Министерство науки и высшего образования Российской Федерации

Московский государственный технический университет имени

Н. Э. Баумана

Факультет: Информатика и системы управления
Кафедра: Информационная безопасность (ИУ8)

Лабораторная работа №1

ПО ДИСЦИПЛИНЕ «ТЕОРИЯ ИГР И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ»

«Метод Брауна-Робинсон»

Вариант 11

Студент: Кириченков А.А., ИУ8-104
Преподаватель: Коннова Н. С.

Москва
2022

Цель и задачи

Цель работы – изучить аналитический (обратной матрицы) и численный (Брауна-Робинсон) методы нахождения смешанных стратегий в антагонистической игре двух лиц в нормальной форме.

Постановка задачи – найти цену игры и оптимальные стратегии обоих игроков методами обратной матрицы и Брауна-Робинсов, затем сравнить полученные результаты.

Выполнение лабораторной работы

1 Аналитический метод

(3 × 3)-игра Г задана платёжной матрицей:

$$C = \begin{pmatrix} 13 & 3 & 14 \\ 15 & 5 & 0 \\ 7 & 19 & 13 \end{pmatrix}.$$

Для расчёта методом обратной матрицы применяются следующие формулы:

$$x^* = \frac{uC^{-1}}{uC^{-1}u^T}, y^* = \frac{C^{-1}u^T}{uC^{-1}u^T}, v = \frac{1}{uC^{-1}u^T},$$

где $u = (1, 1, \dots, 1) \in \mathbb{R}^m$ для (m × n)-игры Г.

В соответствии с расчётом по заданным формулам для метода обратной матрицы получены следующие значения:

– стоимость игры $v = 10.682$,

– оптимальная смешанная стратегия игрока A $x^* = (0.63, 0.24, 0.12)$,

– оптимальная смешанная стратегия игрока B $y^* = (0.34, 0.2, 0.45)$.

2 Метод Брауна-Робинсона

В таблице ниже приведены этапы расчёта смешанных стратегий игроков A и B, а также оценки игры при помощи метода Брауна-Робинсон с уровнем погрешности $\varepsilon \leq 0.1$:

Таблица 1 – Этапы расчёта стратегий методом Брауна-Робинсон

k	A	B	x1	x2	x3	y1	y2	y3	UpBound	LowBound	EPS
1	1	1	13	15	7	13	3	14	15	3	12
2	2	2	16	20	26	28	8	14	13	4	9
3	3	2	19	25	45	35	27	27	15	9	4
4	3	2	22	30	64	42	46	40	16	10	3
5	3	3	36	30	77	49	65	53	77/5	49/5	3
6	3	1	49	45	84	56	84	66	14	28/3	3
7	3	1	62	60	91	63	103	79	13	9	3
8	3	1	75	75	98	70	122	92	49/4	35/4	9/4

9	3	1	88	90	105	77	141	105	35/3	77/9	5/3
10	3	1	101	105	112	84	160	118	56/5	42/5	6/5
11	3	1	114	120	119	91	179	131	120/11	91/11	10/11
12	2	1	127	135	126	106	184	131	45/4	53/6	10/11
13	2	1	140	150	133	121	189	131	150/13	121/13	10/11
14	2	1	153	165	140	136	194	131	165/14	131/14	10/11
15	2	3	167	165	153	151	199	131	167/15	131/15	10/11
16	1	3	181	165	166	164	202	145	181/16	145/16	10/11
17	1	3	195	165	179	177	205	159	195/17	159/17	10/11
18	1	3	209	165	192	190	208	173	209/18	173/18	10/11
19	1	3	223	165	205	203	211	187	223/19	187/19	10/11
20	1	3	237	165	218	216	214	201	237/20	201/20	189/220
21	1	3	251	165	231	229	217	215	251/21	215/21	155/231
22	1	3	265	165	244	242	220	229	265/22	10	155/231
23	1	2	268	170	263	255	223	243	268/23	223/23	155/231
24	1	2	271	175	282	268	226	257	47/4	113/12	155/231
25	3	2	274	180	301	275	245	270	301/25	49/5	155/231
26	3	2	277	185	320	282	264	283	160/13	132/13	155/231
27	3	2	280	190	339	289	283	296	113/9	283/27	127/297
28	3	2	283	195	358	296	302	309	179/14	74/7	26/77
29	3	1	296	210	365	303	321	322	365/29	303/29	26/77
30	3	1	309	225	372	310	340	335	62/5	31/3	26/77
31	3	1	322	240	379	317	359	348	379/31	317/31	26/77
32	3	1	335	255	386	324	378	361	193/16	81/8	26/77
33	3	1	348	270	393	331	397	374	131/11	331/33	26/77
34	3	1	361	285	400	338	416	387	200/17	169/17	26/77
35	3	1	374	300	407	345	435	400	407/35	69/7	26/77
36	3	1	387	315	414	352	454	413	23/2	88/9	26/77
37	3	1	400	330	421	359	473	426	421/37	359/37	26/77
38	3	1	413	345	428	366	492	439	214/19	183/19	26/77
39	3	1	426	360	435	373	511	452	145/13	373/39	26/77
40	3	1	439	375	442	380	530	465	221/20	19/2	26/77
41	3	1	452	390	449	387	549	478	452/41	387/41	26/77
42	1	1	465	405	456	400	552	492	155/14	200/21	26/77
43	1	1	478	420	463	413	555	506	478/43	413/43	26/77
44	1	1	491	435	470	426	558	520	491/44	213/22	26/77
45	1	1	504	450	477	439	561	534	56/5	439/45	26/77
46	1	1	517	465	484	452	564	548	517/46	226/23	26/77
47	1	1	530	480	491	465	567	562	530/47	465/47	26/77

48	1	1	543	495	498	478	570	576	181/16	239/24	26/77
49	1	1	556	510	505	491	573	590	556/49	491/49	26/77
50	1	1	569	525	512	504	576	604	569/50	252/25	26/77
51	1	1	582	540	519	517	579	618	194/17	517/51	26/77
52	1	1	595	555	526	530	582	632	595/52	265/26	26/77
53	1	1	608	570	533	543	585	646	608/53	543/53	26/77
54	1	1	621	585	540	556	588	660	23/2	278/27	26/77
55	1	1	634	600	547	569	591	674	634/55	569/55	26/77
56	1	1	647	615	554	582	594	688	647/56	291/28	26/77
57	1	1	660	630	561	595	597	702	220/19	595/57	26/77
58	1	1	673	645	568	608	600	716	673/58	300/29	26/77
59	1	2	676	650	587	621	603	730	676/59	603/59	26/77
60	1	2	679	655	606	634	606	744	679/60	101/10	26/77
61	1	2	682	660	625	647	609	758	682/61	609/61	26/77
62	1	2	685	665	644	660	612	772	685/62	306/31	26/77
63	1	2	688	670	663	673	615	786	688/63	205/21	26/77
64	1	2	691	675	682	686	618	800	691/64	309/32	101/448
65	1	2	694	680	701	699	621	814	701/65	621/65	97/455
66	3	2	697	685	720	706	640	827	120/11	320/33	97/455
67	3	2	700	690	739	713	659	840	739/67	659/67	97/455
68	3	2	703	695	758	720	678	853	379/34	339/34	97/455
69	3	2	706	700	777	727	697	866	259/23	697/69	97/455
70	3	2	709	705	796	734	716	879	398/35	358/35	97/455
71	3	2	712	710	815	741	735	892	815/71	735/71	97/455
72	3	2	715	715	834	748	754	905	139/12	187/18	97/455
73	3	1	728	730	841	755	773	918	841/73	755/73	97/455
74	3	1	741	745	848	762	792	931	424/37	381/37	97/455
75	3	1	754	760	855	769	811	944	57/5	769/75	97/455
76	3	1	767	775	862	776	830	957	431/38	194/19	97/455
77	3	1	780	790	869	783	849	970	79/7	783/77	97/455
78	3	1	793	805	876	790	868	983	146/13	395/39	97/455
79	3	1	806	820	883	797	887	996	883/79	797/79	97/455
80	3	1	819	835	890	804	906	1009	89/8	201/20	97/455
81	3	1	832	850	897	811	925	1022	299/27	811/81	97/455
82	3	1	845	865	904	818	944	1035	452/41	409/41	97/455
83	3	1	858	880	911	825	963	1048	911/83	825/83	97/455
84	3	1	871	895	918	832	982	1061	153/14	208/21	97/455
85	3	1	884	910	925	839	1001	1074	185/17	839/85	97/455
86	3	1	897	925	932	846	1020	1087	466/43	423/43	97/455

87	3	1	910	940	939	853	1039	1100	940/87	853/87	97/455
88	2	1	923	955	946	868	1044	1100	955/88	217/22	97/455
89	2	1	936	970	953	883	1049	1100	970/89	883/89	97/455
90	2	1	949	985	960	898	1054	1100	197/18	449/45	97/455
91	2	1	962	1000	967	913	1059	1100	1000/91	913/91	97/455
92	2	1	975	1015	974	928	1064	1100	1015/92	232/23	97/455
93	2	1	988	1030	981	943	1069	1100	1030/93	943/93	97/455
94	2	1	1001	1045	988	958	1074	1100	1045/94	479/47	97/455
95	2	1	1014	1060	995	973	1079	1100	212/19	973/95	97/455
96	2	1	1027	1075	1002	988	1084	1100	1075/96	247/24	97/455
97	2	1	1040	1090	1009	1003	1089	1100	1090/97	1003/97	97/455
98	2	1	1053	1105	1016	1018	1094	1100	1105/98	509/49	97/455
99	2	1	1066	1120	1023	1033	1099	1100	1120/99	1033/99	97/455
100	2	1	1079	1135	1030	1048	1104	1100	227/20	262/25	97/455
101	2	1	1092	1150	1037	1063	1109	1100	1150/101	1063/101	97/455
102	2	1	1105	1165	1044	1078	1114	1100	1165/102	539/51	97/455
103	2	1	1118	1180	1051	1093	1119	1100	1180/103	1093/103	158/6695

Для достижения заданной погрешности ε вычислений было выполнено $k = 103$ итерации. При этом были получены следующие результаты:

– смешанная стратегия игрока А $x^* = \left(\frac{34}{103}, \frac{21}{103}, \frac{48}{103}\right)$,

– смешанная стратегия игрока В $y^* = \left(\frac{71}{103}, \frac{23}{103}, \frac{9}{103}\right)$,

– стоимость игры $v_{br} = \frac{1}{2}\left(\max\limits_k \frac{1}{k}\underline{v}[k] + \min\limits_k \frac{1}{k}\overline{v}[k]\right) =$

$$= \min_k \frac{1}{k}\overline{v}[k] - \frac{\varepsilon[k]}{2} = \max_k \frac{1}{k}\underline{v}[k] + \frac{\varepsilon[k]}{2} = \frac{71624}{6695} \approx 10.698.$$

Погрешность стоимости игры, полученной методом Брауна-Робинсон, относительно полученной методом обратной матрицы стоимости составила:

$$\delta_v = \frac{(v - v_{br})}{v} \cdot 100\% \approx 0.016\%.$$

Выводы

В результате выполнения лабораторной работы получены следующие результаты:

– изучен и реализован аналитический (обратной матрицы) метод нахождения смешанных стратегий в антагонистической игре двух лиц в нормальной форме. Получена стоимость игры: 10.682;

– изучен и реализован численный метод (Брауна-Робинсон) нахождения смешанных стратегий в антагонистической игре двух лиц в нормальной форме. За 103 итерации при заданной погрешности вычислений получена стоимость игры: 10.698;

– при заданной погрешности вычислений $\varepsilon \leq 0.1$ для метода Брауна-Робинсон оценка средней стоимости игры относительно стоимости игры, полученной методом обратной матрицы, имеет относительную погрешность 0.016%.

Приложение А

```
import math
import fractions
import random
import numpy as np
from prettytable import PrettyTable
import warnings
warnings.filterwarnings('ignore')

def ask_user():
    check = str(input("Load conditions from file? (Y/N): ")).lower().strip()
    try:
        if check[0] == 'y':
            return True
        elif check[0] == 'n':
            return False
        else:
            print('Invalid Input')
            return ask_user()
    except Exception as error:
        print("Please enter valid inputs")
        print(error)
        return ask_user()

def get_conditions_file():
    filename = str(input("Enter filename: "))
    lines = []

    with open(filename, 'r') as file:
        lines = file.readlines()
    try:
        eps = float(lines[0])
        rows_number = int(lines[1])
        column_number = int(lines[2])
        entries = list(map(int, lines[3].split()))

        matrix = np.array(entries).reshape(rows_number, column_number)
    except ValueError as e:
        print(f"Incorrect values: {e}")
    return eps, matrix

def get_condiditions_user_input():
    try:
        eps = float(input("Enter eps: "))
        rows_number = int(input("Enter the number of rows:"))
        column_number = int(input("Enter the number of columns:"))

        print("Enter the entries in a single line (separated by space): ")

        entries = list(map(int, input().split()))

        matrix = np.array(entries).reshape(rows_number, column_number)
    except ValueError as e:
        print(f"Incorrect values: {e}")
    return eps, matrix
```

```
def get_conditions():
    return get_conditions_file() if ask_user() else
get_condiditions_user_input()

def get_row_by_index(matrix, index):
    return matrix[index]

def get_column_by_index(matrix, index):
    return [matrix[i][index] for i in range(len(matrix))]

def get_max_index(arr):
    result = np.where(arr == np.amax(arr))[0]
    return result[0] if len(result) == 1 else random.choice(result)

def get_min_index(arr):
    result = np.where(arr == np.amin(arr))[0]
    return result[0] if len(result) == 1 else random.choice(result)

def vector_addition(a, b):
    return [i + j for i, j in zip(a, b)]

def brown_robinson_method(matrix, eps, table):
    m = len(matrix)
    n = len(matrix[0])

    x = m * [0]
    y = n * [0]

    curr_strategy_a = 0
    curr_strategy_b = 0

    win_a = m * [0]
    loss_b = n * [0]
    curr_eps = math.inf
    k = 0

    lower_bounds = []
    upper_bounds = []

    while curr_eps > eps:
        k += 1
        win_a = vector_addition(win_a, get_column_by_index(matrix,
curr_strategy_b))
        loss_b = vector_addition(loss_b, get_row_by_index(matrix,
curr_strategy_a))
        x[curr_strategy_a] += 1
        y[curr_strategy_b] += 1

        lower_bound = fractions.Fraction(min(loss_b), k)
        upper_bound = fractions.Fraction(max(win_a), k)
        lower_bounds.append(lower_bound)
        upper_bounds.append(upper_bound)

        curr_eps = min(upper_bounds) - max(lower_bounds)

        table.add_row([k, curr_strategy_a + 1, curr_strategy_b + 1, *win_a,
*loss_b, upper_bound, lower_bound, curr_eps])
```

```
        curr_strategy_a = get_max_index(win_a)
        curr_strategy_b = get_min_index(loss_b)

    cost = max(lower_bounds) + fractions.Fraction(curr_eps, 2)

    x = [fractions.Fraction(i, k) for i in x]
    y = [fractions.Fraction(i, k) for i in y]

    return x, y, cost

def analytical_method(matrix):
    c_inv = np.linalg.inv(matrix)
    u = np.array([[1 for _ in range(len(matrix))]])
    u_t = u.T

    cost = 1 / np.dot(np.dot(u,c_inv),u_t)

    x = c_inv.dot(u_t) * cost
    y = u.dot(c_inv) * cost

    cost = cost[0][0]

    x = [i[0] for i in x]
    y = [i for i in y][0]
    return x, y, cost

def main():
    table = PrettyTable()
    table.field_names = ["k", "A", "B", "x1", "x2", "x3", "y1", "y2", "y3",
"UpBound", "LowBound", "EPS"]
    eps, matrix = get_conditions()

    print("Analytical method")
    x, y, cost = analytical_method(matrix)
    print(f"x = ", *x)
    print(f"y = ", *y)
    print("Cost = {:}, {:.3f}".format(cost, float(cost)))

    print("Brown-Robinson method")
    x, y, cost = brown_robinson_method(matrix, eps, table)
    print(table)
    print("x = (",*x, ")")
    print("y = (",*y, ")")
    print("Cost = {:}, {:.3f}".format(cost, float(cost)))

if __name__ == '__main__':
    main()
```