**ОТЧЁТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ № 1**

**ПРИБЛИЖЕННЫЕ МЕТОДЫ РЕШЕНИЯ**

**НЕЛИНЕЙНЫХ УРАВНЕНИЙ**

**(Вариант 9)**

*Выполнил студент 3 курса МОиАИС*

*Ходосевич Данила*

***Постановка задачи:*** исследовать функцию и решить уравнение .

I. Найти промежуток, содержащий наименьший положительный корень уравнения , для которого выполняются достаточные условия сходимости одного из итерационных методов;

II. Получить приближенное решение (с точностью 10-7) методами:

1) *методом Ньютона (метод касательных)*

;

2) *методом хорд*

;

3) *методом секущих*

;

4) *конечноразностным методом Ньютона*

— малый параметр;

5) *методом Стеффенсена*

;

6) *методом простых итераций*

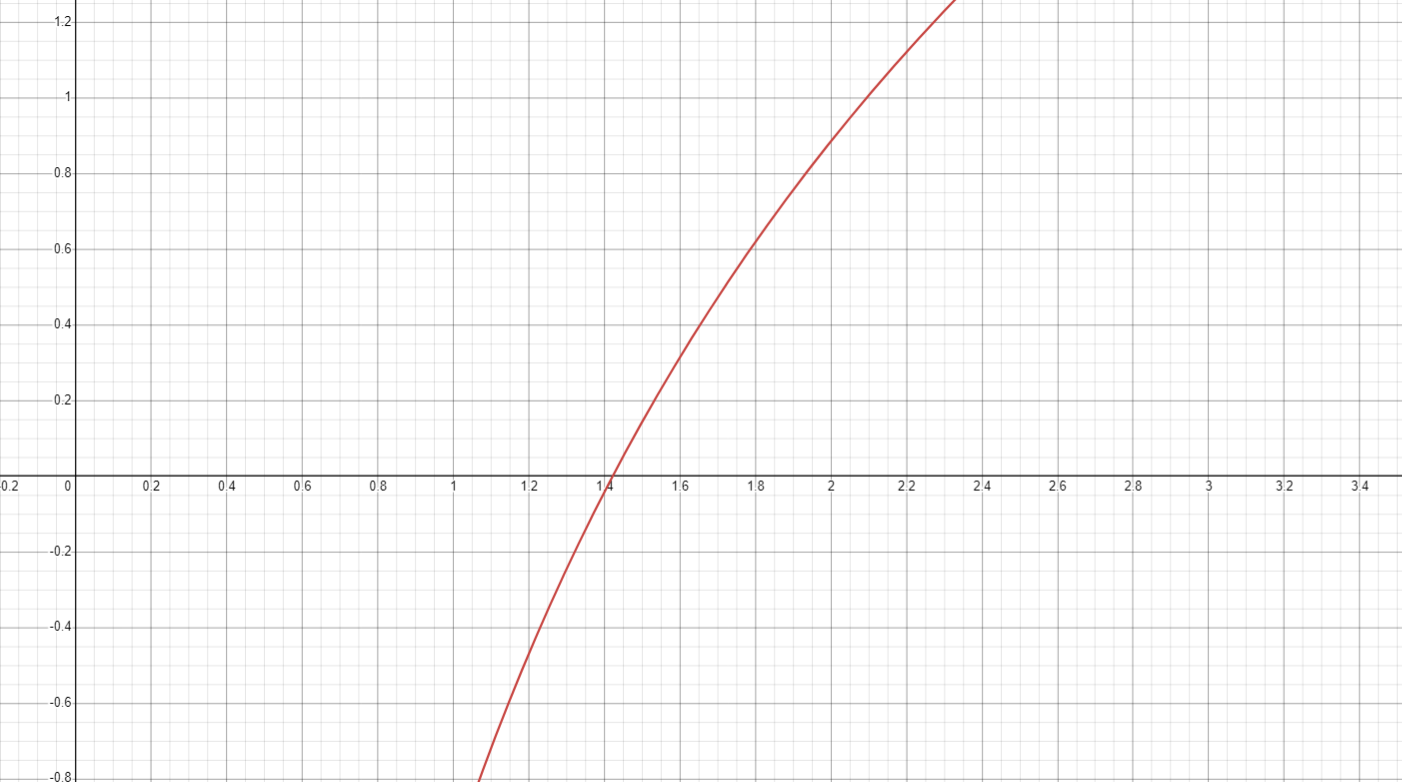
Если , то .

В моём случае , так, по сути, получаем метод Ньютона по одной касательной.

Для оценки погрешности приближенного решения, полученного любым методом, используем разность последних двух проходов. Очевидно, что:

***Результаты расчетов***

Строим график функции



Построив график функции, определяем, что уравнение имеет один корень, в интервале .

Уточним значение корня с точностью 10-7, пользуясь методами 1–6.

**Метод Ньютона (метод касательных).** Для корректного использования данного метода необходимо определить поведение первой и второй производных функции на интервале уточнения корня и правильно выбрать начальное приближение .

Для функции имеет начальное приближение f(

Bычисления проводятся по формуле .

Итерации завершаются при выполнении условия .

|  |  |
| --- | --- |
|  |  |
| 1 | 0.100000000 |
| 2 | 0.221709752 |
| 3 | 0.477909159 |
| 4 | 0.894704632 |
| 5 | 1.279314276 |
| 6 | 1.412237305 |
| 7 | 1.421491572 |
| 8 | 1.421529935 |
| 9 | 1.421529936 |

**Метод хорд.**

Вычисления проводятся по формуле: .

Итерации завершаются при выполнении условия .

|  |  |
| --- | --- |
|  |  |
| 1 | 2.000000000 |
| 2 | 1.891297606 |
| 3 | 1.804271793 |
| 4 | 1.734218413 |
| 73 | 1.421530579 |
| 74 | 1.421530468 |
| 75 | 1.421530377 |

**Метод секущих.**

Вычисления проводятся по формуле

Итерации завершаются при выполнении условия

|  |  |
| --- | --- |
|  |  |
| 1 | 0.100000000 |
| 2 | 2.000000000 |
| 3 | 1.891297606 |
| 4 | 1.314316695 |
| 5 | 1.443065382 |
| 6 | 1.422564710 |
| 7 | 1.421520085 |
| 8 | 1.421529940 |
| 9 | 1.421529936 |

**Конечноразностный метод Ньютона.**

В качестве начального приближения берем

Вычисления проводятся по формуле .

|  |  |
| --- | --- |
|  |  |
| 1 | 0.100000000 |
| 2 | 0.232809135 |
| 3 | 0.509053765 |
| 4 | 0.940931160 |
| 5 | 1.307442255 |
| 6 | 1.416120279 |
| 7 | 1.421540944 |
| 8 | 1.421529887 |
| 9 | 1.421529936 |

**Метод Стеффенсена.**

В качестве начального приближения берем

Вычисления проводятся по формуле .

|  |  |
| --- | --- |
|  |  |
| 1 | 2.000000000 |
| 2 | 1.114597912 |
| 3 | 1.255252177 |
| 4 | 1.380409375 |
| 5 | 1.419283175 |
| 6 | 1.421523432 |
| 7 | 1.421529936 |
| 8 | 1.421529936 |

**Метод простых итераций.**

Выбираем .

Вычисления проводятся по формуле . Выбираем

|  |  |
| --- | --- |
|  |  |
| 1 | 0.100000000 |
| 2 | 5.294258106 |
| 3 | 4.175981034 |
| 20 | 1.421530093 |
| 21 | 1.421529987 |
| 22 | 1.421529952 |

**Итоговая таблица**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Метод решения | Выбранный интервал | Полученное решение | Количество итераций | Погрешность |
| 1. Метод Ньютона | [0.5, 2] | 1.421529936 | 10 | 10-8 |
| 2. Метод Хорд | [0.5, 2] | 1.421530377 | 63 | 10-8 |
| 3. Метод Секущих | [0.5, 2] | 1.421529936 | 12 | 10-8 |
| 4. Конечноразностный метод Ньютона | [0.5, 2] | 1.421529936 | 14 | 10-8 |
| 5. Метод Стефенсена | [0.5, 2] | 1.421529936 | 9 | 10-8 |
| 6. Метод простых итераций | [0.5, 2] | 1.421529952 | 12 | 10-8 |

**Выводы:** Метод Ньютона (метод касательных) самый эффективный для нахождения приближенного решения, метод простых итераций является самым не эффективным, из полученных результатов.

Приближенным решением уравнения является .

Все исходные тексты программ приводятся в Приложении

**ПРИЛОЖЕНИЕ**

***Программа***

# вариант 9

class DiffFunction:

    def \_\_init\_\_(self, \*derivatives):

        self.derivatives = derivatives

    def \_\_call\_\_(self, x, order=0):

        return self.derivatives[order](x)

def try\_spread(arr\_like):

    return list(arr\_like) if hasattr(arr\_like, '\_\_iter\_\_') else [arr\_like]

def iterative\_solver(start, step\_func, epsilon=1e-7, halt\_predicate=lambda pr, nx, e: abs(nx - pr) < e):

    steps = try\_spread(start)

    steps.append(step\_func(steps))

    while not halt\_predicate(steps[-2], steps[-1], epsilon):

        steps.append(step\_func(steps))

    return steps[-1], steps

def newton(f: DiffFunction, a, b, epsilon=1e-7): #Ньютона (метод касательных)

    step\_function = lambda x: x[-1] - f(x[-1]) / f(x[-1], 1)

    start = a if f(a) \* f(a, 2) > 0 else b

    return iterative\_solver(start, step\_function, epsilon)

def chords(f: DiffFunction, a, b, epsilon=1e-7): #метод хорд

    if f(b) \* f(b, 2) <= 0:

        a, b = b, a

    step\_function = lambda x: x[-1] - f(x[-1]) \* (b - x[-1]) / (f(b) - f(x[-1]))

    return iterative\_solver(a, step\_function, epsilon)

def secants(f: DiffFunction, a, b, epsilon=1e-7): #метод секущих

    step\_function = lambda x: x[-1] - f(x[-1]) \* (x[-1] - x[-2]) / (f(x[-1]) - f(x[-2]))

    return iterative\_solver([a, b], step\_function, epsilon)

def finite\_sum\_newton(f: DiffFunction, a, b, epsilon=1e-7, h=0.01): #конечноразностный Нютона

    step\_function = lambda x: x[-1] - h \* f(x[-1]) / (f(x[-1] + h) - f(x[-1]))

    return iterative\_solver(a, step\_function, epsilon)

def simple\_iter(f: DiffFunction, a, b, epsilon=1e-7, tau=None): #просых итераций

    c = (a + b) / 2

    tau = tau or 1 / f(c, 1)  # Default tau based on first derivative at midpoint

    step\_function = lambda x: x[-1] - tau \* f(x[-1])

    return iterative\_solver(a, step\_function, epsilon)