ECEn 671: Mathematics of Signals & Systems

Fall 2011

Homework #3

Problem 1:

Do the following in Matlab. Print both your code and any requested figures, and hand them in with your homework.

a. Write a script that numerically (***not*** analytically or symbolically) generates the first 5 Legendre polynomials by using Gram-Schmidt orthogonalization on the set of vectors $\{1, t, t^2, t^3, t^4\}$ over the interval [-1, 1]. **Plot each of these polynomials for $-1 \le t \le 1$.**

Note: Use enough accuracy for each polynomial vector that your numerical inner products are reasonably accurate. The following code might help you get started…

```
t = linspace(-1, 1, 10000);
p = zeros(10000, 5);
for kk = 1:5
      p(:, kk) = t.^(kk-1);
end
```

b. Perform a linear least-squares approximation of the function $f(t) = e^{-t}$ on the interval [-1, 1] using the Legendre polynomials derived in part 1. **Plot $f(t)$ and your approximation (or expansion) of $f(t)$. Find the norm of the error vector.** That is, find the norm of the difference between $f(t)$ and your approximation of $f(t)$.

c. Modify your results from (a) to generate the first 5 Chebyshev polynomials on the interval [-1, 1]. *Refer to Example 2.15.1 on p. 120 of the book, and note that generation of the Chebyshev polynomials will be identical to the Legendre polynomials but with the use of a weighted inner product.* **Plot each of these polynomials for $-1 \le t \le 1$.**

d. Repeat part (b) using the Chebyshev polynomials instead of the Legendre polynomials.

e. Compare the norm of the error vector from part (b) with that from part (d). **Comment on your answer.**

Problem 2:

Consider the application of some of the math we have developed to image processing. For simplicity, let's deal with simple black and white images at a resolution of 256 x 256 pixels. In order to deal with these images using the vector concepts we have developed, we can "flatten" each image into a single column vector with a length of 256 x 256 = 65,536. Effectively, any 256 x 256 black and white image is then a vector in a 65,536-dimensional space.

Many lossy image compression techniques rely on the "projection" of an image onto a subspace spanned by a basis set of far less than 65,536 images. If we choose our basis set well, it turns out we can get very faithful reproductions of an image by projecting it onto a space of far less than 65,536 dimensions. Wavelet transforms, discrete cosine transforms (DCTs), and simple 2D discrete Fourier transforms are all tools to transform images into representations in alternate basis sets. Often, the representation of the image in these alternate basis sets is "sparse"; that is, much of the image energy is concentrated in a limited set of coefficients (or pixels in the transformed image), with many other coefficients (or pixels) near zero. Lossy compression techniques typically store a carefully chosen subset of the most important coefficients (pixels) in the transformed image, yielding a reasonable approximation (as far as the human eye is concerned) of the uncompressed image.

While we aren't going to deal with image compression in this problem, we are going to get some practice projecting an image onto a very small basis set. ☺

Exercise: Download the files prob2.mat and prob2.m posted on the website. Open Matlab, and run the file prob2.m. A set of 20 black and white images (256 x 256 pixels) is loaded into the 256 x 256 x 20 matrix $\boldsymbol{A}$. One other image (which won't look like much of anything) is loaded into the 256 x 256 matrix $\boldsymbol{x}$.

The image $\boldsymbol{x}$ was formed by taking a random linear combination of the 20 basis images. A 21st image (not in the basis set) was then "encoded" in the image by adding it to the random linear combination image. Your job is to recover (as best you can) the encoded 21st image. **Print the recovered image.**

You'll notice that the quality of the recovered image is quite low. When added to the original linear combination image, the image quality was quite high. **Describe what accounts for the loss in image fidelity of your recovered image.** (You don't need the original image to figure this out: just take my word for it that the image quality has suffered, and figure out why.)