**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ "МИСИС"**

Кафедра инженерной кибернетики

Курсовая работа  
по курсу: «Численные методы»

на тему: «Методы отделения корней алгебраического уравнения»

Выполнила:

Студентка гр.ММ-14-2  
Коротаева В.Д.

Проверил:  
Гопенгауз В.И.

Москва 2016 г.

Оглавление

[Введение 3](#_Toc469660613)

[Границы расположения корней алгебраического уравнения. 4](#_Toc469660614)

[Число действительных корней алгебраического уравнения 7](#_Toc469660615)

[Отделение действительных корней алгебраического уравнения. 11](#_Toc469660616)

[Решение задачи 14](#_Toc469660617)

[Пример работы программы 16](#_Toc469660618)

[Код 16](#_Toc469660619)

[Список литературы 22](#_Toc469660620)

# Введение

Решение уравнений алгебраических представляет собой одну из важнейших задач прикладного анализа, потребность в которой возникает в многочисленных и самых разнообразных разделах физики и механики.

При решении уравнения, прежде всего, важно предварительно изучить расположение корней и заключить каждый корень в достаточно малую область, внутри которой не было бы других корней.

Задача отделения корней уравнения требует знания границы области, в которой расположены все корни уравнения, поэтому сначала необходимо привести ряд способов отыскания этих границ.

В данной работе рассмотрены методы нахождения границ корней уравнения, теорема Штурма для нахождения количества действительных корней и метод Фурье непосредственно для отделения корней.

# Границы расположения корней алгебраического уравнения.

Для алгебраического уравнения

(1)

Пусть *, а .*

**Теорема 1**. Все корни уравнения (1) расположены в кольце (2)

**Доказательство**.

Действительно, , но при имеем:

Следовательно, , как только или , т.е. при . Таким образом, все корни уравнения находятся внутри круга радиуса .

Далее, уравнение (3) имеет корнями величины обратные корням исходного уравнения. По доказанному выше, все корни этого уравнения находятся внутри круга радиуса , т.е. для любого корня исходного уравнения имеет место неравенство или . Объединив результаты, получаем неравенство (2).

Предположим, что все коэффициенты уравнений действительные числа и ­.

Очевидно, достаточно иметь способы определения границ положительных корней, так как, заменяя *х* на *—х*, мы получим уравнение, корни которого отличаются от корней исходного уравнения знаком.

**Теорема 2*.*** Обозначим через *а* максимум абсолютных величин отрицательных коэффициентов уравнения, и пусть первый отрицательный коэффициент в ряду есть . Тогда все положительные корни уравнения меньше . (Если отрицательных коэффициентов нет, то нет и положительных корней)

**Доказательство**. Заменим положительные коэффициенты нулями, а все остальные коэффициенты на *—а*. Тогда при *х > 1* будем иметь:

Отсюда при имеем неравенство , так как , а это и означает, что все положительные корни меньше .

С помощью последней теоремы (т. 2) можно найти границы действительных корней очень грубо. Иногда эти границы можно сузить, применив следующий прием.

Пусть в уравнении коэффициенты неотрицательны, а неположительные и . Введем обозначения:

Тогда

Первое слагаемое содержит только положительные степени *x*, а второе только отрицательные. Следовательно, при первое слагаемое возрастает, а второе убывает с возрастанием *x*, т.е. при функция возрастает вместе с *x.* Найдя какое-либо , для которого , можно гарантировать, что все корни уравнения меньше .

В общем случае представим в виде , где есть многочлен, содержащий все первые старшие степени члены многочлена , имеющие положительные коэффициенты и все члены с отрицательными коэффициентами, а – многочлен, образованный всеми остальными членами исходного многочлена . Тогда, если найти либо , для которого , то при всех , так как при и все корни уравнения будут меньше .

Хороший способ отыскания верхней границы положительных корней указал Ньютон. *Признак Ньютона* основан на утверждении: если при имеют место неравенства то уравнение не имеет корней, больших .

*Замечание*. Нижняя граница положительных корней может быть найдена из уравнения такими же приемами, так как если есть верхняя граница положительных корней этого уравнения, то будет нижней границей положительных корней исходного уравнения.

# Число действительных корней алгебраического уравнения

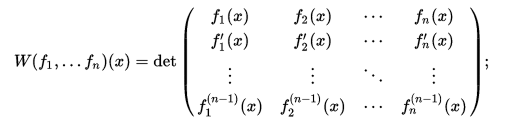
Оценку числа действительных корней алгебраического уравнения можно получить с помощью *правила Декарта*. Получим его как следствие более общей теоремы, которую назовем *обобщенным правилом Декарта*. Эта теорема позволяет находить число действительных корней. Прежде чем ее формулировать, введем некоторые определения.

Пусть дана конечная последовательность действительных чисел . Будем называть индекс *m* местом перемены знака, если и . В этом случае говорят, что  и образуют *перемену знака*.

Очевидны или легко доказываются следующие утверждения:

1. число перемен знака в последовательности не изменится, если члены, равные нулю, будут опущены, а оставшиеся члены сохранят свое расположение
2. число перемен знака в последовательности не изменится, если вставить любое число членов, равных нулю, или рядом с членом последовательности вставить новый член того же знака
3. при вычеркивании членов последовательности число перемен знака не увеличивается
4. если и (j < k) не равны нулю, то в последовательности будет четное или нечетное число перемен знака, в зависимости от того, будут ли и иметь одинаковые или разные знаки
5. пусть *m* — место перемены знака в последовательности . Тогда число перемен знака в этой последовательности на единицу больше числа перемен знака в последовательности .

Пусть последовательность функций, непрерывных вместе со своими производными до порядка *n-1* включительно на отрезке . Рассмотрим определитель Вронского

* (3)*

Имеют место следующие свойства этого определителя:

1. .
2. .

**Теорема (*Обобщенное правило Декарта*).**

Если на отрезке функции непрерывны вместе с производными до порядка (*п—1)* включительно и для любой последовательности на , то число нулей комбинации с действительными коэффициентами (не равными одновременно нулю) на отрезке не превышает числа перемен знака в последовательности .

**Существует несколько методов** для нахождения точного числа корней, причем все они весьма громоздки. Среди них более удобным является метод Штурма.

Воспользуемся введенным ранее определением перемены знака и числом перемен знака. Отметим, что число перемен знаков можно подсчитать, понятно, для любой упорядоченной конечной системы действительных чисел.

Рассмотрим теперь многочлен с действительными коэффициентами, причем будем предполагать, что он не имеет кратных корней, так как иначе мы могли бы его разделить на его наибольший общий делитель с его производной. Конечная упорядоченная система отличных от нуля многочленов с действительными коэффициентами называется системы Штурма для многочлена , если выполняются следующие требования:

1. Соседние многочлены системы Штурма не имеют общих корней.
2. Последний многочлен, , не имеет действительных корней.
3. Если – действительный корень одного из промежуточных многочленов системы, , то и имеют разные знаки.
4. Если – действительный корень многочлена , то произведение меняет знак с минуса на плюс, когда , возрастая, проходит через точку .

Вопрос о том, всякий ли многочлен обладает системой Штурма, будет рассмотрен ниже. Сейчас же, предполагая, что такой системой обладает, покажем, как она может быть использована для нахождения числа действительных корней.

Если действительное число не является корнем данного многочлена , а система Штурма для этого многочлена, то возьмем систему действительных чисел , вычеркнем из нее все числа, равные нулю, и обозначим через число перемен знаков в оставшейся системе. Будем называть числом перемен знаков в системе Штурма многочлена при . Справедлива следующая теорема.

**Теорема Штурма**.

Если действительные числа и , , не являются корнями многочлена , не имеющего кратных корней, то и разность равна числу действительных корней многочлена , заключенных между и .

Таким образом, для определения числа действительных корней многочлена , заключенных между и (по условию не имеет кратных корней), нужно лишь установить, насколько уменьшается число перемен знаков в системе Штурма этого многочлена при переходе от к .

**Доказательство**

Рассмотрим, как меняется число при возрастании . Пока , возрастая, не встретит корня ни одного из многочленов системы Штурма, знаки многочленов этой системы не будут меняться, и поэтому число останется без изменения. Ввиду этого, а также ввиду условия 2) из определения системы Штурма нам остается рассмотреть два случая: переход через корень одного из промежуточных многочленов , и переход через корень самого многочлена .

Пусть – будет корнем многочлена . Тогда, по условию 1), и отличны от нуля. Можно найти, следовательно, такое положительное число , очень малое, что в отрезке многочлены и не имеют корней и поэтому сохраняют постоянные знаки, причем, по условию 3), эти знаки различны. Отсюда следует, что каждая из систем чисел

*(4)*

*и*

*(5)*

обладает ровно одной переменой знаков независимо от того, каковы знаки чисел  и *.* Так, например, если многочлен на рассматриваемом отрезке отрицателен, а положителен и если , то системам (4) и (5) соответствуют системы знаков и .

Таким образом, при переходе через корень одного из промежуточных многочленов системы Штурма перемены знаков в этой системе могут лишь перемещаться, но не возникают вновь и не исчезают, а поэтому число при таком переходе не меняется.

Пусть, с другой стороны, будет корнем самого данного многочлена . По условию 1) не будет корнем для . Существует, следовательно, такое положительное число , что отрезок не содержит корней многочлена , а поэтому сохраняет на этом отрезке сохраняет постоянный знак. Если этот знак положителен, то ввиду условия 4) сам многочлен при переходе через меняет знак с минуса на плюс, т.е. . Системам чисел

и *(6)*

соответствуют системы знаков: и , т.е. в системе Штурма теряется одна перемена. Если же знак на отрезке отрицателен, то снова, ввиду условия 4), многочлен меняет знак с плюса на минус при переходе через *,* т.е. . Системам чисел (6) соответствуют теперь системы знаков и , т.е. в системе Штурма снова теряется одна перемена.

Таким образом, число меняется при возрастании лишь при переходе через корень многочлена , причем в этом случае оно уменьшается ровно на единицу.

Этим доказана, очевидно, теорема Штурма. Для того чтобы воспользоваться ею для разыскания общего числа действительных корней многочлена , достаточно в качестве взять нижний предел отрицательных корней, в качестве – верхний предел положительных корней. Проще, однако, поступить следующим образом. Существует лемма утверждающая, что есть такое положительное число , что знаки всех многочленов системы Штурма будут совпадать со знаками их старших членов. Иными словами, существует столь больше положительное значение неизвестного , что знаки соответствующих ему значений всех многочленов системы Штурма совпадают со знаками их старших коэффициентов. Это значение , вычислять которое нет необходимости, условно обозначается символом . С другой стороны, существует, столь большое по абсолютной величине отрицательное значение , что знаки соответствующих ему значений многочленов системы Штурма совпадают со знаками их старших коэффициентов для многочленов четной степени и противоположны знакам старших коэффициентов для многочленов нечетной степени. Это значение условимся обозначать через . В отрезке содержатся, очевидно, все действительные корни всех многочленов системы Штурма и, в частности, все действительные корни многочлена . Применяя к этому отрезку теорему Штурма, мы найдем число этих корней. Применение же теоремы Штурма к отрезкам и дает соответственно число отрицательных и число положительных корней многочлена .

Остается показать, что всякий многочлен с действительными коэффициентами, не имеющий кратных корней, обладает системой Штурма.

Положим , чем обеспечивается выполнение условия 4) из определения системы Штурма. Действительно, если – действительный корень многочлена , то . Если , то в окрестности , а поэтому меняет знак с минуса на плюс при переходе через . Это же верно тогда и для произведения . Аналогичные рассуждения проходят и в случае Делим затем на и остаток от этого деления, взятый с обратным знаком, принимаем за :

*.*

Вообще, если многочлены и уже найдены, то будет выражаться из:

*. (7)*

Отсюда вытекает, что построенная система нами система многочленов:

удовлетворяет и условию 2) из определения системы Штурма. Для доказательства выполнения условия 1) предположим, что соседние многочлены и обладают общим корнем . Тогда, по (7), будет и корнем многочлена . Переходя к равенству

мы получим, что служит корнем и для . Продолжая далее, мы получим, что служит общим корнем и для и , что противоречит, однако, нашим предположениям. Наконец, выполнение условия 3) вытекает непосредственно из равенства (7): если , то .

# Отделение действительных корней алгебраического уравнения.

Рассмотрим метод Фурье отделения корней, основанный на теореме Бюдана. Но прежде чем излагать сам способ, докажем несколько вспомогательных утверждений.

1. Если уравнение имеет на отрезке корней, а при переходе от к в последовательности теряется перемен знака, то уравнение имеет еще, по крайней мере, комплексных корней.

Действительно, пусть между и имеется действительных корней, а между и действительных корней. Согласно теореме Бюдана, при переходе от к в последовательности теряется перемен знака, а при переходе от к теряется перемен знака, где и — некоторые целые числа. Число перемен знака, теряющихся при переходе от к , равно . Таким образом, , а число комплексных корней Следовательно, .

1. Если функция на отрезке имеет две непрерывные производные и ) на этом отрезке не обращается в нуль, то функция является возрастающей в тех точках, где и имеют одинаковый знак, и убывающей в тех точках, где они имеют разные знаки.

Действительно,  
 , а это означает, что при функция возрастает, а при убывает. Из второго утверждения можно получить два следствия.

***Следствие 1.***

Если на отрезке уравнение имеет два действительных корня, a не обращается в нуль на этом отрезке, то имеет место неравенство

*.*

Действительно, пусть —корни уравнения на отрезке ; имеет на отрезке только один корень. Он расположен между и . Кривая на отрезке или выпукла или вогнута в зависимости от знака , поэтому

и на отрезках и имеют одинаковые знаки. Следовательно, функция на этих отрезках возрастает и Но , а , поэтому и, тем более, , откуда и следует, что *.*

***Следствие 2.***

Если на отрезке уравнение имеет один корень , а функции не обращаются в нуль и на и имеет место неравенство , то его можно нарушить, увеличив или уменьшив на соответствующую величину.

В самом деле, функция возрастает при изменении от до и при изменении от до , а функция будет убывать на от до, а функция будет возрастать от до на . Если и и близки к , то и и, таким образом, , .

Теперь можно изложить способ Фурье отделения действительных корней алгебраического уравнения степени .

Пусть нам дано уравнение степени . Образуем последовательность . На отрезке , на котором могут быть действительные корни, возьмем ряд возрастающих чисел . Если при переходе от к наша последов­ательность не теряет ни одной перемены знака, то уравнение не будет иметь ни одного корня на отрезке . Если теряется только одна перемена знака, то имеется только один корень, и он уже отделен. При потере двух перемен знака могут быть два случая:

1) на имеется два корня уравнения

2) на нет корней.

В последнем случае уравнение имеет обязательно пару комплексных корней. Если потеря перемен знака происходит в первых трех функциях последовательности, то эти два случая можно различить. В самом деле, не имеет корня в . Поэтому, если окажется, что , то по следствию второго утверждения не может иметь двух корней в , т. е. на отрезке нет ни одного корня. Если неравенство не выполняется, то нужно на отрезке выбрать точку и проводить рассуждения с отрезками и .

В общем случае, когда число потерянных перемен знака при переходе от к больше двух или равно двум, но потери происходят не за счет первых трех функций, вопрос может быть решен следующим образом.

Обозначим через число перемен знака, потерянных при переходе от к в последовательности

Общий случай соответствует . В последовательности найдем первое число, равное 1. Пусть это будет . Тогда может иметь значение 0 или 2; первый случай невозможен, так как по условию , и поэтому можно было бы найти с . Если при этом , то всегда можно так уменьшить отрезок , взяв , что корень принадлежит отрезку , а не имеет там корней. Тогда не имеет корней на и и, следовательно, для них равно нулю. Поэтому первое из , равное 1, перемещается для этих отрезков влево. Для отрезка будем иметь . При этом может оказаться, что не будет первым из , равным единице. Тогда мы так же переместимся влево и будем повторять наши рассуждения. Таким образом, нам нужно только рассмотреть случай, когда и — первое из , равное 1.

В этом случае уравнение либо имеет два корня на , либо не имеет ни одного. Эти случаи можно различить так, как мы это делали при . Если имеется два корня, то мы отделяем их изложенным выше способом, при этом отрезок разобьется на два отрезка и , для которых , т. е. первое из , равное 1, переместится влево. Если окажется, что корни комплексные, то можно показать, что все уравнения

имеют пару комплексных корней. Тогда мы уменьшим все при на 2 и получим . Таким образом, мы последовательно подвигаемся влево до тех пор, пока не придем к уже рассмотренным случаям.

Может случиться, что уравнение имеет два равных корня в . Тогда будет справедлив тот же вывод, если только само уравнение не имеет кратных корней. Если же двойной корень принадлежит уравнениям

,

то исходное уравнение в данной точке имеет корень кратности .

# Решение задачи

Пусть дан многочлен

**Найдем границы корней**.

В данном примере . Воспользуемся теоремой (2). Тогда  *, а .*

, и

А значит, все корни уравнения расположены в кольце .

***Верхняя граница положительных корней*:**

Максимум абсолютных величин отрицательных коэффициентов уравнения и первый отрицательный коэффициент в ряду . Тогда все положительные корни уравнения меньше .

***Нижняя граница положительных корней*:**

Нижнюю границу положительных корней найдем из уравнения тем же приемом вычислим, что верхняя граница корней . Так как, если есть верхняя граница положительных корней этого уравнения, то будет нижней границей положительных корней исходного уравнения.

***Границы отрицательных корней*:**

Заменим на *—*, мы получим уравнение , корни которого отличаются от корней исходного уравнения знаком.

Максимум абсолютных величин отрицательных коэффициентов уравнения и первый отрицательный коэффициент в ряду . Тогда все положительные корни уравнения меньше . Нижняя граница отрицательных корней сходного уравнения . Нижнюю границу положительных корней найдем из уравнения тем же приемом вычислим, что верхняя граница корней . Так как, если есть верхняя граница положительных корней этого уравнения, то будет верхней границей отрицательных корней исходного уравнения.

**Отделение корней уравнения**

Для заданного полинома запишем систему функций .

По теореме Штурма мы получили, что это уравнение имеет 2 отрицательных и 2 положительных корня. Поэтому разделим интервалы (0, ) и () на два.

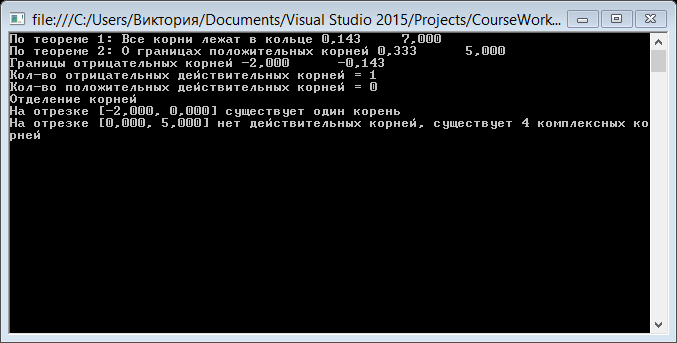
Составим таблицу для подсчета ЧПЗ (число перемен знака):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  | 0 |  |  |
|  | + | - | + | - | + |
|  | - | + | + | - | + |
|  | + | + | - | + | + |
|  | - | - | 0 | + | + |
|  | + | + | + | + | + |
|  | 4 | 3 | 2 | 1 | 0 |

Согласно рассмотренному методу Фурье, тот факт, что разности ЧПЗ, равные 1, достигаются уже на , и доказывает, что на каждом выделенном интервале содержится корень, причем он один.

# Пример работы программы

Вход: многочлен вида



# Код

Нахождение границ

public class FindBorders

{

public static Tuple<double, double> FindUpperAndLowerBorders(double[] arguments)

{

double a\_0 = arguments[0];

double a\_max\_1\_n = Math.Abs(arguments[1]);

double upper\_border, lower\_border;

for (int i = 2; i < arguments.Length; i++)

{

if (Math.Abs(a\_max\_1\_n) < Math.Abs(arguments[i]))

{

a\_max\_1\_n = Math.Abs(arguments[i]);

}

}

double a\_max\_0\_n\_1 = Math.Abs(arguments[0]);

for (int i = 1; i < arguments.Length - 1; i++)

{

if (Math.Abs(a\_max\_0\_n\_1) < Math.Abs(arguments[i]))

{

a\_max\_0\_n\_1 = Math.Abs(arguments[i]);

}

}

upper\_border = 1 + a\_max\_1\_n / Math.Abs(arguments[0]);

lower\_border = Math.Abs(arguments[arguments.Length - 1]) / (a\_max\_0\_n\_1 + Math.Abs(arguments[arguments.Length - 1]));

return new Tuple<double, double>(lower\_border, upper\_border);

}

public static bool FindPositiveUpperBorder(double[] arguments, ref double border)

{

bool q = false;

int k = 0;

double neg\_max\_abs = 0;//макс модуль отрицательного коэффициента

int ind\_first\_neg = 0;//инд первого отрицательного элемента

double[] a = arguments;

if (a[0] < 0)

{

for (int i = 0; i < a.Length; i++)

{

a[i] \*= -1;

}

}

for (int i = 0; i < arguments.Length; i++)

{

if (arguments[i] < 0)

{

q = true;

if (k == 0)

{

ind\_first\_neg = i;

}

k++;

if (neg\_max\_abs < Math.Abs(arguments[i]))

{

neg\_max\_abs = Math.Abs(arguments[i]);

}

}

}

if (k == arguments.Length) { q = false; Debug.WriteLine("Все корни отрицательны"); }

if (q == false)

{

return q;//Корней нет, т.к. нет ни одного отрицательного коэффицента

}

double chisl = neg\_max\_abs / a[0];

double pow = (1.0 / ind\_first\_neg);

border = 1 + Math.Pow(chisl, pow);

return q;

}

public static bool FindPositiveBorders(double[] arguments, ref Tuple<double, double> positiveBorders)

{

bool q;

double upperBorder = 0;

q = FindPositiveUpperBorder(arguments, ref upperBorder);

if (q == false)

{

return q;

}

double[] inverseArguments = new double[arguments.Length];

for (int i = 0; i < arguments.Length; i++)

{

inverseArguments[i] = arguments[arguments.Length - i - 1];

}

double lowerBorder = 0;

FindPositiveUpperBorder(inverseArguments, ref lowerBorder);

lowerBorder = 1 / lowerBorder;

positiveBorders = new Tuple<double, double>(lowerBorder, upperBorder);

return q;

}

public static bool FindNegativeBorders(double[] arguments, ref Tuple<double, double> negativeBorders)

{

double[] a = new double[arguments.Length];

for (int i = 0; i < arguments.Length; i++)

{

a[i] = arguments[i];

}

for (int i = arguments.Length - 2; i >= 0; i = i - 2)

{

a[i] = -arguments[i];

}

if (a[0] < 0)

{

for (int i = 0; i < a.Length; i++)

{

a[i] \*= -1;

}

}

for (int i = 0; i < a.Length; i++)

{

Debug.WriteLine("arg: " + arguments[i] + " \t a: " + a[i]);

}

bool q = FindPositiveBorders(a, ref negativeBorders);

negativeBorders = new Tuple<double, double>(-negativeBorders.Item2, -negativeBorders.Item1);

return q;

}

Количество корней:

public static class RootNumber

{

public static Tuple<int, int> GetRootNumber(double[] dividend)

{

double[] divisor = new double[dividend.Length - 1];

double[,] w = new double[dividend.Length, dividend.Length];

for (int i = 0; i < dividend.Length; i++)

{

w[0, i] = dividend[i];

}

for (int i = 0; i < dividend.Length - 1; i++)

{

w[1, i] = w[0, i + 1] \* (i + 1);

}

int prev\_index = 0;

int curr\_index = 1;

for (int i = 2; i < dividend.Length; i++)

{

double[] prev = new double[dividend.Length - prev\_index];

double[] curr = new double[dividend.Length - curr\_index];

for (int j = 0; j < dividend.Length - prev\_index; j++)

{

prev[j] = w[prev\_index, j];

}

for (int j = 0; j < dividend.Length - curr\_index; j++)

{

curr[j] = w[curr\_index, j];

}

double[] remainder;

remainder = Devision.DeconvWithMinus(prev, curr);

for (int j = 0; j < remainder.Length; j++)

{

w[curr\_index + 1, j] = remainder[j];

}

curr\_index++;

prev\_index++;

}

int cpz\_zero = 0;

for (int i = 1; i < w.GetLength(0); i++)

{

if (Math.Sign(w[i - 1, 0]) \* Math.Sign((w[i, 0])) < 0) cpz\_zero++;

}

int cpz\_minus\_infinity = 0;

for (int i = 1; i < w.GetLength(0); i++)

{

if (Math.Sign((Math.Pow(-1, w.GetLength(0) - i) \* w[i - 1, w.GetLength(0) - i])

\* (Math.Pow(-1, w.GetLength(0) - i - 1) \* w[i, w.GetLength(0) - i - 1])) < 0)

cpz\_minus\_infinity++;

}

int cpz\_plus\_infinity = 0;

for (int i = 1; i < w.GetLength(0); i++)

{

if (Math.Sign(( w[i - 1, w.GetLength(0) - i])

\* w[i, w.GetLength(0) - i - 1]) < 0)

cpz\_plus\_infinity++;

}

System.Diagnostics.Debug.WriteLine("ЧПЗ в НУЛЕ " + cpz\_zero);

System.Diagnostics.Debug.WriteLine("ЧПЗ в –INFINITY " + cpz\_minus\_infinity);

System.Diagnostics.Debug.WriteLine("ЧПз в +INFINITY " + cpz\_plus\_infinity);

return new Tuple<int, int>(cpz\_minus\_infinity - cpz\_zero, cpz\_zero - cpz\_plus\_infinity);

}

}

Деление полиномов для построения ряда Штурма:

public static class Devision

{

public static double[] DeconvWithMinus(double[] dividend, double[] divisor)

{

double[] res = Deconv(dividend, divisor);

for (int i = 0; i < res.Length; i++)

{

res[i] \*= -1;

}

return res;

}

public static double[] Deconv(double[] dividend, double[] divisor)

{

double[] quotient;

double[] res;

double[] remainder = (double[])dividend.Clone();

quotient = new double[remainder.Length - divisor.Length + 1];

for (int i = 0; i < quotient.Length; i++)

{

double coeff = remainder[remainder.Length - i - 1] / divisor.Last();

quotient[quotient.Length - i - 1] = coeff;

for (int j = 0; j < divisor.Length; j++)

{

remainder[remainder.Length - i - j - 1] -= coeff \* divisor[divisor.Length - j - 1];

}

}

int u = dividend.Length - divisor.Length;

if (u > 0)

{

res = new double[divisor.Length-1];

for (int i = 0; i < divisor.Length - 1; i++)

{

res[i] = remainder[i];

}

}

else res = null;

return res;

}

}

Program.cs

static void Main(string[] args)

{

double[] dividend = { 1, 2, -3, 6, -4, 1 };

double lbNeg, ubNeg, lbOp, ubOp;

double[] arrayToFindBorders = new double[dividend.Length];

for (int i = 0; i < dividend.Length; i++)

{

arrayToFindBorders[i] = dividend[dividend.Length - 1 - i];

}

Tuple<double, double> reee = new Tuple<double, double>(0, 0);

reee = FindBorders.FindUpperAndLowerBorders(arrayToFindBorders);

Console.WriteLine("По теореме 1: Все корни лежат в кольце " +

reee.Item1.ToString("0.000") + "\t " + reee.Item2.ToString("0.000"));

System.Diagnostics.Debug.WriteLine(reee.Item1);

System.Diagnostics.Debug.WriteLine(reee.Item2);

FindBorders.FindPositiveBorders(arrayToFindBorders, ref reee);

Console.WriteLine(

"По теореме 2: О границах положительных корней {0} \t {1}",

reee.Item1.ToString("0.000"), reee.Item2.ToString("0.000"));

lbOp = reee.Item1;

ubOp = reee.Item2;

Tuple<double, double> lowerBord = new Tuple<double, double>(0, 0);

FindBorders.FindNegativeBorders(arrayToFindBorders, ref lowerBord);

Console.WriteLine(

"Границы отрицательных корней {0} \t {1}",

lowerBord.Item1.ToString("0.000"), lowerBord.Item2.ToString("0.000"));

lbNeg = lowerBord.Item1;

ubNeg = lowerBord.Item2;

// 2\*x^2+4\*x+3

///2 \* x + 1

var t = RootNumber.GetRootNumber(dividend);

Console.WriteLine("Кол-во отрицательных действительных корней = " + t.Item1 +

"\nКол-во положительных действительных корней = " + t.Item2);

List<double> points = new List<double>();

for (int i = 0; i < t.Item1; i++)

{

points.Add(lbNeg + Math.Abs(lbNeg) \* i / t.Item1);

}

points.Add(0);

if (t.Item2 != 0)

for (int i = t.Item2 - 1; i >= 0; i--)

{

points.Add(ubOp - ubOp \* i / t.Item2);

}

else

{

points.Add(ubOp);

}

//Само отделение

double[,] growth = new double[dividend.Length, dividend.Length];

for (int j = 0; j < dividend.Length; j++)

{

growth[0, j] = dividend[j];

}

for (int i = 1; i < dividend.Length; i++)

{

for (int j = 0; j < dividend.Length - i; j++)

{

growth[i, j] = growth[i - 1, j + 1] \* (j + 1);

}

}

//f - для хранения значения функций в точках

Dictionary<double, double[]> f = new Dictionary<double, double[]>(points.Count);

points.ForEach(x => f.Add(x, new double[dividend.Length]));

points.ForEach(a =>

{

for (int i = 0; i < growth.GetLength(0); i++)

{

double value = growth[i, 0];

for (int j = 1; j < growth.GetLength(0) - i; j++)

{

value += growth[i, j] \* Math.Pow(a, j);

}

f[a][i] = value / factorial(i);

}

}

);

Dictionary<double, int[]> cpz = new Dictionary<double, int[]>(points.Count - 1);

points.ForEach(x => { if (x != points.Last()) cpz.Add(x, new int[dividend.Length]); });

points.ForEach(x =>

{

if (x != points.Last())

{

int chpz\_x, chpz\_y;

double y = points.ToArray()[points.IndexOf(x) + 1];

for (int i = 0; i < dividend.Length; i++)

{

chpz\_x = chpz\_y = 0;

for (int j = i; j < dividend.Length - 1; j++)

{

if (Math.Sign(f[x][j] \* f[x][j + 1]) < 0)

{

chpz\_x++;

}

if (Math.Sign(f[y][j] \* f[y][j + 1]) < 0)

{

chpz\_y++;

}

}

cpz[x][i] = chpz\_x - chpz\_y;

}

}

});

Console.WriteLine("Отделение корней");

for (int m = 0; m < cpz.Count; m++)

{

var item = cpz.ToArray()[m];

int index = 0;

if (item.Value[0] == 1)

{

Console.WriteLine("На отрезке [" + item.Key.ToString("0.000") + ", " +

points.ToArray()[points.IndexOf(item.Key) + 1].ToString("0.000") + "] существует один корень");

}

else

{

int k\_complex = 0;

//index = item.Value.ToList().FindIndex(x => x == 1);

while (true)

{

index = item.Value.ToList().FindIndex(x => x == 1);

double drob\_1 =

f[points.ToArray()[points.IndexOf(item.Key) + 1]][index - 1] /

(/\*index \*\*/ f[points.ToArray()[points.IndexOf(item.Key) + 1]][index]);

double drob\_2 =

f[item.Key][index - 1] / (/\*index \* \*/f[item.Key][index]);

bool q = (drob\_1 - drob\_2) > (points.ToArray()[points.IndexOf(item.Key) + 1] - item.Key);

if (q == true)

{

for (int i = 0; i < item.Value.Length; i++)

{

if (item.Value[i] > 1)

{

item.Value[i] -= 2;

}

else item.Value[i] = 0;

}

}

k\_complex += 2;

index -= 2;

if (index == 0)

{

Console.WriteLine("На отрезке [" + item.Key.ToString("0.000") + ", " +

points.ToArray()[points.IndexOf(item.Key) + 1].ToString("0.000") +

"] нет действительных корней, существует " + k\_complex + " комплексных корней");

break;

}

}

}

}

Console.ReadKey();

}

# Список литературы

1. И.С. Березин, Н.П. Жидков «Методы вычислений. Том 2», М: Государственное издательство физико-математической литературы 1959. — 620 с.
2. А.Г. Курош «Курс Высшей алгебры» М: Издательство Наука 1968. – 431 с.
3. Д. Албахари, Б. Албахари «C# 6.0. Справочник. Полное описание языка» М: Вильямс 2016.– 1040 с.