## РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

### Факультет физико-математических и естественных наук

### Кафедра прикладной информатики и теории вероятностей

#### ОТЧЕТ ПО

#### ЛАБОРАТОРНОЙ РАБОТЕ №6

*дисциплина: Научное программирование*

Студентка: Голос Елизавета Сергеевна  
Группа: НПМмд-02-20  
Ст. билет № 1032202186

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

**Ход работы**

**Пределы. Оценка**

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

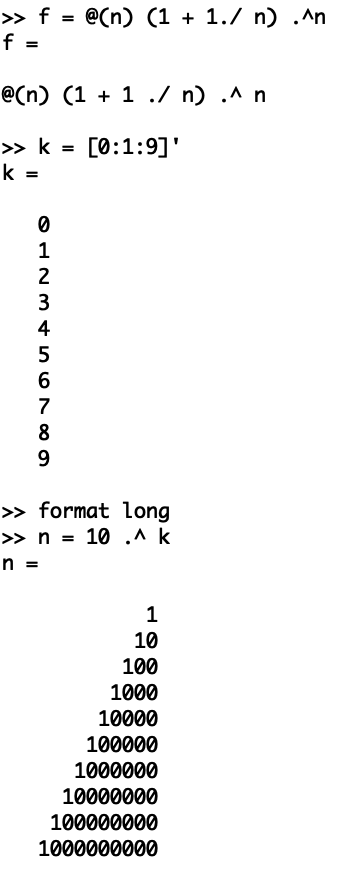


Рис.1 Промежуточные вычисления для рассчета предела

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

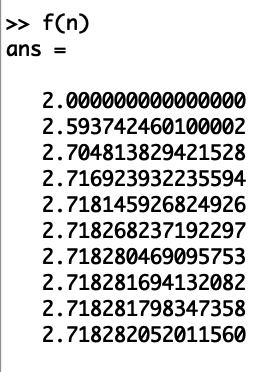


Рис.2 Искомый предел

**Частичные суммы**

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

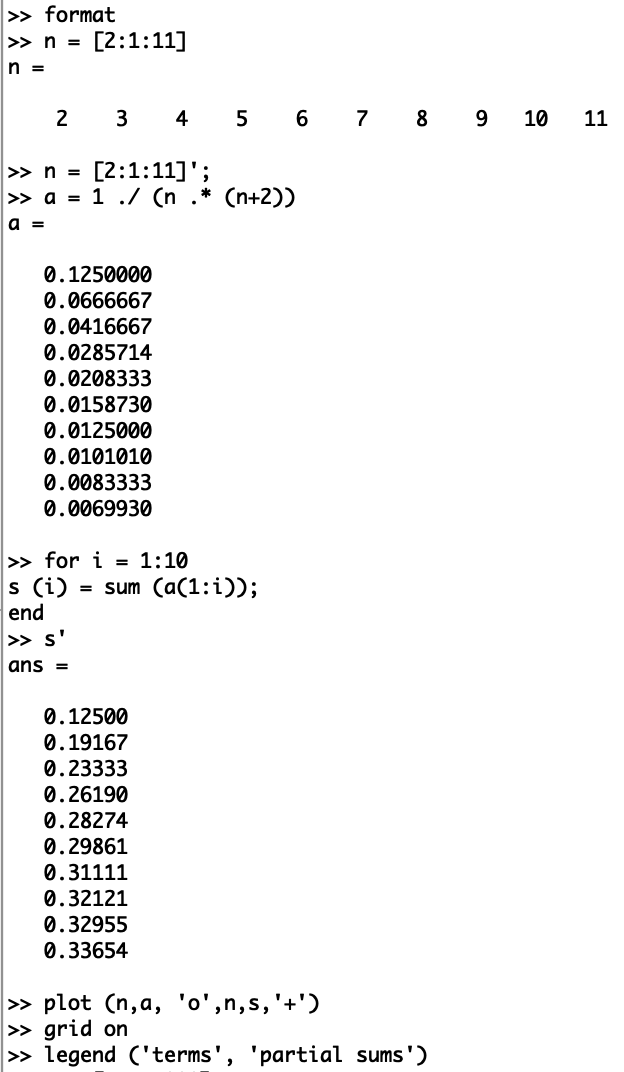


Рис.3 Частичные суммы

Построенные слагаемые и частичные суммы можно увидеть на рисунке 4.

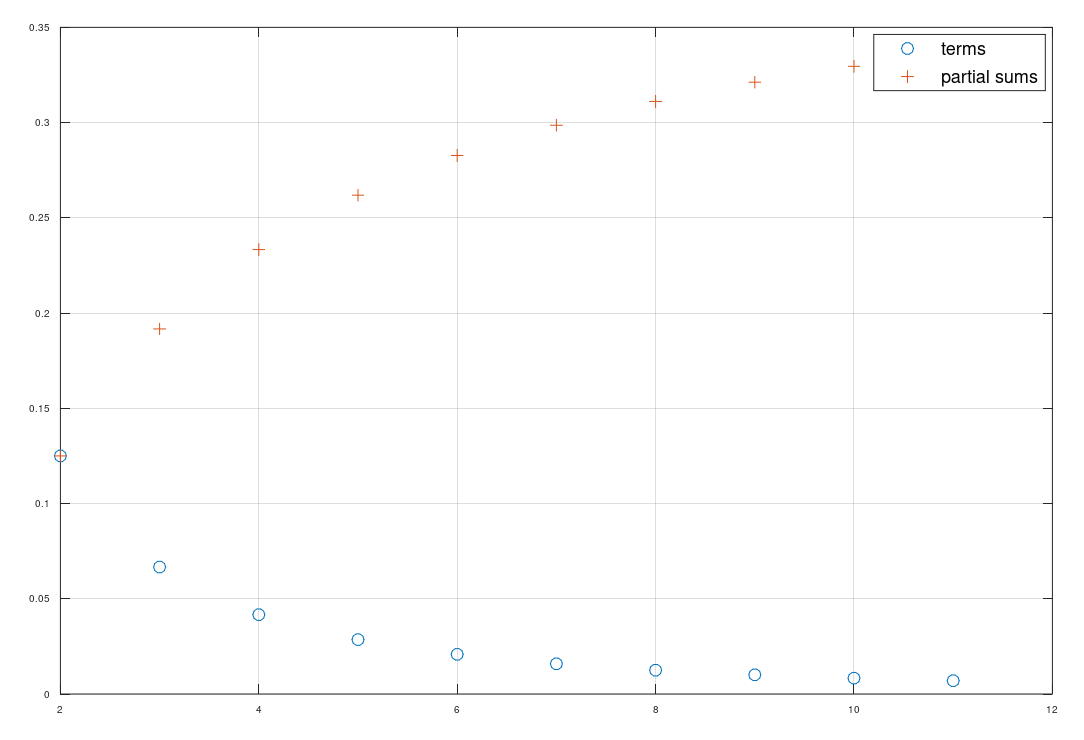
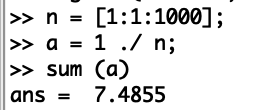


Рис.4 Графическое представление результатов.

**Сумма ряда**

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



**Вычисление интегралов**

Численно посчитаем интеграл. См. рисунок 6.

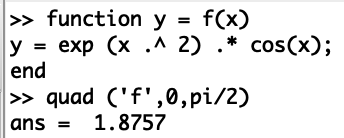


Рис.6 Интегрирование функции

**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 рисунке 7.

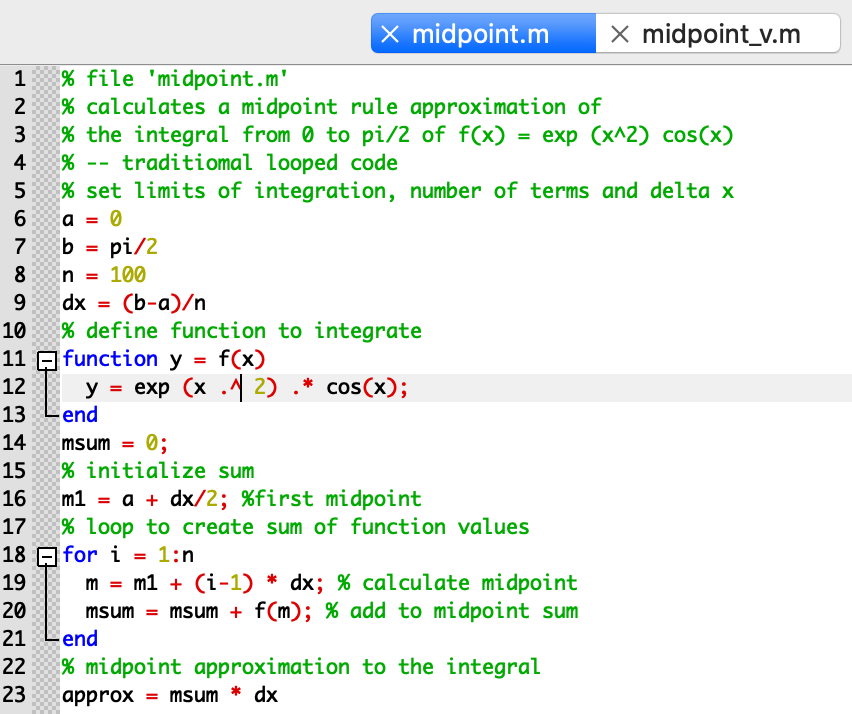


Рис.7 Содержание файла midpoint

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

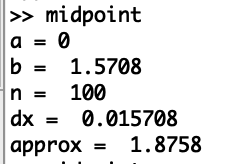


Рис.8 Результаты вывода

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

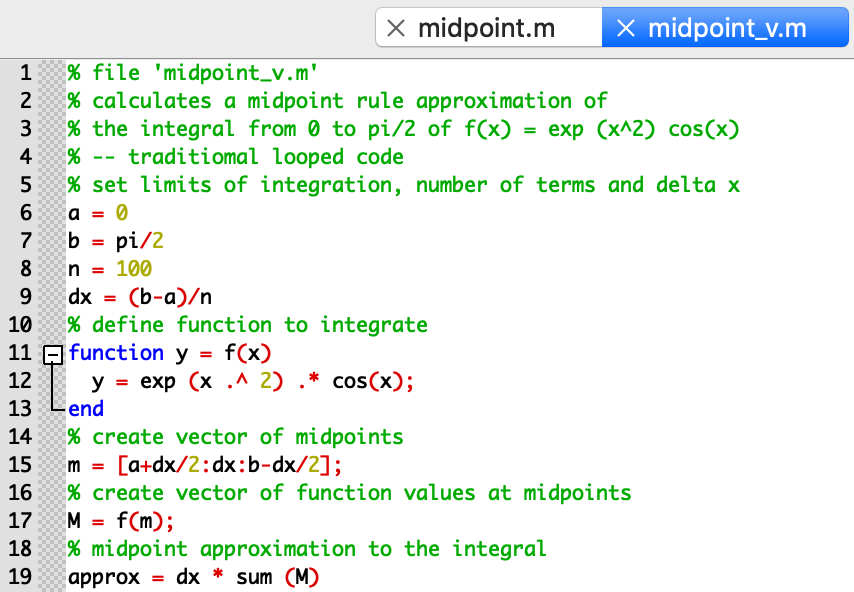


Рис.9 Содержание файла midpoint\_v

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

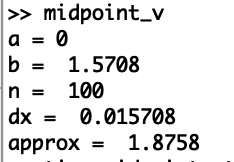


Рис.10 Вывод векторизованного кода программы

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 рисунке 11.

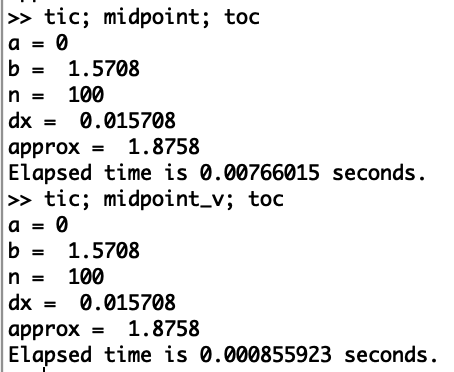


Рис.11 Сравнение полученных результатов

**Вывод**

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