

## EcoLab1

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

1. [Алгоритм](#)
2. [Оценка сложности](#)
3. [Особенности реализации](#)
4. [Результаты тестирования](#)
5. [Поддержка других компонентов](#)
6. [Визуализация](#)

Выполнил

Волков Владислав

21ПИ-3

## 1. Алгоритм

Функция `poly` в MATLAB используется для создания вектора коэффициентов полинома на основе его корней. Эта 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 алгоритма можно оценить как

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

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

### 3. Особенности реализации

Для работы с различными типами данных я выбрал следующую сигнатуру функции

```
/*Функция poly, принимающая r - вектор корней полинома, возвращает коэффициенты полинома*/
/*
  Params:
    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 единиц 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 возвращается 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

*Примечание. Для сравнения использовался MATLAB Online в связи с открытым доступом к ресурсу (20 часов в месяц)*

Ссылка на сервис:

<https://uk.mathworks.com/products/matlab-online.html>

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 первоначально 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
```

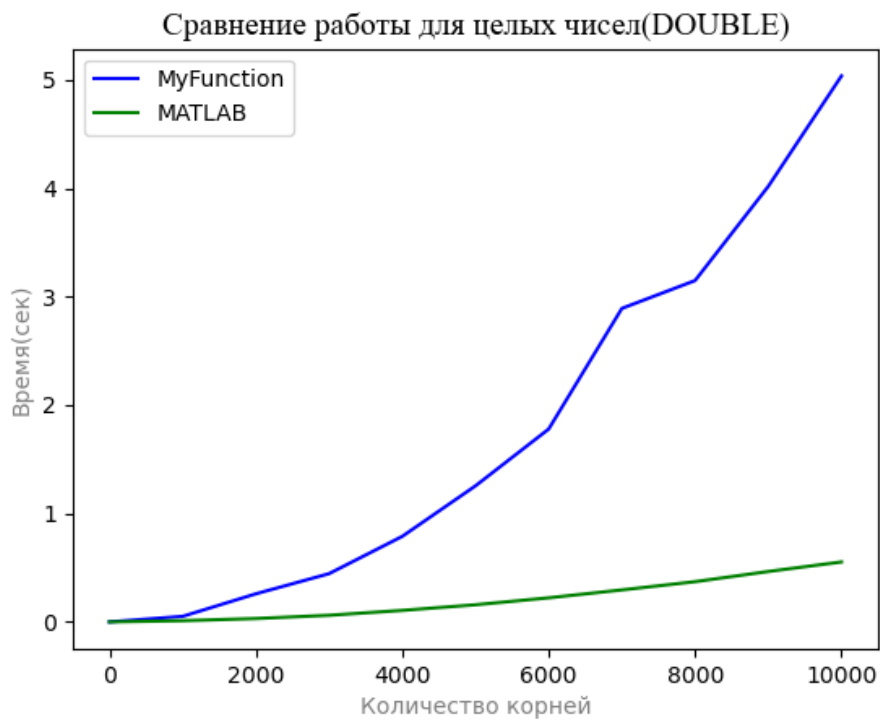
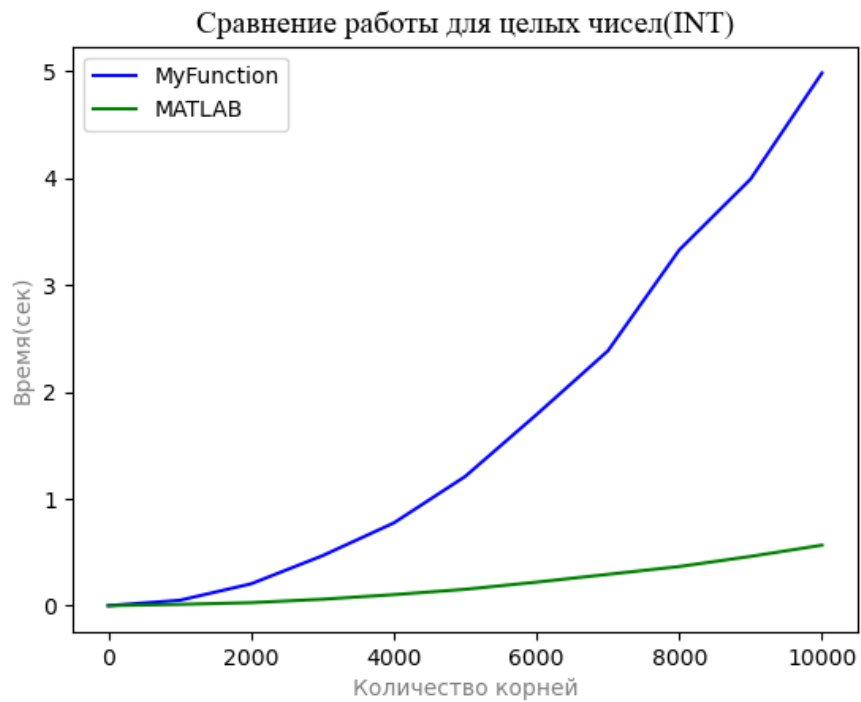
### Комплексные числа(Complex)

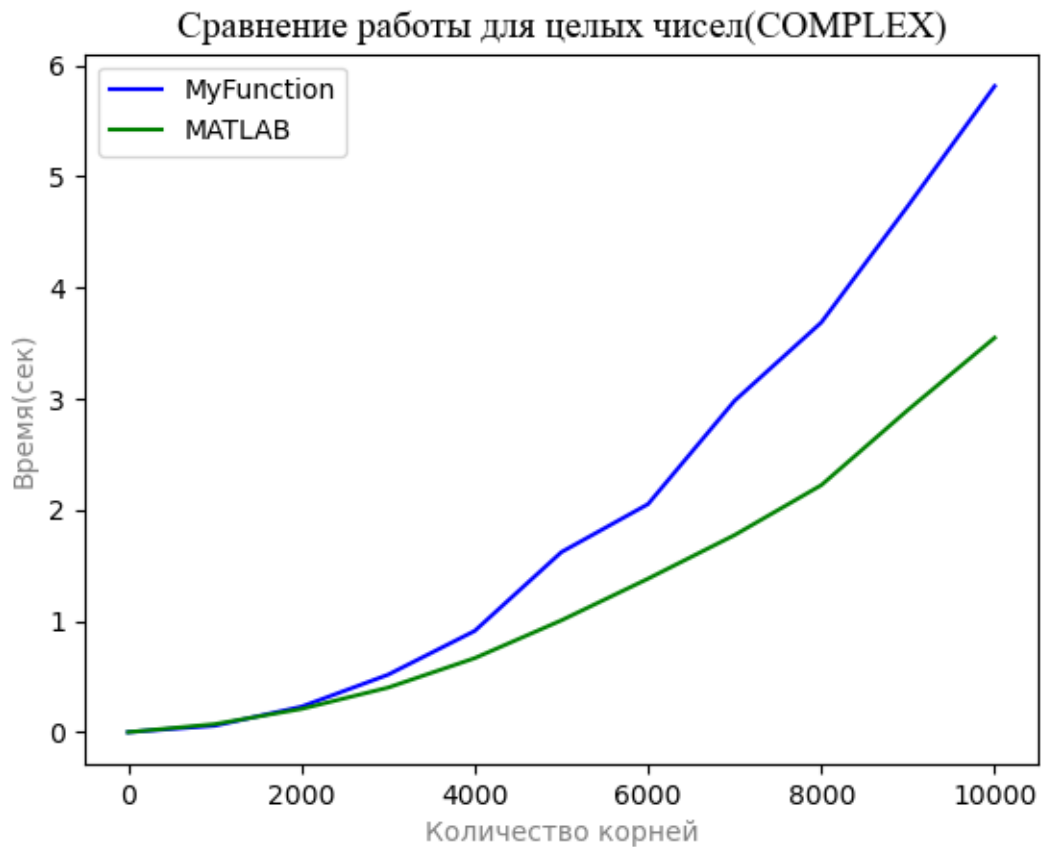
Roots	7.858211-3.642537i	9.588610+2.116306i	-4.286935-4.286782i	-5.370952+0.781121i	-1.562853+1.453444i	
Coefficients	1.000000	-6.226081+3.578448i	-69.277804-25.858592i	210.000165-165.322614i	2864.322301+310.277759i	5659.966176-1907.148248i

```
>> 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





- 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».

## 5. Поддержка других компонентов

Задачей второй лабораторной работы является «используя включение/агрегирование добавить компонент(-ы) калькулятора, выполняющий(-ие) операции сложения, вычитания, деления и умножения». А также продемонстрировать работу запросов соответствующих интерфейсов используя метод QueryInterface.

Для поддержки сторонних компонентов в структуру CEcoLab1 были добавлены указатели на новые интерфейсы и их виртуальные таблицы:

```
typedef struct CEcoLab1 {  
  
    /* Таблица функций интерфейса IEcoLab1 */  
    IEcoLab1Vtbl* m_pVTblIEcoLab1;  
  
    /* Таблица функций интерфейса IEcoCalculatorX */  
    → IEcoCalculatorXVTbl* m_pVTblIEcoCalculatorX;  
  
    /* Таблица функций интерфейса IEcoCalculatorY */  
    → IEcoCalculatorYVTbl* m_pVTblIEcoCalculatorY;  
  
    /* Счетчик ссылок */  
    uint32_t m_cRef;  
  
    /* Интерфейс для работы с памятью */  
    IEcoMemoryAllocator1* m_pIMem;  
  
    /* Системный интерфейс */  
    IEcoSystem1* m_pISys;  
  
    /* Данные экземпляра */  
    char_t* m_Name;  
  
    /* Указатель на интерфейс IEcoCalculatorX включаемого компонента */  
    → IEcoCalculatorX* m_pIEcoCalculatorX;  
  
    /* Указатель на интерфейс IEcoCalculatorY включаемого компонента */  
    → IEcoCalculatorY* m_pIEcoCalculatorY;  
  
    /* Указатель на интерфейс внутреннего агрегируемого компонента */  
    → IEcoUnknown* m_pInnerUnknown;  
  
} CEcoLab1, *CEcoLab1Ptr;
```

В результате выполнения были доработаны следующие функции:

- CEcoLab1\_QueryInterface
- initCEcoLab1
- deleteCEcoLab1

А также добавлены функции, использующие имплементацию функций внешних компонентов в виртуальную таблицу CEcoLab1:

- CEcoLab1\_Addition,
- CEcoLab1\_Subtraction,

- CEcoLab1\_Multiplication,
- CEcoLab1\_Division

Включение и взаимозаменяемость компонентов:

```
/* Включение IEcoCalculatorX из CEcoCalculatorA */
result = pIBus->pVTbl->QueryComponent(pIBus, &CID_EcoCalculatorA, 0, &IID_IEcoCalculatorX, (void**)&pCMe->m_pIEcoCalculatorX);
if (result != 0 || pCMe->m_pIEcoCalculatorX == 0) {
    /* В случае ошибки включение IEcoCalculatorX из CEcoCalculatorB */
    result = pIBus->pVTbl->QueryComponent(pIBus, &CID_EcoCalculatorB, 0, &IID_IEcoCalculatorX, (void**)&pCMe->m_pIEcoCalculatorX);
}
```

Далее, если компонент А или компонент В доступен с интерфейсов IEcoCalculatorX, то мы определяем методы интерфейса в виртуальную таблицу в методе CEcoLab1\_QueryInterface

```
else if (IsEqualUGUID(riid, &IID_IEcoCalculatorX)) {
    *ppv = &pCMe->m_pVTblIEcoCalculatorX;
    pCMe->m_pVTblIEcoLab1->AddRef((IEcoLab1*)pCMe);
}
```

Агрегирование:

```
/* Агрегирование CEcoCalculatorE для доступа к IEcoCalculatorY */
result = pIBus->pVTbl->QueryComponent(pIBus, &CID_EcoCalculatorE, pOuterUnknown, &IID_IEcoUnknown, (void**)&pCMe->m_pInnerUnknown);
if (result != 0 || pCMe->m_pInnerUnknown == 0) {
    /* При ошибке включаем IEcoCalculatorY из CEcoCalculatorD */
    result = pIBus->pVTbl->QueryComponent(pIBus, &CID_EcoCalculatorD, 0, &IID_IEcoCalculatorY, (void**)&pCMe->m_pIEcoCalculatorY);
    if (result == 0) {
        pCMe->m_pVTblIEcoCalculatorY = pCMe->m_pIEcoCalculatorY->pVTbl;
    }
}
```

В методе CEcoLab1\_QueryInterface также добавляем новое условие, что пользователь запрашивает поддерживаемый компонент и указатель на него не равен 0. В этом случае вызов метода просходит по указателю m\_pInnerUnknown.

```
else if (IsEqualUGUID(riid, &IID_IEcoCalculatorX) && pCMe->m_pInnerUnknown != 0) {
    return pCMe->m_pInnerUnknown->pVTbl->QueryInterface(pCMe->m_pInnerUnknown, riid, ppv);
}
```

## Демонстрация работы

Демонстрация работы содержится в файле EcoLab1.c в функции EcoMain.

Проверяется как взаимозаменяемость компонентов, так и работа функций сложения, вычитания, умножения и деления.



```
Create interface...OK
Getting bus interface...OK
Getting memorymanager interface...OK
Getting test interface...OK

Testing interfaces:

Query IEcoCalculatorX from Lab1...OK
Query IEcoCalculatorY from Lab1...OK
Query IEcoCalculatorX from IEcoCalculatorY...OK
Query IEcoCalculatorY from IEcoCalculatorX...OK

Testing operations:

Query interface IEcoCalculatorX via IEcoLab1...OK
-----
Addition test
Input: -1 + 14
Expected: 13

Correct!
-----
Subtraction test
Input: -1 - 14
Expected: -15

Correct!

Query interface IEcoCalculatorY via IEcoLab1...OK
-----
Multiplication test
Input: -1 * 14
Expected: -14

Correct!
-----
Division test
Input: -1 / 14
Expected: 0

Correct!
Для продолжения нажмите любую клавишу . . .
```

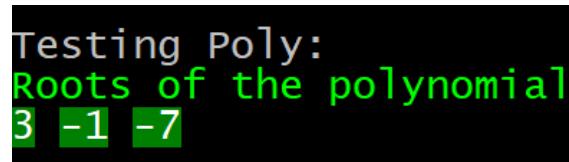
## Визуализация

Для визуализации были имплементированы обратные вызовы, с целью демонстрации работы функции *poly* в консоли.

Для более приятного восприятия использовались дополнительные цвета в консольном выводе:

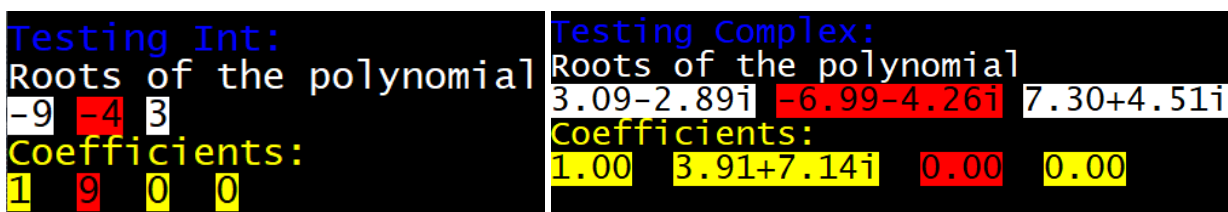
```
void SetColor(HANDLE* handle, const char* color) {
    if (color == "Default") {
        SetConsoleTextAttribute(*handle, FOREGROUND_RED | FOREGROUND_GREEN | FOREGROUND_BLUE);
    }
    if (color == "White") {
        SetConsoleTextAttribute(*handle, FOREGROUND_BLUE | FOREGROUND_RED | FOREGROUND_GREEN | FOREGROUND_INTENSITY);
    }
    if (color == "Red") {
        SetConsoleTextAttribute(*handle, FOREGROUND_INTENSITY | FOREGROUND_RED);
    }
    if (color == "Green") {
        SetConsoleTextAttribute(*handle, FOREGROUND_GREEN | FOREGROUND_INTENSITY);
    }
    if (color == "Blue") {
        SetConsoleTextAttribute(*handle, FOREGROUND_BLUE | FOREGROUND_INTENSITY);
    }
    if (color == "Yellow") {
        SetConsoleTextAttribute(*handle, FOREGROUND_INTENSITY | FOREGROUND_RED | FOREGROUND_GREEN);
    }
    if (color == "YellowBack") {
        SetConsoleTextAttribute(*handle, BACKGROUND_INTENSITY | BACKGROUND_RED | BACKGROUND_GREEN);
    }
    if (color == "GreenBack") {
        SetConsoleTextAttribute(*handle, BACKGROUND_GREEN | FOREGROUND_INTENSITY | FOREGROUND_RED | FOREGROUND_GREEN | FOREGROUND_BLUE);
    }
    if (color == "RedBack") {
        SetConsoleTextAttribute(*handle, BACKGROUND_INTENSITY | BACKGROUND_RED);
    }
    if (color == "WhiteBack") {
        SetConsoleTextAttribute(*handle, BACKGROUND_INTENSITY | BACKGROUND_RED | BACKGROUND_GREEN | BACKGROUND_BLUE);
    }
}
```

Изначально показываются корни



Testing Poly:  
Roots of the polynomial  
3 -1 -7

После чего показывается каждая итерация вычисления функции *poly*, красным подсвечиваются текущие вычисляемые элементы



Testing Int:	Testing Complex:
Roots of the polynomial	Roots of the polynomial
-9 -4 3	3.09-2.89i -6.99-4.26i 7.30+4.51i
Coefficients:	Coefficients:
1 9 0 0	1.00 3.91+7.14i 0.00 0.00

В конце выводится полученный полином



Polynomial:  
1.00\*x^3+(4.51+2.21i)\*x^2+(-11.65+42.98i)\*x-104.17-60.20i

Для такого дизайна было реализовано 3 обратных вызова:

- ShowRoots – показывает корни полинома
- ShowPoly – показывает полином с текущими коэффициентами
- ShowState – показывает текущую итерацию вычисления

```
int16_t (ECOCALLMETHOD *ShowRoots)(struct IEcoLab1Events* me, const void* coefs_data, size_t coefs_count, const char* name_type);  
int16_t (ECOCALLMETHOD *ShowPoly)(struct IEcoLab1Events* me, const void* coefs_data, size_t coefs_count, const char* name_type);  
int16_t (ECOCALLMETHOD *ShowState)(struct IEcoLab1Events* me, const void* buffer_char, const void* coefs_data, size_t coefs_count,  
                                   const char* name_type, size_t buff_ind, size_t coef_ind);
```

Полная демонстрация приведена в файле demo.mp4 в директории с отчетом.