МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ Н. Э. БАУМАНА

Факультет информатики и систем управления

Кафедра теоретической информатики и компьютерных технологий

Лабораторная работа №3

по курсу «Моделирование»

«Сравнительный анализ методов решения систем нелинейных уравненний (метод Ньютона, метод простых итераций)»

Выполнил:

студент ИУ9-91

Выборнов А. И.

Руководитель:

Домрачева А. Б.

Москва 2015

1. Постановка задачи

Провести сравнительный анализ методов Ньютона и простых итераций для решения систем нелинейных уравнений. Реализовать оба метода.

2. Теоретическая часть

Дана система из n нелинейных уравнений и n переменных:

$$\begin{cases} f_1(x_1, x_2, ..., x_n) = 0 \\ f_2(x_1, x_2, ..., x_n) = 0 \\ ... \\ f_n(x_1, x_2, ..., x_n) = 0 \end{cases}$$

В системе $f_i(x_1, ..., x_n)$: $\mathbb{R}^n \to \mathbb{R}$ некоторые нелинейные функции. Для удобства в дальнейшем будем записывает систему в виде: $Fx = 0$, где $F(x) = [f_1(x), ..., f_n(x)]^T$ и $x = (x_1, x_n)^T$.

Требуется найти вектор $x^* = (x_1^*, ..., x_n^*)^T$, который при подстановке в систему превращает каждое уравнение в верное равенство.

2.1. Метод простых итераций

Метод простых итераций — итерационный метод решения системы линейных уравнений.

Для решения системы нелинейных уравнений методом простых итераций необходимо систему $Fx = 0$ привести к виду:

$$\begin{cases} x_1 = \varphi_1(x_1, x_2, ..., x_n) \\ x_2 = \varphi_2(x_1, x_2, ..., x_n) \\ ... \\ x_n = \varphi_n(x_1, x_2, ..., x_n) \end{cases}$$

Для удобства запишем систему в виде $x = \Phi(x)$, где $\Phi(x) = [\varphi_1(x), ..., \varphi_n(x)]^T$. Построим итеративный процесс с помощью формулы: $x^{(k+1)} = \Phi(x^{(k)})$.
Если $||x^{(k+1)} - x^{(k)}|| < \varepsilon$, то заканчиваем процесс итерирования.

2.1.1. Преимущества

- За одну итерацию выполняется многим меньше вычислений, чем в методе Ньютона.
- Простота реализации.

2.1.2. Недостатки

- Отсутствие сходимости для многих задач.
- Необходимо выбирать достаточно близкое к ожидаемому ответу начальное приближение, чтобы данный метод сходился.

2.2. Метод Ньютона

Метод Ньютона — итерационный метод решения системы нелинейных уравнений.

Начинаем с вектора $x^(0)$. Затем строим итеративный процесс, основываясь на формуле $x^{(k+1)} = x^{(k)} - J^{-1}(x^{(k)})F(x^{(k)})$, где J — якобиан матрицы F.

На практике вычисление обратной матрицы достаточно трудоёмкая операция, поэтому вышеприведённую формулу преобразовывают к виду $J(x^{(k)})\Delta x^{(k)} = -F(x^{(k)})$, где $\Delta x^{(k)} = x^{(k+1)} - x^{(k)}$. Полученную формулу можно решить как СЛАУ относительно $\Delta x^{(k)}$, следовательно по $x^{(k)}$ мы можем получить $x^{(k+1)}$.

2.2.1. Преимущества

- Быстрая сходимость — если матрица Якоби невырожденна, то метод имеет квадратичную сходимость из хорошего начального приближения.

2.2.2. Недостатки

- Необходимо задавать достаточно хорошее начальное приближение.
- Остутствие сходимости для многих задач.
- Необходимость вычисления матрицы Якоби.
- Необходимость решения СЛАУ на каждом шаге итерации.

3. Реализация

В рамках лабораторной работы была написана программа на языке python, которая реализует методы Ньютона и простых итераций

3.1. Метод Ньютона

Ниже представлена метод на Python, реализующий метод Ньютона. Метод принимает на вход матрицу F, вектор начальное приближения q и переменную, характеризующую точность вычислений *eps*, и возвращает вектор решения *result*.

```
def newton(F, q, eps):
    W = jacobian(F)
    result = q

    converge = False
    while not converge:
        F_subs = subs_2d(F, result)
        W_subs = subs_3d(W, result)

        #Wdx=-F
        dx = linalg.solve(np.array(W_subs), -np.array(F_subs))
        result = np.add(result, dx)

        converge = linalg.norm(dx) < eps
    return list(result)
```

3.2. Метод простых итераций

Ниже представлена метод на Python, реализующий метод простых итераций. Метод принимает на вход матрицу F, вектор начальное приближения q и переменную, характеризующую точность вычислений *eps*, и возвращает вектор решения *result*.

```
def fixed_point(F, q, eps):
    previous = q
    converge = False
    while not converge:
        result = subs_2d(F, previous)
```

```
        result = map(float, result)

        print previous, result
        delta = np.subtract(result, previous)
        previous = result

        converge = linalg.norm(delta) < eps
    return result
```

4. Выводы

Метод Ньютона сходится очень быстро, но каждая итерация работает относительно медленно, метод простых итерация, напротив, сходится не так быстро, но каждая итерация требует намного меньшего количества вычислений. Также необходимо отметить, что метод простых итераций проще в реализации.