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

УНИВЕРСИТЕТ ИТМО

Системное и прикладное программное обеспечение.
Программная инженерия.

Лабораторная работа №2.

Дисциплина: Вычислительная математика.
Преподаватель: Малышева Татьяна Алексеевна.
Выполнил: Бусыгин Иван.
Группа: P3212.
Вариант: 6.

Санкт-Петербург
2022 год

Цель работы.

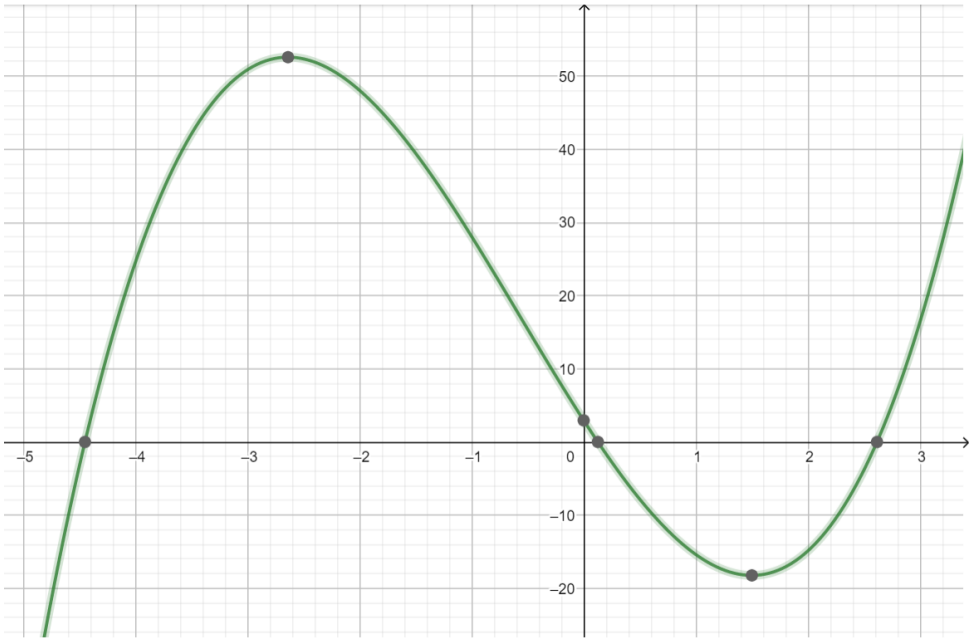
Изучить численные методы решения нелинейных уравнений и их систем, найти корни заданного нелинейного уравнения, выполнить программную реализацию методов.

Часть первая. Решение уравнения “руками”.

Задание:

Имеется кубический многочлен: $2x^3 + 3.41x^2 - 23.74x + 2.95$

Необходимо графически отделить корни и уточнить их с точностью $\varepsilon = 0.01$, используя следующие методы: половинного деления для левого корня, простой итерации для центрального, Ньютона для правого.



Левый корень:

№	a	b	x	$f(a)$	$f(b)$	$f(x)$	$\lvert a-b\rvert$
1	-5.00	-4.00	-4.5	-43.1	24.47	-3.42	1.00
2	-4.50	-4.00	-4.25	-3.42	24.47	11.91	0.50
3	-4.50	-4.25	-4.375	-3.42	11.91	4.60	0.25
4	-4.50	-4.375	-4.438	-3.42	4.60	0.65	0.125
5	-4.50	-4.438	-4.469	-3.42	0.65	-1.36	0.062
6	-4.469	-4.438	-4.454	-1.36	0.65	-0.38	0.031
7	-4.454	-4.438	-4.446	-0.38	0.65	0.14	0.016
8	-4.454	-4.446	-4.450	-0.38	0.14	-0.12	0.008

Результат: -4.450 ± 0.004

Центральный корень:

Чтобы производная была в пределах от -1 до 1, поделим многочлен на 30:

$$f(x) = \frac{2x^3 + 3.41x^2 - 23.74x + 2.95}{30}$$

$$x_{i+1} = \varphi(x_i) = \frac{2x_i^3 + 3.41x_i^2 - 22.74x_i + 2.95}{30}$$

i	x_i	$f(x_i)$	$\varphi(x_i)$	$\lvert x_{i+1} - x_i \rvert$
1	0.000	0.098	0.098	0.098
2	0.098	0.022	0.120	0.022
3	0.120	0.005	0.125	0.005

Результат: 0.125

Правый корень:

№	x_i	$f(x_i)$	$f'(x_i)$	x_{i+1}	$\lvert x_{i+1} - x_i \rvert$
1	3.000	16.420	50.720	2.676	0.324
2	2.676	2.166	37.476	2.618	0.058
3	2.618	0.058	35.24	2.616	0.002

Результат: 2.216

Часть вторая. Программа, ищущая корень выбранной функции.

Код основной функции (метод хорд):

```
def solveEquation(func, a, b, admissibleError):
  fa = func(a)
  fb = func(b)
  if fa * fb > 0:
    print('На концах отрезака функция имеет одинаковый знак. Поиск корня
невозможен. Его и вовсе может не быть.')
    finish()
  if fa == 0:
    return (a, fa, 0)
  if fb == 0:
    return (b, fb, 0)

  iters = 1

  while True:
    x = (a * fb - b * fa) / (fb - fa)
    fx = func(x)
    if b - a < admissibleError:
      return (x, fx, iters)
    if fx == 0:
      return (x, fx, iters)
    if fa * fx < 0:
      b = x
    else:
      a = x
    iters += 1
```

Пример работы программы.

```
Желаете ли вы использовать файл с входными данными? Если да, введите 'y': n

Есть три уравнения:
 1. cos(x) - x = 0
 2. x³ - 3,78x² + 1,25x + 3,49 = 0
 3. e^x - x³

Введите номер уравнения, корень которого необходимо найти: 1
Введите точность (погрешность) вычислений: 0.00001
Введите левую границу интервала, содержащего корень: 0
Введите правую границу интервала, содержащего корень: 1

Найденный корень уравнения: 0.7390856757740472
Значение функции в этой точке: -9.080331878630332e-07
Число осуществлённых итераций при поиске корня: 22

Программа завершила работу.
```

Часть третья. Программа, ищущая корень системы уравнений.

Код основной функции (метод Ньютона):

```
def solveSystem(point, admissibleError, f, fDerX, fDerY, g, gDerX, gDerY):
  if f(point) == 0 and g(point) == 0:
    return (0, point, (0, 0))
  iter = 1
  while True:
    matrix = np.array(
      ((fDerX(point), fDerY(point)), (gDerX(point), gDerY(point)))
    )
    if np.linalg.det(matrix) == 0:
      print('Ошибка! Не удалось вычислить очередное приближение корня из-за
его бесконечности или неоднозначности.')
      finish()
    changesVec = -np.linalg.inv(matrix) @ np.array((f(point), g(point)))
    point += changesVec
    changesVec = np.abs(changesVec)
    if np.max(changesVec) < admissibleError:
      return (iter, point, changesVec)
    iter += 1
```

Пример работы программы.

```
Это программа для нахождения корней системы двух уравнений:
 1.8x³ - 2.47y² - 5.53x + 1.539 = 0
 2x³ + 3.41x² - 23.74y + 2.95 = 0

Введите точность (погрешность) вычислений: 0.00001
Введите через запятую две координаты начальной точки: 0, 0

Найденный корень: [0.27681421 0.13705634]
Вектор величин изменений на последней итерации: [1.542679e-09 1.437891e-09]
Число итераций: 4

Программа завершила работу.
```

Вывод.

Изучил различные методы приближённого нахождения корней нелинейных уравнений и систем, реализовал некоторые из них программно.