Convex-Optimization

1) There are three food items available, corn, milk, and bread. The table contains, the cost per serving, the amount of Vitamin A per serving, and the number of calories per serving for each food item. Also, there are restrictions on the total number of calories (between 2000 and 2250) and the total amount of Vitamin A (between 5000 and 50,000) intake in the diet.The goal of the diet problem is to select a set of food items that will satisfy a set of daily nutritional requirement at minimum cost. The maximum number of servings is 10 per food item. Table 1 shows all the content per serving.

Food	Cost	Vitamin A	calories
Corn	0.18 USD	107	72
Milk	0.23 USD	500	121
Wheat Bread	0.05 USD	0	65

TABLE I

Solution:
Consider,

Description	**Parameter**	**Value**
Cost per serving	**c**	$\begin{pmatrix}0.18\\0.23\\0.05\end{pmatrix}$
Vitamin A per serving	**v**	$\begin{pmatrix}107\\500\\0\end{pmatrix}$
Calories per serving	**u**	$\begin{pmatrix}72\\121\\65\end{pmatrix}$
No. of servings	**x**	$\begin{pmatrix}x\\y\\z\end{pmatrix}$

TABLE II

Objective function

$$z = \min_{\mathbf{x}} \mathbf{c}^T\mathbf{x} \tag{1}$$

Constraints

$$5000 \le 107x + 500y + 0z \le 50000 \tag{2}$$
$$2000 \le 72x + 121y + 65z \le 2250 \tag{3}$$
$$0 \le x \le 10 \tag{4}$$
$$0 \le y \le 10 \tag{5}$$
$$0 \le z \le 10 \tag{6}$$

Writing all the constraints in the matrix form

$$\mathbf{px} = \mathbf{q} \tag{7}$$

$$\begin{pmatrix} 107 & 500 & 0\\ -107 & -500 & 0\\ 72 & 121 & 65\\ -72 & -121 & -65\\ 1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\\ -1 & 0 & 0\\ 0 & -1 & 0\\ 0 & 0 & -1 \end{pmatrix} \mathbf{x} = \begin{pmatrix} 50000\\ -5000\\ 2250\\ -2000\\ 10\\ 10\\ 10\\ 0\\ 0\\ 0 \end{pmatrix} \tag{8}$$

By providing the objective function and constraints to cvxpy, we get the optimal cost (z) and optimal no.of servings (**x**).
cvxpy code,

https://github.com/gadepall/EE5606−optimization/codes/opt_1.py

From cvxpy, we get

$$z = 3.16\text{USD} \tag{9}$$

$$\mathbf{x} = \begin{pmatrix}2\\10\\10\end{pmatrix} \tag{10}$$

2) Funding an expense stream. Your task is to fund an expense stream over n time periods. We consider an expense stream $\mathbf{e} \in R^n$, so that e_t is our expenditure at time t. One possibility for funding the expense stream is through our bank account. At time period t, the account has balance b_t and we withdraw an amount w_t. The value of our bank account accumulates with an

interest rate ρ per time period, less withdrawals

$$b_{t+1} = (1+\rho)b_t - w_t$$

We assume the account value must be nonnegative, so that $b_t \geq 0$ for all t. We can also use other investments to fund our expense stream, which we purchase at the initial time period t = 1, and which pay out over the n time periods. The amount each investment type pays out over the n time periods is given by the payout matrix **P**, defined so that P_{tj} is the amount investment type j pays out at time period t per dollar invested. There are m investment types, and we purchase $x_j \geq 0$ dollars of investment type j. In time period t, the total payout of all investments purchased is therefore given by $(Px)_t$. In each time period, the sum of the withdrawals and the investment payouts must cover the expense stream, so that

$$w_t + (Px)_t \geq e_t$$

for all t = 1, . . . , n. The total amount we invest to fund the expense stream is the sum of the initial account balance, and the sum total of the investments purchased: $b_1 + 1^T\mathbf{x}$.
Using the data in **expense stream data.***,Show that the minimum initial investment that funds the expense stream can be found by solving a convex optimization problem.
Solution:
Consider,

Description	Parameter	Value
Payout Matrix	$\mathbf{P}_{nxm}$	from expense stream data.py
expense stream	$\mathbf{e}_{nx1}$	from expense stream data.py
Interest rate	ρ	from expense stream data.py
No. of investments	m	from expense stream data.py
Time period	n	from expense stream data.py
Present Bank balance	$\mathbf{b}_{nx1}$	?
Bank withdrawals	$\mathbf{w}_{nx1}$	?
Investments purchased	$\mathbf{x}_{mx1}$	?
Total payable for investments purchaced	$\mathbf{Px}_{nx1}$	?

TABLE III

Given to minimize the inital investment.
Objective Function,

$$Z = \min_{b1,\mathbf{x}} b_1 + 1^T\mathbf{x} \tag{11}$$

Constraints,

$$\mathbf{b} \geq 0 \tag{12}$$
$$\mathbf{x} \geq 0 \tag{13}$$
$$\mathbf{w} + \mathbf{Px} \geq \mathbf{e} \tag{14}$$
$$b_{t+1} = (1+\rho)b_t - w_t. \tag{15}$$

Solution of the above objective function with the constraints is obtained by CVXPY.
CVXPY code,

https://github.com/gadepall/EE5606–optimization/codes/opt_2.py

https://github.com/gadepall/EE5606–optimization/codes/expense_stream _data.py

The optimal initial investment Z = 177.51

3) We are tasked with designing a box shaped with width w, height h, and depth d. We are given that the total wall area is at most 200 square units, the total floor area is at most 60 square units, and the aspect ratios (h/w ans d/w) are at least 0.8 and at most 1.2. Formulate an optimization program to solve for the dimensions h,w and d that results in a box of the largest possible volume, and implement in CVXPY.
Solution:
Given

Description	Parameter	Value
Width	w	?
Height	h	?
Depth	d	?
Volume	v	?
Wall area	2(wh+hd)	$\leq$ 200
Floor area	dw	$\leq$ 60
Aspect ratio = h/w and d/w	a_1 and a_2	$\leq$ 1.2 and $\geq$0.8

TABLE IV

Let **w** be the optimization variables vector.

$$\mathbf{w} = \begin{pmatrix} w \\ h \\ d \end{pmatrix} \tag{16}$$

Objective is to maximize the volume of the box.

$$v = whd$$

Objective function is defined as,

$$z = \max_{w,h,d}(whd) \tag{17}$$

constraints are,

$$wh \leq 200 \tag{18}$$
$$dw \leq 60 \tag{19}$$
$$0.8 \leq a_1 \leq 1.2 \tag{20}$$
$$0.8 \leq a_2 \leq 1.2 \tag{21}$$

Writing constraints in the matrix form,

$$\mathbf{w}^{\mathrm{T}}\mathbf{P_1}\mathbf{w} \leq q_1 \tag{22}$$
$$\begin{pmatrix} w & h & d \end{pmatrix}\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}\begin{pmatrix} w \\ h \\ d \end{pmatrix} \leq 100 \tag{23}$$
$$\mathbf{w}^{\mathrm{T}}\mathbf{P_2}\mathbf{w} \leq q_2 \tag{24}$$
$$\begin{pmatrix} w & h & d \end{pmatrix}\begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}\begin{pmatrix} w \\ h \\ d \end{pmatrix} \leq 60 \tag{25}$$
$$\mathbf{P_3}\mathbf{w} \leq \mathbf{q_3} \tag{26}$$
$$\begin{pmatrix} -0.8 & -1 & 0 \\ 1 & -1.2 & 0 \\ -0.8 & 0 & -1 \\ 1 & 0 & -1.2 \end{pmatrix}\begin{pmatrix} w \\ h \\ d \end{pmatrix} \leq \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \tag{27}$$

Solving the objective function and constrains by cvxpy.
cvxpy code,

https://github.com/gadepall/EE5606−optimization/codes/opt_3.py

The maximum optimum value is 387.29 cubic units.
Optimum w = 7.74, h = 6.45, d = 7.74 units.

4) A fictional company AccessApple & Co. produces three types of covers for Apple products: one for iPod, one for iPad, and another for iPhone. The company's production facilities are such that if we devote the entire production to iPod covers, we can produce 6000 of them in a day. If we devote the entire production to iPhone covers or iPad covers, we can produce 5000 or 3000 of them, respectively, in one day. The production schedule is one work week of 5 working days, and the week's production must be stored before distribution. Storing 1000 iPod covers (packaging included) takes up 40 cubic feet of space. Storing 1000 iPhone covers (packaging included) takes up 45 cubic feet of space, and storing 1000 iPad covers (packaging included) takes up 210 cubic feet of space. The total storage space available is 6000 cubic feet. Due to a commercial agreement, AccessApple & Co. has to deliver at least 5000 iPod covers, and 4000 iPad covers per week in order to strengthen the products' diffusion. The marketing department estimates that the weekly demand for iPod covers, iPhone, and iPad covers does not exceed 10000 and 15000 , and 8000 units, respectively. Therefore the company does not want to produce more than these numbers. Finally, the net profits in USD for each iPod cover, iPhone cover, and iPad cover are USD 4, USD 6 and USD 10 respectively. The aim is to determine a weekly production schedule that maximizes the total net profit.

Solution:
Consider,
x= proportion of time devoted each day to iPod cover production,
y= proportion of time devoted each day to iPhone cover production,
z= proportion of time devoted each day to iPad cover production.

Description	Parameter	Value
Proportion of time devoted	$\mathbf{x}$	$\begin{pmatrix} x \\ y \\ z \end{pmatrix}$
Production per day	$\mathbf{p}_p$	$\begin{pmatrix} 6,000 \\ 5,000 \\ 3,000 \end{pmatrix}$
Storage per 1000 units	$\mathbf{s}$	$\begin{pmatrix} 40 \\ 45 \\ 210 \end{pmatrix}$
minium requirement	$\mathbf{m}_r$	$\begin{pmatrix} 5,000 \\ 0 \\ 4,000 \end{pmatrix}$
Maximum Production	$\mathbf{m}_p$	$\begin{pmatrix} 10,000 \\ 15,000 \\ 8,000 \end{pmatrix}$
Profit	$\mathbf{p}$	$\begin{pmatrix} 4 \\ 6 \\ 10 \end{pmatrix}$
profit for week	$\mathbf{c}$	$\begin{pmatrix} 120000 \\ 15000 \\ 15000 \end{pmatrix}$

TABLE V

The elements of $\mathbf{c}$ are the product of production per day, profit and no.of workings day in a week for ipod, iphone and ipad respectively.
Objective function,

$$z = \max_{\mathbf{x}} \mathbf{c}^T\mathbf{x} \tag{28}$$

$$z = \min_{\mathbf{x}} (-\mathbf{c}^T\mathbf{x}) \tag{29}$$

constrains,

$$x + y + z \leq 1 \tag{30}$$

$$1200x + 1125y + 3150 \leq 6000 \tag{31}$$

$$30,0000x \geq 5000 \tag{32}$$

$$15,000z \geq 4000 \tag{33}$$

$$30,000x \leq 10,000 \tag{34}$$

$$25,000y \leq 15,000 \tag{35}$$

$$15,000z \leq 8,000 \tag{36}$$

$$\mathbf{x} \geq 0 \tag{37}$$

Putting all the constrains in the form$\mathbf{Px} = \mathbf{q}$,

$$\begin{pmatrix} 1 & 1 & 1 \\ 1200 & 1125 & 3150 \\ -30,000 & 0 & 0 \\ 0 & 0 & -15,000 \\ 30,000 & 0 & 0 \\ 0 & 25,000 & 0 \\ 0 & 0 & 15,000 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \mathbf{x} \leq \begin{pmatrix} 1 \\ 6,000 \\ -5,000 \\ -4,000 \\ 10,000 \\ 15,000 \\ 8,000 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{38}$$

Solving the objective function and constrains by cvxpy.
cvxpy code,

https://github.com/gadepall/EE5606−optimization/codes/opt_4.py

The maximum weekly profit= 1,45,000 USD.
Optimum time devoted, $\mathbf{x} = \begin{pmatrix} 0.1667 \\ 0.4129 \\ 0.4203 \end{pmatrix}$.

5) Decomposing a PV array output time series: We are given a time series $\mathbf{p} \in \mathfrak{R}_+^T$ that gives the output power of photo-voltaic array in 5 minutes intervals, over T = 2016 periods, given in pv output data.*. In this problem you will use convex optimization to decompose the time series into three components:
   - The clear sky output $\mathbf{c} \in \mathfrak{R}_+^T$, a smmoth daily-periodic component, which gives what the PV output would have been without clouds. This signal is 24-hour-periodic, i,e., $c_{t+288} = c_t$ for t = 1,2, ... T-288. (The clear sky output is zero at night, but we will not use this prior information in our decomposition method).
   - A weather shading loss component $\mathbf{s} \in \mathfrak{R}_+^T$, which gives the loss of power due to clouds. This component satisfies $0 \geq \mathbf{s} \geq \mathbf{c}$, can change rapidly, and is not periodic.
   - A residual $\mathbf{s} \in \mathfrak{R}_+^T$, which accounts for measurements error, anomalies, and other errors.

   These components sastisfy $\mathbf{p} = \mathbf{c} - \mathbf{s} + \mathbf{r}$
   We will assume that the average absolute value of the residual is no more than 4.

Smoothness of $\mathbf{c}$ is measured by its Laplacian,

$$\mathcal{L}(c) = (c_1 - c_2)^2 + .. + (c_{287} - c_{288})^2 + (c_{288} - c_1)^2$$

We will choose $\mathbf{c}, \mathbf{s}$ and $\mathbf{r}$ by minimizing $\mathcal{L}(c)+\lambda 1^T\mathbf{s}$ subject to the constraints described above, where λ is a positive parameter, that we take to be one.

Solve this problem, and plot the resulting $\mathbf{c}, \mathbf{s}, \mathbf{r}$ and $\mathbf{p}$ on separate plots. Give the average values of c, s and p and the average value of r (which should be 4).

Solution:

Consider,

Description	Parameter	Value
PV array output	**p**	pv output data.py
Clear sky output	**c**	?
Weather shading loss component	**s**	?
residual	**r**	?
Total Period	T	2016
Samples	N	288

TABLE VI

Objective function,

$$z = \min_{\mathbf{s},\mathbf{c}} \mathcal{L}(c) + \lambda 1^T\mathbf{s} \tag{39}$$

Writing the objective function in terms of matrices,

Let $\mathbf{c_1}$ is a vector of size 288 X 1, first 288 elements of vector$\mathbf{c}$

$$(c_1 - c_2)^2 + .. + (c_{287} - c_{288})^2 + (c_{288} - c_1)^2 = [\mathbf{c_1} - \mathbf{Ac_1}]^T[\mathbf{c_1} - \mathbf{Ac_1}] \tag{40}$$

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & . & . & . & 0 \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ 0 & 0 & 0 & . & . & . & 1 \\ 1 & 0 & 0 & . & . & . & 0 \end{pmatrix} \tag{41}$$

Objective function can be modified as,

$$z = \min_{\mathbf{s},\mathbf{c}} [\mathbf{c_1} - \mathbf{Ac_1}]^T[\mathbf{c_1} - \mathbf{Ac_1}] + \lambda 1^T\mathbf{s} \tag{42}$$

Constraints are listed below,

$$0 \preceq \mathbf{s} \preceq \mathbf{c} \tag{43}$$

$$\mathbf{p} = \mathbf{c} - \mathbf{s} + \mathbf{r} \tag{44}$$

$$c_{t+288} = c_t \tag{45}$$

$$(\|\mathbf{r}\|_1)/\mathrm{T} \tag{46}$$

Solving the objective and constraints by CVXPY.

CVXPY code,

https://github.com/gadepall/EE5606–optimization/codes/opt_5.py

From cvxpy,

$$z = 98131.26 \tag{47}$$

$$\text{avg of c} = 529.51 \tag{48}$$

$$\text{avg of s} = 4.47 \tag{49}$$

$$\text{avg of p} = 529.03 \tag{50}$$

$$\text{absolute avg of r} = 3.99 \tag{51}$$

We get the plots for $\mathbf{c}, \mathbf{s}$ and $\mathbf{r}$.

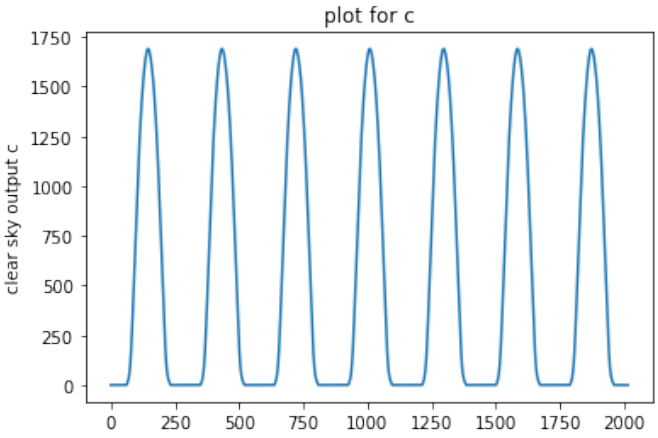

Fig. 0: Clear sky plot

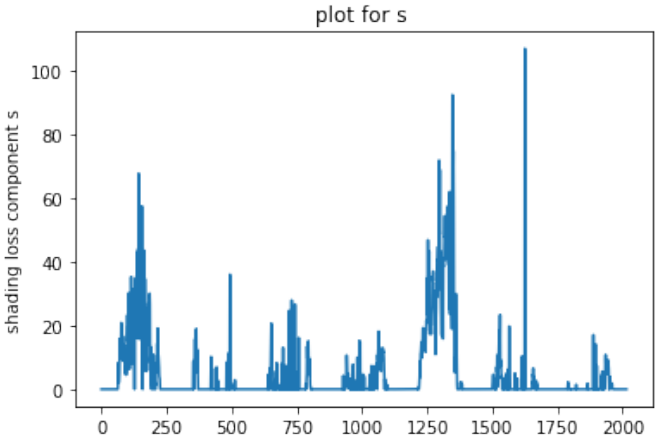

Fig. 0: shading loss plot

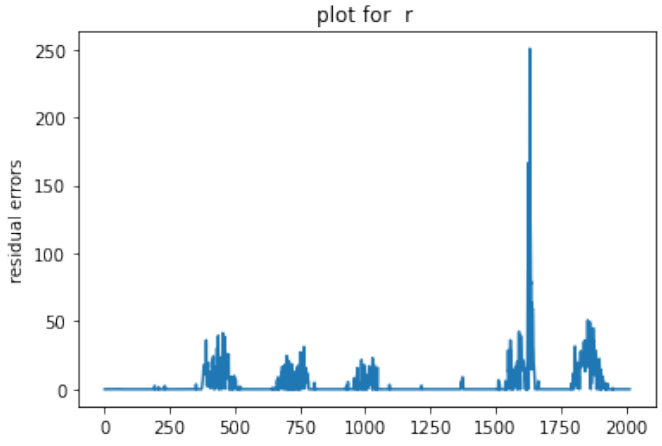

Fig. 0: residual plot

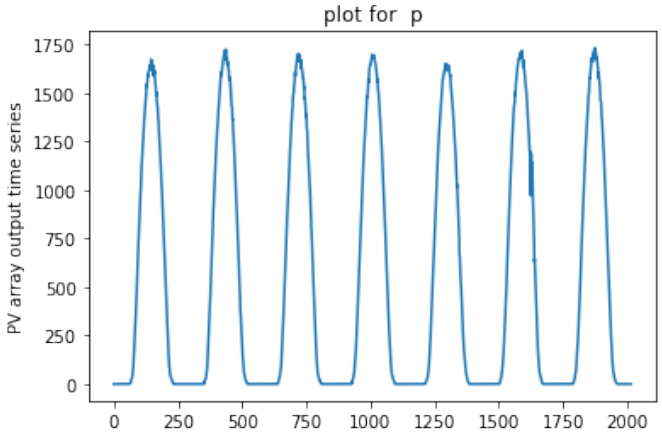

Fig. 0: PV array power plot

6) A company ' XYZ ' is constructing a road through a hilly area. The height of the roadbed should be chosen in such a way that the total cost of construction is minimized. The construction cost depends on the difference between height (h) of the roadbed and the current elevation of the road (e). h_i gives the height of the roadbed at a distance d_i meters down the road, where $d > 0$ is a given discretization. The existing elevation at a point d_i meters down the road is given by e_i.

$$\mathbf{e} = 5\sin\frac{i}{3n\pi} + \sin\frac{i}{10n\pi}$$

where i ranges from 1 to n, where n = 100.
The construction cost is mainly affected by the cuts(roadbed below existing elevation) and fills(roadbed above existing elevation) present in the road. The cut cost ϕ^{cut} and fill cost ϕ^{fill} are the linear functions of the difference between the existing elevation of the road and height of the roadbed. The overall cost (C) is a linear combination of the cut cost and fill cost.

$$\mathbf{u} = \mathbf{h} - \mathbf{e}$$
$$(u)_+ = \max\{u,0\}$$
$$(u)_- = \max\{-u,0\}$$
$$\phi^{\text{fill}}(u) = 2(u)_+^2 + 30(u)_+$$
$$\phi^{\text{cut}}(u) = 12(u)_-^2 + (u)_-$$
$$c = \phi^{\text{cut}} + \phi^{\text{fill}}$$

The goal is to minimize c subject to the following constraints.

- The maximum allowable road slope (first derivative $D^{(1)}$) is 0.08.
- The maximum allowable curvature (second derivative $D^{(2)}$) is 0.025
- The maximum allowable third derivative $D^{(3)}$ is 0.005

Formulate the optimization problem and verify the convexity of cut and fill functions by plotting for u in range (1 : 0.1 : 10). Plot **h**, **e** and **u** for the optimal grading plan and report the associated cost.

Solution:
Consider,

Description	Parameter	Value
Height of the road	**h**	?
Current elevation of the road	**e**	$5\sin\frac{i}{3n\pi} + \sin\frac{i}{10n\pi}$
Discretization unit	d	1
Difference	**u**	**h** – **e**
Filling height	$(u)_+$	max{u,0}
Cutting height	$(u)_-$	max{-u,0}
size of vectors	n	100

TABLE VII

$$\mathbf{u} = \mathbf{h} - \mathbf{e}$$
$$(u)_+ = \max\{u,0\}$$
$$(u)_- = \max\{-u,0\}$$
$$\phi^{\text{fill}}(u) = 2(u)_+^2 + 30(u)_+$$
$$\phi^{\text{cut}}(u) = 12(u)_-^2 + (u)_-$$
$$c = \phi^{\text{cut}} + \phi^{\text{fill}}$$

Objective function,

$$z = \min_{\mathbf{h}} c \tag{52}$$

Constraints,

$$\frac{dh}{dn} \leq 0.08 \tag{53}$$

$$\frac{d^2h}{dn^2} \leq 0.025 \tag{54}$$

$$\frac{d^3h}{dn^3} \leq 0.005 \tag{55}$$

CVXPY code,

https://github.com/gadepall/EE5606–optimization/codes/opt_6.py

From CVXPY, the plots are as follows

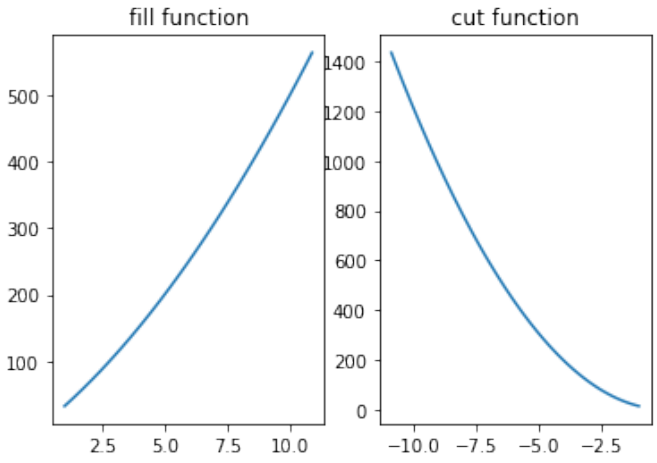

Fig. 0: Fill and Cost plot

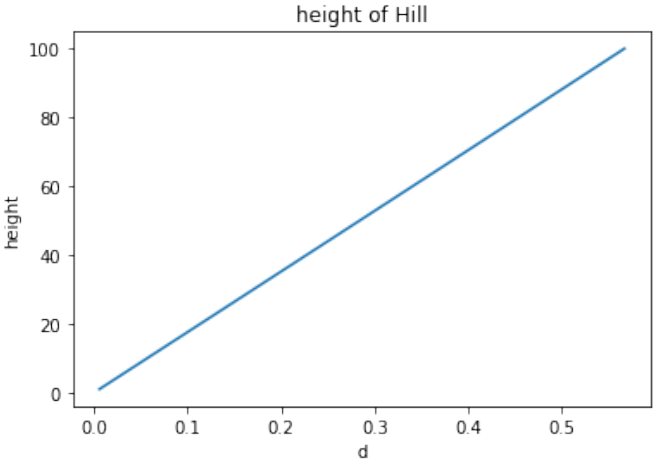

Fig. 0: Height plot

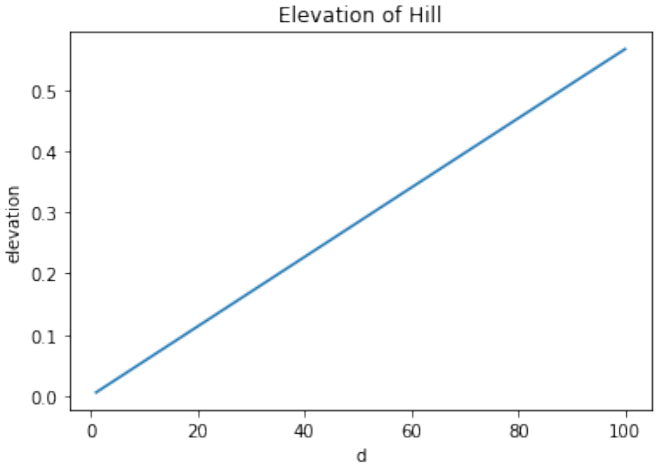

Fig. 0: Elevation plot

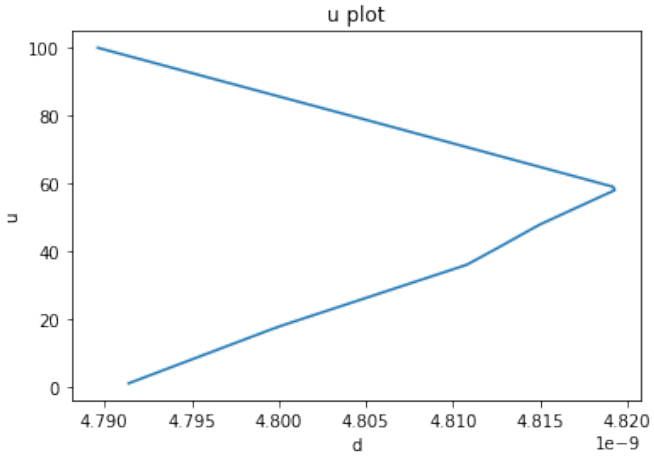

Fig. 0: U plot

7) Power Assignment in a Wireless Communication System: Consider a system with n transmitters, each with power $p_j \geq 0$, transmitting to n receivers. Let $G_{ij} \geq 0$ denote the path gain from transmitter j to receiver i. These path gains form the matrix $\mathbf{G} \in \Re^{nxn}$.Each receiver is assigned to a transmitter such that the signal power at receiver i, $S_i = G_{ii}p_i$ and the interefence power at receiver i is $I_i = \Sigma_{k\neq i}G_{ik}p_k$. Given a noise power σ_i at each receiver, the SINR at receiver i, $\gamma_i = \frac{S_i}{I_i + \sigma_i}$. The objective is to maximise the minimum SINR of the system under certain power constraints. These constraints are:

a) Each transmitter power$p_j \leq P_j^{max}$.
b) If the transmitters are partitioned into m non-overlapping groups, $k_1, k_2....k_m$, which share

a common power supply with total power $P_l^{gp}, \Sigma_{k\in k_i} p_k \leq P_l^{gp}$.

c) There is a maximum power that each receiver can receive $P_i^{rc}, \Sigma_{k=1}^{n} G_{ik} p_k \leq P_i^{rc}$

Formulate SINR as linear-fractional programming.

SINR maximization: We haven 5 transmitters, grouped into two groups : {1,2} and {3,4,5}. The maximum power for each transmitter is 3, the total power limit for the first group is 4, and the total power limit for the second group is 6. The noise σ is equal to 0.5 and the limit on total received power is 5 for each receiver. Finally, the path gain matrix is given by,

$$\mathbf{G} = \begin{pmatrix} 1.0 & 0.1 & 0.2 & 0.1 & 0.0 \\ 0.1 & 1.0 & 0.1 & 0.1 & 0.0 \\ 0.2 & 0.1 & 2.0 & 0.2 & 0.2 \\ 0.1 & 0.1 & 0.2 & 1.0 & 0.1 \\ 0.0 & 0.0 & 0.2 & 0.1 & 1.0 \end{pmatrix}$$

Find the transmitter powers $p_1, ..., p_5$ that maximize the minimum SINR ratio overall receivers. Also report the maximum SINR value. Solving the problem to an accuracy of 0.05 (in SINR) is fine.

Solution:

consider,

Description	Parameter	Value
Powers of transmitter	**p**	?
Gain matrix	**G**	$\begin{pmatrix} 1.0 & 0.1 & 0.2 & 0.1 & 0.0 \\ 0.1 & 1.0 & 0.1 & 0.1 & 0.0 \\ 0.2 & 0.1 & 2.0 & 0.2 & 0.2 \\ 0.1 & 0.1 & 0.2 & 1.0 & 0.1 \\ 0.0 & 0.0 & 0.2 & 0.1 & 1.0 \end{pmatrix}$
Powers of receiver	**r**	**Gp**
noise	σ	$\begin{pmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \end{pmatrix}$
no.of transmitters and receivers	n	5

TABLE VIII

However, since this is a quasiconvex objective function we cannot solve it directly using CVXPY. Instead we must use a bisection method. First we take the step of rewriting the objective, $\alpha = \gamma^{-1}$

Objective function,

$$z = \min_{\mathbf{p}} \alpha \tag{56}$$

Constraints,

$$\alpha \geq 0 \tag{57}$$

$$\mathbf{p} \geq 0 \tag{58}$$

$$\mathbf{p} \leq p^{max} \tag{59}$$

$$\mathbf{Gp} \leq \mathbf{p}^{rc} \tag{60}$$

Solving by CVXPY,
CVXPY code,

https://github.com/gadepall/EE5606−optimization/codes/opt_7.py

$$z = 0.7598$$

$$\mathbf{p} = \begin{pmatrix} 0.51 \\ 0.49 \\ 0.32 \\ 0.54 \\ 0.47 \end{pmatrix}$$

8) Minimum time maneuver for a crane: Minimum time maneuver for a crane. A crane manipulates a load with mass $m > 0$ in two dimensions using two cables attached to the load. The cables maintain angles $\pm\theta$ with respect to vertical, as shown below,

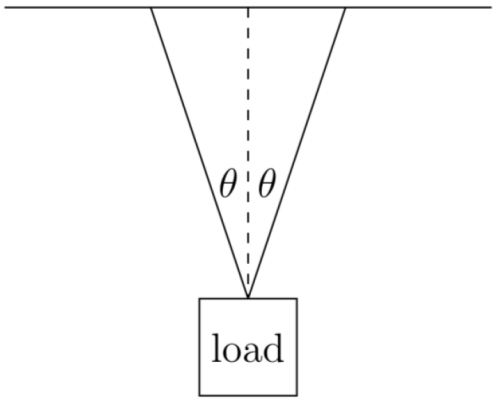

Fig. 0: Load

The (scalar) tensions T^{left} and T^{right} in the two cables are independently controllable, from 0 up to a given maximum tension T^{max}. The total force on the load is

$$F = T^{left} \begin{pmatrix} -\sin\theta \\ \cos\theta \end{pmatrix} + T^{right} \begin{pmatrix} \sin\theta \\ \cos\theta \end{pmatrix} + mg \tag{61}$$

where g=9.8 m/sec^2, accelaration due to gravity.
The acceleration of the load is F/m. We approximate the motion of the load using

$$p_{i+1} = p_i + hv_i$$
$$v_{i+1} = v_i + (h/m)F_i, i = 1, 2, ...,$$

where $p_i \in \mathfrak{R}^2$ is the position of the load, $v_i \in \mathfrak{R}^2$ is the velocity of the load, and $F_i \in \mathfrak{R}^2$ is the force on the load, at time t = ih. Here $h > 0$ is a small (given) time step. The goal is to move the load, which is initially at rest at position p_{init} to the position p^{des}, also at rest, in minimum time. In other words, we seek the smallest k for which

$$p_1 = p^{init}, p_k = p^{des}, v_1 = v_k = (0, 0)$$

is possible, subject to the constraints described above.

a) Explain how to solve this problem using convex (or quasiconvex) optimization.
b) Carry out the method of part (a) for the problem instance with

$$m = 0.1, \theta = 15^\circ, Tmax = 2, p^{init} = (0, 0),$$
$$p^{des} = (10, 2), \text{ with time step } h = 0.1.$$

Report the minimum time k. Plot the tensions versus time, and the load trajectory, i.e., the points $p_1, ..., p_k$ in $\mathfrak{R}^2$. Does the load move along the line segment between p^{init} and p^{des} (i.e., the shortest path from p^{init} and p^{des})? Comment briefly.

Solution:
CVXPY code,

https://github.com/gadepall/EE5606–optimization/codes/opt_8.py

9) Suppose you manage n factories, each producing s_i amount of goods for i = 1, 2, . . . , n. These goods need to be shipped to m destinations, each having a demand d_i for i = 1, 2, . . . , m, and our goal is to move the goods from the factories to the destinations so as to satisfy the demand. However, there is a cost associating to moving the goods: for moving a unit good from factory i to destination j, the cost involved is c_{ij} . Our goal is to decide the amount of quantities to be shipped from each factory to each destination such that the demand is satisfied, and the overall cost is minimized. We assume that the goods are divisible (so the quantity shipped does not have to be an integer), and that the shipping is directly to the respective destinations (meaning there is no routing involved).

a) Formulate the problem above as an LP, and write a CVX/CVXPY code to compute the quantities shipped.
b) Find the optimal transportation strategy for the following costs, supply and demands:

Factory	D_1	D_2	D_3	D_4	D_5	**Total Supply**
Factory 1	8	6	19	9	8	40
Factory 2	9	12	13	7	5	50
Factory 3	14	9	16	5	2	45
Total demand	45	20	30	30	10	

TABLE IX

Solution:
consider,

Description	**Parameter**	**Value**
No. of factories	n	3
No. of destinations	m	5
Supply	**s**	$\begin{pmatrix}40\\50\\45\end{pmatrix}$
demand	**d**	$\begin{pmatrix}45\\20\\30\\30\\10\end{pmatrix}$
cost	**c**	$\begin{pmatrix}8&6&10&9&8\\9&12&13&7&5\\14&9&16&5&2\end{pmatrix}$
Amount of quantities to be shipped from each factory to each destination	**X**	?

TABLE X

Objective function,

$$z = \min_{\mathbf{X}} \text{trace}(\mathbf{c}^T\mathbf{X}) \tag{62}$$

Constraints,

$$\mathbf{X1} \preceq \mathbf{s} \tag{63}$$
$$\mathbf{X}^T\mathbf{1} \succeq \mathbf{d} \tag{64}$$
$$\mathbf{X} \succeq \mathbf{0} \tag{65}$$

solving by cvxpy
cvxpy code,

https://github.com/gadepall/EE5606–optimization/codes/opt_9.py

Optimum value of z is 1025.

10) Consider the polynomial f(x) = x^n. We wish to approximate this polynomial with a degree n-1 polynomial $a_0 + a_1x + a_2x^2 + ... + a_{n-1}x^{n-1}$ on the

interval [-1,1]. Consider partitioning the interval [-1,1] into 2N+1 equispaced points $-1 = x_{-N}, x_{-N+1}, ..., x_{-1}, x_0 = 0, x_1, ..., x_{N-1}, x_N = 1$. Suppose we wish to pick the coefficients a_i such that the following cost function is minimized:

$$\Sigma_{k=-N}^{N}(x_k^n - \Sigma_{i=0}^{n-1} a_i x_k^i)^2$$
$$\Sigma_{k=-N}^{N}|x_k^n - \Sigma_{i=0}^{n-1} a_i x_k^i|$$

a) Formulate the problem of finding the coefficients a_i, and write a CVX/CVXPY code to compute the coefficients.
b) Find the coefficients for n = 5 n = 10, n = 20. Pick N to be large enough.

Solution:
consider,

$$\mathbf{a} = \begin{pmatrix} a_0 \\ . \\ . \\ a_{n-1} \end{pmatrix} \tag{66}$$

Objective function,

$$z_1 = \min_{\mathbf{a}} \Sigma_{k=-N}^{N}(x_k^n - \Sigma_{i=0}^{n-1} a_i x_k^i)^2 \tag{67}$$
$$z_2 = \min_{\mathbf{a}} \Sigma_{k=-N}^{N}|x_k^n - \Sigma_{i=0}^{n-1} a_i x_k^i| \tag{68}$$

Solving by using cvxpy. cvxpy code,

https://github.com/gadepall/EE5606–optimization/codes/opt_10.py

11) Consider following program

$$\begin{aligned} \min_{x_1,x_2} \; & x_1^2 + 2x_2^2 - x_1x_2 - x1 \\ \text{s.t.} \; & x_1 - 2x_2 \le -2 \\ & x_1 + 4x_2 \le -3 \\ & 5x_1 - 76x_2 \le 1 \end{aligned} \tag{69}$$

Solve the above problem using CVXPY
Solution:
let,

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \tag{70}$$

Objective function,

$$z = \min_{x_1,x_2} x_1^2 + 2x_2^2 - x_1x_2 - x1 \tag{71}$$

$$\tag{72}$$

Objective function can be equivalently written as,

$$z = \min_{\mathbf{x}} \mathbf{x^T A x} + \mathbf{b^T x} \tag{73}$$

$$\mathbf{A} = \begin{pmatrix} 1 & -1/2 \\ -1/2 & 2 \end{pmatrix} \tag{74}$$

$$\mathbf{b} = \begin{pmatrix} -1 \\ 0 \end{pmatrix} \tag{75}$$

constraints,

$$x_1 - 2x_2 \le -2 \tag{76}$$
$$x_1 + 4x_2 \le -3 \tag{77}$$
$$5x_1 - 76x_2 \le 1 \tag{78}$$

Equivalent way of writting constraints,

$$\mathbf{Px} \preceq \mathbf{q} \tag{79}$$

$$\mathbf{P} = \begin{pmatrix} 1 & -2 \\ 1 & 4 \\ 5 & -76 \end{pmatrix} \tag{80}$$

$$\mathbf{q} = \begin{pmatrix} -2 \\ -3 \\ 1 \end{pmatrix} \tag{81}$$

Solving by cvxpy.
cvxpy code,

https://github.com/gadepall/EE5606–optimization/codes/opt_11.py

Optimum value of z = 7.44