University of Toronto
Department of Electrical and Computer Engineering

ECE367 MATRIX ALGEBRA AND OPTIMIZATION

Problem Set #2

Fall 2023

Prof. Wei Yu **Due:** 11:59pm (Toronto time) October 11, 2023

Homework policy: Problem sets must be turned by the due date and time. Late problem sets will not be accepted. See the information sheet for further details. The course text "Optimization Models" is abbreviated as "OptM" and "Introduction to Applied Linear Algebra" as "IALA".

Problems are categorized as:

- **"Theory" problems:** These are mostly mathematical questions designed to give you deeper insight into the fundamentals of the ideas introduced in this class.
- **"Application" problems:** These questions are designed to expose you to the breadth of application of the ideas developed in class and to introduce you to useful numerical toolboxes. Problems of this sort often ask you to produce plots and discuss your results; said plots and discussions should be included in and form part of your submission – think of your submitted solution like a lab book. Your attached code simply provides back-up evidence.
- **"Optional" problems:** Optional problems provide extra practice or introduce interesting connections or extensions. They need not be turned in. I will assume you have reviewed and understood the solutions to the optional problems when designing the exams.

Hand-in procedure:

- **Initial submission:** Your initial submission of the "Theory" and "Application" questions must be submitted via Quercus upload by the due date.
- **Self-assessment:** After the problem set is due we will post solutions. You will have one week from the initial due date to submit a commented version of your assignment in which, using as a reference the posted solutions, you highlight your errors or omissions in red. Please annotate the PDF you initially submitted. If you have not submitted a solution you cannot submit the self-assessment.
- **Late problem sets are not accepted**
- **Grading:** Per the course handout problem sets are graded for completion only. Points are assigned to (i) Initial submission of theory part, (ii) Initial submission of application part, (iii) Self-assessment.

These problem sets were initially developed by Prof. Stark Draper.

Points allocation

- Theory parts (initial submission): 1 pt
- Application parts (initial submission): 1 pt
- Self-assessment: 1 pt

Problem categorization and main concepts covered

Theory

- Projection: Problems 2.1, 2.2
- Hyperplanes: Problem 2.3
- Function Approximation: Problem 2.4
- Matrices: Problem 2.5

Application

- Function Approximations: Problems 2.6, 2.7
- Eigenvalue and eigenvector: Problem 2.8

Optional

- Numerically computing rank, range, nullspace: Problems 2.9, 2.10
- Linear map as matrix: Problem 2.11

THEORY PROBLEMS

Problem 2.1 (Gram-Schmidt algorithm)
IALA Prob. 5.6.

Problem 2.2 (Computing projections in Euclidean space)
In this problem we use the notation $\text{Proj}_{\mathcal{S}}(x)$ to denote the projection of a vector x onto some set $\mathcal{S}$, which consists of vectors that are of same dimension as x. Consider the following vectors and subspaces.

$$x_1 = \begin{bmatrix} 3 \\ -1 \end{bmatrix}, \quad b_1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \quad \mathcal{V}_1 = \text{span}\left\{ \begin{bmatrix} 1 \\ 2 \end{bmatrix} \right\}$$

$$x_2 = \begin{bmatrix} 2 \\ 1 \\ -5 \end{bmatrix}, \quad b_2 = \begin{bmatrix} 0 \\ -3 \\ -1 \end{bmatrix}, \quad \mathcal{V}_2 = \text{span}\left\{ \begin{bmatrix} 0 \\ 0 \\ -2 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \right\}$$

$$x_3 = \begin{bmatrix} 3 \\ 0 \\ -1 \\ 2 \\ 2 \end{bmatrix}, \quad b_3 = \begin{bmatrix} -1 \\ 0 \\ 1 \\ -2 \\ 1 \end{bmatrix}, \quad \mathcal{V}_3 = \text{span}\left\{ \begin{bmatrix} 0 \\ 1 \\ 2 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 3 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 5 \\ 0 \\ 1 \end{bmatrix} \right\}$$

(a) Compute $\text{Proj}_{\mathcal{V}_i}(x_i)$ for $i = 1, 2, 3$.

(b) Consider the affine set $\mathcal{A}_i = \{v + b_i \mid v \in \mathcal{V}_i\}$. Compute $\text{Proj}_{\mathcal{A}_i}(x_i)$ for $i = 1, 2, 3$.

(c) On a 2-d map, sketch the subspace $\mathcal{V}_1$ (a line through the origin) and clearly indicate x_1 and $\text{Proj}_{\mathcal{V}_1}(x_1)$. What is the point on $\mathcal{V}_1$ that is the closest to x_1 in Euclidean sense? On the same axes, sketch $\mathcal{A}_1$ (a line shifted from the origin) and indicate $\text{Proj}_{\mathcal{A}_1}(x_1)$.

(d) Compute an orthonormal basis $\mathcal{B}_3$ for the subspace $\mathcal{V}_3$ via Graham-Schmidt. Recompute $\text{Proj}_{\mathcal{V}_3}(x_3)$ and $\text{Proj}_{\mathcal{A}_3}(x_3)$ using $\mathcal{B}_3$, and compare with your previous results.

Problem 2.3 (Distance between a pair of parallel hyperplanes)
Find the distance between the two parallel hyperplanes $\mathcal{H}_i$, $i \in [2]$ where $\mathcal{H}_i = \{x \in \mathbb{R}^n | a^T x = b_i\}$. Your solution should be expressed in terms of the problem parameters, i.e., the vector $a \in \mathbb{R}^n$ and the scalars $b_i \in \mathbb{R}$. (Note: this problem is from "Convex Optimization" by Boyd and Vandenberghe.)

Problem 2.4 (Taylor series expansion)
Consider the function $f(x) = -\sum_{l=1}^{m} \log(b_l - a_l^T x)$, where $x \in \mathbb{R}^n$, $b_l \in \mathbb{R}$ and $a_l \in \mathbb{R}^n$. Compute $\nabla f(x)$ and $\nabla^2 f(x)$. Write down the first three terms of the Taylor series expansion of $f(x)$ around some x_0.

Problem 2.5 (Linear dynamical systems)
OptM Problem 3.4.

APPLICATION PROBLEMS

Problem 2.6 (First-order approximation of functions)
In this exercise you will write MATLAB (or Python, Julia, etc) code to plot linear approximations each of three functions $f_i : \mathbb{R}^2 \to \mathbb{R}$, $i \in [3]$. The three functions are defined pointwise as

$$\begin{aligned} f_1(x, y) &= 2x + 3y + 1, \\ f_2(x, y) &= x^2 + y^2 - xy - 5, \\ f_3(x, y) &= (x - 5)\cos(y - 5) - (y - 5)\sin(x - 5). \end{aligned}$$

For each of the above function, do the following.

(a) Write down the gradient with respect to x and y in closed form. The gradient can be compactly written in the form $\nabla f_i = \left[\frac{\partial f_i}{\partial x} \frac{\partial f_i}{\partial y}\right]^T$ for $i = 1, 2, 3$.

(b) For each function produce a 2-D contour plot indicating the level sets of each function in the range $-2 \leq x, y \leq 3.5$ (i.e., make three plots). An example of a contour plot is illustrated in Fig 2.28 of OptM (the second sub-figure). You may find `meshgrid` and `contour` commands in MATLAB useful. Please refer the MATLAB documentation for further details. In addition, compute the the gradient at the point $(x, y) = (1, 0)$ for each function. On your contour plots also plot the direction of the gradient and the tangent line to the level sets. Your resulting plot should be analogous to Fig 2.29 of OptM.

(c) For the same point $(x, y) = (1, 0)$ where, plot the 3-D linear approximation of the function. Since we are considering only the first derivative, the approximation should be the tangent plane at the specified point. Your plot for $f_2(x, y)$ should look something like Fig. 1. The function approximation is plotted as a tangent plane to the surface plot of $f_2(x, y)$. You may find `meshgrid` and `meshc` (or `mesh`) commands in MATLAB useful. Include your code for all sections.

Note: We recommend you design these plotting scripts as functions (in MATLAB, Python, Julia) so that you can reuse them for to plot approximations for different non-linear functions (or for theses functions at different points). In either case make sure to attach your code.

Problem 2.7 (Second-order approximation of functions)
In this exercise you extend your code from the problem "First-order approximation of functions" to second order. (As before you are welcome to use Matlab or Python or the software of your choice.) The objective is, as before, to write code to plot approximations each of (the same) three functions

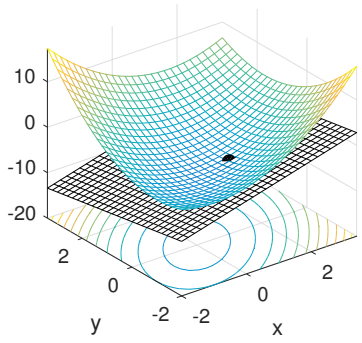

Figure 1: Example plot and approximating tangent plane.

$f_i : \mathbb{R}^2 \to \mathbb{R}$, $i \in [3]$, but now the approximation is quadratic rather than linear. To recall, the three functions are defined pointwise as

$$f_1(x, y) = 2x + 3y + 1,$$
$$f_2(x, y) = x^2 + y^2 - xy - 5,$$
$$f_3(x, y) = (x - 5)\cos(y - 5) - (y - 5)\sin(x - 5).$$

For each of the above function, do the following.

(a) Write down the gradient and Hessian with respect to x and y in closed form. To recall the gradient and Hessian are compactly denoted as ∇f_i and $\nabla^2 f_i$ for $i \in [3]$ where

$$\nabla f_i = \begin{bmatrix} \frac{\partial f_i}{\partial x} \\ \frac{\partial f_i}{\partial y} \end{bmatrix} \quad \nabla^2 f_i = \begin{bmatrix} \frac{\partial^2 f_i}{\partial^2 x} & \frac{\partial^2 f_i}{\partial x \partial y} \\ \frac{\partial^2 f_i}{\partial y \partial x} & \frac{\partial^2 f_i}{\partial^2 y} \end{bmatrix}.$$

(b) For each function produce do the following two things. First, produce a 2-D contour plot indicating the level sets of each function in the range $-2 \leq x, y \leq 3.5$ and plot the direction of the gradient and tanget to the level set at the point (x,y) = 1,0). (Note, you have already produced these plots in the previous problem, we are just reproducing them here to help with visualization). Second, For the same point $(x, y) = (1, 0)$ plot the 3-D quadratic approximation of the function.

(c) Now, repeat the previous plot for point $(x, y) = (-0.7, 2)$ and $(x, y) = (2.5, -1)$.

(d) Comment on where your approximations are accurate and where they are not (if anywhere) for the three functions. Discuss what the reason is behind your observations.

Note: We recommend you design these plotting scripts as Matlab functions so that you can reuse them for to plot approximations for different non-linear functions (or for theses functions at different points). In either case make sure to attach your code.

Problem 2.8 (Google's PageRank algorithm)
In this problem you will implement and analyse approaches to the eigenvector computation that is at the heart of Google's PageRank algorithm (cf. discussion in Example 7.1 of OptM). Download `pagerank_urls.txt` and `pagerank_adj.mat` files from the course website. The first is a plain text file in which each line consists of a URL of a web page. We refer to the web pages by their respective URL-line numbers starting from 1. For example, web page 9 is the page that the URL in line 9 points to. The URLs are of the internal web pages of Hollins University in Roanoke, Virginia. The provided data files consist of a modified version of web crawling data downloaded from `https://www.limfinity.com/ir`. The original dataset has been created in January, 2004 therefore you might notice that some of the links are inactive.

Let N be the number of URLs (therefore the number of web pages) in the first file and $i, j \in [N]$. Execute the command `load 'pagerank_adj.mat'` in MATLAB to load the content of the second file into your MATLAB workspace. You will see that a new $N \times N$ variable `J` has been imported. This variable represents an *adjacency matrix* $J \in \{0,1\}^{N\times N}$ which describes the relationships between the web pages. Specifically, the element in ith row and jth column $J_{i,j} = 1$ if there exists a link from web page j to web page i, and $J_{i,j} = 0$ otherwise. Data have been carefully filtered so that $J_{j,j} = 0$ and $\sum_{i=1}^{N} J_{i,j} > 0$ for all $j \in [N]$. In other words, we do not allow links from a web page to itself, and we do not allow for dangling pages, that is pages with no outgoing links.

Use J to obtain the *link matrix* A where

$$A_{i,j} = \frac{J_{i,j}}{\sum_{k=1}^{N} J_{k,j}}.$$

Also, let $x \in \mathbb{R}^N$ be a vector with all entries equal to 1. Use the matrix A and the vector x the same way as described in Example 7.1 to solve the following.

(a) Verify that each column in the provided matrix A sum to 1. What is the importance of this property for the Google PageRank algorithm?

(b) In the following we are consistent with the notation used in OptM. Let $x(k+1)$ be the approximation of the eigenvector in the $k+1$th iteration. We define the approximation error in the $k+1$th iteration as $e(k+1) = \|Ax(k+1) - x(k+1)\|_2$. Using MATLAB, implement the power iteration algorithm described in Section 7.1.1 in OptM. Run the algorithm for 10 iterations and plot $\log(e(k+1))$ versus k.

(c) Implement the shift-invert power iteration and Rayleigh quotient iteration algorithms presented in Sections 7.1.2 and 7.1.3 of OptM. For the shift-invert power method use $\sigma = 0.99$. In the Rayleigh quotient iteration method, use $\sigma_1 = \sigma_2 = 0.99$ for the first two iterations and $\sigma_k = \frac{x^*(k)Ax(k)}{x^*(k)x(k)}$ for $k > 2$, in a similar manner as the discussion in Example 7.1. Repeat your experiment of part (b) for these two algorithms. Plot $\log(e(k+1))$ for each of your three algorithms on a single plot. Check whether your results are consistent with Example 7.1. Include your MATLAB code with the answers.

(d) List the (page index, PageRank score) pairs of the top 5 and bottom 5 pages according to your PageRank scores. Compare them with the provided web page links and briefly explain whether the ranking seem intuitively correct.

(Note: While we write "MATLAB" in the above you are, as usual, welcome to use any software you would like to, just not to use built-in functions that accomplish the objective.)

OPTIONAL PROBLEMS

Problem 2.9 (Practice computing rank, range, nullspaces)

(a) Find rank(A) and the dimension of, and a basis for, each of the four subspaces $\mathcal{R}(A)$, $\mathcal{R}(A^T)$, $\mathcal{N}(A)$, $\mathcal{N}(A^T)$ when

$$A = \begin{bmatrix} 1 & 2 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

(b) Find rank(B) and the dimension of, and a basis for, each of the four subspaces $\mathcal{R}(B)$, $\mathcal{R}(B^T)$, $\mathcal{N}(B)$, $\mathcal{N}(B^T)$ when

$$B = \begin{bmatrix} 1 & 2 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 2 & 0 & 1 \end{bmatrix}.$$

(b) Find rank(C) and the dimension of, and a basis for, each of the four subspaces $\mathcal{R}(C)$, $\mathcal{R}(C^T)$, $\mathcal{N}(C)$, $\mathcal{N}(C^T)$ when

$$C = \begin{bmatrix} 1 & 2 & 1 & 3 \\ 3 & 4 & 6 & 9 \\ 1 & 4 & 4 & 4 \\ 1 & 0 & 10 & 4 \end{bmatrix}.$$

Problem 2.10 (Rank and nullspace)
OptM Problem 3.6.

Problem 2.11 (The matrix of a linear map)
In the setup of this problem we prove that *any* linear map between vector spaces can be represented by a matrix. You will then extend that reason to show that any linear function on a vector space can be represented by a vector and you will show that the way matrix-matrix multiplication is defined follows from the concatenation of any pair of compatible linear maps.

In this problem we work with a triplet of vector spaces $\mathcal{V}$, $\mathcal{W}$ and $\mathcal{U}$ where $\dim(\mathcal{V}) = n$, $\dim(\mathcal{W}) = m$, and $\dim(\mathcal{U}) = p$. Let $\{v^{(i)}\}_{i\in[n]} = \text{basis}(\mathcal{V})$, $\{w^{(i)}\}_{i\in[m]} = \text{basis}(\mathcal{W})$, and $\{u^{(i)}\}_{i\in[p]} = \text{basis}(\mathcal{U})$. We first recall the definition of a linear map, e.g,. from $\mathcal{V}$ to $\mathcal{W}$.

Definition 1. *A map $f : \mathcal{V} \to \mathcal{W}$ is "linear" if for any two points $x^{(i)} \in \mathcal{V}$, $i \in [2]$, and scalings α_i, $i \in [2]$*

$$f(\alpha_1 x^{(1)} + \alpha_2 x^{(2)}) = \alpha_1 f(x^{(1)}) + \alpha_2 f(x^{(2)}).$$

To introduce you to the perspectives we need in this problem we now show that *any* linear map f can be fully specified by an $m \times n$ matrix of coefficients. First note that any $x \in \mathcal{V}$ can be expressed in terms the the basis elements of $\mathcal{V}$ as $x = \sum_{i\in[n]} \alpha_i v^{(i)}$ for some choice of the α_i. Next, we have

$$f(x) = f\left(\sum_{i=1}^{n} \alpha_i v^{(i)}\right) \overset{(i)}{=} \sum_{i=1}^{n} \alpha_i f(v^{(i)}) \tag{1}$$

$$\overset{(ii)}{=} \sum_{i=1}^{n} \alpha_i \left[\sum_{k=1}^{m} \beta_k^{(i)} w^{(k)}\right], \tag{2}$$

where (i) follows by the linearity of f and the $\beta_k^{(i)}$ in (ii) are the coefficients associated with the basis elements of $\mathcal{W}$ that represent $f(v^{(i)}) \in \mathcal{W}$, the point in $\mathcal{W}$ to which the ith basis element of $\mathcal{V}$ maps. (Note that this point need not itself be a basis element of $\mathcal{W}$.) Expanding this out in matrix form we find that

$$f(x) = \left[w^{(1)} w^{(2)} \dots w^{(m)}\right] \begin{bmatrix} \beta_1^{(1)} & \beta_1^{(2)} & \dots & \beta_1^{(n)} \\ \beta_2^{(1)} & \beta_2^{(2)} & \dots & \beta_2^{(n)} \\ \vdots & & \ddots & \vdots \\ \beta_m^{(1)} & \beta_m^{(2)} & \dots & \beta_m^{(n)} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{bmatrix}. \tag{3}$$

Hence, any linear operator f can be represented by a matrix that relates the basis of the input space to the basis of the output space. If one changes the linear operator then, naturally, the matrix changes. However, one should also note that if one changes either the basis that represents the input space or the basis that represents the output space the matrix will also change. Thus the representation of a linear map by a matrix is not dependent solely on f but *also* on the bases used to describe points in the input and output spaces. To make this dependence explicit, one therefore denotes the matrix of the linear map f as $\mathcal{M}(f, \{v^{(1)}, \dots v^{(n)}\}, \{w^{(1)}, \dots, w^{(m)}\})$. Most of the time one writes down a matrix the bases are not specified which (typically) means that the standard basis is being assumed. Indeed, often when introducing matrices in linear algebra it is not even made explicit that one is using the standard basis to parameterize the input and output spaces of the mapping that the matrix describes. The problem framing above makes that choice explicit and, of course, and one need not have made that (perhaps unknowing) assumption.

The above was a bit hard to work into a problem for you to solve, so we simply provided the framework, derivation, and discussion. Now for the work. In the next part you consider linear

functions (rather than maps) and then extend the above logic and see what happens when you concatenate a pair of linear maps.

(a) Based on the logic above argue why any linear function can be represented by a vector. (Hint: Yes, this part is as easy as it seems.)

(b) Now, let $f : \mathcal{V} \to \mathcal{W}$ be as in the problem introduction. In addition let $g : \mathcal{W} \to \mathcal{U}$. Following the same logic as in the problem introduction, find an expression for $g(f(x))$ of the form $g(f(x)) = \sum_{l=1}^{p} \zeta_l u^{(l)}$. (Note that $g(f(x)) \in \mathcal{U}$ and $\{u^{(1)}, \dots u^{(p)}\}$ is a basis for $\mathcal{U}$.) (Hint: Your expression should contain three sums, one of n terms, one of m terms, and one of p terms.)

(c) Look at your expressions for the ζ_l from part (b) and rewrite in a form that involves two vectors and two matrices. You should now see why matrix-matrix multiplication is defined in the way that it is.