МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

**«Вятский государственный университет»**

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

ЧИСЛЕННЫЕ МЕТОДЫ РЕШЕНИЯ НЕЛИНЕЙНЫХ УРАВНЕНИЙ

Отчет по лабораторной работе №1 дисциплины

«Вычислительная математика»

Вариант 17

Выполнил студент группы ИВТ-22\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Родыгин И. А./

Проверил преподаватель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Архангельский В. В./

Киров 2016

**1 Задание**

1. Построить график функции f(x) и отделить один из корней уравнения: f(x).

2. Сузить интервал изоляции корня, если необходимо, проверив условие: M<=2m.

3. Уточнить корень с погрешностью e<=0,00001 двумя численными методами: комбинированным методом и методом итераций.

4. Проверить полученное значение корня, используя систему Mathcad.

Вариант 17

Уравнение: 2x + ln(x)+0.5

Интервал: [0,0 ; 0,9]

**2 Ход выполнения лабораторной работы**

**2.1 Теоретические сведения**

**2.1.1 Теоретические сведения по уточнению корня комбинированным методом**

Методы хорд и касательных дают приближения корня с разных сторон. Поэтому их часто применяют в сочетании друг с другом, тогда уточнение корня происходит быстрее.

Пусть дано уравнение *f(x) = 0*, корень отделен на отрезке *[a, b]*.

Рассмотрим случай, когда *f ‘(x) f ’’(x)>0*.

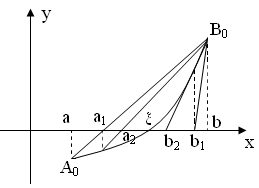


Рисунок 1

В этом случае метод хорд дает приближенное значение корня с недостатком (конец *b* неподвижен), а метод касательных – с избытком (за начальное приближение берем точку *b*).

Тогда вычисления следует проводить по формулам:

http://dit.isuct.ru/ivt/sitanov/Literatura/M501/Pages/Glava2_5.files/image003.gif

http://dit.isuct.ru/ivt/sitanov/Literatura/M501/Pages/Glava2_5.files/image005.gif

Теперь корень *ξ* заключен в интервале *[a1, b1]*. Применяя к этому отрезку комбинированный метод, получим:

http://dit.isuct.ru/ivt/sitanov/Literatura/M501/Pages/Glava2_5.files/image007.gif

http://dit.isuct.ru/ivt/sitanov/Literatura/M501/Pages/Glava2_5.files/image009.gif

и т.д.

|  |  |
| --- | --- |
| http://dit.isuct.ru/ivt/sitanov/Literatura/M501/Pages/Glava2_5.files/image011.gif  http://dit.isuct.ru/ivt/sitanov/Literatura/M501/Pages/Glava2_5.files/image013.gif |  |

Если же *f ‘(x) f ’’(x)<0* (рис. 2.14), то, рассуждая аналогично, получим следующие формулы для уточнения корня уравнения:

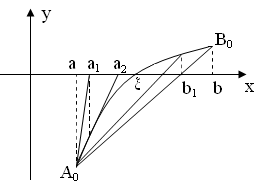


Рисунок 2

http://dit.isuct.ru/ivt/sitanov/Literatura/M501/Pages/Glava2_5.files/image015.gif

http://dit.isuct.ru/ivt/sitanov/Literatura/M501/Pages/Glava2_5.files/image017.gif

Вычислительный процесс прекращается, как только выполнится условие:

http://dit.isuct.ru/ivt/sitanov/Literatura/M501/Pages/Glava2_5.files/image019.gif

**2.1.2** **Теоретические сведения по уточнению корня методом простых итераций**

Представим исходное уравнение http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image003.gif в виде http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image290.gif.

Пусть нам известно начальное приближение к корню http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image168.gif (http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image079.gif). Подставив его в правую часть уравнения (2.21) получим новое приближение http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image294.gif, затем аналогичным образом получим http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image296.gif и так далее,

http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image298.gif,    http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image300.gif.

Оказывается, что при определенных свойствах функции http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image302.gif последовательность http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image304.gif, определяемая по формуле (2.22), сходится к корню уравнения http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image003.gif. Необходимо установить при каких условиях итерационный процесс будет сходящимся.

В начале рассмотрим графически процесс получения приближений в методе простых итераций. При решении уравнения необходимо отыскать точку пересечения кривой http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image307.gif и прямой http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image309.gif. На рисунке изображена некоторая кривая http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image307.gif, которая может представлять собой любую функцию, но сейчас для нас важно то обстоятельство, что производная этой функции в окрестности корня положительна и меньше 1. Пусть http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image311.gif  –  корень уравнения, который, естественно, предполагается неизвестным. Выберем начальное приближение в точке http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image168.gif. Следующее приближении будет равно http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image315.gif. Для того, чтобы отобразить http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image081.gif на графике можно провести через точку http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image270.gif прямую, параллельную оси *OX*, до пересечения с прямой http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image309.gif, а затем в точке пересечения этих прямых опустить перпендикуляр на ось *OX*, который и отметит положение точки http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image081.gif. Аналогично получаются все  последующие приближения. Из рисунка видно, что они сходятся к корню. Напомним, что для рассмотрения мы взяли функцию производная которой   
положительна и меньше 1.

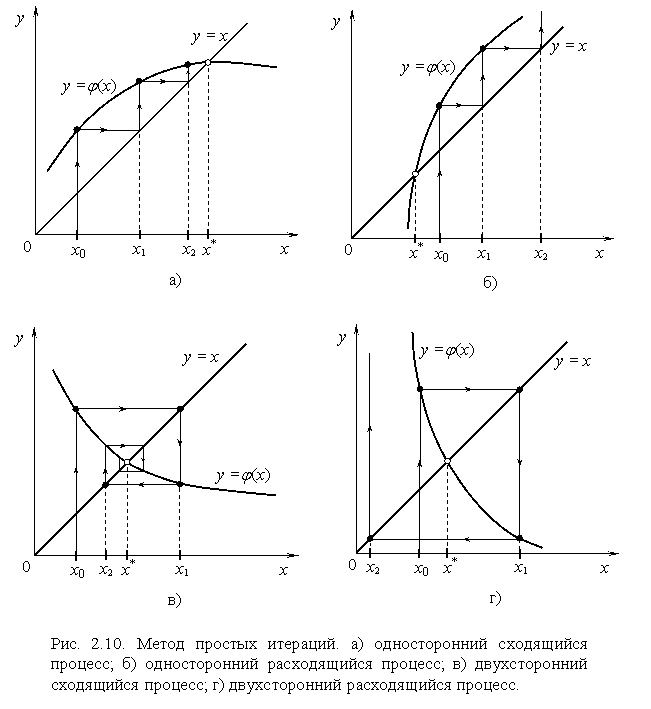


Рисунок 3 – а) Односторонний сходящийся процесс; б) односторонний расходящийся процесс; в) двухсторонний сходящийся процесс; г) двухсторонний расходящийся процесс

Рассмотрим теперь другую функцию http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image307.gif, производная которой отрицательна, но меньше 1 по абсолютному значению. Последовательные приближения также сходятся к корню, но на этот раз каждое последующее приближение находится с противоположной стороны от корня. В то время как в первом случае все последовательные приближения находились с одной стороны от корня.

Наконец, рассмотрим случай, когда производная функции http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image307.gif больше 1 и меньше -1. В обоих случаях каждое последующее приближение отстоит дальше от корня, т.е.  итерационный процесс расходится. Из сказанного выше можно предположить, что итерационный процесс, определяемый формулой сходится при условии, что производная http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image319.gif меньше 1 по абсолютной величине.

Математически условие сходимости можно установить следующим образом. Представим *k*-е и (*k*+1)-е приближения в форме

http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image321.gif,  http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image323.gif,

где http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image325.gif и http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image327.gif – отклонения приближений от корня.

Функцию http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image302.gif вблизи точки http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image075.gif приближенно заменим первыми двумя членами ряда Тейлора. Тогда итерационная формула примет вид

http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image331.gif,

но поскольку http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image075.gif является корнем уравнения, то первые слагаемые в правой и левой частях этого выражения тождественно равны и, следовательно

http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image334.gif.

Для сходимости итерационного процесса необходимо, чтобы погрешность на каждом шаге убывала

http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image336.gif,

откуда следует, что в окрестности корня должно выполняться условие

http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image338.gif.

Таким образом, для того чтобы итерационный процесс был сходящимся, необходимо, чтобы абсолютная величина производной http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image340.gif в окрестности корня была меньше единицы. Если это условие выполняется на отрезке http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image206.gif на котором локализован корень, то в качестве начального приближения можно взять любую точку из этого отрезка http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image079.gif. Скорость сходимости зависит от абсолютной величины производной http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image344.gif:  чем меньше http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image344.gif вблизи корня, тем быстрее сходится процесс.

**2.1.3 Преобразование уравнения к итерационному виду.** Переход от уравнения к уравнению в итерационной форме можно осуществить различными способами в зависимости от вида функции http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image005.gif. При таком переходе необходимо построить функцию http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image302.gif так, чтобы выполнялось условие сходимости (2.23).

Теперь рассмотрим один из общих алгоритмов перехода от уравнения http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image003.gif к уравнению http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image372.gif. Умножим левую и правую части уравнения http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image003.gif на произвольную константу http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image375.gif и добавим к обеим частям неизвестное *x*. При этом корни исходного уравнения не изменятся

http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image377.gif

или

http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image379.gif.

Уравнение эквивалентно уравнению с функцией http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image381.gif. Произвольный выбор константы **  позволяет обеспечить выполнение условия сходимости. Поскольку в данном случае http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image383.gif, значение **  следует выбирать, так чтобы в окрестности корня выполнялось условие

http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image385.gif.

Желательно выбрать величину **  такой, чтобы http://physics.herzen.spb.ru/library/01/01/nm_labs/nonlineareq.files/image387.gif, тогда сходимость будет двухсторонней (рис. 2.11, в).

**2.2 Необходимые расчёты**

Исходное уравнение 2x + ln(x) + 0.5 = 0

Исходная функция f(x) = 2x + ln(x) + 0.5

Первая производная исходной функции f `(x) =

Вторая производная исходной функции f ``(x) =

Преобразование исходного уравнения к каноничному виду φ(x) = x:

1. 2x + ln(x) + 0.5 = 0
2. k < 2 / max(|f’(x)|, |f’(y)|) => k = 0.1
3. k \* (2x + ln(x) + 0.5) = k \* 0
4. φ(x) = x + k \* (2x + ln(x) + 0.5)
5. φ(x) = x - 0.1 \* (2x + ln(x) + 0.5)

Первая производная от φ(x): φ’(x) = 1 - 0.1 \* ( )

**2.3 Скриншоты выполнения программы**

Листинг программы представлен в приложении А.

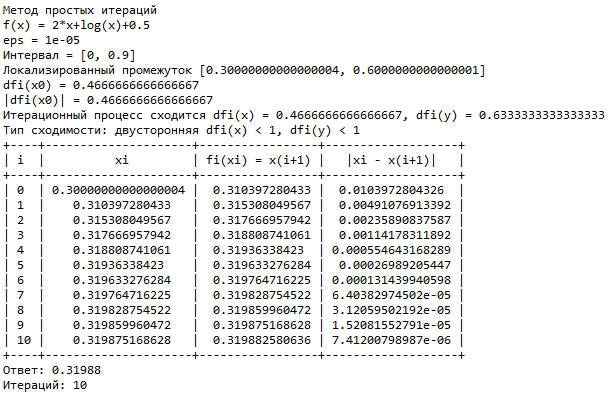


Рисунок 4 – скриншот выполнения программы, реализующей уточнение корней способом простых итераций

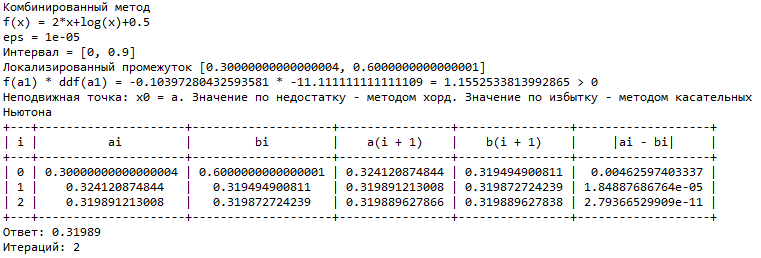


Рисунок 5 – скриншот выполнения программы, реализующей уточнение корней комбинированным методом

2.4 Результат проверки выполнения программ

Для точного построения графика была использована библиотека языка Python под названием matplotlib. Изображение графика показано на рисунке 6.

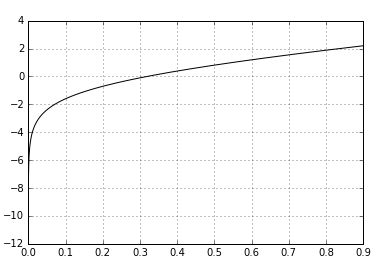


Рисунок 6 – изображение графика f(x) = 2x + ln(x) + 0.5

Проверка корня уравнения 2x + ln(x) + 0.5 = 0 была проведена в математической системе Wolfram Alpha. Результаты проверки представлены на рисунке 7.



Рисунок 7 – приближённые корни уравнения f(x) = 2x + ln(x) + 0.5

Приложение А

Исходный код программы

import matplotlib.pyplot as plt

from numpy import log, arange

from prettytable import PrettyTable

a, b = 0.1, 0.9

eps = 0.00001

def f(x):

return 2\*x + log(x) + 0.5

def df(x):

return 1/x + 2

def ddf(x):

return -1/(x\*x)

def fi(x):

return x - 0.1 \* f(x)

def dfi(x):

return 1 - 0.1 \* df(x)

def staticPoint(x):

if (f(x) \* ddf(x) > 0):

return True

else:

return False

def minAB(a, b):

if (abs(df(a)) > abs(df(b))):

return abs(df(b))

else:

return abs(df(a))

def showPlot(x, y):

x = arange(x, y, eps) #последовательность

plt.plot(x, f(x), 'k') #инициализация графика

#plt.ylim(ymin=0) #ограничение по y

plt.grid(True) #включение сетки

plt.show() #показать график

def outText(startText, x, y, eps):

print(startText)

print('f(x) = 2\*x+log(x)+0.5')

print('eps = {}'.format(eps))

print('Интервал = [{}, {}]'.format(x, y))

def iteration(x, y):

x1 = x

i = 0

x0 = x1

err = False

if ((f(x) \* f(y)) > 0):

print('Одинаковые знаки на концах графика')

err = True

else:

print('dfi(x0) = {}'.format(dfi(x0)))

print('|dfi(x0)| = {}'.format(abs(dfi(x0))))

if (abs(dfi(x0)) < 1):

print('Итерационный процесс сходится dfi(x) = {}, dfi(y) = {}'.format(dfi(x), dfi(y)))

if (dfi(x) < 0) and (dfi(y) < 0):

q = 1

print('Тип сходимости: двусторонняя')

else:

q = max(abs(dfi(x)), abs(dfi(y)))

print('Тип сходимости: односторонняя')

else:

print('Итерационный процесс расходится {}, {}'.format(dfi(x), dfi(y)))

err = True

if err:

print('Нет корней на данном интервале, либо он не единственный')

else:

table = PrettyTable(['i', 'xi', 'fi(xi) = x(i+1)', '|xi - x(i+1)|'])

while True:

x0 = x1

x1 = fi(x0)

table.add\_row([i, x0, fi(x0), abs(x1 - x0)])

i += 1

if (x0 < x) or (x0 > y):

err = True

break

if (abs(x1 - x0) <= q \* eps):

break

if err:

print(table)

print('Нет ответа на промежутке')

else:

print(table)

print('Ответ: {:.7}'.format(x1))

print('Итераций: {}'.format(i - 1))

def combine(x, y, eps):

while True:

if (f(x + 0.1) \* f(y - 0.1)) > 0:

break

else:

x += 0.1

y -= 0.1

print('Локализированный промежуток [{}, {}]'.format(x, y))

err = False

i = 0

if (f(x) \* f(y) > 0):

err = True

if err:

print('Нет корней на данном интервале, либо он не единственный')

else:

a1 = x

b1 = y

if staticPoint(a1):

print('f(a1) \* ddf(a1) = {} \* {} = {} > 0'.format(f(a1), ddf(a1), f(a1)\*ddf(a1)))

print('Неподвижная точка: х0 = a. Значение по недостатку - методом хорд. Значение по избытку - методом касательных Ньютона')

elif staticPoint(b1):

print('f(b1) \* ddf(b1) = {} \* {} = {} > 0'.format(f(b1), ddf(b1), f(b1)\*ddf(b1)))

print('Неподвижная точка: х0 = b. Значение по недостатку - методом касательных Ньютона. Значение по избытку - методом хорд')

else:

print('Ошибка!')

table = PrettyTable(['i', 'ai', 'bi', 'a(i + 1)', 'b(i + 1)', '|ai - bi|'])

while True:

a0 = a1

b0 = b1

if staticPoint(y):

a1 = a0 - (f(a0) \* ((b0 - a0) / (f(b0) - f(a0))))

b1 = b0 - (f(b0) / df(b0))

static = a1

elif staticPoint(x):

a1 = b0 - (f(b0) \* ((a0 - b0) / (f(a0) - f(b0))))

b1 = a0 - (f(a0) / df(a0))

static = b1

else:

print('Ошибка!')

err = True

break

table.add\_row([i, a0, b0, a1, b1, abs(a1 - b1)])

if (abs(f(static) / minAB(x, y)) <= eps):

break

i += 1

if err:

print('Нет корней на данном интервале, либо он не единственный')

else:

print(table)

print('Ответ: {:.7}'.format(b1))

print('Итераций: {}'.format(i))

def main(x, y, eps):

outText('Метод простых итераций', x, y, eps)

iteration(x, y)

print('\n\n\n\n\n')

outText('Комбинированный метод', x, y, eps)

combine(x, y, eps)

if \_\_name\_\_ == '\_\_main\_\_':

showPlot(a, b)

main(a, b, eps)