

```

1  #! python3.7
2  # -*- coding: utf-8 -*-
3  from numpy import zeros, linspace, log, sqrt, sum, amax
4  from matplotlib.pyplot import style, figure, axes
5
6  # Функция f подготавливает массив, содержащий элементы вектор-функции,
7  # определяющей правую часть решаемой системы ОДУ
8  def f(u,m_sun,G) :
9      f = zeros(4)
10     f[0] = u[2]
11     f[1] = u[3]
12     f[2] = - G*m_sun*u[0]/(u[0]**2 + u[1]**2)**(3/2)
13     f[3] = - G*m_sun*u[1]/(u[0]**2 + u[1]**2)**(3/2)
14     return f
15
16 # Функция реализует решение системы ОДУ
17 # на сетке с M_0*r**s интервалами по схеме ERK2
18 def ODESolving(t_0,T,x_0,y_0,v_x_0,v_y_0,M_0,s,r,m_sun,G) :
19     # Входные параметры:
20     # t_0, T - начальный и конечный моменты счёта
21     # x_0, y_0, v_x_0, v_y_0 - начальные условия
22     # M_0 - число интервалов базовой сетки по времени
23     # s - номер сетки, на которой вычисляется решение
24     # (если s = 0, то решение ищется на базовой сетке)
25     # r - коэффициент сгущения сетки
26     # m_sun, G - параметры задачи
27
28     # Выходной параметр:
29     # u_basic - массив, содержащий сеточные значения
30     # решения системы ОДУ только в узлах,
31     # совпадающих с узлами базовой сетки
32
33     # Формирование сгущённой в r^s раз сетки с номером s:
34
35     # Вычисление числа интервалов на сетке с номером s
36     M = M_0*r**s
37     # Определение шага сгущённой сетки
38     tau = (T - t_0)/M
39     # Определение сгущённой сетки
40     t = linspace(t_0,T,M + 1)
41
42     # Выделение памяти под массив сеточных значений решения системы ОДУ,
43     # в котором будут храниться сеточные значения из узлов,
44     # совпадающих с узлами базовой сетки
45     u_basic = zeros((M_0 + 1,4))
46
47     # Выделение памяти под массив сеточных значений
48     # решения на сгущённой сетке
49     # В строке с номером m этого массива хранятся сеточные значения решения,
50     # соответствующие моменту времени t_m
51     u = zeros((M + 1,4))
52
53     # Задание начального условия
54     u[0] = [x_0, y_0, v_x_0, v_y_0]
55
56     # Реализация схемы ERK2
57     for m in range(M) :
58         w_1 = f(u[m],m_sun,G)
59         w_2 = f(u[m] + tau*2/3*w_1,m_sun,G)
60         u[m + 1] = u[m] + tau*(1/4*w_1 + 3/4*w_2)
61
62     # Из массива u выбираются сеточные значения из узлов,
63     # совпадающих с узлами базовой сетки
64     for m in range(M_0 + 1) :
65         u_basic[m] = u[m*r**s]
66
67     return u_basic
68
69 # Определение входных данных задачи

```

```

70 t_0 = 0.; T = 365.25*24*60*60
71 x_0 = 147098291*10**3; y_0 = 0.
72 v_x_0 = 0.; v_y_0 = 30.4*10**3
73 G = 6.674301515151515*10**(-11)
74 m_sun = 1.98847*10**30
75
76 # Определение числа интервалов БАЗОВОЙ сетки,
77 # на которой будет искаться приближённое решение
78 M = 36
79
80 # Число сеток, на которых ищется приближённое решение
81 S = 7
82 # Коэффициент ступенчатости сеток
83 r = 2
84 # Теоретические параметры схемы
85 p = 2; q = 1
86
87 # Выделение памяти под массивы сеточных значений
88 # решений ОДУ на разных сетках с номерами s = 0, ..., S-1,
89 # в которых хранятся сеточные значения решения из узлов,
90 # совпадающих с узлами базовой сетки
91 U = zeros((S,S,M + 1,4))
92
93 # "Большой цикл", который пересчитывает решение S раз
94 # на последовательности ступенчатых сеток
95 # Массив сеточных значений решения содержит только
96 # сеточные значения из узлов, совпадающих с узлами базовой сетки
97 for s in range(S) :
98     U[s,0, :, :] = ODESolving(t_0,T,x_0,y_0,v_x_0,v_y_0,M,s,r,m_sun,G)
99
100 # Выделение памяти под массивы ошибок R,
101 # относительных ошибок R_rel и эффективных порядков точности p_eff
102 R = zeros((S,S,M + 1,4))
103 R_rel = zeros((S,S))
104 p_eff = zeros((S,S))
105
106 for s in range(1,S) :
107     for l in range(s) :
108         R[s,l, :, :] = (U[s,l, :, :] - U[s-1,l, :, :]) / (r**(p + l*q) - 1)
109         U[s,l+1, :, :] = U[s,l, :, :] + R[s,l, :, :]
110         R_rel[s,l] = amax(abs(R[s,l, :, :])) / amax(abs(U[s,l+1, :, :])) * 100
111
112 for s in range(2,S) :
113     for l in range(s-1) :
114         p_eff[s,l] = log(amax(abs(R[s-1,l, :, :])) / amax(abs(R[s,l, :, :]))) / log(r)
115
116 # Функция выводит форматированную таблицу
117 def PrintTriangular(A,i) :
118     print(' ',end=' ')
119     for l in range(len(A)) :
120         print(' p={0:<4d}'.format(p + l*q),end=' ')
121     print()
122     for m in range(len(A)) :
123         print('s={0:<2d}'.format(m),end=' ')
124         for l in range(m + 1 - i) :
125             print('{0:5.2f}'.format(A[m,l]),end=' ')
126         print()
127     print()
128
129 print('Таблица оценок относительных ошибок (в процентах):')
130 PrintTriangular(R_rel,1)
131 print('Таблица эффективных порядков точности:')
132 PrintTriangular(p_eff,2)
133
134 # Отрисовка решения, полученного на сетке с номером S-1
135 # (отмечаются только узлы, совпадающие с узлами базовой сетки)
136 style.use('dark_background')
137
138 fig1 = figure()

```

```

139 ax1 = axes(xlim=(-2*10**11,2*10**11), ylim=(-2*10**11,2*10**11))
140 ax1.set_aspect('equal'); ax1.set_xlabel('x'); ax1.set_ylabel('y');
141 ax1.plot(0,0,'yo',markersize=15)
142 ax1.plot(U[S-1,1,:,0],U[S-1,1,:,1],'-w',markersize=5)
143 ax1.plot(U[S-1,1,M,0],U[S-1,1,M,1], color='w', marker='o', markersize=7)
144 ax1.set_title('Траектория движения Земли')
145
146 # Отрисовка зависимости ошибки от числа интервалов сетки
147 fig2 = figure()
148 ax2 = axes()
149 ax2.plot([r**s*M for s in range(1,S)], [sqrt(sum(R[s,0,:,:]**2)) for s in
150 range(1,S)], '-wo')
151 ax2.set_xscale('log'); ax2.set_yscale('log')
152
153 # Листинг программы, реализующей приближённое решение системы ОДУ
154 # (на примере моделирования движения Земли вокруг Солнца)
155 # с помощью рекуррентного сгущения сеток и многократного повышения
156 # точности по Ричардсону (с вычислением эффективных порядков точности)

```