Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа программной инженерии

Отчёт по лабораторной работе № 3

Решение систем дифференциальных уравнений методом Эйлера-Коши.
Программа RKF45.

Выполнил
студент гр. в3530904/00030 В.С. Баганов

Руководитель
профессор, к.т.н. С.М. Устинов

«____»_____ 202__ г.

Санкт-Петербург
2023

Содержание

1. Постановка задачи

Решить систему дифференциальных уравнений:

$$\frac{dx_1}{dt} = -73x_1 - 210x_2 + ln(1+t^2);$$

$$\frac{dx_2}{dt} = x_1 + e^{-t} + t^2 + 1;$$

$$X_1(0) = -3, \qquad X_2(0) = 1, \qquad t\epsilon[0,1]$$

следующими способами с одним и тем же шагом печати $\mathrm{h}_{print} = 0.05$:

I) по программе RKF45 с EPS=0.0001;

II) методом Эйлера-Коши

$$Z_{n+1} = Z_n + (K_1 + K_2)/2;$$

$$K_1 = hf(t_n, Z_n);$$

$$K_2 = hf(t_n + h, Z_n + K_1);$$

с двумя постоянными шагами интегрирования:

1. $\mathrm{h}_{int} = 0,05$

2. любой другой, позволяющий получить качественно верное решение. Сравнить результаты.

2. RKF45. Решение системы дифференциальных уравнений

В лабораторной работе мы используем программу RKF45, которая контролирует погрешности метода в процессе интегрирования и использует переменный шаг интегрирования.

$$RKF45(F, N, X, T, TOUT, RE, AE, IFLAG, WORK, IWORK)$$

F – имя процедуры, написанной пользователем для вычисления правых частей системы (1). Эта программа должна иметь, в свою очередь, следующие параметры:

F(T, X, DX), (X – вектор решения в точке T, а DX – вектор производных);

N – количество интегрируемых уравнений;

X – вектор решения размерностью N в точке T на входе в программу и в точке TOUT при выходе из нее;

T – начальное значение независимой переменной на входе в программу (при нормальном выходе это TOUT)',

TOUT – точка выхода по независимой переменной;

RE, AE – границы относительной и абсолютной погрешностей;

WORK – рабочий вещественный массив размерности 6N+ 3;

IWORK – рабочий целый массив размерности не менее 5;

IFLAG – указатель режима интегрирования.

Обычно при первом обращении на входе IFLAG = 1, а при последующих обращениях на входе IFLAG = 2.

Нормальное выходное значение IFLAG = 2. Другие выходные значения указывают на возникшие отклонения от нормального процесса интегрирования:

= 3 – заданное значение RE оказалось слишком малым и требуется его увеличить;

= 4 – потребовалось более 3000 вычислений f(t, x) (это отвечает приблизительно 500 шагам). Можно, не изменяя IFLAG, снова обратиться к программе или, если система является жесткой, применить специальные алгоритмы решения жестких систем ;

= 5 – решение обратилось в нуль, а AE равно нулю. Требуется задать не нулевое значение AE;

= 6 – требуемая точность не достигнута даже при наименьшей допустимой величине шага и требуется увеличить AE и RE,

= 7 – слишком большое число требуемых выходных точек препятствует выбору естественной величины шага (он может быть значительно увеличен при заданной точности). Нужно или увеличить TOUT–T или задать значение IFLAG=2 и продолжить работу программы;

= 8 – неправильное задание параметров процедуры (например, N $<$ 0 , AE $<$ 0, RE $<$ 0).

3. Расчет критического по устойчивости шага

Программа RKF45 выбирает шаг интегрирования автоматически, но для вычисления ДУ методом Эйлера-Коши, нам необходимо вычислить критический шаг интегрирования.

$$\frac{dx_1}{dt} = -73x_1 - 210x_2 + ln(1+t^2);$$

$$\frac{dx_2}{dt} = x_1 + e^{-t} + t^2 + 1;$$

$$\begin{vmatrix} -\lambda - 73 & -210 \\ 1 & -\lambda \end{vmatrix} = 0$$

$$(-\lambda - 73) * (-\lambda) - (-210) * 1 = 0$$

$$\lambda^2 + 73 * \lambda + 210 = 0$$

$$\lambda_1 = -70; \qquad \lambda_2 = -3$$

Выбираем большее по модулю собственное значение $\lambda_1 = -70$.

$h|\lambda_k| < 2$ - методы второй степени;

$h|\lambda_k| < 2.513$ - метод третьей степени;

$h|\lambda_k| < 2.785$ - метод Рунге Кутты четвертой степени;

$h|\lambda_k| < 1.0$ - метод Адамса второй степени;

$h|\lambda_k| < 6/11$ - метод Адамса третьей степени;

$h|\lambda_k| < 0.3$ - метод Адамса четвертой степени;

Так как у нас уравнения второй степени выбираем коэффициент равный 2

$$h|\lambda_k| < 2$$

$$h < 2/70$$

$$h < 0,02857$$

наш шаг h не должен превышать значения h<0,02857 для получения устойчивой сходимости результатов.

В задании мы должны посчитать h=0,05, который превышает допустимый критический шаг h<0,02857. Чтобы сравнить и изучить, как величина шага влияет на устойчивость сходимости, посчитаем с двумя шагами с h=0,05 и h=0.025. Для удобства сравнения данных, второй шаг возьмем h=0.025, в 2 раза меньшим, чем задан в исходных данных.

4. Метод Эйлера-Коши

Метод Эйлера - Коши - наиболее точный метод решения дифференциального уравнения (второй порядок точности).

$$y_{i+1} = y_i + \frac{h}{2}(f(x_i, y_i) + f(x_{i+1}, y_i + hf(x_i, y_i)))$$

$$Z_{n+1} = Z_n + (K_1 + K_2)/2;$$
$$K_1 = hf(t_n, Z_n);$$
$$K_2 = hf(t_n + h, Z_n + K_1);$$

Метод Эйлера обладает медленной сходимостью, поэтому чаще применяют методы более высокого порядка точности. Второй порядок точности имеет усовершенствованный метод Эйлера: Этот метод имеет простую геометрическую интерпретацию. Метод Эйлера называют методом ломаных, так как интегральная кривая на отрезке заменяется ломаной с угловым коэффициентом . В усовершенствованном методе Эйлера интегральная кривая на отрезке заменяется ломаной с угловым коэффициентом, вычисленным в средней точке отрезка . Так как значение в этой точке неизвестно, для его нахождения используют метод Эйлера с шагом. Модифицированный метод Эйлера с пересчетом имеет второй порядок точности, однако для его реализации необходимо дважды вычислять правую часть функции. Метод Эйлера с пересчетом представляет собой разновидность методов Рунге-Кутта (предиктор-корректор).

5. Код программы C++

```cpp
#include "rkf45.h"
#include <string.h>
#include <fstream>

using namespace std;

std::ofstream out("out.txt");

char* cmathmsg(int routine, int flag)
{
    static char s[64];
    switch (routine)
    {
        case RKFINIT_C:
            switch (flag)
            {
                case 0: strcpy(s, "rkfinit() : normal return");
                    break;
                case 1: strcpy(s,
                ↪  "rkfinit() : could not allocate workspace");
                    break;
                case 2: strcpy(s, "rkfinit() : illegal value for n");
                    break;
                default: strcpy(s, "rkfinit() : no such error");
            };
            break;

        case RKF45_C:
            switch (flag)
            {
                case -2: strcpy(s, "rkf45() : normal return");
                    break;
                case  2: strcpy(s, "rkf45() : normal return");
                    break;
                case  3: strcpy(s, "rkf45() : relerr too small");
                    break;
                case  4: strcpy(s, "rkf45() : too many steps");
                    break;
                case  5: strcpy(s,
                ↪  "rkf45() : abserr needs to be nonzero");
                    break;
                case  6: strcpy(s,
                ↪  "rkf45() : stepsize has become too small");
                    break;
                case  7: strcpy(s, "rkf45() : rkf45 is inefficient");
                    break;
                case  8: strcpy(s, "rkf45() : invalid user input");
                    break;
                default: strcpy(s, "rkf45() : no such error");
            };
            break;
        default: strcpy(s, "CMATH : no such routine");
    }
    return (s);
}
int f(int n, double t, double x[2], double dxdt[2])
```

```
{


    dxdt[0] = -73.0 * x[0] - 210.0 * x[1] + log(1 + t * t);
    dxdt[1] = x[0] + exp(-t) + (t * t) + 1;
    return (0);
}

void countZ(double h)
{
    int i = 1;
    const double SIZE = 1. / h;
    const int PRINT = SIZE / 20;
    double k1[2], k2[2], t = 0.0, z[2], x_prom[2];
    z[0] = -3;
    z[1] = 1;

    out << "\n============== Метод Эйлера-Коши (h = " << h <<
    ↪  ")================\n"
        << "\nstep       x           y[0]       y[1]\n"
        << "-----------------------------------------\n";



    for (int step = 0; step ⩽ SIZE; ++step)

    {
       // t = step * h;
        // вычисляем k1
        f(2, t, z, k1);
  //    k1[0] *= h;
        k1[0] *= 0;
        k1[1] *= h;

        // вычисляем k2
        x_prom[0] = z[0] + k1[0] ;
        x_prom[1] = z[1] + k1[1] ;
        f(2, t + h , x_prom, k2);   // x_prom промежуточный
//        f(2, t , x_prom, k2);   // x_prom промежуточный
        k2[0] *= h;
        k2[1] *= h;

        // вычисляем z
        z[0] = z[0] + (k1[0] + k2[0]) / 2.0;
        z[1] = z[1] + (k1[1] + k2[1]) / 2.0;
        t = step * h;
        if (step % PRINT == 0) out << std::fixed << step << "\t\t" << t
        ↪  << "\t" << z[0] << "\t" << z[1] << "\n";
    }
}

int main()
{

    out << "==========Расчет критического шага=============="
        << "\nlk_1 = -70,  lk_2 = -3\n"
        << "h*|lk| < 2\n"
        <<  "Критический шаг"
        << " h < 0.0285 \n"
        << "================================================\n\n";
```

```
112
113
114         out << ("\n========================RKF45:====================\n\n");
115         double  h, relerr, abserr, t1, t2;
116         int     n, flag, nfe, maxfe, fail, step;
117         double  x[2], yp[2];
118         n = 2;
119         flag = 1;
120         maxfe = 5000;
121         relerr = 1.0e-4;
122         abserr = 1.0e-4;
123         rkfinit(n, &fail);
124         out << ("%s\n\n", cmathmsg(RKFINIT_C, fail));
125
126         if (fail == 0)
127         {
128             x[0] = -3.0;
129             x[1] = 1.0;
130
131             out << ("\nstep       x           y[0]       y[1]\n")
132                 << ("----------------------------------------\n");
133
134             for (step = 0; step <= 20; ++step)
135             {
136                 t2 = step * 0.05; //t+h
137                 t1 = t2 - 0.05;
138                 rkf45(f, n, x, yp, &t1, t2, &relerr, abserr,
139                       &h, &nfe, maxfe, &flag);
140                 out << std::fixed << step << "\t\t"<< t1 << "\t" << x[0] <<
                    ↪  "\t" << x[1] << "\n";
141                 if (flag != 2)
142                 {
143                     out << ("%s\n", cmathmsg(RKF45_C, flag));
144                     break;
145                 }
146             }
147
148             rkfend();
149             out << ("\n%s\n", cmathmsg(RKF45_C, flag)) << "\n"
150                 << "nfe:          " << nfe << "\n"
151                 << "step size:    " << h << "\n";
152         }
153
154         h = 0.05;
155         countZ(h);
156 //
157 //      h = 0.020;
158 //      countZ(h);
159
160         h = 0.025;
161         countZ(h);
162
163         h = 0.010;
164         countZ(h);
165
166         return 0;
167 }
```

Листинг 1: Код программы

6. Код программы FORTRAN

```
program CM_CW

  use environment
   use RKF45MOD

   implicit none
   character(*), parameter :: output_file = "output.txt"
   integer                 :: Out = 0, i = 0
   !Начальные условия задачи
   real(R_)                :: X(2) = [-3, 1], DX(2) = 0, Z(2) = 0
   real(R_)                :: T = 0, TOUT = 0, h = 0.0025
   !Служебные переменные
   real(R_)                :: RE = 0.0001, AE = 0.0001, WORK(15) = 0
   integer                 :: N = 2, IFLAG = 1, IWORK(5) = 0

   TOUT = T + h
   Z = X

   open (file=output_file, encoding=E_, newunit=Out, position='append')
   do i = 1, 60
      call RK3(Z, T, h)
      call RKF45(F, N, X, T, TOUT, RE, AE, IFLAG, WORK, IWORK)
      write (Out, "(a, i3)") "Точка №" ,i
      write (Out, "(a, f8.4, 1x, a, f8.4)") " T = ", T - h, "TOUT = ",
       ↪  TOUT
      write (Out, "(a, f10.7, a, f10.7)") "X1 = ", X(1), " X2 = ", X(2)
      write (Out, "(a, f10.7, a, f10.7)") "Z1 = ", Z(1), " Z2 = ", Z(2)
      write (Out, "(a, i2)") "IFLAG =", IFLAG
      write (Out, *)
      TOUT = T + h
   end do
   close (Out)

contains

   pure subroutine F(T, X, DX)
      real(R_), intent(in)  :: T, X(2)
      real(R_), intent(out) :: DX(2)

      DX(1) = -73 * X(1)  210 * X(2) + log(1 + T * T)
      DX(2) = X(1) + exp(-T) + (T * T) + 1
   end subroutine F

   pure subroutine RK3(X, T, h)
      real(R_), intent(in)    :: T, h
      real(R_), intent(inout) :: X(2)

      real(R_)  k1(2), k2(2), DX(2), Xprom(2)

      call F(T, X, DX)
    !  k1 = h * DX
      k1 = h
      k1 = h

      Xprom = X + k1
      Xprom = X + k2

```

```
57          call F(T + h, Xprom, DX)
58          k2 = h * DX
59          k2 = h * DX
60
61          X = X + (k1 + k2 ) / 2
62          X = X + (k1 + k2 ) / 2
63
64       end subroutine RK3
65    end program CM_CW
```

Листинг 2: Код программы

7. Результат программы

```
==========Расчет критического шага==============
lk_1 = -70,  lk_2 = -3
h*|lk| < 2
Критический шаг h < 0.0285
================================================


========================RKF45:============================

rkfinit() : normal return
step       x           y[0]       y[1]
----------------------------------------
0    0.000000        -2.789472        0.957720
1    0.050000        -2.675690        0.919974
2    0.100000        -2.573755        0.885461
3    0.150000        -2.480850        0.854054
4    0.200000        -2.396600        0.825674
5    0.250000        -2.320872        0.800244
6    0.300000        -2.253403        0.777696
7    0.350000        -2.194055        0.757966
8    0.400000        -2.142619        0.740992
9    0.450000        -2.098964        0.726719
10    0.500000         -2.062918         0.715093
11    0.550000         -2.034356         0.706066
12    0.600000         -2.013136         0.699589
13    0.650000         -1.999142         0.695620
14    0.700000         -1.992251         0.694118
15    0.750000         -1.992359         0.695043
16    0.800000         -1.999357         0.698358
17    0.850000         -2.013152         0.704030
18    0.900000         -2.033649         0.712025
19    0.950000         -2.060761         0.722313
20    1.000000         -2.094405         0.734865
rkf45() : normal return
nfe:          162
step size:   0.075470

============== Метод Эйлера-Коши (h = 0.050000)================

step       x           y[0]       y[1]
----------------------------------------
0    0.000000        -3.108688        0.960093
1    0.050000        -3.861822        0.935145
2    0.100000        -6.889081        0.945687
3    0.150000        -18.136175         1.076582
4    0.200000        -59.156926         1.635673
5    0.250000        -208.084509         3.739082
6    0.300000        -748.152209         11.432938
7    0.350000        -2706.081180         39.384628
8    0.400000        -9803.736065         140.763594
9    0.450000        -35532.876624         508.309210
10    0.500000         -128801.132383         1840.701850
11    0.550000         -466898.660759         6670.658004
12    0.600000         -1692502.282742         24179.275071
13    0.650000         -6135315.474749         87648.031246
14    0.700000         -22240513.339118         317722.285096
15    0.750000         -80621855.622113         1151741.461532
```

```
16    0.800000        -292254221.404229        4175060.975955
17    0.850000        -1059421547.352525        15134594.209290
18    0.900000        -3840403103.885345        54862902.167485
19    0.950000        -13921461246.269424        198878018.497178
20    1.000000        -50465297012.346909        720932815.167514

============== Метод Эйлера-Коши (h = 0.025000)================

step       x            y[0]       y[1]
---------------------------------------
0                 0.000000        -2.821875        0.975000
2                 0.050000        -2.695310        0.935043
4                 0.100000        -2.590414        0.898797
6                 0.150000        -2.494973        0.865822
8                 0.200000        -2.408590        0.836013
10                 0.250000        -2.330998        0.809277
12                 0.300000        -2.261952        0.785529
14                 0.350000        -2.201232        0.764691
16                 0.400000        -2.148633        0.746689
18                 0.450000        -2.103968        0.731458
20                 0.500000        -2.067062        0.718936
22                 0.550000        -2.037758        0.709065
24                 0.600000        -2.015904        0.701793
26                 0.650000        -2.001363        0.697070
28                 0.700000        -1.994006        0.694849
30                 0.750000        -1.993712        0.695088
32                 0.800000        -2.000368        0.697746
34                 0.850000        -2.013868        0.702784
36                 0.900000        -2.034113        0.710169
38                 0.950000        -2.061009        0.719866
40                 1.000000        -2.094469        0.731844
```

Листинг 3: Результат программы

RKF			Эйлера-Коши (h = 0,05)		Эйлера-Коши (h = 0,025)	
x	y[0]	y[1]	y[0]	y[1]	y[0]	y[1]
0.00	-2,789472	0,95772	-3,108688	0,960093	-2,821875	0,975
0.05	-2,67569	0,919974	-3,861822	0,935145	-2,69531	0,935043
0.10	-2,573755	0,885461	-6,889081	0,945687	-2,590414	0,898797
0.15	-2,48085	0,854054	-18,136175	1,076582	-2,494973	0,865822
0.20	-2,3966	0,825674	-59,156926	1,635673	-2,40859	0,836013
0.25	-2,320872	0,800244	-208,084509	3,739082	-2,330998	0,809277
0.30	-2,253403	0,777696	-748,152209	11,432938	-2,261952	0,785529
0.35	-2,194055	0,757966	-2706,08118	39,384628	-2,201232	0,764691
0.40	-2,142619	0,740992	-9803,736065	140,763594	-2,148633	0,746689
0.45	-2,098964	0,726719	-35532,87662	508,30921	-2,103968	0,731458
0.50	-2,062918	0,715093	-128801,1324	1840,70185	-2,067062	0,718936
0.55	-2,034356	0,706066	-466898,6608	6670,658004	-2,037758	0,709065
0.60	-2,013136	0,699589	-1692502,283	24179,27507	-2,015904	0,701793
0.65	-1,999142	0,69562	-6135315,475	87648,03125	-2,001363	0,69707
0.70	-1,992251	0,694118	-22240513,34	317722,2851	-1,994006	0,694849
0.75	-1,992359	0,695043	-80621855,62	1151741,462	-1,993712	0,695088
0.80	-1,999357	0,698358	-292254221,4	4175060,976	-2,000368	0,697746
0.85	-2,013152	0,70403	-1059421547	15134594,21	-2,013868	0,702784
0.90	-2,033649	0,712025	-3840403104	54862902,17	-2,034113	0,710169
0.95	-2,060761	0,722313	-13921461246	198878018,5	-2,061009	0,719866
1.00	-2,094405	0,734865	-50465297012	720932815,2	-2,094469	0,731844

8. Заключение

Решение данной системы ОДУ с помощью программы RKF45 в один шаг дало результаты, идентичные результатам с решением в 20 шагов. Это говорит об адаптивности программы RKF45, которая сама выбирает шаг для поддержания необходимой точности ограничиваясь примерно 500 шагами (3000 вычислений функции) на помежуток.

9. Выводы

Решая дифференциальные уравнения различными методами, мы должны учитывать критический шаг интегрирования, для каждого отдельного метода (рассчитывая его из собственный значений уравнений), чтобы получать адекватные значения. Программа RKF45 позволяет нам использовать ее для решения дифференциальных уравнений высокой степени (десятой степени с 17 вычислениями f(t, x) на шаге), не переживая за точность полученных данных.

RKF			Эйлера-Коши (h = 0.05)	
x	y[0]	y[1]	Δy[0]=y*-y	Δy[1]=y*-y
0.00	-2,789472	0,95772	-0,319216	0,002373
0.05	-2,67569	0,919974	-1,186132	0,015171
0.10	-2,573755	0,885461	-4,315326	0,060226
0.15	-2,48085	0,854054	-15,655325	0,222528
0.20	-2,3966	0,825674	-56,760326	0,809999
0.25	-2,320872	0,800244	-205,76364	2,938838
0.30	-2,253403	0,777696	-745,89881	10,655242
0.35	-2,194055	0,757966	-2703,8871	38,626662
0.40	-2,142619	0,740992	-9801,5934	140,022602
0.45	-2,098964	0,726719	-35530,778	507,582491
0.50	-2,062918	0,715093	-128799,07	1839,98676
0.55	-2,034356	0,706066	-466896,63	6669,95194
0.60	-2,013136	0,699589	-1692500,3	24178,5755
0.65	-1,999142	0,69562	-6135313,5	87647,3356
0.70	-1,992251	0,694118	-22240511	317721,591
0.75	-1,992359	0,695043	-80621854	1151740,77
0.80	-1,999357	0,698358	-292254219	4175060,28
0.85	-2,013152	0,70403	-1,059E+09	15134593,5
0.90	-2,033649	0,712025	-3,84E+09	54862901,5
0.95	-2,060761	0,722313	-1,392E+10	198878018
1.00	-2,094405	0,734865	-5,047E+10	720932814

RKF			**Эйлера-Коши (h = 0.025)**	
x	**y[0]**	**y[1]**	**Δy[0]=y*-y**	**Δy[1]=y*-y**
0.00	-2,789472	0,95772	-0,032403	0,01728
0.05	-2,67569	0,919974	-0,01962	0,015069
0.10	-2,573755	0,885461	-0,016659	0,013336
0.15	-2,48085	0,854054	-0,014123	0,011768
0.20	-2,3966	0,825674	-0,01199	0,010339
0.25	-2,320872	0,800244	-0,010126	0,009033
0.30	-2,253403	0,777696	-0,008549	0,007833
0.35	-2,194055	0,757966	-0,007177	0,006725
0.40	-2,142619	0,740992	-0,006014	0,005697
0.45	-2,098964	0,726719	-0,005004	0,004739
0.50	-2,062918	0,715093	-0,004144	0,003843
0.55	-2,034356	0,706066	-0,003402	0,002999
0.60	-2,013136	0,699589	-0,002768	0,002204
0.65	-1,999142	0,69562	-0,002221	0,00145
0.70	-1,992251	0,694118	-0,001755	0,000731
0.75	-1,992359	0,695043	-0,001353	4,5E-05
0.80	-1,999357	0,698358	-0,001011	-0,000612
0.85	-2,013152	0,70403	-0,000716	-0,001246
0.90	-2,033649	0,712025	-0,000464	-0,001856
0.95	-2,060761	0,722313	-0,000248	-0,002447
1.00	-2,094405	0,734865	-6,4E-05	-0,003021