**Министерство науки и высшего образования Российской Федерации**

федеральное государственное автономное образовательное учреждение

высшего образования

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Инженерная школа природных ресурсов

Направление подготовки Химическая технология

Отделение химической инженерии

**Численное решение систем дифференциальных уравнений на примере кинетики химических реакций**

Выполнила студентка гр. 2Д93 Филатова О.В.

(Подпись)

14.06. 2020 г.

Отчет принят:

Преподаватель

доцент ОХИ ИШПР, к.т.н. Чузлов В.А.

(Подпись)

\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

Томск 2020 г.

**Цель работы:** изучить численное решение систем дифференциальных уравнений на примере кинетики химических реакций.

**Теоретическая часть**

Метод Эйлера

Пусть дано дифференциальное уравнение:



с начальным условием:



Формула Эйлера:



где h – шаг вычисления;

f(xi, yi) – правая часть дифференциального уравнения

Метод Рунге-Кутты

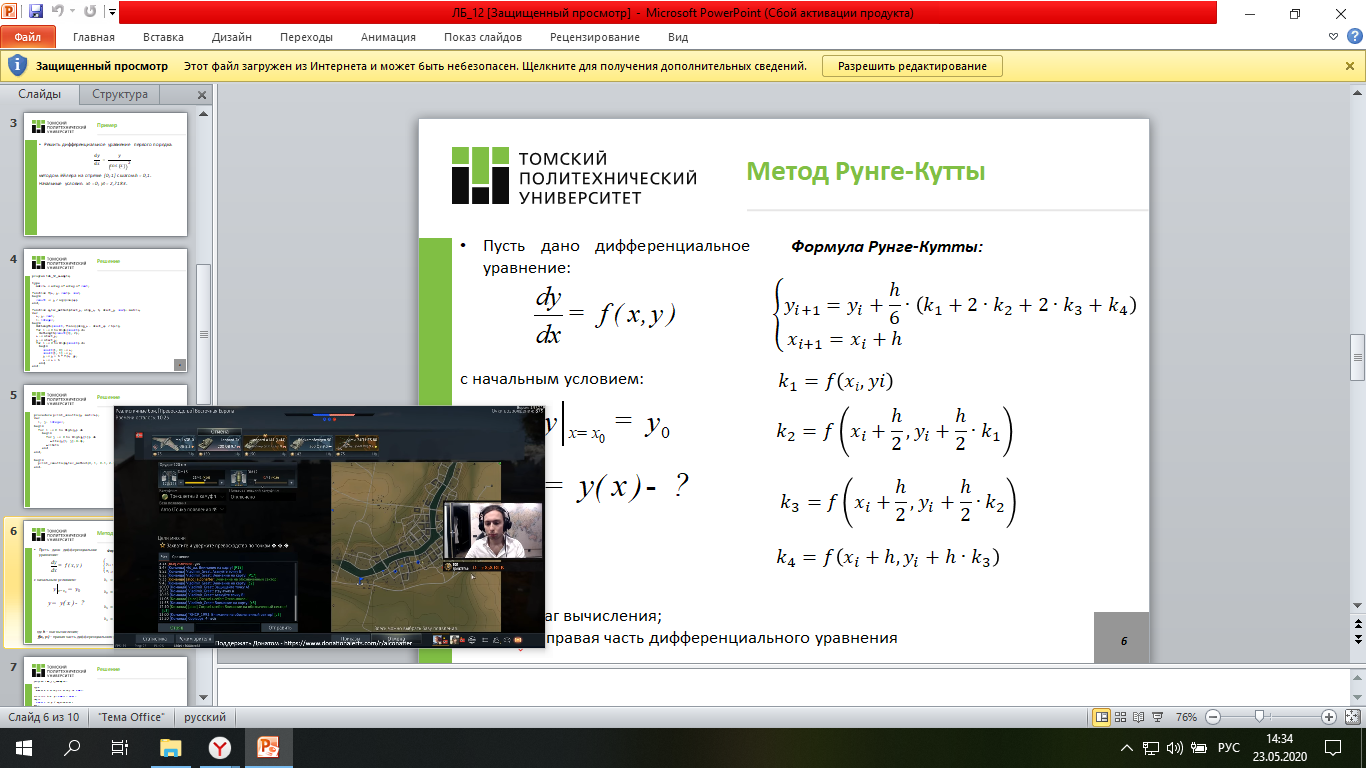
Пусть дано дифференциальное уравнение:



с начальным условием:



Формула Рунге-Кутты:

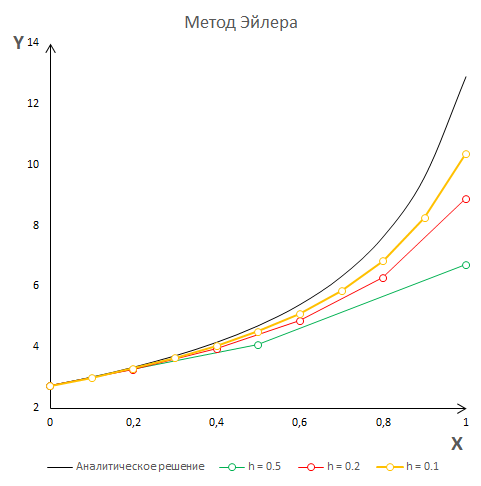


где h – шаг вычисления;

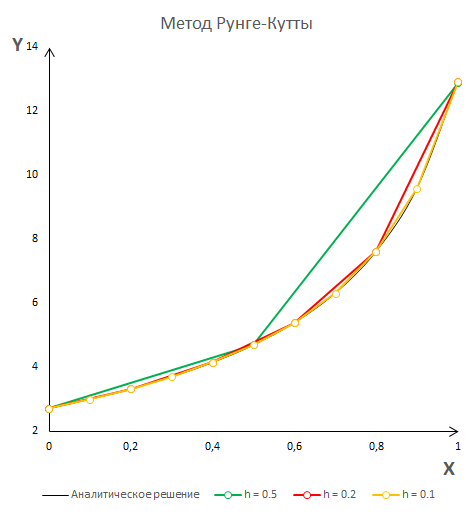
f(xi, yi) – правая часть дифференциального уравнения

*Аналитическое решение*

Метод Эйлера:



Метод Рунге-Кутты

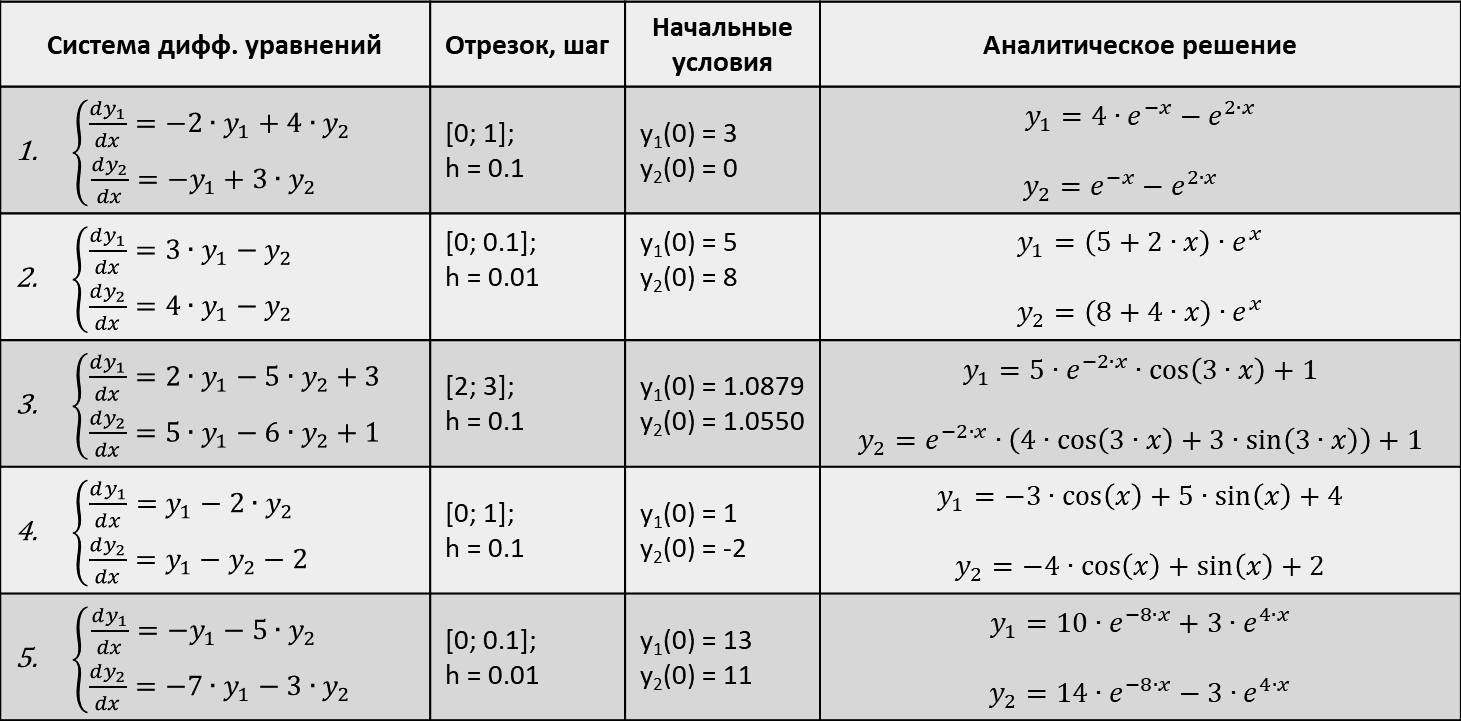


**Практическая часть**

**Задание**

Решить систему дифференциальных уравнений методами Эйлера и Рунге-Кутты. Определить погрешность расчетного значения переменной y для каждого из методов, использовав формулу:

где - расчетное значение, - значение, полученное из аналитического решения.

**Уравнение 1**

**Программная реализация**

**метод Эйлера**

program lb13\_1\_eyler;

type

matrix = array of array of real;

arr = array of real;

function right\_parts(x, y1, y2: real): arr;

begin

SetLength(result, 2);

result[0] := -2\*y1 + 4\*y2;

result[1] := -y1 + 3\*y2

end;

function a\_resh(x: real): arr;

begin

SetLength(result, 2);

result[0] := 4\*exp(-x\*ln(e)) - exp(2\*x\*ln(e));

result[1] := exp(-x\*ln(e)) - exp(2\*x\*ln(e));

end;

function eyler\_method(start\_x, stop\_x, h, start\_y1, start\_y2: real): matrix;

var

i: integer;

x, y1, y2: real;

f: arr;

begin

SetLength(result, Trunc((stop\_x - start\_x) / h) + 1);

for i := 0 to High(result) do

SetLength(result[i], 3);

x := start\_x;

y1 := start\_y1;

y2 := start\_y2;

for i := 0 to High(result) do

begin

result[i, 0] := x;

result[i, 1] := y1;

result[i, 2] := y2;

f := right\_parts(x, y1, y2);

y1 := y1 + h \* f[0];

y2 := y2 + h \* f[1];

x := x + h

end;

end;

function f\_er(y: matrix): matrix;

var

i, j: integer;

x: real;

f1: arr;

begin

SetLength(result, Length (y));

for i := 0 to High(result) do

SetLength(result[i], {4} 7);

for i := 0 to High(result) do

begin

f1 := a\_resh(y[i, 0]);

result[i, 3] := f1[0];

result[i, 4] := f1[1];

if i = 0 then

begin

result[i, 5] := 0;

result[i, 6] := 0;

end

else

begin

result[i, 5] := abs((y[i, 1] - f1[0])/f1[0]) \* 100;

result[i, 6]:= abs((y[i, 2] - f1[1])/f1[1]) \* 100;

end;

for j := 0 to 2 do

result [i, j] := y[i, j];

end;

end;

procedure print\_results(res: matrix);

var

i, j: integer;

begin

for i := 0 to High(res) do

begin

for j := 0 to High(res[i]) do

write(res[i, j]:10:4);

writeln

end;

end;

begin

writeln(' x ',' y1 ', ' y2 ', ' ya1 ', ' ya2 ',

' er1', ' er2 ');

print\_results(f\_er(eyler\_method(0, 1, 0.1, 3, 0)))

end.

**Ответ**

x y1 y2 ya1 ya2 er1 er2

0.0000 3.0000 0.0000 3.0000 0.0000 0.0000 0.0000

0.1000 2.4000 -0.3000 2.3979 -0.3166 0.0856 5.2328

0.2000 1.8000 -0.6300 1.7831 -0.6731 0.9479 6.4024

0.3000 1.1880 -0.9990 1.1412 -1.0813 4.1051 7.6113

0.4000 0.5508 -1.4175 0.4557 -1.5552 20.8586 8.8554

0.5000 -0.1264 -1.8978 -0.2922 -2.1118 56.7496 10.1300

0.6000 -0.8602 -2.4545 -1.1249 -2.7713 23.5272 11.4301

0.7000 -1.6700 -3.1049 -2.0689 -3.5586 19.2795 12.7502

0.8000 -2.5779 -3.8693 -3.1557 -4.5037 18.3086 14.0852

0.9000 -3.6101 -4.7724 -4.4234 -5.6431 18.3858 15.4298

1.0000 -4.7970 -5.8431 -5.9175 -7.0212 18.9355 16.7795

**метод Рунге-Кутты**

program lb\_13\_1\_ kutt;

type

matrix = array of array of real;

arr = array of real;

function right\_parts(x, y1, y2: real): arr;

begin

SetLength(result, 2);

result[0] := -2\*y1 + 4\*y2;

result[1] := -y1 + 3\*y2

end;

function a\_resh(x: real): arr;

begin

SetLength(result, 2);

result[0] := 4\*exp(-x\*ln(e)) - exp(2\*x\*ln(e));

result[1] := exp(-x\*ln(e)) - exp(2\*x\*ln(e));

end;

function runge\_kutt\_method(start\_x, stop\_x, h, start\_y1, start\_y2: real): matrix;

var

i: integer;

x, y1, y2: real;

k1, k2, k3, k4: arr;

begin

SetLength(result, Trunc((stop\_x - start\_x) / h) + 1);

for i := 0 to High(result) do

SetLength(result[i], 3);

x := start\_x;

y1 := start\_y1;

y2 := start\_y2;

for i := 0 to High(result) do

begin

result[i, 0] := x;

result[i, 1] := y1;

result[i, 2] := y2;

k1 := right\_parts(x, y1, y2);

k2 := right\_parts(x + h / 2, y1 + k1[0] \* h / 2, y2 + k1[1] \* h / 2);

k3 := right\_parts(x + h / 2, y1 + k2[0] \* h / 2, y2 + k2[1] \* h / 2);

k4 := right\_parts(x + h, y1 + k3[0] \* h, y2 + k3[1] \* h);

y1 := y1 + h / 6 \* (k1[0] + 2 \* k2[0] + 2 \* k3[0] + k4[0]);

y2 := y2 + h / 6 \* (k1[1] + 2 \* k2[1] + 2 \* k3[1] + k4[1]);

x := x + h

end;

end;

function f\_er(y: matrix): matrix;

var

i, j: integer;

x: real;

f1: arr;

begin

SetLength(result, Length (y));

for i := 0 to High(result) do

SetLength(result[i], {4} 7);

for i := 0 to High(result) do

begin

f1 := a\_resh(y[i, 0]);

result[i, 3] := f1[0];

result[i, 4] := f1[1];

if i = 0 then

begin

result[i, 5] := 0;

result[i, 6] := 0;

end

else

begin

result[i, 5] := abs((y[i, 1] - f1[0])/f1[0]) \* 100;

result[i, 6]:= abs((y[i, 2] - f1[1])/f1[1]) \* 100;

end;

for j := 0 to 2 do

result [i, j] := y[i, j];

end;

end;

procedure print\_results(res: matrix);

var

i, j: integer;

begin

for i := 0 to High(res) do

begin

for j := 0 to High(res[i]) do

write(res[i, j]:10:4);

writeln

end;

end;

begin

writeln(' x ',' y1 ', ' y2 ', ' ya1 ', ' ya2 ',

' er', ' er2 ');

print\_results(f\_er(runge\_kutt\_method(0, 1, 0.1, 3, 0)))

end.

**Ответ**

x y1 y2 ya1 ya2 er1 er2

0.0000 3.0000 0.0000 3.0000 0.0000 0.0000 0.0000

0.1000 2.3980 -0.3166 2.3979 -0.3166 0.0001 0.0009

0.2000 1.7831 -0.6731 1.7831 -0.6731 0.0004 0.0010

0.3000 1.1412 -1.0813 1.1412 -1.0813 0.0012 0.0012

0.4000 0.4558 -1.5552 0.4557 -1.5552 0.0046 0.0013

0.5000 -0.2921 -2.1117 -0.2922 -2.1118 0.0109 0.0015

0.6000 -1.1248 -2.7713 -1.1249 -2.7713 0.0041 0.0016

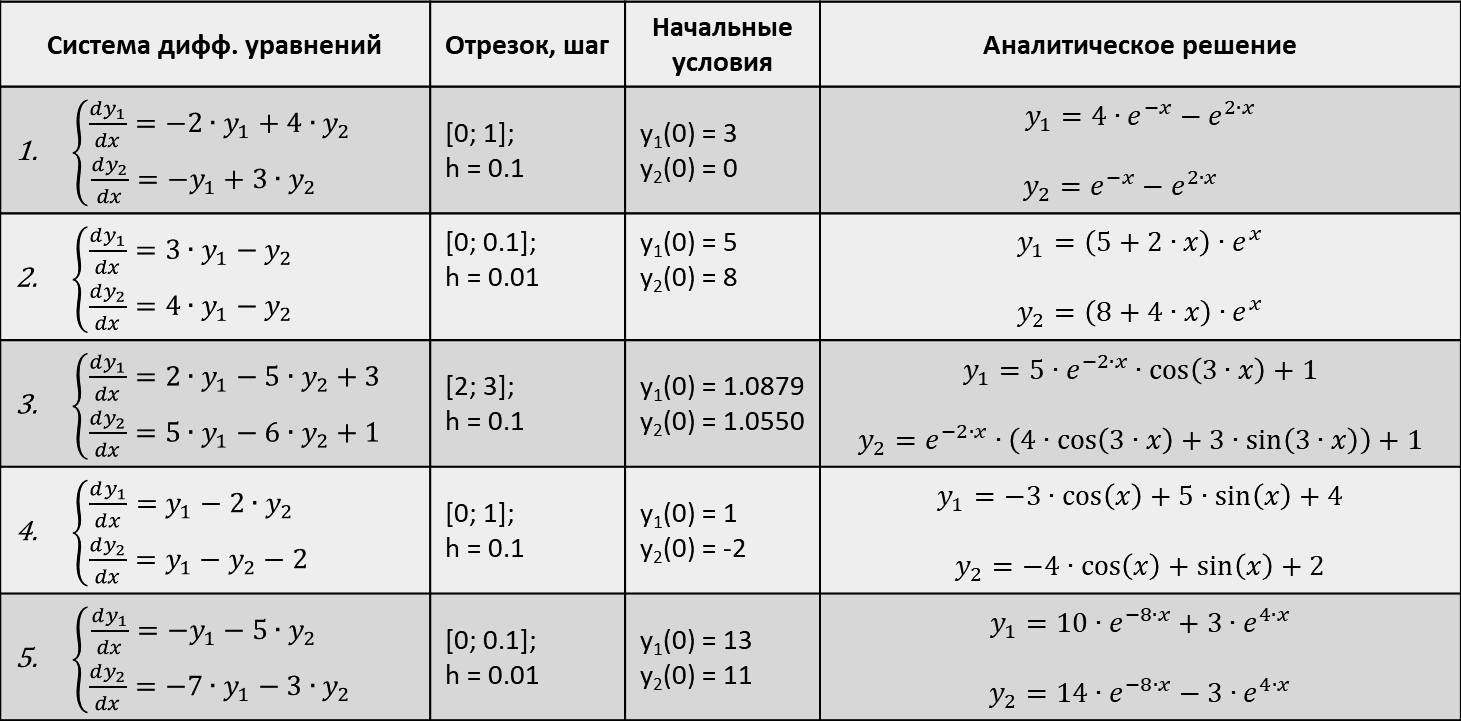
0.7000 -2.0688 -3.5586 -2.0689 -3.5586 0.0032 0.0018

0.8000 -3.1556 -4.5036 -3.1557 -4.5037 0.0029 0.0020

0.9000 -4.4232 -5.6430 -4.4234 -5.6431 0.0028 0.0022

1.0000 -5.9174 -7.0210 -5.9175 -7.0212 0.0028 0.0024

**Уравнение 2**



**Программная реализация**

**метод Эйлера**

program lb13\_2\_eyler;

type

matrix = array of array of real;

arr = array of real;

function right\_parts(x, y1, y2: real): arr;

begin

SetLength(result, 2);

result[0] := 3\*y1 - y2;

result[1] := 4\*y1 - y2

end;

function a\_resh(x: real): arr;

begin

SetLength(result, 2);

result[0] := (5 + 2\*x)\*exp(x\*ln(e));

result[1] := (8 + 4\*x)\*exp(x\*ln(e));

end;

function eyler\_method(start\_x, stop\_x, h, start\_y1, start\_y2: real): matrix;

var

i: integer;

x, y1, y2: real;

f: arr;

begin

SetLength(result, Trunc((stop\_x - start\_x) / h) + 1);

for i := 0 to High(result) do

SetLength(result[i], 3);

x := start\_x;

y1 := start\_y1;

y2 := start\_y2;

for i := 0 to High(result) do

begin

result[i, 0] := x;

result[i, 1] := y1;

result[i, 2] := y2;

f := right\_parts(x, y1, y2);

y1 := y1 + h \* f[0];

y2 := y2 + h \* f[1];

x := x + h

end;

end;

function f\_er(y: matrix): matrix;

var

i, j: integer;

x: real;

f1: arr;

begin

SetLength(result, Length (y));

for i := 0 to High(result) do

SetLength(result[i], {4} 7);

for i := 0 to High(result) do

begin

f1 := a\_resh(y[i, 0]);

result[i, 3] := f1[0];

result[i, 4] := f1[1];

if i = 0 then

begin

result[i, 5] := 0;

result[i, 6] := 0;

end

else

begin

result[i, 5] := abs((y[i, 1] - f1[0])/f1[0]) \* 100;

result[i, 6]:= abs((y[i, 2] - f1[1])/f1[1]) \* 100;

end;

for j := 0 to 2 do

result [i, j] := y[i, j];

end;

end;

procedure print\_results(res: matrix);

var

i, j: integer;

begin

for i := 0 to High(res) do

begin

for j := 0 to High(res[i]) do

write(res[i, j]:10:4);

writeln

end;

end;

begin

writeln(' x ',' y1 ', ' y2 ', ' ya1 ', ' ya2 ',

' er1', ' er2 ');

print\_results(f\_er(eyler\_method(0, 0.1, 0.01, 5, 8)))

end.

**Ответ**

x y1 y2 ya1 ya2 er1 er2

0.0000 5.0000 8.0000 5.0000 8.0000 0.0000 0.0000

0.0100 5.0700 8.1200 5.0705 8.1208 0.0089 0.0099

0.0200 5.1409 8.2416 5.1418 8.2432 0.0178 0.0197

0.0300 5.2127 8.3648 5.2141 8.3673 0.0266 0.0295

0.0400 5.2854 8.4897 5.2873 8.4930 0.0355 0.0393

0.0500 5.3591 8.6162 5.3615 8.6204 0.0442 0.0490

0.0600 5.4337 8.7444 5.4366 8.7495 0.0530 0.0586

0.0700 5.5093 8.8743 5.5127 8.8804 0.0617 0.0682

0.0800 5.5858 9.0059 5.5898 9.0129 0.0704 0.0778

0.0900 5.6633 9.1393 5.6678 9.1473 0.0791 0.0873

0.1000 5.7418 9.2745 5.7469 9.2834 0.0877 0.0968

**метод Рунге-Кутты**

program lb\_13\_2\_kutt;

type

matrix = array of array of real;

arr = array of real;

function right\_parts(x, y1, y2: real): arr;

begin

SetLength(result, 2);

result[0] := -2 \* y1 + 4 \* y2;

result[1] := -y1 + 3 \* y2

end;

function a\_resh(x: real): arr;

begin

SetLength(result, 2);

result[0] := (5 + 2 \* x) \* exp(x \* ln(e));

result[1] := (8 + 4 \* x) \* exp(x \* ln(e));

end;

function runge\_kutt\_method(start\_x, stop\_x, h, start\_y1, start\_y2: real): matrix;

var

i: integer;

x, y1, y2: real;

k1, k2, k3, k4: arr;

begin

SetLength(result, Trunc((stop\_x - start\_x) / h) + 1);

for i := 0 to High(result) do

SetLength(result[i], 3);

x := start\_x;

y1 := start\_y1;

y2 := start\_y2;

for i := 0 to High(result) do

begin

result[i, 0] := x;

result[i, 1] := y1;

result[i, 2] := y2;

k1 := right\_parts(x, y1, y2);

k2 := right\_parts(x + h / 2, y1 + k1[0] \* h / 2, y2 + k1[1] \* h / 2);

k3 := right\_parts(x + h / 2, y1 + k2[0] \* h / 2, y2 + k2[1] \* h / 2);

k4 := right\_parts(x + h, y1 + k3[0] \* h, y2 + k3[1] \* h);

y1 := y1 + h / 6 \* (k1[0] + 2 \* k2[0] + 2 \* k3[0] + k4[0]);

y2 := y2 + h / 6 \* (k1[1] + 2 \* k2[1] + 2 \* k3[1] + k4[1]);

x := x + h

end;

end;

function f\_er(y: matrix): matrix;

var

i, j: integer;

x: real;

f1: arr;

begin

SetLength(result, Length(y));

for i := 0 to High(result) do

SetLength(result[i], {4} 7);

for i := 0 to High(result) do

begin

f1 := a\_resh(y[i, 0]);

result[i, 3] := f1[0];

result[i, 4] := f1[1];

if i = 0 then

begin

result[i, 5] := 0;

result[i, 6] := 0;

end

else

begin

result[i, 5] := abs((y[i, 1] - f1[0]) / f1[0]) \* 100;

result[i, 6] := abs((y[i, 2] - f1[1]) / f1[1]) \* 100;

end;

for j := 0 to 2 do

result[i, j] := y[i, j];

end;

end;

procedure print\_results(res: matrix);

var

i, j: integer;

begin

for i := 0 to High(res) do

begin

for j := 0 to High(res[i]) do

write(res[i, j]:10:4);

writeln

end;

end;

begin

writeln(' x ', ' y1 ', ' y2 ', ' ya1 ', ' ya2 ',

' error1', ' error2 ');

print\_results(f\_er(runge\_kutt\_method(0, 0.1, 0.01, 5, 8)))

end.

**Ответ**

x y1 y2 ya1 ya2 error1 error2

0.0000 5.0000 8.0000 5.0000 8.0000 0.0000 0.0000

0.0100 5.2216 8.1918 5.0705 8.1208 2.9812 0.8738

0.0200 5.4465 8.3871 5.1418 8.2432 5.9257 1.7453

0.0300 5.6747 8.5861 5.2141 8.3673 8.8346 2.6149

0.0400 5.9064 8.7888 5.2873 8.4930 11.7093 3.4826

0.0500 6.1416 8.9953 5.3615 8.6204 14.5508 4.3488

0.0600 6.3804 9.2057 5.4366 8.7495 17.3603 5.2137

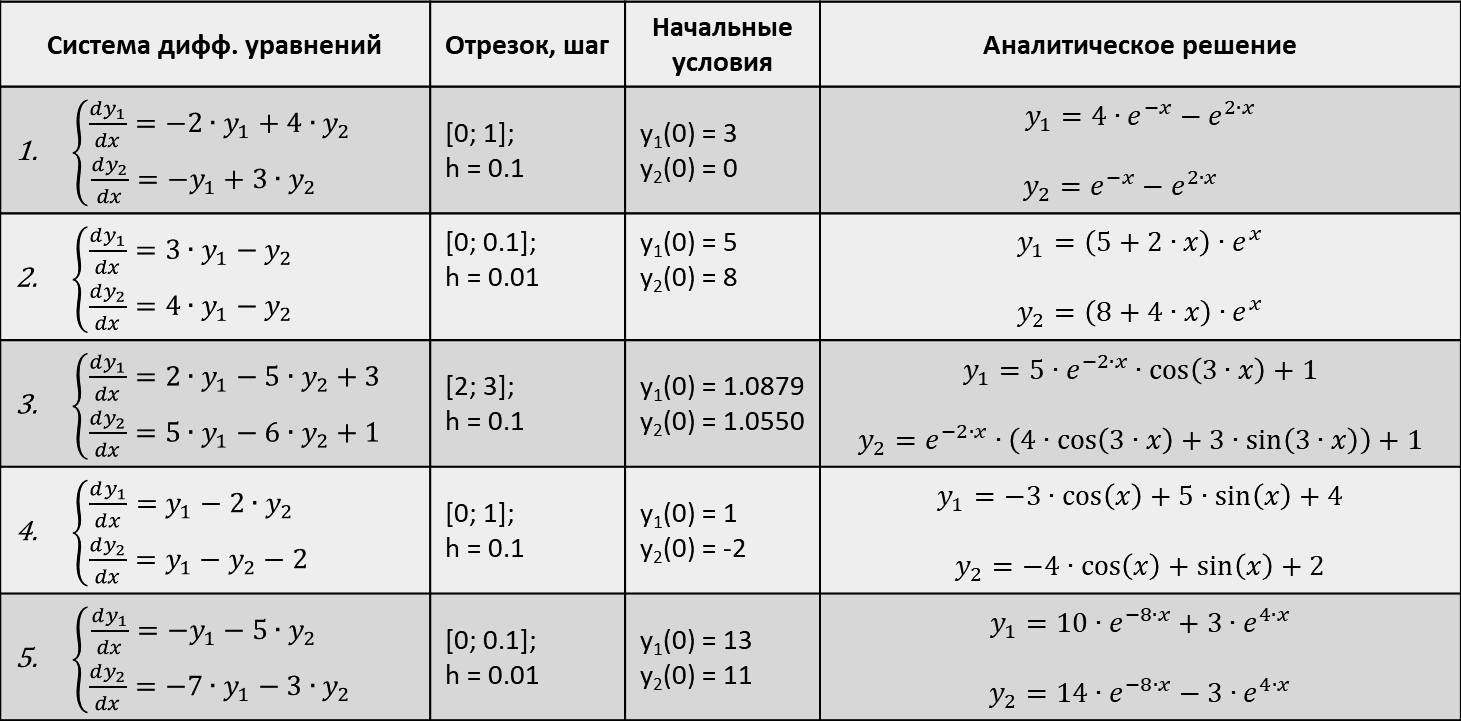
0.0700 6.6229 9.4201 5.5127 8.8804 20.1389 6.0775

0.0800 6.8691 9.6385 5.5898 9.0129 22.8878 6.9404

0.0900 7.1192 9.8610 5.6678 9.1473 25.6079 7.8026

0.1000 7.3733 10.0878 5.7469 9.2834 28.3003 8.6644

**Уравнение 3**



**Программная реализация**

**метод Эйлера**

program lb13\_3\_eyler;

type

matrix = array of array of real;

arr = array of real;

function right\_parts(x, y1, y2: real): arr;

begin

SetLength(result, 2);

result[0] := 2\*y1 - 5\*y2 + 3;

result[1] := 5\*y1 - 6\*y2 + 1;

end;

function a\_resh(x: real): arr;

begin

SetLength(result, 2);

result[0] := 5\*exp(-2\*x\*ln(e)) \* cos(3\*x) + 1;

result[1] := exp(-2\*x\*ln(e))\*(4\*cos(3\*x) + 3\*sin(3\*x)) + 1;

end;

function eyler\_method(start\_x, stop\_x, h, start\_y1, start\_y2: real): matrix;

var

i: integer;

x, y1, y2: real;

f: arr;

begin

SetLength(result, Trunc((stop\_x - start\_x) / h) + 1);

for i := 0 to High(result) do

SetLength(result[i], 3);

x := start\_x;

y1 := start\_y1;

y2 := start\_y2;

for i := 0 to High(result) do

begin

result[i, 0] := x;

result[i, 1] := y1;

result[i, 2] := y2;

f := right\_parts(x, y1, y2);

y1 := y1 + h \* f[0];

y2 := y2 + h \* f[1];

x := x + h

end;

end;

function f\_er(y: matrix): matrix;

var

i, j: integer;

x: real;

f1: arr;

begin

SetLength(result, Length (y));

for i := 0 to High(result) do

SetLength(result[i], {4} 7);

for i := 0 to High(result) do

begin

f1 := a\_resh(y[i, 0]);

result[i, 3] := f1[0];

result[i, 4] := f1[1];

if i = 0 then

begin

result[i, 5] := 0;

result[i, 6] := 0;

end

else

begin

result[i, 5] := abs((y[i, 1] - f1[0])/f1[0]) \* 100;

result[i, 6]:= abs((y[i, 2] - f1[1])/f1[1]) \* 100;

end;

for j := 0 to 2 do

result [i, j] := y[i, j];

end;

end;

procedure print\_results(res: matrix);

var

i, j: integer;

begin

for i := 0 to High(res) do

begin

for j := 0 to High(res[i]) do

write(res[i, j]:10:4);

writeln

end;

end;

begin

writeln(' x ',' y1 ', ' y2 ', ' ya1 ', ' ya2 ',

' er1', ' er2 ');

print\_results(f\_er(eyler\_method(2, 3, 0.1, 1.0879, 1.0550)))

end.

**Ответ**

x y1 y2 ya1 ya2 er1 er2

2.0000 1.0879 1.0550 1.0879 1.0550 0.0000 0.0000

2.1000 1.0780 1.0660 1.0750 1.0607 0.2803 0.4921

2.2000 1.0606 1.0654 1.0583 1.0581 0.2144 0.6833

2.3000 1.0400 1.0564 1.0410 1.0502 0.0924 0.5910

2.4000 1.0198 1.0426 1.0250 1.0396 0.5086 0.2863

2.5000 1.0025 1.0269 1.0117 1.0283 0.9088 0.1317

2.6000 0.9895 1.0120 1.0015 1.0177 1.1964 0.5595

2.7000 0.9814 0.9996 0.9945 1.0087 1.3175 0.9100

2.8000 0.9779 0.9905 0.9904 1.0018 1.2624 1.1256

2.9000 0.9782 0.9852 0.9887 0.9970 1.0574 1.1834

3.0000 0.9813 0.9832 0.9887 0.9940 0.7515 1.0926

**метод Рунге-Кутты**

program lb\_13\_3\_kutt;

type

matrix = array of array of real;

arr = array of real;

function right\_parts(x, y1, y2: real): arr;

begin

SetLength(result, 2);

result[0] := 2 \* y1 - 5 \* y2 + 3;

result[1] := 5 \* y1 - 6 \* y2 + 1;

end;

function a\_resh(x: real): arr;

begin

SetLength(result, 2);

result[0] := 5 \* exp(-2 \* x \* ln(e)) \* cos(3 \* x) + 1;

result[1] := exp(-2 \* x \* ln(e)) \* (4 \* cos(3 \* x) + 3 \* sin(3 \* x)) + 1;

end;

function runge\_kutt\_method(start\_x, stop\_x, h, start\_y1, start\_y2: real): matrix;

var

i: integer;

x, y1, y2: real;

k1, k2, k3, k4: arr;

begin

SetLength(result, Trunc((stop\_x - start\_x) / h) + 1);

for i := 0 to High(result) do

SetLength(result[i], 3);

x := start\_x;

y1 := start\_y1;

y2 := start\_y2;

for i := 0 to High(result) do

begin

result[i, 0] := x;

result[i, 1] := y1;

result[i, 2] := y2;

k1 := right\_parts(x, y1, y2);

k2 := right\_parts(x + h / 2, y1 + k1[0] \* h / 2, y2 + k1[1] \* h / 2);

k3 := right\_parts(x + h / 2, y1 + k2[0] \* h / 2, y2 + k2[1] \* h / 2);

k4 := right\_parts(x + h, y1 + k3[0] \* h, y2 + k3[1] \* h);

y1 := y1 + h / 6 \* (k1[0] + 2 \* k2[0] + 2 \* k3[0] + k4[0]);

y2 := y2 + h / 6 \* (k1[1] + 2 \* k2[1] + 2 \* k3[1] + k4[1]);

x := x + h

end;

end;

function f\_er(y: matrix): matrix;

var

i, j: integer;

x: real;

f1: arr;

begin

SetLength(result, Length(y));

for i := 0 to High(result) do

SetLength(result[i], {4} 7);

for i := 0 to High(result) do

begin

f1 := a\_resh(y[i, 0]);

result[i, 3] := f1[0];

result[i, 4] := f1[1];

if i = 0 then

begin

result[i, 5] := 0;

result[i, 6] := 0;

end

else

begin

result[i, 5] := abs((y[i, 1] - f1[0]) / f1[0]) \* 100;

result[i, 6] := abs((y[i, 2] - f1[1]) / f1[1]) \* 100;

end;

for j := 0 to 2 do

result[i, j] := y[i, j];

end;

end;

procedure print\_results(res: matrix);

var

i, j: integer;

begin

for i := 0 to High(res) do

begin

for j := 0 to High(res[i]) do

write(res[i, j]:10:4);

writeln

end;

end;

begin

writeln(' x ', ' y1 ', ' y2 ', ' ya1 ', ' ya2 ',

' er1', ' er2 ');

print\_results(f\_er(runge\_kutt\_method(2, 3, 0.1, 1.0879, 1.0550)))

end.

**Ответ**

x y1 y2 ya1 ya2 er1 er2

2.0000 1.0879 1.0550 1.0879 1.0550 0.0000 0.0000

2.1000 1.0749 1.0607 1.0750 1.0607 0.0033 0.0004

2.2000 1.0583 1.0581 1.0583 1.0581 0.0035 0.0013

2.3000 1.0410 1.0502 1.0410 1.0502 0.0034 0.0019

2.4000 1.0250 1.0396 1.0250 1.0396 0.0032 0.0022

2.5000 1.0117 1.0283 1.0117 1.0283 0.0028 0.0023

2.6000 1.0015 1.0177 1.0015 1.0177 0.0023 0.0023

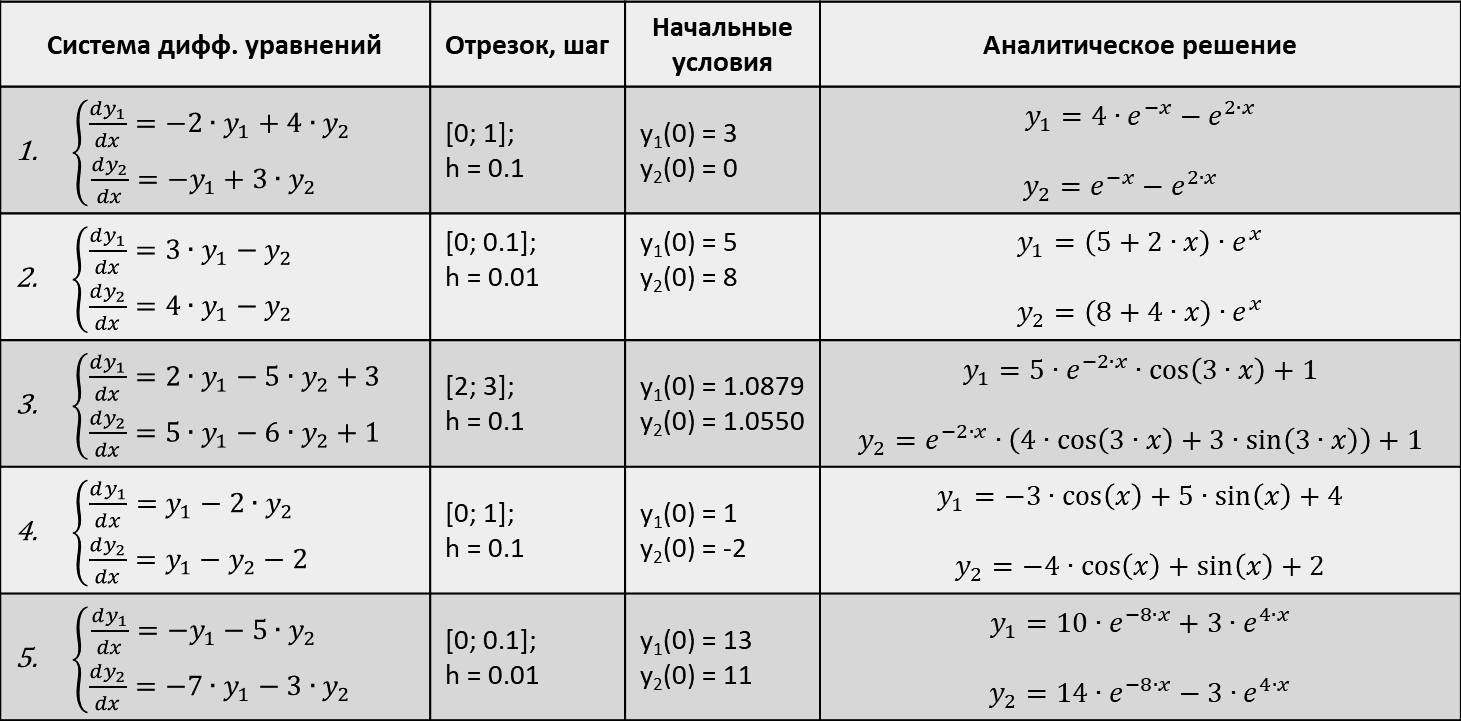
2.7000 0.9945 1.0087 0.9945 1.0087 0.0017 0.0020

2.8000 0.9904 1.0018 0.9904 1.0018 0.0012 0.0017

2.9000 0.9887 0.9969 0.9887 0.9970 0.0007 0.0014

3.0000 0.9887 0.9940 0.9887 0.9940 0.0003 0.0010

**Уравнение 4**



**Программная реализация**

**метод Эйлера**

program lb13\_4\_eyler;

type

matrix = array of array of real;

arr = array of real;

function right\_parts(x, y1, y2: real): arr;

begin

SetLength(result, 2);

result[0] := y1 - 2\*y2;

result[1] := y1 - y2 - 2;

end;

function a\_resh(x: real): arr;

begin

SetLength(result, 2);

result[0] := -3\*cos(x) + 5\*sin(x) + 4;

result[1] := -4\*cos(x) + sin(x) + 2;

end;

function eyler\_method(start\_x, stop\_x, h, start\_y1, start\_y2: real): matrix;

var

i: integer;

x, y1, y2: real;

f: arr;

begin

SetLength(result, Trunc((stop\_x - start\_x) / h) + 1);

for i := 0 to High(result) do

SetLength(result[i], 3);

x := start\_x;

y1 := start\_y1;

y2 := start\_y2;

for i := 0 to High(result) do

begin

result[i, 0] := x;

result[i, 1] := y1;

result[i, 2] := y2;

f := right\_parts(x, y1, y2);

y1 := y1 + h \* f[0];

y2 := y2 + h \* f[1];

x := x + h

end;

end;

function f\_er(y: matrix): matrix;

var

i, j: integer;

x: real;

f1: arr;

begin

SetLength(result, Length (y));

for i := 0 to High(result) do

SetLength(result[i], {4} 7);

for i := 0 to High(result) do

begin

f1 := a\_resh(y[i, 0]);

result[i, 3] := f1[0];

result[i, 4] := f1[1];

if i = 0 then

begin

result[i, 5] := 0;

result[i, 6] := 0;

end

else

begin

result[i, 5] := abs((y[i, 1] - f1[0])/f1[0]) \* 100;

result[i, 6]:= abs((y[i, 2] - f1[1])/f1[1]) \* 100;

end;

for j := 0 to 2 do

result [i, j] := y[i, j];

end;

end;

procedure print\_results(res: matrix);

var

i, j: integer;

begin

for i := 0 to High(res) do

begin

for j := 0 to High(res[i]) do

write(res[i, j]:10:4);

writeln

end;

end;

begin

writeln(' x ',' y1 ', ' y2 ', ' ya1 ', ' ya2 ',

' er1', ' er2 ');

print\_results(f\_er(eyler\_method(0, 1, 0.1, 1, -2)))

end.

**Ответ**

x y1 y2 ya1 ya2 er1 er2

0.0000 1.0000 -2.0000 1.0000 -2.0000 0.0000 0.0000

0.1000 1.5000 -1.9000 1.5142 -1.8802 0.9348 1.0540

0.2000 2.0300 -1.7600 2.0531 -1.7216 1.1274 2.2307

0.3000 2.5850 -1.5810 2.6116 -1.5258 1.0182 3.6160

0.4000 3.1597 -1.3644 3.1839 -1.2948 0.7603 5.3733

0.5000 3.7486 -1.1120 3.7644 -1.0309 0.4205 7.8654

0.6000 4.3458 -0.8259 4.3472 -0.7367 0.0323 12.1129

0.7000 4.9456 -0.5088 4.9266 -0.4152 0.3858 22.5487

0.8000 5.5419 -0.1633 5.4967 -0.0695 0.8227 135.1045

0.9000 6.1287 0.2072 6.0518 0.2969 1.2712 30.2118

1.0000 6.7002 0.5993 6.5864 0.6803 1.7266 11.8948

**метод Рунге-Кутты**

**program** lb\_13\_4\_kutt;

**type**

matrix = **array of array of** real;

arr = **array of** real;

**function** right\_parts(x, y1, y2: real): arr;

**begin**

SetLength(result, 2);

result[0] := y1 - 2\*y2;

result[1] := y1 - y2 - 2;

**end**;

**function** a\_resh(x: real): arr;

**begin**

SetLength(result, 2);

result[0] := -3\*cos(x) + 5\*sin(x) + 4;

result[1] := -4\*cos(x) + sin(x) + 2;

**end**;

**function** runge\_kutt\_method(start\_x, stop\_x, h, start\_y1, start\_y2: real): matrix;

**var**

i: integer;

x, y1, y2: real;

k1, k2, k3, k4: arr;

**begin**

SetLength(result, Trunc((stop\_x - start\_x) / h) + 1);

**for** i := 0 **to** High(result) **do**

SetLength(result[i], 3);

x := start\_x;

y1 := start\_y1;

y2 := start\_y2;

**for** i := 0 **to** High(result) **do**

**begin**

result[i, 0] := x;

result[i, 1] := y1;

result[i, 2] := y2;

k1 := right\_parts(x, y1, y2);

k2 := right\_parts(x + h / 2, y1 + k1[0] \* h / 2, y2 + k1[1] \* h / 2);

k3 := right\_parts(x + h / 2, y1 + k2[0] \* h / 2, y2 + k2[1] \* h / 2);

k4 := right\_parts(x + h, y1 + k3[0] \* h, y2 + k3[1] \* h);

y1 := y1 + h / 6 \* (k1[0] + 2 \* k2[0] + 2 \* k3[0] + k4[0]);

y2 := y2 + h / 6 \* (k1[1] + 2 \* k2[1] + 2 \* k3[1] + k4[1]);

x := x + h

**end**;

**end**;

**function** f\_er(y: matrix): matrix;

**var**

i, j: integer;

x: real;

f1: arr;

**begin**

SetLength(result, Length(y));

**for** i := 0 **to** High(result) **do**

SetLength(result[i], {4} 7);

**for** i := 0 **to** High(result) **do**

**begin**

f1 := a\_resh(y[i, 0]);

result[i, 3] := f1[0];

result[i, 4] := f1[1];

**if** i = 0 **then**

**begin**

result[i, 5] := 0;

result[i, 6] := 0;

**end**

**else**

**begin**

result[i, 5] := abs((y[i, 1] - f1[0]) / f1[0]) \* 100;

result[i, 6] := abs((y[i, 2] - f1[1]) / f1[1]) \* 100;

**end**;

**for** j := 0 **to** 2 **do**

result[i, j] := y[i, j];

**end**;

**end**;

**procedure** print\_results(res: matrix);

**var**

i, j: integer;

**begin**

**for** i := 0 **to** High(res) **do**

**begin**

**for** j := 0 **to** High(res[i]) **do**

write(res[i, j]:10:4);

writeln

**end**;

**end**;

**begin**

writeln(' x ', ' y1 ', ' y2 ', ' ya1 ', ' ya2 ',

' e1', ' e2 ');

print\_results(f\_er(runge\_kutt\_method(0, 1, 0.1, 1, -2)))

**end**.

**Ответ**

x y1 y2 ya1 ya2 e1 e2

0.0000 1.0000 -2.0000 1.0000 -2.0000 0.0000 0.0000

0.1000 1.5142 -1.8802 1.5142 -1.8802 0.0000 0.0000

0.2000 2.0531 -1.7216 2.0531 -1.7216 0.0000 0.0000

0.3000 2.6116 -1.5258 2.6116 -1.5258 0.0001 0.0000

0.4000 3.1839 -1.2948 3.1839 -1.2948 0.0001 0.0001

0.5000 3.7644 -1.0309 3.7644 -1.0309 0.0001 0.0001

0.6000 4.3472 -0.7367 4.3472 -0.7367 0.0001 0.0002

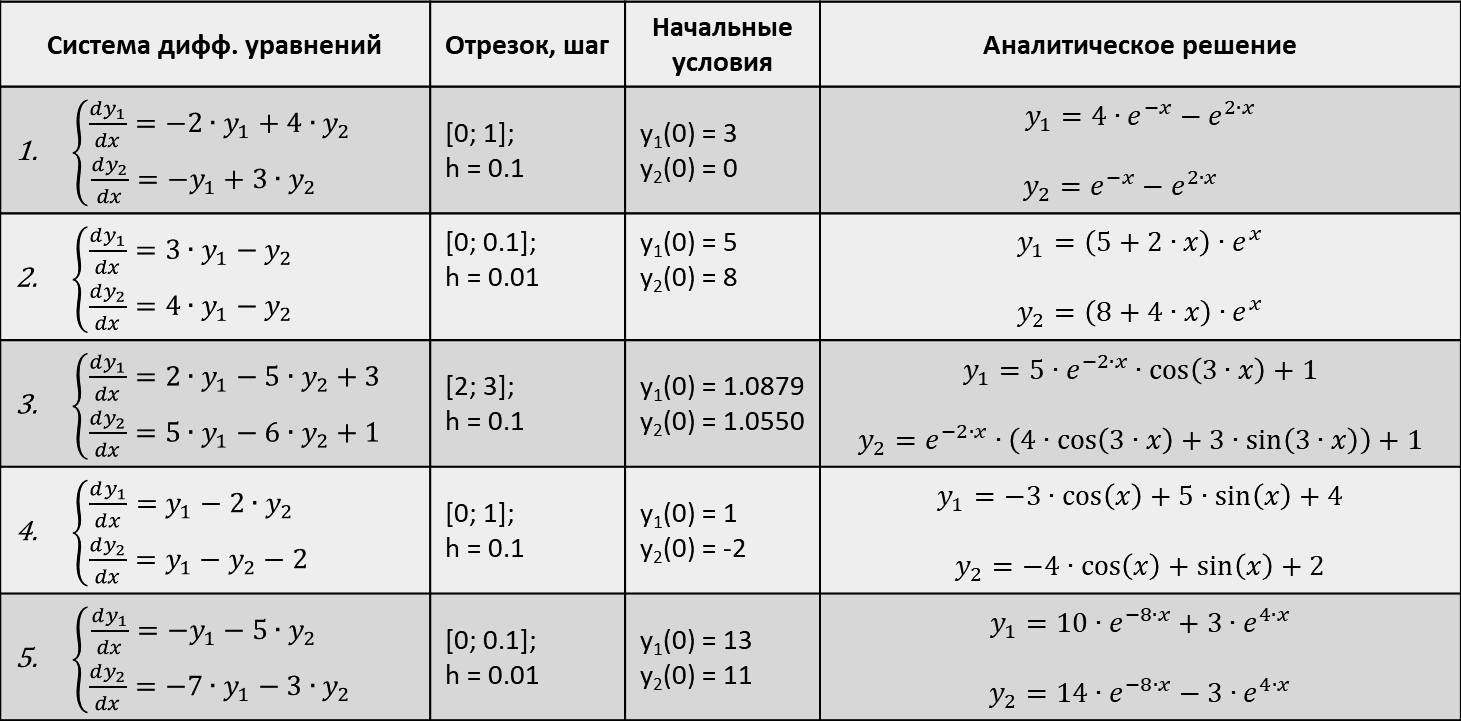
0.7000 4.9266 -0.4152 4.9266 -0.4152 0.0001 0.0004

0.8000 5.4967 -0.0695 5.4967 -0.0695 0.0001 0.0032

0.9000 6.0518 0.2969 6.0518 0.2969 0.0001 0.0009

1.0000 6.5864 0.6803 6.5864 0.6803 0.0001 0.0005

**Уравнение 5**



**Программная реализация**

**Метод Эйлера**

program lb13\_5\_eyler;

type

matrix = array of array of real;

arr = array of real;

function right\_parts(x, y1, y2: real): arr;

begin

SetLength(result, 2);

result[0] := -y1 - 5\*y2;

result[1] := -7\*y1 - 3\*y2;

end;

function a\_resh(x: real): arr;

begin

SetLength(result, 2);

result[0] := 10\*exp(-8\*x\*ln(e)) + 3\*exp(4\*x\*ln(e));

result[1] := 14\*exp(-8\*x\*ln(e)) - 3\*exp(4\*x\*ln(e));

end;

function eyler\_method(start\_x, stop\_x, h, start\_y1, start\_y2: real): matrix;

var

i: integer;

x, y1, y2: real;

f: arr;

begin

SetLength(result, Trunc((stop\_x - start\_x) / h) + 1);

for i := 0 to High(result) do

SetLength(result[i], 3);

x := start\_x;

y1 := start\_y1;

y2 := start\_y2;

for i := 0 to High(result) do

begin

result[i, 0] := x;

result[i, 1] := y1;

result[i, 2] := y2;

f := right\_parts(x, y1, y2);

y1 := y1 + h \* f[0];

y2 := y2 + h \* f[1];

x := x + h

end;

end;

function f\_er(y: matrix): matrix;

var

i, j: integer;

x: real;

f1: arr;

begin

SetLength(result, Length (y));

for i := 0 to High(result) do

SetLength(result[i], {4} 7);

for i := 0 to High(result) do

begin

f1 := a\_resh(y[i, 0]);

result[i, 3] := f1[0];

result[i, 4] := f1[1];

if i = 0 then

begin

result[i, 5] := 0;

result[i, 6] := 0;

end

else

begin

result[i, 5] := abs((y[i, 1] - f1[0])/f1[0]) \* 100;

result[i, 6]:= abs((y[i, 2] - f1[1])/f1[1]) \* 100;

end;

for j := 0 to 2 do

result [i, j] := y[i, j];

end;

end;

procedure print\_results(res: matrix);

var

i, j: integer;

begin

for i := 0 to High(res) do

begin

for j := 0 to High(res[i]) do

write(res[i, j]:10:4);

writeln

end;

end;

begin

writeln(' x ',' y1 ', ' y2 ', ' ya1 ', ' ya2 ',

' er1', ' er2 ');

print\_results(f\_er(eyler\_method(0, 0.1, 0.01, 13, 11)))

end.

**Ответ**

x y1 y2 ya1 ya2 e1 er2

0.0000 13.0000 11.0000 13.0000 11.0000 0.0000 0.0000

0.0100 12.3200 9.7600 12.3536 9.8012 0.2720 0.4203

0.0200 11.7088 8.6048 11.7713 8.6802 0.5309 0.8681

0.0300 11.1615 7.5270 11.2488 7.6303 0.7761 1.3533

0.0400 10.6735 6.5199 10.7820 6.6456 1.0065 1.8904

0.0500 10.2408 5.5772 10.3674 5.7203 1.2215 2.5014

0.0600 9.8595 4.6930 10.0016 4.8492 1.4205 3.2213

0.0700 9.5263 3.8621 9.6815 4.0275 1.6033 4.1087

0.0800 9.2379 3.0794 9.4043 3.2507 1.7695 5.2713

0.0900 8.9915 2.3403 9.1675 2.5145 1.9194 6.9285

0.1000 8.7846 1.6407 8.9688 1.8151 2.0532 9.6095

**метод Рунге-Кутты**

program lb\_13\_3\_kutt;

type

matrix = array of array of real;

arr = array of real;

function right\_parts(x, y1, y2: real): arr;

begin

SetLength(result, 2);

result[0] := -y1 - 5\*y2;

result[1] := -7\*y1 - 3\*y2;

end;

function a\_resh(x: real): arr;

begin

SetLength(result, 2);

result[0] := 10\*exp(-8\*x\*ln(e)) + 3\*exp(4\*x\*ln(e));

result[1] := 14\*exp(-8\*x\*ln(e)) - 3\*exp(4\*x\*ln(e));

end;

function runge\_kutt\_method(start\_x, stop\_x, h, start\_y1, start\_y2: real): matrix;

var

i: integer;

x, y1, y2: real;

k1, k2, k3, k4: arr;

begin

SetLength(result, Trunc((stop\_x - start\_x) / h) + 1);

for i := 0 to High(result) do

SetLength(result[i], 3);

x := start\_x;

y1 := start\_y1;

y2 := start\_y2;

for i := 0 to High(result) do

begin

result[i, 0] := x;

result[i, 1] := y1;

result[i, 2] := y2;

k1 := right\_parts(x, y1, y2);

k2 := right\_parts(x + h / 2, y1 + k1[0] \* h / 2, y2 + k1[1] \* h / 2);

k3 := right\_parts(x + h / 2, y1 + k2[0] \* h / 2, y2 + k2[1] \* h / 2);

k4 := right\_parts(x + h, y1 + k3[0] \* h, y2 + k3[1] \* h);

y1 := y1 + h / 6 \* (k1[0] + 2 \* k2[0] + 2 \* k3[0] + k4[0]);

y2 := y2 + h / 6 \* (k1[1] + 2 \* k2[1] + 2 \* k3[1] + k4[1]);

x := x + h

end;

end;

function f\_er(y: matrix): matrix;

var

i, j: integer;

x: real;

f1: arr;

begin

SetLength(result, Length(y));

for i := 0 to High(result) do

SetLength(result[i], {4} 7);

for i := 0 to High(result) do

begin

f1 := a\_resh(y[i, 0]);

result[i, 3] := f1[0];

result[i, 4] := f1[1];

if i = 0 then

begin

result[i, 5] := 0;

result[i, 6] := 0;

end

else

begin

result[i, 5] := abs((y[i, 1] - f1[0]) / f1[0]) \* 100;

result[i, 6] := abs((y[i, 2] - f1[1]) / f1[1]) \* 100;

end;

for j := 0 to 2 do

result[i, j] := y[i, j];

end;

end;

procedure print\_results(res: matrix);

var

i, j: integer;

begin

for i := 0 to High(res) do

begin

for j := 0 to High(res[i]) do

write(res[i, j]:10:4);

writeln

end;

end;

begin

writeln(' x ', ' y1 ', ' y2 ', ' ya1 ', ' ya2 ',

' e1', ' e2 ');

print\_results(f\_er(runge\_kutt\_method(0, 0.1, 0.01, 13, 11)))

end.

**Ответ**

x y1 y2 ya1 ya2 e1 e2

0.0000 13.0000 11.0000 13.0000 11.0000 0.0000 0.0000

0.0100 12.3536 9.8012 12.3536 9.8012 0.0000 0.0000

0.0200 11.7713 8.6802 11.7713 8.6802 0.0000 0.0000

0.0300 11.2488 7.6303 11.2488 7.6303 0.0000 0.0000

0.0400 10.7820 6.6456 10.7820 6.6456 0.0000 0.0000

0.0500 10.3674 5.7203 10.3674 5.7203 0.0000 0.0000

0.0600 10.0016 4.8492 10.0016 4.8492 0.0000 0.0000

0.0700 9.6815 4.0275 9.6815 4.0275 0.0000 0.0000

0.0800 9.4043 3.2507 9.4043 3.2507 0.0000 0.0001

0.0900 9.1675 2.5145 9.1675 2.5145 0.0000 0.0001

0.1000 8.9688 1.8151 8.9688 1.8151 0.0000 0.0001

**Выводы**

В ходе лабораторной работы были изучены численные решения систем обыкновенных дифференциальных уравнений, с использованием методов Эйлера и Рунге-Кутты. Определена погрешность расчетного значения переменной y для каждого из методов вычисления.