# Оглавление

# Семинар 1

## 1.1 Теоретический материал к семинару №1

### Оценка погрешности по Ричардсону

Пусть некоторая величина $U$ была вычислена на сетке с шагом $h$, тогда

$$U = U_h + R_h, \tag{1.1}$$

где $U_h$ - вычисленное значение, а $R_h$ - его погрешность. Теперь изменим шаг сетки в $r$ раз. Получим аналогично

$$U = U_{rh} + R_{rh}. \tag{1.2}$$

Если метод сходится к точному решению с порядком $p$, то

$$R_h = ch^p + o(h^p), \tag{1.3}$$

где $c$ - некоторая константа, независящая от $h$, причем $c \neq 0$. Записав аналогичное выражение для погрешности расчета с шагом $rh$ и подставив результат в (1.2), а также подставив (1.3) в (1.1) получим

$$\begin{aligned} U &= U_h + ch^p + o(h^p), \\ U &= U_{rh} + cr^p h^p + o(h^p). \end{aligned} \tag{1.4}$$

Пренебрегая членами более высокого порядка, чем $h^p$, вычтем из первого уравнения второе, тогда

$$0 = U_h - U_{rh} + ch^p(1 - r^p). \tag{1.5}$$

Выразим из этого равенства $ch^p$

$$ch^p = \frac{U_{rh} - U_h}{1 - r^p}. \tag{1.6}$$

Если отбросить члены более высокого, чем $p$ порядка, то $R_{rh} = cr^p h^p$. Отсюда, учитывая (1.6), получим

$$R_{rh} = \frac{U_{rh} - U_h}{r^{-p} - 1}. \tag{1.7}$$

Эта формула дает асимптотически точную оценку погрешности на более подробной сетке. Добавив ее к $U_{rh}$, мы можем получить ответ с более высоким порядком точности

$$U^*_{rh} = U_{rh} + R_{rh} = U_{rh} + \frac{U_{rh} - U_h}{r^{-p} - 1} = \frac{r^{-p}U_{rh} - U_h}{r^{-p} - 1}. \tag{1.8}$$

## Оценка погрешности по Эйткену

Недостатком рассмотренного метода вычисления погрешности является необходимость знания порядка метода $p$. Если сделать два последовательных сгущения в одинаковое число раз, то можно оценить порядок. В самом деле, рассмотрим отношение

$$\frac{U_{r^2h} - U_{rh}}{U_{rh} - U_h} = \frac{(1 - r^p)\, cr^p h^p}{(1 - r^p)\, ch^p} = r^p. \tag{1.9}$$

Здесь было дважды использовано выражение (1.6). Подставив $r^p$ в выражение

$$R_{r^2h} = \frac{U_{r^2h} - U_{rh}}{r^{-p} - 1}. \tag{1.10}$$

получим требуемую оценку погрешности

$$R_{r^2h} = \frac{U_{r^2h} - U_{rh}}{\dfrac{U_{rh} - U_h}{U_{r^2h} - U_{rh}} - 1}. \tag{1.11}$$

Аналогично (1.8) можно вычислить уточненное решение

$$U^*_{r^2h} = U_{r^2h} + R_{r^2h} = U_{r^2h} + \frac{U_{r^2h} - U_{rh}}{\dfrac{U_{rh} - U_h}{U_{r^2h} - U_{rh}} - 1}. \tag{1.12}$$

## Рекуррентное сгущение

Формулы (1.8) и (1.12) позволяют получать уточненные решения. Если имеются $N$ последовательных расчетов на сгущающихся сетках, то группируя 1 и 2, 2 и 3, 3 и 4 и т.д. по формуле (1.8) получим $N - 1$ уточненных значений. Для сгущения по Эйткену вместо пар расчетов следует использовать тройки: 1,2 и 3, затем 2,3 и 4 и т.п. При этом получится уже $N - 2$ уточненных значений.

К полученному набору уточненных значений можно вновь применить описанную выше процедуру, но для уточнения по Ричардсону потребуется знание порядка точности уточненных значений, полученных на предыдущем шаге. Для уточнения по Эйткену это не нужно.

Ясно, что этот процесс можно продолжать, пока не останутся два уточненных значения в методе Ричардсона и три – в методе Эйткена.

## Эффективный порядок метода

Из формулы (1.9) можно выразить эффективный порядок метода $p$

$$p = \log_r \frac{U_{r^2h} - U_{rh}}{U_{rh} - U_h}. \tag{1.13}$$

Для вычисления порядка требуется 3 сетки. Если имеются $N$ расчетов, то по 1,2 и 3 расчету можно получить первую оценку порядка, по 2,3 и 4 – вторую и т.д. – всего $N - 2$ оценок. Потом можно построить график эффективного порядка, отложив по оси абсцисс номер тройки, а по оси ординат - соответствующее значение $p$. По такому графику можно наблюдать, к какому значению стремится $p$.

Существует еще один способ определения эффективного порядка – построение графика $\lg|U_{rh} - U_h|$ от $-\lg h$ (или от $\lg N$, где $N$ - число интервалов сетки). Действительно, согласно (1.6)

$$\lg|U_{rh} - U_h| = \lg\big(\,|c|\,(1 - r^p)\,\big) + (-p)\,(-\lg h)\,. \tag{1.14}$$

Отсюда видно, что график должен быть близок к прямой, тангенс угла наклона которой даст нам $-p$. Угол наклона можно определить, например, с помощью метода наименьших квадратов, используя библиотеки Python, такие как NumPy и SciPy (например, функция *numpy.polyfit*).

***Замечание.*** Выше при выводе и конечной записи всех формул использовался шаг сетки $h$ и множитель $r$, означающий отношение шага сгущенной сетки к шагу исходной. Часто более удобно использовать число интервалов сетки $N$, а в качестве множителя $r$ - отношения числа интервалов новой сетки к числу интервалов старой. Очевидно, что «новый» $r$ будет обратной величиной к «старому». Поэтому для перехода от шагов сетки к числам интервалов необходимо провести следующие замены:

$$r \to r^{-1},\ U_{rh} \to U_{rN},\ U_{r^2h} \to U_{r^2N},\ R_{rh} \to R_{rN},\ R_{r^2h} \to R_{r^2N}. \tag{1.15}$$

Например, в этих обозначениях основная формула (1.7) будет выглядеть как

$$R_{rN} = \frac{U_{rN} - U_N}{r^p - 1}. \tag{1.16}$$

# Задачи к семинару №1

1. Вычислить $f = 21b^2 - 2a^2 + 55b^4 - 10a^2b^2 + \dfrac{a}{2b}$ при $a = 77617$ и $b = 33096$.

2. Вычислить собственные значения матрицы $\mathbf{A}$ аналитически и с использованием библиотек Python, таких как NumPy и SciPy. Представить результат на комплексной плоскости.

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & \cdots & \cdots & 0 & 10^{-2m} \\ 1 & 1 & \ddots & & & 0 \\ 0 & 1 & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & \cdots & 1 & 1 \end{pmatrix}_{2m \times 2m}$$

   Расчет выполнить при $m = 20, 23, 30$.

3. Получить формулу Симпсона из формулы трапеций с помощью однократного сгущения сетки в 2 раза и формулы Ричардсона (1.7).

4. Вычислить интеграл

$$\int_0^1 \frac{4}{1+x^2} dx$$

   численно по формуле трапеций при числе интервалов сетки $N = 16, 32, 64, 128, 256$. Затем с помощью формулы (1.13) или формулы (1.14) построить график эффективного порядка. Провести уточнение решения по формуле Ричардсона, получив из 5 расчетных значений 4 уточненных. По уточненным значениям вновь построить график эффективного порядка.

5. Вычислить интеграл

$$\int_0^1 \frac{1}{2\sqrt{x}} dx$$

   по формуле средних прямоугольников при тех же $N$. Определить эффективный порядок, провести уточнение по Эйткену (получив 3 уточненных значения), вновь определить эффективный порядок. После этого провести еще одно уточнение решения по Эйткену и сравнить полученный результат с точным значением интеграла.

# Семинар 2

## 2.1 Теоретический материал к семинару №2

Схема Рунге-Кутты для решения задачи Коши

$$\begin{cases} \dfrac{d\mathbf{u}}{dt} = \mathbf{f}\left(\mathbf{u}, t\right) \\ \mathbf{u}(t_0) = \mathbf{u}_0 \end{cases} \tag{2.1}$$

имеет следующий вид:

$$\begin{aligned} &\mathbf{u}_{n+1} = \mathbf{u}_n + \tau_n \sum_{k=1}^{s} b_k \boldsymbol{\omega}_k, \tau_n = t_{n+1} - t_n; \\ &\boldsymbol{\omega}_k = \mathbf{f}\left(\mathbf{u}_n + \tau_n \sum_{l=1}^{L} \alpha_{kl} \boldsymbol{\omega}_l, t_n + \tau_n a_k\right), 1 \le k \le s. \end{aligned} \tag{2.2}$$

Здесь $\tau_n$ - шаги по времени, $s$ - число стадий, коэффициенты $\alpha_{kl}$ 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 (2.2) в векторной форме

$$\begin{aligned} &\mathbf{u}_{n+1} = \mathbf{u}_n + \tau_n \boldsymbol{\omega} \mathbf{b}^T, \tau_n = t_{n+1} - t_n; \\ &\boldsymbol{\omega}_k = \mathbf{f}\left(\mathbf{u}_n + \tau_n \boldsymbol{\omega} \boldsymbol{\alpha}_k^T, t_n + \tau_n a_k\right), 1 \le k \le s, \end{aligned} \tag{2.3}$$

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

$$\mathbf{A} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \qquad \mathbf{a} = \begin{pmatrix} 0 \\ 1/2 \\ 1/2 \\ 1 \end{pmatrix}, \qquad \mathbf{b} = \begin{pmatrix} 1/6 \\ 1/3 \\ 1/3 \\ 1/6 \end{pmatrix}^T. \tag{2.4}$$

2. Перейти к длине дуги в задаче

$$\begin{cases} \dfrac{du}{dt} = u^2 + t^2 \\ u(t_0) = u_0 \end{cases} . \tag{2.5}$$

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:

```
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:

```
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.

# Семинар 3

## 3.1 Теоретический материал к семинару №3

Семейство одностадийных схем Розенброка для задачи Коши

$$\begin{cases} \dfrac{d\mathbf{u}}{dt} = \mathbf{f}\left(\mathbf{u}, t\right) \\ \mathbf{u}(t_0) = \mathbf{u}_0 \end{cases} \tag{3.1}$$

имеет следующий вид:

$$\begin{cases} \left(\mathbf{E} - \alpha\tau\mathbf{f}_\mathbf{u}\right)\boldsymbol{\omega} = \mathbf{f}\left(\mathbf{u}, t + \dfrac{\tau}{2}\right) \\ \hat{\mathbf{u}} = \mathbf{u} + \tau\,\mathrm{Re}\,\boldsymbol{\omega} \end{cases} \tag{3.2}$$

Здесь $\alpha$ - параметр схемы, $\tau$ - шаг схемы по времени, $\mathbf{f}_\mathbf{u}$ - матрица производных правой части по вектору $\mathbf{u}$, $\mathbf{u}$ и $\hat{\mathbf{u}}$ - численное решение в текущий и следующий моменты времени соответственно. Практический интерес представляют схемы из семейства одностадийных схем Розенброка при

$$\alpha = \frac{1}{2}, \qquad \alpha = 1, \qquad \alpha = \frac{1+i}{2}.$$

Векторная переменная $\boldsymbol{\omega}$ получается из решения системы линейных уравнений с правой частью $\mathbf{f}\left(\mathbf{u}, t + \tau/2\right)$ и матрицей системы $\mathbf{E} - \alpha\tau\mathbf{f}_\mathbf{u}$. Эту матрицу ***не следует*** обращать численно, так как если вычислять $\boldsymbol{\omega}$ как

$$\left(\mathbf{E} - \alpha\tau\mathbf{f}_\mathbf{u}\right)^{-1}\mathbf{f}\left(\mathbf{u}, t + \frac{\tau}{2}\right),$$

то это потребует большего объема вычислений, чем решение линейной системы. Особенно заметна разница в производительности, если матрица системы имеет ленточную форму, что часто бывает при применении схем Розенброка к решению систем дифференциальных уравнений, возникающий при численном решении уравнений в частных производных.

Матрицу производных $\mathbf{f}_\mathbf{u}$ при реализации схем Розенброка в общем виде лучше вычислять численно с помощью центральной разности с шагом $\Delta u = 10^{-5}$. Ее надо строить таким образом, чтобы в 1 строке $\mathbf{f}_\mathbf{u}$ были производные 1 компоненты вектор-столбца $\mathbf{f}$, по всем компонентам $\mathbf{u}$, во 2 строке - производные 2 компоненты $\mathbf{f}$ по всем компонентам $\mathbf{u}$ и т.д.

# Задачи к семинару №3

1. Решить задачу Коши с правой частью, задаваемой функцией

   Python:

```python
import numpy as np

def f(t, u):
    y = np.array([-50*(u[0]-np.cos(t))+10*u[1],
                  1.2*u[0]-u[1]*u[0]])
    return y
```

   MATLAB:

```
function y = f(t, u)
     y = [-50*(u(1)-cos(t))+10*u(2);
           1.2*u(1)-u(2)*u(1)       ];
end
```

   на временном отрезке $[0; 0.75]$ при начальном значении $u0 = [1; 1]$ с помощью явной схемы Рунге-Кутта второго порядка типа предиктор-корректор, задаваемой матрицей Бутчера $\mathbf{A}$ и векторами $\mathbf{a}$ и $\mathbf{b}$ (см. предыдущий семинар)

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

   Расчет провести с шагом по времени $\tau = 1/8$ и $\tau = 1/128$.

2. Реализовать семейство одностадийных схем Розенброка в общем виде. Для тестирования использовать задачу Коши из предыдущей задачи. Положить на один график результаты расчета этой задачи при $\alpha = \frac{1}{2}, \alpha = 1, \alpha = \frac{1+i}{2}$. Взять шаг по времени $\tau = 1/8$. На тот же график положить результат расчета, полученный явной схемой, при $\tau = 1/128$. На графике должны присутствовать обе компоненты вектора решения (т.е. всего должно быть 8 кривых). Для построения графика удобно использовать, например, такие команды:

Python:

```
import matplotlib.pyplot as plt

plt.plot(t, cros, 'r+-', label='cros')
plt.plot(t, ros05, 'g+-', label='ros05')
plt.plot(t, ros1, 'b+-', label='ros1')
plt.plot(t0, rk2, 'k', label='rk2')
plt.legend()
plt.show()
```

MATLAB:

```
plot(t,cros,'r+-',t,ros05,'g+-',t,ros1,'b+-',t0,rk2,'k');
```

Здесь $t$ задает редкую сетку по времени, а $t_0$ – более подробную.

Объяснить поведение кривых численного решения в терминах $A$-устойчивость, $Lp$-устойчивость, $t$–монотонность.

# Семинар 4

## 4.1 Теоретический материал к семинару №4

Дифференциально-алгебраическая система содержит как дифференциальные, так и алгебраические уравнения. В общем виде она записывается как

$$\begin{cases} \mathbf{G}\dfrac{d\mathbf{u}}{dt} = \mathbf{F}\left(\mathbf{u}, t\right) \\ \mathbf{u}(t_0) = \mathbf{u}_0 \end{cases} \tag{4.1}$$

где $\mathbf{G}$ - матрица коэффициентов при производных.
Для ее решения можно использовать семейство схем Розенброка, заменив в них матрицу $\mathbf{E}$ матрицей $\mathbf{G}$

$$\begin{cases} \left(\mathbf{G} - \alpha\tau\mathbf{F}_{\mathbf{u}}\right)\boldsymbol{\omega} = \mathbf{F}\left(\mathbf{u}, t + \dfrac{\tau}{2}\right) \\ \hat{\mathbf{u}} = \mathbf{u} + \tau\,\mathrm{Re}\,\boldsymbol{\omega} \end{cases} . \tag{4.2}$$

Здесь $\alpha$ - параметр схемы, $\tau$ - шаг схемы по времени, $\mathbf{F}_{\mathbf{u}}$ - производная правой части по переменной $\mathbf{u}$, $\mathbf{u}$ и $\hat{\mathbf{u}}$ - численное решение в текущий и следующий моменты времени соответственно.
Для получения второго порядка в одностадийной схеме Розенброка для задачи Коши необходимо брать $\alpha = \frac{1}{2}$ или $\alpha = \frac{1+i}{2}$. Для неавтономных дифференциально-алгебраических систем даже при этих $\alpha$ получить второй порядок точности не удается из-за противоречивых требований к выбору смещения по времени при вычислении правой части. Поэтому для получения второго порядка необходимо провести автономизацию, то есть убрать явную зависимость от времени в правой части.

# Задачи к семинару №4

1. Требуется рассчитать работу транзисторного усилителя. Для этого необходимо решить дифференциально-алгебраическую систему.

   Параметры электрической схемы:

```
r0, r1, r2, r3, r4, r5 = 1000, 9000, 9000, 9000, 9000, 9000
c1, c2, c3 = 1e-6, 2e-6, 3e-6
ub = 6
```

   Начальные условия:

   Python:

```
u0 = np.array([0, ub*r1/(r1+r2), ub*r1/(r1+r2), ub, 0])
```

   MATLAB:

```
u0 = [0; ub*r1/(r1+r2); ub*r1/(r1+r2); ub; 0];
```

   Матрица коэффициентов при производных:

   Python:

```
G = np.array([[-c1,  c1,  0,   0,   0],
              [ c1, -c1,  0,   0,   0],
              [ 0,    0, -c2,  0,   0],
              [ 0,    0,  0,  -c3,  c3],
              [ 0,    0,  0,   c3, -c3]])
```

   MATLAB:

```
G = [-c1  c1  0   0   0;
      c1 -c1  0   0   0;
      0   0  -c2  0   0;
      0   0   0  -c3  c3;
      0   0   0   c3 -c3];
```

Правая часть:

Python:

```python
def ue(t):
    return 0.1 * np.sin(200 * np.pi * t)

def ff(u):
    return 1e-6 * (np.exp(u / 0.026) - 1)

def F(t, u):
    global r0, r1, r2, r3, r4, r5, ub
    y = np.array([u[0]/r0 - ue(t)/r0,
                  0.01 * ff(u[1]-u[2]) - ub/r2 + u[1]*(1/r1 + 1/r2),
                  u[2]/r3 - ff(u[1]-u[2]),
                  0.99 * ff(u[1]-u[2]) - ub/r4 + u[3]/r4,
                  u[4]/r5])
    return y
```

MATLAB:

```
function y = ue(t)
    y = 0.1*sin(200*pi*t);
end

function y = ff(u)
    y = 1e-6*(exp(u/0.026)-1);
end

function y = F(t, u)
    global r0 r1 r2 r3 r4 r5 ub;
    y = [u(1)/r0-ue(t)/r0;
         0.01*ff(u(2)-u(3))-ub/r2+u(2)*(1/r1+1/r2);
         u(3)/r3-ff(u(2)-u(3));
         0.99*ff(u(2)-u(3))-ub/r4+u(4)/r4;
         u(5)/r5                                    ];
end
```

Расчет провести с шагом $h = 1/5000$ на временном отрезке от 0 до 0.3 при $\alpha = \frac{1}{2}$ и $\alpha = \frac{1+i}{2}$. Вывести результат расчета на график (на одном графике будет 2 семейства кривых для двух разных $\alpha$, в каждом семействе по 5 кривых, соответствующих 5 компонентам $\mathbf{u}$). Объяснить разницу между численными решениями при разных $\alpha$.

2. Определить эффективный порядок метода с помощью сгущения сеток. Для экономии времени расчет вести до $t = 0.01$. Провести автономизацию и вновь определить эффективный порядок.

# Семинар 5

## 5.1 Теоретический материал к семинару №5

Необходимо решить краевую задачу

$$\begin{cases} \dfrac{d}{dx}\left(\left(k_0 + k_1 u^2\right)\dfrac{du}{dx}\right) = f(x) \\ u(a) = u(b) = 0 \end{cases}, \tag{5.1}$$

где $[a; b]$ - отрезок, на котором ищется решение.
Для этой задачи можно написать следующую разностную схему:

$$\begin{cases} (u_{n+1} - u_n)\left(k_0 + k_1\dfrac{u_n^2 + u_{n+1}^2}{2}\right) - (u_n - u_{n-1})\left(k_0 + k_1\dfrac{u_n^2 + u_{n-1}^2}{2}\right) - h^2 f_n = 0 \\ u_0 = u_N = 0 \end{cases}. \tag{5.2}$$

Здесь $h$ - шаг равномерной сетки, $N$ - число интервалов сетки. Данная система уравнений является нелинейной и решать ее лучше всего методом Ньютона.
Многомерный метод Ньютона для задачи

$$\mathbf{F}(\mathbf{x}) = 0 \tag{5.3}$$

является итерационным и выглядит так:

$$\begin{cases} \dfrac{\partial \mathbf{F}(\mathbf{x}_s)}{\partial \mathbf{x}_s}\boldsymbol{\Delta}\mathbf{x}_s = -\mathbf{F}(\mathbf{x}_s) \\ \mathbf{x}_{s+1} = \mathbf{x}_s + \boldsymbol{\Delta}\mathbf{x}_s \end{cases}, \tag{5.4}$$

где $\dfrac{\partial \mathbf{F}(\mathbf{x}_s)}{\partial \mathbf{x}_s}$ - матрица первых производных. В качестве условия окончания итераций можно взять

$$\|\boldsymbol{\Delta}\mathbf{x}_s\| < \varepsilon, \tag{5.5}$$

где $\varepsilon \approx 10^{-12}$ для 64-разрядных вычислений. Сходимость метода сильно зависит от выбранного начального приближения. Вблизи решения сходимость квадратичная, вдали от него ее может вообще не быть. Одно из возможных решений – использовать вектор из случайных чисел. В случае неудачи можно сгенерировать еще один и повторить расчет, пока расчет не получится.

# Задачи к семинару №5

Уточним условие. Отрезок, на котором следует искать решение - $[-1; 1]$, число интервалов сетки $N = 128$, $k_0 = 1$, $k_1 = 0.05$, в качестве начального приближения в данном случае удобно брать нулевой вектор. Функция $f(x)$ в правой части задается так:
Python:

```python
import numpy as np

def ff(x):
    return 100 * np.exp(-np.square(10 * (x - 0.5)))
```

MATLAB:

```
function y = ff(x)
    y = 100*exp(-(10*(x-0.5)).^2);
end
```

Построить график решения.

# Семинар 6

## 6.1 Теоретический материал к семинару №6

Видоизменим краевую задачу из предыдущего семинара, сделав из нее задачу на собственные значения

$$\begin{cases} \dfrac{d}{dx}\left(\left(k_0 + k_1 u^2\right)\dfrac{du}{dx}\right) + \lambda u = 0 \\ u(a) = u(b) = 0 \end{cases}, \tag{6.1}$$

где $[a; b]$ - отрезок, на котором ищется решение.

Разностная схема легко получается из схемы для краевой задачи заменой $-h^2 f_n$ на $h^2 \lambda u_n$:

$$\begin{cases} (u_{n+1} - u_n)\left(k_0 + k_1 \dfrac{u_n^2 + u_{n+1}^2}{2}\right) - (u_n - u_{n-1})\left(k_0 + k_1 \dfrac{u_n^2 + u_{n-1}^2}{2}\right) + h^2 \lambda u_n = 0 \\ u_0 = u_N = 0 \end{cases}. \tag{6.2}$$

Здесь $h$ - шаг равномерной сетки, $N$ - число интервалов сетки.

Поскольку это нелинейная задача на собственные значения, то для ее решения надо использовать метод дополненного вектора – вариант метода Ньютона для задач на собственные значения.

В этом методе строится новый вектор по следующему правилу

$$\begin{cases} \nu_n = u_n, & 0 \le n \le N \\ \nu_{N+1} = \lambda \end{cases}. \tag{6.3}$$

Добавление новой переменной потребует увеличения на одно числа уравнений – иначе задача не будет иметь однозначного решения. Так мы снова приходим к необходимости постановки дополнительного граничного условия в задаче на собственные значения.

Дальше задача решается методом Ньютона, подобно тому, как это делалось при решении нелинейной краевой задачи в предыдущем семинаре. Переделки программы будут минимальными.

В данном случае очень полезно провести серию расчетов на сгущающихся сетках и понаблюдать за сходимостью серии найденных собственных значений к некоторому пределу. Это важно, чтобы вовремя заметить «перескок» на другое собственное значение, если он будет иметь место. При «перескоке» серию расчетов придется повторить с другим начальным приближением.

Ясно, что результат расчета на более грубой сетке следует использовать в качестве начального приближения при расчете на более подробной. Для равномерной сетки и

сгущения сетки в 2 раза значения в нечетных узлах переносятся непосредственно (1 в 3, 2 в 5 и т.п.), а в четных получаются интерполяций – полусуммой соседних нечетных узлов (например $u_2 = 0.5(u_1 + u_3)$). Вычисленное на грубой сетке собственное значение переносится на подробную непосредственно, выполняя роль начального приближения. К полученному в результате серии расчетов набору приближенных значений $\lambda$ можно применять методы апостериорной оценки погрешности решения Ричардсона и Эйткена. Используя рекуррентное сгущение, можно получить результат с высокой точностью даже на не слишком подробных сетках.

# Задачи к семинару №6

Задачу следует решать на отрезке $[0; 1]$, $k_0 = 1$, $k_1 = 0.5$, начальное приближение для собственного значения $\lambda_0 = 40$, самая первая сетка пусть имеет 8 интервалов, последняя – 512 (серия из 7 расчетов).

Дополнительное граничное условие выглядит следующим образом (здесь `u` - дополненный вектор):

Python:

```
u[(len(u)//2) + 1] - u[(len(u)//2) - 1] - 2 * h = 0.
```

MATLAB:

```
u(end/2+1) - u(end/2-1) - 2*h = 0.
```

В данном случае ставится условие на производную собственной функции в середине отрезка (она должна быть равна 1).

Построить график $\lambda$ от номера расчета. Также вывести на график собственную функцию с самой подробной сетки. По полученному набору $\lambda$ определить эффективный порядок метода и получить апостериорную оценку погрешности. С помощью техники рекуррентных сгущений вычислить собственное значение с максимально возможной точностью.

# Семинар 7

## 7.1 Теоретический материал

Одномерное уравнение переноса имеет следующий вид:

$$\begin{cases} \dfrac{\partial u}{\partial t} + c\dfrac{\partial u}{\partial x} = f(x,t) \\ u(0,t) = \mu_1(t) \\ u(x,0) = \mu_2(x) \end{cases} . \tag{7.1}$$

Здесь $c > 0$ - скорость переноса, $\mu_1(t)$ - граничное, а $\mu_2(x)$ - начальное условия. Аппроксимируя производные конечными разностями, получим ряд разностных схем для этого уравнения:

$$\frac{\hat{u}_n - u_n}{\tau} + c\frac{u_n - u_{n-1}}{h} = \Phi_n, \tag{7.2}$$

$$\frac{\hat{u}_{n-1} - u_{n-1}}{\tau} + c\frac{\hat{u}_n - \hat{u}_{n-1}}{h} = \Phi_n, \tag{7.3}$$

$$\frac{\hat{u}_n - u_n}{\tau} + c\frac{\hat{u}_n - \hat{u}_{n-1}}{h} = \Phi_n, \tag{7.4}$$

$$\frac{\hat{u}_n + \hat{u}_{n-1} - u_n - u_{n-1}}{\tau} + c\frac{\hat{u}_n + u_n - \hat{u}_{n-1} - u_{n-1}}{h} = 2\Phi_n, \tag{7.5}$$

где $\tau$ - шаг по времени сетки, $h$ - шаг по пространству, а $\Phi_n = f(x_{n-1/2}, t_{n+1/2})$ - значение функции из правой части уравнения в середине ячейки.

Первые две схемы условно устойчивы, две последние – безусловно устойчивы. Граница устойчивости задается с помощью неравенства, в которое входит число Куранта $\kappa = c\tau/h$. Первая схема устойчива при $\kappa \leq 1$, а вторая – при $\kappa \geq 1$. Таким образом, их условия устойчивости противоположны. Это дает возможность построить из них так называемую составную схему, называемую также схемой Карсона. Идея в том, чтобы при $\kappa \leq 1$ использовать первую схему, а при $\kappa \geq 1$ - вторую. В итоге составная схема получается безусловно устойчивой. Если число Куранта $\kappa$ переходит через единицу, например из-за изменения скорости переноса или неравномерности сетки, то часть шагов может быть сделана по первой схеме, а часть - по второй. Оказывается, что составная схема оказывается несколько точнее безусловной устойчивой чисто неявной схемы (7.4). Схема (7.2) - явная, все остальные - формально неявные, хотя на самом деле считать по ним не труднее, чем по явным, поскольку все неизвестные величины на следующем временном слое получаются либо из граничного условия, либо из начального, либо из результата предыдущего расчета. Важно лишь соблюдать правильный порядок вычислений - от левой границы области расчета к правой и от более раннего временного слоя к более позднему.

## 7.2 Задание

Задачу будем решать на отрезке $[0; 100]$ по пространству и $[0;\ 1]$ по времени. Шаг равномерной сетки по пространству $h = 0.1$, шаг по времени $\tau = 0.01$. Скорость переноса $c = 50$. Правая часть - нулевая. Начальное условие

$$\mu_2(x) = \frac{1}{1 + \left(\dfrac{x - 20}{10}\right)^{10}}.$$

Граничное условие $\mu_1(t) = \mu_2(-ct)$.

Реализовать составную схему, чисто неявную схему (7.4) и схему с полусуммой (7.5). Положить на один график точное решение задачи $\mu_2(x - ct)$, а также результаты расчета по всем трем схемам. Объяснить поведение кривых численного решения.

# Семинар 8

## 8.1 Задание

Решить квазилинейное уравнение переноса

$$\begin{cases} \dfrac{\partial u}{\partial t} + u\dfrac{\partial u}{\partial x} = 0 \\ u(x,0) = \dfrac{1}{1 + \left(\dfrac{x-50}{10}\right)^4} \\ u(0,t) = u(0,0) \end{cases} . \tag{8.1}$$

Задачу следует решать на отрезке $[0; 100]$ по пространству. Шаг равномерной сетки по пространству $h = 1$, шаг по времени $\tau = 0.05$. Число шагов по времени $N = 1000$. На каждом временном слое отображать решение на графике с помощью команды `plot`. В результате должна получиться анимационная картинка. Опыт показывает, что для поучения эффекта анимации необходимо после каждой команды рисования вставить команду паузы с минимальным временем задержки, например `pause(1e-6)`.
Расчет вести с помощью консервативной чисто неявной схемы для квазилинейного уравнения переноса

$$\frac{\hat{u}_n - u_n}{\tau} + \frac{\hat{u}_n^2 - \hat{u}_{n-1}^2}{2h} = 0. \tag{8.2}$$

# Семинар 9

## 9.1 Задание

Решить однородное уравнение теплопроводности с граничными условиями Дирихле

$$\begin{cases} \dfrac{\partial u}{\partial t} = k\dfrac{\partial^2 u}{\partial x^2} \\ u(x,0) = e^{-(x-5)^4} + 0.01x \\ u(0,t) = u(0,0) \\ u(a,t) = u(a,0) \end{cases} \tag{9.1}$$

на отрезке $x \in [0;a]$ при $t \in [0;T]$ Выбрать $a = 20$ и $T = 20$, а коэффициент теплопроводности $k = 2$. Шаг по пространству $h = 0.01$, шаг по времени $\tau = 0.05$. Расчет проводить с помощью комплексной схемы Розенброка. После каждого временного слоя выводить решение на текущем слое для получения анимационной картинки. Чтобы избежать постоянного изменения масштаба графика рекомендуется после команды `plot` вставить команду `axis([0 a 0 1])`, не забыв далее поставить команду `pause(1e-6)`. Тот же расчет необходимо повторить с граничными условиями Неймана

$$u_x(0,t) = u_x(a,t) = 0. \tag{9.2}$$

## 9.2 Указания

1. Счет будет идти быстрее, если использовать аппарат разреженных матриц MATLAB (смотри документацию к функции `spdiags`).

2. Граничные условия надо включить в оператор пространственного дифференцирования $\Lambda_x$, видоизменив в нем первую и последнюю строки. Поскольку в данной задаче коэффициент теплопроводности постоянен, то оператор $\Lambda_x$ можно вычислить один раз до начала расчета.

3. Для аппроксимации граничных условий Неймана со 2 порядком следует использовать метод фиктивных точек.

# Семинар 10

## 10.1 Задание

Решить двумерное однородное уравнение теплопроводности с граничными условиями Дирихле

$$\begin{cases} \dfrac{\partial u}{\partial t} = k\left(\dfrac{\partial^2 u}{\partial x^2} + \dfrac{\partial^2 u}{\partial y^2}\right) \\ u(x,y,0) = \mu(x,y) \\ u(0,y,t) = \mu(0,y), \qquad u(a,y,t) = \mu(a,y) \\ u(x,0,t) = \mu(x,0), \qquad u(x,b,t) = \mu(x,b) \end{cases} \tag{10.1}$$

в области $(x,y,t) \in [0;a] \times [0;b] \times [0;T]$.
Функция $\mu(x,y)$ задается следующим образом (`x` и `y` - вектора, содержащие координаты узлов прямоугольной сетки):

```
function z = mu(x, y)
    z = zeros(length(y), length(x));
    for i = 1:length(x)
        for j = 1:length(y)
            z(j,i) = -0.01*sin(x(i)) + 0.05*sin(y(j));
        end
    end
end
```

Параметры области: $a = 6\pi$, $b = 4\pi$, $T = 10$. Выбрать равномерную сетку с $h_x = h_y = \pi/30$ и шагом по времени $\tau = 0.1$. Коэффициент теплопроводности $k = 0.2$. Отображать двумерное решение на каждом временном слое с помощью функции `mesh`.

## 10.2 Указания

Для решения использовать эволюционно-факторизованную схему

$$\begin{cases} \left(E - \dfrac{\tau}{2}\Lambda_x\right)\nu = \Lambda_x u + \Lambda_y u \\ \left(E - \dfrac{\tau}{2}\Lambda_y\right)\Delta u = \nu \\ \hat{u} = u + \tau \Delta u \end{cases} . \tag{10.2}$$

Здесь $\Lambda_x$ и $\Lambda_y$ - операторы пространственного дифференцирования. Проблем с промежуточным граничным условием здесь не возникает, так как $u$ на границе не меняется со временем, а значит на границе $\hat{u} - u = 0$ и $\nu = 0$.
Операторы $P_x = E - \frac{\tau}{2}\Lambda_x$ и $P_y = E - \frac{\tau}{2}\Lambda_y$ могут быть записаны в матричной форме с помощью замены $\Lambda_x$ и $\Lambda_y$ на соответствующие матрицы пространственного дифференцирования $L_x$ и $L_y$. При этом надо помнить, что

$$\begin{aligned} \Lambda_x u &= u L_x \\ \Lambda_y u &= L_y u \end{aligned}, \tag{10.3}$$

т.е. матрицы пространственного дифференцирования $L_x$ и $L_y$ умножаются на матрицу значений в узлах сетки $u$ с разных сторон. Аналогично при обращении операторов $P_x$ и $P_y$ надо использовать правое и левое матричное деление соответственно.
Не следует применять операторы пространственного дифференцирования $\Lambda_x$ и $\Lambda_y$ к крайним строкам и столбцам матрицы $u$, так как в таком случае в $\Delta u$ в крайних строках и столбцах будут содержаться ненулевые значения, что приведет к нарушению граничных условий в расчете.

# Семинар 11

## 11.1 Задание

Решить эллиптическое уравнение

$$\begin{cases} \dfrac{\partial^2 u}{\partial x^2} + \dfrac{\partial^2 u}{\partial y^2} = 0 \\ u(x,y) = \mu(x,y)|_{(x,y)\in\Gamma} \end{cases} \tag{11.1}$$

в области $(x, y) \in [0; a] \times [0; b]$ методом счета на установления с логарифмическим набором шагов. Здесь множество Г - граница расчетной области. Функцию $\mu(x, y)$, размеры области и параметры сетки взять из задания к предыдущему семинару. Необходимо построить график невязки (левой части (11.1)) в норме $C$ (максимум модуля) от номера итерации. В конце построить результирующий трехмерный профиль. Параметр $\varepsilon$, необходимый для построения логарифмического набора шагов, взять равным $10^{-12}$.

## 11.2 Указания

Для решения этой задачи удобно модифицировать программу с прошлого семинара. Фактически задача сводится к построению логарифмического набора шагов. Для этого необходимо определить границы спектра, вычислить необходимое число шагов и собственно построить логарифмический набор по известным границам спектра и числу шагов в наборе. Следует иметь в виду, что в двумерном случае следует взять половину того числа шагов, которое дает оценочная формула для одномерного случая. Для обеспечения соблюдения граничных условий в качестве начального приближения для $u(x, y)$ удобно взять $\mu(x, y)$.

# Семинар 12

## 12.1 Теоретический материал

Для решения гиперболического уравнения

$$\begin{cases} \dfrac{\partial^2 u}{\partial t^2} = c^2 \dfrac{\partial^2 u}{\partial x^2} + f(x,t) \\ u(0,t) = \mu_1(t) \\ u(a,t) = \mu_2(t) \\ u(x,0) = \mu_3(x) \\ u_t(x,0) = \mu_4(x) \end{cases} \tag{12.1}$$

в области $(x,t) \in [0;a] \times [0;T]$ можно использовать «схему с весами»

$$\frac{1}{\tau^2}\left(\hat{u} - 2u + \check{u}\right) = \Lambda\left(\sigma\hat{u} + (1-2\sigma)\,u + \sigma\check{u}\right) + f, \tag{12.2}$$

где $\Lambda$ - оператор пространственного дифференцирования (с учетом умножения на $c^2$). Выражение (12.2) можно преобразовать так:

$$\left(E - \sigma\Lambda^*\right)\left(\hat{u} - 2u - \check{u}\right) = \Lambda^* u, \tag{12.3}$$

где $\Lambda^* = \tau^2\Lambda$. Теперь нетрудно выразить решение на текущем слое через два предыдущих слоя

$$\hat{u} = \left(E - \sigma\Lambda^*\right)^{-1}\Lambda^* u + 2u - \check{u} \tag{12.4}$$

Для вычисления решения на первом слое надо использовать следующую формулу:

$$\begin{gathered} u(x,\tau) ??? u(x,0) + \tau u_t + \frac{\tau^2}{2} u_{tt} = \\ = u(x,0) + \tau\mu_4 + \frac{\tau^2}{2}\left(c^2 u_{xx} + f\right) ??? u(x,0) + \tau\mu_4 + \frac{1}{2}\left(\Lambda^*\mu_3 + \tau^2 f\right). \end{gathered} \tag{12.5}$$

В случае независящих от времени граничных условий Дирихле удобнее всего обеспечить их выполнение, занулив первую и последнюю строку в матрице оператора $\Lambda^*$. Именно так это делалось ранее при решении уравнения теплопроводности с аналогичными граничными условиями.

Для контроля правильности счета следует провести несколько расчетов со сгущением сетки (сгущать сетку необходимо как по пространству, так и по времени) и вычислить эффективный порядок метода. Поскольку теоретическая оценка погрешности метода

$O(\tau^2 + h^2)$, то при одновременном сгущении сетки по пространству и времени в одно и то же число раз должен получиться второй порядок.

Сгущать сетку удобнее всего каждый раз вдвое, а погрешность рассчитывать по общим узлам двух соседних вложенных сеток. Наиболее удобная норма в данном случае – чебышевская, или норма $C$. Формула расчета эффективного порядка по трем сеткам выглядит так:

$$p = -\log_2 \frac{\|U_{4N} - U_{2N}\|_C}{\|U_{2N} - U_N\|_C}. \tag{12.6}$$

Разности вычисляются по общим узлам двух сеток.

## 12.2 Задание

1. Решить задачу (12.1) при $f = 0$, $c = 3$, $\mu_1 = \mu_2 = 0$, $\mu_3(x) = \sin(x)$, $\mu_4 = 0$ в области $[0; 6\pi] \times [0; 10]$. Взять $\tau = 0.01$ и $h = 6\pi/100$. Отобразить решение на каждом временном слое.

2. В условии 1 задания взять

$$\mu_3(x) = \sin\left(x\left(1 + 0.1e^{-(x-10)^2}\right)\right),$$

подобрать согласованные граничные условия. Все остальное оставить как в 1 задании. Повторить расчет.

3. В условии 2 задания провести расчет на сгущающихся сетках и доказать второй порядок метода. Задачу решать в области $[0; 6\pi] \times [0; 1]$, для первой сетки взять $\tau = 1/16$ и $h = 6\pi/16$. Провести расчеты на 7 сетках. Решение на каждом слое не отображать. Построить график эффективного порядка от номера самой грубой сетки из трех сеток, участвующих в расчете эффективного порядка.

# Семинар 13

## 13.1 Теоретический материал

При движении тела в атмосфере Земли на него помимо силы тяжести также действует сила сопротивления воздуха

$$F(\nu) = iS\frac{\rho\nu^2}{2}C_x\left(\frac{\nu}{a}\right) \tag{13.1}$$

где $i$ - коэффициент формы, $S$ - площадь лобового сечения, $\rho$ - плотность среды, $C_x$ - закон сопротивления, $a$ - скорость звука.
Для тела с круглым сечением, очевидно,

$$S = \frac{\pi d^2}{4}. \tag{13.2}$$

Плотность атмосферы $\rho$ является функцией виртуальной температуры $T_\nu$, давления $P_0$ в точке бросания, высоты полета $y$

$$\rho = \rho(y) = 1.225\left(\frac{T(y)}{T_\nu}\right)^{4.256}\frac{P_0}{T_\nu}\frac{288.15}{760}, \tag{13.3}$$

где $T(y)$ - зависимость температуры от высоты

$$T(y) = T_\nu - 0.0065y. \tag{13.4}$$

Скорость звука $a$ зависит от температуры в точке полета следующим образом:

$$a = 340.294\left(\frac{T(y)}{288.15}\right)^{1/2}. \tag{13.5}$$

Для учета влажности необходимо во всех формулах вместо реальной температуры $T_0$ использовать так называемую виртуальную температуру

$$T_\nu = \frac{T_0 + 273.15}{1 - \frac{3}{8}\frac{12.7}{P_0}w}, \tag{13.6}$$

где $w$ - влажность, выраженная в долях 1. В этой формуле $T_0$ - температура в градусах Цельсия в точке бросания, то есть в числителе дроби стоит абсолютная температура.
Для расчета закона сопротивления $C_x$ можно использовать следующий код:

```
function r = cx(x)
    pa = [0.0525   -0.9476    8.9342   -9.4610   ...
          0.3207    4.2980   -1.9382              ];
    pb = [1.0000   -15.4071   178.6690 -580.8643 ...
          985.5873 -853.9492  296.9213             ];
    pc = [0.0531    0.9449    90.5063   0.1639     ];
    r = polyval(pa,x.^2) ./ polyval(pb,x.^2) + ...
        pc(1) ./ (1+exp(-(x-pc(2))*pc(3))) + pc(4);
end
```

Для учета деривации надо систему дифференциальных уравнений внешней баллистики для движения центра масс тела дополнить еще двумя уравнениями

$$\begin{aligned} \frac{dz}{dt} &= q\nu_x\pi\nu_0 c_d, \\ \frac{dq}{dt} &= \frac{e^{-m_3 t}}{\nu^2}, \end{aligned} \tag{13.7}$$

где $\nu_x$ - горизонтальная компонента скорости, $\nu_0$ - начальная скорость, $\nu$ - модуль скорости, $c_d$ - коэффициент деривации, $m_3$ - коэффициент убывания угловой скорости вращения.

Если метание тела осуществляется с помощью порохового заряда, то начальная скорость тела будет зависеть от температуры заряда $T_z$

$$\nu_0 = \nu_{15}\left(1 + z_t\left(T_z - 15\right)\right), \tag{13.8}$$

где $\nu_{15}$ - начальная скорость при $15°C$, $z_t$ - коэффициент температуры заряда.

Для учета влияния ветра необходимо сперва перейти в систему отсчета, связанную с ветром, где атмосфера неподвижна, затем решить задачу и при необходимости вернуться в исходную систему отсчета.

Заметим, что в баллистике принято углы выражать в так называемых делениях угломера (д.у.). По определению, окружность делится на 6000 таких делений, т.е.

$$1\partial.y. = \frac{\pi}{3000} = 0.06°. \tag{13.9}$$

## 13.2 Задание

1. Определить угол бросания для пули Б-32 пулемета НСВ-12.7 на дальность 2000м с точностью не хуже 0.01 д.у., если

$$\begin{aligned} g &= 9.80665 м/с^2, \\ d &= 12.7 мм, \\ m &= 48.3 г, \\ i &= 1.0629, \\ \nu_{15} &= 820 м/с, \\ P_0 &= 750 мм.рт.с., \\ T_0 &= 15°C, \\ w &= 0.5, \\ z_t &= 1.35 \cdot 10^{-3}, \\ c_d &= 0.0423, \\ m_3 &= 0.1744. \end{aligned}$$

Здесь $m$ - масса пули. Повторить расчет для $T = 5°C$.

***Указание:*** перейти от независимой переменной $t$ к переменной $x$ (координата пули вдоль траектории).

2. Определить угол бросания и горизонтальную угловую поправку с учетом ветра и деривации в условиях предыдущей задачи ($T_0 = 15°C$), если скорость ветра 10м/с, он дует справа налево перпендикулярно траектории, а деривация приводит к смещению пули вправо.

3. Определить максимальную дальность полета в условиях 1 задания ($T_0 = 15°C$).