Теоретический материал к семинару №2

Схема Рунге-Кутты для решения задачи Коши

$$\begin{cases} \dfrac{d\mathbf{u}}{dt} = \mathbf{f}\left(\mathbf{u}, t\right) \\ \mathbf{u}(t_0) = \mathbf{u}_0 \end{cases} \tag{1}$$

имеет следующий вид:

$$\begin{aligned} &\mathbf{u}_{n+1} = \mathbf{u}_n + \tau_n \sum_{k=1}^{s} b_k \boldsymbol{\omega}_k, \tau_n = t_{n+1} - t_n; \\ &\boldsymbol{\omega}_k = \mathbf{f}\left(\mathbf{u}_n + \tau_n \sum_{l=1}^{L} \alpha_{kl} \boldsymbol{\omega}_l, t_n + \tau_n a_k\right), 1 \le k \le s. \end{aligned} \tag{2}$$

Здесь τ_n - шаги по времени, s - число стадий, коэффициенты α_{kl} образуют матрицу Бутчера $\mathbf{A}$, а a_k и b_k - элементы векторов $\mathbf{a}$ и $\mathbf{b}$, вместе с матрицой Бутчера полностью задающих схему Рунге-Кутта.

Для реализации на компьютере с использованием Python удобнее записать (2) в векторной форме

$$\begin{aligned} &\mathbf{u}_{n+1} = \mathbf{u}_n + \tau_n \boldsymbol{\omega} \mathbf{b}^T, \tau_n = t_{n+1} - t_n; \\ &\boldsymbol{\omega}_k = \mathbf{f}\left(\mathbf{u}_n + \tau_n \boldsymbol{\omega} \boldsymbol{\alpha}_k^T, t_n + \tau_n a_k\right), 1 \le k \le s, \end{aligned} \tag{3}$$

где $\boldsymbol{\omega}_k$ - k-тый столбец матрицы промежуточных результатов $\boldsymbol{\omega}$, первоначально полагаемой нулевой, $\mathbf{b}$ - вектор-строка коэффициентов b и $\boldsymbol{\alpha}_k$ - k-тая строка матрицы Бутчера. Верхний индекс T означает транспонирование.

Задачи к семинару №2

1. Записать расчетные формулы для схемы Кутта, если

$$\mathbf{A} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \qquad \mathbf{a} = \begin{pmatrix} 0 \\ 1/2 \\ 1/2 \\ 1 \end{pmatrix}, \qquad \mathbf{b} = \begin{pmatrix} 1/6 \\ 1/3 \\ 1/3 \\ 1/6 \end{pmatrix}^T. \tag{4}$$

2. Перейти к длине дуги в задаче

$$\begin{cases} \dfrac{du}{dt} = u^2 + t^2 \\ u(t_0) = u_0 \end{cases} . \tag{5}$$

3. Реализуйте схему Кутта на компьютере, используя Python (или MATLAB) и соответствующие библиотеки.
   Тестовые функции (правые части):

   a)

   Python:

```python
def myfun(t, u):
    return u + t**2 + 1
```

   MATLAB:

```matlab
function y = f(t, u)
    y = u + t^2 + 1;
end
```

   Начальное условие: $u_0 = 0.5$

   b)

   Python:

```python
import numpy as np

def f(t, u):
    om = np.array([np.sin(t), np.cos(t), np.sin(t + np.pi/4)])
    Omega = np.array([[0, -om[2], om[1]],
                      [om[2], 0, -om[0]],
                      [-om[1], om[0], 0]])
    return np.dot(Omega, u)
```

   MATLAB:

```matlab
function y = f(t, u)
    om    = [ sin(t) cos(t) sin(t+pi/4) ];
    Omega = [ 0      -om(3)   om(2);
              om(3)   0      -om(1);
             -om(2)   om(1)   0           ];
    y = Omega * u;
end
```

Начальное условие: $u_0 = [1; -0.5; 0.6]$;
Временной отрезок для обеих функций - от 0 до 1.
Провести 7 расчетов на сгущающихся вдвое сетках, начиная с минимально возможной сетки из 1 интервала.
Для первой функции построить график эффективного порядка метода от числа интервалов сетки (по последнему узлу, т.е. в последнем узле сетки при $t = 1$), для второй - построить график решения (3 кривые на одном графике).

4. Реализовать явную схему Рунге-Кутты в общем виде. Для отладки использовать 7-стадийную схему Хаммуда 6 порядка:

   Python:

```
import numpy as np

butcher = np.array([
    [0, 0, 0, 0, 0, 0, 0],
    [4/7, 0, 0, 0, 0, 0, 0],
    [115/112, -5/16, 0, 0, 0, 0, 0],
    [589/630, 5/18, -16/45, 0, 0, 0, 0],
    [229/1200-29/6000*5**0.5, 119/240 - 187/1200*5**0.5, -14/75+34/375*5**0.5, -3/100*5**0.5,
    0, 0, 0],
    [71/2400 - 587/12000*5**0.5, 187/480 - 391/2400*5**0.5, -38/75 + 26/375*5**0.5,
    27/80 - 3/400*5**0.5, (1+5**0.5)/4, 0, 0],
    [-49/480+43/160*5**0.5, -425/96+51/32*5**0.5, 52/15-4/5*5**0.5,
    -27/16+3/16*5**0.5, 5/4-3/4*5**0.5,5/2-0.5*5**0.5, 0]
])

a = np.array([0, 4/7, 5/7, 6/7, (5-5**0.5)/10, (5+5**0.5)/10, 1])
b = np.array([1/12, 0, 0, 0, 5/12, 5/12, 1/12])
```

MATLAB:

```
butcher = [ 0                      0                      ...
            0                      0                      ...
            0                      0                      0;
            4/7                    0                      ...
            0                      0                      ...
            0                      0                      0;
            115/112                -5/16                  ...
            0                      0                      ...
            0                      0                      0;
            589/630                5/18                   ...
           -16/45                  0                      ...
            0                      0                      0;
            229/1200-29/6000*5^.5  119/240-187/1200*5^.5 ...
           -14/75+34/375*5^.5      -3/100*5^.5            ...
            0                      0                      0;
            71/2400-587/12000*5^.5 187/480-391/2400*5^.5 ...
           -38/75+26/375*5^.5      27/80-3/400*5^.5       ...
            (1+5^.5)/4             0                      0;
           -49/480+43/160*5^.5     -425/96+51/32*5^.5     ...
            52/15-4/5*5^.5         -27/16+3/16*5^.5       ...
            5/4-3/4*5^.5           5/2-0.5*5^.5           0 ];
a       = [ 0    4/7 5/7 6/7 (5-5^.5)/10 (5+5^.5)/10 1     ];
b       = [ 1/12 0   0   0   5/12        5/12        1/12  ];
```

Провести 7 расчетов на сгущающихся вдвое сетках, начиная с минимально возможной сетки из 1 интервала.
Протестировать на тех же тестовых функциях, построить такие же графики.