Math 314 – MATLAB Exercise 2 Name:_____
Due: Friday, November 7, 2014

The Fourier Basis

In this exercise you will be introduced to a particularly useful basis for the vector space $\mathbb{R}^n$ that is *not* the standard basis. This basis and variations on it have many applications in science and technology: the ideas you will be exposed to are behind such diverse applications as the JPEG format for storing digital images, the analyses of experimental data collected over time for cyclical behavior, and some methods of removing noise from signals.

In what follows you will perform some steps to analyze some data that you should pretend came from an experiment. (Several possible sources of the data will be given below.) While the source of the data could be almost anything, the techniques you will use to analyze it will be from linear algebra.

As you work through the numbered items below, write answers to any questions directly on this handout.

1. After starting MATLAB, use a web browser to access the course web page

   `http://www.dms.uaf.edu/~jrhodes/M314.html`

   and next to the 10/22 homework assignment click on the link for the file `fourier.m`. Copy and paste the entire contents of this file into the MATLAB session window. This should create 4 vectors of data for you, called `data7`, `data256`, `data256b`, and `data256c`. Verify that they have been stored with these names by using the command `who` to see a list of all stored variables.

   We will be interested mostly in `data256`, but since it has so many entries, `data7` has been provided as a simpler example.

   In MATLAB, enter the commands `data`N first for $N = 7$ and then for $N = 256$. These are your data vectors and their entries represent measurements taken at either 7 or 256 different times. For instance, the numbers might be 1) successive measurements of voltage taken by an EKG device at a number of equally spaced times, 2) the air pressure recorded by a microphone as a bird chirped, 3) numbers recorded by a seismograph that track the vibrations of the earth during an earthquake, 4) the price of some stock on successive days, or 5) a signal that must be sent down a telephone line to transmit a message.

2. In order to visualize the data, enter `plot(data`N`)` for $N = 7$ and 256. MATLAB will plot the points (i, x_i), where x_i is the ith entry in the data vector, and then connect these points with straight lines. Be sure you understand you are *not* drawing the data vector as an arrow in $\mathbb{R}^N$.

Instead, you are thinking of the position within the vector as representing time, and the entry in that position as representing a function value at that time.

Sketch the plots produced by MATLAB here:

3. The data vector is a vector in $\mathbb{R}^N$, so regardless of where you think your data is coming from, it is natural to try to use your understanding of $\mathbb{R}^N$ to understand the data. In particular, we know $\mathbb{R}^N$ has dimension N, and any basis will have N elements. The obvious basis is the standard one. Enter `I=eye(N)` to generate a matrix with the standard basis as columns. Plot the first column by entering `plot(I(:,1))`, the second by `plot(I(:,2))`, etc. Plot all the columns at once with `plot(I)`. (Again, be sure you understand that you are *not* drawing the basis vectors as arrows in $\mathbb{R}^N$, but visualizing them as if they specified functions of time.)

   Sketch the plot that would be produced by $\mathbf{e}_{37} \in \mathbb{R}^{256}$ here:

4. Write `data7` as a linear combination of the standard basis vectors in the space below. Does this give you any insight? (Be honest!) Would you learn anything from writing `data256` in terms of the standard basis? Explain.

5. Now we'll develop a different basis that will give us more insight into the data. The idea (originally due to Joseph Fourier in 1822, in a slightly different form) is to use cosine waves. Since the plot of the data shows

lots of oscillations, this is at least vaguely reasonable.

To produce the cosine waves, we first need to generate a vector `s`N of N equally spaced numbers between 0 and 1 (actually, from $\frac{1}{2N}$ to $1 - \frac{1}{2N}$). For $N = 7$, enter the following MATLAB commands and explain what each does.

```
[0:6]'
(1/7)*[0:6]'
s7=1/(2*7)+(1/7)*[0:6]'
```

6. You have already created `s7`, so now create `s256` in a similar manner. Write down the MATLAB command you used.

7. For each i from 0 to $N-1$, we will define a vector `B`i by entering a command like `B`i`=cos(`i`*pi*s`N`)`. With $N = 7$, do this first for `B0`, and then for `B1`.

   Visualize these vectors by entering `plot(B0)` and `plot(B1)`. If you think of your data vector as giving N values of some data function at N successive times, then `B0` corresponds to the constant function whose values are 1, and `B1` corresponds to the cosine function (between 0 and π). Sketch the plots here.

8. Still with $N = 7$, enter `B2=cos(2*pi*s`N`)` and then plot it also. Then enter `B3=cos(3*pi*s`N`)` and plot it. Continue on this way until you've generated the N vectors `B0`, `B1`, `B2`, `B3`, ... `B`$(N-1)$. Each vector `B`k, then, is a vector in $\mathbb{R}^N$ but should be thought of as representing part of a cosine wave. The larger k is, the more oscillations `B`k has. k is referred to as the *frequency* of `B`k.

   Why is it that if k is small (say 1,2, or 3), the plot of `B`k closely resembles a cosine wave, but for larger k (close to N), `B`k looks more jagged?

9. For $N = 7$, you now have the N vectors `B0`, `B1`, `B2`, ..., `B`$(N-1)$ in $\mathbb{R}^N$, but you don't yet know that they are a basis. Put them into the columns of a matrix by entering `Q`N`=[B0 B1 B2 B3 ... B`$(N-1)$`]` and do Gaussian elimination with the `rref` command to show that they are a basis. Write the output of the `rref` command here, and explain why this shows the vectors are a basis.

10. For $N = 256$ you will find it very time consuming to produce `Q`N in the above way. A shorter way to create `Q`N is to enter

$$\texttt{Q}N = \texttt{cos}(\texttt{pi} * \texttt{s256} * [\texttt{0} : N-\texttt{1}]).$$

    You can then pick off individual columns of `Q`N with commands like `Q`N`(:,1)` to get the first column.

    Explain how the command `Q`N`=cos(pi*s256*[0:`$N-1$`])`works to produce `Q`N correctly.

11. Verify that the columns of `Q256` are a basis for $\mathbb{R}^{256}$. Since the output of `rref`(`Q256`) is hard to look at, explain why it is enough to enter the commands

```
U=rref(Q256);
U(256,256)
```

    and only look at one entry of U.

12. You can plot all columns of `Q256` at once with the command `plot(Q256)`. However, that is a mess to look at, so instead just plot a few of the columns of the matrix, say columns 1, 5, 20, and 240. Sketch the plots here.

13. The Fourier basis is not just a basis, but has an additional nice property. With $N = 7$, compute the product $\mathtt{B}i^T\mathtt{B}j$ for a few different choice of i and j. What is $\mathtt{B}i^T\mathtt{B}j$ if $i \neq j$? What is $\mathtt{B}i^T\mathtt{B}j$ if $i = j$?

14. Computing $\mathtt{Q7}^T\mathtt{Q7}$ is a fast way to compute all the products needed in the last question at once. Explain why.

15. Compute $\mathtt{Q7}^T\mathtt{Q7}$ in MATLAB and write it here.

16. The last computation suggests that the inverse of `Q7` is 'nearly' $\mathtt{Q7}^T$. We will now modify `Q7` a bit to make this exactly true. (If we can do this, we will know the inverse of this matrix *without* having to do any additional work — for a large matrix this is a huge savings in effort over computing the inverse by any general method.)

Note that the diagonal entries of the product $\texttt{Q7}^T\texttt{Q7}$ come from computing $\texttt{B}i^T\texttt{B}i$. If we replace $\texttt{B}i$ by $\texttt{cB}i$ for some scalar c, how will it change the diagonal entry?

17. What scalar should each column be multiplied by so that the diagonal entries in the product $\texttt{Q7}^T\texttt{Q7}$ turn out to be 1? (Warning: one column behaves differently from all the others.)

18. Why will changing the columns as in the last question not change the off-diagonal entries of $\texttt{Q7}^T\texttt{Q7}$?

19. Before going on, make sure you change *both* the Bks and `Q7` as so that $\texttt{Q7}^T\texttt{Q7} = I$. (To multiple all columns of `Q7` by a scalar `c`, and save this as the new `Q7`, you can enter `Q7=c*Q7`. To multiple a single column, say the ith, by `c`, use `Q7(:,`i`)= c*Q7(:,`i`)`.)

    Replot a few of the Bks by entering `plot(B0), plot(B1),` etc. to see how you've changed them. Sketch the plots here, explaining the changes from what you produced in questions 7 and 8.

20. Similarly modify `Q256` so that $\texttt{Q256}^T\texttt{Q256} = I$. Do it. Record here what you multiplied the columns by to achieve this.

21. We can now see that the columns of our modified $\mathtt{Q}N$ form a basis for $\mathbb{R}^N$ without doing elimination. Since $\mathtt{Q}N$ has an inverse, how many pivots must it have? Why does this show the columns form a basis?

22. Now that we've developed the tool, we can begin our analysis of the data. We want to begin by expressing $\mathtt{data}N$ in terms of the Fourier basis given by the columns of $\mathtt{Q}N$. (In other words, we're redoing what we did in question 4 but using our new basis.) First explain why doing this requires that we solve $\mathtt{Q}N\,\mathbf{c} = \mathtt{data}N$ for $\mathbf{c}$. Then explain why we know there will be one and only one solution to this equation.

23. There are two ways we can get the solution.

    One is simply Gaussian elimination using the MATLAB command $\mathtt{Q}N\backslash\mathtt{data}N$. Do it, for $N = 7$ and 256.

24. The other is to take advantage of the fact that we know $\mathtt{Q}N^{-1} = \mathtt{Q}N^T$ already. Therefore $\mathbf{c} = \mathtt{Q}N^T\mathtt{data}N$. Enter the appropriate MATLAB code for this and make sure the solutions agree with what you produced in the last question.

25. Which of the two methods of finding $\mathbf{c}$ requires less work for the computer? Think about doing the work by hand to decide.

26. The vector $\mathbf{c}$ is called the *Fourier transform* of the vector $\mathtt{data}N$, and $\mathtt{data}N$ is called the *inverse Fourier transform* of $\mathbf{c}$. If we have $\mathtt{data}N$, we find $\mathbf{c}$ by multiplying by $\mathtt{Q}N^T$ and if we are told only $\mathbf{c}$, we can figure

out `data`N simply by multiplying **c** by `Q`N Thus knowledge of either **c** or **data**N is equivalent to knowledge of the other.

Enter the command `plot(c)` to see what the numbers in the transform vector look like. For both $N = 7$ and 256, sketch your plot of the appropriate **c** here.

27. Now we will see that the Fourier transform of our data can be useful.

    We know `data`$N = $ `Q`$N\,\mathbf{c} = c(1)$`B0`$+ c(2)$`B1`$+ c(3)$`B2`$+ \cdots + c(N)$`B`$(N-1)$ where $c(k)$ is the kth entry of **c**. Now we'll investigate how this expression 'builds up' `data`N by looking at partial sums of only the first few terms. (In what follows, be sure you are using the modified `B`k vectors, or the columns of the modified matrix `Q`N of question 20!)

    With $\mathbf{c} =$ `Q7`$^{-1}$`data7`, enter `plot([data7,c(1)*B0])` to produce a plot of `data7` and $c(1)B0$ on the same graph. Then enter `plot([data7, c(1)*B0+c(2)*B1])`, then `plot([data7, c(1)*B0+c(2)*B1+c(3)*B2])`, etc., and explain what you see. As you include more and more of the Fourier basis vectors, what happens to the graphs? Why?

28. You can do the work of the last question more easily with the following MATLAB commands:

```
ones(1,7)
c*ones(1,7)
triu(c*ones(1,7))
P=Q7*triu(c*ones(1,7))
plot([data7 P])
```

    Explain how these commands work.

29. With $\mathbf{c} = \texttt{Q256}^{-1}\texttt{data256}$, use the commands

```
P=Q256*triu(c*ones(1,256))
plot([data256 P])
```

to produce similar plots. Explain what happens as each additional column of P is plotted.

30. If you compare the plots of `data256` and $c(1)\texttt{B0}+c(2)\texttt{B1}+\cdots+c(128)\texttt{B127}$, how close are they? (The second vector here is the 128th column of P, which you can refer to as `P(:,128)` in MATLAB. The command `plot([data256,P(:,128)])` plots the two.) Does this mean that knowing just the first 128 numbers in the vector $\mathbf{c}$ is almost as good as knowing all 256 numbers in the vector `data256`? Would knowing only the first 128 coordinates of `data256` with respect to the standard basis be almost as good as knowing all 256? Explain.

31. Plot `data256` and other columns of `P`, just as you did for the 128th, and decide which is the first column of P that gives a reasonable approximation to the data. (This is entirely a subjective decision; there is no right or wrong answer.) Write out, in terms of the entries of $\mathbf{c}$ and the vectors `B`k, exactly what this column of P is. How many entries of the vector $\mathbf{c}$ went into calculating it?

32. Here are two possible applications of Fourier analysis to think about that should help you see why it is so useful.

    a) Suppose our data represents a message that must be sent down a transmission line. If we are unable to send all 256 numbers in `data256` down the line (due to the excessive time it might take to send so much information), and only send the first 128 numbers of **c**, but the receiver knows all about Fourier analysis and tries to reconstruct `data256` from this information, what features of the graph of the function have been lost to the receiver? Given that only half as many numbers were sent, did we really lose half of the message?

    Such a process is called by engineers a *low-pass filter* since only the low frequencies in the data are passed on. This idea underlies one approach to the important field of *data compression*.

    b) Now suppose our data represents measurements taken in the real world, as in the interpretations given above in item (1). Real measurements are subject to random fluctuations that are referred to as *noise*. In some settings, these tend to be small fluctuations of brief duration. In other words, the noise is what shows up in the high frequency components when we express our data in terms of the Fourier basis. Thus we can remove the noise by removing these high frequency components. This again is just a low-pass filter.

33. Now consider the three data vectors `data256`, `data256b`, and `data256c`. Sketch the plot of each here.

34. For each of `data256`, `data256b`, and `data256c`, compute the Fourier transform **c** and plot it. Sketch the three plots here.

35. How do the plots of the Fourier transforms in the last problem relate to the plots of the data vectors in the problem before that? How is the jaggedness of the data is reflected in the size of the entries of the Fourier transform?

The last few questions relate to another use of Fourier analysis, to describe different types of noise. If the vectors `data256`, `data256b`, and `data256c` are imagined to represent samples of pure noise (and not any meaningful data), the fact that their Fourier transforms look so different tells us something important about the differences between them. For instance *white noise* is defined as noise for which the Fourier transform has entries that fluctuate equally around zero. *Pink noise* is defined as noise for which the magnitude of the fluctuations of the Fourier transform entries gets smaller as the frequency increases. Would these be white or pink noise?

Final Remarks: The ideas used above are commonly used in a few variations. Sometimes sines are used either in place of, or in addition to, cosines. Sometimes complex numbers are also used (which ultimately makes things easier, though you may not believe that now). There are also versions of Fourier analysis (including the version introduced by Fourier himself) that are built not on vectors in $\mathbb{R}^N$, but on using the sine and cosine functions directly in a vector space of functions.

Computationally most of what you did here was quite fast. In fact, the hardest computation was finding $\mathbf{c}$ from `data`N, which involved either Gaussian elimination (requiring approximately $N^3/3$ steps) or matrix multiplication `Q`N^T`data`N (requiring approximately N^2 steps). If N is large, and we need to compute many Fourier transforms, this can be too much work. Fortunately, there is a very clever way to do the calculation of Fourier transforms in about $N(\log_2 N)/2$ steps. This algorithm, called the *Fast Fourier Transform*, or FFT, was invented in the 1960's and has saved a tremendous amount of time and money since then. Although we will not talk about it in this course, the FFT is the way Fourier transforms are usually computed in practice.