2021

Лабораторная работа по курсу

«Методы Численного Анализа»

Решение нелинейных уравнений

Вариант 15

Выполнил: Прудникович

Владислав Вячеславович

2 курс 6 группа

Преподаватель: Будник Анатолий Михайлович

**Постановка задачи**

Найти корень нелинейного уравнения

Применяя следующие методы:

1. Метод простой итерации
2. Метод Ньютона
3. Метод Стеффенсена

Необходимо:

А. Отделить хотя бы один корень уравнения методом дихотомии.

Б. Выбрать начальное приближение исходя из условий теоремы о сходимости метода простой итерации

В. Используя выбранное найти тремя вышеуказанными способами решение нелинейного уравнения с точностью = . Критерий остановки итерационных процессов

=

Г. Сравнить методы по скорости сходимости и точности(невязка)

Пункт А. Отделение корня методом дихотомии.

Для того, чтобы запустить процесс отделения корня методом деления отрезка пополам, нам необходимо найти значения Для более корректного выбора начальных значений , проведем анализ производной нашего уравнения.

. Наша функция стремительно растёт при значениях || > 1, следовательно, наши корни будут располагаться в окрестности 0, что позволяет нам выбрать относительно небольшие .

Возьмём = -10, = 10. Заметим, что они действительно подходят начальному условию, и запустим процесс дихотомии.

Для этого возьмём середину отрезка [] и обозначим полученное значение, как

= .

Если то = ; если то = .

Будем повторять данный процесс, пока не достигнем промежутка, на котором наша производная не будет знакопостоянна.

> 0 [] или < 0 []. Из формулы производной следует, что данное условие будет выполняться, когда < по модулю, либо > , но одного знака.

В моём случае потребовалось 6 итераций метода дихотомии, чтобы корректно отделить корень.

Корень будем искать на отрезке [ 0.9375 ; 1.25 ].

Для прехода к методу простой итерации перепишем исходное уравнение в каноническом виде.

= =

=

Пункт Б. Выбор начального приближения, руководствуясь теоремой о сходимости МПИ.

Для сходимости метода простой итерации необходимо и достаточно выполнение следующих условий:

1. q() должна быть определена и непрерывна в области

={ :|<=δ}

1. |q(’) - q(’’)|<=|’- ’’|, где ||< 1, ’ ’’ . (условие Липшица)
2. || <= m : m, δ и связаны неравенством <= δ.

Рассмотрим каждое условие по порядку.

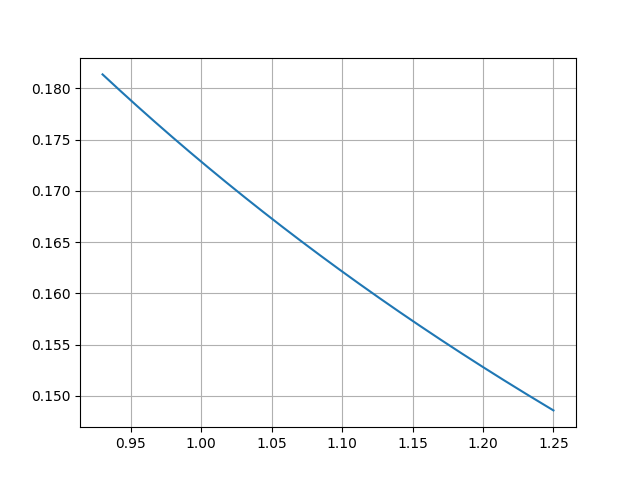
1. Заданная функция q() определена и непрерывна для любых вещественных. Следовательно, первое условие выполняется для любой области .
2. Заменим данное условие на более строгое:

. В нашем случае областью является отрезок [ 0.9375 ; 1.25 ]. Графически проверим на нём значения производной .

Значение нашей производной не превышает = 0.185.

1. В качестве возьмем середину нашего отрезка.

= 1.09375.

|| <= m = 0.04089

Теперь вычислим окрестность для точки , для которой выполняется неравенство.

<= 0.05 = δ. Тогда нашнй областью сходимости МПИ будет являться отрезок [-δ; +δ] = [1.04375 ; 1.14375].

Для заданного отрезка и начального приближения метод простой итерации является сходящимся.

Пункт В1. Решение нелинейного уравнения методом простой итерации.

1. Отделим корень методом дихотомии и возьмём середину получившегося отрезка за начальное приближение.
2. Запишем уравнение исходной функции в каноническом виде

= =

1. Построим график функции на отрезке [] (найденном методом дихотомии).
2. Запустим итерационный процесс с начальным приближением вида:

= , пока = , = 0, 1, 2, …   
Мы можем воспользоваться данным критерием, т.к. коэффициент сжатия < 0.185 < 0.5.

1. Находим значение невязки, подставив в функцию

Код программы.

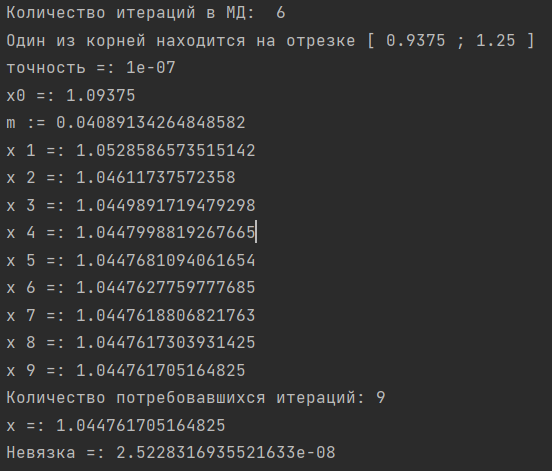
import numpy as np  
import matplotlib.pyplot as plt

def mpi(z):  
 otr = 0  
 z = z+0.2  
 if z < 0:  
 z = -z  
 otr = 1  
 z = pow(z, 0.2)  
 if otr == 1: z = -z  
 return z  
  
def f(x1):  
 x1 = pow(x1, 5) - x1 - 0.2  
 return x1

def dihotom(a, b):  
 count = 0  
 while (a < 1 or b < 1) and (abs(a) > 1 or abs(b) > 1) and (a > -1 or b > -1):  
 xd = (a+b)/2  
 if f(xd) < 0: a = xd  
 else: b = xd  
 count += 1  
 print("Количество итераций в МД: ", count)  
 print("Один из корней находится на отрезке [", a, ";", b, "]")  
 xd = (a+b)/2  
 return xd  
  
eps = 0.0000001  
a = -10  
b = 10  
x0 = 0  
delta = 1  
y = lambda x: x\*\*5 - 0.2 - x  
y1 = lambda x: 0\*x  
fig, ax = plt.subplots()  
t = np.linspace(-1.5, 1.5, 1000)  
plt.plot(t, y(t))  
plt.plot(t, y1(t))  
ax.grid()  
plt.show()  
t2 = np.linspace(1, 1.1, 1000)  
y2 = lambda x: 1/(5\*pow((x+0.2), 0.8))  
fig2, ax2 = plt.subplots()  
plt.plot(t2, y2(t2))  
ax2.grid()  
plt.show()  
numb = 1  
x0 = dihotom(-10, 10)  
m = abs(x0 - mpi(x0))  
print("точность =:", eps)  
print("x0 =:", x0)  
print("m :=", m)  
i = 1  
while delta > eps:  
 x = mpi(x0)  
 delta = abs(x - x0)  
 x0 = x  
 print("x", i, "=:", x)  
 i += 1  
print("Количество потребовавшихся итераций:", i-1)

print("х =:", x)  
r = f(x)  
print("Невязка =:", r)

Результат работы программы.



Пункт В2. Решение нелинейного уравнения

методом Ньютона.

1. Рассмотрим значение второй производной F(x) в точке и сравним его знак со значением .

F’’() =

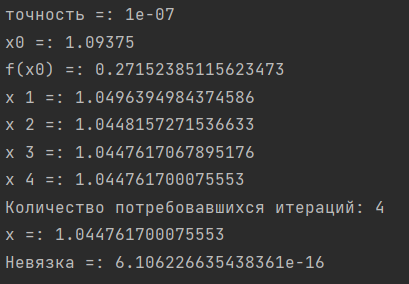
F() = 0,2715 > 0. Мы можем запустить итерационный из данной точки.

1. Запустим итерационный процесс с начальным приближением вида:

= , пока = , = 0, 1, 2, …

1. Находим значение невязки, подставив в функцию

Код программы.

def fpr(x):  
 x = 5\*pow(x, 4)-1  
 return x  
def f(x1):  
 x1 = pow(x1, 5) - x1 - 0.2  
 return x1  
def newton(x):  
 x = x - f(x)/fpr(x)  
 return x  
eps = 0.0000001  
delta = 1  
x0 =   
print("точность =:", eps)  
print("x0 =:", x0)  
print("f(x0) =:", f(x0))  
i = 1  
while delta > eps:  
 x = newton(x0)  
 delta = abs(x - x0)  
 x0 = x  
 print("x", i, "=:", x)  
 i += 1  
print("Количество потребовавшихся итераций:", i-1)  
print("х =:", x)  
r = f(x)  
print("Невязка =:", r)

Результат работы программы

Пункт В3. Решение нелинейного уравнения

методом Стеффенсена.

1. Приведём исходное уравнение к каноническому виду = .
2. Запустим итерационный процесс с начальным приближением вида:

= пока = , = 0, 1, 2, …

1. Находим значение невязки, подставив в функцию .

Код программы.

def mpi(z):  
 otr = 0  
 z = z+0.2  
 if z < 0:  
 z = -1\*z  
 otr = 1  
 z = pow(z, 1/5)  
 if otr == 1:  
 z = -1\*z  
 return z

def steff(x2):  
 phi = mpi(x2)  
 x2 = (x2\*mpi(phi)-phi\*phi)/(mpi(phi)-2\*phi+x2)  
 return x2

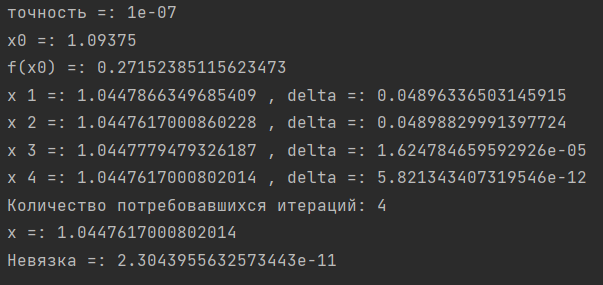
def f(x1):  
 x1 = pow(x1, 5) - x1 - 0.2  
 return x1

eps = 0.0000001  
delta = 1x0 = 1.09375  
x1 = 1.09375  
print("точность =:", eps)  
print("x0 =:", x0)  
print("f(x0) =:", f(x0))  
i = 1  
count = 0

while delta > eps:  
 x = steff(x0)  
 delta = abs(x - x1)  
 count+=1  
 x0 = x  
 if count == 2:  
 x1 = x  
 count = 0  
 print("x", i, "=:", x, ", delta =:", delta)  
 i += 1

print("Количество потребовавшихся итераций:", i-1)  
print("х =:", x)  
r = f(x)  
print("Невязка =:", r)

Результат работы программы.



Пункт Г. Сравнение скорости сходимости и точности полученного результата.

**Анализ методов.**

Первым делом рассчитаем априорную оценку сходимости для МПИ. Пусть n - количество требуемых итерации в методе для удовлетворения заданной точности.

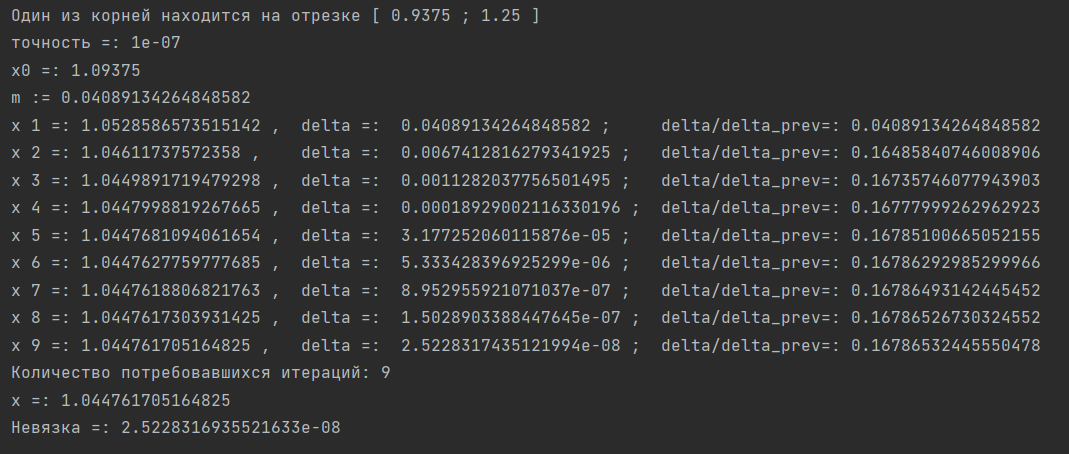
Для метода простой итерации , где q - коэффициент сжатия, и в нашем случае q = 0.185 (пункт Б.2), = ,

m = 0.04089.

= 7.78. Тогда, для достижения заданной точности в методе МПИ нам необходимо сделать не менее 8 итераций.   
Практика показывает, что их потребовалось 9.

Заметим, что для соблюдения заданной точности 8 итераций в нашем случае будет недостаточно. Это значит, что априорная оценка снизу соответстует работе метода на практике.

Вычислим отношения погрешностей = для соседних итераций.



Заметим, что 0.164 < /< 0.168 > 1, => МПИ имеет линейную сходимость.

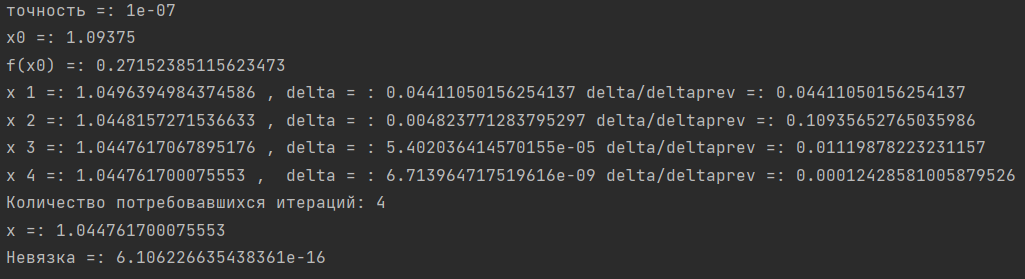
Значение невязки для МПИ = 2.5228316935521633e-08

Количество итераций по априорной оценка сходимости для метода Ньютона

В нашем случае n = 1.889. То есть потребуется минимум 2 итерации для того, чтобы значение полученного решениея удовлетворяло заданной точности.

На практике метод Ньютона для исходного уравнения сходится за 4 итерации.

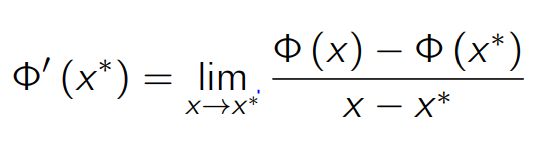
Если рассмотреть отношения и для > 1, то заметим, что коэффициент k = / стремительно падает при приближении к корню. Это обусловлено квадратичной сходимостью метода Ньютона.



Значение невязки для метода Ньютона = 6.106226635438e-16.

Проведем аналогичные вычисления для Метода Стеффенсона, который является модернизацией МПИ.

Рассомтрим приближенное значение производной функции в найденной точке решения.

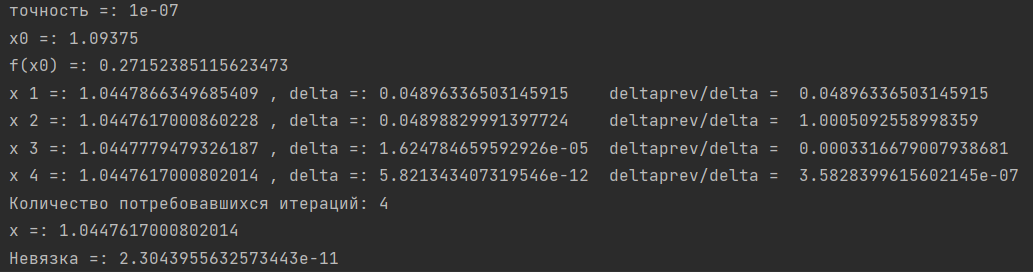


Возьмем x = x\*+. и найдём приближённо Ф’(x\*).

Ф’(x\*) = 597.1478431065597 >>0

Полученный результат означает, что мы не можем рассчитывать на квадратичную скорость сходимости метода Стеффенсена.

Однако, на практике мы получаем k = / => 0, что свидетельствует о нелинейной скорости сходимости метода.



Значение невязки для метода Стеффенсона = 6.10622663e-16.

**Вывод.**

Для решения уравнения методом Стеффенсена и методом Ньютона понадобилось 4 итерации, в то время как МПИ справляется только за 9. Проведя анализ погрешности на каждом шаге итерации мы выяснили, что в заданном случае метод Стеффенсена и Ньютона обладают нелинейной сходимостью, в отличии от МПИ, погрешность которого уменьшается в соответствии с коэффициентом сжатия q на выбранном отрезке.

Самое точное значение невязки показал метод Ньютона:

= 6.106226635438e-16.

Значение невязки для метода Стеффенсена:

= 6.10622663e-16.

Значение невязки для метода простой итерации:

= 2.5228316935521633e-08