Math-Net.Ru

Общероссийский математический портал

Я. Е. Ромм, Идентификация области, диапазонов и значений комплексных корней полинома с комплексными коэффициентами на основе устойчивой адресной сортировки, *Итоги науки и техн. Сер. Современ. мат. и ее прил. Темат. обз.*, 2019, том 166, 66–76

DOI: <https://doi.org/10.36535/0233-6723-2019-166-66-76>

Использование Общероссийского математического портала Math-Net.Ru подразумевает, что вы прочитали и согласны с пользовательским соглашением

<http://www.mathnet.ru/rus/agreement>

Параметры загрузки:

IP: 3.85.73.92

26 апреля 2023 г., 08:42:07

ИТОГИ НАУКИ И ТЕХНИКИ.
Современная математика и ее приложения.
Тематические обзоры.
Том 166 (2019). С. 66–76
DOI: 10.36535/0233-6723-2019-166-66-76

УДК 519.61; 519.67

ИДЕНТИФИКАЦИЯ ОБЛАСТИ, ДИАПАЗОНОВ И ЗНАЧЕНИЙ КОМПЛЕКСНЫХ КОРНЕЙ ПОЛИНОМА С КОМПЛЕКСНЫМИ КОЭФФИЦИЕНТАМИ НА ОСНОВЕ УСТОЙЧИВОЙ АДРЕСНОЙ СОРТИРОВКИ

© 2019 г. Я. Е. РОММ

Аннотация. Излагается метод программной идентификации комплексных корней многочленов с комплексными коэффициентами без указания области расположения корней. Метод основан на алгоритме устойчивой адресной сортировки, минимально использует вычисления. Программно определяются числовые диапазоны действительной и мнимой части корней, корни идентифицируются без потери значащих цифр мантиссы в формате представления числовых данных.

Ключевые слова: локализация комплексных корней многочлена, область корней многочлена, вычисления без потери значащих цифр, устойчивая адресная сортировка.

IDENTIFICATION OF THE DOMAIN, RANGES, AND VALUES OF COMPLEX ROOTS OF A POLYNOMIAL WITH COMPLEX COEFFICIENTS BASED ON STABLE ADDRESS SORTING

© 2019 YA. E. ROMM

ABSTRACT. In this paper, we present a method of programmatic identification of complex roots of polynomials with complex coefficients without specifying the domain of localization of their roots. The method is based on the algorithm of stable address sorting with minimal amount of calculations. Numerical ranges of the real and imaginary parts of the roots are programmatically determined, the roots are identified without loss of significant figures of the mantissa in the format of presentation of numerical data.

Keywords and phrases: localization of complex roots of a polynomial, domain of roots of a polynomial, calculations without losing significant figures, stable address sorting.

AMS Subject Classification: 65H04, 65H05

1. Введение и постановка вопроса. Нахождение всех корней многочлена имеет теоретические, прикладные и практические аспекты. На этой основе выполняется анализ устойчивости линейной системы обыкновенных дифференциальных уравнений (см. [5, 7, 8]). Корни характеристических многочленов матриц являются коэффициентами ортогонального базиса при моделировании процессов в полупроводниках (см. [1, 9]), при обработке сигналов в радиоэлектронике (см. [3, 12]). Известны другие разнообразные приложения следствий основной теоремы высшей алгебры. Несмотря на то, что существует много способов вычисления комплексных корней

Работа выполнена при поддержке Российского фонда фундаментальных исследований (проект № 16-07-00100-а).

(см. [4, 21, 22]), сохраняются проблемы локализации, накопления погрешности, вычислительной устойчивости (см. [20, 22]), трудоемкости процесса для многочленов высокой степени. Эти проблемы усугубляются при отсутствии значений границ области корней на комплексной плоскости (см. [2, 6]). Ниже излагается метод программной идентификации комплексных корней многочленов с комплексными коэффициентами высоких степеней без априорных сведений о границах области их расположения. На выходе программы идентифицируются все корни без потери значащих цифр мантиссы в формате представления данных языка программирования. Метод основан на алгоритмах сортировки, в основе излагался в [13, 15, 16], при этом, однако, не содержал программной идентификации области корней, в случае комплексных корней использовался узкий диапазон их расположения и невысокая степень многочлена. В статье показана возможность устранить эти недостатки. Без потери значащих цифр идентифицируются 45 комплексных корней многочлена 45-й степени с комплексными коэффициентами, представленных в практически произвольном числовом диапазоне, среди таких корней имеются плохо отделенные.

2. Идентификация корней многочлена в заданной прямоугольной области. Для удобства изложения вначале используется простейшая сортировка со свойствами устойчивости и явного задания взаимно однозначного соответствия входных и выходных индексов сортируемых элементов. Она является частным случаем сортировки многопутевым слиянием на основе матриц сравнения (см. [17, 18]); в то же время это простая модификация сортировки подсчетом (см. [19]). Пусть, например, требуется упорядочить по нестрогому возрастанию одномерный массив $a = (7, 5, -5, -2, 12)$. Матрица сравнений (МС) имеет вид:

	7	5	-5	-2	12
7	0	-1	-1	-1	+1
5	+1	0	-1	-1	+1
-5	+1	+1	0	+1	+1
-2	+1	+1	-1	0	+1
12	-1	-1	-1	-1	0

Отсортированный массив обозначается c . Номер j -го элемента входного массива a в отсортированном массиве c подсчитывается как число нулей и плюсов в j -м столбце МС над диагональю, включая диагональный элемент, сложенное с числом плюсов под диагональю (нумерация от $j = 1$). Так,

$$a[1] = c[4], \quad a[2] = c[3], \quad a[3] = c[1], \quad a[4] = c[2], \quad a[5] = c[5].$$

Ядро процедуры (в дальнейшем \texttt{sort00}) сводится к циклическим операторам:

```
for j:=1 to n do
begin k:= 0;
  for i:=1 to j do if a[j]>=a[i] then k:=k+1;
  for i:=j+1 to n do if a[j]>a[i] then k:=k+1;
  c[k]:=a[j];
  e[k]:=j;
end;
```

Сортировка устойчива. Операторы $c[k] := a[j]$; $e[k] := j$; в цикле `for j:=1 to n do` каждому входному индексу j сопоставляют единственный выходной индекс k , и обратно: $e[k] := j$; Входные индексы располагаются в порядке отсортированных элементов. Пусть n элементов массива $a = (a_0, a_1, \dots, a_n)$ после сортировки принимают вид $c = (c_0, c_i, \dots, c_n)$, $c_i \leq c_{i+1}$, $0 \leq i \leq n-1$. Соответствие индексов образует подстановку

$$\begin{pmatrix} 1 & 2 & \dots & i & \dots & n \\ e_1 & e_2 & \dots & e_i & \dots & e_n \end{pmatrix},$$

где нижний ряд задается массивом обратных адресов $e = (e_1, e_2, \dots, e_n)$, образуемых операторами $c[k] := a[j]$; $e[k] := j$; . Массив e определяет количество и положение локально экстремальных элементов массива a при произвольно заданном радиусе окрестности локализации ε_0 , где ε_0 —

натуральное число. Именно, локально минимальный элемент с индексом k , где $1 \leq k \leq n$, массива a определяется тем, что должно выполняться условие

$$\neg \exists \ell \in \overline{1, k-1} : |e_k - e_{k-\ell}| \leq \varepsilon_0, \quad 1 \leq k \leq n \quad (1)$$

(см. [18, 19]). Соотношение (1) означает, что ни один из входных индексов отсортированных элементов, предшествующих элементу c_k , не может принадлежать окрестности радиуса ε_0 индекса e_k этого элемента во входном массиве. Поэтому e_k — индекс наименьшего элемента в окрестности радиуса ε_0 . Аналогично, локально максимальный элемент идентифицируется из условия, что неравенство $|e_k - e_{k+\ell}| \leq \varepsilon_0$ не должно выполняться ни при одном ℓ , где $1 \leq \ell \leq n - k$. Программная идентификация минимального элемента имеет вид (ε_0 в программных фрагментах обозначается `eps0`):

```
{sort00(n,a,c,e);} k:=1; while k <= n do
begin for L:=1 to k-1 do if abs(e[k]-e[k-L]) <= eps0 then goto 11;
    ik:=e[k];
    11: k:= k+1
end;
```

В этом фрагменте `ik` означает индекс идентифицированного локально минимального элемента. Минимальность понимается исключительно в смысле отношения порядка. Идентифицированный минимум может оказаться первым (наименьшим), максимум — последним (наибольшим) в цепочке равных элементов. В результате выполнения цикла по k локализуются одновременно все минимальные элементы последовательности в окрестности произвольно фиксированного радиуса ε_0 . Предложенная идентификация исключает накопление погрешности, поскольку не выполняет вычислений: она использует только сравнения элементов при сортировке и сравнение их индексов для локальной минимизации. В результате минимальные элементы идентифицируются с точностью до формата представления данных. Для идентификации локальных минимумов действительной функции $f(x)$ одной действительной переменной x на отрезке длины H входной массив a формируется посредством дискретизации значений функции с равномерным шагом h . Результат получится с точностью до погрешности дискретизации. Согласно численному эксперименту (см. [13]) следует задавать $h \leq \varepsilon_0/33$, ε_0 — произвольно выбранный радиус окрестности локализации искомого минимума; для идентификации всех минимумов выбор его значения обусловлен тем, что оно не должно достигать половины расстояния между ближайшими друг к другу минимумами. Количество элементов сортируемого массива составит $[H/h]$. Целесообразно H выбрать в пределах быстрого выполнения сортировки, а поиск минимумов на всем промежутке выполнять с помощью его разбиения на отрезки длины H . Для проверки минимальности граничных значений каждого отрезка достаточно небольшой набор элементов из их окрестности просортировать с идентификацией минимумов по индексам (см. [19]). Способ распространяется на функции двух и более переменных (см. [14, 15, 19]) для поиска минимумов и, аналогично, максимумов. Ниже будет рассматриваться только случай минимумов в приложении к идентификации корней многочлена. С этой целью в качестве функции рассматривается абсолютная величина многочлена. Как известно, модуль аналитической функции, отличной от константы, не может иметь локальных минимумов внутри области аналитичности, минимумы модуля могут достигаться в тех и только в тех точках, где эта функция обращается в нуль (см. [10, 11]). Поэтому минимумами модуля многочлена являются его корни и только они. Для идентификации действительных корней многочлена с действительными коэффициентами (локальных минимумов) на произвольном отрезке этот отрезок разбивается на части длины H , на каждом из которых программно локализуются все корни. Для уточнения местоположения и значения каждого корня в отдельности ε_0 -окрестность идентифицированной точки корня покрывается равномерной сеткой, на которой выбирается наименьшее значение модуля многочлена. Точка наименьшего значения окружается окрестностью меньшего радиуса, где процесс повторяется. Спуск к наименьшему значению продолжает воспроизводиться до достижения априори заданной точности, измеряемой диаметром сужаемой окрестности. В рассматриваемом приложении проверить наличие корня на границе отрезка заданного разбиения проще без дополнительной сортировки. Именно, достаточно

проверить, что отношение вычисленного значения модуля многочлена к локализованному до начала спуска значению не превышает, например, 0,001. Согласно эксперименту, если ограничение нарушено, — это не точка корня, иначе — приближение к точке корня. Примеры с применением сортировки приводятся в [19]. Для поиска комплексных корней многочлена с комплексными коэффициентами в способ вносятся следующие изменения. Пусть поиск выполняется в прямоугольной области. Многочлен $P_n(z)$, $z = x + iy$, $i = \sqrt{-1}$, преобразуется в неотрицательную действительную функцию двух действительных переменных посредством умножения на комплексно сопряженное значение. Функция $f(x, y) = P_n(z) \times \overline{P_n(z)}$ поступает на вход алгоритма. Минимумами этой функции на декартовой плоскости могут быть корни многочлена и только они. Сложность состоит в компьютерном приближении локальных минимумов, среди которых необходимо выделить наиболее близкие к точным математическим минимумам. Для простоты вся область покрывается равномерной квадратной сеткой размера $H \times H$. Каждый горизонтальный слой сетки обходится слева направо с шагом h . При обходе выполняется идентификация каждого минимума по текущей переменной при фиксированном значении второй переменной. В качестве элемента сортируемого массива берется минимизированное (наименьшее) по всем элементам сетки при фиксированной текущей ординате значение: $\min_{x, y=\text{const}} f(x, y)$. Как только (в таком дискретном приближении) идентифицирован минимум $x_k = \min_{x, y=\text{const}} f(x, y)$, он фиксируется, и непосредственно при этом приближенном значении абсциссы минимума, в полной аналогии случаю одной переменной, на данной равномерной сетке изложенным способом в дискретном приближении идентифицируется приближение к минимуму по другой переменной: $y_k = \min_{y, x_k=\text{const}} f(x_k, y)$. Найденное y_k фиксируется и аналогично с помощью сортировки ищется новое, более точное приближение x_k . Значения координат идентифицированной точки определяются как $x_k = x_0 + e_{xk}h$, $y_k = y_0 + e_{yk}h$, где e_{xk} , e_{yk} — входные индексы элементов на выходе сортировки. Точка (x_k, y_k) принимается за приближение к точке искомого минимума, которое уточняется путем описанного для случая одной переменной спуска, причем по каждой отдельной переменной и с двукратным повторением. Далее процесс продолжается переходом на шаг x_{k+1} при исходно зафиксированном y . Данные шаги выполняются в каждом отдельном квадрате $H \times H$, из разбиения всей области поиска корней. На границах каждого квадрата выполняется проверка идентифицированного наименьшего значения на локальный минимум по аналогии со случаем функции одной переменной. По окончании прохода по одному горизонтальному слою прямоугольной области выполняется переход к следующему снизу вверх горизонтальному слою, где полностью повторяются действия алгоритма на предшествующем слое. Процесс продолжается до окончания обхода всей прямоугольной области. Все эти действия формализует процедура `korniko` в программе `korkomp`, код программы полностью представлен в [14] (эти имена процедуры и программы используются в дальнейшем изложении).

Пример 1. Пусть действительные и мнимые части корней заданы соответственно массивами

$$a = (0,505, 0,602, 0,701, 0,805, 1,5, 1,6), \quad b = (0,503, 0,603, 0,702, 0,806, 1,5, 1,6).$$

Элементы с одинаковыми индексами i составляют действительную и мнимую часть комплексного корня $z[i]$. Нетрудно видеть, что произведение многочлена с такими корнями на комплексно сопряженное значение задается выражением (см. [16])

$$f(x, y) = P_n(z) \times \overline{P_n(z)} = \prod_{i=1}^n \left((x - a[i])^2 + (y - b[i])^2 \right). \quad (2)$$

В рассматриваемом примере $n = 6$. В области $[-4, 4] \times [-4, 4]$ при $\varepsilon_0 = 0,049/2$ программа `korkomp` на выходе даст следующие значения:

$5,05 \times 10^{-1}$		$7,01 \times 10^{-1}$		$1,5 \times 10^0$	
$5,03 \times 10^{-1}$	$0,0 \times 10^0$	$7,02 \times 10^{-1}$	$0,0 \times 10^0$	$1,5 \times 10^0$	$0,0 \times 10^0$
$6,02 \times 10^{-1}$		$8,05 \times 10^{-1}$		$1,6 \times 10^0$	
$6,03 \times 10^{-1}$	$0,0 \times 10^0$	$8,06 \times 10^{-1}$	$0,0 \times 10^0$	$1,6 \times 10^0$	$0,0 \times 10^0$

В правой колонке — значение многочлена в точке корня, действительную и мнимую часть которого соответственно задают два числа без пропуска строки в соседней левой колонке.

Точность вычисления характерна для излагаемого метода (см. [13, 15, 16]); как отмечалось, она обусловлена всевозможной допустимой заменой вычислительных операций на операции сравнения значений и индексов элементов массивов. Размеры области влияют на время работы программы, но не влияют на правильность и точность идентификации корней. Относительно выбора радиуса окрестности локализации точки минимума (аналогично, максимума) необходимо принять во внимание следующее. В окрестность локализации не должны попасть две или более точек минимумов, иначе условие определит наименьший из них, остальные в такой окрестности игнорируются, и программа может не идентифицировать пропущенные минимумы. Поэтому в случае функции одной действительной переменной ε_0 надлежит выбирать из условия $\varepsilon_0 < r_{\min}/2$, где $r_{\min}$ — минимальное расстояние между точками соседних минимумов. В случае функции двух действительных переменных такое же условие должно выполняться хотя бы для одной из двух переменных, точнее, для расстояния между проекциями точек соседних минимумов на одну из координатных осей. В соответствующих обозначениях должно выполняться либо $\varepsilon_0 < r_{\min}/X/2$, либо $\varepsilon_0 < r_{\min}/Y/2$, либо оба неравенства. В силу приближенности начальной локализации минимума это неравенство необходимо усилить, взяв, например, $\varepsilon_0 < r_{\min}/Y/3$. В частности, в программе для примера 1 выбиралось $\varepsilon_0 := 0,049/1,9$. Дальнейшее уменьшение радиуса локализации ведет лишь к уточнению минимума, но это замедляет выполнение программы. Согласно эксперименту для поиска комплексных корней шаг дискретизации следует выбирать из неравенства $h \leq \varepsilon_0/43$. Для уточнения местоположения и значения каждого корня в отдельности, ε_0 -окрестность идентифицированной точки корня покрывается равномерной двумерной сеткой, на которой выбирается наименьшее значение квадрата модуля многочлена. В силу локализованности корня достаточно рекуррентно сужать окрестность точки наименьшего значения до достижения априори заданной точности, измеряемой диаметром сужаемой окрестности. Следует оговорить параметры сокращения диаметра окрестности спуска к наименьшему значению. Корректным относительно сохранения верных значащих цифр корней было бы малое сокращение в пропорции 1,2 или 1,02 и т. д. Однако это влечет замедление программы. Согласно эксперименту можно взять коэффициенты пропорции mm (в программном обозначении) для сокращения диаметра окрестности корней, в зависимости от поиска диапазона корней, $mm := 8$ и $mm := 4$ — без потери их значащих цифр в формате **extended** представления данных. Для повышения точности спуска к значениям корней с нулевой действительной или мнимой частью целесообразно использовать $mm := 128$.

3. Идентификация границ области всех корней многочлена, выделение числовых диапазонов и вычисление корней. Для идентификации границ области всех комплексных корней многочлена используется та же программа **korkomp** (см. [14]), ядро которой составляет отмеченная выше процедура **korniko**. С целью ускорения программы необходимы следующие изменения. Вместо ранее описанной сортировки **sort00** применяется сортировка слиянием по матрицам сравнений (см. [13, 17]) (в [14] представлена под именем **sort**). Она обладает теми же качествами устойчивости и явного операторного выражения взаимно однозначного соответствия входных и выходных индексов сортируемых элементов (см. [13, 17]), но при этом имеет временную сложность $O(N \log_2 N)$. Сортировка **sort00** также используется, но только для упорядочения последовательности значений действительной и мнимой частей найденных приближений корней. Строится следующая схема идентификации границ расположения корней. Пусть на плоскости OXY выбран заведомо вмещающий искомую область квадрат. Для определенности все приводимые ниже данные и параметры соответствуют начально выбранному квадрату $[-4000, 4000] \times [-4000, 4000]$, ограничивающему 45 комплексных корней многочлена 45-й степени. Если с относительно большим радиусом окрестности локализации минимума пройти весь этот квадрат посредством процедуры **korniko**, то первые в порядке обхода приближенные значения точек минимума функции $f(x, y)$ из (2) окажутся левосторонними границами области корней. Согласно принципу минимума до их идентификации других минимумов при обходе не могло

быть. Аналогично, последние в порядке обхода приближенные значения точек минимума функции $f(x, y)$ окажутся правосторонними границами области расположения всех комплексных корней многочлена. Программа сможет идентифицировать все корни в границах получившегося прямоугольника, однако, только при выборе весьма малого радиуса локализации, что неприемлемо замедлит ее работу. Целесообразно разделить работу программы на части, соответственные числовому диапазону корней. Пусть, например, диапазон включает корни, содержащие действительные или мнимые части, по модулю большие либо равные 100 и меньшие 1000. При сохранении обозначений (2) соответственное условное выражение примет вид

$$(100 \leq \text{abs}(a[i])) \text{ and } (\text{abs}(a[i]) < 1000) \text{ or } (100 \leq \text{abs}(b[i])) \text{ and } (\text{abs}(b[i]) < 1000).$$

Предполагается (первоначально, на момент изложения), что в этом диапазоне либо действительные части корней взаимно отделены не менее чем на 10, либо так отделены их мнимые части. В данных ограничениях с параметрами $\varepsilon := 1,1 \times 10^{-46}$ (граница абсолютной погрешности), $\varepsilon_0 := 10/2,2$, $h := \varepsilon_0/43$; $H := Nh$; процедура **korniko** идентифицирует все корни из диапазона

$$(\text{abs}(a[i]) < 1000) \text{ or } (\text{abs}(b[i]) < 1000), \quad (3)$$

которые взаимно отделены не менее чем на 10. Условно согласованные обозначения для идентификации диапазона: $\varepsilon = \mathbf{eps}$ — априори заданная граница абсолютной погрешности вычисления корней; $\varepsilon_0 = \mathbf{eps0}$ — радиус окрестности локализации каждого корня; h — шаг дискретизации; $H = \mathbf{hh}$ — длина отрезка, являющегося стороной элементарного квадрата $H \times H$ (на такие квадраты априори равномерно разбит внешний квадрат области поиска корней); $N = \mathbf{n00}$ — число сортируемых на каждом отрезке длины H элементов одномерного массива. Принципиально, что все корни из (3) будут идентифицированы в том и только в том случае, если их действительные части взаимно отделены не менее чем на $\varepsilon_0 \cdot 2,2 = 10$, либо так отделены их мнимые части. Для отделенных на меньшую величину корней необходимо будет определить соответственно меньший радиус локализации (что выполняется в других диапазонах). Набор корней на входе программы, соответствующий (3), выбран с целью иллюстрации метода. Наряду с корнями из (3), в качестве минимумов, с необходимостью окажутся идентифицированными приближения корней из более узких диапазонов (если многочлен таковые имеет). В частности, идентифицируются те из них, которые имеют действительные или мнимые части, заключенные по модулю между 0 и 100, — из диапазона

$$(\text{abs}(a[i1]) < 100) \text{ or } (\text{abs}(b[i1]) < 100). \quad (4)$$

Эта идентификация выполнится с необходимостью согласно выбору радиуса и сопоставлению диапазонов (3), (4), если имеющиеся корни из (4) отделены от корней из (3) в действительной или мнимой части не менее чем на 10. Более точно, ранее найденные приближения корней из (4), равно как и корней из (3), взаимно отделяются не менее чем на $2\varepsilon_0 = 2 \cdot 10/2,2$. Для идентификации других корней в границах (4) можно снова применить процедуру **korniko**, предварительно уточнив границы области расположения таких корней. Можно было бы непосредственно взять границы (4), но для искомого подмножества корней это не вполне точно и может неоправданно замедлить работу программы. Границы уточняются с помощью сортировки **sort00** и адресации на ее основе к входным индексам граничных элементов. С этой целью в процедуре **korniko** запоминаются все минимумы, идентифицированные на этапе поиска корней (3), и формируются два массива: массив их действительных частей (**av00**) и массив их мнимых частей (**ak00**). Процедура отдельно сортирует массивы действительных и мнимых частей идентифицированных минимумов (приближений корней). При этом сохраняются одинаковые входные индексы, соответствующие точкам входных минимумов из **av00** и **ak00**. В результате сортировки становится известным наименьшее значение действительной части идентифицированных минимумов **c01[1]** (здесь и ниже сохраняются обозначения программы **korkomp** из [14]), а также его адрес **e01[1]** во входном массиве **av00**. Элементы по этому адресу идентифицируются как **av00[e01[1]]**. Аналогично, становится известным наименьшее значение мнимой части **c011[1]**, его адрес **e011[1]** во входном массиве мнимых частей **ak00**. Чтобы определить наибольшее значение действительных частей, надо взять последний отсортированный элемент **av00[e01[nn00]]** и т. д. Но таким

способом определяются границы уже найденных корней диапазона (3). Чтобы выделить среди них искомые «наименьшие» границы корней из (4), выполняется цикл с постусловием:

```
k:=0;
repeat if k<=nn00-1 then k:=k+1; until (abs(av00[e01[k]])< 1e2);
```

Цикл выполняется по ложности условия с действительными частями корней диапазона (3). Как только проверка условия покажет его истинность, цикл прекратит работу, но условие проверится уже после выполнения очередного шага. Поэтому номер k , значение которого установится на выходе цикла, будет номером первого элемента из диапазона (4). Поскольку проверяются отсортированные элементы, этот номер будет номером элемента с наименьшей действительной частью из (4) среди идентифицированных минимумов диапазона (3). Его точное значение определяется как $av00[e01[k]]$. Аналогично, из цикла

```
j:=0;
repeat if j<=nn00-1 then j:=j+1;until (abs(av00[e01[nn00-j]])< 1e2);
```

определяется номер элемента с наибольшей действительной частью в диапазоне (4) среди минимумов, идентифицированных при поиске корней в диапазоне (3). Его точное значение есть $av00[e01[nn00-j]]$. Чтобы не пропустить корни из диапазона (4) (при этом не искать их во всем промежутке от -100 до 100), берется среднее арифметическое: $(av00[e01[k]]+av00[e01[nn00-j]])/2$. От этого значения выполняется отступ к левой границе диапазона на модуль наибольшего значения действительной части и на значение $otstup$: $-abs(av00[e01[nn00-j]])-otstup$. Это определяет левую границу действительной части корней $x00$. Величина отступа $otstup:=eps0*2.2\{10\}$; обусловлена тем, что охватывает использованный для (3) радиус окрестности точки локального минимума, увеличивая его до предположенного для (3) значения отдаленности. Если бы левее найденной границы находился корень, отдаленный более чем на $otstup$, он мог быть априори идентифицирован при поиске корней в диапазоне (3), но он означал бы более грубую границу в новом диапазоне. В диапазоне (4) потребуются идентифицировать корни с радиусом локализации в десять раз меньшими предыдущего ($eps0 \leq 1$). Для определения правой границы действительной части корней из (4) такой же отступ от среднего взят с положительным знаком:

```
x11:=(av00[e01[k]]+av00[e01[nn00-j]])/2+abs(av00[e01[k]])+otstup;
```

Среднее значение может оказаться смещенным к левой или правой границе рассматриваемой области. Поэтому отступ от него следует брать с двойной величиной $otstup:=2*otstup$. Границы мнимой части в диапазоне (4) определяются с помощью аналогичных циклов для приближений мнимых частей корней:

```
y00:=(ak00[e011[k1]]+ak00[e011[nn00-j1]])/2-abs(ak00[e011[nn00-j1]])-otstup;
y11:=(ak00[e011[k1]]+ak00[e011[nn00-j1]])/2+abs(ak00[e011[k1]])+otstup;
```

Если бы левее или правее найденных границ находился корень, отдаленный более чем на $otstup$, он мог быть идентифицирован ранее, при обработке приближений в диапазоне (3), однако этот корень не требуется в качестве завышенной границы в данном случае. Соответственный радиус локализации приводится непосредственно ниже. С параметрами

```
mm:=4; eps:=1.1e-46; eps0:=1.0/2.9; h:=eps0/43; hh:=n00*h;
```

к полученным границам применяется процедура *korniko*. Вслед за описанной обработкой приближений действительных и мнимых частей корней диапазона (4), аналогичный прием применяется к (4) для идентификации границ в диапазоне

$$(abs(a[i2]) < 10) \text{ or } (abs(b[i2]) < 10). \quad (5)$$

С оговоркой, аналогичной предыдущему, часть корней из диапазона (5) с необходимостью идентифицируется при обработке диапазона (4). Это даст основу для адресной идентификации границ корней в диапазоне (5). В диапазоне (5) потребуются идентифицировать корни с радиусом локализации, меньшим 10^{-1} . Для этого в найденных границах применяется процедура *korniko* с значениями параметров


```
eps:=1.1e-46; eps0:=0.1/2.9; h:=eps0/43;
```

Далее, аналогичным способом, из диапазона (5) выделяются границы действительных и мнимых частей приближений корней в диапазоне

$$(\text{abs}(a[i3]) < 1) \text{ or } (\text{abs}(b[i3]) < 1). \quad (6)$$

В найденных границах из диапазона (6) корни идентифицируются процедурой *korniko* с параметрами

```
eps:=1.1e-46*1.1e-22; eps0:=0.049/1.9; h:=eps0/43; hh:=n00*h;
```

Процесс можно было бы организовать как в сторону уменьшения границ диапазона, так и в сторону более широкого начального диапазона.

4. Восстановление пропущенных корней многочлена и уточнение значений идентифицированных приближений. Представленная идентификация границ области корней, раздельная по числовым диапазонам, не исключает пропуск корня, который отделен менее чем на половину радиуса окрестности локализации минимума в последующем, более узком, вложенном диапазоне, или в предыдущем — более широком. Устанавливается факт потерянных корней просто: достаточно подсчитать их количество, сравнить со степенью многочлена, при несовпадении проверить, нет ли среди них кратных подстановкой в производную от многочлена. Восстановить потерянные корни несложно при некотором дополнении к программе. С этой целью действительные и мнимые части всех корней, идентифицированных во всех диапазонах, априори накапливаются (с сохранением индексов комплексных корней) в массивы *avii* и *akii* соответственно, после чего эти массивы сортируются. Сортировка позволит выполнять правильную адресацию к ранее идентифицированным корням и даст возможность их предварительного вывода, упорядоченного по значениям действительных и мнимых частей. Затем в цикле по номерам всех корней в окрестности каждого из них применяется процедура *korniko* с новым радиусом локализации, который исключает пропуск минимумов в текущем диапазоне. Так, для подмножества корней диапазона (4)

```
( abs ( avii[e01[jj]] ) >= 1e1 ) and ( abs ( avii[e01[jj]] ) < 1e2) or  
( abs ( akii[e01[jj]] ) >= 1e1 ) and ( abs ( akii[e01[jj]] ) < 1e2);
```

применяется отступ влево и вправо от действительной части корня на 1,125

```
x00:=-1.125+avii[e01[jj]]; x11:=1.125+avii[e01[jj]];
```

аналогично устанавливается отступ от мнимых частей этого же диапазона

```
y00:=-1.125+akii[e01[jj]]; y11:=1.125+akii[e01[jj]];
```

В границах этого отступа выполняется процедура *korniko* с параметрами

```
eps:=1.1e-46; eps0:=0.049/1.9; h:=eps0/43; hh:=n00*h; mm:=4;
```

В результате идентифицируются все корни данного диапазона, взаимно отделенные в действительных или мнимых частях не менее чем на $2 \cdot 0.049$. Аналогичные уточнения выполняются для подмножества корней диапазона (5) и диапазона (6). При этом для (6) используются параметры «с запасом» с учетом более малых границ диапазонов:

```
eps:=1.1e-46; eps0:=0.00049/1.9; h:=eps0/43; mm:=4;
```

Наконец дополнительно уточняются корни с нулевой действительной или мнимой частью, поскольку специфика формата числовых данных снижает наглядность идентификации таких корней. К таким корням снова применяется процедура *korniko* с значениями параметров, взятыми с «опережающим запасом точности»:

```
eps:=1.1e-100; eps0:=0.000000001125; h:=eps0/43/1.5; hh:=n00*h; mm:=128;  
x00:=-0.00000001125+avii[e01[jj]]; x11:=0.00000001125+avii[e01[jj]];  
y00:=-0.00000001125+akii[e01[jj]]; y11:=0.00000001125+akii[e01[jj]];
```

В результате все корни с нулевыми действительными либо мнимыми частями будут идентифицированы наглядно и с заданной точностью.

Пример 2. Ниже приводятся массивы действительных и мнимых частей многочлена 45-й степени на входе программы **korkomp** с начальной областью поиска $[-4000, 4000] \times [-4000, 4000]$:

$a = (-4, 4, 0, 0, 0, 0, 0, 0, -777.23, 777.23, -555.077, 555.077, -111, 111, -2.1, -2, -1, 0.102, 0.203, 0.302, 0.401, 0.505, 0.602, 0.701, 0.805, 1.5, 1.6, 2.77, 79.01, 8.09, 19.010203, 20, -77, 55.01, 54.1, 80.01, 0.101, 0.202, 8.109, 7.109, 7.109, 0, -1.000001, 0);$

$b = (4, -4, 2, -2, -5, -5.1, 5, 5.1, 555.077, -555.077, 777.23, -777.23, -111, 111, 2.1, 2, -1, 0.107, 0.203, 0.309, 0.404, 0.503, 0.603, 0.702, 0.806, 1.5, 1.6, -2, -55, -79.01, -4.03, 19.0987, -20, 67, 8.02, -4.11, 0.906, 54.101, 80.202, -4.103, -4.103, -4.203, 0.000001, 0, -1.000001);$

Результат работы программы:

$-7,7723 \times 10^2$ $5,55077 \times 10^2$	$0,0 \times 10^0$	$-1,11 \times 10^2$ $-1,11 \times 10^2$	$0,0 \times 10^0$	$-4,0 \times 10^0$ $4,0 \times 10^0$	$0,0 \times 10^0$
$-5,55077 \times 10^2$ $7,7723 \times 10^2$	$0,0 \times 10^0$	$-7,7 \times 10^1$ $6,7 \times 10^1$	$0,0 \times 10^0$	$-2,1 \times 10^0$ $2,1 \times 10^0$	$0,0 \times 10^0$
$-2,0 \times 10^0$ $2,0 \times 10^0$	$0,0 \times 10^0$	$1,01 \times 10^{-1}$ $5,4101 \times 10^1$	$0,0 \times 10^0$	$2,02 \times 10^{-1}$ $8,0202 \times 10^1$	$0,0 \times 10^0$
$-1,0 \times 10^0$ $-1,0 \times 10^0$	$0,0 \times 10^0$	$1,02 \times 10^{-1}$ $1,07 \times 10^{-1}$	$0,0 \times 10^0$	$2,03 \times 10^{-1}$ $2,03 \times 10^{-1}$	$0,0 \times 10^0$
$3,02 \times 10^{-1}$ $3,09 \times 10^{-1}$	$0,0 \times 10^0$	$5,05 \times 10^{-1}$ $5,03 \times 10^{-1}$	$0,0 \times 10^0$	$7,01 \times 10^{-1}$ $7,02 \times 10^{-1}$	$0,0 \times 10^0$
$4,01 \times 10^{-1}$ $4,04 \times 10^{-1}$	$0,0 \times 10^0$	$6,02 \times 10^{-1}$ $6,03 \times 10^{-1}$	$0,0 \times 10^0$	$8,05 \times 10^{-1}$ $8,06 \times 10^{-1}$	$0,0 \times 10^0$
$1,5 \times 10^0$ $1,5 \times 10^0$	$0,0 \times 10^0$	$2,0 \times 10^0$ $-2,0 \times 10^0$	$0,0 \times 10^0$	$7,109 \times 10^0$ $-4,103 \times 10^0$	$0,0 \times 10^0$
$1,6 \times 10^0$ $1,6 \times 10^0$	$0,0 \times 10^0$	$4,0 \times 10^0$ $-4,0 \times 10^0$	$0,0 \times 10^0$	$7,109 \times 10^0$ $-4,203 \times 10^0$	$0,0 \times 10^0$
$8,09 \times 10^0$ $-4,03 \times 10^0$	$0,0 \times 10^0$	$1,9010203 \times 10^1$ $1,90987 \times 10^1$	$0,0 \times 10^0$	$5,41 \times 10^1$ $-4,11 \times 10^0$	$0,0 \times 10^0$
$8,109 \times 10^0$ $-4,103 \times 10^0$	$0,0 \times 10^0$	$2,0 \times 10^1$ $-2,0 \times 10^1$	$0,0 \times 10^0$	$5,501 \times 10^1$ $8,02 \times 10^0$	$0,0 \times 10^0$
$7,7 \times 10^1$ $-5,5 \times 10^1$	$0,0 \times 10^0$	$8,001 \times 10^1$ $0,906 \times 10^0$	$0,0 \times 10^0$	$5,55077 \times 10^2$ $-7,7723 \times 10^2$	$0,0 \times 10^0$
$7,901 \times 10^1$ $-7,901 \times 10^1$	$0,0 \times 10^0$	$1,11 \times 10^2$ $1,11 \times 10^2$	$0,0 \times 10^0$	$7,7723 \times 10^2$ $-5,55077 \times 10^2$	$0,0 \times 10^0$
$-1,000001 \times 10^0$ $6,90341 \times 10^{-101}$	$6,2 \times 10^{-107}$	$6,7 \times 10^{-101}$ $-5,0 \times 10^0$	$1,42 \times 10^{-92}$	$6,86 \times 10^{-101}$ $-1,000001 \times 10^0$	$1,06 \times 10^{-107}$
$6,7 \times 10^{-101}$ $-5,1 \times 10^0$	$3,17 \times 10^{-92}$	$6,82 \times 10^{-101}$ $-2,0 \times 10^0$	$2,84 \times 10^{-102}$	$6,76 \times 10^{-101}$ $1,0 \times 10^{-6}$	$4,09 \times 10^{-115}$
$7,12 \times 10^{-101}$ $2,0 \times 10^0$	$4,38 \times 10^{-105}$	$6,94 \times 10^{-101}$ $5,0 \times 10^0$	$1,91 \times 10^{-93}$	$6,71 \times 10^{-101}$ $5,1 \times 10^0$	$4,44 \times 10^{-93}$

В [14] описаны варианты универсализации изложенной схемы. Кроме того, отмечены возможности различных способов задания многочлена на входе программы. Достаточно любой идентификации значения многочлена, включая неявно заданное значение, по шагам дискретизации. Там же приводятся примеры с полными кодами программ вычисления всех корней многочленов, действительные или мнимые части которых отделены на 0.0001. Обсуждаются аспекты распараллеливания. Пусть на отрезке длины H содержится N шагов дискретизации h , тогда идентификация с точностью до ε всех корней многочлена произвольной степени выполняется параллельно с временной сложностью $T((C^2 N^4)/(12H^2)) = O(\log_2(\varepsilon^{-1}))$, где Ch — сторона квадрата начального поиска области корней, в скобках левой части — число процессоров (см. [14]).

5. Заключение. Изложена схема идентификации комплексных корней многочленов с комплексными коэффициентами, отличающаяся построением на основе устойчивой сортировки с прямой и обратной адресацией к входным и выходным индексам сортируемых элементов. Вследствие минимального количества вычислительных операций программы не создают накопления погрешности. В результате корни многочленов до 45-й степени включительно идентифицируются в области $[-4000, 4000] \times [-4000, 4000]$ без потери значащих цифр в формате представления данных **extended**. Область поиска может варьироваться в сторону увеличения размера, корни могут быть плохо отделенными; в [14] приведены примеры взаимной отделенности действительной либо мнимой частей корней на 0,0001. Результат допускает улучшение (см. [13]). На основе адресности сортировок программы идентифицируют границы областей расположения корней на комплексной плоскости, а также числовые диапазоны их действительных и мнимых частей. На входе метода эти границы указывать не необходимо. Схемы поиска корней максимально распараллеливаются, без изменения применимы для идентификации нулей функций комплексных переменных в области аналитичности.

СПИСОК ЛИТЕРАТУРЫ

1. Багманов А. Т., Санин А. Л. Структуры волновых пакетов в квантовой яме// Усп. совр. радиоэлектроники. — 2005. — № 12. — С. 25–34.
2. Базвалов Н. С. Численные методы. Т. 1. — М.: Наука, 1973.
3. Бендат Дж., Пирсол А. Прикладной анализ случайных данных. — М.: Мир, 1989.
4. Березин И. С., Жидков Н. Г. Методы вычислений. Т. 2. — М.: Физматгиз, 1962.
5. Бесежерский В. А., Попов Е. П. Теория систем автоматического регулирования. — М.: Наука, 1987.
6. Воеводин В. В. Вычислительные основы линейной алгебры. — М.: Наука, 1977.
7. Гантмахер Ф. Р. Теория матриц. — М.: Наука, 1988.
8. Демидович Б. П. Лекции по математической теории устойчивости. — СПб.: Лань, 2008.
9. Демидович Б. П. Математические основы квантовой механики. — СПб.: Лань, 2006.
10. Лаврентьев М. А., Шабат Б. В. Методы теории функций комплексного переменного. — М.: Лань, 2002.
11. Маркушевич А. И. Краткий курс теории аналитических функций. — М.: Мир, 2006.
12. Пистолькорс А. А., Литвинов О. С. Введение в теорию адаптивных антенн. — М.: Наука, 1991.
13. Ромм Я. Е. Локализация и устойчивое вычисление нулей многочлена на основе сортировки, I// Кибернет. систем. анал. — 2007. — № 1. — С. 165–183.
14. Ромм Я. Е. Локализация области всех комплексных корней многочлена, выделение их диапазонов и вычисление на основе сортировки/ Деп. в ВИНТИ 18.07.2017, № 83-B2017. — Таганрог: Таганрог. ин-т имени А. П. Чехова (филиал), 2017.
15. Ромм Я. Е. Метод вычисления нулей и экстремумов функций на основе сортировки с приложением к поиску и распознаванию. I// Кибернет. систем. анал. — 2001. — № 4. — С. 142–159.
16. Ромм Я. Е. Метод вычисления нулей и экстремумов функций на основе сортировки с приложением к поиску и распознаванию. II// Кибернет. систем. анал. — 2001. — № 5. — С. 81–101.
17. Ромм Я. Е. Параллельная сортировка слиянием по матрицам сравнений. I// Кибернет. систем. анал. — 1994. — № 5. — С. 3–23.
18. Ромм Я. Е. Параллельная сортировка слиянием по матрицам сравнений. II// Кибернет. систем. анал. — 1995. — № 4. — С. 13–37.

19. Ромм Я. Е., Заика И. В. Численная оптимизация на основе алгоритмов сортировки с приложением к дифференциальным и нелинейным уравнениям общего вида// Кибернет. систем. анал. — 2011. — № 2. — С. 165–180.
20. Уилкинсон Д. Х. Алгебраическая проблема собственных значений. — М.: Наука, 1970.
21. Фаддеев Д. К., Фаддеева В. Н. Вычислительные методы линейной алгебры. — М.: Наука, 1963.
22. Форсайт Л. Э., Малькольм М., Моулер К. Машинные методы математических вычислений. — М.: Мир, 1980.

Ромм Яков Евсеевич

Таганрогский институт им. А. П. Чехова (филиал)

Ростовского государственного экономического университета

E-mail: romm@list.ru