Overview

Suppose we have a sequence of measurements made at equally spaced times. For example

- Temperature at Fairbanks International Airport measured every hour.
- Atmospheric pressure at the surface of a microphone measured 44100 times a second.
- Population size of a herd of caribou measured monthly.

After making N such measurements, we obtain an N-dimensional vector of data called a time series. Its first component is the first measurement, and its last component is the last measurement. In this lab we will examine a basis for $\mathbb{R}^N$ that is useful for analyzing time series data.

Before starting this lab, you should follow the instructions on the web page for creating a "path" in Octave to tell it about places where to look for data files, scripts, and new functions. Then download the data file `lab2.mat` from the course web page and save it to a location in Octave's path. To load the data, use the command

```
load  lab2
```

Don't worry if you see a warning about a load path; that's normal. You can verify you loaded the data set for this lab by typing `who` at the Octave prompt. This command lists all the currently known variables. If you have loaded the file correctly, you will then have the variables

- `data16`
- `data256`
- `data_sound`

Please consult with me if you have trouble loading the data file into Matlab or Octave.

The exercises in this lab all have a lot of text in them. You'll need to read them carefully to determine what to hand in for your answers. Key words include "write", "explain", and "draw". These are all asking you to do something. Any questions asked in the exercise must be answered (look for the question marks!). There are only two Octave-generated graphs that need to be handed in; look for the words "Attach a plot" in the exercise.

Exercise 1: The variable `data16` contains 16 measurements. To visualize it, we don't think of it as a geometric vector in $\mathbb{R}^{16}$. Instead, we plot the measurements as a function of time (or of the sample index). Do this for yourself using `plot(data16)`. We can express `data16` in terms of the standard basis for $\mathbb{R}^{16}$:

$$\texttt{data16} = x_1\mathbf{e}_1 + \cdots x_n\mathbf{e}_n.$$

What is the specific value of x_5 in this case? What does x_5 tell you?

Exercise 2: Many time-series measurements can be thought of as a sum of oscillations at different frequencies. For example, a caribou population can be expected to oscillate over a period of one year due to annual season changes, but might also undergo oscillations over a fifty-year time period due to longer-term effects.

In this lab we will look at a basis for $\mathbb{R}^N$ that allows us to pick data apart into pieces oscillating at different frequencies. To begin, we need a nice way to represent the times corresponding to the measured data samples. We will scale time so that the measurements occur over the time interval $[0,1]$. If there are N samples, we will break this interval up into N subintervals. And we will assume that each sample occurs at the **middle** of its corresponding time interval.

Under these hypotheses, write down the sample times for N measurements in the following cases: $N = 1$, $N = 2$, $N = 3$, and $N = 4$. In the $N = 4$ case, also draw a schematic diagram of the interval $[0,1]$ broken into time intervals, with each of the sample times marked with an asterisk.

Exercise 3: Write down a formula for sample time t_k assuming that there are N samples, starting at $k = 1$ and ending at $k = N$.

Exercise 4: Write dow a one-line Matlab command that you could use to build the vector of sample times `t16` corresponding to `data16`. Your command should involve the vector `[0:15]'`. For yourself, `plot(t16,data16)`. What's different between this plot and the one you made in Exercise 1?

Exercise 5: For yourself, plot each of the following vectors:

- `g1 = cos( 0 * pi * t16 )`
- `g2 = cos( 1 * pi * t16 )`
- `g3 = cos( 2 * pi * t16 )`
- `g4 = cos( 3 * pi * t16 )`

In general, let gk `= cos((`$k-1$`) * pi * t16)`.

1. What frequency does gk oscillate at? Recall that frequency is the reciprocal of the period.
2. How many total peaks and troughs do you expect that gk has over the time interval $[0,1]$?
3. Why does gk look jaggier as k goes up?
4. Plot `g17` for yourself. Then **explain** what you see. A full answer does not just describe the graph, but also gives an explanation for why it is what it is. Be sure you look at the scale on the y-axis. It might be helpful to compare the graph of `g17` to some of the earlier gk's. Keep in mind that the diameter of the nucleus of a gold atom is about 10^{-14} meters.

Exercise 6: We will now show that the 16 vectors g1 through g16 form a basis for $\mathbb{R}^{16}$. The first step is to make a matrix G such that column k of G is gk. Here's a slick way to do this. Let

$$\texttt{T} = \texttt{t16} * [\texttt{0:15}] * \texttt{pi}.$$

How is column k of T related to t16? Write down a one-line Matlab command that builds the matrix G. Verify for yourself that you have the right matrix by comparing the plots of G(:,1), G(:,2), G(:,3), and G(:,4) with those of g1, g2, g3, g4.

Exercise 7: Since G is a 16×16 matrix, its columns form a basis for $\mathbb{R}^{16}$ **if** the matrix is invertible. One way to test this is do compute the reduced row echelon form of G. Do this using rref and check that you obtain the identity matrix. Then explain clearly why it would be enough simply to verify that the lower-right entry is a 1.

Exercise 8: Since the gk's form a basis for $\mathbb{R}^{16}$, every vector **b** in $\mathbb{R}^{16}$ can be written uniquely in the form

$$\mathbf{b} = x_1\ \texttt{g1} + \cdots + x_{16}\ \texttt{g16}$$

for certain scalars $x_1, \ldots, x_{16}$. To get a feeling for what a representation of this type means, plot for yourself 3*g1 + 1/2*g4 - 1/8*g10. You can do this using

```
b = 3 * G(:,1) + (1/2) * G(:,4) - (1/8) * G(:,10)
plot(t16,b)
```

What part of the resulting graph does the term 3*g0 contribute? What about 1/2*g3? What about -(1/8)*g10? You might find it easiest to answer these questions by making (for yourself) graphs that exclude one or more of the three terms.

Exercise 9: Suppose we want to find scalars x_i such that

$$\texttt{data16} = x_1\ \texttt{g0} + \cdots + x_{16}\ \texttt{g15}.$$

Write this problem down as a matrix problem involving G.

Exercise 10: To solve this problem we would normally use a method like LU factorization. But in this case, G has a very nice inverse. Compute G'*G in Matlab and then describe all of the entries. Be careful: one entry is different from all the others. This exercise shows that $\texttt{G}^{-1}$ is nearly the same as $\texttt{G}^T$.

Exercise 11: What is the value of the dot product gi · gj? Your answer should depend on i and j. Give an explicit formula. Hint: this has something to do with the previous Exercise.

Exercise 12: Find a diagonal matrix S such that F = G * S satisfies $\texttt{F}^{-1} = \texttt{F}^T$. Write down the Matlab commands you used to build S (your answer must involve the Matlab command eye). How is each column of F related to the corresponding column of G? Why are we multiplying G on the right and not on the left?

Exercise 13: The columns of F are known as the Fourier basis for $\mathbb{R}^{16}$. They are better-scaled vectors from the original basis g1 through g16 you were working with before. I'll use the notation f1 through f16 to denote this new basis.

Given a vector d in $\mathbb{R}^{16}$ we can write

$$\mathtt{d} = x_1\,\mathtt{f1} + \cdots + x_{16}\,\mathtt{f16}$$

for unique numbers x_1 through x_{16}. To compute these numbers, we would need to solve

$$\mathtt{F} * \mathtt{x} = \mathtt{d}.$$

But since $\mathtt{F}^{-1} = \mathtt{F}^T$, the solution is simply

$$\mathtt{x} = \mathtt{F}' * \mathtt{d}.$$

The vector x is called the Fourier transform of d.

Notice the very nice time savings. About how many multiplications would it take to solve `F*x = d` using LU-factorization? How many multiplications does it take to compute `F'*d`?

Exercise 14: If I give you the Fourier transform x, how can you reconstruct d using F?

Exercise 15: Let `x = F' * data16` be the Fourier transform of data16. We want to examine what happens as we build up data16 from x by adding more and more terms. That is, we want to consider the vectors

$$\begin{aligned}
\mathtt{y1} &= \mathtt{x}(1) * \mathtt{F}(:,1)\\
\mathtt{y2} &= \mathtt{x}(1) * \mathtt{F}(:,1) + \mathtt{x}(2) * \mathtt{F}(:,2)\\
\mathtt{y3} &= \mathtt{x}(1) * \mathtt{F}(:,1) + \mathtt{x}(2) * \mathtt{F}(:,2) + \mathtt{x}(3) * \mathtt{F}(:,3)
\end{aligned}$$

and so forth. What does the graph of y1 look like? How are y16 and data16 related?

Exercise 16: Create the variables y1, y3, and y5 according to the formulas above. The visualize them (for yourself) as follows:

$$\mathtt{plot}([\mathtt{data16}, \mathtt{y1}, \mathtt{y3}, \mathtt{y5}])$$

This plots the original data vector and then the three approximations obtained by adding up the first view terms. Explain what you see in your plots as you add more terms of the Fourier basis vectors to your graphs.

Exercise 17: Enter the following Matlab commands

$$\begin{aligned}
&\mathtt{z} = \mathtt{x}\\
&\mathtt{z}(9:16) = 0
\end{aligned}$$

How is z different from x? Then enter the following Matlab commands

$$\begin{aligned}
&\mathtt{w} = \mathtt{x}\\
&\mathtt{w}(1:8) = 0
\end{aligned}$$

How is w different from x? What is w + z? What is `F(w)+F(z)`?

Exercise 18: Now `plot([data16,F*z,F*w])`. What effect does leaving out early terms from the Fourier basis have? What effect does leaving out later terms have?

Exercise 19: Enter the following Matlab commands.

```
x = zeros(16,1)
x(2) = 1
x(15) = -.5
d = F * x
```

For yourself, separately plot the Fourier transform `x` and the data vector `d`. Notice that data vector is a sum of a low frequency component and a high frequency component. How can you tell this by looking at the graph of the **Fourier transform**?

Exercise 20: The data vector `data256` contains 256 measurement samples. Describe all of the Matlab command you would use to construct the 256×256 matrix `F256` corresponding to the 16×16 matrix `F` we have been working with. You should give commands to make

- `t256` corresponding to `t16`
- `G256` corresponding to `G`
- `S256` corresponding to `S`
- `F256` corresponding to `F`

Now explain two different ways you could verify that the columns of `F256` are a basis for $\mathbb{R}^{256}$.

Exercise 21: The measurement samples in `data256` have been contaminated by noise. We can use the Fourier transform to implement a **low-pass** filter, which allows low frequency components to pass through but removes high frequency components. This will eliminate the high frequency noise in the signal. Write down Matlab commands that use `F256` and that remove much of the noise from the data. Attach a plot of the original data and your version with the noise removed.

Exercise 22: The vector `data_sound` contains 8820 samples corresponding to 1/5 of a second of me trying to sing a note. We would like to analyze it using the Fourier basis, but the methods we have been using are not efficient enough. For example, the matrix `F8820` would needlessly take up over half a gigabyte of memory. Computing

```
F8820' * data_sound
```

would take roughly $8820^2 \approx 10^8$ multiplications. This is far better than the $8820^3/3 \approx 10^{11}$ multiplications required by LU factorization, but is still a bit large.

I need to confess at this stage I've been telling you a white lie. The transform we are using here is called the **discrete cosine transform**. The Fourier transform is a similar, but more

complex object. In practice, discrete cosine transforms are computed using a fantastic algorithm called the Fast Fourier Transform, which requires only $O(N\log(N))$ multiplications. Note that $8820\log(8820) \approx 3.5 \cdot 10^4$, which is dirt cheap!

Those of you using Matlab (not Octave) can compute discrete cosine transforms, based on the Fast Fourier Transform, using the command `dct`. You can convert transform data back into time-series data using the command `idct` (inverse discrete cosine transform). There are also official versions of `dct` and `idct` for Octave, but they are a hassle to install. I have posted on my web page cheesy versions of these functions. They work, and they don't use up excessive memory, but they are $O(N^2)$ algorithms, not $O(N\log(N))$ algorithms. Please see the instructions on the course web page on how to install these files if you are using Octave.

Do the following (assuming you still have the matrix `F` you computed earlier still in memory):

```
x16_dct = dct(data16)
x16 = F' * data16
plot([x16, x16_dct])
```

and verify that the vectors `x16_dct` and `x16` are the same. The algorithm in `dct` is exactly what we have been doing (inefficiently). Then compute the discrete cosine transform `x_sound` of `data_sound`. Plot `x_sound` for yourself, and then describe the features of the plot.

Exercise 23: What is the sample rate of the audio data in data_sound? Your answer should be in Hz.

Exercise 24: What is the period and the frequency of the function `f4` in this case? It might be helpful to go look at how you found your answer to Exercise 5. But keep in mind that for that problem we assumed time was scaled so that all the data was sampled between $t = 0$ and $t = 1$. You'll need to adjust this scaling to get a meaningful answer.

Exercise 25: What are all of the dominant frequencies in the audio sample? That is, what are the frequencies associated with the spikes? What note was I trying to sing? Provide justification for your answers using the tools developed in this lab. Keep in mind your answer to Exercise 24.

Exercise 26: Thinking of the audio sample, notice that the discrete cosine transform of the full signal is nearly zero for most of the frequencies. If we needed to transmit (or store) this signal, we don't really need all 8820 components of `data_sound`. We could transmit (or store) just 1000 numbers instead. What 1000 numbers should we store? How would we reconstruct the signal based on those 1000 numbers? Attach a plot of the **first 4 milliseconds** of the reconstructed and original signal.

This principle underlies many techniques of "lossy compression". High frequencies are omitted with the expectation that they carry little data. The JPEG image standard uses a two dimensional discrete cosine transform applied to little 8x8 tiles in the picture. The MP3 audio format uses a variation of the discrete cosine transform as well.