|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ Робототехника и комплексная автоматизация

КАФЕДРА Системы автоматизированного проектирования (РК-6)

**ОТЧЕТ О ВЫПОЛНЕНИИ ДОМАШНЕЙ РАБОТЫ**

по дисциплине: «Модели и методы анализа проектных

решений»

Студент Гусаров Аркадий Андреевич

Группа РК6-63Б

Тип задания Домашняя работа №1

Вариант лабораторной работы 2

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_Гусаров А.A.**

*подпись, дата фамилия, и.о.*

Преподаватель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_Трудоношин В.А.**

*подпись, дата фамилия, и.о.*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Москва, 2022 г.*

Оглавление

[Текст задания 3](#_Toc102125732)

[Описание структуры программы 3](#_Toc102125733)

[Используемые данные 3](#_Toc102125734)

[Блок-схема 4](#_Toc102125735)

[Пример работы программы 5](#_Toc102125736)

[Текст программы 5](#_Toc102125737)

Задание на лабораторную работу

С помощью неявной разностной схемы решить нестационарное уравнение теплопроводности для пластины, изображенной на рисунке 1, там же указаны размеры сторон.

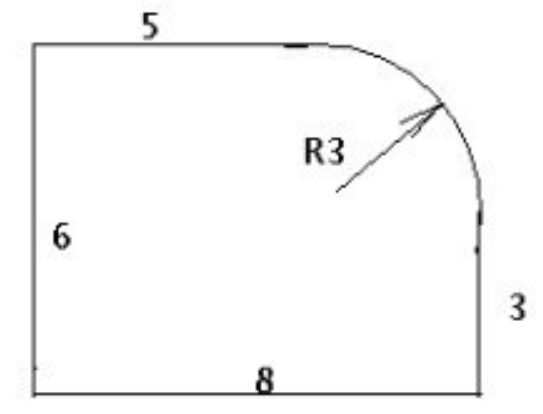


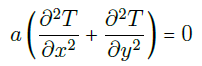
Рис. 1. Пластина

Начальное значение температуры пластины – 0 градусов. Граничные условия следующие: слева 200 градусов, внизу задано условие dT/dn = T, справа, на округлой границе и вверху 100.

Теоретическая часть

Процессы теплопередачи происходят в пространстве и времени. Поэтому исследование теплопроводности сводится к изучению пространственно-временного изменения температуры, т. е. к нахождению зависимости 𝑇=𝑇(𝑥,𝑦,𝑡), где 𝑥,𝑦 – пространственные координаты в декартовой системе, 𝑡 – время.

Состояние пластины описывается стационарным дифференциальным уравнением:

 (1)

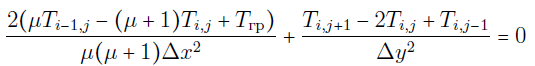
Разностный аналог уравнения 1 имеет вид:

,

где Δ𝑥 и Δ𝑦 – шаги по 𝑥 и 𝑦, соответственно.

Скругленная граница считается границей неправильной формы, в связи с этим уравнения будут иметь другой вид, будет 3 случая. Эти уравнения предназначены для точек около границы внутри пластины.

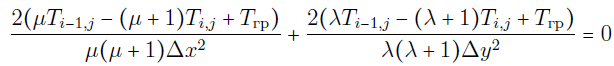
1. Следующая точка по лежит за границей, по оси 𝑦 – в пределах пластины:



1. Следующая точка по 𝑦 лежит за границей, по оси 𝑥 – в пределах пластины:



1. Следующие точки по осям x и y лежат за границей:



Коэффициенты 𝜇 и 𝜆 равны 𝑙/Δ𝑥 и 𝑙/Δ𝑦 , соответственно, 𝑙 – расстояние от точки до границы по выбранной оси, 𝑇гр –– значение температуры на скругленной границе.

Зададим граничные условия:

1. Для левой грани граничное условие 1-го рода: 𝑇*i,j* = 200.
2. Для нижней грани граничное условие 3-го рода:

.

1. Для правой грани граничное условие 1-го рода: 𝑇*i,j* = 100.
2. Для круглой грани граничное условие 1-го рода: 𝑇*i,j* = 100.

Алгоритм решения:

1. Нанесение на объект сетки или дискретизация пространства.
2. Нумерация узлов сетки.
3. Запись уравнений граничных условий для приграничных узлов и занесение коэффициентов при 𝑇 в матрицу 𝐴, а решений в вектор 𝑏.
4. Запись разностного уравнения для каждого внутреннего узла сетки и занесение коэффициентов при 𝑇 в матрицу 𝐴, а решений в вектор 𝑏.
5. Решение СЛАУ 𝐴𝑥 = 𝑏 методом Гаусса.
6. Визуализация температурного поля.

Текст программы

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include <math.h>

#define CIRCLE\_CONSTRAIN 100

#define RIGTH\_CONSTRAINT 100

#define LEFT\_CONSTRAINT 200

#define TOP\_CONSTRAINT 100

int M, N; //размерность пластины

long double \*\*A; // массив коэффициентов

long double \*b; // массив значений

int \*num; // массив номеров узлов

long double c = 1; // удельная теплоемкость

long double ro = 1; // плотность

long double lyambda = 1; // коэффициент теплопроводности

long double dx = 0.5;

long double dy = 0.5;

//заполнение матриц для точек не на границах

void make\_matrix(int id, int k) {

long double a = lyambda / c / ro;

A[k][num[id + M]] = a / dy / dy;

A[k][num[id - M]] = a / dy / dy;

A[k][num[id + 1]] = a / dx / dx;

A[k][num[id - 1]] = a / dx / dx;

A[k][num[id]] = - (2 \* a / dx / dx) - (2 \* a / dy / dy);

}

//заполнение граничных условий

void mysolver(void) {

int i, j; // столбцы, строки

int x, y;

int k = 0;

int id;

float l, m; // лямбда, мю

int r = 3/dx - 1; // радиус

long double a = lyambda / c / ro;

//Перебираем всю матрицу

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

{

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

{

id = j \* M + i; //номер текущего элемента

x = i - (5/dx);

y = j - (3/dy);

if (((x\*x + y\*y) >= r\*r) && (i >= M-r) && (j >= N-r)) //за круглой границей

{

continue;

}

else if (((x\*x + y\*y) <= r\*r) && (((x+1)\*(x+1) + y\*y) > r\*r) && ((x\*x + (y+1)\*(y+1)) < r\*r) && (i > M-r) && (j > N-r)) //около круглой границы // по х за границей

{

m = (sqrt(r\*r - y\*y) - x) / dx;

A[k][num[id + M]] = a / dy / dy;

A[k][num[id - M]] = a / dy / dy;

A[k][num[id - 1]] = 2 \* a / (m+1) / dx / dx;

A[k][num[id]] = - (2 \* a / m / dx / dx) - (2 \* a / dy / dy);

b[k] = (-2) \* a \* CIRCLE\_CONSTRAIN / m / (m+1) / dx / dx;

}

else if (((x\*x + y\*y) <= r\*r) && (((x+1)\*(x+1) + y\*y) < r\*r) && ((x\*x + (y+1)\*(y+1)) > r\*r) && (i > M-r) && (j > N-r)) //около круглой границы // по у за границей

{

l = (sqrt(r\*r - x\*x) - y) / dy;

A[k][num[id + 1]] = a / dx / dx;

A[k][num[id - 1]] = a / dx / dx;

A[k][num[id - M]] = 2 \* a / (l+1) / dy / dy;

A[k][num[id]] = - (2 \* a / dx / dx) - (2 \* a / m / dy / dy);

b[k] = (-2) \* a \* CIRCLE\_CONSTRAIN / l / (l+1) / dy / dy;

}

else if (((x\*x + y\*y) <= r\*r) && (((x+1)\*(x+1) + y\*y) >= r\*r) && ((x\*x + (y+1)\*(y+1)) >= r\*r) && (i > M-r) && (j > N-r)) //около круглой границы // обе за границей

{

m = (sqrt(r\*r - y\*y) - x) / dx;

l = (sqrt(r\*r - x\*x) - y) / dy;

A[k][num[id - M]] = 2 \* a / (l+1) / dy / dy;

A[k][num[id - 1]] = 2 \* a / (m+1) / dx / dx;

A[k][num[id]] = - (2 \* a / m / dx / dx) - (2 \* a / l / dy / dy);

b[k] = ((-2) \* a \* CIRCLE\_CONSTRAIN / m / (m+1) / dx / dx) + ((-2) \* a \* CIRCLE\_CONSTRAIN / l / (l+1) / dy / dy);

}

else if (j == 0) //нижняя граница

{

// A[k][num[id]] = 1;

// b[k] = BOTTOM\_CONSTRAINT;

A[k][num[id+1]] = 1;

A[k][num[id]] = -1 - dx;

b[k] = 0;

}

else if (j == N - 1) //верхняя граница

{

A[k][num[id]] = 1;

b[k] = TOP\_CONSTRAINT;

}

else if (i == 0) //левая граница

{

A[k][num[id]] = 1;

b[k] = LEFT\_CONSTRAINT;

}

else if (i == M - 1) //правая граница

{

A[k][num[id]] = 1;

b[k] = RIGTH\_CONSTRAINT;

}

else

{

make\_matrix(id, k);

}

k++;

}

}

}

void gauss(int n) { //решение систем с помощью метода Гаусса

int i, j, k;

long double a;

//прямой ход метода Гаусса

for (k = 0; k < n; k++)

for (j = k + 1; j < n; j++)

{

a = - A[j][k] / A[k][k];

b[j] += b[k] \* a;

for (i = k ; i < n; i++)

A[j][i] += A[k][i] \* a;

}

//Обратный ход метода Гаусса

b[n - 1] /= A[n - 1][n - 1];

for (i = n - 2; i >= 0; i--)

{

for (j = i + 1; j < n; j++)

b[i] -= b[j] \* A[i] [j];

b[i] /= A[i][i];

}

}

int main(int argc, char\* argv[])

{

int i, j, k, r, x, y;

int n; // счетчик узлов

M = 8/dx; //размерность матрицы

N = 6/dy;

num = (int\*) malloc(sizeof(long double) \* N \* M); // массив для хранения номеров узлов

n = 0; // счетчик для заполнения узлов

r = 3/dx-1; // радиус

// заполнение матрицы номеров

for (j = 0; j < N; j++) { // строки

for (i = 0; i < M; i++) { // столбцы

x = i - (5/dx); // расстояние по x

y = j - (3/dy); // расстояние по y

if (((x\*x + y\*y) >= r\*r) && (i >= M-r) && (j >= N-r))

{ // если за границей

num[j \* M + i] = -1;

}

else

{

num[j \* M + i] = n;

n++;

}

}

}

for (j = N-1; j >= 0; j--) {

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

if(num[j \* M + i] != (-1))

printf("%d\t",num[j \* M + i]);

printf("\n");

}

printf("\n\n");

A = (long double\*\*) malloc(sizeof(long double\*) \* n); //создаем массивы для коэффициентов и искомых значений

b = (long double\*) malloc(sizeof(long double) \* n);

for (k = 0; k < n; k++)

{

A[k] = (long double\*) malloc(sizeof(long double) \* n); //массив коэффициентов двумерный

}

memset(b, 0, sizeof(long double) \* n); // обнуление х

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

memset(A[i], 0, sizeof(long double) \* n); // обнуление А

mysolver();

gauss(n);

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

{

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

{

x = i - (5/dx);

y = j - (3/dy);

if (((x\*x + y\*y) >= r\*r) && (i >= M-r) && (j >= N-r))

continue;

printf("%.0f\t",(double) b[num[j \* M + i]]);

}

printf("\n");

}

printf("\n\n");

FILE \*fd = fopen("out.txt", "w");

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

{

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

{

x = i - (5/dx);

y = j - (3/dy);

if (((x\*x + y\*y) >= r\*r) && (i >= M-r) && (j >= N-r))

continue;

fprintf(fd, "%d\t%d\t%f\n", i, j, (double) b[num[j \* M + i]]);

}

fprintf(fd, "\n");

}

fclose(fd);

FILE \*fg = fopen("gra.gnu", "w");

fprintf(fg, "#!/usr/bin/gnuplot -persistent\n");

fprintf(fg, "set size square\n");

fprintf(fg, "set output 'plot.eps'\n");

fprintf(fg, "set cbrange [55:200]\n");

fprintf(fg, "set xrange[0:%d]\nset yrange[0:%d]\n", M - 1, N - 1);

fprintf(fg, "set palette rgbformulae 22,13,-31\n");

fprintf(fg, "set pm3d map\n");

fprintf(fg, "set pm3d flush begin ftriangles scansforwar interpolate 5,5\n");

fprintf(fg, "splot 'out.txt' using 1:2:3 with pm3d title 'var'\n");

fprintf(fg, "pause -1");

system("gnuplot gra.gnu");

return 0;

}

Результаты работы программы

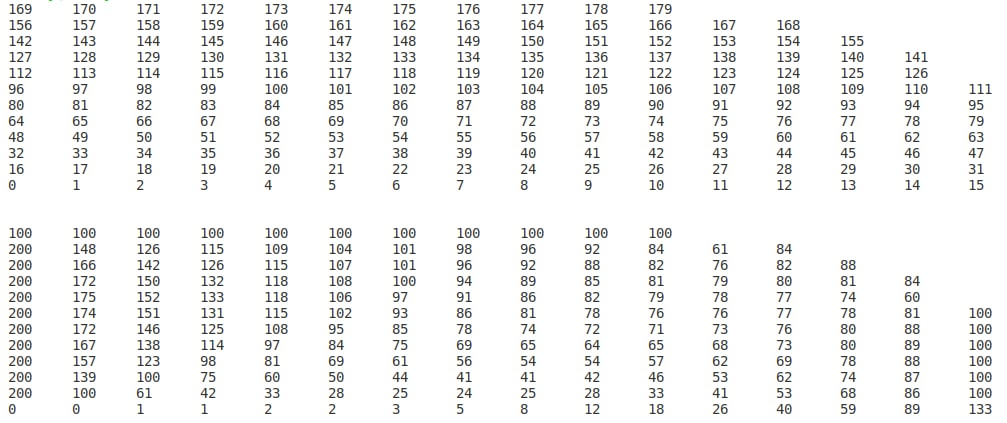


Рис. 2. Таблица узлов

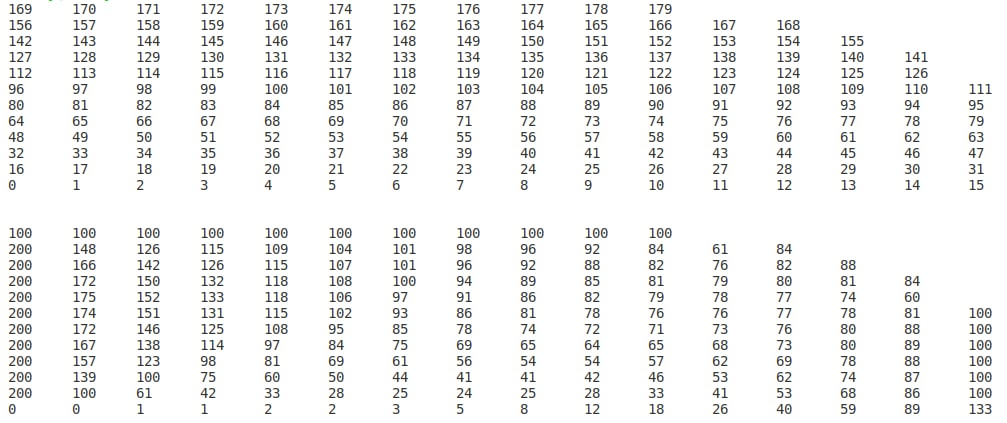


Рис. 3. Таблица температур в узлах для шага сетки 0.5

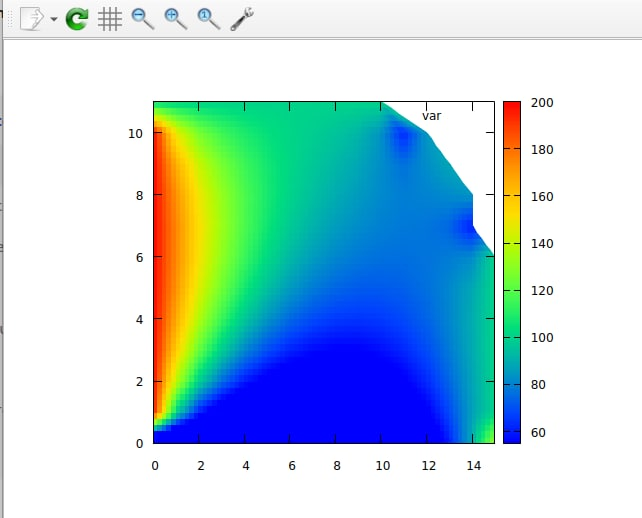


Рис. 4. Температурное поле, рассчитанное с помощью утилиты gnuplot

Рис. 5. Решение задачи в Ansys