МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. БАУМАНА

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

ТЕОРИЯ ИГР И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ

Лабораторная работа №3 на тему:

«Решение матричных игр с нулевой суммой аналитическим (матричным) и численным (Брауна–Робинсона) методами»

Вариант 4

Преподаватель:
Коннова Н.С.

Студент:
Куликова А.В.

Группа:
ИУ8-21М

Цель работы

Найти оптимальные стратегии непрерывной выпукло-вогнутой антагонистической игры аналитическим и численным методами

Постановка задачи

Пусть функция выигрыша (ядро) антагонистической игры, заданной на единичном квадрате непрерывна:

$$H(x, y) \in C(\Pi), \ \Pi = [0,1] \times [0,1].$$

Тогда существуют нижняя и верхняя цены игры, и кроме того

$$h = \overline{h} \equiv \max_F \min_y E(F, y) = \min_G \max_x E(x, G) \equiv \underline{h},$$

где F(x), G(y) – произвольные вероятностные меры выбора стратегий для обоих игроков, заданные на единичном интервале.

Выпукло-вогнутая игра всегда разрешима в чистых стратегиях.

Ход работы

Данные для игры представлена в таблице 1.

Таблица 1 – матрица стратегий

a	b	c	d	e
-15	20/3	40	-12	-24

Результат аналитического метода представлен в рисунке 1.

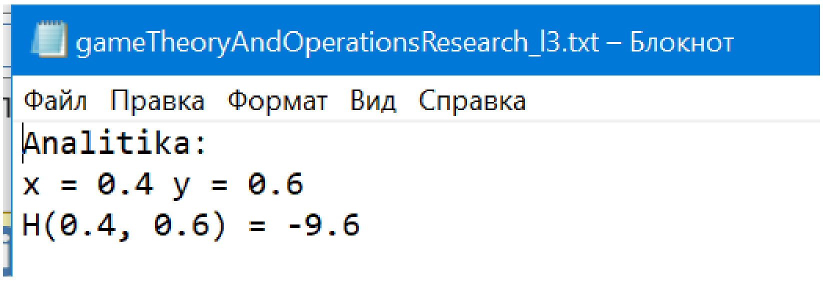
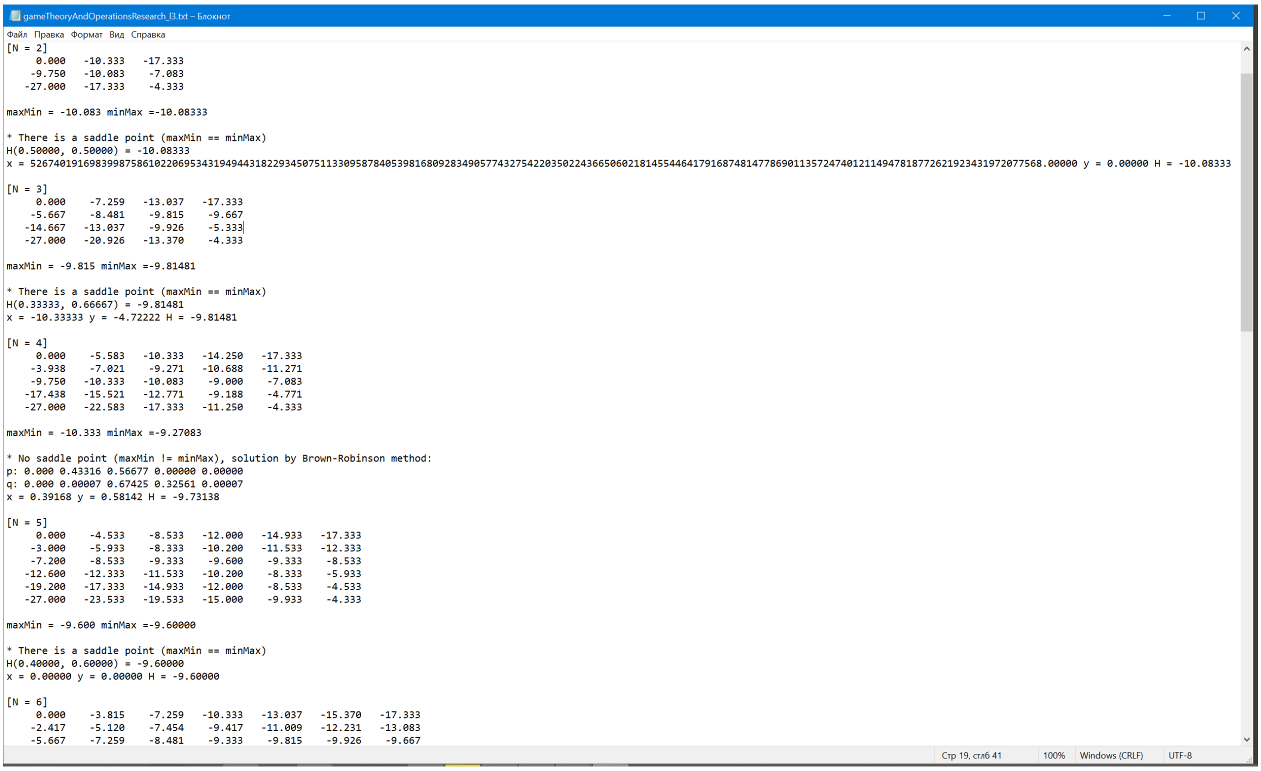


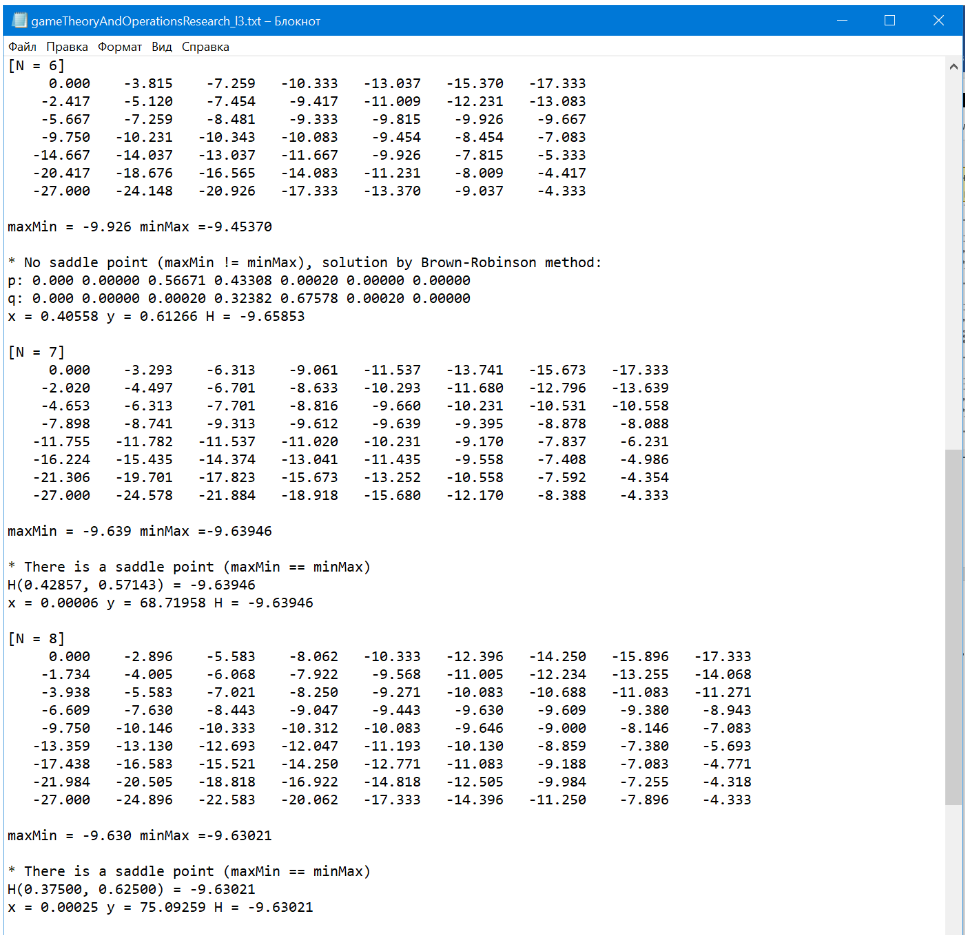
Рисунок 1 – Результат аналитического метода

Результат численного метода представлен в рисунках 2 - 4.



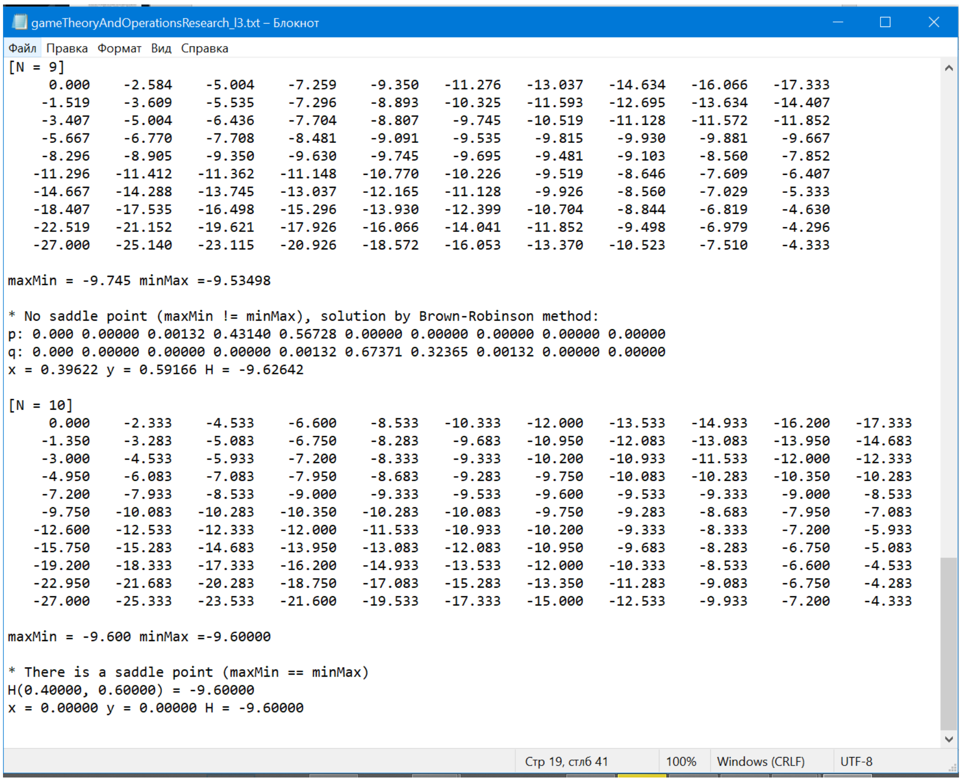
```
gameTheoryAndOperationsResearch_l3.txt – Блокнот
Файл  Правка  Формат  Вид  Справка
[N = 2]
     0.000   -10.333   -17.333
    -9.750   -10.083    -7.083
   -27.000   -17.333    -4.333

maxMin = -10.083 minMax =-10.08333

* There is a saddle point (maxMin == minMax)
H(0.50000, 0.50000) = -10.08333
x = 5267401916983998758610220695343194944318229345075113309587840539816809283490577432754220350224366506021814554464179168748147786901135724740121149478187726219234319720775 68.00000 y = 0.00000 H = -10.08333

[N = 3]
     0.000    -7.259   -13.037   -17.333
    -5.667    -8.481    -9.815    -9.667
   -14.667   -13.037    -9.926    -5.333
   -27.000   -20.926   -13.370    -4.333

maxMin = -9.815 minMax =-9.81481

* There is a saddle point (maxMin == minMax)
H(0.33333, 0.66667) = -9.81481
x = -10.33333 y = -4.72222 H = -9.81481

[N = 4]
     0.000    -5.583   -10.333   -14.250   -17.333
    -3.938    -7.021    -9.271   -10.688   -11.271
    -9.750   -10.333   -10.083    -9.000    -7.083
   -17.438   -15.521   -12.771    -9.188    -4.771
   -27.000   -22.583   -17.333   -11.250    -4.333

maxMin = -10.333 minMax =-9.27083

* No saddle point (maxMin != minMax), solution by Brown-Robinson method:
p: 0.000 0.43316 0.56677 0.00000 0.00000
q: 0.000 0.00007 0.67425 0.32561 0.00007
x = 0.39168 y = 0.58142 H = -9.73138

[N = 5]
     0.000    -4.533    -8.533   -12.000   -14.933   -17.333
    -3.000    -5.933    -8.333   -10.200   -11.533   -12.333
    -7.200    -8.533    -9.333    -9.600    -9.333    -8.533
   -12.600   -12.333   -11.533   -10.200    -8.333    -5.933
   -19.200   -17.333   -14.933   -12.000    -8.533    -4.533
   -27.000   -23.533   -19.533   -15.000    -9.933    -4.333

maxMin = -9.600 minMax =-9.60000

* There is a saddle point (maxMin == minMax)
H(0.40000, 0.60000) = -9.60000
x = 0.00000 y = 0.00000 H = -9.60000

[N = 6]
     0.000    -3.815    -7.259   -10.333   -13.037   -15.370   -17.333
    -2.417    -5.120    -7.454    -9.417   -11.009   -12.231   -13.083
    -5.667    -7.259    -8.481    -9.333    -9.815    -9.926    -9.667
Стр 19, стлб 41   100%   Windows (CRLF)   UTF-8
```

Рисунок 2 – Результат численного метода



```
gameTheoryAndOperationsResearch_l3.txt – Блокнот
Файл  Правка  Формат  Вид  Справка
[N = 6]
     0.000    -3.815    -7.259   -10.333   -13.037   -15.370   -17.333
    -2.417    -5.120    -7.454    -9.417   -11.009   -12.231   -13.083
    -5.667    -7.259    -8.481    -9.333    -9.815    -9.926    -9.667
    -9.750   -10.231   -10.343   -10.083    -9.454    -8.454    -7.083
   -14.667   -14.037   -13.037   -11.667    -9.926    -7.815    -5.333
   -20.417   -18.676   -16.565   -14.083   -11.231    -8.009    -4.417
   -27.000   -24.148   -20.926   -17.333   -13.370    -9.037    -4.333

maxMin = -9.926 minMax =-9.45370

* No saddle point (maxMin != minMax), solution by Brown-Robinson method:
p: 0.000 0.00000 0.56671 0.43308 0.00020 0.00000 0.00000
q: 0.000 0.00000 0.00020 0.32382 0.67578 0.00020 0.00000
x = 0.40558 y = 0.61266 H = -9.65853

[N = 7]
     0.000    -3.293    -6.313    -9.061   -11.537   -13.741   -15.673   -17.333
    -2.020    -4.497    -6.701    -8.633   -10.293   -11.680   -12.796   -13.639
    -4.653    -6.313    -7.701    -8.816    -9.660   -10.231   -10.531   -10.558
    -7.898    -8.741    -9.313    -9.612    -9.639    -9.395    -8.878    -8.088
   -11.755   -11.782   -11.537   -11.020   -10.231    -9.170    -7.837    -6.231
   -16.224   -15.435   -14.374   -13.041   -11.435    -9.558    -7.408    -4.986
   -21.306   -19.701   -17.823   -15.673   -13.252   -10.558    -7.592    -4.354
   -27.000   -24.578   -21.884   -18.918   -15.680   -12.170    -8.388    -4.333

maxMin = -9.639 minMax =-9.63946

* There is a saddle point (maxMin == minMax)
H(0.42857, 0.57143) = -9.63946
x = 0.00006 y = 68.71958 H = -9.63946

[N = 8]
     0.000    -2.896    -5.583    -8.062   -10.333   -12.396   -14.250   -15.896   -17.333
    -1.734    -4.005    -6.068    -7.922    -9.568   -11.005   -12.234   -13.255   -14.068
    -3.938    -5.583    -7.021    -8.250    -9.271   -10.083   -10.688   -11.083   -11.271
    -6.609    -7.630    -8.443    -9.047    -9.443    -9.630    -9.609    -9.380    -8.943
    -9.750   -10.146   -10.333   -10.312   -10.083    -9.646    -9.000    -8.146    -7.083
   -13.359   -13.130   -12.693   -12.047   -11.193   -10.130    -8.859    -7.380    -5.693
   -17.438   -16.583   -15.521   -14.250   -12.771   -11.083    -9.188    -7.083    -4.771
   -21.984   -20.505   -18.818   -16.922   -14.818   -12.505    -9.984    -7.255    -4.318
   -27.000   -24.896   -22.583   -20.062   -17.333   -14.396   -11.250    -7.896    -4.333

maxMin = -9.630 minMax =-9.63021

* There is a saddle point (maxMin == minMax)
H(0.37500, 0.62500) = -9.63021
x = 0.00025 y = 75.09259 H = -9.63021
```

Рисунок 3 – Результат численного метода



```
[N = 9]
     0.000    -2.584    -5.004    -7.259    -9.350   -11.276   -13.037   -14.634   -16.066   -17.333
    -1.519    -3.609    -5.535    -7.296    -8.893   -10.325   -11.593   -12.695   -13.634   -14.407
    -3.407    -5.004    -6.436    -7.704    -8.807    -9.745   -10.519   -11.128   -11.572   -11.852
    -5.667    -6.770    -7.708    -8.481    -9.091    -9.535    -9.815    -9.930    -9.881    -9.667
    -8.296    -8.905    -9.350    -9.630    -9.745    -9.695    -9.481    -9.103    -8.560    -7.852
   -11.296   -11.412   -11.362   -11.148   -10.770   -10.226    -9.519    -8.646    -7.609    -6.407
   -14.667   -14.288   -13.745   -13.037   -12.165   -11.128    -9.926    -8.560    -7.029    -5.333
   -18.407   -17.535   -16.498   -15.296   -13.930   -12.399   -10.704    -8.844    -6.819    -4.630
   -22.519   -21.152   -19.621   -17.926   -16.066   -14.041   -11.852    -9.498    -6.979    -4.296
   -27.000   -25.140   -23.115   -20.926   -18.572   -16.053   -13.370   -10.523    -7.510    -4.333

maxMin = -9.745 minMax =-9.53498

* No saddle point (maxMin != minMax), solution by Brown-Robinson method:
p: 0.000 0.00000 0.00132 0.43140 0.56728 0.00000 0.00000 0.00000 0.00000 0.00000
q: 0.000 0.00000 0.00000 0.00000 0.00132 0.67371 0.32365 0.00132 0.00000 0.00000
x = 0.39622 y = 0.59166 H = -9.62642

[N = 10]
     0.000    -2.333    -4.533    -6.600    -8.533   -10.333   -12.000   -13.533   -14.933   -16.200   -17.333
    -1.350    -3.283    -5.083    -6.750    -8.283    -9.683   -10.950   -12.083   -13.083   -13.950   -14.683
    -3.000    -4.533    -5.933    -7.200    -8.333    -9.333   -10.200   -10.933   -11.533   -12.000   -12.333
    -4.950    -6.083    -7.083    -7.950    -8.683    -9.283    -9.750   -10.083   -10.283   -10.350   -10.283
    -7.200    -7.933    -8.533    -9.000    -9.333    -9.533    -9.600    -9.533    -9.333    -9.000    -8.533
    -9.750   -10.083   -10.283   -10.350   -10.283   -10.083    -9.750    -9.283    -8.683    -7.950    -7.083
   -12.600   -12.533   -12.333   -12.000   -11.533   -10.933   -10.200    -9.333    -8.333    -7.200    -5.933
   -15.750   -15.283   -14.683   -13.950   -13.083   -12.083   -10.950    -9.683    -8.283    -6.750    -5.083
   -19.200   -18.333   -17.333   -16.200   -14.933   -13.533   -12.000   -10.333    -8.533    -6.600    -4.533
   -22.950   -21.683   -20.283   -18.750   -17.083   -15.283   -13.350   -11.283    -9.083    -6.750    -4.283
   -27.000   -25.333   -23.533   -21.600   -19.533   -17.333   -15.000   -12.533    -9.933    -7.200    -4.333

maxMin = -9.600 minMax =-9.60000

* There is a saddle point (maxMin == minMax)
H(0.40000, 0.60000) = -9.60000
x = 0.00000 y = 0.00000 H = -9.60000
```

Рисунок 4 – Результат численного метода

Результат всей программы представлен в приложении А.

Выводы

В ходе проделанной работы были найдены оптимальные стратегии непрерывной выпукло-вогнутой антагонистической игры аналитическим и численным методом.

.

Контрольные вопросы

1. Что такое ядро игры.

Ядро-принцип оптимальности**,** набор из возможных распределений.

Пусть функция выигрыша **(ядро)** антагонистической игры, заданной на единичном квадрате непрерывна:

$$H(x, y) \in C(\Pi), \ \Pi = [0, 1] \times [0, 1].$$

Тогда существует нижняя и верхняя цены игры, и, кроме того,

$$h = \overline{h} \equiv \max_F \min_y E(F, y) = \min_G \max_x E(x, G) \equiv \underline{h},$$

А для среднего выигрыша игры имеют место равенства

$$E(x, G) = \int_0^1 H(x, y) \mathrm{d} G(y), \quad E(F, y) = \int_0^1 H(x, y) \mathrm{d} F(x),$$

Где F(x), G(y)-произвольные вероятностные меры выбора стратегий для обоих игроков, заданные на единичном интервале. Выпукло-вогнутая игра всегда разрешима в чистых стратегиях.

2. Почему выпукло-вогнутая игра всегда разрешима в чистых стратегиях?

Выпукло-вогнутые игры, имеет седлообразное ядро, а так как ядро седлообразное, то **игра** имеет седловую точку **в чистых стратегиях**.

3. Каковы условия выпуклости игры для одного игрока и вогнутости для другого?

Игры с выпуклыми непрерывными функциями выигрышей, называемые часто ядром*,* называются выпуклыми.

Пусть функция ядра имеет вид

$$H(x, y) = ax^2 + by^2 + cxy + dx + ey.$$

Если выполняются условия

$$H_{xx} = 2a < 0, \quad H_{yy} = 2b > 0,$$

То игра является выпукло-вогнутой. Для нахождения оптимальных стратегий находим производные функции ядра по каждой переменной.

$$H_x = 2ax + cy + d, \quad H_y = 2by + cx + e.$$

После приравнивания производных к нулю имеем:

$$x = -\frac{cy + d}{2a}, \quad y = -\frac{cx + e}{2b}.$$

Совместное аналитическое решение имеет вид

$$h = H(x^*, y^*).$$

Приложение А

Результат кода:

```
Analitika:
x = 0.4 y = 0.6
H(0.4, 0.6) = -9.6

[N = 2]
    0.000   -10.333   -17.333
   -9.750   -10.083    -7.083
  -27.000   -17.333    -4.333

maxMin = -10.083 minMax =-10.08333

* There is a saddle point (maxMin == minMax)
H(0.50000, 0.50000) = -10.08333
x =
526740191698399875861022069534319494431822934507511330958784053981680928349057743275422035022436650602
181455446417916874814778690113572474012114947818772621923431972077568.00000 y = 0.00000 H = -10.08333

[N = 3]
    0.000    -7.259   -13.037   -17.333
   -5.667    -8.481    -9.815    -9.667
  -14.667   -13.037    -9.926    -5.333
  -27.000   -20.926   -13.370    -4.333

maxMin = -9.815 minMax =-9.81481

* There is a saddle point (maxMin == minMax)
H(0.33333, 0.66667) = -9.81481
x = -10.33333 y = -4.72222 H = -9.81481

[N = 4]
    0.000    -5.583   -10.333   -14.250   -17.333
   -3.938    -7.021    -9.271   -10.688   -11.271
   -9.750   -10.333   -10.083    -9.000    -7.083
  -17.438   -15.521   -12.771    -9.188    -4.771
  -27.000   -22.583   -17.333   -11.250    -4.333

maxMin = -10.333 minMax =-9.27083

* No saddle point (maxMin != minMax), solution by Brown-Robinson method:
p: 0.000 0.43316 0.56677 0.00000 0.00000
q: 0.000 0.00007 0.67425 0.32561 0.00007
x = 0.39168 y = 0.58142 H = -9.73138

[N = 5]
    0.000    -4.533    -8.533   -12.000   -14.933   -17.333
   -3.000    -5.933    -8.333   -10.200   -11.533   -12.333
   -7.200    -8.533    -9.333    -9.600    -9.333    -8.533
  -12.600   -12.333   -11.533   -10.200    -8.333    -5.933
  -19.200   -17.333   -14.933   -12.000    -8.533    -4.533
  -27.000   -23.533   -19.533   -15.000    -9.933    -4.333

maxMin = -9.600 minMax =-9.60000

* There is a saddle point (maxMin == minMax)
H(0.40000, 0.60000) = -9.60000
x = 0.00000 y = 0.00000 H = -9.60000

[N = 6]
    0.000    -3.815    -7.259   -10.333   -13.037   -15.370   -17.333
```

-2.417	-5.120	-7.454	-9.417	-11.009	-12.231	-13.083
-5.667	-7.259	-8.481	-9.333	-9.815	-9.926	-9.667
-9.750	-10.231	-10.343	-10.083	-9.454	-8.454	-7.083
-14.667	-14.037	-13.037	-11.667	-9.926	-7.815	-5.333
-20.417	-18.676	-16.565	-14.083	-11.231	-8.009	-4.417
-27.000	-24.148	-20.926	-17.333	-13.370	-9.037	-4.333

maxMin = -9.926 minMax =-9.45370

* No saddle point (maxMin != minMax), solution by Brown-Robinson method:
p: 0.000 0.00000 0.56671 0.43308 0.00020 0.00000 0.00000
q: 0.000 0.00000 0.00020 0.32382 0.67578 0.00020 0.00000
x = 0.40558 y = 0.61266 H = -9.65853

[N = 7]

0.000	-3.293	-6.313	-9.061	-11.537	-13.741	-15.673	-17.333
-2.020	-4.497	-6.701	-8.633	-10.293	-11.680	-12.796	-13.639
-4.653	-6.313	-7.701	-8.816	-9.660	-10.231	-10.531	-10.558
-7.898	-8.741	-9.313	-9.612	-9.639	-9.395	-8.878	-8.088
-11.755	-11.782	-11.537	-11.020	-10.231	-9.170	-7.837	-6.231
-16.224	-15.435	-14.374	-13.041	-11.435	-9.558	-7.408	-4.986
-21.306	-19.701	-17.823	-15.673	-13.252	-10.558	-7.592	-4.354
-27.000	-24.578	-21.884	-18.918	-15.680	-12.170	-8.388	-4.333

maxMin = -9.639 minMax =-9.63946

* There is a saddle point (maxMin == minMax)
H(0.42857, 0.57143) = -9.63946
x = 0.00006 y = 68.71958 H = -9.63946

[N = 8]

0.000	-2.896	-5.583	-8.062	-10.333	-12.396	-14.250	-15.896	-17.333
-1.734	-4.005	-6.068	-7.922	-9.568	-11.005	-12.234	-13.255	-14.068
-3.938	-5.583	-7.021	-8.250	-9.271	-10.083	-10.688	-11.083	-11.271
-6.609	-7.630	-8.443	-9.047	-9.443	-9.630	-9.609	-9.380	-8.943
-9.750	-10.146	-10.333	-10.312	-10.083	-9.646	-9.000	-8.146	-7.083
-13.359	-13.130	-12.693	-12.047	-11.193	-10.130	-8.859	-7.380	-5.693
-17.438	-16.583	-15.521	-14.250	-12.771	-11.083	-9.188	-7.083	-4.771
-21.984	-20.505	-18.818	-16.922	-14.818	-12.505	-9.984	-7.255	-4.318
-27.000	-24.896	-22.583	-20.062	-17.333	-14.396	-11.250	-7.896	-4.333

maxMin = -9.630 minMax =-9.63021

* There is a saddle point (maxMin == minMax)
H(0.37500, 0.62500) = -9.63021
x = 0.00025 y = 75.09259 H = -9.63021

[N = 9]

0.000	-2.584	-5.004	-7.259	-9.350	-11.276	-13.037	-14.634	-16.066	-17.333
-1.519	-3.609	-5.535	-7.296	-8.893	-10.325	-11.593	-12.695	-13.634	-14.407
-3.407	-5.004	-6.436	-7.704	-8.807	-9.745	-10.519	-11.128	-11.572	-11.852
-5.667	-6.770	-7.708	-8.481	-9.091	-9.535	-9.815	-9.930	-9.881	-9.667
-8.296	-8.905	-9.350	-9.630	-9.745	-9.695	-9.481	-9.103	-8.560	-7.852
-11.296	-11.412	-11.362	-11.148	-10.770	-10.226	-9.519	-8.646	-7.609	-6.407
-14.667	-14.288	-13.745	-13.037	-12.165	-11.128	-9.926	-8.560	-7.029	-5.333
-18.407	-17.535	-16.498	-15.296	-13.930	-12.399	-10.704	-8.844	-6.819	-4.630
-22.519	-21.152	-19.621	-17.926	-16.066	-14.041	-11.852	-9.498	-6.979	-4.296
-27.000	-25.140	-23.115	-20.926	-18.572	-16.053	-13.370	-10.523	-7.510	-4.333

maxMin = -9.745 minMax =-9.53498

* No saddle point (maxMin != minMax), solution by Brown-Robinson method:
p: 0.000 0.00000 0.00132 0.43140 0.56728 0.00000 0.00000 0.00000 0.00000 0.00000
q: 0.000 0.00000 0.00000 0.00000 0.00132 0.67371 0.32365 0.00132 0.00000 0.00000
x = 0.39622 y = 0.59166 H = -9.62642

[N = 10]

0.000	-2.333	-4.533	-6.600	-8.533	-10.333	-12.000	-13.533	-14.933	-16.200	-17.333
-1.350	-3.283	-5.083	-6.750	-8.283	-9.683	-10.950	-12.083	-13.083	-13.950	-14.683
-3.000	-4.533	-5.933	-7.200	-8.333	-9.333	-10.200	-10.933	-11.533	-12.000	-12.333
-4.950	-6.083	-7.083	-7.950	-8.683	-9.283	-9.750	-10.083	-10.283	-10.350	-10.283
-7.200	-7.933	-8.533	-9.000	-9.333	-9.533	-9.600	-9.533	-9.333	-9.000	-8.533
-9.750	-10.083	-10.283	-10.350	-10.283	-10.083	-9.750	-9.283	-8.683	-7.950	-7.083
-12.600	-12.533	-12.333	-12.000	-11.533	-10.933	-10.200	-9.333	-8.333	-7.200	-5.933
-15.750	-15.283	-14.683	-13.950	-13.083	-12.083	-10.950	-9.683	-8.283	-6.750	-5.083
-19.200	-18.333	-17.333	-16.200	-14.933	-13.533	-12.000	-10.333	-8.533	-6.600	-4.533
-22.950	-21.683	-20.283	-18.750	-17.083	-15.283	-13.350	-11.283	-9.083	-6.750	-4.283
-27.000	-25.333	-23.533	-21.600	-19.533	-17.333	-15.000	-12.533	-9.933	-7.200	-4.333

maxMin = -9.600 minMax =-9.60000

* There is a saddle point (maxMin == minMax)
H(0.40000, 0.60000) = -9.60000
x = 0.00000 y = 0.00000 H = -9.60000

Приложение Б

Листинг Б.1 — braun_robinson.cpp

```
#include "braun_robinson.h"

#include "maxmin.h"
#include <memory.h>
#include <cmath>

namespace gameTheoryAndOperationsResearch {
    gameTheoryAndOperationsResearch::_gameTheoryAndOperationsResearch_maxmin
braun_robinson_maxmin;

    // Классичекий метод Брауна-Робинсона для матрицы игры
    void _gameTheoryAndOperationsResearch_braun_robinson::braun_robinson(double
**pM, double *p, double *q, int n, int m, double& vmin, double& vmax,
std::ofstream & fout)
    {
        // Выделение памяти для хранения частот по строкам и столбцам
        unsigned __int64 *pX = new unsigned __int64[n]; // Частоты по строкам
        unsigned __int64 *pY = new unsigned __int64[m]; // Частоты по столбцам

        // Инициализация переменных и массивов
        unsigned __int64 k1 = 1, k2 = 0; // Общее число выбора строк и столбцов
        double *pV1 = new double[n]; // Суммарный выигрыш 1-го игрока
        double *pV2 = new double[m]; // Суммарный выигрыш 2-го игрока

        for (int i = 0; i < n; i++)
            pV1[i] = pX[i] = 0;
        for (int i = 0; i < m; i++)
            pV2[i] = pY[i] = 0;

        // Находим минимум в матрице игры
        double mymin = braun_robinson_maxmin.min_matrix(pM, n, m);

        // Если минимум отрицательный (есть в матрице отриц элементы), делаем все
элементы положительными, прибавлением одного и того же числа
        if (mymin < 0)
            for (int i = 0; i < n; i++)
                for (int j = 0; j < m; j++)
                    pM[i][j] -= (mymin - 1);

        // Определение первого хода первого игрока по максимину
        int iMax, iMin;
        braun_robinson_maxmin.max_min(pM, n, m, iMax); pX[iMax]++; // Первый ход
первого игрока (по максимину)

        for (int i = 0; i < m; i++)
            pV2[i] += pM[iMax][i];
```

```
        do
        {
            // Выбор второго игрока
            double min = 9e99;
            for (int i = 0; i < m; i++)
                if (pV2[i] < min)
                {
                    min = pV2[i];
                    iMin = i;
                }

            pY[iMin]++; k2++;
            vmin = min / k1; // Верхняя цена игры
            for (int i = 0; i < n; i++)
                pV1[i] += pM[i][iMin];

            // Выбор первого игрока
            double max = 0;
            for (int i = 0; i < n; i++) if (pV1[i] > max)
            {
                max = pV1[i];
                iMax = i;
            }

            pX[iMax]++; k1++;
            vmax = max / k2; // Нижняя цена игры

            for (int i = 0; i < m; i++)
                pV2[i] += pM[iMax][i];

        } while (fabs(vmax - vmin) > 0.001); // Условие остановки

//          // Печатаем число ходов, сделанных каждым игроком
//          fout << "[" << std::endl;
//          fout << "Printing the number of moves made by each player:" <<
std::endl;
//          fout << "k1 = " << k1 << " k2 = " << k2 << std::endl;
//          fout << "]" << std::endl;

        // В случае отрицательных элементов обратный переход к исходной матрице
        if (mymin < 0) {
            vmax += (mymin - 1);
            vmin += (mymin - 1);
        }

        // Расчет оценок вероятностей
        for (int i = 0; i < n; i++)
            p[i] = (double)pX[i] / k1;
        for (int i = 0; i < m; i++)
            q[i] = (double)pY[i] / k2;
```

```
        delete[]pX;
        delete[]pY;
        delete[]pV1;
        delete[]pV2;
    }
}
```

Продолжение приложения Б

Листинг Б.2 — braun_robinson.h

```
#ifndef BRAUN_ROBINSON_H
#define BRAUN_ROBINSON_H

#include <iostream>
#include <fstream>
#include <iomanip>

namespace gameTheoryAndOperationsResearch {
    class _gameTheoryAndOperationsResearch_braun_robinson
    {
        public:
            void braun_robinson(double **, double *, double *, int, int, double&,
double&, std::ofstream &);
    };
}

#endif // BRAUN_ROBINSON_H
```

Продолжение приложения Б

Листинг Б.3 — CMakeLists.txt

```
cmake_minimum_required(VERSION 2.8)

add_executable(main
        main.cpp
        maxmin.h maxmin.cpp
        braun_robinson.h braun_robinson.cpp
        print.h print.cpp
        main.define.h
    )
```

Продолжение приложения Б

Листинг Б.4 — main.cpp
```
#include <iostream>
#include <fstream>
#include <iomanip>

#include "main.define.h"

#include "maxmin.h"
#include "braun_robinson.h"
#include "print.h"

namespace gameTheoryAndOperationsResearch {

    void _gameTheoryAndOperationsResearch_readfile() {
        std::string line;std::ifstream
in(gameTheoryAndOperationsResearch_filename);
        while (std::getline(in, line))
            std::cout << line << std::endl;
    }

    void _gameTheoryAndOperationsResearch_deletefile() {
        std::remove(gameTheoryAndOperationsResearch_filename);
    }
    gameTheoryAndOperationsResearch::_gameTheoryAndOperationsResearch_maxmin
       class_maxmin_main;
    gameTheoryAndOperationsResearch::_gameTheoryAndOperationsResearch_print
      class_print_main;
    gameTheoryAndOperationsResearch::_gameTheoryAndOperationsResearch_braun_robin
son    class_braun_robinson;

    int _main(int argc, char* argv[])
    {
        // Удаляем файл
        _gameTheoryAndOperationsResearch_deletefile();

        // Аналитическое решение
        std::ofstream fout(gameTheoryAndOperationsResearch_filename);

        double x, y;

        // Решение уравнений для определения x и y
        x = ((c*e / (2 * b) - d) / (2 * a - c * c / (2 * b)));
        y = ((c*d / (2 * a) - e) / (2 * b - c * c / (2 * a)));

//        fout << "Kulikova Alyona - v4\n";
        fout << "Analitika: " << std::endl;
        fout << "x = " << x
             << " y = " << y << std::endl;
```

```
        fout << "H(" << x
             << ", " << y
             << ") = " << H(x, y)
             << std::endl;

        // Численный метод
        int N = 2;
        double minMax, maxMin;
        int i, j; // Индексы
        while (true)
        {
            double **Matr = new double*[N + 1];
            double *p = new double[N + 1];
            double *q = new double[N + 1];
            double h = 1. / N;
            x = 0;

            // Заполняем матрицу игры и выводим ее на экран
            fout << std::endl;
            fout << "[" << "N = " << N << "]" << std::endl;
            for (int i = 0; i <= N; i++, x+=h)
            {
                Matr[i] = new double[N + 1];
                y = 0;
                for (int j = 0; j <= N; j++, y += h)
                {
                    Matr[i][j] = H(x, y);
                    fout << std::fixed << std::setprecision(3)
                         << std::setw(10) << Matr[i][j];
                }
                fout << std::endl;
            }

            fout << std::endl;

            // Находим maxMin и minMax для матрицы игры
            maxMin = class_maxmin_main.max_min(Matr, N + 1, N + 1, i);
            minMax = class_maxmin_main.min_max(Matr, N + 1, N + 1, j);

            fout << "maxMin = " << maxMin << std::fixed << std::setprecision(5)
<< std::setw(6)
                 << " minMax =" << minMax << std::fixed << std::setprecision(5)
                 << std::endl;

            fout << std::endl;

            if (maxMin == minMax) {
                // Проверяем наличие седловой точки
//                fout << "Saddle point - Yes (maxMin == minMax)" << std::endl;
```

```
                fout << "* There is a saddle point (maxMin == minMax)" <<
std::endl;
                fout << "H(" << h * i << ", "
                     << h * j << ") = "
                     << Matr[i][j]
                        << std::endl;

                // Вычисляем среднее взвешенное значение
                double x_sr = 0;
                x = 0;

                for (int i = 0; i <= N; i++, x += h)
                    x_sr += x * p[i];

                double y_sr = 0;
                y = 0;

                for (int i = 0; i <= N; i++, y += h)
                    y_sr += y * q[i];

                fout << "x = " << x_sr
                     << " y = " << y_sr
                     << " H = " << minMax
                     << std::endl;

                if (N >= 10)
                    break;
            }
            else
            {
                fout << "* No saddle point (maxMin != minMax), solution by Brown-
Robinson method:" << std::endl;
//                fout << "Saddle point - No (maxMin != minMax)" << std::endl; //
Sedlovaya tochka
//                fout << std::endl;
//                fout << "Braun Robinson" << std::endl;

                // Применяем алгоритм Брауна-Робинсона для решения матричной игры
                class_braun_robinson.braun_robinson(Matr, p, q, N + 1, N + 1,
minMax, maxMin, fout);
                class_print_main.print_vector(fout, (char *)"p", p, N + 1);
                class_print_main.print_vector(fout, (char *)"q", q, N + 1);

                // Вычисляем среднее взвешенное значение
                double x_sr = 0;
                x = 0;

                for (int i = 0; i <= N; i++, x += h)
                    x_sr += x * p[i];
```

```
                double y_sr = 0;
                y = 0;

                for (int i = 0; i <= N; i++, y += h)
                    y_sr += y * q[i];

                fout << "x = " << x_sr
                     << " y = " << y_sr
                     << " H = " << minMax
                     << std::endl;

                if (N >= 10)
                    break;
            }

            delete[] p;
            delete[] q;

            for (int i = 0; i < N + 1; i++)
                delete[] Matr[i];

            delete[] Matr;

            ++N;
        }

        // Считываем файл
        _gameTheoryAndOperationsResearch_readfile();

        return 0;
    }
}

int main(int argc, char* argv[])
{
    gameTheoryAndOperationsResearch::_main(argc, argv);
    return 0;
}
```

Продолжение приложения Б

Листинг Б.5 — main.define.h

```
#ifndef MAIN_DEFINE
#define MAIN_DEFINE

// -15 20/3 40 -12 -24

#define a (double)(-15)
#define b (double)(20. / 3)
#define c (double)40
#define d (double)(-12)
#define e (double)(-24)

#define gameTheoryAndOperationsResearch_filename
"gameTheoryAndOperationsResearch_l3.txt"

#define H(x, y)\
    ((double)(a * x*x + b * y*y + c * x*y + d * x + e * y))

#endif // MAIN_DEFINE
```

Продолжение приложения Б

Листинг Б.6 — maxmin.cpp

```
#include "maxmin.h"

namespace gameTheoryAndOperationsResearch {
    // Функция находит максимальный минимум в матрице и возвращает его значение
    // Параметры:
    // _matrix - двумерный массив значений
    // n - количество строк в матрице
    // m - количество столбцов в матрице
    // iMax - индекс строки, содержащей максимальный минимум
    // Возвращает максимальный минимум в матрице

    double _gameTheoryAndOperationsResearch_maxmin::max_min(double **_matrix, int
n, int m, int& iMax)
    {
        double min;
        double max = -9e99; // Инициализация максимального значения

        for (int i = 0; i < n; i++) // Итерация по строкам матрицы
        {
            min = _matrix[i][0]; // Инициализация минимума для текущей строки

            for (int j = 1; j < m; j++) // Итерация по столбцам матрицы
                if (min > _matrix[i][j]) // Если текущий элемент меньше минимума
                    min = _matrix[i][j]; // Обновляем значение минимума

            if (max < min) // Если текущий минимум больше записанного максимума
            {
                max = min; // Обновляем значение максимума
                iMax = i; // Запоминаем индекс строки
            }
        }
        return max; // Возвращаем найденный максимальный минимум
    }

    // Функция нахождения минимального из максимальных значений в строках матрицы
    // Принимает двумерный массив _matrix, количество строк n, количество
столбцов m и ссылку на iMin
    // Возвращает минимальное из максимальных значений и индекс строки, в которой
оно находится
    double _gameTheoryAndOperationsResearch_maxmin::min_max(double **_matrix, int
n, int m, int& iMin)
    {
        double min = 9e99; // Начальное значение минимума - очень большое число
        double max; // Переменная для хранения максимального значения в столбце
        for (int j = 0; j < m; j++)
        {
```

```
            max = _matrix[0][j]; // Изначально первый элемент в столбце
становится максимальным

            // Находим максимальный элемент в столбце
            for (int i = 1; i < n; i++) // Перебираем все столбцы
                if (max < _matrix[i][j])
                    max = _matrix[i][j];

            // Если найденный максимум меньше текущего минимума
            if (max < min)
            {
                min = max; // Запоминаем новый минимум
                iMin = j; // Сохраняем индекс столбца с минимальным максимальным
значением
            }
        }
        return min; // Возвращаем минимальное из максимальных значений
    }

    // Функция для поиска минимального значения в матрице
    double _gameTheoryAndOperationsResearch_maxmin::min_matrix(double **pM, int
n, int m)
    {
        // Инициализация переменной для хранения минимального значения
        double mymin = pM[0][0];

        // Циклы для перебора всех элементов матрицы
        for (int i = 0; i < n; i++)
            for (int j = 0; j < m; j++)
                // Проверка текущего элемента на то, меньше ли он текущего
минимального значения
                if (pM[i][j] < mymin)
                    // Если элемент меньше текущего минимума, обновляем
минимальное значение
                    mymin = pM[i][j];

        // Возвращаем найденное минимальное значение
        return mymin;
    }
}
```

Листинг Б.7 — maxmin.h

```
#ifndef MAXMIN_H
#define MAXMIN_H

namespace gameTheoryAndOperationsResearch {
    class _gameTheoryAndOperationsResearch_maxmin
    {
        public:
            double max_min(double **, int, int, int&);
            double min_max(double **, int, int, int&);
            double min_matrix(double **, int, int);
    };
}

#endif // MAXMIN_H
```

Продолжение приложения Б

Листинг Б.8 — print.cpp

```
#include "print.h"

namespace gameTheoryAndOperationsResearch {
    void _gameTheoryAndOperationsResearch_print::print_vector(std::ostream& out, char * str, double *p, int n)
    {
        out.precision(3);
        out << str << ": ";
        for (int i = 0; i < n; i++)
            out << p[i] << std::fixed << std::setprecision(5) << " ";
        out << std::endl;
    }
}
```

Продолжение приложения Б

Листинг Б.9 — print.h

```
#ifndef PRINT_H
#define PRINT_H

#include <iostream>
#include <fstream>
#include <iomanip>

namespace gameTheoryAndOperationsResearch {
    class _gameTheoryAndOperationsResearch_print
    {
        public:
           void print_vector(std::ostream&, char *, double *, int);
    };
}


#endif // PRINT_H
```

Приложение В

Ссылка на исходный код: https://github.com/Kulikova-A18/gameTheoryAndOperationsResearch_lab3