Университет ИТМО

Факультет ФПИ и КТ

### Отчёт

### по лабораторной работе 2

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

Вариант 11

Студент:

Преподаватель:  
Малышева Татьяна Алексеевна

**Вычислительная реализация задачи:**

### Задача:

Метод половинного деления:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № шага | a | b | x | f(a) | f(b) | f(x) | |a-b| |
| 1 | 1 | 2 | 1.5 | -5.53 | 39.43 | 9.99125 | 1 |
| 2 | 1 | 1.5 | 1.25 | -5.53 | 9.99125 | 0.69953125 | 0.5 |
| 3 | 1 | 1.25 | 1.125 | -5.53 | 0.69953125 | -2.771933594 | 0.25 |
| 4 | 1.125 | 1.25 | 1.1875 | -2.771933594 | 0.69953125 | -1.128635254 | 0.125 |
| 5 | 1.1875 | 1.25 | 1.21875 | -1.128635254 | 0.69953125 | -0.2380679321 | 0.0625 |
| 6 | 1.21875 | 1.25 | 1.234375 | -0.2380679321 | 0.69953125 | 0.2248017502 | 0.03125 |
| 7 | 1.21875 | 1.234375 | 1.2265625 | -0.2380679321 | 0.2248017502 | -0.008109202385 | 0.015625 |
| 8 | 1.2265625 | 1.234375 | 1.23046875 | -0.008109202385 | 0.2248017502 | 0.1079764503 | 0.0078125 |

Метод хорд:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № шага | a | b | x | f(a) | f(b) | f(x) | |a-b| |
| 1 | -3 | -2 | -2.187012263 | -29.17 | 6.71 | 3.675109992 | 1 |
| 2 | -2.187012263 | -2 | -2.413475375 | 3.675109992 | 6.71 | -2.019015195 | 0.187012 |
| 3 | -2.413475375 | -2 | -2.317838806 | -2.019015195 | 6.71 | 0.673122037 | 0.413475 |
| 4 | -2.317838806 | -2 | -2.353278367 | 0.673122037 | 6.71 | -0.273894084 | 0.317839 |
| 5 | -2.353278367 | -2 | -2.33942351 | -0.273894084 | 6.71 | 0.1033294 | 0.353278 |
| 6 | -2.33942351 | -2 | -2.344732149 | 0.1033294 | 6.71 | -0.040141404 | 0.339424 |
| 7 | -2.344732149 | -2 | -2.342682113 | -0.040141404 | 6.71 | 0.015419401 | 0.344732 |
| 8 | -2.342682113 | -2 | -2.343471401 | 0.015419401 | 6.71 | -0.005948805 | 0.342682 |

Метод простой итерации:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № шага | xk | f(xk ) | xk+1 |  | │xk − xk+1│ |
| 1 | -0.6 | -0.5476 | -0.656923077 | -0.656923077 | 0.056923077 |
| 2 | -0.656923077 | 0.258441945 | -0.63005801 | -0.63005801 | 0.026865067 |
| 3 | -0.63005801 | -0.121498617 | -0.642687804 | -0.642687804 | 0.012629794 |
| 4 | -0.642687804 | 0.057258579 | -0.636735769 | -0.636735769 | 0.005952035 |
| 5 | -0.636735769 | -0.026956902 | -0.639537941 | -0.639537941 | 0.002802173 |
| 6 | -0.639537941 | 0.012697541 | -0.638218031 | -0.638218031 | 0.001319911 |
| 7 | -0.638218031 | -0.00597955 | -0.638839605 | -0.638839605 | 0.000621575 |

### Цель

Научиться как реализовать процесс решения нелинейных уравнений и систем нелинейных уравнений.

### Описание программ

Процесс программы:

**日程表

描述已自动生成**

Ключевые класс:

1. Solvers

图形用户界面, 文本, 应用程序, 聊天或短信

描述已自动生成

Они выполняет решение функций. Ключевой метод - solve(). Draw() рисует графку

SimpleIterate:

def solve(self, inputs):  
 input1 = inputs[0]  
 input2 = inputs[1]  
 self.accuracy = float(input2)  
 times = 0  
 ans: float  
 caculateaccuracy = 10  
 model = 0  
 start = float(input1)  
 if abs(self.myfunction.getpossibilityder1(start)) < 1:  
 print("Possibility 1 is running")  
 model = 1  
 elif abs(self.myfunction.getpossibilityder2(start)) < 1:  
 print("Possibility 2 is running")  
 model = 2  
 elif abs(self.myfunction.getpossibilityder3(start)) < 1:  
 print("Possibility 3 is running")  
 model = 3  
 else:  
 raise InputError("This point can't be a start point cause it doesn't meet the convergence requirements")  
 print("-------------------Method Simple iterate-------------------")  
 print("%-10s %-10s %-10s %-10s %-10s" % ("x(k)", "f(x(k))", "x(k+1)", "φ(x(k))", "|x(k)-x(k+1)|"))  
 while caculateaccuracy >= self.accuracy:  
 self.a = start  
 times = times + 1  
 f1 = self.myfunction.getvalue(self.a)  
 if model == 1:  
 start = self.myfunction.getpossibilityv1(start)  
 if abs(self.myfunction.getpossibilityder1(start)) >= 1:  
 raise InputError("During the process we got a point doesnt meet requirement at %d iterate" % times)  
 elif model == 2:  
 start = self.myfunction.getpossibilityv2(start)  
 if abs(self.myfunction.getpossibilityder2(start)) >= 1:  
 raise InputError("During the process we got a point doesnt meet requirement at %d iterate" % times)  
 elif model == 3:  
 start = self.myfunction.getpossibilityv3(start)  
 if abs(self.myfunction.getpossibilityder3(start)) >= 1:  
 raise InputError("During the process we got a point doesnt meet requirement at %d iterate" % times)  
 self.b = start  
 caculateaccuracy = abs(self.a - self.b)  
 print("%-10.4f %-10.4f %-10.4f %-10.4f %-10.4f" %  
 (self.a, f1, self.b, self.b, caculateaccuracy))  
 print("-----------------------------------------------------------")  
 print("Iterate times:%d" % times)  
 print("Answer is %.4f" % self.b)  
 print("Accuracy is %.4f" % caculateaccuracy)  
 print("F(x) of answer is:%.4f" % self.myfunction.getvalue(self.b))  
 self.myfunction.draw(int(self.a-1), int(self.b+1))

ScantSolver:

def solve(self, inputs):  
 input1 = inputs[0]  
 input2 = inputs[1]  
 input3 = inputs[2]  
 self.a = float(input1)  
 self.b = float(input2)  
 self.accuracy = float(input3)  
 if self.myfunction.getvalue(self.a)\*self.myfunction.getvalue(self.b) >= 0:  
 raise InputError("In this range no solution")  
 answer: float  
 times = 0  
 print("-------------------Method Secant Start-------------------")  
 caculateaccuracy = 10  
 print("%-10s %-10s %-10s %-10s %-10s %-10s %-10s" % ("x(k-1)", "f(x(k-1))", "x(k)",  
 "f(x(k))", "x(k+1)", "f(x(k+1))", "|x(k)-x(k+1)|"))  
 while caculateaccuracy >= self.accuracy:  
 times = times + 1  
 printa = self.a  
 printb = self.b  
 f1 = self.myfunction.getvalue(self.a)  
 f2 = self.myfunction.getvalue(self.b)  
 nextpoint = self.getnextpoint()  
 temp = self.b  
 self.b = nextpoint  
 self.a = temp  
 caculateaccuracy = abs(self.a - self.b)  
 printc = self.b  
 f3 = self.myfunction.getvalue(self.b)  
 answer = printc  
 print("%-10.4f %-10.4f %-10.4f %-10.4f %-10.4f %-10.4f %-10.4f" %  
 (printa, f1, printb, f2, printc, f3, caculateaccuracy))  
 print("-----------------------------------------------------------")  
 print("The answer is:%.4f" % answer)  
 print("The error is:%.4f" % caculateaccuracy)  
 print("Iterated %d" % times)  
 print("F(x) of answer:%.4f" % self.myfunction.getvalue(answer))  
 self.myfunction.draw(int(self.a-1), int(self.b+1))

SolverForSystem

def solve(self, inputs):  
 input1 = inputs[0]  
 input2 = inputs[1]  
 sumder1 = abs(self.firstderonx(float(input1))) + abs(self.firstderony(float(input2)))  
 sumder2 = abs(self.secondderonx(float(input1), float(input2))) + abs(self.secondderony(float(input1)))  
 if sumder1 < 1 and sumder2 < 1:  
 self.a = float(input1)  
 self.b = float(input2)  
 input3 = inputs[2]  
 self.accuracy = float(input3)  
 caculateaccuracy = 10  
 times = 0  
 print("-------------------Method Simple iterate(for system)-------------------")  
 print("%-10s %-10s %-10s %-10s %-10s" % ("x1", "x2", "φ(x1)", "φ(x2)", "max(|xi(k+1)-xi(k)|)"))  
 while caculateaccuracy >= self.accuracy:  
 times = times +1  
 tempa = self.a  
 tempb = self.b  
 self.a = self.firstiterator(tempa, tempb)  
 self.b = self.seconditerator(tempa, tempb)  
 caculateaccuracy = max(abs(tempb - self.b), abs(tempa - self.a))  
 print("%-10.4f %-10.4f %-10.4f %-10.4f %-10.4f" % (tempa, tempb, self.a, self.b, caculateaccuracy))  
 print("------------------------------------------------------------------------")  
 print("Iterated %d times" % times)  
 print("Result for the first equation: %.4f" % self.getvalue1())  
 print("Result for the second equation: %.4f" % self.getvalue2())  
 self.draw(int(self.b-1),int(self.b+1))  
 else:  
 raise InputError("Near this point no solution")

1. Function

日程表

描述已自动生成

Базовые процесс функции. a, b, c ,d можем сам определить

class AbstractFunction(ABC):  
 a = 0  
 b = 0  
 c = 0  
 d = 0  
 type = "0"  
  
 def \_\_init\_\_(self, a, b, c, d):  
 self.a = a  
 self.b = b  
 self.c = c  
 self.d = d  
  
 @abstractmethod  
 def getfirstDer(self, x: float):  
 pass  
  
 @abstractmethod  
 def getvalue(self, x: float):  
 pass  
  
 # For method iterate，They are different ways to build x = φ(x) and φ'(x)  
 @abstractmethod  
 # φ'(x)  
 def getpossibilityder1(self, x: float):  
 pass  
  
 @abstractmethod  
 # φ(x)  
 def getpossibilityder2(self, x: float):  
 pass  
  
 @abstractmethod  
 def getpossibilityder3(self, x: float):  
 pass  
  
 @abstractmethod  
 def getpossibilityv1(self, x: float):  
 pass  
  
 @abstractmethod  
 def getpossibilityv2(self, x: float):  
 pass  
  
 @abstractmethod  
 def getpossibilityv3(self, x: float):  
 pass  
  
 @abstractmethod  
 def draw(self,a,b):  
 pass

### Примеры результатов

1.

图表

描述已自动生成

文本

描述已自动生成

2.

图表, 折线图

描述已自动生成

文本

描述已自动生成矩形

描述已自动生成

3.

图表, 折线图

描述已自动生成

文本

描述已自动生成

图形用户界面, 文本

描述已自动生成

4.

图表, 折线图

描述已自动生成

文本

描述已自动生成文本

中度可信度描述已自动生成

### Вывод

Нам сложно реализовать программу, которая может прямо найти конкретное решение нелинейного уравнения и систем нелинейного уравнение. Но как знаем, компьютер хорошо делает повторяемые работы. Поэтому мы можем реализовать итерацию, при которой компьютер приближается к точному решению и нам легче реализовать.

В процессе реализаций, я создал AbstractFunction, чтобы моя программа не только решала какое уравнение, а именно какой вид уравнения.