Лекция 1

1. Лекция 1: Вводное занятие

1.1. Немного линейной алгебры: системы уравнений

Предположим, у нас есть два связанных товара, например,

- пропан и этанол, и
- рис и пшеница и т. д.

Для простоты обозначим их как товар 0 и товар 1.

Спрос на каждый товар зависит от цены обоих товаров:

$$\begin{aligned} q_0^d &= 100 - 10p_0 - 5p_1 \\ q_1^d &= 50 - p_0 - 10p_1 \end{aligned} \tag{8.1}$$

Мы предполагаем, что спрос снижается при росте цены любого из товаров, но возможны и другие случаи.

Предположим, что предложение определяется как

$$\begin{aligned} q_0^s &= 10p_0 + 5p_1 \\ q_1^s &= 5p_0 + 10p_1 \end{aligned} \tag{8.2}$$

В равновесии спрос равен предложению ($q_0^s = q_0^d$ и $q_1^s = q_1^d$).

Это дает линейную систему

$$\begin{aligned} 100 - 10p_0 - 5p_1 &= 10p_0 + 5p_1 \\ 50 - p_0 - 10p_1 &= 5p_0 + 10p_1 \end{aligned} \tag{8.3}$$

Мы можем решить эту систему с помощью карандаша и бумаги и получить

$$p_0 = 4.41 \quad \text{and} \quad p_1 = 1.18.$$

Подставив эти результаты в (8.1) или (8.2), получим равновесные величины

$$q_0 = 50 \quad \text{and} \quad q_1 = 33.82.$$

1.1.1. Модель в матричной форме

Теперь мы можем численно решить (8.3) с помощью матричной алгебры.

Сначала перепишем (8.1) как

$$q^d = Dp + h \quad \text{где} \quad q^d = \begin{bmatrix} q_0^d \\ q_1^d \end{bmatrix} \quad D = \begin{bmatrix} -10 & -5 \\ -1 & -10 \end{bmatrix} \quad \text{и} \quad h = \begin{bmatrix} 100 \\ 50 \end{bmatrix}. \tag{8.5}$$

Напомним, что $p \in \mathbb{R}^2$ — это цены двух товаров.

Перепишем (8.2) в виде

$$q^s = Cp \quad \text{где} \quad q^s = \begin{bmatrix} q_0^s \\ q_1^s \end{bmatrix} \quad \text{и} \quad C = \begin{bmatrix} 10 & 5 \\ 5 & 10 \end{bmatrix}. \tag{8.6}$$

Теперь условие равновесия примет вид $q^s = q^d$ или

$$Cp = Dp + h.$$

Перегруппируем

$$(C - D)p = h.$$

Если бы все члены были числами, мы могли бы найти решение так: $p = h/(C - D)$.

Матричная алгебра позволяет нам сделать нечто подобное: мы можем найти равновесные цены, используя обратную матрицу к $C - D$:

$$p = (C - D)^{-1}h. \tag{8.7}$$

[1]:
```
import numpy as np
import matplotlib.pyplot as plt

A = ((1, 2), (3, 4))
type(A)
```

[1]:
```
tuple
```

[2]:
```
A: np.array = np.array(A)
type(A)
```

[2]:
```
numpy.ndarray
```

[3]:
```
C = ((10, 5), (5, 10))
C: np.array = np.array(C)
```

[4]:
```
D = ((-10, -5), (-1, -10))
D: np.array = np.array(D)
```

[5]:
```
h: np.array = np.array((100, 50))  # Vector h
h.shape = 2, 1  # Transforming h to a column vector
```

[6]:
```
from numpy.linalg import det, inv

A = C - D
det(A)
```

[6]:
```
np.float64(340.0000000000001)
```

[7]:
```
A_inv = inv(A)  # compute the inverse
p = A_inv @ h  # equilibrium prices
p
```

[7]:
```
array([[4.41176471],
       [1.17647059]])
```

[8]:
```
q = C @ p  # equilibrium quantities
q
```

[8]:
```
array([[50.        ],
       [33.82352941]])
```

[9]:
```
from numpy.linalg import solve

p = solve(A, h)  # equilibrium prices
p
```

[9]:
```
array([[4.41176471],
       [1.17647059]])
```

1.1.2. МНК

...

1.2. Немного линейной алгебры: QR-разложение и простейшие алгоритмы его получения

QR-разложение — разложение матрицы в произведение двух матриц

$$A = QR$$

где Q — ортогональная матрица, то есть $Q'Q = QQ' = I$; R — верхнетреугольная матрица.

QR-разложение существует для любой матрицы $A_{r \times c}$. Следует отметить, что есть два вида QR-разложения:

1. Полное. В этом случае матрица Q имеет размер $r \times r$, а матрица R — $r \times c$.
2. Сокращенное. В этом случае есть три варианта:

- $r = c$. Тогда Q и R имеют размер $r \times c = r \times r$.
- $r > c$. Тогда Q имеет размер $r \times c$, а R — $c \times c$.
- $r < c$. Тогда Q имеет размер $r \times r$, а R — $r \times c$.

1.2.1. Процедура Грама-Шмидта

QR-разложение проще всего найти при помощи процедуры Грама-Шмидта.

Следует отметить, что эта процедура позволяет получить сокращенное QR-разложение. Более того, во всех просмотренных мною источниках, в которых описывалась эта процедура, она описывалась для случаев $r \leq c$.

Рассмотрим матрицу $A = (a_1 \mid a_2 \mid \cdots \mid a_c)$.

1. Примем $u_1 = a_1$. Нормализуем его $e_1 = \frac{u_1}{\|u_1\|}$.
2. Рассчитаем $u_2 = a_2 - (a_2 \cdot e_1)e_1$. Соответственно $e_2 = \frac{u_2}{\|u_2\|}$.
3. Продолжая рассуждения дальше, получим:

$$u_{k+1} = a_{k+1} - (a_{k+1} \cdot e_1)e_1 - \cdots - (a_{k+1} \cdot e_k)e_k = a_{k+1} - \sum_{i=1}^{k}(a_{k+1} \cdot e_i)e_i$$

и $e_{k+1} = \frac{u_{k+1}}{\|u_{k+1}\|}$.

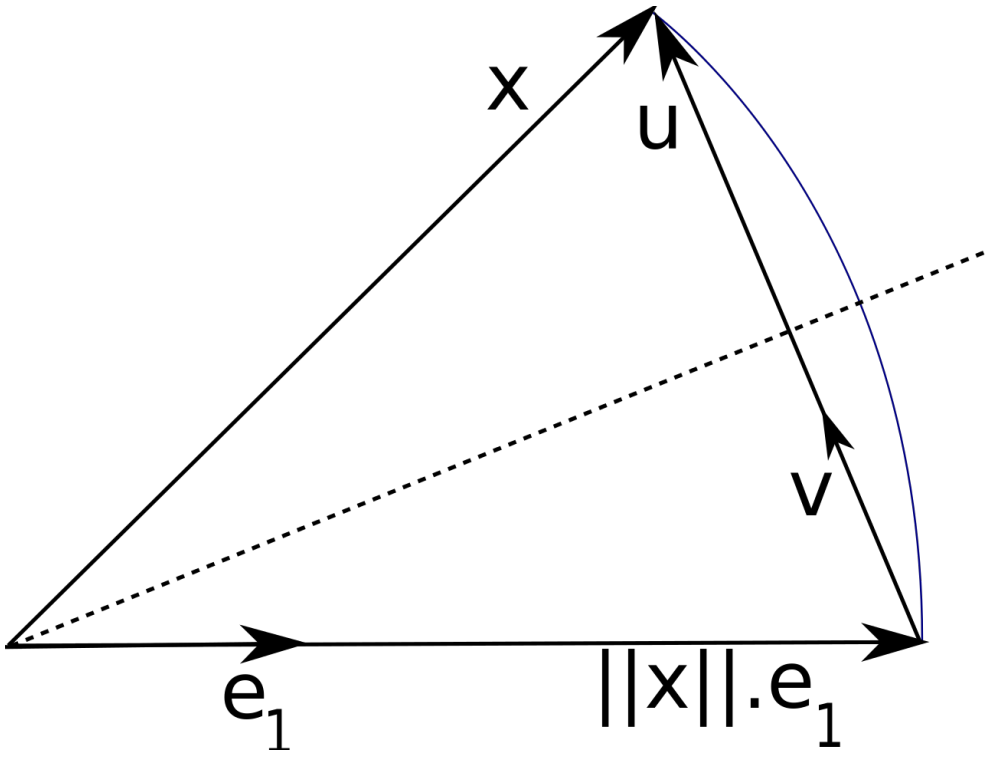
В итоге имеем

$$Q = (e_1 \mid e_2 \mid \cdots \mid e_r)$$

$$R = \begin{pmatrix} a_1 \cdot e_1 & a_2 \cdot e_1 & \dots & a_n \cdot e_1 & \dots \\ 0 & a_2 \cdot e_2 & \dots & a_n \cdot e_2 & \dots \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & a_n \cdot e_n & \dots \end{pmatrix} = Q'A$$

[10]:
```python
import numpy as np


def QR(A: np.ndarray) -> tuple[np.ndarray, np.ndarray]:
    r, c = A.shape

    # assert r ≤ c

    Q = np.empty((r, min(r, c)))
    u = np.empty(r)

    for i in range(min(r, c)):
        u[:] = A[:, i]
        for j in range(i):
            u[:] -= (A[:, i] @ Q[:, j]) * Q[:, j]  # Внимание
        Q[:, i] = u / np.linalg.norm(u)

    R = Q.T @ A

    return Q, R
```

1.3. Отражения Хаусхольдера



Bruguiea, CC BY-SA 3.0 https://creativecommons.org/licenses/by-sa/3.0, via Wikimedia Commons

Процедура Грама-Шмидта численно неустойчива. Особенно большие проблемы могут возникнуть в случае, когда x и $e_1 = [1, 0, \dots, 0]$ почти ортогональны.

Более устойчивой процедурой является разложение на основе отражений Хаусхольдера. На рисунке показана основная идея метода. Мы не ищем проекцию x на e_1, а находим гиперплоскость (на рисунке — пунктирная линия), являющуюся биссектрисой угла между x и e_1. Гиперплоскость определяется через ортогональный ей вектор v.

Рассмотрим матрицу $A = [a_1 \mid a_2 \mid \cdots \mid a_c]$.

Приняв $e_1 = [1, 0, \dots, 0]'$, в соответствии с рисунком получим

$$u = x - \underbrace{\|x\|}_{\alpha} e_1$$

$$v = u / \|u\|$$

Построим следующую матрицу

$$Q = I_r - 2vv'$$

Основное свойство этой матрицы в том, что $Qx = [\alpha, 0, \dots, 0]$.

Если умножить A слева на Q, построенную по первому столбцу, то мы получим

$$QA \equiv Q_1 A = \left(\begin{array}{c|ccc} \alpha & \star & \star & \star \\ \hline 0 & & & \\ \vdots & & \tilde{A} & \\ 0 & & & \end{array}\right)$$

Вышеописанную процедуру можно повторить для подматрицы $\tilde{A}$, получив матрицу Q_2 размера $(r-1)\times(c-1)$. Умножив $\tilde{A}$ на Q_2 получим подматрицу $\hat{A}$.

Продолжая эти шаги, мы получим набор матриц $Q_1, Q_2, \dots, Q_{\min(r,c)}$.

Матрицы размера меньшего, чем $r \times c$, дополним до нужного размера следующим образом

$$\tilde{Q}_k = \left(\begin{array}{c|c} I_{r-k} & 0 \\ \hline 0 & Q_k \end{array}\right)$$

Матрица Q интересующего нас QR-разложения равна

$$Q = \tilde{Q}_1 \tilde{Q}_2 \dots \tilde{Q}_{\min(r,c)}$$

[11]:
```
def QRH(A):
    r, c = A.shape

    M = np.copy(A)
    Q = np.eye(r)

    e = np.zeros((r, 1))
    e[0, 0] = 1

    v = np.zeros((r, 1))

    for i in range(min(r, c)):
        v[:, :] = M[:, [0]] - np.linalg.norm(M[:, [0]]) *
↪e[:, [0]]
        v = v / sl.norm(v)

        # Матрица Хаусхольдера
        H = np.eye(r - i) - 2 * v @ v.T

        # Расширяем матрицу H и умножаем ее слева на Q
        Q = Q @ sl.block_diag(np.eye(i), H)

        # Домножаем слева и отбрасываем первый столбец и
↪строку
        M = (H @ M)[1:, 1:]

        # Укорачиваем e и v
        e = e[:-1, :]
        v = v[:-1, :]

    R = Q.T @ A

    return Q, R
```

Смелые и отважные могут ознакомиться с подробным описанием и выкладками здесь:

Golub G.H. Matrix computations : Johns Hopkins studies in the mathematical sciences / G.H. Golub, C.F. Van Loan. – Fourth edition. – Baltimore: The Johns Hopkins University Press, 2013. – 756 c.