Московский авиационный институт
(Национальный исследовательский университет)
Факультет прикладной математики и физики
Кафедра вычислительной математики и программирования

Лабораторная работа № 4

по курсу «Численные методы»

Тема: численные методы решения СЛАУ.

Студент: Сыроежкин К.Г.
Группа: 80-304б
Преподаватель: Гидаспов В.Ю.
Оценка:

Москва, 2021

Задание 1

1) Постановка задачи:

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 16:

$(x^2-1)y''-2xy'+2y=0,$ $y(2)=7,$ $y'(2)=5,$ $x\in[2,3], h=0.1$	$y=x^2+x+1$

2) Теория:

Итерационная формула метода Эйлера:

$$y_{k+1}=y_k+hf(x_k,y_k)$$

Совокупность формул для семейства методов Рунге-Кутты:

$$y_{k+1}=y_k+\Delta y_k$$

$$\Delta y_k=\sum_{i=1}^{p}c_iK_i^k$$

$$K_i^k=hf(x_k+a_ih,y_k+h\sum_{j=1}^{i-1}b_{ij}K_j^k)$$

$$i=2,3.....p$$

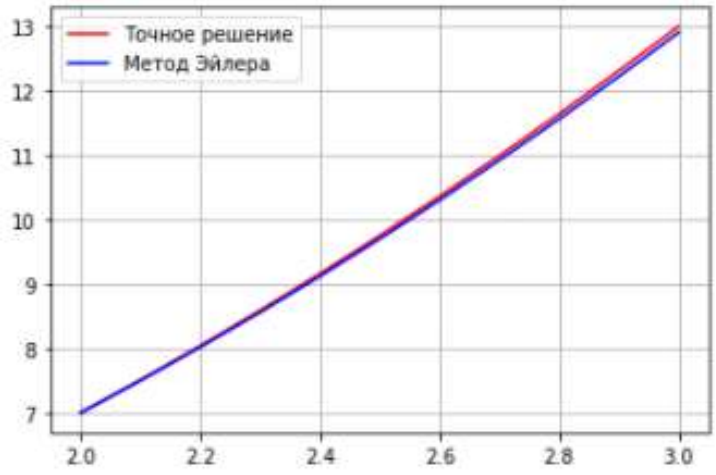
Коэффициенты подбираются так, чтобы значение у совпадало со значением разложения точного решения в ряд Тейлора в точке х. Решение дифф. уравнения удовлетворяет интегральному соотношению:

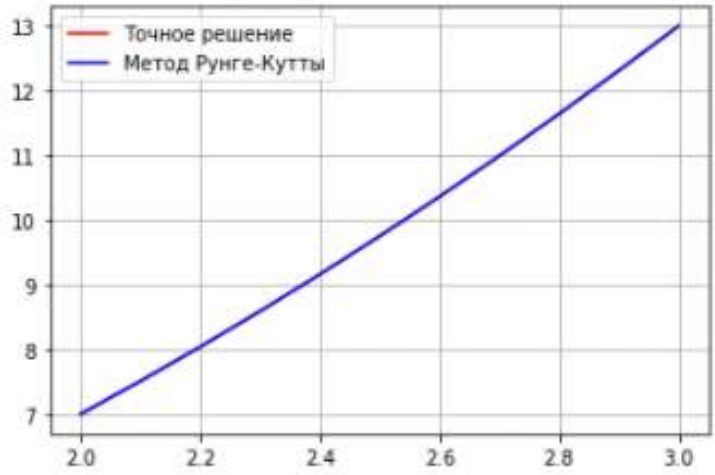
$$y_{k+1}=y_k+\int_{x_k}^{x_{k+1}}f(x,y(x))dx$$

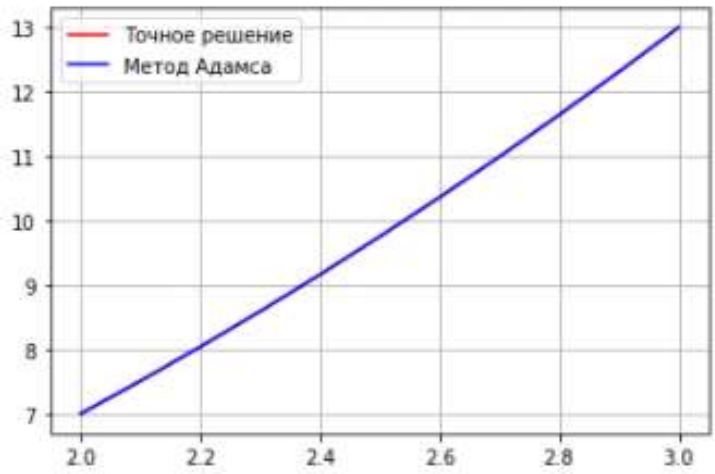
Если аппроксимировать подынтегральную ф-цию многочленом какой-либо степени, а затем вычислить интеграл, то можно получить формулу Адамса. При использовании многочлена 3 степени получим формулу Адамса 4 порядка:

$$y_{k+1} = y_k + \frac{h}{24}(55f_k - 59f_{k-1} + 37f_{k-2} - 9f_{k-3})$$

3) Полученный ответ:








```
Метод Эйлера
x = 2.000000000000000; y = 7.000000000000000, y (точный) = 7.000000000000000
погрешность: 0.000000000000000
```

```
x = 2.100000000000000; y = 7.500000000000000, y (точный) = 7.510000000000000
погрешность: 0.010000000000000
x = 2.200000000000000; y = 8.020000000000000, y (точный) = 8.040000000000001
погрешность: 0.020000000000001
x = 2.300000000000000; y = 8.560058651026392, y (точный) = 8.590000000000002
погрешность: 0.029941348973610
x = 2.400000000000000; y = 9.120228189149559, y (точный) = 9.160000000000002
погрешность: 0.039771810850443
x = 2.500000000000000; y = 9.700555492894201, y (точный) = 9.750000000000002
погрешность: 0.049444507105800
x = 2.600000000000001; y = 10.301082910507423, y (точный) = 10.360000000000003
погрешность: 0.058917089492580
x = 2.700000000000001; y = 10.921848918396593, y (точный) = 10.990000000000004
погрешность: 0.068151081603411
x = 2.800000000000001; y = 11.562888653003162, y (точный) = 11.640000000000004
погрешность: 0.077111346996842
x = 2.900000000000001; y = 12.224234346007126, y (точный) = 12.310000000000006
погрешность: 0.085765653992880
x = 3.000000000000001; y = 12.905915684482165, y (точный) = 13.000000000000007
погрешность: 0.094084315517842
Метод Рунге-Кутты
x = 2.000000000000000; y = 7.000000000000000, y (точный) = 7.000000000000000
погрешность: 0.000000000000000
x = 2.100000000000000; y = 7.510000166569061, y (точный) = 7.510000000000000
погрешность: 0.000000166569062
x = 2.200000000000000; y = 8.040000329712100, y (точный) = 8.040000000000001
погрешность: 0.000000329712099
x = 2.300000000000000; y = 8.590000491967928, y (точный) = 8.590000000000002
погрешность: 0.000000491967926
x = 2.400000000000000; y = 9.160000655199950, y (точный) = 9.160000000000002
погрешность: 0.000000655199948
x = 2.500000000000000; y = 9.750000820804896, y (точный) = 9.750000000000002
погрешность: 0.000000820804894
x = 2.600000000000001; y = 10.360000989848992, y (точный) = 10.360000000000003
погрешность: 0.000000989848989
x = 2.700000000000001; y = 10.990001163159294, y (точный) = 10.990000000000004
погрешность: 0.000001163159290
x = 2.800000000000001; y = 11.640001341386474, y (точный) = 11.640000000000004
погрешность: 0.000001341386470
x = 2.900000000000001; y = 12.310001525048920, y (точный) = 12.310000000000006
погрешность: 0.000001525048914
x = 3.000000000000001; y = 13.000001714564299, y (точный) = 13.000000000000007
погрешность: 0.000001714564291
Метод Адамса
x = 2.000000000000000; y = 7.000000000000000, y (точный) = 7.000000000000000
погрешность: 0.000000000000000
x = 2.100000000000000; y = 7.510000166569061, y (точный) = 7.510000000000000
погрешность: 0.000000166569062
x = 2.200000000000000; y = 8.040000329712100, y (точный) = 8.040000000000001
погрешность: 0.000000329712099
x = 2.300000000000000; y = 8.590000491967928, y (точный) = 8.590000000000002
```

```
погрешность: 0.000000491967926
x = 2.400000000000000; y = 9.160000565490698, y (точный) = 9.160000000000002
погрешность: 0.000000565490696
x = 2.500000000000000; y = 9.750000628987861, y (точный) = 9.750000000000002
погрешность: 0.000000628987859
x = 2.600000000000001; y = 10.360000709470556, y (точный) = 10.360000000000003
погрешность: 0.000000709470553
x = 2.700000000000001; y = 10.990000792478256, y (точный) = 10.990000000000004
погрешность: 0.000000792478252
x = 2.800000000000001; y = 11.640000878408921, y (точный) = 11.640000000000004
погрешность: 0.000000878408917
x = 2.900000000000001; y = 12.310000969558789, y (точный) = 12.310000000000006
погрешность: 0.000000969558783
x = 3.000000000000001; y = 13.000001065458894, y (точный) = 13.000000000000007
погрешность: 0.000001065458887
```

4) Код программы:

```
"""Дифференциальное уравнение"""
def function(x,y,z):
    return (2*x*z-2*y)/(x**2-1)

"""Аналитическое решение"""
def solution(x):
    return x**2+x+1

"""Метод Эйлера"""
def method_Eiler(f, y0, z0, a, b, h):
    y = [y0]
    z = [z0]
    x = []
    i = 0
    for xi in np.arange(a,b+h,h):
        x.append(xi)
        z.append(z[i]+h*f(x[i],y[i],z[i]))
        y.append(y[i]+h*z[i])
        i += 1
    z.pop()
    y.pop()
    return x,y,z

"""Метод Рунге-Кутты"""
def runge_kutt(f, y0, z0, a, b, h):
    y = [y0]
    z = [z0]
    x = []
    i = 0
    for xi in np.arange(a,b+h,h):
        k = np.zeros(4)
```

```
        l = np.zeros(4)
        x.append(xi)
        for j in range(4):
            if j == 0:
                    l[j] = h * f(x[i], y[i], z[i])
                    k[j] = h * z[i]
            elif j == 3:
                    l[j] = h * f(x[i] + h, y[i] + k[j-1], z[i] + l[j-1])
                    k[j] = h * (z[i] + l[j-1])
            else:
                    l[j] = h * f(x[i] + 0.5*h, y[i] + 0.5 * k[j-1], z[i] +
0.5*l[j-1])
                    k[j] = h * (z[i] + 0.5*l[j-1])
        z.append(z[i] + (l[0] + 2 * (l[1] + l[2]) + l[3]) / 6)
        y.append(y[i] + (k[0] + 2 * (k[1] + k[2]) + k[3]) / 6)
        i+=1
    y.pop()
    z.pop()
    return x,y,z

"""Метод Адамса"""
def adams(f, y0, z0, a, b, h):
    x, y, z = runge_kutt(f, y0, z0, a, b, h)
    y = y[:4]
    z = z[:4]
    for i in range(3, len(x) - 1):
        f1 = f(x[i], y[i], z[i])
        f2 = f(x[i-1], y[i-1], z[i-1])
        f3 = f(x[i-2], y[i-2], z[i-2])
        f4 = f(x[i-3], y[i-3], z[i-3])
        z.append(z[i] + h / 24 * (55*f1 - 59*f2 + 37*f3 - 9*f4))
        y.append(y[i] + h / 24 * (55*z[i] - 59*z[i-1] + 37*z[i-2] - 9*z[i-3]))
    return x, y, z


"""Получить ошибку метода"""
def get_error(x,y,z,method, sol):
    print(method)
    a = 2
    for i in range(len(x)):
        print("x = {0:.15f};".format(x[i],), "y = {0:.15f},".format(y[i]), "y
(точный) = {0:.15f}".format(sol(x[i])), "погрешность:
{0:.15f}".format(np.abs(y[i]-sol(x[i]))))

"""Получить точное решение для заданных X"""
def pres(x,sol):
    pres = []
    for i in range(len(x)):
        pres.append(sol(x[i]))
    return pres
```

```
def main():
    y0 = 7
    z0 = 5
    a = 2
    b = 3
    h = 0.1
    x, y, z = method_Eiler(function, y0, z0, a, b, h)
    get_error(x,y,z,"Метод Эйлера",solution)
    y_pres = pres(x,solution)
    fig, ax = plt.subplots()
    ax.grid()
    ax.plot(x,y_pres, color="red", label="Точное решение")
    ax.plot(x,y, label="Метод Эйлера", color="blue")
    ax.legend(loc='best')
    x, y, z = runge_kutt(function, y0, z0, a, b, h)
    get_error(x,y,z,"Метод Рунге-Кутты",solution)
    fig, ax = plt.subplots()
    ax.grid()
    ax.plot(x,y_pres, label="Точное решение", color="red")
    ax.plot(x,y, label="Метод Рунге-Кутты", color="blue")
    ax.legend(loc='best')
    x, y, z = adams(function, y0, z0, a, b, h)
    get_error(x,y,z,"Метод Адамса",solution)
    fig, ax = plt.subplots()
    ax.grid()
    ax.plot(x,y_pres, label="Точное решение", color="red")
    ax.plot(x,y, label="Метод Адамса", color="blue")
    ax.legend(loc='best')
if __name__ == "__main__":
    main()
```

Задание 2

1) Постановка задачи:

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 16:

y'' - tgx y' +2y=0, y(0)=2, $y(\frac{\pi}{6}) = 2.5 - 0.5 \cdot \ln 3$	$y(x) = \sin x + 2 - \sin x \cdot \ln\left(\frac{1+\sin x}{1-\sin x}\right)$

2) Теория:

пусть необходимо решить задачу Коши

$$\{y'' = f(x, y, y') \; y(a) = y_0 \; y(b) = y_1$$

Заменяем начальные условия на

$$y(a) = y_0$$
$$y'(b) = \eta$$

Задача формулируется таким образом: требуется найти такое значение η, чтобы решение $y(b, y_0, \eta)$ в правом конце отрезка совпало со значением y_1.

Это эквивалентно нахождению корня уравнения:

$$\Phi(\eta) = y(b, y_0, \eta) - y_1 = 0$$

Для решения уравнения применяется итерационная формула:

$$\eta_{j+2} = \eta_{j+1} - \frac{\eta_{j+1} - \eta_j}{\Phi(\eta_{j+1}) - \Phi(\eta_j)} \Phi(\eta_{j+1})$$

Итерации выполняются до удовлетворения заданной точности.

пусть необходимо решить краевую задачу вида:

$$y'' + p(x)y' + q(x)y = f(x)$$

$$y(a) = y_0, y(b) = y_1 \quad .$$

Разобьем отрезок [a,b] на интервалы и аппроксимируем производные:

$$y_k' = \frac{y_{k+1} - y_{k-1}}{2h} + O(h^2);$$

$$y_k'' = \frac{y_{k+1} - 2y_k + y_{k-1}}{h^2} + O(h^2);$$

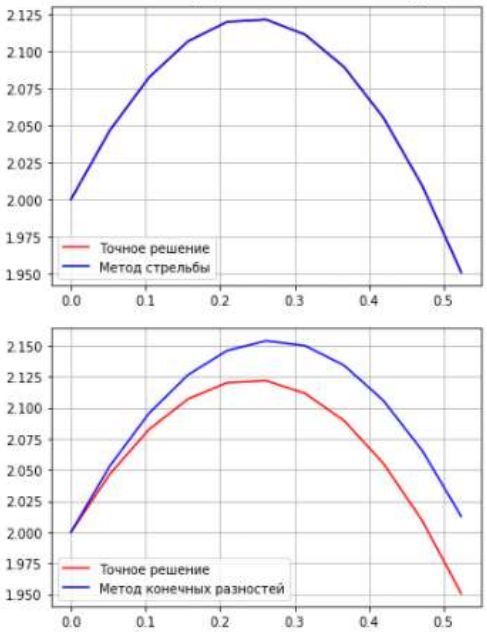
Подставляя аппроксимации и приведя подобные, получим систему линейных уравнений с трехдиагональной матрицей, которую удобно решать методом прогонки:

$$\begin{cases} (-2+h^2q(x_1))y_1+(1+\frac{p(x_1)h}{2})y_2=h^2f(x_1)-(1-\frac{p(x_1)h}{2})y_a \\ (1-\frac{p(x_k)h}{2})y_{k-1}+(-2+h^2q(x_k))y_k+(1+\frac{p(x_k)h}{2})y_{k+1}=h^2f(x_k) \qquad ,k=2,\dots,N-2 \\ (1-\frac{p(x_{N-1})h}{2})y_{N-1}+(-2+h^2q(x_{N-1}))y_{N-1}=h^2f(x_{N-1})-(1+\frac{p(x_{N-1})h}{2})y_b \end{cases}$$

Решение системы будет являтся решением исходной задачи в виде табличной функции.

3) Полученный ответ:

```
Метод стрельбы
x = 0.000000000000000; y = 2.000000000000000, y (точный) = 2.000000000000000 погрешность: 0.000000000000000
x = 0.052359877559830; y = 2.046852821828083, y (точный) = 2.046852841770247 погрешность: 0.000000019942164
x = 0.104719755119660; y = 2.082595914520098, y (точный) = 2.082595950246334 погрешность: 0.000000035726237
x = 0.157079632679490; y = 2.107085723392036, y (точный) = 2.107085770657747 погрешность: 0.000000047265711
x = 0.209439510239320; y = 2.120178020545785, y (точный) = 2.120178075017404 погрешность: 0.000000054471619
x = 0.261799387799149; y = 2.121726552516562, y (точный) = 2.121726609765882 погрешность: 0.000000057249320
x = 0.314159265358979; y = 2.111580876535678, y (точный) = 2.111580932028880 погрешность: 0.000000055493202
x = 0.366519142918809; y = 2.089583261591196, y (точный) = 2.089583310669430 погрешность: 0.000000049078234
x = 0.418879020478639; y = 2.055564477248579, y (точный) = 2.055564515095233 погрешность: 0.000000037846654
x = 0.471238898038469; y = 2.009338223019732, y (точный) = 2.009338244606631 погрешность: 0.000000021586899
x = 0.523598775598299; y = 1.950693855665946, y (точный) = 1.950693855665945 погрешность: 0.000000000000001
0.05235987755982988
Метод конечных разностей
x = 0.000000000000000; y = 2.000000000000000, y (точный) = 2.000000000000000 погрешность: 0.000000000000000
x = 0.052359877559830; y = 2.053345783141122, y (точный) = 2.046852841770247 погрешность: 0.006492941370875
x = 0.104719755119660; y = 2.095563955030877, y (точный) = 2.082595950246334 погрешность: 0.012968004784542
x = 0.157079632679490; y = 2.126493185826004, y (точный) = 2.107085770657747 погрешность: 0.019407415168257
x = 0.209439510239320; y = 2.145971628169024, y (точный) = 2.120178075017404 погрешность: 0.025793553151619
x = 0.261799387799149; y = 2.153835617300845, y (точный) = 2.121726609765882 погрешность: 0.032109007534963
x = 0.314159265358979; y = 2.149917561463338, y (точный) = 2.111580932028880 погрешность: 0.038336629434458
x = 0.366519142918809; y = 2.134042899162646, y (точный) = 2.089583310669430 погрешность: 0.044459588493216
x = 0.418879020478639; y = 2.106025946910452, y (точный) = 2.055564515095233 погрешность: 0.050461431815219
x = 0.471238898038469; y = 2.065664391267229, y (точный) = 2.009338244606631 погрешность: 0.056326146660597
x = 0.523598775598299; y = 2.012732084169326, y (точный) = 1.950693855665945 погрешность: 0.062038228503381
```



4) Код программы:

```
def function(x,y,z):
    return np.tan(x)*z-2*y

def solution(x):
    return np.sin(x)+2-np.sin(x)*np.log((1+np.sin(x))/((1-np.sin(x))))

def get_error(x,y,method, sol):
    print(method)
    a = 2
    for i in range(len(x)):
        print("x = {0:.15f};".format(x[i],), "y = {0:.15f},".format(y[i]), "y
(точный) = {0:.15f}".format(sol(x[i])), "погрешность:
{0:.15f}".format(np.abs(y[i]-sol(x[i]))))


def f(x, y, z):
    if x**2 - 1 == 0:
        return 0
    return (y - (x-3)*z) / (x**2 - 1)


def shooting(f,a,b,h,y0,y1,n1,n2,eps):
    x = np.arange(a,b+h,h)
    xi,yi,zi = runge_kutt(f,y0,n1,a,b,h)
    xii,yii,zii = runge_kutt(f,y0,n2,a,b,h)
    if abs(yi[-1] - y1) < eps:
        return x, yi
    if abs(yii[-1] - y1) < eps:
        return x, yii
    k = 0
    while k < 100 and abs(yii[-1] - y1) > eps:
        phi1 = yi[-1] - y1
        phi2 = yii[-1] - y1
        tmp = n2 - (n2 - n1) / (phi2 - phi1) * phi2
        n1 = n2
        n2 = tmp
        yi = yii
        xii,yii,zii = runge_kutt(f,y0,n2,a,b,h)
        k+=1
    return x,yii


def p(x):
    return -np.tan(x)


def q(x):
    return 2

def f(x):
    return 0
```

```
"""Метод конечных разностей"""
def finit_difference(f,q,p,a,b,h,y0,y1,eps):
    x = np.arange(a,b+h,h)
    N = len(x)-1
    mat = np.zeros((N,N))
    answ = np.zeros((N,1))
    mat[0,0] = -2 + h**2 * q(x[1])
    mat[0,1] = 1 + p(x[1]) * h / 2
    print(x[1])
    answ[0,0] = h**2 * f(x[1]) - (1 - p(x[1]) * h / 2) * y0
    for i in range(2,N):
        mat[i-1,i-2] = (1 - p(x[i]) * h / 2)
        mat[i-1, i-1] = (-2 + h**2 * q(x[i]))
        mat[i-1,i] = 1 + p(x[i]) * h / 2
        answ[i-1,0] = h**2 * f(x[i])
    mat[N-1,N-2] = 1 - p(x[N-1]) * h / 2
    mat[N-1,N-1] = -2 + h**2 * q(x[N-1])
    answ[N-1,0] = -(1 + p(x[N]) * h / 2) * y1
    y_tmp = run(mat,answ)
    y = np.zeros((N+1,1))
    y[1:,0] = y_tmp
    y[0,0] = y0
    return x, y.reshape(1,-1)[0]


"""Получить точное решение для заданных X"""
def pres(x,sol):
    pres = []
    for i in range(len(x)):
        pres.append(sol(x[i]))
    return pres



def main():
    y0 = 2
    y1 = 2.5-0.5*np.log(3)
    a = 0
    b = math.pi/6
    h = math.pi/60
    n1 = 4
    n2 = 3
    eps = 0.1
    x,y = shooting(function,a,b,h,y0,y1,n1,n2,eps)
    y_pres = pres(x,solution)
    get_error(x,y,"Метод стрельбы",solution)
    fig, ax = plt.subplots()
    ax.grid()
    ax.plot(x,y_pres, color="red", label="Точное решение")
    ax.plot(x,y, label="Метод стрельбы", color="blue")
```

```
    ax.legend(loc='best')
    x,y = finit_difference(f,q,p,a,b,h,y0,y1,eps)
    y_pres = pres(x,solution)
    get_error(x,y,"Метод конечных разностей",solution)
    fig, ax = plt.subplots()
    ax.grid()
    ax.plot(x,y_pres, color="red", label="Точное решение")
    ax.plot(x,y, label="Метод конечных разностей", color="blue")
    ax.legend(loc='best')
if __name__ == "__main__":
    main()
```