ћ»ЌќS–Ќj” » –ќ——»»

’едеральное государственное бюджетное образовательное’
учреждение высшего образовани
Ђярославский государственный университет им. s.. емидоваЇ

афедра математического анализа

урсова работа
ычисление геометрических характеристик симплекса
(—пециальность 01.03.02 sрикладна математика и информатика)

Ќаучный руководитель
____
(степень, звание)
____________ ____________
(подпись,‘»ќ)
Ђ__Ї _______ 20 _ г.
—тудент группы sћ»-31Sќ
____________ ____________
(подпись,‘»ќ)
Ђ__Ї _______ 20 _ г.

ярославль, 2017 г.

Содержание

1 ведение

данной курсовой работе, с помощью программы написанной на зыке $C\#$, будут вычислены некоторые геометрические характеристики симплексов.

sусть $n \in N$, элемент $x \in R^n$ будем записывать в виде $x = (x_1, \dots, x_n)$. „ерез $e_1, \dots, e_n$ обозначаетс канонический базис R^n; считаем $e := (1, \dots, 1)$. л $x \in R^n$ через $\|x\|$ ниже обозначаетс обычна евклидова норма x:

$$\|x\| := (\sum_{i=1}^{n} x_i^2)^{1/2}.$$

sусть C Ч выпуклое тело в R^n, т. е. компактное выпуклое подмножество R^n с непустой внутренностью.

„ерез σC обозначим результат гомотетии C относительно центра тжести с коэффициентом σ. Sудем говорить, что n-мерный симплекс описан вокруг выпуклого тела C, если кажда (n-1)-мерна грань этого симплекса содержит точку C. sримем по определению, что выпуклый многогранник вписан в C, если люба его вершина принадлежит границе C.

ќпределение 1 *–имплекс(размерности n) — это выпукла оболочка $n+1$ точки аффинного пространства, которые предполагаютс аффинно независимыми (то есть не лежат в подпространстве размерности $n-1$). Ёти точки называютс вершинами симплекса.*

sусть S Ч невырожденный симплекс в R^n: ќбозначим вершины S через $x^{(j)} := \{x_1^{(j)}, \dots, x_n^{(j)}\}$, $j = 1, \dots, n+1$. ћатрица A влетс невырожденной:

$$A := \begin{pmatrix} x_1^{(1)} & \dots & x_n^{(1)} & 1 \\ x_1^{(2)} & \dots & x_n^{(2)} & 1 \\ \vdots & \dots & \dots & \vdots \\ x_1^{(n+1)} & \dots & x_n^{(n+1)} & 1 \end{pmatrix} \quad (1)$$

ќбозначим через $\Delta_j(x)$ определитель, который получаетс из Δ заменой j-й строки на строку $(x_1, \dots, x_n, 1)$. –ассмотрим многочлены $\lambda_j(x) := \Delta_j(x)/\Delta$ из $\Pi_1(R^n)$, они обладают свойством $\lambda_j(x(k)) = \delta_j^k$ (здесь δ_j^k Ч символ ронекера). оэффициенты λ_j составлют j-й столбец обратной к A матрице A^{-1}: дальнейшем считаем $A^{-1} = (l_{ij})$, иначе говор,

$$\lambda_j(x) = l_{1,j}x_1 + \dots + l_{n,j}x_n + l_{n+1,j}. \quad (2)$$

ќпределение 2 *л выпуслого тела ¯ обозначим через $d_i(C)$ максимальную длину отрезка, содержащегос в C и параллельного оси x_i и будем называть i-м осевым диаметром C.*

ќпределение 3 *sусть S — невырожденный симплекс, C — выпуклое тело в R^n. ведЄм в рассмотрение величину*

$$\xi(C, S) := \min\{\sigma \geq 1 : \; C \subset \sigma S\}. \quad (3)$$

soложим $\xi(S) := \xi(Q_n; S)$. ќчевидно, $\xi(C; S) = 1$ тогда и только тогда, когда $C \subset S$.

ќпределение 4 *sусть $C = Q_n$, S — невырожденный n-мерный симплекс. ведЄм в рассмотрение характеристику*

$$\xi_n = \min\{\xi(S) : \ S \subset Q_n\}. \tag{4}$$

»так, целью данной работы влетс вычисление таких характеристик симплекса, как $\xi(S)$ и ξ_n. ќ том, как они вычислютс, будет более подробно изложенно в следующих главах работы.

2 еличина $\xi(C;S)$

“еорема 1 *сусть $C \not\subset S$ и $1 \le j \le n$. sредположим, что j- $(n-1)$-мерна грань симплекса $\xi(C,S)S$ (параллельна грани S с уравнением $\lambda_j(x) = 0$) содержит точку C. “огда*

$$\xi(C,S) = (n+1)\max_{x\in C}(-\lambda_j(x)) + 1. \tag{5}$$

“еорема 2 *сусть S — невырожденный симплекс, C — выпуклое тело в R^n. sредположим $C \not\subset S$. сли симплекс $\xi(S)S$ описан вокруг C, то*

$$\max_{x\in C}(-\lambda_1(x)) = \cdots = \max_{x\in C}(-\lambda_{n+1}(x)) \tag{6}$$

“еорема 3 *сусть S — невырожденный симплекс, C — выпуклое тело в R^n и $C \not\subset S$, тогда*

$$\xi(C,S) = (n+1)\max_{1\leqslant k\leqslant n+1}\max_{x\in C}(-\lambda_k(x)) + 1. \tag{7}$$

ќтдельно остановимс на случае $C = Q_n$. ћногочлен из $\Pi_1(R^n)$ принимает минимальное и максимальное значени на Q_n в вершинах куба. свзи с этим в соотношених настоще-го пункта величина $max_C(-\lambda_j)$ при $C = Q_n$ может быть заменена на равную величину $\max_{ver(Q_n)}(-\lambda_j)$. частности, равенство (7) принимает вид

$$\xi(C,S) = (n+1)\max_{1\leqslant k\leqslant n+1}\max_{x\in ver(Q_n)}(-\lambda_k(x)) + 1. \tag{8}$$

а условие (6) сводитс к соотношению

$$\max_{x\in ver(Q_n)}(-\lambda_1(x)) = \cdots = \max_{x\in ver(Q_n)}(-\lambda_{n+1}(x)) \tag{9}$$

Ќапомню, что

$$\xi_n = \min\{\xi(S) : \ S \subset Q_n\}.$$

де $C = Q_n$, S — невырожденный n-мерный симплекс.

—тоит упомнуть, что были найдены точные значени ξ_n дл случаев $n = 1, 2, 3$ и соот-ветствующие им симплексы.

sри $n = 1$ $\xi_1 = 1$. л любого проектора существует 1-вершина Q_1 относительно соот-ветствующего симплекса (который в этой ситуации влетс отрезком).

sри $n = 2$ $\xi_2 = \dfrac{3\sqrt{5}}{5} + 1$, причем симплекс, соответствующий ξ_2, обладает интересным свойством: его одномерные грани(стороны треугольника) отсекают от квадрата треуголь-ники равных площадей.

sри $n = 3$ $\xi_3 = 3$, причЄм существует только два симплекса, которым соответствует ξ_3, и что интересно, их двумерные грани(плоскости, ограничивающие симплекс) отсекают от куба фигуры равных объЄмов.

3 –абота программы

ќписание работы программы стоит разделить на две части. sерва будет рассказывать о нахождении $\xi(S)$, по заданным координатам вершин соответствующего симплекса. тора опишет процесс нахождени ξ_n по заданной размерности n. jлгоритм первой части рассмотрим на примере. »так, n-мерный симплекс задаЄтс через n+1 точку n-мерного пространства. «анесЄм данные в матрицу, как показано в (1), в результате получаем (10)

$$A := \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0.5 & 1 \end{pmatrix} \tag{10}$$

алее дл (10) находим обратную матрицу.

$$A^{-1} := \begin{pmatrix} -0.5 & -0.5 & 1 \\ -1 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \tag{11}$$

—оставим базисные многочлены Ћагранжа:

$$\lambda_1(x) = -0.5x_1 - x_2 + 1$$

$$\lambda_2(x) = -0.5x_1 + x_2$$

$$\lambda_3(x) = 1$$

оспользуемс формулой (8), где $C = Q_n$. ќчевидно, что $-\lambda_j(x)$ достигает своего максимума в точке $x = (x_1, x_2, \ldots, x_n)$, где

$$x_i = \begin{cases} 1, & \text{если } -l_{i,j} > 0 \\ 0, & \text{если } -l_{i,j} \leq 0 \end{cases}$$

л нашего примера имеем:

$$\max(-\lambda_1(x)) = 0.5 \text{при} x_1 = x_2 = 1$$

$$\max(-\lambda_2(x)) = 0.5 \text{при} x_1 = 1, x_2 = 0$$

$$\max(-\lambda_3(x)) = 0 \text{при} x_1 = 0$$

‘ормула (8) даЄт $\xi(S) = 3 * 0.5 + 1 = 2.5$ sроцесс выполнени программы практически ничем не отличаетс от описанного выше алгоритма.

тора часть программы занимаетс приближЄнным вычислением ξ_n, путЄм условно-полного перебора всех n-мерных симплексов в Q_n. ”словность заключаетс в том, что на самом деле перебираютс не все возможные симплексы, коих несчЄтное множество, а с некоторым малым шагом. т.е. кажда координата каждой вершины симплекса менетс в диапазоне от 0 до 1 с шагом в 0.001, и дл каждого такого симплекса находитс $\xi(S)$ по вышеуказанной процедуре. sосле чего по формуле (4) выбираетс минимальный $\xi(S)$. “ак, например, $\xi_2 = 2,3418\ldots$, а $\xi_3 = 3$.

4 «аключение

данной работе поднимаетс така актуальна проблема в математике, как изучение геометрических характеристик симплекса - $\xi(S)$ и ξ_n. Sыли повторно вычислены ξ_2 и ξ_3. ычисление ξ_n при большем n затруднительно, из-за необходимости перебора огромного количества вариантов, хот это и возможно. „то касаетс вычислени $\xi(S)$, то даже при достаточно больших n, врем выполнени программы будет мало.

Список литературы

[1] ħ..Ḱевский - еометрические оценки в полиномиальной интерполции. ярославль, 2012г.

[2] “роелсен Ë. - язык программировани C# 5.0 и платформа .NET 4.5. 2015г.

5 sриложение

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Matrix
{
    class Simplex
    {
        int size;//n характеристика
        public Matrix matrix;
        public Matrix array;//та же сама матрица, но без столбца единиц
        public int Size { get { return size; } }

        public Simplex(Simplex simplex)
        {
            size = simplex.Size;
            this.matrix = new Matrix(size + 1, size + 1);
            this.array = new Matrix(size + 1, size);

            for (int i = 0; i < size + 1; i++)
            {
                for (int j = 0; j < size; j++)
                {
                    this.matrix.array[i, j] = simplex.matrix.array[i, j];
                    this.array.array[i, j] = simplex. matrix.array[i, j];
                }
                this.matrix.array[i, size] = simplex.matrix.array[i, size];
            }
        }

        public Simplex(Matrix matrix)//конструктор
        {
            if(matrix.Row!=matrix.Column+1)
            {
                throw new Exception("ћатрица не подходит");
            }
            this.matrix = new Matrix(matrix.Row, matrix.Column + 1);
            this.array = new Matrix(matrix.Row, matrix.Column);
            size = matrix.Column;
            for(int i = 0; i<size + 1; i++)
            {
                for(int j = 0; j<size; j++)
                {
                    this.matrix.array[i, j] = matrix.array[i, j];
```

```
                this.array.array[i, j] = matrix.array[i, j];
            }
            this.matrix.array[i, size] = 1;
        }
    }

    public Decimal LambdaMax()
    {
        Matrix inverseMatrix = new Matrix(this.matrix.Inverse());
        Decimal[] lambda_i = new Decimal[size+1];
        for (int i = 0; i <= size; i++)
            lambda_i[i] = 0;
        //находим максимальное значение дл -Ћмбда_i
        for (int j=0; j<=size;j++)
        {
            for (int i=0; i<size; i++)
            {
                if(inverseMatrix.array[i,j]<0)
                {
                    lambda_i[j] -= inverseMatrix.array[i, j];
                }
            }
            lambda_i[j] -= inverseMatrix.array[size, j];
        }
        return lambda_i.Max();
    }


    public Decimal Ksi()
    {
        decimal lambda = LambdaMax();
        return (size + 1) * lambda + 1;
    }


    //обновлет матрицу симплекса на новую
    public void RefreshMatrix(Matrix matr)
    {
        if (matr.Row != matr.Column + 1 && matr.Row != this.matrix.Row)
        {
            throw new Exception("Ћатрица не подходит");
        }
        for (int i = 0; i < size + 1; i++)
        {
            for (int j = 0; j < size; j++)
            {
                this.matrix.array[i, j] = matr.array[i, j];
                this.array.array[i, j] = matr.array[i, j];
```

```
                }
                this.matrix.array[i, size] = 1;
            }
        }

        //обновлет матрицу симплекса на новую
        public void RefreshMatrix(Decimal[,] Array)
        {
            Matrix newMatrix =
                new Matrix(this.matrix.Row, this.matrix.Column-1, Array);
            RefreshMatrix(newMatrix);
        }


    }
}


using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;

namespace Matrix
{
    class Simplex_n
    {
        int size;//n характеристика(размер пространства)
        public int Size
        {
            get { return size; }
            set { size = value; }
        }

        public Simplex_n(int Size)
        {
            size = Size;
        }

        //проверет на наличие 0 и 1
        public bool CheckHave1Or0(Decimal[] point)
        {
            for (int i = 0; i < size; i++)
                if (point[i] == 0 || point[i] == 1.000m)
                    return true;
            return false;
        }
```

```
//по точке возвращает следующую
public Decimal[] NewPoint(Decimal[] point)
{
    Decimal[] newPoint = new Decimal[size];
    for (int i = 0; i < size; i++)
        newPoint[i] = point[i];
    int j = size - 1;

    while (j >= 0)
    {
        if (newPoint[j] <= 0.999m)
        {
            newPoint[j] += 0.001m;
            return newPoint;
        }
        /*уменьшаем j до тех пор,
        пока нельз будет прибавить 0,001*/
        while (j >= 0 && newPoint[j] > 0.999m)
        {
            j--;
        }
        /*можем увеличить j-ую координату на 0,001*/
        for (int k = j + 1; k < size; k++)
            newPoint[k] = 0;
    }
    return point;
    //возвращаем саму точку, если не можем перейти к следующей.
    //т.е. вектор уже из всех единиц
}

//получени следующей матрицы, зна предыдущую
public Decimal[,] NextMatrix(Decimal[,] Matrix, int Row)
{
    Matrix matr = new Matrix(Row, Row - 1, Matrix);
    int k = Row;//индекс на единицу больше последней строки
    Decimal[] tmp; //текуща строка
    Decimal[] nextTmp;
    do
    {
        k--;
        tmp = matr.GetRow(k);
        nextTmp = NewPoint(tmp);
        while(!CheckHave1Or0(nextTmp))
            nextTmp = NewPoint(nextTmp);
    }
    //ищем строку, которую можно преобразовать в следующую
    while (k > 0 && Comparison(tmp, nextTmp));
```

```
    //если строчку можно преобразовать
    if (!Comparison(tmp, nextTmp))
    {
        matr.SetRow(k, nextTmp);
        for (int i = k + 1; i < Row; i++)
            for (int j = 0; j < Row - 1; j++)
                matr.array[i, j] = 0m;
    }
    return matr.array;
}

//проверка на равенство двух массивов
public bool Comparison(Decimal[] m1, Decimal[] m2)
{
    if (m1.Length != m2.Length)
        return false;
    int i = 0;
    while (i < m1.Length)
    {
        if (m1[i] != m2[i])
            return false;
        i++;
    }
    return true;
}

//проверка на равенство двух матриц
public bool Comparison(Matrix m1, Matrix m2)
{
    if (m1.Row != m2.Row || m1.Column != m2.Column)
        return false;
    for (int i = 0; i < m1.Row; i++)
        for (int j = 0; j < m1.Column; j++)
            if (m1.Array[i, j] != m2.Array[i, j])
                return false;
    return true;
}

//проверка на то, что достигли конца перебора матриц
public bool CheckOnFinish(Simplex simplex)
{

    for (int i = 0; i < simplex.array.Row; i++)
        for (int j = 0; j < simplex.array.Column; j++)
            if (simplex.array.array[i, j] != 1.000m)
                return false;
```

```
            return true;
        }

        //пропускаем те симплексы, у которых не можем посчитать кси
        public void Skip(ref Simplex sim)
        {
            while (sim.matrix.Determinant() == 0)
            {
                sim.RefreshMatrix(NextMatrix(sim.matrix.Array, sim.matrix.Row));
                if (CheckOnFinish(sim))
                    break;
            }
        }

        public Decimal Ksi_n()
        {
            Decimal[,] array = new Decimal[size + 1, size];
            for(int i = 0; i < size+1; i++)
                for(int j = 0; j < size; j++)
                    array[i, j] = 0m;
            Matrix matrix = new Matrix(size + 1, size, array);
            Simplex sim = new Simplex(matrix);

            Skip(ref sim);
            Decimal minKsi = sim.Ksi();
            Decimal ksi;

            sim.RefreshMatrix(NextMatrix(sim.matrix.Array, sim.matrix.Row));
            Skip(ref sim);

            do
            {
                ksi = sim.Ksi();
                if (ksi < minKsi)
                    minKsi = ksi;

                sim.RefreshMatrix(NextMatrix(sim.matrix.Array, sim.matrix.Row));
                Skip(ref simplex);
            }
            while (!CheckOnFinish(sim));
            return minKsi;
        }

    }
}


using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Matrix
{
    public class Matrix
    {
        static Random rnd = new Random();

        public Decimal[,] array;
        int row, column;

        public Decimal[,] Array
        {
            get { return array; }
            set { array = value; }
        }

        public int Row { get { return row; } }

        public int Column { get { return column; } }

        public Matrix(int row, int column, Decimal [,] array)
        {
            this.row = row;
            this.column = column;
            this.array = new Decimal[row, column];
            for(int i = 0; i<row; i++)
                for(int j = 0; j<column; j++)
                {
                    this.array[i, j] = array[i, j];
                }
        }
        public Matrix(int row, int column)
        {
            this.row = row;
            this.column = column;
            array = new Decimal[row, column];
        }
        public Matrix(Matrix matrix)
        {
            this.column = matrix.Column;
            this.row = matrix.Row;
            this.array = new Decimal[row, column];
            for (int i = 0; i < row; i++)
                for (int j = 0; j < column; j++)
                {
```

```
                this.array[i, j] = matrix.array[i, j];
            }
    }

    public void Random()
    {
        for (int i = 0; i < row; i++)
        {
            for (int j = 0; j < column; j++)
            {
                array[i, j] = rnd.Next(10);
            }
        }
    }

    public void Random(int min, int max)
    {
        for (int i = 0; i < row; i++)
        {
            for (int j = 0; j < column; j++)
            {
                array[i, j] = rnd.Next(min, max);
            }
        }
    }

    public Matrix Transpose()
    {
        Matrix m = new Matrix(column, row);

        for (int i = 0; i < row; i++)
        {
            for (int j = 0; j < column; j++)
            {
                m.array[j, i] = array[i, j];
            }
        }

        return m;
    }

    public void TransposeMyself()
    {
        array = Transpose().array;
    }

    public Matrix Inverse()
    {
```

```
    Decimal det = Determinant();
    if (det == 0)
    {
        throw new Exception("Ћатрица вырождена");
    }

    Matrix m = new Matrix(row, column);

    for (int i = 0; i < row; i++)
    {
        for (int j = 0; j < column; j++)
        {
            m.array[i, j] = Cofactor(array, i, j) / det;
        }
    }

    return m.Transpose();
}

public Decimal Determinant()
{
    if (column != row)
    {
        throw new Exception("-асчет определител невозможен");
    }
    return Determinant(array);
}

private Decimal Determinant(Decimal[,] array)
{
    int n = (int)Math.Sqrt(array.Length);

    if (n == 1)
    {
        return array[0, 0];
    }

    Decimal det = 0;

    for (int k = 0; k < n; k++)
    {
        det += array[0, k] * Cofactor(array, 0, k);
    }

    return det;
}

private Decimal Cofactor(Decimal[,] array, int row, int column)
```

```
{
    return Convert.ToDecimal(Math.Pow(-1, column + row)) *
                        Determinant(Minor(array, row, column));
}

private Decimal[,] Minor(Decimal[,] array, int row, int column)
{
    int n = (int)Math.Sqrt(array.Length);
    Decimal[,] minor = new Decimal[n - 1, n - 1];

    int _i = 0;
    for (int i = 0; i < n; i++)
    {
        if (i == row)
        {
            continue;
        }
        int _j = 0;
        for (int j = 0; j < n; j++)
        {
            if (j == column)
            {
                continue;
            }
            minor[_i, _j] = array[i, j];
            _j++;
        }
        _i++;
    }
    return minor;
}

public Decimal[] GetRow(int IndexRow)
{
    Decimal[] resault = new Decimal[column];
    for(int i = 0; i < column; i++)
    {
        resault[i] = array[IndexRow, i];
    }
    return resault;
}

public bool SetRow(int IndexRow, Decimal[] Row)
{
    if (column != Row.Length || IndexRow < 0 || IndexRow > row - 1)
        return false;
    for(int i = 0; i < column; i++)
    {
```

```
            array[IndexRow, i] = Row[i];
        }
        return true;
    }

    public static Matrix operator +(Matrix m1, Matrix m2)
    {
        if (m1.row != m2.row || m1.column != m2.column)
        {
            throw new Exception("-ложение невозможно");
        }

        Matrix m = new Matrix(m1.row, m1.column);

        for (int i = 0; i < m1.row; i++)
        {
            for (int j = 0; j < m1.column; j++)
            {
                m.array[i, j] = m1.array[i, j] + m2.array[i, j];
            }
        }

        return m;
    }

    public static Matrix operator -(Matrix m1, Matrix m2)
    {
        if (m1.row != m2.row || m1.column != m2.column)
        {
            throw new Exception("ычитание невозможно");
        }

        Matrix m = new Matrix(m1.row, m1.column);

        for (int i = 0; i < m1.row; i++)
        {
            for (int j = 0; j < m1.column; j++)
            {
                m.array[i, j] = m1.array[i, j] - m2.array[i, j];
            }
        }

        return m;
    }

    public static Matrix operator *(Matrix m1, Matrix m2)
    {
        if (m1.column != m2.row)
```

```
            {
                throw new Exception("”’множение невозможно");
            }

            Matrix m = new Matrix(m1.row, m2.column);

            for (int i = 0; i < m1.row; i++)
            {
                for (int j = 0; j < m2.column; j++)
                {
                    decimal sum = 0;

                    for (int k = 0; k < m1.column; k++)
                    {
                        sum += m1.array[i, k] * m2.array[k, j];
                    }

                    m.array[i, j] = sum;
                }
            }

            return m;
        }

        public override string ToString()
        {
            string str = "";

            for (int i = 0; i < row; i++)
            {
                for (int j = 0; j < column; j++)
                {
                    str += array[i, j] + "\t";
                }
                str += "\n";
            }

            return str;
        }
    }
}
```