Московский ордена Ленина, ордена Октябрьской Революции и ордена Трудового Красного Знамени Государственный Технический Университет им. Н. Э. Баумана

*Факультет* ***«Робототехники и комплексной автоматизации»***

*Кафедра* ***«Системы автоматизированного проектирования»***

ОТЧЕТ

По лабораторной работе

По курсу «Модели и методы проектных решений»

Вариант 14

Выполнил студент группы РК6-51

Радочин И. С.

Проверил: к.т.н., доцент кафедры РК-6

Трудоношин В. А.

Москва, 2014

Задание 14

С помощью неявной разностной схемы решить нестационарное уравнение теплопроводности для прямоугольной пластины размером 12\*4см. Начальное значение температуры пластины - 10 градусов.

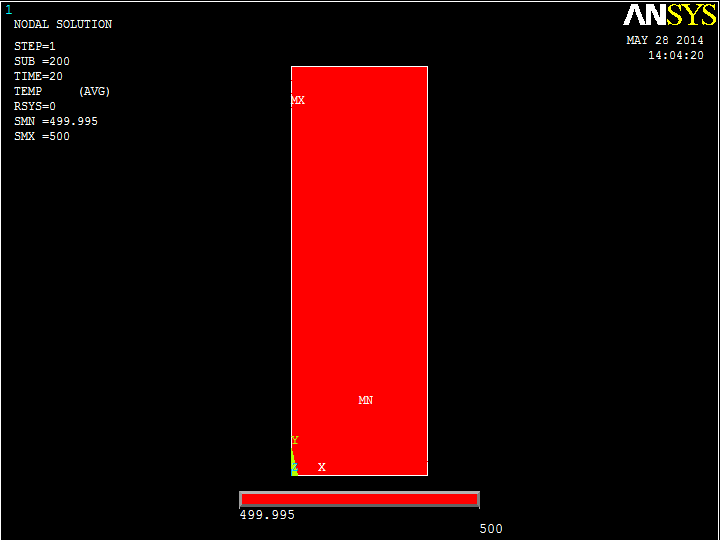
Граничные условия следующие: нижняя и правая половина верхней границы теплоизолированы, на остальной части границы температура 500 градусов.

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 20 сек, сравнение результатов расчета с результатами, полученными с помощью пакета ANSYS.

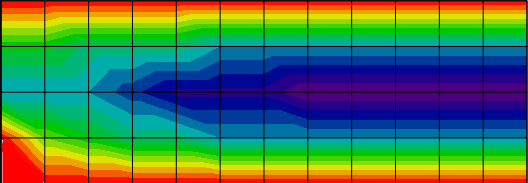
Полученные результаты

Результат работы программного комплекса ANSYS:



Смотря на результат выполнения лабораторной работы, который представлен ниже, можно заметить, что полученный результат очень сходится с результатом, полученным с помощью программного комплекса ANSYS. Расхождение можно понять, посмотрев год выпуска программного пакета ansys9.0 и разрядности тех времен.

Результат выполнения лабораторной работы:



500.00 500.00 499.94 499.96 500.00

500.00 499.97 499.94 499.96 500.00

500.00 499.96 499.94 499.95 500.00

500.00 499.95 499.93 499.95 500.00

500.00 499.95 499.92 499.95 500.00

500.00 499.94 499.92 499.94 500.00

500.00 499.94 499.92 499.94 500.00

500.00 499.94 499.91 499.94 500.00

500.00 499.94 499.91 499.94 500.00

500.00 499.94 499.91 499.94 500.00

500.00 499.94 499.91 499.94 500.00

500.00 499.94 499.91 499.94 500.00

500.00 499.94 499.91 499.94 500.00

Исходный код программы:

#include <iostream>

#include <cstdlib>

#include <cstdio>

double\* reverseGauss( double\* matrix, double\* result, size\_t n )

{

result[n-1] = matrix[ ( n - 1 ) \* ( n + 1 ) + n] / matrix[ (n-1) \* (n+1) + (n-1) ];

int i;

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

{

result[i] = matrix[i \* (n + 1) + n];

size\_t j;

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

{

result[i] -= matrix[i \* ( n + 1 ) + (j + 1)] \* result[ j + 1 ];

}

result[i] /= matrix[ i \* (n+1) + i ];

}

return result;

}

double\* forwardGauss( double\* matrix, size\_t n )

{

size\_t i;

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

{

size\_t j;

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

{

double m;

m = matrix[j \* (n + 1) + (i - 1)] / matrix[(i - 1) \* (n+1) + (i - 1)];

size\_t k;

for ( k = i - 1; k <= n; ++k )

{

matrix[j \* (n + 1) + k] -= m \* matrix[(i - 1) \* (n+1) + k];

}

}

}

return matrix;

}

void printResult( double\* result, size\_t inColumn, size\_t inRow )

{

for ( size\_t i = 0; i < inColumn; ++i )

{

for ( size\_t j = 0; j < inRow; ++j )

{

printf( "%10.2lf", result[i \* inRow + j]);

}

printf("\n");

}

}

void printMatrix( double\* matrix, size\_t n )

{

for ( size\_t i = 0; i < n; ++i )

{

for ( size\_t j = 0; j < n + 1; ++j )

printf( "%4.lf", matrix[i \* (n+1) + j]);

printf("\n");

}

}

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

{

if ( argc != 8 )

{

std::cout << "ОМГ, аргументиков, плизки!\n";

return EXIT\_FAILURE;

}

double height;

height = atof( argv[1] );

double width;

width = atof( argv[2] );

double elementSize;

elementSize = atof( argv[3] );

// Мона замутить прямоугольные конечные элементы, но лень.

double dx;

double dy;

dx = dy = elementSize;

double dt;

dt = atoi( argv[4] );

double tEnd;

tEnd = atoi( argv[5] );

double startTemperature;

startTemperature = atof( argv[6] );

double a;

a = atof( argv[7] );

size\_t numberOfNodes;

numberOfNodes = static\_cast<size\_t>( (height + 1) \* (width/dx + 1) );

size\_t numberOfElements;

numberOfElements = static\_cast<size\_t>( ( height \* width ) / ( dx \* dy) );

size\_t numberOfTimes;

numberOfTimes = static\_cast<size\_t>( tEnd / dt );

size\_t numberInRow;

numberInRow = static\_cast<size\_t>( width/dx + 1 );

size\_t numberInColumn;

numberInColumn = static\_cast<size\_t>( height/dy + 1 );

double\*\* matrix;

matrix = new double\* [numberOfTimes - 1];

for ( size\_t i = 0; i < numberOfTimes - 1; ++i )

{

matrix[i] = new double [numberOfNodes \* ( numberOfNodes + 1 )];

}

double\*\* results;

results = new double\* [numberOfTimes];

for ( size\_t i = 0; i < numberOfTimes; ++i )

{

results[i] = new double [numberOfNodes];

}

std::cout << "height(y size): " << height << "\n";

std::cout << "width(x size): " << width << "\n";

std::cout << "element size: " << elementSize << "\n";

std::cout << "dx: " << dx << "\n";

std::cout << "dy: " << dy << "\n";

std::cout << "dt: " << dt << "\n";

std::cout << "Конечное время: " << tEnd << "\n";

std::cout << "Начальная температура: " << startTemperature << "\n";

std::cout << "Постоянная теплопроводности: " << a << "\n";

std::cout << "Количество узлов: " << numberOfNodes << "\n";

std::cout << "Количество строк: " << numberInColumn << "\n";

std::cout << "Количество узлов в строке: " << numberInRow << "\n";

std::cout << "Количество конечных элементов: " << numberOfElements << "\n";

std::cout << "Количество шагов времени: " << numberOfTimes << "\n";

std::cout << "Размер матрицы коэффициентов в байтах: " << numberOfNodes \* ( numberOfNodes + 1 ) \* numberOfTimes << "\n";

std::cout << "Размер матрицы ответов в байтах: " << numberOfNodes \* numberOfTimes << "\n";

// Начальные условия

for ( size\_t i = 0; i < numberOfNodes; ++i)

{

results[0][i] = startTemperature;

}

// std::cout << "------------------------\n";

// printResult(results[0], (int)(height/dy), int(width/dx));

// std::cout << "------------------------\n";

// Здеся граничные условия первого рода на границах

// double first = 500;

// double second = 10;

// Здеся они закончилися

for ( size\_t i = 0; i < numberOfTimes - 1; ++i )

{

for( size\_t j = 0; j < numberOfNodes; ++j )

{

if ( ( j >= width/( 2 \* dx) && j < width/dx ) )

{

// std::cout << j << "\n";

matrix[i][j \* (numberOfNodes+1) + j] = 1;

matrix[i][j \* (numberOfNodes+1) + j + numberInRow] = -1;

matrix[i][j \* (numberOfNodes+1) + numberOfNodes] = 0;

} else

if ( j >= numberOfNodes - numberInRow+1 && j < numberOfNodes-1 )

{

// std::cout << j << "\n";

matrix[i][j \* (numberOfNodes+1) + j] = 1;

matrix[i][j \* (numberOfNodes+1) + j - numberInRow] = -1;

matrix[i][j \* (numberOfNodes+1) + numberOfNodes] = 0;

} else

if ( ( (j+1) % (numberInRow) == 0 ) ||

( (j+1) % (numberInRow) == 1 ) )

{

// std::cout << j << "\n";

matrix[i][j \* (numberOfNodes+1) + j] = 1;

matrix[i][j \* (numberOfNodes+1) + numberOfNodes] = 500;

} else

if ( j < width/( 2 \* dx) )

{

// std::cout << j << "\n";

matrix[i][j \* (numberOfNodes+1) + j] = 1;

matrix[i][j \* (numberOfNodes+1) + numberOfNodes] = 500;

} else

{

matrix[i][j \* (numberOfNodes+1) + numberOfNodes] =

(-1) \* results[i][j] \* (dx\*dx\*dy\*dy/(a\*dt));

matrix[i][j \* (numberOfNodes+1) + j - 1] = dy\*dy;

matrix[i][j \* (numberOfNodes+1) + j + 1] = dy\*dy;

matrix[i][j \* (numberOfNodes+1) + j + numberInRow] = dx\*dx;

matrix[i][j \* (numberOfNodes+1) + j - numberInRow] = dx\*dx;

matrix[i][j \* (numberOfNodes+1) + j] =

(-1) \* (2\*dy\*dy + 2\*dx\*dx + dx\*dx\*dy\*dy/(a\*dt));

}

} }

forwardGauss(matrix[i], numberOfNodes);

reverseGauss(matrix[i], results[i+1], numberOfNodes );

// printResult(results[i+1], numberInColumn, numberInRow);

// std::cout << "------------------------\n";

}

printResult(results[numberOfTimes - 1], numberInColumn, numberInRow);

for (size\_t i = 0; i < numberOfTimes - 1; ++i)

{

delete[] matrix[i];

}

delete[] matrix;

for (size\_t i = 0; i < numberOfTimes; ++i)

{

delete[] results[i];

}

delete[] results;

return 0;

}