EcoLab1

Создание компонента, реализующего алгоритм вычисления коэффициентов полинома по его корням (функция ***poly*** из MATLAB)

Выполнил

Волков Владислав

21ПИ-3

1. Алгоритм

Функция poly в MATLAB используется для создания вектора коэффициентов полинома на основе его корней. Этот функция очень полезна, когда перед нами стоит задача – найти коэффициенты полинома, если известны его корни – x_1, $x_2, \dots x_n$. Полиномов с одинаковыми корнями существует бесчисленное множество. Однако среди них существует единственный полином с коэффициентом a_n, равным единице. Этот полином называется приведенным, как раз его вычисляет данная функция. Все остальные полиномы получаются из приведенного полинома умножением всех коэффициентов на произвольное число a_n, от которого требуется лишь, чтобы оно не было равно нулю. Поэтому для однозначного решения задачи требуется задать n корней и коэффициент при старшем члене полинома.

$$P_n(x) = a_n(x - x_n)(x - x_{n-1}) \dots (x - x_1)$$

В случае приведенного полинома

$$P_n(x) = 1 \cdot (x - x_n)(x - x_{n-1}) \dots (x - x_1)$$

Реализация в MATLAB выглядит следующим образом

```
c = zeros(n+1,1);
c(1) = 1;
for j = 1:n
        c(2:j+1) = c(2:j+1)-z(j)*c(1:j);
end
```

c – массив коэффициентов
z – массив корней

2. Оценка сложности

Если посмотреть на алгоритм, то скорость работы алгоритмы можно оценить как

$$(n+1) \cdot 1 + (n+1) \cdot 2 + (n+1) \cdot 3 + \dots + (n+1) \cdot n$$
$$= (n+1) \cdot \left(\frac{n+2}{2}\right) = O(n^2)$$

По памяти данный алгоритм занимает дополнительно $O(n)$ памяти для промежуточного хранения коэффициентов

3. Особенности реализации

Для работы с различными типа данных я выбрал следующую сигнатуру функции

```
/*Функция poly, принимающая r - вектор корней полинома, возвращает коэффициенты полинома*/
/*
    Parametrs:
        roots_data: void*
            Pointer to array of roots
        roots_count: size_t
            Number of roots
        type_size: size_t
            Size of one element
        mult: int (const void*, const void*, void*)
            Fuction of multiply
        sub: int (const void*, const void*, void*)
            Function of subtraction
        initOnes: int (void*)
            Initialization by ones
        coefs_data: void*
            Pointer to array of coefficients of the polynomial
    Returns:
        code: int16_t
            Return code
*/
int16_t (ECOCALLMETHOD *poly)(/* in */ struct IEcoLab1* me, /* in */ void* roots_data, size_t roots_count,
                                       size_t type_size,
                                       int (*mult)(const void* first, const void* second, void* res),
                                       int (*sub)(const void* first, const void* second, void* res),
                                       int (*initOnes)(void* res),
                                       /* out */ void* coefs_data);
```

Для того чтобы использовать свой тип, пользователь должен определить для него операцию умножения, операцию вычитания и инициализацию единицей в следующей сигнатуре: ***int(*<name>)(/*in*/, /*out*/)***, где функция возвращает 0 в случае корректного исполнения.

Также пользователь передает в функцию массив корней, его размер и размер одного элемента.

Для удобства функция возвращает код, в случае если возвращется 0, функция отработала корректно, если -1 – переданы не все данные (передан нулевой указатель), если -2 – переданные функции, при исполнении вернули ненулевой код

4. Результаты тестирования

Результаты также есть в отдельном файле – *ComparisonPOLY.png*

В качестве тестирования я использовал следующие типы данных:

- Int
- Float
- Double
- Complex

Тестирующий интерфейс вынесен в отдельные заголовки (EcoLab1TestInt.h, EcoLab1TestDouble.h, EcoLab1TestCmplx.h).

! Стоит отметить, что коэффициенты **очень быстро** растут.

Это приводит к переполнениям, чего не избежать. Поэтому первостепенно было проведено сравнение результатов на корректность и точность получаемых значений:

Целые числа (Int)

Roots	8	-2	3	8	-10	
Coefficients	1	-7	-96	772	-64	-3840

```
>> poly([8 -2 3 8 -10])

ans =

          1          -7         -96         772         -64       -3840
```

Вещественные числа(Double)

Roots	7.858211	-7.285073	9.588610	4.232612	-4.286935	
Coefficients	1.0	-10.107425	-70.449	730.498274	968.856734	-9960.213853

```
>> poly([7.858211       -7.285073        9.588610        4.232612       -4.286935       ])

ans =

   1.0e+03 *

    0.0010   -0.0101   -0.0704    0.7305    0.9689   -9.9602
```

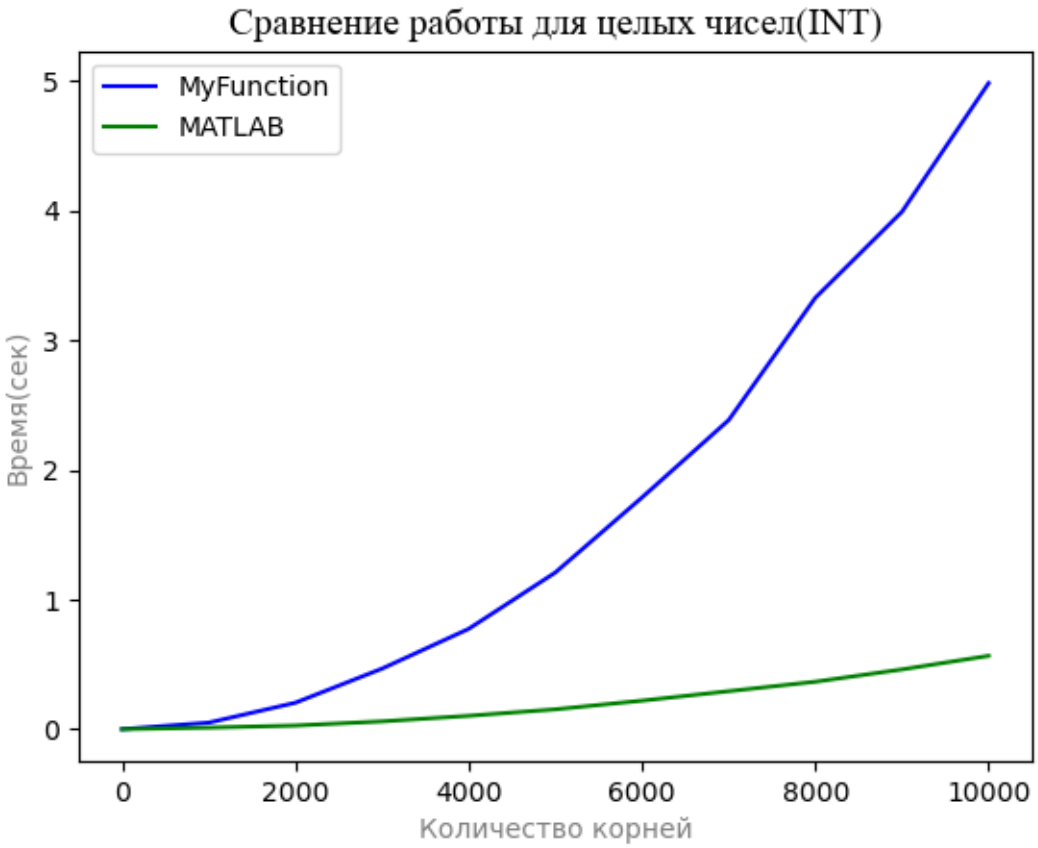
Комплексные числа(Complex)

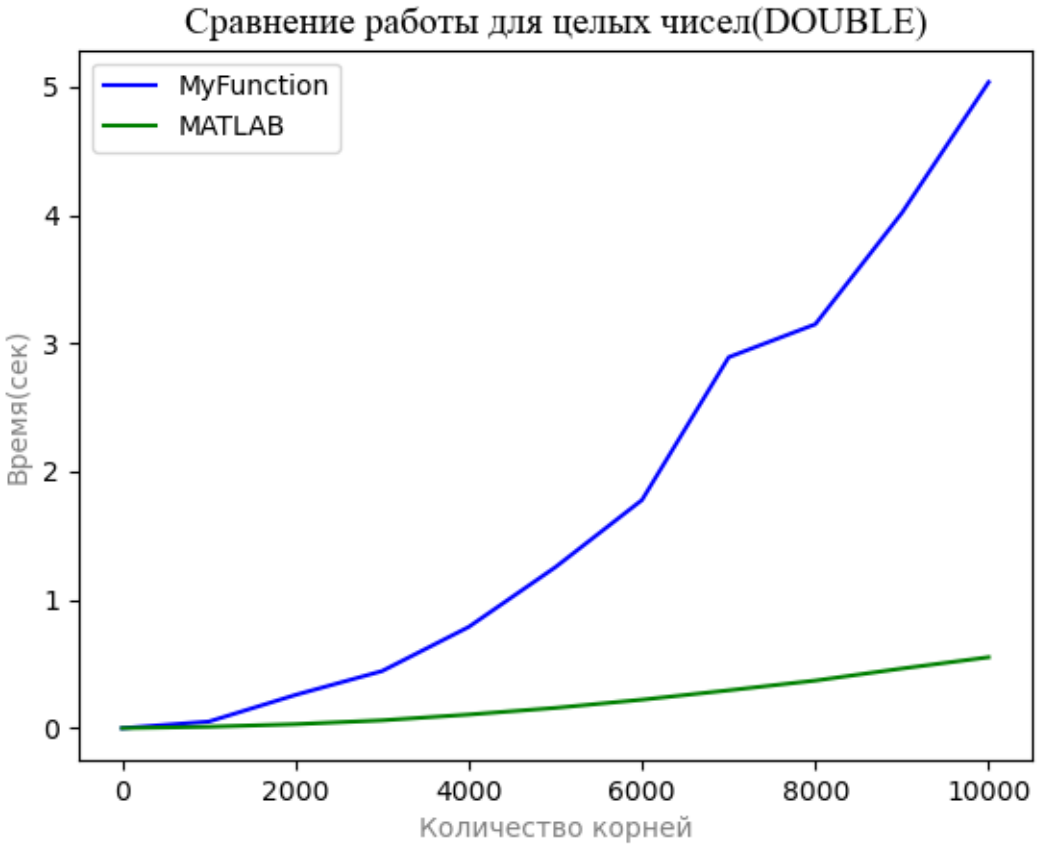
Roots	7.858211-3.642537i	9.588610+2.116306i	-4.286935-4.286782i	-5.370952+0.781121i	-1.562853+1.453444i	
Coefficients	1.000000	-6.226081+3.578448i	-69.277804-25.858592i	210.000165-165.322614i	2864.322301+310.277759i	5659.966176-1907.148248i

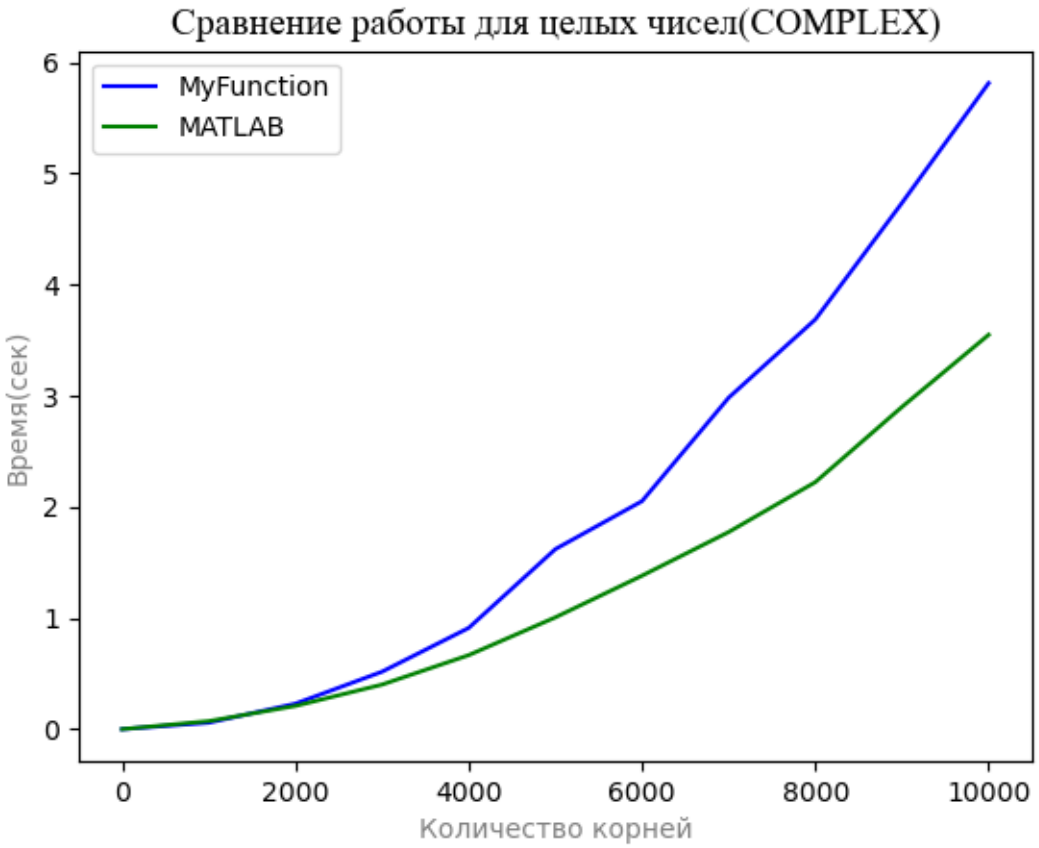
```
>> poly([7.858211-3.642537i 9.588610+2.116306i -4.286935-4.286782i -5.370952+0.781121i -1.562853+1.453444i])

ans =

   1.0e+03 *

   0.0010 + 0.0000i  -0.0062 + 0.0036i  -0.0693 - 0.0259i   0.2100 - 0.1653i   2.8643 + 0.3103i   5.6600 - 1.9071i
```

- Точность данных совпадает, что говорит о корректности реализованной функции

Затем было проведено тестирование времени работы относительно увеличения входных данных в диапазоне от 0 до 10000 корней


Сравнение работы для целых чисел(INT)
MyFunction
MATLAB
Время(сек)
0
1
2
3
4
5
0
2000
4000
6000
8000
10000
Количество корней


Сравнение работы для целых чисел(DOUBLE)
MyFunction
MATLAB
Время(сек)
0
1
2
3
4
5
0
2000
4000
6000
8000
10000
Количество корней



- Можно заметить, что моя реализация заметно уступает на Int и Double начиная с 1000, однако на комплексных числах до 2000 идентичные результаты и далее не сильная разница. Могу предположить, что виновата разница реализаций, так как скорее всего в MATLAB присутствуют некоторые оптимизации. Также попробую предположить, что MATLAB Online который я использовал, исполняет функции на сервере с более мощной ЭВМ из-за чего мои результаты так «просели».