Московский государственный университет
имени М. В. Ломоносова
Факультет вычислительной математики и кибернетики

Отчет по заданию №2

«Численное решение третьей краевой задачи для стационарного уравнения теплопроводности с непрерывными коэффицентами. Метод прогонки.»

Выполнил:
студент 305 группы
Татосьян В. Г.

Преподаватель:
Есикова Н. Б.

Москва 2020

Содержание

Постановка задачи

- Найти решение обыкновенного дифференциального уравнения второго порядка с непрерывными коэффицентами, с двумя краевыми условиями третьего рода с точностью $\varepsilon = 0.01$:
  $\frac{d}{dx}[k(x)\frac{du}{dx}] - q(x)u = -f(x); 0 < x < 1; k(x) \geq c_1 > 0; q(x) \geq 0$
  Краевые условия:
  $k(0)u'(0) = \beta_1 u(0) + \mu_1$
  $-k(1)u'(1) = \beta_2 u(1) + \mu_2$
  где $\beta_1, \beta_2 \geq 0; \beta_1 + \beta_2 > 0; q(x), k(x), f(x)-$непрерывные функции

- Для моего варианта:
  $k(x) = x^2 + 1; q(x) = x; f(x) = e^{-x};$
  $\beta_1 = 0; \beta_2 = 1; \mu_1 = 0; \mu_2 = 0; \varepsilon = 0.01$

- Реализация конечно-разностной схемы второго порядка апроксимации для решения данной задачи.

- Значение шага h для разностной схемы выбрать изходя из требований к точности решения. Решение разностной задачи искать методом прогонки.

- Для оценки погрешности решения краевой задачи воспользоваться правилом Рунге.

Аналитическое решение задачи

Решаем модельную задачу: $\bar{k} = k(0.5); \bar{q} = q(0.5); \bar{f} = f(0.5)$
Решение $u(x)$ обыкновенного диффернциального уравнения второго порядка выглядит следующим образом:
$u(x) = C_1 e^{\sqrt{\frac{q}{k}}x} + C_2 e^{-\sqrt{\frac{q}{k}}x} + \frac{f}{q}$
Вычисляя производную:
$u(x) = C_1 \frac{q}{k} e^{\sqrt{\frac{q}{k}}x} - C_2 \frac{q}{k} e^{-\sqrt{\frac{q}{k}}x}$
И подставляя краевые условия:
$ku'(0) = 0$
$-ku'(1) = u(1)$
Получим:
$C_1 = C_2 = C = \frac{f}{q((\sqrt{kq}-1)e^{-\sqrt{\frac{q}{k}}} - (\sqrt{kq}+1)e^{\sqrt{\frac{q}{k}}})}$

Численное решение задачи

Для численного решения задачи введем равномерную сетку ω.
$\omega = \{x_i = ih | \, i = [0, N], h = \frac{1}{N}\}$
Заменим производную второго порядка конечно-разностным отношением вида:
$\frac{d}{dx}(k(x)\frac{du}{dx}) = \frac{k(\frac{x_{i+1}-xi}{h}) - k(\frac{x_i - xi-1}{h})}{h}$
Получим уравнение приближающее дифференциальное уравнение в узле:
$k(x)\frac{y(x_{i+1})-2y(x_i)+y(x_{i-1})}{h^2} - q(x)y(x_i) = -f(x)$ при $x \in (0, 1)$
Далее решаем систему линейных уравнений методом прогонки.

Пример работы программы

i	x_i	$u(x_i)$	$y(x_i)$	$delta$
0	0	0.516205	0.516205	9.53146e-08
10	0.05	0.515857	0.515857	9.56527e-08
20	0.1	0.514811	0.514811	9.66675e-08
30	0.15	0.513067	0.513067	9.8361e-08
40	0.2	0.510623	0.510623	1.00736e-07
50	0.25	0.507476	0.507476	1.03798e-07
60	0.3	0.503624	0.503624	1.07551e-07
70	0.35	0.499062	0.499062	1.12003e-07
80	0.4	0.493786	0.493787	1.17163e-07
90	0.45	0.487791	0.487791	1.23039e-07
100	0.5	0.481071	0.481071	1.29642e-07
110	0.55	0.473618	0.473619	1.36986e-07
120	0.6	0.465426	0.465427	1.45082e-07
130	0.65	0.456487	0.456487	1.53947e-07
140	0.7	0.44679	0.446791	1.63596e-07
150	0.75	0.436328	0.436328	1.74048e-07
160	0.8	0.425088	0.425088	1.85321e-07
170	0.85	0.413061	0.413061	1.97436e-07
180	0.9	0.400233	0.400233	2.10416e-07
190	0.95	0.386593	0.386593	2.24283e-07
200	1	0.372126	0.372126	2.39063e-07

Таблица 1: Значения аналитического решения $u(x)$ и численного $y(x)$ для модельной задачи
при $N = 100$

i	x_i	$u(x_i)$	$y(x_i)$	$delta$
0	0	0.516205	0.516205	9.56265e-10
100	0.05	0.515857	0.515857	9.59655e-10
200	0.1	0.514811	0.514811	9.69796e-10
300	0.15	0.513067	0.513067	9.86705e-10
400	0.2	0.510623	0.510623	1.0104e-09
500	0.25	0.507476	0.507476	1.04094e-09
600	0.3	0.503624	0.503624	1.07836e-09
700	0.35	0.499062	0.499062	1.12278e-09
800	0.4	0.493786	0.493786	1.17425e-09
900	0.45	0.487791	0.487791	1.23292e-09
1000	0.5	0.481071	0.481071	1.29889e-09
1100	0.55	0.473618	0.473618	1.37222e-09
1200	0.6	0.465426	0.465426	1.45305e-09
1300	0.65	0.456487	0.456487	1.54154e-09
1400	0.7	0.44679	0.44679	1.63783e-09
1500	0.75	0.436328	0.436328	1.74214e-09
1600	0.8	0.425088	0.425088	1.8547e-09
1700	0.85	0.413061	0.413061	1.97572e-09
1800	0.9	0.400233	0.400233	2.1054e-09
1900	0.95	0.386593	0.386593	2.244e-09
2000	1	0.372126	0.372126	2.39174e-09
1	1	1	0	

Таблица 2: Значения аналитического решения $u(x)$ и численного $y(x)$ для модельной задачи
при $N = 1000$

i	x_i	$y(x_i)$
0	0	0.600579
10	0.05	0.599363
20	0.1	0.595864
30	0.15	0.590319
40	0.2	0.582982
50	0.25	0.574114
60	0.3	0.563975
70	0.35	0.552818
80	0.4	0.540882
90	0.45	0.528388
100	0.5	0.515537
110	0.55	0.502505
120	0.6	0.489446
130	0.65	0.476491
140	0.7	0.463748
150	0.75	0.451308
160	0.8	0.439242
170	0.85	0.427603
180	0.9	0.416433
190	0.95	0.405762
200	1	0.395609

Таблица 3: Значения численного $y(x)$ для общей задачипри $N = 100$

i	x_i	$y(x_i)$
0	0	0.600574
100	0.05	0.599359
200	0.1	0.59586
300	0.15	0.590316
400	0.2	0.582979
500	0.25	0.574111
600	0.3	0.563972
700	0.35	0.552815
800	0.4	0.540879
900	0.45	0.528386
1000	0.5	0.515535
1100	0.55	0.502503
1200	0.6	0.489444
1300	0.65	0.476489
1400	0.7	0.463747
1500	0.75	0.451307
1600	0.8	0.43924
1700	0.85	0.427601
1800	0.9	0.416432
1900	0.95	0.405761
2000	1	0.395608

Таблица 4: Значения численного $y(x)$ для общей задачипри $N = 1000$

Исходный код

```
#include <iostream>
#include <cmath>


double estimationError (double * u, double * y, int N);
void realSolv(double * u, int N, double x0);
void sweepMethod(double * y, int N, double x0);
void generalProblem(int N);
void modelProblem(int N);


int main() {
    int N, model;

    std::cout << "Hello!" << std::endl;
    std::cout << "Please, choose which problem we will solve: model - 0, general - 1.\
    std::cin  >> model;
    std::cout << "Please, enter N: ";
    std::cin  >> N;

    if (model == 0) {
        modelProblem(N);
    } else if (model == 1) {
        generalProblem(N);
    } else {
        std::cout << "When entering the first digit, choose between 0 and 1!\n
            Please, restart program!" << std::endl;
    }

    return 0;
}


double estimationError(double * y2, double * y, int N) {
    double res;

    N = N / 2;

    for (int i = 0; i < N + 1; i++) {
        if (res < abs(y[2 * i] - y2[i]) / 3) {
            res = abs(y[2 * i] - y2[i]) / 3;
        }
    }

    return res;
}


void realSolv(double * u, int N, double x0) {
    double * x;

    double h = 1.0 / N;

```

```
54      double k = x0 * x0 + 1;
55      double q = x0;
56      double f = exp(-x0);
57
58      double C = f / (q * ((sqrt(k * q) - 1) * exp(-sqrt(q / k))
59          - (sqrt(k * q) + 1) * exp(sqrt(q / k))));
60      x = new double [N + 1];
61
62      for (int i = 0; i < N + 1; i++) {
63          x[i] = i * h;
64      }
65
66      for (int i = 0; i < N + 1; i++) {
67          u[i] = C * (exp(sqrt(q / k) * x[i]) + exp(-sqrt(q / k) * x[i])) + f / q;
68      }
69
70      delete [] x;
71  }
72
73
74  void sweepMethod(double * y, int N, double x0) {
75      double * k;
76      double * q;
77      double * fi;
78      double * x;
79
80      double * alf;
81      double * bet;
82      double * a;
83      double * b;
84      double * c;
85
86      double h = 1.0 / N;
87
88      k = new double [N + 1];
89      q = new double [N + 1];
90      fi = new double [N + 1];
91      x = new double [N + 1];
92
93      alf = new double [N + 2];
94      bet = new double [N + 2];
95      a = new double [N + 2];
96      b = new double [N + 2];
97      c = new double [N + 2];
98
99      for (int i = 0; i < N + 1; i++) {
100         x[i] = i * h;
101
102         if (x0 == 0.0) {
103             k[i] = x[i] * x[i] + 1;
104             q[i] = x[i];
105             fi[i] = exp(-x[i]);
106         } else {
107             k[i] = x0 * x0 + 1;
108             q[i] = x0;
109             fi[i] = exp(-x0);
```

```
110             }
111         }
112 
113         for (int i = 1; i < N; i++) {
114             a[i] = 1 / (2 * h * h) * (k[i - 1] + k[i]);
115             b[i] = 1 / (2 * h * h) * (k[i] + k[i + 1]);
116             c[i] = q[i] + 1 / (2 * h * h) * (k[i - 1] + 2 * k[i] + k[i + 1]);
117         }
118 
119         alf[1] = (k[0] + k[1]) / (k[0] + k[1] + h * h * q[0]);
120         bet[1] = h * h * fi[0] / (k[0] + k[1] + h * h * q[0]);
121 
122         for (int i = 1; i < N; i++) {
123             alf[i + 1] = b[i] / (c[i] - a[i] * alf[i]);
124             bet[i + 1] = (fi[i] + a[i] * bet[i]) / (c[i] - a[i] * alf[i]);
125         }
126 
127         double mu2 = 1 + 0.5 * h * q[N] + (k[N] + k[N - 1]) / (2 * h);
128         double bet2 = (k[N] + k[N - 1]) / (2 * h * mu2);
129 
130         y[N] = (0.5 * h * fi[N] / mu2 + bet2 * bet[N]) / (1 - alf[N] * bet2);
131 
132         for (int i = N - 1; i >= 0; i--) {
133             y[i] = alf[i + 1] * y[i + 1] + bet[i + 1];
134         }
135 
136         delete [] k;
137         delete [] q;
138         delete [] fi;
139         delete [] x;
140 
141         delete [] alf;
142         delete [] bet;
143         delete [] a;
144         delete [] b;
145         delete [] c;
146 }
147 
148 
149 void modelProblem(int N) {
150         double * u;
151         double * y;
152         double * y2;
153 
154         int j = 0;
155 
156         double eps = 0.01, x0 = 0.5;
157 
158         while (N < 100000) {
159             u = new double [N + 1];
160             y = new double [N + 1];
161 
162             realSolv(u, N, x0);
163             sweepMethod(y, N, x0);
164 
165             if (j == 1) {
```

```
166             if (estimationError(y2, y, N) < eps) {
167                 for (int i = 0; i < N + 1; i += 50) {
168                     std::cout << "x[" << i << "] = " << i * 1.0 / N << " , u[" << i <<
169                         i << "] = " << y[i] << " , delta = " << abs(u[i] - y[i]) << st
170                 }
171 
172                 delete [] u;
173                 delete [] y;
174                 delete [] y2;
175 
176                 break;
177             } else {
178                 delete [] y2;
179 
180                 y2 = new double [N + 1];
181 
182                 for (int i = 0; i < N + 1; i++) {
183                     y2[i] = y[i];
184                 }
185 
186                 N = 2 * N;
187             }
188         } else {
189             y2 = new double [N + 1];
190 
191             for (int i = 0; i < N + 1; i++) {
192                 y2[i] = y[i];
193             }
194 
195             N = 2 * N;
196         }
197 
198         delete [] u;
199         delete [] y;
200 
201         j = 1;
202     }
203 }
204 
205 
206 void generalProblem(int N) {
207     double * u;
208     double * y;
209     double * y2;
210 
211     int j = 0;
212 
213     double eps = 0.01, x0 = 0.0;
214 
215     while (N < 100000) {
216         y = new double [N + 1];
217 
218         sweepMethod(y, N, x0);
219 
220         if (j == 1) {
221             if (estimationError(y2, y, N) < eps) {
```

```
222                 for (int i = 0; i < N + 1; i += 50) {
223                     std::cout << "x[" << i << "] = " << i * 1.0 / N << " , y[" <<
224                         i << "] = " << y[i] << std::endl;
225                 }
226 
227                 delete [] y;
228                 delete [] y2;
229 
230                 break;
231             } else {
232                 delete [] y2;
233 
234                 y2 = new double [N + 1];
235 
236                 for (int i = 0; i < N + 1; i++) {
237                     y2[i] = y[i];
238                 }
239 
240                 N = 2 * N;
241             }
242         } else {
243             y2 = new double [N + 1];
244 
245             for (int i = 0; i < N + 1; i++) {
246                 y2[i] = y[i];
247             }
248 
249             N = 2 * N;
250         }
251 
252         delete [] y;
253 
254         j = 1;
255     }
256 }
```

Листинг 1: Исходный код программы