

## Вычислительная физика, Осень 2020 ВШЭ. Задание 6.<sup>a</sup>

1. (10) Реализуйте метод простой итерации для нахождения решения следующих уравнений относительно  $x$ :

$$(i) \ 1 + \cos x = 0, \quad (ii) \ x^2 = 2.$$

Используйте следующие итерационные формулы:

$$(i) \ x_{k+1} = x_k + \frac{\cos x_k + 1}{\sin x_k}, \quad (ii) \ x_{k+1} = \frac{1}{2} \left( x_k + \frac{2}{x_k} \right).$$

В обоих случаях, стартуйте с  $x_0 = 1$ . Какова сходимость итераций (линейная/квадратичная) для случаев (i) и (ii)?

2. (15) Реализуйте алгоритм, который выполняет итерации Ньютона для заданной функции  $f(x)$  с известной производной  $f'(x)$ . Ваша функция должна находить корни  $f(x)$  с заданной точностью  $\epsilon$ . Заголовок функции должен иметь следующий вид:

---

```
def newton_iteration(f, fder, x0, eps=1e-5, maxiter=1000):
    """Newton's root finding method for f(x)=0
    Parameters
    -----
    f : callable
        Function f.
    fder : callable
        Derivative of f.
    x0 : float
        Initial point for iterations.
    eps : float
        Requested accuracy.
    maxiter : int
        Maximal number of iterations.

    Returns
    -----
    x : float
        Approximate root.
    niter : int
        Number of iterations.
    """
```

---

Протестируйте вашу функцию на примере  $f(x) = x^2 - 1$ . Постройте логарифм ошибки найденного решения от количества итераций. Какова сходимость метода (линейная или квадратичная)?

3. (15) Рассмотрите Ньютоновские итерации для системы двух нелинейных уравнений

$$x_1^2 - 2x_2^4 + 1 = 0, \quad x_1 - x_2^3 + 1 = 0.$$

Найдите явно выражение для  $\Delta x_k(x_k)$  (удобно сделать это в **Mathematica**). Реализуйте итерации Ньютона и найдите действительное решение этой системы в единичном круге на плоскости  $x_1, x_2$ .

4. (20) Реализуйте метод итераций для решения системы линейных уравнений (метод Якоби). Для этого перепишите уравнение  $Ax = b$ , выделив диагональную часть матрицы  $A$ :

$$A = D + (A - D),$$

---

<sup>a</sup> Дополнительно указаны: (количество баллов за задачу)[имя задачи на nbgrader]

в виде

$$x_{n+1} = Bx_n + c,$$

где  $B = D^{-1}(A - D)$ . Найдите  $c$ .

Создайте случайную матрицу с диагональным доминированием:

---

```
import numpy as np
rnd = np.random.RandomState(1234)
n = 10
A = rnd.uniform(size=(n, n)) + np.diag([15]*n)
b = rnd.uniform(size=n)
```

---

Вычислите норму соответствующей матрицы  $B$  и выполните итерации Якоби. Убедитесь, что результирующий вектор  $x$  действительно решает исходную систему.

Матрица  $A$ , с которой вы работали выше, по построению доминируется диагональю. Что произойдёт, если уменьшать величину диагональных элементов? Проверьте сходимость итераций Якоби (вычислите также норму матрицы  $B$ ).

5. (20) Напишите программу, которая решает нелинейное уравнение Пуассона:

$$\phi''(x) = e^{\phi(x)} - n(x), \quad \text{где } n(x) = 1 + e^{-3(x-5)^2},$$

в области  $0 \leq x \leq 10$  с граничными условиями  $\phi(0) = \phi(10) = 0$ . Для этого дискретизируйте дифференциальное уравнение на равномерную решётку  $x_{j=1,\dots,N-1}$ , так что значения потенциала в точках  $x_0 = 0$  и  $x_N = 10$  зафиксированы граничными условиями, а внутри определяются дискретной версией исходного дифференциального уравнения:  $G_1 = 0$ ,  $G_2 = 0$ , ...,  $G_{N-1} = 0$ , где

$$G_j = \frac{\phi_{j+1} - 2\phi_j + \phi_{j-1}}{\delta x^2} - e^{\phi_j} + n(x_j) = 0.$$

Используйте метод Ньютона для того, чтобы найти решение этой системы. Сколько итераций нужно, чтобы получить решение с 10ю значащими цифрами?

6. (20) Рассмотрим систему линейных уравнений, матрица правой части которой является ‘ленточной’ и имеет следующую структуру: ненулевые элементы расположены на трех центральных диагоналях и на двух ‘крыльях’. Матрицы такой структуры возникают, например, при решении задачи на нахождение электростатического потенциала  $\phi(x, y)$ , создаваемого двумерным распределением заряда  $\rho(x, y)$  при дискретизации на сетке уравнения Пуассона

$$\Delta\phi = -4\pi\rho$$

(детали см. напр. А.А. Самарский, А.В. Гулин, Численные методы, ч. 3 гл. 1, параграф 1).

Количество точек сетки (определяющее размер возникающей матрицы) растет с уменьшением шага сетки  $h$  как  $O(1/h^2)$ . Таким образом, приходится иметь дело с разреженными матрицами огромного размера. Нужную нам матрицу  $A$  можно создать следующим образом:

---

```
n = 5

a = np.zeros((n-1, n-1))
idx = np.arange(n-1)
a[idx, idx] = -4
a[idx[:-1], idx[:-1]+1] = 1
a[idx[1:], idx[1:]-1] = 1

A = block_diag(a, a, a, a, a)
idx = np.arange(A.shape[0])
A[idx[:-n+1], idx[:-n+1] + n-1] = 1
A[idx[n-1:], idx[n-1:] - n+1] = 1
```

---

Проинспектируйте получившуюся матрицу:

---

```
with np.printoptions(linewidth=99):  
    print(m)  
  
plt.matshow(m)
```

---

Вектор правой части задайте в виде:

---

```
b = np.zeros(A.shape[0])  
b[A.shape[0]//2] = -1
```

---

Составьте функцию, вычисляющую решение системы уравнений  $Ax = b$  методом Зейделя с заданной точностью  $\epsilon$ . Прокомментируйте зависимость числа итераций, требуемых для достижения заданной точности, от  $\epsilon$ .