Домашнее задание II

Кобылянский А.В.
Группа 5381
Вариант 2

Задача 1

Пользуясь определением выпуклой функции

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y)$$

Где $\lambda \in [0,1]$, показать, что следующие функции выпуклые

$$f(x) = \|x\|_1, x \in \mathbb{R}^n, (l_1 - norm)$$
$$f(x) = \|x\|_2, x \in \mathbb{R}^n, (l_2 - norm)$$
$$f(x) = \|x\|_\infty, x \in \mathbb{R}^n, (infinity - norm)$$

Если для функции $\|.\| : \mathbb{R}^n \to \mathbb{R}$ выполняются 2 аксиомы нормы

$$1. \forall \alpha \in \mathbb{R} \, \forall x \in \mathbb{R}^n : \|\alpha x\| = |\alpha| \, \|x\|$$
$$2. \forall x, y \in \mathbb{R}^n : \|x + y\| \leq \|x\| + \|y\|$$

то эта функция выпукла

$$\|\lambda x + (1-\lambda)y\| \leq \|\lambda x\| + \|(1-\lambda)y\| = |\lambda| \, \|x\| + |1-\lambda| \, \|y\| = \lambda \|x\| + (1-\lambda) \|y\|$$

Выполнение первой аксиомы очевидно для всех трех функций. Покажем, что выполняется вторая аксиома.

1. $f(x) = \|x\|_1 = \sum_{i=1}^n |x_i|$
Так как $\forall a, b \in \mathbb{R} : |a+b| \leq |a| + |b|$, то

$$\sum_{i=1}^n |x_i + y_i| \leq \sum_{i=1}^n |x_i| + |y_i| = \sum_{i=1}^n |x_i| + \sum_{i=1}^n |y_i|$$

Левая сумма поэлементно меньше правой. Аксиома выполняется.

2. $f(x) = \|x\|_2 = \sqrt{\sum_{i=1}^n |x_i|^2} = \sqrt{x^T x}$

$$\sqrt{(x+y)^T(x+y)} \leq \sqrt{x^T x} + \sqrt{y^T y}$$
$$(x+y)^T(x+y) \leq x^T x + y^T y + 2\sqrt{x^T x y^T y}$$
$$x^T x + y^T y + 2x^T y \leq x^T x + y^T y + 2\sqrt{x^T x y^T y}$$
$$x^T y \leq \sqrt{x^T x y^T y}$$

В первом неравенстве обе части неотрицательны, так что можно возвести их в квадрат, получится эквивалентное неравенство.

Последнее неравенсто - более слабая версия неравенства Коши — Буняковского.

$$\forall \lambda \in \mathbb{R} \, \forall x, y \in \mathbb{R}^n : (\lambda x + y)^T(\lambda x + y) \geq 0$$
$$\lambda^2 x^T x + 2\lambda x^T y + y^T y \geq 0$$
$$D = 4(x^T y)^2 - 4x^T x y^T y \leq 0$$
$$|x^T y| \leq \sqrt{x^T x y^T y}$$

3. $f(x) = \|x\|_\infty = \max|x_i|$

$$\max|x_i + y_i| = |x_m + y_m| \leq |x_m| + |y_m| \leq \max|x_i| + \max|y_i|$$

Задача 2

Пусть $S = \{x \in \mathbb{R}^n \mid x^T Ax + b^T x + c \leq 0\}$, где A - симметричная матрица. Показать, что множество S выпукло, если $A \succeq 0$ (матрица A является положительно полуопределенной).

Множество линий уровня выпуклой функции

$$Lev_f(\alpha) := \{x \mid f(x) \leq \alpha\}$$

является выпуклым.

Покажем, что наша функция выпукла

$$\begin{aligned}
f(x) &= x^T Ax + b^T x + c \\
df(x) &= 2x^T Adx + b^T dx = (2Ax + b)^T dx \\
d((2Ax + b)^T dx_1) &= (dx_1)^T 2Adx_2 \\
\nabla^2 f &= 2A \succeq 0
\end{aligned}$$

По дифференциальному признаку функция выпукла, значит выпукло и исходное множество.

Задача 3

Пусть $S \subset \mathbb{R}^n$ - замкнутое выпуклое множество и $y \in \mathbb{R}^n$. Введем обозначение $\Pi_s(y)$, обозначающее евклидову проекцию точки y на множество S, Можно показать, что для любой точки $x \in S$ выполняется неравенство

$$(x - \Pi_s(y))^T (y - \Pi_s(y)) \leq 0 \tag{1}$$

Доказать, что для любых $z, y \in \mathbb{R}^n$ верно

$$\|\Pi_s(z) - \Pi_s(y)\|_2 \leq \|z - y\|_2$$

Доказательство

$$\begin{aligned}
\|(\Pi_s(z) - \Pi_s(y)) - (z - y)\|^2 &= \|\Pi_s(z) - \Pi_s(y)\|^2 + \|z - y\|^2 - 2(\Pi_s(z) - \Pi_s(y))^T (z - y) \\
(\Pi_s(z) - \Pi_s(y))^T (z - y) &= \|\Pi_s(z) - \Pi_s(y)\|^2 + (\Pi_s(z) - \Pi_s(y))^T (z - \Pi_s(z)) + \\
&+ (\Pi_s(z) - \Pi_s(y))^T (\Pi_s(y) - y) \\
\|z - y\|^2 - \|\Pi_s(z) - \Pi_s(y)\|^2 &= \|(\Pi_s(z) - \Pi_s(y)) - (z - y)\|^2 + \\
&+ 2(\Pi_s(z) - \Pi_s(y))^T (z - \Pi_s(z)) + \\
&+ 2(\Pi_s(z) - \Pi_s(y))^T (\Pi_s(y) - y)
\end{aligned}$$

В последнем равенстве в правой части первое слагаемое неотрицательно, потому что это квадрат нормы, второе и третье слагаемое неотрицательны из-за неравенства (1). Значит

$$\|z-y\|^2 - \|\Pi_s(z) - \Pi_s(y)\|^2 \geq 0 \Rightarrow \|\Pi_s(z) - \Pi_s(y)\| \leq \|z-y\|$$

что и требовалось доказать.

Это значит, что отображение $\Pi_s(y) : y \mapsto \arg\min_{x \in S} \|x-y\|_2^2$ является нерастягивающим. В случае если для некоторого отображения F выполняется

$$\|F(z) - F(y)\|_2 < \|z-y\|_2$$

то оно называется сжимающим.

Показать, что композиция нерпстягивающего и сжимающего отображений является сжимающим отображением.

Пусть F - нерастягивающиее отображение, а G - сжимающиее, тогда

$$\|F(G(z)) - F(G(y))\|_2 \leq \|G(z) - G(y)\|_2 < \|z-y\|_2$$
$$\|G(F(z)) - G(F(y))\|_2 < \|F(z) - F(y)\|_2 \leq \|z-y\|_2$$

$F \circ G$ и $G \circ F$ являются сжимающими отображениями.

Задача 4

Рассмотрим задачу оптимизации

$$\min_x \|x\|$$
$$s.t. Ax = b$$

Показать, что x_* является единственным решением этой задачи только если

$$T_C(x_*) \cap null(A) = \{0\}$$

где C - произвольный шар нормы $\|.\|$, а $null(A) = \{x \mid Ax = 0\}$ обозначает ядро или нуль-пространство линейного оператора A.

Пусть $T_C(x_*) \cap null(A) \neq \{0\}$

$$\begin{aligned} z - x_* \in T_C(x_*) \cap null(A) \setminus \{0\} = \\ = \text{cone}\{z - x_* \mid \|z\| \leq \|x_*\|\} \cap \{x \mid Ax = 0\} \setminus \{0\} \end{aligned}$$

тогда

$$z - x_* \neq 0 \Rightarrow z \neq x_*$$
$$z - x_* \in \text{cone}\{z - x_* \mid \|z\| \leq \|x_*\|\} \Rightarrow \|z\| \leq \|x_*\|$$
$$z - x_* \in \{x \mid Ax = 0\} \Rightarrow Az = A(z - x_* + x_*) = A(z - x_*) + Ax_* = b$$

Значит x_* не единственный минимальный элемент, удовлетворяющий условиям $Ax = b$.

Задача 5

Рассмотрим задачи поиска евклидова расстояния между замкнутыми выпуклыми множествами. В общем виде такие задачи можно сформулировать следующим образом:

$$\min_{x,y} \|x-y\|^2$$
$$s.t.\ x \in C_1, y \in C_2$$

Пусть $\mathcal{A}$ - центрально симметричный ($a \in \mathcal{A} \Leftrightarrow -a \in \mathcal{A}$) набор векторов ("атомов"), такой, что элементы $\mathcal{A}$ есть крайние точки выпуклой оболочки $\mathcal{A}$, обозначаемой $conv(\mathcal{A})$. Определим атомарную норму для множества $\mathcal{A}$ следующим образом

$$\|x\|_{\mathcal{A}} = \inf\{t > 0 \mid x \in tconv(\mathcal{A})\}$$

Пусть $\mathcal{A} = \{(1,0),(0,1),(-1,1),(-1,0),(0,-1),(1,-1)\}$, $\Phi = [1,2]$ и $y = 10$. Найти решение задачи.

Пусть $\mathrm{dist}(t)$ - расстояние от шара нормы $\|.\|_{\mathcal{A}}$ с радиусом t до множества $\{x \mid Ax = y\}$. Будем искать

$$t_0 = \min_{\mathrm{dist}(t)=0} t$$

t_0 можно найти бинарным поиском, $\mathrm{dist}(t)$ при этом можно вычислить, решая задачу квадратичного программирования.

Вектор переменных равен $[a, b]$, где a - точка из шара, b точка из $\{x \mid Ax = y\}$. Условия на попадания точки в шар линейны:

$$\begin{cases} a_0 + a_1 & \le t \\ a_0 + a_1 & \ge -t \\ a_0 & \le t \\ a_0 & \ge -t \\ a_1 & \le t \\ a_1 & \ge -t \end{cases}$$

Целевую функцию можно записать как

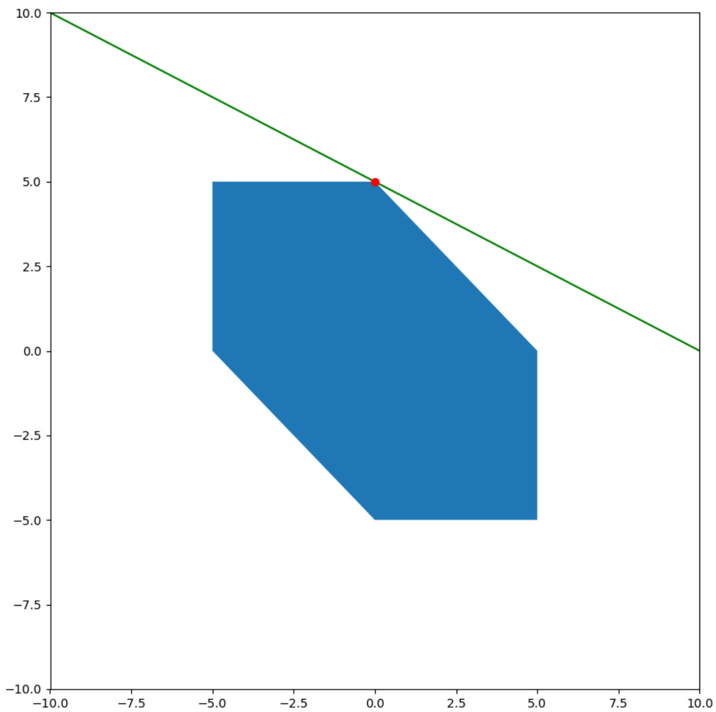
$$(a-b)^T(a-b) = \begin{bmatrix} a & b \end{bmatrix} \begin{bmatrix} I_2, & -I_2 \\ -I_2, & I_2 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}$$

Код решения:

```python
from cvxopt import solvers, matrix
import numpy as np

solvers.options['show_progress'] = False
solvers.options['feastol'] = 10**(-10)

def distance(t):
    G = np.array(
        [[1, 1, 0, 0],
        [-1, -1, 0, 0],
        [1, 0, 0, 0],
        [-1, 0, 0, 0],
        [0, 1, 0, 0],
        [0, -1, 0, 0]])

    h = 6*[t]

    A = [0, 0, 1, 2]
    b = [10]

    P = 2*np.kron([[1, -1], [-1, 1]], np.eye(2))
    q = 4*[0]

    P = matrix(P, tc='d')
    q = matrix(q, tc='d')
    G = matrix(G, tc='d')
    h = matrix(h, tc='d')
    A = matrix(A, (1, 4), tc='d')
    b = matrix(b, tc='d')

    sol = solvers.qp(P, q, G, h, A, b)

    x = np.array(sol['x']).reshape((4, 1))
    return (0.5 * x.T @ P @ x)[0][0]

def is_almost_zero(x):
    return abs(x) < 10**(-9)

# binary search

left = 0
right = 100
eps = 10**(-5)

```

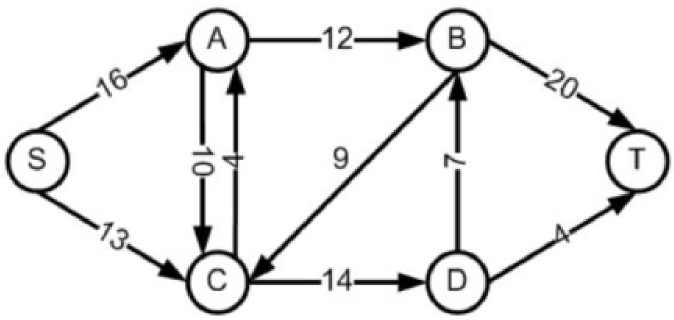
```python
while  right - left >= eps:
    mid = (right + left) / 2
    if is_almost_zero( distance(mid) ):
        right = mid
    else:
        left = mid
t = mid

print(t)  # 5.0001
```

Визуализация:



Задача 6

Найти значение максимального потока из S в T для графа на рисунке ниже. Каждое ребро имеет максимальную пропускную способность.



Задача линейного программирования:

$$\min_x c^T x$$
$$s.t. Gx \leq h$$
$$Ax = b$$

Опишем x, c, G, h, A, b.

x - вектор потоков через каждое ребро.

$$x = [SA, SC, AC, CA, AB, CD, BC, DB, BT, DT]^T$$

Нам надо максимизировать поток через ребра BT и DT или минимизировать $-(BT + DT)$.

$$c = [0, 0, 0, 0, 0, 0, 0, 0, -1, -1]^T$$

Потоки должны быть неотрицательны и меньше пропускной способности данного ребра.

$$G = \begin{bmatrix} -I \\ I \end{bmatrix}, h = \begin{bmatrix} 0 \\ h_1 \end{bmatrix}$$

Где $I \in \mathbb{R}^{10\times 10}$ - единичная матрица, $0 \in \mathbb{R}^{10}$ - столбец из нулей, h_1 - вектор пропускных способностей ребер.

$$h_1 = [16, 13, 10, 4, 12, 14, 9, 7, 20, 4]^T$$

Для узлов A, B, C, D входящий поток должен быть равен выходящиему.

$$\begin{cases} SA + CA & = AC + AB \\ AB + DB & = BC + BT \\ AC + SC + BC & = CA + CD \\ CD & = DB + DT \end{cases}$$

$$A = \begin{pmatrix} 1 & 0 & -1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 & 1 & -1 & 0 \\ 0 & 1 & 1 & -1 & 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & -1 \end{pmatrix}, b = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Код решения

```python
from cvxopt import solvers, matrix
import numpy as np

solvers.options['show_progress'] = False

capacities = [16, 13, 10, 4, 12, 14, 9, 7, 20, 4]

c = np.array([0, 0, 0, 0, 0, 0, 0, 0, -1, -1])

G = np.vstack( (-np.eye(10), np.eye(10)) )
h = 10*[0] + capacities

A = [[1, 0, -1, 1, -1, 0, 0, 0, 0, 0],
     [0, 0, 0, 0, 1, 0, -1, 1, -1, 0],
     [0, 1, 1, -1, 0, -1, 1, 0, 0, 0],
     [0, 0, 0, 0, 0, 1, 0, -1, 0, -1]]
b = [0, 0, 0, 0]

c = matrix(c, tc='d')
G = matrix(G, tc='d')
h = matrix(h, tc='d')
A = matrix(np.array(A), tc='d')
b = matrix(b, tc='d')

x = solvers.lp(c, G, h, A, b)['x']
print(x)
# [12.968, 10.032, 3.854, 2.886, 12.0, 11.0, 0.0, 7.0, 19.0, 4.0]
print(x[-2] + x[-1])  # 22.9999
```

Найденое решение действительно оптимально. В узел B не может попасть больше чем 19 единиц потока, из D максимум выходит только 4 единицы, $4 + 19 = 23$.

Задача 7

Для заданных постоянных $\Phi \in \mathbb{R}^{m\times n}, y \in \mathbb{R}^m$ и вектора переменных $x \in \mathbb{R}^n$ переформулировать следующие задачи оптимизации как задачи линейного программирования:

$$\begin{aligned} \min_x c^T x \\ s.t. Gx \leq h \\ Ax = b \end{aligned}$$

Иными словами, выразить c, G, h, A, b через Φ и y так, чтобы получились задачи оптимизации, эквивалентные следующим:

3. $\min_x \|\Phi x - y\|_1 \ s.t. \|x\|_\infty \leq \gamma$

Пусть $z = \Phi x - y$, $s = [|z_1|, |z_2|, ..., |z_m|]^T$, γ_v - вектор со значениями γ. Вектор переменных установим равным $[s; x]$. Тогда задачу оптимизации можно переписать как

$$\begin{aligned} \min 1^T s \\ s.t.\ \Phi x - y \leq s \\ \Phi x - y \geq -s \\ x \leq \gamma_v \\ -x \leq \gamma_v \end{aligned}$$

Выразми c, G, h:

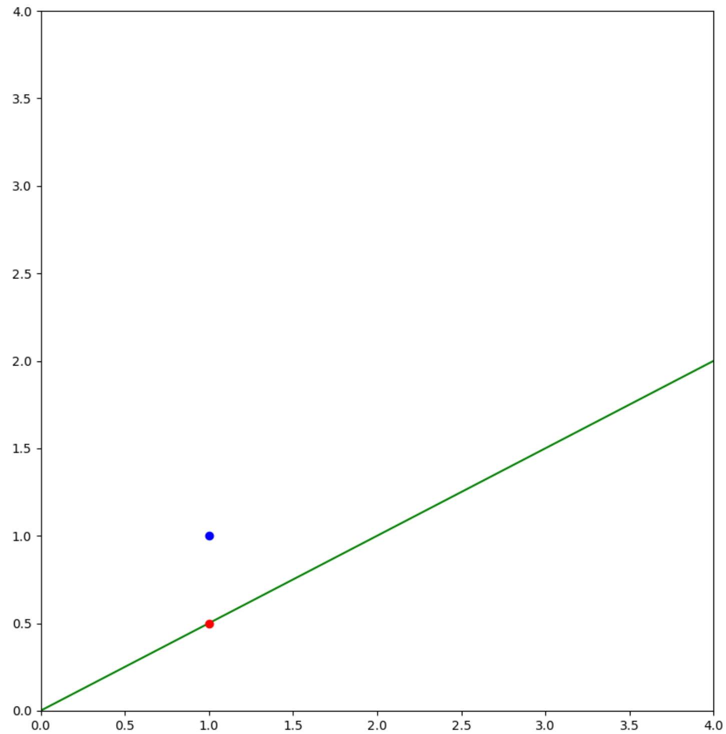
$$c = \begin{bmatrix} 1_v \\ 0_v \end{bmatrix},\ G = \begin{bmatrix} -I & \Phi \\ -I & -\Phi \\ 0_m & I \\ 0_m & -I \end{bmatrix},\ h = \begin{bmatrix} y \\ -y \\ \gamma_v \\ \gamma_v \end{bmatrix}$$

где $0_v, 1_v$ - векторы из нулей и единиц соответственно, 0_m - матрица из нулей, I - единичная матрица.

Код решения:

```
import matplotlib.pyplot as plt
from matplotlib.patches import Polygon

from cvxopt import solvers, matrix
import numpy as np

phi = np.array([2, 1]).reshape((2, 1))
y = np.array([1, 1]).reshape((2, 1))
gamma = 1

m, n = phi.shape

c = m*[1] + n*[0]

Im, In = np.eye(m), np.eye(n)
zero = np.zeros((n, m))
G1 = np.vstack((-Im, -Im, zero, zero))
G2 = np.vstack((phi, -phi, In, -In))
G = np.hstack((G1, G2))

g = gamma*np.ones((n, 1))
h = np.vstack((y, -y, g, g))

c = matrix(c, tc='d')
G = matrix(G, tc='d')
h = matrix(h, tc='d')

x = solvers.lp(c, G, h)['x'][-1]
print(x)  # 0.5

# plots

fig, ax = plt.subplots()
xs = list(range(5))
ys = [0.5*i for i in xs]
plt.plot(xs, ys, 'g')

plt.plot([2*x], [1*x], 'ro')
plt.plot([1], [1], 'bo')

ax.set_xlim(0, 4)
ax.set_ylim(0, 4)
plt.show()
```

Визуализация решения. Красная точка - Φx_*, синяя - y.



4. $\min_x \|x\|_1 \ s.t. \|\Phi x - y\|_\infty \le \epsilon$

Пусть $s = [|x_1|, |x_2|, ..., |x_n|]$, ϵ_v - вектор состоящий из ϵ, вектор переменных $[s; x]$. Тогда задачу оптимизации можно переписать как

$$\begin{aligned} \min 1^T s \\ s.t.\ x \le s \\ x \ge -s \\ \Phi x - y \le \epsilon_v \\ -\Phi x + y \le \epsilon_v \end{aligned}$$

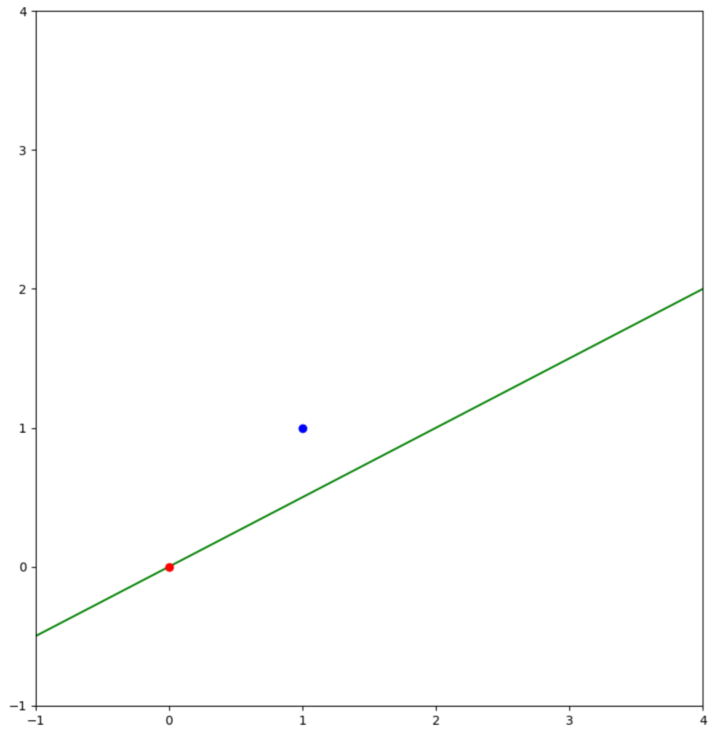
Выразми c, G, h:

$$c = \begin{bmatrix} 1_v \\ 0_v \end{bmatrix},\ G = \begin{bmatrix} -I & I \\ -I & -I \\ 0_m & \Phi \\ 0_m & -\Phi \end{bmatrix},\ h = \begin{bmatrix} 0_v \\ 0_v \\ \epsilon_v + y \\ \epsilon_v - y \end{bmatrix}$$

Код решения:

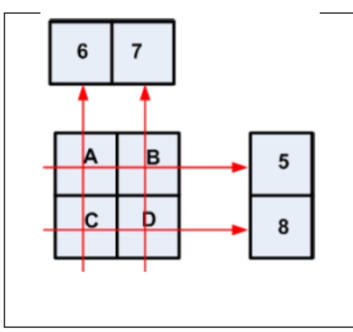
```
import matplotlib.pyplot as plt
from matplotlib.patches import Polygon

from cvxopt import solvers, matrix
import numpy as np

phi = np.array([2, 1]).reshape((2, 1))
y = np.array([1, 1]).reshape((2, 1))
epsilon = 1

m, n = phi.shape

c = n*[1] + n*[0]

I = np.eye(n)
zero = np.zeros((m, n))
G1 = np.vstack((-I, -I, zero, zero))
G2 = np.vstack((I, -I, phi, -phi))
G = np.hstack((G1, G2))

e = epsilon*np.ones((m, 1))
zero = np.zeros((1, n))
h = np.vstack((zero, zero, e + y, e - y))

c = matrix(c, tc='d')
G = matrix(G, tc='d')
h = matrix(h, tc='d')

x = solvers.lp(c, G, h)['x'][-1]
print(x)  # 4.637e-09

# plots

fig, ax = plt.subplots()
xs = list(range(-1, 5))
ys = [0.5*i for i in xs]
plt.plot(xs, ys, 'g')

plt.plot([2*x], [1*x], 'ro')
plt.plot([1], [1], 'bo')

ax.set_xlim(-1, 4)
ax.set_ylim(-1, 4)
plt.show()
```

Визуализация решения. Красная точка - Φx_*, синяя - y.



Задача 8

Найти $Z = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$, или $z = [A, B, C, D]^T = \text{vec}(Z^T)$, если известны значения вертикальной и горизонтальной проекций (суммы элементов столбцов и строк, соответственно).



$$\begin{bmatrix} 5 \\ 8 \\ 6 \\ 7 \end{bmatrix} = \begin{matrix} A+B \\ C+D \\ A+C \\ B+D \end{matrix}$$

или

$$\begin{bmatrix} 5 \\ 8 \\ 6 \\ 7 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix}$$

или

$$\boldsymbol{y} = \boldsymbol{\Phi z}$$

Существует бесконечне число векторов z, которые являются решениями данной системы уравнений. Пусть также известно, что z имеет разреженное представление в базисе Ψ:

$$\Psi = \begin{bmatrix} 1 & 3 & 0 & 5 \\ 2 & 0 & 2 & 1 \\ 1 & 1 & 4 & 0 \\ 0 & 2 & 3 & 2 \end{bmatrix},$$

Найти Z решая задачу оптимизации:

$$\min_x \|x\|_p \, s.t. \Phi\Psi x = y$$

для различных значений p.

Матрица Φ имеет ранг равный 3, и 4 строки. Уберем из неё последнюю строку строку, являющуюся линейной комбинацей верхних трех, и не на что не влеющую.

Инициализация:

```
from cvxopt import solvers, matrix
import numpy as np
from itertools import combinations

solvers.options['show_progress'] = False

phi = np.array([[1, 1, 0, 0],
                [0, 0, 1, 1],
                [1, 0, 1, 0]])

psi = np.array([[1, 3, 0, 5],
                [2, 0, 2, 1],
                [1, 1, 4, 0],
                [0, 2, 3, 2]])

y = np.array([5, 8, 6])

A = phi @ psi
m = np.linalg.matrix_rank(phi)
n = 4
```

1. $p = 0$. Полным перебором для $k = 1, 2, ..., m$

Будем перебирать комбинации столбцов, которые мы оставим и затем пробовать решить полученную систему методом наименьших квадратов. Если ошибка полученного решения мала ($< 10^{-10}$), то решение найдено. Добавим его в массив решений и запомним, что для данного k (количества ненулевых столбцов) решение найдено и большие k рассматривать нет смысла.

```
xs = []
is_solution_found = False
for k in range(1, m + 1):
    for c in combinations(range(m), k):
        a = A[:, c]
        solution, residuals, *_ = np.linalg.lstsq(a, y)

        if residuals[0] < 10**(-10):
            x = n*[0]
            for counter, index in enumerate(c):
                x[index] = solution[counter]
            xs.append(x)

            is_solution_found = True

    if is_solution_found:
        break

for i, x in enumerate(xs):
    print("x{} = {}".format(i, x))
    # x0 = [1.0, 0, 1.0000000000000002, 0]
    print("z{} = {}".format(i, psi @ x))
    # z0 = [ 1.  4.  5.  3.]
```

Найдено единственное решение для $k = 2$

$$x = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}, z = \begin{bmatrix} 1 \\ 4 \\ 5 \\ 3 \end{bmatrix}$$

2. $p = 1$. Переформулировать как задачу линейного программирования и использовать функцию cvxopt.solvers.lp для получения решения.

Задача оптимизации

$$\min_x \|x\|_1$$
$$s.t.\ Ax = y$$

Пусть $s = [|x_1|, |x_2|, ..., |x_n|]^T$, вектор переменных $[s; x]$, условия:

$$x \leq s$$
$$x \geq -s$$
$$Ax = y$$

тогда решение методом линейного программирования выглядит следующим образом.

```
c = n*[1] + n*[0]

G = np.kron([[-1, 1], [-1, -1]], np.eye(n))
h = 2*n*[0]

A_ = np.hstack((np.zeros((n - 1, n)), A))
b = y

c = matrix(c, tc='d')
G = matrix(G, tc='d')
h = matrix(h, tc='d')
A_ = matrix(A_, tc='d')
b = matrix(b, tc='d')
sol = np.array(solvers.lp(c, G, h, A_, b)['x'])

x = sol[n:]
print("x = {}".format(x))
# x = [ 9.999e-01, -1.016e-09, 1.000e+00, 1.1773e-09]
print("z = {}".format(psi @ x))
# z = [1., 4., 5., 3.]
```

Получен результат аналогичный результату в пункте 1.

3. $p = 2$. Перейти к задаче оптимизации без ограничений используя представление $x = x_0 + Uw$, где x_0 - произвольное решение, а столбцы матрицы U образуют базис $null(A)$.

Решая недоопределенную систему $Ax = y$ получаем

$$x = Uw + x_0,$$
$$\text{где } U = \frac{1}{29}\begin{bmatrix} 29 \\ 19 \\ -6 \\ -22 \end{bmatrix}, x_0 = \frac{1}{29}\begin{bmatrix} 0 \\ -19 \\ 35 \\ 22 \end{bmatrix}$$

Решим задачу оптимизации:

$$\min_w \|Uw + x_0\|_2^2$$

$$d\,\|Uw + x_0\|_2^2 = d(Uw + x_0)^T(Uw + x_0) = 2(Uw + x_0)^T U dw$$
$$U^T(Uw + x_0) = 0$$
$$w = -\frac{U^T x_0}{U^T U}$$
$$x = U\left(-\frac{U^T x_0}{U^T U}\right) + x_0 = \left(I - \frac{UU^T}{U^T U}\right)x_0$$

Функция выпуклая, значит найденая стационарная точка будет глобальным минимумом.

```
u = np.array([29, 19, -6, -22]).reshape((n, 1)) / 29
x0 = np.array([0, -19, 35, 22]).reshape((n, 1)) / 29

x = (np.eye(n) - u @ u.T / (u.T @ u)) @ x0

print("x = {}".format(x))
# x = [ 0.612, -0.253, 1.0801, 0.293]
print("z = {}".format(psi @ x))
# z = [1.321, 3.679, 4.679, 3.321]
```

Решение отличается от первых двух, но похоже на них.