Министерство Образования и Науки РФ

Новосибирский Государственный Технический Университет

Кафедра прикладной математики и информатики

Курсовой проект по курсу   
«Численные методы»

Вариант №3

Факультет: ПМИ  
Группа: ПМ-01  
Студентка: Бебишева О.М.

Преподаватель: Персова М.Г.

Новосибирск,

2012г.

1. **Задание.**

МКЭ для одномерной краевой задачи для эллиптического уравнения в сферической системе координат. Базисные функции линейные и кубические (лагранжевы). Краевые условия всех типов. Коэффициент диффузии разложить по линейным базисным функциям. Матрицу СЛАУ генерировать в профильном формате. Для решения СЛАУ использовать LLт–разложение.

1. **Постановка задачи.**

Пусть дана эллиптическая краевая задача.

Уравнение задано в некоторой области  с границей . Пусть на границе  заданы

краевые условия:

 -краевое условие первого рода;

 -краевое условие второго рода;

 -краевое условие третьего рода.

В сферических координатах под одномерной задачей понимают случай, когда решение u краевой задачи зависит только от координаты r. Дифференциальное уравнение эллиптического типа  принимает вид: .

1. **Вариационная постановка.**

Переходим от дифференциального уравнения к вариационному. Потребуем, чтобы  - уравнение Галеркина, где функция  - подпространству пробных функций  пространства . Пространство  состоит из функций, обладающих вместе со своими первыми производными конечной энергией на (и функция, и ее первая производная должны быть интегрированы).

.

К первому слагаемому применяется формула Грина:

.

S-вся граница:

а) интеграла поне будет, так как ;

б) по :, ;

в) по :.

Вариационное уравнение:.



Представим решение уравнения Галеркина - функцию u в виде линейной комбинации базисных функций с весами:

 - - базисные функции.

Далее подставляем  в вариационное уравнение выше:

.

Ищем q:

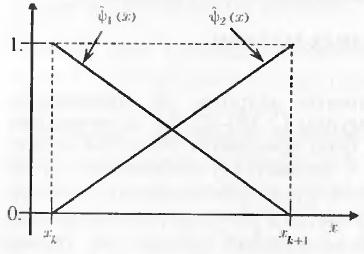


Таким образом, мы получили СЛАУ для определения неизвестных .

1. **Построение базисных функций**

*4.1) Кусочно-линейные функции.*

Разобьем расчетную область Ω=(a,b) на конечные элементы . Возьмем в качестве базисных функций финитные функции, то есть такие функции, каждая из которых отлична от нуля лишь на нескольких конечных элементах расчетной области . Они имеют следующий аналитический вид

и 

*Локальная матрица и локальный вектор правой части.*

Для определения неизвестных  в разложении  необходимо найти матрицы жесткости и массы и вектор правой части . Компоненты этих матриц вычисляются:



Поскольку в качестве базисных функций  выбираются финитные функции, то среди элементов матриц G и C будет много нулевых элементов. Ненулевой вклад в эти матрицы будут давать те базисные функции, для которых их конечные элементы пересекаются.

Перейдем к поэлементной сборки глобальной матрицы  и глобального вектора правой части . Для этого с учетом  перепишем в виде: где  - множество номеров тех функций, которые в узлах на границе  принимают значение 0.

Для каждого конечного элемента задается локальная нумерация базисных функций, вычисляются локальные матрицы и вектор правой части. Затем по соответствию локальных номеров матрицы и вектор f заносятся в глобальные объекты.

Сделав переход в интегралах  к сферической системе координат, разложив коэффициент диффузии по линейным базисам , где и значения в узлах и , получим:

Матрица жесткости:



Матрица масс:



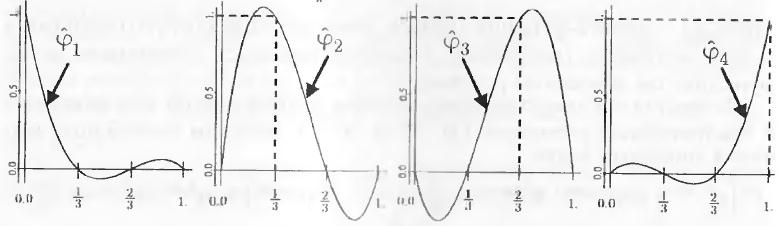
Функцию  будем заменять интерполянтом  (где и – значения  в узлах  и  конечного элемента). Тогда , где .

*4.2) Кусочно-кубические функции.*

Каждый конечный элемент с кусочно-кубическими лагранжевыми базисными функциями должен иметь по четыре узла – два граничных и два внутренних. Поэтому область Ω=(a,b) разобьем на конечные элементы ,k=0,..n-1. Шаблонные базисные функции, являющиеся базисными полиномами Лагранжа имеют вид:



, где , длина элемента.



*Локальная матрица и локальный вектор правой части.*

Элементы локальной матрицы жёсткости, масс и вектора правой части конечного элементаопределяются по тем же формулам:



Вычислив интегралы, получим:

Матрица жесткости

**G[0][0]** = 1/h\*(11004.0\*x[i]\*x[i]\*lambda1+2056.0\*x[i]\*h\*lambda1+

800.0\*x[i]\*h\*lambda2+1428.0\*x[i]\*x[i]\*lambda2+181.0\*h\*h\*lambda1+219.0\*h\*h\*lambda2)/3360.0;

**G[0][1]** = -3.0/1120.0/h\*(1526.0\*x[i]\*x[i]\*lambda1+228.0\*x[i]\*h\*

lambda1+248.0\*x[i]\*h\*lambda2+238.0\*x[i]\*x[i]\*lambda2+25.0\*h\*h\*lambda1+99.0\*h\*h\*lambda2);

**G[0][2]** = -3.0/1120.0/h\*(-364.0\*x[i]\*x[i]\*lambda1+24.0\*x[i]\*h\*

lambda1+h\*h\*lambda1-140.0\*x[i]\*x[i]\*lambda2-304.0\*x[i]\*h\*lambda2-153.0\*h\*h\*lambda2);

**G[0][3]** = 1/h\*(-546.0\*x[i]\*x[i]\*lambda1+212.0\*x[i]\*h\*lambda1+53.0\*h

\*h\*lambda1-546.0\*x[i]\*x[i]\*lambda2-1304.0\*x[i]\*h\*lambda2-705.0\*h\*h\*lambda2)/3360.0;

**G[1][0]** = -3.0/1120.0/h\*(1526.0\*x[i]\*x[i]\*lambda1+228.0\*x[i]\*h\*

lambda1+248.0\*x[i]\*h\*lambda2+238.0\*x[i]\*x[i]\*lambda2+25.0\*h\*h\*lambda1+99.0\*h\*h\*lambda2);

**G[1][1]** = 27.0/1120.0/h\*(308.0\*x[i]\*x[i]\*lambda1+104.0\*x[i]\*h\*

lambda1+23.0\*h\*h\*lambda1+140.0\*x[i]\*x[i]\*lambda2+176.0\*x[i]\*h\*lambda2+65.0\*h\*h\*lambda2);

**G[1][2]** = -27.0/1120.0/h\*(154.0\*x[i]\*x[i]\*lambda1+76.0\*x[i]\*h\*

lambda1+19.0\*h\*h\*lambda1+154.0\*x[i]\*x[i]\*lambda2+232.0\*x[i]\*h\*lambda2+97.0\*h\*h\*lambda2);

**G[1][3]** = -3.0/1120.0/h\*(-140.0\*x[i]\*x[i]\*lambda1+24.0\*x[i]\*h\*

lambda1+11.0\*h\*h\*lambda1-364.0\*x[i]\*x[i]\*lambda2-752.0\*x[i]\*h\*lambda2-387.0\*h\*h\*lambda2);

**G[2][0]** = -3.0/1120.0/h\*(-364.0\*x[i]\*x[i]\*lambda1+24.0\*x[i]\*h\*

lambda1+h\*h\*lambda1-140.0\*x[i]\*x[i]\*lambda2-304.0\*x[i]\*h\*lambda2-153.0\*h\*h\*lambda2);

**G[2][1]** = -27.0/1120.0/h\*(154.0\*x[i]\*x[i]\*lambda1+76.0\*x[i]\*h\*

lambda1+19.0\*h\*h\*lambda1+154.0\*x[i]\*x[i]\*lambda2+232.0\*x[i]\*h\*lambda2+97.0\*h\*h\*lambda2);

**G[2][2]** = 27.0/1120.0/h\*(140.0\*x[i]\*x[i]\*lambda1+104.0\*x[i]\*h\*

lambda1+29.0\*h\*h\*lambda1+308.0\*x[i]\*x[i]\*lambda2+512.0\*x[i]\*h\*lambda2+227.0\*h\*h

\*lambda2);

**G[2][3**] = -3.0/1120.0/h\*(238.0\*x[i]\*x[i]\*lambda1+228.0\*x[i]\*h\*

lambda1+89.0\*h\*h\*lambda1+1526.0\*x[i]\*x[i]\*lambda2+2824.0\*x[i]\*h\*lambda2+1323.0\*h\*h\*lambda2);

**G[3][0]** = 1/h\*(-546.0\*x[i]\*x[i]\*lambda1+212.0\*x[i]\*h\*lambda1+53.0\*h

\*h\*lambda1-546.0\*x[i]\*x[i]\*lambda2-1304.0\*x[i]\*h\*lambda2-705.0\*h\*h\*lambda2)/3360.0;

**G[3][1]** = -3.0/1120.0/h\*(-140.0\*x[i]\*x[i]\*lambda1+24.0\*x[i]\*h\*

lambda1+11.0\*h\*h\*lambda1-364.0\*x[i]\*x[i]\*lambda2-752.0\*x[i]\*h\*lambda2-387.0\*h\*h\*lambda2);

**G[3][2]** = -3.0/1120.0/h\*(238.0\*x[i]\*x[i]\*lambda1+228.0\*x[i]\*h\*

lambda1+89.0\*h\*h\*lambda1+1526.0\*x[i]\*x[i]\*lambda2+2824.0\*x[i]\*h\*lambda2+1323.0\*h\*h\*lambda2);

**G[3][3]** = 1/h\*(1428.0\*x[i]\*x[i]\*lambda1+2056.0\*x[i]\*h\*lambda1+847.0

\*h\*h\*lambda1+11004.0\*x[i]\*x[i]\*lambda2+19952.0\*x[i]\*h\*lambda2+9129.0\*h\*h\*lambda2)/3360.0;

Матрица масс

С[0][0] = h\*(2.0\*h\*h+19.0\*x[i]\*h+128.0\*x[i]\*x[i])/1680.0;

С[0][1] = -3.0/1120.0\*h\*(h\*h-2.0\*x[i]\*h-22.0\*x[i]\*x[i]);

С[0][2] = 3.0/560.0\*h\*(h\*h+x[i]\*h-4.0\*x[i]\*x[i]);

С[0][3] = h\*(15.0\*h\*h+38.0\*x[i]\*h+38.0\*x[i]\*x[i])/3360.0;

С[1][0] = -3.0/1120.0\*h\*(h\*h-2.0\*x[i]\*h-22.0\*x[i]\*x[i]);

С[1][1] = 27.0/560.0\*h\*(h\*h+5.0\*x[i]\*h+8.0\*x[i]\*x[i]);

С[1][2] = -27.0/1120.0\*h\*(h\*h+2.0\*x[i]\*h+2.0\*x[i]\*x[i]);

С[1][3] = -3.0/560.0\*h\*(4.0\*h\*h+9.0\*x[i]\*h+4.0\*x[i]\*x[i]);

С[2][0] = 3.0/560.0\*h\*(h\*h+x[i]\*h-4.0\*x[i]\*x[i]);

С[2][1] = -27.0/1120.0\*h\*(h\*h+2.0\*x[i]\*h+2.0\*x[i]\*x[i]);

С[2][2] = 27.0/560.0\*h\*(4.0\*h\*h+11.0\*x[i]\*h+8.0\*x[i]\*x[i]);

С[2][3] = 3.0/1120.0\*h\*(19.0\*h\*h+22.0\*x[i]\*x[i]+42.0\*x[i]\*h);

С[3][0] = h\*(15.0\*h\*h+38.0\*x[i]\*h+38.0\*x[i]\*x[i])/3360.0;

С[3][1] = -3.0/560.0\*h\*(4.0\*h\*h+9.0\*x[i]\*h+4.0\*x[i]\*x[i]);

С[3][2] = 3.0/1120.0\*h\*(19.0\*h\*h+22.0\*x[i]\*x[i]+42.0\*x[i]\*h);

С[3][3] = h\*(111.0\*h\*h+128.0\*x[i]\*x[i]+237.0\*x[i]\*h)/1680.0;

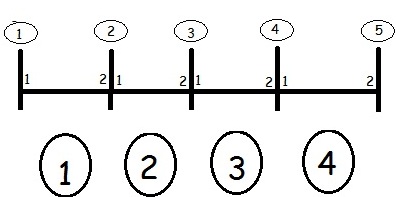
Функцию  будем заменять интерполянтом . Тогда вектор правой части , где .

1. **Структура программы**
2. Ввод данных: параметров расчетной области и краевых условий;
3. Ввод параметров сетки;
4. Генерация сетки и конечных элементов;
5. Генерация портрета матрицы в профильном формате;
6. Подсчет матриц масс и жесткости. Сборка глобальной матрицы и вектора правой части;
7. Учет краевых условий;
8. Решение СЛАУ;
9. Результаты.

**Расчетная область:**

Она записана в файле input/uzli.txt в следующем формате:

x1,x2,…,xk+1, где k- кол–во конечных элементов. ( double \*setka - массив координат узлов сетки )

Так как одномерный случай, конечные элементы представляются в виде:

**Краевые условия:**

В файле input/dannie.txt в следующем формате:

Кол-во узлов

Тип 1го условия

Тип 2го условия

Если 1 – краевое условие первого рода (условие Дирихле)

Если 2 – краевое условие второго рода (условие Неймана)

Если 3 – краевое условие третьего рода (условие Ньютона)

**Генерация сетки**

Сетка генерируется в соответствии с параметрами, описанными выше.

**Генерация портрета**

Профиль матрицы создавался с учетом специфики одномерных задач, т.е. относительной цикличности количества ненулевых элементов в профиле.

ig - массив под указатели на строки

di - массив диагональных элементов

gg - массив элементов нижнего (верхнего треугольника)

Структура матрицы: Линейные элементы: 

Лагранжевы элементы: 

\* - ненулевые элементы

**Сборка глобальной матрицы и вектора правой части**

Локальные матрицы и векторы правой части вычислялись по формулам п. 4. После заносились в глобальную СЛАУ по соответствию локальной и глобальной нумерации.

**Учет краевых условий**

1. Пусть в -ом узле задано первое краевое условии, равное В глобальной СЛАУ оно учитывается следующим образом: на диагональ на позицию  ставится число , а в глобальный вектор на эту же позицию ставится число 

2. Пусть в -ом узле задано второе краевое условие, равное . Идет добавка только в вектор правой части. В глобальный вектор на позицию  добавляется число . (якобиан - )

3. Пусть теперь в -ом узле задано третье краевое условие с параметрами  и . В глобальной СЛАУ оно учитывается следующим образом: на диагональ в -ую позицию добавляется число , а в глобальный вектор на эту же позицию число .

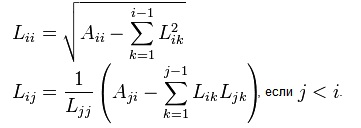
Используются функции, например:

void left\_usl\_1(); //добавляет краевые условия 1-го рода слева

void right\_ usl\_2(); //добавляет краевые условия 2-го рода справа

void right\_ usl\_3(); //добавляет краевые условия 3-го рода справа

**Решение СЛАУ**

Полученную СЛАУ решаем методом  факторизации, где элементы нижней треугольной матрицы L вычисляются по формулам:

1. **Текст программы**

#include<stdio.h>

#include<math.h>

int kol\_elem; //количество конечных элементов

int left\_usl; //тип краевого условия на левой границе

int right\_usl; //тип краевого условия на правой границе

int N; //размерность СЛАУ

int \*ig; //массив под указатели на строки

double \*di; //массив диагональных элементов

double \*gg; //массив элементов нижнего (верхнего треугольника)

double \*f; //массив векnора правой части

double \*setka; //массив координат узлов сетки

double \*q; //массив коэффициентов для аппроксимации искомой функции

double gamma = 1.; //коэффициент гамма

double beta = 1.; //коэффициент beta для третьих краевых

double lambda(double r); //коэффициент лямбда

double f\_function(double r); //возвращает значение функции правой части в заданной точке

double u\_function(double r); //возвращает аналитическое значение функции

double du\_function(double r); //производная аналитической функции

void clear\_mat(); //функция очищающая СЛАУ

void matrix\_mass\_M(); //добавляет в глобальную матрицу матрицу масс

void matrix\_zest\_G(); //добавляет в глобальную матрицу матрицу жесткости

void right\_vector\_F(); //генерирует вектор правой части, массив должен быть чистым

void left\_usl\_1(); //добавляет левые краевые условия 1-го рода

void right\_usl\_1(); //добавляет правые краевые условия 1-го рода

void left\_usl\_2(); //добавляет левые краевые условия 2-го рода

void right\_usl\_2(); //добавляет правые краевые условия 2-го рода

void left\_usl\_3(); //добавляет левые краевые условия 3-го рода

void right\_usl\_3(); //добавляет левые краевые условия 3-го рода

int gen\_mat(); //функция выделяющая память под СЛАУ и задающая портрет

int razlo\_LLt(); //разложение razlo\_LLt (метод квадратного корня)

int gauss(); //функция решающая СЛАУ

int output(); //вывод результата в файл

double lambda(double r) //коэффициент лямбда

{

return r;

}

double f\_function(double r) //возвращает значение функции правой части в заданной точке

{

return (2+r);

}

double u\_function(double r) //возвращает аналитическое значение функции

{

return r;

}

double du\_function(double r) //возвращает производную аналитическое значение функции

{

return 1;

}

void left\_usl\_1() //добавляет левые краевые условия 1-го рода

{

di[0] = 1e+60;

f[0] = 1e+60 \* u\_function(setka[0]);

return;

}

void right\_usl\_1() //добавляет правые краевые условия 1-го рода

{

di[N-1] = 1e+60;

f[N-1] = 1e+60 \* u\_function(setka[kol\_elem]);

return;

}

void left\_usl\_2() //добавляет левые краевые условия 2-го рода

{

f[0] += -lambda(setka[0])\*du\_function(setka[0])\*setka[0]\*setka[0];

return;

}

void right\_usl\_2() //добавляет правые краевые условия 2-го рода

{

f[N-1] += lambda(setka[kol\_elem])\*du\_function(setka[kol\_elem])\*setka[kol\_elem]\*setka[kol\_elem];

}

void left\_usl\_3() //добавляет правые краевые условия 3-го рода

{

di[0] += beta;

f[0] += setka[0]\*setka[0]\*beta\*u\_function(setka[0]);

return;

}

void right\_usl\_3() //добавляет левые краевые условия 3-го рода

{

di[N-1] += beta;

f[N-1] += setka[kol\_elem]\*setka[kol\_elem]\*beta\*u\_function(setka[kol\_elem]);;

return;

}

int gen\_mat() //функция выделяющая память под СЛАУ и задающая портрет

{ //и считывающая сетку из файла

FILE \*in;

int i,j;

if ((in=fopen("input\\dannie.txt","r"))==NULL)

{

printf("OSHIBKA OTKRbITIYA FAILA dannie.txt\n");

return 1;

}

if (fscanf(in,"%i",&kol\_elem)==EOF)

{

printf("NET DANNbIX V dannie.txt\n");

return 1;

}

if (kol\_elem<1)

{

printf("MALO KONECHNbIX ELEMENTOV\n");

return 1;

}

N = kol\_elem+1; //вычисляем размерность СЛАУ

ig = new int[N+1]; //выделяем память под массив указателей на строки

if (ig == NULL)

{

printf("NEDOSTATOCHNO PAMYATI\n");

return 1;

}

//заполняем массив указателей на строки

ig[0] = 1;

ig[1] = 1;

for (i=0,j=2;i<kol\_elem;i++)

{

ig[j] = ig[j-1] + 1; j++;

}

//выделяем память под матрицу СЛАУ, вектор правой части и вектор решения

di = new double[N];

f =new double[N];

gg = new double[ig[N]-1];

q =new double[N];

if (di==NULL || gg==NULL || f==NULL || q==NULL)

{

printf("NEDOSTATOCHNO PAMYATI\n");

return 1;

}

//считываем типы краевых условий

if (fscanf (in,"%i",&left\_usl)==EOF)

{

printf ("NET LEVOGO KY V dannie.txt\n");

return 1;

}

if (left\_usl<1||left\_usl>3)

{

printf ("NEVERNOE LEVOE KY V dannie.txt\n");

return 1;

}

if (fscanf (in,"%i",&right\_usl)==EOF)

{

printf ("NET PRAVOGO KY V dannie.txt\n");

return 1;

}

if (right\_usl<1||right\_usl>3)

{

printf ("NEVERNOE PRAVOGO KY V dannie.txt\n");

return 1;

}

fclose(in);

//открытие файла

if ((in=fopen("input\\uzli.txt","r"))==NULL)

{

printf("OSHIBKA OTKRbITIYA FAILA uzli.txt\n");

return 1;

}

//выделение памяти

setka =new double[kol\_elem+1];

if (setka==NULL)

{

printf("NEDOSTATOCHNO PAMYATI\n");

return 1;

}

//считывание сетки из файла

for(i=0;i<=kol\_elem;i++)

{

if (fscanf(in,"%lf ",&setka[i])==EOF)

{

printf("MALO DANNbIX V uzli.txt\n");

return 1;

}

if (i!=0 && setka[i-1]>=setka[i])

{

printf ("NEVERNbIIDANNbIE V uzli.txt\n");

return 1;

}

}

fclose(in);

return 0;

}

void clear\_mat() //функция очищающая СЛАУ

{

int i;

//очистка вектора правой части и диагонали матрицы

for (i=0;i<N;i++)

{

f[i] = 0;

di[i] = 0;

}

//очистка нижнего треугольника

for (i=0;i<ig[N]-1;i++)

gg[i] = 0;

return;

}

int razlo\_LLt() //разложение razlo\_LLt (метод квадратного корня)

{

int i,j,k;

double per;

int a;

int b;

di[0] = sqrt(di[0]);

for (i=1;i<N;i++) //пробегаем по строкам матрицы

{

a = i - ig[i+1] + ig[i]; //начало i-й строки в абсолютной нумерации

for (j=0;j<ig[i+1]-ig[i];j++) //пробегаем по столбцам до диагонали

{

b = a + j - ig[a+j+1] + ig[a+j]; //начало j-й строки в абсолютной нумерации

per = gg[ig[i]+j-1];

if (a<b) //если i-я строка началась раньше j-й строки

{

for (k=ig[a+j+1]-ig[a+j]-1;k>=0;k--)

{

per -= gg[ig[a+j]+k-1] \* gg[ig[i]+b-a+k-1];

}

}

else //если j-я строка началась раньше i-й строки

{

for (k=a-b;k<ig[a+j+1]-ig[a+j];k++)

{

per -= gg[ig[i]+k-1-(a-b)] \* gg[ig[a+j]+k-1];

}

}

gg[ig[i]+j-1] = per/di[a+j];

}

//обработка диагонального элемента

per = di[i];

for (k=0;k<ig[i+1]-ig[i];k++)

per -= gg[ig[i]+k-1] \* gg[ig[i]+k-1];

di[i] = sqrt(per);

if (di[i]==0)

return 1;

}

return 0;

}

int gauss() //функция решающая СЛАУ

{

int i,j;

//прямой ход метода Гаусса

q[0] = f[0] / di[0];

for (i=1;i<N;i++)

{

q[i] = f[i];

for (j=0;j<ig[i+1]-ig[i];j++)

{

q[i] -= gg[ig[i]+j-1] \* q[i-ig[i+1]+ig[i]+j];

}

q[i] = q[i]/di[i];

}

//Заносим в вектор правой части решение обратного хода метода Гаусса

for (i=0;i<N;i++)

f[i] = q[i];

//обратный ход метода Гаусса

for (i=N-1;i>=0;i--)

{

q[i] = f[i]/di[i];

for (j=0;j<ig[i+1]-ig[i];j++)

{

f[i-ig[i+1]+ig[i]+j] -= q[i] \* gg[ig[i]+j-1];

}

}

return 0;

}

int output() //вывод результата в файл

{

int i;

FILE \*out;

if ((out=fopen("output\\output.txt","wt"))==NULL)

{

printf("OSHIBKA OTKRbITIYA FAILA output.txt\n");

return 1;

}

double sum1=0,sum2=0;

for (i=0;i<=kol\_elem;i++) {sum1 +=pow(q[i]-u\_function(setka[i]),2); sum2 += pow(u\_function(setka[i]),2);}

for (i=0;i<=kol\_elem;i++)

fprintf (out,"%2.4lf;%2.18le;%2.16le;%2.16le\n",setka[i],q[i],u\_function(setka[i]),abs(q[i]-u\_function(setka[i])));

fprintf(out,"%.2le",sqrt(sum1)/sqrt(sum2));

fclose(out);

return 0;

}

void main()

{

if (gen\_mat())return; //генерируем портрет матрицы и выделяем память под СЛАУ

//и считываем из файла узлы сетки

clear\_mat(); //очищаем матрицу

matrix\_mass\_M(); //добавляем в глобальную матрицу матрицу масс

matrix\_zest\_G(); //добавляем в глобыльную матрицу матрицу жесткости

right\_vector\_F(); //генерирует вектор правой части

if (left\_usl==1) //добавляем левое краевое условие

left\_usl\_1();

if (left\_usl==2)

left\_usl\_2();

if (left\_usl==3)

left\_usl\_3();

if (right\_usl==1) //добавляем правое краевое условие

right\_usl\_1();

if (right\_usl==2)

right\_usl\_2();

if (right\_usl==3)

right\_usl\_3();

if (razlo\_LLt()) //razlo\_LLt-разложение для матрицы СЛАУ

printf ("Error razlo\_LLt\n");

return;

}

if (gauss()) return; //прямой и обратный ход Гаусса, решение СЛАУ

output(); return; }

**Линейные элементы:**

void matrix\_mass\_M() //добавляет в глобальную матрицу матрицу масс

{

int i;

double h;

for (i=0;i<kol\_elem;i++)

{

//вычисляем шаг

h = setka[i+1]-setka[i];

//вычисляем матрицу массы

di[i]+= gamma\*(h/60.)\*(setka[i]\*setka[i]\*20+h\*setka[i]\*10+h\*h\*2);

di[i+1]+= gamma\*(h/60.)\*(setka[i]\*setka[i]\*20+h\*setka[i]\*30+h\*h\*12);

gg[ig[i+2]-2] += gamma\*(h/60.)\*(setka[i]\*setka[i]\*10+h\*setka[i]\*10+h\*h\*3);

}

return;

}

void right\_vector\_F() //генерирует вектор правой части, массив должен быть чистым

{

int i;

double h,a,c;

double c11,c22,c21;

for (i=0;i<kol\_elem;i++)

{

//вычисляем шаг

h = setka[i+1]-setka[i];

//вычисляем вектор правой части

a = f\_function(setka[i]);

c = f\_function(setka[i+1]);

c11 = (h/60.)\*(setka[i]\*setka[i]\*20+h\*setka[i]\*10+h\*h\*2);

c22 = (h/60.)\*(setka[i]\*setka[i]\*20+h\*setka[i]\*30+h\*h\*12);

c21 = (h/60.)\*(setka[i]\*setka[i]\*10+h\*setka[i]\*10+h\*h\*3);

f[i] += c11\*a + c21\*c;

f[i+1] += c21\*a + c22\*c;

}

return;

}

void matrix\_zest\_G() //добавляет в глобальную матрицу матрицу жесткости

{

int i;

double h;

double lambda1, lambda2;

for (i=0;i<kol\_elem;i++)

{

//вычисляем шаг

h = setka[i+1]-setka[i];

lambda1=lambda(setka[i]);

lambda2=lambda(setka[i+1]);

//вычисляем коэффициенты для разложения коэффициента диффузии

//вычисляем матрицу жескости

di[i] += 1/h\*((6.0\*setka[i]\*setka[i]+4.0\*setka[i]\*h+h\*h)\*lambda1+(6.0\*setka[i]\*setka[i]+8.0\*setka[i]\*h+3.0\*h\*h)\*lambda2)/12.0;

di[i+1] += 1/h\*((6.0\*setka[i]\*setka[i]+4.0\*setka[i]\*h+h\*h)\*lambda1+(6.0\*setka[i]\*setka[i]+8.0\*setka[i]\*h+3.0\*h\*h)\*lambda2)/12.0;

gg[ig[i+2]-2] += -1/h\*((6.0\*setka[i]\*setka[i]+4.0\*setka[i]\*h+h\*h)\*lambda1+(6.0\*setka[i]\*setka[i]+8.0\*setka[i]\*h+3.0\*h\*h)\*lambda2)/12.0;

}

return;

}

**Лагранжевы элементы:**

void matrix\_mass\_M() //добавляет в глобальную матрицу матрицу масс

{

int i;

double h;

for (i=0;i<kol\_elem;i++)

{

//вычисляем шаг

h = setka[i+1]-setka[i];

//вычисляем матрицу массы

di[3\*i] += h\*(19.0\*setka[i]\*h+128.0\*setka[i]\*setka[i]+2.0\*h\*h)/1680.0\*gamma;

di[3\*i+1] += 27.0/560.0\*h\*(5.0\*setka[i]\*h+8.0\*setka[i]\*setka[i]+h\*h)\*gamma;

di[3\*i+2] += 27.0/560.0\*h\*(11.0\*setka[i]\*h+8.0\*setka[i]\*setka[i]+4.0\*h\*h)\*gamma;

di[3\*i+3] += h\*(237.0\*setka[i]\*h+128.0\*setka[i]\*setka[i]+111.0\*h\*h)/1680.0\*gamma;

gg[ig[3\*i+2]-2] += -3.0/1120.0\*h\*(-2.0\*setka[i]\*h-22.0\*setka[i]\*setka[i]+h\*h)\*gamma;

gg[ig[3\*i+2]-1] += 3.0/560.0\*h\*(setka[i]\*h-4.0\*setka[i]\*setka[i]+h\*h)\*gamma;

gg[ig[3\*i+2]] += -27.0/1120.0\*h\*(2.0\*setka[i]\*h+2.0\*setka[i]\*setka[i]+h\*h)\*gamma;

gg[ig[3\*i+3]-1] += h\*(38.0\*setka[i]\*h+38.0\*setka[i]\*setka[i]+15.0\*h\*h)/3360.0\*gamma;

gg[ig[3\*i+3]] += -3.0/560.0\*h\*(9.0\*setka[i]\*h+4.0\*setka[i]\*setka[i]+4.0\*h\*h)\*gamma;

gg[ig[3\*i+3]+1] += 3.0/1120.0\*h\*(42.0\*setka[i]\*h+22.0\*setka[i]\*setka[i]+19.0\*h\*h)\*gamma;

}

return;

}

void right\_vector\_F()//генерирует вектор правой части, массив должен быть чистым

{

int i;

double h,a,b,c,d;

double c11,c22,c33,c44,c21,c31,c32,c41,c42,c43;

for (i=0;i<kol\_elem;i++)

{

//вычисляем шаг

h = setka[i+1]-setka[i];

//вычисляем вектор правой части

a = f\_function(setka[i]);

b = f\_function(setka[i]+h/3.);

c = f\_function(setka[i]+2\*h/3.);

d = f\_function(setka[i+1]);

c11 = h\*(19.0\*setka[i]\*h+128.0\*setka[i]\*setka[i]+2.0\*h\*h)/1680.0;

c22 = 27.0/560.0\*h\*(5.0\*setka[i]\*h+8.0\*setka[i]\*setka[i]+h\*h);

c33 = 27.0/560.0\*h\*(11.0\*setka[i]\*h+8.0\*setka[i]\*setka[i]+4.0\*h\*h);

c44 = h\*(237.0\*setka[i]\*h+128.0\*setka[i]\*setka[i]+111.0\*h\*h)/1680.0;

c21 = -3.0/1120.0\*h\*(-2.0\*setka[i]\*h-22.0\*setka[i]\*setka[i]+h\*h);

c31 = 3.0/560.0\*h\*(setka[i]\*h-4.0\*setka[i]\*setka[i]+h\*h);

c32 = -27.0/1120.0\*h\*(2.0\*setka[i]\*h+2.0\*setka[i]\*setka[i]+h\*h);

c41 = h\*(38.0\*setka[i]\*h+38.0\*setka[i]\*setka[i]+15.0\*h\*h)/3360.0;

c42 = -3.0/560.0\*h\*(9.0\*setka[i]\*h+4.0\*setka[i]\*setka[i]+4.0\*h\*h);

c43 = 3.0/1120.0\*h\*(42.0\*setka[i]\*h+22.0\*setka[i]\*setka[i]+19.0\*h\*h);

f[3\*i] += c11\*a + c21\*b + c31\*c + c41\*d;

f[3\*i+1] += c21\*a + c22\*b + c32\*c + c42\*d;

f[3\*i+2] += c31\*a + c32\*b + c33\*c + c43\*d;

f[3\*i+3] += c41\*a + c42\*b + c43\*c + c44\*d;

}

return;

}

void matrix\_zest\_G() //добавляет в глобальную матрицу матрицу жесткости

{

int i;

double h;

double lambda1,lambda2;

for (i=0;i<kol\_elem;i++)

{

//вычисляем шаг

h = setka[i+1]-setka[i];

lambda1=lambda(setka[i]);

lambda2=lambda(setka[i+1]);

//вычисляем коэффициенты для разложения коэффициента диффузии

//вычисляем матрицу жескости

di[3\*i] += 1/h\*((181.0\*h\*h+11004.0\*setka[i]\*setka[i]+2056.0\*setka[i]\*h)\*lambda1+(1428.0\*setka[i]\*setka[i]+800.0\*setka[i]\*h+219.0\*h\*h)\*lambda2)/3360.0;

di[3\*i+1] += 27.0/1120.0/h\*((23.0\*h\*h+308.0\*setka[i]\*setka[i]+104.0\*setka[i]\*h)\*lambda1+(140.0\*setka[i]\*setka[i]+176.0\*setka[i]\*h+65.0\*h\*h)\*lambda2);

di[3\*i+2] += 27.0/1120.0/h\*((29.0\*h\*h+140.0\*setka[i]\*setka[i]+104.0\*setka[i]\*h)\*lambda1+(308.0\*setka[i]\*setka[i]+512.0\*setka[i]\*h+227.0\*h\*h)\*lambda2);

di[3\*i+3] += 1/h\*((847.0\*h\*h+1428.0\*setka[i]\*setka[i]+2056.0\*setka[i]\*h)\*lambda1+(11004.0\*setka[i]\*setka[i]+19952.0\*setka[i]\*h+9129.0\*h\*h)\*lambda2)/3360.0;

gg[ig[3\*i+2]-2] += -3.0/1120.0/h\*(228.0\*setka[i]\*h\*lambda1+1526.0\*setka[i]\*

setka[i]\*lambda1+25.0\*h\*h\*lambda1+99.0\*h\*h\*lambda2+248.0\*setka[i]\*h\*lambda2+238.0

\*setka[i]\*setka[i]\*lambda2);

gg[ig[3\*i+2]-1] += -3.0/1120.0/h\*(24.0\*setka[i]\*h\*lambda1-364.0\*setka[i]\*setka

[i]\*lambda1+h\*h\*lambda1-153.0\*h\*h\*lambda2-304.0\*setka[i]\*h\*lambda2-140.0\*setka[i]

\*setka[i]\*lambda2);

gg[ig[3\*i+2]] += -27.0/1120.0/h\*(76.0\*setka[i]\*h\*lambda1+154.0\*setka[i]\*setka

[i]\*lambda2+232.0\*setka[i]\*h\*lambda2+19.0\*h\*h\*lambda1+97.0\*h\*h\*lambda2+154.0\*

setka[i]\*setka[i]\*lambda1);

gg[ig[3\*i+3]-1] += 1/h\*(212.0\*setka[i]\*h\*lambda1-546.0\*setka[i]\*setka[i]\*

lambda2-1304.0\*setka[i]\*h\*lambda2+53.0\*h\*h\*lambda1-705.0\*h\*h\*lambda2-546.0\*setka

[i]\*setka[i]\*lambda1)/3360.0;

gg[ig[3\*i+3]] += -3.0/1120.0/h\*(24.0\*setka[i]\*h\*lambda1-364.0\*setka[i]\*setka

[i]\*lambda2-752.0\*setka[i]\*h\*lambda2+11.0\*h\*h\*lambda1-387.0\*h\*h\*lambda2-140.0\*

setka[i]\*setka[i]\*lambda1);

gg[ig[3\*i+3]+1] += -3.0/1120.0/h\*(228.0\*setka[i]\*h\*lambda1+1526.0\*setka[i]\*

setka[i]\*lambda2+2824.0\*setka[i]\*h\*lambda2+89.0\*h\*h\*lambda1+1323.0\*h\*h\*lambda2+

238.0\*setka[i]\*setka[i]\*lambda1);

}

return;

}

**Исследования.**

**Исследования на линейных базисных функциях.**

λ=r,:

 ,  , ,. (первые-первые)

|  |  |  |  |
| --- | --- | --- | --- |
| X (узлы) | q=u(x) (численное значение) | Uan (точное значение) | | u(x)-uan | (погрешность) |
| 0.1000  0.3000  0.5000  0.7000  1.0000 | 9.999999999999999200e-002  1.097673185078543300e+000  1.170721396992277800e+000  1.129618989516358400e+000  9.999999999999998900e-001 | 1.0000000000000001e-001  2.9999999999999999e-001  5.0000000000000000e-001  6.9999999999999996e-001  1.0000000000000000e+000 | 1.3877787807814457e-017  7.9767318507854323e-001  6.7072139699227784e-001  4.2961898951635846e-001  1.1102230246251565e-016 |

Относительная погрешность (||uan-u(x)||/||uan||):8.31e-001

,. (первые-вторые)

|  |  |  |  |
| --- | --- | --- | --- |
| х (узлы) | q=u(x) (численное значение) | Uan (точное значение) | | u(x)-uan | (погрешность) |
| 0.1000  0.3000  0.5000  0.7000  1.0000 | 9.999999999999999200e-002  4.160085515628591900e+000  4.854890684073607700e+000  5.175522960906579200e+000  5.504710535500295900e+000 | 1.0000000000000001e-001  2.9999999999999999e-001  5.0000000000000000e-001  6.9999999999999996e-001  1.0000000000000000e+000 | 1.3877787807814457e-017  3.8600855156285920e+000  4.3548906840736077e+000  4.4755229609065790e+000  4.5047105355002959e+000 |

Относительная погрешность (||u(x)-uan ||/||uan||):6.35e+000

,. (первые-третьи)

|  |  |  |  |
| --- | --- | --- | --- |
| х (узлы) | q=u(x) (численное значение) | Uan (точное значение) | | u(x)-uan | (погрешность) |
| 0.1000  0.3000  0.5000  0.7000  1.0000 | 9.999999999999999200e-002  1.353836494504330900e+000  1.478893156767712400e+000  1.468048971851266100e+000  1.376808030475668900e+000 | 1.0000000000000001e-001  2.9999999999999999e-001  5.0000000000000000e-001  6.9999999999999996e-001  1.0000000000000000e+000 | 1.3877787807814457e-017  1.0538364945043308e+000  9.7889315676771238e-001  7.6804897185126619e-001  3.7680803047566891e-001 |

Относительная погрешность (||u(x)-uan ||/||uan||): 1.23e+000

,. (вторые-вторые)

|  |  |  |  |
| --- | --- | --- | --- |
| х (узлы) | q=u(x) (численное значение) | Uan (точное значение) | | u(x)-uan | (погрешность) |
| 0.1000  0.3000  0.5000  0.7000  1.0000 | 5.100000000000015600e+000  5.300000000000015800e+000  5.500000000000015100e+000  5.700000000000014400e+000  6.000000000000014200e+000 | 1.0000000000000001e-001  2.9999999999999999e-001  5.0000000000000000e-001  6.9999999999999996e-001  1.0000000000000000e+000 | 5.0000000000000160e+000  5.0000000000000160e+000  5.0000000000000151e+000  5.0000000000000142e+000  5.0000000000000142e+000 |

Относительная погрешность (||u(x)-uan ||/||uan||): 8.24e+000

,. (вторые-третьи)

|  |  |  |  |
| --- | --- | --- | --- |
| х (узлы) | q=u(x) (численное значение) | Uan (точное значение) | | u(x)-uan | (погрешность) |
| 0.1000  0.3000  0.5000  0.7000  1.0000 | 1.640839469120895400e+000  1.627311041288851600e+000  1.584086888685585000e+000  1.526876668415508000e+000  1.414983587625961100e+000 | 1.0000000000000001e-001  2.9999999999999999e-001  5.0000000000000000e-001  6.9999999999999996e-001  1.0000000000000000e+000 | 1.5408394691208953e+000  1.3273110412888516e+000  1.0840868886855850e+000  8.2687666841550800e-001  4.1498358762596110e-001 |

Относительная погрешность (||u(x)-uan ||/||uan||):1.83e+000

,. (третьи-третьи)

|  |  |  |  |
| --- | --- | --- | --- |
| х (узлы) | q=u(x) (численное значение) | Uan (точное значение) | | u(x)-uan | (погрешность) |
| 0.1000  0.3000  0.5000  0.7000  1.0000 | 6.952604945624650500e-002  1.348427852020346100e+000  1.476812687875124000e+000  1.466885507160992800e+000  1.376053013445087700e+000 | 1.0000000000000001e-001  2.9999999999999999e-001  5.0000000000000000e-001  6.9999999999999996e-001  1.0000000000000000e+000 | 3.0473950543753500e-002  1.0484278520203461e+000  9.7681268787512399e-001  7.6688550716099280e-001  3.7605301344508768e-001 |

Относительная погрешность (||u(x)-uan ||/||uan||):1.23e+000

Дробление сетки при третьих краевых условиях:

1. 4 конечных элемента, сетка 0.0 1.0 2.0 3.0 4.0

|  |  |  |  |
| --- | --- | --- | --- |
| х (узлы) | q=u(x) (численное значение) | Uan (точное значение) | | u(x)-uan | (погрешность) |
| 0.0000  0.4000  0.8000  1.2000  1.6000 | 1.219672161025978900e-001  3.129332467601105200e+000  3.401449935457964400e+000  3.519862172886166500e+000  3.587118023939922300e+000 | 0.0000000000000000e+000  4.0000000000000002e-001  8.0000000000000004e-001  1.2000000000000000e+000  1.6000000000000001e+000 | 1.2196721610259789e-001  2.7293324676011053e+000  2.6014499354579641e+000  2.3198621728861664e+000  1.9871180239399222e+000 |

Относительная погрешность (||u(x)-uan ||/||uan||):2.22e+000

1. 8 конечных элементов, сетка 0.0 0.2 0.4 0.6 0.8 1.0 1.2 1.4 1.6

|  |  |  |  |
| --- | --- | --- | --- |
| х (узлы) | q=u(x) (численное значение) | Uan (точное значение) | | u(x)-uan | (погрешность) |
| 0.0000  0.2000  0.4000  0.6000  0.8000  1.0000  1.2000  1.4000  1.6000 | 3.022194363224387300e-002  3.033217595854329500e+000  3.278717522051963800e+000  3.378929554301086700e+000  3.448044425401562300e+000  3.503054422732950500e+000  3.548051560492629700e+000  3.584219497125867200e+000  3.611854697484623500e+000 | 0.0000000000000000e+000  2.0000000000000001e-001  4.0000000000000002e-001  5.9999999999999998e-001  8.0000000000000004e-001  1.0000000000000000e+000  1.2000000000000000e+000  1.3999999999999999e+000  1.6000000000000001e+000 | 3.0221943632243873e-002  2.8332175958543293e+000  2.8787175220519639e+000  2.7789295543010866e+000  2.6480444254015625e+000  2.5030544227329505e+000  2.3480515604926300e+000  2.1842194971258673e+000  2.0118546974846234e+000 |

Относительная погрешность (||u(x)-uan ||/||uan||):2.52e+000

**Исследования на кубических лагранжевых базисных функциях.**

**1)**

λ=r, :

 , , ,. (первые-третьи)

|  |  |  |  |
| --- | --- | --- | --- |
| х (узлы) | q=u(x) (численное значение) | Uan (точное значение) | | u(x)-uan | (погрешность) |
| 0.1000  0.3000  0.5000  0.7000  1.0000 | 1.000000000000000200e-003  3.940483866154056000e+000  4.398279129806448800e+000  4.495140188164831400e+000  4.039757119993908900e+000 | 1.0000000000000002e-003  2.7000000000000000e-002  1.2500000000000000e-001  3.4299999999999992e-001  1.0000000000000000e+000 | 0.0000000000000000e+000  3.9134838661540559e+000  4.2732791298064488e+000  4.1521401881648314e+000  3.0397571199939089e+000 |

Относительная погрешность (||u(x)-uan ||/||uan||):7.28e+000

,. (первые-вторые)

|  |  |  |  |
| --- | --- | --- | --- |
| х (узлы) | q=u(x) (численное значение) | Uan (точное значение) | | u(x)-uan | (погрешность) |
| 0.1000  0.3000  0.5000  0.7000  1.0000 | 1.000000000000000200e-003  1.767347514278819900e+001  2.009805386321467000e+001  2.149563429865154200e+001  2.284456531519633100e+001 | 1.0000000000000002e-003  2.7000000000000000e-002  1.2500000000000000e-001  3.4299999999999992e-001  1.0000000000000000e+000 | 0.0000000000000000e+000  1.7646475142788198e+001  1.9973053863214670e+001  2.1152634298651542e+001  2.1844565315196331e+001 |

Относительная погрешность (||u(x)-uan ||/||uan||):3.80e+001

,. (вторые-вторые)

|  |  |  |  |
| --- | --- | --- | --- |
| х (узлы) | q=u(x) (численное значение) | Uan (точное значение) | | u(x)-uan | (погрешность) |
| 0.1000  0.3000  0.5000  0.7000  1.0000 | 1.920923233785185100e+001  2.020764299502000900e+001  2.142554630492937300e+001  2.255671273201508600e+001  2.383239907697288400e+001 | 1.0000000000000002e-003  2.7000000000000000e-002  1.2500000000000000e-001  3.4299999999999992e-001  1.0000000000000000e+000 | 1.9208232337851850e+001  2.0180642995020008e+001  2.1300546304929373e+001  2.2213712732015086e+001  2.2832399076972884e+001 |

Относительная погрешность (||u(x)-uan ||/||uan||):4.45e+001

,. (третьи-третьи)

|  |  |  |  |
| --- | --- | --- | --- |
| х (узлы) | q=u(x) (численное значение) | Uan (точное значение) | | u(x)-uan | (погрешность) |
| 0.1000  0.3000  0.5000  0.7000  1.0000 | 8.779545372286704000e-002  3.949467577076279900e+000  4.401456924935271100e+000  4.496880456241053800e+000  4.040842249354860800e+000 | 1.0000000000000002e-003  2.7000000000000000e-002  1.2500000000000000e-001  3.4299999999999992e-001  1.0000000000000000e+000 | 8.6795453722867039e-002  3.9224675770762798e+000  4.2764569249352711e+000  4.1538804562410538e+000  3.0408422493548608e+000 |

Относительная погрешность (||u(x)-uan ||/||uan||):7.28e+000

**2)**

 , , ,. (первые-первые)

|  |  |  |  |
| --- | --- | --- | --- |
| х (узлы) | q=u(x) (численное значение) | Uan (точное значение) | | u(x)-uan | (погрешность) |
| 0.1000  0.3000  0.5000  0.7000  1.0000 | 1.000000000000000500e-004  7.329237342344657700e-001  8.535732490068681500e-001  9.487712610914784700e-001  1.000000000000000000e+000 | 1.0000000000000003e-004  8.0999999999999996e-003  6.2500000000000000e-002  2.4009999999999992e-001  1.0000000000000000e+000 | 1.3552527156068805e-020  7.2482373423446578e-001  7.9107324900686815e-001  7.0867126109147849e-001  0.0000000000000000e+000 |

Относительная погрешность (||u(x)-uan ||/||uan||): 1.25e+000

,. (первые-вторые)

|  |  |  |  |
| --- | --- | --- | --- |
| х (узлы) | q=u(x) (численное значение) | Uan (точное значение) | | u(x)-uan | (погрешность) |
| 0.1000  0.3000  0.5000  0.7000  1.0000 | 1.000000000000000200e-004  1.053456649053903700e+001  1.205896763042963600e+001  1.308252759975354700e+001  1.442154875928658700e+001 | 1.0000000000000003e-004  8.0999999999999996e-003  6.2500000000000000e-002  2.4009999999999992e-001  1.0000000000000000e+000 | 1.3552527156068805e-020  1.0526466490539036e+001  1.1996467630429636e+001  1.2842427599753547e+001  1.3421548759286587e+001 |

Относительная погрешность (||u(x)-uan ||/||uan||):2.38e+001

,. (вторые-вторые)

|  |  |  |  |
| --- | --- | --- | --- |
| х (узлы) | q=u(x) (численное значение) | Uan (точное значение) | | u(x)-uan | (погрешность) |
| 0.1000  0.3000  0.5000  0.7000  1.0000 | 1.143133222207307000e+001  1.204270428786712300e+001  1.284898693607829400e+001  1.371399819862328500e+001  1.500942987864494800e+001 | 1.0000000000000003e-004  8.0999999999999996e-003  6.2500000000000000e-002  2.4009999999999992e-001  1.0000000000000000e+000 | 1.1431232222073071e+001  1.2034604287867122e+001  1.2786486936078294e+001  1.3473898198623285e+001  1.4009429878644948e+001 |

Относительная погрешность (||u(x)-uan ||/||uan||): 2.77e+001

,. (первые-третьи)

|  |  |  |  |
| --- | --- | --- | --- |
| х (узлы) | q=u(x) (численное значение) | Uan (точное значение) | | u(x)-uan | (погрешность) |
| 0.1000  0.3000  0.5000  0.7000  1.0000 | 1.000000000000000200e-004  9.046947619964342800e-001  1.049944628502120800e+000  1.161411922328053500e+000  1.235208860581777700e+000 | 1.0000000000000003e-004  8.0999999999999996e-003  6.2500000000000000e-002  2.4009999999999992e-001  1.0000000000000000e+000 | 1.3552527156068805e-020  8.9659476199643429e-001  9.8744462850212078e-001  9.2131192232805348e-001  2.3520886058177770e-001 |

Относительная погрешность (||u(x)-uan ||/||uan||):1.59e+000

,. (вторые-третьи)

|  |  |  |  |
| --- | --- | --- | --- |
| х (узлы) | q=u(x) (численное значение) | Uan (точное значение) | | u(x)-uan | (погрешность) |
| 0.1000  0.3000  0.5000  0.7000  1.0000 | 9.488639710718386200e-001  1.002896002775489000e+000  1.084681212002637900e+000  1.180434850141936800e+000  1.247070445462887800e+000 | 1.0000000000000003e-004  8.0999999999999996e-003  6.2500000000000000e-002  2.4009999999999992e-001  1.0000000000000000e+000 | 9.4876397107183863e-001  9.9479600277548896e-001  1.0221812120026379e+000  9.4033485014193685e-001  2.4707044546288781e-001 |

Относительная погрешность (||u(x)-uan ||/||uan||):1.91e+000

,. (третьи-третьи)

|  |  |  |  |
| --- | --- | --- | --- |
| х (узлы) | q=u(x) (численное значение) | Uan (точное значение) | | u(x)-uan | (погрешность) |
| 0.1000  0.3000  0.5000  0.7000  1.0000 | 1.995730276308850200e-002  9.067500801043193400e-001  1.050671653216958900e+000  1.161810065647735300e+000  1.235457119463244000e+000 | 1.0000000000000003e-004  8.0999999999999996e-003  6.2500000000000000e-002  2.4009999999999992e-001  1.0000000000000000e+000 | 1.9857302763088502e-002  8.9865008010431935e-001  9.8817165321695888e-001  9.2171006564773528e-001  2.3545711946324399e-001 |

Относительная погрешность (||u(x)-uan ||/||uan||):1.59e+000