ISyE524: Introduction to Optimization
Problem Set #4
Due: *Tuesday* April 7, at 11:59PM

Notes and Deliverables:

- Submit solutions to all problems in the form of a single PDF file of the IJulia notebook to the course dropbox. (If you do not submit as a single PDF you will lose points.)
- Be sure that your answers (including any pictures you include in the notebook) are in the correct order. (If you do not submit the problems in order, you will lose points.)

1 Portfolio Optimization

We have been given the opportunity to help manage the 524Fund, a portfolio/hedge fund with 1 billion dollars of assets for the next year. We are given the expected value of the return μ and covariance matrix Q of the returns between 255 assets for the next year in which the 524Fund may invest.

The 524Fund currently holds 40 of these assets, and the asset holdings (in percentage of the portfolio) are given in a vector h. The 524Fund has changed management, and we have been given the directive to transition the fund to match/track a new benchmark, the Linde50, whose percentages are given in the vector b.

When fund managers wish to track a benchmark b, they measure of closeness of a portfolio x to the benchmark with "active risk", defined as

$$\sqrt{(x-b)^T Q (x-b)}$$

The current active risk of the 524Fund (holdings h) with respect to the Linde50 (benchmark b) is far too high—almost 33%.

We will write some optimization problems that will reduce the active risk, but do so in a way that minimizes the impact of the transactions.

1-1 Problem Write an optimization problem that will minimize the total number of dollars that must be bought or sold in order to reduce the active risk of the (new) 524Fund holdings to be ≤ 10. (The transaction size is the amount bought + the amount sold)

hint: You should have (at least) variables for the new holdings in the 524Fund, variables for how much of each asset to buy, and how much of each asset to sell.

1-2 Problem Using the data in the given files, `folio_mean.csv`, `folio_cov.csv`, `folio_holding_benchmark.csv`, implement your model from Problem 1-1 to find the minimum total transaction size.

The code below (also available as `portfolio_market_impact-start.ipynb`) will read the data and also transform the given data for this problem.

Julia Code

```
using DataFrames, CSV, LinearAlgebra, NamedArrays

```

```
df = CSV.read("folio_mean.csv", DataFrame, header=false, delim=',')
(n,mmm) = size(df)

# Weekly numbers to annual (and flip returns to make more positive)
mu = -100/7*365*Vector(df[1:n,1])

df2 = CSV.read("folio_cov.csv",DataFrame,header=false,delim=',')

# Weekly numbers to annual, also reduce the risk a bit
Q = 0.5* (100/7*365)^2 * Matrix(df2)

df3 = CSV.read("folio_holding_benchmark.csv", DataFrame, header=false, delim=',')

h = Vector(df3[1:n,1])
b = Vector(df3[1:n,2])

# Current tracking risk
benchmark_return = mu'b

# Current holdings return
holdings_return = mu'*h

# Current tracking risk
active_risk = sqrt((h-b)'Q*(h-b))

println("Benchmark expected return: ", benchmark_return)
println("Holdings expected return: ", holdings_return)
println("Active Risk: ", active_risk)
```

Be sure to print the following statistics after solving your optimization model:

- The total number of dollars transacted. (You should multiply your objective value by \$1B if you are using portfolio fraction as the decision-variable units)
- The average size of the largest 10 transactions (\$)
- The portfolio expected return (%)
- The portfolio active risk (%)

1-3 Problem The 524Fund management has said that your transaction sizes from Problem 1-2 are too large—trades of that size would negatively impact the prices of the assets—and they would like for you to plan a different set of trades with less market impact.

Write an optimization model that minimizes the total *market impact* of the trades necessary to get the active risk to below 10%. The 524Fund manages total market impact as

$$\sum_{i=1}^{n} y_i^{3/2},$$

where y_i is the size of the transaction for asset i (again, the size of the transaction is the amount bought + amount sold).

You have been instructed to model this using the epigraph of the individual nonlinear functions:

$$\sum_{i=1}^{n} t_i,$$

where $t_i \geq y_i^{3/2}\ \forall i = 1, 2, \ldots n$, and rotated second order cone constraints. Be sure to clearly define your new decision variables, including any auxiliary variables you need to model the cone constraints.

1-4 Problem Implement your model from Problem 1-3 to find the new suggested set of holdings. Print the same set of statistics as Problem 1-2.

- The total number of dollars transacted. (You should multiply your objective value by \$1B)
- The average size of the largest 10 transactions (\$)
- The portfolio expected return (%)
- The portfolio active risk (%)

2 Making An Indefinite Matrix Definite

The following matrix is indefinite:

$$Q = \begin{pmatrix} 0 & 0 & -2 & -4 & 0 & 1 \\ 0 & 1 & -1 & -1 & 3 & -4 \\ -2 & -1 & -1 & -5 & 7 & -4 \\ -4 & -1 & -5 & -3 & 7 & -2 \\ 0 & 3 & 7 & 7 & -1 & -2 \\ 1 & -4 & -4 & -2 & -2 & 0 \end{pmatrix}$$

2-1 Problem Use Julia to compute the eigenvalues of Q and print them out. You will need to use the LinearAlgebra package, and then `eigen(Q)` returns the eigenvalues and eigenvectors of Q

2-2 Problem It is well-known that if a real symmetric matrix $Q \prec 0$ with minimum eigenvalue $\lambda_1 < 0$, then the matrix obtained by subtracting λ_1 from the diagonal is PSD, i.e. $Q - \lambda_1 I \succeq 0$. Thus if we add a *total* of $6 \times |\lambda_1|$ to the diagonal of Q, the results matrix is PSD.

But we could add *less* to make the matrix PSD. Write a semidefinite program that will find a ***non-negative*** diagonal matrix D such that $Q + D \succeq 0$ and $\sum_{i=1}^{6} D_{ii}$ is minimized?

2-3 Problem Implement your SDP from Problem 2-2 in Julia and JuMP. Solve the resulting SDP with the SCS solver. Print the (diagonal) entries of your matrix D and their sum.

2-4 Problem Now we would like to find a (not-necessarily diagonal) matrix X such that the matrix $Q + X$ is positive-semidefinite, and the total (absolute-value) of the sum of the entries of X is as small as possible. Write a semidefinite program to find a symmetric matrix X such that $\sum_{i=1}^{6} \sum_{j=1}^{6} |X_{ij}|$

is as small as possible, and $Q + X \succeq 0$. Note here that unlike problem 2-2, we are not requiring the individual elements of our matrix X to be non-negative, but we want to minimize the sum of their absolute values.

2-5 Problem Implement your model from Problem (2-4) in Julia and JuMP. Print your matrix X and the sum of the absolute values of its entries.

3 Checked Luggage

Suppose that you trying to pack as many souvenirs as possible to bring home from your trip, but your suitcase has a limited capacity. It can hold a maximum of 30 pounds of weight and 15 gallons of volume. The weights and volumes are as follows:

Souvenir number	1	2	3	4	5	6	7	8	9	10
Weight (lbs)	5	6	7	6	4	6	7	3	8	5
Volume (gal)	2	4	5	3	3	2	3	1	2	4

3-1 Problem Write an integer programming model to determine which souvenirs to pack.

3-2 Problem Implement your model from Problem 3-1 in JuMP and display the selected items.

3-3 Problem Now suppose that you have a very demanding partner, and they are not happy with the answer obtained when you have just 30 pounds max of weight and 15 gallons of volume. They would like to know *all* their options. That is, for each possible (integer) value of (weight, volume) pairs, how many items can you pack? You can get it to look fancy if you use the code below, and then your partner will be very impressed!

(Make an array of size #volumes, and columns #weights, and then fill in the matrix entry with the value you would like to plot (the objective value))

This code will make a 3D bar plot of the array in vals

Julia Code

```
using PyPlot

function plot3d(vals)
    nrows = size(vals,1)
    ncols = size(vals,2)
    _x = 1:nrows
    _y = 1:ncols
    # Make a meshgrid
    x = _x' .* ones(ncols)
    y = ones(nrows)' .* _y
    # Unravel
    x = vec(x)
    y = vec(y)
    # Heights
    dz = vec(vals')
```

```
        z = zeros(length(dz))

        dx = 0.4
        dy = 0.4

        bar3D(x,y,z,dx,dy,dz)
        # Only needed in vscode
        display(gcf())
    end
    ;
```

4 Steam Power

A power plant has three boilers. If a given boiler is operated, it can be used to produce a quantity of steam (in tons) between the minimum and maximum given in the table below. This table also gives the cost and carbon emissions per ton of steam produced from each boiler.

Boiler Number	Min Steam (Tons)	Max Steam (Tons)	Cost/Ton ($)	Carbon/ton (lbs)
B1	400	1000	10	20
B2	200	900	8	30
B3	300	800	7	35

Steam from the boilers is used to produce power on three turbines. If operated, each turbine can process an amount of steam (in tons) between the minimum and maximum given in the table below. The table also gives the amount of power (in KwH) produced and the procesing cost per ton of steam processed for each turbine. The power plant must produce 8000 KwH of power.

Turbine	Min (tons)	Max (tons)	KwH per Ton of Steam	Processing Cost per Ton ($)
T1	300	600	4	2
T2	500	800	5	3
T3	600	900	6	4

4-1 Problem Formulate an integer program to help the power plant determine how to meet the requirements at minimum cost.

4-2 Problem Implement your model from Problem 4-1 in Julia and JuMP to determine the optimal way to meet power plant demand.

5 524 Cheese

The 524 Cheese Company (524CC) produces two types of cheese: Colby and Mozzarella. Currently, 524CC has 120 pounds of Colby and 80 pounds of Mozzarella in stock.

Processing a pound of milk costs $0.20 and yields 0.5 pounds of Colby cheese and 0.4 pounds of Mozzarella. Either type of cheese can be stored from week to week, but milk cannot be stored. It costs

$0.25 per pound of cheese (either type) that is stored from one week to the next, and at most 500 pounds (total) can be stored at any time. Colby cheese is sold for $2.50 per pound, and Mozzarella cheese is sold for $3 per pound.

The price of milk is a complicated function of future (per unit) prices and delivery costs. There is a fixed cost of f_t if delivery of milk occurs in week t, and the per unit price of milk in week t is p_t

The table below shows the maximum amount of each type of cheese 524CC could sell in the next 8 weeks as well as the fixed and variable costs for milk. Demand that is not met in the week it occurs cannot be carried over to a later week (i.e., no backlogging is allowed).

Week	Colby	Mozzarella	f_t	p_t
1	150	200	1000	1
2	200	400	1400	0.8
3	225	300	800	0.8
4	50	500	1200	1.2
5	400	100	600	1.2
6	50	500	1000	1.0
7	300	200	400	1.5
8	200	350	800	0.6

5-1 Problem Write an integer programming model that maximizes the 524CC profit over the planning horizon.

Hint: Your model should probably have (at least) the following decision variables:

- Pounds of milk to buy each week
- Pounds of colby to sell each week
- Pounds of mozerrella to sell each week
- Inventory of colby at end of each each
- Inventory of mozerelaa at end of each week
- Indicator if whether or not milk was bought each week

5-2 Problem Implement your model from Problem 5-1 in JuMP to find the optimal purchase and production schedule for 524CC.

Print out the following information:

- Total maximum profit
- The weeks that milk is purchased
- The total inventory of cheese at the end of each week.

You can use the code here to load the data for the problem. (Also available on Canvas)

Julia Code

```
# initial colby (pounds)
i_colby = 120

# initial mozerella (pounds)
i_moz = 80

# Demand for colby cheese (pounds)
d_colby = [150 200 225 50 400 50 300 200]
# Demand for mozerella cheese (pounds)
d_moz = [200 400 300 500 100 500 200 350]

# Fixed cost for a delivery of milk ($)
fc_milk = [1000 1400 800 1200 600 1000 400 800]

# Per-unit cost for milk ($/pound)
p_milk = [1 0.8 0.8 1.2 1.2 1.0 1.5 0.6]

# Processing cost of milk ($/pound)
milk_proc_cost = 0.2

# inventory cost ($/pound)
cheese_inventory_cost = 0.25

# max total inventory (pounds)
max_inventory = 500

# colby price ($/pound)
colby_price  = 2.5

# mozzerella price ($/pound)
moz_price = 3.0

# colby/milk
colby_per_milk = 0.5

# moz/milk
moz_per_milk = 0.4

# Number of time periods
T = 8

# The maximum milk you would buy in any period is surely no more than the total demand for cheese
max_milk = [sum(d_colby[1:t])+sum(d_moz[1:t]) for t in 1:T]

```