Лабораторная работа №1.
Численное решение задачи Коши для ОДУ

Выполнил: Киселева Ксения

Группа: 3821Б1ПМоп2 Вариант: 4

1. Постановка задачи
(тестовая, основная №1, основная №2)

Цель: освоение одношаговых методов численного интегрирования задачи Коши для ОДУ и систем ОДУ с элементами оценки погрешности и управления шагом.

Тестовая задача:

$$\begin{cases} \frac{dy}{dx} = 2y \\ y(0) = y_0 \end{cases}$$

Основная задача №1:

$$\begin{cases} \frac{dy}{dx} = \frac{x^3+1}{x^5+1} y^2 + y - y^3 \sin 10x \\ y(0) = y_0 \end{cases}$$

Основная задача №2:

$$\begin{cases} \frac{d^2y}{dx^2} + a y'|y'| + b y' + c y = 0 \\ y(0) = y_0 \\ y'(0) = y_0' \end{cases}$$

2. Краткие сведения по численным метода решения ОДУ

(запись метода, оценка погрешности, управление шагом метода)

Метод Рунге-Кутта IV порядка

$$
\begin{cases}
x_0,\ v_0 = u_0 \\
x_{n+1} = x_n + h_n \\
v_{n+1} = v_n + \frac{h_n}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\
k_1 = f(x_n, v_n) \\
k_2 = f\left(x_n + \frac{h_n}{2},\ v_n + \frac{h_n}{2}k_1\right) \\
k_3 = f\left(x_n + \frac{h_n}{2},\ v_n + \frac{h_n}{2}k_2\right) \\
k_4 = f(x_n + h_n,\ v_n + h_n \cdot k_3)
\end{cases}
$$

Оценка локальной погрешности:

$$e_{n+1} \approx A(x_n, v_n) h^{p+1} \approx 2^p \cdot S, \quad S = \frac{\tilde{v}_{n+1} - v_{n+1}}{2^p - 1}$$

$\tilde{v}_{n+1}$ – получено из v_n с помощью двойного счёта с половин. шагом

Управление шагом

Выбираем $\varepsilon > 0$ – малый параметр для контроля локальной погрешности

1) Если $\frac{\varepsilon}{2^{p+1}} \le |S| \le \varepsilon$, точка (x_{n+1}, v_{n+1}) считается следующей точкой численной траектории, счёт продолжается с тем же шагом

2) Если $|S| < \frac{\varepsilon}{2^{p+1}}$, точка (x_{n+1}, v_{n+1}) считается следующей точкой численной траектории, счёт продолжается с двойным шагом

3) Если $|S| > \varepsilon$, расчеты считаются грубыми, повторяем счёт из точки (x_n, v_n) с половинным шагом

2

3. Исследование порядка сходимости для тестовой задачи

Метод РК IV

h	*погрешность*
$h_1 = 0{,}01$	$2{,}674 \cdot 10^{-11}$
$h_2 = 0{,}02$	$87{,}543 \cdot 10^{-11}$
$h_3 = 0{,}04$	$2931{,}998 \cdot 10^{-11}$
$h_4 = 0{,}08$	$102782{,}16 \cdot 10^{-11}$
Порядок	4

Проверка порядка метода:

$$\frac{87{,}543 \cdot 10^{-11}}{2{,}674 \cdot 10^{-11}} \approx 32 \approx 2^{p+1} \Rightarrow p = 4$$

$$\frac{2931{,}998 \cdot 10^{-11}}{87{,}543 \cdot 10^{-11}} \approx 32 \approx 2^{p+1} \Rightarrow p = 4$$

$$\frac{102782{,}16 \cdot 10^{-11}}{2931{,}998 \cdot 10^{-11}} \approx 32 \approx 2^{p+1} \Rightarrow p = 4$$

3

4. Результаты численных экспериментовдля основных задач (графики, таблицы)

4.1 Основная задача 1

u0	1
x0	0
N	100
b	0,45
step	0,001
epsilon	0,001
ksi	0,00001

Рисунок 1
(Параметры)

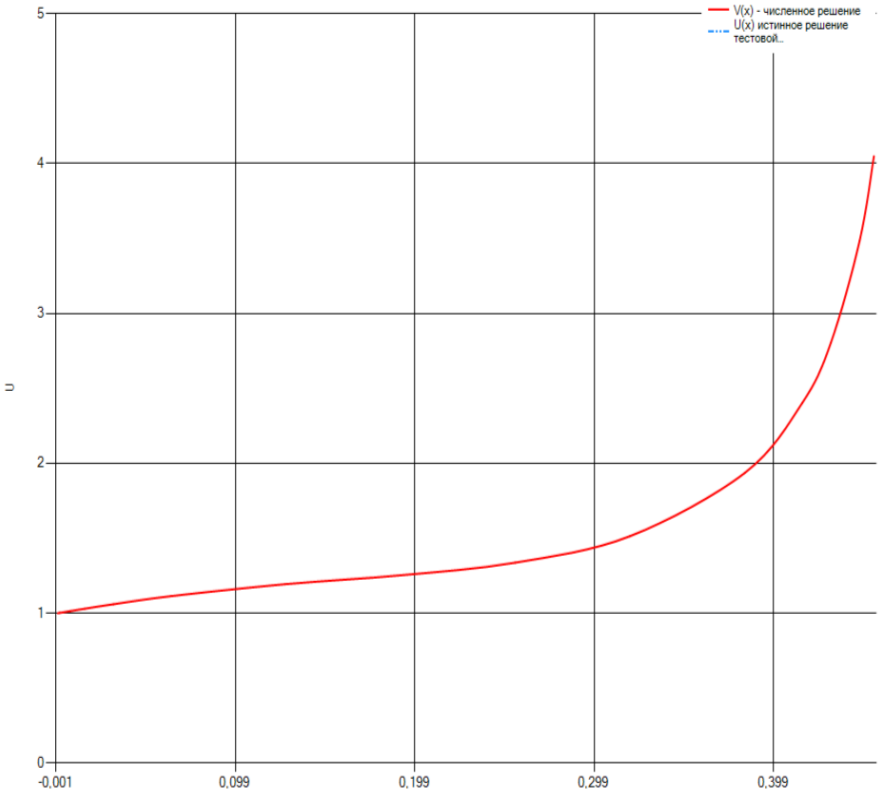

Рисунок 2 (Численный график)

n	x	v1	v2	v1-v2	OLP	h
0	0	1	1	0	0	0,001
1	0,001	1,0019979793431	1,0019979793431	2,44249065417534E-15	2,6053233644537E-15	0,002
2	0,003	1,00598144284753	1,00598144284744	8,30446822419617E-14	8,85809943914258E-14	0,004
3	0,007	1,01389493984127	1,01389493983856	2,70961031390016E-12	2,8902510014935E-12	0,008
4	0,015	1,02948099634196	1,02948099625264	8,9319884821748E-11	9,52745438098646E-11	0,016
5	0,031	1,0594813681483	1,05948136527759	2,87070833770997E-09	3,0620888935573E-09	0,032
6	0,063	1,11340673833402	1,11340666880801	6,95260073957371E-08	7,41610745554529E-08	0,064
7	0,127	1,19325468941764	1,19325668761655	-1,99819890944752E-06	2,13141217007736E-06	0,128
8	0,255	1,33626902021647	1,33677335792289	-0,00050433770641822 3	0,00053796022017943 8	0,128
9	0,383	1,93250669810748	1,93490035187291	-0,00239365376542877	0,00255323068312402	0,128

Рисунок 3 (Таблица с численным решением)

4.2 Основная задача 2

u0	1
x0	0
N	100
b	20
step	0,001
epsilon	0,0001
ksi	0,00001

a	1
b	1
c	1

Рисунки 1, 2
(Параметры)

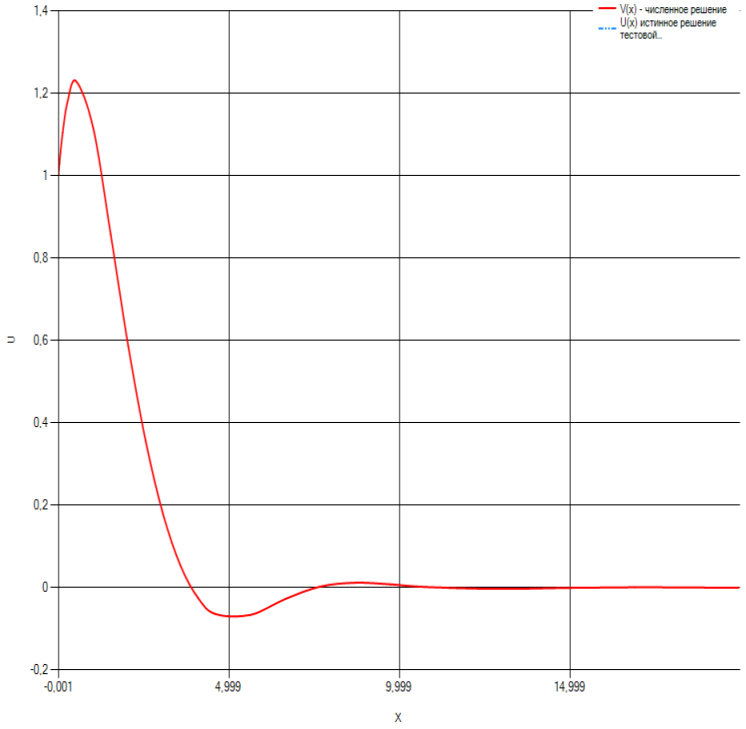

Рисунок 3 (График численного решения)

n	x	v1	v2	v1-v2	OLP	h
0	0	1	1	0	0	0,001
1	0,001	1,00099850133171	1,00099850133171	-4,44089209850063E-16	4,736951571734E-16	0,002
2	0,003	1,00298653586887	1,00298653586888	-1,62092561595273E-14	1,72898732368291E-14	0,004
3	0,007	1,00692695346626	1,00692695346676	-5,02931030155196E-13	5,36459765498876E-13	0,008
4	0,015	1,0146669192869	1,0146669193022	-1,53075330189267E-11	1,63280352201885E-11	0,016
5	0,031	1,02959677807073	1,02959677851386	-4,43127534666132E-10	4,72669370310541E-10	0,032
6	0,063	1,05735621394392	1,05735622556777	-1,16238521030709E-08	1,2398775576609E-08	0,064
7	0,127	1,10517419703318	1,10517444810635	-2,51073168477944E-07	2,67811379709807E-07	0,128
8	0,255	1,17439200197063	1,17439566023349	-3,65826285908533E-06	3,90214704969101E-06	0,256
9	0,511	1,2301648827916	1,23018390362336	-1,90208317556451E-05	2,02888872060214E-05	0,512
10	1,023	1,11513006522641	1,11487369830634	0,00025636692006991 7	0,00027345804807 4578	0,512
11	1,535	0,857108558375261	0,857771336558113	-0,00066277818285132	0,00070696339504 1408	0,512
12	2,047	0,587172000230408	0,587616794305625	-0,00044479407521 7136	0,00047444701356 4946	0,512
13	2,559	0,356139722024471	0,356348559500363	-0,00020883747589262	0,00022275997428 5462	0,512
14	3,071	0,178036799134986	0,178153567756386	-0,00011676862139922 7	0,00012455319615 9175	0,512
15	3,583	0,0529221267328233	0,0529996267134465	-7,74999806231955E-05	8,26666459980752E-05	0,512
16	4,095	-0,0249255924657229	-0,0248706197744193	-5,49726913035842E-05	5,86375373904898E-05	0,512
17	4,607	-0,0639872069260541	-0,0639477922033199	-3,94147227341241E-05	4,2042370916399E-05	1,024
18	5,631	-0,066060919306206	-0,0652283609271859	-0,00083255837902010 4	0,00088062270954778	1,024
19	6,655	-0,0273006040832722	-0,0271204634112484	-0,00018014067202379 6	0,00019215005015871 6	1,024
20	7,679	0,00278127276617176	0,00261527486948485	0,00016599789668690 8	0,00017706442313270 2	1,024

Рисунок 4 (Таблица с численным решением)

_5. Код программы

```
class RK4
{
public:
  explicit RK4(const LD& _x0 = NULL, const LD& _u0 = 1.0,
               const LD& _du0 = 1.0) : x0(_x0), y0(_u0), y1(_du0) {}
  virtual ~RK4() = default;

  void run_const_step(size_t n = 10000, const LD& step = 0.001, const LD& accuracy = 1e1,
          const LD& b = INFINITY, const std::tuple<double, double, double>& parameters = { 1, 1, 1 })
  {
    if (!y.empty())
    {
      y.clear();
      y2.clear();
      dy.clear();
      dy2.clear();
      estimate.clear();
      c.clear();
    }
    LD x = x0;
    LD y = y0;
    LD y1 = this->y1;
    LD y2 = y0;
    LD y21 = this->y1;
    LD s = NULL;

    for (size_t i = 0; i < n && abs(x) < b; ++i)
    {
      this->y.emplace_back(x, y);
      this->dy.emplace_back(x, y1);
      this->y2.emplace_back(x, y2);
      this->dy2.emplace_back(x, y21);
      this->estimate.push_back(s);
      y2 = y;
      y21 = y1;

      auto kq = this->k(x, y1, y, step, parameters);
      y += (std::get<0>(kq)[0] + 2 * std::get<0>(kq)[1] + 2 * std::get<0>(kq)[2] + std::get<0>(kq)[3]) / 6.0;
      y1 += (std::get<1>(kq)[0] + 2 * std::get<1>(kq)[1] + 2 * std::get<1>(kq)[2] + std::get<1>(kq)[3]) / 6.0;

      kq = this->k(x, y21, y2, step / 2.0, parameters);
      y2 += (std::get<0>(kq)[0] + 2 * std::get<0>(kq)[1] + 2 * std::get<0>(kq)[2] + std::get<0>(kq)[3]) / 6.0;
      y21 += (std::get<1>(kq)[0] + 2 * std::get<1>(kq)[1] + 2 * std::get<1>(kq)[2] + std::get<1>(kq)[3]) / 6.0;

      kq = this->k(x + step / 2.0, y21, y2, step / 2.0, parameters);
      y2 += (std::get<0>(kq)[0] + 2 * std::get<0>(kq)[1] + 2 * std::get<0>(kq)[2] + std::get<0>(kq)[3]) / 6.0;
      y21 += (std::get<1>(kq)[0] + 2 * std::get<1>(kq)[1] + 2 * std::get<1>(kq)[2] + std::get<1>(kq)[3]) / 6.0;

      s = (y2 - y1) / 15.0;
      x += step;
      if (fabs(y1) > accuracy || fabs(y2) > accuracy || isnan(s))
        break;
    }
  }

  std::vector<LD> run_non_const_step(const size_t& n = 10000, const LD& step = 0.01,
                                     const LD& epsilon = 1e-6, const LD& accuracy = 1e1,
                                     const LD& b = INFINITY, const std::tuple<double, double, double>& parameters = { NULL, NULL, NULL }, const LD ksi = 0)
  {
    if (!y.empty())
    {
      y.clear();
      y2.clear();
      dy.clear();
      dy2.clear();
      estimate.clear();
      c.clear();
    }
    LD x = x0;
    LD y = y0;
    LD y1 = this->y1;
    LD y2 = y0;
    LD y21 = this->y1;
    LD s = NULL;

    std::vector<LD> steps{ step };
    this->y.emplace_back(x, y);
    this->dy.emplace_back(x, y1);
    this->y2.emplace_back(x, y2);
    this->dy2.emplace_back(x, y21);
    this->estimate.push_back(s);
    c.emplace_back(NULL, NULL, NULL);

    for (size_t i = 0; i < n && abs(x + steps[i]) < b; ++i)
    {
      if (x > b - ksi)
        break;
      y2 = y;
      y21 = y1;

      auto kq = this->k(x, y1, y, steps[i], parameters);
      y += (std::get<0>(kq)[0] + 2 * std::get<0>(kq)[1] + 2 * std::get<0>(kq)[2] + std::get<0>(kq)[3]) / 6.0;
      y1 += (std::get<1>(kq)[0] + 2 * std::get<1>(kq)[1] + 2 * std::get<1>(kq)[2] + std::get<1>(kq)[3]) / 6.0;

      kq = this->k(x, y21, y2, steps[i] / 2.0, parameters);
      y2 += (std::get<0>(kq)[0] + 2 * std::get<0>(kq)[1] + 2 * std::get<0>(kq)[2] + std::get<0>(kq)[3]) / 6.0;
      y21 += (std::get<1>(kq)[0] + 2 * std::get<1>(kq)[1] + 2 * std::get<1>(kq)[2] + std::get<1>(kq)[3]) / 6.0;

      kq = this->k(x + steps[i] / 2.0, y21, y2, steps[i] / 2.0, parameters);
      y2 += (std::get<0>(kq)[0] + 2 * std::get<0>(kq)[1] + 2 * std::get<0>(kq)[2] + std::get<0>(kq)[3]) / 6.0;
      y21 += (std::get<1>(kq)[0] + 2 * std::get<1>(kq)[1] + 2 * std::get<1>(kq)[2] + std::get<1>(kq)[3]) / 6.0;

      s = (y - y2) / 15.0;
      if (fabs(s) > epsilon)
      {
        steps[i] /= 2.0;
        y = std::get<1>(this->y[i]);
        y1 = std::get<1>(this->dy[i]);
        std::get<0>(c[i]) += 1;
        --i;
        continue;
```

```
    else if (fabs(s) > epsilon / 32.0 && fabs(s) - epsilon < 1e-32)
    {
      steps.push_back(steps[i]);
    }
    else
    {
      steps.push_back(steps[i] * 2.0);
      std::get<1>(c[i]) += 1;
    }
    x += steps[i];
    if (fabs(y) > accuracy || fabs(y2) > accuracy)
      break;
    if (fabs(y1) > b)
      break;
    this->y.emplace_back(x, y);
    this->y2.emplace_back(x, y2);
    this->dy.emplace_back(x, y1);
    this->dy2.emplace_back(x, y21);
    estimate.push_back(fabs(s) * 16);
    c.emplace_back(NULL, NULL, NULL);
  }
  if (x < b - ksi && fabs(y) < accuracy && fabs(y2) < accuracy && steps.size() < n)
  {
    steps.push_back(steps[steps.size() - 1] / 2.0);
    auto kq = this->k(x, y1, y, steps[steps.size() - 1], parameters);
    y += (std::get<0>(kq)[0] + 2 * std::get<0>(kq)[1] + 2 * std::get<0>(kq)[2] + std::get<0>(kq)[3]) / 6.0;
    y1 += (std::get<1>(kq)[0] + 2 * std::get<1>(kq)[1] + 2 * std::get<1>(kq)[2] + std::get<1>(kq)[3]) / 6.0;

    kq = this->k(x, y21, y2, steps[steps.size() - 1] / 2.0, parameters);
    y2 += (std::get<0>(kq)[0] + 2 * std::get<0>(kq)[1] + 2 * std::get<0>(kq)[2] + std::get<0>(kq)[3]) / 6.0;
    y21 += (std::get<1>(kq)[0] + 2 * std::get<1>(kq)[1] + 2 * std::get<1>(kq)[2] + std::get<1>(kq)[3]) / 6.0;

    kq = this->k(x + steps[steps.size() - 1] / 2.0, y21, y2, steps[steps.size() - 1] / 2.0, parameters);
    y2 += (std::get<0>(kq)[0] + 2 * std::get<0>(kq)[1] + 2 * std::get<0>(kq)[2] + std::get<0>(kq)[3]) / 6.0;
    y21 += (std::get<1>(kq)[0] + 2 * std::get<1>(kq)[1] + 2 * std::get<1>(kq)[2] + std::get<1>(kq)[3]) / 6.0;

    s = (y - y2) / 15.0;
    x += steps[steps.size() - 1];
    this->y.emplace_back(x, y);
    this->y2.emplace_back(x, y2);
    this->dy.emplace_back(x, y1);
    this->dy2.emplace_back(x, y21);
    estimate.push_back(fabs(s) * 16);
    c.emplace_back(NULL, NULL, NULL);
  }
  return steps;
}
public:
  const VECTOR_tuple& get_y() const
  {
    return y;
  }
  const VECTOR_tuple& get_y2() const
  {
    return y2;
  }
  const VECTOR_tuple& get_dy() const
  {
    return dy;
  }

  const VECTOR_tuple& get_dy2() const
  {
    return dy2;
  }

  const std::vector<std::tuple<int, int, int>>& get_c() const
  {
    return c;
  }

  const std::vector<LD>& get_estimates() const
  {
    return estimate;
  }

  virtual const VECTOR_tuple& get_anal_solution()
  {
    throw "The DE can't solved with default analytic methods";
  };

protected:
  virtual LD f1(const LD& x, const LD& dy,
                const LD& y, const std::tuple<double, double, double>& parameters) = NULL;

  virtual LD f2(const LD& x, const LD& y1, const LD& y) = NULL;

  tuple_VEC k(const LD& x, const LD& y1, const LD& y,
           const LD& step, const std::tuple<double, double, double>& parameters)
  {
    const LD k1 = step * f2(x, y1, y), m1 = step * f1(x, y1, y, parameters), s1 = step * f1(x, y1, y, parameters),
      k2 = step * f2(x + step / 2.0, y1 + m1 / 2.0, y + k1 / 2.0), m2 = step * f1(x + step / 2.0, y1 + m1 / 2.0, y + k1 / 2.0, parameters), s2 = step * f1(x, y1, y, parameters),
      k3 = step * f2(x + step / 2.0, y1 + m2 / 2.0, y + k2 / 2.0), m3 = step * f1(x + step / 2.0, y1 + m2 / 2.0, y + k2 / 2.0, parameters), s3 = step * f1(x, y1, y, parameters),
      k4 = step * f2(x + step, y1 + m3, y + k3), m4 = step * f1(x + step, y1 + m3, y + k3, parameters), s4 = step * f1(x, y1, y, parameters);
    return tuple_VEC{ std::vector<LD>{k1, k2, k3, k4}, std::vector<LD>{m1, m2, m3, m4},std::vector<LD>{s1, s2, s3, s4} };
  }


class RK4_main2 : public RK4
{
public:
  RK4_main2(const LD& _x0 = NULL, const LD& _u0 = 1.0,
    const LD& _du0 = 1.0) : RK4(_x0, _u0, _du0)
  {}
  RK4_main2(const RK4_main2& cp) = default;
  ~RK4_main2() override = default;

protected:
  LD f1(const LD& x, const LD& dy,
                const LD& y, const std::tuple<double, double,  double>& parameters) override
  {
    return -(dy * fabs(dy) * std::get<0>(parameters) + dy * std::get<1>(parameters) + y* std::get<2>(parameters));
  }

  LD f2(const LD& x, const LD& y1, const LD& y) override
  {
    return y1;
  }
};

class RK4_main1 : public RK4
{
public:
  RK4_main1(const LD& _x0 = NULL, const LD& _u0 = 1.0) : RK4(_x0, _u0, NULL)
  {}
  RK4_main1(const RK4_main1& cp) = default;
  ~RK4_main1() override = default;
protected:
  LD f1(const LD& x, const LD& dy,
        const LD& y, const std::tuple<double, double, double>& parameters) override
  {
    return NULL;
  }

  LD f2(const LD& x, const LD& y1, const LD& y) override
  {
    return ((pow(y, 2) *(pow(x,3) + 1)) / (pow(x, 5) + 1)) + y - pow(y, 3) * sin(10 * x);
  }
};
```

```cpp
class RK4_test1 : public RK4
{
public:
  RK4_test1(const LD& _x0 = NULL, const LD& _u0 = 1.0) : RK4(_x0, _u0, NULL), coeff(_u0 / exp(4.0 / 2.0 * _x0))
  {}
  RK4_test1(const RK4_test1& cp) = default;
  ~RK4_test1() override = default;

  const VECTOR_tuple& get_anal_solution() override
  {
    enter_solution();
    return anal_sol;
  }

private:
  void enter_solution()
  {
    for (size_t i = 0; i < y.size(); ++i)
    {
      anal_sol.push_back({ std::get<0>( y[i]), true_func(std::get<0>(y[i]))});
    }
  }

protected:
  LD f1(const LD& x, const LD& dy,
    const LD& y, const std::tuple<double, double, double>& parameters) override
  {
    return NULL;
  }

  LD f2(const LD& x, const LD& y1, const LD& y) override
  {
    return 4.0 / 2.0 * y;
  }

  LD true_func(const LD& x) const
  {
    return coeff * exp(4.0 / 2.0 * x);
  }
private:
  VECTOR_tuple anal_sol;
  LD coeff;
};
```