Лабораторная работа 6

Отчет по лабораторной работе 6

Милёхин Александр НПМмд-02-21

Содержание

# Цель работы

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

# 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

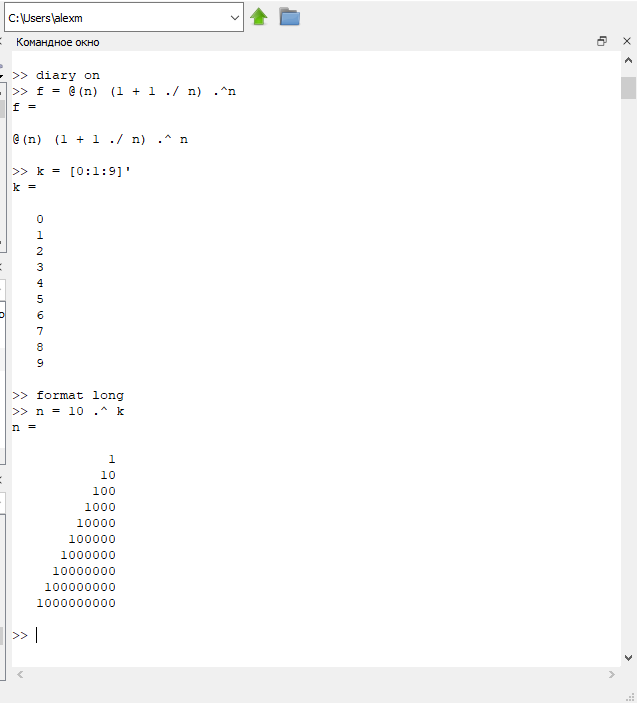
# Задание

Выполните работу и задокументируйте процесс выполнения.

# Выполнение лабораторной работы

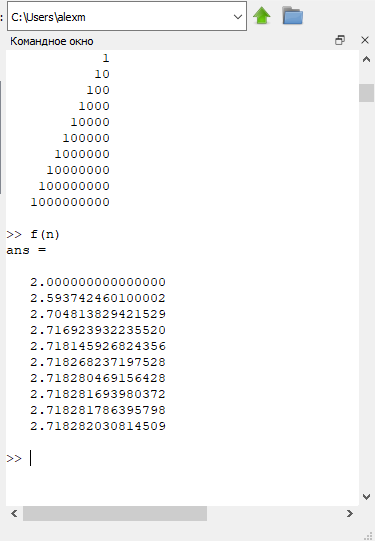
**1. Пределы. Оценка**

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



Промежуточные вычисления для рассчета предела

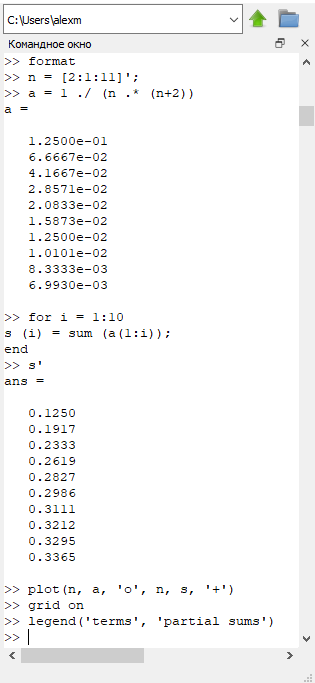
Получим ответ. На Fig. 2 видно, что предел сходится к значению 2.71828.



Искомый предел

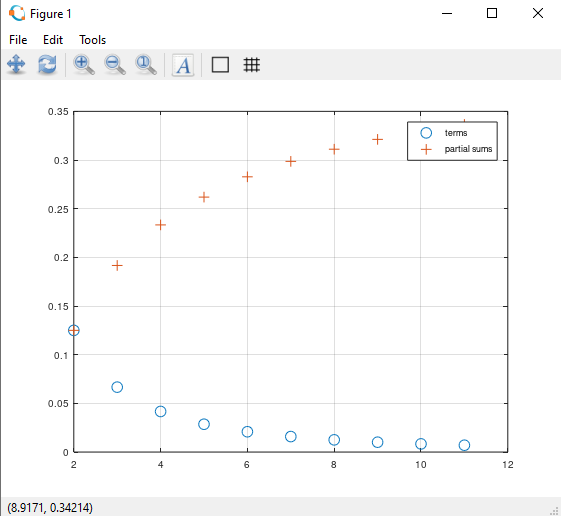
**2. Частичные суммы**

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



Частичные суммы

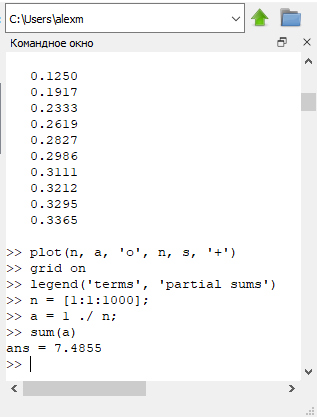
Построенные слагаемые и частичные суммы можно увидеть на Fig. 4.



Графическое представление результатов

**3. Сумма ряда**

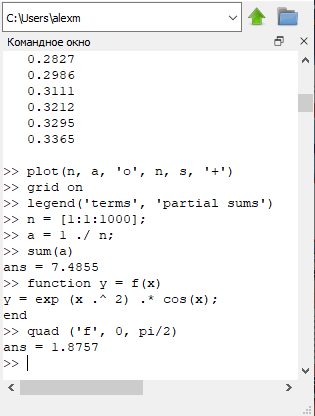
Найдём сумму первых 1000 членов гармонического ряда 1/n. Действия показаны на Fig. 5.



Сумма ряда

**4. Вычисление интегралов**

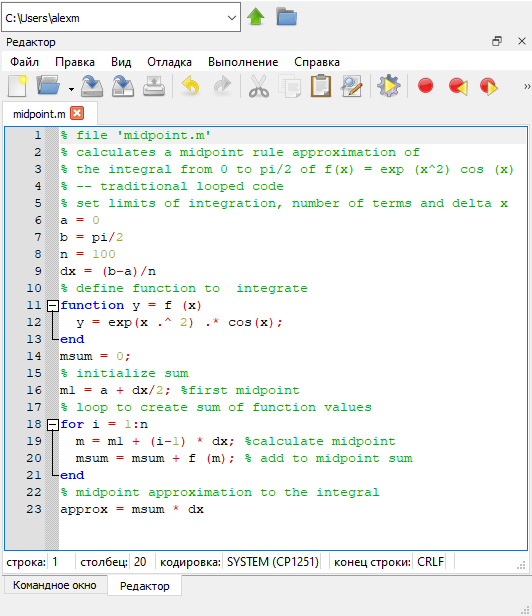
Численно посчитаем интеграл. См. Fig. 6.



Интегрирование функции

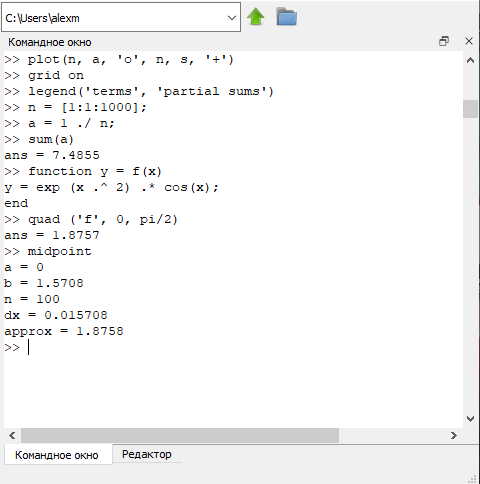
**5. Аппроксимирование суммами**

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.



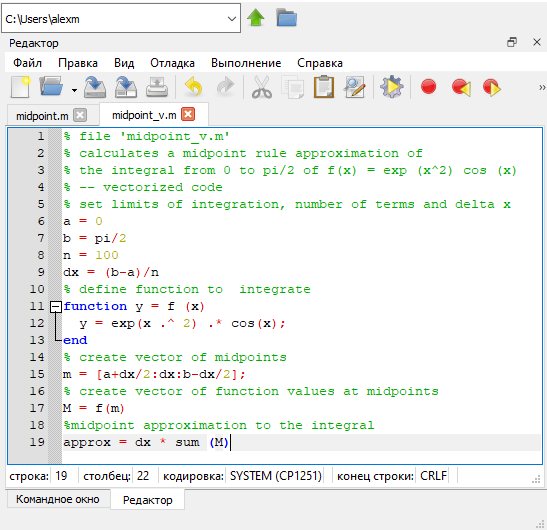
Содержание файла midpoint

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



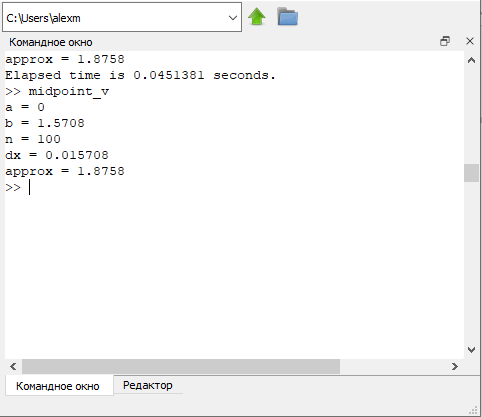
Результаты вывода

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



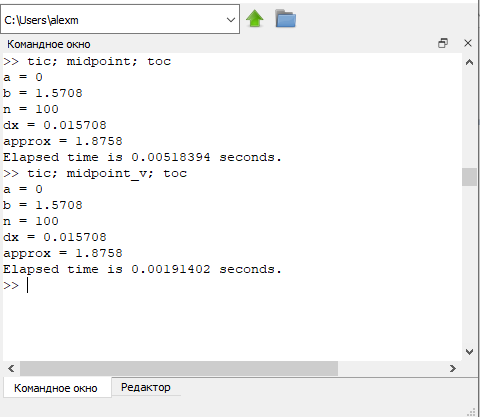
Содержание файла midpoint\_v

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



Сравнение полученных результатов

# Выводы

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