Theoretical Computer Science 489–490 (2013) 48–57

Contents lists available at SciVerse ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

On parallel complexity of analytic functions[☆]

Fuxiang Yu[a], Ker-I Ko[b,c,*]

[a] Knight Capital Group, NJ, USA
[b] Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan
[c] Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia

ARTICLE INFO

Article history:
Received 8 February 2013
Accepted 3 April 2013
Communicated by D.-Z. Du

Keywords:
Complexity
Analytic functions
Logarithmic space
NC
Derivatives
Integration
Zero-finding

ABSTRACT

In this paper, we study the parallel complexity of analytic functions. We investigate the complexity of computing the derivatives, integrals, and zeros of *NC* or logarithmic-space computable analytic functions, where *NC* denotes the complexity class of sets acceptable by polynomial-size, polylogarithmic-depth, uniform Boolean circuits. It is shown that the derivatives and integrals of *NC* (or logarithmic-space) computable analytic functions remain *NC* (or, respectively, logarithmic-space) computable. We also study the problem of finding all zeros of an *NC* computable analytic function inside an *NC* computable Jordan curve, and show that, under a uniformity condition on the function values on the Jordan curve, all zeros can be found in *NC*.

1. Introduction

We are interested in the parallel-time complexity of analytic functions defined on the real line $\mathbb{R}$ or the complex plane $\mathbb{C}$, under the Turing machine-based model of computation. The general computational complexity theory of real-valued functions in this computational model has been established by Ko and Friedman [17], who introduced the concept of polynomial-time computable real functions and studied the computational complexity of numerical operations on polynomial-time computable real functions (see also Ko [16]). Hoover [14,15] extended this theory to the study of the parallel-time complexity of real functions. He introduced the notion of *NC* computable real functions as the mathematical equivalence of the class of real functions that are efficiently solvable by parallel machines, where *NC* denotes the complexity class of sets acceptable by polynomial-size, polylogarithmic-depth, uniform Boolean circuits.

While polynomial-time computable real functions are considered feasibly computable functions, numerical operations, such as differentiation and integration, on these functions are often infeasible. On the other hand, these numerical operations become feasible when they are applied only to polynomial-time computable analytic functions. More precisely, Ko and Friedman [18] and Müller [22] showed that, for a polynomial-time computable analytic function, its derivative, integral, and zeros are all polynomial-time computable. In this paper, we extend this study to the class of *NC* computable analytic functions and investigate the parallel complexity of the derivatives, integrals, and zeros of these functions.

It is known, in the study of polynomial-time computable analytic functions, that, for a polynomial-time computable function f that is analytic in a neighborhood containing the origin 0, the sequence $\{f^{(n)}(0)/n!\}$ is polynomial-time uniformly computable (see Ko [16]). We extend this result to *NC* and logarithmic-space computable analytic functions; namely, we

[☆] An extended abstract has appeared in *Proceedings of the Third International Conference on Theory and Applications of Models of Computation*, 2006 (*TAMC* 2006) under the title "On some complexity issues of *NC* analytic functions".
[*] Corresponding author at: Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan.
E-mail addresses: fuxiang@gmail.com (F. Yu), keriko@cs.sunysb.edu (K. Ko).

0304-3975/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.tcs.2013.04.008

show that if f is an NC or logarithmic-space computable analytic function defined on a neighborhood containing the origin 0, then the sequence $\{f^{(n)}(0)/n!\}$ of derivatives is NC or logarithmic-space uniformly computable, respectively.

An immediate consequence of the above result is that the integral $\int f(t)dt$ of an NC or logarithmic-space computable analytic function is also NC or logarithmic-space computable, respectively. In addition, we can also apply this result to show that the zeros of NC computable analytic functions are also NC computable. We consider the problem of finding all zeros of an analytic function inside a Jordan curve (see, e.g., Henrici [13]). This problem has been attempted by various methods, including methods based on bisection algorithms (see, e.g., Yakoubsohn [26] and Meylan et al. [21]), simultaneous iterative methods based on Newton's method (see, e.g., Petkovic et al. [24,25]), and the quadrature methods based on numerical integration (see, e.g., Kravanja et al. [20] and Delves et al. [10]). We show, in this paper, that the quadrature method can be parallelized so that the zeros of an NC computable analytic function inside an NC computable Jordan curve Γ are all NC computable if (i) f is analytic on a simply connected domain that covers Γ, and (ii) there is an absolute constant $c > 0$ such that $|f(\mathbf{z})| > c$ for all $\mathbf{z}$ on or near Γ.

The computational models used in this paper include the oracle Turing machine model of Ko [16] and the Boolean circuit model of Hoover [14]. We give, in Section 2, the definitions and basic properties of these computational models. The results about NC computability of the derivatives and integrals of NC computable analytic functions are presented in Section 3. The NC computability of zeros of NC computable analytic functions is studied in Section 4.

2. Preliminaries

This paper involves notions used in both discrete computation and continuous computation. The basic computational objects in discrete computation are integers and binary strings. We write $\ell(w)$ to denote the length of a binary string w, and reserve the notation $|x|$ to denote the absolute value of a real number x. We write $\langle s, t\rangle$ to denote the pairing function on two strings (or integers), and write $\|S\|$ to denote the number of elements in a (finite) set S.

The basic computational objects in continuous computation are *dyadic rationals* in $\mathbb{D} = \{m/2^n : m \in \mathbb{Z}, n \in \mathbb{N}\}$. For a fixed integer $n \in \mathbb{N}$, we write $\mathbb{D}_n = \{m/2^n : m \in \mathbb{Z}\}$. Each dyadic rational d has infinitely many binary representations, each with a different number of trailing zeros. For each such representation s, we write $\ell(s)$ to denote its length. If the specific representation of a dyadic rational d is understood (often the shortest binary representation), then we write $\ell(d)$ to denote the length of this representation. We also use *dyadic complex numbers* $\mathbf{d} = d_x + id_y$ whose real and imaginary parts d_x and d_y are both dyadic rationals, and we define the length of $\mathbf{d}$ as $\ell(\mathbf{d}) = \max(\ell(d_x), \ell(d_y))$.

For a subset S of $\mathbb{C}$, we write ∂S to denote its boundary. For a point $\mathbf{z} \in \mathbb{C}$ and a set $S \subseteq \mathbb{C}$, we let $\delta(\mathbf{z}, S)$ be the distance between $\mathbf{z}$ and S, i.e., $\delta(\mathbf{z}, S) = \inf\{|\mathbf{z} - \mathbf{z}'| : \mathbf{z}' \in S\}$.

The main complexity classes that are used in this paper include the circuit complexity class NC and the following complexity classes defined by Turing machine complexity (see, e.g., Du and Ko [11]).

P: the class of sets accepted by deterministic polynomial-time Turing machines.
NP: the class of sets accepted by nondeterministic polynomial-time Turing machines.
L the class of sets accepted by deterministic logarithmic-space Turing machines.
NL: the class of sets accepted by nondeterministic logarithmic-space Turing machines.
$\#P$: the class of functions that count the number of accepting paths of nondeterministic polynomial-time machines.
$\#L$: the class of functions that count the number of accepting paths of nondeterministic logarithmic-space machines.[1]

For each $i \geq 0$, the complexity class NC^i consists of languages $A \subseteq \{0, 1\}^*$ for which there exists a family $\{C_n\}$ of Boolean circuits with the following properties (see, e.g., Du and Ko [11]).

(a) There exists a Turing machine M that constructs (the encoding of) each C_n in space $O(\log n)$.
(b) For all $n > 0$, C_n has n input nodes and accepts $A_n = A \cap \{0, 1\}^n$.
(c) There exist a polynomial function p and a constant $k > 0$, such that, for all $n > 0$, $size(C_n) \leq p(n)$ and $depth(C_n) \leq k \log^i n$.

We call $\{C_n\}$ an NC^i circuit family. We let NC be the union of NC^i for all $i \geq 0$.

For each deterministic complexity class of languages, we add the prefix F to denote the corresponding class of functions; for example, FP is the class of polynomial-time computable functions (mapping strings to strings).

The inclusive relations among these complexity classes are

$$NC^1 \subseteq L \subseteq NL \subseteq NC^2 \subseteq NC \subseteq P \subseteq NP$$

and

$$\#L \subseteq FNC^2 \subseteq FP \subseteq \#P.$$

Whether any of these inclusions is proper is unknown. The reader is referred to Du and Ko [11] for general properties of these complexity classes and to Alvarez and Jenner [1] for the properties of $\#L$.

[1] We assume that there is a polynomial-time clock attached to such machines, for otherwise there can be infinitely many accepting paths.

A continuous object such as a real number or a real function contains an infinite amount of information, and we need to use discrete functions to represent it. We say that a function $\phi : \mathbb{N}\to\mathbb{D}$ *binary converges to* (or *represents*) a real number x if (i) for all $n \geq 0$, $\phi(n) \in \mathbb{D}_n$, and (ii) for all $n \geq 0$, $|\phi(n) - x| \leq 2^{-n}$. We call such a function ϕ a *Cauchy function* of x. For a complex number $\mathbf{z} = x + iy$, where $x, y \in \mathbb{R}$, we use two functions $\phi_x, \phi_y : \mathbb{N}\to\mathbb{D}$ that binary converge to x and y, respectively, to represent it. Suppose that $\mathcal{C}$ is a complexity class such as L, NC, or P. We say that a real number x is a $\mathcal{C}$ computable number if there exists a Cauchy function ϕ for x such that ϕ is $\mathcal{C}$ computable.

Our basic computational model for computing a real function is the oracle Turing machine (see Ko [16]). Intuitively, to compute a function $f : [0, 1]\to\mathbb{R}$, a Cauchy function ϕ for the input real number x is given in the form of an oracle function, and the oracle machine asks the oracle function ϕ for some values $\phi(n)$ as the approximation to input x and outputs an approximate value to the real number $f(x)$. In other words, the oracle machine maps a Cauchy function for x to a Cauchy function for $y = f(x)$. Since the oracle machine must halt in a finite number of steps, it cannot determine the exact value of x and, hence, it only produces approximate values for the real number y. Therefore, a function f computable by such an oracle machine must be continuous, and its complexity depends on its modulus of continuity. In the following, we say that $m : \mathbb{N}\to\mathbb{N}$ is a *modulus of continuity* of f if, for $n \in \mathbb{N}$ and $x, y \in [0, 1]$,

$$|x - y| \leq 2^{-m(n)} \Rightarrow |f(x) - f(y)| \leq 2^{-n}.$$

With this notion of modulus of continuity, we can define computable real functions without dealing with the oracle Turing machines.

Definition 2.1. A real function $f : [0, 1]\to\mathbb{R}$ is *computable* if

(a) f has a computable modulus of continuity, and
(b) there exists a computable function $\psi : (\mathbb{D} \cap [0, 1]) \times \mathbb{N}\to\mathbb{D}$ such that, for all $d \in \mathbb{D} \cap [0, 1]$ and all $n \in \mathbb{N}$, $|\psi(d, n) - f(d)| \leq 2^{-n}$.

Definition 2.2. A real function $f : [0, 1]\to\mathbb{R}$ is *polynomial-time computable* if

(a) f has a polynomial modulus of continuity, and
(b) there exists a function $\psi : (\mathbb{D} \cap [0, 1]) \times \mathbb{N}\to\mathbb{D}$ such that (i) for all $d \in \mathbb{D} \cap [0, 1]$ and all $n \in \mathbb{N}$, $|\psi(d, n) - f(d)| \leq 2^{-n}$, and (ii) ψ is polynomial-time computable, where the complexity is measured in terms of $\ell(d) + n$ (i.e., there exist a polynomial p and a Turing machine M such that on input $d \in \mathbb{D} \cap [0, 1]$ and $n \in \mathbb{N}$, M outputs $\psi(d, n)$ in time $p(\ell(d) + n)$).

We call the integer n in $\psi(d, n)$ in the above definition the *(output) precision parameter* for f. We use n instead of $\log n$ for the complexity measure in Definition 2.2, since we actually require the error to be within 2^{-n} instead of within n^{-1}.

The notion of polynomial-time computable real functions can be extended to NC computable real functions as follows.

Definition 2.3. Suppose that $i \geq 0$. A function $f : [0, 1]\to\mathbb{R}$ is said to be *NC^i computable* if

(a) f has a polynomial modulus of continuity, and
(b) there exists an NC^i circuit family $\{C_n\}$ with the following properties:
   (i) the circuit $C_{\langle n,m\rangle}$ has m input gates and n output gates; and
   (ii) for any integers $m, n > 0$ and any $d \in \mathbb{D}_m \cap [0, 1]$, $C_{\langle n,m\rangle}$ outputs, on input d, a dyadic rational number e such that $|e - f(d)| \leq 2^{-n}$.

We say that a real function f is NC computable if it is NC^i computable for some $i \geq 0$.

The above definitions are certainly extendible to functions from $[0, 1]$ or $[0, 1]^2$ to $\mathbb{R}$ or $\mathbb{C}$, and to other complexity classes. We omit the precise definitions. Suppose that $\mathcal{C}$ is a complexity class such as P, L, or NC. We write $\mathcal{C}_{\mathbb{R}}$ to denote the class of all $\mathcal{C}$ computable real numbers, and $\mathcal{C}_{C[0,1]}$ to denote the class of all $\mathcal{C}$ computable real functions defined on $[0, 1]$. Relationships among complexity classes of real functions are closely related to those among complexity classes of discrete languages. For instance, Hoover [15] proved that $NC_{C[0,1]} = P_{C[0,1]}$ if and only if $NC = P$.

Let $S \subseteq \mathbb{C}$ be a bounded domain, that is, a bounded, nonempty, open and connected subset of $\mathbb{C}$. Note that the notions of computability and complexity of a function defined on a compact set cannot be extended directly to that of a function defined on an open set. For example, consider $f(x) = 1/x$ on the open set $(0, 1)$: $f(x)$ does not have a modulus function, because $\lim_{x\to 0} f(x) = \infty$, so $f(x)$ is not computable by Definition 2.1. In other words, when a point $\mathbf{z} \in S$ gets closer to the boundary ∂S of S, it may require more resources (time, space, etc.) to approximate $f(\mathbf{z})$ to a predetermined precision 2^{-n}. Our remedy is to modify the precision parameter; that is, if n is the precision parameter and $\delta(\mathbf{z}, \partial S) \geq 2^{-k}$ for some integer $k \geq 0$, we use $n + k$, instead of n, as the complexity measure. Similar treatments have been used by Chou and Ko [8] and Ko and Yu [19]. The following definition is the NC version of this approach.

Definition 2.4. Let $S \subseteq \mathbb{C}$ be a bounded domain, and ∂S its boundary. Suppose that $i \geq 0$. A complex function $f : S\to\mathbb{C}$ is *NC^i computable* if the following conditions hold.

(a) f has a polynomial modulus. More precisely, there exists a polynomial function $p : \mathbb{N}\to\mathbb{N}$ such that, for all $m, n \in \mathbb{N}$ and all $\mathbf{z}_1, \mathbf{z}_2 \in S$, if $\delta(\mathbf{z}_i, \partial S) \geq 2^{-m}$ for both $i = 1, 2$, and $|\mathbf{z}_1 - \mathbf{z}_2| \leq 2^{-p(m+n)}$, then $|f(\mathbf{z}_1) - f(\mathbf{z}_2)| \leq 2^{-n}$.
(b) There exists an NC^i circuit family $\{C_n\}$ with the following properties:
   (i) the circuit $C_{\langle n,m,k\rangle}$ has $2m$ input gates and $2n$ output gates; and
   (ii) for any integers $n, m, k > 0$ and a dyadic point $\mathbf{d} \in S \cap \mathbb{D}_m^2$, $C_{\langle n,m,k\rangle}$ outputs, on input $\mathbf{d} \in S$, a dyadic point $\mathbf{e}$ such that $|\mathbf{e} - f(\mathbf{d})| \leq 2^{-n}$, provided that $\delta(\mathbf{d}, \partial S) \geq 2^{-k}$.

3. Derivatives and integration of *NC* analytic functions

It is known that the computation of derivatives and integrals for polynomial-time computable real functions is, in general, infeasible, but it becomes feasible if we restrict our attention to polynomial-time computable analytic functions (see Ko [16]). We will prove similar results for *NC* computable real functions.

We first extend the infeasibility results about derivatives of polynomial-time computable real functions to *NC* computable real functions. The proofs of Theorems 3.1–3.3 below are similar to those for polynomial-time computable functions in Ko [16]. We omit them here.

The first result is that an *NC* computable real function is not necessarily differentiable.

Theorem 3.1. *There exists an NC^1 computable function $f : [0, 1] \to \mathbb{R}$ that is nowhere differentiable.*

When the derivatives exist, their complexity is determined by their moduli of continuity. In the worst case, the derivatives may exist but could be incomputable in a dense set of $[0, 1]$. In the following, $C^k[0, 1]$ denotes the set of functions $f : [0, 1] \to \mathbb{R}$ that have continuous kth derivatives $f^{(k)}$, and $C^\infty[0, 1]$ denotes the set of infinitely differentiable functions $f : [0, 1] \to \mathbb{R}$.

Theorem 3.2. *Let $f : [0, 1] \to \mathbb{R}$ be an NC computable function.*

(a) *Suppose that f has a continuous derivative on $[0, 1]$. Then, f' is NC computable on $[0, 1]$ if and only if f' has a polynomial modulus of continuity on $[0, 1]$.*

(b) *If $f \in C^k[0, 1]$ for some $k > 0$, then $f^{(i)}$ is NC computable for $i < k$.*

(c) *If $f \in C^\infty[0, 1]$, then $f^{(i)}$ is NC computable for all $i > 0$.*

Theorem 3.3. *There exists an NC computable function $f : [0, 1] \to \mathbb{R}$ whose derivative f' exists everywhere in $[0, 1]$ but $f'(d)$ is not a computable real number for all $d \in \mathbb{D} \cap [0, 1]$.*

Next, we consider the complexity of integration. As integration is in some sense a summation, it is not surprising that it is related to counting classes. Friedman [12] showed that the complexity of the integral of a polynomial-time computable real function can be characterized by the counting class #*P*. In the following, we show that this result can be extended to *NC* computable real functions.

Theorem 3.4. *Let $\mathcal{C}$ be one of the complexity classes L, NC, or P and $F\mathcal{C}$ be the corresponding class of functions. The following are equivalent:*

(a) *Let $f : [0, 1] \to \mathbb{R}$ be a $\mathcal{C}$ computable function. Then, $h(x) = \int_0^x f(t)dt$ is a $\mathcal{C}$ computable function.*

(b) *Let $f : [0, 1] \to \mathbb{R}$ be a $\mathcal{C}$ computable function in $C^\infty[0, 1]$. Then, $h(x) = \int_0^x f(t)dt$ is a $\mathcal{C}$ computable function.*

(c) *$F\mathcal{C} = \#P$.*

Proof (*Sketch*). For each $\mathcal{C}$ of complexity classes L, NC, and P, we define two complexity classes: $\exists\mathcal{C}$ and $\widetilde{\#}\mathcal{C}$. The class $\exists\mathcal{C}$ consists of languages A for which there exist a language $B \in \mathcal{C}$ and a polynomial function p such that, for all $w \in \{0, 1\}^*$,

$$w \in A \Leftrightarrow (\exists v \in \{0, 1\}^{p(\ell(w))})\langle w, v\rangle \in B.$$

The class $\widetilde{\#}\mathcal{C}$ consists of functions g for which there exist a language $B \in \mathcal{C}$ and a polynomial function p such that, for all $w \in \{0, 1\}^*$,

$$g(w) = \|\{u : \ell(u) = p(\ell(w)) \text{ and } \langle w, u\rangle \in B\}\|.$$

It is obvious that $\exists P = NP$ and $\widetilde{\#}P = \#P$ (see, e.g., Du and Ko [11]). However, we do not know whether $\exists L = NL$ or $\widetilde{\#}L = \#L$. In fact, it is not hard to prove that $\exists L = \exists NC = NP$ and $\widetilde{\#}L = \widetilde{\#}NC = \#P$. For instance, since the evaluation of a 3-CNF (conjunctive normal form) Boolean formula is in NC^1, the famous *NP*-complete problem Sat is in $\exists L$, and the associated counting problem #Sat is in $\widetilde{\#}L$. From this relation, it is most likely that $\widetilde{\#}L \neq \#L$ (recall that $\#L \subseteq FNC^2 \subseteq FP$).

Now, a proof similar to that of Theorem 5.33 in Ko [16] establishes that (a) $\Leftrightarrow$ (b) $\Leftrightarrow$ $F\mathcal{C} = \widetilde{\#}\mathcal{C}$, and the theorem follows from the fact that $\widetilde{\#}L = \widetilde{\#}NC = \#P$. □

In part (c) of Theorem 3.2, we showed that, for a given *NC* computable function $f \in C^\infty[0, 1]$, each derivative $f^{(n)}$ is *NC* computable. However, the sequence $\{f^{(n)}\}$ is not necessarily *NC uniformly* computable (see Bläser [3]). For integration, Theorem 3.4 shows that it has higher complexity #*P* than *FNC*, under the assumption that $\#P \neq FNC$. In the following, we show that, for *NC* computable analytic functions, the sequence $\{f^{(n)}\}$ of derivatives must be uniformly *NC* computable, and the integral $\int f$ must be in *FNC*. We consider complex analytic functions instead of real analytic functions, but the results also hold for the real analytic case.

Let S be a domain. A function $f : S \to \mathbb{C}$ is called an *analytic function* if, for every point $\mathbf{z} \in S$, $f'(\mathbf{z})$ exists, or, equivalently, if, for every point $\mathbf{z} \in S$, there is a power series that converges to f at a neighborhood of $\mathbf{z}$. It is obvious that, if f is analytic, then it is infinitely differentiable. If f is analytic in a domain containing a real interval $[a, b] \subseteq S$ such that, for every $x \in [a, b]$, the coefficients of the power series of f at x are all real numbers, we say that f is *real analytic* on $[a, b]$.

Next, let us define the concept of *uniform computability* (see [16]).

Definition 3.5. (a) A sequence $\{x_n\}$ of real (or complex) numbers is *logarithmic-space uniformly computable* if there exists a Turing machine M that, on input $\langle n, k\rangle$, computes, in space $O(\log(n+k))$, a dyadic number (or, respectively, a dyadic complex number) approximating x_n within an error $\le 2^{-k}$.

(b) A sequence $\{x_n\}$ of real (or complex) numbers is *NC uniformly computable*[2] if there exists an NC^i circuit family $\{C_n\}$ for some $i \ge 0$ such that, for any $n, k > 0$, $C_{\langle n,k\rangle}$ outputs a dyadic number (or, respectively, a dyadic complex number) d such that $|d - x_n| \le 2^{-k}$. We also say that $\{x_n\}$ is *NC^i uniformly computable* if the circuits C_n are NC^i circuits.

Similar to the above definition, the concept of logarithmic-space or NC uniformly computable sequences $\{f_n\}$ of real functions can be defined by adding uniformity requirements to Definitions 2.1–2.4. For example, we say that a sequence $\{f_n\}$ has a uniform polynomial modulus if there exists a polynomial function p such that, for any $n, k > 0$ and any $x_1, x_2 \in [0, 1]$, $|x_1 - x_2| \le 2^{-p(n+k)} \Rightarrow |f_n(x_1) - f_n(x_2)| \le 2^{-k}$.

In the following, we use boldface symbols to indicate complex numbers. In particular, $\mathbf{0}$ denotes the origin of the complex plane.

Theorem 3.6. *Suppose that f is an analytic function defined on a domain S that contains the unit disk, and that $\mathcal{C}$ is either the complexity class L or NC^i for some $i \ge 1$. If f is $\mathcal{C}$ computable, then $\{f^{(k)}(\mathbf{0})/k!\}$ is a $\mathcal{C}$ uniformly computable sequence, with the complexity measured in terms of $n + k + \lceil \log M\rceil$, where n is the size of the sequence, k is the output precision, and M is an upper bound of $|f(\mathbf{z})|$ on the unit disk.*

Proof. We present a proof for the case where $\mathcal{C} = NC^i$ for some fixed $i > 0$ and f is real analytic on $[0, 1]$. The proofs of the other cases where f is a complex analytic function or $\mathcal{C} = L$ are similar.

Let $a_n = f^{(n)}(0)/n!$, $n \in \mathbb{N}$. In the following, we apply the method of Newton interpolation to compute approximate values of $a_1, a_2, \ldots, a_n$, with error $\le 2^{-n}$. (For convenience, we make the output precision equal to the size of the sequence.) Let M be an upper bound of $|f|$ on the closed unit disk. Let $b = 2^{-cn}$ for some constant c such that $bM(n+1) \le 2^{-(n+2)}$ and $(1-b)^{n+2} \ge 1/2$. Also let $x_k = k \cdot 2^{-(c+1)n}$, $0 \le k \le n$. Then, we have $0 = x_0 < x_1 < \cdots < x_n \le b = 2^n x_1$. Now, define

$$a'_k = \frac{\sum_{j=0}^{k} (-1)^j \binom{k}{j} f(x_{k-j})}{k!\, x_1^k}, \quad 1 \le k \le n.$$

It is known that $a'_k = f^{(k)}(\xi_k)/k!$ for some $\xi_k \in [0, x_k] \subseteq [0, b]$ (see, e.g., Burden and Faires [5]),[3] and thus

$$|a'_k - a_k| = \frac{1}{k!} \cdot \left| \int_{[0,\xi_k]} f^{(k+1)}(t)dt \right| \le \frac{b}{k!} \cdot \max_{t\in[0,b]} \left|f^{(k+1)}(t)\right|.$$

By Cauchy's integral formula [13], we have, for $t \in [0, b]$,

$$f^{(k+1)}(t) = \frac{(k+1)!}{2\pi i} \int_{|\mathbf{z}-t|=1-b} \frac{f(\mathbf{z})}{(\mathbf{z}-t)^{k+2}} d\mathbf{z}$$

Therefore,

$$\max_{t\in[0,b]} \left|f^{(k+1)}(t)\right| \le \frac{M(k+1)!}{(1-b)^{k+2}},$$

and

$$|a'_k - a_k| \le \frac{bM(k+1)}{(1-b)^{k+2}} \le \frac{bM(n+1)}{(1-b)^{n+2}} \le 2^{-(n+1)}.$$

Next, we design a circuit $C_{\langle k,n\rangle}$ of four layers to approximate a'_k with error $\le 2^{-(n+1)}$.

(1) The bottom layer of $C_{\langle k,n\rangle}$ contains $2k+2$ circuits. Among them, $k+1$ circuits $D_{1,j}$, for $0 \le j \le k$, compute the integers $(-1)^j\binom{k}{j}$, for $0 \le j \le k$; and another $k+1$ circuits $D_{2,j}$, for $0 \le j \le k$, compute approximate values of $f(x_j)$, $0 \le j \le k$, with error $\le 2^{-(k(c+1)n+n+2k+1)}$.

(2) The second layer from the bottom contains $k+1$ circuits $D_{3,j}$, for $0 \le j \le k$, each multiplying the outputs of $D_{1,j}$ and $D_{2,k-j}$ to get an approximate value of $(-1)^j\binom{k}{j}f(x_{k-j})$. It also contains a circuit D_4 that computes $k!$.

[2] The word "uniformly" here refers to the uniform computation of the sequence $\{x_n\}$, and is not to be confused with the uniform computation of the Boolean circuit family in the definition of "uniform NC".

[3] This is the only place where we need the assumption of the real analyticity of f. Formula $a'_k = f^{(k)}(\xi_k)/k!$ does not hold, in general, for complex analytic functions. However, for the case that f is a complex analytic function, we can define two real analytic functions g and h such that $f(x) = g(x) + ih(x)$ on the x-axis. Then, the derivatives of f at $\mathbf{0}$ can be found from those of g and h.

(3) The third layer from the bottom has two circuits. The first one is an addition circuit D_5 that adds the outputs of circuits $D_{3,j}$, $0 \le j \le k$, to get an approximate value of $\sum_{j=0}^{k}(-1)^j\binom{k}{j}f(x_{k-j})$. The other is a multiplication circuit D_6 that computes, from the output of D_4, $k!x_1^k$ (since $x_1^k = 2^{-k(c+1)n}$, this circuit is actually a shifting circuit).
(4) The top layer is a division circuit D_7 that divides the output of D_5 by the output of D_6. The output of D_7 is the output of $C_{\langle n,k\rangle}$.

We note that, for each $j = 0, 1, \ldots, k$, the output of $D_{2,j}$ is an approximation to $f(x_j)$ with error $\le 2^{-(k(c+1)n+n+2k+1)}$. Since $\binom{k}{j} \le 2^k$, the output of $D_{3,j}$ is an approximation to $(-1)^j\binom{k}{j}f(x_{k-j})$ with error $\le 2^{-(k(c+1)n+n+k+1)}$, and the output of D_5 is an approximation to $\sum_{j=0}^{k}(-1)^j\binom{k}{j}f(x_{k-j})$ with error $\le 2^{-(k(c+1)n+n+1)}$. Finally, $k!x_1^k \ge 2^{-k(c+1)n}$, and so the output of D_7 is an approximate to a'_k with total error $\le 2^{-(n+1)}$, as is required.

We note that the integers involved in the above computation are all of length polynomial in n (for example, $n!$ is represented by a binary string of length $O(n \log n)$). Also, apart from the circuits $D_{2,j}$ that compute $f(x_j)$, all other subcircuits of $C_{\langle n,k\rangle}$ are in NC^1; that is, they are of size polynomial in n, and of depth linear in $\log n$. In particular, the circuit $D_{1,j}$ that computes $(-1)^j\binom{k}{j}$ is in NC^1, since the iterated product of n n-bit integers, as well as the division of two n-bit numbers, is computable in NC^1 (see Beame et al. [2] and Chiu et al. [7]). Thus, the whole circuit is of size polynomial in n and of depth $O(\mathit{depth}(f) + \log n)$, where $\mathit{depth}(f)$ is the depth of the circuit family that computes f. This completes the proof of the theorem. □

Theorem 3.7. *Suppose that f is an analytic function defined on a domain S that contains the unit disk, Γ is a $\mathcal{C}$ computable simple curve inside the unit disk, and $\mathcal{C}$ is either the complexity class L or NC^i for some $i \ge 1$. If f is $\mathcal{C}$ computable, then $\int_\Gamma f(\mathbf{z})d\mathbf{z}$ is a $\mathcal{C}$ computable complex number, with the complexity measured in terms of $k + \lceil \log M \rceil$, where k is the output precision and M is an upper bound of $|f(\mathbf{z})|$ on the unit disk.*

Proof. The proof is similar to that in Ko [16] (pages 208–209). Again, we only present a proof for the case $\mathcal{C} = NC^i$ for $i > 0$.

First, we note that, since Γ lies inside the unit disk, the integral $\int_\Gamma f$ depends only on the two end points of Γ. Assume that Γ is an arc starting at $\mathbf{x}$ and ending at $\mathbf{y}$, where $\mathbf{x}$ and $\mathbf{y}$ both are NC^i computable complex numbers. Now, suppose that f has a power series

$$f(\mathbf{z}) = \sum_{n=0}^{\infty} a_n(\mathbf{z} - \mathbf{z}_0)^n$$

at some point $\mathbf{z}_0$ with radius of convergence $r < 1$ such that both $\mathbf{x}$ and $\mathbf{y}$ lie in the closed disk $N(\mathbf{z}_0, r/2) = \{\mathbf{z} : |\mathbf{z} - \mathbf{z}_0| \le r/2\}$. Then, $\int_\Gamma f = \int_{\Gamma_1} f + \int_{\Gamma_2} f$, where Γ_1 is the line segment from $\mathbf{x}$ to $\mathbf{z}_0$, and Γ_2 is the line segment from $\mathbf{z}_0$ to $\mathbf{y}$. To compute $\int_{\Gamma_2} f$, we note that $|a_n| \le M/r^n$ for all $n \ge 0$, where M is an upper bound of $|f(\mathbf{z})|$ on the set $\{\mathbf{z} : |\mathbf{z} - \mathbf{z}_0| = r\}$. From this bound, we can compute an approximate value of the integral $\int_{\Gamma_2} f$ within error 2^{-k} as follows.

(1) Compute the approximate values of the first $p = k + \lceil \log M \rceil + 1$ coefficients a_n, $0 \le n < p$, each of error $\le 2^{-(k+p+1)}$.
(2) Compute approximate values of the integrals

$$\int_{\Gamma_2} (\mathbf{z} - \mathbf{z}_0)^n d\mathbf{z} = \frac{(\mathbf{y} - \mathbf{z}_0)^{n+1}}{n+1}$$

for $n = 0, 1, \ldots, p-1$, each within error $2^{-(k+p+\lceil \log M \rceil + \lceil \log r \rceil n + 2)}$.
(3) For each $n = 0, 1, \ldots, p-1$, multiply the approximate value of a_n from step (1) and the approximate value of $\int_{\Gamma_2} (\mathbf{z} - \mathbf{z}_0)^n d\mathbf{z}$ from step (2).
(4) Add all values from step (3).

To check the error of the above approximation, we let b_n denote $(\mathbf{y} - \mathbf{z}_0)^{n+1}/(n+1)$, for $0 \le n < p$. Also, let a'_n, $0 \le n < p$, denote the outputs of step (1), and b'_n, $0 \le n < p$, be the outputs of step (2). We first note that $|a_n| \le M/r^n$ and $|b'_n| < 1$, and so

$$\begin{aligned}|a_n b_n - a'_n b'_n| &\le |a_n| \cdot |b_n - b'_n| + |b'_n| \cdot |a_n - a'_n| \\ &\le \frac{M}{r^n} \cdot 2^{-(k+p+\lceil \log M \rceil + \lceil \log r \rceil n + 2)} + 2^{-(k+p+2)} \\ &\le 2^{-(k+p+2)} + 2^{-(k+p+2)} = 2^{-(k+p+1)}.\end{aligned}$$

Next, note that

$$\sum_{n=p}^{\infty} a_n b_n \le \sum_{n=p}^{\infty} \frac{M}{r^n} \cdot \frac{r^{n+1}}{2^{n+1}(n+1)} \le \sum_{n=p}^{\infty} \frac{M}{2^{n+1}} = \frac{M}{2^p} = 2^{-(k+1)}.$$

Therefore, the error of the above approximation for $\int_{\Gamma_2} f$ is at most

$$\sum_{n=p}^{\infty} a_n b_n + \sum_{n=0}^{p-1} |a_n b_n - a'_n b'_n| \le 2^{-(k+1)} + \sum_{n=0}^{p-1} 2^{-(k+p+1)} \le 2^{-k}.$$

For the complexity of this computation, we know, from Theorem 3.6, that step (1) can be done in NC^i. Furthermore, steps (2)–(4) are easily seen to be computable in NC^1. So, the integral $\int_{\Gamma_2} f$ is NC^i computable, and, similarly, $\int_{\Gamma_1} f$ (and hence $\int_{\Gamma} f$) is NC^i computable.

Next, for the general case, we note, from the compactness of the unit disk, that there exist a finite number of open disks $D_i = N(\mathbf{z}_i, r_i/2)$, $1 \le i \le m$, that cover the unit disk such that f has a power series at $\mathbf{z}_i$ with radius of convergence r_i. For any two disks D_i and D_j with a nonempty intersection, select a point $\mathbf{w}_{ij}$. Then, we can find a path from $\mathbf{x}$ to $\mathbf{y}$ in the unit disk,

$$\mathbf{x}, \mathbf{z}_{i_1}, \mathbf{w}_{i_1 i_2}, \mathbf{z}_{i_2}, \mathbf{w}_{i_2 i_3}, \mathbf{z}_{i_3} \ldots, \mathbf{z}_{i_t}, \mathbf{y},$$

where $1 \le i_1, \ldots, i_t \le m$, such that the following hold.

(i) $\mathbf{x} \in D_{i_1}$ and $\mathbf{y} \in D_{i_t}$.
(ii) For each $j = 1, 2, \ldots, t-1$, D_{i_j} and $D_{i_{j+1}}$ have a nonempty intersection.

Now, we can compute the integral of f on each segment of this path as above and then sum them up to get $\int_{\Gamma} f$. Note that the finite covering $\{D_i\}$ of the unit disk (and hence $\{\mathbf{z}_i\}$ and $\{\mathbf{w}_{ij}\}$) depends only on the function f. Therefore, the computation of the subpath from $\mathbf{z}_1$ to $\mathbf{z}_t$ can be precomputed and stored in a table. All we need is to find the disks D_{i_1} and D_{i_t} that contain $\mathbf{x}$ and $\mathbf{y}$, respectively. Since $\mathbf{x}$ and $\mathbf{y}$ are NC^i computable complex numbers, the whole computation of $\int_{\Gamma} f$ can be done in NC^i. □

4. Zeros of an *NC* analytic function

We consider, in this section, the following problem. Given a simply connected domain S that contains a Jordan curve Γ and an NC computable function f that is analytic on S, find all zeros of f inside Γ. To study this problem in our setting, we first assume that the given Jordan curve Γ is NC computable; that is, there exists an NC computable function $g : [0, 1] \to \mathbb{C}$ such that $g([0, 1]) = \Gamma$, g is 1–1 on $[0, 1)$ and $g(0) = g(1)$. In the above, if p is a modulus function of g, we also say that it is a modulus function of Γ. In addition, we assume that the function f has no zeros on Γ, because it is, in general, undecidable whether a zero of f lies on Γ, or equivalently, whether the minimum modulus $\min_{\mathbf{z}\in\Gamma} |f(\mathbf{z})|$ of f on Γ is greater than zero. Intuitively, the smaller $\min_{\mathbf{z}\in\Gamma} |f(\mathbf{z})|$ is, the harder it is to compute the zeros of f inside Γ.

We are going to apply the quadrature method to find all zeros of an analytic function f inside a Jordan curve Γ. This method can be described in the following four steps (cf. Kravanja et al. [20] and Delves et al. [10]).

Step 1. *(Computing the number of zeros.)*
Compute the number of zeros by the principle of the argument (see, e.g., Henrici [13]):

$$n = \frac{1}{2\pi i} \int_{\Gamma} \frac{f'(\mathbf{z})}{f(\mathbf{z})} d\mathbf{z}.$$

Step 2. *(Computing the Newton sums.)*
Let $\mathbf{z}_1, \ldots, \mathbf{z}_n$ be all zeros of f inside Γ. The ith Newton sum s_i, for $1 \le i \le n$, is defined as $s_i = \mathbf{z}_1^i + \cdots + \mathbf{z}_n^i$. We can compute them by

$$s_i = \frac{1}{2\pi i} \int_{\Gamma} \mathbf{z}^i \frac{f'(\mathbf{z})}{f(\mathbf{z})} d\mathbf{z},$$

for $1 \le i \le n$ (see, e.g., Henrici [13]).

Step 3. *(Computing the associated polynomial.)*
Compute the associated polynomial $p_n(\mathbf{z}) = \Pi_{i=1}^{n}(\mathbf{z} - \mathbf{z}_i) = \mathbf{z}^n + \sigma_1 \mathbf{z}^{n-1} + \cdots + \sigma_n$ for zeros of f in Γ. The coefficients $\sigma_1, \ldots, \sigma_n$ can be computed using Newton's Identities (see, e.g., Carpentier and Dos Santos [6]):

$$\begin{aligned} &s_1 + \sigma_1 = 0 \\ &s_2 + s_1\sigma_1 + 2\sigma_2 = 0 \\ &\vdots \\ &s_n + s_{n-1}\sigma_1 + \cdots + s_1\sigma_{n-1} + n\sigma_n = 0. \end{aligned} \tag{4.1}$$

Step 4. *(Solving the associated polynomial.)*
Compute the zeros of the polynomial $p_n(\mathbf{z})$.

To show that the above method can be parallelized in our formal computational model, we need more precise assumptions about f and Γ.

Theorem 4.1. *Let S be a simply connected domain that contains an NC computable Jordan curve Γ. Let f be an analytic function on S. Assume that there exist two constants $n_0, n_1 \in \mathbb{N}$ such that*

(a) *for all $\mathbf{z} \in \Gamma$, $\delta(\mathbf{z}, \partial S) \geq 2^{-n_0}$;*
(b) *$|f(\mathbf{z})| > 2^{-n_1}$ for all $\mathbf{z} \in S_1 = \{\mathbf{z} \in \mathbb{C} : \delta(\mathbf{z}, \Gamma) \leq 2^{-n_0}\}$; and*
(c) *$f(\mathbf{z})$ is NC computable on S_1.*

Then, the problem of finding all zeros of f inside Γ is NC solvable, with the complexity measured in terms of $2^{q(n_0+1)} + n_1 + n + k$, where n is the number of zeros of f inside Γ, k is the precision parameter, and q is the modulus function of Γ.

Proof. We check that each of the four steps above can be implemented in *NC*.

Steps 1 *and* 2. These two steps involve integration of meromorphic functions f'/f and $\mathbf{z}^i(f'(\mathbf{z})/f(\mathbf{z}))$, for $i = 1, 2, \ldots, n$. Note that these functions do not have zeros on S_1, and so we may treat them as *NC* computable analytic functions on S_1 and compute the integrals as in Theorem 3.7.

To be more precise, assume that $g : [0, 1] \to \mathbb{C}$ is an *NC* computable function whose image is Γ. For $j = 0, 1, \ldots, 2^{q(n_0+1)}$, we compute $\mathbf{z}_j$ as an approximation to $g(j2^{-q(n_0+1)})$ with error $\leq 2^{-(n_0+1)}$. Then, we get a piecewise linear closed curve Γ' that is at most 2^{-n_0} away from Γ, with breakpoints $\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_{2^{q(n_0+1)}-1}, \mathbf{z}_{2^{q(n_0+1)}} = \mathbf{z}_0$. Also, since $|f(\mathbf{z})| > 2^{-n_1}$, we know that $f'(\mathbf{z})/f(\mathbf{z})$ and $\mathbf{z}^i(f'(\mathbf{z})/f(\mathbf{z}))$ have upper bound $M2^{n_1}$, for some constant $M > 0$, on $\mathbf{z} \in \Gamma$.

Let Γ'_j, $0 \leq j \leq 2^{q(n_0+1)-1}$, be the line segment connecting $\mathbf{z}_j$ to $\mathbf{z}_{j+1}$. Then, by Theorem 3.7, each integral $\int_{\Gamma'_j}(f'/f)$ or $\int_{\Gamma'_j}\mathbf{z}^i(f'/f)$, with $1 \leq i \leq n$, can be computed in *NC*, with complexity measured in terms of $\log(M2^{n_1}) + k = O(n_1 + k)$. The total value

$$\sum_{k=1}^{2^{q(n_0+1)}} \int_{\Gamma'_k} \mathbf{z}^i(f'/f) = \int_{\Gamma'} \mathbf{z}^i(f'/f) = \int_{\Gamma} \mathbf{z}^i(f'/f)$$

is also *NC* computable, with complexity measured in terms of $2^{q(n_0+1)} + n_1 + k$, since iterated addition is in NC^1 (see, e.g., Beame et al. [2]). It follows that the Newton sums can be computed in *NC*, with complexity measured in terms of $2^{q(n_0+1)} + n_1 + n + k$.

Step 3. This step involves the computation of the inverse of a lower triangular matrix, which is NC^2 computable.

We first rewrite the Eq. (4.1) as follows:

$$\begin{aligned} &\sigma_1 = -s_1 \\ &s_1\sigma_1 + 2\sigma_2 = -s_2 \\ &\vdots \\ &s_{n-1}\sigma_1 + \cdots + s_1\sigma_{n-1} + n\sigma_n = -s_n. \end{aligned} \tag{4.2}$$

That is, coefficients $\sigma_1, \ldots, \sigma_n$ are the unknowns of an equation $AX = -B$, where $A = (a_{ij})_{n\times n}$ is an $n \times n$ lower triangular matrix with entries $a_{ij} = s_{i-j}$ for $1 \leq j < i \leq n$ and $a_{ii} = i$ for $1 \leq i \leq n$, X is the column vector $(\sigma_1, \ldots, \sigma_n)^T$, and B is the column vector $(s_1, \ldots, s_n)^T$. So $X = -A^{-1}B$, where A^{-1} is the inverse matrix of A. It is known that computing the inverse of a triangular matrix and computing the product of two matrices are NC^2 computable (see, e.g., Csanky [9] and Borodin et al. [4]). In the following, we present an error analysis to show that the computation of the inverse matrix A^{-1} is still NC^2 computable when the coefficients of A are given by oracles.

Assume that $|s_i| \leq 2^m$ for some $m > 0$, for $1 \leq i \leq n$ (so that $m = O(n_1 + \lceil \log M \rceil + n)$). To get approximate values for σ_i, for $1 \leq i \leq n$, we need, from Step 2, approximate values $\mathbf{d}_i$ to s_i, for $1 \leq i \leq n$, with $|\mathbf{d}_i - s_i| \leq 2^{-cn-k-1}$, where $c = \lceil \log n \rceil + m$. We can estimate the error caused by using $\mathbf{d}_i$ to replace s_i in solving Eq. (4.2) as follows.

We know that, for any $n \times n$ matrix $D = (d_{ij})_{n\times n}$ with $\det(D) \neq 0$,

$$D^{-1} = \frac{1}{\det(D)}\Big((-1)^{i+j}\det(D_{ij})\Big)_{n\times n},$$

where $\det(D)$ is the determinant of D, and D_{ij} is the submatrix of D with the ith row and jth column of D removed. In addition, note that the determinant of D is the sum of all $(-1)^{p(\pi)}d_{1\pi(1)}\cdots d_{n\pi(n)}$ over all permutations π of $\{1, 2, \ldots, n\}$, where $p(\pi)$ is the minimum number of exchanges to obtain π from the identical permutation.

Let A' denote the matrix A with each s_i replaced by $\mathbf{d}_i$, for $1 \leq i \leq n$. We first observe that $\det(A) = \det(A') = n!$, and, for each pair (i, j) with $1 \leq i, j \leq n$, $|\det(A_{ij})| \leq (n-1)!2^{m(n-1)}$. Next, to estimate $|\det(A_{ij}) - \det(A'_{ij})|$, we note that, for real numbers $a_1, a_2, \ldots, a_n$ and $b_1, b_2, \ldots, b_n$, if $|a_i| \leq M$, $|b_i| \leq M$, and $|a_i - b_i| \leq \epsilon$, for all $1 \leq i \leq n$, then we have

$$\begin{aligned} |a_1a_2\cdots a_n - b_1b_2\cdots b_n| &\leq |a_1\cdots a_{n-2}a_{n-1}a_n - a_1\cdots a_{n-2}a_{n-1}b_n| \\ &\quad + |a_1\cdots a_{n-2}a_{n-1}\,b_n - a_1\cdots a_{n-2}\,b_{n-1}b_n| \\ &\quad + \cdots + |a_1\,b_2\,\cdots\,b_{n-1}\,b_n - b_1\,b_2\,\cdots\,b_{n-1}\,b_n| \\ &\leq n\,M^{n-1}\,\epsilon. \end{aligned}$$

Therefore, we get, for $1 \le i, j \le n$,

$$| \det(A_{ij}) - \det(A'_{ij}| \le \sum_{\pi} \left| \prod_{\substack{1\le p\le n \\ p\ne i}} a_{p,\pi(p)} - \prod_{\substack{1\le p\le n \\ p\ne i}} a'_{p,\pi(p)} \right| \le (n-1)!\,(n-1)\,2^{m(n-1)}\,2^{-cn-k-1} \le 2^{-m-k-1},$$

where π ranges over all one-to-one mappings from $\{1, 2, \dots, n\} - \{i\}$ to $\{1, 2, \dots, n\} - \{j\}$.

Now recall that, for each $1 \le i \le n$,

$$\sigma_i = -\sum_{j=1}^{n} \frac{(-1)^{i+j}\det(A_{ij})s_j}{n!}, \quad \text{and} \quad \sigma'_i = -\sum_{j=1}^{n} \frac{(-1)^{i+j}\det(A'_{ij})\mathbf{d}_j}{n!}. \tag{4.3}$$

So, we get, for each $1 \le i \le n$,

$$\begin{aligned} |\sigma_i - \sigma'_i| &\le \frac{1}{n!}\sum_{j=1}^{n} |\det(A_{ij})s_j - \det(A'_{ij})\mathbf{d}_j| \\ &\le \frac{1}{n!}\sum_{j=1}^{n} \left(|s_j - \mathbf{d}_j| \cdot |\det(A_{ij})| + |\mathbf{d}_j| \cdot |\det(A_{ij}) - \det(A'_{ij})|\right) \\ &\le \frac{n}{n!} \cdot \left(2^{-cn-k-1}(n-1)! \cdot 2^{m(n-1)} + 2^m \cdot 2^{-m-k-1}\right) \\ &\le 2^{-(k+1)} + 2^{-(k+1)} = 2^{-k}. \end{aligned}$$

Step 4. For polynomial functions, it has been shown in Neff [23] that all zeros of a polynomial can be found in NC^3. His model is also based on bit-operation measurement, and his algorithm can be easily adapted to our oracle model. Indeed, Neff [23] has already given the error analysis as follows.

Lemma 4.2 *([23]). Suppose that $P(\mathbf{z}) = \mathbf{z}^n + a_{n-1}\mathbf{z}^{n-1} + \cdots + a_1\mathbf{z} + a_0$ and $Q(\mathbf{z}) = \mathbf{z}^n + b_{n-1}\mathbf{z}^{n-1} + \cdots + b_1\mathbf{z} + b_0$, and that M and k are integers such that, for all $1 \le i \le n-1$, $|a_i| \le M - 1$ and $|a_i - b_i| \le 2^{-\ell}$. Then, there is a one-to-one correspondence between zeros of P and zeros of Q such that, for each pair of corresponding zero $\mathbf{u}$ of P and zero $\mathbf{v}$ of Q, $|\mathbf{u} - \mathbf{v}| < 2^{\log M + \log n + 2 - \ell/n}$.*

Now, from (4.3), we know that, for $1 \le i \le n$,

$$|\sigma_i| \le \frac{1}{n!}\sum_{j=1}^{n} |\det(A_{i,j})| \cdot |s_j| \le \frac{n}{n!} \cdot (n-1)!2^{m(n-1)}2^m = 2^{mn}.$$

Therefore, from the above lemma, if we compute, in Step 3, approximate values σ'_i to σ_i within error $\le 2^{-(mn+\lceil \log n \rceil + k + 2)n}$, then the zeros found by Neff's algorithm are within error 2^{-k} of the zeros of $p_n(\mathbf{z})$.

Let us review the total complexity of the four steps. Suppose that we compute, in Step 2, the Newton sums $s_1, s_2, \dots, s_n$ with errors at most $2^{-(2c+mn+k+2)n-1}$, where $c = m + \lceil \log n \rceil$. Then, using these approximate values in Step 3, we get approximate values of $\sigma_1, \sigma_2, \dots, \sigma_n$ of errors at most $2^{-(c+mn+k+2)n}$. This error bound in turn guarantees that the output of Step 4 approximates the zeros of f inside Γ with errors at most 2^{-k}. This shows that the whole process can be accomplished in NC, with complexity measured in terms of $2^{q(n_0+1)} + n_1 + n + k$ (note that $m = O(n_1 + \lceil \log M \rceil + n)$). $\square$

We remark that, when f and Γ are log-space computable, so are the first two steps of the above quadrature method. However, we do not know whether Step 3 can be done in logarithmic space. Moreover, Neff's NC algorithm of computing all zeros of a polynomial is of depth $\log^3 n$. Thus, it is still open whether the problem is log-space solvable for this case.

In addition, it is important to point out that our result is of only theoretical interests. We proved that the zeros of analytic functions can be found in NC, but the complexity bound we used may be very loose. In particular, the constant $2^{q(n_0+1)}$ and the measure m used in the complexity measure appear quite large for practical implementation of the quadrature method.

References

[1] C. Àlvarez, B. Jenner, A very hard log-space counting class, Theoret. Comput. Sci. 107 (1993) 3–30.
[2] P. Beame, S. Cook, H.J. Hoover, Log depth circuits for division and related problems, SIAM J. Comput. 15 (1986) 994–1003.
[3] M. Bläser, Uniform computational complexity of the derivatives of C^∞-functions, Theoret. Comput. Sci. 284 (2002) 199–206.
[4] A. Borodin, J. von zur Gathen, J. Hopcroft, Fast parallel matrix and GCD computations, Inform. Control 52 (1982) 241–256.
[5] R.L. Burden, J.D. Faires, Numerical Analysis, Brooks/Cole, 2001.
[6] M.P. Carpentier, A.F. Dos Santos, Solution of equations involving analytic functions, J. Comput. Phys. 45 (1982) 210–220.
[7] A. Chiu, G.I. Davida, B.E. Litow, Division in logspace-uniform NC^1, Theor. Inform. Appl. 35 (2001) 259–275.
[8] A. Chou, K. Ko, Computational complexity of two-dimensional regions, SIAM J. Comput. 24 (1995) 923–947.
[9] L. Csanky, Fast parallel matrix inversion algorithms, SIAM J. Comput. 5 (1976) 618–623.
[10] L.M. Delves, J.N. Lyness, A numerical method for locating the zeros of an analytic function, Math. Comp. 21 (1967) 543–560.

[11] D.-Z. Du, K. Ko, Theory of Computational Complexity, John Wiley & Sons, New York, 2000.
[12] H. Friedman, On the computational complexity of maximization and integration, Adv. Math. 53 (1984) 80–98.
[13] P. Henrici, Applied and Computational Complex Analysis, vol. 1, John Wiley & Sons, New York, 1974.
[14] H.J. Hoover, Feasible real functions and arithmetic circuits, SIAM J. Comput 19 (1990) 182–204.
[15] H.J. Hoover, Real functions, contraction mappings, and P-completeness, Inform. Comput. 93 (1991) 333–349.
[16] K. Ko, Complexity Theory of Real Functions, Birhäuser-Boston, Cambridge, MA, 1991.
[17] K. Ko, H. Friedman, Computational complexity of real functions, Theoret. Comput. Sci. 20 (1982) 323–352.
[18] K. Ko, H. Friedman, Computing power series in polynomial time, Adv. Appl. Math. 9 (1988) 40–50.
[19] K. Ko, F. Yu, On the complexity of convex hulls of subsets of the two-dimensional plane, Electron. Notes Theoret. Comput. Sci. 202 (2008) 121–135.
[20] P. Kravanja, M. Van Barel, Computing the Zeros of Analytic Functions, Springer-Verlag, Berlin, 2000.
[21] M.H. Meylan, L. Gross, A parallel algorithm to find the zeros of a complex analytic function, ANZIAM J. 44 (2003) E236–E254.
[22] N. Müller, Proceedings of Uniform Computational Complexity of Taylor Series, ICALP, 1987, pp. 435–444.
[23] C.A. Neff, Specified precision polynomial root isolation is in *NC*, J. Comput. System Sci. 48 (1994) 429–463.
[24] M.S. Petkovic, On initial conditions for the convergence of simultaneous root finding methods, Computing 57 (1996) 163–178.
[25] M.S. Petkovic, C. Carstensen, M. Trajkovác, Weierstrass formula and zero-finding methods, Numer. Math. 69 (1995) 353–372.
[26] J.C. Yakoubsohn, Numerical analysis of a bisection–exclusion method to find zeros of univariate analytic functions, J. Complexity 21 (2005) 652–690.