Задача 3. (Метод Ньютона)

В методе Ньютона осуществляется экстраполяция с помощью касательной к кривой в данной точке.
Возьмем некоторую точку $x_1 \in [a,b]$. Уравнение касательной к графику функции f(x) в точке $(x_1, f(x_1))$ имеет вид:

$$0 - f(x_1) = f'(x_1)*(x_2 - x_1),$$

где x_2 – следующее приближение к корню: $x_2 = x_1 - f(x_1)/ f'(x_1)$
Таким образом, последовательно переходя от x_1 к x_2 и далее к x_3, x_4 … от точки начального приближения переходим к корню уравнения. Корень достигается, когда $|f(x)| < \varepsilon$. ε – требуемая точность решения уравнения.

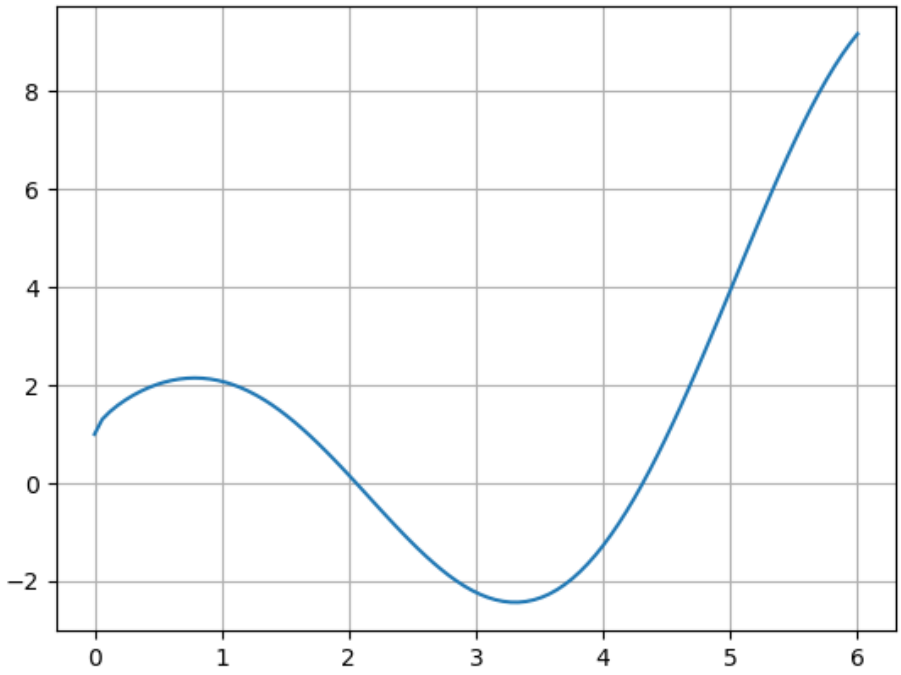
Проиллюстрируем метод на примере решения задач.

Задача 3.1
Найти корень уравнения $f(x) = \sqrt{x} + (x+1)\cos(x)$ на интервале x = [0,6].

Программа поиска корня (Python)

```
# Решение уравнения методом Ньютона
import math
import numpy as np
import matplotlib.pyplot as plt
def funct(x):
    y = math.sqrt(x)+(x+1)*math.cos(x)
    return y
def functxx(x):
     return  np.sqrt(x)+(np.x+1)*np.cos(x)
def prfunct(x):
    y = 0.5/math.sqrt(x)+math.cos(x) - (x+1)*math.sin(x)
    return y
x0 = 2                  # Начальное приближение
eps = 10**(-5)          # Точность вычислений
#print("x0=", x0,"funct = ", funct(x0),"prfunct = ", prfunct(x0))
while (abs(funct(x0))>eps):
    xnew = x0 - funct(x0)/prfunct(x0)
#    print("x0=", x0,"xnew = ", xnew)
    x0 = xnew
print("root = ", x0, " discrepancy", funct(x0))
#    Построение графика функции f(x) f(x)=√x+(x+1)cos(x)  на интервале x = [0,6]
n = 100
x = np.linspace(0, 6, n)
y = np.zeros(n)
for i in range(n):
    y[i] = funct(x[i])
plt.plot(x,y)
plt.grid(True)
plt.show()
```

Распечатка результата
root = 2.059045263010118 discrepancy -2.7079375852778753e-08



Из графика функции $f(x) = \sqrt{x} + (x+1)\cos(x)$ следует, что на интервале x = [0,6] существует два корня. Первый корень root = 2.059045263010118 discrepancy -2.7079375852778753e-08 был определен при задании начального приближения x0 = 2. Если задать начальное приближение x0 = 4, то программа определит второй корень - root = 4.31072633971601 discrepancy 4.677549138243364e-06 .

Решение уравнения, с использованием GNU Octave.

На этом примере продемонстрируем интервальное расширение системы компьютерной математики Octave.

Литература:

- С.П.Шарый. Конечномерный интервальный анализ. Издательство «XYZ» Новосибирск 2021. 646 с.
- Алексеев Е.Р., Чеснокова О.В. GNU Octave для студентов и преподавателей. - Донецк.: ДонНТУ, Технопарк ДонНТУ УНИТЕХ, 2011. - 332с.

Решение в GNU Octave.

Набираем в «Командном окне»

```
1>> f=@(x)  sqrt(x)+(x+1).*cos(x);
2>> df=@(x)  -(x+1).*sin(x)+cos(x)+1./(2*sqrt(x));
3>> fzero (f, infsup ("[0,6]"),df)
```

Ответ. В «Командном окне» представлено два интервала нахождения корня [2.059, 2.0591] и [4.3107, 4.3108]

```
ans = 2x1 interval vector

   [2.059, 2.0591]
```

[4.3107, 4.3108]

Комментарий.

В строке 1 задана функция $f(x) = \sqrt{x} + (x+1)\cos(x)$ Для функции используется псевдоним с помощью знака @. Тот факт, что в функции обрабатывается каждый элемент множества x, перед знаком операции ставится точка («.»)

В строке 2 задана производная от функции - $f'(x)$.

В строке 3 ищутся корни посредством обращения к функции fzero (функция поиска корней нелинейного уравнения).

Пример обращения к функции fzero в полном формате:

[X,Y]=fzero('f1', [a b]);

X – решение уравнения, Y - значение функции в точке X, f1 – имя функции, вычисляющей левую часть тождества, a, b – интервал нахождения корня.

Проверка решения, построением графика.

Программа набирается в Редакторе. Предназначена для построения графика функции $f(x) = \sqrt{x} + (x+1)\cos(x)$.

```
% Задание функции в файле fN.m
function f = fN (x)
f =  sqrt(x)+(x+1).*cos(x);
endfunction

% Программа в файле intervalN.m
cla;              % Очистка графических объектов
okno1=figure();   % Создание окна с идентификатором okno1
a=0.1;          % Задаем начальный интервал нахождения корня
b=6;
format short      % Задание формата вывода чисел
%printf("    There are a= %d \n ", a )  %  Для контроля
%printf("    There are b= %d \n ", b )
%yy1 = fN(a)
%yy2 = fN(b)
x= 0:0.2:6; % Создание массива переменой x
y=fN(x);              % Создание массива переменой y
pol=plot(x,y);        % Создание графика
set(pol,'LineWidth',3,'Color','k'); % Установка толщины и цвета линии
set(gca,'xlim',[a,b]);              % Установка осей
set(gca,'ylim',[-5,10]);
set(gca,'xtick',[0:0.5:6]);    % Вывод сетки с заданным диапозоном
set(gca,'ytick',[-6:2:10]);
grid on;                        % Установка сетки
xlabel('x');ylabel('y');             % Подписи осей
title('Plot y = sqrt(x)+(x+1).*cos(x)');        % подпись графика
```

График представлен на рисунке

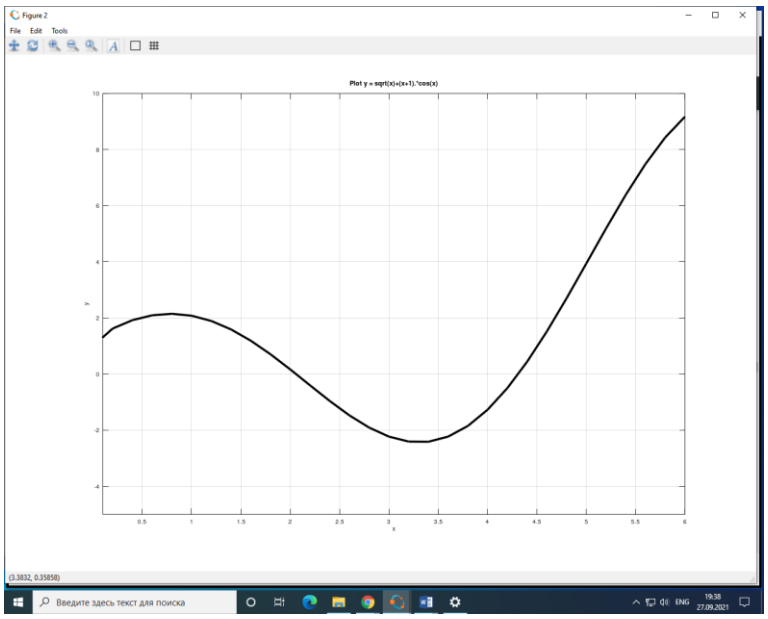


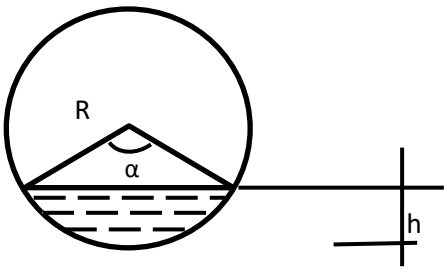
Рис. 1. График функции $f(x) = \sqrt{x} + (x+1)\cos(x)$

Из графика следует, что уравнение $f(x) = \sqrt{x} + (x+1)\cos(x) = 0$ имеет два корня, то есть линия графика пересекает линию y = 0 в двух точках, что отвечает интервалам найденным в GNU Octave.

Задача

Резервуар для жидкости имеет форму лежащего цилиндра с радиусом R. Рассчитать шкалу для соответствующего измерительного стержня, то–есть, например, определить высоты всех уровней, отвечающих наполнениям 5, 10, 15…50%.

Решение.



При длине резервуара l зависимость объема жидкости V от угла α, выраженного в радианах, дается выражением

$$V = r^2 l * \frac{1}{2}(\alpha - sin\alpha).$$

Если резервуар заполнен на q своей ёмкости, то $V = r^2 l\pi q$. При задании q в процентах от объема емкости $V = r^2 l\pi q/100$.

Из сравнения двух последних выражений приходим к уравнению относительно α

$$\alpha - sin\alpha - 2\pi q/100 = 0. \qquad (1)$$

Для решения (1) воспользуемся методом Ньютона

$$\alpha_{n+1} = \alpha_n - \frac{\alpha_n - \sin\alpha_n - 2\pi q/100}{1 - \cos\alpha_n}.$$

В качестве начального приближения α_0 естественно взять линейную функцию

$$\alpha_0 = \frac{\pi}{50} q, \;\; q[\%].$$

Высота уровня h, соответствующая наполнению резервуара на q своей ёмкости, связана с углом α соотношением $h = r\left(1 - cos\frac{\alpha}{2}\right)$.

Листинг программы решения задачи в GNU Octave (Вариант Болтачев И., КоролевК.)

```
% В файле с именем fx записываем вид функции (1)
function fx = fx(x,q)
fx = x-sin(x) - 2*pi*q/100.0;
endfunction
% В файле с именем pfx записываем вид производной от функции (1)
function pfx = pfx(x)
pfx = 1.0 - cos(x);
endfunction
% В файле с именем task3 составляем программу реализации алгоритма решения задачи
eps = 1e-7;          % Задание точности решения уравнения (невязка)
for kq = 5:5:50    % Формирование шкалы измерительного стержня
  x = pi*kq/50.0;  % Начальное приближение корня
 for i = 1:100       % Итерационный цикл поиска корня
   xnew = x - fx(x,kq)/pfx(x);    % Новое значение корня
   nev = abs(fx(xnew, kq));       % Невязка
   if ( nev < eps)                    % Проверка условия выхода из итерационного цикла
     hh = 1 - cos(xnew/2);          % Высота уровня жидкости
% Печать таблицы результатов.
% Объем, h/r, Корень уравнения, Невязка, Число итераций для достижения корня
     printf(" q = %i   \t h/r = %f \t    x= %f \t nev = %e \t n= %i \n", kq, hh,xnew,nev,i);
     break
   end
   x=xnew;
  end
 end
 % Построение графика функции f(α)= α-sinα-2πq/100=0 для случая q = 25%.
q = 25;
x = pi*q/50;
cla;                        % Очистка графических объектов
okno1=figure();         % Создание окна с идентификатором okno1
x= 2:0.01:3;              % Создание массива переменой x
y=fx(x,q);                % Создание массива переменой y
pol=plot(x,y, x, 0);    % Создание графика (два графика: y=f(x), y=0 )
set(pol,'LineWidth',3,'Color','k') % Установка толщины и цвета линии
set(gca,'xlim',[2,3]);           % Установка осей
set(gca,'ylim',[-1,1]);
set(gca,'xtick',[2:0.25:3]);     % Вывод сетки с заданным диапазоном
set(gca,'ytick',[-1:0.5:2]);
grid on;                          % Установка сетки
xlabel('x');ylabel('y');           % Подписи осей
title('Plot f(x)=x-sin(x)-2πq/100');      % подпись графика
```

Результат решения задачи (в командном окне)

```
>> task_3
q = 5     h/r = 0.194616     x= 1.268948   nev = 2.844069e-11    n= 16
 q = 10    h/r = 0.312951     x= 1.626753   nev = 1.601431e-10    n= 5
 q = 15    h/r = 0.414863     x= 1.891494   nev = 7.813084e-11    n= 5
 q = 20    h/r = 0.508138     x= 2.113139   nev = 2.664535e-15    n= 5
 q = 25    h/r = 0.596027     x= 2.309881   nev = 2.847202e-10    n= 4
 q = 30    h/r = 0.680308     x= 2.490785   nev = 1.698641e-13    n= 4
 q = 35    h/r = 0.762118     x= 2.661222   nev = 1.509516e-08    n= 3
 q = 40    h/r = 0.842264     x= 2.824797   nev = 3.015277e-11    n= 3
 q = 45    h/r = 0.921379     x= 2.984188   nev = 8.881784e-16    n= 3
 q = 50    h/r = 1.000000     x= 3.141593   nev = 0.000000e+00    n= 1
```

В результатах представлена шкала измерительного стержня, корень уравнения, невязка корня (то есть значение функции после подстановки в функцию значения корня), а также число итераций, которое потребовалось для достижения корня с заданной степенью точности.

На рисунке 3 представлен график функции $f(x) = \mathrm{x} - \sin(\mathrm{x}) - 2\pi q/100$, который порожден программой. Выделена линия y = 0, позволяющая отметить на графике корень уравнения f(x) = 0.

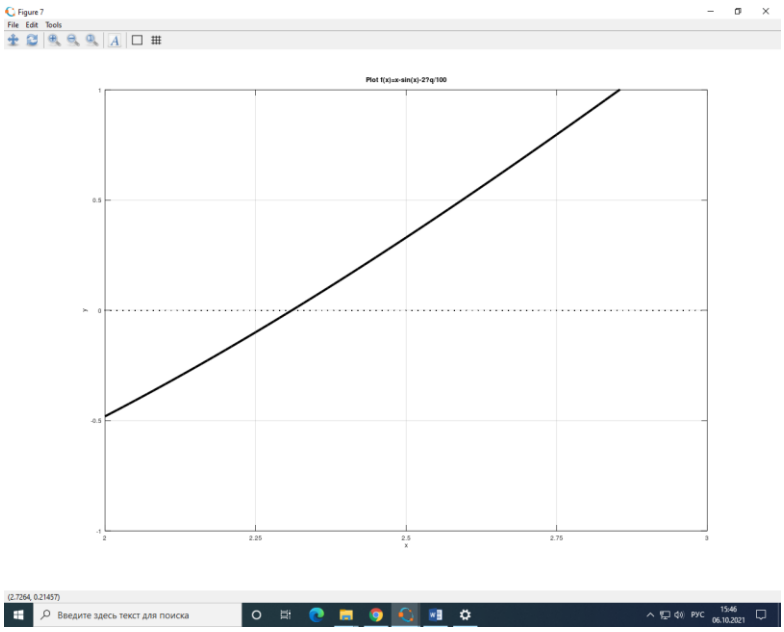


Рис. 3 График функции $f(x) = \mathrm{x} - \sin(\mathrm{x}) - 2\pi q/100$

Листинг программы на C++ решения уравнения методом Ньютона представлен ниже.

```
#include <stdio.h>
#include <math.h>
#include <iostream>
#include <string>
```

```
const double PI = 3.14159265;
double fx(double x, int q)    // Функция определения уравнения
{       double ff;
        ff = x-sin(x) - 2*PI*q/100.0;
   return(ff);
}

double pfx(double x)          // Определение производной от функции
{       double ff;
        ff = 1.0 - cos(x);
   return(ff);
}

int main()
{
        std::cout << "          Newton's method  \n";
        std::cout << "          The equation  x-sin(x) - 2*PI*q/100.0 = 0  \n";

        double x, xnew, nev, hh;
        double eps = 1.0E-7;
        int i, kq, niter = 100;

        for ( kq = 5; kq<=50; kq=kq+5)
        {
           x = PI*kq/50.0;
        for (i = 1; i<niter; i++)
            {
              xnew = x - fx(x,kq)/pfx(x);
              nev = fabs(fx(xnew, kq));
              if ( nev < eps) {             hh = 1.0 - cos(xnew/2);
                 printf(" \t  q = %i %%       h/r = %f     (nev = %e)   \n", kq, hh, nev);       break;}
              x=xnew;
            }
        }
        return 0;
}
```

Листинг с результатом решения –

```
        Newton's method
        The equation  x-sin(x) - 2*PI*q/100.0 = 0
          q = 5 %       h/r = 0.194616     (nev = 3.380640e-11)
          q = 10 %       h/r = 0.312951     (nev = 1.601431e-10)
          q = 15 %       h/r = 0.414863     (nev = 7.813084e-11)
          q = 20 %       h/r = 0.508138     (nev = 2.664535e-15)
          q = 25 %       h/r = 0.596027     (nev = 2.847200e-10)
          q = 30 %       h/r = 0.680308     (nev = 1.705303e-13)
          q = 35 %       h/r = 0.762118     (nev = 1.509516e-08)
          q = 40 %       h/r = 0.842264     (nev = 3.015277e-11)
          q = 45 %       h/r = 0.921379     (nev = 8.881784e-16)
          q = 50 %       h/r = 1.000000     (nev = 4.440892e-16)
```