Лабораторная работа №6

Научное программирование

Алексей Бондарь

Содержание

# 1 Цель работы

Изучить в Octave методы расчета пределов, частичных сумм, суммы ряда, а также методы вычисления интегралов и аппроксимирования суммами.

# 2 Теоретическое введение

**Анонимная функция** - особый вид функций, которые объявляются в месте использования и не получают уникального идентификатора для доступа к ним. Обычно при создании анонимные функции либо вызываются напрямую, либо ссылка на функцию присваивается переменной, с помощью которой затем можно косвенно вызывать данную функцию.

В Octave aнонимные функции определяются с помощью синтаксиса @(argument-list) expression. Любые переменные, которые не найдены в списке аргументов, наследованы от объема включения. Анонимные функции полезны для создания простых функций без имени от выражений или для обертывания вызовов к другим функциям для адаптации их к использованию функциями как quad, которая применяется при вычислении интегралов.

Более подробно см. в [@Octave\_1:bash] и [@Octave\_2:bash].

# 3 Выполнение лабораторной работы

Рассмотрим предел:

Оценим данное выражение. Для этого определим функцию с помощью метода анонимной функции. Создадим также индексную переменную, состоящую из целых чисел от 0 до 9. Возьмем степени 10, которые будут входными значениями, а затем оценим (рис. fig. 1) и (рис. fig. 2).

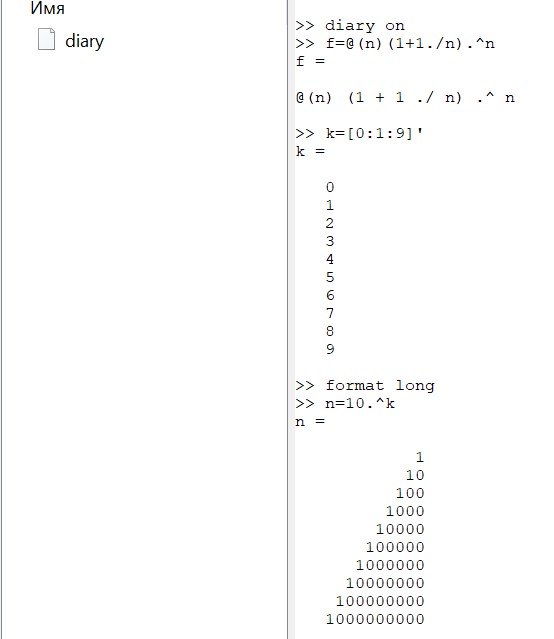


Рис. 1: Оценка выражения под знаком предела

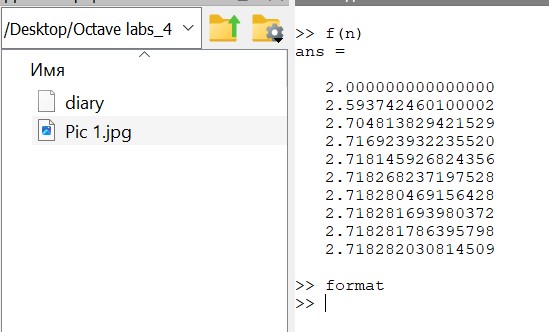


Рис. 2: Оценка выражения под знаком предела

Полученный результат близок к теоретическому значению предела - .

Пусть - ряд, n-й член равен . Определим индексный вектор n от 2 до 11 и вычислим члены. Для получения последовательности частичных сумм используем цикл и функцию sum(a). На выходе получаем 10-элементный вектор частичных сумм. Строим слагаемые и частичные суммы для на графике (рис. fig. 3).

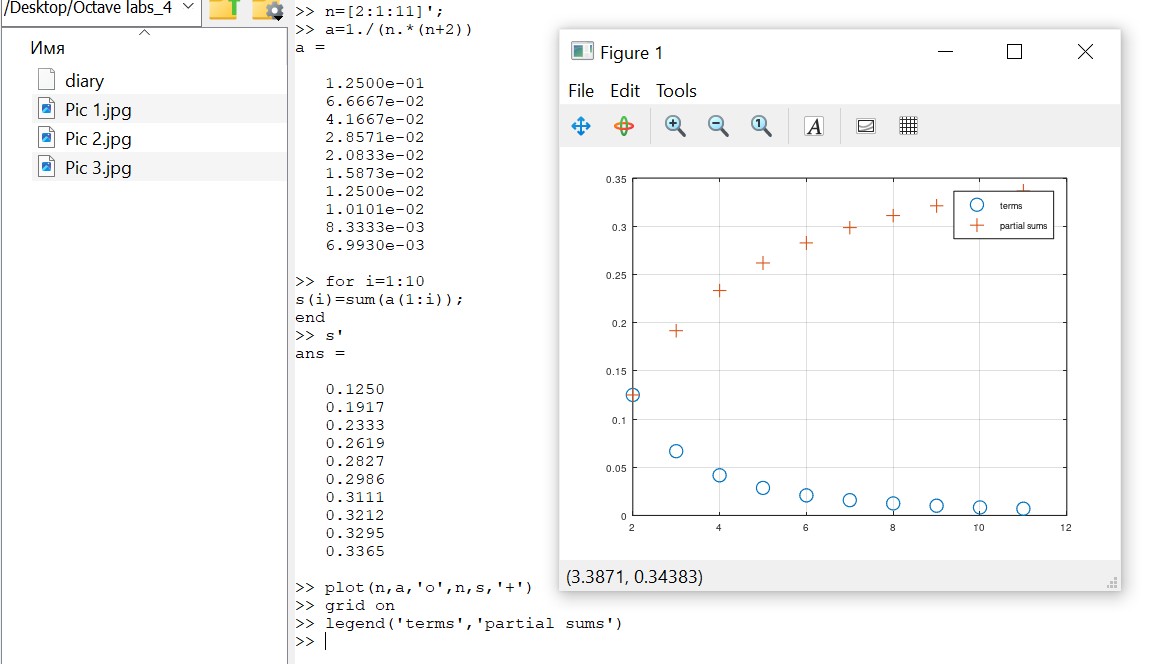


Рис. 3: Частичные суммы

Найдем сумму первых 1000 членов гармонического ряда (рис. fig. 4):

.

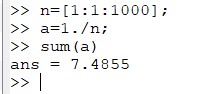


Рис. 4: Сумма ряда

Вычислим интеграл (рис. fig. 5):

.

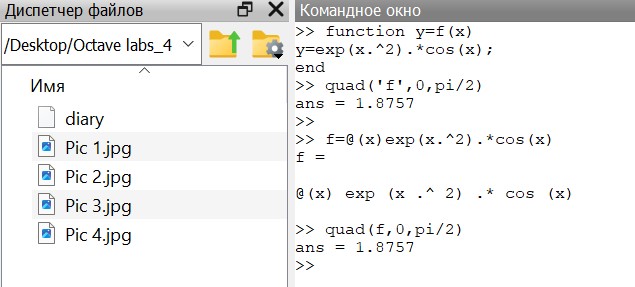


Рис. 5: Вычисление интеграла

Пишем код, чтобы вычислить указанный ранее интеграл по правилу средней точки для n=100: используем цикл, который добавляет значение функции к промежуточной сумме с каждой итерацией, а в конце сумму умножаем на (рис. fig. 6) и (рис. fig. 7).

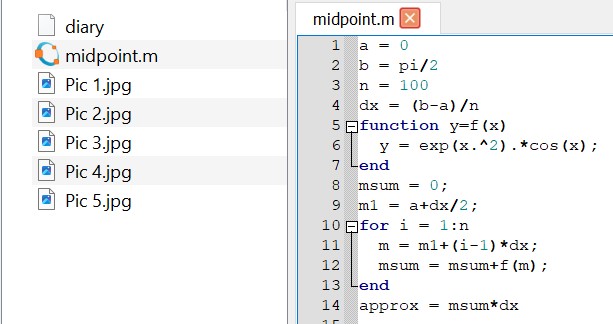


Рис. 6: Аппроксимирование суммами

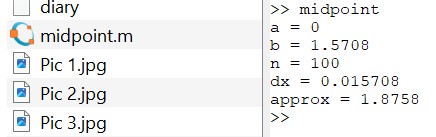


Рис. 7: Аппроксимирование суммами

Напишем векторизованный код. Создадим вектор x-координат средних точек, далее оцениваем f по этому вектору средней точки, чтобы получить вектор значений функции. Аппроксимация средней точки - это сумма компонент вектора, умноженная на (рис. fig. 8) и (рис. fig. 9).

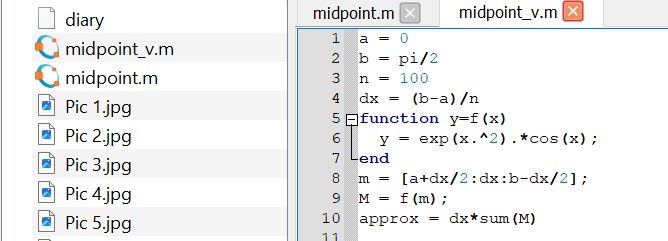


Рис. 8: Аппроксимирование суммами - векторизованный код

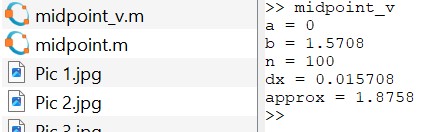


Рис. 9: Аппроксимирование суммами - векторизованный код

Полученные результаты совпадают с предыдущими. Сравним время выполнения для каждой реализации. Получили, что векторизованный код более эффективен по времени, чем традиционный (рис. fig. 10).

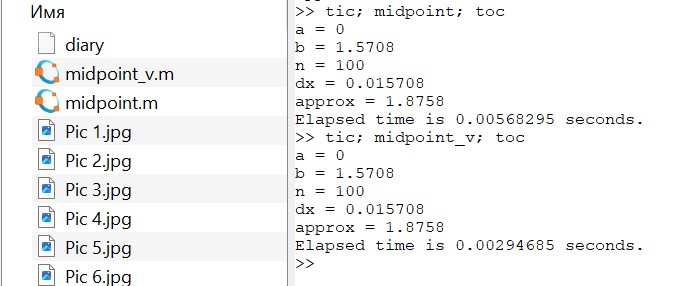


Рис. 10: Сравнение кодов

# 4 Вывод

В ходе выполнения данной лабораторной работы я изучил в Octave методы расчета пределов, частичных сумм, суммы ряда, а также методы вычисления интегралов и аппроксимирования суммами.