Нижегородский государственный университет им. Н. И. Лобачевского

Радиофизический факультет
Кафедра математики

Отчет по лабораторной работе:

Задача Коши для обыкновенных дифференциальных уравнений

Введение в численные методы

Выполнил: студент 421 группы **Зайцев Юрий**

Проверил: **Кулинич Виктор Валентинович**

Нижний Новгород
2006 год

Содержание

Введение

В настоящей работе анализируется процесс решения задачи Коши для заданного обыкновенного дифференциального уравнения с интегралом в правой части с помощью численных методов на персональном компьютере. Для этого используются два стандартных численных метода решения обыкновенных дифференциальных уравнений: двукратный метод Рунге-Кутты – метод Хойна и метод Рунге-Кутты-Фельберга 4-5 порядка (RKF-45). Кроме того, оба полученных численными методами решения сравниваются с аналитическим. В конце работы произведён общий анализ полученных результатов.

1 Постановка учебно-практической задачи

Целью данной лабораторной работы являлось решение задачи Коши для обыкновенного дифференциального уравнения с интегралом в правой части

$$\begin{cases} y'' - 2y' + y = 2xe^x + \int\limits_0^x e^\xi \sin 2\xi d\xi \\ y(0) = 1 \\ y'(0) = 0 \\ x \in [0, 2] \end{cases} \tag{1}$$

Для написания программы решения уравнения (1) методом Хойна и RKF-45 использовать математический пакет Matlab. Решение методом Хойна должно быть реализовано самостоятельно, в то время как для решения методом RKF-45 можно воспользоваться стандартной программой из пакета Matlab. Интеграл в правой части должен быть взят при помощи квадратурной формулы Симпсона.

Результаты решения представить в графической форме (явная зависимость $y(x)$ и фазовая плоскость), кроме того, для заданного уравнения провести теоретическое исследование устойчивости и порядка аппроксимации заданного метода численного решения.

2 Описание алгоритма

Итак, уравнение (1) требуется решить методом Хойна и RKF-45, а также сравнить полученные решения. Прежде чем решать наше уравнение 2-го порядка, нужно привести его к системе первого порядка в нормальной форме. Для этого делается следующая замена:

$$\begin{cases} u_1(x) = y(x) \\ u_2(x) = y'(x) \end{cases}$$

В результате получаем эквивалентную систему:

$$\begin{cases} u_1' = u_2 \\ u_2' = -u_1 + 2u_2 + 2xe^x + \int\limits_0^x e^\xi \sin 2\xi d\xi \\ u_1(0) = 1 \\ u_2(0) = 0 \\ x \in [0, 2] \end{cases} \tag{2}$$

Если ввести обозначение $g(x) = -u_1 + 2u_2 + 2xe^x + \int\limits_0^x e^\xi \sin 2\xi d\xi$, получим, в итоге, следующую систему (при тех же начальных условиях и на том же интервале решения, что и в (2)):

$$\begin{cases} \dfrac{du_1}{dx} = u_2(x) \\ \dfrac{du_2}{dx} = g(x) \end{cases}$$

После этого способ распространения метода решения ОДУ первого порядка на уравнения более высоких порядков (приводимых к системе уравнений первого порядка при помощи правильной параметризации) становится вполне очевидным. На нем мы подробно и остановимся ниже, сначала рассмотрев вопрос о численном нахождении интеграла в правой части.

2.1 Вычисление интеграла по формуле Симпсона

Задача вычисления интеграла Римана является одной из старейших задач теории численных методов и состоит в замене вычисления интеграла

$$I = \int\limits_\alpha^\beta f(x)dx$$

неким алгоритмом из конечного числа операций. В нашем случае, её решение сводится к применению т.н. квадратурной формулы, позволяющей предварительно разбив интервал интегрирования $[\alpha, \beta]$ на сетку $\omega_n = \{x_i^{(n)}, \alpha \le x_1 \le x_2 \le \cdots \le x_n \le \beta\}$ и просуммировав значения функции $f(x_i)$ с определенными весами m_i получить приближение

$$I \approx I_n = \sum_{i=1}^{n} m_i f(x_i), \tag{3}$$

отличающееся от точного значения I не более чем на наперед заданную малую константу ε.

Процесс получения использованной в данной работе квадратурной формулы Симпсона подробно рассмотрен в методическом пособии [2] (он основан на идее локально-параболической аппроксимации подинтегральной функции), поэтому ограничимся

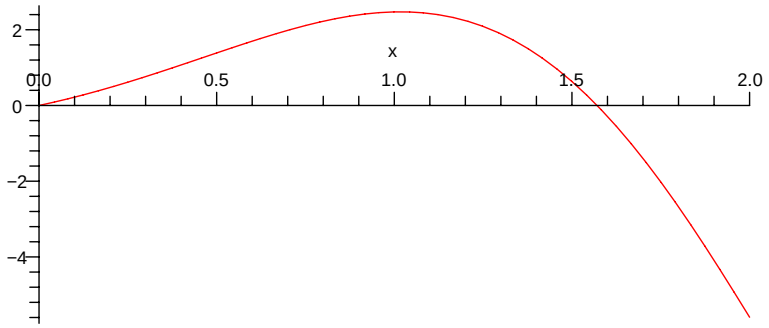

Рис. 1: Подинтегральная функция $p(\xi) = e^{\xi} \sin 2\xi$, $\xi \in [0, 2]$

рассмотрением её применимости в нашем конкретном случае и приведем использованные значения ε и максимальную потребовавшуюся частоту сетки n.

Итак, нам необходимо вычислить интеграл

$$I = \int_0^x e^{\xi} \sin 2\xi d\xi, \text{ где } x \in [0, 2] \tag{4}$$

при помощи формулы Симпсона на равномерной сетке ω_n

$$\begin{aligned} I = \int_{\alpha}^{\beta} f(x)dx = \frac{h}{3}(f(\alpha) + 4f(\alpha + h) + 2f(\alpha + 2h) + \cdots + \\ + 2^{1\ +\ n \bmod 2} f(\alpha + (n-2)h) + f(\beta)) + R_n[f(x)], \end{aligned} \tag{5}$$

остаточный член которой оценивается как

$$R_n[f(x)] \leq \frac{h^4}{180}(\beta - \alpha) \max_{[\alpha,\beta]} |f^{(4)}(x)|.$$

Подинтегральная функция $p(\xi) = e^{\xi} \sin 2\xi$ в интервале $\xi \in [0, 2]$ изображена на Рис. 1. Эта функция непрерывна и имеет бесконечное число непрерывных производных. Например, график её четвертой производной в исследуемом интервале изображен на Рис. 2. Как известно из курса математического анализа, в ограниченной области такие функции удовлетворяют условию Липшица и не имеют особенностей. Поэтому очевидно, что при $n \to \infty$ и $h \to 0$ остаток $R_{\infty}[p(\xi)] = \lim\limits_{h \to 0} \dfrac{h^4}{180}(2 \times 157) = 0$ стремится к нулю.

Таким образом применимость формулы Симпсона к вычислению интеграла (4) на заданном интервале доказана. Точность вычисления контролировалась по правилу Рунге (т.е. сравнивались значения, полученные при сетке с шагом h и $h/2$) и

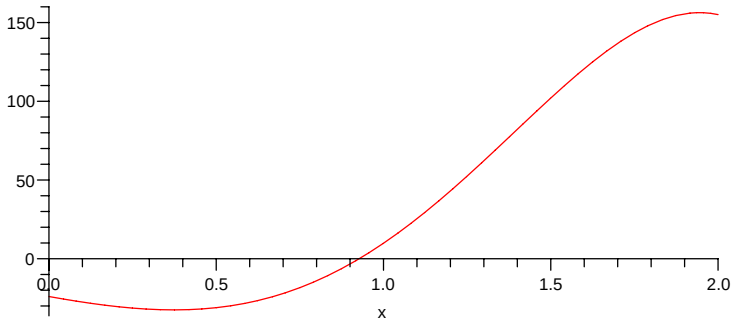

Рис. 2: Четвертая производная $p^{(4)}(\xi) = -7e^{\xi}\sin 2\xi - 24e^{\xi}\cos 2\xi$, $\xi \in [0, 2]$

при выполнении условия $|I_h - I_{h/2}| \leq \varepsilon_i$ вычисление прекращалось. В данной работе для обеспечения $\varepsilon_i = 10^{-5}$ наибольшее потребовавшееся кол-во вычислений функции $f(x)$ было $n = 81$. Воспользовавшись этими данными, можно провести оценку остаточного члена $R_n[f]$: в нашем случае, $R_n[f] \leq 6.8 \times 10^{-7}$.

2.2 Метод Хойна

Пусть поставлена задача Коши

$$\begin{cases} y'(x) = f(x, y(x)), \ x \in (a, b) \\ y(a) = y^0 \end{cases}$$

Для построения приближенного решения в точке x_{n+1} сетки ω_n, разложим в ряд Тейлора решение в предыдущей точке сетки x_n по степеням шага h:

$$y(x_{n+1}) = y(x_n) + h\Delta(x_n, y_n, h),$$
$$\Delta(x, y, h) \equiv y'(x) + \frac{h}{2}y''(x) + \frac{h^2}{3!}y'''(x) + \ldots,$$

и ограничимся учетом первых p членов этого ряда:

$$\varphi_p(x, y, h) \equiv y'(x) + \frac{h}{2}y''(x) + \cdots + \frac{h^{p-1}}{p!}y^{(p)}(x).$$

Рассмотрим метод Хойна. Он является одним из методов семейства Рунге-Кутты. Идея этих методов заключается в построении функций φ, которые не используют явно производных функции $f(x, y)$ и, в то же время, “наилучшим образом” приближают соответствующие отрезки φ_p ряда Тейлора Δ. Если ограничиться членами до h^2, получим

$$\varphi(x, y, h) \equiv c_1 f(x, y) + c_2(f(x + ha_2, y + b_{21}hf(x, y)),$$

где c_1, c_2, a_2, b_{12} - произвольные постоянные подлежащие определению. Определив их, разлагая φ по степеням h до членов порядка h^2 и сравнивая полученное разложение с рядом Тейлора Δ, придем к системе со свободным параметром, зафиксировав который получим двукратный метод Рунге-Кутты - *метод Хойна*:

$$\begin{cases} y_{n+1} = y_n + \frac{h}{2}[f(x_n, y_n) + f(x_{n+1}, y_n + hf(x_n, y_n))], \\ y_0 = y^0, n = \overline{0, N-1}, x_n \in \omega_n \end{cases}$$

Распространив его на нашу систему уравнений (2) получим формулы, которые непосредственно были заложены в программу:

$$\begin{cases} u_1^{n+1} = u_1^n + \frac{h}{2}[u_2^n + u_2^{n+1}] = \\ \qquad = u_1^n + \frac{h}{2}(u_2^n + u_2^n + \frac{h}{2}[g(x^n, u_1^n, u_2^n) + g(x^{n+1}, u_1^n + hu_2^n, u_2^n + hg(x^n, u_1^n, u_2^n))]) \\ u_2^{n+1} = u_2^n + \frac{h}{2}[g(x^n, u_1^n, u_2^n) + g(x^{n+1}, u_1^{n+1}, u_2^n + hg(x^n, u_1^n, u_2^n))] \\ u_1^1 = 1 \\ u_2^1 = 0 \\ h = \dfrac{b-a}{N} \end{cases}$$

Точность решения, как и при взятии определенного интеграла контролировалась по правилу Рунге. При решении на каждом шаге вычислялось максимальное отклонение компонент векторов решений $\overrightarrow{Y_N}$ и $\overrightarrow{Y_{N/2}}$ (см. файл *deviation.m*) $d = \max\limits_{i=1..N/2} |Y_N^{2i-1} - Y_{N/2}^i|$, и если оно превосходило наперед заданную постоянную ε, программа вдвое увеличивала число точек разбиения сетки ω_n и переходила к следующей итерации цикла. Определявшая точность решения уравнения константа ε_d была выбрана равной $\varepsilon_d = 5 \times 10^{-3}$.

2.3 Метод Рунге-Кутты 4-5 порядка

ODE45 - это встроенная в Matlab функция, решающая задачу Коши методом Рунге-Кутты 4-5 порядка. Данная функция была использована нами для сравнения результатов решения системы этим методом и методом, предлагаемым в задании.

2.4 Аналитическое решение

Решая уравнение (1) аналитически как обыкновенное дифференциальное уравнение второго порядка, т.е. сначала найдя общее решение однородного уравнения, затем – частное неоднородного уравнения и суммируя их с учетом начальных условий, получим следующее выражение для явной зависимости $y(x)$:

$$y(x) = \frac{3}{5}e^x - \frac{2}{5}xe^x + \frac{2}{5} + \frac{1}{30}(10x^3 - 6 + 6\cos^2 x - 3\sin x \cos x - 3x)e^x. \qquad (6)$$

3 Исследование применимости алгоритма

3.1 Теоретическое исследование устойчивости метода

Дадим определение устойчивости и приведем критерий устойчивости для нашей двухслойной разностной схемы. Под *двухслойной схемой* понимают разностное уравнение, связывающее значение вектора $y(x)$ для двух слоёв значений аргументов $x = x_n$ и $x = x_{n+1}$. В канонической форме это соотношение задают в виде:

$$\begin{cases} B\dfrac{y_{n+1} - y_n}{h} + Ay_n = \varphi_n \\ n = 0, 1, 2, \ldots, y_0 = u_0 \end{cases}, \tag{7}$$

где B, A - квадратные матрицы $s \times s$, а y_n и φ_n - векторы размерности s. Если $B = E$, то каноническую схему называют *явной*, в противном случае - *неявной*. В нашем случае

$$\begin{cases} \dfrac{y^1_{n+1} - y^1_n}{h} + 0 + 0 - y^2_n = \frac{1}{4}h(g_n + g_{n+1}) \\ 0 + \dfrac{y^2_{n+1} - y^2_n}{h} + 0 + 0 = \frac{1}{2}(g_n + g_{n+1}) \end{cases} \tag{8}$$

матрицы A и B соответственно равны

$$B = E = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad A = \begin{pmatrix} 0 & -1 \\ 0 & 0 \end{pmatrix}.$$

Определение. *Каноническую схему назовем* ***устойчивой по начальным данным и правой части****, если существуют положительные постоянные M_1, M_2, не зависящие от h, n, y_0, φ_n, такие, что для решения y_n такой задачи имеет место оценка*

$$\|y_n\| \le M_1\|y_0\| + M_2 \max_{0 \le k \le n-1} \|\varphi_k\|.$$

Определение. *Каноническую схему назовем* ***равномерно устойчивой по начальным данным****, если существует положительная постоянная ρ, не зависящая от h, n, y_0, такая, что для решения однородной задачи*

$$\begin{cases} B\dfrac{y_{n+1} - y_n}{h} + Ay_n = 0 \\ n = 0, 1, 2, \ldots, y_0 = u_0, \end{cases}$$

имеет место оценка

$$\|y_{n+1}\| \le \rho\|y_n\|, n = 0, 1, \ldots.$$

Можно показать, что из равномерной устойчивости схемы по начальным данным вытекает её устойчивость по начальным данным и правой части с постоянными M_1 и M_2, вычисляемыми через постоянную ρ.

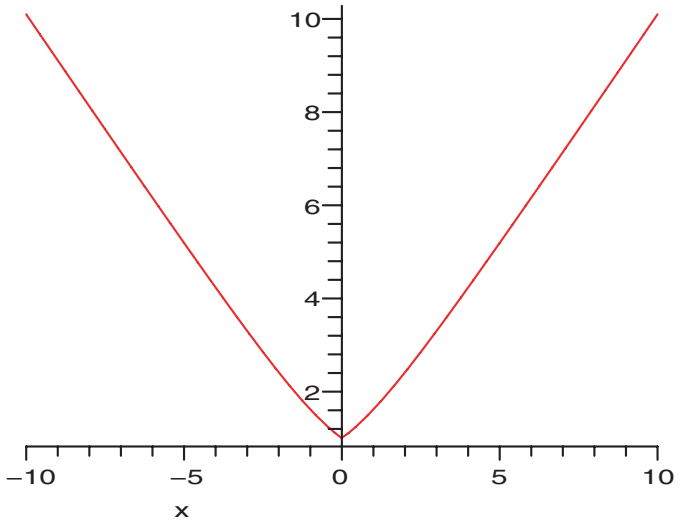

Рис. 3: Зависимость нормы оператора перехода от h

Если переписать это уравнение в операторной форме

$$y_{n+1} = Sy_n,$$

где $S \equiv E - hB^{-1}A$, то нетрудно видеть, что введенное выше условие равномерной устойчивости схемы по начальным данным эквивалентно условию ограниченности нормы оператора перехода:

$$\|S\| \leq \rho.$$

В нашем случае

$$\|S\| = \left\| \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - h \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}^{-1} \begin{pmatrix} 0 & -1 \\ 0 & 0 \end{pmatrix} \right\| = \left\| \begin{matrix} 1 & h \\ 0 & 1 \end{matrix} \right\|$$

$$\|S\|_2 = \frac{\sqrt{4 + 2\,h^2 + 2\sqrt{4h^2 + h^4}}}{2} \backsim h \text{ при } h \to \infty, \text{ и } \|S\| \to 1 \text{ при } h \to 0.$$

Т.е. для любого выбранного h $\|S\|$ ограничена и для любого выбранного постоянного ρ равномерная устойчивость с постоянной $\rho > 1$ гарантируется при выполнении следующего условия для h:

$$h \leq \frac{\rho^2 - 1}{\rho}.$$

3.2 Теоретическое исследование порядка аппроксимации метода

Пусть $u(x)$ – точное решение поставленной задачи. y_n – решение, полученное при помощи канонической схемы (7). Тогда величину $z_n \equiv y_n - u(x_n)$ назовем *погрешностью решения*. Поставляя в схему y_n получим уравнения для погрешности решения

$$\begin{cases} B\dfrac{z_{n+1}-z_n}{h} + Az_n = \psi_n \\ n = 0, 1, 2, \ldots, z_0 = 0, \end{cases}$$

где

$$\psi_n \equiv \varphi_n - Au(x_n) - B\frac{u(x_{n+1}) - u(x_n)}{h}$$

– *погрешность аппроксимации (невязка)* для схемы (7).

Определение. *Схема имеет m-ый порядок аппроксимации на решении поставленной задачи, если для невязки ψ_n выполняется оценка*

$$\|\psi_n\| = O(h^m).$$

Опустив подробный вывод условия аппроксимации и сходимости приближенного решения, рассмотренный в [1], приведем условия, при выполнении которых каноническая схема имеет m-ый порядок аппроксимации:

$$\begin{aligned} &\left\|\varphi_n - \varphi(x_{n+\frac{1}{2}})\right\| = O(h^m), m = \overline{1,2}, \\ &\left\|\left(E - B + \frac{h}{2}A\right)\frac{du}{dx}\right\| = O(h^m), m = \overline{1,2}. \end{aligned} \tag{9}$$

Для нашей *чисто явной схемы* (8), т.е. при $B = E$, условие (9) вырождается в следующее:

$$\left\|A\frac{du}{dx}\right\|\frac{h}{2} = O(h),$$

поэтому, если

$$\|\varphi_n - \varphi(x_{n+\frac{1}{2}})\| = O(h),$$

то такая схема имеет 1-ый порядок аппроксимации. Таким образом, порядок аппроксимации использованной схемы установлен.

4 Результаты расчетов и заключение

Результаты расчетов программы приведены на Рис. 4 и Рис. 5. На Рис. 4 изображена явная зависимость $y(x)$, а на Рис. 5 представлена фазовая плоскость, т.е. зависимость $\dot{y}(y(x))$. На Рис. 4 сплошной линией изображено решение $y(x)$, полученное при помощи расчетов по методу Хойна, красными крестиками изображено решение,

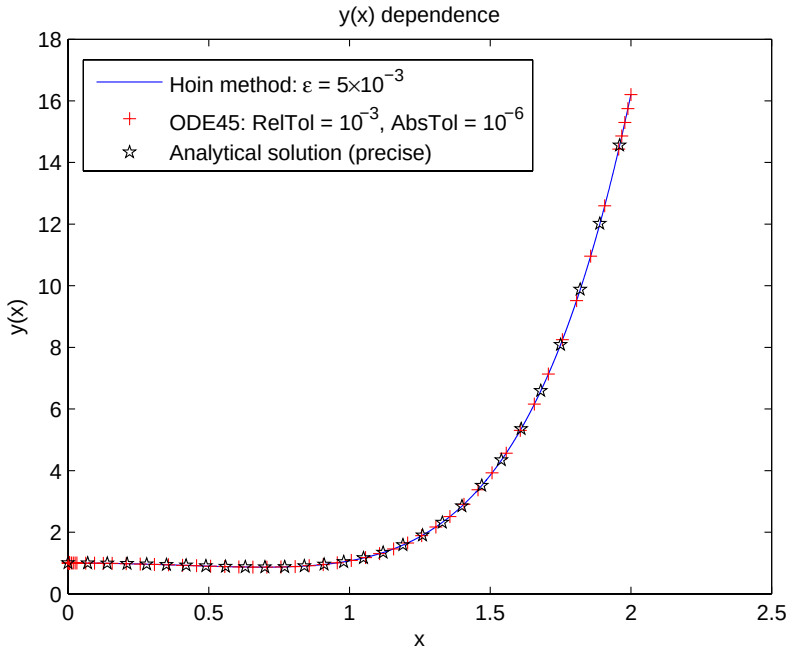

Рис. 4: Полученный график явной зависимости $y(x)$.

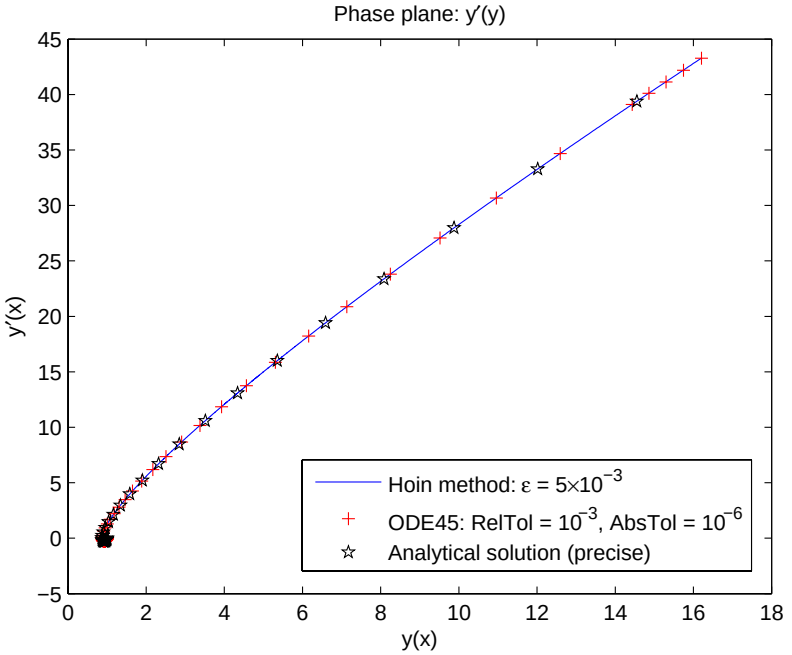

Рис. 5: Фазовая плоскость $\dot{y}(y(x))$.

полученное при помощи метода Рунге-Кутты 4-5 порядка (функция ODE45 пакета Matlab) и, наконец черными звездочками — график аналитического решения (6) соответственно. Аналогично на Рис. 5 изображена фазовая плоскость решения полученная по методу Хойна, RKF-45 и аналитическим способом.

Можно отметить отличное совпадение решений, полученных при помощи численных методов с решением, полученным аналитически, а также самих численных между собой. Таким образом, оба рассмотренных численных метода хорошо находят решение заданной системы и могут использоваться для нахождения решения в рамках поставленной нами задачи.

5 Приложения

В приложениях приводятся исходные тексты программы, написанной на языке Matlab. Программа построена из нескольких модулей, управляемых головной программой *dsolve.m*. Главная часть программы — *dsolve.m* — инициализирует параметры, вызывает необходимое число раз модуль *hoin.m*, который занимается непосредственно решением системы (2) методом Хойна, находит решение при помощи солвера ODE45, вычисляет $y(x)$ и $y'(x)$, а также строит графики зависимостей $y(x)$ и $\dot{y}(y(x))$.

5.1 dsolve.m

```
clear all

N = 1000; % Initial grid

x1 = 0; % X - starting point
x2 = 2; % X - ending   point

u10 = 1; % y (0) = 1
u20 = 0; % y'(0) = 0

global epsi
global ki

ki = 0;
epsd = 5e-3;
epsi = 1e-5;

fprintf('\n\nHoin solver\n\n');
fprintf('Dsolve precision: %d\n', epsd);
fprintf('Integration precision: %d\n', epsi);
fprintf('Solving, please stand by...\n');
```

```
k = 0;

r = hoin(N, x1, x2, epsi);
while (1)
    k = k + 1;

    rp = hoin(N*2, x1, x2, epsi);

    y1 = r(:,2);
    y2 = rp(:,2);

    d = deviation(y1, y2);
    if (d < epsd)
        break
    else
        N = N*2;
        r = rp;
    end
end

fprintf('Finished! Number of passes k = %i, achieved precision
   d = %f, grid N = %i.\n', k, d, N);

x  = rp(:,1); y  = y2; yp = rp(:,3);

% Phase plane
figure(1);
plot(y,yp);
title('Phase plane: y\prime(y)'); xlabel('y(x)'); ylabel('y\
   prime(x)');

hold on

% RKF45-based solver
[T,Y] = ode45(@rkff,[x1 x2],[u10 u20]);
plot(Y(:,1),Y(:,2),'+r')

% Analytical solution
XG = x1:0.07:x2;
AY  = 3/5 .* exp(XG) - 2/5 .* XG .* exp(XG) + 2/5 + 1/30 .*(10
   .* XG .^ 3 - 6 + 6 .* cos(XG) .^ 2 - 3 .* sin(XG) .* cos(XG)
    - 3 .* XG) .* exp(XG);
```

```
AYP = 1/5 .* exp(XG) - 2/5 .* XG .* exp(XG) + 1/30 .* (30 .* XG
    .^ 2 - 12 .* sin(XG) .* cos(XG) - 3 .* cos(XG) .^ 2 + 3 .*
   sin(XG) .^ 2 - 3) .* exp(XG) + 1/30 .* (10 .* XG .^ 3 - 6 +
   6 .* cos(XG) .^ 2 - 3 .* sin(XG) .* cos(XG) - 3 .* XG) .*
   exp(XG);
plot(AY, AYP, 'pk')

legend('Hoin method: \epsilon = 5\times10^{-3}','ODE45: RelTol
   = 10^{-3}, AbsTol = 10^{-6}','Analytical solution (precise)'
   );

hold off

% Graph
figure(2);
plot(x,y);
title('y(x) dependence'); xlabel('x'); ylabel('y(x)');

hold on

plot(T,Y(:,1),'+r')
plot(XG, AY, 'pk')

legend('Hoin method: \epsilon = 5\times10^{-3}','ODE45: RelTol
   = 10^{-3}, AbsTol = 10^{-6}','Analytical solution (precise)'
   );

hold off

ki
```

5.2 hoin.m

```
function r = hoin(N, x1, x2, epsi)

h  = (x2 - x1)/N; % Grid step

k       = 1;
u1(k)   = 1;
u2(k)   = 0;
x(k)    = x1;
x(k + 1) = x1 + h;
```

```
while(k <= N)
%     u1(k + 1) = u1(k) + h*u2(k);
      u1(k + 1) = u1(k) + h/2*(u2(k) +  u2(k) + h/2*(right(x(k),
         u1(k), u2(k)) + right(x(k + 1), u1(k) + h*u2(k), u2(k) +
          h*right(x(k), u1(k), u2(k))))    );
      u2(k + 1) = u2(k) + h/2*(right(x(k), u1(k), u2(k)) + right(
         x(k + 1), u1(k + 1), u2(k) + h*right(x(k), u1(k), u2(k))
         ));

      x(k + 2)  = x(k + 1) + h;

      k = k + 1;
end

x(k + 1) = []; % Delete last element - stupid ML can't plot
    otherwise

h

r = [x' u1' u2'];
```

5.3 deviation.m

```
function r = deviation(a, b)

% First vector wil be shortest
if (size(a) > size(b))
      c = a; a = b; b = c;
end

% Build c - odds from b (longest)
n = 1;
for i = 1:size(b)
      if (mod(i, 2) == 1)
            c(n) = b(i);
            n = n + 1;
      end
end

r = max(abs((c' - a)'));
```

5.4 rkff.m

```
function z=rkff(x,t);
```

```
z = zeros(2,1);
z(1) = t(2);
z(2) = 2*t(2) - t(1) + 2 * x * exp(x) + integral(x);
```

5.5 right.m

```
function r = right(x, y, yp)

r = 2 * x * exp(x) + integral(x) - y + 2 * yp;
%r = + y*500 + yp/10;
```

5.6 integral.m

```
function r = integral(x2)

global epsi

N = 20;       % Initial integration grid
x1 = 0;       % Int(f(x), x = 0 .. x2);
eps = epsi;   % Precision

i = simpson(N, x1, x2);
while (1)

    ip = simpson(N*2, x1, x2);

    if (abs(ip-i) < eps)
        break
    else
        N = N*2;
        i = ip;
    end
end

r = ip;
```

5.7 simpson.m

```
function r = simpson(N, x1, x2)
global ki

h = (x2 - x1)/N;
```

```
k = 0;
sum = 0;
xcr = x1;
while (k <= N-2)
    k = k + 1;
    xcr = xcr + h;
    m = 2^(mod(k, 2) + 1);
    sum =  sum + m*fi(xcr);
end
sum = (sum + (fi(x1) + fi(x2))) * h / 3;

if (k > ki)
        ki = k;
end

r = sum;
%r = quad(@fi, x1, x2);
```

5.8 fi.m

```
function f = fi(x);

f = exp(x) .* sin(2*x);
```

Список литературы

[1] Кулинич В.В., Смирнов И.П. Задача Коши для обыкновенных дифференциальных уравнений. Введение в численные методы. Лабораторная работа для студентов радиофизического факультета ННГУ. Н. Новгород: изд. ННГУ им. Лобачевского, 2002 г. 32 с.

[2] Квадратурная формула. Лабораторная работа для студентов 2 курса дневого отделения радиофизического факультета. Горький: Типография ГГУ, 1990. 22 с.

[3] Калиткин Н.Н., Численные методы. М.: Наука. 1978. 512 с.

[4] Самарский А. А., Введение в численные методы. М.: Наука. 1987. 288 с.