Московский Государственный Университет
имени М. В. Ломоносова

Компьютерный практикум по учебному курсу
«ВВЕДЕНИЕ В ЧИСЛЕННЫЕ МЕТОДЫ»

ЗАДАНИЕ № 2
Подвариант №2

«РЕШЕНИЕ КРАЕВОЙ ЗАДАЧИ ДЛЯ ОБЫКНОВЕННОГО ДИФФЕРЕНЦИАЛЬНОГО УРАВНЕНИЯ ВТОРОГО ПОРЯДКА, РАЗРЕШЕННОГО ОТНОСИТЕЛЬНО СТАРШЕЙ ПРОИЗВОДНОЙ»

ОТЧЕТ
о выполненном задании

студента 206 учебной группы факультета ВМК МГУ
Оганисяна Эдгара Гагиковича

Москва, 2019 г.

Цель работы

Освоить метод прогонки решения краевой задачи для дифференциального уравнения второго порядка

Постановка задачи

Рассматривается линейное дифференциальное уравнение второго порядка:

$$y'' + p(x)\cdot y' + q(x)\cdot y = f(x),\ \ 1<x<0,$$

с дополнительными условиями в начальных точках:

$$\begin{cases} \sigma_1 y(0) + \gamma_1 y'(0) = \delta_{1,} \\ \sigma_2 y(0) + \gamma_2 y'(0) = \delta_{2.} \end{cases}$$

Цели и задачи практической работы

1) Решить краевую задачу методом конечных разностей, аппроксимировав ее разностной схемой второго порядка точности (на равномерной сетке); полученную систему конечно-разностных уравнений решить методом прогонки;

2) Найти разностное решение задачи и построить его график;

3) Найденное разностное решение сравнить с точным решением дифференциального уравнения.

Описание метода решения

Сначала строим равномерную сетку с шагом *h:* $x_i = x_0 + i \cdot h, \quad h = \dfrac{b-a}{n}$

Заменяем производные на разностные формулы:

$$y' = \frac{y_{i+1} - y_{i-1}}{2h} \qquad y'' = \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2}$$

После преобразования этих формул получим систему уравнений:

$$\begin{cases} A_i = 1 - p(x_i) \cdot \dfrac{h}{2} \\ B_i = 1 + p(x_i) \cdot \dfrac{h}{2} \\ C_i = 2 - q(x_i) \cdot \dfrac{h}{2} \\ F_i = f(x_i) \cdot h^2 \\ A_i y_{i-1} - C_i y_i + B_i y_{i+1} = F_i, \quad i = 1,2,\dots,n-1 \end{cases}$$

Она содержит *n-1* неизвестных, а матрица данной система является трехдиагональной, следовательно можем решить ее методом прогонки. Решения ищем рекуррентно: $y_i = \alpha_{i+1} y_{i+1} + \beta_{i+1}, \ \ 0 \le i \le n-1$, где α *и* β — прогоночные коэффициенты, которые мы находим по рекуррентным формулам:

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

$$\alpha_1 = 0, \quad \beta_1 = q_{0,} \quad y_n = q_n$$

Остальные значения y_i находятся по указанной выше формуле.

Описание и листинг программы

Т.к программа достаточна велика, здесь приведем пояснения ко всем функциям. Текст программы будет доступен в приложении.

- В программе присутствуют две основные функции:

```
void alpha_beta_search(double *alpha, double *beta,
double a, double h, double s1, double g1, double d1,
double (*p)(double), double (*q)(double),
double (*f)(double), int n);
```

Данная ф-ция вычисляет коэффициенты ***α, β :*** $y_i = \alpha_{i+1} y_{i+1} + \beta_{i+1},\ \ 0 \le i \le n-1$

```
void sweep_method(double *y, double *alpha, double *beta,
double s2, double g2, double d2, double h, int n);
```

Данная ф-ция уже зная коэффициенты ***α, β*** *находит **y** методом прогонки.*

Имена всех параметров ф-ции соответствуют их значениям в теоретических расчетах представленных в описании метода и постановке задачи

- Вспомогательные / тестовые функции:

```
double f1(double x) {return 1;}
double p1(double x) {return 0;}
double q1(double x) {return 1;}
double f2(double x) {return 2.*x;}
double p2(double x) {return 0;}
double q2(double x) {return -1;}
double f3(double x) {return 0;}
double p3(double x) {return -1;}
double q3(double x) {return 0;}
```

- Точные решения дифференциальных уравнений:

```
double y1_exac(double x) {return 1 - sin(x) - cos(x);}
double y2_exac(double x) {return sinh(x)/sinh(1) - 2*x;}
double y3_exac(double x) {return pow(2.7182818284, x)-2;}
```

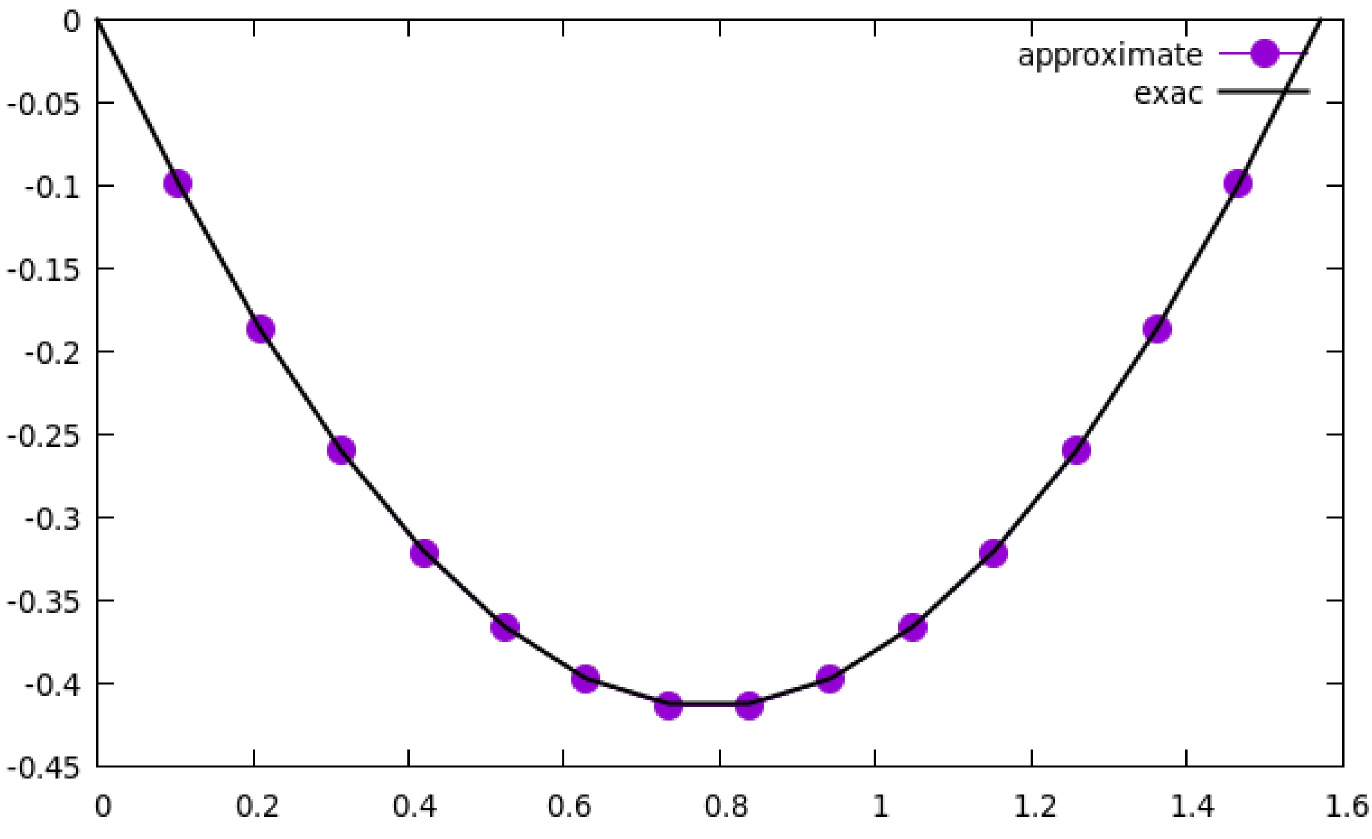
Тесты

Результаты тестов будут представлены в виде графиков с приближенными и точными решениями

- **Тест №1**

  $y'' + y = 1;\ y(0) = 0;\ y(\pi/2) = 0$ Решение: $y = 1 - \sin(x) - \cos(x)$

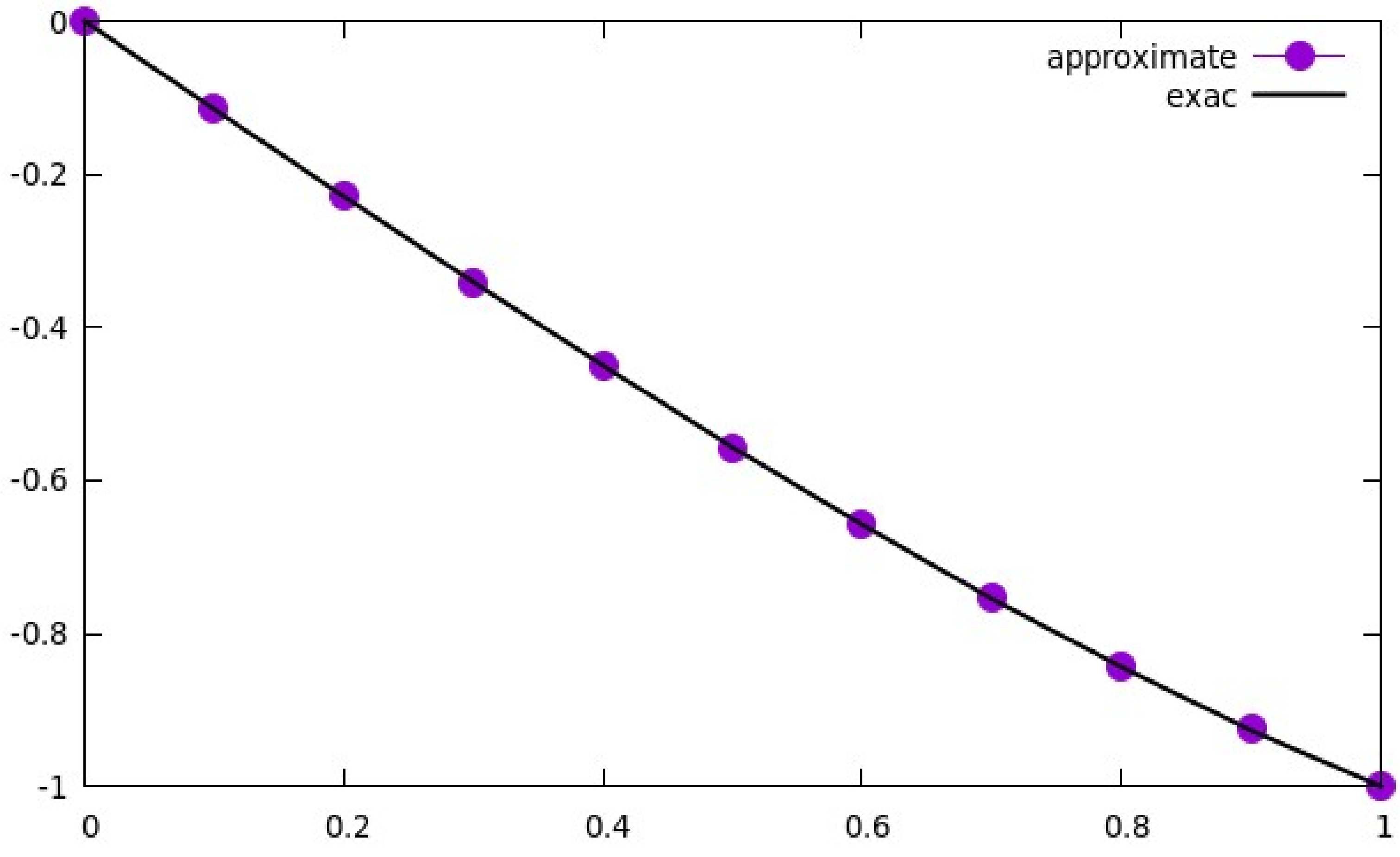
x	**approximate y**	**real y**
0.000	0.000	0.000
0.105	-0.099	-0.099
0.209	-0.186	-0.186
0.314	-0.260	-0.260
0.419	-0.321	-0.320
0.524	-0.366	-0.366
0.628	-0.397	-0.397
0.733	-0.413	-0.412
0.838	-0.413	-0.412
0.942	-0.397	-0.397
1.047	-0.366	-0.366
1.152	-0.321	-0.320
1.257	-0.260	-0.260
1.361	-0.186	-0.186
1.466	-0.099	-0.099
1.571	0.000	-0.000



- Тест №2

$$y'' - y = 2x;\ \ y(0) = 0;\ \ y(1) = -1 \quad \text{Решение:} \quad y = \frac{sh(x)}{sh(1)} - 2x$$

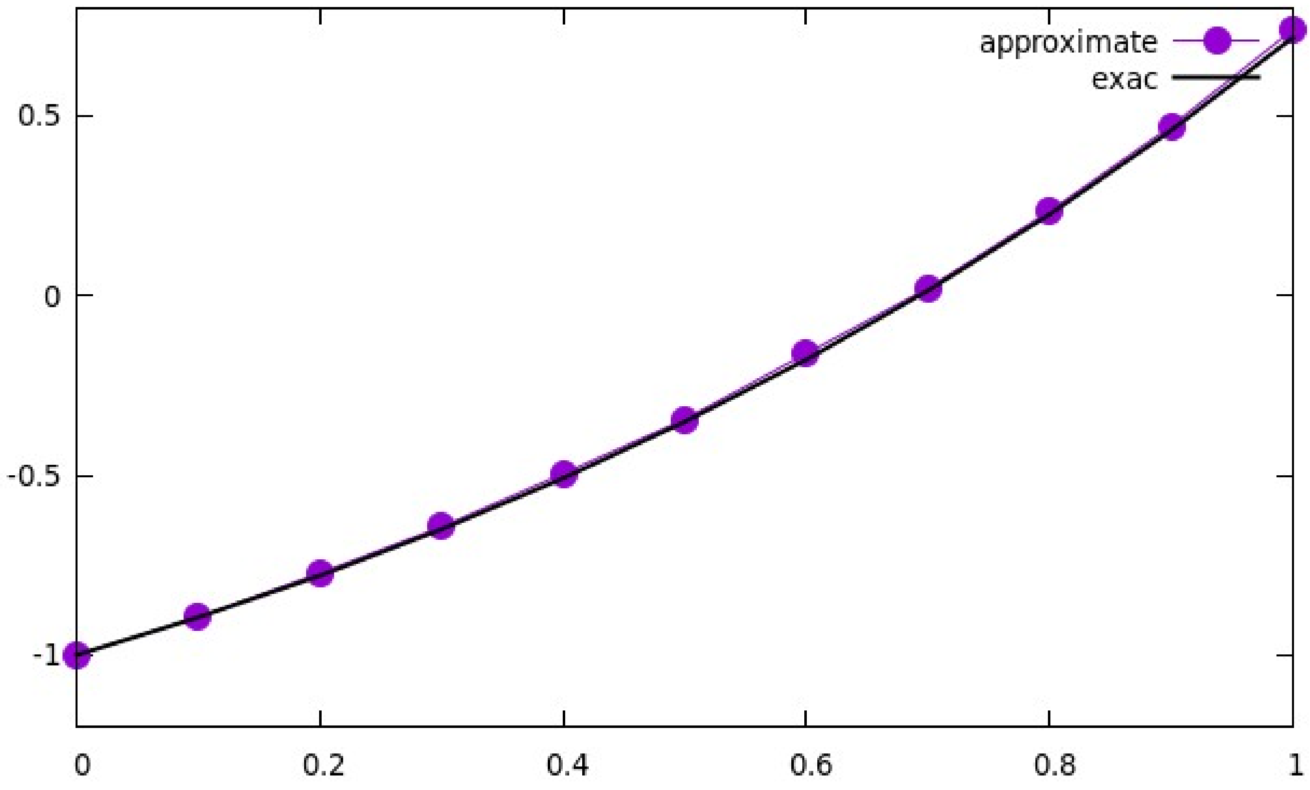
x	**approximate y**	**real y**
0.000	0.000	0.000
0.100	-0.115	-0.115
0.200	-0.229	-0.229
0.300	-0.341	-0.341
0.400	-0.450	-0.450
0.500	-0.557	-0.557
0.600	-0.658	-0.658
0.700	-0.754	-0.755
0.800	-0.844	-0.844
0.900	-0.926	-0.927
1.000	-1.000	-1.000



- Тест №3

$y'' - y' = 0;\ \ y(0) = -1;\ \ y'(1) - y(1) = 2$ Решение: $y = e^x - 2$

x	approximate y	real y
0.000	-0.100	-1.000
0.100	0.130	-0.895
0.200	0.384	-0.779
0.300	0.664	-0.650
0.400	0.975	-0.508
0.500	1.318	-0.351
0.600	1.696	-0.178
0.700	2.115	0.014
0.800	2.578	0.226
0.900	3.090	0.460
1.000	3.656	0.718



Выводы

Был освоен метод прогонки решения краевой задачи для дифференциального уравнения второго порядка. Экспериментально показана высокая точность вычислений.

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

double pi = 3.1415926535;
double e = 2.7182818284;

// y"+y=1, y(0) = 0, y(pi/2) = 0
double f1(double x) {return 1;}
double p1(double x) {return 0;}
double q1(double x) {return 1;}
double koef_data_1[6] = {1, 0, 0, 1, 0, 0}; // sigma1 gamma1 delta1 / sigma 2 gamma2 delta2
double ab_data_1[2] = {0, 3.1415926535/2};
double y1_exac(double x) {return 1 - sin(x) - cos(x);} // y = 1 -sinx - cosx

// y"-y=2x, y(0) = 0, y(1) = -1
double f2(double x) {return 2.*x;}
double p2(double x) {return 0;}
double q2(double x) {return -1;}
double koef_data_2[6] = {1, 0, 0, 1, 0, -1};
double ab_data_2[2] = {0, 1};
double y2_exac(double x) {return sinh(x)/sinh(1) - 2*x;} // y = sh(x)/sh(1) - 2x

// y"-y'=0, y(0) = -1, y'(1) - y(1) = 2
double f3(double x) {return 0;}
double p3(double x) {return -1;}
double q3(double x) {return 0;}
double koef_data_3[6] = {1, 0, -1, -1, 1, 2};
double ab_data_3[2] = {0, 1};
double y3_exac(double x) {return pow(2.7182818284, x) - 2;} // y = e^x - 2

double ab_data[6] = {0, 3.1415926535/2, 0, 1, 0, 1}; // common data
double koef_data[18] = {1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, -1, 1, 0, -1, -1, 1, 2}; // common koef

double ((*function_f[3]))() = {f1, f2, f3};
double (*function_p[3])() = {p1, p2, p3};
double (*function_q[3])() = {q1, q2, q3};
double (*function_exac[3])() = {y1_exac, y2_exac, y3_exac};

void sweep_method(double *y, double *alpha, double *beta, double s2, double g2, double d2,
                  double h, int n);

void alpha_beta_search(double *alpha, double *beta, double a, double h, double s1,
                       double g1, double d1, double (*p)(double), double (*q)(double),
                       double (*f)(double), int n);

int main(int argc, char **argv) // на вход номер ф-ции и кол-во иттераций n
{
    int fnum, n;
    sscanf(argv[1], "%d", &fnum); fnum--;
    sscanf(argv[2], "%d", &n);

    double ((*f))()  = function_f[fnum];
    double ((*p))()  = function_p[fnum];
    double ((*q))()  = function_q[fnum];

    double a = ab_data[fnum*2];
    double b = ab_data[fnum*2+1];
    double h = (b-a)/n;

    //sigma gamma delta
    double s1 = koef_data[fnum*6 + 0];
    double g1 = koef_data[fnum*6 + 1];
```

```c
    double d1 = koef_data[fnum*6 + 2];
    double s2 = koef_data[fnum*6 + 3];
    double g2 = koef_data[fnum*6 + 4];
    double d2 = koef_data[fnum*6 + 5];

    // y - solution; alpha, beta - koef
    double *y = calloc(n+1, sizeof(double));
    double *alpha = calloc(n+1, sizeof(double));
    double *beta = calloc(n+1, sizeof(double));

    alpha_beta_search(alpha, beta, a, h, s1, g1, d1, p, q, f, n);
    sweep_method(y, alpha, beta, s2, g2, d2, h, n);

    char name[128];
    sprintf(name, "test_%d.txt", fnum+1);
    FILE *out = fopen(name, "w");

    double ((*exac))()  = function_exac[fnum];
    for(int i = 0; i < n + 1; i++) {
        double x = a + h * i;
        fprintf(out, "%9.3lf %9.3lf %9.3lf\n", x, y[i], exac(x));
    }
    return 0;
}

void alpha_beta_search(double *alpha, double *beta, double a, double h, double s1,
                       double g1, double d1, double (*p)(double), double (*q)(double),
                       double (*f)(double), int n)
{
    alpha[1] = -1. * (g1)  / (s1 - g1); //&
    beta[1] = (d1 * h) / (s1 - g1);
    for(int i = 1; i < n; i++) {
        double x = a + i * h;
        double P = p(x);
        double Q = q(x);
        double F = f(x);

        alpha[i + 1] = (1 / (h*h) + P / (2*h)) /
                ((2./(h*h) - Q) - (1./(h*h) - P/(2.*h)) * alpha[i]);
        beta[i + 1] = (beta[i]*(1/(h*h) - P/(2*h)) - F) /
                ((2./(h*h) - Q) - (1./(h*h) - P/(2.*h)) * alpha[i]);
    }
    return;
}
void sweep_method(double *y, double *alpha, double *beta, double s2, double g2, double d2,
                  double h, int n)
{
    y[n] = (g2 * beta[n] + d2 * h) / (g2 * (1 - alpha[n]) + s2 * h);
    for (int i = n - 1; i >= 0; i--) {
        y[i] = y[i + 1] * alpha[i + 1] + beta[i + 1];
    }
    return;
}
```