Лабораторная работа 6

Отчет по лабораторной работе 6

Милёхин Александр НПМмд-02-21

Содержание

[1 Цель работы 1](#_Toc96079518)

[2 Теоретические сведения 1](#_Toc96079519)

[3 Задание 1](#_Toc96079520)

[4 Выполнение лабораторной работы 1](#_Toc96079521)

[5 Выводы 8](#_Toc96079522)

# 1 Цель работы

Научиться работать в Octave с пределами, последовательностями и рядами, а также научиться писать векторизованный программный код.

# 2 Теоретические сведения

Вся теоретическая часть по выполнению лабораторной работы была взята из инструкции по лабораторной работе №5 (“Лабораторная работа №6. Описание”) на сайте: https://esystem.rudn.ru/course/view.php?id=12766

# 3 Задание

Выполните работу и задокументируйте процесс выполнения.

# 4 Выполнение лабораторной работы

**1. Пределы. Оценка**

Определяем с помощью анонимной функции простую функцию. Создаём индексную переменную, возьмём степени 10, и оценим нашу функцию. Показано на Fig. 1.

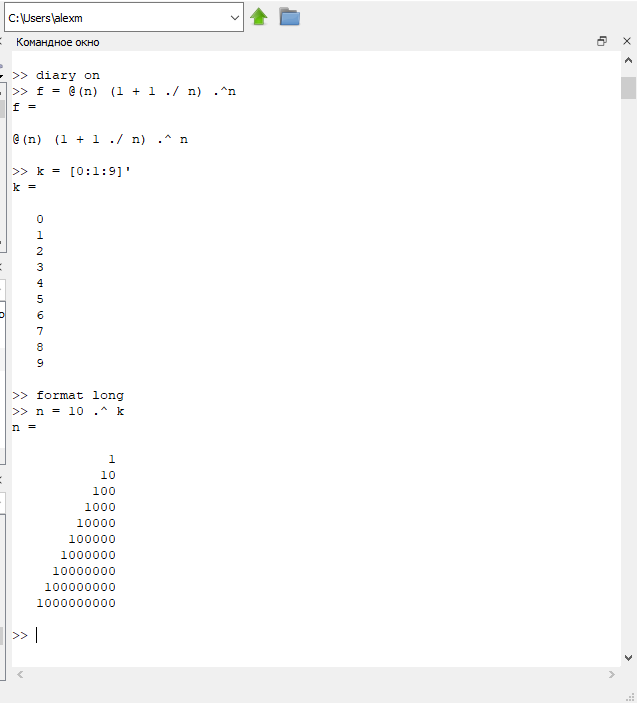


Figure 1: Промежуточные вычисления для расчета предела

Получим ответ. На Fig. 2 видно, что предел сходится к значению 2.71828.

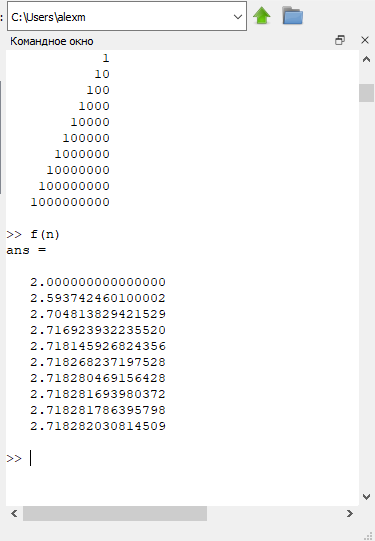


Figure 2: Искомый предел

**2. Частичные суммы**

Определим индексный вектор, а затем вычислим члены. После чего введем последовательность частичных сумм, используя цикл. Показано на Fig .3

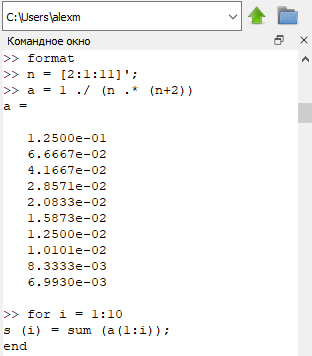
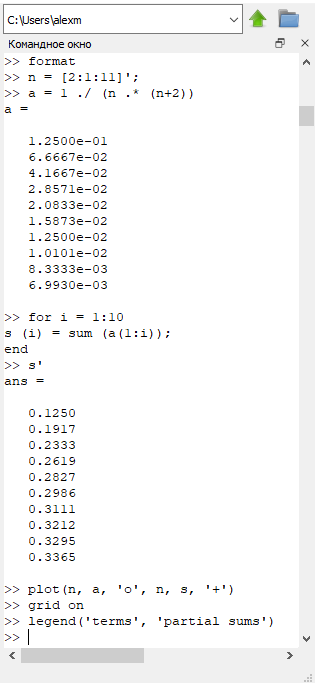


Figure 3: Частичные суммы

Построенные слагаемые и частичные суммы можно увидеть на Fig. 4.

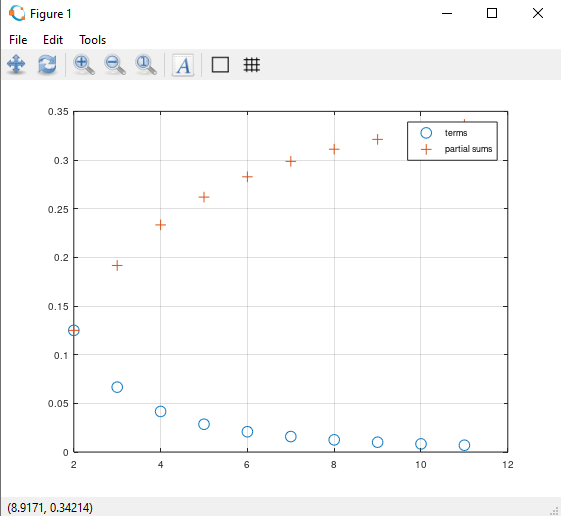


Figure 4: Графическое представление результатов

**3. Сумма ряда**

Найдём сумму первых 1000 членов гармонического ряда 1/n. Действия показаны на Fig. 5.

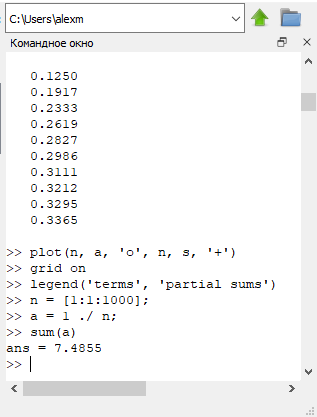


Figure 5: Сумма ряда

**4. Вычисление интегралов**

Численно посчитаем интеграл. См. Fig. 6.

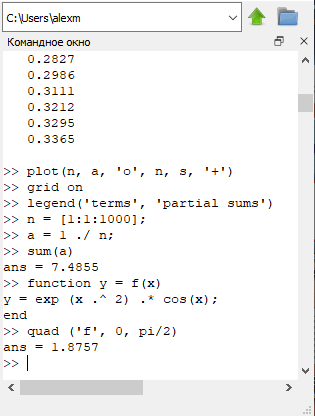


Figure 6: Интегрирование функции

**5. Аппроксимирование суммами**

Напишем скрипт для того, чтобы вычислить интеграл по правилу средней точки. Введём код в текстовый файл и назовём его midpoint.m. Скрипт показан на Fig. 7.

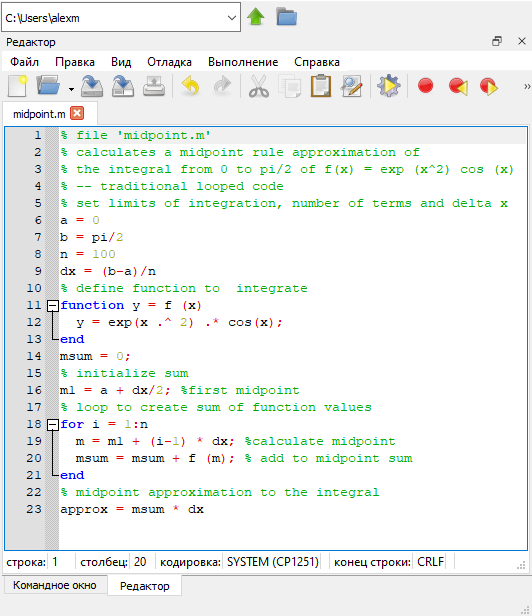


Figure 7: Содержание файла midpoint

Запустим этот файл в командной строке. Вывод см. на Fig. 8

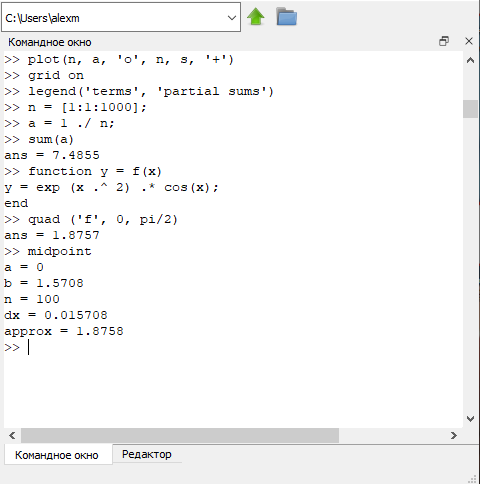


Figure 8: Результаты вывода

Теперь напишем векторизованный код, не требующий циклов. Для этого создадим вектор х-координат средних точек. Показано на Fig. 9.

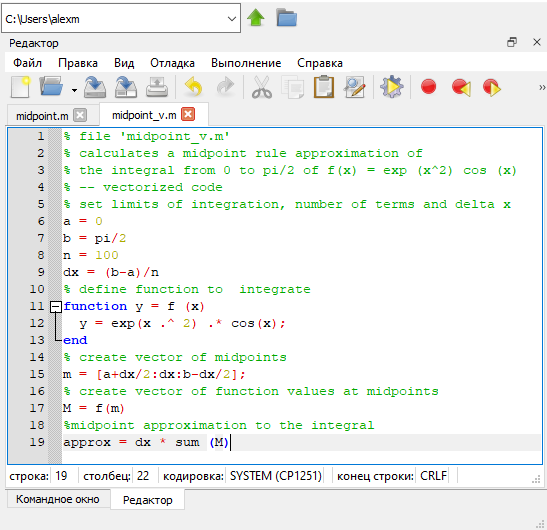


Figure 9: Содержание файла midpoint\_v

Запустим этот файл в командной строке. Вывод см. на Fig. 10

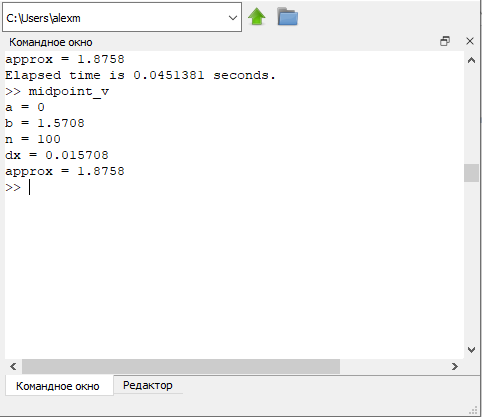


Figure 10: Вывод векторизованного кода программы

Запустив оба кода, можно заметить, что ответы совпадают, однако векторизованный код считает быстрее, так как в нём не использованы циклы, которые значительно замедляют работу программы. Сравнение показано на Fig. 11.

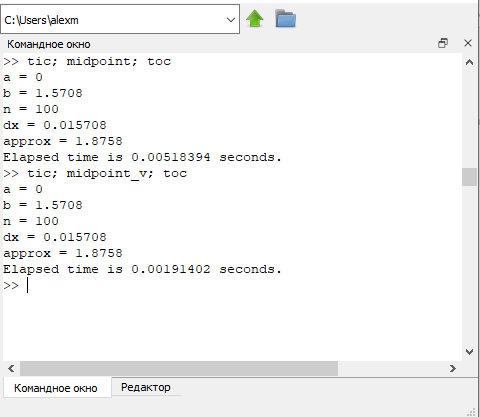


Figure 11: Сравнение полученных результатов

# 5 Выводы

Я научился работать в Octave с пределами, последовательностями и рядами, а также научился писать векторизованный программный код. Более того, я определил, что векторизованный код работает существенно быстрее, чем код с циклами.