Донецкий Национальный Технический Университет

Лабораторная работа № 4

«Приближенное решение матричных игр»

Выполнил:

ст. группы ИПЗ -13

Лысенко А. С.

Проверила:

доцент каф. ПМИ

Дмитриева О. А.

Покровск 2017

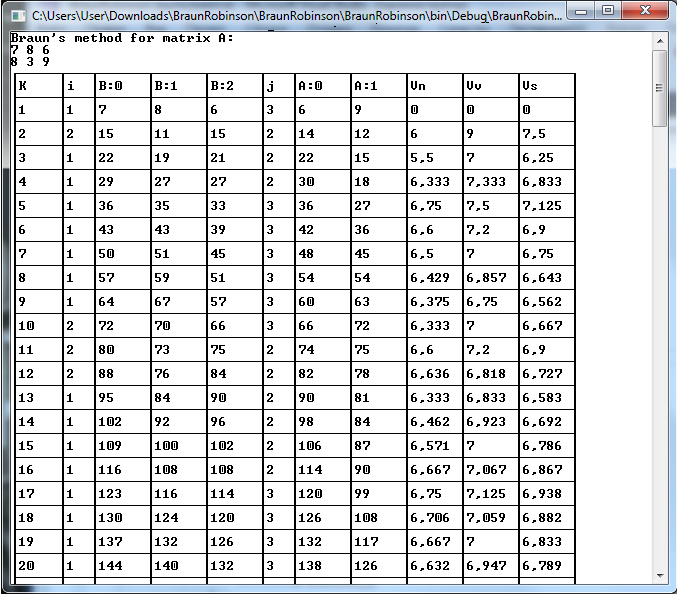
***Цель работы:*** приобретение практических навыков в организации многократного фиктивного разыгрывания матричной игры, определении условий сходимости и статистических вероятностей смешанных стратегий игроков.

A =

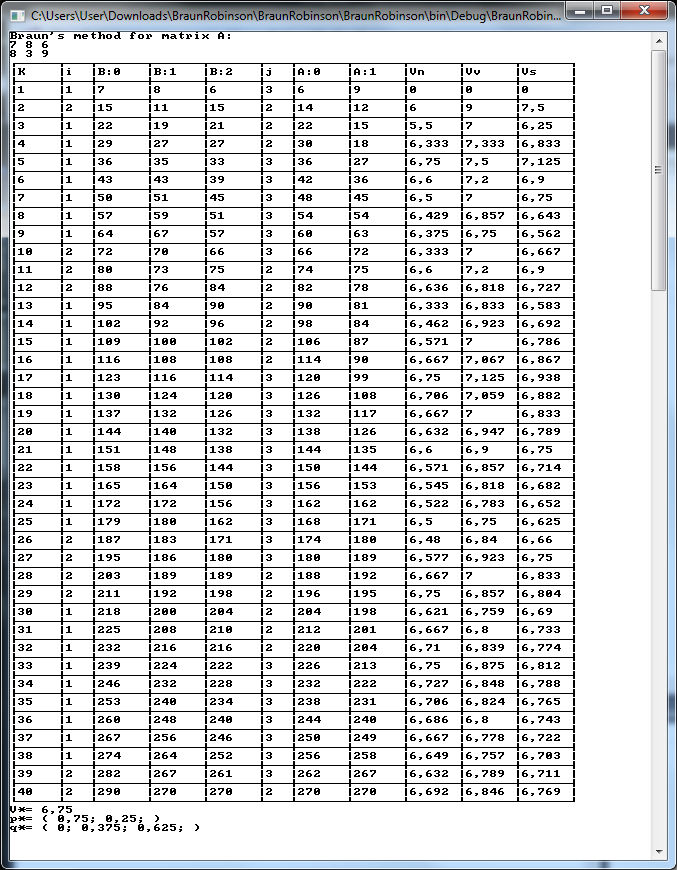
Количество итераций – 20

Т.к. стратегия А1, доминирует над стратегиями А2, А3,А5 получим следующую матрицу

A =



,где  
k - номер партии.  
i - номер стратегии, выбираемой игроком A.  
j - номер стратегии, выбираемой игроком В.  
Bi - накопленный игроком А выигрыш за k партий, при условии, что в данной партии B выбирает стратегию Bi.  
Аj - накопленный игроком В проигрыш за k партий, при условии, что в данной партии A выбирает стратегию Аj.  
Vmin - нижняя оценка игры = min (накопленный выигрыш)/k.  
Vmax - верхняя оценка игры = max (накопленный проигрыш)/k.   
Доказано, что:  
W=(Vmin + Vmax)/2, при k → ∞ и   
pi = Ni/k  
qj = Nj/k  
Ni - сколько раз выбирается Аi стратегия.  
Nj - сколько раз выбирается Bj стратегия.  
NA1 = 16  
P(A1) = 16/20 = 4/5  
NA2 = 4  
P(A2) = 4/20 = 1/5  
NB1 = 0  
P(B3) = 0/20 = 0  
NB2 = 9  
P(B3) = 9/20 = 9/20  
NB3 = 11  
P(B3) = 11/20 = 11/20  
Цена игры, W = 27/4  
Стратегия игрока I: p = (4/5, 1/5)  
Стратегия игрока II: q = (0, 9/20, 11/20)



NA1 = 30  
P(A1) = 30/40 = 3/4  
NA2 = 10  
P(A2) = 10/40 = 1/4  
NB1 = 0  
P(B3) = 0/40 = 0  
NB2 = 15  
P(B3) = 15/40 = 3/8  
NB3 = 25  
P(B3) = 25/40 = 5/8  
Цена игры, W = 27/4  
Стратегия игрока I: p = (3/4, 1/4)  
Стратегия игрока II: q = (0, 3/8, 5/8)

При увеличении числа итераций в двое, заметим что стратегия А1 используется реже, а стратегия А2 чаще, а также стратегия B2 используется реже, а стратегия B3 чаще.

Код программы С#

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace BraunRobinson

{

class Program

{

static void Main(string[] args)

{

Console.BackgroundColor = ConsoleColor.White;

Console.ForegroundColor = ConsoleColor.Black;

double[,] A = { { 7, 8, 6 }, { 8, 3, 9 } };

Console.WriteLine("Braun's method for matrix A:");

printMatr(A);

Braun(A, 0.01);

//double[,] B = { { 1, 6 }, { 5, 4 }, { 0, 7 }, { 3, 5} };

//Console.WriteLine("\n\nBraun's method for matrix B:");

//printMatr(B);

//Braun(B, 0.001);

Console.ReadKey();

}

private static void printMatr(double[,] m)

{

for (int i = 0; i < m.GetLength(0); i++)

{

for (int j = 0; j < m.GetLength(1); j++)

Console.Write(Math.Round(m[i, j], 3) + " ");

Console.WriteLine();

}

}

private static void Braun(double[,] matrix, double eps)

{

int m = matrix.GetLength(0);

int n = matrix.GetLength(1);

List<double> B = new List<double>();

List<double> cB = new List<double>();

List<double> A = new List<double>();

List<double> cA = new List<double>();

for (int i = 0; i < n; i++) { B.Add(0.0); cB.Add(0.0); }

for (int j = 0; j < m; j++) { A.Add(0.0); cA.Add(0.0); }

Random rand = new Random();

int f = rand.Next(m);

int min = 0;

double Vn = 0.0, Vv = 0.0, Vs = 0.0, Vold = 0.0;

int count=0;

int width=6;

Console.Write("┌─────┬───┬");

for (int i = 0; i < B.Count; i++) Console.Write("──────┬");

Console.Write("───┬");

for (int i = 0; i < A.Count + 2; i++) Console.Write("──────┬");

Console.Write("──────┐");

Console.WriteLine();

Console.Write("│K".PadRight(width) + "│" + "i │");

for (int i = 0; i < B.Count; i++) Console.Write("B:"+i.ToString().PadRight(width-2) + "│");

Console.Write("j │");

for (int i = 0; i < A.Count; i++) Console.Write("A:"+i.ToString().PadRight(width-2) + "│");

Console.Write("Vn".PadRight(width)+"│"+"Vv".PadRight(width)+"│"+"Vs".PadRight(width)+"│");

Console.WriteLine();

while (true)

{

Console.Write("├─────┼───┼");

for (int i = 0; i < B.Count; i++) Console.Write("──────┼");

Console.Write("───┼");

for (int i = 0; i < A.Count + 2; i++) Console.Write("──────┼");

Console.Write("──────┤");

Console.WriteLine();

count++;

cA[f]++;

for (int i = 0; i < B.Count; i++) B[i] += matrix[f, i];

min = B.IndexOf(B.Min());

cB[min]++;

for (int j = 0; j < A.Count; j++) A[j] += matrix[j, min];

Console.Write("│" + count.ToString().PadRight(width-1) + "│" + (f + 1).ToString().PadRight(3) + "│");

for (int i = 0; i < B.Count; i++) Console.Write(Math.Round(B[i], 3).ToString().PadRight(width) + "│");

Console.Write((min + 1).ToString().PadRight(3) + "│");

for (int j = 0; j < A.Count; j++) Console.Write(Math.Round(A[j], 3).ToString().PadRight(width) + "│");

Console.Write(Math.Round(Vn,3).ToString().PadRight(width)+"│"+Math.Round(Vv,3).ToString().PadRight(width)+"│"+Math.Round(Vs,3).ToString().PadRight(width)+"│");

Console.WriteLine();

Vn = B.Min() / count;

Vv = A.Max() / count;

Vs = (Vv + Vn)/2;

if ((Math.Abs(Vs - Vv)) < eps) break;

//if ((count%20)==0) Console.ReadLine();

Vold = Vs;

f = A.IndexOf(A.Max());

}

Console.Write("└─────┴───┴"); for (int i = 0; i < B.Count; i++) Console.Write("──────┴");

Console.Write("───┴"); for (int i = 0; i < A.Count + 2; i++) Console.Write("──────┴");

Console.Write("──────┘");

Console.WriteLine();

Console.WriteLine("V\*= " + Math.Round(Vs, 3));

double[] p = new double[A.Count];

for (int i = 0; i < p.Length; i++) p[i] = cA[i] / count;

Console.Write("p\*= ( ");

for (int i = 0; i < p.Length; i++) Console.Write(Math.Round(p[i],3)+"; ");

Console.WriteLine(")");

double[] q = new double[B.Count];

for (int i = 0; i < q.Length; i++) q[i] = cB[i] / count;

Console.Write("q\*= ( ");

for (int i = 0; i < q.Length; i++) Console.Write(Math.Round(q[i],3)+"; ");

Console.WriteLine(")");

}

}

}