Научное програмирование

Отчет по лабораторной работе № 6

Коняева Марина Александровна НФИбд-01-21

Содержание

[Цель работы 1](#_Toc135308466)

[Выполнение лабораторной работы 1](#_Toc135308467)

[Пределы. Оценка 1](#_Toc135308468)

[Частичные суммы 2](#_Toc135308469)

[Сумма ряда 4](#_Toc135308470)

[Вычисление интегралов 4](#_Toc135308471)

[Аппроксимирование суммами 4](#_Toc135308472)

[Вывод 6](#_Toc135308473)

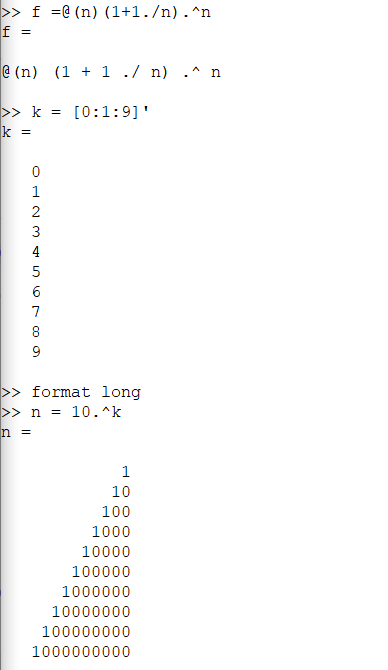
# Цель работы

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

# Выполнение лабораторной работы

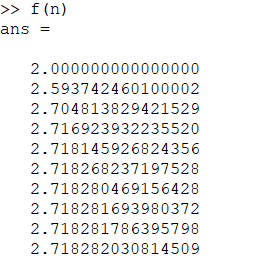
## Пределы. Оценка

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



Пределы код 01

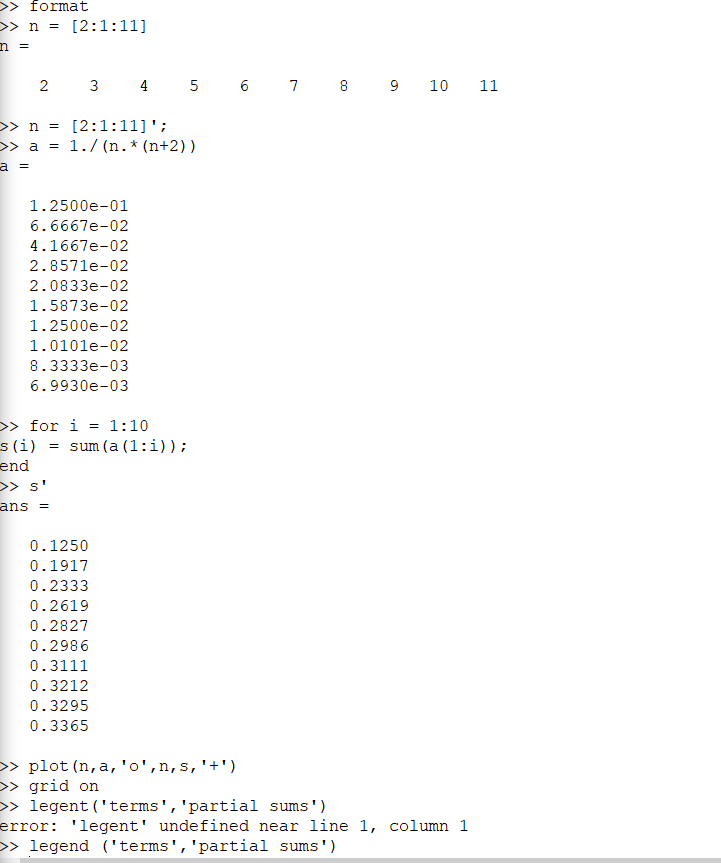
Получим ответ. На следующей фигуре видно, что предел сходится к значению 2.71828.



Пределы код 02

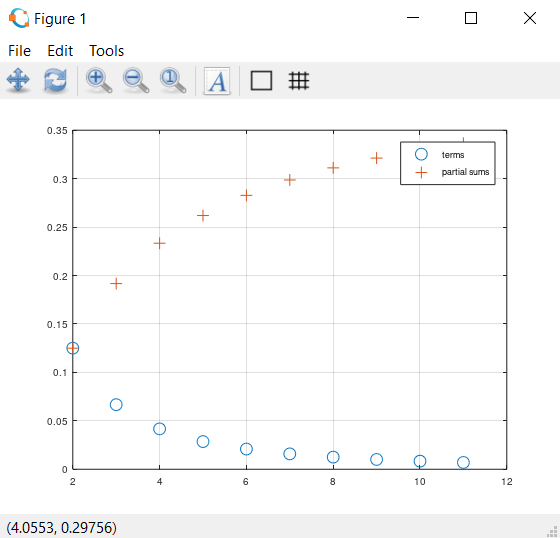
## Частичные суммы

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



Частичные суммы код 01

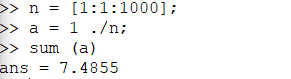
Построенные слагаемые и частичные суммы можно увидеть на следующем русинке:



Частичные суммы код 02

## Сумма ряда

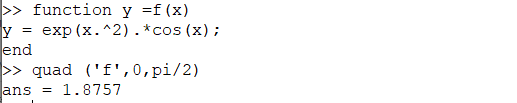
Найдём сумму первых 1000 членов гармонического ряда 1/n.



Сумма ряда код 01

## Вычисление интегралов

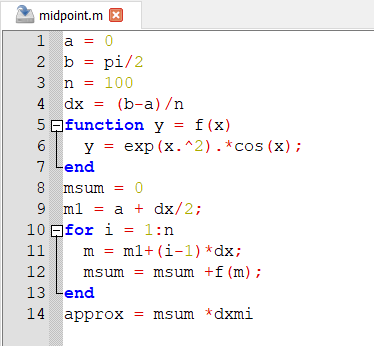
Численно посчитаем интеграл.



Вычисление интегралов код 01

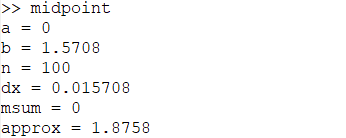
## Аппроксимирование суммами

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



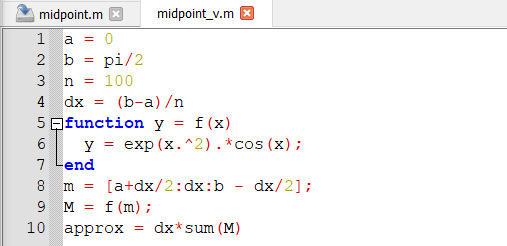
Аппроксимирование суммами код 01

Запустим этот файл в командной строке.



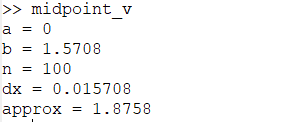
Аппроксимирование суммами код 02

Теперь напишем векторизованный код, не требующий циклов. Для этого создадим вектор х-координат средних точек.Введём код в текстовый файл и назовём его midpoint\_v.m.



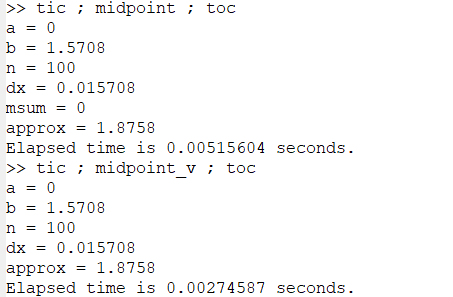
Аппроксимирование суммами код 03

Запустим этот файл в командной строке.



Аппроксимирование суммами код 04

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.

)

# Вывод

В ходе выполнения данной работы я научилась работать с пределами, последовательностями и рядами, а также научилась писать векторизованный программный код. Более того, удалось определить, что векторизованный код работает намного быстрее, чем код с циклами.