Шаблон отчёта по лабораторной работе

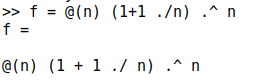
Турсунов Баходурхон Азимджонович

Содержание

# Выполнение 6 лабораторной работы

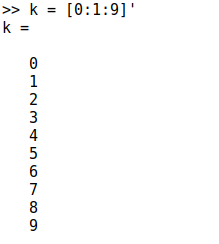
## Пределы, последовательности и ряды

* Octave - полноценный язык программирования, поддерживающий множество типов циклов и условных операторов.
* Расспишем простую функцию:

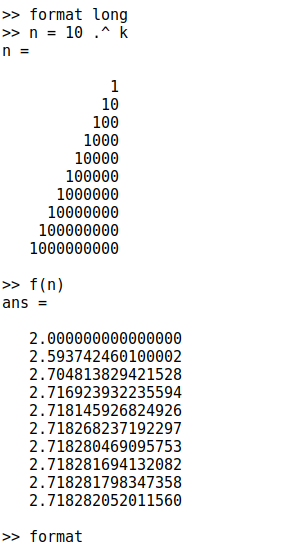
(Рис 1)

* метод которую я использовал здесь называется анонимной функцией. Это хороший способ быстро определить простую функцию

1. Далее создал индексную переменную, состоящую из целых чисел от 0 до 9

(Рис 2)

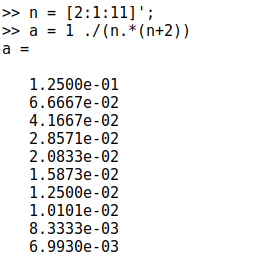
* синтаксис [0:1:9] создает вектор строки, который начинается с 0 и увеличивается с шагом от 1 до 9. Обратите внимание, что мы использовали операцию транспонирования просто потому что, наши результаты будут легче читать как векторы столбцы. Теперь мы возьмем степени 10, которые буду входными значениями, а затем оценим f(n).:

(Рис 3)

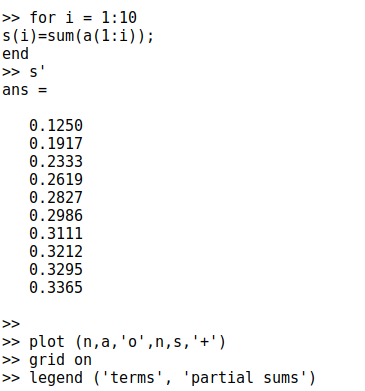
* Предел сходится к конечному значение, которое составляет приблизительно 2,71828… Подобные методы могут быть использованы для численного исследования последовательностей и рядов.

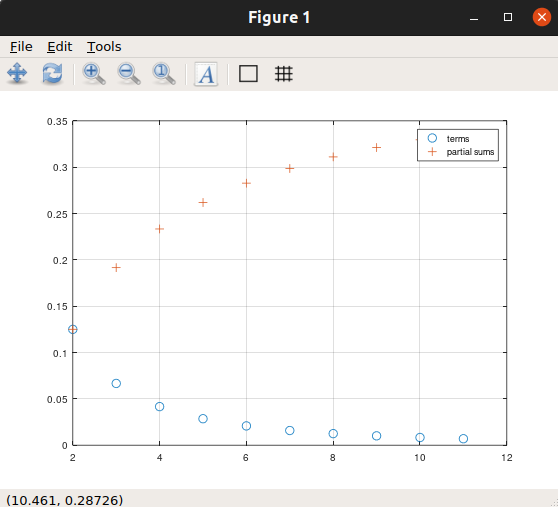
## Частичные суммы

1. Определим индексный вектор n от 2 до 11, а затем вычислим члены.

(Рис 4)

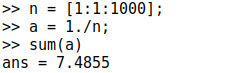
1. Если мы хотим знать частичную сумму, нам нужно написать *sum(a)*. Если мы хотим получить последовательность частичных сумм, нам нужно использовать цикл. Мы будем использовать цикл for с индексом *i* от 1 до 10. Для каждошо *i* мы получим частичную сумму последовательности *an* от первого слагаемого до *i*-го слагаемого. На выходе получается 10-элементный вектор этих частичных сумм.

(Рис 5)

(Рис 6)

## Сумма ряда

1. Найдем сумму первых 1000 членов гармоничного ряда. Нам нужно сгенерировать чекы как ряда вектор, а затем взять их сумму.

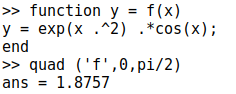
(Рис 6)

## Численное интегрирование

### Вычисление интегралов

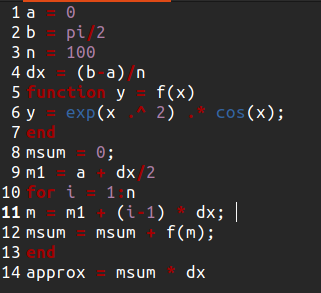
* Octave имеет несколько встроенных функций для вычисления определенных интегралов. Мы будем использовать команду quad(сокращение от слова квадратура).

1. Определяем функцию, чей интеграл мы будем считать, и считаем определенный интеграл командой quad.

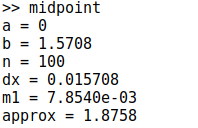
(Рис 7)

## Аппроксимирование суммами

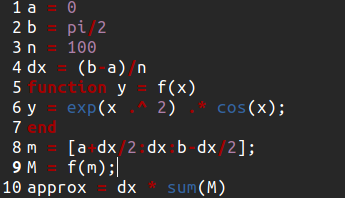
1. Создал файл midpoint.m и записал туда код

(Рис 8)

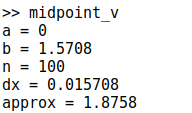
1. Запустил файл и вот результат:

(Рис 9)

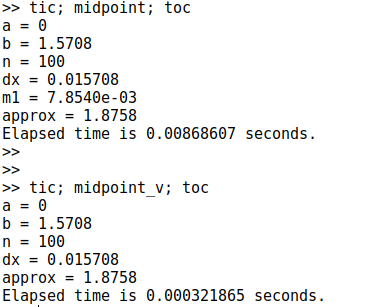
1. Далее создал еще один файл с название midpoint\_v.m и записал туда код:

(Рис 10)

1. Результат:

(Рис 11)

1. Сравнил результаты выполнения каждой реализации:

(Рис 12)

# Вывод

В ходе выполнения работы я научился считать пределы и частичные суммы рядов, считать определенный интеграл встроенной окмандой quad и методом средней точки, а так же увидел разницу в скорости работы трандиционного кода (с циклами) и векторизированного кода.