

**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение высшего образования  
«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ  
УНИВЕРСИТЕТ имени Н.Э.БАУМАНА  
(национальный исследовательский университет)»**

Факультет: Информатика и системы управления

Кафедра: Теоретическая информатика и компьютерные технологии

**Лабораторная работа № 9**

«Изучение скорости сходимости однопараметрического метода»

по дисциплине «Численные методы линейной алгебры»

Работу выполнил

студент группы ИУ9-72Б

Жук Дмитрий

**Цель работы**

Изучить зависимость скорости сходимости однопараметрического метода в зависимости от значения .

**Реализация**

Был создан репозиторий на GitHub <https://github.com/ZhukDmitryOlegovich/num-methods>

Используя ранее реализованный функционал, а именно классы, представляющие из себя абстракцией квадратной матрицы и вектора. По аналогии с методом Зейделя, была создана похожая реализация метода.

Так как однопараметрический метод для своей работы требует значение , а для нахождения необходимо было найти собственное значение матрицы, использовалась библиотека mathjs, а именно её функция eigs, которая и находит собственные значение.

**Вывод**

В ходе выполнения лабораторной работы был реализован выше заявленные метод, а также был произведен анализ количества итераций в зависимости от , который показал, что наиболее благоприятным является .

Данную работу так же можно открыть и посмотреть по ссылке <https://zhukdmitryolegovich.github.io/num-methods/lab9/>.

**Приложение**

    singleParameterMethod(

        other: Vector<A>,

        option: { t: number; eps?: number; maxCount?: number; },

    ) {

        const { t, eps = 0, maxCount = Infinity } = option;

        const N = this.countColons();

        const E = new SquareMatrix(fromLength(N, (i) => fromLength(N, (j) => +(i === j))));

        const a = this;

        const f = other;

        let x = new Vector(fromLength(N, () => 0));

        let xBefore = x;

        let count = 0;

        do {

            xBefore = x;

            const P = E.add(a.mulN(-t));

            const g = f.mulN(t);

            x = Vector.fromMatrix(P.mul(x).add(g));

            count++;

        } while (

            count < maxCount

            && eps < Math.abs(Vector.fromMatrix(xBefore.add(x.mulN(-1))).norma())

        );

        return { count, result: x };

    }

