| 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 «Информатика и системы управления»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

***Лабораторная работа № 3***

**Тема:** Построение и программная реализация алгоритма сплайн-интерполяции табличных функций.

**Студент:** Козлова И. В.

**Группа:** ИУ7-42Б

**Оценка (баллы):** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Преподаватель:** Градов В.М.

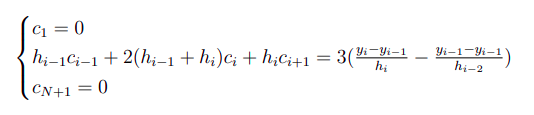
*Москва*

*2021 г*

### Цель работы

Получение навыков владения методами интерполяции таблично заданных функций с помощью кубических сплайнов.

### Алгоритм решения

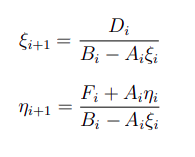
1. Решить СЛАУ 

2 <= n <= N

Данная система уравнений решается методом прогонки, которая состоит из двух этапов.

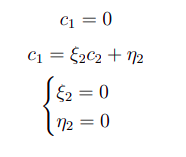
1. Первый этап - прямой ход.

при заданных начальных значениях прогоночных коэффициентов 𝜉𝑖 и 𝜂𝑖 определяются все прогоночные коэффициенты:



1. Второй этап - обратный ход.

при известном 𝑐𝑁 определяются все 𝑐𝑖 , 𝑖 = 1...𝑁 .



Имея граничные условия, находим начальные коэффициенты (прямой ход).

Нахождение 𝑐𝑖 (обратный ход):



### Исходные данные

1. Таблица функции с количеством узлов N. Задать с помощью формулы y = x^2 в диапазоне [0..10] с шагом 1.

2. Значение аргумента x в первом интервале, например, при х=0.5 и в середине таблицы, например, при x= 5.5. Сравнить с точным значением

### Код программы

Файл для генерации таблицы значений.

| **#include <stdio.h>** **#include <stdlib.h>** **#include <math.h>**  **#define OK 0**  **double** **F**(**double** x) {  **return** x \* x; }  **int** **main**() {  FILE \*f = fopen("table.txt", "w");   **double** start, stop, step;  printf("Input Start X, Stop of X, Step X: ");  scanf("%lf%lf%lf", &start, &stop, &step);   fprintf(f, "Count %d\n\n", (**int**)((stop - start) / step) + 1);   **for** (**double** cur = start; cur <= stop; cur += step)  {  fprintf(f, "%lf %lf\n", cur, F(cur));  }   fclose(f);   **return** OK; } |
| --- |

Файл основной программы.

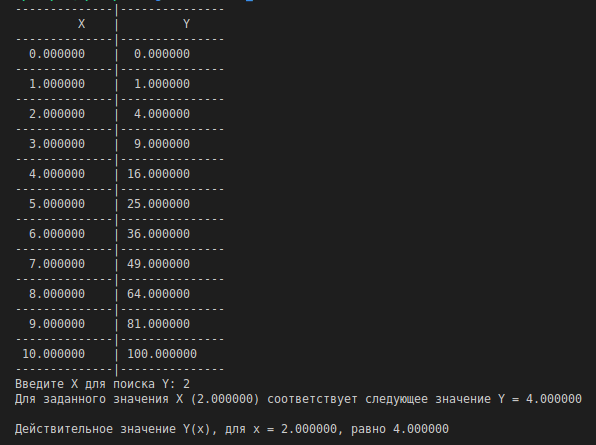
| **#include <stdio.h>** **#include <stdlib.h>**  **#define FILE\_NAME "table.txt"**  **#define OK 0** **#define FILE\_ERROR -1** **#define SOME\_ERROR -2**  **typedef** **struct** {  **int** count;  **double** \*x\_arr;  **double** \*y\_arr; } table\_r;   **double** **F**(**double** x) {  **return** x \* x; }  **void** **free\_table**(table\_r \*table) {  **if** (table == NULL)  {  **return**;  }   table->count = 0;   **if** (table->x\_arr)  {  free(table->x\_arr);  }   **if** (table->y\_arr)  {  free(table->y\_arr);  } }   **int** **read\_table**(FILE \*f, table\_r \*table) {  **if** (f == NULL)  {  **return** FILE\_ERROR;  }   **int** rc = OK, count;\   rc = fscanf(f, "Count %d", &count);  **if** (rc != 1 || count <= 0)  {  printf("Error while reading file.\n");    **return** FILE\_ERROR;  }   table->count = count;  table->x\_arr = calloc(count, **sizeof**(**double**));  table->y\_arr = calloc(count, **sizeof**(**double**));   rc = 2;  **for** (**int** i = 0; i < count && rc == 2; ++i)  {  rc = fscanf(f, "%lf%lf", table->x\_arr + i, table->y\_arr + i);  }   **if** (rc != 2)  {  printf("Error while reading file.\n");    **return** FILE\_ERROR;  }   **return** OK; }   **void** **print\_table**(table\_r table) {  printf("--------------|---------------\n");  printf("%10s |%10s ", "X", "Y");  printf("\n");  printf("--------------|---------------\n");   **for** (**int** i = 0; i < table.count; i++)  {  printf("% 10lf |% 10lf \n", \*(table.x\_arr + i), \*(table.y\_arr + i));  printf("--------------|---------------\n");  } }   **int** **found\_in\_x**(**double** x, table\_r table) {  **int** find = 0;   **for** (**int** i = 0; i < table.count; ++i)  {  **if** (x < table.x\_arr[i])  {  find = i;  **break**;  }  }   **return** find; }  *// Сплайн используется для нахождения приближенного* *// значения функции, заданной в табличном представлении.* **int** **interp\_spline**(table\_r table, **double** x\_arg) {  **if** (table.count <= 0)  {  printf("Some error with table.\n");   **return** SOME\_ERROR;  }   **double** \*arr\_a = calloc(**sizeof**(**double**), table.count);  **double** \*arr\_b = calloc(**sizeof**(**double**), table.count);  **double** \*arr\_d = calloc(**sizeof**(**double**), table.count);  **double** \*arr\_A = calloc(**sizeof**(**double**), table.count);  **double** \*arr\_B = calloc(**sizeof**(**double**), table.count);  **double** \*arr\_D = calloc(**sizeof**(**double**), table.count);  **double** \*arr\_F = calloc(**sizeof**(**double**), table.count);  **double** \*arr\_C = calloc(**sizeof**(**double**), table.count);  **double** \*arr\_Kci = calloc(**sizeof**(**double**), table.count);  **double** \*arr\_Etta = calloc(**sizeof**(**double**), table.count);  **double** \*arr\_Hn = calloc(**sizeof**(**double**), table.count);   **for** (**int** i = 1; i < table.count; ++i)  {  arr\_Hn[i] = table.x\_arr[i] - table.x\_arr[i - 1];  }   *// Прямой ход (1 этап алгоритма)*  **for** (**int** i = 2; i < table.count; ++i)  {  arr\_A[i] = arr\_Hn[i - 1];  arr\_D[i] = arr\_Hn[i];   arr\_B[i] = (-2) \* (arr\_A[i] + arr\_D[i]);   arr\_F[i] = (-3) \* ((table.y\_arr[i] - table.y\_arr[i - 1]) / arr\_Hn[i] -  (table.y\_arr[i] - table.y\_arr[i - 2]) / arr\_Hn[i - 1]);   arr\_Kci[i + 1] = arr\_D[i] / (arr\_B[i] - arr\_A[i] \* arr\_Kci[i]);  arr\_Etta[i + 1] = (arr\_A[i] \* arr\_Etta[i] + arr\_F[i]) /  (arr\_B[i] - arr\_A[i] \* arr\_Kci[i]);  }   *// Обратный ход (2 этап)*  **for** (**int** i = table.count - 2; i >= 2; i--)  {  arr\_C[i] = arr\_Kci[i + 1] \* arr\_C[i + 1] + arr\_Etta[i + 1];  }   **for** (**int** i = table.count - 1; i > 0; i--)  {  arr\_a[i] = table.y\_arr[i - 1];  arr\_d[i] = (arr\_C[i + 1] - arr\_C[i]) / (3 \* arr\_Hn[i]);  arr\_b[i] = (table.y\_arr[i] - table.y\_arr[i - 1]) / arr\_Hn[i] -  (1 / 3) \* arr\_Hn[i] \* (arr\_C[i + 1] + 2 \* arr\_C[i]);  }   *// Нахождение значения по заданному X*  **int** found\_ix = found\_in\_x(x\_arg, table);   **double** x = x\_arg - table.x\_arr[found\_ix - 1];  **double** x2 = x \* x;  **double** x3 = x \* x \* x;   **double** result = arr\_a[found\_ix] + arr\_b[found\_ix] \* x + arr\_C[found\_ix] \* x2 +  arr\_d[found\_ix] \* x3;   printf("Для заданного значения X (%lf) соответствует следующее значение Y = %lf\n\n", x\_arg, result);    free(arr\_Hn);  free(arr\_A);  free(arr\_B);  free(arr\_D);  free(arr\_F);  free(arr\_C);  free(arr\_a);  free(arr\_b);  free(arr\_d);  free(arr\_Kci);  free(arr\_Etta);   **return** OK; }   **int** **main**() {  FILE \*f = fopen(FILE\_NAME, "r");  **if** (f == NULL)  {  printf("Error while reading file.\n");    **return** FILE\_ERROR;  }   table\_r table = { 0 };   read\_table(f, &table);  fclose(f);   print\_table(table);    printf("Введите X для поиска Y: ");  **double** x\_arg;  **int** rc = scanf("%lf", &x\_arg);   **if** (rc != 1)  {  printf("Ошибка при считывания данного.\n");   **return** SOME\_ERROR;  }   interp\_spline(table, x\_arg);  printf("Действительное значение Y(x), для x = %lf, равно %lf\n\n", x\_arg, F(x\_arg));   **return** OK; } |
| --- |



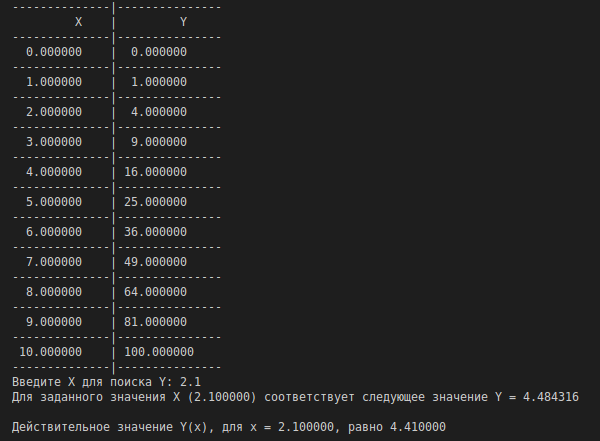
### Результаты работы

1. Значения y(x).

Примеры работы программы (рис. 1 и рис. 2).



*рис. 1*



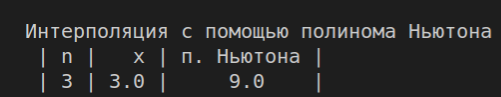
*рис. 2*

2. Сравнить результаты интерполяции кубическим сплайном и полиномом Ньютона 3-ей степени.

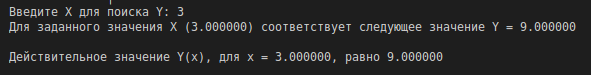
Результаты интерполяции полиномом Ньютона 3 степени (n = 3) - рис. 3 и рис. 5

Результаты интерполяции кубическим сплайном - рис. 4 и рис. 6

Результаты при X = 3

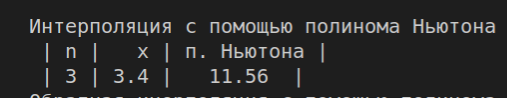


*рис. 3*

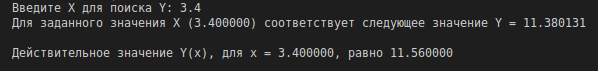


*рис. 4*

Результаты при X = 3.4



*рис. 5*

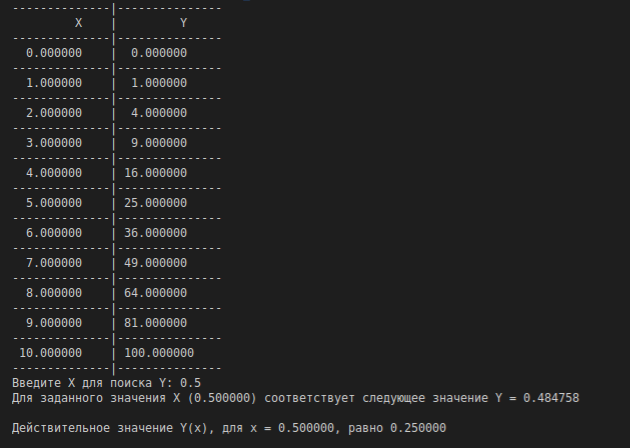


*рис. 6*

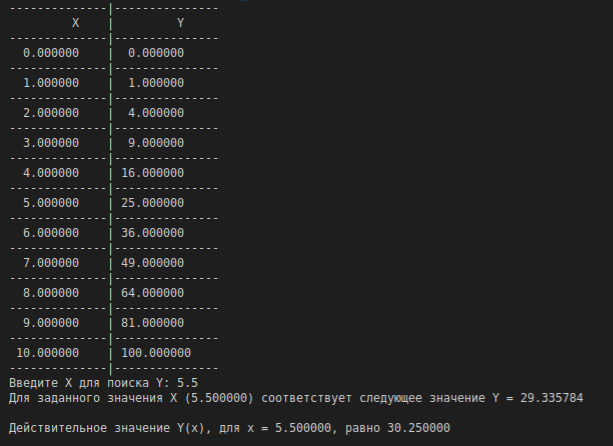
Для сравнения была использована лабораторная работа №1, в которой реализована интерполяция полиномом Ньютона.

3. Выходные данные с учетом входных данных из условия.

Рисунки 7 и 8.



*рис. 7*



*рис. 8*

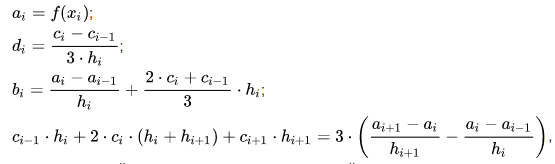
### Контрольные вопросы

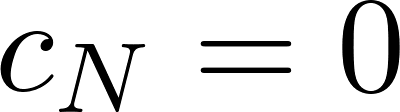
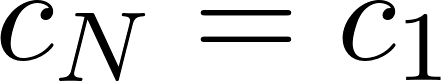
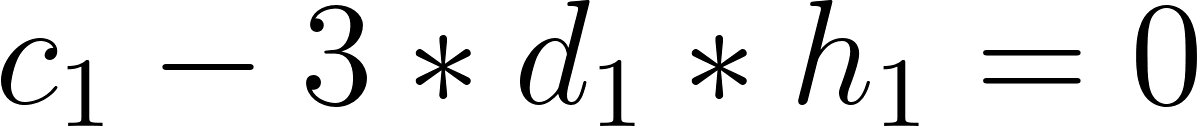
***1. Получить выражения для коэффициентов кубического сплайна, построенного на двух точках.***

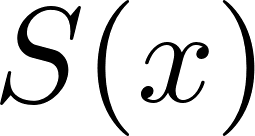
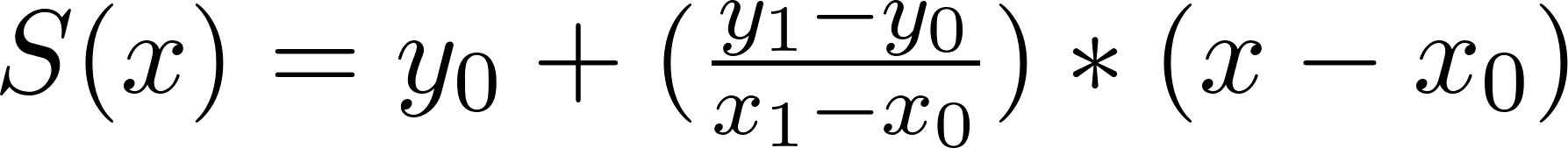
Функция выродится в прямую, так как коэффициенты С и D будут равны 0.

Полином третьей степени.

Формулы для вычисления коэффициентов.

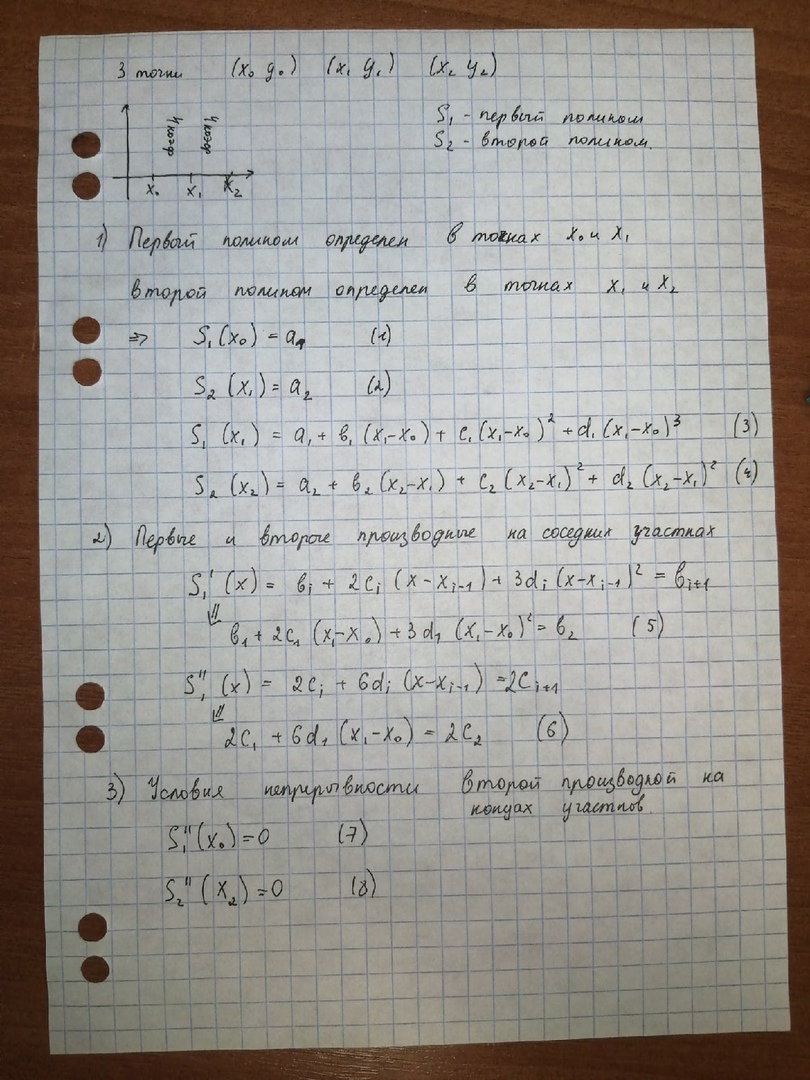


Причем [](https://www.codecogs.com/eqnedit.php?latex=c_%7BN%7D%20%3D%200#0), а так как сплайн строится на 2-х точках, то можно сказать, что [](https://www.codecogs.com/eqnedit.php?latex=c_%7BN%7D%20%3D%20c_%7B1%7D#0), отсюда можно написать, что [](https://www.codecogs.com/eqnedit.php?latex=c_%7B1%7D-3*d_%7B1%7D*h_%7B1%7D%20%3D%200#0) => d = 0. То есть можно сказать, что из-за того, что c и d равны 0, то на 2-х точках мы получим прямую.

Так как при подстановке в полином [](https://www.codecogs.com/eqnedit.php?latex=S(x)#0) пропадают 2-ая и 3-ья степени, и мы получаем [](https://www.codecogs.com/eqnedit.php?latex=S(x)%20%3D%20y_%7B0%7D%20%2B%20(%5Cfrac%7By_%7B1%7D%20-%20y_%7B0%7D%7D%7Bx_%7B1%7D%20-%20x_%7B0%7D%7D)%20*%20(x%20-%20x_%7B0%7D)#0)

***2. Выписать все условия для определения коэффициентов сплайна, построенного на 3-х точках.***

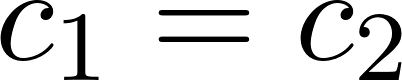
Так как сплайн построен на 3-х точках, то условий будет 8, так как у нас есть 2 участка. Далее формулы записаны на рис. 9



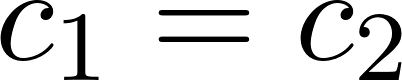
*рис. 9*

***3. Определить начальные значения прогоночных коэффициентов, если принять, что для коэффициентов сплайна справедливо C1=C2.***

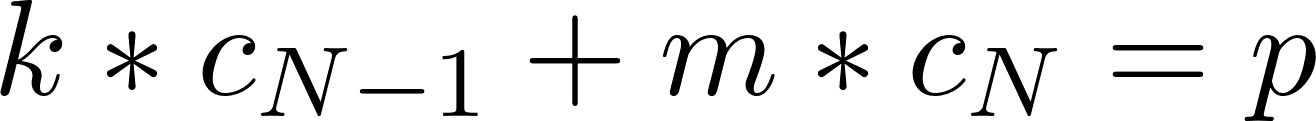
Из условия дано, что

[******](https://www.codecogs.com/eqnedit.php?latex=c_%7B1%7D%20%3D%20c_%7B2%7D#0)   
Тогда, в данную функцию вместо [](https://www.codecogs.com/eqnedit.php?latex=c_%7B1%7D#0) можно подставить [](https://www.codecogs.com/eqnedit.php?latex=c_%7B2%7D#0)

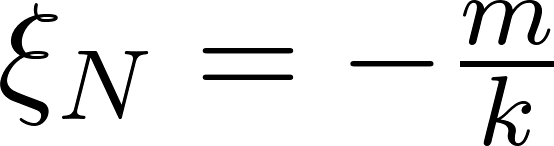
[](https://www.codecogs.com/eqnedit.php?latex=c_%7B1%7D#0) = [](https://www.codecogs.com/eqnedit.php?latex=%5Cxi_%7B2%7D#0)\*[](https://www.codecogs.com/eqnedit.php?latex=c_%7B2%7D#0) + [](https://www.codecogs.com/eqnedit.php?latex=%5Ceta_%7B2%7D#0)

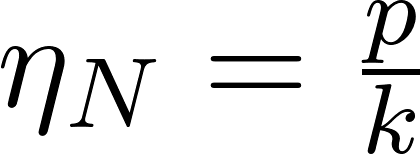
Из этих условий следует, что так как [******](https://www.codecogs.com/eqnedit.php?latex=c_%7B1%7D%20%3D%20c_%7B2%7D#0), то [](https://www.codecogs.com/eqnedit.php?latex=%5Cxi_%7B2%7D#0) = 1, а [](https://www.codecogs.com/eqnedit.php?latex=%5Ceta_%7B2%7D#0) = 0.

***4. Написать формулу для определения последнего коэффициента сплайна СN , чтобы можно было выполнить обратный ход метода прогонки, если в качестве граничного условия задано kCN-1+mCN=p, где k,m и p - заданные числа.***

По условию дано: [](https://www.codecogs.com/eqnedit.php?latex=k%20*%20c_%7BN-1%7D%20%2B%20m%20*%20c_%7BN%7D%20%3D%20p#0) (1)

Из этой формулы можно вывести

[](https://www.codecogs.com/eqnedit.php?latex=%5Cxi_%7BN%7D%20%3D%20-%5Cfrac%7Bm%7D%7Bk%7D#0) (2)

[](https://www.codecogs.com/eqnedit.php?latex=%5Ceta_%7BN%7D%20%3D%20%5Cfrac%7Bp%7D%7Bk%7D#0) (3)

Далее по формуле (далее вычисления на рис. 10)

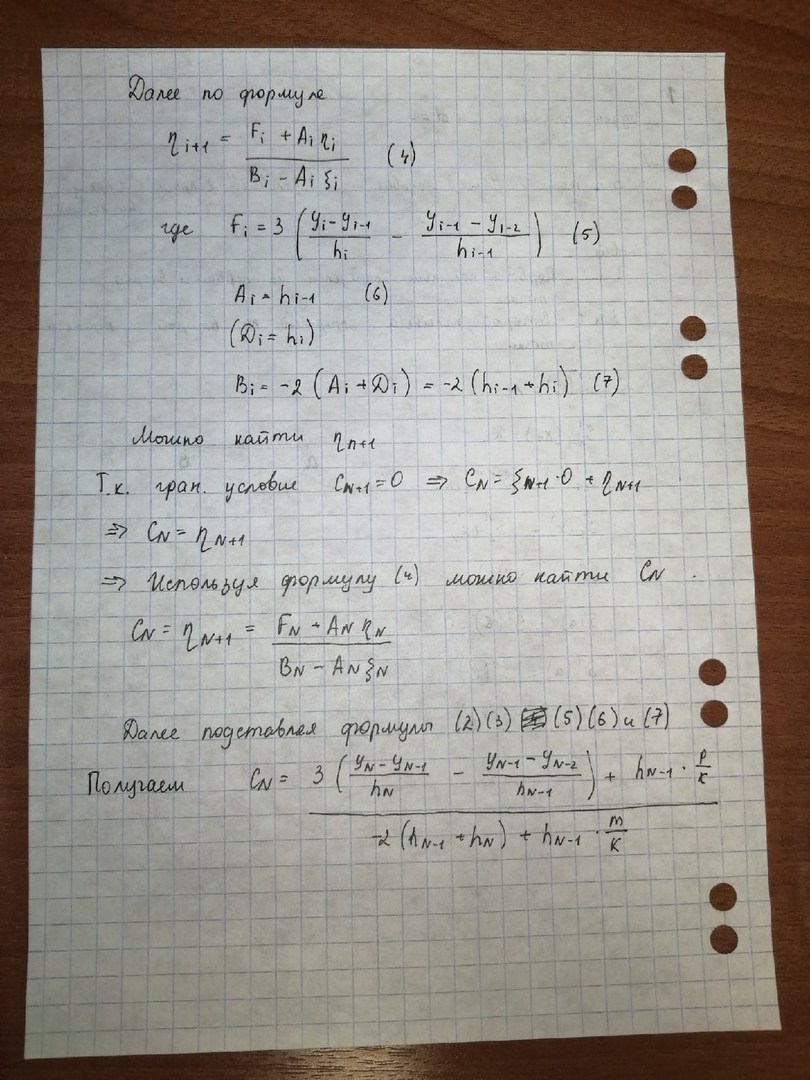


рис. 10