Министерство науки и высшего образования
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Московский государственный технический университет
имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

Факультет «Фундаментальные науки»
Кафедра «Высшая математика»

Отчет по технологической практике за 5-й семестр 2020-2021 учебного года

Руководитель практики _____ Кравченко О. В.
подпись, дата

Студенты группы ФН1–51Б _____ Бочкарева А. И.
подпись, дата

_____ Слесарчук Е.
подпись, дата

Москва
2020

Содержание

1 Задача

Решить сингулярную двуточечную краевую задачу методом конечных разностей

$$\varepsilon y''(x) + p(x)y'(x) + q(x)y(x) + f(x) = 0, \quad x \in [0,\, 1],$$

со смешанными граничными условиями

$$\begin{aligned} -\alpha_1 y'(0) + \alpha_2 y(0) &= \gamma_1, \\ \beta_1 y'(1) + \beta_2 y(1) &= \gamma_2, \end{aligned}$$

при различных значениях параметра

$$\varepsilon = 1,\ 0.1,\ 0.01,\ 0.001.$$

№	α_1	α_2	β_1	β_2	γ_1	γ_2	$p(x)$	$q(x)$	$f(x)$
3	1	0	1	0	4	7	$x - x^2$	$\sqrt[3]{\sin^2(x) + \cos(x)}$	$x^3 - x^2 + \sqrt{x}$
18	0	1	0	1	1	4	$\sqrt{x\sin(x) + x^2}$	$\sin(x) + \cos(x)$	$\sqrt[3]{\sqrt{x^3 - x^2} + x}$

2 Теоретическая часть

Наиболее распространенным и универсальным численным методом решения дифференциальных уравнений является метод конечных разностей. Основное содержание метода заключается в следующем. Область непрерывного изменения аргумента (например, отрезок) заменяется дискретным множеством точек, называемых узлами. Эти узлы составляют разностную сетку. Искомая функция непрерывного аргумента приближенно заменяется функцией дискретного аргумента на заданной сетке. Эта функция называется сеточной. Исходное дифференциальное уравнение заменяется разностным уравнением относительно сеточной функции. При этом для входящих в уравнение производных используются соответствующие конечно-разностные соотношения. Такая замена дифференциального уравнения разностным называется его аппроксимацией на сетке (или разностной аппроксимацией).

Решение дифференциального уравнения сводится к отысканию значений сеточной функции в узлах сетки. Обоснованность замены дифференциального уравнения разностным, точность получаемых решений, устойчивость метода ? важнейшие вопросы, которые требуют тщательного изучения.

Рассмотрим содержание метода на примере решения дифференциального уравнения второго порядка

$$y'' = f\left(x, y, y'\right) \tag{1}$$

при заданных граничных условиях

$$y(0) = y_0, \quad y(1) = y_1 \tag{2}$$

Разобьем отрезок $[0, 1]$ на n равных частей точками $x_i = ih, (i = 0, 1, ..., n)$. Решение краевой задачи (1), (2) сведем к вычислению значений сеточной функции y_i в узловых точках x_i. Для этого воспользуемся уравнением (2) для внутренних узлов:

$$y''(x_i) = f\left(x_i, y\left(x_i\right), y'\left(x_i\right)\right), \quad i = 1, 2, ..., n-1 \tag{3}$$

Заменим производные, входящие в эти соотношения, их конечноразностными аппроксимациями:

$$y'\left(x_i\right) = \frac{y_{i+1} - y_{i-1}}{2h} + O(h^2), \quad y''\left(x_i\right) = \frac{y_{i+1} - 2y_i + y_{i-1}}{h} + O(h^2), \tag{4}$$

Подставляя эти выражения в (2), получаем систему разностных уравнений

$$F(x_i, y_{i-1}, y_i, y_{i+1}) = 0, \quad i = 1, 2, ..., n-1 \tag{5}$$

являющуюся системой $(n-1)$ алгебраических уравнений относительно значений сеточной функции $y_1, y_2, ..., y_{n-1}$. Входящие в данную систему y_0 (при $i = 1$) и yn (при $i = n-1$) выбираются из граничных условий (2):

$$y_0 = y(0), \quad y_n = y(1)$$

На практике часто граничные условия задаются в более общем виде:

$$\begin{aligned} \alpha_1 y(0) + \beta_1 y'(0) = \gamma_1, \\ \alpha_2 y(1) + \beta_2 y'(1) = \gamma_2 \end{aligned} \tag{6}$$

В этом случае граничные условия также должны представляться в разностном виде путем аппроксимации производных $y'(0)$ и $y'(1)$ с помощью конечно-разностных соотношений.

Если использовать односторонние разности, при которых производные аппроксимируются с первым порядком точности, то разностные граничные условия примут вид

$$\begin{aligned} \alpha_1 y_0 + \beta_1 \frac{y_1 - y_0}{h} &= \gamma_1, \\ \alpha_2 y_n + \beta_2 \frac{y_n - y_{n-1}}{h} &= \gamma_2 \end{aligned} \tag{7}$$

Из этих соотношений легко находятся значения y_0 и y_n.
Однако, как правило, предпочтительнее аппроксимировать производные, входящие в (6), со вторым порядком точности с помощью центральных разностей

$$y'(0) = \frac{y_1 - y_{-1}}{2h} + O(h^2), \qquad y''(1) = \frac{y_{n+1} - y_{n-1}}{2h} + O(h^2),$$

В данные выражения входят значения сеточной функции y_{-1} и y_{n+1} в так называемых фиктивных узлах $x = -h$ и $x = 1 + h$. В этих узлах значения искомой функции также должны быть найдены. Следовательно, количество неизвестных значений сеточной функции увеличивается на два. Для замыкания системы привлекают еще два разностных уравнения (5) при $i = 0$ и $i = n$.
Аппроксимировать граничные условия со вторым порядком можно и иначе. В этом случае используются следующие аппроксимации:

$$y'(0) = \frac{-3y_0 + 4y_1 - y_2}{2h} + O(h^2), \qquad y'(1) = \frac{y_{n-2} - 4y_{n-1} + 3y_n}{2h} + O(h^2),$$

Таким образом, решение краевой задачи для дифференциального уравнения сведено к решению системы алгебраических уравнений вида (5). Эта система является линейной или нелинейной в зависимости от того, линейно или нелинейно, искомое дифференциальное уравнение. Для нелинейной системы необходимо использовать специальные методы, в том числе итерационные. Для решения линейной системы уравнений (5) используется метод прогонки.
Линейная система уравнений (5) приводится к виду

$$\begin{cases} B_0 y_0 + C_0 y_1 = F_0, \\ A_i y_{i-1} + B_i y_i + C_i y_{i+1} = F_i, \quad i = 1, 2, ..., n-1, \\ A_n y_{n-1} + B_n y_n = F_n. \end{cases} \tag{8}$$

Система уравнений (8) имеет ненулевые элементы матрицы коэффициентов только на трех диагоналях, так называемая трехдиагональная матрица.

$$\begin{vmatrix} B_0 & C_0 & & & & 0 \\ A_1 & B_1 & C_1 & & & \\ & A_2 & B_2 & C_2 & & \\ & & ... & ... & ... & \\ & & & A_{n-1} & B_{n-1} & C_{n-1} \\ 0 & & & & A_n & B_n \end{vmatrix}$$

Такую систему уравнений удобнее всего решать методом прогонки. Будем искать решение системы уравнений в виде:

$$y_i = \alpha_i y_{i+1} + \beta_i \tag{9}$$

Здесь α_k и β_k - неизвестные коэффициенты, обычно называемые в литературе "прогоночные коэффициенты"или "коэффициенты прогонки". Равенство (9) имеет место при всех i, поэтому можно записать

$$y_{i-1} = \alpha_{i-1} y_i + \beta_{i-1} \tag{10}$$

Подставим (10) в (8), получим:

$$A_i(\alpha_{i-1} y_i + \beta_{i-1}) + B_i y_i + C_i y_{i+1} = F_i,$$

И перепишем его в форме (9):

$$y_i = -\frac{C_i}{A_i \alpha_{i-1} + B_i} y_{i+1} + \frac{F_i - A_i \beta_{i-1}}{A_i \alpha_{i-1} + B_i} \tag{11}$$

Сравнение (11) и (9) дает рекуррентные соотношения для определения прогоночных коэффициентов α_i и β_i:

$$\alpha_i = -\frac{C_i}{A_i \alpha_{i-1} + B_i}, \quad \beta_i = \frac{F_i - A_i \beta_{i-1}}{A_i \alpha_{i-1} + B_i}, \quad i = 1, 2, ..., n-1 \tag{12}$$

Недостающие коэффициенты α_0 и β_0 определяются из первого граничного условия (8):

$$y_0 = -\frac{C_0}{B_0} y_1 + \frac{F_0}{B_0} = \alpha_0 y_1 + \beta_0$$

Откуда

$$\alpha_0 = -\frac{C_0}{B_0}, \quad \beta_0 = \frac{F_0}{B_0} \tag{13}$$

Для точки $i = n - 1$ справедливо уравнение

$$y_{n-1} = \alpha_{n-1} y_n + \beta_{n-1} \tag{14}$$

Привлекая второе граничное условие из (8)

$$A_n y_{n-1} + B_n y_n = F_n, \tag{15}$$

решаем систему уравнений (14) (15) относительно yn и находим

$$y_n = \frac{F_n - A_n \beta_{n-1}}{A_n \alpha_{n-1} + B_n}. \tag{16}$$

Таким образом, схема решения системы уравнений (8) состоит в следующей последовательности действий:
1. По формулам (13) вычисляем α_0 и β_0.
2. По реккурентным формулам (12) вычисляем последовательно $\alpha_1, \beta_1, ..., \alpha_{n-1}, \beta_{n-1}$.
3. По формуле (16) определяем y_n.
4. По формуле (9) вычисляем $y_{n-1}, y_{n-2}, ..., y_0$.
После выполнения действий 1 - 4 все значения y_i будут определены.

3 Код программы

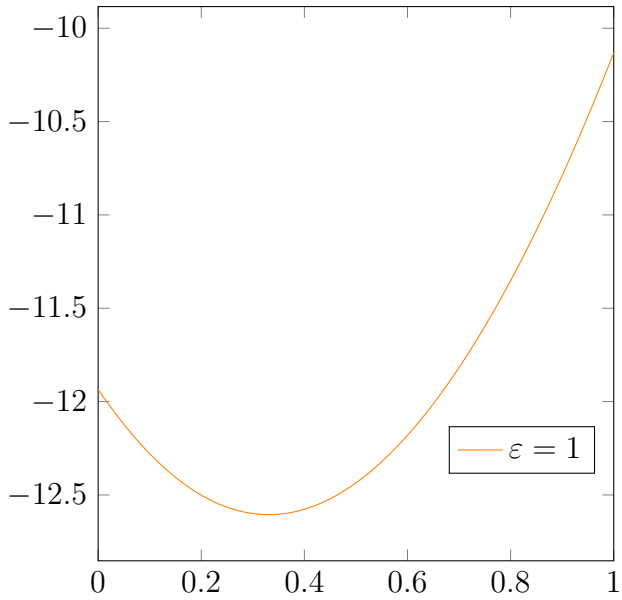
```
1  #include "pch.h"
2  #include <iostream>
3  #include <cmath>
4  #include <vector>
5  #include <fstream>
6  using namespace std;
7  double p1(double x)
8  {
9  return x - pow(x, 2);
10 }
11 double q1(double x)
12 {
13 return pow(pow(sin(x), 2) + cos(x), 1 / 3);
14 }
15 double f1(double x)
16 {
17 return pow(x, 3) - pow(x, 2) + sqrt(x);
18 }
19 double p2(double x)
20 {
21 return sqrt(x*sin(x) + pow(x, 2));
22 }
23 double q2(double x)
24 {
25 return sin(x) + cos(x);
26 }
27 double f2(double x)
28 {
29 return pow(pow(pow(x, 3) - pow(x, 2), 0.5) + x, 1/3);
30 }
31 /********МЕТОД КОНЕЧНЫХ РАЗНОСТЕЙ********/
32 //       для уравнения вида
33 //       eps * y'' + p(x) * y' + q(x) * y + f(x) = 0
34 //       с граничными условиями вида
35 //       -alpha1 * y'(0) + alpha2 * y(0) = gamma1
36 //       beta1 * y'(1) + beta2 * y(1) = gamma2
37 void FiniteDifferenceMethod(double alpha1, double alpha2, double
      beta1, double beta2, double gamma1, double gamma2, double epsilon
      , double p(double x), double q(double x), double f(double x),
      const char string[])
38 {
39 double a = 0, b = 1;
40 int n = 100;
41 double h = (b - a) / n;
42 vector <double> x(n + 1, 0), y(n + 1, 0);
43 //создаем сетку иксов
44 for (int i = 0; i < n + 1; i++)
```

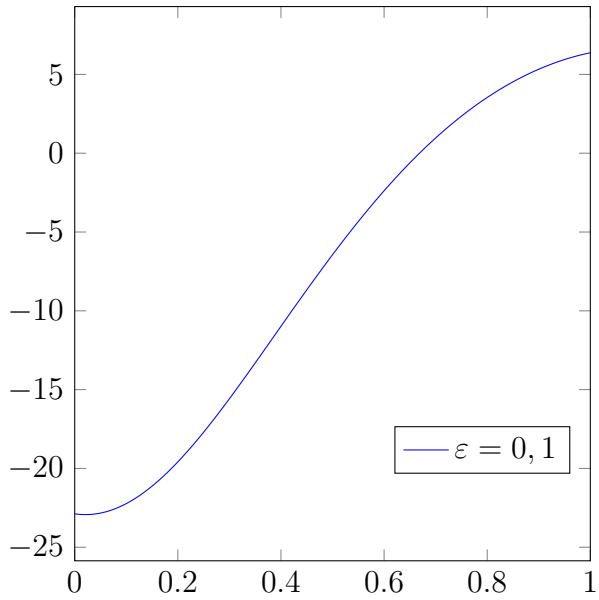
```
45  {
46  x[i] = a + i * h;
47  }
48  //создание векторов для значений трехдиаг. матрицы.
49  vector <double> A(n, 0), B(n + 1, 0), C(n, 0), F(n + 1, 0);
50  //заполняем "нижнюю" диагональ A
51  for (int i = 0; i < n - 1; i++)
52  {
53  A[i] = epsilon - p(x[i + 1])*h / 2;
54  }
55  A[n - 1] = -beta1;
56  //заполняем главную диагональ B
57  B[0] = alpha1 + alpha2 * h;
58  for (int i = 1; i < n; i++)
59  {
60  B[i] = pow(h, 2)* q(x[i]) - 2 * epsilon;
61  }
62  B[n] = beta1 + beta2 * h;
63  //заполняем "верхнюю" диагональ C
64  C[0] = - alpha1;
65  for (int i = 1; i < n; i++)
66  {
67  C[i] = epsilon + p(x[i])*h / 2;
68  }
69  //заполняем столбец свободных членов F
70  F[0] = gamma1 * h;
71  for (int i = 1; i < n; i++)
72  {
73  F[i] = -f(x[i])* pow(h, 2);
74  }
75  F[n] = gamma2 * h;
76  /**********МЕТОД ПРОГОНКИ**********/
77  //создаем дополнительные вектора для коэффициентов P и Q
78  vector <double> P(n, 0), Q(n + 1, 0);
79  //заполняем первые элементы векторов P и Q
80  P[0] = - C[0] / B[0];
81  Q[0] = F[0] / B[0];
82  //вычисляем остальные коэффициенты P
83  for (int i = 1; i < n; i++)
84  {
85  P[i] = - C[i] / (B[i] + A[i - 1] * P[i - 1]);
86  }
87  //вычисляем остальные коэффициенты Q
88  for (int i = 1; i < n + 1; i++)
89  {
90  Q[i] = (F[i] - A[i - 1] * Q[i - 1]) / (B[i] + A[i - 1] * P[i - 1]);
91  }
92  //вычисляем значения y
93  y[n] = Q[n];
```

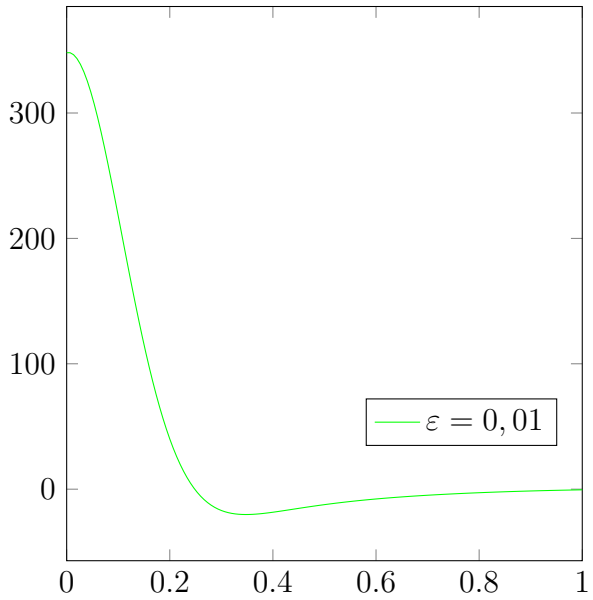
```
94   for (int i = n - 1; i > -1; i--)
95   {
96   y[i] = P[i] * y[i + 1] + Q[i];
97   }
98   ofstream fout(string);
99   for (int i = 0; i < n + 1; i++)
100  {
101  fout << x[i] << "," << y[i] << endl;
102  }
103  fout.close();
104  }
105  int main()
106  {
107  FiniteDifferenceMethod(1, 0, 1, 0, 4, 7, 1,p1,q1,f1, "var3_1.txt");
108  FiniteDifferenceMethod(1, 0, 1, 0, 4, 7, 0.1, p1, q1, f1, "var3_0.1.
       txt");
109  FiniteDifferenceMethod(1, 0, 1, 0, 4, 7, 0.01, p1, q1, f1, "var3_0
       .01.txt");
110  FiniteDifferenceMethod(1, 0, 1, 0, 4, 7, 0.001, p1, q1, f1, "var3_0
       .001.txt");
111
112  FiniteDifferenceMethod(0, 1, 0, 1, 1, 4, 1, p2, q2, f2, "var18_1.txt
       ");
113  FiniteDifferenceMethod(0, 1, 0, 1, 1, 4, 0.1, p2, q2, f2, "var18_0
       .1.txt");
114  FiniteDifferenceMethod(0, 1, 0, 1, 1, 4, 0.01, p2, q2, f2, "var18_0
       .01.txt");
115  FiniteDifferenceMethod(0, 1, 0, 1, 1, 4, 0.001, p2, q2, f2, "var18_0
       .001.txt");
116
117  cout << "Good work!";
118  }
```

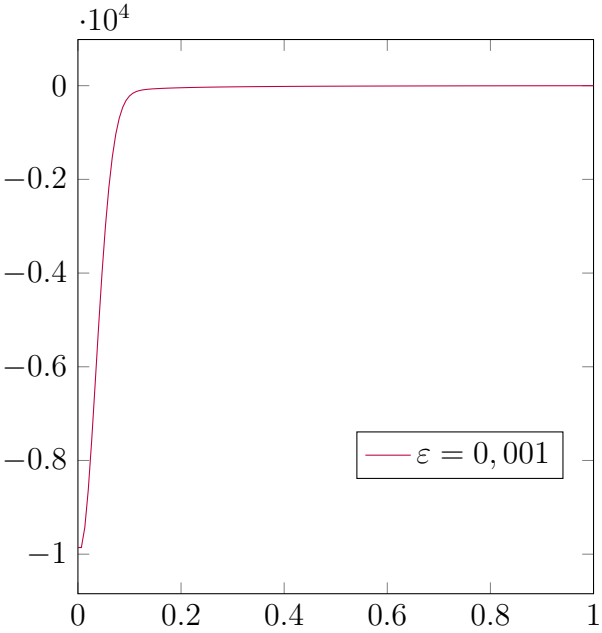
4 Результаты

Первая задача









Вторая задача