University of Toronto
Department of Electrical and Computer Engineering

ECE367 MATRIX ALGEBRA AND OPTIMIZATION

Problem Set #2

Autumn 2021

Prof. S. C. Draper **Due:** 8pm (Toronto time) Saturday, 02 October 2021

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 futher 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.
- **"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 procudure:

- **Initial submission:** Your initial submission of the "Theory" and "Application" questions must be submitted via Quercus upload by the due date. Click on the `Assignments` tab, then look for the `Initial submission` tab and upload under the correct problem set number.
- **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 omissings in red. Annotate the PDF you initially submitted. If you have not submitted a solution you cannot submit the self-assessment. To submit the self-assessment on Quercs, click on the `Assignments` tab, then look for the `Self-assessment` tab and upload under the correct problem set number.
- **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) Submission of application part, (iii) Self-assessment of theory part. The relative points breakdown is detailed in each problem set.

Points allocation

- Theory parts (initial assessment): 1 pt
- Application parts (initial assessment): 1 pt
- Theory parts (self-assessment): 1 pt

Problem categorization and main concepts covered Theory

- Projection: Problems 2.1, 2.2
- Approximation and hyperplanes: Problems 2.3, 2.4
- Matrics and linear maps: Problems 2.5-2.7

Application

- Projection and Fourier Series: Problem 2.8
- Projection with non-Euclidean norms: Problem 2.9
- Affine approximations: Problem 2.10

Optional

- Other vector spaces and more on linear maps: Problems 2.11, 2.12

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 (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 2.4 (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 2.5 (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 2.6 (Rank and nullspace)
OptM Problem 3.6.

Problem 2.7 (Linear dynamical systems)
OptM Problem 3.4.

APPLICATION PROBLEMS

Problem 2.8 (Approximating a square wave: The discrete-time Fourier series)
In this problem you will approximate a given discretized square wave signal using a *sinusoidal basis*. The following MATLAB function produces the discrete approximation of two periods of a period-10 square wave signal $x(t)$ where $t \in [0, 20]$, discretized in increments of 0.0001 seconds. In addition, the script also produces 30 orthogonal basis vectors that we will later use to approximate the given signal. The function returns two vectors `time_pos` and `sq_wave`, and a matrix `B_unnorm`. The vector `sq_wave` consists of the signal output for each time position specified in `time_pos` vector. The matrix `B_unnorm` includes 30 unnormalized row vectors that are of the same length as `sq_wave`.

```
function [time_pos, sq_wave, B_unnorm] = generate_data
    n_comps = 30; period = 10; fundFreq = 1/period;
    time_pos = 0:0.0001:2*period; harmonics = 2*(1:n_comps)-1;
    sq_wave = floor(0.9*sin(2*pi*fundFreq*time_pos))+.5;   %% generate the signal
    B_unnorm = sin(2*pi*fundFreq*(harmonics'*time_pos))/2; %% generate the basis
end
```


(a) Plot the square wave signal `sq_wave` against `time_pos`.

(b) How would you numerically test for the orthogonality of the basis vectors (rows of `B_unnorm`)? Plot the first 6 and 30th basis vectors against `time_pos` in separate sub-plots that share the same time axis.

(c) Normalize the given basis vectors to obtain an orthonormal basis. Project the square wave signal onto the normalized basis vectors using ℓ_2 projection and compute the projection coefficients. Arrange the basis vectors in the decreasing order of the magnitude of the projection coefficients. Approximate the square wave using the first $1, 2, 3, 4, 5, 6, 30$ basis vectors, and plot the approximations in separate sub-plots that share the same time axis. Plot the original signal on top of each approximation and compare.

Problem 2.9 (Projection with different norms)
Consider the vector $x = [3, 2]^T$ and the line defined by the set $A = \{v^{(0)} + tv \mid t \in \mathbb{R}\}$ where $v^{(0)} = [-2, -4]^T$ and $v = [-2, 5]^T$. The projection of x on to the affine set $\mathcal{A}$ under ℓ_p norm, which we denote by $y^{(p)}$, is the minimizer of optimization problem

$$\begin{aligned} \min_{y} \quad & \|x - y\|_p \\ \text{subject to} \quad & y \in \mathcal{A}. \end{aligned}$$

(a) Sketch the line $\mathcal{A}$ and indicate x on a 2-d map. Without solving for $y^{(2)}$, sketch the smallest norm ball under ℓ_2 norm that includes $y^{(2)}$, and mark the position of $y^{(2)}$.

(b) Write down the solution for $y^{(2)}$ in closed-form and solve for $y^{(2)}$.

(c) Repeat part (a) for $p = 1$ and $p = \infty$ cases.

(d) The solutions for $y^{(1)}$ and $y^{(\infty)}$ can not be expressed in closed-form, but they can be computed using *linear programs* (LPs). In this part we use the software package CVX which runs in MATLAB and is available at `http://cvxr.com/cvx/`. You may find the CVX package useful throughout this course. For more information, please check the user guide available at the above website.

As an example, after installing CVX, the following MATLAB function solves for $y^{(\mathtt{nrm})}$. Execute the function with given parameters and verify your result obtained in part (b).

```
function [y, r] = proj_cvx(x, v0, v, nrm)%% x, v0 and v must be column vectors
    objtv = @(y) norm(x-y, nrm);        %% objective is L_2 norm
    cvx_begin
        variable y(2)                   %% 2-d variable we are optimizing over
        variable t(1)                   %% real valued parameter that defines
        minimize(objtv(y))              %% defining the objective
        subject to
        v0 + t*v == y                   %% the projection y must be in set A
```

```
    cvx_end
    r = objtv(y);                          %% minimum value of the objective
end
```

(e) Use the given function to solve for $y^{(1)}$ and $y^{(\infty)}$. Include your code (MATLAB, Python, Julia) with the answers.

Problem 2.10 (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.

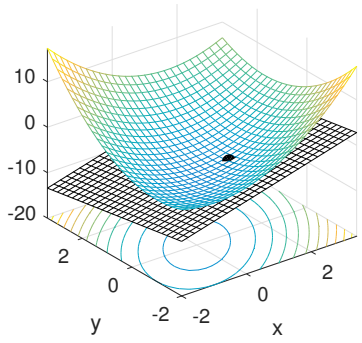

Figure 1: Example plot and approximating tangent plane.

OPTIONAL PROBLEMS

Problem 2.11 (Inner products and projection in function spaces)
(Note: This problem can be solved analytically in closed-form or numerically. The former is significantly more work. Both methods should yield the same final insight. Some tips about numerical methods are given following the problem statement.)

While the focus of this course is on finite (and mostly real) vector spaces, notions of inner products spaces and the approximation of a vector by its projection into a subspace, also hold for infinite-dimensional vector spaces where each vector is a function $f : \mathbb{R} \to \mathbb{R}$. In this problem let $C[-\pi, \pi]$ denote the set of continuous real-valued functions on $[-\pi, \pi]$. Observe that one can add and scale functions pointwise, thereby defining $C[-\pi, \pi]$ to be a vector space. We pick

$$\langle f, g \rangle = \int_{-\pi}^{\pi} f(x)g(x)dx \tag{1}$$

to be the inner product of two functions $f, g \in C[-\pi, \pi]$. One can verify that all properties of an inner product are satisfied by (1). The norm induced by this inner product is

$$\|f\| = \sqrt{\langle f, f \rangle} = \sqrt{\int_{-\pi}^{\pi} (f(x))^2 dx}.$$

In this problem you consider the the function $g : [-\pi, \pi] \to \mathbb{R}$ defined pointwise as $g(x) = \sin(x)$. Your task is to find the best approximation to g in the subspace of $C[-\pi, \pi]$ that consists of polynomials with real coefficients and degree at most five. We denote this subspace as $\mathcal{U} = \{u | u(x) = \alpha_0 + \alpha_1 x + \alpha_2 x^2 + \alpha_3 x^3 + \alpha_4 x^4 + \alpha_5 x^5, \alpha_i \in \mathbb{R} \ \forall \ i \in \{0, 1, \ldots 5\}\}$. By best we mean that the optimizing u^* is

$$u^* = \arg\min_{u \in \mathcal{U}} \|g - u\|^2.$$

where, to be explicit,

$$\|g - u\|^2 = \langle g - u, g - u \rangle = \int_{-\pi}^{\pi} (g(x) - u(x))^2 dx = \int_{-\pi}^{\pi} (\sin(x) - u(x))^2 dx.$$

In the following you are asked to apply the Gram-Schmidt procedure to produce an orthogonal basis. You are welcome to do this either numerically or analytically. By "numerically" we mean you are welcome to use a numerical integration technique to compute the integrations involved in the inner products. The integrations can be computed analytically to produce closed-form solutions, but it is a non-trivial amount of work. However you chose to do it, please do the following.

(a) First apply the Gram-Schmidt procedure using the inner production defined in (1) to the basis $\{v^{(0)}, v^{(1)}, v^{(2)}, v^{(3)}, v^{(4)}, v^{(5)}\}$ of $\mathcal{U}$ where, defined pointwise, $v^{(i)}(x) = x^i$, $i \in \{0, 1, 2, 3, 4, 5\}$ to produce an orthonormal basis $\{e^{(0)}, e^{(1)}, e^{(2)}, e^{(3)}, e^{(4)}, e^{(5)}\}$ where each $e^{(i)} \in C[-\pi, \pi]$.

(b) Next, using (1) and the formulas for ℓ_2 projection we developed in class (which apply again here), compute $\alpha_0, \ldots, \alpha_5$. (To check correctness note that, up to numerical precision, you should find that $\alpha_1 = 0.987862$.)

(c) You should see a sensical pattern to the even coefficients $\alpha_0, \alpha_2, \alpha_4$. What is the pattern you observe and why is it intuitively correct?

(d) Another often used polynomial approximation to the $\sin x$ function the Taylor approximation

$$\sin x \simeq x - \frac{x^3}{3!} + \frac{x^5}{5!}.$$

Plot your approximation from part (b) against the Taylor approximation. You should observe that your approximation looks much better over the entire interval $[-\pi, \pi]$. Where is the Taylor approximation accurate and where is it not accurate? What was it about the formulation of the ℓ_2 projection problem that makes your approximation better in the regions where the Taylor approximation is not good?

While you are welcome to use any method to perform the integration in 1, we describe below two numerical methods that you may use. The following MATLAB code snippet defines a few *anonymous functions* that help us compute the inner product of two functions. We use anonymous functions since they do not have to be defined in separate `.m` files. Please read MATLAB documentation for further information about this type of functions. Executing the following piece of code will print to console $\langle \sin, v^{(3)} \rangle$ and $\| \sin \|$. The purpose of each line of code is explained in the comments.

```
%% divide the interval [-pi, pi] into 1000 small intervals
eps = 1000;
xi = -pi:pi/eps:pi;
dx = pi/eps;

%% utility functions
intgrt = @(f,g) dot(f(xi),g(xi))*dx; % discrete approx. of the continous integral
inprod = @(f,g) intgrt(f,g);    % compute inner product between f and g
normf = @(f) sqrt(inprod(f,f)); % compute the norm of f

%% test
```

```
sx = @(x) sin(x);
v3 = @(x) x.^3;

inprod(sx,v3)                    % prints the inner product <sin(x), x^3>
normf(sx)                        % prints the norm of sin(x)
```

Alternatively, you may use MATLAB symbolic functions to implement $g, v^{(0)}, \ldots, v^{(5)}$, and use the built-in `int` command to perform the integration.

Problem 2.12 (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

$$f(x) = f\left(\sum_{i=1}^{n} \alpha_i v^{(i)}\right) \overset{(i)}{=} \sum_{i=1}^{n} \alpha_i f(v^{(i)}) \tag{2}$$

$$\overset{(ii)}{=} \sum_{i=1}^{n} \alpha_i \left[\sum_{k=1}^{m} \beta_k^{(i)} w^{(k)}\right], \tag{3}$$

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

$$f(x) = \left[w^{(1)} w^{(2)} \ldots w^{(m)}\right] \begin{bmatrix} \beta_1^{(1)} & \beta_1^{(2)} & \ldots & \beta_1^{(n)} \\ \beta_2^{(1)} & \beta_2^{(2)} & \ldots & \beta_2^{(n)} \\ \vdots & & \ddots & \vdots \\ \beta_m^{(1)} & \beta_m^{(2)} & \ldots & \beta_m^{(n)} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{bmatrix}. \tag{4}$$

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.

