University of Toronto
Department of Electrical and Computer Engineering

ECE367 MATRIX ALGEBRA AND OPTIMIZATION

Problem Set #1

Autumn 2020

Prof. S. C. Draper **Due:** 5pm (Toronto time) Friday, 25 September 2020

Homework policy: Problem sets must be turned by the due date and time. We will follow up with instructions on the submission procedure. Late problem sets will receive deductions for lateness. See information sheet for the problem set policy.

Problems are drawn from the course text "Optimization Models" by G. Calafiore and L. El Ghaoui, abbreviated as "OptM", others from the text "Introduction to Applied Linear Algebra", abbreviated "IALA", by S. Boyd and L. Vandenberghe. An electronic version of the latter is available on the authors' websites. In both cases we indicate the appropriate problem number and name.

"Non-graded", "graded" and "optional" problems: Problems are categorized as "non-graded", "graded", and "optional". Non-graded problems are **not** to be turned in. Your solution to graded problems **must** be turned in. Optional problems are a subset of the non-graded category.

- **"Non-graded" problems:** These are mostly "theory"-type problems. These problems are designed to give you deeper insight into the fundamentals of the ideas introduced in this class. Solutions will be provided for these non-graded problems after the turn-in date for the graded parts of the problem set. I reemphasize that it is really crucial for you to struggle with and attempt to solve these problems on your own. Many exam problems are likely to be along the lines of these problems. Some numerical/implementation-type problems may also be included in the non-graded category.
- **"Graded" problems:** Typically these questions are more numerical and design-oriented in nature. You must turn in your solutions to these questions. Details will follow. These questions are designed, in part, to expose you to the breadth of application of the ideas developed in class and to introduce you to useful numerical toolboxes. Note also that the allocation of marks amongst these problems is not necessarily uniform.
- **"Optional" problems:** Occasionally I will include "optional" problems. These are mostly a subset of the "non-graded" category. (Graded parts sometimes have "bonus" parts.) Sometimes these provide extra practice, sometimes these introduce connections or interesting extensions of the material. You can think of these as lower-priority problems. Work on these if you have the inclination, time and/or interest. I do expect you to review and understand the solutions to these problems, and will assume you have digested the solution to these problems when designing course exams.

Other items to note:

- For the numerical parts of the problem sets, while we phrase the problem in terms of MATLAB, and as I mentioned in lecture, you are welcome to use any numerical toolbox (MATLAB, Python, Julia, etc) that you wish. Data sets will be distributed using MATLAB. I have posted some introductions to MATLAB on the course website.

Problem Set #1 problem categories: A quick categorization by topic of the problems in this problem set is as follows:

Non-graded:

- Norms: Problems 1.1, 1.2
- Linear independence and orthogonality: Problems 1.3-1.5
- Inner products: Problems 1.6-1.7
- Approximation & projection: Problems 1.8

Optional non-graded:

- Inner products: Problem 1.9

Graded:

- Inner products: Problems 1.10 (has a bonus part)
- Approximation & projection: Problems 1.11, 1.12

Final note and recommendation: If it feels it has been a long time since you took linear algebra, and you are looking for additional resources and practice problems, I again direct you to "Introduction to linear algebra" by Gilbert Strang, Wellesley-Cambridge Press. This classic text is the text I used to learn linear algebra. Prof. Strang is a fantastic teacher. His website has lots of online resources, including old exams. Please see: http://math.mit.edu/∼gs/linearalgebra/.

NON-GRADED PROBLEMS

Problem 1.1 (Showing ℓ_1 and ℓ_∞ are both norms)
Show

(a) that the functions $\ell_1 : \mathbb{R}^n \to \mathbb{R}$ is a norm, and

(b) that $\ell_\infty : \mathbb{R}^n \to \mathbb{R}$ is a norm

by verifying that they satisfy all the properties of a norm, cf., OptM Definition 2.1.

Problem 1.2 (Norm inequalities)
OptM Prob. 2.6.
Note: This problem shows that each norm (l_1, l_2, l_∞) is both upper- and lower-bounded by each of the other norms to within a constant (dimension-dependent) factor.

Problem 1.3 (Linear independence of stacked vectors)
IALA Prob. 5.1.

Problem 1.4 (Orthogonality)
OptM Prob. 2.5.

Problem 1.5 (Gram-Schmidt algorithm)
IALA Prob. 5.6.

Problem 1.6 (Inner products)
OptM Prob. 2.4.

Problem 1.7 (Distance versus angle nearest neighbors)
IALA Problem 3.24.

Problem 1.8 (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.

OPTIONAL, NON-GRADED PROBLEMS

Problem 1.9 (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 $\mathcal{U} = \{u | u(x) = \alpha_0 + \alpha_1 x + \alpha_2 x^2 + \alpha_3 x^3 + \alpha_4 x^4 + \alpha_5 x^5, \alpha_i \in \mathbb{R} \ \forall \ i \in \{0, 1, \dots 5\}\}$. 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.

GRADED PROBLEMS

Problem 1.10 (Angles between word vectors)
In this problem you investigate how geometric concepts such as distance and angle can be applied to quantify similarity between text documents. Download the files `wordVecArticles.txt`, `wordVecTitles.txt`, `wordVecWords.txt` and `wordVecV.mat` from the course website. The first two files each have ten lines. Each line in the first file consists of the text of one Wikipedia article. The corresponding line of the second file is the title of the article. The last two files are described in detail below.

Denote by $\mathcal{D}$ the set of documents where the number of documents is $|\mathcal{D}|$. (In our dataset $|\mathcal{D}| = 10$). Let $\mathcal{W}$ denote the union of words in all articles, i.e., the lexicon of the set of documents. We denote the cardinality of $\mathcal{W}$ by $|\mathcal{W}|$. Assume the lexicon is ordered "lexiographically" (e.g., alphabetically) so that there is a one-to-one mapping from each word $w \in \mathcal{W}$ to an element of the index set $t \in [|\mathcal{W}|]$. Let $f_{\text{term}}(t, d)$ denote the number of times the word $w \in \mathcal{W}$ that is indexed as $t \in [|\mathcal{W}|]$ appears in the dth article where $d \in [|\mathcal{D}|]$. Note that $\sum_{t=1}^{|\mathcal{W}|} f_{\text{term}}(t, d)$ is the number of words (the length) of the dth article. We refer to $f_{\text{term}}(t, d)$ as the *term frequency* (really "term count").

For the first few parts of this problem you will be using a pre-processed $\mathcal{W}$ set and pre-computed $f_{\text{term}}(t, d)$ values. The pre-processed data appears in the files `wordVecWords.txt` and `wordVecV.mat`. The first file represents the set $\mathcal{W}$ where elements of $\mathcal{W}$ are listed line by line, for 1651 lines, i.e., $|\mathcal{W}| = 1651$. You can load the content in the second file into MATLAB by using command `load 'wordVecV.mat'`. After loading, you will see a matrix `V` of dimensions 1651×10. The value in the tth row and dth column of this matrix is $f_{\text{term}}(t, d)$. Use the provided data in `V` to answer parts (a) to (d) of this problem.

(a) Let the $|\mathcal{W}|$-dimensional vectors v_d, $d \in [|\mathcal{D}|]$ be defined as $v_d = (f_{\text{term}}(1, d), f_{\text{term}}(2, d), \ldots, f_{\text{term}}(|\mathcal{W}|, d))$. Using v_d to represent the dth document, which two articles are closest in Euclidean distance (smallest distance)? Which two are closest in angle distance (smallest angle)? Are they the same pair, if not, what could be a reason for them being different? You may find the `pdist` command in MATLAB useful for computing pairwise Euclidean and angle distances of vectors.

(b) In this part let the $|\mathcal{W}|$-dimensional *normalized* vectors $\tilde{v}_d$, $d \in [|\mathcal{D}|]$ be defined as $\tilde{v}_d = v_d / \sum_{t=1}^{|\mathcal{W}|} f_{\text{term}}(t, d)$, where the v_d are defined as in the previous part. Using $\tilde{v}_d$ to represent the dth document, which two articles are closest in Euclidean distance (smallest distance)? Which two are closest in angle distance (smallest angle)? Are your answers the same as in the previous part? What would be a reason for using this normalization?

Now, let $f_{\text{doc}}(t) = \sum_{d=1}^{|\mathcal{D}|} \mathbb{I}[f_{\text{term}}(t, d) > 0]$ where $\mathbb{I}(\cdot)$ is the indicator function taking value one if the clause is true and zero else. The function $f_{\text{doc}}(t)$ counts in how many documents the tth word appears. We refer to $f_{\text{doc}}(t)$ as the *document frequency*.

We combine the term and document frequency definitions into what is called the *term frequency-inverse document frequency score* (TF-IDF), defined as

$$w(t,d) = \frac{f_{\text{term}}(t,d)}{\sum_{t=1}^{|\mathcal{W}|} f_{\text{term}}(t,d)} \sqrt{\log\left(\frac{|\mathcal{D}|}{f_{\text{doc}}(t)}\right)}.$$

Note, the denominator of the log is never zero since, by definition, each term appears in at least one document.

(c) Now let the $|\mathcal{W}|$-dimensional vectors w_d, $d \in [|\mathcal{D}|]$ be defined as $w_d = (w(1,d), w(2,d), \ldots, w(|\mathcal{W}|,d))$. Using w_d to represent the dth document, which two articles are closest in Euclidean distance (smallest distance)?

(d) What might be a reason for using the "inverse document frequency" adjustment? What is the adjustment doing geometrically?

BONUS PART: The following part (e) is a *bonus* part. It's worth is equal to 50% of parts (a)-(d). Your score on this part will be added to your overall score for the problem set, which will then be the max of your score or 100%. (This means that if you get all other parts correct your score will not be increased through completion of this problem part.)

(e) In the previous parts of the problem you used the pre-processed $\mathcal{W}$ set and pre-computed $f_{\text{term}}(t,d)$ values provided in `wordVecV.mat` and term indexes in `wordVecWords.txt`. In this part of the problem you will reproduce your results from (a) to (d) without using this pre-computed data. Specifically, start from the raw text files `wordVecArticles.txt` and `wordVecTitles.txt`, and write code to obtain $\mathcal{W}$ and $f_{\text{term}}(t,d)$. As always, you are welcome to use whichever software language you wish to solve this problem. We note that Python is particularly well suited to text processing. For example, you may find the snippet of Python code following the problem statement useful. This snippet loads in data from the given text file and stores it in the variable `articles`. It also counts the number of word occurrences in the first article and stores the resulting (word, count) pairs in the variable `wordcounts`. You could use this as a starting point in the generation of the vectors we provide in `wordVecV.mat` and then calculate angles and distances in MATLAB. Alternately, you could show how to complete the entire processing pipeline in Python. Be sure to include a printout of your code.

```
from collections import Counter
articles = [line.rstrip('\n') for line in open('wordVecArticles.txt')]
wordcounts = Counter(articles[0].split())
```

Problem 1.11 (First-order approximation of functions)
In this exercise you will write MATLAB (or Python, Julia, etc) code to plot linear approximations

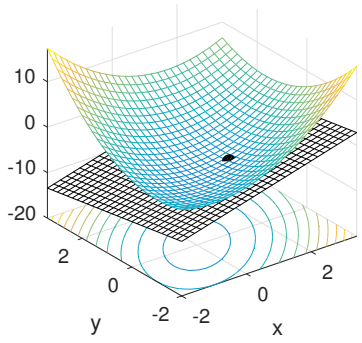


Figure 1: Example plot and approximating tangent plane.

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 1.12 (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.

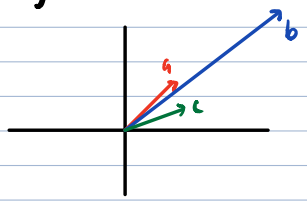
As an example, after installing CVX, the following MATLAB function solves for $y^{(\texttt{nrm})}$. Execute the function with given parameters and verify your result obtained in part (b).

```
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.

Question 10

a) The two articles closest in distance are v_7 and v_8 with a distance of 24.718 units. The two articles closest in angle distance are v_4 and v_{10} with a distance of 0.533 radians. The two angles are not the same pair which is a reasonable result.



As shown in the diagram, vector a and b are closest in angle while vector a and c are closest in distance, which is the same situation as the articles.

b) The two articles closest in distance are v_4 and v_{10} with a distance of 0.0643 units. The two articles closest in angle distance are also v_4 and v_{10} with a distance of 0.533 radians. The distance results are different, which can be expected, since each vector is being scaled based on the size of the article, but not as a Euclidian norm, which can cause different results as shown. The angle distance stayed the same as expected since the magnitude of the vectors changed which means the relative angles will remain unchanged.

c) The two articles closest in distance are w_4 and w_{10} with a distance of 0.05 units when using log base 2.

d) One reason for using the TF-IDF score is to scale each article relative to how "unique" the choice of words are:

more "unique" words in article → lower scaling → larger vector

Geometrically, if we look at the log term, the numerator is always constant, only the denominator changes. $f_{doc}(t)$ represents how many documents the t^{th} word appears in. This means that if multiple vectors have some magnitude in the t^{th} dimension (less "unique") they are scaled at a larger value and lead to a smaller normalized vector.

Question 11

a) $\nabla f_1 = [2 \quad 3]^T$

$\nabla f_2 = [2x - y \quad 2y - x]^T$

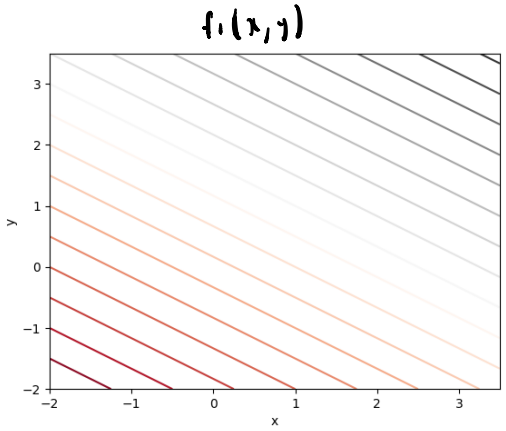
$\nabla f_3 = [\cos(y-5) - (y-5)\cos(x-5) \quad (5-x)\sin(y-5) - \sin(x-5)]^T$

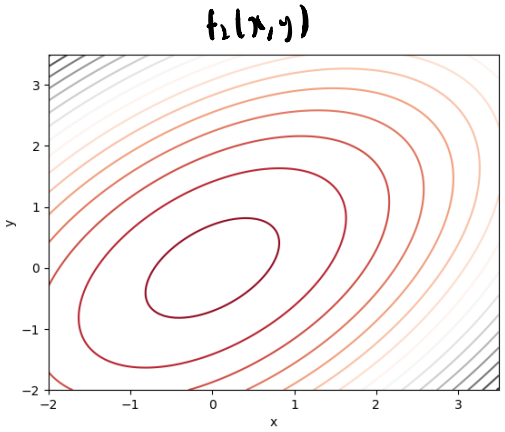
b) $\nabla f_1(1,0) = [2 \quad 3]^T$

$\nabla f_2(1,0) = [2 \quad -1]^T$

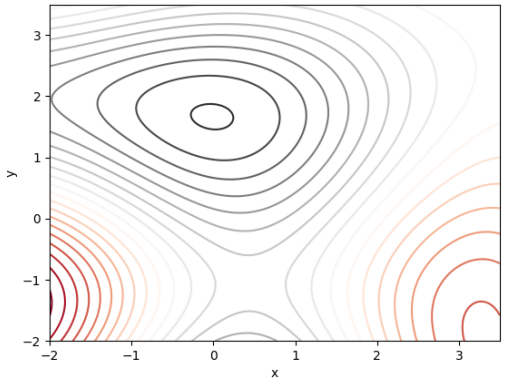
$\nabla f_3(1,0) = [\cos(-5) + 5\cos(-4) \quad 4\sin(-5) - \sin(-4)]^T$

$= [-2.98 \quad 3.08]$





$f_3(x,y)$



c) $i=1$: $T_1(x,y) = f(x_0,y_0) + f_x(x_0,y_0)(x-x_0) + f_y(x_0,y_0)(y-y_0)$

$T_1(x,y) = 3 + 2(x-1) + 3(y)$

$= 2x + 3y + 1$

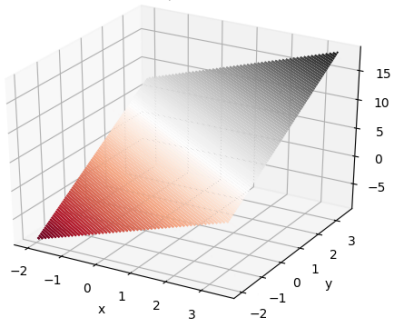
$i=2$: $T_2(x,y) = 1 - 5 + 2(x-1) - 1(y)$

$= 2x - y - 6$

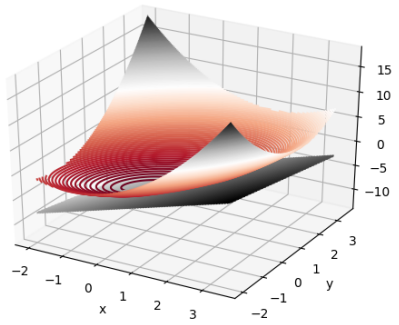
$i=3$: $T_3(x,y) = -4\cos(-5) + 5\sin(-4) + [\cos(-5) + 5\cos(-4)]$
$(x-1) + [4\sin(-5) - \sin(-4)](y)$

$= [\cos(-5) + 5\cos(-4)]x + [4\sin(-5) - \sin(-4)]y -$
$5\cos(-5) + 5\sin(-4) - 5\cos(-4)$

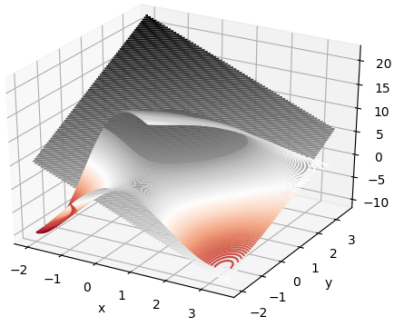
$f_1(x,y)$



$f_2(x,y)$

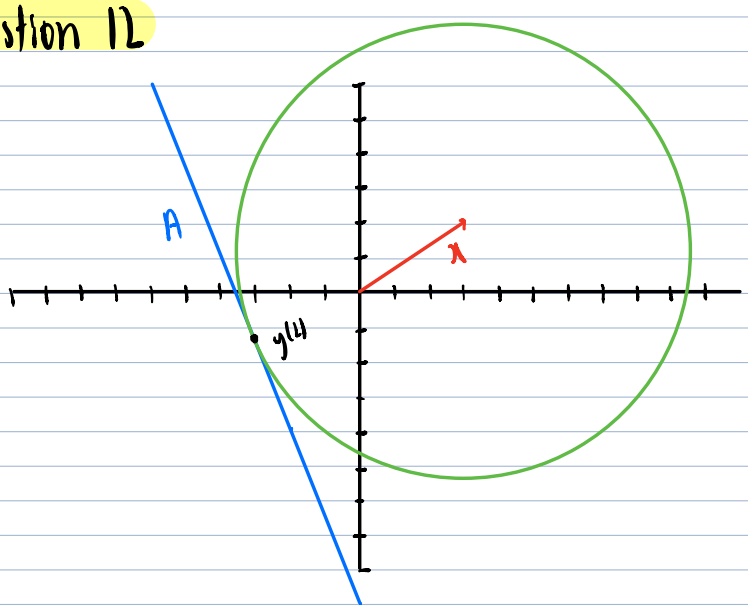


$f_3(x, y)$



Question 12

a)



b) ① Translate x and A by $-x^{(0)}$

$A = \{t[-2\ -4]^T \mid t \in \mathbb{R}\}$ $\quad x = [3\ 2]^T - [-2\ -4]^T$

$= [5\ 6]$

② Project $x - x^{(0)}$ onto S

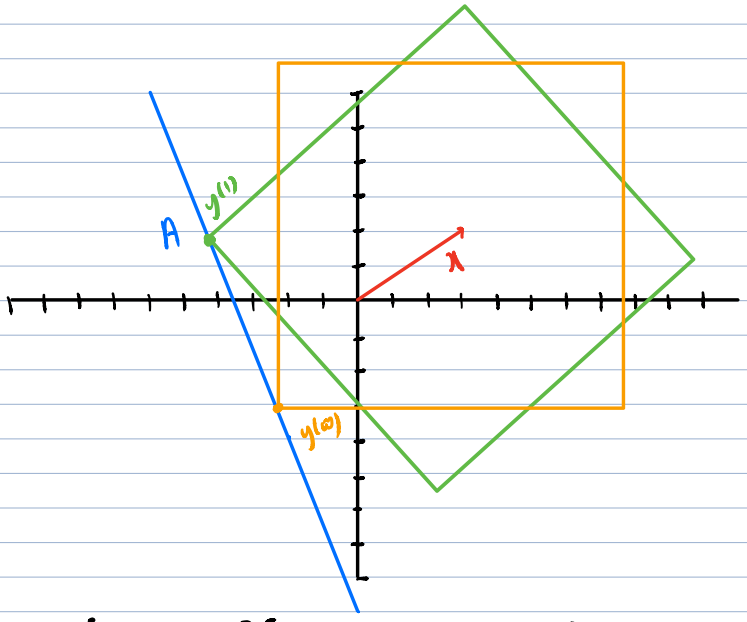
$\text{proj} = \frac{[5\ 6]\cdot[-2\ 5]}{(2^2+5^2)} \cdot [-2\ 5]^T$

$= \left[-\frac{40}{29}\ \ \frac{100}{29}\right]^T$

③ Shift result

$$y^{(2)} = \left[-\frac{98}{29} \quad \frac{100}{29}\right]^T + \left[-2 \quad -4\right]^T$$
$$= \left[-\frac{98}{29} \quad -\frac{16}{29}\right]^T$$
$$= \frac{2}{29}\left[-49 \quad -8\right]^T$$

c)



e) $y^{(1)} = [-4.4 \quad 2]^T$ $\qquad$ $y^{(\infty)} = [-2.2857 \quad -3.2857]^T$

$r^{(1)} = 7.4$ $\qquad$ $r^{(\infty)} = 5.2857$

Question 10 (Python)

Note: in order to prepare data.csv which contains the data in wordVecV.mat file, I first imported the datafile into a Matlab file and then exported it as a csv file. I then imported the data.csv file into a python file as shown below.

Python Setup:

```
import numpy as np
import math

V = np.loadtxt('data.csv', delimiter=',')

v = np.zeros((10,1651))

for i in range(0,10,1):
    for j in range(0,1650,1):
        v[i,j] = V[j,i]
```

Part A:

```
mindist = 100
distvals = [0,0]
minang = 180
angvals = [0,0]

for i in range(0,10,1):
    for j in range(i+1,10,1):
        dist = np.linalg.norm(v[j]-v[i])

        unit_vi = v[i] / np.linalg.norm(v[i])
        unit_vj = v[j] / np.linalg.norm(v[j])

        dot = np.dot(unit_vi, unit_vj)
        ang = np.arccos(dot)
        if dist < mindist:
            mindist = dist
            distvals = [i, j]
        if ang < minang:
            minang = ang
            angvals = [i, j]

print("Minimum distance is: " + str(mindist) + " between v" + str(distvals[0]+1) + "
and v" + str(distvals[1]+1))
print("Minimum angle is: " + str(minang) + " between v" + str(angvals[0]+1) + " and v"
+ str(angvals[1]+1))
```

Part B:

```
mindist = 100
distvals = [0,0]
minang = 180
angvals = [0,0]

for i in range(0,10,1):
    for j in range(i+1,10,1):
        sum_vi = np.sum(v[i])
        sum_vj = np.sum(v[j])
        norm_vi = v[i]/sum_vi
        norm_vj = v[j]/sum_vj

        dist = np.linalg.norm(norm_vi - norm_vj)

        unit_vi = norm_vi / np.linalg.norm(norm_vi)
        unit_vj = norm_vj / np.linalg.norm(norm_vj)
        dot = np.dot(unit_vi, unit_vj)
        ang = np.arccos(dot)
        if dist < mindist:
            mindist = dist
            distvals = [i, j]
        if ang < minang:
            minang = ang
            angvals = [i, j]

print("Minimum distance is: " + str(mindist) + " between v" + str(distvals[0]+1) + "
and v" + str(distvals[1]+1))
print("Minimum angle is: " + str(minang) + " between v" + str(angvals[0]+1) + " and v"
+ str(angvals[1]+1))
```

Part C:

```
w = np.zeros((10,1651))
for i in range(0,10,1):
    for j in range(0,1651,1):
        fterm = v[i,j]
        sum_fterm = np.sum(v[i])
        fdoc = 0
        for k in range(0,10,1):
            if V[j,k] > 0:
                fdoc = fdoc + 1
        w[i,j] = (fterm/sum_fterm) * (math.log(10/fdoc, 2.712)) ** 0.5

mindist = 100
distvals = [0,0]
```

```
for i in range(0,10,1):
    for j in range(i+1,10,1):
        dist = np.linalg.norm(w[j]-w[i])
        if dist < mindist:
            mindist = dist
            distvals = [i, j]

print("Minimum distance is: " + str(mindist) + " between w" + str(distvals[0]+1) + "
and w" + str(distvals[1]+1))
```

Part E:

```
from collections import Counter
import numpy as np

articles = [line.rstrip('\n') for line in open ('wordVecArticles.txt')]

words = []
for i in range(0,10,1):
    wordcounts = Counter(articles[i].split())
    for j in wordcounts:
        if j not in words:
            words.append(j)

w = sorted(words)
# print(w)

v = np.zeros((10, 1651))
for i in range(0, 10, 1):
    wordcounts = Counter(articles[i].split())
    for j in range(0,1651,1):
        if w[j] in w:
            v[i,j] = wordcounts[w[j]]
        else:
            v[i,j] = 0
```

This code is the setup for the vectors ‘W’ and ‘v’ which can then be used with the code from parts A to C to obtain the same results.

Question 11 (Python)

Python Setup:

```
import numpy as np
import matplotlib.pyplot as plt
```

```
def f1(x,y):
    return 2*x + 3*y + 1

def f2(x,y):
    return x**2 + y**2 - x*y - 5

def f3(x,y):
    return (x-5)*np.cos(y-5) - (y-5)*np.sin(x-5)
```

Part B:

```
x = np.linspace(-2,3.5,200)
y = np.linspace(-2,3.5,200)

X, Y = np.meshgrid(x,y)
Z1 = f1(X,Y)
Z2 = f2(X,Y)
Z3 = f3(X,Y)

plt.contour(X, Y, Z1, 20, cmap='RdGy')
plt.xlabel('x')
plt.ylabel('y')
plt.show()

plt.contour(X, Y, Z2, 20, cmap='RdGy')
plt.xlabel('x')
plt.ylabel('y')
plt.show()

plt.contour(X, Y, Z3, 20, cmap='RdGy')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

Part C:

```
fig = plt.figure()
ax = plt.axes(projection = '3d')
ax.contour3D(X,Y,Z1,150,cmap='RdGy')
ax.contour3D(X,Y,2*X + 3*Y + 1,150,cmap='RdGy')
plt.xlabel('x')
plt.ylabel('y')
plt.show()

fig = plt.figure()
ax = plt.axes(projection = '3d')
```

```
ax.contour3D(X,Y,Z2,150,cmap='RdGy')
ax.contour3D(X,Y,2*X - Y - 6,150,cmap='binary')
plt.xlabel('x')
plt.ylabel('y')
plt.show()

fig = plt.figure()
ax = plt.axes(projection = '3d')
ax.contour3D(X,Y,Z3,150,cmap='RdGy')
ax.contour3D(X,Y,X*(5*np.cos(4)+np.cos(5)) + Y*(np.sin(4)-4*np.sin(5)) - 5*np.sin(4) -
5*np.cos(5) - 5*np.cos(4),150,cmap='binary')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

Question 12 (MATLAB)

Part E:

```
x = [3;2]
v0 = [-2;-4]
v = [-2;5]

[y,r] = proj_cvx(x,v0,v,1)
[y,r] = proj_cvx(x,v0,v,inf)
```

```
function [y,r] = proj_cvx(x,v0,v,nrm)
    y = [0;0];
    objtv = @(y) norm(x-y,nrm);
    cvx_begin
        variable y(2)
        variable t(1)
        minimize(objtv(y))
        subject to
        v0 + t*v == y;
    cvx_end
    r = objtv(y);
    disp(y)
    disp(r)
end
```