Московский авиационный институт
(Национальный Исследовательский Институт)

Институт №8 информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

# Лабораторная работа №4
# по курсу «Численные методы»

Студент: Суханов Е.А

Группа: М8О-406Б-19

Оценка:

Преподаватель: Ревизников Д.Л.

Подпись:

Москва
2022

# Задание 4.1

Реализовать методы Эйлера, Рунге-Кутты и Адамса 4-го порядка в виде программ, задавая в качестве входных данных шаг сетки h . С использованием разработанного программного обеспечения решить задачу Коши для ОДУ 2- го порядка на указанном отрезке. Оценить погрешность численного решения с использованием метода Рунге – Ромберга и путем сравнения с точным решением.

Вариант:

$$4xy''-(4x^2+2)*y=0$$

$$y(0)=1, y'(0)=1$$

$$x\in[0,1], h=0.1,0.05$$

# Листинг

main.go:

```
package main

import (
  "fmt"
  "math"
)

type Point struct {
  x, y float64
}
type fn func(x, y, dydx float64) float64

func eulerMethod(f, g fn, a, b, h, y0, dydx0 float64) []Point {
  res := make([]Point, 1)
  res[0] = Point{
    x: a,
    y: y0,
  }

  b -= h
  for zk, yk, xk := dydx0, y0, a; xk < b; xk += h {
    zk += h * f(xk, yk, zk)
    yk += h * g(xk, yk, zk)

    res = append(res, Point{
      x: xk + h,
      y: yk,
```

```
    })
  }

  return res
}

func rungeKuttaMethod(f, g fn, a, b, h, y0, dydx0 float64) []Point {
  res := make([]Point, 1)
  res[0] = Point{
    x: a,
    y: y0,
  }

  b -= h
  for zk, yk, xk := dydx0, y0, a; xk <= b; xk += h {
    k1 := h * f(xk, yk, zk)
    l1 := h * g(xk, yk, zk)

    k2 := h * f(xk+h/2, yk+l1/2, zk+k1/2)
    l2 := h * g(xk+h/2, yk+l1/2, zk+k1/2)

    k3 := h * f(xk+h/2, yk+l2/2, zk+k2/2)
    l3 := h * g(xk+h/2, yk+l2/2, zk+k2/2)

    k4 := h * f(xk+h, yk+l3, zk+k3)
    l4 := h * g(xk+h, yk+l3, zk+k3)

    zk += (k1 + 2*k2 + 2*k3 + k4) / 6
    yk += (l1 + 2*l2 + 2*l3 + l4) / 6

    res = append(res, Point{
      x: xk + h,
      y: yk,
    })
  }

  return res
}

func adamsMethod(f, g fn, a, b, h, y0, dydx0 float64) []Point {
  res := make([]Point, 1)
  res[0] = Point{
    x: a,
    y: y0,
  }

  x := make([]float64, 1)
  y := make([]float64, 1)
  z := make([]float64, 1)
  x[0] = a
  y[0] = y0
```

```
  z[0] = dydx0

  // b -= h
  for zk, yk, xk, b := dydx0, y0, a, a+2*h; xk <= b; xk += h {
    k1 := h * f(xk, yk, zk)
    l1 := h * g(xk, yk, zk)

    k2 := h * f(xk+h/2, yk+l1/2, zk+k1/2)
    l2 := h * g(xk+h/2, yk+l1/2, zk+k1/2)

    k3 := h * f(xk+h/2, yk+l2/2, zk+k2/2)
    l3 := h * g(xk+h/2, yk+l2/2, zk+k2/2)

    k4 := h * f(xk+h, yk+l3, zk+k3)
    l4 := h * g(xk+h, yk+l3, zk+k3)

    zk += (k1 + 2*k2 + 2*k3 + k4) / 6
    yk += (l1 + 2*l2 + 2*l3 + l4) / 6

    x = append(x, xk+h)
    y = append(y, yk)
    z = append(z, zk)
    res = append(res, Point{
      x: xk + h,
      y: yk,
    })
  }

  b -= h
  for xk, yk, zk := x[3], y[3], z[3]; xk <= b; xk += h {
    n := len(x) - 1
    zk += (55*f(x[n], y[n], z[n]) -
      59*f(x[n-1], y[n-1], z[n-1]) +
      37*f(x[n-2], y[n-2], z[n-2]) -
      9*f(x[n-3], y[n-3], z[n-3])) * h / 24
    yk += (55*g(x[n], y[n], z[n]) -
      59*g(x[n-1], y[n-1], z[n-1]) +
      37*g(x[n-2], y[n-2], z[n-2]) -
      9*g(x[n-3], y[n-3], z[n-3])) * h / 24

    x = append(x, xk+h)
    y = append(y, yk)
    z = append(z, zk)
    res = append(res, Point{
      x: xk + h,
      y: yk,
    })
  }

  return res
}
```

```
func rungeRomberg(h1, h2 float64, p1, p2 []Point, p float64) []Point {
  k := h1 / h2
  div := math.Pow(k, p) - 1
  res := make([]Point, len(p1))
  for i := 0; i < len(p1); i++ {
    res[i].x = p1[i].x
    res[i].y = (p2[i].y - p1[i].y) / div
  }

  return res
}

func err(p []Point, anf fn) []Point {
  res := make([]Point, len(p))
  for i := 0; i < len(p); i++ {
    res[i].x = p[i].x
    res[i].y = math.Abs(p[i].y - anf(p[i].x, 0, 0))
  }
  return res
}

func main() {

  var (
    anf   fn         = func(x, y, dydx float64) float64 { return (1 + x) *
math.Exp(x*x) }
    f     fn         = func(x, y, z float64) float64 { return 4*x*z - (4*x*x+2)*y }
    g     fn         = func(x, y, z float64) float64 { return z }
    a     float64    = 0
    b     float64    = 1
    h     []float64 = []float64{0.1, 0.05}
    y0    float64    = 1
    dydx0 float64    = 1
  )
  {
    p1 := eulerMethod(f, g, a, b, h[0], y0, dydx0)
    fmt.Println("Метод эйлера: h: ", h[0], "\n", p1)

    p2 := eulerMethod(f, g, a, b, h[1], y0, dydx0)
    fmt.Println("Метод эйлера: h: ", h[1], "\n", p2)

    fmt.Println("Точность:\n", "\tРунге-Ромберг:\t", rungeRomberg(h[0], h[1], p1,
p2, 4), "\nОностиельно точного решения:\n\th: ", h[0], " err:\t", err(p1, anf), "\
n\th: ", h[1], " err:\t", err(p2, anf))
  }
  {
    p1 := rungeKuttaMethod(f, g, a, b, h[0], y0, dydx0)
    fmt.Println("Метод Рунге-Кутты четвертого порядка: h: ", h[0], "\n", p1)

    p2 := rungeKuttaMethod(f, g, a, b, h[1], y0, dydx0)
```

```
    fmt.Println("Метод Рунге-Кутты четвертого порядка: h: ", h[1], "\n", p2)

    fmt.Println("Точность:\n", "\tРунге-Ромберг:\t", rungeRomberg(h[0], h[1], p1,
p2, 4), "\nОтностиельно точного решения:\n\th: ", h[0], " err:\t", err(p1, anf), "\
n\th: ", h[1], " err:\t", err(p2, anf))
  }
  {
    p1 := adamsMethod(f, g, a, b, h[0], y0, dydx0)
    fmt.Println("Метод Адамса четвертого порядка: h: ", h[0], "\n", p1)

    p2 := adamsMethod(f, g, a, b, h[1], y0, dydx0)
    fmt.Println("Метод Адамса четвертого порядка: h: ", h[1], "\n", p2)

    fmt.Println("Точность:\n", "\tРунге-Ромберг:\t", rungeRomberg(h[0], h[1], p1,
p2, 4), "\nОтностиельно точного решения:\n\th: ", h[0], " err:\t", err(p1, anf), "\
n\th: ", h[1], " err:\t", err(p2, anf))
  }
}
```

# Вывод программы

```
Метод эйлера: h:  0.1

 [{0 1} {0.1 1.08} {0.2 1.141168} {0.30000000000000004 1.1825802112} {0.4
1.20105299475968} {0.5 1.1907736246272531} {0.6 1.1427151717295234} {0.7
1.043813288228843} {0.7999SS999999999999 0.8758838711341097} {0.8999999999999999
0.6142767360453465} {0.9999999999999999 0.22630293135585244}]

Метод эйлера: h:  0.05

 [{0 1} {0.05 1.045} {0.1 1.085198875} {0.15000000000000002 1.1206672132375} {0.2
1.1513441154329591} {0.25 1.1770308354928987} {0.3 1.1973810931061875} {0.35
1.2118878177269472} {0.39999999999999997 1.2198660114058097} {0.44999999999999996
1.220431344903703} {0.49999999999999994 1.212474028218458} {0.5499999999999999
1.1946274246530504} {0.6 1.1652308096126072} {0.65 1.1222856158046421}
{0.7000000000000001 1.063404461995844} {0.7500000000000001 0.9857522424800552}
{0.8000000000000002 0.8859785724605475} {0.8500000000000002 0.7601409595118682}
{0.9000000000000002 0.6036182291318809} {0.9500000000000003 0.41101400848186825}]

Точность:

        Рунге-Ромберг:   [{0 0} {0.1 -0.0023333333333333426} {0.2 -
0.0037312750000000044} {0.30000000000000004 -0.004127533197499996} {0.4 -
0.0033139252884480572} {0.5 -0.0009161859422902966} {0.6 0.003644394758444269} {0.7
0.011204968633206953} {0.7999999999999999 0.022932142684780003} {0.8999999999999999
0.04041030725722376} {0.9999999999999999 0.06574473979084038}]

Относиельно точного решения:

        h:  0.1  err:    [{0 0} {0.1 0.031055183792584984} {0.2 0.1078049290308658}
{0.30000000000000004 0.23984635761677375} {0.4 0.4418622246288544} {0.5
0.735264500404359} {0.6 1.1506118915670211} {0.7 1.7311242856953013}
```

{0.7999999999999999 2.537781711614802} {0.8999999999999999 3.6567484386399487}
{0.9999999999999999 5.210260725562237}]

h:  0.05  err:   [{0 0} {0.05 0.00762828398608506} {0.1
0.02585630879258516} {0.15000000000000002 0.05550107605161281} {0.2
0.09762881359790665} {0.25 0.15358723815442588} {0.3 0.2250454757105862} {0.35
0.3140429943729679} {0.39999999999999997 0.4230492079827246} {0.44999999999999996
0.5550357785230733} {0.4999999999999994 0.713564096813154} {0.5499999999999999
0.9028909737997661} {0.6 1.1280962536839374} {0.65 1.3952368965410695}
{0.7000000000000001 1.7115331119283006} {0.7500000000000001 2.085593407200468}
{0.8000000000000002 2.5276870102883664} {0.8500000000000002 3.0500741208744024}
{0.9000000000000002 3.667406945553417} {0.9500000000000003 4.397217624145518}]

Метод Рунге-Кутты четвертого порядка: h:  0.1

[{0 1} {0.1 1.090249575} {0.2 1.1613060530246289} {0.30000000000000004
1.2119705364153124} {0.4 1.2385071235174996} {0.5 1.2339998494300153} {0.6
1.1873425436600522} {0.7 1.0817314689596087} {0.7999999999999999
0.8924743109878579} {0.8999999999999999 0.5838549756489031} {0.9999999999999999
0.10469948503015652}]

Метод Рунге-Кутты четвертого порядка: h:  0.05

[{0 1} {0.05 1.0475364517578125} {0.1 1.0902494230477204} {0.15000000000000002
1.1281976206706923} {0.2 1.1613056494929208} {0.25 1.189354744049456} {0.3
1.21196971531513} {0.35 1.2286016281367922} {0.39999999999999997
1.2385056057551411} {0.44999999999999996 1.2407130195093874} {0.4999999999999994
1.2339971610928773} {0.5499999999999999 1.2168313081412012} {0.6 1.187337877757751}
{0.65 1.1432271129902982} {0.7000000000000001 1.0817234606099897}
{0.7500000000000001 0.9994774717477111} {0.8000000000000002 0.8924606881994106}
{0.8500000000000002 0.7558405669611842} {0.9000000000000002 0.5838320479867578}
{0.9500000000000003 0.36952189542848657}]

Точность:

Рунге-Ромберг:   [{0 0} {0.1 -0.0028475415494791712} {0.2 -
0.004737108665127229} {0.30000000000000004 -0.00558486104964134} {0.4 -
0.005146764934971918} {0.5 -0.002976340358703947} {0.6 0.001641811443671859} {0.7

0.009791343945145565} {0.7999999999999999 0.02306875298448555} {0.8999999999999999
0.04379053625736562} {0.9999999999999999 0.07528651173751472}]

Остностиельно точного решения:

h:  0.1  err:    [{0 0} {0.1 0.020805608792584973} {0.2 0.0876668760062369}
{0.30000000000000004 0.2104560324014612 7} {0.4 0.40440809587103477} {0.5
0.6920382756015968} {0.6 1.1059845196364924} {0.7 1.6932061049645355}
{0.7999999999999999 2.5211912717610536} {0.8999999999999999 3.6871701990363923}
{0.9999999999999999 5.331864171887933}]

h:  0.05  err:   [{0 0} {0.05 0.005091832228272475} {0.1
0.02080576074486462} {0.15000000000000002 0.047970668618420476} {0.2
0.08766727953794495} {0.25 0.14126332959786847} {0.3 0.21045685350164356} {0.35
0.2973291839631229} {0.39999999999999997 0.40440961363339323} {0.44999999999999996
0.5347541039173889} {0.4999999999999994 0.6920409639387348} {0.5499999999999999
0.8806870903116153} {0.6 1.1059891855387936} {0.65 1.3742953993554134}
{0.7000000000000001 1.693214113314155} {0.7500000000000001 2.071868177932812}

{0.8000000000000002 2.5212048945495034} {0.8500000000000002 3.054374513425086}
{0.9000000000000002 3.6871931266985403} {0.9500000000000003 4.4387097371988995}]

Метод Адамса четвертого порядка: h:  0.1

 [{0 1} {0.1 1.090249575} {0.2 1.1613060530246289} {0.30000000000000004
1.2119705364153124} {0.4 1.2386742508701516} {0.5 1.2346596208054759} {0.6
1.1887487531714054} {0.7 1.0843388526512427} {0.7999999999999999
0.8969583897324218} {0.8999999999999999 0.5911557219263768} {0.9999999999999999
0.11615825759592618}]

Метод Адамса четвертого порядка: h:  0.05

 [{0 1} {0.05 1.0475364517578125} {0.1 1.0902494230477204} {0.15000000000000002
1.1281976206706923} {0.2 1.1613082202872418} {0.25 1.1893638109922635} {0.3
1.2119869510800978} {0.35 1.2286306100056448} {0.39999999999999997
1.2385502732055012} {0.44999999999999996 1.2407778570844812} {0.49999999999999994
1.2340875665689741} {0.5499999999999999 1.216953694432017} {0.6 1.187499907644062}
{0.65 1.1434379616080075} {0.7000000000000001 1.0819941087946296}
{0.7500000000000001 0.999821052060639} {0.8000000000000002 0.8928928782169837}
{0.8500000000000002 0.7563800295477111} {0.9000000000000002 0.5845009141029406}
{0.9500000000000003 0.370346275319686}]

Точность:

        Рунге-Ромберг:   [{0 0} {0.1 -0.0028475415494791712} {0.2 -
0.004737108665127229} {0.30000000000000004 -0.00558486104964134} {0.4 -
0.005157735372193987} {0.5 -0.003019720654214157} {0.6 0.0015492131939128277} {0.7

0.009619450490293469} {0.7999999999999999 0.02277279223153862} {0.8999999999999999
0.043308142343873625} {0.9999999999999999 0.0745286205982032}]

Относиельно точного решения:

        h:  0.1  err:    [{0 0} {0.1 0.020805608792584973} {0.2 0.087666760062369}
{0.30000000000000004 0.21045603240146127} {0.4 0.40424096851838276} {0.5
0.6913785042261362} {0.6 1.1045783101251392} {0.7 1.6905987212729015}
{0.7999999999999999 2.51670719301649} {0.8999999999999999 3.6798694527589184}
{0.9999999999999999 5.320405399322163}]

        h:  0.05  err:   [{0 0} {0.05 0.005091832228272475} {0.1
0.020805760744864 62} {0.15000000000000002 0.047970668618420476} {0.2
0.08766470874362398} {0.25 0.14125426265506102} {0.3 0.21043961773667585} {0.35
0.2973002020942703} {0.39999999999999997 0.4043649461830332} {0.44999999999999996
0.5346892663422951} {0.4999999999999994 0.691950558462638} {0.5499999999999999
0.8805647040207996} {0.6 1.1058271556524826} {0.65 1.3740845507377042}
{0.7000000000000001 1.692943465129515} {0.7500000000000001 2.0715245976198844}
{0.8000000000000002 2.52077270453193} {0.8500000000000002 3.0538350508385594}
{0.9000000000000002 3.6865242605823574} {0.9500000000000003 4.4378853573077}]

# Задание 4.2

Реализовать метод стрельбы и конечно-разностный метод решения краевой задачи для ОДУ в виде программ. С использованием разработанного программного обеспечения решить краевую задачу для обыкновенного дифференциального уравнения 2-го порядка на указанном отрезке. Оценить погрешность численного решения с использованием метода Рунге – Ромберга и путем сравнения с точным решением.

Вариант:

| | |
|---|---|
| $y''-2(1+(tgx)^2)y=0,$<br>$y(0)=0,$<br>$y(\frac{\pi}{6})=-\frac{\sqrt{3}}{3}$ | $y(x)=-tgx$ |

# Листинг

main.go:

```
package main

import (
  "fmt"
  "math"

  "github.com/Reterer/number_methods/internal/run_through"
  "github.com/Reterer/number_methods/pkg/matrix"
)

type fn func(x, y, z float64) float64
type Point struct {
  x, y float64
}

func rungeKuttaMethod(f, g fn, a, b, h, y0, dydx0 float64) []Point {
  res := make([]Point, 1)
  res[0] = Point{
    x: a,
    y: y0,
  }

  b -= h
  for zk, yk, xk := dydx0, y0, a; xk <= b; xk += h {
    k1 := h * f(xk, yk, zk)
```

```
    l1 := h * g(xk, yk, zk)

    k2 := h * f(xk+h/2, yk+l1/2, zk+k1/2)
    l2 := h * g(xk+h/2, yk+l1/2, zk+k1/2)

    k3 := h * f(xk+h/2, yk+l2/2, zk+k2/2)
    l3 := h * g(xk+h/2, yk+l2/2, zk+k2/2)

    k4 := h * f(xk+h, yk+l3, zk+k3)
    l4 := h * g(xk+h, yk+l3, zk+k3)

    zk += (k1 + 2*k2 + 2*k3 + k4) / 6
    yk += (l1 + 2*l2 + 2*l3 + l4) / 6

    res = append(res, Point{
      x: xk + h,
      y: yk,
    })
  }

  return res
}

func shootingMethod(f, g fn, a, b, y0, y1, h, etaprev, etacurr, eps float64)
[]Point {
  zprev := etaprev
  yprev := rungeKuttaMethod(f, g, a, b, h, y0, zprev)
  zcurr := etacurr
  ycurr := rungeKuttaMethod(f, g, a, b, h, y0, zcurr)

  F := func(y []Point) float64 { return y[len(y)-1].y - y1 }

  for math.Abs(F(ycurr)) > eps {
    temp := zcurr
    zcurr = zcurr - (zcurr-zprev)/(F(ycurr)-F(yprev))*F(ycurr)
    zprev = temp

    yprev = ycurr
    ycurr = rungeKuttaMethod(f, g, a, b, h, y0, zcurr)
  }

  return ycurr
}

func finiteDifferenceMethod(p, q fn, a, b, y0, y1, h, alpha, beta, gamma, delta
float64) []Point {
  n := int((b-a)/h) + 1
  diag := matrix.MakeRealMatrix(n, n)
  bias := matrix.MakeRealMatrix(n, 1)
  points := make([]Point, n)
  xk := a
```

```
  for i := 0; i < n; i++ {
    points[i].x = xk
    // A
    if i > 0 {
      if i == n-1 {
        diag.SetEl(i, i-1, -gamma)
      } else {
        diag.SetEl(i, i-1, 1-p(xk, 0, 0)*h/2)
      }
    }
    // B
    if i == 0 {
      diag.SetEl(i, i, alpha*h-beta)
    } else if i == n-1 {
      diag.SetEl(i, i, delta*h+gamma)
    } else {
      diag.SetEl(i, i, q(xk, 0, 0)*h*h-2)
    }

    // C
    if i < n-1 {
      if i == 0 {
        diag.SetEl(i, i+1, beta)
      } else {
        diag.SetEl(i, i+1, 1+p(xk, 0, 0)*h/2)
      }
    }
    xk += h
  }
  // bias
  bias.SetEl(0, 0, y0*h)
  bias.SetEl(n-1, 0, y1*h)

  ans := run_through.Do(diag, bias)
  for i := 0; i < n; i++ {
    points[i].y = ans.GetEl(i, 0)
  }

  return points
}

func rungeRomberg(h1, h2 float64, p1, p2 []Point, p float64) []Point {
  k := h1 / h2
  div := math.Pow(k, p) - 1
  res := make([]Point, len(p1))
  for i := 0; i < len(p1); i++ {
    res[i].x = p1[i].x
    res[i].y = (p2[i].y - p1[i].y) / div
  }

  return res
```

```
}

func err(p []Point, anf fn) []Point {
  res := make([]Point, len(p))
  for i := 0; i < len(p); i++ {
    res[i].x = p[i].x
    res[i].y = math.Abs(p[i].y - anf(p[i].x, 0, 0))
  }
  return res
}

func main() {
  var (
    anf fn = func(x, y, z float64) float64 { return -math.Tan(x) }
    f   fn = func(x, y, z float64) float64 { return 2 * (1 + (math.Pow(math.Tan(x),
2))) * y }
    g   fn = func(x, y, z float64) float64 { return z }
    p   fn = func(x, y, z float64) float64 { return 0 }
    q   fn = func(x, y, z float64) float64 { return -2 * (1 +
(math.Pow(math.Tan(x), 2))) }

    a       float64   = 0
    b       float64   = math.Pi / 6
    h       []float64 = []float64{math.Pi / 30, math.Pi / 60}
    y0      float64   = 0
    y1      float64   = anf(b, 0, 0)
    alpha   float64   = 1
    delta   float64   = 1
    gamma   float64   = 0
    beta    float64   = 0
    etaprev float64   = 0.8
    etacurr float64   = 1
    eps     float64   = 0.001
  )

  {
    p1 := shootingMethod(f, g, a, b, y0, y1, h[0], etaprev, etacurr, eps)
    fmt.Println("Метод стрельбы: h: ", h[0], "\n", p1)

    p2 := shootingMethod(f, g, a, b, y0, y1, h[1], etaprev, etacurr, eps)
    fmt.Println("Метод стрельбы: h: ", h[1], "\n", p2)

    fmt.Println("Точность:\n", "\tРунге-Ромберг:\t", rungeRomberg(h[0], h[1], p1,
p2, 4), "\nОностиельно точного решения:\n\th: ", h[0], " err:\t", err(p1, anf), "\
n\th: ", h[1], " err:\t", err(p2, anf))
  }
  {
    p1 := finiteDifferenceMethod(p, q, a, b, y0, y1, h[0], alpha, beta, gamma,
delta)
    fmt.Println("Конечно-разностный метод: h: ", h[0], "\n", p1)
```

```
    p2 := finiteDifferenceMethod(p, q, a, b, y0, y1, h[1], alpha, beta, gamma,
delta)
    fmt.Println("Конечно-разностный метод: h: ", h[1], "\n", p2)

    fmt.Println("Точность:\n", "\tРунге-Ромберг:\t", rungeRomberg(h[0], h[1], p1,
p2, 4), "\nОностиельно точного решения:\n\th: ", h[0], " err:\t", err(p1, anf), "\
n\th: ", h[1], " err:\t", err(p2, anf))
  }
}
```

# Вывод программы

```
Метод стрельбы: h:  0.10471975511965978

 [{0 0} {0.10471975511965978 -0.10510458949111323} {0.20943951023931956 -
0.21255722014933065} {0.3141592653589793 -0.3249205180705746}
{0.4188790204786391 -0.44522938060627965} {0.5235987755982989 -
0.5773502691896268}]

Метод стрельбы: h:  0.05235987755982989

 [{0 0} {0.05235987755982989 -0.05938398324911917} {0.10471975511965978 -
0.1190950695899825} {0.15707963267948966 -0.17946760892989602}
{0.20943951023931956 -0.2408507890131094} {0.26179938779914946 -
0.3036169423053533} {0.31415926535897937 -0.36817100125875385}
{0.36651914291880927 -0.43496163424485546} {0.4188790204786392 -
0.50449474775313} {0.4712388980384691 -0.577350269189625 8}]

Точность:

        Рунге-Ромберг:   [{0 0} {0.10471975511965978 0.0030480404161329373}
{0.20943951023931956 0.006230810037289877} {0.3141592653589793
0.009696860609378572} {0.4188790204786391 0.013625239439544683}
{0.5235987755982989 0.01824888845895157}]

Ностиельно точного решения:

        h:  0.10471975511965978  err:    [{0 0} {0.10471975511965978
3.5422543676055795e-07} {0.20943951023931956 6.584793085240292e-07}
{0.3141592653589793 8.218376683077899e-07} {0.4188790204786391
6.952977434360186e-07} {0.5235987755982989 9.992007221626409e-16}]

        h:  0.05235987755982989  err:    [{0 0} {0.05235987755982989
0.006976203966077961} {0.10471975511965978 0.013990834324306034}
{0.15707963267948966 0.021083168605359748} {0.20943951023931956
0.028294227343087286} {0.26179938779914946 0.03566774987423055}
{0.31415926535897937 0.04325130502584745} {0.36651914291880927
0.05109759920943957} {0.4188790204786392 0.0592660624445937}
{0.4712388980384691 0.06782481969519694}]

Конечно-разностный метод: h:  0.10471975511965978
```

[{0 0} {0.10471975511965978 -0.10518457862235336} {0.20943951023931956 -0.2127015979128524} {0.3141592653589793 -0.3250944543997975} {0.4188790204786391 -0.4453701768432036} {0.5235987755982989 -0.5773502691896258}]

Конечно-разностный метод: h:  0.05235987755982989

[{0 0} {0.05235987755982989 -0.05241815523894563} {0.10471975511965978 -0.10512451458158852} {0.15707963267948966 -0.15841365111514813} {0.20943951023931956 -0.21259317706660097} {0.26179938779914946 -0.26799104097440124} {0.31415926535897937 -0.3249638301824856} {0.36651914291880927 -0.3839065440381282} {0.4188790204786392 -0.44526443665017484} {0.4712388980384691 -0.5095477270224349} {0.5235987755982989 -0.5773502691896258}]

Точность:

Рунге-Ромберг:  [{0 0} {0.10471975511965978 0.003517761558893849} {0.20943951023931956 0.007171805555417592} {0.3141592653589793 0.011112053552309958} {0.4188790204786391 0.01551846665177351} {0.5235987755982989

0.02062394854768164}]

Относиельно точного решения:

h:  0.10471975511965978  err:  [{0 0} {0.10471975511965978 8.034335667689296e-05} {0.20943951023931956 0.00014503624283027094} {0.3141592653589793 0.00017475816689122015} {0.4188790204786391 0.00014149153466741504} {0.5235987755982989 0}]

h:  0.05235987755982989  err:  [{0 0} {0.05235987755982989 1.0375955904418088e-05} {0.10471975511965978 2.027931591204457e-05} {0.15707963267948966 2.9210790611855497e-05} {0.20943951023931956 3.661539657884294e-05} {0.26179938779914946 4.184854327848875e-05} {0.31415926535897937 4.413394957919481e-05} {0.36651914291880927 4.250900271229874e-05} {0.4188790204786392 3.575134163857907e-05} {0.4712388980384691 2.2277528005987257e-05} {0.5235987755982989 0}]