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

ЛАБОРАТОРНАЯ РАБОТА № 3

по дисциплине «Вычислительная математика»

Выполнил

студент гр. 3530904/80004 Петров А.И.

Руководитель,

к.т.н, доцент Леонтьева Т.В.

Санкт-Петербург

2020

Формулировка задания

Рещить систему дифференциальных уравнений:

следующими способами с одним и тем же шагом печати :

1. по программе RKF45 с EPS =0.0001
2. методом Рунге-Кутты 3-й степени точности

с двумя постоянными шагами интегрирования:

* 1. hint = 0.1
  2. любой другой, позволяющий получить качественно верное решение.

Сравнить результаты.

Выполнение работы

Задана функция f(t, x), которую будем интегрировать. Методами Рунге-Кутты 4 и 5 степени, сделаем это. Затем реализуем метод Рунге-Кутты 3 степени точности. Рекомендуемый шаг, оцениваемый программой, равен 0.05026. Используем метод с шагом 0.1. Заметим, что при этом шаге метод не сходится. Используем шаг 0.05. Все работает, результаты работы качественно совпадают с результатами методов 4 и 5 степени, но тем не менее недостаточно точны.

Вывод программы

------------RK45-------------

0.10 : [0.8696553 0.07504291]

0.20 : [0.79199395 0.07099203]

0.30 : [0.72561594 0.06730768]

0.40 : [0.66883776 0.0643855 ]

0.50 : [0.62039372 0.06210347]

0.60 : [0.57915384 0.0603443 ]

0.70 : [0.54410223 0.05900419]

0.80 : [0.51432507 0.05798819]

0.90 : [0.48899858 0.05721071]

1.00 : [0.46737974 0.0565944 ]

1.10 : [0.44879839 0.05606974]

1.20 : [0.43265077 0.05557455]

1.30 : [0.41839406 0.05505368]

1.40 : [0.40554159 0.05445922]

1.50 : [0.39366006 0.05374818]

1.60 : [0.3823648 0.05288523]

1.70 : [0.37131802 0.05184048]

1.80 : [0.36022603 0.05058993]

1.90 : [0.34883708 0.04911518]

2.00 : [0.33693941 0.04740321]

Max step for RK3 (recommendation): 0.05026

------------RK3--------------

Integration step: 0.1

0.10 : [0.36831126 1.07768223]

0.20 : [ 6.9715121 -12.28812684]

0.30 : [-75.48903008 152.4964949 ]

0.40 : [ 940.64875509 -1879.89556733]

0.50 : [-11592.46606662 23186.23489936]

0.60 : [ 142981.97797263 -285962.73741959]

0.70 : [-1763436.70887125 3526874.56482704]

0.80 : [ 21749059.9667982 -43498118.8470772]

0.90 : [-2.68238399e+08 5.36476800e+08]

1.00 : [ 3.3082736e+09 -6.6165472e+09]

1.10 : [-4.08020411e+10 8.16040821e+10]

1.20 : [ 5.03225173e+11 -1.00645035e+12]

1.30 : [-6.20644380e+12 1.24128876e+13]

1.40 : [ 7.65461402e+13 -1.53092280e+14]

1.50 : [-9.44069063e+14 1.88813813e+15]

1.60 : [ 1.16435184e+16 -2.32870369e+16]

1.70 : [-1.43603394e+17 2.87206788e+17]

1.80 : [ 1.77110853e+18 -3.54221705e+18]

1.90 : [-2.18436718e+19 4.36873437e+19]

2.00 : [ 2.69405286e+20 -5.38810572e+20]

------------RK3--------------

Integration step: 0.05

0.10 : [ 0.90832808 -0.00230832]

0.20 : [ 0.82933051 -0.00369071]

0.30 : [ 0.76141275 -0.00429814]

0.40 : [ 0.70315679 -0.0042663 ]

0.50 : [ 0.65329692 -0.00371744]

0.60 : [ 0.6106996 -0.00276191]

0.70 : [ 0.57434682 -0.00149943]

0.80 : [ 5.43322275e-01 -2.00619181e-05]

0.90 : [0.51679996 0.00159492]

1.00 : [0.49403469 0.00327243]

1.10 : [0.47435426 0.00494699]

1.20 : [0.45715293 0.00656035]

1.30 : [0.44188599 0.00806106]

1.40 : [0.42806524 0.00940427]

1.50 : [0.41525518 0.01055136]

1.60 : [0.40306973 0.01146984]

1.70 : [0.39116948 0.01213303]

1.80 : [0.37925921 0.01251995]

1.90 : [0.36708574 0.01261509]

2.00 : [0.35443591 0.01240823]

Вывод

В ходе работы было получено решение системы методами Рунге-Кутты 4 и 5 степени точности, которое примем за эталон. Затем было выяснено, что для метода 3 степени точности шаг 0.1 слишком велик и метод не сходится. Однако шага в 0.01 оказалось достаточно и полученное таким образом решение совпадает с эталонным. Зависимость сходимости решения от величины шага подтверждается теорией.

Исходный код программы:

Основной класс программы.

import numpy

from scipy.integrate import RK45

from python.RK3 import print\_integrated, RK3

def f(t, x):

return numpy.array([

-4 \* x[0] + 23 \* x[1] + numpy.exp(-t),

4 \* x[0] - 48 \* x[1] + numpy.sin(t)]

)

initial = numpy.array([

1,

0.]

)

lower\_bound = 0.0

upper\_bound = 2

step = 0.1

current = lower\_bound

res = initial.copy()

print("------------RK45-------------")

while current < upper\_bound:

integrator = RK45(

fun=f,

t0=current,

y0=res.copy(),

t\_bound=current + step,

rtol=0.00001

)

while integrator.status == "running":

integrator.step()

res = integrator.y

current += step

print("{0:1.2f} : {1}".format(current, res))

integrator = RK3(f, lower\_bound, initial.copy(), upper\_bound)

print("Max step for RK3 (recommendation): {0}".format(integrator.max\_step()))

print\_integrated(

integrator=integrator,

integration\_step=0.1,

print\_step=0.1)

integrator = RK3(f, lower\_bound, initial.copy(), upper\_bound)

print\_integrated(

integrator=integrator,

integration\_step=0.05,

print\_step=0.1)

RK3

import numpy

from scipy import linalg

from scipy.linalg import norm

from scipy.optimize import toms748, minimize

class RK3:

def \_\_init\_\_(self, f, t0, y0, t):

self.f = f

self.lower\_bound = t0

self.t = t0

self.y = y0

self.upper\_bound = t

self.status = "running"

def step(self, h):

if self.status == "completed":

return

if self.t + h > self.upper\_bound:

h = self.upper\_bound - self.t

k1 = self.k1(h)

k2 = self.k2(h, k1)

k3 = self.k3(h, k1, k2)

self.y = self.z(k1, k2, k3)

self.t += h

if self.t >= self.upper\_bound:

self.status = "completed"

def k1(self, h):

return h \* self.f(self.t, self.y)

def k2(self, h, k1):

return h \* self.f(self.t + h / 2, self.y + k1 / 2)

def k3(self, h, k1, k2):

return h \* (self.f(self.t + h, self.y - k1 + 2 \* k2))

def z(self, k1, k2, k3):

return self.y + (k1 + 4 \* k2 + k3) / 6

def max\_step(self):

t\_dependent = self.f(0, [0, 0])

matrix = []

for i in range(self.y.size):

x = numpy.array([1 if i == j else 0 for j in range(self.y.size)])

column = self.f(0, x) - t\_dependent

matrix.append(column)

lam = abs(min(filter(lambda l: l < 0, linalg.eig(matrix)[0])).real)

return 2.513 / lam

def print\_integrated(integrator, integration\_step, print\_step):

print("------------RK3--------------")

print("Integration step: {0}".format(integration\_step))

cur\_step = 0

while integrator.status == "running":

integrator.step(integration\_step)

cur\_step += integration\_step

if numpy.abs(cur\_step - print\_step) < 0.000001:

cur\_step = 0

print("{0:1.2f} : {1}".format(integrator.t, integrator.y))integrator.t, rk3\_integrator.y))