HW 2: Bases, Projections, Filtering and the FFT
Due Thursday Sept. 5 at class

The vectors

$$\mathbf{u}_k = 1/\sqrt{N}\begin{bmatrix} 1 \\ e^{-j2\pi\frac{k}{N}} \\ \vdots \\ e^{-j2\pi\frac{k}{N}(N-1)} \end{bmatrix}$$

are the discrete-time complex sinusoids used in the DFT and FFT (Recall that the FFT is simply an efficient algorithm for computing the DFT). The DFT coefficients of an N point signal $x(n), n = 0, \ldots, N-1$ are computed according to

$$X(k) = \frac{1}{\sqrt{N}}\sum_{n=0}^{N-1} x(n)e^{-j2\pi\frac{k}{N}n}, \quad k = 0, \ldots, N-1.$$

If we define

$$\mathbf{x} = \begin{bmatrix} x(0) \\ x(1) \\ \vdots \\ x(N-1) \end{bmatrix},$$

then we can express this computation in vector notation as the inner product

$$X(k) = \mathbf{x}^T\mathbf{u}_k.$$

a. Show that the DFT vectors $\{\mathbf{u}_k\}_{k=0}^{N-1}$ form an orthonormal basis for C^N (space of N-dimensional vectors with complex-valued elements). (HINT: First show that the DFT vectors are orthonormal, then show verify that every $\mathbf{x} \in C^N$ can be expressed as a linear combination of $\{\mathbf{u}_k\}_{k=0}^{N-1}$.)

b. Suppose that we are studying signals of the following form

$$x(n) = \cos(2\pi f n + \phi), \quad n = 0, \ldots, N-1,$$

where $-L/N \leq f \leq L/N$, where $L < \frac{N}{2}$, and the phase ϕ is arbitrary. In terms DFT basis vectors, describe the approximate signal subspace $\langle \mathbf{H} \rangle$ in which such signals reside.

c. Now suppose we observe

$$\mathbf{y} = \mathbf{x} + \mathbf{n},$$

where $\mathbf{n}$ is a vector of white noise. Design a projection matrix that is suitable for reducing the noise in the observation $\mathbf{y}$ without degrading the signal. That is, design a projection matrix $\mathbf{P}_H$ so that $\hat{\mathbf{x}} \equiv \mathbf{P}_H\mathbf{y}$ is a good estimate of $\mathbf{x}$. Can you interpret the projection operation as a frequency domain filter?

d. Implement the projection/filtering scheme developed above in Matlab. Test the performance of your algorithm by generating random signals and noise using the following Matlab code:

```
N = 256;
L = 8;
x = cos(2*pi*(2*rand-1)*(L/N)*(0:(N-1))'+pi*rand);
y = x + 0.5 * randn(N,1);
```

Also, devise an alternative implementation using Matlab's built-in commands **fft** and **ifft**. Comment on the efficiency of the projection matrix implementation relative to the fft-based implementation.