**Министерство науки и высшего образования Российской Федерации**

Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

**Дисциплина:** Вычислительная математика

**Лабораторная работа №2**

**«Численное решение нелинейных уравнений и систем»**

**Вариант №10**

Выполнил:

Мокров Семён Андреевич

Группа:

P3215

Проверила:

Малышева Татьяна Алексеевна

Санкт-Петербург

2022

# Цель работы

Изучить численные методы решения нелинейных уравнений и их систем, найти корни заданного нелинейного уравнения, выполнить программную реализацию методов.

# Задание лабораторной работы

**Вычислительная реализация задачи:**

Уточнить корни нелинейного уравнения с точностью =10-2. Вычисления оформить в виде таблиц, удержать 3 знака после запятой.

**Программная реализация задачи:**

**Для нелинейных уравнений:**

1. Все численные методы должны быть реализованы в виде отдельных подпрограмм или классов.

2. Пользователь выбирает уравнение, корень/корни которого требуется вычислить (3-5 функций, в том числе и трансцендентные), из тех, которые предлагает программа.

3. Предусмотреть ввод исходных данных (границы интервала/начальное приближение к корню и погрешность вычисления) из файла или с клавиатуры по выбору конечного пользователя.

4. Выполнить верификацию исходных данных. Для метода Ньютона (метода секущих) – выбор начального приближения (а или b). Для метода простой итерации – достаточное условие сходимости метода. Программа должна реагировать на некорректные введенные данные.

5. Предусмотреть вывод результатов (найденный корень уравнения, значение функции в корне, число итераций) в файл или на экран.

6. Организовать вывод графика функции, график должен полностью отображать весь исследуемый интервал (с запасом).

**Для систем нелинейных уравнений:**

1. Рассмотреть систему двух уравнений.

2. Организовать вывод графика функций.

3. Для метода простой итерации проверить достаточное условие сходимости.

4. Вывод вектора неизвестных: x1, x2.

5. Вывод количества итераций, за которое было найдено решение.

6. Вывод вектора погрешностей: xi(k)- xi(k-1)

# Описание метода, расчетные формулы

**Решение нелинейных уравнений:**

* ***Метод простой итерации:***

*Рабочая формула метода:*

*Геометрический смысл*:

Изображение выглядит как текст, небо, карта

Автоматически созданное описание

Уравнение fx=0 приводится к эквивалентному виду: *x* = φ(*x*), выражая *x* из исходного

уравнения.

Через начальное приближение: x0 ∈ a, b, находятся очередные приближения:

*Достаточное условие сходимости метода:*

, где q-некоторая константа.

*Критерий окончания итерационного процесса:*

* ***Метод хорд:***

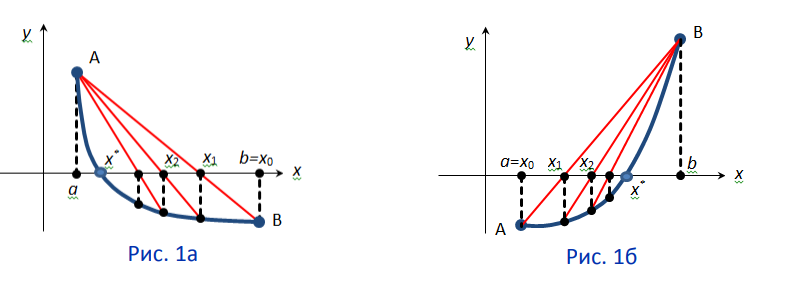
а) при фиксированном левом конце хорд, тогда х0=b:

Рабочая формула метода:

б) при фиксированном правом конце хорд, тогда х0=a:

Рабочая формула метода:

*Геометрический смысл*:

****

*Достаточное условие сходимости метода:*

* функция y= f(x) определена и непрерывна на отрезке [𝑎; 𝑏];
* f(a)·f(b) < 0 (на концах отрезка [a;b] функция имеет разные знаки);
* производные f′(x) и f′′(x) сохраняют знак на отрезке [a;b]

*Критерий окончания итерационного процесса:*

****

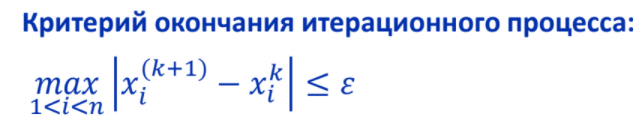
**Решение систем нелинейных уравнений:**

* ***Метод Ньютона:***

К основе метода лежит использование разложения функций Fi(x1, x2, …, xn) в окрестности некоторой фиксированной точки в ряд Тейлора, причем члены, содержащие вторые (и более высоких порядков) производные, отбрасываются.

Изображение выглядит как текст

Автоматически созданное описание



# Заполненные таблицы

***Уравнение:***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № итерации | *xk* | *f*(*xk* ) | *f '*(*xk*) | *xk*+1 | │*xk* − *xk*+1│ |
| 1 | 4.500 | 14.552 | 29.125 | 4.000 | 0.499 |
| 2 | 4.000 | 2.465 | 19.507 | 3.874 | 0.126 |
| 3 | 3.874 | 0.140 | 17.311 | 3.866 | 0.008 |

*Уточнение крайнего правого корня методом Ньютона*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № шага | a | b | x | f(a) | f(b) | f(x) | |a-b| |
| 1 | -1.6 | -1 | -1.3 | -4.038 | 1.833 | -0.47 | 0.6 |
| 2 | -1.3 | -1 | -1.15 | -0.47 | 1.833 | 0.829 | 0.3 |
| 3 | -1.3 | -1.15 | -1.225 | -0.47 | 0.829 | 0.218 | 0.15 |
| 4 | -1.3 | -1.225 | -1.263 | -0.47 | 0.218 | -0.117 | 0.075 |
| 5 | -1.263 | -1.225 | -1.244 | -0.117 | 0.218 | 0.053 | 0.0375 |
| 6 | -1.263 | -1.244 | -1.253 | -0.117 | 0.053 | -0.031 | 0.019 |
| 7 | -1.253 | -1.244 | -1.248 | -0.031 | 0.053 | 0.011 | 0.009 |

*Уточнение крайнего левого корня уравнения методом половинного деления*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № итерации | *xk* | *f*(*xk* ) | *xk*+1 |  | │*xk* − *xk*+1│ |
| 1 | 0.4 | 0.622 | 0.513 | 0.513 | 0.113 |
| 2 | 0.513 | -0.025 | 0.509 | 0.509 | 0.004 |

*Уточнение центрального корня уравнения методом простой итерации*

# Листинг программы

Весь исходный код программы располагается на GitHub (адрес: https://github.com/semwett0301/lab2\_math)

from math\_part import validation, utils  
  
  
def hord\_method(function, a, b, e):  
 if validation.validate\_hord(a, b, function):  
 def compare\_signs(first\_param, second\_param):  
 return first\_param \* second\_param > 0  
  
 iter\_count = 1  
  
 if compare\_signs(function(a), utils.calculate\_second\_det(function)(a)):  
 x0 = a  
 fixed = b  
 else:  
 x0 = b  
 fixed = a  
  
 x = x0 - (fixed - x0) / (function(fixed) - function(x0)) \* function(x0)  
  
 while abs(x - x0) > e and abs(function(x)) > e:  
 x0 = x  
 x = x0 - (fixed - x0) / (function(fixed) - function(x0)) \* function(x0)  
 iter\_count += 1  
  
 return x, iter\_count  
 else:  
 return None, None  
  
  
def simple\_iterations\_method(function, a, b, e):  
 det\_a = utils.calculate\_det(function)(a)  
 det\_b = utils.calculate\_det(function)(b)  
 if det\_a > det\_b:  
 lam = - 1 / det\_a  
 last\_x = a  
 else:  
 lam = - 1 / det\_b  
 last\_x = b  
  
 fi\_function = lambda x: x + lam \* function(x)  
  
 if validation.validate\_simple\_iterations(a, b, fi\_function):  
 iter\_count = 1  
 curr\_x = fi\_function(a)  
  
 q = utils.get\_max\_det\_value(function, a, b)  
 if q > 0.5:  
 while abs(last\_x - curr\_x) > e:  
 iter\_count += 1  
 last\_x = curr\_x  
 curr\_x = fi\_function(last\_x)  
 else:  
 while abs(last\_x - curr\_x) > (1 - q) \* e / q:  
 iter\_count += 1  
 last\_x = curr\_x  
 curr\_x = fi\_function(last\_x)  
  
 return curr\_x, iter\_count  
 else:  
 return None, None  
  
  
def neuton\_method(function1, function2, x0, y0, e):  
 iter\_counter = 1  
 last\_x = x0  
 last\_y = y0  
  
 jackobian = utils.calculate\_jackobian(function1, function2, last\_x, last\_y)  
 del\_x = function1(last\_x, last\_y) / jackobian  
 del\_y = function2(last\_x, last\_y) / jackobian  
  
 curr\_x = last\_x - del\_x \* utils.calculate\_det\_y(function2, last\_x, last\_y) + del\_y \* utils.calculate\_det\_y(  
 function1, last\_x, last\_y)  
  
 curr\_y = last\_y + del\_x \* utils.calculate\_det\_x(function2, last\_x, last\_y) - del\_y \* utils.calculate\_det\_x(  
 function1, last\_x, last\_y)  
  
 while abs(curr\_x - last\_x) > e or abs(curr\_y - last\_y) > e:  
 iter\_counter += 1  
 last\_x = curr\_x  
 last\_y = curr\_y  
 jackobian = utils.calculate\_jackobian(function1, function2, last\_x, last\_y)  
 del\_x = function1(last\_x, last\_y) / jackobian  
 del\_y = function2(last\_x, last\_y) / jackobian  
  
 curr\_x = last\_x - del\_x \* utils.calculate\_det\_y(function2, last\_x, last\_y) + del\_y \* utils.calculate\_det\_y(  
 function1, last\_x, last\_y)  
  
 curr\_y = last\_y + del\_x \* utils.calculate\_det\_x(function2, last\_x, last\_y) - del\_y \* utils.calculate\_det\_x(  
 function1, last\_x, last\_y)  
  
 if iter\_counter > 200:  
 return None, None, None, None, None  
  
 return curr\_x, curr\_y, abs(curr\_x - last\_x), abs(curr\_y - last\_y), iter\_counter

# Примеры и результаты работы программы

Изображение выглядит как текст, устройство, счетчик

Автоматически созданное описание

**Чтение из файла:**

Изображение выглядит как текст

Автоматически созданное описание

**Ввод данных вручную:**

Изображение выглядит как текст

Автоматически созданное описание

# Выводы

В данной лабораторной работе я:

* Ознакомился с вычислительными методами решения неоднородных уравнений­ ­– методом Ньютона, методом половинного деления, методом простой итерации, методом хорд;
* Ознакомился с вычислительным методом решения неоднородных систем уравнений­ ­– методом Ньютона;
* Ознакомился с принципами итерационных методов вычисления корней неоднородных уравнений (в частности, с принципом уточнения до определенной погрешности .