# EECS 16A Designing Information Devices and Systems I
# Summer 2020 Final Practice (Optional)

***Learning Goals:*** *The first problem of this practice set guides you through the development of another greedy algorithm. The second problem is a mechanical OMP problem meant to distill the algorithm to strictly what is necessary, removing the consideration of time delayed signals in GPS applications. The last two problems explore the effect of design decisions in applying OMP and least squares.*

**1. Greedy Algorithm for Calculating Matrix Eigenvalues**

In the real world, it is not computationally practical to directly solve for the eigenvectors for large matrices as you might do for small matrices on paper. You would like to build an algorithm that sequentially computes the eigenvectors for a square symmetric matrix $\mathbf{Q} = \mathbf{A}^T\mathbf{A}$ (Note: A symmetric matrix $\mathbf{P}$ is a matrix such that its transpose is equal to itself, i.e. $\mathbf{P}^T = \mathbf{P}$. Component-wise this means $p_{ij} = p_{ji}$.)

To accomplish this we are given access to a function,

$$(\vec{v}_1, \lambda_1) = f(\mathbf{Q}), \tag{1}$$

that returns the **largest** eigenvalue of the matrix $\mathbf{Q}$ and the corresponding eigenvector. You do not need to know about the origins of the function, but if you are curious to learn more you can look up "Power iteration". We will use this function to build a greedy algorithm similar in the spirit of orthogonal matching pursuit that computes the eigenvectors and eigenvalues in descending order. By greedy algorithm we mean that we will choose eigenvectors one by one, at any time picking the eigenvector corresponding to the largest eigenvalue.

**IMPORTANT: Throughout the problem, assume that the magnitude of each eigenvector is 1 (i.e. $\|\vec{v}_i\| = 1$), that the eigenvalues are real, unique, and distinct, and that the eigenvalues are indexed in a descending order $\lambda_1 > \lambda_2 > \cdots > \lambda_N > 0$.**

(a) Let

$$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}. \tag{2}$$

Calculate $\mathbf{Q}$ and $\mathbf{Q}^\mathbf{T}$, where $\mathbf{Q} = \mathbf{A}^T\mathbf{A}$. Show that $\mathbf{Q} = \mathbf{Q}^\mathbf{T}$ (i.e. $\mathbf{Q}$ is symmetric).

Now, consider a general matrix $\mathbf{A} \in \mathbb{R}^{2x2}$. Let $\mathbf{Q} = \mathbf{A}^T\mathbf{A}$. Show that $\mathbf{Q} \in \mathbb{R}^{2x2}$ is symmetric.

(b) Let us consider the matrix $\mathbf{V} \in \mathbb{R}^{N\times N}$:

$$\mathbf{V} = \begin{bmatrix} | & | & & | \\ \vec{v}_1 & \vec{v}_2 & \dots & \vec{v}_N \\ | & | & & | \end{bmatrix}. \tag{3}$$

Show that if $\langle \vec{v}_i, \vec{v}_j \rangle = 0$ for $i, j = \{1, \dots, N\}$ when $i \neq j$ (all the columns of $\mathbf{V}$ are orthogonal to each other), then $\mathbf{V}^T\mathbf{V} = \mathbf{I}$.

(c) Show that if the columns of matrix $\mathbf{V}$ are orthogonal to each other, then the columns form a basis for $\mathbb{R}^N$.

(d) Assume that the columns of matrix $\mathbf{V}$ are orthogonal to each other for the remainder of the problem. Since the columns of $\mathbf{V}$ form a basis, let $\vec{b} = \alpha_1 \vec{v}_1 + \cdots + \alpha_N \vec{v}_N$ for some scalars $\alpha_i$. Find $\langle \vec{v}_i, \vec{b} \rangle$.

(e) Find the $\hat{\vec{x}}$ that minimizes the following least squares problem:

$$\min_{\vec{x}} ||\mathbf{V}\vec{x} - \vec{b}||$$

Let $\vec{e} = \mathbf{V}\vec{x} - \vec{b}$. Also find $||\vec{e}||$ for the optimal $\hat{\vec{x}}$.
*Hint:* Use what you proved in the earlier parts of this problem.

(f) Let us define

$$\mathbf{V}_2 = \begin{bmatrix} | & & | \\ \vec{v}_2 & \dots & \vec{v}_N \\ | & & | \end{bmatrix} \tag{4}$$

where we have removed the first column of $\mathbf{V}$. Note that $\mathbf{V}_2 \in \mathbb{R}^{N \times (N-1)}$, and $\mathbf{V}_2$ is not invertible. Assume $\vec{\alpha} = [\alpha_1 \dots \alpha_N]^T$ are the same weights as in part (d). Find the $\hat{\vec{x}}$ that minimizes the following least squares problem:

$$\min_{\vec{x}} ||\mathbf{V}_2\vec{x} - \vec{b}||.$$

Let $\vec{e} = \mathbf{V}_2\vec{x} - \vec{b}$. Also find $||\vec{e}||$.

(g) Consider a matrix $\mathbf{Q}$ given as:

$$\mathbf{Q} = \mathbf{V}\Lambda\mathbf{V}^T = \sum_{i=1}^{N} \lambda_i \vec{v}_i \vec{v}_i^T \tag{5}$$

$$\text{where } \Lambda = \begin{bmatrix} \lambda_1 & 0 & 0 & \dots & 0 \\ 0 & \lambda_2 & 0 & \dots & 0 \\ 0 & 0 & \lambda_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \lambda_N \end{bmatrix} \tag{6}$$

It turns out that the eigenvectors of $\mathbf{Q}$ are given by the columns of $\mathbf{V}$. Try to prove this yourself. Consider:

$$\begin{aligned} \mathbf{Q}\vec{v}_1 &= \mathbf{V}\Lambda\mathbf{V}^T\vec{v}_1 \\ &= \mathbf{V}\Lambda \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \\ &= \mathbf{V} \begin{bmatrix} \lambda_1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \\ &= \lambda_1 \vec{v}_1 \end{aligned}$$

Can you justify why each of the steps in the proof above is correct?

(h) Now consider $\mathbf{Q}_2 = \mathbf{Q} - \lambda_1 \vec{v}_1 \vec{v}_1^T$. Thus, $\mathbf{Q}_2$ represents $\mathbf{Q}$ after the component associated with direction $\vec{v}_1$ is removed. Show that $\vec{v}_1$ is in the null space of $\mathbf{Q}_2$, and $\vec{v}_2$ to $\vec{v}_N$ are eigenvectors of $\mathbf{Q}_2$.
Hint: *Can you write* $\mathbf{Q}_2$ *using Eq. (5)?*

(i) Recall the function that returns the **largest** eigenvalue and corresponding eigenvector of a matrix,

$$(\vec{v}_1, \lambda_1) = f(\mathbf{Q}). \tag{7}$$

Design an algorithm that returns a list of eigenvalues of matrix $\mathbf{Q}$ in descending order of values. You may assume that all the eigenvalues of $\mathbf{Q}$ are positive ($> 0$). You are allowed to use the function defined in Eq. (7) that returns the largest eigenvalue and corresponding eigenvector and what you know from part (g).

**2. OMP Exercise**

Suppose we have a vector $\vec{x} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \end{bmatrix}^T \in \mathbb{R}^4$. The vector $\vec{x}$ is related to the vector $\vec{b}$ in the following way:

$$\mathbf{M}\vec{x} \approx \vec{b}$$

$$\begin{bmatrix} 1 & 1 & 0 & -1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \approx \begin{bmatrix} 4 \\ 6 \\ 3 \end{bmatrix}$$

For this undetermined and possibly noisy problem of finding $\vec{x}$, assume that $\vec{x}$ is sparse: it has 2 non-zero entries and 2 zero entries. Use Orthogonal Matching Pursuit to estimate $x_1$ to $x_4$.

(Note: Unlike previous examples you may have seen, we will not cross correlate $\vec{b}$ with the columns of $\mathbf{M}$. Instead, we will just compute the inner product of $\vec{b}$ with every column of $\mathbf{M}$.)

**3. Sparse Imaging**

Recall the imaging lab where we projected masks onto an object to scan it into our computer using a single pixel measurement device, that is, a photoresistor. In that lab, we were scanning a $30 \times 40$ image having 1200 pixels. In order to recover the image, we took exactly 1200 measurements because we wanted our 'mask matrix' $\mathbf{A}$ to be invertible. In this problem, we have a 2D image $\mathbf{I}$ of size $91 \times 120$ pixels for a total of 10920 pixels. We will explore how to reconstruct the image using only 6500 measurements.

The image is made up of mostly black pixels (represented by a zero) except for 476 white ones (represented by a one). In cases where there are a small number of non-black pixels, we can reduce the overall number of samples necessary using the orthogonal matching pursuit algorithm. This reduces the time required for scanning an image, a real-world concern for lengthy processes like MRI where people have to stay still while being imaged.

Although the imaging illumination masks we used in the lab consisted of zeros and ones, in this question, we are going to have masks with real values. Say that we have an imaging mask $\mathbf{M_0}$ of size $91 \times 120$. The measurements using this imaging mask can be represented as follows:

First, let us put every element in the matrix $\mathbf{I}$ into a column vector $\vec{i}$ of length 10920. This operation is referred to as vectorization. Likewise, let us vectorize the mask $\mathbf{M_0}$ to $\vec{m}_0$ which is a column vector of length 10920. Then the measurement using the mask $\mathbf{M_0}$ can be represented as

$$b_0 = \vec{m}_0^T \vec{i}.$$

Say we have a total of $K$ measurements, each taken with a different illumination mask. Then, these measurements can collectively be represented as

$$\vec{b} = \mathbf{A}\vec{i},$$

where $\mathbf{A}$ is an $K \times 10920$ size matrix whose rows contain the vectorized forms of the illumination masks, that is

$$\mathbf{A} = \begin{bmatrix} — & \vec{m}_1^T & — \\ — & \vec{m}_2^T & — \\ & \vdots & \\ — & \vec{m}_K^T & — \end{bmatrix}.$$

To show that we can reduce the number of samples necessary to recover the sparse image $\mathbf{I}$, we are going to only generate 6500 masks. We will generate $\mathbf{A}$ so that the columns of $\mathbf{A}$ are approximately orthogonal with each other. The following question refers to the part of IPython notebook file accompanying this homework related to this question.

(a) In the IPython notebook, we call a function `simulate` that generates masks and the measurements. You can see the masks and the measurements in the IPython notebook file. Complete the function `OMP` that does the OMP algorithm described in lecture.

**Remark:** When you look at the vector `measurements` you will see that it has zero average value. Likewise, the columns of the matrix containing the masks $\mathbf{A}$ also have zero average value. To satisfy these conditions, they need to have negative values. However, in an imaging system, we cannot project negative light. One way to get around this problem is to find the smallest value of the matrix $\mathbf{A}$ and subtract it from all entries of $\mathbf{A}$ to get the actual illumination masks. This will yield masks with positive values, hence we can project them using our real-world projector. After obtaining the readings using these masks, we can remove their average value from the readings to get measurements as if we had multiplied the image using the matrix $\mathbf{A}$.

(b) Run the code `rec = OMP((height, width), sparsity, measurements, A)` and see the image being correctly reconstructed from a number of samples smaller than the number of pixels of your figure. What is the image?

(c) **PRACTICE:** We have supplied code that reads a PNG file containing a sparse image, takes measurements, and performs OMP to reconstruct the original image. An example input image file is also supplied together with the code. Recover `smiley.png` using OMP with 6500 measurements.

You can answer the following parts of this question in very general terms. Try reducing the number of measurements. Does the algorithm start to fail in recovering your sparse image? Why do you think it fails?

## 4. How Much Is Too Much?

When discussing circuits in this course, we only talked about resistor $I$-$V$ curves. There are many other devices that can be found in nature that do not have linear $I$-$V$ relations. Instead, $I$ is some general function of $V$, that is $I = f(V)$. Often times, the function describing the $I$-$V$ relationship is not known beforehand. The function $f$ is assumed to be a polynomial, and the parameters of $f$ (the coefficients for every power of $V$) are computed using least squares.

Throughout this problem, we are provided with $\vec{x}$, a set of voltage measurements, and $\vec{y}$, a set of current measurements.

(a) Let's first try to model a resistor *I*-*V* curve. Run the code in the attached IPython notebook. What is the degree of the polynomial that fits an ideal resistor *I*-*V* curve? Play around the with degree in the IPython notebook and observe the best fit polynomial's shape. Is the noise influencing the higher degree polynomials?

(b) The attached IPython notebook provides functions `data_matrix, leastSquares` that allow you to fit polynomials of different degrees to the data provided. We also provide a function `cost` that computes the squared error of the fit. In the attached IPython notebook, plot the cost of various degree polynomials fitting to the measured *I*-*V* data points for a resistor using the given `cost` function. The `cost` function returns $||\vec{y} - \mathbf{A}\vec{f}||^2$, i.e. the squared magnitude of the error vector.

$$\mathbf{A} = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots \\ 1 & x_2 & x_2^2 & \dots \\ 1 & x_3 & x_3^2 & \dots \\ & \vdots & & \ddots \end{bmatrix}$$

As seen above $\mathbf{A}$ is the appropriately sized matrix containing powers of the elements of $\vec{x}$ and the vector $\vec{f}$ contains entries $f_n$ that are the coefficients for the nth power of the elements of $\vec{x}$. Comment on the shape of the "Cost vs. Degree" graph. Do we want to choose a best fit polynomial of degree greater than one if the cost is lower than the polynomial of degree one? Should we choose the degree of the polynomial based on this graph? This question is meant to make you think, do not worry too much about getting a precise right answer here.