Отчёт по лабораторной работе №6

Сырцов Александр Юрьевич

Содержание

# Цель работы

Научится находить частичные и полные суммы рядов, значения пределов и интегралов в Octave.

# Задание

• Сделать отчёт по лабораторной работе в формате Markdown.

• В качестве ответа предоставить отчёты в 3 форматах: pdf, docx и md (в архиве, поскольку он должен содержать скриншоты, Makefile и т.д.)

# Выполнение лабораторной работы

## Пределы, последовательности и ряды

1. Реализую функцию для оценки некоторого предела:

Функция реализована в функциональном стиле и представляет собой анонимное лямбда-выражение, присвоенное объекту f, которое в качестве своего терма принимает вектор значений n. Вместе с тем сразу задаём вектор k и меняем формат отображения чисел для следующего шага (рис. -fig. 1).

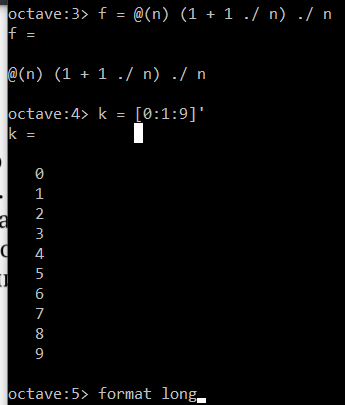


Figure 1: Анонимная функция и вектор *k*

Замечание 1: вектор k задан как транспонированный конвектор, что крайне удобно и быстро в сочетании с range-оператором.

Замечание 2: Реализованная функция чистая и является функцией первого класса, однако перестаёт быть анонимной после присвоения объекту f, так как он становится “allias” для изначального выражения.

1. Вектор индексов k определил собой количество итераций для вектора n (рис. -fig. 2), представляя собой степени от 0 до 9. Итоговый вектор значений – это вектор чисел от 1 до 1000000000. При его подстановке в функцию f мы постепенно приближаемся к окрестности нужного значения и по закономерности чисел можно сказать, к какому значению стремиться предел, а именно к значению числа *e*, так как это второй замечательный предел (рис. -fig. 3).

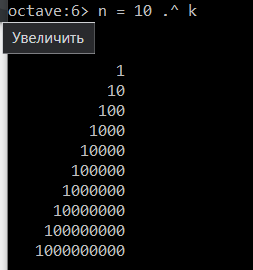


Figure 2: Инициалзация вектора значений

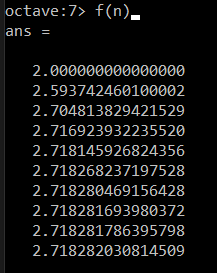


Figure 3: Подстановка значений в функцию

замечание: можно было не инициализировать отдельно k, а сразу подставить вектор для экономии памяти.

## Частичные суммы

1. Необходимо рассчитать частичные суммы ряда от второго до одинадцатого элемента включительно:

где

Для этого я инициализирую вектор индексов и подставляю в n-й член ряда, получая десять элементов ряда (рис. -fig. 4).

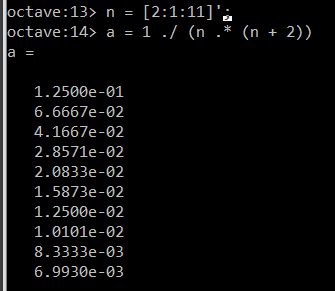


Figure 4: Значения для нахождения частичных сумм

1. Частичные суммы расчитываются, как суммы от одного элемента до другого, не покрывая исходный ряд. Следуя такой логике, реализую цикл от 1 до 10, где в конвектор частичных сумм записывается сумма от первого элемента вектора a до i-го с помощью функции sum(). В конце выодим все значения через транспонированный конвектор (рис. -fig. 5).

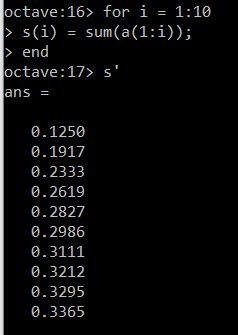


Figure 5: Частичные суммы

1. Визуализирую резултаты и очевидно получаю обратную корелляцию между значениями ряда и значением частичных сумм (рис. -fig. 6).

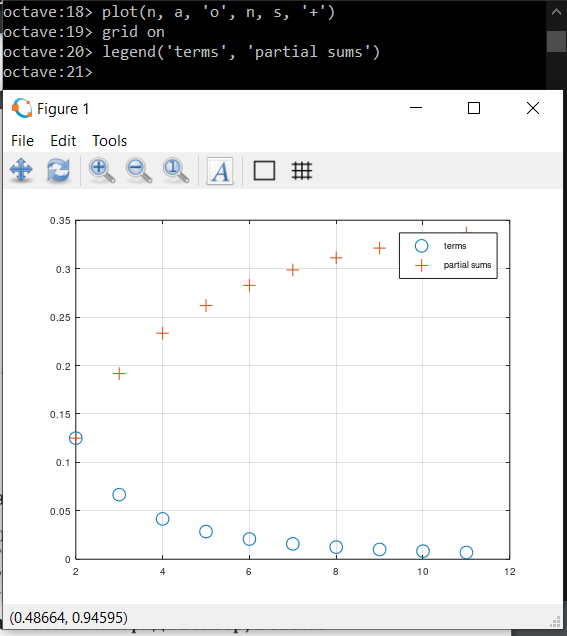


Figure 6: Граафик значений элементов ряда и частичных сумм ряда

## Сумма ряда

1. Найду сумму элеметов горманического ряда:

Аналогично задав вектор индексов, находим элементы ряда и используем стандартную функцию sum() (рис. -fig. 7).

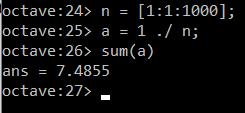


Figure 7: Сумма ряда

## Численное интегрирование

1. Нахожу значение определённого интеграла:

Для этого явно задаю функцию f и использую стандартную функцию quad() (рис. -fig. 8).

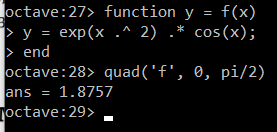


Figure 8: определённый интеграл функцией *quad()*

## Аппроксимированние суммами

1. Рассчитаем тот же интеграл, но по правилу средней точки:

Из комментариев к коду понятно, как работает правило и как работает код. Сначала реализация программы с использованием циклов midpoint (рис. -fig. 9).

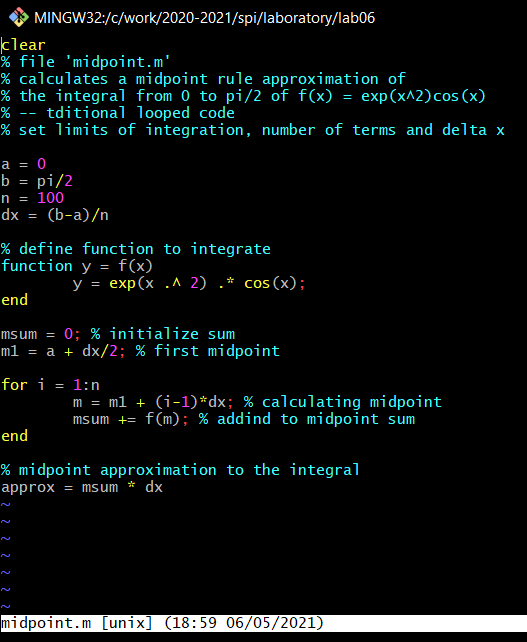


Figure 9: Реализация *midpoint*

1. Запускаю код (рис. -fig. 10).

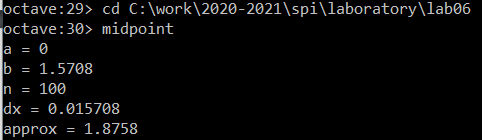


Figure 10: Запуск *midpoint*

1. Теперь аналогично реализую векторизованную программу midpoint\_v (рис. -fig. 11) и запускаю (рис. -fig. 12).

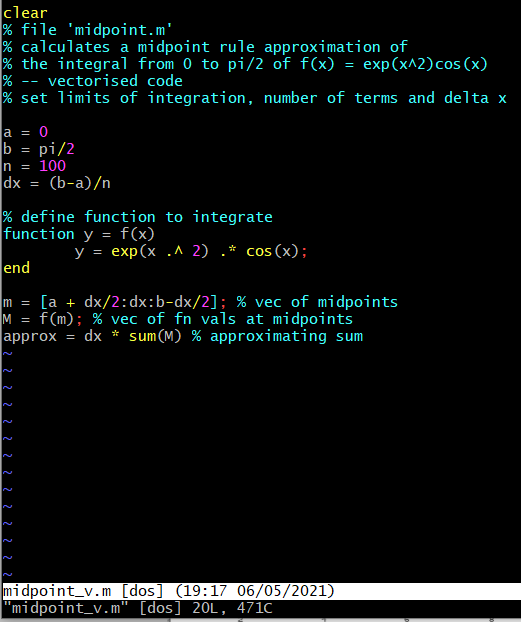


Figure 11: Реализация *midpoint\_v*

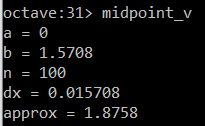


Figure 12: Запуск *midpoint\_v*

1. Сравниваю результаты выполнения двух программ с помощью макросов (команд) tic и toc. Из результатов видно, что векторизванный код работает быстрее подобно тому, что мы наблюдали в одной из прошлых лабораторных (рис. -fig. 13).

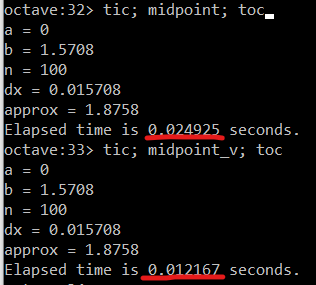


Figure 13: Сравнение скорости исполнения кода

# Выводы

Я успешно пополнил опыт вычисления приближённых и точных значений рядов, пределов, сумм и интегралов, научившись этому в языке программирования Octave.