МИНИСТЕРСТВО НАУКИ И ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«Вятский государственный университет»**
**(ФГБОУ ВО «ВятГУ»)**
Факультет автоматики и вычислительной техники
Кафедра электронных вычислительных машин

# ЧИСЛЕННЫЕ МЕТОДЫ РЕШЕНИЯ НЕЛИНЕЙНЫХ УРАВНЕНИЙ

Отчет по лабораторной работе №1 дисциплины
«Вычислительная математика»

Выполнил студент группы ИВТ-21 _____ / Рзаев А. Э./
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 Wolphram Alpha.

Уравнение: sin(x / 2) + 1 - x ^ 2 = 0
Интервал: [1.0; 2.8]

2 Теоретические сведения

2.1 Теоретические сведения об уточнении 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).

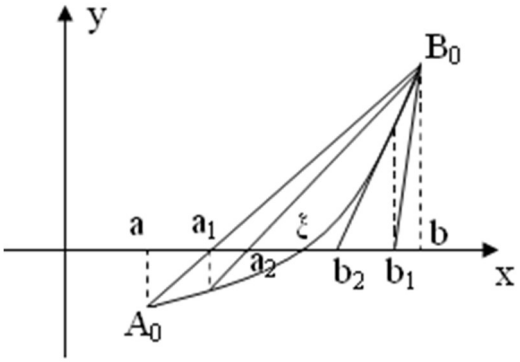


Рисунок 1

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:

$$a_1 = a - \frac{f(a)(b-a)}{f(b)-f(a)}$$

$$b_1 = b - \frac{f(b)}{f'(b)}$$

Теперь корень $\xi$ заключен в интервале $[a_1, b_1]$. Применяя к этому отрезку комбинированный метод, получим:

$$a_2 = a_1 - \frac{f(a_1)(b_1-a_1)}{f(b_1)-f(a)}$$

$$b_2 = b_1 - \frac{f(b_1)}{f'(b_1)}$$

и т.д.

$$a_{n+1} = a_n - \frac{f(a_n)(b_n - a_n)}{f(b_n) - f(a_n)}$$
$$b_{n+1} = b_n - \frac{f(b_n)}{f'(b_n)} \qquad (2.6)$$

Если же $f'(x)\,f''(x)<0$ (рисунок 2), то, рассуждая аналогично, получим следующие формулы для уточнения корня уравнения:

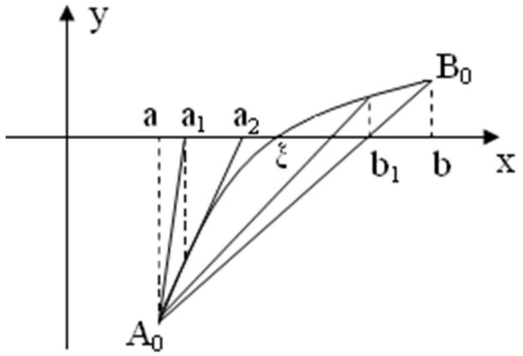


Рисунок 2

$$a_{n+1} = a_n - \frac{f(a_n)}{f'(a_n)}$$
$$b_{n+1} = b_n - \frac{f(b_n)(b_n - a_n)}{f(b_n) - f(a_n)}$$

Вычислительный процесс прекращается, как только выполнится условие:

$$|b_{n+1} - a_{n+1}| < \varepsilon$$

2.2 Теоретические сведения об уточнении корня методом итераций

Представим исходное уравнение $f(x) = 0$ в виде x=φ(x). (2.21)

Пусть нам известно начальное приближение к корню $x_0$ ($x_0 \in [a,b]$). Подставив его в правую часть уравнения (2.21) получим новое приближение $x_1 = \varphi(x_0)$, затем аналогичным образом получим $x_2 = \varphi(x_1)$ и так далее,

$x_{k+1} = \varphi(x_k)$, $k = 0,1,\ldots$. (2.22)

Для того чтобы итерационный процесс (2.22) был сходящимся, необходимо, чтобы абсолютная величина производной $\varphi'(x)$ в окрестности корня была меньше единицы. Если это условие выполняется на отрезке $[a,b]$ на котором локализован корень, то в качестве начального приближения можно взять любую точку из этого отрезка $x_0 \in [a,b]$. Скорость сходимости зависит от абсолютной величины производной $|\varphi'(x)|$: чем меньше $|\varphi'(x)|$ вблизи корня, тем быстрее сходится процесс.

Переход от уравнения (2.1) к уравнению в итерационной форме (2.21) можно осуществить различными способами в зависимости от вида функции $f(x)$. При таком переходе необходимо построить функцию $\varphi(x)$ так, чтобы выполнялось условие сходимости.

Рассмотрим один из общих алгоритмов перехода от уравнения $f(x)=0$ к уравнению $x=\varphi(x)$. Умножим левую и правую части уравнения $f(x)=0$ на произвольную константу $\tau \neq 0$ и добавим к обеим частям неизвестное *x*. При этом корни исходного уравнения не изменятся

$$x+\tau \cdot f(x)=x+0\cdot\tau$$

или

$$x=x+\tau f(x). \qquad (2.24)$$

Уравнение (2.24) эквивалентно уравнению (2.21) с функцией $\varphi(x)=x+\tau f(x)$. Произвольный выбор константы *t* позволяет обеспечить выполнение условия сходимости. Поскольку в данном случае $\varphi'(x)=1+\tau f'(x)$, значение *t* следует выбирать, так чтобы в окрестности корня выполнялось условие

$$|\varphi'(x)|=|1+\tau f'(x)|<1. \qquad (2.25)$$

Наибольшую скорость сходимости в методе простых итераций получим при $\varphi'(x)=0$. Этого можно добиться, если выбрать параметр *t* зависящим от *x* в виде

$$\tau(x)=-\frac{1}{f'(x)}. \qquad (2.26)$$

При этом итерационная формула (2.22) переходит в формулу Ньютона

$$x_{k+1}=x_k-f(x_k)/f'(x_k).$$

Таким образом, метод Ньютона можно трактовать как частный случай метода простых итераций, обладающий максимальной скоростью сходимости.

## 3 Необходимые расчеты

Исходное уравнение

sin(x / 2) + 1 - x ^ 2 = 0

Исходная функция

f(x) = sin(x / 2) + 1 - x ^ 2

Первая производная

f ’= 0.5 * cos(x / 2) - 2 * x

Вторая производная

f ’’ = 0.25 * sin(x / 2) - 2

Преобразование исходного уравнения к виду φ(x) = x

φ(x) = x – f(x) / f’(x)

φ(x) = (cos(x / 2) – 2 * sin (x / 2) – 2 * x ^ 2 – 2) / (cos(x / 2) – 4 * x)

## 4 Результаты выполнения программы

Листинг программ представлен в приложении А

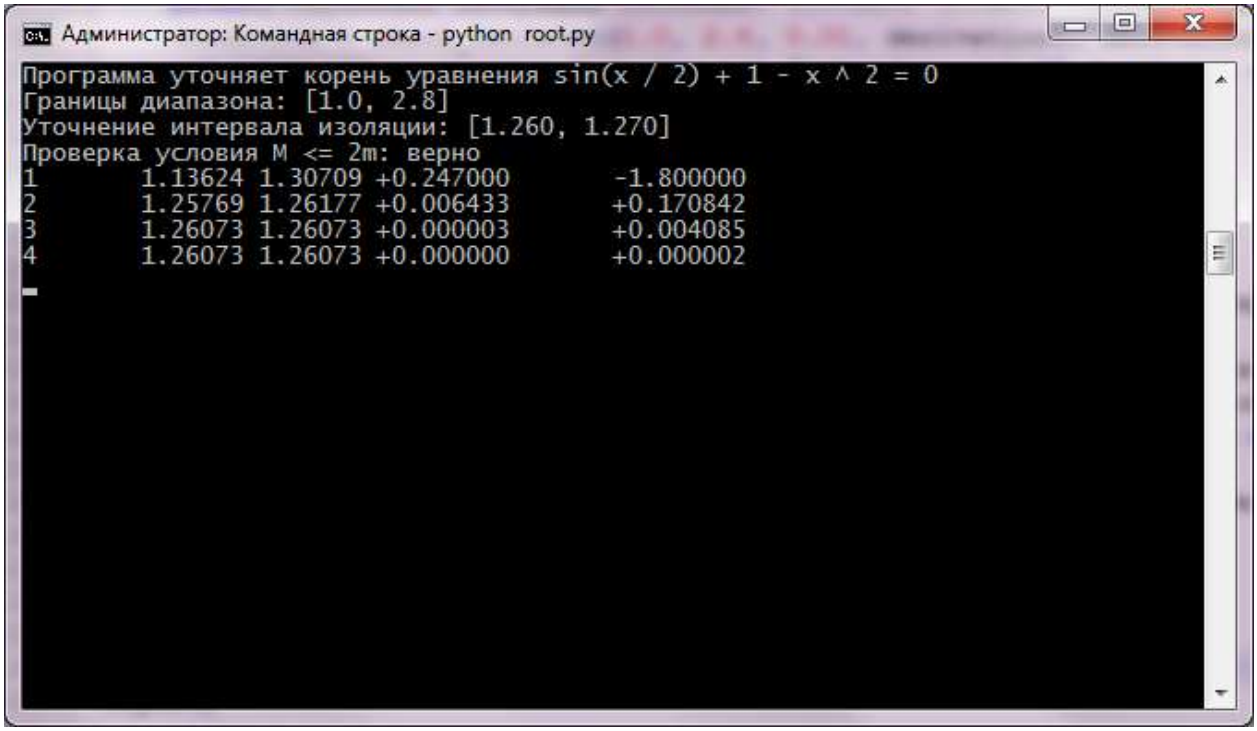


Рисунок 3 – Результат выполнения программы, реализующей комбинированный метод

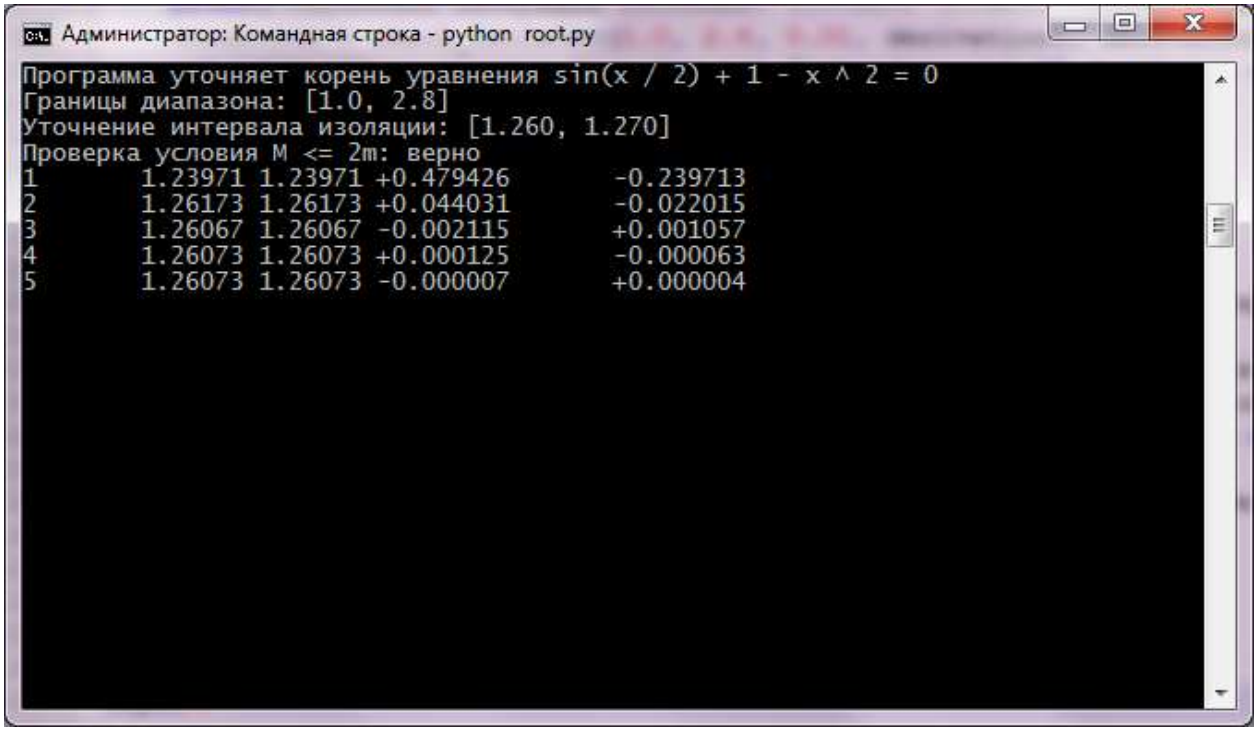


Рисунок 4 – Результат выполнения программы, реализующей метод итераций

## 5 Результат проверки выполнения программы

Итоговый ответ, полученный с помощью программы: x = 1.26073 ± 0,00001.

Для построения графика функции и проверки корня уравнения была использована математическая система Wolfram Alpha. Изображение графика представлено на рисунке 5. Результаты проверки корня представлены на рисунке 6.

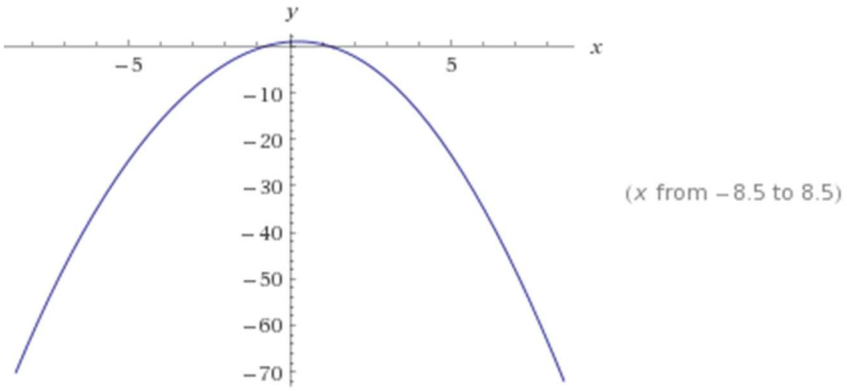


Рисунок 5 – График функции f(x) = sin(x / 2) + 1 - x ^ 2

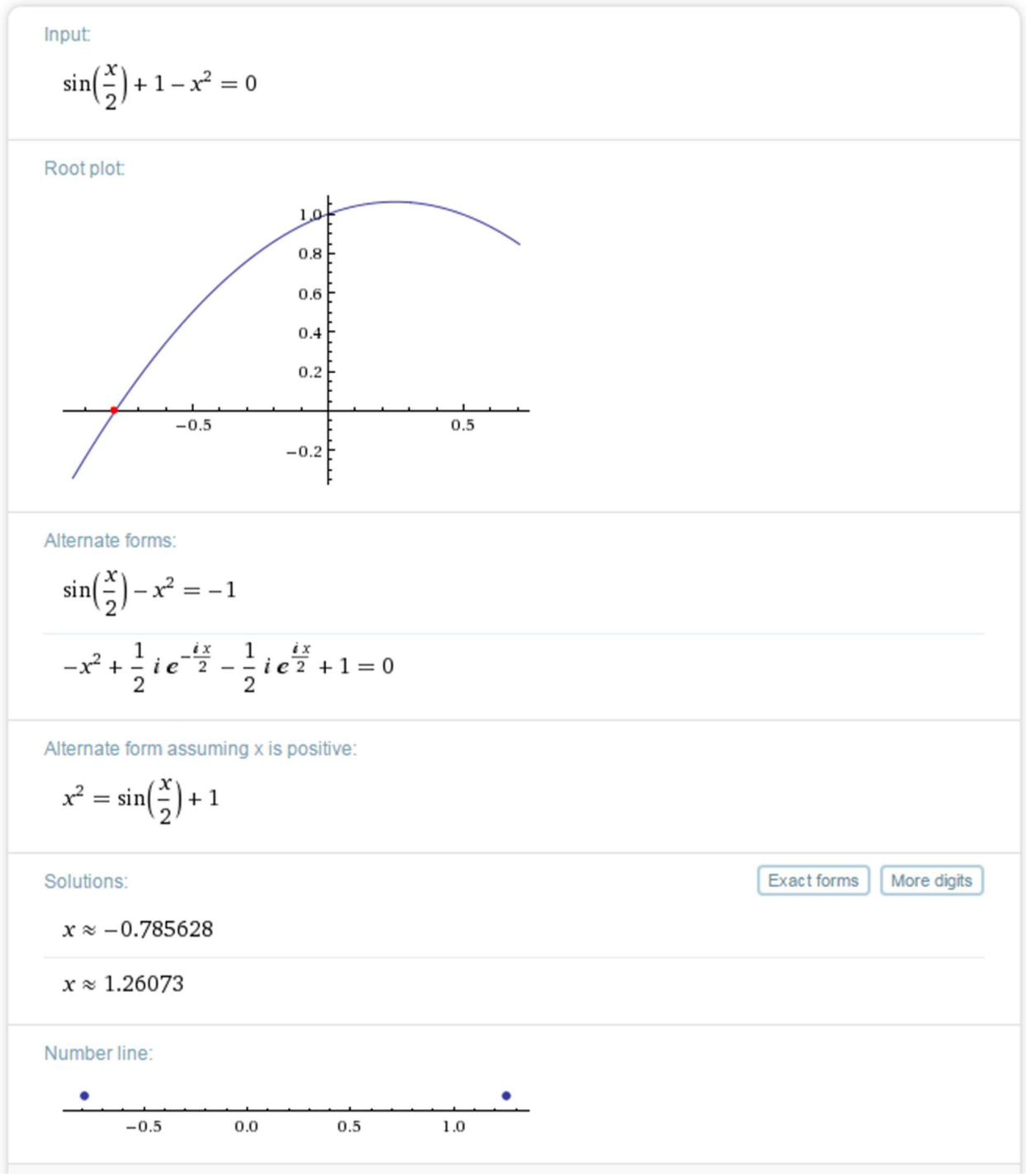


Рисунок 6 – Результат проверки корня

## 6 Выводы

В ходе лабораторной работы были реализованы алгоритмы для уточнения корня комбинированным методом хорд и касательных и методом итераций. Из полученных результатов следует, что комбинированный метод является более быстрым по сравнению с методом итераций.

Приложение А

Комбинированный метод

```
def func(x):
    from math import sin
    return sin(x / 2) + 1 - x ** 2


def derivation1(x):
    from math import cos
    return 0.5 * cos(x / 2) - 2 * x


def derivation2(x):
    from math import sin
    return 0.25 * sin(x / 2) - 2


def applicator(left, right, eps, functor, applicable):
    res = abs(functor(left))
    while left < right:
        left += eps / 2
        res = applicable(res, abs(functor(left)))

    return res


def combined_method(left, right, eps, functor, deriv1, deriv2):
    def hord1(xn, xi):
        return xi - functor(xi) * (xi - xn) / (functor(xi) -
functor(xn))

    fixed = functor(left) * deriv2(left) > 0
    if fixed:
        xn, xi = left, right
    else:
        xn, xi = right, left

    n = 0

    while True:
        n += 1

        delta = xi - xn
        if fixed:
            xi = hord1(xn, xi)
            xn = xn - functor(xn) / deriv1(xn)
        else:
            xn = hord1(xi, xn)
            xi = xi - functor(xi) / deriv1(xi)

        y = functor(xn)

        yield n, xn, xi, y, delta

        if abs(delta) < eps:
            break
```

```
def isolate(acc, left, right, functor):
    while functor(left + acc) * functor(right - acc) < 0:
        left += acc
        right -= acc

    while functor(left) * functor(left + acc) > 0:
        left += acc

    while functor(right) * functor(right - acc) > 0:
        right -= acc

    return left, right


def check_isolated(left, right, eps, deriv1, deriv2):
    mx = applicator(left, right, eps, deriv2, max)
    mn = applicator(left, right, eps, deriv1, min)

    return mx < 2 * mn

func_repr = 'sin(x / 2) + 1 - x ^ 2'
deriv1_repr = '0.5 * cos(x / 2) - 2 * x'
deriv2_repr = '0.25 * sin(x / 2) - 2'


def show_combined_method():
    print('Программа уточняет корень уравнения {} =
0'.format(func_repr))
    print('Границы диапазона: [{}, {}]'.format(1.0, 2.8))
    print('Уточнение интервала изоляции: [{:.3f},
{:.3f}]'.format(*isolate(0.01, 1.0, 2.8, func)))
    condition = check_isolated(1.0, 2.8, 0.01, derivation1,
derivation2)
    print('Проверка условия M <= 2m: {}'.format('верно' if condition
else 'неверно'))

    for t in combined_method(1.0, 2.8, 0.00001, func, derivation1,
derivation2):
        print('{}\t{:.5f}\t{:.5f}\t{:+.6f}\t{:+.6f}'.format(*t))

show_combined_method()
input()
```

Метод итераций

```
def iterative_method(left, right, eps, functor, deriv1):
    sign = lambda x: 1 if x >= 0 else -1
    ceil = lambda x: x if int(x) == x else int(x) + 1
    N = max(deriv1(left), deriv1(right))
    k = sign(deriv1(left)) * (ceil(N / 2) + 1)

    xn = left
    n = 0

    while True:
        n += 1
```

```
        xn1 = xn - functor(xn) / k
        y = functor(xn)
        delta = xn - xn1
        xn = xn1

        yield n, xn, xn1, y, delta

        if abs(delta) < eps:
            break

def show_iterative_method():
    print('Программа уточняет корень уравнения {} =
0'.format(func_repr))
    print('Границы диапазона: [{}, {}]'.format(1.0, 2.8))
    print('Уточнение интервала изоляции: [{:.3f},
{:.3f}]'.format(*isolate(0.01, 1.0, 2.8, func)))
    condition = check_isolated(1.0, 2.8, 0.01, derivation1,
derivation2)
    print('Проверка условия M <= 2m: {}'.format('верно' if condition
else 'неверно'))

    for t in iterative_method(1.0, 2.8, 0.00001, func, derivation1):
        print('{}\t{:.5f}\t{:.5f}\t{:+.6f}\t{:+.6f}'.format(*t))


show_iterative_method()
input()
```