**1. Постановка задачи**

Решить уравнение  согласно варианту, используя инструментарий Excel и методы приближенного численного решения нелинейных уравнений: метод хорд, метод касательных и метод половинного деления. Приближенные численные методы решения должны быть реализованы в виде программы на языке Python.

Для отделения одного из корней уравнения  необходимо определить отрезок , которому принадлежит искомый корень. Для этого необходимо:

1. Построить в Excel таблицу функции , задача отрезок  сначала произвольным образом. При этом шаг для формирования множества точек  выбирается произвольно.
2. Построить график по полученной таблице.
3. Подобрать отрезок  так, чтобы внутри него функция  меняла знак с минуса на плюс или с плюса на минус.

Найти приближенное решение уравнения , применяя инструмента Excel «Анализ «чтоб если», используя в качестве стартовой точки для реализации итерационного процесса одну из границ подобранного отрезка , например, задав  или .

Найти приближенное решение уравнения , применяя инструмента Excel «Поиск решения».

Изучить численные методы решения нелинейных уравнений: метод хорд, метод касательных и метод половинного деления, используя рекомендуемую литературы. Представить в отчете краткое описание этих методов.

Разработать оригинальный алгоритм и программу на языке Python для решения задачи:  при  следующими методами:

1. методом половинного деления;
2. методом хорд;
3. методом касательных.

Решение найти с точностью 0,001. Программа должна содержать соответствующие подпрограммы, которые реализуют указанные методы.

Выполнить проверку работу алгоритмов с помощью пакета Excel.

Сравнить полученные результаты решения уравнения  при использовании перечисленного инструментария. Выполнить анализ полученных результатов.

**2. Решение уравнения**

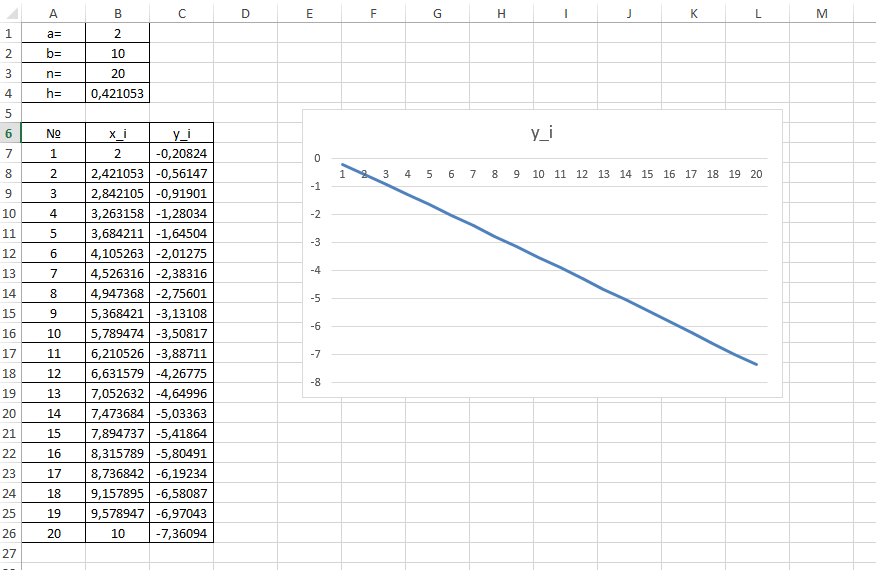
**2.1. Метод отделения корней**

Рассмотрим нелинейное уравнение:

ln(x+4) = x

Поставим задачу найти отрезок , содержащий хотя бы один из корней уравнения.

Зададим , ,  и используя Excel, построим таблицу функции (рис.2.1).



**Рис. 2.1. Первоначальная таблица и график функции**

Из рис. 2.1 видно, что обе границы выбранного отрезка  необходимо сместить влево, а сам отрезок уменьшить. Перестроим таблицу функции, задав a = -2, b = 5 (рис. 2.2).

Из рис. 2.2 видно, что на отрезке [-2,5] функция y = F(x) = ln(x+4) - x пересекает ось абсцисс, а, следовательно, этот отрезок можно использовать для поиска корня уравнения.

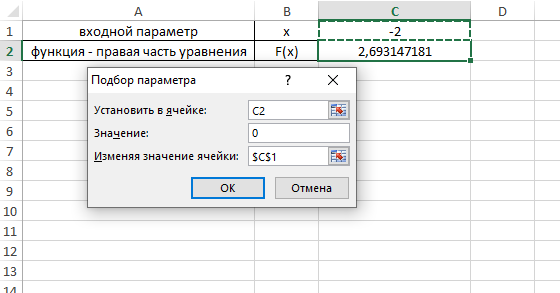


**Рис. 2.2. Таблица и график функции на отрезке [-2, 5]**

**2.2. Применение инструмента Excel «Анализ «что, если» для решения нелинейных уравнений**

Используя инструмента «Анализ «что, если» найдем решение задачи на отрезке [-2 , 5].

Заполняем вычислительную область (рис. 2.3).



**Рис. 2.3. Решение задачи**

В ячейке С1 указываем начальное значение для входного параметра  – одну из границ отрезка. В ячейке С2 задаем формулу для вычисления правой части уравнения:

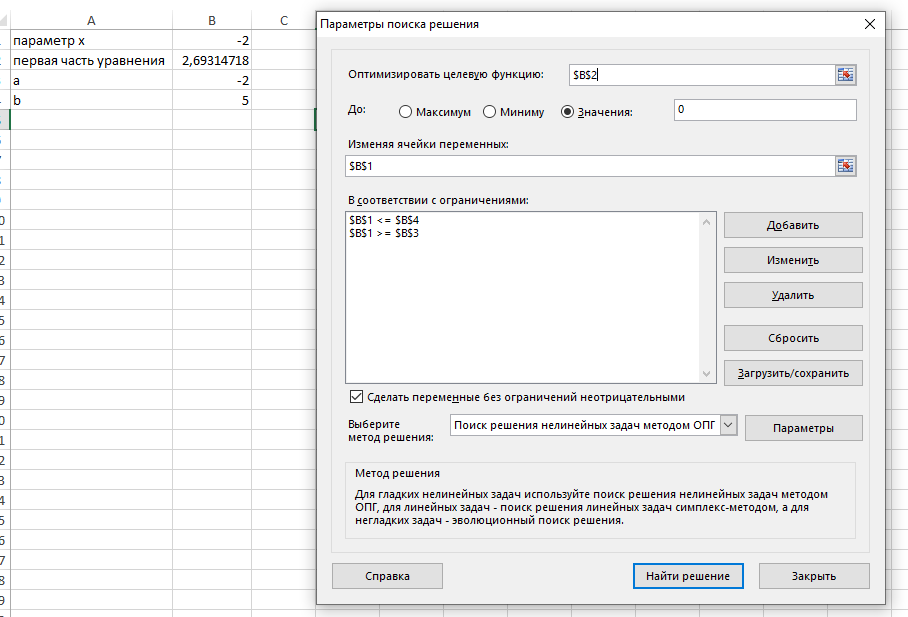
=LN(C1+4)-C1

Далее, заполняем окно «Подбор решения» и нажимаем кнопку «ok». В результате получаем в ячейке С1 решение х ≈ 1,748271378. При этом в ячейке С2 указана абсолютная погрешность вычисления: . В данном случае погрешность приближенно равна 0,000627802.

**2.3. Применение инструмента** **Excel «Поиск решения»   
для решения нелинейных уравнений**

Используя инструмент «Поиск решения», решим задачу на отрезке [-2,5].

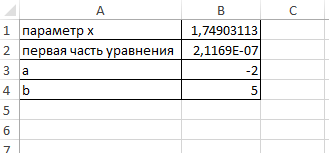
Формируем вычислительную область на листе Excel (рис. 2.4).



**Рис. 2.4. Вычислительная область для решения задачи**

**и заполнение окна «Поиск решения»**

В результате получаем решение (рис. 2.5).



**Рис. 2.5. Решение задачи с использованием   
инструмента «Поиск решения»**

Решением задачи является значение х ≈ 1,74903113, при этом погрешность решения Δ = 2,117×.

**2.4. Разработка алгоритма для решения задачи ln(x+4)=x**

***Код консольного приложения «Программа для практики»:***

import math  
  
def tangent\_method ():  
 *'''  
 Реализация метода касательных  
 Выводит в консоль табличку со значениями  
 '''* a = -2 # начало отрезка  
 b = 5 # Конец отрезка  
 F\_a = (math.log(a + 4)) - a # Функция от a  
 F\_b = (math.log(b + 4)) - b # Функция от b  
 F1\_a = (1 / (a + 4)) - 1 # Производная F(a) (F(a)')  
 F1\_b = (1 / (b + 4)) - 1 # Производная F(b) (F(b)')  
 x\_k = a # Искомый х  
 Fx\_k = (math.log(x\_k + 4)) - x\_k # Функция от искомого х  
 F1x\_k = (1 / (x\_k + 4)) - 1 # Производная F(x) (F(x)')  
 print("-------------------------------------------------------------------------------------")  
 print("|", "{:^25}".format("x\_k"), "|", "{:^25}".format("F(x\_k)"), "|", "{:^25}".format("F(x\_k)'"), "|")  
 print("-------------------------------------------------------------------------------------")  
 print("|", "{:^25}".format(x\_k), "|", "{:^25}".format(Fx\_k), "|", "{:^25}".format(F1x\_k), "|")  
 while ((abs(Fx\_k)) > 0.001):  
 x\_k = x\_k - (Fx\_k / F1x\_k)  
 Fx\_k = (math.log(x\_k + 4)) - x\_k  
 F1x\_k = (1 / (x\_k + 4)) - 1  
 print("|", "{:^25}".format(x\_k), "|", "{:^25}".format(Fx\_k), "|", "{:^25}".format(F1x\_k), "|")  
  
 print("-------------------------------------------------------------------------------------")  
 print("\n\nx ≈ ", x\_k, "\nПогрешность = ", abs(Fx\_k))  
  
def chord\_method():  
 *'''  
 Реализация метода хорд  
 Выводит в консоль табличку со значениями  
 '''* a = -2  
 b = 5  
 F\_a = (math.log(a + 4)) - a  
 F\_b = (math.log(b + 4)) - b  
 F2\_a = -1 / ((a + 4) \* (a + 4)) # Двойная производная F(a) (F(a)'')  
 F2\_b = -1 / ((b + 4) \* (b + 4)) # Двойная производная F(b) (F(b)'')  
 if (a\*F2\_a):  
 x\_k = b  
 elif (b\*F2\_b):  
 x\_k = a  
 Fx\_k = math.log(x\_k + 4) - x\_k  
 print("---------------------------------------------------------")  
 print("|", "{:^25}".format("x\_k"), "|", "{:^25}".format("F(x\_k)"), "|")  
 print("---------------------------------------------------------")  
 print("|", "{:^25}".format(x\_k), "|", "{:^25}".format(Fx\_k), "|")  
 while ((abs(Fx\_k)) > 0.001):  
 x\_k = a - (((F\_a) / (Fx\_k - F\_a)) \* (x\_k - a))  
 Fx\_k = math.log(x\_k + 4) - x\_k  
 print("|", "{:^25}".format(x\_k), "|", "{:^25}".format(Fx\_k), "|")  
  
 print("---------------------------------------------------------")  
 print("\n\nx ≈ ", x\_k, "\nПогрешность = ", abs(Fx\_k))  
  
def method\_of\_half\_division():  
 *'''  
 Реализация метода половинного деления  
 Выводит в консоль табличку со значениями  
 '''* a = -2  
 b = 5  
 F\_a = (math.log(a + 4)) - a  
 F\_b = (math.log(b + 4)) - b  
 x\_k = (a + b) / 2  
 Fx\_k = (math.log(x\_k + 4)) - x\_k  
 print(  
 "-------------------------------------------------------------------------------------------------------------------------------------------------------------------------")  
 print("|", "{:^25}".format("a"), "|", "{:^25}".format("b"), "|", "{:^25}".format("F(a)"), "|",  
 "{:^25}".format("F(b)"), "|", "{:^25}".format("x\_k"), "|", "{:^25}".format("F(x\_k)"), "|")  
 print(  
 "-------------------------------------------------------------------------------------------------------------------------------------------------------------------------")  
 print("|", "{:^25}".format(a), "|", "{:^25}".format(b), "|", "{:^25}".format(F\_a), "|", "{:^25}".format(F\_b), "|",  
 "{:^25}".format(x\_k), "|", "{:^25}".format(Fx\_k), "|")  
 while ((abs(Fx\_k)) > 0.001):  
 x = x\_k  
 F\_x = (math.log(x + 4)) - x  
 if (F\_x \* F\_b < 0):  
 a = x  
 b = b  
 F\_a = (math.log(a + 4)) - a  
 F\_b = (math.log(b + 4)) - b  
 x\_k = (a + b) / 2  
 Fx\_k = (math.log(x\_k + 4)) - x\_k  
 print("|", "{:^25}".format(a), "|", "{:^25}".format(b), "|", "{:^25}".format(F\_a), "|",  
 "{:^25}".format(F\_b), "|", "{:^25}".format(x\_k), "|", "{:^25}".format(Fx\_k), "|")  
 elif (F\_a \* F\_x):  
 a = a  
 b = x  
 F\_a = (math.log(a + 4)) - a  
 F\_b = (math.log(b + 4)) - b  
 x\_k = (a + b) / 2  
 Fx\_k = (math.log(x\_k + 4)) - x\_k  
 print("|", "{:^25}".format(a), "|", "{:^25}".format(b), "|", "{:^25}".format(F\_a), "|",  
 "{:^25}".format(F\_b), "|", "{:^25}".format(x\_k), "|", "{:^25}".format(Fx\_k), "|")  
  
 print(  
 "-------------------------------------------------------------------------------------------------------------------------------------------------------------------------")  
 print("\n\nx ≈ ", x\_k, "\nПогрешность = ", abs(Fx\_k))  
  
i = 0  
print("Добро пожаловать в программу для решения уравнения: \"ln(x + 4) = x\" \n"  
 "Для решения уравнения доступно 3 метода:\n"  
 "1. Метод половинного деления\n"  
 "2. Метод хорд\n"  
 "3. Метод касательных\n"  
 "Выберите один из них")  
choice = int(input())  
if choice == 1:  
 method\_of\_half\_division()  
elif choice == 2:  
 chord\_method()  
elif choice == 3:  
 tangent\_method()  
else:  
 print("Unknown")  
  
while (i == 0):  
 print("Хотите продолжить?\n"  
 "1. Да\n"  
 "2. Закрыть программу")  
 n = int(input())  
 if n == 1:  
 print("Выберите метод для решения задачи:\n"  
 "1. Метод половинного деления\n"  
 "2. Метод хорд\n"  
 "3. Метод касательных")  
 choice = int(input())  
 if choice == 1:  
 method\_of\_half\_division()  
 elif choice == 2:  
 chord\_method()  
 elif choice == 3:  
 tangent\_method()  
 else:  
 print("Unknown")  
 elif n == 2:  
 i = 1  
 print ("Завершение работы...")  
 else:  
 i = 1  
 print ("Unknown")  
 print ("Завершение работы...")

**2.5. Метод половинного деления**

Метод деления пополам — это метод поиска корней в математике, который применяется к любым непрерывным функциям, для которых известны два значения с противоположными знаками.

Он состоит из многократного деления пополам интервала, определяемого этими значениями, и последующего выбора подинтервала, в котором функция меняет знак и, следовательно, должна содержать корень.

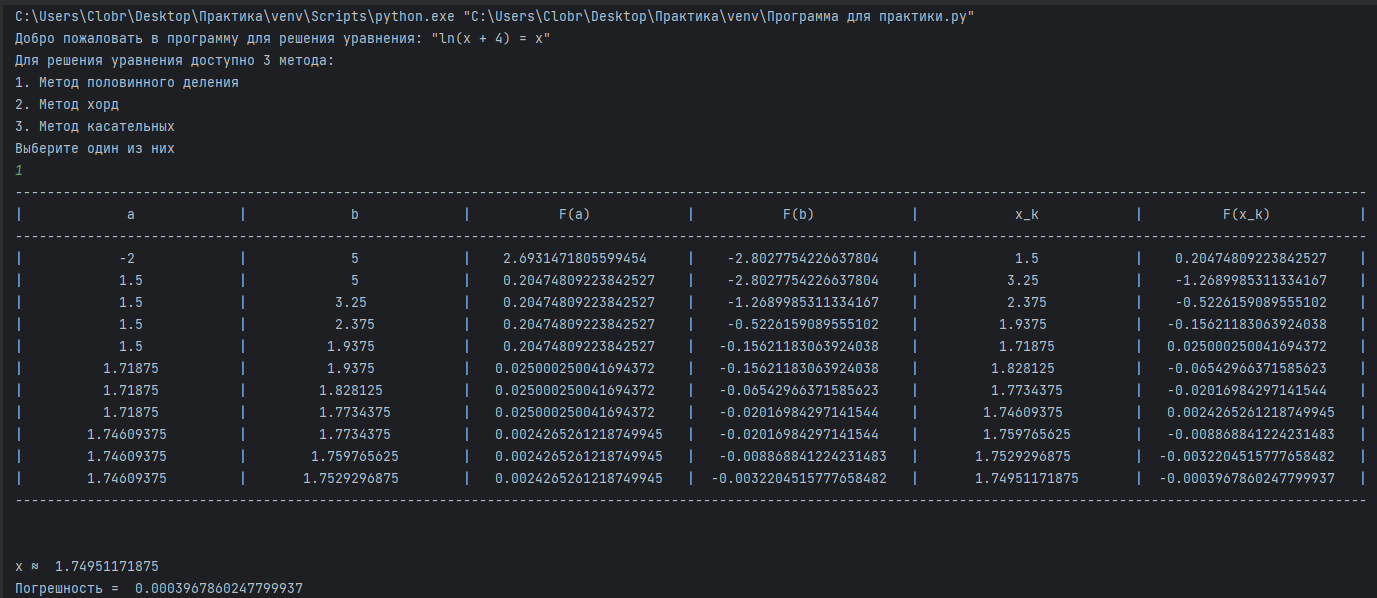
Это очень простой и надёжный метод, но он также относительно медленный. Из-за этого его часто используют для получения грубого приближения к решению, которое затем используется в качестве отправной точки для более быстро сходящихся методов.

Этот метод также называют методом деления интервала вдвое, методом двоичного поиска или методом дихотомии.

***Листинг метода половинного деления:***

a = -2  
b = 5  
F\_a = (math.log(a + 4)) - a  
F\_b = (math.log(b + 4)) - b  
x\_k = (a + b) / 2  
Fx\_k = (math.log(x\_k + 4)) - x\_k  
print(  
 "-------------------------------------------------------------------------------------------------------------------------------------------------------------------------")  
print("|", "{:^25}".format("a"), "|", "{:^25}".format("b"), "|", "{:^25}".format("F(a)"), "|",  
 "{:^25}".format("F(b)"), "|", "{:^25}".format("x\_k"), "|", "{:^25}".format("F(x\_k)"), "|")  
print(  
 "-------------------------------------------------------------------------------------------------------------------------------------------------------------------------")  
print("|", "{:^25}".format(a), "|", "{:^25}".format(b), "|", "{:^25}".format(F\_a), "|", "{:^25}".format(F\_b), "|",  
 "{:^25}".format(x\_k), "|", "{:^25}".format(Fx\_k), "|")  
while ((abs(Fx\_k)) > 0.001):  
 x = x\_k  
 F\_x = (math.log(x + 4)) - x  
 if (F\_x \* F\_b < 0):  
 a = x  
 b = b  
 F\_a = (math.log(a + 4)) - a  
 F\_b = (math.log(b + 4)) - b  
 x\_k = (a + b) / 2  
 Fx\_k = (math.log(x\_k + 4)) - x\_k  
 print("|", "{:^25}".format(a), "|", "{:^25}".format(b), "|", "{:^25}".format(F\_a), "|",  
 "{:^25}".format(F\_b), "|", "{:^25}".format(x\_k), "|", "{:^25}".format(Fx\_k), "|")  
 elif (F\_a \* F\_x):  
 a = a  
 b = x  
 F\_a = (math.log(a + 4)) - a  
 F\_b = (math.log(b + 4)) - b  
 x\_k = (a + b) / 2  
 Fx\_k = (math.log(x\_k + 4)) - x\_k  
 print("|", "{:^25}".format(a), "|", "{:^25}".format(b), "|", "{:^25}".format(F\_a), "|",  
 "{:^25}".format(F\_b), "|", "{:^25}".format(x\_k), "|", "{:^25}".format(Fx\_k), "|")  
  
print(  
 "-------------------------------------------------------------------------------------------------------------------------------------------------------------------------")  
print("\n\nx ≈ ", x\_k, "\nПогрешность = ", abs(Fx\_k))

***Тестирование:***

**Рис. 2.6. Решение уравнения методом**

**Половинного деления**

***Проверка работы алгоритма с помощью пакета Excel.***

Пусть точность вычисления корня . Выполним необходимые действия:

1. Вычисляем значение правой части уравнения в точках a = -2 и b = 5:

Необходимое условие выполняется: , поэтому корень уравнения принадлежит отрезку [-2, 5].

2. Определяем середину отрезка [-2, 5]: , и находим значение функции .

3. Проверяем условие:. Оно, очевидно, не выполняется, поэтому вычисление корня продолжаем.

4. Из двух отрезков [-2, 1.5] и [1.5, 5] выбираем второй, так как для него выполняется условие . Действительно,

и F(1.5)F(5)<0.

Теперь в качестве отрезка  рассматриваем отрезок [1.5, 5]: a = 1.5, b = 5.

Продолжаем описанные вычисления до тех пор, пока не выполнится условие . Расчеты приведены в таблице Excel (рис. 2.7), где для ячеек введены следующие формулы:

D2 → =LN(B2+4)-B2

E2 → =LN(C2+4)-C2

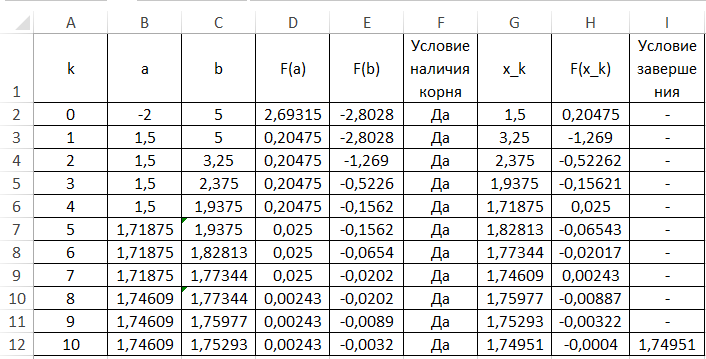
F2 → =ЕСЛИ(D2\*E2<0;"Да";"Нет")

G2 → =(B2+C2)/2

H2 → =LN(G2+4)-G2

I2 → =ЕСЛИ(ABS(H2)>0,001;"-";G2)

которые в дальнейшем были распространены для получения решения.



**Рис. 2.7. Решение уравнения методом**

**Половинного деления (Excel)**

**2.6. Метод хорд**

Метод хорд (метод секущих) — итерационный метод решения нелинейного уравнения.

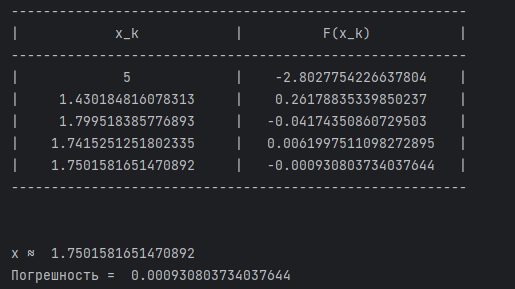
Решение уравнения заключается в многократном повторении этого алгоритма. Полученное в результате вычислений решение является приближенным, но его точность можно сделать такой, какой требуется, задав нужное значение погрешности ε.

В начале вычислений методом хорд требуется указать границы области поиска корня. В общем случае эта граница может быть произвольной.

***Листинг метода хорд:***

a = -2  
b = 5  
F\_a = (math.log(a + 4)) - a  
F\_b = (math.log(b + 4)) - b  
F2\_a = -1 / ((a + 4) \* (a + 4)) # Двойная производная F(a) (F(a)'')  
F2\_b = -1 / ((b + 4) \* (b + 4)) # Двойная производная F(b) (F(b)'')  
if (a\*F2\_a):  
 x\_k = b  
elif (b\*F2\_b):  
 x\_k = a  
Fx\_k = math.log(x\_k + 4) - x\_k  
print("---------------------------------------------------------")  
print("|", "{:^25}".format("x\_k"), "|", "{:^25}".format("F(x\_k)"), "|")  
print("---------------------------------------------------------")  
print("|", "{:^25}".format(x\_k), "|", "{:^25}".format(Fx\_k), "|")  
while ((abs(Fx\_k)) > 0.001):  
 x\_k = a - (((F\_a) / (Fx\_k - F\_a)) \* (x\_k - a))  
 Fx\_k = math.log(x\_k + 4) - x\_k  
 print("|", "{:^25}".format(x\_k), "|", "{:^25}".format(Fx\_k), "|")  
  
print("---------------------------------------------------------")  
print("\n\nx ≈ ", x\_k, "\nПогрешность = ", abs(Fx\_k))

***Тестирование:***



**Рис. 2.8. Решение уравнения методом хорд**

***Проверка работы алгоритма с помощью пакета Excel.***

Пусть точность вычисления корня . Выберем формулу по которой будем вести вычисления. Для этого оценим знак функции и ее второй производной на концах отрезка [-2, 5]:

– для точки a = -2: F(-2) ≈ 2.7 и F´´(-2) = -0.25;

– для точки b = 5: F(5) ≈ -2.8 и F´´(5) = -0.012.

Так как , , то выбираем для расчетов формулу и . Выполним необходимые действия:

1. Строим хорду, проходящую через точки (-2, F(-2)) и (5, F(5)) и находим первое приближение корня:

При этом значение функции  в точке  равно

.

2. Проверяем условие: :. Оно не выполняется.

Процесс расчета продолжаем, пока не выполнится условие . Процесс вычислений иллюстрирует рис. (2.9.) где для ячеек введены следующие формулы:

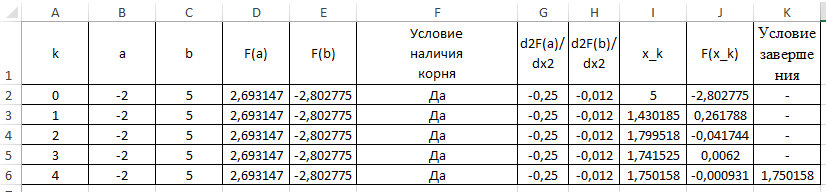
G2 → =-1/((B2+4)\*(B2+4))

H2 → =-1/((C2+4)\*(C2+4))

В ячейке I2 задана формула, которая определяет выбор стартовой точки:

=ЕСЛИ(B2\*G2>0;C2;ЕСЛИ(C2\*H2>0;B2)).

Условие проверки наличия корня в выбранном отрезке и условие завершения записаны, как и в методе половинного деления.



**Рис. 2.9. Решение уравнения методом хорд (Excel)**

**2.7. Метод касательных.**

Метод Ньютона (также известный как метод касательных) — это итерационный численный метод нахождения корня (нуля) заданной функции.

Метод был впервые предложен английским физиком, математиком и астрономом Исааком Ньютоном (1643–1727).

Поиск решения осуществляется путём построения последовательных приближений и основан на принципах простой итерации. Метод обладает квадратичной сходимостью.

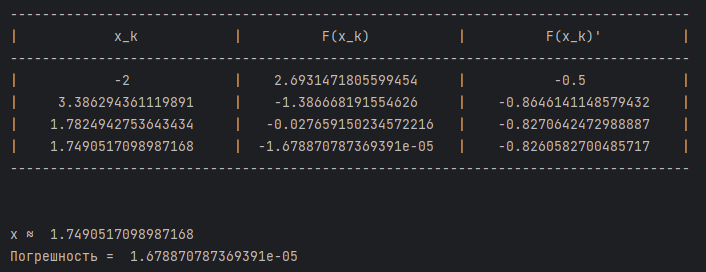
Модификацией метода является метод хорд и касательных.

Также метод Ньютона может быть использован для решения задач оптимизации, в которых требуется определить ноль первой производной либо градиента в случае многомерного пространства.

***Листинг метода касательных:***

a = -2 # начало отрезка  
b = 5 # Конец отрезка  
F\_a = (math.log(a + 4)) - a # Функция от a  
F\_b = (math.log(b + 4)) - b # Функция от b  
F1\_a = (1 / (a + 4)) - 1 # Производная F(a) (F(a)')  
F1\_b = (1 / (b + 4)) - 1 # Производная F(b) (F(b)')  
x\_k = a # Искомый х  
Fx\_k = (math.log(x\_k + 4)) - x\_k # Функция от искомого х  
F1x\_k = (1 / (x\_k + 4)) - 1 # Производная F(x) (F(x)')  
print("-------------------------------------------------------------------------------------")  
print("|", "{:^25}".format("x\_k"), "|", "{:^25}".format("F(x\_k)"), "|", "{:^25}".format("F(x\_k)'"), "|")  
print("-------------------------------------------------------------------------------------")  
print("|", "{:^25}".format(x\_k), "|", "{:^25}".format(Fx\_k), "|", "{:^25}".format(F1x\_k), "|")  
while ((abs(Fx\_k)) > 0.001):  
 x\_k = x\_k - (Fx\_k / F1x\_k)  
 Fx\_k = (math.log(x\_k + 4)) - x\_k  
 F1x\_k = (1 / (x\_k + 4)) - 1  
 print("|", "{:^25}".format(x\_k), "|", "{:^25}".format(Fx\_k), "|", "{:^25}".format(F1x\_k), "|")  
  
print("-------------------------------------------------------------------------------------")  
print("\n\nx ≈ ", x\_k, "\nПогрешность = ", abs(Fx\_k))

***Тестирование:***



**Рис. 2.10. Решение уравнения методом хорд**

***Проверка работы алгоритма с помощью пакета Excel.***

Пусть точность вычисления корня, как и ранее, . Выполним необходимые действия:

1. Убедимся, что на отрезке [-2, 5] выполняется условие:

Так как F(-2)F(5)<0, то условие наличия решения выполняется.

2. Выберем начальное значение приближенного корня. Для этого необходимо проверить условие  для точки a = -2 и точки b=5. Учитывая, что F´(x) = , получаем:

F(-2)F´(-2) = 2.7 \* -0.5 < 0;

Поэтому выбираем.

3. Применим формулу для вычисления следующего приближенного значения корня: .

При этом 

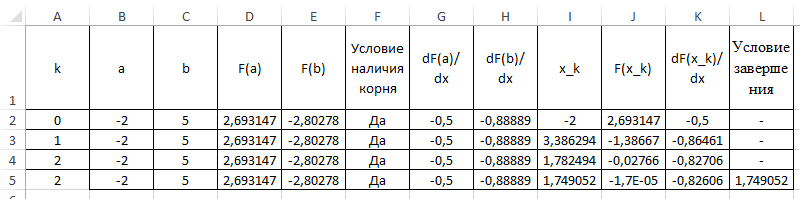
4. Как и ранее, проверяем условие завершения вычислений . Оно не выполняется, следовательно, процесс вычислений по формуле следует продолжать.

Порядок вычислений выполнен в Excel (рис. 2.11). Здесь в чеках G2 и Н2 вычислены производные функции  в точках  и . В ячейке I2 записана формула для выбора первоначального приближения корня уравнения:

=ЕСЛИ(B2\*G2>0;В2;ЕСЛИ(E2\*H2>0;С2)).

Далее, вычисление приближенного значения корня уравнения на каждой итерации выполняется по формуле.

В результате получаем приближенное значение корня уравнения: x ≈ 1.75. При этом погрешность вычислений равна 1.7E-05.



**Рис. 2.11. Решение уравнения методом хорд(Excel)**

Сравним все полученные результаты вычисления корня для задачи на отрезке [-2, 5]. Построим для удобства таблицу решений, полученных разными методами:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Метод/инструмент решения | Значение корня | Погрешность |
| 1. | Excel, «Подбор параметра» | 1,748271378 | 0,000627802 |
| 2. | Excel, «Поиск решения» | 1,74903113 | 2,1169E-07 |
| 3. | Метод половинного деления | 1,749511719 | 0,000396786 |
| 4. | Метод хорд | 1,750158165 | 0,000930804 |
| 5. | Метод касательных | 1,74905171 | -1,67887E-05 |

При использовании инструментария Excel самая высокая точность решения получена при применении надстройки «Поиск решения». Это же значение решения x = 1,74903113 самым точным из полученных, учитывая оценку погрешности. Среди численных методов решения самым точным оказался метод касательных, где погрешность составила 2,1169E-07. Таким образом, в качестве решения задачи будем использовать значение 1,74903113.