000000 and #FFFFFF). *Hint: it's not 127.*

You'll want to use `imshow` for this, since other methods of displaying an image don't match one screen pixel to one data pixel.

Problem 2: Cone responses

Given a multispectral image of a scene and an illuminant, compute the cone responses across the image. The file `coneresponse.m` contains the starter code for the problem, loading the multispectral image, the illuminant spectra, and the cone responses.

First, compute the image of cone responses under D65 illumination and display it with a gamma of 2.4. The colors won't look right, since we're displaying the cone responses in the RGB channels, not properly converted RGB values (we'll do that in the next HW). However, the colors should roughly match those shown on the Metacow page (`https://www.rit.edu/science/munsell-color-lab?ref=rit-search#resources`, under "research databases"). For example, the yellow cow should at least look yellow-ish, and the blue cow should be blue-ish.

Second, repeat the calculation using the `light_fluorescent` illumination. The (lack of) metamerism should be obvious.

Problem 3: Make your own meta-cow

Construct your own "reverse metacow", which is metameric under fluorescent illumination but appears different under D65. That is, you should design two reflectance spectra (light reflected as a function of wavelength) which give the same cone responses under fluorescent illumination but different responses under D65.

Hint: It may help to start by writing out the system of equations and thinking about what quantities should be equal (and not equal). You'll have lots of free variables, so you don't necessarily need them all. More direct approaches include finding the nullspace of a matrix or formulating a linear programming problem.

Plot your two reflectance spectra, and include a brief paragraph in a comment at the top of your script explaining how you chose these spectra.

Problem 4: Color matching

Suppose you have a display with three primaries, with spectra given in the variable `display_primaries`.

For each of the four cone response sets (stored as column vectors in `tristimulus`), find the intensity for the display primaries that would produce that response, or indicate that it is not possible (i.e., out of gamut). You should put your answers in a comment at the top of `colormatching.m`.

Again, we can do this more properly (and with meaningful units) using the XYZ color space, but that's for later.