**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 №5**

**ПО ДИСЦИПЛИНЕ «ТЕОРИЯ ИГР И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ»**

«Монотонный алгоритм»

**Вариант 1**

**Студент**: Анаян М. С., ИУ8-104

**Преподаватель:** Коннова Н. С.

# Цель и задачи выполнения лабораторной работы

**Цель работы –** изучить аналитический (обратной матрицы) и численный (монотонный алгоритм) методы нахождения смешанных стратегий в антагонистической игре двух лиц в нормальной форме.

**Постановка задачи –** найти цену игры и оптимальную стратегию обоих игрока A методами обратной матрицы и монотонного алгоритма, затем сравнить полученные результаты.

# Выполнение лабораторной работы

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

Для расчёта методом обратной матрицы применяются следующие формулы:

где для (m × n)-игры Г.

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

* стоимость игры ,
* оптимальная смешанная стратегия игрока A .

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 1 приведён Вывод результатов работы монотонного алгоритма с :

Таблица 1 – Вывод результатов работы монотонного алгоритма

|  |
| --- |
| Enter current accuracy: 6  Cost matrix C:  [1, 11, 11]  [7, 5, 8]  [16, 6, 2]  ----------------------------------------------------------------------------------------------------------  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Game solving with monotonous method...  Current iteration: 1  Current strategy: [1, 0, 0]  Current cost: [1, 11, 11]  J: [0]  Current optimal strategy: [1.0994932919678094e-08, 1.944470646334113e-08, 0.9999999695603606]  Current optimal cost: [15.999999660073648, 6.000000035529958, 2.0000002156226353]  Aplha = 0.4166666765702822  ----------------------------------------------------------------------------------------------------------  Current iteration: 2  Current strategy: [0.58333332801094, 8.101961218965034e-09, 0.41666666388709883]  Current cost: [7.250000006918251, 8.916666631952738, 7.2500000007102265]  J: [0, 2]  Current optimal strategy: [1.7448702481586535e-08, 0.9333333040635483, 0.06666667848774918]  Current optimal cost: [7.600000001697527, 5.066666783179964, 7.599999981419612]  Aplha = 0.3968254052787554  ----------------------------------------------------------------------------------------------------------  Current iteration: 3  Current strategy: [0.35185185063448193, 0.3703703715320746, 0.27777777783344343]  Current cost: [7.388888896694099, 7.3888888816403355, 7.388888884902785]  Game solution was found.  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Player A strategy x = [0.351852, 0.37037, 0.277778]  Game cost = 7.388889 |

Получены следующие результаты:

* смешанная стратегия игрока A ,
* стоимость игры .

С исходным кодом на языке Python версии 3.\*, реализующим монотонный метод, можно ознакомиться в приложении А либо в репозитории hms2010/GameTheory на github.com по ссылке: <https://github.com/hms2010/GameTheory/blob/master/lab5/src.py>.

Погрешность стоимости игры, полученной монотонного метода, относительно полученной методом обратной матрицы стоимости составила:

.

# 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 (монотонный алгоритм) нахождения смешанных стратегий в антагонистической игре двух лиц в нормальной форме;
* при заданной погрешности вычислений для монотонного алгоритма оценка стоимости игры относительно стоимости игры, полученной методом обратной матрицы, имеет относительную погрешность .

Исходные коды программ представлены по ссылке: <https://github.com/hms2010/GameTheory/tree/master/lab5>.

# Приложение А

Исходный код в репозитории по ссылке: <https://github.com/hms2010/GameTheory/tree/master/lab5>.

|  |
| --- |
| from scipy.optimize import linprog  from c import \*  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 transpose\_matrix(matrix):      return [get\_column\_by\_index(matrix, i) for i in range(len(matrix[0]))]  def is\_equal\_with\_eps(a, b, eps = 10\*\*-6):      return (abs(a - b) < eps)  def get\_min\_indexes\_list(arr, eps = 10\*\*-6):      min\_i = []      min\_el = min(arr)      for i in range(len(arr)):          if is\_equal\_with\_eps(arr[i], min\_el, eps):              min\_i.append(i)      return min\_i  def simplex\_method(C):      A = transpose\_matrix(C)      for i in range(len(A)):          for j in range(len(A[0])):              A[i][j] \*= -1      b = [-1] \* len(A)      c = [1] \* len(A[0])      res = linprog(c, A\_ub=A, b\_ub=b, bounds=[(0, None) for i in range(len(c))])      cost = 1 / res.fun      strats = [i \* cost for i in res.x]      return strats  def get\_cur\_opt\_cost(A, x):      res = [0] \* len(A[0])      for i in range(len(A[0])):          for j in range(len(A)):              res[i] += x[j] \* A[j][i]      return res  def get\_cur\_strat\_opt(alpha, prev, cur\_opt):      res = [0] \* len(prev)      for i in range(len(prev)):          res[i] += (1 - alpha)\*prev[i] + alpha\*cur\_opt[i]      return res  def check\_all\_cost\_equal(cur\_cost, eps):      for i in range(len(cur\_cost)):          for j in range(i, len(cur\_cost)):              if not is\_equal\_with\_eps(cur\_cost[i], cur\_cost[j], eps):                  return False          return True  def monotonous\_method(C, eps = 10\*\*-6):      print("\*" \* 128)      print("Game solving with monotonous method...")      cur\_strat, cur\_cost = [1] + [0] \* (len(C) - 1), get\_row\_by\_index(C, 0)      N = 0      while True:          N += 1          print("Current iteration: {:d}".format(N))          print("    Current strategy:", cur\_strat)          print("    Current cost:    ", cur\_cost)          if check\_all\_cost\_equal(cur\_cost, eps):              print("Game solution was found.")              print("\*" \* 128)              return cur\_strat, min(cur\_cost)          J = sorted(get\_min\_indexes\_list(cur\_cost, eps))          print("    J:", J)          subgame = []          for i in range(len(C)):              line = []              for j in J:                  line.append(C[i][j])              subgame.append(line)          cur\_opt\_strat = simplex\_method(subgame)          print("    Current optimal strategy:", cur\_opt\_strat)          cur\_opt\_cost = get\_cur\_opt\_cost(C, cur\_opt\_strat)          print("    Current optimal cost:    ", cur\_opt\_cost)          subgame = [cur\_cost, cur\_opt\_cost]          subgame\_cost = simplex\_method(subgame)          alpha = subgame\_cost[1]          print("    Aplha = ", alpha)          print("-" \* 128)          if is\_equal\_with\_eps(0, alpha, eps):              print("Game solution was found.")              print("\*" \* 128)              return cur\_strat, min(cur\_cost)          cur\_strat = get\_cur\_strat\_opt(alpha, cur\_strat, cur\_opt\_strat)          cur\_cost = get\_cur\_strat\_opt(alpha, cur\_cost, cur\_opt\_cost)  def main():      accuracy = int(input("Enter current accuracy: "))      print("Cost matrix C:")      for line in C:          print(line)      print("-" \* 128)      x, cost = monotonous\_method(C, 10\*\*(-accuracy))      print("Player A strategy x =", [round(i, accuracy) for i in x])      print("Game cost = ", round(cost, accuracy))  if \_\_name\_\_ == '\_\_main\_\_':      main() |