Белорусский государственный университет

Факультет прикладной математики и информатики

Методы решения граничной задачи для ОДУ-1

Отчет по лабораторной работе

студента 3 курса 3 группы

Аквуха Джеймса

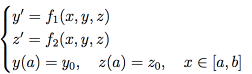
Преподаватель:

Будник А.М.

**1. Постановка задачи**

Поставлена задача Коши для одного уравнения и системы 2 уравнений:





Требуется найти значения искомых функций на отрезке [a, b] на равномерной сетке из N + 1 точки.

**2. Методы решения**

**2.1 Метод рядов**

Искомая функция приближается своим разложением в ряд Тейлора в окрестности узловых точек. Точки вычисляются последовательно, начиная с точки, для которой известно точное значение из постановки задачи Коши:



Для системы уравнений записываются два аналогичных приближения искомых функций.

**2.2 Явный метод Эйлера**

Изменение значения искомой функции приближается с помощью формулы левых прямоугольников.



Для системы уравнений записываются два аналогичных приближения искомых функций.

**2.3 Неявный метод Эйлера**

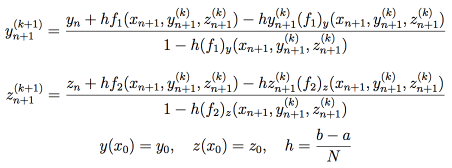
Изменение значения искомой функции приближается с помощью формулы правых прямоугольников. Полученная система решается методом Ньютона.



Начальное приближение для метода Ньютона находится по формуле явного метода Эйлера. Итерации проводятся пока норма невязки изменения решения не окажется меньше .



Для системы уравнений записываются два аналогичных приближения искомых функций.



**2.4 Метод предиктор-корректор**

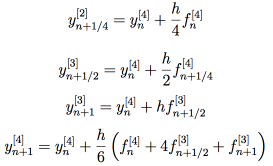
Изменение искомой функции приближается квадратурной формулой:



Требуя, чтобы формула была точна для всех многочленов степени , получаем систему уравнений:



, решая которую для , получаем:

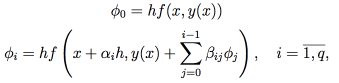


В случае системы полученная система рассматривается для обоих уравнений.

**2.5 Метод Рунге-Кутты**

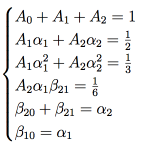
Правило Рунге-Кутты основано на приближении изменения искомой функции некоторой линейной комбинацией.

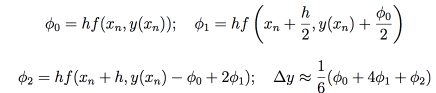




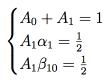


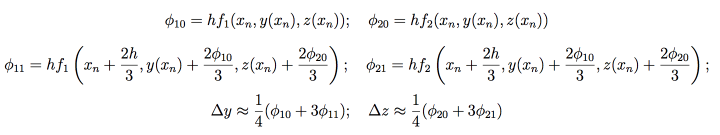
При получим метод Рунге-Кутты третьего порядка точности:





А при - второго порядка точности:





**2.6 Метод Адамса**

Метод Адамса - многошаговый метод, поэтому искомая функция аппроксимируется используя значения в нескольких предыдущих узлах. Для экстраполяционного метода Адамса аппроксимация примет следующий вид:





При получаем:





Значения в первых трех точках находятся с помощью метода предиктор-корректор того же порядка точности.

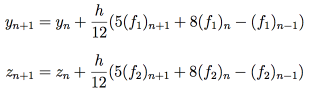
Для интерполяционного метода Адамса аппроксимация примет следующий вид:





При получаем:





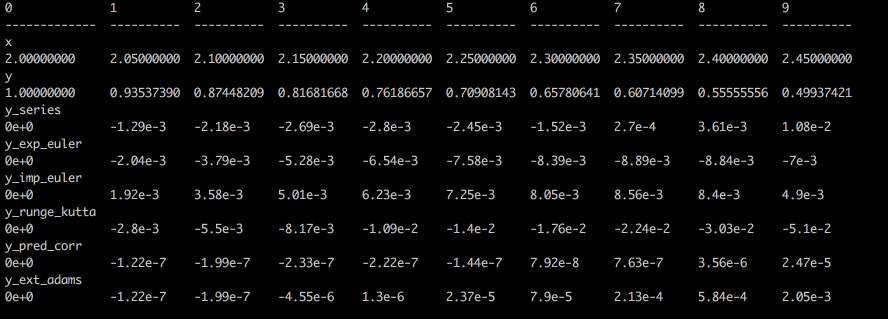
Полученная система решается с помощью МПИ. Итерации проводятся пока норма невязки изменения решения не окажется меньше . Значения в первых трех точках также находятся с помощью метода предиктор-корректор того же порядка точности

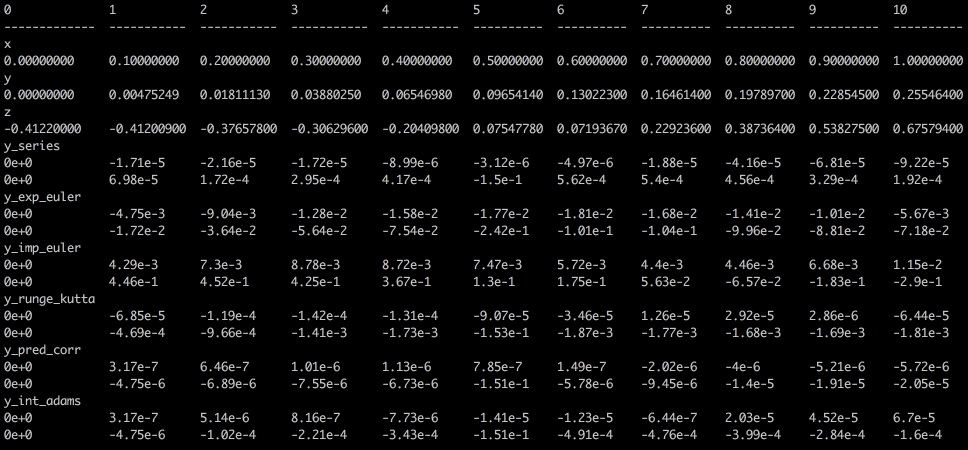
**3. Результаты**

Задача Коши для ОДУ-1:

Задача Коши для системы ОДУ-1.

Точные решения находились с помощью Wolfram Mathematica.





Ожидаемые погрешности методов:

|  |  |  |  |
| --- | --- | --- | --- |
| Метод (ОДУ-1) | Погрешность метода | Метод (ОДУ-1 система) | Погрешность метода |
| Рядов |  | Рядов |  |
| Явный Эйлера |  | Явный Эйлера |  |
| Неявный Эйлера |  | Неявный Эйлера |  |
| Рунге-Кутта |  | Рунге-Кутта |  |
| Предиктор-корректор |  | Предиктор-корректор |  |
| Адамса |  | Адамса |  |

**4. Вывод**

Как видно из сравнительной таблицы ожидаемых погрешностей, все полученные погрешности не выходят за пределы оценок. Однако не во всех методах сохраняется отношение реальной и ожидаемой погрешностей. Одной из причин этого является то, что погрешность оценена асимптотически, и настоящие константы для нее не учитываются. Для системы ОДУ-1 реальные погрешности относятся так же, как ожидаемые. Это говорит о том, что система ОДУ-1 устойчива. Наиболее точными оказались методы предиктор-корректор, Адамса, Рунге-Кутта и рядов. Все эти методы обладают порядком точности не менее 2. Тем не менее метод рядов имеет меньшее практическое применение по сравнению с остальными названными методами, так как для его реализации требуются производные высших порядков исходных функций, нахождение которых не всегда является возможным в случае работы с сеточными функциями.

**4. Листинги**

**require('console.table')  
const {eval, sum, subtract, factorial} = require('mathjs')  
const {last, takeRight} = require('lodash')  
const {sqrt, pow, abs} = Math  
  
const array = n => [...Array(n).keys()]  
const range = (a, b, n = 1) => array(n + 1).map(i => a + (b - a) / n \* i)  
const toFixed = n => v => Number(v.toPrecision(3)).toExponential()  
  
const solution = x => eval(`(sqrt(25 - 4x^2) + 5) / (2x^2)`, {x})  
const yd = [  
 (x, y) => y,  
 (x, y) => eval(`2 x y^3 / (1 - x^2 y^2)`, {x, y}),  
 (x, y) => eval(`2 y^2 (-x^3 y^2 y' + x^2 y^3 + 3 x y' + y) / (1 - x^2 y^2)^2`, {x, y, "y'": yd[1](x, y)}),  
 (x, y) => eval(`(4 y \* (-x \* (x^2 y^2 + 3) \* yd ^2 - x \* (x^2 y^2 + 3) \* y^4 + (x^4 y^4 - 6 x^2 y^2 - 3) \* y yd ) - 2 x y^2 (x^4 y^4 - 4 x^2 y^2 + 3) ydd )/(x^2 y^2 - 1)^3`, {x, y, "yd": yd[1](x, y), "ydd": yd[2](x, y)})  
]  
  
const f = yd[1]  
const EPS = 1E-8  
const v = range(2, 2.45, 9)  
const y0 = 1  
const y = v.map(solution)  
const h = v[1] - v[0]  
  
const taylor = yd => (x, y) => sum(yd.map((f, i) => f(x, y) \* pow(h, i) / factorial(i)))  
  
const newton = (xn, y, yn = taylor(yd.slice(0, 2))(xn - h, y)) => {  
 const fy = yn => eval(`(6 x y^2 - 10 x^3 y^4) / (1 - x^2 y^2)^2`, {x: xn, y: yn})  
 const phi = yn => ((y + h \* f(xn, yn) - h \* yn \* fy(yn)) / (1 - h \* fy(yn)))  
 while (abs(yn - (yn = phi(yn))) > EPS);  
 return yn  
}  
  
const next\_series = ({xn, yn}) => taylor(yd)(xn, yn);  
const next\_exp\_euler = ({xn, yn}) => taylor(yd.slice(0, 2))(xn, yn);  
const next\_imp\_euler = ({xn, yn}) => newton(xn + h, yn)  
const next\_runge\_kutta = ({xn, yn}) => yn + h \* f(xn - h / 2, yn + h / 2 \* f(xn - h, yn))  
const next\_pred\_corr = ({xn, yn}) => {  
 const yn1\_4 = yn + h / 4 \* f(xn, yn)  
 const yn1\_2 = yn + h / 2 \* f(xn + h / 4, yn1\_4)  
 const yn1 = yn + h \* f(xn + h / 2, yn1\_2)   
 return yn + h / 6 \* (f(xn, yn) + 4 \* f(xn + h / 2, yn1\_2) + f(xn + h, yn1))  
}  
  
const run = next => v.slice(1).reduce((yv, x) => {  
 const xn = x - h  
 const yn = last(yv)  
 return [...yv, next({xn, yn})]  
}, [y0])  
  
const y\_series = run(next\_series)  
const y\_exp\_euler = run(next\_exp\_euler)  
const y\_imp\_euler = run(next\_imp\_euler)  
const y\_runge\_kutta = run(next\_runge\_kutta)  
const y\_pred\_corr = run(next\_pred\_corr)  
  
const y\_ext\_adams = v.slice(3).reduce((yv, x) => {  
 const ys = takeRight(yv, 3), xs = [x - 3 \* h, x - 2 \* h, x - h]  
 return [...yv, ys[2] + h / 12 \* (23 \* f(xs[2], ys[2]) - 16 \* f(xs[1], ys[1]) + 5 \* f(xs[0], ys[0]))]  
}, y\_pred\_corr.slice(0, 3))  
  
const toAnswer = obj => {  
 const key = Object.keys(obj)[0]  
 return [[key], subtract(obj[key], y).map(toFixed(8))]  
}  
  
console.table([  
 ['x'], v.map(v => v.toFixed(8)),  
 ['y'], y.map(v => v.toFixed(8)),  
 ...toAnswer({y\_series}),  
 ...toAnswer({y\_exp\_euler}),  
 ...toAnswer({y\_imp\_euler}),  
 ...toAnswer({y\_runge\_kutta}),  
 ...toAnswer({y\_pred\_corr}),  
 ...toAnswer({y\_ext\_adams})  
])**

**require('console.table')  
const {eval, sum, subtract, factorial} = require('mathjs')  
const {last, takeRight} = require('lodash')  
const {sqrt, pow, abs, max} = Math  
  
const array = n => [...Array(n).keys()]  
const range = (a, b, n = 1) => array(n + 1).map(i => a + (b - a) / n \* i)  
const toFixed = n => v => Number(v.toPrecision(3)).toExponential()  
  
let zd = [], yd = []  
  
const solutiony = x => {  
 if (x < 0.05) return 0  
 else if (x < 0.15) return 0.00475249  
 else if (x < 0.25) return 0.0181113  
 else if (x < 0.35) return 0.0388025  
 else if (x < 0.45) return 0.0654698  
 else if (x < 0.55) return 0.0965414  
 else if (x < 0.65) return 0.130223  
 else if (x < 0.75) return 0.164614  
 else if (x < 0.85) return 0.197897  
 else if (x < 0.95) return 0.228545  
 else if (x < 1.05) return 0.255464  
}  
const solutionz = x => {  
 if (x < 0.05) return -0.4122  
 else if (x < 0.15) return -0.412009  
 else if (x < 0.25) return -0.376578  
 else if (x < 0.35) return -0.306296  
 else if (x < 0.45) return -0.204098  
 else if (x < 0.55) return 0.0754778  
 else if (x < 0.65) return 0.0719367  
 else if (x < 0.75) return 0.229236  
 else if (x < 0.85) return 0.387364  
 else if (x < 0.95) return 0.538275  
 else if (x < 1.05) return 0.675794  
}  
  
yd = [  
 (x, y, z) => y,  
 (x, y, z) => eval(`-y z + sin(x) / (1 + x)`, {x, y, z}),  
 (x, y, z) => eval(`-yd z - y zd + cos(x) / (1 + x) - sin(x) / (1 + x)^2`, {x, y, z, yd: yd[1](x, y, z), zd: zd[1](x, y, z)}),  
 (x, y, z) => eval(`-ydd z - 2 yd zd - y zdd + -sin(x) / (1 + x) - 2 cos(x) / (1 + x)^2 + 2 sin(x) / (1 + x)^3`, {x, y, z, yd: yd[1](x, y, z), ydd: yd[2](x, y, z), zd: zd[1](x, y, z), zdd: zd[2](x, y, z)})  
]  
  
zd = [  
 (x, y, z) => z,  
 (x, y, z) => eval(`-z^2 + 3.5 x / (1 + x^2)`, {x, z}),  
 (x, y, z) => eval(`-2 z zd + 3.5 / (1 + x^2) - 7x^2 / (1 + x^2)^2`, {x, z, zd: zd[1](x, y, z)}),  
 (x, y, z) => eval(`-2 z zdd - 2 zd ^ 2 - 7x / (1 + x^2)^2 - 14x / (1 + x^2)^2 + 28 x^3 / (1 + x^2)^3`, {x, z, zd: zd[1](x, y, z), zdd: zd[2](x, y, z)})  
]  
  
const f1 = yd[1]  
const f2 = zd[1]  
const EPS = 1E-8  
const v = range(0, 1, 10)  
const y0 = 0  
const z0 = -0.4122  
const h = v[1] - v[0]  
  
const y = v.map(solutiony)  
const z = v.map(solutionz)  
  
const taylor = fd => (x, y, z) => sum(fd.map((f, i) => f(x, y, z) \* pow(h, i) / factorial(i)))  
  
const newton = (xn1, yn, zn) => {  
 let yn1 = taylor(yd.slice(0, 2))(xn1 - h, yn, zn)  
 let zn1 = taylor(zd.slice(0, 2))(xn1 - h, yn, zn)  
 const f1y = (y, z) => -z  
 const f2z = (y, z) => -2 \* z  
  
 const phi = (f, fd) => (yn1, zn1) => ((yn + h \* f(xn1, yn1, zn1) - h \* yn1 \* fd(yn1, zn1)) / (1 - h \* fd(yn1, zn1)))  
 const phi1 = phi(f1, f1y)  
 const phi2 = phi(f2, f2z)  
 let err = 10  
 while (err > EPS) {  
 let prev\_yn1 = yn1, prev\_zn1 = zn1  
 yn1 = phi1(yn1, zn1)  
 zn1 = phi2(yn1, zn1)  
 err = max(abs(yn1 - prev\_yn1), abs(zn1 - prev\_zn1))  
 }  
 return [yn1, zn1]  
}  
  
const next\_series = ({xn, yn, zn}) => [taylor(yd)(xn, yn, zn), taylor(zd)(xn, yn, zn)];  
const next\_exp\_euler = ({xn, yn, zn}) => [  
 taylor(yd.slice(0, 2))(xn, yn, zn),   
 taylor(zd.slice(0, 2))(xn, yn, zn)  
];  
const next\_imp\_euler = ({xn, yn, zn}) => newton(xn + h, yn, zn)  
const next\_runge\_kutta = ({xn, yn, zn}) => {  
 yn1 = h \* f1(xn, yn, zn)  
 zn1 = h \* f2(xn, yn, zn)  
 yn2 = h \* f1(xn + 2 / 3 \* h, yn + 2 / 3 \* yn1, zn + 2 / 3 \* zn1)  
 zn2 = h \* f2(xn + 2 / 3 \* h, yn + 2 / 3 \* yn1, zn + 2 / 3 \* zn1)  
 return [  
 yn + 1 / 4 \* (yn1 + 3 \* yn2),  
 zn + 1 / 4 \* (zn1 + 3 \* zn2)  
 ]  
}  
const next\_pred\_corr = ({xn, yn, zn}) => {  
 const yn1\_4 = yn + h / 4 \* f1(xn, yn, zn)  
 const zn1\_4 = zn + h / 4 \* f2(xn, yn, zn)  
 const yn1\_2 = yn + h / 2 \* f1(xn + h / 4, yn1\_4, zn1\_4)  
 const zn1\_2 = zn + h / 2 \* f2(xn + h / 4, yn1\_4, zn1\_4)  
 const yn1 = yn + h \* f1(xn + h / 2, yn1\_2, zn1\_2)   
 const zn1 = zn + h \* f2(xn + h / 2, yn1\_2, zn1\_2)   
 return [  
 yn + h / 6 \* (f1(xn, yn, zn) + 4 \* f1(xn + h / 2, yn1\_2, zn1\_2) + f1(xn + h, yn1, zn1)),  
 zn + h / 6 \* (f2(xn, yn, zn) + 4 \* f2(xn + h / 2, yn1\_2, zn1\_2) + f2(xn + h, yn1, zn1)),  
 ]  
}  
  
const run = next => v.slice(1).reduce((yv, x) => {  
 const xn = x - h  
 const [yn, zn] = last(yv)  
 return [...yv, next({xn, yn, zn})]  
}, [[y0, z0]])  
  
const y\_series = run(next\_series)  
const y\_exp\_euler = run(next\_exp\_euler)  
const y\_imp\_euler = run(next\_imp\_euler)  
const y\_runge\_kutta = run(next\_runge\_kutta)  
const y\_pred\_corr = run(next\_pred\_corr)  
  
const y\_int\_adams = v.slice(2).reduce((yv, x) => {  
 const yzs = takeRight(yv, 2), xs = [x - 2 \* h, x - h, x]  
 let yn1 = taylor(yd.slice(0, 2))(xs[1], yzs[1][0], yzs[1][1])  
 let zn1 = taylor(zd.slice(0, 2))(xs[1], yzs[1][0], yzs[1][1])  
  
 let err = 10  
 while (err > EPS) {  
 let prev\_yn1 = yn1, prev\_zn1 = zn1  
 yn1 = yzs[1][0] + h / 12 \* (5 \* f1(xs[2], yn1, zn1) + 8 \* f1(xs[1], yzs[1][0], yzs[1][1]) - f1(xs[0], yzs[0][0], yzs[0][1]))  
 zn1 = yzs[1][1] + h / 12 \* (5 \* f2(xs[2], yn1, zn1) + 8 \* f2(xs[1], yzs[1][0], yzs[1][1]) - f2(xs[0], yzs[0][0], yzs[0][1]))  
 err = max(abs(yn1 - prev\_yn1), abs(zn1 - prev\_zn1))  
 }  
 return [...yv, [yn1, zn1]]  
}, y\_pred\_corr.slice(0, 2))  
  
const toAnswer = obj => {  
 const key = Object.keys(obj)[0]  
 return [  
 [key],   
 subtract(obj[key].map(v => v[0]), y).map(toFixed(8)),  
 subtract(obj[key].map(v => v[1]), z).map(toFixed(8))  
 ]  
}  
  
console.table([  
 ['x'], v.map(v => v.toFixed(8)),  
 ['y'], y.map(v => v.toFixed(8)),  
 ['z'], z.map(v => v.toFixed(8)),  
 ...toAnswer({y\_series}),  
 ...toAnswer({y\_exp\_euler}),  
 ...toAnswer({y\_imp\_euler}),  
 ...toAnswer({y\_runge\_kutta}),  
 ...toAnswer({y\_pred\_corr}),  
 ...toAnswer({y\_int\_adams})  
])**