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 1

Студент: Анаян М. С., ИУ8-104
Преподаватель: Коннова Н. С.

Москва, 2020

Цель и задачи выполнения лабораторной работы

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

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

$$C = \begin{pmatrix} 1 & 11 & 11 \\ 7 & 5 & 8 \\ 16 & 6 & 2 \end{pmatrix}.$$

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 $v = \frac{133}{18} \approx 7.39$,

– оптимальная смешанная стратегия игрока А $x^* = \left(\frac{19}{54}, \frac{10}{27}, \frac{5}{18}\right) = (0.35, 0.37, 0.27)$,

– оптимальная смешанная стратегия игрока В $y^* = \left(\frac{13}{36}, \frac{1}{12}, \frac{5}{9}\right) = (0.36, 0.08, 0.56)$.

Вычисления произведены при помощи SageMath 8.9, с исходным кодом можно ознакомиться в репозитории по ссылке: https://github.com/hms2010/GameTheory/blob/master/lab1/lab1-analytical.ipynb

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 strat	B strat	Win A			Loss B			UpBound	LowBound	Eps
1	1	1	1	7	16	1	11	11	16/1	1/1	15/1
2	3	1	2	14	32	17	17	13	16/1	13/2	19/2
3	3	3	13	22	34	33	23	15	34/3	5/1	29/6
4	3	3	24	30	36	49	29	17	9/1	17/4	5/2
5	3	3	35	38	38	65	35	19	38/5	19/5	11/10
6	3	3	46	46	40	81	41	21	23/3	7/2	11/10
7	1	3	57	54	42	82	52	32	57/7	32/7	11/10
8	1	3	68	62	44	83	63	43	17/2	43/8	11/10
9	1	3	79	70	46	84	74	54	79/9	6/1	11/10
10	1	3	90	78	48	85	85	65	9/1	13/2	11/10
11	1	3	101	86	50	86	96	76	101/11	76/11	38/55
12	1	3	112	94	52	87	107	87	28/3	29/4	7/20
13	1	1	113	101	68	88	118	98	113/13	88/13	7/20
14	1	1	114	108	84	89	129	109	57/7	89/14	7/20
15	1	1	115	115	100	90	140	120	23/3	6/1	7/20
16	1	1	116	122	116	91	151	131	61/8	91/16	7/20
17	2	1	117	129	132	98	156	139	132/17	98/17	7/20
18	3	1	118	136	148	114	162	141	74/9	19/3	7/20
19	3	1	119	143	164	130	168	143	164/19	130/19	7/20
20	3	1	120	150	180	146	174	145	9/1	29/4	7/20
21	3	3	131	158	182	162	180	147	26/3	7/1	7/20
22	3	3	142	166	184	178	186	149	92/11	149/22	7/20
23	3	3	153	174	186	194	192	151	186/23	151/23	7/20
24	3	3	164	182	188	210	198	153	47/6	51/8	7/20
25	3	3	175	190	190	226	204	155	38/5	31/5	7/20
26	2	3	186	198	192	233	209	163	99/13	163/26	7/20
27	2	3	197	206	194	240	214	171	206/27	19/3	7/20
28	2	3	208	214	196	247	219	179	107/14	179/28	7/20
29	2	3	219	222	198	254	224	187	222/29	187/29	7/20
30	2	3	230	230	200	261	229	195	23/3	13/2	7/20
31	2	3	241	238	202	268	234	203	241/31	203/31	7/20
32	1	3	252	246	204	269	245	214	63/8	107/16	7/20
33	1	3	263	254	206	270	256	225	263/33	75/11	7/20
34	1	3	274	262	208	271	267	236	137/17	118/17	7/20
35	1	3	285	270	210	272	278	247	57/7	247/35	7/20
36	1	3	296	278	212	273	289	258	74/9	43/6	7/20
37	1	3	307	286	214	274	300	269	307/37	269/37	61/185
38	1	3	318	294	216	275	311	280	159/19	275/38	61/185
39	1	1	319	301	232	276	322	291	319/39	92/13	61/185
40	1	1	320	308	248	277	333	302	8/1	277/40	61/185
41	1	1	321	315	264	278	344	313	321/41	278/41	61/185
42	1	1	322	322	280	279	355	324	23/3	93/14	61/185
43	2	1	323	329	296	286	360	332	329/43	286/43	61/185
44	2	1	324	336	312	293	365	340	84/11	293/44	61/185
45	2	1	325	343	328	300	370	348	343/45	20/3	61/185
46	2	1	326	350	344	307	375	356	175/23	307/46	61/185
47	2	1	327	357	360	314	380	364	360/47	314/47	61/185
48	3	1	328	364	376	330	386	366	47/6	55/8	61/185
49	3	1	329	371	392	346	392	368	8/1	346/49	61/185
50	3	1	330	378	408	362	398	370	204/25	181/25	61/185
51	3	1	331	385	424	378	404	372	424/51	124/17	26/85
52	3	3	342	393	426	394	410	374	213/26	187/26	26/85
53	3	3	353	401	428	410	416	376	428/53	376/53	26/85
54	3	3	364	409	430	426	422	378	215/27	7/1	26/85
55	3	3	375	417	432	442	428	380	432/55	76/11	26/85
56	3	3	386	425	434	458	434	382	31/4	191/28	26/85
57	3	3	397	433	436	474	440	384	436/57	128/19	26/85
58	3	3	408	441	438	490	446	386	441/58	193/29	26/85
59	2	3	419	449	440	497	451	394	449/59	394/59	26/85
60	2	3	430	457	442	504	456	402	457/60	67/10	26/85
61	2	3	441	465	444	511	461	410	465/61	410/61	26/85
62	2	3	452	473	446	518	466	418	473/62	209/31	26/85
63	2	3	463	481	448	525	471	426	481/63	142/21	26/85

64	2	3	474	489	450	532	476	434	489/64	217/32	26/85		
65	2	3	485	497	452	539	481	442	497/65	34/5	26/85		
66	2	3	496	505	454	546	486	450	505/66	75/11	26/85		
67	2	3	507	513	456	553	491	458	513/67	458/67	26/85		
68	2	3	518	521	458	560	496	466	521/68	233/34	26/85		
69	2	3	529	529	460	567	501	474	23/3	158/23	26/85		
70	1	3	540	537	462	568	512	485	54/7	97/14	26/85		
71	1	3	551	545	464	569	523	496	551/71	496/71	26/85		
72	1	3	562	553	466	570	534	507	281/36	169/24	26/85		
73	1	3	573	561	468	571	545	518	573/73	518/73	26/85		
74	1	3	584	569	470	572	556	529	292/37	529/74	26/85		
75	1	3	595	577	472	573	567	540	119/15	36/5	26/85		
76	1	3	606	585	474	574	578	551	303/38	29/4	26/85		
77	1	3	617	593	476	575	589	562	617/77	562/77	116/385		
78	1	3	628	601	478	576	600	573	314/39	191/26	33/130		
79	1	3	639	609	480	577	611	584	639/79	577/79	33/130		
80	1	1	640	616	496	578	622	595	8/1	289/40	33/130		
81	1	1	641	623	512	579	633	606	641/81	193/27	33/130		
82	1	1	642	630	528	580	644	617	321/41	290/41	33/130		
83	1	1	643	637	544	581	655	628	643/83	7/1	33/130		
84	1	1	644	644	560	582	666	639	23/3	97/14	33/130		
85	2	1	645	651	576	589	671	647	651/85	589/85	33/130		
86	2	1	646	658	592	596	676	655	329/43	298/43	33/130		
87	2	1	647	665	608	603	681	663	665/87	201/29	33/130		
88	2	1	648	672	624	610	686	671	84/11	305/44	33/130		
89	2	1	649	679	640	617	691	679	679/89	617/89	33/130		
90	2	1	650	686	656	624	696	687	343/45	104/15	33/130		
91	2	1	651	693	672	631	701	695	99/13	631/91	33/130		
92	2	1	652	700	688	638	706	703	175/23	319/46	33/130		
93	2	1	653	707	704	645	711	711	707/93	215/31	33/130		
94	2	1	654	714	720	652	716	719	360/47	326/47	33/130		
95	3	1	655	721	736	668	722	721	736/95	668/95	33/130		
96	3	1	656	728	752	684	728	723	47/6	57/8	33/130		
97	3	1	657	735	768	700	734	725	768/97	700/97	33/130		
98	3	1	658	742	784	716	740	727	8/1	358/49	33/130		
99	3	1	659	749	800	732	746	729	800/99	81/11	13/55		
100	3	3	670	757	802	748	752	731	401/50	731/100	13/55		
101	3	3	681	765	804	764	758	733	804/101	733/101	13/55		
102	3	3	692	773	806	780	764	735	403/51	245/34	13/55		
103	3	3	703	781	808	796	770	737	808/103	737/103	13/55		
104	3	3	714	789	810	812	776	739	405/52	739/104	13/55		
105	3	3	725	797	812	828	782	741	116/15	247/35	13/55		
106	3	3	736	805	814	844	788	743	407/53	743/106	13/55		
107	3	3	747	813	816	860	794	745	816/107	745/107	13/55		
108	3	3	758	821	818	876	800	747	821/108	83/12	13/55		
109	2	3	769	829	820	883	805	755	829/109	755/109	13/55		
110	2	3	780	837	822	890	810	763	837/110	763/110	13/55		
111	2	3	791	845	824	897	815	771	845/111	257/37	13/55		
112	2	3	802	853	826	904	820	779	853/112	779/112	13/55		
113	2	3	813	861	828	911	825	787	861/113	787/113	13/55		
114	2	3	824	869	830	918	830	795	869/114	265/38	13/55		
115	2	3	835	877	832	925	835	803	877/115	803/115	13/55		
116	2	3	846	885	834	932	840	811	885/116	811/116	13/55		
117	2	3	857	893	836	939	845	819	893/117	7/1	13/55		
118	2	3	868	901	838	946	850	827	901/118	827/118	13/55		
119	2	3	879	909	840	953	855	835	909/119	835/119	13/55		
120	2	3	890	917	842	960	860	843	917/120	281/40	13/55		
121	2	3	901	925	844	967	865	851	925/121	851/121	13/55		
122	2	3	912	933	846	974	870	859	933/122	859/122	13/55		
123	2	3	923	941	848	981	875	867	941/123	289/41	13/55		
124	2	3	934	949	850	988	880	875	949/124	875/124	13/55		
125	2	3	945	957	852	995	885	883	957/125	883/125	13/55		
126	2	3	956	965	854	1002	890	891	965/126	445/63	13/55		
127	2	2	967	970	860	1009	895	899	970/127	895/127	13/55		
128	2	2	978	975	866	1016	900	907	489/64	225/32	13/55		
129	1	2	989	980	872	1017	911	918	23/3	911/129	13/55		
130	1	2	1000	985	878	1018	922	929	100/13	461/65	13/55		
131	1	2	1011	990	884	1019	933	940	1011/131	933/131	13/55		
132	1	2	1022	995	890	1020	944	951	511/66	236/33	13/55		
133	1	2	1033	1000	896	1021	955	962	1033/133	955/133	13/55		
134	1	2	1044	1005	902	1022	966	973	522/67	483/67	13/55		
135	1	2	1055	1010	908	1023	977	984	211/27	977/135	13/55		
136	1	2	1066	1015	914	1024	988	995	533/68	247/34	13/55		
137	1	2	1077	1020	920	1025	999	1006	1077/137	999/137	13/55		
138	1	2	1088	1025	926	1026	1010	1017	544/69	505/69	13/55		
139	1	2	1099	1030	932	1027	1021	1028	1099/139	1021/139	13/55		

140	1	2	1110	1035	938	1028	1032	1039	111/14	257/35	13/55		
141	1	1	1111	1042	954	1029	1043	1050	1111/141	343/47	13/55		
142	1	1	1112	1049	970	1030	1054	1061	556/71	515/71	13/55		
143	1	1	1113	1056	986	1031	1065	1072	1113/143	1031/143	13/55		
144	1	1	1114	1063	1002	1032	1076	1083	557/72	43/6	13/55		
145	1	1	1115	1070	1018	1033	1087	1094	223/29	1033/145	13/55		
146	1	1	1116	1077	1034	1034	1098	1105	558/73	517/73	13/55		
147	1	1	1117	1084	1050	1035	1109	1116	1117/147	345/49	380/1617		
148	1	1	1118	1091	1066	1036	1120	1127	559/74	7/1	155/814		
149	1	1	1119	1098	1082	1037	1131	1138	1119/149	1037/149	240/1639		
150	1	1	1120	1105	1098	1038	1142	1149	112/15	173/25	17/165		
151	1	1	1121	1112	1114	1039	1153	1160	1121/151	1039/151	100/1661		

Для достижения заданной погрешности ε вычислений было выполнено $k = 151$ итераций. При этом были получены следующие результаты:

– смешанная стратегия игрока A $x^* = \left(\frac{60}{151}, \frac{53}{151}, \frac{38}{151}\right) = (0.40, 0.35, 0.25)$,

– смешанная стратегия игрока B $y^* = \left(\frac{54}{151}, \frac{14}{151}, \frac{83}{151}\right) = (0.36, 0.1, 0.54)$,

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

$$= \min_k \frac{1}{k}\overline{v}[k] - \frac{\varepsilon[k]}{2} = \max_k \frac{1}{k}\underline{v}[k] + \frac{\varepsilon[k]}{2} = \frac{12281}{1661} \approx 7.39.$$

С исходным кодом на языке Python версии 3.*, реализующим метод Брауна-Робинсон, можно ознакомиться в приложении А либо в репозитории hms2010/GameTheory на github.com по ссылке: https://github.com/hms2010/GameTheory/blob/master/lab1/brown-robinson.py.

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

$$\delta_v = \frac{(v - v_{br})}{v} \cdot 100\% \approx 0.07\%.$$

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;

– изучен и реализован численный метод (Брауна-Робинсон) нахождения смешанных стратегий в антагонистической игре двух лиц в нормальной форме;

– 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 0.07%.

Исходные коды программ представлены по ссылке: https://github.com/hms2010/GameTheory/tree/master/lab1.

Приложение А

```
import math
import fractions
import random
from c import *
from sys import stdout

def get_rand_max_index(arr):
    max_i = []
    max_el = max(arr)
    i = arr.index(max_el)
    max_i.append(i)
    while i < len(arr):
        try:
            i = arr.index(max_el, i + 1)
            max_i.append(i)
        except ValueError:
            break
    return random.choice(max_i)

def get_rand_min_index(arr):
    min_i = []
    min_el = min(arr)
    i = arr.index(min_el)
    min_i.append(i)
    while i < len(arr):
        try:
            i = arr.index(min_el, i + 1)
            min_i.append(i)
        except ValueError:
            break
    return random.choice(min_i)

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):
    return arr.index(max(arr))

def get_min_index(arr):
    return arr.index(min(arr))

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

```
int_formatter = "{:^9d}"
float_formatter = "{:>4d}/{:<4d}"
str_formatter = "{:^9s}"
outline = "||"
separator = "|"
line_formatter = (outline + separator.join([int_formatter] * 3)
+ outline + separator.join([int_formatter] * 3)
+ outline + separator.join([int_formatter] * 3)
+ outline + separator.join([float_formatter] * 3) + outline)

header_formatter = (outline + separator.join([str_formatter] * 3)
+ outline + "{:^29s}"
+ outline + "{:^29s}"
+ outline + separator.join([str_formatter] * 3) + outline)

def printHeader(curr_file):
    if curr_file:
        print(header_formatter.format("k", "A strat", "B strat", "Win A", "Loss B",
 "UpBound", "LowBound", "Eps"), file=curr_file)
        print(126 * "-", file=curr_file)

def printLine(curr_file, k, A, B, win_a, loss_b, upper_bound, lower_bound, eps):
    if curr_file:
        print(line_formatter.format(k, A, B, win_a[0], win_a[1], win_a[2], loss_b[0
], loss_b[1], loss_b[2], upper_bound.numerator, upper_bound.denominator, lower_boun
d.numerator, lower_bound.denominator, eps.numerator, eps.denominator), file=curr_fi
le)

out_file = stdout

def brown_robinson_method(C, eps):
    m = len(C)    # A player strategies: strategy row consists of win of A
    n = len(C[0]) # B player strategies: strategy column consist of loss of B

    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 = []

    printHeader(out_file)
```

```
    while (curr_eps > eps):
        k += 1
        win_a = vector_addition(win_a, get_column_by_index(C, curr_strategy_b))
        loss_b = vector_addition(loss_b, get_row_by_index(C, 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)

        printLine(out_file, k, curr_strategy_a + 1, curr_strategy_b + 1, win_a, los
s_b, upper_bound, lower_bound, curr_eps)

        curr_strategy_a = get_rand_max_index(win_a)
        curr_strategy_b = get_rand_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 main():
    eps = float(input("Enter eps: "))
    x, y, cost = brown_robinson_method(C, eps)
    print("x = (",*x, ")", file=out_file)
    print("y = (",*y, ")", file=out_file)
    print("x = (", *[float(i) for i in x], ")", file=out_file)
    print("y = (", *[float(i) for i in y], ")", file=out_file)
    print("Cost is {} = {}".format(cost, float(cost)), file=out_file)

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