**Московский авиационный институт**

**(Национальный исследовательский университет)**

Факультет прикладной математики и физики

Кафедра вычислительной математики и программирования

**Лабораторная работа № 1**

по курсу «Математическое моделирование»

Студент: Аксенов А. Е.

Группа: М80-408Б-20

Преподаватель: Харченко Н. А.

Оценка:

Москва, 2023

**Цель работы**

Целью работы является изучение условно и безусловносходящихся итерационных методов решения нелинейных уравнений. Одной из наиболее частых задач, с которыми сталкиваются при вычислительных экспериментах – это решение нелинейных уравнений вида Нелинейные уравнения подразделяются на два вида: алгебраические и трансцендентные.

Алгебраические уравнения содержат только алгебраические функции (рациональные, иррациональные, целые). Данный тип нелинейных уравнений можно представить в виде полинома n-ой степени с действительными коэффициентами:

Трансцендентные уравнения – это уравнения, содержащие показательные, тригонометрические логарифмические и другие функции. К примеру:

Решение ищутся методами последовательных приближений или итерационными методами. Начальное приближение может находится из физических соображений, из опыта решения аналогичных задач, с помощью графических методов и т.д. Поиск корня уравнения математически осуществляется при помощи построения последовательности Коши , когда при заданном ε существует такое N, что для всех n и p, превышающих N, выполняется , причем корень находится внутри этого отрезка неопределенности.

**Постановка задачи**

***Задание 1:*** на клин набегает поток совершенного, невязкого газа с пока-  
зателем адиабаты =1.4. Требуется определить угол наклона КСУ с из уравнения  
(4), результат привести в градусах. Рассчитать параметры (p2, T2, ρ2, M2) за ударной  
волной. Привести все полученные значения в таблицу. Использовать метод и обо-  
значенные индексом 1 параметры набегающего потока, приведенные для каждого  
варианта в таблице 1 (приложение А).

***Задание 2:*** найти корень уравнения на интервале [a, b]. Исходные данные и метод  
определения корня приведены в таблице 2 (приложение А).

***Задание 3:*** использовать метод определения корня из задания No2 для решения урав-  
нения (4) с параметрами из задания No1. Сравнить полученные данные с результата-  
ми задания No1: значение, количество итераций, время расчета. Привести результаты  
сравнения в таблицу. Сделать выводы о проделанной работе.

***Вариант 1:***

*Параметры задания 1:*  
Давление p = 101325 Па  
Температура T = 288 К  
Число Маха M = 3  
Угол наклона клина β = 15  
Метод решения = Метод половинного деления

*Параметры задания 2:*  
Уравнение = x2 − 7x = 0  
Область решения = [6, 8]  
Метод решения = Метод Ньютона

**Метод решения**

Уравнение угла наклона ударной волны имеет вид:

При использовании метода половинного деления считается, что функция непрерывна и имеет на концах интервала разный знак. Находим интервал знакопеременности , содержащий корень данного уравнения, делим отрезок пополам. Если , то - корень уравнения. Если , то выбираем ту из половин или , на концах которой имеет разные знаки. Новый суженный вдвое отрезок снова делим пополам и из двух полученных отрезков снова выбираем интервал знакопеременности . Итерации метода деления пополам прекращаются, если интервал становится достаточно малым.

В методе Ньютона шаг определяется таким уравнением:  
Производная вычислялась по определению:

С достаточно малым *d*. Область решения в 3-ей части подбиралась так же, как и в 1-ой: две случайные точки до и после перемены знака.

**Описание программ**

Функции обоих методов могут обработать любую функцию, принимающую и возвращающую одно дробное значение. Они рассчитаны но то, что функция меняет знак  
один раз в некой близости от начальной точки. Также строятся графики этих функций и считается число итераций каждого метода.

halfdiv.py

import time

import math, sys

import numpy as np

from tabulate import tabulate

import matplotlib.pyplot as plt

def gtr(grad):

""" Перевод из радиан в градусы """

return grad/180\*math.pi

def f(x):

""" В правой части решаемое уравнение """

try:

y = pow(math.sin(gtr(x)),2)-(((1.4+1)/2)\*math.sin(gtr(x))\*math.sin(gtr(15))/(math.cos(gtr(x-15))))-pow(1/3,2)

return y

except ValueError:

print ("Invalid argument")

sys.exit()

def div\_half(a, b, f, eps):

""" Метод дихотомии """

n = 0

if f(a) \* f(b) > 0:

print("No solution")

print("f(a)=",f(a))

print("f(b)=",f(b))

sys.exit()

while True:

x = (a + b) / 2

y = [f(a), f(b), f(x)]

sign = ["0", "0", "0"]

m = 0

for yy in y:

if yy > 0:

sign[m] = "+"

else:

sign[m] = "-"

m+=1

print("%d %.5f %s %s" % ( n, a, sign[0], sign[1]), end=" ")

print("%.5f %.5f %s" % (b, x, sign[2]))

if y[0]\*y[2] > 0:

a = x

else:

b = x

n+=1

if (b - a) <= eps:

break

x = (a + b) / 2

return x

start = time.time()

def main():

# a = float(input("Введите левую границу: "))

# b = float(input("Введите правую границу: "))

# eps = float(input("Введите точность: "))

a = 15.0; b = 60.0; eps = 0.000001

x = div\_half(a, b, f, eps)

print("\n x = %.6f°" % (x))

p1=101325

T1=288

M1=3

R=286.7

ro1=p1/(R\*T1)

print("\nro\_1 = %.6f кг/м^3" % ro1)

u1=1-(math.sin(gtr(x-15))\*math.cos(gtr(x)))/(math.sin(gtr(x))\*math.cos(gtr(x-15)))

print("u\_1 = %.6f" % u1)

v1=M1\*pow(1.4\*R\*T1,0.5)

print("v\_1 = %.6f" % v1)

v2n=(v1\*math.sin(gtr(x))\*math.tan(gtr(x-15)))/math.tan(gtr(x))

print("v\_2n = %.6f" % v2n)

v2=v2n/math.sin(gtr(x-15))

print("v\_2 = %.6f" % v2)

ro2=ro1/(1-u1)

print("\nro\_2 = %.6f кг/м^3" % ro2)

p2=p1+ro1\*u1\*pow(v1\*math.sin(gtr(x)),2)

print("p\_2 = %.6f Па" % p2)

T2=p2/(ro2\*R)

print("T\_2 = %.6f K" % T2)

M2=v2/pow(1.4\*R\*T2,0.5)

print("M\_2 = %.6f" % M2)

"""Отрисовка графика f(x)"""

fv = np.vectorize(f)

t1 = np.arange(0, 90, 0.0001)

plt.plot(t1, fv(t1), label="f(x)")

plt.legend()

plt.grid()

plt.show()

main()

end = time.time() - start

print(end)

newton1.py

import numpy as np

import math

from tabulate import tabulate

import matplotlib.pyplot as plt

def newton(fun, der, x, epsilon=0.001):

iter\_info = [[0, x, fun(x)]]

iter\_number = 1

while True:

new\_x = x - fun(x) / der(x)

iter\_info.append([iter\_number, new\_x, fun(new\_x)])

if np.abs(new\_x - x) < epsilon:

return iter\_info

x = new\_x

iter\_number = iter\_number + 1

#начало

def f(x):

return np.power(x,2) - 7\*x

# функция фи, полученная заменой уравнения f(x) = 0 эквивалентным x = fi(x)

def fi(x):

return np.sqrt(7\*x)

def fi\_der(x):

return np.sqrt(7) / (2 \* np.sqrt(x))

def get\_derivative(fun, epsilon=0.0001):

return lambda x: (fun(x + epsilon) - fun(x)) / epsilon

t1 = np.arange(6, 8, 0.0001)

plt.plot(t1, f(t1), label="f(x)")

plt.legend()

plt.grid()

plt.show()

epsilon = 0.001

newton\_roots = [newton(f, get\_derivative(f, epsilon), 6.0, epsilon), newton(f, get\_derivative(f, epsilon), 8.0, epsilon)]

print("Newton roots for x^2-7x")

for info in newton\_roots:

print(tabulate(info, headers=['iteration', 'x', 'f(x)']))

newton2.py

import time

import numpy as np

import math

from tabulate import tabulate

import matplotlib.pyplot as plt

def gtr(grad):

""" Перевод из радиан в градусы """

return grad/180\*math.pi

def newton(fun, der, x, epsilon=0.001):

iter\_info = [[0, x, fun(x)]]

iter\_number = 1

while True:

new\_x = x - fun(x) / der(x)

iter\_info.append([iter\_number, new\_x, fun(new\_x)])

if np.abs(new\_x - x) < epsilon:

return iter\_info

x = new\_x

iter\_number = iter\_number + 1

#начало

start = time.time()

def f(x):

return pow(math.sin(gtr(x)),2)-(((1.4+1)/2)\*math.sin(gtr(x))\*math.sin(gtr(15))/(math.cos(gtr(x-15))))-pow(1/3,2)

# функция фи, полученная заменой уравнения f(x) = 0 эквивалентным x = fi(x)

def fi(x):

return math.asin(pow((((1.4+1)/2)\*math.sin(gtr(x))\*math.sin(gtr(15))/(math.cos(gtr(x-15))))+pow(1/3,2),0.5))

def fi\_der(x):

return 0.25\*math.sin(gtr(30))/(pow(math.cos(gtr(x-15)),2)\*pow(-math.sin(gtr(15))\*math.sin(gtr(x))+0.740740740740741\*math.cos(gtr(x-15))/(math.cos(gtr(x-15))),0.5)\*pow(math.sin(gtr(15))\*math.sin(gtr(x))+0.0925925925925926\*math.cos(gtr(x-15))/(math.cos(gtr(x-15))),0.5))

def get\_derivative(fun, epsilon=0.0001):

return lambda x: (fun(x + epsilon) - fun(x)) / epsilon

#Отрисовка графика f(x)

fv = np.vectorize(f)

t1 = np.arange(0, 90, 0.0001)

plt.plot(t1, fv(t1), label="f(x)")

plt.legend()

plt.grid()

plt.show()

epsilon = 0.001

newton\_roots = [newton(f, get\_derivative(f, epsilon), 15.0, epsilon), newton(f, get\_derivative(f, epsilon), 58.0, epsilon)]

print("Newton roots")

for info in newton\_roots:

print(tabulate(info, headers=['iteration', 'x', 'f(x)']))

kl = info[6]

ks = float(kl[1])

print("\nx = %.12f°" % ks)

p1=101325

T1=288

M1=3

R=286.7

ro1=p1/(R\*T1)

print("\nro\_1 = %.6f кг/м^3" % ro1)

u1=1-(math.sin(gtr(ks-15))\*math.cos(gtr(ks)))/(math.sin(gtr(ks))\*math.cos(gtr(ks-15)))

print("u\_1 = %.6f" % u1)

v1=M1\*pow(1.4\*R\*T1,0.5)

print("v\_1 = %.6f" % v1)

v2n=(v1\*math.sin(gtr(ks))\*math.tan(gtr(ks-15)))/math.tan(gtr(ks))

print("v\_2n = %.6f" % v2n)

v2=v2n/math.sin(gtr(ks-15))

print("v\_2 = %.6f" % v2)

ro2=ro1/(1-u1)

print("\nro\_2 = %.6f кг/м^3" % ro2)

p2=p1+ro1\*u1\*pow(v1\*math.sin(gtr(ks)),2)

print("p\_2 = %.6f Па" % p2)

T2=p2/(ro2\*R)

print("T\_2 = %.6f K" % T2)

M2=v2/pow(1.4\*R\*T2,0.5)

print("M\_2 = %.6f" % M2)

end = time.time() - start

print(end)

**Результаты**

fallfire13@DESKTOP-M7F3IHA:~/Matmod$ python3 halfdiv.py

0 15.00000 - + 60.00000 37.50000 +

1 15.00000 - + 37.50000 26.25000 -

2 26.25000 - + 37.50000 31.87500 -

3 31.87500 - + 37.50000 34.68750 +

4 31.87500 - + 34.68750 33.28125 +

5 31.87500 - + 33.28125 32.57812 +

6 31.87500 - + 32.57812 32.22656 -

7 32.22656 - + 32.57812 32.40234 +

8 32.22656 - + 32.40234 32.31445 +

9 32.22656 - + 32.31445 32.27051 +

10 32.22656 - + 32.27051 32.24854 +

11 32.22656 - + 32.24854 32.23755 -

12 32.23755 - + 32.24854 32.24304 +

13 32.23755 - + 32.24304 32.24030 -

14 32.24030 - + 32.24304 32.24167 +

15 32.24030 - + 32.24167 32.24098 +

16 32.24030 - + 32.24098 32.24064 +

17 32.24030 - + 32.24064 32.24047 +

18 32.24030 - + 32.24047 32.24038 -

19 32.24038 - + 32.24047 32.24042 +

20 32.24038 - + 32.24042 32.24040 +

21 32.24038 - + 32.24040 32.24039 -

22 32.24039 - + 32.24040 32.24040 -

23 32.24040 - + 32.24040 32.24040 -

24 32.24040 - + 32.24040 32.24040 +

25 32.24040 - + 32.24040 32.24040 +

x = 32.240400°

ro\_1 = 1.227147 кг/м^3

u\_1 = 0.507983

v\_1 = 1019.988706

v\_2n = 267.724451

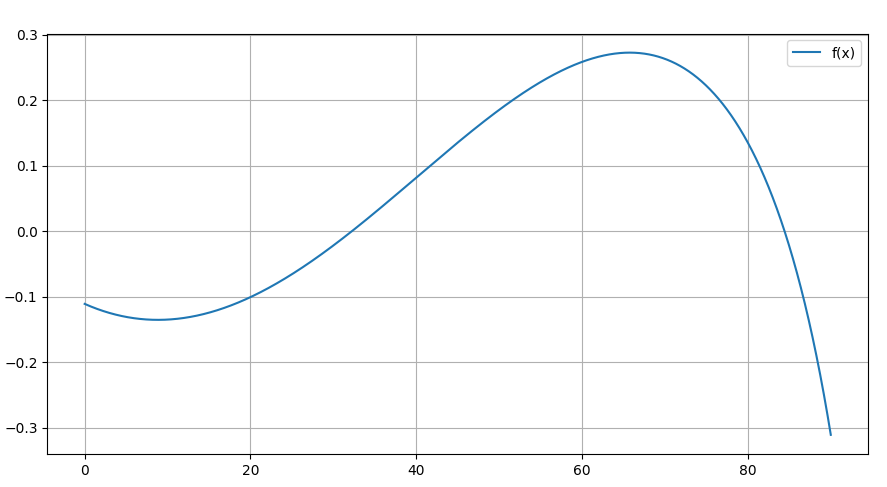
v\_2 = 903.310075

ro\_2 = 2.494113 кг/м^3

p\_2 = 285894.803235 Па

T\_2 = 399.818164 K

M\_2 = 2.254902

3.0440194606781006

fallfire13@DESKTOP-M7F3IHA:~/Matmod$ python3 newton1.py

Newton roots for x^2-7x

iteration x f(x)

----------- ------- ------------

0 6 -6

1 7.19976 1.43822

2 7.00542 0.0379628

3 7 3.47265e-05

4 7 4.98482e-09

iteration x f(x)

----------- ------- -----------

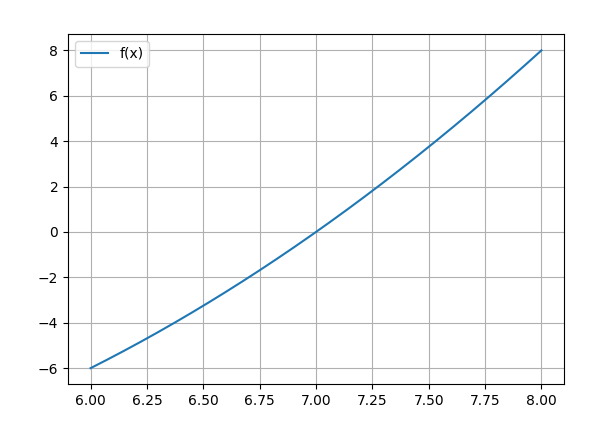
0 8 8

1 7.11121 0.790837

2 7.00173 0.0120959

3 7 4.709e-06

4 7 6.73069e-10



fallfire13@DESKTOP-M7F3IHA:~/Matmod$ python3 newton2.py

Newton roots

iteration x f(x)

----------- ------- ------------

0 15 -0.124509

1 50.6664 0.191453

2 29.8109 -0.0236733

3 32.3146 0.000743673

4 32.2405 5.55188e-07

5 32.2404 5.86176e-12

iteration x f(x)

----------- ------- ------------

0 58 0.247935

1 15.9593 -0.120919

2 46.2604 0.148376

3 31.8139 -0.00425093

4 32.2423 1.89485e-05

5 32.2404 5.48232e-10

6 32.2404 5.38458e-15

x = 32.240400182745°

ro\_1 = 1.227147 кг/м^3

u\_1 = 0.507983

v\_1 = 1019.988706

v\_2n = 267.724448

v\_2 = 903.310076

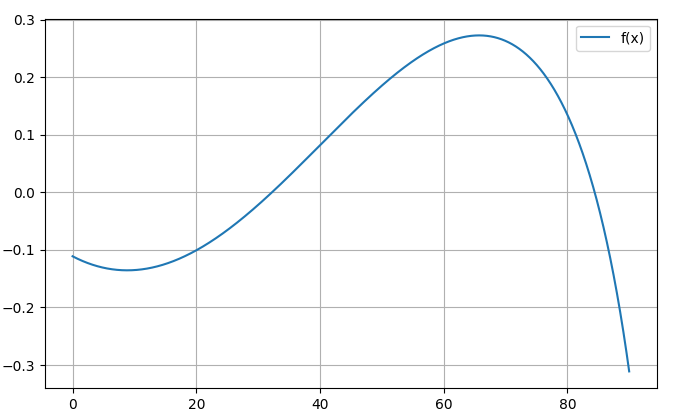
ro\_2 = 2.494113 кг/м^3

p\_2 = 285894.802203 Па

T\_2 = 399.818161 K

M\_2 = 2.254902

3.4607341289520264

****

**Сравнение результатов**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Число итераций | Значение угла |  |  |  |  | Время выполнения (time) |
| МПД | 25 | 32.240400351584 | 285894.803235 | 399.818164 | 2.494113 | 2.254902 | 3.0440194606781006 |
| МН | 6 | 32.240400182745 | 285894.802203 | 399.818161 | 2.494113 | 2.254902 | 3.4607341289520264 |

**Выводы**

За скачком уплотнения: увеличиваются плотность потока и температура, возрастает давление и снижается число Маха. Вторая точка (в районе ) в расчёт не берётся, т.к. при таком угле скачок будет неустойчивым.