Министерство образования и науки 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 Робототехники и комплексной автоматизации

КАФЕДРА Системы автоматизированного проектирования (РК-6)

ОТЧЕТ О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ

Студент	Журавлев Николай Вадимович
Группа	РК6-62б
Тип задания	Лабораторная работа
Тема лабораторной работы	Метод конечных разностей

Студент _____ **Журавлев Н.В.**
подпись, дата *фамилия, и.о.*

Преподаватель _____ **Трудоношин В.А**
подпись, дата *фамилия, и.о.*

Оценка _____

Москва, 2022 г.

Оглавление

Цель выполнения лабораторной работы

Цель выполнения лабораторной работы – ознакомиться с задачей теплопроводности и рассмотреть ее частный случай при программировании решения конкретной задачи.

Задание

С помощью явной разностной схемы решить нестационарное уравнение теплопроводности для трубы, изображенной на рис.1, там же указаны размеры сторон.

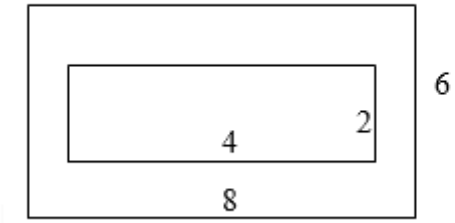

Рисунок 1

Граничные условия : внутри трубы протекает жидкость с температурой 100 градусов, на внешней границе задано условие:

dT/dn = T .

Начальное значение температуры трубы - 10 градусов.

При выводе результатов показать динамику изменения температуры (например, с помощью цветовой гаммы). Отчет должен содержать: текст программы, рисунок объекта с распределением температуры в момент времени 25 сек сравнение результатов расчета с результатами, полученными с помощью пакета ANSYS .

Теоретическая часть

В условии указано: с помощью явной разностной схемы решить нестационарное уравнение теплопроводности для прямоугольной пластины. Уравнение имеет вид:

$$\frac{dT}{dt} = a_1 \frac{d^2T}{dx^2} + a_2 \frac{d^2T}{dy^2}$$

Или в разностной схеме это будет выглядеть:

$$\frac{T_{i,j}^{k+1} - T_{i,j}^{k}}{\Delta t} = a_1 \frac{T_{i+1,j}^{k} - 2T_{i,j}^{k} + T_{i-1,j}^{k}}{\Delta x^2} + a_2 \frac{T_{i,j+1}^{k} - 2T_{i,j}^{k} + T_{i,j-1}^{k}}{\Delta y^2}$$

Боковые граничные условия:

- Для нижней стороны $\frac{T_{i,1} - T_{i,0}}{\Delta y} = T_{i,0}$;
- Для левой стороны $\frac{T_{2,j} - T_{0,j}}{\Delta x} = T_{0,j}$;
- Для правой стороны $\frac{T_{n-1,j} - T_{n,j}}{\Delta x} = T_{n,j}$;
- Для верхней стороны $\frac{T_{i,n-1} - T_{i,n}}{\Delta y} = T_{i,n}$;

Так же имеется начальный нагрев трубы – 10 градусов. Время моделирования 25 секунд.

Текст программы

Реализация представлена в листинге 1.

Листинг 1

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <string.h>
#define H 6.0
#define W 8.0
#define h 2.0
#define w 4.0
#define T_LIQUID 100.0
#define T_PIPE 10
#define TIME_RUNNING 25
#define A 1
#define DX 0.1
#define DY 0.1
#define PATH_PLT "t.plt"
#define PATH_RES "r.txt"
void get_index(int index, int ny, int* ix, int* iy) {
    *iy = index % ny;
    *ix = index / ny;
}
void write_to_file(double** T, FILE* fds1, int ny, int nx, int lx, int ly) {
```

```
    FILE *fd;
    if ((fd = fopen(PATH_PLT,"w")) == NULL) {
        printf("Cannot open file!\n");
        exit(0);
    }
    fprintf(fd, "set terminal png size 2000, 1000\n");
    fprintf(fd, "set output 'screen.png'\n");
    fprintf(fd, "set palette defined (0 \"black\", 1\"blue\", 2 \"green\", 3 \"yellow\", 4 \"orange\", 5 \"red\")\n");
    fprintf(fd, "set xrange[0:%d]\n", lx);
    fprintf(fd, "set yrange[0:%d]\n", ly);
    fprintf(fd, "set cbrange[0:100]\n");
    fprintf(fd, "plot '-' using 1:2:3 with image notitle\n");
    for (int k = 0; k < ny * nx; k++) {
        int i, j;
        get_index(k, ny,&i, &j);
        fprintf(fd, "%d %d %lf\n",i, j, T[i][j]);
        fprintf(fds1,"%d %d %3lf\n", i, j, T[i][j]);
    }
    fprintf(fd, "e\n");
    fprintf(fds1, "\n\n");
    fclose(fd);
}
void initial_pipe(double **T, double **T_next, int N, int li, int bj, int tj, int ri, int ny) {
    for (int k = 0; k < N; k++) {
        int i, j;
        get_index(k, ny, &i, &j);
        if (i == li && j >= bj && j <= tj) {
            T_next[i][j] = T_LIQUID;
            T[i][j] = T_LIQUID;
            continue;
        }
        // Правая
        if (i == ri && j >= bj && j <= tj) {
            T_next[i][j] = T_LIQUID;
            T[i][j] = T_LIQUID;
            continue;
        }
```

```
        // Верхняя
        if (j == tj && i >= li && i <= ri) {
            T_next[i][j] = T_LIQUID;
            T[i][j] = T_LIQUID;
            continue;
        }
        // Нижняя
        if (j == bj && i >= li && i <= ri) {
            T_next[i][j] = T_LIQUID;
            T[i][j] = T_LIQUID;
            continue;
        }
        if (i > li && i < ri && j > bj && j < tj) {
            T_next[i][j] = T_LIQUID;
            T[i][j] = T_LIQUID;
            continue;
        } else {
            T[i][j] = T_PIPE;
        }
    }
}
int main(int argc, char* argv[]) {
    if (argc != 4) {
        printf("Use %s nx ny file_name\n", argv[0]);
        exit(1);
    }
    FILE *file;
    char *file_name;
    int nx = atoi(argv[1]);
    int ny = atoi(argv[2]);
    file_name = argv[3];
    if ((file = fopen(file_name,"w")) == NULL) {
        printf("Cannot open file!\n");
        exit(0);
    }
    int N = nx * ny;
    double l = (W - w) / 2.0;
```

```
double b = (H - h) / 2.0;
double hx = W / (nx - 1);
double hy = H / (ny - 1);
int li = (int)(l / hx);
int ri = (int)((l + w) / hx);
int bj = (int)(b / hy);
int tj = (int)((b + h) / hy);
double **T = (double **)malloc(sizeof(double *) * nx);
double **T_next = (double **)malloc(sizeof(double *) * nx);

for (int k = 0; k < nx; k++) {
   T[k] = (double *) malloc(sizeof(double) * ny);
   T_next[k] = (double *) malloc(sizeof(double) * ny);
}

initial_pipe(T, T_next, N, li, bj, tj, ri, ny);

hx = DX;
hy = DY;

int i, j;
for (int g = 0; g < TIME_RUNNING; g++) {
   double ht = 0;
   for (int k = 0; k < N; k++) {
      get_index(k, ny, &i, &j);
      // Внутренние границы
      // Левая
      if (i == li && j >= bj && j <= tj) {
         T_next[i][j] = T_LIQUID;
         T[i][j] = T_LIQUID;
         continue;
      }
      // Правая
      if (i == ri && j >= bj && j <= tj) {
         T_next[i][j] = T_LIQUID;
         T[i][j] = T_LIQUID;
         continue;
```

```
        }
        // Верхняя
        if (j == tj && i >= li && i <= ri) {
            T_next[i][j] = T_LIQUID;
            T[i][j] = T_LIQUID;
            continue;
        }
        // Нижняя
        if (j == bj && i >= li && i <= ri) {
            T_next[i][j] = T_LIQUID;
            T[i][j] = T_LIQUID;
            continue;
        }

        if (i > li && i < ri && j > bj && j < tj) {
            T_next[i][j] = T_LIQUID;
            continue;
        }
        // Внешние границы
        // Левая и правая
        if (i == 0) {
            continue;
        }
        if (i == nx - 1) {
            continue;
        }
        //нижняя и верхняя
        if (j == 0) {
            continue;
        }
        if (j == ny - 1) {
            continue;
        }
        T_next[i][j] = ((T[i + 1][j] - 2 * T[i][j] + T[i - 1][j]) / (hx * hx) +
                        (T[i][j + 1] - 2 * T[i][j] + T[i][j - 1]) / (hy * hy)) * A;
        if (ht < fabs(T_next[i][j])) {
            ht = fabs(T_next[i][j]);
```

```
        }
    }
    for (int col = 0; col < nx; col++) {
        for (int row = 0; row < ny; row++) {
            if ((col == li && row >= bj && row <= tj) || (col == ri && row >= bj && row <= tj)
                || (row == tj && col >= li && col <= ri) || (row == bj && col >= li && col <= ri)
                || (col == 0) || (col == nx - 1) || (row == 0) || (row == ny - 1)
                || (col > li && col < ri && row > bj && row < tj)) {
                T[col][row] = T_next[col][row];
            } else {
                T[col][row] = T_next[col][row] * (2 / ht) + T[col][row];
            }
        }
    }
    for (int k = 0; k < N; k++) {
        get_index(k, ny, &i, &j);
        if (i == 0) {
            T[0][j] = (T[1][j]) / (1 + hx);
            continue;
        }
        if (i == nx - 1) {
            T[i][j] = (T[i - 1][j]) / (1 + hx);
            continue;
        }
        //нижняя и верхняя
        if (j == 0) {
            T[i][0] = (T[i][1]) / (1 + hy);
            continue;
        }
        if (j == ny - 1) {
            T[i][j] = (T[i][j - 1]) / (1 + hy);
            continue;
        }
    }
    T[0][0] = (T[0][1]) / (1 + hx);
    T[0][ny - 1] = (T[0][ny - 2]) / (1 + hy);
}
```

```
    for (j = ny - 1; j >= 0; j--) {
        for (i = 0; i < nx; i++) {
            printf("%8.3f", T[i][j]);
        }
        printf("\n");
    }
    FILE *fd;
    if ((fd = fopen(PATH_RES,"w")) == NULL) {
        printf("Cannot open file!\n");
        exit(0);
    }
    for (j = ny - 1; j >= 0; j--) {
        for (i = 0; i < nx; i++) {
            fprintf(fd, "%8.3f", T[i][j]);
        }
        fprintf(fd, "\n");
    }
    printf("\nhx = %g\nhy = %g\n", hx, hy);
    write_to_file(T,file, ny, nx, atoi(argv[1]) - 1, atoi(argv[2]) - 1);
    fclose(file);
}
```

Результаты работы программы

В результате работы программы получился следующий график распределения температуры в трубе для сетки 41 на 31 (в силу симметрии представлена ¼ часть таблицы):

24.627	26.755	28.735	30.891	32.779	34.867	36.589	38.485	39.948	41.535	42.669	43.892	44.682	45.549	46.024	46.581	46.782	47.071	47.026
27.089	29.430	31.608	33.980	36.056	38.354	40.248	42.333	43.943	45.689	46.936	48.281	49.150	50.104	50.627	51.239	51.460	51.778	51.729
29.646	31.949	34.712	36.854	39.685	41.632	44.349	46.022	48.426	49.734	51.688	52.588	54.066	54.580	55.632	55.811	56.508	56.394	56.789
31.903	34.923	37.225	40.493	42.570	45.881	47.697	50.781	52.237	54.847	55.852	57.887	58.447	59.951	60.130	61.201	61.062	61.786	61.359
34.597	37.150	40.711	43.002	46.813	48.863	52.612	54.340	57.656	58.900	61.533	62.236	64.194	64.439	65.859	65.747	66.759	66.352	67.043
36.634	40.319	42.894	47.057	49.409	53.753	55.875	59.947	61.625	64.936	65.939	68.366	68.771	70.503	70.480	71.722	71.385	72.273	71.668
39.342	42.248	46.576	49.264	54.054	56.630	61.451	63.782	67.944	69.511	72.471	73.176	75.181	75.319	76.713	76.496	77.495	77.020	77.736
41.108	45.407	48.429	53.451	56.449	61.878	64.950	70.156	72.766	76.495	77.725	79.982	80.359	81.812	81.753	82.760	82.444	83.170	82.655
43.644	46.981	52.003	55.327	61.089	64.738	70.890	74.894	80.285	82.536	85.089	85.776	87.160	87.270	88.157	88.007	88.630	88.320	88.771
45.039	49.856	53.420	59.205	63.134	69.751	74.668	81.993	87.821	90.955	91.945	93.029	93.239	93.829	93.810	94.200	94.075	94.355	94.154
47.187	50.944	56.508	60.510	67.095	72.058	79.881	87.624	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
48.087	53.269	57.291	63.581	68.246	75.599	81.861	90.654	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
49.680	53.737	59.614	64.039	70.962	76.407	84.263	91.477	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
49.991	55.368	59.644	66.131	71.058	78.426	84.577	92.469	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
50.952	55.154	61.150	65.733	72.701	78.172	85.745	92.435	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
50.634	56.069	60.420	66.946	71.921	79.232	85.271	92.868	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000

Рисунок 2. Таблица результатов работы программы

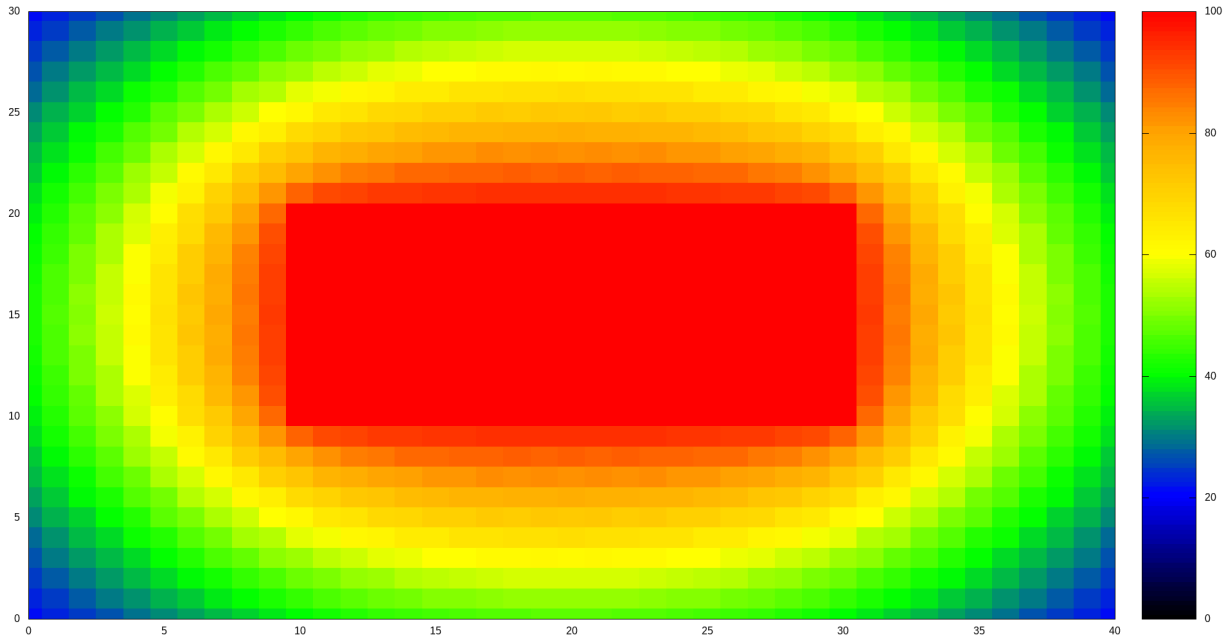

Рисунок 3. График результата работы программы

Результат работы ANSYS

Строим схему. Выбираем тип конечного элемента:

Main Menu: Preprocessor → Element Type → Add/Edit/Delete

Выбираем **Thermal Solid** (вид анализа) и **Triangl 6node**.

Теперь разбиваем область на конечные элементы:

Main Menu: Preprocessor → Meshing → Mesh Tool

Прикладываем граничное условие первого и третьего рода к линиям:

Main Menu: Preprocessor → Loads → Define Loads → Apply → Thermal Temperature → On Lines

Main Menu: Preprocessor →Loads → Loads- Apply →-> Thermal-Convection → On Areas

Задаем тип анализа:

Main Menu: Solution → Analysis Type→New Analysis

Выполняем команду расчета:

Main Menu: Solution → Solve → Current LS

Для просмотра поля температур выполняем команду:

Main Menu: General PostProc → Plot Results → Contour Plot → Nodal Solution

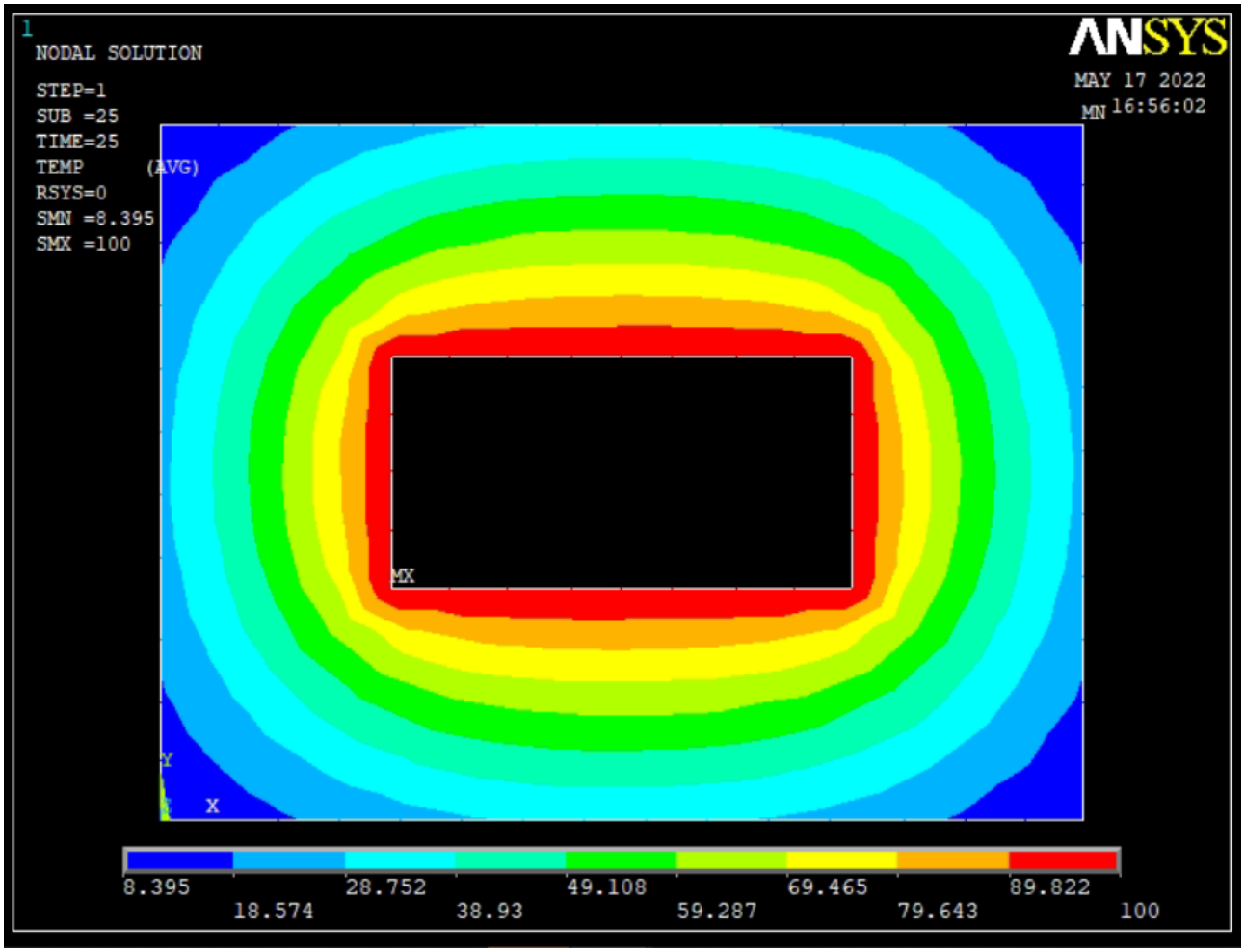

Рисунок 4. Результат в программе Ansys

Вывод

При выполнении домашнего задание работ была решена двумерная стационарная задача теплопроводности методом конечных разностей и проведена проверка результатов в Ansys.