**МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**имени М.В.Ломоносова**

**Факультет вычислительной математики и кибернетики**

**Компьютерный практикум по курсу**

**«ВВЕДЕНИЕ В ЧИСЛЕННЫЕ МЕТОДЫ»**

**ЗАДАНИЕ № 2.**

**Часть 2**

**Численные методы решения дифференциальных уравнений**

**ОТЧЕТ**

**о выполненном задании**

студента 206 учебной группы факультета ВМК МГУ

Мукина Николая Михайловича

Москва, 2014

**Цель работы:**

Освоить метод прогонки решения краевой задачи для дифференциального уравнения 2-го порядка

**Постановка задачи:**

Рассматривается линейное дифференциальное уравнение 2-го порядка вида

y`` + p(x) \* y` + q(x) \* y = f(x) (1)

с дополнительными условиями на граничных точках

k1 \* y(xb) + l1 \* y`(xb) = a

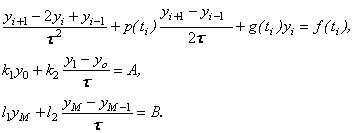
k2 \* y(xe) + l2 \* y`(xe) = b (2)

**Описание метода:**

Для дифференциального уравнения второго порядка

http://pers.narod.ru/algorithms/pas_odu/image001.gif

http://pers.narod.ru/algorithms/pas_odu/image002.gifпоставлена краевая задача: 

http://pers.narod.ru/algorithms/pas_odu/image003.gifЗдесь *u(t) -*искомое решение*, A, B -*краевые условия*,*заданные функции коэффициентов. Введем на отрезке *[a,b]*разностную сетку *(t0, t1, t2, ..., tM,), ti=a+τ\*i, i=0,1,...,M, τ=(b-a)/M, M* – параметр задачи (число шагов сетки). Заменим уравнение разностной схемой   


Собирая коэффициенты при *yi, yi+1 , yi-1,*получим следующую СЛАУ на вектор неизвестных *(y0, y1, y2, ..., yM):*

(k1—k2/τ)y0+ k2 y1/τ =A

AA*i yi-1-*CC*iyi +*BB*iyi+1 =*FF*i, i=1,2,...,M-1*

(l1+l2/τ) yM- l2 yM-1/τ=B,

где **CC***i = 2 - g(ti) τ2 ,*AA*i = 1 - p(ti) τ/2,*BB*i = 1 + p(ti) τ/2,*FF*i = τ2f(ti)*.

Система решается методом прогонки, в котором решение отыскивается в виде *yi=αi+1yi+1+βi+1, i=0,1,…,M-1.*Алгоритм состоит из следующих шагов.

*1.*Поскольку *y0=α1y1+β1,*и из первого уравнения *y0 = (A - k2 y1/τ) / (k1—k2/τ),*то *α1 = - k2/ [ τ (k1—k2/τ) ], β1 = A/(k1—k2/τ)*

*http://pers.narod.ru/algorithms/pas_odu/image006.gifhttp://pers.narod.ru/algorithms/pas_odu/image005.gif2.*Следующие прогоночные коэффициенты *ai, bi* определяются по рекурентным формулам (прямой ход метода прогонки с трёхдиагональной матрицей):

*αi+1 = βi+1 =  i=1,2,...,M-1.*

3. Для определения решения на правой границе отрезка воспользуемся последним уравнением системы *(l1+l2/τ) yM - l2yM-1/τ=B* и соотношением *yM-1=αM yM + βM*, откуда *yM* определяется как решение системы двух линейных алгебраических уравнений.

4. Обратный ход метода прогонки для определения решения.

*yi=αi+1yi+1+βi+1, i = M-1,M-2, ..., 0.*

**Тестирование:**

Вариант 15:

y`` - 3 \* x \* y` + 2 \* y = 1.5

y`(0.7) = 1.3

0.5 \* y(1) + y`(1) = 2

n = 3

|  |  |
| --- | --- |
| x | y |
| 0.7 | 11.910085 |
| 0.8 | 12.040085 |
| 0.9 | 11.948947 |
| 1 | 11.570426 |

n = 10

|  |  |
| --- | --- |
| x | y |
| 0.7 | 4.0376065 |
| 0.73 | 4.0766065 |
| 0.76 | 4.1120645 |
| 0.79 | 4.1437678 |
| 0.82 | 4.1714743 |
| 0.85 | 4.1949092 |
| 0.88 | 4.2137608 |
| 0.91 | 4.2276751 |
| 0.94 | 4.2362506 |
| 0.97 | 4.2390312 |
| 1 | 4.2354987 |

n = 30

|  |  |
| --- | --- |
| x | y |
| 0.7 | 3.6564974 |
| 0.71 | 3.6694974 |
| 0.72 | 3.6821871 |
| 0.73 | 3.694561 |
| 0.74 | 3.7066135 |
| 0.75 | 3.7183386 |
| 0.76 | 3.7297301 |
| 0.77 | 3.7407815 |
| 0.78 | 3.7514861 |
| 0.79 | 3.7618366 |
| 0.8 | 3.7718259 |
| 0.81 | 3.7814461 |
| 0.82 | 3.7906892 |
| 0.83 | 3.7995468 |
| 0.84 | 3.8080101 |
| 0.85 | 3.81607 |
| 0.86 | 3.823717 |
| 0.87 | 3.830941 |
| 0.88 | 3.8377318 |
| 0.89 | 3.8440784 |
| 0.9 | 3.8499696 |
| 0.91 | 3.8553936 |
| 0.92 | 3.860338 |
| 0.93 | 3.86479 |
| 0.94 | 3.8687362 |
| 0.95 | 3.8721626 |
| 0.96 | 3.8750546 |
| 0.97 | 3.8773969 |
| 0.98 | 3.8791738 |
| 0.99 | 3.8803685 |
| 1 | 3.8809636 |

**Вывод:**

Метод прогонки решения краевой задачи позволяет численно решить дифференциальное уравнение второго порядка на отрезке [a; b] с наперед заданной точностью. Порядок точности – О(h2), где h = (b - a) / n, n – параметр программы.

**Исходный код:**

#include <stdio.h>

#include <stdlib.h>

double p(double x){

return ( - 3 \* x);

}

double q(double x){

return (2);

}

double f(double x){

return (1.5);

}

int main(){

double xb = 0.7, xe = 1;

int i, n;

double k1 = 0, k2 = 1, l1 = 0.5, l2 = 1, a = 1.3, b = 2;

printf ("Enter n\n");

scanf ("%d", &n);

double h = (xe - xb) / n;

double \*aa = malloc ((n + 1) \* sizeof(double));

double \*bb = malloc ((n + 1) \* sizeof(double));

double \*cc = malloc ((n + 1) \* sizeof(double));

double \*ff = malloc ((n + 1) \* sizeof(double));

double \*x = malloc ((n + 1) \* sizeof(double));

for (i = 0; i <= n; ++i){

x[i] = xb + h \* i;

aa[i] = 1 - p(x[i]) \* h/2;

bb[i] = 1 + p(x[i]) \* h / 2;

cc[i] = 2 - q(x[i]) \* h \* h;

ff[i] = h \* h \* f(x[i]);

}

double \*al = malloc ((n + 1) \* sizeof(double));

double \*bet = malloc ((n + 1) \* sizeof(double));

double \*y = malloc ((n + 1) \* sizeof(double));

al[1] = k2 / (k2 - k1 \* h);

bet[1] = - (a \* h) / (k2 - k1 \* h);

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

al[i + 1] = bb[i] / (cc[i] - al[i] \* aa[i]);

bet[i + 1] = (aa[i] \* bet[i] - ff[i]) / (cc[i] - al[i] \* aa[i]);

}

y[n] = (l2 \* bet[n] + b \* h) / (l2 + h \* l1 - l2 \* al[n]);

for (i = n-1; i >=0; --i){

y[i] = al[i + 1] \* y[i + 1] + bet[i + 1];

}

for (i = 0; i <= n; ++i){

printf("%5.5lg | %.8lg\n", x[i], y[i]);

}

return 0;

}