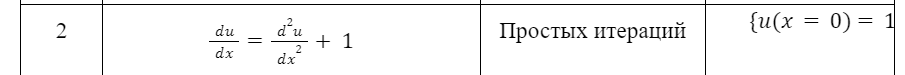
Задание №4

1. Представить задачу в нестационарном виде.
2. Записать разностную схему в соответствии с методом варианта.
3. Для метода установления, требующего прогонки, привести схему к виду, удобному для использования метода прогонки, проверить сходимость прогонки, вывести 1, 1, uNn+1 из граничных условий.
4. Записать рекуррентное соотношение.
5. Составить алгоритм (блок-схему) расчёта.
6. Построить программу на любом удобном языке программирования
7. Для метода установления с использованием явной схемы провести расчёты при h= 0,1;0,01 и трёх различных t (подобрать их самим), где

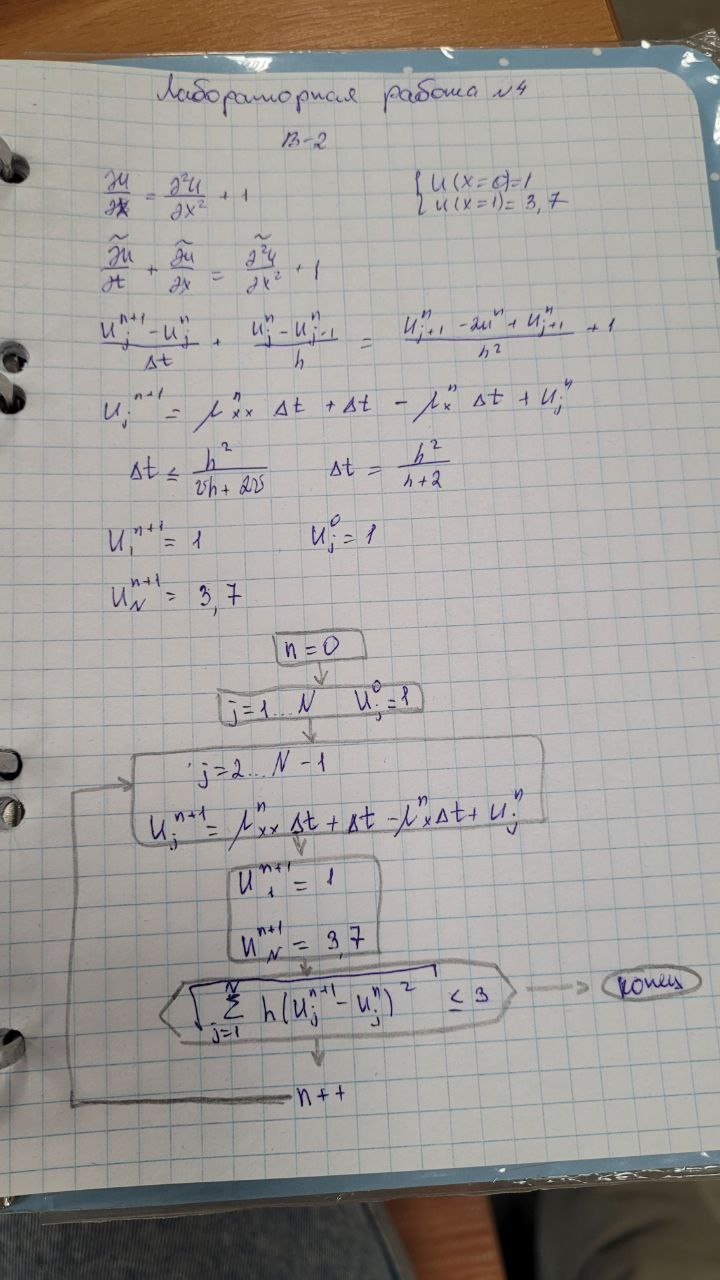
th2vh+2σ

1. Для метода установления с использованием неявных схем проводить расчёты при h= 0,1;0,01 и t={ 0,1;0,01;0,001 }.
2. Сравнить результаты вычислений между собой в точках:

x={ 0;0,1;0,2;0,3;0,4;0,5;0,6;0,7;0,8;0,9;1}



|  |  |  |  |
| --- | --- | --- | --- |
| 2 | du/dx=d2u/dx2+1 | Простых итераций | u(x=0)=1 u(x=1)=3,7 |



#include <stdio.h>

#include <stdlib.h>

#include <math.h>

double newU(double u\_jp, double u\_j, double u\_jm, double h, double dt) {

return (u\_j - dt \* (u\_j - u\_jm)/h + dt \* (u\_jp - 2 \* u\_j + u\_jm) / (h \* h) + dt);

}

double zeroU(double h, double dt) {

return 1;

}

double sU(double u\_jp, double u\_j, double u\_jm, double h, double dt) {

return 1;

}

double eU(double u\_jp, double u\_j, double u\_jm, double h, double dt) {

return 3.7;

}

int main()

{

double h = 0.1;

double dt = 0.001;

double \*u[2];

int size = (int)(1 / h);

// Allocate memory for u

for (int i = 0; i < 2; i++) {

u[i] = (double\*)malloc(size \* sizeof(double));

if (u[i] == NULL) {

printf("Memory allocation failed\n");

return 1;

}

}

// Initialize u

for (int i = 0; i < 2; i++) {

for (int j = 0; j < size; j++) {

u[i][j] = 0;

}

}

for (int j = 0; j < size; j++) {

u[0][j] = zeroU(h, dt);

}

// Iterative loop

double e = 10;

double eps = 0.000001;

while (e > eps) {

u[1][0] = sU(u[0][0], u[0][0], u[0][0], h, dt);

u[1][size - 1] = eU(u[0][0], u[0][0], u[0][0], h, dt);

for (int j = 1; j < size - 1; j++) {

u[1][j] = newU(u[0][j-1], u[0][j], u[0][j+1], h, dt);

}

double s = 0;

for (int j = 0; j < size; j++) {

s += h \* (u[1][j] - u[0][j]) \* (u[1][j] - u[0][j]);

}

e = sqrt(s);

if (e > eps) {

for (int j = 0; j < size; j++) {

u[0][j] = u[1][j];

}

}

}

// Output results

for (int j = 0; j < size; j++) {

printf("%lf ", u[0][j]);

}

printf("\n");

for (int j = 0; j < size; j++) {

printf("%lf ", u[1][j] - u[0][j]);

}

// Free allocated memory

for (int i = 0; i < 2; i++) {

free(u[i]);

}

return 0;

}

Результат:

1.000000 1.468232 1.884812 2.254439 2.581386 2.869533 3.122407 3.343212 3.534861 3.700000

0.000000 0.000001 0.000001 0.000001 0.000001 0.000001 0.000001 0.000001 0.000000 0.000000