Содержание

Лабораторная работа 3

Теоретическая часть

Метод итераций

Дана система:

$$\Big\{ f_1(x_1, x_2, \dots, x_n) = 0 \; f_2(x_1, x_2, \dots, x_n) = 0 \; \dots \; f_n(x_1, x_2, \dots, x_n) = 0$$

В векторном виде:

$$F(X) = \Big(f_1(x_1, x_2, \dots, x_n) \; f_2(x_1, x_2, \dots, x_n) \; \dots \; f_n(x_1, x_2, \dots, x_n) \Big) = 0$$

Пусть каждая $f_i, \; i = 1, 2, \dots, n$ действительна, определена и непрерывна в некоторой области ω, в которой так же находится решение системы.

Решение данной системы будем искать следующим итерационным процессом:

$$X^{(p+1)} = X^{(p)} + \lambda F(X^{(p)})$$

Где:

$$\lambda = -\Big[F'(X^{(0)}) \Big]^{-1}$$

$X^{(0)}$ — первое приближение решения исходной системы.

Метод Ньютона

$$F(X) = 0 \; F = \Big(f_1 \; f_2 \; \cdots \; f_n \Big) \; X = \Big(x_1 \; x_2 \; \cdots \; x_n \Big) \tag{1}$$

Предположим, что найдено некоторое приближённое решение системы $X^{(p)}$.

$$X = X^{(p)} + \varepsilon^{(p)}$$

Тогда:

$$F(X^{(p)} + \varepsilon^{(p)}) = 0$$

Мы можем разложить левую часть этого тождества по степеням ε, ограничиваясь линейными членами:

$$F(X^{(p)} + \varepsilon^{(p)}) = F(X^{(p)}) + F'(X^{(p)})\varepsilon^{(p)}$$

Это можно представить в развёрнутом виде:

$$f_1(x_1^{(p)} + \varepsilon_1^{(p)}, x_1^{(p)} + \varepsilon_2^{(p)}, \dots, x_n^{(p)} + \varepsilon_n^{(p)}) = f_1(x_1^{(p)}, x_1^{(p)}, \dots, x_n^{(p)}) + f'1, x_1(x_1^{(p)}, x_1^{(p)}, \dots, x_n^{(p)}) \cdot \varepsilon_1^{(p)} + f'1, x_2(x_1^{(p)}, x_1^{(p)}, \dots, x_n^{(p)}) \cdot \varepsilon_2^{(p)} + \cdots f'1, x_n(x_1^{(p)}, x_1^{(p)}, \dots, x_n^{(p)}) \cdot \varepsilon_n^{(p)} = 0 \; f_2(x_1^{(p)} + \varepsilon_1^{(p)}, x_1^{(p)} + \varepsilon_2^{(p)}, \dots, x_n^{(p)} + \varepsilon_n^{(p)}) = f_2(x_1^{(p)}, x_1^{(p)}, \dots, x_n^{(p)}) + f'2, x_1(x_1^{(p)}, x_1^{(p)}, \dots, x_n^{(p)}) \cdot \varepsilon_1^{(p)} + f'2, x_2(x_1^{(p)}, x_1^{(p)}, \dots, x_n^{(p)}) \cdot \varepsilon_2^{(p)} + \cdots f'2, x_n(x_1^{(p)}, x_1^{(p)}, \dots, x_n^{(p)}) \cdot \varepsilon_n^{(p)} = 0 \; \cdots \; f_n(x_1^{(p)} + \varepsilon_1^{(p)}, x_1^{(p)} + \varepsilon_2^{(p)}, \dots, x_n^{(p)} + \varepsilon_n^{(p)}) = f_n(x_1^{(p)}, x_1^{(p)}, \dots, x_n^{(p)}) + f'n, x_1(x_1^{(p)}, x_1^{(p)}, \dots, x_n^{(p)}) \cdot \varepsilon_1^{(p)} + f'n, x_2(x_1^{(p)}, x_1^{(p)}, \dots, x_n^{(p)}) \cdot \varepsilon_2^{(p)} + \cdots f'_{n,x_n}(x_1^{(p)}, x_1^{(p)}, \dots, x_n^{(p)}) \cdot \varepsilon_n^{(p)} = 0$$

(1)

Тогда получаем матрицу Якоби:

$$W(X) = F'(X) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \dots & \frac{\partial f_1}{\partial x_n} & \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \dots & \frac{\partial f_2}{\partial x_n} & \dots & \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \dots & \frac{\partial f_n}{\partial x_n} \end{pmatrix}$$

Или, кратко:

$$F'(X) = W(X) = \left[\frac{\partial f_i}{\partial x_j}\right],\ (i, j = 1 \dots n)$$

Из (1), мы получили линейную систему относительно $\varepsilon^{(p)}$:

$$F(X^{(p)}) + W(^{(p)}) \cdot \varepsilon^{(p)} = 0$$

Для нахождения $\varepsilon^{(p)}$ решим систему:

$$\varepsilon^{(p)} = -W^{-1}(X^{(p)}) \cdot F(X^{(p)})$$

Если найдём $\varepsilon^{(p)}$, то:

$$X^{(p+1)} = X^{(p)} - W^{-1}(X^{(p)}) \cdot F(X^{(p)}),\ (p = 0, 1, 2 \dots)$$

(2)

Равенство (2) — метод Ньютона.

Практическая часть

Задача

Решить систему нелинейных уравнений

Методом Ньютона

$$\begin{cases} \cos(x_1) + x_2 = 1.5 \\ 2x_1 - \sin(x_2 - 0.5) = 1 \end{cases}$$

С точностью 0.001

Предложенная система нелинейных уравнений записывается в виде отдельных функций, как показано в **Приложении 1**.

Главная функция программы - **Newton()**. Она следит за выполнением условия $|X^0 - X^n| < \varepsilon$, проверяет существование обратной матрицы и вызывает вспомогательные функции, которые необходимы для расчетов.

Матрица Якоби

$$\begin{bmatrix} -\sin(x_1) & 1 \\ 2 & -\cos(x_2 - 0.5) \end{bmatrix}$$

Функция **check_result()** - подставляет полученные функцией **Newton()** значения в исходную систему для проверки равенств.

При запуске программы, мы получим следующий вывод на экран:

After substitution into system: -9.498364037519025e-06 0.0007243978077879909
Check Newton OK!
Solution 1 is: [0.5824167 0.6648547]
After substitution into system: -1.614166890373525e-05 -0.0004806001040225105
Check iteration result OK!
Solution 2 is: [0.80116765 -0.59803829]

Первая строка это решение системы, вторая строка - результат подставления полученого решения в исходную систему. Третья строка - проверка полученого решения.

Решить систему нелинейных уравнений

Методом итераций

$$\begin{cases} \sin(x+y) = 1.5 * x - 1 \\ x^2 + y^2 = 1 \end{cases}$$

С точностью 0.001

Функция из **Приложения 3** решает систему нелинейных уравнений методом итераций с точностью **0.001**.

Алгоритм преобразует систему $F(x) = 0$ к виду $x_1 = \varphi(x_1, ..., x_n)$ с помощью замены системы вида $F(x) = 0$ системой $x = x + \Lambda F(x)$, где в качестве Λ используется матрица вида $-W^{-1}(x^{(0)})$. где $W(x)$ - матрица Якоби.

Для выбора начального приближения найдем координаты точек пересечения кривых, соответствующих первому и второму уравнениям

$X_0 = [0.8, -0.4]$

Полученная матрица Якоби с подставленным первым приближением:

[[-0.57893901 0.92106099]
 [1.6   -0.8   ]]

Алгоритм проверяет, что определитель матрицы $W(x)$ не равен 0.

Det = -1.0105463856069243

Видим, что определитель матрицы Якоби не равен 0, получим обратную матрицу.

Inv matrix =
[[-0.79165094 -0.91144851]
 [-1.58330189 -0.57289701]]

Далее запускается итерационный процесс, который остановится при выполнении условия

$$|X^0 - X^n| < \varepsilon$$

После нахождения решения, программа подставляет найденные значения в исходную систему, после запуска программы на экране можно уидеть

Solving system by ITERATION METHOD
Det = -1.0105463856069243
Solution by Iteration method is: [0.80160075 -0.59774184]
After substitution into system: 4.8702043651926985e-05 -0.0001409209542611034
Check iteration result OK!

Приложение 1

```
from Newton import Newton, check_result
from Iteration import Iteration, check_iteration
from math import cos, sin, sqrt

F1 = [lambda x,y: cos(x) + y - 1.5, lambda x,y: 2*x - sin(y - 0.5) - 1]
F_d1 = [
  [lambda x,y: -sin(x), lambda x,y: 1],
  [lambda x,y: 2, lambda x,y: -cos(y - 0.5)]
]

F2 = [lambda x,y: sin(x+y) - 1.5*x + 1, lambda x, y: x**2 + y**2 - 1]
F_d2 = [
  [lambda x,y: cos(x+y) - 1.5, lambda x,y: cos(x+y)],
  [lambda x,y: 2*x, lambda x,y: 2*y]
]

if __name__ == '__main__':
  result = Newton([3.4, 3.2], 0.001, F1, F_d1)
  result1 = Iteration([0.5, 0.1], 0.001, F2, F_d2)
```

Приложение 2

```
from math import cos, sin
from numpy.linalg import det, inv
from numpy import array

def Newton(X0, e, F, F_d):
  print("Solving system by NEWTON METHOD")
  yako_m = F_d
  X0 = array(X0)
  XN = array([0,0])
  while all([abs(i) >= e for i in X0 - XN]):
    Fx_0 = array([f(*X0) for f in F])
    Wx_0 = [[0 for y in range(len(F))] for x in range(len(F))]
    for i in range(len(F)):
      for j in range(len(F)):
        Wx_0[i][j] = yako_m[i][j](*X0)
    det_ = det(Wx_0)
    if not det_:
      print("Определитель равен 0, обратная матрица не существует")
      return
    inv_ = array(inv(Wx_0))
    XN = X0.copy()
    X0 = X0 - (inv_.dot(Fx_0))
  print("Solution by Newton method is:", XN)
  check_result(XN, F)
```

```
    return XN


def check_result(result, F):
    r1 = F[0](*result)
    r2 = F[1](*result)
    print('After substitution into system:', r1, r2)
    if r1 < 0.001 and r2 < 0.001:
        print("Check Newton OK!")
    else:
        print("Check Newton FAIL!")
    print()
```

Приложение 3

```
from numpy.linalg import det, inv
from numpy import array

def Iteration(X0, e, F, F_d):
    print("Solving system by ITERATION METHOD")
    X0 = array(X0)
    X1 = X0.copy()
    l = array([
        [F_d[0][0](*X0), F_d[0][1](*X0)],
        [F_d[1][0](*X0), F_d[1][1](*X0)]
    ])
    if not det(l):
        print("Determinant = 0, inv matrix does not exist!")
        return
    print("Det =", det(l))
    lambda_ = -inv(array(l))
    print("Inv matrix =", lambda_)
    while True:
        X0 = X1.copy()
        X1 = X0 + lambda_.dot(array([F[0](*X0), F[1](*X0)]))
        if all([abs(i) <= e for i in X0 - X1]):
            break
    print("Solution by Iteration method is:", X1)
    print("Check iteration result", check_iteration(X1, F))
    print()
    return X1


def check_iteration(X1, F):
    r1 = F[0](*X1)
    r2 = F[1](*X1)
    print('After substitution into system:', r1, r2)
    return "OK!" if r1 < 0.001 and r2 < 0.001 else "FAIL!"
```